REFERENCE MANUAL

Inspector PIM60 ver 2.0

Vision sensor

SICK
Sensor Intelligence.

WARNING

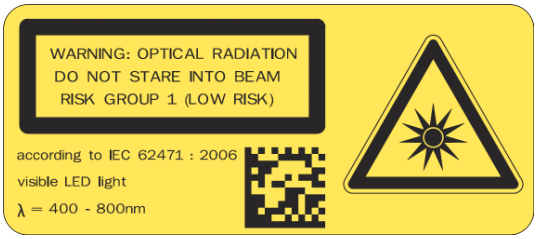
VSPM-6F2113 (Inspector PIM60), VSPM-6B2113 (Inspector PIM60 Base), VSPM-6F2113S19 (Inspector PIM60 Bead)

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 x 10^4 W/(m^2 sr) within 100 s

L_H <10^6 W/(m^2 sr) within 10 s



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

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

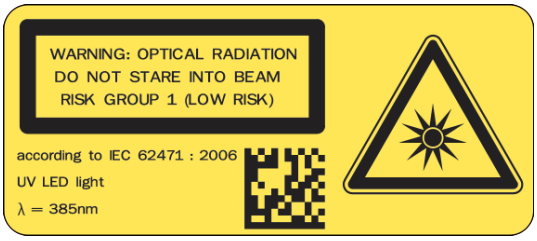
The Inspector is equipped with a LED illumination that must be considered as a lamp system of Risk Group 1 (low risk) according to IEC 62471:2006

Accessible irradiance at distances > 200 mm:

E_S < 3 x 10^{-3} W/m^2 within 10^4 s

E_{UVA} < 33 W/m^2 within 300 s

L_R < 7 x 10^6 W/(m^2 sr) within 10 s



WARNING: OPTICAL RADIATION DO NOT STARE INTO BEAM
RISK GROUP 1 (LOW RISK) according to IEC 62471:2006
UV LED light λ = 385 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 W/m^2 within 10^3 s

L_{IR} < 1.2 x 10^6 W/(m^2 sr) within 10^3 s

NOTICE: IR EMITTED FROM THIS PRODUCT
RISK GROUP 0 (EXEMPT RISK) according to IEC 62471:2006
IR LED light λ = 850 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.

8015726/ZPP7/2018-09

Table of Contents

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

2 I/O extension box

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.

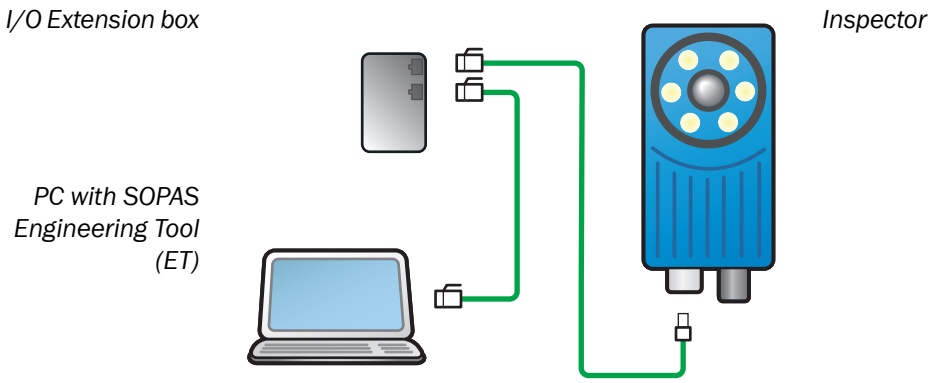


Figure 2.1 Physical network connection

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.

Inspector PIM60	I/O Extension Box	PC
192.168.1.110	192.168.1.3	192.168.1.30

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.

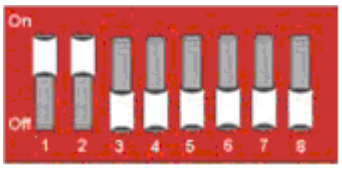


Figure 2.2 Example

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.

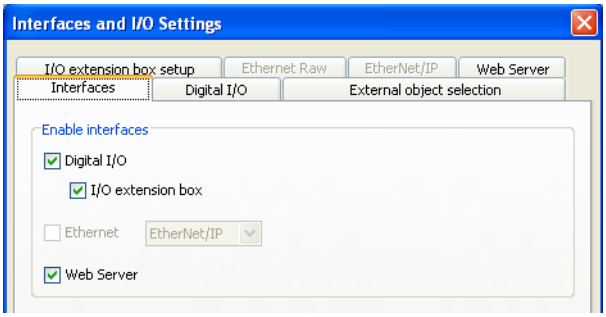


Figure 2.3 I/O Extension Box setup

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.

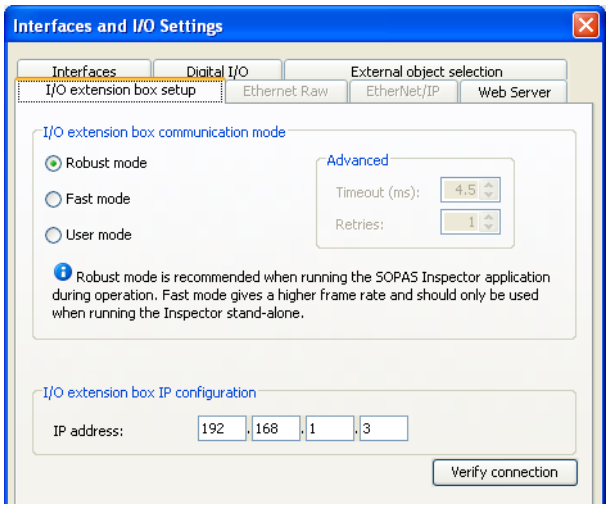


Figure 2.4 Set up mode and IP address

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:

```
http://192.168.1.110/LiveImage.jpg?ShowOverlay
```

Request a live image with simplified overlay graphics:

```
http://192.168.1.110/LiveImage.jpg?SimplifiedOverlay
```

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:

`http://192.168.1.110/LiveImage.jpg?ShowOverlay&id=ID_ABC123`

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

`http://192.168.1.110/LiveImage.jpg?SimplifiedOverlay&id=ID_ABC123`

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`

`http://192.168.1.110/LockLog?Unlock`

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 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 **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 **InspectorPIM60** 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 .spb 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 buses, **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>
  <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>

    ...
```

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:

```
// 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:

```
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

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

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:

```
...
<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.

```
...
<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);
  };
```

```
      // Initiate the image and refresh when page is loaded
      window.onload = function(){
        refreshLiveImage();
      };
      //-->
    </script>

    ...
  </head>
  <body>
    <img id="liveImage" src="/LiveImage.jpg?ShowOverlay"/>
  </body>
```

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.

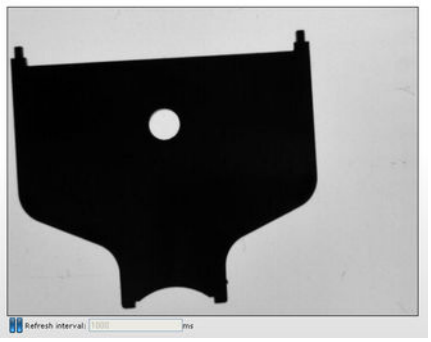


Figure 3.1 Inspector's default live image component

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 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 **InspectorPIM60** 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:

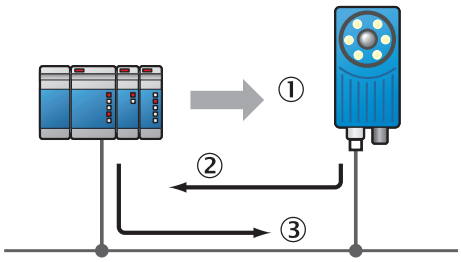
- Ethernet Result Output = start port (default 2114)
- Command channel = start port + 1
- Dedicated image trig = start port + 2

4.2 Get results via Ethernet Raw

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

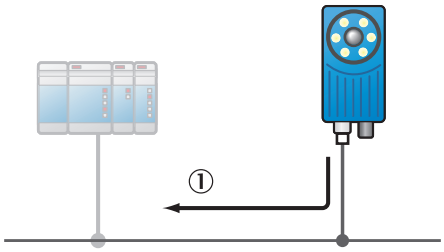
4.2.1 TCP versus UDP

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.

	Most significant byte	Least significant byte
Value to be sent from device:	10000100	01110000

Transfer order	First transferred byte	Second transferred byte
Little endian	01110000	10000100
Big endian	10000100	01110000

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/><NEWLINE/> ①
Image_number:<SPACE/><IMAGE_NUMBER/><NEWLINE/>  ②
Object_locator.<NEWLINE/> ③
<OBJECT_LOC> ④
Located:<SPACE/><DECISION/><NEWLINE/> ⑤
Score:<SPACE/><SCORE/><NEWLINE/> ⑥
Scale:<SPACE/><SCALE/><NEWLINE/> ⑦
Position_(X,Y):<SPACE/>(<X/>,<Y/>)<NEWLINE/> ⑧
Rotation:<SPACE/><ROTATION/><NEWLINE/>  ⑨
</OBJECT_LOC>  ⑩
```

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.<NEWLINE/>  ①
<BLOB index="0" name="Blob 1"> ②
Found_blobs:<SPACE/><FOUND_BLOBS/><NEWLINE/>  ③
-------------------------<NEWLINE/>  ④
Blob_information:<NEWLINE/>  ⑤
Position_(X,Y):<SPACE/>(<X/>,<SPACE/><Y/>)<NEWLINE/>  ⑥
```

```
Area:<SPACE/><AREA/><NEWLINE/>  ⑦
Angle:<SPACE/><ANGLE/><NEWLINE/>  ⑧
Structure:<SPACE/><EDGE_PIXELS/><NEWLINE/>  ⑨
Touches_ROI_border:<SPACE/><EDGE_FLAG/><NEWLINE/>  ⑩
</BLOB>  ⑪
```

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"><NEWLINE/>  ①
Corners:<SPACE/><NUM_CORNERS/><NEWLINE/>  ②
<CORNERS corners="all">(X,Y):<SPACE/>(<X/>,<Y/>)<NEWLINE/>  ③
</CORNERS>  ④
</POLYGON>  ⑤
```

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

⑤ End of container for polygon

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[1] format. This is typically used in conjugation with web based HMIs for easier result extraction.

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

{"MESSAGE":{"IMAGE_NUMBER":"2975780", "Pixel_counter_1":{"DECISION":"1", "PIXELS
```

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:

[1] JSON (JavaScript Object Notation) in accordance with RFC 4627

`<command> <identifier> <arg1> <arg2> ... <argN>`

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

```
<IMAGE_NUMBER dataType="DINT" pos="0"/> ①
<OBJECT_LOC> ②
<DECISION dataType="SINT" pos="0"/> ③
<SCORE dataType="REAL" pos="0"/> ④
<SCALE dataType="REAL" pos="1"/> ⑤
<X dataType="REAL" pos="2"/> ⑥
<Y dataType="REAL" pos="3"/> ⑦
<ROTATION dataType="REAL" pos="4"/> ⑧
</OBJECT_LOC> ⑨
```

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)

④ Score expressed in percent how well the taught object is matched against the live image
⑤ Scale is the factor of analyzed live image compared to taught reference object
⑥ Position (x) of the reference point of the object locator
⑦ Position (y) of the reference point of the object locator
⑧ Rotation of the object locator, in degrees or radians depending on the configured value in the **Ethernet Result Output** dialog
⑨ 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.

Position ref (pos) from XML configuration	Data type (dataType)	Offset byte	Variable from example above
0	SINT	0	DECISION
1	SINT	1	
2	SINT	2	
3	SINT	3	
4	SINT	4	
5	SINT	5	
6	SINT	6	
7	SINT	7	
0	INT	8	
1	INT	10	
2	INT	12	
3	INT	14	
4	INT	16	
5	INT	18	
6	INT	20	
7	INT	22	
0	DINT	24	IMAGE_NUMBER
1	DINT	28	

Position ref (pos) from XML configuration	Data type (dataType)	Offset byte	Variable from example above
2	DINT	32	
3	DINT	36	
4	DINT	40	
0	REAL	44	SCORE
1	REAL	48	SCALE
2	REAL	52	X
3	REAL	56	Y
4	REAL	60	ROTATION

Example string for configuration with only a Blob

```
<IMAGE_NUMBER dataType="DINT" pos="0"/> ①
<BLOB index="0" name="Blob 1"> ②
<FOUND_BLOBS dataType="SINT" pos="0"/> ③
<X dataType="REAL" pos="0"/> ④
<Y dataType="REAL" pos="1"/> ⑤
<AREA dataType="DINT" pos="1"/> ⑥
<ANGLE dataType="REAL" pos="2"/> ⑦
<EDGE_PIXELS dataType="DINT" pos="2"/> ⑧
<EDGE_FLAG dataType="SINT" pos="1"/> ⑨
</BLOB> ⑩
```

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.

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

Part of default string for configuration with only a Polygon

```
<IMAGE_NUMBER dataType="DINT" pos="0"/> ①
<POLYGON name="Polygon 1"> ②
<NUM_CORNERS dataType="SINT" pos="0"/> ③
<CORNERS corners="0"> ④
```

```
<X dataType="INT" pos="1"/> ⑤
<Y dataType="INT" pos="2"/> ⑥
</CORNERS> ⑦
</POLYGON> ⑧
```

① Analyzed image's number, attributes dataType and pos
② Start of container for Polygon, Name refers to the Polygon tool's name in the **Tools** tab
③ Number of corners used for this Polygon tool
④ 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
⑤ Polygon corner coordinate (x), "pixels" or "mm"
⑥ Polygon corner coordinate (y), "pixels" or "mm"
⑦ End of tag for corners
⑧ 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 **InspectorPIM60** 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.

Position ref (pos) from XML configuration	Data type (dataType)	Offset byte	Variable from example above
0	SINT	0	NUM_CORNERS
1	SINT	1	
2	SINT	2	
3	SINT	3	
4	SINT	4	
5	SINT	5	

Position ref (pos) from XML configuration	Data type (dataType)	Offset byte	Variable from example above
6	SINT	6	
7	SINT	7	
0	INT	8	X
1	INT	10	Y
2	INT	12	
3	INT	14	
4	INT	16	
5	INT	18	
6	INT	20	
7	INT	22	
0	DINT	24	IMAGE_NUMBER
1	DINT	28	
2	DINT	32	
3	DINT	36	
4	DINT	40	
0	REAL	44	
1	REAL	48	
2	REAL	52	
3	REAL	56	
4	REAL	60	

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.
-

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

- 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

Datatype	Number/ size	Offset (bytes)	Total size
SINT	8/ 1 byte each	0	8 bytes
INT	8/ 2 bytes each	8	16 bytes
DINT	5/ 4 bytes each	24	20 bytes
REAL	5/ 4 bytes each	44	20 bytes

Assembly 2 - medium result channel

Instance ID: 105

Size: 124 bytes

Table 5.2 Input Assembly 2

Datatype	Number/ size	Offset (bytes)	Total size
SINT	12/ 1 byte each	0	12 bytes
INT	12/ 2 bytes each	12	24 bytes
DINT	11/ 4 bytes each	36	44 bytes
REAL	11/ 4 bytes each	80	44 bytes

Assembly 3 - large result channel

Instance ID: 107

Size: 248 bytes

Table 5.3 Input Assembly 3

Datatype	Number/ size	Offset (bytes)	Total size
SINT	24/ 1 byte each	0	24 bytes
INT	24/ 2 bytes each	24	48 bytes
DINT	22/ 4 bytes each	72	88 bytes
REAL	22/ 4 bytes each	160	88 bytes

Assembly 4 - extra large result channel

Instance ID: 109

Size: 484 bytes

Table 5.4 Input Assembly 4

Datatype	Number/ size	Offset (bytes)	Total size
SINT	44/ 1 byte each	0	44 bytes
INT	44/ 2 bytes each	44	88 bytes
DINT	44/ 4 bytes each	132	176 bytes
REAL	44/ 4 bytes each	308	176 bytes

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.

Instance ID: 100
Size: 4 bytes

Table 5.5 Slim Command channel

Data	Type	Offset (bytes)	Values
Select reference object	SINT	0	0-31: Selected reference object
Trigger	SINT	1	1: Trigger inspection. Set to 0 before triggering next inspection.
Reserved	SINT	2	
Reserved	SINT	3	

Command channel

Instance ID: 102
Size: 32 bytes

Table 5.6 102 output, Command channel

Data	Type	Offset (bytes)
Command	DINT	0
Identifier	DINT	4
Argument 1	DINT	8
Argument 2	DINT	12
Argument 3	DINT	16
Argument 4	DINT	20
Argument 5	DINT	24
Argument 6	DINT	28

Command channel result

Instance ID: 101
Size: 36 bytes

Table 5.7 101 input Command channel result

Data	Type	Offset (bytes)
Command	DINT	0
Identifier	DINT	4
Error code	DINT	8
Returnvalue 1	DINT	12
Returnvalue 2	DINT	16
Returnvalue 3	DINT	20
Returnvalue 4	DINT	24
Returnvalue 5	DINT	28
Returnvalue 6	DINT	32

Appendix

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:

Container tag	Explanation
OBJECT_LOC	Used to present values concerning the **Object locator**
BLOB	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.
PIXEL_COUNTER, PATTERN, EDGE_PIXEL_COUNTER	Used to present values concerning inspections.
POLYGON	Used to present values concerning a defined **Polygon**

CORNERS	Container tag within the <POLYGON> tag for presenting values concerning the polygon corners. See example:
	`<POLYGON>`
	`  <CORNERS>`
	`  <X/>,<Y/>`
	`  </CORNERS>`
	`</POLYGON>`

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.

Note

The Binary column in the tables below describes how the data should be interpreted when received from the device.

Table A.1 Container output string tags

Container tag	Value tag	Attribute	Range	Binary	Comment
OBJECT_ LOC	X	coordUnit		REAL	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.
	Y	coordUnit		REAL	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.
	ROTATION	unit	[-180, 180]	REAL	In degrees or radians depending on the configured value in the **Ethernet Result output** dialog.
	SCALE		[0.8, 1.2]	REAL	Scale factor of analyzed live image compared to taught reference object.
	SCORE		[0, 100]	REAL	Object locator matching score.
	DECISION		{0, 1}	USINT	0=not found, 1=found
EDGE_ PIXEL_		name	any string		Name attribute required if more than one Edge Pixel Counter exist

Container tag	Value tag	Attribute	Range	Binary	Comment
COUNTER	PIXELS			UDINT	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.
	DECISION		{0, 1, 2}	USINT	0=fail, 1=pass, 2=Outside image
PIXEL_ COUNTER		name	any string		Name attribute required if more than one Pixel Counter exist
	PIXELS			UDINT	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.
	DECISION		{0, 1, 2}	USINT	0=not found, 1=found, 2=Outside image
PATTERN		name	any string		Name attribute required if more than one Pattern inspection exists
	X	coordUnit		REAL	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.
	Y	coordUnit		REAL	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.
	SCORE		[0, 100]	REAL	Pattern matching score.

Container tag	Value tag	Attribute	Range	Binary	Comment
	DECISION		{0, 1, 2}	USINT	0=not found, 1=found, 2=Outside image
BEAD		name	any string		Name attribute required if more than one Bead exists.
	MIN_WIDTH	coordUnit		REAL	Minimum width of bead.
	MAX_WIDTH	coordUnit		REAL	Maximum width of bead.
	MEAN_WIDTH	coordUnit		REAL	Mean width of bead.
	NUM_TOO_NARROW		[0, 256]	UINT	Number of consecutive probes where the width was too narrow.
	NUM_TOO_WIDE		[0, 256]	UINT	Number of consecutive probes where the width was too wide.
	TOTAL_NUM_TOO_NARROW		[0, 256]	UINT	Total number of probes where the width was too narrow.
	TOTAL_NUM_TOO_WIDE		[0, 256]	UINT	Total number of probes where the width was too wide.
	DECISION		[{1,2,4,6,8,...,54,56,58,60,62}]	USINT	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.
POLYGON		name	any string		Name attribute required if more than one Polygon exists.
	NUM_CORNERS		[2, 16]	USINT	Number of corners used for this polygon tool.
	DECISION		{0, 1, 2}	USINT	0=not found, 1=defect, 2 = pass
	SCORE		[0, 100]	REAL	Polygon matching score.
	NUM_PIXELS			UDINT	Number of defect pixels inside crack detection region in polygon. Undefined for single edge tool.
	CORNER_OUTSIDE		{0,1}	USINT	0=polygon completely inside image, 1=one or more polygon corner(s) are outside image. Cannot be used for single edge tool.
	DEFECT_X	coordUnit		REAL	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.

Container tag	Value tag	Attribute	Range	Binary	Comment
					Return -1 if defect detection is not active or no defect found. Undefined for single edge tool.
	DEFECT_Y	coordUnit		REAL	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. Return -1 if defect detection is not actived or no defect found. Undefined for single edge tool.
CORNERS[a]		corners	{0, 1, ...,15, all}		"All" [b]iterates 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.
	X	coordUnit	REAL	REAL[c]	Polygon corner coordinate. In “pixels” or “mm” 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.
	Y	coordUnit	REAL	REAL[c]	Polygon corner coordinate. “pixels” or “mm” 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.

Container tag	Value tag	Attribute	Range	Binary	Comment
BLOB		name	any string		Name attribute required if more than one Blob tool exists. The name refers to the Blob tool's name in the tool tab
		index	[0, 15]		Index of found blob according to current blob sorting order. Index 0 is the first blob
	X	coordUnit		REAL	Blob center of gravity (x position). “pixels” or “mm” depending on attribute “coordUnit”
	Y	coordUnit		REAL	Blob center of gravity (y position). “pixels” or “mm” depending on attribute “coordUnit”
	FOUND_BLOBS[d]			USINT	Number of found blobs.
	LIVE_THRESHOLD_LOW[d]			USINT	The lower threshold of the Blob tool's intensity after applying ambient light compensation.
	LIVE_THRESHOLD_HIGH[d]			USINT	The upper threshold of the Blob tool's intensity after applying ambient light compensation.
	ANGLE	unit	[0, 180]	REAL	In degrees or radians depending on attribute “unit” or configured value in the **Ethernet Result output** dialog.
	AREA			UDINT	Blob area (in pixels). In “pixels” or “mm^2” depending on attribute “coordUnit” or configured value in the **Ethernet Result output** dialog.
	EDGE_FLAG		{0, 1}	USINT	0 = blob fully within ROI, 1 = blob touches ROI border
	EDGE_PIXELS			UDINT	Structure calculation value (number of edge pixels inside the found blob)
	DECISION		{0, 1}	USINT	0=not found, 1=found
EDGE_LOCATOR		name	any string		Name attribute required if more than one Edge tool exist
	X	coordUnit		REAL	X position of the reference point. In “pixels” or “mm” depending on attribute “coordUnit” or configured value in the **Ethernet Result output** dialog.
	Y	coordUnit		REAL	Y position of the reference point. In “pixels” or “mm” depending on attribute “coordUnit” or configured value in the **Ethernet Result output** dialog.

Container tag	Value tag	Attribute	Range	Binary	Comment
	SCORE		[0, 100]	REAL	Edge matching score.
	ANGLE	unit	[0, 180]	REAL	In degrees or radians depending on attribute “unit” or configured value in the **Ethernet Result output** dialog.
	DECISION		{0, 1, 2}	USINT	0=not found, 1=found, 2=Outside image
CIRCLE_LOCATOR		name	any string		Name attribute required if more than one Circle exist
	X	coordUnit		REAL	X position of the reference point. In “pixels” or “mm” depending on attribute “coordUnit” or configured value in the **Ethernet Result output** dialog.
	Y	coordUnit		REAL	Y position of the reference point. In “pixels” or “mm” depending on attribute “coordUnit” or configured value in the **Ethernet Result output** dialog.
	SCORE		[0, 100]	REAL	Circle matching score.
	DIAMETER	coordUnit		REAL	Circle's diameter in pixels or mm depending on attribute "coordUnit" or configured value in the **Ethernet Result output** dialog.
	DECISION		{0, 1, 2, 11, 12}	USINT	0=not found, 1=pass, 2=Outside image, 11=Failed on score value, 12=Failed on diameter
MEASURE_DISTANCE		name	any string		Name attribute required if more than one Measure distance exists
	DISTANCE	coordUnit		REAL	Measured distance in pixels or mm depending on attribute “coordUnit” or configured value in the **Ethernet Result output** dialog.
	VALID		{0, 1}	USINT	Indicates whether the value in the distance tag is a valid measurement or not. 0=Invalid 1=Valid
	DECISION		{0, 1}	USINT	0=fail (measure distance could not be done), 1=pass
MEASURE_ANGLE		name	any string		Name attribute required if more than one Measure Angle exist
	X	coordUnit		REAL	X position of the intersection point. In “pixels” or “mm” depending on attribute “coordUnit” or configured value in the **Ethernet Result output** dialog.

Container tag	Value tag	Attribute	Range	Binary	Comment
	Y	coordUnit		REAL	Y position of the intersection point. In “pixels” or “mm” depending on attribute “coordUnit” or configured value in the **Ethernet Result output** dialog.
	ANGLE		[0, 180]	REAL	In degrees or radians depending on attribute “unit” or configured value in the **Ethernet Result output** dialog.
	VALID		{0, 1, 2}	USINT	Indicates whether the values in the angle tag are valid measurements or not. 0=Invalid 1=Valid 2=Intersection outside image
	DECISION		{0, 1}	USINT	0=fail, 1=pass
EDGE_COUNT		name	any string		Name attribute required if more than one Edge counter exist
		index	[0, 63]		Chosen feature/edge index.
	FOUND_FEATURES[e]		[0, 64]	USINT	Number of found features.
	MIN_PITCH[e]	coordUnit/unit		REAL	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. 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.
	MAX_PITCH[e]	coordUnit/unit		REAL	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. 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.
	MEAN_PITCH[e]	coordUnit/unit		REAL	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.

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.
	DECISION[e]		{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[f]. 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[f]. 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. [g] 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.[h] 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 con-

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

[a]This tag must be used inside the <POLYGON> container

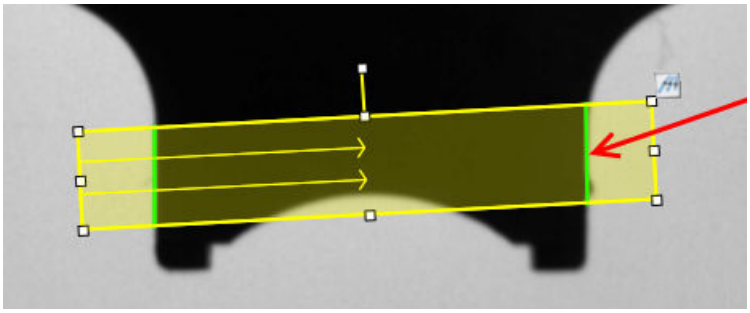
[b]Only available for Ethernet Raw.

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

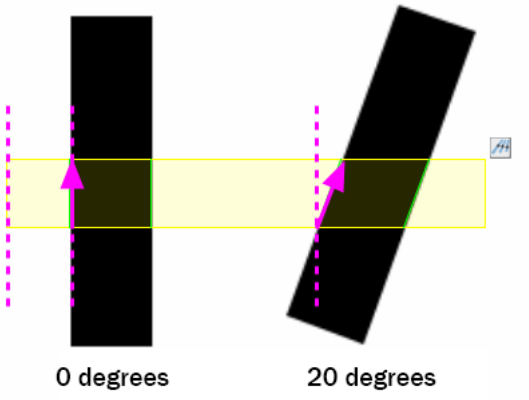
[d]This tag must be used inside the <BLOB> container. The value are given for each Blob ROI (not for each found blob)

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

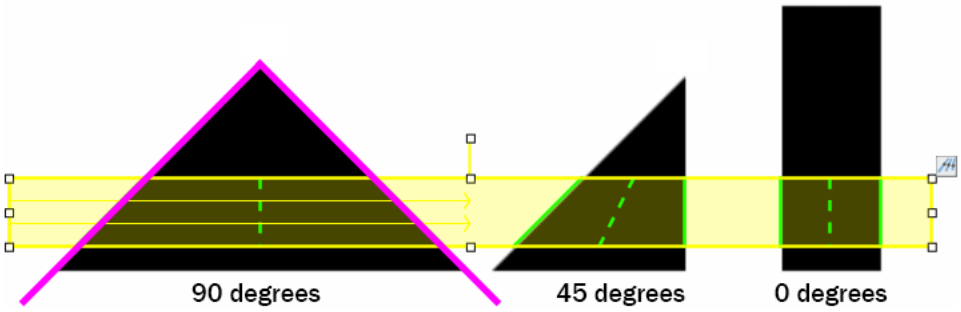
[f]Red arrow points out the midpoint for the Edge count tool



[g]The angle of the edge relative to the region.



[h]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

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

Value tag	Attribute	Range	Binary	Comment
UINT3		[0, 65535]	UINT	Value can be changed through the command channel.

[a]If a tool is fixed in field of view and not relative to the object locator, the Image_decision will report the value 3 in cases when the object locator does not locate the object and the result of the unrelated tool is failed.

[b]A reference object's index is presented as a white number on a black background on each reference object in the **Reference objects** list in the GUI. The first reference object created is assigned index number 0, and each successive reference object's index will be incremented by 1. If a reference object is deleted from the reference object list, the index number assigned to that object will be reused for the next reference object created.

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

Attribute	Range	Default value	Affects	Used in Binary format	Comment
index	[0, 15]	0	Blob	Yes	Index of blob according to current blob sorting order. Index 0 is the first blob.
scale	Any REAL	1.0	All values	Yes	Scales the values before they are printed. Can for example be used to express positions as integers in 1/10 pixel units
base	{decimal, octal, hex}	decimal	Integers	No	
timeUnit	{s, ms}	s	<TIME>	Yes	
name	any string	*none*	Identification of tools	Yes	
value	[0, 255]	0	<ASCII>	No	
intValue	[0, 255], [0, 65535], [0, 2^{32}-1]	0	<USINT>, <UINT>, <UDINT>	Yes	Integer value to be sent.
digits	[0, 9]	0	Integers and REAL	No	Minimum number of characters.
decimals	[0, 9]	2	REAL	No	Number of decimals.
corners	[0, 15]		<CORNERS>	Yes	"All" [a]iterates 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.

Attribute	Range	Default value	Affects	Used in Binary format	Comment
coordUnit	{pixels, mm, aligned}	pixels	Object locator, Blob, Pattern, Single edge, Circle, Edge counters, and Polygon coordinates	Yes	Gives result coordinates in pixel, millimeter or robot aligned millimeter format.[b]
dataType	{SINT, INT, DINT, REAL}		All values	Yes	Casts to the specified datatype. When using EtherNet/IP the attribute DataType specifies the dataType section in the selected assembly.
pos	[0, 43]		All values	No	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.
unit	{radians, degrees}	degrees	Angles	Yes	

[a]Only available for Ethernet Raw.

[b]The device must be calibrated for it to be possible to use the “mm” or "aligned" attribute.

Table A.4 Sizes of datatypes

Datatype	Size	Range	Encoding
USINT	1 byte	[0, 255]	[a]
SINT	1 byte	[-128, 127]	[a]
UINT	2 bytes	[0, 65535]	[a]
INT	2 bytes	[-32768, 32767]	[a]
UDINT	4 bytes	[0, 2^{32}-1]	[a]
DINT	4 bytes	[-2^{31}, 2^{31}-1]	[a]
REAL	4 bytes	Represented as IEEE 754 binary 32	[a]

[a]See Section 4.2.3, “*Attributes*” (page 23).

B 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

Command format	Explanation
gVER	Get protocol version that is supported by the addressed device
sMOD [mode]	Set device mode (0 = Run, 1 = Edit)
gMOD	Get the current device mode from the device
sINT [identifier] [arg1] [arg2] ... [argN]	Set “integer” parameter in the device
gINT [identifier] [arg1] [arg2] ... [argN]	Get “integer” parameter from the device
gSTR [identifier] [arg1] [arg2]	Get "string" parameter from device
aACT [identifier] [arg1] [arg2] ... [argN]	Action commands
TRIG	Trig an image acquisition and analysis
gRES	Retrieve the latest available Ethernet Result Output string
gSTAT	Retrieve the latest statistics from the device

Table B.2 Command response

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

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

Description	Command	ID
Set mode	sMOD	0
Get mode	gMOD	1
Set integer	sINT	2
Get integer	gINT	3
Get string	gSTR	6
Get version	gVER	7
Action command	aACT	8
Trig device	TRIG	9

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

Description	Command	Identifier	Usable in Run mode	Arguments	Return values	Range
Save settings in flash	aACT	1	No			-
Re-teach reference object	aACT	2	Yes, only in Run mode.	int autoExp		0=use exposure settings, 1=auto adjust
Perform calibration[a]	aACT	3	No	int boxSize[b]	- int calibrationCoverage[c]	>0 [0,100]
Remove calibration	aACT	4	No			
Apply IP settings[d]	aACT	5	Yes	int useDHCP		0=use manual settings, 1=use DHCP
Restart the Inspector[e]	aACT	6	Both in run and edit mode			
Perform (calculate) coordinate alignment[f]	aACT	7	No			

[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

Description	Command	Identifier	Usable in Run mode	Arguments	Return values	Range
Set interface permission	sINT	112	Yes	int interface int permission		Interface: 0=Ethernet Raw, 1=HTTP, 2=EtherNet/IP Permission: 1=enable, 0=disable
Get interface permission	gINT	112	Yes	int interface	int permission	Interface: 0=Ethernet Raw, 1=HTTP, 2=EtherNet/IP
Set device IP address[a]	sINT	120	Yes	int a, int b, int c, int d		Address format: a.b.c.d[b]
Get device IP address	gINT	120	Yes		int a, int b, int c, int d	Address format: a.b.c.d
Set device netmask[a]	sINT	121	Yes	int a, int b, int c, int d		Address format: a.b.c.d[b]
Get device netmask	gINT	121	Yes		int a, int b, int c, int d	Address format: a.b.c.d
Set gateway[a]	sINT	122	Yes	int a, int b, int c, int d		Address format: a.b.c.d[b]
Get gateway	gINT	122	Yes		int a, int b, int c, int d	Address format: a.b.c.d

[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

Description	Command	Identifier	Usable in Run mode	Arguments	Return values	Range
Get used protocol version	gVER	-	Yes		int version	6=PIM60 1.0
Set device mode	sMOD	-	Yes	int mode		0=Run, 1= Edit
Get device mode	gMOD	-	Yes		int mode	0=Run, 1= Edit
Trig device	TRIG	-	Yes[a]			-
Select reference object	sINT	1	Yes	int object		[0, 31]
Get active reference object	gINT	1	Yes		int object	[0,31]
Get number of configured reference objects	gINT	2	Yes	-	int object	[0,31]
Change internal illumination mode	sINT	13	No	int illumMode		0=Off, 1=On
Get internal illumination mode	gINT	13	Yes		int illumMode	0=Off, 1=On

Description	Command	Identifier	Usable in Run mode	Arguments	Return values	Range
Change exposure[b]	sINT	14	No	int exp*100		[10,10000]
Get exposure	gINT	14	Yes		int exp*100	[10, 10000]
Change gain	sINT	15	No	int gain		[0, 400]
Get gain	gINT	15	Yes		int gain	[0, 400]
Switch trigger mode(free-running, trig)	sINT	16	No	int mode		0=free-running, 1=trig
Get trigger mode	gINT	16	Yes		int trigMode	0=free-running, 1=trig
Change value of integer tags in result output (UINT1-3)	sINT	18	Yes	int index, int value		[0, 2], [0,65535]
Get value of integer tags in result output (UINT1-3)	gINT	18	Yes		int index, int value	[0, 2], [0,65535]
Get frame period time [microseconds]	gINT	19	Yes		int framePeriod	
Enter/leave calibration mode	sINT	20	No	int mode		0=normal mode, 1=calibration mode
Get calibration parameters	gINT	20	Yes	int parameter	int parameterResult	0=calibration/normal mode[c], 1=calibrated[d], 2=scaling[e], 3=origin[f], 4=rotation[g], 5=mean pixel error, 6=max pixel error
Set external trig delay	sINT	21	No	int type, int delay (milliseconds*10 or ticks)[h]		[0=ms, 1=tick], [1,50000] resp [0 ticks, 200000 ticks]
Get external trig delay	gINT	21	Yes		int type, int delay (milliseconds*10 or ticks)	[0=ms, 1=tick], [1,50000] resp [0 ticks, 200000 ticks]
Set digital output delay	sINT	22	No	int outputIndex, int type, int delay (milliseconds*10 or ticks)		[0,19], [0=ms, 1=tick], [(Min delay time)*10,50000] resp [0 ticks, 200000 ticks]
Get digital output delay	gINT	22	Yes	int outputIndex	int type, int delay (milliseconds*10 or ticks)	[0,19], [0=ms, 1=tick], [1,50000] resp [0

Description	Command	Identifier	Usable in Run mode	Arguments	Return values	Range
						ticks, 200000 ticks]
Set digital output active time	sINT	23	No	int outputIndex, int type, int time (milliseconds*10 or ticks)		[0,19], [0=ms, 1=tick], [1,10000] resp [0 ticks, 200000 ticks]
Get digital output active time	gINT	23	Yes	int outputIndex	int type, int time (milliseconds*10 or ticks)	[0,19], [0=ms, 1=tick], [1,10000] resp [0 ticks, 200000 ticks]
Set alignment control point[i]	sINT	300	No	int pointIndex, int x * 100, int y * 100 int z * 100		[0=A, 1=B, 2=C, 3=D] int x * 100, [-2147483648, 2147483647] int y * 100, [-2147483648, 2147483647] int z * 100, [-2147483648, 2147483647]

[a]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)

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

[c]The result is 0= normal mode or 1= calibration mode.

[d]The result is 0= not calibrated or 1= calibrated.

[e]The result is expressed in mm/pixel x 10000.

[f]The result is expressed in pixels for x and y.

[g]The result is expressed in degrees.

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

[i]In order for the settings to take effect the aACT 7 command needs to be sent to the device

Table B.7 Command channel functions - Object locator

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

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

Table B.8 Command channel functions - Blob

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Change blob intensity thresholds	sINT	48	No	int index, int min, int max		[0, 7], [0, 255], min<=max [0, 255], min<=max

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Get blob intensity thresholds	gINT	48	Yes	int index	- int min, int max	[0, 7], [0, 255], [0, 255]
Change blob area thresholds	sINT	49	No	int index, int min, int max		[0, 7], [10, 307200] pixels, min<=max, [10, 307200] pixels, min<=max
Get blob area thresholds	gINT	49	Yes	int index	- int min, int max	[0, 7], [10, 307200] pixels, [10, 307200] pixels
Change blob angle thresholds	sINT	50	No	int index, int angle, int angletolerance		[0, 7], [0, 180], [0, 90]
Get blob angle thresholds	gINT	50	Yes	int index	- int ref, int tol	[0, 7], [0, 180], [0, 90]
Change structure criteria thresholds	sINT	53	No	int index, int min, int max		[0, 7], [0, 100000], min<=max, [0, 100000], min<=max
Get structure criteria thresholds	gINT	53	Yes	int index	- int min, int max	[0, 7], [0, 100000], [0, 100000]
Change blob edge strength	sINT	54	No	int index, int strength		[0, 7], [0, 100] percent
Get blob edge strength	gINT	54	Yes	int index	- int strength	[0, 7], [0, 100] percent
Change ambient light compensation mode	sINT	55	No	int index, int mode		[0, 7], 0=off, 1=on
Get ambient light compensation mode	gINT	55	Yes	int index	- int mode	[0, 7], 0=off, 1=on
Change blob search method	sINT	56	No	int index, int method		[0, 7], 0=High quality, 1=Normal, 2=High speed
Get blob search method	gINT	56	Yes	int index	- int method	[0, 7], 0=High quality, 1=Normal, 2=High speed

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Move and rotate blob locator	sINT	58	No	int index, int x, int y, int angle		[0, 7], x, y = pixels, angle = degrees. Arguments are delta values as compared to the current position and angle. These can be negative
Get blob ROI position and rotation	gINT	58	Yes	int index	- int x, int y, int angle	[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.
Set number of blobs	sINT	59	No	int index int min int max		[0, 7] [0, 16] min<=max [0, 16] min<=max
Get number of blobs	gINT	59	Yes	int index	- int min, int max	[0, 7] [0, 16] min<=max [0, 16] min<=max
Get blob area thresholds in mm^2	gINT	60	Yes	int index	- int min*1000, int max*1000	[0, 7] [10 * pixel size - 307200 * pixel size], min<=max [10 * pixel size - 307200 * pixel size], min<=max
Get blob region width and height	gINT	61	Yes	int blob	int w, int height	[0-7], [0-640], [0-480]

[a]For information about blob indexing see Section B.2.2, “*Blob indexing*” (page 54)

Table B.9 Command channel functions - Bead

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Get bead edge strength	gINT	250	Yes	int index	int strength	[0, 7],
Set bead edge strength	sINT	250	No	int index, int strength		[0, 7], [0, 255]
Get bead intensity	giNT	251	Yes	int index	int intensity	[0, 7],
Set bead intensity	gINT	251	No	int index, int intensity		[0, 7], [0, 1]

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Get bead min search region width	gINT	252	Yes	int index,	int width	[0, 7],
Set bead min search region width	sINT	252	No	int index, int width		[0, 7], [0, 640]
Get bead max search region width	gINT	253	Yes	int index,	int width	[0, 7],
Set bead max search region width	sINT	253	No	int index, int width		[0,7] [0, 640]
Get bead position tolerance	gINT	254	Yes	int index,	int tolerance	[0, 7],
Set bead position tolerance	sINT	254	No	int index, int tolerance		[0, 7], [0, 320]
Get bead mean width thresholds	gINT	255	Yes	int index	int min, int max	[0, 7]
Set bead mean width thresholds	sINT	255	No	int index, int min, int max		[0, 7], [0, 640], [0, 640]
Get bead max no of consecutive too narrow	gINT	256	Yes	int index	int maxTooNarrow	[0, 7],
Set bead max no of consecutive too narrow	sINT	256	No	int index, int max		[0, 7], [0, 256]
Get bead max no of consecutive too wide	gINT	257	Yes	int index	int maxTooWide	[0, 7],
Set bead max no of consecutive too wide	sINT	257	No	int index, int max		[0, 7], [0, 256]
Move bead	sINT	258	No	int index, int corner, int delta-x, int delta-y		[0, 7], delta-x, delta-y = pixels. Arguments are delta values. These can be negative as compared to the origin.
Get bead corner	gINT	259	Yes	int index int corner	int x, int y	[0, 7], x, y = pixels, Arguments are delta values. These can be negative as compared to the origin.
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

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
						negative as compared to the origin.
Get bead min probe distance	gINT	260	Yes	int index	int distance	[0, 7]
Set bead min probe distance	sINT	260	No	int index	int distance	[2, 10]
Get bead max no of total too narrow	gINT	261	Yes	int index	int maxTotalTooNarrow	[0, 7]
Set bead max no of total too narrow	sINT	261	No	int index, int max		[0, 7], [0, 256]
Get bead max no of total too wide	gINT	262	Yes	int index	int maxTotalTooWide	[0, 7]
Set bead max no of total too wide	sINT	262	No	int index, int max		[0, 7], [0, 256]
Get bead tool name	gSTR	20	Yes	int referenceObject int index		

[a]For information about bead indexing see Section B.2.3, “*Bead indexing*” (page 54)

Table B.10 Command channel functions - Polygon

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

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Get polygon score threshold	gINT	66	Yes	int index	- int threshold	[0, 7], [0, 100] pixels
Change polygon margin	sINT	67	No	int index, int margin		[0, 7], [0, 20] pixels
Get polygon margin	gINT	67	Yes	int index	- int margin	[0, 7], [0, 20] pixels
Change polygon defect detection width	sINT	68	No	int index, int width		[0, 7], [0, 100] pixels
Get polygon defect detection width	gINT	68	Yes	int index	- int width	[0, 7], [0, 100] pixels
Change polygon defect intensity range thresholds	sINT	69	No	int index, int min, int max		[0, 7], [0, 255], min<=max, [0, 255], min<=max
Get polygon defect intensity range thresholds	gINT	69	Yes	int index	- int min, int max	[0, 7], [0, 255], [0, 255]
Change polygon max defects threshold	sINT	70	No	int index, int max		[0, 7], [0, 100] pixels
Get polygon max defects threshold	gINT	70	Yes	int index	- int max	[0, 7], [0, 100] pixels
Change polygon defect detection mode	sINT	71	No	int index, int mode		[0, 7], 0=off, 1=on
Get polygon defect detection mode	gINT	71	Yes	int index	- int mode	[0, 7], 0=off, 1=on
Move polygon	sINT	72	No	int index, int x, int y		[0, 7], x, y = pixels, Arguments are delta values. These can be negative as compared to the origin.
Reserved for future use	gINT	72	-	-	-	-
Move polygon corner	sINT	73	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 negative as

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
						compared to the origin.
Get polygon corner	gINT	73	Yes	int index int corner	- - int x, int y	[0-7], [0-15] x, y = pixels

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

Table B.11 Command channel functions - Pixel counter

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Change pixel counter intensity range thresholds	sINT	80	No	int index, int min, int max		[0, 63], [0, 255], min<=max, [0, 255], min<=max
Get pixel counter intensity range thresholds	gINT	80	Yes	int index	- int min, int max	[0, 63], [0, 255], [0, 255]
Change No. of pixels in range thresholds	sINT	81	No	int index, int min, int max		[0, 63], [0, ROI size[b]] pixels , min<=max, [0, ROI size[b]] pixels, min<=max
Get No. of pixels in range thresholds	gINT	81	Yes	int index	- int min, int max	[0, 63], [0, ROI size[b]] pixels, [0, ROI size[b]] pixels

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

[b]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

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Change edge pixel counter edge strength	sINT	82	No	int index, int strength		[0, 63], [0, 100]
Get edge pixel counter edge strength	gINT	82	Yes	int index	- int strength	[0, 63], [0, 100]
Change No. of edge pixels thresholds	sINT	83	No	int index, int min, int max		[0, 63], [0, ROI size[b]] pixels, min<=max [0, ROI size[b]] pixels, min<=max

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Get No. of edge pixels thresholds	gINT	83	Yes	int index	- int 10000*min, int 10000*max	[0, 63], [0, ROI size[b]] pixels, min<=max [0, ROI size[b]] pixels, min<=max

[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

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Change pattern position tolerance	sINT	84	No	int index, int tolerance		[0, 63], [0, 4] pixels
Get pattern position tolerance	gINT	84	Yes	int index	- int tol	[0, 63], [0, 4] pixels
Change pattern score threshold	sINT	85	No	int index, int threshold		[0, 63], [0, 100] percent
Get pattern score threshold	gINT	85	Yes	int index	- int threshold	[0, 63], [0, 100] percent

[a]For information about tools indexing see Section B.2.5, "*Tools indexing*" (page 54)

Table B.14 Command channel functions - Edge tool

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Change edge tool edge contrast	sINT	150	No	int index, int contrast		[0, 63], [0, 100]
Get edge tool edge contrast	gINT	150	Yes	int index	- int contrast	[0, 63], [0, 100]
Change edge tool line fit criteria	sINT	151	No	int index, int criterion		[0, 63], [0, 2] (0 = best fit, 1 = first, 2 = last)
Get edge tool line fit criteria	gINT	151	Yes	int index	- int criterion	[0, 63], [0, 2] (0 = best fit, 1 = first, 2 = last)
Change edge tool polarity	sINT	152	No	int index, int polarity		[0, 63], [0, 2] (0 = any, 1 = bright to dark, 2 = dark to bright)

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Get edge tool polarity	gINT	152	Yes	int index	- int criterion	[0, 63], [0, 2] (0 = any, 1 = bright to dark, 2 = dark to bright)
Change edge tool score threshold	sINT	153	No	int index, int threshold		[0, 63], [0, 100]
Get edge tool score threshold	gINT	153	Yes	int index	- int threshold	[0, 63], [0, 100]

[a]For information about tools indexing see Section B.2.5, "*Tools indexing*" (page 54)

Table B.15 Command channel functions - Find maximum

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Change find maximum edge contrast	sINT	220	No	int index, int contrast		[0, 63], [0, 100]
Get find maximum edge contrast	gINT	220	Yes	int index	- int contrast	[0, 63], [0, 100]
Change find maximum criteria	sINT	221	No	int index, int criterion		[0, 63], [0, 2] (0 = first, 1 = last)
Get find maximum criteria	gINT	221	Yes	int index	- int criterion	[0, 63], [0, 2] (0 = first, 1 = last)
Change find maximum polarity	sINT	222	No	int index, int polarity		[0, 63], [0, 2] (0 = any, 1 = bright to dark, 2 = dark to bright)
Get find maximum polarity	gINT	222	Yes	int index	- int criterion	[0, 63], [0, 2] (0 = any, 1 = bright to dark, 2 = dark to bright)

[a]For information about tools indexing see Section B.2.5, "*Tools indexing*" (page 54)

Table B.16 Command channel functions - Circle

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Change circle edge contrast	sINT	160	No	int index, int contrast		[0, 63], [0, 100]

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Get circle edge contrast	gINT	160	Yes	int index	- int contrast	[0, 63], [0, 100]
Change circle diameter thresholds (**without search enabled**)	sINT	161	No	int index, int min int max int unit		[0, 63], [1, max] [5, circle within FOV][b] [0, 1] (0 = pixels, 1 = millimeters)
Change circle diameter thresholds (**with search enabled**)	sINT	161	No	int index, int min int max int unit		[0, 63], [shortest search region side/3, max] [min, shortest search region side] [0, 1] (0 = pixels, 1 = millimeters)
Get circle diameter thresholds	gINT	161	Yes	int index, int unit	- - int min int max	[0, 63], [0, 1] (0 = pixels, 1 = millimeters) Min threshold in the unit specified, Max threshold in the unit specified
Change circle line fit criteria	sINT	162	No	int index, int criterion		[0, 63], [0, 2] (0 = strongest, 1 = first, 2 = last)
Get circle line fit criteria	gINT	162	Yes	int index	- int criterion	[0, 63], [0, 2] (0 = strongest, 1 = first, 2 = last)
Change circle polarity	sINT	163	No	int index, int polarity		[0, 63], [0, 2] (0 = any, 1 = bright to dark, 2 = dark to bright)
Get circle polarity	gINT	163	Yes	int index	- int criterion	[0, 63], [0, 2] (0 = any, 1 = bright to dark, 2 = dark to bright)
Change circle robustness	sINT	164	No	int index, int robustness		[0, 63], [0, 4] (0 = high robustness to 4 = high speed)
Get circle robustness	gINT	164	Yes	int index	- int criterion	[0, 63], [0, 4] (0 = high robustness to 4 = high speed)

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Change circle score threshold	sINT	165	No	int index, int threshold		[0, 63], [0, 100]
Get circle score threshold	gINT	165	Yes	int index	- int threshold	[0, 63], [0, 100]
Change circle quality	sINT	166	No	int index, int quality		[0, 63], [0, 6] (0 = tolerant to 6 = exact)
Get circle quality	gINT	166	Yes	int index	- int quality	[0, 63], [0, 6] (0 = tolerant to 6 = exact)
Change circle diameter offset compensation	sINT	167	No	int index, int offset*1000 int unit		[0, 63], [-1000000,1000000] [0, 1] (0 = pixels, 1 = millimeters)
Get circle diameter offset compensation	gINT	167	Yes	int index, int unit	- - int offset*1000	[0, 63], [0, 1] (0 = pixels, 1 = millimeters)
Change circle diameter tolerance thresholds	sINT	169	No	int index, int min*1000 int max*1000 int unit		[0, 63], [0, 640000] min<=max [0, 640000] [0, 1] (0 = pixels, 1 = millimeters)
Get circle diameter tolerance thresholds	gINT	169	Yes	int index, int unit	- - int min*1000 int max*1000	[0, 63], [0, 1] (0 = pixels, 1 = millimeters) [0, 640000] min<=max [0, 640000]

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

[b]When 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

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Move and rotate inspection	sINT	86	No	int index, int x, int y, int angle		[0, 63], x, y = pixels, angle = degrees. Arguments are delta values. These can be negative as compared to the origin.
Get inspection position and rotation	gINT	86	Yes	int index	- int x, int y, int angle	[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.
Get number of pixels in ROI, (Valid for Pixel Counter and Edge Pixel Counter)	gINT	87	Yes	int index	- int pixels	[0, 63], Number of pixels in the reference object's ROI

[a]For information about tools indexing see Section B.2.5, "*Tools indexing*" (page 54)

Table B.18 Command channel functions - Distance

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Change distance measurement thresholds	sINT	170	No	int index, int min*1000 int max*1000 int unit		[0, 63], [0, VGA image diagonal] [0, VGA image diagonal] [0, 1] (0 = pixels, 1 = millimeters) min ≤ max
Get distance measurement thresholds	gINT	170	Yes	int index, int unit	- - int min*1000 int max*1000	[0, 63], [0, 1] (0 = pixels, 1 = millimeters) [0, VGA image diagonal] [0, VGA image diagonal]
Change distance offset compensation	sINT	173	No	int index, int offset*1000 int unit		[0, 63], [-1000000,1000000] [0, 1] (0 = pixels, 1 = millimeters)

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Get distance offset compensation	gINT	173	Yes	int index, int unit	- - int offset*1000	[0, 63], [0, 1] (0 = pixels, 1 = millimeters)

[a]For information about tools indexing see Section B.2.5, "*Tools indexing*" (page 54)

Table B.19 Command channel functions - Angle

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Change angle measurement thresholds	sINT	180	No	int index, int min*1000 int max*1000		[0, 63], [0, 100] [0,180000], min ≤ max
Get angle measurement thresholds	gINT	180	Yes	int index	- int min*1000 int max*1000	[0, 63], [0, 100] [0,180000], min ≤ max
Change angle offset compensation	sINT	181	No	int index, int offset*1000		[0, 63], [-1000000,1000000]
Get angle offset compensation	gINT	181	Yes	int index	- int offset*1000	[0, 63]

[a]For information about tools indexing see Section B.2.5, "*Tools indexing*" (page 54)

Table B.20 Command channel functions - Edge counter

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Change Edge counter edge contrast	sINT	190	No	int index, int contrast		[0, 63], [0, 100]
Get Edge counter edge contrast	gINT	190	Yes	int index	- int contrast	[0, 63], [0, 100]
Change Edge counter edge quality	sINT	191	No	int index, int robustness		[0, 63], [0, 6]
Get Edge counter edge quality	gINT	191	Yes	int index	- int quality	[0, 63], [0, 6]

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Change Edge counter feature width	sINT	192	No	int index, int min*10 int max*10 int unit		[0, 63], [0, VGA image diagonal] [0, VGA image diagonal] [0, 1] (0 = pixels, 1 = millimeters)[b] min ≤ max
Get Edge counter feature width	gINT	192	Yes	int index, int unit	- - int min*10 int max*10	[0, 63], [0, 1] (0 = pixels, 1 = millimeters) [0, VGA image diagonal] [0, VGA image diagonal]
Change Edge counter feature type	sINT	193	No	int index, int type		[0, 63], (0 = bright, 1 = dark, 2 = single edge)
Get Edge counter feature type	gINT	193	Yes	int index	- int type	[0, 63], (0 = bright, 1 = dark, 2 = single edge)
Change edge count min max features	sINT	194	No	int index, int min int max		[0, 63], [0, 63] [0, 63] min ≤ max
Get edge count min max features	gINT	194	Yes	int index	- int min int max	[0, 63], [0, 64] [0, 64]
Change edge count search method robustness	sINT	195	No	int index, int robustness		[0, 63], [0, 4]
Get edge count search method robustness	gINT	195	Yes	int index	- int robustness	[0, 63], [0, 4]
Change edge count pitch threshold	sINT	197	No	int index, int min*1000 int max*1000 int unit		[0, 63], [0, VGA image diagonal] [0, VGA image diagonal] [0, 1] (0 = pixels, 1 = millimeters) min ≤ max

Description	Command	Identifier	Usable in Run mode	Arguments[a]	Return values	Range
Get edge count pitch threshold	gINT	197	Yes	int index, int unit	- - int min*1000 int max*1000	[0, 63], [0, 1] (0 = pixels, 1 = millimeters) [0, VGA image diagonal] [0, VGA image diagonal]
Change edge count single edge polarity	sINT	199	No	int index, int polarity		[0, 63], [0, 2] (0=any, 1=bright to dark, 2=dark to bright)
Get edge count single edge polarity	gINT	199	Yes	int index, int polarity	- int polarity	[0, 63], [0, 2] (0=any, 1=bright to dark, 2=dark to bright)

[a]For information about tools indexing see Section B.2.5, “*Tools indexing*” (page 54)
[b]"0" is the only valid value for circular.

Table B.21 Command channel functions - Get names

Description	Command	Identifier	Usable in Run mode	Arguments	Return values	Range
Get name of device	gSTR	1	Yes			
Get name of reference object	gSTR	2	Yes	int referenceObject	- string name	[0,31]
Get name of Object Locator	gSTR	3	Yes	int referenceObject	- string name	[0,31]
Get name of Pixel counter	gSTR	4	Yes	int referenceObject int index[a]	- - string name	[0, 31], [0, 63]
Get name of Edge pixel counter	gSTR	5	Yes	int referenceObject int index[a]	- - string name	[0, 31], [0, 63]
Get name of Pattern	gSTR	6	Yes	int referenceObject int index[a]	- - string name	[0, 31], [0, 63]
Get name of Blob	gSTR	7	Yes	int referenceObject int index[b]	- - string name	[0, 31], [0, 7]
Get name of Polygon	gSTR	8	Yes	int referenceObject	-	[0, 31],

Description	Command	Identifier	Usable in Run mode	Arguments	Return values	Range
				int index[c]	- string name	[0, 7]
Get name of edge tool	gSTR	9	Yes	int referenceObject int index[a]	- - string name	[0, 31], [0, 63]
Get name of circle	gSTR	10	Yes	int referenceObject int index[a]	- - string name	[0, 31], [0, 63]
Get name of Measure distance	gSTR	11	Yes	int referenceObject int index[a]	- - string name	[0, 31], [0, 63]
Get name of Measure angle	gSTR	12	Yes	int referenceObject int index[a]	- - string name	[0, 31], [0, 63]
Get Edge counter name	gSTR	13	Yes	int referenceObject int index[a]	- - string name	[0, 31], [0, 4]
Get all tool names (except the Object Locator)	gSTR	14	Yes	int referenceObject int format	- - string list of names	[0, 31], [0, 1] (0=CSV format, 1=JSON format)[d]

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

[b]For information about blob indexing see Section B.2.2, “*Blob indexing*” (page 54)

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

[d]CSV (Comma separated values) in accordance with RFC 4180
JSON (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

Error code	Description
0	No error
8000	Index out of bounds, for example trying to use an image bank above 32
8001	Incorrect number of arguments, too many or too few arguments are supplied
8002	A parameter value is out of bounds, for example it was not in the allowed range as described in the command list
8003	Command with no valid identifier, for example sINT 200
8004	An invalid mode was supplied sMOD, for example sMOD 2
8005	The device is performing an operation and cannot accept new command
8006	Set commands are disabled for this interface
8007	This command is only available for GET

Table B.23 Error codes - Configuration errors

Error code	Description
0	No error
8100	Operation is not allowed in current mode
8101	The reference bank is not used on the device
8102	Operation is not allowed, for example trying sINT 20 0 when not in calibration mode
8103	Calibration mode is not enabled when trying to perform calibration or trying to remove non-existent calibration
8104	No object locator is available in current reference bank
8105	No blob tool with this index exists
8106	Polygon defect detection is not enabled
8107	No polygon with supplied index exists
8108	No pixel counter with supplied index exists
8109	No edge pixel counter with supplied index exists
8110	No pattern inspection with supplied index exists
8111	The move or rotation caused the ROI to appear outside of the FOV
8112	Trig is not activated
8113	The specified IP address was invalid, or an invalid combination of addresses was used
8113	The specified network mask was invalid
8113	The specified gateway address was invalid
8113	The combination of IP settings was invalid
8114	Calibration failed
8115	Interface not available. Only interfaces that exist in the product can be enabled/disabled
8116	No edge tool with supplied index exists
8117	No angle tool with supplied index exists
8118	No circle with supplied index exists
8119	No distance tool with supplied index exists
8120	The specified tool does not have a region (it is a measurement)
8121	The tool type of the specified tool does not support this command
8122	No tool with supplied index exists
8123	No edge count tool with supplied index exists
8124	Search region is not enabled
8125	Width or height is invalid
8126	No tools in configuration
8127	Invalid unit specified
8128	Alignment to specified control points failed

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 B.24 Command channel versions

Version	Released in	Comments
6	PIM60 1.0	First official version

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 B.25 Commands Ethernet Raw - general examples

Description	Command	ACK message
Switch to Run mode	sMOD 0	rsMOD 0
Switch to Edit mode	sMOD 1	rsMOD 0
Get device mode	gMOD	rgMOD 0 1
Set trigger mode to free-running	sINT 16 0	rsINT 16 0
Set trigger mode to triggered	sINT 16 1	rsINT 16 0
Get trigger mode	gINT 16	rgINT 16 0 0 – if free-running rgINT 16 0 1 – if triggered by Ethernet
Set defect intensity thresholds to 200-255 for polygon 2	sINT 69 2 200 255	rsINT 69 0
Get defect intensity thresholds for polygon 2	gINT 69 2	rgINT 69 0 200 255 – if OK rgINT 69 8107 polygon does not exist – if not OK
Get name of second reference object	gSTR 2 1	rgSTR 2 0 Object 2 - if OK rgSTR 2 8000 Ref bank index is not used. - if not OK
Get name of second polygon in second reference object	gSTR 8 1 1	rgSTR 8 0 Polygon 2 - if OK rgSTR 8 8107 No polygon with this index exists. - if not OK

Table B.26 Commands Ethernet Raw - device settings and actions examples

Description[a]	Command(s)	ACK message
Set device IP address Apply IP settings	sINT 120 192 168 1 110 aACT 5 0[b]	rsINT 120 0 raACT 5 0
Set device netmask Apply IP settings	sINT 121 255 255 255 0 aACT 5 0[b]	rsINT 121 0 raACT 5 0
Set device gateway (optional) Apply IP settings	sINT 122 192 168 1 1 aACT 5 0[b]	rsINT 122 0 raACT 5 0
Enter calibration mode Perform calibration	sINT 20 1 aACT 3 6	rsINT 20 0 raACT 3 0 75[c]

[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>")
```

```
CLOSE SOCKET
```

URL template (replace items in "<>")

```
POST /HandleConfig HTTP/1.1\r\nHost: <IP address>\r\nConnection:
Keep-Alive\r\nCookie: <Session
cookie>\r\n\r\nsopas_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\r\nHost: <IP
address>\r\nConnection: Keep-alive\r\nCookie: <Session cookie>\r\n\r\n
```

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\r\nContent-Length: <File size>\r\nHost: <IP
address>\r\nCookie: <Session cookie>\r\nConnection:
Keep-Alive\r\nContent-Type: multipart/form-data;
boundary=cd07053eab074616b9c4703b70584d7dwH!aE11@?dP/K:Pd--cd07053eab074616b9c4703b70584d7d\r\nContent-Disposition:
form-data; name="datafile"; filename="<full path to backup
file>"\r\nContent-Type: text/plain;
charset=utf-8\r\n\r\nFormatVersion=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

V

W

X

8015726/ZPP7/2018-09

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 3215-4900
E-Mail marketing@sick.com.br

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

Czech Republic
Phone +420 2 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

Hong Kong
Phone +852 2153 6300
E-Mail ghk@sick.com.hk

Hungary
Phone +36 1 371 2680
E-Mail office@sick.hu

India
Phone +91 22 6119 8900
E-Mail info@sick-india.com

Israel
Phone +972 4 6881000
E-Mail info@sick-sensors.com

Italy
Phone +39 02 274341
E-Mail info@sick.it

Japan
Phone +81 3 5309 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.gsg@sick.com

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

Slovenia
Phone +386 591 788 49
E-Mail office@sick.si

South Africa
Phone +27 11 472 3733
E-Mail info@sickautomation.co.za

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 2 2375-6288
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**

SICK
Sensor Intelligence.