Glaz LineScan-I

High-speed linear array cameras

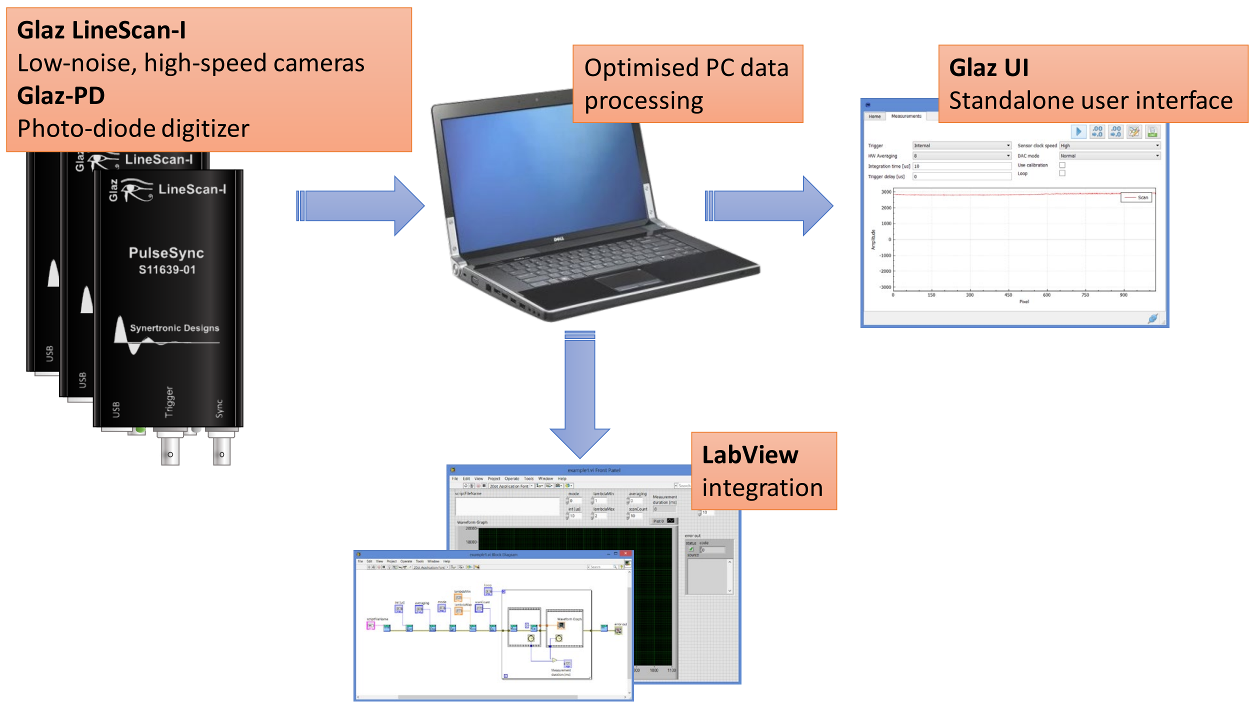
- Low-noise
- 16-bit resolution
- High-speed USB interface
- USB-powered
- LabView integration
- Flexible XML-based configurations
- Application-specific firmware flavours

Overview

Glaz LineScan-I is a low-noise camera with 16-bit resolution. Extremely low noise-levels can be obtained by a selection of hardware and/or software averaging modes. Combined with powerful PC-based data processing software, a multi-camera measuring system can be configured quickly and effortlessly.

Glaz LineScan-I cameras are available with two different firmware flavours: *PulseSync* and *TimeFill*. *PulseSync* is recommended for synchronised single- and multi-camera systems. *TimeFill* is recommended for applications requiring large temporal fill factors.

Glaz UI (a stand-alone application) or LabView can be used to configure and communicate with the cameras. Measurements with single- or multi-camera configurations can be configured, using XML-based script files. This provides a flexible way to define calculations, that need to be performed on the measured data. All data processing is performed in optimised, multi-threaded libraries, thereby providing the best possible performance.



Specifications

Characteristic	Value	Unit
PC Interface		
PC interface	USB 2.0 (high-speed)	
Data transfer rate[1]	>20	MB/s
Maximum USB cable length	3	m
Linear sensor		
Sensor type	CMOS	
Supported sensors	Hamamatsu S12198-1024Q Hamamatsu S10453-1024Q Hamamatsu S11639 Hamamatsu S11639-01	
Optical integration time	PulseSync: 2 – 5,000 TimeFill: 2 – 400,000	µs
Hamamatsu S12198-1024Q/S10453-1024Q		
Pixel count	1024	
Scan rate (half-speed clock)	3,500	lines/s
Scan rate (full-speed clock)	7,000	lines/s
Spectral response range	200 – 1000	nm
Noise level (single scan)	<520	µOD[2]
Noise level (512 averaged scans)	<25	µOD
Dynamic range[3] (single scan)	>1,900	
Dynamic range (512 averaged scans)	>40,000	
Hamamatsu S11639/S11639-01		

[1] Using dedicated USB 2.0 port.

[2] OD is the relative amplitude with respect to full-scale value.

[3] Dynamic range is defined as the ratio of the full-scale value and the RMS noise value.

Characteristic	Value	Unit
Pixel count	2048	
Scan rate (half-speed clock)	1,700	lines /s
Scan rate (full-speed clock)	3,500	lines /s
Spectral response range	200 – 1000	nm
Noise level (single scan)	<530	μOD
Noise level (512 averaged scans)	<30	μOD
Dynamic range (single scan)	>1,800	
Dynamic range (512 averaged scans)	>33,000	
Digitizer		
Resolution	16	bit
Full-scale reading	65535	
ENOB (effective number of bits)	13.5	bit
Drift	100	ppm/^{O}C
Total analogue front-end noise level	<1	mV_{RMS}
External trigger		
Signal type	5V-TTL	
Trigger edge	rising	
Positive-going threshold voltage	2.8	V
Negative-going threshold voltage	2.0	V
Minimum trigger pulse width	1	μs
Programmable trigger delay	0 - 200	ms
Programmable trigger delay resolution	0.1	μs

Table 1 Specifications for *LineScan-I*.

Hardware description

Dimensions

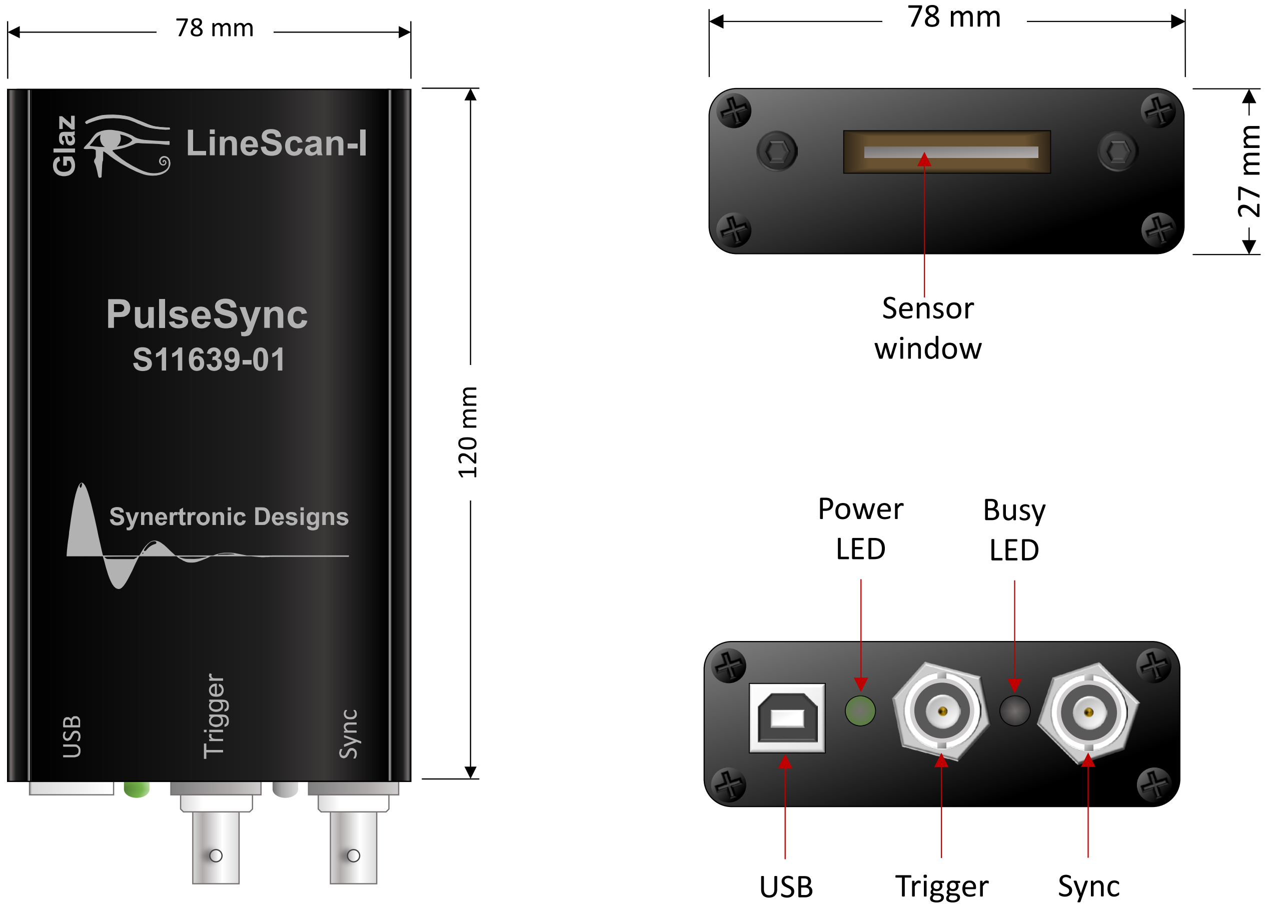


Figure 1 Mechanical dimensions and legend.

Ports

Port	Type	Function
USB	USB-B	Data connection to a PC. Also provides the camera with power.
Trigger	BNC	External trigger input.
Sync	BNC	Synchronisation signal for multi-camera operation.

Table 2 Connectors.

LEDs

LED	States	
Power LED	off	no power
	green	camera has power.
Busy LED	off	idle
	green	waiting for trigger
	green to red	triggered and busy scanning

Table 3 LEDs.

The colour of the *Busy* LED is an indication of how busy the cameras is. When the camera is not busy (i.e. not triggered) the LED will be green. When the camera is working close to its maximum scan rate, the LED will be red. For lower scan rates, the camera will be yellow.

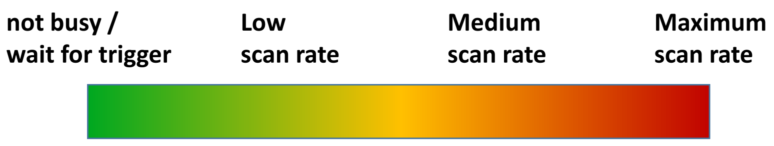


Figure 2 Ready/Busy LED colour legend.

Functional description

Connecting cameras to a PC

Cameras are connected to a PC via USB interfaces. *Glaz LineScan-I* cameras are USB 2.0 high-speed devices, but are backwards compatible with USB 1.1 interfaces. USB 1.1 interfaces have a maximum data transfer rate to less than 0.5 MB/s and will limit maximum scan rate of the camera.

If a PC has a limited number of USB ports, it is possible to use an external USB hub. Up to two cameras can be connected to a single USB 2.0 hub without significant reduction in the data transfer rate. It is possible to connect more than two cameras to a single USB 2.0 hub, but this can reduce the maximum data transfer rate and scan rate. When connecting more than two cameras to a single USB hub, it is advisable to use a USB 3.0 hub.

Camera clock speed

The internal sensor clock of the camera can be operated at two frequencies: half-speed and full-speed. The sensor noise level at full-speed is approximately 20% higher compared to the noise level at half-speed. By default, the camera uses the full-speed clock. When low-noise measurements are a priority, it is advisable to use the half-speed clock.

Clock speed	Internally sensor pixel clock
half-speed	10 MHz
full-speed	5 MHz

Table 4 Sensor pixel clock.

Triggering cameras

Cameras can be triggered either internally or externally. When running in internal trigger mode, no external trigger should be connected to the camera. In this mode, the trigger signal is generated internally by the camera itself. The frequency of the internal trigger signal depends on the firmware flavour. See "Internal trigger generator" for more information.

When running in external trigger mode, each scan is started after an external trigger pulse is applied to the trigger port. A delay can be specified between the positive going edge of the trigger pulse and the start of the sensor integration time. The delay can be specified with a resolution of 0.1 μs and can be set between 0 and 200 ms. When a delayed trigger[4] is received while the camera is busy, this trigger will be ignored.

Integration time

[4] The delayed trigger is the external trigger signal with the specified delay.

During the integration time, the photo-current from the CMOS sensors will be integrated. The integration time can be set between 2 and 2000 µs. For short-pulse laser applications, the preferred integration time is 5 µs.

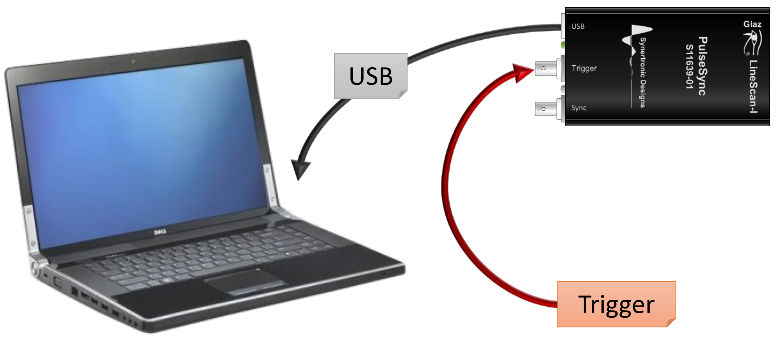


Figure 3 Single-camera configuration.

Single-camera configuration

In a single-camera configuration only a single camera is used. When external trigger mode is used, the camera must be provided with a trigger signal via the *Trigger* port.

⚠ In internal trigger mode, leave the *Trigger* port unconnected. The *Sync* port must always be left unconnected.

Hardware and software averaging

In order to reduce noise levels, averaging must be applied. *Glaz LineScan-I* provides two ways to average scans: hardware and software averaging.

Hardware averaging is performed on the camera. The number of scans to average can be set to: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048 or 4096. The camera will average the raw sensor data for the given number of scans. Only the final averaged result is sent to the PC. This is a useful technique to reduce the required USB bandwidth and processing speed of the PC.

Software averaging is performed on the PC. The difference compared to hardware averaging, is that the calculation result of the scanned data (i.e. processed data) is averaged. Hardware and software averaging can be combined to reduce noise and the required bandwidth and processing speed of the PC.

Hardware averaging ($N_{a,HW}$):
Scans are averaged by hardware on the camera.
Settings: 1, 2, 4, 8, ... , 4096

Software averaging ($N_{a,SW}$):
Averaged scans are processed by the PC. Processed results can again be averaged.

USB:
After scans are averaged, a single averaged nett scan is sent to the PC.

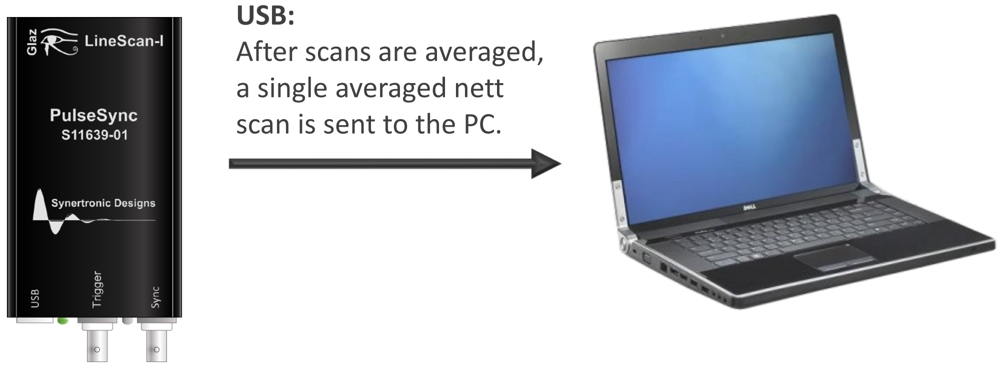


Figure 4 Hardware and software averaging.

Software pixel binning

In the latest software, binning can be enabled. Binning calculates the average of 2 or 4 adjacent. This is not a moving average. Pixels are grouped into adjacent sets of 2 or 4 pixels and the average of each groups is calculated. As a result, binning reduces the effective number of pixels of the linear array:

Binning	Pixel averaging groups	Effective number of pixels
2-pixel	[0,1], [2,3], [3,4], ...	½ $N_{p,sensor}$
4-pixel	[0,1,2,3], [4,5,6,7], [8,9,10,11], ...	¼ $N_{p,sensor}$

Table 5 Pixel binning.

Noise

There are several ways to define the dynamic range of a camera. In this manual, the dynamic range D is defined as the ratio of the sensor full-scale A_{FS} value and the noise RMS value A_{RMS}:

$$D = \frac{A_{FS}}{A_{RMS}}$$

In order to decrease the noise level and increase the dynamic range, averaging can be used. The RMS noise level of an averaged signal is reduced by the square root of the number of averaged scans:

$$A_{avg,RMS} = \frac{A_{RMS}}{\sqrt{N_a}}$$

, where N_a is the number of averaged scans.

Pixel binning further reduces the noise level:

$$A_{avg,binning,RMS} = \frac{A_{avg,RMS}}{\sqrt{N_b}}$$

, where N_b is the level if binning (e.g. 2 or 4).

Drift

Drift of the camera sensor becomes visible, when performing low-noise measurements with a large number of averaged scans. When averaging more than 256 scans, the noise level drops below the camera drift. Drift is mainly cause by temperature changes of the camera sensor. This is more pronounced during the first 15 minutes after the camera is power up. In Figure 5 the drift of the average sensor reading as a function of time is given. The drift is expressed relative to the final average sensor reading after 30 minutes. A change of up to 2% can be expected during the first 15 minutes after power-up.

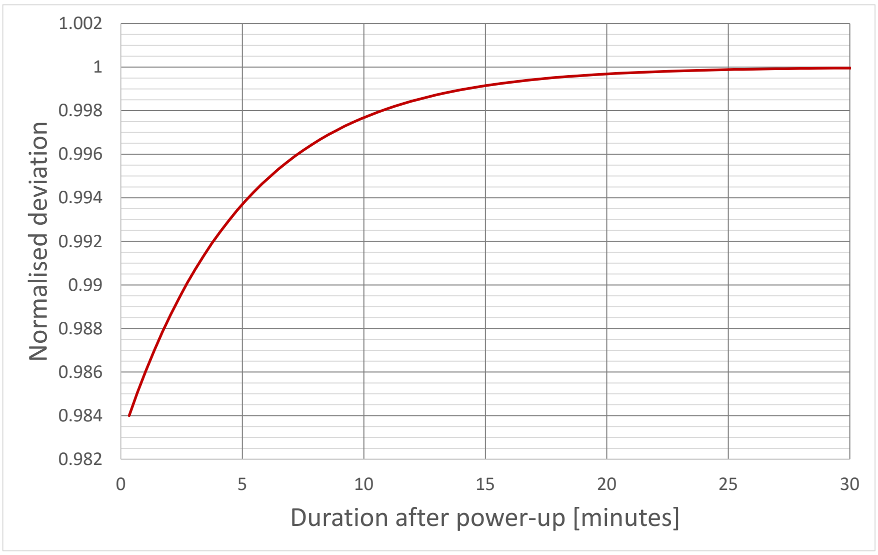


Figure 5 Drift after power-up.

In order to reduce the effect of drift, the camera should be given at least 15 minutes to warm up. In addition, the camera sensor should be shielded from air drafts and room temperatures should be kept constant. It is possible to compensate for drift by performing a background measurement in regular intervals and subtracting this background from measurements.

Photoresponse nonuniformity (PRNU)

The pixels on a linear optical array are not identical. Each pixel has a different offset and gain. *Glaz UI* supports offset calibration per pixel. Gain calibration is also supported, but is not recommended. Calibrations are stored on the camera. A typical PRNU is shown in Figure 6.

Several averaged scans (with 256 hardware averaging) were taken and plotted. Only the data from the first 128 pixels is shown. Note: the resulting dark red trace is not noise, but the sensor PRNU. The noise level for each pixel is below the PRNU and the differences in gain and offset of each pixel is clearly visible.

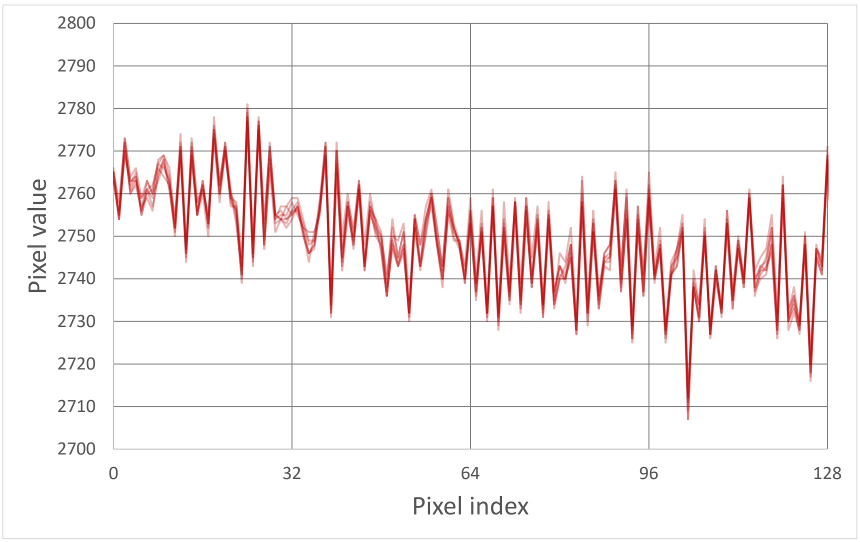


Figure 6 Example of PRNU.

PulseSync and TimeFill

Instead of providing a general solution, the *Glaz LineScan-I* cameras are pre-programmed with application-specific firmware. The most notable differences between different firmware flavours are:

- Sensor read-out scheme
- *Sync* port functions
- Internal trigger generator

LineScan-II cameras provide more flexibility. They can be configured at run-time with either the *PulseSync* or *TimeFill* mode.

Sensor read-out schemes

The sensor integration period and the data read-out can either be interleaved or non-interleaved.

PulseSync – read-out scheme

PulseSync uses the non-interleaved read-out scheme. One integration and read-out cycle must be fully completed, before the next cycle can start. This is especially important for multi-camera systems to ensure reliable synchronisation.

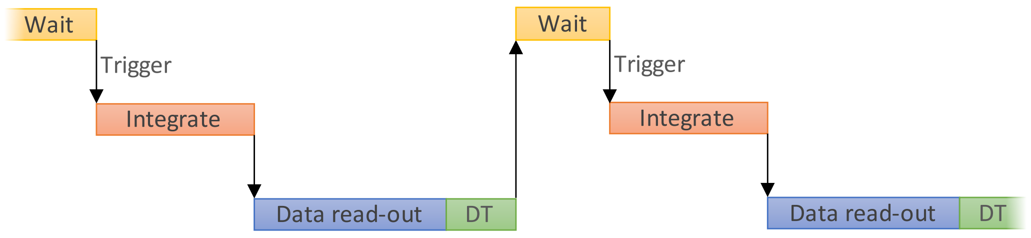


Figure 7 *PuleSync*: Non-interleaved read-out scheme

TimeFill – read-out scheme

TimeFill uses the interleaved read-out scheme. The next integration cycle can start even before the previous data read-out is completed. With this scheme, very high temporal fill factors can be achieved, but cannot be used for robust multi-camera synchronisation.

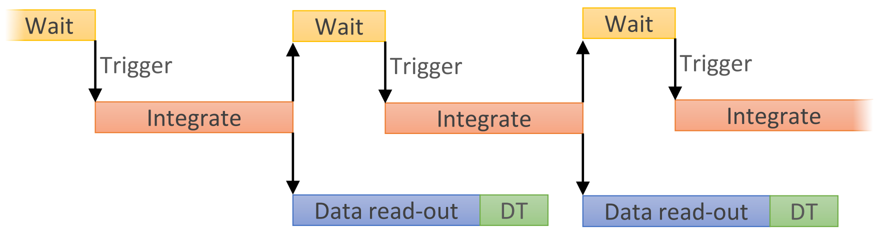


Figure 8 *TimeFill*: Interleaved read-out scheme

Sync port functions

The *Sync* port is an open-collector with internal 1.8 kΩ pull-up. The polarity is active low.

TimeFill – *Sync* port functions

The following *Sync* functions are supported:

- Trigger:
  The *Sync* port is pulled low for 20 µs after each internal/external trigger pulse.
- Integration window:
  The *Sync* port is pulled low during the integration period.

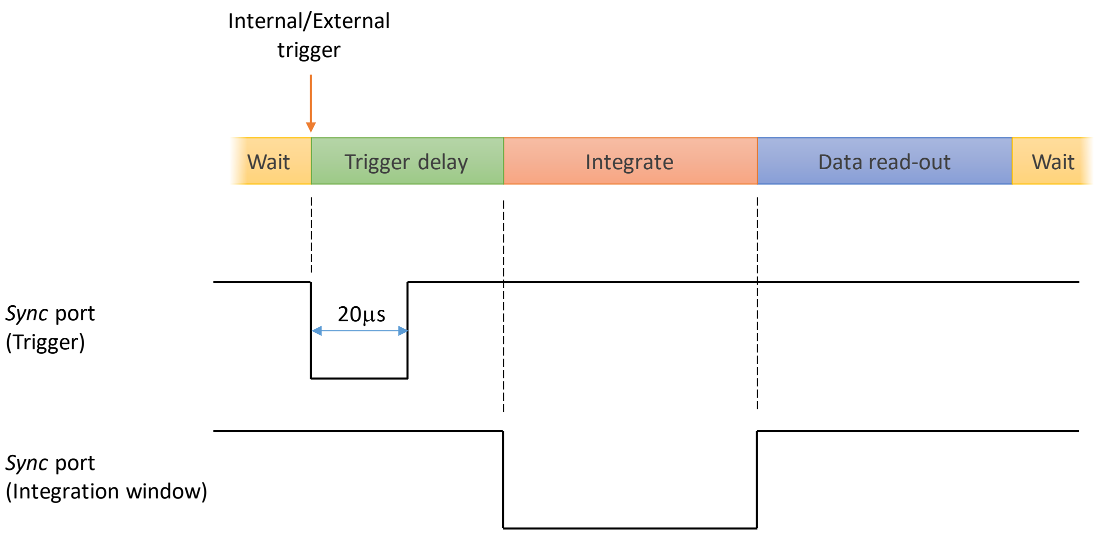


Figure 9 *TimeFill Sync* port functions.

TimeFill cannot be used for multi-camera synchronisation or combined *Glaz LineScan* and *Glaz-PD* measurements.

PulseSync – *Sync* port functions

With *PulseSync* the *Sync* port is exclusively used for synchronisation between *Glaz LineScan* cameras and *Glaz-PD* devices. For more information see "*PulseSync*: Multi-camera configuration and synchronisation" and "*PulseSync*: Combining Glaz LineScan-I and Glaz-PD devices".

Internal trigger generator

Cameras can be configured to run from an internal trigger. In this case, no external trigger signal is required.

TimeFill – internal trigger generator

With *TimeFill* the camera supports a free-running internal trigger generator. The trigger delay is also applied to the internal trigger. The trigger frequency ranges are:

Description	Sensor	Value	Unit
Min. frequency	All	2.4	Hz
Max. frequency	S11639 / S11639-01 S10453-1024Q / S12198-1024Q	4000 8000	Hz

Table 6 Internal trigger frequency ranges.

PulseSync – internal trigger generator

With *PulseSync* the internal trigger generator starts a line scan. Then it waits until the measurement is completed, before initiating the next line scan. A fixed trigger frequency is not supported.

Measurements and scripting

Script file structure

The calculations that need to be performed by the PC with the measured data is defined by XML script files. A typical script file is shown in Figure 1.

```
<!DOCTYPE GlazScript>
<config>
  <camera serial="SYBP005010001" number="1" master="1"/>
  <preprocessor camera="1" type="calibrate"/>
  <preprocessor camera="1" type="subtract_background"/>
  <calculation name="Camera 1" keepscans="1">
    <measurement camera="1"/>
  </calculation>
</config>
```

Camera definitions

Pre-processing steps

Calculations

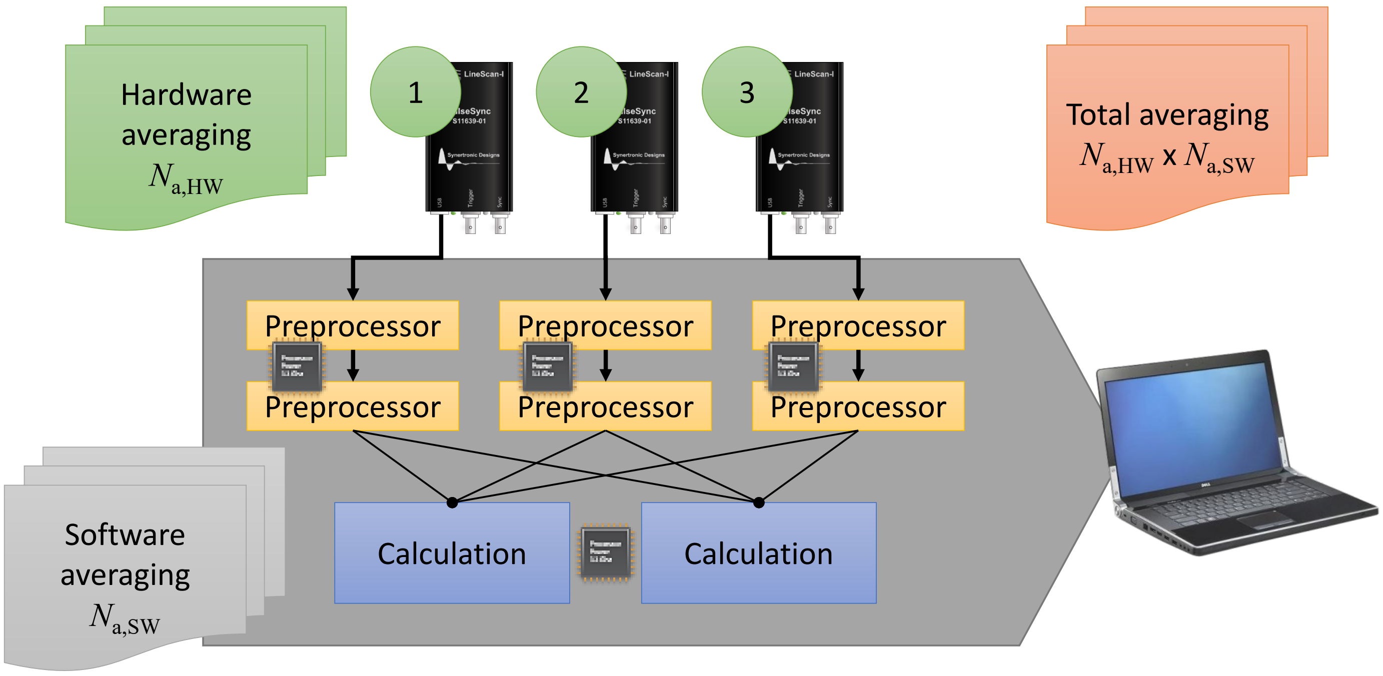
Figure 10 XML script file.

A script file consists of three sections:

1. Camera definitions:
   Each camera used for the measurement must be listed here.
2. Pre-processing steps:
   The pre-processing steps for each camera listed in the camera definitions must be specified.
3. Calculations:
   Any number of calculations can be specified. Each calculation can use the pre-processed results of one or more cameras.

Measurement data flow

Figure 2 depicts a multi-camera configuration with three cameras, two pre-processing steps for each camera and two calculations. Hardware averaged scans are sent from each camera to the PC via USB. The scans are then passed through the pre-processing steps. Data reception and pre-processing are performed on separate threads for each camera. On a quad-core PC, this would imply that the data from each camera is pre-processed on a separate core.



Hardware averaging $N_{a,HW}$
1
2
3
Total averaging $N_{a,HW} \times N_{a,SW}$
Preprocessor
Preprocessor
Preprocessor
Preprocessor
Preprocessor
Preprocessor
Calculation
Calculation
Software averaging $N_{a,SW}$

Figure 11 Measurement data flow.

Calculations are performed only when the pre-processed data from all the cameras for a given hardware averaged scan is available. The process of receiving averaged scans and performing pre-processing and calculations is performed $N_{a,SW}$ times. The results of the calculations are accumulated and at the end of the measurement the software average for each calculation is determined.

Scripting – Camera definitions

In order to use cameras, the must first be uniquely identified. Each camera has an entry in the script file using the following format[5]:

```
    <camera serial="SN" number="NUM" [master="M"] [reverse="R"]
[binning="B"]/>
```

SN : The serial number of a camera is printed onto the back of the camera. The serial number has 13 digits and has the format: SYPB + device number + device version + instance number. For example: SYPB005010001. A serial number may only be defined once in a script file.

NUM : The camera number that will be assigned to the camera with the given serial number. This number will be used in the rest of the script file. This must be a unique number in the range [1 .. 1000].

M : Can be one of the following values: *0*, *1*, *true* or *false*. If the value is *1* or *true* the camera will be configured as the master camera. The master camera is the camera that must receive the trigger signal. Only one camera may be defined as the master. The default value is *0* or *false*.

R : Can be one of the following values: *0*, *1*, *true* or *false*. If the value is *1* or *true* the scan will be reversed and the data from pixels 0 to 1023 will be processed as pixels 1023 down to 0. The default value is *0* or *false*.

[5] Parameters in square brackets [...] are optional.

B : This value specifies the level of binning. It can be one of the following values:

0	no binning (default)
1	2-pixel binning
2	4-pixel binning

Scripting – Pre-processing steps

Each camera in the camera definition section may have its own set of pre-processing steps. The pre-processing steps are executed in the same order as defined in the script file. It is important to specify the pre-processing steps in the correct order. The following pre-processing steps are supported:

- Calibrate:
  Calibrates the scan data using the calibration stored on the corresponding camera.
- Inverse Fourier transform (IFFT):
  Calculates the inverse transform of a scan. It is assumed that a scan is performed with constant wavelength increments. The wavelength for each pixel is given by:

  $$\lambda_n = \lambda_{min} + \frac{\lambda_{min}\text{-}\lambda_{max}}{N\text{-}1} n \qquad \text{with} \qquad n = 0\,..\,1023,$$

  where n is the pixel number and λ_{min} and λ_{max} are the wavelength limits. The wavelength limits are specified outside the script file. In LabView a VI is provided to set the limits and in the stand-alone Glaz UI the limits can be specified on the user interface. The IFFT pre-processor will interpolate the scan in frequency space with constant increments. The IFFT is performed on the interpolated data.

  It is possible to turn interpolation off, by setting the `interpolate` attribute to *false*. In this case, the IFFT is performed directly on the scan data.
- Background subtract:
  Before performing a measurement the background must be measured. In LabView a VI is provided and in the stand-alone Glaz UI an action can be triggered to take a background measurement. Note: The stored background will be the resulting data after pre-processing. The background subtraction must, therefore, always be the last pre-processing step.

A pre-processing step entry in the script file has the following format:

```
<preprocessor camera="NUM" type="T" [interpolate="I"]/>
```

NUM : The camera number that was assigned to a camera in the camera definition section. This pre-processing step will be applied to the scan data from the camera with the given number.

T : Can be one of the following values: *calibrate*, *ifft* or *background_subtract.*

I : This attribute is only applicable to the IFFT pre-processor. It can be one of the following values: *0*, *1*, *true* or *false*. If the value is *1* or *true*, the IFFT is performed on the interpolated data. If the value is *0* or *false*, no interpolation is done and the IFFT is performed directly on the scan data. The default value is *1* or *true*.

Scripting – Calculations

Any number of calculations in the calculations section can be defined. Calculations support operations on complex and real vectors. Complex vectors are supported, because the result from the IFFT pre-processor yields a complex vector. Each calculation uses a nested definition. The definition describes an expression tree (Wikipedia description). Below is an example of a calculation:

```
<calculation name="Example">
    <subtract>
        <divide>
            <measurement camera="1"/>
            <measurement camera="2"/>
        </divide>
        <divide>
            <measurement camera="3"/>
            <measurement camera="4"/>
        </divide>
    </subtract>
</calculation>
```

The equivalent equation for the above definition is:

$$\text{Example} = \frac{M_1}{M_2} - \frac{M_3}{M_4} \quad ,$$

where M_1, M_2, M_3 and M_4 are the pre-processed results from camera 1 to 4, respectively. In the example, the `subtract`, `divide` and `measurement` XML tags define the operations to be performed by the calculation. These tags are also called operators.

There are binary, unary and leaf operators. Binary operators require two child tags. In the example above, each `divide` operator has two `measurement` tags as children. A unary operator requires a single child tag. Leaf operators do not have child tags. Leaf operators are furthest to the right in the nested definition. In the above example, all `measurement` tags are leaf operators.

Table 1, Table 2 and Table 3 list the supported binary, unary and leaf operators, respectively.

Operation	Scalar-Scalar (s_1, s_2)	Scalar-Vector (s, v)	Vector-Vector (v_1, v_2)
Multiplication	$s_1 \cdot s_2$	$v_n \cdot s$ or $s \cdot v_n$, $n = 0..N$	$v_{1,n} \cdot v_{2,n}$, $n = 0..N$
Division	s_1/s_2	v_n/s, $n = 0..N$	$v_{1,n}/v_{2,n}$, $n = 0..N$
Addition	$s_1 + s_2$	$v_n + s$ or $s + v_n$, $n = 0..N$	$v_{1,n} + v_{2,n}$, $n = 0..N$
Subtraction	$s_1 - s_2$	$v_n - s$, $n = 0..N$	$v_{1,n} - v_{2,n}$, $n = 0..N$

Table 7 Binary operators.

Operation	Scalar (s)	Vector (v)
Magnitude	Not supported	$\lvert v_n \rvert$, $n = 0..N$
Phase	Not supported	$\widehat{v_n}$, $n = 0..N$

Table 8 Unary operators.

Operation	Result
Measurement	Returns the data of a given camera. The data is returned as a vector containing the 1024 values of the last pre-processed scan. If the IFFT pre-processor is used, the vector will contain complex values.

Scalar	Defines a single real, scalar value.

Table 9 Leaf operators.

Calculation definition start

A calculation and its definition starts with a calculation entry. A calculation entry in the script file has the following format:

```
<calculation [name="NAME"] [keepscans="K"]>
    … (operations)
</calculation>
```

NAME	:	The name of the calculation. In the Glaz UI this will be displayed in the legend of the measurement plot.
K	:	Can be one of the following values: *0*, *1*, *true* or *false*. If the value is *1* or *true* each hardware averaged scan is saved and can be viewed in the Glaz UI. The default value is *0* or *false*.

The operations of a calculation are described by a nested definition using the operators summarised in Table 7, Table 8 and Table 9. See the sections below for more information on operators.

⚠ The calculation entry must only contain a single operator. This can either be a binary, unary or leaf operator.

After receiving the pre-processed data from each camera, the calculations are performed. The result of each calculation is accumulated and an average is calculated (see "Hardware and software averaging"). It is assumed, that the result is real. If the result contains complex values, the imaginary part will be ignored.

Binary operators

All binary operators have the following format:

```
<OP>
    <…>        (first child operator)
    <…>        (second child operator)
</OP>
```

OP	:	Can be one of the following values: *add*, *subtract*, *multiply* or *divide*.

The function of the first and second child operators are described in the table below:

Operation	First child operator		Second child operator
Add	term 1	+	term 2
Subtract	minuend	-	subtrahend
Multiply	factor 1	·	factor 2
Divide	numerator	/	denominator

Table 10 Child tags of binary operators.

Unary operators

All unary operators have the following format:

```
<OP>
    <…>        (child operator)
</OP>
```

OP : Can be one of the following values: `magnitude` or `phase`.

The magnitude or phase is calculated from the data defined by the child operator.

The phase operator supports phase unwrapping. The algorithm for unwrapping the phase uses a threshold value. If the difference in phase between two successive data points is larger than the threshold value, then 2π will be added or subtracted from the next data point. The threshold value can be tweaked with the `threshold` attribute:

```
<phase [threshold="T"]>
```

T : A decimal number specifying the threshold as a faction of 2π. The default value is 0.75. For **T** > 1.0, phase unwrapping will be disabled.

Leaf operators

Tree leaf operators are supported: *scalar, measurement* and *reference*. The scalar operator has the following format:

```
<scalar value="V"/>
```

V : A decimal number. This is the return value of the scalar operator.

The measurement operator returns the pre-processed scan result of a camera. It has the following format:

```
<measurement camera="NUM" [ifft="IFFT"] [interpolate="I"]/>
```

NUM : The camera number that was assigned to a camera in the camera definition section. The operator will return the pre-processed data for the given camera.

IFFT : Can be one of the following values: *0, 1, true* or *false*. When not specified, the default value is *false*. If the value is *1* or *true* the operator will return the inverse Fourier transform (IFFT) of the pre-processed scan result. This function is useful when both IFFT and non-IFFT results are required. Note: this method is less efficient, than using the IFFT pre-processor.

I : Can be one of the following values: *0, 1, true* or *false*. If the value is *1* or *true*, the IFFT is performed on the interpolated data. If the value is *0* or *false*, no interpolation is done and the IFFT is performed directly on the scan data. The default value is *1* or *true*.

The reference operator is used to access the result of another calculation. This is necessary when gated calculations need to be combined into a higher-level calculation. It has the following format:

```
<reference calculation="CALC"/>
```

CALC : The name of a previously defined calculation (see "Calculation definition start"). The operator will return result of the referenced calculation.

A calculation containing a measurement operator is also called a measurement calculation. A calculation containing a reference operator is also called a reference calculation.

⚠ A calculation may not contain both a measurement and a reference operator. Reference calculations should only reference measurement calculations. Measurement calculations should be defined first, followed by any required reference calculations.

PulseSync: Multi-camera configuration and synchronisation

The *Glaz LineScan-I with PulseSync* was designed to support multi-camera operation. The design ensures that cameras remain synchronised and that they take scans at the same time. To achieve synchronisation, all the *Sync* ports of a multi-camera configuration must be connected together. Each camera is connected to a separate USB port on the PC or USB hub.

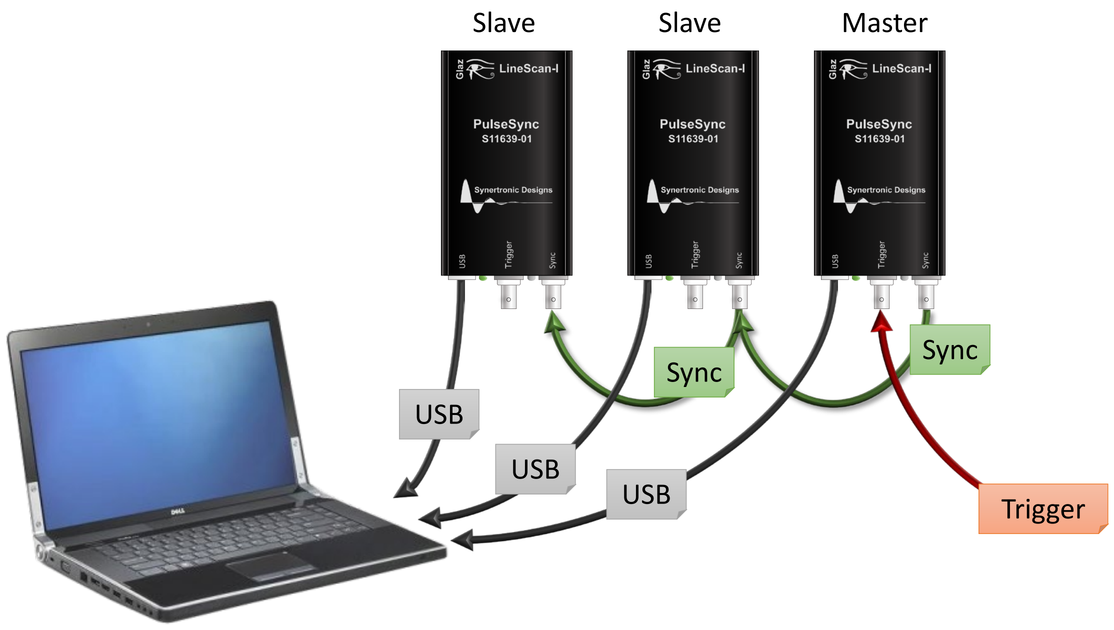


Figure 12 Multi-camera configuration with three cameras.

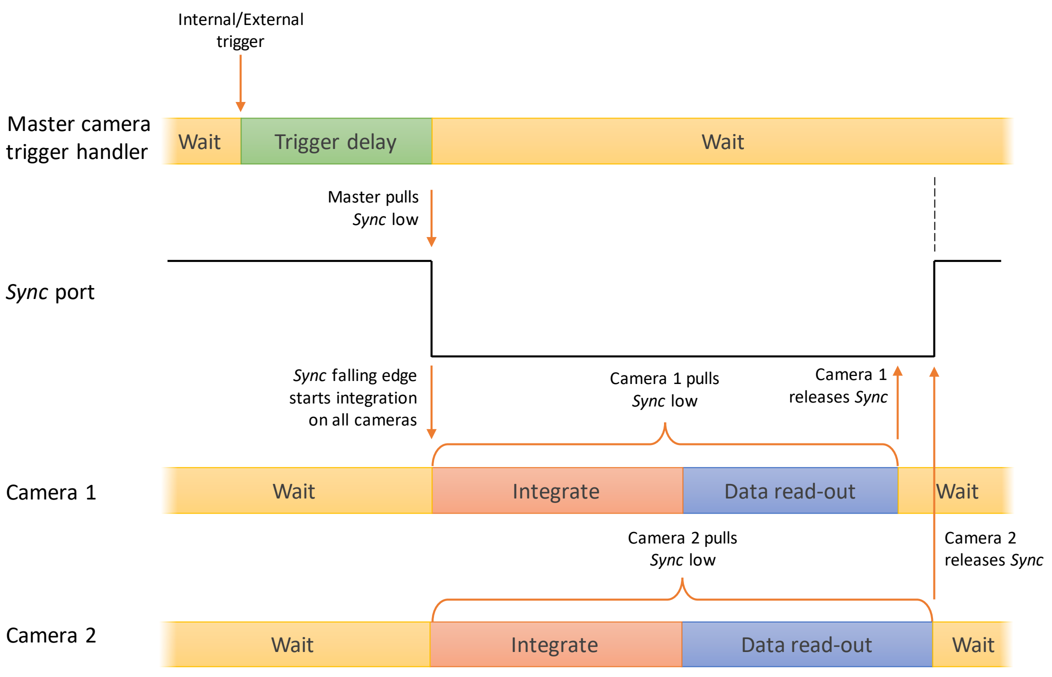


Figure 13 *PulseSync* synchronisation via the *Sync* port.

At least one camera must be defined as the *Master*. The *Master* receives the external trigger signal or generates the internal trigger signal. After the configured trigger delay, the *Master* pulls the *Sync* port low. This initiates the integration period on all connected cameras. Each camera pulls the *Sync* port low during the integration and data read-out period. The *Sync* port is released only after all the cameras have completed the data read-out. As long as any one of the cameras are busy with a scan or are sending data to a PC, the the *Sync* signal will remain asserted low.

⚠ If triggered internally, leave the *Trigger* port unconnected. The *Trigger* port of all slave cameras must be left unconnected.

When a delayed trigger signal is received by the master camera, while the *Sync* signal is asserted, this trigger will be ignored. A multi-camera configuration will remain synchronised, even if the trigger frequency is higher than the processing speed of the cameras and PC. If the trigger frequency is higher than system's processing speed, some of the triggers will be ignored. Important to note, that all cameras will ignore the same trigger signal. It is safe to use a multi-camera configuration with USB 1.1 interfaces, USB 2.0 hubs or with slower PCs. The only side-effect will be a lower achievable scan rate and measurements can take longer.

PulseSync: Combining Glaz LineScan-I and Glaz-PD devices

It is possible to combine *Glaz LineScan-I PulseSync* camera measurements with *Glaz-PD* devices. More information about *Glaz*-PD devices and its concepts can be found in the *Glaz-PD* manual.

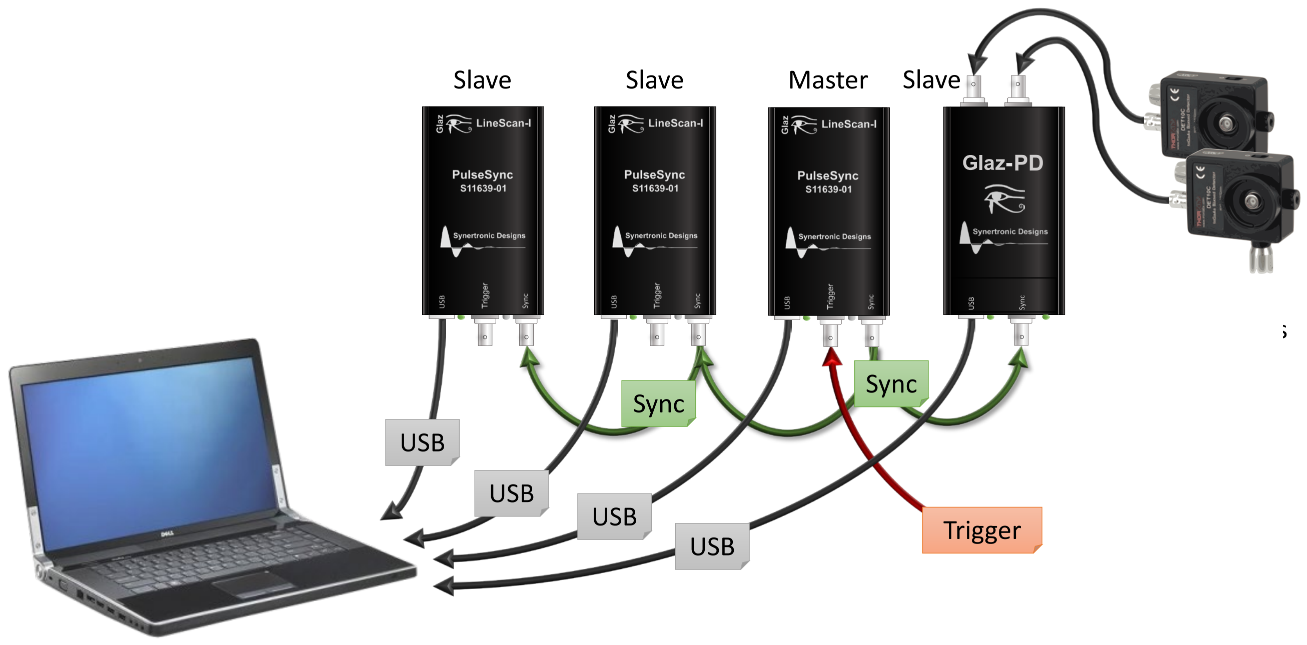


Figure 14 Combining *Glaz LineScan-I PulseSync* cameras and *Glaz-PD*.

The following operations can be performed in conjunction with *Glaz-PD* devices:

- Normalisation of camera scans using measured laser pulse intensities from *Glaz-PD* devices.
- Gating of calculations based on "triggered" and "not triggered" states, detected by *Glaz-PD* devices.

In order to use measurements from *Glaz-PD* devices together with camera measurements, the *Glaz-PD* must be operated in synchronised mode. A typical connection diagram is shown in Figure 14.

Scripting – Glaz-PD definitions

In order to use *Glaz-PD* devices, the must first be uniquely identified. Each *Glaz-PD* has an entry in the script file as depicted in the example in Figure 15.

```
<!DOCTYPE GlazScript>
<config>
  <camera serial="SYBP005010001" number="1" master="1"/>    Camera definitions
  <pd serial ="SYBP006010001" number="1" ch1="1"/>           Glaz-PD definitions
  <preprocessor camera="1" type="calibrate"/>
  <preprocessor camera="1" type="subtract_background"/>      Pre-processing steps
  <calculation name="Camera 1" keepscans="1">
    <measurement camera="1"/>                                 Calculations
  </calculation>
</config>
```

Figure 15 Defining *Glaz-PD* devices in the XML script file.

Each device definition uses the following format:

```
<pd serial="SN" number="NUM" [ch1="E1"] [ch2="E2"]
 [highgain2="G1"] [highgain1="G2"] [window="W"]/>
```

SN	:	The serial number of a device is printed onto the back of the camera. The serial number has 13 digits and has the format: SYPB + device number + device version + instance number. For example: SYPB006010001. A serial number may only be defined once in a script file.
NUM	:	The device number that will be assigned to the *Glaz-PD* device with the given serial number. This number will be used in the rest of the script file. This must be a unique number in the range [1 .. 1000].
E1,E2	:	Can be one of the following values: *0*, *1*, *true* or *false*. If the value is *1* or *true* the given channel will be enabled. The default value is 0 or *false* and when omitted the relevant channel will be disabled.
G1,G2	:	Can be one of the following values: *0*, *1*, *true* or *false*. If the value is *1* or *true* the high-gain stage (x6 gain) for the given channel will be enabled. The default value is *0* or *false* and when omitted the relevant high-gain stage will be disabled.
W	:	This specifies the window period in [μs]. The default value is 10 μs.

Scripting - Normalising camera scans

Calculations can be normalised using one of two methods.

A camera measurement may be directly normalised by adding an additional parameter to the camera measurement leaf operator (see "Leaf operators"):

```
<measurement camera="NUM" pdnorm="PD_LIST"/>
```

NUM : The camera number that was assigned to a camera in the camera definition section. The operator will return the pre-processed data for the given camera.

PD_LIST : This is a comma-separated list of *Glaz-PD* device numbers and channel numbers. Each entry in the list uses the following format: **PD_NUM:PD_CH**
For example: `pdnorm="1:1,1:2"`. This will use channel 1 and 2 of the *Glaz-PD* device with the device number "1" to normalise the camera scan. The resulting normalisation factor for the i^{th} scan will be:

$$\frac{I_{1:1,0}}{I_{1:1,i}} \cdot \frac{I_{1:2,0}}{I_{1:2,i}}$$

A calculation can also be normalised at a higher level by adding a unary normalisation operator. The operator has the following format:

```
<normalise pdnorm="PD_LIST">
    <…>        (child operator)
</normalise>
```

PD_LIST : This is a comma-separated list of *Glaz-PD* device numbers and channel numbers. Each entry in the list uses the following format: **PD_NUM:PD_CH**

For example: `pdnorm="1:1,1:2"`. This will use channel 1 and 2 of the *Glaz-PD* device with the device number "1" to normalise the camera scan. The resulting normalisation factor for the i^{th} scan will be:

$$\frac{I_{1:1,0}}{I_{1:1,i}} \cdot \frac{I_{1:2,0}}{I_{1:2,i}}$$

Scripting - Gating calculations

It is possible to "gate" calculations. This makes it possible to perform defined calculations only if the associated *Glaz-PD* channel was either triggered or not triggered. In order to gate a calculation additional parameters must be specified in the calculation start definition (see "Calculation definition start"):

```
<calculation [name="NAME"] [keepscans="K"] pdgate="PD" gatestate="S">
    … (operations)
</calculation>
```

NAME : The name of the calculation. In the Glaz UI this will be displayed in the legend of the measurement plot.

K : Can be one of the following values: *0*, *1*, *true* or *false*. If the value is *1* or *true* each hardware averaged scan is saved and can be viewed in the Glaz UI. The default value is *0* or *false*.

PD : This is only one entry, instead of a list, defining the *Glaz-PD* device number and channel number. The entry uses the following format: **`PD_NUM:PD_CH`**
For example: `pdgate="1:2"`. This will use channel 2 of the *Glaz-PD* device with the device number "1".

S : Can be one of the following values: *0*, *1*, *true* or *false*. If the value is *1* or *true* the calculation will be performed and accumulated only when the associated Glaz-PD channel was triggered. If the value is *0* or *false* the calculation will be performed and accumulated only when the associated Glaz-PD channel was <u>not</u> triggered.

When `keepscans` is enabled for a gated calculation all scans are kept, even scans for which the calculation was not performed. When a calculation is not performed due to gating, that kept scan will be all zeros.

Error detection

A measurement will be terminated if a digitized value from a *Glaz-PD* was expected, but that channel was not triggered. This will happen if a camera scan needs to be normalised, but the channel used for the normalisation was not triggered during the specified *Glaz-PD* window period.

Examples

In order to get a better understanding of how to compile script files a few examples are presented. The *Glaz UI* installation contains an example script file for each example discussed below.

The following symbols are used:

- S_n The measured spectrum from the nth camera.
- B_n The measured background for the nth camera.
- I_m The intensity measured by a *Glaz-PD* device.
- F_k Functions that need to be calculated.

In each example, a function is presented. The way to compile a script file to calculate this function is shown and how function is mapped onto the script. Mappings are indicated by coloured boxes: Boxes with the same colour in the script and on the function show the equivalent mapping.

All calculations performed by the script are averaged over the specified number of scans (see "Hardware and software averaging").

⚠ The serial numbers used in the example scripts are for illustration only. In a practical set-up, the actual camera serial numbers must be used.

Example 1 – Single camera measurement

$$\boxed{F_1 = S_1\text{-}B_1}$$

The function simply takes scans with a single camera and subtracts the initially measured background. The script is written as follows:

```
    <!DOCTYPE GlazScript>
    <config>
        <camera serial="SYBP005010001" number="1" master="1"/>
        <preprocessor camera="1" type="subtract background"/>
        <calculation name="F1">
            <measurement camera="1"/>
        </calculation>
    </config>
```

The function is mapped as follows:

$$\boxed{F_1 = \boxed{S_1\text{-}B_1}}$$

The blue box represents the pre-processor step in the script. This step already subtracts the background from each camera measurement. The calculation simply returns the camera measurement, which includes the background subtraction.

Example 2 – Ratio of two camera measurements

$$\boxed{F_2 = \frac{S_1\text{-}B_1}{S_2\text{-}B_2}\text{-}1}$$

The function simply takes scans from two cameras and subtracts the initially measured backgrounds from each scan. The ratio of the scans (with subtracted backgrounds are then taken). The script is written as follows:

```
    <!DOCTYPE GlazScript>
    <config>
        <camera serial="SYBP005010001" number="1" master="1"/>
        <camera serial="SYBP005010002" number="2" master="0"/>
        <preprocessor camera="1" type="subtract background"/>
        <preprocessor camera="2" type="subtract_background"/>
        <calculation name="F2">
            <subtract>
                <divide>
                    <measurement camera="1"/>
                    <measurement camera="2"/>
                </divide>
                <scalar value="1"/>
            <subtract>
        </calculation>
    </config>
```

The function is mapped as follows:

$$F_2 = \boxed{\frac{\boxed{S_1\boxed{\text{-}B_1}}}{\boxed{S_2\boxed{\text{-}B_2}}}\text{-}\boxed{1}}$$

The blue and red boxes represent the pre-processor steps in the script. These steps already subtract the backgrounds from each camera measurement. The calculation divides the measurements from camera 1 with the measurements from camera 2 and subtracts the value of 1. The measurements (green and yellow boxes) already include the background subtraction.

Example 3 – Normalised ratio of two camera measurements

$$F_3 = \frac{I_{1,0}}{I_1}\left(\frac{S_1\text{-}B_1}{S_2\text{-}B_2}\text{-}1\right)$$

The function is the same as in Example 2, except the result is also normalised with respect to the first measured intensity $I_{1,0}$ (see "Scripting - Normalising camera scans" for more information). The script is written as follows:

```
    <!DOCTYPE GlazScript>
    <config>
        <camera serial="SYBP005010001" number="1" master="1"/>
        <camera serial="SYBP005010002" number="2" master="0"/>
        <pd serial="SYBP006010001" number="1" ch1="1" ch2="0"/>
        <preprocessor camera="1" type="subtract_background"/>
        <preprocessor camera="2" type="subtract_background"/>
        <calculation name="F3">
            <normalise pdnorm="1:1">
                <subtract>
                    <divide>
                        <measurement camera="1"/>
                        <measurement camera="2"/>
                    </divide>
                    <scalar value="1"/>
                </subtract>
            </normalise>
        </calculation>
    </config>
```

The function is mapped as follows:

$$F_3 = \boxed{\boxed{\frac{I_{1,0}}{I_1}}\boxed{\left(\frac{S_1\text{-}B_1}{S_2\text{-}B_2}\text{-}1\right)}}$$

The grey box represents the same calculation as in Example 2. The red box represents the normalisation. In this example the normalisation is performed using channel 1 of *Glaz-PD* device number 1 (`pdnorm="1:1"`). The measurement of the initial intensity $I_{1,0}$ and calculation of the ratio $I_{1,0}/I_1$ is performed automatically.

Example 4 – Gated (even/odd) measurements with two cameras

$$F_4 = \frac{I_{2,0}}{I_{2,even}} \cdot \frac{I_{1,0}}{I_{1,even}} \left(\frac{S_{1,even} - B_1}{S_{2,even} - B_2} - 1 \right) - \frac{I_{2,0}}{I_{2,odd}} \left(\frac{S_{1,odd} - B_1}{S_{2,odd} - B_2} - 1 \right)$$

In transient absorption spectroscopy (TAS) a pump-probe approach is used. The target illumination alternates between probe-only and pump-probe. The alternating illumination states are also called even/odd states. The *Glaz-PD* device is used to differentiate between these two states. Calculations are then gated based on the *Glaz-PD* states. The script is written as follows:

```
<!DOCTYPE GlazScript>
<config>
    <camera serial="SYBP005010001" number="1" master="1"/>
    <camera serial="SYBP005010002" number="2" master="0"/>
    <pd serial="SYBP006010001" number="1" ch1="1" ch2="1"/>
    <preprocessor camera="1" type="subtract_background"/>
    <preprocessor camera="2" type="subtract_background"/>
    <calculation name="Even" pdgate="1:1" gatestate="1">
        <normalise pdnorm="1:1,1:2">
            <subtract>
                <divide>
                    <measurement camera="1"/>
                    <measurement camera="2"/>
                </divide>
                <scalar value="1"/>
            </subtract>
        </normalise>
    </calculation>
    <calculation name="Odd" pdgate="1:1" gatestate="0">
        <normalise pdnorm="1:2">
            <subtract>
                <divide>
                    <measurement camera="1"/>
                    <measurement camera="2"/>
                </divide>
                <scalar value="1"/>
            </subtract>
        </normalise>
    </calculation>
    <calculation name="F4">
        <subtract>
            <reference calculation="Even"/>
            <reference calculation="Odd"/>
        </subtract>
    </calculation>
</config>
```

The function is mapped as follows:

$$F_4 = \boxed{\underbrace{\boxed{\frac{I_{2,0}}{I_{2,even}} \cdot \frac{I_{1,0}}{I_{1,even}} \left(\frac{S_{1,even}-B_1}{S_{2,even}-B_2} - 1 \right)}}_{\text{Even term}} - \underbrace{\boxed{\frac{I_{2,0}}{I_{2,odd}} \left(\frac{S_{1,odd}-B_1}{S_{2,odd}-B_2} - 1 \right)}}_{\text{Odd term}}}$$

The yellow and green boxes each represent the same calculation as in Example 3. Except, the even term (yellow box) uses two normalisation factors. In this example channel 1 of the *Glaz-PD* device number 1 is used to perform the gating function. When a scan is triggered and *Glaz-PD* channel 1 is also triggered, then the even term will be calculated. When a scan is triggered and *Glaz-PD* channel 1 is not triggered, then the odd term will be calculated (see "Scripting - Gating calculations" for more information). Since gating can only be applied to calculations, the complete function F_4 cannot be represented by a single calculation in the script file. A third calculation needs to be defined. This calculation subtracts the odd term from the even term. This is a reference calculation and it references the "Even" calculation (yellow boxes) and "Odd" calculation (green boxes). The reference calculation is performed each time an "Odd" and "Even" calculation result is available. This happens only with every second scan.

The result for 10 scans could look as follows:

Scan	Glaz-PD channel 1 triggered?	"Odd"		"Even"		"F4"	
		Calculated?	Kept scan result	Calculated?	Kept scan result	Calculated?	Kept scan result
1	×	✓	✓	×	All zeros	×	All zeros
2	✓	×	All zeros	✓	✓	✓	✓
3	×	✓	✓	×	All zeros	×	All zeros
4	✓	×	All zeros	✓	✓	✓	✓
5	×	✓	✓	×	All zeros	×	All zeros
6	✓	×	All zeros	✓	✓	✓	✓
7	×	✓	✓	×	All zeros	×	All zeros
8	✓	×	All zeros	✓	✓	✓	✓
9	×	✓	✓	×	All zeros	×	All zeros
10	✓	×	All zeros	✓	✓	✓	✓
Average		2,4,6,8,10		1,3,5,7,9		1,3,5,7,9	

The ✓ and × indicate whether a calculation is performed or not. In addition, the result for kept scans (when `keepscans="1"`) is also indicated. In the bottom row the scan numbers used to calculate the averages is shown.

IMPORTANT NOTICE

Synertronic Designs on the web:	www.synertronic.co.za
E-mail:	info@synertronic.co.za
Postal address:	Kaneel Cr 34 Stellenbosch 7600 South Africa