ROBOTICS

Application manual

PickMaster® 3

Trace back information:
Workspace Main version a511
Checked in 2023-05-22
Skribenta version 5.5.019

Application manual

PickMaster® 3

PickMaster 3.57, RobotWare 6.15

Document ID: 3HAC031978-001

Revision: V

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, damage 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.

Keep for future reference.

Additional copies of this manual may be obtained from ABB.

Original instructions.

Table of contents

Overview of this manual

About this manual

This manual contains instructions for installation, configuration, and daily operation of PickMaster 3.

Usage

This manual should be used during installation, configuration, and maintenance of a PickMaster 3 system.

Who should read this manual?

This manual is intended for:

- Installation personnel
- Programmers
- Operators

Prerequisites

Any maintenance/repair/installation personnel 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
Product specification - PickMaster® 3	*3HAC041347-001*
Circuit diagram - IRC5 IRC5 of design 14	*3HAC024480-011*
Operating manual - RobotStudio	*3HAC032104-001*
Application manual - Conveyor tracking	*3HAC050991-001*
Product manual - IRC5	*3HAC021313-001*
Product manual - IRC5 Panel Mounted Controller	*3HAC027707-001*
Operating manual - IRC5 with FlexPendant	*3HAC050941-001*
Operating manual - IRC5 Integrator's guide	*3HAC050940-001*
Technical reference manual - System parameters	*3HAC050948-001*
Operating manual - Troubleshooting IRC5	*3HAC020738-001*
Technical reference manual - RAPID Instructions, Functions and Data types	*3HAC050917-001*
Technical reference manual - RAPID Overview	*3HAC050947-001*
Product manual - IRB 140	*3HAC027400-001*
Product manual - IRB 260	*3HAC026048-001*
Product manual - IRB 360	*3HAC030005-001*
Product manual - IRB 1600, type 0	*3HAC023637-001*
Product manual - IRB 4600	*3HAC033453-001*

Continues on next page

Continued

External references

Reference	Description
http://www.hilscher.com	Information about the Hilscher fieldbus plug-in.
Cognex Ethernet Camera Tool	For configuring camera networks.
Gigabit Ethernet Performance Driver	For camera communication.
Avoid_EMI_ESD_in_camera_installations	Information about electrical installations of for example Gigabit Ethernet cameras. Located on the 'PickMaster3 Installation Package' in the folder User Documentation\Vision
CognexPCConfigGuide	Detailed information about PC requirements for the vision system.

Revisions

Revision	Description
-	Released with PickMaster 3.22.
A	Released with PickMaster 3.30.
B	Released with PickMaster 3.31. This revision includes the following changes: • Manual title is updated. • Manual is restructured. • Functionality for color vision is added. • Functionality for handling products with varying heights is added. • Hidden instructions and data types removed from the RAPID reference chapter. • New circuit diagrams are added.
C	Released with PickMaster 3.32. This revision includes the following changes: • Updated the section *Optional robot configuration on page 51*. • Updated the chapter *Configuring position sources on page 99* • Updated the chapter *Configuring Remote Integration Services on page 145* • Updated the section *Exporting and importing the RIS configuration on page 153*. • Circuit diagrams are removed from the manual, but delivered as separate files, see *PickMaster 3 - 3HAC024480-008*
D	Released with PickMaster 3.33. This revision includes the following changes: • Added section *Configuring a simulated camera on page 70*

Continues on next page

Revision	Description
E	Released with PickMaster 3.35. This revision includes the following changes: • Updated the section *Connecting encoders on page 43*. • Figures are added in the sections *Configuring the work area on page 107*, *Introduction to User Hook on page 170* • Added the section *External sensor .NET on page 177*. • The sections about *Warnings 4316, 4317, and 4318* and *Scenes or items that are discarded* are deleted. • Added the section *Warnings 4326 - 4329 on page 209*. • RAPID reference - Instructions *AckItmTgt - Acknowledge an item target on page 251* and *GetItmTgt - Get the next item target on page 254* updated, *ResetFlowCount - Reset flow counter on page 264* is added. • RAPID reference - Functions *GetQueueLevel - Get queue level on page 270* updated, *GetFlowCount - Get number of passed items on page 273* is added. • RAPID reference - Data types *itmtgt - Item target data on page 274* is updated.
F	Released with PickMaster 3.40. This revision includes the following updates: • Added the error code **12337** in the section *Vision error codes on page 228*. • Removed the section **Activating a trial license**.
G	Released with PickMaster 3.41. This revision includes the following updates: • Added *Overview* section in *Welcome to PickMaster on page 19*. • Software installation moved from Configuration to Installation chapter. For more information, see *PickMaster software on page 25*. • Computer Requirements is updated and section title changed to *System requirements on page 30*. • Added Screenshots in *Configuring Remote Integration Services on page 145*. • Added *Post inspection on page 190*. • Updated *Histogram on page 130* and *Caliper on page 133*.
H	Released with PickMaster 3.41. This revision includes the following updates: • Added the sections *RIS2 API on page 233*, *RIS Proxy on page 186*, and *RIS2 plug-in on page 162*.
J	Released with PickMaster 3.42. This revision includes the following updates: • Updated the Introduction section of the chapter *Configuring Remote Integration Services on page 145*. • Updated the description for the error code **4314** in the Troubleshooting chapter.

Continues on next page

Continued

Revision	Description
K	Released with PickMaster 3.43. This revision includes the following updates: • Removed the sections about Coordinated and Uncoordinated routines from the manual. • Information about **Save Configuration** is added to the section *Starting the RIS proxy application on page 186*. • Updated the section *Managing roles and users on page 147*. • Information about auto generate trigger distance and position filter are added to the section *Configuring position sources on page 99*.
L	Released with PickMaster 3.44. This revision includes the following update: • Added information about the RIS2 commands `Project status`, `Start robot`, `Pause robot`, and `Stop robot` in the chapter *RIS2 API on page 233*. • Added information about the error code 8342 in the section *Robot error codes on page 223*.
M	Released with PickMaster 3.44. Minor updates.
N	Released with PickMaster 3.50. • Added information about the support for S4Cplus. • Updated the section *Configuring the line on page 59*. • Updated the section *Managing roles and users on page 147*. • Updated the section *RIS2 plug-in on page 162*.
P	Released with PickMaster 3.51. • Updated the path to the utility files in the section *Robot error codes on page 223*. • Updated the section *Configuring inspection models on page 127*.
Q	Released with PickMaster 3.52. • Migrated information from the Release Notes to the User manual.
R	Released with PickMaster 3.53. • Updated multiple sections for the RAPID program updates. • Updated the section *Tuning on page 206*. • Added the section *AlwaysClearPath on page 285*.
S	Released with PickMaster 3.54. Added the following sections: • Added the section *Trigger strobe connection for 2000034085 (Basler Scout camera) on page 33*. • Added the section *Problem with camera resolution in PickMaster on page 217*. • Added the error codes 12341 and 12342 in the section *Vision error codes on page 228*. Updated the following sections: • Updated the section *System requirements on page 30*. • Added a NOTE in the section *Connecting the encoders on page 43*. • Updated the procedure in the section *Configuring containers on page 97*.

Continues on next page

Revision	Description
T	Released with PickMaster 3.55. Following are the updates: • Added the section *Hilscher CifX 50E-RE plug-in on page 155*. • Added the section *UseReachableTargets - Use reachable targets on page 266*. • Added the section *GetMaxUsageTime - Get max measured usage time on page 269*. • Added the section *ResetMaxUsageTime - Reset max measured usage time on page 265*. • Updated the section *Trigger strobe connection for 2000034085 (Basler Scout camera) on page 33*. • Updated the section *Software requirements on page 30*.
U	Released with PickMaster 3.56. • Added the section *Installation of DSQC2000 on page 54*. • Added the section *Overlap filter on page 102*. • Added the section *RIS request flowchart on page 167*. • Added the error code **4318** and **8337** in the Troubleshooting chapter. • Updated the section *Configuring the camera on page 68*. • Updated the introduction section of *Configuring Remote Integration Services on page 145*. • Updated the section *Conveyor work area signals on page 64*. • Updated the instruction *UseReachableTargets - Use reachable targets on page 266*.
V	Released with PickMaster 3.57. • Added the section *Network security on page 14*. • Added the section *Open source and 3rd party components on page 15*. • Added the chapter *Cyber security for PickMaster system networks on page 321*. • Added the section *Example: Optimize target release zone with real time speed of linear conveyor on page 310*. • Added the section *Robot base frame configuration for ceiling-mounted IRB on page 61*. • Added the chapter spare part, see *Spare parts on page 311*. • Updated the network topology image in the section *Prerequisites for vision networks on page 45*. • Added a NOTE regarding item region in STEP 7 in the section *Configuring a PatMax vision model on page 116*. • Updated the definition of Adjustment speed in section *Six axes robot configuration on page 52*. • Updated the section *PickMaster software on page 25*. • Updated the section *Software requirements on page 30*. • Updated the section *Introduction to the controller network on page 45*. • Updated the section *Configuring the vision network on page 46*.

Network security

Network security

This product is designed to be connected to and to communicate information and data via a network interface. It is your sole responsibility to provide, and continuously ensure, a secure connection between the product and to your network or any other network (as the case may be).

You shall establish and maintain any appropriate measures (such as, but not limited to, the installation of firewalls, application of authentication measures, encryption of data, installation of anti-virus programs, etc) to protect the product, the network, its system and the interface against any kind of security breaches, unauthorized access, interference, intrusion, leakage and/or theft of data or information. ABB Ltd and its entities are not liable for damage and/or loss related to such security breaches, any unauthorized access, interference, intrusion, leakage and/or theft of data or information.

Open source and 3rd party components

Open source and 3rd party components

ABB products use software provided by third parties, including open source software. The following copyright statements and licenses apply to various components that are distributed inside the ABB software. Each ABB product does not necessarily use all of the listed third party software components. Licensee must fully agree and comply with these license terms or the user is not entitled to use the product. Start using the ABB software means accepting also referred license terms. The third party license terms apply only to the respective software to which the license pertains, and the third party license terms do not apply to ABB products. With regard to programs provided under the GNU general public license and the GNU lesser general public license licensor will provide licensee on demand, a machine-readable copy of the corresponding source code. This offer is valid for a period of three years after delivery of the product.

ABB software is licensed under the ABB end user license agreement, which is provided separately.

There is license information in the folder `...\licenses` in PickMaster3 distribution package.

This page is intentionally left blank

1 Safety

Safety of personnel

A robot is heavy and extremely powerful regardless of its speed. A pause or long stop in movement can be followed by a fast hazardous movement. Even if a pattern of movement is predicted, a change in operation can be triggered by an external signal resulting in an unexpected movement.

Therefore, it is important that all safety regulations are followed when entering safeguarded space.

Safety regulations

Before beginning work with the robot, make sure you are familiar with the safety regulations described in the manual *Safety manual for robot - Manipulator and IRC5 or OmniCore controller*.

When using PickMaster® products

- When using with PickMaster® products, it is the user's responsibility to adhere to the relevant standards and safety directives. In addition, the application manuals for proper use must be observed.
- Only personnel with appropriate training and required knowledge are allowed to use PickMaster® products.
- The integrator installing the PickMaster® is responsible for the safety.
- Wherever possible, the auto mode of operation shall be performed with all persons outside the safeguarded space.
- An emergency stop must also be available to make sure the emergency stop function is enabled.
- PickMaster® only provides Operational Stop (Program Stop). The integrator shall make sure that proper Normal Stop (machinery stop) is configured correctly in the system.
- Make sure the hazardous situation that resulted in the emergency stop condition no longer exists. Release the emergency stop button manually to remove the emergency stop condition.
- Stops for the machine is the responsibility of the integrator and must be addressed according to local legislation.
- The integrator is responsible to conduct a risk assessment of the final application.
- Sensitive body parts, such as the eyes and the larynx, must be protected by personal protective equipment (PPE).
- Protective measures should be the precondition when using PickMaster® products. PickMaster® does not guarantee the robot targets are always in safe zone. It is integrator's responsibility to take protection measures, like using safe-move or setting proper robot work range etc.

Continues on next page

Continued

- Safety related status and operations shall be handled on the controller and by safety rated systems. PickMaster® status information shall not be used as input for safety related information and operations.
- Protective measures should be the precondition when install/adjust/replace hardware parts, for example, the camera.
- The stop functions in PickMaster® can never be used to replace A-stop/E-stop or any other safety related stops.

2 Welcome to PickMaster

2.1 Introduction

Overview

PickMaster 3 is a modular product for controlling ABB robots in picking applications through the IRC5 robot controller. It is a programmable application to perform pick and place operations of items. A vision system is used to find randomly placed items on conveying belts. It uses comprehensive graphical interfaces to configure powerful applications, where it can control multiple robots picking and placing sensor detected items on different conveying belts. It is designed to handle one or more cells in production.

The PickMaster 3 can be customized for some of the following special needs:

- With integrated vision it can be used for full random operation on a continuously moving conveyors and for absolute accurate positioning on indexed feeders or trays.
- Without vision recognition it can be used as a tool for the efficient production with guided product flows on multiple conveyors.
- For efficient quality inspection and product categorization alone or together with the position recognition.

Note

PickMaster is delivered with different hardware configurations. For more information, see the *Option Description* chapter in *Product specification - PickMaster® 3*.

2.2 Setting up the project

Introduction

PickMaster 3 consists of two parts, a line and a project. The line contains all the physical objects, such as robots, cameras, and conveyors. The project contains the items to be picked and the connections between work areas, items, and cameras.

In general, a line is created and configured once and things that do not change often are saved in a separate line file. There can be several lines configured. Many projects can be created to use one line.

The line

These objects and configurations are saved in the line.

- Cameras
- Robots and controllers
- Conveyors
- Conveyor and indexed work areas
- Camera calibration
- Base frame and camera tune

It is not possible to have more than one line open at the same time.

The project

These objects and configurations are saved in the project.

- Items
- Containers and patterns
- Position sources defining what, where and when to pick and place
- Vision models
- RAPID program for the robot controllers
- Work area settings and tune
- Robot settings

It is possible to have more than one project open at the same time, but they must all be configured to use the same line.

Related information

Configuring the line on page 59.

Configuring the project on page 89.

PickMaster terms on page 23.

2.3 The line and the project

Introduction to the line and the project

PickMaster consists of two parts that need to be set up. A line and a project. The line contains all the physical objects, such as robots, cameras, and conveyors. The project contains the items to be picked and the connections between work areas, items, and cameras.

This way, things that do not change often are saved in a separate line file. In general, a line will be created and configured once. There can be several lines configured. Many projects can be created to use one line.

The line

The following objects and configurations are saved in the line:

- Cameras
- Robots and controllers
- Conveyors
- Conveyor and indexed work areas
- Camera calibration
- Base frame and camera tune

It is not possible to have more than one line open at the same time.

> **Note**
>
> If a project is opened from the PickMaster user interface while another project is running that is using a different line, Pickmaster might fail and requires a restart of the application.

The project

These objects and configurations are saved in the project.

- Items
- Containers and patterns
- Position sources defining what, where and when to pick and place
- Vision models
- RAPID program for the robot controllers
- Work area settings and tune
- Robot settings

It is possible to have more than one project open at the same time, but they must all be configured to use the same line.

Related information

Configuring the line on page 59.

Configuring the project on page 89.

PickMaster terms on page 23.

2.4 Getting started with PickMaster

Prerequisites

Anyone working with installation of an ABB robot must be trained by ABB and have the required knowledge of mechanical and electrical installation work.

Setting up a PickMaster installation

Use this procedure to set up a PickMaster installation.

1. Install and configure robots and robot controllers. See the product manuals for the respective product, see *References on page 9*.
2. Install the PC cards. See *Setting up the PC on page 39*.
3. Install and configure conveyors. See respective manuals, see *References on page 9*.
4. Mount cameras, encoders, and other external devices.
5. Install all I/O, camera, and encoder cables. See *Electrical installation on page 38*.
6. Configure the software. See *Installation on page 25*.
7. Configure and calibrate the line. See *Configuring the line on page 59*.
8. Configure the project and set up a program. See *Configuring the project on page 168*.
9. If needed, configure Remote Integration Services. See *Configuring RIS on page 146*.
10. If needed, configure a User Hook. See *Customizing PickMaster with a User Hook on page 170*.
11. Test all programs.
12. Start production. See *Managing the robot in production on page 193*.

2.5 PickMaster terms

About these terms

Some words have a specific meaning when used in this manual. This manual's definitions of these words are listed below. Some of the terms are put in their context when describing a picking and placing process.

Term list

Words that have italic font style in the definition column are included in the term list and have their own definitions.

Term	Definition
Container	Defines which patterns to use and what *items* to use for each position in the patterns.
Item	The generic term for a specific object to be picked or placed in a PickMaster application.
Line	Description of static objects in a PickMaster installation, for example robots, *work areas*.
Position source	Defines how to generate *item* positions and which *work area* to send them to.
Project	Description of a PickMaster picking process applied on a specific *line*. Several projects can be defined for each *line*.
Work area	A PickMaster representation of pick and place areas.

This page is intentionally left blank

3 Installation

3.1 PickMaster software

Introduction to PickMaster software

The PickMaster software is delivered as a package. The package contains all the software required to install PickMaster and some additional software that are useful.

The user documentation for PickMaster and the vision system is available in the folder *User Documentation*. The calibration papers are also available in this folder.

Note

Any old version of PickMaster must be uninstalled before installing a newer version.

3.2 PickMaster license key

3.2.1 License key

Introduction to licensing

A license activation key provided by ABB must be installed and activated to run PickMaster.

Why a license?

Without a license the functionality in PickMaster is very limited.

Without an installed license:

- Projects cannot be transferred to the controller.
- Existing lines and projects can be opened but changes cannot be changed.

Information about the current license

To get information about the current license:

	Action
1	Start the PickMaster software.
2	On the **Help** menu, click **About PickMaster**. The **About PickMaster** dialog box appears, where you find the license information.

3.2.2 Activating a license key automatically over the Internet

Activating a license key

Use this procedure to activate a license key automatically over the Internet.

	Action
1	To start the licencing application, either use: • PickMaster, on the **Tools** menu, click **Licensing**. • **Windows Start** menu, point to **Programs**, **ABB Industrial IT**, **Robotics IT**, **PickMaster** and click **Licensing**.
2	In the licensing application, click **PickMaster License Activation Wizard**.
3	Under **Automatic Activation**, select **Activate PickMaster Over the Internet**.
4	Enter your 25 character Activation Key (xxxxx-xxxxx-xxxxx-xxxxx-xxxxx) and click **Next**. Your activation request will be sent to ABB over the Internet. If you are using a valid Activation Key that has not expired or exceeded the number of activations allowed, your PickMaster license will be activated immediately, and your PickMaster is ready for use when started next time.

3.2.3 Activating a PickMaster license manually

Introduction to manual activation

If the computer with PickMaster installed does not have an Internet connection, you must activate the license manually. This is done in three steps:

1. Create a license request file (`*.licreqx`).
2. Download a license file (`*.bin`) using an Internet connected computer.
3. Install the license file (`*.bin`).

Activating a license manually

Use this procedure to activate a PickMaster license manually.

	Action
1	To start the licensing application either use: • PickMaster, on the **Tools** menu, click **Licensing**. • **Windows Start** menu, point to **Programs**, **ABB Industrial IT**, **Robotics IT**, **PickMaster** and click **Licensing**.
2	In the licensing application, click **PickMaster License Activation Wizard**.
3	Under **Automatic Activation**, select **Step 1: Create a license request file** and click **Next**.
4	Enter your 25 character Activation Key (xxxxx-xxxxx-xxxxx-xxxxx-xxxxx) and click **Next**.
5	Click **Save Request**.
6	Type a name for a license request file (*.licreqx), browse to a suitable folder, and click **Save**.
7	Click **Finish**.
8	Use a removable medium, such as a USB device, to transfer the license request file to a computer with an Internet connection.
9	On the computer with internet connection, start the internet browser, and go to the link http://www.manualactivation.e.abb.com/ and follow the instructions to activate your license manually. You are instructed to browse for the saved license request file. The result will be a license file (*.bin) that you must save.
10	Transfer the license file to the PickMaster PC.
11	On the PickMaster computer, start the licensing application.
12	Under **Automatic Activation**, select **Step 3: Install a license file (*.bin)** and click **Next**.
13	Follow the wizard instructions. The PickMaster license will now be activated and the PickMaster installation ready to use.

3.3 PickMaster time synchronization service

Time synchronization service

PickMaster uses a time synchronization service to synchronize the time between the IRC5 robot controllers and the PCs running PickMaster. The synchronization is performed over the same network used for communication between PickMaster and the robot controllers.

Settings

The synchronization service is based on the precision time protocol (PTP), which in turn implements the IEEE 1588 standard. This protocol uses multicast messages over UDP/IP and requires that UDP port 319 and 320 are available (for both incoming and outgoing traffic). It is therefore necessary that any firewall is not blocking these ports. Please contact your system administrator to make sure that the proper configurations are performed.

The time synchronization service must be set to operate on the correct PC network interface port, that is, the network port which communicates with the robot controllers. This can be configured during the installation of PickMaster or by selecting **Options** on the **File** menu.

Related information

Setting PickMaster options on page 204.

3.4 System requirements

3.4.1 Requirements

Software requirements

Following are the software requirements:

Software requirements
Windows 10 (64-bit version).
RobotWare 5.12, RobotWare 5.60, RobotWare 6.00, and further releases of RW 6.

Hardware requirements

Following are the prerequisites for installing the PickMaster:

- A computer that meets or exceeds the system requirements as specified by RobotStudio.
- A log on account with administrator rights on the computer.

Hardware requirements
CPU: 2.0 GHz or faster processor, recommended is multicore processor. Color vision is especially resource demanding.
Memory: Minimum 8 GB or more.
Disk space: More than 5 GB free space, solid state drive (SSD) recommended.
The Gigabit Ethernet vision system requires one PCI Express x4 slot, full height or low profile. It is also compatible with PCI express x8 and x16. **Note** • The Gigabit Ethernet vision system is not compatible with PCI express x1 or with the special type of x16 slots intended only for graphics cards. • Single Vision System (frame grabber) requires two free PCI slots per vision system. Analog multi vision system and Digital Vision system requires one PCI slot per card. • Ethernet network board is required for controller communication. • If RIS field bus card is used, an additional free slot is required.

3.4.2 Ethernet switch

Overview

An Ethernet switch is used to connect the PC with multiple robot controllers. It is recommended to use an unmanaged industrial switch with a communication speed of 100 Mbit/s or higher. Switches that implement the 1588 PTP protocol have been known to interfere with the robot controller communication and should not be used.

3.4.3 Vision system

Overview

PickMaster can acquire images and generate targets by using cameras that communicate over Ethernet. An Ethernet network (network interface card, cables, switches) is used for communication between the cameras and the PickMaster PC. Preferably the power voltage to the Ethernet camera is supplied from a separate source that is independent of the robot controller.

Vision system requirements

The supported network card for Ethernet camera communication is GigE network card DSQC1083 (3HAC078753-001). Other network interface cards can work, but have not been tested.

A Cognex USB license is required for the Gigabit Ethernet vision system. The USB stick must be connected when PickMaster is running.

The maximum number of cameras that can be used is ten.

Insert the vision network card in a free compatible PCI-express slot (PCI-express x4, x8, or x16).

Note

Vision Analyzer files (`.pmv`) saved with PickMaster 3.31 or PickMaster 3.32 cannot be opened using the newer versions of PickMaster. Older versions can be opened.

Color vision support

Color vision support has been added from PickMaster 3.31 and has the following features:

- connectivity for color cameras
- white balance calibration
- color filter configuration

Note

This allows you to define colour filters that will run as a prestep to PatMax and Blob. The filter is available in Standalone, alignment, and sub inspection modes.

The vision system has been tested with `Basler acA1440-73gc` and `Basler Scout ScA1300-32gc` camera.

Trigger strobe cables

There are 4 types of cable used. Trigger strobe connections for those are as below.

Cable	ID	Description
New scout replacement cable (10 m)	2000034085	Power-I/O PLC+ Cable HRS 12p/open, 10m
Old Scout cable (10 m)	2000026632	Power-I/O Cable, HRS 12p, open, 10 m

Continues on next page

Cable	ID	Description
Old Scout cable (10 m)	2000022909	Power-I/O Cable, HRS 12p/open, 10 m
Ace camera cable	2000034084	Power-I/O PLC+ Cable 6p/open, , 10 m

Trigger strobe connection for 2000034085 (Basler Scout camera)

The following table describes the physical interface for trigger/strobe/power connection to the Basler Scout camera.

For further details about how to connect the camera, see the circuit diagram.

Wire Pair	Pin Number	Wire Color	Scout GigE	Function
1	1	White	Camera ground	0V(CamPower-)[i]
1	2	Green	Camera ground	0V(CamPower-)
2	3	Pink	Opto in 1	Trigger[ii]
2	(5)[iii]	Grey	Opto in 1 ground	0V (Cam I/O-)[iv]
3	4	Red	Opto in 2	Not used
3	(5)	Blue	Opto in 2 ground	Not used
4	6	Violet	Opto out 1	Strobe[v]
4	(10)	Black	Opto out 1 VCC	24V (Cam I/O +)
5	7	Red/Blue	Opto out 2	Not used
5	(10)	Grey/Pink	Opto out 2 VCC	Not used
6	8	Brown	Camera VCC	24V (CamPower +)
6	9	Yellow	Camera VCC	24V (CamPower +)
7	(10)	White/Green	Opto Out 3 VCC	Not used
7	11	Brown/Green	Opto Out 3	Not used
8	(10)	White/Yellow	Opto Out 4 VCC	Not used
8	12	Yellow/Brown	Opto Out 4	Not used

i 0/24V for powering the camera. Preferably supplied from a source that remains turned on even if the robot controller is shut down.
ii Input signal that orders the camera to acquire an image.
iii Pin number inside parenthesis "(X)" means that the wire is connected internally to pin "X"
iv 0/24V for the I/O system of the camera.
v Output signal indicating that the camera has acquired an image.

Old Trigger strobe connection for 2000026632 and 2000022909 (Basler Scout camera)

Pin Number	Wire Color	Scout GigE	Function
1	White	Camera Power Ground	0V(CamPower-)
2	Green	Camera Power Ground	0V(CamPower-)
3	Blue	I/O Input 1	Trigger
4	Red	I/O Input 2	Not used
5	Gray	I/O Input Ground	0V (Cam I/O-)
6	Black	I/O Output 1	Strobe
7	Violet	I/O Output 2	Not used

Continues on next page

3.4.3 Vision system
Continued

Pin Number	Wire Color	Scout GigE	Function
8	Brown	Camera Power VCC	24V (CamPower +)
9	Yellow	Camera Power VCC	24V (CamPower +)
10	Pink	I/O Output VCC	24V (Cam I/O +)
11	Gray/Pink	I/O Output 3	Not used
12	Red/Blue	I/O Output 4	Not used

Trigger strobe connection for 2000034084 (Basler Ace camera)

The following table describes the physical interface for trigger/strobe/power connection to the ace camera.

Power-I/O Cable HRS 6p/open, twisted, 10 m - IOs / Power Cables Cable for power supply and trigger of opto coupled I/Os of Basler ace GigE cameras at a length of 10 meters.

The cable has an HRS 6-pin connector on the camera side. The other end is open so that the cable can be shortened to match individual requirements.

Wiring information:

Pin Number	Wire Color	Ace GigEg	Function
1	Brown	Camera Power	24V (CamPower +)
2	Pink	Opto-isolated IN (Line1)	Trigger
3	Green	Not connected	0V (Cam I/O-)
4	Yellow	Opto-isolated OUT (Out1)	Not used
5	Gray	Opto-isolated I/O Ground	0V (CamPower-)
6	White	Camera Power Ground	0V (CamPower-)

Note

There is no strobe output on ace camera, so we need to have a jumper between TrigOut and SyncIn

3.4.4 Camera requirements

Mounting

The cameras must be mounted in a very stable way to avoid vibration and other dynamic movement. The cameras can be mounted in four different orientations but it is recommended to have the "belly" towards the robot and the belly side perpendicular to the work area's x-axis.

Lighting

Even lighting of the image area is very important to obtain reliable results.

Other camera requirements

If the camera is mounted on a moving conveyor then it is necessary to have a progressive scan camera (non-interlaced).

We recommend using a camera that supports electronic shutter control. Then it is possible to set the exposure from PickMaster, otherwise the exposure time must be manually set on the camera.

Camera configuration

Some cameras will need manual configuration to fulfill the above conditions. For detailed information about camera settings, see *Cognex manual* and *PickMaster Release Notes*.

For specific information about Basler Scout Gigabit Ethernet cameras, see *References on page 9*.

Recommendation for lenses

When planning a cell it is important to choose a suitable camera/lens setup that gives an appropriate field of view (FOV).

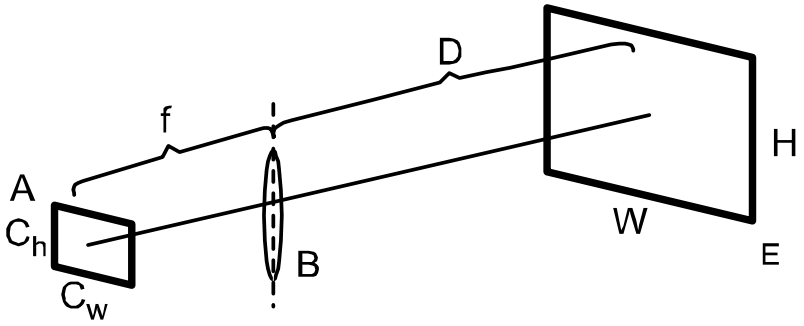
The FOV of a camera is determined by three factors:

- The distance between the camera and the scene.
- The focal length of the lens.
- The size of the camera's sensor chip (normally specified as the distance of the diagonal of the chip, expressed in inches).

Continues on next page

3.4.4 Camera requirements
Continued

The graphic below shows the geometry of the optical setup.



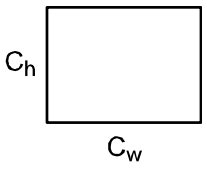
xx0900000550

A	Sensor chip
B	Lens
C	Chip height (mm)
C	Chip width (mm)
D	Distance from lens to scene (mm)
E	Scene
f	Focal length of camera (mm)
H	Scene height (mm)
W	Scene width (mm)

To select a suitable lens, measure the distance between the camera and the items (D), and the size of the image area (W*H).

To calculate the appropriate focal length of the lens:

- If the height of the image area is most important: `f = (D/W)*Cw`
- If the length of the image area is most important: `f = (D/H)*Ch`



xx0900000565

The table below lists the width and height of some common sensor chip sizes, expressed in millimeters.

Sensor chip size (inch)	C_h (mm)	C_w (mm)
1/4"	2.4	3.2
1/3"	3.6	4.8
1/2"	4.8	6.4
2/3"	6.6	8.8

A shorter focal length gives a wider field of view, that is the returned value is the maximum focal length to obtain the specified W and H.

Continues on next page

Example: lens calculation

This example is based on a 1/2" sensor chip, such as the Basler Scout 1390-17gm.

- The FOV should cover a conveyor belt with a width of 500 mm.
- The minimum height of the FOV is not restricted.
- The distance between the camera and the conveyor is 800 mm.
- The camera is mounted with the belly facing the robot (PickMaster default).

Because the width of the conveyor determines the minimum FOV the required focal length is calculated using:

```
f = (D/W)*Cw
```

Enter the known data, C_w is 6.4mm (see graphic above).

```
f = (800/500)*6.4 = 10.24 mm
```

The resulting height H of the FOV is calculated as:

```
H = D*CH/f = 800*4.8/10.24 = 375 mm
```

Alternative with increased height

To increase the height of the FOV (H), the camera can be rotated 90° so that the height dimension of the sensor chip (4.8 mm) is aligned with the width dimension of the conveyor. The width dimension (6.4 mm) is aligned with the x-axis of the conveyor.

```
f = (800/500)*4.8 = 7.68 mm
```

The resulting height H of the FOV is now:

```
H = 800*6.4/7.68 = 666 mm
```

Normally lenses are available in some standard focal lengths. Choose a lens that has a focal length shorter than the calculated value to be sure to capture the entire scene.

3.5 Electrical installation

3.5.1 Installing electrical components for PickMaster

Introduction to electrical installation

The order in the electrical installation procedure has no relevance, but it is easier to configure the networks if all the electrical components are connected.

Installing electrical components for PickMaster

Use this procedure to install the electrical components for PickMaster.

1. Set up the PC and connect it to the controller, see *Setting up the PC on page 39*.
2. Connect the cameras, see *Connecting cameras on page 40*.
3. Connect all I/O signals, see *Connecting I/O signals on page 41*.
4. Connect the encoders, see *Connecting encoders on page 43*.
5. Configure the network, see *Configuring the networks on page 45*.
6. Set up the robot controller, see *Setting up the robot controller on page 50*.
7. Proceed with configuring lines, projects, etc. See the chapter *Configuration on page 57*.

3.5.2 Setting up the PC

Introduction to PC setup

The computer must be connected to the controller network and use a specific vision network card that is included in the delivery.

Setting up the PC

Use this procedure to set up the PC.

1. Install the network interface card (NIC).
2. Insert the USB device.

   The USB device contains the Cognex license for the Gigabit Ethernet vision system and must be connected to the computer at all times when running PickMaster.
3. Connect the PC to the controller network.

Related information

How to configure the network is described in section *Configuring the networks on page 45*.

How to install PickMaster is described in section *Installation on page 25*.

The PC requirements are described in *System requirements on page 30*.

3.5.3 Connecting cameras

Introduction to camera connections

The camera does not receive power voltage through the Ethernet cable. A separate connection provides power and I/O functions, this is the power/trig/strobe cable.

We recommend using an external power supply for the Gigabit Ethernet cameras. This way, they will receive power regardless if the robot controller is turned on or not. If the camera is supplied with power directly from the robot controller it will shut down when the controller is turned off. PickMaster can not reconnect to a camera that has been shut down and restarted. This means that if PickMaster is running when a controller that serves as a camera power supply is shut down, PickMaster must be restarted after the controller has been switched on again. This problem is avoided by using an external power supply.

The schematics of how the trigger strobe and power wires from the camera must be connected to the robot controller I/O board can be seen in the circuit diagrams, see *Circuit diagrams - 3HAC024480-008*. Detailed information about avoiding EMI/ESD problems is described in *Avoid_EMI_ESD_in_camera_installations*, see *References on page 9*.

Prerequisites

Make sure all power is switched off before connecting cameras.

Connecting the cameras

Use this procedure to connect the cameras.

1. Connect the Ethernet cable with screw connector to the camera.
2. Connect the other end of the Ethernet cable to the PC or the switch (if used).
3. If a switch is used, connect the switch to the PC.
4. Connect the power wires of the power/trig/strobe cable to the external 24V power supply

   In case no external power supply is used, connect to the controller.
5. Connect the trig/strobe wires of the power/trig/strobe cable to the robot controller.

Note

If PickMaster is shut down and restarted quickly, and with several Gigabit Ethernet cameras, the Gigabit Ethernet performance driver may not be loaded properly for some cameras. The symptom is that the camera for which the driver is not loaded may occasionally fail to acquire an image, if the system is stressed. This can be avoided by waiting for 15 seconds between shutting down and restarting.

Related information

IRC5 Circuit diagram - 3HAC024480-008.

3.5.4 Connecting I/O signals

Introduction to I/O connections

The PickMaster concept consists of a number of I/O components that need to be connected physically.

Robot controller I/O board

At least one standard DI/DO board is required. Encoder boards are needed for conveyor tracking.

The encoder boards are delivered with a standard address that can differ from the I/O configuration. This address can be changed.

For further information about how to read the encoder board address, see the product manual for the controller, see *References on page 9*.

Prerequisites

Make sure all power has been switched off.

Connecting the I/O signals

Use this procedure to connect the I/O signals.

1. If conveyors are used, connect each conveyor controller to the standard DI/DO board for control from PickMaster.

   The drawings in *Circuit diagrams - 3HAC024480-008*, uses ACS 301-1P6-3 as conveyor controller, but other conveyor controllers can be used. The same applies to the encoder used.
2. Connect the trig/strobe wires of the power/trig/strobe cables from the cameras to the robot controller.
3. Connect the I/O cables from any external tool signals to the robot controller.
4. Connect the I/O cables for other external devices, such as sensors to the robot controller.
5. Connect the encoders, see *Connecting encoders on page 43*.

I/O connections

The trigger strobe loop enables very precise synchronization between the robot controller and the image acquired. The I/O port of the Gigabit Ethernet camera closes this loop.

To be able to use more than one connection in input number 9 (StartSig) on the encoder board we recommend using diodes, for example HER105/Taw diode 1A 400V DO41 (the diodes are not supported by ABB). This will eliminate any possibilities of reverse currents.

When connecting a camera to multiple robot controllers it is important to consider how the system should work if one of the controllers is turned off. We recommend using an external 24V power supply to power the cameras. This way the cameras will have both power and I/O regardless if the controllers are turned off.

Continues on next page

3.5.4 Connecting I/O signals
Continued

Related information

IRC5 Circuit diagram - 3HAC024480-008.

I/O signals on page 57.

Predefined I/O signals on page 58.

Configuring the conveyor and the conveyor work area on page 63.

Configuring an indexed work area on page 66.

3.5.5 Connecting encoders

Introduction to encoder connections

The encoder board is mounted in the controller cabinet on delivery.

One encoder can be used for more than one encoder board. To connect it to up to 10 encoder boards, use a signal enhancer.

Prerequisites

If less than five encoder boards are used, then the number of installed digital I/O boards must be defined. One simulated I/O board is defined by default.

Connecting the encoders

For the details about connecting the encoders, see the manual for the encoder, and *Circuit diagrams - 3HAC024480-008*.

One or more encoders measure the position of a conveyor. Usually an encoder is placed close to an identification and operation area in order to minimize errors caused by the elasticity and inaccurate guidance. The encoder measurement sample rate is configured in the robot controller and by default set to 20 ms.

The interface board DSQC 377 handles one encoder. It is possible to connect one encoder to two interface boards.

The encoder should be of type Open collector PNP output, two phases with 90 degrees phase shift, voltage 10-30V, and current 50 - 100 mA which is normally supplied by 24 VDC from DSQC 377.

The pulse ratio from the encoder should be in the range of 5,000 - 10,000 pulses per meter of conveyor motion (See wiring diagrams in the PickMaster User's Guide). The pulses from channels A and B are used in quadratures to multiply the pulse ratio by four to get the counts. This means that the control software will measure 20,000 - 40,000 counts per meter for an encoder with the pulse ratio given above.

Reducing the number of measured counts below 20,000 may reduce the accuracy of the robot tracking. Also increasing the number of measured counts beyond 40,000 may have no significant effect as the inaccuracies in robot and cell calibration will be the dominating factors for accuracy.

Note

20,000 counts per meter is the recommended value for picking robots. A value smaller than 5,000 counts per meter is not recommend as it may reduce the robot tracking accuracy significantly.

The encoder board (DSQC 377) can handle a minimum of 500 and a maximum of 50,000 counts per meter. The maximum frequency is 50 kHz (may occur with high conveyor speed and many pulses per meter).

Continues on next page

3.5.5 Connecting encoders
Continued

The encoder should be connected to the robot by a screened cable to reduce the noise.

Note

If the cable is long, the inductance in the cable may produce spike pulses on the encoder signal, which may, over a period of time, damage the opto-couplers in the encoder board. The spike pulses can be removed by installing a capacitor between the signal wire and ground for each of the 2 phases. The capacitors should be connected to the terminal board where the encoder is connected and not on the DSQC 377.

The number and speed of the conveyors are CPU consuming factors on the robot controller. Maximum six conveyors may be attached. Maximum four encoder boards may be connected inside the cabinet together with one I/O board.

Related information

IRC5 Circuit diagram - 3HAC024480-008.

3.5.6 Configuring the networks

Introduction to the controller network

The PickMaster and the robot controller communicate through Ethernet. If you have problems in connecting to the network, contact the local network administrator.

Note

The PickMaster must be connected to the:

- LAN port on the controller if using RW 5.xx.
- WAN port on the controller if using RW 6.xx.

Note

Do not connect PickMaster to the service port.

Configuring the controller network

If a new local area network (LAN) is created specifically for PickMaster the following settings can be used.

- Use static IP numbering with different addresses for both the computer and the robot controller.
- IP addresses: 192.168.1.X (where X is between 1 and 253).
- Subnet mask: 255.255.255.0
- Gateway: 192.168.1.254
- DNS: N/A.
- Wins: N/A.

Note

The robot controller has a service Ethernet card configured with an IP address (192.168.125.1). Therefore, the same subnet (192.168.125.X) must not be used for the standard LAN Ethernet card.

For more information, see *the Windows documentation and the product manual for the robot controller* to set up the IP configuration.

Prerequisites for vision networks

The vision network settings must be configured similar to the robot controller network settings.

Use a separate network for the vision system, that is controllers and cameras cannot be connected to the same network port on the PC.

To use more cameras than the number of available Ethernet ports on the PC, use one or two Ethernet switches. Do not connect two switches in series to prevent the risk of latencies in communication.

The maximum number of cameras that can used with one PC is 10. Distribute them evenly on the dedicated vision network ports on the PC. Use normal straight CAT6

Continues on next page

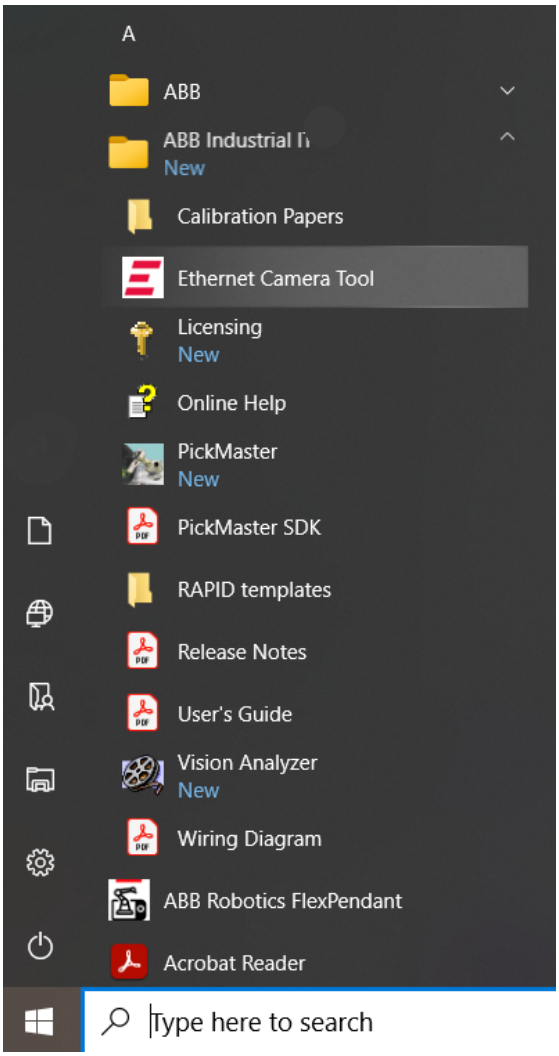
cables between the PC and the switch, and cables with fastening screws between switch and camera.

Configuring the vision network

Use this procedure to configure the vision network.

1 Assign each camera with its own IP-address. The same rules apply as for other Ethernet networks, that is each camera and vision network card must have a unique IP address, and be located on the same subnet. The communication with cameras and controllers should be separated on different subnets. See *Example of suitable network architecture on page 49*.

2 Configure the IP addresses for the cameras using Cognex's *Ethernet Camera Tool* (available on the Windows Start menu in the PickMaster folder). It can be used to set IP addresses of both cameras and network interface cards.



xx2300000275

3 When all cameras are configured, install the *Performance Driver* for Gigabit Ethernet vision for each port, see steps 4-6.

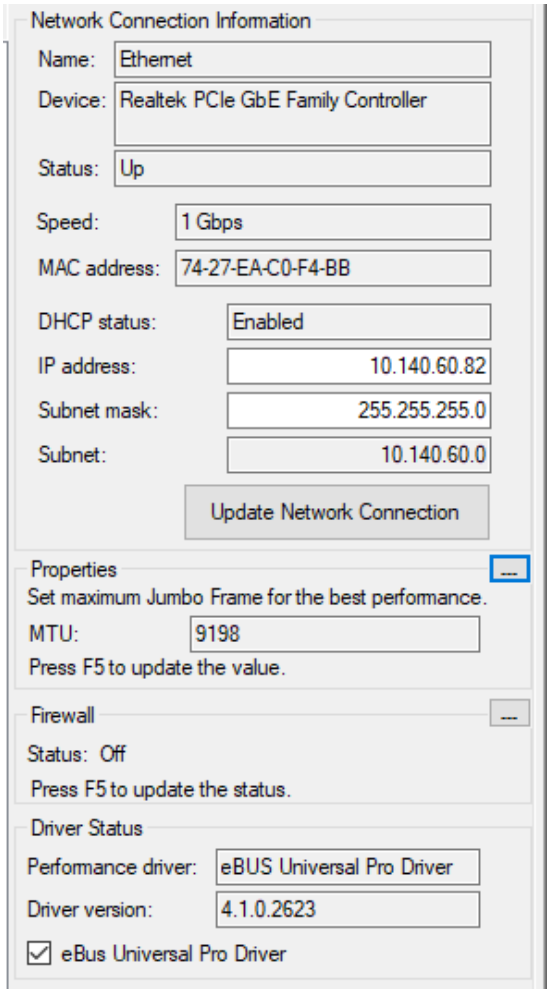
Continues on next page

4 In the **Ethernet Camera Tool**, for each vision network port in the tree view, do the following settings:

a In the **Properties** section set the value of MTU at around 9000. If the MTU value is around 1500, it means that the Jumbo frames is not set.

To set the Jumbo frames:

I Click ...



xx2200001607

The **Ethernet Properties** window is displayed.

II Click the **Networking** tab.

III Click **Configure**.

The properties window is displayed.

IV Click the **Advanced** tab.

V Select **Jumbo Frame** from the **Property** list.

VI Select a value as high as possible from the **Value** drop-down list.

VII Click **OK**/**Apply** until you are back in the **Ethernet Camera Configuration** tool.

VIII Press F5 to refresh the values in the window.

IX Verify that the MTU value is about 9000.

Continues on next page

b Select the **eBus Universal Pro Driver** check box. A warning about installing unsigned software appears.

c Click **OK**.

5 Reboot the PC when the installation is complete for all the vision ports.

6 Start the **Ethernet Camera Tool** and verify that the performance driver has been successfully installed for each vision network port. Also verify that the Jumbo frames MTU value is set to about 9000.

Note

In case you face any issue during image capture, modify the following network configuration on the ethernet where the camera is connected:

- In the **Ethernet Camera Tool** for vision network port in the tree view Click ...

  The Ethernet Properties window is displayed.

  - In the **Networking** tab, clear all the check boxes listed under **This connection uses the following items** except **eBUS Universal Pro Driver** and **Internet Protocol Version 4 (TCP/Ipv4)**.
  - Click **Configure** and then choose the **Advanced** tab.
    # Select the **Receive Buffers** property and choose the highest possible value in the **Value** list.
    # Select the **Interrupt Moderation Rate** property and choose the value as **Extreme**.

Note

Running the Ethernet Camera Tool and PickMaster at the same time may result in unpredictable behavior. To avoid this, use only one of the programs at a time.

CAUTION

Running camera traffic and controller traffic on the same network can cause serious communication failure.

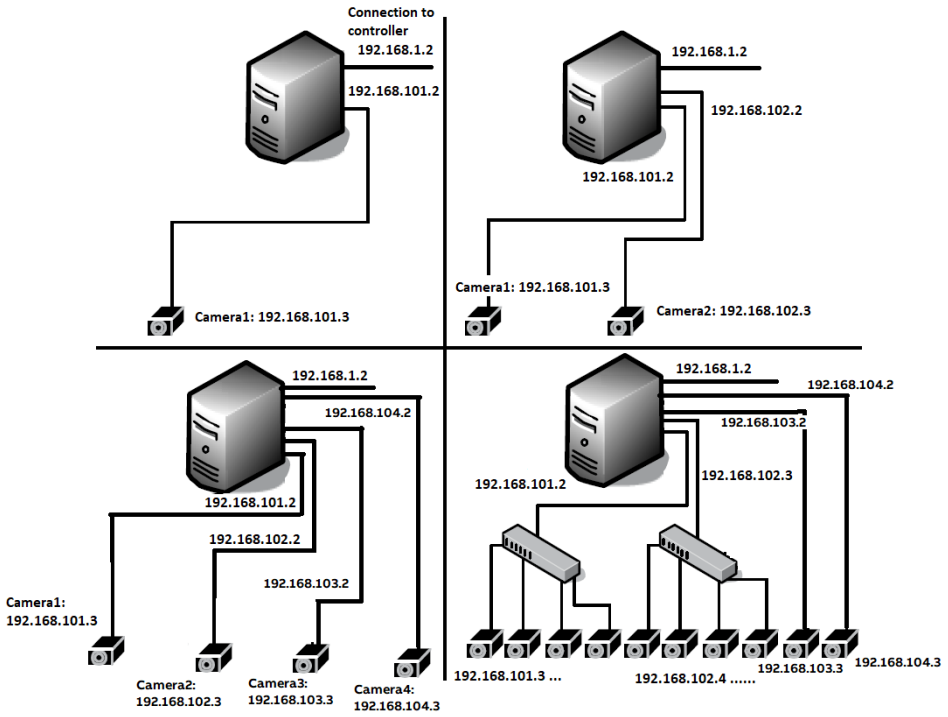
Continues on next page

Example of suitable network architecture

The following example provides a suitable network architecture.

- Use static IP numbering with different addresses for both the computer and the camera(s).
- IP addresses of Port #1 and the cameras connected to it: 192.168.101.X (where X is between 1 and 253).
- IP addresses of Port #2 and the cameras connected to it: 192.168.102.X (where X is between 1 and 253).
- Subnet mask: 255.255.255.0
- Gateway: Not Needed.
- DNS: N/A.
- Wins: N/A.

See the following example of camera network topologies. The cameras are evenly distributed on the two dedicated vision network ports on the PC.



xx0800000319

Note

Changes made to the camera settings outside PickMaster will not be applied until PickMaster is restarted. This means that if a camera is restarted (power on/off) or a camera's IP address is changed, then PickMaster must be restarted to function properly. Therefore, PickMaster and the *Ethernet Camera Tool* program should not be run simultaneously, to avoid unpredictable behavior. Instead, shut down PickMaster before making changes, then start PickMaster after changes are saved.

3.5.7 Setting up the robot controller

RobotWare

PickMaster supports IRC5 robot controllers. RobotWare is installed on the robot controller. The option *Prepared for PickMaster* is required to run PickMaster.

For more information see the product manual for the controller, see *References on page 9*.

System parameters

The number of conveyors must be specified in the system parameters. Some other parameters must also be defined, such as motion, process, and encoder I/O parameters for the conveyors.

System parameters can be changed using the FlexPendant or RobotStudio.

I/O signals

How to configure I/O signals and boards is described in the section *I/O signals on page 57*.

The predefined I/O signals are described in the section *Predefined I/O signals on page 58*.

Related information

Product manual for the controller, see *References on page 9*.

Technical reference manual - System parameters.

Six axes robot configuration on page 52.

3.5.8 Optional robot configuration

Introduction

Fewer modifications can be done on the system parameters

Topic Process

The following parameter can be modified in the topic *Process*. It belongs to the type *Conveyor systems*.

Parameter	Description
maximum distance	Defines the standard tracking distance of a conveyor work object before it is switched to a new work object. This is by default set to 20000mm. The work object switch is done automatically and fast but may steal some process time for a high speed picking application. Increasing the value may improve the cycle time slightly.

3.5.9 Six axes robot configuration

Introduction to six axes configuration

When using PickMaster with a six axes robot, some modifications must be done in the system parameters.

Topic Process

The following three parameters can be modified in the topic *Process*. They belong to the type *Conveyor systems*.

Parameter	Description
Start ramp	This is the correction start filter ramp that is used when connecting to a moving conveyor. This is by default set to 5 (steps). Tune this parameter if higher accuracy is needed. A lower value gives better accuracy but the manipulator may jerk when connecting to the moving object.
Stop ramp	This is the correction stop filter ramp that is used when disconnecting from a moving conveyor. This is by default set to ten (steps). Tune this parameter to eliminate manipulator jerks when leaving the moving object. A lower value gives better accuracy when leaving the conveyor.
Adjustment speed	The speed (in mm/s) at which the robot should catch up to the conveyor. The general recommended value is 130% of the conveyor speed. As minimum, the value should be more than 100% with some margin. If the robots speed is very fast compared to the conveyor speed, a further increase of the value is often necessary. If the value is set too low, robot movements may become jerky or the conveyor tracking accuracy may become reduced. On the other hand, if the value is set too high, the drive system may become overloaded, causing motion supervision errors. Generally, the maximum recommended value is 200%. For IRB360 in applications with high robot speed, the maximum recommended value is 500%.

Topic Motion

The following two parameters can be modified in the topic *Motion*. They belong to the type *Robot* respective type *Motion Planner*.

Parameter	Description
`Use Six Axes Corvec`	This adds position adjustment on six axes and therefore the orientation of the tool is exact. Otherwise the correction is only made on axis 1, 2, and 3, and the orientation accuracy is lower. This parameter can only be set on six axes robots. If the parameter `Use Kinematic Corvec` is set then the parameter `Use Six Axes Corvec` is not needed and should not be used.

Continues on next page

Parameter	Description
`Path Step Adjustment Type`	Adds the string `TWO_STEPS` to change the motion planning to use two steps interpolation. This can be used if the error message *50050: Position outside reach* is displayed without a reason. This parameter can cause motion faults when passing singularities and stretching out axis 3, when the lower and the upper arm are aligned. **Note** This parameter is obsolete in RobotWare 6.0 or later.

3.5.10 Installation of DSQC2000

For details about DSQC2000, refer *Application manual - Conveyor tracking*.

3.6 Application login window

Overview

An application login window can be activated to restrict the access to PickMaster for different users. To activate the login window, Click **File** -> **Options** -> **Enable User Authentication**.

The login window is displayed if the **Lock workplace** check box is selected in the toolbar and when the application is restarted.

A default admin user, with full access, is provided with the following credentials:

- User: admin
- Password: password

This user can be modified. Other users can be created. At least one admin user is required.

This page is intentionally left blank

4 Configuration

4.1 Configuring I/O

4.1.1 I/O signals

Configuring I/O signals

I/O signals are configured using RobotStudio or the FlexPendant. Then they can be used from PickMaster.

The predefined signals can be used without modifications. Edit the predefined signals or add additional signals if needed.

> **Note**
>
> The maximum name length for a PickMaster work area signal is 15 characters.

Related information

Predefined I/O signals on page 58.

Operating manual - RobotStudio.

Operating manual - IRC5 with FlexPendant.

4.1.2 Predefined I/O signals

Predefined I/O signals

The following I/O signals are predefined on delivery. Some of them are used or referenced to when configuring the line. The encoder signals are described in *Application manual - Conveyor tracking*.

I/O signal name	Description
`diX_1`	Digital input signals for custom use, such as generating I/O triggered position or checking a gripper pressure switch.
`doStartCnvX`	Digital output for starting/stopping conveyors.
`doTrigVisX`	Digital output for triggering an image acquisition. This signal is used by PickMaster to order the camera to acquire an image.
`doManSyncX`	Digital output used for triggering predefined positions in a conveyor work area. This output should be connected to the `StartSig` (input 9) on the corresponding encoder board.
`doVacuumX`	Digital output for activating vacuum. For example, for gripping a product. The output signal is set when an item shall be attached to the tool. **Note** The signal is controlled from the RAPID program. In simulation, the RAPID `triggdata SimAttachX` controls when the signal is set. On a real robot, the RAPID `triggdata VacuumActX` controls when the signal is set.
`doBlowX`	Digital output for activating air blow. For example, for releasing a product gripped by the robot. The output signal is set when an item shall be detached from the tool. **Note** The Release signal is controlled from the RAPID program. In simulation, the RAPID `triggdata SimDetachX` controls when the signal is set. On a real robot, the RAPID `triggdatas VacuumRevX` and `VacuumOffX` controls when the signal is set/pulsed.
`goVacBlowX`	Digital I/O group containing *doVacuumX* and *doBlowX*.

4.2 Configuring the line

4.2.1 Introduction

Overview

A line describes the configuration of the physical line objects, that is the cameras, conveyors, work areas, and robots. As long as the objects in the physical line has not changed there is no need to edit the line.

The line view shows a representation of the line. The orientation and placing of the objects in the line view has no relevance for the configuration. It is however convenient to place the objects so that the line view looks like the physical line.

To add or edit objects in the line, right-click the line view. A line is saved as a `.pmline` file.

> **Note**
>
> To work with a line, all projects must be closed.

To configure a new line:

1. On the **File** menu, select **New Line**.
2. Add a robot controller, see *Configuring the robot controller on page 60*.
3. If needed, add a conveyor, see *Configuring the conveyor and the conveyor work area on page 63*.
4. If needed, add a conveyor work area, see *Configuring the conveyor and the conveyor work area on page 63*.
5. If needed, add an indexed conveyor, see *Configuring an indexed work area on page 66*.
6. If needed, add a camera, see *Configuring the camera on page 68*.
7. If needed, add an external sensor, see *Configuring an external sensor on page 71*.
8. Calibrate the line, see *Defining the parameter Counts Per Meter on page 73*, and *Calibrating the camera on page 75*.
9. Verify the calibrations, see *Verifying the calibrations on page 86*.

> **Note**
>
> It is possible to open a new project while another project is running only when the projects have the same line. Any RIS open or start commands will be rejected with the return code "Configuration error".

4.2.2 Configuring the robot controller

Introduction to the robot controller

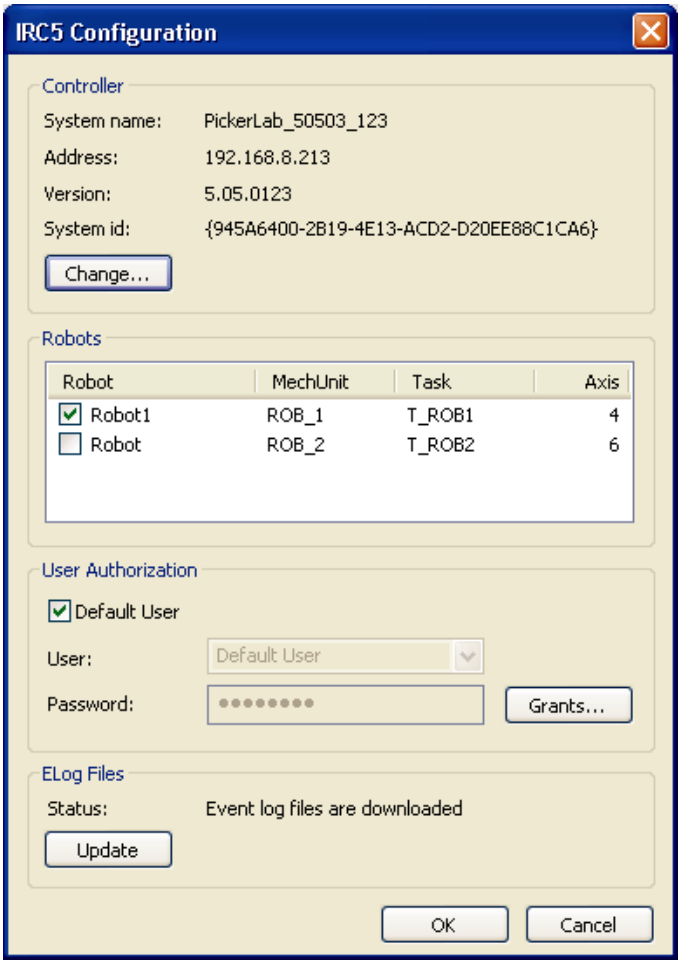
When a robot controller is created in the line, both a controller and a robot object will appear in the line view. For a MultiMove controller, multiple robot objects are created. At least one robot must be defined for the line. The line can have up to ten robots.

Prerequisites

The robot controller must be installed and available on the network.

If RobotWare has been upgraded on the robot controller then the controller must be re-selected in the line.

Illustration IRC5 configuration



en0900000528

Continues on next page

Section	Description
Controller	This section displays the information about the selected controller. Click the **Change...** button to change the robot controller. **Note** If the RobotWare version of the robot controller is changed, the controller is assigned with a new system identification. Hence, the controller must be reselected if the RobotWare is changed.
Robots	Displays a list of configured robots on the selected controller. Select the corresponding check box to use a robot. The name of the robots can be changed in this section.
User Authorization	An IRC5 controller is always delivered with a **Default User** with full access, but in this section you can also select any configured user for the controller.
Elog Files	To display log events from the controller correctly, the error texts must be downloaded from the controller. The files are saved in a subfolder called `Irc5Files` and downloaded only once for each controller. The required files are downloaded automatically when needed but they can be downloaded manually by clicking on the **Update** button.

Configuring the robot controller

Use the following procedure to configure the robot controller in the line.

1. In the line view, right-click and select **Add Robot Controller**.
2. Select a controller from the list and click **OK**.

   The controller configuration window is opened.
3. In the **Robots** section, select the checkboxes to select the robots to use in the line.
4. In the **User Authorization** section, clear the checkbox if other than the default UAS settings are used.

   The User Authorization System is described in *Operating manual - RobotStudio*.
5. In the **Elog files** section, click the **Update** button to download the updated error texts after an upgrade of the robot controller.

   Error texts are downloaded automatically when installing the robot system. The files are saved in *Irc5Files*subfolder.
6. Click **OK**.

Robot base frame configuration for ceiling-mounted IRB

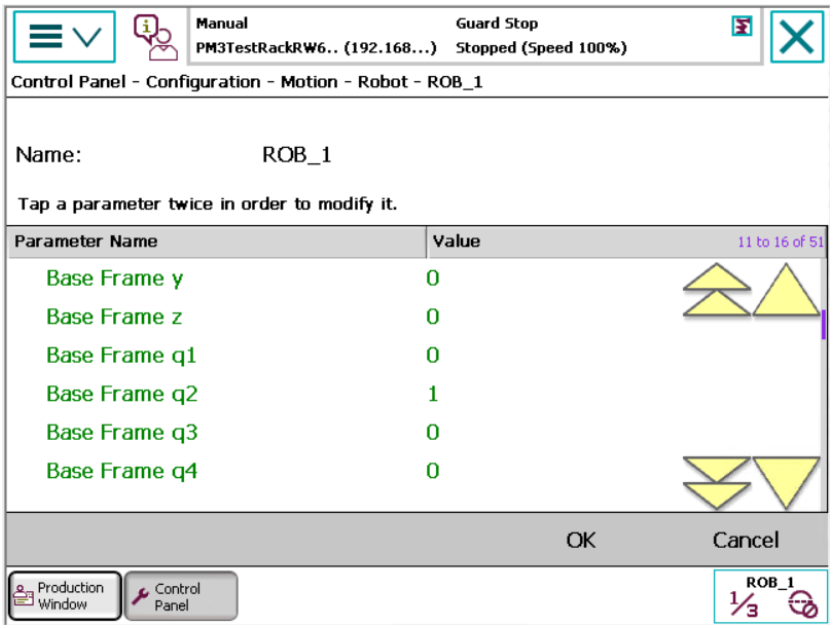
To use a ceiling-mounted IRB with PickMaster, the base frame of the robot must be set up correctly with the correct quadrennia coordinates (q1, q2, q3, q4).

Continues on next page

4.2.2 Configuring the robot controller
Continued

Use the following procedure to configure the Robot base frame for ceiling-mounted IRB:

1 On the FlexPendant tap **Control Panel** > **Configuration** > **Motion** > **Robot** > **Robot***.



xx2200001081

2 Edit the parameters for "BaseFrame q1, q2, q3 and q4" to the values 0,1,0,0.

3 Tap **OK**.

4 Restart the controller.

Note

For IRB360 family this happens automatically for RobotWare system 5.06 or later.

4.2.3 Configuring the conveyor and the conveyor work area

Configuring the conveyor

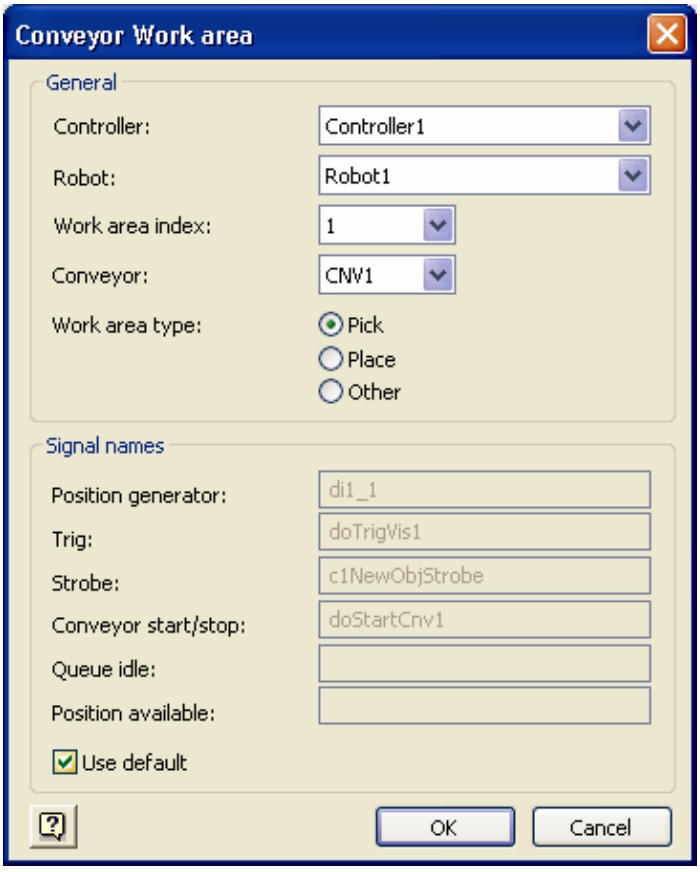
The conveyor object has no configuration parameters. But each conveyor must be configured in the robot controller so that the value of the system parameter *Counts Per Meter* for the encoder corresponds to the distance moved by the conveyor. See *Defining the parameter Counts Per Meter on page 73*.

The rest of the configuration for the conveyor is done in the conveyor work area.

Introduction to the conveyor work area

The conveyor work area is the area on the conveyor where the robot picks or places items. There is one encoder board dedicated for each conveyor work area. A robot usually has one conveyor work area on a conveyor but can have several if the robot has more than one encoder board for that conveyor.

Illustration Conveyor Work Area



en0900000529

Configuring the conveyor work area

Use this procedure to configure the conveyor work area in the line.

1 In the line view, right-click and select **Add Conveyor Work Area**.

The **Conveyor Work Area** dialog is opened.

Continues on next page

4.2.3 Configuring the conveyor and the conveyor work area
Continued

2 Select controller and robot that the conveyor is connected to.
3 Select work area index.

   In the RAPID program the work area index can be used to specify the pick/place order when using several work areas.
4 Select conveyor, as defined in the robot controller.
5 Select type of work area, for example **Pick** or **Place**.
6 In the **Signal names** part, clear the **Use default** checkbox to edit the signals **Position generator**, **Trig**, **Strobe**, **Conveyor start/stop**, **Queue idle**, or **Position available**. See *Conveyor work area signals on page 64*.
7 Click **OK**.

Conveyor work area signals

Signal	Description
Position generator	Digital input signal that tells that it is time to generate a new vision image or generate new predefined positions. This signal is ignored if a distance triggered conveyor is used.
Trig	If vision is used this digital output signal must be connected to the trigger input on the I/O port on the camera. If predefined positions are used this output signal must be connected directly to the start input on the conveyor encoder board. This is best done using the *doManSyncX* signal. If predefined positions are distributed only to this work area (For instance, PickMaster with a single robot),the encoder signal *cXSoftSyncSig* can be used instead of *doManSyncX*, that is, without the need of connecting a signal to the start input of the encoder board. **Note** When encoder is connected to DSQC 2000 use the `cxTrigVis` signal for camera trigger.
Strobe	This is the input signal name for the strobe signal and is the start signal for the encoder board for the conveyor. The signal name is set to *cXNewObjStrobe*. If vision is used the signal must be generated from the strobe output on the I/O port of the camera. When predefined positions are used, the strobe may be generated directly from the *doManSyncX* signal, which is directly connected to the start signal on the encoder board.
Conveyor start/stop	This output signals is used if the conveyor should be controlled by the work area.
Queue idle	This output signal is high when the queue for this work area is empty. The signal goes high when the last item is retrieved from the queue.
Position available	This output signal is high when there is one or more items between the enter and exit limits for the work area. (This can for example be used for post inspection.)

Note

Using distance triggered Positions Source with DSQC2000, camera or predefined source, configure `cxTrigVis` as **Trig** signal. From RW6.10 and later, the Strobe signal is automatically configured and can therefore be omitted in the work area signal configuration.

Continues on next page

Note

When using predefined distance triggered sources with DSQC2000, a jumper is required between **Trig output** pin 6 and **Sync input** pin3 on the camera connector (X21-X28) assigned for the source.

4.2.4 Configuring an indexed work area

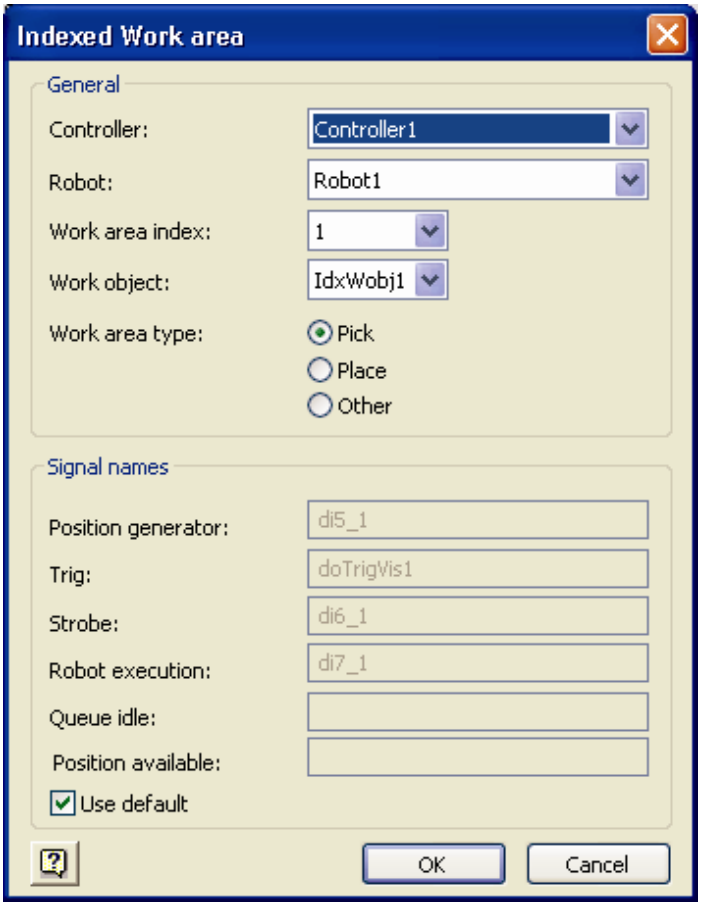
Introduction to indexed work areas

An indexed work area is an area for picking or placing without conveyor tracking. It can be moving or fixed. For a moving indexed work area, an I/O signal is used to indicate when the work area is in position so that PickMaster can get an image of the items and another I/O signal is used to indicate when the robot can start the pick and place execution.

Note

You can use up to 25 indexed work areas in PickMaster when using the IRC5 controller. The number of conveyer work areas is still limited to six and the maximum total number of work areas is 25.

Illustration indexed work area



en0900000530

Configuring an indexed work area

Use this procedure to configure an indexed work area in the line.

1 In the line view, right-click and select **Add Indexed Work Area**.

The **Indexed Work Area** dialog is opened.

Continues on next page

2 Select the controller and the robot that the indexed work area is connected to.
3 Select the work area index.

   In the RAPID program the work area index is used to specify the pick/place order when using several work areas.
4 Select the work object.

   This must be the same as the calibrated work object.
5 In the **Signal names** part, clear the **Use default** checkbox to edit the signals **Position generator**, **Trig**, **Strobe**, **Robot execution**, **Queue idle**, or **Position available**. See *Indexed work area signals on page 67*.
6 Click OK.

Indexed work area signals

Signal	Description
Position generator	Digital input signal that tells that it is time to generate a new vision image or generate a new package of predefined positions.
Trig	If vision is used this digital output signal must be connected to the trigger input on the camera I/O port. If predefined positions are used, this output signal must be connected directly to the strobe input signal. This is best done using a simulated output signal with a logic cross connection to a simulated strobe input signal.
Strobe	This is the input signal name for the strobe signal. If vision is used, the signal must be generated from the strobe output on the I/O port of the camera. If predefined positions are used, the strobe may be generated directly by the trigger output. This is best done using a simulated output signal for the trigger signal and a logic cross connection to a simulated strobe input signal.
Robot execution	This is the name of the digital input I/O-signal that is used to indicate that it is allowed for the robot to execute an item target. This signal is used if conditions must be fulfilled before the item source in the controller, can consider the positions as allowed to be executed. Execution will start when the signal is high and stop when the signal goes low. If the signal goes low, all remaining items in the currently executing frame will be dropped, so when the signal goes high again, the item targets for the next frame will be executed. The signal must also go low after one frame is finished and then go high again to start executing item targets for the next frame.
Queue idle	This output signal is high when the queue for this work area is empty. The signal goes high when the last item is retrieved from the queue.
Position available	This output signal is high when there are one or more items when the *Robot execution* signal is high for the work area. If no *Robot Execution* signal is used the *Position Available* signal will go high as soon as there are any items in the queue.

4.2.5 Configuring the camera

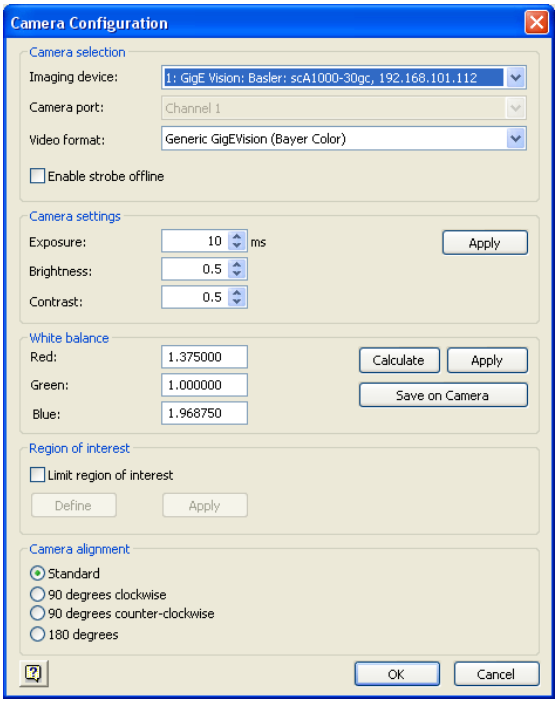
Introduction

Cameras together with vision models are used to locate objects in a specific area. When a camera is created in the line view, it is not connected to any physical camera. This must be done manually in the camera configuration dialog box. The camera in the line view is configured to use one specific physical camera. The camera should also be configured to give an optimal image.

To configure a camera in the line.

1. In the line view, right-click and select **Add Camera**.

   The **Camera Configuration** dialog is opened.



en0900000531

2. In the **Imaging device** list, select the Gigabit Ethernet camera to which the camera is connected.
3. In the **Video format** list, select the type of the connected camera.
4. If the camera should strobe when it is not in production mode, select the **Enable strobe offline** checkbox. This is necessary if, for example, the camera is used together with a strobe light. This setting applies only to Gigabit Ethernet cameras.
5. If the selected camera is a color camera and will be used together with the color video format, it is necessary to calibrate the white balance of the camera using this procedure:
   a. Put a white sheet of paper under the camera. The sheet must cover the entire field of view.

Continues on next page

b Adjust the light settings so that the image looks medium gray. Use either the camera aperture or the exposure time.

c In the **White balance** section, click **Calculate**. This will calculate the white balance calibration parameters.

d Click **Apply**. This will modify the camera's internal settings.

e Click **Save on camera**.

f Once again repeat the steps d and e. This will store the settings in the camera.

For more information about color vision, see *Using color vision on page 135*.

6 If needed, adjust **Exposure**, **Brightness**, and **Contrast** and click **Apply** in the **Camera settings** part.

Adjust the exposure to achieve the best image possible. The exposure together with the camera aperture defines the focus depth and possible motion blur. These two parameters must be suitably adjusted depending on the type of objects to look for and the speed of the conveyor.

Brightness and contrast can be changed to give an optimal image. Some objects might be easier to find by adjusting the ambient lighting together with the brightness and contrast parameters.

The effect of changing these parameter values is not seen until clicking **Apply**.

7 If needed, select **Limit region of interest** checkbox and then click **Define** to set the region. Click **Apply** to confirm the settings.

Sometimes the camera field-of-view contains areas where items will never appear. In this case it is more efficient to only capture the image region where items will appear because model search will take less time in smaller images.

8 If needed, select camera alignment in the **Camera alignment** part.

The camera image can be rotated in steps of 90 degrees depending on how the camera is mounted.

Cameras are normally mounted with the lens facing down and the "belly" towards the robot. This is the standard configuration where the movement direction of the conveyor is from the upper to the lower part of the image. A camera can also be mounted with the "belly" down and lens facing the conveyor direction. This can give a camera view that surveys more of the conveyor area, which might result in a lower image trigger frequency.

Cameras that use checkerboard calibration can have the standard alignment set regardless of how the camera is mounted. Images will always be correctly rotated with the checkerboard calibration. See *Calibrating the camera on page 75*.

9 Click **OK**.

Continues on next page

4.2.5 Configuring the camera
Continued

Configuring a simulated camera

The vision functions in PickMaster can be used without having a physical camera connected. The purpose is to allow vision modeling and evaluation offline on any laptop or PC and only a vision dongle connected. Instead of acquiring images from a physical camera, images are instead loaded from files. The function is for offline purposes only, and is not supported in production mode.

There are two types of dongles, the standard camera dongle and the simulation dongle. With the standard dongle connected, PickMaster automatically enters simulated mode if no camera is present when the program is started. With the simulation dongle, image acquisitions from cameras are not enabled, so all images must be loaded from files.

Use the following procedure to configure a simulated camera.

1. In the line view, right-click and select **Add Camera**.
   The **Camera Configuration** dialog is opened.
2. In the **Imaging device** list, select the **Simulated framegrabber**.

   Note

   The 8100d framegrabber is not compatible from PickMaster 3.41 onwards.

3. Configure a port.
4. Set the **Video format** to show color or monochrome images.
5. Load the images. There are two ways to load images in the various vision dialogs.
   - Load images from any folder using the "Import" button in the various vision dialogs.
   - Read image files from a registered image folder. Each camera has a default image for modeling, and a set of images for calibration which can be toggled by pressing "Acquire" in the calibration dialog. This requires some additional configuration to install a registered image as described below.

   Set the file paths that PickMaster will use to locate the images. This is done by running the file "DongleSettings.reg" found on the "PickMaster3 Installation Package" under "\PickMaster\DongleData\". The search paths are stored in the Windows registry, and may be edited. The default location for the image folder is "C:\DongleImages", so create this directory and copy the images included on the "PickMaster3 Installation Package" under "\PickMaster\DongleData\DongleImages\". The configured port of the simulated camera determines which image is loaded for that camera.
6. Click **OK**.

Related information

Using color vision on page 135.

Calibrating the camera on page 75.

4.2.6 Configuring an external sensor

Overview

An external sensor is a software component that gives external partners full control of how item positions are generated. An external sensor can use any type of item detection such as barcode readers, cameras, or a combination of photo sensors to generate item positions. If cameras are used, any vision hardware or image searching algorithms can be used. There is an interface for implementing external sensors as software components in .NET, for example by using C# as programming language. For more information, see *Overview of External Sensor .NET on page 177*.

An external sensor server can also be implemented as a COM object. To learn more about how external sensors are implemented as COM objects, see the file *PickMaster SDK.pdf* in the installation folder.

If an external vision system is used it may be necessary to turn off PickMaster's internal vision system to avoid that both systems try to connect to the same camera. Refer to section *Setting PickMaster options on page 204*.

4.2.7 Calibrating the line

Overview

The calibrations needed for PickMaster lines are camera and work area calibrations. The work area calibration is a base frame calibration for conveyor work areas and a work object definition for indexed work areas. The key concept is to define a coordinate system origin that is the same for a camera and a robot base frame or work object.

Each camera must be calibrated separately. The base frame calibration is needed whenever conveyor systems are used.

The camera calibration is stored in the line so all projects in that line share the same calibration. If you need to re-calibrate a line, all projects in the line will be updated with the new calibration.

The camera calibration and the work area calibration can be performed independently of each other, but it is very hard to make an accurate new camera calibration after the work area is calibrated.

The work area calibration is stored in the robot controller.

To calibrate the line:

1. Define the parameter *Counts Per Meter* (for conveyors only), see *Defining the parameter Counts Per Meter on page 73*.
2. Calibrate the camera, see *Calibrating the camera on page 75*.

4.2.8 Defining the parameter Counts Per Meter

Introduction

The *Counts Per Meter* system parameter is used to calibrate the conveyor encoder. The *Counts Per Meter* system parameter belongs to the type *Fieldbus Command*, in the topic *I/O*.

Calculation for Counts Per Meter

The value for the *Counts Per Meter* system parameter is calculated as follows:

```
(position1*old_counts_per_meter)/measured_meters
```

Value	Description
`position1`	The conveyor position after moving. Read from FlexPendant Jogging window.
`old_counts_per_meter`	The encoder's old value. **Note** The encoders delivered from the factory have a preset value. For an IRC5 system this value is 20,000. This value can be used to start the calibration.
`measured_meters`	The manually measured distance in millimeters that the conveyor has been moved.

Defining Counts Per Meter

Use the following procedure to define *Counts Per Meter* for the conveyor encoder.

1. Put a mark on the conveyor belt, for example draw a line or attach a piece of tape, and a mark on the side of the conveyor at the same location.
2. In the FlexPendant **Program Editor**, load and run the program ppacal.prg.

   This sets the current position of the conveyor to zero. The value is shown as **CNV** value in the **Position** part of the FlexPendant **Jogging** window.
3. Run the conveyor belt approximately 1 meter.
4. In the FlexPendant **Jogging** window, read the position of the conveyor. This is `position1`.
5. Measure the physical distance between the two marks. This is the value `measured_meters`.
6. Calculate *Counts Per Meter* using the read and measured values.

   For example: `(1010*20000)/1005 = 20099`
7. In RobotStudio, click **Configuration** and select topic **I/O** and type **Fieldbus Command**.
8. Select the unit *Qtrackx* (where x is the number of the conveyor) and update the value for parameter *Counts Per Meter*.
9. Tap **OK**.
10. Restart the controller.

Continues on next page

4.2.8 Defining the parameter Counts Per Meter
Continued

Related information

Application manual - Conveyor tracking.

Technical reference manual - System parameters.

4.2.9 Calibrating the camera

Introduction

Overview

The camera calibration defines the origin for the coordinate system shared by the camera and the robot base frame or work object. If the camera is used with a conveyor work area the camera calibration must be performed before the base frame calibration because the camera calibration origin works as a common reference point for the two calibrations. When a camera calibration is done, the origin is saved and the user can graphically display this origin when the base frame calibration is performed.

> **Note**
>
> PickMaster lines using the grid-of-dots calibration do not work with PickMaster 3.31 and later. Saving such a line with a later version (3.31 or later) resets the calibration. For PickMaster 3.31 or newer versions the camera must be re-calibrated using the checkerboard calibration method.

Checkerboard calibration

The camera calibration method is called *checkerboard calibration*. The calibration is performed in two steps. First the whole image is analyzed and warped into a correct image and then the region of the resulting image is defined.

The algorithm uses the scale in the center of the image, which means that it makes all the tiles the same size as the tile at the center of the original image.

Multi-view calibration

The camera can be calibrated using one or several images. The difference when using more than one image is that the camera's position in space is calculated. This space information is used both for 2.5D applications when the product height needs to be determined, and for compensating parallax errors in pure 2D applications. See *Working with products of varying height (2.5D vision) on page 142*.

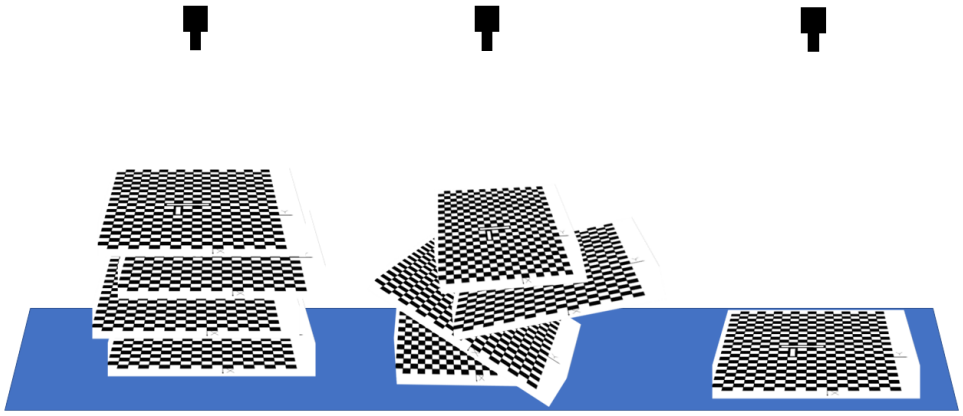
The accuracy of the multi-view calibration increases with the number of input images.

Use atleast 10-15 images with the following specifications:

- A set of images with different heights where the calibration pattern is flat under the camera (3 to 5 images).
- A set of images where the calibration pattern has different tilt and heights. (Minimum 3 images but more images give better results.)
- Place the calibration pattern down on the conveyor surface. This should be the origin image.

Continues on next page

4.2.9 Calibrating the camera
Continued



xx2300000276

Note

Using multiple images of calibration plates in parallel planes does not increase accuracy.

Prerequisites

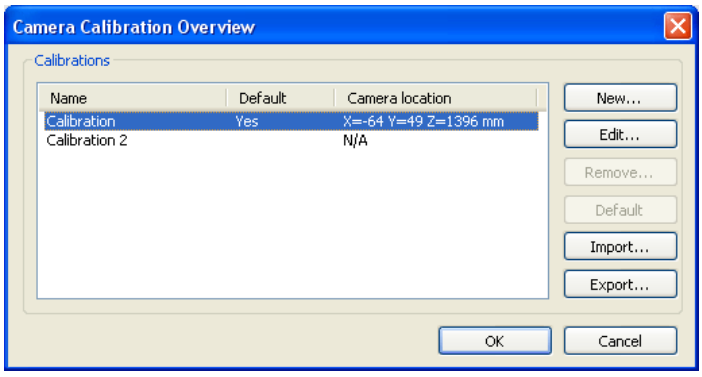
Camera calibration is done using calibration papers that you must print out. The calibration papers are found in the PickMaster installation and on the CD.

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. If the square width or height differs from 10 mm, make a note of the actual measures.

The calibration paper must be adequately illuminated and free from shadows.

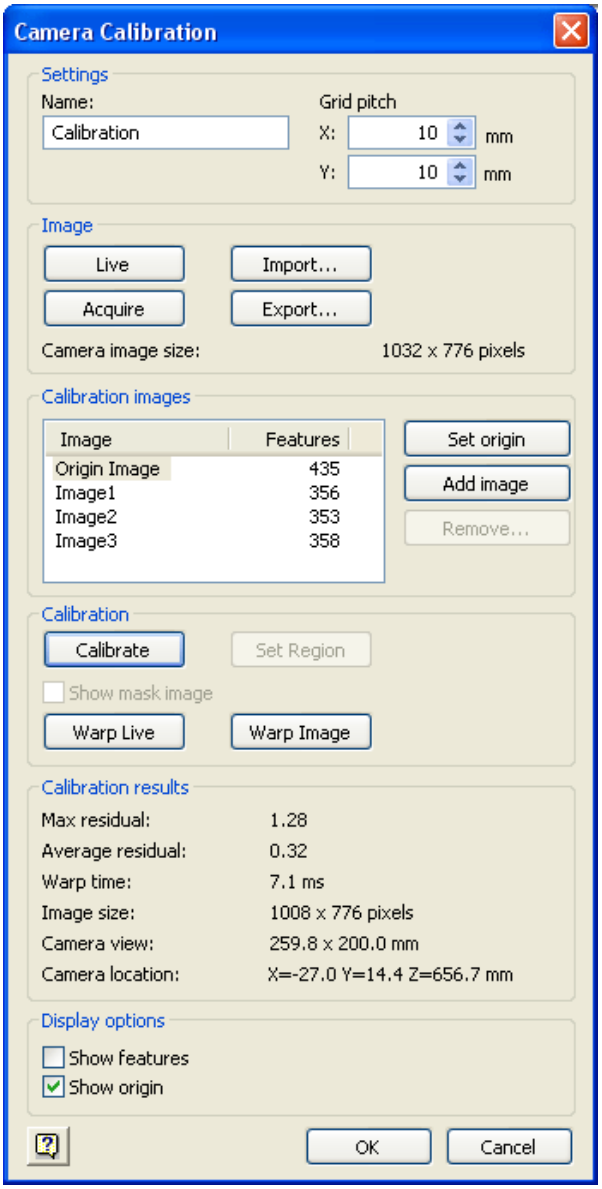
If a conveyor is used, the x-axis of the calibration paper must be aligned with the motion of the conveyor.

Illustration



en0900000647

Continues on next page



en0900000532

Calibrating the camera

The **Camera Calibration** dialog can be used to handle camera calibrations for the specified camera. Calibrations can be created, edited, imported, and exported.

Use this procedure to calibrate the camera.

1 Right-click the camera in the line view and select **Calibrate**.

  The **Camera Calibration Overview** dialog is opened.

2 Select the default calibration from the list and click **Edit**.

  The **Camera Calibration** dialog is opened.

Continues on next page

3 In the **Settings** part, adjust the grid pitch if the squares on the printed calibration paper differs from 20.0 mm. Enter an appropriate name for the calibration.

> **Tip**
>
> Using precise measurements improves accuracy. A good way to get a more precise measurement of the grid pitch is to measure for example 10 tiles and calculate the average size.

4 In the **Image** part, click **Live** to get and show new images continuously, or click **Acquire** to get one new image. To use an image from file or save the current image, click **Import** or **Export**.

5 For single-view calibration: When the calibration plate is in position, acquire an image and click **Set Origin** in the **Calibration images** part. This stores the image and marks it as the origin image (the origin of this image will be the physical origin of the camera's coordinate system).

6 For multi-view calibration: When calibrating a camera with multiple images it is important that the origin image is still in place after finishing the camera calibration. This is because the origin image is used to define the coordinate system of the robot.

   There are two ways of achieving this. One way is to acquire additional views first (click **Acquire** and **Add**) and acquire the origin image last (click **Acquire** and **Set origin**), leaving the calibration plate in the correct place for calibration of the work object/base frame.

   The other way is to use two calibration plates with the exact same grid pitch. Put one calibration plate in the position to represent the origin of the camera. Acquire an image and click **Set origin**. Leave this plate in place while acquiring images of the second calibration plate at different angles and altitudes and click **Add** to save them to the list.

7 In the **Calibration** part, click **Calibrate** to start calibration.

   The image is analyzed and calibration is performed with the specified parameters. A corrected image is shown together with an adjustable rectangle used to define the final image area. The calibration is not complete until the region is defined.

8 Adjust the rectangle to the desired region and click **Set Region** to define the resulting image size.

   The calibration is now completed and the result is displayed in the **Calibration result** part. See *Calibration result on page 79*.

9 If needed, click:

   - **Calib Image** to show the original image used to calibrate the camera.
   - **Warp Live** to show continuously acquired and corrected images.
   - **Warp Image** to correct the current image.

Continues on next page

10 If needed, click:
- **Show features** to show the checkerboard vertices used during the calibration. The features are only shown in the calibration images.
- **Show origin** to show the origin of the resulting coordinate system. The origin is only shown in corrected images.

11 Click **OK**.

Tip

For conveyors, leave the calibration paper as it is until the base frame has been calibrated.

Note

You can export or import camera calibrations. The exported file is stored in `.pmcalib` format. It is also possible to export images from the camera calibration window for storing the images used for a certain calibration.

Calibration result

Result	Description
Max residual	The maximum residual error for the calibration.
Average residual	The average residual error for the calibration.
Warp time	The time required correcting an image. This time has to be considered when calculating the total time for the image analysis.
Image size	The resulting size in pixels of the corrected image
Camera view	The resulting size of the camera view calculated with the new calibration.
Camera location	The position of the camera in relation to the origin of the origin calibration plate.

4.2.10 Calibrating the conveyor

Introduction

For each conveyor work area on a conveyor, a conveyor base frame calibration must be performed. The base frame calibration gives a reference point for the robot when a picking or placing sensor detects objects at the work area.

Preparations for calibrating the conveyor

- Define the `Counts Per Meter` system parameter for each conveyor work area. For more details, see *Defining the parameter Counts Per Meter on page 73*.
- Prepare a calibration tool that can be mounted temporarily on the robots. The calibration tool shall have a pointed TCP. Measure the TCP offset accurately.
- Create a tooldata for the calibration tool in the rapid program for each robot. Update the TCP offset with the measured values. In the FlexPendant **Jogging Window**, select the tooldata for the robot.
- If a camera is used, calibrate the camera. After calibrating the camera, keep the camera calibration pattern attached to the conveyor.

Calibration procedure IRC5

Use the following procedure to calibrate all the base frames for a conveyor in the line:

1. Make sure the reference point for calibration is marked accurately on the conveyor belt.
   - If a camera is used, the reference point is the local origin of the camera view. If the camera has been just calibrated, the reference point is already marked by the origin of the camera calibration pattern that is attached to the conveyor.
   - If an I/O sensor is used to generate predefined positions, the reference point should be marked on the conveyor at the point where the objects are detected by the sensor. This point becomes the local origin of the detected items or containers.
2. Reset the conveyor (encoder board) positions.

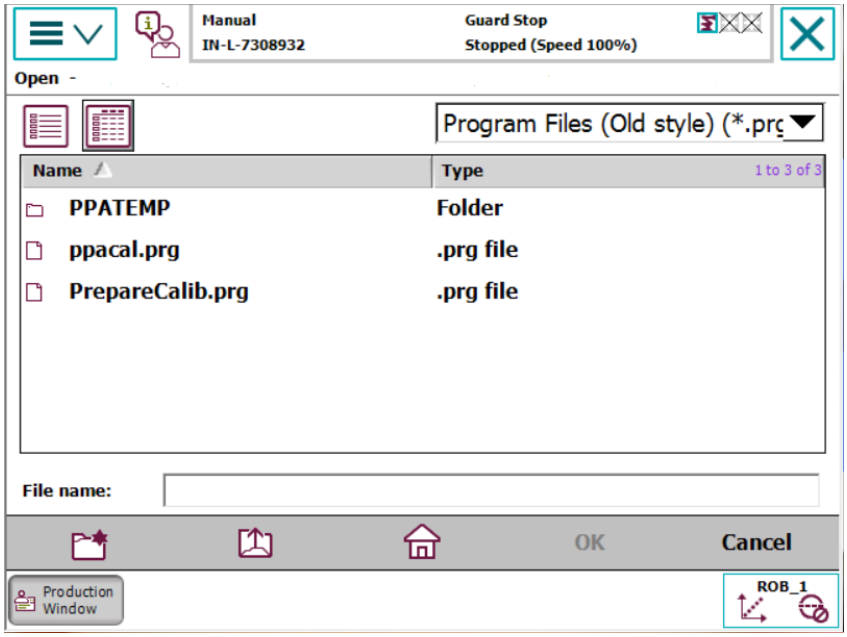
> **Note**
>
> Do not move the conveyor until this step is completely finished.

> **Note**
>
> If DSQC 377 board is used select `ppacal.prg` and if CTM board is used select `PrepareCalib.prg`.

Continues on next page

Do the following for all the robots having work areas that needs to be calibrated along the conveyor:

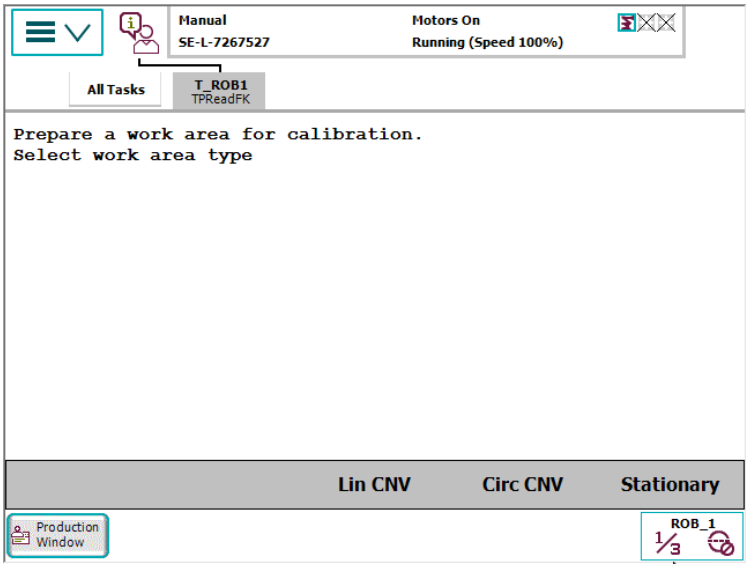
- In the FlexPendant **Program Editor**, load the program `ppacal.prg`. If the robot is a MultiMove robot, load `ppacal.prg` for this robot task (for example, `T_ROB1`), and select only this task for execution.



xx2100002648

- Start the loaded RAPID program
  - Select calibration type: Conveyor.
  - As per conveyor type, select Linear or Circular.

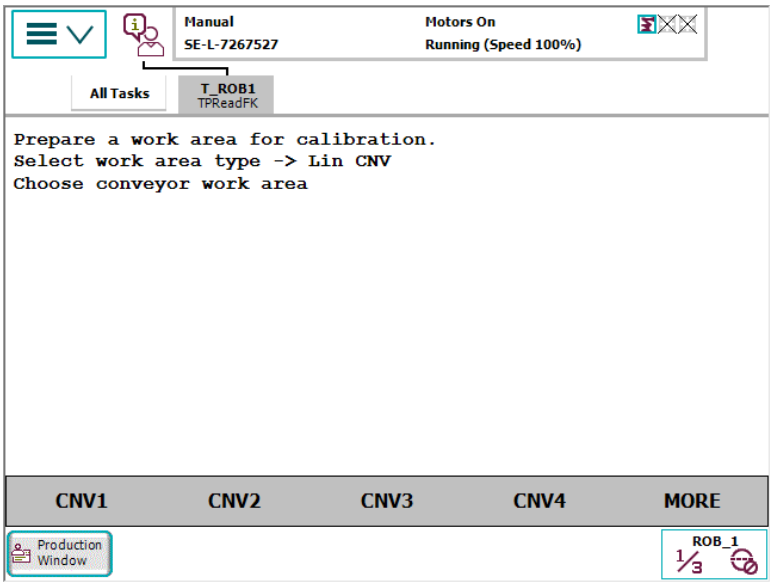
Note



Continues on next page

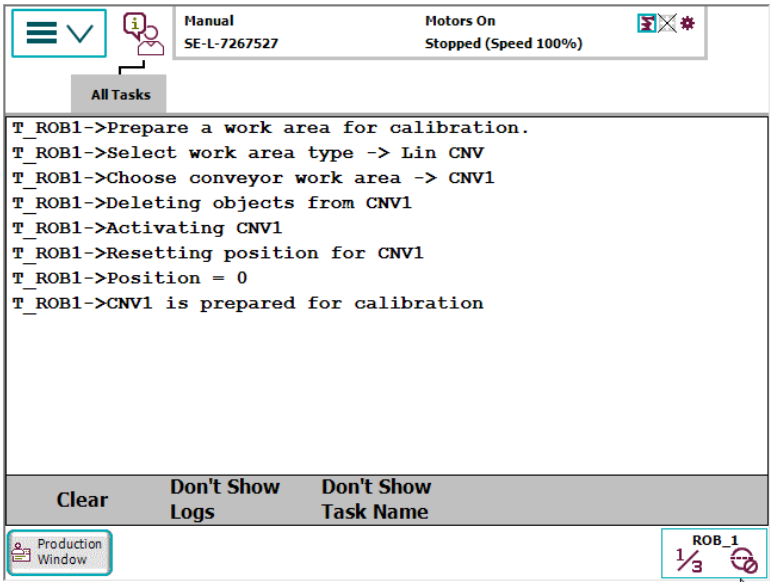
4.2.10 Calibrating the conveyor
Continued

- Select conveyor: for example, CNV1.



xx2100002650

- Wait for the message **READY FOR CALIB**. The conveyor position in the jogging window for CNV1 should now be displayed as “0” mm.



xx2100002651

3 Move the conveyor belt forward until the reference point is just inside the working range of the next robot to calibrate.

The conveyor positions for all the conveyor work areas, in the jogging window should indicate the same total travel distance for the reference point. The nearest robot to the camera or sensor is calibrated first, followed by the next nearest robot and so on until all the robots along the conveyor have been calibrated.

Continues on next page

4 Mount the calibration tool on the robot.
5 Open the Calibration window on the FlexPendant.
6 Select the conveyor, for example, CNV1.
7 Tap **Base Frame**.
8 Tap **4 Pointt**.
9 Select the robot, for example, **T_ROB1**.

   This step is required for MultiMove robots.
10 Select the first point **Point 1**.
11 Jog or move the robot by hand. Point out the reference point on the conveyor accurately with the calibration tool TCP.
12 Modify the selected point (**Point 1**) by tapping the **ModPos** function key.
13 Move the conveyor belt forward a distance where the reference point still can be reached by the robot.

   Long and equally spaced distances between the four calibration points (Point 1-4) are preferred since this increases the accuracy of the calibration.
14 Repeat the steps 10-13 for the points **Point 2**, **Point 3**, and **Point 4**.
15 Tap **OK** to calculate the base frame.
16 Check if the displayed mean error and max error of the base frame calculation is acceptable. If the estimated error is acceptable, tap OK to confirm and store the new base frame.

   Note

   A mean error of less than 1 mm is acceptable in most cases.

   If the estimated error is not ok, this base frame must be re-calibrated:
   - Move the conveyor belt backward until the reference point is just inside the working range of the robot. Repeat the steps 10-13 for all the points **Point1**, **Point 2**, **Point 3**, and **Point 4**.
   - If the conveyor belt cannot be moved backward, start over from step 1.
17 If there are more robots to calibrate along the conveyor, continue from step 3.
18 Restart the controllers to activate the new base frames.

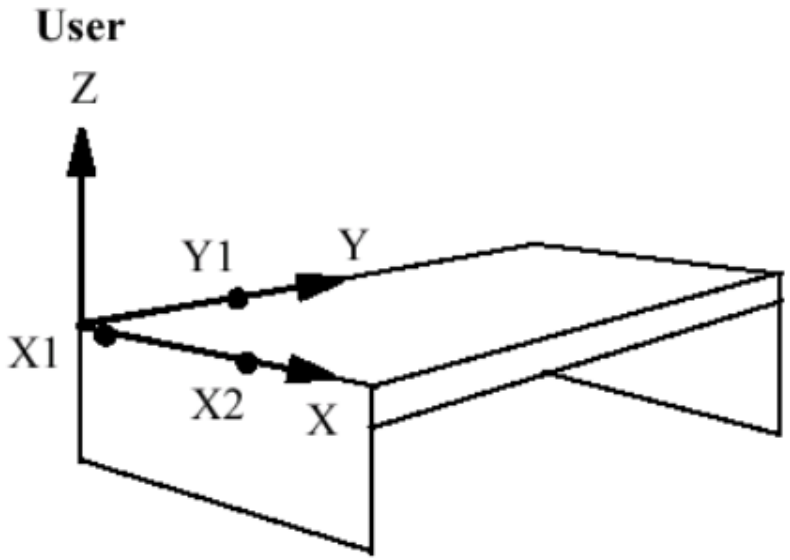
4.2.11 Calibrating the indexed work area

Introduction

For indexed work areas a work object calibration must be performed. The work object calibration gives a reference point for the robot when picking or placing sensor detected objects at the work area.

Preparations for calibrating the indexed work area

- Prepare a calibration tool that can be mounted temporarily on the robot. The calibration tool shall have a pointed TCP. Measure the TCP offset accurately.
- Create a tooldata for the calibration tool in the rapid program for the robot. Update the TCP offset with the measured values. In the FlexPendant Jogging Window, select the tooldata for the robot.
- Calibrate the camera. After calibrating the camera, keep the camera calibration pattern attached to the conveyor.
- Make sure the reference x- and y-axes for work object calibration is marked accurately on the indexed work area. Three reference points are needed for the calibration: two points on the x-axis and one point on the y-axis.
  - If a camera is used, the reference x- and y-axes should be marked with respect to the local origin of the camera view. If the camera just has been calibrated, the local origin is marked by the camera calibration pattern attached to the indexed work area.
  - If a position generator I/O signal is used to generate predefined positions, the reference x- and y-axes should be marked at the desired location for the local origin where items or containers are to be generated.



xx1400002201

Continues on next page

IRC5 procedure

1 Select the work object to be calibrated.
   - In the FlexPendant **Program Editor**, load the program `ppacal.prg`. If the robot is a MultiMove robot, load `ppacal.prg` for this robot task, for example `T_ROB1`, and select only this task for execution.
   - Start the loaded rapid program
     - Select calibration type: Fixed/Indexed.
     - Select work object: For example, `IdxWobj1`.
     - Wait for the message **DEFINE CURRENT WORKOBJECT**.

     Note

     Do not move the program pointer until the calibration has been completed. Otherwise, the calibration is not properly saved.

2 In the FlexPendant **Jogging** window, tap and select **Workobject**. Then tap **Edit** and select **Define**.
3 Select **Object method**: No Change. Select User method: 3 points.
4 Select **User Point X 1**. Point out a point on the x-axis close to the origin with the robot's TCP. Press Modify Position.
5 Select **User Point X 2**. Move the TCP a distance in the direction the x-axis. Point out a point on the x-axis with the robot's TCP. Press Modify Position.
6 Select **User Point Y 1**. Point out a point on the positive y-axis with the robot's TCP. Press Modify Position.
7 Tap **OK**.
8 Restart the RAPID program (without moving the PP) to save the selected work object definition.

   The definition is saved in the rapid data array `NonCnvWOData` located in the `ppaUser` system module.

4.2.12 Verifying the calibrations

Introduction

The calibration is verified by using a calibration verification paper. The paper has a model that is taught and used as a bull's eye for the robot to find. The same tool is used here as for the base frame calibration.

The file with the calibration verification paper is found in the PickMaster package.

To achieve a very good calibration, the camera calibration tune and the base frame calibration tune steps can be performed more than once. Each time the result should be closer to the optimal calibration.

> **Note**
>
> The calibration tuning should only be used for small errors. If the error is large then the line should be recalibrated.

Tuning the camera and base frame calibrations

Use this procedure to tune the camera and base frame calibrations.

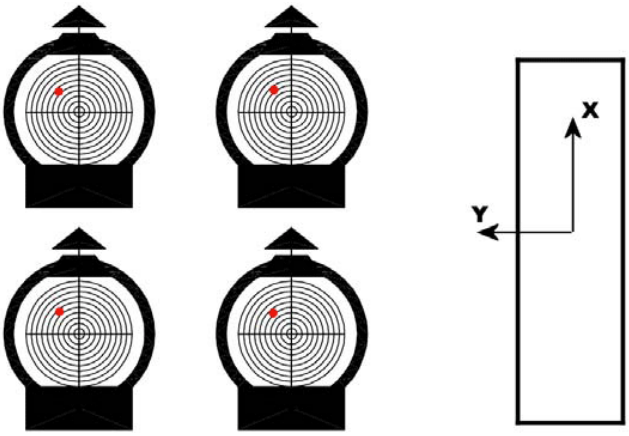
1. Create a new project.
2. Place the calibration verification paper on the conveyor under the camera. The center column of object should be placed close to the center of the camera view. Align the paper with the conveyor as accurately as possible.
3. Use one of the objects on the calibration verification paper as model. See *Calibrating the camera on page 75*.
4. Place the grip position in the center of the model.
5. Run the project.

   The robot will touch the objects on the paper and thereby punch holes.
6. Examine how the robot is placing the holes to adjust possible errors in the camera calibration or the base frame calibration.

xx0900000649

Continues on next page

If the holes are rotated compared to the center of the objects, then tune the camera calibration. In the above graphic a positive value is suitable, for example 0.3.

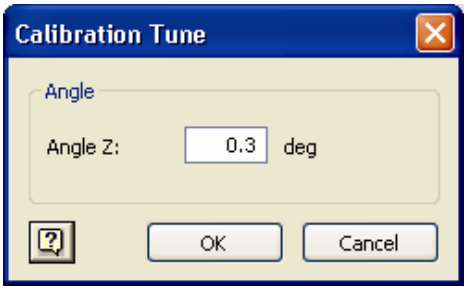


xx0900000650

If the holes are off center of the objects then tune the base frame calibration. In the above graphic, tuning with the following values are suitable: -2, -2, and 0.

It is possible to tune both calibrations but we recommend tuning the camera first and running the project again to verify the result before tuning the base frame.

7 To tune the camera calibration, open the line view, right click the camera and select **Calibration Tune**.



en0900000651

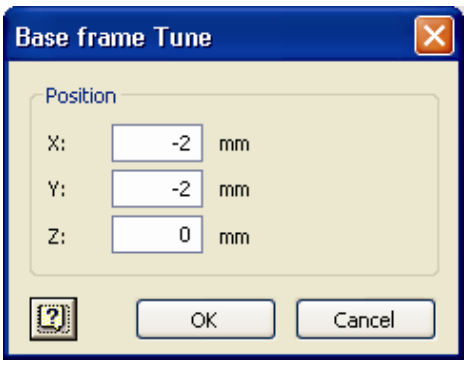
8 Enter angle adjustment in the **Angle Z** box and click **OK**.

The Calibration Tune adjusts the orientation of the camera view origin. A positive angle adjustment rotates the camera view origin in a counter clockwise direction. A negative angle adjustment rotates the camera view in a clockwise direction.

Continues on next page

4.2.12 Verifying the calibrations
Continued

9 To tune the base frame calibration, open the line view, right-click the work area and select **Base Frame Tune**.



en0900000652

The Base Frame Tune adjusts the positions of detected objects in the work area. For a conveyor work area, the adjustment is applied in the conveyor base frame. For an indexed work area, the adjustment is applied in the work object.

10 Run the project again to verify the result. If needed, adjust the calibrations again.

11 When the calibration is approved, the project can be deleted.

4.3 Configuring the project

4.3.1 Configuring the project

Overview

A project defines how items are found and how robots should pick and place these items.

Projects are saved as .pmproj files.

Prerequisites

A line must be configured.

Configuring the project

Use this procedure to configure the project.

1. On the file menu, click **New Project**.
2. Configure items, see *Configuring items on page 91*.
3. Configure patterns, see *Configuring patterns on page 94*.
4. Configure containers, see *Configuring containers on page 97*.
5. Configure position sources, see *Configuring position sources on page 99*.
6. Configure work areas, see *Configuring the work area on page 107*.
7. Configure the robot controller, see *Restarting the robot controller on page 111*.
8. Configure the robots, see *Configuring the robot settings on page 112*.
9. Configuring vision models, see *Vision modeling on page 113*.
10. Configuring inspection models, see *Configuring inspection models on page 127*.

The project overview

An overview of the project configuration is shown in the Project tab in the workspace area.

Icon	Description	Icon	Description
 xx0900000503	Item	 xx0900000504	Container
 xx0900000505	Vision position source	 xx0900000507	Predefined position source
 xx0900000506	External sensor position source	 xx0900000508	Distance trigger

Continues on next page

4.3.1 Configuring the project

Continued

Icon	Description	Icon	Description
 xx0900000509	I/O trigger	 xx0900000510	Conveyor work area
 xx0900000511	Indexed work area	 xx0900000512	Camera
 xx0900000513	External sensor	 xx0900000514	Geometric vision model
 xx0900000515	Blob vision model	 xx0900000516	Inspection vision model
		 xx0900000518	External position generator

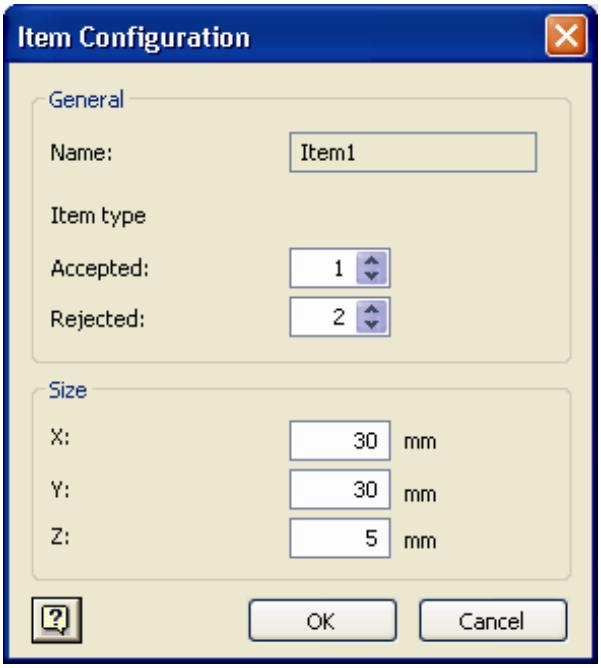
4.3.2 Configuring items

Introduction to items

An item is the object that is picked and placed by the robot. It is most common to use only one item for both pick and place but any number of items can be created.

The grip location of an item defines the pick/place position relative to the item position.

Illustration Item Configuration



en0900000521

Configuring items

Use this procedure to configure items in the project.

1. Right-click the project view and select **New Item**.
2. To edit the item, right-click the item and select **Edit**.
3. If needed, define levels for accepted or rejected item types.

   When inspection is used, a found item will be marked as either accepted or rejected. The values for accepted and rejected item type in the **Item Configuration** dialog are sent to the RAPID program and are processed there. See *Configuring inspection models on page 127*.
4. In the **Size** part, define the item's size.

   The height of the item, **Z**, defines the pick height and is always added to items found by a vision model or a position defined by a predefined position source.
5. Click **OK**.

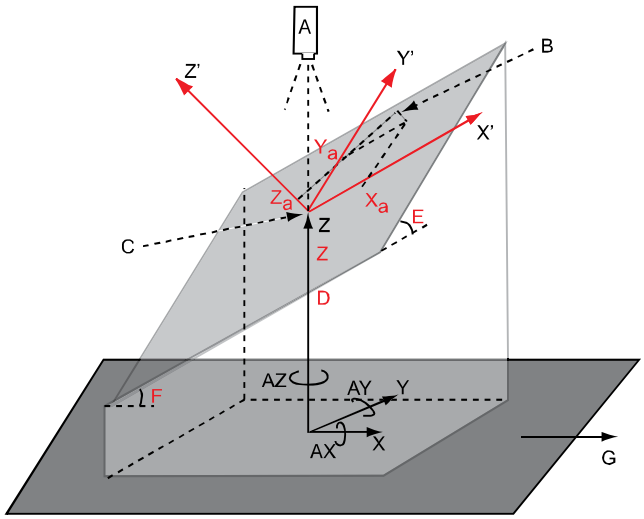
Continues on next page

Configuring the grip location

Use this procedure to configure the item's grip location.

1 Right-click the item and select **Grip Location**.

2 Define the positions in millimeters for the grip position of the item specified in X', Y', and Z' coordinates. The positions are relative to the origin of the taught model (Vision model grip point). See the following graphic.



xx0900000522

A	Camera
B	Adjusted grip point
C	Vision model grip point
D	Item height
E	Angle X
F	Angle Y
G	Conveyor direction

3 Define the Euler orientation in degrees for the grip orientation on the item.

A four axes robot (that is IRB 360) can only rotate around the z-axis and therefore only **Angle Z** can be used.

Six axes robots can pick/place 3D items by defining Euler orientation **AngleX**, **AngleY** and the item height. The grip orientation has an orientation in relation to the origin of the taught model (Vision model grip point). The item height must be specified in the **Item configuration** dialog, as a distance from the base frame to the item origin (vision model grip point).

It is important to define a correct calibration tool when calibrating the base frame of the conveyor, so the orientation in relation to the items grip point

Continues on next page

(place/pick) will be correct. It is also important to do the camera calibration at the same height as the item's grip point, that is vision model grip point.

> **Note**
>
> Select the **Apply Z Grip Location** check box from **Container Configuration**, then the Z offset will work for projects created in Version PM3.54 and above. For projects created in version 3.53 or below, selecting this check box will not have any effect.

4 Click **OK**.

Related information

Configuring inspection models on page 127.

4.3.3 Configuring patterns

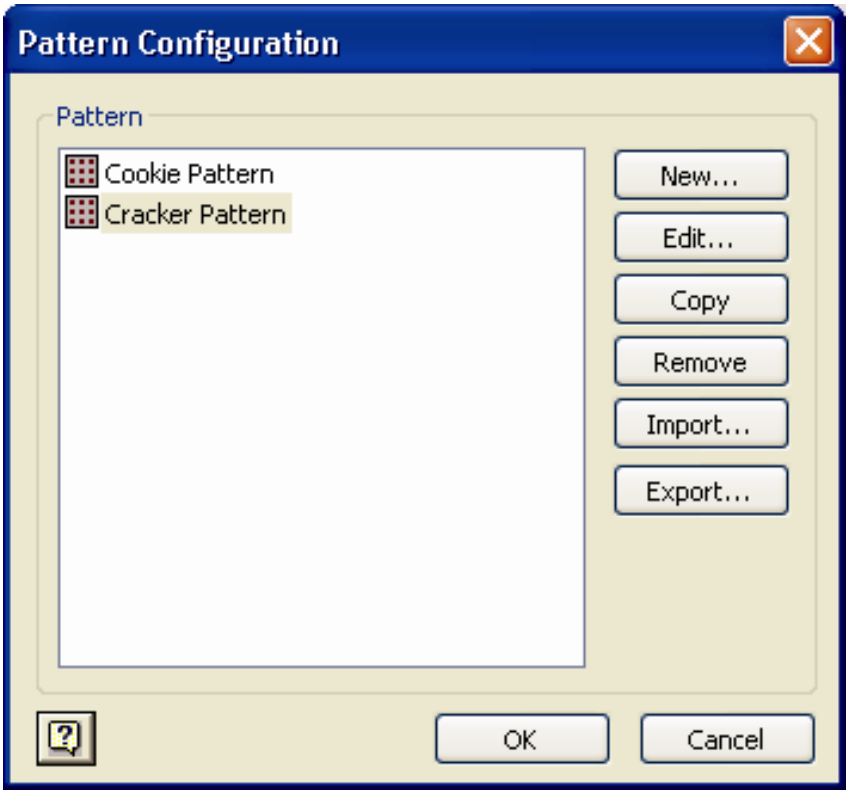
Introduction to patterns

A pattern defines a collection of positions, for example a box with predefined locations for certain objects.

The pattern is configured in three dialogs:

1 Creating the pattern object in the **Pattern Configuration** dialog.
2 Creating the pattern positions in the **Position Configuration** dialog.
3 Defining each position in the **Pattern Position** dialog.

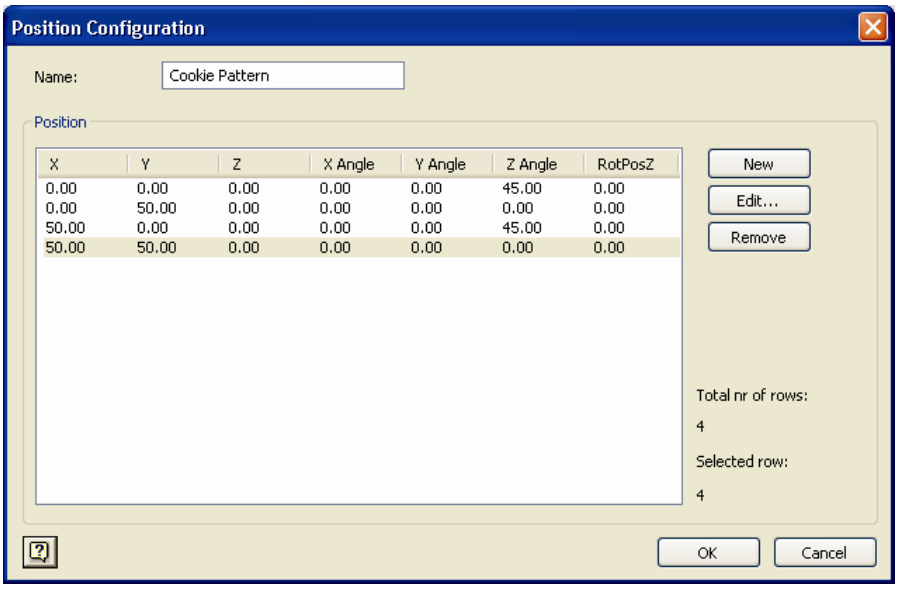
Illustration Pattern Configuration



en0900000523

Continues on next page

Illustration Position Configuration



en0900000524

Configuring patterns

Use this procedure to configure patterns.

1. Right-click the project and select **Edit Patterns**.

   The **Pattern Configuration** dialog is opened.
2. Click **New** to create a new pattern or click **Edit** to edit the selected pattern.

   The **Position Configuration** dialog is opened.

   Patterns can be copied, removed, imported, and exported. A pattern can only be removed if it is not used in any containers in the project. Patterns can be imported from pattern files (*.pmpattern) or from other projects.
3. Click **New** to create a new position or click **Edit** to edit the selected position.

   The **Pattern Position** dialog is opened.
4. Define the position in millimeters and the angles where each item should be placed/picked.

   Each item has a relative position in relation to the origin of the pattern.

   A four axis robot (that is IRB 360) can only rotate around the z-axis and therefore only **Angle Z** is used.

   Six axes robots can pick/place items on 3-D patterns by defining Euler orientation **Angle X**, **Angle Y**, and the distance **Rotating Position from base frame** for each pattern.
5. Click **OK**.

Pattern positions

Angle X and **Angle Y** is the orientation in degrees for the pattern surface where each item shall be placed/picked. Each orientation has an orientation in relation to the origin of the pattern.

Continues on next page

Angle Z is the rotation of the item on the pattern surface where each item shall be placed/picked.

Rotating Position from base frame is the distance from the conveyor's base frame to the pattern origin, that is the height to the taught vision model grip point.

It is important to define a correct calibration tool when calibrating the base frame of the conveyor, so the orientation in relation to the item's grip point (place/pick) will be correct. It is also important to do the camera calibration at the same height as the item's grip point, that is vision model grip point.

4.3.4 Configuring containers

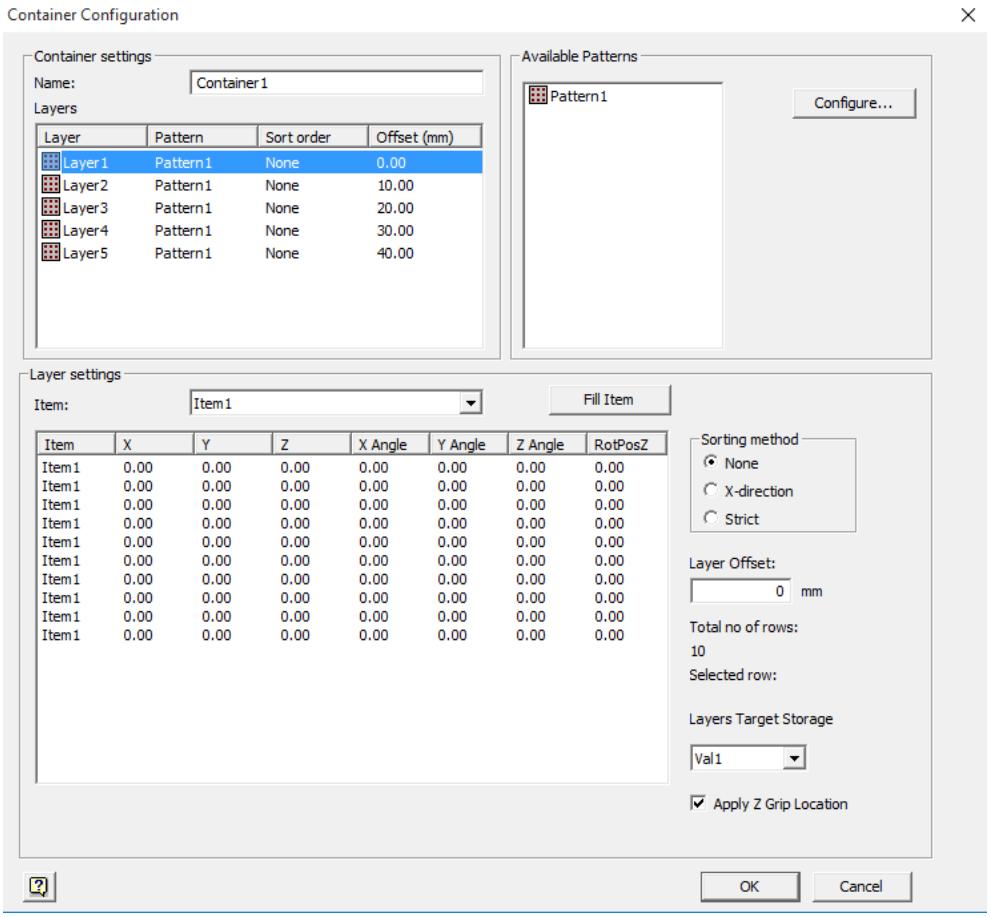
Introduction to containers

A pattern only defines object positions and does not say anything about what type of items to put in the positions. A container defines which patterns to use and what items to use for each position in the patterns. This way, different containers can use the same patterns but with different items.

Prerequisites

At least one item must be defined in the project before configuring the container.

Illustration Container configuration



en0900000525

Configuring containers

Use this procedure to configure containers for the project.

1 Right-click the project view and select **New container**.

The **Container Configuration** dialog is opened.

2 Select a pattern in the **Available Patterns** part.

Continues on next page

If no suitable pattern is available, click **Configure** to open the **Pattern Configuration** dialog and configure the pattern. See *Configuring patterns on page 94*.

3 Drag the pattern to the **Layers** list in the **Container settings** part.

The same pattern can be used in many layers. Change the order of the layers by dragging them up or down in the **Layers** list. Details about the selected layer is shown in the **Layer settings** part.

All item positions that the robot should use in the first layer must be completed before starting with the next layer. Item positions in the layer that are sent to other robots will be ignored.

4 In the **Layer settings** part, select items from the **Item** list for each position. If all positions should use the same item, select the item and click **Fill**.

5 Select sorting method, **None**, **X-direction**, or **Strict**.

None means that the item positions in the layer will be accessed in the same order as they are defined in the pattern, but if the next item position cannot be reached the next one after that will be used.

X-direction means that the item positions will be accessed in the x-direction.

Strict means that the item positions are used in the exact same order as they are defined in the pattern. If a robot cannot access the next item position in a layer, that robot will not use any more item positions in the container. This setting will only take the item positions that the robot is to use in account. Any item positions in the layer that are sent to other robots will be ignored.

> **Note**
>
> **Strict** sorting method cannot be combined with **Load Balancing** or the RAPID data types `sortdata` or `selectiondata`. **Skip** function will not work as expected if combined with the **Strict sorting** function. If multiple items are used in a container all items must be send to the same distributor.

6 Define **Layer offset** as the z-offset in mm for the layer, from the container bottom.

7 **Layers Target Storage** stores container's layer number. A target storage (`ValX`) could be made selectable for a layer number. The RAPID program or user hook would then need to check for the layer number and skip all the targets in the higher layers.

8 Select **Apply Z Grip Location** to make the Z offset for grip location to work in container. This is applicable only for the projects created in PickMaster 3.54 and after. For the projects created using PickMaster 3.53 or before this selection will not have any effect.

9 Click **OK**.

4.3.5 Configuring position sources

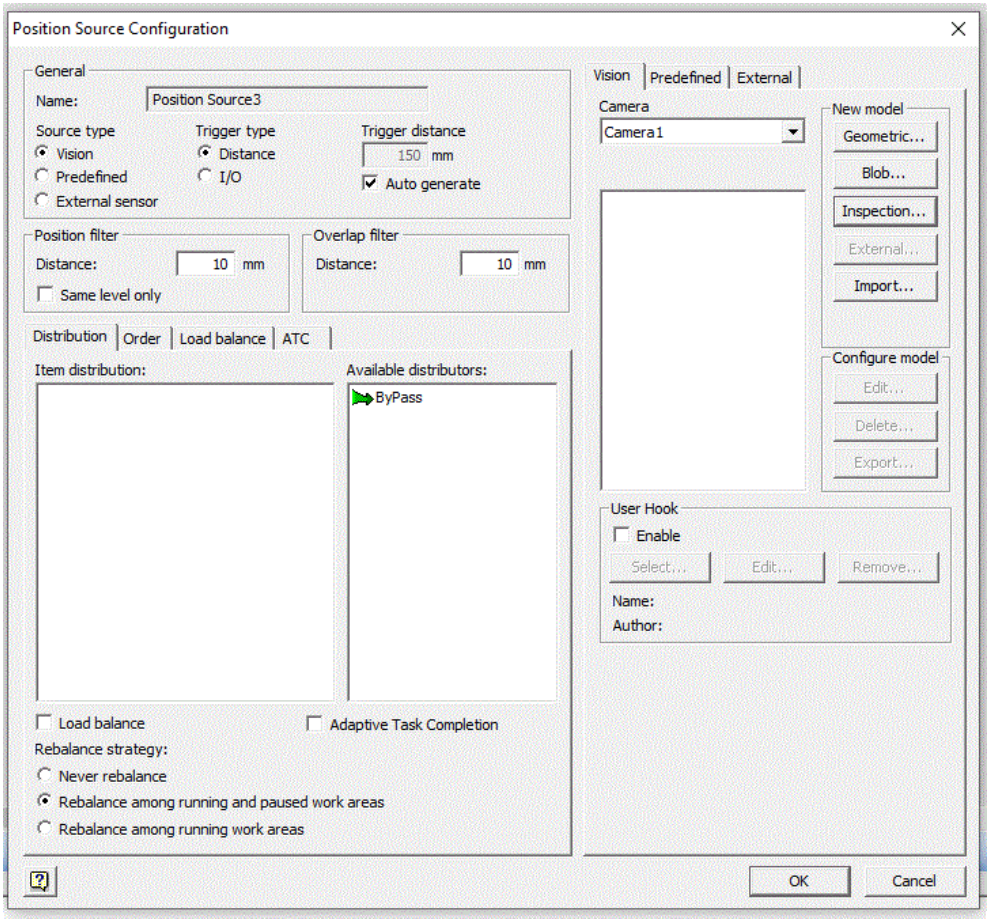
Overview

The central object in the project is the position source. It defines generating the item positions and sending them to work area.

To configure the position source:

1. Double-click on **Position source** or right-click on **Position source** and choose **Edit** to open the **Position Source Configuration** dialog box.
2. Select the Source Type, Trigger Type and Trigger distance.
3. Under **Distribution** Tab, configure the item distribution, see *Item distribution on page 102*.
4. Select the source type and configure the models in the **Vision**, **Predefined**, or **External** tabs, see *Source type and models on page 104*.
5. Configure the third party software components in the **User Hooks** part, see *PickMaster User Hook on page 105*.

Illustration position source configuration



en0900000520

Continues on next page

4.3.5 Configuring position sources
Continued

Configuring position sources

Configuring position sources

Use the following procedure to configure position sources in the project.

1 Right-click the project view and select **New Item Source**.

The **Position Source Configuration** dialog is opened.

2 In the **General** part, select **Source type** to define how the position source should generate item positions.

It is possible to use a camera, an external sensor, or the positions can be predefined. See *Source type and models on page 104*.

3 Select **Trigger type** to define when to generate new item positions. If the trigger type is set to **Distance** the trigger distance must be defined in the **Trigger distance** box.

A distance trigger can only be used with a conveyor work area and the entered value is the distance the conveyor should move between consecutive triggers.

If the position source is vision defined, this distance can be auto generated but it should always be verified that the distance is reasonable. If one of the models is large, compared to the smallest region of interest, it might result in a negative distance. Then the trigger distance must be set manually.

4 In the **Position filter** part, define the minimum allowed distance between two item positions found by a camera or an external sensor.

For example, if two or more models are used to identify the same object, there might be one hit for each model at almost the same location. If two positions for the same item are closer in either x- or y-direction than the defined minimum item distance, only the position with the highest sort value will be sent to the robot controller. The sort value can be set for each vision model, see *Vision modeling on page 113*. To filter only item positions with the same inspection level, select the **Same level only** checkbox.

5 In the **Overlap filter** define the overlap distance.

For example, items can be identified in two consecutive frames due to the overlap. The models can have a small variation in the pick/place position between these frames. Items that are found in two consecutive frames and whose pick/place position between these two frames does not vary by more than the overlap filter distance will be regarded as one item. The first identified hit is sent to the robot, and any subsequent hit is filtered out.

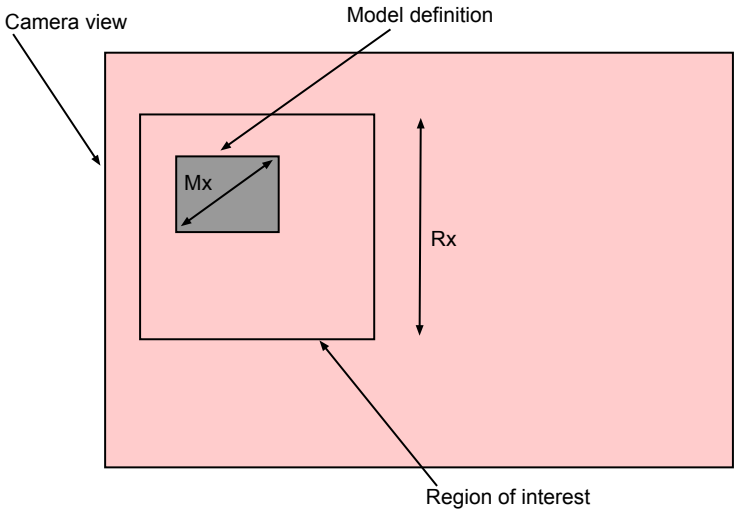
Auto generate trigger distance

The **Auto generate** check box is selected for automatically calculating the trigger distance. The auto generated distance will give enough overlapping of consecutive frames to make sure every object is completely captured in at least one image. The algorithm uses the largest defined models and the smallest region of interest to generate an appropriate trigger distance. The formula used to calculate the auto generated trigger distance, `Td`, is:

```
Td = min (Rx) − max (Mx) − d
```

Continues on next page

where `min (Rx)` is the smallest height of the region of interest and `max(Mx)` is the largest diagonal distance of all the defined models as shown in the following image.



xx1300000654

`d` is a safety margin that represents the estimated accuracy of `min (Rx)`, `max (Mx)` and the resulting trigger distance. PickMaster uses `d` = 7 mm. Decreasing the trigger distance gives a better coverage of objects in consecutive images. However, it also increases the system load. If a too small value of trigger distance is used, some images are lost due to system overload.

It should always be verified that the calculated trigger distance is reasonable. If one of the models is large, compared to the smallest region of interest, it might result in a negative distance. Then the trigger distance must be set manually.

Position filter

The position filter defines the minimum allowed distance between the different item positions found by a camera or an external sensor. For example, if two or more models are used to identify the same object, there might be one hit for each model at about the same location. If the two positions for the same item are closer in either X- or Y-direction the given minimum item distance, only the position with the highest sort value is sent to the robot controller. The sort value can be set for each vision model. For more details, see *Vision modeling on page 113*. If **Same level only** is selected, the filtering will only be done between item positions with the same inspection level.

Note

The position filter is not used while predefined positions are used . The position filter is therefore disabled in the **Position Source Configuration** window when **Predefined** is selected.

Continues on next page

4.3.5 Configuring position sources
Continued

Overlap filter

Items can be identified in two consecutive frames due to the overlap. The models can have a small variation in the pick/place position between these frames. The items that are found in two consecutive frames and whose pick/place position between these two frames does not vary more than the overlap filter distance are regarded as one item.

Item distribution

By default all positions are sent to the same work area. It is possible to distribute item positions to more than one work area to balance the load between several robots or to guarantee that all positions are accessed.

All positions for a specific item type are distributed to the robots by a single item distributor. There are four types of item distributors.

- Work area: The item positions are handled by a single conveyor or indexed work area.
- ByPass: The item positions are discarded, that is not handled by any work area. If no distributor is selected for an item type it will be considered as ByPass.
- LB group: The item positions are handled by the work areas included in a load balance group. Aload balance group is a collection of Work area, ByPass, and ATC group distributors. Item positions will be distributed among the work areas in an optimal way to avoid sending two adjacent positions to the same work area.
- ATC group: Positions are handled by the work areas included in an *Adaptive Task Completion* (ATC) group. An ATC group is a collection of ordered work areas that will get the same item positions. The first robot accesses as many positions as possible. The other robots in the ATC group will access any missed positions. If the last work area in the group is a conveyor work area with start and stop it is guaranteed that all positions will be accessed.

To use either load balancing or ATC the work areas must be arranged in the order that they occur after the position source (for example: the camera or sensor).

The work area that triggers the position source is set automatically. When starting a project, the work area for the robot that is first up and running is set to be the trigger work area. If the robot for a trigger work area is stopped, a work area for another robot that is running will be the one that triggers the position source.

The item distribution tree control shows the items for which positions are to be generated. Accepted and rejected items can be distributed differently.

Rebalancing strategies when a robot goes down

There are three ways to rebalance item positions if a robot goes down. The item positions can automatically be sent to the running robots. However, sometimes it can be more convenient to keep sending item positions to robots that are not running.

When sending item positions to a robot controller that has paused. For example, caused by a motors off state, the positions will not be lost until they have passed the robot. As soon as the robot is running again it can start picking immediately.

Continues on next page

A robot that has been stopped cannot receive any item positions until it is started again. All items that already have been distributed to it will be lost. When the robot is started again, it will have to wait until new positions reaches the robot.

Rebalancing strategies:

- Never rebalance: Item positions will always be distributed as defined in the distribution tree. If a robot is down some item position will be lost.
- Rebalance among running and paused work areas: Item positions will only be sent to work areas with running or paused robots.
- Rebalance among running work areas: Item positions will only be sent to work areas with running robots.

The different robot states are described in *Robot states on page 194*.

The selection of rebalance strategy is important while using load balancing, for example, to minimise production loss.

While using ATC for all work areas it is recommended to use the alternative Never rebalance. ATC will guarantee that every item position can be accessed by every robot in the group. Selecting the alternative Rebalance among running work areas will only increase the startup time for paused robots.

Load balancing

Item positions that are distributed by a load balance group are divided among the distributors in the group. A load balance group can contain any number of item distributors and a single distributor can appear several times. The ratio between the number of times a single distributor is added and the total number of distributors defines the ratio of the item positions that are sent by that particular distributor. Item positions are arranged to the distributors in the group in an optimal way to avoid adjacent positions to be sent to the same work area.

If *Adaptive Task Completion* is selected, any defined ATC groups will be listed among the available distributors. Additionally, ATC groups can be added to load balance groups. However, to achieve task completion, the load balance group should only contain ATC groups.

Adaptive Task Completion

Adaptive Task Completion guarantees the item positions to be accessed by any robot in an ATC group. An ATC group contains ordered work areas and a single work area is allowed to exist once in a group. All item positions distributed to an ATC group are sent to every work area in the group and the positions not accessed by the first work area will be accessed by any of the other work areas. If the last work area is on a conveyor with start and stop it is guaranteed that all item positions will be accessed by one of the robots in the ATC group.

Redistributing items from one robot to downstream robots

It is possible to modify the distribution of alredy distributed item positions when they enter a conveyor work area of a robot. The Rapid program, that controls the robot, based on current flow conditions decides to skip an item position and change the type of it. As a result, PickMaster will redistribute the item position to downstream robots according to the configured distribution strategy for the selected item type.

Continues on next page

4.3.5 Configuring position sources
Continued

Configuring item distribution

Use the following procedure to configure item distribution in the **Position Source Configuration** dialog.

1. In the **Distribution** tab, drag distributors from the **Available distributors** list to the **Item distribution** list. Double-click a distributor to set all item types to the same distributor.

   There can be only one distributor for each item type. If an item type is missing a distributor, it will be regarded as ByPass.
2. If needed, select **Load balance** or **Adaptive Task Completion** in the **Distribution** tab and then arrange the order in the **Order** tab.
3. To add work areas to the **Work area order** list, drag the work area from the **Available work areas** list. The order of the work areas can be modified by dragging an area up or down in the list.

   The **Available work areas** list contains all work areas not used by another position source. Conveyor work areas are arranged under their corresponding conveyors.
4. If using load balancing, in the **Load balance** tab, drag a distributor from the **Available distributors** list to a group in the list **Load balance groups**.

   To create a new load balance group, double-click **<New LbGroup>** in the **Available distributors** list.

   Select rebalancing strategy.
5. If using Adaptive Task Completion, in the **ATC** tab, drag a work area from the **Available work areas** list to the **Adaptive Task Completion groups** list.

Source type and models

If the source type is set to **Vision**, a camera and vision models are used to find the object positions. The vision models are described in section *Vision modeling on page 113*.

If the source type is set to **Predefined**, the positions generated by the position source are statically defined and no camera is used.

If the source type is set to **External sensor**, an external sensor in the line together with external position generators are used to define item positions. See *PickMaster SDK* documentation.

Configuring source type and models

Use the following procedure to define vision models in the **Position Source Configuration** dialog.

1. If using the source type **Vision**:
   a. In the tab **Vision**, select which camera and which models to use with the position source.
   b. In the **New model** panel, click **Geometric**, **Blob**, or **Inspection** to define a new model. See *Configuring a geometric model with PatMax on page 114*, *Configuring blob models on page 121*, or *Configuring inspection models on page 127*.
   c. To edit the model name, select the model and press **F2**.

Continues on next page

d To add a User Hook, select the **Enable** checkbox in the **User Hook** panel.

This enables a *PositionAdjuster*User Hook to manipulate the positions generated by vision models. The User Hook object can adjust the positions in any desired way. Positions can be changed, removed, or added. See *PickMaster User Hook on page 105*.

2 If using the source type **Predefined**:
   a In the tab **Predefined**, select the object to generate positions for. The object can either be an item or a container of several items.
   b Define the predefined positions in base frame coordinates to generate at each trigger. If the predefined object is a container, this position defines the origin of the pattern. Note that the item height is always added to the given Z value when generating the pick/place position.
   c To add a User Hook, select the **Enable** checkbox in the **User Hook** part.

This enables a *PositionGenerator*User Hook for the predefined positions. Each time this position source is triggered, the User Hook object is queried for positions. Any number of positions can be returned from the PositionGenerator. See *PickMaster User Hook on page 105*.

3 If using the source type **External sensor**:
   a In the tab **External Sensor**, select an external sensor or create a new one.
   b If needed, select **Synchronization tune**.

The item positions provided by the external sensor are all marked with a time stamp. This time must correspond to the strobe signal set in the robot controller. It is used to synchronize the positions with the correct scene in the controller. If the external sensor sets the time wrong, the item positions and the scene in the robot controller will not match. The corresponding error messages will be added to the log. With the *synchronization tune value* it is possible to adjust the time stamp of the item positions to any number of milliseconds to better match with the correct scene in the robot controller. See *PickMaster SDK* documentation for more information.

PickMaster User Hook

A PickMaster User Hook is a third party software component that can be designed to customize item positions during runtime. A User Hook can for example be queried for positions instead of using predefined positions. It is also possible for Hook objects to adjust item positions generated by vision models in PickMaster.

For a vision defined position source, only a *PositionAdjuster*User Hook can be used.

For a predefined position source, only a *PositionGenerator*User Hook can be used.

See *Introduction to User Hook on page 170*.

Related information

Vision modeling on page 113.

Continues on next page

4.3.5 Configuring position sources
Continued

Robot states on page 194.

Customizing PickMaster with a User Hook on page 170.

PickMaster SDK documentation, *PickMaster User Hooks.chm*.

4.3.6 Configuring the work area

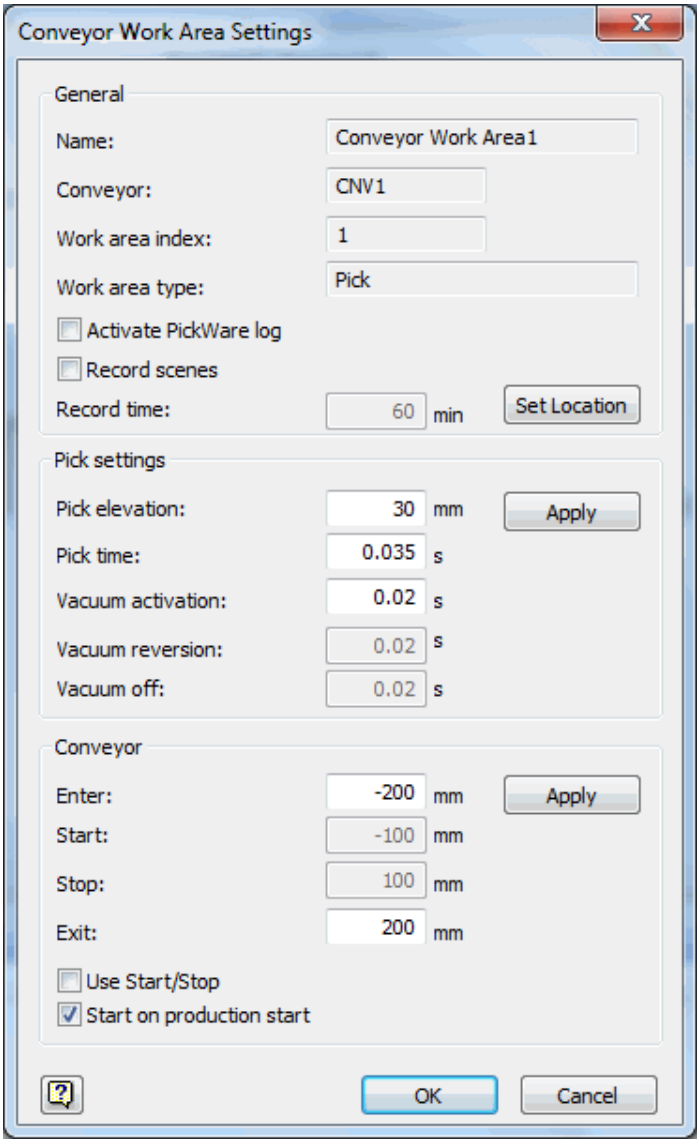
Introduction to the work area

There are several project specific settings that can be configured for each work area.

Depending on type of work area, pick or place, the available settings in the dialog will vary.

Conveyor work areas and indexed work areas are configured in the same way, but the conveyor work area has more settings.

Illustration Conveyor Work Area settings



en0900000527

Continues on next page

4.3.6 Configuring the work area
Continued

Configuring the work areas

To configure a work area in the project:

1. To create a new work area, right-click the project view and select **New work area**.

   The **Conveyor Work Area Settings** or **Indexed Work Area Settings** dialog is opened.

2. If needed, select **Activate PickWare log** during debugging.

   The robot controller PickWare log contains a lot of internal information about the work area.

3. To record the production flow of detected item positions, select the **Record scenes** check box.

   If this is selected the production flow is saved in an XML file. A recorded flow can be imported in Picking PowerPac and used in simulation. Use the **Set Location** button to set a location for the file. Recording is started at the start of the project. Use the **Record time** field to set the duration of the recording.

4. In the **Pick settings** (or **Place settings**) part, define **Pick elevation**/**Place elevation**, **Pick time**/**Place time**, **Vacuum activation**/**Vacuum reversion**, and **Vacuum off**.

   Pick/**place elevation** is the distance, in negative z-direction relative to the tool, from where the robot approaches the item target.

   Pick/**place time** is the time the robot is in the pick/place position. If the conveyor is moving during the pick/place time, the robot will track along the conveyor to keep the relative position on the moving conveyor.

   Vacuum activation is the time in seconds before the middle of the corner path of the approaching position, when the vacuum I/O should be set. If a negative value is entered, the vacuum I/O will be set the time after the middle of the corner path. This value is only valid for work areas of type **Pick** or **Other**.

   Vacuum reversion is the time in seconds before the half place time in the place position, when the blow I/O should be set. If a negative value is entered, the blow I/O will be set the time after the half place time in the place position. This value is only valid for work areas of type **Place** or **Other**.

   Vacuum off is the time in seconds after the half place time in the place position, when the blow I/O should be reset. If a negative value is entered, the blow I/O will be reset the time before the half place time in the place position. This value is only valid for work areas of type **Place** or **Other**.

   Note

   Vacuum event accuracy: The accuracy of the vacuum events (Vacuum Activation/Reversion/Off) is described in the limitations section for the TriggEquip instruction, in the RAPID reference manual. The values typed in the PickMaster dialogs are used as EquipLag in the TriggEquip instruction. Act and Rev times are positive, and the Off time is negative.

Continues on next page

Note

Vacuum activation, reversion, and off does not affect the picking and placing of items in simulation. Items are attached/detached to the picking tool using SimAttach/SimDetach events, for example, in the Pick/Place Routine.

5 For a conveyor work area, in the **Conveyor** part, define **Enter**, **Start**, **Stop**, and **Exit**. See the following figure.

Enter is the limit from where the robot starts to execute item targets on the work area. The distance is calculated in millimeters from the center of the robot. The range is positive if the limit is beyond the center of the robot, relative to the moving direction of the conveyor. Make sure that the enter limit can be reached by the robot.

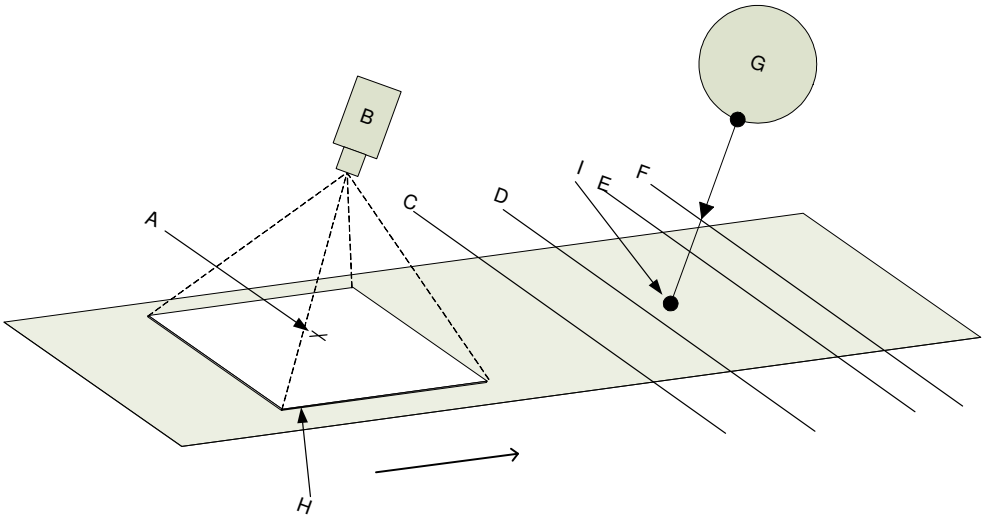
Start is when the next item to execute on the conveyor is above this limit, the conveyor is started. The distance is calculated in millimeters from the center of the robot. The range is positive if the limit is beyond the center of the robot, relative to the moving direction of the conveyor.

Stop is when an item on the conveyor reaches this limit, the conveyor is stopped. The distance is calculated in millimeters from the center of the robot. The range is positive if the limit is beyond the center of the robot, relative to the moving direction of the conveyor.

Exit is the limit from where the robot considers an item target as lost on the work area. The distance is calculated in millimeters from the center of the robot. The range is positive if the limit is beyond the center of the robot, relative to the moving direction of the conveyor. When the tracked item passes beyond this limit it will be dropped. This limit must be chosen well within the maximum reach of the robot. The robot must be able to reach this position

Continues on next page

from an arbitrary position in the robot's working area before the position is out of reach.



xx1000001344

A	Camera and Baseframe origin
B	Camera
C	Enter
D	Start
E	Stop
F	Exit
G	Robot
H	Image frame
I	Center of Robot

6 Select the **Use Start/Stop** checkbox if the work area should supervise the start and stop limits.

This is handled by the *Conveyor start/stop* signal, see *Configuring the conveyor and the conveyor work area on page 63*.

The **Start on production start** functionality is included in this option.

7 Select the **Start on production start** checkbox if the work area should start the conveyor when the project is started, and stopped when the project is stopped.

4.3.7 Restarting the robot controller

Introduction to the robot controller restart

A **Restart**, **Reset RAPID**, or **Reset system** can be done from PickMaster. The default restart mode is a **Restart**.

Restarting the robot controller

Use the following procedure to restart the robot controller from PickMaster.

1. Right-click the controller and select **Restart Controller**.

   The **Restart Controller** dialog is opened.
2. If needed, change restart mode and click **Restart**.

   The restart mode is changed by typing a number code.
   - **Reset system**: 147
   - **Reset RAPID**: 258

   The text on the restart button changes when you change the restart type.

4.3.8 Configuring the robot settings

Introduction to the robot settings

Some robot settings can be modified from PickMaster.

RAPID programs can be imported, edited, and exported.

Configuring the robot settings

Use this procedure to configure the robot settings.

1. Right-click a robot and select **Settings**.

   The **Robot Settings** dialog is opened.
2. To modify the speed, enter a value in the **Speed** box.
3. To run a service routine, select a routine in the **Routine** box and click **Execute**.

   When PickMaster is started, the **Routine** list is generated with the service routines that are defined in the module *PPASERVICE*. It is only possible to execute the routines when the controller is in paused mode with the RAPID-program running.
4. To set a variable value, select the variable and enter the value in the **Service Variables** part. Then click **Apply**.

   The variables are defined in the module PPASERVICE that can be found in the standard RAPID-program. When PickMaster is started the **Variable** list is generated with the names of the variables that are defined. This is possible when the project is running and when the controller is in manual mode.

Configuring RAPID programs

Use this procedure to configure the RAPID program for a robot in the project.

1. Right-click a robot and select **RAPID Program**.

   The **RAPID Program** dialog is opened.
2. To import a program, click Import and then locate the program.

   > **Note**
   >
   > The RAPID program is saved in the project and is not a reference to a file. This means that if the original RAPID program file is changed, the imported program in the project is not updated!
3. To export the program, click **Export**.
4. To edit the program, click **Edit**.

   The program is opened in a text editor. See *RAPID programs on page 281*.

4.3.9 Vision modeling

Introduction to vision modeling

There are three different tools available for generating models in a project. The three tools are:

- *Geometric PatMax* which is a pattern recognition tool. See *Configuring a geometric model with PatMax on page 114*.
- *Blob* which is a detection of two-dimensional shapes within images. See *Configuring blob models on page 121*.
- *Inspection tool* (Inspection II) which makes it possible to combine the *PatMax*, blob, histogram, Caliper, and external tool to generate a model. See *Configuring inspection models on page 127*.

Classification of items

Items identified by vision models can be classified as either accepted or rejected. These two types can be distributed to different work areas and be given different item type values accessible from the RAPID program. Item classification can be done by *PatMax*, *Blob*, and the *Inspection tool*.

Vision model parameters in item targets

Item targets identified by a vision model can store a selection of upto 5 vision model parameters in the components `Val1`, `Val2`, `Val3`, `Val4`, and `Val5`. These parameters can be accessed in user hooks and in the RAPID program.

Item targets identified by an *inspection model* can store a selection of parameters from the alignment model and from the included subinspection models.

For each kind of vision model, a *target storage* can be selected for some vision parameters.

Related information

Source type and models on page 104.

Configuring a geometric model with PatMax on page 114.

Configuring blob models on page 121.

Configuring inspection models on page 127.

4.3.10 Configuring a geometric model with PatMax

Introduction to the geometric model PatMax

PatMax is a pattern location search technology. This tool measures:

- Position of the pattern.
- Size relative to the originally trained pattern.
- Angle relative to the originally trained pattern.

PatMax differs from other pattern location technologies as it is not based on pixel grid representations that cannot be efficiently and accurately rotated or scaled. Instead, *PatMax* uses a feature based representation that can be transformed quickly and accurately for pattern matching.

When creating a pattern the following things should be considered.

- Select a representative pattern with consistent features. Reduce needless features and image noise. Train only important features. If necessary, export the image and use an external program to erase noise.
- Larger patterns will provide greater accuracy because they contain more boundary points to resolve at run-time.
- High frequency features are more significant at the outer edges of the pattern.

Models can be classified with the function *Inspection I*. A model can either be defined as accepted or rejected, see *Configuring items on page 91*.

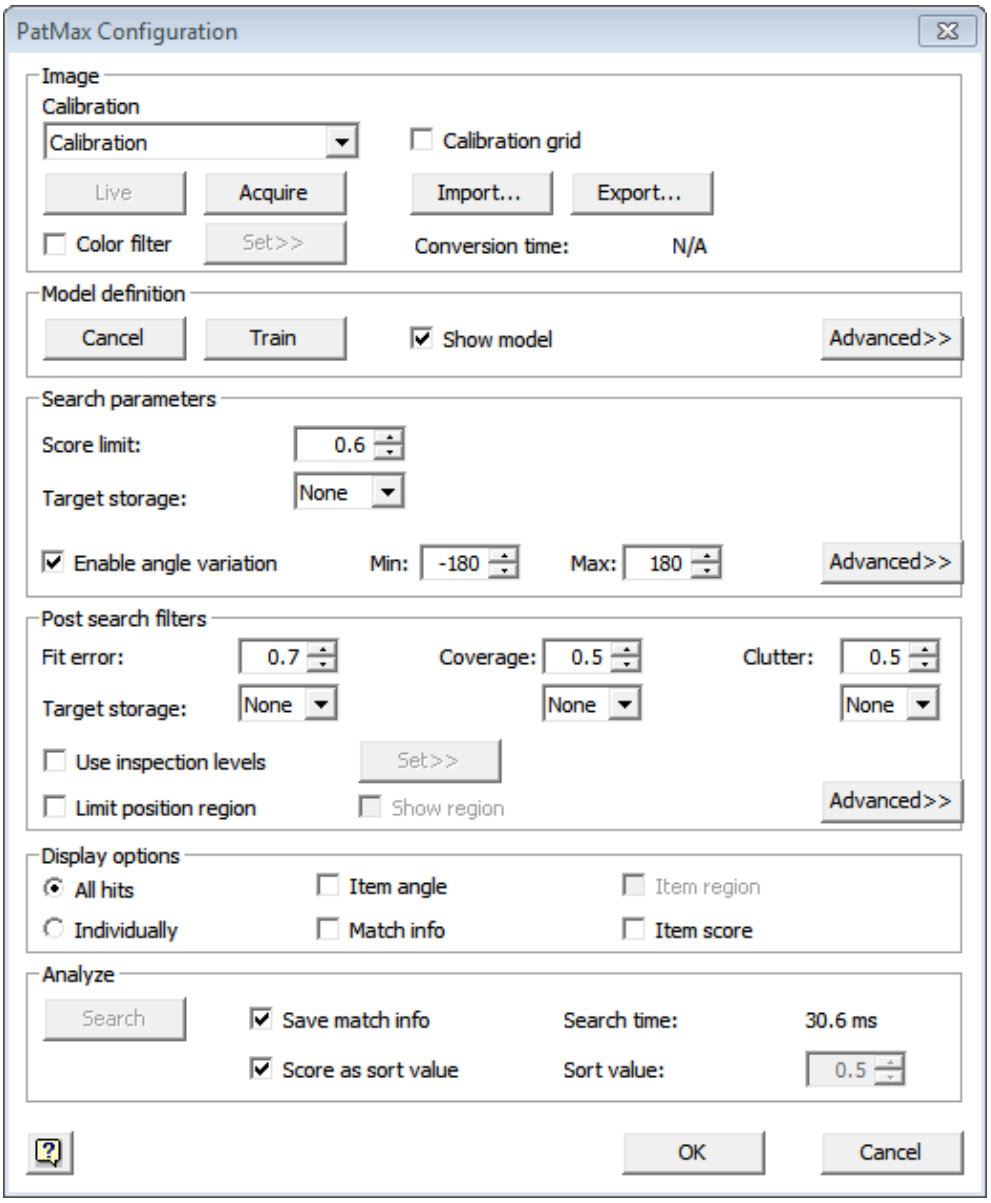
To increase the contrast in images where parts have similar grayscale tone, it is possible to use color filtering. See *Using color vision on page 135*.

Even though this is a 2D pattern matching tool that only determines the x- and y-coordinates and the angle of an object, it is also possible to get height information. There is also functionality to compensate for inaccuracy in 2D for objects that are not located in the calibration plane (parallax compensation). See *Working with products of varying height (2.5D vision) on page 142*.

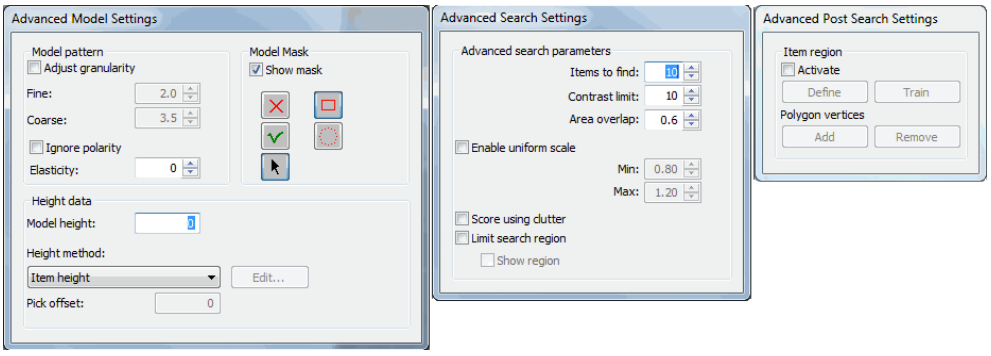
There are several parameters that can be adjusted to make an efficient model. The configuration is done in the **PatMax Model Configuration** dialog and the result is displayed in the **PatMax Image** and the **PatMax Result** dialogs:

Continues on next page

Illustration PatMax Model Configuration



en0900000539



en0900000656

Continues on next page

4.3.10 Configuring a geometric model with PatMax
Continued

Configuring a PatMax vision model

Use this procedure to configure a *PatMax* vision model.

1. In the **Position Source Configuration** dialog, click **Geometric**.

   The **PatMax Model Configuration** dialog is opened.

2. In the **Image** part, click **Live**, **Acquire**, or **Import** to get an image. Select a calibration from the **Calibration** list. Select the **Calibration grid** checkbox to display help lines for the coordinate system.

   The help lines can be moved with the mouse to make it easier to train a pattern.

3. If color filtering should be used select the **Color filter** checkbox to enable the filter. Click **Set** to configure the filter. See *Using color vision on page 135*.

4. In the **Model definition** part, define a model for the pattern using an image in front of the camera or using an imported image. The selected calibration will be used.

   Note

   When importing a vision model it is required to enter model configuration and re-select which calibration to use from the calibration drop-down menu. This is required even if there is only one calibration defined. If this is not performed then further actions may produce the error `No valid calibration for the PatMax model`.

   a. Click **Define** to define a model. Drag the rectangle so it covers the pattern and move the cross to the desired pick/place position. To maintain the greatest accuracy, the pick/place position should be placed close to the center of the trained pattern.

   b. Click **Train** to train the pattern.

   c. Select **Show Model** to show the features of the trained models in the search image.

   d. If needed, click **Advanced** to access more model settings.

   e. Increase the value of **Elasticity** to allow for any expected non-linear shape distortion, for example, for organic products and so on. The value represents the maximum distance between a trained feature and a matched feature in pixels. The lower limit is 0 and upper limit is 25. This setting is useful for products of varying shape.

   f. Select **Ignore polarity** to ignore if the features are dark on bright or bright on dark.

      Note

      PatMax will not care if a product is light on a dark background or dark on a light background. This is useful when the background is, for example, a grid.

Continues on next page

g Click **Adjust Granularity** to define the levels in the **Fine** and **Coarse** boxes. Granularity is a radius of influence, in pixels, which determines the detection of a feature in a pattern. *PatMax* locates patterns in the search image by first searching only for large features. After locating one or more pattern instances, it uses smaller features to determine the precise transformation between the trained pattern and the pattern in the search image. *PatMax* uses the same range of granularity that is computed when training the pattern to detect features in the search image. The granularity parameters *fine* and *coarse* are auto-selected when training the pattern and often these values are the best. These can also be set manually. The lower limit is 1 and upper limit is 25.5.

h Click **Show mask** to edit a mask that determines what parts of the model region to ignore. The effect can be verified by pressing the **Train** button and checking the **Show Model** checkbox. This tool is useful when the training scene contains a lot of clutter (reflections and so on). For more information regarding the effect of the edit controls, hover the mouse pointer over the corresponding control. Click **Adjust Granularity** to define the levels in the **Fine** and **Coarse** boxes.

The **Height data** part contains settings that allow the vision system to calculate the height of a product. It can also compensate for incorrect x- and y-coordinates when the pattern of the item is not located in the specified calibration plane (parallax compensation). See *Working with products of varying height (2.5D vision) on page 142*.

5 In the **Search parameters** part, set parameters to limit the search procedure and the analysis time.

Score Limit indicates how closely the found item matches the trained model. A score of 1 indicates a perfect match while a score of 0 indicates that the pattern does not match at all. The higher a score threshold is defined the faster *PatMax* will be able to perform a search.

Angle variation defines the acceptable rotation for the items. If an item has a rotation outside the valid range it will be discarded by the vision system. Default +/- 180 degrees.

If more settings are required, click **Advanced** to open the **Advanced Search Settings** dialog where the following settings are found:

Items to Find is the number of items that is expected to be present in the image. If there are more items present in the image these will not be reported by *PatMax*.

Contrast Limit defines the minimum image contrast of each item that is found in the image. The contrast is the average difference in gray-level values for all of the boundary points that *PatMax* matched between the trained model and the found item in the search image. *PatMax* considers only items with a contrast value that exceeds the contrast limit.

Area Overlap defines how much multiple patterns in the image are allowed to largely overlap each other. *PatMax* assumes that these patterns actually represent the same item in the image. When two patterns overlap by a

Continues on next page

percentage greater than the area overlap threshold they are treated as a single pattern.

Uniform Scale is a threshold that accepts hits that differ in size relative to the taught vision model. A scale value of 1 indicates that there are no differences between the found item and the taught vision model. A value <1 indicates a smaller model.

Score Using Clutter defines a measure of the extent to which the found item contains features that are not present in the trained vision model. By default the *PatMax* analysis ignores clutter when scoring which means that the patterns receive the same score regardless of the presence of extra features. If this checkbox is selected, clutter is included in the calculation of the score. If the application is an alignment application in which the background does not change, **Score Using Clutter** should be selected.

Limit Search Region limits the search area for the *PatMax* analysis. Only objects within this area will be found. A smaller search area will decrease the search time.

> **Note**
>
> When combining **Fine/Coarse Granularity** and **Uniform Scale** a slight difference in the score can appear between design time and runtime. Therefore, the model should be tested in runtime to verify that items are identified as expected.

6 In the **Post search filters** part, define the score values for each pattern in the search image.

Fit Error Limit is a measure of the variance between the shape of the trained pattern and the shape of the pattern found in the search image. If the found pattern in the search image is a perfect fit for the trained pattern, the fit error is 0.

Coverage Limit is a measure of the extent to which all parts of the trained pattern are also present in the search image. If the entire trained pattern is also present in the search image, the coverage score is 1. Lower coverage scores indicate that less of the pattern is present. This parameter can be used to detect missing features.

Clutter limit is a measure of the extent to which the found pattern contains features that are not present in the trained pattern. A clutter of 0 indicates that the found pattern contains no extra features. A clutter score of 1 indicates that for every feature in the trained pattern there is an additional extra feature in the found pattern. The clutter can exceed 1.0.

Inspection Levels - Inspection I, this inspection is also called *Inspection I* in PickMaster. With this function it is possible to classify the found models into two categories. A model can either be classified as accepted or rejected. An accepted model has better search results than the rejected model. The item type number is defined for the accepted and rejected model in the **Item** dialog, see *Configuring items on page 91*. An item type can be read in the RAPID code, see *RAPID programs on page 281*.

Continues on next page

Click **Set** to open the **Inspection Parameter** dialog. All models that fulfill the conditions specified for the search parameters and the post filters will be classified. Select **Use Inspection Levels** and click **Set** to open the dialog and define the parameter that will divide the found items into the two categories. If **Use inspection levels** is not selected all found models are classified as an accepted model.

For **Score**, **Contrast**, and **Coverage**, items with a value larger than the defined value in **Inspection Parameter** will be defined as accepted.

For **Angle** and **Uniform Scale**, items with a value between the defined values in **Inspection Parameter** will be defined as accepted.

For **Fit Error** and **Clutter** a value less than the defined one will be classified as accepted.

Limit Position Region defines if the *PatMax* analysis is done on the whole image. Objects found within this area will be handled as normal. Object found outside this area will be discarded.

Click **Advanced** to open the **Advanced Post Search Settings** dialog. To define an item region, select the **Activate** checkbox and click **Define**. Adjust the polygon showed around the found object using vertices. Then click **Train**. The polygon can have 2 to 16 vertices.

7 In the **Display options** part, select the type of information to display in graphics.

Item angle displays the angle of the item that will be sent to the robot. This angle is relative to the trained model.

Match Info displays the quality of the matched boundary points in the search image. **Save Match Info** must be selected before doing the analysis. Boundary points drawn in:

- Red are poor matches.
- Yellow are fair matches.
- Green are good matches.

Item score displays the score for the selected item in the image window.

Item region displays the regions in the image window. Red regions indicate an overlap and the corresponding hits will be considered as discarded.

Note

The item region cannot be combined with a user hook since the item region is ignored when using a user hook. The information about the item region of a vision model is removed when the positions are sent to the user hook. Even if the positions are not changed in the user hook the information about the item region is lost. This means that the overlapping items will not be removed.

Continues on next page

4.3.10 Configuring a geometric model with PatMax
Continued

8 In the **Analyze** part, click **Search** to analyze the image. If needed, define sort value.

The result is displayed as an image with numbered hits in the **PatMax Image** dialog, and a corresponding result list in the **PatMax Result** dialog.

Model hits are normally classified as accepted. If inspection is used, hits can be classified as either accepted or rejected. See *Configuring items on page 91*. Hits that do not fulfill all the requirements or hits with overlapping regions will not be accessed by any robot and are classified as discarded. The hits shown in the result list are marked with an icon identifying its classification. For hits that are not accepted, the parameter that failed is marked with either red or blue in the result list.

Save Match Info is used to save the match information for every found model. Searches where match info is generated and stored takes more time. To show the match information, select the **Match Info** checkbox in the **Display options** part.

Sort value is used if there is more than one hit for the same item. Only the hit with the highest sort value will be sent to the robot controller. The sort value can be set individually for all models or the *PatMax* score can be used by selecting **Score as sort value**. See *Configuring position sources on page 99*.

Search Time displays the time it takes to analyze the image in .

9 Click **OK**.

Note

Items located after a search operation in the PatMax configuration window is presented as discarded due to item region overlap even if they are actually rejected due to another parameter (fit error, clutter, and so on). This happens only if the item region is activated and the item regions overlap with each other in runtime. However, the discarded items are removed before applying the item region.

PatMax parameters in item targets

The PatMax parameters `Score`, `fit error`, `coverage`, and `clutter` can be selected for the target storage.

Related information

Configuring items on page 91.

Using color vision on page 135.

Working with products of varying height (2.5D vision) on page 142.

RAPID programs on page 281.

Configuring position sources on page 99.

4.3.11 Configuring blob models

Introduction to blob models

The simplest kinds of images that can be used for machine vision are two-dimensional shapes or blobs. Blob analysis is the detection of two-dimensional shapes within images. It finds objects by identifying groups of pixels that fall into a predefined grayscale range.

This kind of analysis is well suited for applications where:

- Objects vary much in size, shape, and/or orientation.
- Objects are of a distinct shade of gray not found in the background.

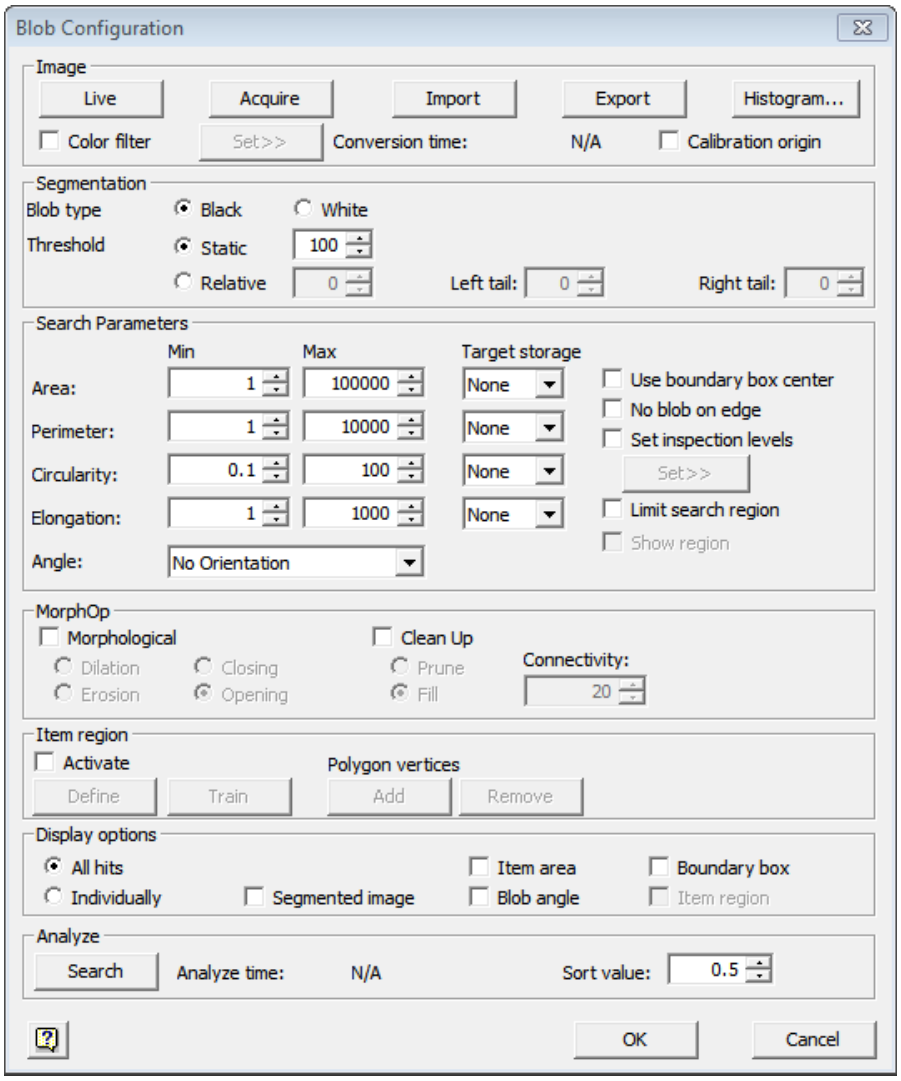
Blob analysis works best with images that can be easily segmented into foreground and background pixels. Typically, strong lighting of scenes with opaque objects of interest produces images suitable for an analysis like this.

To increase the contrast in images where parts have similar grayscale tone, it is possible to use color filtering. See *Using color vision on page 135*.

Continues on next page

4.3.11 Configuring blob models
Continued

Illustration Blob Configuration



xx1500001659

Configuring a blob vision model

Use this procedure to configure a blob vision model.

1 In the **Position Source Configuration** dialog, click **Blob**.

The **Blob Configuration** dialog is opened.

2 In the **Image** part, click **Live**, **Acquire**, or **Import** to get an image. Select the **Calibration origin** checkbox to display help lines for the coordinate system. Click **Histogram** to display a graph of the pixel distribution in the acquired image.

If color filtering should be used, select the **Color filter** checkbox to enable the filter and click **Set** to configure the filter. See *Using color vision on page 135*.

In the histogram, the horizontal axis represents the pixel values of the image from black to white (0 to 255). The vertical axis indicates the number of pixels at each value. There are also a number of statistical values representing

Continues on next page

various properties of the histogram such as the value of the darkest and brightest pixels as well as the mean value and standard deviation of the collection of pixels. For more information, see *Histogram on page 130*.

3 In the **Segmentation** part, select segmentation method and blob type.

Segmentation is the division of the pixels in an image into object pixels and background pixels. Typically objects are assigned a value of 1 while background pixels are assigned a value of 0.

Static method uses gray values to divide blob pixels and background pixels. All pixels with a grayscale value below the threshold are assigned as object pixels, while all pixels with values above the threshold are assigned as background pixels.

Relative method uses a relative threshold expressed as the percentages of the total pixels between the left and right tail to divide blob pixels and background pixels. Tails represent noise-level pixels that lie at the extremes of the histogram (the lowest and the highest values).

Static is faster than relative segmentation because the gray levels corresponding to the percentages do not have to be computed. Static segmentation can test for absence of a feature in a scene, whereas relative segmentation will always find a blob in the scene.

Note

Tune the blob tool by pressing **Search** and the blob algorithm lists all the blobs. Adjust the size threshold limit to filter out blobs that are too large or too small. Tune other parameters if necessary..

4 In the **Search Parameters** part, define the values for the feature.

Area is expressed in mm^2.

Perimeter is expressed in mm.

Circularity defines the circularity. A value of 1 means perfectly circular and completely filled (no holes).

Elongation is the ratio of the feature's second moment of inertia about its second principal axis to the feature's second moment of inertia about its first principal axis.

Angle defines how the found item is sent to the controller.

- **No Orientation** means that the found item is sent to the controller with angle 0 (zero).
- **First Principal Axis** means that the found item is sent down with the angle around the first principal axis. The angle is relative to the x-axis and can be ±90 degrees.

Use boundary box center defines if the position of a blob will be at the center of its boundary box instead of at its center of mass.

No Blob On Edge defines if a blob connected to the edge of the search area should be reported.

Continues on next page

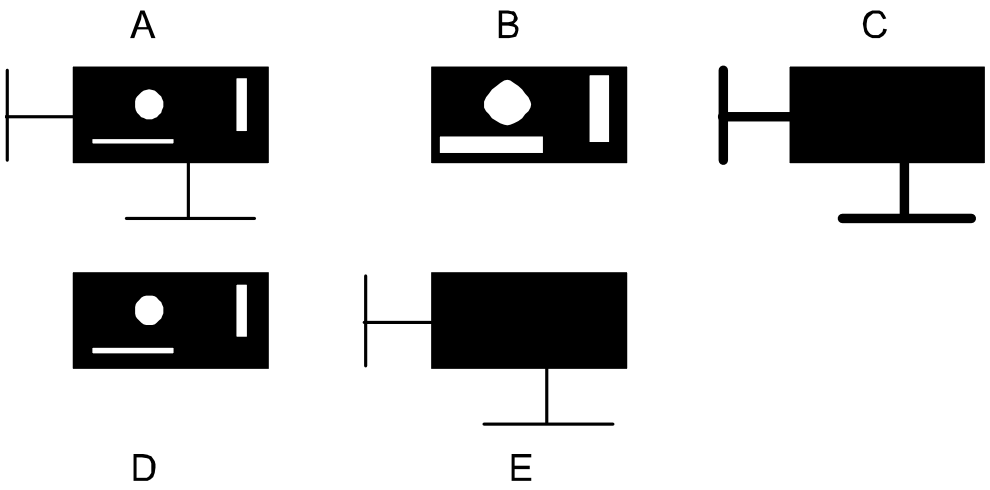
4.3.11 Configuring blob models
Continued

Use Inspection Levels defines if the found models should be classified. See *Configuring items on page 91*. The item type can be read in the RAPID code, see *RAPID programs on page 281*. Select **Set levels** to open the **Blob Inspection Parameters** dialog.

If **Use Inspection Levels** is not selected all found models are classified as accepted. All models that fulfill the conditions specified for the **Search Parameters** will be classified.

Limit Search Region limits the search area for the blob analysis. Only objects within this area will be found.

5 If needed, in the **MorphOp** part, select the **Morphological** and/or **Clean Up** checkboxes and define the settings.



xx0900000542

A	Original
B	Erosion
C	Dilation
D	Opening
E	Closing

Morphological settings:

- **Erode** reduces or eliminates object features, increases the thickness of holes within an object. This operation replaces each pixel in the image with the maximum value of the pixels and each of its eight vertical and horizontal neighbors.
- **Dilation** reduces or eliminates holes within an object, increases the thickness of an object's features. This operation replaces each pixel in the image with the minimum value of the pixel and each of its eight vertical and horizontal neighbors.
- **Closing** eliminates holes. Preserves small features. An erosion operation is applied to the image, followed by a dilation operation.
- **Opening** preserves holes. Eliminates small object features. A dilation operation is applied to the image, followed by an erosion operation.

Continues on next page

Clean up settings:

- **Prune** is used to ignore, but not remove features, that are below a specified size (connectivity size). When an image is pruned of all features below a certain size, the blob measures returned for the blob that enclosed the pruned features are computed as though the pruned features still existed, but the pruned features themselves are not counted.
- **Fill** is used to fill in pruned features with gray values from neighboring pixels on the left. The pixels value that is used to fill the feature is the value of the pixel to the immediate left of the feature being filled. As each row of pixels in the feature is filled, the pixel value to the immediate left of that row of pixels is used as the fill value for that row.
- **Connectivity** defines the minimum size (in pixels) that a blob can have to be considered. Is used with either prune or fill.

6 In the **Item region** part, select the **Activate** checkbox and click **Define**. Adjust the polygon showed around the found object using vertices. Then click **Train**.

The polygon can have 2 to 16 vertices.

7 In the **Display options** part, select **Segmentation image** to display the processed image. Select how the result will be displayed.

- **Item angle** displays the angle of the item that will be sent to the robot.
- **Item score** displays the score for the selected item in the image window.
- **Item area** displays the area of the blob in the image window.
- **Boundary box** displays the minimum horizontal rectangle that contains the whole blob.
- **Item region** displays the regions in the image window. Red regions indicate an overlap and the corresponding hits will be considered as discarded.

> **Note**
>
> Item regions cannot be used, if the item positions are modified by a user hook.

8 In the **Analyze** part, click **Search** to analyze the image. If needed, define sort value.

The result is displayed as hits in the **Blob Result** dialog.

Model hits are normally classified as accepted. If inspection is used, hits can be classified as either accepted or rejected. See *Configuring items on page 91*, for information about classification. Hits that do not fulfill all the requirements or hits with overlapping regions will not be accessed by any robot and are classified as discarded. The hits shown in the result list are marked with an icon identifying its classification. Rejected hits are shown in blue.

Sort value is used if there is more than one hit for the same item. Only the hit with the highest sort value will be sent to the robot controller. The sort

Continues on next page

4.3.11 Configuring blob models
Continued

value can be set individually for all models. See *Configuring position sources on page 99*.

Search Time displays the time it takes to analyze the image in ms.

9 Click **OK**.

Blob parameters in item targets

The blob parameters `Area`, `perimeter`, `circularity`, and `elongation` can be selected for the target storage.

Related information

Using color vision on page 135.

Histogram on page 130.

Configuring items on page 91.

RAPID programs on page 281.

Configuring position sources on page 99.

4.3.12 Configuring inspection models

Introduction to inspection models

PickMaster inspection models make it possible to combine several models of *PatMax*, blob, histogram and *Caliper*. This is sometimes referred to as *Inspection II*.

An inspection model always consists of an alignment model. The alignment model can either be a *PatMax* or blob and works as the reference for the inspection model. It is this model's position and rotation that is the pick/place position and rotation for the item.

Inspection areas are defined relative to the alignment model and either blob, *PatMax*, histogram, *Caliper*, or external analysis can be done within each of these areas. Conditions such as number of found items and location relative to the alignment model can be set.

For a found item to be classified as accepted, all inspection areas and the alignment model must be classified as accepted. If one of the inspection areas does not fulfill the given conditions the corresponding item is classified as rejected.

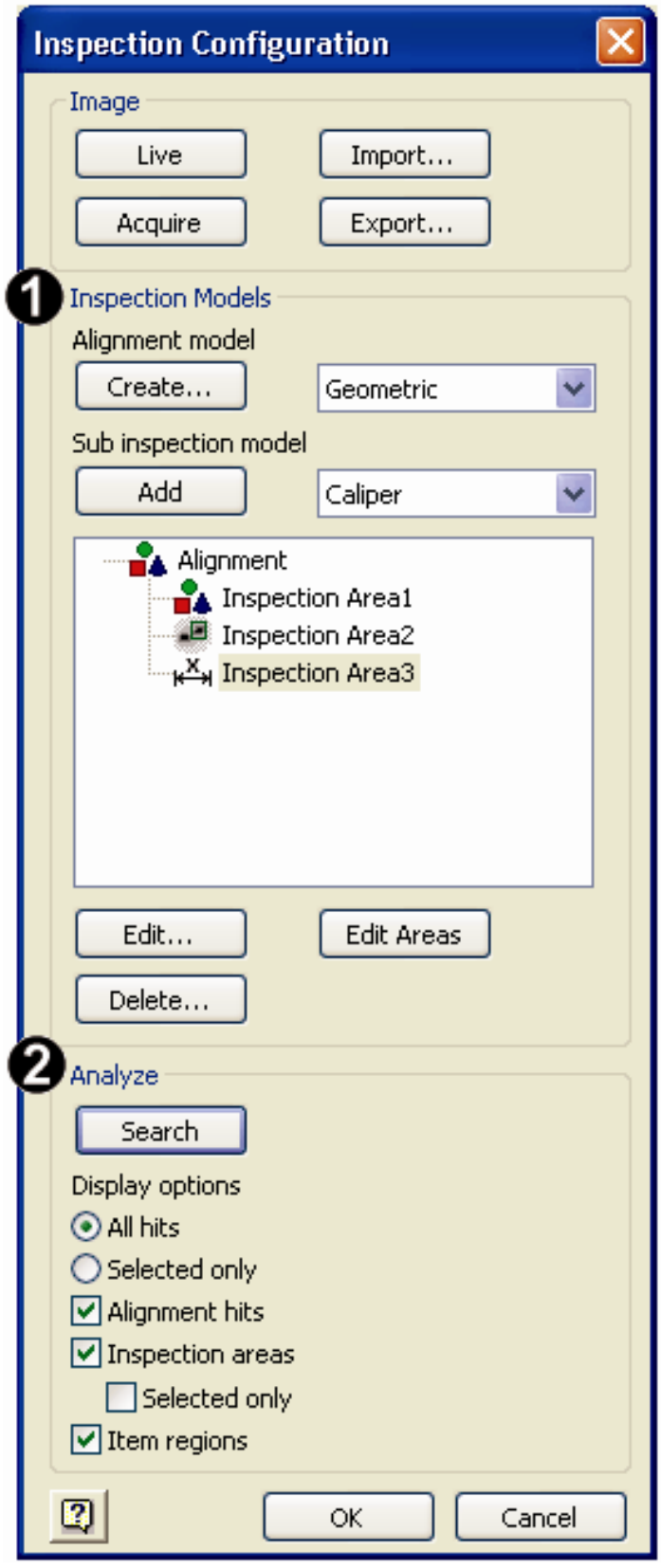
Tip

A more advanced accept or reject decision can be implemented with a user hook. By adding a selection of upto 5 vision parameters for storage in the item targets, a user hook can implement an advanced accept or reject decision based on the selected parameters and other item target data (for example, position and orientation). For example, the combination of 5 measured caliper distances can be evaluated for the accept or reject decision instead of having a fixed allowed maximum or minimum interval for each caliper distance.

Continues on next page

4.3.12 Configuring inspection models
Continued

Illustration Inspection Configuration



en0900000543

Continues on next page

Configuring inspection models

Use this procedure to configure inspection models.

1. In the **Position Source Configuration** dialog, click **Inspection**.

   The **Inspection Configuration** dialog is opened.
2. In the **Image** part, click **Live**, **Acquire**, or **Import** to get an image.
3. In the **Inspection model** part, define the relationships between the alignment model and its corresponding inspection areas.

   The created models are shown in a tree view. To change a name, select the model and press **F2**.

   Alignment model defines the position and orientation of any found items. If an alignment model is already defined, it will be replaced. For more information on the alignment model configuration dialog, see *Vision modeling on page 113*.

   Sub inspection model adds inspection areas to an alignment model. See *Sub inspection models on page 129*.

   Edit opens the configuration dialog for the selected model. When an existing alignment model is modified the relations to the inspection areas must be retrained.

   Delete is used to delete the selected model and corresponding inspection area.

   Edit Areas shows all inspection areas. The areas can be rearranged for the selected sub inspection model.
4. In the **Analyze** part, click **Search** to analyze the image.

   The result is displayed as an image with numbered hits in the **Inspection Image** dialog, and a corresponding detailed list in the **Inspection Result** dialog.

   Display options defines how the results are displayed.
5. Click **OK**.

Sub inspection models

Introduction

Sub inspection models are used to add inspection areas to an alignment model. Each area uses a specified sub inspection model. The inspection area defines where the sub model is to perform its analysis relative the alignment model. The areas are shown in the image and should be moved and resized to cover the area to analyze.

Sub inspection models are configured in their own dialogs. When testing a sub inspection model the alignment hit is shown in the image window together with the corresponding inspection area. Sub inspection models only analyze the part of the image defined by its inspection area. These models are accessed for configuration within the **Position Source Configuration** dialog box. For more information, see *Source type and models on page 104*

Continues on next page

4.3.12 Configuring inspection models
Continued

Geometric

A geometric sub inspection model is configured in the same way as a *PatMax* model. See *Configuring a geometric model with PatMax on page 114*. In addition, the relative positions of the found items and the corresponding alignment hit must be trained.

Required hits defines the number of hits with the sub inspection model within the inspection area that are required for the result to be considered as accepted.

Deviation limits defines the allowed deviations from the trained positions.

After a search and the items are found within the inspection area their positions must be trained. The relative positions are listed as **xDiff**, **yDiff**, and **AngleDiff**.

Click **Train** to save the positions of the found items relative to the alignment hit.

Geometric subinspection parameters in item targets

The parameter `Number of hits` can be selected for the target storage.

Blob

A blob sub inspection model is configured in the same way as a blob model. See *Configuring blob models on page 121*. In addition, the number of required hits must be configured.

Required hits defines the number of hits with the sub inspection model within the inspection area that are required for the result to be considered as accepted.

Blob subinspection parameters in item targets

The parameter `Number of hits` can be selected for the target storage.

Histogram

The histogram tool measures the color or the gray level within any given area. While using a monochrome camera the histogram tool measures the gray level within a given area. Similarly, if a color camera is used each of the three color channels (Red, Green, and Blue) is measured separately. The histogram tool is useful when the objects to be identified and classified have similar shapes but different colors.

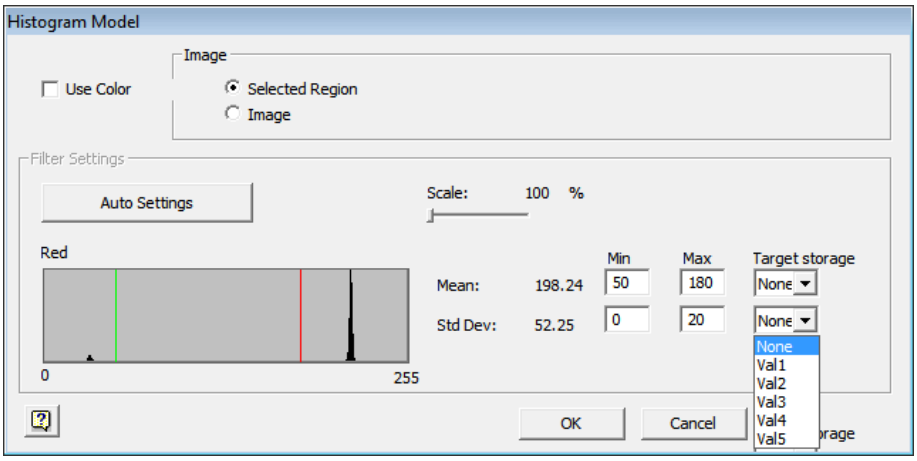
The inspection area for a histogram sub inspection model is graphically represented as a circle. But the area used in the histogram analysis is actually a square aligned with the image but enclosed by the inspection area.

1 On the **Blob Configuration** dialog box, click Histogram button. For more details, see *Configuring blob models on page 121*.

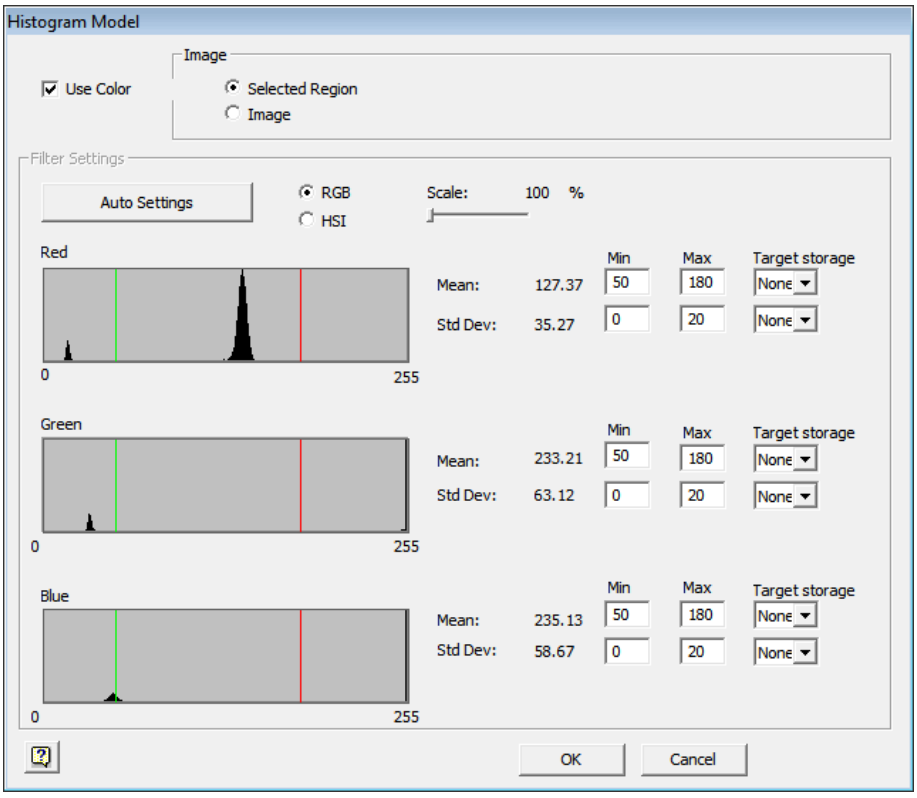
   The **Histogram** dialog appears. It displays a graph of the pixel distribution of the acquired image.In monochrome mode the **Histogram** dialog displays a graph of the pixel distribution of the acquired image. The horizontal axis represents the pixel values of the image from black to white (0 to 255). The vertical axis indicates the number of pixels at each value. While using color vision, three graphs each representing the contents of the different color

Continues on next page

channels (RGB or HSI), are displayed. For more information, see *Using color vision on page 135*.



en1200000684



en1200000685

2 Select **Selected Region** to display the histogram and statistic for the inspection area.
3 Select **Image** to display the histogram and statistic for the whole image.
4 Move the **Scale** to scale the histogram (100%-1000%).
5 Press **Auto Settings** to automatically get a appropriate range limits(**Min.** and **Max.** values) for the histogram. Alternatively, the **Min.** amd **Max.** values can be set manually by sliding the red and green bars across the histogram or

Continues on next page

by simply entering values into the text boxes. For a product to be accepted, both the standard deviation and the mean value have to be within the specified limits. When using color vision the histograms for all channels must fall within the limits.

To classify the inspection area as accepted or rejected the histogram tool evaluates two different magnitudes within the specified region:

Mean defines the min and max value for the inspection model. If the inspection area has a mean value less than min or higher than max the inspection area will be classified as rejected.

Std dev is a statistical measure that illustrates how closely all the various pixel values are clustered around the mean value. An even color tone gives a narrow histogram with low standard deviation while a speckled pattern gives a wide histogram and a high value for **Std dev**.

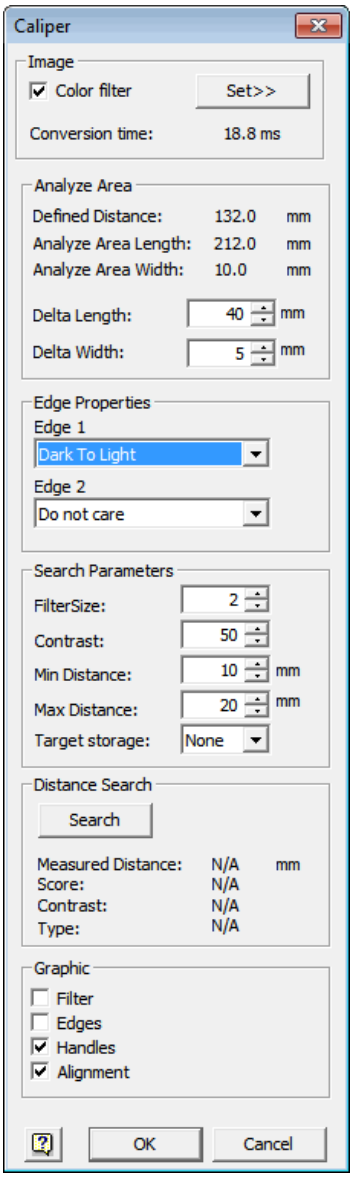
Histogram subinspection parameters in item targets

The `Mean` and `standard deviation` parameters can be selected for the target storage.

Continues on next page

Caliper

The *Caliper* tool identifies edges and measures the distance between them. The analysis is only done within the corresponding inspection area. To increase the contrast in images where parts have similar grayscale tone, it is possible to use color filtering. For more information, see *Using color vision on page 135*



en0900000654

To make a *Caliper* analysis a rectangle is defined around the search line.

Defined Distance is the distance between the end points of the green line located in the **Caliper Image** dialog. Move the line so the end points are located on the edges of the area.

Analyze Area Length is the length of the rectangle within which the Caliper analysis will be performed. To increase the **Analyze Area Length** either increase the **Delta Length** value or resize the **Defined Distance** line.

Continues on next page

4.3.12 Configuring inspection models
Continued

Analyze Area Width is the width of the rectangle within which the Caliper analysis will be performed. To increase the **Analyze Area Width** increase the **Delta Width** value.

Delta Length define the extra mm to add to the **Defined Distance** to get an **Analyze Area Length**.

```
Analyze Area Length=2*Delta Length + Defined Distance
```

Delta Width defines the width of the analyze area.

```
Analyze Area Width=2*Delta Width
```

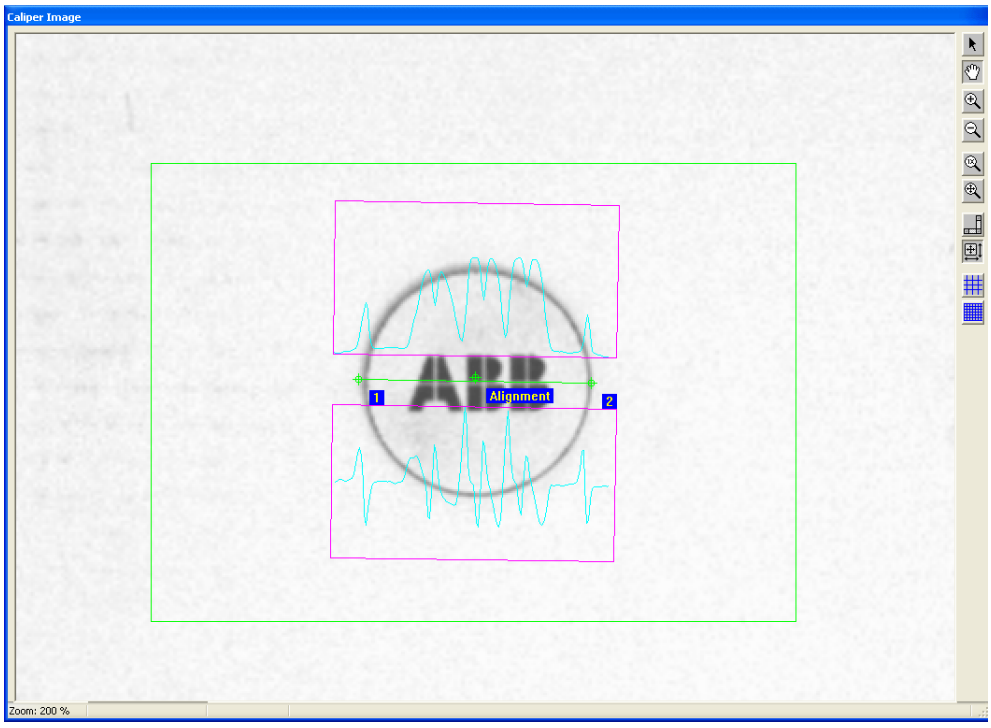
From the analyze area a projection image is created. The operation sums all the information in the analyze area, accentuating the strength of edges that lie parallel to the **Analyze Area Width** and reducing the effects of noise.

Edge Properties defines the polarity of the edge. The polarity is defined as the measure from **Edge1** to **Edge2**.

The **Search Parameters** defines filters using a Gaussian curve. The filter controls how the *Caliper* tool removes noises, how it accentuates the peaks of interest in the image, contrast, and distance.

The **Distance Search** is used to search for two edges with the specified distance (**Defined Distance**) and the defined polarity.

The **Graphic** checkboxes define which results should be displayed in the **Caliper Image** dialog.



en0900000655

Caliper subinspection paramters in term targets

The `Distance` parameter can be selected for the target storage.

4.3.13 Using color vision

Introduction to color vision

PickMaster can either be used with monochrome or color cameras. The difference between the two is that an image acquired with a color camera represents each pixel with three 8-bit values (decimal 0-255) instead of only one 8-bit value for monochrome (grayscale) images. In a monochrome image the 8-bit value represents the gray level from white to black, whereas in a color image the three values represent the content of three separate color channels. These three channels represent red, green, and blue (color space RGB) or hue, saturation, and intensity (color space HSI). Which color space to work with, depends on the content of the image.

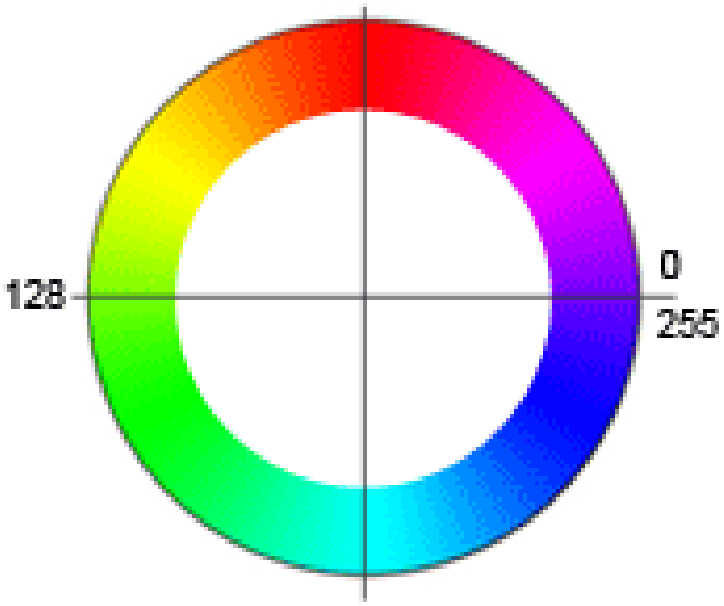
Color spaces

When working with RGB the color of each pixel is represented by its content of red, green, and blue. The numerical representation is straightforward for the three base colors - red (255, 0, 0) green (0, 255, 0), and blue (0, 0, 255). However, it can be difficult to understand the composition of other mixed colors.

HSI is a color space that is more easily translated to the human perception of colors.

- Hue: The location of the color on the on the electromagnetic spectrum. See graphic below.
- Saturation: The purity of the color.
- Intensity: The brightness of the color.

Because the hue spectrum wraps around (both 0 and 255 represent red), it is suitable to display it as a circle.



xx0900000444

Continues on next page

4.3.13 Using color vision
Continued

When using color filtering it is easier to distinguish between colors if they are dissimilar. The level of similarity may be interpreted as the distance between the colors in color space. The difference may be more pronounced in one or the other of the two color spaces and for this reason it is wise to try out filters in both color spaces.

Lighting

Because a color system provides more information about the color contents of an image it is also more sensitive to lighting conditions. It is very important to provide uniform light, that is consistent over time.

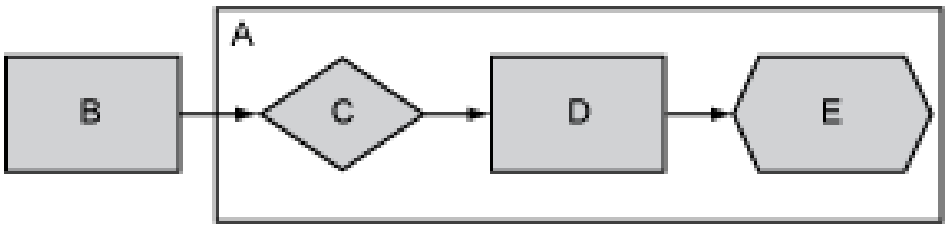
Computer performance

Color vision is very resource consuming; acquisition, warping, and filtering all take more time. It is important to keep the number of cameras and frame rate moderate. The performance limit can vary greatly as it is a combination of the vision task and the computer resources.

Color vision in PickMaster

PickMaster provides color vision in the form of a filter except the histogram. This filter is accessible from the PatMax, Blob and Caliper configuration dialogs, both as standalone, alignment and sub-inspection models. The filter is a pre-processing step which takes place before the object recognition or measurement. Every model can have its own individual filter setting.

The camera acquires a color image, that is converted into a grayscale image by passing it through a color filter, as shown in the following figure. The histogram tool takes a slightly different approach to color vision as it skips the filtering step and instead performs the measurement directly on each of the three color channels of the image.

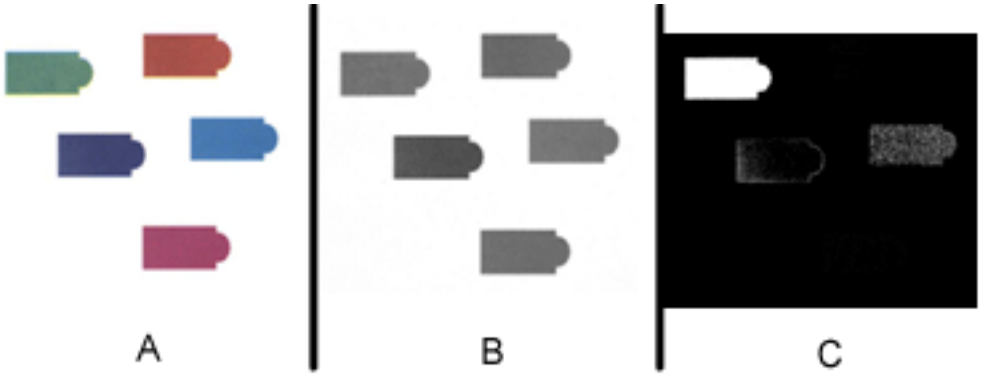


xx0900000445

A	Vision model
B	Color image
C	Color filter
D	grayscale image
E	Object recognition

Continues on next page

The result of the color filter is a grayscale image in which certain colors have been accentuated or attenuated according to the filter settings. The object recognition tools (*Blob*/*PatMax*) operate on this grayscale image.



xx0900000446

A	An image acquired with a color camera.
B	The same scene acquired with a monochrome camera.
C	The color image after having passed through a filter which is set to extract green. This is the image that will be used by *PatMax*/*Blob*.

Prerequisites

The camera must be a color camera.

The color video format must be configured for the camera.

The Cognex vision license must contain the color tool option.

Calibrating the camera's white balance

A camera is delivered with default settings. These include three parameters which represent the white balance of the camera. Depending on the light source, the image can get an undesired color tone. Different light sources emit light of different temperatures (color content) and the camera needs to be color calibrated in order to compensate for this light.

The basic concept is to present the camera with a gray scene, that is a scene that has equal contents of red, green, and blue. The most accurate method is to take a sheet of white paper and adjust the light settings of the camera in order to make the scene appear gray.

Use this procedure to calibrate the white balance for the camera.

1. In the line view, right-click on the camera and select **Edit**.

   The **Camera Configuration** dialog is opened
2. Place a white sheet of paper under the camera. The sheet must cover the whole field of view.
3. Adjust the light settings (aperture or exposure time) to make the scene appear mid-gray. The number of saturated pixels (completely black or white) should be kept to a minimum.
4. Press Calculate. This will calculate the white balance calibration parameters.

Continues on next page

4.3.13 Using color vision
Continued

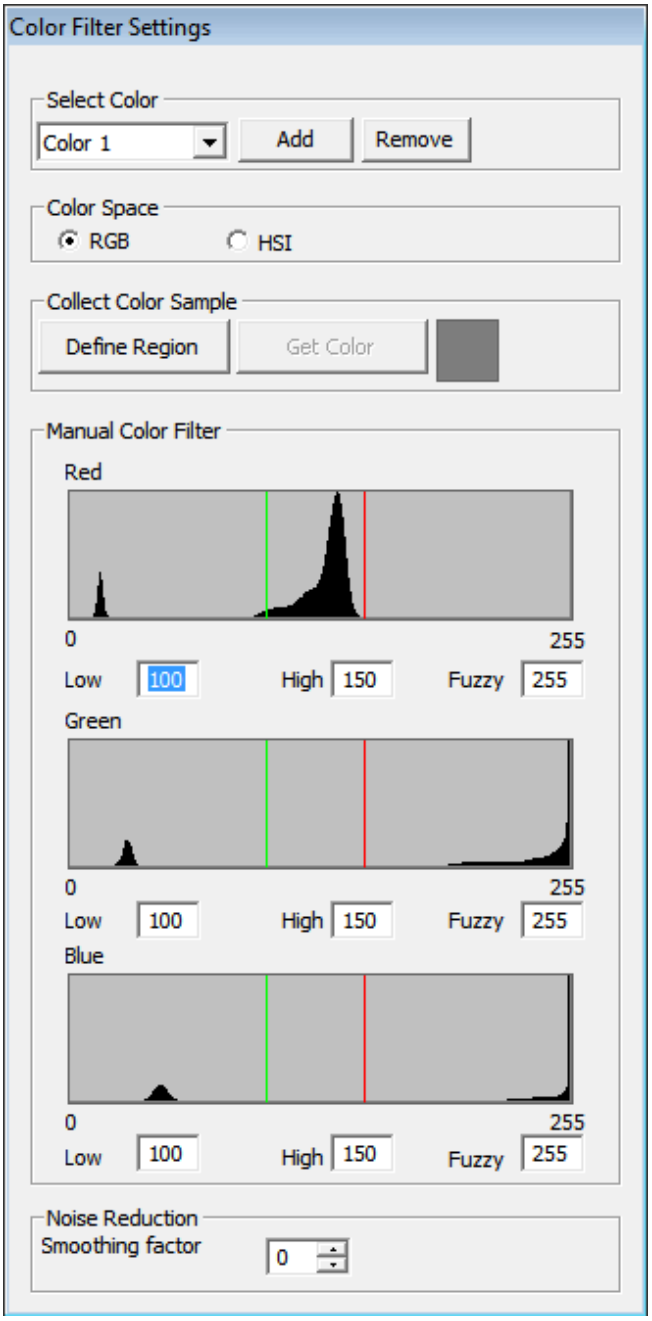
5 Click **Apply**.

The camera's internal settings are now modified. If the calibration is successful the color image and the grayscale image of the white paper sheet should now look the same (gray).

6 Right-click the camera and select **Save**.

The settings are stored in the camera. If the parameters are not saved the camera will loose the calibration next time PickMaster is restarted.

Illustration Color Filter Settings



en0900000657

Continues on next page

Configuring color vision

The *PatMax* and *Blob* configuration dialogs contain a checkbox to enable color filtering (**Color filter**), and a button to display the filter settings (**Set**).

Use this procedure to configure color vision.

1. In the **Image** part of the *PatMax* or *Blob* configuration dialog, select **Color Filter**. This will enable the filter.
2. Click **Set**.

   The **Color Filter Settings** dialog is opened together with a second video window showing the color image.
3. In the **Color space** part, select **RGB** or **HSI**.
4. In the **Collect color sample** part, color samples can be collected from the display to indicate which colors should be enhanced.
   a. Click **Define**. An adjustable rectangle will appear in the color display.
   b. Move/resize the rectangle to indicate what color should pass through the filter. The indicated color range will be converted to white in the output grayscale image. Colors that are dissimilar to the specified color will be converted to black.
   c. Click **Get color** to store this color range.
5. In the **Manual color filter** part, adjust each color channel to improve the result if needed.
   - **Low** specifies the lower limit of the color range that will translate into white pixels in the output image. Minimum is 0 and maximum is 255, except for Hue which has no boundary.
   - **High** specifies the upper limit of the color range that will translate into white pixels in the output image. Minimum is 0 and maximum is 255 except for Hue which has no boundary.
   - **Fuzzy** specifies how colors outside the minimum and maximum thresholds should be filtered to the output grayscale image. A value of 0 indicates that colors outside the range specified by Low and High will be completely removed by the filter - the result is a black and white image. A non-zero value means that colors outside the Low/High range will be weighted in the output image. A higher value produces a smoother grayscale image. Minimum is 0, maximum is 255.
6. If needed, add a new color range to the list in the **Colors** section.

   Each pixel of the output image is computed as the corresponding maximum output pixel of all individual color range filters.
7. If needed, adjust the smoothing factor to reduce noise in the resulting grayscale image.
8. Click **Set filter** to close the dialog.
9. Proceed to define the object recognition model.

Continues on next page

4.3.13 Using color vision
Continued

Tip

Filter ranges should be narrow to provide an output image with high contrast. From an image quality perspective, it is often better to select small homogeneously colored samples and add several ranges to the list of colors.

Tip

Try to filter with both RGB and HSI. Sometimes one may work significantly better than the other.

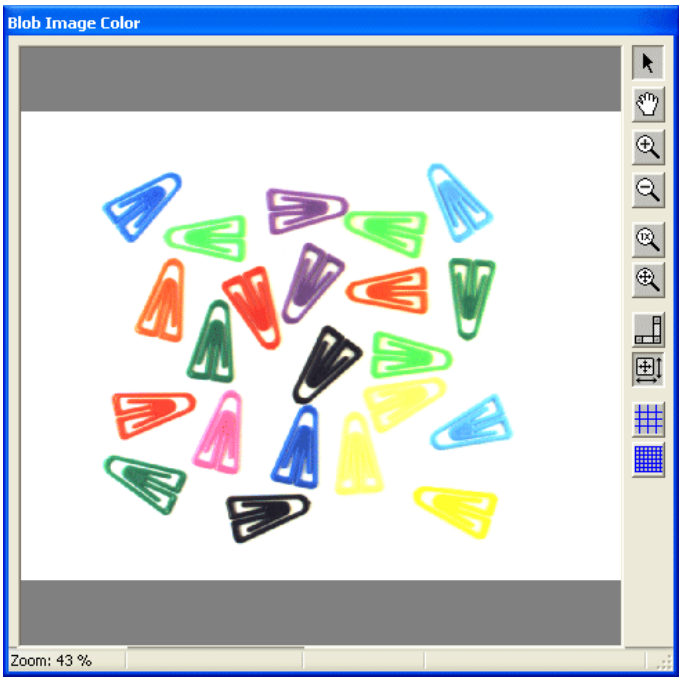
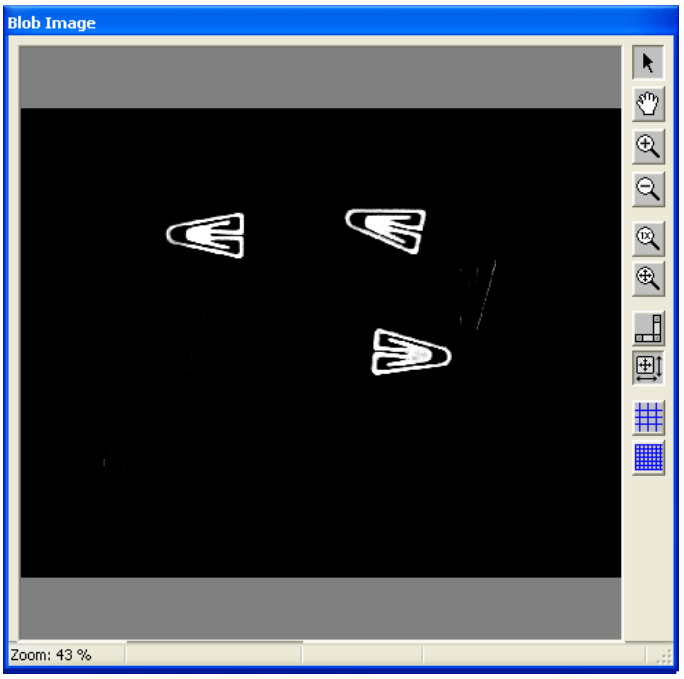
Example 1

This example describes how to locate a part with *PatMax* and inspect the color with *Blob*.

1. Create an inspection model, see *Configuring inspection models on page 127*.
2. Create a *PatMax* alignment model. Use color filtering if contrast needs to be increased, or use the unfiltered monochrome image if there is sufficient contrast.
3. Add a *Blob* sub inspection model.
   a. Select **Color filter** and click **Set**. This opens the **Color Filter Settings** dialog.
   b. Extract the color to be inspected by clicking **Define color**. This filters the desired color into white in the Blob image window.
   c. Click **Set filter** to close the **Color Filter Settings** dialog.
   d. Adjust the Blob settings so as to find the white blob.
   e. If necessary, adjust the settings of the color filter and the Blob analysis.
4. Test the result in the Inspection Configuration dialog.

Continues on next page

Example 2

Color image before filter	 en0900000658
Black and white image after filter	 en0900000659

4.3.14 Working with products of varying height (2.5D vision)

Introduction to height settings

The vision tools in PickMaster normally return a result in a 2D coordinate system: X and Y angle. The trained model is assumed to be located in the plane of the camera calibration.

Working with objects located above or below the calibrated plane raises two questions:

1. At what height is the object located (z-coordinate)?
2. What are the true x- and y-coordinates? The object recognition tools assumes that the object is still located in the calibration plane, and thus will provide coordinates projected on this plane.

To calculate the true x- and y-coordinates the camera's height above the calibration plane, and the product's distance (above/below) to the calibration plane must be known.

Determining the height at which an object is located can be done in two ways with PickMaster. Either the value may be sent to PickMaster from a height sensor, or the height can be estimated by measuring the scale change in relation to the trained object.

The camera's location is provided by performing a Multi-view calibration (a single view calibration is not sufficient). The height of the object, which is necessary to determine the true x- and y-coordinates, can be entered manually or calculated as described above. See *Calibrating the camera on page 75*.

Effectively, the tools described in this section can be used to compensate for parallax error (find the true x- and y-coordinates) and for determining the height of a product.

Prerequisites

The camera must be calibrated with multiple images (Multi-view).

The height settings can only be used together with a *PatMax* standalone model.

Configuring height settings

The height settings belong to a specific model and can only be configured together with *PatMax*.

Use this procedure to configure the height settings.

1. Create a *PatMax* model.
2. In the **Image** part, select calibration from the **Calibration** list. This must be a multi-view calibration.
3. In the **Model definition** part, click **Advanced**. This opens the **PatMax advanced model settings** dialog.
4. In the **Height data** part, enter a value in the **Model height** box.

Continues on next page

This should be the distance between the calibration plane and the plane where the model is located. For example:

- The product to be trained is placed on top of the surface where the calibration plate was located.
- The product is 50mm high, and the model represents the top-side of the product.

The value is in this case 50 mm.

5 Select **Height method** to define how to determine the height of a product.

The following options are available:

- **Item height**: The z-coordinate that will be sent to the robot controller is the height value specified for the item that the model represents. See *Configuring items on page 91*.
- **Vision height**: The z-coordinate is calculated from the scale change (relative to the trained pattern) of the found object. Enable uniform scale must be enabled. The maximum and minimum values must allow for sufficient scale variation.
- **External height**: The z-coordinate is determined by a height sensor. The value is communicated to PickMaster via a UDP port. Click **Edit** to specify the port.

6 Select **Pick offset** to define a direct z-offset from the calculated z-coordinate. This parameter has no effect if **Item height** is selected as method.

Note

The Vision height method may be inaccurate. The accuracy depends on many factors such as camera the camera calibration, camera resolution, model size relative to image etc, thus the obtainable accuracy must be tested for a specific application.

Note

Defining a value for **Model height**, and selecting **Item height** as height method results in parallax compensation but no z-coordinate is calculated by the vision system.

Note

If there is only one object type, and it is always located at the same height, it is most accurate to calibrate the camera at this height instead of using **Model height** to compensate.

Tip

To filter out erroneous height information when using the **Vision height** method, set appropriate scale limits under the **Post search filters** part in the **PatMax Settings** dialog.

Continues on next page

4.3.14 Working with products of varying height (2.5D vision)
Continued

Related information

Calibrating the camera on page 75.

Configuring items on page 91.

4.4 Configuring Remote Integration Services

4.4.1 Introduction

Overview

Remote Integration Service (RIS) is a generic for interfaces that can be used to remote control PickMaster from a remote device (client), such as a PLC. RIS is flexibly designed and can be accessed over a wide range of media (plug-ins), such as HTTP, TCP/IP, RS232, and fieldbus. Some medias, for example, HTTP, offer more functionality than others.

Note

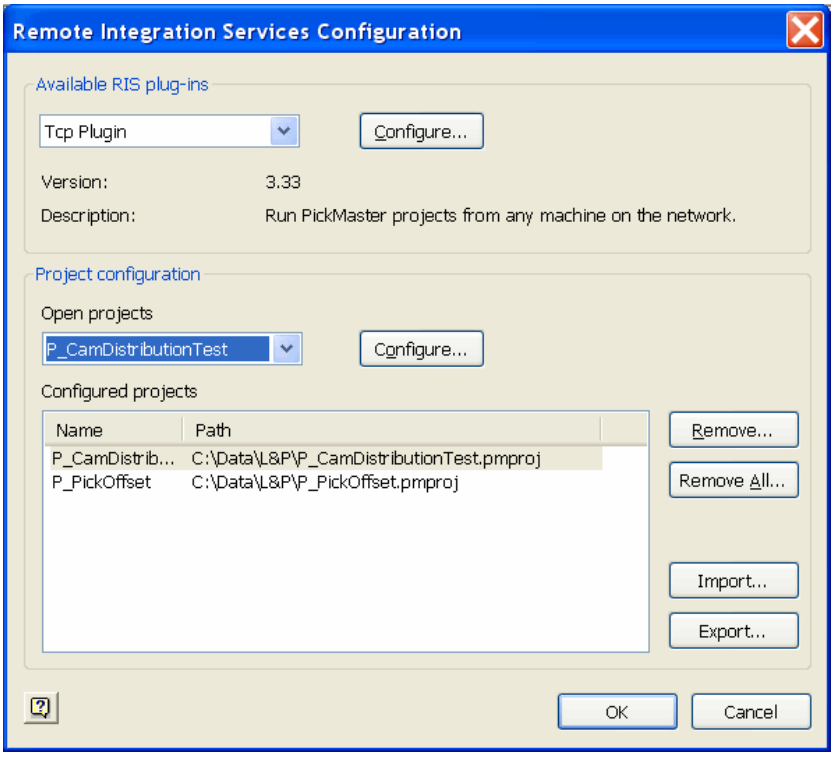
Hilscher CIFX 50E-RE series is supported by PickMaster.

4.4.2 Configuring RIS

Overview

1 Open the project.

2 On the **File** menu, click **Edit RIS plug-in**, the **Remote Integration Service** dialog box appears.



en0900000534

3 In the **Available RIS plug-ins** panel, select the type of plug-in in the list and click **Configure**.

4 Proceed as described for your plug-in:

- *Serial plug-in on page 159*.
- *TCP/IP plug-in on page 161*.
- *RIS2 plug-in on page 162*.
- *Hilscher CifX 50E-RE plug-in on page 155*

5 Continue with configuring the projects. For more information on Serial Plug-in, or TCP Plug-in, see *Configuring the project on page 168*.

4.4.3 Managing roles and users

Introduction

PickMaster manages different roles and users. Each user has a role and each role is associated with an ability. Users, roles, and abilities are all defined in `auth.conf`. This file is located in the PickMaster installation folder under `RIS\RIS2\auth\`.

The roles "administrator" and "operator" are provided. The administrator has the ability to “manage” and the operator has the ability to “operate”.

A default user and password have been created for each role.

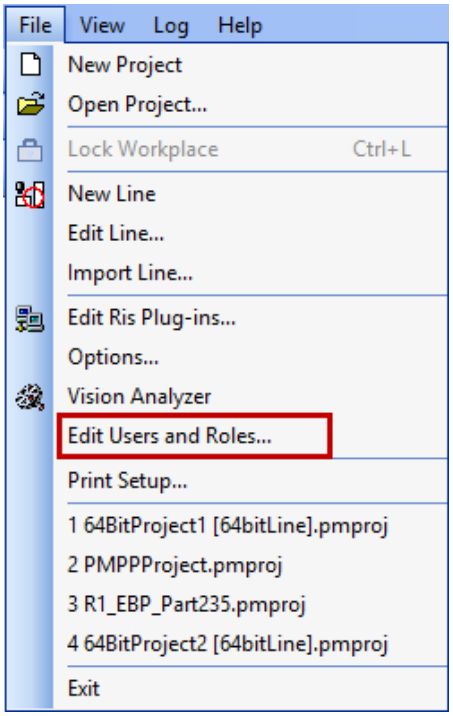
Administrator Username: `admin` with Password: `password`

Operator Username: `operator` with Password: `password`

> **Note**
>
> The Username and Password are case sensitive.

Adding a new user to a group is done in two steps, that is, by creating a user and then assigning it to a role.

You can create or delete roles and users using the **Edit Users and Roles...** option from the **File** menu.



xx1300001102

Creating a role

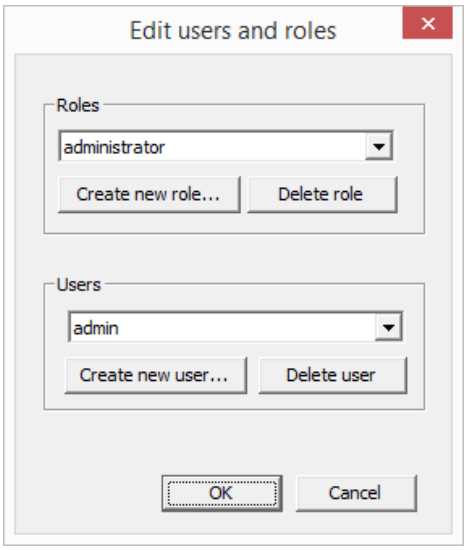
To create a role:

1 Click **File** and select **Edit Users and Roles...**.

Continues on next page

4.4.3 Managing roles and users
Continued

The **Edit users and roles** window is displayed.



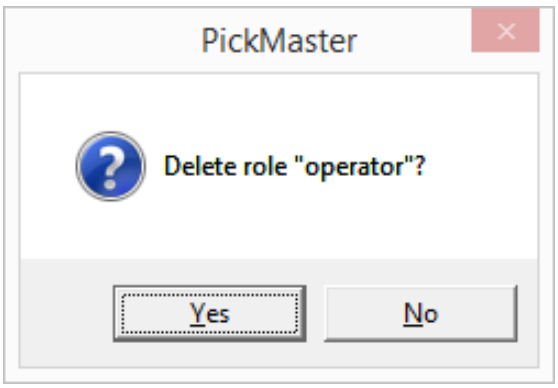
xx1300001077

2 In the **Roles** section, click **Create new role...**.
The **Create new role** window is displayed.
3 Type a name for the new role in the **Roles** field.
4 Type the names of the abilities to be associated with the role in the **Ability** field. Separate the abilities with a blank space.
5 Click **OK**.
The role is created and added to the Roles list.

Deleting a role

To delete a role:

1 Click **File** and select **Edit Users and Roles...**.
The **Edit users and roles** window is displayed.
2 Select the role that you want delete from the **Roles** list.
3 Cick **Delete role**.
A confirmation window to delete the selected role is displayed.



xx1300001078

Continues on next page

4 Click **Yes**.

The selected role is deleted and removed from the **Roles** list.

WARNING

When you delete a role the users in the selected role are also deleted.

Creating a user

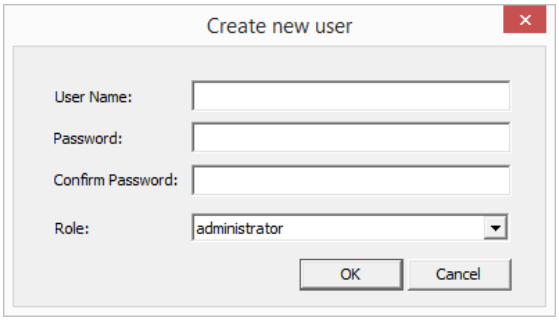
To create a user:

1 Click **File** and select **Edit Users and Roles...**.

The **Edit users and roles** window is displayed.

2 In the **Users** section, click **Create new user**.

The **Create new user** window is displayed.



xx1300001079

3 Type a name for the new user in the **User Name** field.

4 Type a password in the **Password** field.

5 Retype the password in the **Confirm Password** field.

6 Select the role to assign to the user in the **Role** list.

7 Click **OK**.

The user is created and added to the user list with the selected role assigned.

Deleting a user

To delete a user:

1 Click **File** and select **Edit Users and Roles...**.

The **Edit users and roles** window is displayed.

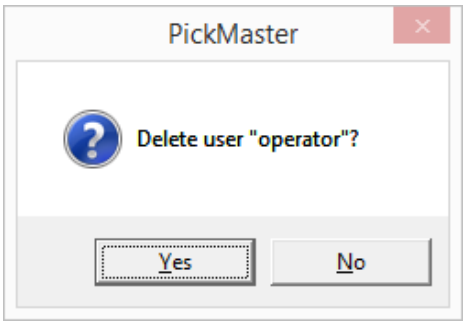
2 Select the user that you want to delete from the **Users** list.

3 Cick **Delete user**.

Continues on next page

4.4.3 Managing roles and users
Continued

A confirmation window to delete the selected user is displayed.



xx1300001080

4 Click **Yes**.

The selected user is deleted and removed from the **Users** list.

4.4.4 Commands and responses

Commands

The RIS protocol specifies project and robot commands. The PickMaster RIS plug-in will acknowledge the command when the state of the project or robot changes. If something is preventing PickMaster to successfully execute the command, PickMaster will inform the RIS client of the cause.

Command	Project response	Robot response
Open project	Open, Error*	n/a
Close project	Closed, Modified, Running, Error*	n/a
Start project	Running, Closed, Configuration error*, Invalid license*, Error*	Running, Shutdown the main computer, Emergency stop state, Error., Manual mode
Stop project	Stopped, Closed, Error*	Stopped, Shut down, Error
Get project status	Open, Closed, Running, Stopped	n/a
Start robot	Stopped, Closed	Running, Shutdown the main computer, Emergency stop state, Error., Manual mode
Pause robot	Stopped, Closed	Stopped, Shutdown the main computer, Paused, Emergency stop state, Error*
Get robot status	Stopped, Closed	Stopped, Running, Shutdown the main computer, Paused, Emergency stop state
Get pick rate	Stopped, Closed	Picks per minute
Reset emergency stop state	Stopped, Closed	Stopped, Running, Shutdown the main computer, Paused, Error*

*) See log view or the *Event Viewer* for more information about the error.

Status responses

The different states are described in *Robot states on page 194*. The *Shutdown* state is specific for *Remote Integration Service*. When a project is stopped it can take a few seconds until the robot is actually stopped. During this time the robot status is the *Shutdown* state.

The project and robot status responses are sent as soon as the corresponding state is changed but the robot pick rate is only sent as a response to a *Get pick rate* command.

Response values

Response	Project values	Robot values
Running	0x01	0x01
Stopped	0x02	0x02
Paused		0x03

Continues on next page

4.4.4 Commands and responses
Continued

Response	Project values	Robot values
Configuration error	0x03	
Shutdown		0x04
No license	0x04	
Emergency stopped		0x05
Error	0x05	0x06
Open	0x06	
Manual mode		0x07
Closed	0x07	
Modified	0x08	

4.4.5 Exporting and importing the RIS configuration

Introduction

For a TCP/IP plugin, the RIS configuration may be exported to an xml file. The xml file may be directly modified in a text editor and then imported back again. Any XML editor (for example, Microsoft Notepad) can be used to edit.

All command values can be modified and new projects can be added.

The import function adds or replaces projects to the current configuration.

The import of an XML file succeeds if:

- The XML syntax is correct, that is as generated by the export function.
- No duplicated projects are defined in the xml file.
- All command values are unique.

Prerequisites

Imported projects can be run if:

1. All project paths are correct, that is the location of the **.pmproj* files are correct.
2. The IDs for the robots of the project are correct, that is, the robot IDs are defined in the **.pmline* file on which the **.pmproj* file is based on.

Example of XML Format

The following example shows a .xml file that can be imported and it contains the following informations:

- Plugin used (TCP in this case.)
- List of projects with its paths.
- Commands to Open, Close, Start, and Stop project
- Details of the Robot - Robot ID, Commands to Open, Close, Start , Stop, Status, PickRate, and ResetEStop the Robot.

```
<?xml version="1.0" encoding="utf-8"?>
  <RisPlugin Type="Tcp">
    <Projects>
      <Project Path="C:\PM projects\ABBchip2
           [PickMasterLab].pmproj">
        <Commands Open="150" Close="51" Start="52" Stop="53"
             Status="54" />
        <Robots>
          <Robot Id="7aa0a489-fbc9-4062-98ae-c1ad6a16f5ae">
          <Commands Start="55" Pause="56" Stop="57" Status="58"
               PickRate="59" ResetEStop="60" />
          </Robot>
          <Robot Id="2b8732f6-8c7f-46fa-b9e0-1c1f94958ed7">
          <Commands Start="61" Pause="62" Stop="63" Status="64"
               PickRate="65" ResetEStop="66" />
          </Robot>
        </Robots>
      </Project>
```

Continues on next page

4.4.5 Exporting and importing the RIS configuration
Continued

```
    <Project Path="C:\PM projects\ABBchip1
         [PickMasterLab].pmproj">
    <Commands Open="67" Close="68" Start="69" Stop="70"
         Status="71" />
      <Robots>
      <Robot Id="7aa0a489-fbc9-4062-98ae-c1ad6a16f5ae">
      <Commands Start="72" Pause="73" Stop="74" Status="75"
           PickRate="76" ResetEStop="77" />
      </Robot> <Robot Id="2b8732f6-8c7f-46fa-b9e0-1c1f94958ed7">
      <Commands Start="78" Pause="79" Stop="80" Status="81"
           PickRate="82" ResetEStop="83" />
      </Robot>
      </Robots>
    </Project>
    <Project Path="C:\PM projects\ABBchip3
         [PickMasterLab].pmproj">
    <Commands Open="84" Close="85" Start="100" Stop="86"
         Status="87" />
    <Robots>
    <Robot Id="7aa0a489-fbc9-4062-98ae-c1ad6a16f5ae">
    <Commands Start="88" Pause="89" Stop="90" Status="91"
         PickRate="92" ResetEStop="93" />
    </Robot>
    <Robot Id="2b8732f6-8c7f-46fa-b9e0-1c1f94958ed7"> <Commands
         Start="94" Pause="95" Stop="96" Status="97"
         PickRate="98" ResetEStop="99" />
    </Robot>
    </Robots>
    </Project>
  </Projects>
</RisPlugin>
```

4.4.6 Hilscher CifX 50E-RE plug-in

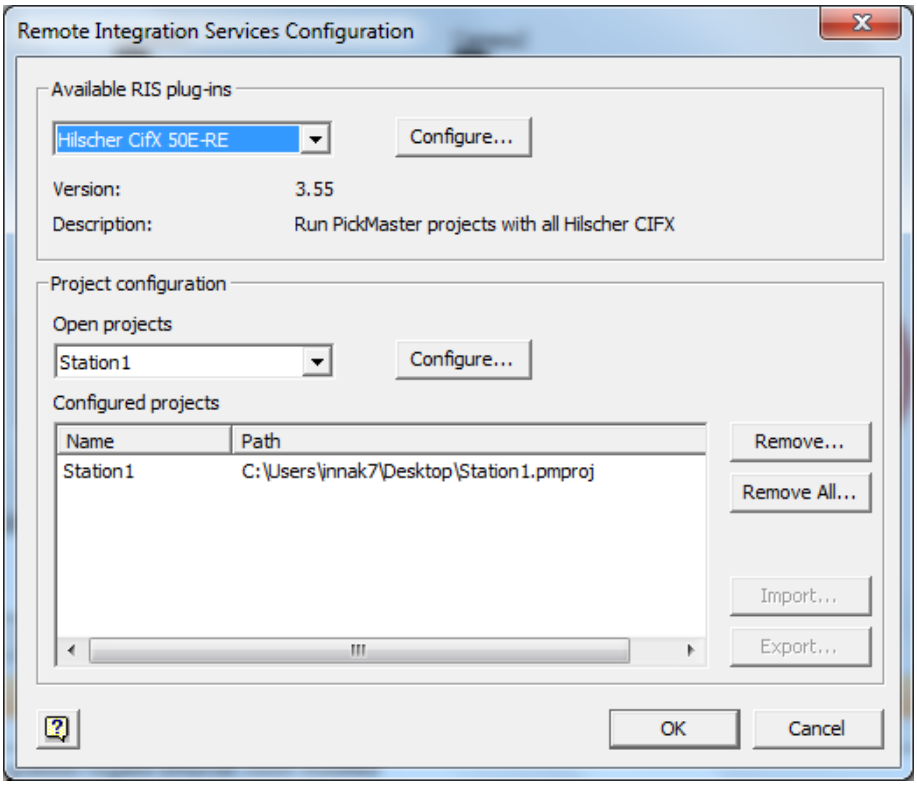
Configuring the Hilscher CifX 50E-RE plug-in

To configure the Hilscher CifX 50E-RE plug-in:

1. In the **File** menu, click **Edit RIS Plug-ins**.

   The **Remote Integration Services Configuration** window is displayed. For more information about this window, see *Configuring RIS on page 146*.

2. Click the **Available RIS plug-ins** drop-down list, and select **Hilscher CifX 50E-RE**.



xx1800001451

3. Click the **Configure** button.

   The **Device Information** window is displayed.

4. Click the **Device Open** button.

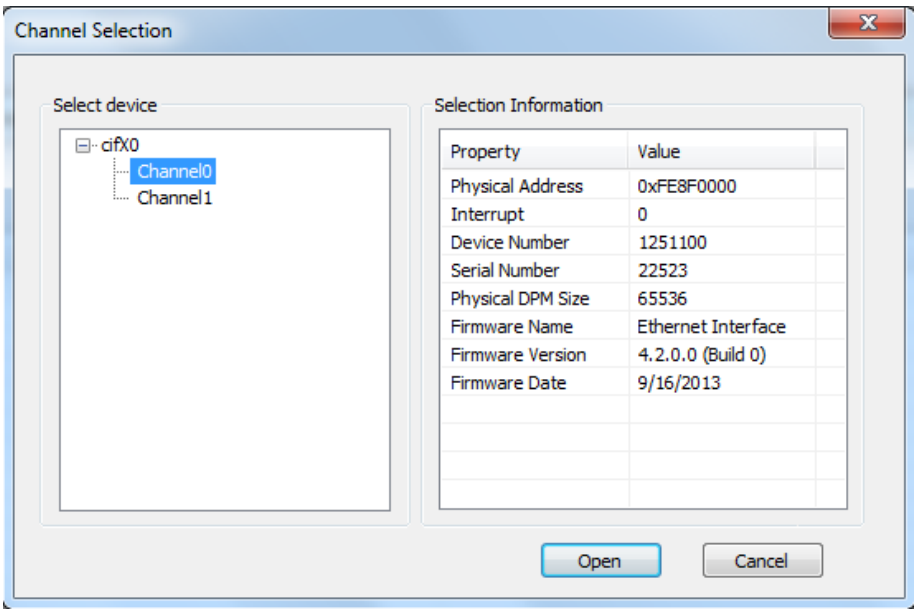
   The **Channel Selection** window is displayed.

5. In the **Select device** section, select **cifX0** > **Channel0**.

Continues on next page

4.4.6 Hilscher CifX 50E-RE plug-in
Continued

The **Selection Information** section displays the information about the selected channel.



xx1800001452

6 Click **Open**.

The **CifX0 Channel0** window is displayed.

> **Note**
>
> To view status about the board, version, driver, or firmware and channel information, click the buttons **GetDriverInformation** and **GetChannelInformation** in the channel window.

7 Click **Close**.

8 In the **Remote Integration Services Configuration** window, in the **Project Configuration** section, select the required project from the **Open projects** drop down.

9 Click **Configure**.

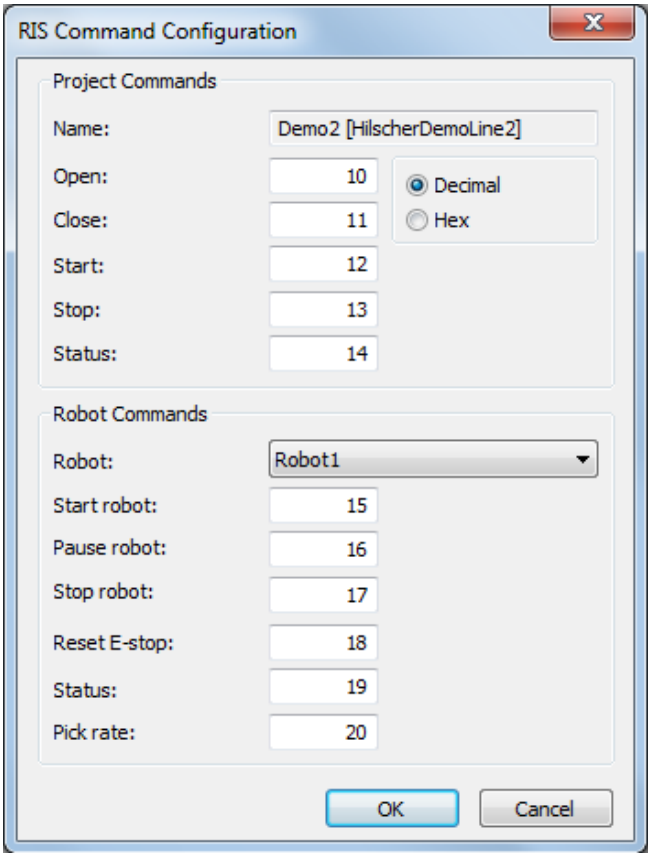
The **RIS Command Configuration** window is displayed.

10 Enter the RIS command values in the **Project Commands** and **Robot Commands** section.

> **Note**
>
> All values must be in whole numbers in the range 1-65535 and the values must be unique.

> **Note**
>
> The configured robots in the project are listed in the **Robot** list. For each selected robot, unique whole number values must be used.

Continues on next page



xx1800001454

11 Click **OK**.

The project is configured and is displayed in the **Configured projects** section.

12 Click **OK**.

The Hilscher EtherNetIP plug-in is configured.

Tip

When programming the master board we recommend clearing the written command after (2 x polling time) milliseconds. For example, the *Remote Integration Service* client sends a command for starting the project. PickMaster responds that a controller is in manual mode. After setting the controller to auto mode the same command (*Start project*), that already was written in the slave's DPM, must be written again. The only way to know whether the user wants to try to start the project again is to clear the command a certain time after each write to the slave's DPM.

Supported hardware

The Hilscher CifX 50E-RE series should be used with PickMaster.

Continues on next page

4.4.6 Hilscher CifX 50E-RE plug-in
Continued

Same Hardware and Interface can be used for all Ethernet protocols (PowerLink, EthernetIP, ProfiNet, Sercos, EtherCAT, ModBus and Varan).

Note

Testing is done only for EthernetIP and ProfiNet protocols.

Communication protocol

The communication over the field bus uses 4 bytes in the protocol.

The examples below illustrate sending and receiving the robot status command (193).

Example: Commands to PickMaster

The command value to PickMaster can be:

- Start project
- Stop project
- Ask project status
- Start robot
- Stop robot
- Reset emergency stop
- Ask Pick rate on robot
- Ask robot status

The fieldbus command value should be sent in *Byte0* and *Byte1*.

Byte3	Byte2	Byte1	Byte0
0	0	0	193

Example: Commands from PickMaster

The commands from PickMaster can be:

- Project status
- Robot status
- Pick rate

The fieldbus command value is sent in *Byte2* and *Byte3* and the status value or pick rate in *Byte0* and *Byte1*.

Byte3	Byte2	Byte1	Byte0
0	193	0	2 (stopped)

Related information

For more information about Hilscher products contact your local Hilscher distributor, see *http://www.hilscher.com*

Details about Hilscher CifX 50E-RE is available here *https://www.hilscher.com/products/product-groups/pc-cards/pci-express/cifx-50e-res3m/*

For troubleshooting the hardware contact your local Hilscher support or Hilscher worldwide support telephone.

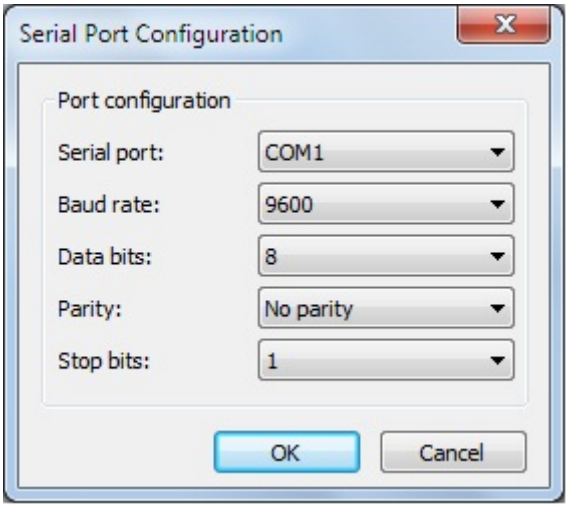
4.4.7 Serial plug-in

Configuring the serial plug-in

To configure the serial plug-in:

> **Note**
>
> It is important to select the same settings for the plug-in as are used by the connected device.

1. In the **File** menu, click **Edit RIS Plug-ins**. The **Remote Integration Services Configuration** dialog appears. For more information on this dialog box, see *Configuring RIS on page 146*.
2. Click the **Available RIS plug-ins** drop-down list, and select **Serial Plugin**.
3. Click **Configure** button. The **Serial Port Configuration** dialog appears.



en1200000672

4. Select the COM port to be used by PickMaster from the **Serial port** list.
5. Select the data speed from the **Baud rate** list.
6. Select the number of data bits from the **Data bits** list.
7. Select the parity scheme from the **Parity** list.
8. Select the number of stop bits from the **Stop bits** list.
9. Click **OK**and then continue configuring the projects, see *Configuring the project on page 168*.

> **Note**
>
> If PickMaster fails to open the selected COM port or if invalid settings are selected, the error will not be shown as a message box. Instead the error will be shown in the log view.
>
> No type of flow control is supported.

Continues on next page

4.4.7 Serial plug-in
Continued

Supported hardware

The serial plug-in is used to access PickMaster over an RS-232 connection. The plug-in can be configured to use any COM port with a wide range of possible configurations.

Communication protocol

When the plug-in is running it is listening for incoming commands. These commands must be 2 bytes in length. If a single byte is read it will be discarded and if 3 bytes are read the last byte will be skipped. The available commands are listed in *Commands and responses on page 151*.

The responses are sent in 4 bytes, with the response type in the first 2 bytes and the status type in the next 2 bytes.

The bytes are sent in the network byte order, big endian. This means that the most significant byte is sent first. The command 256 (0x0100) must be sent as 0x01, 0x00. A response of command 10 (0x000A) with argument 5 (0x0005) is sent as 0x00, 0x0A, 0x00, 0x05.

Example

Assume that the command value for *Get robot status* is 102 (0x0066). To query PickMaster for a robot status, the command must be given as shown below.

Byte1	Byte0
102 (0x66)	0 (0x00)

If the robot is paused, the response will look like this.

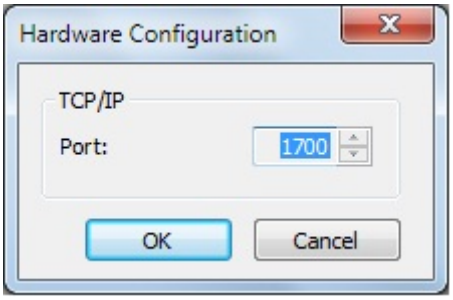
Byte3	Byte2	Byte1	Byte0
3 (0x03)	0 (0x00)	102 (0x66)	0 (0x00)

4.4.8 TCP/IP plug-in

Configuring the TCP/IP plug-in

To configure the TCP/IP plug-in.

1. In the **File** menu, click **Edit RIS Plug-ins**. The **Remote Integration Services Configuration** dialog appears. For more information on this dialog box, see *Configuring RIS on page 146*.
2. Click the **Available RIS plug-ins** drop-down list, and select **TCP Plugin**.
3. Click **Configure** button. The **Hardware Configuration** dialog appears.



en1200000673

4. If required, change the port number.

   The port number does normally not need to be changed.
5. Click **OK** and then continue configuring the projects, see *Configuring the project on page 168*.

Supported hardware

The TCP/IP plug-in is used to access PickMaster over any TCP/IP network.

Communication protocol

The communication protocol used with the TCP/IP plug-in is the same as used with the serial plug-in, see *Serial plug-in on page 159*.

4.4.9 RIS2 plug-in

Introduction

RIS2 (Remote Integration Service 2) has the following features:

- A platform independent communication interface
- Hot tuning of parameters
- Information access
- Service operations
- Project management and
- User management

RIS2 provides access for HMIs and other devices that can communicate through HTTPS.

Available operations

Project management operations

- List project
- Open project
- Close project
- Save project
- Start project
- Stop project
- Get project status
- Get log messages

Robot management Operations

- Start robot
- Pause robot
- Stop robot
- Get robot status
- Reset robot emergency stop

Service variables and routines

- List service variables
- Read service variable
- Write service variable
- List service routines
- Invoke service routine

Vision monitoring

- List position sources
- Launch detailed vision on host PC

Tuning parameters

- List tuning parameters

Continues on next page

- Read tuning parameter
- Write tuning parameter

Configuring the RIS2 plug-in

To configure the RIS2 plug-in:

1. Start the configuration editor. For more information, see *Configuring RIS on page 146*.
2. In the Project Configuration, click **Configure** to add the project to configured projects list.
3. Click OK.

Continue by configuring the web server. For more informations, see the following section Server Configurations.

Tip

When using RIS2, do not use the "&" character in the project path as it might cause the request parameters to the RIS2 web server to be incorrect.

Server Configurations

RIS2 requires the web server embedded in PickMaster. The configuration file `appweb.conf`, located in the PickMaster folder, manages the server's configuration. It is read when the server starts up, that is, when the RIS2 plug-in is started for the first time. Configuration instructions are one per line and are not case-sensitive for the instruction names. Lines beginning with a "#" character are comments and are ignored. For RIS2 the following instructions are mandatory, marked in italics are variables subject to change depending on installation:

```
include RIS\RIS2\log.conf
include RIS\RIS2\auth\auth.conf
SSLCertificateFile "${BIN_DIR}/PickMaster3.crt"
SSLCertificateKeyFile "${BIN_DIR}/PickMaster3.key"
ListenSecure 50000

#Sets the maximum number of simultaneous client sessions to
     unlimited
LimitSessions unlimited

#PutMethod on
Methods set GET POST

#RIS2
LoadModule RESTHandler RIS\RIS2Pluginu
LoadModule WSHandler RIS\RIS2Pluginu

AddHandler fileHandler .html .gif .jpeg .png .pdf ""
```

Continues on next page

4.4.9 RIS2 plug-in
Continued

For more information, see comments in appweb.conf or visit the visit the AppWeb website at http://appwebserver.org. If the web server is running when configuration changes are made, PickMaster needs to be restarted for the changes to take effect.

Note

appweb.conf is write protected after installation. Write protection must be removed before edits.

By default RIS2 server listens to secure ssl (https) connection on port 50000. This ports may be customized per customer need.

The communication is encrypted by the specified certificate and key files stated in the `appweb.conf` file. The default certificate is a temporary self-signed certificate used to make the communication started. It is recommended to exchange this according to each company's IS/IT-policy.

Every RIS2-client implementation requires to follow the same policy.

SSL Configuration

By default is the RIS2 server listening for secure SSL (https) connection on port 50000. This ports may be customized per customer need.

The communication is encrypted by the specified certificate and key files stated in the appweb.conf file. The default certificate is a temporary self-signed certificate used to make the communication started. It is recommended to exchange this according to each company's IS/IT-policy.

Every RIS2-client implementation requires to follow the same policy.

Log Files

Introduction

The embedded web server provides detailed logging for client accesses and operations. This is done by two log files:

- Error log
- Trace log

The logs are located in the PickMaster installation folder under *RIS\RIS2\logs*.

Error log

The error log records the server configuration, details of denied requests, and other trace and error information.

The verbosity level of the messages written can be defined in the ErrorLog directive of log.conf (by default located in the PickMaster installation folder under RIS/RIS2). The level is a digit ranging from 0 to 5, with 0 being the least verbose level and 5 being the most verbose level.

Trace log

The access log records the details of each successful request served by the embedded web server. By default, it will log in the Combined Log File Format that is used by the Apache web server.

Continues on next page

RIS2 user management

RIS2 uses the same users, roles, and abilities as PickMaster. For more information, see *Managing roles and users on page 147*.

Each RIS2 operation has its own route block in the configuration file, *appweb.conf*.

Example (listprojects operation):

```
<Route "/pickmaster/listprojects>
SetHandler RESTHandler
AuthRealm "PickMaster"
AuthDigestQop auth
AuthType Digest
Require ability manage
</Route>
```

The first three instructions are mandatory and should not be changed.

The `AuthRealm` instruction names a discrete region of access. Any number of realms may be created but only one per context is allowed.

The require ability instruction specifies which ability is required to have access to the RIS2 operation. In the above example only users with a role that has the manage ability may use the operation “listprojects”.

In the default configuration only users with a role that has the manage ability can use RIS2 operations.

RIS2 requests

The web server embedded in PickMaster provides a REST interface for accessing RIS2 operations.

Each request is an URL starting with a prefix, for example *https://pickmasterhost.se.abb.com/ pickmaster/*.

The request type is identified by a string followed by a slash, for example: *startrobot/*, parameters are given by a question mark followed by the parameters given in the classical URL syntax, for example: *projectpath=c:\projectpath&robotid=37e8578a-…*

The result of the requests is provided in XML.

RIS2 client

PickMaster is distributed with a RIS2 client. The client is available as open source. It can be used as it is or as an example while designing clients for PickMaster using RIS2 interface.

The client is developed in Visual Studio and the solution consists of two projects, *RIS2Client* and *RIS2ClientHelper*.

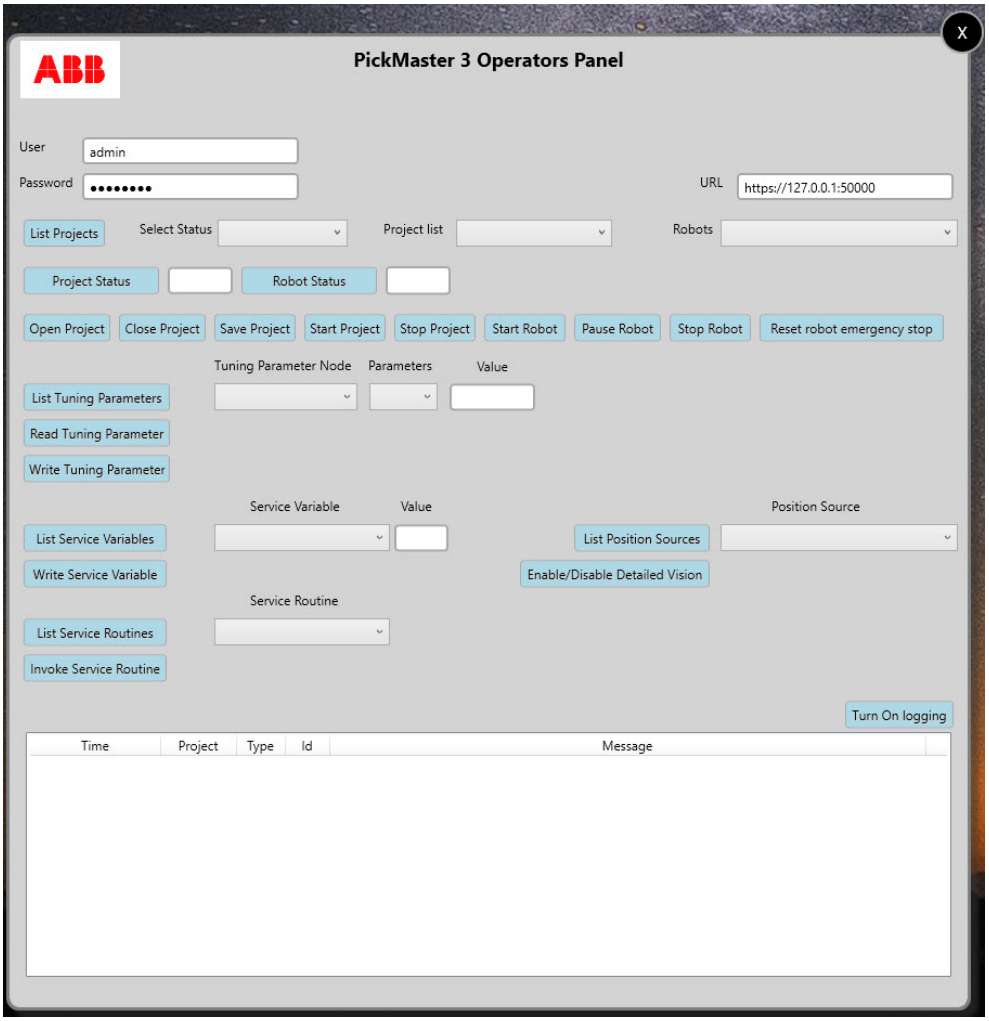
The *RIS2Client* project is implemented using Windows Presentation Foundation (WPF). It takes advantage of the entire RIS2 interface.

Continues on next page

4.4.9 RIS2 plug-in

Continued

Look at the source code, build the project, start it and get a feel of how a PickMaster client can be designed and used.



xx2200001463

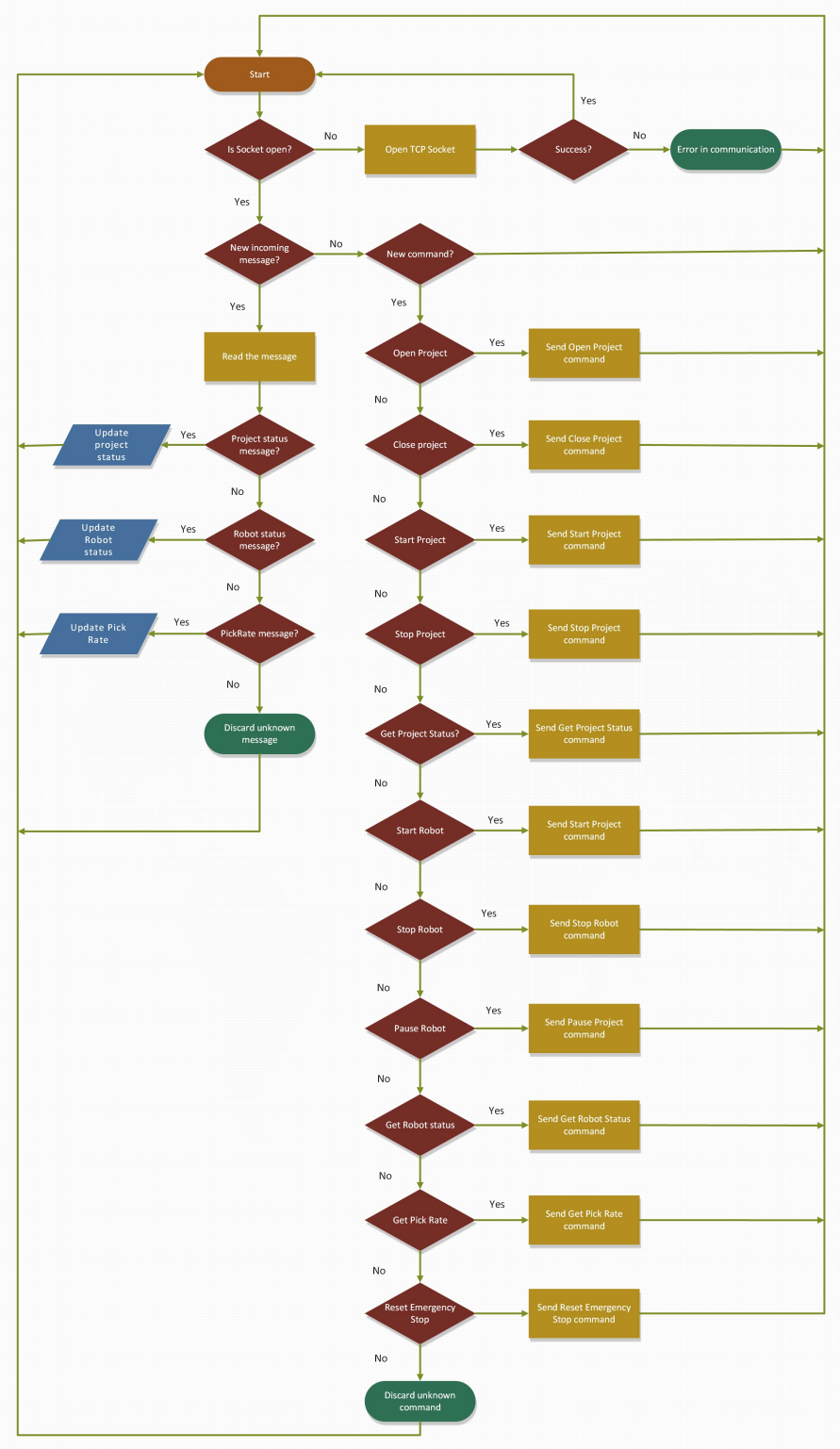
The RIS2ClientHelper is a class library with methods and classes that helps while implementing a PickMaster client. The methods can access the entire RIS2 interface and parse the responses.

The RIS2ClientHelper can be used as is and imported into new solutions or adjusted to fit different user needs. See source code for further documentation.

4.4.10 RIS request flowchart

Flowchart

The following flowchart provides an overview of RIS requests:



xx2000000521

4.4.11 Configuring the project

Introduction

Use this procedure to define how PickMaster should interpret the fieldbus commands. Follow this procedure for Hilscher, serial, and TCP/IP plug-ins.

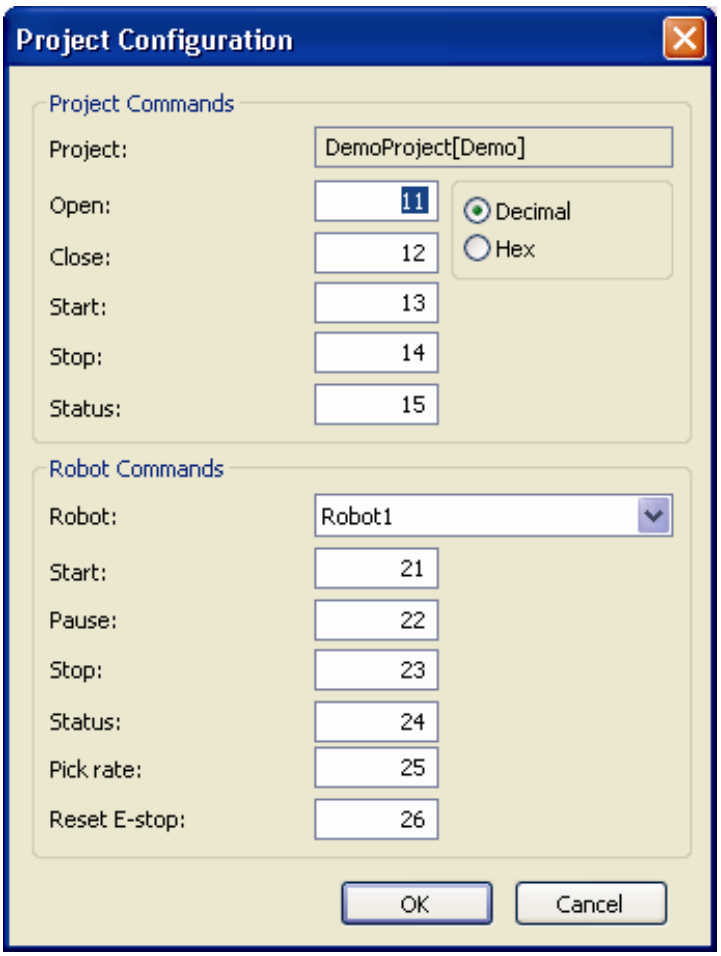
All commands must be defined as integers in the range 1-65535 and they must be unique. The commands are defined with default numbers that usually do not need to be changed.

1. In the **Project Configuration**, click **Configure**.
2. Define the project commands.
3. Continue with configuring the robot commands.
4. Select robot in the **Robot** list.
5. Define the robot commands.

   If there is more than one robot in the project you can define commands for all robots before clicking OK.
6. Click **OK** to close the **Project Configuration**.
7. Click **OK** to close the **Remote Integration Service Configuration**

Continues on next page

Illustration Project Configuration



en0900000535

4.5 Customizing PickMaster with a User Hook

4.5.1 Introduction to User Hook

Introduction to User Hook

A PickMaster User Hook is a third party software component that can be designed to customize item positions during runtime. A User Hook can for example be queried for positions instead of using predefined positions. It is also possible for Hook objects to adjust item positions generated by vision models in PickMaster.

Item positions carry some free usage parameters that can be set by the user hook. These parameters can later on be accessed in RAPID by the robot that handles the position.

A User Hook is a .NET object that implements specific interfaces. Different types of Hook objects use different interfaces. Since a User Hook is a .NET object it can be implemented in either C#, VB.NET, C++, or any other .NET compatible language. The interfaces that a User Hook has to implement are defined in the *PMHook.dll* assembly under the *ABB.PickMaster.UserHooks* namespace.

Most of the User Hook objects control item positions in one way or another. An item position is defined with the ItemPosition class.

Prerequisites

The interface that a Hook must implement contains a single method. But the interface is derived from the *IPMUserHook* interface, that contains some properties and functions that also must be implemented. Furthermore, every User Hook must also inherit from the *HookMarshaller* class because the Hook objects have to be marshaled between different application domains.

A User Hook can generate its item positions in several ways. It might be sufficient with an algorithm in the Hook itself but most probably it is required to query an external object of some kind. The Hook can read from file, communicate with another .NET object using remoting, connecting sockets over Ethernet, or communicate with a PLC.

Supported User Hooks

PickMaster supports two types of User Hooks.

User Hook	Description
PositionAdjuster	The PositionAdjuster is a User Hook that implements the IPositionAdjuster interface. A PositionAdjuster can be used to adjust the positions generated by a vision model. Each time the model generates positions, an array with the positions is sent to the User Hook object. The Hook object can then control the positions in any desired way. Positions can be changed, removed, or added.
PositionGenerator	The PositionGenerator is a User Hook that implements the IPositionGenerator interface. A PositionGenerator object can be used as a flexible predefined PositionSource. Each time predefined positions should be generated, PickMaster queries the PositionGeneratorHook object for positions. Any number of positions can be returned from the PositionGenerator.

Continues on next page

How User Hooks are used by PickMaster

User Hooks are compiled into dynamic link library files (dll) and PickMaster looks for these files in the folder *UserHooks* in the PickMaster folder. Therefore, all User Hooks must be copied into this folder before they can be used from PickMaster.

Once a User Hook has been configured from PickMaster, the file that contains the Hook object will be locked. Even if the PickMaster project is closed the file will still be marked as used by the system. This is by design and the PickMaster application must be closed before a User Hook file can be deleted or exchanged with another file.

Files provided in the installation

The following files are provided in the folder *SDK* on the "PickMaster3 Installation Package".

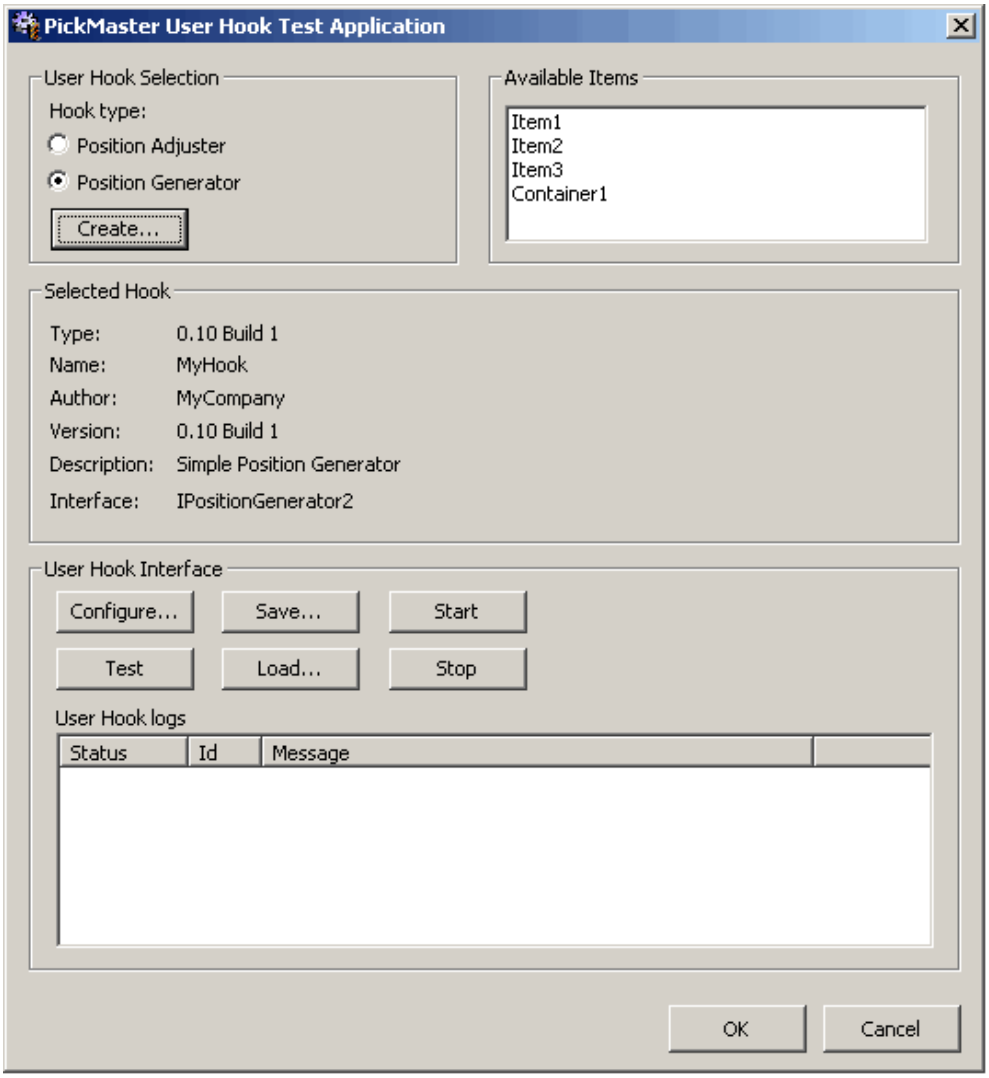
- *PMHook.dll* contains the definitions of the .NET objects.
- *PMHook.xml* includes help texts to be shown from the Object Browser in Visual Studio.
- *SimplePosGenHook.cs* is an implementation of a simple PositionGenerator.
- *HookTester.exe* is a test application for User Hooks.

Continues on next page

4.5.1 Introduction to User Hook
Continued

Test application

There is a test application called *HookTester* (*HookTester.exe*) provided with the "PickMaster3 Installation Package". This program can be used during development to test the User Hook. *HookTester* looks for available User Hook fils in a subfolder called *UserHooks* in the same way as PickMaster. Most of the interface functions can be tested from the test program.



en1000001396

Related information

ABB.PickMaster.UserHooks namespace on page 176.

All classes, interfaces, structures, and enumerations are described in the file *PickMaster User Hooks.chm*.

4.5.2 Customizing PickMaster with a User Hook

Introduction to User Hook customizations

There are two ways to customize PickMaster with a User Hook. Either create a new User Hook, or implement an existing User Hook.

Prerequisites

The following is required.

- Visual Studio 2005.
- Basic knowledge of .NET.
- *PMHook.dll* and *SimplePosGenHook.cs* from the "PickMaster3 Installation Package".
- Optional: *HookTester.exe* and *PMHook.xml* from the "PickMaster3 Installation Package".

Creating a new User Hook

Use this procedure to create a new User Hook.

1 Add the namespace ABB.PickMaster.UserHooks to the project

```
using ABB.PickMaster.UserHooks;
```

2 Create a new class for your User Hook.

```
public class MyUserHook : HookMarshaller, IPositionAdjuster
```

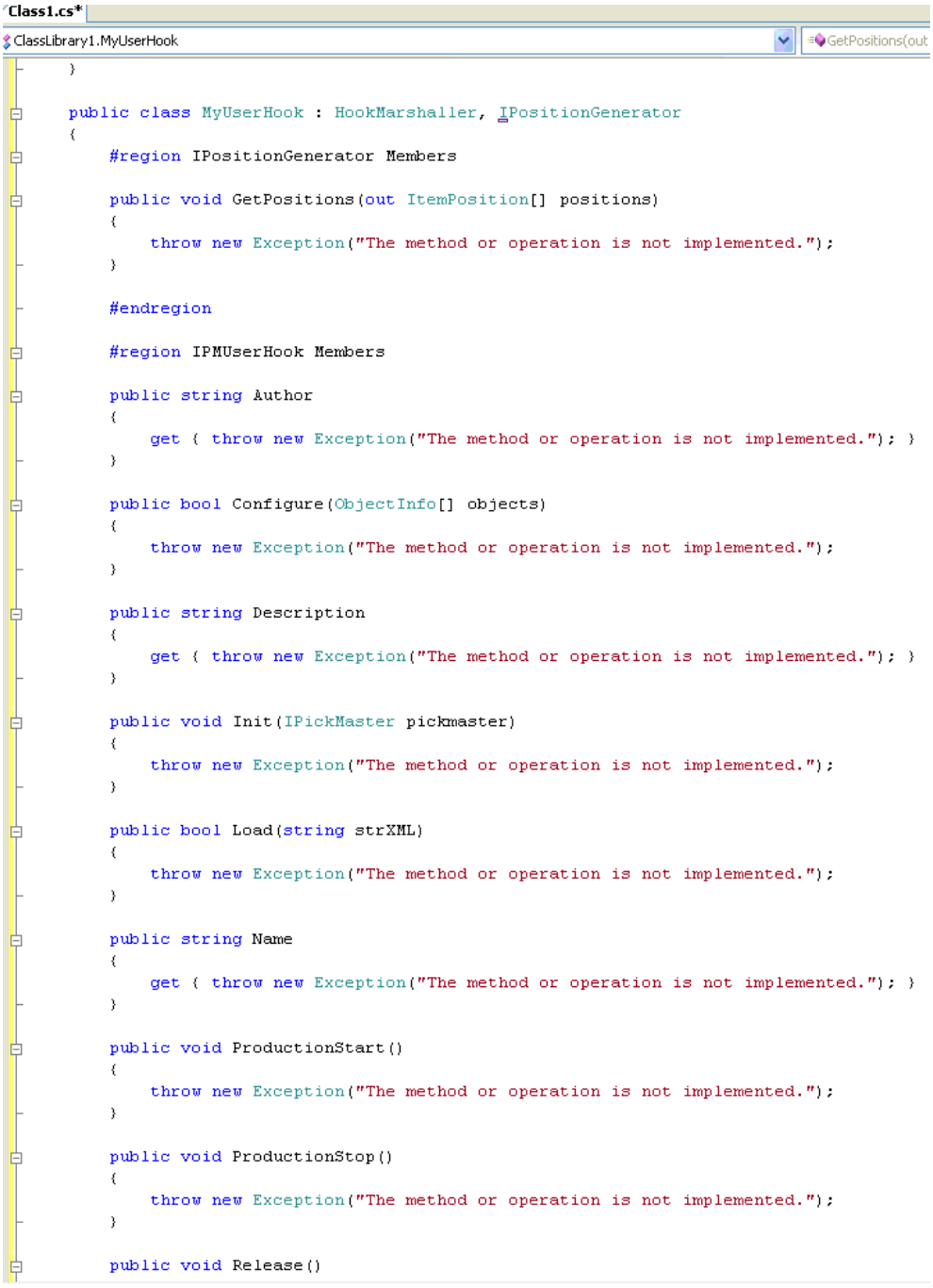
or

```
public class MyUserHook : HookMarshaller, IPositionGenerator
```

Continues on next page

4.5.2 Customizing PickMaster with a User Hook
Continued

3 Right-click on the interface definition and select **Implement the interface**.
The interface is automatically implemented.



en0900000363

Implementing a PickMaster User Hook in C#

Use this procedure to customize a PickMaster User Hook in C#.

1 Create a new C# class library project and name it *MyHook*.
2 Copy the file *PMHook.dll* to the project directory.

Continues on next page

PMHook.dll contains all the definitions of the User Hook interfaces and classes.

3 If needed, copy *PMHook.xml* to the project directory to provide the Object Browser in Visual Studio with help texts.

4 Copy SimplePosGenHook.cs to the project directory

SimplePosGenHook.cs includes a class called *MyHook* that inherits from *IPositionGenerator2* and implements a position generator User Hook. *MyHook* only generates a hardcoded position at each trigger but it shows how little code that is needed to implement a complete User Hook.

5 On the **Project** menu, select **Add Reference**. In the .NET tab, select **System.Windows.Forms** to add the assembly to the project.

6 On the **Project** menu, select **Add Reference**. On the **Browse** tab, locate *PMHook.dll* to add the PMHook assembly to the project.

7 On the **Project** menu, select **Add Reference**. On the **Browse** tab, locate *SimplePosGenHook.cs* to add a simple position generator to the project.

8 Click **OK**.

The User Hook*MyHook.dll* is created.

9 If needed, test the Hook with *HookTester*.

10 Copy *MyHook.dll* to the folder UserHooks.

4.5.3 ABB.PickMaster.UserHooks namespace

Overview

The *ABB.PickMaster.UserHooks* namespace provides definitions for all User Hook interfaces that can be used by PickMaster. They are described in detail in the file *PickMaster User Hooks.chm*.

Classes

Class	Description
AdjustablePosition2	Represents a position in a specific work area for a robot controller. This class is used by a *PositionAdjuster*User Hook. This class cannot be inherited.
HookMarshaller	Base class for all User Hook that enables access to the objects across application domain boundaries. This class is abstract (MustInherit in Visual Basic) and so cannot be instantiated.
ItemPosition2	Represents a position in a specific work area for a robot controller. This class is used by a *PositionGenerator user hook*.
LogMessage	Represents a log message that can be sent to PickMaster and added to the application log. This class cannot be inherited.

Interfaces

Interface	Description
IPickMaster	A reference to the PickMaster application.
IPMUserHook	*IPMUserHook* is the base interface for all User Hooks. It contains basic properties about the Hook and some methods that every Hook must implement.
IPositionAdjuster2	*IPositionAdjuster2* is the interface to implement for User Hooks that is to adjust item positions generated by PickMaster vision models or predefined positions.
IPositionGenerator2	*IPositionGenerator2* is the interface to implement for User Hooks that is to generate item positions.

Structures

Structure	Description
ObjectInfo	An available PickMaster object.

Enumerations

Enumeration	Description
LogType	Log messages added to the PickMaster log with AddLog can be marked as either Status, Warning, or Error.
ObjectType	Identifies the type of a PickMaster object.
PositionCategory	Item positions can be marked with different categories.

4.6 External sensor .NET

4.6.1 Overview of External Sensor .NET

Introduction

PickMaster supports a software development interface that customizes position generation using third party vision/sensor systems. The interface allows position generation from any kind of external sensors, for example, cameras, line scanners, light barriers, bar code readers and so on. As with positions generated by the PickMaster internal vision system, external sensor positions can be distributed in different ways, for example, to multiple robots using load balancing or ATC. Positions can be generated for different items where the items are individually distributed. Positions can be set as accepted or rejected and can be marked with a tag value that can be checked on Rapid level. External sensors can be used with conveyor work areas as well as indexed work areas. Overlap filtering for conveyors can be handled by PickMaster.

An external sensor can be implemented in any .NET compatible language, such as C# or VB.NET. The external sensor is required to implement a set of predefined interfaces. It will typically use an external hardware to generate positions. It can be communicating with the hardware using for example TCP/IP, serial communication or another suitable protocol.

Working of .NET External Sensor

An external sensor is implemented in a .NET class library, for example *MySensor.dll* or *CameraSensor.dll*. It is required to implement a set of interfaces that are defined in the *PMSensor.dll* assembly.

An external sensor library must contain at least one class inheriting from the `ISensorRuntime` interface. Each class that inherits from `ISensorRuntime` represents a specific type of sensor, such as a camera type. An external sensor created in a PickMaster line configuration corresponds to an instance of such a class. An external sensor instance can be compared with a PickMaster camera. It can be selected from a Position Source to be the source of positions generation.

The `ISensorRuntime` interface in turn inherits from the `IPositionGenerator`, `ISensorConfig` and `ISensorInfo` interfaces. All the methods in these interfaces must be implemented. Each sensor class is responsible for keeping a list of Position Generators. A position generator corresponds to a PickMaster vision model and is used to generate positions for a specified item in PickMaster.

Different sensor types can be placed in the same assembly or in different assemblies. When an external sensor is created from PickMaster, the predefined NetWrapper external sensor is selected. The user will then be prompted to select the .NET sensor to use. All classes implementing the ISensorRuntime interface residing in an assembly placed in the installation folder of External sensor .NET (For example: ”*C:\Program Files\ABB Industrial IT\Robotics IT\PickMaster External Sensor .NET”*) will be listed.

Continues on next page

4.6.1 Overview of External Sensor .NET
Continued

External sensor in runtime

During operation, the external sensor:

- Is recorded after a trigger signal is received from the robot controller.
- Must always set the strobe signal to the robot controller exactly when the sensor is recorded. If a camera is used as external sensor the strobe should be set when the image is exposed. For some sensors it might be sufficient to cross connect the trigger with the strobe.
- Must always fetch a timestamp from PickMaster when the sensor is recorded.
- Must always send a position list to PickMaster for every strobe. The position list must include all detected positions for all position generators. If no positions are detected, an empty position list is sent to PickMaster. The recorded time stamp must be supplied with the position list. For a camera, it is recommended that the position list is sent to PickMaster before the next trigger signal arrives. Line scanners may require several strobes to be sent before the positions can be identified and send to PickMaster.

4.6.2 Installing External Sensor

Installing Methods

The installation of a .NET sensor can be done in two ways.

Use the installation of External Sensor .NET

First install the External Sensor .NET software on the PickMaster PC. The software is found on the "PickMaster3 Installation Package". Execute the setup.exe file and follow the installation instructions.

Then copy the customized external sensor assembly, for instance, mySensor.dll, to the installation folder. The example below is based on this version.

Include External Sensor .NET in a customized installation

Include the customized assembly and External Sensor .NET in a customized installation. Add the merge module, NetWrapperModule.msm, to your installation setup. It is found on the "PickMaster3 Installation Package" in the folder External Sensor .NET/Redist. The folder also include PMSensor.dll which needs to be referenced when building the customized assembly.

4.6.3 Implementing and testing a simple External Sensor with C#

Description

Use this description to implement and test an external sensor on a PC without the need of additional hardware.

Requirements

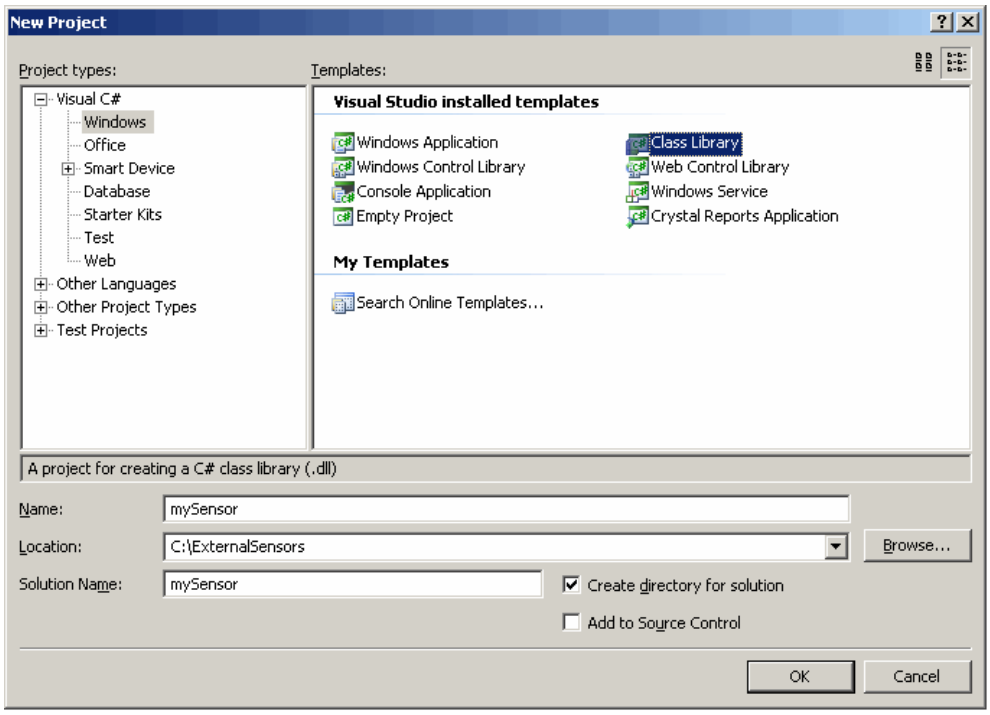
Following is required to run this basic test (using code examples):

- Microsoft Visual Studio 2005
- Microsoft .Net 2.0
- PickMaster 3
- RobotStudio
- Basic knowledge of RobotStudio and PickMaster
- External Sensor .NET
- Knowledge of C# and .Net is required to customize an external sensor, for example, for integration with a real external vision system.

Implementation

Use this procedure to setup a class library

- Launch Microsoft Visual Studio. Create a new C# class library project and give it an appropriate name, for example, MySensor.



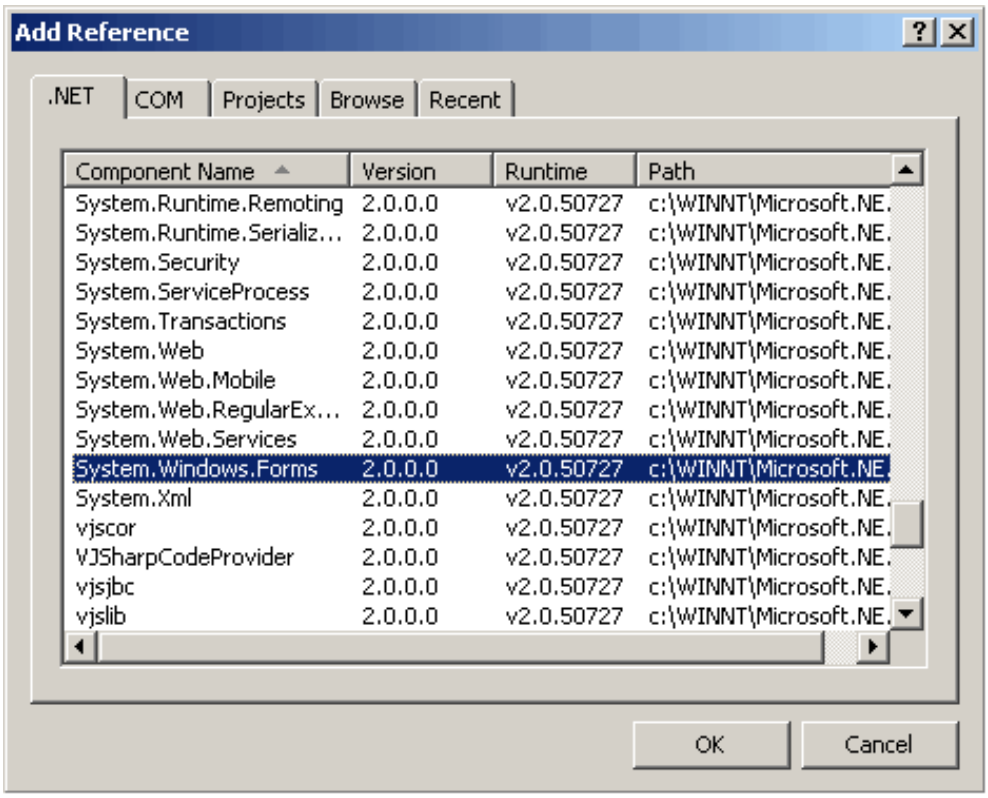
en1000001209

- Copy `PMSensor.dll` from the installation folder of External Sensor .NET (that is, "C:\Program Files\ABB Industrial IT\Robotics IT\PickMaster External Sensor .NET") to the project directory. Optionally copy the file PMSensor.xml

Continues on next page

to the project directory to provide the Object Browser in Visual Studio with help texts.

- Select the **Project menu** and **Add Reference**.



en1000001210

- In the dialog, select **System.Windows.Forms** and **System.Drawing**. Browse for the copied file `PMSensor.dll`. Several other assemblies might be required depending on the implementation of the sensor.

Implementing a Sensor

Example 1: Sensor from code examples

- Delete the template source `Class1.cs`.
- Add existing items: Add the listed *.cs files found on the "PickMaster3 Installation Package", External Sensor .NET/Code examples: `TestSensor.cs, SensorConfig.cs, Generator.cs, PositionCallbackDlg.cs`

Example 2: sensor from scratch

- Use the existing class called Class1, or create a new one and add the Sensor namespace at the top of the file: using `ABB.PickMaster.ExternalSensor;`
- Let the class inherit from `ISensorRuntime` and implement all methods from the four interfaces.

Build the sensor

Build the solution. This will generate `mySensor.dll`.

Continues on next page

4.6.3 Implementing and testing a simple External Sensor with C#
Continued

Test the sensor

The test is based on the sensor implemented from code examples.

The goal is to send positions from the external sensor to PickMaster and let PickMaster distribute the position for actual picking by a virtual robot.

External Sensor .NET

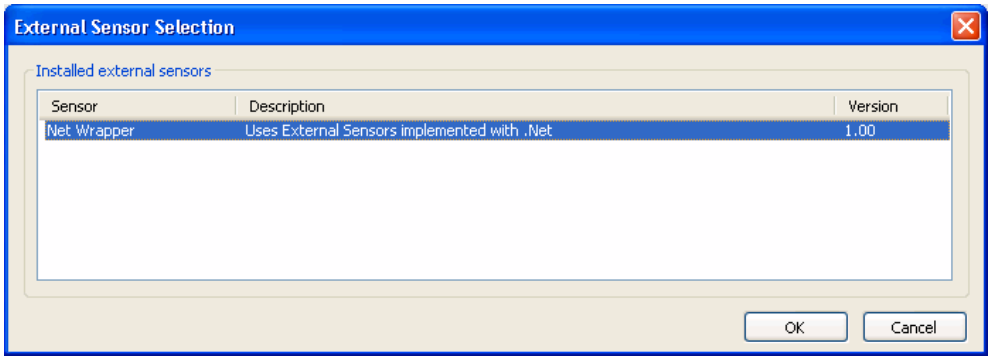
Copy `mySensor dll` to the installation folder of External Sensor .NET.

RobotStudio

- Start RobotStudio and create a station having a virtual robot controller for an IRB360 with the RobotWare option Prepared for PickMaster.
- Load and start `ppacal.prg` to calibrate an indexed work object in a position the robot can reach it. Later, positions sent from the external sensor will be related to this work object.
- In the program dialog, select **IdxWobj1** for calibration and complete all steps of the the calibration procedure. When defining the work object, the z-axis shall be directed up.

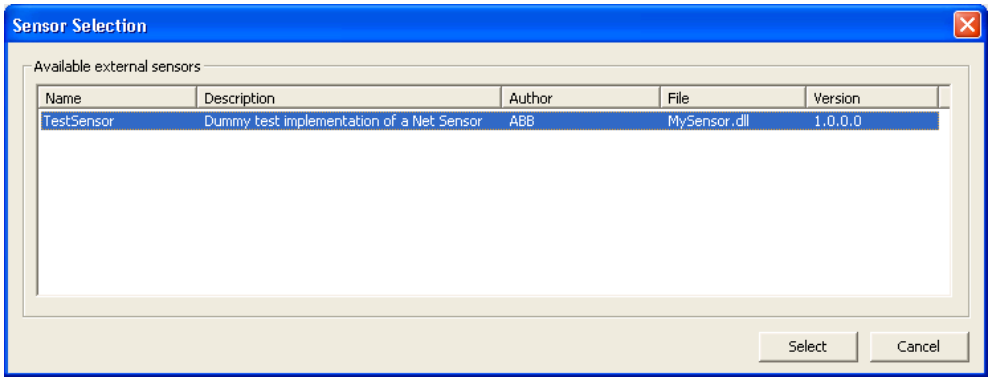
PickMaster

- Launch PickMaster and create a new line.
- Create a new external sensor, in the pop-up window select **NetWrapper** and press **ok**.



en1000001211

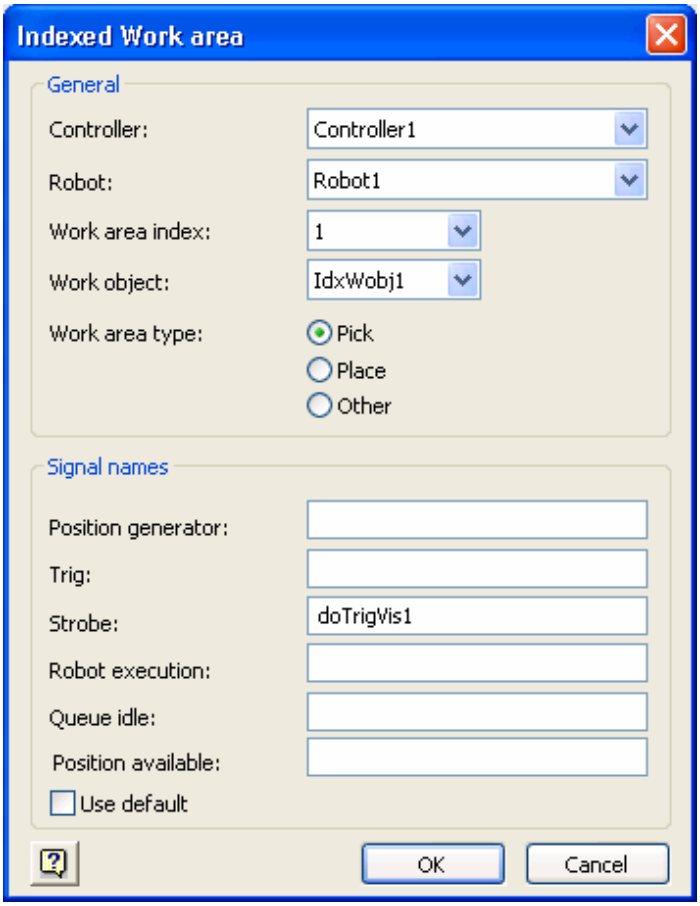
- Select your **TestSensor** (which now shall appear in a new pop-up window) and press **Select**.



en1000001212

Continues on next page

- Create a new IRC5 controller. Select the **virtual controller** created with RobotStudio.
- Create a **new indexed work area**. Select a suitable strobe signal. Leave the other signals as blank.

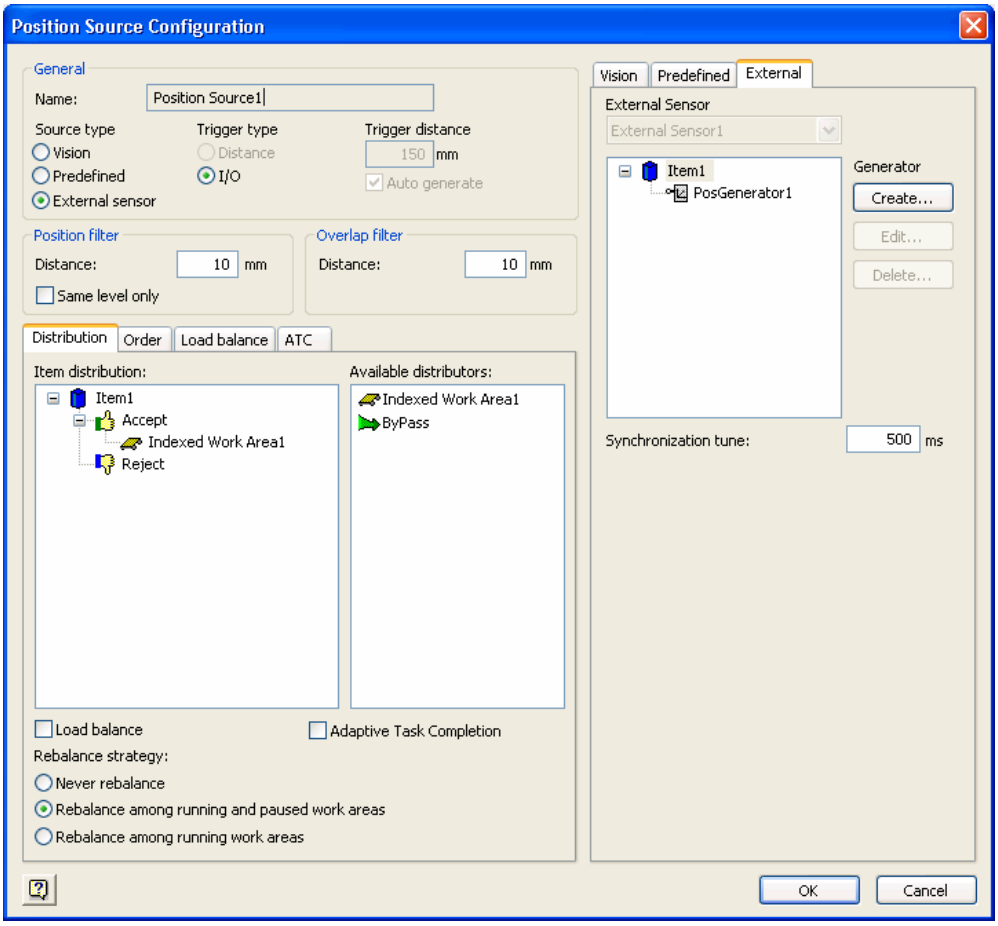


en1000001213

- Create a **new project**.
- Create a **new item**.
- Create a **new position source**, select **external sensor** as source type, select **trigger type I/O**, and select the **created external sensor**. Create a **generator** and select the just created item as object to model. Confirm the pop-up window from the external sensor. Set synchronization tune to 500ms. Distribute the item to the indexed work area as accepted.

Continues on next page

4.6.3 Implementing and testing a simple External Sensor with C#
Continued



en1000001214

- Import the RAPID template program `PMppa360_IRC5.prg` found in the PickMaster installation folder to the robot. Edit the RAPID program and modify to pick from the indexed work object.

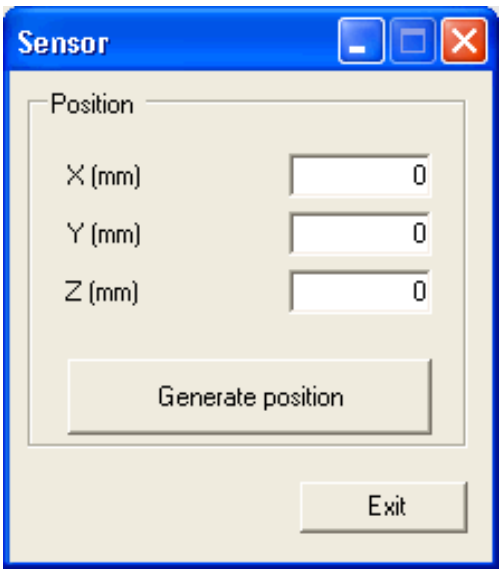
```
PROC PickPlaceSeq()
Pick PickIndex{1};
! Pick PickIndex{2};
! Place PlaceIndex{1};
! Place PlaceIndex{2};
ENDPROC
```

Continues on next page

Run the project

Start the project.

The robot will move to the home position and then wait on `GetItmTgt` in the Pick routine. An external sensor pop-up window will appear from where positions can be generated to PickMaster.



en1000001215

While generating a new position, it is important that the time stamp supplied with the position matches the time when the indexed work area is strobed. With the test sensor, the time stamp is recorded at the same time as the position is sent to PickMaster. In a real implementation, the positions are often sent to PickMaster a time after the time stamp is recorded. That is because it normally takes some time to process recorded sensor data before the positions can be calculated.

Generate a position for the robot to pick in the following way:

1. On the external sensor window, configure x, y and z for a target position that the robot shall pick.
2. Click **Generate position** on Sensor window and after 500ms (which is equal to the configured Synchronization tune) pulse the strobe signal of the indexed work area. For instance, by using the virtual FlexPendant. If the timing is good enough, PickMaster will send the position to the robot for picking. If the timing was not good enough, PickMaster will log a message that there is a mismatch in trigger/strobe time. If it fails just try again until the robot makes the pick.

4.7 RIS Proxy

RIS proxy

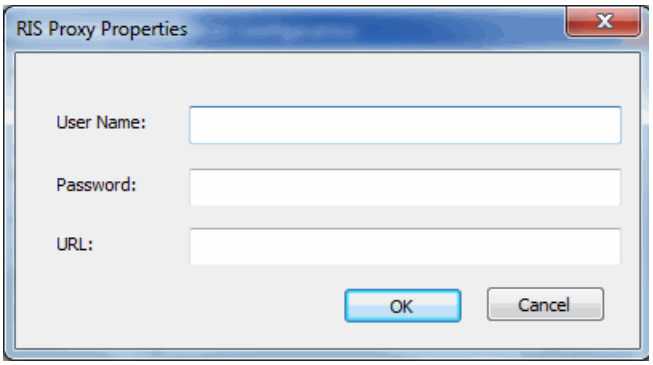
Introduction

The RIS Proxy is a standalone application used to access the PickMaster Remote Integration Service (RIS). Using multiple instances of the RIS Proxy provides the possibility to use different RIS plugins simultaneously. RIS2 functionality is provided through the TCP/IP plug-in.

Starting the RIS proxy application

1 Click `RISProxyu.exe` under RISProxy folder. For example: `C:\Program Files (x86)\ABB Industrial IT\Robotics IT\RIS Proxy\RISProxyu.exe`

The **RIS Proxy Properties** window is displayed.



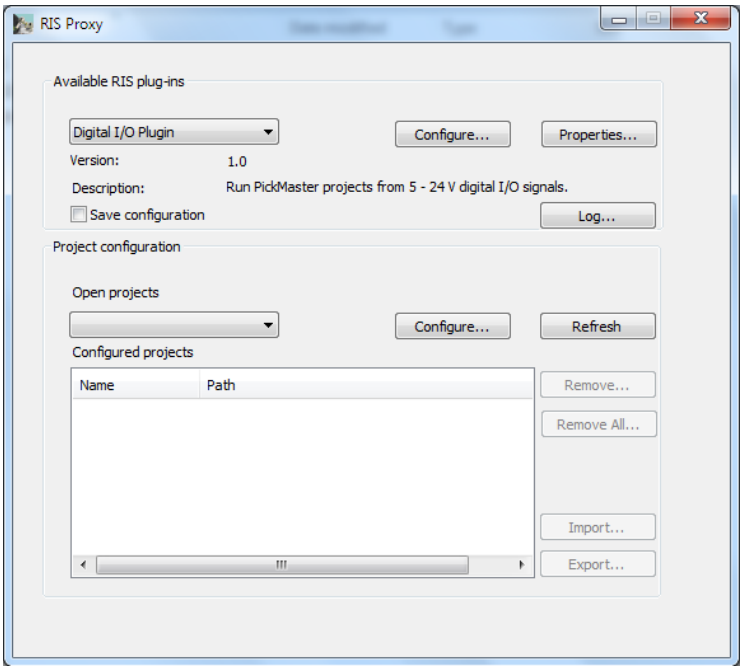
en1000001382

2 Type the PickMaster credentials and the URL of the PickMaster host computer.

3 Click **OK**.

Continues on next page

The **RIS proxy** window is displayed.



en1000001383

> **Note**
>
> The **RIS proxy** is displayed only when the RIS2 Plug-in is selected while configuring RIS. For more information, see *Configuring Remote Integration Services on page 145*.

4 Select a plug-in from the **Available RIS Plug-ins** drop-down menu. For more information on various available Plug-ins, see *Configuring Remote Integration Services on page 145*.

> **Note**
>
> Select the **Save configuration** checkbox if you want to save the selected configuration.

5 Click **Configure**.

> **Tip**
>
> You can access **RIS Proxpy Properties** by clicking the **Properties** button.

6 Click **Refresh**.

The RIS Proxy application with the changes in the PickMaster is updated.

7 Click **Log**.

The RIS Proxy application event log is displayed.

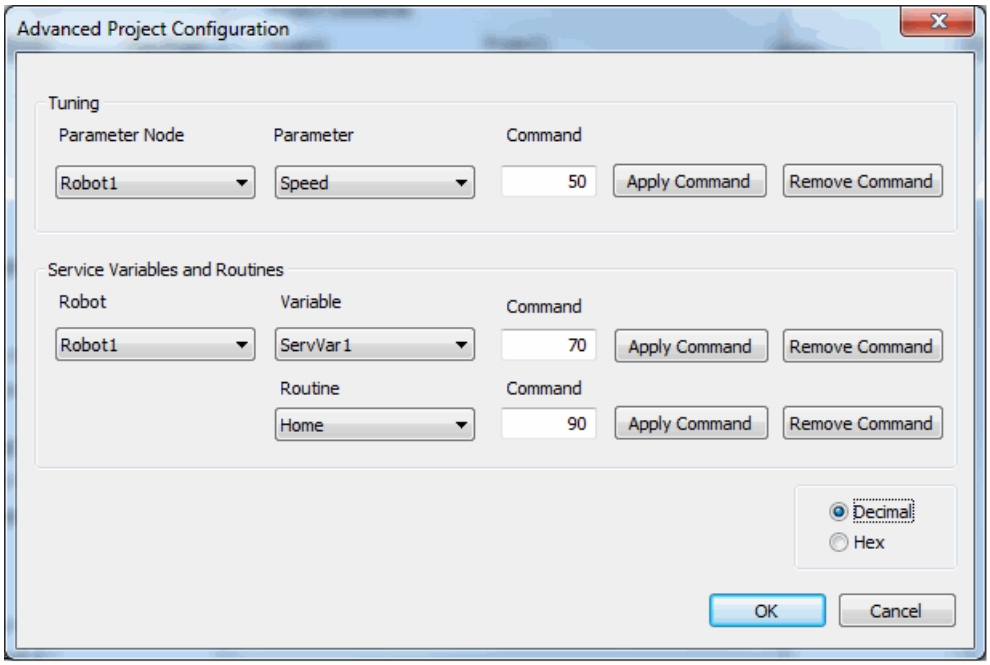
Continues on next page

8 Configure the project. For more information, see *Configuring the project on page 168*.

If **TCP/IP plug-in** is selected, the **Import** and **Export** buttons are enabled. For more information, see *Exporting and importing the RIS configuration on page 153*.

The RIS Proxy **TCP/IP plug-in** is the same as in PickMaster except that it provides RIS2 functionality and an extended communication protocol.

To access the RIS2 functionality configuration, click **Advanced** in the **Project Configuration** dialog.



en1000001384

The **Advanced Project Configuration** provides tuning of parameters and service variables and routines. Click **Apply Command** to define a new command, and **Remove Command** to remove a command. Click **OK** to accept.

Communication protocol

The plug-in is listening for incoming commands while running. These commands must be at least 2 bytes in length.

The responses are sent in 4 bytes, with the response type in the first 2 bytes and the status type in the next 2 bytes.The bytes are sent in the network byte order, big endian. This means that the most significant byte is sent first. The command 256 (0x0100) must be sent as `0x01`, `0x00`. A response of command 10 (0x000A) with argument 5 (0x0005) is sent as `0x00`, `0x0A`, `0x00`, `0x05`.

Example 1:

Assume that the command value for Get robot status is *102 (0x0066)*. To query PickMaster for a robot status, the command must be executed as follows:

If the robot is paused, the response will look like this:

Byte1	Byte2
102 (0x66)	*0 (0x00)*

Continues on next page

Byte3	Byte2	Byte1	Byte0
3 (0x03)	*0 (0x00)*	*102 (0x66)*	*0 (0x00)*

It is also possible to send an argument with the commands in the communication protocol to the TCP/IP plug-in. This is used with the advanced project configuration commands. The argument is a float value, which means that the command must be 6 bytes in length. The first two bytes are the command value, the third byte is the sign, the fourth the exponent and the last two are the significand.

Example 2:

Assume the command value for PickTime for a work area is *50 (0x0032)* and the time to set is 0.04. 0.04 is positive, hence the sign is 1 (0x01), the radix point is placed two steps from the back, hence the exponent is 2 (0x02), the significand is 4 (0x04).

The command must be executed as follows:

Byte5	Byte4	Byte3	Byte2	Byte1	Byte0
4(0x04)	*0(0x00)*	*2 (0x02)*	*1 (0x01)*	*50 (0x32)*	*0 (0x00)*

The status type in the response will be 1 for success and 0 for failure.

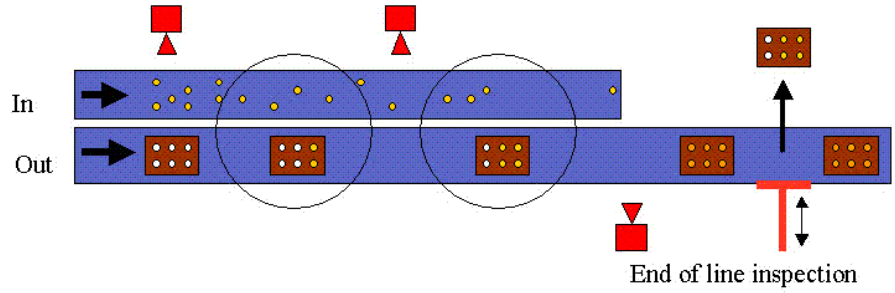
4.8 Post inspection

Introduction

PickMaster 3 can be configured for object recognition, inspection and allow external equipment to take appropriate action. A digital output signal in the robot controller is coordinated with the recognized object on a conveyor. The digital output is connected to the peripheral equipment to indicate the presence of the specific object. The function can be used to inspect objects after they have passed the robot zone or before they enter.

Note

Post Inspection is performed after the place operation is executed. You will need an encoder card for each post inspection camera, to be able to use post inspection.



en0900000882

1. Insert a work area for the conveyor where the post inspection should be done. Set the Work area type to: **Other**
2. Create a camera and calibrate. For more information on camera calibration, see *Configuring the camera on page 68*.
3. Define the base frame distance between the robot and the camera origin in the *MOC.cfg* file.

   Example: *MOC.cfg*

   SINGLE:

   -name "CNV3" -use_single_type "CNV3"\

   -use_joint "CNV3"\ -base_frame_pos_x 1\

   -base_frame_pos_y 0 -base_frame_pos_z 0\

   -base_frame_orient_u0 0.706584\

   -base_frame_orient_u1 0.0100199\

   -base_frame_orient_u2 -0.000157191\

   -base_frame_orient_u3 0.707558

   The conveyor and the robots x-axis are aligned and the distance between the robots origin and the camera origin is 1000 mm.

Continues on next page

4 Create a position source and define the model that shall be recognized by the post inspection camera. Distribute the item to the post inspection conveyor work area.

5 Define enter and exit limits for the post inspection conveyor work area. When the item is between the enter and exit limits the Position Available signal, for the post inspection conveyor work area, will be set high and can be used by external equipment. If the enter value is 300 mm, the Position Available signal will go high when the item is 1.3 m from the robots origin (base_frame_pos_x + Enter value).

6 Set the exit limit to slightly higher than the enter value, in this case it may be 305. The distance between the enter and exit limits will affect for how long the Position Available signal will remain high.

This page is intentionally left blank

5 Running in production

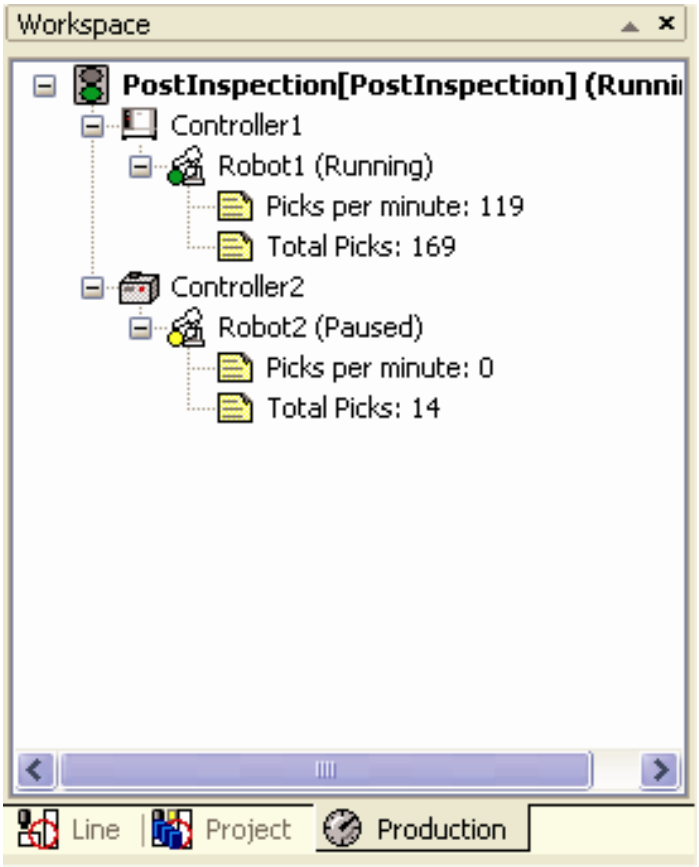
5.1 Running PickMaster

5.1.1 Managing the robot in production

Starting production

Start and stop the project from the Production menu. You can also use toolbar buttons and accelerator keys. F7 starts and Shift + F7 stops the project.

During runtime, the robots are accessed from the **Production** tab in the **Workspace** area.



en0900000533

Prerequisites

The line and the project must be configured to start production.

The project it must be open and active.

Pick rate

The pick rate is shown as icons in the **Production** tab when a robot is running.

The following values are shown:

- Number of pick during the last minute.

Continues on next page

5.1.1 Managing the robot in production
Continued

- Total number of picks since the project was started.

Robot states

The robot can be in different states.

State	Description
Running	The robot can pick and place items.
Paused	The robot is paused in motors off state, or the RAPID program has stopped.
Emergency State	The robot is in emergency stop state.
Stopped	The robot is stopped, that is no items are handled by the robot or distributed to the robot.

Pausing, stopping, and resuming the robot

It is possible to pause or stop a robot during runtime.

Click a robot icon in the **Production** tab and select action from the popup menu.

If more than one robot is connected to a controller (*MultiMove*):

- Restart from stopped state must be performed at the same time for all robots. To do this, right-click the controller icon in the production tab and select **Restart Robots**.
- Stopping one robot will also stop the other robots on the same robot controller.

Note

When pausing a robot while it is waiting on `GetItmTgt` in RAPID, the robot will not go to the Home position until next target is received.

Emergency stop

In case of emergency:

1. Press the emergency stop button on the robot controller or the FlexPendant to stop the robot immediately.

   This sets the controller in emergency state and a warning is displayed on the FlexPendant and in PickMaster.
2. Fix the problem.
3. Release the emergency button.
4. Then acknowledge and reset the emergency state on the FlexPendant or using the popup menu before you restart the robot.

CAUTION

Emergency stop should not be used for normal program stops as this causes extra, unnecessary wear on the robot.

Continues on next page

Note

If an emergency stop occurs after a pause has been requested and before the robot has stopped, the robot goes to the stopped state.

5.1.2 Showing live images

Live images

It is possible to view images from each camera when a project is running.

> **Note**
>
> Showing runtime images requires much processing power and should not be used for a long period of time if complex vision models are used.

Showing live images

To show images, click **Runtime Image** on the popup menu on a vision defined position source. The camera images are shown in the **Vision** tab.

The found objects are shown as green or blue crosses, depending on if they are marked as accepted or rejected by the vision model. See *Vision modeling on page 113*.

5.1.3 Detailed vision information

Detailed vision information

More detailed information than given by the live images is shown in the **Detailed Vision Information** dialog. This dialog box keeps a buffer of images and information about the corresponding vision model hits.

Sequences of images can be record to the buffer and then analyzed individually.

While recording, images are saved in the buffer in a first in, first out basis and the latest image is shown in the dialog.

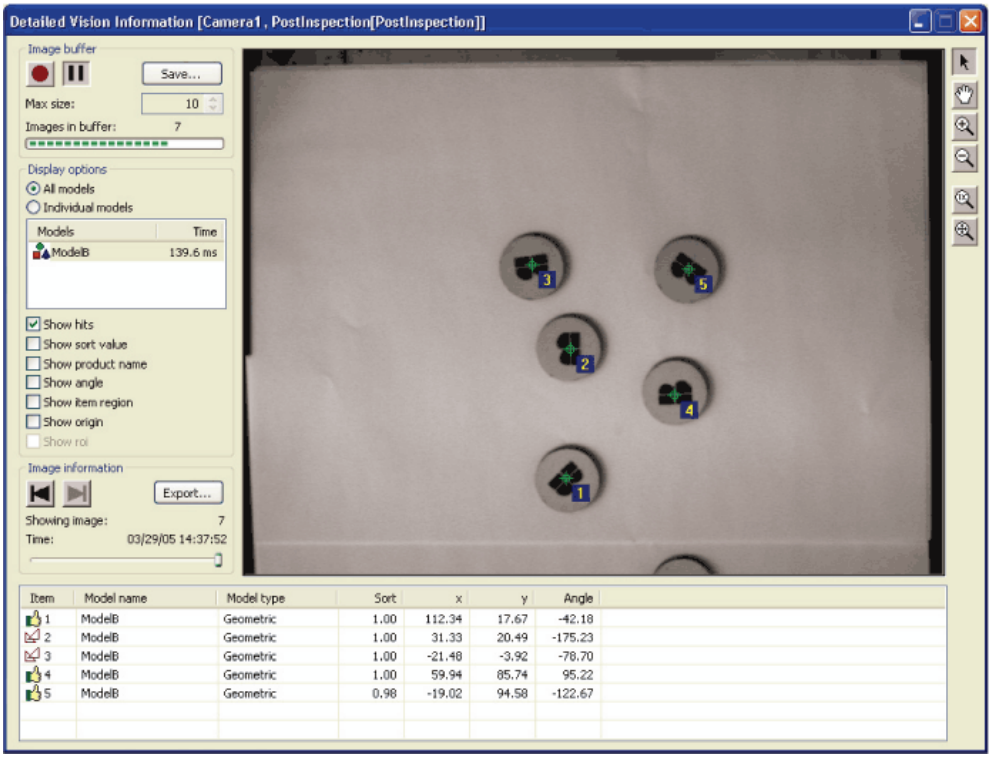
When clicking pause, images are no longer added and the images in the buffer can be analyzed. Save the images in the current buffer to file for later analysis with the Vision Analyzer program, see *Vision Analyzer on page 198*.

One dialog can be opened for each vision defined position source (the name of the camera and the project is shown in the title bar). You can continue working with the PickMaster program while the dialog is opened. Reduce the window size if the window is blocking other information.

The maximum size of the buffer depends on the RAM memory on the computer. If several dialogs are opened at the same time and the maximum memory size is reached, you can reduce the buffer for one dialog to increase the buffer for another.

Illustration, Detailed Vision Information

Click **Detailed Vision Information** on the popup menu of a vision defined position source to open the dialog. By default, the recording state is activated and the buffer is set to 10 images.



xx0800000340

Continues on next page

5.1.3 Detailed vision information
Continued

Image buffer	Used to switch between recording or pause and set the image buffer size. Click **Save** to save all images in the buffer to a .pmv file.
Display options	Select which vision models to display, all together or individually, and other settings for what to show in the images. The settings are valid both for recording and pause.
Image information	Step through the image buffer when recording is paused. ALT + LEFT or RIGHT ARROW can also be used to step. Click **Export** to save he current image to file (.bmp format).
	The list view at the bottom shows information about all the hits. When an individual model is selected, the columns change depending on its type.
	The pan and zoom buttons can be used to analyze the image more closely.

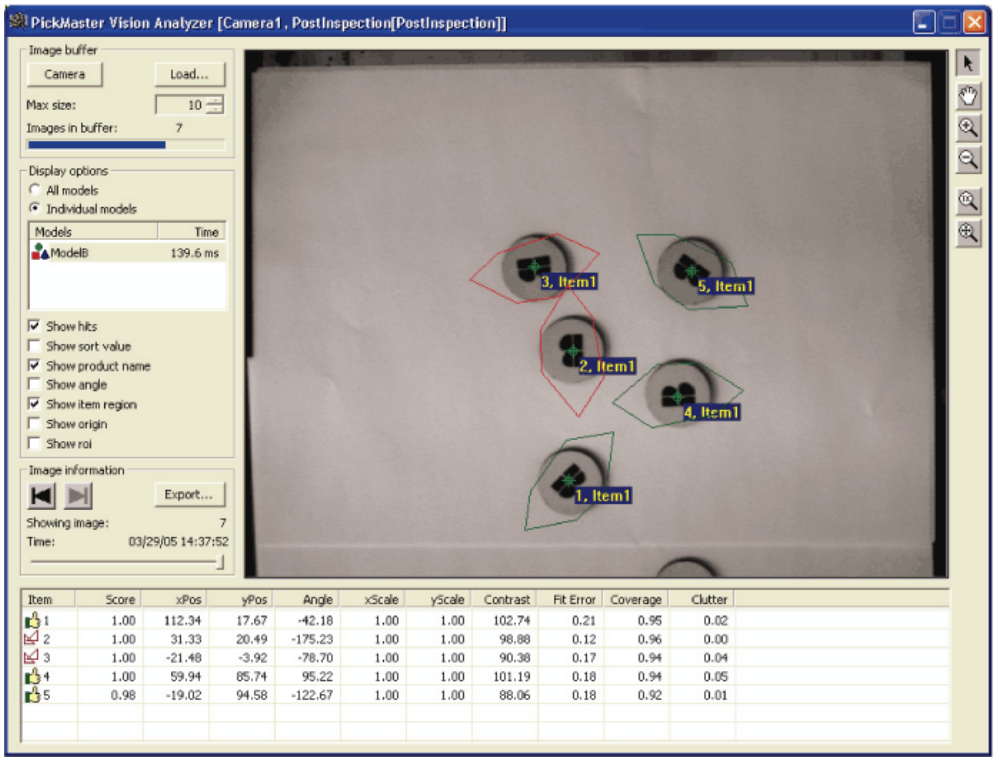
Note

Only overlapping item regions in the same image are marked as overlapped but no robot will access items with regions that overlap with item regions in consecutive images.

Vision Analyzer

Image buffers recorded in the **Detailed Vision Information** dialog can be saved as .pmv files. These files can be viewed with a separate program called PickMaster Vision Analyzer.

Start Vision Analyzer from the **File** menu in PickMaster or from Windows Start menu.



xx0800000342

Click **Load** to open a .pmv file.

Continues on next page

Click **Camera** to see detailed information about the camera that took the images.

Other settings in Vision Analyzer are identical to settings in **Detailed Vision Information**.

Vision Analyzer does not require PickMaster or any vision hardware and can be used stand alone.

5.1.4 Messages and error logs

Introduction to messages and error logs

When an event or a problem occurs it is displayed as a message in the **Log** area. All messages can also be seen from Windows Event Log. Therefore it is important to check the log when starting up a project and regularly during runtime. There can be warnings or errors issued by a robot controller that needs to be handled.

Related information

Administering the event log on page 208.

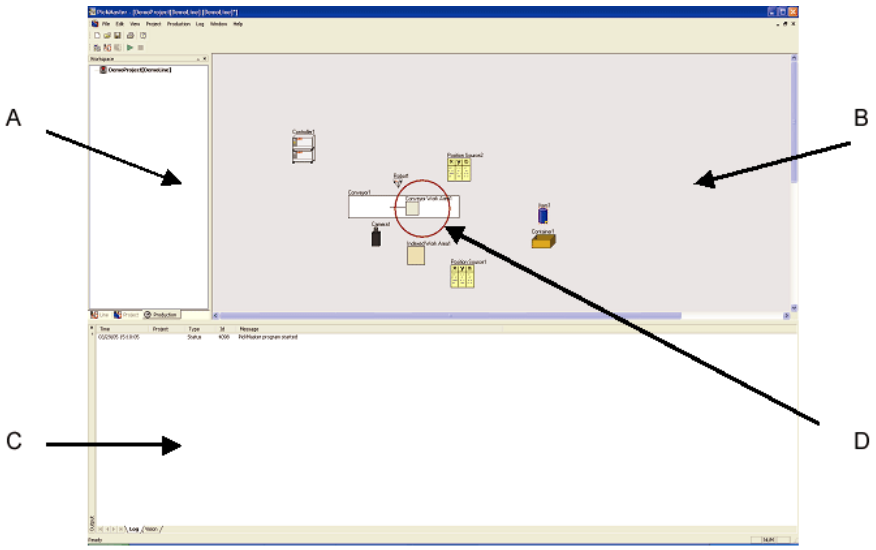
All error messages are listed in *Operating manual - Troubleshooting IRC5*.

5.2 The user interface

5.2.1 PickMaster user interface

PickMaster user interface

The PickMaster program is built as a studio environment where all functionality can be accessed from within the same program.



en0900000360

A	Workspace area
B	Line view and project view
C	Log area and vision area
D	PickMaster objects

The line view and the project view

The combined line view and project view is the central area of the program. From here, all objects are created and configured. Right-click any object or the view itself to open a menu with a list of options for that object.

The orientation and placing of the objects in the view has no relevance for the configuration. It is however convenient to place the objects so that the view looks like the physical installation.

The workspace area

The workspace area has three tabs:

- The **Project** tab shows the objects and their connections (all open projects).
- The **Production** tab is used for starting, stopping, and pausing projects or individual robots. It also shows picking statistics.
- The **Line** tab has no functionality in the current version of PickMaster.

Double-click a project in the **Production** or **Project** tab to open the project in the project view.

Continues on next page

5.2.1 PickMaster user interface
Continued

The log area

All messages from the robot system are shown in the **Log** area. The message contains a time stamp, an error id, and a short description. The messages are categorized as Status, Warning, or Error. Right-click the log area to clear or save the log to a file.

All messages are sent to Windows Event Log. Right-click the log area to open Windows *Event Viewer*.

See *Administering the event log on page 208*.

The vision area

When showing live images during configuration or when running a project, the images appear in the **Vision** area.

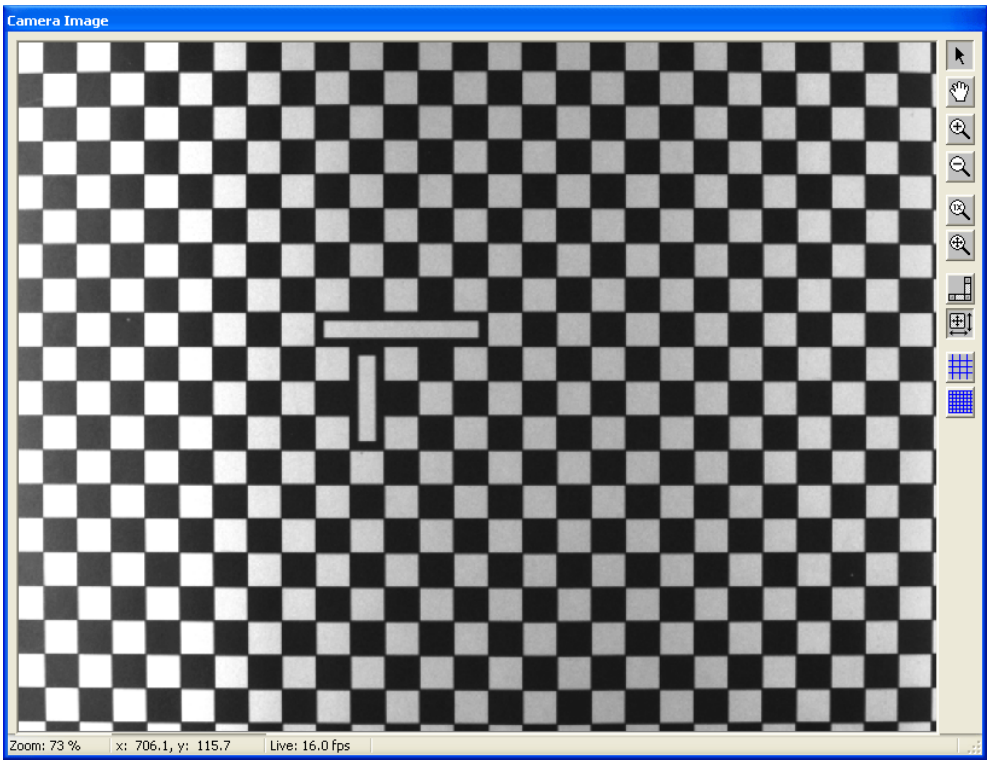
5.2.2 The image windows

The image windows

When configuring a camera or a vision model the camera image is shown in a separate window. The image window is resizable and provides tools to quickly zoom and pan the shown image. Some tools change the appearance of the mouse pointer.

To zoom using the keyboard and mouse, place the pointer over the image, press CTRL and scroll the mouse wheel.

The current zoom level and the world coordinate of the mouse pointer is shown in the status bar. When live images are shown, the current frame rate is also shown in the status bar.



en0900000361

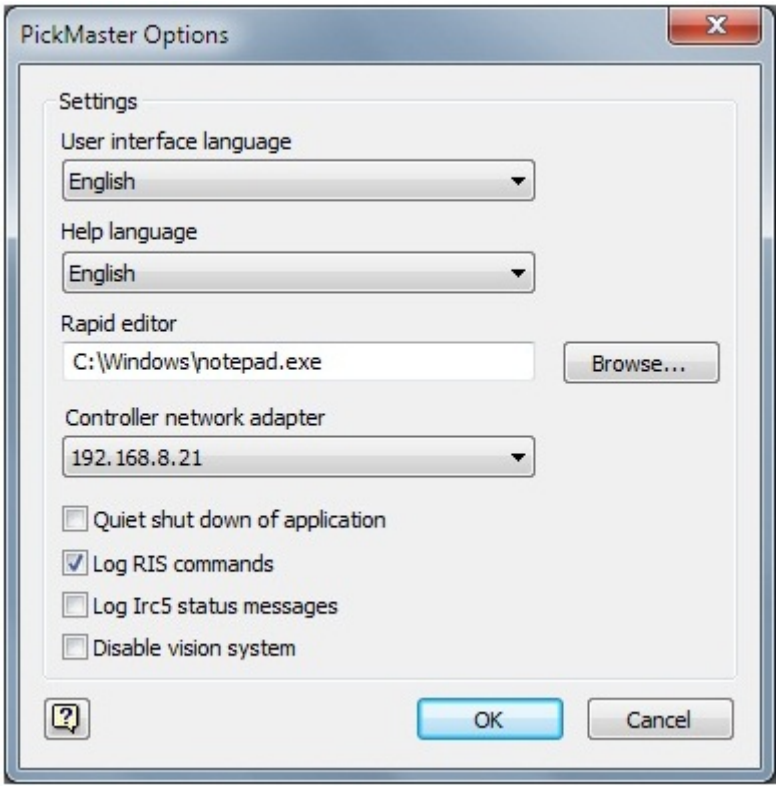
5.2.3 Setting PickMaster options

Setting PickMaster options

Use this procedure to set PickMaster options.

1 On the **File** menu, select **Options**.

The **PickMaster Options** dialog is opened.



en1200000665

2 Select language from the **User interface language** box.

The language is changed after restarting PickMaster.

3 Select language for the online help in the **Help language** box.

4 Select RAPID editor for editing RAPID programs.

5 Select the IP-address of the network adapter that communicates with the robot controller(s) from the **Controller network adapter** box.

6 If needed, select **Quiet shut down of application**.

As default you cannot shut down PickMaster when a project running, all projects must be stopped before the program can be closed. If **Quiet shut down of application** is selected, PickMaster shuts down even if there is a project running. PickMaster will then stop any running project and shut down within 15 seconds.

7 If needed, select **Log RIS commands** to show all RIS commands in the log area.

8 If needed, select **Log IRC5 status messages** to include showing status messages in the log area.

Continues on next page

As default, only warnings and error messages are shown in the log area.

9 Select **Disable vision system** if PickMaster's internal vision system should not be used. PickMaster will then not connect to any attached cameras. This is useful to avoid conflicts when an external vision system is used through the external sensor interface.

10 Click **OK**.

5.3 Tuning

5.3.1 Tuning

Introduction to tuning

Sometimes, the exact pick and place positions are not exactly where expected. This might be caused by a small error in the calibration of either the camera or the work area. It is possible to adjust the positions while running a project. This is called tuning.

Tuning work area

The pick and place positions can be tuned for each work area using the **Work Area Tune** window.

	Action
1	Click **Tune**. The **Work Area Tune** window is displayed.
2	Tune the values x, y, and z.
3	Click **Apply**. The new values are sent to the robot controller.

Work area settings

Most of the work area settings can be tuned when a project is running. To open **Work Area Settings**, click **Settings** in the popup menu for the work area.

New values are sent to the robot controller when you click **Apply**.

Robot settings

The robot settings can be tuned when a project is running, using the **Robot Settings** window. See *Configuring the robot settings on page 112*.

Limitations

The RAPID program must be running to tune the `Robot speed`, `tool vacuum times`, and `pick/place times` parameters. Other parameters like `enter/exit limits` can be tuned without running the RAPID program.

6 Troubleshooting

6.1 Introduction to troubleshooting

Troubleshooting

This chapter describes some of the most common troubles known when installing, configuring, or running PickMaster.

A fault in the robot system first appears as a symptom, which can be:

- An event log message that can be viewed using PickMaster, FlexPendant, RobotStudio, or Windows Event Viewer.
- The system is performing poorly or displaying mechanical disturbances.
- The system can not be started or displays irrational behavior during start.
- Indications on the hardware, such as LEDs.
- Other types of symptoms. The robot system is complex and has a large number of functions and function combinations.

Related information

Generic troubleshooting and all error messages in the robot system are listed in *Operating manual - Troubleshooting IRC5*.

Administering the event log on page 208.

6.2 Administering the event log

The event log

The event log messages that are displayed in the log area of PickMaster are also stored in the Windows event log. The messages can be viewed with Windows *Event Viewer*.

Administering the event log

Use this procedure to administer the event log using Windows *Event Viewer*.

1. Right-click the log area and select **Event Viewer**.

   The Event Viewer can also be started from Windows Control Panel.
2. In the **Event Viewer** tree list, select **Application**.
3. To see only PickMaster messages, right-click **Application**, select **View** and click **Filter**. Then select **PickMaster** as event source.
4. To save the log, right-click **Application**, and select **Save Log File As**.

   The log can be examined over the network from another computer. To see logs from another computer, right-click **Event Viewer**, select **Connect to another computer** and then locate the computer on the network.
5. To setup how the log size is handled, right-click **Application**, and select **Properties**. To ensure that the log file never fails to write events to the log select **Overwrite events as needed**.

Related information

See Windows help for *Event Viewer*.

6.3 Fault symptoms or errors

6.3.1 Warnings 4326 - 4329

Verification actions

The following are the general verification actions for the warning 4326, 4327, 4328, and 4329. For more detailed explanation, see *Warning 4326 on page 210*, *Warning 4327 on page 210*, *Warning 4328 and 4329 received together on page 211*, *Warning 4328 received without 4329 on page 212*, and *Warning 4329 received without 4328 on page 212*.

Action 1

Check the selection of signals for trigger and strobe in the work area configuration of the PickMaster line. Check that the I/O configurations of these signals correspond to the wiring.

Action 2

Check all the trig/strobe wiring. Check if the trig and strobe cables are mixed up. Make sure that the cables are shielded, properly attached and grounded the right way. There should be no current in the shield. Make sure that sources for 24 volt are not mixed. The controller system parameter *SyncSeparation*(Topic: I/O, Type: Fieldbus Command, Name: CNVX) can be modified to filter strobe input events from a camera or sensor.

Action 3

Check all the LAN cables on the robot network. Make sure that the cables are shielded and properly attached. Check that the right IP address, default gateway, and subnet mask is defined (on both PC and robot controller). Note that all three values must be defined even if there is only one computer and one robot controller on the network. For more information, see *Configuring the networks on page 45* .

Action 4

See *Configuring the networks on page 45* .

Action 5

Check that the IP address (goto **File** and click **Options**) in the field "Controller Network Adapter" is the address of the network interface card in the PC that communicates with the robot controller. Check if time sync service has trouble to connect to controller. Stop the service for 30 seconds and then restart it again. Check that there are no firewalls active that are affecting the time synchronization services.

Action 6

Reduce the trigger frequency Sometimes the trigger distance is very short causing the system to trigger much more often than it can handle. How often a trigger can be handled depends on how complicated the models are that are used on the system. Sometimes the frequent triggering can be caused by faulty trigger/strobe wiring or electrical noise.

Continues on next page

6.3.1 Warnings 4326 - 4329
Continued

Action 7

Some switches are buffering data that needs to be present. This buffering time might be too long. Try to switch to a simple hub or to decrease this buffer time. Make sure that you have the newest software running on the hub/switch. Make sure that there are no infinite loops in the RAPID code because it will affect the robot network communication .

Action 8

Debug the implementation of the external sensor.

Action 9

For external sensors there might be a small constant delay between the strobe pulses and the recording of time stamps (For example, if the trigger signal is cross connected with the strobe). Modify the Position Source parameter *Synchronization tune* to modify all time stamps sent to PickMaster with a constant time value.

Warning 4326

For verification actions, see the preceding section.

Error description:

4326 Item positions lost on %s due to missing strobe. See Application manual.

Probable causes:

The following table provides the probable causes of the warning 4326:

Probable cause	Verfication actions
If work area is conveyor:	
The conveyor board does not receive any strobe pulses on the start input.	*Action 1 on page 209*, *Action 2 on page 209*
The strobe signal is not configured as cXNewObjStrobe.	*Action 1 on page 209*
PickMaster has no connection with the robot controller.	*Action 3 on page 209*
If work area is indexed:	
The configured strobe signal does not receive a strobe pulses.	*Action 1 on page 209*, *Action 2 on page 209*
PickMaster has no connection with the robot controller.	*Action 3 on page 209*

Warning 4327

Error description:

4327 Expected item positions missing from %s. See Application manual.

Probable causes:

The following table provides the probable causes of the warning 4327:

Probable cause	Verfication actions
If source type is camera:	
The camera does not receive trigger pulses.	*Action 1 on page 209*, *Action 2 on page 209*
PickMaster has no connection with the camera.	*Action 4 on page 209*

Continues on next page

Probable cause	Verfication actions
If source type is external sensor:	
The external sensor does not receive any trigger pulses.	*Action 1 on page 209*, *Action 2 on page 209*
The external sensor does not send any positions to PickMaster.	*Action 8 on page 210*
If source type is external sensor:	
The external sensor does not receive any trigger pulses.	*Action 1 on page 209*, *Action 2 on page 209*
The external sensor does not send any positions to PickMaster.	*Action 8 on page 210*
If source type is predefined and work area is conveyor:	
The conveyor board does not receive any strobe pulses on the start input.	*Action 1 on page 209*, *Action 2 on page 209*
The strobe signal is not configured as cXNewObjStrobe.	*Action 8 on page 210*
PickMaster has no connection with the robot controller	*Action 3 on page 209*
If source type is predefined and work area is indexed:	
The configured strobe signal does not receive an strobe pulses.	*Action 1 on page 209*, *Action 2 on page 209*
PickMaster has no connection with the robot controller.	*Action 3 on page 209*

Warning 4328 and 4329 received together

Error description:

Typically, a pair of 4328 and 4329 is received for one, several or every trigger/strobe related to a work area.

4328 Trigger/strobe time mismatch (%.1f s). Item positions from %s to %s lost. See Application manual.

4329 Trigger/strobe time mismatch (%.1f s). Strobe from %s was ignored. See Application manual.

Probable causes:

The following table provides the probable causes of the warning 4328 and 4329:

Probable cause	Verfication actions
In order of probability:	
The time synchronisation between controllers and PickMaster is not working.	*Action 6 on page 209*
The trigger frequency is set too high.	*Action 5 on page 209*
Low robot network performance	*Action 7 on page 210*
Low camera network performance	*Action 4 on page 209*
Additional causes for external sensors:	
Time stamps are not enough synchronized with strobes.	*Action 9 on page 210*
The external sensor does not send positions with a correct time stamp..	*Action 8 on page 210*

Continues on next page

6.3.1 Warnings 4326 - 4329
Continued

Warning 4328 received without 4329

Error description:

4328 Trigger/strobe time mismatch (%.1f s). Item positions from %s to %s lost. See Application manual.

Probable causes:

The following table provides the probable causes of the warning 4328 and 4329:

Probable cause	Verfication actions
The trigger signal is not stable.	*Action 2 on page 209*

Warning 4329 received without 4328

Error description:

4329 Trigger/strobe time mismatch (%.1f s). Strobe from %s was ignored. See Application manual.

Probable causes:

The following table provides the probable causes of the warning 4328 and 4329:

Probable cause	Verfication actions
The strobe signal is not stable.	*Action 2 on page 209*

6.3.2 The camera does not take pictures

Error description

The camera does not take pictures.

Probable causes

There can be several causes why the camera does not take pictures. To check all the possible causes the following must be verified.

- Check that the trig cable is properly connected.
- Check that the camera cable is connected to the correct port.

If the camera is distance trigged, the encoder might not be recording any conveyor movement due to

- bad encoder connection or
- wrong conveyor selected in the work area.

If the camera is I/O trigged, the photo eye might not be sensing any part, due to:

- Wrong connection.
- Bad reflection.

6.3.3 Robot does not move

Error description

The camera is identifying objects, but the robot does not move.

Probable causes

There can be several causes why the robot does not move although the camera takes pictures properly. To check all the possible causes the following must be verified.

- To check that the strobe cable is connected, check the StartSig LED on the encoder board.
- Check the distribution in the Position Source.
- Check the AI *c*Speed* in the I/O list if any speed is detected. If not, check encoder signals.
- Check the AI *c*Position* in the I/O list if any position is tracked. If not, check the distribution in the Position Source.
- Check the direction of travel on the DI *c*DirOfTravel*.
- Monitor the signal *Queue Idle*, to see if the queue gets any positions.
- Monitor the *Position Available* signal, to see if the parts are detected.

6.3.4 Bad or varying position accuracy

Error description

The position accuracy is bad or varying.

Probable causes

There can be several causes why the position accuracy is bad or varying. To check all the possible causes the following must be verified.

- Verify that the *Counts Per Meter* calibration is accurate. Verify several times. Include verification in scheduled maintenance.
- Avoid drive shaft encoders, since belt slippage between roller and belt can vary.
- Check the camera calibration. Poor quality of calibration grid will give inaccurate calibration result.
- Check if there are differences between calibration paper height and product height.
- Check if there are parallax errors when identifying high products.
- Make sure that the camera is not mounted on robot frame because this can cause camera vibrations.

6.3.5 Positions are used twice

Error description

The robot uses every position twice.

Probable causes

There can be several causes why the robot uses every position twice. To check all the possible causes the following must be verified.

- If I/O trigged predefined positions or containers are used, set the *SyncSeparation* filter distance to avoid double and ghost triggers.
- If vision is used, increase the overlap and position filter.
- Clear the checkbox **Same level only** in the Position Source.

If a robot downstream in an ATC group tries to use an already used item, then the Work Area order in the Position Source is incorrect.

6.3.6 Problem with camera resolution in PickMaster

Error description

Camera image size decreases to lower resolution as compared to calibration image resolution.

Probable causes

There can be several causes why camera resolution is decreased. To check all the possible causes the following must be verified:

- Is the factory default configuration is active.
- There could be custom configuration activated. Verify if the custom configuration is having reduced ROI (region of interest).

6.4 Diagnostic files collection

Background

The PickMaster system consists of several different parts and each part has its own diagnostic facility. In some difficult cases, the technical support may require some additional background information from the system. Following is a general instruction on how to collect diagnostic information in the most efficient way.

Prerequisites

Following are the prerequisites for diagnostic files collection:

- Make sure that the system configuration parameter Process/Conveyor systems/CNVx/Reach zone accuracy is set to 0 on every robot controller. As a consequence any use of `UseReachableTargets` is disabled. (See detailed description about Reach zone accuracy in Conveyor Tracking application manual).
- Download and install the "`DebugView`" tool from Microsoft. Use optional settings History Depth: 100000, Clock Time and Show: Milliseconds.

Log file collection

Start the `DebugView` log capture before starting the PickMaster application.

When an issue occurs:

1. Stop the PickMaster project.
2. Save the PickMaster event log.
3. Save the DebugView log.
4. Save the System diagnostic file from every robot controller.
5. Send the collected files to the local technical support.

Enhanced internal logging

Sometimes it is required to enhance the specific logging to be able to analyze a certain issue.

Setting the `AdditionalLogType` key to different values in the file `PickMasteru.exe.config`, will activate different levels of logging for debugging purposed by using tools like `DebugView`. The values will be provided by the technical support based on the nature of every specific case. Default value is 129.

Note

If too many additional log levels are enabled it may affect the CPU consumption.

6.5 Error codes

Common error codes

Error code	Type	Description
4097	Error	Undefined error Reason: The occurred error has not been given a correct error ID but the error message should explain the reason.
4098	Status	Information only
4099	Error	Command line options Reason: PickMaster was given an unknown command line option, e.g. /p, at startup.
4100	Error	Description: Unexpected error Reason: An unexpected error occurred in PickMaster. See the log message for more information.
4101	Error	XML parsing error Reason: There was a problem reading either a pmline or pmproj file. See the log message for further information about where in the file the error occurred.
4197	Error	The project has been upgraded to a later version and the file is marked as modified. The file needs to be saved to make changes permanent.
4198	Error	The line has been upgraded to a later version. If the line itself was opened it is marked as modified and needs to be saved. If a project was opened, the line should be opened and saved before continuing.
4199	Error	The project file has an invalid format. It was either created with a beta version of PickMaster or the file is corrupt.
4200	Error	The PickMaster program failed to access the Windows registry when writing or reading its configuration
4202	Warning	The project is not designed on the current line. When trying to open a project, there is already a project open that is built upon a different line. Reason: Only one line can be used at the same time. Solution: Close any open projects and try to open the project again.
4203	Error	Failed to load the corresponding line when opening a project. The line file may be corrupt
4204	Error	Failed to load a line. The file may be corrupt.
4205	Error	The imported line may need to be recalibrated Reason: If the imported line was designed with other cameras or lenses, the cameras as well as the robot’s base frame must be recalibrated.
4206	Error	The selected RIS plug-in could not be loaded at program startup. The file may be corrupt.
4207	Error	The selected RIS plug-in could not be found at program startup.
4208	Error	One of the previously available lines has been overwritten by another line. The old line will not show up as an available line and projects designed on that line cannot be used.

Continues on next page

Error code	Type	Description
4209	Error	The line file is invalid and cannot be opened.
4210	Error	Failed to load resources for the selected language. The default language (English) will be used instead
4211	Status	A notification about the total number of picks done by a robot until the project was stopped.
4212	Error	Failed to remove the line file. The file must be removed manually.
4213	Warning	Failed to find the html help file for the selected language. Make sure the "Application manual xxx.chm" file is in the Documentation folder in the PickMaster folder.
4216	Error	An attempt to open a file not recognized by PickMaster.
4217	Error	No time synchronization service available. Reason: The **PickMaster Time Synchronization Service** might not be properly installed or not started. Solution: Verify the service is installed and try to restart the service.
4218	Warning	Two or more network adapters are configured on the same subnet: x.x.x Refer to the user guide and review the recommended network settings.
4297	Status	Attempt to start a project that is already running.
4298	Status	Attempt to stop a project that was not started.
4300	Error	A camera is currently in use by another project. Reason: When starting a project, one of the position sources is configured with a camera that is currently in use by another project. Solution: A camera can only be used in production in one project at the same time. Reconfigure one project or run them one at a time.
4301	Error	Failed to start project execution Reason: Internal error probably caused by out of memory. Solution: Try restarting the PickMaster program.
4302	Error	When starting a project, a vision defined position source has no camera defined. Solution: Either remove the position source or configure it with the camera to use.
4303	Error	When starting a project, a position source has no work area defined Solution: Either remove the position source or configure it with the work area to use
4304	Warning	When starting a project, a vision defined position source has no configured vision models. Solution: Either remove the position source or define which vision models to use.
4305	Error	When starting a project, a predefined position source has no object defined. Solution: Edit the position source and define the predefined object to use.

Continues on next page

Error code	Type	Description
4306	Status	A model was edited on a different camera than it was created on. Solution: Check that the correct camera is selected in the position source and retrain the model.
4307	Warning	A vision model was created on a camera that has not been calibrated. Solution: Open the corresponding line and calibrate the camera. Then retrain the model.
4308	Error	When running a project, a vision model found an object but could not find the item or container to refer to. Solution: Stop the project, remove the vision model in question and create a new one for the correct item.
4309	Warning	A container is incorrectly configured. Solution: Check the error message for more information.
4310	Status	Production was successfully started.
4311	Status	Production was successfully stopped.
4312	Warning	Indication that PickMaster is running on a demo license with limited production time. Reason: There is only a demo license installed Solution: Request a fully qualified license to run projects for an unlimited time.
4313	Error	PickMaster is running on a demo license and the allowed production time is exceeded. Solution: Request a fully qualified license or restart the PickMaster program to be able to start a project again
4314	Error	Got scene information from an unknown work area.
4315	Status	The work area that triggers a Position Source has changed. This occurs at project startup or when the robot controller with the previous trigger work area has stopped.
4319	Warning	Received item acknowledgment from an unknown work area.
4320	Warning	A project that used load balancing has been upgraded and a work area order was generated. The work area order must be verified in the Position Source configuration dialog box
4321	Warning	An item acknowledge was received from a work area but the corresponding item position could not be found. Following work areas will not be notified that an item position has already been accessed.
4326	Warning	Item positions lost on work area due to missing strobe. For more information, see *Warnings 4326 - 4329 on page 209*.
4327	Warning	Expected item positions missing from position source. For more information, see *Warnings 4326 - 4329 on page 209*.
4328	Warning	Trigger/strobe time mismatch. Item positions from position souce to work area lost. For more information, see *Warnings 4326 - 4329 on page 209*.
4329	Warning	Trigger/strobe time mismatch. Strobe from work area was ignored. For more information, see *Warnings 4326 - 4329 on page 209*.
4396	Error	A COM error occurred in when using an External Sensor. The log message provides more information.

Continues on next page

Error code	Type	Description
4397	Error	An error occurred when calling a function on an External Sensor COM object. The log message provides more information.
4398	Error	When opening a project with en external position generator, its corresponding sensor could not be found in the used line.
4399	Error	An external sensor failed to start when the project was started. The position source will not be used during production.
4596	Error	General User Hook error. See description for more information.
4797	Error	General license error. See description for more information.
4798	Error	More cameras are used than allowed by the currently installed license. Solution: Either remove cameras or request a new license.
4799	Error	More robot controllers are used than allowed by the currently installed license. Solution: Either remove robot controllers or request a new license.
4800	Error	More cameras are using inspection vision models than allowed by the currently installed license. Solution: Either remove inspection models or request a new license.
4804	Error	More robot controllers are using camera distribution than allowed by the currently installed license Solution: Either make sure not to use more camera distribution than allowed or request a new license.
4805	Error	Attempt to start a project with ATC without an appropriate license. Solution: Request a new license including the ATC option or remove ATC from the project.
4806	Warning	The licence will expire in less than 14 days. Solution: Request a new license.
4807	Error	More External Sensors are used than allowed by the currently installed license. Solution: Either remove External Sensors or request a new license.
4808	Error	Attempt to start a project with conveyors without an appropriate license. Solution: Request a new license including the ATC option or remove all conveyors from the project.
4809	Error	The network adapter (IP-address) not found. Solution: Make sure that the specified network card is enabled and that the IP address of the card has not changed.
4810	Error	Access to Service denied. Reason: PickMaster cannot Access Windows Services.
4811	Error	Cannot access PickMaster Time Synchronization Service. Reason: PickMaster Time Synchronization Service is not installed.
4812	Error	Cannot stop PickMaster Time Synchronization Service.
4813	Error	Cannot start PickMaster Time Synchronization Service.

Continues on next page

Robot error codes

Error code	Type	Description
8193	Status	The robot is running.
8194	Status	The robot is stopped.
8195	Status	The robot is paused
8196	Warning	Please set the robot in auto mode. Reason: The robot is started but the controller is not set to auto mode. Solution: Switch the controller to auto mode.
8197	Warning	Please confirm auto mode (on the FlexPendant). Reason: The robot is started and is set to auto mode but the auto mode is not confirmed. Solution: Confirm the auto mode on the FlexPendant.
8198	Status	The robot is in auto mode.
8199	Error	Robot error X (where X is the robot error number). Solution: See the robot documentation for the specific error.
8200	Warning	Robot warning X (where X is the robot warning number). Solution: See the robot documentation for the specific warning.
8201	Warning	Robot program controller in unknown state. Reason: The robot was started but the program controller is in an unknown state.
8202	Warning	Guard stop Reason: The robot has been stopped because a guard has been activated.
8203	Warning	Emergency stop Reason: The robot has been stopped because of an activation of the emergency stop Solution: Remove the reason for the stop and reset the emergency stop. Restart the robot (can be done without stopping the project).
8204	Status	Rapid program stopped
8205	Status	Rapid program has been restarted
8209	Status	Robot controller is in system failure Reason: See event log on the controller for more information
8211	Error	Lost connection Reason: The computer lost the connection to the controller. The network connection can be down. The controller can be shut off or lost its power. Solution: Make sure that the controller is on and has power supply. Also make sure that the network connection is working.
8212	Warning	A robot controller is used by another project Reason: A robot controller may only be used by one project at a time
8213	Warning	Robot controller not in use and may not be accessed. Reason: An attempt was made to access a robot controller that was not configured to be used in the project.

Continues on next page

Error code	Type	Description
8293	Error	Failed to set motors on. Reason: PickMaster failed to set motors on. Some system state prevents PickMaster from setting the motors to on (e.g. emergency stop, guard stop etc.).
8294	Error	Failed to start the RAPID program.
8295	Error	Failed to prepare the RAPID program for start.
8297	Error	Failed to set the RAPID variable “RoutineName” to “ClearAll” Reason: The variable “RoutineName” is probably missing or is of the wrong type (should be a string type).. Solution: Ensure that the variable exists and is of the string type.
8298	Error	Failed to get the robot controller states. Solution: Ensure that the controller is up and running OK. If not, reboot the controller.
8299	Error	Failed to get events from the robot controller. Solution: Ensure that the controller is up and running OK. If not, reboot the controller. Ensure that the correct network adapter is used for the specific controller in the line.
8300	Error	Failed to set the RAPID variable “StopProcess” to TRUE. Solution: Ensure that the RAPID variable“ StopProcess” exists and is of type bool.
8302	Error	Failed to set the RAPID variable “RoutineName” to “Pick-Place”. Reason: The variable “RoutineName” is probably missing or is of the wrong type (should be a string type). Solution: Ensure that the variable exists and is of the string type.
8303	Internal Error	The system failed to apply a new work area tune because the work area ID does not exist.
8304	Internal Error	The system failed to apply new work area settings because the work area ID does not exist.
8305	Internal Error	The system failed to apply a new work area setting.
8306	Error	Failed to set DO signal “doSafeStop". Solution: Verify that the signal exists and is correctly set-up.
8307	Error	Failed to connect to the controller. Solution: Verify that the network address (IP address) to the controller is correct. Verify that the network settings on the computer are correct. Verify that the correct network adapter is used (in the line) to connect to the robot controller.
8308	Error	Failed to write the IP address to the controller. Solution: Verify that the RAPID variable“ RemoteIPNode” exists and is of the correct type (should be of the string type).
8309	Error	Failed to initiate events from the robot controller. Solution: Verify that the robot controller is up and running correctly. If not, reboot the controller.
8310	Error	Failed to get the robot controller states. Solution: Ensure that the controller is up and running OK. If not, reboot the controller.

Continues on next page

Error code	Type	Description
8313	Error	Failed to set the IO signal ppaExe. Solution: Ensure that the signal ppaExe exists and is set-up correctly.
8314	Error	Failed to set the RAPID variable “RoutineName” to “NewSource”. Reason: The variable “RoutineName” is probably missing or is of the wrong type (should be a string type). Solution: Ensure that the variable exists and is of the string type.
8315	Error	The system failed to apply the new robot speed.
8316	Error	Failed to set the IO signal doTune. Solution: Ensure that the signal doTune exists and is set-up correctly.
8317	Error	The system failed to apply a new work area tune. Solution: Verify that the following RAPID variables exist. Num SourceIndex Num TunePosX Num TunePosY Num TunePosZ
8318	Error	Failed to load the RAPID program. Solution: Verify that there are no errors in the RAPID program (otherwise it will fail to load).
8319	Error	Failed to download the RAPID program to the controller.
8320	Error	Failed to stop execution of the RAPID program.
8321	Error	Failed to delete the RAPID program.
8322	Error	Failed to reset emergency stop.
8323	Error	Failed to restart the RAPID program. Solution: Stop the project and restart it.
8324	Error	Failed to get local IP address. Reason: The network set-up is not correct (e.g. wrong IP settings, faulty network adapter configuration, etc.). Solution: Solve the local network problem on the computer.

Continues on next page

Error code	Type	Description
8325	Error	Failed to init queues. Reason: PickMaster failed to initiate an item queue. The queue is initiated by setting several RAPID variables. Those variables must not be removed or changed. The variables are: String ItmSrcName String CnvName String NonCnvWobjName Num SourceType Num SourceIndex Num TunePosX Num TunePosY Num TunePosZ Num FollowTime Num Vtcp Num OffsZ Num VacActDelay Num VacRevDelay Solution: Ensure that all variables exist and are of the correct type (string or num etc.) in the RAPID program or in the PPA sys module (*ppasys.sys*).
8326	Error	Failed to synchronize the time on the robot controller with the PickMaster compute
8327	Error	There is no Rapid program defined for a robot controller when starting a project. Reason: Attempt to start a project without having configured which Rapid program to use for a robot controller. Solution: Select a Rapid program to use for the robot controller in question and restart the project.
8337	Error	Failed to flush item source queue (ItmSrcCnvxx). C0040403: No response from the controller. Reason: For large robots where working range is large, CPU takes more time for indexing it because of `GetReachableTarget` functionality. Solution: The accuracy of the release zone (indexed working range) associated with the function UseReachableTargets can be adjusted from 0% to 100% with a new process system parameter, `Reach Zone Accuracy`, in Type `Conveyor`. Default value is 100%. To make CPU load less make this value zero or very low. If the UseReachableTargets functionality is not used, it may be turned off by setting the `Reach Zone Accuracy` value to 0.
8338	Error	Not connected to controller. Reason: The communication with the controller could not be completed.
8339	Error	Unexpected error when using ABB Industrial Robot Communication Runtime to communicate with controller. Reason: See error log for more information.
8340	Error	Unexpected robot error. Reason: See error log for more information.
8341	Error	Failed to get write access to controller.

Continues on next page

Error code	Type	Description
8342	Error	Item source failed to send positions to the controller. No response from the controller.
8343	Error	The RobotWare version is later than the ABB Industrial Robot Communication Runtime on the PC. The Communication Runtime needs to be updated. Solution: If possible update PickMaster to the latest version. If this dose not solve the problem or for some reason is not possible, update the ABB Industrial Robot Communication Runtime on the PC. The installation can be downloaded from the *RobotStudio Online Community*, where it is included in the *Tools and Utilities* package.
8393	Error	The motion server already exists as an instance (only one instance is allowed).
8394	Error	The robot ID already exists (IDs shall be unique).
8395	Error	No robot defined with that ID.
8396	Error	Work areas still exist. The conveyor cannot be removed before the work areas are removed. Solution: Remove all work areas for the conveyor.
8397	Error	A work area with that ID already exists. (All IDs shall be unique).
8398	Error	No work area with that ID exists. An operation was executed on a non-existing work area. The work area has probably been removed.
8399	Error	Settings on the work area failed due to a bad work area ID.
8400	Error	The system failed to apply new work area settings due to a bad work area ID.
8401	Error	The system failed to set a new work area because the work area ID does not exist.
8402	Error	The system failed to apply a new work area tune because the work area ID does not exist.
8403	Error	The system failed to apply new robot settings because the robot ID does not exist.
8404	Error	The system failed to set new robot settings because the robot ID does not exist.
8406	Error	The system failed to set a new robot speed because the robot ID does not exist.
8407	Error	Failed to update the work area due to wrong work area type (indexed work area / conveyor work area).
8408	Warning	There are no work areas defined for the robot. Solution: Define work areas and set up position sources for the work areas for the robot before project start
8418	Status	Downloading elog files from controller. Reason: If elog files are missing at production start they will be downloaded automatically.

Continues on next page

Vision error codes

Error code	Type	Description
12298	Status	There is no frame grabber/Gigabit Ethernet camera installed
12299	Internal Error	Could not find the camera in question in the vision server.
12300	Internal Error	Could not find the vision model in question in the vision server.
12301	Internal Error	The camera is locked.
12302	Internal Error	Attempt to create or load a camera that already exists.
12305	Error	The current frame grabber does not support the selected video format.
12306	Internal Error	Failed to create camera.
12307	Internal Error	The vision server could not find the acquired camera during runtime.
12308	Warning	A camera is triggered too fast. Reason: A camera was triggered before it was done analyzing the last image. As long as there only are a few messages there will be no lost images. Solution: Adjust the vision models on the camera to yield a faster analyzing time. Adjust models on other cameras since it is the system performance in total that should be improved. Lowering the conveyor speed will also reduce the problem, if applicable.
12309	Error	Failed to get an image from a camera when running a project. Reason: This error probably occurred because the system is too heavily loaded or the frame grabber is triggered way too fast. Solution: Verify system load and make sure the robot controller does not send faulty vision triggers.
12310	Internal Error	Failed to create a geometric model. Reason: See error message for more information.
12312	Internal Error	Attempt to access a camera port on a frame grabber that does not exist.
12313	Internal Error	There is no camera port on the frame grabber specified for the camera. Solution: Open the corresponding line and configure the camera with a camera port.
12315	Error	Could not initiate the camera at project start. Reason: The system is probably out of resources.
12316	Error	External model failed to analyze image. Reason: See log message for more information
12317	Error	Failed to initiate external model at project start. Reason: See log message for more information
12318	Error	Failed to convert image to a format supported by external vision model.

Continues on next page

Error code	Type	Description
12319	Error	External model failed to inspect image. Reason: See log message for more information
12321	Error	When the line was opened, more than one camera was defined to use the same port on the same frame grabber. Only one camera can be configured to use a single camera port and hence the other cameras were reset and must be configured again.
12322	Error	When the line was opened, a camera was defined on a frame grabber that was not available. The camera was reset and must be configured again.
12323	Error	Could not initiate the camera. More information is provided in the log message.
12324	Error	Failed to save camera configuration. More information is provided in the log message
12325	Error	Failed to load camera configuration. More information is provided in the log message.
12326	Error	Failed to load vision model configuration. More information is provided in the log message
12329	Warning	Failed to communicate with Gigabit Ethernet camera. Reason: Bad Ethernet connection or excessive Ethernet communication.
12330	Warning	Images are triggered too frequently. Solution: Adjust vision models to be less time consuming, or decrease trigger frequency.
12331	Warning	Connection to camera is lost, attempting to reconnect. Reason: Ethernet cable or power cable has been disconnected.
12332	Warning	Image Buffer Full. More information is provided in the log message.
12333	Warning	A Gigabit Ethernet camera was found, but no such license was detected. Reason: No USB stick with vision license is inserted in the PC.
12334	Warning	A license for Gigabit Ethernet vision was detected, but no such camera was found. Reason: Camera is not connected, not turned on, or has an invalid IP-address.
12337	Warning	Failed to read parameter from camera. Reason: Check if the appropriate Cognex Drivers are installed. If the problem persits, check network connections.
12341	Status	Cognex USB License dongle is attached.
12342	Warning	Cognex USB License dongle is removed.

Tcp plug-in error codes

Error code	Type	Description
16485	Status	A client was connected to the Tcp plug-in.
16486	Status	A client was disconnected to the Tcp plug-in.

Continues on next page

Serial plug-in error codes

Error code	Type	Description
16585	Error	Failed to open the specified COM port. Solution: The port is probably used by another program or the selected port is not available. Open the hardware configuration dialog box and verify that correct COM port is selected.
16586	Error	Failed to initiate the open COM port with the selected configuration. Solution: Open the hardware configuration dialog box and verify that correct settings are selected for the COM port.
16634	Error	An error occurred with the serial communication. See the error message for further details.

6.6 Safety during troubleshooting

General

All normal service work; installation, maintenance and repair work, is usually performed with all electrical, pneumatic and hydraulic power switched off. All manipulator movements are usually prevented by mechanical stops etc.

Troubleshooting work differs from this. While troubleshooting, all or any power may be switched on, the manipulator movement may be controlled manually from the FlexPendant, by a locally running robot program or by a PLC to which the system may be connected.

Dangers during troubleshooting

This implies that special considerations **unconditionally** must be taken when troubleshooting:

- All electrical parts must be considered as *live*.
- The manipulator must at all times be expected to perform any movement.
- Since safety circuits may be disconnected or strapped to enable normally prohibited functions, the system must be expected to perform accordingly.

DANGER

Troubleshooting on the controller while powered on must be performed by personnel trained by ABB or by ABB field engineers.

This page is intentionally left blank

7 RIS2 API

7.1 Messages

Exception messages

Every RIS2 request returns an exception message if the request fails. The exception messages are provided in XML.

Example:

```
<?xml version="1.0" encoding="utf-16"?>
<Exception>
  <Message>Reason for exception</Message>
</Exception>
```

Status messages

Introduction

Most of the RIS2 requests returns the current status messages for the project or for the robot depending on the request and the current status of the project. The status messages are provided in XML.

Project status

The project status messages provide information about the project path and the current status of the project. The current status is provided as an integer as shown in the following table:

Integer	Status
1	Project running
2	Project stopped
3	Project configuration error
4	No license
5	Project error
6	Project opened
7	Project closed
8	Project modified

Example:

```
<?xml version="1.0" encoding="utf-16"?>
<ProjectStatusMessage>
  <ProjectPath>C:\PMProjects\Project1.pmproj</
ProjectPath>
  <Status>1</Status>
</ProjectStatusMessage>
```

Continues on next page

Robot status

The robot status messages provide information about the robot id, current status of the robot, and the path to the project where the robot exists. The current status is provided as an integer as shown in the following table:

Integer	Status
1	Robot running
2	Robot stopped
3	Robot paused
4	Robot shutdown
5	Robot emergency stop
6	Robot error
7	Robot manual mode

Example:

```
<?xml version="1.0" encoding="utf-16"?>
<RobotStatusMessage>
  <ProjectPath>C:\PMProjects\Project1.pmproj</
  ProjectPath>
    <RobotId>37e8578a-b49c-4fa9-8612-684ebd141441</RobotId>
      <Status>1</Status>
</RobotStatusMessage>
```

7.2 RIS2 requests

List Projects

Introduction

Lists the projects in PickMaster depending on the wanted status and displays the wanted project with the belonging robots. The available project statuses are **Running**, **Stopped**, **Open**, **Closed**, and **Configured**.

URL

https://pickmasterhost.se.abb.com/pickmaster/listprojects

HTTP method

`GET`

Parameters

`status`: Denotes the status of the projects to list.

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/listprojects/?status=configured

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<Projects>
  <Project>
    <ProjectPath>C:\PMProjects\Project1.pmproj</ProjectPath>
    <Robots>
    <Robot>
    <Name>Robot1</Name>
    <Guid>37e8578a-b49c-4fa9-8612-684ebd141441</Guid>
    </Robot>
    </Robots>
  </Project>
  <Project>
    <ProjectPath>C:\PMProjects\Project2.pmproj</ProjectPath>
    <Robots>
    <Robot>
    <Name>Robot1</Name>
    <Guid>37e8578a-b49c-4fa9-8612-684ebd141441</Guid>
    </Robot>
    </Robots>
  </Project>
</Projects>
```

Open Project

Introduction

Opens a project and displays the project status message.

URL

https://pickmasterhost.se.abb.com/pickmaster/openproject

Continues on next page

HTTP method

POST

Parameters

`projectpath`: Denotes the path to the project to open.

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/openproject/?projectpath=C:\PMProjects\Project1.pmproj

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<ProjectStatusMessage>
  <ProjectPath>C:\PMProjects\Project1.pmproj</
    ProjectPath>
  <Status>6</Status>
</ProjectStatusMessage>
```

Close project

Introduction

Closes a project and displays the project status message.

URL

https://pickmasterhost.se.abb.com/pickmaster/closeproject

HTTP method

POST

Parameters

`projectpath`: Denotes the path to the project to close.

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/closeproject/?projectpath=C:\PMProjects\Project1.pmproj

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<ProjectStatusMessage>
  <ProjectPath>C:\PMProjects\Project1.pmproj</
    ProjectPath>
  <Status>7</Status>
</ProjectStatusMessage>
```

Save project

Introduction

Saves a project and displays the saved project path.

URL

https://pickmasterhost.se.abb.com/pickmaster/saveproject

Continues on next page

HTTP Method

POST

Parameters

projectpath: the path to the project to save.

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/saveproject/?projectpath=C:\PMProjects\Project1.pmproj

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<SaveProject>
  <ProjectPath>C:\PMProjects\Project1.pmproj</
    ProjectPath>
</SaveProject>
```

Start project

Introduction

Starts a project and displays the project status message.

URL

https://pickmasterhost.se.abb.com/pickmaster/startproject

HTTP method

POST

Parameters

`projectpath`: Denotes the path to the project to start.

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/startproject/?projectpath=C:\PMProjects\Project1.pmproj

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<ProjectStatusMessage>
  <ProjectPath>C:\PMProjects\Project1.pmproj</
    ProjectPath>
  <Status>1</Status>
</ProjectStatusMessage>
```

Stop project

Introduction

Stops a project and displays the project status message.

URL

https://pickmasterhost.se.abb.com/pickmaster/stopproject

Continues on next page

HTTP Method

POST

Parameters

`projectpath`: Denotes the path to the project to stop.

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/stopproject/
?projectpath=C:\PMProjects\Project1.pmproj

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<RobotStatusMessage>
  <ProjectPath>C:\PMProjects\Project1.pmproj</
    ProjectPath>
    <RobotId>37e8578a-b49c-4fa9-8612-684ebd141441</RobotId>
  <Status>2</Status>
</RobotStatusMessage>
<?xml version="1.0" encoding="utf-16"?>
<ProjectStatusMessage>
  <ProjectPath>C:\PMProjects\Project1.pmproj</
    ProjectPath>
  <Status>2</Status>
</ProjectStatusMessage>
```

Project status

Introduction

Gets the current project status and displays the project status message.

URL

https://pickmasterhost.se.abb.com/pickmaster/projectstatus

HTTP Method

GET

Parameters

`projectpath`: Denotes the path to the project.

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/projectstatus/
?projectpath=C:\PMProjects\Project1.pmproj

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<ProjectStatusMessage>
  <ProjectPath>C:\PMProjects\Project1.pmproj</ProjectPath>
  <Status>2</Status>
</ProjectStatusMessage>
```

Continues on next page

Start robot

Introduction

Starts a robot and displays the robot status message or project status message depending on the current status.

URL

https://pickmasterhost.se.abb.com/pickmaster/startrobot

HTTP method

`POST`

Parameters

`projectpath`: Denotes the path to the project with robot to start.

`robotid`: Denotes the id of the robot to start.

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/startrobot/?projectpath=C:\PMProjects\Project1.pmproj&robotid=37e8578a-b49c-4fa9-8612-684ebd141441

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<RobotStatusMessage>
  <ProjectPath>C:\PMProjects\Project1.pmproj</ProjectPath>
  <RobotId>37e8578a-b49c-4fa9-8612-684ebd141441</RobotId>
  <Status>1</Status>
</RobotStatusMessage>
<?xml version="1.0" encoding="utf-16"?>
<ProjectStatusMessage>
  <ProjectPath>C:\PMProjects\Project1.pmproj</ProjectPath>
  <Status>2</Status>
</ProjectStatusMessage>
```

Pause robot

Introduction

Pauses a robot and displays the robot status message or project status message depending on the current status.

URL

https://pickmasterhost.se.abb.com/pickmaster/pauserobot

HTTP method

`POST`

Parameters

`projectpath`: Denotes the path to the project with robot to pause.

`robotid`: Denotes the id of the robot to pause.

Usage example

Request URL:

Continues on next page

https://pickmasterhost.se.abb.com/pickmaster/pauserobot/?projectpath=C:\PMProjects\Project1.pmproj&robotid=37e8578a-b49c-4fa9-8612-684ebd141441

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<RobotStatusMessage>
  <ProjectPath>C:\PMProjects\Project1.pmproj</ProjectPath>
  <RobotId>37e8578a-b49c-4fa9-8612-684ebd141441</RobotId>
  <Status>3</Status>
</RobotStatusMessage>
<?xml version="1.0" encoding="utf-16"?>
<ProjectStatusMessage>
  <ProjectPath>C:\PMProjects\Project1.pmproj</ProjectPath>
  <Status>2</Status>
</ProjectStatusMessage>
```

Stop robot

Introduction

Stops a robot and displays the robot status message or project status message depending on current status.

URL

https://pickmasterhost.se.abb.com/pickmaster/stoprobot

HTTP method

POST

Parameters

`projectpath`: Denotes the path to the project with robot to stop.

`robotid`: Denotes the id of the robot to stop.

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/stoprobot/?projectpath=C:\PMProjects\Project1.pmproj&robotid=37e8578a-b49c-4fa9-8612-684ebd141441

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<RobotStatusMessage>
  <ProjectPath>C:\PMProjects\Project1.pmproj</ProjectPath>
  <RobotId>37e8578a-b49c-4fa9-8612-684ebd141441</RobotId>
  <Status>2</Status>
</RobotStatusMessage>
<?xml version="1.0" encoding="utf-16"?>
<ProjectStatusMessage>
  <ProjectPath>C:\PMProjects\Project1.pmproj</ProjectPath>
  <Status>2</Status>
</ProjectStatusMessage>
```

Continues on next page

Reset robot emergency stop

Introduction

Resets the robot emergency stop and displays the robot status message or project status message depending on the current status.

URL

https://pickmasterhost.se.abb.com/pickmaster/resetrobotemergencystop

HTTP Method

`POST`

Parameters

`projectpath`: Denotes the path to the project with robot to reset emergency stop on.

`robotid`: Denotes the id of the robot to reset emergency stop on.

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/resetrobotemergencystop/?projectpath=C:\PMProjects\Project1.pmproj&robotid=37e8578a-b49c-4fa9-8612-684ebd141441

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<RobotStatusMessage>
  <ProjectPath>C:\PMProjects\Project1.pmproj</ProjectPath>
  <RobotId>37e8578a-b49c-4fa9-8612-684ebd141441</RobotId>
  <Status>1</Status>
</RobotStatusMessage>
<?xml version="1.0" encoding="utf-16"?>
<ProjectStatusMessage>
  <ProjectPath>C:\PMProjects\Project1.pmproj</ProjectPath>
  <Status>2</Status>
</ProjectStatusMessage>
```

Robot status

Introduction

Gets the current robot status and displays the robot status message.

URL

https://pickmasterhost.se.abb.com/pickmaster/robotstatus

HTTP Method

`GET`

Parameters

`projectpath`: Denotes the path to the project with robot.

`robotid`: Denotes the id of the robot.

Continues on next page

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/robotstatus/?projectpath=C:\PMProjects\Project1.pmproj&robotid=37e8578a-b49c-4fa9-8612-684ebd141441

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<RobotStatusMessage>
  <ProjectPath>C:\PMProjects\Project1.pmproj</ProjectPath>
  <RobotId>37e8578a-b49c-4fa9-8612-684ebd141441</RobotId>
  <Status>2</Status>
</RobotStatusMessage>
```

List tuning parameters

Introduction

Lists the tuning parameters in a chosen project and displays the name and type of parent nodes with its children. The children have their own path with which to access them.

URL

https://pickmasterhost.se.abb.com/pickmaster/listtuningparameters

HTTP method

`GET`

Parameters

`projectpath,`: Denotes the path to the project with tuning parameters to list.

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/listtuningparameters/?projectpath=C:\PMProjects\Project1.pmproj

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<TuningParameters>
  <Project>C:\PMProjects\Project1.pmproj</ Project>
    <Node>
      <Name>Robot1</Name>
      <Type>Robot</Type>
      <Parameters>
        <Parameter>
        <Path>/RBT:{37E8578A-B49C-4FA9-8612-684EBD141441}/Speed</Path>
        <DataType>Integer</DataType>
        <Limits>
        <Minimum>0</Minimum>
        <Maximum>0</Maximum>
        </Limits>
        </Parameter>
        <Parameter>
        <Path>/RBT:{37E8578A-B49C-4FA9-8612-684EBD141441}/PickRate</Path>
```

Continues on next page

```
        <DataType>Integer</DataType>
        <Limits>
        <Minimum>0</Minimum>
        <Maximum>0</Maximum>
        </Limits>
        </Parameter>
      </Parameters>
    </Node>
    <Node>
      <Name>Item1</Name>
      <Type>Item</Type>
      <Parameters>
        <Parameter>
        <Path>/ITEM:{785A586B-BB09-4D99-9955-9CF1D3B5B0CF}/GripX</Path>
        <DataType>Float</DataType>
        <Limits>
        <Minimum>0</Minimum>
        <Maximum>0</Maximum>
        </Limits>
        </Parameter>
        <Parameter>
        <Path>/ITEM:{785A586B-BB09-4D99-9955-9CF1D3B5B0CF}/GripZ</Path>
        <DataType>Float</DataType>
        <Limits>
        <Minimum>0</Minimum>
        <Maximum>0</Maximum>
        </Limits>
        </Parameter>
      </Parameters>
    </Node>
</TuningParameters>
```

Read tuning parameters

Introduction

Reads a tuning parameter and displays the current value.

URL

https://pickmasterhost.se.abb.com/pickmaster/readtuningparameter

HTTP method

`GET`

Parameters

`projectpath`: Denotes the path to the project with the tuning parameters to read.

`parampath`: Denotes the path to the parameter to read.

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/readtuningparameter/?projectpath=C:\PMProjects\Project1.pmproj&parampath=/RBT:{37E8578A-B49C-4FA9-8612-684EBD141441}/Speed

Continues on next page

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<TuningParameter>
  <Value>5000</Value>
</TuningParameter>
```

Write tuning parameters

Introduction

Writes a tuning parameter and displays a set value.

URL

https://pickmasterhost.se.abb.com/pickmaster/writetuningparameter

HTTP Method

`POST`

Parameters

`projectpath`: Denotes the path to the project with the tuning parameters to write.

`parampath`: Denotes the path to the parameter to write.

`value`: Denotes the value to write.

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/writetuningparameter/?projectpath=C:\PMProjects\Project1.pmproj&parampath=/RBT:{37E8578A-B49C-4FA9-8612-684EBD141441}/Speed&value=5000

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<TuningParameter>
  <Value>4500</Value>
</TuningParameter>
```

List service variables

Introduction

Lists service variables in a chosen project and displays the variable names and current value.

URL

https://pickmasterhost.se.abb.com/pickmaster/listservicevariables

HTTP Method

`GET`

Parameters

`projectpath`: Denotes the path to the project with robot to list service variables on.

`robotid`: Denotes the id of the robot to list service variables on.

Continues on next page

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/listservicevariables/?projectpath=C:\PMProjects\Project1.pmproj&robotid=37e8578a-b49c-4fa9-8612-684ebd141441

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<ServiceVariables>
  <ServiceVariable>
    <Name>ServVar1</Name>
    <Value>0</Value>
  </ServiceVariable>
  <ServiceVariable>
    <Name>ServVar2</Name>
    <Value>0</Value>
    <ServiceVariable>
      <Name>ServVar3</Name>
      <Value>0</Value>
    </ServiceVariable>
  </ServiceVariable>
</ServiceVariables>
```

Write service variables

Introduction

Writes to a service variable and displays the service variable name, set value, project path, and robot id.

URL

https://pickmasterhost.se.abb.com/pickmaster/writeservicevariable

HTTP Method

POST

Parameters

`projectpath`: Denotes the path to the project with robot to write service variables on.

`robotid`: Denotes the id of the robot to write service variables on.

`servicevariable`: Denotes the name of service variable.

`value`: Denotes the value to write

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/writeservicevariable/?projectpath=C:\PMProjects\Project1.pmproj&robotid=37e8578a-b49c-4fa9-8612-684ebd141441&servicevariable=ServVar1&value=1

Response:

```
<?xml version="1.0" encoding="utf-16"?>
```

Continues on next page

```
<ServiceVariableValue>
  <ServiceVariable>
    <Name>ServVar1</Name>
    <Value>1</Value>
    ProjectPath>C:\PMProjects\Project1.pmproj</ ProjectPath>
    <RobotId>37e8578a-b49c-4fa9-8612- 684ebd141441</RobotId>
  <ServiceVariable>
</ServiceVariableValue>
```

List service routines

Introduction

Lists the service routines in chosen project and displays the routine names.

URL

https://pickmasterhost.se.abb.com/pickmaster/listserviceroutines

HTTP Method

`GET`

Parameters

`projectpat`: Denotes the path to the project with robot to list service routines on.

`robotid`: Denotes the id of the robot to list service routines on.

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/listserviceroutines/?projectpath=C:\PMProjects\Project1.pmproj&robotid=37e8578a-b49c-4fa9-8612-684ebd1414411

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<ServiceRoutines>
  <ServiceRoutine>
    <Name>TestCycle</Name>
  </ServiceRoutine>
  <ServiceRoutine>
    <Name>Home</Name>
  </ServiceRoutine>
  <ServiceRoutine>
    <Name>WashDown</Name>
  </ServiceRoutine>
  <ServiceRoutine>
    <Name>Homepos</Name>
  </ServiceRoutine>
</ServiceRoutines>
```

Invoke service variables

Introduction

Invokes a service routine and displays the service routine name, project path, and robot id.

Continues on next page

URL

https://pickmasterhost.se.abb.com/pickmaster/invokeserviceroutine

HTTP Method

`POST`

Parameters

`projectpath`: Denotes the path to the project with robot to invoke service routine on.

`robotid`: Denotes the id of the robot to invoke service routines on.

`serviceroutine`: Denotes the name of service routine.

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/invokeserviceroutine/?projectpath=C:\PMProjects\Project1.pmproj&robotid=37e8578a-b49c-4fa9-8612-684ebd141441&serviceroutine=TestCycle

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<ServiceRoutine>
  <Name>testcycle</Name>
  <ProjectPath>C:\PMProjects\Project1.pmproj</ProjectPath>
  <RobotId>37e8578a-b49c-4fa9-8612-684ebd141441</RobotId>
</ServiceRoutine>
```

</ServiceVariableValue>

List position sources

Introduction

Lists the position sources in a chosen project and displays the position source names and ids.

URL

https://pickmasterhost.se.abb.com/pickmaster/listpositionsources

HTTP Method

`GET`

Parameters

`projectpath`: Denotes the path to the project to list position sources on.

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/listpositionsources/?projectpath=C:\PMProjects\Project1.pmproj

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<PositionSources>
  <PositionSource>
    <Name>Position Source1</Name>
```

Continues on next page

```
      <Guid>855501ed-3fb0-418c-9316-7880e94f8024</Guid>
    </PositionSource>
  </PositionSources>
```

Detailed vision

Introduction

Launches the detailed vision information dialog for the chosen position source. The dialog is launched in the PickMaster application. Displays the project path and position source id.

URL

https://pickmasterhost.se.abb.com/pickmaster/detailedvision

HTTP Method

`POST`

Parameters

`projectpath`: Denotes the path to the project with position source to launch detailed vision on.

`possrcid`: Denotes the id of position source to launch detailed vision on.

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/detailedvision/?projectpath=C:\PMProjects\Project1.pmproj&posSrcId=855501ed-3fb0-418c-9316-7880e94f8024

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<DetailedVision>
  <ProjectPath>C:\PMProjects\Project1.pmproj</ProjectPath>
  <PositionSourceId>855501ed-3fb0-418c-9316-7880e94f8024</PositionSourceId>
</DetailedVision>
```

Log messages

Introduction

Collects upto 100 latest log messages from PickMaster. Uses long polling, that is, the requests waits for new messages from PickMaster for upto 30 seconds before returning. Preferably used in its own thread. Displays the log messages with the message, origin project, time stamp, status, and type.

URL

https://pickmasterhost.se.abb.com/pickmaster/logmessages

HTTP Method

`GET`

Parameters

None

Continues on next page

Usage example

Request URL:

https://pickmasterhost.se.abb.com/pickmaster/logmessages/

Response:

```
<?xml version="1.0" encoding="utf-16"?>
<LogMessages>
  <LogMessage>
    <Message>[Controller1] -- Entering state Connected </Message>
    <ProjectName>Project1</ProjectName>
    <Time>11/03/10 15:40:11</Time>
    <Status>0</Status>
    <Type>Status</Type>
  </LogMessage>
  <LogMessage>
    <Message>[Robot1] Controller state: MotorsOn</Message>
    <ProjectName>Project1</ProjectName>
    <Time>11/03/10 15:40:11</Time>
    <Status>8194</Status>
    <Type>Status</Type>
  </LogMessage>
  <LogMessage>
    <Message>[Controller1] -- Entering state MotorsOn </Message>
    <ProjectName>Project1</ProjectName>
    <Time>11/03/10 15:40:11</Time>
    <Status>0</Status>
    <Type>Status</Type>
  </LogMessage>
  <LogMessage>
    <Message>[Robot1] -- T_ROB1 entering state InitRapid </Message>
    <ProjectName>Project1</ProjectName>
    <Time>11/03/10 15:40:11</Time>
    <Status>0</Status>
    <Type>Status</Type>
  </LogMessage>
</LogMessages>
```

This page is intentionally left blank

8 RAPID reference

8.1 Instructions

8.1.1 AckItmTgt - Acknowledge an item target

Usage

`AckItmTgt` is used to acknowledge that an `itmtgt` received with `GetItmTgt` from an item source has been used (For example, handled by the robot, skipped or put back in the queue for later usage). Normally, acknowledge is setup as a `TriggL` event on the path (using the `Ack` or `Nack` triggdata from sourcedata) to make sure acknowledge does not occur before any movements related to the target has been finished. However, if the received `itmtgt` shall be skipped or put back in the queue for later usage, movements related to the target may not be needed. Then it is convenient to use this instruction instead. Only after the acknowledge has been made, a new `itmtgt` can be fetched from the item source.

Basic example

```
VAR itmtgt PlaceTarget;
GetItmTgt ItmSrcData{Index}.ItemSource, PlaceItem;
AckItmTgt ItmSrcData{Index}.ItemSource, PlaceItem, FALSE
     \Skip:=TRUE;
```

Arguments

```
AckItmTgt ItemSource ItemTarget Acknowledge [\Skip] [\Type]
```

`ItemSource`

Data type: `itmsrc`

The item source from where the item target has been received with `GetItmTgt`.

`ItemTarget`

Data type: `itmtgt`

The item target to acknowledge.

`Acknowledge`

Data type: `bool`

The status of acknowledge. TRUE if the `itmtgt` has been handled (picked or placed)by the robot and FALSE otherwise, in which case the `itmtgt` is put back into the queue.

`Skip`

Data type: `bool`

Indicates if the `itmtgt` shall be skipped. If set to TRUE it will not be possible to receive the `itmtgt` again with `GetItmTgt`. If combined with `Acknowledge` = FALSE the `itmtgt` will be passed on for possible handling by downstream robots. If combined with `Acknowledge` = TRUE, skip will have no effect. If Skip is set to FALSE the `itmtgt` will either be considered as handled by the robot (when

Continues on next page

8.1.1 AckItmTgt - Acknowledge an item target
Continued

combined with `Acknowledge` = TRUE), or put back in the queue for later usage (when combined with `Acknowledge` = FALSE).

`Type`

Data type: `num`

Modifies the type of the `itmtgt`. If combined with `Acknowledge` = FALSE and `Skip` = TRUE, the item will be passed on to downstream robots according to the configured distribution of the new item type.

If combined with `Acknowledge` = FALSE and `Skip` = FALSE, the item will be put back in the queue with the new item type and can still be received with `GetItmTgt`. The item type will only be changed locally; the item type and the distribution of the item will not change for downstream robots.

If combined with `Acknowledge` = TRUE, type change will have no effect.

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:

Error code	Description
`ERR_ITMSRC_UNDEF`	`itmsrc` undefined.

Limitations

The `itmtgt` must be received with the instruction `GetItmTgt`.

Syntax

```
AckItmTgt
  [ItemSource ':=' ] <variable (VAR) of itmsrc>,
  [ItemTarget ':=' ] <var or pers (INOUT) of itmtgt>,
  [Acknowledge':=' ] <expression (IN) of bool>,
  [\Skip ':=' ] <expression (IN) of bool>,
  [\Type ':=' ] <expression (IN) of num>;
```

Related information

For information about	See
The data type `itmtgt`	*itmtgt - Item target data on page 274*.

8.1.2 FlushItmSrc - Flush an item source

Usage

`FlushItmSrc` is used to flush an item source. The instruction clears the item source buffers, sets the scene number to one and flushes the encoder board.

Basic example

```
FlushItmSrc PlaceSource;
```

Flushes the earlier created item source object *PlaceSource*.

Arguments

```
FlushItmSrc ItemSource
```

`ItemSource`

Data type: `itmsrc`

The created item source.

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:

Error code	Description
`ERR_ITMSRC_UNDEF`	`itmsrc` undefined

Limitations

To avoid potential problems, this instruction should be executed only when the last item target definitely has been acknowledged.

Syntax

```
FlushItmSrc
  [ItemSource ':=' ] <variable (VAR) of itmsrc>;
```

8.1.3 GetItmTgt - Get the next item target

Usage

`GetItmTgt` is used to get the next available `itmtgt` in the item source queue between the enter and the exit limit of the work area. The RAPID program waits in this instruction until the next item is possible to reach or the timeout occurs.

Basic examples

Basic examples of the instruction `GetItmTgt` are illustrated below.

Example 1

```
GetItmTgt PlaceSource, PlaceItem;
```

Receives a place item from the PlaceSource when there is one that can be used.

Example 2

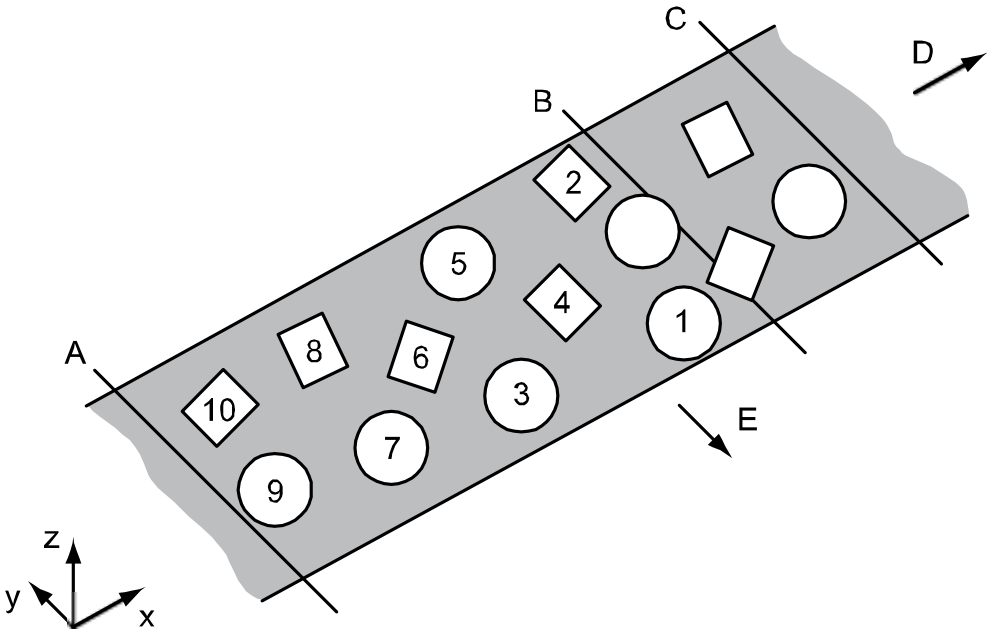
```
...
VAR selectiondata neg_y_sort;

neg_y_sort.ShapeType:=BOX;
neg_y_sort.ConsiderType:=BitOr(ITEMS_TO_USE,ITEMS_BYPASS);
neg_y_sort.GeometricData.x:=60;
neg_y_sort.GeometricData.y:=500;
neg_y_sort.GeometricData.z:=10;
neg_y_sort.GeometricData.radius:=0;
neg_y_sort.Offset.OffsetRelation:=FRAME_COORD_DIR;
neg_y_sort.Offset.OffsetPose.trans.x:=0;
neg_y_sort.Offset.OffsetPose.trans.y:=-500;
neg_y_sort.Offset.OffsetPose.trans.z:=0;
neg_y_sort.Offset.OffsetPose.rot.q1:=1;
neg_y_sort.Offset.OffsetPose.rot.q2:=0;
neg_y_sort.Offset.OffsetPose.rot.q3:=0;
neg_y_sort.Offset.OffsetPose.rot.q4:=0;
IF pick_type = 2 THEN pick_type := 1; ELSE
  pick_type := 2
ENDIF

GetItmTgt PickSource, PickItem \ItemType:=pick_type \Limit:=100
     \Selection:=neg_y_sort;
```

Retrieves a pick item from the *PickSource* with negative y-sorting and type request. The type is alternating between two types. The `Limit` argument tells from where to start the search.

In the example graphic below, the sorting is in positive x-direction, negative y-direction, and operating on two different object types. The two object types should

Continues on next page

be chosen in an alternating pattern starting with the circular. This will give the order as numbered 1-10 in the graphic.



xx0900000451

A	Enter
B	Check limit
C	Exit
D	Product flow direction
E	Sort direction
1-10	Sort order

Arguments

```
GetItmTgt ItemSource, ItemTarget [\MaxTime] [\TimeFlag] [\ItemType]
     [\Limit] [\SortData] [\Selection] [\Val1Min] [\Val1Max]
     [\Val2Min] [\Val2Max] [\Val3Min] [\Val3Max] [\Val4Min]
     [\Val4Max] [\Val5Min] [\Val5Max]
```

`ItemSource`

Data type: `itmsrc`

The item source from which the item target should be received.

`ItemTarget`

Data type: `itmtgt`

The received item target.

`[\MaxTime]`

Data type: `num`

The maximum waiting time permitted, expressed in seconds. If this time runs out before the item target is retrieved and no `TimeOut` flag is given, the error handler

Continues on next page

8.1.3 GetItmTgt - Get the next item target
Continued

will be called with the error code ERR_PPA_TIMEOUT. If there is no error handler, the execution will be stopped.

[\TimeFlag]

Data type: bool

The output parameter that contains the value TRUE if the maximum permitted waiting time runs out before an item target is received. If this parameter is included in the instruction, it is not considered to be an error if the max time runs out. This argument is ignored if the MaxTime argument is not included in the instruction.

[\ItemType]

Data type: num

Specifies which item type number is requested. The instruction waits until an item target with the requested type number is available to be executed.

[\Limit]

Data type: num

Modifies the distance from where the item target is received. The instruction will return the next item target above this limit. If this argument is excluded, the instruction will return the next item target above the exit limit.

The distance is specified in millimeters from the center of the robot. The value is positive if the limit is beyond the center of the robot, in the moving direction of the feeder. This argument is only valid when a conveyor is used.

[\SortData]

Data type: sortdata

This data structure defines how the items shall be sorted.

[\Selection]

Data type: selectiondata

This data structure defines how the items are selected.

[\Val1Min]

Data type: num

Specifies minimum value for itmtgt parameter Val1. The instruction waits until an item target fulfilling this condition is available for execution.

[\Val1Max]

Data type: num

Specifies maximum value for itmtgt parameter Val1. The instruction waits until an item target fulfilling this condition is available for execution.

[\Val2Min]

Data type: num

Specifies minimum value for itmtgt parameter Val2. The instruction waits until an item target fulfilling this condition is available for execution.

[\Val2Max]

Data type: num

Continues on next page

Specifies maximum value for `itmtgt` parameter Val2. The instruction waits until an item target fulfilling this condition is available for execution.

`[\Val3Min]`

Data type: `num`

Specifies minimum value for `itmtgt` parameter Val3. The instruction waits until an item target fulfilling this condition is available for execution.

`[\Val3Max]`

Data type: `num`

Specifies maximum value for `itmtgt` parameter Val3. The instruction waits until an item target fulfilling this condition is available for execution.

`[\Val4Min]`

Data type: `num`

Specifies minimum value for `itmtgt` parameter Val4. The instruction waits until an item target fulfilling this condition is available for execution.

`[\Val4Max]`

Data type: `num`

Specifies maximum value for `itmtgt` parameter Val4. The instruction waits until an item target fulfilling this condition is available for execution.

`[\Val5Min]`

Data type: `num`

Specifies minimum value for `itmtgt` parameter Val5. The instruction waits until an item target fulfilling this condition is available for execution.

`[\Val5Max]`

Data type: `num`

Specifies maximum value for `itmtgt` parameter Val5. The instruction waits until an item target fulfilling this condition is available for execution.

Program execution

If there is no item target in buffer or any item targets available in the working area, the program execution waits in this instruction until an item is considered as inside the working area.

If the `MaxTime` argument is specified then the wait time is supervised. If the waiting time exceeds the value of `MaxTime` and the `TimeFlag` argument is used, then the program will continue. If `TimeFlag` is not used, then an error is raised. If `TimeFlag` is specified, it will be set to TRUE if the time is exceeded, otherwise it will be set to `FALSE`.

The `Limit` argument modifies the limit from where the item target shall be received.

If the `SortData` argument is specified the instruction will return the item target that is the closest to the exit limit in x-direction and depending of the absence of other objects in direction of the sorting, the first object in the sort direction will be selected. The `CheckBoundry` distance defines the required clearance distance

Continues on next page

around an object. The sorting will check both upwards and downwards the production flow for presence of other item targets. If this argument is combined with the `Limit` argument the sorting algorithm will also take all objects between the limit and the exit limit into consideration when checking the safety distance for the nearest objects. If more than one robot is used in a shared position source system, that is load balancing or `ATC`, we strongly recommend using the `Selection` argument instead with a proper selection data, as `SortData` does not take items that are bypassing in consideration when sorting.

If the `Selection` argument is specified, the instruction will return the item target that is the closest to the exit limit in x-direction, which has no other item targets inside the specified shape. If this argument is combined with the `Limit` argument the selection algorithm will also take all objects between the limit and the exit limit into consideration when checking the distance for the nearest objects. This is highly recommended to avoid collisions.

If values are specified for the optional arguments `Va1Xmin` or `Va1Xmax`, the instruction will return an item target that fulfills the required maximum and minimum values for `Va1X`.

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:

Error code	Description
`ERR_ITMSRC_UNDEF`	`itmsrc` undefined.
`ERR_PPA_TIMEOUT`	Timeout without any error flag.

Syntax

```
GetItmTgt
  [ItemSource ':=' ] <variable (VAR) of itmsrc>,
  [ItemTarget ':=' ] <var or pers (INOUT) of itmtgt>
  [\MaxTime ':=' ] <expression (IN) of num>
  [\TimeFlag ':=' ] <var or pers (INOUT) of bool>
  [\ItemType ':=' ] <expression (IN) of num>
  [\Limit ':=' ] <expression (IN) of num>
  [\SortData ':=' ] <expression (IN) of sortdata>
  [\Selection ':=' ] <expression (IN) of selectiondata>
  [\Val1Min ':=' ] <expression (IN) of num>
  [\Val1Max ':=' ] <expression (IN) of num>
  [\Val2Min ':=' ] <expression (IN) of num>
  [\Val2Max ':=' ] <expression (IN) of num>
  [\Val3Min ':=' ] <expression (IN) of num>
  [\Val3Max ':=' ] <expression (IN) of num>
  [\Val4Min ':=' ] <expression (IN) of num>
  [\Val4Max ':=' ] <expression (IN) of num>
  [\Val5Min ':=' ] <expression (IN) of num>
  [\Val5Max ':=' ] <expression (IN) of num>;
```

Continues on next page

Related information

For information about	See
The data type `itmtgt`	*itmtgt - Item target data on page 274*.
The data type `selectiondata`	*selectiondata - Selection data on page 277*.
The data type `sortdata`	*sortdata - Sort data on page 280*.

8.1.4 NextItmTgtType - Get the type of the next item target

Usage

`NextItmTgtType` is used to get the type of the next item target (`itmtgt`) in the item source buffer. If the `Limit` distance parameter is given, the instruction will return the type of the next item target above the limit. The RAPID program waits in this instruction until there is an item in this queue.

Basic examples

```
NextItmTgtType PlaceSource, PlaceType
```

Retrieves the type of the next `itmtgt` in the *PlaceSource*.

Arguments

```
NextItmTgtType ItemSource ItemType [\Limit] [\MaxTime] [\TimeFlag]
```

`ItemSource`

Data type: `itmsrc`

The item source that the item target type should be retrieved from.

`ItemType`

Data type: `num`

The retrieved item target type.

`[\Limit]`

Data type: `num`

This is the limit from where the type is retrieved. The instruction will return the type of the next item target above this limit. If this argument is excluded, the instruction will return the type of the next item target above the exit limit.

The distance is calculated in millimeters from the center of the robot. The value is positive if the limit is beyond the center of the robot, in the moving direction of the conveyor.

This argument is only valid when a conveyor is used.

`[\MaxTime]`

Data type: `num`

The maximum waiting time permitted, expressed in seconds. If this time runs out before the item target is retrieved and no `TimeOut` flag is given, the error handler will be called with the error code `ERR_PPA_TIMEOUT`. If there is no error handler, the execution is stopped.

`[\TimeFlag]`

Data type: `bool`

The output parameter that contains the value `TRUE` if the maximum permitted waiting time runs out before an item target is retrieved. If this parameter is included in the instruction it is not considered to be an error if the max time runs out.

This argument is only used if the `MaxTime` argument is used.

Continues on next page

Program execution

If there is no item target in buffer or any item targets above the `Limit`, the program execution waits in this instruction until there is an item in the buffer.

If the `MaxTime` argument is specified then the wait time is supervised. If the waiting time exceeds the value of `MaxTime` and the `TimeFlag` argument is used, then the program will continue. If `TimeFlag` is not used, then an error is raised. If `TimeFlag` is specified, this will be set to TRUE if the time is exceeded, otherwise it will be set to FALSE.

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:

Error code	Description
`ERR_ITMSRC_UNDEF`	`itmsrc` undefined.
`ERR_PPA_TIMEOUT`	Timeout without any error flag

Syntax

```
NextItmTgtType
  [ItemSource ':=' ] <variable (VAR) of itmsrc>,
  [ItemType ':=' ] <var or pers (INOUT) of num>
  [\Limit ':=' ] <expression (IN) of num>
  [\MaxTime ':=' ] <expression (IN) of num>
  [\TimeFlag ':=' ] <var or pers (INOUT) of bool>;
```

Related information

For information about	See
The data type `itmtgt`	*itmtgt - Item target data on page 274*.

8.1.5 QStartItmSrc - Start queue in item source

Usage

`QStartItmSrc` is used to start the queue in an item source. This instruction must be used when starting a new program or after flushing.

Basic example

```
QStartItmSrc PlaceSource;
```

The queue of objects in the item source *PlaceSource* is started.

Arguments

```
QStartItmSrc ItemSource
```

`ItemSource`

Data type: `itmsrc`

The started item source.

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:

Error code	Description
`ERR_ITMSRC_UNDEF`	`itmsrc` undefined

Syntax

```
QStartItmSrc
  [ItemSource ':=' ] <variable (VAR) of itmsrc>;
```

Related information

For information about	See
The instruction `QStopItmSrc`	*QStopItmSrc - Stop queue in item source on page 263*.

8.1.6 QStopItmSrc - Stop queue in item source

Usage

`QStopItmSrc` is used to stop the queue in an item source.

Basic example

```
QStopItmSrc PlaceSource;
```

The queue of objects in the item source *PlaceSource* is stopped.

Arguments

```
QStopItmSrc ItemSource
```

`ItemSource`

Data type: `itmsrc`

The stopped item source.

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:

Error code	Description
`ERR_ITMSRC_UNDEF`	`itmsrc` undefined

Syntax

```
QStopItmSrc
  [ItemSource ':=' ] <variable (VAR) of itmsrc>;
```

Related information

For information about	See
The instruction `QStartItmSrc`	*QStartItmSrc - Start queue in item source on page 262*.

8.1.7 ResetFlowCount - Reset flow counter

Usage

`ResetFlowCount` is used to reset the flow counter. The flow counter indicates the number of objects that has passed the exit limit of a conveyor work area since last reset. The value of the flow counter can be retreived with the function `GetFlowCount`

Basic example

```
ResetFlowCount PlaceSource;
```

Resets the flow counter for an item source.

Arguments

```
ResetFlowCount ItemSource
```

`ItemSource`

Data type: `itmsrc`

The item source.

Error handling

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

Error code	Description
`ERR_ITMSRC_UNDEF`	`itmsrc` undefined

Syntax

```
ResetFlowCount[ItemSource ':=' ] <variable (VAR) of itmsrc>;
```

Related information

For information about	See
The function `GetFlowCount`	*GetFlowCount - Get number of passed items on page 273*.

8.1.8 ResetMaxUsageTime - Reset max measured usage time

Description

ResetMaxUsageTime is used to reset the maximum measured usage time of the previously handled objects. This is the time between receiving a target with *GetItmTgt*, until the object is handled by the robot (acknowledge time). *ResetMaxUsageTime* is only available with the PickMaster 3 option.

Example

```
ResetMaxUsageTime ItmSrcData{PickWorkArea{1}}.ItemSource;
```

Resets the maximum usage time for an item source.

Arguments

```
ResetMaxUsageTime ItemSource
```

`Item Source`

ItemSource

Data type: `itmsrc`

The item source.

Error handling

The following recoverable errors are generated. They are handled in an error handler. The system variable `ERRNO` will be set to:

`ERR_ITMSRC_UNDEF`	The itmsrc is undefined.

Syntax

```
ResetMaxUsageTime[ItemSource ':=' ] <variable (VAR) of itmsrc>;
```

8.1.9 UseReachableTargets - Use reachable targets

Description

UseReachableTargets is used to activate a functional mode, where the robot only receives reachable targets for object handling.

When activated, non-reachable targets are filtered out for target requests with *GetItmTgt*, see *Application manual - PickMaster® 3*.

UseReachableTargets sets an optimal target release zone with a variable size. The size of the release zone depends on the robot's reach and the real-time speed of the conveyor. When the conveyor speed increases, the size of the release zone decreases, thereby decreasing the amount of targets available for use. If the conveyor speed is too high, the release zone disappears completely and no targets will be received until the speed is reduced.

UseReachableTargets is available only with the PickMaster 3 option.

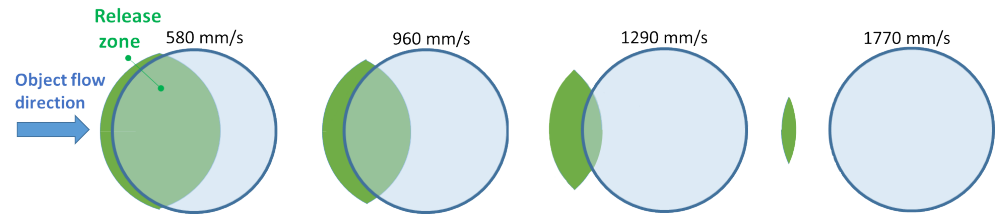
WARNING

The target release zone depends on the selection of the enter/exit limits, see *Application manual - PickMaster® 3*. The resulting target release zone will be the intersection of the optimal target release zone and the enter/exit region.

The recommended exit/enter values to avoid any impact on the optimal target zone are as follows:

- Enter = -10000 mm (this signifies, a distance well outside the robot reach in an upstream direction)
- Exit = 10000 mm (this signifies, a distance well outside the robot reach in a downstream direction)

The following figure shows the target release zone for an IRB 360 (as seen from above) at 4 different conveyor speeds. The light blue area is the working range of the robot and the green area is the target release zone.



xx1800000383

Example

```
UseReachableTargets ItmSrcData{PlaceWorkArea{1}}.ItemSource, TRUE,
     0.7 \ReleaseTime:=0.1;
GetItmTgt PlaceSource, PlaceItem;
```

Activate *UseReachableTargets* in the place work area of a linear conveyor. The targets in use are expected to be placed within a maximum time of 0.7 seconds after being received with *GetItmTgt*. Targets become available for use 0.1 second

Continues on next page

before they enter robot reach. Then, the targets remain available for use until they leave the release zone.

Arguments

```
UseReachableTargets ItemSource, Enable, UsageTime [\ReleaseTime]
```

`Item Source`

ItemSource

Data type: `itmsrc`

The item source where *UseReachableTargets* is activated.

`Enable`

Enable

Data type: `bool`

This activates/deactivates *UseReachableTargets*.

`Usage Time`

UsageTime

Data type: `num`

The expected usage time of the targets. This is the time between receiving the target with *GetItmTgt*, until the object is handled (for example picked) by the robot (acknowledge time). The actual usage time is continuously measured and the maximum measured usage time can be received with *GetMaxUsageTime*. To avoid reach errors, the *UsageTime* value should be defined as a sum of the maximum measured usage time and a margin. For example, set UsageTime = Maximum measured usage time + 0.1 second. The drawback of having a large safety margin is an unnecessary reduction of the target release zone, which may decrease the pick rate.

`[\ReleaseTime]`

Release Time

Data type: `num`

The *ReleaseTime* defines the time when the targets enter the release zone, before entering robot reach. If the value is negative, targets enter the release zone after they enter robot reach. A value of 0.1 or less is recommended to avoid reach errors. A higher value can be useful to handle high speed conveyors. The drawback of a higher value is an increasing risk of having upstream reach errors at low speeds.

Note

It is possible to change *UsageTime* or *ReleaseTime* at any time. For example, a temporary reduction in the robot speed requires a longer usage time to avoid reach errors.

Syntax

```
UseReachableTargets
  [ItemSource ':=' ] <variable (VAR) of itmsrc>,
  [Enable ':=' ] <var or pers (IN) of bool>
```

Continues on next page

8.1.9 UseReachableTargets - Use reachable targets

RobotWare - OS

Continued

```
[UsageTime ':=' ] <var or pers (IN) of num>
[\ReleaseTime ':=' ] <expression (IN) of num>;
```

Limitations

If the robot work area is limited in motion configuration, there is a possibility that targets upto 20 mm outside of the working area perpendicular to the conveyor moving direction, may be retrieved by the `GetItmTgt` instruction.

A work around to avoid the outside reach errors is to put an extra check on the Y-value of the `itemtarget` before moving towards it.

8.2 Functions

8.2.1 GetMaxUsageTime - Get max measured usage time

Description

GetMaxUsageTime is used to get the maximum measured usage time of the previously handled objects. It is the time between receiving a target with *GetItmTgt*, until the object is handled by the robot (acknowledge time). The actual usage time is continuously measured for each handled object. *GetMaxUsageTime* is only available with the PickMaster 3 option.

Example

```
VAR num usetime;
usetime := GetMaxUsageTime(ItmSrcData{PickWorkArea{1}}.ItemSource);
```

usetime is the the maximum measured usage time since starting production or since executing *ResetMaxUsageTime*.

Return value

Data type: `num`

The maximum measured usage time since starting production or since executing *ResetMaxUsageTime*.

Arguments

```
GetMaxUsageTime (ItemSource)
```

`Item Source`

ItemSource

Data type: `itmsrc`

The item source.

Error handling

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

`ERR_ITMSRC_UNDEF`	The *itmsrc* is undefined.

Syntax

```
GetMaxUsageTime '('[ItemSource ':=' ] <variable (VAR) of itmsrc>
    ')';
```

This function returns the value of the data in num type.

8.2.2 GetQueueLevel - Get queue level

Usage

`GetQueueLevel` is used to get current number of item targets in an item source fulfilling certain conditions.

Basic example

```
reg1 := GetQueueLevel(PlaceSource);
```

`reg1` is assigned the current number of item targets in the item source `PlaceSource`.

Return value

Data type: `num`

The current number of item targets in the item source.

Arguments

```
GetQueueLevel (ItemSource [\ItmType] [\MinLimit] [\MaxLimit])
```

`ItemSource`

Data type: `itmsrc`

The item source that the current number of item targets should be retrieved from.

`\ItmType`

Data type: `num`

Only items of the specified type number will be counted.

`\MinLimit`

Data type: `num`

Defines the minimum distance to the robot center from where an item will be counted. A negative value indicates that the limit is upstreams from the robot center. A positive value indicates that the limit is downstreams. The parameters does not affect indexed work areas.

`\MaxLimit`

Data type: `num`

Defines the maximum distance to the robot center from where an item will be counted. A negative value indicates that the limit is upstreams from the robot center. A positive value indicates that the limit is downstreams. The parameter does not affect indexed work areas.

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:

Error code	Description
`ERR_ITMSRC_UNDEF`	`itmsrc` undefined

Continues on next page

Syntax

```
GetQueueLevel '('
  [ItemSource ':=' ] <variable (VAR) of itmsrc> ')'
  [\ItmType ':=' ] <expression (IN) of num>
  [\MinLimit ':=' ] <expression (IN) of num>
  [\MaxLimit ':=' ] <expression (IN) of num>;
```

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

8.2.3 GetQueueTopLevel - Get queue top level

Usage

`GetQueueTopLevel` is used to get the maximum number of item targets that simultaneously have been in the buffer of an item source.

Basic examples

```
reg1 := GetQueueTopLevel(PlaceSource);
```

reg1 is assigned the maximum number of item targets that simultaneously have been in the item source *PlaceSource*.

Return value

Data type: `num`

The maximum number of item targets that simultaneously have been in the item source.

Arguments

```
GetQueueTopLevel (ItemSource)
```

`ItemSource`

Data type: `itmsrc`

The item source that the current number of item targets should be retrieved from.

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:

Error code	Description
`ERR_ITMSRC_UNDEF`	`itmsrc` undefined

Syntax

```
GetQueueTopLevel '('
  [ItemSource ':=' ] <variable (VAR) of itmsrc> ')';
```

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

8.2.4 GetFlowCount - Get number of passed items

Usage

`GetFlowCount` is used to get the total number of items that has passed the exit limit of a conveyor work area since `ResetFlowCount` was executed. Items that the robot handles will not be counted (even if they pass the exit limit before picking/placing occurs).

Basic example

```
VAR num counter;
ResetFlowcount PlaceSource;
WaitTime 10;
counter := GetFlowCount(PlaceSource);
```

counter is assigned the number of items originating from `PlaceSource` that has passed the exit limit.

Return value

Data type: `num`

The number of items that has passed the exit limit since `ResetFlowCount` was executed.

Arguments

```
GetFlowCount (ItemSource)
```

`ItemSource`

Data type: `itmsrc`

The item souce.

Error handling

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

Error code	Description
`ERR_ITMSRC_UNDEF`	`itmsrc` undefined

Syntax

```
GetFlowCount '('[ItemSource ':=' ] <variable (VAR) of itmsrc> ')';
```

A function returns value of the data type num.

Related information

For information about	See
The instruction `ResetFlowCount`	*ResetFlowCount - Reset flow counter on page 264*.

8.3 Data types

8.3.1 itmtgt - Item target data

Usage

`itmtgt` is used to describe one pick or place item.

Description

`Itmtgt` identifies an item to pick or place. It contains the position and some additional data.

Components

`tag`

Data type: `num`

Sequential number identifying the item. Can be modified by a user hook for free usage. Is restricted to integer values.

`type`

Data type: `num`

Type of item.

`scene`

Data type: `num`

Sequential number identifying the scene, corresponding for example to a picture taken by the vision system.

`robtgt`

Data type: `robtgt`

The pick or place position.

`val1`

Data type: `num`

Optional. Can be used to carry additional item specific information, for example, from a user hook. It is of data type float.

`val2`

Data type: `num`

Optional. Can be used to carry additional item specific information, for example, from a user hook. It is of data type float.

`val3`

Data type: `num`

Optional. Can be used to carry additional item specific information, for example, from a user hook. It is of data type float.

`val4`

Data type: `num`

Continues on next page

Optional. Can be used to carry additional item specific information, for example, from a user hook. It is of data type float.

`val5`

Data type: `num`

Optional. Can be used to carry additional item specific information, for example, from a user hook. It is of data type float.

Examples

Example 1

```
CONST itmtgt pickpos :=
     [1,2,1,0,0,0,0,0,[[20,40,8],[1,0,0,0],[0,0,0,0],
     [9E+9,9E+9,9E+9,9E+9,0,0]]];
```

A pick position is defined. The external axis related to the used conveyors must be set to zero, that is not marked as unused (by stating 9E+9). Example: if you have two conveyors, set the two last external axis positions to zero.

Structure

```
<dataobject of itmtgt>
  <tag of num>
  <type of num>
  <scene of num>
  <val1 of num>
  <val2 of num>
  <val3 of num>
  <val4 of num>
  <val5 of num>
  <dataobject of robtarget>
    <trans of pos>
      <x of num>
      <y of num>
      <z of num>
    <rot of orient>
      <q1 of num>
      <q2 of num>
      <q3 of num>
      <q4 of num>
    <robconf of confdata>
      <cf1 of num>
      <cf4 of num>
      <cf6 of num>
      <cfx of num>
    <extax of extjoint>
      <eax_a of num>
      <eax_b of num>
      <eax_c of num>
      <eax_d of num>
      <eax_e of num>
      <eax_f of num>
```

Continues on next page

8.3.1 itmtgt - Item target data
Continued

Related information

For information about	See
Positioning instructions	*Technical reference manual - RAPID Overview*
Coordinate systems	*Technical reference manual - RAPID Overview*
Handling configuration data	*Technical reference manual - RAPID Overview*
Configuration of external axes	*Technical reference manual - System parameters*
What is a quaternion?	*Technical reference manual - RAPID Overview*

8.3.2 selectiondata - Selection data

Usage

`selectiondata` is used to describe the selection criteria. It is also used to describe item sorting.

Description

`selectiondata` is used to set the criteria for sorting and clearance area when retrieving item targets from an item source.

Components

`ShapeType`

Data type: `shapetype`

Specifies the shape of the clearance area that should be used.

- `SHAPE_UNDEFINED` specifies that no selection is used.
- `BOX` specifies that there must be a clear box shape around the item target position where no other item targets are present.
- `CYLINDER` specifies there must be a clear cylinder shape around the item target position where no other item targets are present.
- `SPHERE` specifies that there must be a clear sphere shape around the item target position where no other item targets are present.

`ConsiderType`

Data type: `aconsidertype`

Specifies which items in the queue that should be taken in consideration when selecting.

- `ITEMS_TO_USE` specifies that only items marked for use by this queue are considered in the selection.
- `ITEMS_BYPASS` specifies that only items marked to pass by this queue are considered in the selection.
- `ITEMS_PICKED` specifies that only items marked as already picked, by this queue or by a former queue in the line, are considered in the selection.
- `ITEMS_PLACED` specifies that only items marked as already placed, by this queue or by a former queue in the line, are considered in the selection.

If items with different marks should be taken into consideration when selecting an item, then use a bit-or operation with the consideration types. (RAPID function `BitOr(<byte>,<byte>)`.)

`GeometricData`

Data type: `geodata`

The data that defines the geometric shape dimensions (x, y, z and radius).

- A `BOX` shape is defined by the x, y, and z-values.
- A `CYLINDER` shape is defined by the radius value and the height is defined by the z-value.

Continues on next page

- A SPHERE shape is defined by the radius value.

The orientation of the shape's coordinate system is defined by the offset data component. By default it is the coordinate system of the shape aligned to the workobject or conveyor frame. Note that all shapes origin are placed in the center of the shape and the values are the distance to every plane in both positive and negative direction. That is, if a box is defined as x: 10, y: 15 and z: 20 the box will have a size of 20 mm in x-direction, 30 mm in y-direction and 40 in z-direction. If no offset is used the check for other items in range will be done 10 mm before, 10 mm after, 15 mm left of, 15 mm right of, 20 mm above, and 20 mm underneath every item.

`Offset`

Data type: `offsetdata`

The offset consists of `OffsetRelation` (`offsetreltype`) and `OffsetPose` (`pose`).

The `OffsetRelation` can be of two different types.

- `FRAME_COORD_DIR` indicates that the rotation in the `OffsetPose` is relative to the workobject or conveyor frame coordinate system.
- `ITEM_COORD_DIR` indicates that the rotation in the `OffsetPose` is relative to the item coordinate system of the item to check.

The `OffsetPose` is used to move the center of the shape away from the item position, for example, if the grip position of the item is not at the center of real object to pick.

Examples

```
VAR selectiondata clear_rect:= [BOX,ITEMS_TO_USE,[22,15,5,0],
     [FRAME_COORD_DIR,[[0,7,0],[1,0,0,0]]]];
```

Limitations

The orientation must be normalized; that is the sum of the squares must equal 1.

$q1^2 + q2^2 + q3^2 + q4^2 = 1$

Structure

```
<dataobject of selectiondata>
  <ShapeType of shapetype>
  <ConsiderType of considertype>
  <GeometricData of geodata>
    <x of num>
    <y of num>
    <z of num>
    <radius of num>
  <Offset of offsetdata>
    <OffsetRelation of offsetreltype>
    <OffsetPose of pose>
      <trans of pos>
        <x of num>
        <y of num>
        <z of num>
```

Continues on next page

```
<rot of orient>
  <q1 of num>
  <q2 of num>
  <q3 of num>
  <q4 of num>
```

Related information

For information about	See
The data type `pose`	*Technical reference manual - RAPID Instructions, Functions and Data types*.
The function `BitOr`	*Technical reference manual - RAPID Instructions, Functions and Data types*.
What is a quaternion?	*Technical reference manual - RAPID Overview*.
Example using `selectiondata`	*Example: Selecting item depending on clearance zone on page 302*.

8.3.3 sortdata - Sort data

Usage

`sortdata` is used to describe the sorting criteria.

Description

`sortdata` is used to set the criteria for sorting item targets from an item source.

Components

`SortType`

Data type: `sorttype`

Type of sorting that is going to be used.

- `UNSORT_TYPE` tells that no sorting is used.
- `POS_Y_SORT_TYPE` tells that the sorting shall be done from the positive y-direction of the work area.
- `NEG_Y_SORT_TYPE` tells that the sorting shall be done from the negative y-direction of the work area.

`CheckBoundary`

Data type: `num`

The clearance distance for sorting, in millimeters. The distance is defined as the minimum distance to the next item in the sorting direction.

`SortDirOffset`

Data type: `num`

An offset distance beyond the item target in the sort direction. Is used to define the inner limit for the corridor in which no other item targets are allowed.

Examples

```
VAR sortdata y_sort:=[NEG_Y_SORT_TYPE ,78, 52];
```

Structure

```
<dataobject of sortdata>
  <SortType of sorttype>
  <CheckBoundary of num>
  <SortDirOffset of num>
```

8.4 RAPID program

8.4.1 RAPID programs

Introduction

Overview

Each robot has a default RAPID program that can be edited using a normal text editor from the robot settings of the job dialog. When a job is started, the program is downloaded by PickMaster in the picking controller. The program contains the Main routine where the program execution starts.

> **Note**
>
> Due to the download procedure, this program cannot be modified directly on the robot system.

The installation contains the following program template files:

Template	Customized for
PMppa360_IRC5.prg	Four axes FlexPicker IRB 360.
PMppa360_IRC5_MM2.prg	Four axes FlexPicker IRB 360. To be used for a second multi move robot.
PMppa360_IRC5_DoublePick.prg	Four axes FlexPicker IRB 360. Adapted for double pick, single place.
PMppa360_IRC5_6_02.prg	Four axes FlexPicker IRB 360. To be used with RobotWare 6.02.
PMppa360_IRC5_6_02_DoublePick.prg	Four axes FlexPicker IRB 360. Adapted for double pick, single place. To be used with RobotWare 6.02.
PMppa_6Axes_IRC5.prg	Six axes robots of articulated arm type, for example, IRB 120.
PMppa_6Axes_IRC5_6_02.prg	Six axes robots of articulated arm type, for example, IRB 120. To be used with RobotWare 6.02.
PMppa360_IRC5_RW5_15.prg	Four axes FlexPicker IRB 360 or IRB 340. To be used with RobotWare 5.15.

Program execution - General

The RAPID program is loaded and started from the Main routine by PickMaster when a new job is started.

For every cycle, the default RAPID program performs:

- a pick on a pick work area.
- a place on a place work area.

If there are more than one pick work area with a robot, it uses the one having the lowest configured work area index. If there are more than one place work area with a robot, it uses the one having the lowest configured work area index. The RAPID program can be modified to implement another sequence, for example, to double pick with single place.

Continues on next page

8.4.1 RAPID programs
Continued

Program execution – Work areas

In RAPID, a work area is always associated with an item source object. The item source is sometimes referred to as a queue. The item source holds all target positions related to this work area. Target positions are continuously received in the item source, while being detected with the associated flow handler sensor.

Program execution – Target positions

For each pick, a pick target is fetched from the pick item source. The target position gives the location of the next item to be picked.

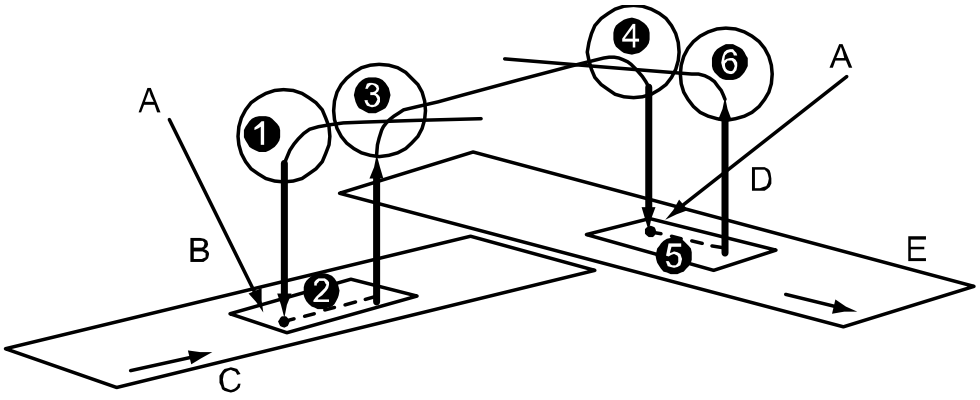
For each place, a place target is fetched from the place item source. The target position gives the location of the next empty place location for the item to be placed.

Movements

The RAPID program is built with six different movements.

For a six axis robot, the following two intermediate points must be used:

- Between position 3 and position 4.
- Between position 6 and the next loop's position 1.



xx0800000326

The following six movements are included.

	Description
1	Approach position above the pick target. The distance above the pick target is the pick elevation value, in negative z-direction of the tool, given in the **Work Area Properties** dialog in the job dialog. The target is of corner path type and the vacuum activation occasion is calculated as the time before the middle of the corner path. The time is entered in the **Work Area Properties** dialog.
2	This is the pick target. The robot TCP is coordinated relative to the conveyor during the pick time entered in the **Work Area Properties** dialog. The TCP follows the pick target during the pick time.
3	Last position in the pick sequence. The distance above the pick target is calculated in the same way as the approach position. The position is coordinated to the conveyor until the middle of the corner path. Therefore the used item target must be acknowledged, so the item source can start tracking the next item target in the pick work area buffer. The target cannot be a fine point.

Continues on next page

	Description
4	Approach position above the place target. The distance above the place target is the pick or place elevation value, in negative z-direction of the tool, given in the **Work Area Properties** dialog.
5	This is the place target. The robot TCP is coordinated relative to the conveyor during the place. The moment for the vacuum reversion event is calculated as the time before the half place time. The vacuum off moment is calculated as a time after the half place time.
6	Last position of the sequence. The position is coordinated to the conveyor until the TCP passes the middle of the corner path or goes into the fine point. Therefore the used item target must be acknowledged, so the item source can start tracking the next item target in the pick work area buffer. The target cannot be a fine point.

Note

When running a pick and place cycle over moving conveyors, the RAPID program pointer runs in advance and picks out a target long before it is going to be used. By the time the robot uses the target it may already have moved past the exit limit. RAPID moves the program pointer in advance about 100ms. In a coordinated fine point the "running in advance" is triggered at the beginning of the fine point movement as the robot locks above the conveyor. If the PickTime is long (for example, 50ms) the next target will be taken out of the queue long before (50ms) the robot is physically going to go there. If the conveyor speed is high 50ms may mean that the target to pick is already beyond the exit limit. Still the robot will try to pick it.

Program modules

The default RAPID program contains three program modules.

Module	Description
PPAMAIN	Handles the main program initiations and execution sequence. Do not edit this module for customization purpose.
PPASERVICE	Contains the service routines and service variables that can be executed and set from PickMaster.
PPAEXECUTING	Handles the pick and place operations. Customize this module for your purposes.

System modules

An ABB IRC5 robot controller with the RobotWare option *Prepared for PickMaster (PickWare)* will always contain the loaded system modules *ppaBase* (crypted) and *ppaUser* (open).

Module	Description
ppaBase	Contains variables for communication with PickMaster, event routines and routines for creating, initiating, and deleting item sources.
ppaUser	Contains declarations of public data types and holds the work object data for indexed work areas. It also contains the declaration of default tool data, for example, `PickAct1` and `PickAct2`.

Continues on next page

8.4.1 RAPID programs
Continued

Public data types

Overview

The system module *ppaUser* contains two record definitions, *sourcedata* and *noncnvwobjdata*.

sourcedata

The *sourcedata* is used in the variable array *ItmSrcData*. This array holds data about every item source.

The record can be extended for other purposes, but do not change or delete any component in the structure.

Name	Alias	Description
`Used`	bool	Flag to indicate that the array index is used.
`ItemSource`	itmsrc	Descriptor to the item source.
`SourceType`	itmsrctype	Type of source, `PICK_TYPE`, `PLACE_TYPE` or `UNDEFINED_TYPE`.
`Ack`	`triggdata`	Triggdata for acknowledging the item targets.
`Nack`	`triggdata`	Triggdata for negative acknowledging the item targets.
`SimAttach1`	`triggdata`	Triggdata for attaching a nearby item to activator 1 in simulation.
`SimAttach2`	`triggdata`	Triggdata for attaching a nearby item to activator 2 in simulation.
`SimDetach1`	`triggdata`	Triggdata for detaching an item held by activator 1 in simulation.
`SimDetach2`	`triggdata`	Triggdata for detaching an item held by activator 2 in simulation.
`VacuumAct1`	`triggdata`	Triggdata for vacuum activation on real robot.
`VacuumAct2`	`triggdata`	Triggdata for vacuum activation on real robot.
`VacuumRev1`	`triggdata`	Triggdata for vacuum blow on real robot.
`VacuumRev2`	`triggdata`	Triggdata for vacuum blow on real robot.
`VacuumOff1`	`triggdata`	Triggdata for vacuum off on real robot.
`VacuumOff2`	`triggdata`	Triggdata for vacuum off on real robot.
`Wobj`	`wobjdata`	Work object data for the source
`VacActDelay`	`num`	Vacuum activation delay
`VacRevDelay`	`num`	Vacuum reversion delay
`VacOffDelay`	`num`	Vacuum off delay
`TunePos`	`pos`	Position tuning for the work area.
`TrackPoint`	`stoppointdata`	Follow time data.
`OffsZ`	`num`	Height for the offset point above the pick or place position.

Continues on next page

noncnvwobjdata

The *noncnvwobjdata* is used in the persistent variable array *NonCnvWOData*. This is only used for indexed work areas. The work object data is stored in this array. This data is then used when the item sources are created.

The record can be extended for other purposes, but do not change or delete any component in the structure.

Name	Alias	Description
Used	bool	Flag to indicate that the array index is used.
NonCnvWobjName	string	Name of the work area.
Wobj	wobjdata	The stored work object data.

AlwaysClearPath

Clear path

The robot path is cleared before the restart when a stop occurs during a motion that is coordinated to a moving work object. Otherwise the coordinated motion continues the stored path, but the position of the object in the conveyor may have changed to a position that is out of reach by the robot.

Unconditional path clearing

The `AlwaysClearPath` (bool always) routine unconditionally clears the path before the restart, if the input parameter value is set to TRUE.

8.4.2 Variables

Introduction to variables

The PickMaster robot controller contains many RAPID variables. The variables are declared in both `ppaBase` and `ppaUser`. Many are not used in customized programs.

Public variables in ppaUser

Overview

The following variables in `ppaUser` can be used.

VAR sourcedata ItmSrcData{MaxNoSources}

This array variable keeps information about all work areas. The index given in the work area configuration is the index of the `ItmSrcData` array.

PERS noncnvwobjdata NonCnvWOData{MaxNoSources}:=[[…

This array variable stores the work object frames for the indexed work areas. The key to find a certain work object calibration is the name, that must be same as the name in the work area configuration.

TASK PERS tooldata PickAct1:=[…]

This tooldata is used for pick and place operations.

> **Note**
>
> The direction of tool must fit the direction of items that are retrieved from the queue. The target positions of the items, which are retrieved from the queue, are rotated 180 degrees around their x-axis from the defined direction.

In an installation with a hanging IRB 360 and items lying on a horizontal conveyor, the tool's z-direction will point out from the nose and down into the conveyor, like `tool0`.

Public variables in ppaBase

The following variables in `ppaBase` can be used.

TASK PERS num Vtcp:=1000

Used for speed adjustment from PickMaster.

TASK PERS speeddata MaxSpeed:=[…]

Highest speed used for movements.

TASK PERS speeddata LowSpeed:=[…]

Low speed used for movements.

TASK PERS speeddata VeryLowSpeed:=[…]

Lowest speed used for movements.

Continues on next page

Public variables in PickMaster template programs

The following public variables are used in the PickMaster template program.

VAR num PickWorkArea{X}:=0

The PickWorkArea array is used to specify from which work area the robot will pick an item. The pick work areas are ordered with respect to selection index.

`PickWorkArea{1}` has the lowest work area selection index.

`PickWorkArea{2}` has the second lowest selection index.

VAR num PlaceWorkArea{X}:=0

The PlaceWorkArea array is used to specify on which work area the robot will place an item. The place work areas are ordered with respect to selection index.

`PlaceWorkArea{1}` has the lowest work area selection index.

`PlaceWorkArea{2}` has the second lowest selection index.

VAR num OtherWorkArea{X}:=0

The OtherWorkArea array is used to specify to which work area the robot will go for a user defined purpose. The other work areas are ordered with respect to selection index.

`OtherWorkArea{1}` has the lowest work area selection index.

`OtherWorkArea{2}` has the second lowest selection index.

VAR itmtgt PickTarget:=[...]

Used to retrieve a pick target from a pick item source.

VAR itmtgt PlaceTarget:=[...]

Used to retrieve a place target from a place item source.

TASK PERS wobjdata WObjPick:=[…]

Holds the wobjdata for the work area. The information is moved from `ItmSrcData` to `WObjPick` in the Pick routine because the motion instructions need to have the wobjdata as `PERS` type.

TASK PERS wobjdata WObjPlace:=[…]

Holds the `wobjdata` for the work area. The information is moved from `ItmSrcData` to `WObjPlace` in the Place routine because the motion instructions need to have the `wobjdata` as `PERS` type.

TASK PERS robtarget SafePos:=[…]

Defined start position for the robot. Edit this robtarget to fit the application.

TASK PERS robtarget IntPosPickX:=[…]

Defined intermediate position for every pick work area robot. Edit this robtarget to fit each work area.

TASK PERS robtarget IntPosPlaceX:=[…]

Defined intermediate position for every place work area robot. Edit this robtarget to fit each work area.

Continues on next page

8.4.2 Variables
Continued

TASK PERS loaddata ItemLoad:=[…]

Load data (`loaddata`) used for pick and place operations. Edit this `loaddata` to fit the picked item. If different item types are used, declare one loaddata for each type. It is important that correct loaddata is used to get the best performance of the robot.

The default loaddata is the same as `tooldataload0`.

8.4.3 Routines

Introduction to routines

The PickMaster RAPID modules contain many routines, some are very useful for the end user, others are only to be used internally by the PickMaster program.

Public routines in PickMaster template programs

The following public routines are available in the PickMaster template programs.

PROC main()

Start routine for the RAPID program. The program will always start from this routine.

PROC InitSafeStop()

Initiates the SafeStop trap. It must be executed at the beginning of the program execution to get a correct robot stop when the PickMaster project is paused or stopped.

PROC InitTriggs()

Sets trigger events for the vacuum activation, reversion and turning off, at the project start for every used work area index. See more at `SetTriggs`.

PROC InitPickTune()

Initiates the PickTune trap. Must be executed at the beginning of the project start so the work areas can be tuned.

PROC SetTriggs(num Index)

Sets trigger events for the vacuum activation, reversion and turning off. The default program only sets up events for one vacuum ejector on the I/O group `goVacBlow1`. If more than one vacuum ejector is used, the new vacuum ejector I/O group must be setup for the correct work area and the default routine must be edited to get the right vacuum ejector to each work area.

PROC InitSpeed()

Sets the robot speed used in the program. The instruction `VelSet` is executed in this routine, which sets the maximum allowed speed for the robot. If a six axes robot is used, this limit can be tuned to avoid motion errors.

PROC PickPlace()

Starts the item queues and initiates the final settings. The pick and place sequence is called from this routine. Do not make changes in this routine.

This routine is called when the pick and place execution is started.

PROC SafeStop()

When the project is stopped or paused this routine will be called either from the `SafeStopTrap` routine or the `PickPlace` routine. The slow motion to the safe position is called from this routine.

Continues on next page

8.4.3 Routines
Continued

PROC GotoRestartPos()

Runs the slow motion to the safe position and sends a negative acknowledge to all item sources. This must be done to tell the sources that the execution was interrupted.

PROC Home()

Service routine that moves the robot to the safe position.

PROC WashDown()

Wash down service routine.

PROC TestCycle()

Test service routine.

PROC Homepos()

Service routine that moves the robot to the synchronization position.

PROC EnumerateWorkAreas()

Sets up the arrays of work areas for Pick, Place, and Other.

PROC PickPlaceSeq()

Specifies the sequence of the application, that is the logic of how the robot will pick and place from different queues.

This routine is called once every loop, which is counted as one pick in the pick rate statistics shown in the PickMaster production tab.

PROC Pick(num Index)

Executes one pick. The index defines which work area the item will be picked from.

PROC Place(num Index)

Executes one place. The given index defines which work area the item will be placed on.

PROC GoInterMidP(robtarget Pos)

Executes a motion to the intermediate position `Pos` (Used only for six axes robots).

TRAP SafeStopTrap

Trap routine to catch the stop I/O signal. This is executed if the stop I/O signal is set before `SafeStop` is called from the `PickPlace` routine.

TRAP PickTuneTrap

Trap routine to attach the tuned values from the PickMaster to the corresponding variables.

Hidden routines in ppaBase module

Overview

Following are the hidden routines in the `ppaBase` module.

PROC ResetEvent()

Resets some variables. This routine is only executed in the `RESET` system event shelf.

Continues on next page

PROC PowerOnEvent()

Resets some variables. This routine is executed only in the `POWER_ON` system event shelf.

PROC StopEvent()

Clears the robot path if the robot is in a coordinated motion when the stop occurs. This routine is only executed in the `STOP` system event shelf.

PROC RestartEvent()

This routine is only executed in the `RESTART` system event shelf. If the robot is currently in a coordinated motion, this routine will force the program to restart the program from the level that has an error handler for the raised error `PPA_RESTART`.

PROC NewSource()

Creates a new item source and initiates the `ItmSrcData` variable. PickMaster calls this routine for each work area when the project starts.

PROC ClearAll()

Resets all important variables and deletes all item sources. This routine is called when the project is stopped.

PROC PickRateInit()

Initiates the pick rate calculation.

PROC PickRateReset()

Resets the pick rate calculation.

PROC CheckAx4Rev ()

Checks if it is necessary to reset the fourth axis on the IRB340.

PROC ResetAx4 (VAR mecunit MechUnit)

Resets the fourth axis.

PROC NotifyClearAll ()

Tells PickMaster that `ClearAll` is executed.

PROC NotifySafeStop ()

Tells PickMaster that `SafeStop` is executed.

PROC NotifyRunning ()

Tells PickMaster that the process is running.

PROC NotifyWaitForExe ()

Tells PickMaster that the RAPID program is waiting for new order.

PROC WaitForExeOrder ()

Instruction where the RAPID program waits for PickMaster to give the next execution order. If no order is given, the RAPID execution will wait and idle on this instruction.

PROC IncrPicks ()

Increments the pick calculation.

Continues on next page

8.4.3 Routines
Continued

PROC ppaDropWobj(PERS wobjdata Wobj)

Encapsulates the `DropWobj` instruction. See *Application manual - Conveyor tracking* for more information

PROC ppaWaitWobj(PERS wobjdata Wobj, \num RelDist \num MaxTime \bool TimeFlag)

Encapsulates the `WaitWobj` instruction. See *Application manual - Conveyor tracking* for more information

PROC ppaCnvGenInstr(VAR mecunit MechUnit, num cnvcmd, cnvgendata Data)

Encapsulates the `CnvGenInstr` instruction. See *Application manual - Conveyor tracking* for more information

PROC WalkTheData()

Traces the content of the array variables *ItmSrcData* and *NonCnvWOData*, which can be useful when trying to find an error. It prints the file TheData.log on the system directory on the controller.

TRAP PickRateTrap

Trap routine to calculate the correct pick rate for the robot.

8.4.4 Service routines

User defined service routines

It is possible to create new routines and variables. All procedure in the PPASERVICE module can be executed as service routines. All service variable names must be set in a ServiceVarX.

The service variables can only be of type num.

Maximum number of service variables is ten.

8.5 Program examples

8.5.1 Example: Mixing one pick work area and two place work areas

Description of example

In this example we use one pick work area with two types of items. The items are put on two out work areas depending on type of item.

1 Pick item from pick work area
2 Define type of item
3 Place on out work area

Example code

```
PROC PickPlaceSeq()
  Pick PickWorkArea{1};
  IF PickTarget.Type = 1 THEN
    Place PlaceWorkArea{1};
  ELSEIF PickTarget.Type = 2 THEN
    Place PlaceWorkArea{2};
  ENDIF
ENDPROC
```

8.5.2 Example: Mixing two pick work areas and one place work area

Description of example

In this example, we use the place work area as master to decide which item is needed to fill a pattern, which in turn defines pick work area to pick from.

1 Check next item target type
2 Decide which work area to pick from
3 Pick item from pick work area
4 Place on out work area

Example code

```
PROC PickPlaceSeq()
  VAR num PlaceType:=0;

  NextItmTgtType
    ItmSrcData{PlaceWorkArea{1}}.ItemSource,
    PlaceType;
  IF PlaceType = 1 THEN
    Pick PickWorkArea{1};
  ELSEIF PlaceType = 2 THEN
    Pick PickWorkArea{2};
  ENDIF
  Place PlaceWorkArea{1};
ENDPROC
```

8.5.3 Example: Mixing with one pick and one place work area

Description of example

In this example we use the place work area as master to decide which item is needed to fill a pattern, which in turn defines which item to pick.

1 Check next item target type
2 Pick item from pick work area
3 Place on out work area

> **Note**
>
> It's recommended to use the **Use Start/Stop** in the **Available Work Areas** setting.

Example code

```
PROC Pick(num Index)
  VAR num PickType:=0;
  VAR num PlaceType:=0;

  WObjPick:=ItmSrcData{Index}.Wobj;
  NextItmTgtType
    ItmSrcData{PlaceWorkArea{1}}.ItemSource,PlaceType;
  TEST PlaceType
  CASE 4:
    PickType:=1;
  CASE 5:
    PickType:=2;
  CASE 6:
    PickType:=3;
  ENDTEST
  GetItmTgt ItmSrcData{Index}.ItemSource, PickTarget
       \ItemType:=PickType;
  TriggL \Conc, RelTool(PickTarget.RobTgt, 0, 0,
       -ItmSrcData{Index}.OffsZ), MaxSpeed,
       ItmSrcData{Index}.VacuumAct1, z20, PickAct1 \WObj:=WObjPick;
  MoveL \Conc, PickTarget.RobTgt, LowSpeed, z5 \Inpos:=
       ItmSrcData{Index}.TrackPoint, PickAct1 \WObj:=WObjPick;
  GripLoad ItemLoad;
  TriggL RelTool(PickTarget.RobTgt, 0, 0, -ItmSrcData{Index}.OffsZ),
       LowSpeed, ItmSrcData{Index}.Ack, z20, PickAct1
       \WObj:=WObjPick;
ENDPROC
```

8.5.4 Example: Double pick single place

Description of example

The robot shall pick up two items, one-by-one, on the infeeder conveyor, and then place both items on the outfeed conveyor. This operation requires a picking tool with two vacuum ejectors.

Implementation

As a starting point, create a simple working setup with one robot.

The RAPID program needs to be modified. To edit the RAPID program, go to the **Project view**, select a robot and display the drop down menu, select the **Rapid program** and select **Edit...**.

The `PickPlaceSeq` routine shall perform two Pick routine calls to handle the first and the second pick. It will then perform one Place routine call to handle the simultaneous placing of the picked up items. See the following example code.

```
!
! Procedure PickPlaceSeq
!
! The Pick and Place sequence.
! Edit this routine to specify how the robot shall execute the
    movements.
!

!***********************************************************
PROC PickPlaceSeq()
  Pick PickWorkArea{1}, 1;
  Pick PickWorkArea{1}, 2;
  Place PickWorkArea{1};
ENDPROC
```

For the Pick routine, see the following example code. Note the usage of `PickAct2` and `VacuumAct2` for the second pick.

```
!***********************************************************
!
! Procedure Pick
!
! Executes a pick movement.
! Edit this routine to modify how the robot shall
! execute the pick movements.
! Needs to be changed if more than one activator is used.
!
!***********************************************************
PROC Pick(num Index, num pickNo)
  IF Index > 0 THEN
    WObjPick:=ItmSrcData{Index}.Wobj;
    GetItmTgt ItmSrcData{Index}.ItemSource,PickTarget;
    IF pickNo = 1 THEN
      TriggL\Conc,RelTool(PickTarget.RobTgt,0,0,
      -ItmSrcData{Index}.OffsZ),
```

Continues on next page

8.5.4 Example: Double pick single place
Continued

```
        MaxSpeed,ItmSrcData{Index}.VacuumAct1,z20,
        PickAct1\WObj:=WObjPick;
        TriggL\Conc,PickTarget.RobTgt,LowSpeed,ItmSrcData{Index}.SimAttach1,
        z5\Inpos:=ItmSrcData{Index}.TrackPoint,
        PickAct1\WObj:=WObjPick;
        GripLoad ItemLoad;
        TriggL
        RelTool(PickTarget.RobTgt,0,0,-ItmSrcData{Index}.OffsZ),
        LowSpeed,ItmSrcData{Index}.Ack,z20,PickAct1\WObj:=WObjPick;
      ELSEIF pickNo = 2 THEN
        TriggL\Conc,RelTool(PickTarget.RobTgt,0,0,-ItmSrcData{Index}.OffsZ),
        MaxSpeed,ItmSrcData{Index}.VacuumAct2,
        z20,PickAct2\WObj:=WObjPick;
        TriggL\Conc,PickTarget.RobTgt,LowSpeed,ItmSrcData{Index}.SimAttach2,
        z5\Inpos:=ItmSrcData{Index}.TrackPoint,
        PickAct2\WObj:=WObjPick;
        GripLoad ItemLoad;
        TriggL
        RelTool(PickTarget.RobTgt,0,0,-ItmSrcData{Index}.OffsZ),
        LowSpeed,ItmSrcData{Index}.Ack,z20,
        PickAct2\WObj:=WObjPick;
      ENDIF
    ELSE
      ErrWrite "Missing item distribution", "Cannot pick because no
           item distribution contains current work area."
      \RL2:="Please check configuration";
      SafeStop;
    ENDIF
  ENDPROC
```

The tooldata `PickAct1` is used at the first pick. The tooldata `PickAct2` is used at the second pick. Update `PickAct1` and `PickAct2` (defined in module `ppaUser.sys`): Define the tool center point in the center of the controlled vacuum ejector. Update also the weight and the center of mass. Save the updates of the RAPID program, close the editor, and apply the updates.

For the Place routine see the following example. Note the usage of `VacuumOff1` and `VacuumOff2` for the simultaneous placing of both held items.

```
!*********************************************************
!
! Procedure Place
!
! Executes a place movement.
! Edit this routine to modify how the robot shall
! execute the place movements.
! Needs to be changed if more than one activator is used.
!
!*********************************************************
PROC Place(num Index)
  IF Index > 0 THEN
    WObjPlace:=ItmSrcData{Index}.Wobj;
    GetItmTgt ItmSrcData{Index}.ItemSource,PlaceTarget;
```

Continues on next page

```
      MoveL\Conc,RelTool(PlaceTarget.RobTgt,0,0,-ItmSrcData{Index}.OffsZ),
      MaxSpeed,z20,PlaceAll\WObj:=WObjPlace;
      TriggL\Conc,PlaceTarget.RobTgt,LowSpeed,ItmSrcData{Index}.VacuumRev1\T2:=
      ItmSrcData{Index}.VacuumOff1\T3:=ItmSrcData{Index}.VacuumOff2\T4:=
      ItmSrcData{Index}.VacuumRev2\T5:=ItmSrcData{Index}.SimDetach1\T6:=
      ItmSrcData{Index}.SimDetach2,z5\Inpos:=
      ItmSrcData{Index}.TrackPoint,PlaceAll\WObj:=WObjPlace;
      GripLoad load0;
      TriggL RelTool(PlaceTarget.RobTgt,0,0,-ItmSrcData{Index}.OffsZ),
      LowSpeed,ItmSrcData{Index}.Ack,z20,PlaceAll\WObj:=WObjPlace;
    ELSE
      ErrWrite "Missing item distribution", "Cannot place because no
           item distribution contains current work area."
      \RL2:="Please check configuration";
      SafeStop;
    ENDIF
  ENDPROC
```

The tooldata `PlaceAll` (defined in module `ppaUser.sys`) is used at place. Update `PlaceAll`: Define the tool center point in the center of the controlled vacuum ejectors. Update also the weight and the center of mass. Save the updates of the RAPID program, close the editor, and apply the updates.

Note

Use the same method to setup a tool with more than two activators. However, a few additional setup steps are required. For example, using a tool with 3-4 activators requires the following additional steps:

1. Select two I/O boards as controller option. Alternatively, create additional signals `goVacBlow3`, `goVacBlow4`, `doVacuum3`, `doVacuum4`, `doBlow3`, and `doBlow4`. The first bit of `goVacBlowX` shall overlap the signal `doVacuumX`. The second bit of `goVacBlowX` shall overlap the signal `doBlowX`.
2. Update the `SetTriggs` routine. Enable the `TriggEquip` events `VacuumAct3`, `VacuumOff3`, `VacuumAct4`, and `VacuumOff4` by removing the comments on these lines.

8.5.5 Example: Placing a predefined pattern on indexed work area

Description of example

In this example we place a predefined pattern on an indexed work area. The position generator signal is triggered from RAPID.

Four new signals must be defined.

1. Position generator signal set from RAPID, `doSIMPosGen`.
2. Position generator signal that generates an event from the controller to the computer, `diSIMPosGen`.
3. Trigger signal that tells the system on the computer to send a predefined position, `doSIMTrig`.
4. Strobe signal that tells the system a position is sent, `diSIMStrobe`.

The signals can be defined on the PPASIM board. For example:

```
-Name "doSIMPosGen" -SignalType "DO" -Unit "PPASIM" -UnitMap "6"
     -Access "ALL"
-Name "doSIMTrig" -SignalType "DO" -Unit "PPASIM" -UnitMap "7"
     -Access "ALL"
-Name "diSIMPosGen" -SignalType "DI" -Unit "PPASIM" -UnitMap "6"
     -Access "ALL"
-Name "diSIMStrobe" -SignalType "DI" -Unit "PPASIM" -UnitMap "7"
     -Access "ALL"
```

Cross connect the trigger and strobe signal and the position generator signals. For example:

```
EIO_CROSS
  -Res "diSIMPosGen" -Act1 "doSIMPosGen"
  -Res "diSIMStrobe" -Act1 "doSIMTrig"
```

In the RAPID code, create a control of the place queue. If the queue is empty (all positions in the pattern are used) set the signal `doSIMPosGen` high (in the RAPID code). This signal is cross connected with the `diSIMPosGen` and an event will be sent to the computer from the controller that a new pattern has to be sent to the controller. The trigger strobe signals are also cross connected and the `diSIMStrobe` will be used to strobe the system.

Example code

```
PROC Place(num Index)
  VAR bool flagplace:=TRUE;

  WObjPlace:=ItmSrcData{Index}.Wobj;
  flagplace:=TRUE;

  WHILE flagplace=TRUE DO
    GetItmTgt ItmSrcData{Index}.ItemSource,
         PlaceTarget\MaxTime:=1\TimeFlag:=flagplace;
    IF flagplace=TRUE THEN
      PulseDO\PLength:=0.2,doSIMPosGen;
    ENDIF
  ENDWHILE
```

Continues on next page

```
  MoveL\Conc, RelTool(PlaceTarget.RobTgt, 0, 0,
       ItmSrcData{Index}.OffsZ), MaxSpeed, z20,
       PickAct1\WObj:=WObjPlace;
  TriggL\Conc, PlaceTarget.RobTgt, LowSpeed,
       ItmSrcData{Index}.VacuumRev1
       \T2:=ItmSrcData{Index}.VacuumOff1, z5
       \Inpos:=ItmSrcData{Index}.TrackPoint,
       PickAct1\WObj:=WObjPlace;
  GripLoad load0;
  TriggL RelTool(PlaceTarget.RobTgt, 0, 0, ItmSrcData{Index}.OffsZ),
       LowSpeed, ItmSrcData{Index}.Ack, z20,
       PickAct1\WObj:=WObjPlace;
ENDPROC
```

8.5.6 Example: Selecting item depending on clearance zone

Description of example

In this example, we select items on a conveyor belt depending on the clearance zone around the item, that is if there is any other item target within a specified area. This is useful when it is important that the gripper does not touch surrounding objects.

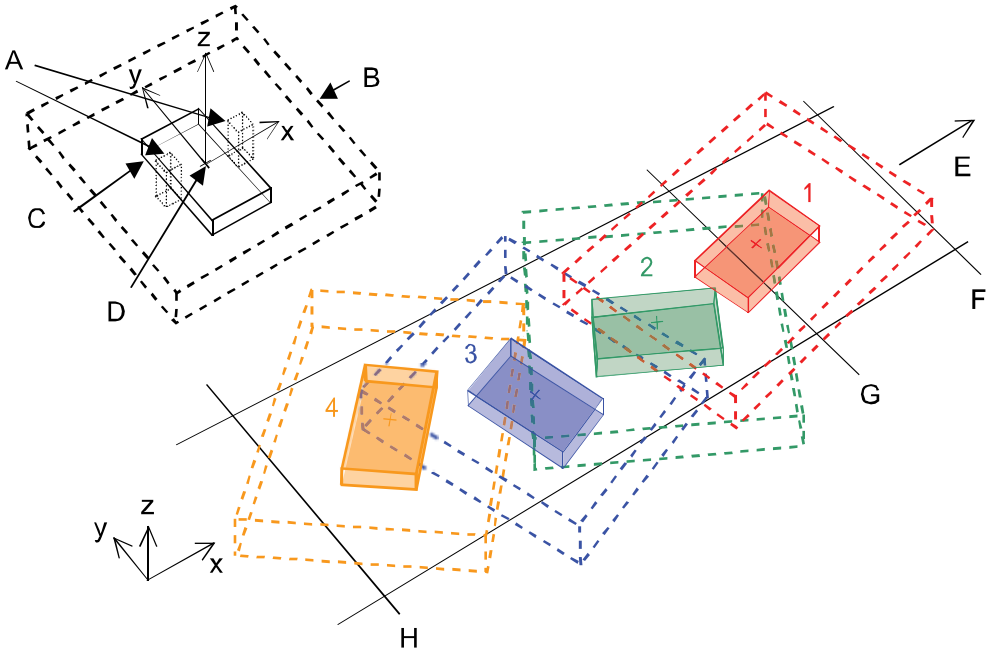
The selection algorithm selects the object that is closest to the exit limit in x-direction and has no locking objects in the selection shape.

Use the check limit in x-direction as a parameter to the `GetItmTgt` instruction. This makes it possible to define the starting point from where the first object will be picked. The instruction will try to retrieve the first object between the check and enter limits. This will cause the selection algorithm to take all objects between the check limit and the exit limit into consideration when checking for the nearest objects. Therefore the distance between the check limit and the exit limit will be at least the diameter of the largest item.

The illustration below shows how the items are selected depending on the position and the orientation. The robot will first pick item 4 and then item 3. The other two will never be picked.

- Item 1 cannot be picked because it has passed the check limit, and item 2 is inside its selection shape.
- Item 2 cannot be picked because the positions of items 1 and 3 are inside its selection shape.
- Item 3 cannot be picked because item 4 is inside its selection area.
- Item 4 can be picked because no other item is its selection shape.
- Item 3 will be picked after item 4 is no longer present.

Continues on next page



xx0800000323

A	Grippers
B	Selection shape
C	Item
D	Item target position
E	Product flow direction
F	Exit
G	Check limit
H	Enter

See *selectiondata - Selection data on page 277*.

Example code

```
PROC Pick(num Index)
  VAR selectiondata sel_data;
  VAR robtarget draw_target;
  VAR num check_limit;

  sel_data.ShapeType:=BOX;
  sel_data.ConsiderType:=BitOr(ITEMS_TO_USE,ITEMS_BYPASS);
  sel_data.GeometricData.x:=60;
  sel_data.GeometricData.y:=70;
  sel_data.GeometricData.z:=10;sel_data.GeometricData.radius:=0;
  sel_data.Offset.OffsetRelation:=ITEM_COORD_DIR;
  sel_data.Offset.OffsetPose.trans.x:=0;
  sel_data.Offset.OffsetPose.trans.y:=0;
  sel_data.Offset.OffsetPose.trans.z:=0;
  sel_data.Offset.OffsetPose.rot.q1:=1;
  sel_data.Offset.OffsetPose.rot.q2:=0;
```

Continues on next page

8.5.6 Example: Selecting item depending on clearance zone
Continued

```
  sel_data.Offset.OffsetPose.rot.q3:=0;
  sel_data.Offset.OffsetPose.rot.q4:=0;
  check_limit:=150;

  WObjPick:=ItmSrcData{Index}.Wobj;
  GetItmTgt ItmSrcData{Index}.ItemSource,PickTarget
       \Limit:=check_limit\Selection:=sel_data;
  TriggL \Conc, RelTool(PickTarget.RobTgt, 0, 0,
       -ItmSrcData{Index}.OffsZ), MaxSpeed,
       ItmSrcData{Index}.VacuumAct1, z20, PickAct1\WObj:=WObjPick;
  MoveL \Conc, PickTarget.RobTgt, LowSpeed, z5 \Inpos:=
       ItmSrcData{Index}.TrackPoint, PickAct1\WObj:=WObjPick;
  GripLoad ItemLoad;
  TriggL RelTool(PickTarget.RobTgt, 0, 0, -ItmSrcData{Index}.OffsZ),
       LowSpeed, ItmSrcData{Index}.Ack, z20,
       PickAct1\WObj:=WObjPick;
ENDPROC
```

8.5.7 Example: Sorting in negative y-direction

Description of example

In this example, we shuffle items off a conveyor belt without touching surrounding objects. The shuffle movement is done perpendicular on the horizontal plane to the right side of the conveyor and the manipulator motion is coordinated with the conveyor motion.

The sorting algorithm selects the item closest to the exit limit in x-direction and has no locking objects in its selection shape.

The selection shape is defined as a long box. The shape's x-value is used to define the corridor width, the y-value must be more than half the width of the conveyor belt and the z-value must be greater than the largest difference in height among all items.

Set the y-value in the `OffsetData` to the negative y-value of the shape, the selection box will be moved out to the right.

As a result there must be a clear corridor to the right of every item before it is shuffled.

The algorithm will check both upwards and downwards the production flow for other items.

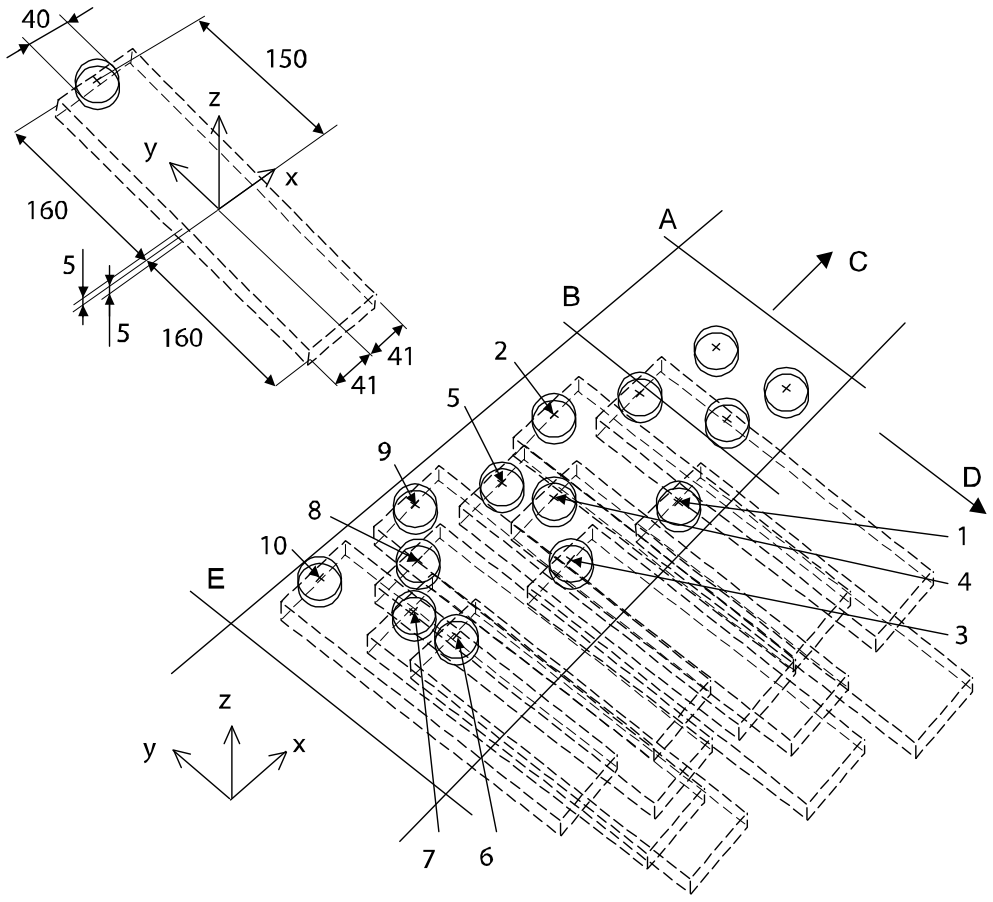
Use the check limit in the x-direction as a parameter to the `GetItmTgt` instruction, to define the starting point from where the first item will be shuffled. The instruction will try to shuffle the first item between the check and enter limits. This will also cause the selection algorithm to take all items between the check limit and the exit limit into consideration when checking for the nearest items. Therefore the distance between the check limit and the exit limit will be at least the diameter of the largest item.

In the illustration below, all items will be shuffled off to the right side of the conveyor belt. Because each item needs a clear zone, that is the shape of the *ShapeType*,

Continues on next page

8.5.7 Example: Sorting in negative y-direction
Continued

the items will be shuffled off in the order 1 to 10 as numbered in the illustration below.



xx0800000324

A	Exit
B	Check limit
C	Product flow direction
D	Sort direction
E	Enter

Example code

```
PROC Pick(num Index)
  VAR selectiondata y_sort;
  VAR robtarget draw_target;
  VAR num check_limit;

  y_sort.ShapeType:=BOX;
  y_sort.ConsiderType:=BitOr(ITEMS_TO_USE,ITEMS_BYPASS);
  y_sort.GeometricData.x:=41;
  y_sort.GeometricData.y:=160;
  y_sort.GeometricData.z:=5;
  y_sort.GeometricData.radius:=0;
  y_sort.Offset.OffsetRelation:=FRAME_COORD_DIR;
```

Continues on next page

```
    y_sort.Offset.OffsetPose.trans.x:=0;
    y_sort.Offset.OffsetPose.trans.y:=-150;
    y_sort.Offset.OffsetPose.trans.z:=0;
    y_sort.Offset.OffsetPose.rot.q1:=1;
    y_sort.Offset.OffsetPose.rot.q2:=0;
    y_sort.Offset.OffsetPose.rot.q3:=0;
    y_sort.Offset.OffsetPose.rot.q4:=0;
    check_limit:=150;

    WObjPick:=ItmSrcData{Index}.Wobj;
    GetItmTgt ItmSrcData{Index}.ItemSource,PickTarget
         \Limit:=check_limit\Selection:= y_sort;
    TriggL\Conc, RelTool(PickTarget.RobTgt, 0, 0,
         -ItmSrcData{Index}.OffsZ), MaxSpeed,
         ItmSrcData{Index}.VacuumAct1, z20, Gripper\WObj:=WObjPick;
    MoveL\Conc, PickTarget.RobTgt, LowSpeed, z5
         \Inpos:=ItmSrcData{Index}.TrackPoint, Gripper
         \WObj:=WObjPick;
    GripLoad ItemLoad;
    draw_target:=PickTarget.RobTgt;
    draw_target.trans.y:=-200;
    draw_target.rot:=[0,1,0,0];
    TriggL draw_target, LowSpeed, ItmSrcData{Index}.Ack, z20,
         Gripper\WObj:=WObjPick;
ENDPROC
```

8.5.8 Example: Indexed work area with predefined position

Description of example

In this example we use an indexed work area with predefined positions.

When using predefined positions with the indexed work area, we must modify the configuration, that is the EIO.cfg file. We will cross connect the trigger and strobe signals because with predefined positions there is no system generating the strobe signal. Without the predefined positions, the trigger signal is sent to the vision system to acquire an image. The strobe is then sent back from the vision system to acknowledge that the image has been acquired.

This is an example setup for a line that is triggered externally by an I/O signal and the position source is a predefined positions type. We recommend defining unique signal names for all new signals when setting up a system that is much different from the standard system.

Two new signals are used in this line:

- The trigger signal, `doTrigSignal`.
- The strobe signal, `diStrobeSignal`.

Modify the signal configurations by adding the two signals.

```
EIO_SIGNAL:
-Name "doTrigSignal" -SignalType "DO" -Unit "PPASIM" -UnitMap "6"
     -Access "ALL"
-Name "diStrobeSignal" - SignalType "DI" -Unit "PPASIM" -UnitMap
     "6" -Access "ALL"
```

The trigger and strobe signals are cross connected since there is no vision system that can send back a strobe signal.

```
EIO_CROSS
-Res "diStrobeSignal" -Act1 "doTrigSignal"
```

The Position generator signal in this case is `di1_1`, which is connection 1 on the DSQC 328A:X3 board, see *Circuit diagrams - 3HAC024480-008*.

When the `di1_1` goes high (by an external I/O signal) the trigger signal is pulsed. Since the trigger and strobe signals are cross connected, the strobe will be received immediately. An event will then be sent from the controller to the computer, which it is ready for new item positions and the predefined positions will then be sent to the controller. If a pattern is used, several positions are sent for every signal.

In this example the robot execution signal is not used and was therefore removed.

8.5.9 Example: Automatically generating new positions to indexed work area

Description of example

In this example we configure an indexed work area and the queue will automatically be refilled with new positions when it is empty.

The trigger and strobe signals are set up as in *Example: Indexed work area with predefined position on page 308*.

Instead of using an external input I/O signal, we will use a new simulated input I/O signal as position generator signal. This signal is set by a cross connected simulated output signal.

Two new signals are used in this line:

- The output position generator signal, `doPosGenSignal`.
- The input position generator signal, `diPosGenSignal`.

Modify the signal configurations by adding the two signals.

```
EIO_SIGNAL:
-Name "doPosGenSignal" - SignalType "DO" -Unit "PPASIM" - UnitMap
     "7" -Access "ALL"
-Name "diPosGenSignal" - SignalType "DI" -Unit "PPASIM" - UnitMap
     "7" -Access "ALL"
```

The position generator signals are cross connected.

```
EIO_CROSS
-Res "diPosGenSignal" -Act1 "doPosGenSignal"
```

`diPosGenSignal` is defined in the line as the position generator signal and `doPosGenSignal` is defined as queue idle signal.

When the queue goes empty the queue idle signal `doPosGenSignal` will go high. This cross connection will make `diPosGenSignal` go high and new positions will be pushed to the queue according to the earlier described principles.

8.5.10 Example: Optimize target release zone with real time speed of linear conveyor

Description of example

In this example the instructions `GetMaxUsageTime` and `UseReachableTargets` are used to set an optimal target release zone with a variable size.

```
PERS num pickusetime1:=0.6;
PERS num placeusetime1:=0.6;
    PROC PickPlaceSeq()
      VAR num readtime;
      readtime:=GetMaxUsageTime(ItmSrcData{PickWorkArea{1}}.ItemSource)+0.15;
        IF readtime>pickusetime1 THEN
          pickusetime1:=readtime;
        ENDIF
      UseReachableTargets ItmSrcData{PickWorkArea{1}}.ItemSource,
           TRUE,pickusetime1\ReleaseTime:=0.1;
      readtime:=GetMaxUsageTime(ItmSrcData{PlaceWorkArea{1}}.ItemSource)+0.15;
        IF readtime>placeusetime1 THEN
          placeusetime1:=readtime;
        ENDIF
      UseReachableTargets ItmSrcData{PlaceWorkArea{1}}.ItemSource,
           TRUE,placeusetime1\ReleaseTime:=0.1;

      Pick PickWorkArea{1};
      Place PlaceWorkArea{1};
    ENDPROC
```

9 Spare parts

9.1 Introduction

Overview

This section provides details about the spare parts of PickMaster 3.

9.2 Licenses

PickMaster® 3 Software

Spare part number	Description	Note
3HAC035057-001	PickMaster® 3 Software	

9.3 Camera parts

PickMaster camera

Spare part number	Description	Note
3HAC072140-001	PickMaster camera	DSQC1066 CAM-ABB-1440-GC

xx1900001574

The Basler acA1440-73gc GigE camera (including ABB firmware) with the Sony IMX273 CMOS sensor delivers 73 frames per second at 1.6 MP resolution.

For more details on the camera's installation, see the documentation on the Basler Ace website, *Basler Ace*.

PickMaster cam I/O cable

Spare part number	Description	Note
3HAC072141-001	PickMaster cam I/O cable	2000034084 (185-10031R)

xx2200000589

Continues on next page

PickMaster cam com cable

Spare part number	Description	Note
3HAC072142-001	PickMaster cam com cable	

xx2200000590

Cable GigE Cat 6, S/STP, 1x screw lock horizontal, DrC, 20 m

GigE cable for data transmission with RJ-45 plug with horizontal locking screws on the camera side at a length of 20 meter.

The twisted, shielded cable has an RJ-45 click-lock plug on the host side and is suitable for drag chain applications.

Camera mount adapter

Spare part number	Description	Note
3HAC074680-001	Camera mount adapter	

Camera mount for Basler ace cameras.

For mounting the camera onto tripod threads.

PickMaster camera

Spare part number	Description	Note
3HAC035020-001	Camera Color Resolution 1296 x 966	CAM-ABB-SCT-SXC

Continues on next page

xx2200002026

Basler Scout scA1300-32gc (including ABB firmware): Std resolution (resolution 1296x966, sensor size 1/3”) digital CCD camera.

PickMaster I/O cable

Spare part number	Description	Note
3HAC061656-001	Cable Power-I/O	2000034085

xx2200002027

Continues on next page

9.3 Camera parts
Continued

Spare part number	Description	Note
3HAC031721-001	Adapter Cable, 12p HRS/24V , open, 10m	2000026632/2000022909 (185-1094R)

xx2300000344

Camera Ethernet Cable

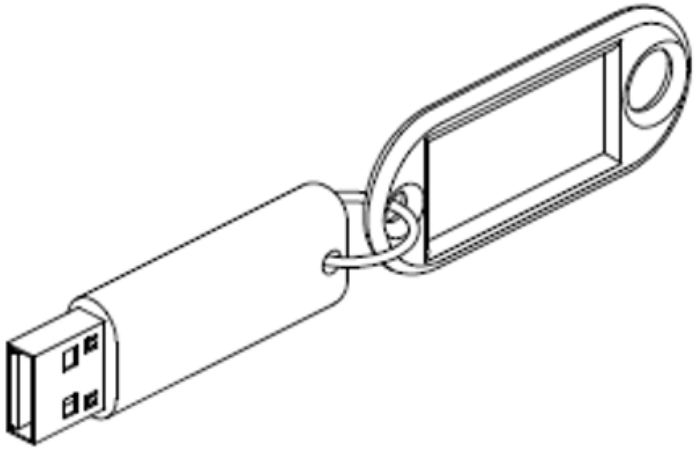
Spare part number	Description	Note
3HAC031806-001	Camera Ethernet Cable	

xx2200002028

9.4 USB dongle parts

Spare part

Spare part number	Description	Note
3HAC036512-001	Software Upgrade for color camera	
3HAC031782-001	USB License (without color tools)	
3HAC036252-001	USB License (with color tools)	
3HAC039556-001	Vision simulation license for upto 10 simulated cameras	

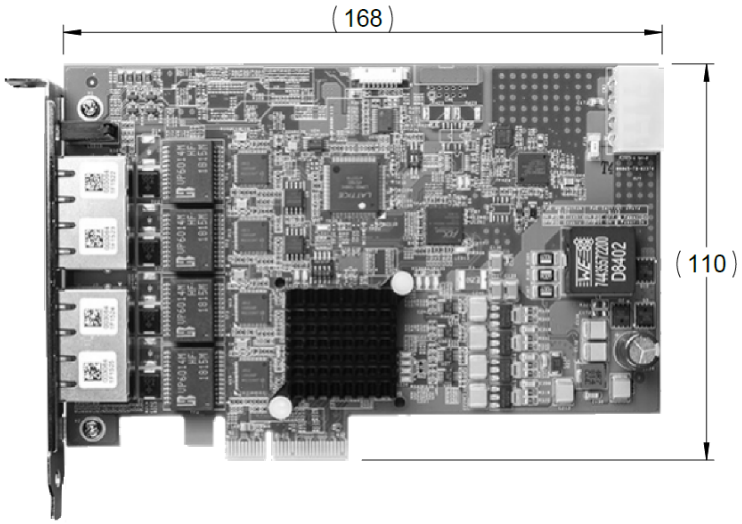


xx1900001747

9.5 GigE Network card parts

Spare part

Spare part number	Description	Note
3HAC078753-001	GigE network card	DSQC1083 Gig ethernet card



xx2200000591

Spare part number	Description	Note
3HAC069300-001	Gig ethernet card	DSQC1065 Gig ethernet card

10 Circuit diagram

10.1 Circuit diagrams

Overview

The circuit diagrams are not included in this manual, but are available for registered users on myABB Business Portal, *www.abb.com/myABB*.

See the article numbers in the tables below.

Controllers

Product	Article numbers for circuit diagrams
Circuit diagram - IRC5	*3HAC024480-011*
Circuit diagram - IRC5 Compact	*3HAC049406-003*
Circuit diagram - IRC5 Panel Mounted Controller	*3HAC026871-020*

RobotWare options

Product	Article numbers for circuit diagrams
Circuit diagram - PickMaster 3	*3HAC024480-008*

This page is intentionally left blank

11 Cyber security for PickMaster system networks

11.1 Introduction

About this chapter

This chapter discusses security in a network installation with PickMaster systems.

First, it gives an overview about PickMaster, IRC5 controllers and other products typically deployed in a network installation and the communications between them. A security analysis identifies the most critical assets and security threats targeting them. To mitigate these security threats, every PickMaster implements UAS (User Authorization System).

Security features, however, such as UAS and testing, are only components of an effective security strategy. It is equally important to define, implement, and maintain a security policy, which covers security processes, procedures, and mechanisms. This chapter lists requirements for such a security policy.

Section *PickMaster application protocols on page 328* contains descriptions of the communication protocols used by IRC5 products, which may be useful for configuring elements of the system security architecture, as well as a list of useful security tools.

Disclaimer

The intent of this chapter is to raise awareness about critical security threats and to provide guidance to address them as well as to inform how ABB is working on security assurance. However, due to the high number of different security risks and complex dependencies within actual installations, this document can neither cover all possible security risks, nor guarantee the success of the presented security mechanisms.

The benefits and risks of using open networking technology for PickMaster systems

ABB IRC5 products use standard internet transport protocols, TCP and UDP. This means that PickMaster products can be connected to a normal network (TCP/IP/Ethernet) like any other computer or network product, which reduces costs and unifies network management. Furthermore, the interconnection of control systems and "office" systems, such as ERP, enable a wide range of new applications, which take advantage of such vertical integration from the shop floor up to the enterprise management. Section *Network architecture and communication on page 323* describes a typical PickMaster network.

However, the direct connection of control systems to the plant network also creates security risks (for example, malware infections (viruses, worms, Trojans), denial of service, disclosure of confidential data). Section *Security analysis on page 325* discusses these security threats in detail.

Continues on next page

Mitigating the risks through a comprehensive security policy and architecture

It is generally accepted that the security features of a product or system are only one part in a successful protection strategy. It is equally important to define, implement, and maintain an effective security policy, which covers (among other issues) risk analysis, procedures, responsibilities, and regular auditing. Section *Security policy on page 327* discusses general and PickMaster specific requirements for a security policy and shows, how such a security policy can be used to mitigate critical security threats targeting a robot control system. It is important to note though that security cannot be achieved by a one-time investment in a product or process but requires ongoing effort to operate and maintain.

11.2 Network architecture and communication

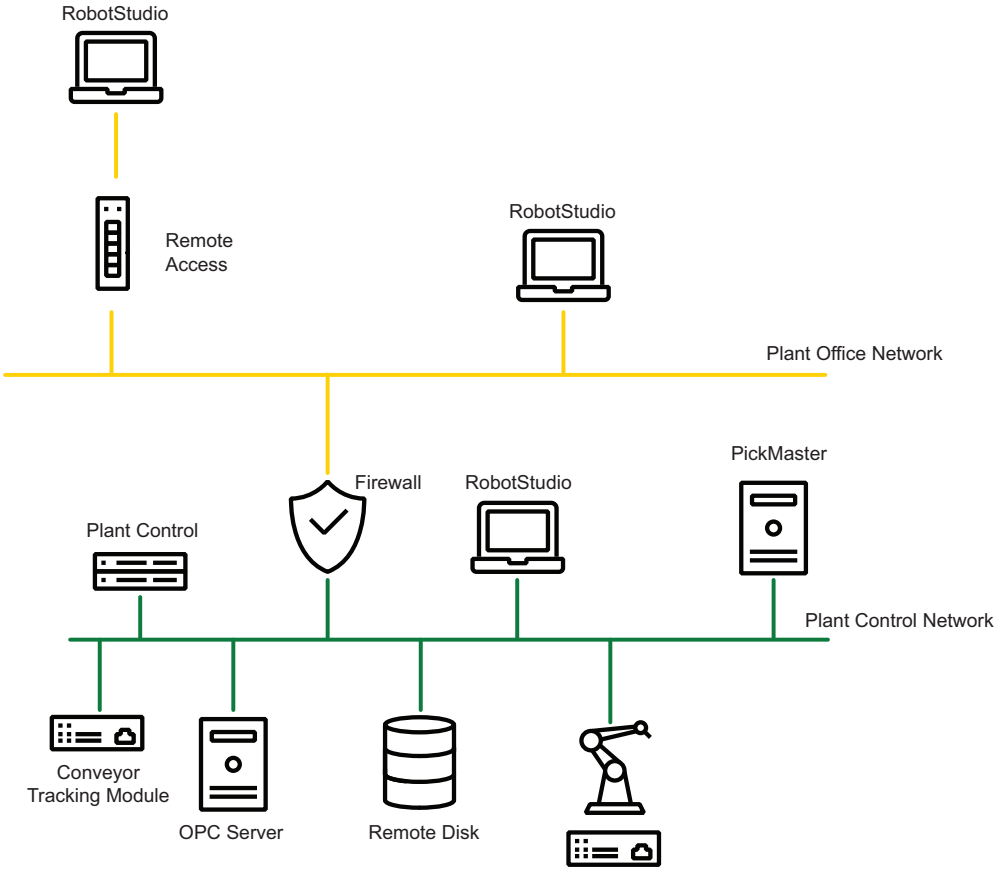
Introduction

This section gives an overview about typical components of an PickMaster installation, which are attached to the network, and the communication between them. Furthermore, this section serves as basis for the threat analysis (*Security analysis on page 325*) and the requirements to a security policy (*Security policy on page 327*).

Simplified network with PickMaster systems

The following figure shows a simplified network with PickMaster. The intention of this picture is to show typical components, which may be part of a robot control system installation, and where they may be installed.

The objective of the shown network security architecture is to protect the plant control network from threats, which are originating in the plant office network and especially in remote networks and which are caused for example by viruses and worms. Therefore, it is strongly suggested to separate the plant control network and the plant office network with a protection device, such as a firewall.

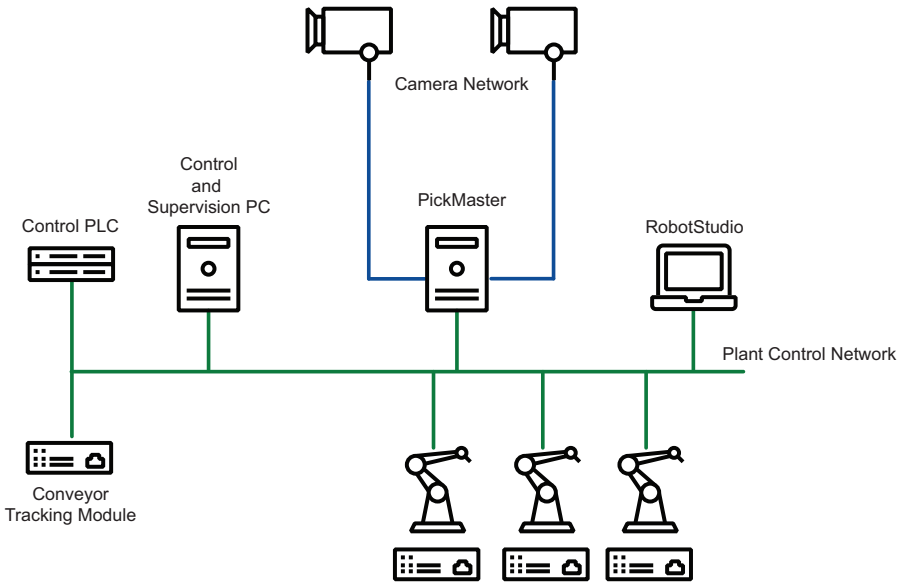


xx2300000611

Continues on next page

Network with PickMaster

The Plant Control Network architecture details with the PickMaster product is slightly different and has other security aspects as shown in the following figure. In a PickMaster installation, the cameras are connected directly to the PC or via a switch to the PC over a separate network than the Plant Control Network.



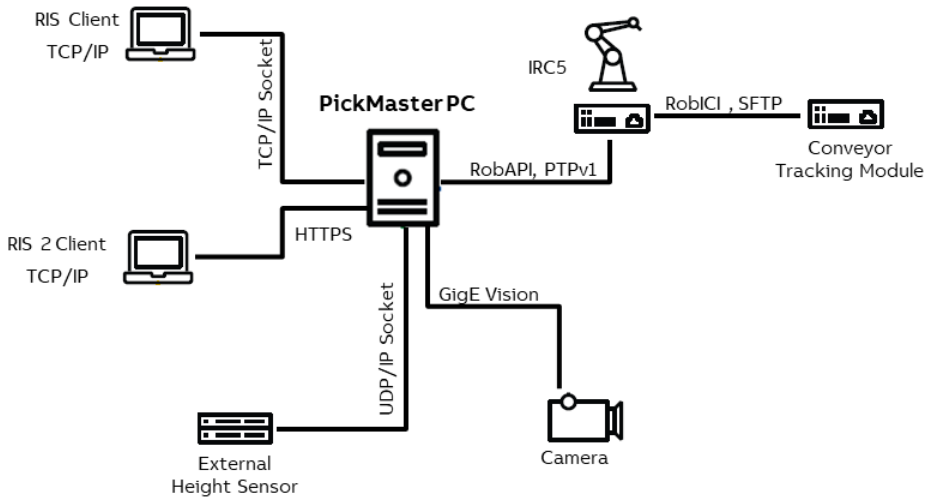
xx2300000617

Communication protocols between PickMaster related products

The following figure shows all components, which are involved in communication around PickMaster products, and the communication between them.

The components are clustered according to where they reside: remote/plant office network or plant control network.

The lines between the components indicate the used communication protocol. Note that the RobAPI protocol is a combined TCP/IP and FTP protocol. For more details, see *RobAPI application protocol on page 328*.



xx2300000626

11.3 Security analysis

For detailed Security analysis, refer the Security analysis described in the manual *Operating manual - IRC5 Integrator's guide*.

11.4 PickMaster User Authorization System

For details on UAS refer *Application login window on page 55*.

11.5 Security policy

For detailed Security policy, refer the Security policy described in the manual *Operating manual - IRC5 Integrator's guide*.

11.6 PickMaster application protocols

Used protocols

The following table lists the used communication protocols between PickMaster system, their port numbers ("any" represents any port ≥ 1024 and "assigned" is a port, which has been defined (at runtime) by the communication partner; arrow direction denotes direction of session establishment), and their usage:

Protocol	Type(s)	Port (client, server)	Client(s)	Server(s)	Usage
NetScan	UDP	broadcast: any → 5512 broadcast response: 5513 ← any static handling: any → 5514	PickMaster RobotStudio	IRC5 Robot Controllers	Detection of available IRC5 Robot Controllers
FTP	TCP	active: any → 21 assigned <- 20 passive: any → 21 any+1 → assigned	PickMaster RobotStudio	FTP Servers IRC5 Robot Controllers IPS	File transfer Configuration Software upgrade
RNP	TCP	any → 5515	PickMaster RobotStudio	IRC5 Robot Controllers	Communication with IRC5 Robot Controller
RIS (Remote Integration Services)	TCP	Any → assigned	Other clients	PickMaster	Default port for TCP is 1700
RIS2	TCP	Any → assigned	Other clients	PickMaster	Default port is 50000
PTP v1	PTP (UPD/IP)	any -> 319 any ->320 any <- 319 any <-320	IRC5 Robot Controllers	PickMaster	Time synchronization according to IEEE 1588
Sentinel License Manager service	TCP UDP	localhost -> 1947	Cognex Security Service	Sentinel License Manager	Cognex USB-key license management

RobAPI application protocol

RobAPI uses three different protocols:

- NetScan for discovering industrial robot controllers available on the network
- RobAPI network protocol (RNP) for sending commands to and receiving events from robot controllers
- FTP for file transfer (see *FTP on page 329*)

NetScan is an application protocol built on UDP/IP for discovering industrial robot controllers available on the network and for collecting basic data. The data collected

Continues on next page

by NetScan includes version info and unique id of the software loaded on the controller.

The PC-side implementation of NetScan sends out requests, to which all active controllers respond. Both requests and responses are UDP broadcasts. The requests are sent to UDP port 5512, and the responses are sent to UDP port 5513.

PC-NetScan also supports the option to configure specific "non-local" IP-addresses of controllers, i.e., controllers not on the local subnet. This makes it possible to discover and connect to remote controllers, e.g., over the Internet. The PC-side NetScan sends out UDP unicast requests to port 5514, and the controller responds with UDP unicasts to the source address with the (dynamically allocated) source port.

The RobAPI network protocol (RNP) is an application protocol built on TCP/IP for communicating with industrial robot controllers over a network. The protocol uses by default TCP port 5515. The RobAPI network protocol is event-driven, i.e. RobAPI clients do not need to poll for I/O changes on the controller. Events are sent to the PC-side, whenever a change has occurred. However, only the PC establishes TCP connections.

The RobAPI network protocol has an are-you-alive (AYA) mechanism. With AYA it is possible for the robot controllers to detect unexpected disconnection of clients and for clients to detect unexpected disconnection of robot controllers. AYA messages are sent every 5 second from both the client and the server.

For file transfers, RobAPI uses standard FTP.

FTP

RobAPI Network Protocol uses FTP internally for file transfer. The FTP server of the IRC5 controller supports both active and passive FTP. However, if RobAPI is acting as FTP client, it uses only passive mode.

FTP is also used independently of RobAPI, for file transfer and file system access to the Main Computer and IPS. This requires logging in with a user name and password that is defined for the controller, see *PickMaster User Authorization System on page 326*.

Note

The controller supports anonymous FTP login on the LAN/Service Port with any user name and password to access files on the controller.

The anonymous access is read only and the access is limited to certain directories located in the RobotWare installation.

Any attempt to read or write to any other location will be denied.

RIS

RIS is the Remote Integration Services proprietary protocol based on TCP/IP to provide bi-directional communications between a command and control program on a PC or other device to the PickMaster product running on a Windows PC. Port 1700 is the default port but may be configured in PickMaster.

Continues on next page

RIS2

RIS2 is Remote Integration Service 2. It provides access for HMIs and other devices that can communicate through HTTPS. Port 50000 is the default port but may be configured in PickMaster.

Allowing incoming connections from Public Networks using port 1947

The run-time environment installer adds a firewall rule named “Sentinel License Manager” that allows incoming connections from private networks using port 1947.

You can manually allow access from public networks using this port, but it is not recommended. If you do plan to allow incoming connections from public networks using port 1947, create a rule with a different name in order to prevent future RTE upgrades from removing this access.

Index

ABB AB
Robotics & Discrete Automation
S-721 68 VÄSTERÅS, Sweden
Telephone +46 (0) 21 344 400

ABB AS
Robotics & Discrete Automation
Nordlysvegen 7, N-4340 BRYNE, Norway
Box 265, N-4349 BRYNE, Norway
Telephone: +47 22 87 2000

ABB Engineering (Shanghai) Ltd.
Robotics & Discrete Automation
No. 4528 Kangxin Highway
PuDong New District
SHANGHAI 201319, China
Telephone: +86 21 6105 6666

ABB Inc.
Robotics & Discrete Automation
1250 Brown Road
Auburn Hills, MI 48326
USA
Telephone: +1 248 391 9000

abb.com/robotics

3HAC031978-001, Rev V, en