Level 1 for Inspecta-5

Software-Manual

Level 1 -Software Manual
For Inspecta-5 Frame Grabbers

Rev. 0.4.0.5

-- preliminary --

Mikrotron GmbH
Landshuter Str. 20-22
D-85716 Unterschleissheim
Germany

Tel.: +49 89 726342 00
Fax: +49 89 726342 99
info@mikrotron.de
www.mikrotron.de

INDEX

1 General

This manual describes the Level1 Software Interface to the Inspecta-5 High Performance Camera Link® Frame Grabber.

Requirements:

- Pentium III or better
- Windows 2000 or Windows XP™ Microsoft
- 256 Mbytes of Main Memory or more

Features of the Inspecta-5:

- Frame grabber for digital matrix cameras with „Full“ Camera Link® interface, „Medium“ Camera Link® interface and „Base“ Camera Link® interface.
- Compatible to Camera Link® Specification 1.0 and 1.1.
- Support for line scan cameras.
- Two 26-pin. Connectors with full support of the “Full” Camera Link® specification for video data, camera control and –configuration with build in serial interface.
- Video data rate of up to 660 Mbytes/sec.
- PCIX bus interface for 32 Bit data width and 33 MHz clock frequency.
- One Gigabyte Onboard Memory for fast video streams.
- Four opt coupled input- output ports for external trigger and encoder signals.
- PCI – X bus interface for 64 Bit data width and 66 MHz clock frequency.
- 528 Mbytes/sec. maximum data rate on the PCI–X Bus.
- SDK for Windows 2000/XP

The next revision of the Inspecta-5 FPGA program gives you:

- Parallel grab to internal memory and to PC – memory.
- SDK for HALCON Comprehensive Machine Vision Software.

The Inspecta-5 hardware is controlled by a programmable FPGA. The features of the next firmware revision is given to you for free and can be downloaded at our homepage as soon as available.

1.1 About Level1

Common to all of our frame grabbers products, is an Application Programming Interface (API) called Level1.

With the Level1 API you will need only a few function calls to initialize the Inspecta, choose a camera and get an image.
Level1-functions are developed to speed up programming the Inspecta and to make it much easier.

In this manual you get a description of the Level1 API and it's special implementations for the new frame grabber '**Inspecta-5**'. Inspecta-5 is a Full Camera Link® Frame Grabber with On Board Memory. The Inspecta-5 is our latest model of a production line of several different frame grabbers.

The current Level1 API for Inspecta-5, differs in some ways from the Level1 API existing for the frame grabber models Inspecta-2 to Inspecta-4.

On the next pages, we describe the Level1 API for the Inspecta-5.

To get the latest information on driver and DLL development concerning the Inspecta-5, please visit out homepage

http://www.mikrotron.de/

1.2 Revision history

Information presented in this publication has been carefully checked for reliability, however, no responsibility is assumed for inaccuracies. The information contained in this document is subject to change without notice.

1.3 Trademarks

All brand and product names that appear in this manual may be trademarks or registered trademarks of the corresponding companies.
Intel, the Intel Inside logo, Pentium® are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries, and are used under license.

2 Installation

2.1 Setup-disk / CD

The Inspecta frame-grabber is delivered with a CD for installing the Inspecta Software.
The CD consists of all device drivers, libraries and other files you need to program the Inspecta-5. There is a program on it, you can use to test the frame grabber. The source code for the program is also included on the CD.

The newest version of the setup CD can be found at

http://www.mikrotron.de.

2.2 Installation of Inspecta-5 for Windows 2000/XP

☝	**Please install all device drivers and driver updats of your motherboard (have a look at the driver CD delivered with your motherboard) before installing the Inspecta-5 hard- or software.** **A subsequently installation of the drivers could lead to an instable or invalid computer system and could require a new installation of your operation system!**

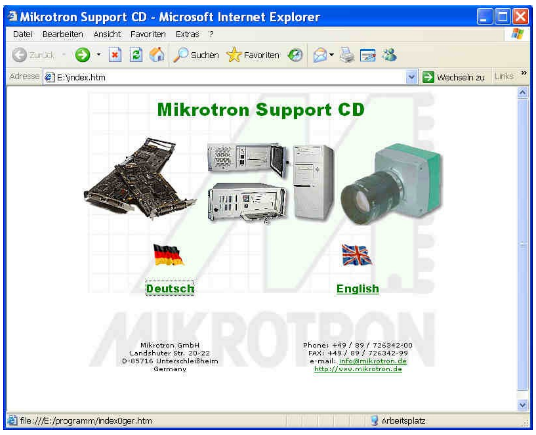
Before starting installation…

- Be sure the frame grabber is **not** installed in your computer.
- Check if there is an option called ‘plug&play operation system installed’ in your BIOS. If this option exists, set it to ‘yes’.
- Install the latest service packs of your operation system.

2.2.1 Setup the Inspecta-5 basic components

First you have to install some pasic components of the Inspecta-5:

- Start Windows 200/XP. You should not have installed the Inspecta hardware for now!
- Insert the Inspecta Setup CD, into your CD-ROM drive and close it. After a short time Windows should show you the start screen shown below. If the setup does not start by itself, you have possibly disabled the Auto Start function of your CD drive. Enable this option and open and close the drawer of your CD driver again. You can also start the setup directly by clicking on the file **INDEX.HTM** located on the setup CD.
-

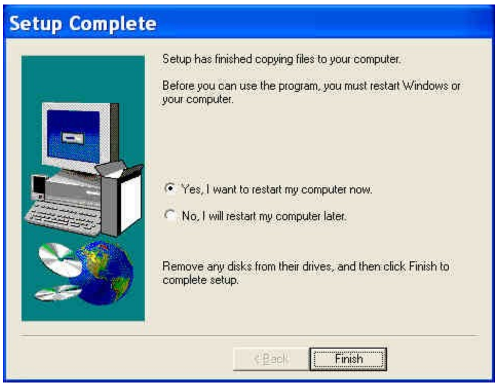

Mikrotron Support CD - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück
Suchen
Favoriten
Adresse
E:\index.htm
Wechseln zu
Links
Deutsch
English
file:///E:/programm/index0ger.htm
Arbeitsplatz

Mikrotron Support CD

Mikrotron GmbH
Landshuter Str. 20-22
D-85716 Unterschleißheim
Germany

Phone: +49 / 89 / 726342-00
FAX: +49 / 89 / 726342-99
e-mail: info@mikrotron.de
http://www.mikrotron.de

- To start the English installation, click the text‘English’ on the mask. On the next screen select ‘**Win 2000/XP**’ from the left column headed by the text ‘**Drivers**’. This starts the Inspecta-5 setup.
- If Windows shows a warning, please ignore it.
- Follow the instructions of the Setup program.
- After you have finished the Setup program, shut down your computer and switch it off.
- Please do not remove the Inspecta-5 setup CD from the CD-ROM drive!



Setup has finished copying files to your computer.

Before you can use the program, you must restart Windows or your computer.

● Yes, I want to restart my computer now.

○ No, I will restart my computer later.

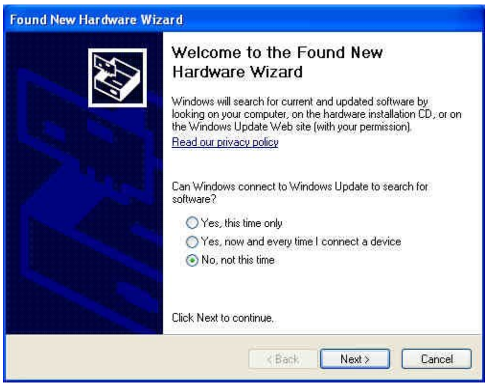
Remove any disks from their drives, and then click Finish to complete setup.

2.2.2 Installing the Inspecta-5 hardware

- Be sure that your computer is switched off (better: disconnect it from the power supply system).
- Now install the Inspecta-5 hardware into your computer. Inserting the frame grabber card into a free PCI slot of your motherboard does this.
- Restart your computer.

2.2.3 Installing the device driver

- Wait until Windows recognizes new hardware (Video controller for multimedia). Windows needs some seconds to find the new hardware, so be patient.



Welcome to the Found New Hardware Wizard

Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission).

Read our privacy policy

Can Windows connect to Windows Update to search for software?

○ Yes, this time only

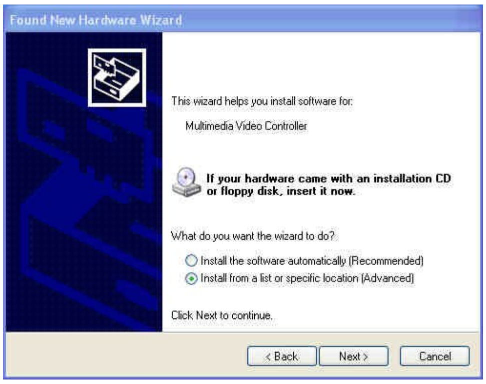
○ Yes, now and every time I connect a device

● No, not this time

Click Next to continue.

- If you have installed the Windows Service Pack 2, the message shown above will eventually be displayed. Select ‘No, not this time’ and continue pressing button ‘Next’.

- When the ‘Found New Hardware Wizard’ appears, select ‘**install from a list or specific location**’ and click ‘**Next‘.**
  On the next dialog select ‘**Include this location in the search**’ and browse to the drive you put the Inspecta-5 CD in. Click ‘**Next**’ to install the driver.



This wizard helps you install software for:

Multimedia Video Controller

If your hardware came with an installation CD or floppy disk, insert it now.

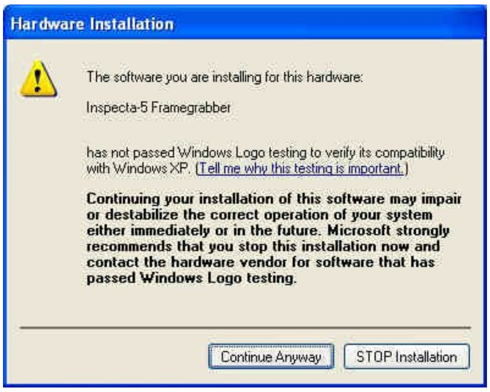
What do you want the wizard to do?

○ Install the software automatically (Recommended)

● Install from a list or specific location (Advanced)

Click Next to continue.

- A warning message may appear, stating that the hardware has not passed the Windows Logo Test. Select ‘**Continue Anyway**’ to allow the drivers to install.



The software you are installing for this hardware:

Inspecta-5 Framegrabber

has not passed Windows Logo testing to verify its compatibility with Windows XP. (Tell me why this testing is important.)

Continuing your installation of this software may impair or destabilize the correct operation of your system either immediately or in the future. Microsoft strongly recommends that you stop this installation now and contact the hardware vendor for software that has passed Windows Logo testing.

- Now Windows installs the Inspecta driver. After a few seconds you should get a message from Windows, pointing out that the new driver has been installed.

The hardware of the Inspecta-5 consists of 1 main and 7 subcomponents. After installing the driver of the main component, Windows start to request drivers for the seven subcomponents. Please repeat the instructions starting from paragraph 1.1.3 another 7 times.

- After installing the last driver, the Inspecta-5 installation has finished. Now you can start to use the frame grabber.

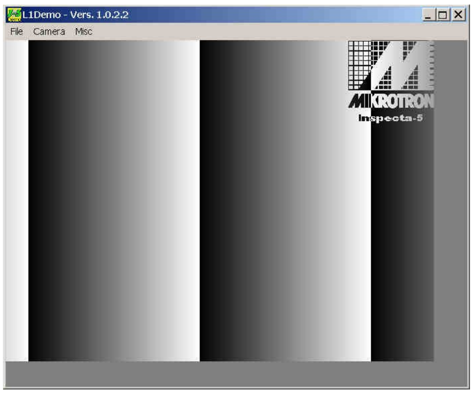
2.2.4 A first test

Now that you have installed the frame grabber, the device driver and all software components, you can do a first test to check if the Inspecta-5 works.

This can be done with the sample program ‘L1DEMO.EXE’ which can be started from `‘Start->Programs->Inspecta-5->L1Demo’`.

The source code of L1DEMO is also available at the installation directory of the Inspecta-5 software.

After L1DEMO has started, select ‘**Camera->Live picture**’ from the main menu. Now you should see the test pattern below. The test pattern is a grey scale image, which pixels values are reaching from 0 to 255. The test pattern is moving steadily from right to left.



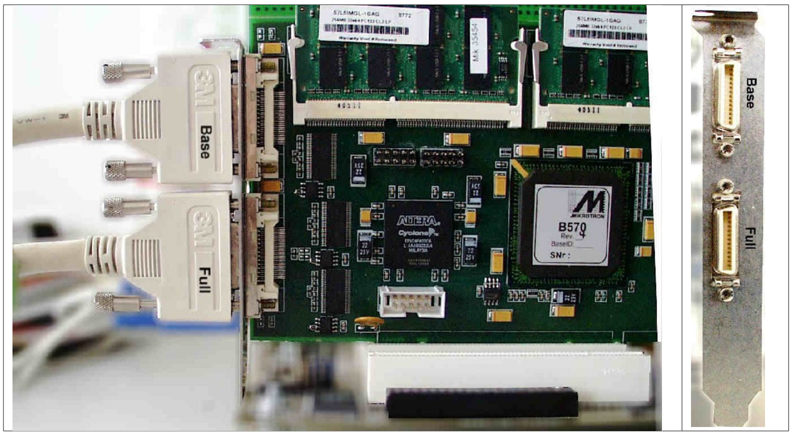
If the program show error messages or the test pattern is not displayed or not correctly shown, please do this:

- Check if the setup program finished without errors. If you are not sure, please reinstall it.
- There are all device drivers for the inspecta-5 installed? Use the Windows Device Manager to check it.
- There are any problems with the computer or frame grabber hardware? Check if the frame grabber is plugged well into the PCI slot
- Did you set the BIOS option ‘Pluy&Play OS’ to ‘YES’?
- Restart your computer and do the test again.

If the Inspecta-5 does not work at all, please send an Email to our support.

If the test runs without any problems, your Inspecta-5 is ready to connect a camera to it.

There are two connectors, each with 26 pins, on the slot bracket of the frame grabber. The upper one of the connectors is used for Base Camera Link® cameras. If you want to use Medium or Full Camera Link® cameras you have to use both connectors together.



Connect your Camera Link® cable to connect the connector on your camera to the appropriate connector on the frame grabber. Please take attention to connect the Base connector on your camera to the Base connector on the frame grabber. Check the same for the Medium/Full connector.

Aim the lenses of your camera to a light motive.

Stop displaying the test pattern using 'Stop' from the Camera menu (**Camera->Stop**) and select 'Load profile' from the same menu (**Camera->Load profile**). In the shown dialog select a camera profile from the list box '**Profiles**', which fits to your camera. If there is no predefined profile for your camera, please contact our support at info@mikrotron.de.

After selecting a profile and pressing the button '**OK**', the frame grabber gets initialized with the values from the camera profile. Now select '**Camera->Live picture**' again to get a live.

If you can not get a live picture or the shown pictures seems to be corrupted, please check this:

- Is the power supply of your camera connected to it?
- Is the aperture of the lenses opened?
- Are all cables connected (maybe you have to exchange the lines for channel 0 and channel 1 on the frame grabber)?
- Check if all device drivers of the Inspecta-5 are installed and running (use 'Device Manager')
- Check if the hardware fits well in the PCI slot of your computer
- Try to change the PCI slot for the frame grabber card.

If you continue to have problems, please contact our service by Email at

info@mikrotron.de.

2.2.5 The Inspecta-5 frame buffers

The Inspecta-5 uses two independence buffers to store camera frames:

- ‘On Board Memory’ placed on the frame grabber board itself
- A part of the main memory of the computer, which is reserved exclusively for the frame grabber (image memory).

Frames captured from the camera are saved in the on board memory of the Inspecta-5. Because a user application cannot directly access data in this buffer, the Inspecta-5 uses a second buffer, hosted in the main memory of the computer. The memory of the buffer is mapped to the address space of your application, so you can directly access image data in this area. The API of the Inspecta exports some function to copy image data from the on board memory to the main memory.

The dimension of the main memory buffer is defined in the Windows Registry. The default size of the buffer is 8 Mbytes.

The Inspecta buffer in main memory can be defined in two ways:

- **Definition of image-memory, compatibility mode**
  The definition of the image buffer in main memory, takes place at the first installation (or the first after a de-installation) in a mode called "compatibility mode" (don’t’ get confused with the ‘compatibility mode’ of the frame grabber to capture images, see below). An image-memory of 8 MB from the „NonPagedMemoryPool“ of Windows is reserved for each Inspecta.
  This procedure needs no manual operation for changes in system-files.

  Because of the limited size of the ‘NonPagedMemoryPool’ the memory you can get from Windows for storing images is restricted to a few megabytes. If you need a larger buffer in main memory you should use the ‘MAXMEM’ method described below.

- **Definition of image-memory, "maxmem mode"**
  This mode uses the MAXMEM switch of the Windows BOOT.INI file.
  The MAXMEM switch reduces the amount of memory Windows can use, up to the value assigned to this parameter. Physical memory above this value is not visible to Windows, so the Inspecta frame grabber can use it to store camera frames.
  You have to restart the computer after doing any changes to BOOT.INI.

  Example:
  A computer has 128 Mbytes of main memory. We want to use 32 Mbytes of main memory as a frame buffer for the Inspecta-5. So we have to restrict the Windows usable memory to 96 Mbytes. Therefore we set the MAXMEM switch in the BOOT.INI file to 96. This has the effect, that the memory above 96 Mbytes is available for the Inspecta frame buffer without conflicts to Windows.

  Note: BOOT.INI is a Windows system file. To make any changes to it, you must have administration rights! The file is also hidden. You can make it visible by setting the attributes of the file by typing this command in a Windows Command window:

```
cd \
attrib boot.ini -r -h -s
```

Now it should be possible to edit the file. Add the MAXMEM switch to the end of the line containing the boot information for the Windows operation system.

☝	**Before doing any changes to BOOT.INI, we urgent recommend to save a copy of the original file on a save place!**

Example: Excerpt from a BOOT.INI file from a Windows XP system

```
[boot loader]

timeout = 30

default=multi (0) disk (0) rdisk (0) partition (1) \WINDOWS

[operating system]
multi (0) disk (0) rdisk (0) partition (1) \WINDOWS
="Microsoft Windows XP Professional" /fastdetect /noguiboot  /MAXMEM=96
```

2.2.6 Registry entries

The Inspecta uses the following entry in the Services branch:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\MpfgI5Xp
```

There are three entries you can modify to define the frame grabber buffer in main memory:

Key	Sample value	Description
`BlockSize`	`0x00800000`	length of image memory in Bytes (default: 8MB)
`MemStartPage`	`0x02000000`	compatibility mode: not used maxmem mode: Start address of the Inspecta frame buffer in main memory (default: 0)
`MemoryAllocationMode`	`1`	Selects the mode for frame buffer allocation: 0 = compatibility mode (default) 1 = maxmem mode

Do not change any other value!

2.3 Multiple Inspectas-5 in one PC

The current version of the device driver handles just one Inspecta-5 per computer. This will change soon, so you can handle up to 4 frame grabbers per computer in the near future.

To select an Inspecta-5 Frame Grabber, all Level1 functions have a ‘DeviceNumber’ (0…3) as a parameter in its function body.

Actual only DeviceNumber 0 is valid.

3 Functions

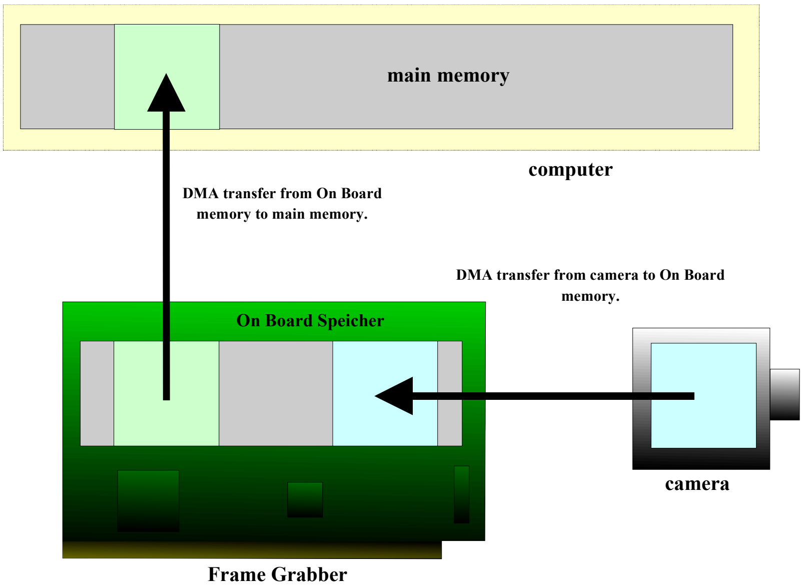
3.1 Overview

As mentioned above, the Inspecta-5 uses two kinds of memory to store images from a camera:

- On Board Memory of the Frame Grabber. This memory stores images directly captured from a connected camera. User applications cannot access this memory directly.
- An area in the main memory of the computer (frame buffer). This memory is allocated at boot time and cannot be changed in size and position at runtime. If you want to change the size of the buffer, you have to modify the Registry and reboot Windows. Frames copied from the On Board Memory of the Frame Grabber are stored here. This memory can be accessed for reading or/and writing by user applications.

Therefore, to get access to captured frames, you need two steps:

1. Save the image captured from camera in the On Board Memory of the Frame Grabber.
2. Copy the complete captured frame from On Board Memory to the Inspecta buffer in main memory.


main memory
computer
DMA transfer from On Board memory to main memory.
DMA transfer from camera to On Board memory.
On Board Speicher
camera
Frame Grabber

Because capturing image data from the camera and copy a frame to main memory is done by two independent DMA channels, it is possible to run these two streams concurrently. But take care not to read from the same memory where the other stream is writing to at the same time.
This concept opens you a wide range of possible solutions to realize your applications.

3.2 Inspecta-5 operation modes

The Inspecta-5 can be driven in two modes:

- **The Compatibility Mode (ExtFlag=0)**
  This mode gives certain compatibility to the Level1 API of Inspecta-2 to Inspecta-4.
  If you request a new image, it will be captured to the On Board Memory of the Frame Grabber

and then copied to user memory automatically (see chapter 3.2.5, function ‘mvfg_grab’). This mode captures images and sends them to main memory by just calling **one** function.

- **The Extended Mode (ExtFlag=1)**
  This mode separates the capturing of images from the camera to the On Board Memory of the Frame Grabber and the transport of image data from On Board Memory to the main memory. This gives the software developer much more flexibility on developing his projects.

The Extended Mode is the preferred mode for working with the Inpsecta-5 Frame Grabber.

To switch from one mode to another use function ‘mvfg_setparam’ (see chapter 3.3.3) by setting/resetting flag ‘MVFGPAR_EXT_FLAG’.

☝ **Please Note:**
You may not mix Compatibility and Extended mode!

3.3 Camera communication

The Camera Link® Specification assumes a serial interface to be integrated on the Frame Grabber board. This interface is used to send commands to a connected camera. These commands are different among different manufacturers.

The Inspecta-5 is compatible to the Camera Link® Specification 1.0 and 1.1.

There are some functions implemented in a separate DLL to send or receive bytes by the serial interface of the Frame Grabber. The name of the DLL is 'CLSERMI5.DLL' and will be installed on your computer together with the device driver of the Inspecta-5 Frame Grabber.

The Camera Link® Specification 1.0 defines these functions for communication:

Function	Description
clSerialInit	Initializes the serial interface
clSerialRead	Reads bytes from the camera
clSerialWrite	Writes bytes to the camera
clSerialClose	Closes the connection

The Camera Link® Specification 1.1 defines these functions for communication:

Function	Description
clSerialInit	Initializes the serial interface
clSerialRead	Reads bytes from the camera
clSerialWrite	Writes bytes to the camera
clSerialClose	Closes the connection
clFlushPort	Flushes the receive buffer
clGetErrorText	Get error text if function fails
clGetNumBytesAvail	Returns number of unread bytes in receive buffer
clGetSupportedBaudRates	Returns the supported baud rates of the frame grabber.
clSetBaudRate	Sets the baud rate of the serial line
clGetNumSerialPorts	Returns the number of serial ports on the frame grabber card
clGetManufacturerInfo	Returns manufacturer info
clGetSerialPortIdentifier	Returns the ID of the selected serial line

See the Camera Link® Specification for a more detailed description.

Before using the CLSERMI5.DLL, make sure to run an Inpsecta-5 application, which uses the function 'mvfg_open()' (e.g. DEMO1.EXE)!

If you want to communicate over the serial line of the frame grabber by your application, you have to bind to the '**ClserMi5.lib**' library of the Inspecta-5. The prototyping for the functions above can be find in the header file '**ClserMi5.h**'.

3.4 Level-1 Compatibility Mode

3.4.1 mvfg_open

Synopsis:

```
LONG WINAPI mvfg_open( char *   pcCameraProfile,
                       LONG     DeviceNumber )
```

Description:

Opens, initializes and configures the Inspecta-5 Frame Grabber.

The Inspecta is configured by a set of parameters. All parameters are summarized in a parameter block, also called camera profile or camera section.

The function ‘mvfg_open()’ expects the data of a profile written to a text file. We call this file ‘camera file’ (see camera profile).
There can be multiple profiles in a camera file, each headed by a unique profile name.

A profile may include the name of a file that includes control sequences for the connected camera. The content of the file is send to the camera via the Inspecta-5 serial interface.

The file L1DEMO.CAM, which is a part of the L1DEMO, shows some examples of camera profiles. The configuration refers to Mikrotron cameras of type MC 131x.

Sample:
See sample **Opening and closing the driver**

The current version of the Inspecta 5 device driver supports only one frame grabber per computer. So, the DeviceNumber has to be set to 0.

Parameters:

*char * pcCameraProfile*
pcCameraProfile points to a c-string, which contains the filename of the camera file and the name of a specific profile name.
(e.g. "Inspecta-4A.cam;TestMode").
Filename and profile name are separated by ";".
A profile in the camera file is marked by "[camera name]".
All parameters needed to initialize the frame grabber have to be listed in the profile. The parameters are read and set automatically by mvfg_open.

LONG DeviceNumber
Grabber-number (0 to 3)

Return value (LONG):

MVFG_OK
Initialization succeed

EMVFG_NO_VXD
Error: Grabber isn't installed

EMVFG_CAMFILE_NOTFOUND
Error: configuration-file doesn't exist.

EMVFG_CAMSECTION_NOTFOUND
Error: Section within the configuration-file doesn't exist.

EMVFG_CAMSTRG_FILE_NOTFOUND
Error: Camera-string-file indicated but not found.

Other
general error

3.4.2 mvfg_setparam

Synopsis:

```
LONG WINAPI mvfg_setparam( char * pcParamName,
                           char * pcParamValue,
                           LONG   DeviceNumber )
```

Description:

Specific parameters of the Inspecta can be set by this function during runtime. The values described below can also be defined statically by an entry in a camera profile.

The effect of a parameter depends of the selected Inspecta-5 mode. This section describes the use of the parameters for the Compatibility Mode. For Extended Mode see chapter 3.3, ‘Level-1 Extended Mode’.

Sample:

See sample **Setting and reading parameters**

Parameters:

*char * pcParamName*

Name of the parameter that is to be changed.
(Constants to use for pcParamName are listed in the table at the next page.)

*char * pcParamValue*

Value to set for this parameter
(you can find allowed values at the table at the next page)

LONG DeviceNumber

grabber-number

Return value (LONG):

MVFG_OK

parameter is set

EMVFG_NO_VXD

Error: grabber isn't installed

EMVFG_CAMPARAM_UNKNOWN

Error: parameter is unknown

EMVFG_CAMPARAM_BADVALUE

Error: value is not possible

EMVFG_NOT_OPEN

Error: driver hasn't been opened with mvfg_open or it was closed

<table>
<tr><th>pcParamName</th><th colspan="2">pcParamValue</th></tr>
<tr><td>MVFGPAR_LINELEN</td><td colspan="2">Length of an image-line in bytes (e.g. "640").

On the one hand, the length of an image line has to be a multiple of 32. On the other hand, the length of a line is always a multiple of the number of tabs the selected pixel router mode uses (see Pixelrouter).
<b>So the lengths of a line has to be always the least common multiple of the used tabs and 32!</b>

<b>Example:</b>

2 tabs mode -> least common multiple of 2 and 32 is 32 -> the length of a line has to be a multiple of 32.

10 tabs mode -> least common multiple of 10 and 32 is 160 -> the length of a line has to be a multiple of 160.</td></tr>
<tr><td>MVFGPAR_NUMLIN</td><td colspan="2">Number of image-lines for record (e.g. "602")</td></tr>
<tr><td rowspan="3">MVFGPAR_REQFRAME</td><td rowspan="3">Number / position of the frames to copy. The frames will be read from frame grabber on board memory and written to the target buffer in the main memory of the computer.</td><td>"0" ... "n" =
write one frame
at frame position “n” in the target buffer in main memory.</td></tr>
<tr><td>„-2“ ... „-n“ =
write (n – 1) frames.
The frames will be written at offset 0 in the target buffer.</td></tr>
<tr><td>"-1" = write as many frames as fit into the buffer in main memory.
The frames will be written at offset 0 in the target buffer.</td></tr>
<tr><td>MVFGPAR_TIMEOUT</td><td colspan="2">Timeout in ms</td></tr>
<tr><td>MVFGPAR_PHOTO,</td><td colspan="2">Defines the exposure Time for frame capturing with variable shutter. This parameter is only valid for cameras who supports this feature. The time base for the photo parameter depends on the current value of parameter MVFGPAR_PRESCALER_ACD. E.g. setting it to 8, means a time base of about 1 micro second.</td></tr>
<tr><td>MVFGPAR_PHOTOFLAG</td><td colspan="2">The photo flag defines whether recording is done asynchronously or in free running mode. Asynchronously means that the camera is triggered by an external event to make a photo. Free running means, the camera takes continuously photos without any external interaction.
If the photo flag is set, a photo can be triggered by the frame grabber itself (software trigger) or an external signal on input port 0 of the Inspecta-5 (hardware trigger). This depends of the setting of parameter <b>MVFGPAR_EXTPHOTOFLAG</b> (see below). If a trigger occurs, a signal is sent to the camera via the camera control bit CC1 of the Camera Link Interface. The length of the signal is defined by the value of <b>MVFGPAR_PHOTO</b> (see above). If the camera supports variable shutter, this signal can be used as exposure time for the camera.

<table>
<tr><th>Parameter</th><th>Bedeutung</th></tr>
<tr><td>MVFGVAL_NO</td><td>Don’t use asynchronous mode.</td></tr>
<tr><td>MVFGVAL_YES</td><td>Activates asynchronous mode. Call function mvfg_grab to initiate a photo and to capture the image.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>pcParamName</th><th>pcParamValue</th></tr>
<tr><td>MVFGPAR_EXTPHOTOFLAG</td><td>If photo flag is set, (see <b>MVFGPAR_PHOTOFLAG</b>, above), this flag defines if a photo is initiated by the frame grabber itself (software trigger) or by an external signal on the digital input port 0 (hardware trigger) of the Inspecta-5. Recording is started if the signal on port 0 gets active high.
<table>
<tr><th>Parameter</th><th>Bedeutung</th></tr>
<tr><td>MVFGVAL_NO</td><td>Software trigger is active. Call function mvfg_grab to start recording by the camera and to grab the image.</td></tr>
<tr><td>MVFGVAL_YES</td><td>Hardware trigger is active. Recording is started by an external signal on input bit 0 of the frame grabber. The camera itself is triggered by an signal on camera control bit 1 (CC1) of the Camera Link interface. Start grabbing by calling function mvfg_grab before the external signal gets active.</td></tr>
</table></td></tr>
<tr><td>MVFGPAR_TRIGGERMODE</td><td>This parameter is only valid for line scan cameras. It defines the way, camera lines are sampled by the frame grabber. If you want to use it, MVFGPAR_EXTPHOTOFLAG has to be set to hardware trigger active.
<table>
<tr><th>Parameter</th><th>Bedeutung</th></tr>
<tr><td>MVFGVAL_TRIGGERMODE _EDGE</td><td>An <b>pulse</b> on input port 0 of the frame grabber starts the camera to take a photo. This is done by the Inspecta-5, setting the camera control bit 1 (CC1) to level high. Then the frame grabber starts sampling as much lines coming from the camera, as defined by the parameter MVFGPAR_NUMLIN.</td></tr>
<tr><td>MVFGVAL_TRIGGERMODE _CONTINUOUS</td><td>An <b>signal</b> on input port 0 of the frame grabber starts the camera to take a photo. As long as the signal stays active (level high), lines are sampled from the camera. If the signal goes inactive, recording stops. So the vertical size of the grabbed frame depends of the length of the signal (variable frame size). The size of the buffer used to store the lines is defined by parameter MVFGPAR_NUMLIN. This buffer is treated as a ring buffer., i.e. if more than MVFGPAR_NUMLIN lines are sampled, the frame grabber starts again to write them from the beginning of the buffer. The number of lines sampled, can by read by using parameter MVFGPAR_CAPFRAME.</td></tr>
</table></td></tr>
</table>

3.4.3 mvfg_getparam

Synopsis:

```
LONG WINAPI mvfg_getparam( char * pcParamName,
                           void * pValueBuffer,
                           LONG   DeviceNumber )
```

Description:

The function returns the current or value of the Inspecta-5 parameters listet below.

The meaning of the parameters is the same as on function ‘mvfg_setparam()’ respectively is described below.

Sample:

See sample **Setting and reading parameters**

Parameters:

*char * pcParamName*

Name of the parameter to be read.
(Constants to use for pcParamName are listed in the table at the next page.)

*void * pVauleBuffer*

Address of a buffer to which the value of the parameter is written.
The buffer must be from the same type as the parameter, so that the returned value fits into it.
(You can find the specific types at the table at the next page.)

LONG DeviceNumber

Grabber-number

Return value (LONG):

MVFG_OK

the parameter was read

EMVFG_NO_VXD

Error: grabber isn't installed

EMVFG_CAMPARAM_UNKNOWN

Error: parameter is unknown

EMVFG_NOT_OPEN

Error: driver hasn't been opened with mvfg_open or it was closed

The buffer pValueBuffer must be the right type. Within the table for the parameters you will find the types at the left side (vertical).

<table>
<tr><th>Type</th><th>pcParamName</th><th>Description</th></tr>
<tr><td>LONG</td><td>MVFGPAR_REQFRAME</td><td>Number / position of frames to capture from camera.</td></tr>
<tr><td></td><td>MVFGPAR_TIMEOUT</td><td>Timeout in ms</td></tr>
<tr><td>DWORD</td><td>MVFGPAR_LINELEN</td><td>Length of a image-line for record in bytes</td></tr>
<tr><td></td><td>MVFGPAR_NUMLIN</td><td>Number of image-lines for record</td></tr>
<tr><td>FORMAT_INFO</td><td>MVFGPAR_DATAFORMAT</td><td>Structure with information about the format of the image in the frame grabber memory (current only 8 Bit Black&White format is supported):
<table>
<tr><td><b>iNumberOfPlanes</b></td><td>Number of <b>planes</b> which are read by the grabber.</td></tr>
<tr><td><b>iChannelsPerPlane</b></td><td>Number of channels per <b>plane</b>. (Represents e.g. the color-channels at RGB 8:8:8 = 3 channels).</td></tr>
<tr><td><b>iBitsPerChannel [ ]</b></td><td>Array with the number of bits of each channel.
(e.g. at RGB 5:6:5 = { 5, 6, 5 },
at 8 bit B&W = { 8 })</td></tr>
<tr><td><b>iOffsetNIOC</b></td><td>Offset to the next pixel in one channel in bytes.
(e.g. at RGB 5:6:5 (16 bit) = 2
at RGB 8:8:8 (24 bit) = 3
at B&W 8 bit = 1
at B&W 10 bit = 2)</td></tr>
<tr><td><b>lImageWidth</b></td><td>Width of an camera image in pixel.</td></tr>
<tr><td><b>lImageHeight</b></td><td>Height of an image in pixel.</td></tr>
<tr><td><b>lLineSize</b></td><td>Length of one line in bytes (dependes on the width and the format of an image).
(e.g. Width = 640 Pixel, Format = RGB 8:8:8
lLineSize = 640 * 3 = 1920 bytes)</td></tr>
<tr><td><b>lPlaneSize</b></td><td>The size of one <b>planes</b> in bytes.</td></tr>
<tr><td><b>lFrameSize</b></td><td>The size of one frame (all <b>planes</b>) in bytes.</td></tr>
<tr><td><b>lColorFormat</b></td><td>MVFG_RGB = color, MVFG_GRAY = gray-scale</td></tr>
</table>
</td></tr>
</table>

3.4.4 mvfg_getbufptr

Synopsis:

```
void * WINAPI mvfg_getbufptr( LONG DeviceNumber )
```

Description:

Returns a pointer to the image buffer in the computers main memory. Frames read from the on board memory of the frame grabber are written to this buffer.

The pointer returned is always a pointer to the start of the frame buffer in main memory. If you request frames with the parameter '**MVFGPAR_REQUFRAME**' (see chapter 3.2.2) with a value N, N > 0, you have to add N * frame size to this address to get a pointer to the image.

Sample:

See sample **Getting an image and its measurements**

Parameters:

LONG DeviceNumber

Grabber-number (0 to 3)

Return value (void *):

Returns a pointer to the start of the image buffer in main memory. If the operation fails, the function returns a NULL pointer.

The function mvfg_errmessage cannot be used here because the return value is a pointer to a buffer

3.4.5 mvfg_grab

Synopsis:

```
LONG WINAPI mvfg_grab( DWORD  iCommand,
                       LONG   DeviceNumber )
```

Description:

This function controls the request of frames from the camera and the transfer of captured image data from the On Board Memory of the Frame Grabber to main memory.

In Compatibility Mode, these two actions are encapsulated in one function. In Extended Mode you have to do this in two successive function calls (see chapter 3.2.5).

Sample:

See sample **Getting an image and its measurements**

Parameters:

DWORD iCommand

Behaviour (Constants for iCommand are listed at the table at the next page).

LONG DeviceNumber

Grabber-number

Return value (LONG):

Depends on the value of iCommand (listed at the table at the next page)

iCommand	Description		
GRAB_WAIT	Description	Requests one or more frames from the camera, then copy the frames from grabber memory to main memory and returns to caller.	
	Return-value	MVFG_GRAB_READY	Grab succeed
		EMVFG_TIMEOUT	Error: image couldn't be grabbed within timeout
		EMVFG_NO_VXD	Error: Inspecta isn't installed
		EMVFG_NOT_OPEN	Error: driver hasn't been opened by mvfg_open or it was closed

iCommand	Description		
GRAB_NOWAIT	Description	Requests one or more frames from the camera and returns at once to the caller.	
	Return-value	MVFG_OK	Grab started
		EMVFG_NO_VXD	Error: Inspecta isn't installed
		EMVFG_NOT_OPEN	Error: driver hasn't been opened by mvfg_open or it was closed
GET_STATUS	Description	Get the status of the last grab request. If status 'MVFG_GRAB_READY' is returned, the captured frames are copied to main memory.	
	Return-value	MVFG_GRAB_READY	Grab succeed
		MVFG_NOT_READY	Grab started but not complete yet
		EMVFG_TIMEOUT	Error: image couldn't be grabbed within timeout
		EMVFG_NO_VXD	Error: Inspecta isn't installed
		EMVFG_NOT_OPEN	Error: driver hasn't been opened by mvfg_open or it was closed
GET_STATUS_WAIT	Description	Waits for the end of a previously started grab request. If the function returns, the requested frames are captured from the camera and stored in the main memory.	
	Return-value	MVFG_GRAB_READY	Grab succeed
		EMVFG_TIMEOUT	Error: image couldn't be grabbed within timeout
		EMVFG_NO_VXD	Error: Inspecta isn't installed
		EMVFG_NOT_OPEN	Error: driver hasn't been opened by mvfg_open or it was closed

3.4.6 mvfg_close

Synopsis:

```
LONG WINAPI mvfg_close( LONG DeviceNumber )
```

Description:

This function stops and deactivates the driver.

Sample

See sample **Opening and closing the driver**

Parameters:

LONG DeviceNumber
Grabber-number

Return value (LONG)

MVFG_OK
Driver deactivated
EMVFG_NO_VXD
Error: grabber isn't installed

3.4.7 mvfg_errmessage

Synopsis:

```
LONG WINAPI mvfg_errmessage( LONG iCode )
```

Description:

This function handles a return-value on another MVFG-function (which can return error-codes) and shows a Windows-message-box with the error. If iCode was MVFG_OK, nothing happens.
iCode is returned unchanged.

Sample:

See sample **Opening and closing the driver**

Parameters:

LONG iCode

Return-value of an MVFG-function (error- or function-code).
iCode cannot be obtained from mvfg_getbufptr because mvfg_getbufptr returns no error-codes but a pointer.
All other level1-functions return a function- or error-code.

Return value (LONG):

The parameter iCode is returned unchanged.

3.5 Level-1 Extended Mode

3.5.1 mvfg_open

Synopsis:

```
LONG WINAPI mvfg_open( char *   pcCameraProfile,
                       LONG     DeviceNumber )
```

Description:

The Inspecta is configured by a set of parameters. All parameters are summarized in a parameter block, also called camera profile or camera section.

The function 'mvfg_open()' expects the data of a profile written to a text file. We call this file 'camera file' (see camera profile).
There can be multiple profiles in a camera file, each headed by a unique profile name.

A profile may include the name of a file that includes control sequences for the connected camera. The content of the file is send to the camera via the Inspecta-5 serial interface.

The file L1DEMO.CAM, which is a part of the L1DEMO, shows you some examples of camera profiles. It also shows for the Mikrotron cameras of type MC 131x, how to add a camera configuration file.
.

Sample:
See sample **Opening and closing the driver**

This version of the Inspecta 5 device driver supports only one frame grabber per computer. So, the DeviceNumber have to be set to 0.

Parameters:

*char * pcCameraProfile*
pcCameraProfile points to a c-string, which contains the filename of the camera file and the name of a specific profile name.
(e.g. "Inspecta-4A.cam;TestMode").
Filename and profile name are separated by ";".
A profile in the camera file is marked by "[camera name]".
All parameters needed to initialize the frame grabber have to be listed in the profile. The parameters are read and set automatically by mvfg_open.

LONG DeviceNumber
Grabber-number (0 to 3)

Return value (LONG):

MVFG_OK
Initialization succeed

EMVFG_NO_VXD

Error: Grabber isn't installed

EMVFG_CAMFILE_NOTFOUND

Error: configuration-file doesn't exist.

EMVFG_CAMSECTION_NOTFOUND

Error: Section within the configuration-file doesn't exist.

EMVFG_CAMSTRG_FILE_NOTFOUND

Error: Camera-string-file indicated but not found.

Other

General error

3.5.2 mvfg_setparam

Synopsis:

```
LONG WINAPI mvfg_setparam( char * pcParamName,
                           char * pcParamValue,
                           LONG   DeviceNumber )
```

Description:

Specific parameters of the Inspecta can be set by this function during runtime. The values described below can also be defined statically by an entry in a camera profile. The call of this function overwrites values set by function ‘mvfg_open’.

The effect of a parameter depends of the selected Inspecta-5 mode. This section describes the use of the parameters for the Extended Mode. For Compatibility Mode see Level-1 Compatibility Mode.

Sample:

See sample **Setting and reading parameters**

☝	**All addresses in the descriptions below have the meaning of ‘byte offsets’. The offsets references a position in the image buffers, counted from the start of a buffer.**

Parameters:

*char * pcParamName*
Name of the parameter that is to be changed.
(Constants to use for pcParamName are listed in the table at the next page.)

*char * pcParamValue*
Value to set for this parameter
(you can find allowed values at the table at the next page)

LONG DeviceNumber
grabber-number

Return value (LONG):

MVFG_OK
Parameter is set

EMVFG_NO_VXD
Error: grabber isn't installed

EMVFG_CAMPARAM_UNKNOWN
Error: parameter is unknown

EMVFG_CAMPARAM_BADVALUE
Error: value is not possible

EMVFG_NOT_OPEN
Error: driver hasn't been opened with mvfg_open or it was closed

Miscellaneous parameters

pcParamName	pcParamValue

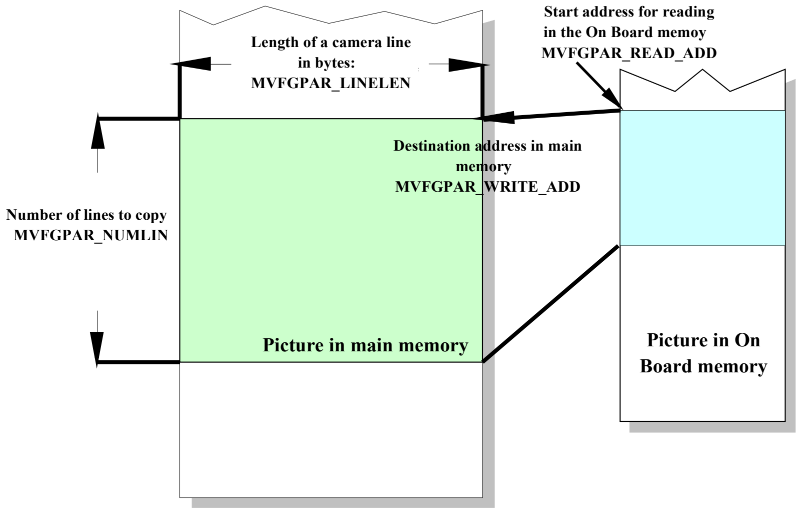
MVFGPAR_EXT_MODE_FLAG	Set/reset the extended mode flag.
	Flag = "0": Compatibility mode on
	Flag = "1": Extended mode on

MVFGPAR_TIMEOUT	Timeout in ms
MVFGPAR_FRAME_CNTR	Activates the In-Frame-Counter of the Inspecta-5. The frame counter counts continuously captured camera frames. The current frame number is written to the first byte respectively to the first two bytes of a captured frame. If the Camera Link interface is configured
MVFGPAR_OUTPUT_PORT	Sets the digital output ports 0 …3 to logic 0 or 1, in dependency of the values in the parameter. Parameter Bit 0 = value for port 0 Parameter Bit 1 = value for port 1 Parameter Bit 2 = value for port 2 Parameter Bit 3 = value for port 3 The current state of the output ports can be read by function ‚mvfg_getparam'.

How to get access to image data - overview

As described in chapter 3.1, 'Overview', frame acquisition from the camera and the transport to user memory is done by two independent components on the Frame Grabber. Frames captured from a camera are stored in the On Board memory of the Frame Grabber. For the user to get access to the image data, the frames have to be copied to a buffer in main memory.
The number of frames and the size of the frames have to be defined before grabbing frames from the camera or to copy them to main memory. The sections below describe the parameters needed to define transactions from the camera and to the main memory.

Copying frames from the On Board memory to main memory


Length of a camera line in bytes: MVFGPAR_LINELEN
Start address for reading in the On Board memoy MVFGPAR_READ_ADD
Destination address in main memory MVFGPAR_WRITE_ADD
Number of lines to copy MVFGPAR_NUMLIN
Picture in main memory
Picture in On Board memory

Before copying data from the On Board memory of the Frame Grabber to the buffer in main memory, you have to define the address you want to read data from and the destination address you want to copy the data to. Additional you have to define the amount of data you want to transfer (see picture on the left). The number of frames to copy is defined by the parameter MVFGPAR_REQFRAME, the size of a frame by the parameters MVFGPAR_LINELEN and MVFGPAR_NUMLIN.

Please note, that the length of a line is defined in bytes per line. E.g. the line length of a RGB camera with 640 pixel/line and 24 bits/pixel, would be defined as

Line length = 640 pixels * 3 byte per pixel = 1920

So the formula for calculating the line length is:

Line Length = Pixel Per Line * Bytes Per Pixel

MVFGPAR_READ_ADDR is the address in the On Board memory from where we want to read data. The address is relative to the start of the buffer. MVFGPAR_WRITE_ADDR is the destination address of the grabber buffer in main memory. These addresses also starts with address 0 for the first byte in the buffer. Please note the parameter MVFG_REQFRAME (see below) that also influences the destination address of the data.

pcParamName	pcParamValue	
MVFGPAR_LINELEN	Length of an image-line to copy to main memory in bytes (e.g. "1920").	
MVFGPAR_NUMLIN	Number of image-lines to copy to main memory (e.g. "480")	
MVFGPAR_REQFRAME	Number / position of the frames to copy. The frames will be read from frame grabber on board memory and written to the target buffer in the main memory of the computer.	"0" ... "n" = write one frame at position “n” in the target buffer in main memory.
		„-2“ ... „-n“ = write (n – 1) frames. The frames will be written at offset MVFGPAR_WRITE_ADDR in the target buffer.
		"-1" = write as many frames as fit into main memory buffer, starting at offset MVFGPAR_WRITE_ADDR.
MVFGPAR_READ_ADDR	Source address (offset) in frame grabber memory to read image data from.	
MVFGPAR_WRITE_ADDR	Data read from grabber memory will be written to this offset in the target buffer in main memory.	

Defining frames to get from the camera

The Frame Grabbers stores data read from the camera in its On Board memory. The transfer is defined by a number of parameters which similar to the parameters described above for the transfer of frames to main memory.
The Inspecta-5 has the ability to treat a part or the whole On Board memory as a ring buffer. Setting the parameter MVFGPAR_G_CONT_FLAG to 1 causes the Frame Grabber to write sampled frames continuously to its memory. If it reaches the end of the defined buffer, it continues writing at the starts of the buffer, and so on. To stop recording, you have to call function 'mvfg_grab()' (see mvfg_grab) with the argument 'GRAB_G_STOP'. This will stop recording by the end of the current frame. If you want to get a number of frames even after you stopped recording, you can define a trailer count. Recording will not stop until the number of frames defined in parameter MVFGPAR_G_TRAILER is captured from the camera.

pcParamName	pcParamValue
MVFGPAR_G_LINELEN	Line length of a frame from the connected camera in bytes. On the one hand, the length of an image line has to be a multiple of 32. On the other hand, the length of a line is always a multiple of the number of tabs the selected pixel router mode uses (see Pixelrouter). **So the lengths of a line has to be always the least common multiple of the used tabs and 32!** **Example:** 2 tabs mode -> least common multiple of 2 and 32 is 32 -> the length of a line has to be a multiple of 32. 10 tabs mode -> least common multiple of 10 and 32 is 160 -> the length of a line has to be a multiple of 160.
MVFGPAR_G_NUMLIN	Number of lines of a frame from the connected camera.
MVFGPAR_G_REQFRAME	Number/count of requested frames from the camera. These frames will be written to the frame grabber on board memory (cf. parameter MVFGPAR_REQFRAME, above).
MVFGPAR_G_READ_ADDR	Reserved
MVFGPAR_G_WRITE_ADDR	Offset in the on board memory of the frame grabber, image data from the camera will be written to.
MVFGPAR_G_TRAILER	Trailer count. Number of frames read from the connected camera after stop signal. The trailer count have to be > 0.
MVFGPAR_G_CONT_FLAG	Continuous flag. "0" = Frame grabber stops after the number of MVFGPAR_G_REQFRAME's frames are recorded. "1" = The frame grabber writes frames from the camera in a circular way to the on board memory. If the defined buffer is full, the frame grabber starts again at the top of the buffer to write images (ring buffer). The continuous recording of frames can be stopped sending the command 'GRAB_G_STOP' (cf. function 'mvfg_grab()) to the frame grabber. After the stop signal is set, a number of MVFGPAR_G_TRAILER frames will be written before the frame grabber stops finally.

pcParamName	pcParamValue
MVFGPAR_G_WRAP_FLAG	Wrap around flag. If the continuous flag is set, the frame grabber handles the on board memory as a ring buffer. If the frame grabber has written the whole buffer and starts again at the top of the buffer, this flag is set to “1” to signal it to the user. Writing any value to it resets the flag.

Inspecta-5 trigger logic

The Inspecta-5 contains hardware logic to generate control signals for the lines CC1 to CC4 of the Camera Link® Interface.

The main part of the logic consists of four programmable timers (Timer A-D), which can be gated to each other and the digital input ports of the frame grabber.

Each timer needs a clock and a start signal as input signal. The output of a timer is a pulse of an adjustable duration.

The output of the timers or the lines of the digital input ports can be connected to the camera control lines CC1 to CC4 of the Camera Link® interface. This gives you the opportunity to control a connected camera by these signals.

☝	**A timer, which is not started by an external signal (e.g. by the end of timer B), has to be toggled after its counter is loaded, to start working. This is done by setting the trigger source to 'Software Trigger' and then set it to the chosen source (e.g. 'Timber B End').**

An additional feature of the Inspecta trigger logic is the possibility to use an external signal to start or stop grabbing of the frame grabber (e.g. by a signal on one of the digital input ports).

The configuration of the trigger logic is done by a number of registers in the frame grabber, which can be set by these parameters:

<table>
<tr><th>pcParamName</th><th>PcParamValue</th></tr>
<tr><td>MVFGPAR_TIMER_A_START
MVFGPAR_TIMER_B_START</td><td>Defines the start event for timer A and B. Possible events:
<table>
<tr><th>Parameter</th><th>Meaning</th></tr>
<tr><td>MVFGVAL_SW_TRIGGER</td><td>A software generated trigger signal.</td></tr>
<tr><td>MVFGVAL_PPIN0
MVFGVAL_PPIN1
MVFGVAL_PPIN2
MVFGVAL_PPIN3</td><td>A signal on one oft the digital input ports 0...3 of the frame grabber.</td></tr>
<tr><td>MVFGVAL_QUADDEC</td><td>Output of the quadrature decoders of the frame grabber.</td></tr>
<tr><td>MVFGVAL_TIMER_A_END
MVFGVAL_TIMER_B_END</td><td>End of timer A or timer B.</td></tr>
</table></td></tr>
<tr><td>MVFGPAR_TIMER_C_START</td><td>Defines the start event for timer C.
Possible events:
<table>
<tr><th>Parameter</th><th>Meaning</th></tr>
<tr><td>MVFGVAL_TIMER_A_ RISING</td><td>Rising edge of the output signal of Timer A.</td></tr>
<tr><td>MVFGVAL_TIMER_B_ RISING</td><td>Rising edge of the output signal of Timer B.</td></tr>
</table>
Timer C and D are mainly used as a slope detector for the falling or raising edge of the output signal of timer A and B.</td></tr>
<tr><td>MVFGPAR_TIMER_D_START</td><td>Defines the start event for timer D.
Possible events:
<table>
<tr><th>Parameter</th><th>Meaning</th></tr>
<tr><td>MVFGVAL_TIMER_A_ FALLING</td><td>Falling edge of the output signal of Timer A.</td></tr>
<tr><td>MVFGVAL_TIMER_B_ FALLING</td><td>Falling edge of the output signal of Timer B.</td></tr>
</table></td></tr>
<tr><td>MVFGPAR_TIMER_A_COUNT</td><td>32 bit counter for timer A. The counter is decremented by the clock of the timer. The output signal of the timer remains active until the counters gets 0. (Pulse duration).
The clock of the timers is calculated by
<b>Timebase/Prescaler</b>.
(see parameter MVFGPAR_TIMER_ACD_CLOCK and MVFGPAR_PRESCALER_ACD) .</td></tr>
<tr><td>MVFGPAR_TIMER_B_COUNT</td><td>Function as timer A, but with a fixed clock of 7,3728 MHz.</td></tr>
<tr><td>MVFGPAR_TIMER_CD_COUNT</td><td>Counter for timer C and D. Function as timer A.</td></tr>
<tr><td>MVFGPAR_TIMER_ACD_CLOCK</td><td>Defines the source for the clock of timer A, C and D.
Possible values:
<table>
<tr><td>MVFGVAL_PRESCALER_ OUT</td><td>Output of the internal clock generator divided by the current value of the ‚Prescaler’. The clock generator has a frequency of 7,3728 Mhz (see MVFGPAR_PRESCALER_ACD).</td></tr>
<tr><td>MVFGVAL_LVAL_EDGE</td><td>Rising/falling edge of the Line Data Valid signal of the connected camera.</td></tr>
</table>
<b>The clock of timer B is fixed to 7,3728 MHz!</b></td></tr>
</table>

<table>
<tr><th>pcParamName</th><th>PcParamValue</th></tr>
<tr><td>MVFGPAR_TRIGGER_SYNC_A
MVFGPAR_TRIGGER_SYNC_B</td><td>Synchronizes the output signal of timer A or B to the Line Data Valid signal of the camera.
<table>
<tr><th>Parameter</th><th>Bedeutung</th></tr>
<tr><td>MVFGVAL_NO</td><td>Don’t synchronize.</td></tr>
<tr><td>MVFGVAL_YES</td><td>Synchronize to LDV.</td></tr>
</table></td></tr>
<tr><td>MVFGPAR_LVAL_EDGE_A
MVFGPAR_LVAL_EDGE_B</td><td>Defines the edge of the Line Data Valid Signals to which timer A respectively. timer B is synchronized.
<table>
<tr><th>Parameter</th><th>Bedeutung</th></tr>
<tr><td>MVFGVAL_RISING</td><td>Rising edge.</td></tr>
<tr><td>MVFGVAL_FALLING</td><td>Falling edge.</td></tr>
</table></td></tr>
<tr><td>MVFGPAR_PRESCALER_ACD</td><td>Divider for timer clock. The base clock of 7,3728 MHz is divided by this value and then used as clock for timer A, C and D. Range: 1-255</td></tr>
<tr><td>MVFGPAR_CC1_SOURCE
MVFGPAR_CC2_SOURCE
MVFGPAR_CC3_SOURCE
MVFGPAR_CC4_SOURCE</td><td>Defines the signal, which is connected to Camera Control Signal CC1, CC2, CC3 or CC4 of the Camera Link® interface.
Defined signals:
<table>
<tr><th>Parameter</th><th>Bedeutung</th></tr>
<tr><td>MVFGVAL_CC_NOP</td><td>The Camera Control Signal is fixed to logically 0.</td></tr>
<tr><td>MVFGVAL_PPIN0
MVFGVAL_PPIN1
MVFGVAL_PPIN2
MVFGVAL_PPIN3</td><td>The lines of the digital input port 0, 1, 2 or 3 are connected directly to one of the CCx lines</td></tr>
<tr><td>MVFGVAL_TIMER_A
MVFGVAL_TIMER_C
MVFGVAL_TIMER_D</td><td>The output signal of timer A, C or D is connected to one of the CCx lines.</td></tr>
</table></td></tr>
<tr><td>MVFGPAR_CC1_POLARITY
MVFGPAR_CC2_POLARITY
MVFGPAR_CC3_POLARITY
MVFGPAR_CC4_POLARITY</td><td>Set the polarity oft the output signal of Camera Control Line CC1...CC4.
Possible values:
<table>
<tr><th>Parameter</th><th>Meaning</th></tr>
<tr><td>MVFGVAL_POS</td><td>Active level is positive.</td></tr>
<tr><td>MVFGVAL_NEG</td><td>Active level is negative.</td></tr>
</table></td></tr>
<tr><td>MVFGPAR_QUADDEC_DIV</td><td>Defines the divider for the output signal of the quad decoders. The decoder uses the signals on input port 1 and 2 to build the output signal.
Range: 1-255.</td></tr>
<tr><td>MVFGPAR_EX_GRAB_START</td><td>Used to start a previously initiated recording by an external signal.
Possible values:
<table>
<tr><th>Parameter</th><th>Meaning</th></tr>
<tr><td>MVFGVAL_CC_NOP</td><td>Deactivated.</td></tr>
<tr><td>MVFGVAL_SW_TRIGGER</td><td>Software Trigger</td></tr>
<tr><td>MVFGVAL_PPIN0
MVFGVAL_PPIN1
MVFGVAL_PPIN2
MVFGVAL_PPIN3</td><td>Pulse on digital input port 0,1,2 or 3.</td></tr>
<tr><td>MVFGVAL_TIMER_A
MVFGVAL_TIMER_B</td><td>Outgoing pulse from timer A or B.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>pcParamName</th><th>PcParamValue</th></tr>
<tr><td>MVFGPAR_EX_GRAB_STOP</td><td>Stops a running recording of frames by an external signal.
Possible values:<table>
<tr><th>Parameter</th><th>Meaning</th></tr>
<tr><td>MVFGVAL_CC_NOP</td><td>Deactivated.</td></tr>
<tr><td>MVFGVAL_SW_TRIGGER</td><td>Software Trigger</td></tr>
<tr><td>MVFGVAL_PPIN0
MVFGVAL_PPIN1
MVFGVAL_PPIN2
MVFGVAL_PPIN3</td><td>Pulse on digital input port 0,1,2 or 3.</td></tr>
<tr><td>MVFGVAL_TIMER_A
MVFGVAL_TIMER_B</td><td>Outgoing pulse from timer A or B.</td></tr>
</table></td></tr>
<tr><td>MVFGPAR_EX_GRAB_START_POL
MVFGPAR_EX_GRAB_STOP_POL</td><td>Defines the edge of the external signal, recording is started or stopped.<table>
<tr><th>Parameter</th><th>Meaning</th></tr>
<tr><td>MVFGVAL_RISING</td><td>Action on rising edge.</td></tr>
<tr><td>MVFGVAL_FALLING</td><td>Action on falling edge.</td></tr>
</table></td></tr>
<tr><td>MVFGPAR_PPIN_INVERSION</td><td>Invert the input signals of the digital input ports of the frame grabber:<table>
<tr><th>Parameter</th><th>Meaning</th></tr>
<tr><td>MVFGVAL_YES</td><td>Yes</td></tr>
<tr><td>MVFGVAL_NO</td><td>No.</td></tr>
</table></td></tr>
<tr><td>MVFGPAR_SW_TRIGGER</td><td>By setting this value to low and high, you can generate a software trigger pulse. This trigger pulse can be used to start timer A or B. Or you can use it as signal for the external start/stop logic (use it to test your trigger logic!).<table>
<tr><th>Parameter</th><th>Meaning</th></tr>
<tr><td>MVFGVAL_LOW</td><td>Set the signal to logic high.</td></tr>
<tr><td>MVFGVAL_HIGH</td><td>Set the signal to logic low</td></tr>
</table></td></tr>
</table>

Examples:

- Record control with trigger logic
- Record stop by an external signal

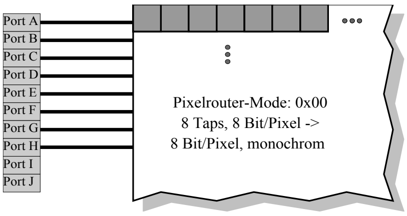
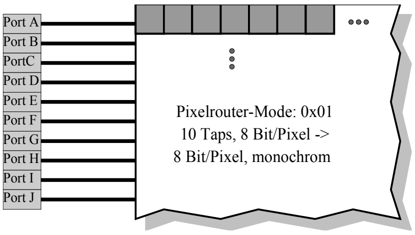
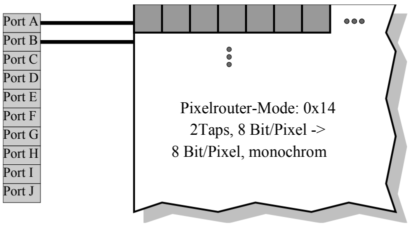
The Inspecta-5 Pixel router

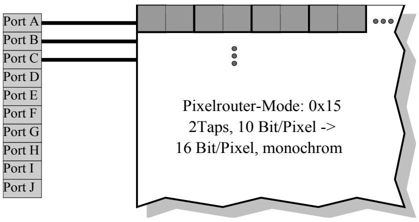
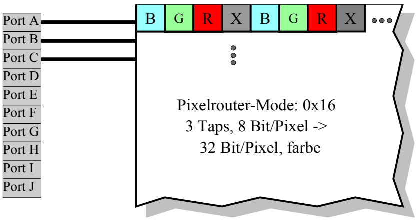
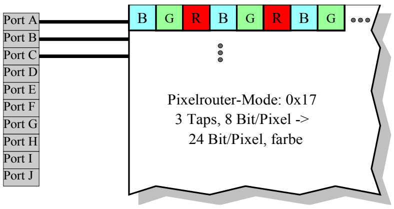
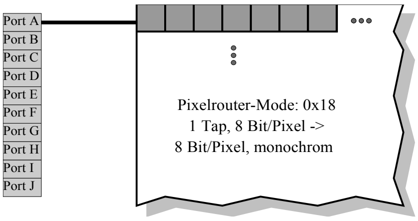
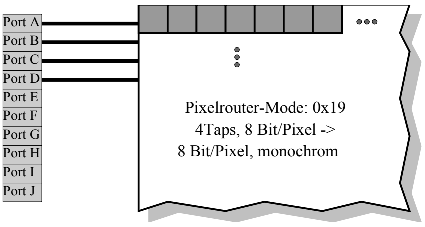
To connect Camera Link® cameras using different numbers of parallel transferred pixels and bits/pixel to the frame grabber, the Inspecta-5 is equipped with a ‘Pixel Router’. Depending on the configuration of the router, cameras with 1 to 10 taps and 8 to 14 bits (color or b/w) can be connected to the frame grabber.

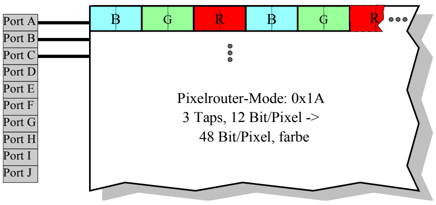
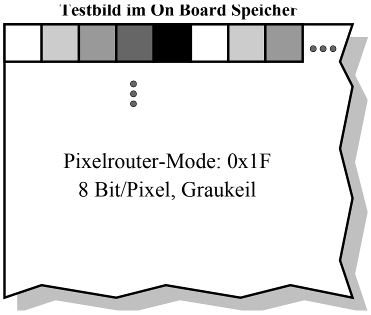
The configuration of the router is done by ‚Magic Numbers’. Each number stays for a specific configuration of the Camera Link® interface. A numbers is set by the function ‘mvfg_setparam()’ using the command ‘MVFGPAR_PIXEL_ROUTER.

pcParamName	PcParamValue
MVFGPAR_PIXEL_ROUTER	MagicNumber. Defines the configuration of the Camera Link® interface. E.g. mvfg_setparam(MVFGPAR_PIXEL_ROUTER, “14”, ID);

The table below shows the possible configurations of the Camera Link® interface up to now:

0x00	8 Taps, 8 Bit/Pixel, Ports ABCDEFGH Full Camera Link®	1 Byte / Pixel, 8 Bit / Pixel, Format: 0:7 0:7 0:7 0:7 0:70:7...  Port A Port B Port C Port D Port E Port F Port G Port H Port I Port J Pixelrouter-Mode: 0x00 8 Taps, 8 Bit/Pixel -> 8 Bit/Pixel, monochrom
0x01	10 Taps, 8 Bit/Pixel, Ports ABCDEFGHI + Steuerleitungen (J), Full Camera Link®.	1 Byte / Pixel, 8 Bit / Pixel, Format: 0:7 0:7 0:7 0:7 0:7...  Port A Port B PortC Port D Port E Port F Port G Port H Port I Port J Pixelrouter-Mode: 0x01 10 Taps, 8 Bit/Pixel -> 8 Bit/Pixel, monochrom
0x14	2 Taps, 8 Bit/Pixel, Ports AB Base Camera Link®	1 Byte / Pixel, 8 Bit / Pixel, Format: 0:7 0:7 0:7 0:7 0:7...  Port A Port B Port C Port D Port E Port F Port G Port H Port I Port J Pixelrouter-Mode: 0x14 2Taps, 8 Bit/Pixel -> 8 Bit/Pixel, monochrom

0x15	2 Taps, 10 Bit/Pixel, Ports ABC Base Camera Link®	2 Byte / Pixel, 10 Bit / Pixel, `Format:` 0:7 8:9 0:7 8:9 0:7 8:9 0:7...  Port A Port B Port C Port D Port E Port F Port G Port H Port I Port J Pixelrouter-Mode: 0x15 2Taps, 10 Bit/Pixel -> 16 Bit/Pixel, monochrom
0x16	3 Taps, 8 Pixel/Bit, Ports ABC Base Camera Link®	4 Bytes / Pixel, 8 Bit / color, 24 Bit color depth. The data is arranged as 32 Bit/Pixel Bitmap, The colors are in the order BGR stored in memory. The three color bytes are extended by a fourth byte (dummy byte) to get DWORD aligned. Format: B0:7 G0:7 R0:7 X0:7 B0:7 G0:7 R0:7 X0:7 B0:7...  Port A Port B Port C Port D Port E Port F Port G Port H Port I Port J B G R X B G R X Pixelrouter-Mode: 0x16 3 Taps, 8 Bit/Pixel -> 32 Bit/Pixel, farbe
0x17	3 Taps, 8 Pixel/Bit, Ports ABC Base Camera Link®	3 Bytes / Pixel, 8 Bit / color, 24 Bit color depth. The data is arranged as 24 Bit/Pixel Bitmap. The colors are in the order BGR stored in memory. Format: B0:7 G0:7 R0:7 B0:7 G0:7 R0:7 B0:7...  Port A Port B Port C Port D Port E Port F Port G Port H Port I Port J B G R B G R B G Pixelrouter-Mode: 0x17 3 Taps, 8 Bit/Pixel -> 24 Bit/Pixel, farbe
0x18	1 Taps, 8 Pixel/Bit, Port A Base Camera Link®	1 Byte / Pixel, 8 Bit / Pixel. Order: 0:7 0:7 0:7 0:7 0:7...  Port A Port B Port C Port D Port E Port F Port G Port H Port I Port J Pixelrouter-Mode: 0x18 1 Tap, 8 Bit/Pixel -> 8 Bit/Pixel, monochrom
0x19	4 Taps, 8 Pixel/Bit, Ports ABCD Medium Camera Link®	1 Byte / Pixel, 8 Bit / Pixel. Order: 0:7 0:7 0:7 0:7 0:8...  Port A Port B Port C Port D Port E Port F Port G Port H Port I Port J Pixelrouter-Mode: 0x19 4Taps, 8 Bit/Pixel -> 8 Bit/Pixel, monochrom

1A	3 Taps, 12 Pixel/Bit, Ports ABCDE Medium Camera Link®	6 Byte / Pixel, 12 Bit / color, 36 color depth. The data is arranged as color image, 16/Bit/color and 48 Bit/Pixel. The colors are in the order BGR stored in memory. Format: B0:7 B8:11 G0:7 G8:11 R0:7 R8:11 B0:7 B8:11...  Port A Port B Port C Port D Port E Port F Port G Port H Port I Port J B G R B G R Pixelrouter-Mode: 0x1A 3 Taps, 12 Bit/Pixel -> 48 Bit/Pixel, farbe
0x1F	Frame Grabber internal test pattern generator.	1 Byte / Pixel, 8 Bit / Pixel, Format: 0:7 0:7 0:7 0:7 0:7 0:7 ...  Testbild im On Board Speicher Pixelrouter-Mode: 0x1F 8 Bit/Pixel, Graukeil

Nomenclature used for describing the order of image data in the On Board Memory:

X:Y	Number and position of used pixels in a byte
Example:	0:7 -> All 8 bits of the byte are used. 0:3 -> bits 0 to 3 are used

A letter can prefix the pixel definition:

B	The blue part of a color image
G	The green part of a color image
R	The red part of a color image
X	By the pixel router inserted dummy value

Pixel with more than 8 bits per pixel are stored in memory in intel format (Little Endian)

Example:

0:7 0:7 0:7	Monochrome picture, 8 Bit/Pixel.
B0:7 G0:7 R0:7	Picture using t 8 Bit/color und 24 Bit color depth. Alignment in memory: blue-green-red
B0:7 B8:11 G0:7 G8:11 R0:7 R8:11 B0:7 B8:11...	Picture using 12 Bit/color and 36 Bit color depth. We need 2 Bytes/color, so we need 6 Byte/pixel in memory. The alignment of the colors in memory: BGR

Before changing the configuration of the Camera Link® interface, stop sampling frames!

If your camera has a configurable interface (as e.g. the Mikrotron MC 13xx cameras) don't forget to set your camera to the same configuration as you choose for the frame grabber.

Line scan cameras

The Inspecta-5 can be driven in a 'full frame camera mode' or in 'line scan camera mode'.
To connect line scan cameras to the Frame Grabber, we have to switch to line scan mode. Setting the parameter MVFGPAR_LS_CAMERA to MVFGVAL_YES does this.

The definition of the number of camera lines to record can be done in two ways:

- Setting the parameters MVFGVAL_G_NUMLIN and MVFGVAL_C_NUMLIN to the number of lines we want to capture. Now the Frame Grabber treats the line scan camera like a full frame camera. That is, after sampling the number of defined lines, the Frame Grabbers notifies the user of getting a new frame.
- Setting the parameter MVFGVAL_C_NUMLIN to 0, causes the Frame Grabber to capture lines from the camera as long as the signal MVFGPAR_LS_FRM_START is active. If the signal gets inactive, recording stops and the user is notified. The number of lines captured can be calculated using the value MVFGPAR_G_DMA_PTR (see mvfg_getparam).

<table>
<tr><th>pcParamName</th><th>PcParamValue</th></tr>
<tr><td>MVFDGPAR_LS_CAMERA</td><td>Selects the ‘Line Scan Camera Mode’:<table><tr><th>Parameter</th><th>Meaning</th></tr><tr><td>MVFGVAL_YES</td><td>The connected camera is a line scan camera. Switches the Frame Grabber to Line Scan Mode.</td></tr><tr><td>MVFGVAL_NO</td><td>The connected camera is not a line scan camera. Switches the Frame Grabber to Full Frame Mode.</td></tr></table></td></tr>
<tr><td>MVFGPAR_LS_FRM_START</td><td>Defines the source for the ‘Frame-Start-Signal’ for capturing line scan camera lines:<table><tr><th>Parameter</th><th>Meaning</th></tr><tr><td>MVFGVAL_CC_NOP</td><td>Signal disabled</td></tr><tr><td>MVFGVAL_SW_TRIGGER</td><td>Software trigger</td></tr><tr><td>MVFGVAL_PPIN0
MVFGVAL_PPIN1
MVFGVAL_PPIN2
MVFGVAL_PPIN3</td><td>Recording is controlled by a signal on port 0,1,2 or 3.</td></tr><tr><td>MVFGVAL_TIMER_A
MVFGVAL_TIMER_B</td><td>Recording is controlled by the output of Timer A or B.</td></tr></table></td></tr>
<tr><td>MVFGPAR_LS_FRM_START_POL</td><td>Defines the polarity of the ‘Frame-Start-Signal’:<table><tr><th>Parameter</th><th>Meaning</th></tr><tr><td>MVFGVAL_POS</td><td>Positive polarity</td></tr><tr><td>MVFGVAL_NEG</td><td>Negative polarity</td></tr></table></td></tr>
<tr><td>MVFGPAR_C_NUMLIN</td><td>Number of lines to capture from line scan camera.</td></tr>
</table>

3.5.3 mvfg_getparam

Synopsis:

```
LONG WINAPI mvfg_getparam( char * pcParamName,
                           void * pValueBuffer,
                           LONG   DeviceNumber )
```

Description:

The function returns the current or value of the Inspecta-5 parameters listed below.

The meaning of the parameters is the same as on function 'mvfg_setparam()' respectively is described below.

Sample:

See sample **Setting and reading parameters**

Parameters:

*char * pcParamName*

Name of the parameter to be read.
(Constants to use for pcParamName are listed in the table at the next page.)

*void * pValueBuffer*

Address of a buffer to which the value of the parameter is written.
The buffer must be from the same type as the parameter, so that the returned value fits into it.
(You can find the specific types at the table at the next page.)

LONG DeviceNumber

Grabber-number

Return value (LONG):

MVFG_OK

The parameter was read

EMVFG_NO_VXD

Error: grabber isn't installed

EMVFG_CAMPARAM_UNKNOWN

Error: parameter is unknown

EMVFG_NOT_OPEN

Error: driver hasn't been opened with mvfg_open or it was closed

☝	**The buffer pValueBuffer must be the right type. Within the table for the parameters you will find the types at the left side (vertical).**

Type	pcParamName	Description
LONG	MVFGPAR_REQFRAME	Returns the number of requested frames in main memory.
	MVFGPAR_G_REQFRAME	Returns the number of requested frames in On Board memory.
	MVFGPAR_TIMEOUT	Timeout in ms
DWORD	MVFGPAR_LINELEN	Length of a image-line for record in bytes
	MVFGPAR_NUMLIN	Number of image-lines for record

	MVFGPAR_EXT_MODE_FLAG	Returns the current state of the 'Extended Mode Flag'.
		Flag = 0: Compatibility mode on
		Flag = 1: Extended mode on

	MVFGPAR_READ_ADDR	Source address (offset) in frame grabber memory to read image data from.
	MVFGPAR_WRITE_ADDR	Data read from grabber memory will be written to this offset in the target buffer in main memory.
	MVFGPAR_MEM_SIZE	Size of the image buffer in main memory of the computer in bytes.
	MVFGPAR_G_READ_ADDR	Reserved
	MVFGPAR_G_WRITE_ADDR	Offset in the on board memory of the frame grabber, image data from the camera will be written to.
	MVFGPAR_G_MEM_SIZE	Size of the on board memory of the frame grabber in bytes.
	MVFGPAR_G_LINELEN	Line length of a frame from the connected camera in bytes.
	MVFGPAR_G_NUMLIN	Number of lines of a frame from the connected camera.
	MVFGPAR_G_TRAILER	Trailer count. Number of frames read from the connected camera after stop signal. The trailer count is always > 0.
	MVFGPAR_G_CONT_FLAG	Returns the current state of the 'Continuous Flag' (cf. function mvfg_setparam()).
	MVFGPAR_G_WRAP_FLAG	Ring buffer flag. Flag = 0: ring buffer was not wrapped around. Flag = 1: ring buffer was wrapped around
	MVFGPAR_C_NUMLIN	Number of line scan camera lines to read.
	MVFGPAR_INPUT_PORT	Reads the current state of the digital input ports of the frame grabber. Returned value Bit 0 = Port0 Returned value Bit 1 = Port1 Returned value Bit 2 = Port2 Returned value Bit 3 = Port3
	MVFGPAR_OUTPUT_PORT	Reads the current state of the digital output ports of the frame grabber. Returned value Bit 0 = Port0 Returned value Bit 1 = Port1 Returned value Bit 2 = Port2 Returned value Bit 3 = Port3

<table>
<tr><th>Type</th><th>pcParamName</th><th>Description</th></tr>
<tr><td>FORMAT_INFO</td><td>MVFGPAR_DATAFORMAT</td><td>Structure with information about the format of the image in the frame grabber memory (current only 8 Bit Black&White format is supported):
<table>
<tr><td><b>iNumberOfPlanes</b></td><td>Number of <b>planes</b> which are read by the grabber.</td></tr>
<tr><td><b>iChannelsPerPlane</b></td><td>Number of channels per <b>plane</b>. (Represents e.g. the color-channels at RGB 8:8:8 = 3 channels).</td></tr>
<tr><td><b>iBitsPerChannel [ ]</b></td><td>Array with the number of bits of each channel.
(e.g. at RGB 5:6:5 = { 5, 6, 5 },
at 8 bit B&W = { 8 })</td></tr>
<tr><td><b>iOffsetNIOC</b></td><td>Offset to the next pixel in one channel in bytes.
(e.g. at RGB 5:6:5 (16 bit) = 2
at RGB 8:8:8 (24 bit) = 3
at B&W 8 bit = 1
at B&W 10 bit = 2)</td></tr>
<tr><td><b>lImageWidth</b></td><td>Width of an camera image in pixel.</td></tr>
<tr><td><b>lImageHeight</b></td><td>Height of an image in pixel.</td></tr>
<tr><td><b>lLineSize</b></td><td>Length of one line in bytes (dependes on the width and the format of an image).
(e.g. Width = 640 Pixel, Format = RGB 8:8:8
lLineSize = 640 * 3 = 1920 bytes)</td></tr>
<tr><td><b>lPlaneSize</b></td><td>The size of one <b>planes</b> in bytes.</td></tr>
<tr><td><b>lFrameSize</b></td><td>The size of one frame (all <b>planes</b>) in bytes.</td></tr>
<tr><td><b>lColorFormat</b></td><td>MVFG_RGB = color, MVFG_GRAY = gray-scale</td></tr>
</table>
</td></tr>
</table>

3.5.4 mvfg_getbufptr

Synopsis:

```
void * WINAPI mvfg_getbufptr( LONG DeviceNumber )
```

Description:

Returns a pointer to the **start** of the image buffer in main memory of the computer. Frames read from the on board memory of the frame grabber are written to this buffer. If frames are read with an offset (cf. parameter **MVFGPAR_WRITE_ADDR**) you have to add the offset to this address to get a pointer to the image.

Sample:

See sample **Getting an image and its measurements**

Parameters:

LONG DeviceNumber

Grabber-number (0 to 3)

Return value (void *):

Pointer

To the beginning of the image-buffer of the Inspecta

NULL

Error

☝	**The function mvfg_errmessage cannot be used for this function, because the return value is a pointer to a buffer not an error code.**

3.5.5 mvfg_grab

Synopsis:

```
LONG WINAPI mvfg_grab( DWORD  iCommand,
                       LONG   DeviceNumber )
```

Description:

This function controls the frame acquisition from the camera to the on board memory of the frame grabber and the transfer from it to the main memory of the computer in Extended Mode. This is done by a number of commands; send to the driver respectively the frame grabber, by this function.

To switch to Extended Mode, set flag 'MVFGPAR_EXT_FLAG' to "1".

You have to wait until commands in extended mode have finished, before you can switch to non-extended mode and vice versa.

New to Inspecta-5 is the 'On Board Memory'. Frames read from a camera are not stored in the main memory of the computer but in a memory available one the frame grabber itself. The user has no access to this memory. To work with the image data, he has to copy the frames from the on board memory to a buffer in main memory.

So, it requires two steps to get an image from a camera to the main memory of your computer:

1. Requests the frame grabber to store images from the camera into his on board memory.
2. Copy the stored frames from grabber memory to the main memory of your computer.

We added some new commands to the function 'mvfg_grab()', to support this 'two phases of image acquisition'. This commands gives you new possibilities and an extra flexibility in grabbing images. E.g. the number and size of the frames requested to capture from camera can differ in number and size of the data copied to main memory. Another example would be to copy image data from one region of the On Board Memory of the Frame Grabber to main memory, while grabbing new frames from a camera to another region in the Grabber Memory. This can be also done simultaneously because of two independent hardware components for these functions.

You may not read data from the same memory where the Frame Grabber is writing data from the camera, because this would give you undefined image data.

Sample:

See sample **Getting an image and its measurements**

Parameters:

DWORD iCommand

Behaviour (Constants for iCommand are listed at the table at the next page).

LONG DeviceNumber

Grabber-number

Return value (LONG):

Depends on the value of iCommand.

iCommand	Description		
GRAB_WAIT	Description	Copy one or more frames from the on board memory of the frame grabber to main memory and returns to caller.	
	Returns	MVFG_GRAB_READY	Grab succeed
		EMVFG_TIMEOUT	Error: image couldn't be grabbed within timeout
		EMVFG_NO_VXD	Error: Inspecta isn't installed
		EMVFG_NOT_OPEN	Error: driver hasn't been opened by mvfg_open or it was closed
GRAB_NOWAIT	Description	Starts the copy of one or more frames from on board memory of the frame grabber and returns at once to the caller.	
	Returns	MVFG_OK	Grab started
		EMVFG_NO_VXD	Error: Inspecta isn't installed
		EMVFG_NOT_OPEN	Error: driver hasn't been opened by mvfg_open or it was closed
GET_STATUS	Description	Get status for image transfer from frame grabber to main memory.	
	Returns	MVFG_GRAB_READY	Grab succeed
		MVFG_NOT_READY	Grab started but not complete yet
		EMVFG_TIMEOUT	Error: image couldn't be grabbed within timeout
		EMVFG_NO_VXD	Error: Inspecta isn't installed
		EMVFG_NOT_OPEN	Error: driver hasn't been opened by mvfg_open or it was closed

iCommand	Description		
GET_STATUS_WAIT	Description	Waits until a previously started copy request for images from the grabber memory to the main memory are finished.	
	Returns	MVFG_GRAB_READY	Grab succeed
		EMVFG_TIMEOUT	Error: image couldn't be grabbed within timeout
		EMVFG_NO_VXD	Error: Inspecta isn't installed
		EMVFG_NOT_OPEN	Error: driver hasn't been opened by mvfg_open or it was closed
GRAB_G_WAIT	Description	Starts image requisition from the camera and waits until all frames are stored in the ‘on board memory’ of the frame grabber. Then it returns to caller.	
	Returns	MVFG_GRAB_READY	Grab succeed
		EMVFG_TIMEOUT	Error: image couldn't be grabbed within timeout
		EMVFG_NO_VXD	Error: Inspecta isn't installed
GRAB_G_NOWAIT	Description	Starts image requisition from the camera and stores data in the ‘on board memory’ of the frame grabber. If the continuous mode is set, frames will be captured frequently until user stops recording. The function returns back to caller, as soon as recording is started	
	Returns	Same as function ‘GRAB_NOWAIT’	
GRAB_G_STOP	Description	Stops a currently running continuous record loop. If you have set the Continuous Flag ‘MVFGPAR_G_CONT_FLAG’ before sending the command ‘GRAB_G_NOWAIT’ to the frame grabber, it will continuously grab images to it’s ‘on board memory’. To stop the recording loop, you have to send this command to the frame grabber. If there are defined a number of ‘Trailer Frames’ (cf. parameter MVFGPAR_G_TRAILER), recording stops not before grabbing this frames. After the stop signal was send, you have to wait for the end of recording, using one of the status functions below.	
	Returns	MVFG_OK	Grab started
		EMVFG_NO_VXD	Error: Inspecta isn't installed
		EMVFG_NOT_OPEN	Error: driver hasn't been opened by mvfg_open or it was closed

iCommand	Description		
GET_G_STATUS	Description	Get status information if a ‘non waiting’ capture request is still running. If recording runs in ‘Continuous Mode’, this command returns valid information only if recording was stopped by command ‘GRAB_G_STOP’. This function not waits until recording is finished but returns at once.	
	Returns	MVFG_GRAB_READY	Grab succeed
		MVFG_NOT_READY	Grab started but not complete until now
		EMVFG_TIMEOUT	Error: image couldn't be grabbed within timeout
		EMVFG_NO_VXD	Error: Inspecta isn't installed
GET_G_STATUS_WAIT	Description	Wait for a previously started recording to frame grabber memory gets finished. If recording runs in ‘Continuous Mode’, this command returns valid information only if recording was stopped by command ‘GRAB_G_STOP’.	
	Returns	MVFG_GRAB_READY	Grab succeed
		MVFG_NOT_READY	Grab started but not complete yet
		EMVFG_TIMEOUT	Error: image couldn't be grabbed within timeout
		EMVFG_NO_VXD	Error: Inspecta isn't installed
		EMVFG_NOT_OPEN	Error: driver hasn't been opened by mvfg_open or it was closed

3.5.6 mvfg_close

Synopsis:

```
LONG WINAPI mvfg_close( LONG DeviceNumber )
```

Description:

This function stops and deactivates the driver.

Sample

See sample **Opening and closing the driver**

Parameters:

LONG DeviceNumber
grabber-number

Return value (LONG)

MVFG_OK
Driver deactivated
EMVFG_NO_VXD
Error: grabber isn't installed

3.5.7 mvfg_errmessage

Synopsis:

```
LONG WINAPI mvfg_errmessage( LONG iCode )
```

Description:

This function handles a return-value on another MVFG-function (which can return error-codes) and shows a Windows-message-box with the error. If iCode was MVFG_OK, nothing happens.
iCode is returned unchanged.

Sample:

See sample **Opening and closing the driver**

Parameters:

LONG iCode

Return-value of an MVFG-function (error- or function-code).
iCode cannot be obtained from mvfg_getbufptr because mvfg_getbufptr returns no error-codes but a pointer.
All other level1-functions return a function- or error-code.

Return value (LONG):

The parameter iCode is returned unchanged.

4 Camera Profile

4.1 Overview

One of the arguments of function ‘mvfg_open’ (see chapter 3.2.1 and 3.3.1) is the name of a camera profile.

Camera profiles are a list of parameters and its values, written to an ASCII file. To host more then one camera profile in a file, each profile is headed by a profile name.

The parameters of a camera profile are read by function ‘mvfg_open’ and used to initialize the Frame Grabber and a connected camera. So, a camera profile consists of Frame Grabber and camera specific values. If a parameter is not defined, a default value is used by function ‘mvfg_open’.

Function ‘mvfg_setparam’ (mvfg_setparam) uses a subset of these parameters to configure the Frame Grabber at runtime.

A parameter name used in a camera profile and the same parameter name used in API function differ in its diction. The same is true for the values that are assigned to the parameters; in a camera profile no symbolic names allowed for the values.

Example: Giving the parameter ‘MVFGPAR_EX_GRAB_START’ a value, using function ‘mvfg_grab’:

```
Mvfg_setparam( MVFGPAR_EX_GRAB_START, MVFGVAL_PPIN0, 0 );
```

Defining the same parameter in a camera profile would look like this:

```
ExGrabStart=1
```

The table below lists all parameters that can be used in a camera profile. It also shows the diction of the names used in API functions and theirs equivalent used in a camera profile. The table after this shows symbolic parameter names used by API functions and theirs equivalent used in camera profiles.

Comparison of the different diction of the parameter names used by API functions and used in a camera profile:

Name of a parameter using function mvfg_setparam	Namer of a parameter in a camera profile
MVFGPAR_LINELEN	LineLen
MVFGPAR_NUMLIN	NumLin
MVFGPAR_REQFRAME	ReqFrame
MVFGPAR_TIMEOUT	Timeout
MVFGPAR_READ_ADDR	ReadAddr
MVFGPAR_WRITE_ADDR	WriteAddr
MVFGPAR_G_READ_ADDR	GreadAddr
MVFGPAR_G_WRITE_ADDR	GwriteAddr
MVFGPAR_READ_ADDR	ReadAddr
MVFGPAR_G_LINELEN	GLineLen
MVFGPAR_G_NUMLIN	GNumLin
MVFGPAR_ G_TRAILER	GTrailer
MVFGPAR_G_REQFRAME	GReqFrame
MVFGPAR_G_CONT_FLAG	GContinuousFlag
MVFGPAR_C_LINELEN	CLineLen
MVFGPAR_C_NUMLIN	CNumlin
MVFGPAR_C_SKIP_LEFT	CSkipLeft
MVFGPAR_C_SKIP_TOP	CSKIPTop
MVFGPAR_LS_CAMERA	LSCamera
MVFGPAR_LS_FRM_START	LSFrmStart
MVFGPAR_LS_FRM_START_POL	LSFrmStartPol
MVFGPAR_TIMER_A_START	TimerAStart
MVFGPAR_TIMER_B_START	TimerBStart
MVFGPAR_TIMER_D_START	TimerCStart
MVFGPAR_TIMER_D_START	TimerDStart
MVFGPAR_TIMER_ACD_CLOCK	TimerACDClock
MVFGPAR_TIMER_A_COUNT	TimerACount
MVFGPAR_TIMER_B_COUNT	TimerBCount
MVFGPAR_TIMER_CD_COUNT	TimerCDCount
MVFGPAR_CC1_SOURCE	CC1Source
MVFGPAR_CC2_SOURCE	CC2Source
MVFGPAR_CC3_SOURCE	CC3Source
MVFGPAR_CC4_SOURCE	CC4Source
MVFGPAR_CC1_POLARITY	CC1Polarity
MVFGPAR_CC2_POLARITY	CC2Polarity
MVFGPAR_CC3_POLARITY	CC3Polarity
MVFGPAR_CC4_POLARITY	CC4Polarity
MVFGPAR_PRESCALER_ACD	PrescalerACD
MVFGPAR_TRIGGER_SYNC_A	TriggerSyncA
MVFGPAR_TRIGGER_SYNC_B	TriggerSyncB
MVFGPAR_LVAL_EDGE_A	LvalEdgeA
MVFGPAR_LVAL_EDGE_B	LvalEdgeB
MVFGPAR_QUADDEC_DIV	QuaddecDiv
MVFGPAR_PPIN_INVERSION	PpinInversion
MVFGPAR_OUTPUT_PORT	OutputPort
MVFGPAR_EX_GRAB_START	ExGrabStart
MVFGPAR_EX_GRAB_STOP	ExGrabStop
MVFGPAR_EX_GRAB_START_POL	ExGrabStartPol
MVFGPAR_EX_GRAB_STOP_POL	ExGrabStopPol
MVFGPAR_PIXEL_ROUTER	PixelRouter
MVFGPAR_PIXEL_ROUTER_ZONES	PixelRouterZones
MVFGPAR_FRAME_CNTR	FrameCntr

Parameters which can only be used in a camera profile:

Parameter	Description
CamString	Name and path of a binary file, which content are send via the serial interface of the Frame Grabber to the connected camera. Normally the file consists of camera and manufacturer specific values, which are used to initialize the camera.
FpgaFile	Name and path of a FPGA file, which content is a specific firmware, used to program the Frame Grabber. As the functionality of the Frame Grabber is defined by its firmware, it is possible to adapt it to special requirements. If there is no FPGA file defined in a profile, a default file is used to initialize the Frame Grabber.

Programming the firmware can only be done by the Mikrotron GmbH.

Comparison of symbolic values used for API functions and theirs equivalent used in camera profiles:

Parameterwert symbolisch	Parameterwert in Kameraprofil
MVFGVAL_SW_TRIGGER	0
MVFGVAL_PPIN0	1
MVFGVAL_PPIN1	2
MVFGVAL_PPIN2	3
MVFGVAL_PPIN3	4
MVFGVAL_QUADDEC	5
MVFGVAL_TIMER_A	6
MVFGVAL_TIMER_B	7
MVFGVAL_TIMER_C	8
MVFGVAL_TIMER_D	9
MVFGVAL_NOP	10
MVFGVAL_PRESCALER_OUT	0
MVFGVAL_LVAL	1
MVFGVAL_NO	0
MVFGVAL_YES	1
MVFGVAL_FALLING	0
MVFGVAL_RISING	1
MVFGVAL_POS	0
MVFGVAL_NEG	1
MVFGVAL_LOW	0

Parameterwert symbolisch	Parameterwert in Kameraprofil
MVFGVAL_HIGH	1
MVFGVAL_TIMER_A_RISING	0
MVFGVAL_TIMER_B_RISING	1
MVFGVAL_TIMER_A_FALLING	0
MVFGVAL_TIMER_B_FALLING	1

The numeric parameter values and theirs symbolic equivalents can be also found in header file ‘MVFGDRV.H’.

4.2 Sample of a camera profile

```
# Profile name
[MC1311 640x480 1071fps ReqFrm -1 400 Trailer]

# Parameters for Frame Grabber configuration
CamMode=0x5000
LineLen=640
NumLin=480
ReqFrame=0
Timeout=2000

ReadAddr=0
GWriteAddr=0
GLineLen=640
GNumLin=480
GTrailer=400
GReqFrame=-1
GContinuousFlag=0

# File with data for camera configuration
CamString=.\MC131_640x480_1071.mcf
```

5 Samples

If you want to build an application, using the Level-1 API, you have to bind your program with the import library **MVFGI5.LIB**. The library resolves references to the Dynamic Link Library **MVFGI5.DLL** of the Inspecta-5.

Function prototyping, parameter names, constants etc. for the programming language 'Microsoft™ C' can be found in the file **MVFGDRF.H**.

All files you will need to build an application are installed by the Inpsecta-5 setup. Libraries are stored in the subdirectory **LIB**, include files in the subdirectory **INCLUDE** of your installation directory.

You can find the sample program 'L1DEMO', which shows you some of the new possibilities of the Inspecta-5 frame grabber, in the installation directory of the Inspecta-5 software.

5.1 Opening and closing the driver

```
/*  This example opens the driver and sets the
 *  parameters for the Testmode. The TestMode is
 *  defined in the section "TestMode” in the file
 *  "DEMO1.CAM”. If the camera-profile-file could
 *  not be found you have to put the complete path
 *  to the file in mvfg_open-parameter.
 */

#include "windows.h"
#include "mvfgdrv.h"      //  for the mvfgd32.dll

//  Windows main-function
int APIENTRY WinMain ( HINSTANCE hInstance,
                       HINSTANCE hPrevInstance,
                       LPSTR     lpCmdLine,
                       int       nCmdShow )
{
    LONG iRc;      //  return-value of this function

    //  Opens the driver, loads and sets the profile.
    //  You can also use another profile than
    //  "TestMode” to use your camera.
    iRc  =  mvfg_open( "Demo1.cam;Testmode", 0 );

    //  If mvfg_open returned another value than
    //  MVFG_OK a message is shown.
    mvfg_errmessage( iRc );

    if ( iRc == MVFG_OK )
    {
        //  Puts a message to the screen that the
        //  driver has been opened.
        MessageBox( NULL,   "Driver opened.”,
                    "MVFG”, MB_OK );
    }


    //  Now we set the set the API to extended mode
    //  to use the new features of the Inspecta-5
    mvfg_errmessage(
      mvfg_setparam( MVFGPAR_EXT_MODE_FLAG, “1”, 0 ));

    //  If the driver has been opened, it must be
    //  closed by mvfg_close at the end of the program
    iRc  =  mvfg_close( 0 );
    mvfg_errmessage( iRc );

    if ( iRc == MVFG_OK )
    {
        //  Puts a message to the screen.
        MessageBox( NULL,   "Driver closed.”,
                    "MVFG”, MB_OK );
    }

    //  If an error occurred during mvfg_close, the
    //  error-code is returned.
    //  If the driver was not open, nothing happens.
    return iRc;
}

}
```

5.2 Setting and reading parameters

```
/*  This example sets a new line length of the
    defined image.
 */

#include "windows.h"
#include "mvfgdrv.h"      //  for the mvfgd32.dll

//  Windows main-function
int APIENTRY WinMain( HINSTANCE  hInstance,
                      HINSTANCE  hPrevInstance,
                      LPSTR      lpCmdLine,
                      int        nCmdShow )
{
    char acBuffer[256]; //  buffer for the messages
    DWORD dwValue;      //  buffer for the white-level

    //  Open the driver, load and set the profile.
    //  You can use another profile than "TestMode"
    mvfg_errmessage(
              mvfg_open("Demo1.cam;Testmode", 0)
                   );

    //  Read the actual line length of the image.
    //  The value is written to dwValue.
    // The instance dwValue must have the
    //  right type. (here the type must be DWORD).
    mvfg_getparam( MVFGPAR_LINELEN, &dwValue, 0 );

    //  Windows-functions to shwo a message
    wsprintf(  acBuffer,
               "Linelen before setting: %d",
                dwValue );
    MessageBox( NULL, acBuffer, "MVFG", MB_OK );

    //  Now the line length is set to 480.
    mvfg_errmessage(
          mvfg_setparam( MVFGPAR_LINELEN, "480", 0 ));

    //  Read the new line length and put a
    //  message to the screen again.
    mvfg_getparam( MVFGPAR_LINELEN, &dwValue, 0 );

    //  Put a message to the screen.
    wsprintf( acBuffer,
              "Line length now: %d",
              dwValue );
    MessageBox( NULL, acBuffer, "MVFG", MB_OK );

    //  Close the driver.
    mvfg_close( 0 );

    return 0;
}
```

5.3 Getting an image and its measurements

```
/*  This example shows how to get the dimensions of
 *  an image. For this the structure FORMAT_INFO is
 *  used. This structure is defined in the file
 *  MVFGDRV.H.
 *  You can use this functions in your own program.
 *  You must use mvfgd32.lib for the mvfgd32.dll.
 */

//  used the first grabber
#define    GRAB_ID    0

//  include the mvfg
#include  "mvfgdrv.h"

//  decleration of global variables
LONG lFrameWidth;
LONG lFrameHeight;
LONG lFrameSize;

//  function declerations
LONG InitMvfg( void );
void GetDimensions( void );
LONG GrabAndCopyImage( void* pBitMap );


/*  Call this function from your program to initialize
 *  he frame-grabber and the camera(s)
 */
LONG InitMvfg()
{
    int iRc;

    //  initialize the grabber and the camera
    iRc = mvfg_open("Demo1.cam;Testmode",GRAB_ID)

    if( iRc  !=  MVFG_OK )
        return mvfg_errmessage( iRc );
    else
        GetDimensions();



    return MVFG_OK;
}


/*  Call this function every time a camera-parameter
 *  changes.
 */
void GetDimensions( void )
{
    FORMAT_INFO sFormat;

    //  The buffer (second parameter) must be of the
    //  right type. Here the type must be FORMAT_INFO.
    mvfg_getparam( MVFGPAR_DATAFORMAT,
                   &sFormat,
                   GRAB_ID );

    //  get the width of the image in pixel
    lFrameWidth     = sFormat.lImageWidth;

    //  get the hight of the image in pixel / lines
    //  (all planes)
    lFrameHeight    = sFormat.lImageHeight *
                      sFormat.iNumberOfPlanes;

    //  get the size of the image in byte (here all
    //  planes)
    lFrameSize      = sFormat.lFrameSize;
}


/*  This function copies a image into main memory.
 *  Be sure that the buffer pBitmap is large enough
 *  (at least lFrameSize bytes)
 *  If in non extended mode, the call of 'mvfg_grab()'
 *  samples one frame from the camera and copy the
 *  image to main memory.
 *  If in extended mode, the function would copy a
 *  frame from the DRAM of the framegrabber to main
 *  memory.
 */
LONG GrabAndCopyImage( void* pBitmap )
{
    int iRc;

    iRc = mvfg_grab( GRAB_WAIT, GRAB_ID );

    memcpy( pBitmap,
            mvfg_getbufptr( GRAB_ID ),
            FrameSize );

    return mvfg_errmessage( iRc );
}
```

```
/*  This function requests a sequence of 8 frames.
 *  The frames are stored in the on board memory
 *  of the framegrabber, starting at position 0.
 *  !!! Be sure the ‘Extended Mode’ flag is set !!!
 */
LONG GrabSequence( void* pBitmap )
{
    int iRc;

    mvfg_setparam( MVFGPAR_REQFRAME, “-9”, 0 ));

    // You have to define the 8 frames by setting
    // the parameter to -9, because you want to use
    // a offset of 180 bytes to the start of the
    // image memory of the framegrabber (read
    // description of parameter MVFGPAR_G_REQFRAME
    mvfg_setparam( MVFGPAR_REQFRAME, “-9”, 0 ));

    // Set the offset to write the frame to
    mvfg_setparam( MVFGPAR_G_WRITE_ADDR, “180”, 0 ));

    // Get the sequence and wait until finished
    iRc = mvfg_grab( G_GRAB_WAIT, GRAB_ID );

    return mvfg_errmessage( iRc );
}
```

```
/*  Copy the second frame from the above sampled
 *  sequence from framegrabber DRAM to computer
 *  memory.
 *  Return a pointer to the image.
 */
LONG GetFrameFromFramegrabber( void* pBitmap )
{
   DWORD dwXSize, dwYSize;

   // Get X and Y size of the frames in DRAM
   mvfg_getparam( MVFGPAR_G_LINELEN, &dwXSize, 0 ));
   mvfg_getparam( MVFGPAR_G_NUMLIN, &dwYSize, 0 ));

   // Set destination offset in main memory
   // the frame should be saved (200 bytes offset
   // from the start of the user buffer).
   mvfg_setparam( MVFGPAR_REQFRAME, "-2", 0 ));
   mvfg_setparam( MVFGPAR_WRITE_ADDR, "200", 0 ));

   // Set offset in DRAM we want to read from
   sprintf( acBuffer, "%u", dwXSize * dwYSize );
   mvfg_setparam( MVFGPAR_READ_ADDR, acBuffer, 0 ));

   // Start transfer to user ram
   iRc = mvfg_grab( GRAB_WAIT, GRAB_ID );

   // Calculate frame address in main memory
   pBitmap = mvfg_getbufptr( GRAB_ID ) + 200;
}

/*--------------------------------------------------*/

/*  The main-function could be WinMain or some thread
 *  or so.
 */
"MAINFUNCTION()"
{
    ...  //  more code

    void * pBitmap;

    //  open and initialize the grabber and the camera
    if( InitMvfg()  ==  MVFG_OK )
    {
        //  Allocate memory for the image-copy.
        //  You can also create a Windows-bitmap here
        //  to display the image.
        pBitmap = malloc( lFrameSize );
    }
    else
    {
        return -1;
    }

    ...  //  more code

    /*  Grab the picture and copy it to the allocated
     *  memory. For example you can create a Windows-
     *  bitmap to display the image. If you use this
     *  in a loop you will get a livepicture.
     */
    GrabAndCopyImage( pBitmap );

    ...  //  more code

    mvfg_close( GRAB_ID );
}
```

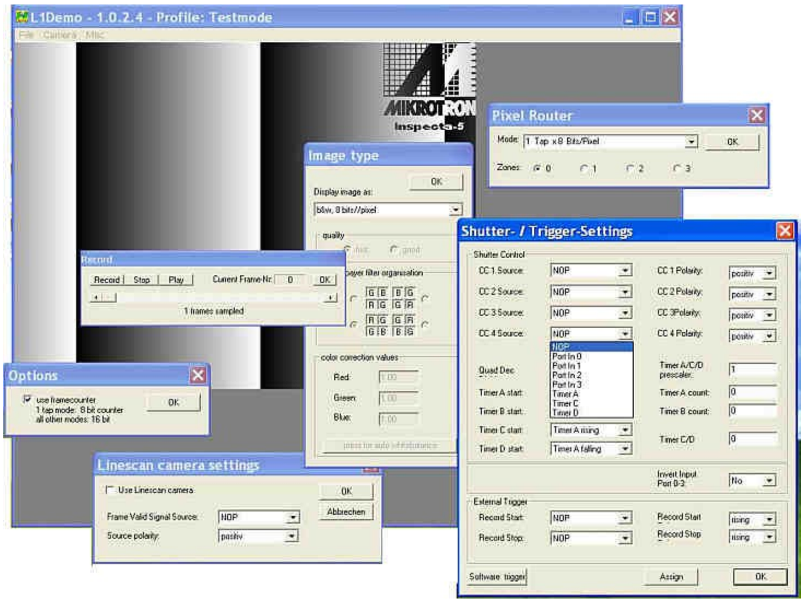
5.4 Record control with trigger logic

```
//  We want the camera to take a photo every 58 ms,
//  with a shutter time of 17ms.
//  For this, we have to set periodically a puls on
//  Camera Control Line 1 (CC1) of the Camera Link®
//  Interface.
//  Timer B is used to trigger Timer A to generate
//  the puls every 58 ms.
//  Don't forget to set the camera to 'async puls
//  width' mode before running the program.

Void StartPeriodicShutter()
{
   // Disable any signals on CC2 to CC4.
   mvfg_setparam( MVFGPAR_CC2_SOURCE,
                  MVFGVAL_CC_NOP, 0 );
   mvfg_setparam( MVFGPAR_CC3_SOURCE,
                  MVFGVAL_CC_NOP, 0 );
   mvfg_setparam( MVFGPAR_CC4_SOURCE,
                  MVFGVAL_CC_NOP, 0 );

    // Set timer A and B inactiv by selecting
    // 'Software Trigger' mode.
    mvfg_setparam( MVFGPAR_TIMER_A_START,
                  MVFGVAL_SOFTWARE_TRIGGER, 0 );
   mvfg_setparam( MVFGPAR_TIMER_B_START,
                  MVFGVAL_SOFTWARE_TRIGGER, 0 );

   // Connect output of Timer A to Camera Control
   // Line CC1.
   mvfg_setparam( MVFGPAR_CC1_SOURCE,
                  MVFGVAL_TIMER_A, 0 );
   // The camera needs a positiv puls on CC1.
   mvfg_setparam( MVFGPAR_CC1_POLARITY,
                  MVFGVAL_POS, 0 );

   // We use the output of the prescaler as
   // clock for Timer A. The clock of Timer
   // B is fix connected to the systemclock
   // of the Frame Grabber (7,3728 MHz).
   mvfg_setparam( MVFGPAR_TIMER_ACD_CLOCK,
                  MVFGVAL_PRESCALER_OUT, 0 );

   // There is no need to divide the clock of
   // the prescaler.
   mvfg_setparam( MVFGPAR_PRESCALER_ACD, “1”, 0 );

   // We need a duratin of 58 ms for the output
   // puls of Timer B. So, we have to load
   // the counter of the timer by a value of
   // 58ms * 7,3278MHz = 427622.
   mvfg_setparam( MVFGPAR_TIMER_B_COUNT,
                  “427622”, 0 );

   // We want a puls of 17ms at the output
   // of Timer A. So, we have to load
   // the counter of the timer by a value of
   // 17ms * 7,3278MHz/1 = 125338.
   mvfg_setparam( MVFGPAR_TIMER_A_COUNT,
                  “125338”, 0 );

   // Timer B is started by its own end.
   // Changing the operation mode of timer B,
   // loads and starts the timer.
   mvfg_setparam( MVFGPAR_TIMER_B_START,
                  MVFGVAL_TIMER_B_END, 0 );


   // Timer A is started by the end of Timer B.
   mvfg_setparam( MVFGPAR_TIMER_A_START,
                  MVFGVAL_TIMER_B_END, 0 );

   // Now we get every 58ms a puls of 17ms
   // on Camer Link Line CC1.
}
```

5.5 Record stop by an external signal

```
//  Sample how to stop recording by
//  an external signal on digital input
//  port 0 of the frame grabber.
//  (Driver has to be set to extended mode).

Void RecExternalStop()
{
   // Enable to stop recording by an external signal
   // on input port 0.
   mvfg_setparam( MVFGPAR_EX_GRAB_STOP,
                  MVFGVAL_PPIN0, 0 );

   // Stop on positive pulse.
   mvfg_setparam( MVFGPAR_EX_GRAB_STOP_POL,
                  MVFGVAL_POS, 0 );

   // No inversion of the input signals on
   // the digital input ports.
   mvfg_setparam( MVFGPAR_PPIN_INVERSION,
                  MVFGVAL_NO, 0 );

   // Set timeout to INFINITY
   mvfg_setparam( MVFGPAR_TIMEOUT,
                  MVFGVAL_INFINITY_TIMEOUT );

   // Use all on board memory of the frame grabber
   // as ring buffer.
   mvfg_setparam( MVFGPAR_G_REQFRAME, “-1” );

   // Set record mode to continuous
   // (‘infinity record loop’).
   mvfg_setparam(MVFGPAR_G_CONT_FLAG, “1” );


   // Start record loop and wait for external
   // trigger signal on input port 0 to stop.
   mvfg_grab( MVFG_NOWAIT, 0 );
}
```

6 Program L1DEMO



Current Frame-Nr: 0
1 frames sampled

☑ use framecounter
1 tap mode: 8 bit counter
all other modes: 16 bit

□ Use Linescan camera
Frame Valid Signal Source: NOP
Source polarity: positiv

Display image as: b&w, 8 bits//pixel
Red: 1.00
Green: 1.00
Blue: 1.00

Mode: 1 Tap x 8 Bits/Pixel
Zones: ● 0 ○ 1 ○ 2 ○ 3

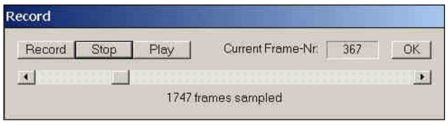
CC 1 Source: NOP
CC 2 Source: NOP
CC 3 Source: NOP
CC 4 Source: NOP
Quad Dec
Timer A start:
Timer B start:
Timer C start: Timer A rising
Timer D start: Timer A falling
CC 1 Polarity: positiv
CC 2 Polarity: positiv
CC 3Polarity: positiv
CC 4 Polarity: positiv
Timer A/C/D prescaler: 1
Timer A count: 0
Timer B count: 0
Timer C/D 0
Invert Input Port 0-3 No
Record Start: NOP
Record Stop: NOP
Record Start rising
Record Stop rising

L1DEMO is a simple program, which shows you some of the fundamental procedures to work with the Inspecta-5 frame grabber.
You can start the program from the Windows Start menu. The Source is stored in the installation directory of the frame grabber driver (directory '<installdir>\Inspecta-5\L1Demo').
There is a pre compiled Version of the program in the directory '<installdir>\Inspecta-5'.

The program is written in 'C' and uses the Windows Win32 API. It is compiled with the Microsoft Visual Studio 6.0.

6.1 Program options

The list below, gives you a short description of the functions you can start from the program menu:

<table>
<tr><th>Function</th><th>Description</th></tr>
<tr><td><b>File->Exit</b></td><td>Exits the program</td></tr>
<tr><td><b>Camera->Stop</b></td><td>Stops the current record mode</td></tr>
<tr><td><b>Camera->Live Picture</b></td><td>Shows a live picture from the camera</td></tr>
<tr><td><b>Camera->
Single frame shoot</b></td><td>Every time you select this function from the menu, the frame grabber captures exactly one new frame from the camera and displays it.</td></tr>
<tr><td><b>Camera->Capture sequence</b>


Current Frame-Nr: 367
1747 frames sampled</td><td>Starts the control buttons for a simple video recorder. The recorder samples the number of frames as defined in the camera profile (see below) and displays it on the window.

Description of the buttons:
<table>
<tr><td>Record</td><td>Set the frame grabber to record mode.
It starts to sampling images from the camera in rotating mode, unless you press the 'Stop' button. While recording, a live picture is shown on the window.</td></tr>
<tr><td>Stop</td><td>Stops sampling images.</td></tr>
<tr><td>Play</td><td>Displays the recorded sequence of images on the window. You can also use the Scrollbar of the dialog to select single frames.</td></tr>
</table></td></tr>
<tr><td><b>Load Profile</b></td><td>Shows a dialog from which you can select a new camera profile. Loading a camera profile configures the frame grabber AND the camera with new parameters. We defined some profiles in the file 'L1DEMO.CAM' in the installation directory of the driver. Below are a list and a short description of the profiles in the cam-file.</td></tr>
</table>

<table>
<tr><th>Function</th><th>Description</th></tr>
<tr><td>

Misc->Image type

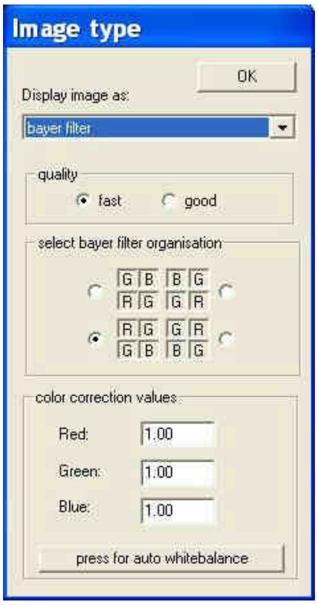


Image type

OK

Display image as: bayer filter

quality ● fast ○ good

select bayer filter organisation

○ GB/RG ○ BG/GR

● RG/GB ○ GR/BG

color correction values

Red: 1.00

Green: 1.00

Blue: 1.00

press for auto whitebalance

</td><td>

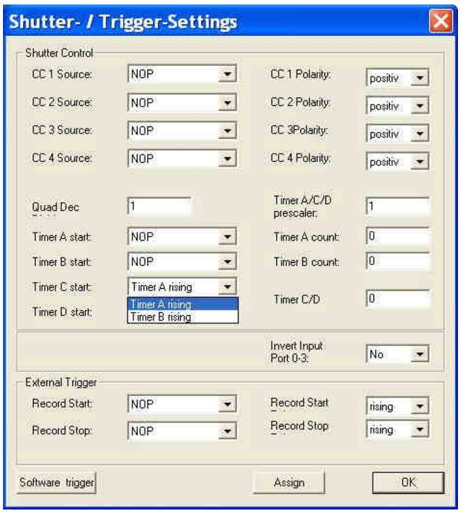
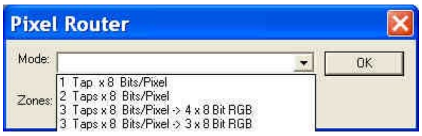
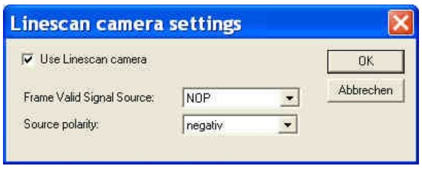
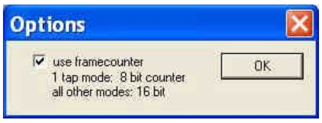
Shows a dialog to select selection the graphic mode for visualization.

Description of the dialog:

<table>
<tr><td>Display image as:</td><td>Selected visualization mode:
• b&w, 8 bit/pixel
• RGB color, 24 bits/pixel
• RGB color, 32 bits/pixel
• RGB color 3x16 bits/pixel
• B&w, 1x10 bits/pixel
• Bayer filter</td></tr>
<tr><td>Bayer Filter On</td><td>Activates/Deactivates the Bayer Filter.</td></tr>
<tr><td>Quality</td><td>Quality of the algorithm used to convert the raw image data from camera to a color image.
Fast: fast conversion, but lower image quality.
Good: slower conversion, but better image quality.</td></tr>
<tr><td>Select Bayer Filter Organization</td><td>Selects the layout of the color filters on top of the sensor. To sele the right layout is important for the right color association.</td></tr>
<tr><td>Color correction values</td><td>Defines the color correction factor for Red, green and blue.</td></tr>
<tr><td>Press for auto white balance</td><td>Starts automatic white balance correction.
To get best results, follow this instructions:
1. Point camera to a white plane
2. Select ‘Live picture’ from menu
3. Adjust exposure to a not to light image
4. Press button for white balance

Please note: different light sources or different light accommodations require a new white balance correction.</td></tr>
</table>

</td></tr>
</table>

<table>
<tr><th>Function</th><th>Description</th></tr>
<tr><td><b>Misc-&gt;Shutter</b>
Shutter- / Trigger-Settings
Shutter Control
CC 1 Source: NOP
CC 2 Source: NOP
CC 3 Source: NOP
CC 4 Source: NOP
CC 1 Polarity: positiv
CC 2 Polarity: positiv
CC 3Polarity: positiv
CC 4 Polarity: positiv
Quad Dec 1
Timer A/C/D prescaler: 1
Timer A start: NOP
Timer B start: NOP
Timer C start: Timer A rising
Timer D start:
Timer A rising
Timer B rising
Timer A count: 0
Timer B count: 0
Timer C/D 0
Invert Input Port 0-3: No
External Trigger
Record Start: NOP
Record Stop: NOP
Record Start rising
Record Stop rising
Software trigger
Assign
OK</td>
<td>Shows a dialog for configuration of the shutter logic and the external start/stop signals.
Description of the dialog:
<table>
<tr><td>CC x Source</td><td>For each of the camera control lines CC1...CC4 you can define a signal of the frame grabber, which is connected to it.</td></tr>
<tr><td>CC x Polarity</td><td>Defines if the signal for line CCx will be inverted.</td></tr>
<tr><td>QuadDecDivider</td><td>Divider for the output signal of the quadrature decoder of the frame grabber.</td></tr>
<tr><td>Timer A/C/D prescaler</td><td>Timer A/C/D has the same. Dividing it to the value of the ‘Prescaler’ can modify the frequency of the clock.</td></tr>
<tr><td>Timer A start
Timer B start
Timer C start
Timer D start</td><td>Defines the start signal of timer A, B, C, und D.
<b>Please take attention to the note at the end of the table.</b></td></tr>
<tr><td>Timer A count
Timer B count
Timer C/D count</td><td>Value for the counter of timer A, B, C and D. The counter is decremented by the timer clock. If the counter reaches zero, the timer gets active.</td></tr>
<tr><td>Invert Input Port 0-3</td><td>Defines if the signals on the digital input port 0,1,2 will be inverted.</td></tr>
<tr><td>Record Start</td><td>Defines the signal for external record start.</td></tr>
<tr><td>Record Stop</td><td>Defines the signal for external record stop</td></tr>
<tr><td>Record Start Pol.</td><td>Polarity of the record start signal.</td></tr>
<tr><td>Record Stop Pol.</td><td>Polarity of the record stop signal.</td></tr>
<tr><td>Software trigger</td><td>For test purposes you can produce a software generated trigger signal by pressing this button.</td></tr>
<tr><td>Assign</td><td>All changes in the input mask will be just getting active after pressing this button.</td></tr>
</table></td></tr>
<tr><td><b>Misc-&gt;Pixel Router</b>
Pixel Router
Mode:
Zones:
1 Tap x 8 Bits/Pixel
2 Taps x 8 Bits/Pixel
3 Taps x 8 Bits/Pixel -> 4 x 8 Bit RGB
3 Taps x 8 Bits/Pixel -> 3 x 8 Bit RGB
OK</td>
<td>In this dialog you can select the mode of the Inspecta-5 Pixel Router. The Pixel Router defines the configuration of the Camera Link interface as well as the conversion of the camera data saved in the On Board Memory of the Frame Grabber.</td></tr>
<tr><td><b>Misc-&gt;Line Scan Camera settings</b>
Linescan camera settings
☑ Use Linescan camera
Frame Valid Signal Source: NOP
Source polarity: negativ
OK
Abbrechen</td>
<td>This dialog configures the Frame Grabber for linescan camera mode.
Description of the dialog:
<table>
<tr><td>Use linescan camera:</td><td>Activates/deactivates the lines can mode of the Frame Grabber.</td></tr>
<tr><td>Frame Valid Signal Source:</td><td>Defines the source for the Frame Valid Signal if grabbing a variable number of camera lines.</td></tr>
<tr><td>Source Polarity</td><td>Polarity of the Frame Valid Signal</td></tr>
</table></td></tr>
<tr><td><b>Misc-&gt;Options</b>
Options
☑ use framecounter
1 tap mode: 8 bit counter
all other modes: 16 bit
OK</td>
<td>Activates/deactivates the ‘in frame counter’ of the Inspecta-5.</td></tr>
</table>

6.2 Pre defined camera profiles in file ‘L1DEMO.CAM’

The L1DEMO program uses a file named ‘L1DEMO.CAM’ to configure the frame grabber and, if necessary, the connected camera. The file is a database of a collection of pre defined camera profiles we tested with the Inspecta-5. If the camera you want to use is not listed in the file, you can use to modify an existing profile to fit to your camera. If you have any problems defining a profile for your camera, please contact our support at

support@mikrotron.de

For a description of the entries in the profile section, have a look at mvfg_setparam and Camera Profile, of this manual.