# HDV62A

High-Definition Video/Audio Capture Card

**Function Library Reference**


**Revision:** 2.00

**Revision Date:** Nov.23, 2012

**Part No:** 50-11248-1000

Recycled Paper

**Advance Technologies; Automate the World.**

# Revision History

| Revision | Release Date | Description of Change(s) |
|---|---|---|
| 2.00 | Nov. 23, 2012 | 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 reference 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

This page intentionally left blank.

# List of Figures

This page intentionally left blank.

# List of Tables

This page intentionally left blank.

# 1 Introduction

## 1.1 Overview

The HDV62A full HD video/audio capture card, based on the PCI Express® x4 interface, enables acquisition of video and digital audio streams from both HD (high-definition) and SD (standard-definition) video at up to 170 MHz DVI input.

ADLINK's HDV62A not only delivers superior quality high-definition video data from DVI or HDMI, suitable for medical, scientific imaging and multi media device testing, but also provides an analog video decoder comprehensively supporting CVBS, S-video, analog RGB, and Component signal capture.

Equipped with a FPGA (field programmable gate array) and a 512 MB memory buffer, the HDV62A enables image streaming of specified target areas to a host PC, as well as real-time hardware color space conversion to offload repetitive tasks from the host CPU.

The HDV62A is equipped with the ViewCreator Pro® utility, enabling setup, configuration, testing, and system debugging without requiring any software programming. As well, ADLINK's WDM driver is compatible with Microsoft® Directshow to reduce engineering effort and accelerate time to market.

## 1.2 Features

- Uncompressed 1920 x 1080p, 60 fps video stream
- Support for CVBS, S-video, RGB and component video input
- Support for HDMI and DVI video input
- Support for HDMI, S/PDIF audio input
- PCI Express® x4 interface
- Hardware color spacezRAM frame buffer
- Configurable EDID
- Direct SDK support

This page intentionally left blank.

# 2 Function Library

This chapter describes the API (Application Programming Interface) for the HDV62A, for development of application programs within Visual C++, C#, Visual Basic.Net, Delphi, and Borland C++ Builder.

While the HDV62 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.

DirectShow technologies can alternativerly be utilized, as described in the HDV62A User's Manual, to program applications.

Please note that the API and DirectShow programming cannot be combined to access a single HDV62A card at the same time.

## 2.1 List of Functions

| Category | Function Name |
|---|---|
| Device Control | Hdv62_GetDeviceCount |
| | Hdv62_DeviceOpen |
| | Hdv62_DeviceClose |
| | Hdv62_GetDeviceVendorName |
| | Hdv62_GetDeviceModelName |
| | Hdv62_GetDeviceVersion |
| | Hdv62_GetDeviceFirmwareVersion |
| | Hdv62_GetDriverVersion |
| | Hdv62_GetLibraryVersion |
| | Hdv62_GetDeviceID |
| | Hdv62_DeviceReset |

| Category | Function Name |
|---|---|
| Image Format Control | Hdv62_SetChannel |
| | Hdv62_GetChannel |
| | Hdv62_SetSensorFormat |
| | Hdv62_GetSensorFormat |
| | Hdv62_GetSensorWidth |
| | Hdv62_GetSensorHeight |
| | Hdv62_SetWidth |
| | Hdv62_GetWidth |
| | Hdv62_SetHeight |
| | Hdv62_GetHeight |
| | Hdv62_SetXOffset |
| | Hdv62_GetXOffset |
| | Hdv62_SetYOffset |
| | Hdv62_GetYOffset |
| | Hdv62_SetOutputFormat |
| | Hdv62_GetOutputFormat |
| | Hdv62_SetHDelay |
| | Hdv62_GetHDelay |
| | Hdv62_SetContrast |
| | Hdv62_GetContrast |
| | Hdv62_SetHue |
| | Hdv62_GetHue |
| | Hdv62_SetSaturation |
| | Hdv62_GetSaturation |
| | Hdv62_SetBrightness |
| | Hdv62_GetBrightness |
| | Hdv62_GetDetectedSensorFormat |
| | Hdv62_GetHdmiSensorResolution |
| | Hdv62_GetSDSensorResolution |
| | Hdv62_GetAnalogSensorResolution |
| | Hdv62_GetVideoCapabilities |
| | Hdv62_SetImageOrientation |
| | Hdv62_GetImageOrientation |

| Category | Function Name |
|---|---|
| Event & Callback | Hdv62_SetEventSelector |
| | Hdv62_GetEventSelector |
| | Hdv62_SetEventHandle |
| | Hdv62_GetEventHandle |
| | Hdv62_SetCallbackSelector |
| | Hdv62_GetCallbackSelector |
| | Hdv62_SetCallback |
| Acquisition Control | Hdv62_SetAcquisitionFrameCount |
| | Hdv62_GetAcquisitionFrameCount |
| | Hdv62_AcquisitionStart |
| | Hdv62_AcquisitionStop |
| | Hdv62_OneShot |
| | Hdv62_GetImageStream |
| | Hdv62_GetAcquisitionStatus |
| | Hdv62_GetAcquisitionStatistics |
| | Hdv62_GetSensorStatus |
| | Hdv62_SaveImage |
| | Hdv62_GetAudioStream |
| Others | Hdv62_GetErrorText |
| EDID | Hdv62_SetEdidReadyStatus |
| | Hdv62_GetEdidReadyStatus |
| | Hdv62_SetEdidAccessPermission |
| | Hdv62_GetEdidAccessPermission |
| | Hdv62_SetEdidWriteProtection |
| | Hdv62_GetEdidWriteProtection |
| | Hdv62_SetEdidRomSelector |
| | Hdv62_GetEdidRomSelector |
| | Hdv62_SetEdidRom |
| | Hdv62_GetEdidRom |

| Category | Function Name |
|---|---|
| Audio Format Control | Hdv62_SetAudioInput |
| | Hdv62_GetAudioInput |
| | Hdv62_SetAudioChannels |
| | Hdv62_GetAudioChannels |
| | Hdv62_SetAudioSamplesPerSec |
| | Hdv62_GetAudioSamplesPerSec |
| | Hdv62_SetAudioBitsPerSample |
| | Hdv62_GetAudioBitsPerSample |
| | Hdv62_GetAudioCapabilities |

**Table 2-1: API Functions**

## 2.2 Setting Up the Build Environment

### 2.2.1 Include Files

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

| Include File | |
|---|---|
| Hdv62.h | The header file required for all **C/C++** applications. |
| Hdv62.vb | The function definitions required for all **VB.Net** applications. |
| Hdv62.cs | The function definitions required for all **C#** applications. |

### 2.2.2 Library Files

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

| Library File | |
|---|---|
| Hdv62.lib | Exports API function definitions. Required for all Visual **C/C++** applications.. |
| Hdv62_bcb.lib | Exports API function definitions. Required for all Borland C++ Builder applications. |

### 2.2.3 DLL Files

All applications using API require the following DLL files.

.

| DLL File | |
|---|---|
| Hdv62.dll | Dynamic link library. Required for all applications. |

All files are located in the directory \ADLINK\hdv62\Include

## 2.3 Device Control Functions

### 2.3.1 Device Count

Returns the total number of HDV62A devices in the system, with a maximum of 16 devices detectable.

**Syntax**

**C/C++**

```
int Hdv62_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 devices installed.

**Return Value**

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

### 2.3.2 Device Open

Initializes a specified device. Should be called before other functions except those with no Number parameter.

**Syntax**

**C/C++**

```
int Hdv62_DeviceOpen (UINT Number)
```

**C#**

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

**VB.Net**

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

**Parameter(s)**

*Number*

The number of the device to be opened, with allowed values from 0 to 15.

**Return Value**

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

### 2.3.3 Device Close

Closes the device and releases all allocated resources, this function should be called before terminating the application.

**Syntax**

**C/C++**

```
int Hdv62_DeviceClose (UINT Number)
```

**C#**

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

**VB.Net**

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

**Parameter(s)**

*Number*

The number of the device to be closed, with allowed values from 0 to 15.

**Return Value**

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

### 2.3.4 Device Vendor Name

Retrieves or returns the vendor name.

**Syntax**

**C/C++**

```
int Hdv62_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 a negative value, please refer to Other Functions for return code error information.

**C#**

Return vendor name.

**VB.Net**

Return vendor name.

### 2.3.5 Device Model Name

Retrieves or returns the model name.

**Syntax**

**C/C++**

```
int Hdv62_GetDeviceModelName (UINT Number,
char *Name)
```

**C#**

```
string GetDeviceModelName (unit Number)
```

**VB.Net**

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

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Name*

Points to a user-allocated buffer into which the function copies the model name string, for example, “HDV62A”. The name is NULL-terminated.

**Return Value**

**C/C++**

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

**C#**

Return model name.

**VB.Net**

Return model name.

### 2.3.6 Device Version

Retrieves or returns the version of the hardware device.

**Syntax**

**C/C++**

```
int Hdv62_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 device, with allowed values from 0 to 15.

*Version*

Points to a user-allocated buffer into which the function copies the version string. The version is NULL-terminated.

“A3/A1” is for carrier board plus daughter board configurations, with the former carrier board version and latter daughter board version.

“A3” is for single board configuration.

**Return Value**

**C/C++**

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

**C#**

Return version string.

**VB.Net**

Return version string.

### 2.3.7 Device Firmware Version

Retrieves or returns the version of the firmware.

**Syntax**

**C/C++**

```
int Hdv62_GetDeviceFirmwareVersion (UINT Number, char *Version)
```

**C#**

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

**VB.Net**

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

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Version*

Points to a user-allocated buffer into which the version string is entered in a “Year/Month/Day Hour:Minute” format, such as “2012/11/19 14:22”. The version is NULL-terminated.

**Return Value**

**C/C++**

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

**C#**

Return version string.

**VB.Net**

Return version string.

### 2.3.8 Driver Version

Retrieves or returns the driver version.

**Syntax**

**C/C++**

```
int Hdv62_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 device, with allowed values from 0 to 15.

*Version*

Points to a user-allocated buffer into which the version string is entered, such as “1.2.6.0”. The version is NULL-terminated.

**Return Value**

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

**C#**

Return version string.

**VB.Net**

Return version string.

### 2.3.9 Library Version

Retrieves or returns the library version.

**Syntax**

**C/C++**

```
int Hdv62_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 device, with allowed values from 0 to 15.

*Version*

Points to a user-allocated buffer into which the version string is entered, such as “1.2.2.1”. The version is NULL-terminated.

**Return Value**

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

**C#**

Return version string.

**VB.Net**

Return version string.

### 2.3.10 Device ID

Acquires device card ID.

**Syntax**

**C/C++**

```
int Hdv62_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 device, with allowed values from 0 to 15.

*ID*

Card ID can be set by DIP Switch on the card, with possible values from 0 to 15. Card ID can distinguish cards when multiple cards are installed, with different number settings as shown in the HDV62A User's Manual. Default card ID ignores multiples.

**Return Value**

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

### 2.3.11 Device Reset

Restores the HDV62A to its initial boot state, after which all settings must be reconfigured. Call only when the device behaves abnormally and will not restore. The effect of this function is essentiually that of a reboot in less time.

**Syntax**

**C/C++**

```
int Hdv62_DeviceReset (UINT Number)
```

**C#**

```
int DeviceReset (uint Number)
```

**VB.Net**

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

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

**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 Image Format Control Functions

### 2.4.1 Channel

Sets or retrieves channel of device, based on multi-source inputs, only one of which is available at one time. The desired channel should be selected. Please see the HDV62A User's Manual for source input connections.

**Syntax**

**C/C++**

```
int Hdv62_SetChannel (UINT Number, UINT Chan-
nel)
int Hdv62_GetChannel (UINT Number, UINT &Chan-
nel)
```

**C#**

```
int SetChannel (uint Number, uint Channel)
int GetChannel (uint Number, out uint Channel)
```

**VB.Net**

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

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Channel*

Input Source, with 5 allowed values from 0 to 4, for:

0. Analog RGB signal from DVI-I connector
1. YPbPr signal from external I/O bracket
2. HDMI signal from DVI-I connector
3. Composite signal from external I/O bracket
4. S-Video signal from external I/O bracket

**Return Value**

No error occurs if return value ≥ 0; if a negative value, please refer to Section 2.7 Other Functions for return code error information.

### 2.4.2 Sensor Format

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

**Syntax**

**C/C++**

```
int Hdv62_SetSensorFormat (UINT Number, UINT Format)
int Hdv62_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 device, with allowed values from 0 to 15.

*Format*

Source input format, with values differing by channel as:

Channel 0: Analog RGB signal from DVI-I connector

0: VGA 60 fps (640 x 480)

1: SVGA 60 fps (800 x 600)

2: XVGA 60 fps (1024 x 768)

3: SXVGA 60 fps (1280 x 1024)

4 : UXGA 60fps (1600 x 1200)

5 : WXGA+ 60fps (1280 x 768)

6 : WSXGA+ 60fps (1680 x 1050)

7 : XGA 75fps (1024 x 768)

8 : XGA 85fps (1024 x 768)

9 : SXGA 75fps (1280 x 1024)

10 : VGA 75fps (640 x 480)

11 : VGA 85fps (640 x 480)

12 : SVGA 75fps (800 x 600)

13 : SVGA 85fps (800 x 600)

14 : WXGA+ 75fps (1280 x 768)

15 : WXGA+ 60fps (1280 x 800)

16 : WXGA+ 75fps (1280 x 800)

17 : WXGA+ 85fps (1280 x 800)

18 : WXGA+ 85fps (1280 x 768)

19: SXGA 85 fps (1280 x 1024)

20 : XGA+ 75fps (1152 x 864)

21 : Wide XGA 60fps (1360 x768)

22 : SXGA+ 60fps (1400 x 1050)

23 : SXGA+ 75fps (1400 x 1050)

Channel 1: YPbPr signal from external I/O bracket

0: 525i 30 fps (720 x 480 interlace, in frames per second)

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

2: 525p 60 fps (720 x 480 progressive)

3: 625p 50 fps (720 x 576 progressive)

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

6: 720p 60 fps (1280 x 720 progressive)

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

8: 1080i 30 fps (1920 x 1080 interlace, in frames per second)

9: 1080p 50 fps (1920 x 1080 progressive)

10: 1080p 60 fps (1920 x 1080 progressive)

Channel 2: HDMI signal from DVI-I connector

0: 720p 50 fps YCrCb In (1280 x 720 progressive)

1: 720p 60 fps YCrCb In (1280 x 720 progressive),

2: 1080i 25 fps YCrCb In (1920 x 1080 interlace, in frames per second),

3: 1080i 30 fps YCrCb In (1920 x 1080 interlace, in frames per second),

4: 1080p 25 fps YCrCb In (1920 x 1080 progressive)

5: 1080p 30 fps YCrCb In (1920 x 1080 progressive)

6: 1080p 50 fps YCrCb In (1920 x 1080 progressive)

7: 1080p 60 fps YCrCb In (1920 x 1080 progressive)

8: VGA 60 fps (640 x 480)

9: SVGA 60 fps (800 x 600)

10: XGA 60 fps (1024 x 768)

11: SXGA 60 fps (1280 x 1024)

12: UXGA 60 fps (1600 x 1200)

13: 720p 50 fps RGB In (1280 x 720 progressive)

14: 720p 60 fps RGB In (1280 x 720 progressive)

15: 1080i 25 fps RGB In (1920 x 1080 interlace, in frames per second)

16: 1080i 30 fps RGB In (1920 x 1080 interlace, in frames per second)

17: 1080p 25 fps RGB In (1920 x 1080 progressive)

18: 1080p 30 fps RGB In (1920 x 1080 progressive)

19: 1080p 50 fps RGB In (1920 x 1080 progressive)

20: 1080p 60 fps RGB In (1920 x 1080 progressive)

21: 1080p 24 fps YCrCb In (1920 x 1080 progressive)

22: 1080p 48 fps YCrCb In (1920 x 1080 progressive)

23: WXGA 60 fps YCrCb In (1360 x 768)

24: 1200p 50 fps YCrCb In (1920 x 1200 progressive)

25: 1200p 60 fps YCrCb In (1920 x 1200 progressive)

26: 1080p 24 fps RGB In (1920 x 1080 progressive)

27: 1080p 48 fps RGB In (1920 x 1080 progressive)

28: WXGA 60 fps RGB In (1360 x 768)

29: 1200p 50 fps RGB In (1920 x 120 progressive)

30: 1200p 60 fps RGB In (1920 x 1200 progressive)

31: 525i 30 fps YCrCb In (720 x 480 interlace, in frame per second)

32: 625i 25 fps YCrCb In (720 x 576 interlace, in frame per second)

33: 525i 30 fps RGB In (720 x 480 interlace, in frame per second)

34: 625i 25 fps RGB In (720 x 576 interlace, in frame per second)

35: WXGA+ 60 fps RGB In (1280 x 768)

36: WSXGA+ 60 fps RGB In (1680 x 1050)

37: VGA 60 fps YCrCb In (640 x 480)

38: SVGA 60 fps YCrCb In (800 x 600)

39: XGA 60 fps YCrCb In (1024 x 768)

40: WXGA+ 60 fps YCrCb In (1280 x 768)

41: WXGA 60 fps YCrCb In (1280 x 800)

42: SXGA 60 fps YCrCb In (1280 x 1024)

43: UXGA 60 fps YCrCb In (1600 x 1200)

44: WSXGA+ 60 fps YCrCb In (1680 x 1050)

Channel 3: Composite from external I/O bracket

0: NTSC MJ

1: PAL 60

2: NTSC 4.43

3: PAL BGHID

4: PAL M

5: PAL Nc

Channel 4: S-Video from external I/O bracket

0: NTSC MJ

1: PAL 60

2: NTSC 4.43

3: PAL BGHID

4: PAL M

5: PAL Nc


NOTE:

- Final resolutions of channels 0,1, and 2 can be acquired with GetVideoCapabilities()
- Noise and black images lasting for a few seconds (actual time of noise and black images depends on the source) may occur if the input source is HDMI plus HDCP, and can be prevented by delaying commencing capture for a few seconds after initial connection of the HDMI device

**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 Sensor Width

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

**Syntax**

**C/C++**

```
int Hdv62_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 device, with allowed values from 0 to 15.

*Width*

The image width of source input as shown.

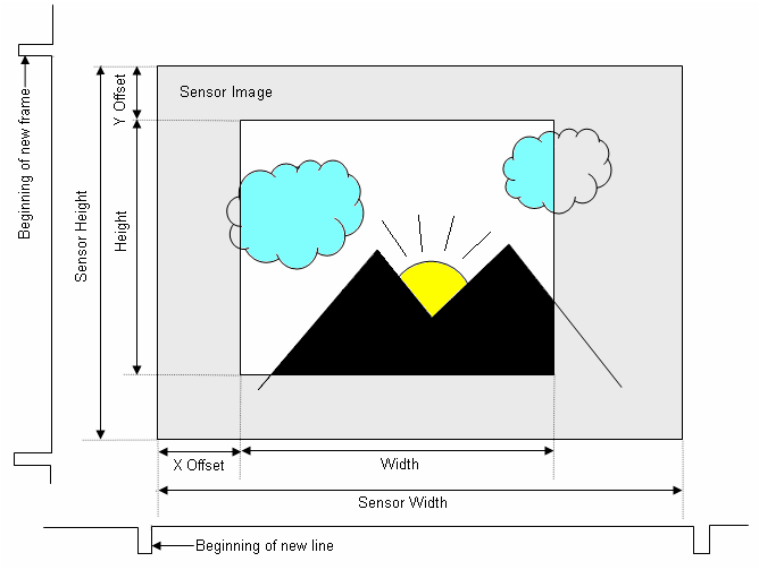


**Figure 2-1: Image Layout**

**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 Sensor Height

Acquires image height of the source input, the same value as shown in Sensor Format.

**Syntax**

**C/C++**

```
int Hdv62_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 device, with allowed values from 0 to 15.

*Height*

The image height of source input as shown.

**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 Width

Sets or retrieves active width of image, can reduce the number of horizontal pixels per line.

**Syntax**

**C/C++**

```
int Hdv62_SetWidth (UINT Number, UINT Width)
int Hdv62_GetWidth (UINT Number, UINT & Width)
```

**C#**

```
int SetWidth (uint Number, uint Width)
int GetWidth (uint Number, out uint Width)
```

**VB.Net**

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

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Width*

Active width of the image as shown, the width coupled with XOffset acting as cropping parameters. The device crops the sensor image if the width is less than SensorWidth, according to:

Width ≦ SensorWidth

Width + XOffset ≦ SensorWidth

Width must be a multiple of 16 pixels

**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 Height

Sets or retrieves active height of image, can reduce the number of lines.

**Syntax**

**C/C++**

```
int Hdv62_SetHeight (UINT Number, UINT Height)
int Hdv62_GetHeight (UINT Number, UINT& Height)
```

**C#**

```
int SetHeight (uint Number, uint Height)
int GetHeight (uint Number, out uint Height)
```

**VB.Net**

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

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Height*

Active height of the image as shown, which when coupled with YOffset, acts as cropping parameters. The device crops the sensor image if the height is less than SensorHeight, according to:

Height <= SensorHeight

Height + YOffset <= SensorHeight

Height is an even number.

YOffset is an even number if sensor source is interlaced.

**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 X Offset

Sets or retrieves X axis image cropping start lines.

**Syntax**

**C/C++**

```
int Hdv62_SetXOffset (UINT Number, UINT XOffset)
int Hdv62_GetXOffset (UINT Number, UINT & XOffset)
```

**C#**

```
int SetXOffset (uint Number, uint XOffset)
int GetXOffset (uint Number, out uint XOffset)
```

**VB.Net**

```
SetXOffset (ByVal Number as UInteger, ByVal XOffset as UInteger) As Integer
GetXOffset (ByVal Number as UInteger, ByRef XOffset as UInteger) As Integer
```

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*XOffset*

Start pixels of image cropping per line as shown, which, when coupled with Width, act as cropping parameters. The device crops sensor image if the XOffset exceeeds 0, according to:

Width <= SensorWidth

Width + XOffset <= SensorWidth

Width must be multiple of 16 pixels

**Return Value**

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

### 2.4.8 Y Offset

Sets or retrieves Y axis image cropping start lines.

**Syntax**

**C/C++**

```
int Hdv62_SetYOffset (UINT Number, UINT YOffset)
int Hdv62_GetYOffset (UINT Number, UINT &YOffset)
```

**C#**

```
int SetYOffset (uint Number, uint YOffset)
int GetYOffset (uint Number, out uint YOffset)
```

**VB.Net**

```
SetYOffset (ByVal Number as UInteger, ByVal YOffset as UInteger) As Integer
GetYOffset (ByVal Number as UInteger, ByRef YOffset as UInteger) As Integer
```

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*YOffset*

Image cropping start lines, as shown, which, coupled with Height, act as cropping parameters. The device crops sensor image if the YOffset exceeds 0, according to:

Height <= SensorHeight

Height + YOffset <= SensorHeight

Height is an even number

YOffset is an even number if sensor source is interlaced

**Return Value**

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

### 2.4.9 Output Format

Sets or retrieves pixel output format.

**Syntax**

**C/C++**

```
int Hdv62_SetOutputFormat (UINT Number, UINT Format)
int Hdv62_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 device, with allowed values from 0 to 15.

*Format*

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

0: 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 |

1: 30bit RGB (BGR30) - 10bit R + 10bit G + 10bit B

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

2: 32bit RGB (RGB32) – 8bit R + 8bit G + 8bit B + 8bit Alpha

| DWORD | Pixel Data [31:0] | | | |
|---|---|---|---|---|
| | [31:24] | [23:16] | [15:8] | [7:0] |
| dw0 | Alpha | R | G | B |

3: 8bit gray scale (Y8) – 8bit Y

| DWORD | Pixel Data [31:0] | | | |
|---|---|---|---|---|
| | [31:24] | [23:16] | [15:8] | [7:0] |
| dw0 | Y3 | Y2 | Y1 | Y0 |

4: 24bit YCbCr 4:4:4 (YUV8) - 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 |

5: 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 |


NOTE:

Actual width of the image setting in a WriteCropping routine must be multiple of 8 and actual height must be an even number

Total bytes of one scan line must align with a multiple of 16. If the requirement is not met, the HDV62A appends dummy bytes to the end of each line, for example, if 20bit YCbCr 4:2:2 is selected and:

- width = 800 pixels, each line = 2144 bytes (11 dummy bytes appended)
- width = 640 pixles, each line = 1712 bytes (5 dummy bytes appended)
- width = 720 pixles, each line = 1920 bytes (0 dummy bytes appended)

Formula: total bytes of each line = ((width * 16 / 6) + 15) & ~15 with all integer calculation

Formula of YCbCr to RGB:

- R = Y + 1.371(Cr-128)
- G = Y – 0.698(Cr-128) – 0.336(Cb-128)
- B = Y + 1.732(Cb-128)

**Return Value**

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

### 2.4.10 HDelay

Sets or retrieves horizontal delay of frame images, similar to X offset, shifting images left or right to remove black vertical lines. HDelay can be applied to HDMI, RGB and YPbPr inputs only. Other inputs ignore the setting.

**Syntax**

**C/C++**

```
int Hdv62_SetHDelay (UINT Number, int Delay)
int Hdv62_GetHDelay (UINT Number, int & Delay)
```

**C#**

```
int SetHDelay (uint Number, int Delay)
int GetHDelay (uint Number, out int Delay)
```

**VB.Net**

```
SetHDelay (ByVal Number as UInteger, ByVal
Delay as Integer) As Integer
GetHDelay (ByVal Number as UInteger, ByRef
Delay as Integer) As Integer
```

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Delay*

The horizontal delay of frame images, with allowed values from -3 to 3. Each resolution has a default value. Call this routine after channel and sensor format have been set.

**Return Value**

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

### 2.4.11 Contrast

Sets or retrieves contrast of input YPbPr frame images.

**Syntax**

**C/C++**

```
int Hdv62_SetContrast (UINT Number, int Value)
int Hdv62_GetContrast (UINT Number, int & Value)
```

**C#**

```
int SetContrast (uint Number, int Value)
int GetContrast (uint Number, out int Value)
```

**VB.Net**

```
SetContrast (ByVal Number as UInteger, ByVal Value as Integer) As Integer
GetContrast (ByVal Number as UInteger, ByRef Value as Integer) As Integer
```

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Value*

Contrast of frame images, with allowed values from 0 to 255 and default value 128.

**Return Value**

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

### 2.4.12 Hue

Sets or retrieves hue of input YPbPr frame images.

**Syntax**

**C/C++**

```
int Hdv62_SetHue (UINT Number, int Value)
int Hdv62_GetHue (UINT Number, int & Value)
```

**C#**

```
int SetHue (uint Number, int Value)
int GetHue (uint Number, out int Value)
```

**VB.Net**

```
SetHue (ByVal Number as UInteger, ByVal Value
as Integer) As Integer
GetHue (ByVal Number as UInteger, ByRef Value
as Integer) As Integer
```

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Value*

Frame image hues, with allowed values from -128 to 127 and default value 0.

**Return Value**

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

### 2.4.13 Saturation

Sets or retrieves saturation of input YPbPr frame images.

**Syntax**

**C/C++**

```
int Hdv62_SetSaturation (UINT Number, int
Value)
int Hdv62_GetSaturation (UINT Number, int &
Value)
```

**C#**

```
int SetSaturation (uint Number, int Value)
int GetSaturation (uint Number, out int Value)
```

**VB.Net**

```
SetSaturation (ByVal Number as UInteger, ByVal
Value as Integer) As Integer
```

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

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Value*

Frame image saturation, with allowed values from 0 to 255 and default value 128.

**Return Value**

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

### 2.4.14 Brightness

Sets or retrieves brightness of frame images.

**Syntax**

**C/C++**

```
int Hdv62_SetBrightness (UINT Number, int
Value)
int Hdv62_GetBrightness (UINT Number, int &
Value)
```

**C#**

```
int SetBrightness (uint Number, int Value)
int GetBrightness (uint Number, out int Value)
```

**VB.Net**

```
SetBrightness (ByVal Number as UInteger, ByVal
Value as Integer) As Integer
GetBrightness (ByVal Number as UInteger, ByRef
Value as Integer) As Integer
```

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Value*

Brightness of frame images, with allowed values from -128 to 127 and default value 0.

**Return Value**

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

### 2.4.15 HdmiSensorResolution

Retrieves the detected HDMI sensor resolution.

**Syntax**

**C/C++**

```
int  Hdv62_GetHdmiSensorResolution(UINT  Number, SENSOR_RESOLUTION& Resolution)
```

**C#**

```
int GetHdmiSensorResolution (uint Number, out SENSOR_RESOLUTION Resolution)
```

**VB.Net**

```
GetHdmiSensorResolution (ByVal Number as UInteger, ByRef Resolution As SENSOR_RESOLUTION) As Integer
```

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*SENSOR_RESOLUTION*

Structure of HDMI sensor resolution defined as:

**C/C++**

typedef struct _SENSOR_RESOLUTION

{

UINT Width;

UINT Height;

UINT FrameRate;

```
UINT Interlace;
} SENSOR_RESOLUTION;
```

**C#**

```
public struct SENSOR_RESOLUTION
{
public uint Width;
public uint Height;
public uint FrameRate;
public uint Interlace;
}
```

**VB.Net**

```
Public Structure SENSOR_RESOLUTION
Public Width As UInteger
Public Height As UInteger
Public FrameRate As UInteger
Public Interlace As UInteger
End Structure
```

NOTE:

SD format (525i and 625i) may exceed the defined width, such as 625i being 720 x 576, but 1440 x 576 is usable from this function.

**Return Value**

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

### 2.4.16 SDSensorResolution

Retrieves the detected resolution of the analog sensor including composite and S-Video signals. When the channel is set to one of the inputs described, after at least 1 second, the signal is detected correctly.

**Syntax**

**C/C++**

```
int  Hdv62_GetSDSensorResolution(UINT  Number,
int& Value)
```

**C#**

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

**VB.Net**

```
GetSDSensorResolution (ByVal Number as UInte-
ger, ByRef Value As Integer) As Integer
```

**Parameter(s)**

*Value*

Read back resolution of the sensor, defined as:

0: NTSC MJ

1: PAL 60

2: NTSC 4.43

3: PAL BGHID

4: PAL M

5: PAL Nc

-1: Not detected

**Return Value**

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

### 2.4.17 AnalogSensorResolution

Acquires the detected resolution of the analog sensor including analog RGB and YPbPr. When the channel is set to one of the decribed inputs, after least 1 second the signal can be detected correctly.

**Syntax**

**C/C++**

```
int Hdv62_GetAnalogSensorResolution(UINT Number, SENSOR_RESOLUTION& Resolution)
```

**C#**

```
int GetAnalogSensorResolution (uint Number, out SENSOR_RESOLUTION Resolution)
```

**VB.Net**

```
GetAnalogSensorResolution (ByVal Number as UInteger, ByRef Resolution As SENSOR_RESOLUTION) As Integer
```

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*SENSOR_RESOLUTION*

Structure of analog sensor resolution, defined as:

**C/C++**

```
typedef struct _SENSOR_RESOLUTION
{
UINT Width;
UINT Height;
UINT FrameRate;
UINT Interlace;
} SENSOR_RESOLUTION;
```

**C#**

```
public struct SENSOR_RESOLUTION
{
public uint Width;
public uint Height;
public uint FrameRate;
```

public uint Interlace;

}

**VB.Net**

Public Structure SENSOR_RESOLUTION

Public Width As UInteger

Public Height As UInteger

Public FrameRate As UInteger

Public Interlace 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.

### 2.4.18 Video Capabilities

Acquires list of resolutions the card supports.

**Syntax**

**C/C++**

```
int   Hdv62_GetVideoCapabilities(UINT   Number,
RESOLUTION_CAPABILITIES & Caps)
```

**C#**

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

**VB.Net**

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

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*RESOLUTION_CAPABILITIES*

Supported resolutions defined as:

**C/C++**

```
typedef struct _SENSOR_PROPERTIES
{
    char Name[ 256 ];
    unsigned long Width;
    unsigned long Height;
    unsigned long FrameRate;
    unsigned long Interlace;
    unsigned long Hf;
    unsigned long HTotal;
    unsigned long Hsw;
    unsigned long Hbp;
    unsigned long Vf;
    unsigned long VTotal;
    unsigned long Vsw;
    unsigned long Vbp;
} SENSOR_PROPERTIES;

typedef struct _RESOLUTION_CAPABILITIES
{
    unsigned long NumRgbResolution;
    SENSOR_PROPERTIES *RgbResolutions;
    unsigned long NumYPbPrResolution;
    SENSOR_PROPERTIES *YPbPrResolutions;
    unsigned long NumHdmiResolution;
```

```
    SENSOR_PROPERTIES *HdmiResolutions;
} RESOLUTION_CAPABILITIES;
```

**C#**

```
struct SENSOR_PROPERTIES
{
    // name
MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
    public string Name;

    // video setting
    public uint Width;
    public uint Height;
    public uint FrameRate;
    public uint Interlace;

    // video timing
    public uint Hf;// horizontal frequency
    public uint HTotal;// horizontal total line
    public uint Hsw;// horizontal sync width
    public uint Hbp;// horizontal back porch
    public uint Vf;// vertical frequency
    public uint VTotal;// vertical total line
    public uint Vsw;// vertical sync width
    public uint Vbp;// vertical back porch
}

struct RESOLUTION_CAPABILITIES
```

```
{
    public uint NumRgbResolution;
    public IntPtr RgbResolutions;

    public uint NumYPbPrResolution;
    public IntPtr YPbPrResolutions;

    public uint NumHdmiResolution;
    public IntPtr HdmiResolutions;
}
```

**VB.Net**

```
Structure SENSOR_PROPERTIES

                    <MarshalAs(UnmanagedType.ByValTStr,
SizeConst:=256)> _
    Dim Name As String
    Dim Width As UInteger
    Dim Height As UInteger
    Dim FrameRate As UInteger
    Dim Interlace As UInteger

    Dim Hf As UInteger      ' horizontal frequency
    Dim HTotal As UInteger  ' horizontal total line
    Dim Hsw As UInteger     ' horizontal sync width
    Dim Hbp As UInteger     ' horizontal back porch
    Dim Vf As UInteger      ' vertical frequency
    Dim VTotal As UInteger  ' vertical total line
```

```
    Dim Vsw As UInteger     ' vertical sync width
    Dim Vbp As UInteger     ' vertical back porch

End Structure

Structure RESOLUTION_CAPABILITIES

    Dim NumRgbResolution As UInteger
    Dim RgbResolutions As IntPtr

    Dim NumYPbPrResolution As UInteger
    Dim YPbPrResolutions As IntPtr

    Dim NumHdmiResolution As UInteger
    Dim HdmiResolutions 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.

### 2.4.19 Image Orientaton

Sets or retrieves orientation of images arranged in memory.

**Syntax**

**C/C++**

```
int  Hdv62_SetImageOrientation  (UINT  Number,
UINT Value)
int  Hdv62_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 device, with allowed values from 0 to 15.

*Value*

Indicates the image orientation, from among:

0: Bottom-up, in which the image buffer starts with the bottom row of pixels, followed by the next row up, and so forth, 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. Physical layout of a bottom-up image is as shown.

E.g. Color space = RGB24

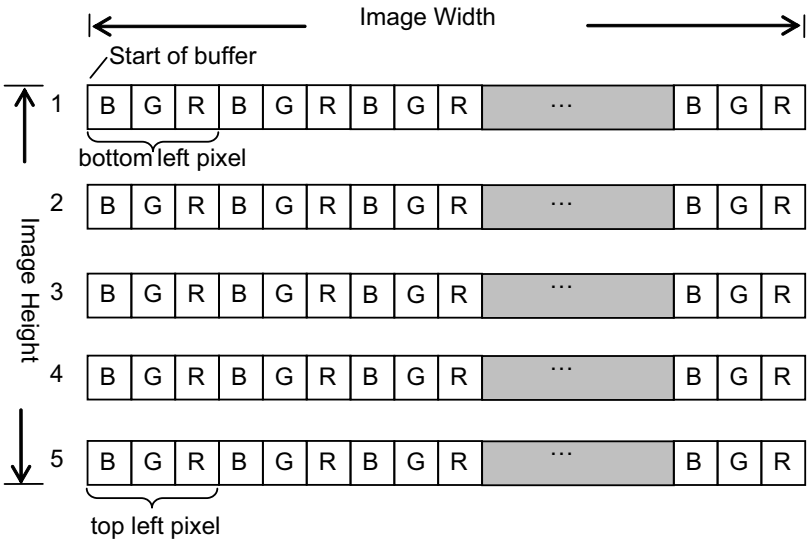


**Figure 2-2: Bottom-up Image Orientation**

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

E.g. Color space = RGB24

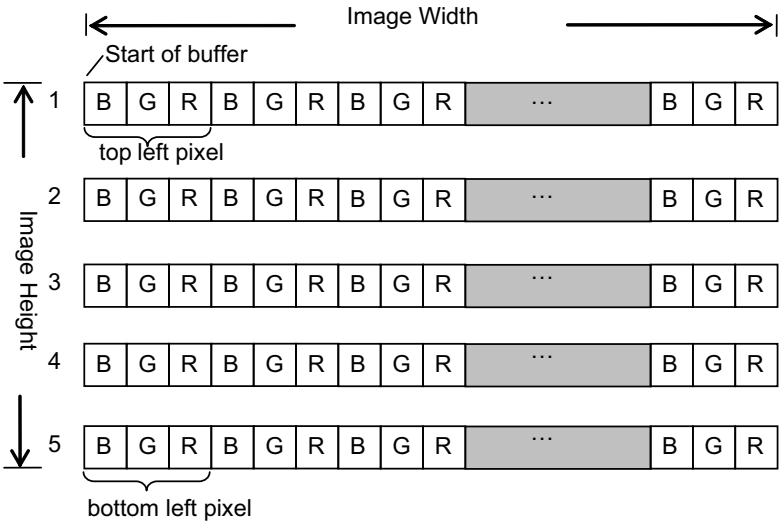


**Figure 2-3: Top-down Image Orienation**

**Return Value**

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

## 2.5 Event and Callback Functions

### 2.5.1 Event Selector

Sets or retrieves the event type.

**Syntax**

**C/C++**

```
int Hdv62_SetEventSelector (UINT Number, UINT
Mode)
int Hdv62_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 device, with allowed values from 0 to 15.

*Mode*

Event type, comprising frame and audio events, with frame events the result of a 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: Frame event

2: Audio event

And SetEventHandle sets 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.

### 2.5.2 Event Handle

Sets or retrieves event handle. While event and callback are the most frequently used methods determining when to recover frame or audio information, only one is normally allowed, but here both can be set.

**Syntax**

**C/C++**

```
int Hdv62_SetEventHandle (UINT Number, HANDLE Handle)
```

```
int Hdv62_GetEventHandle (UINT Number, HANDLE
&Handle)
```

**C#**

```
int SetEventHandle (uint Number, IntPtr Han-
dle)

int SetEventHandle (uint Number, SafeWaitHan-
dle Handle)

int GetEventHandle (uint Number, out IntPtr
Handle)

int GetEventHandle (uint Number, out SafeWait-
Handle 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 device, with allowed values from 0 to 15.

*Handle*

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

- **GetImageStream** to acquire the pointer of the 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.

### 2.5.3 CallbackSelector

Sets or retrieves callback function type.

**Syntax**

**C/C++**

```
int  Hdv62_SetCallbackSelector  (UINT  Number,
UINT Mode)
int  Hdv62_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 device, with allowed values from 0 to 15.

*Mode*

Callback type, from among frame callback,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: Frame callback

2: Audio callback

**Return Value**

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

### 2.5.4 Callback

Sets or retreives callback handle. While event and callback are the most frequently used methods determining when to recover frame or audio information, only one is normally allowed, but here both can be set.

**Syntax**

**C/C++**

```
int Hdv62_SetCallback (UINT Number, HDV62CALLBACK Fun)
```

**C#**

```
int SetCallback (uint Number, HDV62CALLBACK Fun)
```

**VB.Net**

```
SetCallback (ByVal Number as UInteger, ByVal Fun as HDV62CALLBACK) As Integer
```

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*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 the frame buffer
- ▷ **GetAudioStream** to acquire audi 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.

## 2.6 Acquisition Control Functions

### 2.6.1 Acquisition Frame Count

Sets or retrieves the number of frames to be simulatane-ously captured. Call:

- **AcquisitionStart** to initiate capture
- **GetAcquisitionStatus** to retreive acquisition state
- **AcquisitionStop** to terminate capture

**Syntax**

**C/C++**

```
int Hdv62_SetAcquisitionFrameCount (UINT Num-
ber, UINT Count)
int Hdv62_GetAcquisitionFrameCount (UINT Num-
ber, 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 device, with allowed values from 0 to 15.

*Count*

The frame count to be captured, from among:

- 0: Capture until AcquisitionStop is called.
- >0: Acquires the desired frame count, and, when reached, acquisition status is changed to 0 (stopped). 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.

### 2.6.2 Acquisition Start

Initiates frame capture

**Syntax**

**C/C++**

```
int Hdv62_AcquisitionStart (UINT Number)
```

**C#**

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

**VB.Net**

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

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

**Return Value**

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


Noise and black images lasting for a few seconds (actual time of noise and black images depends on the source) may occur if the input source is HDMI plus HDCP, and can be prevented by delaying commencing capture for a few seconds after initial connection of the HDMI device

### 2.6.3 Acquisition Stop

Terminates frame capture

**Syntax**

**C/C++**

```
int Hdv62_AcquisitionStop (UINT Number)
```

**C#**

```
int AcquisitionStop (uint Number)
```

**VB.Net**

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

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

**Return Value**

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

### 2.6.4 One Shot

Acquires a single 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 and/or GetAudioStream retreives the frame image pointer.

**Syntax**

**C/C++**

```
int Hdv62_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 device, with allowed values from 0 to 15.

*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.

### 2.6.5 Image Stream

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

**Syntax**

**C/C++**

```
int Hdv62_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 device, with allowed values from 0 to 15.

*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.

### 2.6.6 Acquisition Status

Retreives current status of the image acquisition.

**Syntax**

**C/C++**

```
int Hdv62_GetAcquisitionStatus (UINT Number,
UINT &Status)
```

**C#**

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

**VB.Net**

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

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*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.

### 2.6.7 Acquisition Statistics

Acquires the number of frames captured since Acquisition Start.

**Syntax**

**C/C++**

```
int Hdv62_GetAcquisitionStatistics (UINT Num-
ber, 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 device, with allowed values from 0 to 15.

*Count*

Number 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.

### 2.6.8 Sensor Status

Determines whether the device is connected to a suitable sensor.

**Syntax**

**C/C++**

```
int Hdv62_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 device, with allowed values from 0 to 15.

*Locked*

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

- ▷ 0: No proper signal is detected
- ▷ 1: A proper signal is detected


NOTE:

- ▶ For Analog RGB and YPbPr inputs, Locked = 1 is returned only if the detected sensor format is that selected
- ▶ For HDMI input, Locked = 1 if the sensor is a valid HDMI source
- ▶ For composite (or S-Video) input, Locked = 1 if the sensor is a valid composite (or S-Video) source

**Return Value**

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

### 2.6.9 Save Image

Saves the contents of the last image buffer as an image or raw data file, depending on filename.

**Syntax**

**C/C++**

```
int Hdv62_SaveImage (UINT Number, LPTSTR File-
Name)
```

**C#**

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

**VB.Net**

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

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*FileName*

The library supports:

- ▷ BMP: with extension *.bmp
- ▷ JPEG: with extension *.jpg or *.jpeg
- ▷ JIFF: with extension *.tif
- ▷ PNG: with extension *.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.

### 2.6.10 Audio Stream

Acquires audio data pointer and size, normally called in callback after waiting for a frame event, or after calling OneShot.

**Syntax**

**C/C++**

```
int Hdv62_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 device, with allowed values from 0 to 15.

*StreamInfo*

Audio data structure 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.

## 2.7 Other Functions

### 2.7.1 Error Text

Retreives error text string

**Syntax**

**C/C++**

```
int Hdv62_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 bytes must be allocated

**Return Value**

**C/C++**

Always return 0

**C#**

Return the error text

**VB.Net**

Return the error text

## 2.8 EDID Functions

### 2.8.1 Ready Status

Sets or retrieves ready status of the EDID ROM.

**Syntax**

**C/C++**

```
int  Hdv62_SetEdidReadyStatus  (UINT  Number,
UINT Status)
int  Hdv62_GetEdidReadyStatus  (UINT  Number,
UINT& Status)
```

**C#**

```
int SetEdidReadyStatus (uint Number, uint Sta-
tus)
int GetEdidReadyStatus (uint Number, out uint
Status)
```

**VB.Net**

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

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

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Status*

Indicates ready status of the EDID ROM, can be read by external device through DVI-I connector, with some external devices capable of resolution auto-adjustment accordingly. Content can be configured and EDID ROM set to ready state, based on:

0: EDID ROM is not ready

1: EDID ROM is ready

**Return Value**

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


NOTE:

The HDMI/DVI source must be disconnected before writing the EDID.

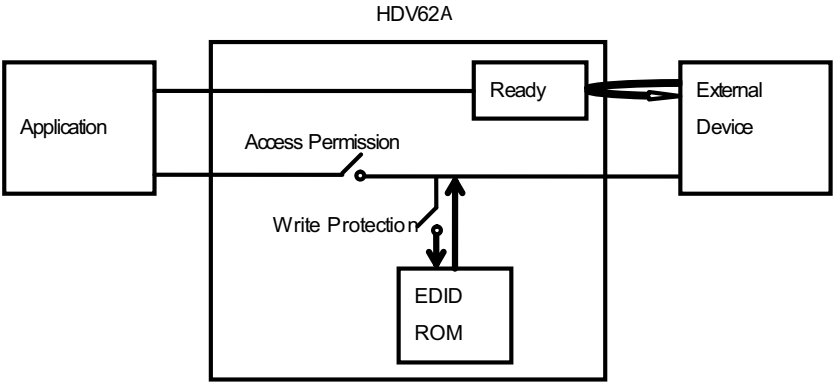


**Figure 2-4: EDID ROM Architecture**

### 2.8.2 Access Permission

Sets or retrieves application access to the EDID ROM.

**Syntax**

**C/C++**

```
int Hdv62_SetEdidAccessPermission (UINT Number, UINT Status)
int Hdv62_GetEdidAccessPermission (UINT Number, UINT& Status)
```

**C#**

```
int SetEdidAccessPermission (uint Number, uint Status)
int GetEdidAccessPermission (uint Number, out uint Status)
```

**VB.Net**

```
SetEdidAccessPermission (ByVal Number As UInteger, ByVal Status As UInteger) As Integer
GetEdidAccessPermission (ByVal Number As UInteger, ByRef Status As UInteger) As Integer
```

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Status*

Indicates accessibility of the EDID ROM. Since access can be obtained by application or external device at the same time, to open EDID:

Establish access permission

Configure EDID ROM

Close access permission

Connect external device via DVI-I

From among the following values:

0: EDID ROM is not accessible

1: EDID ROM is accessible

**Return Value**

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

### 2.8.3 Write Protection

Sets or retrieves writability of the EDID ROM.

**Syntax**

**C/C++**

```
int Hdv62_SetEdidWriteProtection (UINT Number,
UINT Status)
int Hdv62_GetEdidWriteProtection (UINT Number,
UINT& Status)
```

**C#**

```
int SetEdidWriteProtection (uint Number, uint
Status)
int GetEdidWriteProtection (uint Number, out
uint Status)
```

**VB.Net**

```
SetEdidWriteProtection (ByVal Number As UInte-
ger, ByVal Status As UInteger) As Integer

GetEdidWriteProtection (ByVal Number As UInte-
ger, ByRef Status As UInteger) As Integer
```

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Status*

Indicates writeability of the EDID ROM. Any write protection must be cleared bfore data can be saved to the EDID. From among the following values:

0: EDID ROM is writeable

1: EDID ROM is protected

**Return Value**

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

### 2.8.4 ROM Selector

Sets or retrieves the offset of the currently accessed EDID ROM.

**Syntax**

**C/C++**

```
int  Hdv62_SetEdidRomSelector  (UINT  Number,
UINT Offset)

int  Hdv62_GetEdidRomSelector  (UINT  Number,
UINT& Offset)
```

**C#**

```
int SetEdidRomSelector (uint Number, uint Off-
set)

int GetEdidRomSelector (uint Number, out uint
Offset)
```

**VB.Net**

```
SetEdidRomSelector (ByVal Number As UInteger,
ByVal Offset As UInteger) As Integer
GetEdidRomSelector (ByVal Number As UInteger,
ByRef Offset As UInteger) As Integer
```

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Offset*

Indicates the offset of the EDID ROM, with allowed values from 0 to 255.

**Return Value**

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

### 2.8.5 ROM Read/Write

Sets or retrieves value of the EDID ROM.

**Syntax**

**C/C++**

```
int Hdv62_SetEdidRom (UINT Number, UINT Value)
int Hdv62_GetEdidRom (UINT Number, UINT&
Value)
```

**C#**

```
int SetEdidRom (uint Number, uint Value)
int GetEdidRom (uint Number, out uint Value)
```

**VB.Net**

```
SetEdidRom (ByVal Number As UInteger, ByVal
Value As UInteger) As Integer
GetEdidRom (ByVal Number As UInteger, ByRef
Value As UInteger) As Integer
```

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Value*

Indicates the value of the EDID ROM, with allowed values from 0 to 255.

**Return Value**

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

## 2.9 Audio Format Control Functions

### 2.9.1 Audio Input

Sets or retrieves audio input configuration, only one of which is available at one time. Please see the HDV62A User's Manual for connection details.

**Syntax**

**C/C++**

```
int Hdv62_SetAudioInput (UINT Number, UINT Channel)
int Hdv62_GetAudioInput (UINT Number, UINT& Channel)
```

**C#**

```
int SetAudioInput (uint Number, uint Channel)
int GetAudioInput (uint Number, out uint Channel)
```

**VB.Net**

```
SetAudioInput (ByVal Number As UInteger, ByVal Channel As UInteger) As Integer
GetAudioInput (ByVal Number As UInteger, ByRef Channel As UInteger) As Integer
```

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Channel*

Audio source configuration, from among:

0: HDMI audio

1: SPDIF audio (Toslink)

2: SPDIF audio (RCA)

**Return Value**

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

### 2.9.2 Audio Channels

Sets or retrieves the number of channels of the audio input.

**Syntax**

**C/C++**

```
int Hdv62_SetAudioChannels (UINT Number, UINT Channels)
int Hdv62_GetAudioChannel (UINT Number, UINT& Channels)
```

**C#**

```
int SetAudioChannels (uint Number, uint Channels)
int GetAudioChannels (uint Number, out uint Channels)
```

**VB.Net**

```
SetAudioChannels (ByVal Number As UInteger, ByVal Channels As UInteger) As Integer
GetAudioChannels (ByVal Number As UInteger, ByRef Channels As UInteger) As Integer
```

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Channels*

Specifies the number of channels of audio data, according to:

Stereo audio: 2

5.1 channel audio: 6

7.1 channel audio: 8

Currently SPDIF audio only supports up to 2 channels.

**Return Value**

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

### 2.9.3 Samples Per Second

Sets or receives the sample frequency.

**Syntax**

**C/C++**

```
int Hdv62_SetAudioSamplesPerSec (UINT Number, UINT Value)
int Hdv62_GetAudioSamplesPerSec (UINT Number, UINT& Value)
```

**C#**

```
int SetAudioSamplesPerSec (uint Number, uint Value)
int GetAudioSamplesPerSec (uint Number, out uint Value)
```

**VB.Net**

```
SetAudioSamplesPerSec (ByVal Number As UInteger, ByVal Value As UInteger) As Integer
```

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

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*Value*

Specifies sample frequency for each channel, such as 48000 when the sample frequency is 48kHz

**Return Value**

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

### 2.9.4 Bits Per Sample

Sets or retrieves the number of bits per sample.

**Syntax**

**C/C++**

```
int Hdv62_SetAudioBitsPerSample (UINT Number,
UINT Value)
int Hdv62_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 device, with allowed values from 0 to 15.

*Value*

Specifies the number of bits per sample, which the hardware extends to 24 bits if the audio source is 16 or 20 bit input.

**Return Value**

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

### 2.9.5 Audio Capabilties

Sets or retrieves the supported audio properties of the card.

**Syntax**

**C/C++**

```
int Hdv62_GetAudioCapabilities (UINT Number,
AUDIO_CAPABILITIES &Caps)
```

**C#**

```
int GetAudioCapabilities (uint Number, out
AUDIO_CAPABILITIES Caps)
```

**VB.Net**

```
GetAudioCapabilities (ByVal Number As UInte-
ger, ByRef Caps As AUDIO_CAPABILITIES) As
Integer
```

**Parameter(s)**

*Number*

The number of the device, with allowed values from 0 to 15.

*AUDIO_CAPABILITIES*

Structure defined as:

**C/C++**

```
typedef struct _AUDIO_FORMATS
{
   unsigned long NumChannels;
```

```
    unsigned long *Channels;
    unsigned long NumBitsPerSample;
    unsigned long *BitsPerSamples;
    unsigned long NumSamplesPerSec;
    unsigned long *SamplesPerSecs;
} AUDIO_FORMATS;

typedef struct _AUDIO_CAPABILITIES
{
    AUDIO_FORMATS HdmiAudioFormats;
    AUDIO_FORMATS Spdif1AudioFormats;
    AUDIO_FORMATS Spdif2AudioFormats;
} AUDIO_CAPABILITIES;
```

**C#**

```
struct AUDIO_FORMATS
{
    public uint NumChannels;
    public IntPtr Channels;
    public uint NumBitsPerSample;
    public IntPtr BitsPerSamples;
    public uint NumSamplesPerSec;
    public IntPtr SamplesPerSecs;
}

struct AUDIO_CAPABILITIES
{
    public AUDIO_FORMATS HdmiAudioFormats;
    public AUDIO_FORMATS Spdif1AudioFormats;
```

```
    public AUDIO_FORMATS Spdif2AudioFormats;
}
```

**VB.Net**

```
Structure AUDIO_FORMATS
    Dim NumChannels As UInteger
    Dim Channels As IntPtr
    Dim NumBitsPerSample As UInteger
    Dim BitsPerSamples As IntPtr

    Dim NumSamplesPerSec As UInteger
    Dim SamplesPerSecs As IntPtr
End Structure
Public Structure AUDIO_CAPABILITIES
Dim HdmiAudioFormats As AUDIO_FORMATS
Dim Spdif1AudioFormats As AUDIO_FORMATS
Dim Spdif2AudioFormats As AUDIO_FORMATS
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.

# 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