

Application manual Integrated Vision

Power and productivity
for a better world™



Trace back information:
Workspace Main version a117
Checked in 2015-08-31
Skribenta version 4.6.081

Application manual

Integrated Vision

RobotWare 6.02

Document ID: 3HAC044251-001

Revision: E

The information in this manual is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this manual.

Except as may be expressly stated anywhere in this manual, nothing herein shall be construed as any kind of guarantee or warranty by ABB for losses, damages to persons or property, fitness for a specific purpose or the like.

In no event shall ABB be liable for incidental or consequential damages arising from use of this manual and products described herein.

This manual and parts thereof must not be reproduced or copied without ABB's written permission.

Additional copies of this manual may be obtained from ABB.

The original language for this publication is English. Any other languages that are supplied have been translated from English.

© Copyright 2013-2015 ABB. All rights reserved.

ABB AB
Robotics Products
Se-721 68 Västerås
Sweden

Table of contents

Overview of this manual	7
Product documentation, IRC5	9
Safety	11
1 Introduction to Integrated Vision	13
1.1 System overview	13
1.2 Vision safety	15
1.3 Getting started with Integrated Vision	16
1.4 Glossary	17
2 Installation	19
2.1 Installing the hardware	19
2.2 Installing the software	21
3 The RobotStudio user interface	23
3.1 The main window	23
3.2 Online help	24
3.3 The ribbon	25
3.4 The controller browser	29
3.5 The image capture and configuration area	30
3.6 The filmstrip	32
3.7 The palette window	34
3.8 The context window	35
3.9 Options dialog	36
3.10 The spreadsheet view	37
4 The FlexPendant user interface	41
4.1 RobotWare Integrated Vision	41
4.2 Operator view	42
5 Configuring Integrated Vision	45
5.1 Recommended working procedure	45
5.2 Preparations	46
5.3 Setting up the camera	47
5.3.1 Basic procedures	47
5.3.2 Additional camera configuration	49
5.3.3 Restricting user access	53
5.4 Setting up a new vision job	57
5.5 Setting up the image	58
5.6 Calibration	60
5.7 Adding vision tools	63
5.8 Output to RAPID	66
5.9 I/O handling	70
5.10 Preparing the RAPID program	72
5.10.1 RAPID snippets in RobotStudio	72
5.10.2 Basic programming example	73
5.10.3 Advanced programming examples	75
5.11 Starting production	77
6 Reference information	79
6.1 Relationships between coordinate systems	79
6.2 Calibration theory	82
6.3 Best practise	85
6.3.1 Evaluate performance before adopting a solution	85
6.3.2 How to mount the camera	87

Table of contents

6.3.3	Obtain accuracy	88
6.3.4	Obtain good lighting	90
6.3.5	Structuring the vision job	91
6.3.6	Init routine	92
6.3.7	Enabling and disabling vision tools during runtime	93
6.3.8	Avoid running out of space on the camera	95
6.3.9	Backup a camera to the controller	96
6.3.10	Sort items of different types	97
6.3.11	Finding multiple items of the same type	99
6.3.12	Always check that the vision target is within expected limits	101
7	RAPID reference information	103
7.1	Instructions	103
7.1.1	CamFlush - Removes the collection data for the camera	103
7.1.2	CamGetParameter - Get different named camera parameters	104
7.1.3	CamGetResult - Gets a camera target from the collection	106
7.1.4	CamLoadJob - Load a camera task into a camera	108
7.1.5	CamReqImage - Order the camera to acquire an image	110
7.1.6	CamSetExposure - Set camera specific data	112
7.1.7	CamSetParameter - Set different named camera parameters	114
7.1.8	CamSetProgramMode - Orders the camera to go to program mode	116
7.1.9	CamSetRunMode - Orders the camera to run mode	117
7.1.10	CamStartLoadJob - Start load of a camera task into a camera	118
7.1.11	CamWaitLoadJob - Wait until a camera task is loaded	120
7.2	Functions	121
7.2.1	CamGetExposure - Get camera specific data	121
7.2.2	CamGetLoadedJob - Get name of the loaded camera task	123
7.2.3	CamGetName - Get the name of the used camera	125
7.2.4	CamNumberOfResults - Get number of available results	126
7.3	Data types	128
7.3.1	cameradev - camera device	128
7.3.2	cameratarget - camera data	129
	Index	131

Overview of this manual

About this manual

This manual contains instructions for installation, configuration, and daily operation of the option Integrated Vision.

Usage

This manual should be used during installation, configuration, and maintenance of a system with the option Integrated Vision.

Who should read this manual?

This manual is intended for:

- Installation personnel
- Programmers
- Operators

Prerequisites

A maintenance/repair/installation craftsman working with an ABB robot must be trained by ABB and have the required knowledge of mechanical and electrical installation/repair/maintenance work.

References

Reference	Document ID
<i>Operating manual - RobotStudio</i>	3HAC032104-001
<i>Operating manual - IRC5 with FlexPendant</i>	3HAC050941-001
<i>Operating manual - Trouble shooting IRC5</i>	3HAC020738-001
<i>Technical reference manual - RAPID Instructions, Functions and Data types</i>	3HAC050917-001
<i>Technical reference manual - RAPID overview</i>	3HAC050947-001
<i>Technical reference manual - System parameters</i>	3HAC050948-001
<i>Product manual - IRC5</i>	3HAC021313-001
<i>Product manual - IRC5 Compact</i>	3HAC035738-001
<i>Product manual - IRC5 Panel Mounted Controller (IRC5 with main computer DSQC 639)</i>	3HAC027707-001
<i>Product manual - IRC5</i>	3HAC047136-001
<i>Product manual - IRC5 Compact</i>	3HAC047138-001
<i>Product manual - IRC5 Panel Mounted Controller (IRC5 with main computer DSQC1000)</i>	3HAC047137-001
<i>Product manual - IRB 14000</i>	3HAC052983-001

Continues on next page

External references

The following external manuals are included on the user documentation DVD:

Reference	Document ID
<i>Cognex In-Sight® 7000 Series Vision System Installation Manual</i> This manual describes how to install the vision camera.	P/N 597-0138-01

Revisions

Revision	Description
-	Released with RobotWare 5.15.01. First release.
A	Released with RobotWare 5.15.03 and RobotWare 5.60. Updates and corrections throughout the manual, among others: <ul style="list-style-type: none">• The list of limitations is updated, see Introduction to Integrated Vision on page 13.• The installation chapter is updated and restructured, see Installation on page 19.• The options dialog is added, see Options dialog on page 36.• The language settings for vision parameters is changed, see Changing the language on page 38.• Information about removing a camera, updating the camera firmware, and connecting to a camera emulator is updated and added, see Setting up the camera on page 47.• The best practise chapter is updated with multiple new sections, see Best practise on page 85.
B	Released with RobotWare 5.61. Updates and corrections throughout the manual, among others: <ul style="list-style-type: none">• The error recovery <code>ERR_CAM_COM_TIMEOUT</code> is added to several RAPID instructions.
C	Released with RobotWare 6.0. <ul style="list-style-type: none">• New functionality to restrict user access, see Restricting user access on page 53.• Preprinted calibration plates are now available, see Camera calibration on page 60.• The best practise chapter is updated with a new section, see Always check that the vision target is within expected limits on page 101.• Minor corrections.
D	Released with RobotWare 6.01. <ul style="list-style-type: none">• Updated references to other manuals, see References on page 7.• Updated the list of required hardware and software, see Checklist on page 16.• Updated the installation procedure, see Installing the hardware on page 19.• Minor corrections.
E	Released with RobotWare 6.02. <ul style="list-style-type: none">• Updated the information about Image trigger on page 58.• Added the argument <code>\AwaitComplete</code> to the RAPID instruction CamReqImage - Order the camera to acquire an image on page 110.• The RAPID instructions, functions, and data types are now also available in <i>Technical reference manual - RAPID Instructions, Functions and Data types</i>.

Product documentation, IRC5

Categories for user documentation from ABB Robotics

The user documentation from ABB Robotics is divided into a number of categories. This listing is based on the type of information in the documents, regardless of whether the products are standard or optional.

All documents listed can be ordered from ABB on a DVD. The documents listed are valid for IRC5 robot systems.

Product manuals

Manipulators, controllers, DressPack/SpotPack, and most other hardware is delivered with a **Product manual** that generally contains:

- Safety information.
- Installation and commissioning (descriptions of mechanical installation or electrical connections).
- Maintenance (descriptions of all required preventive maintenance procedures including intervals and expected life time of parts).
- Repair (descriptions of all recommended repair procedures including spare parts).
- Calibration.
- Decommissioning.
- Reference information (safety standards, unit conversions, screw joints, lists of tools).
- Spare parts list with exploded views (or references to separate spare parts lists).
- Circuit diagrams (or references to circuit diagrams).

Technical reference manuals

The technical reference manuals describe reference information for robotics products.

- *Technical reference manual - Lubrication in gearboxes*: Description of types and volumes of lubrication for the manipulator gearboxes.
- *Technical reference manual - RAPID overview*: An overview of the RAPID programming language.
- *Technical reference manual - RAPID Instructions, Functions and Data types*: Description and syntax for all RAPID instructions, functions, and data types.
- *Technical reference manual - RAPID kernel*: A formal description of the RAPID programming language.
- *Technical reference manual - System parameters*: Description of system parameters and configuration workflows.

Continues on next page

Application manuals

Specific applications (for example software or hardware options) are described in **Application manuals**. An application manual can describe one or several applications.

An application manual generally contains information about:

- The purpose of the application (what it does and when it is useful).
- What is included (for example cables, I/O boards, RAPID instructions, system parameters, DVD with PC software).
- How to install included or required hardware.
- How to use the application.
- Examples of how to use the application.

Operating manuals

The operating manuals describe hands-on handling of the products. The manuals are aimed at those having first-hand operational contact with the product, that is production cell operators, programmers, and trouble shooters.

The group of manuals includes (among others):

- *Operating manual - Emergency safety information*
- *Operating manual - General safety information*
- *Operating manual - Getting started, IRC5 and RobotStudio*
- *Operating manual - Introduction to RAPID*
- *Operating manual - IRC5 with FlexPendant*
- *Operating manual - RobotStudio*
- *Operating manual - Trouble shooting IRC5, for the controller and manipulator.*

Safety

Safety of personnel

When working inside the robot controller it is necessary to be aware of voltage-related risks.

A danger of high voltage is associated with the following parts:

- Devices inside the controller, for example I/O devices, can be supplied with power from an external source.
- The mains supply/mains switch.
- The power unit.
- The power supply unit for the computer system (230 VAC).
- The rectifier unit (400-480 VAC and 700 VDC). Capacitors!
- The drive unit (700 VDC).
- The service outlets (115/230 VAC).
- The power supply unit for tools, or special power supply units for the machining process.
- The external voltage connected to the controller remains live even when the robot is disconnected from the mains.
- Additional connections.

Therefore, it is important that all safety regulations are followed when doing mechanical and electrical installation work.

Safety regulations

Before beginning mechanical and/or electrical installations, ensure you are familiar with the safety regulations described in *Operating manual - General safety information*¹.

¹ This manual contains all safety instructions from the product manuals for the manipulators and the controllers.

This page is intentionally left blank

1 Introduction to Integrated Vision

1.1 System overview

What is Integrated Vision?

The purpose of ABB's Integrated Vision system is to provide a robust and easy-to-use vision system for general purpose Vision Guided Robotics (VGR) applications.

The system includes a complete software and hardware solution that is fully integrated with the IRC5 robot controller and the RobotStudio programming environment. The vision capability is based on the *Cognex In-Sight®* smart camera family, with embedded image processing and an Ethernet communication interface.

RobotStudio is equipped with a vision programming environment that exposes the full palette of *Cognex EasyBuilder®* functionality with robust tools for part location, part inspection and identification. The RAPID programming language is extended with dedicated instructions and error tracing for camera operation and vision guidance.

Hardware

The camera system is based on the *Cognex In-Sight® 7000 series*, but any *Cognex In-Sight®* camera can be used. The camera is supplied with 24 VDC and Ethernet from the controller.

The cameras are connected to the supplied Ethernet switch. The maximum number of cameras are three.

For more information see *Cognex In-Sight® 7000 Series Vision System Installation Manual*.

Software

The software solution is based on three components – RobotStudio, the IRC5 controller with the RAPID programming language, and the FlexPendant.

RobotStudio presents vision and robot configuration parameters side by side, providing a convenient VGR programming environment.

The IRC5 controller enables easy creation of RAPID programs that make maximum use of the camera system's capability. Among other features, the controller has a RAPID interface with a pre-established camera communication interface and vision target queue handling.

The FlexPendant is equipped with an operator interface to allow supervision of the system when deployed in production.

Limitations

- A FlexPendant of type SxTPU3 is required to run Integrated Vision.
- The Integrated Vision add-in does not run on the 64-bit version of RobotStudio.

Continues on next page

1 Introduction to Integrated Vision

1.1 System overview

Continued

- The integrated vision cameras are only intended for use on the service port network of the robot controller, and must be configured using the RobotStudio Integrated Vision add-in.
- A vision program created with RobotStudio contains special configuration data. Existing vision programs made with *Cognex EasyBuilder®* will have to be modified by the Integrated Vision add-in in order to be compatible with the IRC5 controller.

Any other use has not been verified and may result in unpredictable behavior.

1.2 Vision safety

General principles

Using a vision sensor for robot guidance requires that the user observes caution when handling, installing, and configuring the system.

The user must always assume that the vision sensor is active even if the manipulator is not moving.

Before entering the working range of the manipulator, the user must take the following precautions to prevent the manipulator from starting to move.



WARNING

If work must be carried out within the manipulator's work area, the following points must be observed:

- The operating mode selector on the controller must be in the manual mode position to render the enabling device operational and to block operation from a vision sensor, a computer link, or a remote control panel.
- Anyone entering the manipulator working space must always bring the FlexPendant with him/her. This is to prevent anyone else from taking control of the manipulator without his/her knowledge.
- The enabling device must never be rendered inoperational in any way.
- During programming and testing, the enabling device must be released as soon as there is no need for the manipulator to move.

1 Introduction to Integrated Vision

1.3 Getting started with Integrated Vision

1.3 Getting started with Integrated Vision

Checklist

Before setting up the system, make sure that the necessary preparations have been made.

Hardware

- Complete IRB robot system with manipulator, IRC5 controller, and a FlexPendant of type SxTPU3.
- The option *Integrated Vision interface* which includes:
 - Ethernet switch.
 - Ethernet cable for connecting the camera to the switch.
 - Ethernet cable for connecting the switch to the service port of the IRC5 main computer.
 - Ethernet cable for connecting the switch to the cabinet service port connector.
 - Customer power supply.
 - Wires for supplying 24 VDC from the customer power supply to the Ethernet switch.
 - Cable for supplying 24 VDC to the camera.
- A pointing tool to be mounted on the robot for accurately defining work objects.
- PC.
- Ethernet cable for connecting the PC to the controller.
- *Cognex In-Sight®* camera.
- Camera lens.
- Lighting device.

Software

- RobotStudio 5.60 or later, full installation. A RobotStudio license is not required.
- RobotWare 5.60 or later.
- A RobotWare license with the option *Integrated Vision* enabled.

For information on how to configure Integrated Vision, see [Configuring Integrated Vision on page 45](#).

1.4 Glossary

Term list

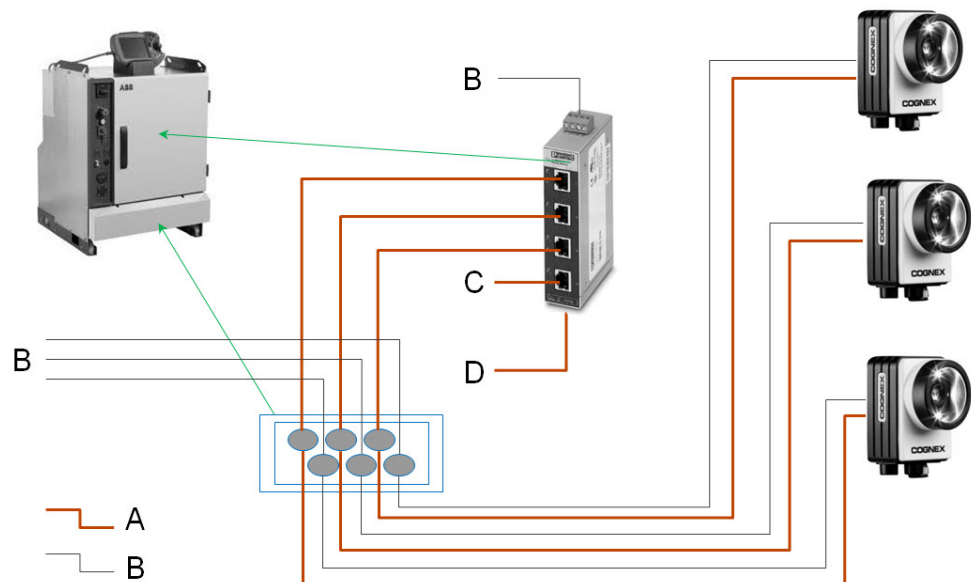
Term	Explanation
camera calibration	To compute the transformation that converts pixel coordinates in the image to physical coordinates in space. The calibrated frame is most commonly defined by a checkerboard calibration plate.
camera to robot calibration	The relation between the calibrated camera frame, computed by the camera calibration, and the robot base frame. The result is normally stored as a work object user frame.
checkerboard calibration plate/pattern	A calibration pattern that is used for calibrating cameras. The pattern is placed in the field of view during calibration.
<i>Cognex EasyBuilder®</i>	The vision camera software from <i>Cognex</i> that has been seamlessly integrated into RobotStudio to provide Integrated Vision.
<i>Cognex In-Sight®</i>	The specific product line of smart cameras supported by Integrated Vision.
fiducial	A fiducial marker, or fiducial, is an object placed in the calibration plate image for use as a point of reference for the origin of coordinates.
pointing tool	A robot tool with a well defined tool center point (TCP) that is used for accurately teaching targets to the robot.
smart camera	A camera with embedded image processing.
snippet	Snippets are pieces of predefined RAPID code which can be inserted into the RAPID program.
VGR	Vision guided robot, vision guided robotics.
vision calibration	The camera calibration and camera to robot calibration combined. The result is a common frame which allows the robot to accurately move to vision targets.
vision job	A vision job, or job, is the vision program loaded into the camera.

This page is intentionally left blank

2 Installation

2.1 Installing the hardware

Overview



xx1200000992

A	Ethernet
B	24V power supply from the customer power supply to the switch and to the cameras
C	Ethernet between the switch and the controller cabinet service port (inside)
D	Ethernet between the switch and the main computer service port

Installation procedure

Connect the following components and cables according to the figure:

	Action
1	Make sure the controller mains switch is turned off.
2	Connect an Ethernet cable from the controller cabinet service port (inside) to one of the four Ethernet connectors on the switch.
3	Connect the Ethernet cable from each camera, through the cable gland on the controller cabinet, to any available Ethernet connector on the switch. Carefully strip off 20 mm of insulation and strap the cables to the ground plate on the cable gland.
4	Connect the 24 VDC power cables from each camera, through the cable gland on the controller cabinet, to the 24 VDC power supply. Carefully strip off 20 mm of insulation and strap the cables to the ground plate on the cable gland.

Continues on next page

2 Installation

2.1 Installing the hardware

Continued

For more information see the *Cognex In-Sight® 7000 Series* manual, and the product manual and circuit diagram for the corresponding controller. See [References on page 7](#).



CAUTION


When using a robot held camera, or by other means moving camera, it is important to have a good cable routing along the robot arm.

When routing the cables caution has to be taken to avoid mechanical stress on the connectors, allowing sufficient bend radius for the cables, and minimizing the wear on the cables. It is also recommended to fit the cables with extra wear protection at the attachment points and at especially exposed areas.

2.2 Installing the software

Installing RobotStudio

The Integrated Vision configuration environment is designed as a RobotStudio add-in, and is included in the standard installation.

Action	
1	Install RobotStudio. Select complete installation.
2	Start RobotStudio.
3	Go to the Controller tab on the ribbon menu and start the Integrated Vision add-in. After the add-in has finished loading a new tab named Vision is visible.
 Tip When a controller is connected, the Integrated Vision add-in can be started from the context menu of the controller node, the vision system node, or the camera node in the controller browser.	

For more information, see *Operating manual - RobotStudio*.



Note

It is not recommended to change the RobotStudio language after the configuration of an Integrated Vision system has started.

For more information, see [Changing the language on page 38](#).



Tip

A 3D model of the vision camera is included in the RobotStudio library.

Installing RobotWare to the IRC5 controller

A RobotWare license with the option *Integrated Vision* enabled is required to to run Integrated Vision.

Use RobotStudio to configure, build, and download a RobotWare system to the IRC5 controller.

For more information, see *Operating manual - RobotStudio*.

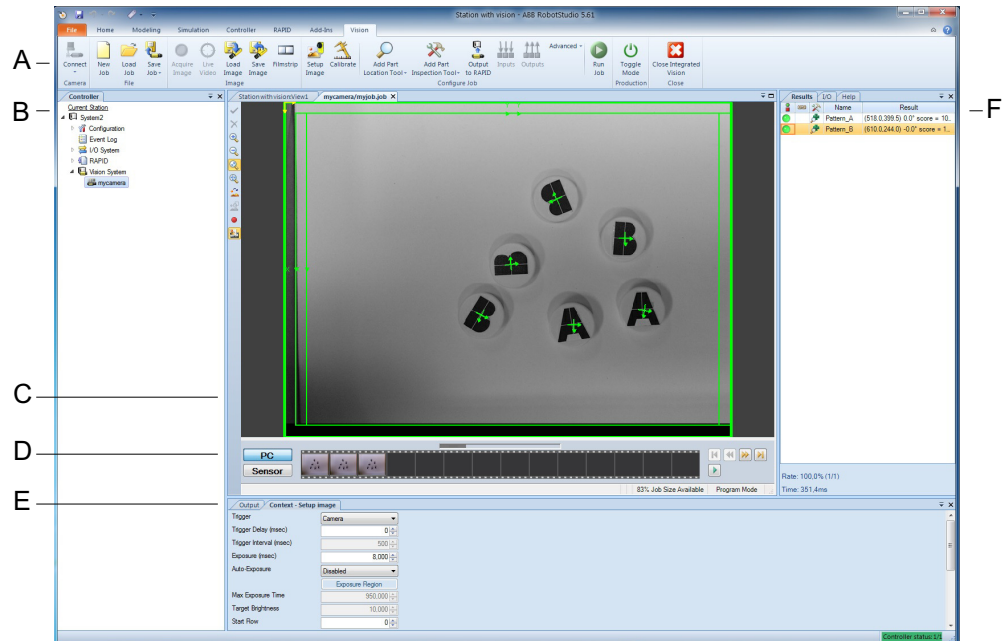
This page is intentionally left blank

3 The RobotStudio user interface

3.1 The main window

Overview of the main window

This section presents an overview of the Integrated Vision graphical user interface.



xx120000989

	Parts	Description
A	Ribbon	Displays groups of icons organized in a logical sequence of function.
B	Controller browser	The vision system node displays all vision cameras on the network.
C	Image capture and configuration area	Displays an image acquired by the vision camera with configuration guides for locating and inspecting parts.
D	Filmstrip bar	Used to record a sequence of images for later analysis.
E	Context window	Contains the available properties, settings, and events of the selected controls.
F	Palette window	Following tabs are available: <ul style="list-style-type: none"> • Results tab - displays the setup of the active vision job with a list of all used location and inspection tools. • I/O tab - displays the I/O setup. • Help tab - provides online help.



Tip

If any window is accidentally closed, it can be restored from the **Customize Quick Access Toolbar**.

3 The RobotStudio user interface

3.2 Online help

3.2 Online help

General

The experienced vision camera user, familiar with the *Cognex* line of products, will recognize most of the graphical user interface for vision in RobotStudio.

Cognex EasyBuilder® is the vision camera software from *Cognex* that is seamlessly *integrated* into RobotStudio to create the Integrated Vision System for IRC5. The graphical user interface is only slightly adapted to give the best performance for Vision Guided Robotics.

This effects the online **Help** tab, since the online **Help** is also integrated without modifications of the content.

All references to parameters, settings, and technical explanations are correct. But any references to the GUI can be slightly incorrect depending on the implementation in RobotStudio.

Therefore use the online **Help** tab as the technical reference manual for all parameters and settings, and use this application manual as the reference for the graphical user interface.

The online help tab

The online **Help** tab is located in the palette window.

The online **Help** tab is a context sensitive technical reference manual for all Integrated Vision parameters and settings. This means that the content of the help tab adjusts to the operation that is currently performed in Integrated Vision. For example setting up the image, calibrating, adding location tools, etc.



Note

Most parameters and settings are only described in the online **Help** tab, and not in this application manual.

Terminology

The list below describes the most common differences in terminology between the online help tab and the Integrated Vision application.

If the online help tab says...	...then it means:
<i>EasyBuilder®</i>	RobotStudio vision add-in tab
edit acquisition settings (group box)	the Setup Image button
calibrating the image to real world units (group box)	the Calibrate button
online	Run Mode
offline	Program Mode

3.3 The ribbon

Layout of the ribbon


The Integrated Vision tab contains groups of commands organized in a logical sequence of functions that simplifies the managing of Integrated Vision projects.

The tab consists of the following groups:

Group	Functions used for
Camera	Connecting to the vision cameras and setting camera properties.
File	Creating, saving and loading vision jobs.
Image	Loading and saving images. Images can both be acquired from a vision camera or loaded from a previously saved file.
Configure Job	Work flow of configuring vision jobs, organized in a logical sequence of functions.
Production	Setting the Integrated Vision system in program mode or run mode. (Similar to the robot controller auto/manual key switch on the control panel.)

Camera group

The Camera group is used when connecting to cameras and configuring cameras.

Button	Description
Connect	Connect to the selected camera.
Disconnect	Disconnect the selected camera.  Note A camera is also disconnected when the corresponding tab in the image capture and configuration area is closed.
Rename	Rename the controller to make the camera available in RAPID with a name.
Network Settings	Change the IP-settings of a connected camera.
Set Date and Time	Set the date and time of the camera.
User Access Settings	Edit the list of users on the camera and their associated privileges.
Set Controller User	Select the user profile to be used by the controller when communicating with the camera.
Add Sensor	Find a camera on the network and change its IP-settings.
Properties	View the camera properties.

The same settings can also be accessed from the camera node context menu, see [Camera node context menu on page 29](#).

File group

The File group is used for loading and saving jobs.

Button	Description
New Job	Create a new vision job.
Load Job	Load a vision job from file into the camera.

Continues on next page

3 The RobotStudio user interface

3.3 The ribbon

Continued

Button	Description
Save Job	Save the vision job to file.
Save Job As	

For more information on jobs and where to save them, see [Setting up a new vision job on page 57](#).

Image group

The **Image** group is used for handling images.

Button	Description
Acquire Image	Get a new image.
Live Video	Turn camera live video mode on/off.
Load Image	Load an image from file into the camera.
Save Image	Save the current image to file.
Show Filmstrip	Edit record/playback settings.

Configure Job group

The **Configure Job** group is used for making the necessary settings to the current vision job. What objects the camera should locate, how to inspect the objects, and finally how the vision data should be transferred to the RAPID program.

Button	Description
Setup Image	Modify image acquisition parameters.
Calibrate	Calibrate image to real world units.
Add Part Location Tool	Add part location tool to the vision job.
Add Part Inspection Tool	Add an inspection tool to the vision job.
Output to RAPID	Select which vision job results are to be available in RAPID.
Inputs	Define camera inputs.
Outputs	Define camera outputs.
Advanced	Advanced settings and editing mode.
Run Job	Run the vision job.

For more information about configuring Integrated Vision, see [Configuring Integrated Vision on page 45](#).

Production group

The **Run Mode** button in the **Production** group is used for manually switching the camera between program mode and run mode.

When running the vision system in production, switching between program mode and run mode is done from the RAPID program.

Button	Description
Run Mode	Switch between program mode (button inactive) and run mode (button active).

Continues on next page

For information on setting the run mode from the RAPID program, see [Preparing the RAPID program on page 72](#).

Advanced settings

Following functionality and settings can be found under the **Advanced** button.

Button	Description
View Spreadsheet	Advanced mode editing. See section The spreadsheet view on page 37 .
Unprotect Job / Protect Job	Protects locked cells in the spreadsheet from being edited. See section The spreadsheet view on page 37 .
Displayed Image Settings	Configures the resolution and frame rate of live and online images. See section Displayed Image Settings dialog on page 27 .
Job Size	Displays the size of the current job. See section Job Size dialog on page 27 .
Update Firmware	Update the firmware of the vision camera. See section Updating the camera firmware on page 50 .

The same settings can also be accessed from the camera node context menu, see [Camera node context menu on page 29](#).

Displayed Image Settings dialog

The **Displayed Image Settings** dialog configures the resolution and frame rate of live and online images.

Setting	Description
Live Acquisition • Resolution	Full, half and quarter resolution for live images.
Live Acquisition • Limit Maximum Rate	Enables the Maximum Rate (Frames/sec) text box.
Live Acquisition • Maximum Rate (Frames/sec)	Specifies the maximum number of images the sensor may send per second (0.016 to 100). This feature can be used to reduce network traffic by limiting the number of images sent.
Online (Run mode) • Resolution	Full, half and quarter resolution for live images. Optimized gives half when using Camera or Continuous mode, and full when using Manual mode.
Online (Run mode) • Limit Maximum Rate	Enables the Maximum Rate (Frames/sec) text box.
Online (Run mode) • Maximum Rate (Frames/sec)	Specifies the maximum number of images the sensor may send per second (0.016 to 100). This feature can be used to reduce network traffic by limiting the number of images sent.

Job Size dialog

The **Job Size** dialog is used to configure the job size memory allocation that can be utilized by the vision system. The slider control allows to configure the memory allocation of the vision system. Adjust the slider to set the job's memory size allocation.

Continues on next page

3 The RobotStudio user interface

3.3 The ribbon

Continued

Adjusting the **Job Size** limit of a vision system requires the vision system be rebooted, and will erase all job and settings files from non-volatile flash memory. Save or backup the job before continuing.

3.4 The controller browser

Layout of the the controller browser

The **Controller** browser is a hierarchical display of controller and configuration elements as found in the **Controller** tab view in RobotStudio.

For a detailed description of the contents and functionality of the **Controller** tab view in general, see *Operating manual - RobotStudio*.

For Integrated Vision a separate **Vision System** node is available. Under this node all cameras connected to the robot appear. Named cameras are identified with their name, unnamed cameras are identified with their MAC-id.



Tip

When a controller is connected, the Integrated Vision add-in can be started from the context menu of the controller node, the vision system node, or the camera node in the controller browser.

Camera node context menu

On the individual camera node, a context menu is available with similar content as the **Camera** group, and the **Advanced** button on the ribbon.

Button	Description
Integrated Vision	Start the Integrated Vision add-in.
Connect	Connect to the selected camera.
Disconnect	Disconnect the selected camera. <div data-bbox="710 1245 770 1301" data-label="Image"> </div> Note A camera is also disconnected when the corresponding tab in the image capture and configuration area is closed.
Rename	Rename the controller to make the camera available in RAPID with a name.
Restart	Restart the camera.
Properties	View the camera properties.
Advanced	Same settings as under the Advanced button on the Configure Job group, see Advanced settings on page 27 .

3 The RobotStudio user interface

3.5 The image capture and configuration area

3.5 The image capture and configuration area

Layout of the image capture and configuration area

The Image capture and configuration area is used to display the image that should be processed with Integrated Vision. The image can be acquired live by any online vision camera, or be loaded from a file.

Depending on the situation, different graphical configuration guides for locating and inspecting parts, coordinate systems, and graphic data appear in the area.

Buttons in the image capture and configuration area

Button	Description
Accept changes (ENTER)	Used in some dialogs to accept a change, for example when modifying the calibration region.
Cancel changes (ESC)	Used in some dialogs to cancel a change, for example when modifying the calibration region.
Zoom In	Zoom buttons.
Zoom Out	
Zoom to Fit	
Zoom to Fill	
Rotate Image	Rotate the image 90 degrees.
Show Selected Tool Graphics Only	Show graphics for selected tool only.
Filmstrip	Show the filmstrip bar.
Record	Start recording images to the PC.

Shortcuts in the image capture and configuration area

Functionality	Shortcut
Zoom	SHIFT + scroll wheel.
Zoom using window	CTRL + left mouse button while dragging the mouse.
Pan	CTRL + SHIFT + left mouse button while dragging the mouse.

Tabs in the image capture and configuration area

Each camera is displayed in a separate tab in the Image capture and configuration area. The heading of the tab displays the name of the camera together with the name of the current job.



Note

To close the tab is the same as disconnecting from the camera.

Continues on next page

The status bar

On the bottom of the image capture and configuration area there is a status bar displaying the following information:

Information	Description
(R, G, B)@(x, y)	Syntax: (Red, Green, Blue)@(x-coordinate, y-coordinate) This information is only visible when the cursor is moved over the image area. The information shows the color and the coordinates of the pixel at the location of the cursor.
% Job Size Available	Displays the memory status of the vision camera. The percentage value shows how much memory is left to use. When the value is close to zero, all memory is consumed.
Program Mode / Run Mode	Displays the same status as the Run Mode button in the ribbon.

3.6 The filmstrip

Introduction

The filmstrip is used to play back images recorded to the PC or review images and results stored to the sensor.

This functionality is useful for trouble shooting and fault tracing activities. For example if there are intermittent problems of locating and inspecting parts during production.

Filmstrip settings

Click the large **Filmstrip** button on the ribbon to access the **Display and record** dialog settings window.

There are three main groups of settings, sensor settings, record settings, and playback settings.

For information about the different **Filmstrip** settings, see the *Filmstrip* section in the online **Help** tab.

The Filmstrip bar

Click the small **Filmstrip** button in the image capture and configuration area to show the filmstrip bar.

Pass images are indicated with green color, and fail images are indicated with red color.

For more information on the pass and fail status of vision tools see [Pass and fail of vision tools on page 64](#).

PC filmstrip

After the filmstrip settings have been configured, and the **Record** button is pressed, the pass and fail images are recorded and viewed in the filmstrip bar. The images can be saved from an online or offline camera to any folder on the PC.

The PC playback mode can be used to play back the recorded images, the image is sent to the vision camera and the job runs against the image being played back. Up to 10,000 images can be saved to the PC.

Sensor filmstrip

When the sensor is online and acquiring images, the sensor filmstrip can be used to monitor a job's performance.

As images are acquired, the job results including the acquired image and accompanying job data are stored to the sensor's RAM. As results are stored to the sensor, a pass or fail graphic is added to the filmstrip.

When a result is highlighted in the filmstrip, the filmstrip display changes from graphics to thumbnail images and the corresponding image is loaded to the display area.

Use the controls in the **Sensor Settings** tab in the **Display and record** dialog settings window to configure the sensor's behavior.

Continues on next page

Up to 20 results can be saved, depending on the vision system's resolution and available RAM.

3 The RobotStudio user interface

3.7 The palette window

3.7 The palette window

Results tab

The **Results** tab displays the performance of each tool, and allows to troubleshoot or optimize the job settings.

The **Results** tab can be used to:

- Determine at a glance which tools are passing (green) or failing (red) by monitoring the semaphore of each tool.
- Visually identify the type of tool by its tool icon.
- Identify the tool by name.
- Double-click on a tool to modify it. The tool then appears in the **Context** window. Edit the parameters or graphics as needed.

It is also possible to right-click on a tool in the **Results** tab to access a short editing menu, which allows to copy, paste, delete, or edit the selected tool.

I/O tab

The I/O tab shows the active (green) or inactive (grey) status of each of the job's input and output lines. This allows monitoring of the I/O lines.

Help tab

See section [Online help on page 24](#).

3.8 The context window

Introduction

The context window automatically updates to display the parameters for the selected application step. The context window is designed to guide the user through the step by presenting general parameters on the left side, and increasingly more specific parameters to the right.

In addition, depending on the tool added, the pane for the locate part and inspect part steps might also display visual aids, such as a feedback table or graph (to help configure the parameters for the application) and a range limits tab (for setting pass/fail criteria). Once a job is built, the steps can be revisited in any order, so that job parameters can be corrected and fine-tuned until the desired result is achieved.

Most of the steps require that the parameters be configured in a specific order. See the **Help** tab in the palette window for instructions for each step while developing the job.

Usage

The following functions uses the context window for settings and configurations:

- **Filmstrip**
- **Setup Image**
- **Calibrate**
- **Add Part Location Tool**
- **Add Part Inspection Tool**
- **Output to RAPID**
- **Inputs**
- **Outputs**

3.9 Options dialog

Accessing the options dialog

	Action
1	Click the File tab.
2	Click Options .
3	Scroll to the Integrated Vision section.

Integrated Vision options

Topic	Setting	Description
Behavior	Immediate Feed-back	If checked, the search region in the Image capture and configuration area will be processed in <i>real-time</i> when adding and configuring vision tools. If unchecked (default), the search region in the Image capture and configuration area will be processed at <i>mouse click</i> when adding and configuring vision tools.
Camera Emulator	Camera Model	Determines the behavior of the camera emulator. When the emulator is used to view an image, the image is cropped to the size of the emulated sensor. Therefore it is necessary to select the correct camera model to be emulated. The drop-down list includes all supported <i>Cognex In-Sight®</i> vision systems, and the vision tools and functions that are available on those sensors. Default camera model is is7200. For more information, see Connecting to a camera emulator on a virtual controller on page 51 .

3.10 The spreadsheet view

Introduction

Integrated Vision inserts snippets into a spreadsheet when adding vision tools or otherwise editing the job. This is normally not shown to the user.

The spreadsheet view is an advanced mode that displays the configured job with all included vision tools in a spreadsheet mode. This is mainly intended for advanced users familiar with the *Cognex In-Sight Explorer®* software.



CAUTION

An incorrect use of the spreadsheet may result in unmanageable errors in the vision job or the RAPID program.

General

The spreadsheet is similar to other spreadsheet applications in terms of its standard operations and functionality, such as manipulating blocks of cells, editing cells, referencing cells, and inserting functions. Organized into a table of cells arranged in 400 rows (numbered 0 to 399) and 26 columns (labeled A to Z), each cell is identified by its column letter and row number. For example, cell A2 is located at the intersection of column A and row 2.

The spreadsheet is configured one cell at a time. The content of each cell is defined as a formula, and any piece of information inserted into a cell (whether a single numeric value or a complex vision processing function) is considered a part of the formula.

The spreadsheet is equivalent with the camera memory. That is when the spreadsheet is full, the camera memory is full.

For more information on using the spreadsheet see the *Cognex In-Sight Explorer®* help.



Note

A spreadsheet can contain a maximum of 4,096 active cells. Inserting a function (such as *FindBlobs*) that, in turn, inserts multiple vision data access functions into the spreadsheet can cause the spreadsheet to attempt to exceed the limit. In this case, no warning dialog is displayed. The cells over the 4,096 limit are simply not inserted.

Transfer data to RAPID

Any data that gets updated with each image acquisition can be transferred from the spreadsheet to RAPID using the **Output to RAPID** table. To make the data visible in the **Output to RAPID** table, a **Symbolic Tag** must be added to the cell holding the data.

The name of the tag must have the format `<Group>.<Result>`, for example `mydata.data`.

The **Symbolic Tag** is added from the right-click menu.

Continues on next page

3 The RobotStudio user interface

3.10 The spreadsheet view

Continued

Changing the language

When changing the application language for the GUI in RobotStudio, the GUI language of the Integrated Vision add-in will also change.

The integrated *Cognex EasyBuilder®* software creates new jobs in the currently set GUI language and populates the cells of the spreadsheet with translated names. This results in that when the job is opened using another GUI language setting, a lot of the data will have names in the language that was set when first creating the job. Any subsequent additions are created in the currently set language.

For this reason, it is highly recommended to use the same GUI language while programming and using the job.

Even though some labels describing data and settings are created in the local language, RobotStudio is set to create data with names (symbolic tags) in English. The reason is that data which may be accessed from RAPID shall not require language encoding, for example when using `CamGetParameter`.



Note

It is not recommended to change the RobotStudio language after the configuration of an Integrated Vision system has started.

Shortcuts in the spreadsheet view

Shortcut	Functionality
CTRL + 1	Open the Format Cells dialog

Right-click menu

Right-clicking within the spreadsheet displays a menu that allows various spreadsheet operations to be performed:

Setting	Description
Cut	Cuts the selected cell(s).
Copy	Copies the selected cell(s) to the clipboard.
Paste	Pastes cell(s) that were cut or copied to the clipboard.
Insert	Inserts rows or columns equal to the number of rows or columns selected. If entire rows or columns are not selected, the Insert dialog opens to insert cells, rows or columns.
Delete	Deletes the selected rows or columns. If entire rows or columns are not selected, the Delete dialog opens to delete cells, rows or columns.
Clear Contents	Clears the contents of the active cell(s).
Insert Function	Opens the dialog to insert a function into the active cell. This option is available only if the active cell does not contain a function.
Edit Function	Opens the property sheet if the active cell contains a function with an associated property sheet, otherwise opens the Insert Function dialog. This option is available only if the active cell contains a function.
Insert Absolute Reference	Inserts an absolute cell reference into the active cell.

Continues on next page

Setting	Description
Insert Relative Reference	Inserts a relative cell reference into the active cell.
Insert Comment	Opens the dialog to insert a new comment for the active cell. This option is available only if the active cell does not contain a comment.
Edit Comment	Opens the dialog to edit an existing comment for the active cell. This option is available only if the active cell contains a comment.
Insert Symbolic Tag	Opens the symbolic tag editor to insert a new symbolic tag for the active cell. This option is available only if the active cell does not contain a symbolic tag.
Edit Symbolic Tag	Opens the symbolic tag editor to edit an existing symbolic tag for the active cell. This option is available only if the active cell contains a symbolic tag.
Cell Graphic	Enters interactive graphics mode to edit the cell graphic. This option is available only when the active cell contains a function that has an associated cell graphic.
Cell State	Opens the dialog to enable or disable the execution of cells either explicitly or conditionally, based upon the value of a referenced cell.
Set Job Pass/Fail	Opens the setup dialog to monitor a cell, which contains a numeric result, to determine the job's overall pass/fail status. The job status can be sent to the RAPID program.
Custom View Settings	Opens the dialog to configure the properties of the custom view of the spreadsheet.
EasyView Settings	Opens the dialog to customize how data is displayed.
Format	<ul style="list-style-type: none"> • Format Cells: Opens the dialog to format the number, alignment, font and protection of the selected cell(s). • Row Height: Opens the dialog to adjust the height of one or more spreadsheet rows. • Column Width: Opens the dialog to adjust the width of one or more spreadsheet columns. • Hide: Hides the selected row(s) or column(s). • Unhide: Reveals the hidden row(s) or column(s) that lie between the selected row(s) or column(s).
Snippet	<ul style="list-style-type: none"> • Import: Opens the dialog to import the snippet, saved as a .CXD file, into the spreadsheet. This data file can be loaded from the snippets folder on the PC. • Export: Opens the dialog to export the snippet, saved as a .CXD file, to the snippets folder on the PC.
Import Cells	Imports cell data as a .CXD file into the spreadsheet.
Export Cells	Exports cell data as a .CXD file.

Unprotect Job / Protect Job dialog

The protect job dialog prevents the modification of all locked cells in the job. This can be useful for jobs with cells that must remain unchanged for the job to function correctly, but allow the rest of the job to be customized.

A job is protected with a user defined password, which can remain blank. The password is required to unprotect the job.

When the job is protected, formulas within any locked cell cannot be modified, but the formatting of the locked cells can still be modified (font, color, etc.). The formula

Continues on next page

3 The RobotStudio user interface

3.10 The spreadsheet view

Continued

of a locked cell remains visible, but grayed out, when the mouse pointer is resting on the locked cell.



Note

Use the **Protection** tab in the **Format Cells** dialog to lock cells.

4 The FlexPendant user interface

4.1 RobotWare Integrated Vision

Introduction

This section gives an overview of the graphical user interface of the Integrated Vision application on the FlexPendant. Views, buttons, and other parts of the user interface are described in respect of their content and how they are accessed.

For more information on how to use the FlexPendant in general, see *Operating manual - IRC5 with FlexPendant*.

Use this procedure to start RobotWare Integrated Vision.

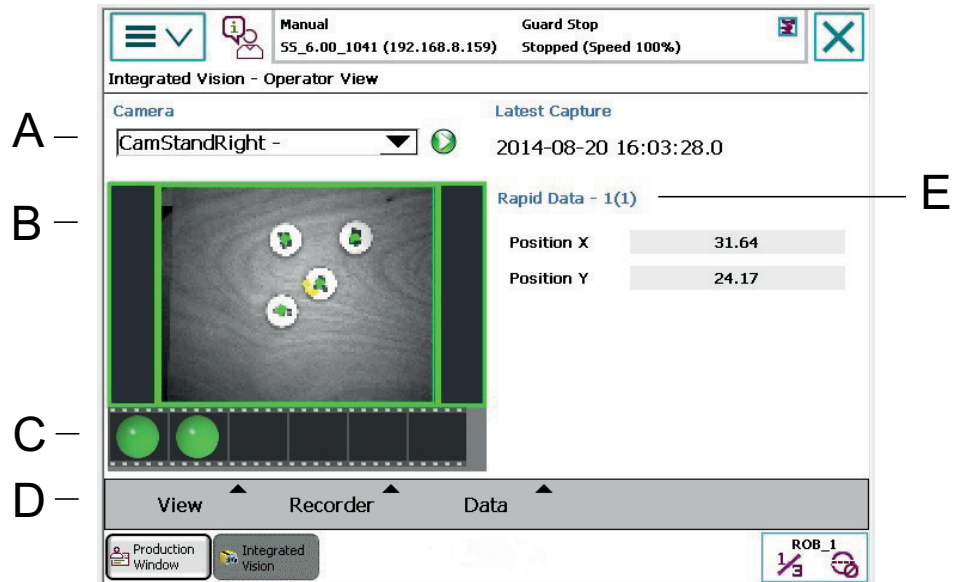
	Action
1	Tap the ABB menu.
2	Tap Integrated Vision .

4 The FlexPendant user interface

4.2 Operator view

4.2 Operator view

The Operator View interface



	Parts	Description
A	Camera	Selects connected camera and displays active job.
B	Image area	Displays an image acquired by the vision camera.
C	Recorder bar	Used to record a sequence of images for later analysis.
D	View	Used to change the view to show either image, results, or both.
	Recorder	Used to show and freeze the recorder bar and to save images.
	Data	Used to setup the results view.
E	Results area	Used to display camera results or RAPID data. User configured.

View settings

The default setting is to display image and results side by side, but it is also possible to only view either the image or the results.

Setting	Description
Image & Results	Displays image and results side by side.
Image	Displays the image only.
Results	Displays the results only.



Tip

Tap the image area to go to full screen mode. Full screen mode is active for 30 seconds, then it is automatically disabled.

Continues on next page

Recorder settings

The recorder bar on the FlexPendant is a simplified version of the filmstrip bar in RobotStudio, see [The filmstrip on page 32](#).

Pass images are indicated with green color, and fail images are indicated with red color.

Setting	Description
Show	Shows the recorder bar.
Freeze	Freezes the recorder bar. When the recorder bar is frozen it is possible to view the individual images by tapping on them.
Save	Save can only be performed when the recorder bar is frozen. The active image is saved as a <i>.bmp</i> file with the current timestamp to the "...\\HOME\\IV" folder of the controller flash disk.

Data settings

The results area can be customized to display any RAPID data that is of interest to the user. The layout is job specific, so a new layout needs to be created for each job. It is also possible to configure a default layout that can be displayed when no job is loaded.

**Note**

Only RAPID data declared as persistent (**PERS**) can be displayed in the results area.

Setting	Description
Configure	Configure the layout of the results area for the job by adding labels and RAPID data.
Camera Results	Displays the vision camera parameters that are mapped in the Output to RAPID dialog. See Output to RAPID on page 66 .
Default	Displays the default layout of the results area when no job is loaded.

The configurations are saved in the *IVSetup.xml* file found in the "...\\HOME\\IV" folder on the controller flash disk. Users familiar with xml-files can edit the *IVSetup.xml* file directly instead of using the configuration tool on the FlexPendant. Restart the RobotWare Integrated Vision application on the FlexPendant to apply the changes.

This page is intentionally left blank

5 Configuring Integrated Vision

5.1 Recommended working procedure

General

This section describes the recommended working procedure when creating a new vision application. The working procedure helps to understand the dependencies between the different objects. A good approach when creating a new application is to start with the basic functionality. When that works as expected, expand the application.

A prerequisite is that all steps in the hardware and software installation procedures must have been performed, see chapter [Installation on page 19](#).

Basic steps

Use this procedure to create a new vision application.

	Action	Further information
1	Make some initial preparations.	Preparations on page 46
2	Setup the camera.	Setting up the camera on page 47
3	Create a new vision job.	Setting up a new vision job on page 57
4	Adjust the image settings of the vision camera.	Setting up the image on page 58
5	Calibrate the camera and the robot.	Calibration on page 60 Calibration theory on page 82
6	Add vision tools to locate and inspect parts in the image.	Adding vision tools on page 63
7	Make the vision data available to the RAPID program.	Output to RAPID on page 66
8	Setup the inputs and outputs of the vision camera, if any.	I/O handling on page 70
9	Prepare the RAPID program on the controller.	Preparing the RAPID program on page 72 RAPID reference information on page 103
10	Start production.	Starting production on page 77

For more useful tips about setting up a vision system, see [Best practise on page 85](#).

5.2 Preparations

Preparations

Experience shows that when starting with a clean system it is good to first load a RAPID program and make some initial preparations.

- Create tool data for all needed tools, and define the TCPs.
- Create work object data for all needed fixtures and define them.
- Etc.

The recommendation is to create a module and add the `MoveToDetectedObject` snippet. This way all the data that is edited during calibration, grip point training etc. will be in place.



Tip

Use the `MoveToDetectedObject` snippet as a base when creating a new vision program.

For more information on adding snippets, see [RAPID snippets in RobotStudio on page 72](#).

5.3 Setting up the camera

5.3.1 Basic procedures

Configuring the camera network and connecting to a camera


When all cameras are physically connected, each camera needs to be configured with an IP-address and a name.

The IP-address for the camera is by default assigned automatically by the controller, using DHCP, but it is also possible to set a static IP-address.

The camera name is used as a unique identifier for the camera in all parts of the system, for example RobotStudio, RAPID programs etc. This enables that the IP-address of the camera can be changed, for example if the camera is replaced, without having to modify the program.

The IRC5 controller browser in RobotStudio has a node called **Vision System**. This is used for configuring and connecting to cameras. A connection to a camera is established through the robot controller. The camera is connected as a FTP remote mounted disk.

Use this procedure to assign cameras to the controller:



	Action
1	Make sure that the network adapter of the PC is set to obtain an IP-address automatically.
2	Make sure that any installed firewalls on the PC allows communication with the camera, or are turned off.
3	Connect the PC to the service port of the IRC5 controller.
4	Start RobotStudio.
5	Connect to the controller and request write access over the controller.
6	Go to the Controller tab on the ribbon menu and start the Integrated Vision add-in.
7	Go to the Vision tab on the ribbon menu.
8	<p>Expand the Vision System node of the controller browser. Named cameras are displayed with their name, unnamed cameras are displayed with their MAC-id.</p> <div data-bbox="504 1525 564 1581">  </div> <p>Note</p> <p>If the camera does not show in the Vision System list, the IP-address of the camera can be set to a different subnet. For more information, see Connecting to a camera on a different subnet on page 49.</p>
9	<p>Right-click on the camera and select Connect. The image from the camera should now appear in a separate tab in the Image capture and configuration area. Use the camera image to identify the correct camera.</p>
10	If necessary, update the image by pressing the Acquire Image button.
11	Right-click on the camera and select Rename .

Continues on next page

5 Configuring Integrated Vision

5.3.1 Basic procedures

Continued

Action	
12	In the Rename dialog, enter the name of the camera in the RAPID Camera Name field.  Note It is advisable to also set the camera Host Name to the same name as the RAPID Camera Name .
13	Restart the controller and the camera.  Note It is important that both the controller and the camera is restarted.
14	The configured camera should now appear in the Vision System node of the controller browser.



Note

The names of the configured cameras are stored in the system parameters of the controller, topic **Communication** (`SIO.cfg`). The IP-settings are stored in the cameras.

Disconnecting a camera

To disconnect from a camera, click **Disconnect**, or simply close the corresponding tab in the image capture and configuration area.

Removing a camera

To remove a configured camera, the camera has to be deleted in the system parameters of the controller, topic **Communication** (`SIO.cfg`).

Use this procedure to remove a camera:

Action	
1	Use the Configuration Editor in RobotStudio, or an offline editor, to open the system parameters of the controller, topic Communication (<code>SIO.cfg</code>).
2	Delete the application protocol for the selected camera: COM_APP: -Name "MyCamera" -Type "CAMERA" -Trp "TCPIP1" -MAC "..."
3	Restart the controller.

For more information about using the **Configuration Editor** in RobotStudio, see *Operating manual - RobotStudio*.


For more information about system parameters, see *Technical reference manual - System parameters*.

5.3.2 Additional camera configuration

Changing the IP-address of a camera

The IP-address of the camera can be changed from the **Network Settings** dialog. It is advisable to set the camera **Host Name** to the same name as the **RAPID Camera Name** given in the **Configuration** dialog. The system works even if this is not the case, but some of the integrated *Cognex EasyBuilder®* dialogs in RobotStudio use the host name to identify the camera.

Use this procedure to change the network setting and the camera host name:

	Action
1	Select the camera in the Vision System node of the controller browser.
2	Click on the Connect button drop-down menu and select Network Settings .
3	Either set a fixed IP-address on the same subnet as the controller and the PC, or enable DHCP.
4	It is advisable to also set the camera Host Name to the same name as the RAPID Camera Name given in the Configuration dialog.
	 Note Do not change the Telnet Port or any other settings.
5	Click Ok .
6	Restart the camera.
7	Restart the controller.

Connecting to a camera on a different subnet

If the IP-address of the camera is not on the same subnet as the controller and the PC, the camera will not show in the **Vision System** node of the controller browser. Therefore it is not possible to set a new IP-address using the **Network Settings** dialog.

Use this procedure to connect to cameras on a different subnet:

	Action
1	Click on the Connect button drop-down menu and select Add Sensor .
2	Click the camera in the list and either set a fixed IP-address on the same subnet as the controller and the PC, or enable DHCP.
3	Click Apply .
4	Restart the camera.
5	Restart the controller.

Changing the date and time of a camera

Use this procedure to change the date and time of a camera:

	Action
1	Select the camera in the Vision System node of the controller browser.
2	Click on the Connect button drop-down menu and select Set Date and Time .

Continues on next page

5 Configuring Integrated Vision

5.3.2 Additional camera configuration

Continued

	Action
3	Change the date and time settings.
4	Click Ok .
5	Restart the camera.

Updating the camera firmware

When using a vision camera that was not delivered together with the Integrated Vision option, it may be necessary to update the camera firmware.

The latest firmware for the *Cognex In-Sight®* cameras is included in the RobotStudio installation.

Use this procedure to update the camera firmware:

	Action
1	Select or right-click the camera in the Vision System node of the controller browser.
2	Click on the Advanced drop-down menu and select Update Firmware . The camera model, the current version, and the new version of firmware is displayed.
3	Click Update .
4	Restart the camera.
5	Restart the controller.



Note

When running Integrated Vision it is recommended that the camera has firmware version 4.08.02 (023), or later.

Connecting to a camera on a virtual controller


If there is no real controller present, a virtual controller can be used instead.

The procedure to connect to a camera on a virtual controller is the same as connecting to a camera on a real controller.

However, before connecting to the camera, a network configuration file must be added to the HOME directory of the virtual controller system on the PC.

Use this procedure to create, add, and modify the network configuration file:

	Action
1	Create the <i>vc_network_definition.xml</i> file according to the separate instruction below.
2	Place the <i>vc_network_definition.xml</i> file in the HOME directory of the virtual controller system on the PC.
3	Check the IP-address of the PC network adapter to which the camera is connected.
4	Change the service port IP-address in the XML-file to the IP-address of the PC network adapter. <pre><PORT name="SERVICE"> <IPAddress>192.168.125.100</IPAddress></pre>

**Note**

Do not change any other settings.

Continues on next page

	Action
5	Restart the virtual controller.
6	Now it is possible to connect to the camera using the same procedure as for a real controller. See Configuring the camera network and connecting to a camera on page 47 .

Creating the `vc_network_definition.xml` file

Copy the following content into a text file using Notepad or any similar text editor. Name the file `vc_network_definition.xml`.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<VC_NETWORK_DEF>
  <PORT name="LAN">
    <IPAddress>10.46.77.126</IPAddress>
    <SubnetMask>255.255.252.0</SubnetMask>
  </PORT>
  <PORT name="SERVICE">
    <IPAddress>192.168.125.100</IPAddress>
    <SubnetMask>255.255.255.0</SubnetMask>
  </PORT>
</VC_NETWORK_DEF>
```

Limitations



Note

- The virtual controller does not have the same performance as a real controller when communicating with the camera.
- The virtual controller is provided "as is", therefore it should only be used for evaluation, training, support, testing, and similar.

Connecting to a camera emulator on a virtual controller

A camera emulator is available when creating a system with a virtual controller. The emulator can be used for evaluation, training, support, testing, and similar if there is no real camera available. In the emulator, it is always the currently loaded image that is evaluated. The emulator has no functionality to loop through images.

The procedure to connect to a camera emulator is the same as connecting to a real camera.

The emulator is started by default by the Integrated Vision add-in, and connects itself to one of the available network adapters on the PC. It is possible to change what type of camera that is emulated, see [Integrated Vision options on page 36](#).

Use this procedure to detect the camera emulator:



	Action
1	If possible, disable all network adapters on your PC except one. <div data-bbox="501 1863 563 1921" data-label="Image"> </div> <div data-bbox="585 1874 651 1906" data-label="Section-Header"> <h4>Note</h4> </div> <div data-bbox="493 1930 1450 1989" data-label="Text"> <p>If there are more network adapters available, the below steps may have to be repeated until the correct network adapter is found.</p> </div>

Continues on next page

5 Configuring Integrated Vision

5.3.2 Additional camera configuration

Continued

Action	
2	<p>Check the IP-address of the PC network adapter.</p> <p> Note</p> <p>If the network adapter on your PC is set to DHCP, the IP-address can change if the PC is restarted.</p>
3	<p>Change the SERVICE port IP-address in the <code>vc_network_definition.xml</code>-file to the IP-address of the PC network adapter.</p> <pre><PORT name="SERVICE"> <IPAddress>192.168.1.1</IPAddress></pre> <p> Note</p> <p>Do not change any other settings.</p>
4	<p>Restart the virtual controller.</p>
5	<p>The camera emulator is now detected in the Vision System node in the controller browser.</p>

Limitations



Note

- The camera emulator is only accessible when using a virtual controller.
- The camera emulator is provided "as is", therefore it should only be used for evaluation, training, support, testing, and similar.
- It is not possible to set a host name for the camera emulator.

5.3.3 Restricting user access

Introduction

RobotStudio communicates directly with the camera. For that reason the camera itself has got a user authentication method to provide the possibility to restrict the use of certain functionality.

RobotStudio and the controller both use a password to log on to the camera. By default the user name is "*admin*" and the password is blank "".

It is possible to edit the list of users on the camera and their associated privileges, and also to select which user name and password to be used the controller when it communicates with the camera. When updating the user list on the camera, make sure to also update the user profile used by the robot controller by clicking **Set controller user**.

User authentication system (UAS)

The user authentication method of the camera is separate from the user authentication system (UAS) of the robot controller.

Some actions specifically related to the controller requires write access, such as naming the camera. This is since the information is stored in the controller configuration. For all other actions, the user authentication of the camera applies. For more information about UAS, see *Operating manual - RobotStudio*.

Editing user access settings

The **User Access Settings** dialog maintains the access level and FTP read/write privileges for authorized users of *Cognex In-Sight®* vision systems and emulators. The **User Access Settings** dialog determine which users may log onto a particular camera, as well as the types of changes they can make to the active job. Each camera has its own user list, separate from every other camera on the network. If a user needs access to a particular camera, they must know a user name and password that already exists in that camera's user list.



Note

Every camera is pre-configured with three users: *admin*, *operator*, and *monitor*. These users are configured for *Full*, *Protected*, and *Locked* access levels, respectively.



Note

The maximum number of users that can be added to one camera is 32.

Setting the controller user

Use this procedure to select the user profile to be used by the controller when communicating with the camera.

	Action
1	Request write access over the controller.

Continues on next page

5 Configuring Integrated Vision



5.3.3 Restricting user access

Continued

	Action
2	Click on the Connect button drop-down menu and select Set Controller user .
3	Click on the Select User drop-down list and select a user.
4	Click OK .
5	Restart the controller.

Adding a new user

Use this procedure to add a new user.

	Action
1	Click on the Connect button drop-down menu and select User Access Settings .
2	Click Add .
3	Enter an alphanumeric string for the User Name field.  Note The length of the user name and password strings cannot exceed 30 characters, and both are case-sensitive.
4	Enter a password for the new user.
5	Click on the Access drop-down list and select an access level for the new user. For more information, see Access levels on page 54 .
6	Optionally, disable the Allow Run Mode/Program Mode checkbox to restrict users with <i>Protected</i> access from toggling the program mode/run mode state of the sensor. This checkbox is grayed out when the selected access level is <i>Full</i> or <i>Locked</i> .
7	Optionally, enable the Allow Run Mode Job Save checkbox to allow users with <i>Full</i> or <i>Protected</i> access to save jobs while in run mode. This checkbox is grayed out when the selected access level is <i>Locked</i> .  Note When checked, users with <i>Protected</i> access can save jobs in run mode even if the FTP write privilege is not enabled.
8	Specify the user's FTP, read, and write privileges. For more information, see FTP privileges on page 55 .
9	Click OK .

Access levels

The access level controls how much interaction is allowed for the current user to prevent inadvertent or unauthorized changes to the configuration. The selected access level will be in effect whenever anyone logs on to the RobotStudio vision add-in with the selected user name and password.

Three access levels are available:

Access level	Description
Full	This offers complete, unrestricted access to the camera. Any job created in RobotStudio can be loaded, modified or saved, and all menu selections are available. The default <i>admin</i> user account has full access.

Continues on next page

Access level	Description
Protected	<p>The user has limited access to the sensor.</p> <p>Protected mode allows you to access <i>Live Video</i> mode, toggle the <i>Run Mode/Program Mode</i> status of the camera (if permitted by your allow run mode/program mode privileges), and open or save jobs (if permitted by your FTP read/write privileges).</p> <p>In the standard view (not spreadsheet view), protected access allows you to edit tool parameters, but not to add or delete tools.</p> <p>In spreadsheet view, protected access always displays the custom view of the spreadsheet. A user in protected mode can edit the values of any graphics controls functions visible in the custom view, but cannot change the functions themselves. The default operator user account has protected access.</p> <p>For more information about the spreadsheet view and the custom view, please refer to <i>Cognex In-Sight® Explorer User's Guide</i> available on the Cognex web-site.</p>
Locked	<p>This offers the most restrictive access.</p> <p>You can only monitor the operation of the current camera. The default monitor user account has locked access.</p>

FTP privileges

FTP read privileges apply to opening job or image files from the camera, while FTP write privileges apply to saving jobs or images to the camera.

Additionally, these permissions are in effect when the user attempts to log on to the active camera from a remote FTP client on the network. More specifically, these privileges need to be enabled for the controller to be allowed to carry out file operations on the camera.

Privileges and access levels needed by the controller user

The user profile used by the robot needs FTP read/write privileges to allow file transfer between the camera and the controller.

The required access level is **Protected**, since this is the lowest level that allows changing between run mode and program mode. Make sure that **Allow Run Mode/Program Mode** is checked.



Note

The user profile and associated password used by the controller to log onto the camera is stored in plain text in the robot configuration. Keep the privileges for this user profile to the necessary minimum.

Editing an existing user

Use this procedure to edit an existing user.

	Action
1	Click on the Connect button drop-down menu and select User Access Settings .
2	Click Edit .
3	<p>Modify the user settings as necessary.</p> <p>For more information about the settings see Adding a new user on page 54.</p>

Continues on next page

5 Configuring Integrated Vision

5.3.3 Restricting user access

Continued



Note

- If the *admin* password is changed, you must log off then log back on to the sensor with the new password or errors may occur.
- The access level, allow run mode/program mode privileges, FTP privileges, and user name for *admin* cannot be modified.
- The **Show Custom View at Log On** checkbox affects only the spreadsheet view. The standard RobotStudio view does not support custom views.

Deleting an existing user

Use this procedure to delete an existing user.

Action	
1	Click on the Connect button drop-down menu and select User Access Settings .
2	Click Delete .
3	Confirm the deletion.



Note

The user *admin* cannot be deleted.

5.4 Setting up a new vision job

Connecting to the camera

Open RobotStudio and make sure that the camera is connected and tested according to the instructions in [Configuring the camera network and connecting to a camera on page 47](#).

Creating and saving a new job

All camera configurations and settings in Integrated Vision put together, is called a *job*.

The active job is stored in the working memory of the camera and is lost in the case of a power fail. The job shall be permanently stored as a job file (.job) either on the flash disk of the camera or on the flash disk of the robot controller.

To be able to load the job file from the RAPID vision instructions, the job must be stored on the camera flash disk, but it is strongly recommended to also create a backup elsewhere in case the camera gets damaged.

If the robot controller is used for job storage, then the job must first be copied from the controller disk to the camera flash disk before loading the job into the camera memory. This is done with standard RAPID file handling instructions.



Tip

For file transfer examples see the backup and restore snippets in RobotStudio.

Use this procedure to create a new job.

	Action
1	Make sure that the camera is in Program Mode .
2	Click New Job in the ribbon.
3	Click Yes to clear all data from the current job.
4	Click Save Job or Save Job As in the ribbon. The Save As dialogue will appear since the job has not been saved before.
5	Browse to the desired location, preferably on the camera flash disk.
6	Name the job and click Save . The name of the job will appear on the image tab in the image capture and configuration area.

Position the camera

If the camera is mounted on a moving part of the robot, now is the time to jog the camera into position and to store this position.

Before acquiring a new image, the camera always needs to be brought back to exactly the same position to maintain accuracy.

5 Configuring Integrated Vision

5.5 Setting up the image

5.5 Setting up the image

Introduction

The most commonly used setting is the exposure time. A longer time allows more light into the camera and makes the image brighter.

Adjusting the settings of a vision job is often an iterative process and the exposure time often needs to be modified once or more before the job is ready. Sometimes the setting that is required for obtaining a clear image for calibration may not be exactly the same as the optimal setting for detecting the product. If a subsequent step proves that the image settings are not optimal, do not hesitate to go back and modify them.



Tip

For more information see the [Best practise on page 85](#) chapter. Especially sections [Obtain good lighting on page 90](#) and [How to mount the camera on page 87](#).

Image trigger

The image trigger setting decides which event that triggers the camera to acquire an image. To be able to trigger the image from the Integrated Vision RAPID instructions, the trigger setting must be set to **Camera** or **External**.



Note

If the RAPID instruction `CamReqImage` is used, set the camera image trigger type to **Camera**. If the same RAPID instruction, `CamReqImage`, is used with the optional argument `\AwaitComplete`, then the camera image trigger type has to be set to **External**.

Image settings

Use this procedure to setup the image.

	Action
1	Make sure that the camera is in Program Mode .
2	Click Setup Image in the ribbon.
3	In the Context window, change the Trigger setting to Camera or External if using <code>CamReqImage</code> with optional argument <code>\AwaitComplete</code> .
4	If necessary, adjust the other settings to get the best image quality.
5	Save the job.

Continues on next page

For information about all parameters and settings, see the *Set Up Image / Adjusting the Edit Acquisition Settings* section in the online **Help** tab.



Note

The image settings are connected directly to the camera memory. All changes are applied immediately. Therefore there are no apply, save or undo buttons available. Save and backup the job frequently to avoid loss of data.

5 Configuring Integrated Vision

5.6 Calibration

5.6 Calibration

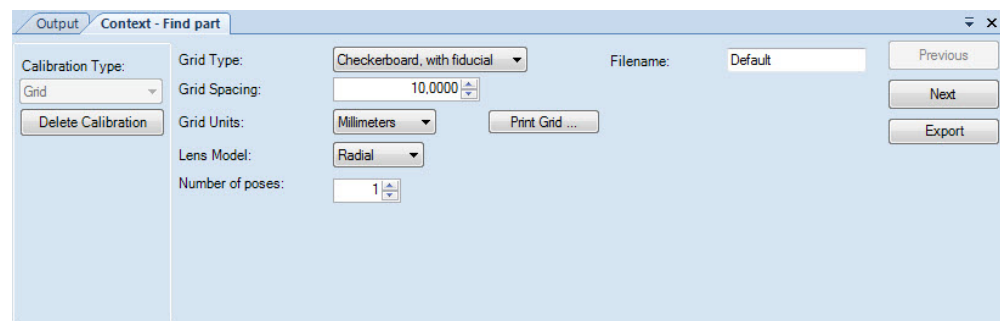
Introduction

An image consists of pixels, so in order to get the result in mm the camera needs to be calibrated. The **Calibrate** function is used to calibrate the image to real-world units.

The calibration consists of two basic steps. First the camera calibration which converts the image pixels to mm, and second the camera to robot calibration which relates the camera coordinates to a robot frame (work object).

The settings are displayed in the **Context** window. The different camera calibration types are described in the online **Help** tab, and in section [Calibration theory on page 82](#).

To get the best accuracy the recommended calibration type for Integrated Vision is to use a checkerboard with fiducial, see [Camera to robot calibration on page 62](#). The fiducial mark provides a clear reference which is later used for defining a corresponding work object.



xx1300001097



Note


It is outmost important to setup the right size of the calibration grid, otherwise the calibration will be wrong.

Camera calibration

Use this procedure to calibrate the camera.

	Action
1	Make sure that the camera is in Program Mode .
2	Click Calibrate in the ribbon.
3	In the Context window, change the Calibration Type to Grid .
4	From the Grid Type drop-down menu select one of the checkerboard calibration plates with fiducial, reference point.

Continues on next page

	Action
5	<p>If necessary, adjust the spacing, units, lens model, and number of poses settings.</p> <ul style="list-style-type: none"> • Use mm as the unit • The lens model depends on from where the most distortion is expected. Either because the camera is viewing from an angle (projection), or that the lens itself is distorting the image (radial). • Number of poses allows to use more than one image of the calibration plate to calibrate the camera in case the plate does not cover the full field of view.
6	<p>Click Print Grid to print the calibration plate. The printed image must have a high contrast and the paper must not be reflective (high gloss). Verify with a ruler that the squares are proportional.</p> <p> Tip</p> <p>There are preprinted calibration plates located under the RobotStudio installation folder. These may be used if it is not possible to enter the calibration wizard when printing:</p> <p><i>C:\...ABB Industrial IT\Robotics IT\RobotStudio x.xx\Bin\Addins\IntegratedVision\PDF Grids</i></p>
7	<p>Place the calibration plate on a fixed position in the center of the camera image, at the same height as the objects that the camera shall identify. The calibration paper must be completely flat, adequately illuminated, and free from gloss and shadows. Rotate the calibration plate so that the X and Y arrows corresponds to the desired direction of the camera work object.</p>
8	<p>In the Context window, click Next. The calibration is now being calculated by the camera, and the number of found feature points are displayed.</p>
9	<p>Click Next.</p>
10	<p>Click Calibrate to apply the calibration.</p>
11	<p>Click Finish to complete the calibration.</p>
12	<p>Save the job.</p>
13	<p>Do not change the position of the calibration plate until the work object has been defined.</p>

For more information about parameters and settings, see the *Set Up Image / Calibrating the Image to Real World Units* section in the online **Help** tab.

Continues on next page

5 Configuring Integrated Vision

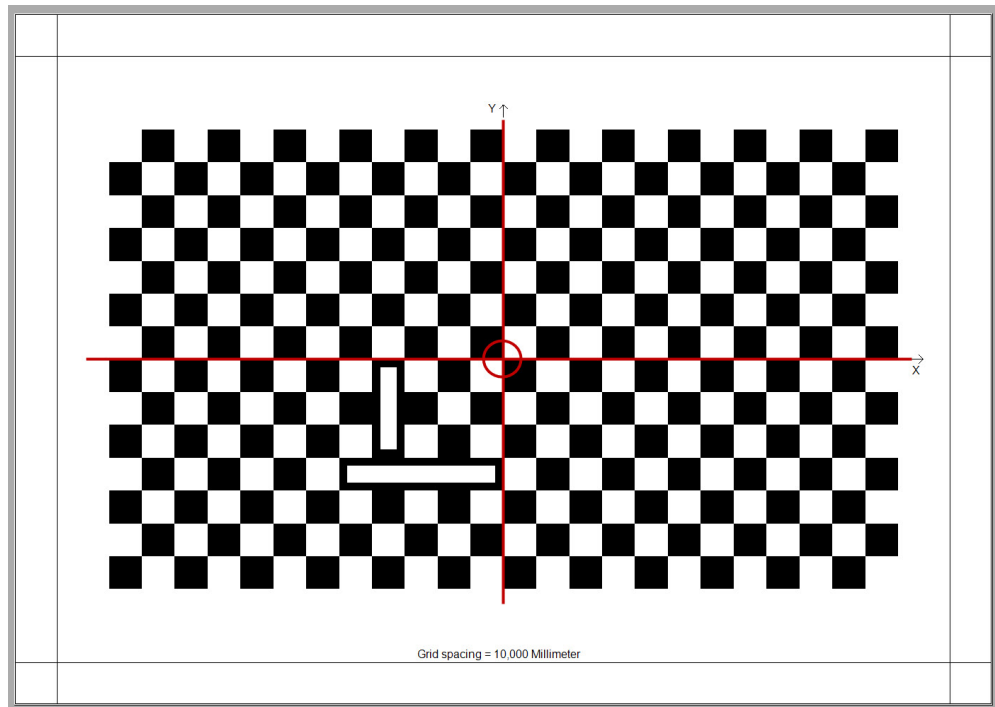
5.6 Calibration

Continued

Camera to robot calibration

The camera is calibrated to the robot by defining a work object with the same origin of coordinates as the calibration plate.

On a checkerboard calibration plate with fiducial, the origin of coordinates is located at the intersection of the extended X- and Y-arrows as seen in the picture below.



xx1200000994



Note

Before defining the work object, make sure to check where the origin is located on the currently used calibration plate.

Use this procedure to calibrate the robot.

	Action
1	Create a pointing tool and define the tool TCP using an accurate method.
2	Create one work object for each camera.
3	Activate the pointing tool and define the user frame of the camera work object along the corresponding x- and y-axes of the calibration plate. Leave the object frame empty.
4	Test that the calibration is correct by jogging the robot in the work object.
5	The calibration plate can now be removed.

For information on how to create and define tools and work objects, see chapter *Programming and testing in Operating manual - IRC5 with FlexPendant*.

5.7 Adding vision tools

Introduction

The location tools are used to define a feature in the image that provides positional data. The location tools creates a reference point, which is used to locate a part in the image quickly and reliably, even if the part being inspected rotates or appears in different locations in the image.

The inspection tools are used to examine the parts located by the location tool. Depending on the requirements of the current application different tools are available for checking presence/absence, measurement, counting, geometry, etc. The settings are displayed in the **Context** window.

Adding a location tool

Use this procedure to add a location tool.

	Action
1	Load or acquire a new image. Make sure that the part to be located is present within the areas in the image where the part may appear.
2	Click Add Part Location Tool , then click the desired tool from the drop-down menu.
3	Follow the tool specific instructions in the context window.
4	If necessary, adjust the settings to get the best performance.
5	Save the job.

For information about all parameters and settings, see the *Locate Part* section in the online **Help** tab.

Most commonly used location tools

Part location tool	Description
PatMax® Pattern PatMax® Patterns (1-10)	Locates a single pattern, or up to ten patterns using the PatMax® algorithms, and reports the x- and y-coordinates, angle and score of the found patterns.
Blob Blobs (1-10)	Locates a single group, or up to ten groups, of dark or light-colored connected pixels, called blobs, and reports the x- and y-coordinates of the found blob. This tool is commonly used as a fixture to orient other vision tools.

Most commonly used location tool settings

Setting	Description
Number To Find	Defines the number of instances to detect. The default value is often 1 and must be increased to detect multiple instances.
Rotation Tolerance	Defines how far the the found pattern can be rotated from the trained pattern and still be recognized as a valid pattern. The default value of $\pm 10^\circ$ to 15° is often too small and needs to be increased.

Continues on next page

5 Configuring Integrated Vision

5.7 Adding vision tools

Continued

Adding an inspection tool

Use this procedure to add an inspection tool.

	Action
1	Load or acquire a new image. Make sure that the pattern to be located is present within the areas in the image where the pattern may appear.
2	Click Add Part Inspection Tool , then click the desired tool from the drop-down menu.
3	Follow the tool specific instructions in the context window.
4	If necessary, adjust the settings to get the best performance.
5	Save the job.

For information about all parameters and settings, see the *Inspect Part* section in the online **Help** tab.

Most commonly used inspection tools

Tool group and part inspection tool	Description
Presence/Absence Tools <ul style="list-style-type: none">• PatMax® Pattern	Determines whether or not a trained pattern is present or absent, using the PatMax® algorithm. Reports a pass if the pattern is present and within limits, or a fail if it is outside of the limits.
Presence/Absence Tools <ul style="list-style-type: none">• Blob Blobs (1-10)	Determines whether or not a group of dark or light-colored connected pixels, called blobs, are present or absent. Reports a pass if the blob feature is present and within limits, or a fail if it is outside of the limits.
Measurement Tools <ul style="list-style-type: none">• Distance	Measures the distance between any two features such as edges, circles, patterns, and/or blobs. Reports a pass and the distance in millimeters or pixels (unless the image is calibrated), or a fail if the reported distance is outside of the limits.
Identification Tools <ul style="list-style-type: none">• PatMax® Patterns (1-10)	Determines from a library of trained patterns which pattern best matches the pattern in the image, using the PatMax® algorithm. Reports the name of the found pattern and it's score compared to the trained model and results in a pass if the found pattern falls within limits, or a fail if it is outside of the limits or the pattern wasn't found.

Pass and fail of vision tools

For each location and inspection tool there is a check box that defines whether or not the tool's pass/failure status should be included in the job's overall pass/fail status. By default, it is checked and will be included in the job's overall pass/fail status.

Uncheck the check box to keep the tool's pass/failure status separate from the other tool's in the job.

This control should be unchecked if a tool is expected to fail. For example, if two identification tools were being used to determine whether or not the part was a right-hand or a left-hand part. One of the tools would be expected to fail every time. If the checkbox was checked, this situation would result in the job failing every time regardless of which side of the part was identified.


Continues on next page

Links between vision tools

The output results of one vision tool can be used as input parameters for another tool. Then the tools are linked.

Tool properties are divided into input and output properties, and links can only be established from output properties to input properties. Output properties can be linked to multiple separate input properties, while an input property only accepts one output property linked into it. The most common example is when a location tool is used as a fixture for the inspection tool.

The links are displayed visually as graphical arrows in the **Results** tab. The graphical arrows cannot be edited. To remove or change a link, the parameters has to be removed from the vision tool.

View	Description
Camera target inputs for selected item type	<ul style="list-style-type: none"> Component: The components corresponds to the arguments of the resulting <code>cameratarget</code>. Data type: The data type that corresponds to the arguments of the resulting <code>cameratarget</code>. Group: The group is the source of data, typically the configured inspection or location tools in the job. It is also possible to get data from the job itself, the input signals, the camera, or a constant. Each data group is an array that consists of many results (arguments). Result: The results (arguments) of the group data. Typically x, y, z, and angle coordinates for the robot. A number of arguments are available depending on the data. These arguments are described in the online Help tab for each individual tool under the section "... Tool Inputs/Outputs Properties" Value: The value of the Result. Typically a numerical value or a text string.
Resulting camera target	<p>Displays the values that will be transferred to the resulting RAPID <code>cameratarget</code>.</p> <div>  Note </div> <p>The camera can only identify the rotation of the located part, z-angle. This value is converted to quaternion values in RAPID since the corresponding RAPID component is <code>pose</code>.</p>

Mapping data

Use this procedure to make the vision data available to the RAPID program.

	Action
1	Click Output to RAPID .
2	Click Add in the Item types view to create a new item type.
3	Click Rename to give the item type an explaining name. For example the name of the part that the location- or inspection tool has identified (e.g. "Nut", "Screw", "Bolt", etc). The name is copied to the <code>type</code> argument of the resulting <code>cameratarget</code> .
4	Add vision data to the RAPID <code>cameratarget</code> by selecting them from the Group followed by the Result drop-down menus.
5	Check the result in the Resulting camera target view.
6	Save the job.
7	Run the RAPID instruction <code>CamSetRunMode</code> to update the controller on the current output to RAPID configuration.

For information about the different location tools and their parameters and settings, see the *Locate Part* section in the online **Help** tab.

Continues on next page

5 Configuring Integrated Vision

5.8 Output to RAPID

Continued

For information about the different inspection tools and their parameters and settings, see the *Inspect Part* section in the online **Help** tab.



Note

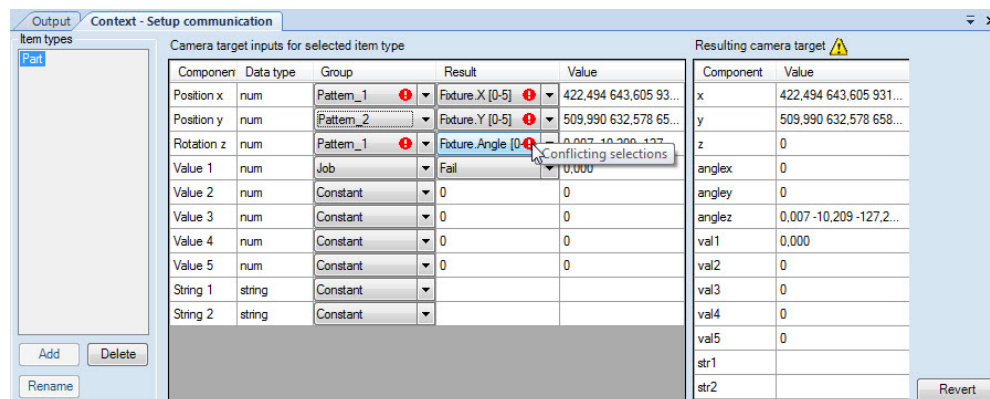
It is not recommended to change the RobotStudio language after the configuration of an Integrated Vision system has started.

For more information, see [Changing the language on page 38](#).

Limitations

There are some restrictions to think about when completing the output configuration. The number of values mapped to each component has to match. For example if a tool that finds 10 values, like **PatMax® Patterns (1-10)**, is mapped to the X-coordinate, it is not advisable to map tool output of another size to the other components. This would result in unpredictable behavior

Please observe warnings and cautions given by the Integrated Vision add-in:



xx1300001100


Furthermore, only values that are expected to change with each image acquisition are to be included in the result output. Other parameter values needed during runtime should preferably be fetched using the RAPID instruction `CamGetParameter`.

Most commonly used vision data

Location tool data

The following data results are typically used when locating parts:

Symbolic name	Description
Fixture.X	The x-coordinate of the located part.
Fixture.Y	The y-coordinate of the located part.
Fixture.Angle	The rotation of the selected part.

**Note**

The camera can only identify the rotation of the located part, z-angle. This value is converted to quaternion values in RAPID since the corresponding RAPID component is `pose`.

Continues on next page

Symbolic name	Description
Fixture.Score	A percentage value that references how well the found pattern resembles the trained pattern.
Pass	Set to 1 if the part is located, set to 0 if the part is not located.
Fail	Set to 0 if the part is located, set to 1 if the part is not located.

Inspection tool data

The following data results are typically used when inspecting parts:

Symbolic name	Description
Distance	The measured distance between two points.
Result	Returns the string <i>Present</i> if the trained object is found, or <i>Not Present</i> if the trained object is not found.
String	Returns the value of an identified bar-code number as a string.

Job data

The following data results from the job are typically used:

Symbolic name	Description
Pass	Set to 1 if all vision tools that are set to Include In Job Pass are pass, set to 0 if one or more of the included vision tools are fail.
Fail	Set to 0 if all vision tools that are set to Include In Job Pass are pass, set to 1 if one or more of the included vision tools are fail.

5 Configuring Integrated Vision

5.9 I/O handling

5.9 I/O handling

Introduction

I/O is mostly used if the application includes external lighting that needs to be controlled by the camera. If there are no I/O signals connected to the camera, no configuration needs to be done.



Note

The **Inputs** and **Outputs** functions are only used to define the inputs and outputs of the camera, not the inputs and outputs of the robot controller.

General

The **Inputs** function is used to define the settings of the discrete inputs of installed *Cognex* camera I/O modules by customizing the name of the input line, setting the signal type, selecting the edge transition of the signal and forcing inputs to test the application.

The **Outputs** function is used to define the settings of the discrete outputs of installed *Cognex* camera I/O modules by customizing the name of the output line, setting the signal type, selecting tool results and forcing outputs to test the application. Some cameras have integrated outputs and LEDs. Those output lines are predefined and cannot be used for other purpose.

The settings are displayed in the **Context** window.

Configuring inputs

Use this procedure to configure the inputs of the vision camera, or any installed camera I/O modules.

	Action
1	Click Inputs . The available I/O lines are displayed in the Context window.
2	Use the Signal Type , Edge Type , and the Force Input drop-down menus to configure the desired behavior of the input.
3	To test the inputs, the camera must be in Run Mode .
4	Save the job.

For information about parameters and settings, see the *Inputs* section in the online Help tab.

Configuring outputs

Use this procedure to configure the outputs of the vision camera, or any installed camera I/O modules.

	Action
1	Click Outputs . The available I/O lines are displayed in the Context window.
2	Use the Signal Type , Job Result , and the Force Output drop-down menus to configure the desired behavior of the output.

Continues on next page

	Action
3	To test the outputs, the camera must be in Run Mode .
4	Save the job.

For information about parameters and settings, see the *Outputs* section in the online Help tab.

5 Configuring Integrated Vision

5.10.1 RAPID snippets in RobotStudio

5.10 Preparing the RAPID program

5.10.1 RAPID snippets in RobotStudio

Introduction

Snippets are pieces of predefined RAPID code which can be inserted into the RAPID program.

A number of snippets have been created to facilitate the programming and commissioning of a vision system.



Tip

Use the `MoveToDetectedObject` snippet as a base when creating a new vision program.

Adding a snippet

The RAPID snippets are located in **Snippets** drop-down menu in the RobotStudio **RAPID** tab.

Use this procedure to add a snippet.

	Action
1	Request write access over the controller.
2	Expand the RAPID node of the controller browser and navigate to the Program Modules .
3	Create a new program module or open an existing module. Place the cursor on the desired insertion point for the snippet.
4	Click the Snippets drop-down menu, and select Integrated Vision to view the list of included snippets.
5	Click a snippet to include it in the program module.

For more information on how to program the controller using the RobotStudio **RAPID** tab, see *Operating manual - RobotStudio*.

5.10.2 Basic programming example

Introduction

This section describes how to write a basic vision guided robot program. The main purpose is to give an overview of what instructions need to be called. Some more advanced examples are also available.

For a detailed description of the vision specific RAPID instructions, functions, and data types see [RAPID reference information on page 103](#).



Note

Before running a RAPID program, all previous configuration steps in this chapter has to be completed. See [Configuring Integrated Vision on page 45](#).

Creating a basic RAPID program

The following example shows the basic steps to run a vision guided robot program. Error handling and other enhancements have been left out to provide a better overview.

The purpose of the following RAPID program is to move the robot to a position where the robot can pickup a part detected by the vision camera. It is based on the snippet `MoveToDetectedObject` found in RobotStudio. For more information, see [RAPID snippets in RobotStudio on page 72](#).

```

1      ...
2      PERS wobjdata mywobj := ... ;
3      PERS tooldata mytool := ... ;
4      PERS robtargt myrobtargt := ... ;
5      CONST string myjob := "myjob.job";
6      VAR cameratarget mycameratarget;
7      ...
8      PROC MoveToDetectedObject()
9          CamSetProgramMode mycamera;
10         CamLoadJob mycamera, myjob;
11         CamSetRunMode mycamera;
12         CamReqImage mycamera;
13         CamGetResult mycamera, mycameratarget;
14         mywobj.oframe := mycameratarget.cframe;
15         MoveL myrobtargt, v100, fine, mytool \WObj:=mywobj;
16     ENDPROC
17     ...

```

Row	Comment
2 - 6	Declaration of data.
9 - 10	Set the camera to program mode and load the job.
11 - 12	Set the camera to run mode and acquire an image.
13	Get the vision result and store it in a camera target.

Continues on next page

5 Configuring Integrated Vision

5.10.2 Basic programming example

Continued

Row	Comment
14	Copy the vision coordinates from the camera target to the object frame of the work object.
15	Move the robot to the pickup position.
-	(Pickup the part.)

For information on how to create a module, create a routine, and add RAPID-instructions, see chapter *Programming and testing in Operating manual - IRC5 with FlexPendant*.

Training the part grip location

Gripping a part is often not the same as moving the TCP to the target reported by the camera. Often this position must first be offset and rotated by some value to accommodate a good grip.

The easiest way to do this is by jogging the robot to the specified position and then modify the position.

	Action
1	Make sure that only one part is visible to the camera.
2	Run the program in the previous example down to the <code>MoveL</code> instruction and stop the execution. At this point the object frame of <code>mywobj</code> has been modified and the correct tool, <code>mytool</code> , is activated.
3	Verify that the part was successfully located by the vision camera.
4	Jog the robot to a good gripping position.
5	Mark the position <code>myrobtarg</code> and tap Modify Position .
6	Run the program from the top and make sure that the part is gripped according to the taught position.
7	Move the part and run the program from the top again.

5.10.3 Advanced programming examples

Introduction

The following examples show some of the more advanced features that can be used when running a vision guided robot program. The examples show the use of error handling, scene IDs, and other ways of increasing production and saving cycle time.

Data declaration

```
CONST string myjob := "myjob.job";
PERS robtargt myrobtargt :=
  [[100,200,300],[1,0,0,0],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
VAR cameratargt mycameratargt;
...
PROC integratedvisionadvanced()
  VAR num mysceneid:=0;
  VAR num myexposuretime:=10;
  VAR bool targetsavailable := TRUE;
  ...
  ! Insert the code here
  ...
ENDPROC
```

Asynchronous loading

Loading a job asynchronously while moving the robot and then acquire an image as soon as the job finishes loading. This can optimize cycle time by having the robot performing other tasks while loading

```
CamSetProgramMode mycamera;
CamStartLoadJob mycamera, myjob;
!MoveJ myrobtargt, v100, fine, toolvision \WObj:=wobjvision;
CamWaitLoadJob mycamera;
CamSetRunMode mycamera;
CamReqImage mycamera;
```

Using the SceneId argument

Get an image and get the result as soon as it is available. Using the SceneId argument makes it possible to quickly notify the user if no target were found in the latest image.

```
CamReqImage mycamera \SceneId:=mysceneid;
CamGetResult mycamera, mycameratargt \SceneId:=mysceneid;
```

Changing parameters

Add 5 ms to the current exposure time

```
myexposuretime := CamGetExposure (mycamera \ExposureTime);
myexposuretime := myexposuretime + 5;
CamSetExposure mycamera \ExposureTime:=myexposuretime;
```

Continues on next page

5 Configuring Integrated Vision

5.10.3 Advanced programming examples

Continued

Disable a vision tool

Disable a specific vision tool in the specified job. In order to avoid time consuming job loading, it may be desirable to configure various vision tools in a single job. By enabling/disabling the tools that are not currently used, the processing time can be minimized.

```
CamSetParameter mycamera, "Pattern_1.Tool_Enabled" \BoolVal:=FALSE;
```

Finding multiple parts

Find multiple parts in a single image and use the targets one by one. When there are no more targets left, the error `ERR_CAM_NO_MORE_DATA` is generated, see [ERROR handler on page 76](#).

```
CamReqImage mycamera \SceneId:=mysceneid;
WHILE targetsavailable DO
  CamGetResult mycamera, mycameratarget \SceneId:=mysceneid;
  TPWrite "Current camera target is: "
    \Pos:=mycameratarget.cframe.trans;
ENDWHILE
```

Check if the job is already loaded

The job should only be loaded into the camera memory if it is not already loaded, in order to save cycle time.

```
IF CamGetLoadedJob(mycamera) <> myjob THEN
  CamLoadJob mycamera, myjob;
ENDIF
```

Sort parts by name

Sort parts depending on the part name property.

```
CamGetResult mycamera, mycameratarget;
IF mycameratarget.name = "wrench" THEN
  !Do something with the wrench
ELSEIF mycameratarget.name = "screwdriver" THEN
  !Do something with the screwdriver
ENDIF
```

ERROR handler

```
ERROR
IF ERRNO = ERR_CAM_NO_MORE_DATA THEN
  TPWrite "There are no more targets originating from image with
    scene id "\Num:=mysceneid;
  targetsavailable:=FALSE;
  TRYNEXT;
ENDIF
```

5.11 Starting production

Setting the camera to Run Mode

To start and run production the camera must be in run mode. The camera can manually be set to **Run Mode** from RobotStudio. This is mainly for testing purposes. The recommended procedure is to set the run mode from RAPID using the instruction `CamSetRunMode`.

Starting the robot program

Start the program from the **Production Window** on the FlexPendant.

This page is intentionally left blank

6 Reference information

6.1 Relationships between coordinate systems

Introduction

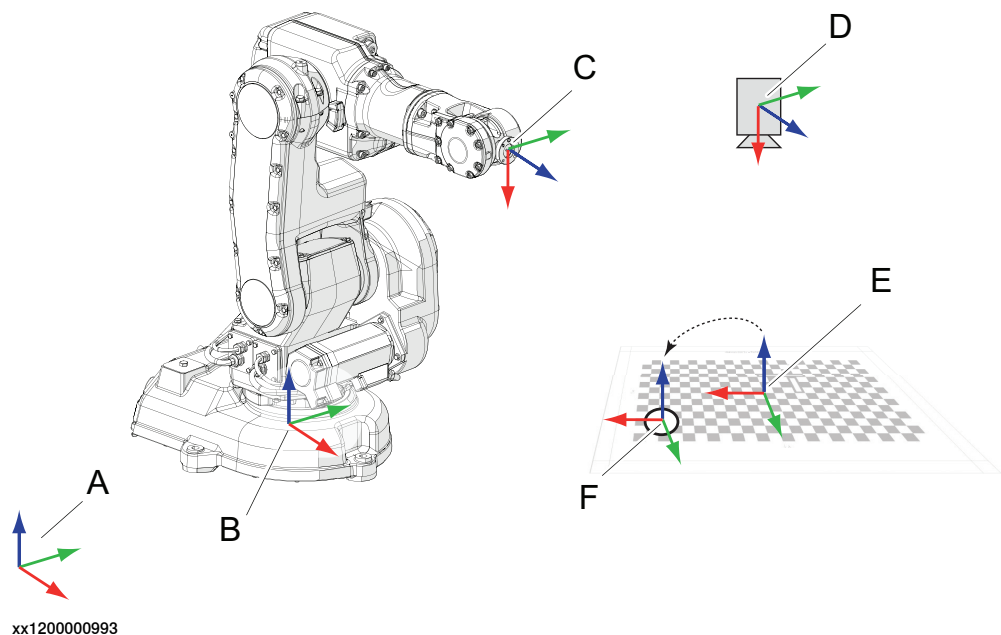
The robot controller has a number of built-in coordinate systems that all relate to each other in a predefined chain, WORLD, BASE, tool, work object, etc.


The vision camera also has coordinate systems to define the origin of the image and to define the distance in mm to the located parts within the image. Integrated vision provides means to synchronize the coordinate systems of the camera with the coordinate systems of the robot controller.

For more information about coordinate systems, and how to use them, see *Operating manual - IRC5 with FlexPendant*.

Coordinate systems in general

The image shows the most common coordinate systems of the robot controller. It is optional to define the WORLD coordinate system, but all other coordinate systems need to be known by the robot controller.



A	WORLD coordinate system.
B	BASE coordinate system.
C	Tool coordinate system (<code>tool0</code>).
D	Fixed camera position in space. The camera position is unknown to the robot controller unless the camera is held by the robot.
<div>  Note </div> <p>If the camera is held by the robot, then the robot must go to the same position (<code>robtarg</code>) every time an image is acquired.</p>	

Continues on next page

6 Reference information

6.1 Relationships between coordinate systems

Continued

E	Work object - user frame (<code>wobj.uframe</code>). Coincides with the camera frame when the camera to robot calibration has been performed.
F	Work object - object frame (<code>wobj.oframe</code>). It is recommended to use this coordinate system for the located part.

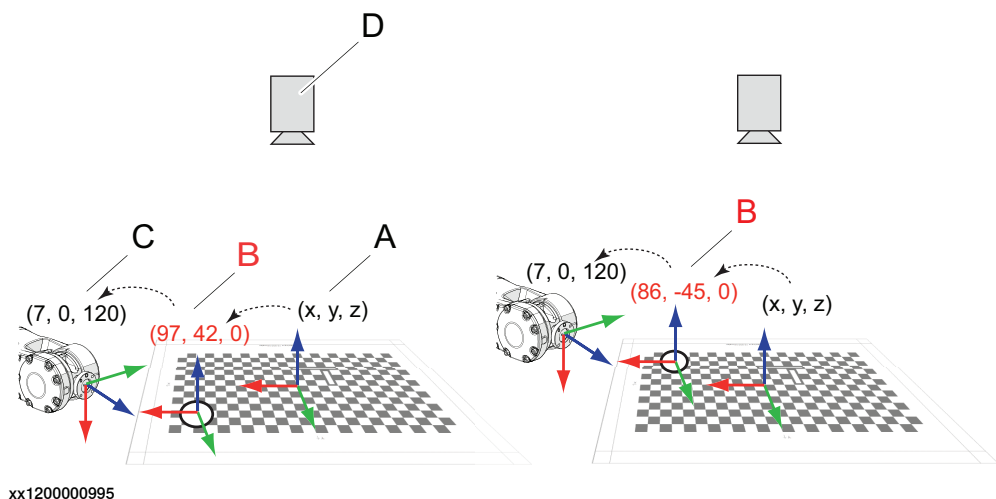
Calibrated camera frame (work object)

The following illustration and example shows the synchronization of the camera coordinate system with the coordinate systems of the robot controller.



Note

After the camera to robot calibration, the grid is no longer needed and can be removed.



A	Work object - user frame (<code>wobj.uframe</code>). Coincides with the camera frame when the camera to robot calibration has been performed.
B	Work object - object frame (<code>wobj.oframe</code>). This coordinate system is used for the located part.
C	The gripping position of the robot (<code>robtarg</code>). (<code>tool0</code> is used in the illustration) The gripping position is related to the camera work object.
D	Fixed camera position in space (not held by the robot).

Example

The illustration to the left shows the basic setup when a part is located by the vision camera. The orientation, angle, has been left out to provide a better overview.

- The camera is located at a fixed position in space.
- The camera is calibrated by using a 10 mm calibration grid so that the camera can convert the image pixels to mm in x- and y-coordinates.
The origin of coordinates is located at the intersection of the fiducial.
- The calibration grid is calibrated to the robot controller by the *work object - user frame*.

Continues on next page

The origin of the work object is also placed at the intersection of the fiducial.

- A part is located by the vision camera and the coordinates are sent to the robot controller.

The grid spacing is 10 mm, giving the x-, y-, and z-coordinates (97, 42, 0).

- The coordinates of the part are written into the *work object - object frame*.
- The robot moves to pickup the part.

The picking position has been modified in relation to the *work object - object frame*. For example 120 mm above the part and slightly offset in x by 7 mm, giving the coordinates (7, 0, 120).

- This means that the picking position can be the same no matter where the part is located.

In the illustration to the right, a new part has been located by the vision camera.

- The camera position is the same
- The calibration of the camera is the same
- The picking position is the same in relation to the part.
- The part has a new position, (86, -45, 0).

The coordinates of the part are written into the *Work object - object frame*.

- The robot can now pickup the part.

6 Reference information

6.2 Calibration theory

6.2 Calibration theory

Introduction

The goal of most VGR applications is to provide positioning data from the camera to guide the robot. This requires the vision system to provide targets in a coordinate system shared with the robot. Creating a vision calibration can be divided into two steps, camera calibration and camera to robot calibration.

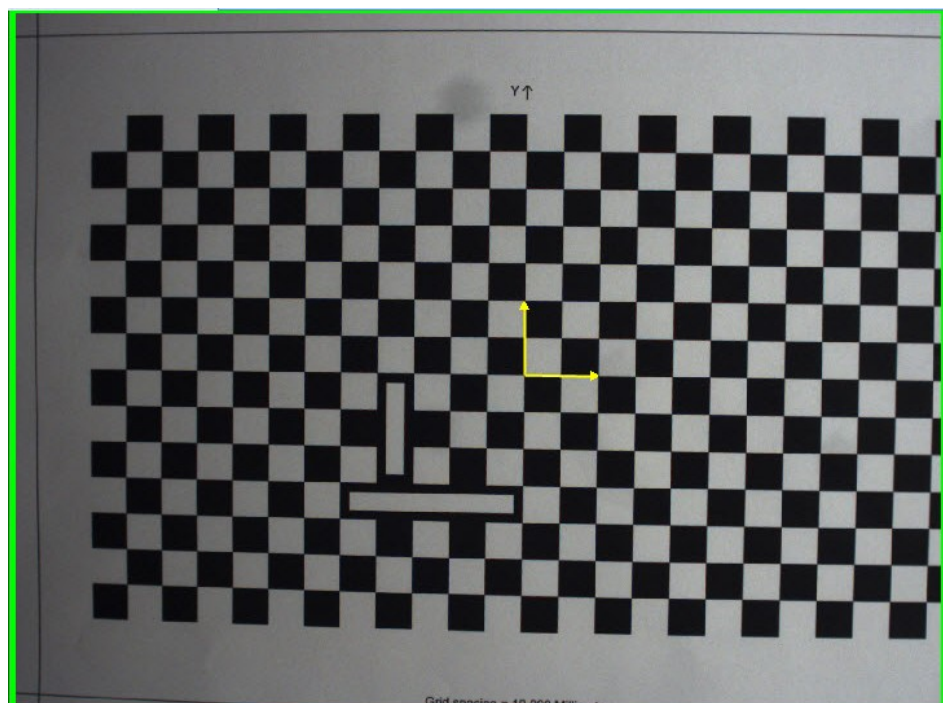
Camera calibration

The purpose of the camera calibration is to compute the transformation used for converting pixel coordinates in the image to physical coordinates in space (the relation between the camera frame and the checkerboard frame).

The Integrated Vision system works in 2D, meaning that all imaged parts must be located in a single calibrated plane or the result will not be accurate.

The Integrated Vision system provides several methods for calibrating the camera, but the most accurate and convenient method is to use a checkerboard plate. It is a pattern of known dimensions that also includes an encoded coordinate system. The relation between the camera and the checkerboard is fixed meaning that the camera always has to image the scene from the same point where it was calibrated.

In case the camera is mounted on the robot it has to move to the calibration pose before taking a photo during production. Once the camera has been calibrated any output from the image processing is expressed in millimeter in relation to the checkerboard origin.



xx1200000996

For information on how to perform a camera calibration, see [Camera calibration on page 60](#).

Continues on next page

Camera calibration methods

The table below describes the available camera calibration methods in Integrated Vision.

The **X/Y Scale**, **Edge to Edge**, **X/Y Edge-to-Edge**, and **Circle** calibration options are useful in applications where the real-world measurements of the part or object are known.

Only the **Grid** calibration provides a reference mark that can be used for teaching a work object. The other calibration methods also create a pixel-to-mm relation, but they are less suitable for part location applications.



Note

The scale calibration options do not account for lens distortion, therefore, for the greatest degree of accuracy, make sure that the vision system is mounted perpendicular to the inspected surface. Otherwise, consider using **Grid** calibration to remove the distortion and more accurately calibrate the image.

Group	Description
X/Y Scale	Used when the X and Y scale values are known for calculating a simple calibration.
Edge to Edge	Used when distance between two edges are known for calculating a simple calibration.
X/Y Edge-to-Edge	Used when distance between two edge pairs are known. One in the horizontal direction, and another in the vertical direction. Combining the horizontal measurement and dimension with the vertical measurement and dimension creates two independent scales.
Circle	Used when the diameter of a circle is known.
Grid	The Grid calibration option creates a "map" of the image area by acquiring an image of a grid pattern of either dots or checkerboard squares. This option then relates the found features of the grid (the dots or intersections of squares) to the user-specified grid spacing of the grid pattern. This process creates a highly accurate computation that can then be used to transform distorted features into their correct shape.
Import	The Import calibration type is used to load a grid calibration file that has been exported during the configuration step. The calibration file (.cxd) must be located on the flash disk of the camera when loading the job that utilizes the calibration.

Camera to robot calibration

Once the camera has been calibrated any image processing output is expressed in the calibrated camera frame. To create a common frame for the robot and the camera, such that the robot can correctly interpret the position of the vision targets, the robot must be taught the calibrated camera frame.

The way to do this is to teach a work object that coincides with the calibrated camera frame. In other words, if a checkerboard pattern is used the work object is taught at the very same position (and orientation) as the checkerboard frame. A work object consists of two different frames – a user frame and an object frame. In this calibration the user frame is modified and the object frame is left untouched.

Continues on next page

6 Reference information

6.2 Calibration theory

Continued

For information on how to perform a camera to robot calibration, see [Camera to robot calibration on page 62](#).

6.3 Best practise

6.3.1 Evaluate performance before adopting a solution

Introduction

When deciding to deploy a vision solution it is necessary to evaluate if the expected result can be achieved.

The safest way to ensure that required results can be achieved is to perform a test, and the closer the lab setup is to the intended installation the better the result.

It is of course not possible to make a full setup, but it is important to consider critical factors such as:

- Required field of view
- Required accuracy
- Robustness

Identify the expectations

Before moving into a design phase it is necessary to identify the expectations so that appropriate actions can be acquired to maximize performance.

The concept of accuracy is relatively easy to understand, and in many cases there is a good perception of what is required of the system. However, the overall accuracy of a robot-guided vision system, and even the vision system alone, is quite difficult to foresee.

Even though the individual vision tools such as a pattern recognition tool or caliper may have an extremely good theoretical accuracy, the true accuracy is highly dependent on external factors such as the product to be identified and the light setting.

Evaluate the vision accuracy

The general recommendation is to evaluate vision accuracy using the part or product that is actually going to be processed on the finished line. Furthermore, the light setup is critical to both accuracy and robustness.

Lighting is what many times makes or breaks final result, and important to note, something that may be difficult to change once the line has been designed and is being deployed.

In conclusion, when deploying vision systems it pays off to set up a vision trial in a lab environment.

Installation checklist

As good practice the following requirements shall be identified, quantified, and verified:

Requirement	Description
Samples	Collect good and bad samples of the actual customer product to be used for evaluation.

Continues on next page

6 Reference information

6.3.1 Evaluate performance before adopting a solution

Continued

Requirement	Description
Accuracy	What accuracy is required? The overall number combines robot accuracy, influence by part variation, lighting etc.
Tolerance	Can the part vary in size? Uniformly or irregularly?
Cycle time	The vision system requires processing time. Depending on the application this may or may not affect the cycle time.
Part positioning	Make sure to know the perspective from which the camera will observe the object. A simple thing like looking at the object from the side may affect the result.
Variations in the process	Apart from the verified variables, can something else change?
Lighting needs	Lighting is extremely important. Shield out ambient light and applying light that brings out the desired features of the part. Experimentation is the only reliable method.
Physical space constraints	Taking all factors into consideration such as field of view, lighting solution, and point of view. Does everything fit together?

6.3.2 How to mount the camera

Introduction

Depending on the application requirements and physical constraints, the camera may be mounted in different ways.

Generally it can be said that mounting the camera on a fixed structure is more efficient unless requirements are such that the camera must be carried by the robot.

Stationary camera

A stationary camera generally provides faster cycle times since the robot does not have to stop on its path to acquire an image. Setup and calibration is generally easier with fixed cameras since the point from which the image is acquired is fixed. When mounting the camera on a fixed structure it is important that the camera is not subject to vibrations which can cause motion blur.

Robot held camera

If it is not possible to mount the camera at a fixed location, the camera can also be mounted on a moveable part of the manipulator. In that case the camera would generally be mounted on the robot tool to avoid occlusion. Each application is different, and tool designs as well as dress packs for the cables differ from case to case.

When placing a camera on a moving position it is the user's responsibility to assure that the camera is not subjected to mechanical forces greater than what is specified in the camera specification. The cables are of a flexible type, but wear depends greatly on both the cable routing and the programmed robot path.



CAUTION

When using a robot held, or by other means moving camera, it is important to have a good cable routing.

When routing the cables caution has to be taken to avoid mechanical stress on the connectors, assure sufficient bend radius for the cables, and minimize the wear on the cables. It is also recommended to fit the cables with extra wear protection at the attachment points and at especially exposed areas.

6 Reference information

6.3.3 Obtain accuracy

6.3.3 Obtain accuracy

Introduction

This section contains useful tips to optimize the overall accuracy of a vision guided robot solution. Knowing which factors influence accuracy can greatly help avoiding the most common pitfalls.

Training the vision tool

When training a vision system to recognize or measure a part, first choose which section of the part that is of interest. This often turns into a trade-off between what relevant features are available and how consistent they are.

Consistency of the feature to be detected is critical and comes into play in various different ways. Is the relevant feature present and similar looking on different individual parts, and can it still be recognized if the part is moved across the field view? This is where lighting comes into play. If the appropriate lighting is not applied, the image processing will be prone to fail either when the part is presented to the camera at different positions, or because the feature to be detected is not sufficiently visible on all parts.

To summarize, it is important to decide which section of the part to measure, and to select illumination that brings out the interesting features of that section in a consistent way.

For more information, see [Obtain good lighting on page 90](#).

The field of view

Many times it may not be necessary to fit the whole part within the field of view in order to perform the desired measurement or identification.

Most important is that relevant features are clearly and consistently visible. Reducing the field not only increases the pixel-to-mm ratio which provides higher accuracy, but it also reduces the area where appropriate lighting needs to be applied. Furthermore it should be said that the perspective from which the camera views the part can influence the result.

A general rule of thumb is that the camera shall, if possible, image the part from right angle. In cases where this is not possible a non-linear calibration such as the checkerboard helps compensate for non perpendicular viewing angles. While having the capability to accurately translate coordinates seen from a tilted angle, it does not automatically solve the problem that vision tools may fail to operate robustly when the image is distorted due to perspective or lens effects.

As a remedy in cases when the image is distorted either by the optics or the viewing angle it is possible to apply a rectifying filter that unwraps the image. Under **Add Part Inspection Tool** and **Image Filter Tools** select **Transform**. This tool takes the result from a grid calibration and uses that information to calculate an "undistorted" version of the image. For subsequently added vision tools the user may choose whether they shall run on the rectified image or the original image.

Continues on next page

The checkerboard / calibration plane

As already mentioned there are various essential factors to the imaging of a scene, and the subsequent image processing that greatly influence the accuracy of the result.

But none of that matters if the geometry, or calibration, of the system is not true to the physical setup. The camera coordinate system is, normally, established by placing a calibration plate with an origin marker under the camera and performing a calibration routine.

The calibration of the camera is most commonly referenced by the robot in the form of a work object for which the user frame is placed in the exact same position as the camera coordinate system.

It is clear that if the work object is not accurately defined to the position of the origin marker of the camera accuracy is lost. But what is more, the results reported by the vision system are always the projection of identified and measured features as projected on the calibration plane. This means that in order to obtain accurate results the features to be measured shall be located in the calibration plane.

6 Reference information

6.3.4 Obtain good lighting

6.3.4 Obtain good lighting

Introduction

It is important to understand that good lighting for machine vision is not the same as good lighting perceived by a human being.

The specific features of the part to be inspected shall be clearly and consistently illuminated so that the result of the image processing is repeatable throughout the full active field of view.

Common problems

The most common problems arise from uneven light distribution, reflections, shadows, and glare. It is critical to the outcome of the image processing that such unwanted artifacts are eliminated or at least minimized. Shiny parts such as metallic items of transparent plastic surfaces generally require careful consideration before selecting a lighting system.

There is no single universal lighting technique that is suitable for all parts and situations. However, by understanding the fundamentals of how image processing works and how lighting can be adjusted to provide good input data, performance and robustness can be greatly improved.

Lighting techniques

As a rule of thumb it can be said that the key to control the light, is to shield out unwanted light and to apply light suitable for the application. To rely on ambient light (partially or fully) is not advisable.

One commonly used technique to shield out light is to use a light hub that provides an enclosure around the work piece to be observed. The inside of the box is fitted with the type of light that best brings out the interesting features. Another method for blocking out unwanted light is to use a combination of an optical bandpass filter mounted on the lens while applying color lighting.

The filter is matched so as only to let through light of the same wavelength (or color) as that applied by the lighting system.

Another technique that is used very successfully in many situations is called backlighting. Instead of illuminating the face of the part, light is instead applied from behind the product so that the contours are brought out. This provides a simplified black on white image that is often easier for the vision tools to process.

How to select a lighting system

The safest way to determine what is an appropriate lighting system for the application at hand is, by far, to perform a practical vision evaluation using the actual part and applying different light sources from different angles.

During an evaluation it is important to consider factors such as stand-off distance, the size of the field of view, what ambient light exists and which vision tool is to be used for the task. All of these factors affect one another and must be tuned to match.

6.3.5 Structuring the vision job

Introduction

Often a vision system is required to detect different parts, either in each image or between different production shifts.

Different approaches can be applied to structure the vision job depending on the current requirements:

Create separate jobs for each product or task

Creating separate jobs provides a structured setup but jobs take several seconds, even up to a minute to load. Each job also contains its own camera calibration, which may be useful if the different tasks require different calibrations.

Create a single job for all products and tasks

Create a single job that contains vision tools for all different parts and production scenarios.

This approach is typically used if the system is required to look for different parts in a single cycle. Adding tools makes the job file bigger, and finally the maximum limit is reached. Unless tools are disabled they all execute with each image acquisition which slows down the vision execution considerably. When placing all vision tools in a single job the various tools are typically switched on and of using RAPID calls, see [Enabling and disabling vision tools during runtime on page 93](#).

6.3.6 Init routine

Description

Always run an initiation routine after powering up or restarting the controller. This ensures that the proper job is loaded, and that the controller and the cameras are in the correct mode.

Event	Description
After a power failure:	<ul style="list-style-type: none">• The camera will loose the job.• The controller will loose the output to RAPID configuration.
After a controller restart:	<ul style="list-style-type: none">• The controller will loose the current status of the camera, program mode/run mode.• The controller will loose the output to RAPID configuration.

Example

```
PROC IV_Init(VAR cameradev cam,string jobname)
  VAR num maxloadtime:=15;

  CamSetProgramMode cam;
  CamLoadJob cam,jobname\MaxTime:=maxloadtime;
  CamSetRunMode cam;

  MoveAbsJ safepos,vl00,fine,tool0;

ERROR
  IF ERRNO=ERR_CAM_BUSY THEN
    TPWrite "ERR_CAM_BUSY. Calling RETRY";
    WaitTime 1.0;
    RETRY;
  ELSEIF ERRNO=ERR_CAM_MAXTIME THEN
    TPWrite "ERR_CAM_MAXTIME. Increasing timeout by 10s and calling
      RETRY";
    maxloadtime:=maxloadtime+10;
    WaitTime 1.0;
    RETRY;
  ELSEIF ERRNO=ERR_CAM_NO_PROGMODE THEN
    TPWrite "ERR_CAM_NO_PROGMODE. Setting camera to program mode
      and calling RETRY";
    WaitTime 1.0;
    CamSetProgramMode cam;
    RETRY;
  ENDIF
ENDPROC
```

6.3.7 Enabling and disabling vision tools during runtime

Description

Sometimes it is known which type of part to look for in the image. If so, the vision tools that are not currently needed can be disabled to reduce the camera processing time.

Consider an example where two vision tools, `Pattern_1` and `Pattern_2` which produce items of type `Item1` and `Item2` respectively. The following procedure describes how to turn the two vision tools on and off.

The disabled tool still produces results with the values from the latest active execution. In order to not use these targets, sort them out in the RAPID program.

Example

This procedure shows how to turn tools on and off to shorten the camera processing time. It also shows how to handle the result queue, when disabling tools.

```
PROC Enable_Disable_Tools(VAR cameradev cam,bool enabletool1,bool
  enabletool2)
  VAR cameratarget mycameratarget;
  VAR string tool1_propertyname:="Pattern_1.Tool_Enabled";
  VAR string tool2_propertyname:="Pattern_2.Tool_Enabled";
  VAR string tool1_corresp_item_name:="Item1";
  VAR string tool2_corresp_item_name:="Item2";
  VAR num maxresulttime:=5;
  CamSetProgramMode cam;
  CamSetParameter cam,tool1_propertyname\BoolVal:=enabletool1;
  CamSetParameter cam,tool2_propertyname\BoolVal:=enabletool2;
  CamSetRunMode cam;
  CamReqImage cam;
  CamGetResult cam,mycameratarget;
  !Before using the result, make sure that it originates from an
    enabled tool.
  !Disabling tools reduces the processing time, but the latest
    result produced by the tool is still communicated.
  IF mycameratarget.name=tool1_corresp_item_name AND enabletool1
    THEN
    TPWrite "Received target of type "+mycameratarget.name+" with
      position "\Pos:=mycameratarget.cframe.trans;
    !Run the robot..
  ELSEIF mycameratarget.name=tool2_corresp_item_name AND enabletool2
    THEN
    TPWrite "Received target of type "+mycameratarget.name+" with
      position "\Pos:=mycameratarget.cframe.trans;
    !Run the robot..
  ENDIF
  !Flush the result queue to get rid of any remaining targets.
  CamFlush cam;
  ERROR
  IF ERRNO=ERR_CAM_BUSY THEN
    TPWrite "ERR_CAM_BUSY. Calling RETRY";
```

Continues on next page

6 Reference information

6.3.7 Enabling and disabling vision tools during runtime

Continued

```
        WaitTime 1.0;
        RETRY;
ELSEIF ERRNO=ERR_CAM_MAXTIME THEN
    TPWrite "ERR_CAM_MAXTIME. Increasing timeout by 5s and acquiring
            new image";
    CamFlush cam;
    CamReqImage cam;
    maxresulttime:=maxresulttime+5;
    WaitTime 1.0;
    RETRY;
ELSEIF ERRNO=ERR_CAM_NO_RUNMODE THEN
    TPWrite "ERR_CAM_NO_RUNMODE. Setting camera to run mode and
            calling RETRY";
    WaitTime 1.0;
    CamSetRunMode cam;
    RETRY;
ELSEIF ERRNO=ERR_CAM_NO_PROGMODE THEN
    TPWrite "ERR_CAM_NO_PROGMODE. Setting camera to program mode
            and calling RETRY";
    WaitTime 1.0;
    CamSetProgramMode cam;
    RETRY;
ENDIF
ENDPROC
```

6.3.8 Avoid running out of space on the camera

Description

The flash disc of the camera has got a limited storage space. If a robot cell requires a lot of different jobs, the camera will at some point not be able to store all jobs.

It can then be convenient to keep only the necessary jobs on the camera and store the rest on the robot controller. When they are needed they can be uploaded from the controller flash disk to the camera.

Moving files between the camera and the controller can be carried out either using RAPID or the FlexPendant explorer, where the flash disks of the cameras appear next to the controller drive.

Below is an example showing how to move files from the home catalog of the system to the camera.

See also [Backup a camera to the controller on page 96](#).

Example

```
VAR string campath;  
VAR string controllerpath;  
...  
campath := CamGetName(mycamera) + "://" + "myjob.job";  
controllerpath := "HOME:/myjob.job";  
CopyFile controllerpath, campath;
```

6.3.9 Backup a camera to the controller

Description

The contents of a camera are not automatically backed up when taking a regular backup of the robot controller. However, the following RAPID routine copies all of the files from a camera to the home directory of the controller. This can be used as a service routine that can be called before taking the regular backup.



Tip

The code is available as a snippet in RobotStudio. There is also a snippet for restoring the files.

Example

```
PROC BackupCamToCtrl(var cameradev cam,bool replaceexistingfiles)
  VAR string ctrldirname:="HOME/IV/";
  VAR dir camdirectory;
  VAR string camdirname;
  VAR string tempfilename;
  VAR string tempcamfilepath;
  VAR string tempctrlfilepath;
  ...
  camdirname:=CamGetName(cam)+":/";
  ctrldirname:=ctrldirname+CamGetName(cam)+"/";
  MakeDir ctrldirname;
  OpenDir camdirectory,camdirname;
  WHILE ReadDir(camdirectory,tempFileName) DO
    tempcamfilepath:=camdirname+tempfilename;
    tempctrlfilepath:=ctrldirname+tempfilename;
    CopyFile tempcamfilepath,tempctrlfilepath;
  ENDWHILE
  CloseDir camdirectory;
ERROR
  IF ERRNO=ERR_FILEEXIST THEN
    IF replaceexistingfiles THEN
      RemoveFile tempctrlfilepath;
      RETRY;
    ELSE
      TRYNEXT;
    ENDIF
  ENDIF
ENDPROC
```

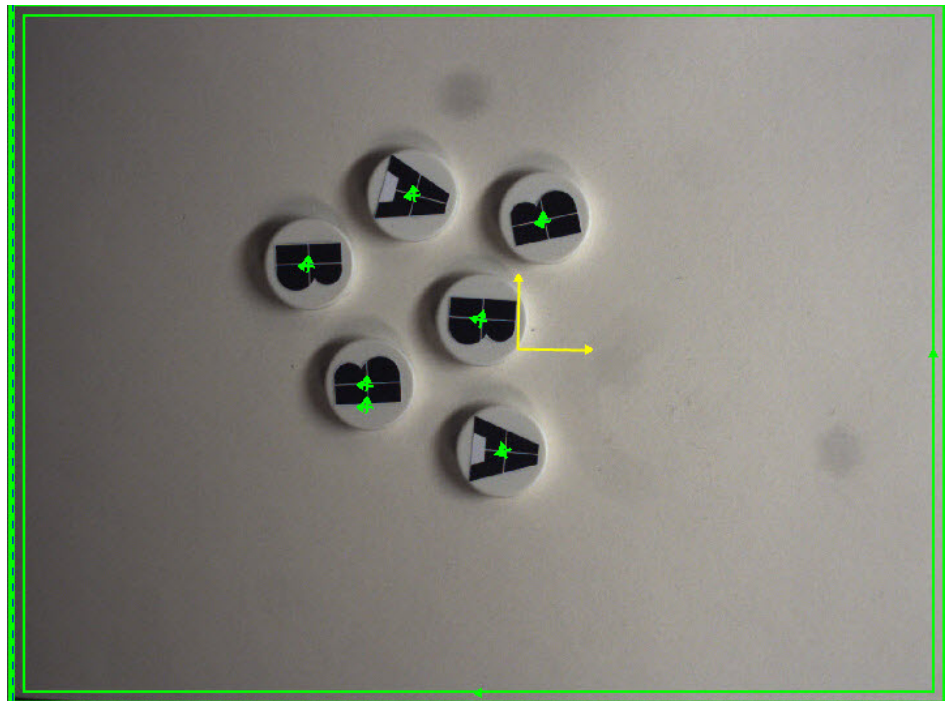

6.3.10 Sort items of different types

Description

Sometimes it is needed to distinguish between products of different types. The following example describes how to distinguish between two chips which have the letters A respective B printed on them.

The solution can be divided into three basic steps:

- 1 Train a vision job that is able to distinguish between the two chip types
- 2 Configure the output to RAPID so that the RAPID program can retrieve both the position and the type of the object.
- 3 Program a RAPID program that retrieves vision targets and checks which type it is before acting on it.



xx1300000190

Train the vision job

First, the vision system must be taught to distinguish between the two part types. This problem can be solved in various ways using the available vision tools. In this case two different part location tools, *PatMax®*, are trained on the features that differ between the two product types - the letters A and B.

There are two reasons for not including the circle in either of the models. The first reason is that the circle is identical for both parts, making them more similar from a vision perspective. The second reason is that the circular feature makes up a large percentage of the model content while not providing any information about the angle of the part. This results in angular uncertainty.

Continues on next page

6 Reference information

6.3.10 Sort items of different types

Continued

Configure the output to RAPID

Second, the output to RAPID must be configured so that the information about which part is which is preserved and can be used for the robot to act on. To do this, two items are configured and named after the part types they represent – A_Chip and B_chip. The outputs of the tool that identifies the A-chip is connected to the components of A_Chip and the same is done for the B-chip.

Program the RAPID program

Finally, the RAPID program must be programmed to act based on the type of the identified target.

Example

```
...
CamGetResult mycamera, mycamtarget;
IF mycamtarget.name = "A_Chip" THEN
    !Do something with the A-Chip
ELSEIF mycamtarget.name = "B_Chip" THEN
    !Do something with the B-Chip
ENDIF
...
```

6.3.11 Finding multiple items of the same type

Description

Often the vision task involves finding multiple parts of the same type spread across the field of view of the camera. Configuring such a task requires some configuration of the vision job and the RAPID program to be made.

- 1 Typically a vision tool that finds multiple items is used, `PatMax[1-10]` or `Blob[1-10]`.
- 2 In the **Output to RAPID** dialog the vision tool is linked to a single item type, for example called *Part_A*. This means that for each found item a camera target with the name property *Part_A* will be produced and sent to the vision queue.
- 3 The task of the RAPID program is to loop through each produced cameratarget and pick up the corresponding object.

Example

The following example acquires an image and moves to each of the reported targets. The camera job may produce multiple targets for each acquired image.

```
PROC Multiple_Target_Pick(VAR cameradev cam)
  VAR bool continueloop:=TRUE;
  VAR num maxresulttime:=5;
  VAR cameratarget mycameratarget;
  VAR num zoffset:=200;
  CamReqImage cam;
  WHILE continueloop DO
    CamGetResult cam, mycameratarget;
    camwobj.oframe:=mycameratarget.cframe;
    MoveL offs(picktarget,0,0,zoffset), v500, z0, picktool
      \WObj:=camwobj;
    MoveL picktarget, v100, fine, picktool \WObj:=camwobj;
    WaitTime 1.0;
    MoveL offs(picktarget,0,0,zoffset), v500, z0,picktool
      \WObj:=camwobj;
    IF CamNumberOfResults(cam)<1 THEN
      continueloop:=FALSE;
    ENDIF
  ENDWHILE
  ERROR
  IF ERRNO=ERR_CAM_BUSY THEN
    TPWrite "ERR_CAM_BUSY. Calling RETRY";
    WaitTime 1.0;
    RETRY;
  ELSEIF ERRNO=ERR_CAM_MAXTIME THEN
    TPWrite "ERR_CAM_MAXTIME. Increasing timeout by 5s and acquiring
      new image";
    CamFlush cam;
    CamReqImage cam;
    maxresulttime:=maxresulttime+5;
```

Continues on next page

6 Reference information

6.3.11 Finding multiple items of the same type

Continued

```
        WaitTime 1.0;
        RETRY;
ELSEIF ERRNO=ERR_CAM_NO_RUNMODE THEN
    TPWrite "ERR_CAM_NO_RUNMODE. Setting camera to run mode and
            calling RETRY";
    WaitTime 1.0;
    CamSetRunMode cam;
    RETRY;
ENDIF
ENDPROC
```

6.3.12 Always check that the vision target is within expected limits

Description

When using sensors it is important to always make sure that a detected position is within the expected work space before attempting approach the position with the robot.

A poorly trained model or calibration failure could produce targets in unexpected positions. As an example the following procedure, also available as a snippet, should be used to minimize the risk of such problems.

Example

```
IF (CamCheckLimits(mycameratarget.cframe, -100, 100, -100, 100,
-90, 90)) THEN
    !Perform move instruction
ELSE
    !Perform recovery routine
ENDIF
FUNC BOOL CamCheckLimits(pose current_pose, num X_min, num X_max,
    num Y_min, num Y_max, num Angle_min, num Angle_max)
    !Checks that the pose is within the specified limits.
    IF (current_pose.trans.X < X_min) RETURN FALSE;
    IF (current_pose.trans.X > X_max) RETURN FALSE;
    IF (current_pose.trans.Y < Y_min) RETURN FALSE;
    IF (current_pose.trans.Y > Y_max) RETURN FALSE;
    IF (EulerZYX(\Z, current_pose.rot) < Angle_min) RETURN FALSE;
    IF (EulerZYX(\Z, current_pose.rot) > Angle_max) RETURN FALSE;
    RETURN TRUE;
ENDFUNC
```

This page is intentionally left blank

7 RAPID reference information

7.1 Instructions

7.1.1 CamFlush - Removes the collection data for the camera

Usage

`CamFlush` is used to flush (remove) the `cameratarget` collection for the camera.

Basic examples

The following example illustrates the instruction `CamFlush`.

Example 1

```
CamFlush mycamera;
```

The collection data for camera `mycamera` is removed.

Arguments

```
CamFlush Camera
```

Camera

Data type: `cameradev`

The name of the camera.

Syntax

```
CamFlush  
[ Camera ':= ' ] < variable (VAR) of cameradev > ';' ;
```

7 RAPID reference information

7.1.2 CamGetParameter - Get different named camera parameters

7.1.2 CamGetParameter - Get different named camera parameters

Usage

`CamGetParameter` is used to get named parameters that the camera may expose. The user has to know the name of the parameter and its return type in order to retrieve its value.

Basic examples

The following example illustrates the instruction `CamGetParameter`.

Example 1

```
VAR bool mybool:=FALSE;
...
CamGetParameter mycamera, "Pattern_1.Tool_Enabled_Status"
  \BoolVar:=mybool;
TPWrite "The current value of Pattern_1.Tool_Enabled_Status is: "
  \Bool:=mybool;
```

Get the named boolean parameter `Pattern_1.Tool_Enabled_Status` and write the value on the `FlexPendant`.

Arguments

`CamGetParameter Camera ParName [\Num] | [\Bool] | [\Str]`

Camera

Data type: `cameradev`
The name of the camera.

ParName

Parameter Name
Data type: `string`
The name of the parameter in the camera.

`[\NumVar]`

Data type: `num`
Variable (**VAR**) to store the numeric value of the data object retrieved.

`[\BoolVar]`

Data type: `bool`
Variable (**VAR**) to store the boolean value of the data object retrieved.

`[\StrVar]`

Data type: `string`
Variable (**VAR**) to store the string value of the data object retrieved.

Program execution

The instruction reads the specified parameter directly when the instruction is executed and returns the value.

Continues on next page

If the instruction is used to read a result from the image analysis, make sure that the camera has finished processing the image before getting the data.

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable `ERRNO` will be set to:

Name	Cause of error
ERR_CAM_BUSY	The camera is busy with some other request and cannot perform the current order.
ERR_CAM_COM_TIMEOUT	Communication error with camera. The camera is probably disconnected.
ERR_CAM_GET_MISMATCH	The parameter fetched from the camera with instruction <code>CamGetParameter</code> has the wrong data type.

Syntax

```
CamGetParameter
[ Camera ':' = ] < variable (VAR) of cameradev > ','
[ ParName ':' = ] < expression (IN) of string >
[ '\NumVar ':' = < variable (VAR) of num > ]
| [ '\BoolVar ':' = < variable (VAR) of bool > ]
| [ '\StrVar ':' = < variable (VAR) of string > ] ';'

```

7 RAPID reference information

7.1.3 CamGetResult - Gets a camera target from the collection

7.1.3 CamGetResult - Gets a camera target from the collection

Usage

`CamGetResult` (*Camera Get Result*) is used to get a camera target from the vision result collection.

Basic examples

The following example illustrates the instruction `CamGetResult`.

Example 1

```
VAR num mysceneid;  
VAR cameratarget mycamtarget;  
...  
CamReqImage mycamera \SceneId:= mysceneid;  
CamGetResult mycamera, mycamtarget \SceneId:= mysceneid;
```

Order camera `mycamera` to acquire an image. Get a vision result originating from the image with `SceneId`.

Arguments

`CamGetResult Camera CamTarget [\SceneId] [\MaxTime]`

Camera

Data type: `cameradev`
The name of the camera.

CamTarget

Camera Target
Data type: `cameratarget`
The variable where the vision result will be stored.

[\SceneId]

Scene Identification
Data type: `num`
The `SceneId` is an identifier that specifies from which image the `cameratarget` has been generated.

[\MaxTime]

Maximum Time
Data type: `num`
The maximum amount of time in seconds that program execution waits. The maximum allowed value is 120 seconds.

Program execution

`CamGetResult` gets a camera target from the vision result collection. If no `SceneId` or `MaxTime` is used, and there is no result to fetch, the instruction will hang forever. If a `SceneId` is used in `CamGetResult` it should have been generated in a preceding `CamReqImage` instruction.

Continues on next page

The `SceneId` can only be used if the image has been ordered from instruction `CamReqImage`. If images are generated by an external I/O signal, the `SceneId` cannot be used in instruction `CamGetResult`.

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable `ERRNO` will be set to:

Name	Cause of error
<code>ERR_CAM_BUSY</code>	The camera is busy with some other request and cannot perform the current order.
<code>ERR_CAM_MAXTIME</code>	No result could be fetched within the time-out time.
<code>ERR_CAM_NO_MORE_DATA</code>	No more vision results can be fetched for used <code>SceneId</code> , or the result could not be fetched within the time-out time.

Syntax

```
CamGetResult
[ Camera ':' = ] < variable (VAR) of cameradev > ','
[ CamTarget ':' = ] < variable (VAR) of CameraTarget >
[ '\SceneId ':' = ] < expression (IN) of num > ]
[ '\MaxTime ':' = ] < expression (IN) of num > ] ';' ;
```

7 RAPID reference information

7.1.4 CamLoadJob - Load a camera task into a camera

7.1.4 CamLoadJob - Load a camera task into a camera

Usage

`CamLoadJob` (*Camera Load Job*) loads a camera task, *job*, describing exposure parameters, calibration, and what vision tools to apply.

Basic examples

The following example illustrates the instruction `CamLoadJob`.

Example 1

```
CamSetProgramMode mycamera;  
CamLoadJob mycamera, "myjob.job";  
CamSetRunMode mycamera;
```

The job `myjob` is loaded into the camera named `mycamera`.

Arguments

```
CamLoadJob Camera JobName [\KeepTargets] [\MaxTime]
```

Camera

Data type: `cameradev`

The name of the camera.

Name

Data type: `string`

The name of the job to load into the camera.

`[\KeepTargets]`

Data type: `switch`

This argument is used to specify if any existing camera targets produced by the camera should be kept.

`[\MaxTime]`

Data type: `num`

The maximum amount of time in seconds that program execution waits. The maximum allowed value is 120 seconds.

Program execution

The execution of `CamLoadJob` will wait until the job is loaded or fail with a time-out error. If the optional argument `KeepTargets` is used, the old collection data for the specified camera is kept. The default behavior is to remove (flush) the old collection data.

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable `ERRNO` will be set to:

Name	Cause of error
<code>ERR_CAM_BUSY</code>	The camera is busy with some other request and cannot perform the current order.

Continues on next page

Name	Cause of error
ERR_CAM_COM_TIMEOUT	Communication error with camera. The camera is probably disconnected.
ERR_CAM_MAXTIME	The camera job was not loaded within the time-out time.
ERR_CAM_NO_PROGMODE	The camera is not in program mode

Limitations

It is only possible to execute `CamLoadJob` when the camera is set in program mode. Use instruction `CamSetProgramMode` to set the camera in program mode. To be able to load the job, the job file must be stored on the camera flash disk.

Syntax

```
CamLoadJob
[ Camera ':= ' ] < variable (VAR) of cameradev > ', '
[ JobName ':= ' ] <expression (IN) of string >
[ '\KeepTargets ]
[ '\MaxTime ':= ' <expression (IN) of num>]';'
```

7 RAPID reference information

7.1.5 CamReqImage - Order the camera to acquire an image

7.1.5 CamReqImage - Order the camera to acquire an image

Usage

`CamReqImage` (*Camera Request Image*) orders the camera to acquire an image.

Basic examples

The following example illustrates the instruction `CamReqImage`.

Example 1

```
CamReqImage mycamera;
```

Order camera `mycamera` to acquire an image.

Arguments

```
CamReqImage Camera [\SceneId] [\KeepTargets] [\AwaitComplete]
```

Camera

Data type: `cameradev`

The name of the camera.

`[\SceneId]`

Scene Identification

Data type: `num`

The optional argument `SceneId` is an identifier for the acquired image. It is generated for each executed `CamReqImage` using the optional argument `SceneId`. The identifier is an integer between 1 and 8388608. If no `SceneId` is used, the identifier value is set to 0.

`[\KeepTargets]`

Data type: `switch`

This argument is used to specify if old collection data for a specified camera should be kept.

`[\AwaitComplete]`

Data type : `switch`

If the optional argument `\AwaitComplete` is specified the instruction waits until the results from the image have been received. If no result was generated, for example because there was not a part in the image, the error `ERR_CAM_REQ_IMAGE` is raised.

When `\AwaitComplete` is used, the camera trigger type has to be set to **External**.

Program execution

`CamReqImage` is ordering a specified camera to acquire an image. If the optional argument `SceneId` is used, the available vision results of an acquired image is marked with the unique number generated by the instruction.

If optional argument `KeepTargets` is used, the old collection data for the specified camera is kept. The default behavior is to remove (flush) any old collection data.

Continues on next page

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable `ERRNO` will be set to:

Name	Cause of error
ERR_CAM_BUSY	The camera is busy with some other request and cannot perform the current order.
ERR_CAM_COM_TIMEOUT	Communication error with camera. The camera is probably disconnected.
ERR_CAM_NO_RUNMODE	The camera is not in running mode
ERR_CAM_REQ_IMAGE	The camera could not generate any result from the image.

Limitations

It is only possible to execute `CamReqImage` when the camera is set in running mode. Use instruction `CamSetRunMode` to set the camera in running mode.

Syntax

```
CamReqImage
[ Camera ':' = ' ] < variable (VAR) of cameradev > ', '
[ '\SceneId ':' = ' < variable (VAR) of num > ]
[ '\KeepTargets ]
[ '\AwaitComplete ]';'
```

7 RAPID reference information

7.1.6 CamSetExposure - Set camera specific data

7.1.6 CamSetExposure - Set camera specific data

Usage

`CamSetExposure` (*Camera Set Exposure*) sets camera specific data and makes it possible to adapt image parameters depending on ambient lighting conditions.

Basic examples

The following example illustrates the instruction `CamSetExposure`.

Example 1

```
CamSetExposure mycamera \ExposureTime:=10;
```

Order the camera `mycamera` to change the exposure time to 10 ms.

Arguments

```
CamSetExposure Camera [\ExposureTime] [\Brightness] [\Contrast]
```

Camera

Data type: `cameradev`

The name of the camera.

`[\ExposureTime]`

Data type: `num`

If this optional argument is used, the exposure time of the camera is updated. The value is in milliseconds (ms).

`[\Brightness]`

Data type: `num`

If this optional argument is used, the brightness setting of the camera is updated. The value is normally expressed on a scale from 0 to 1.

`[\Contrast]`

Data type: `num`

If this optional argument is used, the contrast setting of the camera is updated. The value is normally expressed on a scale from 0 to 1.

Program execution

The instruction updates the exposure time, brightness and contrast if it is possible to update those for the specific camera. If a setting is not supported by the camera an error message will be presented to the user, and the program execution stops.

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable `ERRNO` will be set to:

Name	Cause of error
<code>ERR_CAM_COM_TIMEOUT</code>	Communication error with camera. The camera is probably disconnected.

Continues on next page

Syntax

```
CamSetExposure
[ Camera ':' = ' ] < variable (VAR) of cameradev > ', '
[ '\ 'ExposureTime ':' = ' < variable (IN) of num > ]
[ '\ 'Brightness ':' = ' < variable (IN) of num > ]
[ '\ 'Contrast ':' = ' < variable (IN) of num > ] ';'

```

7 RAPID reference information

7.1.7 CamSetParameter - Set different named camera parameters

7.1.7 CamSetParameter - Set different named camera parameters

Usage

`CamSetParameter` is used to set different named camera parameters that a camera may expose. With this instruction it is possible to change different parameters in the camera in runtime. The user has to know the name of the parameter and its type in order to set its value.

Basic examples

The following example illustrates the instruction `CamSetParameter`.

Example 1

```
CamSetParameter mycamera, "Pattern_1.Tool_Enabled" \BoolVal:=FALSE;  
CamSetRunMode mycamera;
```

In this example the parameter named "Pattern_1.Tool_Enabled" is set to false, which means that the specified vision tool shall not execute when an image is acquired.

This will give a faster execution of the vision tool. However, the tool still produces results with the values from the latest active execution. In order to not use these targets, sort them out in the RAPID program.

Arguments

```
CamSetParameter Camera ParName [\Num] | [\Bool] | [\Str]
```

Camera

Data type: `cameradev`

The name of the camera.

ParName

Data type: `string`

The name of the parameter in the camera.

[\NumVal]

Data type: `num`

The numeric value to set for the camera parameter with the name set in argument `ParName`.

[\BoolVal]

Data type: `bool`

The boolean value to set for the camera parameter with the name set in argument `ParName`.

[\StrVal]

Data type: `string`

The string value to set for the camera parameter with the name set in argument `ParName`.

Continues on next page

7.1.7 CamSetParameter - Set different named camera parameters
Continued

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable `ERRNO` will be set to:

Name	Cause of error
ERR_CAM_BUSY	The camera is busy with some other request and cannot perform the current order.
ERR_CAM_COM_TIMEOUT	Communication error with camera. The camera is probably disconnected.
ERR_CAM_SET_MISMATCH	The parameter written to the camera with instruction CamSetParameter has the wrong data type, or the value is out of range.

Syntax

```
CamSetParameter
[ Camera ':' = ' ] < variable (VAR) of cameradev > ' , '
[ ParName ':' = ' ] < expression (IN) of string >
[ '\NumVal ':' = ' < expression (IN) of num > ]
| [ '\BoolVal ':' = ' < expression (IN) of bool > ]
| [ '\StrVal ':' = ' < expression (IN) of string > ] ';'

```

7 RAPID reference information

7.1.8 CamSetProgramMode - Orders the camera to go to program mode

7.1.8 CamSetProgramMode - Orders the camera to go to program mode

Usage

`CamSetProgramMode` (*Camera Set Program Mode*) orders the camera to go to program mode (offline).

Basic examples

The following example illustrates the instruction `CamSetProgramMode`.

Example 1

```
CamSetProgramMode mycamera;  
CamLoadJob mycamera, "myjob.job";  
CamSetRunMode mycamera;  
...
```

First, change the camera to programming mode. Then load `myjob` into the camera. Then, order the camera to go to running mode.

Arguments

`CamSetProgramMode Camera`

Camera

Data type: `cameradev`

The name of the camera.

Program execution

When ordering a camera to go to program mode with instruction `CamSetProgramMode`, it will be possible to change settings and load a job into the camera.

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable `ERRNO` will be set to:

Name	Cause of error
<code>ERR_CAM_BUSY</code>	The camera is busy with some other request and cannot perform the current order.
<code>ERR_CAM_COM_TIMEOUT</code>	Communication error with camera. The camera is probably disconnected.

Syntax

```
CamSetProgramMode  
[ Camera ':' = ' ] < variable (VAR) of cameradev > ';' ;
```

7.1.9 CamSetRunMode - Orders the camera to run mode

Usage

`CamSetRunMode` (*Camera Set Running Mode*) orders the camera to go to run mode (online), and updates the controller on the current output to RAPID configuration.

Basic examples

The following example illustrates the instruction `CamSetRunMode`.

Example 1

```
CamSetProgramMode mycamera;
CamLoadJob mycamera, "myjob.job";
...
CamSetRunMode mycamera;
```

First, change the camera to programming mode. Then load `myjob` into the camera. Then, order the camera to go to running mode with instruction `CamSetRunMode`.

Arguments

`CamSetRunMode Camera`

Camera

Data type: `cameradev`

The name of the camera.

Program execution

When ordering a camera to go to run mode with `CamSetRunMode` it is possible to start taking images.

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable `ERRNO` will be set to:

Name	Cause of error
<code>ERR_CAM_BUSY</code>	The camera is busy with some other request and cannot perform the current order.
<code>ERR_CAM_COM_TIMEOUT</code>	Communication error with camera. The camera is probably disconnected.

Syntax

```
CamSetRunMode
[ Camera ':' = ] < variable (VAR) of cameradev > ';' ;
```

7 RAPID reference information

7.1.10 CamStartLoadJob - Start load of a camera task into a camera

7.1.10 CamStartLoadJob - Start load of a camera task into a camera

Usage

`CamStartLoadJob` will start the loading of a job into a camera, and then the execution will continue on the next instruction. When loading is in progress other instructions can be executed in parallel.

Basic examples

The following example illustrates the instruction `CamStartLoadJob`.

Example 1

```
...  
CamStartLoadJob mycamera, "myjob.job";  
MoveL p1, v1000, fine, tool2;  
CamWaitLoadJob mycamera;  
CamSetRunMode mycamera;  
CamReqImage mycamera;  
...
```

First a job loading is started to the camera, and while the loading is proceeding, a movement to position p1 is done. When the movement is ready, and the loading has finished, an image is acquired.

Arguments

`CamStartLoadJob` Camera Name [`\KeepTargets`]

Camera

Data type: `cameradev`

The name of the camera.

Name

Data type: `string`

The name of the job to load into the camera.

[`\KeepTargets`]

Data type: `switch`

This argument is used to specify if old collection data for a specified camera should be kept.

Program execution

Execution of `CamStartLoadJob` will only order the loading and then proceed directly with the next instruction without waiting for the loading to be completed. If optional argument `\KeepTargets` is used, the old collection data for the specified camera is not removed. The default behavior is to remove (flush) old collection data.

Continues on next page

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable `ERRNO` will be set to:

Name	Cause of error
<code>ERR_CAM_BUSY</code>	The camera is busy with some other request and cannot perform the current order.

Limitations

It is only possible to execute `CamStartLoadJob` when the camera is set in program mode. Use instruction `CamSetProgramMode` to set the camera in program mode.

When an ongoing load of a job is executing, it is not possible to access that specific camera with any other instruction or function. The following camera instruction or function must be a `CamWaitLoadJob` instruction.

To be able to load the job, the job file must be stored on the camera flash disk.

Syntax

```
CamStartLoadJob
[ Camera ':= ' ] < variable (VAR) of cameradev > ', '
[ Name ':= ' ] < expression (IN) of string >
[ '\KeepTargets ' ] ;
```

7 RAPID reference information

7.1.11 CamWaitLoadJob – Wait until a camera task is loaded

7.1.11 CamWaitLoadJob – Wait until a camera task is loaded

Usage

CamWaitLoadJob (*Camera Wait Load Job*) will wait until the loading of a job into a camera is ready.

Basic examples

The following example illustrates the instruction CamWaitLoadJob.

Example 1

```
...
CamStartLoadJob mycamera, "myjob.job";
MoveL p1, v1000, fine, tool2;
CamWaitLoadJob mycamera;
CamSetRunMode mycamera;
CamReqImage mycamera;
...
```

First a job loading is started to the camera, and while the loading is proceeding, a movement to position p1 is done. When the movement is ready, and the loading has finished, an image is acquired.

Arguments

CamWaitLoadJob Camera

Camera

Data type: cameradev

The name of the camera.

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable ERRNO will be set to:

Name	Cause of error
ERR_CAM_COM_TIMEOUT	Communication error with camera. The camera is probably disconnected.

Limitations

It is only possible to execute CamWaitLoadJob when the camera is set in program mode. Use instruction CamSetProgramMode to set the camera in program mode. When an ongoing load of a job is executing, it is not possible to access that specific camera with any other instruction or function. The following camera instruction or function must be a CamWaitLoadJob instruction.

Syntax

```
CamWaitLoadJob
[ Camera ':' = ' ] < variable (VAR) of cameradev > ';'

```

7.2 Functions

7.2.1 CamGetExposure - Get camera specific data

Usage

CamGetExposure (Camera Get Exposure) is a function that reads the current settings for a camera. With this function and with the instruction **CamSetExposure** it is possible to adapt the camera images depending on environment in runtime.

Basic examples

The following example illustrates the function **CamGetExposure**.

Example 1

```
VAR num exposuretime;
...
exposuretime:=CamGetExposure(mycamera \ExposureTime);
IF exposuretime = 10 THEN
  CamSetExposure mycamera \ExposureTime:=9.5;
ENDIF
```

Order camera **mycamera** to change the exposure time to 9.5 ms if the current setting is 10 ms.

Return value

Data type: num

One of the settings exposure time, brightness, or contrast returned from the camera as a numerical value.

Arguments

```
CamGetExposure (Camera [\ExposureTime] | [\Brightness] |
[\Contrast])
```

Camera

Data type: cameradev

The name of the camera.

[\ExposureTime]

Data type: num

Returns the cameras exposure time. The value is in milliseconds (ms).

[\Brightness]

Data type: num

Returns the brightness setting of the camera

[\Contrast]

Data type: num

Returns the contrast setting of the camera

Continues on next page

7 RAPID reference information

7.2.1 CamGetExposure - Get camera specific data

Continued

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable `ERRNO` will be set to:

Name	Cause of error
ERR_CAM_BUSY	The camera is busy with some other request and cannot perform the current order.
ERR_CAM_COM_TIMEOUT	Communication error with camera. The camera is probably disconnected.

Syntax

```
CamGetExposure '('  
  [ Camera ':' = ] < variable (VAR) of cameradev >  
  [ '\ExposureTime]  
  | [ '\Brightness]  
  | [ '\Contrast ] ')'
```

A function with a return value of the data type `num`.

7.2.2 CamGetLoadedJob - Get name of the loaded camera task

Usage

CamGetLoadedJob (*Camera Get Loaded Job*) is a function that reads the name of the current loaded job from the camera and returns it in a string.

Basic examples

The following example illustrates the function CamGetLoadedJob.

Example 1

```
VAR string currentjob;
...
currentjob:=CamGetLoadedJob(mycamera);
IF CurrentJob = "" THEN
  TPWrite "No job loaded in camera "+CamGetName(mycamera);
ELSE
  TPWrite "Job "+CurrentJob+" is loaded in camera "
    "+CamGetName(mycamera);
ENDIF
```

Write the loaded job name on the FlexPendant.

Return value

Data type: string

The current loaded job name for the specified camera.

Arguments

CamGetLoadedJob (Camera)

Camera

Data type: cameradev

The name of the camera.

Program execution

The function CamGetLoadedJob gets the current loaded job name from the camera. If no job is loaded into the camera, an empty string is returned.

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable ERRNO will be set to:

Name	Cause of error
ERR_CAM_BUSY	The camera is busy with some other request and cannot perform the current order.
ERR_CAM_COM_TIMEOUT	Communication error with camera. The camera is probably disconnected.

Syntax

```
CamGetLoadedJob '('
  [ Camera ':= ' ] < variable (VAR) of cameradev > ')'
```

Continues on next page

7 RAPID reference information

7.2.2 CamGetLoadedJob - Get name of the loaded camera task

Continued

A function with a return value of the data type `string`.

7.2.3 CamGetName - Get the name of the used camera

Usage

CamGetName (Camera Get Name) is used to get the configured name of the camera.

Basic examples

The following example illustrates the function CamGetName.

Example 1

```
...
logcameraname camera1;
CamReqImage camera1;
...
logcameraname camera2;
CamReqImage camera2;
...
PROC logcameraname(VAR cameradev camdev)
  TPWrite "Now using camera: "+CamGetName(camdev);
ENDPROC
```

The procedure logs the name of the currently used camera to the FlexPendant.

Return value

Data type: string

The name of the currently used camera returned as a string.

Arguments

CamGetName (Camera)

Camera

Data type: cameradev

The name of the camera.

Syntax

```
CamGetName( '('
  [ Camera ':= ' ] < variable (VAR) of cameradev > ')'
```

A function with a return value of the data type string.

7 RAPID reference information

7.2.4 CamNumberOfResults - Get number of available results

7.2.4 CamNumberOfResults - Get number of available results

Usage

`CamNumberOfResults` (Camera Number of Results) is a function that reads the number of available vision results and returns it as a numerical value.

Basic examples

The following example illustrates the function `CamNumberOfResults`.

Example 1

```
VAR num foundparts;  
...  
CamReqImage mycamera;  
WaitTime 1;  
FoundParts := CamNumberOfResults(mycamera);  
TPWrite "Number of identified parts in the camera image:  
        "\Num:=foundparts;
```

Acquire an image. Wait for the image processing to complete, in this case 1 second. Read the number of identified parts and write it to the FlexPendant.

Return value

Data type: num

Returns the number of results in the collection for the specified camera.

Arguments

`CamNumberOfResults` (Camera [`\SceneId`])

Camera

Data type: cameradev

The name of the camera.

[`\SceneId`]

Scene Identification

Data type: num

The `SceneId` is an identifier that specifies from which image to read the number of identified parts.

Program execution

`CamNumberOfResults` is a function that reads the number of available vision results and returns it as a numerical value. Can be used to loop through all available results.

The function returns the queue level directly when the function is executed. If the function is executed directly after requesting an image, the result is often 0 because the camera has not yet finished processing the image.

Continues on next page

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable `ERRNO` will be set to:

Name	Cause of error
ERR_CAM_BUSY	The camera is busy with some other request and cannot perform the current order.

Syntax

```
CamNumberOfResults '('  
  [ Camera ':= ' ] < variable (VAR) of cameradev >  
  [ '\SceneId ':= ' < expression (IN) of num > ] ')'
```

A function with a return value of the data type `num`.

7 RAPID reference information

7.3.1 cameradev - camera device

7.3 Data types

7.3.1 cameradev - camera device

Usage

`cameradev` (*camera device*) is used to define the different camera devices which can be controlled and accessed from the RAPID program. The data type `cameradev` is used for instructions and functions communicating with a camera. The names of the camera devices are defined in the system parameters and, consequently, must not be defined in the program.

Description

Data of the type `cameradev` only contains a reference to the camera device.

Limitations

Data of the type `cameradev` must not be defined in the program. However, if it is then an error message will be displayed as soon as an instruction or function that refers to this `cameradev` is executed. The data type can, on the other hand, be used as a parameter when declaring a routine.

Predefined data

All cameras defined in the system parameters are predefined in every program task.

Basic examples

The following example illustrates the data type `cameradev`.

Example 1

```
CamLoadJob mycamera, "myjob.job";
```

Characteristics

`cameradev` is a non-value data type. This means that data of this type does not permit value-oriented operations.

7.3.2 cameratarget - camera data

Usage

`cameratarget` is used to exchange vision data from the camera image to the RAPID program.

Description

Data of the type `cameratarget` is a user defined collection of data that can be set up to exchange vision data from the camera image to the RAPID program. The data has a range of components that can be set up according to the specific needs in the current vision application. The `cframe` component is meant for transmitting information about the location of an object whereas the numerical values and the strings are meant to hold inspection data.

Components

The data type has the following components:

`name`

Data type: `string`

The name identifier of the `cameratarget`.

`cframe`

current frame

Data type: `pose`

For storing position data which is normally used for guiding the robot by modifying the work object.

`val1`

value 1

Data type: `num`

For storing numerical outputs such as measurements.

...

`val5`

value 5

Data type: `num`

For storing numerical outputs such as measurements.

`string1`

Data type: `string`

For storing numerical vision output such as inspection or identification output.

`string2`

Data type: `string`

For storing numerical vision output such as inspection or identification output.

Continues on next page

7 RAPID reference information

7.3.2 cameratarget - camera data

Continued

type

Data type: num

A numerical identifier of the camera target. Similar purpose as the `name` component.

cameraname

Data type: string

The name of the camera.

sceneid

scene identification

Data type: num

The unique identifier of the image used to generate the `cameratarget`.

Basic examples

The following example illustrates the data type `cameratarget`.

Example 1

```
VAR cameratarget target1;  
...  
wobjmycamera.oframe := target1.cframe;  
MoveL pickpart, v100, fine, mygripper \WObj:= wobjmycamera;
```

The `cframe` coordinate transformation is assigned to the object frame of the work object. The `robtargt pickpart` has previously been tuned to a correct picking position within the object frame of the work object.

Structure

```
< dataobject of cameratarget >  
  < name of string >  
  < cframe of pose >  
    < trans of pos >  
    < rot of orient >  
  < val1 of num >  
  < val2 of num >  
  < val3 of num >  
  < val4 of num >  
  < val5 of num >  
  < string1 of string >  
  < string2 of string >  
  < type of num >  
  < cameraname of string >  
  < sceneid of num >
```

Index

A

advanced settings, 27

B

backup, 96

C

calibrated camera frame, 80

calibrating

camera , 17, 60, 82

camera to robot , 17, 62, 83

checkerboard , 17

vision, 17

cameradev, 128

camera emulator, 36, 51

camera network, 47

cameratarget, 129

CamFlush, 103

CamGetExposure, 121

CamGetLoadedJob, 123

CamGetName, 125

CamGetParameter, 104

CamGetResult, 106

CamLoadJob, 108

CamNumberOfResults, 126

CamReqlImage, 110

CamSetExposure, 112

CamSetParameter, 114

CamSetProgramMode, 116

CamSetRunMode, 117

CamStartLoadJob, 118

CamWaitLoadJob, 120

checklist, 16

Cognex EasyBuilder®, 13, 17

Cognex In-Sight®, 13, 17

Configuring Integrated Vision, 45

context window, 23, 35

controller browser, 23, 29

coordinate systems, 79

D

data mapping, 67

date, 49

disconnect camera, 25, 29, 48

E

emulator, camera, 36, 51

F

fiducial, 17, 62

filmstrip, 23, 32

firmware, 50

G

getting started, 16

glossary, 17

gripping parts, 74

H

hardware, 13, 16

I

I/O, 70

inputs, 70

inspection tools, 64

installing

hardware, 19

RobotStudio, 21

RobotWare, 21

software, 21

IP-address, 49

J

job, 17

L

language, 38

lighting, 90

limitations, 68

links, 65

location tools, 63

N

network settings, 49

O

online help, 24

options dialog, 36

outputs, 70

P

palette window, 23, 34

pass and fail, 64

pointing tool, 17

protect job, 39

R

RAPID program, 72

remove camera, 48

restore, 96

ribbon, 23, 25

S

safety, 11

shortcuts, 30, 38

smart camera, 17

snippet, 17, 72

software, 13, 16

sort items, 97

spreadsheet, 37

starting production, 77

status bar, 31

subnet, 49

T

terminology, 24

term list, 17

time, 49

U

user interface

FlexPendant, 41

RobotStudio, 23

V

vc_network_definition, 50–51

VGR, 17

virtual controller, 50

vision job, 17

vision tools, 63

Contact us

ABB AB

**Discrete Automation and Motion
Robotics**

S-721 68 VÄSTERÅS, Sweden

Telephone +46 (0) 21 344 400

ABB AS, Robotics

Discrete Automation and Motion

Nordlysvegen 7, N-4340 BRYNE, Norway

Box 265, N-4349 BRYNE, Norway

Telephone: +47 51489000

ABB Engineering (Shanghai) Ltd.

No. 4528 Kangxin Hingway

PuDong District

SHANGHAI 201319, China

Telephone: +86 21 6105 6666

www.abb.com/robotics