vimba

AVT SOFTWARE DEVELOPMENT KIT

AVT Vimba

User Guide for Windows

V1.2
2013-Jun-25

Allied Vision Technologies GmbH
Taschenweg 2a
D-07646 Stadtroda / Germany

ALLIED Vision Technologies

Contents

List of Figures

1 Contacting Allied Vision Technologies

Note

- **Technical Information**
  http://www.alliedvisiontec.com
- **Support**
  support@alliedvisiontec.com

Allied Vision Technologies GmbH (Headquarters)
Taschenweg 2a
07646 Stadtroda, Germany
Tel.: +49 36428-677-0
Fax.: +49 36428-677-28
Email: info@alliedvisiontec.com

Allied Vision Technologies Canada Inc.
101-3750 North Fraser Way
Burnaby, BC, V5J 5E9, Canada
Tel: +1 604-875-8855
Fax: +1 604-875-8856
Email: info@alliedvisiontec.com

Allied Vision Technologies Inc.
38 Washington Street
Newburyport, MA 01950, USA
Toll Free number +1 877-USA-1394
Tel.: +1 978-225-2030
Fax: +1 978-225-2029
Email: info@alliedvisiontec.com

Allied Vision Technologies Asia Pte. Ltd.
82 Playfair Road
#07-02 D'Lithium
Singapore 368001
Tel. +65 6634-9027
Fax:+65 6634-9029
Email: info@alliedvisiontec.com

Allied Vision Technologies (Shanghai) Co., Ltd.
2-2109 Hongwell International Plaza
1602# ZhongShanXi Road
Shanghai 200235, China
Tel: +86 (21) 64861133
Fax: +86 (21) 54233670
Email: info@alliedvisiontec.com

2 Introduction

2.1 Document history

Version	Date	Changes
1.0	2012-Nov-26	Initial version
1.1	2013-Apr-03	Different links, small changes
1.2	2013-Jun-18	Added chapter for Class Generator, small corrections, layout changes

2.2 Conventions used in this manual

To give this manual an easily understood layout and to emphasize important information, the following typographical styles and symbols are used:

2.2.1 Styles

Style	Function	Example
Bold	Programs, inputs or highlighting important things	**bold**
Courier	Code listings etc.	`Input`
Upper case	Constants	CONSTANT
Italics	Modes, fields	*Mode*
Parentheses and/or blue	Links	(Link)

2.2.2 Symbols

Note

This symbol highlights important information.

Caution

This symbol highlights important instructions. You have to follow these instructions to avoid malfunctions.

www

This symbol highlights URLs for further information. The URL itself is shown in blue.
Example: http://www.alliedvisiontec.com

3 Vimba SDK Overview

The Vimba SDK is a camera-independent SDK for the 32-bit and 64-bit operating systems Windows XP, Windows 7, Windows 8, and various Linux distributions that can be applied for both AVT 1394 (Windows only) and AVT GigE Vision cameras. With the Vimba SDK, your application immediately supports AVT's 1394a / 1394b digital cameras (Windows only) as well as AVT's GigE Vision cameras.

3.1 Architecture

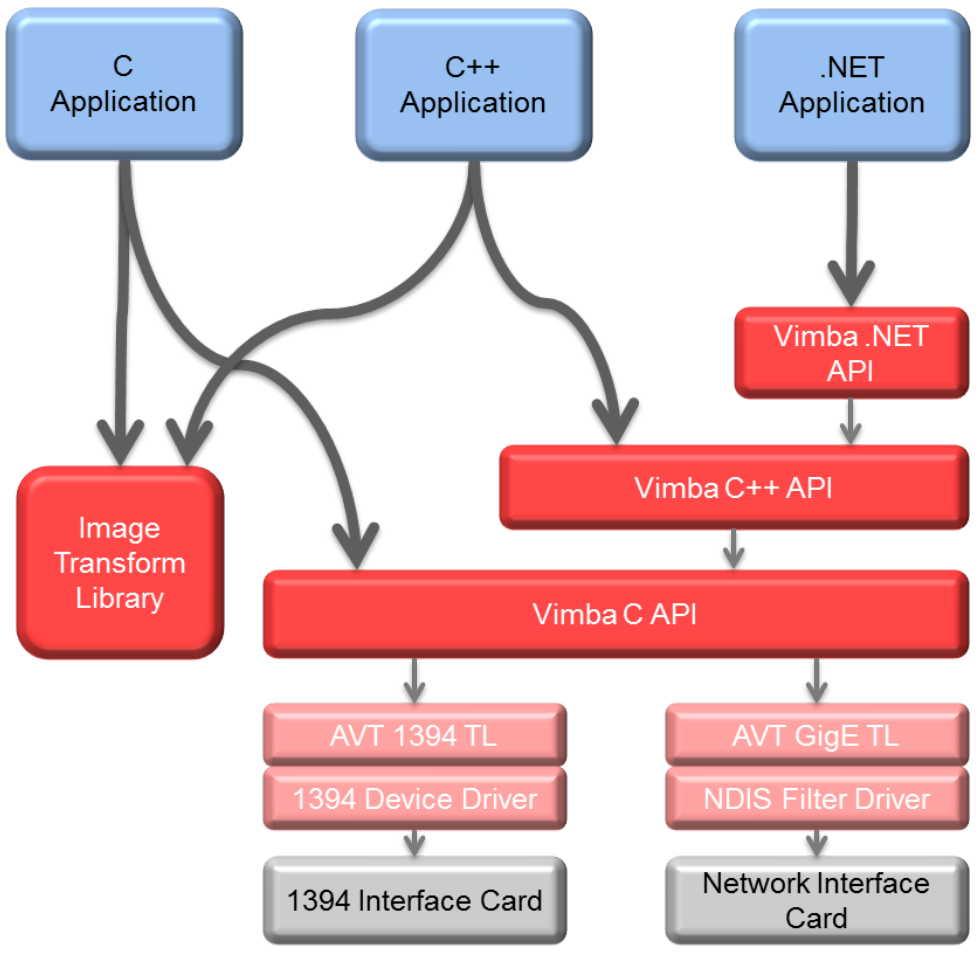


Figure 1: Vimba Architecture

The Vimba SDK provides application programming interfaces (APIs) for different programming languages: C, C++, and .NET. With the .NET API you can use any of the .NET programming languages like C#, C++/cli, or Visual Basic .NET. All APIs cover the following functions:

- listing currently connected cameras
- controlling camera features
- receiving images from the camera
- notifications about cameras connecting or disconnecting

The Image Transform Library converts camera images into other pixel formats and creates color images from raw images (debayering). While this is separated for the C and C++ API, the .NET API includes these functions. Therefore, a .NET application doesn't have to access the Image Transform Library.

The APIs use transport layer (TL) modules to actually communicate with the cameras. These modules (AVT 1394 TL and AVT GigE TL) are not directly accessible for the user application.

For more detailed information about the different APIs please look at the documents listed in section **??**.

3.2 API Object Model

The Vimba APIs use a defined object model for providing access to the different entities. For object oriented programming languages like C++, this object model is reflected in the API's class design, but even the C API supports this model by using handles as a representation of the different objects.

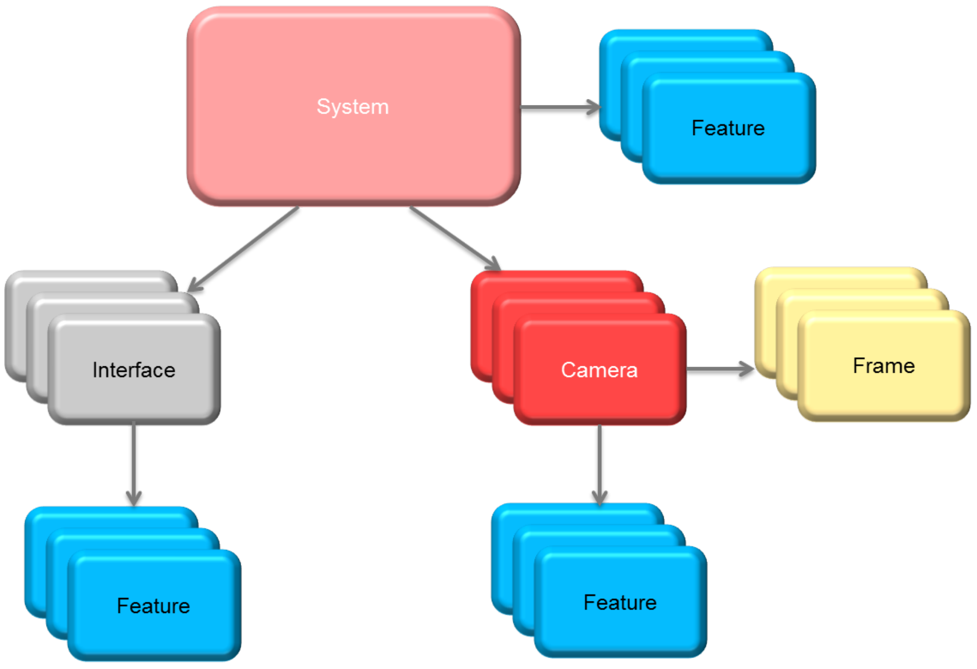


Figure 2: Vimba API Object Model

The *System* object represents the API itself. Thus only one instance of it is available. The application has to initialize the *System* object before using any other function. When the application has finished using the API, it shuts it down through the *System* object. The *System* object holds a list of interfaces and cameras internally and serves as the main access point to these objects.

A *Camera* object controls a physical camera and receives images. It provides the same set of functions regardless of the underlying interface technology.

An *Interface* object represents a port on a physical interface card in the PC. Although a camera is connected through an interface, it is not necessary to use an *Interface* object to access a *Camera* object. This can be done directly via the *System* object. So the only purpose of the *Interface* object is to control the settings of the corresponding interface card.

All these objects - *System*, *Camera*, and *Interface* - have a list of *Feature* objects. They reflect the settings of these entities. The *System Features* contain information about API wide settings like what kind of transport layer modules are loaded. The *Camera Features* can be used to configure camera settings like exposure time or pixel format. *Interface Features* represent the settings of a physical interface card in the PC like an IP address of a network interface card. *Camera* and *Interface Features* are usually different according to the underlying interface technology.

Frame objects receive image data from the camera. They are created by the application and queued at the corresponding *Camera* object. When an image was received, the next available *Frame* is filled and handed over to the application through a dedicated notification. After the application processed the image data, it should return the *Frame* to the API by re-enqueuing it at the corresponding *Camera*.

3.3 Notifications

In general, a Vision system consisting of cameras and PCs is asynchronous, which means that certain events usually occur unexpectedly. This includes - among others - detection of cameras connected to the PC or frame reception. A Vimba application can react on a particular event by registering a corresponding handler function at the API, which in return will be called when the event occurs. The exact method how to register an event handler depends on the used programming language. Have a look at the example programs for more details.

Caution

The registered functions are usually called from a different thread than the application. So extra care must be taken when accessing data shared between these threads (multithreading environment).
Furthermore, the Vimba API might be blocked while the event handler is executed. Thus, it is highly recommended to exit the event handler function as fast as possible.
Not all API functions may be called from the event handler function. For more details, see the Programmer's Reference document for the programming language of your choice.

4 Vimba Class Generator

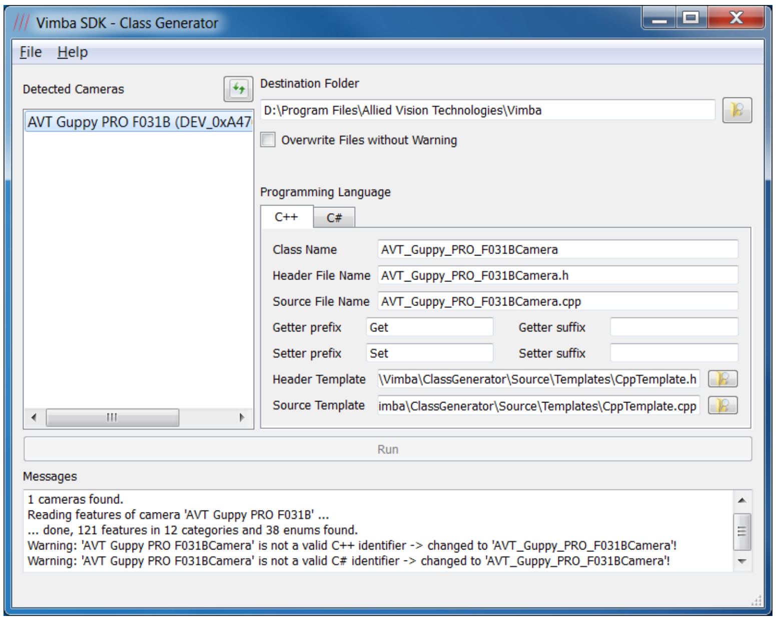
The Vimba Class Generator is a tool for easily creating classes for Vimba C++ (Windows and Linux) and Vimba.NET (Windows only) APIs that are more comfortable to use than the standard API. The generated classes offer access functions for each found feature, depending on the type of the feature.

Note

After a firmware update, re-generate the files and merge the access functions for new features manually into your previously generated code.

4.1 Main window

Upon startup the Vimba Class Generator assembles all the cameras accessible by Vimba.



Destination Folder: D:\Program Files\Allied Vision Technologies\Vimba
Overwrite Files without Warning

Programming Language: C++ | C#

Class Name	AVT_Guppy_PRO_F031BCamera		
Header File Name	AVT_Guppy_PRO_F031BCamera.h		
Source File Name	AVT_Guppy_PRO_F031BCamera.cpp		
Getter prefix	Get	Getter suffix	
Setter prefix	Set	Setter suffix	
Header Template	\Vimba\ClassGenerator\Source\Templates\CppTemplate.h		
Source Template	imba\ClassGenerator\Source\Templates\CppTemplate.cpp		

```
1 cameras found.
Reading features of camera 'AVT Guppy PRO F031B' ...
... done, 121 features in 12 categories and 38 enums found.
Warning: 'AVT Guppy PRO F031BCamera' is not a valid C++ identifier -> changed to 'AVT_Guppy_PRO_F031BCamera'!
Warning: 'AVT Guppy PRO F031BCamera' is not a valid C# identifier -> changed to 'AVT_Guppy_PRO_F031BCamera'!
```

Figure 3: Vimba Class Generator - Main Window

The available cameras are listed in the box on the left side of the screen below *Detected cameras*. Select the camera that you want code generated for.

Note

You may click the Refresh button above the listed cameras at any time to re-scan for new cameras.

In the edit field below *Destination Folder*, you may enter the folder where the files will be generated. By clicking the associated button, you may select a folder from the file system.

The Vimba Class Generator will not overwrite existing files without warning. By selecting the check-box *Overwrite Files without Warning*, you may change this behavior.

The options below *Programming Language* allow you to configure the code generation. By selecting the tab, you may switch the programming language for creation. For more language-dependent options, see chapters C++ code generation and C# code generation.

If everything is configured, the *Run* button becomes active. Clicking it will generate the code for the selected camera, programming language and options.

Any message that is generated during selection of the options or during code generation is displayed in the text box at the bottom of the window.

4.2 C++ code generation

In the *C++* tab of the main window, you have the following options

- Class Name: The name of the generated class.
- Header File Name: The name of the header file to create.
- Source File Name: The name of the cpp file to create.
- Getter prefix: The text that is put before the feature name for each getter function.
- Getter suffix: The text that is put after the feature name for each getter function.
- Setter prefix: The text that is put before the feature name for each setter function.
- Setter suffix: The text that is put after the feature name for each setter function.
- Header template: The file that is used as a template for generating the header file.
- Source template: The file that is used as a template for generating the cpp file.

The template file for the header file may contain the following placeholders:

- ### HEADER_FILE_MACRO_NAME ###: Generated from the *Header File Name* in the main window.
- ### CLASS_NAME ###: Corresponds to *Class Name* in the main window.
- ### ENUM_DECLARATIONS ###: This is where the enum declarations are put.
- ### METHOD_DECLARATIONS ###: This is where the method declarations are put.
- ### VARIABLE_DECLARATIONS ###: This is where the variable declarations are put.

The template file for the cpp file may contain the following placeholders:

- ### HEADER_FILE_NAME ###: Corresponds to *Header File Name* in the main window.
- ### CLASS_NAME ###: Corresponds to *Class Name* in the main window.
- ### METHOD_IMPLEMENTATIONS ###: This is where the method implementations are put.

You may move around these variables in the template file to generate a file that better suits your requirements.

4.3 C# code generation

In the *C#* tab of the main window, you have the following options

- Namespace: The namespace of the generated class.
- Class Name: The name of the generated class.
- Class File Name: The name of the class file to create.
- Template File: The file that is used as a template for generating the class file.

The template file for the header file may contain the following placeholders:

- ### NAMESPACE_NAME ###: Generated from the *Namespace* in the main window.
- ### CLASS_NAME ###: Corresponds to *Class Name* in the main window.
- ### PUBLIC_PROPERTIES ###: This is where the generated properties are put.
- ### PUBLIC_METHODS ###: This is where the generated methods are put.
- ### ENUM_DECLARATIONS ###: This is where the enum declarations are put.

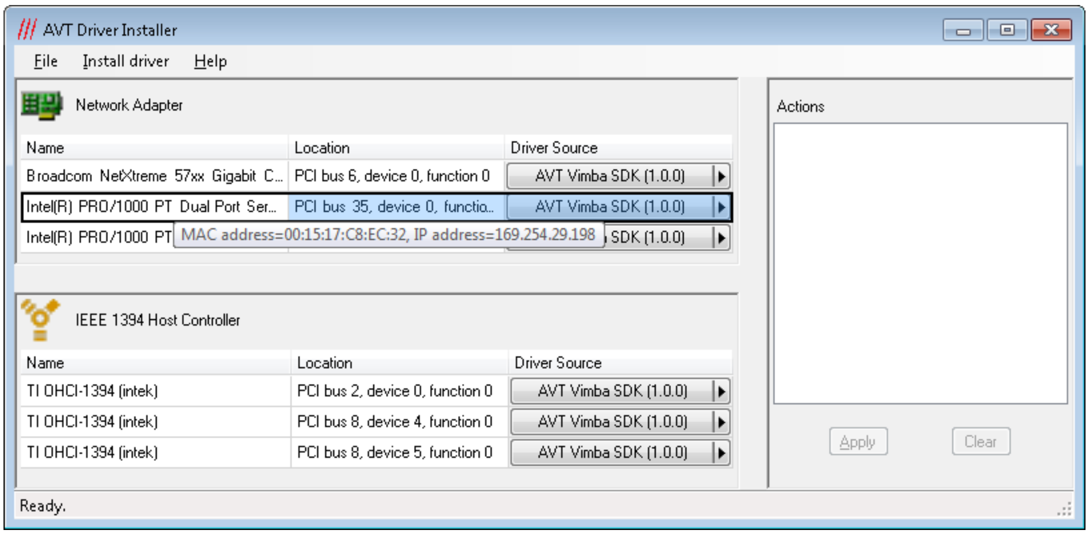
You may move around these variables in the template file to generate a file that better suits your requirements.

5 AVT Driver Installer

The AVT Driver Installer is a tool to easily install and configure hardware devices or filter drivers on the system. Although the same functionality can be achieved with the Windows device manager or through the network settings, the AVT Driver Installer provides a more convenient way to select the correct driver for the AVT software.

5.1 Main window

Upon startup, the Driver Installer examines the current hardware configuration, which can take a while. After the necessary information is collected, the main window is shown.



Name	Location	Driver Source
Broadcom NetXtreme 57xx Gigabit C...	PCI bus 6, device 0, function 0	AVT Vimba SDK (1.0.0)
Intel(R) PRO/1000 PT Dual Port Ser...	PCI bus 35, device 0, functio...	AVT Vimba SDK (1.0.0)
Intel(R) PRO/1000 PT		SDK (1.0.0)

MAC address=00:15:17:C8:EC:32, IP address=169.254.29.198

Name	Location	Driver Source
TI OHCI-1394 (intek)	PCI bus 2, device 0, function 0	AVT Vimba SDK (1.0.0)
TI OHCI-1394 (intek)	PCI bus 8, device 4, function 0	AVT Vimba SDK (1.0.0)
TI OHCI-1394 (intek)	PCI bus 8, device 5, function 0	AVT Vimba SDK (1.0.0)

Figure 4: Driver Installer - Main Window

The window shows two panels on the left side: the top one lists the network and the bottom one the 1394 adapters. If no network or 1394 adapters are installed, the corresponding panel is hidden. If no Vimba suitable adapters are present in the system, an error message is shown and the application is closed.

All adapters are listed with their name and their physical location in the PC. To easily identify a network adapter, its MAC address and IP addresses are shown in a tool tip. The Driver Source column displays which version of the currently installed driver is used.

The Actions panel shows the actions to be performed by the Driver Installer if the Apply button is clicked. It is initially empty.

5.2 Changing a driver

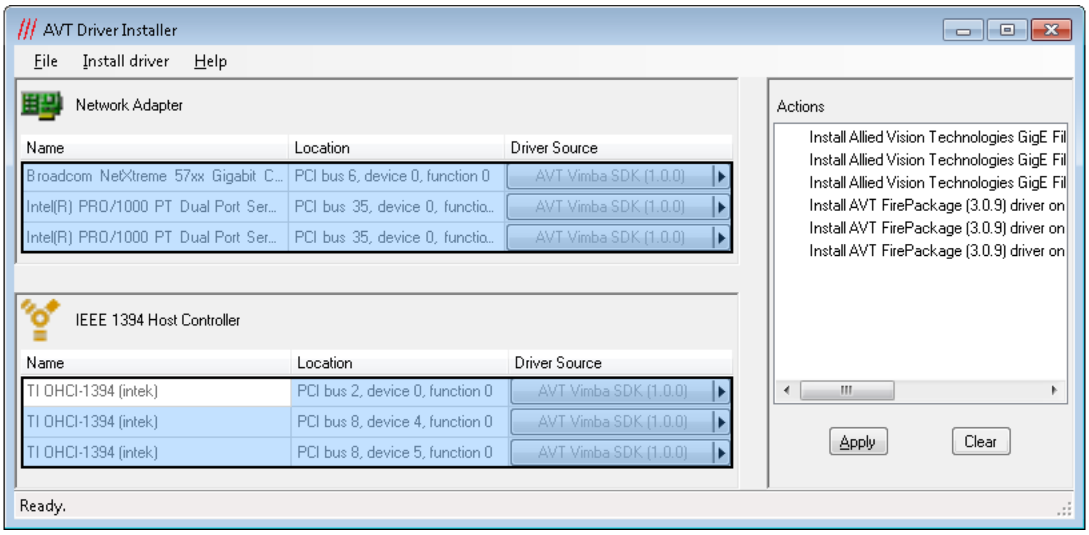
To change the driver for an adapter, select the product whose driver should be installed. This can be done by three different methods:

- Click the button in the column *Driver Source* of the corresponding adapter and select an entry from the popup menu,
- Right-click anywhere in the row of the corresponding adapter and select an entry from the popup menu, or
- select an adapter by clicking in the corresponding row and choose an entry from the *Install Driver* menu.

Note

The Driver Installer supports multi-selection of adapters: On your keyboard, press Ctrl + A or hold down the *Shift* or *Control* key while clicking.

The list of products to choose from is determined at startup by searching the PC for currently installed AVT products. Additionally, one item is added to each list that can be used to remove the AVT product from the adapter: <*disable AVT filter driver*> for network adapters and *Microsoft* for 1394 adapters. After selection of a new product, one or more actions that should be performed by the Driver Installer are added to the right panel. The corresponding adapters are disabled, so you cannot add a second action for the same adapter. To undo your selections, click the *Clear* button. This will empty the list of actions so that a new product can be selected.



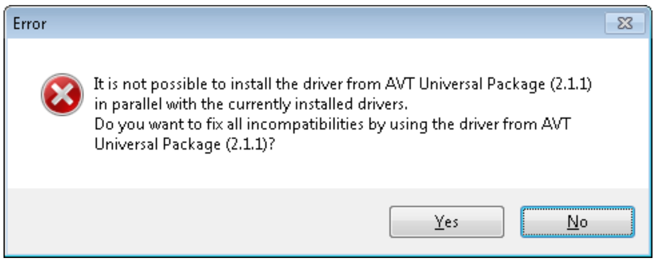
Name	Location	Driver Source
Broadcom NetXtreme 57xx Gigabit C...	PCI bus 6, device 0, function 0	AVT Vimba SDK (1.0.0)
Intel(R) PRO/1000 PT Dual Port Ser...	PCI bus 35, device 0, functio...	AVT Vimba SDK (1.0.0)
Intel(R) PRO/1000 PT Dual Port Ser...	PCI bus 35, device 0, functio...	AVT Vimba SDK (1.0.0)

Name	Location	Driver Source
TI OHCI-1394 (intek)	PCI bus 2, device 0, function 0	AVT Vimba SDK (1.0.0)
TI OHCI-1394 (intek)	PCI bus 8, device 4, function 0	AVT Vimba SDK (1.0.0)
TI OHCI-1394 (intek)	PCI bus 8, device 5, function 0	AVT Vimba SDK (1.0.0)

Install Allied Vision Technologies GigE Fil
Install Allied Vision Technologies GigE Fil
Install Allied Vision Technologies GigE Fil
Install AVT FirePackage (3.0.9) driver on
Install AVT FirePackage (3.0.9) driver on
Install AVT FirePackage (3.0.9) driver on

Figure 5: Driver Installer - Prepare actions

Some AVT products are incompatible with each other. When this is detected, the following error message is shown:

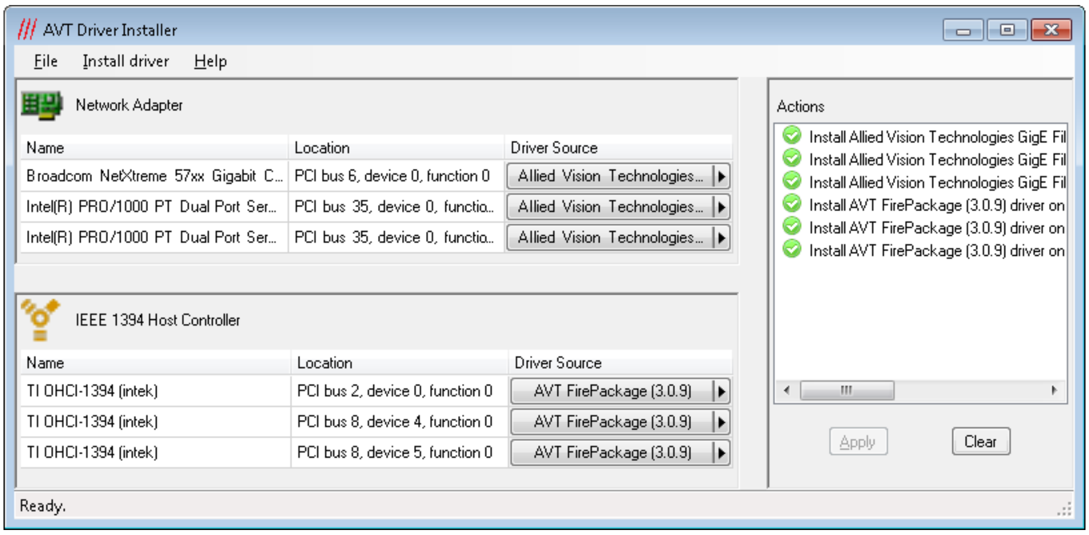

Error

It is not possible to install the driver from AVT Universal Package (2.1.1) in parallel with the currently installed drivers.
Do you want to fix all incompatibilities by using the driver from AVT Universal Package (2.1.1)?

Figure 6: Driver Installer - Incompatible products detected

If you click *Yes*, additional actions will be added to install the driver of new product on the other adapters too. In case of *No* the action will be removed.

To start the actual installation, click the *Apply* button. During the installation, which can take a while, the operating system might bring up other windows asking for permission to install the driver. The result of each action is displayed by a small icon in front of the action.



Network Adapter

Name	Location	Driver Source
Broadcom NetXtreme 57xx Gigabit C...	PCI bus 6, device 0, function 0	Allied Vision Technologies...
Intel(R) PRO/1000 PT Dual Port Ser...	PCI bus 35, device 0, functio...	Allied Vision Technologies...
Intel(R) PRO/1000 PT Dual Port Ser...	PCI bus 35, device 0, functio...	Allied Vision Technologies...

IEEE 1394 Host Controller

Name	Location	Driver Source
TI OHCI-1394 (intek)	PCI bus 2, device 0, function 0	AVT FirePackage (3.0.9)
TI OHCI-1394 (intek)	PCI bus 8, device 4, function 0	AVT FirePackage (3.0.9)
TI OHCI-1394 (intek)	PCI bus 8, device 5, function 0	AVT FirePackage (3.0.9)

Actions

✓ Install Allied Vision Technologies GigE Fil
✓ Install Allied Vision Technologies GigE Fil
✓ Install Allied Vision Technologies GigE Fil
✓ Install AVT FirePackage (3.0.9) driver on
✓ Install AVT FirePackage (3.0.9) driver on
✓ Install AVT FirePackage (3.0.9) driver on

Ready.

Figure 7: Driver Installer - Installation successful

5.3 Command line options

The Driver Installer can either be used in interactive mode or perform a single task without showing a graphical user interface (GUI). The syntax for starting the application is:

```
AVTDriverInstaller.exe [command] [options]
```

The following commands are available:

Command	Description	Mandatory options
`help`	Displays infos about the usage of the Driver Installer	*none*
`info`	Displays infos about installed AVT products and hardware.	*none*
`add`	Adds a driver of an AVT product to the driver store.	*/product*
`remove`	Removes a driver of an AVT product from the driver store.	*/product*
`exists`	Checks whether the given AVT product is installed.	*/product*
`install`	Installs a driver of an AVT product on the given adapters.	*/product* and */deviceClass*

The following options can be used:

Option	Description
`/l filename`	Log trace messages into the given file.
`/product upgradeCode`	Identifies a product by its upgrade code (example: 'd108d42f-c1d4-4d86-aa1f-e3ec4a137aaf').
`/product productNameAndVersion`	Identifies a product by its name and version (example: 'AVT Vimba (1.0.0)'.
`/deviceClass deviceClassName`	Identifies the hardware devices by their device class (either 'net' for network adapter or '1394' for 1394 adapter).
`/force`	Force removal of a product even if it is in use. A reboot might be necessary when this option is used. This option is valid for the "remove" command only.
`/hidden`	Suppress output to console window. Normally, when the Driver Installer is run without a GUI, it creates a console window to display its output. This can be disabled by using this option.
`/adjustMaxFilterDriverNums`	Windows operating systems limit the number of network filter drivers that can be installed on a system at the same time. Starting with Windows Vista, this number is configurable. Use this option if you want to adjust the number automatically to ensure that the AVT GigE Filter Driver can be installed. This option is valid only when the "install" command is used together with a network device.

The options */product* and */deviceClass* can be specified multiple times.

Note

Use the *info* command to find out the upgrade codes or exact names and versions for installed AVT products.

If started without parameters, the application is displayed in interactive mode with GUI.

The following table lists the return code when started in silent mode:

Return value	Description
1	Success, reboot required.
0	Success.
-1	Unknown error.
-2	Unknown command.
-10	Given device not found.
-11	At least one device must be specified for this command.
-12	Exactly one device must be specified for this command.
-13	Given device cannot be modified, because it is in use.
-20	Given product not found.
-21	At least one product must be specified for this command.
-22	Exactly one product must be specified for this command.
-23	Given product cannot be removed from the driver store, because it is in use.
-24	Given product is not found in the driver store.
-25	Given product is not installed on the given device.
-1001	The network configuration system is locked by another application. Terminate all other applications and run the Driver Installer again.
-1002	The maximum number of installed network filter drivers is reached. Use the "adjustMaxFilterDriverNums" option or remove a network filter driver by uninstalling the corresponding application.

Examples:

- `AVTDriverInstaller info`
  Displays infos about installed AVT products and hardware.
- `AVTDriverInstaller remove /product "AVT Vimba (1.0.0)"`
  Removes all drivers of Vimba 1.0 from the driver store.
- `AVTDriverInstaller install /product d108d42f-c1d4-4d86-aa1f-e3ec4a137aaf /deviceClass 1394 /deviceClass net`
  Install drivers of Vimba 1.0 on all 1394 and network devices. Here the product is given by its upgrade code.

6 Vimba Deployment

With Vimba it is easy to integrate the setup into your own projects. Of course you can use the Vimba setup executable to manually install the software on the target machine, but it is also possible to run the installer silently. At first you need the Microsoft Installer files (MSI). As they are contained in the setup executable, it is not necessary to download them separately from the AVT website. Just run

```
AVTVimba.exe /extract
```

to extract the MSIs from the setup executable. Please note that this command will neither install anything nor will it affect a current installation on the machine. After selecting the destination folder, the following MSI files are created:

Name	Description
`AVT1394TL_Win32.msi`	AVT 1394 Transport Layer for 32-bit Windows operating systems.
`AVT1394TL_Win64.msi`	AVT 1394 Transport Layer for 64-bit Windows operating systems.
`AVTGigETL_Win32.msi`	AVT GigE Vision Transport Layer for 32-bit Windows operating systems.
`AVTGigETL_Win64.msi`	AVT GigE Vision Transport Layer for 64-bit Windows operating systems.
`AVTVimba_Win32.msi`	AVT Vimba for 32-bit Windows operating systems.
`AVTVimba_Win64.msi`	AVT Vimba for 64-bit Windows operating systems.

If you want to install Vimba for an application using the GenICam Transport Layer interface, it is sufficient to install one or more AVT transport layer modules. If your application uses one of the Vimba APIs, it is necessary to additionally install Vimba.

Microsoft provides a tool named *msiexec* to install an MSI file. The following command line installs the core components of the AVT 1394 Transport Layer module to the default programs folder:

```
msiexec -i AVT1394TL_Win32.msi
ADDLOCAL="VCRedist100,FireWireTL_RegisterGenICamPathVariable,FireWireTL_Core"
```

And the same for the AVT GigE Vision Transport Layer module:

```
msiexec -i AVTGigETL_Win32.msi
ADDLOCAL="VCRedist100,GigETL_RegisterGenICamPathVariable,GigETL_Core"
```

The installation of Vimba depends on what kind of Vimba API should be provided. For the Vimba C API, the command line is:

```
msiexec -i AVTVimba_Win32.msi ADDLOCAL="VCRedist90,VCRedist100,
VCRedist100_MFC,Vimba_Core,VimbaC_Core,AvtImageTransform_Core,Vimba_DrvInst"
```

For the Vimba C++ API it is:

```
msiexec -i AVTVimba_Win32.msi ADDLOCAL="VCRedist90,VCRedist100,
VCRedist100_MFC,Vimba_Core,VimbaCPP_Core,AvtImageTransform_Core,Vimba_DrvInst"
```

And the Vimba .NET API is installed with:

```
msiexec -i AVTVimba_Win32.msi ADDLOCAL="VCRedist90,VCRedist100,
VCRedist100_MFC,Vimba_Core,VimbaNET_Core,AvtImageTransform_Core,Vimba_DrvInst"
```

Other useful parameters for *msiexec* are:
INSTALLDIR="*path to desired installation directory*"
ALLUSERS="1" Performs a machine-wide installation. If omitted, a per user installation is performed

Note

The installation of the AVT Transport Layer modules does not automatically install the corresponding hardware drivers. This has to be done afterwards by calling the *AVTDriverInstaller* (see section *Command line options* for further reference).

7 Compiling the Vimba C++ API

If the feature *C++ API development components* is installed, the Vimba C++ API source code along with Microsoft Visual Studio solution files for version 2005, 2008, and 2010 can be found in a sub-folder `VimbaCPP_Source` under the examples installation folder. The main reason is that you can inject your own shared pointer implementation into the Vimba C++ API. For more information on how to use shared pointers in the API, refer to the corresponding section in the Vimba C++ API Programmer's Reference Manual. A link to this document is given in section *References*.

Note

Please note that Allied Vision Technologies can give only limited support if an application uses a modified version of the Vimba C++ API.

8 Vimba Cognex Adapter

Since version 1.2, there is an adapter for Cognex software in the Vimba package. With this adapter, all cameras that are supported in Vimba are automatically supported in Cognex software.

For a detailed description, see the user manual in chapter References.

9 References

The following table lists some documents with more detailed information about the components of the Vimba SDK. Please note that the links are valid only if the corresponding component has been installed.

AVT 1394 Transport Layer

- 1394 Transport Layer Feature Description.

AVT GigE Vision Transport Layer

- GigE Vision Transport Layer Feature Description.

AVT Image Transform Library

- Programmer's Manual.

Vimba C API

- Programmer's Manual
- Function Reference
- Vimba SDK Features

Vimba C++ API

- Programmer's Manual
- Function Reference
- Vimba SDK Features

Vimba .NET API

- Programmer's Manual
- Vimba SDK Features

AVT Vimba Cognex Adapter

- Vimba Cognex Adapter User Guide.