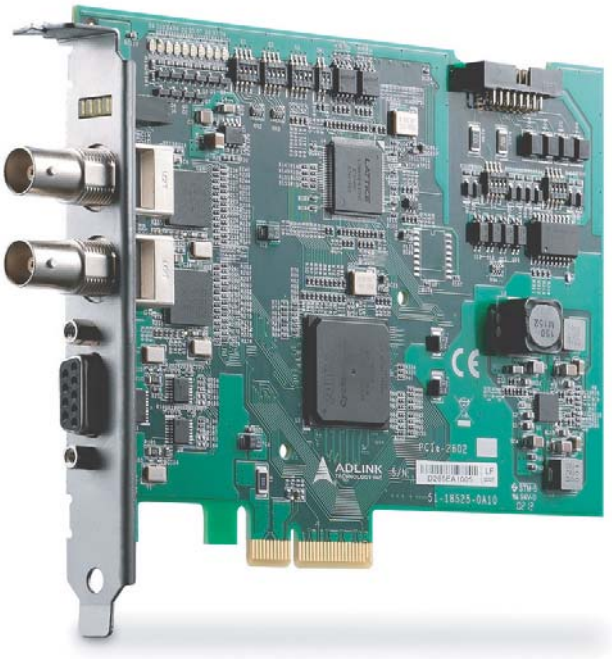
PCIe-2602

3G-SDI Audio/Video Capture Card

Function Library Reference



Manual Rev.: 2.00

Revision Date: Nov. 13, 2013

Part No: 50-11252-1000

Recycled Paper

Advance Technologies; Automate the World.

Revision History

Revision	Release Date	Description of Change(s)
2.00	Nov. 13, 2013	Initial release

Preface

Disclaimer

The information in this document is subject to change without prior notice in order to improve reliability, design, and function and does not represent a commitment on the part of the manufacturer.

In no event will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the product or documentation, even if advised of the possibility of such damages.

Environmental Responsibility

ADLINK is committed to fulfill its social responsibility to global environmental preservation through compliance with the European Union's Restriction of Hazardous Substances (RoHS) directive and Waste Electrical and Electronic Equipment (WEEE) directive. Environmental protection is a top priority for ADLINK. We have enforced measures to ensure that our products, manufacturing processes, components, and raw materials have as little impact on the environment as possible. When products are at their end of life, our customers are encouraged to dispose of them in accordance with the product disposal and/or recovery programs prescribed by their nation or company.

Trademarks

Product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

Conventions

Take note of the following conventions used throughout this manual to make sure that users perform certain tasks and instructions properly.

Additional information, aids, and tips that help users perform tasks.

Information to prevent ***minor*** physical injury, component damage, data loss, and/or program corruption when trying to complete a task.

Information to prevent ***serious*** physical injury, component damage, data loss, and/or program corruption when trying to complete a specific task.

Table of Contents

List of Figures

This page intentionally left blank.

List of Tables

This page intentionally left blank.

1 Introduction

1.1 Overview

The PCIe-2602 3G-SDI Audio/Video Capture Card, based on the PCI Express® x4 interface, enables acquisition of 2 channels 3G-SDI, low latency, and raw video data signals up to 1920x1080P/60fps (frames per second).

Thanks to 3G capability, ADLINK's PCIe-2602 supports high accuracy color format, such as 12 bit 4:4:4 1080i/60fps or 10 bit 4:2:2 1080P/60fps, and when combined with a suitable 75Ω coaxial cable, 3G-SDI signals can be transmitted over 100 m, suiting the PCIe-2602 for medical imaging and intelligent video surveillance and analysis.

The included ViewCreator Pro® utility enables setup, configuration, testing, and system debugging without requiring any software programming. As well, ADLINK's drivers are compatible with Microsoft® DirectShow, reducing engineering efforts and accelerating time to market.

1.2 Features

- Support for 2-CH 3G-SDI video signal, up to 1920 x 1080P/60fps video stream
- Low latency, uncompressed video streaming
- High accuracy color format support, 12 bit 4:4:4 1080i/60fps or 10 bit 4:2:2 1080P/60fps
- Cabling distances up to 100m (w/ compatible 75Ω coaxial cable)
- Directshow support
- RS-485 and Digital I/O provided
- PCI Express x4 compliant signal
- Connection Status LED

1.3 Applications

The PCIe-2602, featuring 3G signal capture ability and support for highly accurate color formatting, is ideal for frame grab function in

a wide variety of applications, including medical imaging and intelligent video surveillance or analytics.

2 Function Library

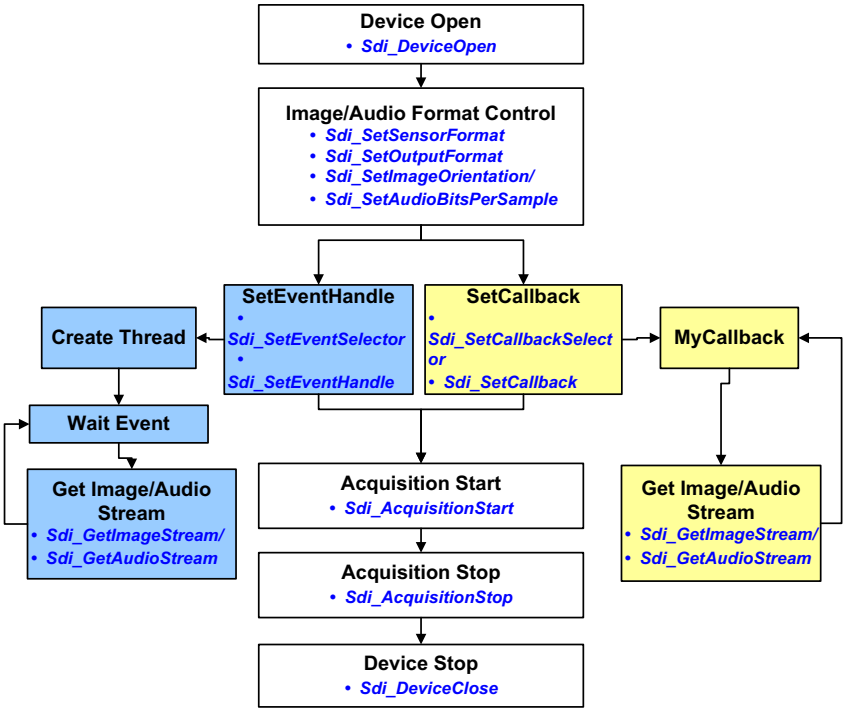
This describes details of the API (Application Programming Interface) for the PCIe-2602, developing application programs within Visual C++, C#, Visual Basic.Net, Delphi, Visual Basic, and Borland C++ Builder.

While the PCIe-2602 API is based on DirectShow technologies, complexity of DirectShow programming has been eliminated in favor of simple API functions requiring no familiarity with DirectShow programming.

Please note that the API and DirectShow programming cannot be combined to access a single PCIe-2602 card at the same time.

2.1 Image Acquisition Processes

An acquisition flowchart follows, showing two alternate capture processes.



2.2 List of Functions

Category	Function
Device Control	Device Count
	Device Open
	Device Close
	Device Vendor Name
	Device Model Name
	Device Version
	Device Firmware Version
	Driver Version
	Library Version
	Device ID
	Channel Number
Image Format	Sensor Format
	Sensor Width
	Sensor Height
	Output Format
	Video Capabilities
	Detected Sensor Format
	Detected Output Format
	Image Orientation
Event & Callback	Event Selector
	Event Handle
	Callback Selector
	Callback

Category	Function
Acquisition Control	Acquisition Frame Count
	Acquisition Start
	Acquisition Stop
	One Shot
	Image Stream
	Acquisition Status
	Acquisition Statistics
	Sensor Status
	Save Image
	Audio Stream
	Dropped Frame Count
	Overflow Frame Count
	Mismatch Frame Count
Digital I/O	Digital I/O Selector
	DI
	DO
Audio Format Control	Audio Bits Per Sample
	Audio Clock Phase Data Bypass
Other	Error Text

Table 2-1: API Functions

2.3 Setting Up the Build Environment

2.3.1 Include Files

All applications using API are required to include the following files.

Include File	
Sdi.h	Header file required for all C/C++ applications
Sdi.vb	Function definitions required for all VB.Net applications
Sdi.cs	Function definitions required for all C# applications

2.3.2 Library Files

All **C/C++** applications using API require the following library files.

Library File	Description
P260x.lib	Exports API function definitions. Required for all Visual C/C++ 32 bits applications.
P260x64.lib	Exports API function definitions. Required for all Visual C/C++ 64 bits applications.
P260x_bcb.lib	Exports API function definitions. Required for all Borland C++ Builder applications.

2.3.3 DLL Files

All applications using API require the following DLL files.

DLL File	Description
P260x.dll	Dynamic link library. Required for all applications.

All files are located in the directory \ADLINK\PCIe-260x\Include

2.4 API Functions

2.4.1 Device Control Functions

Device Count

Returns the total number of supported devices in the system, with a maximum of 32 channels detectable.

Syntax

C/C++

```
int Sdi_GetDeviceCount(UINT &Count)
```

C#

```
int GetDeviceCount(out uint Count)
```

VB.Net

```
GetDeviceCount (ByRef Count as UInteger) As
Integer
```

Parameter(s)

Count

The total number of supported input channels installed.

Return Value

No error occurs if return value >= 0; if negative, refer to Other Functions for return code error information.

Device Open

Initializes a specified SDI channel, and should be called before other functions except those with no Number parameter.

Syntax

C/C++

```
int Sdi_DeviceOpen (UINT Number)
```

C#

```
int DeviceOpen (uint Number)
```

VB.Net

```
DeviceOpen (ByVal Number As UInteger) As Inte-
ger
```

Parameter(s)

Number

The number of SDI channels to be opened, with allowed value from 0 to 31.

Return Value

No error occurs if return value >= 0; if negative, refer to Other Functions for return code error information.

Device Close

Closes the channel and releases all allocated resources; should be called before terminating the application.

Syntax

C/C++

```
int Sdi_DeviceClose (UINT Number)
```

C#

```
int DeviceClose (uint Number)
```

VB.Net

```
DeviceClose (ByVal Number As UInteger) As
Integer
```

Parameter(s)

Number

The number of the SDI channel to be closed with allowed value from 0 to 31.

Return Value

No error occurs if return value >= 0; if negative, refer to Other Functions for return code error information.

Device Vendor Name

Returns the vendor name.

Syntax

C/C++

```
int Sdi_GetDeviceVendorName (char *Name)
```

C#

```
string GetDeviceVendorName ()
```

VB.Net

```
GetDeviceVendorName () As String
```

Parameter(s)

Name

Points to a user-allocated buffer into which the function copies the vendor name string, such as “ADLINK”. The name is NULL-terminated.

Return Value

C/C++

No error occurs if return value >= 0; if negative, refer to Other Functions for return code error information.

C#

Return vendor name.

VB.Net

Return vendor name.

Device Model Name

Returns the model name.

Syntax

C/C++

```
int Sdi_GetDeviceModelName (UINT Number, char
*Name)
```

C#

```
string GetDeviceModelName (uint Number)
```

VB.Net

```
GetDeviceModelName (ByVal Number as UInteger)
As String
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Name

Points to a user-allocated buffer into which the function copies the model name string, for example, “PCIe-2602”. The name is NULL-terminated.

Return Value

C/C++

No error occurs if return value >= 0; if negative, refer to Other Functions for return code error information.

C#

Return model name.

VB.Net

Return model name.

Device Version

Returns the hardware device version.

Syntax

C/C++

```
int Sdi_GetDeviceVersion (UINT Number, char
*Version)
```

C#

```
string GetDeviceVersion (uint Number)
```

VB.Net

```
GetDeviceVersion (ByVal Number as UInteger) As
String
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Version

Points to a user-allocated buffer into which the version string is entered, such as “A1”. The name is NULL-terminated.

Return Value

C/C++

No error occurs if return value >= 0; if negative, refer to Other Functions for return code error information.

C#

Return device version string.

VB.Net

Return device version string.

Device Firmware Version

Returns the firmware version.

Syntax

C/C++

```
int Sdi_GetDeviceFirmwareVersion (UINT Number,
char *Version)
```

C#

```
string GetDeviceFirmwareVersion (uint Number)
```

VB.Net

```
GetDeviceFirmwareVersion (ByVal Number as UIn-
teger) As String
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Version

Points to a user-allocated buffer into which the version string is entered in “Year/Month/Day Hour:Minute” format. The version is NULL-terminated.

Return Value

C/C++

No error occurs if return value >= 0; if negative, refer to Other Functions for return code error information.

C#

Return firmware version string.

VB.Net

Return firmware version string.

Driver Version

Returns the driver version.

Syntax

C/C++

```
int Sdi_GetDriverVersion (UINT Number, char *Version)
```

C#

```
string GetDriverVersion (uint Number)
```

VB.Net

```
GetDriverVersion (ByVal Number as UInteger) As String
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Version

Points to a user-allocated buffer into which the version string is entered in a “n.n.n.n” format. The version is NULL-terminated.

Return Value

C/C++

No error occurs if return value >= 0; if negative, refer to Other Functions for return code error information.

C#

Return driver version string.

VB.Net

Return driver version string.

Library Version

Returns the library version.

Syntax

C/C++

```
int Sdi_GetLibraryVersion (UINT Number, char *Version)
```

C#

```
string GetLibraryVersion (uint Number)
```

VB.Net

```
GetLibraryVersion (ByVal Number as UInteger) As String
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Version

Points to a user-allocated buffer into which the version string is entered in a “n.n.n.n” format. The version is NULL-terminated.

Return Value

C/C++

No error occurs if return value >= 0; if negative, refer to Other Functions for return code error information.

C#

Return library version string.

VB.Net

Return library version string.

Device ID

Acquires device card ID.

Syntax

C/C++

```
int Sdi_GetDeviceID (UINT Number, UINT& ID)
```

C#

```
int GetDeviceID (uint Number, out uint ID)
```

VB.Net

```
GetDeviceID (ByVal Number as UInteger, ByRef
ID As UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

ID

Card ID can be set by DIP switch on the card, with possible values from 0 to 15, and can distinguish individual cards when multiples are installed, with different number settings as shown in the User's Manual.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Channel Number

Retrieves the SDI channel number.

Syntax

C/C++

```
int Sdi_GetChannelNumber (UINT Number, UINT& Channel)
```

C#

```
int GetChannelNumber (uint Number, out uint Channel)
```

VB.Net

```
GetChannelNumber (ByVal Number as UInteger, ByRef Channel As UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Channel

The channel number of this card, with 0 for SDI 0 and 1 for SDI 1.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

2.4.2 Image Format Functions

Sensor Format

Sets or retrieves image format of source input, with format differing according to the input channel.

Syntax

C/C++

```
int Sdi_SetSensorFormat (UINT Number, UINT Format)
int Sdi_GetSensorFormat (UINT Number, UINT &Format)
```

C#

```
int SetSensorFormat (uint Number, uint Format)
int GetSensorFormat (uint Number, out uint Format)
```

VB.Net

```
SetSensorFormat (ByVal Number as UInteger, ByVal Format as UInteger) As Integer
GetSensorFormat (ByVal Number as UInteger, ByRef Format as UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Format

Source input format, with possible values of:

0: 525i 29.97/30 fps (720 x 486 interlace, in fps),

1: 625i 25 fps (720 x 576 interlace, in fps),

2: 720p 24 fps (1280 x 720 progressive),

3: 720p 25 fps (1280 x 720 progressive),

4: 720p 30 fps (1280 x 720 progressive),

5: 720p 50 fps (1280 x 720 progressive),

6: 720p 59.94/60 fps (1280 x 720 progressive),

7: 1080i 25 fps (1920 x 1080 interlace, in fps),

8: 1080i 29.97/30 fps (1920 x 1080 interlace, in fps),

9: 1080p 23.98/24 fps (1920 x 1080 progressive)

10: 1080p 25 fps (1920 x 1080 progressive)

11: 1080p 30 fps (1920 x 1080 progressive)

12: 1080p 50 fps (1920 x 1080 progressive)

13: 1080p 59.94/60 fps (1920 x 1080 progressive)


NOTE:

Resolutions can be acquired with *GetVideoCapabilities()*

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Sensor Width1

Acquires image width of the source input, as shown in Sensor Format.

Syntax

C/C++

```
int Sdi_GetSensorWidth (UINT Number, UINT &Width)
```

C#

```
int GetSensorWidth (uint Number, out uint Width)
```

VB.Net

```
GetSensorWidth (ByVal Number as UInteger, ByRef Width as UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Width

The image width of source input.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Sensor Height

Retrieves image height of the source input, as shown in Sensor Format.

Syntax

C/C++

```
int Sdi_GetSensorHeight (UINT Number, UINT &
Height)
```

C#

```
int GetSensorHeight (uint Number, out uint
Height)
```

VB.Net

```
GetSensorHeight (ByVal Number as UInteger,
ByRef Height as UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Height

The image height of source input.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Output Format

Sets or retrieves pixel output format.

Syntax

C/C++

```
int Sdi_SetOutputFormat (UINT Number, UINT Format)
int Sdi_GetOutputFormat (UINT Number, UINT & Format)
```

C#

```
int SetOutputFormat (uint Number, uint Format)
int GetOutputFormat (uint Number, out uint Format)
```

VB.Net

```
SetOutputFormat (ByVal Number as UInteger, ByVal Format as UInteger) As Integer
GetOutputFormat (ByVal Number as UInteger, ByRef Format as UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Format

Pixel output format, one of (wherein x denotes neutral bit):

0: 16bit YCbCr 4:2:2 (YUY2) – 8bit Y + 8bit Cb/Cr

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Cr	Y	Cb	Y

1: 24bit YCbCr 4:4:4 – 8bit Y + 8bit Cb + 8bit Cr

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Y1	Cr0	Cb0	Y0
dw1	Cb2	Y2	Cr1	Cb1
dw2	Cr3	Cb3	Y3	Cr2

2: 24bit RGB (RGB24) – 8bit R + 8bit G + 8bit B

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	B1	R0	G0	B0
dw1	G2	B2	R1	G1
dw2	R3	G3	B3	R2

3: 20bit YCbCr 4:2:2 – 10bit Y + 10bit Cb/Cr

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Cr0[1:0]Y1[9:4]	Y1[3:0]Cb0[9:6]	Cb0[5:0]Y0[9:8]	Y0[7:0]
dw1	Y3[3:0]Cb2[9:6]	Cb2[5:0]Y2[9:8]	Y2[7:0]	Cr0[9:2]
dw2	Cb4[5:0]Y4[9:8]	Y4[7:0]	Cr2[9:2]	Cr2[1:0]Y3[9:4]

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw3	Y6[7:0]	Cr4[9:2]	Cr4[1:0]Y5[9:4]	Y5[3:0]Cb4[9:6]
dw4	Cr6[9:2]	Cr6[1:0]Y7[9:4]	Y7[3:0]Cb6[9:6]	Cb6[5:0]Y6[9:8]

4: 30bit YCbCr 4:4:4 – 10bit Y + 10bit Cb + 10bit Cr

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	xxCr[9:4]	Cr[3:0]Cb[9:6]	Cb[5:0]Y[9:8]	Y[7:0]

5: 30bit RGB (BGR30) – 8bit R + 8bit G + 8bit B

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	xxB[9:4]	B[3:0]G[9:6]	G[5:0]R[9:8]	R[7:0]

6: 24bit YCbCr 4:2:2 – 12bit Y + 12bit Cb/Cr

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Y1[7:0]	Cb0[11:4]	Cb0[3:0] Y0[11:8]	Y0[7:0]
dw1	Cb2[3:0] Y2[11:8]	Y2[7:0]	Cr0[11:4]	Cr0[3:0] Y1[11:8]
dw2	Cr2[11:4]	Cr2[3:0] Y3[11:8]	Y3[7:0]	Cb2[11:4]

7: 36bit YCbCr 4:4:4 – 12bit Y + 12bit Cb + 12bit Cr

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Cr0[7:0]	Cb0[11:4]	Cb0[3:0] Y0[11:8]	Y0[7:0]
dw1	Cr1[3:0] Cb1[11:8]	Cb1[7:0]	Y1[11:4]	Y1[3:0] Cr0[11:8]
dw2	Cb2[11:4]	Cb2[3:0] Y2[11:8]	Y2[7:0]	Cr1[11:4]
dw3	Cb3[7:0]	Y3[11:4]	Y3[3:0] Cr2[11:8]	Cr2[7:0]
dw4	Cb4[3:0] Y4[11:8]	Y4[7:0]	Cr3[11:4]	Cr3[3:0] Cb3[11:8]
dw5	Y5[11:4]	Y5[3:0] Cr4[11:8]	Cr4[7:0]	Cb4[11:4]
dw6	Y6[7:0]	Cr5[11:4]	Cr5[3:0] Cb5[11:8]	Cb5[7:0]
dw7	Y7[3:0] Cr6[11:8]	Cr6[7:0]	Cb6[11:4]	Cb6[3:0] Y6[11:8]
dw8	Cr7[11:4]	Cr7[3:0] Cb7[11:8]	Cb7[7:0]	Y7[11:4]

8: 36bit RGB – 12bit R + 12bit G + 12bit B

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	R0[7:0]	G0[11:4]	G0[3:0] B0[11:8]	B0[7:0]
dw1	R1[3:0] G1[11:8]	G1[7:0]	B1[11:4]	B1[3:0] R0[11:8]
dw2	G2[11:4]	G2[3:0] B2[11:8]	B2[7:0]	R1[11:4]
dw3	G3[7:0]	B3[11:4]	B3[3:0] R2[11:8]	R2[7:0]
dw4	G4[3:0] B4[11:8]	B4[7:0]	R3[11:4]	R3[3:0] G3[11:8]

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw5	B5[11:4]	B5[3:0] R4[11:8]	R4[7:0]	G4[11:4]
dw6	B6[7:0]	R5[11:4]	R5[3:0] G5[11:8]	G5[7:0]
dw7	B7[3:0] R6[11:8]	R6[7:0]	G6[11:4]	G6[3:0] B6[11:8]
dw8	R7[11:4]	R7[3:0] G7[11:8]	G7[7:0]	B7[11:4]

Not all sensor formats support the output formats shown. Please see the Specification section in the PCIe-2602 User's Manual.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Get Video Capabilities

Acquires li1st of supported resolutions.

Syntax

C/C++

```
int    Sdi_GetVideoCapabilities(UINT    Number,
SDI_RESOLUTION_CAPABILITIES & Caps)
```

C#

```
int  GetVideoCapabilities  (uint  Number,  out
SDI_RESOLUTION_CAPABILITIES Caps)
```

VB.Net

```
GetVideoCapabilities (ByVal Number as UInte-
ger,          ByRef          Caps          As
SDI_RESOLUTION_CAPABILITIES) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

SDI_RESOLUTION_CAPABILITIES

A structure of resolutions that the card supports that is defined as:

C/C++

```
typedef struct _SDI_SENSOR_PROPERTIES
{
char Name[ 256 ];
unsigned long Width;
unsigned long Height;
unsigned long FrameRate;
unsigned long Interlace;
} SDI_SENSOR_PROPERTIES;

typedef struct _SDI_RESOLUTION_CAPABILITIES
{
unsigned long NumSdiResolution;
SDI_SENSOR_PROPERTIES *SdiResolutions;
} SDI_RESOLUTION_CAPABILITIES;
```

C#

```
struct SDI_SENSOR_PROPERTIES
{
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
    public string Name;
```

```
    public uint Width;
    public uint Height;
    public uint FrameRate;
    public uint Interlace;
}

struct RESOLUTION_CAPABILITIES
{
    public uint NumSdiResolution;
    public IntPtr SdiResolutions;
}
```

VB.Net

```
Structure SENSOR_PROPERTIES
<MarshalAs(UnmanagedType.ByValT-
Str,SizeConst:=256)> _
    Dim Name As String
    Dim Width As UInteger
    Dim Height As UInteger
    Dim FrameRate As UInteger
    Dim Interlace As UInteger
End Structure

Structure RESOLUTION_CAPABILITIES
    Dim NumSdiResolution As UInteger
    Dim SdiResolutions As IntPtr
End Structure
```

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Detected Sensor Format

Acquires the sensor format of the currently selected video channel, reading back resolution from the source input.

Syntax

C/C++

```
int  Sdi_GetDetectedSensorFormat(UINT  Number,
int& Format)
```

C#

```
int GetDetectedSensorFormat (uint Number, out
int Format)
```

VB.Net

```
GetDetectedSensorFormat (ByVal Number as UIn-
teger, ByRef Format As Integer) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Format

Read-back format of video sensor as defined in Sensor Format, with format is set as -1 if no sensor is detected or sensor format is not supported. Please see Specification section in PCIe-2602 User's Manual for a list of supported sensors.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Detected Output Format

Acquires the output format of the currently selected video channel, reading back color space from the source input.

Syntax

C/C++

```
int  Sdi_GetDetectedOutputFormat(UINT  Number,
int& Format)
```

C#

```
int GetDetectedOutputFormat (uint Number, out
int Format)
```

VB.Net

```
GetDetectedOutputFormat (ByVal Number as UIn-
teger, ByRef Format As Integer) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Format

Read-back format of video sensor as defined in Output Format, with format set as -1 if no sensor is detected or sensor format is not supported. Please see Specification section in PCIe-2602 User's Manual for a list of supported sensors.

Output format can be detected only if the input sensor conforms to SMPTE 352M.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Image Orientation

Sets or retrieves orientation of images in memory.

Syntax

C/C++

```
int Sdi_SetImageOrientation (UINT Number, UINT Value)
int Sdi_GetImageOrientation (UINT Number, UINT& Value)
```

C#

```
int SetImageOrientation (uint Number, uint Value)
int GetImageOrientation (uint Number, out uint Value)
```

VB.Net

```
SetImageOrientation (ByVal Number as UInteger, ByVal Value As UInteger) As Integer
GetImageOrientation (ByVal Number as UInteger, ByRef Value As UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Value

Indicates the image orientation, as one of:

0: Bottom-up, in which the image buffer starts with the bottom row of pixels, followed by the next row up, and next, with the top row of the image the last row in the buffer, such that the first byte in memory is the bottom-left pixel of the image, with physical layout of a bottom-up image as shown.

E.g. Color space = RGB24

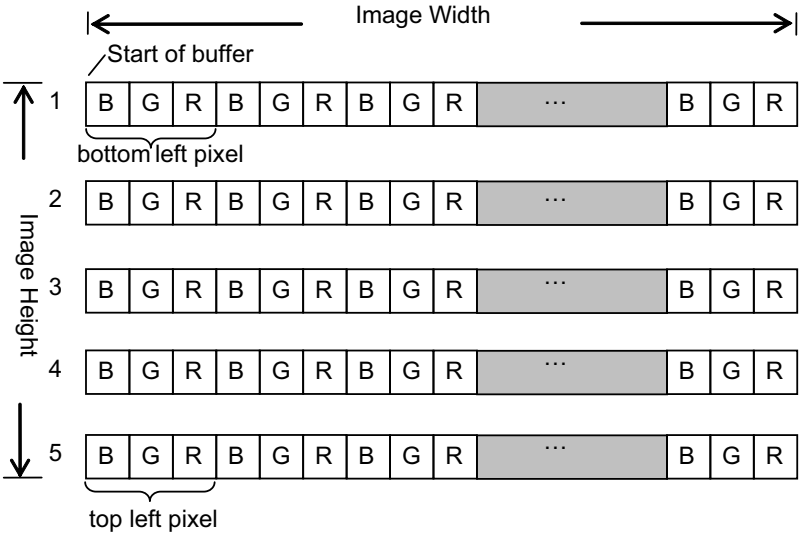


Figure 2-1: Bottom-up Image Orientation

1: Top-down, in which order of the rows is reversed, with the top row the first row in memory, followed by the next row down, with the bottom row of the image the last row in the buffer, such that first byte in memory is the top-left of the image, with physical layout of a top-down image as shown.

E.g. Color space = RGB24

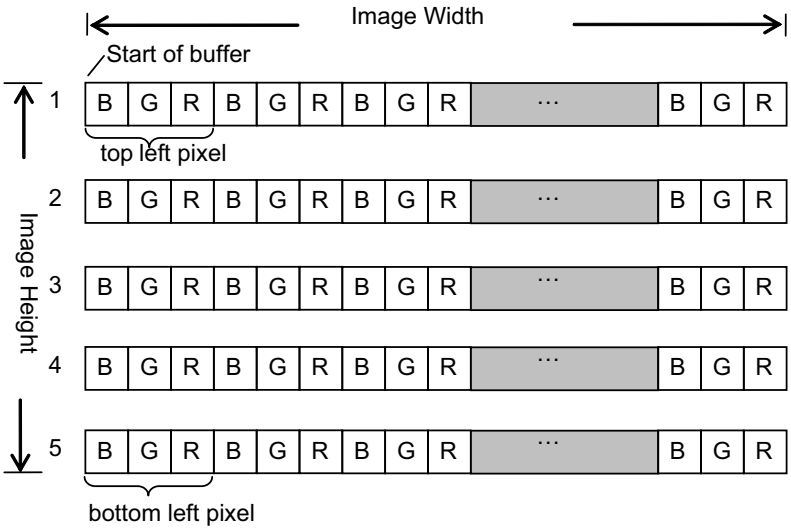


Figure 2-2: Top-down Image Orientation

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

2.4.3 Event & Callback Functions

Event Selector

Sets or retrieves the event type.

Syntax

C/C++

```
int Sdi_SetEventSelector (UINT Number, UINT Mode)
int Sdi_GetEventSelector (UINT Number, UINT & Mode)
```

C#

```
int SetEventSelector (uint Number, uint Mode)
int GetEventSelector (uint Number, out uint Mode)
```

VB.Net

```
SetEventSelector (ByVal Number as UInteger, ByVal Mode as UInteger) As Integer
GetEventSelector (ByVal Number as UInteger, ByRef Mode as UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Mode

Event type, comprising frame and audio events, with frame events the result of library issue of an event when a frame is ready, and audio events the result of library issue of an event when audio data is ready, wherein mode can be:

0: Video frame event

2: Audio data event

Call the function and then call SetEventHandle to set an event handle.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Event Handle

Sets or retrieves event handle, with event and callback determining recovery of video frame or audio information.

Syntax

C/C++

```
int Sdi_SetEventHandle (UINT Number, HANDLE Handle)
int Sdi_GetEventHandle (UINT Number, HANDLE &Handle)
```

C#

```
int SetEventHandle (uint Number, IntPtr Handle)
int SetEventHandle (uint Number, SafeWaitHandle Handle)
int GetEventHandle (uint Number, out IntPtr Handle)
int GetEventHandle (uint Number, out SafeWaitHandle Handle)
```

VB.Net

```
SetEventHandle (ByVal Number as UInteger, ByVal Handle as IntPtr) As Integer
SetEventHandle (ByVal Number as U Integer, ByVal Handle As SafeWaitHandle) As Integer
GetEventHandle (ByVal Number as UInteger, ByRef Handle as IntPtr) As Integer
GetEventHandle (ByVal Number as UInteger, ByRef Handle As SafeWaitHandle) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Handle

Event handle created by the application. After the application waits for an event, call:

GetImageStream to acquire the pointer of frame buffer

GetAudioStream to acquire audio data pointer and size.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Callback Selector

Sets or retrieves callback function type.

Syntax

C/C++

```
int Sdi_SetCallbackSelector (UINT Number, UINT Mode)
int Sdi_GetCallbackSelector (UINT Number, UINT Mode)
```

C#

```
int SetCallbackSelector (uint Number, uint Mode)
int GetCallbackSelector (uint Number, out uint Mode)
```

VB.Net

```
SetCallbackSelector (ByVal Number as UInteger, ByVal Mode as UInteger) As Integer
GetCallbackSelector (ByVal Number as UInteger, ByRef Mode as UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Mode

Callback type, in which the library calls the callback routine when a frame is ready, or audio callback. SetCallback sets a callback function. Mod can be:

0: Video frame callback

2: Audio data callback

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Callback

Sets or retrieves callback handle, with event and callback determining when to recover video frame or audio data information.

Syntax

C/C++

```
int Sdi_SetCallback (UINT Number, SDICALLBACK Fun)
int Sdi_GetCallback (UINT Number, SDICALLBACK Fun)
```

C#

```
int SetCallback (uint Number, SDICALLBACK Fun)
int GetCallback (uint Number, out SDICALLBACK Fun)
```

VB.Net

```
SetCallback (ByVal Number as UInteger, ByVal Fun as SDICALLBACK) As Integer
GetCallback (ByVal Number as UInteger, ByRef Fun as SDICALLBACK) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Fun

Pointer for callback routine, in which a callback function must be declared and set as the parameter. In the callback function, call:

`GetImageStream`	to acquire the pointer of frame buffer
`GetAudioStream`	to acquire audio data pointer and size.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

2.4.4 Acquisition Control Functions

Acquisition Frame Count

Sets or retrieves the total number of frames to be simultaneously captured. Call:

`AcquisitionStart`	to initiate capture
`GetAcquisitionStatus`	to retrieve acquisition state
`AcquisitionStop`	to terminate capture if the acquisition state is stopped

Syntax

C/C++

```
int Sdi_SetAcquisitionFrameCount (UINT Number,
UINT Count)

int Sdi_GetAcquisitionFrameCount (UINT Number,
UINT & Count)
```

C#

```
int  SetAcquisitionFrameCount  (uint  Number,
uint Count)

int GetAcquisitionFrameCount (uint Number, out
uint Count)
```

VB.Net

```
SetAcquisitionFrameCount (ByVal Number as UIn-
teger, ByVal Count as UInteger) As Integer
```

```
GetAcquisitionFrameCount (ByVal Number as UIn-
teger, ByRef Count as UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Count

The frame count to be captured, from among:

0: Capture until

```
AcquisitionStop
```

is called.

> 0: Acquires the desired frame count, and when reached, changes acquisition status to 0 (stopped), where

```
AcquisitionStop
```

must be called to stop acquisition.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Acquisition Start

Initiates video frame and audio data capture.

Syntax

C/C++

```
int Sdi_AcquisitionStart (UINT Number)
```

C#

```
int AcquisitionStart (uint Number)
```

VB.Net

```
AcquisitionStart (ByVal Number as UInteger) As
Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Acquisition Stop

Terminates video frame and audio data capture.

Syntax

C/C++

```
int Sdi_AcquisitionStop (UINT Number)
```

C#

```
int AcquisitionStop (UINT Number)
```

VB.Net

```
AcquisitionStop (ByVal Number as UInteger) As
Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

One Shot

Acquires a single video frame image within a specific time, as an independent function which cannot be used with AcquisitionStart, Callback, or Event. When complete without errors, calling GetImageStream retrieves the frame image pointer.

Syntax

C/C++

```
int Sdi_OneShot (UINT Number, UINT Timeout)
```

C#

```
int OneShot (uint Number, uint Timeout)
```

VB.Net

```
OneShot (ByVal Number as UInteger, ByVal Time-
out as UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Timeout

Maximum waiting time for acquisition, in milliseconds.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Image Stream

Retrieves the image buffer pointer. Usually called during callback, after waiting for a frame event, or after calling One-Shot.

Syntax

C/C++

```
int Sdi_GetImageStream (UINT Number, void
**Buffer)
```

C#

```
int GetImageStream (uint Number, out IntPtr
Buffer)
```

VB.Net

```
GetImageStream (ByVal Number as UInteger,
ByRef Buffer as IntPtr) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Buffer

Image buffer pointer.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Acquisition Status

Retrieves current acquisition status.

Syntax

C/C++

```
int Sdi_GetAcquisitionStatus (UINT Number, UINT &Status)
```

C#

```
int GetAcquisitionStatus (uint Number, out uint Status)
```

VB.Net

```
GetAcquisitionStatus (ByVal Number as UInteger, ByRef Status as UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Status

Acquisition status, from among:

0: Stopped

1: Running

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Acquisition Statistics

Acquires the number of frames captured since Acquisition Start.

Syntax

C/C++

```
int Sdi_GetAcquisitionStatistics (UINT Number,
UINT &Count)
```

C#

```
int GetAcquisitionStatistics (uint Number, out
uint Count)
```

VB.Net

```
GetAcquisitionStatistics (ByVal Number as UIn-
teger, ByRef Count as UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Count

Total amount of frames captured.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Sensor Status

Determines whether the device is connected to a compatible sensor.

Syntax

C/C++

```
int Sdi_GetSensorStatus (UINT Number, UINT&
Locked)
```

C#

```
int GetSensorStatus (uint Number, out uint
Locked)
```

VB.Net

```
GetSensorStatus (ByVal Number as UInteger,
ByRef Locked as UInterger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Locked

In determining whether the input signal is locked, establishes connection of a compatible sensor, from among:

0: No proper signal is detected

1: A proper signal is detected

This function detects whether the input signal is suitable, and GetDetectedSensorFormat must be called to ascertain whether the PCIe-2602 supports the signal.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Save Image

Saves contents of the most recent image buffer as an image file or a raw data file, depending on file extension.

Syntax

C/C++

```
int Sdi_SaveImage (UINT Number, LPTSTR File-
Namet)
```

C#

```
int SaveImage (uint Number, string FileNamet)
```

VB.Net

```
SaveImage (ByVal Number as UInteger, ByVal
FileNamet as String) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

FileName

The library supports:

BMP: if FileName is *.bmp

JPEG: if FileName is *.jpg or *.jpeg

JIFF: if FileName is *.tif

PNG: if FileName is *.png

Raw data: other file type

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Audio Stream

Acquires audio data pointer and size; normally called in call-back routine after receiving an audio data event.

Syntax

C/C++

```
int Sdi_GetAudioStream (UINT Number,
AUDIO_STREAM_INFO& StreamInfo)
```

C#

```
int GetAudioStream (uint Number, out
AUDIO_STREAM_INFO StreamInfo)
```

VB.Net

```
GetAudioStream (ByVal Number as UInteger,
ByRef StreamInfo as AUDIO_STREAM_INFO) As
Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

StreamInfo

Structure of audio data and size as:

C/C++

```
typedef struct _ AUDIO_STREAM_INFO
{
void *Data;
UINT Size;
} AUDIO_STREAM_INFO;
```

C#

```
public struct AUDIO_STREAM_INFO
{
public IntPtr Data;
public uint Size;
}
```

VB.Net

```
Public Structure AUDIO_STREAM_INFO
Public Data As IntPtr
Public Size As UInteger
End Structure
```

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Dropped Frame Count

Acquires the total number of frames dropped since Acquisition Start, as a result of:

- ▷ The buffer queue is full and incoming frames are dropped
- ▷ PCIe bandwidth is insufficient to transfer all frame data
- ▷ Pixel count of the input sensor is less than that set in Sensor Format

Syntax

C/C++

```
int  Sdi_GetDroppedFrameCount  (UINT  Number,
UINT &Count)
```

C#

```
int  GetDroppedFrameCount  (uint  Number,  out
uint Count)
```

VB.Net

```
GetDroppedFrameCount (ByVal Number as UInte-
ger, ByRef Count as UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Count

Number of frames dropped, where DroppedFrameCount = OverflowFrameCount + MismatchFrameCount

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Overflow Frame Count

Acquires the number of frames dropped since Acquisition Start, resulting from:

- ▷ Full buffer queue causes the incoming frame to be dropped
- ▷ PCIe bandwidth is insufficient to transfer all frames' data.

Syntax

C/C++

```
int  Sdi_GetOverflowFrameCount  (UINT  Number,
UINT &Count)
```

C#

```
int  GetOverflowFrameCount  (uint  Number,  out
uint Count)
```

VB.Net

```
int  GetOverflowFrameCount  (uint  Number,  out
uint Count)
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Count

Number of frames dropped

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Mismatch Frame Count

Acquires the number of frames dropped since Acquisition Start, resulting from pixel count of the input sensor being less than that set in Sensor Format.

Syntax

C/C++

```
int  Sdi_GetMismatchFrameCount  (UINT  Number,
UINT &Count)
```

C#

```
int  GetMismatchFrameCount  (uint  Number,  out
uint Count)
```

VB.Net

```
GetMismatchFrameCount (ByVal Number as UInte-
ger, ByRef Count as UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Count

Number of frames dropped

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

2.4.5 Digital I/O Functions

Digital I/O Selector

Sets or retrieves the DI and DO channel set when GetDI and SetDO are called.

Syntax

C/C++

```
int  Sdi_SetDigitalIOSelector  (UINT  Number,
UINT Channel)
int  Sdi_GetDigitalIOSelector  (UINT  Number,
UINT& Channel)
```

C#

```
int SetDigitalIOSelector (uint Number, uint Channel)
int GetDigitalIOSelector (uint Number, out uint Channel)
```

VB.Net

```
SetDigitalIOSelector (ByVal Number As UInteger, ByVal Channel As UInteger) As Integer
GetDigitalIOSelector (ByVal Number As UInteger, ByRef Channel As UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Count

Number of I/O channels, with allowed values from 0 to 3 and 0xFFFFFFFF for all channels

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

DI

Retrieves DI channel status.

Syntax

C/C++

```
int Sdi_GetDI (UINT Number, UINT& Value)
```

C#

```
int GetDI (uint Number, out uint Value)
```

VB.Net

```
GetDI (ByVal Number As UInteger, ByRef Value As UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Value

1. State of the DI channel if SetDigitalIOSelector channel is between 0 and 3, with:

0: Low

1: High

2. State of all DI channels if SetDigitalIOSelector channel is 0xFFFFFFFF, with:

bit n = 0: Low

bit n = 0: High

in which n is 0 to 3 representing the state of DI0 to DI3 respectively.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

DO

Retrieves DO channel status.

Syntax

C/C++

```
int Sdi_SetDO (UINT Number, UINT Value)
int Sdi_GetDO (UINT Number, UINT& Value)
```

C#

```
int SetDO (uint Number, uint Value)
int GetDO (uint Number, out uint Value)
```

VB.Net

```
SetDO (ByVal Number As UInteger, ByVal Value
As UInteger) As Integer
GetDO (ByVal Number As UInteger, ByRef Value
As UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Value

State of the DO channel of SetDigitalIOSelector is between 0 to 3, with the following values:

0: Low

1: High

2. State of all DO channels if Channel of SetDigitalIOSelector is 0xFFFFFFFF, with the following values:

bit n = 0: Low

bit n = 0: High

where n is from 0 to 3 representing the state of DO0 to DO3 respectively.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

2.4.6 Audio Format Functions

Bits Per Sample

Sets or retrieves the number of bits per sample.

Syntax

C/C++

```
int  Sdi_SetAudioBitsPerSample  (UINT  Number,
UINT Value)
int  Sdi_GetAudioBitsPerSample  (UINT  Number,
UINT& Value)
```

C#

```
int SetAudioBitsPerSample (uint Number, uint
Value)
```

```
int GetAudioBitsPerSample (uint Number, out
uint Value)
```

VB.Net

```
SetAudioBitsPerSample (ByVal Number As UInte-
ger, ByVal Value As UInteger) As Integer
GetAudioBitsPerSample (ByVal Number As UInte-
ger, ByRef Value As UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Value

Specifies the number of bits per sample, with the allowed values:

16: 16 bits per sample

20: 24 bits per sample with 20 most significant bits valid

24: 24 bits per sample

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

Audio Clock Phase Data Bypass

Sets or retrieves the bypass status of audio clock phase data, with audio clock phase indicated by the number of video clocks between the video sample and the audio sample, and, at the receiver side, the audio clock generation extracts the phase data and re-generates audio clock as the same phase difference. If the incoming HD/3G SDI stream includes no phase data or incorrect phase data, audio noise can result, while SD SDI has no clock phase data, and thus is unaffected by this setting.

Syntax

C/C++

```
int   Sdi_SetAudioClockPhaseDataBypass   (UINT
Number, UINT Status)

int   Sdi_GetAudioClockPhaseDataBypass   (UINT
Number, UINT& Status)
```

C#

```
int SetAudioClockPhaseDataBypass (uint Number,
uint Status)

int GetAudioClockPhaseDataBypass (uint Number,
out uint Status)
```

VB.Net

```
SetAudioClockPhaseDataBypass (ByVal Number As
UInteger, ByVal Status As UInteger) As Integer

GetAudioClockPhaseDataBypass (ByVal Number As
UInteger, ByRef Status As UInteger) As Integer
```

Parameter(s)

Number

The number of the SDI channel, with allowed values from 0 to 31.

Status

Specifies the bypass status of audio clock phase data, with the allowed values:

0: no bypass

1: bypass audio clock phase data generating audio clocks only based on the assumption that A/V clocks are synced.

Return Value

No error occurs if return value ≥ 0; if a negative value, please refer to Other Functions for return code error information.

2.4.7 Other Functions

Error Text

Retrieves error text string.

Syntax

C/C++

```
int Sdi_GetErrorText (int code, char *Text)
```

C#

```
string GetErrorText (int code)
```

VB.Net

```
GetErrorText (ByVal code As Integer) As String
```

Parameter(s)

Code

Error code returned by other functions.

Text

Error text string, for containment of which a buffer of maximum 160 bytes8 must be allocated.

Return Value

C/C++ This page intentionally left blank.

Always return 0.

C#

Return the error text

VB.Net

Return the error text

3 DirectShow Programming Guide

.


NOTE:

Complete documentation for DirectShow application programming can be found at:
http://msdn.microsoft.com/en-us/library/dd390351(VS.85).aspx
by searching for “DirectShow”. If a DirectX SDK is installed, this documentation is also available from DirectX SDK Help. The most recent version of DirectShow SDK was moved to Windows SDK.

The goal when writing a DirectShow Application is to construct a filter graph by connecting several filters together, to execute tasks such as previewing or capturing video/audio, and multiplexing to write to a file. Each filter performs a single operation and pass data from its output is pinned to the input of the next filter in the graph.

To build a capture graph using a program, the interface pointer of the capture filter must first be obtained. The ADLINK SDI Capture filter can be obtained through the system device enumerator.

After holding an interface pointer to the capture filter object, method IGraphBuilder::AddSourceFilter adds the source filter object to the filter graph. IFilterGraph::AddFilter adds other downstream filters to the filter graph.

After filters are added, calling IFilterGraph::ConnectDirect or IGraphBuilder::Connect connect output pins from upstream filters to the input pins of the downstream filters.

Calling IAMCrossbar::Route to switch source channel, via methods IMediaControl::Run, IMediaControl::Pause or IMediaControl::Stop changes the filter state to running, paused or stopped.

Filters required for capturing video streams are

listed as follows, with detailed information for each and its pins. Example filter graphs for capturing video streams are also illustrated in this chapter with two ways of controlling the device driver.

3.1 Filters

This section lists the filters necessary to construct a filter graph for capturing a video stream.

3.1.1 Source Filters

ADLINK SDI Capture

A WDM Streaming Capture Device, it is actually a kernel-mode KsProxy plug-in. An application can treat it simply as a filter. Use System Device Enumerator to add this filter to a filter graph.

Filter	ADLINK SDI Capture
Filter CLSID	N/A
Filter Category Name	WDM Streaming Capture Devices
Filter Category	AM_KSCATEGORY_CAPTURE
Capture Pin Supported Media Types	MEDIATYPE_Video Subtypes: MEDIASUBTYPE_RGB24 MEDIASUBTYPE_RGB36* MEDIASUBTYPE_BGR30* MEDIASUBTYPE_YUV8* MEDIASUBTYPE_YUY2 MEDIASUBTYPE_YU10* MEDIASUBTYPE_YUV10* MEDIASUBTYPE_YU12* MEDIASUBTYPE_YUV12*
Exported Interfaces	IAMAnalogVideoDecoder IAMDroppedFrames IAMStreamConfig
Merit	MERIT_DO_NOT_USE

* Please see Section 3.3: Color Space

3.1.2 Example Graphs

The Microsoft DirectX SDK provides the GraphEdit debugging utility, which can simulate graph building. Desired filters can be chosen from the the Insert Filters command of the Graph menu. Filters are organized by categories. Insert Filter adds the filters to

a graph, and two filters' pins can be selected by dragging from one output pin to another. An arrow will be drawn if both pins agree on the connection.

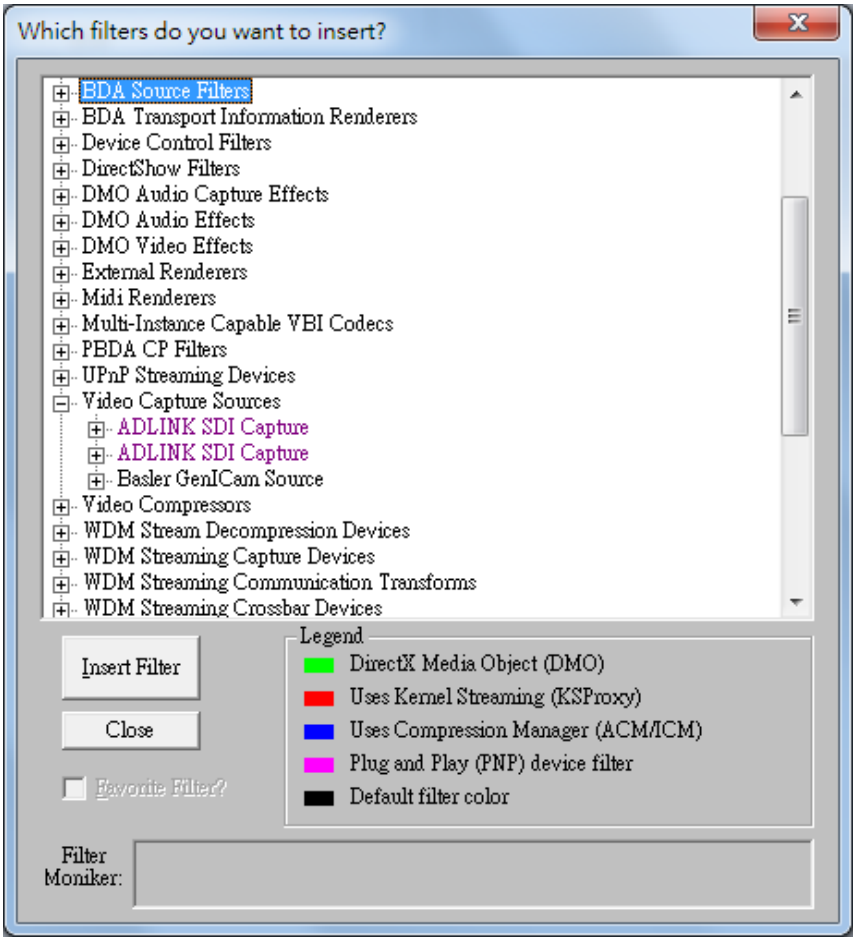


Figure 3-1: GraphEdit Insert Filters Dialog

Example Graph

To generate an example graph:

1. Open GraphEdit.exe.
2. Select ‘Insert a filter into the graph’ from the toolbar and enter'ADLink SDI Capture', 'Video Render' and 'Default DirectSound Device' filters in the 'Video Capture Source' group, the 'DirectShow Filters' group and 'Audio Renders' group.
3. Drag 'Video Capture' pin to 'VMR Input0' pin and 'Audio Capture' pin to 'Audio Input pin' as shown.
4. Select ‘Play the graph’ from the toolbar to begin preview.

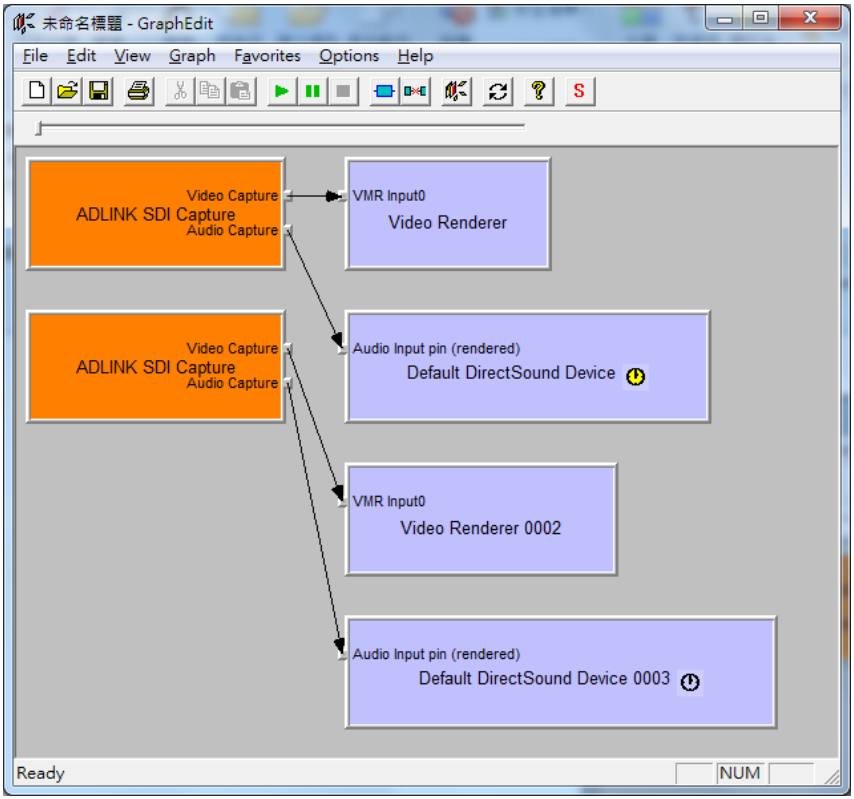


Figure 3-2: GraphEdit Interface

NOTE: If VMR substitutes for the default Video Renderer, the preview video shows a vertical mirror video, which can be resolved by inserting a Color Space Converter filter before.

3.2 Driver Control

The ADLINK SDI Capture filter provides control of video configuration by either property pages or access to COM interfaces.

3.2.1 Property Pages

The driver provides two embedded property pages. To show these property pages, use Windows API: OleCreatePropertyFrame.

Details about Displaying a Filter's Property Page can be found on the Microsoft MSDN homepage.

Sample code for adding property pages is as follows.

```
// pFilter points to an ADLINK SDI Capture filter

ISpecifyPropertyPages *pSpecify;
HRESULT hr;
hr = pFilter->QueryInterface(IID_ISpecifyPropertyPages, (void **)&pSpecify);
if (SUCCEEDED(hr))
{
 FILTER_INFO FilterInfo;
 pFilter->QueryFilterInfo(&FilterInfo);
 FilterInfo.pGraph->Release();

 CAUUID caGUID;
 pSpecify->GetPages(&caGUID);
 pSpecify->Release();
```

```
    OleCreatePropertyFrame(
     NULL,    // Parent window
     0,    // x (Reserved)
     0,    // y (Reserved)
     FilterInfo.achName,  // Caption for the dialog box
     1,    // Number of filters
     (IUnknown **)&m_pFilter, // Pointer to the filter
     caGUID.cElems,  // Number of property pages
     caGUID.pElems,  // Pointer to property page CLSIDs
     0,    // Locale identifier
     0,    // Reserved
     NULL    // Reserved
     );
    CoTaskMemFree(caGUID.pElems);
   }
```

3.3 Color Space

ADLINK SDI Capture supports the following color spaces:

('x' indicates an ignored bit)

MEDIASUBTYPE_YUY2 - 16bit YCbCr 4:2:2 – 8bit Y + 8bit Cb/Cr

8.5	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Cr	Y	Cb	Y

MEDIASUBTYPE_YUV8 - 24bit YCbCr 4:4:4 – 8bit Y + 8bit Cb + 8bit Cr

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Y1	Cr0	Cb0	Y0
dw1	Cb2	Y2	Cr1	Cb1
dw2	Cr3	Cb3	Y3	Cr2

Compression (FOURCC code) is ‘YUV8’ and GUID is 38565559-0000-0010-8000-00AA00389B71

MEDIASUBTYPE_RGB24 - 24bit RGB – 8bit R + 8bit G + 8bit B

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	B1	R0	G0	B0
dw1	G2	B2	R1	G1
dw2	R3	G3	B3	R2

MEDIASUBTYPE_YU10 - 20bit YCbCr 4:2:2 – 10bit Y + 10bit Cb/Cr

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Cr0[1:0]Y1[9:4]	Y1[3:0]Cb0[9:6]	Cb0[5:0]Y0[9:8]	Y0[7:0]
dw1	Y3[3:0]Cb2[9:6]	Cb2[5:0]Y2[9:8]	Y2[7:0]	Cr0[9:2]
dw2	Cb4[5:0]Y4[9:8]	Y4[7:0]	Cr2[9:2]	Cr2[1:0]Y3[9:4]
dw3	Y6[7:0]	Cr4[9:2]	Cr4[1:0]Y5[9:4]	Y5[3:0]Cb4[9:6]
dw4	Cr6[9:2]	Cr6[1:0]Y7[9:4]	Y7[3:0]Cb6[9:6]	Cb6[5:0]Y6[9:8]

Compression (FOURCC code) is ‘YU10’ and GUID is 30315559-0000-0010-8000-00AA00389B71

MEDIASUBTYPE_YUV10 - 30bit YCbCr 4:4:4 – 10bit Y + 10bit Cb + 10bit Cr

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	xxCr[9:4]	Cr[3:0]Cb[9:6]	Cb[5:0]Y[9:8]	Y[7:0]

Compression (FOURCC code) is ‘YUVA’ and GUID is 41565559-0000-0010-8000-00AA00389B71

MEDIASUBTYPE_BGR30 - 30bit RGB – 8bit R + 8bit G + 8bit B

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	xxB[9:4]	B[3:0]G[9:6]	G[5:0]R[9:8]	R[7:0]

Compression (FOURCC code) is ‘BGRA’ and GUID is 41524742-0000-0010-8000-00AA00389B71

MEDIASUBTYPE_YU12 - 24bit YCbCr 4:2:2 – 12bit Y + 12bit Cb/Cr

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Y1[7:0]	Cb0[11:4]	Cb0[3:0]Y0[11:8]	Y0[7:0]
dw1	Cb2[3:0]Y2[11:8]	Y2[7:0]	Cr0[11:4]	Cr0[3:0]Y1[11:8]
dw2	Cr2[11:4]	Cr2[3:0]Y3[11:8]	Y3[7:0]	Cb2[11:4]

Compression (FOURCC code) is ‘YU12’ and GUID is 32315559-0000-0010-8000-00AA00389B71

MEDIASUBTYPE_YUV12 - 36bit YCbCr 4:4:4 – 12bit Y + 12bit Cb + 12bit Cr

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	Cr0[7:0]	Cb0[11:4]	Cb0[3:0]Y0[11:8]	Y0[7:0]
dw1	Cr1[3:0]Cb1[11:8]	Cb1[7:0]	Y1[11:4]	Y1[3:0]Cr0[11:8]
dw2	Cb2[11:4]	Cb2[3:0]Y2[11:8]	Y2[7:0]	Cr1[11:4]
dw3	Cb3[7:0]	Y3[11:4]	Y3[3:0]Cr2[11:8]	Cr2[7:0]
dw4	Cb4[3:0]Y4[11:8]	Y4[7:0]	Cr3[11:4]	Cr3[3:0]Cb3[11:8]
dw5	Y5[11:4]	Y5[3:0]Cr4[11:8]	Cr4[7:0]	Cb4[11:4]
dw6	Y6[7:0]	Cr5[11:4]	Cr5[3:0]Cb5[11:8]	Cb5[7:0]
dw7	Y7[3:0]Cr6[11:8]	Cr6[7:0]	Cb6[11:4]	Cb6[3:0]Y6[11:8]
dw8	Cr7[11:4]	Cr7[3:0]Cb7[11:8]	Cb7[7:0]	Y7[11:4]

Compression (FOURCC code) is ‘YUVC’ and GUID is 43565559-0000-0010-8000-00AA00389B71

MEDIASUBTYPE_RGB36 - 36bit RGB – 12bit R + 12bit G + 12bit B

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw0	R0[7:0]	G0[11:4]	G0[3:0]B0[11:8]	B0[7:0]
dw1	R1[3:0]G1[11:8]	G1[7:0]	B1[11:4]	B1[3:0]R0[11:8]
dw2	G2[11:4]	G2[3:0]B2[11:8]	B2[7:0]	R1[11:4]

DWORD	Pixel Data [31:0]			
	[31:24]	[23:16]	[15:8]	[7:0]
dw3	G3[7:0]	B3[11:4]	B3[3:0]R2[11:8]	R2[7:0]
dw4	G4[3:0]B4[11:8]	B4[7:0]	R3[11:4]	R3[3:0]G3[11:8]
dw5	B5[11:4]	B5[3:0]R4[11:8]	R4[7:0]	G4[11:4]
dw6	B6[7:0]	R5[11:4]	R5[3:0]G5[11:8]	G5[7:0]
dw7	B7[3:0]R6[11:8]	R6[7:0]	G6[11:4]	G6[3:0]B6[11:8]
dw8	R7[11:4]	R7[3:0]G7[11:8]	G7[7:0]	B7[11:4]

Compression (FOURCC code) is ‘RGBC’ and GUID is 43424752-0000-0010-8000-00AA00389B71

3.4 Proprietary Interfaces

The following interfaces are specific to the ADLINK SDI Capture filter, not being standard interfaces in DirectShow. COM interfaces, they can be acquired from the ADLINK SDI Capture filter by calling IKsPropertySet::Set and IKsPropertySet::Get.

The parameter rguidPropSet of IKsPropertySet::Set/Get function is defined as:

GUID PROPSETID_AVS260X_CUSTOM =

{ 0xf7c238b6L, 0x3209, 0x4497, 0x9a, 0xa6, 0x24, 0x70, 0x77, 0x20, 0x6, 0xe0 };

The parameter dwPropID of IKsPropertySet::Set/Get function is defined as:

typedef enum {

CUSTOM_PROPERTY_GET_DEVICE_VERSION = 10,

CUSTOM_PROPERTY_GET_FIRMWARE_VERSION = 11,

CUSTOM_PROPERTY_GET_DRIVER_VERSION = 12,

CUSTOM_PROPERTY_GET_DEVICE_MODEL = 13,

CUSTOM_PROPERTY_GET_CARD_ID = 14,

```
    CUSTOM_PROPERTY_GET_CHANNEL_NUMBER = 15,
    CUSTOM_PROPERTY_XET_SENSOR_FORMAT = 20,

CUSTOM_PROPERTY_GET_DETECTED_SENSOR_FORMAT
= 21,
    CUSTOM_PROPERTY_XET_IMAGE_ORIENTATION = 22,

CUSTOM_PROPERTY_GET_DETECTED_OUTPUT_FORMAT =
23,
    CUSTOM_PROPERTY_XET_DO = 30,
    CUSTOM_PROPERTY_GET_DI = 31,
    CUSTOM_PROPERTY_GET_OVERFLOW_COUNTER = 40,
    CUSTOM_PROPERTY_GET_MISMATCH_COUNTER = 41,

};
```

The parameter pPropData is a pointer to ULONG or LONG variable and parameter cbPropData is 4.

1. CUSTOM_PROPERTY_GET_DEVICE_VERSION

   The property allows acquisition of the hardware device version. The version is a 2 hexadecimal integer, with one letter plus one digit.

   EXAMPLE:

```
ULONG Version;
DWORD dwReturn;
m_pKsPropertySet->Get( PROPSETID_AVS260X_CUSTOM,
CUSTOM_PROPERTY_GET_DEVICE_VERSION,
NULL, 0,
```

&Version, sizeof(ULONG), &dwReturn);

2. CUSTOM_PROPERTY_GET_FIRMWARE_VERSION

The property allows retrieval of the firmware version with the formula:

Year = (Version >> 28) + 2000

Month = (Version >> 24) & 0x0F

Day = (Version >> 16) & 0xFF

Hour = (Version >> 8) & 0xFF

Minute = Version & 0xFF

EXAMPLE:

ULONG Version;

DWORD dwReturn;

m_pKsPropertySet->Get(PROPSETID_AVS260X_CUSTOM,

CUSTOM_PROPERTY_GET_FIRMWARE_VERSION,

NULL, 0,

&Version, sizeof(ULONG), &dwReturn);

3. CUSTOM_PROPERTY_GET_DRIVER_VERSION

The property allows retrieval of the driver version with the formula:

Major = (Version >> 24) & 0xFF

Minor = (Version >> 16) & 0xFF

Release = (Version >> 8) & 0xFF

EXAMPLE:

ULONG Version;

DWORD dwReturn;

```
m_pKsPropertySet->Get(
PROPSETID_AVS260X_CUSTOM,
CUSTOM_PROPERTY_GET_DRIVER_VERSION,
NULL, 0,
&Version, sizeof(ULONG), &dwReturn);
```

4. CUSTOM_PROPERTY_GET_DEVICE_MODEL

The property allows retrieval of the device model with the values:

0: PCIe-2602

1: PCIe-2604

EXAMPLE:

```
ULONG Vaule;
DWORD dwReturn;
m_pKsPropertySet->Get( PROPSETID_AVS260X_CUSTOM,
CUSTOM_PROPERTY_GET_DEVICE_MODEL,
NULL, 0,
&Value, sizeof(ULONG), &dwReturn);
```

5. CUSTOM_PROPERTY_GET_CARD_ID

The property acquires the card ID.

EXAMPLE:

```
ULONG id;
DWORD dwReturn;
m_pKsPropertySet->Get( PROPSETID_AVS260X_CUSTOM,
CUSTOM_PROPERTY_GET_CARD_ID,
NULL, 0,
```

&id, sizeof(ULONG), &dwReturn);

6. CUSTOM_PROPERTY_GET_CHANNEL_NUMBER

The property allows acquisition of the hardware channel number with value 0 for SDI channel 0 and 1 for SDI channel 1.

EXAMPLE:

ULONG Number;

DWORD dwReturn;

m_pKsPropertySet->Get(PROPSETID_AVS260X_CUSTOM,

CUSTOM_PROPERTY_GET_FIRMWARE_VERSION,

NULL, 0,

&Number, sizeof(ULONG), &dwReturn);

7. CUSTOM_PROPERTY_XET_SENSOR_FORMAT

The property allows setting or retrieval of the sensor format, with supported formats of:

Format	Resolution	Value
525i 29.97/30 fps	720x486	0
625i 25 fps	720x576	1
720p 24 fps	1280x720	2
720p 25 fps	1280x720	3
720p 30 fps	1280x720	4
720p 50 fps	1280x720	5
720p 59.94/60 fps	1280x720	6
1080i 25 fps	1920x1080	7
1080i 29.97/30 fps	1920x1080	8
1080p 23.98/24 fps	1920x1080	9
1080p 25 fps	1920x1080	10

Format	Resolution	Value
1080p 30 fps	1920x1080	11
1080p 50 fps	1920x1080	12
1080p 59.94/60 fps	1920x1080	13

EXAMPLE #01: Retrieve the format value

```
ULONG Format;
DWORD dwReturn;
m_pKsPropertySet->Get( PROPSETID_AVS260X_CUSTOM,
CUSTOM_PROPERTY_XET_SENSOR_FORMAT,
NULL, 0,
&Format, sizeof(ULONG), &dwReturn);
```

EXAMPLE #02: Set the format value

```
ULONG Format = 11; // = 1080p 30 fps
m_pKsPropertySet->Set( PROPSETID_AVS260X_CUSTOM,
CUSTOM_PROPERTY_XET_SENSOR_FORMAT,
NULL, 0,
&Format, sizeof(ULONG);
```

8. CUSTOM_PROPERTY_GET_DETECTED_SENSOR_FORMAT

The driver automatically detects video format and reports it to the software, with supported formats as shown, where -1indicates a non-supported format.

EXAMPLE

```
LONG Format;
DWORD dwReturn;
```

```
m_pKsPropertySet->Get( PROPSETID_AVS260X_CUSTOM,
CUSTOM_PROPERTY_GET_DETECTED_SENSOR_FORM
AT,
NULL, 0,
&Format, sizeof(LONG), &dwReturn);
```

9. CUSTOM_PROPERTY_XET_IMAGE_ORIENTATION

The property allows you to set or retrieve the image orientation arranged in the memory, with supported values:

0: bottom-up

1: top-down (default)

EXAMPLE #01: Retrieve the image orientation

```
ULONG Orientation;
DWORD dwReturn;
m_pKsPropertySet->Get( PROPSETID_AVS260X_CUSTOM,
CUSTOM_PROPERTY_XET_IMAGE_ORIENTATION,
NULL, 0,
&Orientation, sizeof(ULONG), &dwReturn);
```

EXAMPLE #02: Set the image orientation

```
ULONG Orientation = 1; // = top-down
m_pKsPropertySet->Set( PROPSETID_AVS260X_CUSTOM,
CUSTOM_PROPERTY_XET_IMAGE_ORIENTATION,
NULL, 0,
& Orientation, sizeof(ULONG);
```

10. CUSTOM_PROPERTY_GET_DETECTED_OUTPUT_FORMAT

Output format can be detected and reported to the application, with supported formats as described, where -1 indicates a non-supported format, or no embedded video payload identification. The output format can be detected only if the input sensor conforms to SMPTE 352M.

Format	Value
YCbCr 4:2:2 8 bit	0
YCbCr 4:4:4 8 bit	1
RGB 8bit	2
YCbCr 4:2:2 10 bit	3
YCbCr 4:4:4 10 bit	4
RGB 10 bit	5
YCbCr 4:2:2 12 bit	6
YCbCr 4:4:4 12 bit	7
RGB 12bit	8

EXAMPLE

```
LONG Format;
DWORD dwReturn;
m_pKsPropertySet->Get( PROPSETID_AVS260X_CUSTOM,
CUSTOM_PROPERTY_GET_DETECTED_OUTPUT_FORMAT,
NULL, 0,
&Format, sizeof(LONG), &dwReturn);
```

11. CUSTOM_PROPERTY_XET_DO

The property sets or retrieves of all DO status, with supported values of each bit (bit 0 to bit 4):

0: low

1: high

EXAMPLE #01: Retrieves all DO status

```
ULONG State;
DWORD dwReturn;
m_pKsPropertySet->Get( PROPSETID_AVS260X_CUSTOM,
CUSTOM_PROPERTY_XET_DO,
NULL, 0,
&State, sizeof(ULONG), &dwReturn);
```

EXAMPLE #02: Sets all DO status

```
ULONG State = 0xF; // = all DOs output high
m_pKsPropertySet->Set( PROPSETID_AVS260X_CUSTOM,
CUSTOM_PROPERTY_XET_DO,
NULL, 0,
&State, sizeof(ULONG);
```

12.CUSTOM_PROPERTY_GET_DI

The property retrieves all DI status, with supported values of each bit (bit 0 to bit 4):

0: low

1: high

EXAMPLE: Retrieves all DI status

```
ULONG State;
DWORD dwReturn;
m_pKsPropertySet->Get( PROPSETID_AVS260X_CUSTOM,
CUSTOM_PROPERTY_GET_DI,
NULL, 0,
&State, sizeof(ULONG), &dwReturn);
```

13.CUSTOM_PROPERTY_GET_OVERFLOW_COUNTER

The property retrieves the number of frames dropped since start capture due to:

The buffer queue is full and incoming frames were dropped

PCIe bandwidth is insufficient to transfer all frame data.

EXAMPLE:

```
ULONG Count;
DWORD dwReturn;
m_pKsPropertySet->Get( PROPSETID_AVS260X_CUSTOM,
CUSTOM_PROPERTY_GET_OVERFLOW_COUNTER,
NULL, 0,
&Count, sizeof(ULONG), &dwReturn);
```

14.CUSTOM_PROPERTY_GET_MISMATCH_COUNTER

The property retrieves the number of frames dropped since start capture due to pixel count of the input sensor being less than that set in Sensor Format.

EXAMPLE:

```
ULONG Count;
DWORD dwReturn;
m_pKsPropertySet->Get( PROPSETID_AVS260X_CUSTOM,
CUSTOM_PROPERTY_GET_MISMATCH_COUNTER,
NULL, 0,
&Count, sizeof(ULONG), &dwReturn);
```

3.5 Build Environment Settings

3.5.1 Include Files

Applications must include the files as shown.

Include File	Description
DShow.h	The header file is required for all C++ applications, and Microsoft DirectX SDK must be first installed..
DirectShowLib	Imports this name space for all Microsoft .Net applications by adding a reference to file DirectShowLib-2005.dll.

3.5.2 Library Files

Applications must include the library files as shown.

Library File	Description
Strmiids.lib	Exports class identifiers (CLSIDs) and interface identifiers (IIDs). All C++ applications require this library.
Quartz.lib	Exports the AMGetErrorText function for C++ applications. If you do not call this function, this library is not required.
DirectShowLib-2005.dll	The class library of DirectShow is required for all Microsoft .Net applications.

NOTE:

The libraries for Microsoft .Net applications work and test on .Net Framwork 2.0, Microsoft Visual Studio 2005 is recco-mended for building .Net applications.

3.5.3 Microsoft Visual C++

For VC++, the build environment must be set up prior to building, as follows.

1. Open the solution file (baseclasses.sln) or the project file (baseclasses.dsw) under %DXSDK%\Samples\C++\DirectShow\BaseClasses and build it.
2. Add the paths to the include directory inproject settings:
   %DXSDK%\include
   %DXSDK%\Samples\C++\DirectShow\BaseClasses
3. Add the paths to the additional library directory in project settings:
   %DXSDK%\Lib
   %DXSDK%\Samples\C++\DirectShow\BaseClasses\Release
   %DXSDK%\Samples\C++\DirectShow\BaseClasses\Debug

For the above, %DXSDK% is the installation path of DirectX SDK.

3.5.4 .Net Programming Users

Microsoft DirectShow provides only C++ programming. .Net users must convert DirectShow COM objects to .net classes. Source codes and samples from a supporting sourceforge project can be downloaded from http://sourceforge.net/projects/directshownet/.

This page intentionally left blank.

Important Safety Instructions

For user safety, please read and follow all **instructions**, **WARNINGS**, **CAUTIONS,** and **NOTES** marked in this manual and on the associated equipment before handling/operating the equipment.

- Read these safety instructions carefully.
- Keep this user's manual for future reference.
- Read the specifications section of this manual for detailed information on the operating environment of this equipment.
- When installing/mounting or uninstalling/removing equipment:
  - Turn off power and unplug any power cords/cables.
- To avoid electrical shock and/or damage to equipment:
  - Keep equipment away from water or liquid sources;
  - Keep equipment away from high heat or high humidity;
  - Keep equipment properly ventilated (do not block or cover ventilation openings);
  - Make sure to use recommended voltage and power source settings;
  - Always install and operate equipment near an easily accessible electrical socket-outlet;
  - Secure the power cord (do not place any object on/over the power cord);
  - Only install/attach and operate equipment on stable surfaces and/or recommended mountings; and,
  - If the equipment will not be used for long periods of time, turn off and unplug the equipment from its power source.

- Never attempt to fix the equipment. Equipment should only be serviced by qualified personnel.

A Lithium-type battery may be provided for uninterrupted, backup or emergency power.

Risk of explosion if battery is replaced with one of an incorrect type. Dispose of used batteries appropriately.

- Equipment must be serviced by authorized technicians when:
  - The power cord or plug is damaged;
  - Liquid has penetrated the equipment;
  - It has been exposed to high humidity/moisture;
  - It is not functioning or does not function according to the user's manual;
  - It has been dropped and/or damaged; and/or,
  - It has an obvious sign of breakage.

Getting Service

Contact us should you require any service or assistance.

ADLINK Technology, Inc.
Address: 9F, No.166 Jian Yi Road, Zhonghe District
New Taipei City 235, Taiwan
新北市中和區建一路 166 號 9 樓
Tel: +886-2-8226-5877
Fax: +886-2-8226-5717
Email: service@adlinktech.com

Ampro ADLINK Technology, Inc.
Address: 5215 Hellyer Avenue, #110, San Jose, CA 95138, USA
Tel: +1-408-360-0200
Toll Free: +1-800-966-5200 (USA only)
Fax: +1-408-360-0222
Email: info@adlinktech.com

ADLINK Technology (China) Co., Ltd.
Address: 上海市浦东新区张江高科技园区芳春路 300 号 (201203)
300 Fang Chun Rd., Zhangjiang Hi-Tech Park,
Pudong New Area, Shanghai, 201203 China
Tel: +86-21-5132-8988
Fax: +86-21-5132-3588
Email: market@adlinktech.com

ADLINK Technology Beijing
Address: 北京市海淀区上地东路 1 号盈创动力大厦 E 座 801 室(100085)
Rm. 801, Power Creative E, No. 1,
Shang Di East Rd., Beijing, 100085 China
Tel: +86-10-5885-8666
Fax: +86-10-5885-8626
Email: market@adlinktech.com

ADLINK Technology Shenzhen
Address: 深圳市南山区科技园南区高新南七道 数字技术园
A1 栋 2 楼 C 区 (518057)
2F, C Block, Bldg. A1, Cyber-Tech Zone, Gao Xin Ave. Sec. 7,
High-Tech Industrial Park S., Shenzhen, 518054 China
Tel: +86-755-2643-4858
Fax: +86-755-2664-6353
Email: market@adlinktech.com

LiPPERT ADLINK Technology GmbH
Address: Hans-Thoma-Strasse 11, D-68163, Mannheim, Germany
Tel: +49-621-43214-0
Fax: +49-621 43214-30
Email: emea@adlinktech.com

ADLINK Technology, Inc. (French Liaison Office)
Address: 15 rue Emile Baudot, 91300 Massy CEDEX, France
Tel: +33 (0) 1 60 12 35 66
Fax: +33 (0) 1 60 12 35 66
Email: france@adlinktech.com

ADLINK Technology Japan Corporation
Address: 〒101-0045 東京都千代田区神田鍛冶町 3-7-4
神田 374 ビル 4F
KANDA374 Bldg. 4F, 3-7-4 Kanda Kajicho,
Chiyoda-ku, Tokyo 101-0045, Japan
Tel: +81-3-4455-3722
Fax: +81-3-5209-6013
Email: japan@adlinktech.com

ADLINK Technology, Inc. (Korean Liaison Office)
Address: 서울시 서초구 서초동 1675-12 모인터빌딩 8 층
8F Mointer B/D,1675-12, Seocho-Dong, Seocho-Gu,
Seoul 137-070, Korea
Tel: +82-2-2057-0565
Fax: +82-2-2057-0563
Email: korea@adlinktech.com

ADLINK Technology Singapore Pte. Ltd.
Address: 84 Genting Lane #07-02A, Cityneon Design Centre,
Singapore 349584
Tel: +65-6844-2261
Fax: +65-6844-2263
Email: singapore@adlinktech.com

ADLINK Technology Singapore Pte. Ltd. (Indian Liaison Office)
Address: 1st Floor, #50-56 (Between 16th/17th Cross) Margosa Plaza,
Margosa Main Road, Malleswaram, Bangalore-560055, India
Tel: +91-80-65605817, +91-80-42246107
Fax: +91-80-23464606
Email: india@adlinktech.com

ADLINK Technology, Inc. (Israeli Liaison Office)
Address: 6 Hasadna St., Kfar Saba 44424, Israel
Tel: +972-9-7446541
Fax: +972-9-7446542
Email: israel@adlinktech.com