Contents

1 About this manual  .......................................................... 2
  1.1 Composition of the manual .................................................. 2
  1.2 How to get started? ......................................................... 2
    1.2.1 Introduction ......................................................... 2
    1.2.2 Basics ............................................................... 3
    1.2.3 Image acquisition concept ......................................... 5
    1.2.4 Installation ......................................................... 5
    1.2.5 Programming ....................................................... 6

2 Imprint  ........................................................................... 7

3 Revisions ........................................................................ 8

4 Graphic Symbols ............................................................. 11
  4.1 Notes, Warnings, Attentions ............................................... 11
  4.2 Webcasts .................................................................... 11

5 Important information .......................................................... 12
  5.1 European Union Declaration of Conformity statement .............. 12

6 Introduction ................................................................. 13
  6.1 What’s inside and accessories ............................................. 13

7 Quickstart ................................................................. 14
  7.1 Hardware installation ...................................................... 14
  7.2 Windows .................................................................. 14
    7.2.1 System Requirements ................................................ 14
    7.2.2 Software installation .................................................. 15
  7.3 Linux ........................................................................ 20
    7.3.1 System Requirements ................................................ 20
    7.3.2 Software Installation .................................................. 21
    7.3.3 Writing your own applications ..................................... 27
    7.3.4 Special notes for Linux ............................................... 28
    7.3.5 Copyrights ............................................................. 28
  7.4 Settings behavior during startup ........................................ 29
14 DirectShow Interface

14.1 Supported Interfaces

14.1.1 IAMCameraControl

14.1.2 IAMDroppedFrames

14.1.3 IAMStreamConfig

14.1.4 IAMVideoProcAmp

14.1.5 IKsPropertySet

14.1.6 ISpecifyPropertyPages

14.2 Logging

14.3 Registering and renaming devices for DirectShow usage

14.3.1 Registering devices

14.3.2 Renaming devices

14.3.3 Make silent registration

15 Glossary
1 About this manual

1.1 Composition of the manual

The mvIMPACT Acquire manual for the MATRIX VISION frame grabbers is based on a modular concept. That means like in many object-oriented programming languages you have for each functionality your own "class". Instead of classes, you have books. For example, if you want to know how images are acquired with the frame grabbers, have a look in the respective programming language chapter.

Here is a short summary about all books of the frame grabber manual:

- The manual starts with technical data of the frame grabber as well as a quick start chapter.

Afterwards, you will find the different books:

- **Application Usage** (p. 42)
  - The frame grabbers can also be managed via user interface. The program is called wxPropView (p. 42).
- **DirectShow developers** (p. 90)
  - This is the documentation of the MATRIX VISION DirectShow_acquire interface.

**Note**

For C, C++, .NET developers, there are separate mvIMPACT Acquire manuals

- "mvIMPACT_Acquire_API_CPP_manual.chm",
- "mvIMPACT_Acquire_API_C_manual.chm", and
- "mvIMPACT_Acquire_API_NET_manual.chm"

available as downloads from our website [http://www.matrix-vision.com](http://www.matrix-vision.com). The manuals contain chapter about

- how to link and build applications using mvIMPACT Acquire,
- how the log output for "mvIMPACT Acquire" devices is configured and how it works in general,
- how to create your own installer packages for Windows and Linux, and
- the general mvIMPACT Acquire API documentation.

1.2 How to get started?

1.2.1 Introduction

This chapter gives you a short overview, how to get started with a MATRIX VISION frame grabber and where to find the necessary information in the manual. It will also explain or link to the concepts behind the driver and the image acquisition. Furthermore it shows you how to get start programming own applications.
1.2 How to get started?

1.2.2 Basics

1.2.2.1 Driver concept

The driver supplied with the MATRIX VISION product represents the port between the programmer and the hardware. The driver concept of MATRIX VISION provides a standardized programming interface to all image processing products made by MATRIX VISION GmbH. The advantage of this concept for the programmer is that a developed application runs without the need for any major modifications to the various image processing products made by MATRIX VISION GmbH. You can also incorporate new driver versions, which are available for download free of charge on our website.

The following diagram shows a schematic structure of the driver concept:

![Diagram of Driver Concept]

- 1 Part of any mvIMPACT Acquire driver installation package (Windows).
- 2 Separately available for 32 bit and 64 bit. Requires at least one installed driver package.
- 3 See 2, but requires an installed version of the mvBlueFOX driver.
- 4 Part of the NeuroCheck installer but requires at least one installed frame grabber driver.
- 5 Part of the mvIMPACT SDK installation. However, new designs should use the .NET libs that are now part of mvIMPACT Acquire ("mv.impact.acquire.dll"). The namespace "mv.impact.acquire" of "mv.impact.acquire.dll" provides a more natural and more efficient access to the same features as contained in the namespace "mvIMPACT.NET.acquire" of "mvIMPACT.NET.dll", which is why the latter one should only be used for backward compatibility but NOT when developing a new application.
- 6 Part of Micro-Manager.
1.2.2.2 NeuroCheck support

A couple of devices are supported by NeuroCheck. However between NeuroCheck 5.x and NeuroCheck 6.x there has been a breaking change in the internal interfaces. Therefore also the list of supported devices differs from one version to another and some additional libraries might be required.

For NeuroCheck 5.x the following devices are supported:

<table>
<thead>
<tr>
<th>Device</th>
<th>Additional software needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>mvTITAN-G1</td>
<td>mvSDK driver for mvTITAN/mvGAMMA devices</td>
</tr>
<tr>
<td>mvTITAN-CL</td>
<td>mvSDK driver for mvTITAN/mvGAMMA devices</td>
</tr>
<tr>
<td>mvGAMMA-CL</td>
<td>mvSDK driver for mvTITAN/mvGAMMA devices</td>
</tr>
<tr>
<td>mvBlueFOX</td>
<td>mvIMPACT Acquire driver for mvBlueFOX devices, &quot;NCUSBmvBF.dll&quot;</td>
</tr>
</tbody>
</table>

For NeuroCheck 6.0 the following devices are supported:

<table>
<thead>
<tr>
<th>Device</th>
<th>Additional software needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>mvTITAN-G1</td>
<td>mvIMPACT Acquire driver for mvTITAN/mvGAMMA devices</td>
</tr>
<tr>
<td>mvTITAN-CL</td>
<td>mvIMPACT Acquire driver for mvTITAN/mvGAMMA devices</td>
</tr>
<tr>
<td>mvGAMMA-CL</td>
<td>mvIMPACT Acquire driver for mvTITAN/mvGAMMA devices</td>
</tr>
<tr>
<td>mvHYPERION-CLb</td>
<td>mvIMPACT Acquire driver for mvHYPERION devices</td>
</tr>
<tr>
<td>Every other mvIMPACT Acquire compliant device</td>
<td>mvIMPACT Acquire driver for the corresponding device family, &quot;mv.impact.acquire.NeuroCheck6_.dll&quot; (comes with the driver package, but the driver package must be installed AFTER installing NeuroCheck 6)</td>
</tr>
</tbody>
</table>

For NeuroCheck 6.1 the following devices are supported:

<table>
<thead>
<tr>
<th>Device</th>
<th>Additional software needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>mvTITAN-G1</td>
<td>mvIMPACT Acquire driver for mvTITAN/mvGAMMA devices</td>
</tr>
<tr>
<td>mvTITAN-CL</td>
<td>mvIMPACT Acquire driver for mvTITAN/mvGAMMA devices</td>
</tr>
<tr>
<td>mvGAMMA-CL</td>
<td>mvIMPACT Acquire driver for mvTITAN/mvGAMMA devices</td>
</tr>
<tr>
<td>mvHYPERION-CLb</td>
<td>mvIMPACT Acquire driver for mvHYPERION devices</td>
</tr>
<tr>
<td>Every other mvIMPACT Acquire compliant device</td>
<td>mvIMPACT Acquire driver for the corresponding device family, &quot;mv.impact.acquire.NeuroCheck6_1.dll&quot; (comes with the driver package, but the driver package must be installed AFTER installing NeuroCheck 6.1)</td>
</tr>
</tbody>
</table>

1.2.2.3 VisionPro support

Every *mvIMPACT Acquire* driver package under Windows comes with an adapter to VisionPro from Cognex. The installation order does not matter. After the driver package and VisionPro has been installed, the next time VisionPro is started it will allow selecting the *mvIMPACT Acquire* device. No additional steps are needed.

MATRIX VISION devices that also comply with the GigE Vision or USB3 Vision standard don't need any software at all, but can also use VisionPro's built-in GigE Vision or USB3 Vision support.
1.2 How to get started?

1.2.2.4 HALCON support

HALCON comes with built-in support for `mvIMPACT Acquire` compliant devices, so once a device driver has been installed for the `mvIMPACT Acquire` device, it can also be operated from a HALCON environment using the corresponding acquisition interface. No additional steps are needed.

MATRIX VISION devices that also comply with the GigE Vision or USB3 Vision standard don't need any software at all, but can also use HALCON's built-in GigE Vision or USB3 Vision support.

As some `mvIMPACT Acquire` device driver packages also come with a GenTL compliant interface, these can also be operated through HALCON's built-in GenTL acquisition interface.

1.2.2.5 LabVIEW support

Every `mvIMPACT Acquire` compliant device can be operated under LabVIEW through an additional set of VIs which is shipped by MATRIX VISION as a separate installation ("mvLabVIEW Acquire").

MATRIX VISION devices that also comply with the GigE Vision or USB3 Vision standard don't need any additional software at all, but can also be operated through LabVIEW's GigE Vision or USB3 Vision driver packages.

1.2.2.6 DirectShow support

Every `mvIMPACT Acquire` compliant device driver package comes with an interface to DirectShow. In order to be usable from a DirectShow compliant application, devices must first be registered for DirectShow support. How to do this is explained here (p.91).

1.2.2.7 Micro-Manager support

Every `mvIMPACT Acquire` compliant device can be operated under [https://micro-manager.org](https://micro-manager.org) when using `mvIMPACT Acquire` 2.18.0 or later and at least `Micro-Manager` 1.4.23 build AFTER 15.12.2016. The adapter needed is part of the `Micro-Manager` release. Additional information can be found here: [https://micro-manager.org/wiki/MatrixVision](https://micro-manager.org/wiki/MatrixVision).

1.2.3 Image acquisition concept

The image acquisition is based on queues to avoid the loss of single images. With this concept you can acquire images via single acquisition or triggered acquisition. For detailed description of the acquisition concept, please have a look at "How the capture process works" in the `mvIMPACT_Acquire_API` manual matching the programming language you are working with.

1.2.4 Installation

To install the frame grabber properly you have to follow these steps:

(Please follow the links for detailed descriptions.)

- **Windows:**
  - Please check the system requirements (p.14).
  - Please install the software and driver (p.15).
  - Please install the hardware (p.14).

- **Linux:**
  - Please check the system requirements (p.20).
  - Please install the software and driver (p.21).
  - Please install the hardware (p.14).
1.2.5 Programming

To control the camera and handle the images, you will have a good introduction by reading the main pages of the "mvIMPACT Acquire" interface references. Additionally, please have a look at the example programs. Several basic examples are available. The separate mvIMPACT Acquire manuals

- "mvIMPACT_Acquire_API_CPP_manual.chm",
- "mvIMPACT_Acquire_API_C_manual.chm", and
- "mvIMPACT_Acquire_API_NET_manual.chm"

are available as downloads from our website [http://www.matrix-vision.com](http://www.matrix-vision.com).

Only mvTITAN/mvGAMMA and mvSIGMA/mvDELTA series using Linux:

Please have a look at the chapter Writing your own applications (p.27) for details.
This document assumes a general knowledge of PCs and programming.

Since the documentation is published electronically, an updated version may be available online. For this reason we recommend checking for updates on the MATRIX VISION website.

MATRIX VISION cannot guarantee that the data is free of errors or is accurate and complete and, therefore, assumes no liability for loss or damage of any kind incurred directly or indirectly through the use of the information of this document.

MATRIX VISION reserves the right to change technical data and design and specifications of the described products at any time without notice.
3 Revisions

• 09. November 2018
  – Added “Hard Disk Recording” in wxPropView (p. 42).

• 21. December 2016
  – Added Setting up multiple display support and/or work with several capture settings in parallel (p. 55).

• 15. December 2016
  – Added Micro-Manger in Driver concept (p. 3).

• 11. March 2015
  – Added chapter Accessing log files (p. 54).

• 21. October 2014
  – Added description about the record mode in How to see the first image (p. 44).

• 06. December 2013
  – Added information about Changing the view of the property grid to assist writing code that shall locate driver features (p. 53).

• 15. October 2013
  – Added Webcasts (p. 11) links.
  – Added chapter Bit-shifting an image (p. 62).

• 24. January 2013
  – Added information about image error counts and disabling CPU sleep states: How to disable CPU sleep states a.k.a. C states (< Windows 8) (p. 79).

• 14. December 2012
  – New version of technical documentation.

• 20. September 2012
  – Added chapter "Porting existing code written with versions earlier than 3.0.0"

• 06. July 2012
  – Changed manual structure.

• 17. Feb 2012
  – Renewed chapter wxPropView (p. 42).

• 09. November 2011
  – Added Settings behavior during startup (p. 29) in chapter Quickstart (p. 14).

• 26. July 2011
  – Removed chapter EventHandling. See "Porting existing code written with versions earlier than 2.0.0".

• 11. July 2011:
  – Added chapter "Callback demo".

• 06. June 2011:
• Added chapter "Porting existing code written with versions earlier than 2.0.0".

• 18. January 2011:
  – Added description Setting up multiple display support and/or work with several capture settings in parallel (p. 55).

• 19. October 2010:
  – Added chapter "Chunk data format".

• 02. Aug. 2010:
  – Added chapter Import and Export images (p. 54).

• 19. April 2010:
  – Added example ContinuousCaptureDirectX.

• 28. January 2010:
  – Added chapter Copy grid data to the clipboard (p. 53).

• 13. January 2010:
  – Added chapter "Porting existing code written with versions earlier than 1.12.0".

• 10. November 2009:
  – Added Windows 7 as supported operating system.

• 05. November 2009:
  – Added example CaptureToUserMemory_C.

• 19. October 2009:
  – Updated wxPropView (p. 42) description about handling settings.

• 09. April 2009:
  – Added chapter "Porting existing code written with versions earlier than 1.11.0".

• 30. March 2009:
  – Added information for - DirectShow developers (p. 90).
  – Added new examples
    - DigitalIOs,
    - ContinuousCaptureMultipleInputs and
    - ContinuousCaptureMultipleVideoSignals.

• 14. November 2008:
  – Added wxPropView example wxPropView.

• 18. August 2008:
  – Added new example CaptureToUserMemory.

• 19. February 2008:
  – Added DIG-IN image in mvSIGMA-SQ/-SQe (p. 36).

• 07. November 2007:
– Added chapter "Porting existing code written with versions earlier than 1.10.0".

• 25. September 2007:
  – Added chapter mvDeviceConfigure (p.77).

• 1. August 2007:
  – Rewritten "How to use this manual". This book now includes a getting started chapter (see: Composi-
tion of the manual (p.2)).

• 29. May 2007:
  – Updated information about digital I/O Technical specifications (p.32).

• 14. February 2007:
  – Updated the description of the installable features (see: Software installation (p.15)).

• 12. February 2007:
  – Changed the screenshots of the driver installation in Software installation (p.15).

• 29. January 2007:
  – Update current consumption of PCI Express frame grabbers in Technical data (p.32).

• 25. January 2007:
  – Added mvDELT Ae description in Technical specifications (p.32).

• 24. January 2007:
  – Added note about the advanced user right "Increase scheduling priority" ("Anheben
der Zeitplanungsrioritaet") in System Requirements (p.14).

• 26. October 2006:
  – Added new examples EventHandling and CameraDescriptions.

• 23. October 2006:
  – Added new example ContinuousAllFormats and SequenceCapture.

• 2. October 2006:
  – Added Linux® installation description (see: Linux (p.20)).

• 28. July 2006:
  – Removed some linking errors.

• 19. July 2006:
  – Added WEEE-Reg.-No. (see: European Union Declaration of Conformity statement (p.12))

• 18. July 2006:
  – Added new sample ContinuousCaptureToAVIFile.

• 17. July 2006:
  – Added switch S1 and S2 description (see: Connectors (p.36)).

• 1. June 2006:
  – Updated the chm index.

• May 2006: Initial version
4 Graphic Symbols

4.1 Notes, Warnings, Attentions

**Note**

A *note* indicates important information that helps you optimize usage of the products.

**Warning**

A *warning* indicates how to avoid either potential damage to hardware or loss of data.

**Attention**

An *attention* indicates a potential for property damage, personal injury, or death.

All due care and attention has been taken in preparing this manual. In view of our policy of continuous product improvement, however, we can accept no liability for completeness and correctness of the information contained in this manual. We make every effort to provide you with a flawless product.

In the context of the applicable statutory regulations, we shall accept no liability for direct damage, indirect damage or third-party damage resulting from the acquisition or operation of a MATRIX VISION product. Our liability for intent and gross negligence is unaffected. In any case, the extend of our liability shall be limited to the purchase price.

4.2 Webcasts

This icon indicates a webcast about an issue which is available on our website.
5 Important information

We cannot and do not take any responsibility for the damage caused to you or to any other equipment connected to the mvDELTA / mvSIGMA frame grabber. Similarly, warranty will be void, if a damage is caused by not following the manual.

Handle the mvDELTA / mvSIGMA frame grabber with care. Do not misuse the mvDELTA / mvSIGMA frame grabber. Avoid shaking, striking, etc. The mvDELTA / mvSIGMA frame grabber could be damaged by faulty handling or shortage.

- Handle with care and avoid damage of electrical components by electrostatic discharge (ESD):
  - Discharge body static (contact a grounded surface and maintain contact).
  - Avoid all plastic, vinyl, and styrofoam (except antistatic versions) around printed circuit boards.
  - Do not touch components on the printed circuit board with your hands or with conductive devices.

5.1 European Union Declaration of Conformity statement

MATRIX VISION corresponds to the EU guideline WEEE 2002/96/EG on waste electrical and electronic equipment and is registered under WEEE-Reg.-No. DE 25244305.
6 Introduction

The frame grabbers of the mvDELTA and mvSIGMA series are powerful image capture boards suitable for different capturing and digitizing signals. For more information about the single frame grabbers and the supported signals, please have a look at the Technical data (p. 32).

6.1 What's inside and accessories

Due to the varying fields of application the frame grabber is shipped without accessories. The package contents:

• Frame grabber

For the first use of the frame grabber a cable for the camera connection is needed.
7 Quickstart

7.1 Hardware installation

**Warning**

Please take all proper Electro Static Discharge (ESD) precautions during the installation of your new hardware!

Before starting the installation, turn off your computer and all peripheral devices. Disconnect the computer from the power supply and all necessary components.

**Note**

To avoid doing damage to the hardware, discharge yourself of static charge by touching e.g. the casing. Beware of touching contacts of the frame grabber or of the computer.

- Select a free busmaster slot (PCI). Remove the slot's cover at the back of the computer and keep the screw.
- Carefully insert the board into the slot by holding the board at the top and gently pushing both ends into the slot at the same time. Press onto the upper edge of the board to make sure it is seated in the slot firmly.
- Do not force the board into the slot! You run the risk of bending the contacts. If the board does not fit easily, pull it back out, and try again.
- Fasten the board's bracket at the back of the computer using the screws you saved from the shield.
- Put the cover back on the computer and reconnect the peripheral devices.
- Start the computer.

7.2 Windows

7.2.1 System Requirements

Currently supported Windows versions are:

- Microsoft Windows 7 (32-bit, 64-bit) (requires min. 2 GB main memory)
- Microsoft Windows 8.1 (32-bit, 64-bit) (requires min. 2 GB main memory)

**Note**

Working under Windows XP you must enable at least the following advanced user right when not working with an Administrator account: "Increase scheduling priority" (for the German version of Windows: "Anheben der Zeitplanungsrioritaet"). These settings can be made under Windows XP via 'Settings -> Control Panel -> Administrative Tools -> Local Security Policy -> Local Policies -> User Rights Assignment').

Consecutively the installation for Windows will be described. The description for the Linux installation can be found here: Linux (p. [20](#)).
7.2 Windows

7.2.2 Software installation

All necessary drivers for Windows and Linux are contained in the mvIMPACT CD-ROM or DVD-ROM. For newer driver versions we recommend to visit the MATRIX VISION website at www.matrix-vision.de, section Support/Download/Hardware.

After the **Hardware installation** (p. 14) the boot sequence shows "**Found New Hardware**" and starts the Windows Hardware Wizard. Closed this windows and insert the mvIMPACT CD-ROM or DVD-ROM into your drive and select "**Driver installation ...**" and the needed mvIMPACT Acquire driver (e.g. "mvTITAN / mvGAMMA").

![Start window](image)

Figure 1: Start window

After the click on the needed driver the installation process starts.
Select the folder, where you want to install the software.

Select the features, which you want to install. Following features exist:
7.2 Windows

- "Base Libraries"
  This feature contains all necessary files for property handling and display. Therefore, it is not selectable.

- "mvDELTA/mvSIGMA driver"
  This is also not selectable.

- "Tools"
  This feature contains tools for the frame grabber (e.g. to acquire images (wxPropView (p. 42)).

- "mvIMPACT acquire API"
  The "mvIMPACT acquire API" contains the header for own programming. Additionally you can choose the examples, which installs the sources of wxPropView, mvIPConfigure and various small examples. The project files shipped with the examples have been generated with Visual Studio 2013. However projects and make-files for other compilers can be generated using CMake fairly easy. See CMake section in the C++ manual for additional details.

- "Documentation"
  This will install this manual as single HTML help file (.chm).

![Figure 4: mvDELTA/mvSIGMA installer - Select features](image)

Confirm the installation by clicking "Next".
The installation process copies the files to Windows. Then Windows shows two messages to signal that the drivers are not checked through Microsoft. This is only an attempt to make insecure and it is recommended to ignore it.

Press "Continue Anyway"...
... and again and finish the driver installation.

Figure 7: mvDELTA/mvSIGMA installer - Setup of the DMA buffer

Now, the installation process is finished.

Figure 8: mvDELTA/mvSIGMA installer - Installation complete
After this, you have to restart the system.

Figure 9: mvDELTA/mvSIGMA installer - Restart the system

After the restart, you can acquire images with the frame grabber. Simply start the application **wxPropView** (wxPropView.exe).

See also

wxPropView (p. 42)

### 7.3 Linux

The standard MATRIX VISION frame grabber drivers have been ported to Linux. A common code base ensures that the names and parameters of all the functions are identical to the Windows / DOS versions. This makes porting your existing Windows applications to Linux especially easy. We are constantly improving and updating so you should always take a look at our website for newer versions (http://www.matrix-vision.de). The driver consists of two parts:

- A kernel module called matrixfg.o, sigma.o, xsigma.o or titan.o. (From Linux kernel 2.5 onwards these will be called matrixfg.ko, sigma.ko, xsigma.ko and titan.ko). These modules handle kernel mode memory and I/O access and/or interrupts.
- A library which provides user programs with access to all the MATRIX VISION standard frame grabber functions as defined in the header file mv.h

#### 7.3.1 System Requirements

**Software requirements**

- Linux kernel 2.4.x or 2.6.x running on i386 CPU.
Note

It is possible that older kernels will also work.

You will need to have support for kernel modules turned on in your kernel configuration. (This is the default for all major Linux distributions). For the shared versions of the library, the GNU C runtime library libc6 is required. All recent Linux distributions should have this (e.g. SuSE Linux 9.0, 9.1, 9.2). The libraries have been compiled using glibc 2.3.3 on a SuSE 9.0 installation. If you use an older version of glibc you may experience problems. The libraries have been compiled using gcc version 3.3.1 (SuSE Linux). Using a system based on gcc 2.0 is not recommended and may not work correctly. For display functions: a framebuffer, the svga library or for X applications, a correctly installed X system. It is also possible to use a framebuffer device. Check your distribution for a package called svga (or similar) or download and compile the sources for this library if you are not using X and you wish to display images or use a framebuffer (e.g. vesafb, rivafb etc.). To compile the kernel modules you will need a correctly configured GNU compiler and at least the kernel header files and Makefiles installed. It is usually advisable to install all of the kernel sources. The kernel configuration should exactly match the kernel you are using. During installation the kernel modules will be compiled for your system using exactly the same configuration as for the kernel. If you have not compiled your own kernel you will probably have to install the source package for your kernel and retrieve the correct kernel configuration for the running kernel too. This is often stored as file in the /boot directory (Red Hat, Debian, Mandrake) or can be generated from /proc/config.gz in the case of SuSE or for 2.6.x kernels which have the appropriate kernel option turned on.

Hardware requirements

- i386 compatible PC. (PowerPC versions of the software are currently being prepared.)

One of the following types of MATRIX VISION frame grabber:

- mvDELTA
- mvSIGMA
- mvGAMMA
- mvTITAN

7.3.2 Software Installation

1. Copy the appropriate driver tarball from the CD or web to your computer (e.g. CD-ROM is mounted on /cdrom and directory /mypath exists):

   >cp -v /cdrom/linux/dist_titan-030801.tgz /mypath

2. Unpack the tgz-file using:

   >tar -xvzf dist_titan-030801.tgz

3. Make the test programs and kernel modules by changing to the top-level directory:

   >cd /mypath/dist_titan-030801

4. and then...

   > ./configure
   > make

   It is also possible (and sometimes necessary) to start the configure program with options. See the INSTALL file for full details. Two common options are:
> ./configure --with-kernel=<kernel source directory>

(to enter an alternative location for the kernel sources. The default location is /usr/src/linux).

5. Ensure you are logged on as "root" and then install the kernel modules on your computer with:

> make install

Allocator kernel module

Some MATRIX VISION frame grabbers support a linear DMA buffer in Hard Live mode. In order to ensure that we have a linear memory area, we use a modified version of the allocator kernel module from Alessandro Rubini's book Linux Device Drivers (O'Reilly 1998 ISBN: 1-56592-292-1). This originally handled only linear memory blocks above the top of the memory range usually used by the kernel (highmemory). The allocator module from Rubini will not work correctly on systems with more than 1 GB of memory. In fact, on 2.4.x kernels the upper limit is mem=896M. In order to address this problem we have significantly rewritten parts of the allocator module to use a different method and this is now used by default. The two methods are described below:

1. Method 1 "MATRIX VISION kmalloc method":

   This method makes use of the kernel function kmalloc to reserve blocks of memory. Since this function is only capable of reserving small blocks of memory (normally 128 kB) it must be called several times to reserve a block of, say 2 MB. If kmalloc is called early in the boot process before too many applications have used up memory (and possibly returned it to the pool, thereby creating "holes" in the memory map) then it is very likely that successive calls to kmalloc will return successive linear blocks of memory. On a machine with, say, 1GB of RAM the chances are very high that you will be able to get 2MB of continuous memory. The longer the computer runs the less likely is this method to work. Therefore it is essential that the allocator module is loaded as early as possible in the boot process and not unloaded again, as long as it is needed. The way to do this depends on your Linux distribution. Some, like Debian, allow you to enter the name of kernel modules that should be loaded during boot in a file (/etc/modules). Others, like SuSE or United Linux, allow boot scripts in /etc/init.d/boot.d which are capable of loading required modules. If you can find a way to load the allocator module during booting you may use this method and you will not need to pass a mem= parameter to your kernel in the boot loader.

2. Method 2 "Rubini's Method":

   In order to reserve space for allocator you need to start the kernel with an extra parameter which instructs it to use slightly less than all the memory. e.g If your computer contains 256 MB of RAM start the kernel with the parameter:

   mem=252M

   The kernel will be able to use 252MB of memory and 4 MB may be used by allocator. A computer with 512 MB of RAM could be started with:

   mem=508M

   Note

   Method 1 is only available if you are using a kernel with a version number >= 2.4.20.

   If you do not start your kernel using this parameter the allocator module will still load without an error but linear DMA access will not work correctly. The amount of high memory you should reserve depends on the resolution and colour depth you are going to use and on your application.

   Method 2 will not work correctly on systems with more than 1 GB of memory. In fact, on 2.4.x kernels the upper limit is mem=896M

3. Parameters:

   Some kernel module parameters are available to affect the way allocator works:

   Examples:

   If you need to reserve more RAM (e.g. 8MB) then use linsizemb=8.
   If you want a more conservative search (100 MB) then use maxresmb=100
   If you want to force Method 2 use use_kmalloc=0 (only on 2.4.x kernels).
   If you know that you have reserved exactly 8MB using the kernel mem=parameter (Method 2) but you do not want the allocator module to search for this free memory then try use_kmalloc=0 himem=8
4. **Summary:**

Assuming you have an up-to-date kernel method 1 will be used by default to reserve 4 MB of RAM and you will normally not need to provide any parameters. Up to half of your available RAM will be searched for this 4 MB. You should load allocator during the boot process. If you have a kernel older than 2.4.20 then method 2 will be used and you must provide a mem= parameter to your kernel boot loader and cannot use more than 896 MB of RAM (even if more is installed in your computer). You can load or unload allocator anytime you want.

**Frame grabber kernel module**

You can create the kernel modules individually, if you want:

1. Login as root.
2. Change directory to the
   
   `driver/os/linux/allocator`

   and

   `driver/os/linux/matrixfg` (or `driver/os/linux/titan` for e.g. mvTITAN)

   subdirectories in succession and
3. compile the kernel modules using the following command in each directory:

   `>./configure`

   `> make`

4. You can install the kernel modules (`allocator.o` and `matrixfg.o` or `titan.o`) by copying them to the appropriate directory on your computer (e.g. `"/lib/modules/2.2.16/misc"`) and then typing this command in the appropriate directory:

   `>make install`

You can now load the kernel modules by calling the script

`mvload matrixfg` (or `mvload titan`)

`mvunload matrixfg`

unloads both modules and deletes the device nodes.

---

**Note**

Device numbers for MATRIX VISION frame grabbers have not yet been officially assigned so we use a dynamic major device number technique which will usually assign a number of 254. See the script `mvmodprobe` or Rubini's book for more details. If you want to assign a fixed number (e.g. one of the numbers used for experimental drivers, for example 60) you can load matrixfg with a parameter (see below). Fixed major numbers also allow automatic loading of the kernel modules if you enter the appropriate alias lines in `/etc/modules.conf`. If everything has worked correctly you should see that both allocator and matrixfg(or titan) have been loaded by using the command:

`>lsmod`

This will list all the loaded kernel modules like this....

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Size</th>
<th>Used by</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrixfg</td>
<td>17212</td>
<td>(unused)</td>
</tr>
<tr>
<td>allocator</td>
<td>2104</td>
<td>[matrixfg]</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>etc.</td>
</tr>
</tbody>
</table>

>`
A very small part of the driver is supplied in binary form only. If you do not wish to "taint" your kernel then please use the GPL version of the driver. The GPL version contains most of the features of the non-GPL version with the exception of e.g. fast channel switching and triggered capture of images.

5. **Parameters:**

The kernel modules matrixfg or titan may be loaded using one or more of the following parameters:

**VGA card**

In order to display live images on your screen the driver attempts to determine the starting address of the linear memory used by the installed VGA card. It is possible to do this if the X-Windows system is running using function calls to an extended library (DGA). However, this library is not installed on all X-Windows systems, so the configure script will attempt to find out if DGA is being used on your system. Sometimes it may be helpful to turn on DGA by altering the configuration file for X-Windows (usually /etc/X11/XF86Config). To do this you need to have "root" access and you need to add a line to the section called "Module" like this:

```
Section "Module"
Load "extmod"
```

Using PCI mapping information the driver can determine which of your cards is the VGA card and which memory areas are mapped. However, it is not always possible to tell which of the mapped areas is the linear image memory. If the VGA card is a common one it should be known by the driver and the VGA detection will work correctly. New cards are constantly appearing on the market and it is not always possible to keep the driver up-to-date. In order to ensure that your VGA card is correctly detected please follow the procedure described below:

You may check whether the VGA card has been correctly detected by looking at your kernel message file (using dmesg or look at the file /var/log/messages). If you find (after some searching) a line like this

```
Sep 10 11:00:45 .... kernel: mv_vga Unknown video memory base address.
```

it means that the VGA card is unknown. If we don't know the VGA card then we can't be sure of the linear display memory address which we need as the destination for our DMA transfers. In this case we need:

1. to find out the address and start the kernel module by hand with an extra parameter and/or
2. add the information to a table in the kernel module sources and recompile.

In fact, the best way is to try it out using (1) and then do (2).

Here's how to do it: Open the file `driver/os/linux/kmod/vgatable.h` and read the comments in the header area. It tells you how to add the vendor ID and device ID for your graphic card to the driver, Alternatively you can load the matrixfg kernel module using an extra parameter like this:

`vidmem=0xd8000000` (Example for a NVidia card)

Just replace 0xd8000000 with the correct PCI memory address for your card. You can find this out by using "lspci -v" or "cat /proc/pci". If your card has more than one memory area you might have to try them all out until, when loading the kernel module, the error message (Unknown video memory base address) disappears and the test program works.
Note

The table in `driver/os/linux/kmod/vgatable.h` shows you which of the base addresses (PCI_BASE_ADDRESS_0 or PCI_BASE_ADDRESS_1) most of the usual cards use. PCI_BASE_ADDRESS_0 is the first address that "lspci -v" or "cat /proc/pci" shows and PCI_BASE_ADDRESS_1 is the second address. Some dual-head cards have 3 addresses. Note that the command lspci is actually deprecated and it may be necessary to examine the PCI cards using a different command or the command "cat /proc/pci".

Example Here is part of the output for Linux computer at MATRIX VISION ("lspci -v"):

```
01:00.0 VGA compatible controller: nVidia Corporation GeForce 256 (rev 10)
        (prog-if 00 [VGA])
        Subsystem: Guillemot Corporation: Unknown device 5022
        Flags: bus master, 66Mhz, medium devsel, latency 32, IRQ 11
        Memory at e0000000 (32-bit, non-prefetchable) [size=16M]
        Memory at d8000000 (32-bit, prefetchable) [size=128M]
        Expansion ROM at <unassigned> [disabled] [size=64K]
        Capabilities: [60] Power Management version 1
        Capabilities: [44] AGP version 2.0

And here's the same thing using "lspci -nv" (showing numeric IDs). Class 0300 is a VGA card:

```
01:00.0 Class 0300: 10de:0100 (rev 10)
        Subsystem: 14af:5022
        Flags: bus master, 66Mhz, medium devsel, latency 32, IRQ 11
        Memory at e0000000 (32-bit, non-prefetchable) [size=16M]
        Memory at d8000000 (32-bit, prefetchable) [size=128M]
        Expansion ROM at <unassigned> [disabled] [size=64K]
        Capabilities: [60] Power Management version 1
        Capabilities: [44] AGP version 2.0
```

Trying out the address 0xe0000000 does not work here. But the second address (0xd8000000) does work. So the following line needs to be added to the table in `vgatable.h` and `matrixfg.o` needs to be recompiled, installed and loaded. The vendor ID is 0x10de and the device ID is 0x0100 (from the output above):

```
{0x10de, 0x0100, "Geforce 256", PCI_BASE_ADDRESS_1},
```

Sometimes the IDs have already been defined in the kernel headers (`/usr/source/linux/include/linux/pci.h` or `pci.ids.h`) and you can use sensible names instead of numbers like this (an invented example, doesn't really exist):

```
{PCI_VENDOR_ID_NVIDIA, PCI_DEVICE_ID_NVIDIA_GEFORCE2_256, "Geforce 256",
PCI_BASE_ADDRESS_1}
```

If you are successful you might like to inform MATRIX VISION what you have added to the table so that it can be included in future releases.
Normally all chips from the same family use the same address base. E.g. All current NVidia chips use PCI←_BASE_ADDRESS_1 which is the second address shown in lspci -v.

Libraries

The libraries are supplied as shared versions. The libraries are to be found in the subdirectory lib. The command

```
make install
```

will copy the appropriate library file for your frame grabber to the standard library path on your computer e.g. /usr/lib depending on the prefix you passed to

```
./configure
```

Alternatively you may leave the libraries in the lib subdirectory or copy the library to any other directory but you will have to set the environment variable LD_LIBRARY_PATH to include this directory.

E.g. for the bash shell:

```
>export LD_LIBRARY_PATH=$ LD_LIBRARY_PATH:/mypath/lib
```

Yet another alternative is to change the configuration file for your ldconfig command or use the mvldconfig script in the MATRIX VISION bin directory. The shared libraries contain the version number within the name. E.g. if the library for the mvSIGMA-SLC is called libslc.so.3.22.0, you will find symbolic links to the libraries within the lib directory:

```
>ls -l lib
drwxrwxr-x 3 technik users 1024 Sep 25 17:20 .
drwxrwxrwX 21 technik users 2048 Sep 25 17:26 ..
lrwxrwxrwx 1 technik users 16 Sep 24 09:53 libslc.so -> libslc.so.3.22.0
lrwxrwxrwx 1 technik users 16 Sep 24 09:54 libslc.so.3 -> libslc.so.3.22.0
-rwxrwxr-x 1 technik users 561114 Aug 30 13:11 libslc.so.3.22.0
>
```

The following is a list of the libraries available and their function. The names of the libraries actually supplied may vary depending on the current version/build number (e.g. libtitan.so.1.0.2 instead of libtitan.so.1.0.1).

The term mvTITAN series is used for all mvTITAN and mvGAMMA products.
7.3 Linux

7.3.3 Writing your own applications

A number of test applications have been included to show you how to write programs that use the MATRIX VISION frame grabber driver. Some of these test programs may not be contained in the driver version you receive but at least linuxtest should always be available.

**linuxtest**

The files contained in the subdirectory `linuxtest` can be used to make simple test programs for all the MATRIX VISION frame grabbers. An image will be shown on the screen in hard live or soft live mode. You can make versions without any display at all (nodisp), VGA (vga), X (x11) or Frame buffer (fb) versions. The top-level configure script will already have made the test programs for you and you will find them in the subdirectories called nodisp, vga, x11 and fb. Alternatively you may change directory to the linuxtest subdirectory and type

```
>./configure
> make
```

to (re)make the programs. The configure script will examine your system and only make test programs according to the libraries installed on your system. I.e. the SVGA versions will only be made if the SVGA library has been installed and can be found by the configure script. Likewise, if you only have the MATRIX VISION libraries for the mvTITAN series, the test programs for mvTITAN will be the only ones created. If you install other display libraries later you will have to repeat the configure, make process so that they are recognised.

Example for mvTITAN series: The command

```
>xtitantest --help
```

will show a list of possible parameters for the test program. You will also need a file called `titantest.ini` which contains information about the color depth, camera definitions etc. A number of example INI files are supplied within the `linuxtest/examples` subdirectory. You should copy and rename or use symbolic links as appropriate. In general, the INI file required is named after the test program:

See the README file in the linuxtest directory for more details.

**sdltest**

The files contained in the subdirectory `sdltest` can be used to make simple SDL based test programs for all the MATRIX VISION frame grabbers. These programs are heavily based on `linuxtest` but use SDL to create and manage a Window. If you are running X and you have the correct SDL libraries installed on your system these programs will be created by the top level configure script. Again, you may change to the subdirectory and call configure and make there if you wish.

See the README file in the sdltest directory for more details.
VGA applications

When using the VGA library for display you should add an extra, as yet undocumented function to the INI file used to initialize the frame grabber:

```
UseLibVga 1024 768 8 0
```

The parameters are

- x resolution in pixels for the display
- y resolution in pixels for the display
- color depth in pixels for the display
- set to 0 if using the first 3 parameters

Alternatively you may specify a number representing the VGA mode as the fourth parameter and leave the first three parameters as 0. See the VGA library documentation for details about the modes. E.g.

```
UseLibVga 0 0 0 12
```

will set VGA mode 12.

**Note**

In order to change VGA modes you need access rights as root.

Frame buffer applications

When using the frame buffer display library (libmvfb) for display you will need to be running a valid framebuffer otherwise you will get unpredictable results and/or crashes. The kernel must contain frame buffer support - either vesafb (for VESA-compatible cards) or a frame buffer that supports your graphic card directly (e.g. rivafb).

7.3.5 Copyrights

We have tried our best to adhere to the terms of all software licenses. Please contact us if you feel we have misinterpreted anything.

Some of the original work on the matrixfg, sigma and titan kernel modules was based on the source code for the btvv driver, in particular the routines to detect the VGA card and some memory management routines. Since the original code was released under the terms of the GNU General Public License (GPL), we are happy to supply our kernel module source code and give full credit to the original authors (see header files). Our kernel driver source code is therefore also released under the terms of the GPL, the full text of which may be found in the file GPL/COPYING. Loading these kernel modules will not "taint" your kernel!

The xsigma kernel module is an exception. Not all the source code for this module is supplied and this module is not released under the GPL. It contains no “foreign” GPL code (i.e. copyright not held by MATRIX VISION GmbH) or code derived from “foreign” GPL code. If this is a problem for you please use the sigma kernel module and libslc instead. The GPL version has a slightly reduced functionality.
The allocator module is Copyright © 1998 Alessandro Rubini and is released under the terms of the GPL. We have made some alterations which are documented in the source files.

The tutorial and test program for X are Copyright © 1998-2005 MATRIX VISION GmbH and are distributed under the terms of the GPL. Please note that KDE/QT programs you develop may require a commercial license from TrollTech unless you also distribute them under the GPL or QPL. See http://www.trolltech.no or http://www.trolltech.com for more details. The VGA library used by some of the display libraries is free software and may be distributed and modified without restriction. The X libraries appear to be distributed under similar terms. See the header files on your Linux system for details.

The GNU C library is distributed under the terms of the GNU Library General Public License (GLPL). Applications. According to the terms of this license, “work that uses the library” is not restricted by the license. The full text of this license may be found in the file GPL/GLPL.html.

The frame grabber shared libraries for MATRIX VISION frame grabbers are based solely on the original Windows source code developed by us and contain no GPL code. These files are Copyright © 1992-2005 MATRIX VISION GmbH and the source code is not included here. You may distribute the binaries together with your own applications if you include the above Copyright notice in your documentation. Programs that use kernel services and functions are not derived form the kernel and are therefore not covered by the GPL like the kernel sources themselves.

The library libtitan.so.x.x for the mvTITAN series of frame grabbers uses some binary code from the Trimedia® code relocation library (libload.a) supplied by Philips. These functions are © Copyright Philips Semiconductors Trimedia Product Group 1997-2000.

7.4 Settings behavior during startup

Settings contain all the parameters that are needed to prepare and program the device for the image capture. Every image can be captured with completely different set of parameters. In almost every case, these parameters are accessible via a property offered by the device driver. A setting e.g. might contain

- the gain to be applied to the analog to digital conversion process for analog video sources or

- the AOI to be captured from the incoming image data.

So for the user a setting is the one an only place where all the necessary modifications can be applied to achieve the desired form of data acquisition.

Now, whenever a device is opened, the driver will execute following procedure:
Figure 10: wxPropView - Device setting start procedure

- Please note that each setting location step in the figure from above internally contains two search steps. First the framework will try to locate a setting with user scope and if this can’t be located, the same setting will be searched with global (system-wide) scope. Under Windows® this e.g. will access either the HKEY_CURRENT_USER or (in the second step) the HKEY_LOCAL_MACHINE branch in the Registry.

- Whenever storing a product specific setting, the device specific setting of the device used for storing will be deleted (if existing). So when the user is currently working with a device 'VD000001' belonging to the product group 'VirtualDevice' and there is a setting exclusively for this device storing a product specific setting now will automatically delete the setting for 'VD000001'. Otherwise a product specific setting would never be loaded as a device specific setting will always be found first.

- The very same thing will also happen when opening a device from any other application! wxPropView (p.42) does not behave in a special way but only acts as an arbitrary user application.

- Whenever storing a device family specific setting, the device specific or product specific setting of the device used for storing will be deleted (if existing). See above to find out why.

- Under Windows® the driver will not look for a matching XML file during start-up automatically as the native storage location for settings is the Windows® Registry. This must be loaded explicitly by the user by using the appropriate API function offered by the SDK. However, under Linux XML files are the only setting formats understood by the driver framework thus here the driver will also look for them at start-up. The device specific setting will be an XML file with the serial number of the device as the file name, the product specific setting will be an XML file with the product string as the filename, the device family specific setting will be an XML file with the device family name as the file name. All other XML files containing settings will be ignored!

- Only the data contained in the lists displayed as "Image Setting", "Digital I/O" and "Device Specific Data" under wxPropView (p.42) will be stored in these settings!

- Restoring of settings previously stored works in a similar way. After a device has been opened the settings will be loaded automatically as described above.
A detailed description of the individual properties offered by a device will not be provided here but can be found in the C++ API reference, where descriptions for all properties relevant for the user (grouped together in classes sorted by topic) can be found. As `wxPropView` doesn't introduce new functionality but simply evaluates the list of features offered by the device driver and lists them any modification made using the GUI controls just calls the underlying function needed to write to the selected component. `wxPropView` also doesn't know about the type of component or e.g. the list of allowed values for a property. This again is information delivered by the driver and therefore can be queried by the user as well without the need to have special inside information. One version of the tool will always be delivered in source so it can be used as a reference to find out how to get the desired information from the device driver.
8 Technical data

8.1 mvDELTA

8.1.1 Connectors

![Connectors mvDELTA](image1)

Figure 1: Connectors mvDELTA

![Connectors mvDELTAe](image2)

Figure 2: Connectors mvDELTAe

8.1.2 Technical specifications

<table>
<thead>
<tr>
<th></th>
<th>mvDELTA</th>
<th>mvDELTAe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video input</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input signal</td>
<td>Interlaced, monochrome</td>
<td>Interlaced, color</td>
</tr>
<tr>
<td>50Hz</td>
<td>CCIR, PAL, SECAM, S-VHS, Y/C</td>
<td></td>
</tr>
<tr>
<td>60 Hz</td>
<td>RS-170, RS-330, NTSC</td>
<td></td>
</tr>
<tr>
<td>Number of video inputs</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Pixel clock</td>
<td>17.7 MHz @ 50 Hz</td>
<td>14.3 MHz @ 60 Hz</td>
</tr>
<tr>
<td>Resolution</td>
<td>768 x 576 pixels (50Hz)</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------</td>
<td></td>
</tr>
<tr>
<td>Digitalisation</td>
<td>768 x 576 pixels (50Hz)</td>
<td></td>
</tr>
<tr>
<td>Storage</td>
<td>0..768 x 0..576 pixels (50Hz)</td>
<td></td>
</tr>
<tr>
<td>Grey scale</td>
<td>8 bit</td>
<td></td>
</tr>
<tr>
<td>Color</td>
<td>24 bit true color</td>
<td></td>
</tr>
<tr>
<td>Memory formats</td>
<td>32 bit RGB, 24 bit RGB, 16 bit RGB, 15 bit RGB, 16 bit YUV packed, 16 bit planar</td>
<td></td>
</tr>
<tr>
<td>Image memory</td>
<td>Main memory of PC or image memory of VGA</td>
<td></td>
</tr>
<tr>
<td>Interface</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bus</td>
<td>PCI bus</td>
<td></td>
</tr>
<tr>
<td>Transfer rate</td>
<td>DMA, 0-wait bursts, max. 132 MB/sec</td>
<td></td>
</tr>
<tr>
<td>Intel FX chip set</td>
<td>Approx. 90 MB/sec</td>
<td></td>
</tr>
<tr>
<td>Current consumption</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCI 5V</td>
<td>max. 0.21A</td>
<td></td>
</tr>
<tr>
<td>PCIe 3.3V</td>
<td>max. 0.15A</td>
<td></td>
</tr>
<tr>
<td>PCIe 12V</td>
<td>max. 0.2A</td>
<td></td>
</tr>
<tr>
<td>Environmental conditions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ambient temperature</td>
<td>0 up to 50 C</td>
<td></td>
</tr>
<tr>
<td>Storage temperature</td>
<td>-20 up to 70 C</td>
<td></td>
</tr>
<tr>
<td>Humidity</td>
<td>10 up to 90 % non-condensing</td>
<td></td>
</tr>
<tr>
<td>Dimensions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Length</td>
<td>123 mm</td>
<td></td>
</tr>
<tr>
<td>Width</td>
<td>95 mm</td>
<td></td>
</tr>
</tbody>
</table>

8.2 mvSIGMA-SLC

8.2.1 Connectors

![Connectors mvSIGMA-SLC](image)

Default settings are bold.
8.2.1.1 J1: D-Sub HD26 digital input / output & power (rev. 5.00 and higher)

Figure 4: D-Sub HD26 pin numbering (female)

<table>
<thead>
<tr>
<th>Pin.</th>
<th>Signal</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+12V</td>
<td>12V camera power supply (0.7A)</td>
</tr>
<tr>
<td>2</td>
<td>Video 1</td>
<td>Video input 1, (with S-VHS mode -&gt; Y1) parallel to cinch jack (cable inside)</td>
</tr>
<tr>
<td>3</td>
<td>Video 2</td>
<td>Video input 2, (with S-VHS mode -&gt; Y2) parallel to S-Video jack (pin 3)</td>
</tr>
<tr>
<td>4</td>
<td>Video 3</td>
<td>Video input 3, (with S-VHS mode -&gt; Y3)</td>
</tr>
<tr>
<td>5</td>
<td>Video 4</td>
<td>Video input 4, (with S-VHS mode -&gt; Y4)</td>
</tr>
<tr>
<td>6</td>
<td>GP-OUT 2</td>
<td>Digital TTL output port 2</td>
</tr>
<tr>
<td>7</td>
<td>GP-OUT 3</td>
<td>Digital TTL output port 3</td>
</tr>
<tr>
<td>8</td>
<td>Trigger-In</td>
<td>External trigger input / digital input</td>
</tr>
<tr>
<td>9</td>
<td>GP-IN 4</td>
<td>Digital TTL input port 4</td>
</tr>
<tr>
<td>10</td>
<td>Ground camera supply</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>GP-IN 5</td>
<td>Digital TTL input port 5</td>
</tr>
<tr>
<td>12</td>
<td>Ground Video 1</td>
<td>Ground video input 1 and chroma input 1</td>
</tr>
<tr>
<td>13</td>
<td>Ground Video 2</td>
<td>Ground video input 2 and chroma input 2</td>
</tr>
<tr>
<td>14</td>
<td>Ground Video 3</td>
<td>Ground video input 3 and chroma input 3</td>
</tr>
<tr>
<td>15</td>
<td>Ground Video 4</td>
<td>Ground video input 4 and chroma input 4</td>
</tr>
<tr>
<td>16</td>
<td>GP-IN 6</td>
<td>Digital TTL input port 6</td>
</tr>
<tr>
<td>17</td>
<td>Ground ext Trigger</td>
<td>Ground ext. Trigger / digital input and output ports 0/1</td>
</tr>
<tr>
<td>18</td>
<td>GP-IN 7</td>
<td>Digital TTL input port 7</td>
</tr>
<tr>
<td>19</td>
<td>GP-OUT 0</td>
<td>Digital TTL output port 0</td>
</tr>
<tr>
<td>20</td>
<td>GP-OUT 1</td>
<td>Digital TTL output port 1</td>
</tr>
<tr>
<td>21</td>
<td>Chroma input 1</td>
<td>Chroma input 1 (C1) with S-VHS mode</td>
</tr>
<tr>
<td>22</td>
<td>Chroma input 2</td>
<td>Chroma input 2 (C2) with S-VHS mode, parallel to S-Video jack (pin 3)</td>
</tr>
<tr>
<td>23</td>
<td>Chroma input 3</td>
<td>Chroma input 3 (C3) with S-VHS mode</td>
</tr>
<tr>
<td>24</td>
<td>Chroma input 4</td>
<td>Chroma input 4 (C4) with S-VHS mode</td>
</tr>
<tr>
<td>25</td>
<td>Video out</td>
<td>Y out</td>
</tr>
<tr>
<td>26</td>
<td>Video out</td>
<td>only in S-VHS mode: C out</td>
</tr>
</tbody>
</table>

8.2.1.2 J4: Power supply (floppy)

You can connect a free power supply cable for floppy drives on connector J4 to increase the available current on the power supply pins on J1 to 2A.

Figure 5: J4

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+12V</td>
</tr>
<tr>
<td>2</td>
<td>Ground</td>
</tr>
<tr>
<td>3</td>
<td>Ground</td>
</tr>
<tr>
<td>4</td>
<td>Not connected</td>
</tr>
</tbody>
</table>
8.2.1.3 Termination of video input

With switch S1 the termination of the video inputs can be switched on/off. Default adjustment of switch S1 is on.

Meaning of switch S1, 75Ohm terminator

![Figure 6: Video](image)

<table>
<thead>
<tr>
<th>Position</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CVBS0 / Y0</td>
</tr>
<tr>
<td>2</td>
<td>CVBS1 / Y1</td>
</tr>
<tr>
<td>3</td>
<td>CVBS2 / Y2</td>
</tr>
<tr>
<td>4</td>
<td>CVBS3 / Y3</td>
</tr>
<tr>
<td>5</td>
<td>C0</td>
</tr>
<tr>
<td>6</td>
<td>C1</td>
</tr>
<tr>
<td>7</td>
<td>C2</td>
</tr>
<tr>
<td>8</td>
<td>C3</td>
</tr>
</tbody>
</table>

8.2.2 Technical specifications

<table>
<thead>
<tr>
<th></th>
<th>mvSIGMA-SLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video input</td>
<td>Interlaced, monochrome</td>
</tr>
<tr>
<td>Input signal</td>
<td>Interlaced, color</td>
</tr>
<tr>
<td>50 Hz</td>
<td>CCIR, PAL, SECAM, S-VHS, Y/C</td>
</tr>
<tr>
<td>60 Hz</td>
<td>RS-170, RS-330, NTSC</td>
</tr>
<tr>
<td>Number of video inputs</td>
<td>4</td>
</tr>
<tr>
<td>Sync signal</td>
<td>External source</td>
</tr>
<tr>
<td>Pixel clock</td>
<td>17.7 MHz @ 50 Hz</td>
</tr>
<tr>
<td></td>
<td>14.3 MHz @ 60 Hz</td>
</tr>
<tr>
<td>Resolution</td>
<td>768 x 576 pixels (50 Hz)</td>
</tr>
<tr>
<td>Digitalisation</td>
<td>0..768 x 0..576 pixels (50 Hz)</td>
</tr>
<tr>
<td>Grey scale</td>
<td>8 bit</td>
</tr>
<tr>
<td>Color</td>
<td>24 bit true color</td>
</tr>
<tr>
<td>Memory formats</td>
<td>32 bit RGB, 24 bit RGB, 16 bit RGB, 15 bit RGB, 16 bit YUV packed, 16 bit planar</td>
</tr>
<tr>
<td>Image memory</td>
<td>Main memory of PC or image memory of VGA</td>
</tr>
<tr>
<td>Interface</td>
<td>PCI bus</td>
</tr>
<tr>
<td>Transfer rate</td>
<td>DMA, 0-wait bursts, max. 132 MB/sec</td>
</tr>
</tbody>
</table>
## 8.2.3 Device Feature And Property List

### 8.3 mvSIGMA-SQ/-SQe

#### 8.3.1 Connectors

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel FX chip set</td>
<td>Approx. 90 MB/sec</td>
</tr>
<tr>
<td>Current consumption</td>
<td></td>
</tr>
<tr>
<td>PCI 5V</td>
<td>max. 0.3A</td>
</tr>
<tr>
<td>PCI 12V</td>
<td>max. 0.05 A (without camera)</td>
</tr>
<tr>
<td>Camera supply</td>
<td>Via PCI: 12V max. 0.7A fused</td>
</tr>
<tr>
<td></td>
<td>Via additional power plug: 12V max. 1.5V fused</td>
</tr>
<tr>
<td>Environmental conditions</td>
<td></td>
</tr>
<tr>
<td>Ambient temperature</td>
<td>0 up to 50 C</td>
</tr>
<tr>
<td>Storage temperature</td>
<td>-20 up to 70 C</td>
</tr>
<tr>
<td>Humidity</td>
<td>10 up to 90 % non-condensing</td>
</tr>
<tr>
<td>Dimensions</td>
<td></td>
</tr>
<tr>
<td>Length</td>
<td>125 mm</td>
</tr>
<tr>
<td>Width</td>
<td>76 mm</td>
</tr>
</tbody>
</table>

![Image of mvSIGMA-SQ/-SQe connectors](image.jpg)

Figure 7: Connectors mvSIGMA-SQ (Rev. 1.0)
8.3 mvSIGMA-SQ/-Sqe

Figure 8: Connectors mvSIGMA-SQe (Rev. 1.0)

Default settings are bold.

8.3.1.1 D-Sub HD26BU digital input / output & power (rev. 5.00 and higher)

Figure 9: D-Sub HD26BU pin numbering (female)

<table>
<thead>
<tr>
<th>Pin</th>
<th>Jack J6</th>
<th>Jack J7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12V/0.6A camera power supply</td>
<td>12V/0.6A camera power supply</td>
</tr>
<tr>
<td>2</td>
<td>Video 0 / Luminance 0</td>
<td>Video 8 / Luminance 2</td>
</tr>
<tr>
<td>3</td>
<td>Video 1 / Luminance 1</td>
<td>Video 9 / Luminance 3</td>
</tr>
<tr>
<td>4</td>
<td>Video 2</td>
<td>Video 10</td>
</tr>
<tr>
<td>5</td>
<td>Video 3</td>
<td>Video 11</td>
</tr>
<tr>
<td>6</td>
<td>DIG-IN 0 / Trigger-In (Decoder 0)</td>
<td>DIG-IN 8 / Trigger-In (Decoder 2)</td>
</tr>
<tr>
<td>7</td>
<td>DIG-IN 1 (Decoder 0)</td>
<td>DIG-IN 9 (Decoder 2)</td>
</tr>
<tr>
<td>8</td>
<td>DIG-OUT 0 (Decoder 0)</td>
<td>DIG-OUT 4 (Decoder 2)</td>
</tr>
<tr>
<td>9</td>
<td>DIG-OUT 1 (Decoder 0)</td>
<td>DIG-OUT 5 (Decoder 2)</td>
</tr>
<tr>
<td>10</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>11</td>
<td>DIG-IN 4 / Trigger-In (Decoder 1)</td>
<td>DIG-IN 12 / Trigger-In (Decoder 3)</td>
</tr>
<tr>
<td>12</td>
<td>DIG-IN 5 (Decoder 1)</td>
<td>DIG-IN 13 (Decoder 3)</td>
</tr>
<tr>
<td>13</td>
<td>DIG-IN 6 (Decoder 1)</td>
<td>DIG-IN 14 (Decoder 3)</td>
</tr>
<tr>
<td>14</td>
<td>DIG-IN 7 (Decoder 1)</td>
<td>DIG-IN 15 (Decoder 3)</td>
</tr>
<tr>
<td>15</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>16</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>17</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>18</td>
<td>DigOut VCC (&lt; 24V)</td>
<td>DigOut VCC (&lt; 24V)</td>
</tr>
<tr>
<td>19</td>
<td>DIG-IN 2 (Decoder 0)</td>
<td>DIG-IN 10 (Decoder 2)</td>
</tr>
<tr>
<td>20</td>
<td>DIG-IN 3 (Decoder 0)</td>
<td>DIG-IN 11 (Decoder 2)</td>
</tr>
<tr>
<td>21</td>
<td>Video 4 / Chrominance 0 (Decoder 0)</td>
<td>Video 12 / Chrominance 2 (Decoder 2)</td>
</tr>
<tr>
<td>22</td>
<td>Video 5 / Chrominance 1 (Decoder 1)</td>
<td>Video 13 / Chrominance 3 (Decoder 3)</td>
</tr>
</tbody>
</table>
8.3.1.1.1 Characteristics of the digital I/Os

**Digital inputs**

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input impedance</td>
<td>10 KΩ</td>
</tr>
<tr>
<td>Downstream pre-divider</td>
<td>1:4.7</td>
</tr>
<tr>
<td>Trigger level</td>
<td>digital adjustable</td>
</tr>
<tr>
<td>Max. input voltage</td>
<td>24 V</td>
</tr>
</tbody>
</table>

![Figure 10: digital input of decoder 0](image)

**Digital outputs**

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output voltage</td>
<td>12 V or DigOut VCC</td>
</tr>
<tr>
<td>Max. output current</td>
<td>200 mA per pin</td>
</tr>
</tbody>
</table>

The digital outputs are conducted as so called high side drivers. For the default switching voltage the PC's internal supply of 12V is provided. If required this can be increased up to 24V by conducting an external voltage on pin 18.
8.3.1.2 J8: Power supply (floppy)

You can connect a free power supply cable for floppy drives on connector J8 to increase the available current on the power supply pins on J6 and J7 to 2A.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+12V</td>
</tr>
<tr>
<td>2</td>
<td>Ground</td>
</tr>
<tr>
<td>3</td>
<td>Ground</td>
</tr>
<tr>
<td>4</td>
<td>Not connected</td>
</tr>
</tbody>
</table>

8.3.1.3 Termination of video input

(since mvSIGMA-SQ Rev. 2.0 and mvSIGMA-SQe Rev. 1.0)

With switch S1 (above) the termination of the video inputs can be switched on/off. Default adjustment of switch S1 is on.

Meaning of switch S1, 75Ohm terminator
With switch S2 (below) the termination of the video inputs can be switched on/off. Default adjustment of switch S2 is on.

Meaning of switch S2, 75Ohm terminator

<table>
<thead>
<tr>
<th>Position</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Y0</td>
</tr>
<tr>
<td>2</td>
<td>Y1</td>
</tr>
<tr>
<td>3</td>
<td>Y2</td>
</tr>
<tr>
<td>4</td>
<td>Y3</td>
</tr>
<tr>
<td>5</td>
<td>Y4/C0</td>
</tr>
<tr>
<td>6</td>
<td>Y5/C1</td>
</tr>
<tr>
<td>7</td>
<td>Y6</td>
</tr>
<tr>
<td>8</td>
<td>Y7</td>
</tr>
</tbody>
</table>

Figure 13: video

<table>
<thead>
<tr>
<th>Position</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Y8</td>
</tr>
<tr>
<td>2</td>
<td>Y9</td>
</tr>
<tr>
<td>3</td>
<td>Y10</td>
</tr>
<tr>
<td>4</td>
<td>Y11</td>
</tr>
<tr>
<td>5</td>
<td>Y12/C2</td>
</tr>
<tr>
<td>6</td>
<td>Y13/C3</td>
</tr>
<tr>
<td>7</td>
<td>Y14</td>
</tr>
<tr>
<td>8</td>
<td>Y15</td>
</tr>
</tbody>
</table>

8.3.2 Technical specifications

<table>
<thead>
<tr>
<th></th>
<th>mvSIGMA-SQ</th>
<th>mvSIGMA-SQe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video input</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input signal</td>
<td>Interlaced, monochrome</td>
<td>Interlaced, color</td>
</tr>
<tr>
<td>50 Hz</td>
<td>CCIR, PAL, SECAM, S-VHS, Y/C</td>
<td>RS-170, RS-330, NTSC</td>
</tr>
<tr>
<td>60 Hz</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of video inputs</td>
<td>16, terminated with 75Ohm by default</td>
<td></td>
</tr>
<tr>
<td>Sync signal</td>
<td>External source</td>
<td></td>
</tr>
<tr>
<td>Resolution</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Digitalisation</td>
<td>768 x 576 (PAL) / 640 x 480 (NTSC)</td>
<td></td>
</tr>
<tr>
<td>Grey scale</td>
<td>8 bit</td>
<td></td>
</tr>
<tr>
<td>Color</td>
<td>24 bit true color</td>
<td></td>
</tr>
<tr>
<td>Memory formats</td>
<td>32 bit RGB, 24 bit RGB, 16 bit RGB, 15 bit RGB, 16 bit YUV packed, 16 bit planar</td>
<td></td>
</tr>
<tr>
<td>Image memory</td>
<td>Host RAM or overlay</td>
<td></td>
</tr>
</tbody>
</table>
### Interface

<table>
<thead>
<tr>
<th>Bus</th>
<th>PCI bus</th>
<th>PCI Express x1 bus</th>
</tr>
</thead>
</table>

### Current consumption

<table>
<thead>
<tr>
<th>Source</th>
<th>PCI 5V</th>
<th>PCI 12V</th>
<th>PCIE 3.3V</th>
<th>PCIE 12V</th>
<th>Camera supply</th>
</tr>
</thead>
<tbody>
<tr>
<td>max.</td>
<td>1.5A</td>
<td>0.2A</td>
<td>max. 0.15A</td>
<td>max. 0.6A</td>
<td>Via PCI: 12V max. 0.7A fused</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Via additional power plug: 12V max. 1.5V fused</td>
</tr>
</tbody>
</table>

### Environmental conditions

<table>
<thead>
<tr>
<th>Condition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ambient temperature</td>
<td>0 up to 50 °C</td>
</tr>
<tr>
<td>Storage temperature</td>
<td>-20 up to 70 °C</td>
</tr>
<tr>
<td>Humidity</td>
<td>10 up to 90% non-condensing</td>
</tr>
</tbody>
</table>

### Dimensions

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>150 mm</td>
<td>170 mm</td>
</tr>
<tr>
<td>Width</td>
<td>106 mm</td>
<td>111 mm</td>
</tr>
</tbody>
</table>
9 Application Usage

9.1 wxPropView

wxPropView (p. 42) is an interactive GUI tool to acquire images and to configure the device and to display and modify the device properties of MATRIX VISION GmbH hardware. After the installation you can find wxPropView (p. 42)

- as an icon with the name "wxPropView" on the desktop (Windows) or
- in "~/mvimpact-acquire/apps/mvPropView/x86" (Linux).

9.1.1 How to work with wxPropView

wxPropView - Introduction:

9.1.1.1 First View of wxPropView

wxPropView (p. 42) consists of several areas:
Figure 1: wxPropView started

- "Menu Bar" (to work with wxPropView on page 42 using the menu)
- "Upper Tool Bar" (to select and initialize a device, acquire images, play a recorder sequence)
- "Left Tool Bar" (to hide and show parts of the GUI)
- "Status Tool Bar"

- "Main Window" with
  - "Grid" (tree control with the device settings accessible by the user)
  - "Display" (for the acquired images)

- "Analysis" (information about whole images or an AOI)

By clicking on F1 you will get the HELP dialog.

Now, you can initialize a device by
After having successfully initialized a device the tree control in the lower left part of the "Main Window" will display the properties (settings or parameters) (according to the "interface layout") accessible by the user.

You've also got the possibility to set your "User Experience". According to the chosen experience, the level of visibility is different:

- **Beginner** (basic camera settings/properties are visible)
- **Expert** (e.g. all advanced image processing are visible)
- **Guru** (all settings/properties are visible)

Properties displayed in light grey cannot be modified by the user. Only the properties, which actually have an impact on the resulting image, will be visible. Therefore, certain properties might appear or disappear when modifying another properties.

To permanently commit a modification made with the keyboard the **ENTER** must be pressed. If leaving the editor before pressing **ENTER** will restore the old value.

### 9.1.1.2 How to see the first image

As described earlier, for each recognized device in the system the devices serial number will appear in the drop down menu in the upper left corner of the "Upper Tool Bar". When this is the first time you start the application after the system has been booted this might take some seconds when working with devices that are not connected to the host system via PCI or PCIe.

Once you have selected the device of your choice from the drop down menu click on the "Use" button to open it.

When the device has been opened successfully, the remaining buttons of the dialog will be enabled:

**Note**

Following screenshots are representative and were made using a mvHYPERION frame grabber as the capturing device.
Now, you can capture an image ("Acquisition Mode": "SingleFrame") or display live images ("Continuous"). Just

- select an "Acquisition Mode" e.g. "SingleFrame" and
- click the "Acquire" button.

**Note**

The techniques behind the image acquisition can be found in the developers sections.

The frame rate depends on

- the camera,
- the pixel clock of the sensor

Figure 2: wxPropView - First start
9.1.1.2.1 Record Mode

It is also possible to record image sequences using wxPropView.

1. For this, you have to set the size of the recorder in “System Settings -> RequestCount” e.g. to 100. This will save the last 100 requests in the request queue of the driver, i.e. the image data including the request info like frame number, time stamp, etc.

2. Afterwards you can start the recording by clicking the Rec. button.

3. With the Next and Prev. buttons you can display the single images.

If you switched on the request info overlay (right-click on the display area and select the entry to activate this feature), these information will be displayed on the image, too. With the timestamp you can see the interval of the single frames in microseconds.

9.1.1.2.2 Hard Disk Recording

You can save acquired images to the hard disk the following way:

1. In the "Menu Bar" click on "Capture -> Recording -> Setup Hard Disk Recording".

2. Confirm with "Yes".

3. Afterwards select the target folder for the images.

4. Finally, choose the file format of the acquired images.

9.1.1.3 Using the analysis plots

With the analysis plots you have the possibility to get image details and to export them (p. 53).

9.1.1.3.1 Spatial noise histogram

The spatial noise histogram calculates and evaluates statistically the difference between two neighbouring pixels in vertical and horizontal direction. I.e. it shows the sensor's spatial background pattern like the sensitivity shifts of each pixel. An ideal sensor or camera has a spatial noise of zero. However, you have to keep in mind the temporal noise as well.
9.1 wxPropView

Read: Channel::Direction (Mean difference, most frequent value count/value, Standard deviation)

**Example:** For a single channel (Mono) image the output of ‘C0Hor(3.43, 5086/ 0, 9.25), C0Ver(3.26, 4840/ 0, 7.30)’ will indicate that the mean difference between pixels in horizontal direction is 3.43, the most frequent difference is 0 and this difference is present 5086 times in the current AOI. The standard deviation in horizontal direction is 9.25. The C0Ver value list contains the same data but in vertical direction.

9.1.1.3.2 Temporal noise histogram

The temporal noise histogram shows the changes of a pixel from image to image. This method is more stable because it is relatively independent from the image content. By subtracting two images, the actual structure is eliminated, leaving the change of a pixel from image to image, that is, the noise. When capturing images, all parameters must be frozen, all automatic mechanisms have to be turned off and the image may not have underexposed or saturated areas. However, there are no picture signals without temporal noise. Light is a natural signal and the noise always increases with the signal strength. If the noise only follows the natural limits, then the camera is good. Only if additional noise is added the camera or the sensor has errors.
Read: Channel# (Mean difference, most frequent value count/ value, Standard deviation)

Example: For a single channel(Mono) image the output of 'C0(3.43, 5086/0, 9.25) will indicate that the mean difference between pixels in 2 consecutive images is 3.43, the most frequent difference is 0 and this difference is present 5086 times in the current AOI. The standard deviation between pixels in these 2 images is 9.25. Please note the impact of the 'Update Interval' in this plot: It can be used to define a gap between 2 images to compare. E.g. if the update interval is set to 2, the differences between image 1 and 3, 3 and 5, 5 and 7 etc. will be calculated. In order to get the difference between 2 consecutive images the update interval must be set to 1!

9.1.1.4 Storing and restoring settings

When wxPropView (p. 42) is started for the first time, the values of properties set to their default values will be displayed in green to indicate that these values have not been modified by the user so far. Modified properties (even if the value is the same as the default) will be displayed in black.

Settings can be stored in several ways (via the "Menu Bar": "Action -> Capture Settings -> Save Active Device Settings"): 

- "As Default Settings For All Devices Belonging To The Same Family (Per User Only)". As the start-up parameters for every device belonging to the same family, e.g. for mvBlueCOUGAR-X, mvBlueCOUGAR-XD.
- "As Default Settings For All Devices Belonging To The Same Family And Product Type". As the start-up parameters for every device belonging to the same product, e.g. for any mvBlueCOUGAR-X but not for mvBlueCOUGAR-XD.
- "As Default Settings For This Device(Serial Number)". As the start-up parameters for the currently selected device.
- "To A File". As an XML file that can be used e.g. to transport a setting from one machine to another or even to use the settings configured for one platform on another (Windows <-> Linux).
During the startup of a device, all these setting possibilities show different behaviors. The differences are described in chapter **Settings behavior during startup** (p. 29).

Restoring of settings previously stored works in a similar way. After a device has been opened the settings will be loaded automatically as described in **Settings behavior during startup** (p. 29).

However, at runtime the user has different load settings possibilities (via the "Menu Bar": "Action -> Capture Settings -> Load Active Device Settings"):

- explicitly load the device family specific settings stored on this machine (from "The Default Settings Location For This Devices Family (Per User Only)"")
- explicitly load the product specific settings stored on this machine (from "The Default Settings Location For This Devices Family And Product Type")
- explicitly load the device specific settings stored on this machine (from "The Default Settings Location For This Device(Serial Number)"")
- explicitly load device family specific settings from a XML file previously created ("From A File")

**Note**

With "Action -> Capture Settings -> Manage..." you can delete the settings which were saved on the system.

![Figure 9: wxPropView - Restoring settings](image)

9.1.1.5 Properties

All properties and functions can be displayed in the list control on the lower left side of the dialog. To modify the value of a property select the edit control right of the properties name. Property values, which refer to the default value of the device, are displayed in green. A property value once modified by the user will be displayed in black (even if the value itself has not changed). To restore its default value of a single property:

- right click on the name of the property and
- select "Restore Default".
To restore the default value for a complete list (which might include sub-lists)

- right click on the name of a list and
- select “Restore Default”.

In this case a popup window will be opened and you have to confirm again.

![Figure 10: wxPropView - Restore the default value of a property](image)

Most properties store one value only, thus they will appear as a single entry in the property grid. However, properties are capable of storing more than one value, if this is desired. A property storing more than one value will appear as a parent list item with a **WHITE** background color (lists will be displayed with a grey background) and as many child elements as values stored by the property. The **PARENT** grid control will display the number of values stored by the property, every child element will display its corresponding value index.

If supported by the property, the user might increase or decrease the number of values stored by right clicking on the **PARENT** grid element. If the property allows the modification the pop up menu will contain additional entries now:

![Figure 11: wxPropView - A resizable property](image)

When a new value has been created it will be displayed as a new child item of the parent grid item:
Currently, only the last value can be removed via the GUI and a value can't be removed, when a property stores one value only.

Also the user might want to set all (or a certain range of) values for properties that store multiple values with a single operation. If supported by the property, this can also be achieved by right clicking on the PARENT grid element. If the property allows this modification the pop up menu will again contain additional entries.
It's possible to either set all (or a range of) elements of the property to a certain value **OR** to define a value range, that then will be applied to the range of property elements selected by the user. The following example will explain how this works:

![Figure 14: wxPropView - Setting multiple property values within a certain value range](image1)

In this sample the entries 0 to 255 of the property will be assigned the value range of 0 to 255. This will result in the following values **AFTER** applying the values:

![Figure 15: wxPropView - After applying the value range to a property](image2)
9.1 wxPropView

9.1.1.6 Methods

Method appears as entries in the tree control as well. However, their name and behavior differs significantly from the behavior of properties. The names of method objects will appear in ‘C’ syntax like e.g. ‘int function(char*, int)’. This will specific a function returning an integer value and expecting a string and an integer as input parameters. To execute a method object

- right click on the name of a method and
- select “Execute” from the popup menu:

![Figure 16: wxPropView - Calling a method object](image)

Parameters can be passed to methods by selecting the edit control left of a method object. Separate the parameters by blanks. So to call a function expecting a string and an integer value you e.g. might enter “testString 0” into the edit control left of the method.

The return value (in almost every case an error code as an integer) will be displayed in the lower right corner of the tree control. The values displayed here directly correspond the error codes defined in the interface reference and therefore will be of type TDMR_ERROR or TPROPHANDLING_ERROR.

9.1.1.7 Copy grid data to the clipboard

Since wxPropView (p. 42) version 1.11.0 it is possible to copy analysis data to the clipboard. The data will be copied in CSV style thus can be pasted directly into tools like Open Office™ or Microsoft® Office™.

Just

- right-click on the specific analysis grid when in numerical display mode and
- select “Copy grid to clipboard” from the pop up menu.
9.1.1.8 Import and Export images

wxPropView offers a wide range of image formats that can be used for exporting captured image to a file. Some formats e.g. like packed YUV 4:2:2 with 10 bit per component are rather special thus they can’t be stored into a file like e.g. offered by the BMP file header. When a file is stored in a format, that does not support this data type wxPropView will convert this image into something that matches the original image format as close as possible. This, however, can result in the loss of data. In order to allow the storage of the complete information contained in a captured image wxPropView allows to store the data in a raw format as well. This file format will just contain a binary dump of the image with no leader or header information. However, the file name will automatically be extended by information about the image to allow the restoring of the data at a later time.

All image formats, that can be exported can also be imported again. Importing a file can be done in 3 different ways:

- via the menu (via the "Menu Bar": "Action -> Load image..."
- by dragging an image file into an image display within wxPropView
- by starting wxPropView from the command line passing the file to open as a command line parameter (under Windows® e.g. "wxPropView.exe MyImage.png" followed by [ENTER])

When importing a "*.raw" image file a small dialog will pop up allowing the user to define the dimensions and the pixel format of the image. When the file name has been generated using the image storage function offered by wxPropView, the file name will be passed and the extracted information will automatically be set in the dialog thus the user simply needs to confirm this information is correct.
9.1.1.9 Setting up multiple display support and/or work with several capture settings in parallel

`wxPropView` (p. 42) is capable of

- dealing with multiple capture settings or acquisition sequences for a single device and in addition to that
- it can be configured to deal with multiple image displays.

For frame grabbers with multiple input channels this e.g. can be used to display live images from all input channels simultaneously. This even works if each input channel is connected to a different video signal in terms of resolution and timing.

The amount of parallel image displays can be configured via the command line parameter (p. 76) "dcx" and "dcy". In this step by step setup `wxPropView` (p. 42) has been started like this from the command line:

wxPropView dcx=1 dcy=2

This will result in 1 display in horizontal direction and 2 in vertical direction.

Since

mvlIMPACT Acquire 2.18.1

It is also possible to change the amount of display at runtime via "Settings -> Image Displays -> Configure Image Display Count".
Additional capture settings can be created via "Menu Bar": "Capture -> Capture Settings -> Create Capture Settings". The property grid will display these capture settings either in "Developers" or in "Multiple Settings View".

Now, in order to set up wxPropView (p. 42) to work with 2 instead of one capture setting,

1. Various additional capture setting can be created. In order to understand what a capture setting actually is please refer to
   • "Working with settings" chapter of the "mvIMPACT Acquire API" manuals.

Creating a capture setting is done via "Capture -> Capture Settings -> Create Capture Setting".
2. Then, the user is asked for the name of the new setting.

![Figure 21: wxPropView - Create capture setting - Choosing name](image)

3. And finally for the base this new setting shall be derived from.

![Figure 22: wxPropView - Create capture setting - Choosing base](image)

Afterwards, in this example we end up having 2 capture settings:

- a "Base" setting, which is always available
- a "NewSetting1", which has been derived from "Base".
As "NewSetting1" has been derived from "Base" changing a property in "Base" will automatically change this property in "NewSetting1" if this property has not already been modified in "NewSetting1". Again to get an understanding for this behaviour please refer to

- "Working with settings" chapter of the "mvIMPACT Acquire API" manuals.

Now, to set up wxPropView (p.42) to display all images taken using capture setting "Base" in one display and all image taken using capture setting "NewSetting1" in another display the capture settings need to be assigned to image displays via "Capture -> Capture Settings -> Assign To Display(s)".
By default a new setting when created will be assigned to one of the available displays in a round-robin scheme, thus when there are 3 displays, the first (Base) setting will be assigned to "Display 0", the next to "Display 1", the next to "Display 2" and a fourth setting will be assigned to "Display 0" again. The setting to display relationships can be customized via "Capture -> Capture Settings -> Assign to Display(s)".

As each image display keeps a reference to the request, this image belongs to the driver can't re-use the request buffer until a new request is blitted into this display. Thus, it might be necessary to increase the number of request objects the driver is working with if a larger number of displays are involved. The minimum number of requests needed is 2 times the amount of images displays. The number of requests used by the driver can be set up in the drivers property tree:
Finally, **wxPropView** must be configured in order to use all available capture settings in a round-robin scheme. This can be done by setting the capture setting usage mode to "Automatic" via "Capture -> Capture Settings -> Usage Mode".

**Figure 26: wxPropView - Setting up request count**

That's it. Now, starting a live acquisition will display live images in both displays and each display is using a different set of capture parameters. If a device supports parallel acquisition from multiple input channels, this will increase

- the used bandwidth and also
- the CPU load
as \texttt{wxPropView} now needs to display more images per second. Each display can be configured independently thus e.g. one display can be used scaled while the other displays 1:1 data. The analysis plots can be assigned to a specific display by left-clicking on the corresponding image display, the info plot will plot a graph for each capture setting in parallel.

![wxPropView](image)

\textbf{Figure 28: wxPropView - Running example}

When only one setting shall be used at a given time, this can be achieved by setting the capture setting usage mode back to "Manual" via "Capture \rightarrow Capture Settings \rightarrow Usage Mode". Then the setting that shall be used can be manually selected in the request control list:
This can even be changed during a running acquisition.

9.1.1.10 Bit-shifting an image

wxPropView (p.42) shows snapped or live images in the display area of the GUI. The area, however, shows the most significant bits (msb) of the image in the 8 bit display.

The following image shows how a mid-grey 12 bit pixel of an image is displayed with 8 bit. Additionally, two shifts are shown.

```
Mono12 (12 bit):

0 1 1 1 1 0 0 1 1 1 1 0 0

Display (8 bit):

0 1 1 1 1 0 0 1 1 1 1 0 0

Display (8 bit); Shift: 1:

0 1 1 1 1 0 0 1 1 1 1 0 0

Display (8 bit); Shift: 2:

0 1 1 1 1 0 0 1 1 1 1 0 0
```

Figure 30: Mid-grey 12 bit pixel image and 8 bit display with 2 example shifts
In this particular case, the pixel will be brighter (as the most significant bits are 1’s). Perhaps you already recognized it. Each shift means that each pixel value is multiplied or divided by 2 according to the direction.

Anyway, there is one restriction in the 8 bit display:

If the pixel value is greater than 255, the pixel value will be clipped to 255. To describe this from a programmer’s view; \( a \) represents the pixel value:

\[
 a = \begin{cases} 
 255 & \text{if } a > 255 \\
  a & \text{otherwise} 
\end{cases}
\]

With \texttt{wxPropView} (p. 42) you can shift the bits in the display using the left and right arrow keys. Furthermore you can turn on the monitor display to compare the images synchronously.

9.1.1.11 Changing the view of the property grid to assist writing code that shall locate driver features

With \texttt{wxPropView} (p. 42) it is possible to switch the views between "Standard View" (user-friendly) and "Developers View". While the first (default) view will display the device drivers feature tree in a way that might be suitable for most users of a GUI application it might present the features in a slightly different order as they actually are implemented in the device driver. The developers view switches the tree layout of the application to reflect the feature tree exactly like it is implemented and presented by the SDK. It can be helpful when writing code that shall locate a certain property in the feature tree of the driver using the C, C++, or .NET interface. The feature hierarchy displayed here can directly be used for searching for the features using the "ComponentLocator (C++/.NET)" objects or "DMR_FindList (C)" and "OBJ_GetHandleEx (C)" functions.

![Figure 31: Developers View](https://www.matrix-vision.com/tl_files/mv11/trainings/wxPropView/wxPropView_Bit-shifting/index.html)
9.1.1.12 Accessing log files

Since mvIMPACT Acquire 2.11.9

Using Windows, it is possible to access the log files generated by MATRIX VISION via the Help menu. Sending us the log files will speed up support cases.

![Image of wxPropView - Help menu]

Figure 32: wxPropView - Help menu

The options are to

- directly open the logs folder, to
- create a zip file with all the logs, and to
- open the systems default email client to send an email to support@matrix-vision.com.

9.1.2 How to configure a device

As described above, after the device has been initialized successfully in the "Grid" area of the GUI the available properties according to the chosen "interface layout" (e.g. GenICam) are displayed in a hierarchy tree.

wxPropView - Configuring a device:

The next chapter will show how to set the interface layout and which interface you should use according to your needs.
9.1.2.1 Different interface layouts

Devices belonging to this family only support the **Device Specific** interface layout which is the common interface layout supported by most MATRIX VISION devices.

**GenICam** compliant devices can be operated in different interface layouts. Have a look at a GenICam compliant device for additional information.

9.1.2.2 Configuring different trigger modes

To configure a device for a triggered acquisition, in **wxPropView** (p. 42) the property **“Image Setting -> Camera -> TriggerMode”** ("DeviceSpecific interface layout") or **“Setting -> Base -> Camera -> GenICam -> Acquisition Control -> Trigger Selector”** ("GenICam interface layout") is available.

9.1.2.3 Testing the digital inputs

**Note**

The following description will be significant if you are using the "DeviceSpecific interface layout". In GenICam layout, the "Digital I/O" section can be found in **"Setting -> Base -> Camera -> GenICam -> Digital I/O Control"**.

For performance reasons, device drivers will not automatically update their digital input properties if nobody is interested in the current state. Therefore, in order to check the current state of a certain digital input, it is necessary to manually refresh the state of the properties. To do this please right-click on the property you are interested in and select "Force Refresh" from the pop-up menu.

**GenICam interface layout only:**

Some devices might also offer an event notification if a certain digital input changed its state. This event can then be enabled

- via the "**EventSelector**" in **"Setting -> Base -> Camera -> GenICam -> Event Control"**.
- Afterwards, a callback can be registered by right-clicking on the property you are interested in again.
- Now, select "Attach Callback" from the pop-up menu and switch to the "**Output**" tab in the lower right section of wxPropView (Analysis tabs).

Whenever an event is send by the device that updates one of the properties a callback has been attached to, the output window will print a message with some information about the detected change.
9.1.2.4 Working with camera descriptions

Certain capture device (e.g. frame grabber) can process data from a wide range of imaging devices (e.g. cameras). However, in order to interpret the incoming data from an imaging device correctly, the capture device needs to be given a certain amount of information about the structure of the video signal.

The "mvIMPACT Acquire" interface addresses this necessity by the introduction of so called "camera descriptions". A "camera description" is a certain set of parameters that should enable the capture device to cope with the incoming image data to reconstruct a correct image from the imaging device in the memory of the host system. For instance, this information may contain information whether the image is transmitted as a whole or if it’s transmitted as individual blocks (e.g. when dealing with interlaced cameras) that need to be reconstructed in a certain way to form the complete image.

Each capture device will support different sets of parameters. For example some capture devices will only be able to capture image data from standard video source such as a PAL or NTSC compliant camera, while others might only be capable to acquire data from digital image source such as CameraLink® compliant cameras. To reflect these device specific capabilities "camera descriptions" have been grouped into different base classes. See e.g. **mvIMPACT::acquire::CameraDescriptionStandard** to find out how the basic structure of these objects look. Which basic "camera description" classes are supported by an individual device can be seen directly after the device has been initialised by looking in the "camera description" list. By default this list will contain one description for each supported basic family:
To select a certain camera description to be used to prepare the capture device for the expected data the property "Type" under "Image Settings -> Camera" can be modified. Here every available set of camera parameters will be listed:
Now, when a camera is connected, that differs in one or more parameters from the default offered by one of the available base classes and no special description for the imaging device in question is available a new matching description must be generated.

**Note**

It's also possible to modify one of the standard descriptions to adapt the parameter set to the used imaging device, but this method is not recommend as this would define something to be "standard", which in fact is not. Therefore it is not possible to store the standard descriptions permanently. It is, however, possible to modify and work with the changed parameters, but these changes will be lost once the device is closed.

The recommended way of adapting an imaging source to a capture device is to create a new description for a imaging device that does not completely fall into one of the offered standard descriptions. The first thing to decide when creating a new camera description is to which existing description offers the closest match for the new description. Once this has been decided a copy of this description can be created with an arbitrary name (that must be unique within the family the description is created from). Under **wxPropView** (p.42) this can be achieved by

- typing the new name in the parameter edit control right of the "Copy" method of the camera description to create the copy from.
- Afterwards, press **ENTER** to commit the new name and then the "Copy" method can be invoked
- by right clicking on the name of the function and
- selecting "Call" from the popup menu:
Afterwards, the newly created camera description will be added to the list of existing ones. Its parameters at this point will match the parent description (the one the "Copy" method was executed from) completely.

Figure 33: wxPropView - The newly created camera description

Now, the reason for creating a new camera description was that the parameters in the existing description didn’t exactly match the connected imaging device. Therefore, the next step would probably be to modify some of the parameters. Once this has been done (or before) the newly created description can be selected via the property "Type" under "Image Settings -> Camera".
Figure 34: wxPropView - Selecting the newly created camera description

**Note**

A new camera description will **NOT** be stored permanently by default. In order to make this description available the next time the capture device is initialised, the newly created description must be exported via a function call.

To store a camera description permanently the "Export" method of the new camera description must be invoked. The method does not require any parameters so it can be executed directly by right clicking on the name of the function and selecting "Execute" from the popup menu:
Figure 35: wxPropView - Exporting a created camera description

As a direct result the modified settings will become the new default values of this particular camera description. wxPropView (p. 42) indicates this by displaying all values belonging to the description in green now:
Figure 36: wxPropView - After exporting a new camera description

Note

Again please note, that this will NOT work for one of the standard camera descriptions. Whenever the user tries to export one of these, the error DMR_EXECUTION_PROHIBITED will be returned.

When exporting a camera description a file in XML format will be written to disc. Under Windows® camera descriptions will be stored under "%ALLUSERS%\Documents\MATRIX VISION\mvIMPACT acquire\CameraFiles" or "%MVIMPACT_ACQUIRE_DATA_DIR%\CameraFiles" which will point to the same folder. Under Linux® this directory will be "/etc/matrix-vision/mvimpact-acquire/camerafiles" while under other platforms these files will end up in the current working directory.

Now, when closing and re-opening a device only the default camera descriptions an the one selected before settings have been saved will appear in the list of camera descriptions. This is to save memory. However, all detected camera descriptions will be available via the property "Type" under "Image Settings -> Camera":

MATRIX VISION GmbH
Once a description is selected, that hasn't been in the list of camera descriptions before, it will be created and thus will become available for modifications again:
Again: For a different camera a new description should be generated, to operate complex cameras in different modes, a either a new description can be generated or an existing one can be modified.

After a camera has been modified the "Import" method can be used to fall back to the values stored in the camera description file:
Figure 39: wxPropView - Invoking the "Import" command of a camera description

This will restore the default settings for this description:
9.1.3 Command-line options

It is possible to start wxPropView via command line and controlling the starting behavior using parameters. The supported parameter are as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>width or w</td>
<td>Defines the startup width of wxPropView. Example: width=640</td>
</tr>
<tr>
<td>height or h</td>
<td>Defines the startup height of wxPropView. Example: height=460</td>
</tr>
<tr>
<td>xpos or x</td>
<td>Defines the startup x position of wxPropView.</td>
</tr>
<tr>
<td>ypos or y</td>
<td>Defines the startup y position of wxPropView.</td>
</tr>
<tr>
<td>splitterRatio</td>
<td>Defines the startup ratio of the position of the property grids splitter. Values between &gt; 0 and &lt; 1 are valid. Example: splitterRatio=0.5</td>
</tr>
<tr>
<td>propgridwidth or pgw</td>
<td>Defines the startup width of the property grid.</td>
</tr>
<tr>
<td>debuginfo or di</td>
<td>Will display debug information in the property grid.</td>
</tr>
<tr>
<td>dic</td>
<td>Will display invisible (currently shadowed) components in the property grid.</td>
</tr>
<tr>
<td>displayCountX or dcx</td>
<td>Defines the number of images displayed in horizontal direction.</td>
</tr>
<tr>
<td>displayCountY or dcy</td>
<td>Defines the number of images displayed in vertical direction.</td>
</tr>
</tbody>
</table>
9.2 mvDeviceConfigure

mvDeviceConfigure (p. 77) is an interactive GUI tool to configure MATRIX VISION devices. It shows all connected devices.

Various things can also be done without user interaction (e.g. updating the firmware of a device). To find out how to do this please start mvDeviceConfigure and have a look at the available command line options presented in the text window in the lower section (the text control) of the application.

9.2.1 How to set the device ID

The device ID is used to identify the devices with a self defined ID. The default ID on the device's EEPROM is "0". If the user hasn't assigned unique device IDs to his devices, the serial number can be used to selected a certain device instead. However, certain third-party drivers and interface libraries might rely on these IDs to be set up in a certain way and in most of the cases this means, that each device needs to have a unique ID assigned and stored in the devices non-volatile memory. So after installing the device driver and connecting the devices setting up these IDs might be a good idea.

To set the ID please start the mvDeviceConfigure (p. 77) tool. You will see the following window:

```
wXPropView.exe d=* fulltree=1 qsw=0
```

This will start the first available device, will hide the Quick Setup Wizard, and will display the complete property tree.
Whenever there is a device that shares its ID with at least one other device belonging to the same device family, `mvDeviceConfigure` will display a warning like in the following image, showing in this example two `mvBlueFOX` cameras with an ID conflict:

![Figure 39:mvDeviceConfigure - Conflicting device IDs](image)

9.2.1.1 Step 1: Device Selection

Select the device you want to set up from the list box.

9.2.1.2 Step 2: Open dialog to set the ID

With the device selected, select the menu item *Action* and click on *Set ID*.

**Note**

It is also possible to select the action with a right click on the device.

![Figure 40:mvDeviceConfigure - Select action](image)
9.2.1.3 Step 3: Assign the new ID

Enter the new ID and click OK.

![mvDeviceConfigure - New ID](image1)

Figure 41: mvDeviceConfigure - New ID

Now the overview shows you the list with all devices as well as the new ID. In case there has been an ID conflict before that has been resolved now mvDeviceConfigure (p. 77) will no longer highlight the conflict now:

![mvDeviceConfigure - Resolved ID conflict](image2)

Figure 42: mvDeviceConfigure - Resolved ID conflict

9.2.2 How to disable CPU sleep states a.k.a. C states (< Windows 8)

Modern PC’s, notebook’s, etc. try to save energy by using a smart power management. For this several hardware manufacturers specified the ACPI standard. The standard defines several power states. For example, if processor load is not needed the processor changes to a power saving (sleep) state automatically and vice versa. Every state change will stop the processor for microseconds. This time is enough to cause image error counts!
To disable the power management on the processor level (so-called "C states"), you can use **mvDeviceConfigure**:

**Note**

With Windows XP it is only possible to disable the C2 and C3 states. With Windows Vista / 7 / 8 all C states (1, 2, and 3) will be disabled.

**Warning**

Please be sure you know what you do! To turn off the processor's sleep states will lead to a higher power consumption of your system. Some processor vendors might state that turning off the sleep states will result in the processors warranty will expire.

**Note**

Modifying the sleep states using **mvDeviceConfigure** does only affects the current power scheme. For notebooks this will e.g. make a difference depending on whether the notebook is running on battery or not. E.g. if the sleep states have been disabled while running on battery and then the system is connected to an external power supply, the sleep states might be active again. Thus in order to permanently disable the sleep states, this needs to be done for all power schemes that will be used when operating devices.

1. Start **mvDeviceConfigure**.
2. Go to tab "Settings" and unselect "CPU Idle States Enabled".

---

**Available command line options:**

```
calib <id> or "id" to update the firmware of one or many devices(syntax: 'id = <serial>, id')
update_firmware to update the firmware of one or many devices
update_firmware to update the firmware of one or many devices
IPv4 netmask to specify an IPv4 address mask to use as a filter for the selected update operations
fw_path to specify a custom path for the firmware files
update_kernel driver to update the kernel driver of one or many devices
cut or 'c' to end the application automatically after all updates have been applied
```

The number of commands that can be passed to the application is not limited.

Usage examples:

- `mvDeviceConfigure calib=BF000666` (will update the firmware of a mvBlueFOX with the serial number BF000666)
- `mvDeviceConfigure update_firmware=BF000666` (will update the firmware of ALL mvBlueFOX devices in the current system)
- `mvDeviceConfigure setid=BF000666` (will assign the device ID "BF000666" to a mvBlueFOX with the serial number BF000666)
- `mvDeviceConfigure ubuntu update_firmware` (will update the firmware of every device in the system)
The sleep states can also be enabled or disabled from a script by calling `mvDeviceConfigure` like this:

```bash
mvDeviceConfigure.exe set_processor_idle_states=1 quit
```

or

```bash
mvDeviceConfigure.exe set_processor_idle_states=0 quit
```

The additional `quit` will result in the application to terminate after the new value has been applied.

**Note**

With Windows Vista or newer `mvDeviceConfigure` must be started from a command shell with administrator privileges in order to modify the processors sleep states.

### 9.2.3 Command-line options

It is possible to start `mvDeviceConfigure` via command line and controlling the starting behavior using parameters. The supported parameter are as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>setid</code> or <code>id</code></td>
<td>Updates the firmware of one or many devices(syntax: <code>id=&lt;serial&gt;,&lt;id&gt;</code> or <code>id=&lt;product&gt;,&lt;id&gt;</code>).</td>
</tr>
<tr>
<td><code>set_processor_idle_states</code> or <code>spis</code></td>
<td>Changes the C1, C2 and C3 states for ALL processors in the current system(syntax: <code>spis=1</code> or <code>spis=0</code>).</td>
</tr>
<tr>
<td><code>set_userset_persistence</code> or <code>sup</code></td>
<td>Sets the persistency of UserSet settings during firmware updates (syntax: <code>sup=1</code> or <code>sup=0</code>).</td>
</tr>
<tr>
<td><code>update_fw</code> or <code>ufw</code></td>
<td>Updates the firmware of one or many devices.</td>
</tr>
<tr>
<td><code>update_fw_file</code> or <code>ufwf</code></td>
<td>Updates the firmware of one or many devices. Pass a full path to a text file that contains a serial number or a product type per line.</td>
</tr>
<tr>
<td><code>custom_genicam_file</code> or <code>cgf</code></td>
<td>Specifies a custom GenICam file to be used to open devices for firmware updates. This can be useful when the actual XML on the device is damaged/invalid.</td>
</tr>
<tr>
<td><code>update_kd</code> or <code>ukd</code></td>
<td>Updates the kernel driver of one or many devices.</td>
</tr>
<tr>
<td><code>ipv4_mask</code></td>
<td>Specifies an IPv4 address mask to use as a filter for the selected update operations. Multiple masks can be passed here separated by semicolons.</td>
</tr>
<tr>
<td><code>fw_file</code></td>
<td>Specifies a custom name for the firmware file to use.</td>
</tr>
<tr>
<td><code>fw_path</code></td>
<td>Specifies a custom path for the firmware files.</td>
</tr>
<tr>
<td><code>log_file</code> or <code>lf</code></td>
<td>Specifies a log file storing the content of this text control upon application shutdown.</td>
</tr>
<tr>
<td><code>quit</code> or <code>q</code></td>
<td>Ends the application automatically after all updates have been applied.</td>
</tr>
<tr>
<td><code>force</code> or <code>f</code></td>
<td>Forces a firmware update in unattended mode, even if it isn't a newer version.</td>
</tr>
<tr>
<td><code>*</code></td>
<td>Can be used as a wildcard, devices will be searched by serial number AND by product. The application will first try to locate a device with a serial number matching the specified string and then (if no suitable device is found) a device with a matching product string.</td>
</tr>
</tbody>
</table>
The number of commands that can be passed to the application is not limited.

9.2.3.1 Sample (Windows)

mvDeviceConfigure ufw=BF000666

This will update the firmware of a mvBlueFOX with the serial number BF000666.

mvDeviceConfigure update_fw=BF* 

This will update the firmware of ALL mvBlueFOX devices in the current system.

mvDeviceConfigure update_fw=mvBlueFOX-2* 1f=output.txt quit

This will update the firmware of ALL mvBlueFOX-2 devices in the current system, then will store a log file of the
executed operations and afterwards will terminate the application.

mvDeviceConfigure setid=BF000666.5

This will assign the device ID '5' to a mvBlueFOX with the serial number BF000666.

mvDeviceConfigure ufw=* 

This will update the firmware of every device in the system.

mvDeviceConfigure ufw=BF000666 ufw=BF000667

This will update the firmware of 2 mvBlueFOX cameras.

mvDeviceConfigure ipv4_mask=169.254.*;192.168.100* update_fw=GX* 

This will update the firmware of all mvBlueCOUGAR-X devices with a valid IPv4 address that starts with '169.254.'
or '192.168.100.'.
The description for the mvIMPACT Acquire SDK for C developers is available as a separate file: mvIMPACT
Acquire_API_C_manual.chm which is

- either part of the installed package or

Here an online version of the documentation is available as well.
11 C++ developers

The description for the mvIMPACT Acquire SDK for C++ developers is available as a separate file: `mvIMPACT_Acquire_API_CPP_manual.chm` which is

- either part of the installed package or

Here an online version of the documentation is available as well.
12 .NET developers

The description for the mvIMPACT Acquire SDK for .NET developers is available as a separate file: mvIMPACT\_Acquire\_API\_NET\_manual.chm which is

- either part of the installed package or

Here an online version of the documentation is available as well.
13 Python developers

There is no separate manual available for the Python API right now. For documentation please refer to the C++ manual instead. Almost everything stated there will be valid for Python as well!

13.1 Introduction

As supporting all the various distributions and versions of Python out there with a binary interface is almost impossible the mvIMPACT acquire Python API needs to be compiled for a specific version of Python it shall be used with. Because of that what is shipped at the moment is source code that has been generated using SWIG which before using it requires a compilation operation.

This requires a compiler matching the version of Python that shall be used. E.g. to use mvIMPACT Acquire with Python 2.7 on Windows systems requires Visual Studio 2008. Additional information about the compiler that works for a particular Python version on a particular platform can be found online. See e.g.

- https://docs.python.org/2/extending/building.html
- https://wiki.python.org/moin/WindowsCompilers

Microsoft provides free-of-charge a 'Microsoft Visual C++ Compiler for Python 2.7' which can be used to compile 'mvIMPACT Acquire for python' for Python versions 2.7 to 3.2:
The use of this compiler packet has been tested by MATRIX VISION and it is is highly recommended.

When installing the mvIMPACT Acquire Python API on a target system all files needed for building the actual extension module can be found in /mvIMPACT_Python.

13.2 Building

During the compilation process Pythons distutils package will be used

13.2.1 Windows

On Windows systems running /mvIMPACT_Python/compileWrapperCode.bat will build and install the Python API in the site-packages sub-folder of your Python installation provided a matching compiler could be found.

The script will assume the Python interpreter can be found in the systems Path variable. If this is not the case you need to append the path to the directory containing Python.exe to this variable either permanently using the Systems environment variable dialog or temporary within the command shell you are calling the script from like this:

set Path=%Path%;C:\Python27
compileWrapperCode.bat

The command shell your are calling the compilation script from depending on the version of Windows you are working with and the folder you have installed the mvIMPACT Acquire package to might require elevated rights thus you might need to start the command shell with the Run as administrator option.
13.3 Using

13.2.2 Linux

On Linux systems running `/mvIMPACT_Python/setup.py` can be used to build and install the Python API in the `site-packages` sub-folder of your Python installation providing a matching compiler could be found. However one must be familiar with Python's `distutils` package.

Note

It is mandatory that the `python-dev` package is installed on the target Linux system, otherwise the binaries cannot be built!
The invoking user of the script must have the rights to install the generated binaries in the python directory of his system. If this is not the case, a recommended way to call the installation script is:

```
sudo -E python setup.py install
```

The building process may take literally a few minutes, so please be patient!

Attention

The SWIG generated wrapper code source file is very large! On some embedded systems or on 32-bit systems gcc might have trouble to digest this file as it is known to consume a lot of memory per source line in a given translation unit. When you encounter such a situation please get in touch with us!

13.3 Using

The actual API is almost the same as in C++ thus for now the C++ manual can be used as a reference and function description. There are just some minor differences between the C++ and the Python API which shall be explained here briefly:

- Stuff that has been declared deprecated at the time of publishing the Python API will not be available
- Simple getter functions may be wrapped as Python properties to have a more Python-like interface. So e.g. the function `Component::isValid()` will be a property in Python
- Code that in C++ resides in sub-namespaces like e.g. `mvIMPACT::acquire::GenICam` will all end up in `acquire` in Python (this is likely to change in future versions!)
- Some functions that use Python style built-in names like `mvIMPACT::acquire::Component::type()` will use a slightly different name in Python like `getType` in order to avoid confusion

Apart from that if someone is familiar with the C++ interface it shouldn't be too difficult to use the API. This is how an acquisition from a user selectable device can be done:

```python
from __future__ import print_function
import os
import platform
import string
import sys
# import all the stuff from mvIMPACT Acquire into the current scope
from mvIMPACT import acquire
# import all the mvIMPACT Acquire related helper function such as 'conditionalSetProperty' into the current scope
# If you want to use this module in your code feel free to do so but make sure the 'Common' folder resides in a sub-folder
from Common import *

# For systems with NO mvDisplay library support
import ctypes
import Image
```

MATRIX VISION GmbH
#import numpy
defMgr = acquire.DeviceManager()
for i in range(devMgr.deviceCount():
    pDev = devMgr.getDevice(i)
    print("[" + str(i) + "]: " + pDev.serial.read() + "(" + pDev.product.read() + ", " + pDev.family.read() + ", " + pDev.interfaceLayout.readS() + ", interface layout: " + pDev.acquisitionStartStopBehaviour.readS() + ", acquisition start/stop behaviour: " + pDev.isInUse() + ")
    if pDev.interfaceLayout.isValid:
        conditionalSetProperty(pDev.interfaceLayout, acquire.dilGenICam)
        print("", interface layout: " + pDev.interfaceLayout.readS(), end='')
    if pDev.acquisitionStartStopBehaviour.isValid:
        conditionalSetProperty(pDev.acquisitionStartStopBehaviour, acquire.assbUser)
        print("", acquisition start/stop behaviour: " + pDev.acquisitionStartStopBehaviour.readS(), end='')
    if pDev.isInUse():
        print("!!!ALREADY IN USE!!!", end='')
    print(")")
print("Please enter the number in front of the listed device followed by [ENTER] to open it: ", end='')
devNr = int(raw_input())
if (devNr < 0) or (devNr >= devMgr.deviceCount()):
    print("Invalid selection!")
sys.exit(-1)
pDev = devMgr.getDevice(devNr)
pDev.open()
print("Please enter the number of buffers to capture followed by [ENTER]: ", end='')
framesToCapture = int(raw_input())
if framesToCapture < 1:
    print("Invalid input! Please capture at least one image")
sys.exit(-1)

# The mvDisplay library is only available on Windows systems for now
isDisplayModuleAvailable = platform.system() == "Windows"
if isDisplayModuleAvailable:
    display = acquire.ImageDisplayWindow("A window created from Python")
else:
    print("The mvIMPACT Acquire display library is not available on this('" + platform.system() + ") system.

# For systems with NO mvDisplay library support
channelType = numpy.uint16 if channelBitDepth > 8 else numpy.uint8
fi = acquire.FunctionInterface(pDev)
statistics = acquire.Statistics(pDev)
while fi.imageRequestSingle() == acquire.DMR_NO_ERROR:
    print("Buffer queued")
pPreviousRequest = None
manuallyStartAcquisitionIfNeeded(pDev, fi)
for i in range(framesToCapture):
    requestNr = fi.imageRequestWaitFor(-1)
    if fi.isRequestNrValid(requestNr):
        pRequest = fi.getRequest(requestNr)
        if pRequest.isOK():
            if isDisplayModuleAvailable:
                display.GetImageDisplay().SetImage(pRequest)
                display.GetImageDisplay().Update()
            # For systems with NO mvDisplay library support
            #cbuf = (ctypes.c_char * imageSize).from_address(long(req.imageData.read()))
            #arr = numpy.fromstring(cbuf, dtype = channelType)
            #arr.shape = (height, width, channelCount)
            #if channelCount == 1:
            #    img = Image.fromarray(arr)
            #else:
            #    img = Image.fromarray(arr, 'RGB' if alpha else 'RGBA')
            if pPreviousRequest != None:
                pPreviousRequest.unlock()
pPreviousRequest = pRequest
fi.imageRequestSingle()
else:
    print("imageRequestWaitFor failed (" + str(requestNr) + ", " + ImpactAcquireException.getErrorCodeAsString(requestNr) + ")")
manuallyStopAcquisitionIfNeeded(pDev, fi)
raw_input("Press Enter to continue...")

Note

The above code uses the Python 3 style print. Because of the line
from __future__ import print_function

This will also work with Python versions starting with version 2.6. For smaller versions of Python the code needs to be changed!
14 DirectShow Interface

Note

DirectShow can only be used in combination with the Microsoft Windows operating system. Since Windows Vista, Movie Maker does not support capturing from a device registered for DirectShow anymore.

This is the documentation of the MATRIX VISION DirectShow_acquire interface. A MATRIX VISION specific property interface based on the IKsPropertySet has been added. All other features are related to standard DirectShow programming.

- Supported Interfaces (p. 90)
- Logging (p. 90)
- Registering and renaming devices for DirectShow usage (p. 91)

14.1 Supported Interfaces

14.1.1 IAMCameraControl

14.1.2 IAMDroppedFrames

14.1.3 IAMStreamConfig

14.1.4 IAMVideoProcAmp

14.1.5 IKsPropertySet

The DirectShow_acquire supports the IKsPropertySet Interface. For further information please refer to the Microsoft DirectX 9.0 Programmer's Reference.

Supported property set GUID's:

- AMPROPERTY_PIN_CATEGORY
- DIRECT_SHOW_ACQUIREPROPERTYSET

14.1.6 ISpecifyPropertyPages

14.2 Logging

The DirectShow_acquire logging procedure is equal to the logging of the MATRIX VISION products which uses mvIMPACT Acquire. The log output itself is based on XML.

If you want more information about the logging please have a look at the Logging chapter of the respective "mvIMPACT Acquire API" manual.
14.3 Registering and renaming devices for DirectShow usage

**Note**

Please be sure to register the MV device for DirectShow with the right version of mvDeviceConfigure (p. 77). I.e. if you have installed the 32 bit version of the VLC Media Player, Virtual Dub, etc., you have to register the MV device with the 32 bit version of mvDeviceConfigure (p. 77) (“C:\Program Files\MATRIX VISION\mvIM←PACT Acquire\bin”).

14.3.1 Registering devices

To register a device/devices for access under DirectShow please perform the following registration procedure:

1. Start mvDeviceConfigure.
   
   If no device has been registered the application will more or less (depending on the installed devices) look like this.

   ![mvDeviceConfigure - start window](image)

   **Figure 1: mvDeviceConfigure - start window**

2. To register every installed device for DirectShow access click on the menu item "DirectShow" → "Register all devices".
3. After a successful registration the column "registered for DirectShow" will display 'yes' for every device and the devices will be registered with a default DirectShow friendly name.
14.3 Registering and renaming devices for DirectShow usage

14.3.2 Renaming devices

If you want to modify the friendly name of a device under DirectShow, please perform the following procedure:

1. If mvDeviceConfigure is already not running, please start it.
2. Now, select the device you want to rename, click the right mouse button and select "Set DirectShow friendly name":

3. Then, a dialog will appear. Please enter the new name and confirm it with "OK".
4. Afterwards the column "DirectShow friendly name" will display the newly assigned friendly name.

![Figure 5: mvDeviceConfigure - enter new name](image)

![Figure 6: mvDeviceConfigure - renamed device](image)

**Note**

Please do not select the same friendly name for two different devices. In theory this is possible, however the mvDeviceConfigure GUI will not allow this to avoid confusion.

14.3.3 Make silent registration

To make a silent registration without dialogs, the Windows tool "regsvr32" via command line can be used. The following command line options are available an can be passed during the silent registration:

**EXAMPLES:**

Register ALL devices that are recognized by mvIMpact Acquire (this will only register devices which have drivers installed).

```
regsvr32 <path>\DirectShow_acquire.ax /s
```
### 15 Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/D reference</td>
<td>Upper threshold of video signal to be digitized. All values above this limit value are digitized to 255. Increasing the reference level results in contrast deterioration and vice versa.</td>
</tr>
<tr>
<td>ADC</td>
<td>Analog to digital converter (A/D converter)</td>
</tr>
<tr>
<td>Resolution</td>
<td>Number of pixels (horizontal x vertical)</td>
</tr>
<tr>
<td>Base address</td>
<td>Starting address from which the memory or register are inserted.</td>
</tr>
<tr>
<td>Image refresh rate</td>
<td>Number of transferred images per second. Normally specified in Hz (e.g. 70 Hz)</td>
</tr>
<tr>
<td>Bpp</td>
<td>Bits per pixel</td>
</tr>
<tr>
<td>Bus</td>
<td>A group line via which the various parts of the computer communicate with one another.</td>
</tr>
<tr>
<td>CCIR</td>
<td>Comité Consultatif International of the Radio Communications European video standard for 50 Hz gray scale.</td>
</tr>
<tr>
<td>Clamp signal</td>
<td>Clamp signal means, that a AC coupled video signal is clamped on the porch to get a signal transfer with less noise and independent from the d.c. voltage portion.</td>
</tr>
<tr>
<td>CPU</td>
<td>Central processing unit</td>
</tr>
<tr>
<td>DAC</td>
<td>Digital to analog converter (D/A converter)</td>
</tr>
<tr>
<td>Defaults</td>
<td>Standard system settings</td>
</tr>
<tr>
<td>DIO</td>
<td>Digital inputs and outputs</td>
</tr>
<tr>
<td>DIP switch</td>
<td>Dual inline package (housing design)</td>
</tr>
<tr>
<td>External trigger</td>
<td>External event used to initiate image capture.</td>
</tr>
<tr>
<td>False colors</td>
<td>Colors are assigned to gray scale via a look-up table. This allows even small gray scale differences can be displayed clearly.</td>
</tr>
<tr>
<td>Field</td>
<td>All odd lines of a field (odd field) or all even lines of a field (even field) of an interlaced video image.</td>
</tr>
<tr>
<td>Frame grabber</td>
<td>Here: PC plug-in card for digitization and storage of video images.</td>
</tr>
<tr>
<td>Horizontal sync</td>
<td>The portion of the analog signal which specifies the line end of the video signal.</td>
</tr>
<tr>
<td>Host</td>
<td>Here: the PC</td>
</tr>
<tr>
<td>Interfaced</td>
<td>Interfacing method; conforming to the television standard, this method involves acquiring two fields in succession (all odd lines, all even lines) and combining them to create a frame. The result is greatly reduced flicker during on-screen display.</td>
</tr>
<tr>
<td>Interrupt</td>
<td>Interrupt signal sent to the processor. The program currently running is interrupted and a predefined function is executed.</td>
</tr>
<tr>
<td>ISR</td>
<td>Interrupt service routine</td>
</tr>
<tr>
<td>IRQ</td>
<td>Interrupt request</td>
</tr>
<tr>
<td>Look-up table</td>
<td>Table of assignments. Here, new gray scale or colors are normally assigned to gray scale. Look-up tables can, however, also be used for any other math functions.</td>
</tr>
<tr>
<td>LSB</td>
<td>Least significant bit</td>
</tr>
<tr>
<td>LUT</td>
<td>Look-up table</td>
</tr>
<tr>
<td>Monochrome</td>
<td>A single-color (black and white) image</td>
</tr>
<tr>
<td>MSB</td>
<td>Most significant bit</td>
</tr>
<tr>
<td>Non-interlaced</td>
<td>Image acquisition and output line by line</td>
</tr>
<tr>
<td>NTSC</td>
<td>National Television Standard Code. US video standard for 60 Hz colors.</td>
</tr>
<tr>
<td>Overlay</td>
<td>Image memory for outputting text and graphics via the video monitor.</td>
</tr>
<tr>
<td>PAL</td>
<td>Phase alteration line; 50 Hz video standard for color.</td>
</tr>
<tr>
<td>Pixels</td>
<td>Picture element</td>
</tr>
<tr>
<td>PoCL</td>
<td>Power over CameraLink - The cameras are powered over CL cable and therefore need no additional power supply. The mvHYPERIONs which supports PoCL, are <strong>Switchable PoCL frame grabbers</strong> as described in the CameraLink™ specification. This means that both camera and cable have to support PoCL otherwise Pin 1 and Pin 26 of the CL connectors act like internal shields.</td>
</tr>
<tr>
<td>Pseudo colors</td>
<td>Display of gray scale images in false colors. A corresponding color is assigned to a specific gray scale value.</td>
</tr>
<tr>
<td><strong>Square pixels</strong></td>
<td>Square-shaped pixels (height-width ratio 1:1)</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td><strong>RS170</strong></td>
<td>US video standard for 60 Hz b/w colors</td>
</tr>
<tr>
<td><strong>TFT display</strong></td>
<td>Thin film transistor display</td>
</tr>
<tr>
<td><strong>True color</strong></td>
<td>24-bit true color; 16.7 million colors</td>
</tr>
<tr>
<td><strong>Vertical sync</strong></td>
<td>Synchronization pulse in video signal for field end recognition.</td>
</tr>
<tr>
<td><strong>Zero signal</strong></td>
<td>The zero signal was needed with the old frame grabbers, to calibrate the analog/digital converter (ADC) (signal and parameter aren't important anymore).</td>
</tr>
</tbody>
</table>