WARNING
VSPM-6F2113 (Inspector PIM60), VSPM-6B2113 (Inspector PIM60 Base), VSPM-6F2113S19 (Inspector PIM60 Bead)

The Inspector is equipped with a LED illumination that must be considered as a lamp system of Risk Group 1 (low risk) according to IEC 62471:2006
Accessible irradiance at distances > 200 mm:
\[ L_B < 4 \times 10^4 \text{ W/}(m^2 \text{ sr}) \text{ within 100 s} \]
\[ L_H < 10^6 \text{ W/}(m^2 \text{ sr}) \text{ within 10 s} \]

WARNING: OPTICAL RADIATION DO NOT STARE INTO BEAM
RISK GROUP 1 (LOW RISK) according to IEC 62471:2006
Visible LED light \( \lambda = 400 - 800 \text{ nm} \)

VSPM-6F2313 (Inspector PIM60-LUT), VSPM-6F2313S20 (Inspector PIM60-LUT Bead)

The Inspector is equipped with a LED illumination that must be considered as a lamp system of Risk Group 1 (low risk) according to IEC 62471:2006
Accessible irradiance at distances > 200 mm:
\[ E_S < 3 \times 10^{-3} \text{ W/m}^2 \text{ within 10^4 s} \]
\[ E_UVA < 33 \text{ W/m}^2 \text{ within 300 s} \]
\[ L_R < 7 \times 10^6 \text{ W/}(m^2 \text{ sr}) \text{ within 10 s} \]

WARNING: OPTICAL RADIATION DO NOT STARE INTO BEAM
RISK GROUP 1 (LOW RISK) according to IEC 62471:2006
UV LED light \( \lambda = 385 \text{ nm} \)

VSPM-6F2413 (Inspector PIM60-IR), VSPM-6B2413 (Inspector PIM60-IR Base), VSPM-6F2413S18 (Inspector PIM60-IR Bead)

The Inspector is equipped with an LED illumination that must be considered as a lamp system of Risk Group 0 / Free Group (exempt risk) according to IEC 62471:2006
Accessible irradiance at distances > 200 mm:
\[ E_IR < 100 \text{ W/m}^2 \text{ within 10^3 s} \]
\[ L_IR < 1.2 \times 10^6 \text{ W/}(m^2 \text{ sr}) \text{ within 10^3 s} \]

NOTICE: IR EMITTED FROM THIS PRODUCT
RISK GROUP 0 (EXEMPT RISK) according to IEC 62471:2006
IR LED light \( \lambda = 850 \text{ nm} \)

DISCLAIMER
SICK uses standard IP technology for its products, e.g. IO Link, industrial PCs. The focus here is on providing availability of products and services. SICK always assumes that the integrity and confidentiality of data and rights involved in the use of the above-mentioned products are ensured by customers themselves. In all cases, the appropriate security measures, e.g. network separation, firewalls, antivirus protection, patch management, etc., are always implemented by customers themselves, according to the situation.

©SICK AG 2018-09-05
All rights reserved
8015726/ZPP7/2018-09
Subject to change without notice
Table of Contents

Introduction ................................................................. 6
  1.1 Interfaces overview .................................................. 6
  1.2 Intended readers ..................................................... 6

Interfaces ........................................................................... 7
  2.1 I/O extension box .......................................................... 8
      2.1.1 Physical network connection ...................................... 8
      2.2 Configuration of the IP address on the I/O extension box .... 8
      2.2.1 Basic configuration of the IP address ......................... 9
      2.3 Setup of the I/O extension box in the SOPAS Engineering Tool (ET) application .............................................. 9
      2.3.1 Enabling the I/O extension box ...................................
      2.4 Input and output connections ......................................
      2.4.1 Special conditions during startup ............................. 10
      2.4.2 Connection to the I/O extension box lost during operation .... 11
      2.4.3 Object selection with I/O extension box .................... 11
      2.4.4 Timing issues ....................................................... 11
      2.4.5 Use of the digital outputs for logic ............................ 11
      2.4.6 Change of Modules in the I/O extension box .............. 11
      2.5 Troubleshooting ....................................................... 11
      2.5.1 The I/O LED flashes 10 times ...................................
      2.5.2 No contact with the I/O extension box ....................... 11
      2.5.3 High number of unanswered requests to the I/O extension box ............................... 12

Web interface ...................................................................... 13
  3.1 Introduction .................................................................... 13
  3.2 Get results via Web API .................................................. 13
  3.2.1 Live image ................................................................ 13
  3.2.2 Detailed results ......................................................... 13
  3.2.3 Synchronize live image with result ............................... 14
  3.2.4 Logged images .......................................................... 14
  3.2.5 Statistics .................................................................. 15
  3.3 Control the sensor via Web API ....................................... 15
  3.3.1 Basic principles ......................................................... 15
  3.3.2 Command syntax ....................................................... 15
  3.3.3 Current reference object .......................................... 16
  3.3.4 Backup and restore configuration ............................... 16
  3.4 Create custom web pages ................................................. 17
  3.4.1 Example: Display live image ..................................... 19
  3.5 Handle the Web API ....................................................... 21

Ethernet Raw ...................................................................... 22
  4.1 Introduction .................................................................... 22
  4.1.1 Port interval ............................................................. 22
  4.2 Get results via Ethernet Raw .......................................... 22
  4.2.1 TCP versus UDP ....................................................... 22
  4.2.2 ASCII versus binary .................................................. 22
  4.2.3 Attributes ................................................................. 23
  4.2.4 Example formatting strings ....................................... 23
  4.3 Control the sensor via Ethernet Raw ............................... 26
  4.3.1 Basic principles ......................................................... 26
  4.3.2 Command syntax ....................................................... 26
## EtherNet/IP

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Introduction</td>
<td>28</td>
</tr>
<tr>
<td>5.2 Get results via EtherNet/IP</td>
<td>28</td>
</tr>
<tr>
<td>5.2.1 Attributes</td>
<td>28</td>
</tr>
<tr>
<td>5.2.2 Example formatting strings</td>
<td>28</td>
</tr>
<tr>
<td>5.3 Control the sensor via EtherNet/IP</td>
<td>33</td>
</tr>
<tr>
<td>5.3.1 Basic principles</td>
<td>34</td>
</tr>
<tr>
<td>5.3.2 Command syntax</td>
<td>34</td>
</tr>
<tr>
<td>5.3.3 Select reference object</td>
<td>34</td>
</tr>
<tr>
<td>5.3.4 Image triggering</td>
<td>35</td>
</tr>
<tr>
<td>5.3.5 Input assemblies, result channel</td>
<td>35</td>
</tr>
<tr>
<td>5.3.6 Assemblies command channel</td>
<td>36</td>
</tr>
</tbody>
</table>

## Appendix

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1 XML based formatting</td>
<td>39</td>
</tr>
<tr>
<td>A.2 XML formatting</td>
<td>39</td>
</tr>
<tr>
<td>A.3 Container specific tags</td>
<td>40</td>
</tr>
<tr>
<td>A.3.1 General tags</td>
<td>49</td>
</tr>
<tr>
<td>A.3.2 Attributes</td>
<td>50</td>
</tr>
<tr>
<td>B.1 Command syntax</td>
<td>52</td>
</tr>
<tr>
<td>B.1.1 Commands ID numbers for EtherNet/IP</td>
<td>53</td>
</tr>
<tr>
<td>B.2 Command channel index handling</td>
<td>54</td>
</tr>
<tr>
<td>B.2.1 Introduction</td>
<td>54</td>
</tr>
<tr>
<td>B.2.2 Blob indexing</td>
<td>54</td>
</tr>
<tr>
<td>B.2.3 Bead indexing</td>
<td>54</td>
</tr>
<tr>
<td>B.2.4 Polygon indexing</td>
<td>54</td>
</tr>
<tr>
<td>B.2.5 Tools indexing</td>
<td>54</td>
</tr>
<tr>
<td>B.3 Command descriptions</td>
<td>54</td>
</tr>
<tr>
<td>B.4 Error codes</td>
<td>74</td>
</tr>
<tr>
<td>B.5 Version information</td>
<td>77</td>
</tr>
<tr>
<td>B.6 Command examples</td>
<td>77</td>
</tr>
<tr>
<td>B.6.1 Command examples Ethernet Raw</td>
<td>77</td>
</tr>
<tr>
<td>C.1 Restore configuration</td>
<td>79</td>
</tr>
<tr>
<td>C.2 Create session cookie</td>
<td>79</td>
</tr>
<tr>
<td>C.3 Login</td>
<td>79</td>
</tr>
<tr>
<td>C.4 Prepare restore mode</td>
<td>80</td>
</tr>
<tr>
<td>C.5 Transfer restore file to device</td>
<td>80</td>
</tr>
<tr>
<td>C.6 Device restart</td>
<td>80</td>
</tr>
</tbody>
</table>

Index ........................................................................................................... 81
Introduction
1 Introduction

The Reference Manual is a complement to the Operating Instructions for Inspector PIM60 and covers the functionality of all product variants.

The Reference Manual contains detailed information about the interfaces including syntax and available functionality. It focuses on Inspector PIM60 specific topics and does not describe the basic technology behind each interface.

The details of the result output formatting and the contents and syntax of the command channel are shared by several interfaces. They are described in an appendix valid for all relevant interfaces.

For instructions on configuring the interfaces, refer to the Operating Instructions.

1.1 Interfaces overview

The Reference Manual contains detailed information for the following interfaces:

- **I/O Extension Box** is used to increase the number of available input and output connections
- **Web API** interface is intended for integration with external HMI implementations, and for customized web pages on the Inspector
- **Ethernet Raw** interface is intended for integration with external PLC equipment
- **EtherNet/IP** interface is intended for integration with external PLC equipment following the EtherNet/IP communication standard

1.2 Intended readers

The intended readers of the Reference Manual are users working with integration between the Inspector PIM60 and other equipment, for example PLC programmers and customized human machine interface (HMI) developers.

The readers are assumed to have knowledge about the Inspector PIM60 product and features as described in the Operating Instructions for Inspector PIM60. The readers are also assumed to have knowledge about the basic functionality of the technology of the interfaces used for the integration.
Interfaces
The Inspector PIM60 can be connected to an I/O extension box that increases the number of digital inputs and outputs. The I/O Extension box is available as an accessory from SICK. This section covers how the I/O extension box is connected to the Inspector, and how it is configured.

The following basic steps are required to use the I/O extension box with the Inspector. Details about the steps are found in the subsequent sections.

1. Connect the I/O extension box to the network.
2. Configure the IP address of the I/O extension box to match the settings of the network, and the Inspector.
3. Enter the IP address of the I/O extension box in the SOPAS Engineering Tool (ET) application.
4. Activate the inputs and/or outputs on the I/O extension box depending on the application.

**Note**

The SOPAS Engineering Tool (ET) application should be closed or set to offline when the power to the I/O box is disconnected. The I/O extension box needs to be restarted if the IP address is changed or if the connections to the inputs and outputs on the box are changed.

### 2.1 Physical network connection

To minimize network latency, it is recommended that the I/O extension box is connected directly to the Inspector. The I/O box has a network switch so that a PC running SOPAS Engineering Tool (ET) can be connected via the I/O box.

**Figure 2.1 Physical network connection**

### 2.2 Configuration of the IP address on the I/O extension box

This section briefly describes how to configure the I/O extension box for operation with the Inspector. For details, please refer to the user manual delivered with the I/O extension box. The IP address of the I/O extension box must be compatible with the addresses of the Inspector and of the PC. For details of how to set and view the IP address of the Inspector, please refer to the Operating Instructions for Inspector PIM60.

The following is an example of how the IP addresses can be configured for the Inspector, the I/O box and the PC.

<table>
<thead>
<tr>
<th>Inspector PIM60</th>
<th>I/O Extension Box</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.1.110</td>
<td>192.168.1.3</td>
<td>192.168.1.30</td>
</tr>
</tbody>
</table>
2.2.1 Basic configuration of the IP address

The address selection switch on the I/O extension box configures the host part of the IP address, that is, the last of the four parts of the IP address. By default, the first three parts of the address (also known as the network address) are set to 192.168.1. If the switch is set to a value other than 0 (all switches set to Off) or 255 (all switches set to On), the I/O extension box will use the host part of the IP address assigned by the switch.

Figure 2.2 Example

The setting above configures the I/O extension box to have a host ID of 3 corresponding to the binary value “00000011” where switch 1 is bit 0 (LSB) and switch 8 is bit 7 (MSB). The I/O box will then have an IP address of 192.168.1.3.

Advanced configuration of the IP address

If the network part of the IP address must be changed from the default 192.168.1 for the I/O extension box, the internal web server of the I/O extension box can be used. For details please refer to the manual delivered with the I/O extension box.

2.3 Setup of the I/O extension box in the SOPAS Engineering Tool (ET) application

The communication with the I/O extension box is configured using the Interfaces and I/O Settings dialog from the InspectorPIM60 menu. Check the Digital I/O and I/O extension boxes in the Interfaces tab. The I/O extension box is disabled if EtherNet/IP is selected in the same tab.

Communication mode

It is possible to adjust the way that the Inspector is communicating with the I/O extension box. The settings are made in the I/O extension box setup tab in the Interfaces and I/O Settings dialog from InspectorPIM60 menu. There are three modes available:

- Robust mode. This is the default communication mode, and it is the recommended one if the Inspector is connected to the SOPAS Engineering Tool (ET) application during operation.
- Fast mode. This mode allows the Inspector to operate at a higher frame rate but there is a risk that some data in the communication with the I/O extension box is lost if there is high load on the network. This mode shall not be used if the Inspector is connected to the SOPAS Engineering Tool (ET) application during operation.
- User mode. This is the advanced communication mode where it is possible to configure the number of retries that the Inspector performs, and the timeout for each retry. The timeout is the time (in milliseconds) that the Inspector is waiting for a reply from the I/O extension box for a request to set outputs or read inputs.
IP configuration
To be able to connect to the I/O extension box, the IP address of the I/O extension box must be specified in the SOPAS Engineering Tool (ET) application.
To specify the IP address of the I/O extension box:

1. Open the Interfaces and I/O Settings dialog from the InspectorPIM60 menu. Enter the selected IP address of the I/O extension box setup tab in the four fields separated with dots.
2. Click Apply to store the settings.

![Set up mode and IP address](image)

Verify connection
It is possible to verify that the connection to the I/O extension box can be established by clicking the Verify connection button. The SOPAS Engineering Tool (ET) application will then try to connect to the I/O extension box, and a message will be displayed informing if the I/O extension box was found.

Note
It is possible to configure the Inspector for use with the I/O extension box even when the I/O extension box is not available. As soon as the Inspector detects the I/O extension box on the network, it will connect to it and start using it as configured.

2.3.1 Enabling the I/O extension box
The use of the inputs and outputs on the I/O extension box is enabled on the Digital I/O tab of the Interface and I/O Settings dialog from the InspectorPIM60 menu.

2.4 Input and output connections
When delivered, the I/O extension box contains 4 digital inputs and 8 digital outputs. The digital outputs can be expanded to 16, and the digital inputs of the Inspector PIM60 can be extended to 5.
Please refer to the manual delivered with the I/O extension box for details on how to connect the power supply to the box, and where to find the digital inputs and outputs.

2.4.1 Special conditions during startup
The following applies if the Inspector is configured to use the inputs of the I/O extension box for object selection:
If the I/O extension box is not available when the Inspector starts up, the Inspector will use the last reference object that was selected in the SOPAS Engineering Tool (ET) application before
data was saved to flash. Once the I/O extension box is available, the Inspector will read the inputs of the box, and select the corresponding reference object.

2.4.2 Connection to the I/O extension box lost during operation
If the connection to the I/O extension box is lost during operation, the last status of the inputs on the box will be used until the connection is re-established.

2.4.3 Object selection with I/O extension box
The status of the inputs on the I/O extension box is checked in the end of each inspection cycle. If the Inspector is configured to use external image trig, the status of the external inputs will only be checked when an image trig signal has been received.

2.4.4 Timing issues
The digital outputs on the I/O extension box shall be read at minimum delay time as displayed in the SOPAS Engineering Tool (ET) application.

2.4.5 Use of the digital outputs for logic
The digital outputs on the I/O extension box are not guaranteed to be jitter-free. It is not recommended to use these outputs for direct control of other devices. The I/O extension box shall be connected to a PLC for process control.

2.4.6 Change of Modules in the I/O extension box
The Inspector PIM60 supports I/O extension box configurations with up to 16 digital outputs and 5 digital inputs (The standard configuration of the I/O extension box contains 8 digital outputs and 4 digital inputs.). The configuration of an I/O extension box can be changed by adding/removing I/O modules to/from the I/O extension box. I/O modules are available as an accessory from SICK. For details about Accessories Ordering Information see the Operating Instructions for Inspector PIM60.

Perform the following steps to connect and use more I/O modules:
1. Close the SOPAS Engineering Tool (ET) application.
2. Disconnect the power from the I/O extension box.
3. Connect the additional I/O modules (inputs and/or outputs) to the I/O extension box. Please refer to the manual delivered with the I/O extension box for details.
4. Re-connect the power to the I/O extension box.
5. Re-start the SOPAS Engineering Tool (ET) application.

The additional digital outputs are now be available in the SOPAS Engineering Tool (ET) application.

2.5 Troubleshooting

2.5.1 The I/O LED flashes 10 times
If the power to the I/O extension box has been disconnected for a longer period of time, the internal clock in the box will be reset. The I/O LED on the box will then flash 10 times in red. This is not a serious error, and the I/O extension box can still be used together with the Inspector without any problems.

2.5.2 No contact with the I/O extension box
Ensure that the network card on the PC has the same network address, for instance 192.168.1, as the I/O extension box. The host part of the IP address (that is the last number in the IP address) must not be the same as for the I/O extension box or the Inspector.

There are two tools available in Windows to check the network connection and the IP settings:
• Ping. Open the command prompt, and type ping followed by the IP address of the I/O extension box. If the I/O extension box is available the following text will be displayed: Reply from x.x.x.x (where x.x.x.x is the IP address of the I/O extension box). If the I/O extension box
could not be found an error message is displayed, for instance Request timed out or Destination host unreachable.
Example: ping 192.168.1.3

- Ipconfig. Open the command prompt and type ipconfig. The current status for the network cards on the PC will then be displayed. Ensure that the network settings are corresponding to the setting for the I/O extension box. The current IP address for the Inspector can be viewed by selecting Device Info from the InspectorPIM60 menu.

The web browser on the PC must be configured not to use a proxy when communicating with the web server in the I/O extension box.

### 2.5.3 High number of unanswered requests to the I/O extension box

The advanced communication mode, User mode, can be used to fine tune the communication with the I/O extension box. It is recommended to try to increase the timeout as a first step, and if this does not work, try to increase the number of retries. Increasing the number of retries will reduce the inspection speed.

If the problem persists even if the timeout and the number of retries have been increased, verify that the network topology does not block the use of UDP packets.
3 Web interface

3.1 Introduction

The Inspector PIM60's web interface can be used in two different ways:

- Users can use a web browser to open web pages served by the Inspector's built-in web server.
  The Inspector PIM60 is delivered with a set of web pages for handling the Inspector, but you can also create custom web pages that can be accessed through the Inspector's built-in web server. When you create custom web pages, all functions available through the Web API can be used.

- Custom applications running on separate systems can use the Web API to directly retrieve images and results, and retrieve and set parameters on the Inspector.

Details on how to manage and access the web pages served by the Inspector is described in the Operating Instructions for Inspector PIM60.

Note

All URLs on the Inspector are case sensitive. For example, trying to use /LiveImage.jpg to retrieve the live image will not work.

3.2 Get results via Web API

3.2.1 Live image

The live image can be retrieved through the Web API by a live image request using the URL:

http://<IP-address>/LiveImage.jpg

The response to the request is a data buffer containing a JPEG image.

If the image is not available, an empty image is returned with a smaller size than a normal image.

Note

The live image is not available when SOPAS is connected to the Inspector PIM60.
Live image response can be much slower when activating the Send to FTP feature.

Example URLs

Request a live image without overlay graphics:

http://192.168.1.110/LiveImage.jpg

The response to the request is a JPEG image.

Request a live image with overlay graphics:


Request a live image with simplified overlay graphics:


3.2.2 Detailed results

The result string, containing the results from the last analyzed image, can be retrieved through the Web API by a request using the URL:

http://<IP-address>/CmdChannel?gRES

The response is a string that has the following syntax:

rgRES <errorCode> <resultString>
The result string is the string that is output over Ethernet, and which is defined in the Ethernet Results Output dialog.

For example, if the current configuration has an object locator and an edge pixel counter, and Ethernet Results Output uses the default example formatting string, the Inspector would return the following string:

```
rgRES 0 194
Image_number: 9639
Object_locator.
Located: 1
Score: 99.00
Scale: 1.00
Position_(X,Y): (237.78,202.05)
Rotation: 20.05

Edge_pixel_counter_1
Decision: 1
Pixels: 724
```

The result is not synchronized with the live image. This means that in some situations, if you retrieve a live image and after that a result string, that string may not contain the result for the retrieved image, but for an image captured and analyzed later. See Section 3.2.3, “Synchronize live image with result” (page 14) for a solution.

### 3.2.3 Synchronize live image with result

To synchronize the live image with the result, you need to assign an id to the image. The id can be up to 16 characters long and can consist of numbers, letters, and special characters.

http://<IP-address>/LiveImage.jpg?id=<ID>

The result for the live image assigned to the <ID> can be listed using the following syntax:

http://<IP-address>/ImageResult?id=<ID>

**Note**

The image might not update if this method is used due to caching, see Section 3.4.1, “Example: Display live image” (page 19) for more information.

**Example URLs**

- Request a live image without overlay graphics and assign an id to the image:
  
  http://192.168.1.110/LiveImage.jpg?id=ID_ABC123

- Request a live image with overlay graphics and assign an id to the image:
  

- Request a live image with simplified overlay graphics and assign an id to the image:
  

### 3.2.4 Logged images

Logged images can be retrieved using the URL:

http://<IP-address>/LogImage.jpg?00

where the argument "00" is the image number. The image number is two digits in the range [00, 29]. The device keeps writing to the log and therefore the log first has to be locked to be able to retrieve an image. This is done by using the URL http://<IP-address>/LockLog

The response to the request is a JPEG image. An empty image with a smaller size than a normal image is returned if no log image is available for a certain position.

To start logging images again the log has to be unlocked first and this is done by using the URL http://<IP-address>/LockLog?Unlock
Example URLs
http://192.168.1.110/LockLog
Retrieve a logged image without overlay graphics:
http://192.168.1.110/LogImage.jpg?00
Retrieve a logged image with overlay graphics:
http://192.168.1.110/LogImage.jpg?00&ShowOverlay
Retrieve a logged image with simplified overlay graphics:
http://192.168.1.110/LogImage.jpg?00&SimplifiedOverlay

3.2.5 Statistics
To retrieve statistics using the command channel, execute the following command:
gSTAT
Statistics can also be retrieved using the URL:
http://<IP-address>/CmdChannel?gSTAT
The response is rgSTAT 0 followed by an XML formatted string. To read the response in an
web browser, change the view in the web browser to reflect the source code.

Note
The statistic response is only for the active reference object.

3.3 Control the sensor via Web API
The Web API supports using the command channel for reading and updating parts of the
device configuration.
The Web API also supports the functionality to do a backup of the device configuration to a
file and to restore the configuration again. This is a convenient way to handle configurations
without installing and using SOPAS Engineering Tool (ET).

3.3.1 Basic principles
The command channel has a set of basic principles:
• Only one command at a time can be executed.
• Inspector PIM60 responds to each command with a response that includes the result of
  the command as well as error codes.
• A specific task to control the Inspector PIM60 includes the command together with its
  parameters, see list of command types and parameters in Appendix B, “Command chan-
  nel” (page 52).
• Writing a parameter can typically only be done when the device is in Edit mode. Reading
  a parameter can be done in both Edit and Run mode.
• It is possible to block configuration changes by deselecting Allow changes via Web Server in
  the Web Server tab in the dialog Interfaces and I/O Settings in InspectorPIM60 menu.

3.3.2 Command syntax
The Web API command channel has the following syntax:
http://<IP-address>/CmdChannel?<command>_<identifier>_<argument 1>_<argument 2>...<argument N>
The ACK message has the following syntax:
<ACK Command> <identifier> <errorCode> <returnValue1> <returnValue2> ... <returnValueN> <errorMessage>
The command is sent as an ASCII string. The combination of a command with its parameters
will either change the device configuration or fetch information from the device. For more
command examples see Section B.1, “Command syntax” (page 52) and Section B.6, “Command examples” (page 77).

Note
The command syntax differs from other interfaces where the initial part http://<IP-address>/CmdChannel? is added and all space characters (" ") are replaced by an underscore character ("_"). The ACK messages still contain spaces.

Example URL
The successful execution of the following command
http://192.168.1.110/CmdChannel?sINT_1_1
will perform the command (to select reference object with index 1) and then return the following string:
rsINT 1 0
while a failed command may return:
rsINT 1 8101 Ref bank index is not used.

3.3.3 Current reference object
The reference image of the current reference object can be retrieved using the URL:
http://<IP-address>/ActiveReferenceImage.jpg
The response to the request is a JPEG image.
The reference image of any reference object in the Inspector can be retrieved using the URL:
http://<IP-address>/getRefObject?0
where the argument "0" is the index of the reference object. The object index that corresponds to each reference object can be found in the Reference object list in the Main view.
The response to the request is a JPEG image. An empty image with a smaller size than a normal image is returned if no reference object is available for a certain position.

Example URL
http://192.168.1.110/ActiveReferenceImage.jpg
http://192.168.1.110/getRefObject?1

3.3.4 Backup and restore configuration
It is possible to backup and restore the device configuration through the Web API. This is the same functionality also available through the standard web pages of the Web Server interface.
The backup data contains the device name and reference objects including corresponding inspection and interface settings.

Note
The backup and restore functionality of the Web Server and the Web API corresponds to the Export Sopas Parameter backup and Import Sopas Parameter backup in the Inspector PIM60 menu.
The backup data used by the Web API is saved as .sbp files, which can be imported to and exported from SOPAS Engineering Tool (ET). The Web API can not use .sdv files.

Backup configuration
The URL to export a configuration is http://<IP-address>/backup_config?config1

Example URL:
http://192.168.1.110/backup_config?config1
The result of the request is an .sbp file containing the device configuration. This file can be stored in the file system of the receiving unit and used later in the restore procedure.
The Web Server standard web pages requires a login to perform a backup. A login is not required when doing a backup through the Web API.

**Restore configuration**

The restore operation takes a device configuration created with the backup functionality and replaces the current configuration with the configuration in the backup file.

The operation is a multiple step procedure that requires a login. The details of the procedure is described in Appendix C, “Restore configuration over Web API” (page 79).

The operation may take several minutes and the Inspector PIM60 is automatically restarted after the configuration has been transferred to the Inspector PIM60.

**Warning**

During the restore operation the device is set in a special restore mode only expecting restore operation requests. Operations and requests via other interfaces like field busses, SOPAS Engineering Tool (ET), or other web browsers shall then be avoided since they may interfere with the restore operation.

### 3.4 Create custom web pages

When creating customized web pages to be stored on and served by the Inspector, you use the Web API to display images, retrieve results and settings, and change parameters in the Inspector.

The functions that use the command channel (retrieving results, and getting and setting parameters) returns the result in text strings, which you need to parse in order to extract the information that you are interested in. To make this easier, the Inspector PIM60 provides a JavaScript that you can use in your web pages, and that helps parsing the results.

You use the functions by including the script file `inspector.js` in your page, and then create an `Inspector` object in your own script.

**Note**

The `inspector.js` script uses JQuery, so you need to also include the provided `jquery.js` script. For more information on JQuery, see www.jquery.com.

```html
<html>
<head>
<title>Custom page</title>
<script type="text/javascript" src="/jquery.js"></script>
<script type="text/javascript" src="/inspector.js"></script>

<script type="text/javascript">
  //<!--
  var inspector = new Inspector();
  ...
  //-->
</script>

...</html>
```

The `Inspector` object has methods that correspond to the commands that can be sent over the command channel. The available methods are listed in Table 3.1, “Methods in inspector.js” (page 19). When using these methods, you pass the same arguments as when using the "raw" command channel commands, as described in Appendix B, “Command channel” (page 52).

The Inspector object returns the (parsed) command response through a callback function. To use the response, you define a function that takes a single argument – the response object – and pass that function as an argument to the Inspector's method. In the function you can...
then check whether the command succeeded, and retrieve the information you are interested in.

For example, to select the first reference object in the current configuration (with index 0), you would call:

```javascript
// First, define a callback function that handles the result
function setRefObjResponse(response) {
    if (response.httpStatus != 200 || response.errorCode != 0) {
        alert(response.errorMessage);
    }
}

// Then, call the inspector object's setInt function
// with the arguments:
//   identifier = 1 for "Set reference object"
//   arg1 = 0, for reference object with index 0
function setRefObj() {
    inspector.setInt(1, 0, setRefObjResponse);
}
```

**Tip**

As an experienced JavaScript developer, you would probably define your callback function as an anonymous function directly in the call to the inspector methods:

```javascript
inspector.setInt(1, 0, function(response) {
    if (response.httpStatus != 200 || response.errorCode != 0) {
        alert(response.errorMessage);
    }
});
```

The content of the response object depends on the function that you called, as well as on the outcome of the command. The following attributes are common for all methods:

- **type**
  The response type, which is basically the same as the command type. See Table B.3, “Command ID numbers - for EtherNet/IP” (page 54).

- **errorCode**
  If non-zero, the command failed for some reason. See Section B.4, “Error codes” (page 74).

- **errorMessage**
  A text message that describes the error. Exists only if the command failed.

- **httpStatus**
  The HTTP status code. If this is not set to 200, the HTTP request failed and the Inspector didn't return any result at all.

The following table lists the attributes that are specific to the called method:
Table 3.1  Methods in inspector.js

<table>
<thead>
<tr>
<th>Method</th>
<th>Corresponding command</th>
<th>Response attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>gVER</code></td>
<td>getVersion(callback)</td>
<td>protocolVersion</td>
</tr>
<tr>
<td><code>gMOD</code></td>
<td>getMode(callback)</td>
<td>mode</td>
</tr>
<tr>
<td><code>sMOD</code></td>
<td>setMode(mode, callback)</td>
<td>values[0]</td>
</tr>
<tr>
<td><code>gINT</code></td>
<td>getInt(identifier, arg1, ..., argn, callback)</td>
<td>identifier values[0 ... n]</td>
</tr>
<tr>
<td><code>sINT</code></td>
<td>setInt(identifier, arg1, ..., argn, callback)</td>
<td>identifier values[0 ... n]</td>
</tr>
<tr>
<td><code>gSTR</code></td>
<td>getString(identifier, arg1, ..., argn, callback)</td>
<td>identifier value</td>
</tr>
<tr>
<td><code>aACT</code></td>
<td>performAction(identifier, args, callback)</td>
<td>identifier</td>
</tr>
<tr>
<td><code>TRIG</code></td>
<td>trig(callback)</td>
<td>-</td>
</tr>
<tr>
<td><code>gRES</code></td>
<td>getResult(callback)</td>
<td>resultString</td>
</tr>
<tr>
<td><code>/LockLog</code></td>
<td>lockLog(callback)</td>
<td>-</td>
</tr>
<tr>
<td><code>/LockLog?Unlock</code></td>
<td>unlockLog(callback)</td>
<td>-</td>
</tr>
</tbody>
</table>

The result string returned by the `getResult()` method is the same string that is output over Ethernet, and which is defined in the Ethernet Results Output dialog.

**Note**

Do not make another call to the Inspector before the current call has returned a response. If you do, the current command will be interrupted and will not return any response at all, making it difficult to figure out whether or not the command was performed on the Inspector. Normally, this is not a problem, but if you are using timed triggers, you should make sure that the triggered functions don't interrupt any commands that may currently be performed on the Inspector.

### 3.4.1  Example: Display live image

To display the current live image, simply include the image from `/LiveImage.jpg` on your web page:

```html
...<image src="/LiveImage.jpg"/>
```

This image will not update automatically, so you could add some JavaScript that makes the live image refresh with a certain interval:

```javascript
<script type="text/javascript">
    //<!--
    function refreshLiveImage() {
        var image = document.getElementById("liveImage");
        image.src = "/LiveImage.jpg?ShowOverlay" + (new Date()).getTime();
        // The (new Date()...) is a trick to make the browser
        // retrieve the image from the Inspector and not from cache
        setTimeout(refreshLiveImage, 500);
    }
    //-->
</script>
```
Alternatively, you can use the live image component that is used on the Inspector’s default Live image page, which has automatic refresh, setting for refresh interval, and magnifier.

![Figure 3.1 Inspector’s default live image component](image)

To use the default live image component, include the `userliveimage.js` script in your page, and call the `sickLiveImage()` function as in the following example:

```html
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Live Image</title>
<script type="text/javascript" src="/jquery.js"></script>
<script type="text/javascript" src="/userliveimage.js"></script>
<script type="text/javascript">
//-->
$(document).ready(function () {
  $('#liveImage').sickLiveImage({
    width: 640,
    height: 480,
    nocache: true,
    minInterval: 500,
    refreshInterval: 1000,
    magnifier: true,
    imgUrl: "/LiveImage.jpg",
    refreshText: "Refresh interval"
  });
//-->
</script>
</head>
<body>
<div id="liveImage"></div>
</body>
</html>
```
A Custom Web Toolkit can be found on the support pages (supportportal.sick.com) which provides a framework that simplifies the process of making an HMI as well as additional templates and examples.

### 3.5 Handle the Web API

The Web Server and Web API interfaces can be activated or deactivated. When activated, it is possible to select port number and to allow command channel changes. The same settings apply both to the Web Server and to the Web API. The Web interfaces are configured in the **Interfaces and I/O settings** dialog in the InspectorPIM menu.

The Web API is based on standard HTTP request and responses. Recommended request timeout time is 3 seconds to allow for images to be transferred properly.
4 Ethernet Raw

4.1 Introduction
To set up the connection and output results for Inspector PIM60 using Ethernet Raw see Operating Instructions for Inspector PIM60.

4.1.1 Port interval
The default interval for the ports used by the communication channels is 2114-2116. This interval can be changed, e.g. if the controlling device does not support the default interval. The interval is controlled by the field Start port in the Ethernet Raw tab of the Interface and I/O settings dialog.
The ports are assigned according to the following:
• Ethernet Result Output = start port (default 2114)
• Command channel = start port + 1
• Dedicated image trig = start port + 2

4.2 Get results via Ethernet Raw
The following settings are configured in the Ethernet Result Output dialog under InspectorPIM60 menu.

4.2.1 TCP versus UDP
The basic difference between these protocols, for the Ethernet result output function, is which side initiates the connection to receive/send the data.

TCP:
1. PC/PLC initiates the connection
2. Inspector sends results to the PC/PLC
3. PC/PLC acknowledges that results are received (built into the TCP protocol)

UDP:
1. Inspector sends results to the specified IP address and port, without knowing if it has been received

Note
For TCP the default port number that the Inspector listens to is 2114.

4.2.2 ASCII versus binary
The Inspector supports the possibility to choose whether the configured output is to be sent in ASCII format or in a binary format. The parameters that should be transferred in binary format are also defined in the XML based formatting, but some tags are not supported in the binary format.

If such a parameter is added to the formatting it will be ignored by the Inspector. In binary mode all added text and text formatting, for example `<SPACE/>`, are ignored. Only the values
of the parameters describing the results of inspected images or device information will be sent. For details on which tags can be used in binary output see the tables in chapter Appendix A, “Result output formatting” (page 39).

4.2.3 Attributes

Attributes are used to control the formatting and identification of inspections. Some of them can be controlled directly in the Ethernet Result Output dialog in the section Message settings. All available attributes are listed in the table in section XML Formatting in Section A.3.2, “Attributes” (page 50).

- **Min number of digits**
  Specifies the minimum number of digits (including decimal point) to include in the result. If the value to be sent out has fewer digits, the result is padded with leading zeros. The default setting is 0 which means the number of digits that will be sent will differ depending on how many digits are needed. The maximum number of digits is 9. **Note:** This attribute is only applicable for ASCII

- **Number of decimals**
  Specifies the number of digits to include after the decimal point for values with decimals. The value will be rounded to the specified number of decimals. Default value is 2. The maximum number of decimals is 9. **Note:** This attribute is only applicable for ASCII

- **Degrees/Radians**
  Specifies the unit for the rotation of the object locator, angle for blobs, angle for edges, and angle measurements.

- **Little/Big Endian**
  This specifies the order of the bytes transferred from the device on Ethernet. When using Little endian the least significant byte is transferred first and for Big endian the most significant byte is transferred first. See the 2-byte example in tables below. **Note:** Only applicable when using binary format.

<table>
<thead>
<tr>
<th>Transfer order</th>
<th>First transferred byte</th>
<th>Second transferred byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Little endian</td>
<td>01110000</td>
<td>10000100</td>
</tr>
<tr>
<td>Big endian</td>
<td>10000100</td>
<td>01110000</td>
</tr>
</tbody>
</table>

- **Pixels/Millimeters**
  Specifies whether position coordinates and distance measurements should be expressed in pixels or millimeters. **Note:** The device must be calibrated to be able to use millimeters as unit of measurement.

4.2.4 Example formatting strings

The auto-generated example string will vary depending on the configuration in the selected reference object. The intention with the example string is to give an idea of the available tags and to be a good starting point for creating a suitable format.

Below follow some short descriptions of example strings for different configurations. For more information about the XML formatting see Appendix A, “Result output formatting” (page 39).
Example string for configuration with only an Object locator

```
<Message_Size/>
Image_number:<Space/>Image_Number/>
Object_locator/>
Located:<Space>/Decision/>
Score:<Space>/Score/>
Scale:<Space>/Scale/>
Position_(X,Y):<Space>(<X>,<Y>)
Rotation:<Space>/Rotation/>
</Object_Loc>
```

1. Size of the message, number of characters (ASCII) or bytes (binary)
2. Explanatory text and analyzed images number
3. Explanatory text
4. Start of container for object locator
5. Explanatory text and value for locator decision; 0=not found, 1=found
6. Explanatory text and locator score value. The score value shows in percent how well the located object matches the reference object.
7. Explanatory text and locator scale value, factor of analyzed live image compared to taught reference object
8. Explanatory text and x and y position of the reference point. This can be outside the image and therefore negative. Shown in "pixels" or "mm"
9. Explanatory text and locator rotation, in degrees or radians depending on the configured value in the Ethernet Result Output settings dialog
10. End of container for object locator

Result of validating output string with only an Object locator

The result of validating the example formatting output string with output format ASCII can be as follows:

97
Image_number: 14471
Object_locator.
Located: 1
Score: 96.00
Scale: 1.00
Position_(X,Y): (291.52,238.55)
Rotation: 0.22

The result of validating the example formatting output string for only an object locator with output format binary will be as follows:

Binary output OK. Number of bytes: 27

Part of example string for configuration with a Blob

```
<Blob_tool/>
<Blob index="0" name="Blob 1">
Found_blobs:<Space>/Found_Blobs/>
----------------------------------
Blob_information/>
Position_(X,Y):<Space>(<X>,<Y>)
```

1. Blob_tool
2. Blob
3. Found_blobs
4. Blob_information
5. Position_
Area: <AREA/> Area: 7
Angle: <ANGLE/> Angle: 8
Structure: <EDGE_PIXELS/> Structure: 9
Touches_ROI_border: <EDGE_FLAG/> Touches_ROI_border: 10
</BLOB>

1. Explanatory text
2. Start of container for the blob tool named "Blob 1" and instruction to fetch the first (index="0") blob in accordance with the Sort by criteria
3. Explanatory text and number of found blobs in analyzed image
4. Separator
5. Explanatory text
6. Explanatory text and information of blob with index="0" concerning position and center of gravity (x and y position), in "pixels" or "mm"
7. Explanatory text and blob (index="0") area, in "pixels"
8. Explanatory text and blob (index="0") angle value, in degrees or radians depending on the configured value in the Ethernet Result Output settings dialog
9. Explanatory text and blob (index="0") structure value, number of edge pixels inside the blob
10. Explanatory text and blob (index="0") edge value, 0=blob fully within ROI, 1= blob touches ROI border
11. End of container for blob tool

Result of validating output string with a Blob

The result of validating the example formatting output string with output format ASCII can be as follows:

Blob_tool.
Found_blobs: 16
-------------------------
Blob_information:
Position_(X,Y): (177.00, 156.89)
Area: 75
Angle: 154.33
Structure: 0
Touches_ROI_border: 0

The result of validating the example formatting output string for a blob with output format binary will be as follows:

Binary output OK. Number of bytes: 28

Part of example string for configuration with a Polygon

Polygon1<POLYGON name="Polygon1"> Polygon1
Corners:<NUM_CORNERS/> Corners:
<CORNERS corners="all"> Corners
{X,Y}:<NEWLINE/> {X,Y}<NEWLINE/>
</CORNERS>
</POLYGON>

1. Explanatory text and start of polygon container tag for the polygon tool named "Polygon 1"
2. Explanatory text and number of polygon corners
3. Start of container tag for polygon corners with instruction to loop over all polygon corners, explanatory text, and corner position
4. End of container for polygon corners
Result of validating output string with a Polygon
The result of validating the example formatting output string with output format ASCII will be as follows:

Polygon_1
Corners: 4
(X,Y): (329.15,235.70)
(X,Y): (371.31,235.60)
(X,Y): (372.58,314.97)
(X,Y): (329.82,315.22)

The result of validating the example formatting output string for a polygon with output format binary will be as follows:

Binary output OK. Number of bytes: 39

Example of a JSON formatted string
It is possible to get an example string in JSON format. This is typically used in conjugation with web based HMIs for easier result extraction.

{"MESSAGE": {"IMAGE_NUMBER":"<IMAGE_NUMBER/>"},"Spaces":0,"Pixel_counter_1":{"PIXEL_COUNTER name="Pixel counter 1"},"DECISION":<DECISION/>"
"PIXELS":<PIXELS/>
"<PIXEL_COUNTER>"
}}

{"MESSAGE":{"IMAGE_NUMBER":"2975780","Pixel_counter_1":{"DECISION":"1","PIXELS":5433}}

4.3 Control the sensor via Ethernet Raw
The Inspector has a command channel accessible via the Ethernet Raw interface. The command channel makes it possible to read and write a defined set of configuration parameters, and to trigger image acquisition, via UDP or TCP. This section describes how to setup image triggering and command channel settings in SOPAS Engineering Tool (ET), as well as the syntax of the command channel.

4.3.1 Basic principles
The command channel has a set of basic principles:
• Only one command at a time can be executed.
• Each command is followed by a return message (ACK) that includes result of the command as well as error codes.
• A specific task to control the Inspector PIM60 includes the command together with its parameters, see list of command types and parameters in Appendix B, Command channel (page 52)).
• Writing a parameter can typically only be done when the device is in Edit mode. Reading a parameter can be done in both Edit and Run mode.
• It is possible to block configuration changes by deselecting the setting Allow changes via Ethernet Raw in the Ethernet Raw tab in the dialog Interfaces and I/O Settings in the InspectorPIM60 menu.

4.3.2 Command syntax
The commands have the following syntax:

JSON (JavaScript Object Notation) in accordance with RFC 4627
The ACK message has the following syntax:

```
<STX><ACK Command> <identifier> <errorCode> <returnValue1> <returnValue2>
... <returnValueN> <errorMessage><ETX>
```

where `<STX>` and `<ETX>` are the START OF TEXT and END OF TEXT characters (ASCII codes 2 and 3 respectively).

The command is sent as an ASCII string. The combination of a command with its parameters will either change the device's configuration or fetch information from the device. For more command examples see Section B.6, “Command examples” (page 77) and Section B.1, “Command syntax” (page 52).

### 4.3.3 Select reference object

To enable reference object selection via **Ethernet Raw** do the following:

1. Choose **Interface and I/O Settings** from the **InspectorPIM60** menu.
2. In the **Interface** tab choose **Ethernet** and **Ethernet Raw** in the listbox.

To select reference object via the command channel, use the command `sINT 1 <object index>`. The object index that corresponds to each reference object is shown in the **Reference objects** list in the **Main view**.

### 4.3.4 Image triggering

It is possible to trigger image acquisition via Ethernet. The communication runs on UDP or TCP port 2116 (configurable). In order to use this function the triggering has to be enabled in **SOPAS Engineering Tool (ET)**. In the **InspectorPIM60** menu and **Interfaces and I/O settings** dialog check the **Ethernet** box and in the list **Ethernet Raw** in the **Interfaces** tab. For the selected reference object, choose **Triggered by Ethernet** in the **Image settings** tab.

### 4.3.5 Single port solution

In real-time applications, the Inspector is controlled using three ports. However, it is possible to use only the command port (default 2115) to control the sensor. The single port solution is only recommended for applications where the cycle time is significantly larger than the image analysis time. One reason for this is that the image acquisition has a lower priority on the command port. Another reason is that the Ethernet Result string must be retrieved from the sensor, therefore image trig and result handling cannot be performed in parallel when using the single port solution.

This is how the Inspector is controlled by using only the command port:

- The image acquisition is performed by the TRIG command (with lower priority).
- The Ethernet Result Output string is retrieved explicitly by the controlling device, e.g. a PLC. This is done by the command `gRES`. The sensor does not send the result automatically on this port.
- All other commands on the command channel are available as in the standard three port solution.
5 EtherNet/IP

5.1 Introduction

The Inspector PIM60 can be controlled and results retrieved using the EtherNet/IP™ standard, see http://www.odva.org/.

To be able to use EtherNet/IP, the EtherNet/IP option has to be enabled and the connection and output result setup has to be made, see the Operating Instructions for Inspector PIM60.

5.2 Get results via EtherNet/IP

The following settings are configured in the Ethernet Result Output dialog in the InspectorPIM60 menu.

5.2.1 Attributes

Attributes are used to control the formatting and identification of inspection results. Some of them can be controlled directly in the Ethernet Result Output dialog in the section Message settings. All available attributes are listed in the table in section XML Formatting in Section A.3.2, “Attributes” (page 50).

- **Degrees/Radians**: Choose unit for the rotation for object locator, angle for blobs, angle for edges, and angle measurements.
- **Pixels/Millimeters**: Choose if position coordinates and distance measurements should be sent in pixel or millimeter unit.

**Note:** The device must be calibrated for it to be possible to use the “mm” attribute. An error message is given in the output string if the device is not calibrated and mm is chosen.

5.2.2 Example formatting strings

The auto-generated example string will vary depending on the configuration in the selected reference object. The intention with the example string is to give an idea of the available tags and to be a good starting point for creating a suitable format.

Below follow some short descriptions of example strings for different configurations. For more information about the XML formatting see Appendix A, “Result output formatting” (page 39).

**Example string for configuration with only an Object locator**

```xml
<IMAGE_NUMBER dataType="DINT" pos="0"/>
<Object_LOC>
<DECISION dataType="SINT" pos="0"/>
<SCORE dataType="REAL" pos="0"/>
<SCALE dataType="REAL" pos="1"/>
<X dataType="REAL" pos="2"/>
<Y dataType="REAL" pos="3"/>
<ROTATION dataType="REAL" pos="4"/></Object_LOC>
```

1. Analyzed image’s number
2. Start of container for object locator
3. Decisions reports whether the object was found (=1) or not found (=0)
Score expressed in percent how well the taught object is matched against the live image.
Scale is the factor of analyzed live image compared to taught reference object.
Position (x) of the reference point of the object locator.
Position (y) of the reference point of the object locator.
Rotation of the object locator, in degrees or radians depending on the configured value in the Ethernet Result Output dialog.
End of container for object locator.

Attribute `dataType` specifies the data type to use for this result. When using EtherNet/IP the attribute `dataType` specifies the `dataType` section in the selected assembly. The attribute can be SINT, INT, DINT or REAL. For more details about `dataType` and `pos` see table in Section A.3.2, “Attributes” (page 50).

Attribute `pos` used by EtherNet/IP to determine a position in the `dataType` section in the selected assembly. The first position number of the `dataType` section is 0. The range depends on which assembly is used. For example if assembly 1 and `dataType` section SINT is selected the range of position is [0, 7].

The combination of `dataType` and `pos` determine which parameter the result will be mapped to. For more details about `dataType` and `pos` see table in Section A.3.2, “Attributes” (page 50).

Result of validating output string with only an Object locator
The validating in SOPAS Engineering Tool (ET) will give the following result:

`EtherNet/IP assembly string OK.`

Result in PLC with only an Object locator
The table below describes how the Assembly 1’s data structure will be populated when using the configuration example above.

<table>
<thead>
<tr>
<th>Position ref (pos) from XML configuration</th>
<th>Data type (dataType)</th>
<th>Offset byte</th>
<th>Variable from example above</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SINT</td>
<td>0</td>
<td>DECISION</td>
</tr>
<tr>
<td>1</td>
<td>SINT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SINT</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SINT</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SINT</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>SINT</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>SINT</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>SINT</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>INT</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>INT</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>INT</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>INT</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>INT</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>INT</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>INT</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>INT</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>DINT</td>
<td>24</td>
<td>IMAGE_NUMBER</td>
</tr>
<tr>
<td>1</td>
<td>DINT</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>Position ref (pos) from XML configuration</td>
<td>Data type (dataType)</td>
<td>Offset byte</td>
<td>Variable from example above</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>----------------------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>2</td>
<td>DINT</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>DINT</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>DINT</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>REAL</td>
<td>44</td>
<td>SCORE</td>
</tr>
<tr>
<td>1</td>
<td>REAL</td>
<td>48</td>
<td>SCALE</td>
</tr>
<tr>
<td>2</td>
<td>REAL</td>
<td>52</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>REAL</td>
<td>56</td>
<td>Y</td>
</tr>
<tr>
<td>4</td>
<td>REAL</td>
<td>60</td>
<td>ROTATION</td>
</tr>
</tbody>
</table>

Example string for configuration with only a Blob

```xml
<IMAGE_NUMBER dataType="DINT" pos="0"/>
<BLOB index="0" name="Blob 1">
<FOUND_BLOBS dataType="SINT" pos="0"/>
<X dataType="REAL" pos="0"/>
<Y dataType="REAL" pos="1"/>
<AREA dataType="DINT" pos="1"/>
<ANGLE dataType="REAL" pos="2"/>
<EDGE_PIXELS dataType="DINT" pos="2"/>
<EDGE_FLAG dataType="SINT" pos="1"/>
</BLOB>
```

1. Analyzed image's number, attributes dataType and pos
2. Start of container for blob, Index number of the found blob according to current blob sorting order. Index 0 is the first blob. Name refers to the blob tool's name in the Tools tab
3. Number of found blobs
4. Blob center of gravity (x position), "pixels" or "mm"
5. Blob center of gravity (y position), "pixels" or "mm"
6. Blob area in pixels
7. Angle of the blob, in degrees or radians depending on the configured value in the Ethernet Result Output dialog
8. Structure value (number of edge pixels inside the blob)
9. Edge flag: 0= the blob is fully within the ROI, 1=the blob touches ROI border
10. End of container for Blob

Attribute dataType

Casts to the specified datatype. When using EtherNet/IP the attribute dataType specifies the dataType section in the selected assembly. The attribute dataType can be SINT, INT, DINT or REAL. For more details about dataType and pos see table in Section A.3.2, "Attributes" (page 50).

Attribute pos

Used by EtherNet/IP to determine a position in the dataType section in the selected assembly. The first position number of the dataType section is 0. The range of the attribute pos depends on which assembly is used. For example if assembly 1 and dataType section SINT is selected the range of position is 8, i.e. [0, 7]. For more details about dataType and pos see table in Section A.3.2, "Attributes" (page 50).
Therefore the value of the attributes dataType and pos together specifies which parameter in the assembly the result value should be mapped to.

**Result of validating output string with only a Blob**
The validating in SOPAS Engineering Tool (ET) will give the following result:

EtherNet/IP assembly string OK.

**Result in PLC with only a Blob**
The table below describes how the Assembly 1's data structure will be populated when using the configuration example above.

<table>
<thead>
<tr>
<th>Position ref (pos) from XML configuration</th>
<th>Data type (dataType)</th>
<th>Offset byte</th>
<th>Variable from example above</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SINT</td>
<td>0</td>
<td>FOUND_BLOBS</td>
</tr>
<tr>
<td>1</td>
<td>SINT</td>
<td>1</td>
<td>EDGE_FLAG</td>
</tr>
<tr>
<td>2</td>
<td>SINT</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SINT</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SINT</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>SINT</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>SINT</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>SINT</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>INT</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>INT</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>INT</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>INT</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>INT</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>INT</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>INT</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>INT</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>DINT</td>
<td>24</td>
<td>IMAGE_NUMBER</td>
</tr>
<tr>
<td>1</td>
<td>DINT</td>
<td>28</td>
<td>AREA</td>
</tr>
<tr>
<td>2</td>
<td>DINT</td>
<td>32</td>
<td>EDGE_PIXELS</td>
</tr>
<tr>
<td>3</td>
<td>DINT</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>DINT</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>REAL</td>
<td>44</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>REAL</td>
<td>48</td>
<td>Y</td>
</tr>
<tr>
<td>2</td>
<td>REAL</td>
<td>52</td>
<td>ANGLE</td>
</tr>
<tr>
<td>3</td>
<td>REAL</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>REAL</td>
<td>60</td>
<td></td>
</tr>
</tbody>
</table>

**Part of default string for configuration with only a Polygon**

```xml
<IMAGE_NUMBER dataType="DINT" pos="0"/>
<POLYGON name="Polygon 1">
  <NUM_CORNERS dataType="SINT" pos="0"/>
  <CORNERS corners="0"/>
</POLYGON>
```

©SICK AG • Identification & Measuring • www.sick.com • All rights reserved
Subject to change without notice
Analyzed image's number, attributes `dataType` and `pos`  
Start of container for `Polygon`, Name refers to the `Polygon` tool's name in the `Tools` tab.  
Number of corners used for this `Polygon` tool.  
Number 0 to 15 gives the properties of a single corner. The index of this corner is the order in which the polygon corner was added when the polygon was drawn.  
`Polygon` corner coordinate (x), "pixels" or "mm",  
`Polygon` corner coordinate (y), "pixels" or "mm",  
End of tag for corners.  
End of container for `Polygon`.

Attribute `dataType`  
Casts to the specified datatype. When using EtherNet/IP the attribute `dataType` specifies the `dataType` section in the selected assembly. The attribute `dataType` can be SINT, INT, DINT or REAL. For more details about `dataType` and `pos` see table in Section A.3.2, "Attributes" (page 50).

Attribute `pos`  
Used by EtherNet/IP to determine a position in the `dataType` section in the selected assembly. The first position number of the `dataType` section is 0. The range of the attribute `pos` depends on which assembly is used. For example if assembly 1 and `dataType` section SINT is selected the range of position is 8, i.e. [0, 7]. For more details about `dataType` and `pos` see table in Section A.3.2, "Attributes" (page 50).

Therefore the value of the attributes `dataType` and `pos` together specifies which parameter in the assembly the result value should be mapped to.

**Result of validating output string with only a `Polygon`**

The validating in SOPAS Engineering Tool (ET) will give the following result:

EtherNet/IP assembly string OK.

If the used assembly is too small the validating will give the following result:

EtherNet/IP assembly string not OK. Out of slots for data type INT.

Use a larger assembly to solve this problem. Choose a larger assembly in the dialog `Interfaces` and I/O settings in the InspectorPIM menu and the EtherNet/IP tab.

**Result in PLC with only a `Polygon`**

The table below describes how the Assembly 1's data structure will be populated when using the configuration example above.

<table>
<thead>
<tr>
<th>Position ref (pos) from XML configuration</th>
<th>Data type (dataType)</th>
<th>Offset byte</th>
<th>Variable from example above</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SINT</td>
<td>0</td>
<td>NUM_CORNERS</td>
</tr>
<tr>
<td>1</td>
<td>SINT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SINT</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SINT</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SINT</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>SINT</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
5.3 Control the sensor via EtherNet/IP

The Inspector PIM60 has the following EtherNet/IP characteristics:

- Device type: Communication adapter
  The Inspector relies on a Scanner device to set up the communication channel. The IP address of the Inspector can be found by choosing Device Info from the InspectorPIM60 menu.

<table>
<thead>
<tr>
<th>Assemblies</th>
<th>Instance no.</th>
<th>Size (bytes)</th>
<th>Comment</th>
<th>Assembly no.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>100</td>
<td>4</td>
<td>Slim command channel</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>101</td>
<td>36</td>
<td>Command channel result</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>102</td>
<td>32</td>
<td>Command channel</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>103</td>
<td>64</td>
<td>Small result channel</td>
<td>1</td>
</tr>
<tr>
<td>Input</td>
<td>105</td>
<td>124</td>
<td>Medium result channel</td>
<td>2</td>
</tr>
<tr>
<td>Input</td>
<td>107</td>
<td>248</td>
<td>Large result channel</td>
<td>3</td>
</tr>
<tr>
<td>Input</td>
<td>109</td>
<td>484</td>
<td>Extra large result channel</td>
<td>4</td>
</tr>
</tbody>
</table>

- Minimum RPI: > 16 ms.
  When retrieving inspection results via EtherNet/IP, the time between two inspections should be at least twice the RPI (Requested Packet Interval) specified for the communication channel.

With the shortest possible RPI, the highest recommended inspection rate is therefore approximately 30 Hz.
The EDS file for the Inspector PIM60 can be found in the SICK Support Portal (support-portal.sick.com).

The Inspector PIM60 has two Output assemblies that can be used for controlling the Inspector. To do this the connection has to be set first, see Operating Instructions for Inspector PIM60. The slim command channel assembly (instance no. 100) is used for controlling the Inspector in the following ways:

- Select reference object
- Image trig

The command channel assembly (instance no. 102) is also used for controlling the Inspector. With this assembly you have access to all functions in the command channel, see Section B.3, “Command descriptions” (page 54).

The two output assemblies are described in detail, see Section 5.3.6, "Assemblies command channel" (page 36).

### 5.3.1 Basic principles

The command channel has a set of basic principles:

- In order to be able to change the configuration via EtherNet/IP this must be enabled. This is done in the dialog Interfaces and I/O Settings from the InspectorPIM60 menu. Check Ethernet and EtherNet/IP in the tab Interfaces. In the same dialog and tab EtherNet/IP check Allow changes via EtherNet/IP.

- It is possible to block configuration changes by deselecting the setting Allow changes via EtherNet/IP in the EtherNet/IP tab in the dialog Interfaces and I/O Settings in InspectorPIM60 menu.

- Writing a parameter can typically only be done when the device is in Edit mode. Reading a parameter can be done in both Edit and Run mode.

- The commands is sent with help of output assembly 102 and the result is received with input assembly 101.

- The result for a sent command can be received at the earliest in the next PLC cycle. The PLC program will have to wait for the result for an undefined number of seconds.

- Make sure that the PLC program waits for a response with the same command and ID as the sent command.

### 5.3.2 Command syntax

To send commands through the command channel use output assembly 102. The command channel has the following syntax:

```
<command> <identifier> <arg 1> <arg 2> <arg 3> <arg 4> <arg 5> <arg 6>
```

Replace `<command>` with the commands id, see Table B.3, “Command ID numbers - for EtherNet/IP” (page 54).

The result of a command, sent over output assembly 102, can be received through input assembly 101. The syntax for ACK message is:

```
<command> <identifier> <error code> <retVal1> <retVal2> <retVal3> <retVal4> <retVal5> <retVal6>
```

The combination of a command with its parameters will either change the devices configuration or fetch information from the device. For more command examples see Section B.1, “Command syntax” (page 52) and Section B.6, “Command examples” (page 77).

### 5.3.3 Select reference object

There are two ways to select reference object with EtherNet/IP and command channel.
The first way to select reference object:

To select the reference object via the slim command channel, change the value of Select reference object in the slim command channel assembly (instance no. 100). The object index that corresponds to each reference object can be found in the Reference object list in the Main view.

If the value in Select reference object does not correspond to any reference object, the Inspector will ignore the attempt to switch reference object.

The second way to select reference object:

To select reference object via command channel change to Edit mode, 0 0, change the value to select the reference object, 2 1 <object index> and then change back to Run mode 0 1 in the command channel assembly (instance no. 102). The object index that corresponds to each reference object can be found in the Reference object list in the Main view.

The time it takes to switch reference object depends on the number of inspections, inspection type, and sizes of the regions in the reference object. Typically it takes in the order of one second to switch reference object. For more information see Operating Instructions for Inspector PIM60.

5.3.4 Image triggering

To enable triggering via EtherNet/IP, do the following:

1. Choose Interfaces and I/O Settings from the InspectorPIM60 menu.
2. In the tab Interface choose Ethernet and EtherNet/IP in the list box.
3. In the Image settings tab choose Trig by EtherNet/IP.

To trigger an image acquisition via EtherNet/IP, specify that the slim command channel (instance no. 100) is to be used here and set the value of Trigger to 1. The image capture is made immediately, without any delays.

The Inspector will capture an image each time the value of Trigger is changed to 1 (i.e. rising edge). To trigger the next image caption, you must first set the value to 0.

When triggering via EtherNet/IP, the time between two image captions should be at least 4 times the RPI. This means that the maximum triggering rate via EtherNet/IP is approximately 15 Hz.

5.3.5 Input assemblies, result channel

There are four input assemblies, each assembly corresponds to respective assembly in the EtherNet/IP tab in the Interfaces and I/O Settings dialog. Each assembly has four different dataType sections, SINT, INT, DINT, and REAL. Each dataType section has a different number of positions, the number of positions depends on the assembly and the dataType selected. Example: The dataType SINT in assembly 1 has 8 positions [0, 7] and the dataType REAL in assembly 4 has 44 positions [0, 43]. The contents of the assembly are defined from the Ethernet Result Output dialog.

Note

On the SICK Support portal (supportportal.sick.com) there is an excel file with templates for the four result input assemblies (file name: AssemblyMappingPI50andPIM60.xls). These can be used to document the mapping between position in data structure and what is configured in the Ethernet Result Output dialog.

Assembly 1 - small result channel

| Instance ID: | 103 |
| Size:        | 64 bytes |
Table 5.1  Input Assembly 1

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Number/ size</th>
<th>Offset (bytes)</th>
<th>Total size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINT</td>
<td>8/ 1 byte each</td>
<td>0</td>
<td>8 bytes</td>
</tr>
<tr>
<td>INT</td>
<td>8/ 2 bytes each</td>
<td>8</td>
<td>16 bytes</td>
</tr>
<tr>
<td>DINT</td>
<td>5/ 4 bytes each</td>
<td>24</td>
<td>20 bytes</td>
</tr>
<tr>
<td>REAL</td>
<td>5/ 4 bytes each</td>
<td>44</td>
<td>20 bytes</td>
</tr>
</tbody>
</table>

Assembly 2 - medium result channel
Instance ID: 105
Size: 124 bytes

Table 5.2  Input Assembly 2

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Number/ size</th>
<th>Offset (bytes)</th>
<th>Total size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINT</td>
<td>12/ 1 byte each</td>
<td>0</td>
<td>12 bytes</td>
</tr>
<tr>
<td>INT</td>
<td>12/ 2 bytes each</td>
<td>12</td>
<td>24 bytes</td>
</tr>
<tr>
<td>DINT</td>
<td>11/ 4 bytes each</td>
<td>36</td>
<td>44 bytes</td>
</tr>
<tr>
<td>REAL</td>
<td>11/ 4 bytes each</td>
<td>80</td>
<td>44 bytes</td>
</tr>
</tbody>
</table>

Assembly 3 - large result channel
Instance ID: 107
Size: 248 bytes

Table 5.3  Input Assembly 3

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Number/ size</th>
<th>Offset (bytes)</th>
<th>Total size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINT</td>
<td>24/ 1 byte each</td>
<td>0</td>
<td>24 bytes</td>
</tr>
<tr>
<td>INT</td>
<td>24/ 2 bytes each</td>
<td>24</td>
<td>48 bytes</td>
</tr>
<tr>
<td>DINT</td>
<td>22/ 4 bytes each</td>
<td>72</td>
<td>88 bytes</td>
</tr>
<tr>
<td>REAL</td>
<td>22/ 4 bytes each</td>
<td>160</td>
<td>88 bytes</td>
</tr>
</tbody>
</table>

Assembly 4 - extra large result channel
Instance ID: 109
Size: 484 bytes

Table 5.4  Input Assembly 4

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Number/ size</th>
<th>Offset (bytes)</th>
<th>Total size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINT</td>
<td>44/ 1 byte each</td>
<td>0</td>
<td>44 bytes</td>
</tr>
<tr>
<td>INT</td>
<td>44/ 2 bytes each</td>
<td>44</td>
<td>88 bytes</td>
</tr>
<tr>
<td>DINT</td>
<td>44/ 4 bytes each</td>
<td>132</td>
<td>176 bytes</td>
</tr>
<tr>
<td>REAL</td>
<td>44/ 4 bytes each</td>
<td>308</td>
<td>176 bytes</td>
</tr>
</tbody>
</table>

5.3.6  Assemblies command channel
The value that corresponds to each reference object can be found in the Reference object list in the Main view.

Slim command channel
The Output assembly contains two parameters that are used for selecting reference object and trigger inspections.
Table 5.5  
**Slim Command channel**

<table>
<thead>
<tr>
<th>Data</th>
<th>Type</th>
<th>Offset (bytes)</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select reference object</td>
<td>SINT</td>
<td>0</td>
<td>0-31: Selected reference object</td>
</tr>
<tr>
<td>Trigger</td>
<td>SINT</td>
<td>1</td>
<td>1: Trigger inspection. Set to 0 before triggering next inspection.</td>
</tr>
<tr>
<td>Reserved</td>
<td>SINT</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>SINT</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

**Command channel**

Instance ID: 102

Size: 32 bytes

Table 5.6  
**102 output, Command channel**

<table>
<thead>
<tr>
<th>Data</th>
<th>Type</th>
<th>Offset (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>DINT</td>
<td>0</td>
</tr>
<tr>
<td>Identifier</td>
<td>DINT</td>
<td>4</td>
</tr>
<tr>
<td>Argument 1</td>
<td>DINT</td>
<td>8</td>
</tr>
<tr>
<td>Argument 2</td>
<td>DINT</td>
<td>12</td>
</tr>
<tr>
<td>Argument 3</td>
<td>DINT</td>
<td>16</td>
</tr>
<tr>
<td>Argument 4</td>
<td>DINT</td>
<td>20</td>
</tr>
<tr>
<td>Argument 5</td>
<td>DINT</td>
<td>24</td>
</tr>
<tr>
<td>Argument 6</td>
<td>DINT</td>
<td>28</td>
</tr>
</tbody>
</table>

**Command channel result**

Instance ID: 101

Size: 36 bytes

Table 5.7  
**101 input Command channel result**

<table>
<thead>
<tr>
<th>Data</th>
<th>Type</th>
<th>Offset (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>DINT</td>
<td>0</td>
</tr>
<tr>
<td>Identifier</td>
<td>DINT</td>
<td>4</td>
</tr>
<tr>
<td>Error code</td>
<td>DINT</td>
<td>8</td>
</tr>
<tr>
<td>Returnvalue 1</td>
<td>DINT</td>
<td>12</td>
</tr>
<tr>
<td>Returnvalue 2</td>
<td>DINT</td>
<td>16</td>
</tr>
<tr>
<td>Returnvalue 3</td>
<td>DINT</td>
<td>20</td>
</tr>
<tr>
<td>Returnvalue 4</td>
<td>DINT</td>
<td>24</td>
</tr>
<tr>
<td>Returnvalue 5</td>
<td>DINT</td>
<td>28</td>
</tr>
<tr>
<td>Returnvalue 6</td>
<td>DINT</td>
<td>32</td>
</tr>
</tbody>
</table>

**Table 5.5 Slim Command channel**

<table>
<thead>
<tr>
<th>Data</th>
<th>Type</th>
<th>Offset (bytes)</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select reference object</td>
<td>SINT</td>
<td>0</td>
<td>0-31: Selected reference object</td>
</tr>
<tr>
<td>Trigger</td>
<td>SINT</td>
<td>1</td>
<td>1: Trigger inspection. Set to 0 before triggering next inspection.</td>
</tr>
<tr>
<td>Reserved</td>
<td>SINT</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>SINT</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
A

Result output formatting

A.1 XML based formatting

The formatting of the result string is defined by a formatting string written in XML. It is possible to mix XML tags and free text in the formatting string. The text parts will appear as is in the result string, whereas the XML tags will be replaced by the appropriate values. All white spaces in the formatting strings are ignored. In order to include whitespace in the result string use the tags <SPACE/>, <TAB/> and <NEWLINE/>.

The tags are either container tags or value tags. The container tags do not generate any text on their own. It is the value tags inside the container tags that generate the text. The following container tags are valid in the Inspector PIM60:

<table>
<thead>
<tr>
<th>Container tag</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJECT_LOC</td>
<td>Used to present values concerning the Object locator</td>
</tr>
<tr>
<td>BLOB</td>
<td>Used to present values for a found blob in a blob ROI. The index points out the found blob in accordance with the blob sorting order. If no index is given this is the same as index = 0.</td>
</tr>
<tr>
<td>PIXEL_COUNTER, PATTERN, EDGE_PIXEL_COUNTER</td>
<td>Used to present values concerning inspections.</td>
</tr>
<tr>
<td>POLYGON</td>
<td>Used to present values concerning a defined Polygon</td>
</tr>
<tr>
<td>CORNERS</td>
<td>Container tag within the &lt;POLYGON&gt; tag for presenting values concerning the polygon corners. See example:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;POLYGON&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;CORNERS&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;X/&gt;</code>, <code>&lt;Y/&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/CORNERS&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/POLYGON&gt;</code></td>
</tr>
</tbody>
</table>

The XML based formatting string is entered in the Formatting string for Ethernet Output part of the Ethernet Result Output dialog. To get a default string for the current chosen reference object click Create default formatting string. Click Validate output string to validate the formatting string. The output that will be sent over Ethernet or errors are reported in the Current output string part of the Ethernet Result Output dialog.

Note
When using binary transfer, the Validate output string button will only show how many bytes that will be sent for the current analyzed image and whether the formatting was correct or not.

The maximum size of the XML buffer is 7900 ASCII characters. This means it will not be possible to e.g. paste an XML string into to the input field if it’s too large. For a larger configuration it might not be possible to configure as much output information as wanted due to this limitation.

A.2 XML formatting

The content of the Ethernet output is configured using an XML-based formatting string. The available tags can be categorized into two groups:

- container tags: <OBJECT_LOC>, <POLYGON>, <BLOB>, ...
- value tags: <X/>, <PIXELS/>, <NEWLINE/>, <TIME/>,...
The value tags are replaced with a value whereas the container tags are used to group value tags. The container tags do not generate any text on their own. It is the value tags inside the container tags that generate the text.

Attribute value must always be enclosed in quotes.

There are three integer tags (<UINT1/>), <UINT2/>), <UINT3/> for which the values can be changed (in both Edit and Run mode) using the Command channel.

The <BLOB> container tag contains special functionality for presenting values for a certain blob. The index value specifies which blob ROI:s result to present. The index order is the order specified by the Sort by property configured on the Tools tab. The texts and value tags within the <BLOB> tag will be repeated once for each found blob. If only the properties of a single blob are wanted, this can be controlled with the index attribute. See Section A.3, “Container specific tags” (page 40).

## A.3 Container specific tags

All tags are listed in the table below. For each container tag, the available value tags are listed. The binary column states the used data type when using binary output format. Some parts of the formatting string, such as characters and ASCII tags, are only applicable for the ASCII format and will be ignored when using binary format, this is also stated in the binary column.

![Table A.1 Container output string tags](image)

<table>
<thead>
<tr>
<th>Container tag</th>
<th>Value tag</th>
<th>Attribute</th>
<th>Range</th>
<th>Binary</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJECT_LOC X</td>
<td>coordUnit</td>
<td>REAL</td>
<td>X position of the reference point. Note that this can be outside the image and therefore negative. In “pixels” or “mm” depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y coordUnit REAL</td>
<td>REAL</td>
<td>Y position of the reference point. Note that this can be outside the image and therefore negative. In “pixels” or “mm” depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROTATION unit [-180, 180] REAL</td>
<td>In degrees or radians depending on the configured value in the Ethernet Result output dialog.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCALE [0.8, 1.2] REAL</td>
<td>Scale factor of analyzed live image compared to taught reference object.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCORE [0, 100] REAL</td>
<td>Object locator matching score.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECISION [0, 1] USINT 0=not found, 1=found</td>
<td>Name attribute required if more than one Edge Pixel Counter exist</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDGE_PIXEL_name any string</td>
<td>Name attribute required if more than one Edge Pixel Counter exist</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Result output formatting**

### Appendix

<table>
<thead>
<tr>
<th>Container tag</th>
<th>Value tag</th>
<th>Attribute</th>
<th>Range</th>
<th>Binary</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNTER</td>
<td>PIXELS</td>
<td></td>
<td>UDINT</td>
<td></td>
<td>Number of found edge pixels, expressed in pixels of the inspection region area. The No. of edge pixels interval in the Tools tab is specified as number of pixels within the inspection region. If the located object is scaled, the number of pixels is adjusted to be the number of matching pixels that should have been found if the located object had the same size as the reference object.</td>
</tr>
<tr>
<td>DECISION</td>
<td></td>
<td></td>
<td>[0, 1, 2]</td>
<td>USINT</td>
<td>0=fail, 1=pass, 2=Outside image</td>
</tr>
<tr>
<td>PIXEL_COUNTER</td>
<td>name</td>
<td>any string</td>
<td></td>
<td></td>
<td>Name attribute required if more than one Pixel Counter exist</td>
</tr>
<tr>
<td>PIXELS</td>
<td></td>
<td></td>
<td>UDINT</td>
<td></td>
<td>Number of found pixels, expressed in pixels of the inspection region area. The No. of pixels in range interval in the Tools tab is specified as number of pixels within the inspection region. If the located object is scaled, the number of pixels is adjusted to be the number of matching pixels that should have been found if the located object had the same size as the reference object.</td>
</tr>
<tr>
<td>DECISION</td>
<td></td>
<td></td>
<td>[0, 1, 2]</td>
<td>USINT</td>
<td>0=not found, 1=found, 2=Outside image</td>
</tr>
<tr>
<td>PATTERN</td>
<td>name</td>
<td>any string</td>
<td></td>
<td></td>
<td>Name attribute required if more than one Pattern inspection exists</td>
</tr>
<tr>
<td>X</td>
<td>coordUnit</td>
<td>REAL</td>
<td></td>
<td></td>
<td>X position of the reference point. Note that this can be outside the image and therefore negative. In “pixels” or “mm” depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog.</td>
</tr>
<tr>
<td>Y</td>
<td>coordUnit</td>
<td>REAL</td>
<td></td>
<td></td>
<td>Y position of the reference point. Note that this can be outside the image and therefore negative. In “pixels” or “mm” depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog.</td>
</tr>
<tr>
<td>SCORE</td>
<td></td>
<td></td>
<td>[0, 100]</td>
<td>REAL</td>
<td>Pattern matching score.</td>
</tr>
<tr>
<td>Container tag</td>
<td>Value tag</td>
<td>Attribute</td>
<td>Range</td>
<td>Binary</td>
<td>Comment</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>DECISION</td>
<td>[0, 1, 2]</td>
<td>USINT</td>
<td>0=not found, 1=found, 2=Outside image</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BEAD</td>
<td>name</td>
<td>any string</td>
<td></td>
<td></td>
<td>Name attribute required if more than one Bead exists.</td>
</tr>
<tr>
<td>MIN_WIDTH</td>
<td>coordUnit</td>
<td>REAL</td>
<td>Minimum width of bead.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAX_WIDTH</td>
<td>coordUnit</td>
<td>REAL</td>
<td>Maximum width of bead.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEAN_WIDTH</td>
<td>coordUnit</td>
<td>REAL</td>
<td>Mean width of bead.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUM_TOO_NARROW</td>
<td>[0, 256]</td>
<td>UINT</td>
<td>Number of consecutive probes where the width was too narrow.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUM_TOO_WIDE</td>
<td>[0, 256]</td>
<td>UINT</td>
<td>Number of consecutive probes where the width was too wide.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL_NUM_TOO_NARROW</td>
<td>[0, 256]</td>
<td>UINT</td>
<td>Total number of probes where the width was too narrow.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL_NUM_TOO_WIDE</td>
<td>[0, 256]</td>
<td>UINT</td>
<td>Total number of probes where the width was too wide.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECISION</td>
<td>[(1, 2, 4, 6, 8..., 54, 56, 58, 60, 62)]</td>
<td>USINT</td>
<td>1=pass, 2...62=logical OR of the five tolerances: 2=Mean width fail, 4=Longest sequence of too narrow probes fail, 8=Longest sequence of too wide probes fail, 16=Total amount of too narrow probes fail, 32=Total amount of too wide probes fail.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POLYGON</td>
<td>name</td>
<td>any string</td>
<td>Name attribute required if more than one Polygon exists.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUM_CORNERS</td>
<td>[2, 16]</td>
<td>USINT</td>
<td>Number of corners used for this polygon tool.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECISION</td>
<td>[0, 1, 2]</td>
<td>USINT</td>
<td>0=not found, 1=defect, 2=pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCORE</td>
<td>[0, 100]</td>
<td>REAL</td>
<td>Polygon matching score.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUM_PIXELS</td>
<td></td>
<td>UDINT</td>
<td>Number of defect pixels inside crack detection region in polygon. Undefined for single edge tool.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CORNER_OUTSIDE</td>
<td>[0, 1]</td>
<td>USINT</td>
<td>0=polygon completely inside image, 1=one or more polygon corner(s) are outside image. Cannot be used for single edge tool.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEFECT_X</td>
<td>coordUnit</td>
<td>REAL</td>
<td>Coordinate of the first found pixel that was within the defect thresholds. In “pixels” or “mm” depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Container tag</td>
<td>Value tag</td>
<td>Attribute</td>
<td>Range</td>
<td>Binary</td>
<td>Comment</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Return -1 if defect detection is not active or no defect found. Undefined for single edge tool.</td>
</tr>
<tr>
<td>DEFECT_Y</td>
<td>coordUnit</td>
<td>REAL</td>
<td></td>
<td></td>
<td>Coordinate of the first found pixel that was within the defect thresholds. In &quot;pixels&quot; or &quot;mm&quot; depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog. Return -1 if defect detection is not activated or no defect found. Undefined for single edge tool.</td>
</tr>
<tr>
<td>CORNERSa</td>
<td>corners</td>
<td>[0, 1, ...,15, all]</td>
<td></td>
<td></td>
<td>&quot;All&quot; biterates over all polygon corners. Number 0 to 15 gives the properties of a single corner. The index of this corner is the order in which the polygon corner was added when the polygon was drawn.</td>
</tr>
<tr>
<td>X</td>
<td>coordUnit</td>
<td>REAL</td>
<td>REALc</td>
<td></td>
<td>Polygon corner coordinate. In &quot;pixels&quot; or &quot;mm&quot; depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog. For a polygon with two corners (single edge) the estimated corner positions are the intersection between the found edge and the left and right borders of the search region. The search region is defined by the user drawn edge and the position search parameter. See also Operating Instructions for Inspector PIM60 about Single edge tool.</td>
</tr>
<tr>
<td>Y</td>
<td>coordUnit</td>
<td>REAL</td>
<td>REALc</td>
<td></td>
<td>Polygon corner coordinate. &quot;pixels&quot; or &quot;mm&quot; depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog. For a polygon with two corners (single edge) the estimated corner positions are the intersection between the found edge and the left and right borders of the search region. The search region is defined by the user drawn edge and the position search parameter. See also Operating Instructions for Inspector PIM60 about Single edge tool.</td>
</tr>
</tbody>
</table>
## Result output formatting

<table>
<thead>
<tr>
<th>Container tag</th>
<th>Value tag</th>
<th>Attribute</th>
<th>Range</th>
<th>Binary</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOB</td>
<td>name</td>
<td>any string</td>
<td></td>
<td></td>
<td>Name attribute required if more than one Blob tool exists. The name refers to the Blob tool's name in the tool tab.</td>
</tr>
<tr>
<td></td>
<td>index</td>
<td>[0, 15]</td>
<td></td>
<td></td>
<td>Index of found blob according to current blob sorting order. Index 0 is the first blob.</td>
</tr>
<tr>
<td>X</td>
<td>coordUnit</td>
<td>REAL</td>
<td></td>
<td></td>
<td>Blob center of gravity (x position). “pixels” or “mm” depending on attribute “coordUnit”.</td>
</tr>
<tr>
<td>Y</td>
<td>coordUnit</td>
<td>REAL</td>
<td></td>
<td></td>
<td>Blob center of gravity (y position). “pixels” or “mm” depending on attribute “coordUnit”.</td>
</tr>
<tr>
<td>FOUND_BLOBSd</td>
<td></td>
<td>USINT</td>
<td></td>
<td></td>
<td>Number of found blobs.</td>
</tr>
<tr>
<td>LIVE_THRESHOLD_LOWd</td>
<td></td>
<td>USINT</td>
<td></td>
<td></td>
<td>The lower threshold of the Blob tool's intensity after applying ambient light compensation.</td>
</tr>
<tr>
<td>LIVE_THRESHOLD_HIGHd</td>
<td></td>
<td>USINT</td>
<td></td>
<td></td>
<td>The upper threshold of the Blob tool's intensity after applying ambient light compensation.</td>
</tr>
<tr>
<td>ANGLE</td>
<td>unit</td>
<td>[0, 180]</td>
<td>REAL</td>
<td></td>
<td>In degrees or radians depending on attribute “unit” or configured value in the Ethernet Result output dialog.</td>
</tr>
<tr>
<td>AREA</td>
<td></td>
<td>UDINT</td>
<td></td>
<td></td>
<td>Blob area (in pixels). In “pixels” or “mm²” depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog.</td>
</tr>
<tr>
<td>EDGE_FLAG</td>
<td>[0, 1]</td>
<td>USINT</td>
<td></td>
<td></td>
<td>0 = blob fully within ROI, 1 = blob touches ROI border</td>
</tr>
<tr>
<td>EDGE_PIXELS</td>
<td></td>
<td>UDINT</td>
<td></td>
<td></td>
<td>Structure calculation value (number of edge pixels inside the found blob)</td>
</tr>
<tr>
<td>DECISION</td>
<td>[0, 1]</td>
<td>USINT</td>
<td></td>
<td></td>
<td>0=not found, 1=found</td>
</tr>
<tr>
<td>EDGE_LOCATOR</td>
<td>name</td>
<td>any string</td>
<td></td>
<td></td>
<td>Name attribute required if more than one Edge tool exist</td>
</tr>
<tr>
<td>X</td>
<td>coordUnit</td>
<td>REAL</td>
<td></td>
<td></td>
<td>X position of the reference point. In “pixels” or “mm” depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog.</td>
</tr>
<tr>
<td>Y</td>
<td>coordUnit</td>
<td>REAL</td>
<td></td>
<td></td>
<td>Y position of the reference point. In “pixels” or “mm” depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog.</td>
</tr>
<tr>
<td>Container tag</td>
<td>Value tag</td>
<td>Attribute</td>
<td>Range</td>
<td>Binary</td>
<td>Comment</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>-----------</td>
<td>----------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SCORE</td>
<td></td>
<td></td>
<td>[0, 100]</td>
<td>REAL</td>
<td>Edge matching score.</td>
</tr>
<tr>
<td>ANGLE</td>
<td>unit</td>
<td></td>
<td>[0, 180]</td>
<td>REAL</td>
<td>In degrees or radians depending on attribute “unit” or configured value in the Ethernet Result output dialog.</td>
</tr>
<tr>
<td>DECISION</td>
<td></td>
<td></td>
<td>[0, 1, 2]</td>
<td>USINT</td>
<td>0=not found, 1=found, 2=Outside image</td>
</tr>
<tr>
<td>CIRCLE_LOCATOR</td>
<td>name</td>
<td>any string</td>
<td></td>
<td></td>
<td>Name attribute required if more than one Circle exist</td>
</tr>
<tr>
<td>X</td>
<td>coordUnit</td>
<td></td>
<td>REAL</td>
<td></td>
<td>X position of the reference point. In &quot;pixels&quot; or &quot;mm&quot; depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog.</td>
</tr>
<tr>
<td>Y</td>
<td>coordUnit</td>
<td></td>
<td>REAL</td>
<td></td>
<td>Y position of the reference point. In &quot;pixels&quot; or &quot;mm&quot; depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog.</td>
</tr>
<tr>
<td>SCORE</td>
<td></td>
<td></td>
<td>[0, 100]</td>
<td>REAL</td>
<td>Circle matching score.</td>
</tr>
<tr>
<td>DIAMETER</td>
<td>coordUnit</td>
<td></td>
<td>REAL</td>
<td></td>
<td>Circle’s diameter in pixels or mm depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog.</td>
</tr>
<tr>
<td>DECISION</td>
<td></td>
<td></td>
<td>[0, 1, 2, 11, 12]</td>
<td>USINT</td>
<td>0=not found, 1=pass, 2=Outside image, 11=Failed on score value, 12=Failed on diameter</td>
</tr>
<tr>
<td>MEASURE_DISTANCE</td>
<td>name</td>
<td>any string</td>
<td></td>
<td></td>
<td>Name attribute required if more than one Measure distance exists</td>
</tr>
<tr>
<td>DISTANCE</td>
<td>coordUnit</td>
<td></td>
<td>REAL</td>
<td></td>
<td>Measured distance in pixels or mm depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog.</td>
</tr>
<tr>
<td>VALID</td>
<td></td>
<td></td>
<td>[0, 1]</td>
<td>USINT</td>
<td>Indicates whether the value in the distance tag is a valid measurement or not. 0=Invalid 1=Valid</td>
</tr>
<tr>
<td>DECISION</td>
<td></td>
<td></td>
<td>[0, 1]</td>
<td>USINT</td>
<td>0=fail (measure distance could not be done), 1=pass</td>
</tr>
<tr>
<td>MEASURE_ANGLE</td>
<td>name</td>
<td>any string</td>
<td></td>
<td></td>
<td>Name attribute required if more than one Measure Angle exist</td>
</tr>
<tr>
<td>X</td>
<td>coordUnit</td>
<td></td>
<td>REAL</td>
<td></td>
<td>X position of the intersection point. In &quot;pixels&quot; or &quot;mm&quot; depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog.</td>
</tr>
<tr>
<td>Container tag</td>
<td>Value tag</td>
<td>Attribute</td>
<td>Range</td>
<td>Binary</td>
<td>Comment</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>Y</td>
<td>coordUnit</td>
<td>REAL</td>
<td></td>
<td></td>
<td>Y position of the intersection point. In “pixels” or “mm” depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog.</td>
</tr>
<tr>
<td>ANGLE</td>
<td>[0, 180]</td>
<td>REAL</td>
<td></td>
<td></td>
<td>In degrees or radians depending on attribute “unit” or configured value in the Ethernet Result output dialog.</td>
</tr>
<tr>
<td>VALID</td>
<td>[0, 1, 2]</td>
<td>USINT</td>
<td></td>
<td></td>
<td>Indicates whether the values in the angle tag are valid measurements or not. 0=Invalid 1=Valid 2=Intersection outside image</td>
</tr>
<tr>
<td>DECISION</td>
<td>[0, 1]</td>
<td>USINT</td>
<td></td>
<td></td>
<td>0=fail, 1=pass</td>
</tr>
<tr>
<td>EDGE_COUNT</td>
<td>name</td>
<td>any string</td>
<td></td>
<td></td>
<td>Name attribute required if more than one Edge counter exist</td>
</tr>
<tr>
<td></td>
<td>index</td>
<td>[0, 63]</td>
<td></td>
<td></td>
<td>Chosen feature/edge index.</td>
</tr>
<tr>
<td>FOUND_FEATURES</td>
<td>[0, 64]</td>
<td>USINT</td>
<td></td>
<td></td>
<td>Number of found features.</td>
</tr>
<tr>
<td>MIN_PITCH</td>
<td>coordUnit</td>
<td>REAL</td>
<td></td>
<td></td>
<td>If the chosen Edge counter is rectangular the value shows the pitch “pixels” or “mm” depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog.</td>
</tr>
<tr>
<td>MAX_PITCH</td>
<td>coordUnit</td>
<td>REAL</td>
<td></td>
<td></td>
<td>If the chosen Edge counter is circular the value is the pitch in degrees or radians depending on attribute “unit” or configured value in the Ethernet Result output dialog.</td>
</tr>
<tr>
<td>MEAN_PITCH</td>
<td>coordUnit</td>
<td>REAL</td>
<td></td>
<td></td>
<td>If the chosen Edge counter is rectangular the value shows the pitch “pixels” or “mm” depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog.</td>
</tr>
</tbody>
</table>
### Container tag | Value tag | Attribute | Range | Binary | Comment
--- | --- | --- | --- | --- | ---
 | | | | | **If the chosen Edge counter is circular the value is the pitch in degrees or radians depending on attribute “unit” or configured value in the Ethernet Result output dialog.**

| DECISION⁶ | [0, 1, 2] | USINT | | 0=Not found, 1=Found, 2=Outside image |

| X | coordUnit | REAL | X position for the mid point of the feature/edge, see footnote. In “pixels” or “mm” depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog. |

| Y | coordUnit | REAL | Y position for the mid point of the feature/edge, see footnote. In “pixels” or “mm” depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog. |

| ANGLE | [0, 180] | REAL | Only applicable for Feature type Single edge. A positive value is a clockwise rotation and a negative value is a counter clockwise rotation. The angle of the single edge in degrees or radians depending on attribute “unit” or configured value in the Ethernet Result output dialog. |

| INTERNAL_ANGLE | [0, 180] | REAL | Only applicable for Feature type Dark or Bright. The edge's or feature's angle in the search region. The angle of the feature in degrees or radians depending on attribute “unit” or configured value in the Ethernet Result output dialog. |

| WIDTH | | REAL | Only applicable for Feature type Dark or Bright. If the chosen Edge counter is rectangular the value shows the width “pixels” or “mm” depending on attribute “coordUnit” or configured value in the Ethernet Result output dialog. If the chosen Edge counter is circular the value is the width in degrees or radians depending on attribute “unit” or configured value in the Ethernet Result output dialog. |
**Result output formatting**

<table>
<thead>
<tr>
<th>Container tag</th>
<th>Value tag</th>
<th>Attribute</th>
<th>Range</th>
<th>Binary</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>POLARITY</td>
<td>[3, 4]</td>
<td>USINT</td>
<td></td>
<td>figured value in the Ethernet Result output dialog. Only applicable for Feature type Single edge. The chosen edge's polarity in the search direction: 3 = dark to bright, 4 = bright to dark</td>
</tr>
</tbody>
</table>

^This tag must be used inside the <POLYGON> container

^Only available for Ethernet Raw.

^For EtherNet/IP the position is represented as an INT value. Use the scale attribute to get more decimals

^This tag must be used inside the <BLOB> container. The value are given for each Blob ROI (not for each found blob)

^This tag concerns the whole tool, i.e. not specific to the feature/edge chosen by the index

^Red arrow points out the midpoint for the Edge count tool

---

^The angle of the edge relative to the region.

^The INTERNAL_ANGLE is the angle of a feature, i.e. the relation of the two edges of the feature.

---

**Note**

When a tool is related to the object locator and the object locator is not found in the live image the presented results for the related tools are undefined.
### A.3.1 General tags

#### Table A.2 Generic output string tags

<table>
<thead>
<tr>
<th>Value tag</th>
<th>Attribute</th>
<th>Range</th>
<th>Binary</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESSAGE_SIZE</td>
<td>UINT</td>
<td></td>
<td></td>
<td>• Binary format: The size of the message in bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• ASCII format: The number of characters in the message</td>
</tr>
<tr>
<td>IMAGE_NUMBER</td>
<td>UDINT</td>
<td>[0, 31]</td>
<td></td>
<td>Analyzed image's number (Resets at power-up or device reset)</td>
</tr>
<tr>
<td>IMAGE_DECISION</td>
<td>USINT</td>
<td>[0, 3]</td>
<td></td>
<td>0=Not located, 1=Detail failed, 2=All passed 3=Not located and detail failed</td>
</tr>
<tr>
<td>REF_OBJECT</td>
<td>USINT</td>
<td>[0, 31]</td>
<td></td>
<td>Reference object index</td>
</tr>
<tr>
<td>ASCII value</td>
<td>IGORDED</td>
<td>[0, 255]</td>
<td></td>
<td>Used to send single control characters</td>
</tr>
<tr>
<td>SPACE</td>
<td>IGNORED</td>
<td></td>
<td></td>
<td>Same as &lt;ASCII value=&quot;32&quot;/&gt;</td>
</tr>
<tr>
<td>TAB</td>
<td>IGNORED</td>
<td></td>
<td></td>
<td>Same as &lt;ASCII value=&quot;9&quot;/&gt;</td>
</tr>
<tr>
<td>LAB</td>
<td>IGNORED</td>
<td></td>
<td></td>
<td>Left angular bracket, &quot;&lt;&quot;. Useful when generating XML-formatted output.</td>
</tr>
<tr>
<td>RAB</td>
<td>IGNORED</td>
<td></td>
<td></td>
<td>Right angular bracket, &quot;&gt;&quot;. Useful when generating XML-formatted output.</td>
</tr>
<tr>
<td>NEWLINE</td>
<td>IGNORED</td>
<td></td>
<td></td>
<td>Same as &lt;ASCII value=&quot;10&quot;/&gt;</td>
</tr>
<tr>
<td>RETURN</td>
<td>IGNORED</td>
<td></td>
<td></td>
<td>Same as &lt;ASCII value=&quot;13&quot;/&gt;</td>
</tr>
<tr>
<td>TIME</td>
<td>UDINT</td>
<td>[s, ms]</td>
<td></td>
<td>Current time since device boot. Restarts from zero after ~10 years (using seconds) and ~49 days (using milliseconds)</td>
</tr>
<tr>
<td>SERIALCODE</td>
<td>UDINT</td>
<td></td>
<td></td>
<td>Device serial code.</td>
</tr>
<tr>
<td>FOCUS</td>
<td>REAL</td>
<td>[0, 100]</td>
<td></td>
<td>Only valid while the device is in Edit mode. This is the focus value from the Image settings tab</td>
</tr>
<tr>
<td>TELEGRAM_COUNTER</td>
<td>UINT</td>
<td></td>
<td></td>
<td>A counter that increments for each telegram sent over the result channel. Resets at power-up or device reset.</td>
</tr>
<tr>
<td>USINT intValue</td>
<td>USINT</td>
<td>[0, 255]</td>
<td></td>
<td>If the intValue attribute is not specified the default value will be zero and the tag can be used for padding.</td>
</tr>
<tr>
<td>UINT intValue</td>
<td>UINT</td>
<td>[0, 65535]</td>
<td></td>
<td>If the intValue attribute is not specified the default value will be zero and the tag can be used for padding.</td>
</tr>
<tr>
<td>UDINT intValue</td>
<td>UDINT</td>
<td>[0, 2^32-1]</td>
<td></td>
<td>If the intValue attribute is not specified the default value will be zero and the tag can be used for padding.</td>
</tr>
<tr>
<td>UINT1 intValue</td>
<td>UINT</td>
<td>[0, 65535]</td>
<td></td>
<td>Value can be changed through the command channel.</td>
</tr>
<tr>
<td>UINT2 intValue</td>
<td>UINT</td>
<td>[0, 65535]</td>
<td></td>
<td>Value can be changed through the command channel.</td>
</tr>
</tbody>
</table>
A.3.2 Attributes

Attributes are used to control the formatting and identification of inspections. The table below describes the formatting attributes for Inspector. Some attributes can also be set, for the whole formatting string, in the Ethernet Result Output dialog in the InspectorPIM60 menu in the section Message settings. The attributes operate in a hierarchical way using inheritance. So if Number of decimals has been set to 3 in the Ethernet Result Output dialog, all REAL will be printed with 3 decimals unless they are inside a tag that states otherwise. Some attributes can also be set from the Ethernet Result Output dialog from the InspectorPIM60 menu in SOPAS Engineering Tool (ET), see also Section 4.2.3, “Attributes” (page 23).

**Table A.3 Formatting attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Range</th>
<th>Default value</th>
<th>Affects</th>
<th>Used in Binary format</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>[0, 15]</td>
<td>0</td>
<td>Blob</td>
<td>Yes</td>
<td>Index of blob according to current blob sorting order. Index 0 is the first blob.</td>
</tr>
<tr>
<td>scale</td>
<td>Any REAL</td>
<td>1.0</td>
<td>All values</td>
<td>Yes</td>
<td>Scales the values before they are printed. Can for example be used to express positions as integers in 1/10 pixel units</td>
</tr>
<tr>
<td>base</td>
<td>[decimal, octal, hex]</td>
<td>decimal</td>
<td>Integers</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>timeUnit</td>
<td>[s, ms]</td>
<td>s</td>
<td>&lt;TIME&gt;</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>any string</td>
<td>none</td>
<td>Identification of tools</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>[0, 255]</td>
<td>0</td>
<td>&lt;ASCII&gt;</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>intValue</td>
<td>[0, 255], [0, 65535], [0, 2^32-1]</td>
<td>0</td>
<td>&lt;USINT&gt;, &lt;UINT&gt;, &lt;UDINT&gt;</td>
<td>Yes Integer value to be sent.</td>
<td></td>
</tr>
<tr>
<td>digits</td>
<td>[0, 9]</td>
<td>0</td>
<td>Integers and REAL</td>
<td>No</td>
<td>Minimum number of characters.</td>
</tr>
<tr>
<td>decimals</td>
<td>[0, 9]</td>
<td>2</td>
<td>REAL</td>
<td>No</td>
<td>Number of decimals.</td>
</tr>
<tr>
<td>corners</td>
<td>[0, 15]</td>
<td>&lt;CORNERS&gt;</td>
<td>Yes</td>
<td>*All “iterate over all polygon corners. Number 0 to 15 gives the properties of a single corner. The index of this corner is the order in which the polygon corner was added when the polygon was drawn.</td>
<td></td>
</tr>
</tbody>
</table>
### Result output formatting

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Range</th>
<th>Default value</th>
<th>Affects</th>
<th>Used in Binary format</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>coordUnit</td>
<td>[pixels, mm, aligned]</td>
<td>pixels</td>
<td>Object locator, Blob, Pattern, Single edge, Circle, Edge coordinates,</td>
<td>Yes</td>
<td>Gives result coordinates in pixel, millimeter or robot aligned millimeter format.</td>
</tr>
<tr>
<td>dataType</td>
<td>[SINT, INT, DINT, REAL]</td>
<td>All values</td>
<td></td>
<td>Yes</td>
<td>Casts to the specified datatype. When using EtherNet/IP the attribute DataType specifies the dataType section in the selected assembly.</td>
</tr>
<tr>
<td>pos</td>
<td>[0, 43]</td>
<td>All values</td>
<td></td>
<td>No</td>
<td>Used by EtherNet/IP to determine a position in the dataType section in the selected assembly. The first position number of the dataType section is 0. The range of the attribute pos depends on which assembly is used.</td>
</tr>
<tr>
<td>unit</td>
<td>[radians, degrees]</td>
<td>degrees</td>
<td>Angles</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

*Only available for Ethernet Raw.

*The device must be calibrated for it to be possible to use the “mm” or “aligned” attribute.

**Table A.4 Sizes of datatypes**

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Size</th>
<th>Range</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>USINT</td>
<td>1 byte</td>
<td>[0, 255]</td>
<td>a</td>
</tr>
<tr>
<td>SINT</td>
<td>1 byte</td>
<td>[-128, 127]</td>
<td>a</td>
</tr>
<tr>
<td>UINT</td>
<td>2 bytes</td>
<td>[0, 65535]</td>
<td>a</td>
</tr>
<tr>
<td>INT</td>
<td>2 bytes</td>
<td>[-32768, 32767]</td>
<td>a</td>
</tr>
<tr>
<td>UDINT</td>
<td>4 bytes</td>
<td>[0, 2^32-1]</td>
<td>a</td>
</tr>
<tr>
<td>DINT</td>
<td>4 bytes</td>
<td>[-2^31, 2^31-1]</td>
<td>a</td>
</tr>
<tr>
<td>REAL</td>
<td>4 bytes</td>
<td>Represented as IEEE 754 binary 32</td>
<td>a</td>
</tr>
</tbody>
</table>

*See Section 4.2.3, “Attributes” (page 23).
Command channel

The Command Channel is used to read and update a selected set of device parameters. This section describes the Command Channel from a generic point of view. The Command Channel is available via several of the device interfaces: Ethernet Raw, EtherNet/IP, Web API. There are differences depending on the possibilities each interface provides. The differences are described in the chapters about each interface.

It is possible to block changes via the command channel individually for each interface using a setting in the interface configuration, as described in the Operating Instructions for Inspector PIM60. This makes it possible to allow changes via a PLC oriented interface while blocking changes via the Web API.

B.1 Command syntax

The tables below describe the different command types as well as ACK messages and their syntax. The basic principle is that there are three major types of commands (sINT, gINT, and aACT) and some special commands.

Table B.1 Command syntax

<table>
<thead>
<tr>
<th>Command format</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>gVER</td>
<td>Get protocol version that is supported by the addressed device</td>
</tr>
<tr>
<td>sMOD [mode]</td>
<td>Set device mode (0 = Run, 1 = Edit)</td>
</tr>
<tr>
<td>gMOD</td>
<td>Get the current device mode from the device</td>
</tr>
<tr>
<td>sINT [identifier] [arg1] [arg2] ... [argN]</td>
<td>Set “integer” parameter in the device</td>
</tr>
<tr>
<td>gINT [identifier] [arg1] [arg2] ... [argN]</td>
<td>Get “integer” parameter from the device</td>
</tr>
<tr>
<td>gSTR [identifier] [arg1] [arg2]</td>
<td>Get &quot;string&quot; parameter from device</td>
</tr>
<tr>
<td>aACT [identifier] [arg1] [arg2] ... [argN]</td>
<td>Action commands</td>
</tr>
<tr>
<td>TRIG</td>
<td>Trig an image acquisition and analysis</td>
</tr>
<tr>
<td>gRES</td>
<td>Retrieve the latest available Ethernet Result Output string</td>
</tr>
<tr>
<td>gSTAT</td>
<td>Retrieve the latest statistics from the device</td>
</tr>
</tbody>
</table>
Table B.2  Command response

<table>
<thead>
<tr>
<th>ACK message</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>rgVER [errorCode] [protocolVersion]</td>
<td>Response to protocol version including the version that is supported by the device</td>
</tr>
<tr>
<td>rsMOD [errorCode] [errorMessage]</td>
<td>Response to set mode (Run/Edit) including error code and error message</td>
</tr>
<tr>
<td>rgMOD [errorCode] [mode] [errorMessage]</td>
<td>Response to fetch current mode (Run/Edit) including the mode, error code, and error message</td>
</tr>
<tr>
<td>rsINT [identifier] [errorCode] [errorMessage]</td>
<td>Response to set integer parameter and action commands including error code and error message</td>
</tr>
<tr>
<td>rgINT [identifier] [errorCode] [ret1] [ret2] ...[retN] [errorMessage]</td>
<td>Response to fetch integer parameter including parameter value, error code and error message</td>
</tr>
<tr>
<td>raACT [identifier] [errorCode] [errorMessage]</td>
<td>Response to the action command including error code and error message</td>
</tr>
<tr>
<td>rTRIG [errorCode] [errorMessage]</td>
<td>Response to the trig command including error code and error message</td>
</tr>
<tr>
<td>rgSTR [identifier] [errorCode] [errorMessage/nameString]</td>
<td>Response to the get string command. If errorCode is 0 (No error) the errorMessage is instead the actual response string</td>
</tr>
<tr>
<td>rgRES [errorCode] [errorMessage/resultString]</td>
<td>Response to the get latest available Ethernet Result Output string. If errorCode is 0 (No error) the errorMessage is instead the actual Ethernet Result Output string</td>
</tr>
<tr>
<td>rgSTAT [statistics in XML format]</td>
<td>Response to get the latest statistics from the device in XML format</td>
</tr>
</tbody>
</table>

If returned errorCode is 0 no errorMessage will be shown. For explanation of errorCode and errorMessage see Section B.4, “Error codes” (page 74).

The response message is a receipt that the command is valid and is executed on the Inspector. However, the following commands take longer time to execute, and may not have finished executing when you receive the command response:

- All action commands (aACT)
- Select reference object (sINT 1)
- Set mode (sMOD)

When sent over Ethernet Raw, all command responses have a start and end character:

- Start character = STX (This character has the ASCII decimal number 2)
- End character = ETX (This character has the ASCII decimal number 3)

**B.1.1 Commands ID numbers for EtherNet/IP**

Table with command ID numbers to be used as replacement for the normal command strings for interfaces where strings not are possible or preferred.
Table B.3  Command ID numbers - for EtherNet/IP

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set mode</td>
<td>sMOD</td>
<td>0</td>
</tr>
<tr>
<td>Get mode</td>
<td>gMOD</td>
<td>1</td>
</tr>
<tr>
<td>Set integer</td>
<td>sINT</td>
<td>2</td>
</tr>
<tr>
<td>Get integer</td>
<td>gINT</td>
<td>3</td>
</tr>
<tr>
<td>Get string</td>
<td>gSTR</td>
<td>6</td>
</tr>
<tr>
<td>Get version</td>
<td>gVER</td>
<td>7</td>
</tr>
<tr>
<td>Action command</td>
<td>aACT</td>
<td>8</td>
</tr>
<tr>
<td>Trig device</td>
<td>TRIG</td>
<td>9</td>
</tr>
</tbody>
</table>

B.2  Command channel index handling

B.2.1  Introduction
The index argument in the command descriptions below refers to tool's index when configured in SOPAS Engineering Tool (ET). The index can be found in the Tools tab in SOPAS Engineering Tool (ET). Hold the mouse pointer over the current tool to get the index number. There are different indexes depending on which tool is used.

B.2.2  Blob indexing
The blob index argument in the commands corresponds to the order in which the blobs are listed in the Tools tab in the SOPAS Engineering Tool (ET), starting with 0.

B.2.3  Bead indexing
The bead index argument in the commands corresponds to the order in which the beads are listed in the Tools tab in the SOPAS Engineering Tool (ET), starting with 0.

B.2.4  Polygon indexing
The polygon index argument corresponds to the order in which the polygons are listed in the Tools tab in the SOPAS Engineering Tool (ET), starting with 0.

B.2.5  Tools indexing
The index argument corresponds to the order in which the pixel counter, edge pixel counter, pattern, edge tool, edge counter, circle locator, measure distance, and measure angle are listed in the Tools tab in the SOPAS Engineering Tool (ET), starting with 0. The types of the tools (pixel counter, edge pixel counter, pattern, edge tool, edge counter, circle locator, measure distance, or measure angle) do not matter, i.e. if two Pattern tool are listed above a Pixel counter tool, the Pixel counter tool has index 2. If a tool (pixel counter, edge pixel counter, pattern, edge tool, edge counter, circle locator, measure distance, or measure angle) in the beginning of the list is deleted, the following tool (pixel counter, edge pixel counter, pattern, edge tool, edge counter, circle locator, measure distance, or measure angle) will be updated with a new index.

B.3  Command descriptions
The way to configure the device through the Ethernet based command channel is based on the set of commands described above with parameters depending on what the user wants to do. See tables below with a complete list of command channel actions and functions.

The index argument in the command descriptions below refers to tool's index when configured in SOPAS Engineering Tool (ET). The index can be found in the Tools tab in SOPAS Engineering Tool (ET). Hold the mouse pointer over the current tool to get the index number.
### Table B.4 Command channel - actions

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save settings in flash</td>
<td>aACT</td>
<td>1</td>
<td>No</td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Re-teach reference object</td>
<td>aACT</td>
<td>2</td>
<td>Yes, only in Run mode.</td>
<td>int auto-Exp</td>
<td></td>
<td>0=use exposure settings, 1=auto adjust</td>
</tr>
<tr>
<td>Perform calibration(^a)</td>
<td>aACT</td>
<td>3</td>
<td>No</td>
<td>int box-Size(^b)</td>
<td></td>
<td>&gt;0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- int calibrationCoverage(^c)</td>
<td></td>
<td>[0,100]</td>
</tr>
<tr>
<td>Remove calibration</td>
<td>aACT</td>
<td>4</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apply IP settings(^d)</td>
<td>aACT</td>
<td>5</td>
<td>Yes</td>
<td>int useDH-CP</td>
<td></td>
<td>0=use manual settings, 1=use DHCP</td>
</tr>
<tr>
<td>Restart the Inspector(^e)</td>
<td>aACT</td>
<td>6</td>
<td>Both in run and edit mode</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perform (calculate) coordinate alignment(^f)</td>
<td>aACT</td>
<td>7</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^a\)In order to run this command the device must be set to Calibration mode (sINT 20 1).

\(^b\)The argument must be given in mm.

\(^c\)Returned value is the calibration target coverage in percent.

\(^d\)After the aACT 5 command has been executed the Inspector need to be restarted (e.g. using aACT 6) before the new IP settings are in use.

\(^e\)The aACT 6 command will make the device being temporarily disconnected while it is rebooted.

\(^f\)This command can only be used if the 4 control points have been set via the sINT 300 command.
### Table B.5 Command channel functions - Device settings

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set interface permission</td>
<td>siINT</td>
<td>112</td>
<td>Yes</td>
<td>int interface int permission</td>
<td>Interface: 0=Ethernet Raw, 1=HTTP, 2=EtherNet/IP Permission: 1=enable, 0=disable</td>
<td></td>
</tr>
<tr>
<td>Get interface permission</td>
<td>giINT</td>
<td>112</td>
<td>Yes</td>
<td>int interface int permission</td>
<td>Interface: 0=Ethernet Raw, 1=HTTP, 2=EtherNet/IP</td>
<td></td>
</tr>
<tr>
<td>Set device IP address</td>
<td>siINT</td>
<td>120</td>
<td>Yes</td>
<td>int a, b, c, d</td>
<td>Address format: a.b.c.d</td>
<td></td>
</tr>
<tr>
<td>Get device IP address</td>
<td>giINT</td>
<td>120</td>
<td>Yes</td>
<td>int a, b, c, d</td>
<td>Address format: a.b.c.d</td>
<td></td>
</tr>
<tr>
<td>Set device net-mask</td>
<td>siINT</td>
<td>121</td>
<td>Yes</td>
<td>int a, b, c, d</td>
<td>Address format: a.b.c.d</td>
<td></td>
</tr>
<tr>
<td>Get device net-mask</td>
<td>giINT</td>
<td>121</td>
<td>Yes</td>
<td>int a, b, c, d</td>
<td>Address format: a.b.c.d</td>
<td></td>
</tr>
<tr>
<td>Set gateway</td>
<td>siINT</td>
<td>122</td>
<td>Yes</td>
<td>int a, b, c, d</td>
<td>Address format: a.b.c.d</td>
<td></td>
</tr>
<tr>
<td>Get gateway</td>
<td>giINT</td>
<td>122</td>
<td>Yes</td>
<td>int a, b, c, d</td>
<td>Address format: a.b.c.d</td>
<td></td>
</tr>
</tbody>
</table>

*a* In order for the settings to take effect the aACT 5 command needs to be sent to the device

*b* There should be no dots in the argument

### Table B.6 Command channel functions - general

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get used protocol version</td>
<td>giVER</td>
<td>-</td>
<td>Yes</td>
<td>int version</td>
<td>6=PIM60 1.0</td>
<td></td>
</tr>
<tr>
<td>Set device mode</td>
<td>siMOD</td>
<td>-</td>
<td>Yes</td>
<td>int mode</td>
<td>0=Run, 1=Edit</td>
<td></td>
</tr>
<tr>
<td>Get device mode</td>
<td>giMOD</td>
<td>-</td>
<td>Yes</td>
<td>int mode</td>
<td>0=Run, 1=Edit</td>
<td></td>
</tr>
<tr>
<td>Trig device</td>
<td>TRIG</td>
<td>-</td>
<td>Yes a</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Select reference object</td>
<td>siINT</td>
<td>1</td>
<td>Yes</td>
<td>int object</td>
<td>[0, 31]</td>
<td></td>
</tr>
<tr>
<td>Get active reference object</td>
<td>giINT</td>
<td>1</td>
<td>Yes</td>
<td>int object</td>
<td>[0,31]</td>
<td></td>
</tr>
<tr>
<td>Get number of configured reference objects</td>
<td>giINT 2</td>
<td>Yes</td>
<td>int object</td>
<td>[0,31]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change internal illumination mode</td>
<td>siINT 13</td>
<td>No</td>
<td>int illumMode</td>
<td>0=Off, 1=On</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Get internal illumination mode</td>
<td>giINT 13</td>
<td>Yes</td>
<td>int illumMode</td>
<td>0=Off, 1=On</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Command</td>
<td>Identifier</td>
<td>Usable in Run mode</td>
<td>Arguments</td>
<td>Return values</td>
<td>Range</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------</td>
<td>------------</td>
<td>--------------------</td>
<td>--------------------</td>
<td>----------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Change exposure³</td>
<td>sINT</td>
<td>14</td>
<td>No</td>
<td>int exp*100</td>
<td></td>
<td>[10,10000]</td>
</tr>
<tr>
<td>Get exposure</td>
<td>gINT</td>
<td>14</td>
<td>Yes</td>
<td>int exp*100</td>
<td></td>
<td>[10,10000]</td>
</tr>
<tr>
<td>Change gain</td>
<td>sINT</td>
<td>15</td>
<td>No</td>
<td>int gain</td>
<td></td>
<td>[0, 400]</td>
</tr>
<tr>
<td>Get gain</td>
<td>gINT</td>
<td>15</td>
<td>Yes</td>
<td>int gain</td>
<td></td>
<td>[0, 400]</td>
</tr>
<tr>
<td>Switch trigger mode (free-running, trig)</td>
<td>sINT</td>
<td>16</td>
<td>No</td>
<td>int mode</td>
<td>0=free-running, 1=trig</td>
<td></td>
</tr>
<tr>
<td>Get trigger mode</td>
<td>gINT</td>
<td>16</td>
<td>Yes</td>
<td>int trigMode</td>
<td>0=free-running, 1=trig</td>
<td></td>
</tr>
<tr>
<td>Change value of integer tags in result output (UINT1-3)</td>
<td>sINT</td>
<td>18</td>
<td>Yes</td>
<td>int index, int value</td>
<td></td>
<td>[0, 2], [0,65535]</td>
</tr>
<tr>
<td>Get value of integer tags in result output (UINT1-3)</td>
<td>gINT</td>
<td>18</td>
<td>Yes</td>
<td>int index, int value</td>
<td></td>
<td>[0, 2], [0,65535]</td>
</tr>
<tr>
<td>Get frame period time</td>
<td>gINT</td>
<td>19</td>
<td>Yes</td>
<td>int framePeriod</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enter/leave calibration mode</td>
<td>sINT</td>
<td>20</td>
<td>No</td>
<td>int mode</td>
<td>0=normal mode, 1=calibration mode</td>
<td></td>
</tr>
<tr>
<td>Get calibration parameters</td>
<td>gINT</td>
<td>20</td>
<td>Yes</td>
<td>int parameter</td>
<td>int parameterResult 0=calibration/normal mode³, 1=calibrated⁴, 2=scaling⁵, 3=origin⁵, 4=rotation⁵, 5=mean pixel error, 6=max pixel error</td>
<td></td>
</tr>
<tr>
<td>Set external trig delay</td>
<td>sINT</td>
<td>21</td>
<td>No</td>
<td>int type, int delay (milliseconds*10 or ticks)¹</td>
<td>[0=ms, 1=tick], [1,50000] resp [0 ticks, 200000 ticks]</td>
<td></td>
</tr>
<tr>
<td>Get external trig delay</td>
<td>gINT</td>
<td>21</td>
<td>Yes</td>
<td>int type, int delay (milliseconds*10 or ticks)</td>
<td>[0=ms, 1=tick], [1,50000] resp [0 ticks, 200000 ticks]</td>
<td></td>
</tr>
<tr>
<td>Set digital output delay</td>
<td>sINT</td>
<td>22</td>
<td>No</td>
<td>int outputIndex, int type, int delay (milliseconds*10 or ticks)</td>
<td>[0,19], [0=ms, 1=tick], [[Min delay time]*10,50000] resp [0 ticks, 200000 ticks]</td>
<td></td>
</tr>
<tr>
<td>Get digital output delay</td>
<td>gINT</td>
<td>22</td>
<td>Yes</td>
<td>int outputIndex</td>
<td>int type, int delay (milliseconds*10 or ticks)</td>
<td>[0,19], [0=ms, 1=tick], [1,50000] resp [0 ticks, 200000 ticks]</td>
</tr>
</tbody>
</table>
## Command channel

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set digital output active time</td>
<td>sINT</td>
<td>23</td>
<td>No</td>
<td>int outputIndex, int type, int time (milliseconds*10 or ticks)</td>
<td>[0,19], [0=ms, 1=tick], [1,100000] resp [0 ticks, 200000 ticks]</td>
<td></td>
</tr>
<tr>
<td>Get digital output active time</td>
<td>gINT</td>
<td>23</td>
<td>Yes</td>
<td>int outputIndex</td>
<td>int type, int time (milliseconds*10 or ticks)</td>
<td>[0,19], [0=ms, 1=tick], [1,10000] resp [0 ticks, 200000 ticks]</td>
</tr>
<tr>
<td>Set alignment control point</td>
<td>sINT</td>
<td>300</td>
<td>No</td>
<td>int pointIndex, int x * 100, int y * 100, int z * 100</td>
<td>[0=A, 1=B, 2=C, 3=D] int x * 100, [-2147483648, 2147483648] int y * 100, [-2147483648, 2147483648] int z * 100, [-2147483648, 2147483648]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change object locator match threshold</td>
<td>sINT</td>
<td>32</td>
<td>No</td>
<td>int threshold</td>
<td>[0, 100], percent</td>
<td></td>
</tr>
<tr>
<td>Get object locator match threshold</td>
<td>gINT</td>
<td>32</td>
<td>Yes</td>
<td></td>
<td></td>
<td>[0, 100], percent</td>
</tr>
<tr>
<td>Change object locator rotation search mode</td>
<td>sINT</td>
<td>33</td>
<td>No</td>
<td>int mode</td>
<td>0=off, 1=on</td>
<td></td>
</tr>
</tbody>
</table>

---

While possible to trig over all interfaces, it is recommended to use only the following for high speed applications:
- Digital I/O (In3)
- Ethernet Raw (port 2116)
- EtherNet/IP (small assembly)

The exposure is expressed in ms multiplied by 100 i.e. 3.8 ms is expressed as 380

The result is 0= normal mode or 1= calibration mode.

The result is not calibrated or 1= calibrated.

The result is expressed in mm/pixel x 10000.

The result is expressed in pixels for x and y.

The result is expressed in degrees.

The delay is expressed in ms multiplied by 10 i.e. 1.5 ms is expressed as 15

In order for the settings to take effect the aACT 7 command needs to be sent to the device
### Command channel

#### Table B.8 Command channel functions - Blob

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get object locator rotation search mode</td>
<td>gINT</td>
<td>33</td>
<td>Yes</td>
<td>int mode</td>
<td></td>
<td>0=off, 1=on</td>
</tr>
<tr>
<td>Change object locator rotation search limit</td>
<td>sINT</td>
<td>34</td>
<td>No</td>
<td>int limit</td>
<td>[0, 180] degrees</td>
<td></td>
</tr>
<tr>
<td>Get object locator rotation search limit</td>
<td>gINT</td>
<td>34</td>
<td>Yes</td>
<td>int limit</td>
<td>[0, 180] degrees</td>
<td></td>
</tr>
<tr>
<td>Change object locator scale search mode</td>
<td>sINT</td>
<td>35</td>
<td>No</td>
<td>int mode</td>
<td>0=off, 1=on</td>
<td></td>
</tr>
<tr>
<td>Get object locator scale search mode</td>
<td>gINT</td>
<td>35</td>
<td>Yes</td>
<td>int mode</td>
<td>0=off, 1=on</td>
<td></td>
</tr>
<tr>
<td>Change object locator robustness</td>
<td>sINT</td>
<td>36</td>
<td>No</td>
<td>int rob</td>
<td>0=High robustness, 1=Normal, 2=High speed</td>
<td></td>
</tr>
<tr>
<td>Get object locator robustness</td>
<td>gINT</td>
<td>36</td>
<td>Yes</td>
<td>int rob</td>
<td>0=High robustness, 1=Normal, 2=High speed</td>
<td></td>
</tr>
<tr>
<td>Change object locator accuracy</td>
<td>sINT</td>
<td>37</td>
<td>No</td>
<td>int acc</td>
<td>0=High accuracy, 1=Normal, 2=High speed</td>
<td></td>
</tr>
<tr>
<td>Get object locator accuracy</td>
<td>gINT</td>
<td>37</td>
<td>Yes</td>
<td>int acc</td>
<td>0=High accuracy, 1=Normal, 2=High speed</td>
<td></td>
</tr>
<tr>
<td>Move and rotate object locator</td>
<td>sINT</td>
<td>38</td>
<td>No</td>
<td>int x, int y, int angle</td>
<td>x, y = pixels, angle = degrees. Arguments are delta values as compared to the current position and angle. These can be negative</td>
<td></td>
</tr>
<tr>
<td>Get object locator position and rotation</td>
<td>gINT</td>
<td>38</td>
<td>Yes</td>
<td>int x, int y, int angle</td>
<td>x, y = pixels, angle = degrees. Return values are absolute values for the center of the ROI. These can be negative as compared to the origin</td>
<td></td>
</tr>
<tr>
<td>Get object locator width and height</td>
<td>gINT</td>
<td>39</td>
<td>Yes</td>
<td>int w, int height</td>
<td>[0-640], [0-480]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change blob intensity thresholds</td>
<td>sINT</td>
<td>48</td>
<td>No</td>
<td>int index, int min, int max</td>
<td></td>
<td>[0, 7], [0, 255], min&lt;=$max</td>
</tr>
</tbody>
</table>

©SICK AG • Identification & Measuring • www.sick.com • All rights reserved
Subject to change without notice
<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get blob intensity thresholds</td>
<td>gINT</td>
<td>48</td>
<td>Yes</td>
<td>int index</td>
<td>- int min, int max</td>
<td>[0, 7], [0, 255], [0, 255]</td>
</tr>
<tr>
<td>Change blob area thresholds</td>
<td>sINT</td>
<td>49</td>
<td>No</td>
<td>int index, int min, int max</td>
<td>[0, 7], [10, 307200] pixels, min&lt;=max, [10, 307200] pixels, min&lt;=max</td>
<td></td>
</tr>
<tr>
<td>Get blob area thresholds</td>
<td>gINT</td>
<td>49</td>
<td>Yes</td>
<td>int index</td>
<td>- int min, int max</td>
<td>[0, 7], [10, 307200] pixels, [10, 307200] pixels</td>
</tr>
<tr>
<td>Change blob angle thresholds</td>
<td>sINT</td>
<td>50</td>
<td>No</td>
<td>int index, int angle, int angle tolerance</td>
<td>[0, 7], [0, 180], [0, 90]</td>
<td></td>
</tr>
<tr>
<td>Get blob angle thresholds</td>
<td>gINT</td>
<td>50</td>
<td>Yes</td>
<td>int index</td>
<td>- int ref, int tol</td>
<td>[0, 7], [0, 180], [0, 90]</td>
</tr>
<tr>
<td>Change structure criteria thresholds</td>
<td>sINT</td>
<td>53</td>
<td>No</td>
<td>int index, int min, int max</td>
<td>[0, 7], [0, 100000], min&lt;=max, [0, 100000], min&lt;=max</td>
<td></td>
</tr>
<tr>
<td>Get structure criteria thresholds</td>
<td>gINT</td>
<td>53</td>
<td>Yes</td>
<td>int index</td>
<td>- int min, int max</td>
<td>[0, 7], [0, 100000], [0, 100000]</td>
</tr>
<tr>
<td>Change blob edge strength</td>
<td>sINT</td>
<td>54</td>
<td>No</td>
<td>int index, int strength</td>
<td>[0, 7], [0, 100] percent</td>
<td></td>
</tr>
<tr>
<td>Get blob edge strength</td>
<td>gINT</td>
<td>54</td>
<td>Yes</td>
<td>int index</td>
<td>- int strength</td>
<td>[0, 7], [0, 100] percent</td>
</tr>
<tr>
<td>Change ambient light compensation mode</td>
<td>sINT</td>
<td>55</td>
<td>No</td>
<td>int index, int mode</td>
<td>[0, 7], 0=off, 1=on</td>
<td></td>
</tr>
<tr>
<td>Get ambient light compensation mode</td>
<td>gINT</td>
<td>55</td>
<td>Yes</td>
<td>int index</td>
<td>- int mode</td>
<td>[0, 7], 0=off, 1=on</td>
</tr>
<tr>
<td>Change blob search method</td>
<td>sINT</td>
<td>56</td>
<td>No</td>
<td>int index, int method</td>
<td>[0, 7], 0=High quality, 1=Normal, 2=High speed</td>
<td></td>
</tr>
<tr>
<td>Get blob search method</td>
<td>gINT</td>
<td>56</td>
<td>Yes</td>
<td>int index</td>
<td>- int method</td>
<td>[0, 7], 0=High quality, 1=Normal, 2=High speed</td>
</tr>
<tr>
<td>Description</td>
<td>Command</td>
<td>Identifier</td>
<td>Usable in Run mode</td>
<td>Arguments (^a)</td>
<td>Return values</td>
<td>Range</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>---------</td>
<td>------------</td>
<td>--------------------</td>
<td>------------------</td>
<td>---------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Move and rotate blob locator</td>
<td>sINT</td>
<td>58</td>
<td>No</td>
<td>int index, int x, int y, int angle</td>
<td>[0, 7]. x, y = pixels, angle = degrees. Arguments are delta values compared to the current position and angle. These can be negative</td>
<td></td>
</tr>
<tr>
<td>Get blob ROI position and rotation</td>
<td>gINT</td>
<td>58</td>
<td>Yes</td>
<td>int index, int x, int y, int angle</td>
<td>[0, 7] x, y = pixels, angle = degrees. Return values are absolute values for the center of the ROI. These can be negative as compared to the origin.</td>
<td></td>
</tr>
<tr>
<td>Set number of blobs</td>
<td>sINT</td>
<td>59</td>
<td>No</td>
<td>int index, int min, int max</td>
<td>[0, 7] [0, 16] min≤max [0, 16] min≤max</td>
<td></td>
</tr>
<tr>
<td>Get number of blobs</td>
<td>gINT</td>
<td>59</td>
<td>Yes</td>
<td>int index, int min, int max</td>
<td>[0, 7] [0, 16] min≤max [0, 16] min≤max</td>
<td></td>
</tr>
<tr>
<td>Get blob area thresholds in (\text{mm}^2)</td>
<td>gINT</td>
<td>60</td>
<td>Yes</td>
<td>int index, int min<em>1000, int max</em>1000</td>
<td>[0, 7] [10 * pixel size - 307200 * pixel size], min≤max [10 * pixel size - 307200 * pixel size], min≤max</td>
<td></td>
</tr>
<tr>
<td>Get blob region width and height</td>
<td>gINT</td>
<td>61</td>
<td>Yes</td>
<td>int blob, int w, int height</td>
<td>[0-7], [0-640], [0-480]</td>
<td></td>
</tr>
</tbody>
</table>

\(^a\)For information about blob indexing see Section B.2.2, “Blob indexing” (page 54)

**Table B.9 Command channel functions - Bead**

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments (^a)</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get bead edge strength</td>
<td>gINT</td>
<td>250</td>
<td>Yes</td>
<td>int index, int strength</td>
<td>[0, 7],</td>
<td></td>
</tr>
<tr>
<td>Set bead edge strength</td>
<td>sINT</td>
<td>250</td>
<td>No</td>
<td>int index, int strength</td>
<td>[0, 7], [0, 255]</td>
<td></td>
</tr>
<tr>
<td>Get bead intensity</td>
<td>gINT</td>
<td>251</td>
<td>Yes</td>
<td>int index, int intensity</td>
<td>[0, 7],</td>
<td></td>
</tr>
<tr>
<td>Set bead intensity</td>
<td>gINT</td>
<td>251</td>
<td>No</td>
<td>int index, int intensity</td>
<td>[0, 7], [0, 1]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Command</td>
<td>Identifier</td>
<td>Usable in Run mode</td>
<td>Arguments</td>
<td>Return values</td>
<td>Range</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>---------</td>
<td>------------</td>
<td>---------------------</td>
<td>-----------</td>
<td>---------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Get bead min search region width</td>
<td>gINT</td>
<td>252</td>
<td>Yes</td>
<td>int index, int width</td>
<td>int width</td>
<td>[0, 7],</td>
</tr>
<tr>
<td>Set bead min search region width</td>
<td>sINT</td>
<td>252</td>
<td>No</td>
<td>int index, int width</td>
<td></td>
<td>[0, 7], [0, 640]</td>
</tr>
<tr>
<td>Get bead max search region width</td>
<td>gINT</td>
<td>253</td>
<td>Yes</td>
<td>int index, int width</td>
<td>int width</td>
<td>[0, 7],</td>
</tr>
<tr>
<td>Set bead max search region width</td>
<td>sINT</td>
<td>253</td>
<td>No</td>
<td>int index, int width</td>
<td></td>
<td>[0, 7], [0, 640]</td>
</tr>
<tr>
<td>Get bead position tolerance</td>
<td>gINT</td>
<td>254</td>
<td>Yes</td>
<td>int index, int tolerance</td>
<td>int tolerance</td>
<td>[0, 7],</td>
</tr>
<tr>
<td>Set bead position tolerance</td>
<td>sINT</td>
<td>254</td>
<td>No</td>
<td>int index, int tolerance</td>
<td></td>
<td>[0, 7], [0, 320]</td>
</tr>
<tr>
<td>Get bead mean width thresholds</td>
<td>gINT</td>
<td>255</td>
<td>Yes</td>
<td>int index</td>
<td>int min, int max</td>
<td>[0, 7]</td>
</tr>
<tr>
<td>Set bead mean width thresholds</td>
<td>sINT</td>
<td>255</td>
<td>No</td>
<td>int index, int min, int max</td>
<td></td>
<td>[0, 7], [0, 640], [0, 640]</td>
</tr>
<tr>
<td>Get bead max no of consecutive too narrow</td>
<td>gINT</td>
<td>256</td>
<td>Yes</td>
<td>int index</td>
<td>int max-TooNarrow</td>
<td>[0, 7],</td>
</tr>
<tr>
<td>Set bead max no of consecutive too narrow</td>
<td>sINT</td>
<td>256</td>
<td>No</td>
<td>int index, int max</td>
<td></td>
<td>[0, 7], [0, 256]</td>
</tr>
<tr>
<td>Get bead max no of consecutive too wide</td>
<td>gINT</td>
<td>257</td>
<td>Yes</td>
<td>int index</td>
<td>int max-TooWide</td>
<td>[0, 7],</td>
</tr>
<tr>
<td>Set bead max no of consecutive too wide</td>
<td>sINT</td>
<td>257</td>
<td>No</td>
<td>int index, int max</td>
<td></td>
<td>[0, 7], [0, 256]</td>
</tr>
<tr>
<td>Move bead</td>
<td>sINT</td>
<td>258</td>
<td>No</td>
<td>int index, int corner, int delta-x, int delta-y</td>
<td>[0, 7], delta-x, delta-y = pixels. Arguments are delta values. These can be negative as compared to the origin.</td>
<td></td>
</tr>
<tr>
<td>Get bead corner</td>
<td>gINT</td>
<td>259</td>
<td>Yes</td>
<td>int index int corner</td>
<td>int x, int y</td>
<td>[0, 7], x, y = pixels, Arguments are delta values. These can be negative as compared to the origin.</td>
</tr>
</tbody>
</table>
| Move bead corner                                      | sINT    | 259        | No                  | int index, int corner, int delta-x, int delta-y | [0, 7], [0, 15], delta-x, delta-y = pixels. Arguments are delta values. These can be
## Command channel functions - Polygon

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change polygon position search tolerance</td>
<td>sINT</td>
<td>64</td>
<td>No</td>
<td>int index, int index</td>
<td>[0, 7], [5, 100] pixels for single edge [5, 100] pixels for polygon</td>
<td></td>
</tr>
<tr>
<td>Get polygon position search tolerance</td>
<td>gINT</td>
<td>64</td>
<td>Yes</td>
<td>int index</td>
<td>- int tol</td>
<td>[0, 7], [5, 100] pixels for single edge [5, 100] pixels for polygon</td>
</tr>
<tr>
<td>Change polygon flexibility search tolerance</td>
<td>sINT</td>
<td>65</td>
<td>No</td>
<td>int index, int index</td>
<td>[0, 7], [0, 100] pixels</td>
<td></td>
</tr>
<tr>
<td>Get polygon flexibility search tolerance</td>
<td>gINT</td>
<td>65</td>
<td>Yes</td>
<td>int index</td>
<td>- int tol</td>
<td>[0, 7], [0, 100] pixels</td>
</tr>
<tr>
<td>Change polygon score threshold</td>
<td>sINT</td>
<td>66</td>
<td>No</td>
<td>int index, int threshold</td>
<td>[0, 7], [0, 100] pixels</td>
<td></td>
</tr>
</tbody>
</table>

*For information about bead indexing see Section B.2.3, “Bead indexing” (page 54)
<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments*</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get polygon score threshold</td>
<td>gINT</td>
<td>66</td>
<td>Yes</td>
<td>int index</td>
<td>-</td>
<td>[0, 7], [0, 100] pixels</td>
</tr>
<tr>
<td>Change polygon margin</td>
<td>sINT</td>
<td>67</td>
<td>No</td>
<td>int index, int margin</td>
<td></td>
<td>[0, 7], [0, 20] pixels</td>
</tr>
<tr>
<td>Get polygon margin</td>
<td>gINT</td>
<td>67</td>
<td>Yes</td>
<td>int index</td>
<td>-</td>
<td>[0, 7], [0, 20] pixels</td>
</tr>
<tr>
<td>Change polygon defect detection width</td>
<td>sINT</td>
<td>68</td>
<td>No</td>
<td>int index, int width</td>
<td></td>
<td>[0, 7], [0, 100] pixels</td>
</tr>
<tr>
<td>Get polygon defect detection width</td>
<td>gINT</td>
<td>68</td>
<td>Yes</td>
<td>int index</td>
<td>-</td>
<td>[0, 7], [0, 100] pixels</td>
</tr>
<tr>
<td>Change polygon defect intensity range thresholds</td>
<td>sINT</td>
<td>69</td>
<td>No</td>
<td>int index, int min, int max</td>
<td></td>
<td>[0, 7], [0, 255], min&lt;=max, [0, 255], min&lt;=max</td>
</tr>
<tr>
<td>Get polygon defect intensity range thresholds</td>
<td>gINT</td>
<td>69</td>
<td>Yes</td>
<td>int index</td>
<td>-</td>
<td>[0, 7], [0, 255], [0, 255]</td>
</tr>
<tr>
<td>Change polygon max defects threshold</td>
<td>sINT</td>
<td>70</td>
<td>No</td>
<td>int index, int max</td>
<td></td>
<td>[0, 7], [0, 100] pixels</td>
</tr>
<tr>
<td>Get polygon max defects threshold</td>
<td>gINT</td>
<td>70</td>
<td>Yes</td>
<td>int index</td>
<td>-</td>
<td>[0, 7], [0, 100] pixels</td>
</tr>
<tr>
<td>Change polygon defect detection mode</td>
<td>sINT</td>
<td>71</td>
<td>No</td>
<td>int index, int mode</td>
<td></td>
<td>[0, 7], O=off, 1=on</td>
</tr>
<tr>
<td>Get polygon defect detection mode</td>
<td>gINT</td>
<td>71</td>
<td>Yes</td>
<td>int index</td>
<td>-</td>
<td>[0, 7], O=off, 1=on</td>
</tr>
<tr>
<td>Move polygon</td>
<td>sINT</td>
<td>72</td>
<td>No</td>
<td>int index, int x, int y</td>
<td></td>
<td>[0, 7], x, y = pixels, Arguments are delta values. These can be negative as compared to the origin.</td>
</tr>
<tr>
<td>Reserved for future use</td>
<td>gINT</td>
<td>72</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Move polygon corner</td>
<td>sINT</td>
<td>73</td>
<td>No</td>
<td>int index, int corner, int delta-x, int delta-y</td>
<td>[0, 7], [0,15] delta-x, delta-y = pixels, Arguments are delta values. These can be negative as...</td>
<td></td>
</tr>
</tbody>
</table>
### Range

<table>
<thead>
<tr>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
</tr>
</thead>
<tbody>
<tr>
<td>gINT</td>
<td>73</td>
<td>Yes</td>
<td>int index, int corner</td>
<td>[0-7], [0-15]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x, y = pixels</td>
<td></td>
</tr>
</tbody>
</table>

For information about polygon indexing see Section B.2.4, “Polygon indexing” (page 54).

### Table B.11 Command channel functions - Pixel counter

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change pixel counter intensity range thresholds</td>
<td>sINT</td>
<td>80</td>
<td>No</td>
<td>int index, int min, int max</td>
<td>[0, 63], [0, 255], min&lt;=max</td>
</tr>
<tr>
<td>Get pixel counter intensity range thresholds</td>
<td>gINT</td>
<td>80</td>
<td>Yes</td>
<td>int index</td>
<td>[0, 63], [0, 255]</td>
</tr>
<tr>
<td>Change No. of pixels in range thresholds</td>
<td>sINT</td>
<td>81</td>
<td>No</td>
<td>int index, int min, int max</td>
<td>[0, 63], [0, ROI size^{b}], pixels, min&lt;=max</td>
</tr>
<tr>
<td>Get No. of pixels in range thresholds</td>
<td>gINT</td>
<td>81</td>
<td>Yes</td>
<td>int index</td>
<td>[0, 63], [0, ROI size^{b}] pixels</td>
</tr>
</tbody>
</table>

For information about tools indexing see Section B.2.5, “Tools Indexing” (page 54).

ROI size is the size of the pixel counter ROI in the reference object. Value can be fetched with the command gINT 87.

### Table B.12 Command channel functions - Edge pixel counter

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change edge pixel counter edge strength</td>
<td>sINT</td>
<td>82</td>
<td>No</td>
<td>int index, int strength</td>
<td>[0, 63], [0, 100]</td>
</tr>
<tr>
<td>Get edge pixel counter edge strength</td>
<td>gINT</td>
<td>82</td>
<td>Yes</td>
<td>int index</td>
<td>[0, 63], [0, 100]</td>
</tr>
<tr>
<td>Change No. of edge pixels thresholds</td>
<td>sINT</td>
<td>83</td>
<td>No</td>
<td>int index, int min, int max</td>
<td>[0, 63], [0, ROI size^{b}] pixels, min&lt;=max</td>
</tr>
</tbody>
</table>

ROI size is the size of the pixel counter ROI in the reference object. Value can be fetched with the command gINT 87.

©SICK AG • Identification & Measuring • www.sick.com • All rights reserved
### Description

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get No. of edge pixels thresholds</td>
<td>gINT</td>
<td>83</td>
<td>Yes</td>
<td>int index</td>
<td>[0, 63], [0, ROI size(^b)] pixels, (\min \leq \max)</td>
<td></td>
</tr>
</tbody>
</table>

\(^a\)For information about tools indexing see Section B.2.5, “Tools indexing” (page 54)

\(^b\)ROI size is the size of the edge pixel counter ROI in the reference object. Value can be fetched with the command gINT 87

### Table B.13  Command channel functions - Pattern

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change pattern position tolerance</td>
<td>sINT</td>
<td>84</td>
<td>No</td>
<td>int index, int tolerance</td>
<td>[0, 63], [0, 4] pixels</td>
<td></td>
</tr>
<tr>
<td>Get pattern position tolerance</td>
<td>gINT</td>
<td>84</td>
<td>Yes</td>
<td>int index</td>
<td>- int tol</td>
<td>[0, 63], [0, 4] pixels</td>
</tr>
<tr>
<td>Change pattern score threshold</td>
<td>sINT</td>
<td>85</td>
<td>No</td>
<td>int index, int threshold</td>
<td>[0, 63], [0, 100] percent</td>
<td></td>
</tr>
<tr>
<td>Get pattern score threshold</td>
<td>gINT</td>
<td>85</td>
<td>Yes</td>
<td>int index</td>
<td>- int threshold</td>
<td>[0, 63], [0, 100] percent</td>
</tr>
</tbody>
</table>

\(^a\)For information about tools indexing see Section B.2.5, “Tools indexing” (page 54)

### Table B.14  Command channel functions - Edge tool

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change edge tool edge contrast</td>
<td>sINT</td>
<td>150</td>
<td>No</td>
<td>int index, int contrast</td>
<td>[0, 63], [0, 100]</td>
<td></td>
</tr>
<tr>
<td>Get edge tool edge contrast</td>
<td>gINT</td>
<td>150</td>
<td>Yes</td>
<td>int index</td>
<td>- int contrast</td>
<td>[0, 63], [0, 100]</td>
</tr>
<tr>
<td>Change edge tool line fit criteria</td>
<td>sINT</td>
<td>151</td>
<td>No</td>
<td>int index, int criterion</td>
<td>[0, 63], [0, 2] ((0 = \text{best fit, } 1 = \text{first, } 2 = \text{last}))</td>
<td></td>
</tr>
<tr>
<td>Get edge tool line fit criteria</td>
<td>gINT</td>
<td>151</td>
<td>Yes</td>
<td>int index</td>
<td>- int criterion</td>
<td>[0, 63], [0, 2] ((0 = \text{best fit, } 1 = \text{first, } 2 = \text{last}))</td>
</tr>
<tr>
<td>Change edge tool polarity</td>
<td>sINT</td>
<td>152</td>
<td>No</td>
<td>int index, int polarity</td>
<td>[0, 63], [0, 2] ((0 = \text{any, } 1 = \text{bright to dark, } 2 = \text{dark to bright}))</td>
<td></td>
</tr>
</tbody>
</table>
### Table B.15 Command channel functions - Find maximum

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get edge tool polarity</td>
<td>glINT</td>
<td>152</td>
<td>Yes</td>
<td>int index</td>
<td>- int criterion</td>
<td>[0, 63], [0, 2] (0 = any, 1 = bright to dark, 2 = dark to bright)</td>
</tr>
<tr>
<td>Change edge tool score threshold</td>
<td>sINT</td>
<td>153</td>
<td>No</td>
<td>int index, int threshold</td>
<td></td>
<td>[0, 63], [0, 100]</td>
</tr>
<tr>
<td>Get edge tool score threshold</td>
<td>glINT</td>
<td>153</td>
<td>Yes</td>
<td>int index, int threshold</td>
<td></td>
<td>[0, 63], [0, 100]</td>
</tr>
</tbody>
</table>

*For information about tools indexing see Section B.2.5, “Tools indexing” (page 54)*

### Table B.16 Command channel functions - Circle

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change circle edge contrast</td>
<td>sINT</td>
<td>160</td>
<td>No</td>
<td>int index, int contrast</td>
<td></td>
<td>[0, 63], [0, 100]</td>
</tr>
</tbody>
</table>

*For information about tools indexing see Section B.2.5, “Tools indexing” (page 54)*
<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get circle edge contrast</td>
<td>gINT</td>
<td>160</td>
<td>Yes</td>
<td>int index</td>
<td>-</td>
<td>[0, 63], [0, 100]</td>
</tr>
<tr>
<td>Change circle diameter thresholds (without search enabled)</td>
<td>sINT</td>
<td>161</td>
<td>No</td>
<td>int index, int min, int max, int unit</td>
<td>[0, 63], [1, max]</td>
<td>[5, circle within FOV]</td>
</tr>
<tr>
<td>Change circle diameter thresholds (with search enabled)</td>
<td>sINT</td>
<td>161</td>
<td>No</td>
<td>int index, int min, int max, int unit</td>
<td>[0, 63], [shortest search region side/3, max]</td>
<td>[min, shortest search region side]</td>
</tr>
<tr>
<td>Get circle diameter thresholds</td>
<td>gINT</td>
<td>161</td>
<td>Yes</td>
<td>int index, int unit</td>
<td>- int min, int max</td>
<td>[0, 63], [0, 1] (0 = pixels, 1 = millimeters)</td>
</tr>
<tr>
<td>Change circle line fit criteria</td>
<td>sINT</td>
<td>162</td>
<td>No</td>
<td>int index, int criterion</td>
<td></td>
<td>[0, 63], [0, 2] (0 = strongest, 1 = first, 2 = last)</td>
</tr>
<tr>
<td>Get circle line fit criteria</td>
<td>gINT</td>
<td>162</td>
<td>Yes</td>
<td>int index</td>
<td>- int criterion</td>
<td>[0, 63], [0, 2] (0 = strongest, 1 = first, 2 = last)</td>
</tr>
<tr>
<td>Change circle polarity</td>
<td>sINT</td>
<td>163</td>
<td>No</td>
<td>int index, int polarity</td>
<td></td>
<td>[0, 63], [0, 2] (0 = any, 1 = bright to dark, 2 = dark to bright)</td>
</tr>
<tr>
<td>Get circle polarity</td>
<td>gINT</td>
<td>163</td>
<td>Yes</td>
<td>int index</td>
<td>- int criterion</td>
<td>[0, 63], [0, 2] (0 = any, 1 = bright to dark, 2 = dark to bright)</td>
</tr>
<tr>
<td>Change circle robustness</td>
<td>sINT</td>
<td>164</td>
<td>No</td>
<td>int index, int robustness</td>
<td></td>
<td>[0, 63], [0, 4] (0 = high robustness to 4 = high speed)</td>
</tr>
<tr>
<td>Get circle robustness</td>
<td>gINT</td>
<td>164</td>
<td>Yes</td>
<td>int index</td>
<td>- int criterion</td>
<td>[0, 63], [0, 4] (0 = high robustness to 4 = high speed)</td>
</tr>
<tr>
<td>Description</td>
<td>Command</td>
<td>Identifier</td>
<td>Usable in Run mode</td>
<td>Arguments a</td>
<td>Return values</td>
<td>Range</td>
</tr>
<tr>
<td>--------------------------------------------------------------</td>
<td>---------</td>
<td>------------</td>
<td>--------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Change circle score threshold</td>
<td>sINT</td>
<td>165</td>
<td>No</td>
<td>int index,</td>
<td></td>
<td>[0, 63],</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>int threshold</td>
<td></td>
<td>[0, 100]</td>
</tr>
<tr>
<td>Get circle score threshold</td>
<td>gINT</td>
<td>165</td>
<td>Yes</td>
<td>int index</td>
<td>int threshold</td>
<td>[0, 63],</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0, 100]</td>
</tr>
<tr>
<td>Change circle quality</td>
<td>sINT</td>
<td>166</td>
<td>No</td>
<td>int index,</td>
<td></td>
<td>[0, 63],</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>int quality</td>
<td></td>
<td>[0, 6], (0 = tolerant to 6 = exact)</td>
</tr>
<tr>
<td>Get circle quality</td>
<td>gINT</td>
<td>166</td>
<td>Yes</td>
<td>int index</td>
<td>int quality</td>
<td>[0, 63],</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0, 6], (0 = tolerant to 6 = exact)</td>
</tr>
<tr>
<td>Change circle diameter offset compensation</td>
<td>sINT</td>
<td>167</td>
<td>No</td>
<td>int index,</td>
<td>int offset*1000 int unit</td>
<td>[0, 63], [1000000, 1000000]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>int unit</td>
<td>[0, 1], (0 = pixels, 1 = millimeters)</td>
</tr>
<tr>
<td>Get circle diameter offset compensation</td>
<td>gINT</td>
<td>167</td>
<td>Yes</td>
<td>int index,</td>
<td>int unit</td>
<td>[0, 63],</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>int unit</td>
<td>[0, 1], (0 = pixels, 1 = millimeters)</td>
</tr>
<tr>
<td>Change circle diameter tolerance thresholds</td>
<td>sINT</td>
<td>169</td>
<td>No</td>
<td>int index,</td>
<td>int min<em>1000 int max</em>1000 int unit</td>
<td>[0, 63], [6400000] min=max [0, 640000]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>int unit</td>
<td>[0, 1], (0 = pixels, 1 = millimeters)</td>
</tr>
<tr>
<td>Get circle diameter tolerance thresholds</td>
<td>gINT</td>
<td>169</td>
<td>Yes</td>
<td>int index,</td>
<td>int unit</td>
<td>[0, 63],</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>int unit</td>
<td>[0, 1], (0 = pixels, 1 = millimeters)</td>
</tr>
</tbody>
</table>

aFor information about tools indexing see Section B.2.5, “Tools indexing” (page 54)

bWhen creating the circle without search, the max diameter that can be set is the diameter that keeps the circle within the configured FOV. E.g. if using the full resolution FOV and placing the circle in the center of the image, the max value without search can be set to 480 pixels.

The table below lists the move commands for the Pixel counter, Edge pixel counter, Pattern, Edge, and Circle tools.
### Table B.17  Command channel functions - Common commands for Pixel counter, Edge pixel counter, Pattern, Edge, and Circle tools

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move and rotate inspection</td>
<td>sINT</td>
<td>86</td>
<td>No</td>
<td>int index, int x, int y, int angle</td>
<td>[0, 63], x, y = pixels, angle = degrees. Arguments are delta values. These can be negative as compared to the origin.</td>
<td></td>
</tr>
<tr>
<td>Get inspection position and rotation</td>
<td>gINT</td>
<td>86</td>
<td>Yes</td>
<td>int x, int y, int angle</td>
<td>[0, 63], x, y = pixels, angle = degrees. Return values are absolute values for the center of the ROI. These can be negative as compared to the origin.</td>
<td></td>
</tr>
<tr>
<td>Get number of pixels in ROI</td>
<td>gINT</td>
<td>87</td>
<td>Yes</td>
<td>int index, int pixels</td>
<td>[0, 63], Number of pixels in the reference object’s ROI</td>
<td></td>
</tr>
</tbody>
</table>

*aFor information about tools indexing see Section B.2.5, “Tools indexing” (page 54)*

### Table B.18  Command channel functions - Distance

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change distance measurement thresholds</td>
<td>sINT</td>
<td>170</td>
<td>No</td>
<td>int index, int min<em>1000 int max</em>1000 int unit</td>
<td>[0, 63], [0, VGA image diagonal] [0, VGA image diagonal] [0, 1] (0 = pixels, 1 = millimeters) min ≤ max</td>
<td></td>
</tr>
<tr>
<td>Get distance measurement thresholds</td>
<td>gINT</td>
<td>170</td>
<td>Yes</td>
<td>int index, int unit</td>
<td>[0, 63], [0, 1] (0 = pixels, 1 = millimeters) [0, VGA image diagonal] [0, VGA image diagonal]</td>
<td></td>
</tr>
<tr>
<td>Change distance offset compensation</td>
<td>sINT</td>
<td>173</td>
<td>No</td>
<td>int index, int offset*1000 int unit</td>
<td>[0, 63], [-1000000, 1000000] [0, 1] (0 = pixels, 1 = millimeters)</td>
<td></td>
</tr>
</tbody>
</table>
### Table B.19  Command channel functions - Angle

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments a</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get distance offset compensation</td>
<td>gINT</td>
<td>173</td>
<td>Yes</td>
<td>int index, int unit</td>
<td>- int offset*1000</td>
<td>[0, 63], [0, 1] (0 = pixels, 1 = millimeters)</td>
</tr>
<tr>
<td>Change angle measurement thresholds</td>
<td>sINT</td>
<td>180</td>
<td>No</td>
<td>int index, int min<em>1000 int max</em>1000</td>
<td>- int offset*1000</td>
<td>[0, 63], [0, 100] [0,180000], min ≤ max</td>
</tr>
<tr>
<td>Get angle measurement thresholds</td>
<td>gINT</td>
<td>180</td>
<td>Yes</td>
<td>int index</td>
<td>- int min<em>1000 int max</em>1000</td>
<td>[0, 63], [0, 100] [0,180000], min ≤ max</td>
</tr>
<tr>
<td>Change angle offset compensation</td>
<td>sINT</td>
<td>181</td>
<td>No</td>
<td>int index, int offset*1000</td>
<td>- int offset*1000</td>
<td>[0, 63], [-1000000,1000000]</td>
</tr>
<tr>
<td>Get angle offset compensation</td>
<td>gINT</td>
<td>181</td>
<td>Yes</td>
<td>int index</td>
<td>- int offset*1000</td>
<td>[0, 63]</td>
</tr>
</tbody>
</table>

a

For information about tools indexing see Section B.2.5, “Tools indexing” (page 54)

### Table B.20  Command channel functions - Edge counter

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments a</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get Edge counter edge contrast</td>
<td>gINT</td>
<td>190</td>
<td>Yes</td>
<td>int index</td>
<td>- int contrast</td>
<td>[0, 63], [0, 100]</td>
</tr>
<tr>
<td>Change Edge counter edge contrast</td>
<td>sINT</td>
<td>190</td>
<td>No</td>
<td>int index, int contrast</td>
<td>- int contrast</td>
<td>[0, 63], [0, 100]</td>
</tr>
<tr>
<td>Get Edge counter edge quality</td>
<td>gINT</td>
<td>191</td>
<td>Yes</td>
<td>int index</td>
<td>- int quality</td>
<td>[0, 63], [0, 6]</td>
</tr>
<tr>
<td>Change Edge counter edge quality</td>
<td>sINT</td>
<td>191</td>
<td>No</td>
<td>int index, int robustness</td>
<td>- int quality</td>
<td>[0, 63], [0, 6]</td>
</tr>
<tr>
<td>Description</td>
<td>Command</td>
<td>Identifier</td>
<td>Usable in Run mode</td>
<td>Arguments</td>
<td>Return values</td>
<td>Range</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>---------</td>
<td>------------</td>
<td>--------------------</td>
<td>-----------</td>
<td>---------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>Change Edge counter feature width</td>
<td>siNT</td>
<td>192</td>
<td>No</td>
<td>int, int min<em>10, int max</em>10, int unit</td>
<td>[0, 63], [0, VGA image diagonal]</td>
<td>[0, 63], [0, VGA image diagonal], [0, 1] (0 = pixels, 1 = millimeters) min ≤ max</td>
</tr>
<tr>
<td>Get Edge counter feature width</td>
<td>gINT</td>
<td>192</td>
<td>Yes</td>
<td>int, int unit</td>
<td>[0, 63], [0, 1] (0 = pixels, 1 = millimeters)</td>
<td>[0, VGA image diagonal]</td>
</tr>
<tr>
<td>Change Edge counter feature type</td>
<td>siNT</td>
<td>193</td>
<td>No</td>
<td>int, int type</td>
<td>[0, 63], (0 = bright, 1 = dark, 2 = single edge)</td>
<td></td>
</tr>
<tr>
<td>Get Edge counter feature type</td>
<td>gINT</td>
<td>193</td>
<td>Yes</td>
<td>int, int type</td>
<td>[0, 63], (0 = bright, 1 = dark, 2 = single edge)</td>
<td></td>
</tr>
<tr>
<td>Change edge count min max features</td>
<td>siNT</td>
<td>194</td>
<td>No</td>
<td>int, int min, int max</td>
<td>[0, 63], [0, 63] min ≤ max</td>
<td></td>
</tr>
<tr>
<td>Get edge count min max features</td>
<td>gINT</td>
<td>194</td>
<td>Yes</td>
<td>int, int min, int max</td>
<td>[0, 63], [0, 64]</td>
<td></td>
</tr>
<tr>
<td>Change edge count search method robustness</td>
<td>siNT</td>
<td>195</td>
<td>No</td>
<td>int, int robustness</td>
<td>[0, 63], [0, 4]</td>
<td></td>
</tr>
<tr>
<td>Get edge count search method robustness</td>
<td>gINT</td>
<td>195</td>
<td>Yes</td>
<td>int, int robustness</td>
<td>[0, 63], [0, 4]</td>
<td></td>
</tr>
<tr>
<td>Change edge count pitch threshold</td>
<td>siNT</td>
<td>197</td>
<td>No</td>
<td>int, int min<em>1000, int max</em>1000, int unit</td>
<td>[0, 63], [0, VGA image diagonal]</td>
<td></td>
</tr>
</tbody>
</table>

a

b

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get edge count pitch threshold</td>
<td>gINT</td>
<td>197</td>
<td>Yes</td>
<td>int index, int unit</td>
<td>-</td>
<td>[0, 63], [0, 1] (0 = pixels, 1 = millimeters) [0, VGA image diagonal] [0, VGA image diagonal]</td>
</tr>
<tr>
<td>Change edge count single edge polarity</td>
<td>sINT</td>
<td>199</td>
<td>No</td>
<td>int index, int polarity</td>
<td></td>
<td>[0, 63], [0, 2] (0=any, 1=bright to dark, 2=dark to bright)</td>
</tr>
<tr>
<td>Get edge count single edge polarity</td>
<td>gINT</td>
<td>199</td>
<td>Yes</td>
<td>int index, int polarity</td>
<td>- int polarity</td>
<td>[0, 63], [0, 2] (0=any, 1=bright to dark, 2=dark to bright)</td>
</tr>
</tbody>
</table>

For information about tools indexing see Section B.2.5, “Tools indexing” (page 54)

*“0” is the only valid value for circular.*

Table B.21 Command channel functions - Get names

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>Identifier</th>
<th>Usable in Run mode</th>
<th>Arguments</th>
<th>Return values</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get name of device</td>
<td>gSTR</td>
<td>1</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Get name of reference object</td>
<td>gSTR</td>
<td>2</td>
<td>Yes</td>
<td>int referenceObject</td>
<td>- string name</td>
<td>[0,31]</td>
</tr>
<tr>
<td>Get name of Object Locator</td>
<td>gSTR</td>
<td>3</td>
<td>Yes</td>
<td>int referenceObject</td>
<td>- string name</td>
<td>[0,31]</td>
</tr>
<tr>
<td>Get name of Pixel counter</td>
<td>gSTR</td>
<td>4</td>
<td>Yes</td>
<td>int referenceObject int index</td>
<td>- string name</td>
<td>[0, 31], [0, 63]</td>
</tr>
<tr>
<td>Get name of Edge pixel counter</td>
<td>gSTR</td>
<td>5</td>
<td>Yes</td>
<td>int referenceObject int index</td>
<td>- string name</td>
<td>[0, 31], [0, 63]</td>
</tr>
<tr>
<td>Get name of Pattern</td>
<td>gSTR</td>
<td>6</td>
<td>Yes</td>
<td>int referenceObject int index</td>
<td>- string name</td>
<td>[0, 31], [0, 63]</td>
</tr>
<tr>
<td>Get name of Blob</td>
<td>gSTR</td>
<td>7</td>
<td>Yes</td>
<td>int referenceObject int index</td>
<td>- string name</td>
<td>[0, 31], [0, 7]</td>
</tr>
<tr>
<td>Get name of Polygon</td>
<td>gSTR</td>
<td>8</td>
<td>Yes</td>
<td>int referenceObject</td>
<td>- string name</td>
<td>[0, 31], [0, 63]</td>
</tr>
<tr>
<td>Description</td>
<td>Command</td>
<td>Identifier</td>
<td>Usable in Run mode</td>
<td>Arguments</td>
<td>Return values</td>
<td>Range</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------</td>
<td>------------</td>
<td>--------------------</td>
<td>---------------------------------</td>
<td>--------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Get name of edge tool</td>
<td>gSTR</td>
<td>9</td>
<td>Yes</td>
<td>int referenceObject, int index²</td>
<td>- string name</td>
<td>[0, 7]</td>
</tr>
<tr>
<td>Get name of circle</td>
<td>gSTR</td>
<td>10</td>
<td>Yes</td>
<td>int referenceObject, int index²</td>
<td>- string name</td>
<td>[0, 31], [0, 63]</td>
</tr>
<tr>
<td>Get name of Measure distance</td>
<td>gSTR</td>
<td>11</td>
<td>Yes</td>
<td>int referenceObject, int index²</td>
<td>- string name</td>
<td>[0, 31], [0, 63]</td>
</tr>
<tr>
<td>Get name of Measure angle</td>
<td>gSTR</td>
<td>12</td>
<td>Yes</td>
<td>int referenceObject, int index²</td>
<td>- string name</td>
<td>[0, 31], [0, 63]</td>
</tr>
<tr>
<td>Get Edge counter name</td>
<td>gSTR</td>
<td>13</td>
<td>Yes</td>
<td>int referenceObject, int index²</td>
<td>- string name</td>
<td>[0, 31], [0, 4]</td>
</tr>
<tr>
<td>Get all tool names (except the Object Locator)</td>
<td>gSTR</td>
<td>14</td>
<td>Yes</td>
<td>int referenceObject, int format</td>
<td>- string list of names</td>
<td>[0, 31], [0, 1]</td>
</tr>
</tbody>
</table>

²For information about tools indexing see Section B.2.5, “Tools indexing” (page 54)
³For information about blob indexing see Section B.2.2, “Blob indexing” (page 54)
⁴For information about polygon indexing see Section B.2.4, “Polygon indexing” (page 54)
⁵CSV (Comma separated values) in accordance with RFC 4180
⁶JSON (JavaScript Object Notation) in accordance with RFC 4627

### B.4 Error codes

The tables below list error codes that may result from commands or configuration of the device. The error codes are valid for EtherNet/IP, Ethernet Raw, and Web Server. Both error code and an explaining text are shown when using Ethernet Raw for configuring the device through the command channel. When using EtherNet/IP for configuring the device through the command channel will only view the error code.
Table B.22  Error codes - Command errors

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>8000</td>
<td>Index out of bounds, for example trying to use an image bank above 32</td>
</tr>
<tr>
<td>8001</td>
<td>Incorrect number of arguments, too many or too few arguments are supplied</td>
</tr>
<tr>
<td>8002</td>
<td>A parameter value is out of bounds, for example it was not in the allowed range as described in the command list</td>
</tr>
<tr>
<td>8003</td>
<td>Command with no valid identifier, for example sINT 200</td>
</tr>
<tr>
<td>8004</td>
<td>An invalid mode was supplied sMOD, for example sMOD 2</td>
</tr>
<tr>
<td>8005</td>
<td>The device is performing an operation and cannot accept new command</td>
</tr>
<tr>
<td>8006</td>
<td>Set commands are disabled for this interface</td>
</tr>
<tr>
<td>8007</td>
<td>This command is only available for GET</td>
</tr>
<tr>
<td>Error code</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>8100</td>
<td>Operation is not allowed in current mode</td>
</tr>
<tr>
<td>8101</td>
<td>The reference bank is not used on the device</td>
</tr>
<tr>
<td>8102</td>
<td>Operation is not allowed, for example trying sINT 20 0 when not in calibration mode</td>
</tr>
<tr>
<td>8103</td>
<td>Calibration mode is not enabled when trying to perform calibration or trying to remove non-existent calibration</td>
</tr>
<tr>
<td>8104</td>
<td>No object locator is available in current reference bank</td>
</tr>
<tr>
<td>8105</td>
<td>No blob tool with this index exists</td>
</tr>
<tr>
<td>8106</td>
<td>Polygon defect detection is not enabled</td>
</tr>
<tr>
<td>8107</td>
<td>No polygon with supplied index exists</td>
</tr>
<tr>
<td>8108</td>
<td>No pixel counter with supplied index exists</td>
</tr>
<tr>
<td>8109</td>
<td>No edge pixel counter with supplied index exists</td>
</tr>
<tr>
<td>8110</td>
<td>No pattern inspection with supplied index exists</td>
</tr>
<tr>
<td>8111</td>
<td>The move or rotation caused the ROI to appear outside of the FOV</td>
</tr>
<tr>
<td>8112</td>
<td>Trig is not activated</td>
</tr>
<tr>
<td>8113</td>
<td>The specified IP address was invalid, or an invalid combination of addresses was used</td>
</tr>
<tr>
<td>8113</td>
<td>The specified network mask was invalid</td>
</tr>
<tr>
<td>8113</td>
<td>The specified gateway address was invalid</td>
</tr>
<tr>
<td>8113</td>
<td>The combination of IP settings was invalid</td>
</tr>
<tr>
<td>8114</td>
<td>Calibration failed</td>
</tr>
<tr>
<td>8115</td>
<td>Interface not available. Only interfaces that exist in the product can be enabled/disabled</td>
</tr>
<tr>
<td>8116</td>
<td>No edge tool with supplied index exists</td>
</tr>
<tr>
<td>8117</td>
<td>No angle tool with supplied index exists</td>
</tr>
<tr>
<td>8118</td>
<td>No circle with supplied index exists</td>
</tr>
<tr>
<td>8119</td>
<td>No distance tool with supplied index exists</td>
</tr>
<tr>
<td>8120</td>
<td>The specified tool does not have a region (it is a measurement)</td>
</tr>
<tr>
<td>8121</td>
<td>The tool type of the specified tool does not support this command</td>
</tr>
<tr>
<td>8122</td>
<td>No tool with supplied index exists</td>
</tr>
<tr>
<td>8123</td>
<td>No edge count tool with supplied index exists</td>
</tr>
<tr>
<td>8124</td>
<td>Search region is not enabled</td>
</tr>
<tr>
<td>8125</td>
<td>Width or height is invalid</td>
</tr>
<tr>
<td>8126</td>
<td>No tools in configuration</td>
</tr>
<tr>
<td>8127</td>
<td>Invalid unit specified</td>
</tr>
<tr>
<td>8128</td>
<td>Alignment to specified control points failed</td>
</tr>
</tbody>
</table>
B.5 Version information

The command channel is continually extended with new commands. The intention is to always keep the command set backwards compatible with earlier versions. This table lists the available versions and the updates between each version.

<table>
<thead>
<tr>
<th>Version</th>
<th>Released in</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>PIM60 1.0</td>
<td>First official version</td>
</tr>
</tbody>
</table>

B.6 Command examples

B.6.1 Command examples Ethernet Raw

Note that the ACK messages in the below tables don't include the start and end character that is included in all ACK messages. See Section B.1, “Command syntax” (page 52) for more information on start and end character.

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
<th>ACK message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch to Run mode</td>
<td>sMOD 0</td>
<td>rsMOD 0</td>
</tr>
<tr>
<td>Switch to Edit mode</td>
<td>sMOD 1</td>
<td>rsMOD 0</td>
</tr>
<tr>
<td>Get device mode</td>
<td>gMOD</td>
<td>_rgMOD 0 1</td>
</tr>
<tr>
<td>Set trigger mode to free-running</td>
<td>sINT 16 0</td>
<td>rsINT 16 0</td>
</tr>
<tr>
<td>Set trigger mode to triggered</td>
<td>sINT 16 1</td>
<td>rsINT 16 0</td>
</tr>
<tr>
<td>Get trigger mode</td>
<td>gINT 16</td>
<td>_rgINT 16 0 0 – if free-running _rgINT 16 0 1 – if triggered by Ethernet</td>
</tr>
<tr>
<td>Set defect intensity thresholds to 200-255 for</td>
<td>sINT 69 2 200 255</td>
<td>rsINT 69 0</td>
</tr>
<tr>
<td>polygon 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Get defect intensity thresholds for polygon 2</td>
<td>gINT 69 2</td>
<td>_rgINT 69 0 200 255 – if OK _rgINT 69 8107 polygon does not exist – if not OK</td>
</tr>
<tr>
<td>Get name of second reference object</td>
<td>gSTR 2 1</td>
<td>_rgSTR 2 0 Object 2 – if OK _rgSTR 2 8000 Ref bank index is not used. - if not OK</td>
</tr>
<tr>
<td>Get name of second polygon in second reference</td>
<td>gSTR 8 1 1</td>
<td>_rgSTR 8 0 Polygon 2 – if OK _rgSTR 8 8107 No polygon with this index exists. - if not OK</td>
</tr>
</tbody>
</table>
### Table B.26  Commands Ethernet Raw - device settings and actions examples

<table>
<thead>
<tr>
<th>Description</th>
<th>Command(s)</th>
<th>ACK message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set device IP address</td>
<td>sINT 120 192 168 1 110</td>
<td></td>
</tr>
<tr>
<td>Apply IP settings</td>
<td>aACT 5 0(^b)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rsINT 120 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>raACT 5 0</td>
<td></td>
</tr>
<tr>
<td>Set device netmask</td>
<td>sINT 121 255 255 0 0</td>
<td></td>
</tr>
<tr>
<td>Apply IP settings</td>
<td>aACT 5 0(^b)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rsINT 121 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>raACT 5 0</td>
<td></td>
</tr>
<tr>
<td>Set device gateway (optional)</td>
<td>sINT 122 192 168 1 1</td>
<td></td>
</tr>
<tr>
<td>Apply IP settings</td>
<td>aACT 5 0(^b)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rsINT 122 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>raACT 5 0</td>
<td></td>
</tr>
<tr>
<td>Enter calibration mode</td>
<td>sINT 20 1</td>
<td></td>
</tr>
<tr>
<td>Perform calibration</td>
<td>aACT 3 6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rsINT 20 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>raACT 3 0 75(^c)</td>
<td></td>
</tr>
</tbody>
</table>

\(^a\)The below examples can also be done by sending the set device IP address, set device netmask and set device gateway in a sequence and then send the aACT 5 command to activate all these settings.

\(^b\)The aACT 5 command will make the device being temporarily disconnected while the new settings are applied. After this command the new IP address will have to be used in order to connect to the device.

\(^c\)Returned value is the calibration target coverage in percent.
C. Restore configuration over Web API

The restore configuration operation takes a device configuration created with the backup functionality and replaces the current configuration with the configuration in the backup file. The operation is a multiple step procedure with the following steps:

1. Create session cookie
2. Login
3. Prepare restore mode
4. Transfer restore file to device
5. Device restart

The restore operation will remove the previous configuration and replace it with a new configuration. The IP address and the chessboard calibration will not be updated by the restore operation. It is not possible to use the device for other purposes during the restore operation. The operation may take several minutes to perform and the time is partly depending on the size of the backup file.

C.1 Restore configuration

The restore configuration operation takes a device configuration created with the backup functionality and replaces the current configuration with the configuration in the backup file. The operation is a multiple step procedure with the following steps:

1. Create session cookie
2. Login
3. Prepare restore mode
4. Transfer restore file to device
5. Device restart

The restore operation will remove the previous configuration and replace it with a new configuration. The IP address and the chessboard calibration will not be updated by the restore operation. It is not possible to use the device for other purposes during the restore operation. The operation may take several minutes to perform and the time is partly depending on the size of the backup file.

C.2 Create session cookie

A session cookie is used to handle operations requiring login with user name and password. The session cookie is created before performing the login operation and the cookie must then be supplied in the login operation and for all following operations.

Operations

CREATE COOKIE

C.3 Login

A login with the user name "Maintenance" is required to change information on the device. The password is the password stored on the device. Default password is "Inspector".

Operations

CREATE SOCKET
CONNECT TO SOCKET(<IP address>, port = 80)
SEND HTTP POST REQUEST (to="/HandleConfig", data = "sopas_username=Maintenance&sopas_password=<login_password>")
C.4 Prepare restore mode

This operation will terminate normal device operation and set the device to focus on receiving the backup file contents.

After the completion of this step, the device is in transfer file mode. All other interaction with the device except the transfer file requests may interfere with the transfer file operation and should be avoided.

Operations

CREATE SOCKET
CONNECT TO SOCKET(<IP address>, port = 80)
SEND HTTP GET REQUEST (to="/SelectRestore?prepare_on")
CLOSE SOCKET

URL template (replace items in "<>")

GET /SelectRestore?prepare_on HTTP/1.1
Host: <IP address>
Connection: Keep-alive
Cookie: <Session cookie>

C.5 Transfer restore file to device

During the transfer phase the contents of the backup file is transferred to the device.

Operations

CREATE SOCKET
CONNECT TO SOCKET(<IP address>, port = 80)
SEND HTTP POST REQUEST (to="/RestoreConfig", data=<full path to backup file>)
CLOSE SOCKET

URL template (replace items in "<>")

POST /RestoreConfig HTTP/1.1
Host: <IP address>
Cookie: <Session cookie>
Content-Length: <File size>
Connection: Keep-Alive
Content-Type: multipart/form-data;
boundary=cd07053eab074616b9c4703b70584d7d
Content-Disposition: form-data; name="datafile"; filename="<full path to backup file>"
charset=utf-8
Content-Disposition: text/plain;
FormatVersion=RAW01.00 <Data and more data>

C.6 Device restart

When the transfer is completed, the parameters on the device are updated and the configuration is stored permanently on the flash file system. The device is then restarted.
Index

A
Activate and deactivate web interfaces, 21
Assemblies command channel, output, 36
Assemblies result channel, input, 35
Attributes
  Ethernet Raw, 23
  EtherNet/IP, 28
  Result output formatting, 50

B
Backup configuration, 16
Basic principles
  Ethernet Raw, 26
  EtherNet/IP, 34
  Web interface, 15

C
Command channel, 52
  Command types, 52
  Error codes, 74
  EtherNet/IP, 53
  Examples, 77
  Functions, 54
  Version information, 77
Command channel, slim, 36
Command syntax
  Ethernet Raw, 26
  EtherNet/IP, 34
  Web interface, 15
Container specific tags, 40
Control the sensor
  Ethernet Raw, 26
  EtherNet/IP, 33
  Web interface, 15
Coordinates via Ethernet
  Attributes, 23
  Validate output string, 39
  XML based formatting, 39
  XML formatting, 39
Current reference object, 16
Custom web pages, 17
  Display live image, 19

D
Digital inputs and outputs, 8

E
Error codes, Command channels, 74
Ethernet Raw, 22
  ASCII versus binary, 22
  Basic principles, 26
  Command syntax, 26
  Control the sensor via Ethernet Raw, 26
  Image trig, 27
  Port interval, 22
  Reference object, 27
  Single port solution, 27
  TCP versus UDP, 22
EtherNet/IP, 28
  Basic principles, 34
  Command syntax, 34
  Control the sensor via EtherNet/IP, 33
  Image trig, 35
  Input assemblies result channel, 35
  Output assemblies command channel, 36
  Reference object, 34

F
Formatting strings, 23, 28
  Ethernet Raw, 23
  EtherNet/IP, 28

I
I/O extension box, 8
  Configure the IP address, 8
  Input and output connections, 10
  Physical network connection, 8
  Setup the I/O extension box, 9
  Troubleshooting, 11
Image trig
  Ethernet Raw, 27
  EtherNet/IP, 35

P
Port interval
  Ethernet Raw, 22

R
Reference object
  Ethernet Raw, 27
  EtherNet/IP, 34
Restore configuration, 16, 79
Result in PLC, 29, 31-32
Result output formatting, 39
  Container specific tags, 40
  General tags, 49
Results via Ethernet Raw, 22
Results via EtherNet/IP, 28
Results via Web API, 13

S
Setup the I/O extension box in SOPAS Engineering Tool (ET), 9
  Single port solution
    Ethernet Raw, 27
Troubleshooting
  I/O extension box, 11
V
Validate output string, 39
Version information, command channels, 77

W
Web API
  Restore configuration, 79
Web interface, 13
  Basic principles, 15
  Command syntax, 15
  Control the sensor via Web API, 15
  Custom web pages, 17
  Live image, 13
  Logged image, 14
  Results, 13

X
XML based formatting, 39
XML formatting, 39
Australia
Phone +61 3 9457 0600
1800 334 802 – tollfree
E-Mail sales@sick.com.au

Austria
Phone +43 22 36 62 28 8-0
E-Mail office@sick.at

Belgium/Luxembourg
Phone +32 2 466 55 66
E-Mail info@sick.be

Brazil
Phone +55 11 3274 7430
E-Mail info@sick.com.br

Canada
Phone +1 905 771 14 44
E-Mail information@sick.com

Czech Republic
Phone +420 57 91 18 50
E-Mail sick@sick.cz

Chile
Phone +56 2 2274 7430
E-Mail info@schadler.com

China
Phone +86 20 2882 3600
E-Mail info@china@sick.net.cn

Denmark
Phone +45 45 82 64 00
E-Mail sick@sick.dk

Finland
Phone +358-9-2515 800
E-Mail sick@sick.fi

France
Phone +33 1 64 62 35 00
E-Mail info@sick.fr

Germany
Phone +49 211 5301-301
E-Mail info@sick.de

Japan
Phone +81 3 3509 2112
E-Mail support@sick.jp

Malaysia
Phone +6 03 8080 7425
E-Mail enquiry.my@sick.com

Mexico
Phone +52 (472) 748 9451
E-Mail mario.garcia@sick.com

Netherlands
Phone +31 30 2044 000
E-Mail info@sick.nl

New Zealand
Phone +64 9 415 0459
0800 222 278 – tollfree
E-Mail sales@sick.co.nz

Norway
Phone +47 67 81 50 00
E-Mail sick@sick.no

Poland
Phone +48 22 539 41 00
E-Mail info@sick.pl

Romania
Phone +40 356 171 120
E-Mail office@sick.ro

Russia
Phone +7 495 775 05 30
E-Mail info@sick.ru

Singapore
Phone +65 6744 3732
E-Mail sales.sg@sick.com

Slovakia
Phone +421 482 901201
E-Mail mail@sick-sk.sk

Singapore
Phone +65 6744 3732
E-Mail sales.sg@sick.com

South Korea
Phone +82 2 786 6321
E-Mail info@sickkorea.net

Spain
Phone +34 93 480 31 00
E-Mail info@sick.es

Sweden
Phone +46 10 110 10 00
E-Mail info@sick.se

Switzerland
Phone +41 41 619 29 39
E-Mail contact@sick.ch

Taiwan
Phone +886 22375-6288
E-Mail sales@sick.com.tw

Thailand
Phone +66 2645 0009
E-Mail Ronnie.Lim@sick.com

Turkey
Phone +90 216 528 50 00
E-Mail info@sick.com.tr

United Arab Emirates
Phone +971 4 88 65 878
E-Mail info@sick.ae

United Kingdom
Phone +44 1727 831121
E-Mail info@sick.co.uk

USA
Phone +1 800 325 7425
E-Mail info@sick.com

Vietnam
Phone +84 945452999
E-Mail Ngo.Duy.Linh@sick.com

Further locations at www.sick.com