Sherlock CVB Driver Documentation

Summary

This document describes how to use the Sherlock CVB Driver. This is a high performance driver, with which Sherlock can acquire images from every Common Vision Blox compliant image acquisition device.

2019-01-29

Copyright

Copyright © 2019 STEMMER IMAGING AG, Puchheim.

All rights to this manual are the property of STEMMER IMAGING AG, Puchheim/Germany. It may not be reproduced or copied in printed, electronic or photographic form or translated into another language, either in whole or in part, without the written agreement of STEMMER IMAGING AG.

Sherlock CVB Driver Manual

Content

1 Requirements and Components 5

2 Installation 6

3 Licensing the Sherlock CVB Driver 7

3.1 License with a Sherlock dongle 7

3.2 CVB Image Manager 8

3.3 CVB CameraSuite 8

3.4 Free CVB Trial License 8

4 Getting started 9

4.1 Install the necessary CVB software 9

4.2 Install/Configure Grabber Software 9

4.3 Configure the CVB VIN-Driver 10

4.4 Acquire image with CVB 11

4.5 Configure the Sherlock CVB Driver 11

5 CVBDrv.ini Configuration 12

5.1 Configure Cameras 12

5.2 Parameters 12

5.3 Logging 18

5.4 Example Configurations 19

6 Trigger Configuration 21

6.1 Prepare you Sherlock Investigation 21

6.2 Auto Software Trigger 21

6.3 External Trigger / Hardware Trigger 22

6.4 Software Trigger 22

7 Driver specific Sherlock Instructions 23

7.1 Send Command 23

7.2 Set Shutter 26

7.3 Set Brightness 26

7.4 Set / Get Feature 27

7.5 Get missed frame count 31

7.6 Flush acquired images 31

8 Driver Parameters in Sherlock 32
8.1 Parameters of the Image Window 32
8.2 Driver configuration from the Acquisition Menu Entry 32

9 Additional Information 34
9.1 Performance 34
9.2 Input /Output Support 34
9.3 Support for Images with more than 8Bit 34
9.4 Error Handling 37
9.5 Multiple Sherlock Instances 37

10 Trouble Shooting 39
10.1 Old Images are acquired with Run Once 39
10.2 Old Images are acquired in a triggered setup 40

11 Additional Documentation 41

12 Appendix 42
12.1 GRAB_INFO_CMD values 42
12.2 DC_PARAM values 45
12.3 Changed GenICam Features 46

13 Contact and Feedback 47

Sherlock CVB Driver Manual

1 Requirements and Components

- **Required Software:**
  - Sherlock 7.2.0.3 or higher (32Bit or 64Bit)
  - Common Vision Blox 11.1 or higher
  - One of the following licenses:
    - A Sherlock Dongle (Dongle, Board or System) with a matching MagicNumber purchased from Stemmer Imaging for the article *CVB ACQ FOR SHERLOCK*
    - CVB Image Manager
    - CVB CameraSuite over a GigE Vision or USB3 Vision Camera purchased from Stemmer Imaging
    - Free CVB Trial License: Request a license using the CVB Management Console

- **Components:**
  - "Sherlock CVB driver" contains the files
    - CVBDrv.dll
    - CVBDrv.ini
    - CVBDrvHistory.txt
    - CVBDrv_Sherlock CVB driver Documentation.pdf

2 Installation

Use the installer to start the installation of the driver.

The installer contains the 32Bit and the 64Bit Version of the Sherlock CVB Driver. The nececssary files for 64Bit are installed automatically if Sherlock 64Bit is detected.

A dialog box will be opened during the installation, where you can decide if you want to enable the Sherlock CVB driver and if you want to disable all other drivers.

The installed driver is automatically added to the Drivers.ini file from Sherlock. There is no need to manually change this ini-file.

After installation the ini file is available in the %SP7DATA%\Drivers directory and for Sherlock 64Bit in the %SP7DATA64%\Drivers directory.
The other files are located in the %SP7HOME%\Drivers directory and for Sherlock 64Bit in the %SP7HOME64%\Drivers directory.

3 Licensing the Sherlock CVB Driver

One of the following licenses is required:

- A Sherlock dongle (dongle, board or system) with a matching MagicNumber purchased from your Common Vision Blox Distributor (e.g. Stemmer Imaging GmbH)
- CVB Image Manager
- CVB CameraSuite over a GigE Vision or USB3 Vision Camera purchased from Stemmer Imaging
- Free CVB Trial License: Request a license with the CVB Management Console

Why is a License for this Sherlock driver needed?

The Sherlock CVB driver is based on the Common Vision Blox Image Manager, which is developed by Stemmer Imaging.

The Image Manager is the core of Common Vision Blox with which you can acquire images from a variety of highly optimized drivers for most market leading frame grabbers and other image acquisition hardware with fast access to image data. These drivers are free of charge, but the Image Manager has to be licensed.

Nevertheless if a GigE Vision or USB3 Vision Camera was purchased from Stemmer Imaging the CVB CameraSuite License is included which allows the usage of the Sherlock CVB Driver without extra costs.

3.1 License with a Sherlock dongle

If no CVB license is available(CVB Dongle, CVB CameraSuite or Free CVB Trial License) you can license the driver over the Sherlock dongle.

This license type is always available with a Sherlock System. It can either be a standard Sherlock dongle, a Sherlock license via a board (used with the IPD Vision Appliances) or the Sherlock System License.

To activate the license, you have to enter a MagicNumber to the [System]-Section in the CVBDrv.ini file:

```
[System]

MagicNumber = XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX
```

The MagicNumber is only necessary when the driver is licensed over a Sherlock dongle. With a CVB license the MagicNumber can stay empty.

ATTENTION: In a setup with multiple sherlock instances and multiple CVBDrv.ini files it is necessary that the file with the exact name “CVBDrv.ini” exists and that the Magic Number is available in this file.

You automatically get the MagicNumber after purchasing the license from Stemmer Imaging GmbH, Germany. In order to generate the MagicNumber, Stemmer will need the dongle number with which your Sherlock is licensed. You can find your dongle number in the IPD License Manager.

3.2 CVB Image Manager

If you have a CVB Dongle connected to the System, the Sherlock CVB driver is licensed automatically.

All types of CVB Licenses are supported.

3.3 CVB CameraSuite

If you have an image acquisition device (e.g. a GigE Vision camera) that has been supplied with a *CVB CameraSuite Activation Code*, it can be used to activate the Common Vision Blox CameraSuite functionality.

This license allows the usage of the Sherlock CVB Driver without extra costs. Please use the *CVB Management Console* to activate your license.

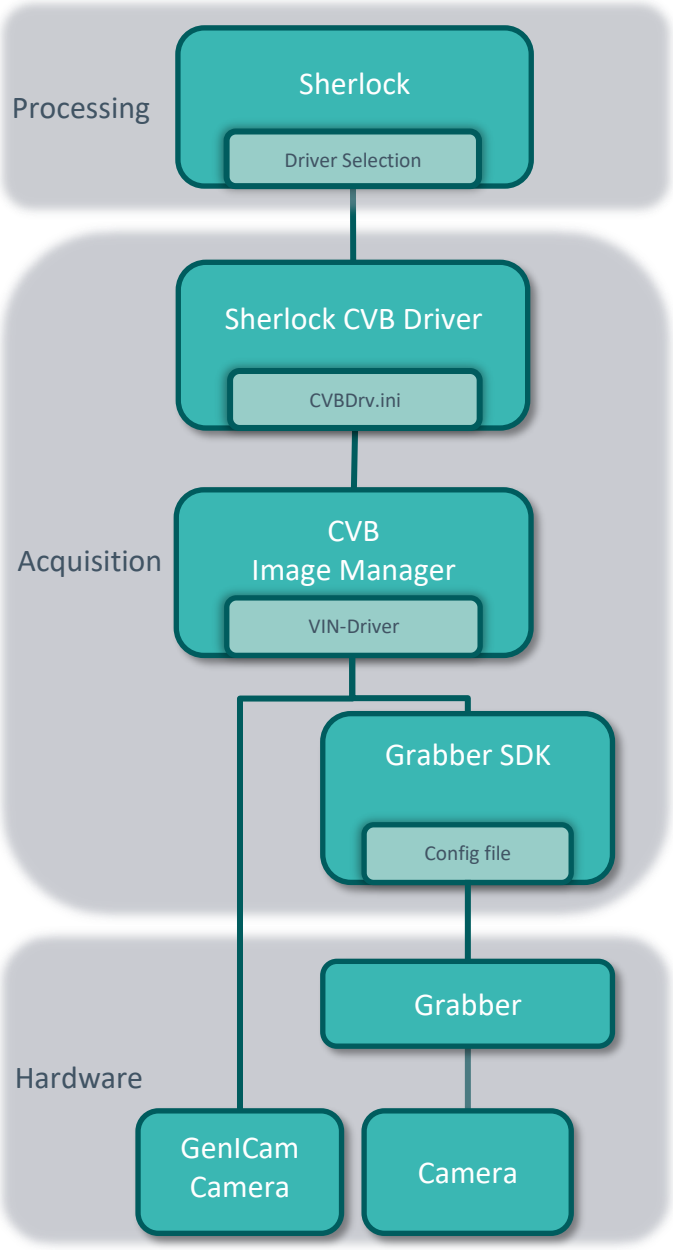
3.4 Free CVB Trial License

If you want to test the Sherlock CVB driver, you can generate a 30 day trial license over the *CVB Management Console.*

Open the Licensing\ Node Locked task and follow the instructions.

4 Getting started

This chapter describes what has to be done to get an image of your device into Sherlock.



4.1 Install the necessary CVB software

The CVB Image Manager need to be installed. This can be done by installing the complete CVB installation, a runtime installation or the Camera Suite installation.

To be able to acquire images from your device the necessary CVB VIN-Driver has to be installed on your system. This can be done with the CVB Installation medium or download the driver on our website:

http://www.commonvisionblox.com/en/download/

The driver for GigE Vision compliant Cameras (GenICam.vin) is installed by default and it is not necessary to be installed separately.

4.2 Install/Configure Grabber Software

If needed install the Grabber SDK and device driver provided by the device manufacturer. Refer to the CVB User Guide for your hardware to know if this step is necessary. The VIN-Driver User Guide can be found in the Start menu (Start -> Programs -> STEMMER IMAGING -> Common Vision Blox -> Documentation -> Hardware).

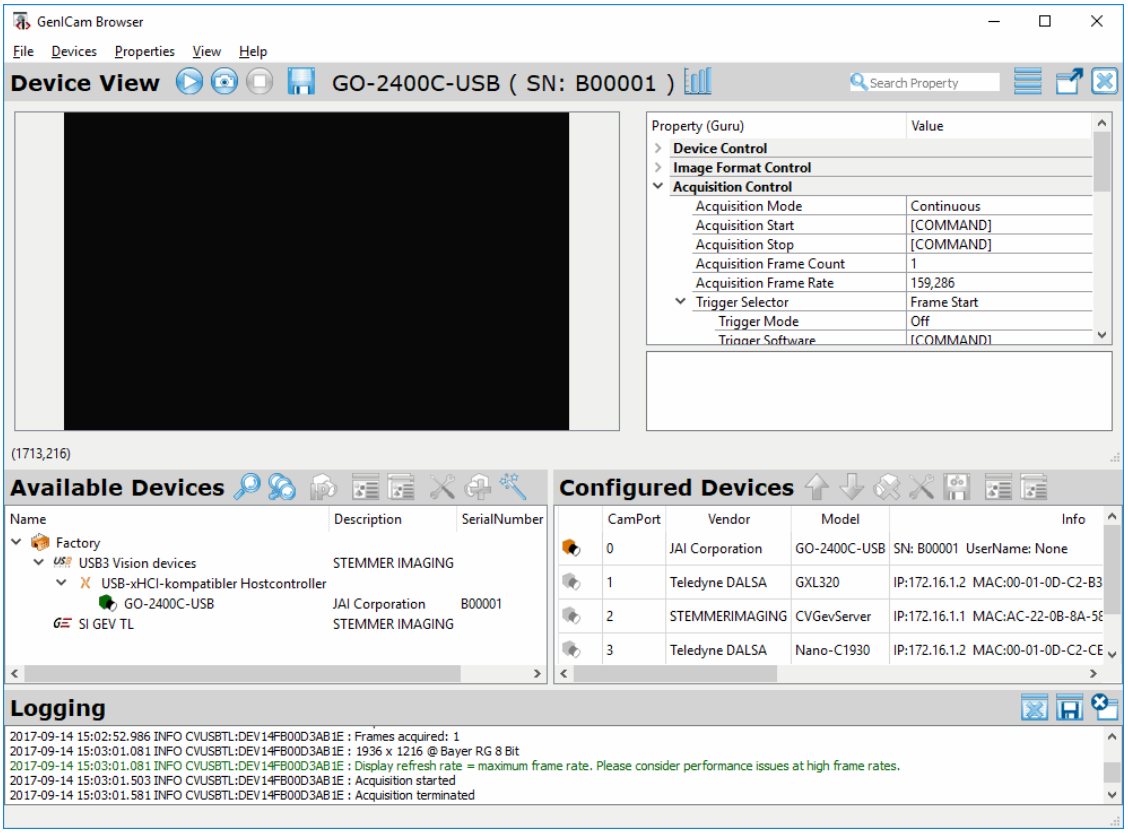
This User Guide has a Quick start or getting started guide which helps you to configure the hard- and software.

If you get an Image over the software from the Grabber manufacturer (e.g. Teledyne Dalsa CamExpert, Silicon Software MicroDisplay) you can go to the next step and configure the CVB VIN-Driver.

4.3 Configure the CVB VIN-Driver

There is no common way how to configure the image acquisition hardware. It depends on what hardware is used. Please refer to the Getting Startet of the *Common Vision Blox User Guide* for your hardware (Start -> Programs -> STEMMER IMAGING -> Common Vision Blox -> Documentation -> Hardware) in order to configure your camera and/or frame grabber. For GenICam compliant Cameras use the Common Vision Blox GenICam Browser to configure your devices.

For browsing, configuring and changing available and configured GenICam complient devices and features the CVB CenICam Browser can be used.



Common Vision Blox GenICam Browser

The GenICam Browser was specifically designed to support all kinds of GenICam compliant devices like GigE Vision, USB 3 Vision cameras and 3rd Party. GenICam compliant transport layer is more flexible and intuitiv which makes it the prevered software.

4.4 Acquire image with CVB

Before trying to get images in Sherlock, check if you get an image in our Common Vision Blox Viewer.

(Start -> Programs -> STEMMER IMAGING -> Common Vision Blox).

Open it and load your desired VIN-Driver located in %CVB%Drivers (e.g. GenICam.vin).

4.5 Configure the Sherlock CVB Driver

Open the CVBDrv.ini file which is located in the drivers folder of Sherlock.

Change the default settings to the needed VIN-Driver.

```
e.g.:       Driver = %CVB%\Drivers\GenICam.vin
```

More information about all parameters in CVBDrv.ini file can be found in the following chapter CVBDrv.ini Configuration.

If this is done, you are ready to start Sherlock and acquire images.

In Sherlock, the currently loaded drivers can be checked over Options -> Acquisition -> Configure driver loading and currently loaded drivers.

5 CVBDrv.ini Configuration

The Sherlock CVB Driver uses the CVBDrv.ini file where all cameras have to be configured. The file can be found in the %SP7DATA%\Drivers or %SP7DATA64%\Drivers directory.

The rows starting with a '#' are just for documentation and do not take effect on the configuration.

5.1 Configure Cameras

The important entries of the CVBDrv.ini should look like this:

```
[Camera0]
Driver = %CVB%\Tutorial\ClassicSwitch.emu

[Camera1]
Driver = %CVB%\Tutorial\ClassicSwitch.emu
```

Each camera section begins with `[CameraN]`, where N is the index of the camera, starting with 0. For every camera you have to load the desired driver (*.vin) with the command Driver = 'path', e.g. `Driver = %CVB%\Drivers\GenICam.vin` for the GeniCam driver.

There are some example configurations in the bottom of the ini file or in this document in chapter Example Configurations

5.2 Parameters

This chapter describes all parameters of the CVBDrv.ini file which are available for every camera section.

5.2.1 Driver

```
Driver = %CVB%\Tutorial\ClassicSwitch.emu
```

This specifies the driver, which is loaded with the configured camera. Every driver or image source, which is supported by Common Vision Blox, is supported by the Sherlock CVB driver as well.

Additional information about the listed drivers below can be found in the Common Vision Blox Online Help under Image Manager->CVB Technology->Good To Know->Supported File Formats.

VIN-Driver (*.vin)

The Video Interface (VIN) file format is used for all Common Vision Blox drivers

Emulation Files (*.emu)

An EMU file is a simple text file defining a number of images to be loaded. Of course all images must have the same size and the same number of planes. It behaves as a vin-file so that it can be used to emulate Common Vision Blox drivers.

Video Files (*.avi/ *.wmv / *.mpg)

This driver enables access to DirectShow compatible video files. Among the supported file types are **AVI, WMV** and **MPEG** formats, but effectively the selection of compatible file formats depends on the codecs installed on your system (for example after installing the free DivX codec on your machine, this driver is also able to playback DivX encoded video streams). As a general rule of thumb, most video formats that can be replayed using Microsoft Media Player can also be used with this driver.

Graphic File Formats (e.g. *.bmp)

Common Vision Blox supports several standard and non-standard image file formats:

- **BMP** (*.bmp)
- **TIFF** (*.tif, *.tiff, *.xif)
- **JPEG** (*.jpg, *.jpeg, *.jpe, *.jif, *.jfif, *.thm)
- **JPEG2000** (*.j2k, *.jp2, *.jp2k, *.jpc, *.jpx)

5.2.2 CameraPort

`CameraPort = 0`

Specifies the Camera Port of the driver. If more than one camera uses the same driver, the driver can be loaded several times. Then the `CameraPort` parameter specifies, which camera of the available cameras has to be used.

Example:

Two GenICam GigE cameras are connected to the system. The GeniCam driver %CVB%\Drivers\GenICam.vin is loaded for `[Camera0]` and `[Camera1]`.

In `[Camera0] CameraPort` is set to 0.

In `[Camera1] CameraPort` is set to 1.

Look into Two GigE-Cameras for an example configuration.

5.2.3 Board

`Board = 0`

If more than one image acquisition device of the same type is installed in the same system the same driver can be loaded several times. Then the board parameter specifies which board of the available image acquisition devices has to be used.

The `Board` parameter may also be needed for Firewire and USB cameras.

Look into Four CL-Cameras on 2 Dalsa X64 CL-Express boards for an example configuration.

5.2.4 AutoSoftwareTrigger

`AutoSoftwareTrigger = 0`

This value deactivates the auto software trigger feature which is activated by default. See chapter External Trigger / Software Trigger / Free Run for more information.

If you want to use an external camera trigger it is enough to activate the external camera trigger value in the image window. It is not necessary to deactivate the AutoSoftwareTrigger feature.

5.2.5 TriggerSelector

`TriggerSelector = MultiFrame`

Only the GenICam.vin driver uses the TriggerSelector parameter.

The Sherlock CVB Driver changes TriggerMode and TriggerSource automatically. By default, it uses the TriggerSelector “FrameStart”. If another Selector value should be used it can be configured with the TriggerSelector parameter.

5.2.6 TriggerSource

`TriggerSource = Line1`

Only the GenICam.vin driver uses the TriggerSource parameter.

The Sherlock CVB Driver changes TriggerMode and TriggerSource automatically. By default, it reads out the TriggerSource value during initialization of the driver and uses this value for external trigger. For software trigger, this feature is changed to “Software” automatically. The driver resets the value to the initial value while unloading.

If you want to set a specific value and do not want to worry about the value in the camera, you can use the TriggerSource parameter to set the trigger source, which is used for external trigger.

5.2.7 BlackImageOnError

```
BlackImageOnError = 1
```

If activated the driver sends a Black Image instead an error when the acquisition failed. Possible errors are:

- Acquisition aborted error
- Timeout error
- Camera disconnected (this causes an acquisition aborted error)

This can be useful if the image window of Sherlock should continue with the instructions within the image window. Especially when communication is done within the image window block.

5.2.8 SafeMode

```
SafeMode = 1
```

Activates the Safe Mode. This means that the image data is always copied even If it is possible to give Sherlock the existing image data.

But it is slower.

This can help to avoid a crash after a reconnect when old image data is accessed which was released for the reconnect. Some Sherlock functions doesn't check if the image window has valid image data.

5.2.9 SetShutterIDDeviceControlID

```
SetShutterIDeviceControlID = 0x00000104
```

If available the Set Shutter instruction in Sherlock tries to use GenICam, to set the exposure time of the camera. If GenICam is not available, it sets the exposure time over CVB IDeviceControl with the ID 0x00000104. On some drivers, the ID for setting the exposure time is different to the default ID. Then you can use this parameter to change the ID which is used.

To find out the correct ID for your driver, please read IDeviceControl.

5.2.10 SetBrightnessIDeviceControlID

```
SetBrightnessIDeviceControlID = 0x00000100
```

If available the Set Brightness instruction in Sherlock tries to use GenICam to set the gain of the camera. If GenICam is not available, it sets the gain over CVB IDeviceControl with the ID 0x00000100. On some drivers, the ID for setting the gain is different to the default ID. Then you can use this parameter to change the ID which is used.

To find out the correct ID for your driver, please read IDeviceControl.

5.2.11 MapTo8BitWindowStart

```
MapTo8BitWindowStart = 0

MapTo8BitWindowStartR = 0
MapTo8BitWindowStartG = 0
MapTo8BitWindowStartB = 0
```

This parameter can be used, if images with more than 8 bit are acquired. Then this parameter is used as the start of the window where the incoming image is mapped to 8Bit.

If you want to have a seperate window start Point for every plane use the MapTo8BitWindowStart[R,G,B] Parameter.

More details about windowing can be found in Support for Images with more than 8Bit

5.2.12 MapTo8BitWindowWidth

```
MapTo8BitWindowWidth = 0

MapTo8BitWindowWidthR = 0
MapTo8BitWindowWidthG = 0
MapTo8BitWindowWidthB = 0
```

This parameter can be used, if images with more than 8 bit are acquired. Then this parameter is used as the width of the window where the incoming image is mapped to 8Bit.

Instead of using the width you can use 8BitMappingWindowEnd.

If you want to have a seperate window width for every plane use the MapTo8BitWindowWidth[R,G,B] Parameter.

More details about windowing can be found in Support for Images with more than 8Bit

5.2.13 MapTo8BitWindowEnd

```
MapTo8BitWindowEnd = 0

MapTo8BitWindowEndR = 0
MapTo8BitWindowEndG = 0
MapTo8BitWindowEndB = 0
```

This parameter can be used, if images with more than 8 bit are acquired. Then this parameter is used as the end of the window where the incoming image is mapped to 8Bit.

If you want to have a seperate window end for every plane use the MapTo8BitWindowEnd[R,G,B] Parameter.

Instead of using the end you can use 8BitMappingWindowWidth.

More details about windowing can be found in Support for Images with more than 8Bit

5.2.14 ObserveGenICamConnection

```
ObserveGenICamConnection = 0
```

This parameter can be used, if the camera can't handle the disconnect or reconnect.
Set to zero it deactivates the Disconnect/Reconnect handling for the configured camera. By default it is activated.

5.2.15 DisableInputs

```
DisableInputs = 1
```

Hides all available inputs of this camera in Sherlock. Useful when you want to use only the inputs of your IO Card and want to be sure that the inputs of the camera are not used by mistake.

5.2.16 DisableOutputs

```
DisableOutputs = 1
```

Hides all available outputs of this camera in Sherlock. Useful when you want to use only the outputs of your IO Card and want to be sure that the outputs of the camera are not used by mistake.

5.2.17 TrevistaAcquisition

```
TrevistaAcquisition = 1
```

This activates an acquisition mode which is compatible to the Trevista System. Get the next 4 images from the CVBDriver and combines them to one mono8 planar Image. The first image has to be triggered by Sherlock over this driver. The next 3 images are software triggered automatically by the TrevistaAcquisition functionality.

This mode is only useful if you purchased a Trevista System from Stemmer Imaging.

5.2.18 ExtractPlane

```
ExtractPlane = 0
```

If set to a plane Index (0:red, 1:green 2:blue) this plane is extracted as mono image from a RGB image. This works also for RGB16Bit (48Bit) which is the normal use case because Sherlock does not support RGB16Bit Images. Default value is -1 which means that this feature is not used. If a plane Index is set which is not available this feature will be ignored.

5.2.19 TriggerOnField

If trigger on field should be used with the configured driver it need to be configured manually with this parameter. By default, asynchronous trigger is used if external camera trigger is enabled. In CVB the Mode is called CVC_C_TRIGGER_ASYNCRON for the asynchronous trigger.

If TriggerOnField is set to 1 external camera trigger is enabled and the ITrigger Interface is supported by the used CVB VIN Driver the used Mode is CVC_C_TRIGGER_FIELD.

```
TriggerOnField = 1
```

5.3 Logging

The Logging parameters influence the way in which the driver logs warnings, errors and additional information. By default the driver is configured with LogLevel “warn” and the destination “application”.

This means that warnings and errors are logged to the Sherlock monitor.

5.3.1 Activate / Deactivate Logging / Change LogLevel

```
Logging = warn
```

Logging is deactivated if Logging is set to "none". It is enabled if one of the other strings is used which represents the LogLevel off The Logging:

- **none** : Logging is deactivated
- **info** : Warnings errors and information will be logged
- **warn** : Warnings and errors will be logged
- **err** : Only errors will be logged
- **all** : Everything will be logged

5.3.2 Logging Destination

`LogDestination = file`

Set the logging destination:

- **application**: log to the Sherlock monitor (trace information will not be logged in the application with Logging=all)
- **file** : log to file. Filename is used from `LogFileName`
- **console** : log to console. Console will be opened automatically
- **trace** : log to trace
- **stdout** : log to stdout

5.3.3 Logging File Name

`LogFileName = %sp7data64%\Drivers\CVBDrv.log`

Filename used when `LogDestination` is set to file

5.4 Example Configurations

5.4.1 Two GigE-Cameras

```
[Camera0]
Driver = %CVB%\Drivers\GenICam.vin
CameraPort = 0

[Camera1]
Driver= %CVB%\Drivers\GenICam.vin
CameraPort = 1
```

5.4.2 One GigE-Camera and One AVT Firewire Camera

```
[Camera0]
Driver = %CVB%\Drivers\GenICam.vin
CameraPort = 0

[Camera1]
Driver = %CVB%\Drivers\cvAVT1394.vin
CameraPort = 0
```

5.4.3 Four CL-Cameras on two Dalsa X64 CL-Express frame grabbers

```
[Camera0]
Driver = %CVB%\Drivers\CVX64CLExpress.vin
CameraPort = 0
Board = 0

[Camera1]
Driver= %CVB%\Drivers\CVX64CLExpress.vin
CameraPort = 1
Board = 0

[Camera2]
Driver = %CVB%\Drivers\CVX64CLExpress.vin
CameraPort = 0
Board = 1

[Camera3]
Driver= %CVB%\Drivers\CVX64CLExpress.vin
CameraPort = 1
Board = 1
```

6 Trigger Configuration

This chapter describes the different trigger options with the Sherlock CVB Driver and gives an important hint to prevent issues, which you can have in triggered configurations. Especially with multi camera configurations.

6.1 Prepare you Sherlock Investigation

Sherlock is an application, which processes the images sequential. If you want to use a triggered configuration keep in mind that the camera need to be configured and prepared for the trigger you want to send. It is also important to know that by default Sherlock starts the acquisition when the Sherlock program executes the Image window instruction.

With triggered configuration, this is could be too late. Often we want to send a software trigger manually or a hardware trigger synchronized with a PLC before the Image window instruction. For example, trigger all cameras before the first image window instruction.

To be able to do that the Sherlock CVB Driver supports the "StartAllAcquisitions" Command which starts the acquisition of all configured cameras.

We strongly recommended starting the acquisition in advance with the "StartAllAcquisitions" command within a subroutine where the "Execute before continuous investigation" checkbox is checked.

Additionally you need to activate the external camera trigger check box in the Sherlock image window in hardware trigger and manual software trigger configurations.

This is important to know especially with multi camera configurations as you can run into unwanted behavior.

For example, you send an external trigger to the second camera before the first camera. Then the trigger is ignored for the second camera, as this camera is not prepared for the trigger.

This is less important for the Auto Software Trigger configuration as it will normally work without any problems. However, even with this configuration a StartAllAcquisitions command will speed up the acquisition of the first image.

6.2 Auto Software Trigger

If a GenICam compliant device is used the Sherlock CVB Driver automatically sends a software trigger to acquire the images. This is called Auto Software Trigger and is activated by default if the parameter is not used in the camera configuration of the CVBDrv.ini file. Even if Live Image (Free Run) is selected in Sherlock, a Software Trigger is used.

If you want to have a free running camera you need to disable the Auto Software Trigger by using the Parameter AutoSoftwareTrigger = 0 for every camera configuration in the initialization file.

6.3 External Trigger / Hardware Trigger

If a hardware trigger need to be used the external camera trigger check box in the Sherlock image window need to be activated. Then the initial trigger source is used which was configured when the driver was loaded. This means that the needed Trigger Source need to be configured before Sherlock is started.

6.4 Software Trigger

It might be useful to send a software trigger manually over the “Trigger” Sherlock command somewhere in the Sherlock program. To do this the auto software trigger need to be disabled in the CVBDrv.ini file. Then you can use the Send Command feature to send a software trigger manually.

If the camera is not configured for software trigger and the acquisition is not already started the driver will do this automatically with the first call of the trigger command. To prevent this and save some time we always recommend

- Start the acquisition in advance with the “StartAllAcquisitions” command within a subroutine where the “Execute before continuous investigation” checkbox is checked.
- Activate the external camera trigger checkbox of the image window.

7 Driver specific Sherlock Instructions

The driver specific commands (IO:Camera Instructions) in Sherlock can be used with the Sherlock CVB driver. This chapter describes the supported commands.

7.1 Send Command

With the Send Command Instruction you can send the driver a message as string.

Following functionality is implemented with the Send Command instruction.

7.1.1 Manual Software Trigger

By default the driver uses the AutoSoftwareTrigger to send a software trigger to the camera if a GenICam compliant camera is used.

A manual software trigger can be used, if you need to start the acquisition of one image at a specific time in your Sherlock investigation and not directly before the image windows instruction in the sherlock program. It can also be used to speed up an investigation when more than 1 camera is used (described in the next chapter).

Due to the fact that Sherlock controls the TriggerMode of the camera to be able to show live images the **external camera trigger** has to be enabled. In this case the manual software trigger is interpreted as an external trigger. See chapter External Camera Trigger. This activates the trigger when it is necessary. Unlike the auto software trigger which sets the trigger source in the camera to software it has to be set manually by the user so that the camera accepts the software trigger.

You can trigger every camera separately with the string “Trigger” in the Send Command instruction. Then the CamIndex in the instruction is used to identify the camera.

You can also send the string “TriggerAll” to trigger all configured cameras within a very short time.

When sending the first Software Trigger in an investigation the driver will enable the Trigger automatically in the camera over the Trigger Interface (ITrigger).

SoftwareTrigger is supported with every driver, which supports the CVB Software Trigger Interface (ISoftwareTrigger)or with every GenICam compliant camera, which uses one of the following feature names:

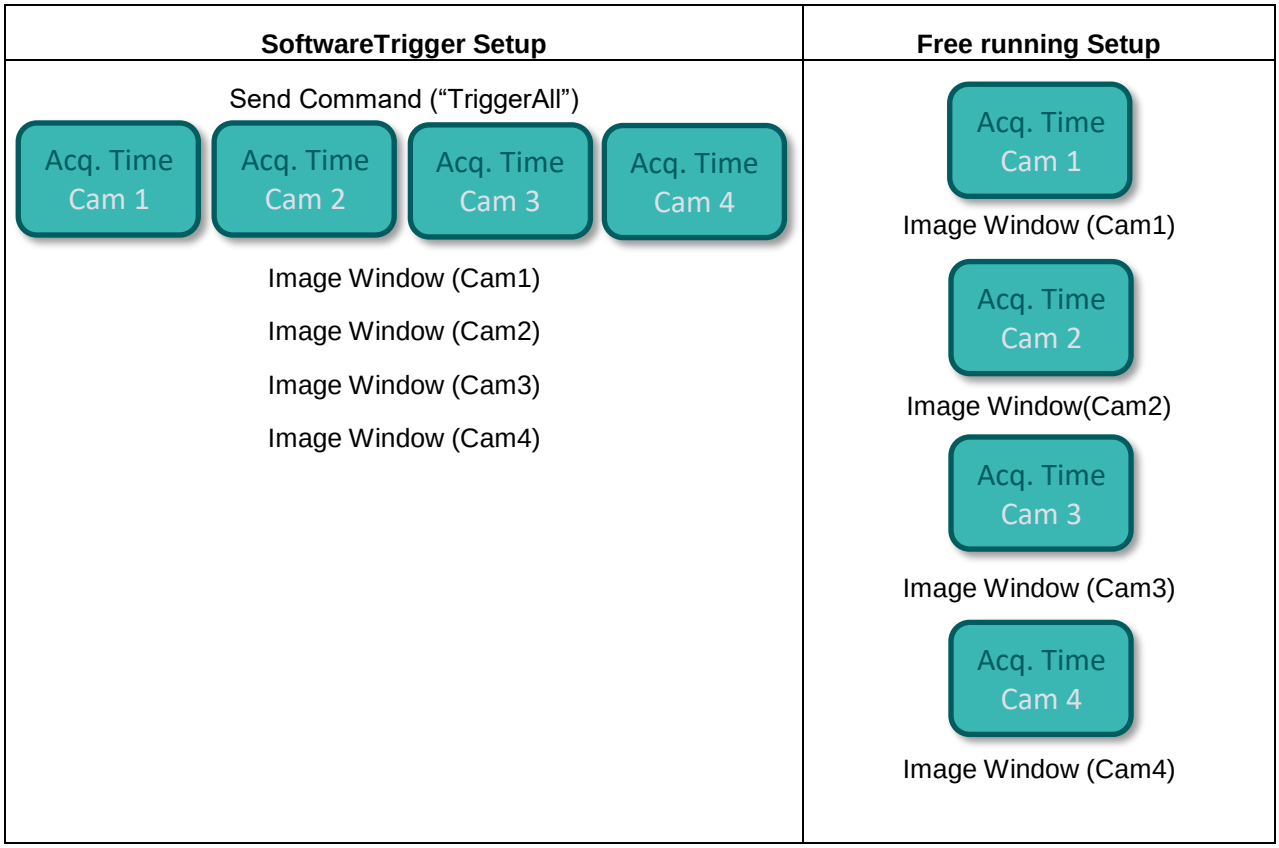
- `::TriggerSoftware`
- `::SoftwareTrigger`
- `::SoftwareTrigger0`
- `::PLC_Q14_Variable0`

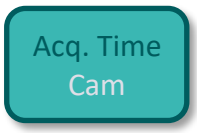
7.1.1.1 Speed up the investigation

To speed up the investigation if more than 1 camera is used, you can sesend a software trigger to all cameras, with which you can trigger all cameras at nearly the same time before the first image window instruction. The string “TriggerAll” can be used for this.

When the software trigger is sent to all cameras, Sherlock can go to the image window instruction and wait for the first image. At this time all cameras have started to acquire their images. When the first image has finished Sherlock makes the processing for this image and goes to the next image window instruction. Now, the time to wait for the next image is shorter because the acquisition has started with the SoftwareTrigger at the beginning. Now Sherlock can start the processing of this image window earlier, which makes the investigation faster.

In the next table you can see a chronological sequence of an investigation with 4 cameras. One with the Software Trigger Setup and one with a Free running Setup. The Acquisition Time rectangle represents the time the camera needs for the acquisition including the exposure time and the time needed for transferring the image from the camera to the PC memory.





Acquisition Time Rectangle

As you can see the time for acquiring several cameras can be much faster with a SoftwareTrigger Setup.

This example is based on a setup with the same exposure time for all cameras. If the exposure time is different the Acquisition Time can be parallel to the Image Window of another camera and its processing.

7.1.2 Acquisition Control

The acquisition can be controlled with string commands in the Send Command Instruction. This is useful if the acquisition has to be stopped and started again within the investigation. For example if you want to change a feature in the camera which can only be changed when the acquisition is stopped you have to stop the acquisition change the feature and then start the acquisition again.

“StartAcquisition”

Starts the acquisition with external trigger / software trigger activated in the used acquisition device. So this command can only be used when software trigger or external trigger is used and external Camera Trigger is selected in the Image window Options.

“StopAcquisition”

Stops the Acquisition.

“StartNonTriggeredAcquisition”

Starts the acquisition and disables the trigger in the used acquisition device. This command can only be used when external Camera Trigger is disabled in the Image window Options.

“StartAllAcquisitions” and **“StopAllAcquisitions”** is also available to start or stop the acquisition of all configured cameras with one command. StartAllAcquisitions calls StartAcquisition for every acquisition device used with the Sherlock CVB Driver in sherlock.

7.1.3 Logging

The Send Command instruction in Sherlock can be used to write a use defined text to the logging of the Sherlock CVB Driver by adding the prefix (LOG::) to the user defined string.

Example:

"LOG::user-defined text"

This writes the text “user-defined text” to the logging of the Sherlock CVB Driver.

7.2 Set Shutter

If available, the Set Shutter instruction in Sherlock tries to use GenICam to set the exposure time of the camera. If GenICam is not available, it sets the exposure time using CVB IDeviceControl with the ID 0x00000104.

On some drivers the ID for setting the exposure time is different to the default ID. Then you have to use the parameter SetShutterIDDeviceControlID in the CVBDrv.ini to change the ID which is used.

To find out the correct ID for your driver, please read IDeviceControl.

7.3 Set Brightness

If available the Set Brightness instruction in Sherlock tries to use GenICam to set the gain of the camera. If GenICam is not available, it sets the gain using CVB IDeviceControl with the ID 0x00000100.

On some drivers the ID for setting the gain is different to the default ID. In this case please use the parameter SetBrightnessIDeviceControlID to change the ID which is used.

To find out the correct ID for your driver, please read IDeviceControl.

7.4 Set / Get Feature

There are 6 Set / Get Feature instructions available in the IO:Camera instructions tree which are all supported by the Sherlock CVB driver:

- Get bool feature
- Get number feature
- Get string feature
- Set bool feature
- Set number feature
- Set string feature

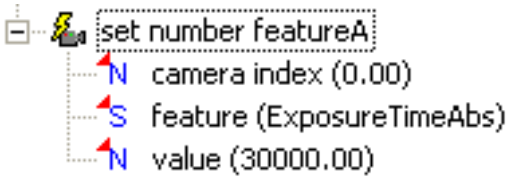
These instructions are used to set and get features of the selected camera. If the selected camera is a GenICam camera it uses the GenICam access.

If GenICam access is not available, these instructions can only be used, if the CVB IDeviceControl interface is supported.

7.4.1 GenICam

To check if your camera has GenICam access, you can open the Driver configuration in the acquisition menu entry (options -> acquisition) and look, if your camera is listed. If the driver configuration button is not available, no camera has GenICam access.

All Set / Get Feature instructions need the camera index and the GenICam feature name as string. To find out the correct feature name, you can use the Driver configuration.



Example: Set Exposuretime to 30ms

7.4.2 CVB IDeviceControl interface

To check if IDeviceControl is available, please have a look at the specific *CVB User Guide* for your image acquisition device. There you can also find a detailed description of the supported commands.

The IDeviceControl interface can be used to access special functionality of the used device. The basic idea is to call a specific driver command by it's command ID and pass arguments or returns for this specific commands into or out of the driver. Each image acquisition device may have it's own list of commands and the parameters used by the commands.

With the Sherlock CVB driver you have to use the hex value of the command ID as the feature string. The hex value for every ID can be found in the driver specific header file which is installed with the driver in the `%CVB%\Lib\C` folder. The name of the header file is documented in the specific CVB User Guide for your image acquisition device. Usually it is called iDC_xxxx.h.

For some vin drivers we added the most relevant enum strings from the header files directly to the Sherlock CVB Driver. Therefor we do not need the hex value from the header file. We can directly use the string which defines the enum value. In the Appendix of this document you can find the most relevant enum values which are available.

The GenICam.vin driver has also iDeviceControl commands. These cannot be used directly because the Set/Get features use the GenICam access by default. If you want to use the access over the IDeviceControl you have to send a prefix in the feature string to inform the driver that you want to use IDeviceControl. The prefix is:

`DC::`

As an example, we want to read out the number of missed packets of the current opened device.

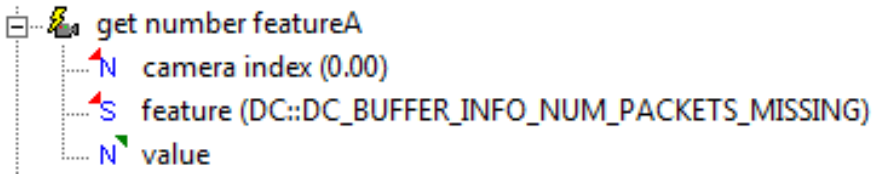
The string from the enum value for the missed packets is:

`DC_BUFFER_INFO_NUM_PACKETS_MISSING`

the string value which we need in the Sherlock instruction get number feature is

`DC::DC_BUFFER_INFO_NUM_PACKETS_MISSING`

The Instruction should look like the following screenshot.



Read out the number of missed packets

Another example is reading out the current frame number of a loaded avi file.

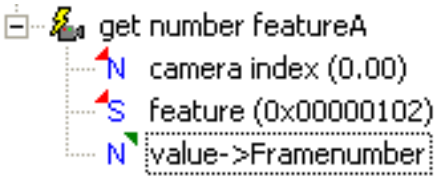
First we have to find out the specific header file. The avi driver is a special driver which is integrated in the CVB Image Manager. So we have no specific CVB User Guide. This is integrated in the Common Vision

Blox Online Help. In the Chapter *Image Manager ->CVB Technology ->Good To Know -> How to deal with Video files* you can read the detailed description of the supported commands. The specific header file is the iDC_AVI.h.

The Command ID for the frame number looks like this in the header file:

DC_FRAMENUMBER = 0x00000102

So the hex value of the Command ID which we need in the Sherlock instruction is 0x00000102. You can copy and paste the whole value to the Instruction which should look like the following screenshot.



Read out the current Framenumber

Advanced Access

With some drivers it is necessary to send a more complex feature string. Examples are the CVB drivers for Teledyne Dalsa and Silicon Software Framegrabbers.

Some of the features need additional information in the feature string. This is separated with the Command ID by a semicolon. If a Set feature function is called the value has to be added to the feature string too.

This is done automatically by the Sherlock CVB Driver when the variable $VALUE is added to the string.

Example for Teledyne Dalsa Framgrabber

In this example a Sapera parameter has to be set over the set number feature. For this, the enum value DC_SAPERA_ACQ_PARAMETER is exchanged by the corresponding hex value which can be found in the iDC_Sapera.h:

```
DC_SAPERA_ACQ_PARAMETER = 0x00000106, // Sapera parameter
```

The Sapera Parameter is CORACQ_PRM_CROP_HEIGHT

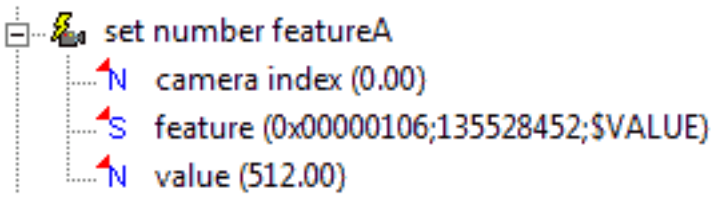
The corresponding integer value for this parameter is 135528452.

The value to be set is 512;

The feature string should look like this:

```
0x00000106;135528452;$VALUE
```

Use 512 as value in the Set Number feature. The second semicolon is needed in the CVB Sapera Drivers. With other drivers this separator may be different or not needed.



Example for Silicon Software Framegrabber

The Silicon Software Framegrabber has a GenICam nodemap by default. With this all the frame grabber parameters can be accessed. It is recommended to use the GenICam access. This is described in the GenICam section of this document.

By default the Set/Get Features use automatically the GenICam access if available. If you really want to use the access over the IDeviceControl you have to send a prefix in the feature string to inform the driver that you want to use IDeviceControl. The prefix is:

DC::

To show you how to do this we want to set the FG_GEN_FREQU parameter to 10.0:

DC_PARAMETER_TYPE_VALUE is exchanged by the corresponding hex value which can be found in the iDC_CVSiSoMe.h:

```
DC_PARAMETER_TYPE_VALUE = 0x00001100,
```

The SiSo Parameter is FG_GEN_FREQU

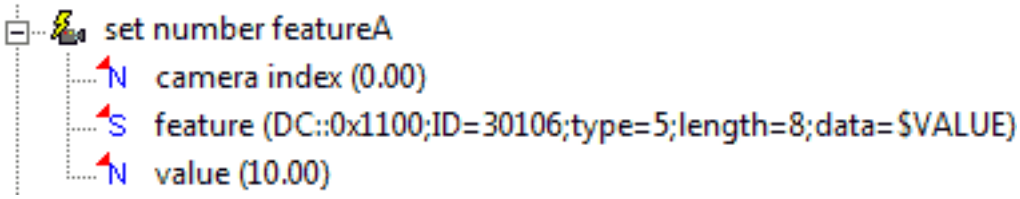
The corresponding integer value for this parameter is 0x1100.

The value to be set is 10.0 and the documentation of the driver tells you that the string has to look like this:

```
"ID=30106;type=5;length=8;data=10.0"
```

For our Set Number feature we need following feature name:

```
DC::0x1100;ID=30106;type=5;length=8;data=$VALUE
```



7.4.3 Get Grab Status information

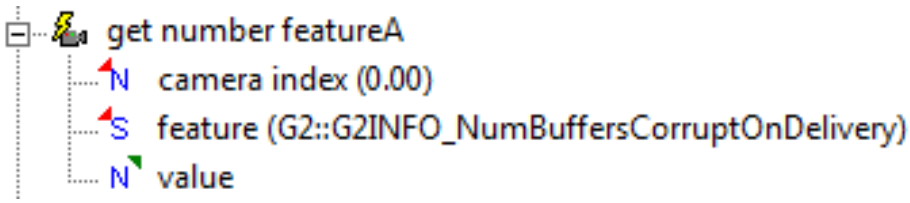
Common Vision Blox supports with its G2Grab Interface the G2GetGrabStatus function with which information regarding the current acquisition can be received.

To get this information in Sherlock over the Sherlock CVB Driver the get number feature need to be used with the prefix “G2::” in the feature name.

To select the information you want to receive you need to specify one of the enum values from GRAB_INFO_CMD which are defined in the `icvcdriver.h` file from Common Vision Blox.

The supported GRAB_INFO_CMD values are listed in the Appendix of this document.

For example you can read out how many received images where corrupt with a GigE Vision camera. For this the enum value ***G2INFO_NumBuffersCorruptOnDelivery*** need to be used which should look like this:



7.5 Get missed frame count

Retrieves the number of missed frames

7.6 Flush acquired images

This instruction is not supported. The acquired images are flushes before a StartAcquisition. So if a flush acquired images is needed and you have a triggered setup you can call a SendCommand with “StopAcquisition” and a SendCommand with “StartAcquisition”. This is documented in the section Acquisition Control.

8 Driver Parameters in Sherlock

8.1 Parameters of the Image Window

8.1.1 Non-triggered/Triggered acquisition timeout

The acquisition timeout parameters override the timeout value from Common Vision Blox. Therefore this parameters can be used as expected.

8.1.2 External Camera Trigger

It is necessary to activate this setting in a triggered setup with an external trigger and a software trigger which is send manually with the Send Command feature.

The “External Camera Trigger” parameter changes the Trigger Mode of the selected camera if it is supported.

Changing the Trigger Mode is supported for GeniCam cameras which comply the Standard Features Naming Convention and all CVB drivers which support the CVB Trigger Interface (ITrigger).

The CVB mode for activating the trigger with ITrigger is CVC_C_TRIGGER_ASYNCRON.

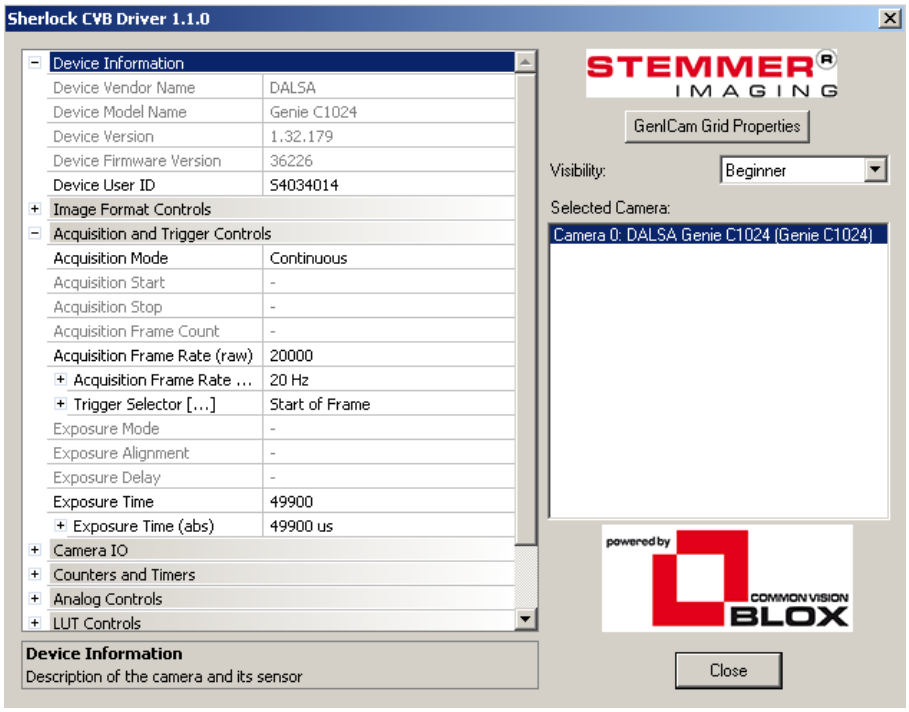
If the mode `CVC_C_TRIGGER_FIELD` is needed the Parameter “TriggerOnField = 1” need to be added to the camera configuration.

8.2 Driver configuration from the Acquisition Menu Entry

When a GenICam Camera is configured in the CVBDrv.ini file the “Driver Configuration” button is available in the acquisition menu entry.

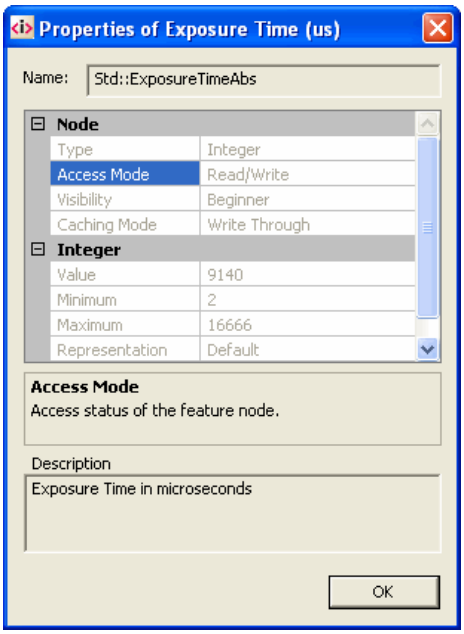
With this dialog it is possible, to change the configuration of the GenICam camera without closing Sherlock. This is also useful, if you need to change features in your investigation and you don’t know the feature name. Then you can use this dialog to look for the correct feature name.

Sherlock CVB Driver Manual



Driver Configuration Dialog

Click on the feature you want to change with the right mouse button. Click on Properties of the opened context menu. Then a new dialog is opened with information about this feature which looks like the image below. The name of the feature is the feature name which has to be used in the Set/Get Feature Instructions.



Properties of Exposure Time

9 Additional Information

9.1 Performance

If a CVB driver is loaded, the Sherlock CVB driver analyzes the image and holds only a copy of the current image, if it is necessary, or it uses optimized methods to copy the image. In combination with the highly optimized CVB drivers, you will get the fastest acquisition you can get with your acquisition device. Since version 2.10.x of the driver it sends automatically a software trigger if an image is requested from the hardware. This reduces the maximum framerate of the acquisition. If the maximum framerate is needed the AutoSoftwareTrigger should be disabled in the CVBDrv.ini file.

9.2 Input /Output Support

I/Os with GenICam cameras are supported if they comply the Standard Features Naming Convention. Then all available I/Os are enumerated and are visible as I/Os in Sherlock.

If GenICam access is not available, the driver uses the DigIO Interface (IBasicDigIO) to enumerate all I/Os and makes them visible in Sherlock.

If more than 1 Camera is configured, the I/Os of all cameras are listed in Sherlock from the first camera to the last camera.

In the case of 3 cameras with 2 Outputs each, the digital output list in Sherlock should look like this:

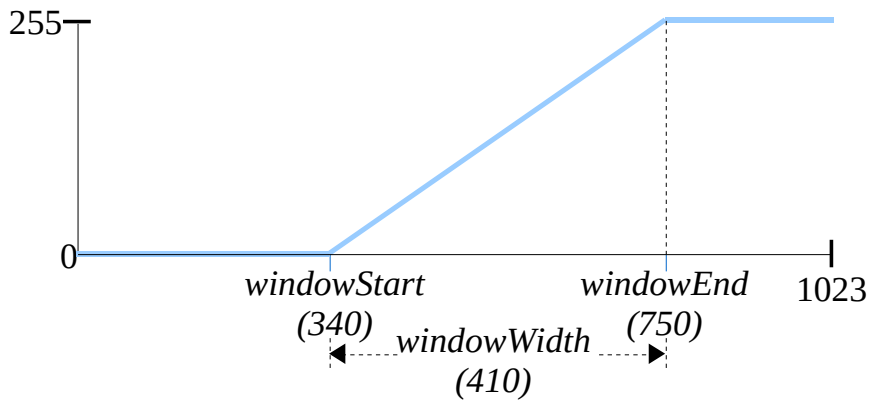
channel 0	Camera 0: Output 0()
channel 1	Camera 0: Output 1()
channel 2	Camera 1: Output 0()
channel 3	Camera 1: Output 1()
channel 4	Camera 2: Output 0()
channel 5	Camera 2: Output 1()

9.3 Support for Images with more than 8Bit

The driver supports images and acquisition devices with more than 8Bit. The driver has native support for linear 16 Bit images. This means that the 16 Bit images are passed to Sherlock directly with no conversion. Other images have to be mapped to 8Bit images which then can be processed in Sherlock with all algorithms. If desired 16 Bit images can also be mapped to 8 Bit.

For this mapping, the windowing technique is used. Therefore, a window in the range of the available gray values is defined. This window is mapped to the 256 gray values of an 8Bit image.

To define the window 2 parameters are needed in the CVBDrv.ini file. One is the start of the window (MapTo8BitWindowStart). The gray value, which is defined as window start, is mapped to the gray value zero. The second parameter can either be the window width (MapTo8BitWindowWidth) or the window end (MapTo8BitWindowEnd). The gray value, which is defined as window end, is mapped to the gray value 255. When the width is used the window end is calculated automatically.



Mapping a 10Bit Image to 8Bit

When the window width is exactly 256 no data will be lost within the window. The grayvalues of the source image are mapped 1:1 to the 8Bit image. With this technique it is possible to process the whole range of a 10Bit image (1024 gray values) with four 8Bit Images in Sherlock.

Example with a 10Bit image:

You have one camera which is configured to output a 10Bit image. You will process the whole range of the 10Bit image in Sherlock.

Then you have to configure the camera 4 times in the CVBDrv.ini with 4 different windows:

```
[Camera0]

Driver = %CVB%\Drivers\GenICam.vin

MapTo8BitWindowStart = 0

MapTo8BitWindowWidth = 256

[Camera1]

Driver = %CVB%\Drivers\GenICam.vin
```

```
MapTo8BitWindowStart = 256
MapTo8BitWindowWidth = 256
[Camera2]
Driver = %CVB%\Drivers\GenICam.vin
MapTo8BitWindowStart = 512
MapTo8BitWindowWidth = 256
[Camera3]
Driver = %CVB%\Drivers\GenICam.vin
MapTo8BitWindowStart = 768
MapTo8BitWindowWidth = 256
```

The driver loads 4 times the same driver with the same 10 Bit image. But the window is different for every configured camera and maps the gray values to four 8Bit images which can be processed in Sherlock with no loss of data. But it is not possible to grab from all 4 Camera configurations simultaneously as this is only one real camera. Because of this you have to call the Instruction Send Command „StopAcquisition“ after every imageWindow with the corresponding camera Index. The Acquisition for the next Camera configuration is startet automatically by sherlock.

9.4 Error Handling

When an error occurs, the error will be visible in the Sherlock Monitor. The error message can have 3 different prefixes:

`**CVB_Sherlock_Error**:`

An error, which is related to the Sherlock CVB driver. E.g. driver not licensed or a problem with the CVBDrv.ini file.

`**CVB_Error<_CVC_ERROR_CODE>**:`

An error, which is related to CVB. This error messages are passed from CVB to Sherlock and are not directly generated by the Sherlock CVB driver. E.g. a CVB_Error is thrown when the configured cameras have a wrong CameraPort or Board number.

If available <_CVC_ERROR_CODE> is exchanged with the CVC_ERROR_CODE defined in the CVCError.h from Common Vision Blox which specifies the error more exactly. In addition an error text is appended to the Error message which describes the error based on the CVC_ERROR_CODE.

`**CVB_GenICam_Error<_CVC_ERROR_CODE>**:`

An error which is related to the GenICam access with CVB.

If available <_CVC_ERROR_CODE> is exchanged with the CVC_ERROR_CODE defined in the CVCError.h from Common Vision Blox which specifies the error more exactly. In addition an error text is appended to the Error message which describes the error based on the CVC_ERROR_CODE.

These kind of errors are returned to Sherlock as general error and if an error occures you can handle it over the standard sherlock mechanism to handle errors. Create a subroutine were you activate the feature "Execute on error". Then you can do the things you need in the subroutine if an error occures. Don´t forget to set the "error handling strategy" to the desired behavior.

9.5 Multiple Sherlock Instances

The driver can be used with multiple Sherlock instances. If only the GenICam.vin driver is used it is not necessary to have an INI-File for each instance. So it is possible to link to the same CVBDrv.ini file from different Sherlock Drivers.ini files. If this approach is used it is necessary to use the AutoSwitchEnable=1 Setting in the GenICam.ini file. AutoSwitchEnable makes it possible to open the needed CamPorts in any order.

If other CVB VIN-Drivers are used, we need separate CVBDrv.ini files for every instance and the different instances need to use different drivers.

If the approach with separate CVBDrv.ini files is used it is still recommended to use the AutoSwitchEnable=1 value in the GenICam.ini file. This makes it possible to open the Sherlock instances in any order. Beyond that the Connection Monitoring (ObserveGenIcamConnection) will still be usable.

But keep in mind that using separate CVBDrv.ini files and using the CVB ACQ For Sherlock License using a MagicNumber in this INI-file makes it necessary that one file exists which has exactly the name “CVBDrv.ini”.

9.6 Silent Setup

The setup of the Sherlock CVB Driver can be executed in silent mode with the Shell-Parameter `/silentex`.

Then no dialog is displayed during installation.

10 Trouble Shooting

10.1 Old Images are acquired with Run Once

Old images are acquired when “Run Once” is pressed. This can happen when the camera is in freerun mode. This section describes why you get old images and how to solve this problem.

10.1.1 General Background

All CVB Drivers use a ring buffer internally to buffer acquired images. The default size depends on the driver. This allows Sherlock to process images delayed when the system is temporarily busy. This is only useful when the camera is triggered externally or the camera is in free run mode and Sherlock gets the time to process all images before the buffer is full. If the buffer is full you will lose frames.

10.1.2 Run Once

Run Once starts the acquisition of the camera in the moment the program gets to the corresponding image window. But the **acquisition is not stopped after run once** is completed. So **the acquisition is still active** and fills the buffer in the background. With the next call to Run Once the oldest image in the buffer is returned. So this is the old image which is acquired by Sherlock.

10.1.3 Solution

To be sure that only one image is acquired when Run Once is clicked you have to use a software trigger.

There are two ways how a software trigger can be used with the Sherlock CVB Driver. The easiest way is to use the Auto Software Trigger feature which can be activated over a parameter in the INI-file. See here for more details.

The other possibility is to use manual software trigger which gives you more flexibility. But this can lead to other problems related to Sherlock described in the next chapter. If you are unsure use the Auto Software Trigger feature. If you know what to do See Chapter Software Trigger for more details of the manual software trigger.

10.2 Old Images are acquired in a triggered setup

If old images are acquired while using Run Once or Run Continuously in a triggered setup read this chapter carefully to avoid this.

If you do not need the flexibility of sending a software trigger manually, you can use the Auto Software Trigger feature of the driver. See here for more details. This avoids that old images are acquired perfectly as for one image which Sherlock wants to acquire one Software Trigger is send to the camera.

10.2.1 General Background

All CVB Drivers use a ring buffer internally to buffer acquired images. The default size depends on the driver.

This allows Sherlock to process images delayed when the system is temporarily busy. This is only useful when the camera is triggered externally or the camera is in free run mode and Sherlock gets the time to process all images before the buffer is full. If the buffer is full you will lose frames.

But Sherlock applications are often synchronized over Software Trigger with the camera. So the buffering of images is not wanted. This would be no problem when the camera is never set to free run. But this is done by Sherlock automatically.

10.2.2 Live image button from image window

Sherlock sets the Trigger Mode automatically depending on the “external camera trigger” setting in the image window options. If live image is activated in the image window the Trigger of the camera is disabled and the camera sends images automatically. At first this is no problem because Sherlock displays the images and the buffer is not filled. But when the live image is disabled after that **no stop acquisition is send to the camera**.

And now the buffer is filled because Sherlock doesn’t display the new images any more. If run continuously or run once is executed and “external camera trigger” is activated the trigger mode of the camera is set to on. When you send a Software Trigger to your camera before the “image window”, a new image is acquired as expected and saved to the buffer. But there are still images in the buffer from the live image before. And then Sherlock gets the oldest image from the buffer.

10.2.3 Run Once

Run Once starts the acquisition of the camera in that moment when the program gets to the corresponding image window. But the **acquisition is not stopped after run once** is completed. **So the acquisition is still active**. In that case you can come to a situation where images are acquired from the camera to the ring buffer but Sherlock doesn´t acquire the new images. When you send a Software Trigger to your camera

before the “image window” a new image is acquired as expected and saved to the buffer. But there are still images in the buffer from the live image before. And then Sherlock gets the oldest image from the buffer.

10.2.4 Solution

To be sure that no image is in the buffer when continuous investigation is started you have to free the buffer.

This is done when the acquisition is stopped. To stop an acquisition you can use the “StopAcquisition” with the Send Command instruction.

As this is only needed at the beginning of a continuous investigation you can create a subroutine where the “Execute before continuous investigation” checkbox is activated. There you have to add a Send Command with the string “StopAcquisition”. To prepare the camera for the first trigger it is recommended to start the acquisition also in this subroutine with Send Command and the string “StartAcquisition”.

With this subroutine and a software triggered camera you can be sure that you get the current image from the camera.

If this is too complicated for your needs please use the Auto Software Trigger feature which can be activated over a parameter in the INI-file. See here for more details.

11 Additional Documentation

Additional Documentation can be found in the CVB Getting Start Guide, the CVB User Guide for the used image acquisition device (camera or frame grabber) or the Sherlock 7 Help.

12 Appendix

12.1 GRAB_INFO_CMD values

These strings can be used with the get number feature as feature name with the prefix “G2::”

GRAB_INFO_NUMBER_IMAGES_AQCUIRED

Inquire the number of acquired images since the last call to G2Grab. This number represents the number of images that have been transferred by the camera into the image acquisition device. It includes images that have been processed as well as images that have been lost for certain reasons.

GRAB_INFO_NUMBER_IMAGES_LOST

Inquire the number of lost images due to image acquisition device performance since the last call to G2Grab.

GRAB_INFO_NUMBER_IMAGES_LOST_LOCKED

Inquire the number of lost images due to the fact that all buffers were locked since the last call of G2Grab.

GRAB_INFO_NUMBER_IMAGES_LOCKED

Inquire the number of actually locked buffers.

GRAB_INFO_NUMBER_IMAGES_PENDIG

Inquire the number of pending buffers (buffers which contain new data but have not been delivered with G2Wait). This basically defines a queue and contains the number of buffers in the ring buffer - 1.

GRAB_INFO_GRAB_ACTIVE

Inquire if a grab (acquisition) is active (`0` indicates that no grab is active, all other values indicate an active grab).

GRAB_INFO_TIMESTAMP

Inquire the timestamp for the current image. This feature is optional and may not be supported by any driver.

GRAB_INFO_NUMBER_TRIGGERS_LOST

Inquire how many trigger signals are lost/ignored while acquisition was active (trigger frequency is too high).

Attention:

At the moment only Dalsa grabbers can count how many triggers are lost.

GRAB_INFO_CAMERA_DETECTED

Inquire if a camera was detected or not.

Attention:

Only SiSo and Dalsa grabbers support this feature. A grabber detects the pixel clock of a camera and inform the application if a camera has a pixel clocks or not. This feature is not supported with GigE or fire wire.

G2INFO_NumBuffersPending

Inquire the number of images/buffers in the output queue, waiting for a call to G2Wait.

Attention:

These images/buffers contain new data not yet seen by the application. The application must call G2Wait to receive the images. It is depending on the configuration of the driver if "not yet delivered, incomplete frames" are counted as pending or not. If the driver is configured to "Deliver corrupt frames" then then these frames are counted as being "Pending". If the driver is configured to discard corrupt frames then these frames are not counted as being pending.

G2INFO_NumBuffersDelivered

Inquire the number of images delivered through G2Wait no matter if these images were corrupt or not.

G2INFO_NumBuffersCorrupt

Inquire the number of images which were corrupt on arrival in the host no matter if or if not resend requests have been issued. This counter indicates if something is wrong with the transmission. In case the user wants to discard corrupt images he needs to configure his vin/tl accordingly. A corrupt image is an image which in the first run did not arrive properly. This includes also images which for example had missing packets and which have been completed later on using resend requests. So a corrupt image may not be corrupt on delivery because the transport mechanisms underneath corrected the failure.

G2INFO_NumBuffersCorruptOnDelivery

Inquire the number of images which are corrupt (not complete) when delivered through G2Wait. A corrupt delivered image cannot/will not be completed after delivery.

G2INFO_NumBuffersLost

Inquire the number of images lost locked + number of images lost because of other reasons For example the image was corrupt and the setting of the driver did not allow delivery of corrupt images.

G2INFO_NumBuffersLostLocked

Inquire the number of lost images due to an overrun in the acquisition because G2Wait was not called fast enough.

G2INFO_NumBuffersAcquired

Inquire the nNumImagesDelivered + NumImagesPending It is depending on the configuration of the driver if "not yet delivered incomplete frames" are counted as pending and therefore be part of NumImagesAcquired or not. See the explanation of "NumPendingImages".

G2INFO_NumBuffersQueued

Inquire the nNumber of buffers which, at a given point in time, can take new data. Only queued buffers can take new image data. Buffers which are not queued are basically locked. In CVB, before a G2Grab starts all announced buffers are also queued for acquisition. So until the first image starts the NumBuffersQueued = NumBuffersAnnounced. While the first image is transferred the number of queued images is one less than at the start. After the first image finished until the first call to G2Wait the NumBuffersQueued = NumBuffersAnnounced-1 and the NumBuffersPending = 1. After G2Wait and before the second image starts NumBuffersQueued = NumBuffersAnnounced-1, NumBufferDelivered=1, NumBufferPending=0. And so on.

G2INFO_NumBuffersAnnounced

Inquire the number of buffers which can hold image data. This has direct influence on the amount of memory allocated by the driver to hold image data. This value is equivalent to the number of buffers in the iRingBuffer interface.

G2INFO_NumBuffersInIncompleteList

Inquire the number of buffers which have been started to fill but are not yet done. In case this (internal) incomplete list is full the buffer will, depending on the configuration of the driver, be delivered as corrupt buffer to the host application or discarded.

G2INFO_NumPacketsReceived

This is related to GigE and indicates the number of data packets received on the GigE Vision streaming channel.

G2INFO_NumResends

This is related to GigE and indicates the number of resend requests issued by the host. If this number increases it indicates that something is wrong with the data transmission. Possible reasons may be: Settings of the network card, CPU-load on the host, cable/switch problems, bandwidth problems on the Link.

G2INFO_NumBuffersLocked

Inquire the number of buffers locked at a given point in time.

12.2 DC_PARAM values

These strings can be used with the get number feature as feature name with the prefix “DC::” to get information from the driver.

Only the widely used strings are listed here. The complete list can be found in the specific iDC_*.h header file. For example, the iDC_GenICam.h.

DC_BUFFER_INFO_TIMESTAMP

This is the timestamp received from the acquisition hardware saved to the acquired buffer in the memory. This string is supported by the GenICam.vin and the CVSisoMe.vin driver

DC_BUFFER_INFO_IMAGEID

This is the image id received from the acquisition hardware saved to the acquired buffer in the memory. This string is supported by the GenICam.vin and the CVSisoMe.vin driver

DC_BUFFER_INFO_NUM_PACKETS_MISSING

This is the number of missing packets from the image transfer of a GigE Vision camera related to the acquired image of the currently used buffer.

12.3 Changed GenICam Features

This chapter lists all GenICam features, which can be changed by the driver automatically depending on settings in the INI-file or settings made in Sherlock.

If the Namespace is not listed as prefix the driver is ignoring the namespace.

Std::TriggerMode/ SequencerMode / ExposureMode / GrbCh0TrigCfgPLCTriggerable

If available, these features are changed depending on the external camera trigger setting in Sherlock.

The initial values are saved when Sherlock starts and changed back to the initial values when Sherlock is closed.

Std::TriggerSource

The Feature „Std::TriggerSouce“ is changed to „Software“ if AutoSoftwareTrigger is activated or a Software Trigger is send over a Sherlock SendCommand call.

The initial value is saved when Sherlock starts and changed back to the initial value when Sherlock is closed.

TriggerSoftware / SoftwareTrigger / SoftwareTrigger0 / PLC_Q14_Variable0 / StartPulse

One of these features is used if available to send the software trigger to the camera.

ExposureTime / ExposureTimeAbs/ ExposureTimeRaw

One of these features is used if available to change the Exposure time if the SetShutter instruction is called within Sherlock.

Gain / GainAbs / GainRaw

One of these features is used if available to change the gain if the SetBrightness instruction is called within Sherlock.

Std::UserOutputValue / OutputLineValue / SyncOutLevels / outputLineValue

One of these features is used if available to change the Outputs

13 Contact and Feedback

We hope that this manual was useful for you and look forward to your feedback. In case of further questions, please do not hesitate to contact our technical support.

Additional information and a lot of valuable details regarding image processing, can also be found on our website.

DENMARK

Phone: +45 33 73 00 10
E-Mail: support@stemmer-imaging.dk
Web: www.stemmer-imaging.dk

FINLAND
BALTICS

Phone: +358 9 43 555 012
E-Mail: support@stemmer-imaging.fi
Web: www.stemmer-imaging.fi

FRANCE

Phone: +33 1 45069560
E-Mail: support@stemmer-imaging.fr
Web: www.stemmer-imaging.fr

GERMANY
AUSTRIA

Phone: +49 89 80902-200
E-Mail: support@stemmer-imaging.de
Web: www.stemmer-imaging.de

POLAND

Phone: +46 8 555 110 27
E-Mail: support@stemmer-imaging.pl
Web: www.stemmer-imaging.pl

SWEDEN
NORWAY · ICELAND

Phone: +46 8 555 110 27
E-Mail: support@stemmer-imaging.se
Web: www.stemmer-imaging.se

SWITZERLAND
LIECHTENSTEIN

Phone: +41 55 415 90 90
E-Mail: support@stemmer-imaging.ch
Web: www.stemmer-imaging.ch

THE NETHERLANDS
BELGIUM · LUXEMBOURG

Phone: +31 575 798881
E-Mail: support@stemmer-imaging.nl
Web: www.stemmer-imaging.nl

UNITED KINGDOM
IRELAND

Phone: +44 1252 780060
E-Mail: support@stemmer-imaging.co.uk
Web: www.stemmer-imaging.co.uk