APPLICATION NOTE

How to Set Up Several Basler racer Cameras in One Inspection Row

Valid only for Basler racer GigE camera models

Document number: AW001311
Version: 02 Language: 000 (English)
Publication date: 13 January 2015

BASLER
the power of sight

How to Set Up Several Basler racer Cameras in One Inspection Row

Enhanced resolutions through series connection of multiple Basler racer line-scan cameras

Table of Contents

When one single line-scan camera is not sufficient to cover the needs of an optical inspection application, multiple line-scan cameras can be configured to work next to one another — allowing for the entire width of a material conveyor or web to be captured or to increase the resolution to detect even smallest defects.

This application note describes the setup of this kind of multi-camera setup using three Basler racer GigE line-scan cameras. It explains in detail what must be taken into account for the configuration of this type of setup and explains step-for-step the necessary settings and potential alternatives.

One exemplary application illustrates how a network is configured and the triggers synchronized, and how settings are made using the pylon Viewer Feature Tree. It then concludes with a sample setup of the software application.

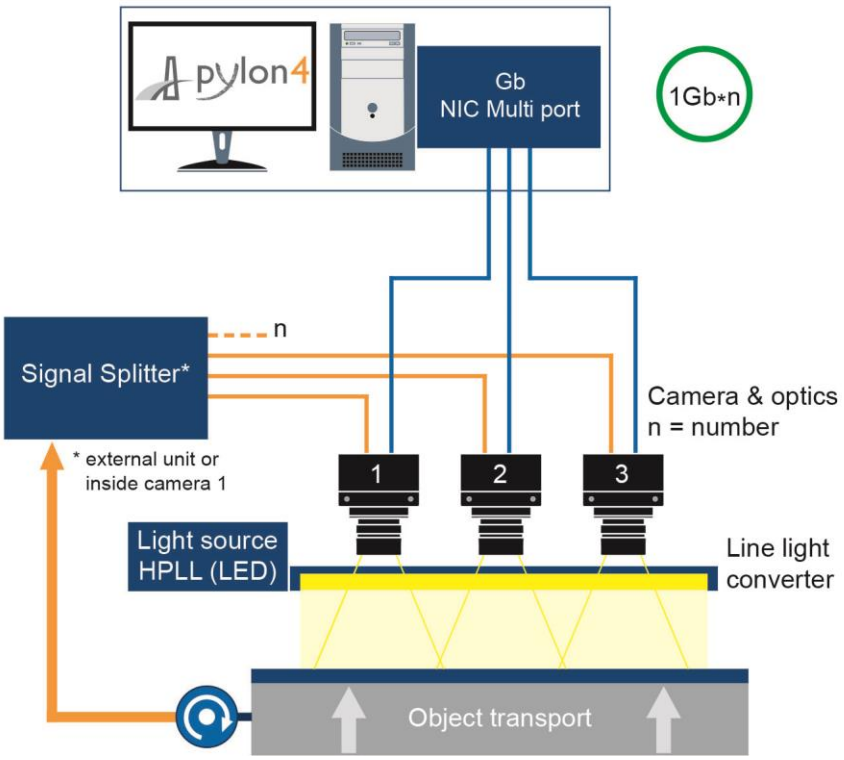
1 Necessary elements – General

As an example for an inspection system the following components were chosen:

- Camera: 3x line-scan Basler racer 2048 GigE monochrome: p/n105993 (*Link*) f/w: 1.0-6 and higher
- Cable for supply voltage, single side with Hirose plug 10A-7P-6S (*Link*)
- IO cable: I/O cable, HRS 12p/open, 3 m: p/n 2000026691 (*Link*)
- Ethernet data cable Cat6 (*Link*)
- Incremental Encoder Stegman DRS61-A4A08192, 8192 counts/rev resolution (*Link*)
- Transport: Drop (continuous revolution) 400 mm/s
- Software SDK: pylon 4.1 (*Link*)
- Light source: HPLL LED Module intensity regulated p/n 106252 (*Link*)
- Lighting rig: Linear light converter (*Link*), plastic lenses; BF method reflection

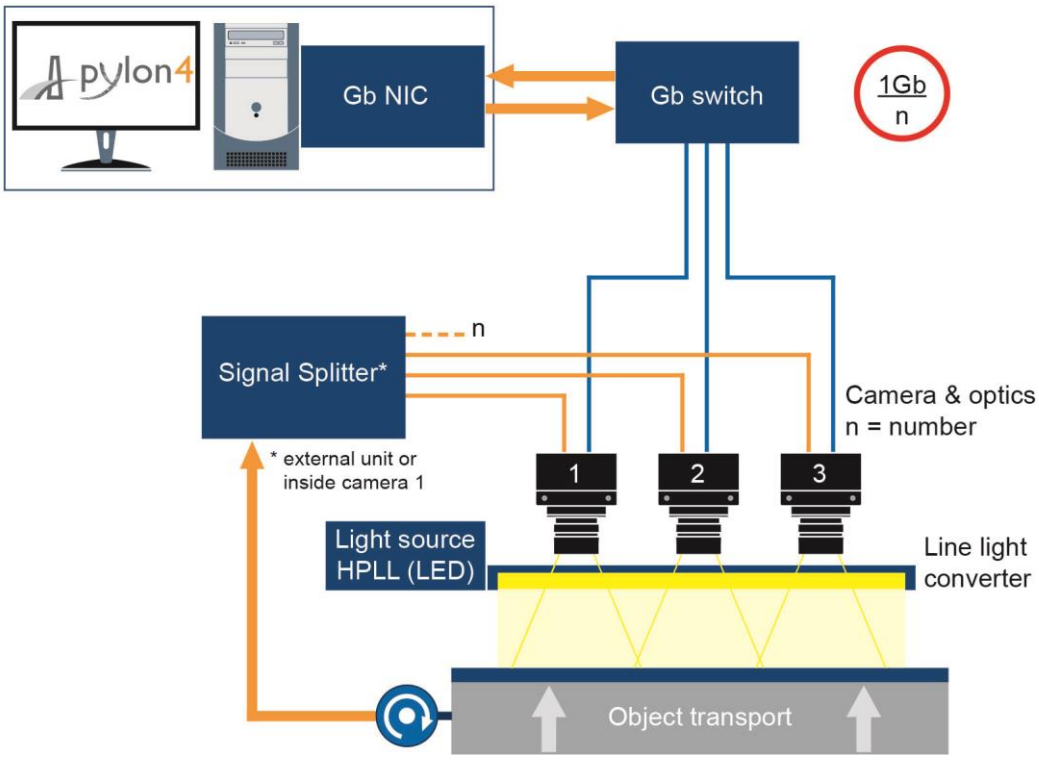
The following images present an overview of all elements involved. They also represent the alternative between

- the use of a multi-port network card for parallel operation of the cameras:



and

- the use of a switch for bundled communication, requiring only a single-port network card:



This depiction shows only one alternative for splitting the encoder signal. The procedure for forwarding exposure signals starting from an encoder-triggered master camera was integrated later and is thus not shown.

Sample optical scenario

Optics in use: Schneider-Kreuznach with 45µm resolution and sensor pixel size of 7µm.

I. The values of flange back and adapter length

	mm
Camera Flange to CCD Distance (Camera Depth)	18
Adapter Thickness (M42x1-Mount)	7

II. The magnification or the values of sizes

Magnification β' (image size / object size)	0,155556	1/β' = 6,429
Image Size	7	
Object Size	45	

Required Extension Tubes (Unifoc 12)

Lens	Required Extension		Object to Image Distance (00')	Working Distance	Flange Focal Distance @ β'	Object to Camera Face (Master Setup)
	min.	max.	[mm]	[mm]	[mm]	[mm]
Mk-CPN 2.8/28	-25	-13	248.49	190.20	29.69	230.49
Mk-CPN 2.8/35	-18	-6	296.73	231.94	36.19	278.73
Mk-APO-CPN 2.8/40	-10	2	354.22	281.05	44.57	336.22
Mk-APO-CPN 4.0/45	-5	7	397.62	319.43	49.59	379.62
Mk-CPN-S 2.8/50	-5	7	427.53	349.12	49.80	409.53
Mk-APO-CPN 4.0/60	8	20	512.68	421.47	62.61	494.68
Mk-CPN-S 4.0/80	36	48	687.84	569.24	90.00	669.84
Makro MSR 5.6/80*	36	48	706.02	586.98	90.44	688.02
Mk-APO-CPN 4.5/90**	46	58	779.27	649.99	100.69	761.27
Mk-CPN-S 5.6/100***	57	69	876.12	735.73	111.79	858.12

* = Due to the long rear mount of the Makro-Makro-Symmar 5.6/80. extension tubes of min. 18 mm are required between lens and the Unifoc-12 Makro-Symmar 80mm only use when β' is between (0,25 - 4,0) in Forward position only.

** = Due to the long rear mount of the Apo-Componon 4.5/90. extension tubes of min. 18 mm are required between lens and the Unifoc-12

*** = Due to the long rear mount of the Componon 5.6/100. extension tubes of min. 10 mm are required between lens and the Unifoc-12

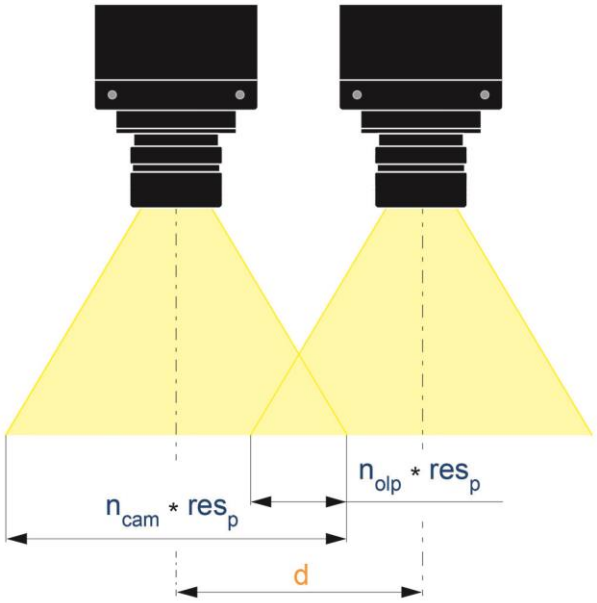
The image shows the results table for a calculation of lenses from the Schneider-Kreuznach company. A magnification factor β` of 0.1555 is presumed (sensor size of 7µm compared with desired object size of 45µm).

2 Overlapping of visible areas

When combining multiple line-scan cameras into an inspection line, the areas of overlapping visible areas must be taken into account. This is necessary to ensure that no pixel information is lost. The overlapping can be detected and removed when image processing is applied.

The physical distance required for the optical axes of the cameras to one another can be calculated based on the object resolution resulting from the desired optical setup and the overlapping.

Distance between the optical axes:



$d = (n_{cam} - n_{olp}) * res_p$

n = Number of pixels,
res = Object plane resolution

In the test setup, this calculation led to the following camera constellation:

Distance opt. axes [mm]	Camera [Pxl]	Overlap [Pxl]	Resolution [µm]
87,66	2048	100	45

If the casing size and an established overlapping are known, then the most dense theoretical camera orientation in the line direction can be provided based on each racer sensor type (line width):

Camera width [mm]	Camera [Pxl]	Overlap [Pxl]	Min Resolution [µm]
56	2048	100,00	28,75
56	4096	100,00	14,01
56	6144	100,00	9,27
56	8192	100,00	6,92
56	12288	100,00	4,59

3 Bandwidths – Lines

GigE

Data transfer for factory settings

<Payload Size> AOI [Pxl*Byte] = 2048*256*1 = 524288 Byte

& <Device Current Throughput>

<Packet Size> 1500 Byte

<Inter-Packet Delay> 0

<Frame Transmission Delay> 0

<Bandwidth Reserve> 10%

= <Resulting Line Rate>

Booster Option

Remove Limits => value C1 = 5.4µs, as part of Exposure Overhead (OH), can be reduced to 3.4µs.

The benefit is 2µs extra exposure time or equivalent line-scan rates.

The following values emerge based on camera type (sensor width) when working under the factory settings:

Camera [Pxl]	Max line rate [Hz]	Exposure OH [µs]	Max Exposure [µs]	Booster on	
2048	51813	5,4	13,90	no	
4096	26000	5,4	33,06	no	
6144	17361	5,4	52,20	no	
8192	12195	5,4	76,60	no	
12288	8136	5,4	117,51	no	

Sample setup: Exposure Mode – Timed (Master Camera) & Trigger Width (Follower Camera)

4 Lighting

Light source

All vision starts with light. Line-scan image capture requires a constant, intensive light source. This is often bundled into one line on the object. Lighting itself is thus already a determining factor for the parameters of image quality. A more intensive beam focus can produce lower gain, thus impacting noise behavior for the better.

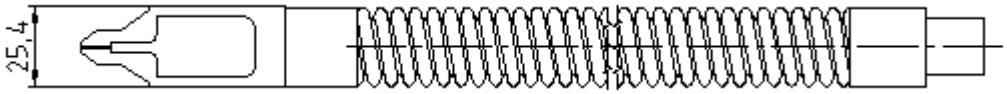
For this setup, a light field approach with HPLL LED source was selected, with direct reflection from the object maintaining the light background and thus requiring less intensity than against a dark field. The dark field by contrast only sends scattered beams into the camera.

Illustration 1: Basler HPLL (high power LED) Module

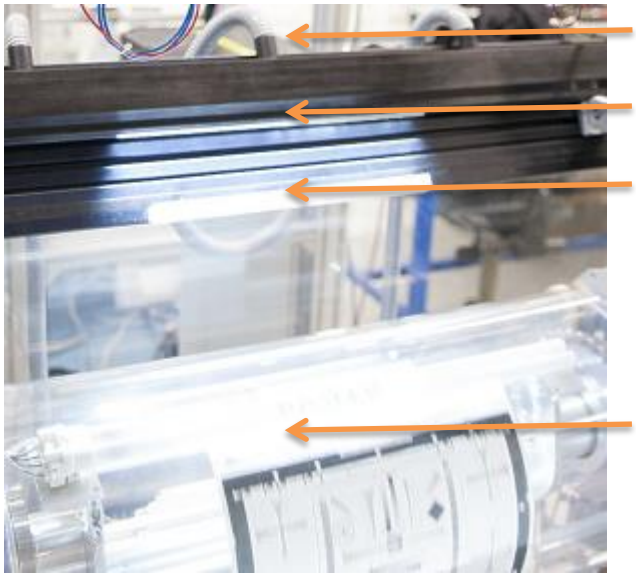
A detailed list of all *Basler partners for lighting components* can be found in the *Support* area of our website.

Linear light adapter

To align the light onto the line recorded by the camera, a *converter* with flexibly twisted and then correspondingly aligned fiber optics is used.



The bundling assumes a correspondingly long plastic or glass lens, depending on the requirements.



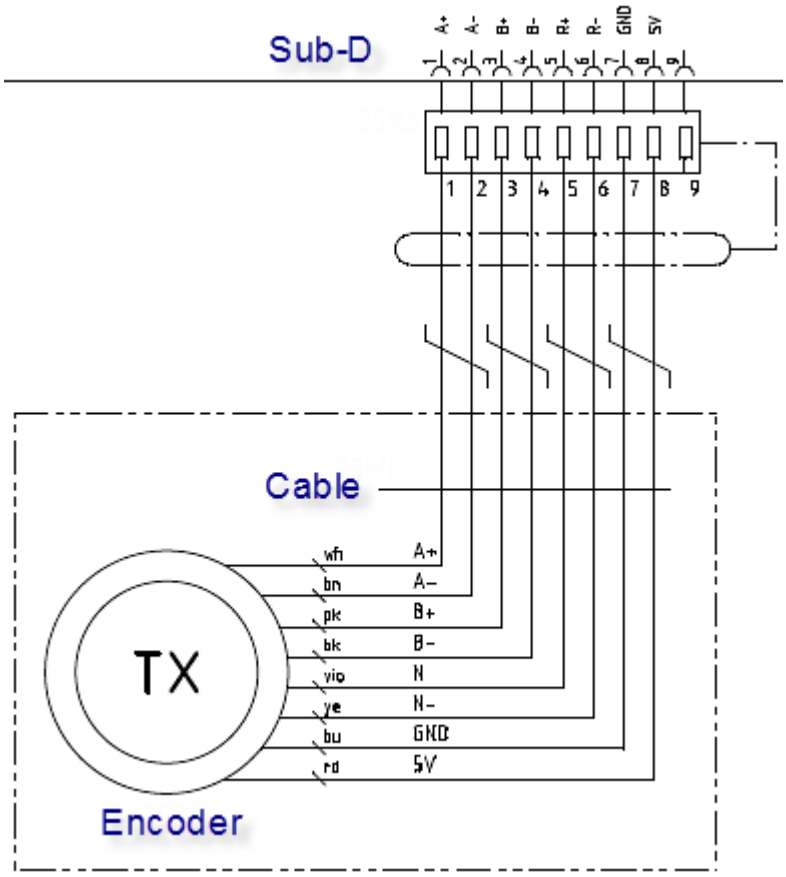
5 Trigger Synchronization

5.1 Shaft Encoder

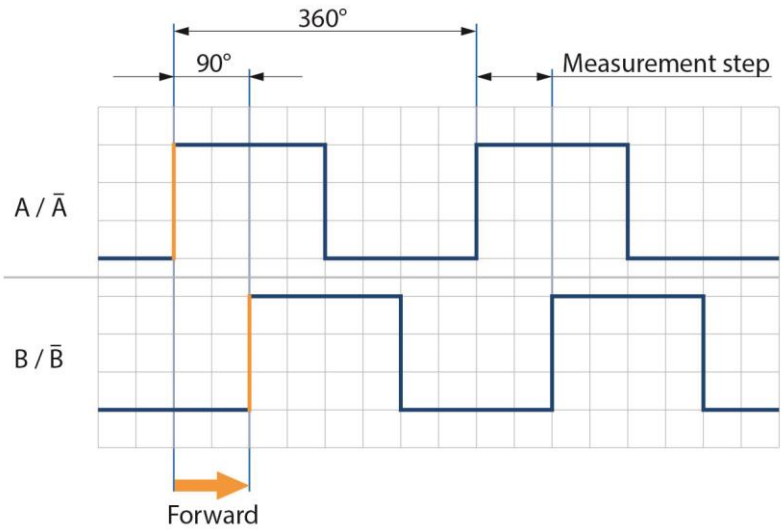
A *Sick* Encoder DRS61 with max. 8192 counts/rev was used.

It is connected as depicted via the Sick accessory cable to a 9-pole sub-D connector. The signals can be switched either directly to one camera or via a splitter to multiple cameras. In the sample setup, only the master camera is operated directly with the shaft encoder signal.

What is important in this case is that the encoder is supplied with +5V/GND. For this reason the cable cannot be directly connected to the camera.



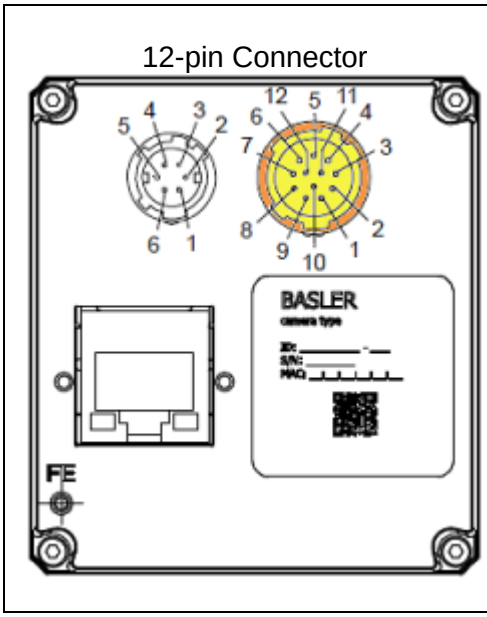
The encoder provides TTL RS422 differential signaling that is received at the camera inputs, as are optional LVDS RS644 differential signaling or LVTTL. See the "*racer GigE Users Manual V3*", notes on special features of the connections.



The camera can evaluate the directional information of both phase-shifted signals A & B. Both signals are to be connected to one input each and parameterized accordingly.

5.2 I/O, Synchronization and Pulsing

To allow for extensive connection options for the signal to the camera, the racer has a separate 12-pole connector. Its assignments are depicted in the following table. The otherwise single 6-pole plug serves here solely to provide the operating voltage.

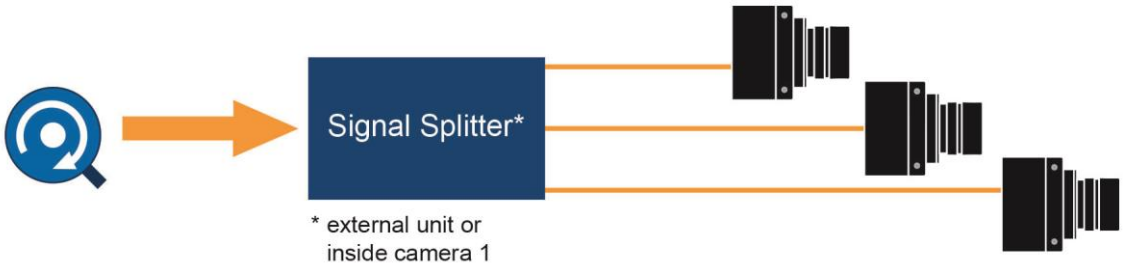


Pin	Designation
1	I/O Input 1-
2	I/O Input 1+
3	I/O Input 3-
4	I/O Input 3+
5	GND
6	I/O Output 1-
7	I/O Output 1+
8	I/O Input 2-
9	I/O Input 2+
10	N/A
11	I/O Output 2-
12	I/O Output 2+

Prefabricated cables can be used for the connection. Assignment differs in some cases from connection cables for cameras of other types. To ensure that you are always using the right cable, first select the camera before calling up the accessories!

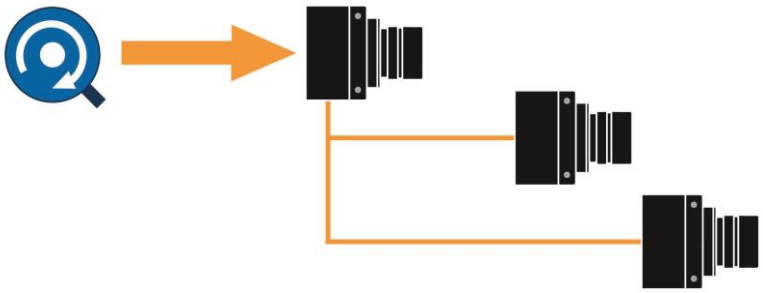
Depending on the requirements, there are several options for synchronous external trigger controls:

- **Encoder signal individually via router to each camera**
  (more complicated; only recommended with a few cameras)



Each camera requires two inputs for the encoder (signal A & B, see illustration for encoder). It can be presumed that there will be synchronous signals, however.

- **Master/follower approach with bus**
  (less expensive, recommended and used in the sample setup; up to 10 followers possible on the bus; activate final resistance(!); same cycle tact on bus)



Up to 10 inputs can be semi-synchronized parallel using an RS-422 bus. [First "Master" camera triggered by encoder and assigned output of

a) Shaft Encoder camera Module (SEM) output,
b) Frequency Converter camera Module (FCM) output,
c) "Exposure Active", respective edge evaluations as trigger equivalent on OUT1].

The options are equally valid

a) generally, if SEM matches the period 1:1 or if the encoder can be programmed,
b) if encoder signal is modified in the FCM or the following cameras require a modified cycle tact,

c) as an alternative to a) and b) and necessitated in connection with "Trigger Width" exposure mode.

All of the following cameras have been synchronized in parallel via the Line1 input. If the master camera using option a) or b) is working with an external frame trigger, then this is also to be passed along to the follower cameras. This is not required in case c) as the master camera passes the tact directly.

Please be sure to take note of the dependencies for signal delays noted hereinafter.

- **Cascade "through" each camera**
  (exotic; possible for different distributors via frequency module)

In principle similar to the master/follower approach, only each camera can modify the tact cycle. Delay times accumulate, however.

In general, the I/O delay should be referenced in terms of the exposure time. Please be sure to review on "Exposure Start Delay" and "Exposure End Delay", totaling between 1-2µs, in "*racer GigE Users Manual V3"*.

In addition to these constants, the parameterized "debouncer" time is also added for input tact cycle signals, as this is in all cases evaluated before each evaluated signal edge. This quickly adds up to several µs that must be at least taken into account in Master/Follower systems.

Shaft Encoder Module (SEM)

The module is described in great detail in the "*racer GigE Users Manual V3*".

It is capable of evaluating directional input signals. In this way the lines can be counted in one or both directions regardless of speed and liberated from minor fluctuations using supplemental counter functions.

Each of the 4 ticks of an encoder counts. For this reason, at the very least use the *Divider = 4* via the frequency converter module.

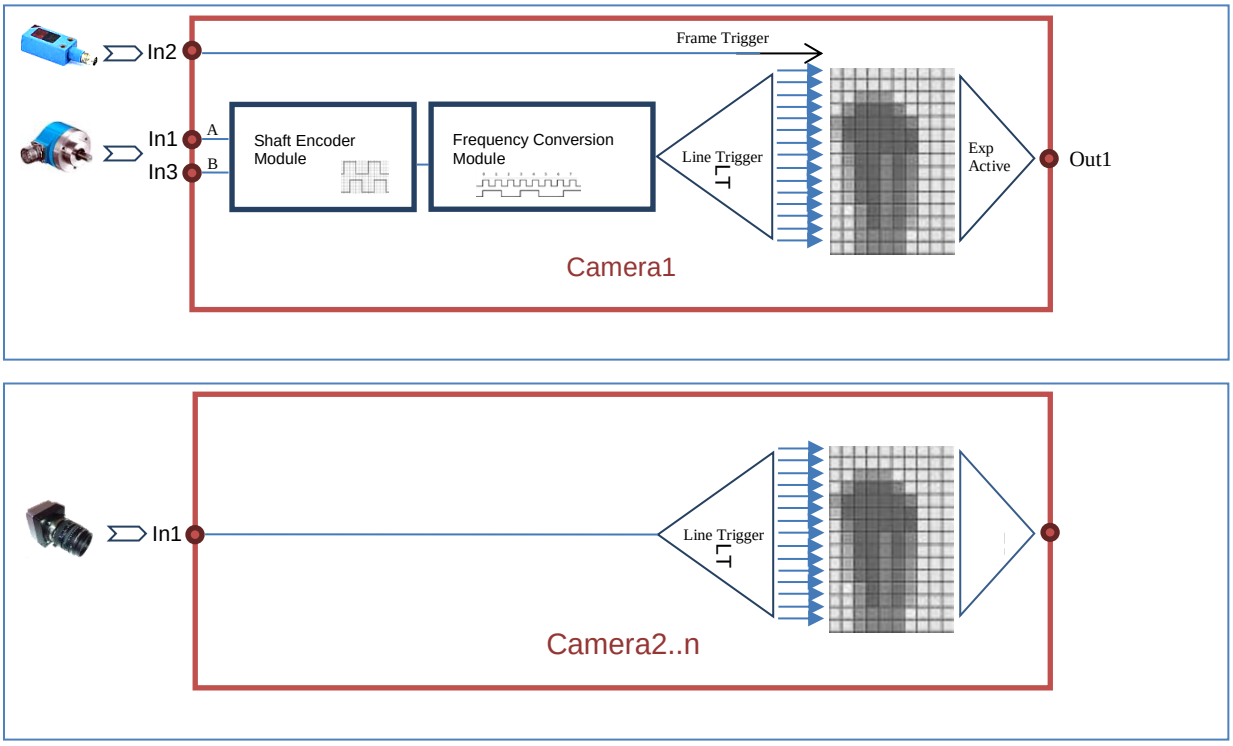
Frequency Converter Module (FCM)

The module is described in great detail in the "*racer GigE Users Manual V3*".

The FCM can record each input, including the encoder module output, as a signal and then — modified again as an internal signal or sent to an output — be made available an external signal. In the process, it draws upon a splitter-amplifier-splitter chain that also permits for non-integer factors. The most common application, as also shown in the example, is adjusting the object pulsing cycle to the camera pulsing cycle.

As an alternative to the aforementioned encoder feeds, it is naturally also possible to use a customer line trigger signal that can be provided in parallel or for the first camera and which must correspond to the input specifications. This preserves the other options, whereby the flexibility has not yet been described in full here. While not addressed in this Application Note, it can be reviewed at any time in the comprehensive handbook "*racer GigE Users Manual V3"*.

Switch to master/follower pulsing in the sample application:



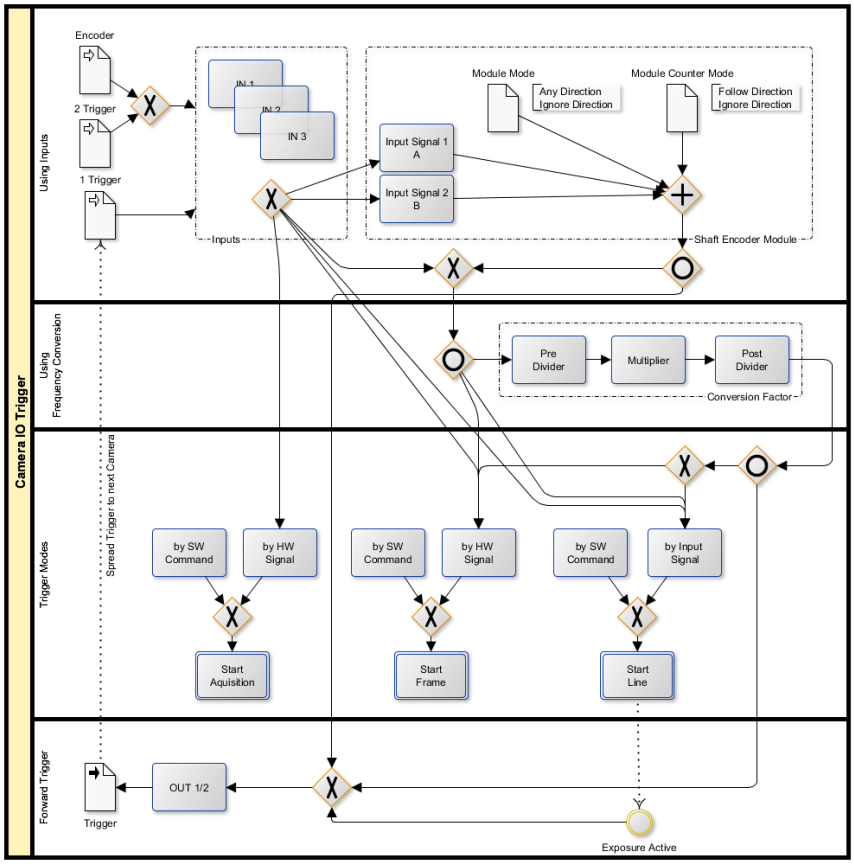


Illustration 2: Options for external signals in synchronization

6 Sample Application

6.1 Sample Setup – Network Configuration

In general, it is important that high-quality components be used to ensure the attainment of the expected transmission rates. To achieve this, Basler offers a selection of *certified accessories*. We have seen again and again that arbitrary, typically low-cost hardware limits what can be achieved.

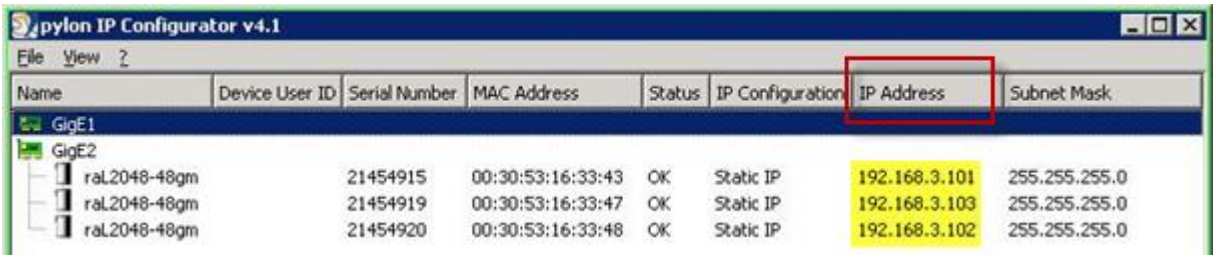
- *POE* does not apply for this camera type.
- *Switch*: Total bandwidth of 1 Gb/s distributed here to all cameras separately; all cameras configured into a subnet (automatic or manually fix addresses possible).
- Better *Multiport NIC*: 1 Gb/s available for each camera; PC must be able to handle this; pay attention to the PCIe version and number of connected lanes; each camera configured into its own subnet (manually fix addresses recommended).
- Normal *Basler pylon GigE filter drive*r (on Compatible Chip Sets with performance driver recommended).

Options for addressing multiple cameras in one application:

	Manual Fix IP Address	Auto DHCP	Auto LL Address
Switch	+ Better if additional NICs automatically receive IP addresses from the PC	+ DHCP server must be present and configured properly for the topology	+ Sensible for varying devices without use of DHCP and only 1 NIC for the PC with LL addresses
Multi Port NIC	+ Recommended for Multi NIC for clearer, more secure division	- Requires harmonized parameterization of the server and good operating system	- Not recommended due to high risks during connection and transmission

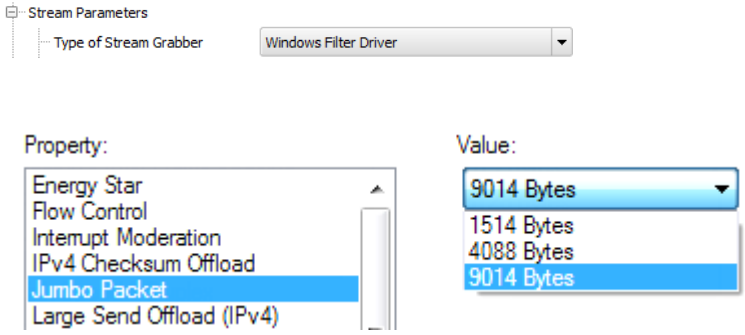
Addressing alternatives tested in the setup example:

	5 Port Netgear Switch	4 Port Intel Pro NIC
Topology	PC NIC 1 (Company Intranet) DHCP 172.100.x.x Subnet 255.255.0.0 Gateway 172.100.0.1 PC NIC 2 (Cameras) 192.168.3.0 Subnet 255.255.255.0 Gateway 0.0.0.0 (unspecified) Cameras 192.168.3.101 192.168.3.102 192.168.3.103	PC NIC 1 (Company Intranet) DHCP 172.100.x.x Subnet 255.255.0.0 Gateway 172.100.0.1 PC NIC 2 - 4 (Cameras) 192.168.1.0, 192.168.2.0, 192.168.3.0 Subnet 255.255.255.0 Gateway 0.0.0.0 (unspecified) Cameras 192.168.1.10 192.168.2.10 192.168.3.10



Packet optimization (Jumbo, serialization for single subnet)

All available parameterization options should be harnessed to ensure the frame rate and image transmission. On the NIC side, the use of the performance driver (hardware-dependent) automatically ensures this; where filter drivers are used for other GigE NIC hardware, "Jumbo Frames" should be activated and set to the maximum value.



On the side of the camera parameter, in the Transport Layer field the packet size should be adjusted to at least 3000.

Other notes for optimal controlling of packet transfer along multiple cameras can be found in the document *"Controlling Packet Transmission Timing with the Interpacket and Frame Transmission Delays on Basler GigE Vision Cameras."*

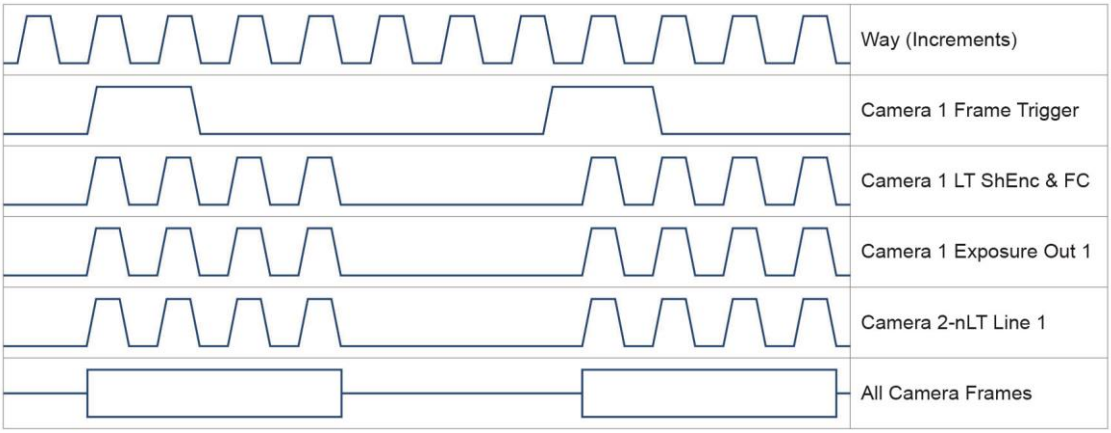
The packet size configured in the camera cannot overrun the packet size as configured in the NIC. Otherwise no packets will be transmitted, and thus no further images.

6.2 Sample Setup – Trigger

The following table compares two image acquisition modes: single frame as a possible alternative to continuous frame acquisition as used in the sample setup.

Acquisition Mode	Single Frame Continuous Frame		*) 1st camera with encoder on L1&3, other cameras are LVTTL triggered on L1
Camera 1	Acquisition Start	Frame Start	Line Start
Trigger Mode Status	ON OFF (free run)	ON OFF (free run)	ON OFF (free run)
Trigger Source	Software Line 1 | 2 | 3 | Frequency Converter | Shaft Encoder	Software Line 1 | 2 | 3 | Frequency Converter | Shaft Encoder	Software Line 1* | 2 | 3*| Frequency Converter | Shaft Encoder*
Camera 2-n	Acquisition Start	Frame Start	Line Start
Trigger Mode Status	ON OFF (free run)	ON OFF (free run)	ON OFF (free run)
Trigger Source	Software Line 1 | 2 | 3 | Frequency Converter | Shaft Encoder	Software Line 1 | 2 | 3 | Frequency Converter | Shaft Encoder	Software Line 1 | 2 | 3 | Frequency Converter | Shaft Encoder

Black font = used, gray font = alternatives

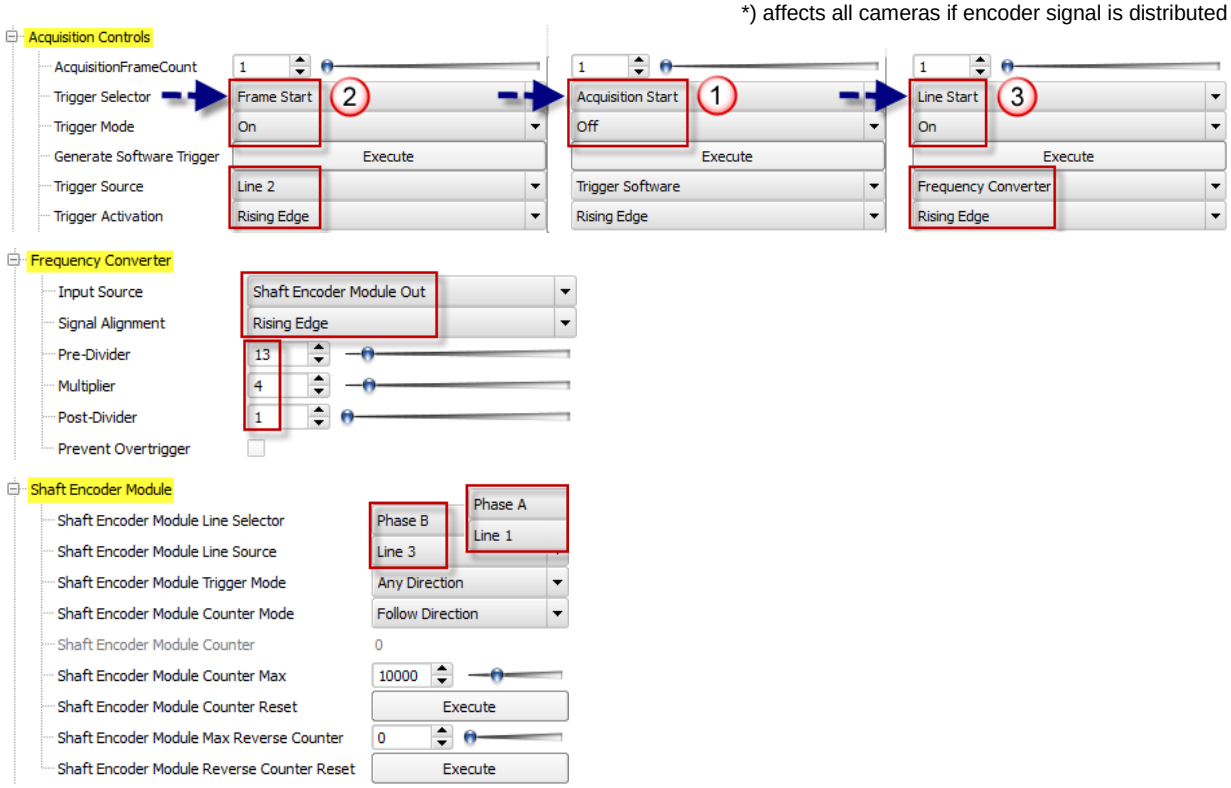


6.3 Sample Setup — Configuration using pylon Viewer Feature Tree Visualization

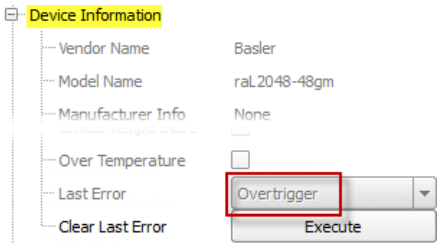
The Basler pylon Viewer Feature Tree displays a clear and easy-to-navigate overview of the required settings.

Camera 1*) trigger / Frequency Converter / Encoder

*) affects all cameras if encoder signal is distributed

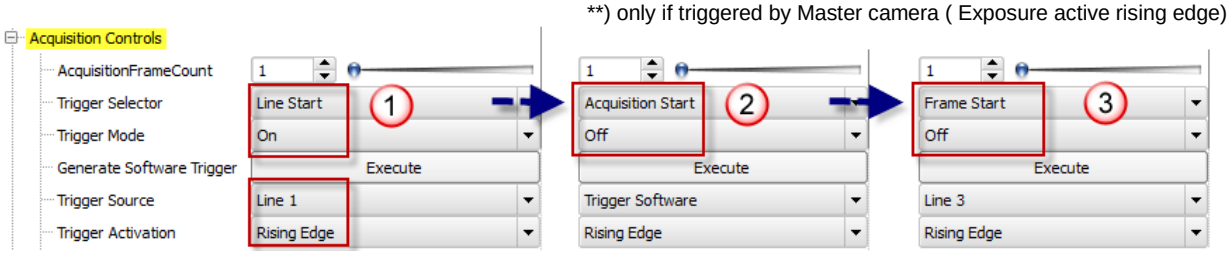


For an external trigger, watch out for overtrigger (overly rapid transport) = loss of lines

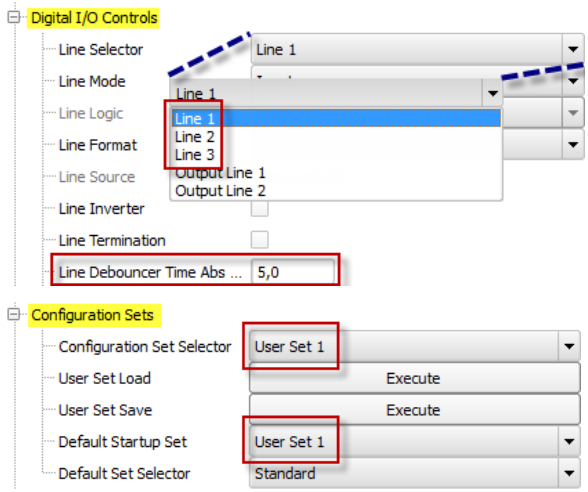


Camera 2-n**) trigger

**) only if triggered by Master camera (Exposure active rising edge)



General adjustments (debouncer, user sets)



6.4 Sample Setup – Software Application

Program C++

"MultiracerGigE_GrabAndProcess"

Filter and find devices

```
// Get the transport layer factory.
CTlFactory& tlFactory = CTlFactory::GetInstance();

// Filter for GigE cameras only.
CDeviceInfo info;
info.SetDeviceClass(BaslerGigEDeviceClass);
DeviceInfoList_t filter;
filter.push_back(info);

// Get all attached devices and exit application if no device is found.
DeviceInfoList_t devices;
if (tlFactory.EnumerateDevices(devices, filter) == 0)
{
    throw RUNTIME_EXCEPTION("At least no GigE camera is present.");
}
```

Sorting performed on basis of assigned IP addresses, in rising order

```
// Will collect and automatically sort cameras IP addresses and according camera context
map<unsigned long, intptr_t> vIPAdresses;

// Iteration to set AOI results and collect IP/CameraContext pairs for sorting
for (size_t i = 0; i < cameras.GetSize(); ++i)
{
    ....

    // Collect pairs of IP address and camera context to be sorted; reverse endian to proprly use the map auto sorting
    vIPAdresses.insert(pair<unsigned long, intptr_t>
        (ntohl(inet_addr(((CBaslerGigEDeviceInfo&)(cameras[i].GetDeviceInfo())).GetIpAddress())), // string > binary > change
endian
        cameras[i].GetCameraContext()));
}

....
// Get ordered GigE camera map by their IP address and assign context value accordingly.
intptr_t contextRunner = 0;
for (std::map<unsigned long, intptr_t>::iterator it = vIPAdresses.begin(); it != vIPAdresses.end(); ++it)
{
    for (int i = 0; i != cameras.GetSize(); i++)
    {
        if (ntohl(it->first) == inet_addr(((CBaslerGigEDeviceInfo&)(cameras[i].GetDeviceInfo())).GetIpAddress()))
        {
            cameras[i].SetCameraContext(contextRunner);

            ....

        }
    }
    contextRunner++;
}
```

Parameterization

```
/* The hardware trigger configuration handler replaces the default configuration
as all currently registered configuration handlers are removed by setting the registration mode to RegistrationMode_ReplaceAll.
However, as chosen either cameras are free running or
hardware trigger setup is used to cause all cameras to grab images synchronously. */
// contextRunner 0 since first camera has master trigger setup (encoder, frame)
if (isHardwareTrigger && contextRunner == 0)
    cameras[i].RegisterConfiguration(new CHardwareTriggerConfiguration(CHwCfgType::Master), RegistrationMode_ReplaceAll,
Cleanup_Delete);
else if (isHardwareTrigger)
    cameras[i].RegisterConfiguration(new CHardwareTriggerConfiguration(CHwCfgType::Slaves), RegistrationMode_ReplaceAll,
Cleanup_Delete);

// The chunk configuration handler is append to the current configuration
// as all currently registered configuration handlers are untouched by setting the registration mode to RegistrationMode_Append.
cameras[i].RegisterConfiguration(new CChunkConfiguration, RegistrationMode_Append, Cleanup_Delete);
```

Grab

All cameras should record images in a manner akin to the sample program "Grab_MultipleCameras" in the SDK. To this end, n threads launch to serve an event handler.

```
// For tread loops, register sample image event handler.
cameras[i].RegisterImageEventHandler(&splitImageHandler, RegistrationMode_Append, Cleanup_None);

....

/*
Start the grabbing using the grab loop thread, by setting the grabLoopType parameter
to GrabLoop_ProvidedByInstantCamera. The grab results are delivered to the image event handlers.
The GrabStrategy_OneByOne default grab strategy is used. */
cameras.StartGrabbing(GrabStrategy_OneByOne, GrabLoop_ProvidedByInstantCamera);
```


Evaluation

The CsplitImageEventHandler takes over every image, splits it by line, if split factor > 1, and fuses the resulting camera images with one another into a composite image.

```
// The image event handler receiving results and splits buffer.
class CSplitImageEventHandler : public CImageEventHandler

{
    // Configuration parameter
    int cameraCount;
    int countImages;

public:
    CSplitImageEventHandler()
        : m_waitObject(WaitObjectEx::Create(false))
    {
    }

    void setter(int numCamerasInUse, int splitFactor)
    {
        cameraCount = numCamerasInUse;
        countImages = splitFactor;
    }

    // Essential parameters must be valid.
    bool ParameterCheck() { if (cameraCount > 0 && countImages > 0) return true; else return false; }

    virtual void OnImageGrabbed(CInstantCamera& camera, const CGrabResultPtr& ptrGrabResult)
    {

        ....

    }
    ....
}

....
// Create image handler. Must be before the CInstantCameraArray.
CSplitImageEventHandler splitImageHandler;
....
// Forward neccessary parameter after array provides valid size.
splitImageHandler.setter(cameras.GetSize(), splitFactor);
....

while (splitImageHandler.GetResultReadyEvent().Wait(5000))
{
    if (_kbhit()) break;
    std::vector<CPylonImage> fullImages;
    if (splitImageHandler.GetResult(fullImages))
    {
        for (auto itImages = fullImages.begin(); itImages != fullImages.end(); ++itImages)
        {
            size_t index = itImages - fullImages.begin();
            DisplayImage(index, *itImages);
        }
    }
    else
    {

        // handle error here .....................

    }
}
```


Change log

Document number	Date	Changes
AW00131101001	12 January 2015	First version created.
AW00131102000	13 Jan 2015	Corrected document number in header and of pdf file name.

Basler AG
Germany, Headquarters
Tel. +49 4102 463 500
Fax +49 4102 463 599

sales.europe@baslerweb.com

www.baslerweb.com

USA
Tel. +1 610 280 0171
Fax +1 610 280 7608

sales.usa@baslerweb.com

Asia
Tel. +65 6367 1355
Fax +65 6367 1255

sales.asia@baslerweb.com