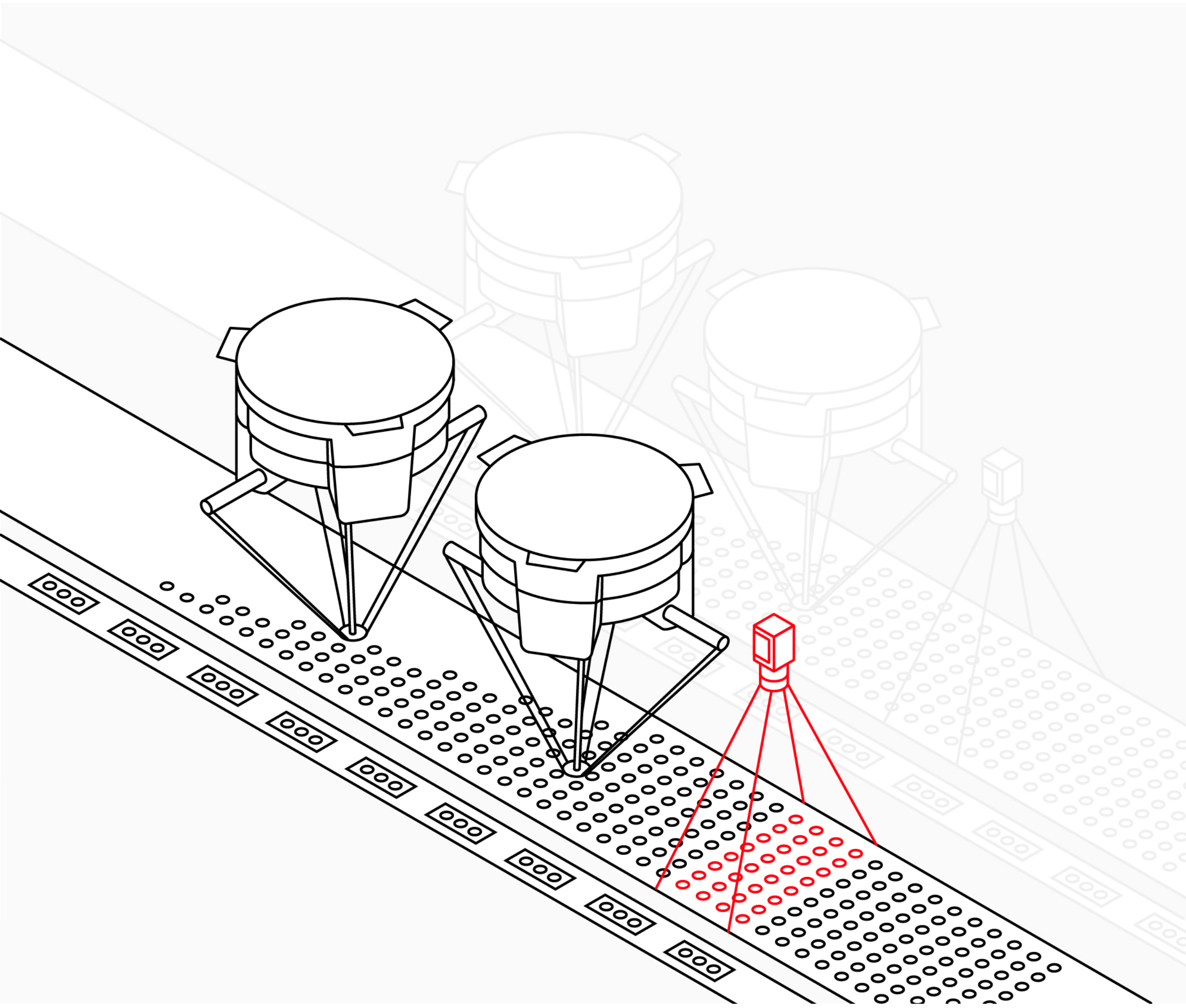
ROBOTICS

Application manual

PickMaster® Twin - PowerPac



Trace back information:
Workspace Main version a589
Checked in 2024-05-28
Skribenta version 5.5.019

Application manual

PickMaster® Twin - PowerPac
Release 2.4

OmniCore and IRC5

Document ID: 3HAC080435-001

Revision: H

Keep for future reference.

Additional copies of this manual may be obtained from ABB.

Original instructions.

Table of contents

This page is intentionally left blank

Overview of this manual

About this manual

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

Note

All safety information for working with the controller is described in the product manual for the controller.

Tip

The function description may differ via the software version.

Always read the corresponding manual version to match with the software version.

Usage

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

PickMaster PowerPac is intended for use as an engineering tool on a portable laptop PC for offline use and online connection to a host computer in the installation for commissioning purposes. PickMaster PowerPac is not intended for use on the host computer under production conditions.

Who should read this manual?

This manual is intended for:

- Installation personnel
- Programmers
- Integrators
- 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.

Continues on next page

Cybersecurity

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.

Note

Only qualified personnel should write or modify the script files.

It is the responsibility of the writer to make sure that the cell is safe when running with the script files.

The PickMaster PowerPac will use the following ports:

- 50000 (PickMaster Runtime communication)
- 9000 (Zenon event port)
- 6001 (PickMaster Runtime communication)
- 502 (For remote control with Modbus)

References

Tip

All documents can be found via myABB Business Portal, *www.abb.com/myABB*.

OmniCore

Reference	Document ID
Application manual - PickMaster Twin - Operator	*3HAC069977-001*
Product specification - PickMaster® Twin	*3HAC073650-001*
Circuit diagram - PickMaster Twin	*3HAC024480-020*
Safety manual for robot - Manipulator and IRC5 or OmniCore controller [i]	*3HAC031045-001*
Application manual - Conveyor tracking	*3HAC066561-001*
Product manual - OmniCore C30	*3HAC060860-001*
Product manual - OmniCore C90XT	*3HAC073706-001*
Operating manual - OmniCore	*3HAC065036-001*
Operating manual - Integrator's guide OmniCore	*3HAC065037-001*
Application manual - Controller software OmniCore	*3HAC066554-001*

Continues on next page

Reference	Document ID
Technical reference manual - Event logs for RobotWare 7	*3HAC042927-001*
Technical reference manual - Lubrication in gearboxes	*3HAC042927-001*
Technical reference manual - System parameters	*3HAC065041-001*

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

IRC5

Reference	Document ID
Application manual - PickMaster Twin - Operator	*3HAC069977-001*
Product specification - PickMaster® Twin	*3HAC073650-001*
Circuit diagram - PickMaster Twin	*3HAC024480-020*
Operating manual - RobotStudio	*3HAC032104-001*
Application manual - Conveyor tracking	*3HAC050991-001*
Product manual - IRC5	*3HAC047136-001*
Product manual - IRC5 Panel Mounted Controller	*3HAC027707-001*
Operating manual - IRC5 with FlexPendant	*3HAC050941-001*
Operating manual - IRC5 Integrator's guide	*3HAC050940-001*
Operating manual - Troubleshooting IRC5	*3HAC020738-001*
Technical reference manual - RAPID Instructions, Functions and Data types	*3HAC050917-001*
Technical reference manual - RAPID Overview	*3HAC050947-001*
Technical reference manual - System parameters	*3HAC050948-001*

External references

Reference	Description
Cognex Ethernet Camera Tool	For configuring camera networks.
Gigabit Ethernet Performance Driver	For camera communication.
aca1440-73gc	Information about Basler Ace Gigabit Ethernet cameras and the switch for Gigabit Ethernet cameras. **Note** This camera uses ABB customized firmware, which needs to be purchased from ABB.
sca1300-32gc	Information about Basler Scout Gigabit Ethernet cameras and the switch for Gigabit Ethernet cameras. **Note** This camera uses ABB customized firmware, which needs to be purchased from ABB.
CognexPCConfigGuide	Detailed information about PC requirements for the vision system.

Continues on next page

Revisions

Revision	Description
A	First edition.
B	Released with PickMaster® Twin 2.0.1. • Added Histogram and Caliper function. • Added gripper related with TCP0 function. • Added troubleshooting for Image Dialog cannot show in section *The Image Dialog cannot show on page 465*. • Minor corrections.
C	Released with PickMaster® Twin 2.1. • Supported multiple languages. • Updated trigger distance function. • Added adjust base frame function. • Added user script function. • Added PMRT login function when connecting to PMRT. • Updated information for circular conveyor calibration. • Added copy function for Item, Container and Flow. • Minor corrections.
D	Released with PickMaster® Twin 2.1.1. • Minor corrections.
E	Released with PickMaster® Twin 2.2. • Supported external sensor function. • Added Self-signing certificate. • Supported multiple languages. • Minor corrections.
F	Released with PickMaster® Twin 2.3. • Supported Runtime file transfer function. • Updated time synchronization service. • Updated PickMaster PowerPac license. • Added 2.5D vision. • Minor corrections.
G	Released with PickMaster® Twin 2.3.1. • Added **Info** page under **File** ribbon tab. • Updated network setting for IPC. • Changed ABB Ability to ABB Connect. • Minor corrections.

Continues on next page

Revision	Description
H	Released with PickMaster® Twin 2.4 • Added vision model preview function for geometric model and inspection model with geometric alignment model. • Added copy as inspection model function for vision models. • Added Advanced function chapter which contains User script, External sensor, Working with products of varying height (2.5D vision) and Production with flow(Ghost Picking). • Added Python version limitation to 3.9.5. • Added Time out setting for User script. • Improved Runtime error codes. • Updated Troubleshooting chapter structure. • Updated User script chapter structure and content. • Updated **Disable vision system** limitation from using external version system to using Basler camera through external sensor interface in Setting PickMaster options and external sensor setting. • Updated PickMaster Twin installation procedure. • Updated PickMaster Twin Hardware connection illustration. • Updated occupied ports in Cybersecurity. • Updated Pack&Go file path default folder to C:\Program-Data\ABB\PickMaster Twin\PackedSolutions. • Updated the Terms for Client and Host. • Supported clockwise rotation of circular conveyor throughout the manual. • Minor corrections.

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® Twin products

- When using with PickMaster® Twin 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® Twin products.
- The integrator installing the PickMaster® Twin 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® Twin 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® Twin products. PickMaster® Twin 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

- Safety related status and operations shall be handled on the controller and by safety rated systems. PickMaster® Twin 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® Twin can never be used to replace A-stop/E-stop or any other safety related stops.

This page is intentionally left blank

1 Welcome to PickMaster PowerPac

1.1 Introduction

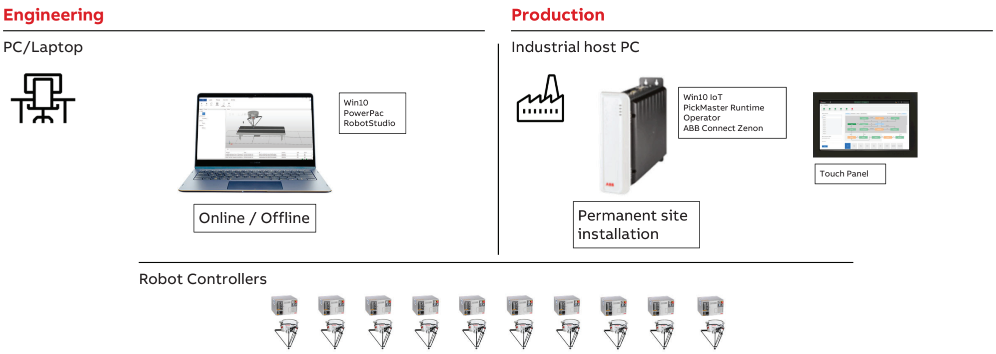
About PickMaster® Twin

PickMaster® Twin is an application product designed for vision based high speed picking of random flow products on the fly. PickMaster® Twin supports ease-of use configuration, simulation and operation of a big variation of smaller or larger line layouts composed of a multitude of robots, cameras, conveyors and fixed work areas. It is a production system that comprises all steps in the life cycle of a picking installation from proposal, engineering, commissioning, operation to maintenance and support.

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

- With the integrated vision system 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.

PickMaster® Twin is a modular product for controlling ABB robots in picking applications through the robot controller. It is configurable to perform pick and place operations of items. A vision system is used to find randomly placed items on conveyor belts or indexing static work areas. PickMaster PowerPac is the engineering software aimed at configuring and validating the application in offline simulation with a virtual system and in online mode directly connected to the real installation. It uses comprehensive graphical interfaces to configure powerful applications, where it can control multiple robots picking and placing sensor-detected items on different conveyor belts.



xx2100001619

PickMaster® Twin comprises the following modules:

PickMaster® PowerPac

Ease of Use software for offline and online configuration and commissioning in a visual 3D environment, powered by RobotStudio™.

Continues on next page

PickMaster® Operator

State-of-the art user interface for operating PickMaster on the shop floor, built on ABB Connect Zenon data management software.

PickMaster® Runtime

Efficient runtime operation software for orchestrating the coordination of the packaging process for a multitude of robots and conveyors including integrated vision software for precise robot guidance and quality inspection.

- Virtual Runtime: running the PickMaster process in a simulated virtual environment on a client system connected to virtual robot controllers.
- Real Runtime: running the PickMaster process in the real production installation on the Host computer connected to real robot controllers.

Note

PickMaster® Twin is delivered with different hardware configurations. For more information, see *Product specification - PickMaster® Twin*.

About PickMaster PowerPac

This manual describes how to install and use PickMaster PowerPac as the engineering software for two modes:

- Configuring and validating the application in offline simulation with a virtual system.
- Commissioning in online mode directly connected to the real installation.

This manual also describes the components of the real system, their installation, configuration and calibration.

1.2 PickMaster PowerPac terms

About these terms

Some words have a specific meaning when used in this manual. Definitions of these words in this manual 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
PickMaster PowerPac	The market name of PickMaster PC engineering software that is used for simulating and commissioning picking lines with virtual and real Runtime.
PickMaster Operator	The market name of PickMaster production operator interface software that is used for running PickMaster applications in production. PickMaster Operator can read and write to solutions generated by PickMaster PowerPac. It has access to real Runtime.
PickMaster Virtual Runtime (VRT)	The core engine that orchestrates all the calculation of virtual pick and place operation in simulations.
PickMaster Real Runtime (RRT)	The core engine that orchestrates all the calculation of pick and place operation in real product. Runtime communicates with cameras and the robot controllers. It's also called as Runtime.
PickMaster Twin Client	It's the Client computer for configuring, simulating, and commissioning a PickMaster PowerPac solution. The PickMaster Twin Client installation package shall be installed on the Client computer. It contains PickMaster PowerPac, PickMaster virtual Runtime and PickMaster real Runtime.
PickMaster Twin Host	It's the Host computer for operating and managing PickMaster Twin in production. The PickMaster Twin Host installation package shall be installed on the Host computer. It contains PickMaster Operator and PickMaster real Runtime.
Solution	Format for storing a PickMaster Twin configuration result.
Recipe	Format and a collection of parameters regarding the process of Pick and Place for storing the process to be executed in a station.
Layout	Description of static objects in a PickMaster installation, for example robots, *work areas*.
Process	Description of a PickMaster picking process and all items, containers, flow and recipes.
Work area	A defined picking and placing area for the robots.
Item	The generic term for a specific object to be picked or placed in a PickMaster PowerPac application.
Container	Defines a shape that can set specific patterns and what *items* to use for each position in the patterns.
Position generator	Defines the sensor configuration on the conveyor and indexed work area.

Continues on next page

Term	Definition
Emulation	An activity of imitating the behavior of real cell or line and display the activity on screen.
Ghost picking	A kind of dry run, when production uses recorded virtual items to pick, thus no real item to pick.
Offline Simulation	Simulation process when connected to the virtual robot.

2 Installation

2.1 PickMaster package

Concepts of using PickMaster Twin

PickMaster PowerPac is designed to be installed on a laptop computer that can host solutions for many different installations that can be connected for commissioning, new recipe introduction, maintenance and servicing purposes to several physical installations, where each one of those have their own permanent host computer.

There are two software installation packages: PickMaster Twin Client for the portable engineering system and PickMaster Twin Host for the permanent factory system.

The Client installation does not require any physical equipment installations. All physical component installations, configurations and calibrations are done on the Host system.

PickMaster Twin Client

The installations package for PickMaster Twin offline configuration, simulation and testing is named as PickMaster Twin Client. It installs the following softwares:

- PickMaster PowerPac
- PickMaster Virtual Runtime
- PickMaster Real Runtime

Note

This package is only intended for engineering and not for the final factory production installation. The ability to switch to real Runtime on the same computer is only intended for test purposes and it can be used for creating and editing vision models offline. A vision demo dongle can be used for this purpose.

Software Installation Package

Registered ABB customers can download the latest version of the PickMaster Twin Client and the user documentation for PickMaster PowerPac from the ABB download center.

Tip

The download center address is https://new.abb.com/products/robotics/application-software/pickmaster.

Note

The PickMaster software is available in 64-bit version.

Continues on next page

When the PickMaster Twin Client is installed successfully, the user documentation for PickMaster PowerPac and the calibration papers are available in the installation folder *Documentation*.

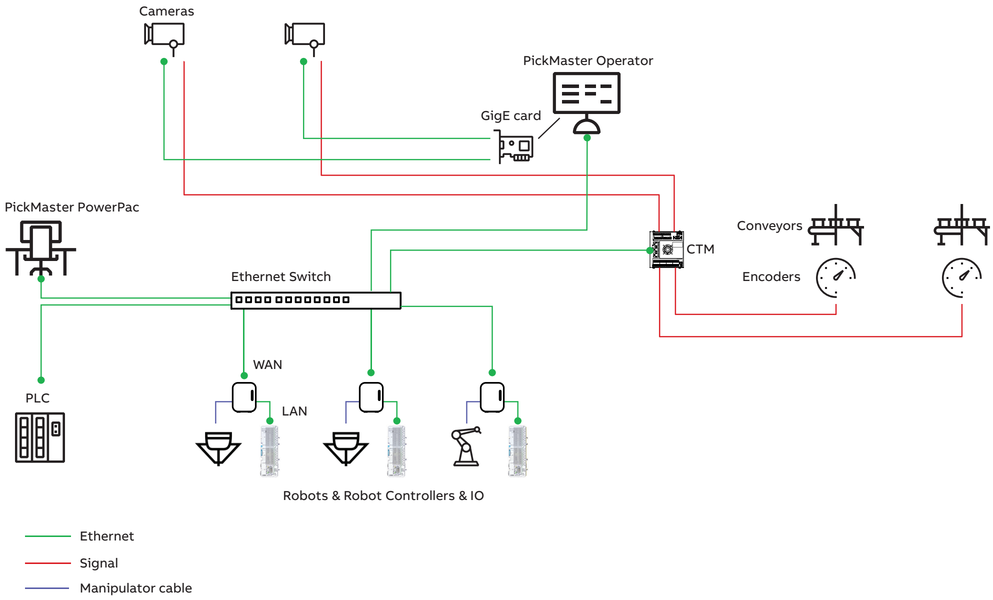
Note

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

2.2 PickMaster® Twin Hardware connection illustration

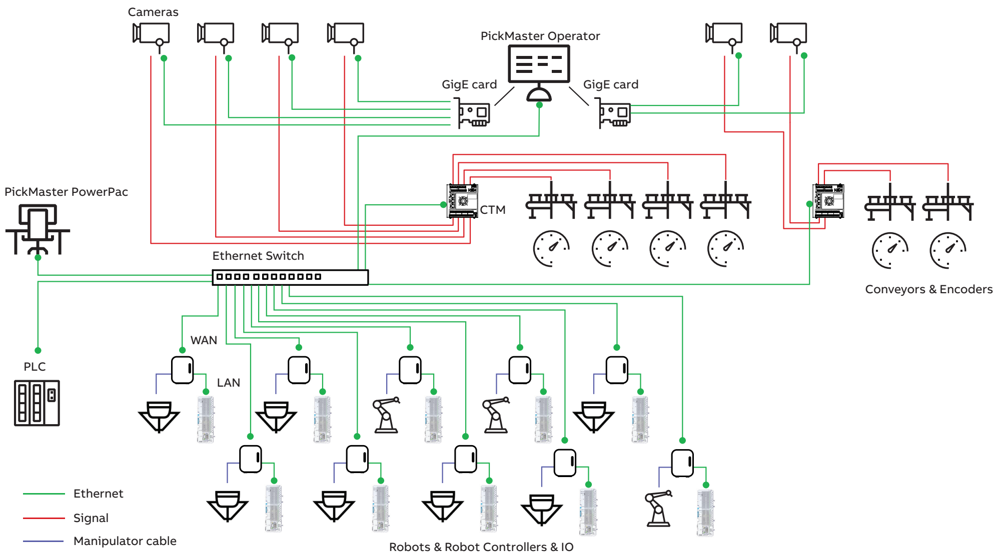
Example

The following illustration is showing an installation example with 3 robots, 2 cameras and 2 conveyors.



xx1900001746

The following illustration is showing an installation example with 10 robots, 6 cameras and 6 conveyors.



xx2400000560

2.3 System requirements

2.3.1 Hardware and software requirements

Hardware and software requirements for PickMaster Twin Client

Hardware requirements

Following are the hardware requirements:

- A log on account with administrator rights on the computer.
- CPU: 2.0 GHz or faster processor. Multicore processor is recommended.
- Memory: 8 GB RAM is the minimum requirement if running Windows 64bit edition. 16 GB or more is recommended if working with vision or heavy CAD models.
- Free disk space: 10+ GB free space, solid state drive (SSD) recommended.
- Graphics card: High-performance, DirectX 11 compatible, gaming graphics card from any of the leading vendors. For the Advanced lightning mode Direct3D feature level 10_1 or higher is required.
- Display settings: 1920 x 1080 pixels or higher resolution is recommended.
- Mouse: Three-button mouse
- If robot movement can be initiated from an external control panel then an emergency stop must also be available.

Note

When running the software, close other software that consumes a lot of memory, otherwise it will affect the software normal use.

Software requirements

Following are the software requirements:

- Windows 10 (64 bit)
- Acrobat reader
- RobotStudio 2024.2
- Omnicore with RobotWare 7.12
- IRC5 with RobotWare 6.15

Hardware and software requirements for PickMaster Twin Host

Recommended hardware

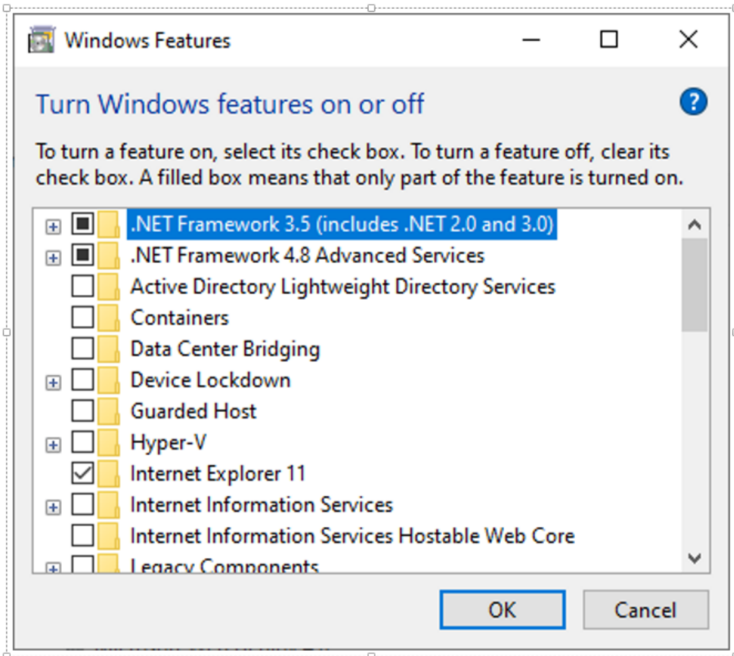
- Windows 10 (64 bit) IPC, 2GHz, 500 GBit SSD, 8 GBit RAM
- Memory: 8 GB RAM is the minimum requirement if running Windows 64bit edition. 16 GB or more is recommended if working with vision or heavy CAD models.
- Recommended 23 inches 1920x1080 multi-touch screen
- Minimum two USB slots, one Ethernet port and one free PCI Express slot for a 168 mm x 110 mm size PCIE card

Continues on next page

- Ethernet switch (robot network)

Software requirements

- Microsoft Windows 10, 64 bit (Home, Pro, Enterprise, Education, IoT, x64 versions) for touch panel
- Environment Requirement : .Net Framework 3.5



xx2300001704

- RobotStudio 2024.2
- Omnicore with RobotWare 7.12
- IRC5 with RobotWare 6.15

2.3.2 Ethernet switch

Overview

An Ethernet switch is used to connect the PC with multiple robot controllers. It is recommended to use an industrial switch with a communication speed of 1000 Mbit/s or higher.

2.3.3 Vision system

Overview

PickMaster PowerPac 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 Runtime PC. Trigger/Strobe and power voltage is connected to a Hirose 12-pin/6-pin connector on the camera housing. 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 Runtime 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).

Color vision

Color vision is available as a standard function and has the following features:

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

Note

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

For the validated cameras, see *Validated cameras on page 60*.

Trigger strobe cables

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

Cable	ID	Description
New 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
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

Continues on next page

Trigger strobe connection for 2000034085 (Basler 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 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
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

Continues on next page

Pin Number	Wire Color	scout GigE	Function
12	Red/Blue	I/O Output 4	Not used

Trigger strobe connection for 2000034084 (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

2.3.4 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 any orientation to the image area.

Lighting

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

Other camera requirements

A PickMaster camera needs to be of type progressive scan (non-interlaced) as it is used to record images of objects on a moving belt.

A PickMaster camera also needs to support electronic shutter control. With this feature it is possible to set the exposure from PickMaster PowerPac, 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 Gigabit Ethernet cameras, see *References on page 10*.

Recommendation for lenses

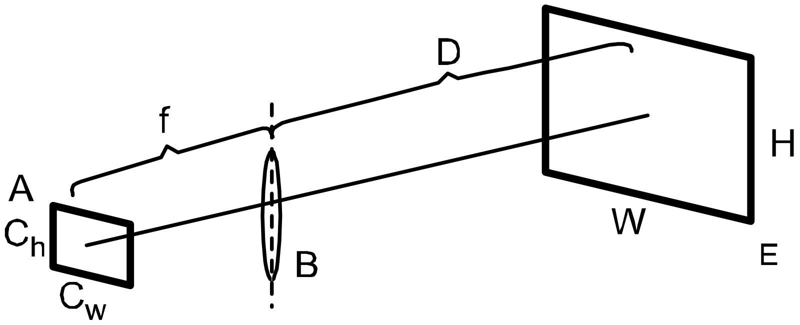
When planning a cell it is important to choose a suitable camera/lens setup that gives an appropriate field of view (FOV).

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

The graphic below shows the geometry of the optical setup.



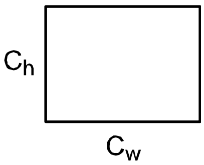
xx0900000550

A	Sensor chip
B	Lens
Ch	Chip height (mm)
Cw	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.

- 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.

2.4 PickMaster PowerPac license

Introduction to licensing

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

PickMaster PowerPac depends on the activation of RobotStudio. You can use PickMaster PowerPac normally only if you activate RS with a license that includes the PickMaster PowerPac option. It can also be activated separately from PickMaster PowerPac, but still invokes the RS activation procedure.

PickMaster PowerPac license options

Two license options are available for PickMaster PowerPac, Basic and Premium.

Users can obtain the Basic option for free and work with limited functions. The Basic option only allows you to calibrate and simulate the existing solutions, and cannot add new components under the Layout, for example, the conveyors, controllers.

The Premium option provides more functions for professional integrators and commissioners.

Function comparison between license options

The following table lists the main application scenarios and differences between three license options.

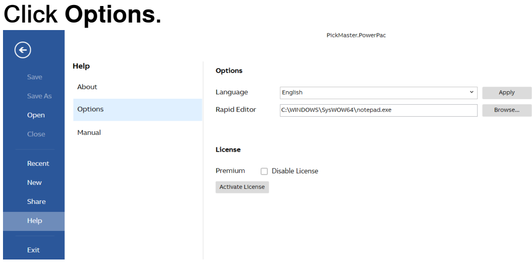
Function		Premium		Basic		Free	
		Robotstudio License Activated	PickMaster PowerPac License Activated	Robotstudio License Activated	PickMaster PowerPac License Unactivated	Robotstudio License Unactivated	PickMaster PowerPac License Unactivated
Open solution		x		x		x	
Layout Edit (Controller,Gripper,Conveyor...)		x					
Process Edit (Item/Container/Flow/Recipe)		x		x			
Pack&Go		x		x		x	
Pack as Template		x		x		x	
Save		x		x		x	
Save as		x		x		x	
Operation-Start (Production/Simulation/Emulation)	Production	x		x			
	Simulation	x		x			
	Emulation	x		x			
Operation-Stop		x		x			
Create solution		x					
Unpack		x		x		x	

Continues on next page

Function		Premium		Basic		Free	
		Robotstudio License Activated	PickMaster PowerPac License Activated	Robotstudio License Activated	PickMaster PowerPac License Unactivated	Robotstudio License Unactivated	PickMaster PowerPac License Unactivated
Calibration		x		x		x	
Reset		x		x		x	
Recording		x		x		x	
Control-Start (Production/Simulation/Emulation)	Production	x		x			
	Simulation	x		x			
	Emulation	x		x			
Connect to RT		x		x		x	
Language switch		x		x		x	
Manual		x		x		x	
Vision Configuration		x					
Digital Twin		x					
User script		x					
External sensor		x					

Information about the current license

Use the following procedure to get information about the current license.

	Action	Note
1	Start the PickMaster PowerPac.	
2	Click the **File** tab.	
3	Click **Help**.	
4	Click **Options**.  xx2300001444	

Activating a license key

Activating a license key automatically over the Internet

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

	Action
1	To start the licencing application, either use: • In the PickMaster PowerPac, on the **Options** menu, click **Activate License**.
2	Under **Standalong License**, choose **I want to Activate a standalong license key** and click **Next**.

Continues on next page

	Action
3	Under **Automatic Activation**, choose **Activate RobotStudio over the internet** and click **Next**.
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 PowerPac license will be activated immediately, and your PickMaster PowerPac is ready for use when started next time.

Activating a license key manually

If the computer with PickMaster PowerPac 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`).

Use this procedure to activate a PickMaster PowerPac license manually.

	Action
1	To start the licensing application either use: • In the PickMaster PowerPac, on the **Options** menu, click **Verify License**.
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 PowerPac PC.
11	On the PickMaster PowerPac 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 for the PickMaster PowerPac and the Runtime, and the PickMaster installation ready to use.

2.5 Self-signing certificate

2.5.1 Certificate handling

Default self-signed certificates

PickMaster® Twin products support the use of X.509 certificates for secure communication over the network. The PickMaster® Runtime generates self-signed X.509 certificates by default for PickMaster® PowerPac and PickMaster® Operator. The generated self-signed certificate has an RSA key pair with a key length of 2048 bits.

Certificate replacement

To enhance the security of the system and to assure that data is being transmitted over a secure connection, it is recommended to replace the self-signed certificates on the PickMaster® Runtime with your own certificates. This provides added security and the ability to use your own trusted certificate chain.

To replace a self-signed certificate, export your desired certificate and private key and replace the certificate in PickMaster® Runtime. It is important to follow the proper procedures for certificate replacement in order to ensure seamless and secure communication.

2.5.2 Replacing PickMaster® Runtime default certificate with self-signing certificate

Procedure

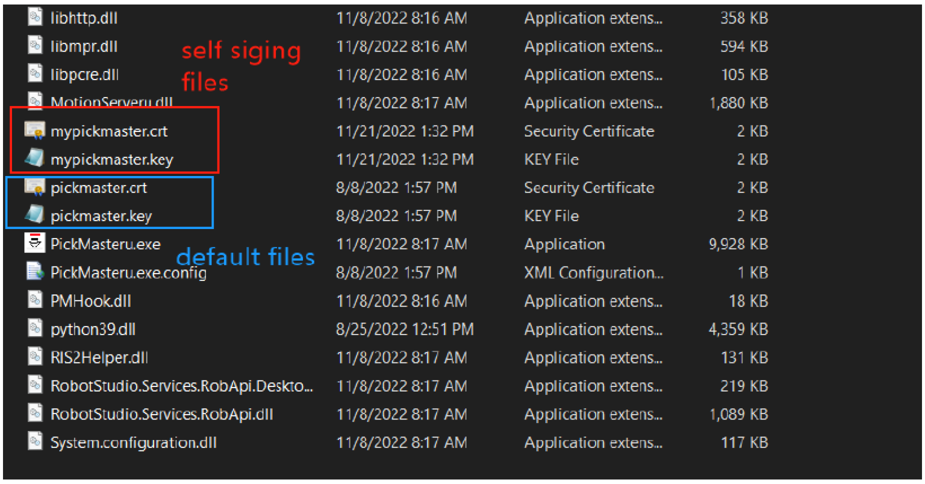
Use the following procedure to replace the PickMaster® Runtime default certificate with the self-signing certificate.

1 Go to the installation path PickMaster® Runtime:

For virtual Runtime: `C:\Program Files (x86)\ABB\PickMaster Twin 2\PickMaster Twin Client 2\PickMaster VirtualRuntime`

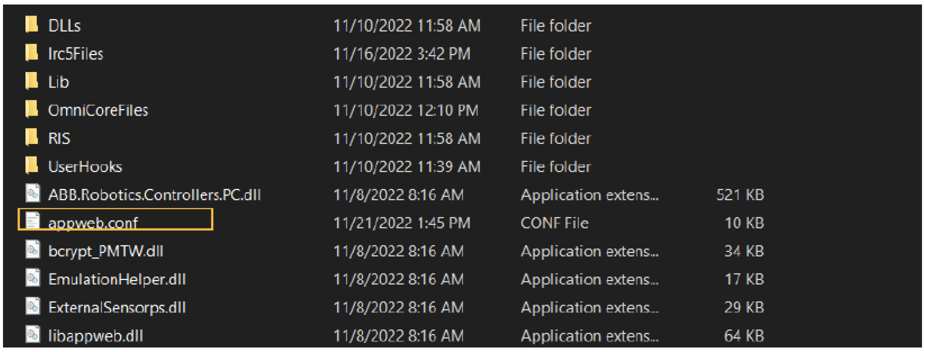
For real Runtime: `C:\Program Files (x86)\ABB\PickMaster Twin 2\PickMaster Twin Client 2\PickMaster Runtime`

2 Add the self-signing certificate `.key` and `.crt` files to the installation path. For example, the file names in red are the self-signing certificate files.



xx2300000747

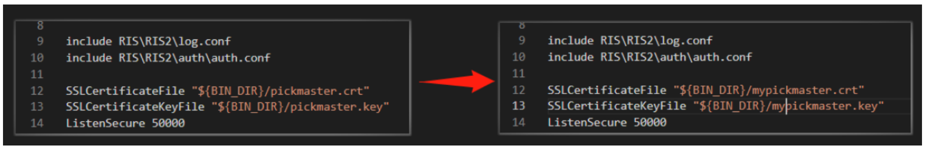
3 Double click to open the `appweb.conf` file.



xx2300000748

Continues on next page

4 Change the `SSLCertificateFile` and `SSLCertificateKeyFile` to the self-signing certificate files.



xx2300000749

5 Save the changes as administrator.

2.5.3 Generating self-signing certificate with OpenSSL

Introduction

When the users need to generate a self-signing certificate file, it's recommended to generate with `OpenSSL`.

Procedure

Use the following procedure to generate self-signing certificate with OpenSSL.

1. Generate a private key.

```
C:\Program Files\OpenSSL-Win64\bin>openssl genrsa -des3 -out mypickmasterwithpassword.key 2048
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

xx2300000740

2. Enter a password for the `.key` file and verify it.

```
C:\Program Files\OpenSSL-Win64\bin>openssl genrsa -des3 -out mypickmasterwithpassword.key 2048
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

xx2300000741

3. Generate the certificate signing request (CSR).

Tip

Enter the name of the key to invoke the generated private key for the `.csr` file.

```
C:\Program Files\OpenSSL-Win64\bin>openssl req -new -key mypickmasterwithpassword.key -out mypickmaster.csr
Enter pass phrase for mypickmasterwithpassword.key:
```

xx2300000742

4. Fill in the mandatory and optional information.

```
C:\Program Files\OpenSSL-Win64\bin>openssl req -new -key mypickmasterwithpassword.key -out mypickmaster.csr
Enter pass phrase for mypickmasterwithpassword.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CN
State or Province Name (full name) [Some-State]:Shanghai
Locality Name (eg, city) []:Shanghai
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ABB
Organizational Unit Name (eg, section) []:Info Technology
Common Name (e.g. server FQDN or YOUR name) []:PickMasterTwin
Email Address []:Default@cn.abb.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

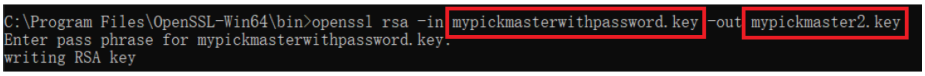
xx2300000743

Tip

User could fill the information based on their own conditions.

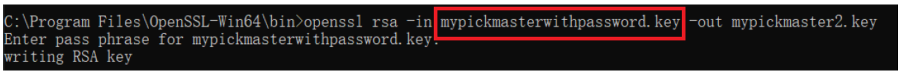
Continues on next page

5. Transform the generated private key file to a new private key file without password.



```
C:\Program Files\OpenSSL-Win64\bin>openssl rsa -in mypickmasterwithpassword.key -out mypickmaster2.key
Enter pass phrase for mypickmasterwithpassword.key:
writing RSA key
```

xx2300000744

> **Note**
>
> Enter the name of the key to invoke the generated private key for the new `.key` file.
>
> 
>
> ```
> C:\Program Files\OpenSSL-Win64\bin>openssl rsa -in mypickmasterwithpassword.key -out mypickmaster2.key
> Enter pass phrase for mypickmasterwithpassword.key:
> writing RSA key
> ```
>
> xx2300000745

6. Generate the self signing certificate.

```
C:\Program Files\OpenSSL-Win64\bin>openssl x509 -req -days 365 -in mypickmaster.csr -signkey mypickmaster2.key -out mypickmaster.crt
```

xx2300000746

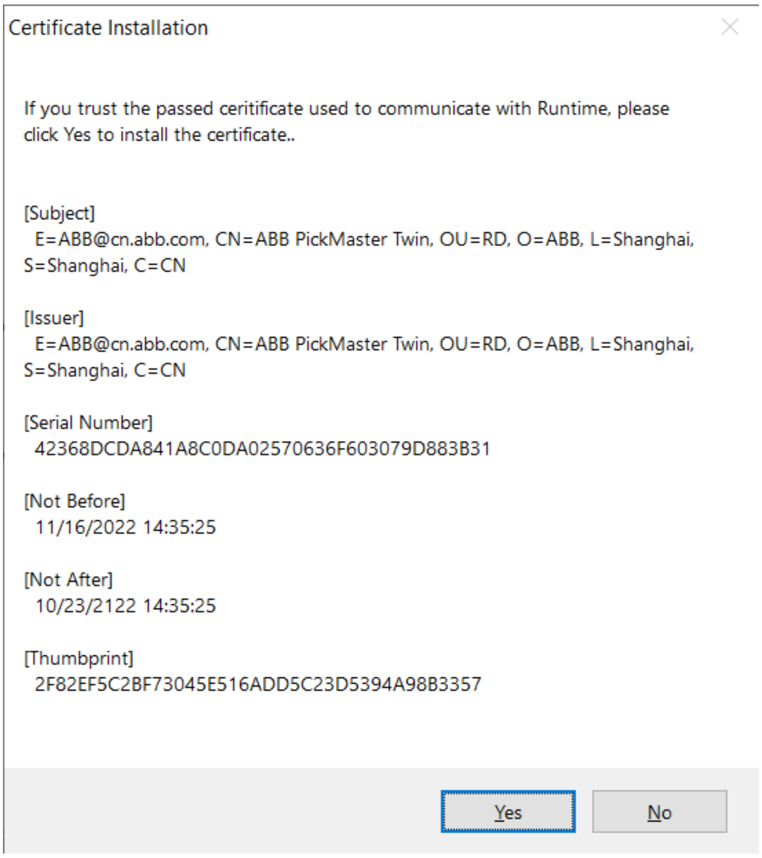
The new private key file without password and the generated self signing certificate `.crt` file are the final output.

2.5.4 Installing self-signing certificate

Procedure

Use the following procedure to install a self-signing certificate.

1 When a PickMaster® Twin product is used for the first time, the following dialog box will pop up.

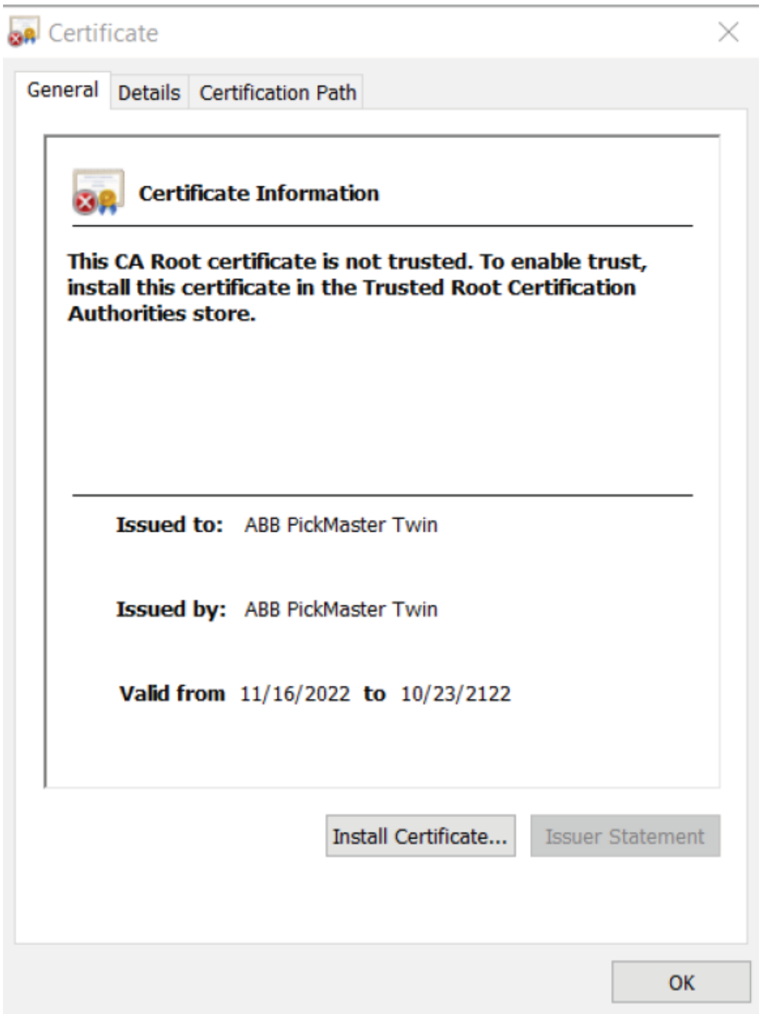


xx2300000751

2 Click **Yes** to install the self-signing certificate.

Continues on next page

3 Click **OK** to finish the installation.



xx2300000752

2.6 PickMaster time synchronization service

Time synchronization service

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

PickMaster Twin PTP v1 is used for IRC5(RobotWare 6) controller.

PickMaster Twin PTP v2 is used for OmniCore(RobotWare 7) controller.

Note

Whenever the Grandmaster is changed, Runtime will take several seconds or minutes to re-synchronize the time in the local area network. During this time, the robots may stop.

Note

To enable the time synchronization service, the user should select the local IP address which is connected to the real controller during installing the PickMaster Twin Client.

If the computer is not yet connected to a real controller, the IP address could also be configured after the installation. For detailed information, see *Configuring local IP address in PickMaster Runtime on page 169*.

Tip

It is recommended to use PTPTrackHound to have an overview on the PTP status of all devices in current local area network.

To download the tool PTPTrackHound, see *www.ptptrackhound.com*.

Settings

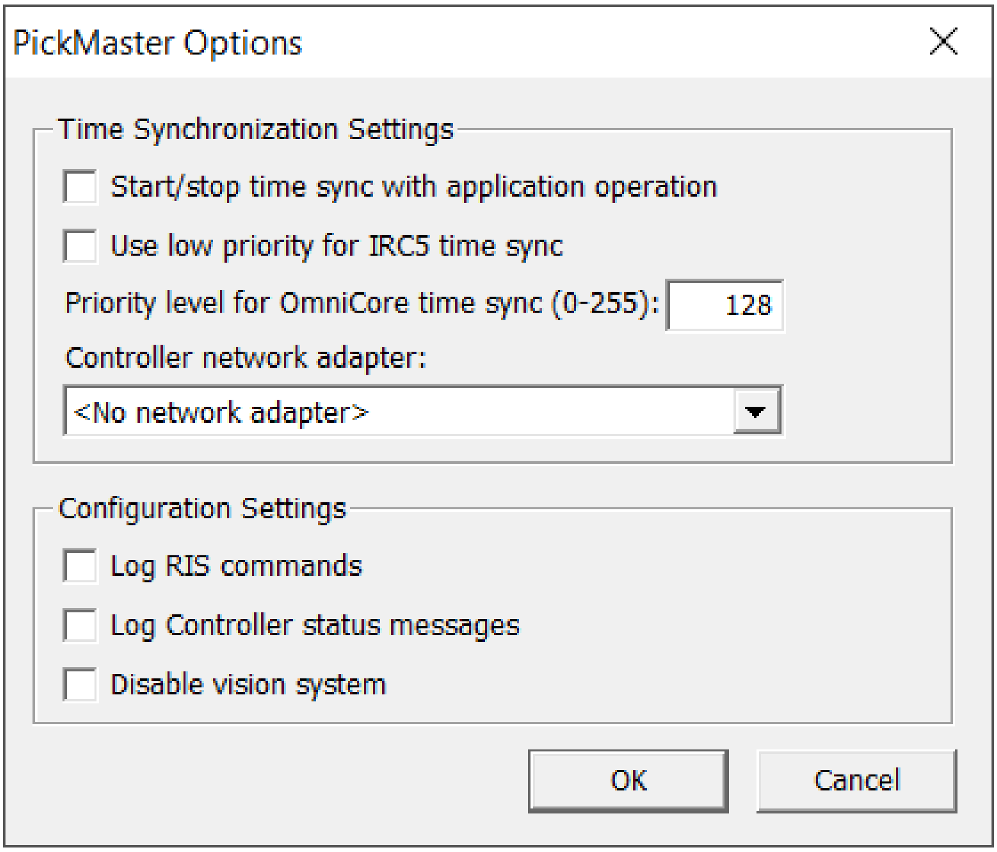
The synchronization service is based on the precision time protocol (PTP), which in turn implements the IEEE 1588-2002 PTP v1(For RobotWare 6)/1588-2008 PTP v2 (For RobotWare 7) standard. This protocol uses multicast messages over UDP/IP and requires that UDP port 319 and 320 are available (for both 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.

PTP was originally defined in the IEEE 1588-2002 standard, officially entitled "Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems" and published in 2002. In 2008, IEEE 1588-2008 was released as a revised standard; also known as PTP Version 2, it improves accuracy, precision and robustness but is not backward compatible with the original 2002 version.

Continues on next page

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.

Time Synchronization Settings interface



xx2300001442

NO.	Name	Value
1	**Start/stop time sync with application operation**	When the checkbox is: • Checked: The PTP server will start/stop automatically with the start/stop of the Runtime. **Tip** The automatically restarted PTP server is the one used last time. The PTP server for IRC5(RobotWare 6) and OmniCore(RobotWare 7) exist independently. • Unchecked: The PTP server will not start/stop automatically with the start/stop of the Runtime.

Continues on next page

NO.	Name	Value
2	**Use low priority for IRC5 time sync**	PickMaster Twin PTPv1 is used for IRC5(RobotWare 6) controller. There are two priorities in PTPv1 strategy: high priority and low priority. The time synchronization server will use high priority device as Grandmaster device. If no high priority device exists or several high priority devices exist in the local area network, the device with better hardware will set as Grandmaster automatically. When the checkbox is: • Checked: The current device is set as low priority. **Tip** The automatically restarted time synchronization server is the one used last time. • Unchecked: The current device is set as high priority. The priority value is saved in the registry.
3	**Priority level for OmniCore time sync (0-255)**	PickMaster Twin PTPv2 is used for OmniCore(RobotWare 7) controller. Use the priority1 value of PTPv2 to control the priority of the devices for the time synchronization service. The range is from 0 to 255. The value is smaller, the priority is higher. The highest priority device in the local area network will be the Grandmaster device. The default value is 128. If the priority1 value is set as the same one on multiple devices in the local area network, the device with better hardware will set as Grandmaster automatically. The priority value is saved in the registry.
4	**Controller network adaptor**	This IP must be set as the IP address of the device which connected to the controller.

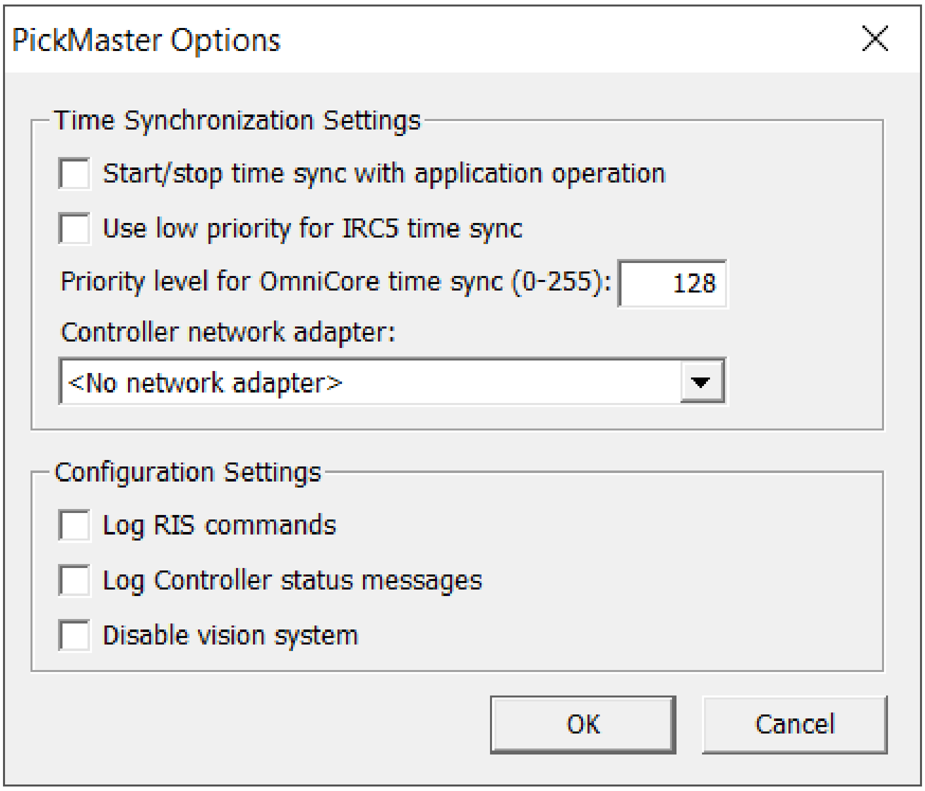
Setting PickMaster options

Use this procedure to set PickMaster options.

1. On the **File** menu, select **Options**.

   The **PickMaster Options** dialog is opened.

Continues on next page



xx2300001442

2. Select to start/stop the PTP server automatically with the start/stop of the Runtime from the **Start/stop time sync with application operation** box.
3. Select to change the priority of current device for PTP v1 server from the **Use low priority for IRC5 time sync** box.
4. Define the value of priority1 of current device for PTP v2 server in the **Priority level for OmniCore time sync (0-255)** text box.

   Tip

   The value is smaller, the priority is higher.

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 **Log RIS commands** to show all RIS commands in the log area.
7. If needed, select **Log IRC5 status messages** to include showing status messages in the log area.

   As default, only warnings and error messages are shown in the log area.
8. 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 a Basler camera is used through the external sensor interface.
9. Click **OK**.

2.7 Software installation

Note

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

Continues on next page

2.7.1 Installing RobotStudio

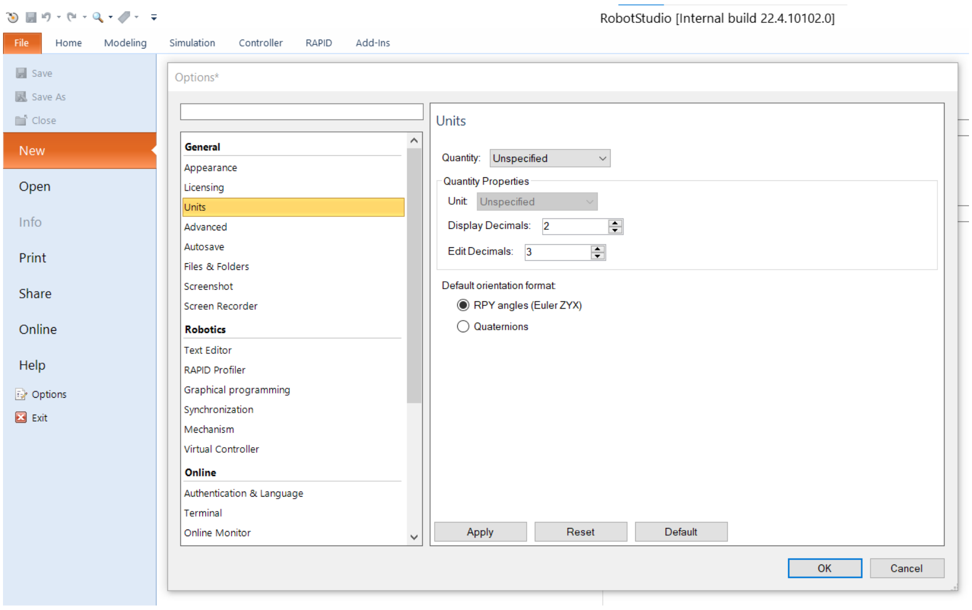
Instruction

For the detailed RobotStudio installation procedure, see *Operating manual - RobotStudio*, *3HAC032104-001*.

Note

When set the unit in RobotStudio, Unspecified is recommended.

The other unit may cause uncertain errors in PickMaster product.



xx2200002058

2.7.2 Installing and uninstalling PickMaster Twin Client

Overview

This section describes the installation process for the PickMaster Twin Client.

Note

The PickMaster 3 and PickMaster Twin Client are not recommended to be installed on a same PC.

They may influence each other.

Note

The PickMaster Twin Client and PickMaster Twin Host are not recommended to be installed on a same PC.

Note

Make sure that you have installed RobotStudio on your computer before installing PickMaster Twin Client. For the installation procedure of RobotStudio, see *Operating manual - RobotStudio*.

Installing PickMaster Twin Client

Use the following procedure to install the PickMaster Twin Client:

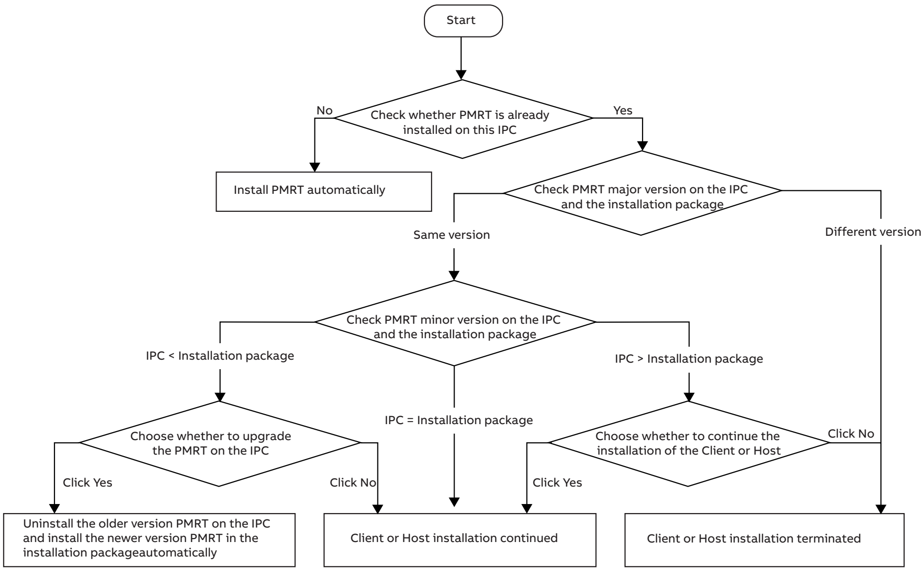
1. Browse to the PickMaster Twin Client installation package and double-click `Setup.exe`.

   The installation starts.
2. Click **Next**.
3. Read the **license agreement** and accept the terms.
4. Click **Next**.
5. PickMaster Twin Client will check whether the **Congnex vision driver** is already installed on this computer automatically.

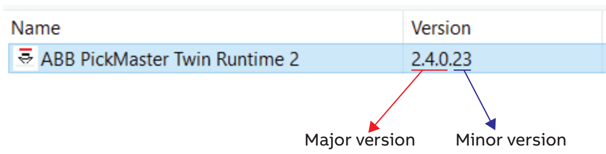
   If not, it will install the **Congnex vision driver** automatically.

Continues on next page

6 PickMaster Twin Client will check to install the PickMaster Runtime according to the following flowchart.



xx2400000341



xx2400000342

7 If installation is continued, click **Next**.

8 Click **Next** to start the installation.

9 When the installation is complete, choose to restart the computer now or later and click **Finish**.

Uninstalling PickMaster Twin Client

Use this procedure to uninstall PickMaster Twin Client:

1 Right click and select **Uninstall** on PickMaster Twin Client in the control panel.

2 If needed, select the **Uninstall Cognex drivers** checkbox to uninstall the cognex drivers on the computer.

3 Click **Next**.

4 Click **Yes** to start the uninstallation.

> **Note**
>
> If only PickMaster Twin Client is existing on the IPC, PickMaster Runtime will be uninstalled automatically at the same time.
>
> If PickMaster Twin Client and PickMaster Twin Host are both existing on the IPC, PickMaster Runtime will not be uninstalled.

Continues on next page

5. When the uninstallation is completed, click **Finish**.

2.8 Electrical connection

2.8.1 Configuring networks

Introduction to the controller network

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

Note

The PickMaster PowerPac must be connected to the WAN port on the controller. Do not use the service port.

Configuring the controller network

If a new local area network (LAN) is created specifically for PickMaster PowerPac 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.

Note

It's not allowed to use any of the following IP addresses which are allocated for other functions:

- 192.168.127.0 - 255

The IP address cannot be on a subnet which overlaps with any of the above reserved IP addresses. If a subnet mask in the class B range has to be used, then a private address of class B must be used to avoid any overlapping. Contact your local network administrator regarding network overlapping.

See the section *Communication* in *Technical reference manual - System parameters*.

Continues on next page

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 additional GigE cards.

The maximum number of cameras that can be used with one PC is 10. Distribute them evenly on the dedicated vision network ports on the PC. Use the supplied cables with fastening screws between GigE card and camera. For the example of network architecture, see *Example of suitable network architecture on page 58*.

Overview

This chapter describes the procedures on setting up the Internet. Otherwise the PackML function cannot work normally.

CAUTION

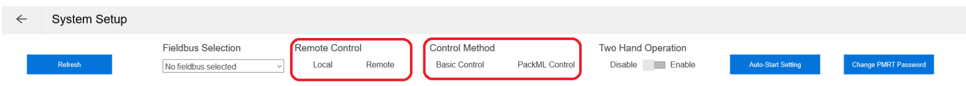
If the Network Adaptor is not renamed correctly, the PickMaster PowerPac cannot work normally.

Use this procedure to set the Network for PickMaster PowerPac:

Note

If the button of control mode selection disappeared, check whether the network name is changed to 'ProfinetIOAdapter' or not.

If not, change the name to fix the problem.



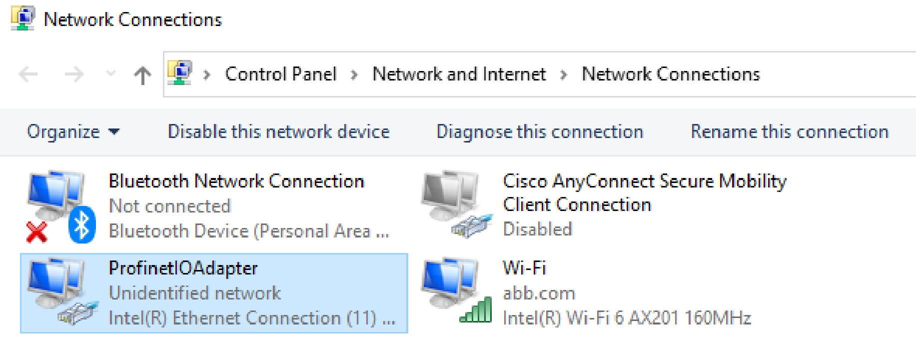
xx1900001727

Continues on next page

Configuring the IPC network

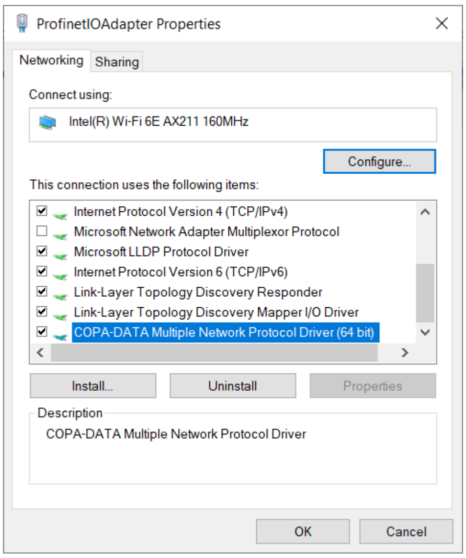
The following procedure is recommended to modify the computer network configuration which camera is connected to:

1. Open the **Network Connections** setting page, right click on the network you are currently using for connecting PickMaster PowerPac and rename the network name to "ProfinetIOAdapter".



xx1900001504

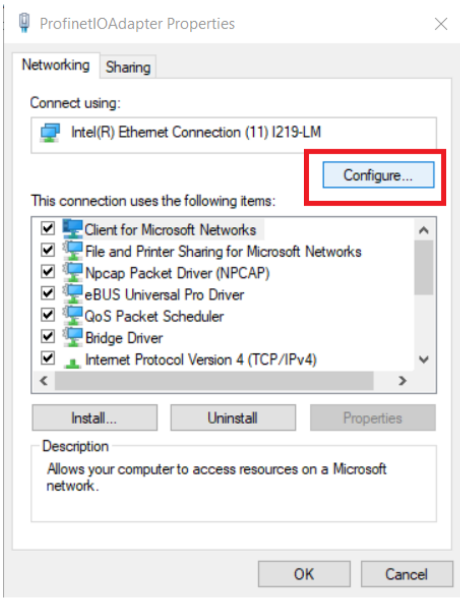
2. Open the property window of the "ProfinetIOAdapter" and make sure that the following protocols or drivers are selected:
   - **eBUS Universal Pro Driver**
   - **COPA-DATA Multiple Network Protocol Driver (64 bit)**
   - **Internet Protocol Version 4(TCP/IPv4)**



xx2300001722

Continues on next page

3 Click **Configure** and then choose the **Advanced** tab.



xx2200002067

4 Modify the following properties as necessary:
- Select the **Jumbo Packet** property and choose the highest possible value in the dialog box.
- In the **Networking** tab, clear all the check boxes listed under **This connection uses the following items** except for **eBUS Universal Pro Driver and Internet Protocol Version 4 (TCP/IpV4).**

5 In addition, Cognex recommends you modify the following properties for this network connection, which may or may not be grouped together with the previous properties:
- Change the **Receive Buffers** property and choose the highest possible value in its **Value** list.
- Change the **Interrupt Moderation Rate** property to **Extreme** in its **Value** list.

6 Click **OK.**

Refer to the embedded Questions and Answers of the Gig Vision Configuration Tool for more details on what system properties you should modify as necessary.

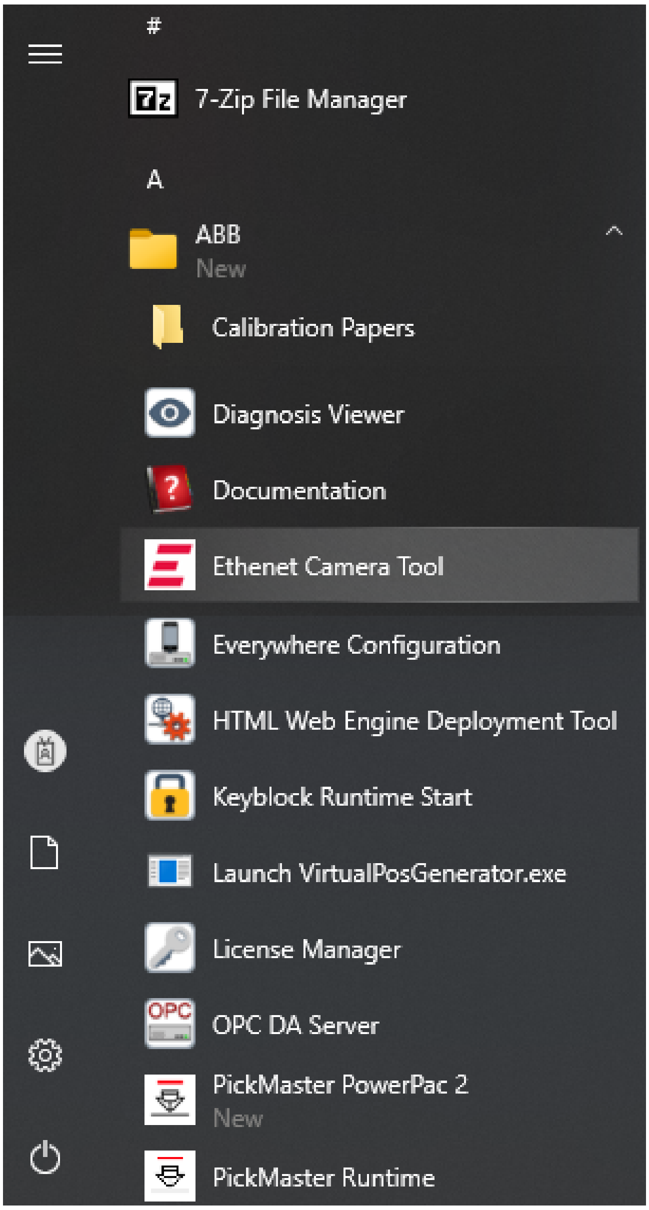
Configuring the vision network

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 58*.

Continues on next page

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.



xx2300001072

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

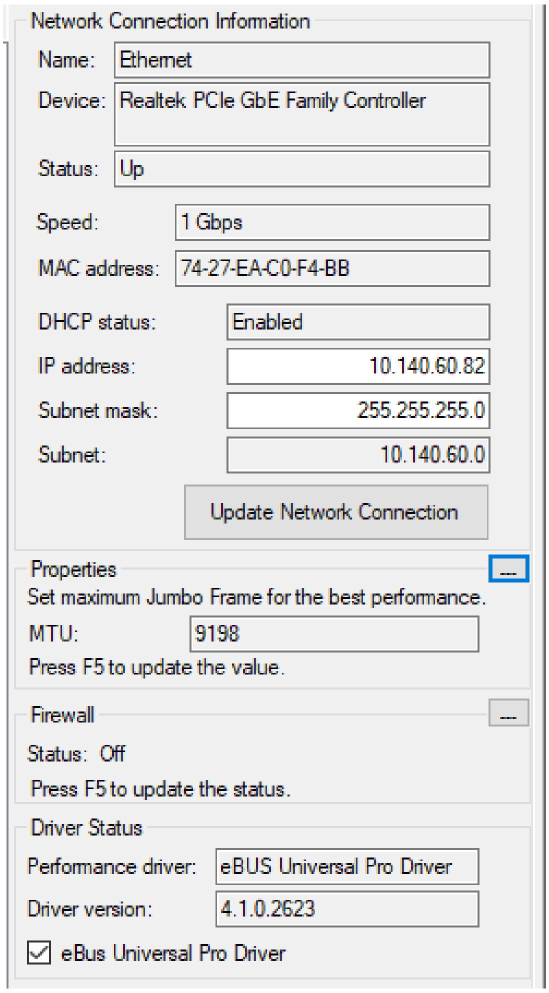
4 In the **Ethernet Camera Tool**, for each vision network port in the tree view, do the following settings:

   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.

Continues on next page

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.

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.

Continues on next page

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 Runtime 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.

Configuring the Runtime network

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

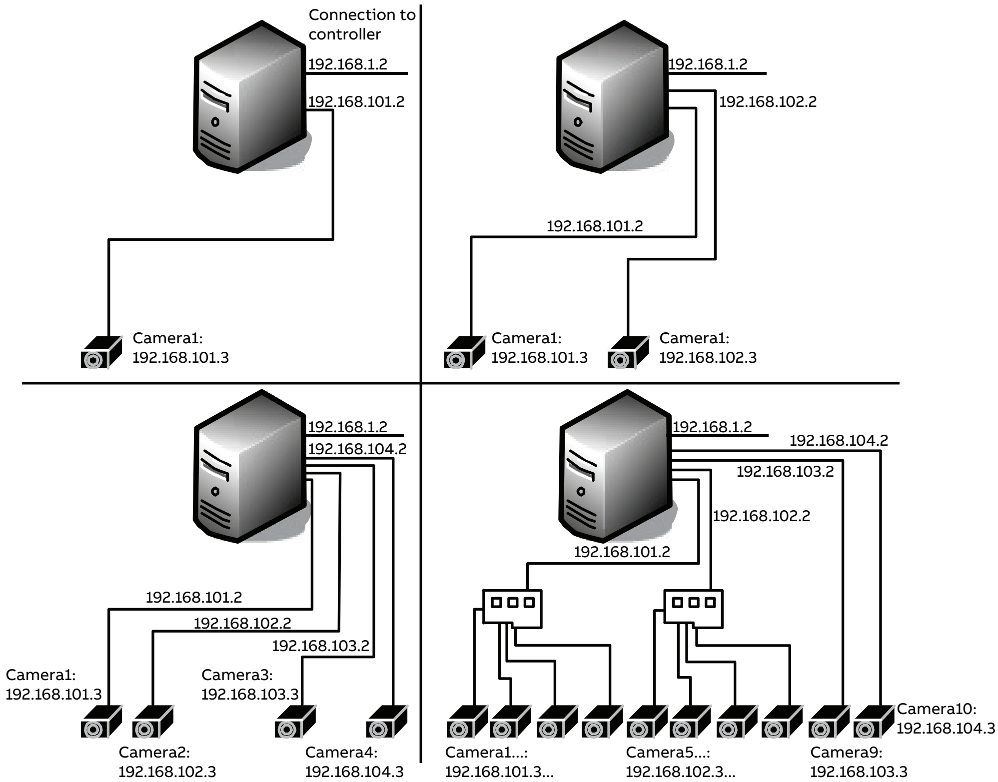
- Use static IP numbering with different addresses for the PickMaster PowerPac and the robot controller.
- IP addresses: 192.168.1.X (where X is between 1 and 253).
- Select **Connect to RRT**, the **Sign in** window is displayed. How to connect to RRT, see *Runtime on page 77*.

Example of 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).

Continues on next page

- Subnet mask: 255.255.255.0
- Gateway: Not Needed.
- DNS: N/A.
- Wins: N/A.



xx2300001025

Note

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

2.8.2 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. Runtime can not reconnect to a camera that has been shut down and restarted. This means that if Runtime is running when a controller that serves as a camera power supply is shut down, Runtime must be restarted after the controller has been switched on again. This problem is avoided by using an external power supply.

A 4-port Gigabit Ethernet board which is included in the GigE Ready option must be used for the Gigabit Ethernet cameras. And the cameras cannot use the same network card with the controller, or the captured images will be affected.

The jumbo packet function of the network card shall be activated when using with the camera.

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 diagram - PickMaster Twin*, *3HAC024480-020*. Detailed information about avoiding EMI/ESD problems is described in *Avoid_EMI_ESD_in_camera_installations*, see *References on page 10*.

Note

All safety information for working with the controller is described in the product manual for the controller.

Prerequisites

Make sure all power is switched off before connecting cameras.

Validated cameras

The following cameras are supported by the PickMaster® Twin:

- `Basler Ace acA1440-73gc`
- `Basler Scout scA1300-32gc`
- `Basler Scout scA1390-17gm`

Continues on next page

CAUTION

Personal injury hazard and risk of damage to camera in case of short circuits.

Short circuits may cause an extreme rise in temperature of the camera's housing. This may damage the camera and may also lead to person injuries, for example, burns. In the worst case, the overheating may cause a fire.

In order to prevent that, limit the current flowing through each individual wire during a short circuit. The maximum current allowed is 2 A. Use a fuse or use a limited power supply.

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 power supply accordingly.

   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 Runtime 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

Circuit diagram - PickMaster Twin, *3HAC024480-020*.

2.8.3 Connecting I/O signals

Introduction to I/O connections

The Runtime 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 10*.

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 Runtime.

   The drawings in *Circuit diagram - PickMaster Twin*, *3HAC024480-020*, uses ACS 301-1P6-3 as conveyor controller, but other conveyor controllers can be 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.

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.

Related information

Circuit diagram - PickMaster Twin, *3HAC024480-020*

Continues on next page

I/O signals on page 175.
Conveyor work area default I/O signals on page 177.

2.8.4 Setting up robot controller

CAUTION

If robot movement can be initiated from an external control panel then an emergency stop must also be available.

RobotWare

PickMaster PowerPac supports OmniCore and IRC5 robot controller. RobotWare is installed on the robot controller. The option *PickMaster Ready* is required to run Runtime. For more details on option, see *PickMaster Twin Product Specification*.

For more information see the product manual for the controller, see *References on page 10*.

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 175*.

The predefined I/O signals are described in the section *Conveyor work area default I/O signals on page 177*.

Related information

Product manual for the controller, see *References on page 10*.

Technical reference manual - System parameters.

Six axes robot configuration on page 66.

2.8.5 Optional robot and process configuration

Conveyor process modification

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.

2.8.6 Six axes robot configuration

Modifications for six axes robots

When using PickMaster with a six axes robot, some modifications must be done in the system parameters to optimize the robot motion with the conveyor tracking process.

Topic Process

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

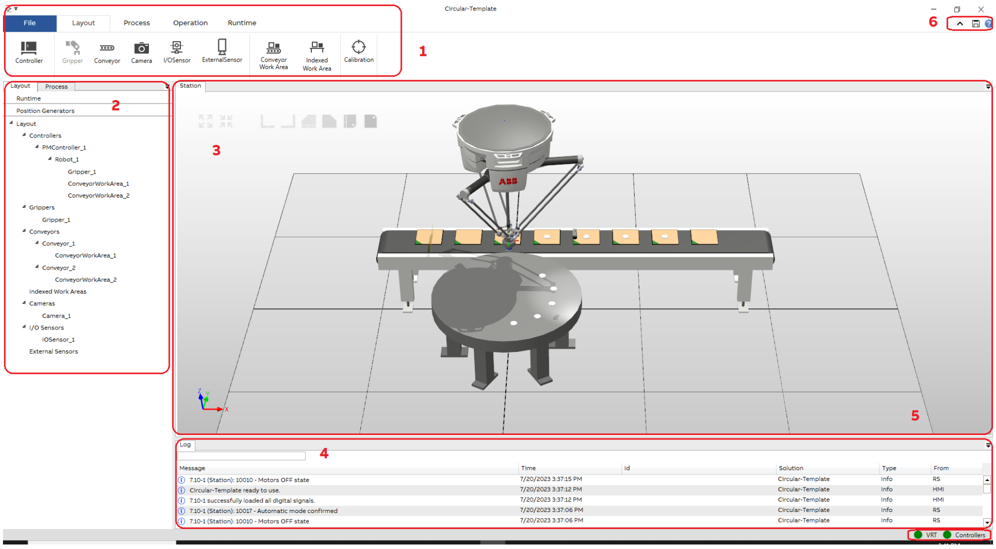
Parameter	Description
Start ramp	This is the correction start filter ramp that is used when connecting to a moving conveyor. This is by default set to 5 (steps). Tune this parameter if higher accuracy is needed. A lower value gives better accuracy but the manipulator may jerk when connecting to the moving 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%.

3 Navigating PickMaster PowerPac

3.1 Main window

Overview

This chapter describes about the user interface of the PickMaster PowerPac. The following figure and table provides information regarding the major elements in the user interface.



xx2100000855

		Description
1	Ribbon tab	Contains the general functions for PickMaster PowerPac. When creating a new solution, the work flow is usually from left to right. For more details, see the section *Ribbon tab on page 69*.
2	Tree view browser	Organizes the programmable objects (for example, robots, sensors, and conveyors) of the picking application in a tree structure. It is separated into **Layout** and **process** tabs. For more details, see the section *Tree view browser on page 76*.
3	Station view	Realistic 3D display of the picking application. The objects in the station view are highlighted when selected or edited using the tree view browser.
4	Log view	Shows all the events happened to current station. **Tip** You can search with key words in the search-box for the specific event.
5	Status view	Shows the status of the controller and system at present.

Continues on next page

		Description
6	Additional operation view	Shows the save button and help button. **Help**: open the PickMaster PowerPac application manual.  xx2100000867

Tip

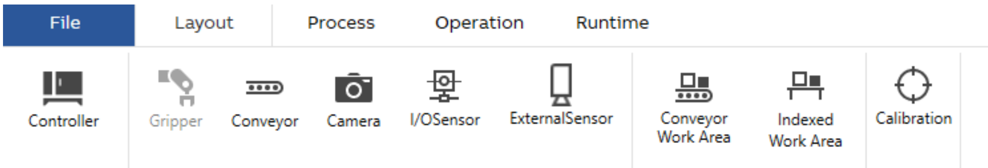
All windows can be distributed and floating freely.

3.2 Ribbon tab

Overview

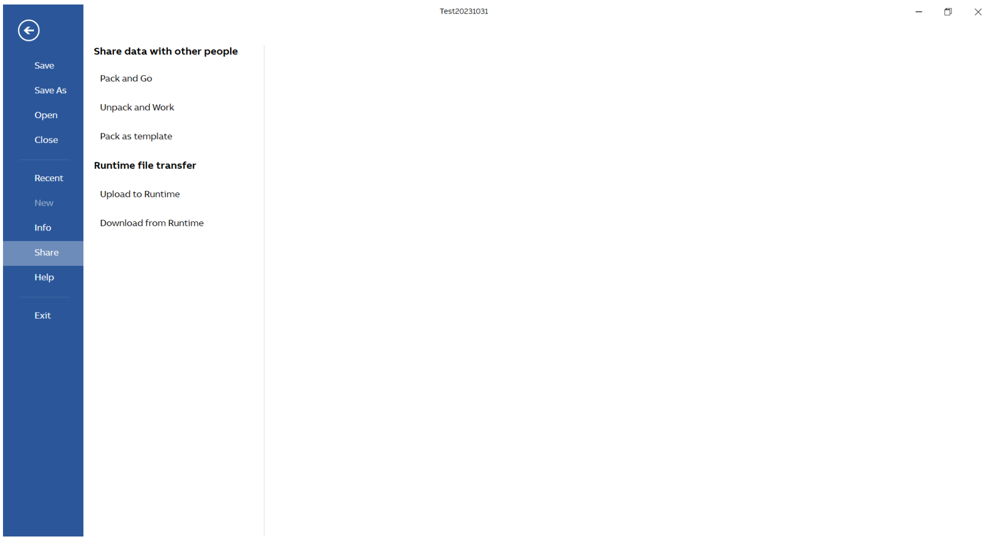
The PickMaster PowerPac ribbon contains elements arranged in various groups. The following figures and tables provide more information regarding the elements in the PickMaster PowerPac ribbon.

Following are the objects and configurations saved in the ribbon tab.



xx2100000856

File



xx2100000854

Button	Description
 xx2100000857	Go back to the main window.
Save	Save the changes for the solution at present. **Note** If the solution will be used in the PickMaster Operator, it must have been connected to a real controller with the same configuration on PickMaster PowerPac.

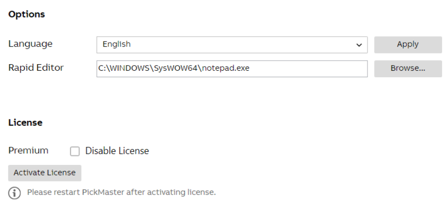
Continues on next page

Button		Description
Save as		Save your present solution as a new solution in desired location. **Note** If the solution will be used in the PickMaster Operator, it must have been connected to a real controller with the same configuration on PickMaster PowerPac.
Open		Open other solutions or any solutions saved in your local folder. **Tip** Only solutions or shared files which are created with PickMaster PowerPac 2.0 or later can be opened.
Close		Close your present solution.
Recent		Open the solutions which has been opened before.
New	**Solution with Empty Station**	Create a new empty solution. **Note** Only a-z, A-Z, 0-9,_ ,- can be used in file name when creating the solution.
	Solution with Cell Template	Create a new solution with the template.
Info		Show the basic information of the opened solution. **Tip** This page will only show up when a solution is opened.

Continues on next page

Button		Description
Share	**Share data with other people**	**Pack and GO** Pack all the information of current solution, controller used in the solution and 3D models into a file so that it makes sharing files between users. **Note** It is not allowed to rename the packed file. Otherwise it may cause unpacking problem. **Note** Python script files will not be included in the Pack&Go file. Copy the Python script files to the desired destination.
		UnPack and Work Unpack the shared files which contains all the information of a solution, controller used in the solution and 3D models.
		Pack As Template Pack your present solution as a template in your local folder.
	Runtime file transfer	**Upload to Runtime** Upload a desired `.rspag` file to the connected Host computer. A suffix is added to the name containing "PP", date and time. **Tip** Connect to the real Runtime on the Host computer before the uploading.
		Download from Runtime Download a desired `.rspag` file from the connected Host computer. **Tip** Connect to the real Runtime on the Host computer before the downloading.

Continues on next page

<table>
<tr><th colspan="2">Button</th><th>Description</th></tr>
<tr><td rowspan="3">Help</td><td>About</td><td>Shows the basic version information.</td></tr>
<tr><td>Options</td><td>Language: choose the applied language. Eight languages are supported:
English
Simplified Chinese
German
Italian
Spanish
Japanese
French
Korean
Rapid Editor: specify the editor to open Rapid.
License: show current license type.
Disable License check box: disable the premium license if checked.
Activate License icon: activate a premium license.


xx2100000858
Note
If the user changes the language during working with PickMaster PowerPac, the selected language will be valid after the PickMaster PowerPac restarted.</td></tr>
<tr><td>Manual</td><td>Open the PickMaster PowerPac application manual.</td></tr>
<tr><td>Exit</td><td></td><td>Close and exit the PickMaster PowerPac.</td></tr>
</table>

Tip

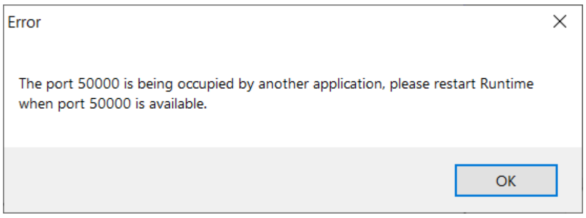
When opening or creating a new solution with PickMaster PowerPac, the **Virtual Runtime** will start and be connected automatically.

Continues on next page

Tip

The PickMaster® Runtime (VRT and RRT) is defined to use 50000 port. If 50000 port is occupied by other program, you will have this warning and not be able to connect to Runtime,

Release the 50000 port and restart the PickMaster® Runtime.

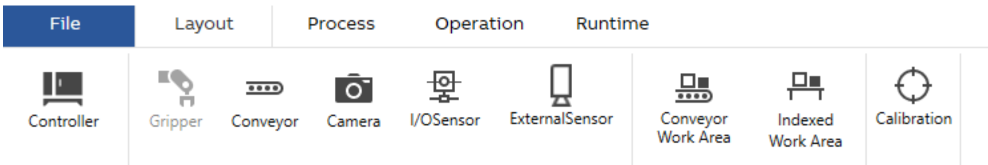


xx2100000868

Use this procedure to release the 50000 port:

1. Enter the command `netstat -aon|findstr "50000"` in the CMD window.
2. The process that occupies port 50000 will be listed in the window. Obtain the PID code of the process.
3. Find the process corresponding to this PID in the task manager and close the it (Make sure that this process is allowed to be closed on this computer).
4. Restart PickMaster® Runtime and connect.

Layout



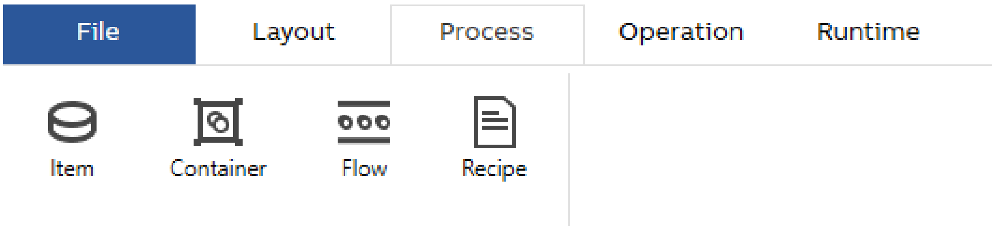
xx2100000856

Button	Description
Controller	Add a controller with a robot system in the station view. More details about creating a controller is available in the section *Controller on page 112*.
Gripper	Add a gripper. More details about creating a gripper is available in the section *Gripper on page 115*.
Conveyor	Add a conveyor. More details about creating a conveyor is available in the section *Conveyor on page 118*.
Camera	Add a camera. More details about creating a camera is available in the section *Camera on page 120*.
I/O Sensor	Add an I/O sensor. More details about creating an I/O sensor is available in the section *Adding an I/O sensor on page 122*.

Continues on next page

Button	Description
External Sensor	Add an external sensor. More details about creating an external sensor is available in the section *Adding an external sensor on page 124*.
Conveyor Work Area	Add a conveyor work area. More details about creating a conveyor work area is available in the section *Work area on page 125*.
Indexed Work Area	Add an indexed work area. More details about creating an indexed work area is available in the section *Adding an indexed work area on page 128*.
Calibration	Calibrate the created solution in PickMaster PowerPac. More details about calibrating the created solution. is available in the section *Calibration on page 135*.

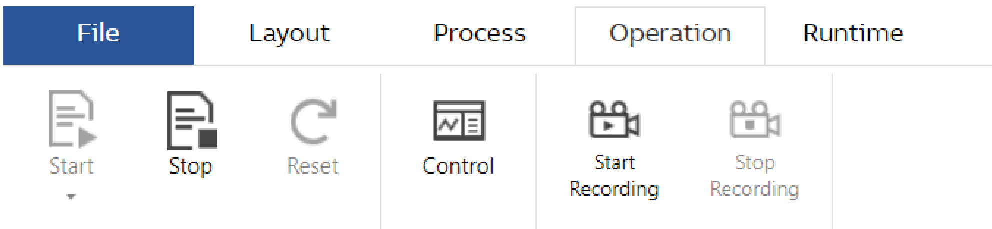
Process



xx2100000859

Button	Description
Items	Add items. More details about creating an item is available in the section *Item on page 136*.
Container	Add containers. More details about creating an container is available in the section *Container on page 142*.
Flow	Define how the items and containers are to be generated in the simulation. More details about creating a flow is available in the section *Flow on page 149*.
Recipe	Create a recipe. More details about creating a recipe is available in the section *Recipe on page 152*.

Operation

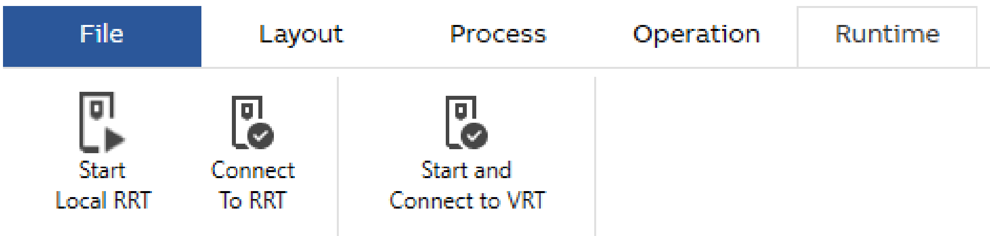


xx2100000860

Continues on next page

Button	Description
Start	Start a simulation. When click on the drop-down arrow, **Start and Record** shows up. **Start and Record**: start and record the simulation as an .exe file.
Stop	Stop the simulation. **Tip** **Stop** will stop the solution and set the robot back to origin.
Reset	Reset the station view from objects temporarily created in the previously run simulation. **Tip** **Reset** will clean the items and containers on the conveyor.
Control	Start a production. More details about how to run the production is available in the section *Simulation on page 163* and *Emulation on page 319*.
Start Recording	Record the simulation including the curser and mouse-clicks as .mp4 file.
Stop Recording	Stop recording the simulation including the curser and mouse-clicks.

Runtime



xx2100000862

Button	Description
Start Local RRT	Start the **Runtime** on the computer. **Note** **Local RRT** means the Runtime installed with PickMaster PowerPac. It can be used for test purposes.
Connect to RRT	Connect to the real **Runtime**.
Start and Connect to VRT	Start the virtual **Runtime** on the computer and connect to it.

3.3 Tree view browser

3.3.1 Layout

Overview

The **Layout** tab displays the Runtime and the application hardware objects such as robots, cameras, conveyors, and so on.

Following are the objects and configurations saved in the **Layout** tab.

xx2100000863

- **Runtime**
- **Position generator**
- **Layout**
  - **Controllers**
  - **Grippers**
  - **Conveyors**
  - **Indexed Work Areas**
  - **Cameras**
  - **I/O Sensors**
  - **External Sensors**

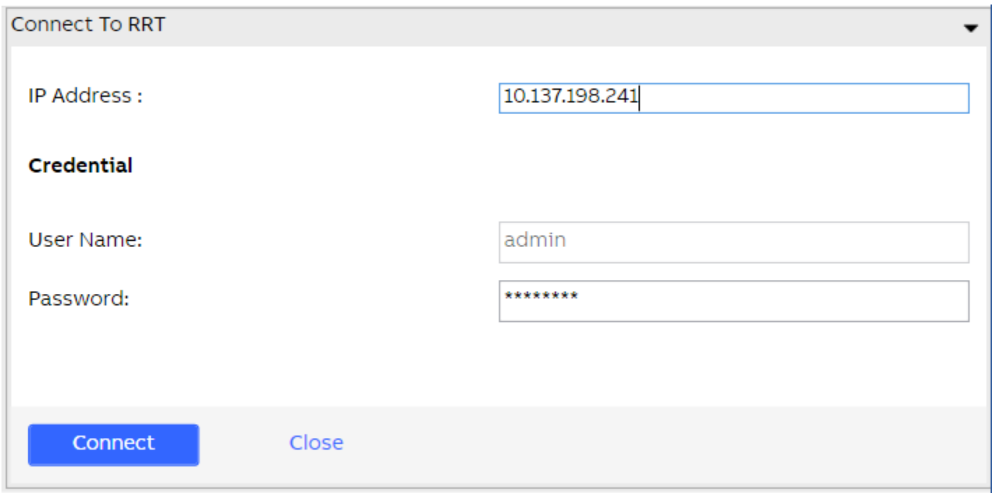
Continues on next page

Runtime

Right-click on **Runtime** to set the connection to the virtual Runtime (VRT) in simulation mode or the real Runtime (RRT) for operating the real robots on the Host computer in emulation mode.

> **Tip**
>
> Before connecting to RRT, start the PickMaster Runtime on the Host computer.

When selecting **Connect to RRT**, the **Sign in** window is displayed.



xx2100000872

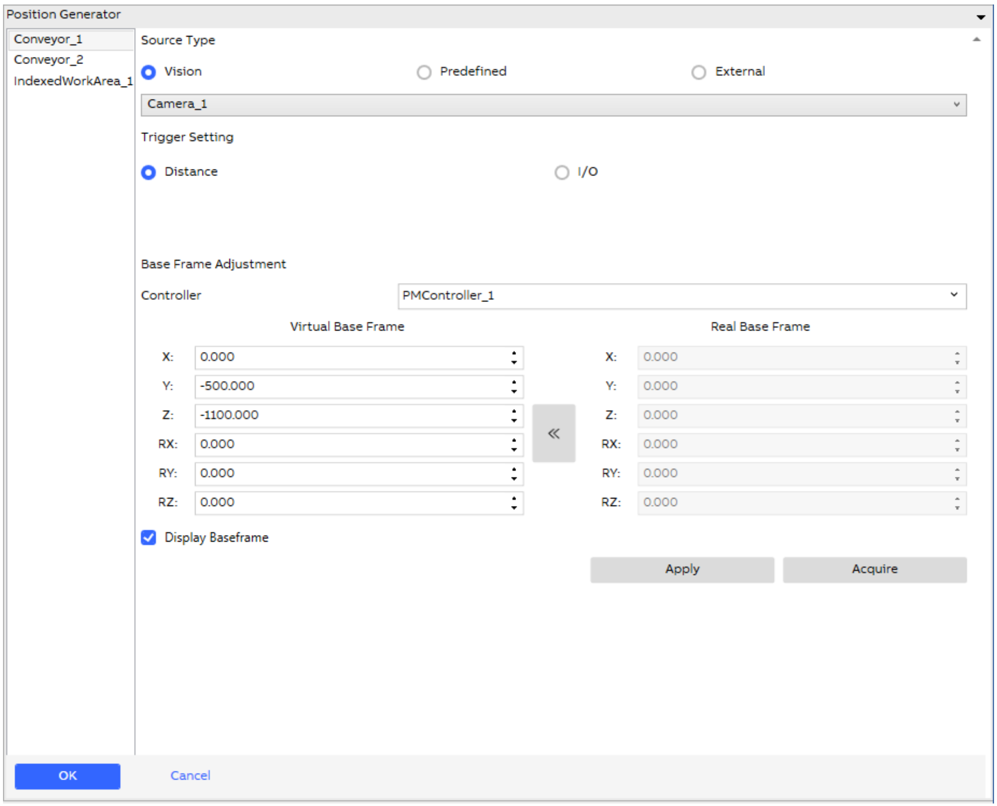
	Description
IP Address	Enter the IP address of the Runtime computer. **Tip** Check the IPv4 address of the computer which the PickMaster Runtime is installed on. **Note** Loopback address is NOT allowed to use as the real PickMaster Runtime IP address, for example 127.0.0.1. Loopback address will cause errors in vision function.
Credential	
UserName	The default user name is admin. And it CANNOT be changed.
Password	Enter the password of your account in the Runtime.

> **Note**
>
> **Local RRT** means the Runtime installed with PickMaster PowerPac. It can be used for test purposes.

Continues on next page

Position generator

Right-click on **Position Generator** will allow you to define the relationship and source type of conveyors.



xx2100000874

	Description
Available conveyor and indexed work area list	Selects a conveyor or indexed work area in order to set the related relationships.
Source Type	Select the input signal source type: • **Vision**: 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 *Adding vision model on page 283*. **Tip** If the source type is set to **Vision**, all available cameras and related items will be listed in the **Available Camera**. • **Predefined**: If the source type is set to **Predefined**, the positions generated by the position source are statically defined and no camera is used. • **External**: If the source type is set to **External**, an external sensor in the solution together with external position generators are used to define item positions. **Tip** If an indexed work area is used, external sensor function will be disabled.

Continues on next page

	Description
Trigger Setting	Select **Trigger type** to define when to generate new item positions. **Note** If the trigger type is set to **Distance** the trigger distance must be defined in the **Trigger Distance** box in **Operation** setting under **Recipe**. 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. **CAUTION** If the **Predefined** and IO sensor are selected in the recipe, tune the pick location in the **Tuning** for a radial distance of the item to make up the offset. **Tip** If an indexed work area is used, **Trigger Setting** is not available.
Base Frame Adjustment	Adjust the base frame for selected conveyor or indexed work areas. For more information, see *Adjusting the base frame on page 131*.

Controllers

Managing controller

Right-click on a **Controller** in the tree view to select and define a controller.

	Description
Edit controller	Change the settings for the selected controller. When you right-click on a controller and select **Edit controller**, the **Edit controller** window is displayed. See the following section for more details about managing a selected controller.
Delete	Delete the selected controller.
Rename	Change the name of the selected controller.

The following table provides details about the **Edit controller** window.

	Description
Controller Name	Displays the name of the selected controller.
System Name	Displays the name of the system.
IP Address	Displays the IP address of the selected controller.
Version	Displays the version of the system.
System ID	Displays the ID of the system.
Select Virtual controller icon	Start the selected virtual controller.
Select Real controller icon	Select a real controller when running production.

Continues on next page

Managing robot

Right-click on a **Robot** in the tree view to manage the robot.

	Description
Jump Home	Move the robot to the home position.
Set Position	Set a position for the selected robot. When you right-click on a robot and select **Set Position**, the **Set Robot Pose** window is displayed. See the following section for more details about managing the position of a selected robot.
Examine	Examine the robot in the Station view.
Rename	Change the name of the selected robot.

Set Position

The following table provides details about the **Set Position** configuration window.

	Description
Reference	Select a coordinate system.
Position X,Y,Z (mm)	Set a new position for the selected robot.
Orientation (deg)	Set a new orientation for the selected robot.

Grippers

Managing grippers

Right-click on a **Gripper** in the tree view to manage the gripper.

	Description
Settings	Manage the settings of the selected gripper. When you select **Setting**, the **Robot_Gripper Setting** window is displayed. More details about managing a selected gripper is available in the section *Gripper on page 115*.
Delete	Delete the selected gripper.
Rename	Change the name of the selected gripper.
Examine	Examine the selected gripper in the Station view.

Conveyors

Managing conveyor

Right-click on a **Conveyor** in the tree view to manage the conveyor.

	Description
Setting	Manage the settings of the selected conveyor. When you select **Setting**, the **Conveyor Setting** window is displayed. More details about managing a selected conveyor is available in the section *Conveyor on page 118*.
Delete	Delete the selected conveyor.
Rename	Change the name of the selected conveyor.

Continues on next page

	Description
Hotspot	Manage the hotspots. When you select **Hotspot**, the **Set Conveyor hotspots** window is displayed. See the following section for more details about the **Set Conveyor hotspots** window. **Note** The hotspot is a saved location on the conveyor. A hotspot is used to define where on the conveyor the flow shall be generated. There is always a default hotspot, Hotspot0, located at the beginning of the conveyor. If the flow appears at a wrong location, modify the hotspot location to adjust it.
Examine	Examine the selected conveyor in the Station view.

Managing work area

Right-click on a **Conveyor WA** in the tree view to manage the work area.

	Description
Setting	Manage the settings of the selected work area. When you right-click on a conveyor work area and select **Setting**, the **Conveyor WA Setting** window is displayed. More details about managing a conveyor work area is available in the section *Work area on page 125*. When you right-click on an indexed work area and select **Settings**, the **Indexed WA Setting** window is displayed. More details about managing an indexed work area is available in the section *Adding an indexed work area on page 128*.
Delete	Delete the selected conveyor work area.
Rename	Change the name of the selected conveyor work area.

Indexed Work Areas

Managing indexed work area

Right-click on a **Indexed Work Areas** in the tree view to manage the indexed work area.

	Description
Setting	Manage the settings of the selected indexed work area. When you select **Setting**, the **Indexed Work Area Setting** window is displayed. More details about managing a selected conveyor is available in the section *Indexed work area on page 128*.
Delete	Delete the selected indexed work area.
Rename	Change the name of the selected indexed work area.

Continues on next page

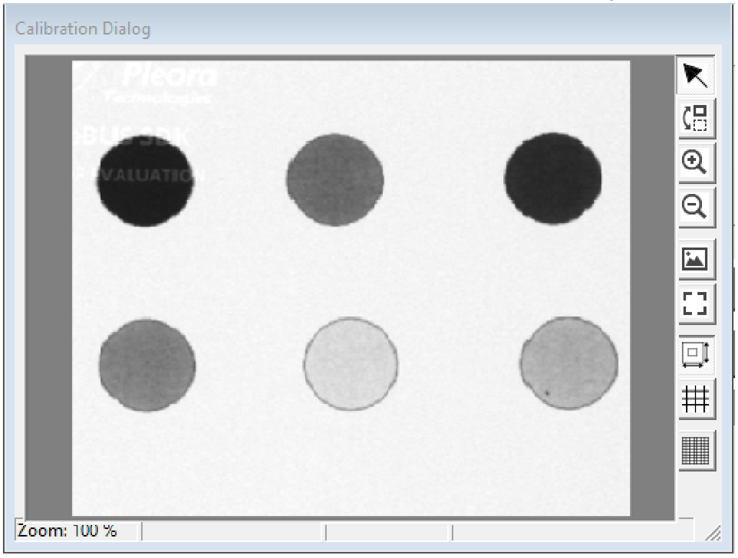
	Description
Hotspot	Manage the hotspots. When you select **Hotspot**, the **Set Indexed Work Area hotspots** window is displayed. See the following section for more details about the **Set Indexed Work Area hotspots** window. **Note** The hotspot is a saved location on the indexed work area. A hotspot is used to define where on the indexed work area the flow shall be generated. There is always a default hotspot, Hotspot0, located at the beginning of the indexed work area. If the flow appears at a wrong location, modify the hotspot location to adjust it.
Examine	Examine the selected indexed work area in the Station view.

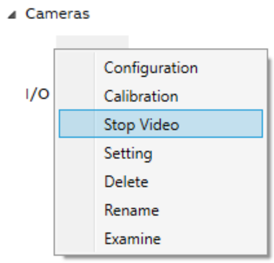
Cameras

Right-click on a **Camera** in the tree view to manage the camera.

	Description
Configuration	Configure the selected camera. When you right-click on a camera and select **Configuration**, the **Camera Configuration** window is displayed. More details about managing a camera is available in the section *Configuring camera on page 180*.
Calibration	Calibrate the selected camera. When you right-click on a camera and select **Calibration**, the **Camera Calibration** window is displayed. More details about managing a camera is available in the section *Calibrating camera on page 273*. **Tip** **Calibration**, **Live Video** and **Setting** are enabled only for real camera.

Continues on next page

<table>
<tr><th></th><th>Description</th></tr>
<tr><td>Live Video</td><td>Shows the view of the scene of the real camera before production.


xx2100001521
When you started the live video, you can stop it by click the Stop Video after right click on the sensor.


xx2100001809
Tip
Calibration, Live Video and Setting are enabled only for real camera. </td></tr>
<tr><td>Setting</td><td> Manage the settings of the selected camera.
When you right-click on a camera and select Setting, the Camera Setting window is displayed. More details about managing a camera is available in the section Camera on page 120.
Tip
Calibration, Live Video and Setting are enabled only for real camera. </td></tr>
<tr><td>Delete</td><td>Delete the selected camera.</td></tr>
<tr><td>Rename</td><td>Change the name of the selected camera.</td></tr>
<tr><td>Examine</td><td>Examine the selected camera in the Station view.</td></tr>
</table>

Continues on next page

I/O Sensors

Right-click on an **I/O Sensor** in the tree view to manage the I/O sensor.

	Description
Setting	Manage the settings of the selected I/O senor. When you right-click on an I/O sensor and select **Setting**, the **I/O Sensor Setting** window is displayed. More details about managing an I/O sensor is available in the section *Adding an I/O sensor on page 122*.
Delete	Delete the selected I/O sensor.
Rename	Change the name of the selected I/O sensor.
Examine	Examine the selected I/O sensor in the Station view.

External Sensors

Right-click on an **External Sensors** in the tree view to manage the external sensor.

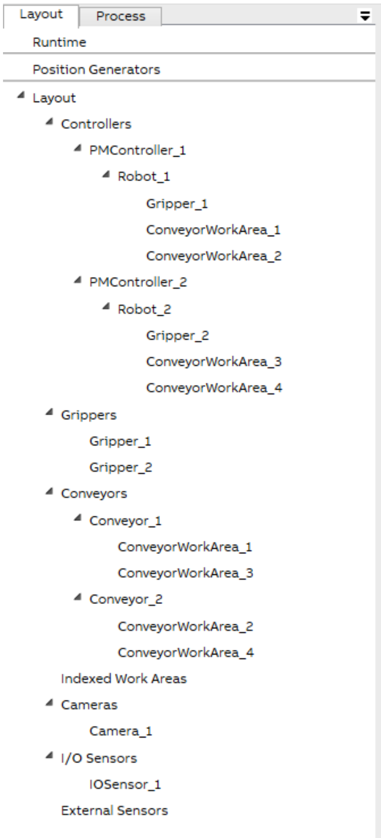
	Description
Setting	Manage the settings of the selected external sensor. When you right-click on an external sensor and select **Setting**, the **External Sensor Setting** window is displayed. More details about managing a camera is available in the section *Adding an external sensor on page 124*.
Configuration	Configure the selected external sensor. When you right-click on an external sensor and select **Configuration**, the **External Sensor Configuration** window is displayed. More details about managing a camera is available in the section *External sensor on page 343*.
Delete	Delete the selected external sensor.
Rename	Change the name of the selected external sensor.
Examine	Examine the selected external sensor in the Station view.

3.3.2 Process

Overview

The **Process** tab displays the configuration file and the application hardware objects such as items, containers, flows, and recipes.

Following are the objects and configurations saved in the **Process** tab.



xx2100000864

- **Items**
- **Containers**
- **Flow**
- **Recipes**

Items

Managing item

Right-click on an **item** in the tree view to manage the item.

	Description
Setting	Manage the settings of the selected item. When you select **Setting**, the **Item Setting** window is displayed. More details about managing a selected item is available in the section *Adding an item on page 136*.
Delete	Delete the selected item.
Rename	Change the name of the selected item.

Continues on next page

	Description
Copy	Create a copy of the selected item with all settings.

Containers

Managing container

Right-click on a **Container** in the tree view to manage the container.

	Description
Setting	Manage the settings of the selected container. When you select **Setting**, the **Container Setting** window is displayed. More details about managing a selected container is available in the section *Adding a container on page 142*.
Delete	Delete the selected container.
Rename	Change the name of the selected container.
Copy	Create a copy of the selected container with all settings.

Flow

Managing flow

Right-click on a **Flow** in the tree view to manage the flow.

	Description
Setting	Manage the settings of the selected flow. When you right-click on a flow and select **Setting**, the **Flow Setting** window is displayed. More details about managing a flow is available in the section *Flow on page 149*.
Delete	Delete the selected flow.
Rename	Change the name of the selected flow.
Copy	Create a copy of the selected flow with all settings.

Recipes

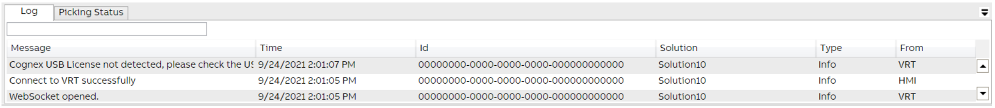
Managing recipe

Right-click on a **Recipe** in the tree view to manage the recipe.

	Description
Setting	Manage the settings of the selected recipe. When you select **Setting**, the **Recipe Setting** window is displayed. More details about managing a selected recipe is available in the section *Recipe on page 152*.
Delete	Delete the selected recipe.
Rename	Change the name of the selected recipe.
Copy	Create a copy file of the selected recipe with all settings.

3.4 Log view

Log



xx2100001518

	Description
Log	Shows all logs. **Note** If right click on one log message, **Save Log** and **Clear All** are available.
Filter box	Filter the specific logs with key words.
Context menu	Expands more operation on the logs, for example export or clean up the current logs.
Picking Status	Shows an overview of the picking status in summary or detail.

3.5 Status view

Status

When the system starts, the status of the controller and the Runtime will show up on the top right corner as the illustration.

xx2100000865

	Description	Note
Runtime	**Grey**: No solution is opened.	
VRT	**Red**: The connection to the virtual Runtime fails. **Green**: The connection to the virtual Runtime successes. **Yellow**: The connection to the virtual Runtime is progressing.	
RRT	**Red**: The connection to the real Runtime fails. **Green**:The connection to the real Runtime successes. **Yellow**: The connection to the real Runtime is progressing.	
Controllers	**Red**: There is at least one controller stopped. **Green**: All controllers are started and auto-running. **Yellow**: There is at least one controller started and under manual controlling or just connected. **Grey**: No controller is added in the existing solution.	
Controller	**Red**: Controller is stopped **Green**: Controller is started and auto-running. **Yellow**: Controller is started and under manual controlling or just connected.	Click on the **Controllers** button , the detailed status for each controller will show up.  xx2100001517

4 Working with PickMaster PowerPac

4.1 Overview

Overview

Working with PickMaster PowerPac in virtual Runtime is to fulfill the simulation function in a visual status.

Working with PickMaster PowerPac in real Runtime is to fulfill the emulation and production function in real stations with real robots and controllers.

Simulation is a previous debugging procedure to save cost and time when creating real stations.

The following is a recommended flow for working with PickMaster PowerPac. After you complete the workflow, you can perform these task in any order.

Note

The controller (contains at least one robot system) should be set up in RobotStudio or PickMaster PowerPac before adding controller to the solution in PickMaster PowerPac.

If multiple controllers is needed in the solution, you need to create multiple controllers in advance. The same controller cannot be imported into the same solution repeatedly in PickMaster PowerPac.

Note

If any firewall or antivirus software is installed, add `pickmasteru.exe` and `visionclient.exe` to the white list.

Otherwise the PickMaster PowerPac cannot connect Runtime and the vision function cannot work normally.

Workflow for PickMaster PowerPac

Use this procedure to work with PickMaster PowerPac:

		Task	Description
VRT	1	Create an empty solution.	For detailed information, see *Solution on page 111*.
	2	Add a controller.	For detailed information, see *Controller on page 112*.
	3	Add a gripper.	For detailed information, see *Gripper on page 115*.
	4	Add a conveyor.	For detailed information, see *Conveyor on page 118*.
	5	Add a camera.	For detailed information, see *Camera on page 120*.
	6	Add an I/O sensor.	For detailed information, see *Adding an I/O sensor on page 122*.
	7	Add an external sensor.	For detailed information, see *External sensor on page 124*.

Continues on next page

		Task	Description
	8	Add a work area.	For detailed information, see *Work area on page 125*.
	9	Add an indexed work area.	For detailed information, see *Indexed work area on page 128*.
	10	Set position generator.	For detailed information, see *Position generator on page 130*.
	11	Calibrate the solution.	For detailed information, see *Calibration on page 135*.
	12	Add an items.	For detailed information, see *Item on page 136*.
	13	Add a container.	For detailed information, see *Container on page 142*.
	14	Add a recipe.	For detailed information, see *Recipe on page 152*.
	15	Do simulation	For detailed information, see *Simulation on page 163*.
RRT	16	Calibrate the robots.	For detailed information, see *Calibrating robot on page 184*.
	17	Switch to real **Runtime**.	For detailed information, see *Switching to real Runtime on page 169*.
	18	Configure the cameras.	For detailed information, see *Configuring camera on page 180*.
	19	Configure the external sensors.	For detailed information, see *External sensor on page 343*.
	20	Calibrate the cameras.	For detailed information, see *Calibrating camera on page 273*.
	21	Calibrate the conveyors or indexed work area.	For detailed information, see *Calibrating linear conveyor on page 185*, *Calibrating circular conveyor on page 217*, *Calibrating indexed work area on page 257*.
	22	Verify the calibrations.	For detailed information, see *Verifying conveyor calibrations on page 271*.
	23	Add a vision model.	For detailed information, see *Adding vision model on page 283*.
	24	Start the production.	For detailed information, see *Starting production on page 319*.

4.2 Frame relationship

4.2.1 What is a coordinate system?

Overview

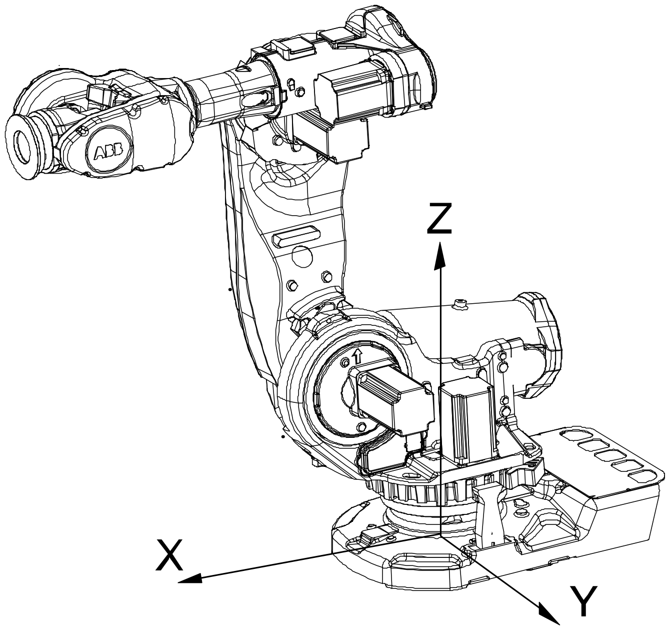
A coordinate system defines a plane or space by axes from a fixed point called the origin. Robot targets and positions are located by measurements along the axes of coordinate systems.

A robot uses several coordinate systems, each suitable for specific types of jogging or programming.

- The *base coordinate system* is located at the base of the robot. It is the easiest one for just moving the robot from one position to another. See *The base coordinate system on page 92* for more information.
- The *world coordinate system* that defines the robot cell, all other coordinate systems are related to the world coordinate system, either directly or indirectly. It is useful for jogging, general movements and for handling stations and cells with several robots or robots moved by external axes. See *The world coordinate system on page 93* for more information.
- The *user coordinate system* is useful for representing equipment that holds other coordinate systems, like work objects. See *The user coordinate system on page 94* for more information.
- The *work object coordinate system* is related to the work piece and is often the best one for programming the robot. See *The work object coordinate system on page 95* for more information.
- The *tool coordinate system* defines the position of the tool the robot uses when reaching the programmed targets. See *The tool coordinate system on page 96* for more information.

Continues on next page

The base coordinate system



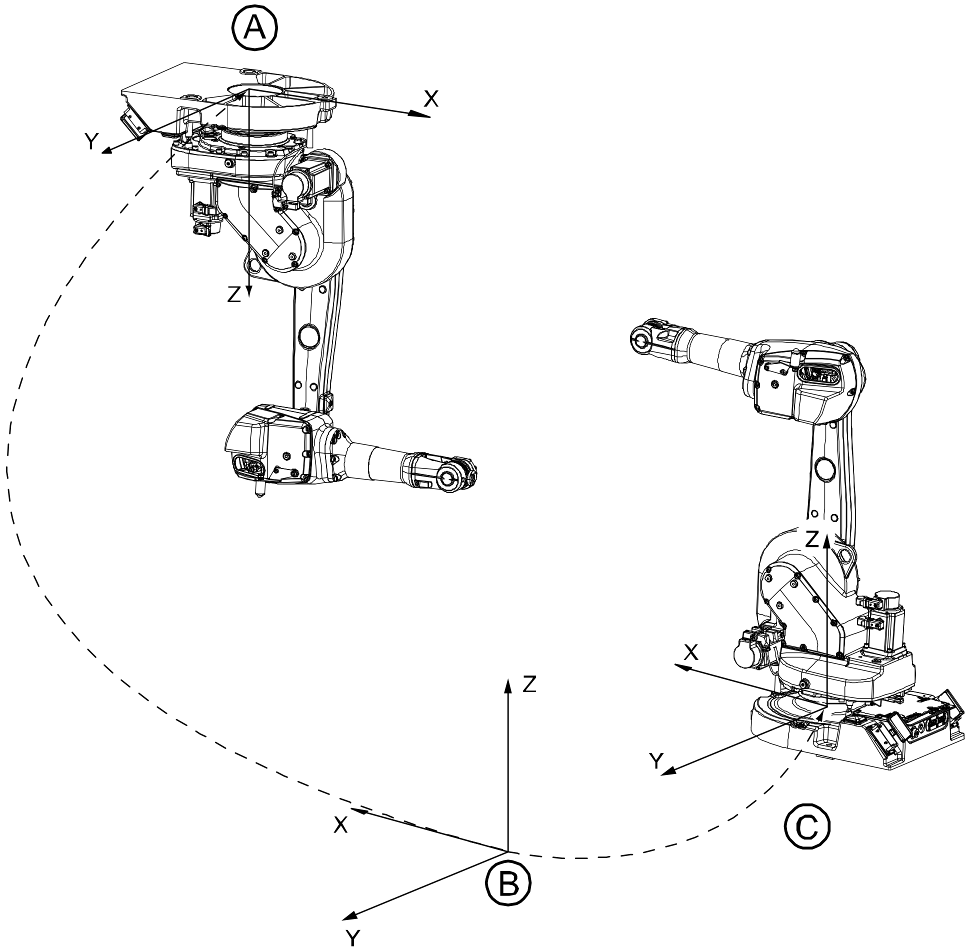
xx0300000495

The base coordinate system has its zero point in the base of the robot, which makes movements predictable for fixed mounted robots. It is therefore useful for jogging a robot from one position to another. For programming a robot, other coordinate systems, like the work object coordinate system are often better choices. See *The work object coordinate system on page 95*for more information.

When you are standing in front of the robot and jog in the base coordinate system, in a normally configured robot system, pulling the joystick towards you will move the robot along the X axis, while moving the joystick to the sides will move the robot along the Y axis. Twisting the joystick will move the robot along the Z axis.

Continues on next page

The world coordinate system



en0300000496

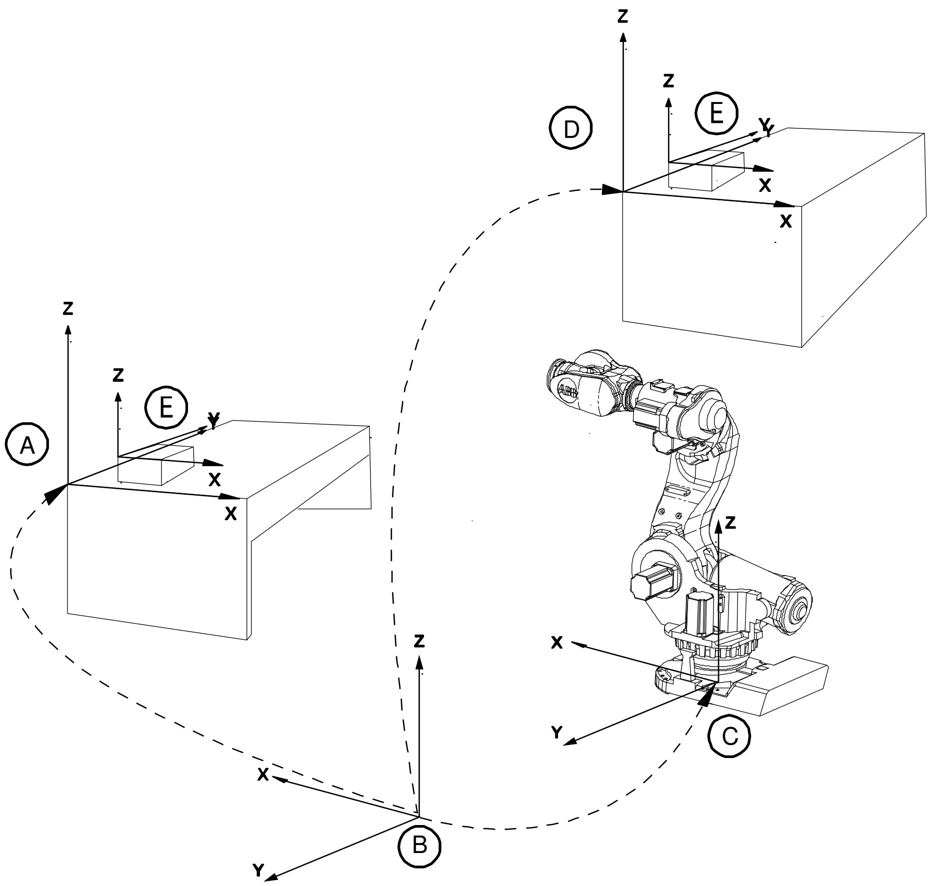
A	Base coordinate system for robot 1
B	World coordinate
C	Base coordinate system for robot 2

The world coordinate system has its zero point on a fixed position in the cell or station. This makes it useful for handling several robots or robots moved by external axes.

By default the world coordinate system coincides with the base coordinate system.

Continues on next page

The user coordinate system



en0400001225

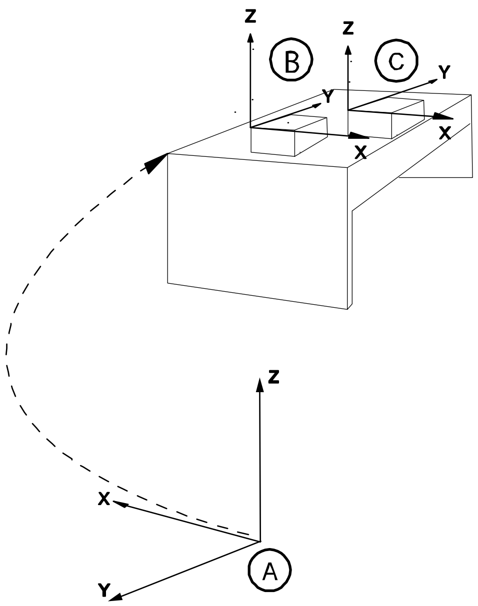
A	User coordinate system
B	World coordinate system
C	Base object coordinate system
D	Moved user coordinate system
E	Work object coordinate system, moved with user coordinate system

The user coordinate system can be used for representing equipment like fixtures, workbenches. This gives an extra level in the chain of related coordinate systems, which might be useful for handling equipment that hold work objects or other coordinate systems.

For information on how to define the user coordinate system, see information about the data type `wobjdata` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Continues on next page

The work object coordinate system



xx0600002738

A	World coordinate system
B	Work Object coordinate system 1
C	Work Object coordinate system 2

The work object coordinate system corresponds to the work piece: It defines the placement of the work piece in relation to the world coordinate system (or any other coordinate system).

A robot can have several work object coordinate systems, either for representing different work pieces or several copies of the same work piece at different locations.

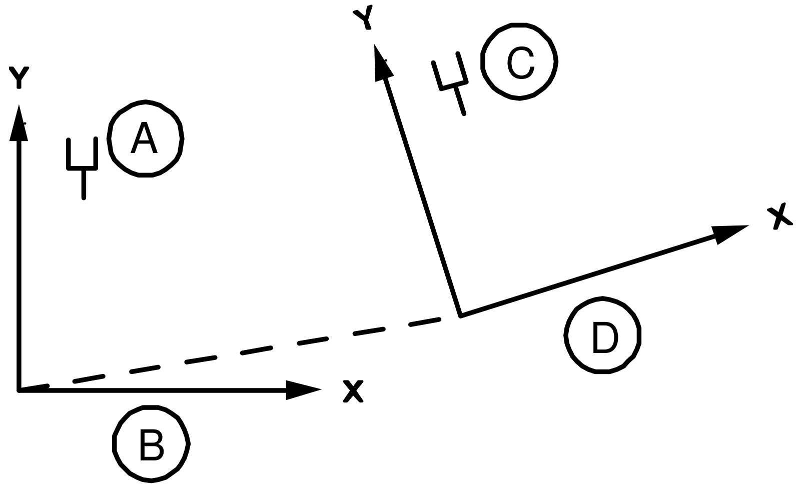
It is in work object coordinate systems you create targets and paths when programming the robot. This gives a lot of advantages:

- When repositioning the work piece in the station you just change the position of the work object coordinate system and all paths are updated at once.
- Enables work on work pieces moved by external axes or conveyor tracks, since the entire work object with its paths can be moved.

For information on how to define the work object coordinate system, see information about the data type `wobjdata` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Continues on next page

The displacement coordinate system



en0400001227

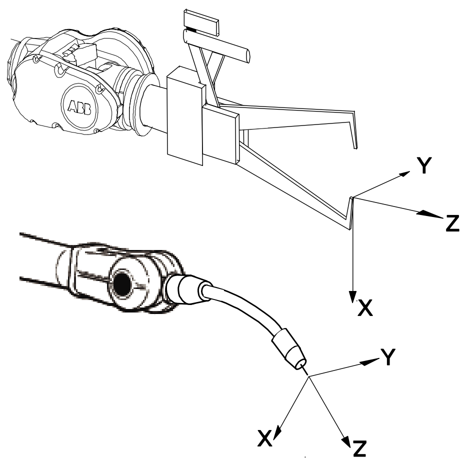
A	Original position
B	Object coordinate system
C	New position
D	Displacement coordinate system

Sometimes, the same path is to be performed at several places on the same object, or on several work pieces located next to each other. To avoid having to reprogram all positions each time a displacement coordinate system can be defined.

This coordinate system can also be used in conjunction with searches, to compensate for differences in the positions of the individual parts.

The displacement coordinate system is defined based on the work object coordinate system.

The tool coordinate system



en0300000497

The tool coordinate system has its zero position at the center point of the tool. It thereby defines the position and orientation of the tool. The tool coordinate system

Continues on next page

is often abbreviated TCPF (Tool Center Point Frame) and the center of the tool coordinate system is abbreviated TCP (Tool Center Point).

It is the TCP the robot moves to the programmed positions, when executing programs. This means that if you change the tool (and the tool coordinate system) the robot's movements will be changed so that the new TCP will reach the target.

All robots have a predefined tool coordinate system, called `tool0`, located at the wrist of the robot. One or many new tool coordinate systems can then defined as offsets from `tool0`.

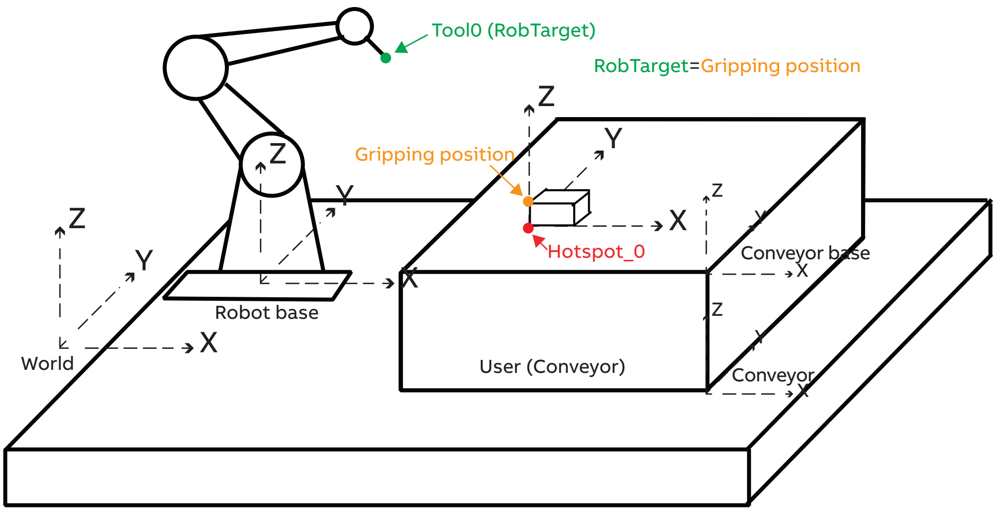
When jogging a robot the tool coordinate system is useful when you don't want to change the orientation of the tool during the movement, for instance moving a saw blade without bending it.

For information on how to define the tool coordinate system, see information about the data type `tooldata` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

4.2.2 Frame relationship in PickMaster® Twin

Overview

The section describes the definition of the coordinate systems regarding conveyor in PickMaster PowerPac solution.



xx2100002290

World frame

World frame is the fundamental frame in a PickMaster PowerPac solution. The location of all the other components like robot, conveyor etc. are expressed in this frame.

Local frame

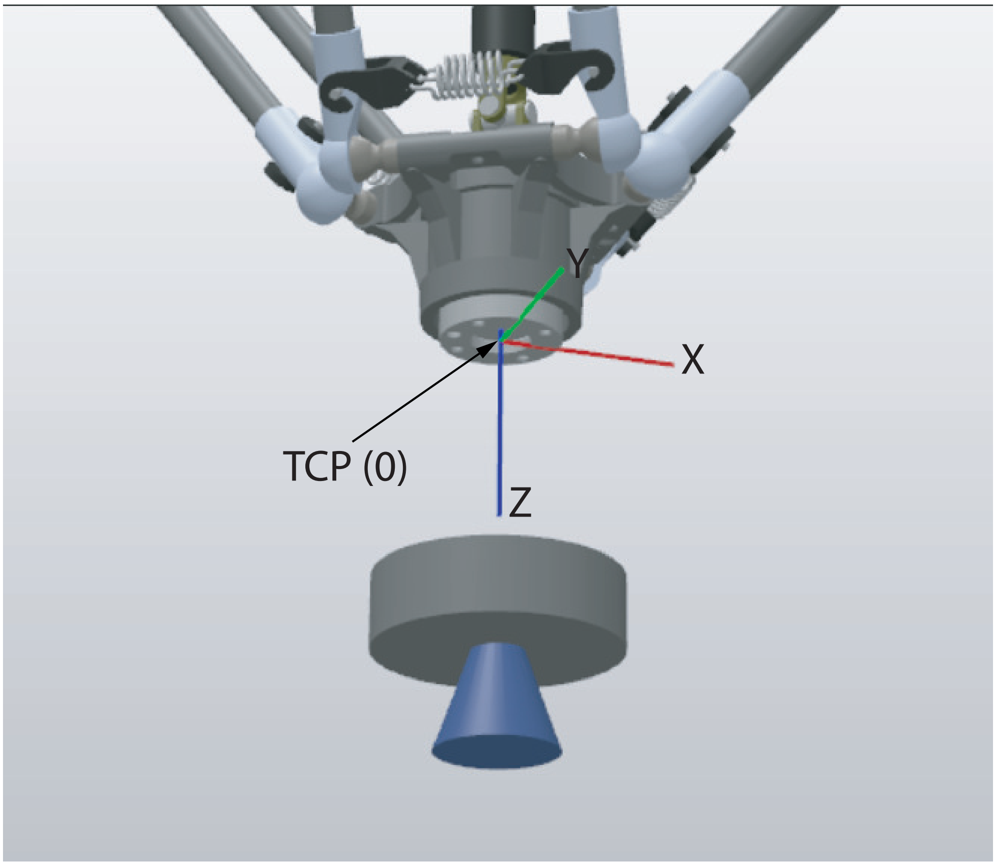
All objects have coordinate systems of its own called the local coordinate system.

Object dimensions are defined with respect to this coordinated system. When the object's position is referred from other coordinate systems like WCS, the local origin of the object is used as the point of reference.

Continues on next page

TCP(0)

Tool center position (0) is the origin position of the tool coordinate system which is expressed in the wrist coordinate system (tool0).



xx2200001148

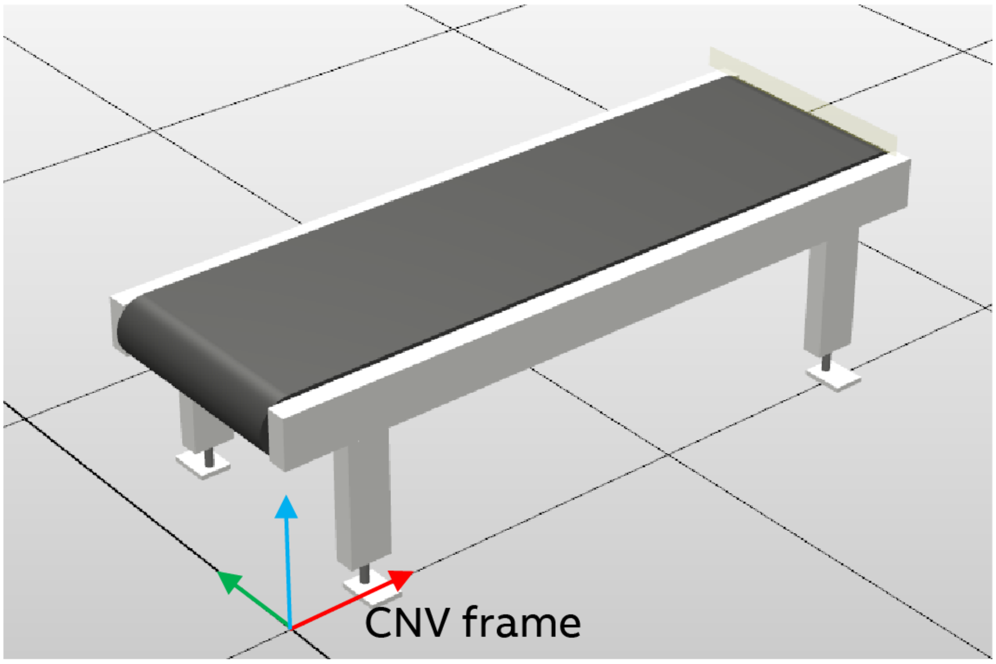
Continues on next page

Conveyor frame

For linear conveyor

A frame that is located at the bottom corner of a linear conveyor as conveyor frame. This frame is fixed relative to the conveyor. The location of a conveyor is defined as the distance (3 dimensional) between the conveyor frame and the world frame expressed in the world frame. The orientation of a conveyor is defined as the angles between the conveyor frame and the world frame expressed in the world frame.

Conveyor frame is used to define where the conveyor is in a PickMaster PowerPac solution but is not directly used in robot controller system.

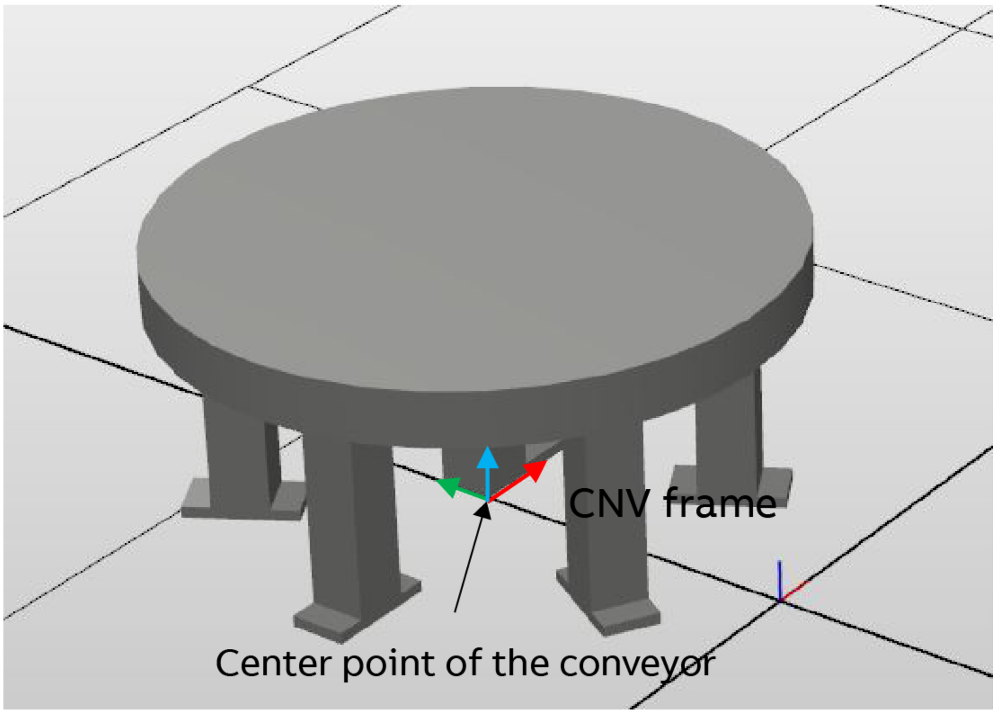


xx2400000394

Continues on next page

For circular conveyor

A frame that is attached to the bottom center of a circular conveyor as conveyor frame. This frame is fixed relative to the conveyor. The location of a conveyor is defined as the distance (3 dimensional) between the conveyor frame and the world frame expressed in the world frame. The orientation of a conveyor is defined as the angles between the conveyor frame and the world frame expressed in the world frame. Conveyor frame is used to define where the conveyor is in a PickMaster PowerPac solution but is not directly used in robot controller system.



xx2400000395

Hotspots frame

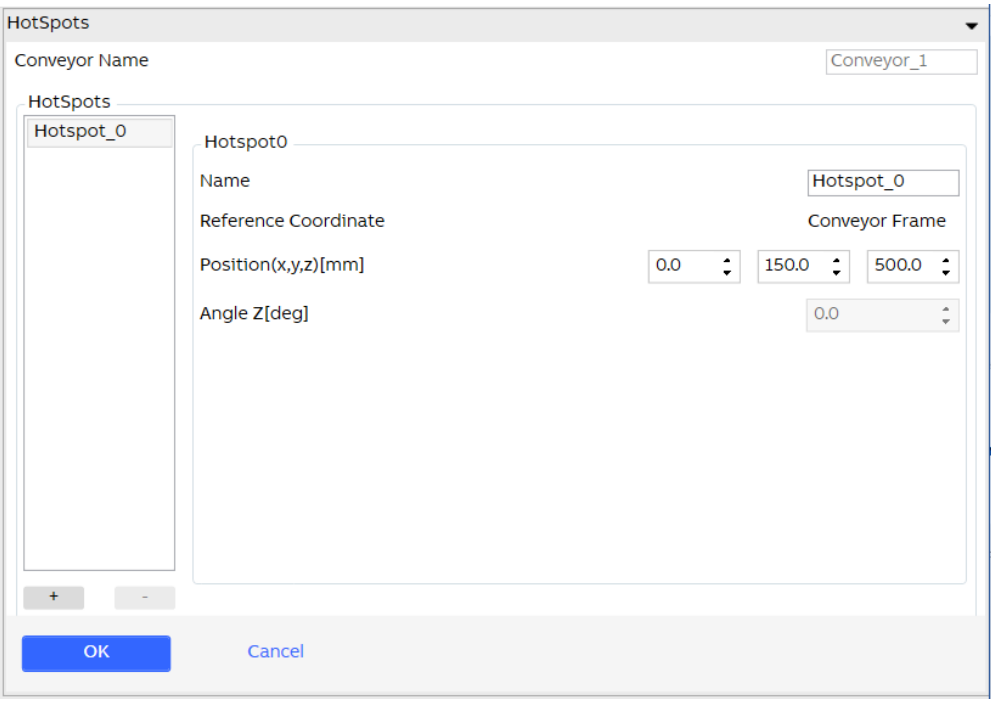
Hotspots is a frame attached to a conveyor but can be configured by user and is expressed in the conveyor frame.

It is where the item or container is generated in solution or come out in emulation.

The predefined value of x, y, z and angle Z indicate where the items or containers come out in a hotspots frame.

Continues on next page

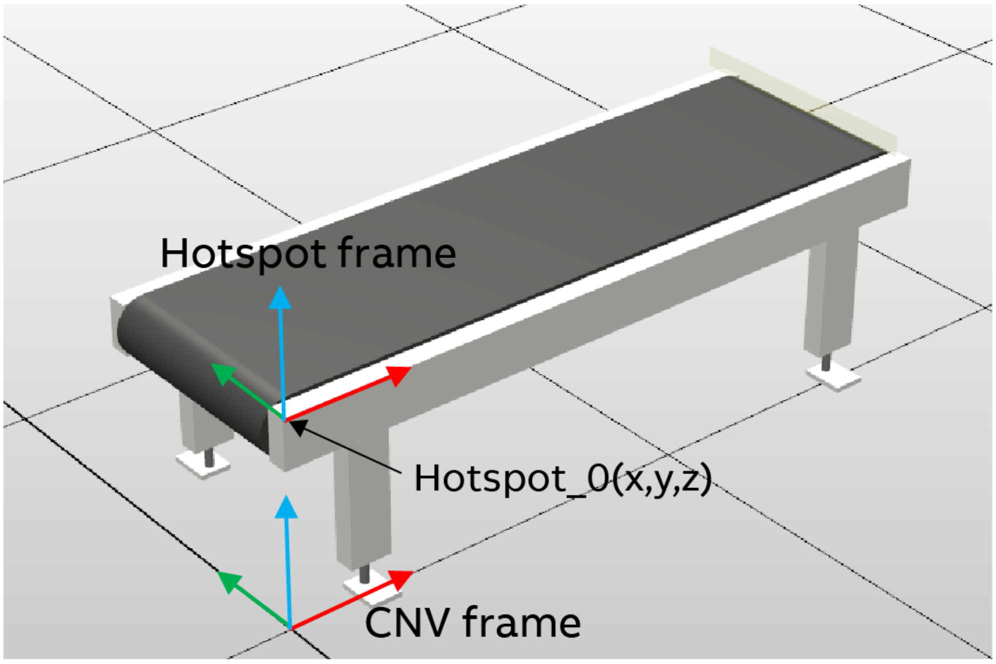
Define the location of the hotspots from the tree view for each conveyor.



xx2400000398

For linear conveyor

The axes of the hotspots frame are always parallel to the axes of the conveyor frame and the location can be configured. The orientation of hotspots frame cannot be configured.

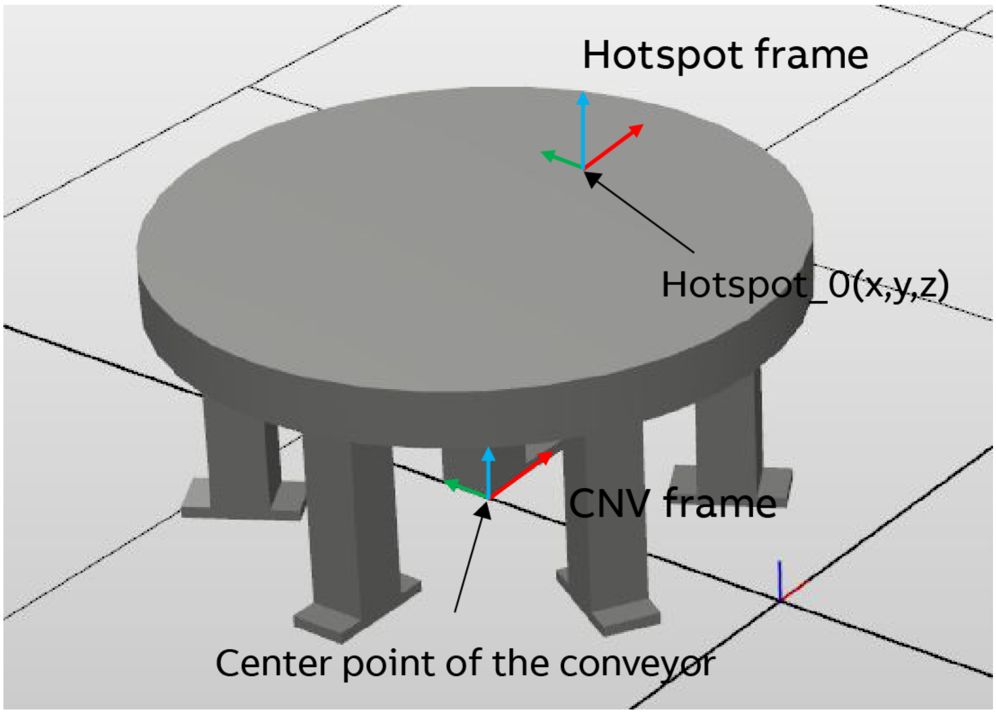


xx2400000396

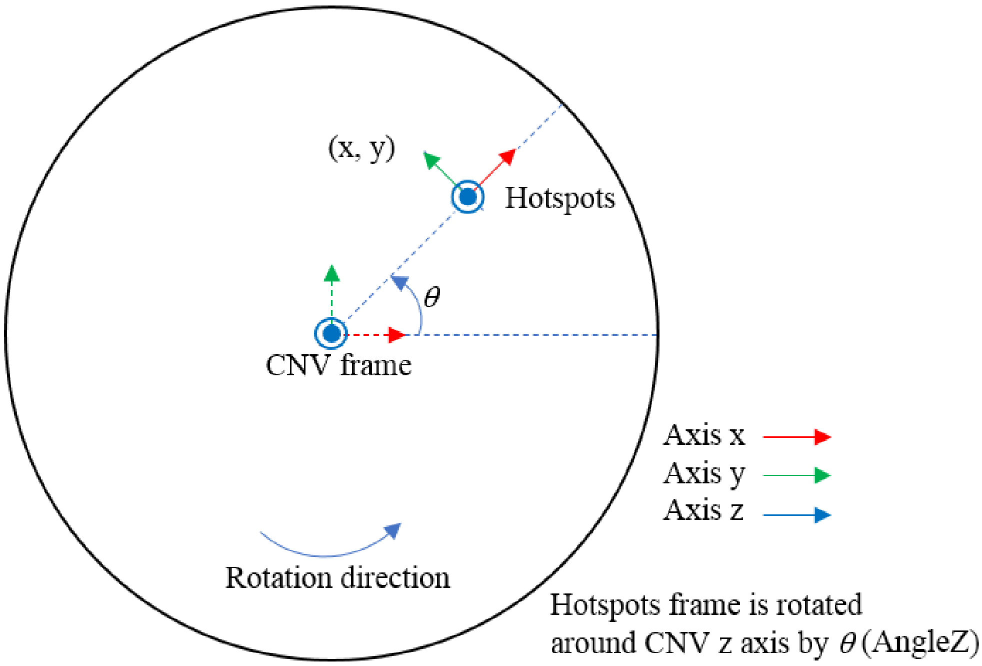
Continues on next page

For circular conveyor

The X axis of the hotspots frame is always along a radius of the circular conveyor and points outwards. The Z axis of the hotspots frame is parallel to that of the conveyor frame.



xx2400000397

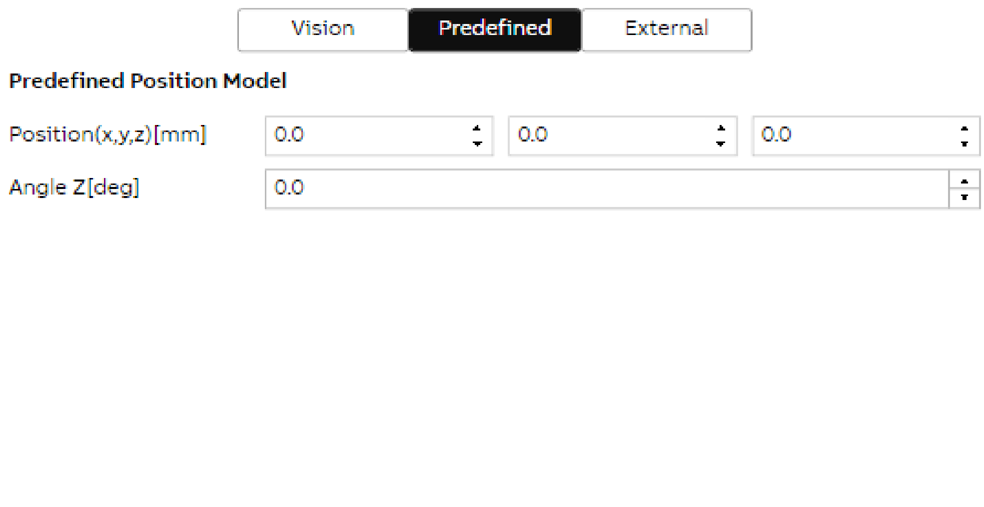


xx2100002385

Continues on next page

Defining the generating position and angle within the hotspot

The user can define the generating position and angle on the plane of the items or containers within the hotspot frame when define the item or container.



xx2100002389

> **Note**
>
> Modify the angle Z of generating item or container in hotspot frame may cause mis-trigger when use an I/O sensor as trigger type in circular conveyor. This may cause mis-picking.

Conveyor base frame

Conveyor base frame is to define a conveyor's location and orientation relative to a robot's base frame. The concept is from ABB conveyor tracking product.

This frame is to tell the robot where the conveyor is and is used to express all the items on the conveyor. To let a robot "know" where an item is, first the conveyor base frame must be defined, and then the items location and orientation need to be detected by certain sensor and is expressed in the conveyor base frame. Conveyor base frame is directly used to calculate the location and orientation of items but not explicitly used in the PickMaster PowerPac solution. For simulation the conveyor base frame is decided by clicking the calibration button in the

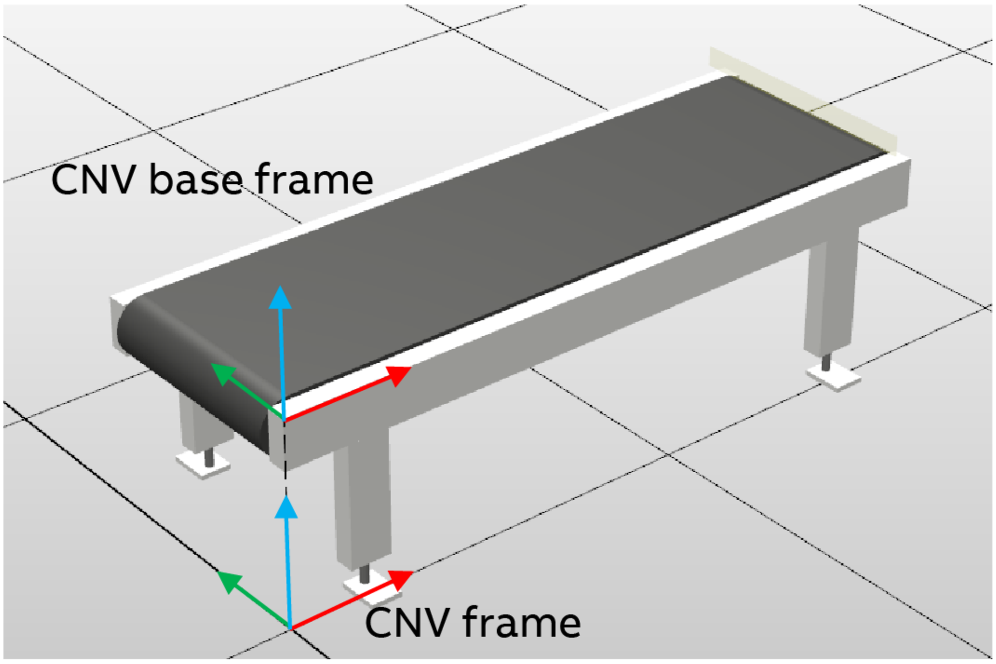
Continues on next page

PickMaster PowerPac. For real system the conveyor base frame is decided by certain measurements in real world.

For linear conveyor

The X, Y and Z axes of a linear conveyor are always parallel to those of conveyor frame respectively.

The location of the conveyor base frame is different depending on the source type and trigger setting for the conveyor.



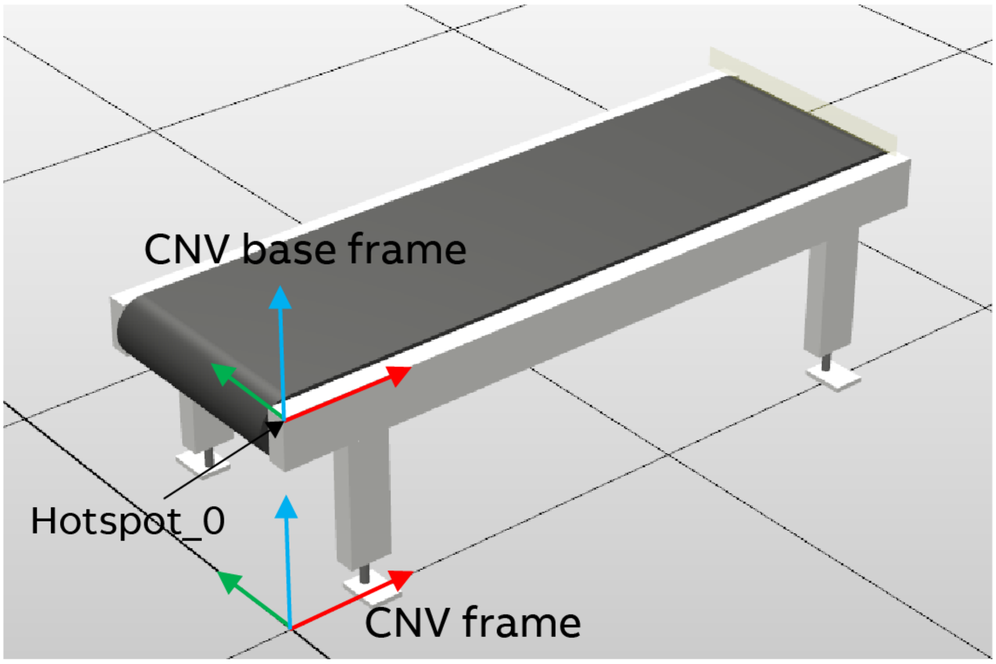
xx2100002388

Continues on next page

Source type as Predefined with Trigger Setting as Distance (No camera and I/O sensor)

X, Y and Z axes are parallel to those of conveyor frame respectively.

The location of conveyor base frame is at Hotspot_0.

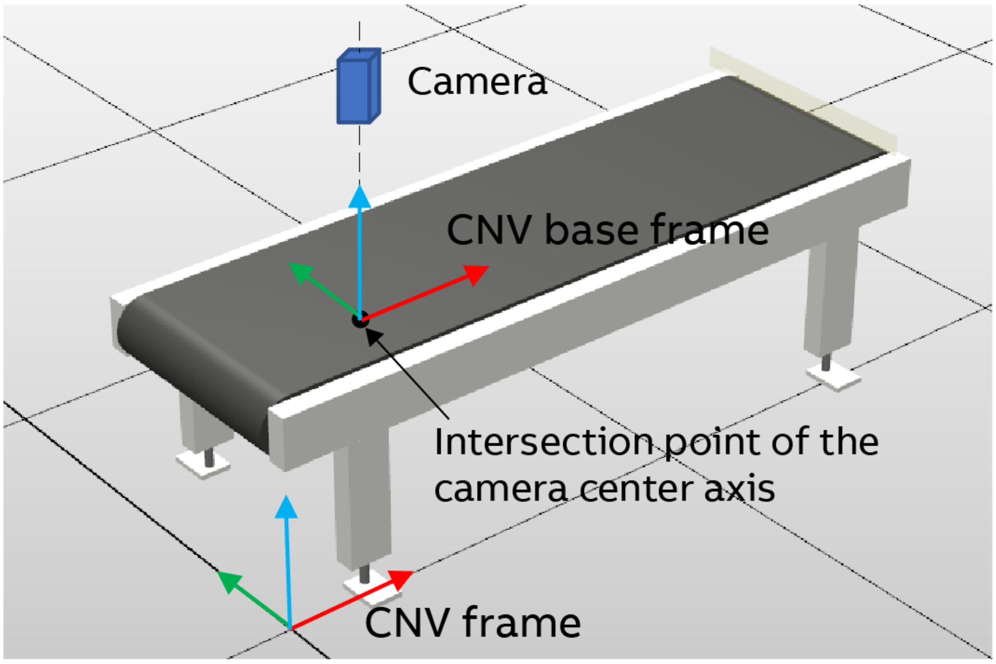


xx2100002469

Source type as Vision with Trigger Setting as Distance or I/O (Camera is used)

X, Y and Z axes are parallel to those of conveyor frame respectively.

The location of conveyor base frame is at the intersection point of the camera center axis and the top surface of the conveyor.



xx2100002470

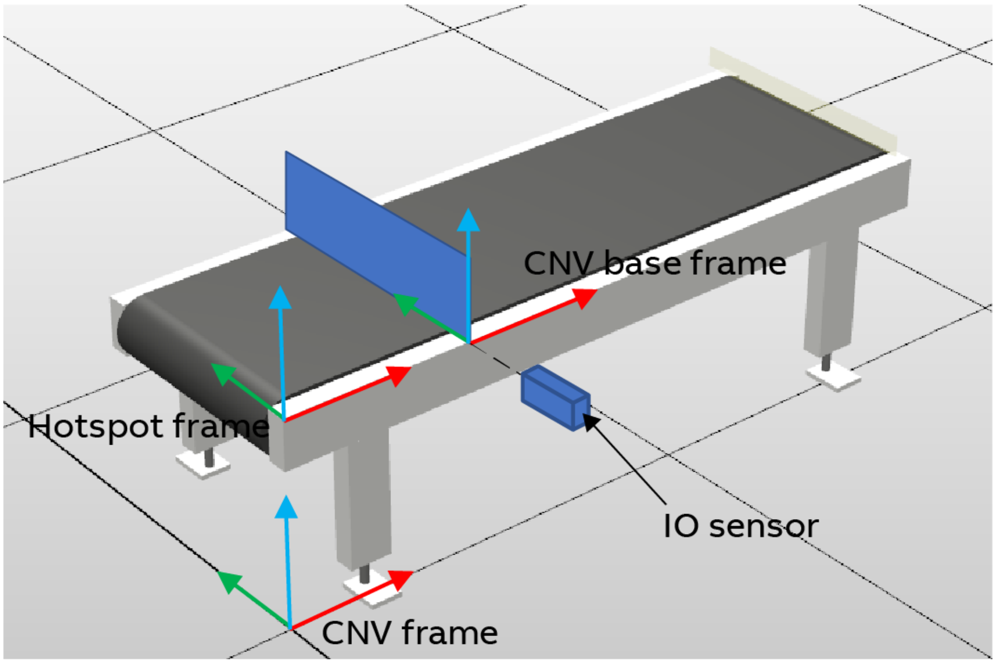
Continues on next page

Source type as Predefined with Trigger Setting as I/O (I/O sensor and predefined source type)

X, Y and Z axes are parallel to those of conveyor frame respectively.

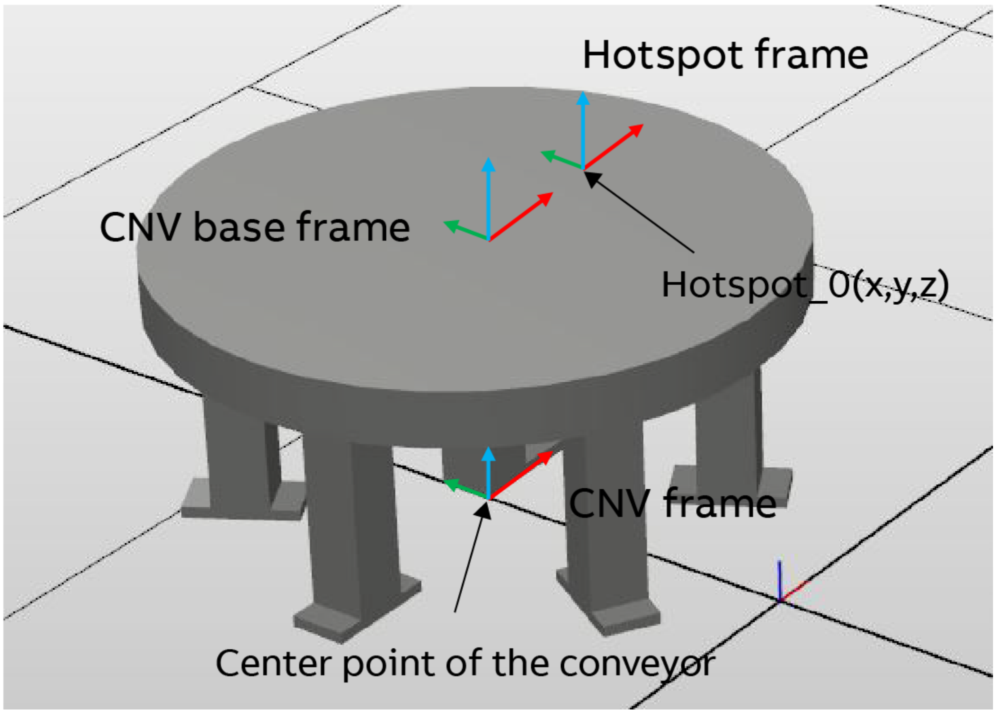
Location along X of conveyor frame is determined by I/O sensor.

Locations along Y and Z of conveyor frame are determined by hotspot_0.



xx2100002471

For circular conveyor rotating counter clockwise



xx2100002386

Continues on next page

Note

The Z axis of conveyor base frame will be defined the direction of positive rotation using the right-hand-rule.

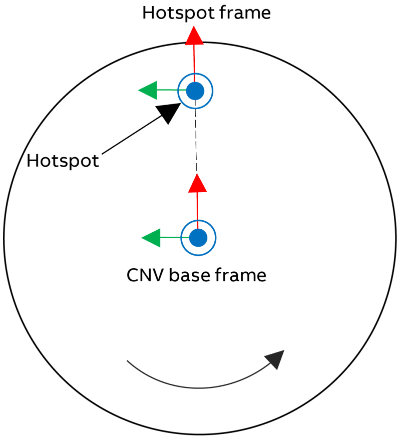
Tip

I/O sensor on a circular CNV is always on a radius and points inward.

Source type as Predefined with Trigger Setting as Distance (No camera and I/O sensor)

The base frame X points to hotspot_0.

Location along Z if conveyor frame is determined by hotspot_0.

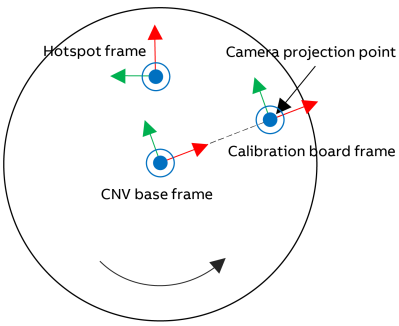


xx2100002466

Source type as Vision with Trigger Setting as Distance (Camera is used)

The base frame X points to the intersection point of the camera center axis and the top surface of the conveyor.

Location is on the top surface of the conveyor.



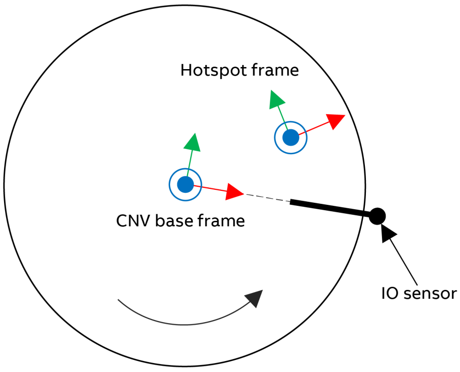
xx2100002467

Continues on next page

Source type as Predefined with Trigger Setting as I/O (I/O sensor and predefined source type)

The base frame X points to the direction of the I/O sensor.

Location along Z of conveyor frame is determined by hotspot_0.



xx2100002468

For circular conveyor rotating clockwise

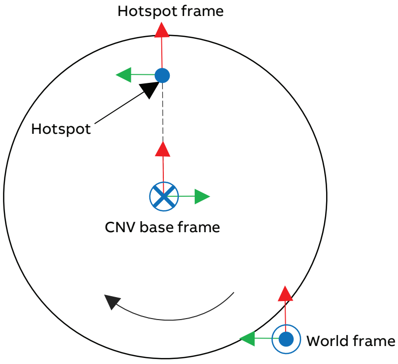
> **Note**
>
> The Z axis of conveyor base frame will be defined the direction of positive rotation using the right-hand-rule.

> **Tip**
>
> I/O sensor on a circular CNV is always on a radius and points inward.

Source type as Predefined with Trigger Setting as Distance (No camera and I/O sensor)

The base frame X points to hotspot_0.

Location along Z if conveyor frame is determined by hotspot_0.



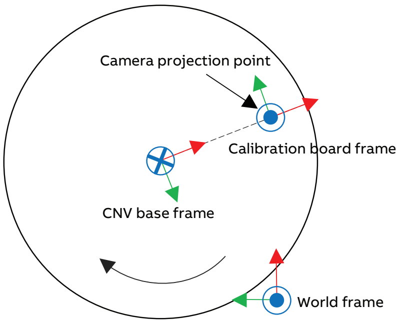
xx2400000402

Continues on next page

Source type as Vision with Trigger Setting as Distance (Camera is used)

The base frame X points to the intersection point of the camera center axis and the top surface of the conveyor.

Location is on the top surface of the conveyor.

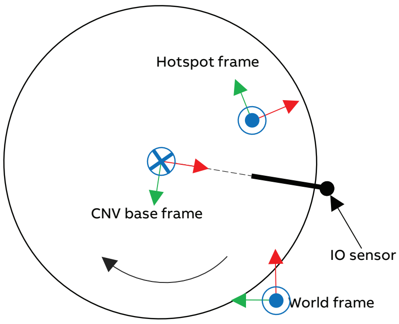


xx2400000403

Source type as Predefined with Trigger Setting as I/O (I/O sensor and predefined source type)

The base frame X points to the direction of the I/O sensor.

Location along Z of conveyor frame is determined by hotspot_0.



xx2400000404

For more information about base frame adjustment, see *Position generator on page 130*.

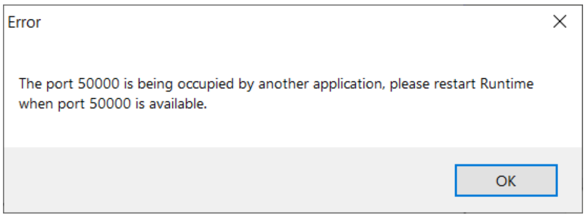
4.3 Setting up Solution with Layout and Process in virtual Runtime (VRT)

4.3.1 Solution

Tip

The PickMaster® Runtime (VRT and RRT) is defined to use 50000 port. If 50000 port is occupied by other program, you will have this warning and not be able to connect to Runtime,

Release the 50000 port and restart the PickMaster® Runtime.



xx2100000868

Use this procedure to release the 50000 port:

1. Enter the command `netstat -aon|findstr "50000"` in the CMD window.
2. The process that occupies port 50000 will be listed in the window. Obtain the PID code of the process.
3. Find the process corresponding to this PID in the task manager and close the it (Make sure that this process is allowed to be closed on this computer).
4. Restart PickMaster® Runtime and connect.

Note

When the **SSL** dialog box pops up during the first operation of the PickMaster PowerPac, click **Yes**.

Otherwise the PickMaster PowerPac cannot work normally.

Note

When the model lacking dialog box pops up, click **Yes** to download the models. Otherwise the solution cannot work normally.

Downloading models may take several minutes.

Opening a solution

The user can create a new solution or open an existing solution from the **File** ribbon tab.

For more information, see *File on page 69*

4.3.2 Layout

4.3.2.1 Controller

Overview

This section describes how to add and modify a controller.

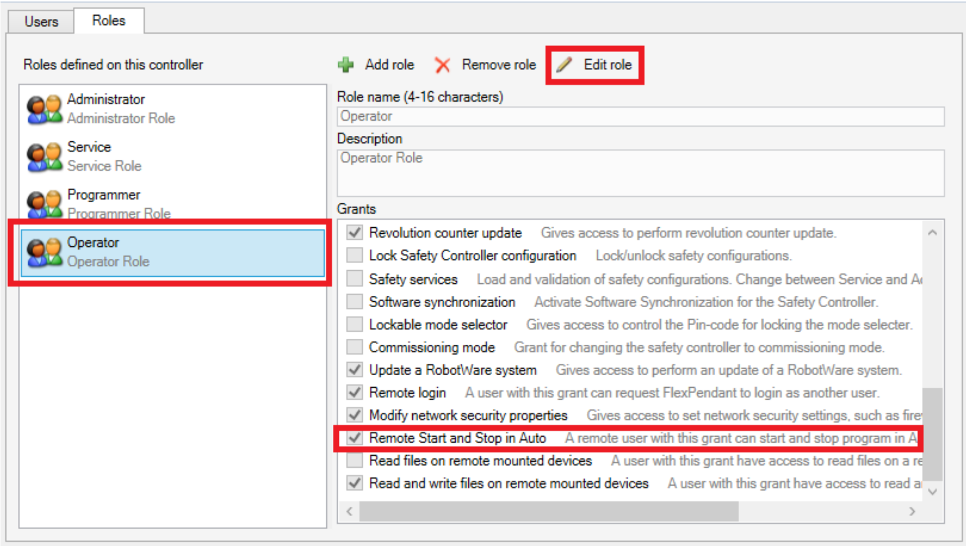
Create a controller

For more information on how to create a controller, refer to *Operating manual - RobotStudio*.

WARNING

When creating an OmniCore controller, the **Remote Start and Stop in Auto** must be selected.

Otherwise, due to RobotWare authority restrictions, the controller cannot be started remotely through PMPP and PMOP. In this case, simulation and production CANNOT operate normally.



xx2100000051

Note

The controller must be created before adding to solution.

Adding a controller

Click **Controller** on the ribbon to add a controller in the solution.

Continues on next page

The following table provides details about the **Controller** adding dialog box.

> **Note**
>
> Only the **Controller** that is created before this page is opened can be found in the **Virtual Controllers**.
>
> If a new **Controller** is created, the user need to refresh the **Controller** dialog box.

> **Note**
>
> It is recommended to calibrate the solution when its virtual controller is used in other solution before simulation.
>
> If different solutions use the same virtual controller, any modification to the controller of one solution will affect other solutions. This will cause unexpected and misleading behavior of other solutions.

	Description
Location	**Location** specify the location and folder of your PC where the required controller systems are stored.
Manage...	Manage the robot system.
Virtual Controllers	Lists the systems found in the selected **system** folder.
Reset system(I-start)	The controller will reset when this is selected. **Note** All parameters and configuration will be restored to factory values.
Import new libraries	Add the predefined robot to the PickMaster PowerPac.
Use existing station librar-ies	Open an existed robot from the **RobotStudio**.
Sync RAPID program to station	Sync the **RAPID** program to the solution.

Procedure

On the PickMaster PowerPac ribbon-tab, click **Layout**.

Use this procedure to add a controller:

> **Note**
>
> You can only add the existing controller in the system to the solution. For more information about create controller, see *Create a controller on page 112*.

1. On the ribbon-tab, click **Controller**.

   The **Controller** adding dialog box is opened.
2. To add a folder to the **Location** list, click **...** button and then browse and select the folder to be added.

Continues on next page

3 The **Virtual Controllers** lists the virtual controller systems found in the selected system folder. Click a system to select it for the new solution.

4 Select the required check boxes in **Options**.

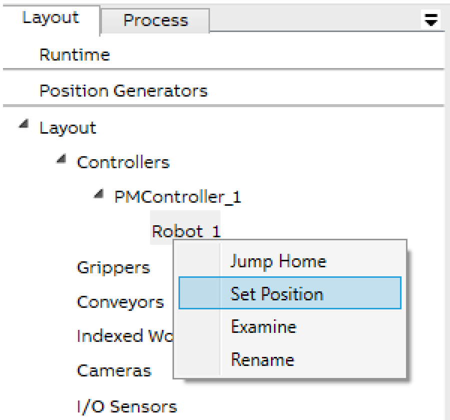
> **Note**
>
> A virtual controller system that has been modified using the **Modify System** function of the System Builder must be restarted with the **Reset System** option for the changes to take effect.

5 In the dialog box, click **OK** to add the selected controller to the solution.

   The selected controller is added into the solution. The new added controller shows up in the **Layout** window **Controller** list.

6 In the **Robot** window, enter numbers in the **Position X Y Z (mm)** text box and **Orientation (deg)** text box according to your requirements.

7 Click **OK**.

> **Note**
>
> The position can be adjusted when the other parts are added.
>
> Right click on the **Robot1** in the **Controller** list.
>
> 
>
>
> xx2100001532

4.3.2.2 Gripper

Overview

This section describes how to add a gripper.

Adding a gripper

Click **Gripper** on the ribbon to add a gripper in the solution.

The following table provides details about the **Gripper** setting dialog box.

	Description
Import	Import a pre-defined gripper from the library or upload an user defined gripper to the library. **Note** To upload an user defined gripper, click **Add to library**, browse to the local folder and select the `*.rslib` file, the gripper will be added to the library automatically.
Gripper Name	Set the name of the gripper.
Controller	Select a controller from the Controller list.
Available Robot	Select a robot from the Robot list.
Reference Coordinate	Select the reference coordinate for the gripper.
Position XYZ(mm)	Set the position of the gripper.
Orientation XYZ(deg)	Set the orientation of the gripper.
Mass Setting	For more details, see following section.
Activator Setting	For more details, see following section.

Mass Setting

Mass Setting	Description
Use Default	Use default setting for the mass setting.
Mass	Type the mass of the tool in the Mass (kg) field.
Center of gravity	Type the coordinates of the center of gravity.
Inertia	Type the values of the inertia in Inertia (kgm^2).

Activator Setting

Activator Setting	Description
Activator Using	Select the activator to be used.

Continues on next page

Activator Setting	Description
Add button	Add a new activator. **Note** When you need to do multiple pick, you should add enough activators for each pick. For example, if you need to pick four items and then place them, you need to add another three activators besides the default one. To do multiple pick, the Multiple-Pick rapid file in the installation package should be imported to the recipe for the required robots.
Delete button	Delete a selected activator.
Rapid tool data	Select a RAPID tool data from **Tool data**. The selected tooldata shall be used by the RAPID program when picking with this activator. **Note** The RAPID program needs to be updated if more than one activator is used. For more details see, *Example: Double pick single place on page 414*.
TCP Position	Type the coordinates of the tool center point. The tool center point defines the location on the tool where an item is attached. **Note** The coordinates are applied to the selected tooldata during the simulation.
TCP Orientation	Type the orientation of the tool center point. The TCP orientation defines the desired orientation of the tool while picking up an item. The orientation shall be specified as Euler XYZ angles (degrees). **Note** The orientation is applied to the selected tooldata during the simulation.
Activator Signal Type	Choose to use the default setting or customized setting for the the signal. **Note** The activator signal setting in PickMaster PowerPac must be exactly same with the signal setting in the connected controller. Otherwise the gripper will not pick or place the items in PickMaster PowerPac.
Default Settings	Shows the detailed default setting of the the signal.
Customized Settings	Shows the detailed customized setting of the signal and allows the user to change the signals.

Procedure

Use this procedure to add grippers:

Continues on next page

On the PickMaster PowerPac ribbon-tab, click **Layout**.

1. On the ribbon-tab, click **Gripper**.

   The **Gripper** window opens.
2. In the **Gripper** window, enter a name in the **Gripper Name** text box or use the default one.
3. In the **Gripper** window, use default for the **Mass Setting** and **Activator Setting**.
4. Click **OK**.

4.3.2.3 Conveyor

Overview

This section describes how to add a conveyor.

Adding a conveyor

Click **Conveyor** on the ribbon to add a conveyor in the solution.

The following table provides details about the **Conveyor** setting dialog box.

	Description
Conveyor Name	Set the name of the conveyor. **Tip** Make sure the name is unique in the current task.
Conveyor Type	Select the a liner conveyor or a circular conveyor.
Size (x,y,z)[mm] [i] **/RH Size [mm mm]** [ii]	Define the size of the conveyor. • For linear conveyor, the value should be within: Length: 1,700 mm - 1,000,000 mm Width: 110 mm - 1,000,000 mm Height: 210 mm - 1,000,000 mm • For circular conveyor, the value should be within: Radius: 251 mm - 1,000,000 mm Height: 210 mm - 1,000,000 mm
Direction [ii]	Select the rotating direction of the circular conveyor.
Reference Coordinate	Select the reference coordinate for the conveyor.
Position XYZ(mm)	Set the position of the conveyor.
Orientation XYZ(deg)	Set the orientation of the conveyor.

Note

If a circular conveyor and camera or I/O sensor are used at the same time, the camera or I/O sensor MUST be set between the conveyor's hotspots and the first robot in the rotation direction.

Otherwise the robots may miss the items.

Procedure

On the PickMaster PowerPac ribbon-tab, click **Layout**.

Use this procedure to add conveyors:

1. On the ribbon-tab, click **Conveyor**.

   The **Conveyor** window opens.
2. In the **Conveyor** window, enter a name in the **Conveyor Name** text box or use the default one.
3. In the **Conveyor** window, select a type as liner or circular in the **Conveyor Type** drop-down list.

Continues on next page

4. If you select a liner conveyor, in the **Conveyor** window, enter numbers in the **Size (x,y,z)[mm]** text box to define the size of the conveyor according to your requirements.
5. If you select a circular conveyor, in the **Conveyor** window, enter numbers in the **RH Size(mm)** text box to define the size of the conveyor according to your requirements.
6. In the **Conveyor** window, enter numbers in the **Position X Y Z (mm)** text box and **Orientation (deg)** text boxes to define the location of the conveyor according to your requirements.
7. Click **OK.**

4.3.2.4 Camera

Overview

This section describes how to add a camera.

Adding a camera

Click **Camera** on the ribbon to add a camera in the solution.

The following table provides details about the **Camera** setting dialog box.

	Description
Name	Set the name of the camera. **Tip** Make sure the name is unique in the current task.
Attached to Conveyor/Index	Choose the conveyor if the camera shall be attached to a conveyor.
Entry (mm)	Type an entry limit for the visible area below the camera along a conveyor. A negative value is used if the visible area starts upstreams from the camera location.
Exit (mm)	Type an exit limit for the visible area below the camera along a conveyor. A positive value is used if the visible area ends downstreams from the camera location.
Enable vision width	Select this to enable a width limitation of the visible area. **Note** Only when the **Enable vision width** checkbox is selected, the **Left (mm)** and **Right (mm)** values would be implemented to the setting. **Note** If the camera is attached to a circular conveyor, the **Enable vision width** checkbox is selected as default and cannot be disabled.
Left (mm)	Type a limit value for the left side of the visible area. A negative value is used if the visible area ends on the left side of the camera location (from an upstream viewpoint). **Note** The robot may catch air or miss some items when the **Left (mm)** and **Right (mm)** are not correctly set.
Right (mm)	Type a limit value for the right side of the visible area. A positive value is used if the visible area ends on the right side of the camera location (from an upstream viewpoint). **Note** The robot may catch air or miss some items when the **Left (mm)** and **Right (mm)** are not correctly set.

Continues on next page

	Description
Reference Coordinate	Select the reference coordinate for the camera.
Position(X,Y,Z)[mm]	Set the position of the camera.
Orientation[deg]	Set the orientation of the camera.

Note

The visible area is not limited if the camera is used with an indexed work area .

Note

The camera will not detect any objects created or placed on the other conveyors or indexed work areas.

Note

If a circular conveyor and camera or I/O sensor are used at the same time, the camera or I/O sensor MUST be set between the conveyor's hotspots and the first robot in the rotation direction.

Otherwise the robots may miss the items.

Procedure

On the PickMaster PowerPac ribbon-tab, click **Layout**.

Use this procedure to add cameras:

1. On the ribbon-tab, click **Camera**.

   The **Camera** window opens.
2. In the **Camera** window, enter a name in the **Camera Name** text box or use the default one.
3. In the **Camera** window, choose a conveyor in the **Attached to Conveyor/Index** box to define which conveyor the new camera is attached to according to your requirements.
4. In the **Camera** window, use default for the other settings.
5. Click **OK**.

4.3.2.5 I/O sensor

Overview

This section describes how to add an I/O sensor.

Adding an I/O sensor

Click **I/O sensor** on the ribbon to add an I/O sensor in the solution.

The following table provides details about the **I/O sensor** setting dialog box.

	Description
Name	Set the name of the I/O sensor. **Tip** Make sure the name is unique in the current task.
LH Size[mm]	The height and length of the new I/O sensor.
Attached to Conveyor/Index	Choose the conveyor if the sensor shall be attached to a conveyor.
Reference Coordinate	Select the reference coordinate for the I/O sensor.
Position(X,Y,Z)[mm]	Set the position for the I/O sensor.
Orientation[deg]	Set the orientation of the I/O sensor.

Note

To function correctly, an I/O sensor must not be in contact with other stationary objects, for example, the conveyor.

Note

If a circular conveyor and camera or I/O sensor are used at the same time, the camera or I/O sensor MUST be set between the conveyor's hotspots and the first robot in the rotation direction.

Otherwise the robots may miss the items.

Procedure

On the PickMaster PowerPac ribbon-tab, click **Layout**.

Use this procedure to add I/O sensors:

1. On the ribbon-tab, click **I/O sensor**.

   The **I/O sensor** window opens.
2. In the **I/O sensor** window, enter a name in the **I/O sensor Name** text box or use the default one.
3. In the **I/O sensor** window, enter numbers in the **I/O sensor Height** text box to define the height of the I/O sensor according to your requirements or use default settings.

Continues on next page

4 In the **I/O sensor** window, enter numbers in the **I/O sensor Length** text box to define the length of the I/O sensor according to your requirements or use default settings.
5 In the **I/O sensor** window, choose a conveyor in the **Attached to Conveyor/Index** box to define which conveyor the new camera is attached to according to your requirements.
6 Click **OK**.

4.3.2.6 External sensor

Overview

The **External sensor** is a function that allows the users to have the full control of generating the item positions.

This section describes how to add an external sensor with using any kind of sensing device or a pure virtual software sensor.

Adding an external sensor

Click **External Sensor** on the ribbon to add an external sensor in the solution.

The following table provides details about the **External Sensor** setting dialog box.

	Description
Name	Set the name of the external sensor. **Tip** Make sure the name is unique in the current task.
Attached to Conveyor/Index	Choose the conveyor if the sensor shall be attached to a conveyor.
Reference Coordinate	Select the reference coordinate for the external sensor.
Position(X,Y,Z)[mm]	Set the position for the external sensor.
Orientation[deg]	Set the orientation of the external sensor.

Procedure

On the PickMaster PowerPac ribbon-tab, click **Layout**.

Use this procedure to add external sensors:

1. On the ribbon-tab, click **External Sensor**.

   The **External Sensor** window opens.
2. In the **External Sensor** window, enter a name in the **External Sensor Name** text box or use the default one.
3. In the **External Sensor** window, choose a conveyor in the **Attached to Conveyor/Index** box to define which conveyor the new camera is attached to according to your requirements.
4. Click **OK**.

Advanced function - External sensor

External sensor is an advanced function for programming user. For detailed information, see *External sensor on page 343*.

4.3.2.7 Work area

Overview

This section describes how to add a work area.

Adding a work area

The conveyor work area is an area on the conveyor where the robot picks or places items. One conveyor board is required for each conveyor work area. A robot usually has only one conveyor work area on each related conveyor, but there is no restriction.

Click **Conveyor Work Area** on the ribbon to add a work area in the solution.

The following table provides details about the **Conveyor Work Area** setting dialog box.

> **Note**
>
> All the signals with "*" is a required signal.

	Description
Work Area Name	Set the name of the conveyor work area.
Controller	Select a controller from the list.
Robot	Select a robot from the list.
Conveyor Board	Select a conveyor board from the list.
Conveyor	Select a conveyor from the list.
Work Area Type	Select work area type from the available options. • **Pick**: Select this if the work area is a picking area. • **Place**: Select this if the work area is a placing area.
Selection Index	Select an index to specify the pick or place order in the RAPID program when using more than one pick work area and one place work area with the selected robot
Signal Type	Configure the signals. Select the **Default Settings** check box to use the default signal configuration. It's recommended to use the default setting when working with virtual Runtime. Use the **Customized Settings** options to manage the signals. The signals should be setting as **Customized Settings** accordingly when working with real Runtime. For more information, see *Configuring the I/O on page 175*.

> **Note**
>
> When any of **Controller**, **Robot** or **Conveyor** is changed in work area setting, the user must reopen the recipe setting page to enable the modification.

Continues on next page

Conveyor work area signals

	Description
Conveyor start/stop	Digital output signal. This signal is used if an overflow shall be avoided by letting the conveyor movement be controlled by the work area. The signal goes high when the conveyor shall start moving and goes low when the conveyor shall stop to avoid an overflow.
Queue idle	Digital output signal. This 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	Digital output signal. This signal is high when there is one or more items between the enter and exit limits for the work area.
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, Runtime 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.
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.

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.

Procedure

On the PickMaster PowerPac ribbon-tab, click **Layout**.

Use this procedure to a work area:

1. On the ribbon-tab, click **Conveyor Work Area**.

   The **Conveyor Work Area** window opens.
2. In the **Conveyor Work Area** window, enter a name in the **Work Area Name** text box or use the default one.
3. If you created several controllers, select the required controller in **Controller**.
4. If you created several robots, select the required robot in **Robot**.
5. Select the required conveyor board in **Conveyor Board.**
6. Select the desired conveyor, set required work area type and configure the settings.

Continues on next page

7 Click **OK**.

4.3.2.8 Indexed work area

Overview

This section describes how to add an index work area.

Adding an indexed work area

An indexed work area is a fixed area where a robot picks or places items without conveyor.

Click **Indexed Work Area** on the ribbon to add an indexed work area in the solution.

The following table provides details about the **Indexed Work Area** setting dialog box.

Note

All the signals with "*" is a required signal.

	Description
Work Area Name	Set the name of the indexed work area.
Size	Define the zone of the indexed work area.
Work Area Type	Select work area type from the available options. • **Pick**: Select this if the indexed work area is a picking area. • **Place**: Select this if the indexed work area is a placing area.
Controller	Select a controller from the list.
Robot	Select a robot from the list.
Work object	Select a RAPID work object data (`wobjdata`). The associated `wobjdata` is automatically used with the indexed work area. **Note** No work object calibration is needed. The selected `wobjdata` is automatically updated when a simulation is started.
Selection Index	Select an index to specify the pick or place order in the RAPID program when using more than one pick work area and one place work area with the selected robot.
Reference Coordinate	Select the reference coordinate for the indexed work area.
Position XYZ(mm)	Set the position for the indexed work area.
Orientation XYZ(deg)	Set the orientation of the indexed work area.
Signal Type	Configure the signals. Use the **Customized Settings** options to manage the signals. For more information regarding indexed work area signals see the following section. Select the **Default Settings** check box to use the default signal configuration.

Continues on next page

Indexed work area signals

> **Note**
>
> In production, it is recommended to set the **Queue idle** signal and **Strobe** signal for indexed work area as the same one, and the other signals as blank.

Signal	Description
Robot execution	This optional digital input I/O signal is used to indicate that it is allowed for the robot to execute an item target in the RAPID program. Execution starts when the signal is high and stops when the signal goes low. If the signal goes low, all remaining items in the currently executing scene is dropped, so when the signal goes high again, the item targets for the next scene is executed. The signal must also go low after one scene is finished and then go high again to start executing item targets for the next scene.
Queue idle	Digital output signal. This 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. **Note** If the robot needs to repeat the motion, this signal should be the same with the signal in **Strobe**.
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.
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.
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.

Procedure

On the PickMaster PowerPac ribbon-tab, click **Layout**.

Use this procedure to add a indexed work area:

1. On the ribbon-tab, click **Indexed Work Area.**

   The **Indexed Work Area** window opens.
2. In the **Indexed Work Area** window, enter a name in the **Work Area Name** text box or use the default one.
3. Select the required work area type.
4. Click **OK.**

4.3.2.9 Position generator

Overview

This section describes how to set the position generator of the created solution.

Setting the position generator

Click **Position Generator** to define where and how positions are generated in a solution. The **Position Generator** should be correctly defined before the station can be calibrated.

The following table provides details about the **Position Generator** setting dialog box.

	Description
Available conveyor and indexed work area list	Selects a conveyor or indexed work area in order to set the related relationships.
Source Type	Select the input signal source type: • **Vision:** 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 *Adding vision model on page 283*. **Tip** If the source type is set to **Vision**, all available cameras and related items will be listed in the **Available Camera**. • **Predefined**: If the source type is set to **Predefined**, the positions generated by the position source are statically defined and no camera is used. • **External**: If the source type is set to **External**, an external sensor in the solution together with external position generators are used to define item positions. **Tip** If an indexed work area is used, external sensor function will be disabled.

Continues on next page

	Description
Trigger Setting	Select **Trigger type** to define when to generate new item positions. **Note** If the trigger type is set to **Distance** the trigger distance must be defined in the **Trigger Distance** box in **Operation** setting under **Recipe**. 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. **CAUTION** If the **Predefined** and IO sensor are selected in the recipe, tune the pick location in the **Tuning** for a radial distance of the item to make up the offset. **Tip** If an indexed work area is used, **Trigger Setting** is not available.
Base Frame Adjustment	Adjust the base frame for selected conveyor or indexed work areas. For more information, see *Adjusting the base frame on page 131*.

Procedure

Note

Any modification on the source type or trigger setting requires a new calibration.

Use this procedure to set the **Position Generator**:

1. On the **Layout** in the PickMaster PowerPac tree view, right-click **Position Generator**.
2. Click to choose one conveyor.
3. Set the source type and the trigger setting.
4. If needed, set the virtual base frame data accordingly.
5. Click to select the other conveyor and set for it.
6. Click **OK**.

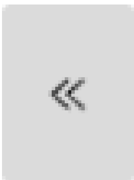
Adjusting the base frame

When the default virtual base frame is inconsistent with the real base frame in the real station, adjust the base frame to ensure the accuracy of the pick and place in production.

The following table provides details about the **Base Frame Adjustment** setting dialog box.

Base Frame Adjustment	Description
Controller	Select the desired conveyor or indexed work area to adjust its base frame.

Continues on next page

Base Frame Adjustment	Description
Virtual Base Frame	Show current virtual base frame data and allows the user to edit the virtual base frame data. **Tip** The virtual base frame data are automatically updated when the virtual station is calibrated. They can also be copied from the real station or manually edited.
Real Base Frame	Show current real base frame data acquired from the real controller. **Tip** The real base frame data CANNOT be changed from PickMaster PowerPac.
 xx2200001993	Synchronize the real base frame data to virtual base frame data.
Display Base Frame	Select to show the base frame on the station view.
Apply	Save and apply the edited virtual base frame data to the virtual controller.
Acquire	Acquire the real base frame data from the real controller. **Tip** The real base frame data only can be acquired when the real Runtime is connected. For more information about connecting to real Runtime, see *Switching to real Runtime on page 169*.

Procedure

Use this procedure to adjust the virtual base frame:

1. Switch to real Runtime.

   For more information, see *Switching to real Runtime on page 169*.
2. On the **Layout** in the PickMaster PowerPac tree view, right-click **Position Generator**.
3. Click to choose the desired conveyor or indexed work area.
4. Click in the **Controller** drop-down list to choose the desired controller.
5. Click **Acquire** to acquire the real base frame data from the real controller.
6. Click the **Sync** button to synchronize the data from real base frame to virtual base frame.
7. Click **Yes.**

Continues on next page

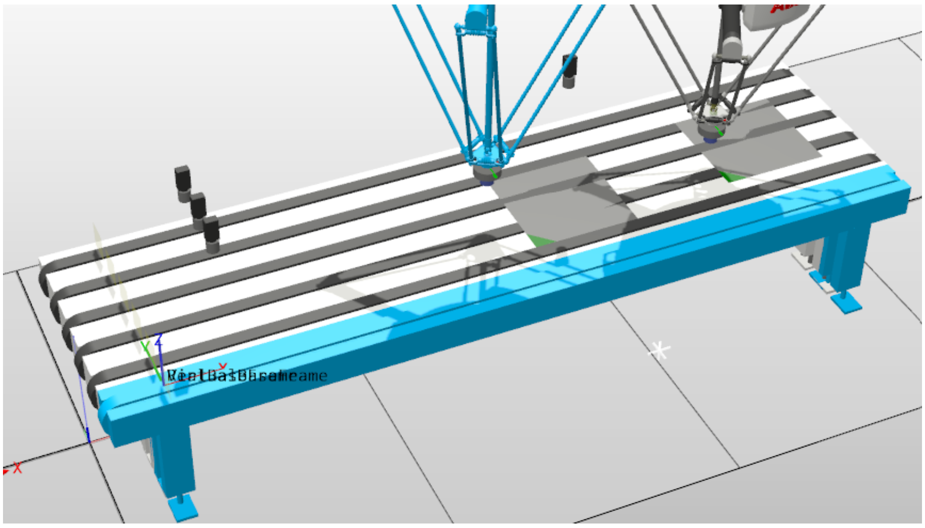
8 Click **Apply**.

9 Click **Yes** in the popped-up message box to save the virtual base frame.

> **Note**
>
> If the user click **No** in this step, the virtual base frame data will not be saved.

10 If needed, click **Yes** in the coming popped-up message box to adjust the station components' position in the station view.

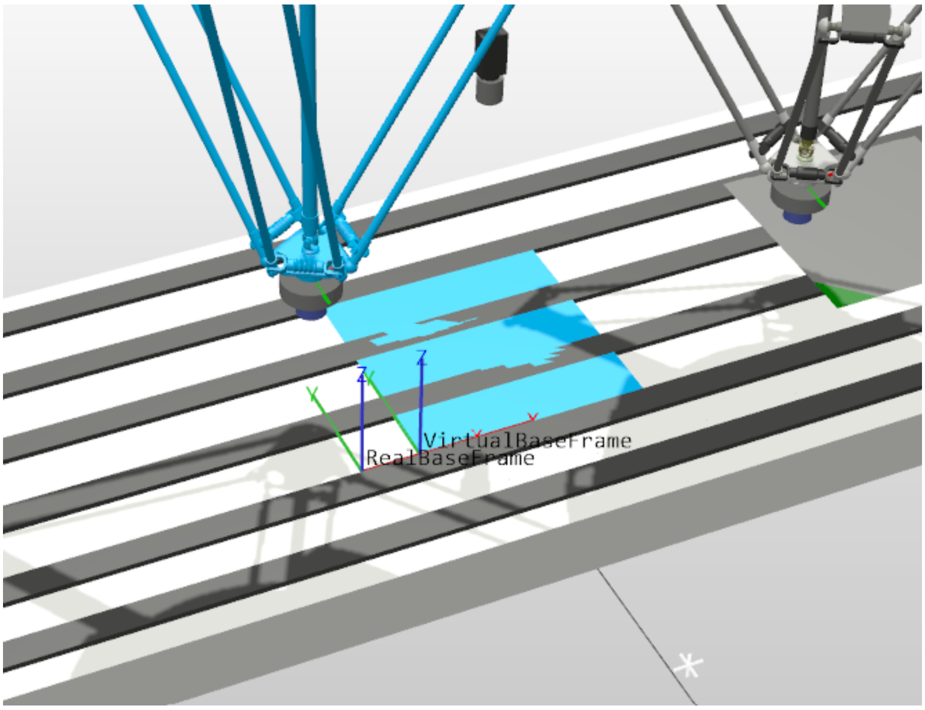
After click **Yes**, the virtual base frame and real base frame will be coincident in the station view.



xx2200002000

Continues on next page

If the user click **No** in this step, the station components will not be moved accordingly.



xx2200001999

11 Click to select the other conveyor and set for it.

12 Click **OK**.

4.3.2.10 Calibration

Overview

This section describes how to calibrate the created solution.

The calibration in PickMaster PowerPac is a prerequisite for running the simulation. The calibration is different with the calibration of the actual hardware (camera, conveyor, IO sensor, etc.). Running this calibration does not mean that the actual hardware calibration has been completed.

The calibration in PickMaster PowerPac is used to establish the relative relationship between the conveyor base frame and the robot base coordinate system in the virtual controller.

If a camera is used for a linear conveyor in the solution, the base frame of the conveyor is directly below the camera after calibration (x is the forward direction). If an IO sensor is used for a linear conveyor, the base frame of the conveyor is located at the IO sensor. If predefined is used with the default setting for a linear conveyor, the base frame of the conveyor overlaps the hotspot0. For more information on frames, see *Frame relationship on page 91*.

Indexed work area calibration is consistent with linear conveyor's calibration.

The calibrated base coordinate system of the circular conveyor belt is located at the center of the conveyor belt, and the x-direction points directly below the camera or along the IO sensor. If the circular conveyor uses a predefined point, the x direction points to a predefined coordinate point(hotspot).

Calibration

On the PickMaster PowerPac ribbon-tab, click **Layout**.

Use this procedure to calibrate:

1. Click **Calibration** on the ribbon-tab. Then it will start to calibrate the created solution automatically.

   The calibration runs automatically.

   > **Note**
   >
   > If the layout in the solution changes, such as changing the camera position or robot position, redo the calibration.

For more details on frames, see *Frame relationship on page 91*.

4.3.3 Process

4.3.3.1 Item

Overview

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.

This section describes how to add an item.

Adding an item

Click **Item** on the ribbon to add an item in the solution.

The following table provides details about the **Item** setting dialog box.

Item Properties tab

Item Properties

	Description
Name	Change the name.
Type	Change the shape of the item. • **Cylinder** • **Box** • **Customized**: import predefined models.
Size(x,y,z)[mm]	Configure the size of the item.

Rapid properties

	Description
Accepted Type	Define the values for accepted item types. The values for the accepted item type are sent to the RAPID program and are supplied with the item targets. For more details see, *GetItmTgt - Get the next item target on page 372*.
Rejected Type	Define the values for rejected item types. The values for the rejected item type are sent to the RAPID program and are supplied with the item targets. For more details see, *GetItmTgt - Get the next item target on page 372*.

Note

If the **Accepted Type** or **Rejected Type** of different items in one solution set as the same value, the **Picking Status** will be influenced.

Continues on next page

Appearance Properties

	Description
Template	**Default Settings** tab: choose one of the preset templates. **Default Name** text box: enter the name for a new template. **Save** icon: save your new template. **Delete** icon: delete your templates. **Tip** If you enter a new template name in the template text box, a new template will be created instead of being renamed. **Note** If you directly modify the appearance of the default template instead of creating a new template, this will modify the default value of the default template. And all items created with default template will be modified too.
Color	Change the color of the new item.
Use Texture	Use a texture image file for the item.
Label Location	Set the location of the label on the item.
Label Picture	Select an image file for the label picture.
Show Contour	Choose to show the contour or not.
Show Orientation Marker	Choose to show the orientation maker or not.
Browse	Select and import a **Customized** model.
Offset [mm]	Set the offset value for the imported **Customized** models.
Orientation [deg]	Set orientation for the imported **Customized** models.

Item Source tab

Note

If the user changes the source type of an item, the user need to redo the selection in the related recipe setting accordingly.

	Description
Vision	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 *Adding vision model on page 283*. For more information regarding Vision Models see the following section.
Predefined	If the source type is set to **Predefined**, the positions generated by the position source are statically defined and no camera is used.
External	If the source type is set to **External**, an external sensor in the solution together with external position generators are used to define item positions. For more information, see *External sensor on page 343*.

Continues on next page

Vision

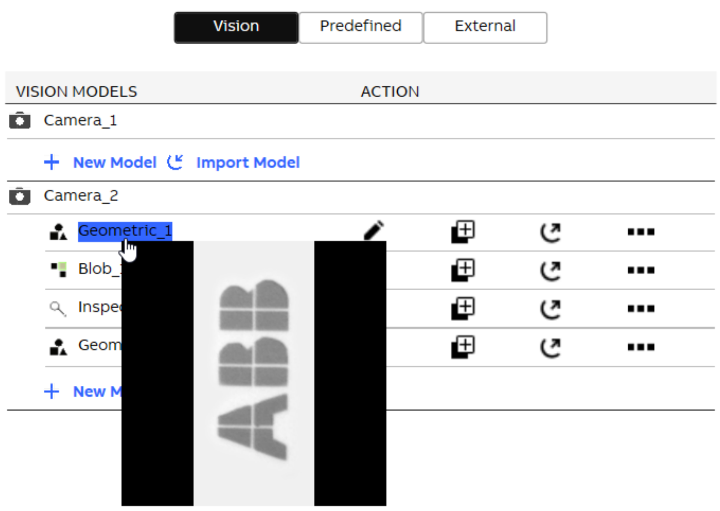
<table>
<tr><th></th><th>Description</th></tr>
<tr><td>New Model</td><td>
Add a new vision model.

Geometric: Add a geometric vision model.
A geometric sub inspection model is configured in the same way as a PatMax model. See Configuring a geometric model with PatMax on page 286. In addition, the relative positions of the found items and the corresponding alignment hit must be trained.
Blob: Add a blob vision model.
A blob sub inspection model is configured in the same way as a blob model. See Configuring blob models on page 294. In addition, the number of required hits must be configured.
Inspection: Add an inspection vision model.

When hovering over the vision model name for one second, the trained model will be displayed as a preview image.

Note
Only geometric model or inspection model with geometric alignment model can be previewed.

Note
All the vision models created with PickMaster Powerpac 2.3.1 or lower version cannot be previewed directly.
Open the edit tab and click OK button to generate the preview image when processing the vision models created with PickMaster Powerpac 2.3.1 or lower version.



xx2400000635
</td></tr>
<tr><td>Import Model</td><td>Import an existed vision model.</td></tr>
<tr><td>Edit</td><td>Edit the selected vision model.</td></tr>
</table>

Continues on next page

	Description
Copy	• **Copy**: Copy the selected vision model to a same type model. • **Copy as an inspection model**: Copy the selected vision model and save as an inspection model with the selected vision model as the alignment model. **Note** For a geometric model, only geometric model with item height setting can be copied as inspection model. Vision height and external height can only be used in geometric model. They are not available for inspection model with geometric alignment model. Then a geometric model with vision height or external height setting cannot be copied to an inspection model. For more information about item height, vision height and external height, see *Configuring height settings on page 358*.
Export	Export the selected vision model.
More	• **Delete**: Delete the selected vision model. • **Rename**: Rename the selected vision model.

Predefined

	Description
Position(X,Y,Z)[mm]	Set the position for the predefined model.
Angle Z[deg]	Set the angle on Z axis of the predefined model.

External

Tip

The **External** configuration for items/containers can only be implemented when real Runtime is connected.

	Description
New position generator	Add an external position generator. When users have not created the position generator for this sensor before, they have to click the **new position generator** button first. Then the python interface of `def configurePosGen(self, posGenId)` will be automatically called, which is the same as the next operation “Configure”. The prerequisite of this operation is that the corresponding external sensor has already been configured according to section 3.2, otherwise there will be a message box showing “The current sensor is not configured. Please configure the sensor before creating the position generator.” For more information on configuring an external position generator, see *External sensor on page 343*.
SYNC TIME[MS]	The time of RT received strobe signal is calculated by the current system time (StrobeTime) minus the time of data process(iTimeSinceStrobe). But the time of from controller trigger strobe signal to RT received strobe signal cannot be calculated. So the value of Synchronization time is used to compensate for this value. This value will be set by users to compensate the time spent for signal transmission on hardware and invoking function. For different external sensor, this value may be set differently.

Continues on next page

	Description
Configure	Once the position generator is created and configured, users could click the button of **Configure** to do configuration again. This operation refers to the Python interface of `def configurePosGen(self, posGenId)`. Users should self-define the position generator configuration behavior in this interface in their own Python class. Although users could only create one position generator in PMPP UI, users could implement more position generation methods in this interface, so that positions could be generated based on one or more methods. The same as sensor configuration, the position generator configuration information should be serialized into a string, so that PMPP solution could get and save this string. This button could be clicked as long as its button state is enabled. If the current row is in disabled state, the corresponding position generator could not be configured until it enters configuration – enabled state.
Delete	Delete the selected position generator.
Save	In the save – enabled state, users could click “Save” button to get the configuration string from the Python program and update in PMPP. This button refers to the Python interface “def savePosGen(self, posGenId)” which is provided by PMTW developer in ExternalSensorInterface.py file and users should not modify the interface content. The content only contains returning the configuration string, so users should make sure that all configured information are included in this string in the “configurePosGen” interface. After “Save” button is clicked, all rows will enter configuration - enabled state.
OK	The “OK” button is for the item/container view. When this button is clicked, all data will be saved, and the item/container view will be closed. If one external sensor position generator is in save – enabled state, the “savePosGen” Python interface will firstly be called before the view is closed.
`Cancel`	The “Cancel” button is for the item/container view. When this button is clicked, all modified data will be abandoned, and the item/container view will be closed.

Procedure

On the PickMaster PowerPac ribbon-tab, click **Process**.

Use this procedure to add an item:

1. On the ribbon-tab, click **Item**.

   The **Item** window opens.
2. In the **RH Size** part, define the item’s size.

   The height of the item defines the pick height and is always added to items found by a vision model or a position defined by a predefined position source.
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 301*.
4. Click **OK**.

Continues on next page

Related information

Configuring inspection models on page 301.

4.3.3.2 Container

Overview

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.

This section describes how to add a container.

Prerequisites

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

Adding a container

Click **Container** on the ribbon to add a container in the solution.

The following table provides details about the **Container** setting dialog box.

Container Properties tab

Container Properties

	Description
Container Name	Change the name.
LWH Size (mm)	Configure the size of the container.
Type	Define the type of the container. • **Box** • **Customized**: import predefined models.

Appearance Properties

	Description
Template	**Default Settings** tab: choose one of the preset templates. **Default Name** text box: enter the name for a new template. **Save** icon: save your new template. **Delete** icon: delete your templates. **Tip** If you enter a new template name in the template text box, a new template will be created instead of being renamed. **Note** If you directly modify the appearance of the default template instead of creating a new template, this will modify the default value of the default template. And all containers created with default template will be modified too.
Color	Change the color of the container.
Use Texture	Use a texture image file for the container.
Label Location	Set the location of the label on the container.
Label Picture	Select an image file for the label picture.
Show Contour	Choose to show the contour or not.

Continues on next page

	Description
Show Orientation Marker	Choose to show the orientation maker or not.
Browse	Select and import a **Customized** model.
Offset [mm]	Set the offset value for the imported **Customized** models.
Orientation [deg]	Set orientation for the imported **Customized** models.

Container Pattern tab

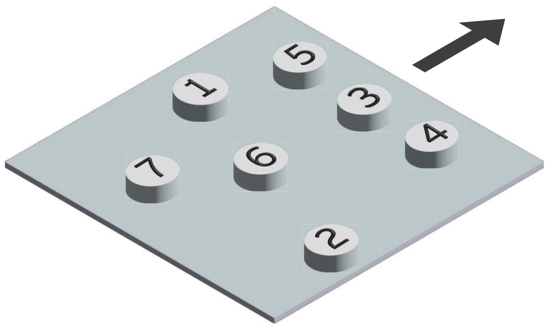
A pattern defines a collection of positions. For example, a box with predefined locations for certain objects. You can change the order, delete, or rearrange the selected layers using the available options. You can adjust the vertical position of each layer by modifying the **Offset** (mm). You can also manage the sorting method. The **Sorting Method** section defines the order in which the items in the container pattern shall be handled by the robots.

	Description
Add Layer	Add a new layer. For more information regarding Add Layer see the following section.
Edit Layer	Edit the selected layer.
Copy	Copy the selected layer.
Delete Layer	Delete the selected layer.
Up	Move the selected layer to a upper level.
Down	Move the selected layer to a lower level.
Delete All	Delete all the existing layers.
Total Weight	Shows the total weight of all the items.
Total Height	Shows the total height of all the items.
Total Count	Shows the total count of all the items.

Add Layer

	Description
Available Items	Select one available item that has been created. **Add** icon: add the selected item onto the layer. **Delete** icon: delete the selected items. **Select All** icon: select all the items in the layer.
Align Style	Define the align style when you have more than one item in the layer. **Left Align** icon: align all the items in this layer from the left. **Center Align** icon: align all the items in this layer from the center. **Right Align** icon: align all the items in this layer from the right. **Top Align** icon: align all the items in this layer at from top. **Middle Align** icon: align all the items in this layer from the middle. **Bottom Align** icon: align all the items in this layer from the bottom.
Distribute Style	Define the distribution style when you have more than one item in the layer. **Horizontally** icon: distribute all the items in the horizontal direction. **Vertically** icon: distribute all the items in the vertical direction.

Continues on next page

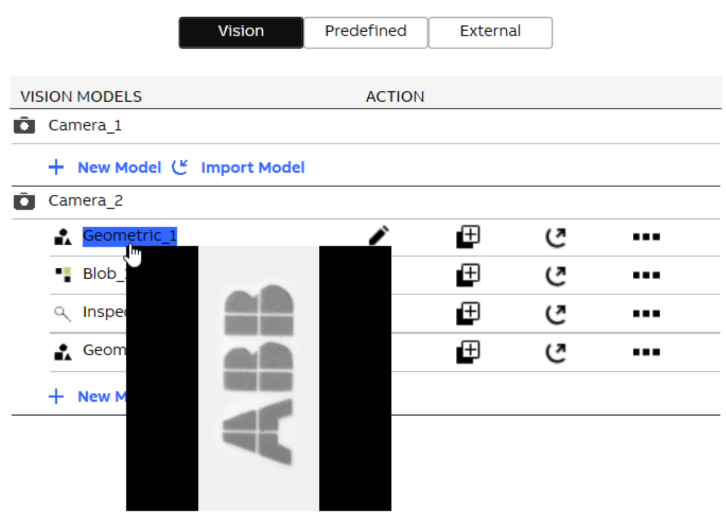
	Description
Else Functions	**Rotate** icon: rotate the selected items.
Sorting Method	Configure the signals. Use the **Customized Settings** options to manage the signals.  xx2400000741 **None** options: The items in the layer shall be accessed in the same order as they are defined in the layout for each layer, but if the next item cannot be reached the next one after that is used. The sorting order for the items in the illustration will be 1 -> 2 -> 3 -> 4 -> 5 -> 6 ->7. **Movement Direction** options: The items shall be accessed along the moving direction for each layer, that is, in the order they travel along a conveyor. The sorting order for the items in the illustration will be 3 -> 4 -> 5 -> 1 -> 6 -> 2 ->7. **Strict** options: The items shall be used in the same order as they are defined in the layout for each layer. If a robot cannot access the next item position in a layer, that robot does not use any more item positions in the container pattern. For example when item 5 is not accessible for the robot with this option selected, item 5, 6, 7 will not be picked, then the sorting order for the items in the illustration will be 1 -> 2 -> 3 -> 4.
Order	Define the order of the layer.
Position X Y Z [mm]	Define the position of the item in the layer.
Angle X Y Z [deg]	Define the angle of the item in the layer.
Show Item Name	Shows the name of the items.
Show Item Order	Shows the added order of the items.

Container Source tab

	Description
Vision	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 *Adding vision model on page 283*. For more information regarding Vision Models see the following section.
Predefined	If the source type is set to **Predefined**, the positions generated by the position source are statically defined and no camera is used.
External	If the source type is set to **External**, an external sensor in the solution together with external position generators are used to define container positions. For more information, see *External sensor on page 343*.

Continues on next page

Vision

	Description
New Model	Add a new vision model. • **Geometric**: Add a geometric vision model. A geometric sub inspection model is configured in the same way as a *PatMax* model. See *Configuring a geometric model with PatMax on page 286*. In addition, the relative positions of the found items and the corresponding alignment hit must be trained. • **Blob**: Add a blob vision model. A blob sub inspection model is configured in the same way as a blob model. See *Configuring blob models on page 294*. In addition, the number of required hits must be configured. • **Inspection**: Add an inspection vision model. When hovering over the vision model name for one second, the trained model will be displayed as a preview image. **Note** Only geometric model or inspection model with geometric alignment model can be previewed. **Note** All the vision models created with PickMaster Powerpac 2.3.1 or lower version cannot be previewed directly. Open the edit tab and click **OK** button to generate the preview image when processing the vision models created with PickMaster Powerpac 2.3.1 or lower version.  xx2400000635
Import Model	Import an existed vision model.
Edit	Edit the selected vision model.

Continues on next page

	Description
Copy	• **Copy**: Copy the selected vision model to a same type model. • **Copy as an inspection model**: Copy the selected vision model and save as an inspection model with the selected vision model as the alignment model. **Note** For a geometric model, only geometric model with item height setting can be copied as inspection model. Vision height and external height can only be used in geometric model. They are not available for inspection model with geometric alignment model. Then a geometric model with vision height or external height setting cannot be copied to an inspection model. For more information about item height, vision height and external height, see *Configuring height settings on page 358*.
Export	Export the selected vision model.
More	• **Delete**: Delete the selected vision model. • **Rename**: Rename the selected vision model.

Predefined

	Description
Position(X,Y,Z)[mm]	Set the position for the predefined model.
Angle Z[deg]	Set the angle on Z axis of the predefined model.

External

Tip

The **External** configuration for items/containers can only be implemented when real Runtime is connected.

	Description
New position generator	Add an external position generator. When users have not created the position generator for this sensor before, they have to click the **new position generator** button first. Then the python interface of `def configurePosGen(self, posGenId)` will be automatically called, which is the same as the next operation “Configure”. The prerequisite of this operation is that the corresponding external sensor has already been configured according to section 3.2, otherwise there will be a message box showing “The current sensor is not configured. Please configure the sensor before creating the position generator.” For more information on configuring an external position generator, see *External sensor on page 343*.
SYNC TIME[MS]	The time of RT received strobe signal is calculated by the current system time (StrobeTime) minus the time of data process(iTimeSinceStrobe). But the time of from controller trigger strobe signal to RT received strobe signal cannot be calculated. So the value of Synchronization time is used to compensate for this value. This value will be set by users to compensate the time spent for signal transmission on hardware and invoking function. For different external sensor, this value may be set differently.

Continues on next page

	Description
Configure	Once the position generator is created and configured, users could click the button of **Configure** to do configuration again. This operation refers to the Python interface of `def configurePosGen(self, posGenId)`. Users should self-define the position generator configuration behavior in this interface in their own Python class. Although users could only create one position generator in PMPP UI, users could implement more position generation methods in this interface, so that positions could be generated based on one or more methods. The same as sensor configuration, the position generator configuration information should be serialized into a string, so that PMPP solution could get and save this string. This button could be clicked as long as its button state is enabled. If the current row is in disabled state, the corresponding position generator could not be configured until it enters configuration – enabled state.
Delete	Delete the selected position generator.
Save	In the save – enabled state, users could click “Save” button to get the configuration string from the Python program and update in PMPP. This button refers to the Python interface “def savePosGen(self, posGenId)” which is provided by PMTW developer in ExternalSensorInterface.py file and users should not modify the interface content. The content only contains returning the configuration string, so users should make sure that all configured information are included in this string in the “configurePosGen” interface. After “Save” button is clicked, all rows will enter configuration - enabled state.
OK	The “OK” button is for the item/container view. When this button is clicked, all data will be saved, and the item/container view will be closed. If one external sensor position generator is in save – enabled state, the “savePosGen” Python interface will firstly be called before the view is closed.
`Cancel`	The “Cancel” button is for the item/container view. When this button is clicked, all modified data will be abandoned, and the item/container view will be closed.

Procedure

On the PickMaster PowerPac ribbon-tab, click **Process**.

Use this procedure to add a container :

1. On the ribbon-tab, click **Container**.

   The **Container** window opens.
2. Define the container with your requirements in the **Container Properties** tab.
3. Define the container pattern with your requirements in the **Container Pattern** tab.
4. In the **Container Pattern** tab, click **New Layer** to define a layer in the container.
5. If need, adjust the layout of the items on the layer.
   - A Select all items on the layer.
   - B Click 'Ctrl' and select the base item at the same time.
   - C Click **Left** to align all items on the left edge according to the base item.

     Click **Right** to align all items on the right edge according to the base item.

Continues on next page

Click **Center** to align all items on the centre line vertically according to the base item.

Click **Middle** to align all items on the centre line horizontally according to the base item.

Click **Top** to align all items on the top edge according to the base item.

Click **Bottom** to align all items on the bottom edge according to the base item.

D Click **Horizontally** to set all items tangent in horizontal direction.

Click **Vertically** to set all items tangent in vertical direction.

6 Click **Save.**

The layer layout is saved.

7 Click **OK**.

The container is saved and the window is closed.

4.3.3.3 Flow

Overview

A flow is used to define how the items and containers are to be generated in the simulation. A flow can be used to simulate the random and irregular incoming material flow in reality. A flow is attached to a hotspot on a conveyor. When attaching the flow, the hotspot becomes a source from where items and containers appear in the simulation according to the flow configuration. Following are the two types of flows:

- Layout: A Layout flow is a predefined layout that is periodically regenerated at the hotspot. The layout may have some random variation regarding the locations of items or containers and the availability of them. The layout may consists of different items or container patterns.
- Recorded: A recorded flow is a recording of a sensor from a simulation or production. The recording is exported from PickMaster PowerPac as an xml file having information of all the detected items and containers during a time interval. When the file is imported, the items detected are mapped to the configured items and container patterns.

Note

Only when the source type of the work area is set as **Vision**, the flow can used.

Adding a flow

Click **Flow** on the ribbon to add a flow in the solution.

The following table provides details about the **Flow** setting dialog box.

	Description
Layout	Define the layout of the flow.
Recorded	Make a flow according to the recorded position of the items and containers.

Layout

	Description
Name	Rename the flow.
Flow Type	Choose the type the flow as **Liner** or **Circular**.
LW Size [mm]	Edit the size of the layout. **Edit Layout**: edit the selected flow. For more information regarding Edit Layout see the following section.
Stability	If set to 100%, all the items in the layout are generated on every trigger without losses. A lower value defines the probability that an item in the layout is generated. For example, if the stability is set as 50%, each item has a half probability of not being generated.
Position Stability	If set to 100%, the generated items always have correct position. A lower value defines the probability that an item gets the correct position.

Continues on next page

	Description
X pos dev min/max [mm]	Defines the minimum and maximum deviation of the **X** position from the correct value.
Y pos dev min/max [mm]	Defines the minimum and maximum deviation of the **Y** position from the correct value.
Orientation Stability	If set to 100%, the generated items always have correct orientation. A lower value defines the probability that an item gets correct orientation.
Z pos dev min/max [deg]	Defines the minimum and maximum deviation of the **Z** position from the correct value.
Rejection Ratio	Defines the probability that an item becomes rejected by a camera. If set to 0%, the item setting “Rejected” in the Layout will decide if the item is rejected.

Edit Layout

> **Note**
>
> For a circular flow, the blue sector view is the area for generating the items and containers in the hotspot frame.
>
> For intuitive visual effect, set the value of the x and y for the hotspots for this conveyor as 0 and 0. This will set the origin of the hotspots at the same position with the center of the circular conveyor. Then the blue sector view is also the quarter view on the circular conveyor.

	Description
Available Objs	Select one item or container you have created for this system. **Add** icon:o add the selected item or container onto the layer. **Delete** icon: delete the selected items. **Select All** icon: select all the items in the layer.
Align Style	Define the align style when you have more than one item in the layer. **Left Align** icon: align all the items in this layer from the left. **Center Align** icon: align all the items in this layer from the center. **Right Align** icon: align all the items in this layer from the right. **Top Align** icon: align all the items in this layer at from top. **Middle Align** icon: align all the items in this layer from the middle. **Bottom Align** icon: align all the items in this layer from the bottom.
Distribute Style	Define the distribution style when you have more than one item in the layer. **Horizontally** icon: distribute all the items in the horizontal direction. **Vertically** icon: distribute all the items in the vertical direction.
Else Functions	**Rotate** icon: rotate the selected items.
Order	Define the order of the layer.
Position [mm]	Define the position of the item in the layer.
Angle [deg]	Define the angle of the item in the layer.
Show Item Name	Shows the name of the items.
Show Item Order	Shows the added order of the items.

Continues on next page

Procedure

On the PickMaster PowerPac ribbon-tab, click **Process.**

Use this procedure to add a rectangle flow:

1. On the ribbon-tab, click **Flow.**

   The **Flow** window is opened.
2. Select a type for the flow in **Flow Type**.
3. Click the **Edit Layout** icon to open the dialog.
4. Click the **Add** icon in the **Edit Layout** dialog to add an item.
   - A Select all items on the layer.
   - B Click 'Ctrl' and select the base item at the same time.
   - C Click **Left** to align all items on the left edge according to the base item.

     Click **Right** to align all items on the right edge according to the base item.

     Click **Center** to align all items on the centre line vertically according to the base item.

     Click **Middle** to align all items on the centre line horizontally according to the base item.

     Click **Top** to align all items on the top edge according to the base item.

     Click **Bottom** to align all items on the bottom edge according to the base item.
   - D Click **Horizontally** to set all items tangent in horizontal direction.

     Click **Vertically** to set all items tangent in vertical direction.
5. Click **OK** to apply the configuration.
6. Click **OK** to close the **Flow** dialog.

4.3.3.4 Recipe

Overview

This section describes how to add a recipe.

In one solution, several recipes can be created. All elements (Robots, sensor and so on) in this solution can be added to any recipes with no limits.

Adding a recipe

Click **Recipe** on the ribbon to add a recipe in the solution.

The following table provides details about the **Recipe** setting dialog box.

Properties

	Description
Available Devices	Define the available devices, including robots and conveyors. All robots and conveyors in the same solution will be listed in every recipe, but they can have different attribute settings in different recipes. For example, the speed of the same robot can be different in different recipes. For more information regarding Available Device see the following section.
Available Workareas	Define the available work areas. All work areas in the same solution will be listed in every recipe, but they can have different attribute settings in different recipes. For more information regarding Available Work Areas see the following section.

Available Devices

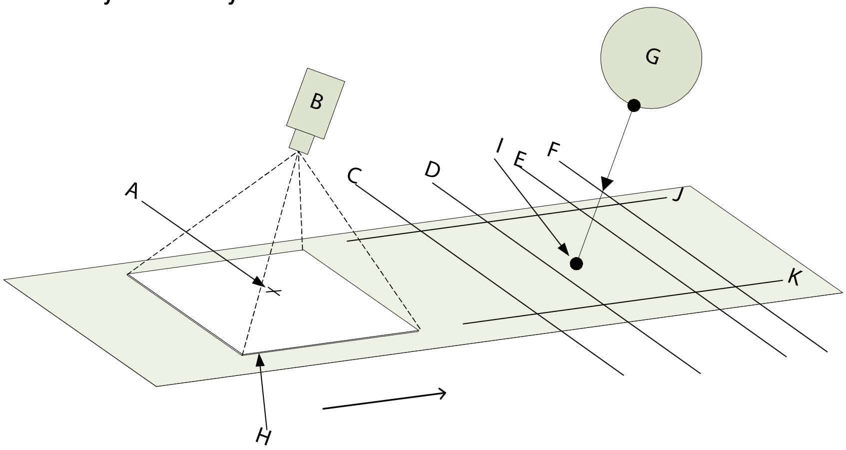
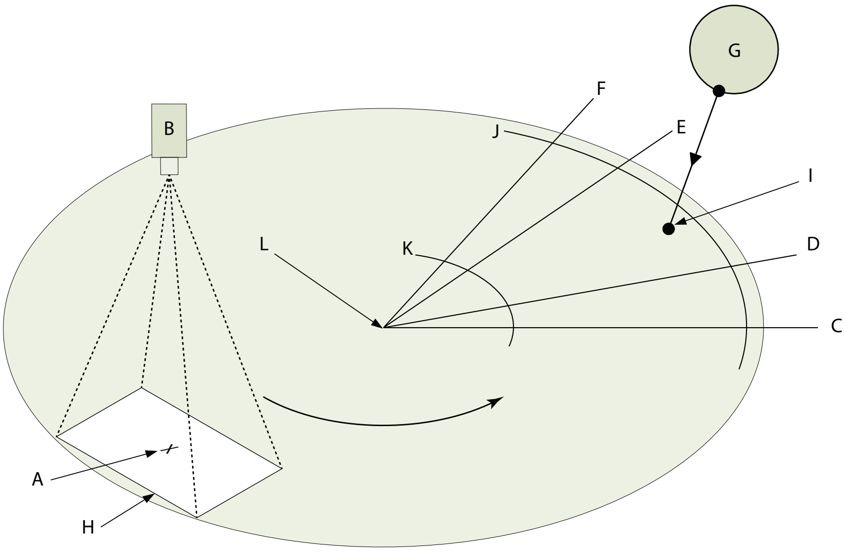
	Description
Robot Setting	**Note** If there are more than one robot in this system, all the robot will be listed here with their defined name. **Rapid Editor**: specify the editor to open Rapid. **Speed**: change the speed of the robot. **Rapid**: import/export/edit the Rapid program of the robot. **Note** The default RAPID module is created for IRB 360. Alternative RAPID template modules for different robot type categories and for double picking can be imported from the installation folder: `C:\Program Files (x86)\ABB\PickMaster Twin 2\PickMaster Twin Client 2\PickMaster PowerPac\RAPID`.
Conveyor Setting	**Speed**: change the speed of the conveyor. **Acceleration**: change the acceleration of the conveyor. **Deceleration**: change the deceleration of the conveyor.

Continues on next page

Available Work Areas

	Description	
Pick Setting	**Pick/place elevation**	The distance, in negative z-direction relative to the tool, from where the robot approaches the item target.
	Pick/place time[s]	The time that 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[s]	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**. **Note** Vacuum activation does not affect the picking of items in simulation. Items are attached to the picking tool using `SimAttach` events, for example, in the Pick Routine.
	Vacuum Reversion[s]	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**. **Note** Vacuum reversion does not affect the placing of items in simulation. Items are detached from the picking tool using `SimDetach` events, for example, in the Place Routine.
	Vacuum Off[s]	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**. **Note** Vacuum Off does not affect the placing of items in simulation. Items are detached from the picking tool using **SimDetach** events, for example, in the Place Routine.
	Load Time[s]	The generation interval time of the objects in the indexed work area. This value is only valid for indexed work areas.

Continues on next page

	Description
Area Setting	After you define a start entry in a work area which may called Start X , you can define a same start entry which may called Start Y at the vertical direction of the Start X.  xx1800001747  xx2400000631

Continues on next page

	Description

A	Camera and Baseframe origin for linear conveyor Camera origin for circular conveyor
B	Camera
C	Enter
D	Start
E	Stop
F	Exit
G	Robot
H	Image frame
I	Center of Robot
J	Y Max/Radius Max
K	Y Min/Radius Min
L	Baseframe origin for circular conveyor

Note

The reference origin for **Enter**, **Exit**, **Start**, and **Stop** is **I** (Center of Robot). The reference base for **Y Max** and **Y Min** is the conveyor base frame.

Enter[mm][i]/**[degree]**[ii]	**Enter** is the limit from where the robot starts to execute item targets on the work area (Start X). 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[mm][i]/**[degree]**[ii]	**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[mm][i]/**[degree]**[ii]	**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[mm][i]/**[degree]**[ii]	**Exit** is the limit from where the robot considers an item target as lost on the work area (Start X). 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 from an arbitrary position in the robot's working area before the position is out of reach.
Y Max[mm][i]/**Radius Max[mm]**[ii]	**Y Max[mm]**/**Radius Max[mm]** is the limit from where robot considers an item target as lost on the work area in End Y.The distance is calculated in millimeter from the center of the robot. The range is positive if the limit is beyond the center of the robot, relative to the moving vertical direction of the conveyor. Make sure that the **Y Max**/**Radius Max** can be reached by the robot. If the y coordinate value of the item's position is greater than the **Y Max**/**Radius Max** , the robot will not grab the item. So 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.

Continues on next page

	Description	
	Y Min[mm] [i]/ **Radius Min[mm]** [ii]	**Y Min[mm]**/**Radius Min[mm]** is the limit from where robot starts to execute item targets on the work area in Start Y. The distance is calculated in millimeter from the center of the robot. The range is positive if the limit is beyond the center of the robot, relative to the moving vertical direction of the conveyor.
	Use Start/Stop	Select the this checkbox if the work area should supervise the start and stop limits. **Note** **Start** and **Stop** values should be within boundaries of **Enter** and **Exit** limits. The value of **Enter** MUST be smaller than the value of **Start**. The value of **Stop** MUST be smaller than the value of **Exit**. Otherwise there will be some errors during simulation. **Note** When **Use Start/Stop** checkbox is selected, the distance between **Stop** and **Exit** should be larger than the size (x direction) of the container . This is handled by the *Conveyor start/stop* signal, see *Work area on page 125*.
	Start with production	Select the this checkbox if the work area should work with the conveyor when the production is started, and stopped when the production is stopped.
	Use Y Max/Y Min [i]/**Use Radius Max/Radius Min** [ii]	Select the this checkbox if the work area should supervise the upper and lower limits.
Record Setting	Record the position of the items and containers in simulation and production. **Note** When **Record scenes** is selected and saved for any work area, the following message will pop up. `Scenes recording is activated for: {0}` After this, the recording will be activated automatically when the simulation or production is started.	

Operation

The operation contains pick operation and place operation.

	Description
Main Setting	Define some basic settings for the operation, such as operation name, flow, source type. For more information regarding Main Setting, see *Main Setting on page 157*.
Filter Setting	Define the filter setting for the operation. For more information regarding Filter Setting, see *Filter Setting on page 159*.
User Script	Select to define the **User Script** function for the operation. For more information regarding User Script, see *User script on page 323*.

Continues on next page

	Description
Distribution Setting	Define the distribution setting for the operation. For more information regarding Distribution Setting, see *Distribution Setting on page 159*.

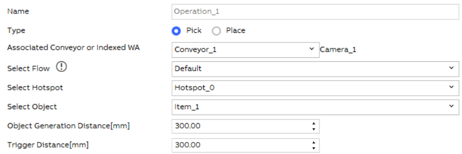
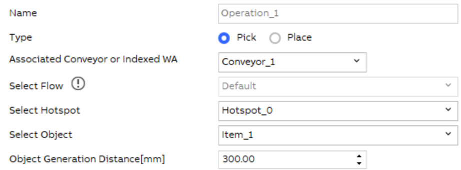
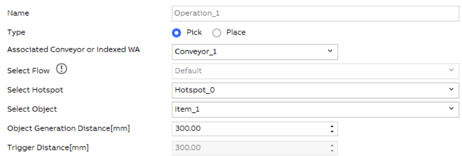
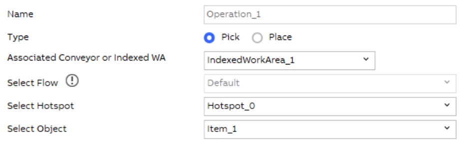
Main Setting

	Description
Operation Name	Rename the operation.
Operation Type	Set the type of the operation.
Associated Conveyor or Indexed WA	Select the associated conveyor or indexed WA.
Select Flow	Select the flow you defined. For more detail on how to add a flow, see *Flow on page 149*. **Tip** If an external sensor is used on the conveyor, Flow function will be disabled.
Select Hotspot	Select the hotspot you defined.
Select Object	Select the available items or containers you defined.
Object Generation Distance[mm]/[degree]	Define the object generated distance value. **Tip** If an indexed work area is used, **Object Generation Distance[mm]/[degree]** is not available. For more information, see the following table.
Trigger Distance[mm]/[degree]	Define the trigger distance value when **Trigger Setting** is set as **Distance**. **Note** When **Source Type** is set as **Predefined** and **Trigger Setting** is set as **Distance**, the trigger distance value comes from the **Object Generation Distance[mm]/[degree]** value. For more information, see the following table.

Continues on next page

Different conditions for using **Object Generation** and **Trigger Distance**

As the **Object Generation Distance[mm]**/**[degree]** and **Trigger Distance[mm]**/**[degree]** are valid in different conditions, we list all conditions with their different options as below:

<table>
<tr><th></th><th>Source Type</th><th>Trigger Setting</th><th>Object Generation Distance[mm]/[degree]</th><th>Trigger Distance[mm]/[degree]</th><th>Main Setting view</th></tr>
<tr><td>Conveyor</td><td>Vision/External Sensor</td><td>Distance</td><td>Available</td><td>Available</td><td>
xx2200002001</td></tr>
<tr><td>Conveyor</td><td>Vision/External Sensor</td><td>I/O</td><td rowspan="2">Available</td><td rowspan="2">Unavailable</td><td rowspan="2">
xx2200002002</td></tr>
<tr><td>Conveyor</td><td>Predefine</td><td>I/O</td></tr>
<tr><td>Conveyor</td><td>Predefine</td><td>Distance</td><td>Available</td><td>Disabled</td><td>
xx2200002003</td></tr>
<tr><td>Indexed work area</td><td>Vision/External Sensor</td><td>Distance</td><td rowspan="4" colspan="2">Unavailable</td><td rowspan="4">
xx2200002004</td></tr>
<tr><td>Indexed work area</td><td>Vision/External Sensor</td><td>Distance</td></tr>
<tr><td>Indexed work area</td><td>Predefine</td><td>I/O</td></tr>
<tr><td>Indexed work area</td><td>Predefine</td><td>I/O</td></tr>
</table>

Continues on next page

Filter Setting

	Description
Position Filter Distance	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 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 *Adding vision model on page 283*. 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.
Overlap Filter Distance	For linear conveyor, 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.
Overlap Filter Angle	For circular conveyor, 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 angle will be regarded as one item. The first identified hit is sent to the robot, and any subsequent hit is filtered out. **Note** For circular conveyor, **Overlap Filter Distance** and **Overlap Filter Angle** are both valid. Which one works depends on which filtering condition is more stringent.

Advanced function - User Script

User script is an advanced function for programming user. For detailed information, see *User script on page 323*.

Distribution Setting

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.

Continues on next page

- 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 production, 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.

Distribution

	Description
Item distribution	Set the distribution strategy as **Accept** or **Reject** for all available items for this operation. **Note** Make sure that at least there is one group valid distribution setting under **Item distribution Accept** or **Reject** for all available items. Otherwise an error will pop up when this recipe is selected to do the simulation or production. `{0} lacked valid distribution. Please check settings in Recipe -> Operation.`
Available Distributor	Shows the available distributor for this operation.

Continues on next page

Load balance

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.

	Description
Load Balance Group	Shows the created load balance group.
Available Distributor	Shows the available distributor for this operation.
New LBGroup	Create a load balance group.
Delete Group	Delete a load balance group.

ATC

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.

	Description
Adaptive Task Completion Group	Shows the created adaptive task completion group.
Available Distributor	Shows the available distributor for this operation.
New ATCGroup	Create a adaptive task completion group.
Delete Group	Delete a adaptive task completion group.

Procedure

On the PickMaster PowerPac ribbon-tab, click **Process.**

Use this procedure to add a recipe:

1. On the ribbon-tab, click **Recipe.**

   The **Recipe** window opens.
2. Click on the **Add Operation** to add a new operation.
3. Click on the **Operation 1** to open the setting window for the operation.
4. Select the operation type as **Pick** or **Place**.
5. If need, click to select the applicable flow in **Select Flow.**
6. Click to select the item in **Available Objects.**

Continues on next page

7. Click to select the work area in **Available Work Areas**.
8. In the **Trigger**/**Filter Setting** tab, define the trigger or filter setting according to your requirements.
9. If need, click to select and configure the **User Script** according to your requirements.
10. In the **Distribution** tab, drag distributors from the **Available distributors** list to the **Distribution** list.

    There can be only one distributor for each item type. If an item type is missing a distributor, it will be regarded as ByPass.
11. 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.
12. 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.
13. Click **OK**.

    The window is closed.

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 PowerPac will redistribute the item position to downstream robots according to the configured distribution strategy for the selected item type.

4.3.4 Operation

4.3.4.1 Simulation

Overview

This section describes how to do the simulation with the created solution.

Control

All operations in the simulation production are reflected in the station view, and all data comes from the solution.

Select one recipe from the tree view and click **Control** on the ribbon to open the control dialog box in the solution.

The following table provides details about the **Control** dialog box.

	Description
Recipes	Control the status of the current recipe and have an overview of the production data. For more information regarding **Recipe** see the following section.
Tuning	Adjust the parameters of the item, work area and robot. For more information regarding **Tuning** see the following section.
Flow Control	Adjust the speed of the conveyor. For more information regarding **Flow Control** see the following section.

Recipe

	Description
Recipe Status	Control the status of the production.
Picking Status	Shows the overview of the picking status in summary or detail.

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.

Tip

For item tuning, the tuning value only affects the new generated item targets. The tuning value will not be effective on the recognized item targets in the queue.

For the work area and robot tuning, the tuning value will be effective immediately.

Tuning the item

	Description
OffsetX	Set the location of the gripper when doing the picking and placing operation in X direction.

Continues on next page

	Description
OffsetY	Set the location of the gripper when doing the picking and placing operation in Y direction.
OffsetZ	Set the location of the gripper when doing the picking and placing operation in Z direction.
RotateRX	Set the angle of the gripper when doing the picking and placing operation in X direction. **Note** The angle cannot be out of the physical limits. Otherwise the robot will not work normally. For example, trying to rotate the gripper of an IRB 360 robot in X or Y direction will cause an error. Redo the simulation after the error occurred.
RotateRY	Set the angle of the gripper when doing the picking and placing operation in Y direction. **Note** The angle cannot be out of the physical limits. Otherwise the robot will not work normally. For example, trying to rotate the gripper of an IRB 360 robot in X or Y direction will cause an error. Redo the simulation after the error occurred.
RotateRZ	Set the angle of the gripper when doing the picking and placing operation in Z direction. **Note** The angle cannot be out of the physical limits. Otherwise the robot will not work normally.

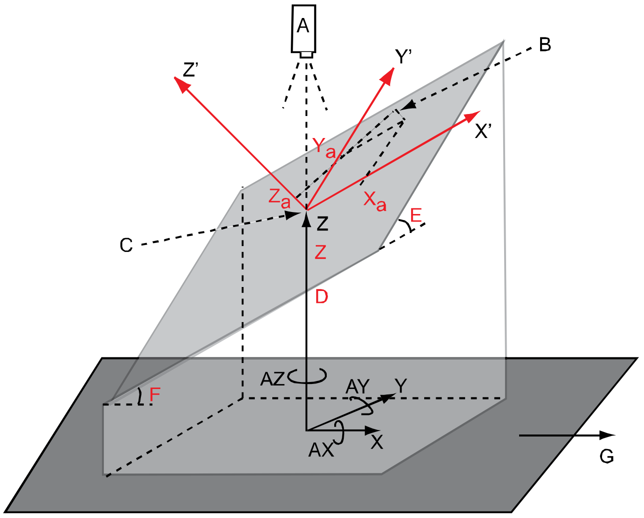
Configuring the grip location

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

1 Select the **Type** as **Item** and select the required item.

Continues on next page

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 can only rotate around the z-axis and therefore only **RotateRZ** can be used.

Six axes robots can pick/place 3D items by defining Euler orientation **RotateRX**, **RotateRY** 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 (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.

Tuning the work area

Note

The parameters of in tuning work area are synchronized with the parameters in the recipe. Any modification in one place will modify the parameters in the other place.

Continues on next page

	Description
OffsetX[mm]	Tune the position of the work area along the X direction when running simulation or production. Tuning the position of the work area along the X direction is equivalent to offsetting the conveyor base frame along the X direction.
OffsetY[mm]	Tune the position of the work area along the Y direction when running simulation or production. Tuning the position of the work area along the Y direction is equivalent to offsetting the conveyor base frame along the Y direction.
OffsetZ[mm]	Tune the position of the work area along the Z direction when running simulation or production. Tuning the position of the work area along the Z direction is equivalent to offsetting the conveyor base frame along the Z direction.
Enter[mm][i]**/[degree]**[ii]	**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. For more details, see *Available Work Areas on page 153*.
Exit[mm][i]**/[degree]**[ii]	**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 from an arbitrary position in the robot's working area before the position is out of reach. For more details, see *Available Work Areas on page 153*.
Elevation[mm]	**Elevation** is the distance, in negative z-direction relative to the tool, from where the robot approaches the item target.
Follow Time[s]/Dwell Time[s]	**Follow Time**/**Dwell 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[s]	**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**. **Note** Vacuum activation does not affect the picking of items in simulation. Items are attached to the picking tool using `SimAttach` events, for example, in the Pick Routine.
Vacuum Reversion[s]	**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**. **Note** Vacuum reversion does not affect the placing of items in simulation. Items are detached from the picking tool using `SimDetach` events, for example, in the Place Routine.
Vacuum Off[s]	**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**. **Note** Vacuum Off does not affect the placing of items in simulation. Items are detached from the picking tool using **SimDetach** events, for example, in the Place Routine.

Continues on next page

	Description
Y Max[i]/**Radius Max**[ii]	**Note** To enable this function, you need to select the **Use Start/Stop** checkbox for this function in the recipe configuration page. **Note** The **Y Max/Radius Max** function in the **Tuning** window has a slight delay. If there is any update for this value, you need to wait a while to see the results. **Y Max/Radius Max** is the limit from where robot considers an item target as lost on the work area in End Y.The distance is calculated in millimeter from the center of the robot. The range is positive if the limit is beyond the center of the robot, relative to the moving vertical direction of the conveyor. Make sure that the **Y Max/Radius Max** can be reached by the robot. If the y coordinate value of the item's position is greater than the**Y Max/Radius Max**, the robot will not grab the item. So 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. For more details, see *Available Work Areas on page 153*.
Y Min[i]/**Radius Min**[ii]	**Note** To enable this function, you need to select the **Use Start/Stop** checkbox for this function in the recipe configuration page. **Note** The **Y Min/Radius Min** function in the **Tuning** window has a slight delay. If there is any update for this value, you need to wait a while to see the results. **Y Min/Radius Min** is the limit from where robot starts to execute item targets on the work area in Start Y. The distance is calculated in millimeter from the center of the robot. The range is positive if the limit is beyond the center of the robot, relative to the moving vertical direction of the conveyor. For more details, see *Available Work Areas on page 153*.
Display Base-frame	Select the this checkbox if you want to show the conveyor base frame in the station view.

i Only available when the conveyor is linear conveyor.
ii Only available when the conveyor is circular conveyor.

Tuning the robot

The robot settings can be tuned when a production is running, using the **Tuning the robot** window.

Limitations

All tunings, including robot tuning, item tuning, and work area tuning, are only valid while the simulation or production is running.

Flow Control

	Description
Conveyor	**Speed[mm/s or rad/s]**Adjust the speed of the conveyor.
Indexed Work Area	**Load Time[s]**: The generation time interval of the objects in the indexed work area. This value is only valid for indexed work areas.

Continues on next page

Simulation

> **Note**
>
> It is recommended to calibrate the solution when its virtual controller is used in other solution before simulation.
>
> If different solutions use the same virtual controller, any modification to the controller of one solution will affect other solutions. This will cause unexpected and misleading behavior of other solutions.

Use this procedure to do the simulation:

1. On the PickMaster PowerPac ribbon-tab, click **Operation**.
2. Click to choose one recipe from the tree view browser.
3. Click **Start** on the ribbon-tab. Then it will start the simulation of created solution.

   The simulation runs automatically.
4. Click **Stop** on the ribbon-tab. Then it will stop the simulation.

4.4 Configuration in real Runtime (RRT)

4.4.1 Switching to real Runtime

Configuring local IP address in PickMaster Runtime

Note

The network interface configurated in Runtime must be the IP address of the local computer connected to the controller using WAN interface.

The local IP address should be configured in the PickMaster Runtime (RRT) in the following cases:

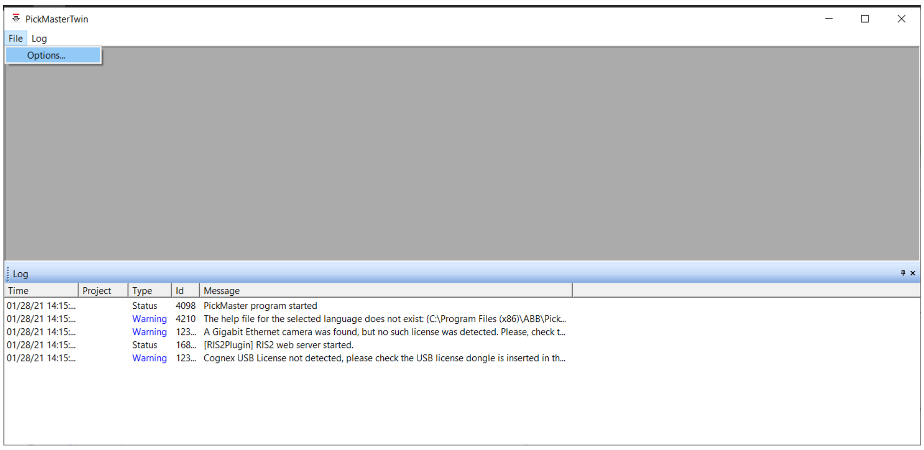
- The IP for the PickMaster Runtime time synchronization service is not configured before.
- The network interface currently used for connecting the real controller has been changed.

Note

The network interface configurated in Runtime must be the IP address of the local computer connected to the controller using WAN interface.

Use the following procedure to configure the local IP address in the PickMaster Runtime (RRT):

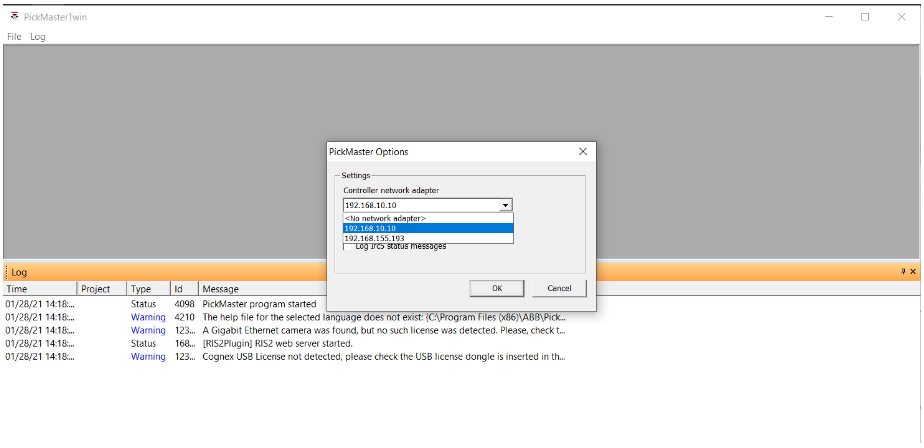
1. Start Runtime.
2. Click **File - Options** to open a pop-up dialog.



xx2100000346

Continues on next page

3 Select the corresponding IP address in the list box and click **OK**.



xx2100000347

Switch Runtime

Note

After install PickMaster Twin Client and PickMaster Twin Host on different PC as recommended, there will be two real Runtime available but only the one connected to controller or camera should be used.

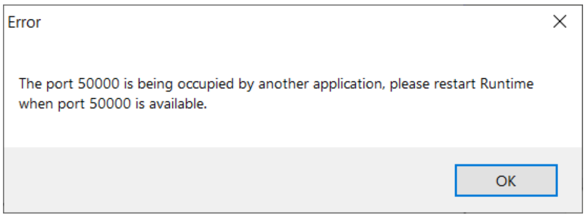
The real Runtime on Host PC and Client PC are identical but the one on Host is for production. Robot controllers and cameras should also be connected to this one.

Continues on next page

Tip

The PickMaster® Runtime (VRT and RRT) is defined to use 50000 port. If 50000 port is occupied by other program, you will have this warning and not be able to connect to Runtime,

Release the 50000 port and restart the PickMaster® Runtime.



xx2100000868

Use this procedure to release the 50000 port:

1. Enter the command `netstat -aon|findstr "50000"` in the CMD window.
2. The process that occupies port 50000 will be listed in the window. Obtain the PID code of the process.
3. Find the process corresponding to this PID in the task manager and close the it (Make sure that this process is allowed to be closed on this computer).
4. Restart PickMaster® Runtime and connect.

Right-click on **Runtime** to set the connection to the virtual Runtime (VRT) in simulation mode or the real Runtime (RRT) for operating the real robots on the Host computer in emulation mode.

Tip

Before connecting to RRT, start the PickMaster Runtime on the Host computer.

When selecting **Connect to RRT**, the **Sign in** window is displayed.

The following table provides details about the **Connect to RRT** dialog box.

	Description
IP Address	Locate the IP address of the Runtime computer. **Tip** Check the IPv4 address of the computer which the PickMaster Runtime is installed on. **Note** Loopback address is NOT allowed to use as the real PickMaster Runtime IP address, for example 127.0.0.1. Loopback address will cause errors in vision function.
Credential	
UserName	The default user name is admin. And it CANNOT be changed.

Continues on next page

	Description
Password	Enter the password of your account in the Runtime.

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

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

Note

If the solution will be used in the PickMaster Operator, it must have been connected to a real controller with the same configuration on PickMaster PowerPac.

Procedure

To connect to Runtime.

1. Right-click the **Runtime** in the tree view **Layout** and select **Start Local RRT**.
2. Right-click the **Runtime** in the tree view **Layout** and select **Connect to RRT**.

   The **ConnectToRRT** window is opened.
3. In the **Sign in** dialog, enter the correct information.
4. Click **OK**.

   Tip

   If switch failed, the message box will show up.

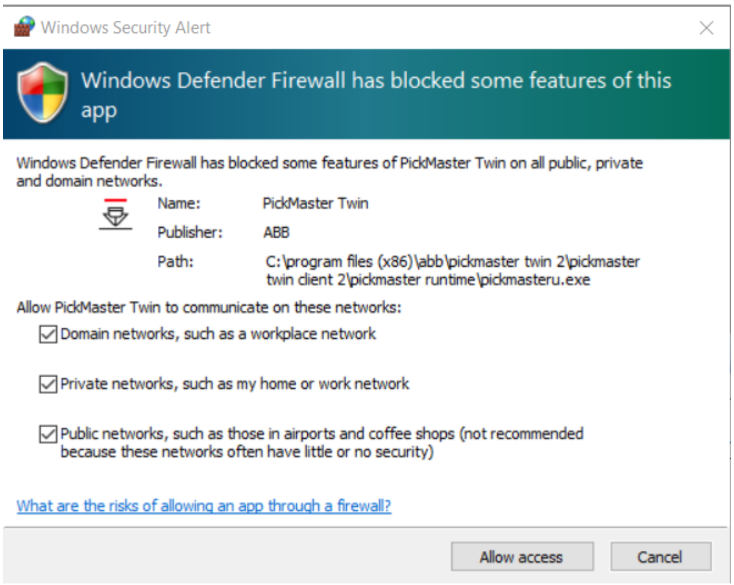
   Message: `Failed to connect to RRT, please make sure the RRT starts or login information is correct.`

Continues on next page

Note

If the user meets any problem when building connection between PickMaster PowerPac and real Runtime, please check from below possible reasons:

1. Using a host account that is not administrator;
2. Firewall blocking;
3. VPN interference;
4. Host IP address incorrect, or not in the same IP segment as the client port.
5. The PickMaster PowerPac is not allowed to communicate in all networks.



xx2100001954

Select a real controller

Note

Make sure that at least one real controller has been selected for the controller which need to run the production.

Otherwise an error will pop up when this recipe is selected to do the production.

```
{0} lacked real controller setting. Please connect to a real controller first.
```

Use this procedure to select a real controller:

1. Right-click the **Controller** in the tree view **Layout** and select **Edit Controller**.

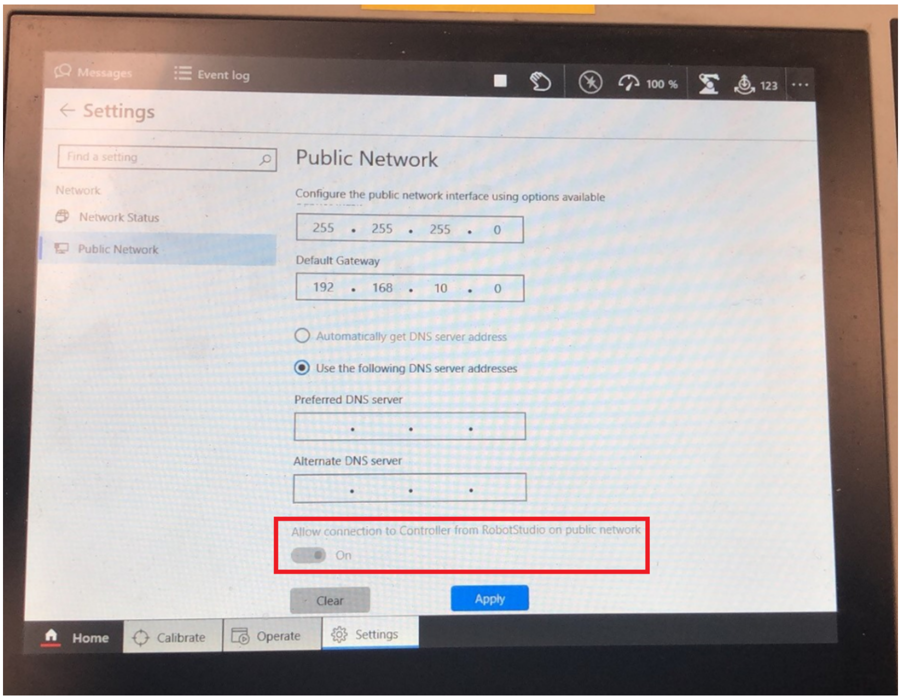
   The **Edit Controller** dialog is opened.

Continues on next page

2 Click on the **Select Real Controller** icon to open the **Select Real Controller** dialog.

> **Note**
>
> User must modify the firewall settings before selecting a real OmniCore controller in PickMaster PowerPac.
>
> For WAN port, under Configuration/Communication/Firewall Manager, the following functions must be enabled.
>
> "RobICI" -EnableOnPublicNet
>
> "IEEE1588" -EnableOnPublicNet
>
> "Netscan" -EnableOnPublicNet
>
> "RobAPI" -EnableOnPublicNet

> **Note**
>
> Make sure that the setting 'Allow connection to controller from RobotStudio on public network' is enabled.



xx2100000506

The **Select Real Controller** dialog is opened.

3 In the dialog box, choose the real controller to be connected.

4 Click **OK** to apply the configuration.

5 Click **Close** to close the **Edit Controller** dialog.

Continues on next page

Modifying I/O signals in work area

Note

Make sure that it is NOT set as **Default** signal type for the work areas which need to run the production.

Otherwise an warning will pop up when this recipe is selected to do the production.

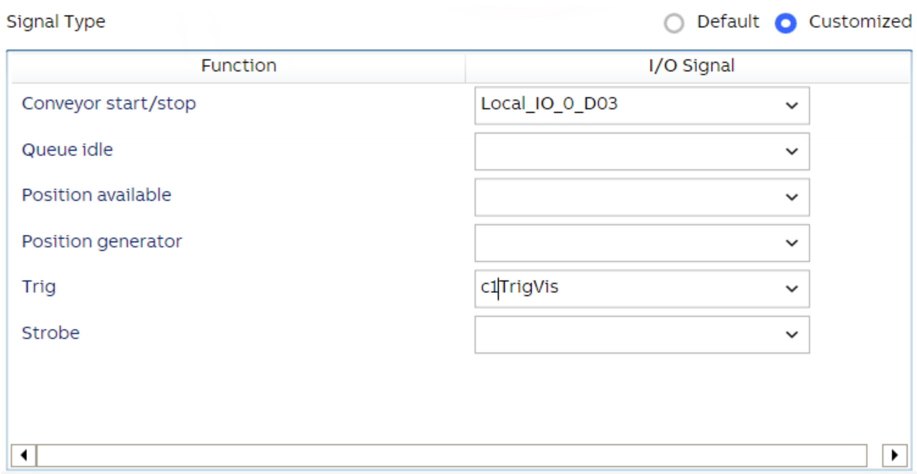
`{0} used default signal type and lacked customized signal type setting. Please check the signal configuration in work area.`

Use this procedure to modify the I/O signals in work area which is in used:

1. Right-click on **Conveyor WorkArea 1** in the tree view **Layout** and select **Setting**.

   The **Conveyor work area setting** window is opened.
2. Select the **Customized Settings** in the **Signal Type** tab.
3. Enter the required data into the **I/O signal** setting table. For more information, see *Configuring the I/O on page 175*.

   For example:

   xx2100001628
4. Click **OK** to close the **Recipe setting** window.
5. Repeat step1 - 4 to the other **Conveyor WA.**

Configuring the I/O

I/O signals

I/O signals are configured using RobotStudio or the FlexPendant.

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 signal is 15 characters.

Continues on next page

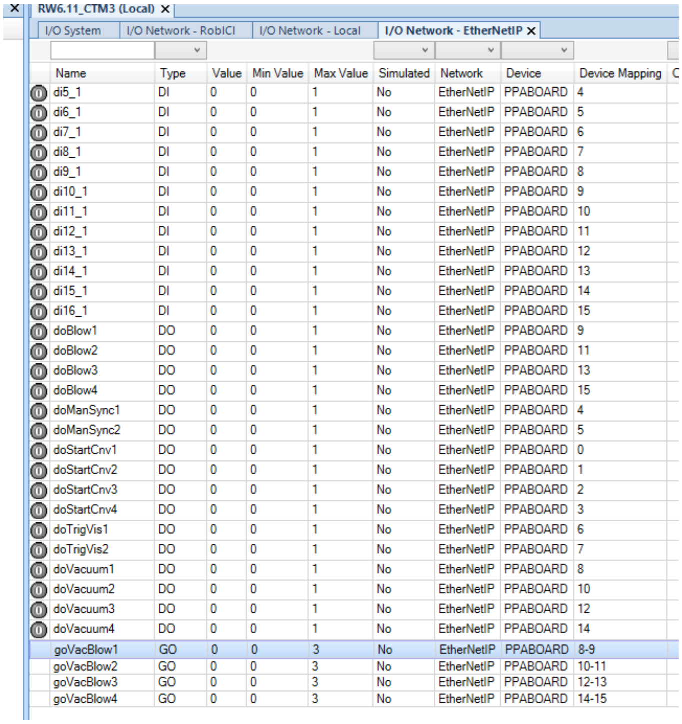
The following I/O signals are used in PickMaster PowerPac. Some of them are used or referenced to when configuring the solution. The encoder signals are described in *Application manual - Conveyor tracking*.

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`[i]	Digital output for starting/stopping conveyors.
`doTrigVisX`/`cXTrigVis`	Digital output for triggering an image acquisition. This signal is used by Runtime to order the camera to acquire an image. For DSQC 377, this output should be connected to the `doTrigVisX` on the corresponding encoder board. For DSQC 2000, this output should be connected to the `cXTrigVis` . For more detail information, see the circuit diagram.
`doManSyncX`	Digital output used for triggering predefined positions in a conveyor work area. For DSQC 377, this output should be connected to the `StartSig` (input 9) on the corresponding encoder board. For DSQC 2000, this output should be connected to the `cXTrigVis` . For more detail information, see the circuit diagram.
`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.

Continues on next page

<table>
<tr><th>I/O signal name</th><th>Description</th></tr>
<tr><td><code>goVacBlowX</code></td><td>Digital I/O group containing doVacuumX and doBlowX.

Note

If this signal group is not defined on the same board, the user needs to define four <code>goVacBlowX</code> signal groups on the corresponding IO board.

<table>
<tr><th>Name</th><th>Type</th><th>Value</th><th>Min Value</th><th>Max Value</th><th>Simulated</th><th>Network</th><th>Device</th><th>Device Mapping</th></tr>
<tr><td>di5_1</td><td>DI</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>4</td></tr>
<tr><td>di6_1</td><td>DI</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>5</td></tr>
<tr><td>di7_1</td><td>DI</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>6</td></tr>
<tr><td>di8_1</td><td>DI</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>7</td></tr>
<tr><td>di9_1</td><td>DI</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>8</td></tr>
<tr><td>di10_1</td><td>DI</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>9</td></tr>
<tr><td>di11_1</td><td>DI</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>10</td></tr>
<tr><td>di12_1</td><td>DI</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>11</td></tr>
<tr><td>di13_1</td><td>DI</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>12</td></tr>
<tr><td>di14_1</td><td>DI</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>13</td></tr>
<tr><td>di15_1</td><td>DI</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>14</td></tr>
<tr><td>di16_1</td><td>DI</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>15</td></tr>
<tr><td>doBlow1</td><td>DO</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>9</td></tr>
<tr><td>doBlow2</td><td>DO</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>11</td></tr>
<tr><td>doBlow3</td><td>DO</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>13</td></tr>
<tr><td>doBlow4</td><td>DO</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>15</td></tr>
<tr><td>doManSync1</td><td>DO</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>4</td></tr>
<tr><td>doManSync2</td><td>DO</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>5</td></tr>
<tr><td>doStartCnv1</td><td>DO</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>0</td></tr>
<tr><td>doStartCnv2</td><td>DO</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>1</td></tr>
<tr><td>doStartCnv3</td><td>DO</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>2</td></tr>
<tr><td>doStartCnv4</td><td>DO</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>3</td></tr>
<tr><td>doTrigVis1</td><td>DO</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>6</td></tr>
<tr><td>doTrigVis2</td><td>DO</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>7</td></tr>
<tr><td>doVacuum1</td><td>DO</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>8</td></tr>
<tr><td>doVacuum2</td><td>DO</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>10</td></tr>
<tr><td>doVacuum3</td><td>DO</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>12</td></tr>
<tr><td>doVacuum4</td><td>DO</td><td>0</td><td>0</td><td>1</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>14</td></tr>
<tr><td>goVacBlow1</td><td>GO</td><td>0</td><td>0</td><td>3</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>8-9</td></tr>
<tr><td>goVacBlow2</td><td>GO</td><td>0</td><td>0</td><td>3</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>10-11</td></tr>
<tr><td>goVacBlow3</td><td>GO</td><td>0</td><td>0</td><td>3</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>12-13</td></tr>
<tr><td>goVacBlow4</td><td>GO</td><td>0</td><td>0</td><td>3</td><td>No</td><td>EtherNetIP</td><td>PPABOARD</td><td>14-15</td></tr>
</table>

xx2100001673</td></tr>
</table>

i For DSQC 2000, there is no predefined port for this signal. Define the real connected port on the board as the signal name.

Conveyor work area default I/O signals

The default I/O signals are used for simulation.

Item	DSQC 377	DSQC 2000
Conveyor start/stop	`cnvX_doStartCnv`	`cnvX_doStartCnv`
Queue idle	`cnvX_doQIdle`	`cnvX_doQIdle`
Position available	`cnvX_doPAvail`	`cnvX_doPAvail`
Position generator	`cnvX_diPosGen`	`cnvX_diPosGen`
Trig	`doTrigVisX`	`cXTrigVis`
Strobe	`cXNewObjStrobe`	`cXNewObjStrobe`

Continues on next page

Conveyor work area customized I/O signals

The customized I/O signals are used for production.

Item	DSQC 377	DSQC 2000
Conveyor start/stop	`doStartCnvX` **Note** This signal can be left as empty if the conveyor is running.	`doStartCnvX` **Note** This signal can be left as empty if the conveyor is running.
Queue idle		
Position available		
Position generator		
Trig	`doTrigVisX`	`cXTrigVis`
Strobe	`cXNewObjStrobe`	

Indexed work area default I/O signals

The default I/O signals are used for simulation.

Item	DSQC 377	DSQC 2000
Conveyor start/stop		
Queue idle	`indX_doQIdle`	`indX_doQIdle`
Position available	`indX_doPAvail`	`indX_doPAvail`
Position generator	`indX_diPosGen`	`indX_diPosGen`
Trig	`indX_doTrigVis`	`indX_doTrigVis`
Strobe	`indX_diStrobe`	`indX_diStrobe`

Indexed work area customized I/O signals

The customized I/O signals are used for production.

Item	DSQC 377	DSQC 2000
Conveyor start/stop		
Queue idle	`doTrigVisX`[i] **Note** The **Queue idle** signal and **Strobe** signal should be the same one.	`cXTrigVis`[i] **Note** The **Queue idle** signal and **Strobe** signal should be the same one.
Position available		
Position generator		
Trig		
Strobe	`doTrigVisX`	`cXTrigVis`

i Any available `do` signals can be used.

Continues on next page

Note

Make sure that the activator signal setting of gripper is exactly same with the connected controller.

Otherwise the gripper will not pick or place the items in PickMaster PowerPac.

4.4.2 Configuring camera

Introduction

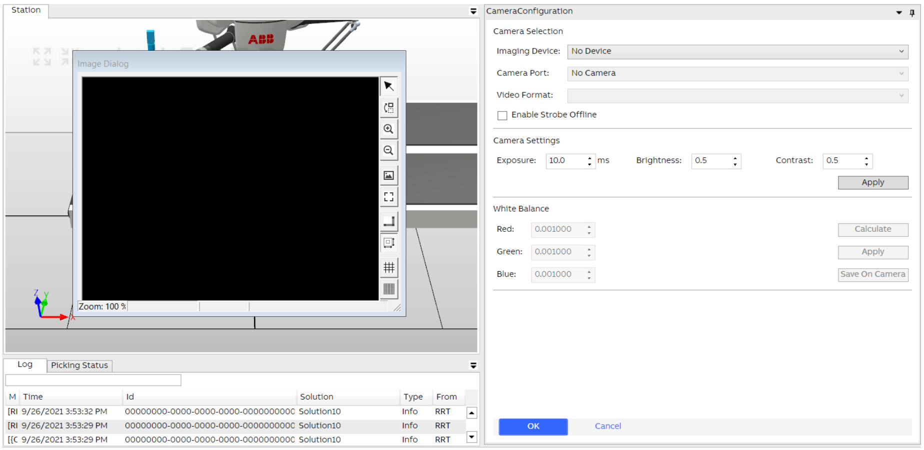
> **Note**
>
> If any firewall or antivirus software is installed, add `pickmasteru.exe` and `visionclient.exe` to the white list.
>
> Otherwise the PickMaster PowerPac cannot connect Runtime and the vision function cannot work normally.

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

To configure a camera.

1 Right-click the camera in the tree view **Layout** and select **Configuration**.

The **Camera Configuration** dialog and the **Image dialog** are opened.



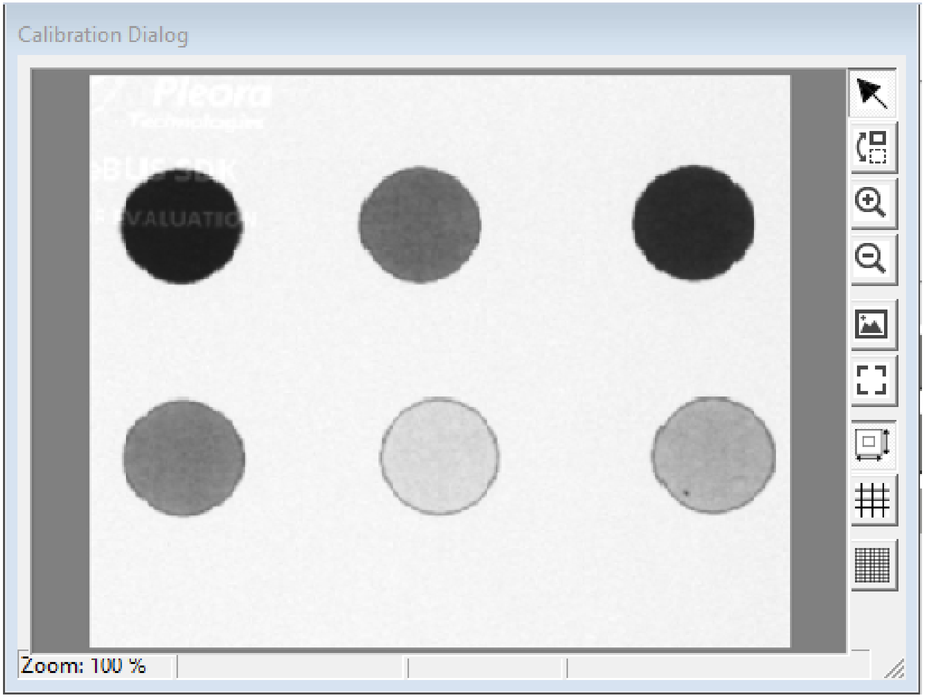
xx2100001622

2 In the **Imaging device** list, select the Gigabit Ethernet camera to which the camera is connected.

Continues on next page

3 In the **Video format** list, select the type of the connected camera.

The image in **Image dialog** shows up.



xx2100001521

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.

   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** part, 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**. This will store the settings in the camera.

For more information about color vision, see *Using color vision on page 310*.

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.

Continues on next page

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.

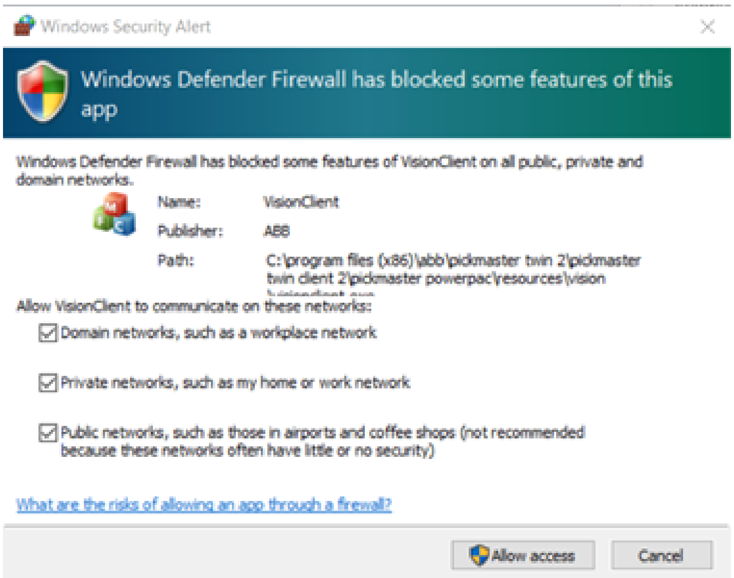
The effect of changing these parameter values is not seen until clicking **Apply**.

7. Click **OK**.

Note

If the user meets any problem when opening **VisionClient**, please check from below possible reasons:

1. Firewall blocking;
2. The **VisionClient** is not allowed to communicate in all networks.



xx2200001061

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. Right-click the camera in the tree view **Cell** and select **Configuration**.

   The **Camera Configuration** dialog is opened.

Continues on next page

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 PickMaster CD 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 PickMaster CD 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 310.

Calibrating camera on page 273.

4.4.3 Calibrating robot

Instruction

Detailed information about how to calibrating the robot are described in the robot product manual.

4.4.4 Calibrating linear conveyor

Overview

Note

The following calibration process is required when running production and emulation. Calibration under the simulation tab in PickMaster PowerPac will not complete the following calibration process.

The calibrations needed for the conveyors 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 solution so all recipe in that solution could share the same calibration. If you need to re-calibrate a camera, all recipes in the solution 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 linear conveyor:

1. For the cable connections from encoder to DSQC 2000, see *Cable connections from encoder to DSQC 2000 for linear conveyor on page 186*.
2. Define the parameter *Counts Per Meter* (for conveyors only), see *Defining the parameter Counts Per Meter on page 187*, *Defining the parameter Counts Per Meter on page 210*.
3. Calibrate the camera, see *Defining the base frame on page 189*, *Defining the base frame on page 212*.

Continues on next page

4.4.4.1 Calibrating linear conveyor with DSQC 2000

4.4.4.1.1 Cable connections from encoder to DSQC 2000 for linear conveyor

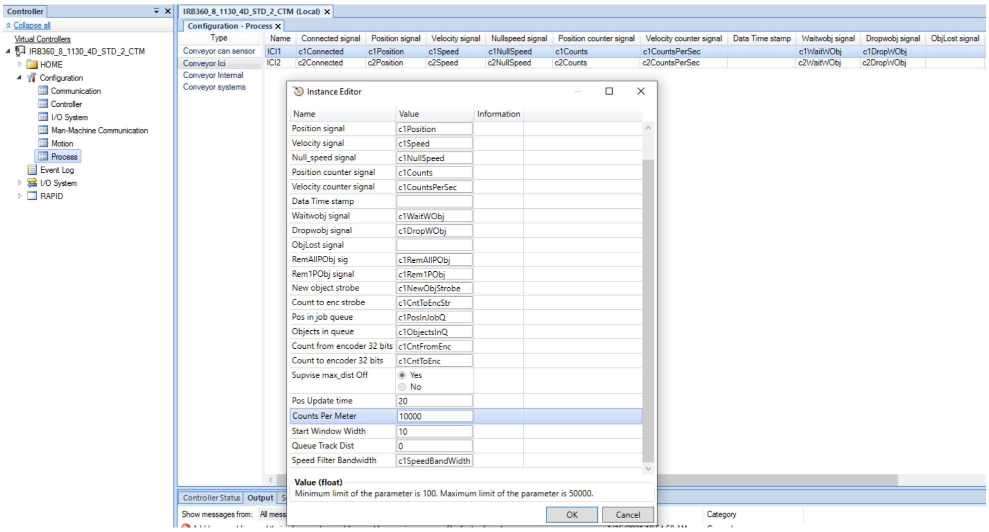
Introduction

For the information on connecting encoder to DSQC 2000 for linear conveyor, see *Application manual - Conveyor tracking*.

4.4.4.1.2 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 *Conveyor Ici*, in the topic *Process*.



xx2100000042

Calculation for Counts Per Meter

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

`counts value/measured_meters`

Value	Description
`counts value`	The conveyor position after moving. For DSQC 2000: Read from predefined I/O signal on the FlexPendant or RobotStudio. For example, CNV1, the signal name is c1counts.
`measured_meters`(m)	The manually measured distance in meters 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`.

Continues on next page

6. Calculate *Counts Per Meter* using the read and measured values.

   For example: `20200/1.005 = 20099`
7. In RobotStudio, click **Configuration** and select topic **Process** and type **Conveyor Ici**.
8. Edit the unit *ICIx* (where x is the number of the conveyor) and update the value for parameter *Counts Per Meter*.
9. Tap **OK**.
10. Restart the controller.

Related information

Technical reference manual - System parameters.

4.4.4.1.3 Defining the base frame

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

- Define the `Counts Per Meter` system parameter for each conveyor work area. For more details, see *Defining the parameter Counts Per Meter on page 210*, *Defining the parameter Counts Per Meter on page 187*.
- 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, see *Calibrating camera on page 273*. After calibrating the camera, keep the camera calibration pattern attached to the conveyor.

Procedure for OmniCore

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

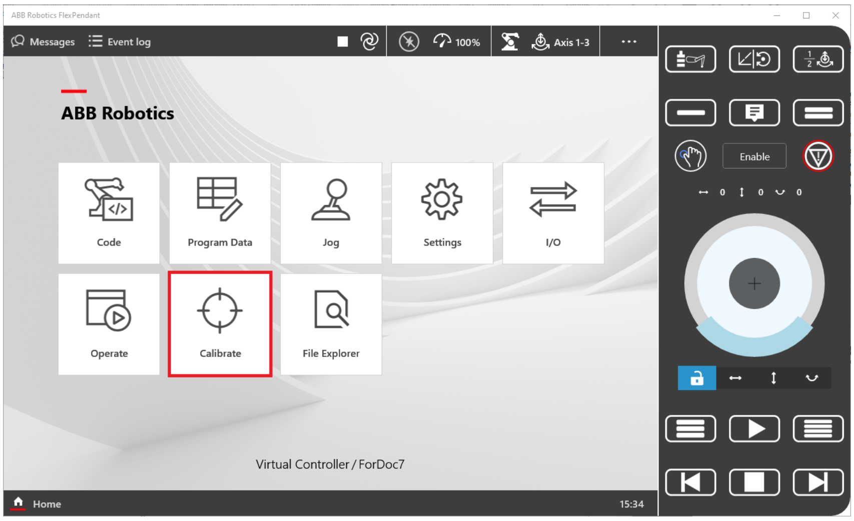
1. Make sure the reference point for calibration is marked accurately on the conveyor belt.
   - If a camera is used, the reference point is the local origin of the camera view. If the camera has been just calibrated, the reference point is already marked by the origin of the camera calibration pattern 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.

Continues on next page

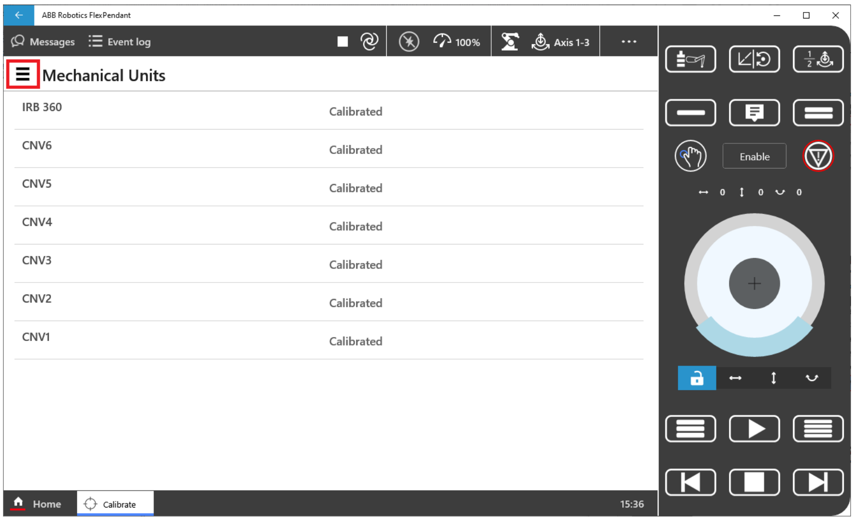
Do the following for all the robots having work areas that needs to be calibrated along the conveyor:

- In the FlexPendant, click **Calibrate**.



xx2100000362

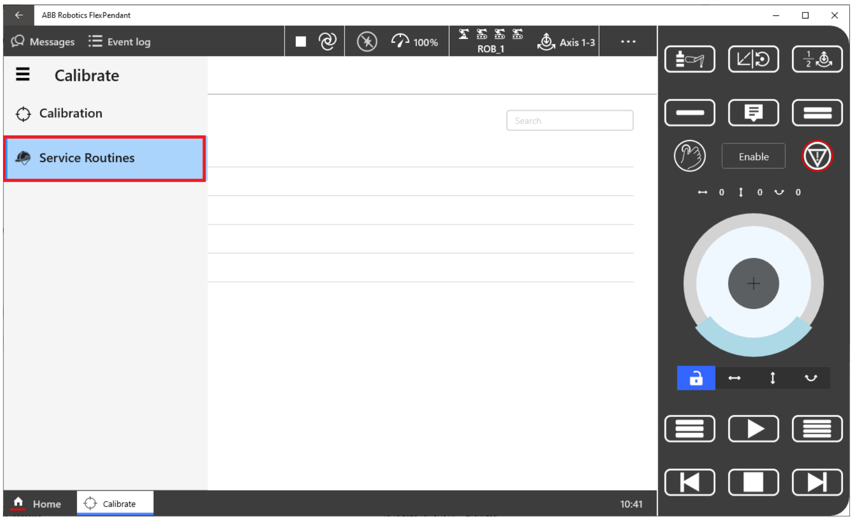
- Click **Option Tab** on the up left corner.



xx2100000363

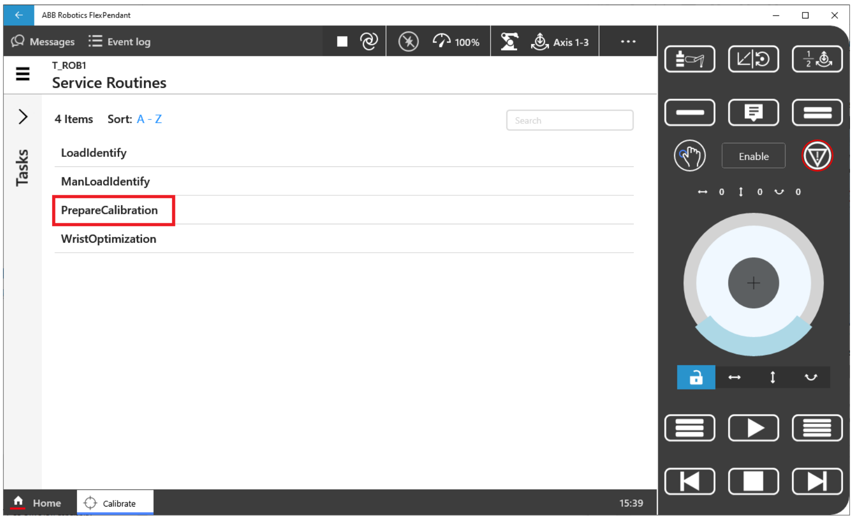
Continues on next page

- Click **Service Routines**.



xx2100000364

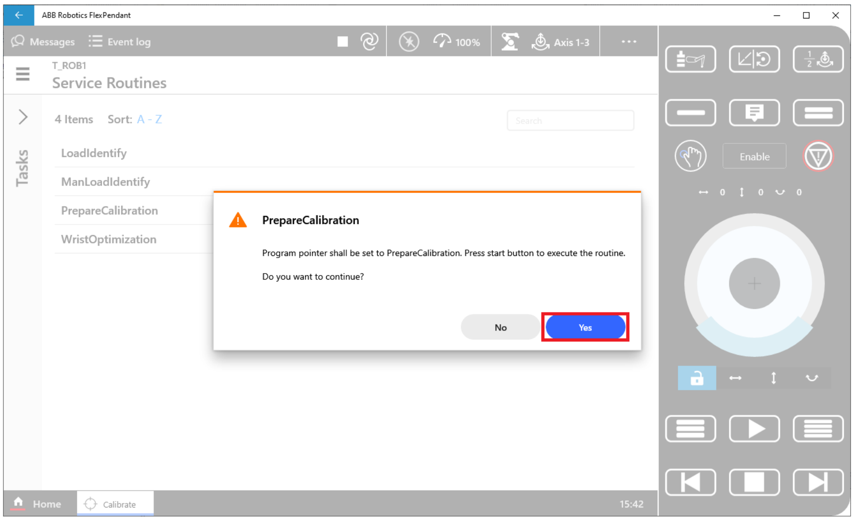
- Click **PrepareCalibration**.



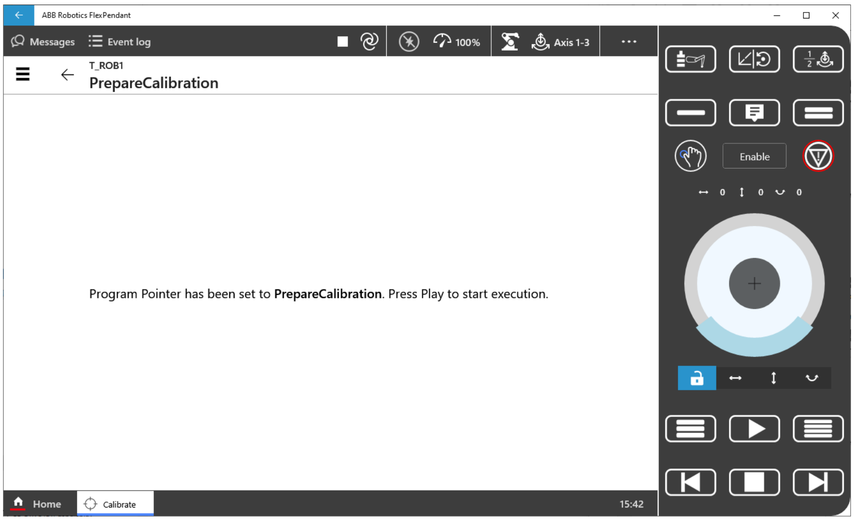
xx2100000365

Continues on next page

- Click **Yes** in the popped up dialog.



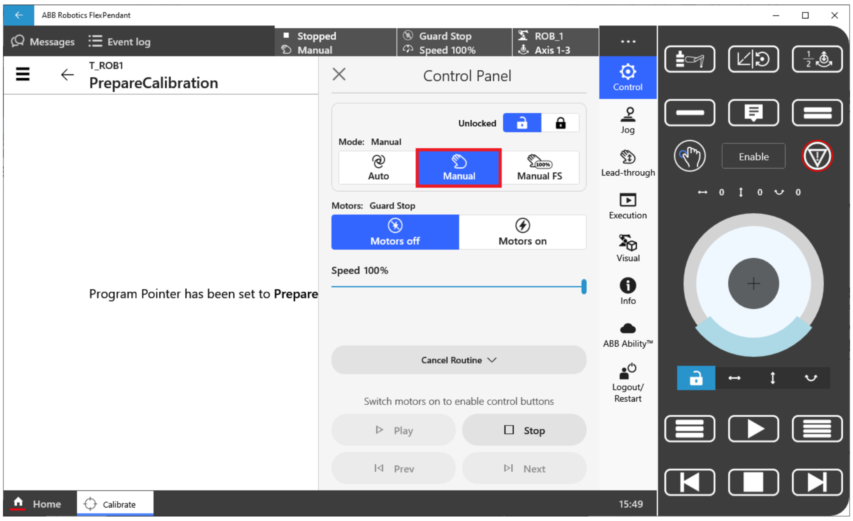
xx2100000366



xx2100000367

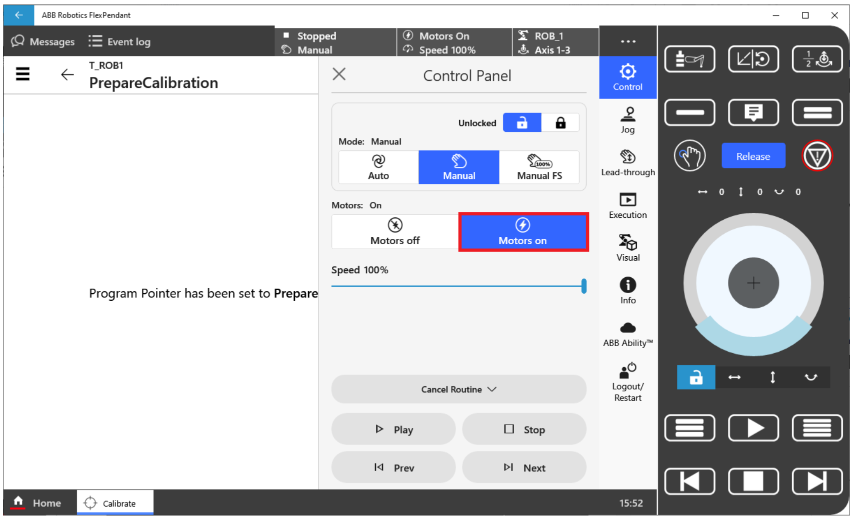
Continues on next page

- Set the controller to **Manual** mode.



xx2100000368

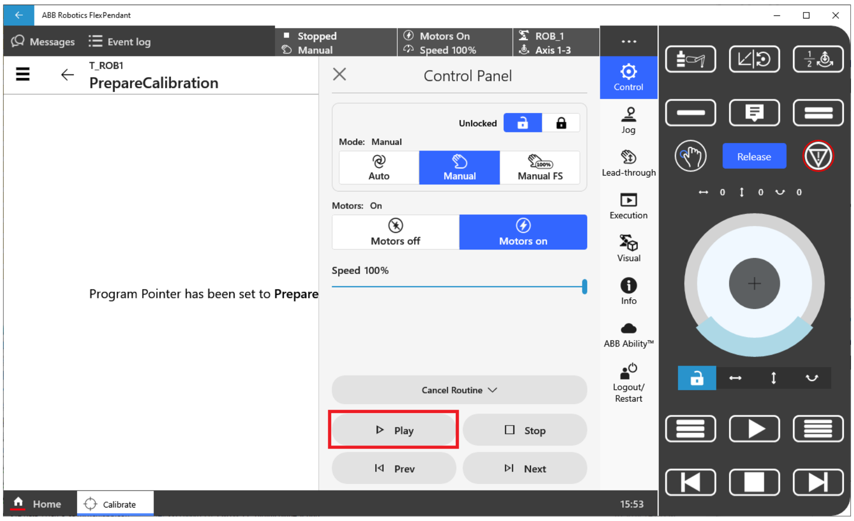
- Enable the Thumb button to motors on the controller.



xx2100000369

Continues on next page

- Click **Play**.



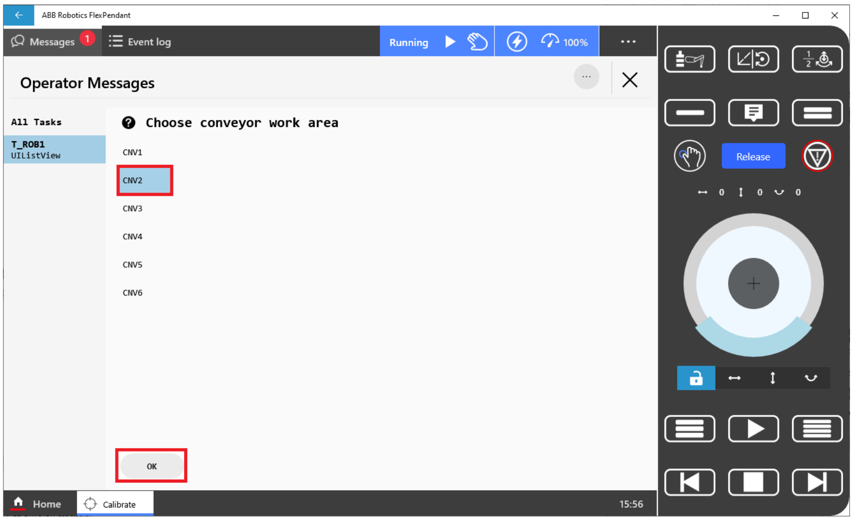
xx2100000370

- Select the work area type **Linear Conveyor**.

xx2100000371

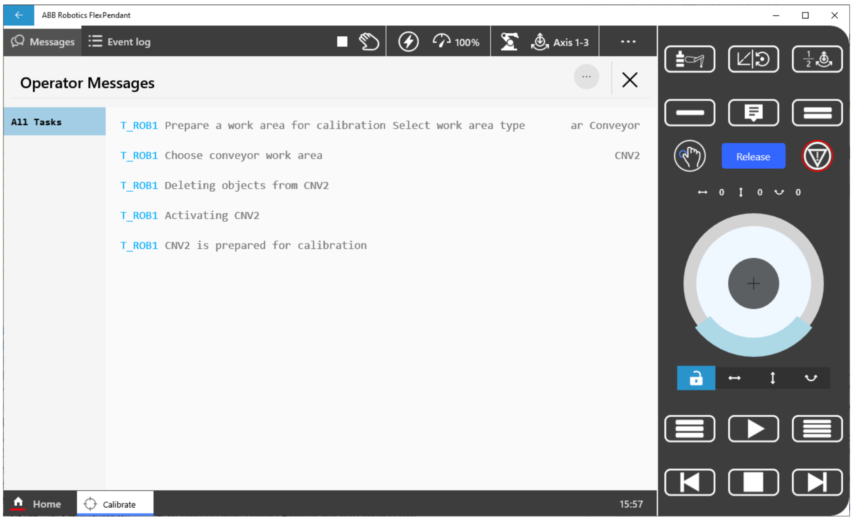
Continues on next page

- Select conveyor: for example, CNV2. Then click **OK**



xx2100000372

- Wait for the message **...is prepared for calibration**. The conveyor position in the jogging window for CNV2 should now be displayed as “0” mm.



xx2100000395

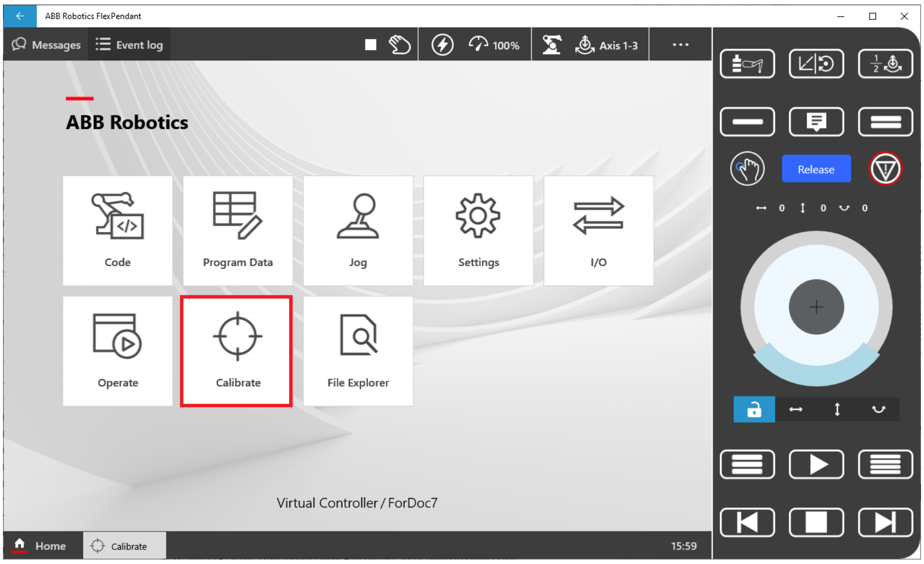
3 Move the conveyor belt forward until the reference point is just inside the working range of the next robot to calibrate.

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.

4 Mount the calibration tool on the robot.

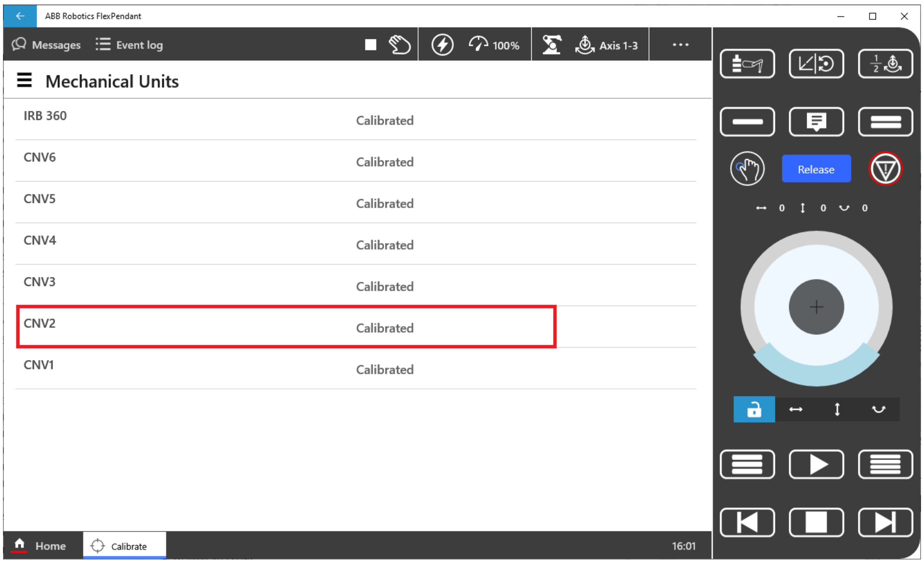
Continues on next page

5 Open the **Calibration** window in **Calibrate** on the FlexPendant.



xx2100000373

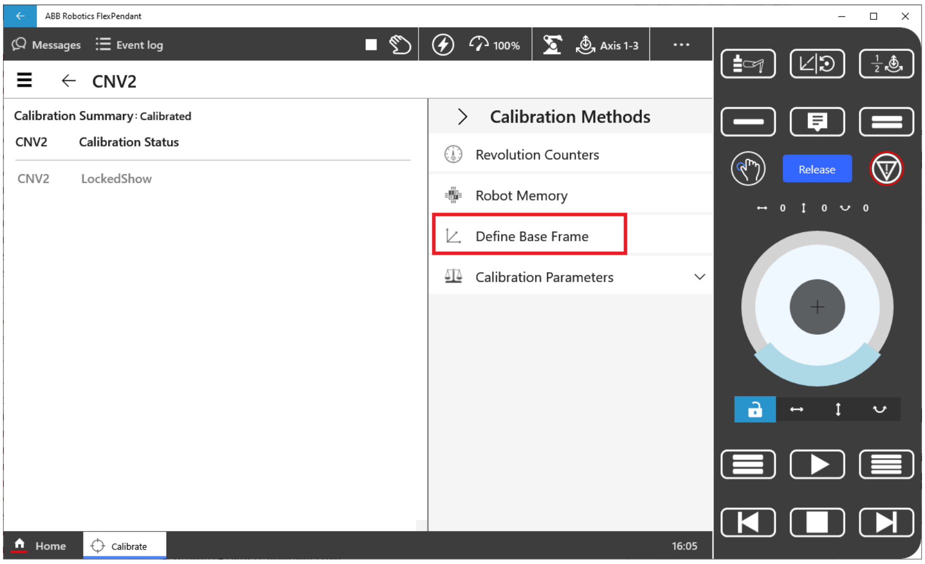
6 Select the conveyor, for example, CNV2.



xx2100000374

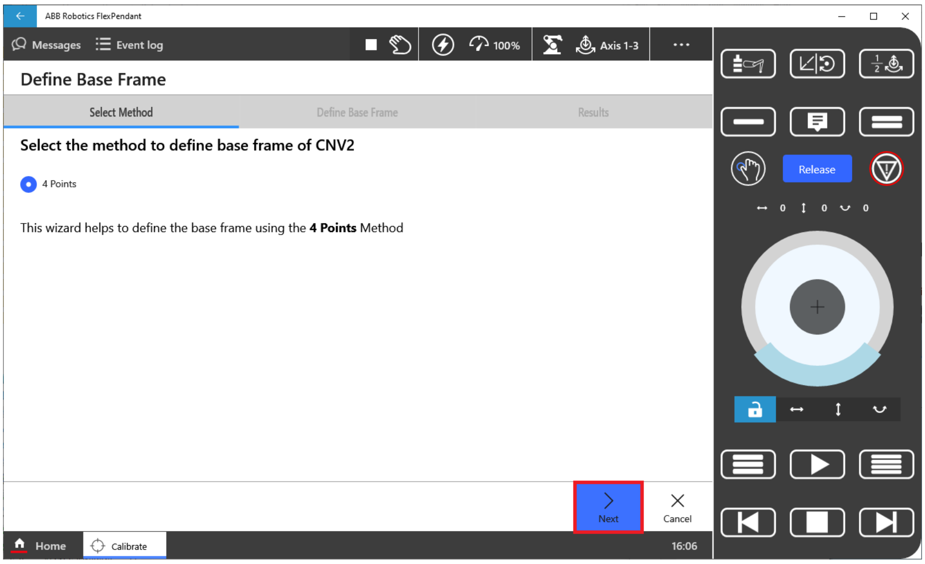
Continues on next page

7 Tap **Define Base Frame**.



xx2100000375

8 Tap **4 Point** and click **Next**.



xx2100000376

9 Select the robot, for example, **T_ROB1**.

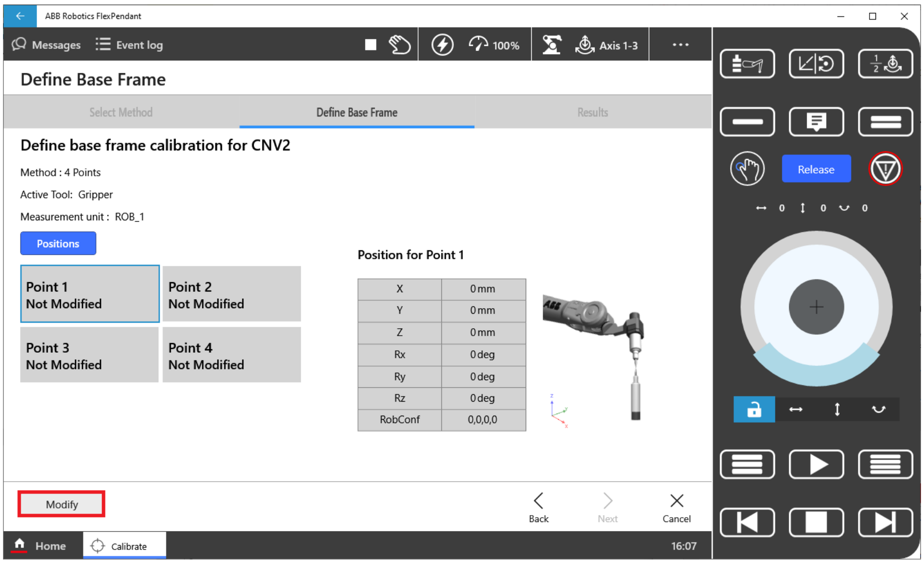
This step is required for MultiMove robots.

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.

Continues on next page

12 Modify the selected point (**Point 1**) by tapping the **Modify Position** function key.



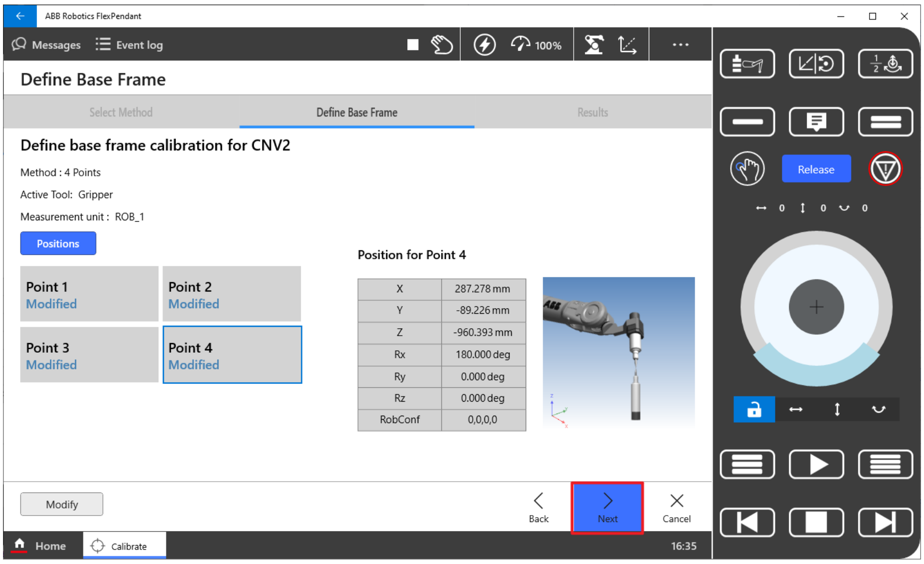
xx2100000377

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 **Next** to calculate the base frame.

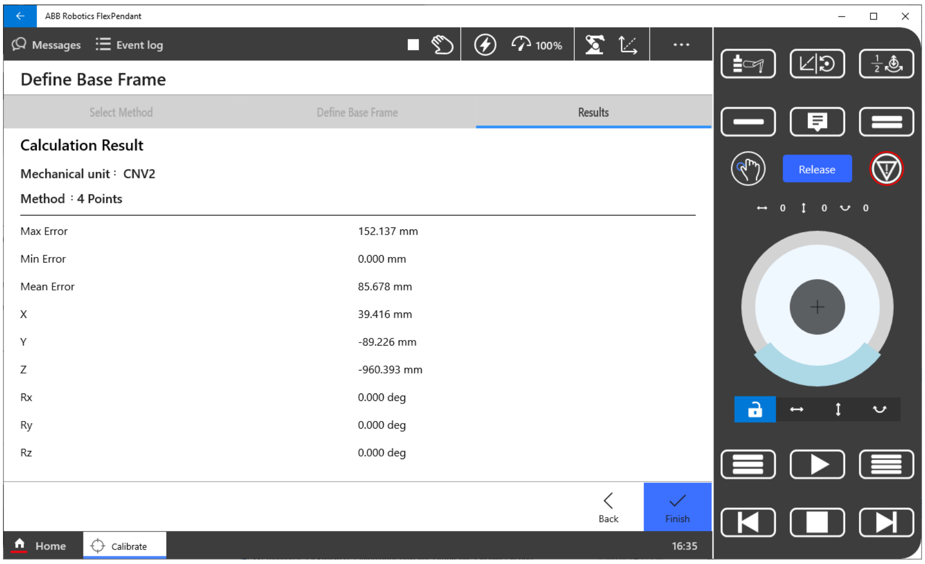


xx2100000378

Continues on next page

16 Check if the displayed mean error and max error of the base frame calculation is acceptable. If the estimated error is acceptable, tap **Finish** to confirm and store the new base frame.

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



xx2100000379

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.

Procedure for IRC5

Use the following procedure to calibrate all the base frames for a circular conveyor with IRC5 controller:

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

Continues on next page

are detected by the sensor. This point becomes the local origin of the detected items or containers.

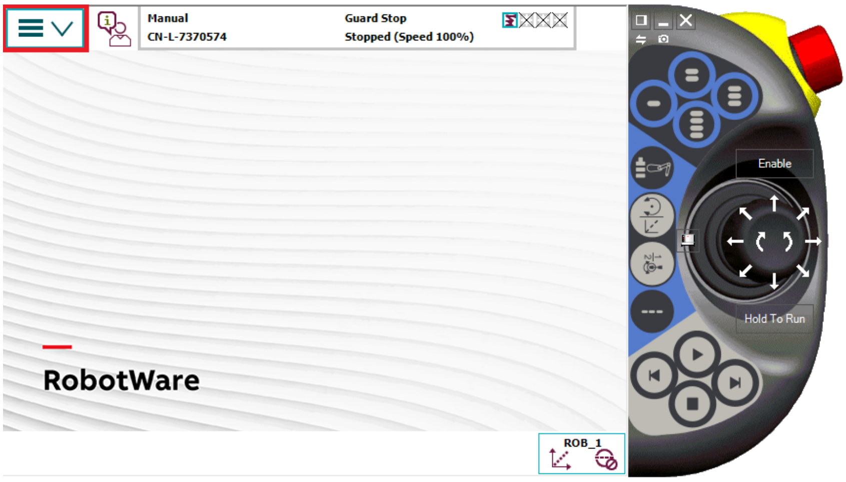
2 Reset the conveyor (encoder board) positions.

Note

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

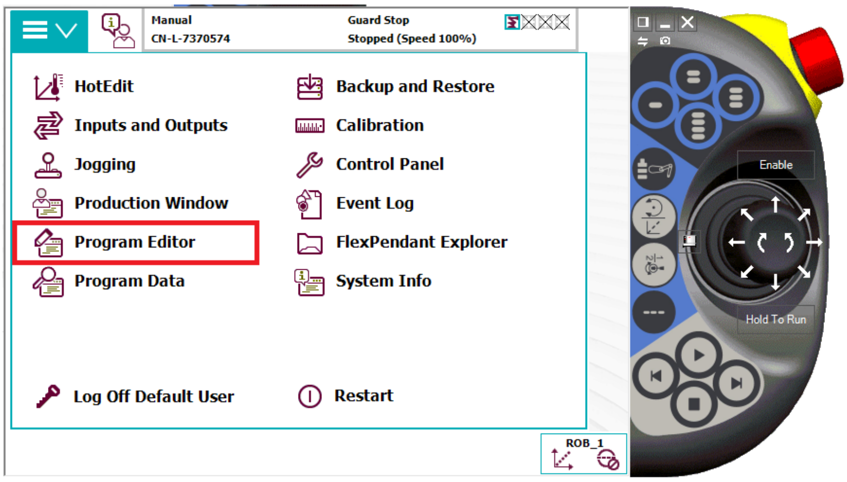
Do the following for all the robots having work areas that needs to be calibrated along the conveyor:

- In the FlexPendant, click **Menu** to open the drop-down list.



xx2200001925

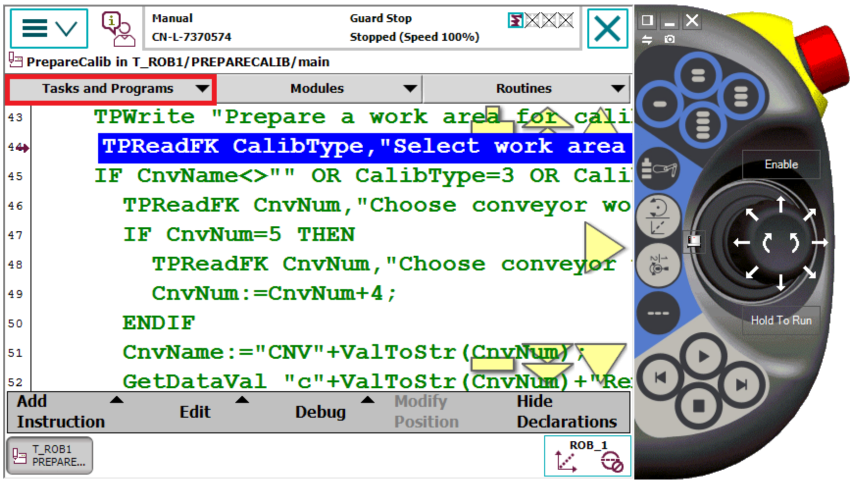
- Click **Program Editor** in the drop-down list.



xx2200001926

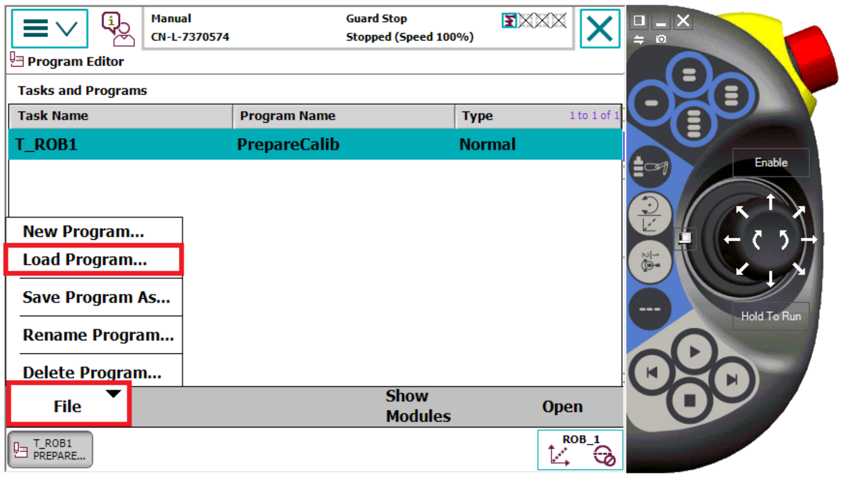
Continues on next page

- Click **Tasks and Programs**.



xx2200001927

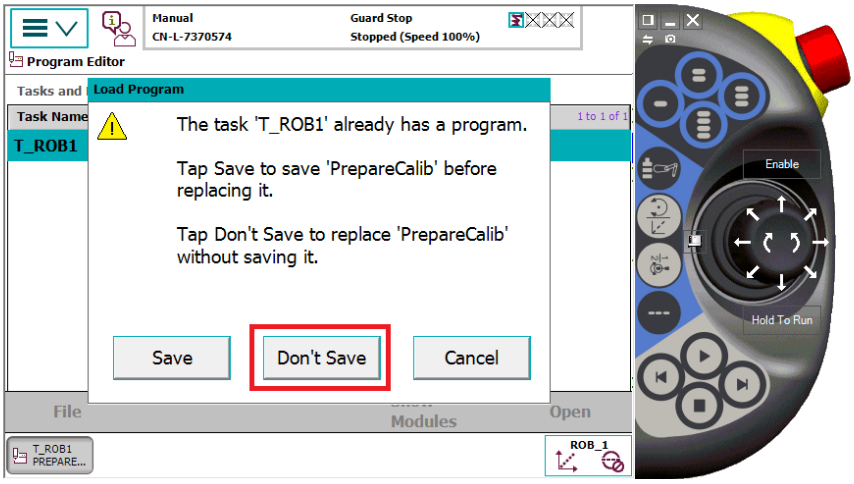
- Click **File** and **Load Program**.



xx2200001928

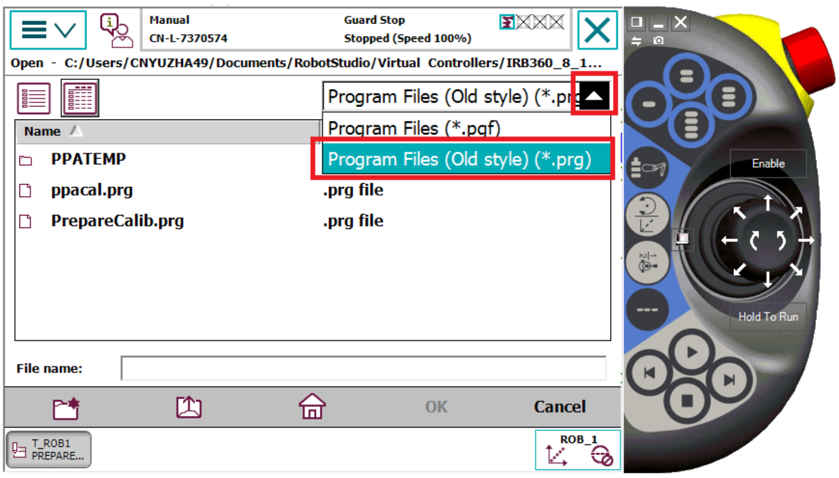
Continues on next page

- Click **Don't Save** in the popped up dialog.



xx2200001929

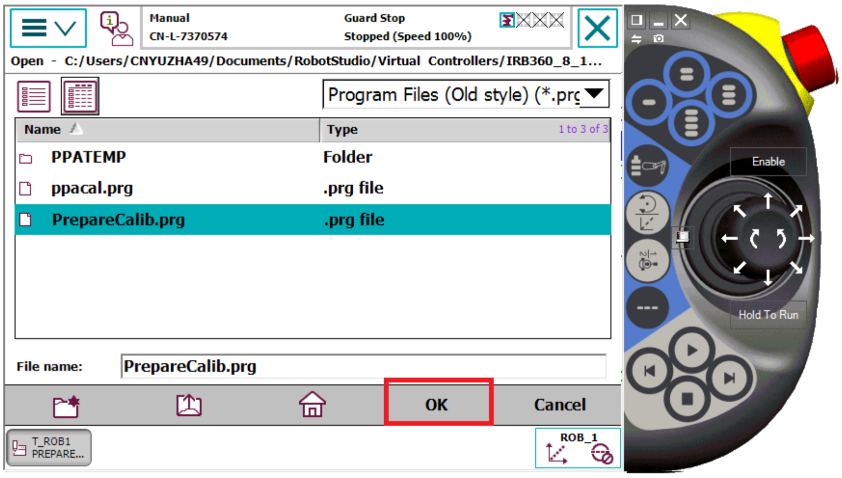
- Click **Program Files (Old style)(.prg)** on the right upper corner drop-down list.



xx2200001930

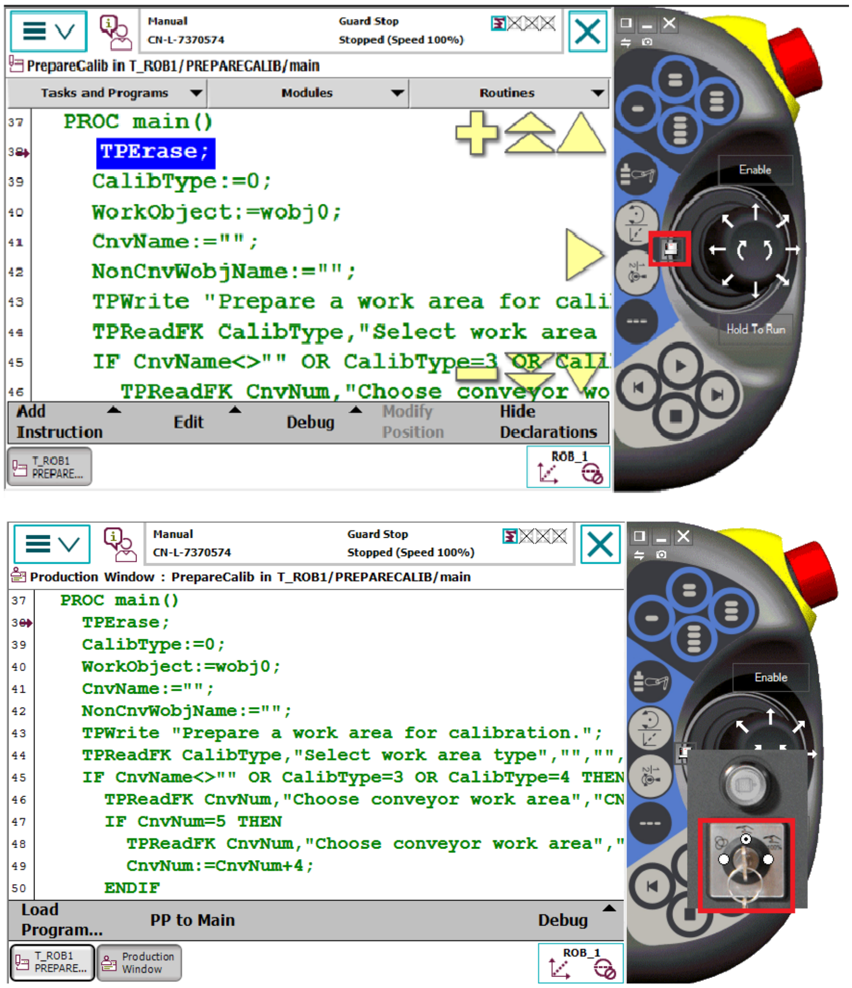
Continues on next page

- Select **PrepareCalib.prg** and click **OK**.



xx2200001931

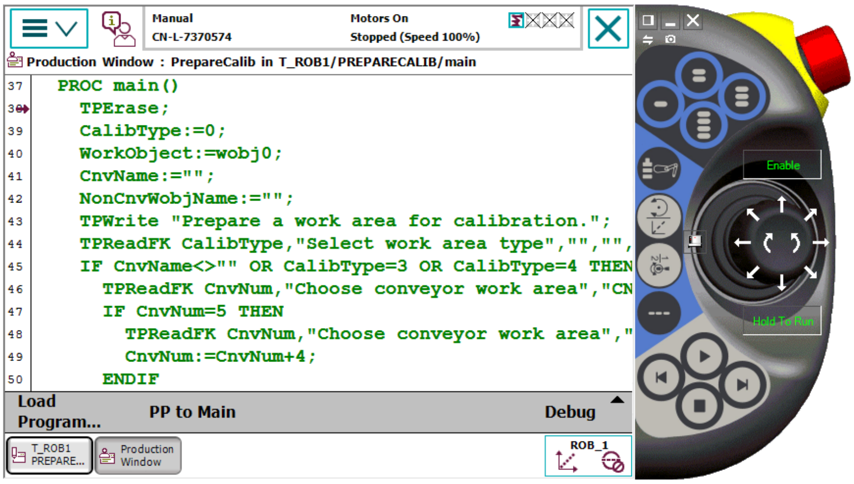
- Set the controller to **Manual** mode.



xx2200001932

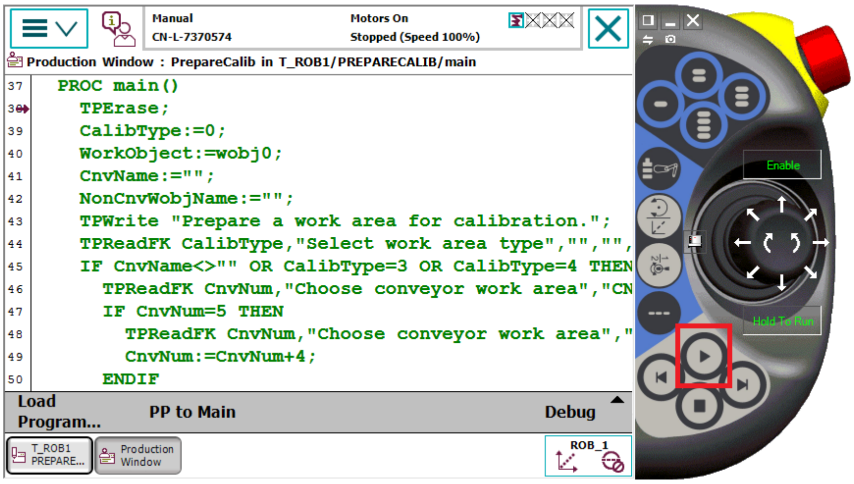
Continues on next page

- Enable the Thumb button to motors on the controller.



xx2200001933

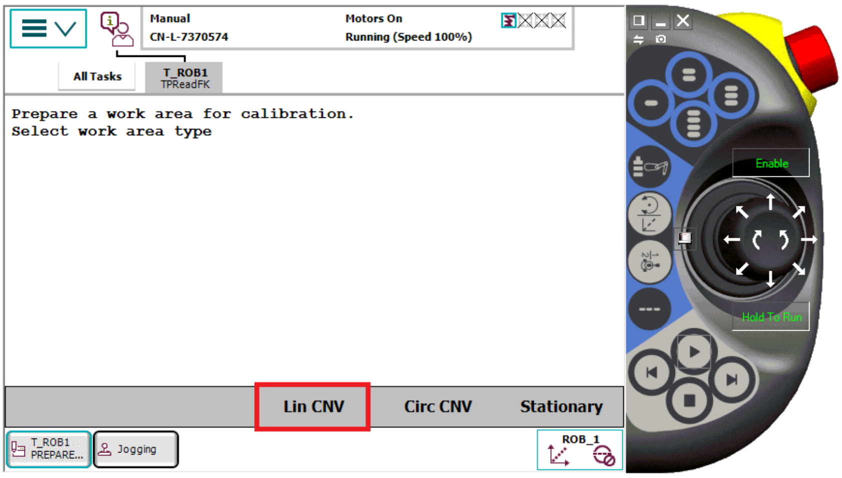
- Click **Play**.



xx2200001934

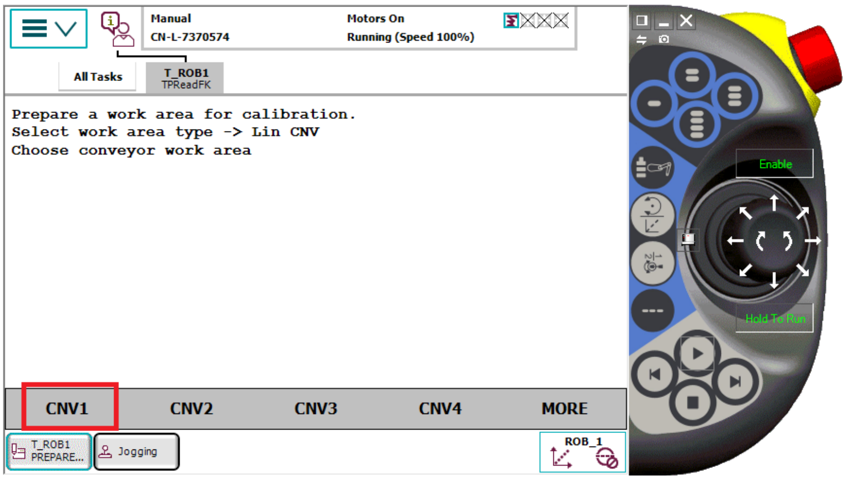
Continues on next page

- Select the work area type **Lin CNV**.



xx2200001943

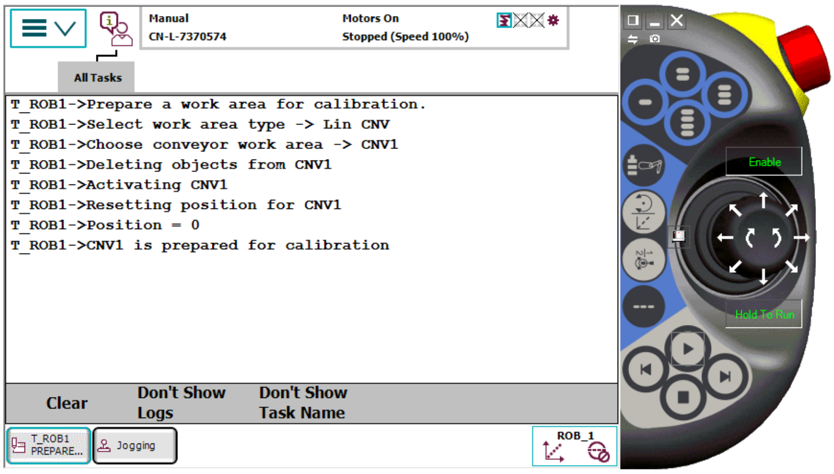
- Select conveyor: for example, **CNV1**.



xx2200001944

Continues on next page

- Wait for the message **...is prepared for calibration.** The conveyor position in the jogging window for CNV1 should now be displayed as “0” mm.



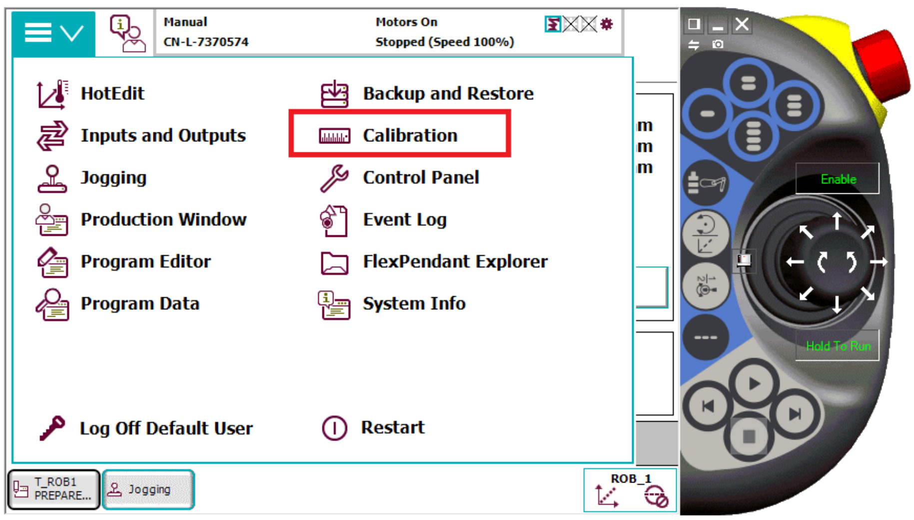
xx2200001945

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.

4 Mount the calibration tool on the robot.

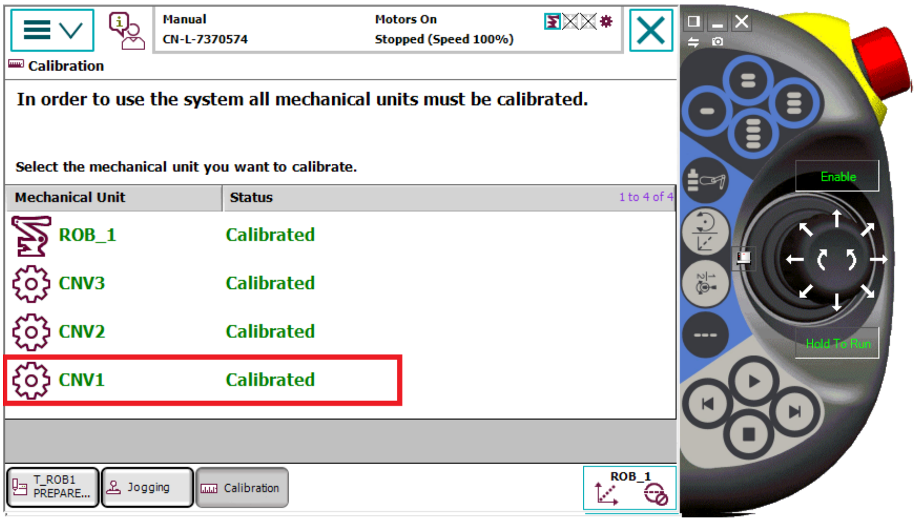
5 Open the **Calibration** window on the FlexPendant.



xx2200001946

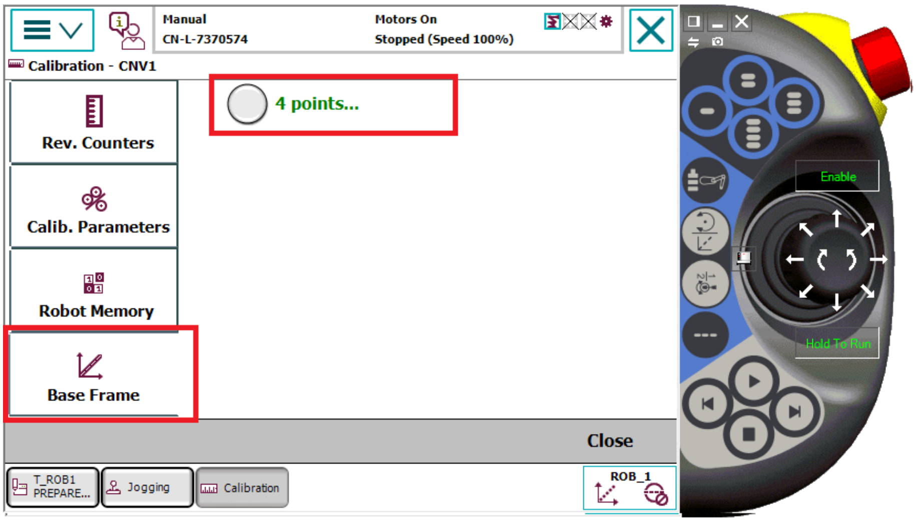
Continues on next page

6 Select the conveyor, for example, **CNV1**.



xx2200001947

7 Tap **Base Frame** and select **4 Point**.



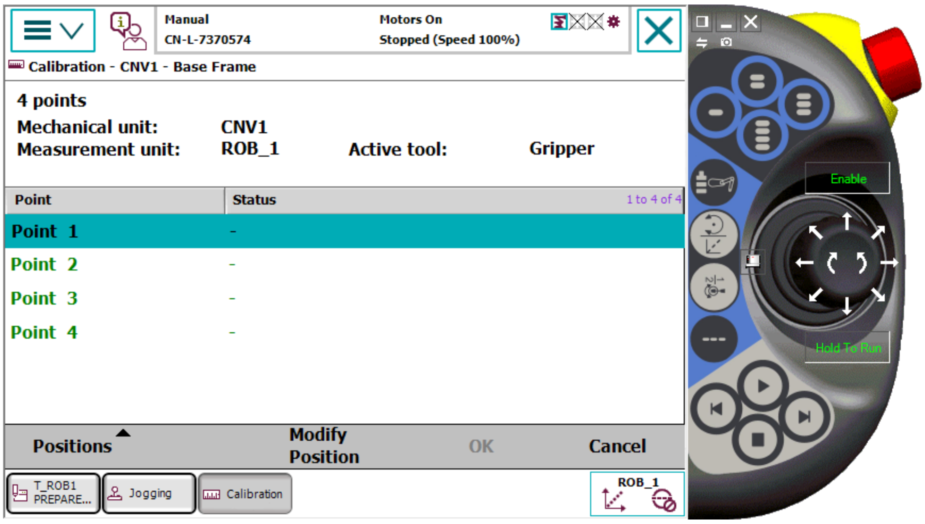
xx2200001948

8 Select the robot, for example, **T_ROB1**.

This step is required for MultiMove robots.

Continues on next page

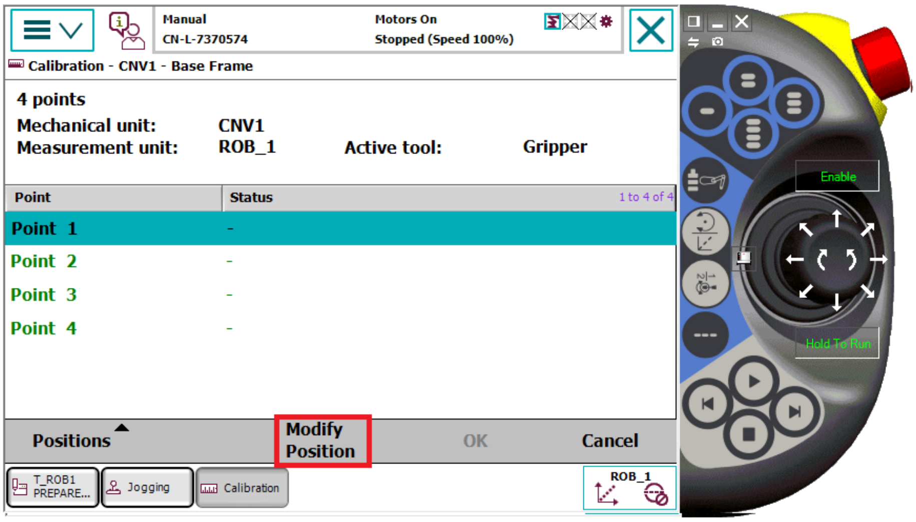
9 Select the first point **Point 1**.



xx2200001949

10 Jog or move the robot by hand. Point out the reference point on the conveyor accurately with the calibration tool TCP.

11 Modify the selected point (**Point 1**) by tapping the **Modify Position** function key.



xx2200001950

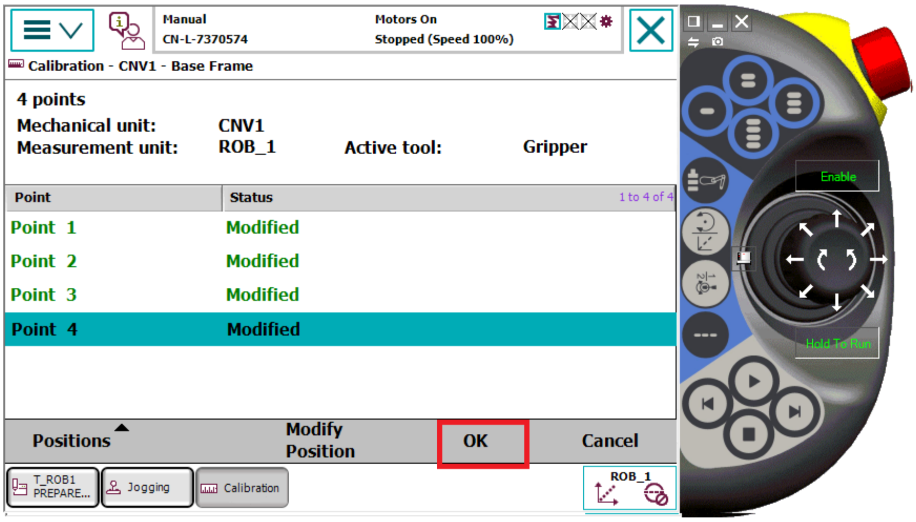
12 Move the conveyor belt forward a distance where the reference point still can be reached by the robot.

Long and equally spaced distances between the four calibration points (Point 1-4) are preferred since this increases the accuracy of the calibration.

13 Repeat the steps 10-13 for the points **Point 2**, **Point 3**, and **Point 4**.

Continues on next page

14 Tap **OK** to calculate the base frame.



xx2200001951

15 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.

16 If there are more robots to calibrate along the conveyor, continue from step 3.

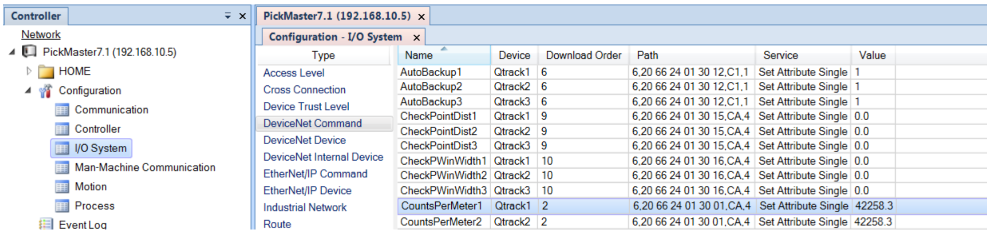
17 Restart the controllers to activate the new base frames.

4.4.4.2 Calibrating linear conveyor with DSQC 377

4.4.4.2.1 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 *DeviceNet Command*, in the topic *I/O System*.



xx1900000551

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 factory have a preset value. For an IRC5 system this value is 20,000. This value can be used to start the calibration with.
`measured_meters`(mm)	The manually measured distance in meters 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.

Continues on next page

For example: `(1010*20000)/1005 = 20099`

7. In RobotStudio, click **Configuration** and select topic **I/O System** and type **DeviceNet 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.

Related information

Application manual - Conveyor tracking.

Technical reference manual - System parameters.

4.4.4.2.2 Defining the base frame

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

- Define the `Counts Per Meter` system parameter for each conveyor work area. For more details, see *Defining the parameter Counts Per Meter on page 210*, *Defining the parameter Counts Per Meter on page 187*.
- 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, see *Calibrating camera on page 273*. After calibrating the camera, keep the camera calibration pattern attached to the conveyor.

Procedure

Use the following procedure to calibrate all the base frames for a conveyor in the line with IRC5 controller:

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.

   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.

Continues on next page

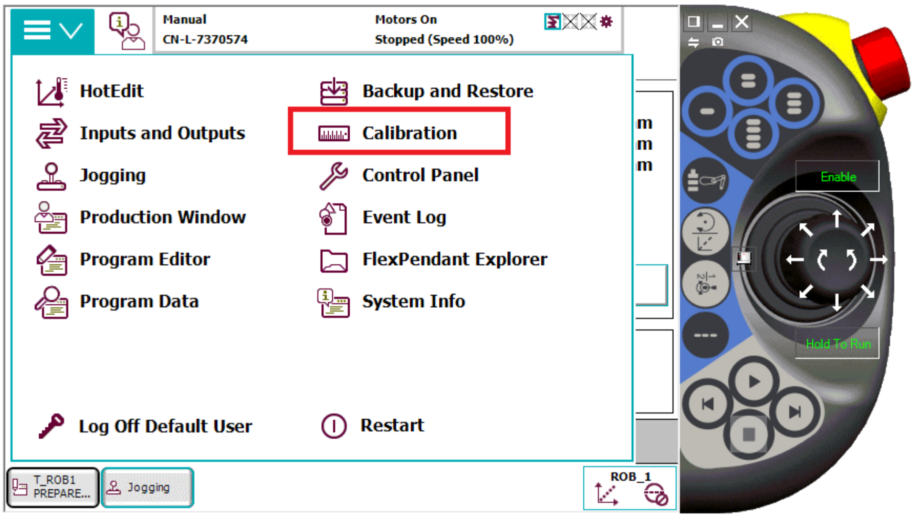
- Start the loaded rapid program
  - Select calibration type: Conveyor.
  - Select conveyor: for example, CNV1.
  - Wait for the message **READY FOR CALIB**. The conveyor position in the jogging window for CNV1 should now be displayed as “0” mm.

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.

4 Mount the calibration tool on the robot.

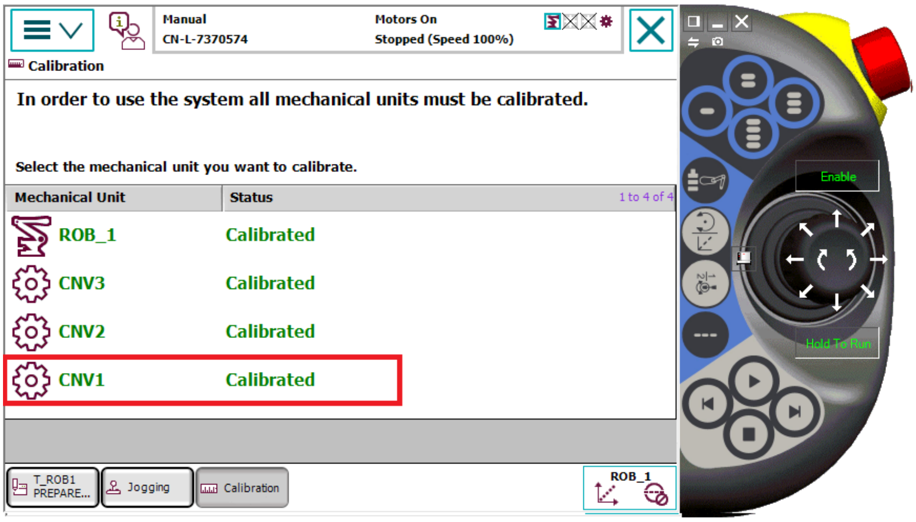
5 Open the **Calibration** window on the FlexPendant.



xx2200001946

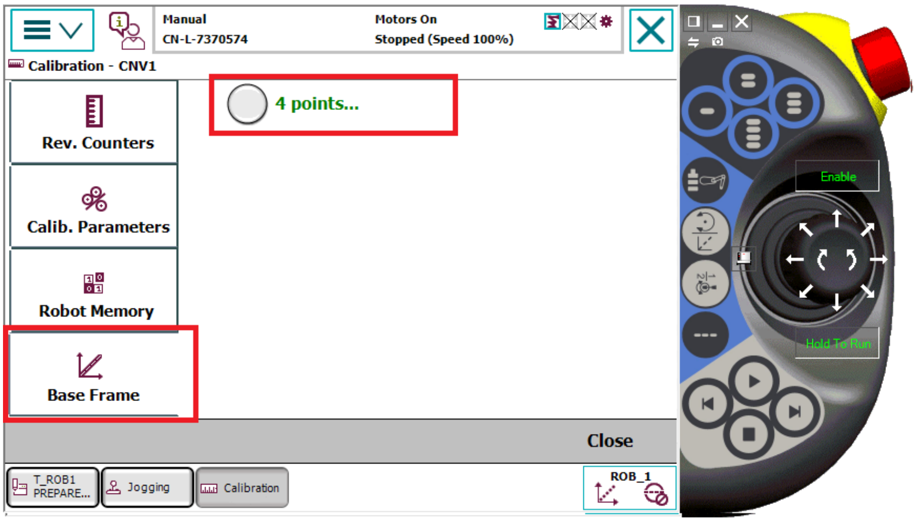
Continues on next page

6 Select the conveyor, for example, **CNV1**.



xx2200001947

7 Tap **Base Frame** and select **4 Point**.



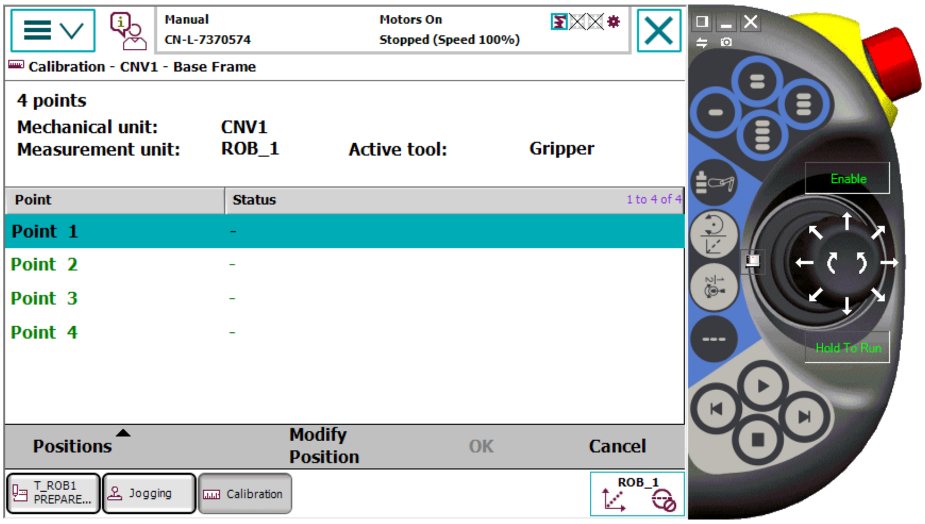
xx2200001948

8 Select the robot, for example, **T_ROB1**.

This step is required for MultiMove robots.

Continues on next page

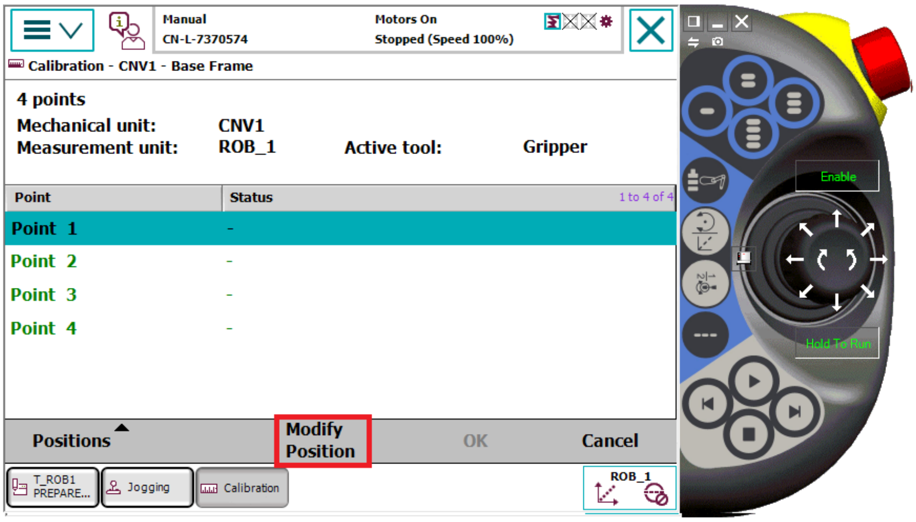
9 Select the first point **Point 1**.



xx2200001949

10 Jog or move the robot by hand. Point out the reference point on the conveyor accurately with the calibration tool TCP.

11 Modify the selected point (**Point 1**) by tapping the **Modify Position** function key.



xx2200001950

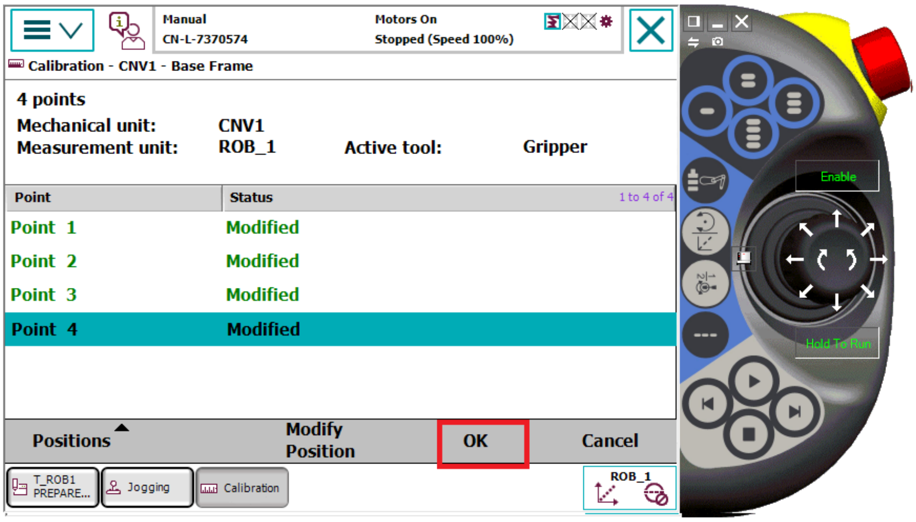
12 Move the conveyor belt forward a distance where the reference point still can be reached by the robot.

Long and equally spaced distances between the four calibration points (Point 1-4) are preferred since this increases the accuracy of the calibration.

13 Repeat the steps 10-13 for the points **Point 2**, **Point 3**, and **Point 4**.

Continues on next page

14 Tap **OK** to calculate the base frame.



xx2200001951

15 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.

16 If there are more robots to calibrate along the conveyor, continue from step 3.

17 Restart the controllers to activate the new base frames.

4.4.5 Calibrating circular conveyor

Overview

Note

The following calibration process is required when running production and emulation. Calibration under the simulation tab in PickMaster PowerPac will not complete the following calibration process.

The calibrations needed for the circular conveyors 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 solution so all recipe in that solution could share the same calibration. If you need to re-calibrate a camera, all recipes in the solution 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 circular conveyor:

1. If needed, modify the cable connections from encoder to DSQC 2000 if the circular conveyor is moving towards clockwise, see *Cable connections from encoder to DSQC 2000 for different conveyor rotating directions on page 219*.

   Note

   Counter clockwise rotation is set as the default rotating direction of the circular conveyor in PickMaster PowerPac with the default hardware connection to DSQC 2000 and software settings.

   If the user want to make the circular conveyor rotate in clockwise which is not the default direction, modify the hardware connection is prerequisite.

   Tip

   Clockwise rotation of circular conveyor has not been validated for DSQC 377.

Continues on next page

2. Define the parameter *Counts Per Meter* (for conveyors only), see *Defining the parameter Counts Per Meter on page 221*.

   Note

   In the circular conveyor, the parameter *Counts Per Meter* indicates counts per radian.

3. Define the base frame, see *Defining the base frame on page 223*, *Defining the base frame on page 251*.

Continues on next page

4.4.5.1 Calibrating circular conveyor with DSQC 2000 (CTM)

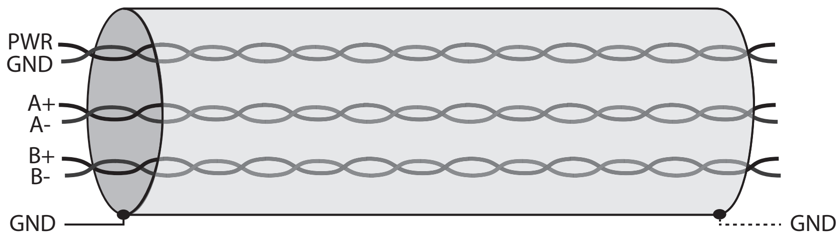
4.4.5.1.1 Cable connections from encoder to DSQC 2000 for different conveyor rotating directions

Introduction

Counter clockwise rotation is set the default movement direction of the circular conveyor in PickMaster PowerPac with the default hardware connection and software settings. For the information, see *Application manual - Conveyor tracking*.

If the user need to set the circular conveyor rotate in clockwise which is not the default direction, modify the hardware connection is prerequisite.

The encoder is connected to the encoder interface on the DSQC 2000. The cable illustration is as below.

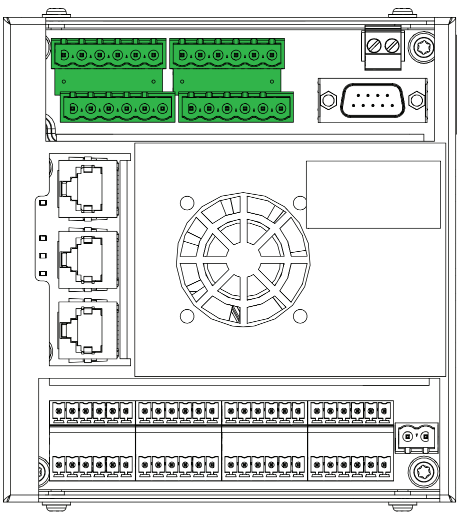


xx1800001539

PWR	Power
GND	Ground
A+, A-	Encoder signal A (0°)
B+, B-	Encoder signal B (90°)

Encoder connections on CTM

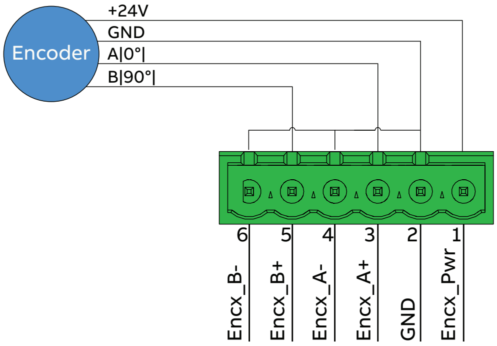
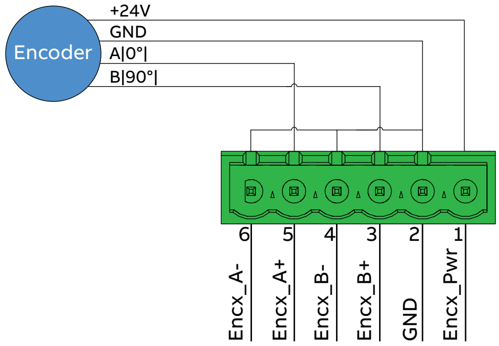
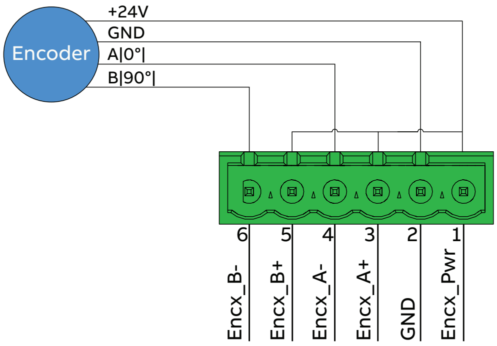
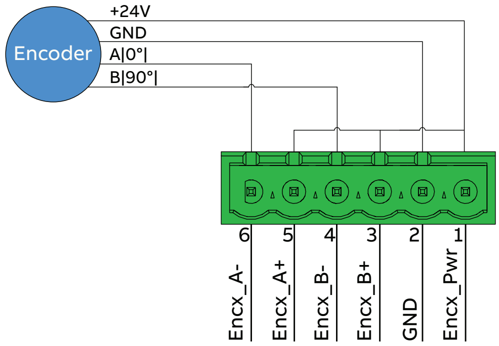
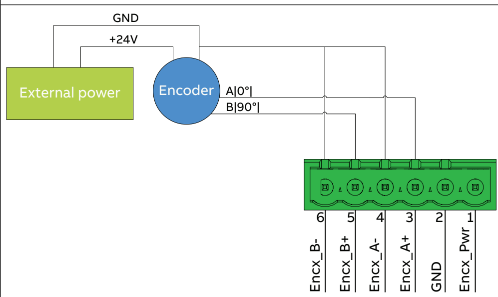
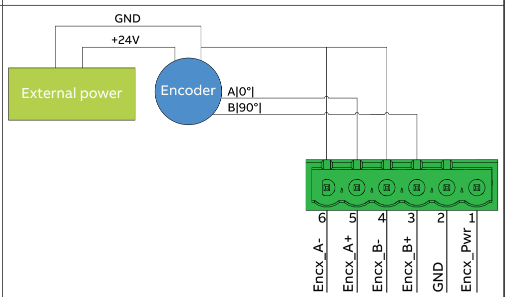
The CTM has four independent encoder interfaces supporting PNP, Push-Pull, and NPN encoders.



xx2400000415

Continues on next page

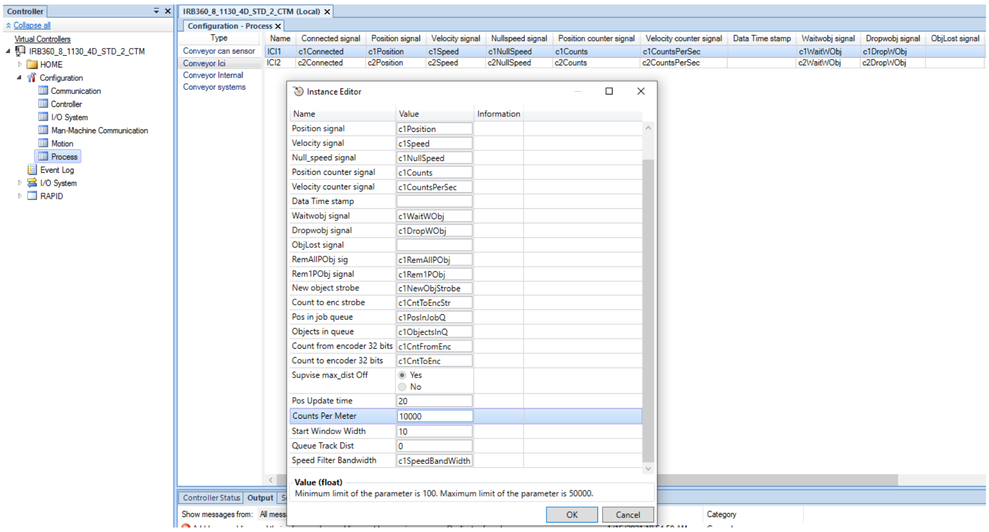
The cable connection from encoder to the CTM should vary with the rotating direction of the conveyor as the following table.

	Counter clockwise rotation	Clockwise rotation
PNP & Push/Pull type encoders	 xx2400000416	 xx2400000417
NPN type encoders	 xx2400000418	 xx2400000419
Encoder is externally powered	 xx2400000420	 xx2400000421

4.4.5.1.2 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 *Conveyor Ici*, in the topic *Process*.



xx2100000042

> **Tip**
>
> For circular conveyor, the parameter *Counts Per Meter* indicates counts per degree.

Calculation for Counts Per Meter

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

`counts value/measured_radians`

Value	Description
`position1/counts value`	Read from predefined I/O signal on the FlexPendant or RobotStudio. For example, CNV1, the signal name is c1counts.
`old_counts_per_meter`	The encoder's old value. **Note** The encoders delivered from factory have a preset value. For an IRC5 system this value is 20,000. This value can be used to start the calibration with.
`measured_radians`(radian)	The manually measured radians that the conveyor has been moved.

When this variable is applied to a circular conveyor, the actual meaning is *counts per radian*.

Continues on next page

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 outside of the conveyor at the same location.
2. Hot start it to set the current position of the conveyor to zero as `position0`.

   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. Rotate the conveyor belt approximately 360 degrees.
4. In the FlexPendant **Jogging** window, read the position of the conveyor. This is `position1`.
5. Measure the physical radians of the conveyor. This is the value `measured_radians`.
6. Calculate *Counts Per Meter* using the read and measured values.

   For example: `60000/1 = 60000`
7. In RobotStudio, click **Configuration** and select topic **Process** and type **Conveyor Ici**.
8. Edit the unit *ICIx* (where x is the number of the conveyor) and update the value for parameter *Counts Per Meter*.
9. Tap **OK**.
10. Restart the controller.

Related information

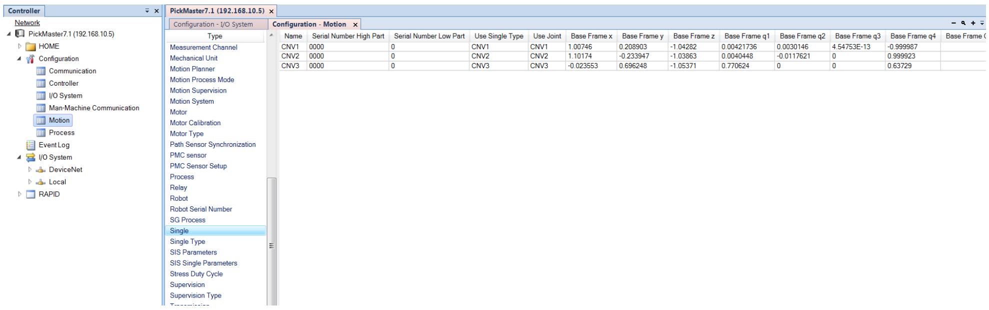
Application manual - Conveyor tracking.

Technical reference manual - System parameters.

4.4.5.1.3 Defining the base frame

Introduction

For each conveyor work area on a circular 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.



xx1900000592

Preparations

- Define the `Counts Per Meter` system parameter for each conveyor work area. For more details, see *Defining the parameter Counts Per Meter on page 249*, *Defining the parameter Counts Per Meter on page 221*.
- 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, see *Calibrating camera on page 273*. After calibrating the camera, keep the camera calibration pattern attached to the conveyor.

Recommendation

This section describes how to use TCP measurements and RAPID programs to calculate the conveyor base frame position and quaternion for a circular conveyor.

This method uses three measured points on the circular conveyor to calculate the center of rotation. The three points should be spaced as far apart as possible around the periphery.

Procedure for OmniCore

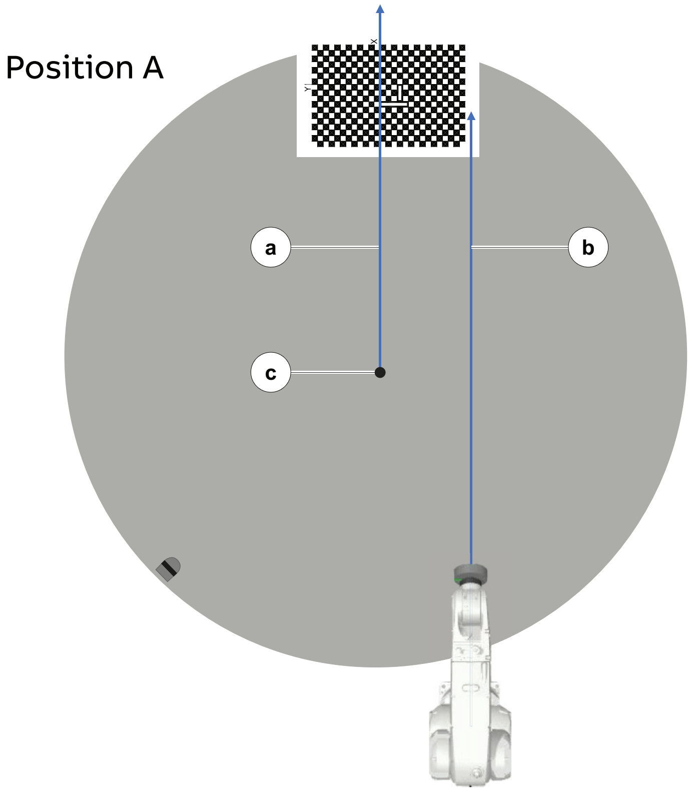
Use the following procedure to calibrate all the base frames for a circular conveyor with OmniCore controller:

1. Mount the calibration tool on the robot.
2. Place the calibration grid X-aligned with the center line(a).

   Make sure that the grid X is pointing outwards of the circular conveyor.
3. Rotate the belt to position A, which make the center line be parallel with the X-axis(b) of the calibrating robot.

Continues on next page

Center line is a line connecting the centre point(c) of the circular conveyor and the X-axis on the calibration grid paper.



xx2200002007

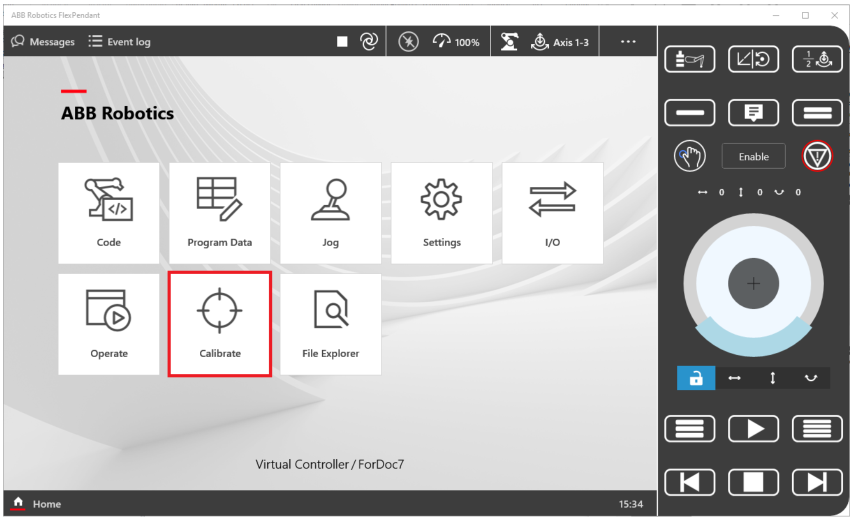
4 Reset the conveyor (encoder board) positions at position A.

Note

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

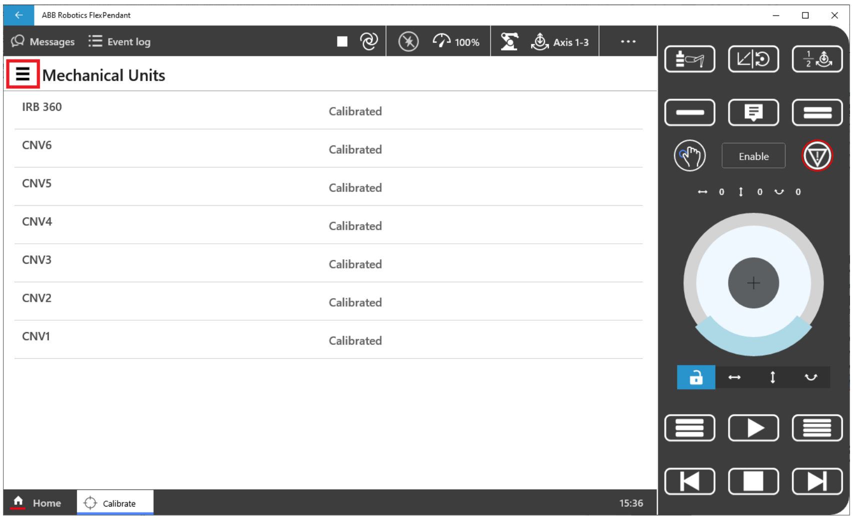
Continues on next page

A In the FlexPendant, click **Calibrate**.



xx2100000362

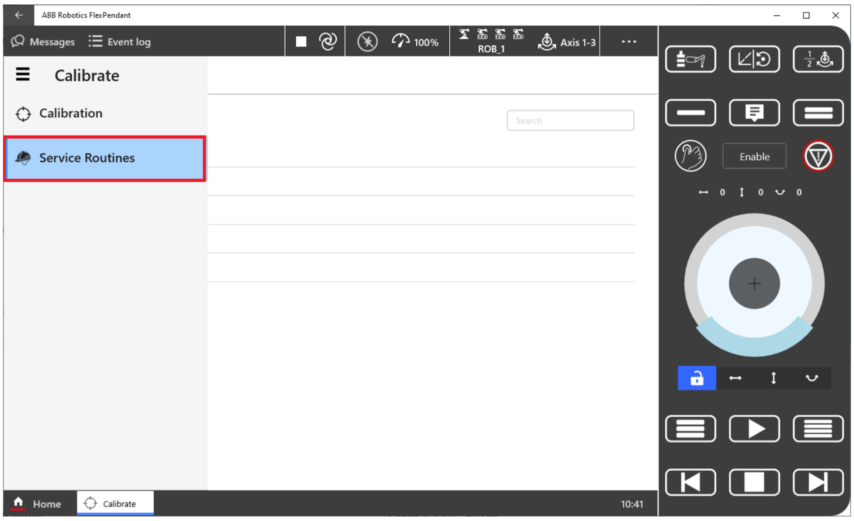
B Click **Option Tab** on the up left corner.



xx2100000363

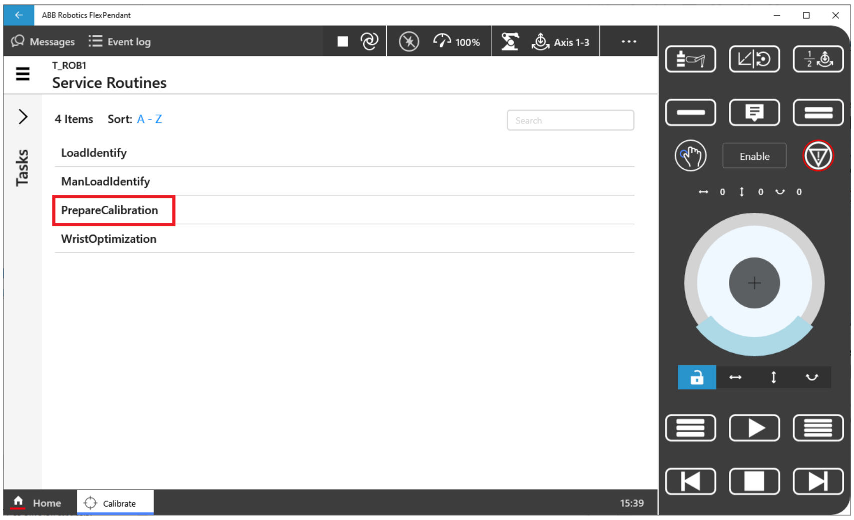
Continues on next page

C Click **Service Routines.**



xx2100000364

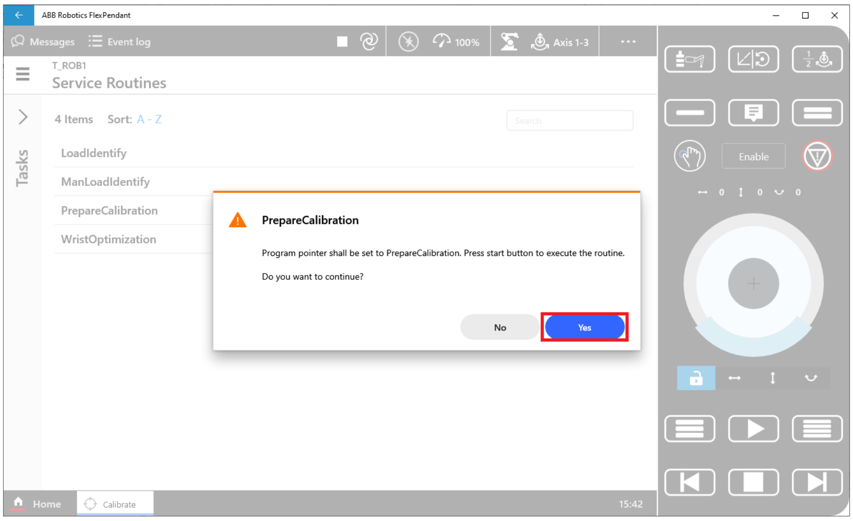
D Click **PrepareCalibration.**



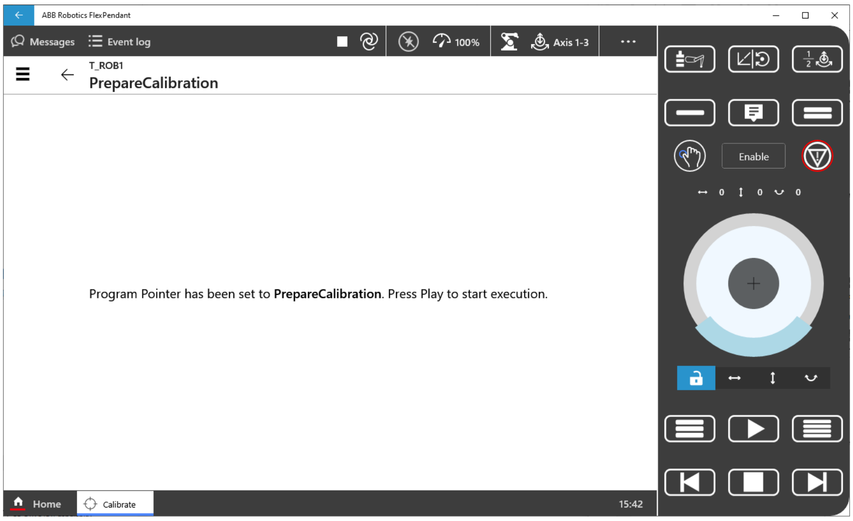
xx2100000365

Continues on next page

E Click **Yes** in the popped up dialog.



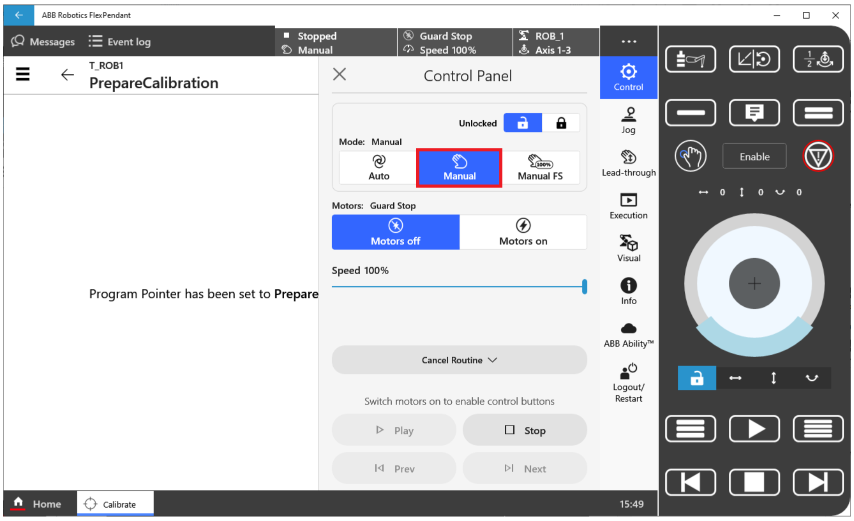
xx2100000366



xx2100000367

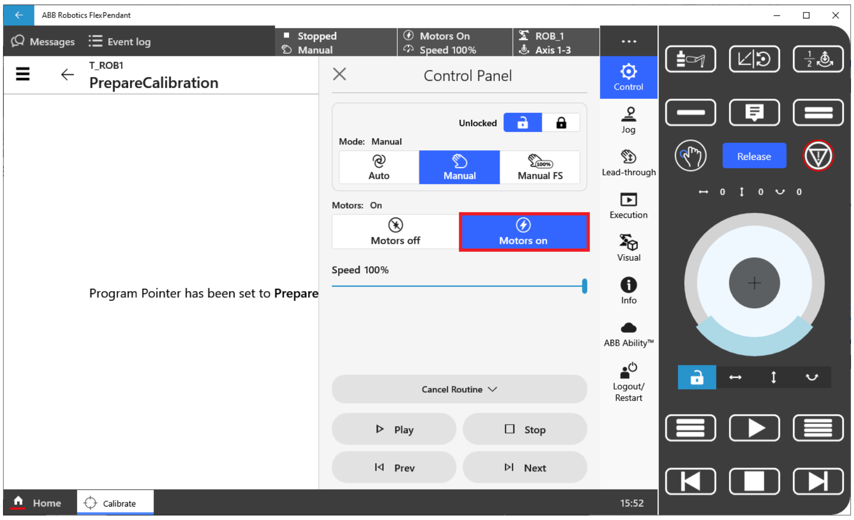
Continues on next page

F Set the controller to **Manual** mode.



xx2100000368

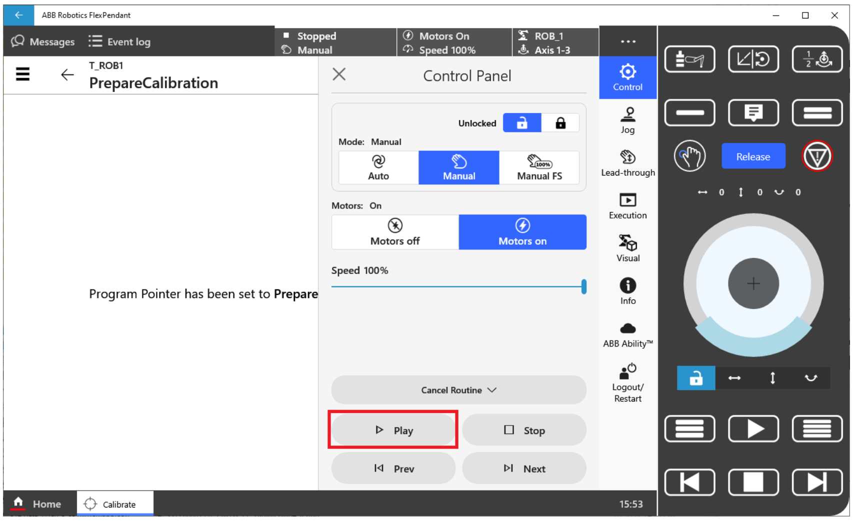
G Enable the Thumb button to motors on the controller.



xx2100000369

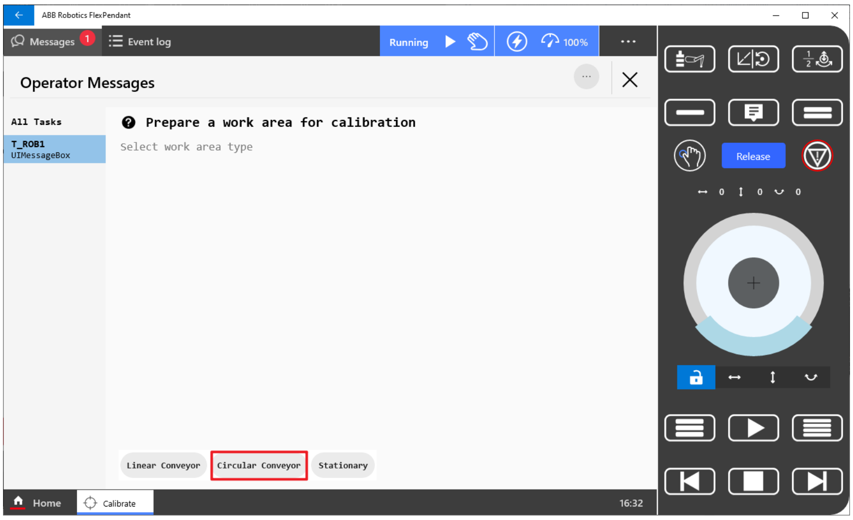
Continues on next page

H Click **Play**.



xx2100000370

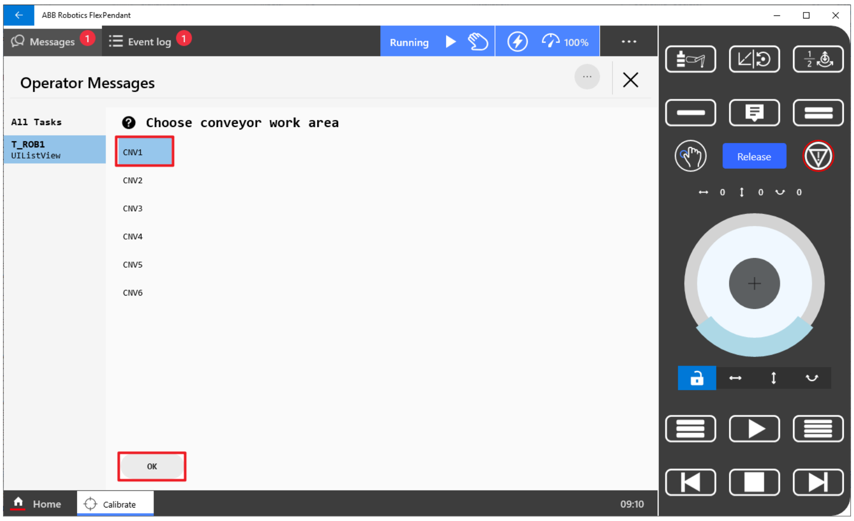
I Select the work area type **Circular Conveyor**.



xx2100000690

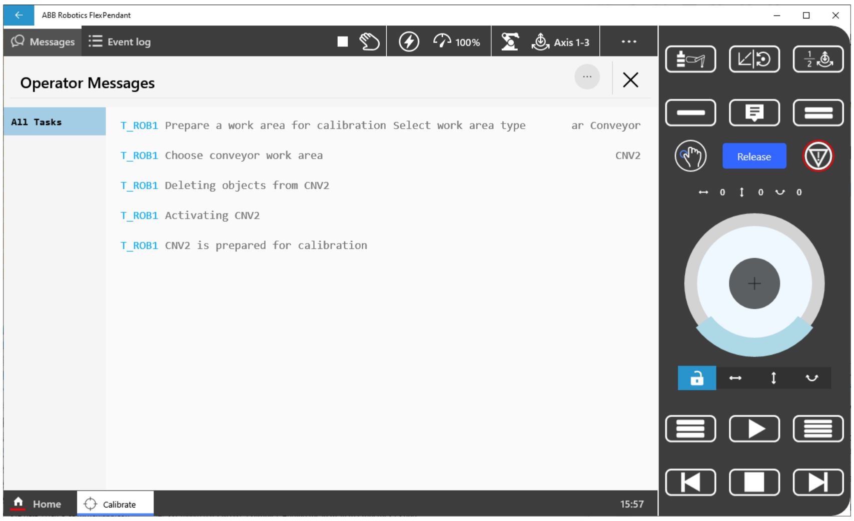
Continues on next page

J Select conveyor: for example, CNV1. Then click **OK**



xx2100000691

K Wait for the message **...is prepared for calibration.** The conveyor position in the jogging window for CNV1 should now be displayed as “0” mm.



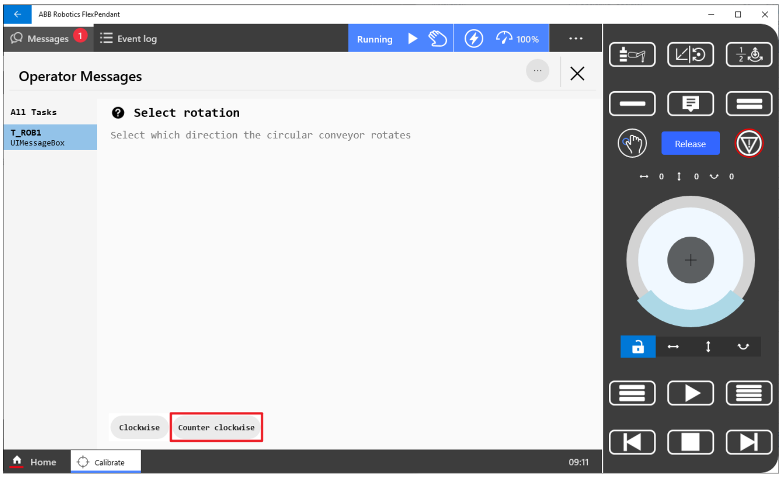
xx2100000395

Continues on next page

5 Do the following steps for the robot having work areas that needs to be calibrated along the conveyor:

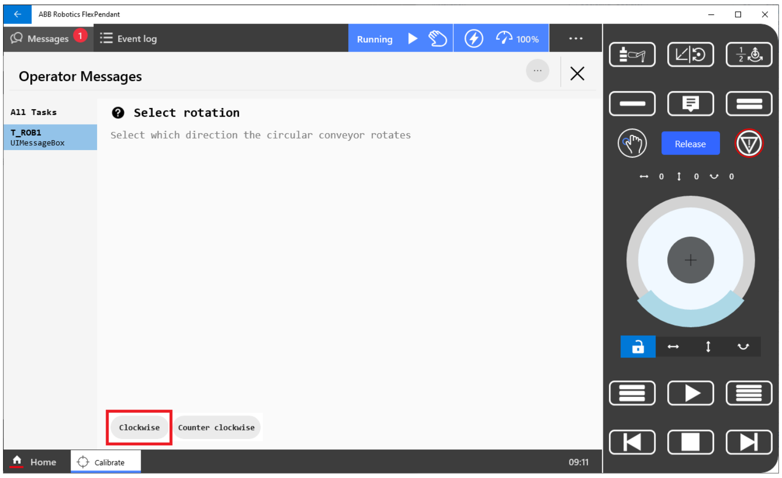
A When the message **Select rotation** pops up, click to select the moving direction of the conveyor.

For counter clockwise moving direction, select **Counter clockwise**.



xx2100000692

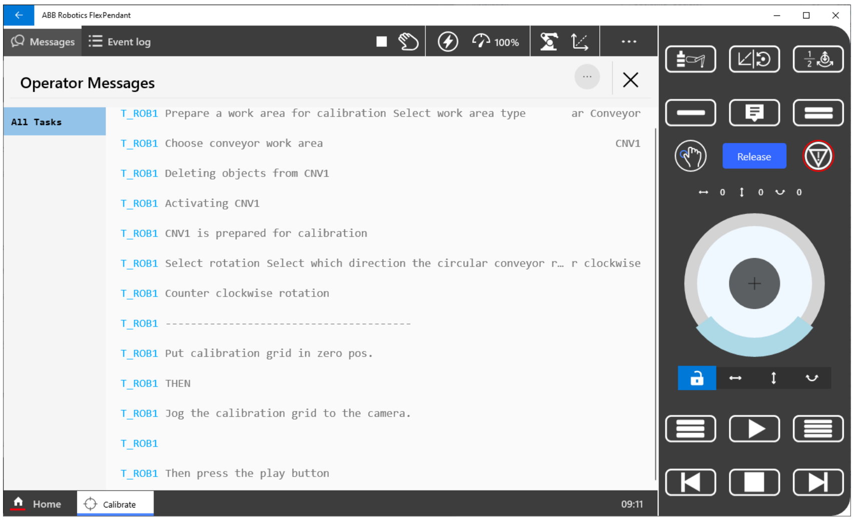
For clockwise moving direction, select **Clockwise**.



xx2400000430

Continues on next page

B The program will continue automatically.



xx2100000693

C Rotate the belt to position B (reference point), which make the calibration grid under the camera(d).

If selected **Counter clockwise**, rotate the belt in counter clockwise direction.

If selected **Clockwise**, rotate the belt in clockwise direction.

Tip

Rotate the conveyor belt by hand to the reference point.

- If source type is Predefined and trigger type as Distance, the reference point is origin of the Hotspot.
- If source type is Predefined and Trigger as I/O sensor, Hotspot is used to generate predefined items or containers, the reference point should be at the point where the objects are detected by the sensor. This point becomes the local origin of the detected items or containers.
- If source type is vision, the reference point is the local origin of the camera view.

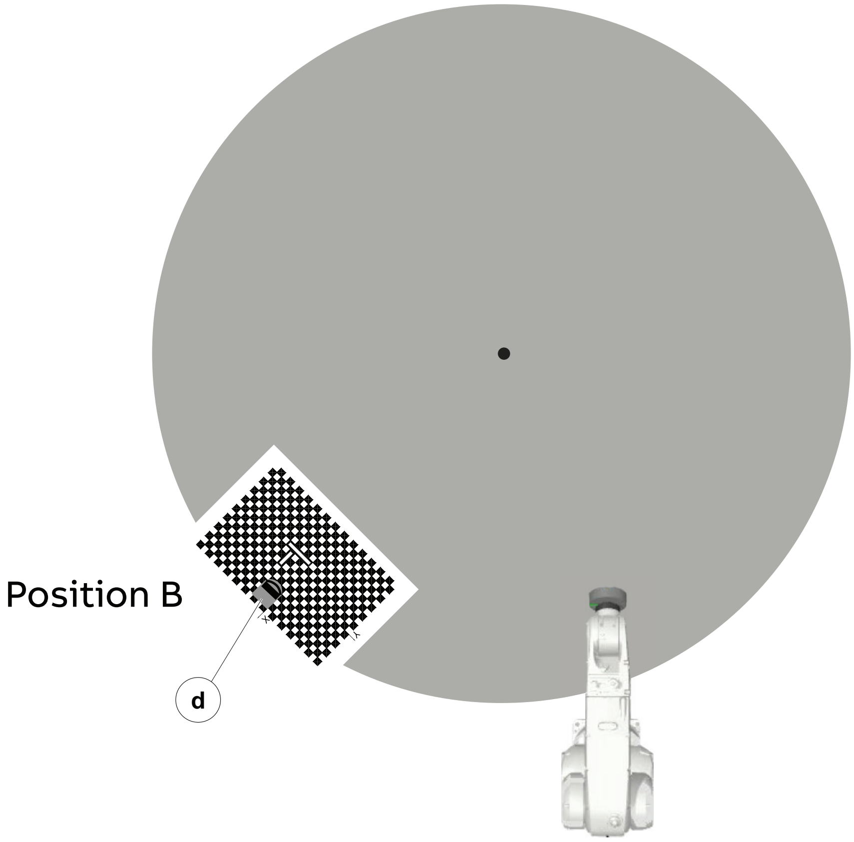
  If a camera is used, calibrate the camera at the same time, see *Calibrating camera on page 273*.

  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.

Note

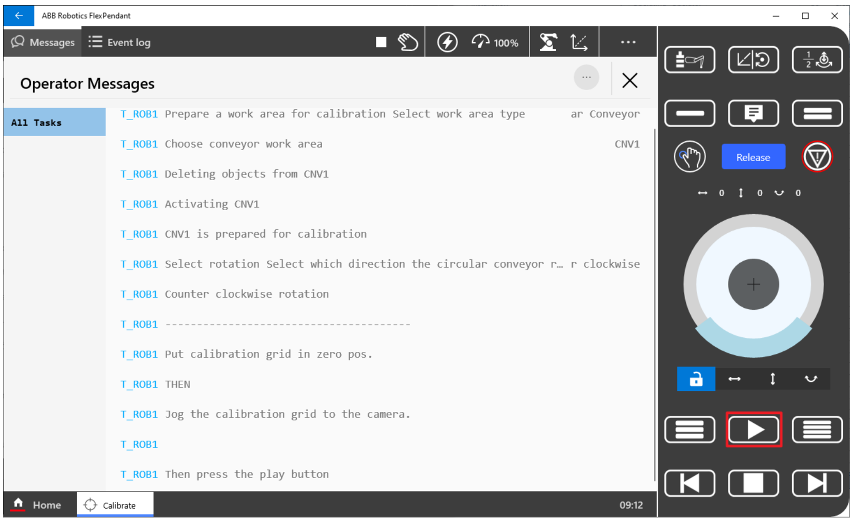
Do not rotate the belt back and forth. This will result in inaccurate calibration data.

Continues on next page



xx2200002008

D Click **Play** on the FlexPendant of the robot(s) which have been reset.



xx2100000694

E 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 three calibration points (Point 1-3) are preferred since this increases the accuracy of the calibration.

Continues on next page

If selected **Counter clockwise**, rotate the belt in counter clockwise direction.

If selected **Clockwise**, rotate the belt in clockwise direction.

> **Note**
>
> Do not rotate the belt back and forth. This will result in inaccurate calibration data.

F Jog or move the robot by hand. Point out the reference point on the conveyor accurately with the calibration tool TCP.

G Modify the point (**Pos 1**) by tapping **Play**.

H Repeat the steps from F to H for the points **Pos 2** and **Pos 3**.

I Check if the displayed mean error and max error of the base frame calculation is acceptable. If the estimated error is acceptable, restart the system to confirm and store the new base frame.

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

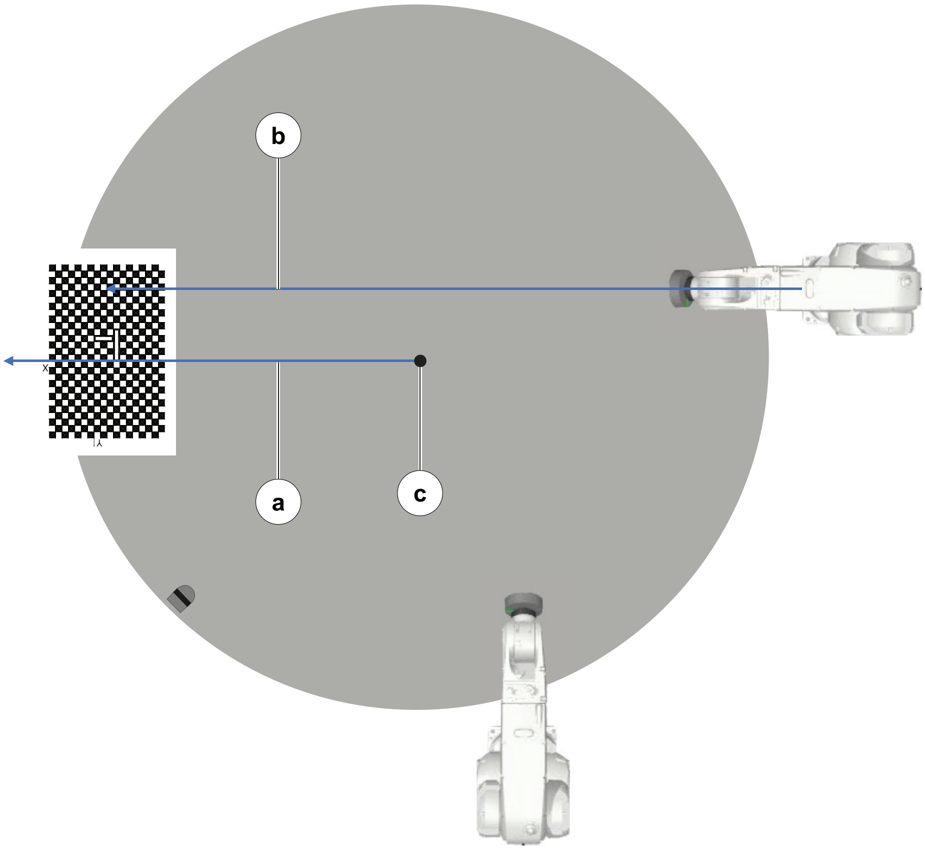
xx2100000695

If the estimated error is not ok, this base frame must be re-calibrated.

> **Tip**
>
> Read the value of **CAMERA OFFSET FROM CENTER**. This value will be used as the input of `Sensor offset` in *Type configuration for circular conveyor on page 255*.

Continues on next page

6 If there are more robots need to be calibrating, repeat from step 3 to step 5 for each robot.



xx2400000405

Procedure for IRC5

Use the following procedure to calibrate all the base frames for a circular conveyor with IRC5 controller:

1 Mount the calibration tool on the robot.

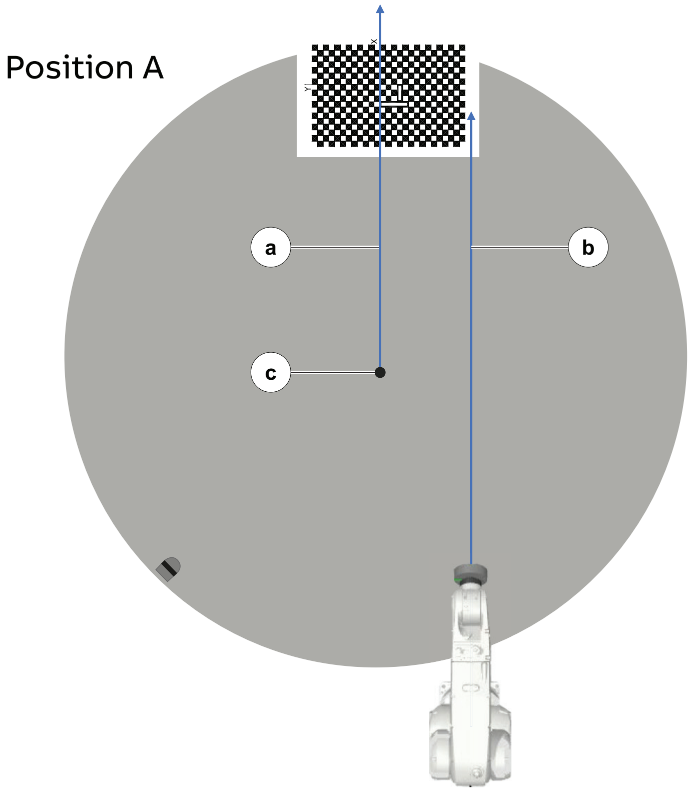
2 Place the calibration grid X-aligned with the center line(a).

   Make sure that the grid X is pointing outwards of the circular conveyor.

3 Rotate the belt to position A, which make the center line be parallel with the X-axis (b) of the calibrating robot.

Continues on next page

Center line is a line connecting the centre point(c) of the circular conveyor and the X-axis on the calibration grid paper.

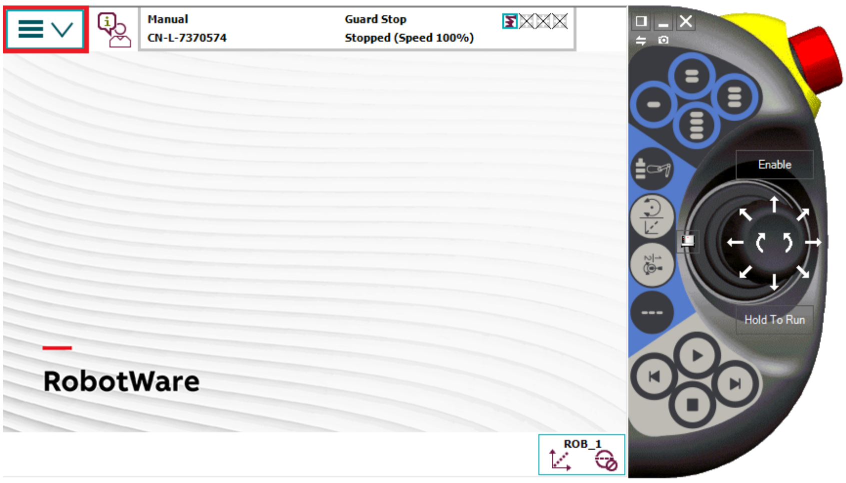


xx2200002007

4 Reset the conveyor (encoder board) positions at position A.

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

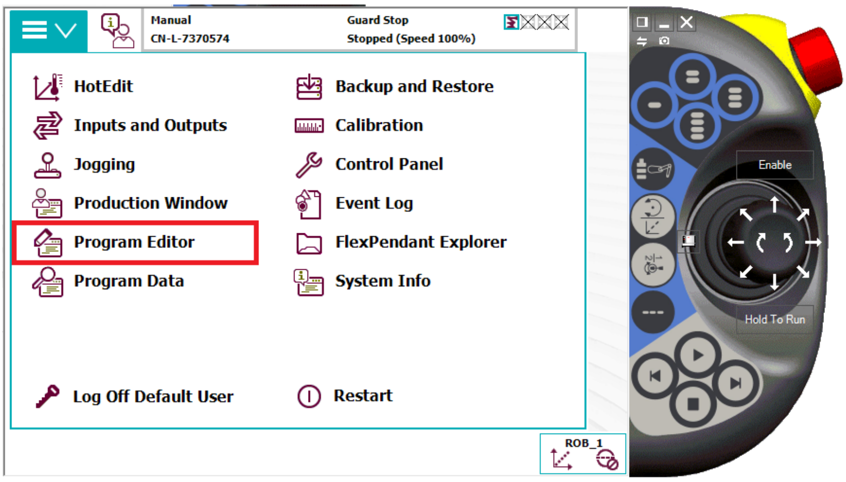
A In the FlexPendant, click **Menu** to open the drop-down list.



xx2200001925

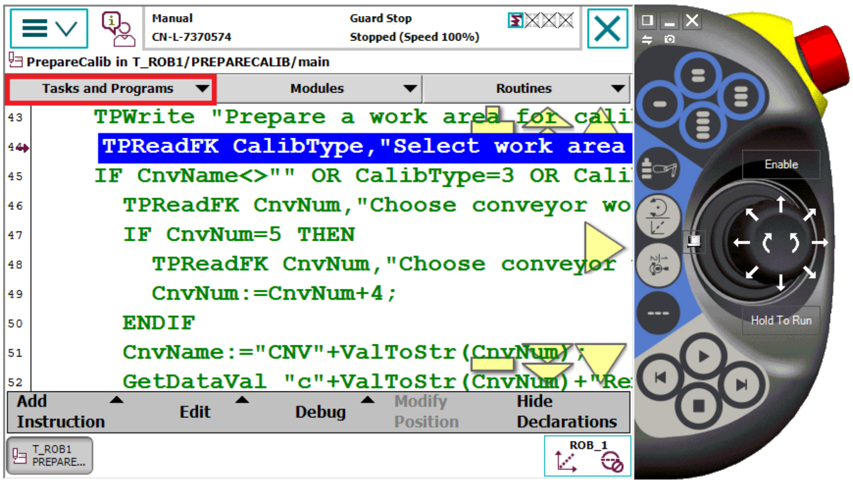
Continues on next page

B Click **Program Editor** in the drop-down list.



xx2200001926

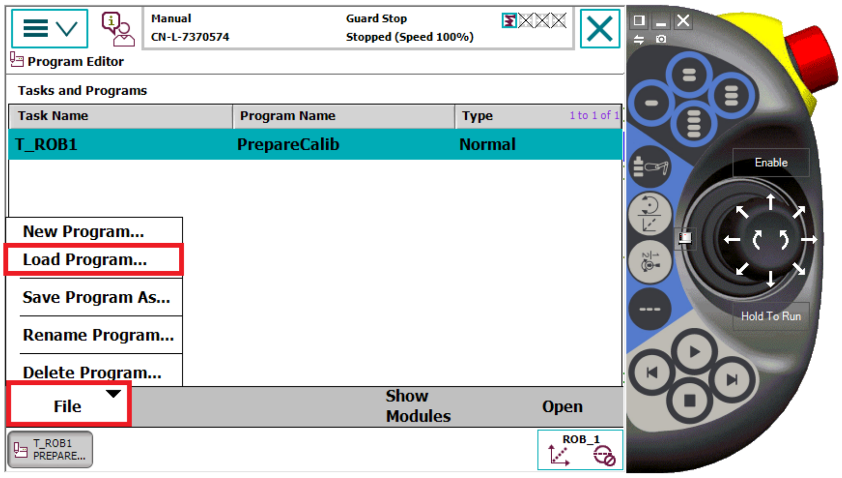
C Click **Tasks and Programs**.



xx2200001927

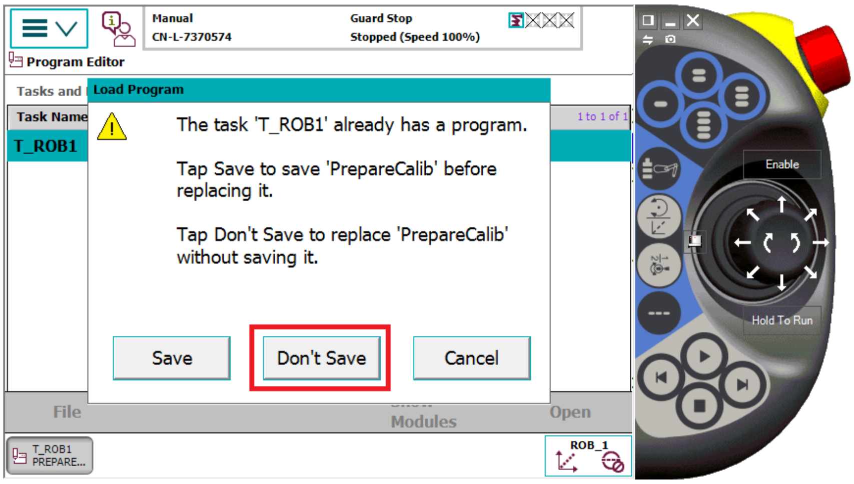
Continues on next page

D Click **File** and **Load Program**.



xx2200001928

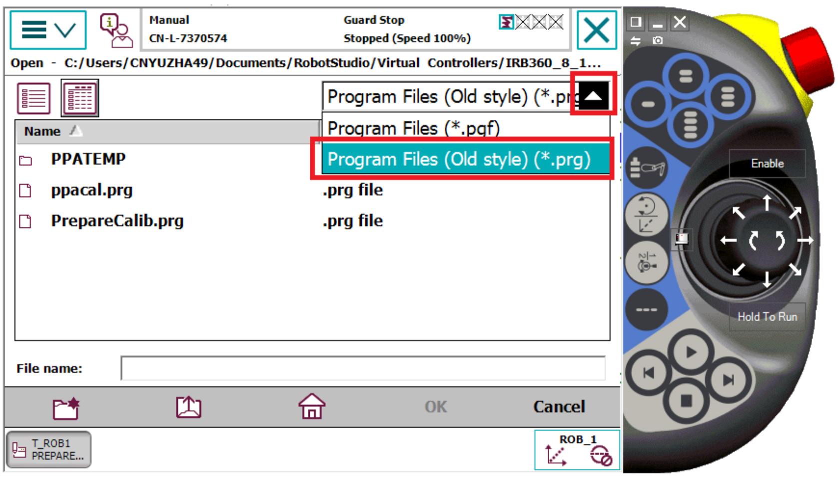
E Click **Don't Save** in the popped up dialog.



xx2200001929

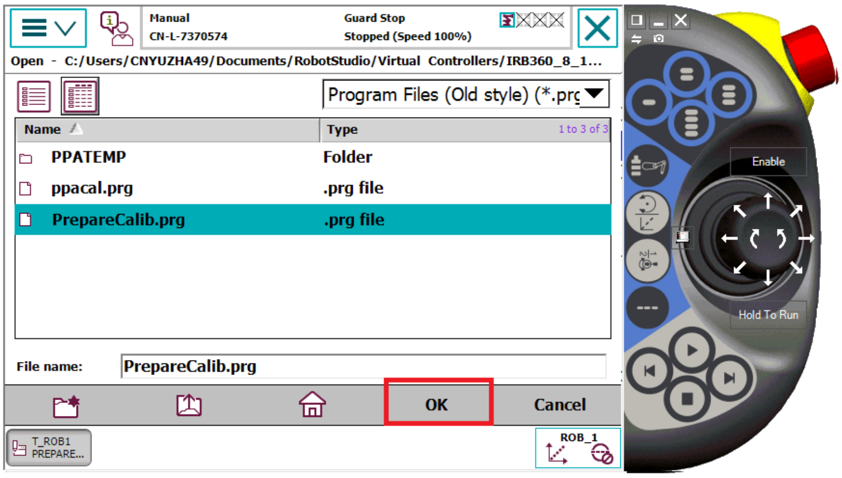
Continues on next page

F Click **Program Files (Old style)(.prg)** on the right upper corner drop-down list.



xx2200001930

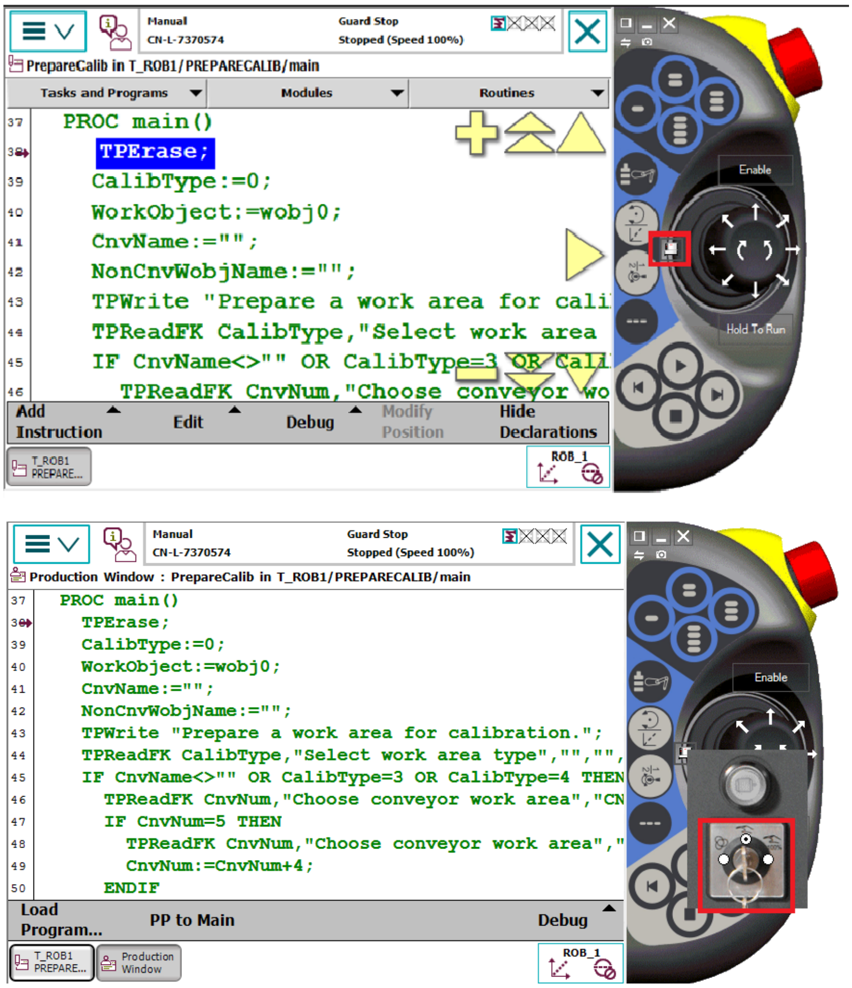
G Select **PrepareCalib.prg** and click **OK**.



xx2200001931

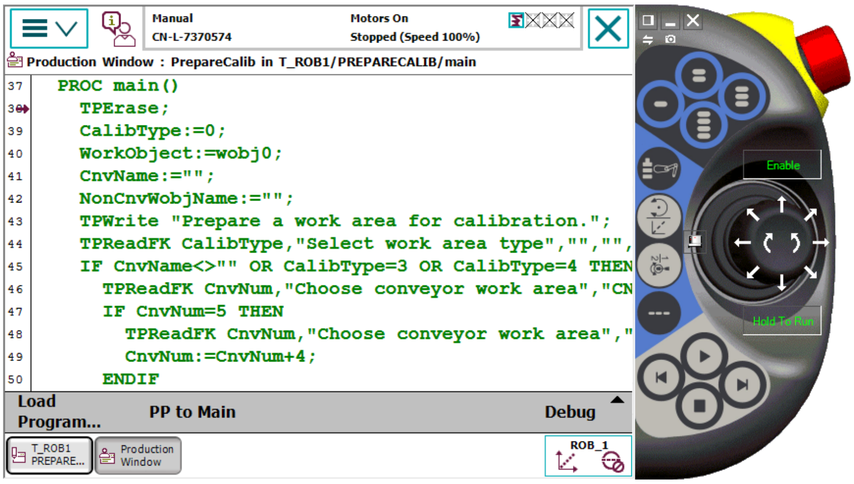
Continues on next page

H Set the controller to **Manual** mode.



xx2200001932

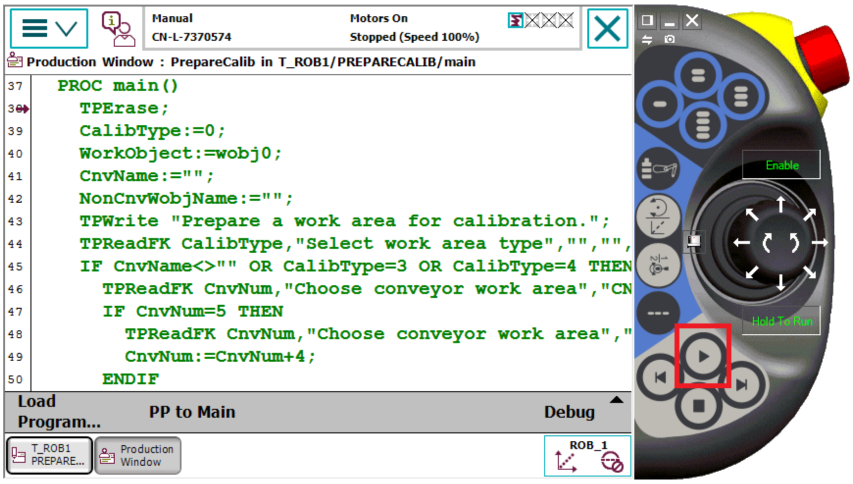
I Enable the Thumb button to motors on the controller.



xx2200001933

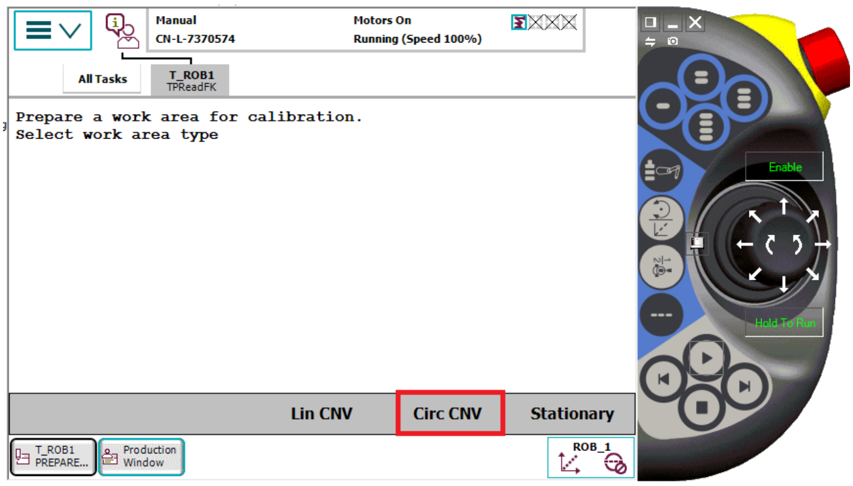
Continues on next page

J Click **Play**.



xx2200001934

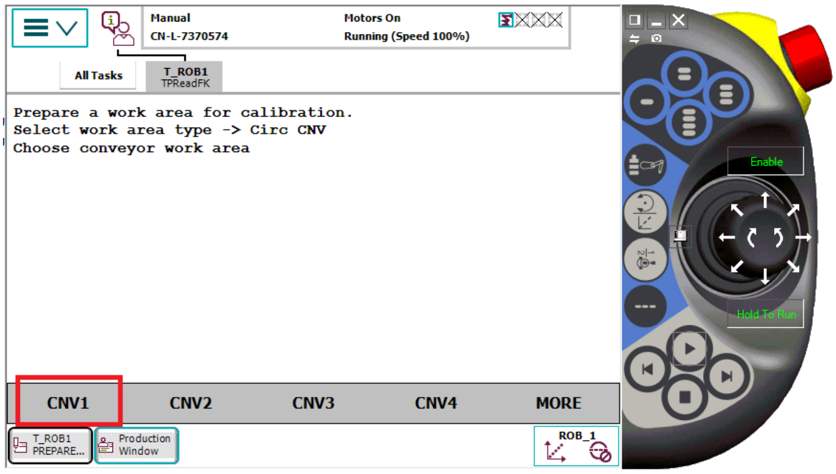
K Select the work area type **Circ CNV**.



xx2200001935

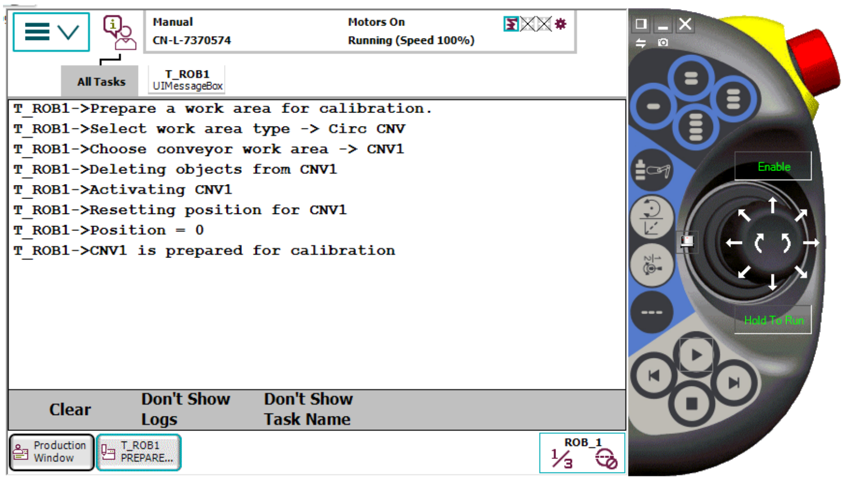
Continues on next page

L Select conveyor: for example, **CNV1**.



xx2200001936

M Wait for the message **...is prepared for calibration.** The conveyor position in the jogging window for CNV1 should now be displayed as "0" mm.



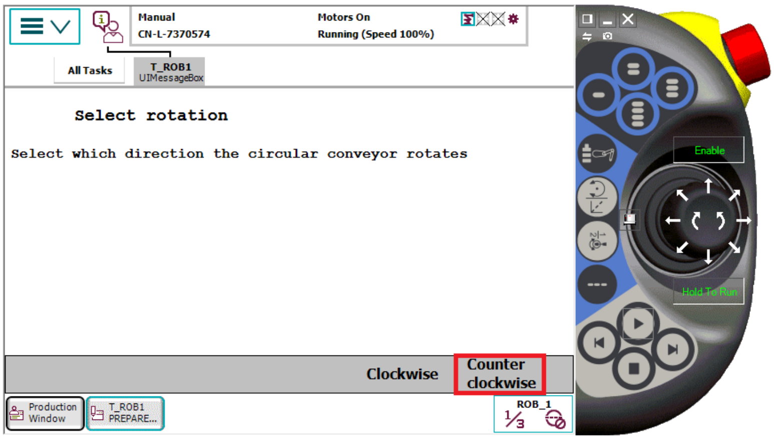
xx2200001937

Continues on next page

5 Do the following for the robot having work areas that needs to be calibrated along the conveyor:

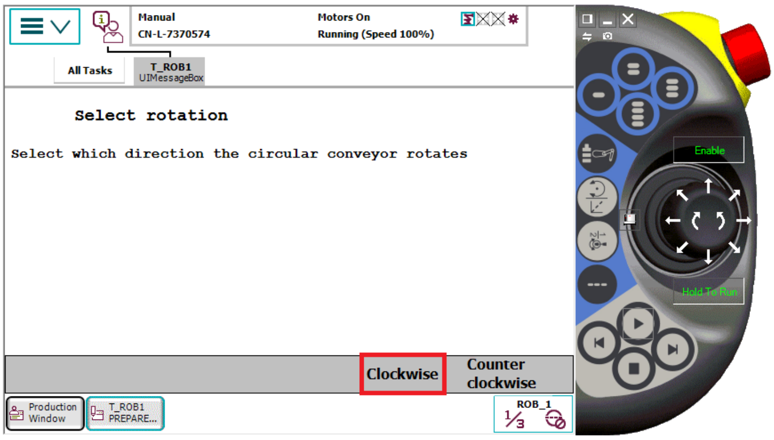
A Wait for the message **Select rotation** and select the direction of the conveyor.

- For counter clockwise moving direction, select **Counter clockwise**.



xx2200001938

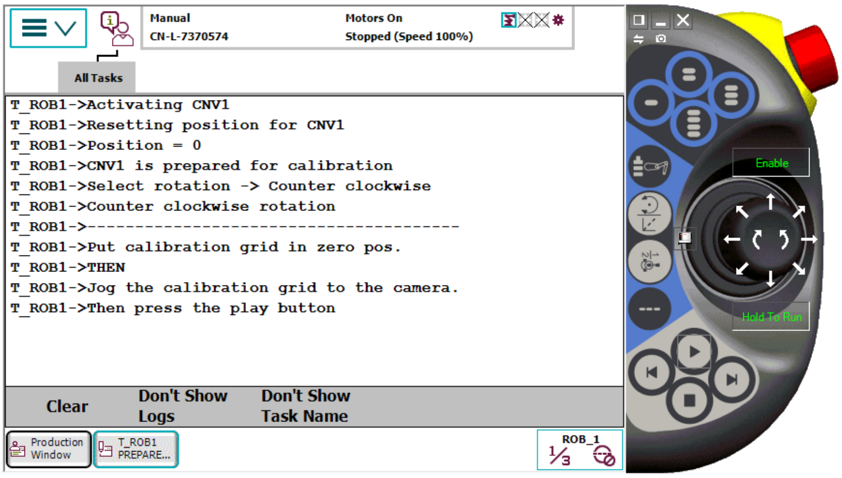
- For clockwise moving direction, select **Clockwise**.



xx2400000743

Continues on next page

B The program will continue automatically.



xx2200001939

C Rotate the belt to position B (reference point), which make the calibration grid under the camera(d) (zero position).

If selected **Counter clockwise**, rotate the belt in counter clockwise direction.

If selected **Clockwise**, rotate the belt in clockwise direction.

Tip

Rotate the conveyor belt by hand to the reference point.

- If source type is Predefined and trigger type as Distance, the reference point is origin of the Hotspot.
- If source type is Predefined and Trigger as I/O sensor, Hotspot is used to generate predefined items or containers, the reference point should be at the point where the objects are detected by the sensor. This point becomes the local origin of the detected items or containers.
- If source type is vision, the reference point is the local origin of the camera view.

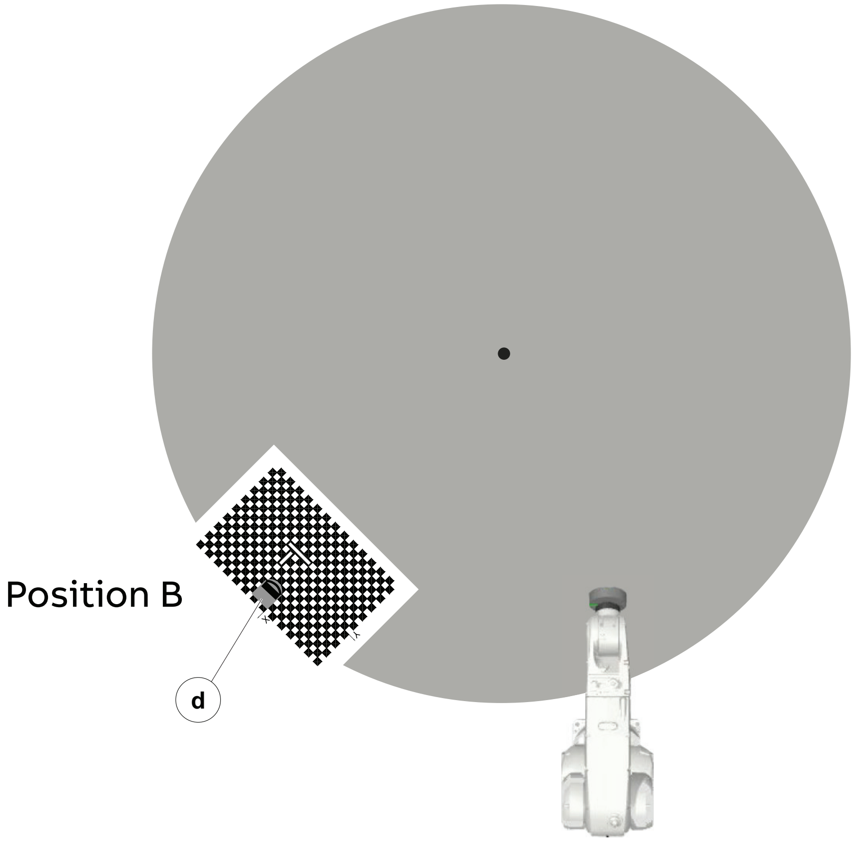
  If a camera is used, calibrate the camera at the same time, see *Calibrating camera on page 273*.

  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.

Note

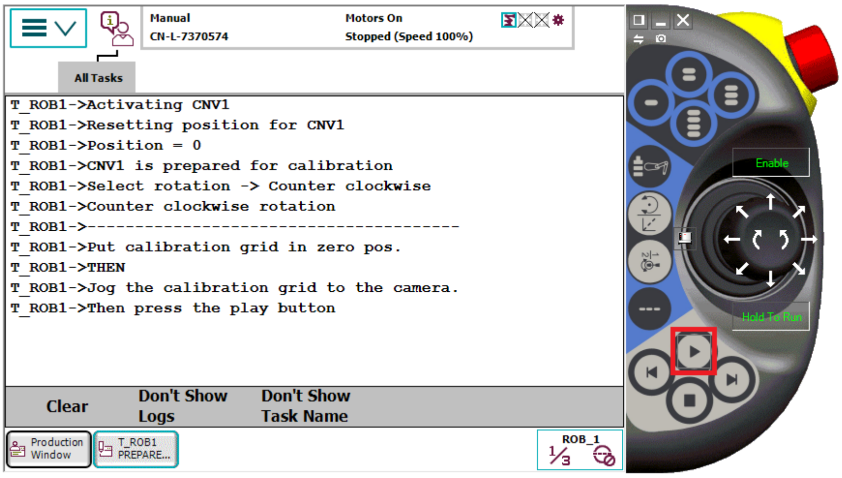
Do not rotate the belt back and forth. This will result in inaccurate calibration data.

Continues on next page



xx2200002008

D Click **Play** on the FlexPendant of the robot(s) which have been reset.



xx2200001940

E 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 three calibration points (Point 1-3) are preferred since this increases the accuracy of the calibration.

If selected **Counter clockwise**, rotate the belt in counter clockwise direction.

Continues on next page

If selected **Clockwise**, rotate the belt in clockwise direction.

> **Note**
>
> Do not rotate the belt back and forth. This will result in inaccurate calibration data.

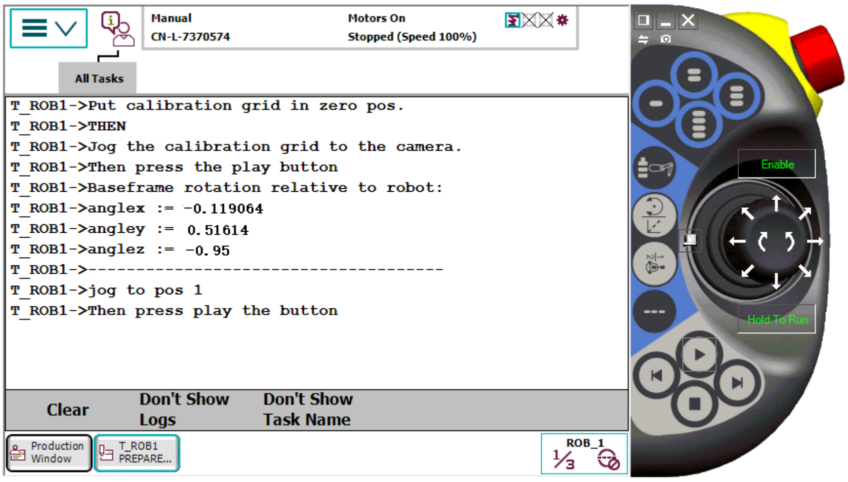
F Jog or move the robot by hand. Point out the reference point on the conveyor accurately with the calibration tool TCP.

G Modify the point (**Pos 1**) by tapping **Play**.

H Repeat the steps for the points **Pos 2** and **Pos 3**.

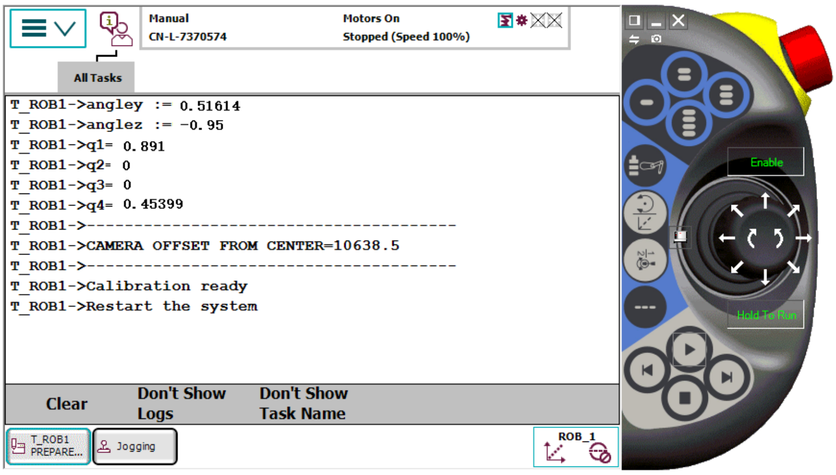
I Check if the displayed mean error and max error of the base frame calculation is acceptable. If the estimated error is acceptable, restart the system to confirm and store the new base frame.

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



xx2200001941

Continues on next page



xx2200001942

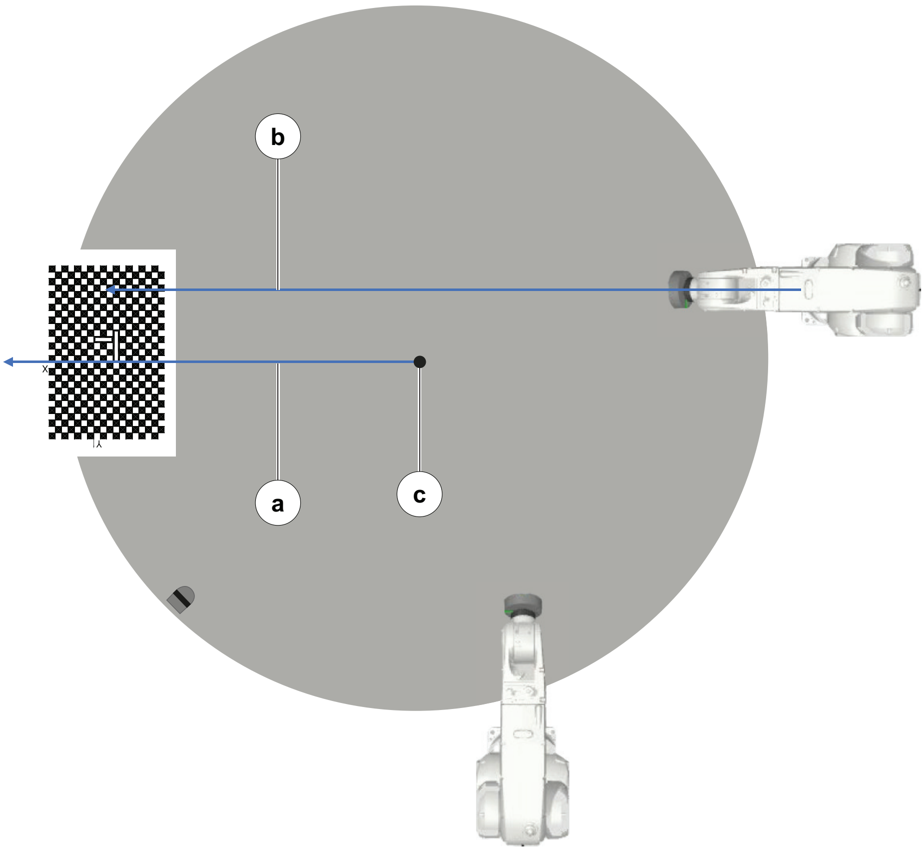
If the estimated error is not ok, this base frame must be re-calibrated.

Tip

Read the value of **CAMERA OFFSET FROM CENTER**. This value will be used as the input of `Sensor offset` in *Type configuration for circular conveyor on page 255*.

Continues on next page

6 If there are more robots need to be calibrating, repeat from step 3 to step 5 for each robot.



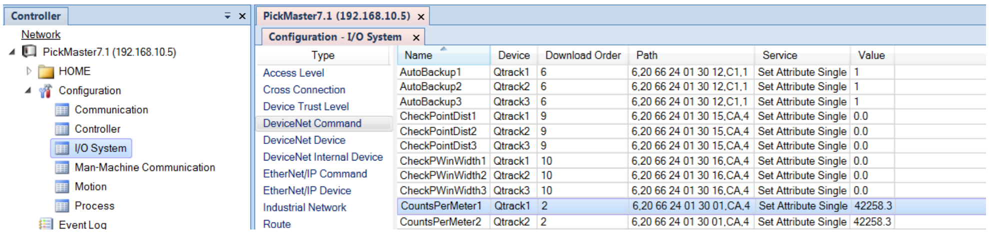
xx2400000405

4.4.5.2 Calibrating circular conveyor with DSQC 377

4.4.5.2.1 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 *DeviceNet Command*, in the topic *I/O System*.



xx1900000551

Calculation for Counts Per Meter

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

`(position1*old_counts_per_meter)/measured_radians`

Value	Description
`position1`	Read from FlexPendant Jogging window.
`old_counts_per_meter`	The encoder's old value. **Note** The encoders delivered from factory have a preset value. For an IRC5 system this value is 20,000. This value can be used to start the calibration with.
`measured_radians`(radian)	The manually measured radians 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. Rotate the conveyor belt approximately 180 degrees.
4. In the FlexPendant **Jogging** window, read the position of the conveyor. This is `position1`.
5. Measure the physical radians between the two marks. This is the value `measured_radians`.
6. Calculate *Counts Per Meter* using the read and measured values.

When this variable is applied to a circular conveyor, the actual meaning is *counts per radian*.

Continues on next page

For example: `(1.5*20000)/0.5 = 60000`

7. In RobotStudio, click **Configuration** and select topic **I/O System** and type **DeviceNet 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.

Related information

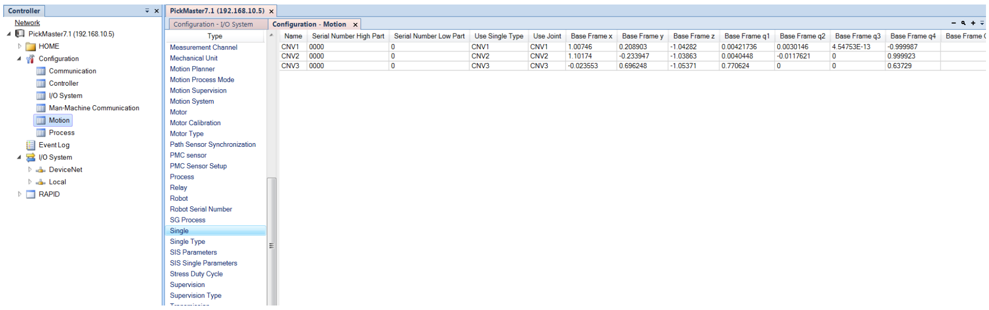
Application manual - Conveyor tracking.

Technical reference manual - System parameters.

4.4.5.2.2 Defining the base frame

Introduction

For each conveyor work area on a circular 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.



xx1900000592

Preparations

- Define the `Counts Per Meter` system parameter for each conveyor work area. For more details, see *Defining the parameter Counts Per Meter on page 249*, *Defining the parameter Counts Per Meter on page 221*.
- 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, see *Calibrating camera on page 273*. After calibrating the camera, keep the camera calibration pattern attached to the conveyor.

Recommendation

This section describes how to use TCP measurements and RAPID programs to calculate the conveyor base frame position and quaternion for a circular conveyor.

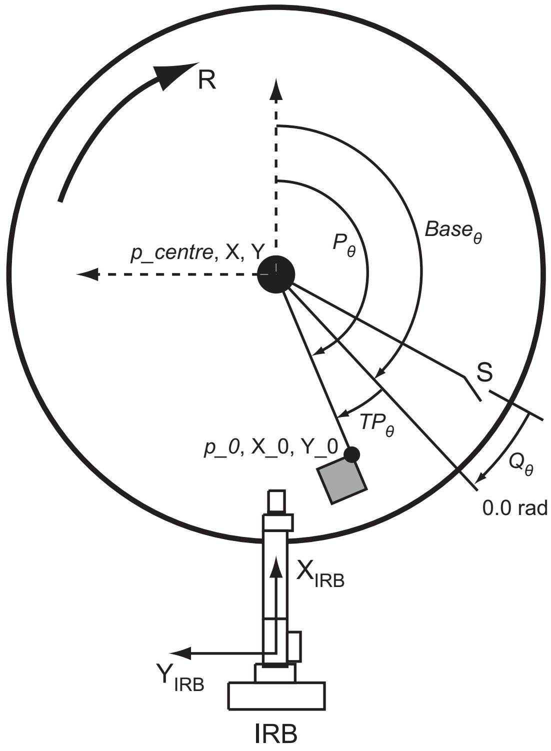
This method uses three measured points on the circular conveyor to calculate the center of rotation. The three points should be spaced as far apart as possible around the periphery.

Continues on next page

Defining the base frame orientation and start window start calibration

The base frame quaternion defines where the 0.0 rad point is for the robot motion.

The following figure shows an example of the angles that are used when defining the base frame orientation for the circular conveyor.



xx1200001103

R	Direction of rotation
S	Synchronization switch
Q_θ	Queue tracking distance angle
TP_θ	Angle shown on FlexPendant
P_θ	Angle calculated from p_0 position
$Base_\theta$	Base frame angle to be converted to a quaternion

Calculating the x and y positions for the base frame

Use this procedure to calculate the x and y positions for the base frame.

1. Use `Wobj0` on the FlexPendant. Pick out a reference point on the circular conveyor, jog the TCP to this point and record *p_0*.
2. Run the conveyor to another position. Jog the TCP to the reference point and record *p_1*.
3. Run the conveyor to a third position, jog the TCP to the reference point and record *p_2*.
4. Use the function `CNVUTL_cirCntr` with the points *p_0*, *p_1*, and *p_2*, to calculate the center of the circle, *p_centre*.

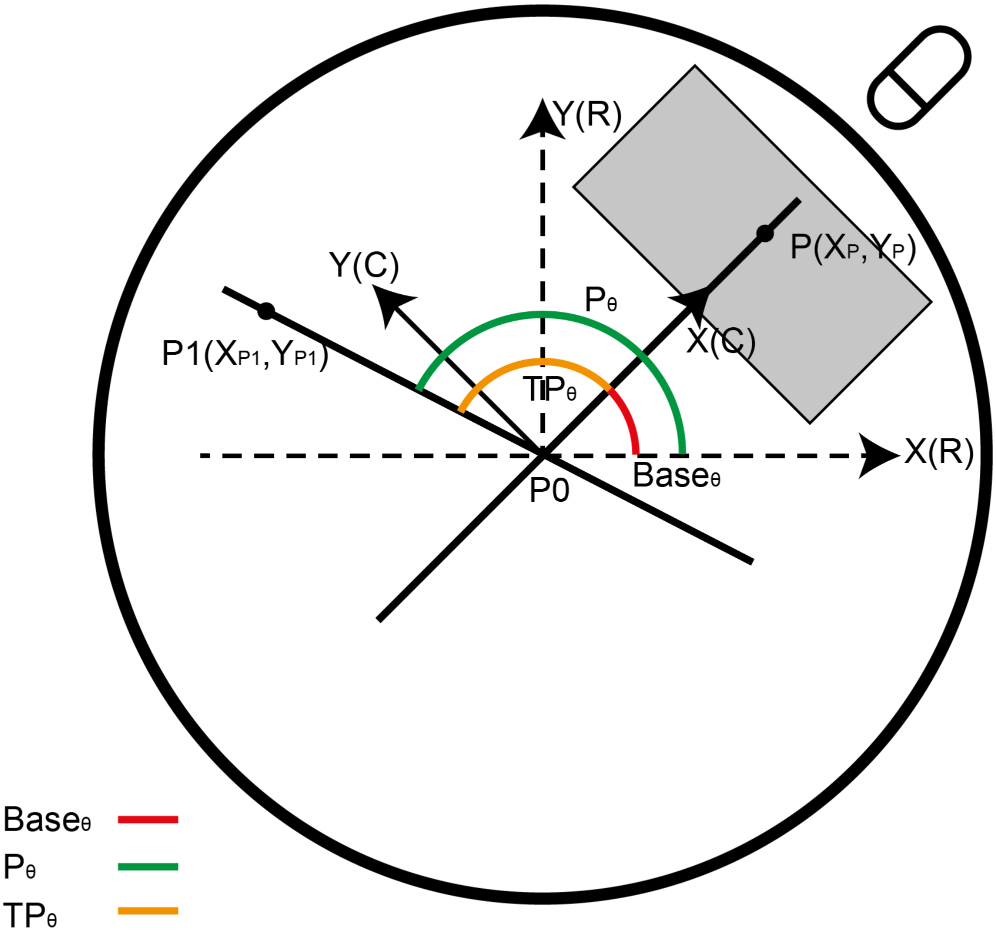
   The system module `cnv_utl.sys` can be found in Robotware.

Continues on next page

5 Take the x and y values from *p_centre* and enter them into the base frame values for the conveyor, converting to meters, see *Application manual - Conveyor tracking*. The z value will be entered later, once the work object zero position has been chosen.

Calculating the quaternion

Use this procedure to calculate the quaternion for the base frame orientation.



xx1900000678

1 With the recorded angle in step 5 when calculating the x and y positions for the base frame. This is angle TP_{θ}, see example measurement points in *Defining the base frame orientation and start window start calibration on page 252*.

2 Calculate Pθ from the *XP1* and *YP1* coordinates of *P0* and the atan function.
If the point is at first quartile or frouth quartile: Pθ = arctan(YP1/XP1)
If the point is at second quartile or third quartile: Pθ = π + arctan(YP1/XP1)

Tip

If the calculation tool provide the arctan2 function, there is no need to judge the quartile and use Pθ = arctan2 (XP1, YP1) directly.

Continues on next page

3 Calculate the value of Base.

$$Base_{\theta} = P_{\theta} - TP_{\theta}$$

4 Calculate the quaternion for the base frame taking into account the direction of rotation:

Counter clockwise rotation:

```
q1 = cos(Base θ /2)
q2 = 0.0
q3 = 0.0
q4 = sin(Base θ /2)
```

Clockwise rotation:

```
q1 = 0.0
q2 = cos(Base θ /2)
q3 = –sin(Base θ /2)
q4 = 0.0
```

5 Enter the value for z (in meters) from *p_0*, and the values for the quaternions, *q1*, *q2*, *q3*, and *q4*, into the base frame for the conveyor.

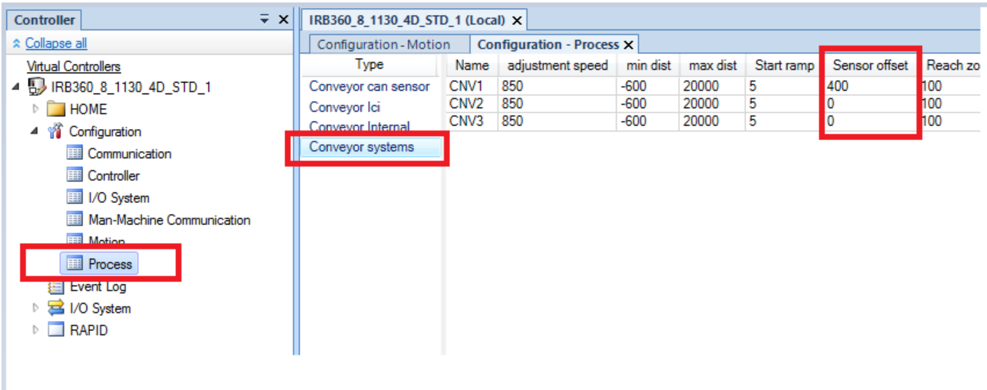
4.4.5.3 Type configuration for circular conveyor

Introduction

For each conveyor work area on a circular conveyor, the type parameters, `Sensor offset`, `Reach zone accuracy`, `Mechanics` and `Rotating Move`, must be set.

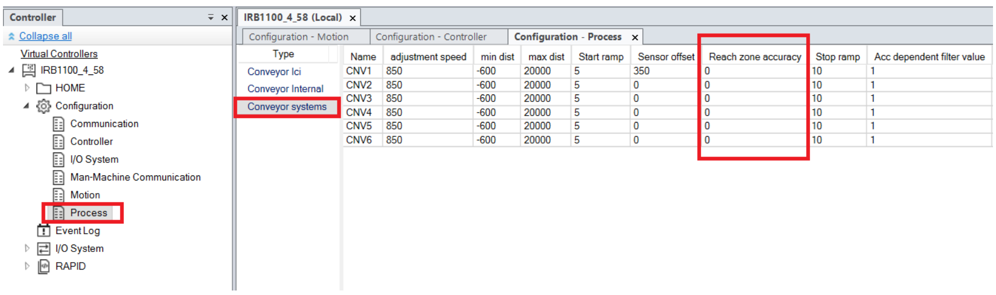
`Sensor offset` defines the distance between the sensor and the conveyor base frame original point. For example, when using a camera, this parameter represents the distance of the projection point of the camera on the conveyor belt from the center of the circle.

> **Note**
>
> For DSQC 377, the distance for `Sensor offset` is measured manually.
>
> For DSQC 2000, the distance for `Sensor offset` is read from the program result. See the value for IRC5 *on page 247* and for OmniCore *on page 234*.



xx2100000066

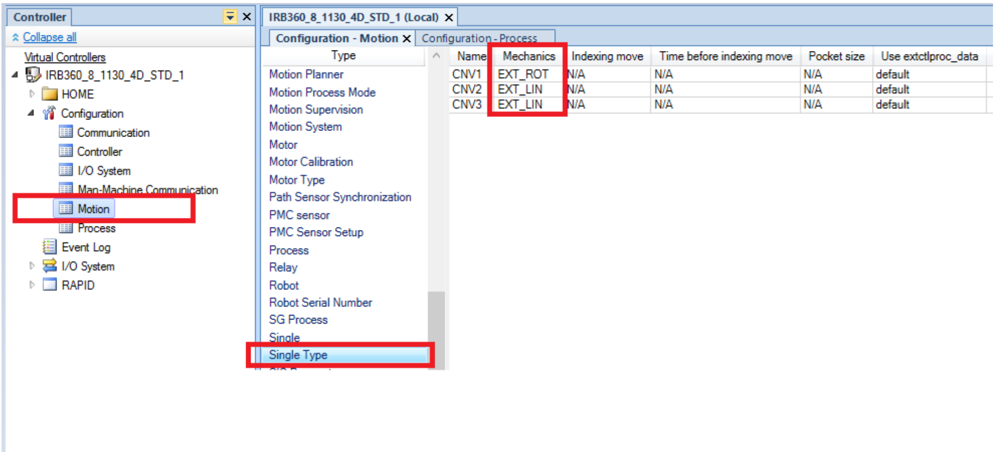
`Reach zone accuracy` defines the exact angle accuracy between the tool_0 and robot target point. The value can be set between 0-100. When the value is bigger, for example 100, that means the robot gripping pose need to be 100 percentage match with the calculated item target point, otherwise the robot cannot grip the item. So when the circular conveyor is used and rotate in clockwise, this parameter must be set as 0 to ignore the influence of the the conveyor base frame' flipping.



xx2400000407

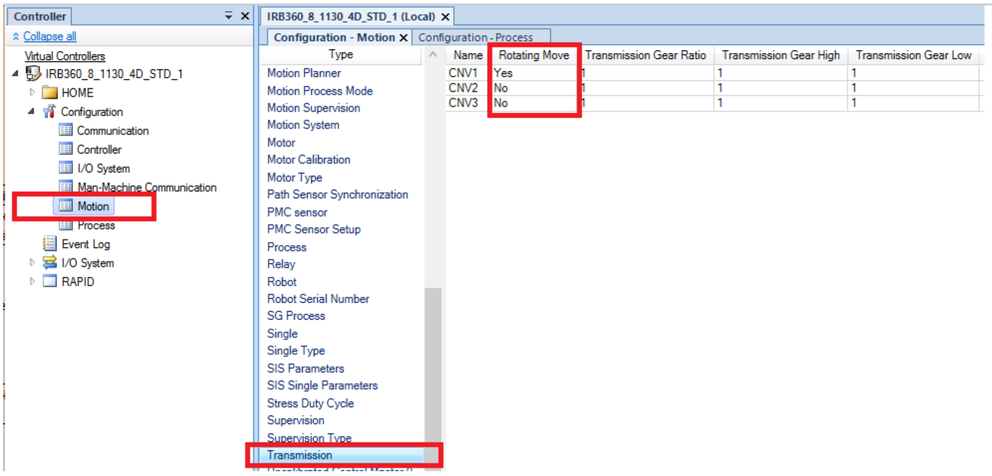
Continues on next page

`Mechanics` defines the moving trajectory of the conveyor. The default value is `EXT_LIN` (linear conveyor). So when the circular conveyor is used, this parameter must be set as `EXT_ROT`.



xx2100000067

`Rotating Move` defines the conveyor's rotating status. The default value is `No` (linear conveyor). So when the circular conveyor is used, this parameter must be set as `Yes`.



xx2100000068

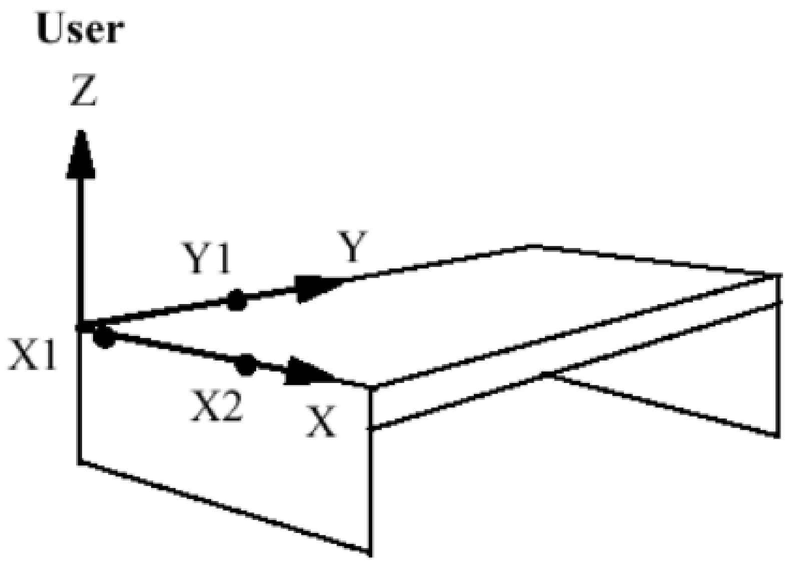
4.4.6 Calibrating indexed work area

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, see *Calibrating camera on page 273*. 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

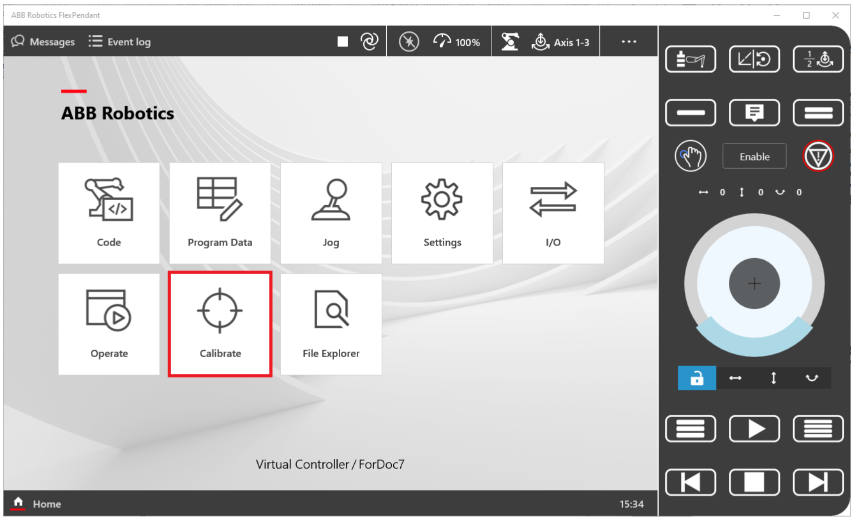
Procedure(OmniCore)

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.

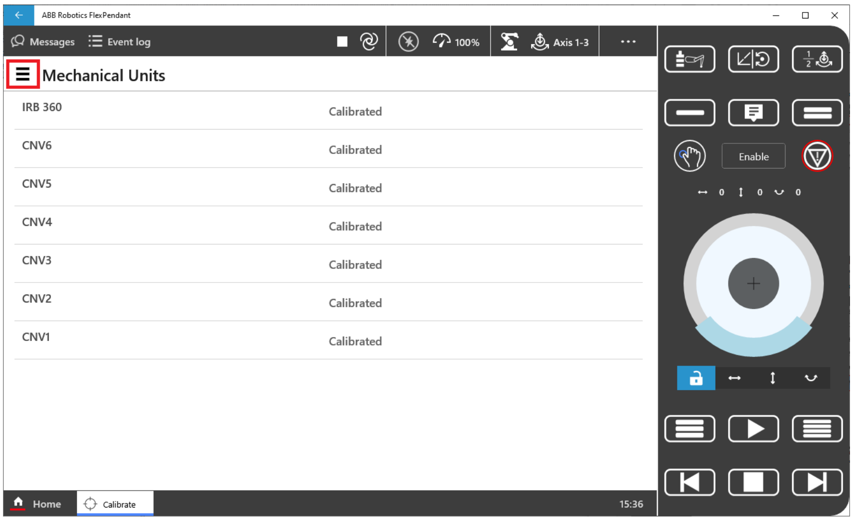
   Do the following for all the robots having work areas that needs to be calibrated along the conveyor:
   - In the FlexPendant, click **Calibrate**.



xx2100000362

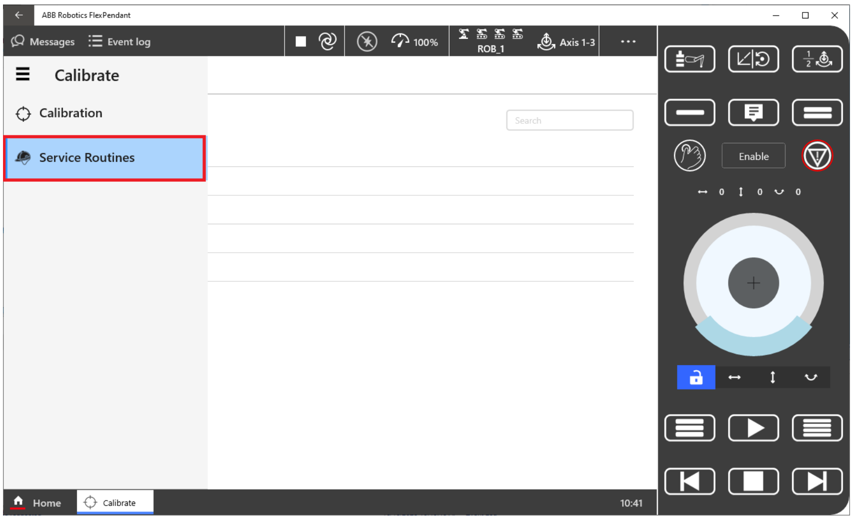
Continues on next page

- Click **Option Tab** on the up left corner.



xx2100000363

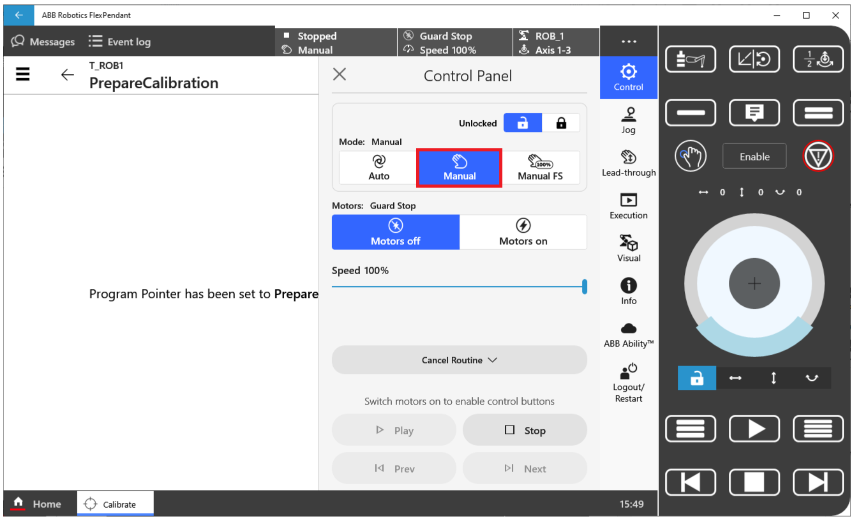
- Click **Service Routines.**



xx2100000364

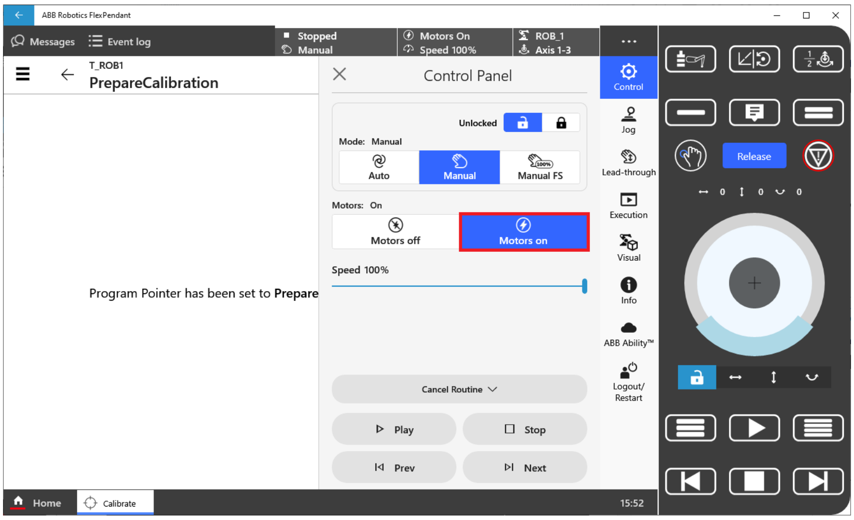
Continues on next page

- Set the controller to **Manual** mode.



xx2100000368

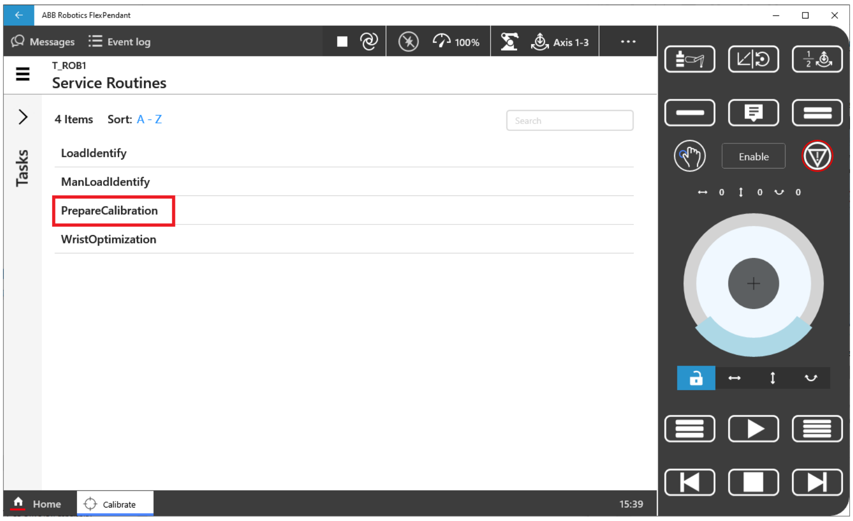
- Enable the Thumb button to motors on the controller.



xx2100000369

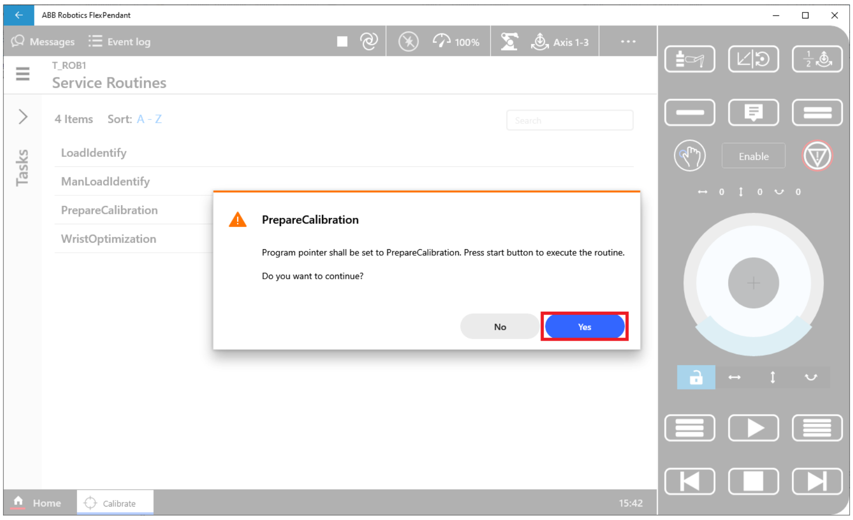
Continues on next page

- Click **PrepareCalibration**.



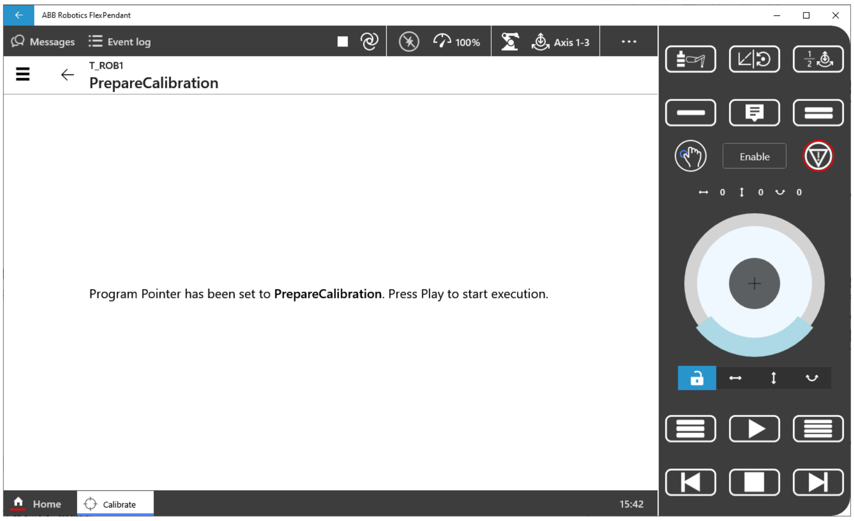
xx2100000365

- Click **Yes** in the popped up dialog.



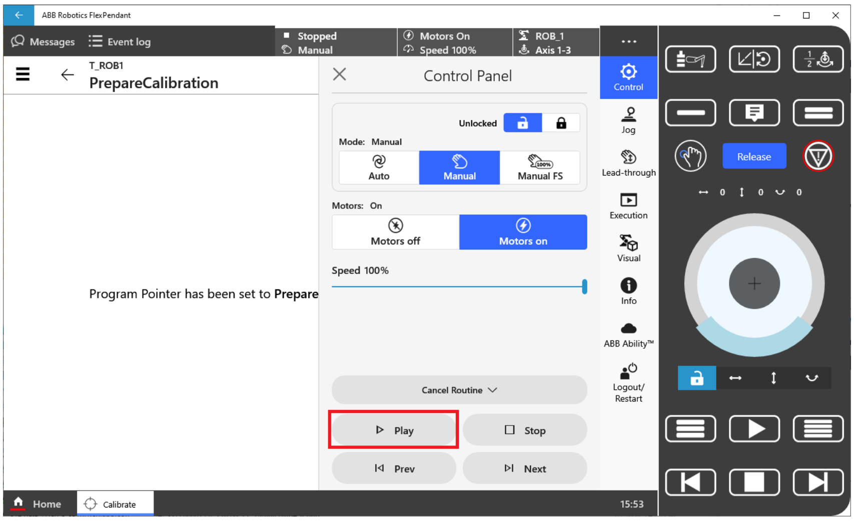
xx2100000366

Continues on next page



xx2100000367

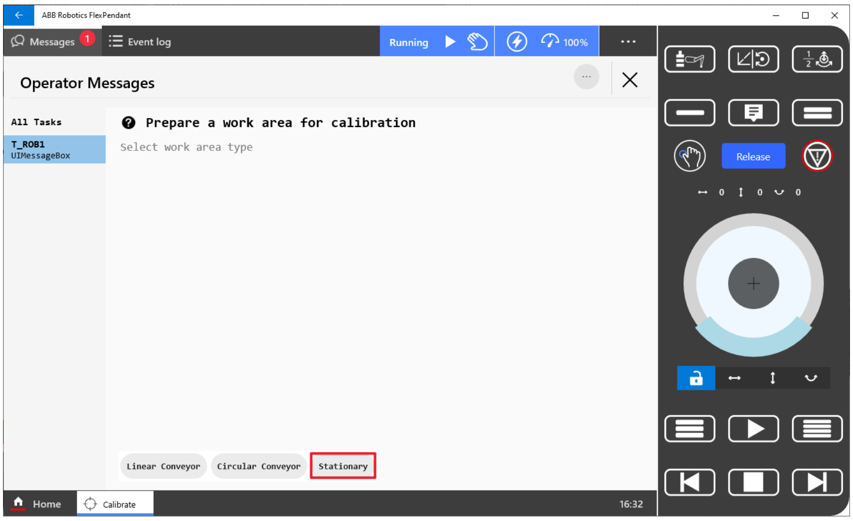
- Click **Play**.



xx2100000370

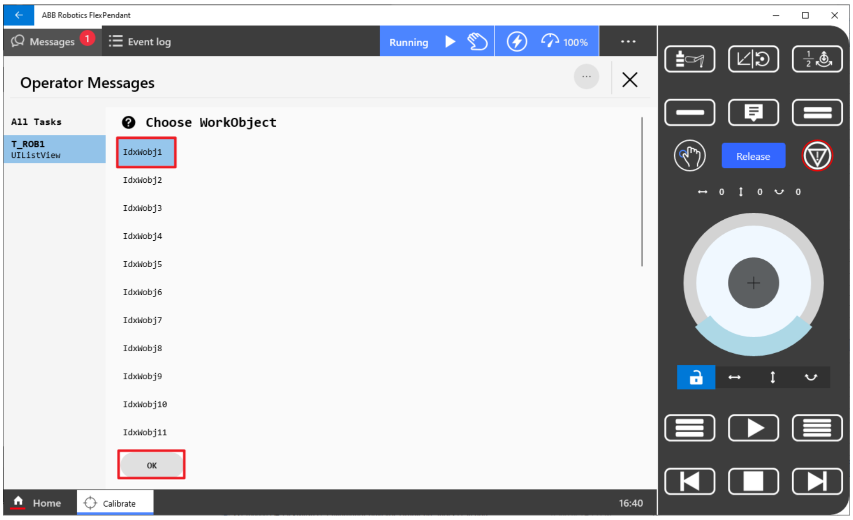
Continues on next page

- Select the work area type **Stationary**.



xx2100000396

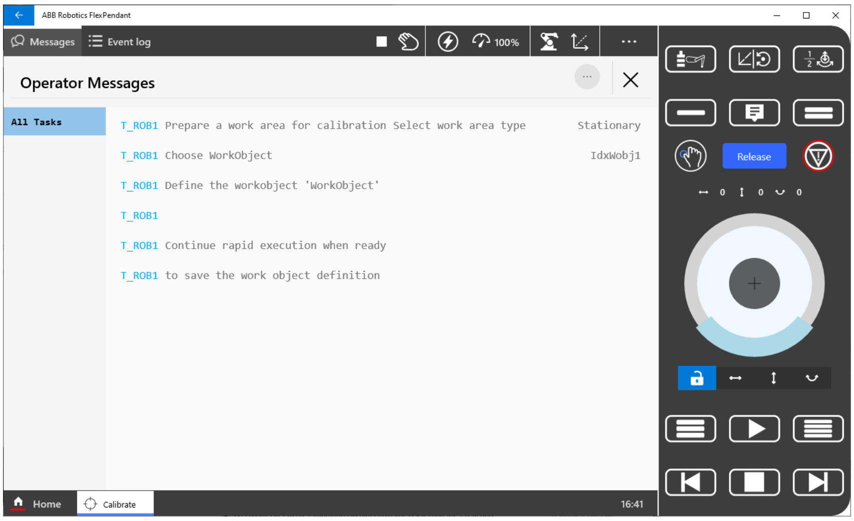
- Select conveyor: for example, Idxwobj1. Then click **OK**



xx2100000397

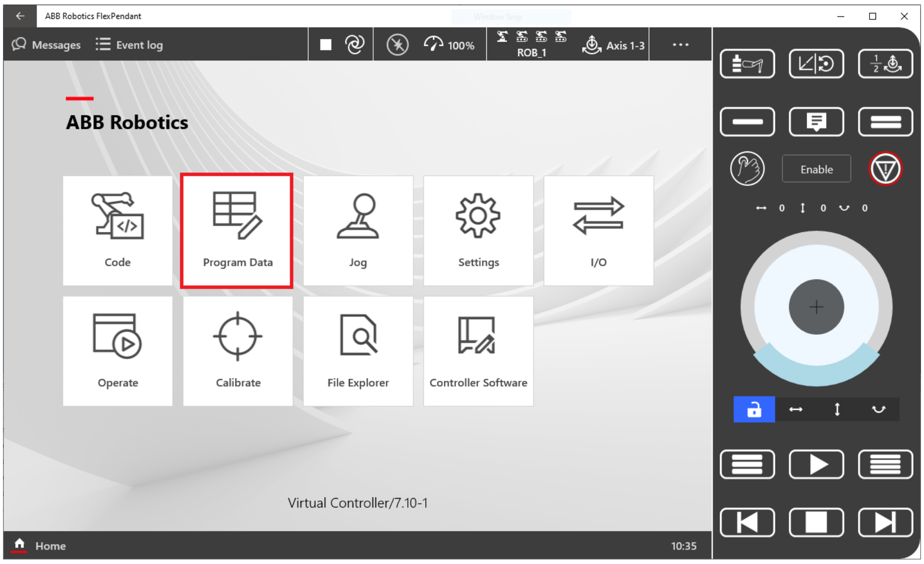
Continues on next page

- Wait for the message **...is prepared for calibration.** The conveyor position in the jogging window for CNV1 should now be displayed as “0” mm.



xx2100000398

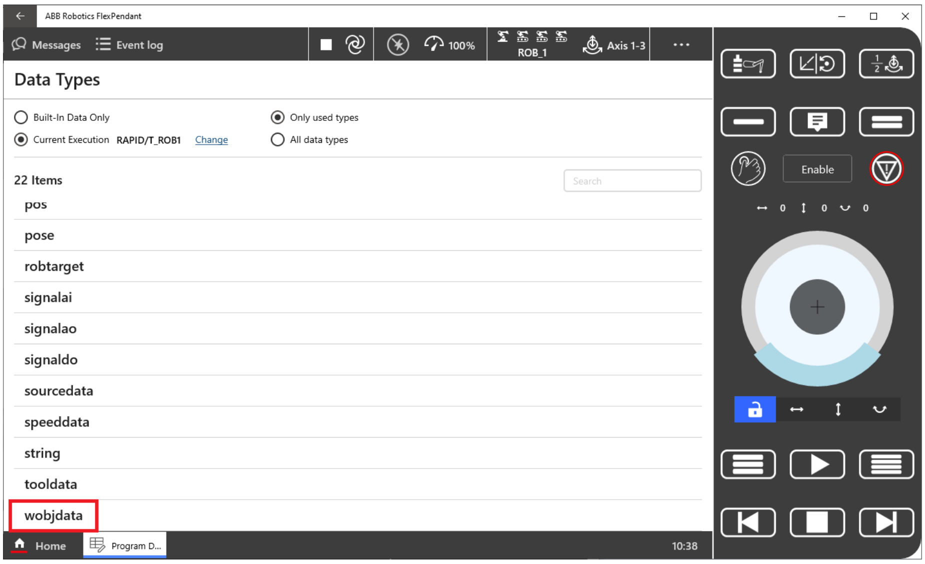
3 Return to the home page and select **Program Data**.



xx2300001587

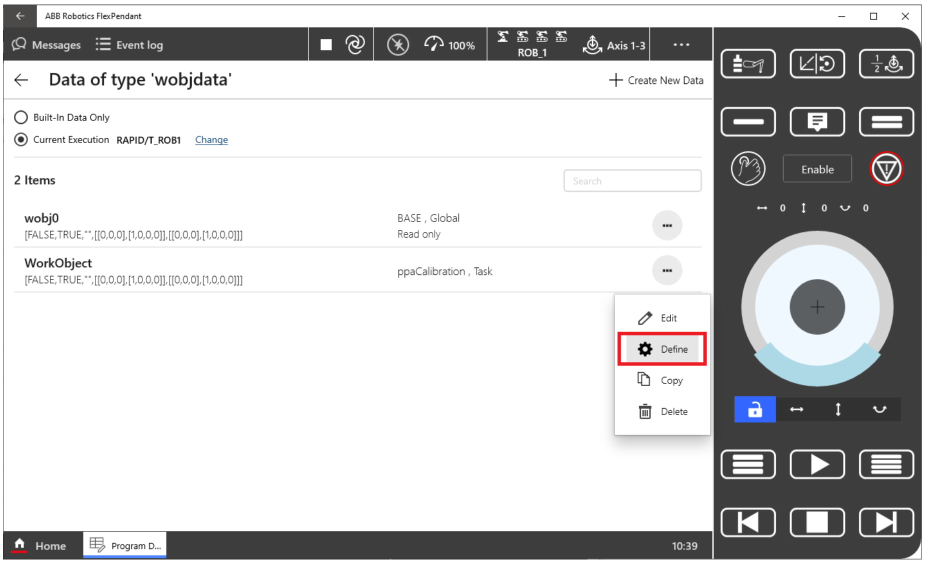
Continues on next page

4 Select **Workobject** in **Data Type**.



xx2100000380

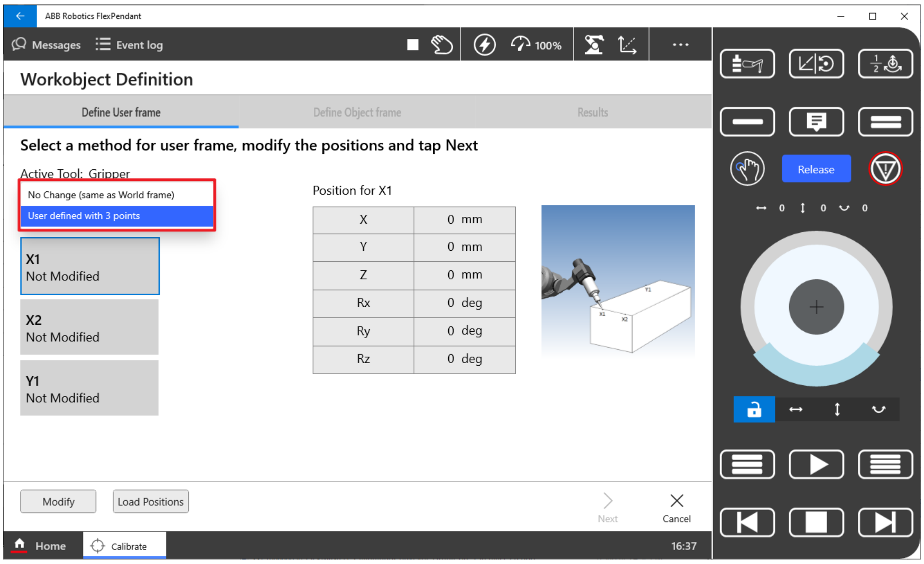
5 In the **Workobject**, tap on the **...** to select **Define**.



xx2100000381

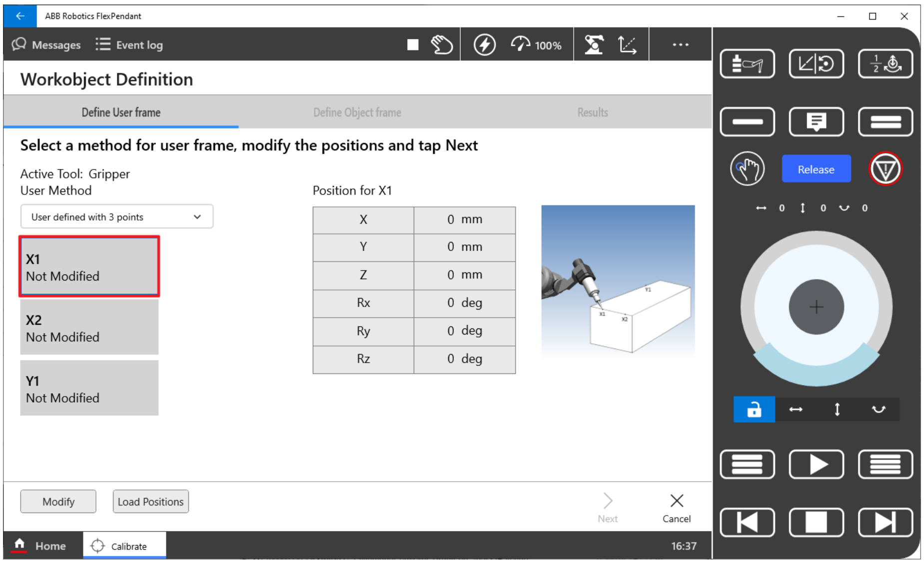
Continues on next page

6 In the **Define User frame** window, set the **User Method** as `User defined with 3 points`.



xx2100000382

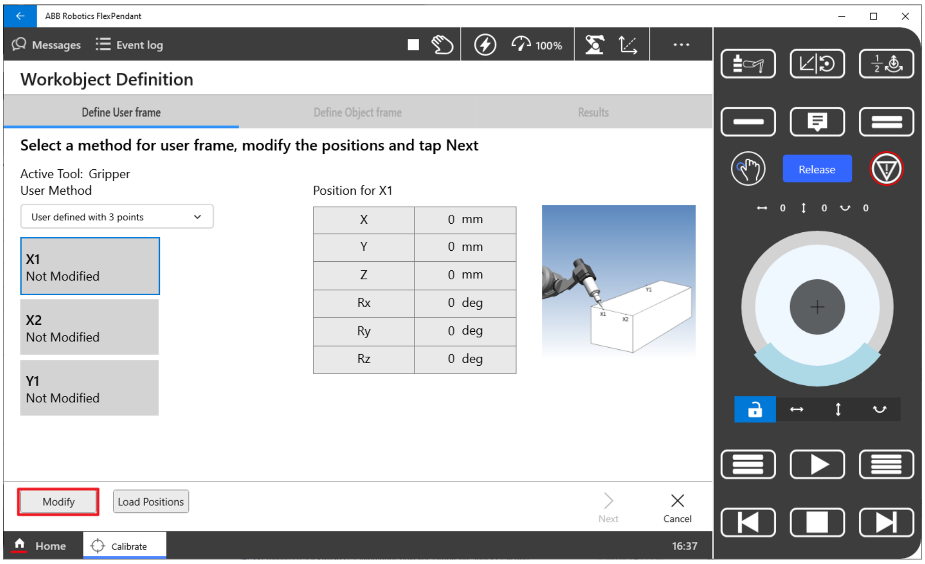
7 Select **X1**. Point out a location on the x-axis close to the origin with the robot's TCP.



xx2100000383

Continues on next page

8 Press **Modify**.



xx2100000384

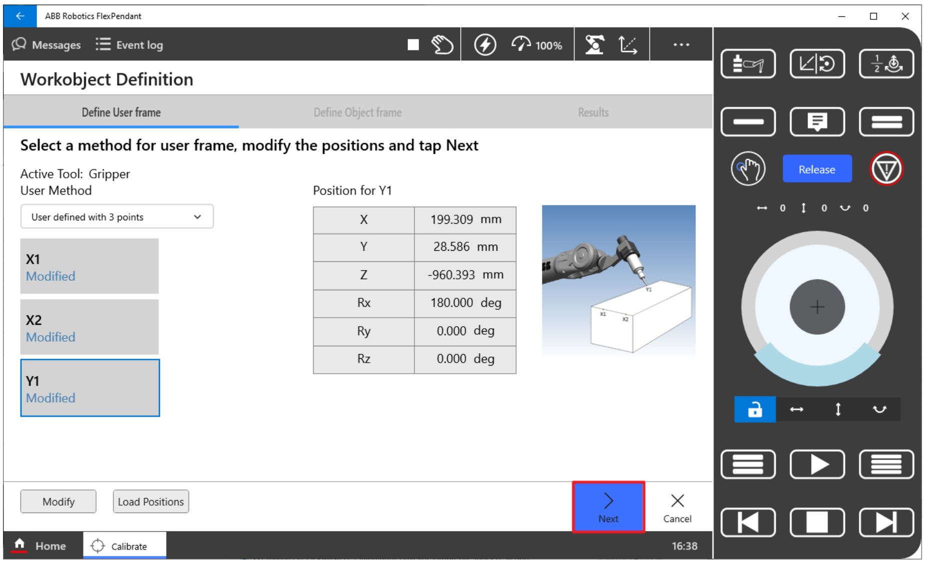
9 Select **X2**. Move the TCP a distance in the direction the x-axis. Point out a location on the x-axis with the robot's TCP.

10 Press **Modify**.

11 Select **Y1**. Point out a location on the positive y-axis with the robot's TCP.

12 Press **Modify**.

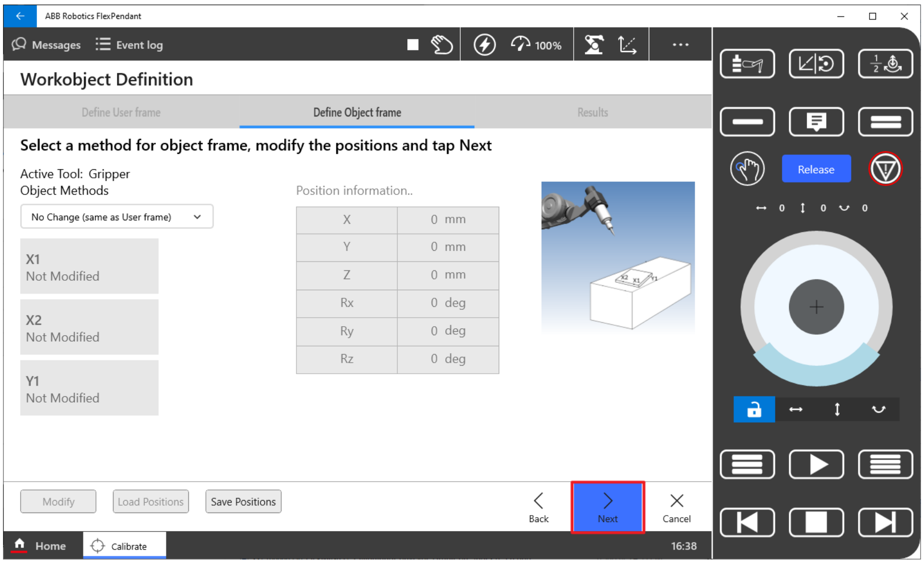
13 Tap **Next**.



xx2100000385

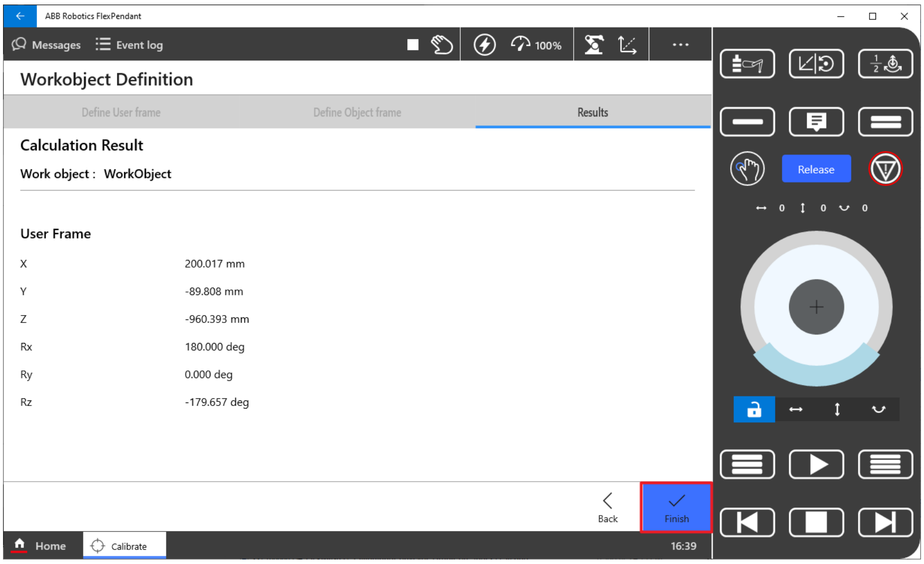
Continues on next page

14 In the **Define Object frame** window, tap **Next**.



xx2100000386

15 Check if the displayed mean error and max error of the user frame calculation is acceptable. If the estimated error is acceptable, tap **Finish** to confirm and store the new user frame.



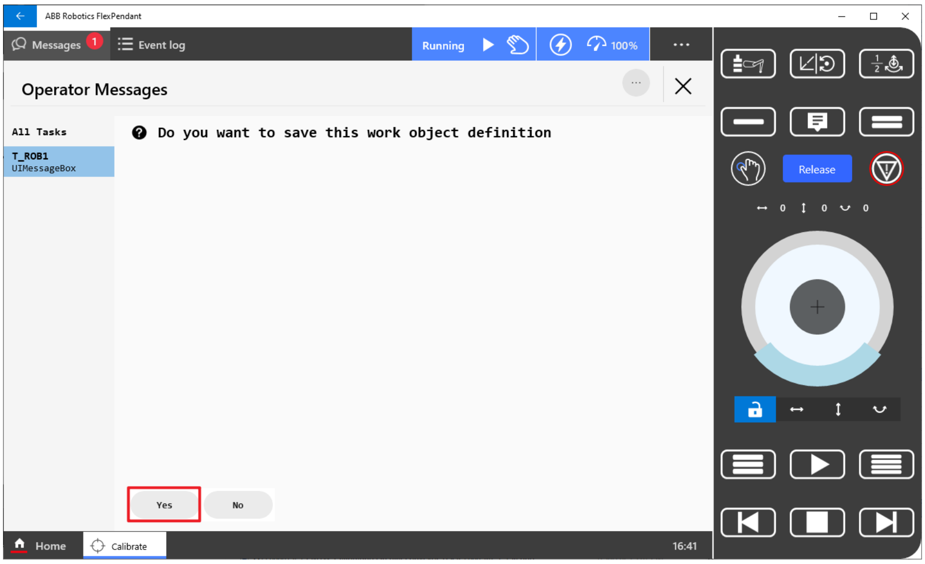
xx2100000391

16 Enable the Thumb button to motors on the controller.

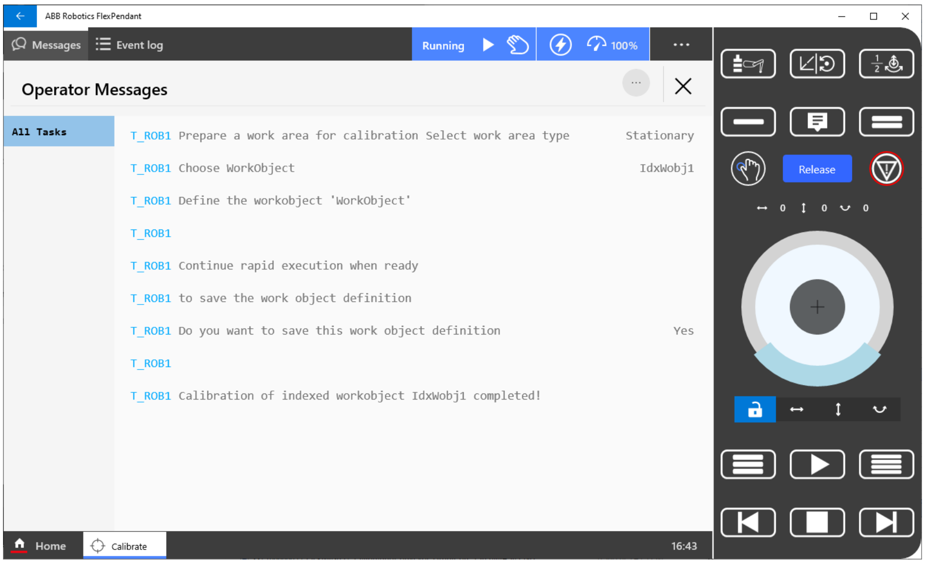
17 Click **Play**.

Continues on next page

18 Click **Yes** on the question: `Do you want to save this work object definition`.



xx2100000393



xx2100000394

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

Procedure(IRC5)

1 Select the work object to be calibrated.

- In the FlexPendant **Program Editor**, load the program `ppacal.prg`(DSQC 377)/ `PrepareCalib.prg`(DSQC 2000). If the robot is a MultiMove robot, load `ppacal.prg`(DSQC 377)/ `PrepareCalib.prg`(DSQC 2000) for this robot task (for example, `T_ROB1`), and select only this task for execution.

Continues on next page

- 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.

4.4.7 Verifying conveyor 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.

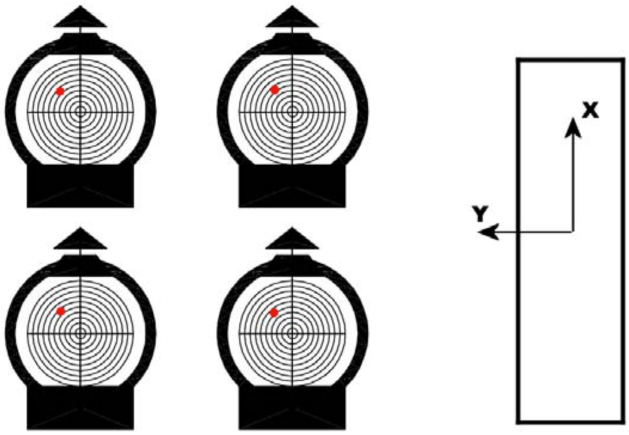
1. Place the calibration verification paper on the conveyor under the camera. The center column of object should be placed close to the center of the camera view. Align the paper with the conveyor as accurately as possible.
2. Use one of the objects on the calibration verification paper as model. See *Calibrating camera on page 273*.
3. Place the grip position in the center of the model.
4. Examine how the robot is placing the holes[3] to adjust possible errors in the camera calibration or the base frame calibration.

xx0900000649

[3] The points on the paper.

Continues on next page

If the holes are rotated[4] too much compared to the center of the objects, which affects the accuracy of the grasp, then recalibrate the cameras.



xx0900000650

If the holes are off center of the objects too much, which affects the accuracy of the grasp, then recalibrate the base frame of the conveyor.

4 The angle between the hole to the center and the X axis.

4.4.8 Calibrating 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

If any firewall or antivirus software is installed, add `pickmasteru.exe`, `sshd.exe`, and `visionclient.exe` to the white list.

Otherwise the PickMaster PowerPac cannot connect Runtime and the vision function cannot work normally.

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

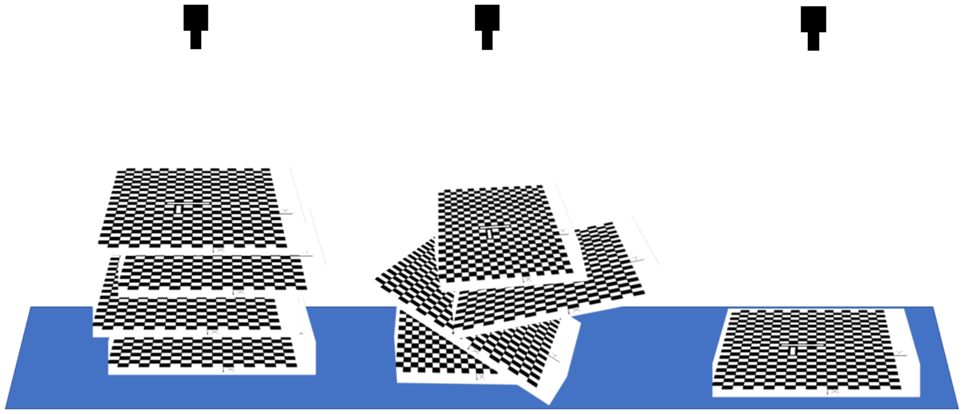
The camera can be calibrated using one or several images. The difference when using more than one image is that the camera's position in space is calculated. This space information is used both for 2.5D applications when the product height needs to be determined, and for compensating parallax errors in pure 2D applications. See *Working with products of varying height (2.5D vision) on page 357*.

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

Use at least 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



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 can be found in *C:\Program Files (x86)\ABB\PickMaster Twin 2\PickMaster Twin Runtime 2\Documents\Callibration Papers*.

The printed image must have a high contrast and the surface must not be reflective (high gloss). Make sure that the calibration paper covers the full field of view of the camera. Verify with a ruler that the squares are proportional. Enter the correct width and length of the squares. The precision must be at a minimum of a tenth of mm. To obtain an accurate value, measure the full length of the calibration sheet in each direction and divide by the number of squares.

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 positive motion direction of the conveyor.

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 tree view **Cameras** and select **Calibration**.

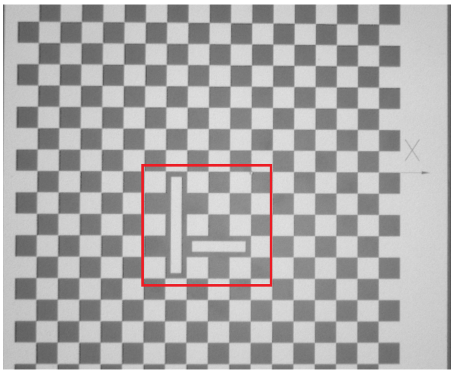
   The **Camera Calibration** window is opened.

Continues on next page

2 Place the verification paper on the conveyor under the camera. Align the paper with the conveyor as accurately as possible.

Tip

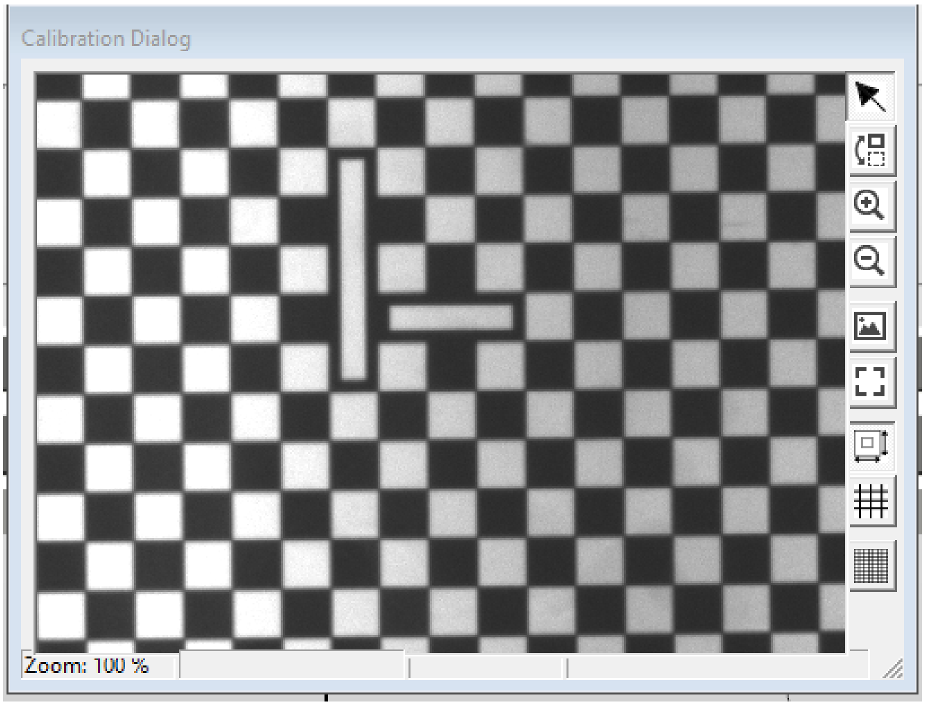
All images used for calibration should be taken with the central part of the calibration board always in the image.



xx2300001516

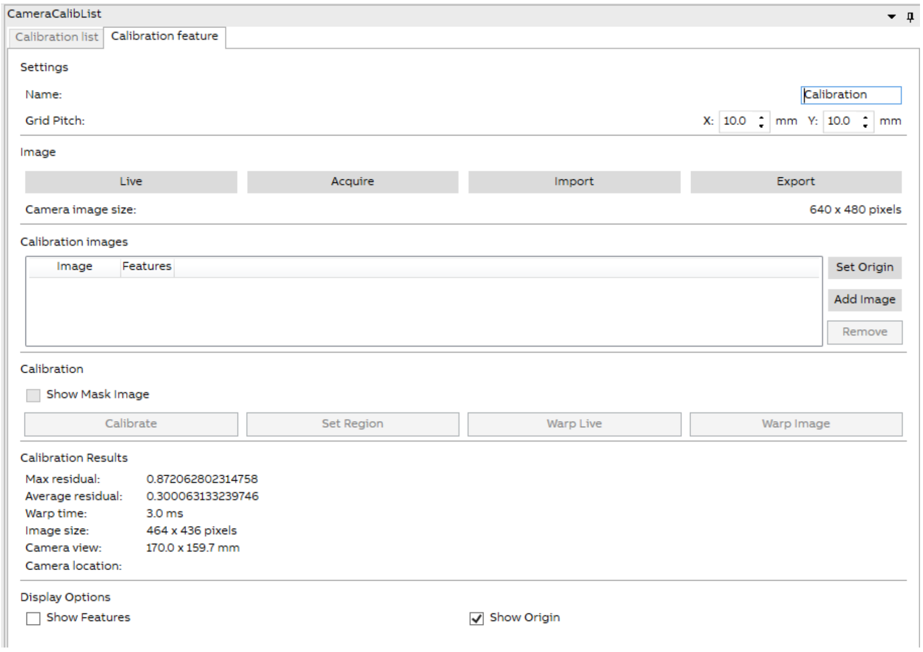
3 Select the default calibration from the list and click **Edit**.

The **Camera Calibration Feature** and **Calibration** dialog are opened.



xx1800001469

Continues on next page



xx2100001642

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.

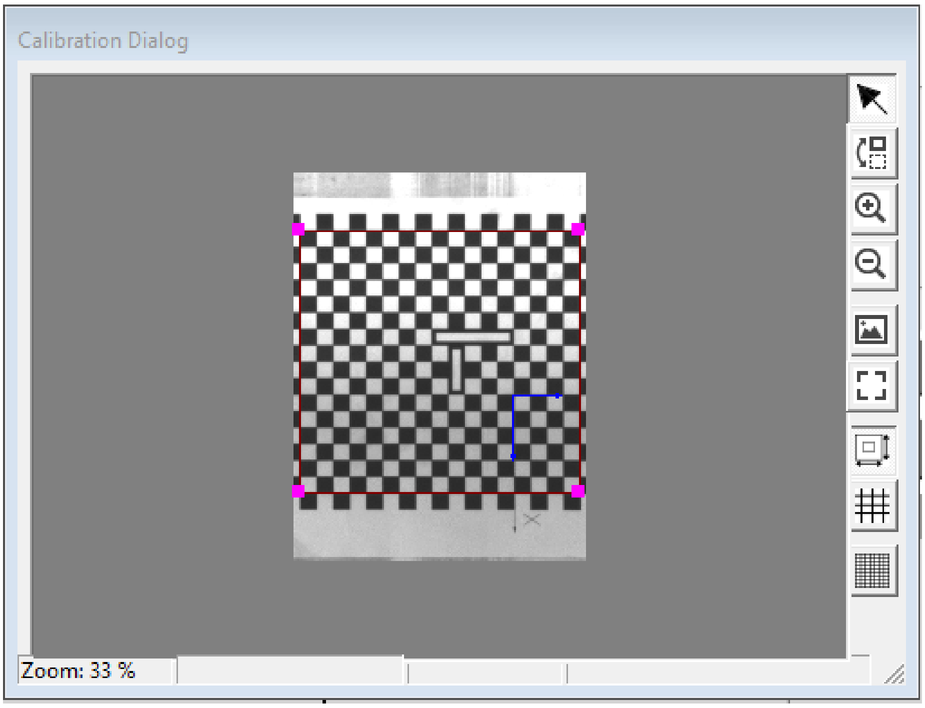
   The other way is to use two calibration plates with the exact same grid pitch. Put one calibration plate in the position to represent the origin of the camera. Acquire an image and click **Set origin**. Leave this plate in place while acquiring images of the second calibration plate at different angles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

> **Tip**
>
> With multi-view calibration, the space information is calculated automatically.

Continues on next page

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.



xx1800001703

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 278*.

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.

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

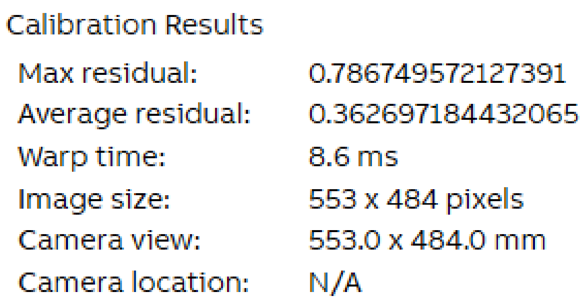
For conveyors, leave the calibration paper as it is until the base frame has been calibrated.

Continues on next page

> **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.

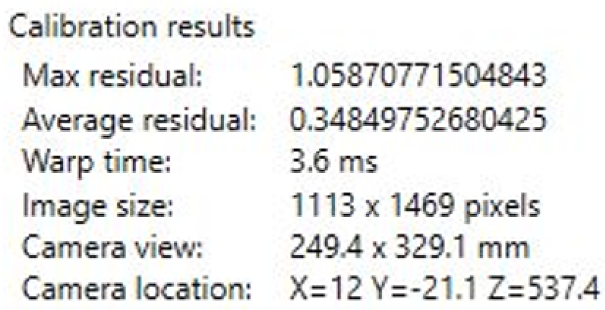
Calibration result

Single-view calibration:



xx2100001646

Multi-view calibration:



xx2300001517

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.

Continues on next page

4.4.8.1 Showing live images

Live images

It is possible to view images from each camera when a production 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 **Control.** 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 283*.

4.4.8.2 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 recorded 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.

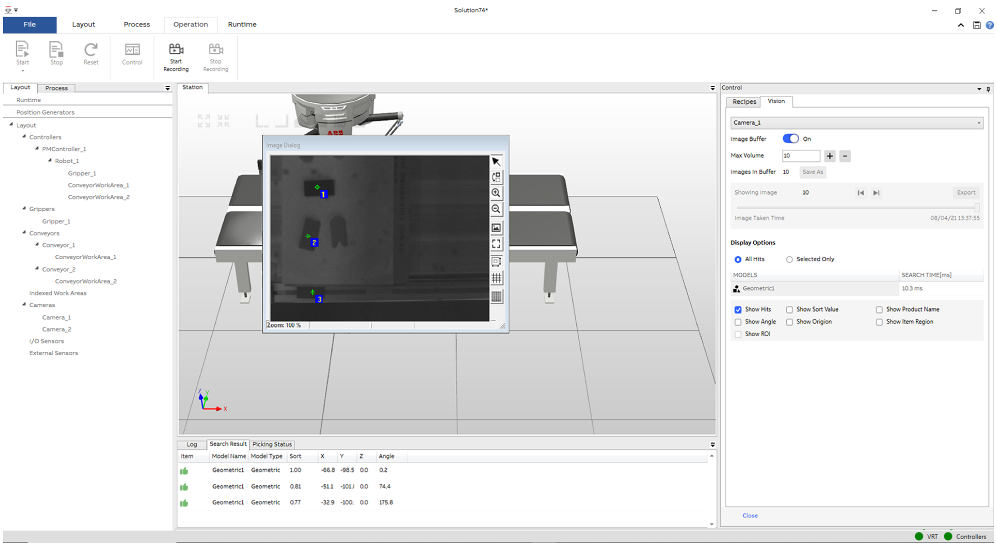
When switching off the Image Buffer function, images are no longer added and the images in the buffer can be analyzed. Save the images in the current buffer to file for later analysis with the Vision Analyzer program, see *Vision Analyzer on page 281*.

You can switch to different cameras from the drop-down list.

The maximum size of the buffer depends on the RAM memory on the computer.

Illustration

Click **Vision** tab under **Control** to open the dialog. By default, the recording state is activated and the buffer max volume is set to 10 images.



xx2100001638

Image buffer	Used to switch between recording or pause and set the image buffer size. Click **Save As** to save all images in the buffer to a .pmv file. Step through the image buffer when recording is paused. LEFT or RIGHT ARROW button can also be used to step. Click **Export** to save the current image to file (.bmp format).
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.
Search Result	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.
Image Dialog	The pan and zoom buttons can be used to analyze the image more closely.

Continues on next page

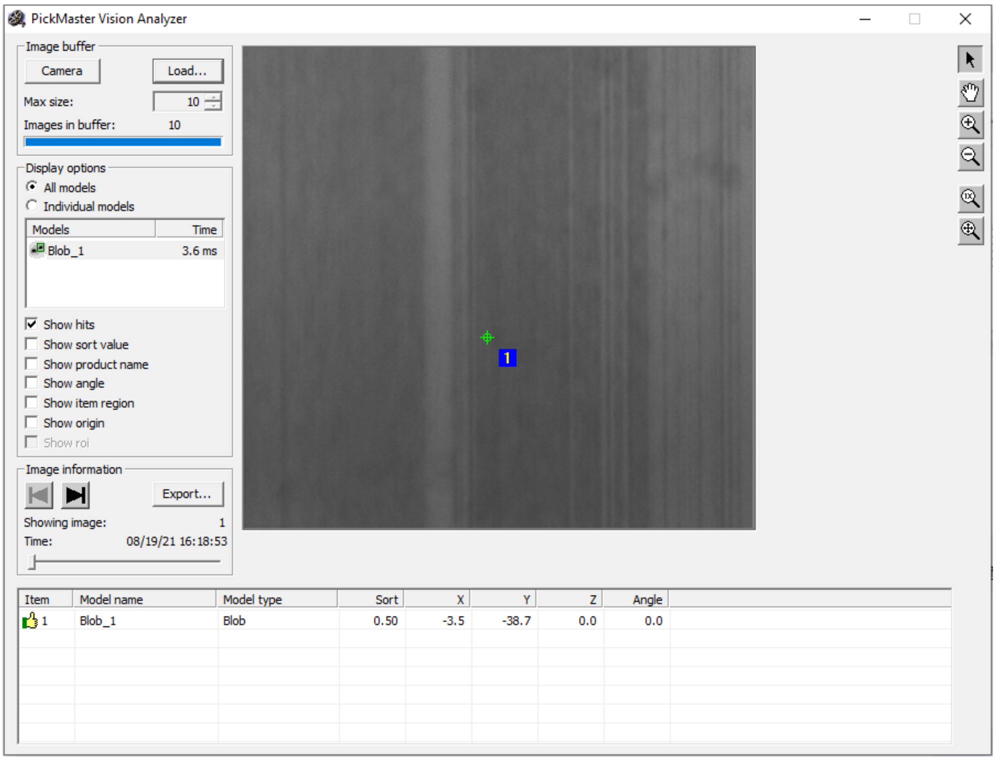
Note

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 PickMaster Twin Client installation folder or from Windows Start menu.



xx2100001831

Click **Load** to open a .pmv file.

Click **Camera** to see detailed information about the camera that took the images.

Other settings in Vision are identical to settings in **Detailed Vision Information.**

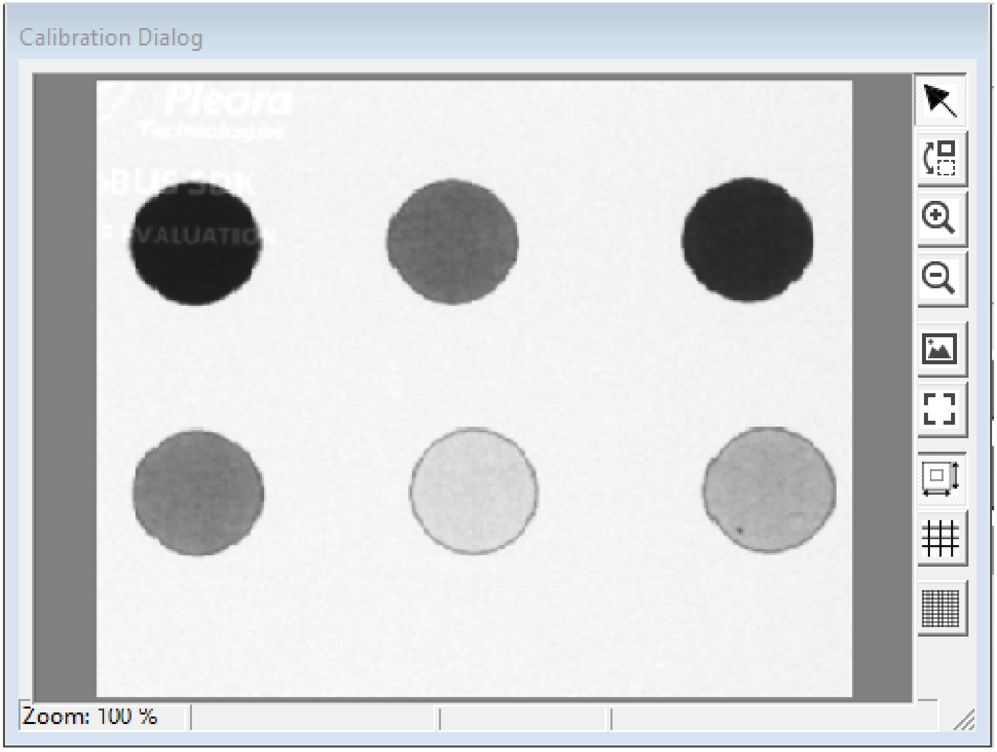
4.4.8.3 The image windows

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.



xx2100001521

4.4.9 Adding vision model

4.4.9.1 Vision modeling

Introduction to vision modeling

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

- *Geometric PatMax* which is a pattern recognition tool. See *Configuring a geometric model with PatMax on page 286*.
- *Blob* which is a detection of two-dimensional shapes within images. See *Configuring blob models on page 294*.
- *Inspection tool* (Inspection II) which makes it possible to combine the *PatMax*, Blob, Histogram and Caliper to generate a model. See *Configuring inspection models on page 301*.

Note

Vision modeling can only be created or edited when the software is connected to real Runtime.

Note

You can import vision models from PickMaster 3 solutions and other PickMaster PowerPac solutions.

Importing an existing vision model

Use this procedure to import an existing vision model.

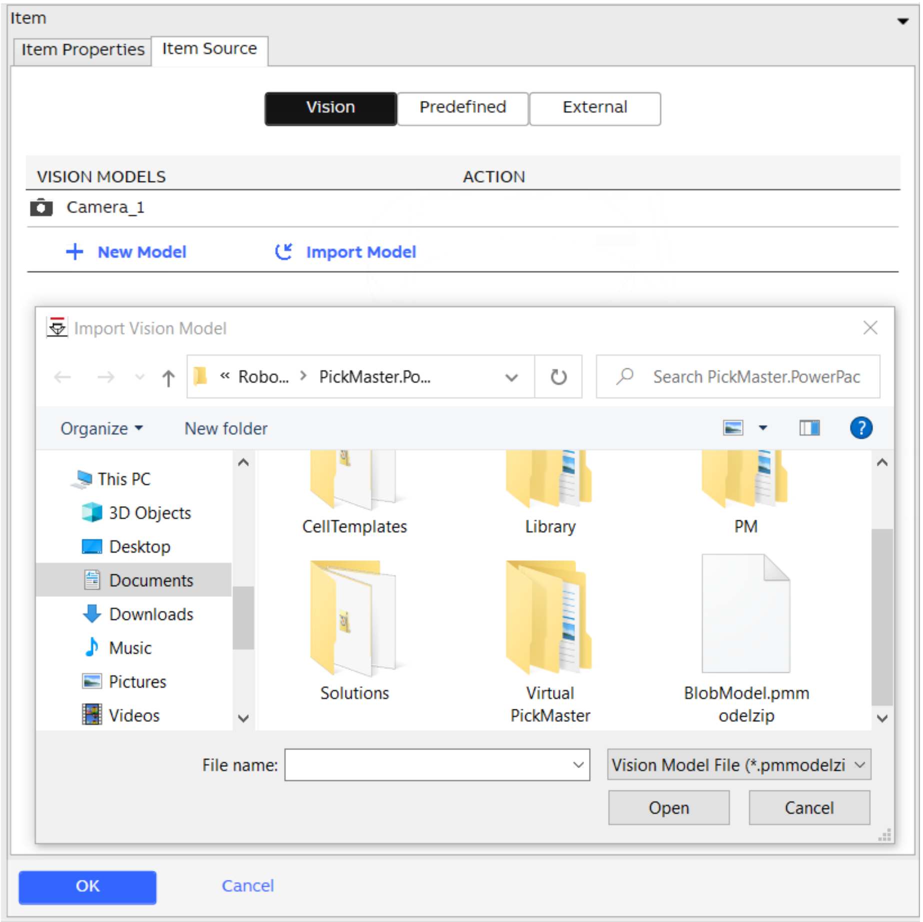
1. Right-click on one **Item** in the tree view **Process** and select **Setting.**

   The **Item Setting** window is opened.
2. Click to select the **Item Source** tab.
3. In the **Item Source** dialog, click **Import Model** under the required camera.

   The **Import Vision Model** window is opened.

Continues on next page

4. Select the valid vision model (`.pmmodel` or `.pmmodelzip`) and click **Open**



xx2200000479

5. Click **OK**.

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 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.

External vision models

This function is reserved for next version.

Continues on next page

Related information

Configuring a geometric model with PatMax on page 286.

Configuring blob models on page 294.

Configuring inspection models on page 301.

4.4.9.2 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.

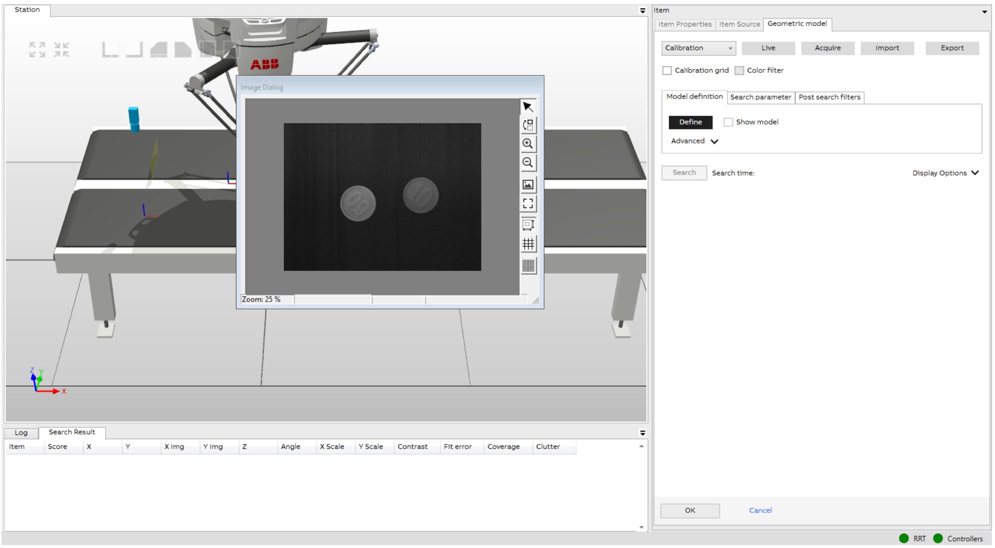
- Select a representative pattern with consistent features. Reduce needless features and image noise. Train only important features. If necessary, export the image and use an external program to erase noise.
- Larger patterns will provide greater accuracy because they contain more boundary points to 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 *Item Properties tab on page 136*.

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 310*.

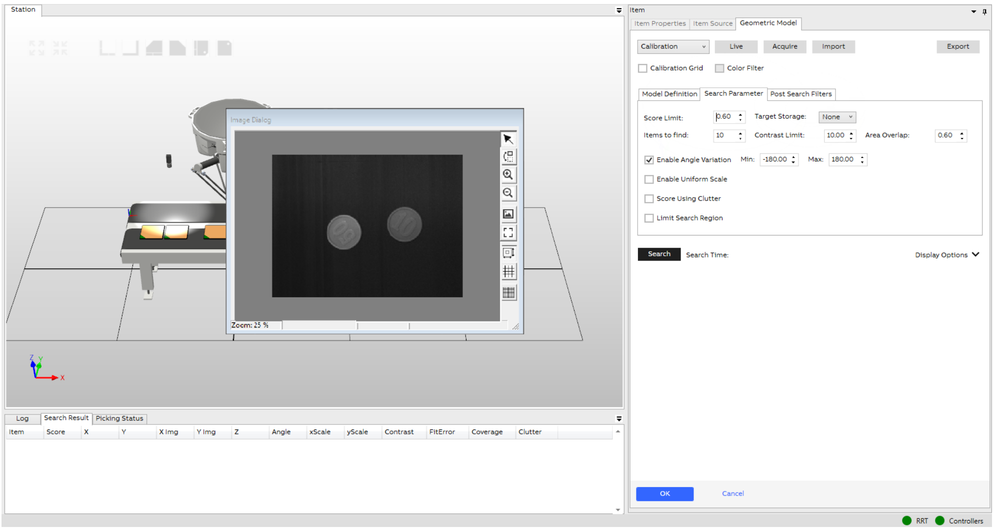
There are several parameters that can be adjusted to make an efficient model. The configuration is done in the **Geometric Model** tab view and the result is displayed in the **Search Result** window and the **Image Dialog**.

Illustration geometric model Configuration

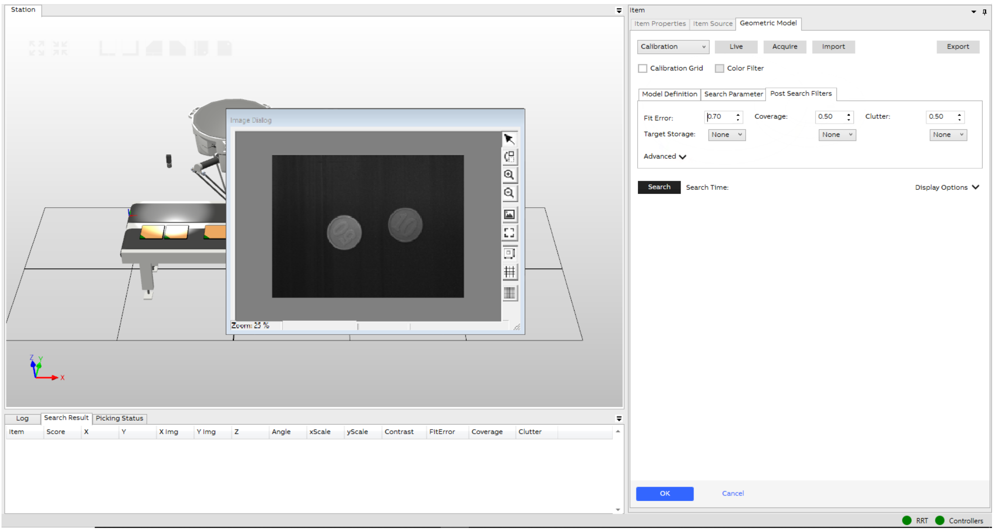


xx2100001647

Continues on next page



xx2100001648



xx2100001649

Configuring a geometric model with *PatMax*

Use this procedure to configure a geometric model with *PatMax*.

1. Right-click on one **Item** in the tree view **Process** and select **Setting.**

   The **Item Setting** window is opened.

2. Click to select the **Item Source** tab.

3. In the **Item Source** dialog, click **New model** under the required camera and select **Geometric.**

   The **Image Dialog** and **Geometric** dialog are opened.

4. In the **Model Definition**, click **Live**, **Acquire**, or **Import** to get an image. Select the calibration that has set in the **Camera Calibration** from the **Calibration** list. Select the **Calibration grid** checkbox to display help lines for the coordinate system.

Continues on next page

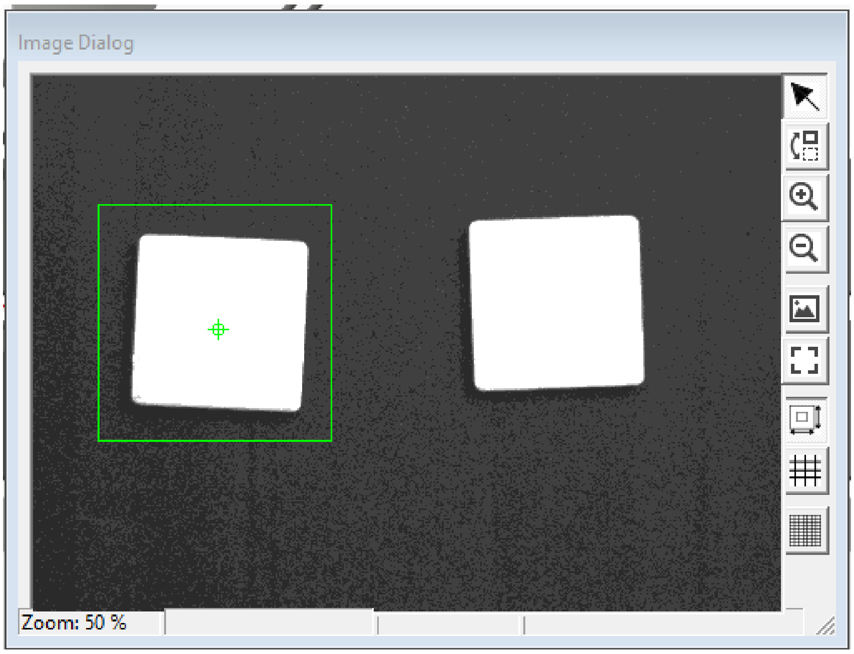
The help lines can be moved with the mouse to make it easier to train a pattern.

5. If color filtering should be used select the **Color filter** checkbox to enable the filter. Configure the filter parameter in the **Color Filter** tab. See *Using color vision on page 310*.
6. 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. If the height of the item is to be defined, choose an appropriate calculation method before training the item. **Model Height** is used as the basic height for the trained item. **Pick Offset** is used to make up the deviation of the picking point with this calculation method. For more information, see *Working with products of varying height (2.5D vision) on page 357*.
   b. 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.



xx1800001708

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

Continues on next page

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

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

f 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.

g 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.

h 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.

7 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.

Target Storage indicates the variables in Rapid. For more details, see *GetItmTgt - Get the next item target on page 372*

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

Continues on next page

represent the same item in the image. When two patterns overlap by a percentage greater than the area overlap threshold they are treated as a single pattern.

Enable 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.

Enable 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 running time. Therefore, the model should be tested in running time to verify that items are identified as expected.

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

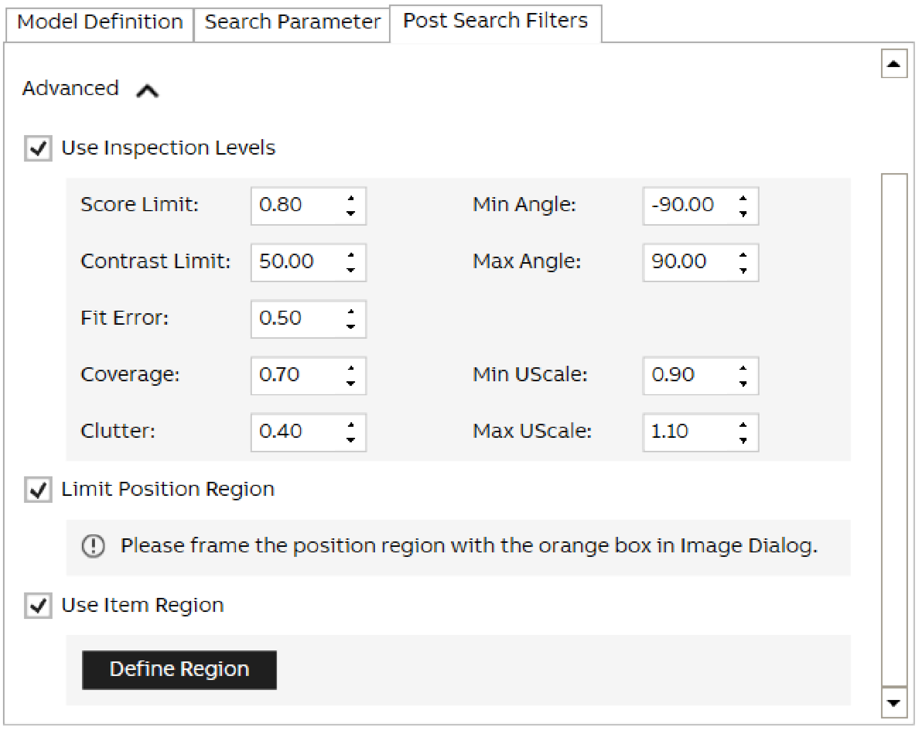
Fit Error Limit is a measure of the variance between the shape of the trained pattern and the shape of the pattern found in the search image. If the found pattern in the search image is a perfect fit for the trained pattern, the fit error is 0.

Coverage 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 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.

Continues on next page

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



xx2100002333

Use Inspection Levels - Inspection I, this inspection is also called *Inspection I* in PickMaster PowerPac. 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 *Item Properties tab on page 136*. An item type can be read in the RAPID code, see *RAPID programs on page 399*.

In the Inspection parameter section, all models that fulfill the conditions specified for the search parameters and the post filters will be classified. Select **Use Inspection Levels** to 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.

Continues on next page

To define an item region, select **Use Item Region** checkbox and click **Define Region**. Adjust the polygon showed around the found object using vertices. Then click **Train**. The polygon can have 2 to 16 vertices.

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

Match Info displays the quality of the matched boundary points in the search image. 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.

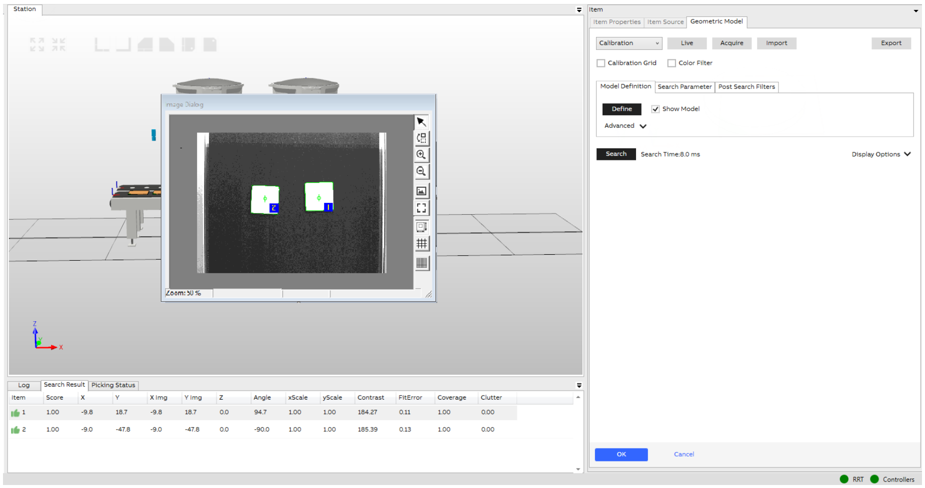
Item Region displays the regions in the image window. Red regions indicate an overlap and the corresponding hits will be considered as discarded.

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

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.**

10 Click **Search** to analyze the image. If needed, define sort value.

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



xx2100001636

Model hits are normally classified as accepted. If inspection is used, hits can be classified as either accepted or rejected. See *Item Properties tab on page 136*. 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.

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

Continues on next page

11 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 running time. 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

Item Properties tab on page 136.

Using color vision on page 310.

RAPID programs on page 399.

4.4.9.3 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.

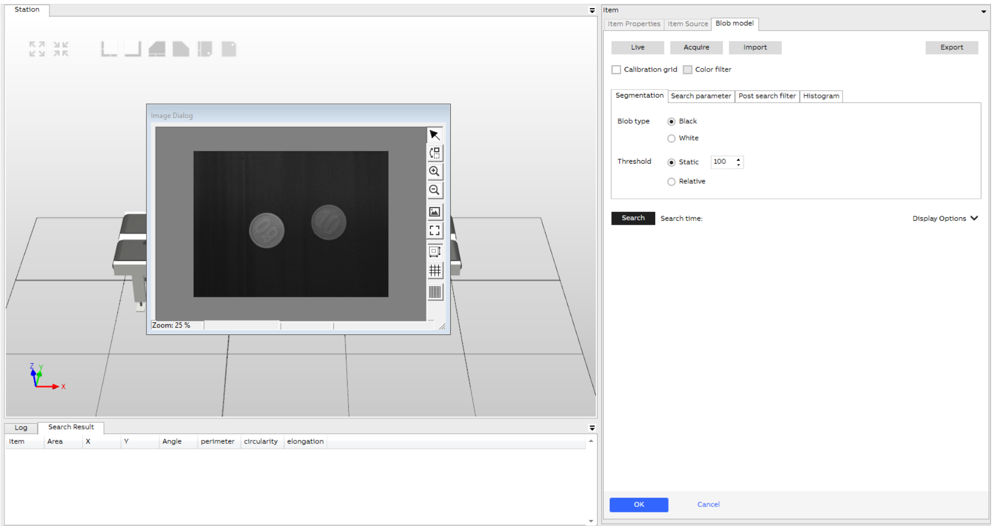
This kind of analysis is well suited for applications where:

- 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.

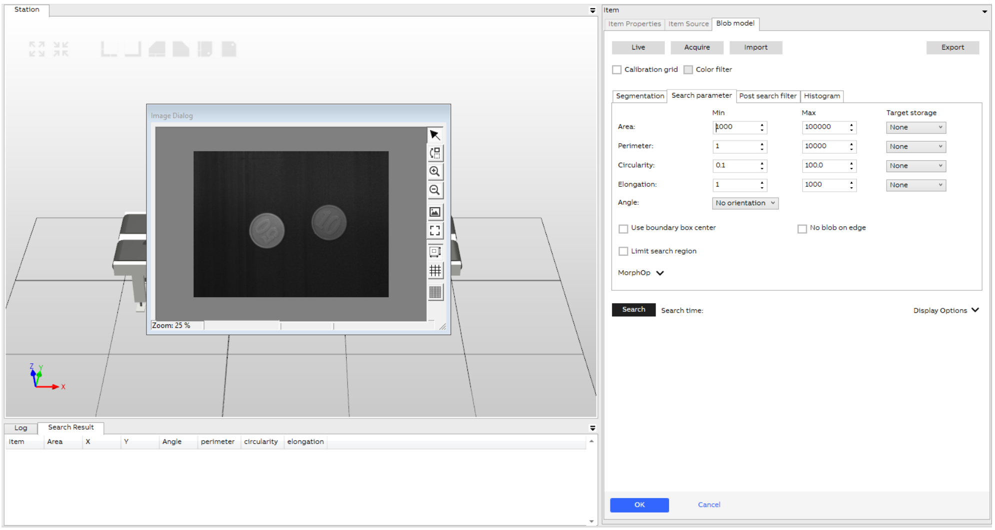
To increase the contrast in images where parts have similar grayscale tone, it is possible to use color filtering. See *Using color vision on page 310*.

Illustration Blob Configuration

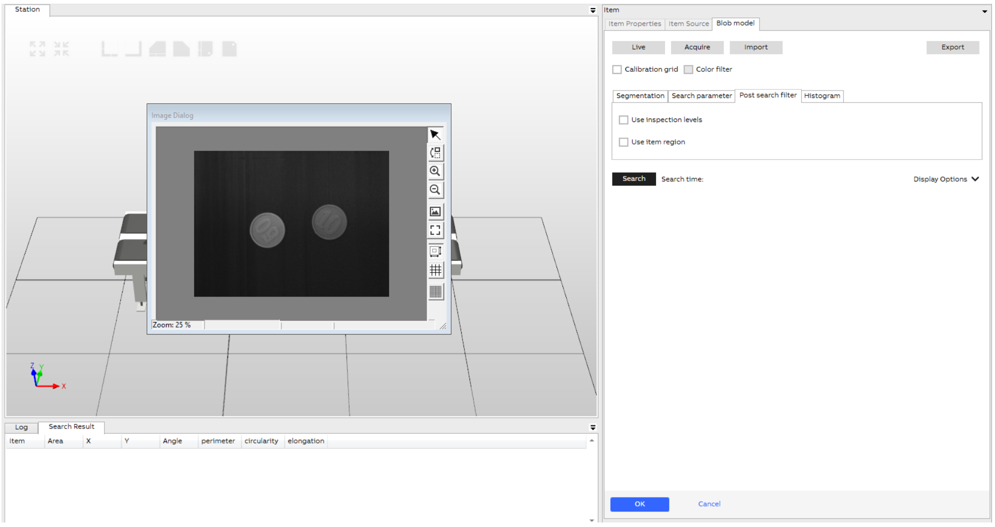


xx2100001661

Continues on next page

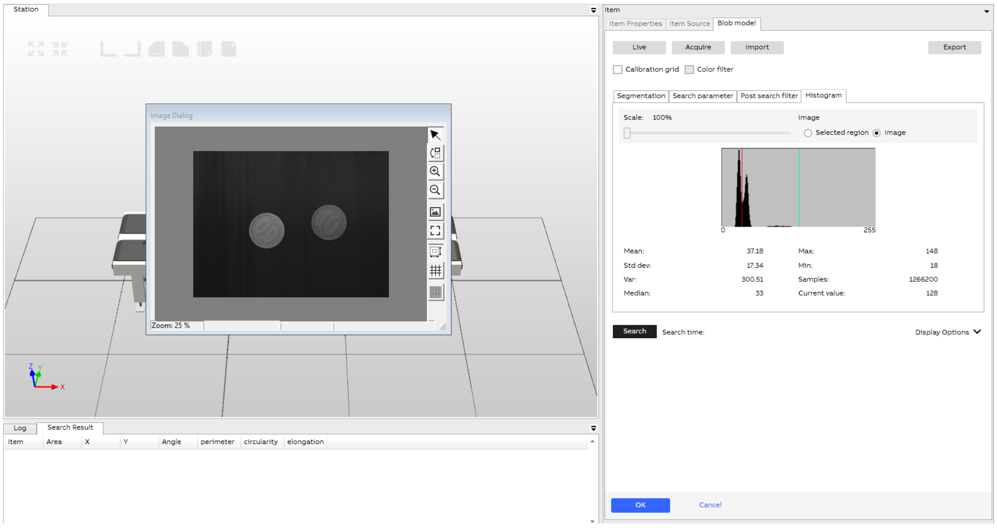


xx2100001662



xx2100001663

Continues on next page



xx2100001664

Configuring a blob vision model

Use this procedure to configure a blob vision model.

1 Right-click on one **Item** in the tree view **Process** and select **Setting**.

   The **Item Setting** window is opened.

2 Click to select the **Item Source** tab.

3 In the **Item Source** dialog, click **New model** and select **Blob**.

4 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 configure the filter parameter in the **Color Filter** tab. See *Using color vision on page 310*.

5 Click to select **White** in the **Segmentation** under **Model Definition**.

   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

Continues on next page

segmentation can test for absence of a feature in a scene, whereas relative segmentation will always find a blob in the scene.

6 Adjust the parameters in the **Search Parameter** according to your requirements.

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.

Use Inspection Levels defines if the found models should be classified. See *Item Properties tab on page 136*. The item type can be read in the RAPID code, see *RAPID programs on page 399*. Select **Use Inspection Levels** to open the Inspection Parameters part.

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.

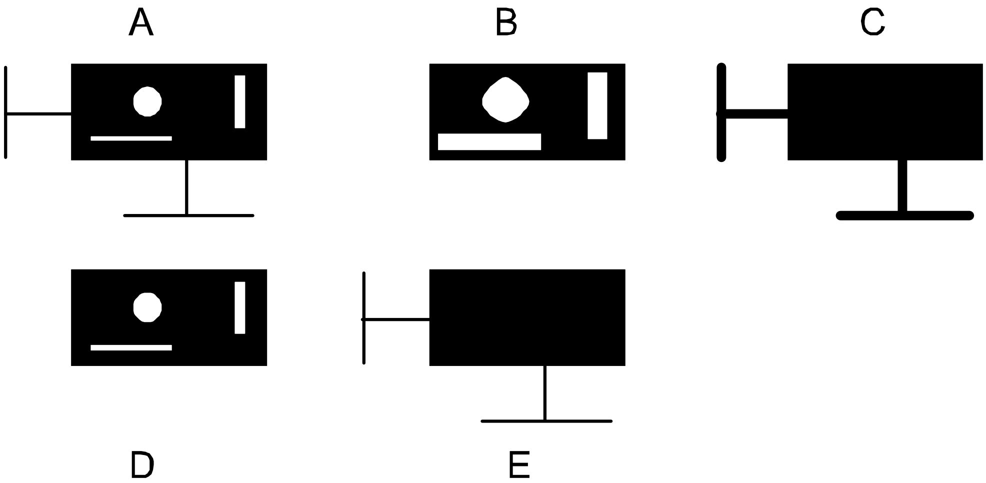
Limit Search Region limits the search area for the blob analysis. Only objects within this area will be found.

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.

Continues on next page

7 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.

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

Continues on next page

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.

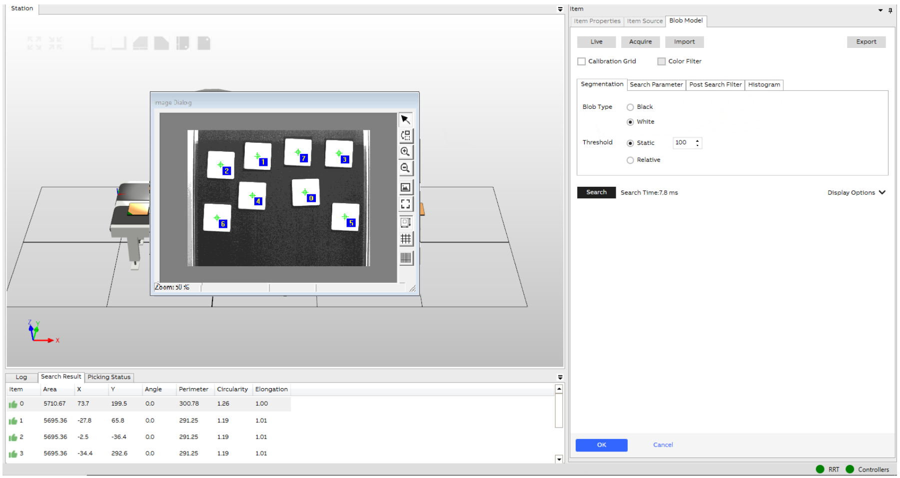
8 In the **Item region** part, select the **Use Item Region** checkbox and click **Define Region**. Adjust the polygon showed around the found object using vertices. Then click **Train**.

The polygon can have 2 to 16 vertices.

9 Click **Search** in the **Display Options**.

Tip

If the search result matches with the **Image Dialog**, the configuration succeeds.



xx2100001637

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

- **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.
- **Blob angle** displays the angle of the item that will be sent to the robot.
- **Score Value** displays the score for the selected item in the image window.

10 Click **OK**.

Blob parameters in item targets

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

Continues on next page

Related information

Item Properties tab on page 136.

Using color vision on page 310.

RAPID programs on page 399.

4.4.9.4 Configuring inspection models

Introduction to inspection models

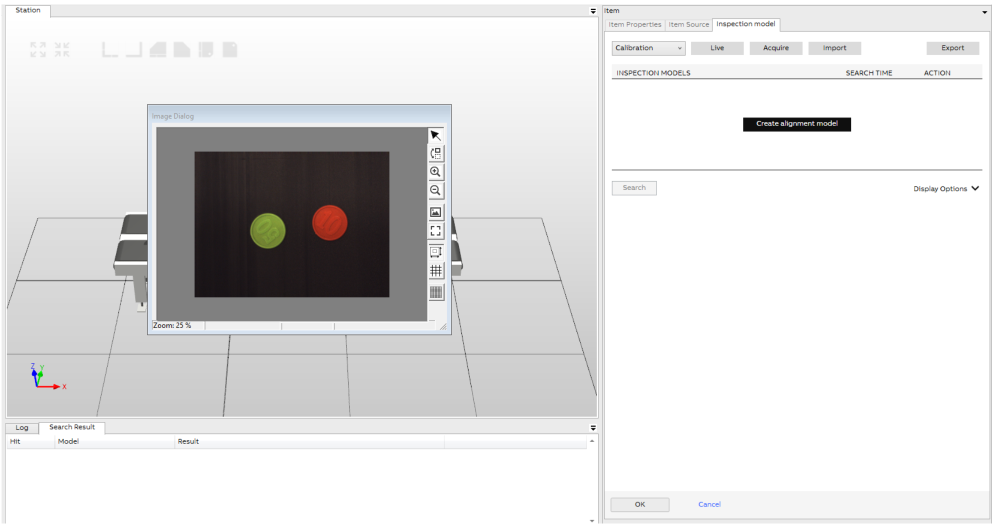
Inspection models make it possible to combine several models of *PatMax*, blob, histogram and Caliper. This is sometimes referred to as *Inspection II*.

An inspection model always consists of an alignment model. The alignment model can either be a *PatMax* or blob 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, histogram, *Caliper* or *PatMax* 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.

Illustration Inspection Configuration



xx2100001668

Configuring inspection models

Use this procedure to configure inspection models.

1. Right-click on one **Item** in the tree view **Process** and select **Setting.**

   The **Item Setting** window is opened.
2. Click to select the **Item Source** tab.
3. In the **Item Source** dialog, click **New model** and select **Inspection.**
4. In the **Image** part, click **Live**, **Acquire**, or **Import** to get an image.
5. 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.

Continues on next page

Alignment Model defines the position and orientation of any found items. For more information on the alignment model configuration dialog, see *Vision modeling on page 283*.

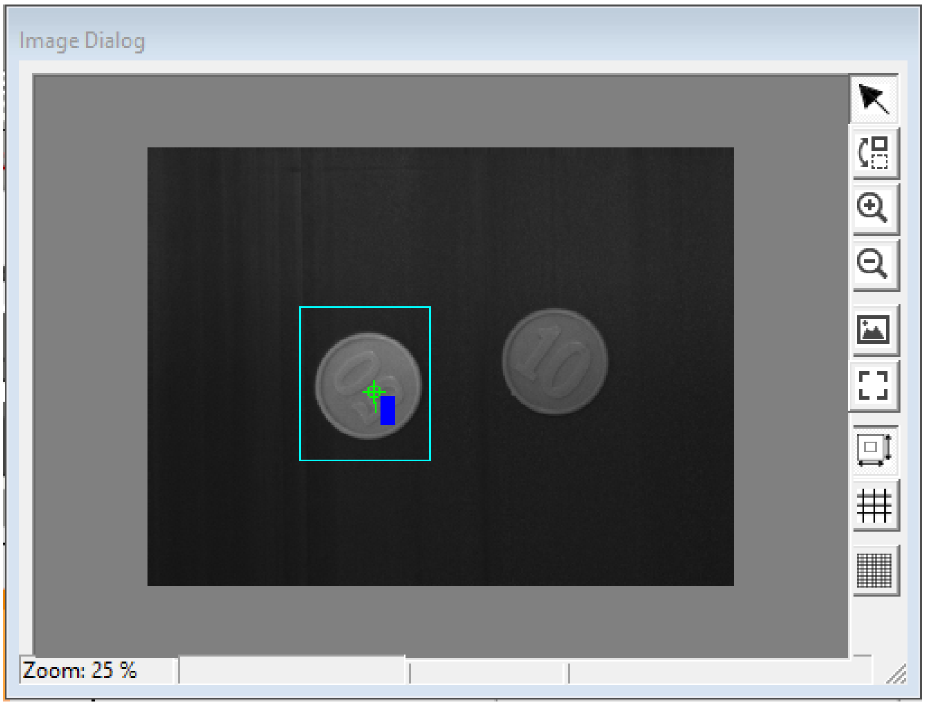
Sub Inspection Model adds inspection areas to an alignment model. See *Sub inspection models on page 303*.

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 Area shows the current model's area. The area can be rearranged for the selected sub inspection model.

6 Click **Create Alignment Model** to open the **Select Model Type** drop-down list.
7 Select **Geometric** or **Blob** in the drop-down list to create the alignment model. For detail procedures on how to create a geometric model or a blob model, see *Configuring a geometric model with PatMax on page 286* or *Configuring blob models on page 294*.
8 Click **+ Sub Inspection Model** to open the **Select Model Type** drop-down list.
9 Select **Geometric**, **Blob**, **Histogram** or **Caliper** in the drop-down list to create the sub model.
10 Click **OK** on the popped-up dialog to edit area.
11 Drag the rectangle so it covers the pattern.



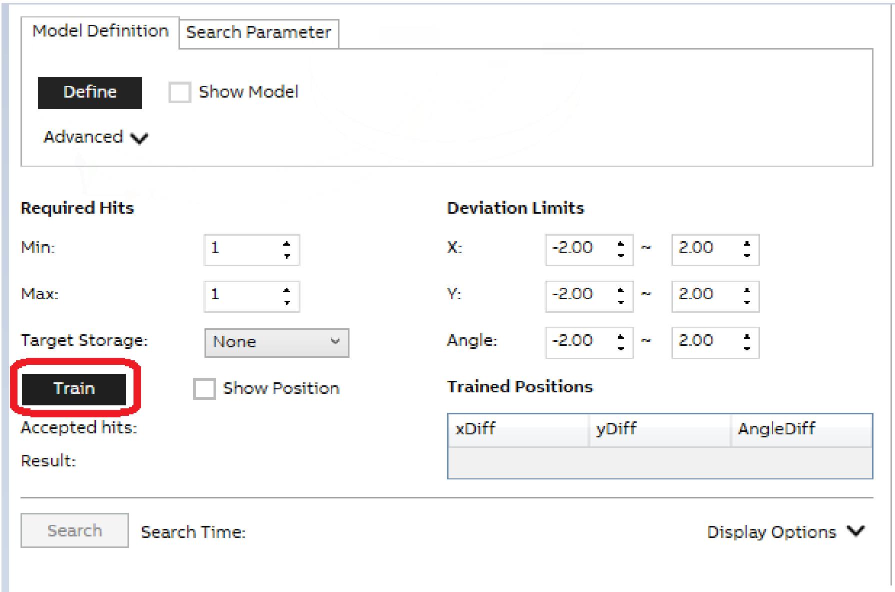
xx2100002275

Continues on next page

12 Click **Edit** button to open the corresponding model creating window. For detail procedures on how to create a Geometric, Blob, Histogram or Caliper model, see *Configuring a geometric model with PatMax on page 286* *Configuring blob models on page 294*, *Histogram on page 304* and *Caliper on page 307*.

Note

For Geometric sub model, after **Define** and **Train** the models, another **Train** need to be done.



xx2100002277

13 Click **Search**.

The result is displayed as an image with numbered hits in the **Image Dialog** and a corresponding detailed list in the **Search Result** window.

Tip

If the search result matches with the **Image Dialog**, the configuration succeeds.

14 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 to the alignment model. The areas are shown in the image and should be moved and resized to cover the area to analyze.

Continues on next page

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.

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 286*. 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 294*. 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

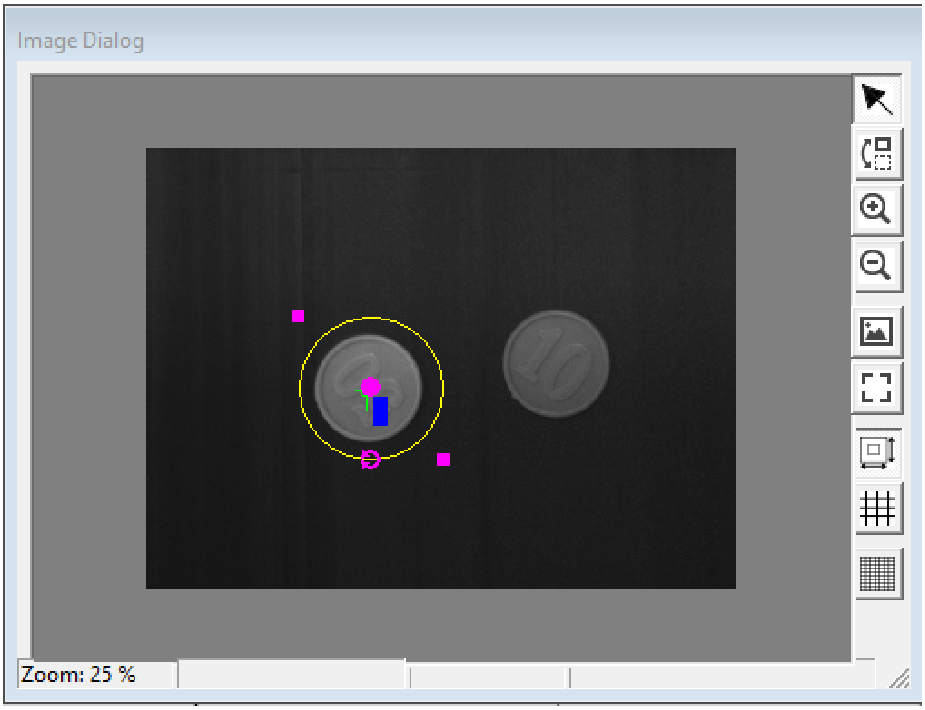
The histogram tool measures the color or the gray level within any given area. While using a monochrome camera the histogram tool measures the gray level within a given area. Similarly, if a color camera is used each of the three color channels (Red, Green, and Blue) is measured separately. The histogram 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. Click **+ Sub Inspection Model** to open the **Select Model Type** drop-down list.
2. Select **Histogram** in the drop-down list to create the sub model.
3. Click **OK** on the popped-up dialog to edit area.

Continues on next page

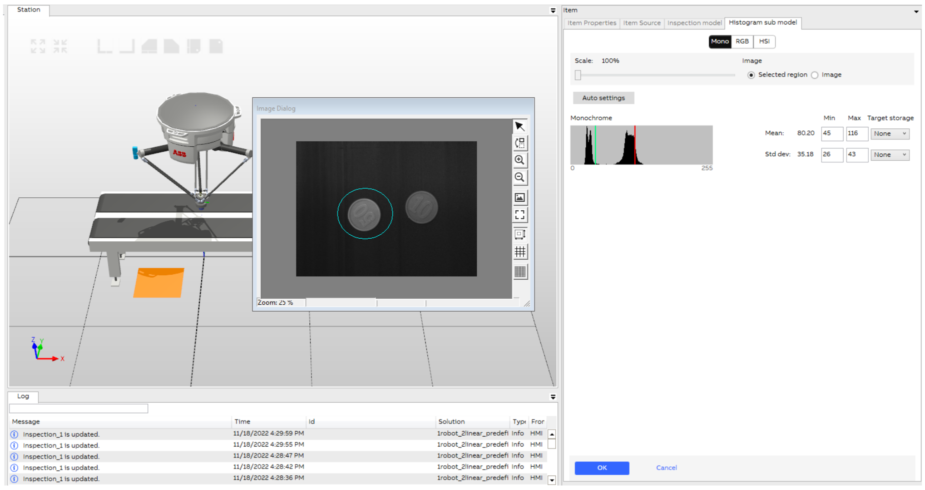
4 Drag the circle so it covers the pattern.



xx2200001129

5 Click **Edit** icon under **Action** to open the histogram model editing window.

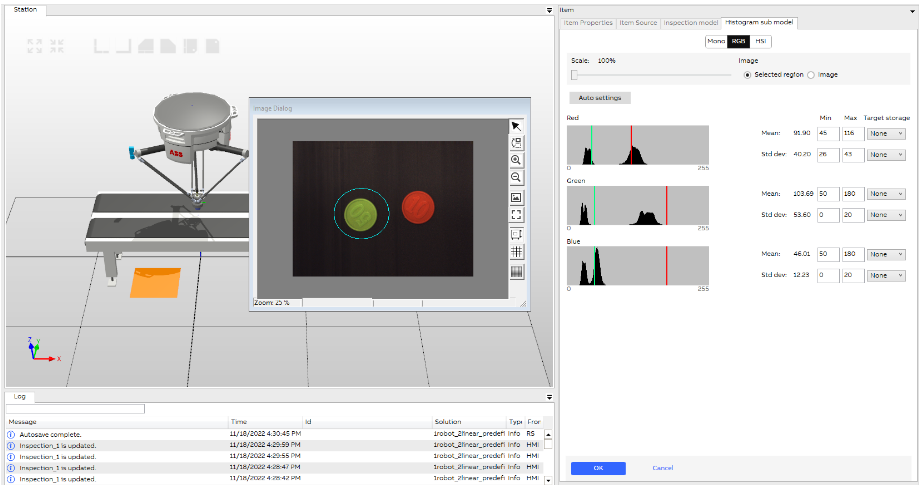
6 Press **Auto Settings** to automatically get an appropriate range limits(**Min.** and **Max.** values) for the histogram. Alternatively, the **Min.** amd **Max.** values can be set manually by sliding the red and green bars across the histogram or by simply entering values into the text boxes. For 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.



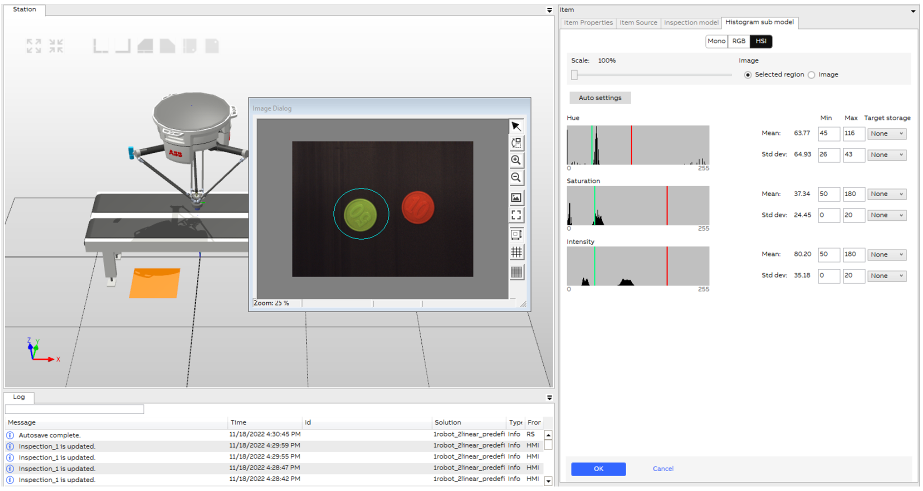
xx2200001126

Continues on next page

7 If change to Tab **RGB** or **HSI**, the window for the colors will show up.



xx2200001127



xx2200001128

8 Click **OK**.

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

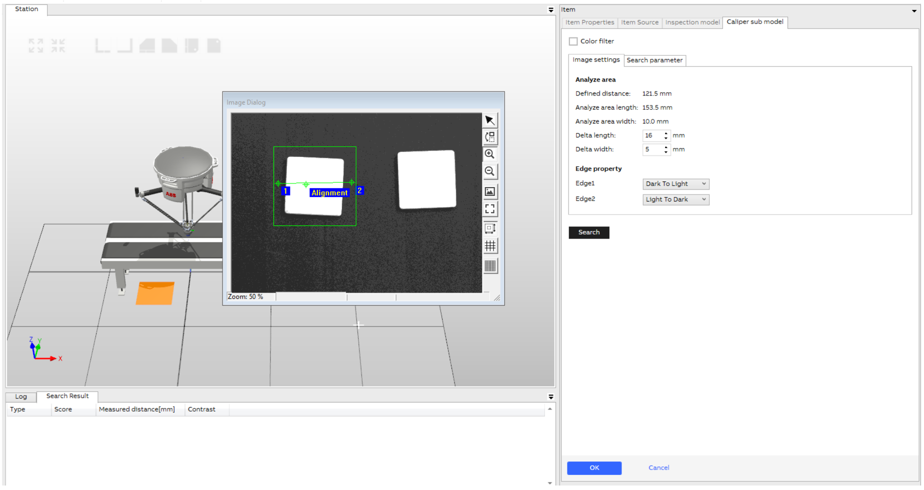
The `Mean` and `standard deviation` parameters can be selected for the target storage.

Continues on next page

Caliper

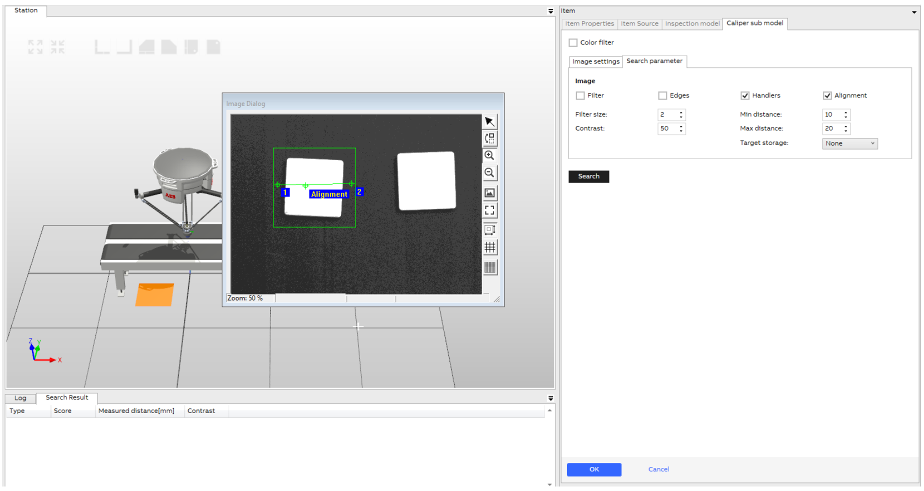
The *Caliper* tool identifies edges and measures the distance between them. The analysis is only done within the corresponding inspection area. To increase the contrast in images where parts have similar grayscale tone, it is possible to use color filtering. For more information, see *Using color vision on page 310*

1. Click **+ Sub Inspection Model** to open the **Select Model Type** drop-down list.
2. Select **Caliper** in the drop-down list to create the sub model.
3. Click **OK** on the popped-up dialog to edit area.
4. Drag the rectangle so it covers the pattern.
5. Click **Edit** icon under **Action** to open the Caliper model editing window.
6. Move the line so the end points are located on the edges of the area under the **Image settings**.



xx2200001119

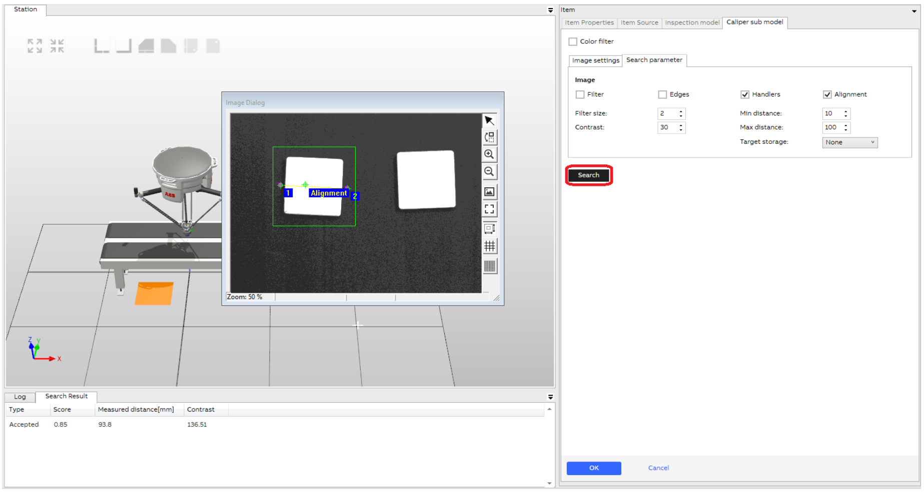
7. Adjust the parameters in the **Search parameter** according to the **Defined distance** in the **Analyze area**.



xx2200001120

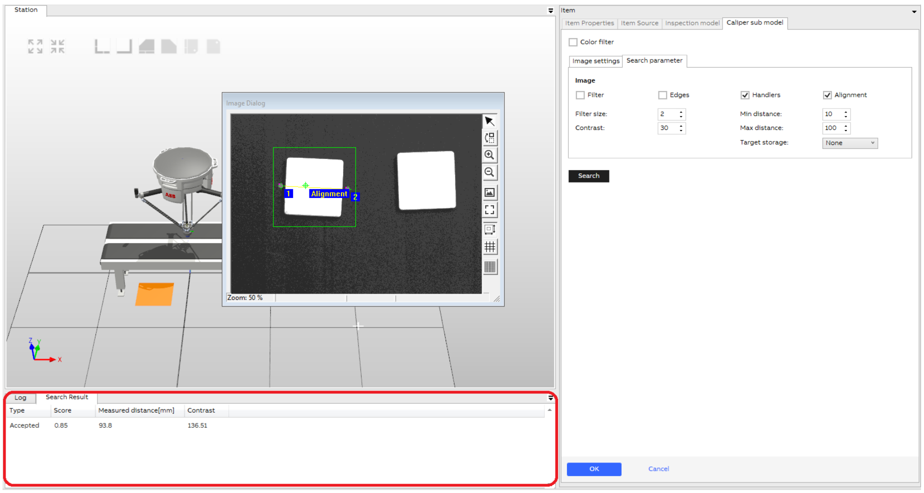
Continues on next page

8 Click **Search**.



xx2200001121

The result is displayed in the **Search Result** tab.



xx2200001122

9 Click **OK**.

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 **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.

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
```

Continues on next page

Delta width defines the width of the analyze area.

```
Analyze area width=2*Delta width
```

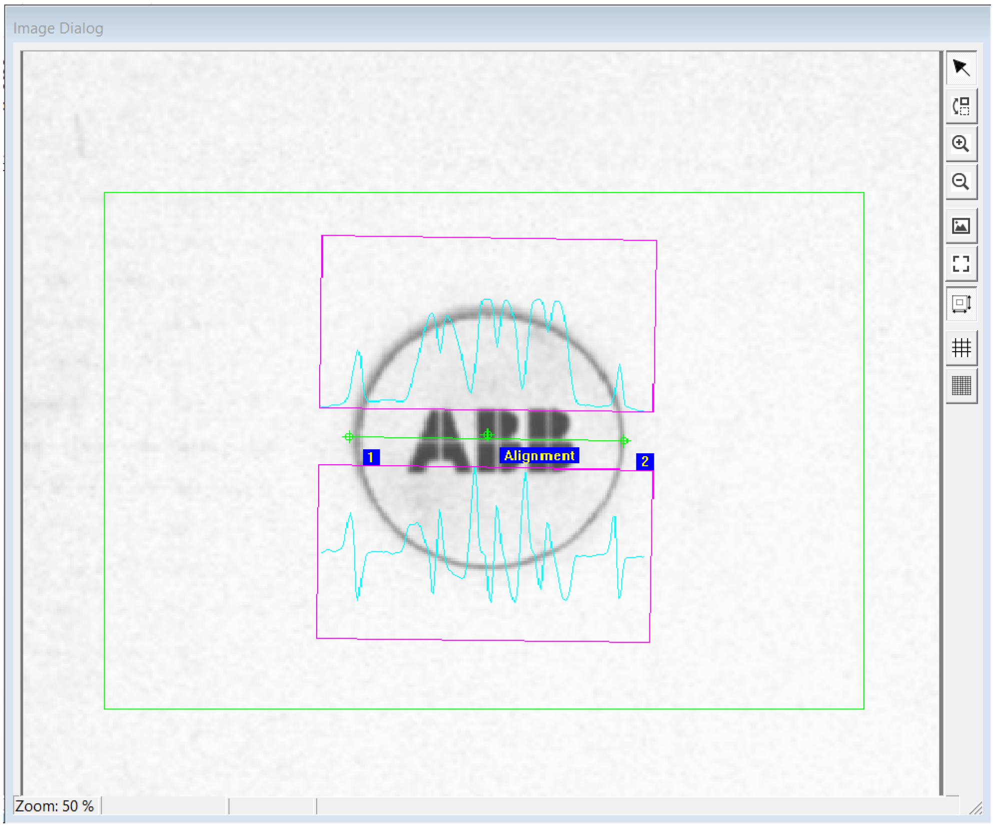
From the analyze area a production image is created. The operation sums all the information in the analyze area, accentuating the strength of edges that lie parallel to the **Analyze area width** and reducing the effects of noise.

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

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

The checkboxes in the **Search parameter** define which results should be displayed in the **Image Dialog**.



xx2200001123

Caliper subinspection parameters in term targets

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

External model

This function is reserved for next version.

4.4.9.5 Using color vision

Introduction to color vision

PickMaster PowerPac 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.

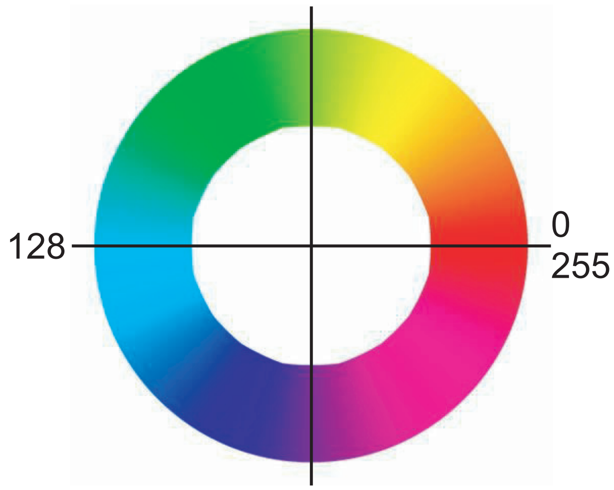
Color spaces

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.



xx2100002336

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.

Continues on next page

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.

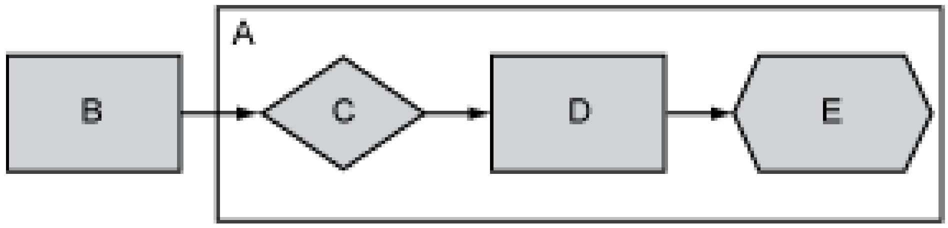
Computer performance

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 PowerPac

PickMaster PowerPac provides color vision in the form of a filter. 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.

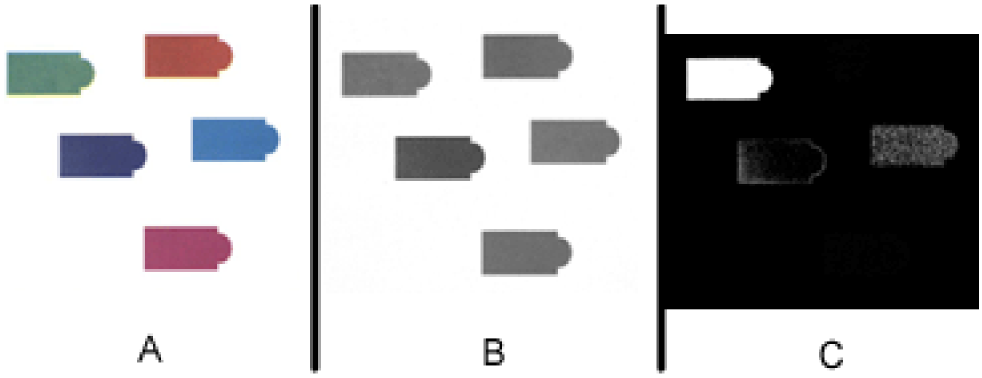


xx0900000445

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 tree view, right-click on the camera and select **Configuration**.

   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

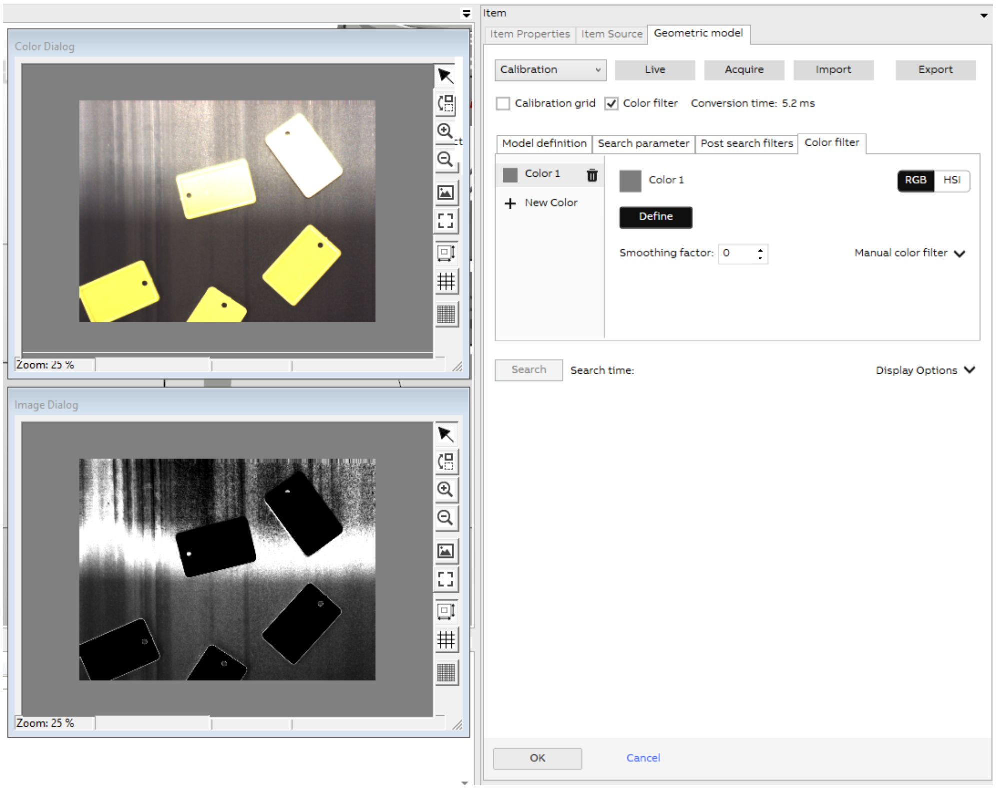
5 Click **Apply**.

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 Click **OK**.

The settings are stored in the camera. If the parameters are not saved, the camera will loose the calibration when PickMaster PowerPac is restarted.

Illustration Color Filter Settings



xx2100002268

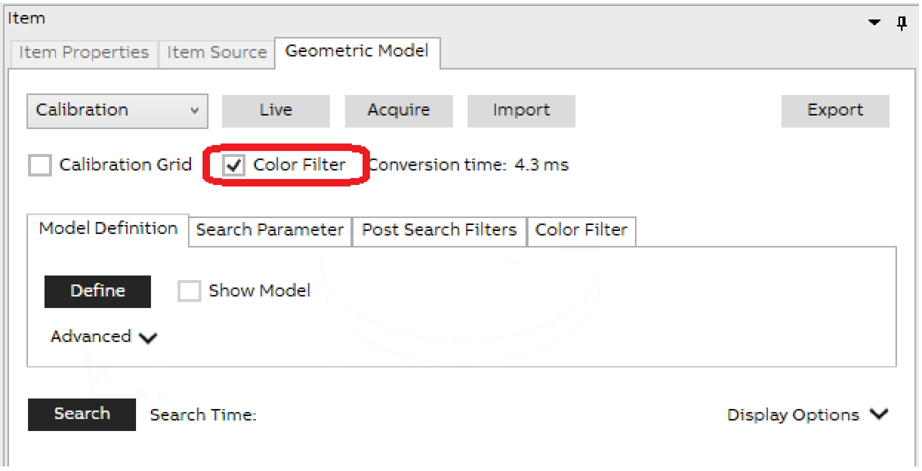
Configuring color vision

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

Continues on next page

Use this procedure to configure color vision.

1 In the *PatMax* or *Blob* configuration dialog, select **Color Filter**. This will enable the filter.



xx2100002266

The **Color Filter Settings** tab is opened together with a second video window showing the color image.

2 In the **Color Filter** tab, select **RGB** or **HSI**.

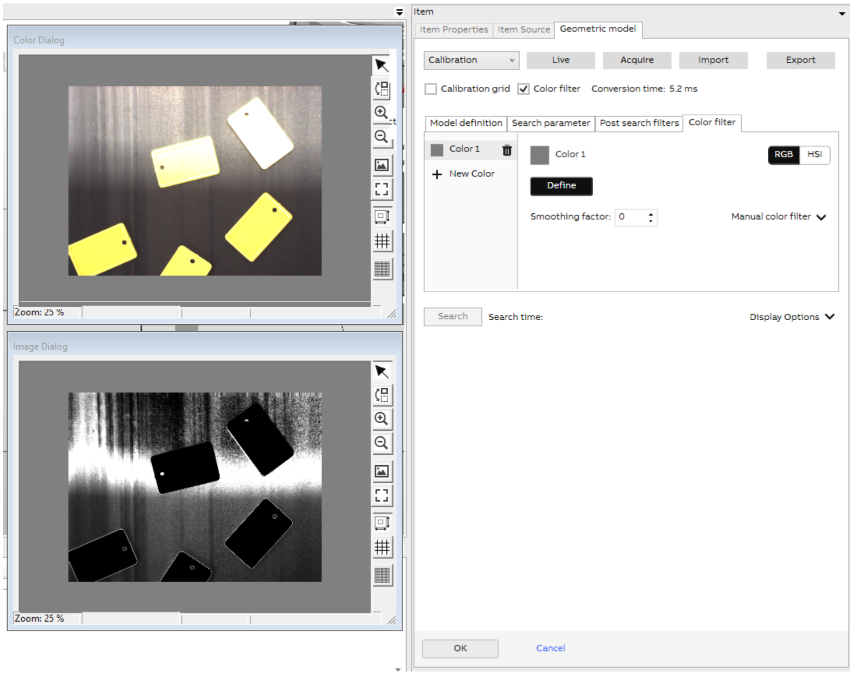
3 In the **Define color** tab, color samples can be collected from the display to indicate which colors should be enhanced.

   a Click **Define**. An adjustable rectangle will appear in the color dialog.

   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

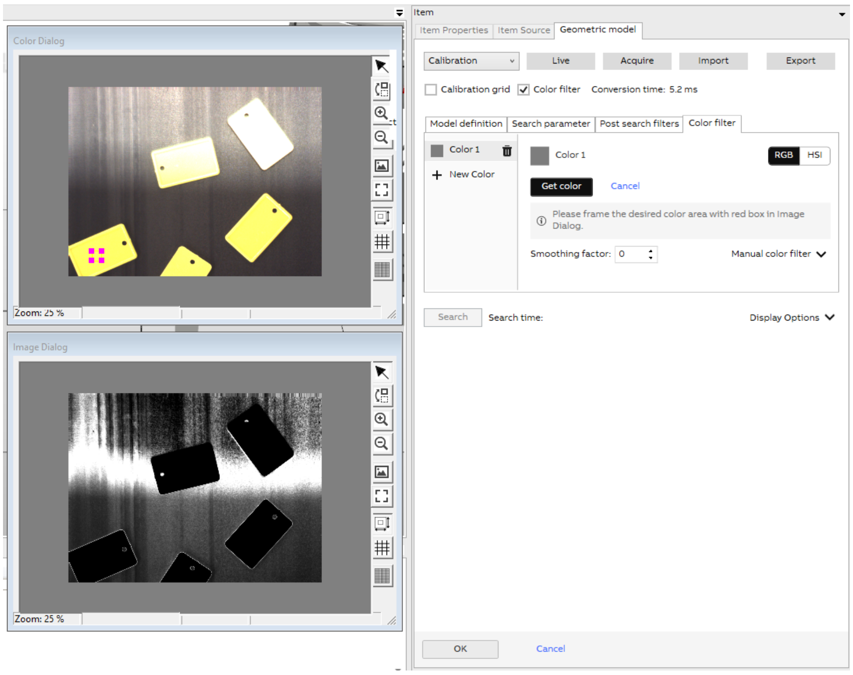
Continues on next page

output grayscale image. Colors that are dissimilar to the specified color will be converted to black.



xx2100002268

c Click **Get color** to store this color range.



xx2100002269

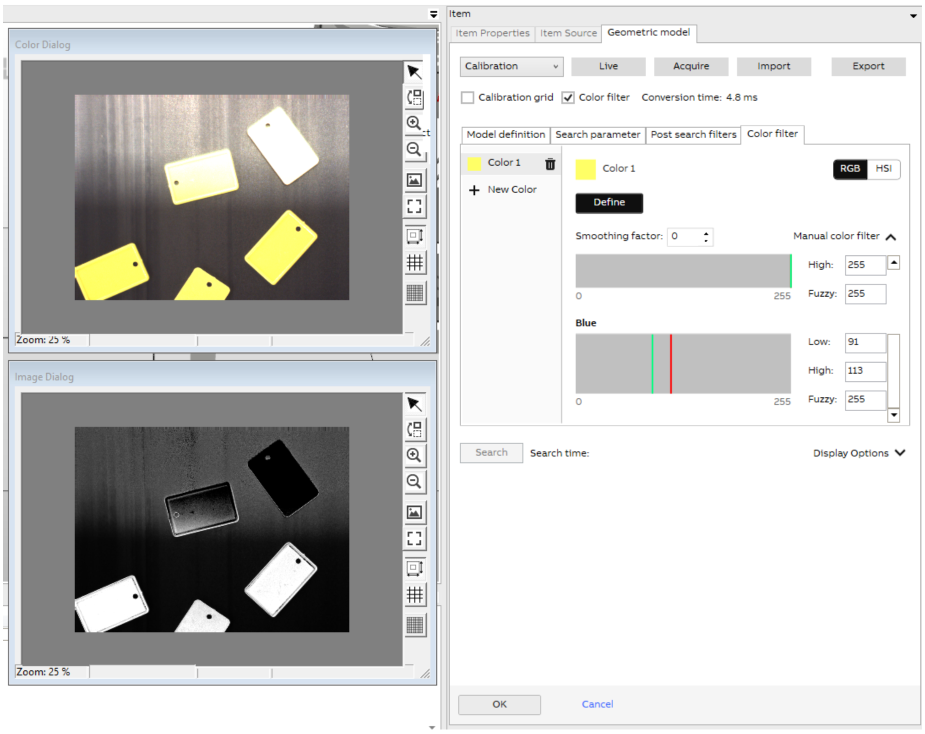
Continues on next page

4 In the **Manual color filter** tab, 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.

5 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.

6 If needed, adjust the smoothing factor to reduce noise in the resulting grayscale image.



xx2100002270

7 Proceed to define the object recognition model.

Continues on next page

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

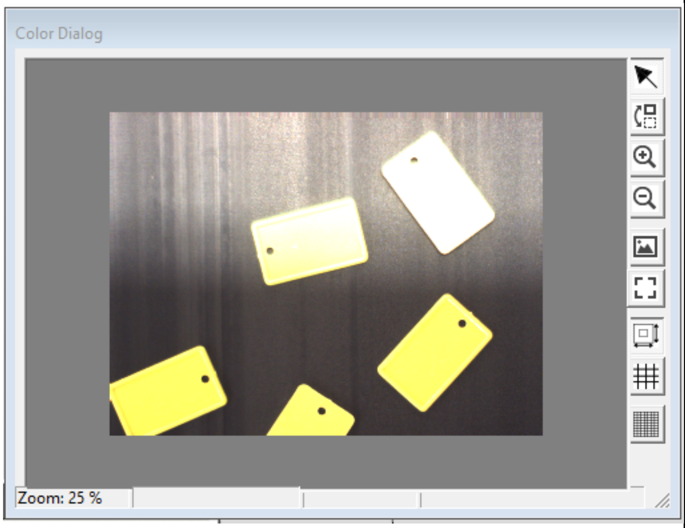
Try to filter with both RGB and HSI. Sometimes one may work significantly better than the other.

Example 1

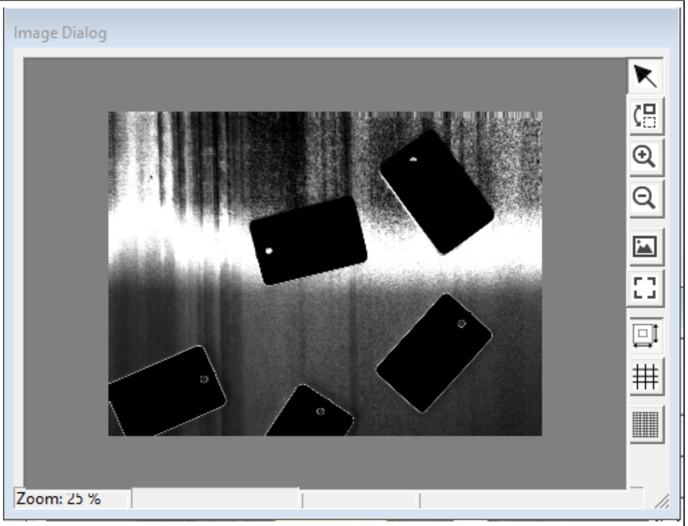
This example describes how to locate a part with *PatMax* and inspect the color with *Blob*.

1. Create an inspection model, see *Configuring inspection models on page 301*.
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** checkbox. This opens the **Color Filter Settings** tab.
   b. Extract the color to be inspected by clicking **Define color**. This filters the desired color into white in the Blob image window.
   c. Switch to other tab to do further configuration.
   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.

Example 2

Color image before filter	 xx1900000946

Continues on next page

Black and white image after filter	 xx1900000947

4.4.10 Starting production

Production

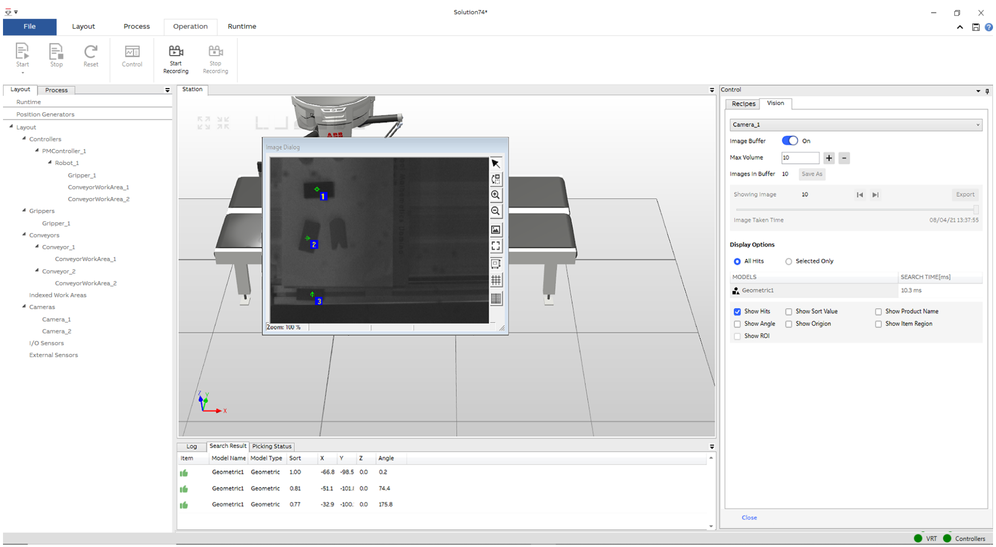
After switching to the real controller and real Runtime, all operations in the production are reflected in the real cell, and all data comes from the real system.

Select one recipe from the tree view and click **Control** on the ribbon to open the control dialog box in the solution.

The following table provides details about the **Control** dialog box.

	Description
Recipes	Control the status of the production and have an overview of the production data. For more information regarding Statics see *Recipe on page 163*.
Tuning	Adjust the parameters of the item, work area and robot. For more information regarding Tuning see *Tuning on page 163*.
Flow Control	Adjust the speed of the conveyor. For more information regarding Flow Control see *Flow Control on page 167*.
Vision	See the live video of the camera. For more information regarding Vision see *Vision on page 319*.

Vision



xx2100001638

For more information, see *Detailed vision information on page 280*.

Emulation

When running the production, the movement of 3D models in PickMaster PowerPac is called as emulation.

Use this procedure to do the emulation:

1. On the PickMaster PowerPac ribbon-tab, click **Operation**.
2. On the **Operation** ribbon-tab, click **Control**.

   The **Control** dialog is opened.

Continues on next page

3. Click **Start** to run the production.

   The emulation starts running.
4. Click **Stop** to stop the emulation.

Note

When running the production, the movement of 3D models in PickMaster PowerPac follows the actual system. However, since the 3D models dimension in PickMaster PowerPac cannot be completely consistent with the real cell. The layout of conveyor, camera, I/O sensor and robot in the emulation may need to be adjusted according to the actual dimension to make the emulation as close to the actual system as possible.

If the item is missing during the emulation, it may be caused by that the size of the PickMaster PowerPac station is not exactly the same with the real station. The item is hidden in the conveyor model.

Adjust the height of the conveyor model to show the item normally.

4.4.11 Managing the robot in production

Starting production

Start and stop the production from the **Control** menu.

During production, the robots are accessed from the **Control** tab in the **Workspace** area. For more details, see *Production on page 319*.

Prerequisites

The solution must be configured to start production.

The recipe 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.
- Total number of picks since the production was started.

Robot states

The robot can be in different states.

State	Color	Description
Running	Green	The robot can pick and place items.
Paused	Red	The robot is paused in motors off state, or the RAPID program has stopped.
Emergency State	Red	The robot is in emergency stop state.
Stopped	Red	The robot is stopped, that is no items are handled by the robot or distributed to the robot.

Stopping and resuming the robot

It is possible to 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.

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 PowerPac and Runtime.
2. Fix the problem.

Continues on next page

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.

5 Advanced function

5.1 User script

Introduction

The **User Script** is a software component provided by PickMaster Twin for users to integrate their custom function.

With this function, user can customize the item position generation, adjustment, filter, or distribution according to their own requirements to achieve user-defined picking and placing of items. For example, the **User Script** can be queried for positions instead of using predefined positions. It is also possible for **User Script** objects to adjust item positions generated by vision models in PickMaster PowerPac. Item positions carry some free usage parameters that can be set by the user script. These parameters can later on be accessed in RAPID by the robot that handles the position.

Note

Only qualified personnel should write or modify the script files.

It is the responsibility of the writer to make sure that the cell is safe when running with the script files.

Note

Only native Python 3.9.5 is supported in PickMaster® Twin products.

Any third-party libraries CANNOT be directly referenced in the script.

Tip

Syntax errors will cause the script files fail to run.

With the following way to avoid the syntax errors:

1. Keep to use the same editor for the same script file.
2. It is recommended to use PyCharm or Notepad++ to edit the script files, as they have syntax checking capabilities for Python files.

CAUTION

It is the responsibility of the integrator to implement that local presence is set up in a correct way.

It is the responsibility of the integrator to implement that single point of control is set up in a correct way.

Continues on next page

DANGER

Protect the script carefully if it is used in the production.

Anyone who has access to the script can modify the script directly. This may cause serious danger.

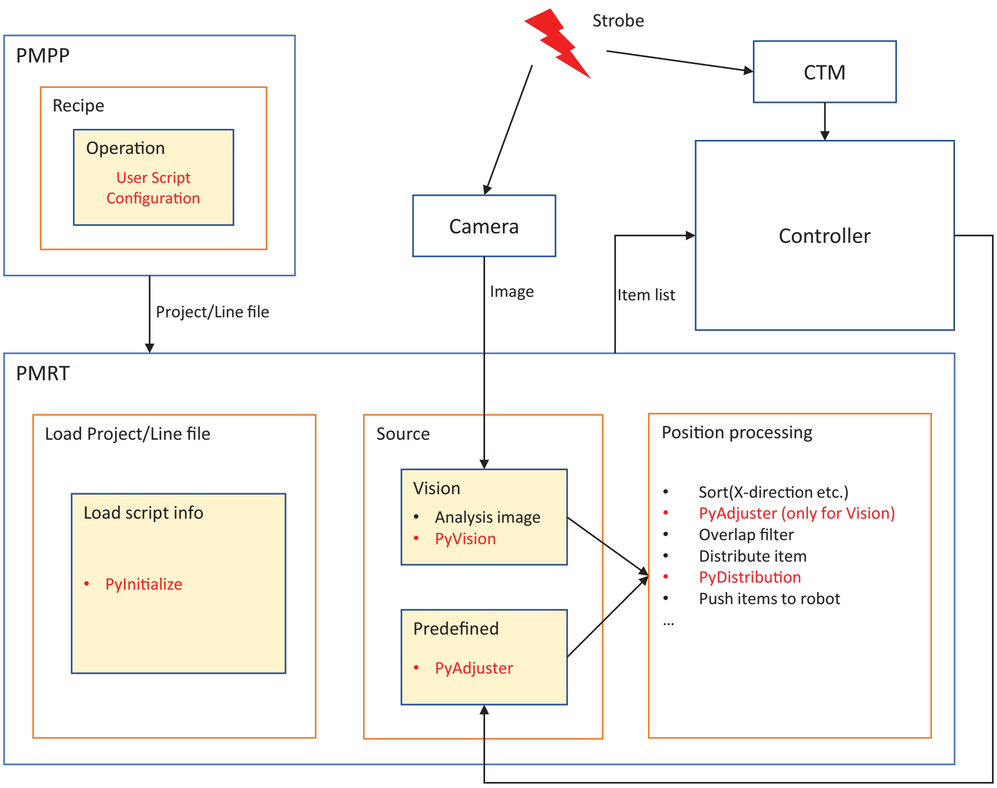
Note

The user script and external sensor cannot be used at the same time in one recipe.

Note

Python script files will not be included in the Pack&Go file. Copy the Python script files to the desired destination.

Flow chart



xx2400000705

Application scenario

User script is an advanced feature provided by PMTW to users, which can be used in the following scenarios:

1 Item generation and deleting

Users can customize the generation and placement of items in the script according to the requirements to meet the actual needs of customers.

Continues on next page

2 Adjust the picking and placing position of items

Users can adjust the position of items statically or in real time in the script according to requirements, meeting users' requirements for real-time adjustment and high precision of material positions.

3 Items filtering and sorting

The user can filter and sort the current items in the script according to the requirements to meet the user's requirements for item screening and capture sequence.

4 Adjustment of item distribution strategy

The current distribution strategies are LoadBalance and ATC. The user can adjust the current distribution strategy to meet the requirements of user-defined distribution.

5 Item identification

When the vision interface is used, the user can further process the pictures taken by the camera and identify new item information.

6 Bind additional information

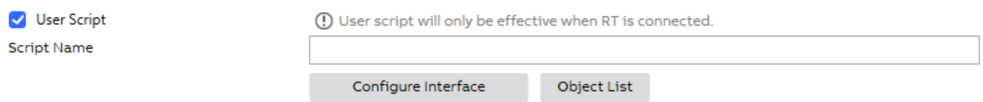
Five optional parameters are provided in the interface parameters. Users can configure optional parameters to bind some additional information with the material and send it to the robot through the software to achieve some special functions, such as item code binding and item tracking.

User value

This function expands the application scenarios of the software. Users can customize the standard functions of the software according to their own needs, which can realize the functions of custom generation, picking, placing, sorting, filtering, and distribution of objects, to meet the needs of users for various specific application scenarios, improve the picking accuracy and production efficiency, and create more value for users.

Configuration overview

When the **User Script** checkbox is selected, the **User Script** setting content will show up.



xx2200001779

	Description
Script Name	Type the predefined script file name with `.py`. **Tip** The predefined script file(s) should be put into *C:\Users\xxxx\Documents\PickMaster\PMScripts* folder before use any script function.

Continues on next page

	Description
Configure Interface	Select which user script interface to be used. Four types user script interfaces are supported by PickMaster Twin.
Object List	Show all available objects (Name and ID) in current operation.

Supported **User Script** interface types overview

PickMaster Twin supports four types of **User Script**.

User script interface	Description
Initialize Interface	This interface is used to provide the user to initialize the User Script program, such as: initialize the parameters, etc. **Tip** **Initialize Interface** will be executed only once when the the **Start** is clicked. The other three interfaces will be executed when DSQC 2000 or DSQC 377 signals are triggered. For more details, see *Initialize Interface PyInitialize: Initialize data on page 328*.
Adjuster Interface	This interface is used to provide the user to realize the customized item position generation and adjustment. Each time the model generates positions, an array with the positions is sent to the **User Script** object. The **User Script** object can then control the positions in any desired way. Positions can be changed, removed, or added. For more details, see *Adjuster Interface PyAdjuster: Modify position on page 329*.
Vision Interface	This interface is used to provide the user to realize the customized item position filter and adjustment by vision result. This interface will be invoked when the Runtime execute to the item recognition section in production. **Tip** The **Vision Interface** can only be used in Production. The other three interfaces can be used in Production and Simulation. For more details, see *Vision Interface PyVision: Recognize items by reanalyzing image on page 330*.
Distribution Interface	This interface is used to provide the user to realize the customized distribution function. This interface will be invoked when the item distribution executes. For more details, see *Distribution Interface PyDistribution: Adjust the target items information after distribution and before push them to robot on page 334*.

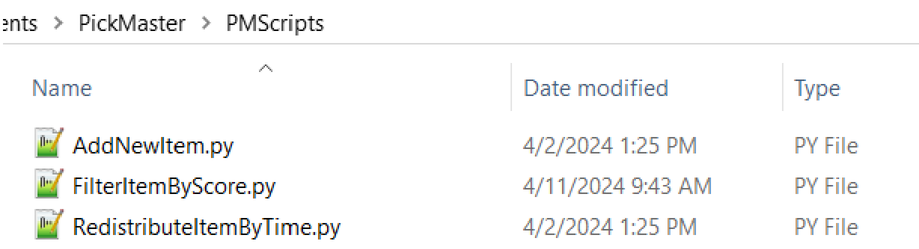
Continues on next page

Configuring the **User Script** function

Follow this procedure to configure the user script function:

1. Put the predefined script files into the destination folder.

   > **Tip**
   >
   > The predefined script file(s) should be put into *C:\Users\xxxx\Documents\PickMaster\PMScripts* folder before use any script function.

   xx2200001784

2. Select the **User Script** checkbox in PickMaster Powerpac **Recipe** setting to open the configuration page.
3. Input the predefined script file name into the **Script Name** text box.
4. Click **Configure Interface** to open the interface type page.
5. On the popped-up page, select the desired interface type.

   > **Tip**
   >
   > The four types can be used at the same time.

6. Click **Done** to finish the user script function setting in PickMaster Powerpac.
7. Set the time out value in Runtime configuration file *PickMasteru.exe.config*.

   For more information about time out setting, see *Time out setting for user script on page 328*.

   > **Tip**
   >
   > The destination folder of the Runtime configuration file *PickMasteru.exe.config*:
   >
   > - VRT: C:\Program Files (x86)\ABB\PickMaster Twin 2\PickMaster Twin Runtime 2\PickMaster VirtualRuntime
   > - RRT: C:\Program Files (x86)\ABB\PickMaster Twin 2\PickMaster Twin Runtime 2\PickMaster Runtime

Continues on next page

Time out setting for user script

Set the execution time limit of user scripts to avoid PickMaster Twin product exceptions caused by excessive execution time of the user scripts.

Template	Key	Value (In the template)	Explanation
`<add key="PyInitializeTimeout" value ="1500"/>`	`PyInitializeTimeout`	**x** (1500)	The timeout of the PyInitialize interface is **x** (1500) ms. When the executing time exceeds the set value, an warning will display in the log view.
`<add key="PyAdjusterTimeout" value ="1500"/>`	`PyAdjusterTimeout`	**x** (1500)	The timeout of the PyAdjuster interface is **x** (1500) ms.
`<add key="PyVisionTimeout" value ="1500"/>`	`PyVisionTimeout`	**x** (1500)	The timeout of the PyVision interface is **x** (1500) ms.
`<add key="PyDistributionTimeout" value ="1500"/>`	`PyDistributionTimeout`	**x** (1500)	The timeout of the PyDistribution interface is **x** (1500) ms.
`<add key="MaxTimeoutCount" value ="5"/>`	`MaxTimeoutCount`	**x** (5)	The maximum consecutive timeouts of each interface is **x** (5) times. When the number of consecutive timeouts exceeds the set maximum value, Runtime will stop the interface function calling, clear all objects and display the error log to notify the user to stop the station and check the script.

User script interface

Initialize Interface PyInitialize: Initialize data

This interface is used to initialize the script, and transfer current RT information, item information, container information, and workarea information to the user script, which can be processed by the user, such as creating a new item. At the same time, users can add user program initialization operations in this interface, such as starting external programs, etc., which can be started at the same time when starting the station.

Argument	Description	Explanation	In the example:
type	Runtime type	• 0:VRT • 1:RRT	

itemInfo	Item information which contains `{Key}:{Name:{} Id:{}}`
	For example:
	`itemInfo=`
	`{`
	`'0':{'Name':Item_1, 'Id':'[illegible]'},`
	`};`
		`{Key}`
		Key: unique index
			`'0'`

		`Name:{}` Name: name of the item	`'Name':Item_1`
		`Id:{}` Id: ID of the item	`'Id':'[illegible]'`

For more example, see *`AddNewItem.py` on page 336* and *`RedistributeItemByTime.py` on page 340*.

Continues on next page

Adjuster Interface PyAdjuster: Modify position

Argument	Description	Explanation	In the example:

items	Item information, which contains `Time:{} {Key}:{X:{} Y:{} Z:{} RX:{} RY:{} RZ:{} Tag:{} Val1:{} Val2:{} Val3:{} Val4:{} Val5:{} Level:{} Id:{}}`.
	For example:
	`items =`
	`{`
	`'Time': 1666849507.969,`
	`'0': {'X': 0.0,`
	`'Y': 150.0,`
	`'Z': 0.0,`
	`'RX': 0.0,`
	`'RY': 0.0,`
	`'RZ': 0.0,`
	`'Tag': 0,`
	`'Val1': 0.0,`
	`'Val2': 0.0,`
	`'Val3': 0.0,`
	`'Val4': 0.0,`
	`'Val5': 0.0,`
	`'Level': 2,`
	`'Id': '35139a6c-56a8-437d-b180-7f40e49bf6ff'`
	`}`
	`}`
		`Time:{}`
		Time: time stamp(s), get the number of milliseconds since 1 Jan 1970
			`'Time': 1666849507.969,`

		`{Key}` Key: unique index	`'0'`
		`X:{}` X: the location value of the item in X direction	`'X': 0.0`
		`Y:{}` Y: the location value of the item in Y direction	`'Y': 150.0`
		`Z:{}` Z: the location value of the item in Z direction	`'Z': 0.0`
		`RX:{}` RX: the rotation angle value of the item in X direction	`'RX': 0.0`
		`RY:{}` RY: the rotation angle value of the item in Y direction	`'RY': 0.0`
		`RZ:{}` RZ: the rotation angle value of the item in Z direction	`'RZ': 0.0`
		`Tag:{}` Tag: used in rapid	`'Tag': 0`
		`Val1:{}` Val1, Val2, Val3, Val4, Val5: optional value, used in rapid	`'Val1': 0.0`
		`Val2:{}` Val2: optional value, used in rapid	`'Val2': 0.0`
		`Val3:{}` Val3: optional value, used in rapid	`'Val3': 0.0`
		`Val4:{}` Val4: optional value, used in rapid	`'Val4': 0.0`
		`Val5:{}` Val5: optional value, used in rapid	`'Val5': 0.0`
		`Level:{}` Level: inspection level	`'Level': 2`

Continues on next page

Argument	Description	Explanation	In the example:
		• 0: Discarded • 1: Rejected • 2: Accepted	

		`Id:{}`
		Id: ID of the item
			`'Id':`
			`'35139a6c-56a8-437d-b180-7f40e49bf6ff'`
			`}`

For more example, see *AddNewItem.py on page 336*.

Vision Interface PyVision: Recognize items by reanalyzing image

Argument	Description	Explanation	

im-ageData	Image data, which contains `Width:{}` `Height:{}` `IsColor:{}` `Grey:{}` `Blue:{}` `Green:{}` `Red:{}`
	For example:
	`Grey image`
	`imageData =`
	`{`
	`'Width': 481,`
	`'Height’: 409,`
	`'IsColor’: 0,`
	`‘Grey’: [56,…,67]`
	`}`
	`Colorful image`
	`imageData =`
	`{`
	`'Width': 481,`
	`'Height’: 409,`
	`'IsColor’: 1,`
	`‘Blue’: [56,…,67],`
	`‘Green': [56,…,67],`
	`‘Red': [56,…,67]`
	`}`
		`Width:{}`
		• Width: image width in pixel
			`'Width': 481,`

		`Height:{}` • Height: image height in pixel	`'Height’: 409,`

		`IsColor:{}`
		• `Grey:{}`
		• `Blue:{}` `Green:{}` `Red:{}`.
		• IsColor:
		  - 0: Grey image
		  - 1: Colorful image
		• Grey: grey data, valid from 0 to 255
		• Blue: blue data, valid from 0 to 255
		• Green: green data, valid from 0 to 255
		• Red: red data, valid from 0 to 255
			For `Grey image`
			`'IsColor’: 0,`
			`‘Grey’: [56,…,67]`
			For `Colorful image`
			`'IsColor’: 1,`
			`‘Blue’: [56,…,67],`
			`‘Green': [56,…,67],`
			`‘Red': [56,…,67]`

Continues on next page

Argument	Description	Explanation	

calibData	Calibration data, which contains `UpperLeftX:{}` `UpperLeftY:{}` `LowerRightX:{}` `LowerRightY:{}` `XScale:{} YScale:{}`.
	For example:
	`calibData =`
	`{`
	`'UpperLeftX’: -313,`
	`'UpperLeftY’: -265,`
	`'LowerRightX’: 168,`
	`‘LowerRightY’: 144,`
	`‘XScale’: 0.415,`
	`‘YScale': 0.415`
	`}`
		`UpperLeftX:{}`
		• UpperLeftX: the upper left point on the X direction in the coordinate system in pixel
			`'UpperLeftX’: -313,`

		`UpperLeftY:{}` • UpperLeftY: the upper left point on the Y direction in the coordinate system in pixel	`'UpperLeftY’: -265,`
		`LowerRightX:{}` • LowerRightX: the lower right point on the X direction in the coordinate system in pixel	`'LowerRightX’: 168,`
		`LowerRightY:{}` • LowerRightY: the lower right point on the Y direction in the coordinate system in pixel	`‘LowerRightY’: 144,`
		`XScale:{}` • XScale: X axial scale of real item and image in pixel	`‘XScale’: 0.415,`
		`YScale:{}` • YScale: Y axial scale of real item and image in pixel.	`‘YScale': 0.415`

Continues on next page

Argument	Description	Explanation	
items	Item information, which contains: `Time:{}` and • Geomatric: `{Key}:{X:{} Y:{} Z:{} RZ:{} SortValue:{} ZValid:{} XImgPos:{} YImgPos:{} Val1:{} Val2:{} Val3:{} Val4:{} Val5:{} Level:{} Id:{} ModelType:{} Score:{} XScale:{} YScale:{} Contrast:{} FitError:{} Coverage:{} Clutter:{}}` • Blob: `{Key}:{X:{} Y:{} Z:{} RZ:{} SortValue:{} ZValid:{} XImgPos:{} YImgPos:{} Val1:{} Val2:{} Val3:{} Val4:{} Val5:{} Level:{} Id:{} ModelType:{} Area:{} Perimeter:{} Elongation:{} Circularity:{}}` • Inspection: `{Key}:{X:{} Y:{} Z:{} RZ:{} SortValue:{} ZValid:{} XImgPos:{} YImgPos:{} Val1:{} Val2:{} Val3:{} Val4:{} Val5:{} Level:{} Id:{} ModelType:{}}` For example: Geomatric	`Time:{}` • Time: time stamp(s), get the number of milli-seconds since 1 Jan 1970	`'Time': 1666849507.969,`
		`{Key}` Key: unique index	`'0’`
		`X:{}` • X: the location value of the item in X direction	`'X’: -80.1,`
		`Y:{}` • Y: the location value of the item in Y direction	`'Y’: -77.2,`
		`Z:{}` • Z: the location value of the item in Z direction	`'Z': 0.0,`
		`RZ:{}` • RZ: the rotation angle value of the item in Z direction	`‘RZ’: -7.22,`
		`SortValue:{}` • SortValue: sort value	`‘SortValue’: 0.976,`
		`ZValid:{}` • ZValid: - 1: valid - 0: invalid	`‘ZValid': 0,`
		`XImgPos:{}` • XImgPos: item position in image on X direction •	`‘XImgPos': -80.1,`
		`YImgPos:{}` • YImgPos: item position in image on Y direction	`‘YImgPos': -77.2,`

		`Val1:{} Val2:{} Val3:{} Val4:{} Val5:{}`
		• Val1, Val2, Val3, Val4, Val5: optional value, used in rapid
			`'Val1': 0.0,`
			`'Val2': 0.0,`
			`'Val3': 0.0,`
			`'Val4': 0.0,`
			`'Val5': 0.0,`

		`Level:{}` • Level: inspection level - 0: Discarded - 1: Rejected - 2: Accepted	`'Level':2,`

Continues on next page

<table>
<tr><th>Argument</th><th>Description</th><th>Explanation</th><th></th></tr>
<tr><td rowspan="10"></td><td rowspan="10">resResult = {
'Time':
1666849507.969,
'0’:{'X’: -80.1,
'Y’: -77.2,
'Z': 0.0,
‘RZ’: -7.22,
‘SortValue’: 0.976,
‘ZValid': 0,
‘XImgPos': -80.1,
‘YImgPos': -77.2,
'Val1': 0.0,
'Val2': 0.0,
'Val3': 0.0,
'Val4': 0.0,
'Val5': 0.0,
'Level':2,
'Id':
'35139a6c-56a8-437d-b180-7f40e49bf6ff’,
'ModelType':1,
'Score':0.747174859046936,
'XScale':0.9995959997177124,
'YScale':0.9995959997177124,
'Contrast':12.289325714111328,
'FitError':0.36996814608573914,
'Coverage':0.747174859046936,
'Clutter':0.10466811060905457
}
}
Blob
resResult = {
'Time':
1666849507.969,
'0’:{'X’: -80.1,
'Y’: -77.2,
'Z': 0.0,
‘RZ’: -7.22,
‘SortValue’: 0.976,
‘ZValid': 0,
‘XImgPos': -80.1,
‘YImgPos': -77.2,
'Val1': 0.0,
'Val2': 0.0,
'Val3': 0.0,
'Val4': 0.0,
'Val5': 0.0,
'Level':2,
'Id':
'35139a6c-56a8-437d-b180-7f40e49bf6ff’,
'ModelType':2,</td><td>Id:{}
• Id: ID of the item</td><td>'Id':
'35139a6c-56a8-437d-b180-7f40e49bf6ff’,</td></tr>
<tr><td>ModelType:{}
• ModelType:
- 1: Geomatric
- 2: Blob
- 3: Inspection</td><td>For Geomatric:
'ModelType':1,
For Blob:
'ModelType':2,
For Inspection:
'ModelType':3 }</td></tr>
<tr><td>For more information, see Configuring a geometric model with PatMax on page 286.
Score:{}
• Score: how closely the found item matches the trained model.</td><td>'Score':0.747174859046936,</td></tr>
<tr><td>XScale:{}
• XScale: X axial scale of real item and image in pixel</td><td>'XScale':0.9995959997177124,</td></tr>
<tr><td>YScale:{}
• YScale: Y axial scale of real item and image in pixel.</td><td>'YScale':0.9995959997177124,</td></tr>
<tr><td>Contrast:{}
• Contrast: the image contrast of each item that is found in the image.</td><td>'Contrast':12.289325714111328,</td></tr>
<tr><td>FitError:{}
• FitError: a measure of the variance between the shape of the trained pattern and the shape of the pattern found in the search image.</td><td>'FitError':0.36996814608573914,</td></tr>
<tr><td>Coverage:{}
• Coverage: a measure of the extent to which all parts of the trained pattern are also present in the search image.</td><td>'Coverage':0.747174859046936,</td></tr>
<tr><td>Clutter:{}
• Clutter: a measure of the extent to which the found pattern contains features that are not present in the trained pattern.</td><td>'Clutter':0.10466811060905457
}</td></tr>
<tr><td></td><td>'Area':0,</td></tr>
</table>

Continues on next page

Argument	Description	Explanation	

	'Area':0,
	'Perimeter':0,
	'Elongation':0,
	'Circularity':0 }
	}
	Inspection
	resResult = {
	'Time':
	1666849507.969,
	'0':{'X': -80.1,
	'Y': -77.2,
	'Z': 0.0,
	'RZ': -7.22,
	'SortValue': 0.976,
	'ZValid': 0,
	'XImgPos': -80.1,
	'YImgPos': -77.2,
	'Val1': 0.0,
	'Val2': 0.0,
	'Val3': 0.0,
	'Val4': 0.0,
	'Val5': 0.0,
	'Level':2,
	'Id':
	'35139a6c-56a8-437d-b180-7f40e49bf6ff',
	'ModelType':3 }
	}
		For more information, see *Configuring blob models on page 294*.
		Area:{}
		• Area: expressed in mm^2	

		Perimeter:{} • Perimeter: expressed in mm	'Perimeter':0,
		Elongation:{} • Elongation: 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.	'Elongation':0,
		Circularity:{} • Circularity: defines the circularity. A value of 1 means perfectly circular and completely filled (no holes).	'Circularity':0 }

For more example, see *FilterItemByScore.py on page 338*.

Distribution Interface PyDistribution: Adjust the target items information after distribution and before push them to robot

Argument	Description		Explain
WaId	Workarea ID, which contains WaId:{}. For example: WaId = (9B336B0C-2B65-4054-92A5-30C36D64B667)	WaId:{} • WaId: Workarea Id	WaId = (9B336B0C-2B65-4054-92A5-30C36D64B667)

Continues on next page

Argument	Description		Explain

items	Item information, which contains `Time:{} {Key}: {X:{} Y:{} Z:{} q1:{} q2:{} q3:{} q4:{} Val1:{} Val2:{} Val3:{} Val4:{} Val5:{} Type:{} Tag:{} Index:{} State:{} Container:{} Layer:{} Group:{} Id:{}}`.
	For example:
	`items =`
	`{`
	`'Time': 1666849507.969,`
	`'0':{'X': 0.0,`
	`'Y': 150.0,`
	`'Z': 0.0,`
	`'q1': 0.0,`
	`'q2': 1.0,`
	`'q3': 0.0,`
	`'q4': 0.0,`
	`'Val1': 0.0,`
	`'Val2': 0.0,`
	`'Val3': 0.0,`
	`'Val4': 0.0,`
	`'Val5': 0.0,`
	`'Type': 2,`
	`'Tag': 0,`
	`'Index': 2,`
	`'State': 0,`
	`'Container': 1,`
	`'Layer': 1,`
	`'Group': 0,`
	`'Id': '35139a6c-56a8-437d-b180-7f40e49bf6ff'`
	`}`
	`}`
		`Time:{}`
		• Time: time stamp(s), get the number of milli-seconds since 1 Jan 1970
			`'Time': 1666849507.969,`

		`{Key}` Key: unique index	`'0'`
		`X:{}` • X: the location value of the item in X direction	`{'X': 0.0,`
		`Y:{}` • Y: the location value of the item in Y direction	`'Y': 150.0,`
		`Z:{}` • Z: the location value of the item in Z direction	`'Z': 0.0,`
		`q1:{}` • q1, q2, q3, q4: the quaternion values of the item	`'q1': 0.0,`
		`q2:{}`	`'q2': 1.0,`
		`q3:{}`	`'q3': 0.0,`
		`q4:{}`	`'q4': 0.0,`

		`Val1:{} Val2:{} Val3:{} Val4:{} Val5:{}`
		• Val1, Val2, Val3, Val4, Val5: optional value, used in rapid
			`'Val1': 0.0,`
			`'Val2': 0.0,`
			`'Val3': 0.0,`
			`'Val4': 0.0,`
			`'Val5': 0.0,`

		`Type:{}` • Index: Index number of the **Accepted Type** or **Rejected Type**	`'Type': 2,`
		`Tag:{}` • Tag: Used in rapid	`'Tag': 0,`
		`Index:{}` • Index: The sequence number of the current item, which increases with the generation of the item on the conveyor and predefined layout in the container.	`'Index': 2,`
		`State:{}`	`'State': 0,`

Continues on next page

Argument	Description		Explain
		• State: item state - 0: Use - 1: Bypass - 2: Used	
		`Container:{}` • Container: container number - 0: it is an item - 1-n: it is a container	`'Container': 1,`
		`Layer:{}` • Layer: layer number - 0: it is an item - 1-n: it is layer in the container	`'Layer': 1,`
		`Group:{}` • Group: sorting method - 0: None or movement direction - 1: Strict	`'Group': 0,`

		`Id:{}`
		• Id: ID of the item
			`'Id':`
			'35139a6c-56a8-437d-b180-7f40e49bf6ff'
			`}`

For more example, see *`RedistributeItemByTime.py` on page 340*.

Template

All the user script templates are also provided in the folder *C:\Program Files (x86)\ABB\PickMaster Twin 2\PickMaster Twin Client 2\PickMaster PowerPac\Template* when PickMaster Client is installed.

`AddNewItem.py`

```
# PMTW user script demo -- AddNewItem
# Add a new item in the default item list.

# Global definition
RTType = 1
item_1 = r''
container_1 = r''
newObject = {'X': 100.0, 'Y': 50.0, 'Z': 5.0, 'RX': 0.0, 'RY': 0.0,
      'RZ': 0.0, 'Tag': -1, 'Val1': 0.0, 'Val2': 0.0, 'Val3': 0.0,
      'Val4': 0.0, 'Val5': 0.0, 'Level': 2, 'Id':
      '5FAD0398-74F8-4786-BF2A-1225924A8A41'}
# This path need to be created by users.
logPath = r'C:\PMScriptsLog\PickInfo.txt'

# PyInitialize interface
```

Continues on next page

```
def PyInitialize(type, itemInfo):
  global RTType
  global item_1
  global container_1
  RTType = type
  f = open(logPath,'a')
  f.write("PyInitialize\n")
  # RT type
  strLine = "RTType:{}\n".format(str(RTType))
  f.write(strLine)
  # Item information
  keys = itemInfo.keys();
  for key in keys:
    strLine = "{} Name:{} Id:{}\n".format(str(key),
         str(itemInfo[key]['Name']), str(itemInfo[key]['Id']))
    if itemInfo[key]['Name'] == 'Item_1':
      item_1 = itemInfo[key]['Id']
    elif itemInfo[key]['Name'] == 'Container_1':
      container_1 = itemInfo[key]['Id']
    f.write(strLine)
  f.close()
# PyAdjuster interface
def PyAdjuster(items):
  global RTType
  global item_1
  global container_1
  global newObject
  f = open(logPath, 'a')
  f.write("PyAdjuster\n")
  # Modify Id
  newObject['Id'] = item_1
  #newObject['Id'] = container_1

  # Add new item
  iSize = len(items)
  newKey = str(iSize - 1)
  items[newKey] = newObject

  # Item information
  keys = items.keys()
  for key in keys:
    if key == 'Time':
      # Time stamp(s), get the number of milliseconds since 1 Jan
           1970.
      strLine = "Time:{}\n".format(str(items[key]))
      f.write(strLine)
    else:
      # Print
      strLine = "{} X:{} Y:{} Z:{} RX:{} RY:{} RZ:{} Tag:{} Val1:{}
           Val2:{} Val3:{} Val4:{} Val5:{} Level:{}
           Id:{}\n".format(str(key), str(items[key]['X']),
```

Continues on next page

```
                str(items[key]['Y']), str(items[key]['Z']),
                str(items[key]['RX']), str(items[key]['RY']),
                str(items[key]['RZ']), str(items[key]['Tag']),
                str(items[key]['Val1']), str(items[key]['Val2']),
                str(items[key]['Val3']), str(items[key]['Val4']),
                str(items[key]['Val5']), str(items[key]['Level']),
                items[key]['Id'])
      f.write(strLine)
  f.close()
  return items;
```

FilterItemByScore.py

```
# PMTW user script demo -- FilterItemByScore
# Filter item according to score value.

# Global definition
# This path need to be created by users.
logPath = r'C:\PMScriptsLog\PlaceInfo.txt'

# PyVision interface
def PyVision(imageData,calibData,items):
  f = open(logPath, 'a')
  f.write("PyVision\n")
  # Image data
  f.write("ImageData:\n")
  strLine = "Width:{} Height:{}
       IsColor:{}\n".format(str(imageData['Width']),
       str(imageData['Height']), str(imageData['IsColor']))
  f.write(strLine)
  if imageData['IsColor'] == 0:
    strLine = "Grey:{}\n".format(str(imageData['Grey']))
  else:
    strLine =
         "Blue:{}\nGreen:{}\nRed:{}\n".format(str(imageData['Blue']),
         str(imageData['Green']), str(imageData['Red']))
  f.write(strLine)
  # Calibration data
  f.write("CalibrationData:\n")
  strLine = "UpperLeftX:{} UpperLeftY:{} LowerRightX:{}
       LowerRightY:{} XScale:{}
       YScale:{}\n".format(str(calibData['UpperLeftX']),
       str(calibData['UpperLeftY']), str(calibData['LowerRightX']),
       str(calibData['LowerRightY']), str(calibData['XScale']),
       str(calibData['YScale']))
  f.write(strLine)
  # Item information
  f.write("Items:\n")
  keys = items.keys();
  for key in keys:
    if key == 'Time':
      # Time stamp(s), get the number of milliseconds since 1 Jan
           1970.
```

Continues on next page

```
    strLine = "Time:{}\n".format(str(items[key]))
    f.write(strLine)
  else:
    if items[key]['ModelType'] == 1:
      # Geometric
      strLine = "{} X:{} Y:{} Z:{} RZ:{} SortValue:{} ZValid:{}
           XImgPos:{} YImgPos:{} Val1:{} Val2:{} Val3:{} Val4:{}
           Val5:{} Level:{} Id:{} ModelType:{} Score:{} XScale:{}
           YScale:{} Contrast:{} FitError:{} Coverage:{}
           Clutter:{}\n".format(str(key), str(items[key]['X']),
           str(items[key]['Y']), str(items[key]['Z']),
           str(items[key]['RZ']), str(items[key]['SortValue']),
           str(items[key]['ZValid']), str(items[key]['XImgPos']),
           str(items[key]['YImgPos']), str(items[key]['Val1']),
           str(items[key]['Val2']), str(items[key]['Val3']),
           str(items[key]['Val4']), str(items[key]['Val5']),
           str(items[key]['Level']), items[key]['Id'],
           str(items[key]['ModelType']),
           str(items[key]['Score']), str(items[key]['XScale']),
           str(items[key]['YScale']),
           str(items[key]['Contrast']),
           str(items[key]['FitError']),
           str(items[key]['Coverage']),
           str(items[key]['Clutter']))
      # Filter
      if items[key]['Score'] < 0.8:
        items[key]['Level'] = 0
    elif items[key]['ModelType'] == 2:
      # Blob
      strLine = "{} X:{} Y:{} Z:{} RZ:{} SortValue:{} ZValid:{}
           XImgPos:{} YImgPos:{} Val1:{} Val2:{} Val3:{} Val4:{}
           Val5:{} Level:{} Id:{} ModelType:{} Area:{}
           Perimeter:{} Elongation:{}
           Circularity:{}\n".format(str(key),
           str(items[key]['X']), str(items[key]['Y']),
           str(items[key]['Z']), str(items[key]['RZ']),
           str(items[key]['SortValue']),
           str(items[key]['ZValid']), str(items[key]['XImgPos']),
           str(items[key]['YImgPos']), str(items[key]['Val1']),
           str(items[key]['Val2']), str(items[key]['Val3']),
           str(items[key]['Val4']), str(items[key]['Val5']),
           str(items[key]['Level']), items[key]['Id'],
           str(items[key]['ModelType']), str(items[key]['Area']),
           str(items[key]['Perimeter']),
           str(items[key]['Elongation']),
           str(items[key]['Circularity']))
      # Filter
      if items[key]['Score'] < 0.8:
        items[key]['Level'] = 0
    else:
      # Inspection
      strLine = "{} X:{} Y:{} Z:{} RZ:{} SortValue:{} ZValid:{}
           XImgPos:{} YImgPos:{} Val1:{} Val2:{} Val3:{} Val4:{}
           Val5:{} Level:{} Id:{}
           ModelType:{}\n".format(str(key), str(items[key]['X']),
```

Continues on next page

```
                str(items[key]['Y']), str(items[key]['Z']),
                str(items[key]['RZ']), str(items[key]['SortValue']),
                str(items[key]['ZValid']), str(items[key]['XImgPos']),
                str(items[key]['YImgPos']), str(items[key]['Val1']),
                str(items[key]['Val2']), str(items[key]['Val3']),
                str(items[key]['Val4']), str(items[key]['Val5']),
                str(items[key]['Level']), items[key]['Id'],
                str(items[key]['ModelType']))
        f.write(strLine)
    f.close()
    return items;
```

RedistributeItemByTime.py

```
# PMTW user script demo -- RedistributeItemByTime
# Every minute a robot is exchanged to pick and place items.

import time

# Global definition
RTType = 1
item_1 = r''
item_2 = r''
workarea_2 = r''
workarea_4 = r''
# This path need to be created by users.
logPath = r'C:\PMScriptsLog\PlaceInfo.txt'

# PyInitialize interface
def PyInitialize(type, itemInfo):
  global RTType
  global item_1
  global item_2
  global workarea_2
  global workarea_4
  RTType = type
  f = open(logPath,'a')
  f.write("PyInitialize\n")
  # RT type
  strLine = "RTType:{}\n".format(str(RTType))
  f.write(strLine)
  # Item information
  keys = itemInfo.keys()
  for key in keys:
    strLine = "{} Name:{} Id:{}\n".format(str(key),
         str(itemInfo[key]['Name']), str(itemInfo[key]['Id']))
    if itemInfo[key]['Name'] == 'Item_1':
      item_1 = itemInfo[key]['Id']
    elif itemInfo[key]['Name'] == 'Item_2':
      item_2 = itemInfo[key]['Id']
    elif itemInfo[key]['Name'] == 'ConveyorWorkArea_2':
      workarea_2 = itemInfo[key]['Id']
```

Continues on next page

```
      elif itemInfo[key]['Name'] == 'ConveyorWorkArea_4':
        workarea_4 = itemInfo[key]['Id']
      f.write(strLine)
    f.close()
  # PyDistribution interface
  def PyDistribution(WaId, items):
    global RTType
    global item_1
    global item_2
    global workarea_2
    global workarea_4
    f = open(logPath, 'a')
    f.write("PyDistribution\n")
    # Workarea Id
    strLine = "WaId:{}\n".format(WaId)
    f.write(strLine)
    # Item information
    keys = items.keys()
    for key in keys:
      if key == 'Time':
        # Time stamp(s), get the number of milliseconds since 1 Jan
             1970.
        strLine = "Time:{}\n".format(str(items[key]))
        f.write(strLine)
      else:
        # Modify before
        strLine = "{} X:{} Y:{} Z:{} q1:{} q2:{} q3:{} q4:{} Tag:{}
             Val1:{} Val2:{} Val3:{} Val4:{} Val5:{} Type:{} Index:{}
             State:{} Container:{} Layer:{} Group:{}
             Id:{}\n".format(str(key), str(items[key]['X']),
             str(items[key]['Y']), str(items[key]['Z']),
             str(items[key]['q1']), str(items[key]['q2']),
             str(items[key]['q3']), str(items[key]['q4']),
             str(items[key]['Tag']), str(items[key]['Val1']),
             str(items[key]['Val2']), str(items[key]['Val3']),
             str(items[key]['Val4']), str(items[key]['Val5']),
             str(items[key]['Type']), str(items[key]['Index']),
             str(items[key]['State']), str(items[key]['Container']),
             str(items[key]['Layer']), str(items[key]['Group']),
             items[key]['Id'])
        f.write(strLine)
        # Modify
        if (divmod(time.localtime().tm_min, 2))[1] == 0:
          if WaId == workarea_2:
            items[key]['State'] = 1
          elif WaId == workarea_4:
            items[key]['State'] = 0
        else:
          if WaId == workarea_2:
            items[key]['State'] = 0
          elif WaId == workarea_4:
            items[key]['State'] = 1
```

Continues on next page

```
        # Modify after
        strLine = "{} X:{} Y:{} Z:{} q1:{} q2:{} q3:{} q4:{} Tag:{}
             Val1:{} Val2:{} Val3:{} Val4:{} Val5:{} Type:{} Index:{}
             State:{} Container:{} Layer:{} Group:{}
             Id:{}\n".format(str(key), str(items[key]['X']),
             str(items[key]['Y']), str(items[key]['Z']),
             str(items[key]['q1']), str(items[key]['q2']),
             str(items[key]['q3']), str(items[key]['q4']),
             str(items[key]['Tag']), str(items[key]['Val1']),
             str(items[key]['Val2']), str(items[key]['Val3']),
             str(items[key]['Val4']), str(items[key]['Val5']),
             str(items[key]['Type']), str(items[key]['Index']),
             str(items[key]['State']), str(items[key]['Container']),
             str(items[key]['Layer']), str(items[key]['Group']),
             items[key]['Id'])
        f.write(strLine)
    f.close()
    return items;
```

5.2 External sensor

External sensor

An external sensor is a software component that gives the user 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. PMTW supports to use Python to implement external sensors programs.

Note

Only qualified personnel should write or modify the script files.

It is the responsibility of the writer to make sure that the cell is safe when running with the script files.

Note

Only native Python 3.9.5 is supported in PickMaster® Twin products.

Any third-party libraries CANNOT be directly referenced in the script.

Tip

Syntax errors will cause the script files fail to run.

With the following way to avoid the syntax errors:

1. Keep to use the same editor for the same script file.
2. It is recommended to use PyCharm or Notepad++ to edit the script files, as they have syntax checking capabilities for Python files.

CAUTION

It is the responsibility of the integrator to implement that local presence is set up in a correct way.

It is the responsibility of the integrator to implement that single point of control is set up in a correct way.

DANGER

Protect the script carefully if it is used in the production.

Anyone who has access to the script can modify the script directly. This may cause serious danger.

Tip

If an external sensor is used on the conveyor, Flow function will be disabled.

Continues on next page

> **Note**
>
> The user script and external sensor cannot be used at the same time in one recipe.

> **Note**
>
> Python script files will not be included in the Pack&Go file. Copy the Python script files to the desired destination.

> **Tip**
>
> If an indexed work area is used, external sensor function will be disabled.

Right-click on an **External Sensors** in the tree view and select **Configuration** to configure the external sensor.

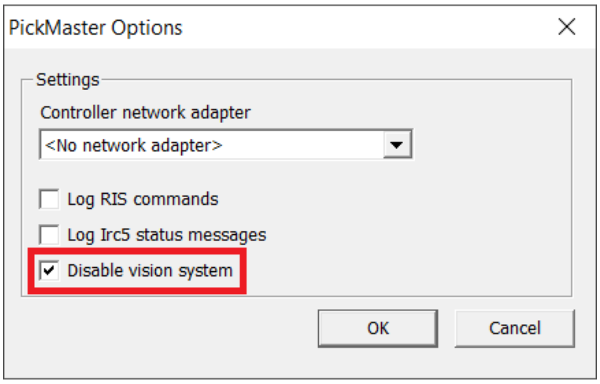
The following table provides details about the **External Sensor Configuration** dialog box.

	Description
Script Name	Type the predefined script file name with `.py`. It's recommended to use the name `ExternalSensors.py` as template. **Tip** The predefined script file(s) should be put into *C:\Users\xxxx\Documents\PickMaster\PMScripts* folder before use any script function.
Configure in user program	When click on this button, it will refer to the Python interface `def configureSensor(self, sensorId)`. Users should self-define the configuration behavior in this interface in their own Python class. When users click this button, the interface will be called.
Get sensor information	When click on this button, it will refer to the Python interface `def getSensorInfo(self)` which is provided by PMTW in the file `ExternalSensorInterface.py`. In this interface, the information of user program, such as name, author, version and description will be sent back to and shown in the **External Sensor Configuration** dialog. The information of user program can be modified directly in Python code or in `configureSensor` interface in the file `ExternalSensors.py`. **Note** The file `ExternalSensorInterface.py` is not allowed to be modified.

If a Basler camera is is used through the external sensor interface, it is necessary to turn off PickMaster's internal vision system to avoid that both systems try to connect to the same camera.

Continues on next page

Turn off the internal vision system in real Runtime.



xx2300000904

Creating a new external sensor main file

Note

All the contents in file `ExternalSensor.py` should inherit the predefined basic classes from the file `ExternalSensorInterface.py`.

Predefined **External Sensor** interface classes

PickMaster Twin predefine four classes for **External Sensor** in `ExternalSensorInterface.py` file.

Except these four classes, a tool class `StoppableThread` is provided which is inherited from the `threading.Thread` class of native Python and has the function of stop thread.

Class	Description

`SensorInfo`	This interface is used to provide the user to initialize the Python program, such as: initialize all basic information and settings.
	• `def registerLogCallback(self, logCallBackFunc)` : users can use the `logCallBackFunc` to show python log in PMTW with the following format:
	`log = {'LogLevel': 0,'Log': 'python log string'}`
	`self.fLogCallback.ShowPythonLog(log)`
	There are four types of the log levels:
	- 0-Status
	- 1-Warning
	- 2-Error
	- 3-Debug
	Tip
	Status, warning and error information will be displayed in Runtime log view.
	Debug information can only be displayed in the Debug view.

Continues on next page

Class	Description
`SensorConfig`	This interface is used to provide the user to realize the operations about the sensor. • `def configureSensor(self, sensorId)`: the content should be implemented by users in their own class. • `def getSensorInfo(self)`: provided by PMTW to get the information of name, author, version and description of user program. This interface is called when the **Get sensor information** button is clicked. • `def loadSensor(self, sensorId, configurationInfo)`: provided by PMTW to load the previously saved configuration information string to the Python dictionary data `sensorConfigurationDict[sensorId]` in the user program to update their last settings. This interface is called when the Python environment is in initialization and the sensor has already been configured previously. • `def saveSensor(self, sensorId)`: provided by PMTW to get the latest configuration information string including possible modification in `configureSensor`. This interface is called when the **OK** button is clicked in the sensor configuration dialog or **Save** button is clicked in **PositionGenerator** dialog.
`PositionGenerator`	This interface is used to provide the user to realize the customized sensor position generation. • `def initializePosGenRelatedMap(self, sensorId, posGenId, objectName)`: provided by PMTW to initialize the relationship between sensor, position generator and object(item/container). Users could then use `objectName (item/container name)` in their Python logic to help distinguish different position generators. This interface is called when the Python environment is in initialization and one position generator is turned from configuration-enabled state into save-enabled state. • `def configurePosGen(self, posGenId)`: the content should be implemented by users in their own class. • `def loadPosGen(self, posGenId, positionGeneratorInfo)`: provided by PMTW to load the previously saved position generator configuration information string to the Python dictionary data `posGenConfigurationDict[posGenId]` in the user program to update their last settings. This interface is called when one existing position generator is turned from configuration-enabled state into save-enabled state. • `def savePosGen(self, posGenId)`: provided by PMTW to get the latest position generator configuration information string including possible modification in configurePosGen. This interface is called when the “Save” button is clicked, or when the “OK” button is clicked and one position generator is in save-enabled state in the item/container source type view.

Continues on next page

Class	Description
`SensorRuntime`	This interface is used to provide the user to realize the customized sensor operations during the production. A flag mechanism is provided to handle the finishing logic of `startSensor` interface. • `def startSensor(self, callBackFunc)`: the content should be implemented by users in their own class and based on the flag mechanism to avoid unnecessary crash. • `def stopSensor(self)`: the content should be implemented by users in their own class. • `def monitorRecipeStatus(self, callBackFunc)`, `def checkRecipeStatus(self, callBackFunc)`, `def waitForRecipeStop(self)`: flag mechanism provided by PMTW to monitor the recipe status in the Runtime. Users could use it with referring to the template in the template folder *C:\Program Files (x86)\ABB\PickMaster Twin 2\PickMaster Twin Client 2\PickMaster PowerPac\Template*.

The script template `ExternalSensor.py` is an example which users should follow the class format to develop their own class. Please note that the class must inherit from the four base classes defined in `ExternalSensorInterface.py` and the class name must be the same as the main file name, otherwise errors exist when PMTW tries to load the Python file.

def configureSensor(self, sensorId) interface

In this interface, the serialization of configuration data into a string and saving the string in the Python dictionary `sensorConfigurationDict[sensorId]` must be included.

This interface will be called when the **Configure in user program** button is clicked in the external sensor configuration view.

For more information on dictionary data, see *Data structure used in Python program on page 353*.

Argument	Description	Note
`self`	Python syntax Refer to the class	
`sensorId`	Sensor id automatically generated by PMPP	

Example:

```
def configureSensor(self, sensorId):# this interface must be
    implemented by users
  # Step 1 & Step 2:
  # analyze self.sensorConfigurationDict[sensorId] to get the
      settings from last configuration.
  # user-defined configuration logic.
  self.name = "ExternalSensorsDemo02"
  if self.sensorIdNameMapDict[sensorId] == 'ExternalSensor_1':
    inputTitle = "ExternalSensor_1 configuration"
  elif self.sensorIdNameMapDict[sensorId] == 'ExternalSensor_2':
    inputTitle = "ExternalSensor_2 configuration"
```

Continues on next page

```
if sensorId in self.sensorConfigurationDict.keys():
  configurationInfo =
        ExternalSensorsDemo.showSensorConfigDialog(self,
        inputTitle, self.sensorConfigurationDict[sensorId])
else:
  configurationInfo =
        ExternalSensorsDemo.showSensorConfigDialog(self,
        inputTitle, '0')
log = {'LogLevel': 0,'Log': configurationInfo}
self.fLogCallback.ShowPythonLog(log)
# Step 3:
# parse the configuration data into one string and update it in
     self.sensorConfigurationDict[sensorId].
self.sensorConfigurationDict[sensorId] = configurationInfo
```

def configurePosGen(self, posGenId) interface

In this interface, the serialization of position generator configuration data into a string and saving the string in the Python dictionary `posGenConfigurationDict[posGenId]` must be included.

This interface will be called when the **Configure** button or **New position generator** button is clicked in the item/container source type view.

For more information on dictionary data, see *Data structure used in Python program on page 353*.

Argument	Description	Note
`self`	Python syntax Refer to the class	
`posGenId`	Position generator id automatically generated by PMPP	

Example:

```
def configurePosGen(self, posGenId):# this interface must be
     implemented by users.
  # Step 1 & Step 2:
  # analyze self.posGenConfigurationDict[posGenId] to get the
       settings from last configuration.
  # user-defined configuration logic.
  if self.sensorIdNameMapDict[self.posGenSensorMapDict[posGenId]]
       == 'ExternalSensor_1':
    inputTitle = "ExternalSensor_1 PosGen configuration"
  elif self.sensorIdNameMapDict[self.posGenSensorMapDict[posGenId]]
       == 'ExternalSensor_2':
    inputTitle = "ExternalSensor_2 PosGen configuration"
  if posGenId in self.posGenConfigurationDict.keys():
    positionGeneratorInfo =
          ExternalSensorsDemo.showPosGenConfigDialog(self,
          inputTitle, self.posGenConfigurationDict[posGenId])
  else:
    positionGeneratorInfo =
          ExternalSensorsDemo.showPosGenConfigDialog(self,
          inputTitle, "0")
  log = {'LogLevel': 0,'Log': positionGeneratorInfo}
```

Continues on next page

```
self.fLogCallback.ShowPythonLog(log)
# Step 3:
# parse the configuration data into one string and update it in
     self.posGenConfigurationDict[posGenId].
self.posGenConfigurationDict[posGenId] = positionGeneratorInfo
```

def startSensor(self, callBackFunc) interface

In this interface, users should handle the logic about letting the sensor run and generate positions. This interface will be called when the running recipe is starting.

The PMTW `callBackFunc` is given as interface input argument, and includes two concrete callback functions, namely `GetStrobeTime()` and `NewPosition(pos)`.

`GetStrobeTime()` should be called when the sensor gets the trigger signal and will get the strobe time. When the position is generated, `NewPosition(pos)` should be called to send newly generated position to PMTW together with the strobe time.

Argument	Description	Note
`self`	Python syntax Refer to the class	

Continues on next page

Argument	Description	Note

`callBackFunc`	Which contains `GetStrobeTime()` and `NewPosition(pos)`	The position format should follow this structure:
		`newPos = {‘SensorId’: sensorId, # sensor id`
		`‘Time’: strobeTime, # time stamp, get from PMTW by calling GetStrobeTime() callback function, unit is ms.`
		`‘key’: {'X': 0.0, # key refers to the position index, start from 0. X refers to the location value of the item in X direction, unit is mm.`
		`'Y': 100.0, # Y refers to the location value of the item in Y direction, unit is mm`
		`'Z': 5.0, # Z refers to the location value of the item in Z direction, unit is mm`
		`'RX': 0.0, # RX refers to the rotation angle value of the item in X direction, unit is degree`
		`'RY': 0.0, # RY refers to the rotation angle value of the item in Y direction, unit is degree`
		`'RZ': 0.0, # RZ refers to the rotation angle value of the item in Z direction, unit is degree`
		`'Tag': 0, # used in rapid`
		`'Score': 1.0, # refers to the score of different position generator methods, could be used to sort the results`
		`'Val1': 0.0, # optional value, used in rapid`
		`'Val2': 0.0, # optional value, used in rapid`
		`'Val3': 0.0, # optional value, used in rapid`
		`'Val4': 0.0, # optional value, used in rapid`
		`'Val5': 0.0, # optional value, used in rapid`
		`'Level': 2, # level, 0: Discarded, 1: Rejected, 2: Accepted`
		`'PosGenId': posGenId}, # external sensor position generator id`

Continues on next page

Argument	Description	Note

		The following example shows the format which contains two positions:
		`newPos = {‘SensorId’: ‘11548258-b028-470a-b399-b780084acc59’,`
		`‘Time’: 376910718,`
		`‘0’: {'X': 0.0,`
		`'Y': 100.0,`
		`'Z': 5.0,`
		`'RX': 0.0,`
		`'RY': 0.0,`
		`'RZ': 0.0,`
		`'Tag': 0,`
		`'Score': 1.0,`
		`'Val1': 0.0,`
		`'Val2': 0.0,`
		`'Val3': 0.0,`
		`'Val4': 0.0,`
		`'Val5': 0.0,`
		`'Level': 2,`
		`'PosGenId': '17dec9b3-8624-45fa-b8cb-366b457e6024'},`
		`‘1’: {'X': 0.0,`
		`'Y': 100.0,`
		`'Z': 5.0,`
		`'RX': 0.0,`
		`'RY': 0.0,`
		`'RZ': 0.0,`
		`'Tag': 0,`
		`'Score': 1.0,`
		`'Val1': 0.0,`
		`'Val2': 0.0,`
		`'Val3': 0.0,`
		`'Val4': 0.0,`
		`'Val5': 0.0,`
		`'Level': 2,`
		`'PosGenId': '5413832d-dbbe-44b9-aab3-6bb64d8d30a1'}}`

Example:

```
def startSensor(self, callBackFunc):# this interface must be
      implemented by users.
  try:
    # Step 1: call classname.monitorRecipeStatus(self, callBackFunc)
         to monitor the recipe status running in PMTW.
    ExternalSensorsDemo.monitorRecipeStatus(self, callBackFunc)

    # Step 2: start logic defined by users. For each sensor, a
         StoppableThread must be created to generate positions and
         appended to self.allThreads.
    for posGenId in self.posGenSensorMapDict:
      sensorId = self.posGenSensorMapDict[posGenId]
```

Continues on next page

```
        windowTitle = self.sensorIdNameMapDict[sensorId] + ' ' +
             self.posGenObjectMapDict[posGenId]
        if self.sensorIdNameMapDict[sensorId] == 'ExternalSensor_1':
          thread_1 =
               StoppableThread(target=ExternalSensorsDemo.showStartDialog,
               args=(self, callBackFunc, sensorId, posGenId,
               self.posGenConfigurationDict[posGenId], windowTitle))
          self.allThreads.append(thread_1)
        elif self.sensorIdNameMapDict[sensorId] == 'ExternalSensor_2':
          thread_2 =
               StoppableThread(target=ExternalSensorsDemo.showStartDialog,
               args=(self, callBackFunc, sensorId, posGenId,
               self.posGenConfigurationDict[posGenId], windowTitle))
          self.allThreads.append(thread_2)
      # Step 3: start all threads in self.allThreads.
      for td in self.allThreads:
        td.start()

      # Step 4: call classname.waitForRecipeStop(self) to wait for
           the stop signal from PMTW.
      ExternalSensorsDemo.waitForRecipeStop(self)
      # Step 5: stop all threads in self.allThreads. Note that the
           stop behavior of startSensor interface should be handled
           by users here.
      for td in self.allThreads:
        td.stop()
      log = {'LogLevel': 0, 'Log': "StartSensor: stopped all
           threads."}
      self.fLogCallback.ShowPythonLog(log)
    except:
      log = {'LogLevel': 2,'Log': "Python Error: Failed to start
           sensor"}
      self.fLogCallback.ShowPythonLog(log)
```

Using `startSensor` with flag mechanism

The users should implement the `startSensor` content based on the flag mechanism provided by PMTW. The following codes show an example.

- At the beginning of the interface, the `monitorRecipeStatus` method provided by PMTW should be called first to start a thread to monitor whether the recipe is running.
- Then users could implement their own logic about position generation logic. In this example, a thread is created to start a position generation simulator. Since it should be guaranteed that interface content can be finished, the thread should be able to be stopped. Therefore, the stoppableThread provided by PMTW developer is used in this example.
- It is possible that there are more than one position generator. If one position generator runs in one thread, there will be more than one thread. Thus, the next step is to start all threads and the content in all threads will be executed.
- The next step is to wait for the flag signal that the recipe is stopped. If the recipe is running, the flag will be 1. If the recipe is stopped, the flag will be

Continues on next page

0. The method of `waitForRecipeStop` is to get the flag value, so it should be called after all threads are called.

- If the flag signal shows that the recipe is stopped, all threads in this interface should be stopped. Please note that before directly stopping the thread, stop the content in the thread first if needed, e.g. close the connection port inside the thread.

def stopSensor(self) interface

In this interface, if users need to implement any logic, they could add contents in this interface.

This interface will be called when the running recipe is to be stopped.

Argument	Description	Note
self	Python syntax Refer to the class	

Example:

```
def stopSensor(self):# this interface must be implemented by users.
  log = {'LogLevel': 0,'Log': "Python info: stop sensor"}
  self.fLogCallback.ShowPythonLog(log)
```

Data structure used in Python program

In the `ExternalSensorInterface.py` file, several data structures are defined to save the relationship between sensor, position generator and object (item/container) and users could use them in their own class to realize the Python logic.

	Description
sensorIdNameMapDict	Store the relationship between sensor id and name. `SensorId` is generated automatically in PMPP, which users could not modify. `SensorName` refers to the name of external sensor node, e.g., **ExternalSensor_1**. Users could use the default name provided by PMPP or modify the name according to their requirements. For most Python interfaces, `SensorId` is given as input argument, so users could directly use the `SensorId` to search in the dictionary to find the corresponding sensor name. If users use the sensor name in their logic and modify the name in PMPP, the string in Python must be updated to keep the same as in PMPP. In most cases, users don’t need to define or assign key and values in this dictionary, they only use `self.sensorIdNameMapDict[sensorId]` to get the sensor name based on the `SensorId`. **Example:** `sensorIdNameMapDict = {'11548258-b028-470a-b399-b780084acc59': ' ExternalSensor_1'}`

Continues on next page

	Description
sensorConfigurationDict	Store the relationship between sensor id and configuration information string. According to previous sections, the configuration information string reflects how the sensor is configured and the data is serialized in a string and saved in PMPP solution. When `loadSensor` is called, the saved string will be updated in this dictionary, which is implemented in `ExternalSensorInterface.py`. When `configureSensor` is called, if the sensor has already been configured, users may need to analyze or deserialize the configuration string first. They could use `SensorId` (interface input argument) to find the string first: `self.sensorConfigurationDict[sensorId]`. At the end of `configureSensor`, users may change the configuration settings in this interface, so a new string should be generated to reflect the latest setting and saved in the dictionary: `self.sensorConfigurationDict[sensorId] = configInfoString`. When `saveSensor` is called, the configuration string could be found in the dictionary with the latest setting and returned to PMPP, which is implemented in `ExternalSensorInterface.py`. Example: `sensorConfigurationDict= {'11548258-b028-470a-b399-b780084acc59': ' SensorType:Camera;IP:192.169.10.10;Brightness:10'}`
posGenConfigurationDict	Store the relationship between position generator id and position generator configuration information string. Similar to sensor configuration, the position generator configuration information string reflects how the position generator is configured and the data is serialized in a string and saved in PMPP solution. When `loadPosGen` is called, the saved string will be updated in this dictionary, which is implemented in `ExternalSensorInterface.py`. When `configurePosGen` is called, if the position generator has already been configured, users may need to analyze or deserialize the configuration string first. They could use `posGenId` (interface input argument) to find the string first: `self.posGenConfigurationDict[posGenId]`. At the end of `configurePosGen` users may change the configuration settings in this interface, so a new string should be generated to reflect the latest setting and saved in the dictionary: `self.posGenConfigurationDict[posGenId] = posGenConfigInfoString`. When `savePosGen` is called, the configuration string could be found in the dictionary with the latest setting and returned to PMPP, which is implemented in `ExternalSensorInterface.py`. Example: `posGenConfigurationDict= {'17dec9b3-8624-45fa-b8cb-366b457e6024': 'PositionGeneratorType:Blob;Type:White;ThresholdValue:100'}`

Continues on next page

	Description
posGenSensorMapDict	Store the relationship between position generator id and sensor id. According to previous introduction, the external sensor position generator can be created for each sensor under different item/containers. If there are more than one external sensors in the PMPP solution, under the same item/container, **externalsensor_1** can own its position generator, and **externalsensor_2** could also have its own position generator. These two position generators belong to different external sensors, so their relationship must be clarified to avoid data confusion. Their relationship will be initialized in the interface `initializePosGenRelatedMap` which is implemented in the `ExternalSensorInterface.py` and users could use them when needed: `self.posGenSensorMapDict[posGenId]`. Example: `posGenSensorMapDict= {'17dec9b3-8624-45fa-b8cb-366b457e6024': '11548258-b028-470a-b399-b780084acc59'}`
posGenObjectMapDict	Store the relationship between position generator id and object name. Here object refers to item or container. According to previous introduction, the external sensor position generator can be created for each sensor under different item/containers. If there are more than one items in the PMPP solution, for the same external sensor, **item_1** can own its position generator, and **item_2** could also have its own position generator. These two position generators belong to different items, so their relationship must be clarified to avoid data confusion. As the object name may be needed in the user program logic, in this map dictionary, the object name is stored instead of item id. Their relationship will be initialized in the interface `initializePosGenRelatedMap` which is implemented in the `ExternalSensorInterface.py` and users could use them when needed: `self. posGenObjectMapDict [posGenId]` Example: `posGenObjectMapDict= {'17dec9b3-8624-45fa-b8cb-366b457e6024': 'Item_1'}`

All the script example files are provided in the folder *C:\Program Files (x86)\ABB\PickMaster Twin 2\PickMaster Twin Client 2\PickMaster PowerPac\Template* when PickMaster Client is installed.

Users could also overwrite the information properties of name, description, author and version. Except these interfaces and properties, users could implement more functions in this class or in other Python files for their own use. It’s just that PMTW software will only be aware of and invoke these four interfaces and ignore others.

All used Python files should locate in the same path under `C:\Users\..\Documents\PickMaster\PMScripts` on the Host PC.

Configuring the external sensor

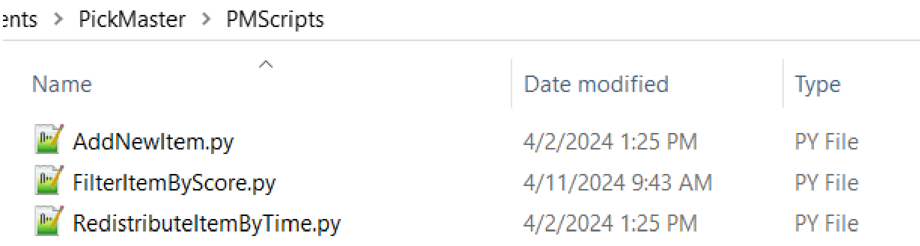
When an external sensor is created in the tree view, it is not connected to any physical sensor. This must be done manually in the external sensor configuration dialog box. The external sensor in the tree view is configured to use one specific physical external sensor. The external sensor should also be configured to give an optimal image.

Continues on next page

To configure an external sensor.

1 Put the predefined script files into the destination folder.

> **Tip**
>
> The predefined script file(s) should be put into *C:\Users\xxxx\Documents\PickMaster\PMScripts* folder before use any script function.



xx2200001784

2 Right-click the external sensor in the tree view **Layout** and select **Configuration**.

  The **External Sensor Configuration** dialog is opened.

3 Input the name of the predefined main file in **Script Name**.

4 Click **Configure in user program** to configure the external sensor.

5 Click **OK** to save the configuration in PMPP.

6 Right-click the desired item/container in the tree view **Process** and select **Setting**.

7 Switch to **Item Source**/**Container Source** page.

8 Click **New Position Generator** to create the position generator for desired external sensor under the **External** tab.

> **Note**
>
> Only the external sensor that has been configured in the **External Sensor Configuration** can create the position generator.

> **Tip**
>
> All created external sensors in this solution will be listed in **External** tab.

9 Configure the position generator according to the user defined in external sensor script file.

10 Click **Save** to save the configuration data to PMPP for the position generator.

11 Click **OK**.

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

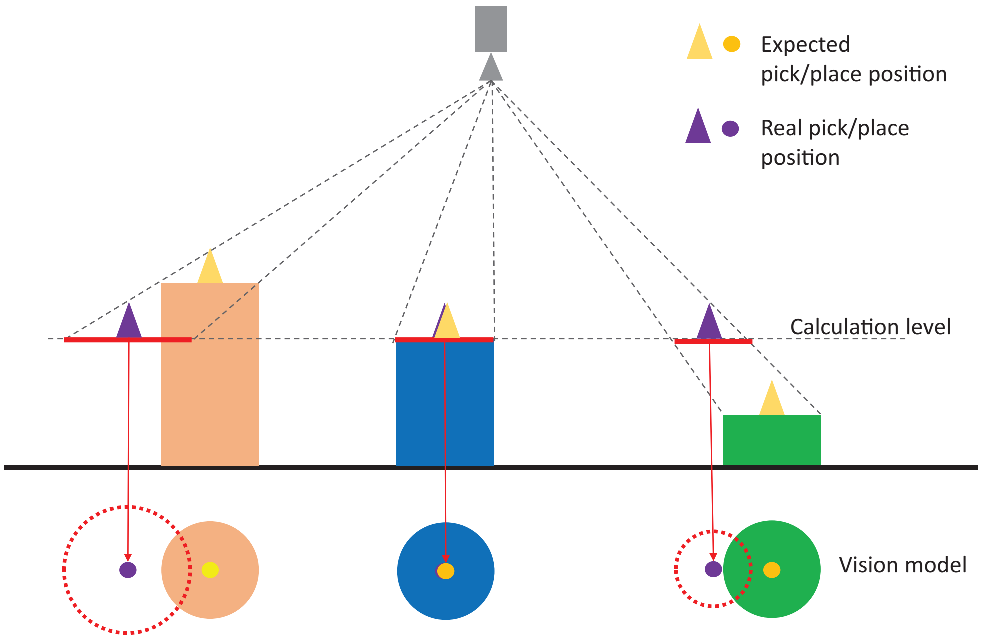
Introduction to height settings

The vision tools in PickMaster PowerPac typically return a result in a 2D coordinate system: X and Y angle, based on a calculation made at a certain height. 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 will result in parallax position problems.

Assuming a calculation on a defined height, any object of a different height will be shifted by the resulting parallax.

The camera is taught to calculate based on the blue block top surface. Without the parallel, the camera will employ this surface for the lower green block or higher orange block. As a result, x- and y-coordinates for the green block and the orange block would move based on the misused calculation plane. That would mislead the robot to target a wrong position.



xx2300001744

With 2.5D calibration, a full 3-dimensional space is calibrated, which allows the system to compensate for the parallax error, given that the system knows the correct height of the object. This raises the following 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, based on the camera location, provided by performing a multi-view calibration. See *Calibrating camera on page 273*.

Continues on next page

Determining the height at which an object is located can be done in three ways with PickMaster PowerPac.

1. Manual input
2. Automatic calculation based on the scale change in relation to the trained object.
3. External input

All three methods will return the parallax compensated x- and y-coordinates, and method 2 and 3 will also return an estimated z-coordinate.

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 be used together with a geometric standalone model, or main geometric-based inspection model.

Note

The main geometric-based inspection model does not compatible with Vision Height or External height.

Configuring height settings

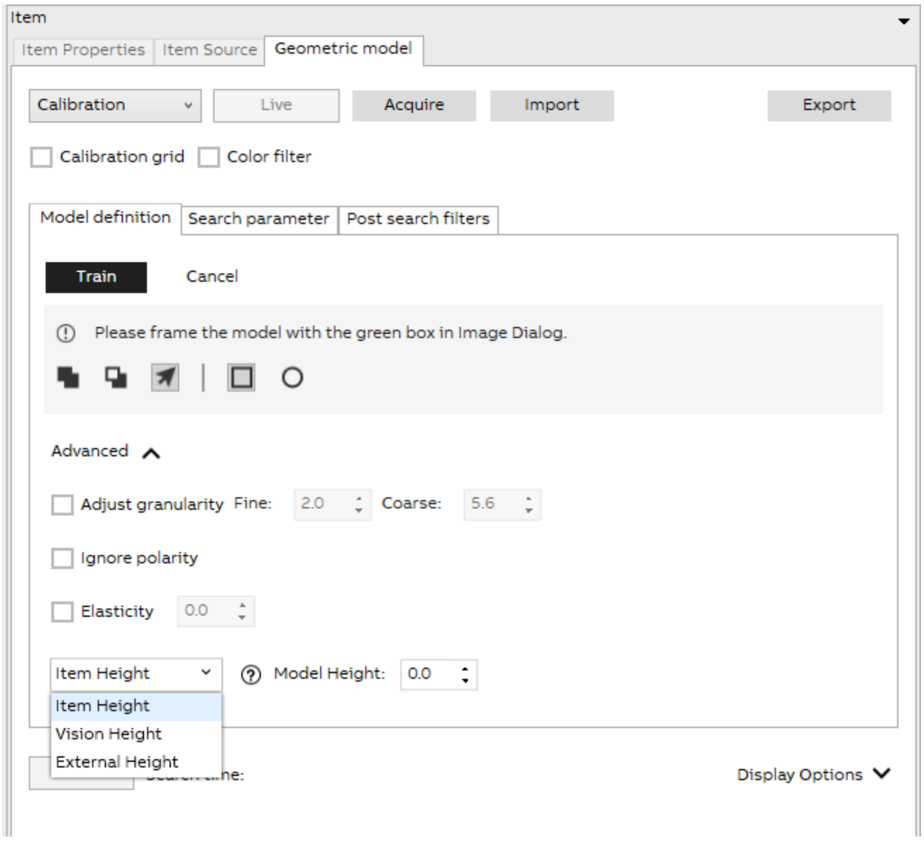
The height settings belong to a specific model and can only be configured together with **Geometric**.

Use this procedure to configure the height settings.

1. Create a **Geometric** 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 **Geometric advanced model settings** dialog.

Continues on next page

4 Choose an appropriate calculation method before training the item.



xx2300001532

5 If the calculation method is set as **Item Height**,

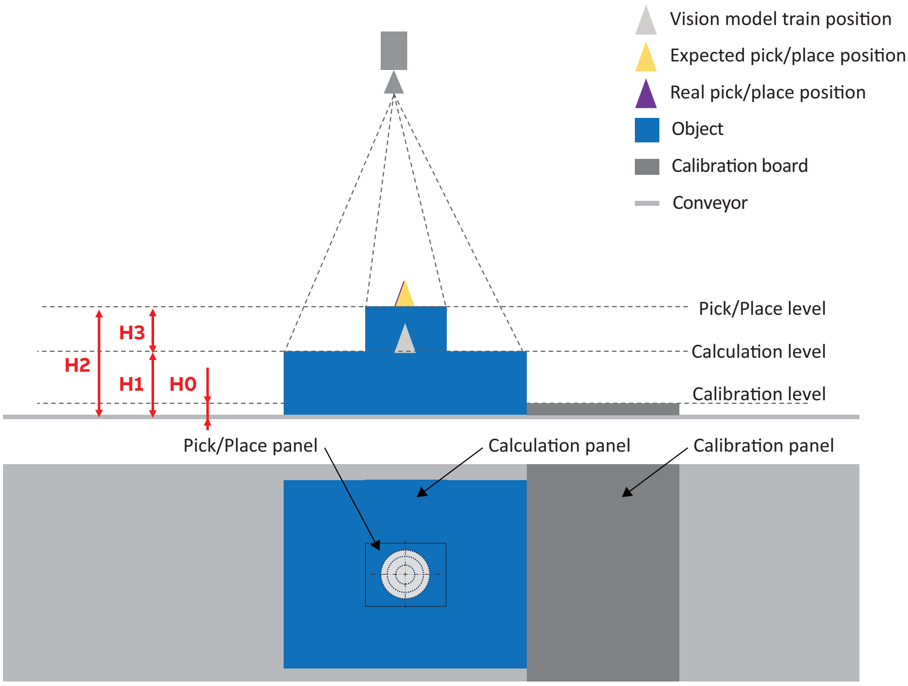
Item height: Manually enter the value for the picking/placing height.

xx2300001533

One parameter should be fulfilled for this calculation method. **Model Height** is literally used to describe the height from the calibration panel to the

Continues on next page

calculation plane. Z-coordinate is defined as the true picking/placing height for the object and would be sent to the robot controller.



xx2300001745

a Measure the height (H0) of the calibration board.

> **Tip**
>
> If the thickness of the calibration board is too thin to measure, for example a normal paper, then the user can ignore the height of calibration board and H0 is 0.

b Measure the height (H1) from the conveyor to the maximum contour panel for the picking/placing object.

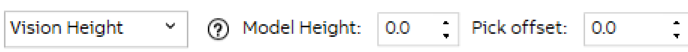
c Measure the height (H2) from the conveyor to the picking/placing panel for the picking/placing object.

d Enter the value (H1-H0) to the **Model Height**.

e Enter the value (H2-H0) to z-coordinate of the item setting **Size(x,y,z)[mm]/RH Size[mm]**. See *Item on page 136*.

6 If the calculation method is set as **Vision Height**,

Vision height:The value from the calibration panel to the calculation plane is calculated from the scale change (relative to the trained pattern) of the found object.



xx2300001534

Continues on next page

Two parameters should be fulfilled for this calculation method. **Model Height** follows the same meaning defined in the **Item Height**. **Pick Offset** is the deviation from the calculation plane to the picking/placing panel. The calculation plane is defined as the maximum contour panel of the identified object.

Note

Enable uniform scale must be enabled. The maximum and minimum values must allow for sufficient scale variation.

a Measure the height (H0) of the calibration board.

b Measure the height (H1) from the conveyor to the maximum contour panel for the picking/placing object.

c Measure the height (H3) from the maximum contour panel to the picking/placing panel of the picking/placing object.

Tip

If the picking/placing panel is higher than the calculation panel on the z-direction, H3 is a positive number.

If the calculation panel is higher than the picking/placing panel on the z-direction, H3 is a negative number.

d Enter the value (H1-H0) to the **Model Height**.

Tip

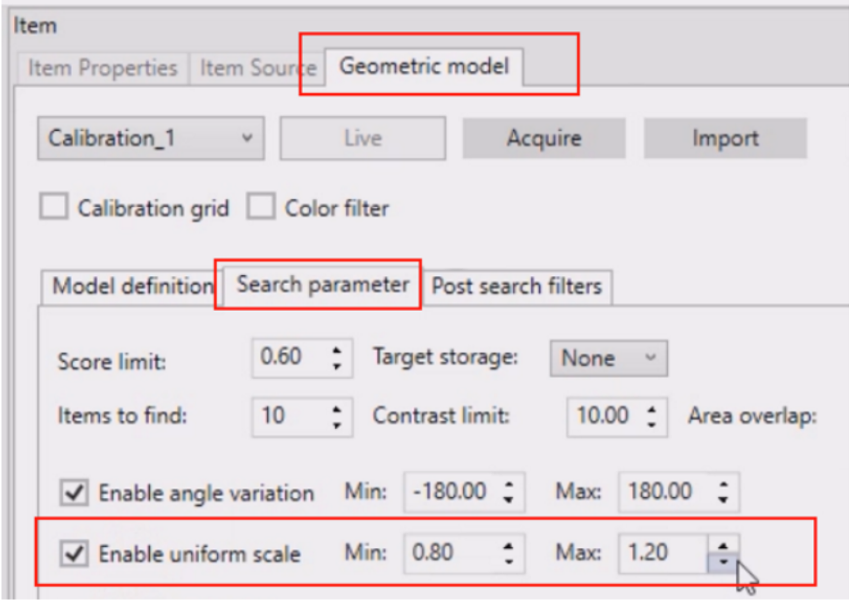
If this value is a positive number, that means the calculation panel is higher than the calibration panel on the z-direction.

If this value is a negative number, that means the calibration panel is higher than the calculation panel on the z-direction.

e Enter the value (H3) to the **Pick Offset**.

Continues on next page

f Enable the **Enable Uniform Scale** and enter a proper range for the scaling.



xx2300001537

7 If the calculation method is set as **External height**,

External height: The product's distance (above/below) to the calculation plane is calculated by the external source. This may be a height sensor or information from a cell PLC or any other external device. The z-coordinate is sent through a UDP port from external source to PickMaster Runtime. The UDP listening port for the external source should be unique for each vision model. Only the position message for current vision model can be sent through this vision model's listening port.

xx2300001535

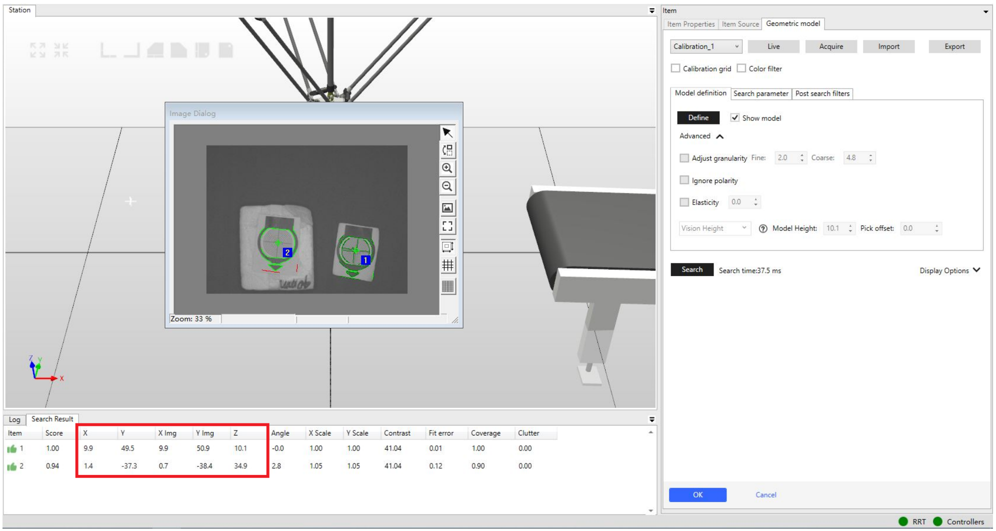
One parameter should be fulfilled for this calculation method.

a Enter the UDP port in **UDP Listening Port**. Then the calculated z-coordinate will be sent to PickMaster Runtime with the UDP message through this port.

> **Tip**
>
> This **UDP Listening Port** should match with the predefined port in external source.

Continues on next page

With the height setting configured during the model training, the search results will contain the space information for all searched objects.



xx2300001536

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 **Geometric model** dialog.

External height protocol message

The IP address and the port of the external source should be predefined by the user.

Continues on next page

The protocol message is composed of command, response port and measured height value. The message is a byte stream, which contains 5 bytes, `[1,y,y,y,y]`, e.g., `[0x01, 0x42, 0xDC, 0x38, 0x52]`.

Variable attributes are:

- `1`: indicates the command, value `0x01` means the external source would set the height value;
- `y,y,y,y`: indicates the measured height value, which is converted from the a float type value with big-endian sorting . The value indicates the target object height. In the example, byte value `0x42, 0xDC, 0x38, 0x52` indicates height value as 110.11.

Tip

When converting from float type to byte stream, sort in large end order.

Related information

Calibrating camera on page 273.

Item on page 136.

5.4 Production with flow(Ghost Picking)

Overview

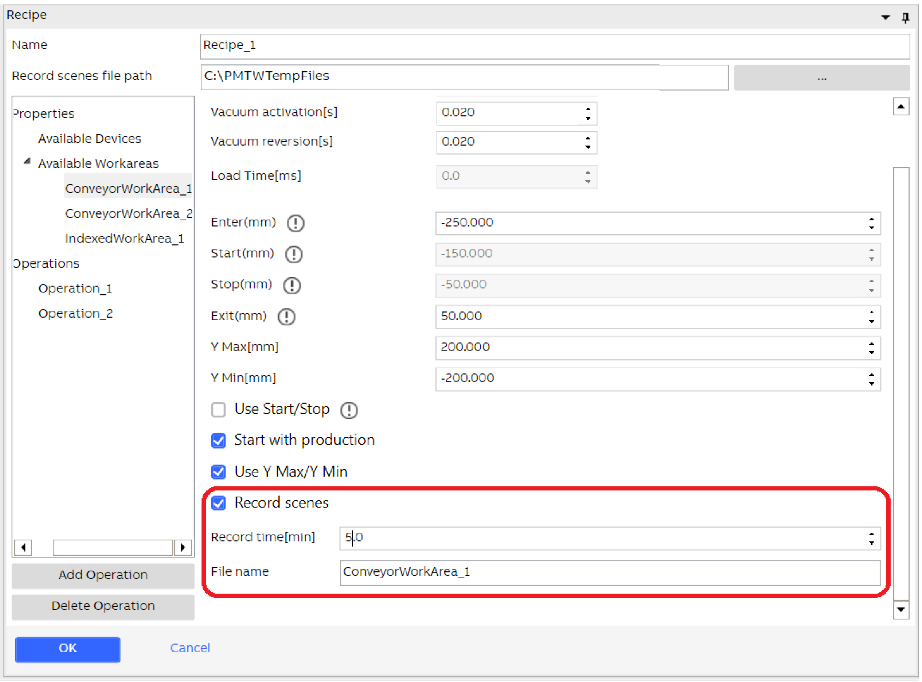
Ghost picking flow is used by the application engineer to run the dry cycle function before the production. The user sees the robot picking up empty objects on the real workstation. This feature differs from the production in that its incoming material is virtual and is provided by the flow generated by the previous record.

Creating a ghost picking flow

Use this procedure to create a ghost picking flow:

1. Open the solution need to do the ghost picking.
2. Right click on the recipe you need in the tree view **Process** and select **Setting**.

   The **Recipe setting** window is opened.
3. Click the **Conveyor WA** in the **Available Workareas** which need to be recorded to open the work area setting window.
4. Select **Record Scenes** checkbox in the **Record Setting**.

   xx2100001677

   > **Note**
   >
   > When **Record scenes** is selected and saved for any work area, the following message will pop up.
   >
   > `Scenes recording is activated for: {0}`
   >
   > After this, the recording will be activated automatically when the simulation or production is started.

5. Set the record time according to your requirements.

Continues on next page

6 Click **OK** to apply the configuration.
7 Click **OK** to close the **Recipe** dialog.
8 Run the production to start the recording.

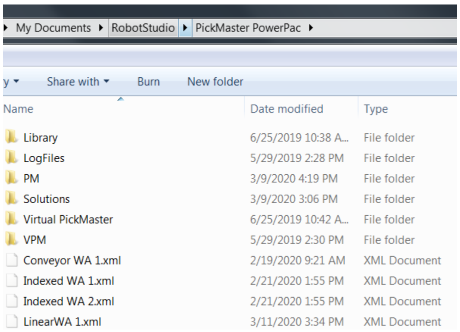
The created `.xml` file is stored in the **C:\PMTWTempFiles** folder.

Note

The `.xml` is stored on the connected Runtime PC.

Tip

The **Record scenes file path** can be changed to an existed folder on the connected Runtime PC.



xx1900000548

Adding a ghost picking flow to a solution

Use this procedure to add a ghost picking flow:

1 On the PickMaster PowerPac ribbon-tab, click **Process**.
2 On the ribbon-tab, click **Flow**.
   The **Flow** window is opened.
3 Click **Recorded** tab.
4 Select the type for the flow in **Flow Type**.

   Note

   The type selected here MUST be same with the imported flow type. Otherwise the flow cannot work normally

5 Click the **Import Flow** icon to import the predefined work area `.xml` file.
6 Click **Open** to apply the configuration.
7 Click **Item/ContainerPattern** drop-down list to select the desired item or container.
8 Click **OK** to save and close the **Flow** dialog.

Continues on next page

Modifying position generator

Use this procedure to modify the position generator for the ghost picking flow:

1. Right-click on one **Position Generator** in the tree view **Layout** and select **Setting**.

   The **Position Generator** window is opened.
2. Click on the conveyor which has added the recorded flow.
3. Select the **Vision** in the **Source Type**.
4. Select the used camera for this conveyor in the drop-down list.
5. Select **Distance** in the **Trigger Setting**.
6. Click **OK** to close the **Position Generator** window.

Selecting ghost picking flow (Modify recipe)

Use this procedure to select a ghost picking flow:

1. Right-click on one **Recipe** in the tree view **Process** and select **Setting**.

   The **Recipe setting** window is opened.
2. Click on the **Operation_1** to open the setting window for the operation.
3. In the **Select Flow**, select **Flow1** in the drop down list.

   Tip

   If the source type of conveyor is set as Predefined or no camera is added to the conveyor, the flow cannot be selected for this conveyor.
4. Add the **Conveyor WA 1** to the **Accept** by dragging under the **Distribution Setting** tab.
5. Click **OK** to close the **Recipe setting** window.

Ghost picking flow

Use this procedure to run a ghost picking flow:

1. On the PickMaster PowerPac ribbon-tab, click **Operation**.
2. Select the recipe which will be running in the tree view and click **Control** on the ribbon tab **Operation**.

   The **Control** dialog is opened.
3. Click **Start** to run the production.

   Note

   The ghost picking is default set as looped. It will repeat sending the recorded position data to the real controller until the **Stop** icon is clicked.

   Note

   The detailed vision is not applicable when running ghost picking flow.

   The emulation starts running.

Continues on next page

4 Click **Stop** to stop the ghost picking in the production.

6 RAPID reference

6.1 Instructions

6.1.1 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

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 392*.

6.1.2 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>;
```

6.1.3 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

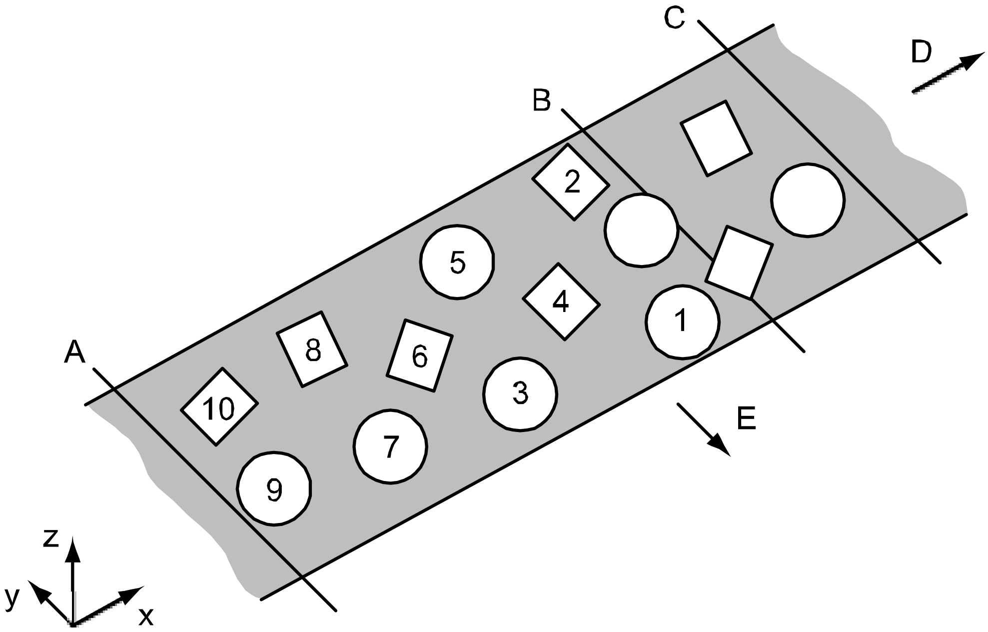
```
...
VAR selectiondata neg_y_sort;

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

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 392*.
The data type `selectiondata`	*selectiondata - Selection data on page 395*.
The data type `sortdata`	*sortdata - Sort data on page 398*.

6.1.4 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 392*.

6.1.5 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 381*.

6.1.6 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 380*.

6.1.7 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 391*.

6.1.8 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 Ready*.

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>;
```

6.1.9 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*.

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 Ready* option.

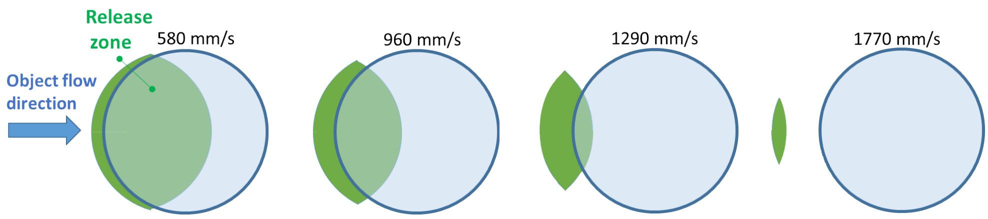
WARNING

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

```
[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.

6.2 Functions

6.2.1 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 Ready*

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.

6.2.2 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`.

6.2.3 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`.

6.2.4 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 382*.

6.3 Data types

6.3.1 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

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*

6.3.2 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 419*.

6.3.3 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>
```

6.4 RAPID program

6.4.1 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.mod`	Four axes FlexPicker IRB 360.
`PMppaDelta.mod`	Five axes FlexPicker IRB 365 and IRB 390.
`PMppa6Axes.mod`	SCARA robots, for example, IRB 910. Four axes robots of articulated arm type IRB 460 and IRB 660. Six axes robots of articulated arm type, for example, IRB 120. Seven axes robots of articulated arm type, for example, IRB 14050.

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.

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.

Continues on next page

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.

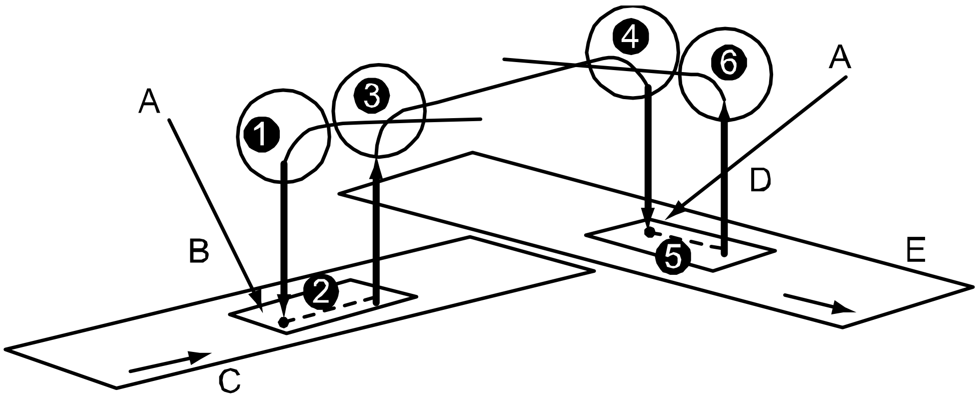
For each place, a place target is fetched from the place item source. The target position gives the location of the next empty place location for the item to be placed.

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.
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.

Continues on next page

	Description
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
PMTWMAIN	Handles the main program initiations and execution sequence. Do not edit this module for customization purpose.

System modules

An ABB robot controller with the RobotWare option *PickMaster Ready* 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`.

Public data types

Overview

The system module *ppaUser* contains two record definitions, *sourcedata* and *noncnvwobjdata*.

Continues on next page

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.

6.4.2 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

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`.

6.4.3 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 SetSimulatedTriggs(num Index)

The offline version of PROC SetTriggs(num Index).

No need to change if the tool has 1 - 2 activators.

PROC SetSimulatedDummyTriggs(num Index)

Sets up all trigger events used in the RAPID code that not is relevant for simulated mode.

No need to change if the tool has 1 - 4 activators.

PROC SetDummyTriggs(num Index)

Sets up all trigger events used in the RAPID code that not is relevant for online mode.

No need to change if the tool has 1 - 4 activators.

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.

Continues on next page

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.

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.

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.

Continues on next page

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.

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.

Continues on next page

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.

PROC ppaDropWobj(PERS wobjdata Wobj)

Encapsulates the `DropWobj` 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.

PROC AlwaysClearPath(bool always)

For more details, see

AlwaysClearPath on page 403

6.5 Program examples

6.5.1 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
```

6.5.2 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
```

6.5.3 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
```

6.5.4 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 **Recipe Setting**, 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 PlaceWorkArea{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

```
      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.

6.5.5 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
```

6.5.6 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.

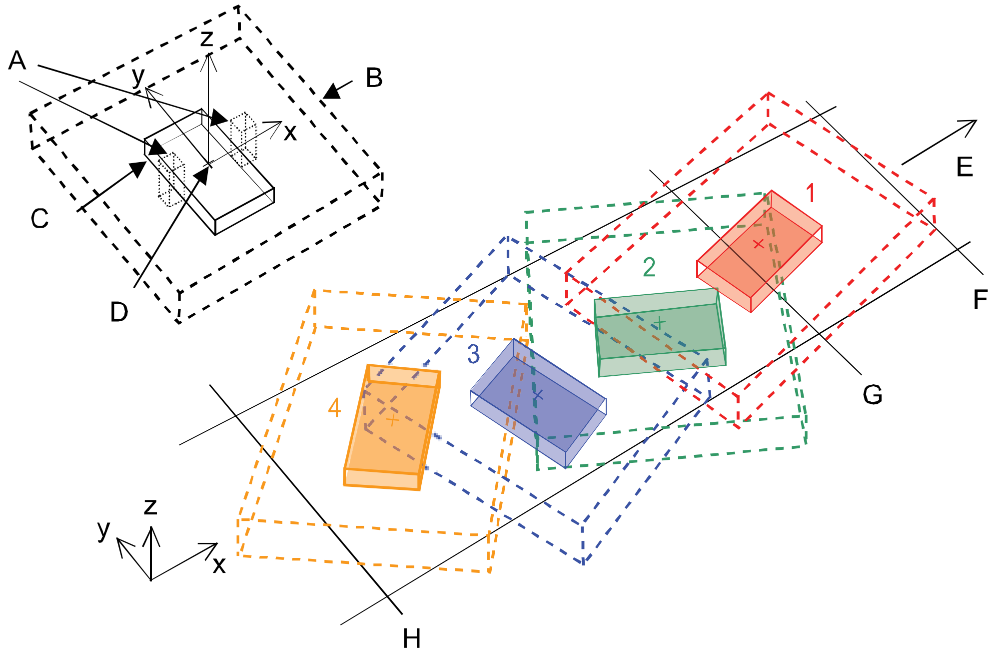
The selection algorithm selects the object that is closest to the exit limit in x-direction and has no locking objects in the selection shape.

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 395*.

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

```
    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
```

6.5.7 Example: Sorting in negative Y-direction

6.5.7.1 Sorting in negative y-direction for linear conveyor

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.

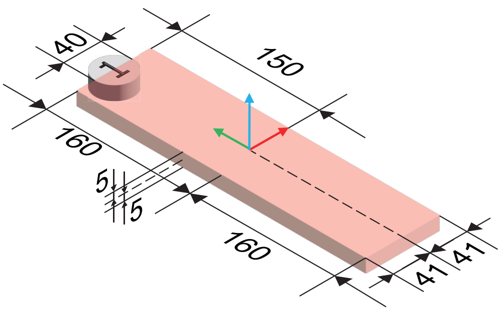
Set the y-value in the `OffsetData` to the negative y-value of the shape, the selection box will be moved out to the right.

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.

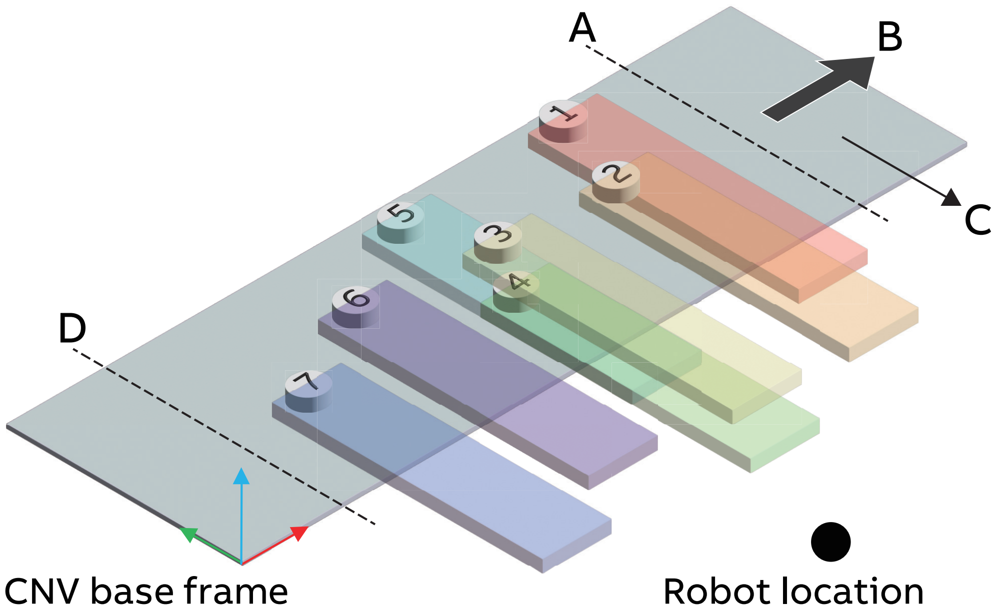
Use the check limit in the x-direction as a parameter to the `GetItmTgt` instruction, to define the starting point from where the first item will be shuffled. The instruction will try to shuffle the first item between the check and enter limits. This will also cause the selection algorithm to take all 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*, the items will be shuffled off in the order 1 to 7 as numbered in the illustration below. The picking order with setting as sorting in negative y-direction will be 2->1->4->3->5->6->7.



xx2400000740

Continues on next page



xx2400000737

A	Exit
B	Product flow direction
C	Sort direction
D	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;
  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;
```

Continues on next page

```
  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
```

6.5.7.2 Sorting in negative Radius-direction for circular conveyor

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 y-value is used to define the corridor width, the x-value must be more than half the radius of the conveyor belt and the z-value must be greater than the largest difference in height among all items.

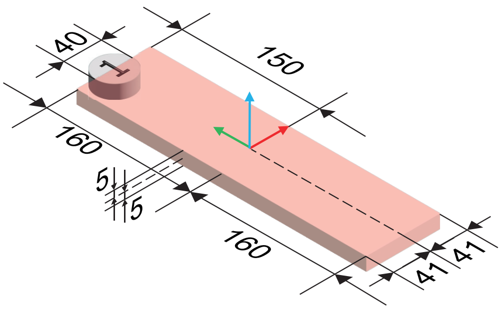
Set the x-value in the `OffsetData` to the negative x-value of the shape, the selection box will be moved out to the right.

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 y-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.

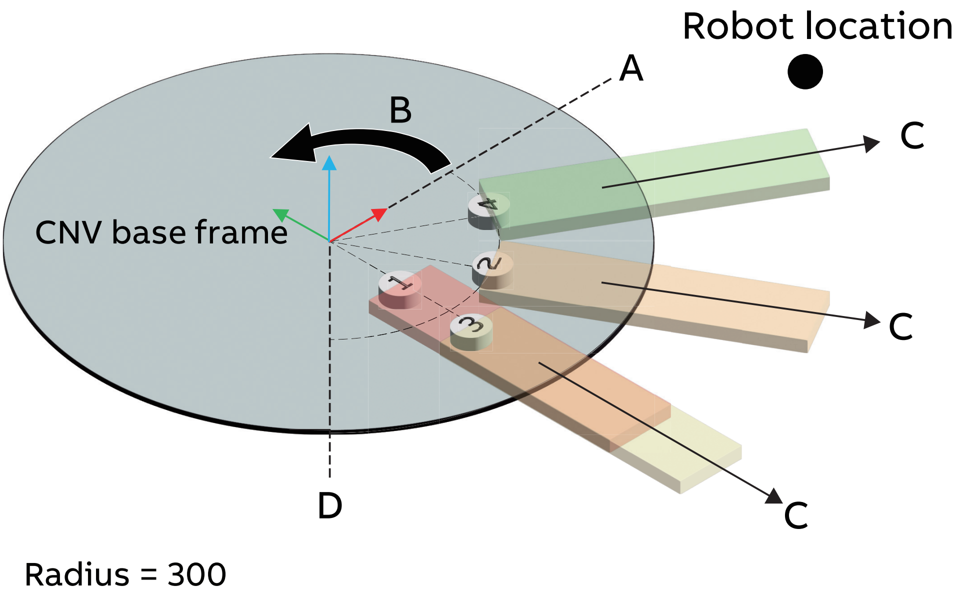
In the illustration below, all items will be shuffled off to the right side of the conveyor belt. Because each item needs a clear zone, that is the shape of the *ShapeType*, the items will be shuffled off in the order 1 to 4 as numbered in the illustration below.



xx2400000740

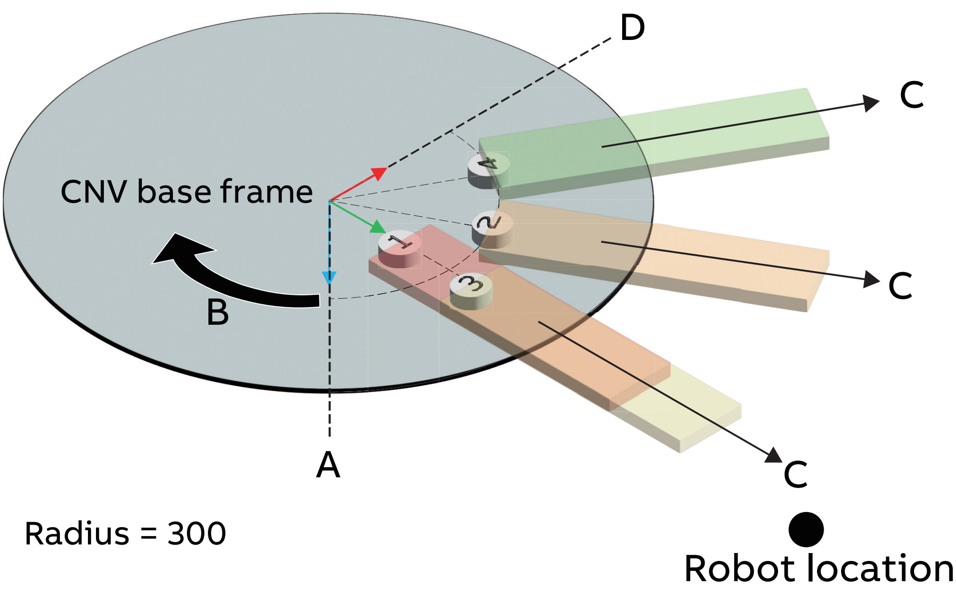
Continues on next page

For the counter clockwise movement direction conveyor, the picking order with setting as sorting in negative y-direction will be 4->2->3->1.



xx2400000738

For the clockwise movement direction conveyor, the picking order with setting as sorting in negative y-direction will be 3->1->2->4.



xx2400000739

A	Exit
B	Product flow direction
C	Sort direction
D	Enter

Continues on next page

Example code

```
PROC Pick(num Index)
  VAR selectiondata x_sort;
  VAR robtarget draw_target;
  VAR num check_limit;

  x_sort.ShapeType:=BOX;
  x_sort.ConsiderType:=BitOr(ITEMS_TO_USE,ITEMS_BYPASS);
  x_sort.GeometricData.x:=160;
  x_sort.GeometricData.y:=41;
  x_sort.GeometricData.z:=5;
  x_sort.GeometricData.radius:=0;
  x_sort.Offset.OffsetRelation:=FRAME_COORD_DIR;
  x_sort.Offset.OffsetPose.trans.x:=-150;
  x_sort.Offset.OffsetPose.trans.y:=0;
  x_sort.Offset.OffsetPose.trans.z:=0;
  x_sort.Offset.OffsetPose.rot.q1:=1;
  x_sort.Offset.OffsetPose.rot.q2:=0;
  x_sort.Offset.OffsetPose.rot.q3:=0;
  x_sort.Offset.OffsetPose.rot.q4:=0;
  check_limit:=150;

  WObjPick:=ItmSrcData{Index}.Wobj;
  GetItmTgt ItmSrcData{Index}.ItemSource,PickTarget
       \Limit:=check_limit\Selection:= x_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.x:=-200;
  draw_target.rot:=[0,1,0,0];
  TriggL draw_target, LowSpeed, ItmSrcData{Index}.Ack, z20,
       Gripper\WObj:=WObjPick;
ENDPROC
```

6.5.8 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 diagram - PickMaster Twin*, *3HAC024480-020*.

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.

6.5.9 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 428*.

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.

6.5.10 Example: Item buffer

Description of example

In this example we use item buffer. The items are put on the predefined buffer position.

1. Pick item from pick work area
2. Place on buffer position

> **Note**
>
> The buffer position must be out of the range in X axis and Y axis within the conveyor. Otherwise the robot will place the item on the conveyor directly rather than on the buffer position.

Example code

Define the buffer position in the RAPID program. See the following example code.

```
!***********************************************************
! Global BUFFER Variables
! Robtarget BufferPos must be defined in wobj0
!***********************************************************
TASK PERS robtarget BufferPos{3}:=[
     [[-200,-10,-1084],[0,-1,0,0],[0,0,0,0],[0,0,0,0,0,0]] ,
     [[0,-10,-1084],[0,-1,0,0],[0,0,0,0],[0,0,0,0,0,0]] ,
     [[200,-10,-1084],[0,-1,0,0],[0,0,0,0],[0,0,0,0,0,0]] ];
VAR num BufferMax{3}:=[3,3,3]; !Number of items i a buffer
VAR num BufferPitch{3}:=[50,50,50]; !distance between buffer
     positions
VAR num DropAction:=0; !What to do if an item can not be used.
     0=Ack 1=Nack 2=Skip
VAR num BufferZ{3}:=[0,0,0]; !Buffer Z-adjustments

VAR num InFlowEnter:=-250; !Set to same as pickarea Enter limit
VAR num InFlowExit:=250; !Set to litte before pickarea Exit limit
VAR num OutFlowEnter:=-250; !Set to same as placearea Enter limit
VAR num OutFlowExit:=250; !Set to litte before placearea Exit limit

VAR num BufferX{3};
VAR num BufferY{3};
VAR num BufferIndex{3}:=[0,0,0];
VAR num IType:=0;
VAR num Picked:=0;
```

Modify the `PickPlaceSeq` routine to perform Pick routine and Place routine on the buffer position. See the following example code.

```
!***********************************************************
!
! Procedure PickPlaceSeq
!
! The Pick and Place sequence.
```

Continues on next page

```
! Edit this routine to modify from which work areas to pick and
     place.
! Needs to be changed if more than one pick work area is used.
! Needs to be changed if more than one place work area is used.
!
!***********************************************************
PROC PickPlaceSeq()
VAR num GQL:=0;
Picked:=0;
WHILE Picked=0 DO
IType:=0;
!=== PickPosAvailable ? ===
GQL:=GetQueueLevel (ItmSrcData{PickWorkArea{1}}.ItemSource
     \MinLimit:=InFlowEnter \MaxLimit:=InFlowExit);
If GQL>0 THEN
NextItmTgtType ItmSrcData{PickWorkArea{1}}.ItemSource, IType;
!=== Matching Item on outfeeder ? ===
GQL:=GetQueueLevel (ItmSrcData{PlaceWorkArea{1}}.ItemSource
     \ItmType:=IType \MinLimit:=OutFlowEnter
     \MaxLimit:=OutFlowExit);
IF GQL>0 THEN
Picked:=1;
ELSE
!=== Empty pos in buffer? ===
IF BufferIndex{IType}<BufferMax{IType} THEN
Picked:=3;
ELSE
!=== What to do with the item on the infeeder? ===
GetItmTgt
     ItmSrcData{PickWorkArea{1}}.ItemSource,PickTarget\ItemType:=IType;
TEST DropAction
CASE 0:
AckItmTgt ItmSrcData{PickWorkArea{1}}.ItemSource,PickTarget,TRUE;
CASE 1:
AckItmTgt ItmSrcData{PickWorkArea{1}}.ItemSource,PickTarget,FALSE;
CASE 2:
AckItmTgt
     ItmSrcData{PickWorkArea{1}}.ItemSource,PickTarget,FALSE\Skip:=TRUE;
ENDTEST
Picked:=0;
ENDIF
ENDIF
ELSE
GQL:=GetQueueLevel (ItmSrcData{PlaceWorkArea{1}}.ItemSource
     \MinLimit:=OutFlowEnter \MaxLimit:=OutFlowExit);
If GQL>0 THEN
NextItmTgtType ItmSrcData{PlaceWorkArea{1}}.ItemSource, IType;
!=== Matching Item on Infeeder ? ===
GQL:=GetQueueLevel (ItmSrcData{PickWorkArea{1}}.ItemSource
     \ItmType:=IType \MinLimit:=InFlowEnter \MaxLimit:=InFlowExit);
IF GQL>0 THEN
```

Continues on next page

```
Picked:=1;
ELSE
!=== Matching pos in buffer? ===
IF BufferIndex{IType}>0 THEN
Picked:=2;
ELSE
Picked:=0;
ENDIF
ENDIF
ELSE
Picked:=0;
ENDIF
ENDIF

TEST Picked
CASE 0:
!=== No pick, no place ===
WaitTime 0.1;
CASE 1:
!=== pick infeed, place outfeed ===
Pick PickWorkArea{1},IType;
Place PlaceWorkArea{1},IType;
CASE 2:
!=== pick buffer, place outfeed ===
PickBuffer PickWorkArea{1},IType;
Place PlaceWorkArea{1},IType;
CASE 3:
!=== pick infeed, place buffer ===
Pick PickWorkArea{1},IType;
PlaceBuffer PlaceWorkArea{1},IType;
ENDTEST
ENDWHILE
ENDPROC
```

Edit this routine to modify how the robot shall execute the pick and place movements.

```
!***********************************************************
!
! Procedure PickBuffer
!
! Executes a pick movement
!
!***********************************************************
PROC PickBuffer(num Index,num TypeNr)
VAR num zboffs;
BufferX{TypeNr}:=(BufferIndex{TypeNr}-1)*BufferPitch{TypeNr};
WObjPick:=Wobj0;
zboffs:=BufferZ{TypeNr};
TriggL\Conc,Offs(RelTool(BufferPos{TypeNr},0,0,-ItmSrcData{Index}.OffsZ),
BufferX{TypeNr},0,0),MaxSpeed,ItmSrcData{Index}.VacuumAct1,z20,
PickAct1\WObj:=WObjPick;
```

Continues on next page

```
TriggL\Conc,Offs(BufferPos{TypeNr},BufferX{TypeNr},0,zboffs),
LowSpeed,ItmSrcData{Index}.SimAttach1,z5\Inpos:=ItmSrcData{Index}.TrackPoint,
PickAct1\WObj:=WObjPick;
GripLoad ItemLoad;
MoveL Offs(RelTool(BufferPos{TypeNr},0,0,-ItmSrcData{Index}.OffsZ),
BufferX{TypeNr}, 0,0), LowSpeed,z20,PickAct1\WObj:=WObjPick;
Decr BufferIndex{TypeNr};
ENDPROC


!***********************************************************
!
! Procedure PlaceBuffer
!
! Executes a place movement
!
!***********************************************************
PROC PlaceBuffer(num Index,num TypeNr)
VAR num zboffs;
Incr BufferIndex{TypeNr};
BufferX{TypeNr}:=(BufferIndex{TypeNr}-1)*BufferPitch{TypeNr};
WObjPlace:=Wobj0;
zboffs:=BufferZ{TypeNr};
MoveL\Conc,Offs(RelTool(BufferPos{TypeNr},0,0,-ItmSrcData{Index}.OffsZ),
BufferX{TypeNr},0,0), MaxSpeed,z20,PickAct1\WObj:=WObjPlace;
TriggL\Conc,Offs(BufferPos{TypeNr},BufferX{TypeNr},0,zboffs),LowSpeed,
ItmSrcData{Index}.VacuumRev1\T2:=ItmSrcData{Index}.VacuumOff1\T3:=
ItmSrcData{Index}.SimDetach1,z5\Inpos:=ItmSrcData{Index}.TrackPoint,
PickAct1\WObj:=WObjPlace;
GripLoad load0;
MoveL Offs(RelTool(BufferPos{TypeNr},0,0,-ItmSrcData{Index}.OffsZ),
BufferX{TypeNr},0,0),LowSpeed,z20,PickAct1\WObj:=WObjPlace;
ENDPROC
```

This page is intentionally left blank

7 Troubleshooting

7.1 Introduction to troubleshooting

Troubleshooting

This chapter describes some of the most common troubles known when installing, configuring, or running PickMaster PowerPac.

A fault in the robot system first appears as a symptom, which can be:

- An event log message that can be viewed using PickMaster Twin, 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 *Technical reference manual - Event logs for RobotWare 7* and *Operating manual - Troubleshooting IRC5*.

7.2 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.

7.3 Administering the log of PickMaster Twin Powerpac

Type of the log messages

There are three types of event log messages.

Type	Description
Information	These messages are used to log information into the event log, but that do not require specific actions on the part of the user. Informational messages do not take focus on a display device for the controller.
Warn	These messages are used to remind the user that something is not necessarily correct in the system but that operation continues. These messages are placed into the event log but do not take focus on a display device.
Error	These messages indicate something is seriously wrong with the system and that operation has stopped. They are used when an immediate action should be taken by the user.

Note

The message type is not contained in the actual message text, but in the code itself when the message is generated. Messages can only appear as one type and the message should be written to provide information about its type.

Administering the log in PickMaster Twin Powerpac

Use this procedure to administer the event log.

1. In the **Log** view, all the event log of current station are showing here.
2. If you need to view the event log without the PickMaster Twin Powerpac, right click in the log area and select **Save Log**. The event log will be saved as `csv` file.
3. If you need to clear the event log for current station, right click in the log area and select **Clear All**. The event log will be cleared all.

7.4 Administering the log of PickMaster Twin Operator

Type of the log messages

There are three types of event log messages.

Type	Description
PMOPInfo	These messages are used to log information into the event log, but that do not require specific actions on the part of the user. Informational messages do not take focus on a display device for the controller.
PMOPWarning	These messages are used to remind the user that something is not necessarily correct in the system but that operation continues. These messages are placed into the event log but do not take focus on a display device.
PMOPError	These messages indicate something is seriously wrong with the system and that operation has stopped. They are used when an immediate action should be taken by the user.
RTInfo/RTStatus	These messages are used to log information into the event log, but that do not require specific actions on the part of the user. Informational messages do not take focus on a display device for the controller.
RTWarning	These messages are used to remind the user that something is not necessarily correct in the system but that operation continues. These messages are placed into the event log but do not take focus on a display device.
RTError	These messages indicate something is seriously wrong with the system and that operation has stopped. They are used when an immediate action should be taken by the user.

Note

The message type is not contained in the actual message text, but in the code itself when the message is generated. Messages can only appear as one type and the message should be written to provide information about its type.

Administering the log in PickMaster Twin Operator

Use this procedure to administer the event log.

1. Click the **Log** in ANALYSIS group. The event log will show up.
2. If you need to view the event log without the PickMaster Twin Operator, click **Export**. The event log will be saved as `txt` file.

7.5 Administering the log of PickMaster Twin Runtime

Type of the log messages

There are three types of event log messages.

Type	Description
Status	These messages are used to log information into the event log, but that do not require specific actions on the part of the user. Informational messages do not take focus on a display device for the controller.
Warning	These messages are used to remind the user that something is not necessarily correct in the system but that operation continues. These messages are placed into the event log but do not take focus on a display device.
Error	These messages indicate something is seriously wrong with the system and that operation has stopped. They are used when an immediate action should be taken by the user.

Note

The message type is not contained in the actual message text, but in the code itself when the message is generated. Messages can only appear as one type and the message should be written to provide information about its type.

Administering the log in PickMaster Twin Runtime

Use this procedure to administer the event log.

1. In the **Log** view, all the event log are showing here.
2. If you need to view the event log without the PickMaster Twin Runtime, click on **Log** ribbon or right click in the log area and select **Save Log...**. The event log will be saved as `txt` file.
3. If you need to clear the event log for current station, right click in the log area and select **Clear All**. The event log will be cleared all.

Administering the log with Windows *Event Viewer*

The event log messages that are displayed in the log area of PickMaster Runtime are also stored in the Windows event log. The messages can be viewed with Windows *Event Viewer*.

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 **Windows Logs** -> **Application**.
3. To see only PickMaster messages, right-click **Application** and click **Filter Current Log...**. Then select **PickMaster** as event source.
4. To save the log, right-click **Application**, and select **Save Filtered 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.

Continues on next page

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**.

7.6 Runtime 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.
4102	Status	User information.
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.

Continues on next page

Error code	Type	Description
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.
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.

Continues on next page

Error code	Type	Description
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.
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 456*.
4327	Warning	Expected item positions missing from position source. For more information, see *Warnings 4326 - 4329 on page 456*.
4328	Warning	Trigger/strobe time mismatch. Item positions from position souce to work area lost. For more information, see *Warnings 4326 - 4329 on page 456*.

Continues on next page

Error code	Type	Description
4329	Warning	Trigger/strobe time mismatch. Strobe from work area was ignored. For more information, see *Warnings 4326 - 4329 on page 456*.
4396	Error	A COM error occurred in when using an External Sensor. The log message provides more information.
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.

Continues on next page

Error code	Type	Description
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.
4814	Error	Configure PickMaster Time Synchronization Service error. Please check the configuration parameter of Time Synchronization Service.
4815	Warning	PickMaster Time Synchronization Service network adapter is not selected.
4896	Error	Description: Unexpected error when creating controller. Reason: An unexpected error occurred in PickMaster. See the log message for more information.
4897	Error	Description: Unexpected error when update robot. Reason: An unexpected error occurred in PickMaster. See the log message for more information.
4898	Error	Description: Unexpected error when configure controller. Reason: An unexpected error occurred in PickMaster. See the log message for more information.
4899	Error	Description: Unexpected error when handle workerea. Reason: An unexpected error occurred in PickMaster. See the log message for more information.
4900	Warning	User management operations are abnormal. See the log message for more information.
4901	Warning	External sensor abnormal. Please check the configuration of external sensor.
4902	Status	Vision information
4903	Status	User hook information
4904	Status	Index workarea information
4905	Status	Line file information
4906	Status	Log management information
4907	Status	Login information
4908	Status	Main form information
4909	Status	Camera model information
4910	Warning	Network setting abnormal. Please review the recommended network settings in the user's guide.
4911	Warning	Option setting changed. See the log message for more information.
4912	Warning	Pattern operations are abnormal. See the log message for more information.
4913	Warning	Description: Unexpected error. Reason: An unexpected error occurred in PickMaster. See the log message for more information.
4914	Warning	Position source setting abnormal. See the log message for more information.

Continues on next page

Error code	Type	Description
4915	Warning	Project setting abnormal. See the log message for more information.
4916	Warning	RIS xml file abnormal. See the log message for more information.
4996	Status	Item position information
4997	Status	Item overlap filter information
4998	Status	Item distribution information
4999	Status	Item information for work area
5000	Status	Item ACK information
16785	Warning	Project open abnormal. Can not open a new project when another project is running.
16885	Status	Ris2 client connection information
16886	Status	Ris2 client disconnection information

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).

Continues on next page

Error code	Type	Description
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.
8214	Warning	The connection to the controller is regained.
8215	Warning	Description: Unexpected error. Reason: An unexpected error occurred in PickMaster. See the log message for more information.
8216	Status	Controller reconnection information
8217	Status	OmniCore robot connection information
8218	Status	OmniCore robot disconnection information
8219	Status	Robot holding information
8220	Status	Robot suspending information
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.

Continues on next page

Error code	Type	Description
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.
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

Continues on next page

Error code	Type	Description
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.
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.

Continues on next page

Error code	Type	Description
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.
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.
8345	Error	Failed to start program in Auto. Possible reason: The RW role setting 'Remote start/stop program in Auto' is not selected.
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.

Continues on next page

Error code	Type	Description
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.
8419	Error	Update item target information
8420	Error	Update item target not supported information
8421	Status	Robot held information
8422	Status	Robot suspended information

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.

Continues on next page

Error code	Type	Description
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.
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

Continues on next page

Error code	Type	Description
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.
12343	Status	Detail vision running information
12344	Warning	No valid calibration selected for PatMax. Can not use camera height.
12345	Warning	No valid calibration selected for PatMax. Please revise models and/or calibrations.
12346	Warning	No valid calibration selected for Inspection. Please revise models and/or calibrations.
16985	Error	Get image failed form camera. Pleae check the parameter Settings and hardware connections.
16986	Error	Description: Unexpected error when use vision. Reason: An unexpected error occurred in PickMaster. See the log message for more information.
16987	Error	Description: Unexpected error when configure vision. Reason: An unexpected error occurred in PickMaster. See the log message for more information.
16988	Error	Camera configuration error. This feature is not available for the actual camera type.
16989	Error	White value error. See the log message for more information.
16990	Error	Camera video format is not support.
16991	Error	The region of interest is invalid. Whole image will be used.
16992	Error	The checkerboard calibration is not trained.

Continues on next page

Error code	Type	Description
16993	Error	Description: Unexpected error about calibration. Reason: An unexpected error occurred in PickMaster. See the log message for more information.
16994	Error	Calibration error. Failed to identify vertices in the checker-board image. Please check configuration.
16995	Error	Calibration error. No origin marker found.
16996	Error	Calibration error. Not enough vertices in the checkerboard image.
16997	Error	Calibration error. An origin image must be set to calibrate the camera.
16998	Error	Calibration error. The destination area is defined outside the source image.
16999	Error	Color filter setting error. Unable to apply all settings. Please review.
17000	Error	PatMax error. The vision license does not have color tools enabled
17001	Error	PatMax error. Bad parameters. Please check.
17002	Error	PatMax error. The search region is defined outside the image area.
17003	Error	PatMax error. The position limit is defined outside the image area.
17004	Error	PatMax error. Could not train the selected area.
17005	Error	PatMax model error. The model is not trained.
17006	Error	PatMax error. Search timeout.
17007	Error	PatMax error. Invalid search region.
17008	Error	PatMax error. The selected region does not fit entirely within the image.
17009	Error	Blob error. The search region is defined outside the image area.
17010	Error	Blob error. There must be at least one hit with the trained model to define a region.
17011	Error	Blob error. The histogram invalid search region.
17012	Error	SubPatMax error. See the log message for more information.
17013	Error	Line file error. Could not find specified camera in the given line file.
17014	Error	Description: Unexpected error when use camera. Reason: An unexpected error occurred in PickMaster. See the log message for more information.

User script error codes

Error code	Type	Description
41989	Message	The execution of {%s} in {%s} timed out.
41990	Message	The module of {%s} in {%s} load failed.
41991	Message	The interface of {%s} in {%s} load failed.

Continues on next page

Error code	Type	Description
41992	Message	The return of {%s} in {%s} is not correct.
41993	Message	The returned data structure of {%s} in {%s} is incorrect.
41995	Message	The element of {%s} was not found in returned data of {%s} in {%s}.
41996	Message	Position at [{%.1f}, {%.1f}] discarded due to unknown object Id. It is from {%s} in {%s}.
41997	Message	The element type of {%s} is incorrect in returned data of {%s} in {%s}.
42003	Message	Failed to obtain the documents folder path when {%s} was executed in {%s}, please check.

7.7 Fault symptoms or errors

7.7.1 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 457*, *Warning 4327 on page 457*, *Warning 4328 and 4329 received together on page 458*, *Warning 4328 received without 4329 on page 459*, and *Warning 4329 received without 4328 on page 459*.

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 networks on page 52*.

Action 4

See *Configuring networks on page 52*.

Action 5

Check that the IP address (goto **File** and click **Options** in RRT) 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

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 456*, *Action 2 on page 456*
The strobe signal is not configured as cXNewObjStrobe.	*Action 1 on page 456*
PickMaster has no connection with the robot controller.	*Action 3 on page 456*
If work area is indexed:	
The configured strobe signal does not receive a strobe pulses.	*Action 1 on page 456*, *Action 2 on page 456*
PickMaster has no connection with the robot controller.	*Action 3 on page 456*

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 456*, *Action 2 on page 456*
PickMaster has no connection with the camera.	*Action 4 on page 456*

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 456*, *Action 2 on page 456*
The external sensor does not send any positions to PickMaster.	*Action 8 on page 457*
If source type is external sensor:	
The external sensor does not receive any trigger pulses.	*Action 1 on page 456*, *Action 2 on page 456*
The external sensor does not send any positions to PickMaster.	*Action 8 on page 457*
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 456*, *Action 2 on page 456*
The strobe signal is not configured as cXNewObjStrobe.	*Action 8 on page 457*
PickMaster has no connection with the robot controller	*Action 3 on page 456*
If source type is predefined and work area is indexed:	
The configured strobe signal does not receive an strobe pulses.	*Action 1 on page 456*, *Action 2 on page 456*
PickMaster has no connection with the robot controller.	*Action 3 on page 456*

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 456*
The trigger frequency is set too high.	*Action 5 on page 456*
Low robot network performance	*Action 7 on page 457*
Low camera network performance	*Action 4 on page 456*
Additional causes for external sensors:	
Time stamps are not enough synchronized with strobes.	*Action 9 on page 457*
The external sensor does not send positions with a correct time stamp..	*Action 8 on page 457*

Continues on next page

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 456*

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 456*

7.7.2 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.

7.7.3 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.

7.7.4 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.

7.7.5 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.

7.7.6 Problem with camera resolution in PickMaster

Error description

Camera image size decreases to lower resolution as compared to calibration image resolution.

Probable causes

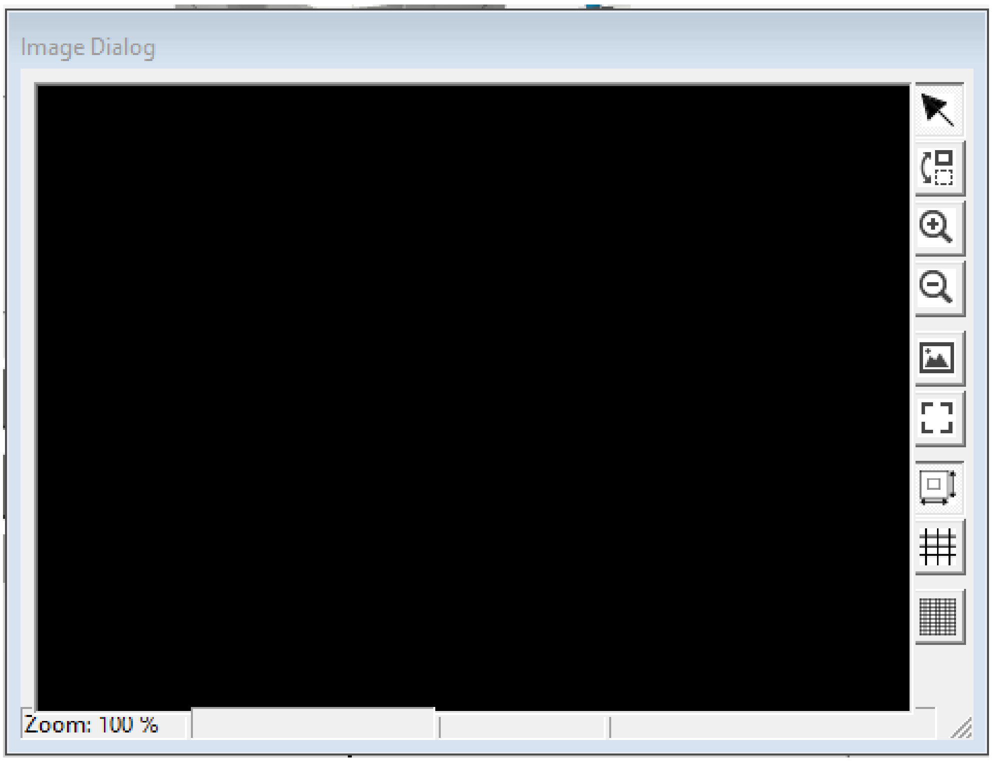
There can be several causes why camera resolution is decreased. To check all the possible causes the following must be verified:

- 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).

7.7.7 The Image Dialog cannot show

Error description

When users try to use camera related functions (camera configuration, camera calibration, geometric model, blob model, inspection model, live video, detail vision), the specific image dialog shown below cannot be displayed. Sometimes a "pure virtual function call" error pops up.

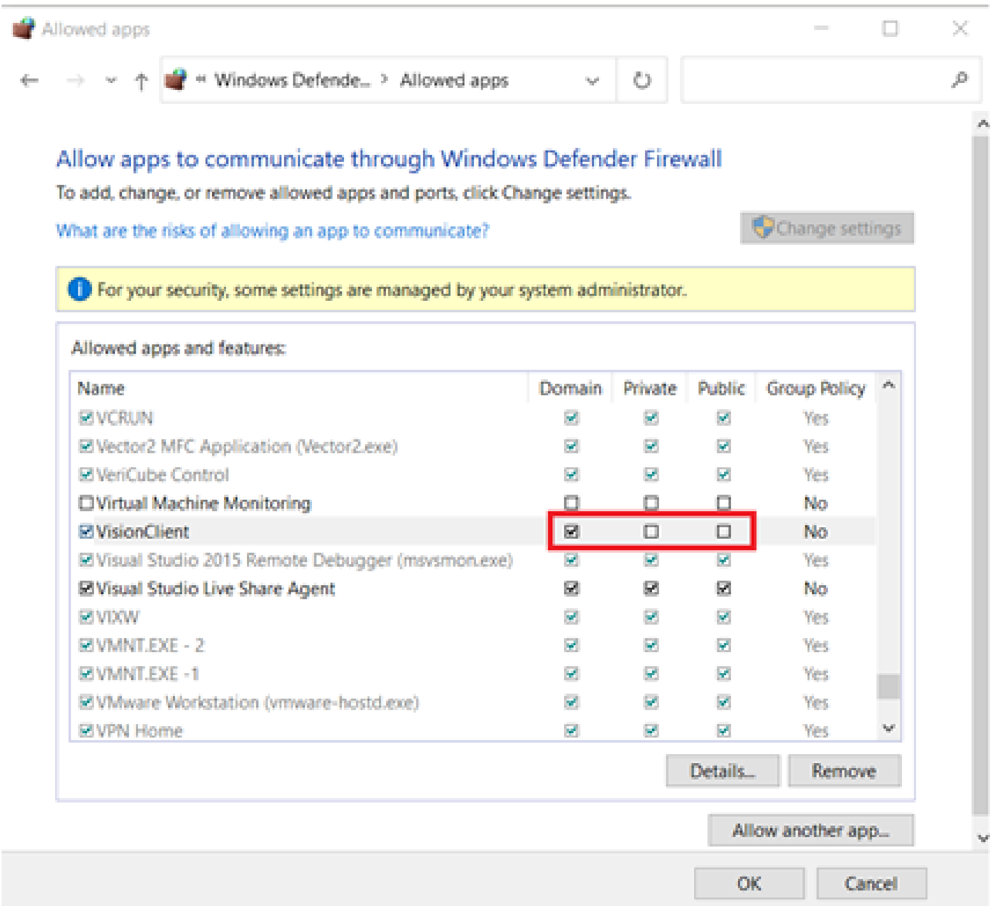


xx2200001057

Continues on next page

Probable causes

Windows firewall blocks **VisionClient.exe**, a camera function related software engine, in some networks. Users should check whether **VisionClient** is available within the network of the computer in the **Allowed apps** window. If not all network settings of **VisionClient** are selected, this issue can occur as shown in the following case.



xx2200001058

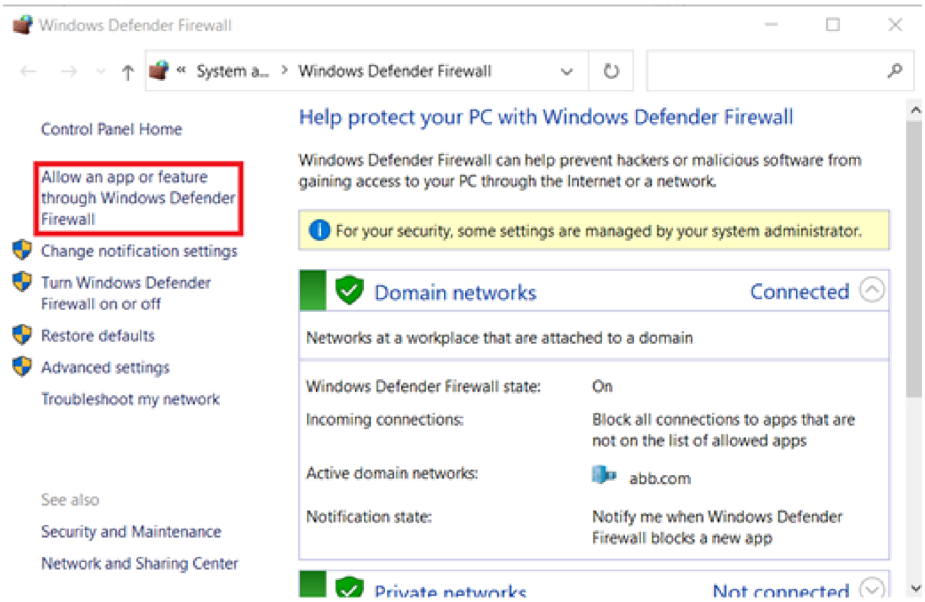
Recommended actions

The following procedure is recommended to change the firewall settings manually:

1 Open **Windows Defender Firewall**.

Continues on next page

2 Click **Allow an app of feature through Windows Defender Firewall** to open the **Allowed apps** window.

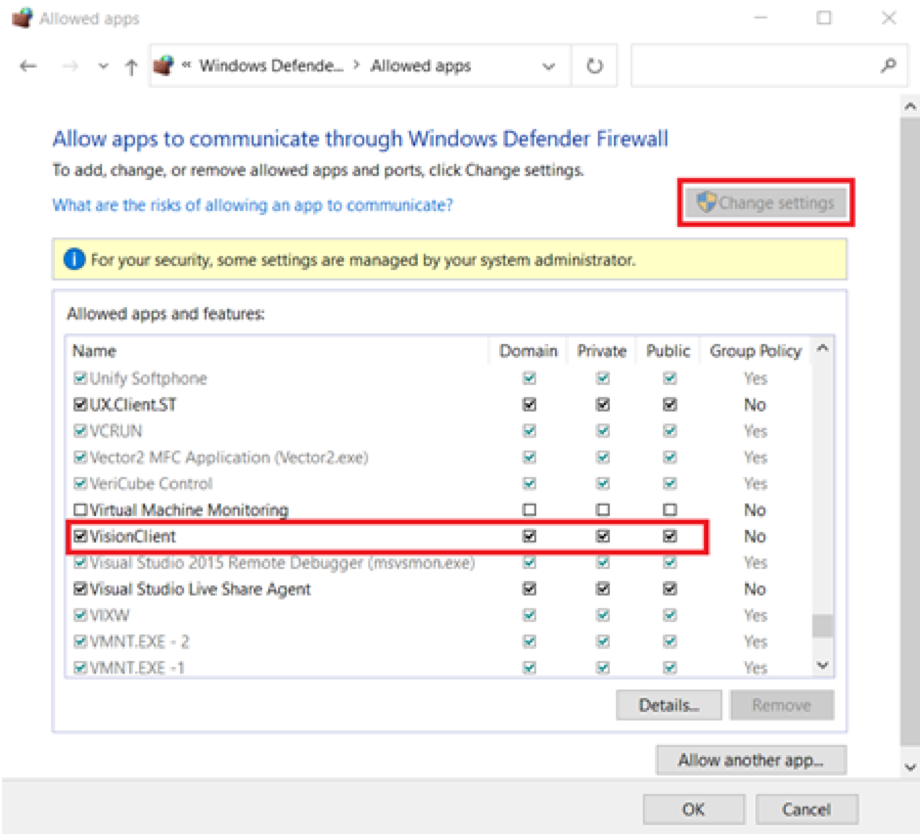


xx2200001059

3 Click **Change settings.**

Continues on next page

4 Find **VisionClient** in the list and check that all network checkboxes for all **VisionClient** or **visionclient.exe** apps are selected.



xx2200001060

5 Click **OK**.

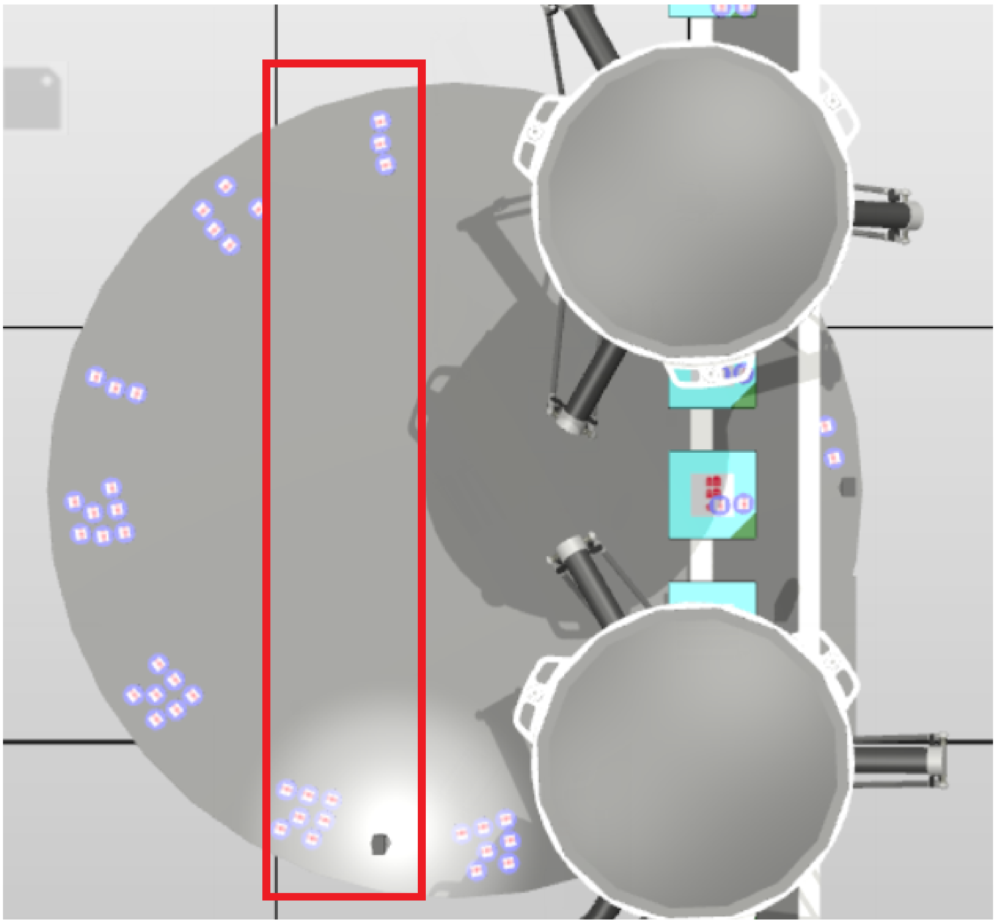
7.7.8 Robot fails to grip item when using camera on a circular conveyor

Error description

In a station with using camera(s) on a circular conveyor, the robot fails to grip item and error message of **Failed to grip item by tool 'PickPlaceTool_1'. Ignoring...** shows up in the log when run the production.

Probable causes

If **Enable vision width** is not enabled, the vision scope will include these items in the red circle showed in the following image. Then wrong position information will be sent to the robot, and the gripping error will occur.



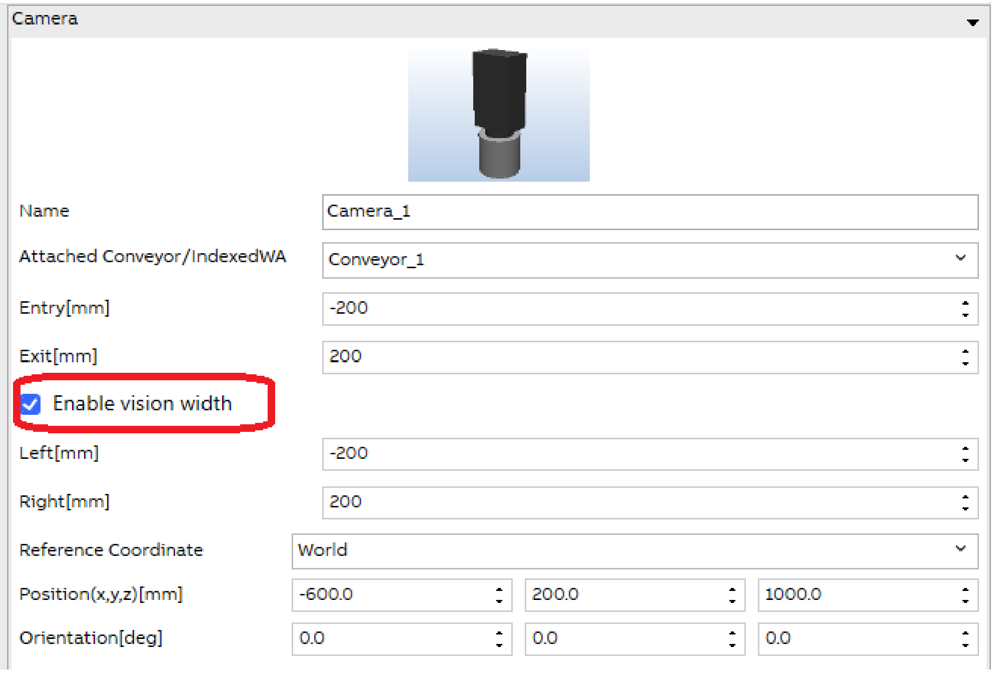
xx2200002019

When using camera(s) on a circular conveyor, **Enable vision width** is preferred to be enabled to limit the vision scope. Otherwise, this vision scope will cover items on the other side of circular conveyor.

Continues on next page

Recommended actions

Select to enable **Enable vision width** in the camera setting view if the camera is used on a circular conveyor.



xx2200002020

7.7.9 Robot fails to start when clicking on Start button after `Arm check point limit` error for an indexed work area

Error description

In a station with using indexed work areas, the robot fails to start directly when clicking on **Unhold** button in PackML or clicking on **Start** button after the error message of **Arm check point limit...** shows up in the log.

Probable causes

Index work area doesn't support this scenario.

Recommended actions

Stop the robot first and then click the **Start** button of the robot or the **Unhold** button in PackML.

This page is intentionally left blank

8 Spare parts

Spare part level

ABB spare parts are categorized into two levels, L1 and L2. Always check the part level before conducting a service work on a spare part.

- L1 spare parts

  The L1 parts can be replaced in the field. The maintenance and replacement instructions given in the related product manuals must be strictly followed. If there are any problems, contact your local ABB for support.
- L2 spare parts

  To replace the L2 parts require specialized training and might need special tools. Only ABB field service personnel or qualified personnel trained by ABB can replace L2 parts.
- L3 spare parts

  L3 spare parts shall only be replaced or repaired by qualified ABB service technician with knowledge of the application due to reduce risk of injury or damage to equipment. Improper installation may void warranty.

Continues on next page

8.1 Licenses

Spare part

	Spare part number	Description	Type	Spare part level
-	3HAC072144-001	PickMaster Runtime license		L1

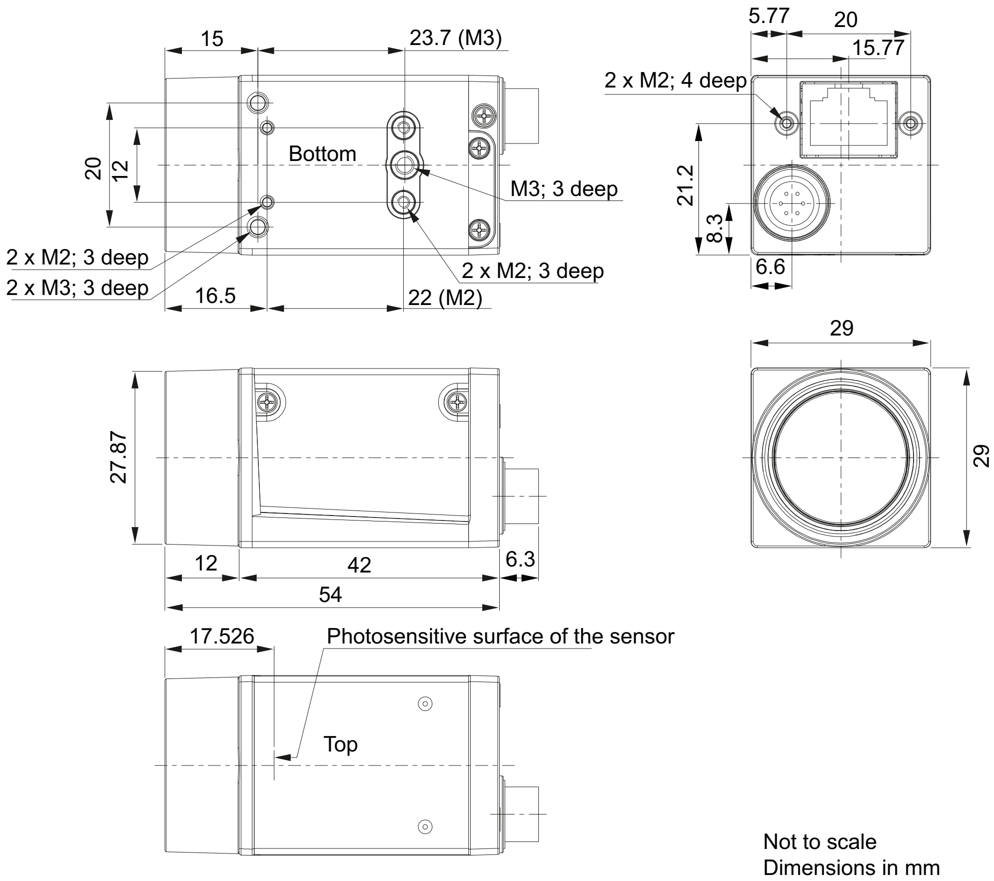
8.2 Camera parts

Spare part - PickMaster camera

	Spare part number	Description	Type	Spare part level
-	3HAC072140-001	PickMaster camera	DSQC1066	L1

xx1900001574

Continues on next page



xx2300001601

The Basler acA1440-73gc GigE camera 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*].

Continues on next page

Spare part - PickMaster cam I/O cable

	Spare part number	Description	Type	Spare part level
-	3HAC072141-001	PickMaster cam I/O cable		L1

xx2200000589

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 (without GPIO)	Ace GigEg (with GPIO)	Aviator CL runner
1	Brown	Camera Power	Camera Power	Camera Power
2	Pink	Opto-isolated IN (Line1)	Opto-isolated IN (Line1)	Camera Power
3	Green	Not connected	GPIO (Line3)	Not connected
4	Yellow	Opto-isolated OUT (Out1)	Opto-isolated OUT	Not connected
5	Gray	Opto-isolated I/O Ground	Opto-isolated I/O Ground	Camera Power Groud
6	White	Camera Power Ground	Camera Power and GPIO Ground	Camera Power Groud

Continues on next page

Spare part - PickMaster cam com cable

	Spare part number	Description	Type	Spare part level
-	3HAC072142-001	PickMaster cam com cable		L1

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.

Spare part - Camera mount adapter

	Spare part number	Description	Type	Spare part level
-	3HAC074680-001	Camera mount adapter		L1

Camera mount for Basler ace cameras.

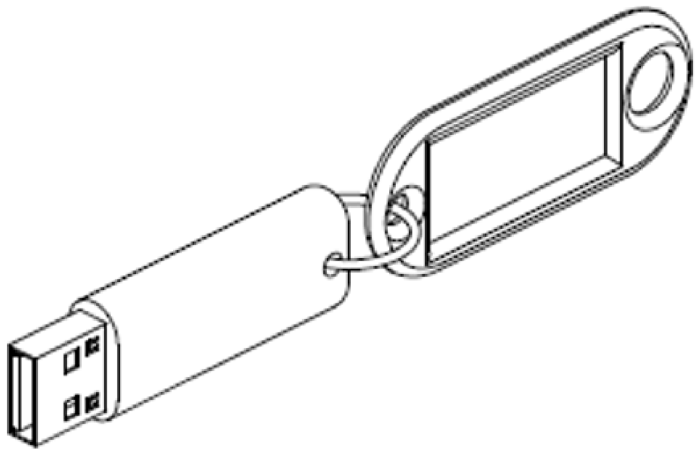
For mounting the camera onto tripod threads.

8.3 USB dongle parts

Spare part

	Spare part number	Description	Type	Spare part level
-	3HAC072139-001	USB dongle (small)[i]	Vision license for up to 2 cameras	L1
-	3HAC073341-001	USB dongle (large)[i]	Vision license for up to 10 cameras	L1
-	3HAC039556-001	USB dongle (sim)[ii]	Vision simulation license for up to 10 simulated cameras	L1

i The dongle can be connected to any USB interface on host computer.
ii The dongle can be connected to any USB interface on client computer.

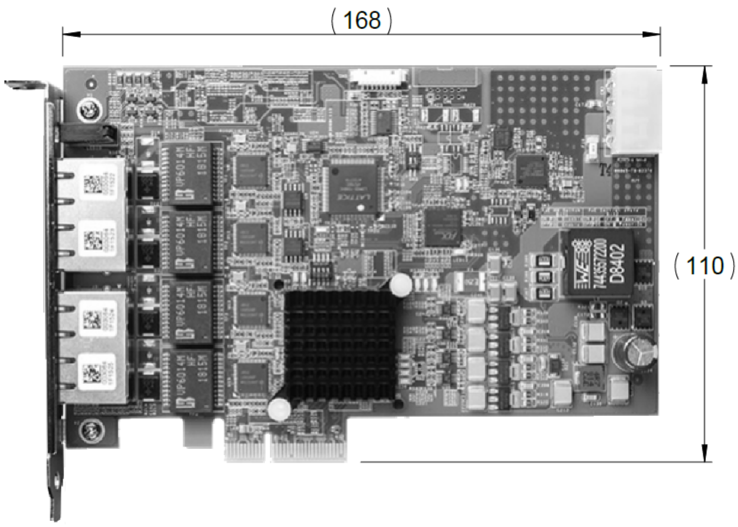


xx1900001747

8.4 GigE Network card parts

Spare part

	Spare part number	Description	Type	Spare part level
-	3HAC078753-001	GigE network card	DSQC1083	L1



xx2200000591

* Standard height, half length, PCI express card.

9 Circuit diagram

9.1 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 - OmniCore C30, *Circuit diagram - OmniCore C30 for IRB 14050*, *Circuit diagram - OmniCore C30 for CRB 15000*	*3HAC059896-009*, *3HAC063898-009*, *3HAC072448-009*
Circuit diagram - OmniCore C90XT	*3HAC065464-009*
Circuit diagram - IRC5	*3HAC024480-011*

RobotWare options

Product	Article numbers for circuit diagrams
Circuit diagram - PickMaster Twin	*3HAC024480-020*

This page is intentionally left blank

Index

ABB AB
Robotics & Discrete Automation
S-721 68 VÄSTERÅS, Sweden
Telephone +46 10-732 50 00

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

3HAC080435-001, Rev H, en