Operating manual
Palletizing PowerPac

RobotStudio 5.14.02

Document ID: 3HAC042340-001
Revision: -

ABB AB
Robotics Products
SE-721 68 Västerås
Sweden

Overview of the manual

About this manual

This manual contains information and instructions for installing, configuring, and running Palletizing PowerPac.

Usage

This manual should be used during installation and configuration of Palletizing PowerPac. It describes Palletizing PowerPac and includes step-by-step instructions on how to perform the tasks from there.

Who should read this manual?

This manual is mainly intended for:

?tSystem integrators

?tABB engineers

?tEnd customers

Prerequisites

The reader should:

?tHave experience with RobotStudio

?tHave experience of installation and configuration work

?tGood skills in the IRC5 robot controller and RAPID programming

Organization of chapters

The manual is organized in the following chapters:

Chapter	Contents
1. Introduction	Introduces Palletizing PowerPac, and palletizing process and terms that are used in following chapters.
2. Installtion	How to install the software and pre-requisites for installa-tion.
3. Workflow for Palletizing PowerPac	Describes Step-by-step procedure to use Palletizing PowerPac.
4. Navigating Palletizing PowerPac	Describes in detail each function of Palletizing PowerPac.
5. RAPID Program	How RAPID programs are used in PickMaster. Program examples for palletizing applications.
6. Runtime Operation	How PickMaster palletizing projects are started, controlled, and supervised on a robot controller.
7. RAPID Reference Information	Descriptions of RAPID instructions, functions and data types, that are specific for PickMaster.
8. Relationships Between Frames	Describes the different frames used by PickMaster and how they relate to each other.

Continues on next page

Continued

References

References	Document ID
Operating manual - IRC5 with FlexPendant	3HAC16590-1
Operating manual - RobotStudio	3HAC032104-001
Operating manual - Trouble shooting	3HAC020738-001
Technical reference manual - RAPID Instructions, Functions and Data types	3HAC16581-1
Technical reference manual - RAPID overview	3HAC16580-1
Technical reference manual - RAPID kernel	3HAC16585-1
Technical reference manual - System parameters	3HAC17076-1
Product manual - IRC5	3HAC021313-001
Product manual - Panel Mounted Controller	3HAC027707-001
Product manual - IRB 260	3HAC026048-001
Product manual - IRB 460	3HAC039842-001
Product manual - IRB 660	3HAC025755-001
Product manual - IRB 760	3HAC039838-001
Product specification - PickMaster 5	3HAC5842-9

Other references

References	Description
http://www.robotstudio.com/forum/	PickMaster Support Forum

Revisions

Revision	Description
-	First edition

Safety

Safety of personnel

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

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

?tUnits inside the controller, for example I/O units, can be supplied with power from an external source.

?tThe mains supply/mains switch.

?tThe power unit.

?tThe power supply unit for the computer system (230 VAC).

?tThe rectifier unit (400-480 VAC and 700 VDC). Capacitors!

?tThe drive unit (700 VDC).

?tThe service outlets (115/230 VAC).

?tThe power supply unit for tools, or special power supply units for the machining process.

?tThe external voltage connected to the controller remains live even when the robot is disconnected from the mains.

?tAdditional connections.

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

Safety regulations

Before beginning mechanical and/or electrical installations, ensure you are familiar with the safety regulations described in *Product manual - IRC5*.

1 Introduction

1.1. Overview

Structure of this chapter

This chapter gives an overview of the Palletizing PowerPac software, and includes the following:

?tThe Palletizing cell concept

?tA description of the palletizing process

?tA terminology list containing PickMaster and specific palletizing terms

1.2. What is Palletizing PowerPac

Overview

Palletizing PowerPac (PzPP) is part of the PickMaster solution for controlling ABB robots in palletizing applications via the IRC5 robot controller. The PowerPac replaces the original ABB PickMaster PC.

PickMaster is designed to handle one or more cells in the production. It is a modular product, which can be customized to your special needs:

?t PickMaster RC is a stand-alone robot kernel, running the process in production. It communicates through the RAPID program, I/O interface, and FlexPendant interface.

?tPzPP allows you to make the configuration and simulation for a palletizing application and process.

Prerequisites

PzPP can be installed on a computer running Windows XP, Windows Vista, or Windows 7. RobotStudio should also be installed before.

In order to work with real controllers, the computer must be connected to a controller over an Ethernet network.

For minimum system requirements, see Product specification - PickMaster. The Prepared for PickMaster option, together with the PickMaster 5 sub-option, are the required RobotWare software options to use PickMaster on the IRC5 controller.

Continues on next page

Palletizing PowerPac functionality

The PowerPac contains:

?tThe Build Tool Function and Tool Events Configuration for configuring a robot tool model. See *Create Tool on page 102*.

?tThe Tool I/O Configuration for configuring the I/O connection between robot tool and controller. See *Edit Tool Signals on page 152*.

?tThe Build Feeder Configuration for configuring the feeder model properties, including hotspots. See *Create Feeder on page 124*.

?tThe Feeder Configuration for configuring the feeder properties, including signals, work objects, feeder CAD models, etc. See *Feeder on page 154*.

?tThe Product/Pallet/Sheet Configuration for configuring the product/pallet/sheet that will be picked and placed by the robot. See *Product/Pallet/Sheet on page 51*.

?tThe Pallet Pattern Configuration for configuring the stack of products organized in different layouts. See *Pallet pattern on page 62*.

?tThe Layout Editor for creating new or modifying existing layouts. See *Layout Editor on page 65*.

?tThe Pick Setting Configuration for configuring the products and tool orientation, as well as the tool functions to activate when gripping the products. See *Pick Setting on page 68*.

?tThe Operation Set Configuration for configuring the group operation set and pattern operation set. See *Operation Set on page 163*.

?tThe Job Wizard Configuration for configuring a new job including pattern/stack operation sets and/or group operation sets. See *Add Job on page 73*.

?tThe Flow configuration for defining accessible feeders in runtime. See *Flow on page 184*.

?tThe Tuning for changing parameter values online from the FlexPendant. See *Tuning on page 249*.

?tThe Project Settings for defining project name, descriptions, restart options and tune limitations. See *Project Settings on page 138*.

?tThe Feeder Order Settings for defining feeder order in the cell layout. See the Feeder Order Settings page in *Controller Settings on page 143*.

?tThe Event Settings for defining signals that can report events from external devices. SeeThe Event Settings page in *Controller Settings on page 143*.

?tThe Message Settings for defining messages that can be reported from external devices. See The Message Settings page in *Controller Settings on page 143*.

?tThe Robot Settings for modifying motion limits and RAPID programs for each robot. See *Robot Settings on page 149*.

?tThe I/O Interface editor for assigning I/O values to products, product groups, pattern operation sets, feeders, flows, and projects. See *I/O Interface on page 97*.

?tThe Palletizing Library for saving products, layouts, I/O connections and messages. See *Library on page 101*.

?tThe Preview Palletizing function to go through each palletizing robot targets for user validation. See *Preview Palletizing on page 80*.

Continues on next page

Continued

?tThe Simulation function to simulate the palletizing application. See *Start Simulation on page 82*.

?tThe Download and Upload function to synchronize programs between RobotStudio and robot controllers. See *Download on page 93* and *Upload on page 95*.

1.3. The palletizing process

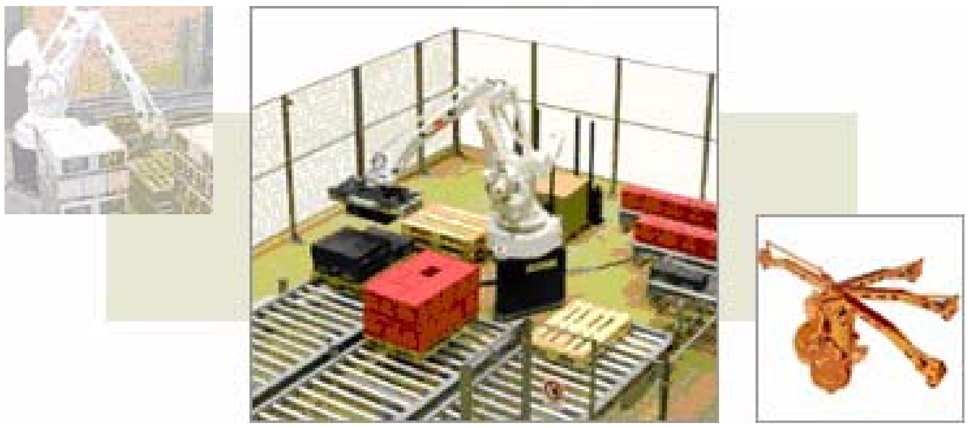
Overview

A palletizing application aims at picking larger size objects from a fixed position and stacking them tightly together in a second fixed position. An important parameter for the palletizing process is the speed of the process, that is the throughput of products in time and the efficiency in stacking the products in a stable configuration without taking up too much space.

After the palletizing process the stacks are loaded into containers or trucks, and the less space the products require, the less transportation costs are involved.

The palletizing cell

The figure illustrates an example of a palletizing cell.



xx110000001

In a palletizing cell, the robot is used for the following tasks:

?tPicking and placing one or more products.

?tPicking and placing slip sheets from a slip sheet stack station to pallet stations. This task is optional.

?tPicking and placing pallets from a pallet stack station to pallet stations. This task is optional.

When working with the optional tasks described above, the robot has to be able to pick the objects off a varying and initially unknown stack size. This is solved by automatically searching the height of the stack, usually with a sensing device in the robot gripper for the first approach and then keeping track of the stack height. For more information, see *Stack search on page 175*.

If the robot is not handling the pallets, they are moved into position by a feeder working in two directions, by AGVs or manually by fork lifts.

The palletizing cells

All goods produced by a factory pass through the palletizing cells before shipping to customers. This means that there is a large number of different products, which have to be guided to the right destination for accumulation. The most common shapes of products are a variety of carton boxes followed by bag types, but increasing numbers of open recycling crates are shaped for tight stacking.

Continues on next page

How to pack the products

The way the products are packed is solved by using optimal layer layouts, and a variety of layouts to build stability in the complete stack. The various layouts can be achieved by using different layouts every second layer or by simply rotating or mirroring the same layout for every second layer.

Further common practice to stabilize the stack and protect the products is to use slip sheets between the layers. The slip sheets are thin cardboard sheets and they can be placed anywhere between the layers, but mostly they are evenly distributed. Slip sheets can also be placed at the bottom and on the top.

How to speed up the process

For the palletizing process to be fast, the robot itself has to be fast and it has to be able to take more than one product at a time. The simplest way is to take boxes in groups and to place them in the same configuration in one drop. However, this reduces the universal flexibility of the robot. It is usually used for half and full layer palletizing, where the layouts are simple and very high throughput is required, often also in retrofits of older hard automated palletizers.

To plan each layout

A more flexible and efficient way is to plan each layout to be processed as efficiently as possible, which usually means as few operations as possible with a limited number of boxes at a time. If a specific product group requested by the pallet stack is not possible to place in one target, the placing operation has to be split into a number of separate placing targets, releasing one box at a time, with the possibility to rotate each box separately if needed. This is referred to as single pick, multiple placing.

Infeeders, outfeeders, and logical devices

To handle many products and pallet loads simultaneously, the installations use multiple infeeders and outfeeders gathered around the robot and logical devices to order the correct products to the robots. Different products have different production cycle durations and any order can be stopped and switched to another at any time, while other orders continue to operate without being affected.

The robot can move between different stacks

During the palletizing process the robot has to be informed about the next product group to pick and where to get it. When an operation is completed, another station can request the robot. In this way the robot has to move constantly and dynamically between different combinations of stacks.

Continues on next page

Continued

The Flow concept

PickMaster introduces the unique concept of *Flow* which is a built-in and automated intelligent order sequence distribution for the robot to act upon. Thereby, no advanced PLC program is required to control the logics of the application.

Execution of a palletizing job

A palletizing job with the flow concept can be described in the following steps:

1. A PLC requests a palletizing job to be done by a robot on a palletizing station. For example build a pallet consisting of seven layers of boxes on outfeeder 1.
2. While executing the job, the robot will step by step request the PLC to feed pallets, products, and slipsheets . For example the robot requests a box to be fed on infeeder 1, or a new stack of slipsheets to be placed at the slip sheet station.
3. When a palletizing job is ready, the robot communicates to the PLC that the job has been completed. For example a full pallet is ready on outfeeder 1.
4. The palletizing station is prepared for a new job to be started. For example the PLC sends the pallet away. A new palletizing job can now be requested by the PLC to be started on the palletizing station.

Sequence of palletizing jobs

Palletizing jobs are configured using the graphical user interface in the PickMaster 5 PC application. Many different jobs can be executed by the robot in a sequence that does not have to be decided in advance. A palletizing job can be modified while executing, for example finishing a job before it has been completed.

Parallel palletizing jobs

The flow concept allows a robot running multiple palletizing jobs in parallel, using one palletizing station for each job. The jobs run independently of each other but can share common resources, for example one infeeder can feed products to several parallel jobs.

The robot can switch between different jobs after each pick-place cycle. If a job not is ready to run after a pick-place cycle (for example the next box to be picked cannot be fed since an error has occurred on the infeeder) the robot continues to work with the other jobs until the error has been resolved. In this way, the productivity of the robot can be maximized.

Running parallel jobs does not add any complexity to programming or operator interaction.

Related information

Terms and concepts on page 18

1.4. Terms and concepts

About these terms

Some words have a specific meaning when used in this manual. This manual's definitions of these words are listed below. Some of the terms are put in their context when describing a palletizing process. See *The palletizing process on page 15*

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

Term list

Term	Definition
Action	Description of one robot movement to get to/on/from a *target*. Every action can have a number of *events*. For more information, see *Relationship between RAPID execution and PickMaster project on page 188*.
Activator	An I/O controlled part of a robot tool, normally a vacuum cup. For more information, see *Create Tool on page 102*
Event	A description of an event on the robot path, for example setting an I/O signal. For more information, see *Relationship between RAPID execution and PickMaster project on page 188*.
Facing	Possibility to select one or more sides of an *item* that should be facing outwards in a *pallet pattern*.
Feeder	A PickMaster's representation of pick and place areas. Often referred to as station, infeeder, or outfeeder. It defines the signals needed to control the in- and outfeeding of groups and pallet patterns on the feeder. Palletizing jobs can only be started on master feeders, see *Flow on page 184*. The feeding on a slave feeder is dictated by one or more master feeders. A feeder also holds all group operation sets and all pallet pattern operation sets. It also contains general robot movement data for the feeder, for example, safe positions. For more information, see the *Feeder on page 154* Configura-tion .
Flow	Logical directions of *items* being moved between pick and place stations when performing a palletizing/depalletizing job. A flow: ?tContains one master station on which palletizing/depallet-izing jobs can be started. The master station dictates the operation sequence for itself and its slave stations. ?tContains one or more slave stations. The slave stations execute operations requested by the master station. For more information, see *Flow on page 184* .
Group	Describes a group of products that can be picked/placed by a robot tool in one operation
Group operation set	One operation that describes how to pick/place a group on a specific feeder. For more information, see the *Operation Set on page 163* Configuration .

Continues on next page

Continued

Term	Definition
Hotspot	A special frame on feeder model that can be used to attach a work object. It also describes the position relationship between an item and the work object.
Item	The generic term for a specific object to be picked or placed. An item can be a product (i.e. box or bag), pallet, or slip sheet. It contains information of geometry size, weight, I/O value, and programming information like pick/place time and motion limitation. For more information, see the *Product/Pallet/Sheet on page 51* Configuration . Box: A rigid box with rectangle shape Bag: Same as box, but the bag has margins on length and width sides, specifying how much distance the bags can be overlapped with each other in a layout. Sheet: A thin sheet that is placed on a pallet before palletizing of products and between two layers to increase stability in the stack. Pallet: The actual wooden or plastic structure that the products are placed on.
Job	A job is a supported program (i.e. Operation Set) on a flow’s master feeder.
Layout	Defines the arranged two dimensional pattern of the *shapes* in a layer.
Master Feeder	The master station of a flow
Operation	An operation describes what a robot shall do when entering a *work area* for a new pick or place. What to do is described as a list of *targets*. For more information, see *Relationship between RAPID execution and PickMaster project on page 188*.
Pallet pattern	Defines a stack of *shapes* organized in different *layouts*. A layer in the stack can either be of *pallet*, *slip sheet*, or *product* type.
Pallet pattern operation set	A sequence of *operations* that describe how to pick/place a *pallet pattern* on a specific feeder.
Pick Setting	Defines how one item, or a group of items can be picked and placed by a robot tool in one operation. In PickMaster 5, this information was contained in Format. The Format concept in PickMaster 5 is separated into concept of Group, and Pick setting. For more information, see the *Pick Setting on page 68* Configuration .
Project	Description of a whole palletizing station, including the cell setup, and programs. For more information, see *Project Overview on page 96* .
Robot Target	A single robot target that robot goes to during picking and placing. An action contains one robot target.
Safe Target	A position that the robot always has to pass through when going to or leaving a feeder
Slave Feeder	A slave station of a flow
Stack	An arranged pile of items consisting of a number of layers.

Continues on next page

Continued

Term	**Definition**
Target	A target describes a robot position used when performing an operation on a feeder. A target has a list of products carried by the robot tool and a list of actions that shall be executed.Note the difference with robot target: a target can have a list of actions, and each action has one robot target.
Tool Function	A single function of a tool that requires input signals to control and will generate results as output signals.
Tool Scenario	One grouping of tool functions to use in all actions of a target during picking or placing.
Work area	The original term used in PickMaster 5 PC to describe feeders, now it is only used in RAPID functions and variables.
Zone	A vacuum tool function is divided into a number of zones. Each zone has one or more activators.

1.5. Using a robot with 6 axes

Overview

Even though running PickMaster 5 applications on a robot with 6 axes is possible, some manual modifications might be required depending on the robot configuration that is used.

The configuration of **SingArea** determines if you have to do manual modifications before running PickMaster 5 on a robot with six axes. This section describes the required modifications depending on the robot configuration.

The following scenarios are described for a bending backwards robot with 6 axes:

?tAxis 6 is pointing down. See *Axis 6 points down on page 21*.

?tThe wrist is tilted. See *Wrist is tilted on page 22*.

?tThe robot is bending backwards. See *Robot is bending backwards on page 23*.

For using a parallel rod robot with 6 axes together with PickMaster 5, see *Parallel rod robot on page 25*.

Configuration of **SingArea**

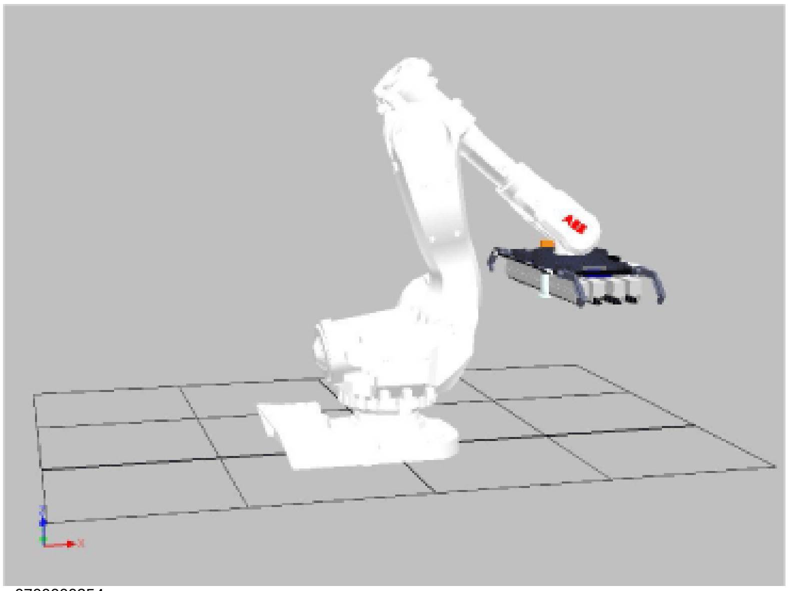
Linear movements are by default configured to use **SingArea \Wrist** path interpolation mode. This setting is configured in the **Operation Set** dialog boxes. See *Group Operation Set on page 165* and *Pattern/Stack Operation Set on page 167*.

Bending backwards robot

This section illustrates the different scenarios for a bending backwards robot, and describes the required manual modifications.

Axis 6 points down

The figure illustrates axis 6 pointing down. In this position the robot works in a similar way as a floor mounted palletizer robot with 4 axes, always having the mounting interface (and the tool) directed downwards in a horizontal orientation. No bending backward positions are used.



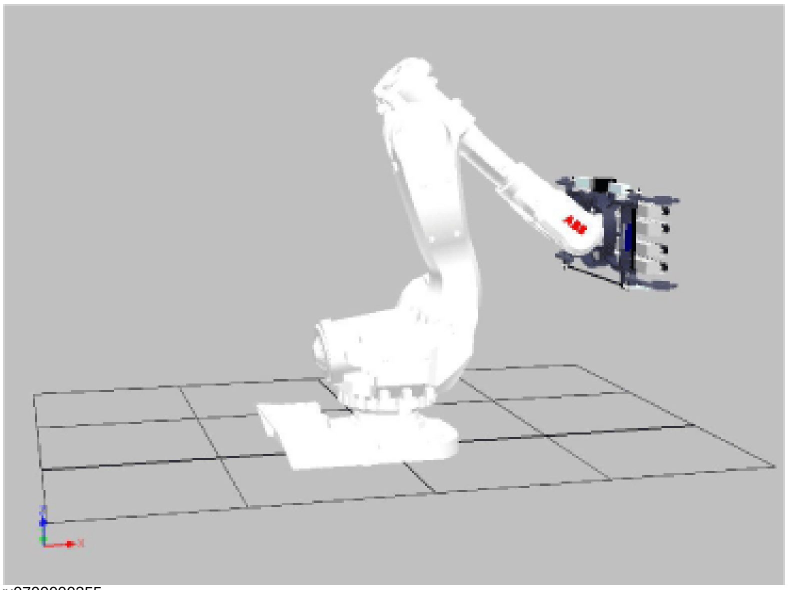
xx0700000354

Continues on next page

Configuration	Description
Using **SingArea \Wrist**	Long linear movements causes varying tool orientation. Tool orientation deviates between targets when using **SingArea /Wrist** interpolation. The deviation is small for short movements but increases with longer distance between robot targets.
Using **SingArea \Off**	See *Manual modifications on page 23*.

Wrist is tilted

The figure illustrates the tilted wrist. In this position the robot uses other tool orientations than a robot with 4 axes for some work areas, for example wall mounted work areas. No bending backward positions are used.



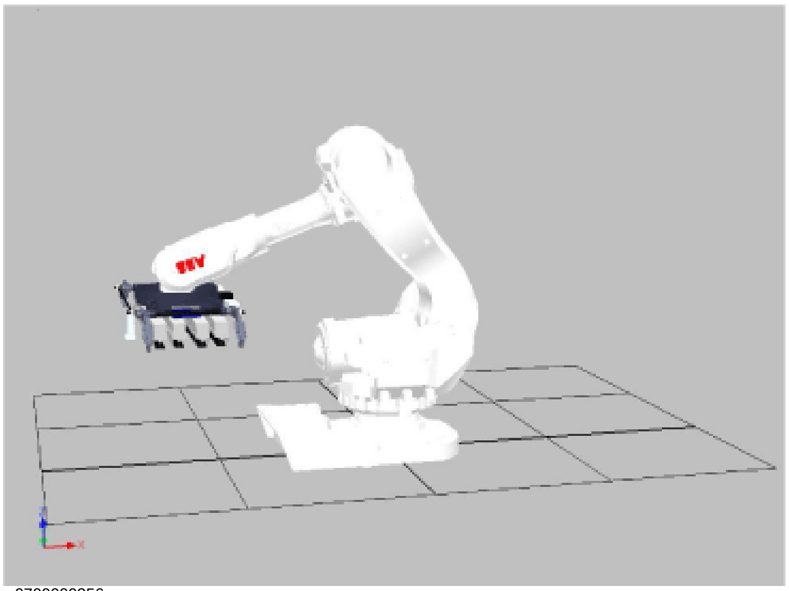
xx0700000355

Configuration	Description
Using **SingArea \Wrist**	Not recommended.
Using **SingArea \Off**	See *Manual modifications on page 23*.

Continues on next page

Robot is bending backwards

The figure illustrates the robot bending backwards. In this case the robot uses bending backward movements to reach some work areas. The tool can have various orientations.



xx0700000356

Configuration	Description
Using **SingArea \Wrist**	Not recommended.
Using **SingArea \Off**	See *Manual modifications on page 23*.

Manual modifications

When the robot configuration is set to **SingArea \Off**, the following modifications are required for robot to run properly:

	Action	Note
1.	Use **ConfL \Off**. Update the system module *pmrcUser*, with the procedure `PmDoAction` from system module *pmrcSys* and rename the procedure to `DoAction`. Modify `DoAction` according to description in section *System module pmrcUser on page 24*.	Linear motion supervision must be turned off to allow movements with large axis reorientations (> 90°).

Continues on next page

	Action	Note
2.	Use a robot position for each work area with `MoveJ`. In PzPP browser, record a safe position for each feeder, and use MoveJ for each position. See Safe Targets *Set WorkObject Position on page 162*.	First ensure the angles of axis 4 and 6 do *not* reach the physical limitation. Then ensure the robot has the correct arm configuration when operating on the work area.
3.	Use **ConfJ \On**. Turn on the configuration control for joint movements. Update the system module *pmrcUser*, with the procedure `PmDoAction` from system module *pmrcSys*, and rename the procedure to `DoAction`. Modify `DoAction` according to description in section *System module pmrcUser on page 24*.	First ensure the angles of axis 4 and 6 do *not* reach the physical limitation. Then ensure the robot has the correct arm configuration when operating on the work area.

System module *pmrcUser*

Modify the system module *pmrcUser*, with the procedure `DoAction` copied from procedure `PmDoAction` in system module *pmrcSys*.

```
...
IF Act.ArmConfMon = TRUE THEN
  ConfL\Off;
  ConfJ\On;
ELSE
  ConfL\Off;
  ConfJ\On;
ENDIF
...
```

You find the complete program code for the system modules *pmrcUser* and *pmrcSys* in section *Program code on page 213*.

Continues on next page

Continued

NOTE!

The following modifications must be done in the program code:

```
?tConfL\Off
?tConfJ\On
```

NOTE!

The procedure `Operate` in program module `PmMain` must be updated with a call to `DoAction` instead of `PmDoAction`.

Parallel rod robot

When using a parallel rod robot, the same modifications need to be done as described for the bending backwards robot. For a parallel robot only the scenarios *Axis 6 points down on page 21* and *Wrist is tilted on page 22* are possible.

Following modification needs to be done in addition to what is described for bending backwards robots:

RAPID calls to `PMCalcArmConf`

All RAPID calls to the routine `PmCalcArmConf` must use the optional argument `\TypeB1` instead of `\cf6`.

pmMain.mod

Modify the program module *pmMain.mod*, procedure `Operate`:

```
...
PmCalcArmConf
     Act.RobTgt,Tgt.TargetTool,Tgt.TargetWobj\TypeB1\MaxAngle:=
     MaxToolAngle\MinAngle:=MinToolAngle;
...
```

Related information

Group Operation Set on page 165.

Pattern/Stack Operation Set on page 167.

Public system module pmrcUser on page 212.

2 Installation

2.1. Installation

Overview

This section describes the installation process.

Prerequisites

To start the installation process, the following must be available:

?tA computer that meets or exceeds the system requirements.

?tA log on account with administrator rights on the computer.

?tRobotStudio installed on the computer.

?tPzPP installation package.

System requirements

High performance desktop or laptop workstation:

?tCPU: 2.0 GHz or faster processor

?tMemory: 1 GB system memory at minimum, 2 GB if running Windows Vista, stations with several robot systems, or large CAD-models.

?tFree disk-space: 5+ GB free space

?tGraphics card: High performance DirectX 9 or OpenGL-compatible graphics card with the corresponding up-to-date drivers installed

?tDisplay settings: Screen resolution:1280 x 1024 pixels or higher

?tDPI: Normal size (96 dpi)

Network setting

The information in this topic describes the network settings for a computer to be connected to a robot controller. Connecting the computer to the robot controller is necessary for working with PickMaster.

Ethernet network

The PickMaster computer and the robot controller communicate via Ethernet. Please contact the local network administrator for the network settings if the system is connected to an already existing network.

New local area network

If a new local area network (LAN) is created especially for PickMaster, the following settings can be used:

IP addresses	192.168.1.X (where X is between 1 and 253)
Gateway	192.168.1.254
Subnet mask	255.255.255.0

Use static IP numbering with different addresses for both the computer and the robot controller.

Continues on next page

NOTE!

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

NOTE!

To be able to install PzPP, RobotStudio must be installed on your computer.

Installing Palletizing PowerPac

To install the PzPP, follow these steps:

1. Browse to PzPP installation package and double-click Setup.exe. The installation starts.
2. Read the License Agreement and accept the terms.
3. Click Install.
4. When the installation is finished, complete the installation wizard by clicking Finish.

Installing a license

To install a license of RobotStudio and PowerPac, see Operating manual - RobotStudio.

3 Workflow for Palletizing PowerPac with PickMaster

3.1. Introduction

The following is a recommended working procedure for PzPP with PickMaster.

3.2. Preparing your controller for PickMaster

Prerequisites

To run a PickMaster application you must prepare your robot controller for PickMaster. You create and install a system for the robot controller using RobotStudio.

The option Prepared for PickMaster, with the sub-option PickMaster 5, is needed to run PickMaster on an IRC5 robot controller.

Preparing your controller for PickMaster

Use this procedure to prepare the controller for PickMaster:

	Action	Note/See
1	Create and install a system for the robot controller using RobotStudio.	Operating manual - RobotStudio.
2	Setup RAPID tool data for the robot tool. Use RobotStudio or the Flex-Pendant.	The tool data will be created and downloaded in PzPP when certain tool is attached to the virtual robot. It is often convenient to define one tool data for picking/placing and another one to be used when calibrating work objects. If the robot tool supports stack search a separate tool data is created for that purpose. Technical reference manual - RAPID Instructions, Functions and Data types, section tooldata - Tool data.
3	Calibrate the work object for each feeder. Use the FlexPendant.	Technical reference manual - RAPID Instructions, Functions and Data types, section wobjdata - Work object data.Operating manual - IRC5 with FlexPendant.
4	Add tool related signals. Use RobotStudio or the FlexPendant.	See *Edit Tool Signals on page 152* for information about the signals.Operating manual - RobotStudio.Operating manual - IRC5 with FlexPendant.
5	Add feeder related signals. Use RobotStudio or the FlexPendant.	See *Edit Detailed Signals for Feeder on page 157* for information about the signals.Operating manual - Robot-Studio.Operating manual - IRC5 with FlexPendant.
6	Add event related signals if you want an external equipment to report errors and/or messages to the PickMaster application. Use RobotStudio or the FlexPendant.	The Controller Settings window .Operating manual - RobotStudio.Operating manual - IRC5 with FlexPen-dant.

3.3. How to add tool

On the Palletizing ribbon-tab, click *Add Tool* to import a tool for the palletizing process.

See *Add tool on page 47* for detailed description.

3.4. How to add feeders

On the Palletizing ribbon-tab, click *Add Feeder* to import a feeder and add to a robot for the palletizing process. Continue to add other feeders. At least two feeders are required for each robot. See *Add feeder on page 49* for detailed description.

3.5. How to create or modify product data

On the Palletizing ribbon-tab, click *Product/Pallet/Sheet* to create or modify items for the palletizing process.

See *Product/Pallet/Sheet on page 51* for detailed description.

On the Palletizing ribbon-tab, click *Pallet Patterns* to create or modify patterns for the palletizing process.

See *Pallet pattern on page 62* for detailed description.

3.6. How to set pick setting

On the Palletizing ribbon-tab, click *Pick Setting* to define how products from an group or a stack are picked by a robot tool.

See *Pick Setting on page 68* for detailed description.

NOTE!

This function is only available when at least one item is created, and one tool is attached to at least one robot.

3.7. How to add job

On the Palletizing ribbon-tab, click *Add a Job* to create a job for palletizing process.

See *Add Job on page 73* for detailed description.

3.8. How to do simulation

On the Palletizing ribbon-tab, click *Start button* to start simulation, Stop button to stop simulation and Reset button to clean the temporary objects generated in the previous simulation.

See *Start Simulation on page 82* for detailed description.

3.9. How to adjust cell layout to achieve reachability

If Check Reach or Simulation has reported unreachable targets, you usually have following ways to adjust the cell:

?tMove feeders: you can enable the Freehand Move, click on the feeder model in the station, and drag it to an appropriate position. You can also use the Set WorkObject Position to change the feeder location.

?tMove robot: you can also move the robot by Freehand Move or by Set Position of the corresponding robot model.

After you have changed robot position, a message box will show up to ask you whether to update the Task Frame:

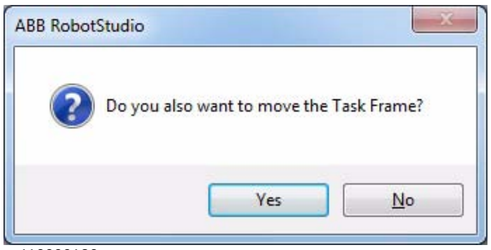


ABB RobotStudio

Do you also want to move the Task Frame?

xx110000180

Choose “Yes” to update Task Frame, otherwise, the controller needs to be warm started.

Then a further message box will ask you to keep the positioning of all stationary RAPID variables (e.g. work object):

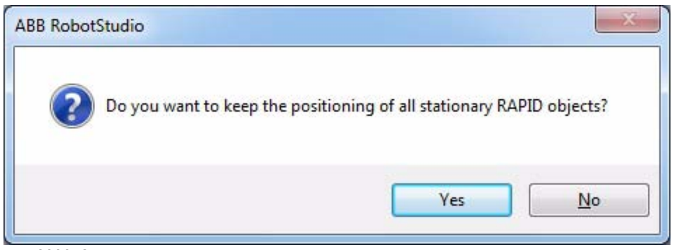


ABB RobotStudio

Do you want to keep the positioning of all stationary RAPID objects?

xx110000181

Choose “Yes” to keep all work object values unchanged, otherwise the work objects will move together with the robot.

3.10. How to transfer the project

On the Palletizing ribbon-tab, click *Add Controller* to connect to online controllers, and Download to check the destinations and transfer the project onto the online controllers.

See *Add Controller on page 92* and *Download on page 93* for detailed description.

3.11. How to start production

Introduction

To run a PickMaster project in production you can use the PickMaster FlexPendant interface.

Starting production

Use this procedure to start a PickMaster project.

	Action	See
1.	Start the PickMaster FlexPendant interface.	
2.	Select the project you want to run.	*Opening a project on page 232*.
3.	Start the project.	*Starting a project on page 233*.
4.	Start flows.	*Starting a specific flow on page 234* and *Starting and stopping all flows on page 236*.

Related information

FlexPendant interface on page 230.

Opening a project on page 232.

Starting and stopping production on page 233.

3.12. How to send my project to get support from ABB

To send your project to ABB engineers for technical support, you can use RobotStudio's P&G function. You can access this function by going to: RobotStudio File tab --> Share --> Pack and Go.

4 Navigating Palletizing PowerPac

4.1. Introduction

This chapter describes how to navigate in PzPP. Windows and other parts of the user interface are described in respect of their content and how they are accessed. The description of the main layout provides an overview of the menus, ribbon commands, and windows in PzPP.

4.2. General

Overview

Before you can start using PzPP, you must load a RobotStudio station.

In a station without system and robot, you are able to edit products, pallet patterns, create custom grippers and feeders, and edit libraries.

In a station with a system and at least one robot, you can continue to add tools and feeders to the robots, to configure flows and pick/place operation sets on feeders. The virtual controller (VC) associated with the robot must be loaded with Prepare for PickMaster option and PickMaster 5 sub-option.

Loading a Station

You can go through following steps to create a station and a system, and start using PzPP:

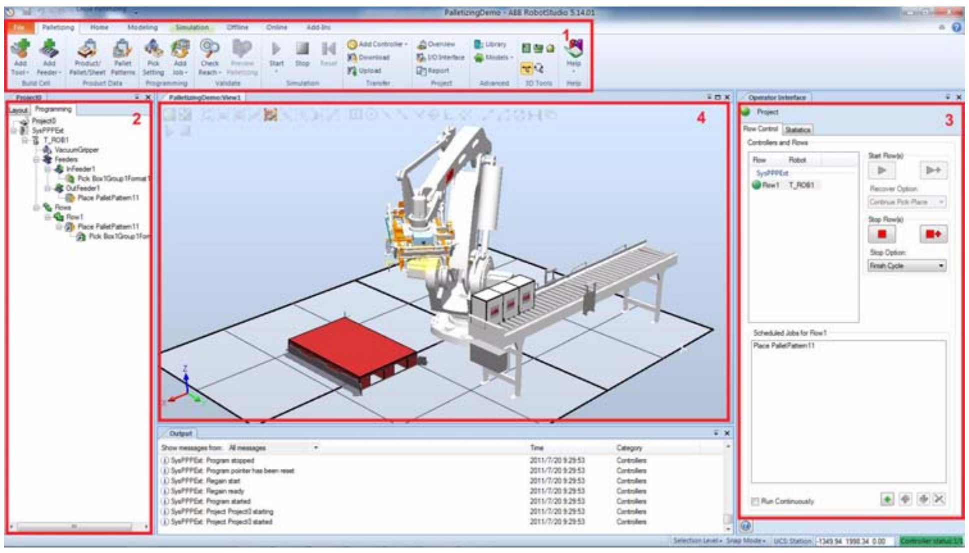
1. Open RobotStudio and create an empty station.
2. Select a palletizing robot and create a system from layout.
3. In the Systems option list, check Prepare for PickMaster – PickMaster 5. Finish system building and wait until system is ready.
4. On the Add-Ins tab in the ribbon, select Palletizing from the PowerPacs group.
5. A dedicated tab for Palletizing is added to the ribbon.
6. The Palletizing ribbon and tree structure browser opens.
7. Start from ribbon left to right: add tool, add feeder, add products, add patterns, add job, simulate, add real controllers and download.

NOTE!

For other ways to create a station and a system, refer to Operating manual - RobotStudio.

The User Interface

The panes and windows of the user interface, described in the following figure, help you create a well-structured palletizing program.



xx110000002

Continues on next page

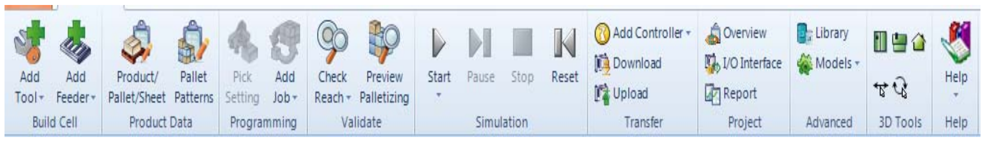
Continued

	Item	Description
1.	Palletizing ribbon tab	Contains the general functions for palletizing application. When creating a new project, the work flow is usually from left to right.See *Ribbon on page 45* for detailed description.
2.	Palletizing browser	Organizes the components of the station and project in a tree structure.See *Layout Browser on page 129* for detailed description.
3.	Tool window	Dialog boxes to edit certain project elements in browser and ribbon.
4.	Graphics window	The graphics window is coordinated with the selection in browser elements and edited object in the tool window, showing context related temporary graphic objects.

4.3 Ribbon

4.3.1. Overview

The PzPP ribbon contains the controls for building cell, creating pick and place programs, operating virtual controller, modifying PzPP data, modeling tools and feeders, and help information.



xx110000003

Elements on the Palletizing ribbon-tab

Group	Button	Description
Build Cell	Add Tool	Add a tool to robot for palletizing/depalletizing. See *Add tool on page 47* for detailed description.
	Add Feeder	Add feeders to the cell for palletizing/depalletizing. At least two feeders are needed for a robot. See *Add feeder on page 49*for detailed description.
Product Data	Product/Pallet/ Sheet	Create products, pallets or sheets to be palletized/depalletized. See *Product/Pallet/Sheet on page 51* for detailed description.
	Pallet Pattern	Create pattern related to the products to be palletized/depalletized. See*Pallet pattern on page 62* for detailed description.
Programming	Pick Setting	Define how robot tool picks the products, pallets or sheets. See *Pick Setting on page 68* for detailed description.
	Add Job	Open Job Wizard to build a new palletizing/depalltizing job. See *Add Job on page 73* for detailed description.
Validate	Check Reach	Check whether all of the picking robot targets and placing robot targets are reachable.
	Settings on Check Reachability	Check whether the specified picking robot targets and placing robot targets are reachable. See *Check Reach on page 78* for detailed description.
	Preview Palletizing	Preview all pick and place targets for all operation sets on feeders, by jumping tool or robot to the robot targets accordingly. See *Preview Palletizing on page 80* for detailed description.

Continues on next page

Continued

Group	Button	Description
Simulation	Quick Start	Transfer program to virtual controller, open operator interface and start the simulation with the default jobs in one click. See *Operator Interface on page 83* for detailed description.
	Operator Interface	Transfer program to virtual controller, open operator interface. Then start the simulation for the specified jobs manually. See *Operator Interface on page 83* for detailed description.
	Record Simulation as Viewer	Select this option to save the simulation as viewer file after the simulation is stopped. See *Record as Viewer File on page 87* for detailed description.
	Speed Mode	Set simulation speed and there are three modes for it, such as Full, Customized and Low. See *Speed Mode on page 88* for detailed description.
Transfer	Add Controller	Connect a controller with this station using RobotStudio Online functionality.
	Download	Download the program to connected controllers. See *Download on page 93* to Controllers for detailed description.
	Upload	Upload the program from the connected controller. See *Upload on page 95* from Controllers for detailed description.
Project	Overview	Show the overview information for all operation sets in the project. See *Project Overview on page 96* for detailed description.
	I/O Interface	Show the I/O information for this project. See *I/O Interface on page 97* for detailed description.
	Report	Generate a report for this project. See *Project Report on page 100* for detailed description.
Advanced	Library	Edit the default libraries for reuse. See *Library on page 101* for detailed description.
	Cell Models	Guide user to design customized tool and feeder.See *Create Tool on page 102* and *Create Feeder on page 124* for detailed description.
3D Tools	View angles	Adjust the view point for the station from top, left and front.
	Freehand move	Enable to adjust the position/orientation for the selected objects by manual dragging in 3D view.
Help	Help	Provide the user manual of PzPP
	About	Provide the general information about PzPP.

4.3.2. Add tool

Overview

xx110000004

It is to add tools for the robots in the station.

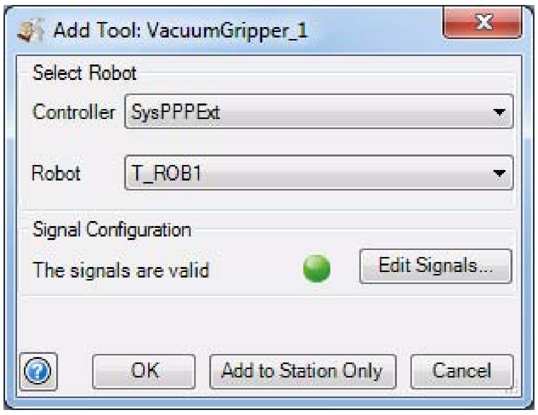
Click *Add Tool* to open a dialog box, or click the little arrow to see a preview of all tool libraries, to locate a tool model to add into station.

There are several default tool libraries provided in the group of ‘ABB Tools’. Other customized tool libraries stored under “My Documents/RobotStudio/Libraries/Palletizing Grippers/” are listed and can be selected as well.

NOTE!

One robot can only hold one tool for palletizing/depalletizing.

After the tool library is selected, a dialog box pops up.


Add Tool: VacuumGripper_1

Select Robot

Controller: SysPPPExt

Robot: T_ROB1

Signal Configuration

The signals are valid

xx110000005

Item	Description
Controller	Select a controller in which the robot to attach the tool belongs to.
Robot	Select a robot which will hold the tool.
Signal Configuration	Provide the match information about signals connection between the tool and the virtual controller. If the signal connection is not valid, the status light will be red, and you can check and modify with button “Edit Signals”. See *Edit Detailed Signals for Feeder on page 157* for detailed description.

Continues on next page

Use this procedure to edit parameters in the dialog box:

1. In the Controller drop-down list, select the currently used controller.
2. In the Robot drop-down list, select a robot.
3. In the Signal Configuration group, verify the signal connection status.
4. Click ‘Attach’ to add the selected tool to the robot. The robot will hold the tool and the signals will be connected between the tool and the virtual controller automatically. Or click ‘Add to Station Only’ to only add the selected tool into the station only. The robot will not hold the tool and there is no signal connection between the tool and the virtual controller.

After the tool is added to the robot, a new tool node will be added under the robot node in the programming browser, and also a tool node will be added in the layout browser.

4.3.3. Add feeder

Overview

xx110000006

It is to add feeders for the robots in the station.

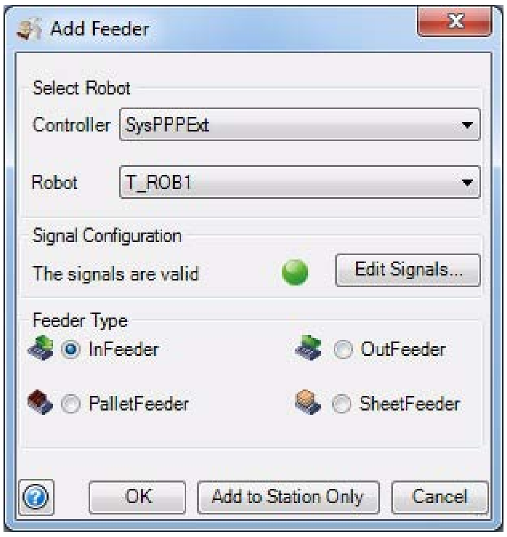
Click *Add Feeder* to open a dialog box, or click the little arrow to see a preview of all feeder libraries, to locate a feeder library to add into station.

There are several default feeder libraries provided in the group of ‘ABB Feeders’. Other customized feeder libraries stored under “My Documents/RobotStudio/Libraries/Palletizing Feeders/” are listed and can be selected as well.

NOTE!

There should be at least two feeders for a robot for palletizing/depalletizing.

After one feeder library is selected, a dialog box pops up.



xx110000007

Item	Description
Controller	Select a controller in which the robot to work with the feeder belongs to.
Robot	Select a robot in which the feeder is added.
Signal Configuration	Provide the match information about signals connection between the feeder and the virtual controller. If the signal connection is not valid, the status light will be red, and you can check and modify with button “Edit Signals”. See *Edit Detailed Signals for Feeder on page 157*. for detailed description.

Continues on next page

Use this procedure to edit parameters in the dialog box:

1. In the Controller drop-down list, select the currently used controller.
2. In the Robot drop-down list, select a robot.
3. In the Signal Configuration group, verify the signal connection status.
4. Click ‘OK’ to add the selected feeder to the robot. The signals will be connected between the feeder and the virtual controller automatically. Or click ‘Add to Station Only’ to add the selected feeder into the station only. There is no signal connection between the feeder and the virtual controller.

After the feeder is added, a new feeder node will be added under the “Feeders” node of the robot in the programming browser, and also a new feed node will be added in the layout browser.

4.3.4. Product/Pallet/Sheet

Overview

xx110000008

It is to input customer’s products.

Click *Product/Pallet/Sheet* to open a dialog box to create or import product/pallet/sheet to be used in this station. Product/pallet/sheet is also sometimes called item for short.

There are four item types: Box, Bag, Pallet, and Sheet.

For each type of item, there are two types for its collection:

?tGroup: a row by column grouping of several items, to describe how many items are handled by the robot at one pick. Usually for boxes and bags that comes with several rows.

?tStack: stacking of items by several layers. Usually for pallets and sheets, which are stacked together with several layers.

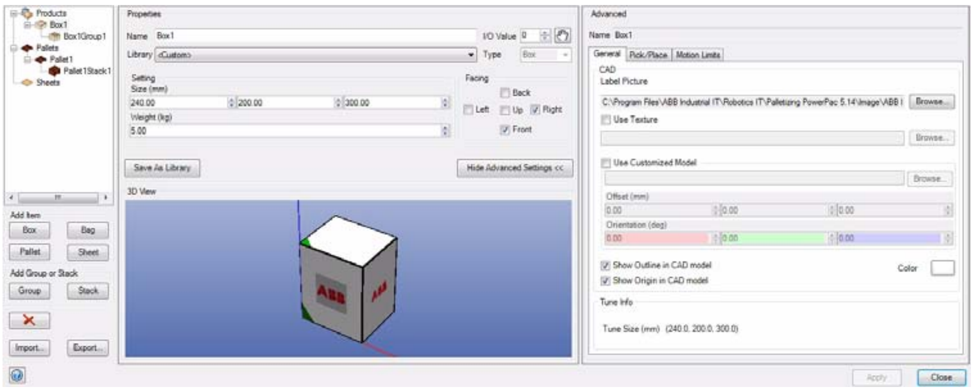
When a box or bag is created, a default group of 1 row and 1 column is added at the same time for it. When a pallet or sheet is created, a default stack of 1 layer is added at the same time for it.

NOTE!

In a project, one box or one bag should be created at least.

To use the item in later programming, such as pick setting, and operation sets, at least one group or stack should be added for each item. See *Pick Setting on page 68* for detailed description on how to program picking for an item.

The interface looks as following:



Name: Box1
Library: <Custom>
I/O Value: 0
Type: Box
Setting
Size (mm): 240.00 200.00 300.00
Weight (kg): 5.00
Facing: [ ] Back [ ] Left [ ] Up [✓ Right] [✓ Front]

Advanced
Name: Box1
Label Picture: C:\Program Files\ABB Industrial IT\Robotics IT\Palletizing PowerPac 5.14\Image\ABB I
[ ] Use Texture
[ ] Use Customized Model
Offset (mm): 0.00 0.00 0.00
Orientation (deg): 0.00 0.00 0.00
[✓ Show Outline in CAD model]
[✓ Show Origin in CAD model]
Color
Tune Info
Tune Size (mm) (240.0, 200.0, 300.0)

xx110000009

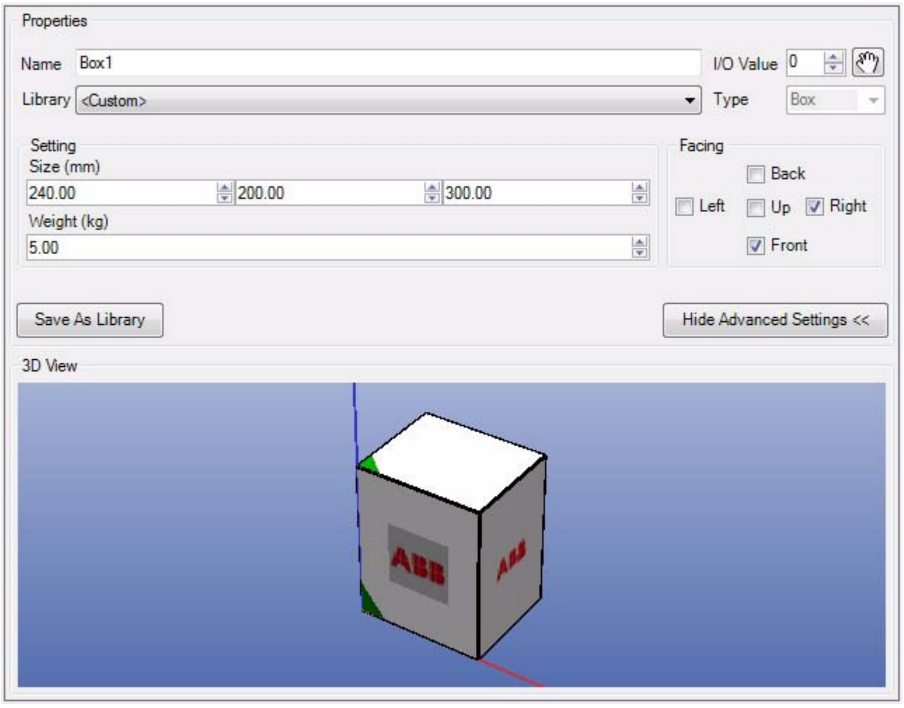
Continues on next page

Item	Description
Box	Create a new box as item with default values, and a default box group will be created simultaneously.
Bag	Create a new bag as item with default values, and a default bag group will be created simultaneously.
Pallet	Create a new pallet as item with default values, and a default pallet stack will be created simultaneously.
Sheet	Create a new sheet as item with default values, and a default sheet stack will be created simultaneously.
Group	Create a new group for the selected item.
Stack	Create a new stack for the selected item.
Delete	Delete the selected object (product, group or stack) in the item list.
Export	Export the existing items to an XML file for reusing.
Import	Import and create items from an XML file.

After one object is selected in the item list, its properties will be shown on the right UI.

Proceed with the Box Properties

This section describes how to edit the box properties.



xx110000010

To...	Do this
Rename a Box	In the Name text box, type a new name for the box. In default, a unique name can be generated automatically when a new box is created.
Set I/O value	In the I/O value number box, type a value to identify different products. Or click the button besides it, a new I/O value can be found automatically.

Continues on next page

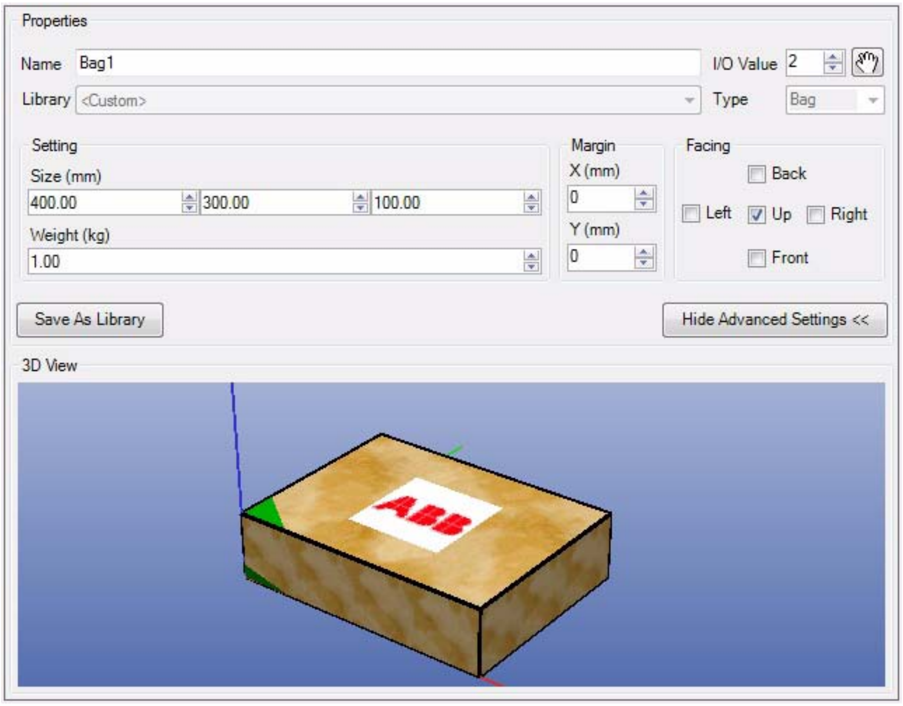
To...	Do this
Select Box from Library	In the Library drop-down combo box, select an existing box to read in its values.
View the Item Type	In the Type combo box, the item type is shown.Its type cannot be changed after the item is created.
Set Box Size	In the Size boxes, type the dimensions for x, y and z.
Set Box Weight	In the Weight number box, type the weight (in kg).
Set Box Facing	Select the check boxes in Facing group to define which sides of the box to attach a label. See below ‘Note’ for detailed description.
Save the Box	Click *Save As Library* button, current box will be saved into the library for reusing.
Edit Advanced Settings for Box	Click *Show Advanced Settings* button, the advanced settings of the box will be shown in the right window and can be modified manually.See *Advanced setting for Item on page 57*.
Show a Box	With the specified size and facing setting, the box can be shown in the 3D View.

NOTE!

The facing defines the sides of a product that are of specific importance. A facing side could be a label or a carton opening (See *Advanced setting for Item on page 57* to know how to change the label picture). If the product is to be used in a pallet pattern, the products can be placed in a way that will maximize labels on the outside or openings in a specific direction.

Proceed with the Bag Properties

This section describes how to edit the bag properties.


Properties
Name: Bag1
I/O Value: 2
Library: <Custom>
Type: Bag
Setting
Size (mm): 400.00 | 300.00 | 100.00
Weight (kg): 1.00
Margin
X (mm): 0
Y (mm): 0
Facing: Back [ ], Left [ ], Up [✓], Right [ ], Front [ ]

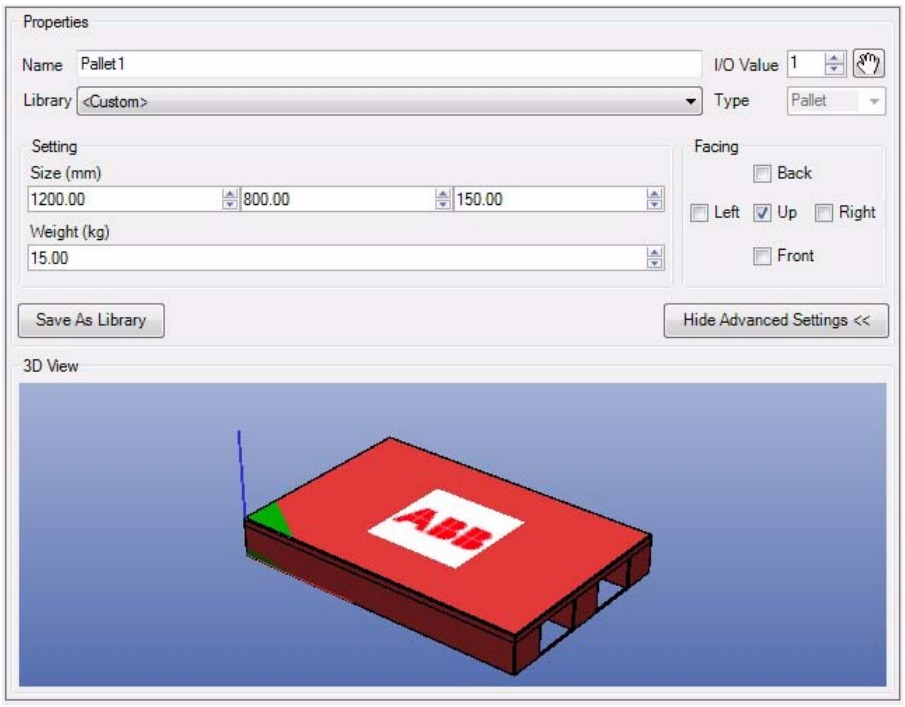
xx110000011

Continues on next page

To...	Do this
Rename a Bag	Same as Box
Set I/O value	Same as Box
Select Bag from Library	Same as Box
View the Item Type	Same as Box
Set Bag Size	Same as Box
Set Bag Weight	Same as Box
Set Bag Margin	In the Margin number boxes, type the margin distance which is allowed for overlapping between multiple bags in a pallet pattern.
Set Bag Facing	Same as Box
Save the Bag	Same as Box
Edit Advanced Settings for Bag	Same as Box
Show a Bag	Same as Box

Proceed with the Pallet Properties

This section describes how to edit the pallet properties.


Properties

Name: Pallet1
I/O Value: 1
Library: <Custom>
Type: Pallet

Setting
Size (mm): 1200.00 800.00 150.00
Weight (kg): 15.00

Facing: Back [ ] Left [ ] Up [✓] Right [ ] Front [ ]

xx110000012

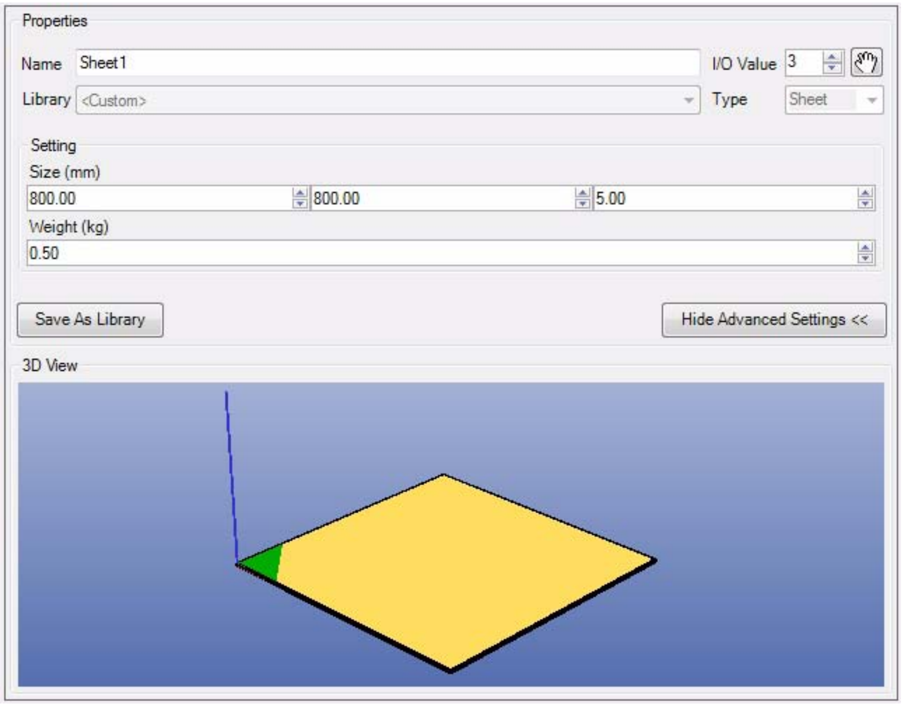
To...	Do this
Rename a Pallet	Same as Box
Set I/O value	Same as Box
Select Pallet from Library	Same as Box
View the Item Type	Same as Box
Set Pallet Size	Same as Box
Set Pallet Weight	Same as Box

Continues on next page

To...	Do this
Set Pallet Facing	Same as Box
Save the Pallet	Same as Box
Edit Advanced Settings for Pallet	Same as Box
Show a Pallet	Same as Box

Proceed with the Sheet Properties

This section describes how to edit the Sheet properties.


Properties
Name: Sheet1
I/O Value: 3
Library: <Custom>
Type: Sheet
Setting
Size (mm): 800.00 | 800.00 | 5.00
Weight (kg): 0.50

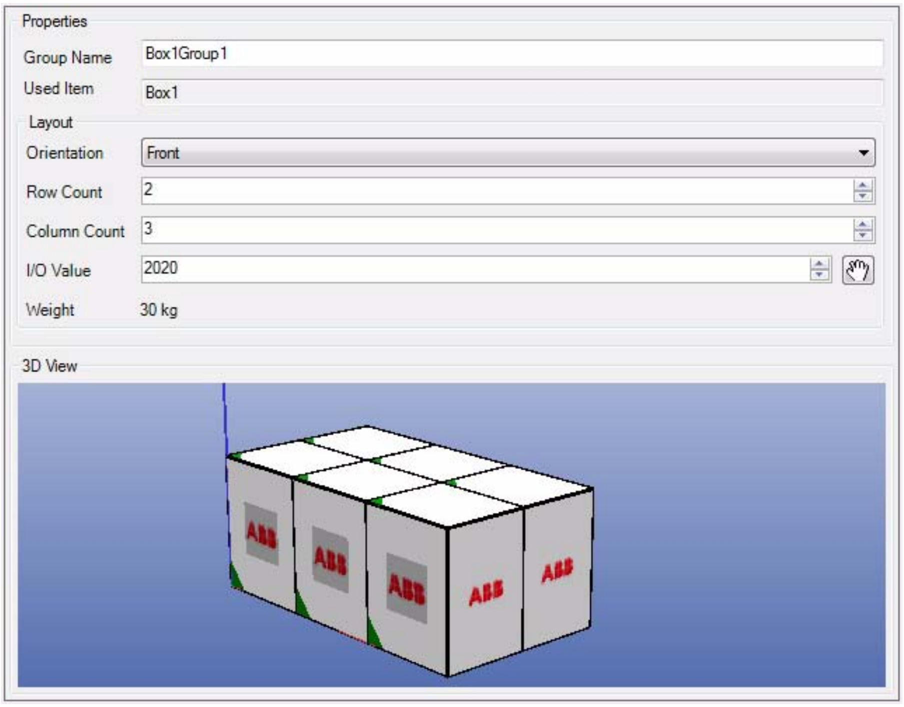
xx110000013

To...	Do this
Rename a Sheet	Same as Box
Set I/O value	Same as Box
Select Sheet from Library	Same as Box
View the Item Type	Same as Box
Set Sheet Size	Same as Box
Set Sheet Weight	Same as Box
Set Sheet Facing	Same as Box
Save the Sheet	Same as Box
Edit Advanced Settings for Sheet	Same as Box
Show a Sheet	Same as Box

Continues on next page

Proceed with the Group Properties

This section describes how to edit the Group properties.


Properties
Group Name: Box1Group1
Used Item: Box1
Layout
Orientation: Front
Row Count: 2
Column Count: 3
I/O Value: 2020
Weight: 30 kg
3D View

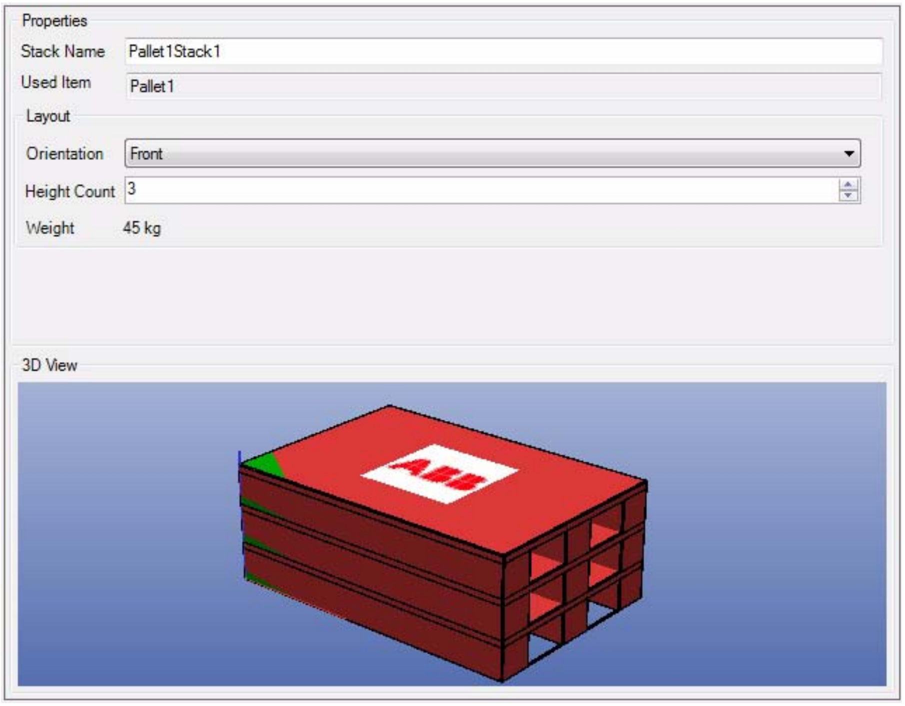
xx110000014

To...	Do this
Rename a Group	In the Group Name text box, type a new name for the group. In default, a unique name can be generated automatically when a new group is created.
View Used Item	In the Used Item combo box, this used item name will be shown.Its value cannot be changed after the group is created.
Set Group Orientation	In the Orientation combo box, select the group orientation to use. The selection defines the side of the group to place in negative y direction. A green marker shows the origin of each item.
Set Group Row	In the Row Count number box, type a value to define the rows of the group.
Set Group Column	In the Column Count number box, type a value to define the columns of the group.
Set I/O value	The I/O value is a combined value and is calculated from the orientation of the group and the number of rows and columns of occurrences of the item in the group. The I/O value represents the different ways a group can arrive on a conveyor. Also it is possible to type a value to identify different products in the I/O value number box.Click the button on the right, a new valid I/O value will be assigned.
Show a Group	With the specified row and column, the group can be shown in the 3D View.

Continues on next page

Proceed with the Stack Properties

This section describes how to edit the stack properties.



Stack Name: Pallet1Stack1
Used Item: Pallet1
Orientation: Front
Height Count: 3
Weight: 45 kg

xx110000015

To...	Do this
Rename a Stack	In the Stack Name text box, type a new name for the stack. In default, a unique name can be generated automatically when a new stack is created.
View Used Item	In the Used Item combo box, this used item name will be shown.Its value cannot be changed after the stack is created.
Set Stack Orientation	In the Orientation combo box, select the stack orientation to use. The selection defines the side of the stack to place in negative y direction. A green marker shows the origin of each item.
Set Stack Height	In the Height Count number box, type a value to define the layers of the stack.
Show a Stack	With the specified row and column, the group can be shown in the 3D View.

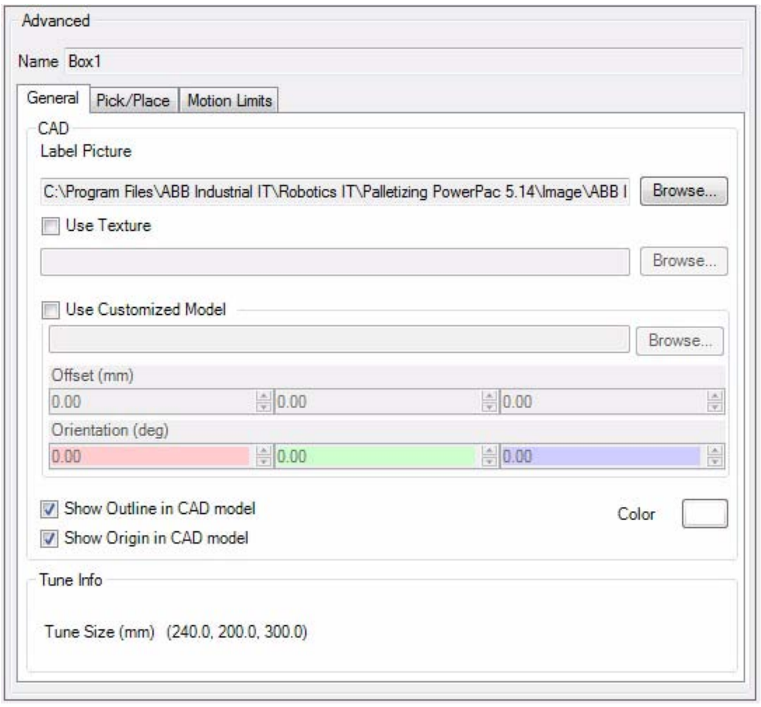
Advanced setting for Item

After one item is created, its advanced settings can be edited as well. There are 3 pages for its settings: General, Pick/Place and Motion Limits.

Continues on next page

Proceed with the General

In this section the general settings about the CAD of the item can be adjusted.


Advanced
Name Box1
General | Pick/Place | Motion Limits
CAD
Label Picture
C:\Program Files\ABB Industrial IT\Robotics IT\Palletizing PowerPac 5.14\Image\ABB I
[ ] Use Texture
[ ] Use Customized Model
Offset (mm) 0.00 0.00 0.00
Orientation (deg) 0.00 0.00 0.00
[✓] Show Outline in CAD model
[✓] Show Origin in CAD model
Color
Tune Info
Tune Size (mm) (240.0, 200.0, 300.0)

xx110000016

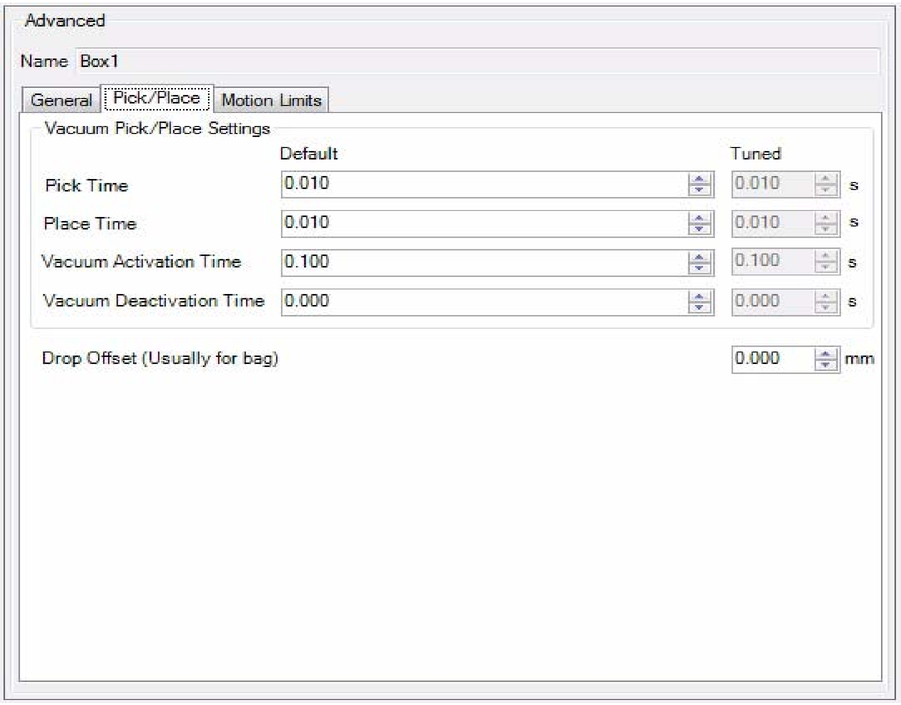
To...	Do this
Set Label Picture	Click the *Browse* button to select a custom picture to use as product label.
Set Texture	Select the Use Texture check box, and then click the *Browse* button to select a picture to use as texture.
Set Customized Model	Select the Use Customized Model check box, and then click the *Browse* button to select a customized CAD model for the item, to replace the default box-like CAD model.Note that if customized model is used, changing size of the item will not affect the model itself.
Show Outline	Select the Show Outline check box, the outline of the item CAD can be shown as thin lines along the product edges.
Show Origin	Select the Show Origin check box, the origin of the item CAD can be shown as green markers on the top and bottom surfaces.
Set Color	Click the *Color* button, the color of the item CAD can be selected.Note that color will be hidden if texture is used.

Continues on next page

Proceed with the Pick/Place

In this section the timing of tool events can be set, in order to pick and place the item as efficient as possible. You can define the pick time, that is, the time the robot is standing still in the pick position when picking up the item. Similarly the place time can also be defined, that is, the time the robot is standing still in the place position when placing the item.

The drop offset for the item in a pattern operation set can be set as well. This is usually used for bag when being released with an offset above the already finished stack. If a pattern operation set is created using this item, the value will be used as each layer's drop offset. See *Offsets for each layer on page 170* for detailed information.



Name Box1

	Default	Tuned	
Pick Time	0.010	0.010	s
Place Time	0.010	0.010	s
Vacuum Activation Time	0.100	0.100	s
Vacuum Deactivation Time	0.000	0.000	s

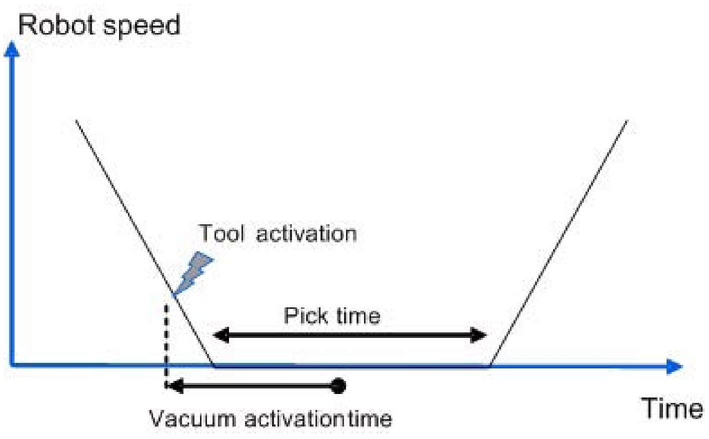
Drop Offset (Usually for bag) 0.000 mm

xx110000017

To...	Do this
Set Time Values for Vacuum Activation and Deactivation.	In the Vacuum Activation Time and Vacuum Deactivation Time number boxes, type the values.
Adjust Time that Robot Stays at the Robot Target Position When Picking or Placing an Item.	In the Pick Time and Place Time number boxes, type the time in seconds.
Set Drop Offset	In the Drop Offset number box, type the offset (in mm).It is useful for releasing bag from a certain height

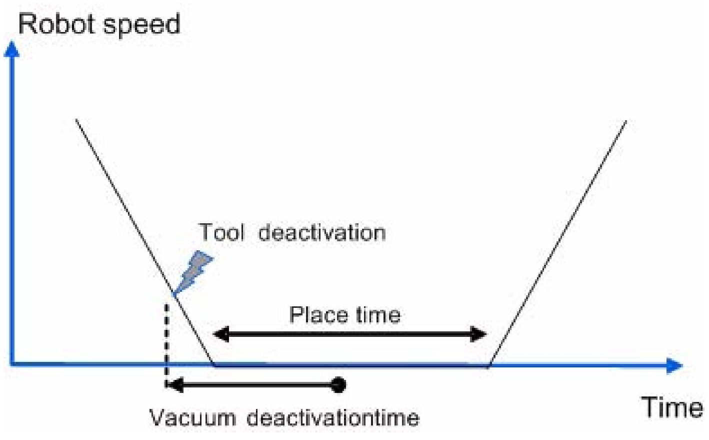
Continues on next page

You can adjust the location, where the tool zone gets activated when picking the item. The time when the zone should activate, Vacuum Activation Time, is specified in seconds before reaching the pick position. If a negative time is specified, activation will take place after reaching the pick position. If the time is set to zero, activation will occur after half the pick time has passed.



xx110000018

You can also adjust the location, where the tool zone gets deactivated when placing the item. The time when the zone should deactivate, Vacuum deactivation time, is specified in seconds before reaching the place position. If a negative time is specified, deactivation will take place after reaching the place position. If the time is set to zero, deactivation will occur after half the place time has passed.

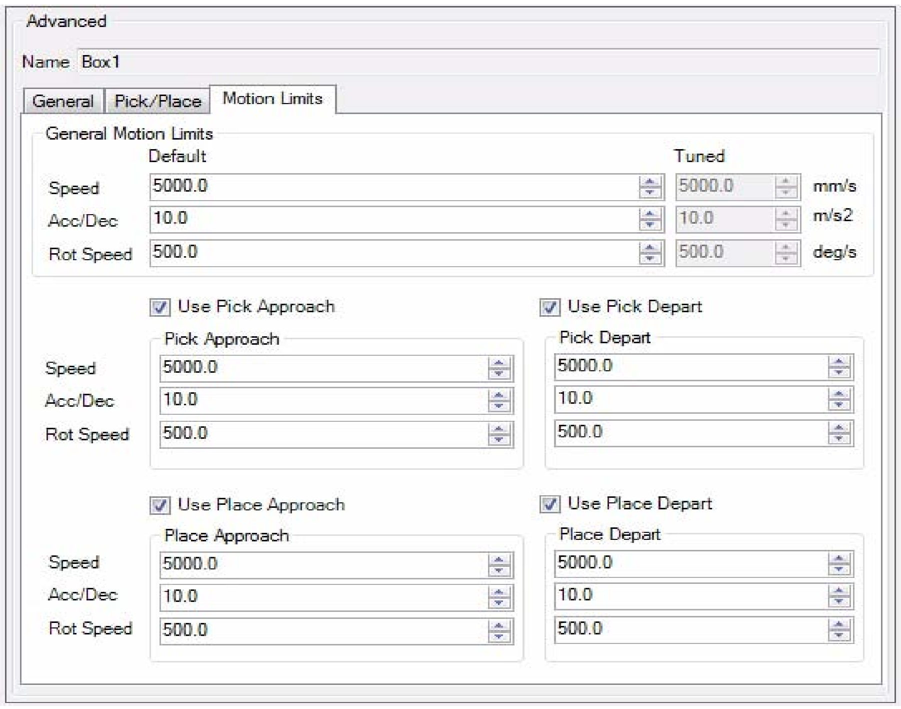


xx110000019

Continues on next page

Proceed with the Motion Limits

In this section the default speed and acceleration limits for the item can be defined. Also the tuned speed and acceleration of the item can be viewed. To edit separate motion configurations for the actions (Pick Approach, Pick Depart, Place Approach and Place Depart), select the Use * check box of the action to edit. If the Use * check box is cleared, the default settings will be used for this action.



Advanced

Name Box1

General | Pick/Place | Motion Limits

General Motion Limits

	Default	Tuned	
Speed	5000.0	5000.0	mm/s
Acc/Dec	10.0	10.0	m/s2
Rot Speed	500.0	500.0	deg/s

[✓] Use Pick Approach — [✓] Use Pick Depart

	Pick Approach	Pick Depart
Speed	5000.0	5000.0
Acc/Dec	10.0	10.0
Rot Speed	500.0	500.0

[✓] Use Place Approach — [✓] Use Place Depart

	Place Approach	Place Depart
Speed	5000.0	5000.0
Acc/Dec	10.0	10.0
Rot Speed	500.0	500.0

xx110000020

To...	Do this
Set Maximum Speed	In the Speed number box, type the maximum allowed speed.
Set Acceleration/Decelera-tion	In the Acc/Dec number box, type the acceleration value (in mm/s).The same value will be used for both acceleration and decel-eration.
Set Maximum Rotation Speed	In the Rot Speed number box, type the maximum allowed rotation speed (in deg/s).

NOTE!

There is NO need to set approach or depart speed because these will be limited by the acceleration and deceleration limits.

4.3.5. Pallet pattern

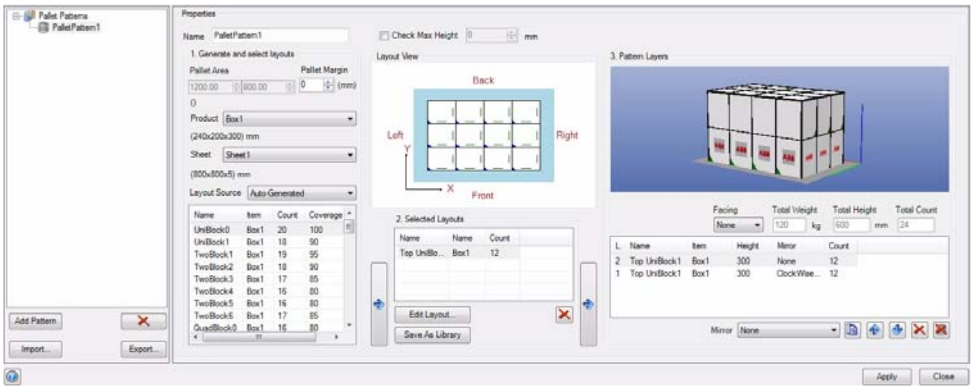
4.3.5.1. Introduction

Pallet Patterns

xx110000021

It is to input customer's pallet patterns.

Click *Pallet Pattern* to open a dialog box to create or import pallet pattern to be used in this station.The item(s) that are defined before will be used in the pattern. In a project, one pallet pattern should be created at least.



Name: PalletPattern1

1. Generate and select layouts

Pallet Area: 1200.00 800.00; Pallet Margin: 0 (mm)

0

Product: Box1

(240x200x300) mm

Sheet: Sheet1

(800x800x5) mm

Layout Source: Auto-Generated

Name	Item	Count	Coverage
UniBlock0	Box1	20	100
UniBlock1	Box1	18	90
TwoBlock1	Box1	19	95
TwoBlock2	Box1	18	90
TwoBlock3	Box1	17	85
TwoBlock4	Box1	16	80
TwoBlock5	Box1	16	80
TwoBlock6	Box1	17	85
QuadBlock0	Box1	16	80

Check Max Height: 0 mm

2. Selected Layouts

Name	Name	Count
Top UniBlo...	Box1	12

Facing: None; Total Weight: 120 kg; Total Height: 600 mm; Total Count: 24

L.	Name	Item	Height	Mirror	Count
2	Top UniBlock1	Box1	300	None	12
1	Top UniBlock1	Box1	300	ClockWise...	12

Mirror: None

xx110000022

Item	Description
Add Pattern	Create a new pallet pattern with items created before.See *Proceed with the Pallet Pattern Layout on page 63* for detailed description.
Delete	Delete the selected pallet pattern in the pattern list.
Export	Export the existing pallet patterns into XML file.
Import	Import and create pallet patterns from XML file. The used items in the XML file will also be created in station.

Continues on next page

Continued

After one pallet pattern is selected in the pattern list, its corresponding properties will be shown in the right.

Item	Description
1. Generate and Select Layouts	Based on the selected pallet, product, sheet and inputted pallet margin, the possible layouts will be calculated. Or you can also select layouts saved in the library.You can choose a layout and click the right arrow button to select into Selected Layouts.
2. Selected Layouts	The selected layouts can be modified, and/or used directly to create layers.The specified layout can also be saved to reuse.You can choose a selected layout and click the right arrow button to add into pattern layers.
3. Pattern Layers	Shows the whole layers in a pallet pattern. For each layer you can also see the layout name, and adjust the mirror type.
Layout View	Shows the 2D view of any selected layout in the above 3 lists of layouts.

Proceed with the Pallet Pattern Layout

This section describes how to proceed with the pallet pattern layout.A pallet pattern is built as a stack of item layouts. Each layout may only contain one type of item.

To...	Do this
Select Pallet Area	In the selection box ‘Pallet Area’, select the pallet to use. If there is no pallet defined before, a default pallet area (1200 mm* 800mm) will be used for calculating the layout, which value you can also modify.The layout will be generated based on the selected pallet or input pallet size, and the margin.
Set Pallet Margin	In the Pallet Margin text box, type the minimum allowed margin (in mm). This will limit the maximum layout size by including a safety margin to the size of the palletizing area.
Select Product	In the combo box ‘Product’, select the product to use, i.e., a box or a bag.
Select Sheet	In the combo box ‘Sheet’, select the sheet to use. That is optional for the pallet pattern.

NOTE!

You must always select a pallet, even if it will not be included in the pallet pattern. The pallet defines the maximum size of each layout. The pallet can NOT be changed once a layout based on the pallet is selected but sheet and product can be changed to build a pallet pattern with different items.

When the appropriate items are selected, the Layout Source list will be propagated with the generated layouts. The layouts are generated for the selected product using different algorithms. Information, such as item count and coverage, is shown for each layout. You can also use layouts that are saved in the library.After you have selected a new sheet, product, or

Continues on next page

given a new margin, new layouts are generated based on the new information. The new layouts are listed in the Layout Source list. All layouts in the Selected Layouts list are not affected and still use the old product and margin.

To...	**Do this**
Access Library Layouts	In the combo box at Layout Source, select From Library. This will show all layouts from the library converted to use the selected items.
Select Layouts to Use in the Pallet Pattern	In the Layout Source list, select the layouts of interest and either use the right arrow button to the Selected Layouts list.
Edit a Layout	In the Selected Layouts list, select the layout and click *Edit Layout* button. This opens up the Layout Editor. See *Layout Editor on page 65* for detailed description.
Export a Layout to the Library	In the Selected Layouts list, select the layout and click ‘Save As Library’
Show a Layout	In either the Layout Source or the Selected Layouts list, select the layout and it will be shown in the Layout View
Remove Layout from Selected List	In the Selected Layouts list, select the layout to remove and click the delete button.

Proceed with the Pallet Pattern Layers

This section describes how to proceed with the pallet pattern layer.When the layouts to use in the pallet pattern are defined, continue with setting the order and number of layers. The resulting pallet pattern is always shown in the Pattern Layers together with the stack details.

To...	**Do this**
Add Layers	In the Selected Layouts list, select the layout you want to use and click the right arrow button to the Pattern Layers list.
Remove Layers	In the Pattern Layers list, select the layer and click the delete button. Also click the *Remove All* button to remove every layer in the Pattern Layers list.
Reorder Layers	In the Pattern Layers list, select the layer and click the *Up* or *Down* button to reorder the layers in the list.
Show a Layer	In the Pattern Layers list, select the layer and it will be shown in the Layout View
Show Pallet Pattern	All of the layers in the Pattern Layers list will be shown in the Pattern Layers View.

4.3.5.2. Layout Editor

Overview

In the Layout Editor you create new or modify existing layouts. The purpose of modifying layouts is to make them fit the operator's specific requirements. A layout can be saved to the Library and/or a pallet pattern configuration for a project.

Start the Layout Editor

Depending on which layout to edit, you can start the Layout Editor in two ways:

?tFrom the Library, if you want to edit a layout that is included in the library, or

?tFrom a pallet pattern configuration, if you want to edit a layout that is included in a pallet pattern.

Start from the Library

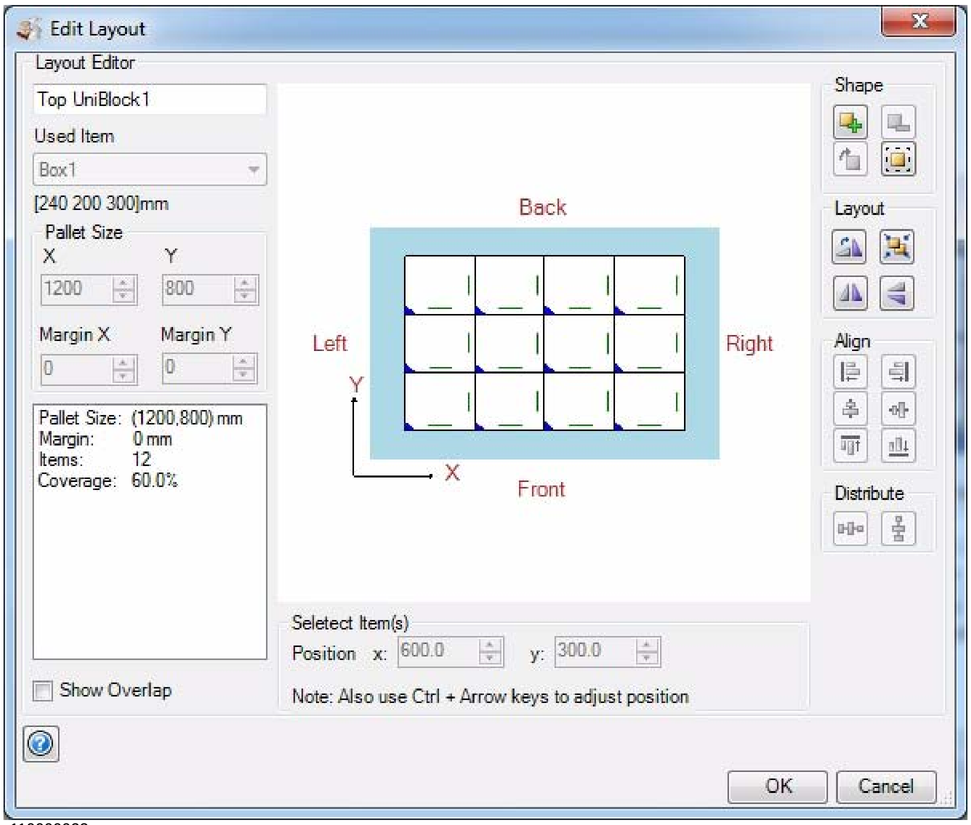
	Action
1.	In the library tree view, select a layout item to edit. See *Library on page 101* for more detailed information.
2.	Click Edit.
3.	The Layout Editor window appears and you can edit the layout.

Start from a pallet pattern configuration

	Action
1.	Select the pallet pattern that includes the layout to edit.
2.	In the Selected layouts list, select the layout to edit.
3.	Click Edit Layout
4.	The Layout Editor window appears and you can edit the layout.

Continues on next page

Illustration, Layout Editor



xx110000023

Item	Description
Layout	Shows the name of the layout and the size of the area where to palletize. The Layout area is mostly the area of a pallet. The Layout margin is the free distance from the edge of the palletizing area to the actual layout. The values for Layout area and Layout margin are only editable when there are no items in the layout.
Item	The drop-down combo box shows the selected item. If you add a new item, it will be added to the layout. If there are no items in the layout, you select which item to add. When you edit a library layout you can change the selected item to any of the available items in the library. When you edit a layout in a pallet pattern, you cannot change the item.
Display	Shows the layout of a layer. Here you modify the layout by using a drag-and-drop operation or the buttons on the right side. The Show Overlap check box provides an option to show the overlapping boxes in red.
Selection	Shows the position and orientation of the selected item. You can adjust the position by editing the values in the text boxes, or by using CTRL + arrow keys.
Layout information	Shows the number of items in the layout and its coverage ratio. If there are any overlapping shapes, this will be noted here as well.

Continues on next page

Continued

Proceed with the item

To...	Do this
Select multiple items	Press and hold the CTRL key while you click the items to select. Or use Press Left mouse key + Drag to multi select items
Select all items	Click Select All, or perform a select all operation (CTRL+A).
Add a new item	Click Add.
Copy an item	Click the item to copy and perform a copy-and-paste operation (CTRL+C for copy, CTRL+V for paste).
Delete the selected item/ items	Click Delete, or press the Delete button on the keyboard.
Rotate the selected item/ items	Click Rotate.

Proceed with the layout

To...	Do this
Mirror the layout in x direction	Click Flip Horizontal.
Mirror the layout in y direction	Click Flip Vertical.
Rotate the entire layout 180°	Click Rotate.
Center the layout	Click Center.

Align and distribute

The selected items can be aligned relative to each other by their edges. When you align items, the marked item always remains stationary. For example, clicking Align Left aligns the left edges of all selected objects with the left edge of the marked item. The selected items can also be distributed relative to the marked item. When items are distributed, all selected items are moved adjacent to the marked item in a horizontal or vertical direction. The marked item always remains stationary. Distribution of items is normally followed by an alignment operation.

TIP!

All the commands are also accessible from the menu that appears when you select an item and right-click.

Related information

Library on page 101

Pallet pattern on page 62

Terms and concepts on page 18

4.3.6. Pick Setting

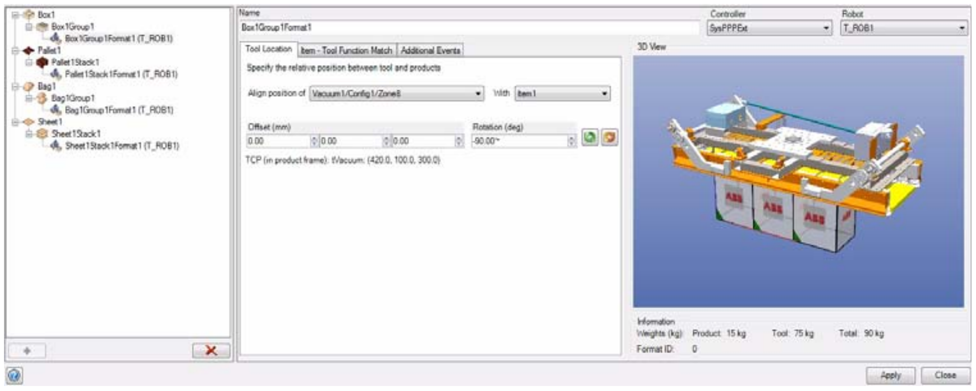
Overview

xx110000024

Click *Pick Setting* to open a dialog box to define how an item group or a stack is held by a robot tool. The item and tool orientations must be configured, as well as the tool functions (and zones if vacuum function is used) to activate when gripping the items. Furthermore, tool I/O signal events can be defined to control specific tool functionality.

NOTE!

It is possible to define more than one pick settings for a same item group or stack.


Box1
Box1Group1
Box1Group1Format1 (T_ROB1)
Pallet1
Pallet1Stack1
Pallet1Stack1Format1 (T_ROB1)
Bag1
Bag1Group1
Bag1Group1Format1 (T_ROB1)
Sheet1
Sheet1Stack1
Sheet1Stack1Format1 (T_ROB1)

Name: Box1Group1Format1
Controller: SysPPPExt
Robot: T_ROB1

Specify the relative position between tool and products

Align position of: Vacuum1/Config1/Zone8 With: Item1

Offset (mm): 0.00 0.00 0.00
Rotation (deg): -90.00°

TCP (in product frame): tVacuum: (420.0, 100.0, 300.0)

Information
Weights (kg): Product: 15 kg Tool: 75 kg Total: 90 kg
Format ID: 0

xx110000025

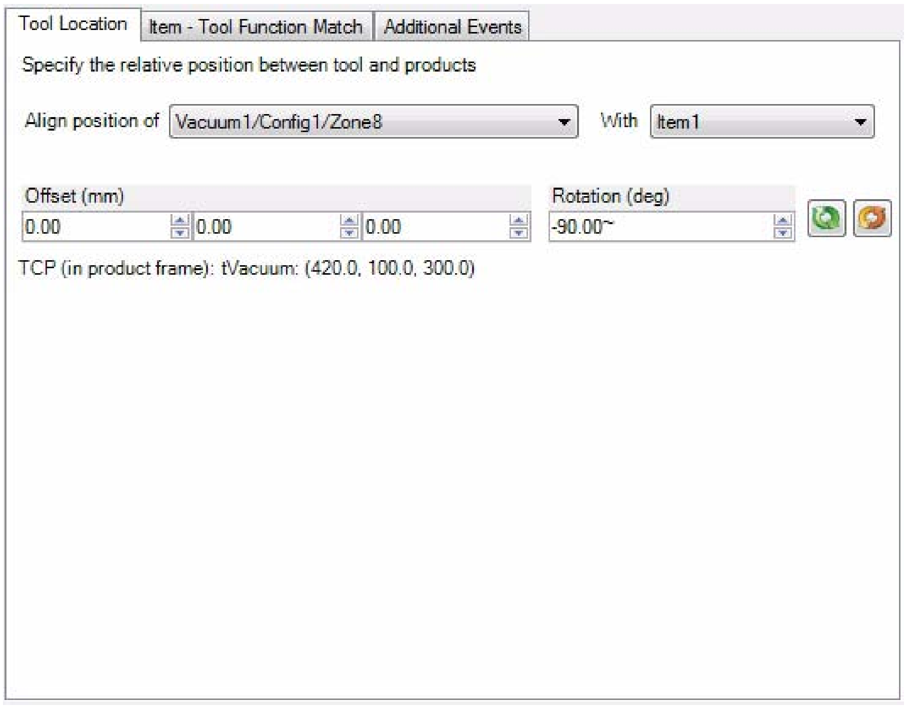
Item	Description
Add	Create a new pick setting for an item group or a stack.
Delete	Delete the selected pick setting.
Show Pick Setting	In the Pick Setting list, select the pick setting and the tool with the items will be shown in the 3D View. Weight information is shown, including product weight, and tool weight. If the product weight is more than the max payload of the tool's used tool data, a warning icon will be shown. Additionally, format ID is shown.

After one object is selected in the pick setting list, its preview and corresponding properties will be shown in the right.

Continues on next page

Proceed with the Tool Location

This section describes how to adjust the tool location relative to the group or the stack.



Specify the relative position between tool and products

Align position of: Vacuum1/Config1/Zone8 With: Item1

Offset (mm): 0.00 0.00 0.00 Rotation (deg): -90.00~

TCP (in product frame): tVacuum: (420.0, 100.0, 300.0)

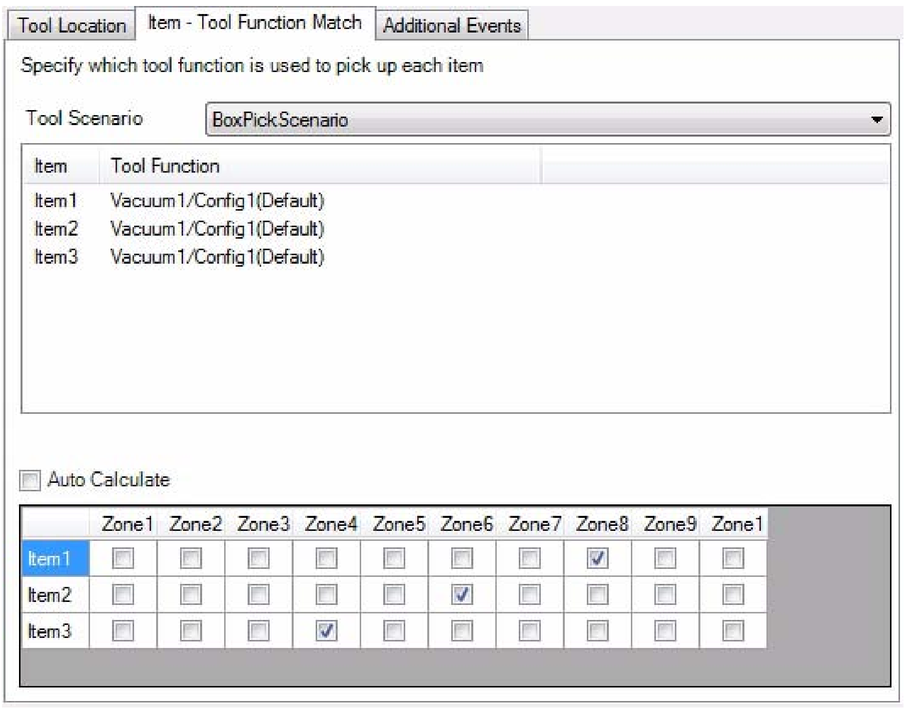
xx110000026

To...	Do this
Align Position between tool functions (and/or zones) with products	Select the tool functions (and zones if the tool contains vacuum function) from the left selection list, and select the product in the right selection list. The tool will then be put to a location where the tool function position is at the center top of the product. Note that this is only used to change tool location, and does not necessarily mean that the selected tool function will be used to pick the product. To set which tool function to use for picking each product, see *Proceed with the Item and Tool Function Match on page 70*.
Adjust Offset and Rotation	Input the translation offset and rotation angles (or click the rotation buttons) to adjust detailed offsets. The offset is relative to the aligned tool function and the selected product.
View TCP location	The TCP name and location is shown. The location is relative to the item group or stack's origin.

Continues on next page

Proceed with the Item and Tool Function Match

This section describes how to adjust the match relationship between item and tool function, i.e., which tool function is used to pick or place each item.



Specify which tool function is used to pick up each item

Tool Scenario: BoxPickScenario

Item	Tool Function
Item1	Vacuum1/Config1(Default)
Item2	Vacuum1/Config1(Default)
Item3	Vacuum1/Config1(Default)

☐ Auto Calculate

	Zone1	Zone2	Zone3	Zone4	Zone5	Zone6	Zone7	Zone8	Zone9	Zone1
Item1	☐	☐	☐	☐	☐	☐	☐	✓	☐	☐
Item2	☐	☐	☐	☐	☐	✓	☐	☐	☐	☐
Item3	☐	☐	☐	✓	☐	☐	☐	☐	☐	☐

xx110000027

To...	Do this
Select Tool Scenario	In the Tool Scenario list, select the tool scenario you want to use. The tool scenario defines a group of tool functions that will be used in the pick setting, and tool functions not included in a tool scenario will not be activated. See *Tool Events and Scenarios on page 120*for detailed information.
Adjust Relationship between Tool Function and Items	For each item, select a tool function to match with this item. If vacuum is selected for one item, it should also be used for other items.
Adjust Relationship between Tool Zone and Items (if vacuum is used)	If vacuum is used, you need to check the match between zones and items, i.e., using which zones to pick/place which items.Select “Auto Calculate” will let PowerPac calculate default item-zone matches.

Continues on next page

Continued

When simulation is running, robot is at a pick position, and tool open signals are activated, the picking sensor of the tool functions will also be activated and any items that intersect with the sensor area will be picked up. Thus to make simulation of picking work, the picking location of the tool, relative to the items, should make the picking sensor areas of the tool function intersect with the matched item.

See *Create Tool on page 102* for detailed information on picking sensor of different type of tool functions.

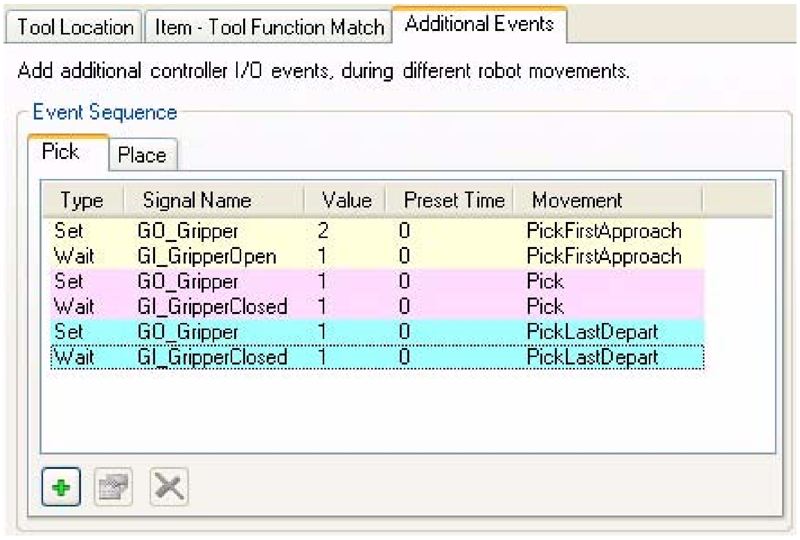
NOTE!

The position of the tool, as well as the match between tool function and item, are important for simulation of picking to work.

Proceed with the Tool Additional Events

This section describes how to edit the additional events for the tool.

For some pick settings, the items cannot be picked or placed by only using the defined tool functions and/or zones - for example, when picking a pallet using a specific I/O driven tool that cannot be modeled by standard tool functions. In such cases it is possible to set digital or group output signals and wait for digital input signals at various positions when picking or placing the products. These I/O events can be set for each pick or place operation at first approach action, last approach action, target action and first depart action, and last depart action. You must also define the value to set for output signals and the value to check for digital input signals. See an example in the following illustration.



Add additional controller I/O events, during different robot movements.

Type	Signal Name	Value	Preset Time	Movement
Set	GO_Gripper	2	0	PickFirstApproach
Wait	GI_GripperOpen	1	0	PickFirstApproach
Set	GO_Gripper	1	0	Pick
Wait	GI_GripperClosed	1	0	Pick
Set	GO_Gripper	1	0	PickLastDepart
Wait	GI_GripperClosed	1	0	PickLastDepart

xx110000028

Type	Type of the event: ?tSet: set an output signal to a value at a specific movement. ?tWait: check the value of an input signal that indicates the actual status of the tool. In this example used for checking that the desired action indicated by the group output signal has been achieved.
Signal Name	The signal that is used in the event

Continues on next page

Continued

Preset Time	The time in advance for the out signal to set to a value (only used for Set output type)
Movement	Describes when event occurs, i.e., first approach action, last approach action, target action and first depart action, or last depart action.

4.3.7. Add Job

Overview

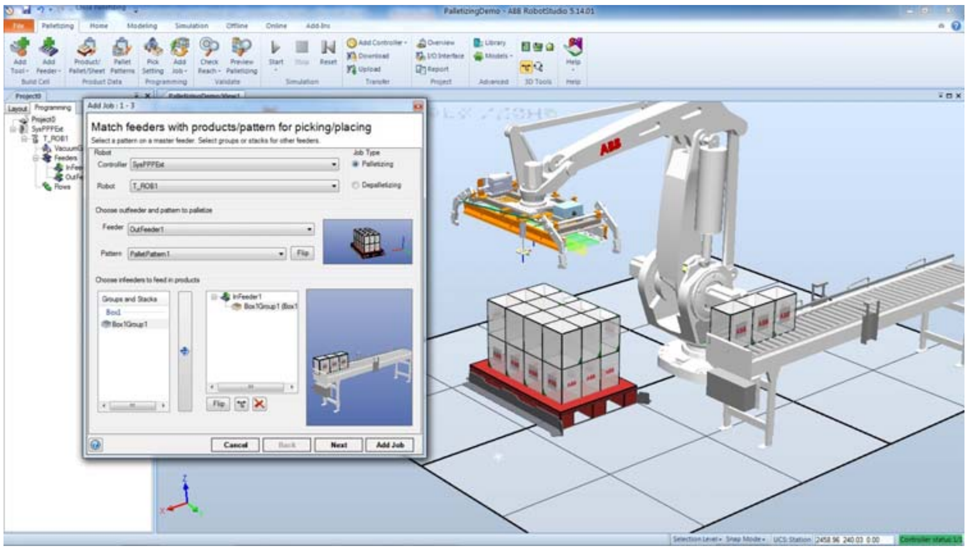
xx110000029

After the cell is built, the product data is created and the pick setting is ready, a new job for palletizing/depalletizing can be created with a wizard.

Click *Add Job* to open a wizard to create a new job for palletizing/depalletizing in three steps.

	Page	Action
1	Match Feeders with Products/ Pattern for Picking/Placing	Select a controller, robot, job type, specify a pallet pattern on a master feeder and select necessary groups/stacks on the other feeders.
2	Configure Operation Set for Group/ Stack	Configure the operation set for selected groups/ stacks in detail.
3	Configure Operation Set for Pallet Pattern	Configure the operation set for selected pallet pattern in detail.

Distribute roles for feeders



Match feeders with products/pattern for picking/placing

Select a pattern on a master feeder. Select groups or stacks for other feeders.

Robot
Controller: SysPPPExt
Robot: T_ROB1

Job Type: [✓ Palletizing] Depalletizing

Choose outfeeder and pattern to palletize
Feeder: OutFeeder1
Pattern: PalletPattern1

Choose infeeders to feed in products
Groups and Stacks: Box1, Box1Group1
InFeeder1: Box1Group1 (Box1

xx110000030

To...	Do this
Set Job Type	There are two types for the job, Palletizing and Depalletizing. It should be defined for the job at first. In the Job Type group, select the radio box to define the job type.
Select Controller and Robot	In the Robot group, select a controller and a robot in which the job is added.

Continues on next page

To...	Do this
Select Feeder as Master	Master feeder is usually the feeder where the palletizing/depalletizing jobs will be started. See *Flow on page 184* for detailed information. In the Feeder selection box, select one feeder as the master. Note that feeders that are not valid to be a master (for example, those being slave feeders for existing flows) are not listed in the selection box. Once the master feeder is selected, the other feeders of the robot valid as slave will be listed in the tree view. If this feeder is not used as master by any existing flow, when the job creation is finished, a new flow will be created as well, including the newly generated job. If this feeder is a master of an existing flow, when the job is created, the existing flow will include the newly created job.
Select Pallet Pattern for the Job	In the Pattern selection box, select one pallet pattern. This pattern will be implemented in this job.In the main 3D view, the selected pallet pattern will be shown on the master feeder.Once the pallet pattern is selected, the necessary groups or stacks to build the pallet pattern will be listed in the list view (i.e. Groups and Stacks list).
Adjust Orientation for Selected Pallet Pattern	Click Flip button, the orientation of selected pallet pattern will be flipped 90 degrees on the master feeder.
Select Feeder to Feed in Products	Select one candidate feeder in the right list, select one group or stack in the left list, click Right Arrow button, the selected group or stack will be added under the selected candidate feeder. It means that this feeder will feed in (or out) products in this palletizing (or de-palletizing) job. Groups or stacks provided by existing operation sets on the feeder will also be listed.
Remove Feeder from Feeding in Products	In the right list, select one group or stack under a candidate feeder, click Delete button, the selected group or stack will be removed from the candidate feeder. It means that this selected group on this feeder will not play a role in this job.
Adjust Orientation for Selected Group or Stack	In the right list, select one group or stack under a candidate feeder, click Flip button, the orientation of selected group or stack will be flipped 90 degrees on this feeder.

Continues on next page

Click *Next button* to continue to the next page.

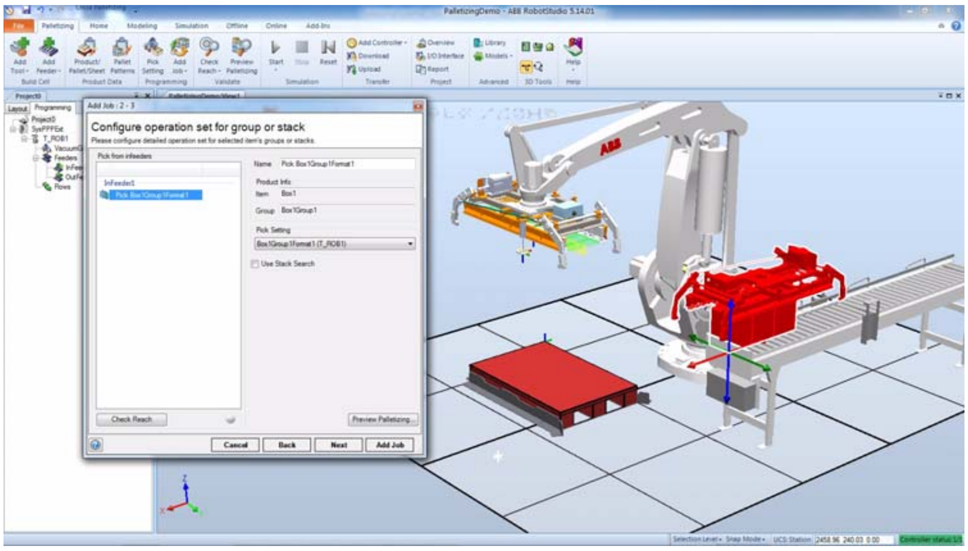
TIP!

In the main 3D View, the products can be moved directly to adjust their position and orientation.

Configure operation set for group or stack

After the group or stack is selected in the previous page, this page will configure the operation set for it. In general, if this job type is palletizing, the action for the operation set in this page is Picking, otherwise the action is Placing.

Each group or stack on each feeder will have its own operation set. The operation set will define how to pick or place the group or stack based on the pick setting related to the corresponding group or stack.


Add Job : 2 - 3

Configure operation set for group or stack

Please configure detailed operation set for selected item's groups or stacks.

Pick from infeeders

InFeeder1

Pick Box1Group1Format1

Name: Pick Box1Group1Format1

Product Info

Item: Box1

Group: Box1Group1

Pick Setting: Box1Group1Format1 (T_ROB1)

Use Stack Search

xx110000031

To...	Do this
Rename the Operation Set	In the Name text box, type a new name for the operation set. In default, a unique name can be generated automatically when a new operation set is created.
Select Pick Setting	Each group or stack should have at least one pick setting defined before. Based on the selected group or stack, one appropriate pick setting can be selected in the Pick Setting selection box for this operation set.
Set Stack Search	In some cases before picking the items (usually for pallet or sheet), the position of the items should be searched.Select Stack Search check box, the robot will search the height of pallet or sheet before picking it.

Continues on next page

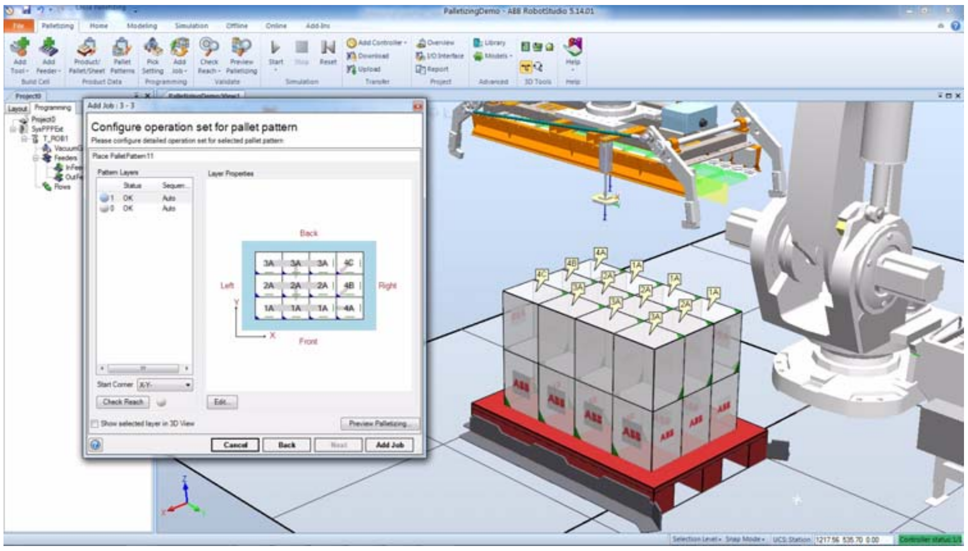
To...	Do this
Check Reach	Click Check Reach button, the reachability of all robot targets related to picking/placing the group/stack will be validated. The light will show the check result: ?tGreen: all robot targets can be reached ?tRed: some of robot targets cannot be reached. If the light is red, move mouse to the light and a tool tip will be shown, indicating which operation sets are not reachable. Select the corresponding operation set, and the unreachable items will turn red in main 3D view. See *Check Reach on page 177* for detailed information. You can adjust feeder position, flip group/stack position, or move robot to make all targets reachable.
Preview Palletizing	Click Preview Palletizing button, a preview dialog will pop out to visualize all the robot targets in the main 3D view, which you can go through step-by-step. See *Preview Palletizing on page 80* for detailed description.

Click Next button to continue to the next page.

Configure operation set for pallet pattern

After the pallet pattern and corresponding group or stack are selected in the first page, and pick settings are selected for slave feeders in second page, this page will configure the operation set for the master feeder. In general, if this job type is palletizing, the action for the operation set in this page is Placing, otherwise the action is Picking.

A new pallet pattern operation set will be created, that defines the sequence and positions of picking/placing operations, using the pick settings selected for the related groups/stacks.



xx110100032

Continues on next page

Continued

To...	Do this
Rename the Operation Set	In the Name text box, type a new name for the operation set. In default, a unique name can be generated automatically when a new operation set is created.
Edit the Pallet Pattern	Select one layer in the Pattern Layers list, this layer layout will be shown in Layer Properties view. Click Edit button, a dialog box will be shown to edit the selected layer. See *Layout Editor on page 65* for detailed description. Select Start Corner combo box, this layer will change its start corner according to the selection.
Check Reach	Click Check Reach button, the reachability of all robot targets related to picking/placing the pattern will be validated. The light will show the check result: ?tGreen: all robot targets can be reached ?tRed: some of robot targets cannot be reached. If the light is red, the unreachable items in the selected pattern will turn red in main 3D view. See *Check Reach on page 177* for details. You can adjust feeder position, flip group/stack position, or move robot to make all targets reachable.
Preview Palletizing	Click Preview Palletizing button, a preview dialog will pop out to visualize all the robot targets in the main 3D view, which you can go through step-by-step.See *Preview Palletizing on page 80* for detailed description.
Show Selected Layer in 3D View	Select Show Selected Layer in 3D View check box and select one layer in the Pattern Layers list, the selected layer will be shown in 3D View and layers above it will be invisible.

Click *Finish button* to complete the wizard.

After a job is created, the corresponding program nodes will be added into the browser, including new operation sets, flow and new jobs under the flow.

4.3.8. Check Reach

Overview

xx110000032

When any robot targets are generated in the station, it is possible to validate whether they are reachable.

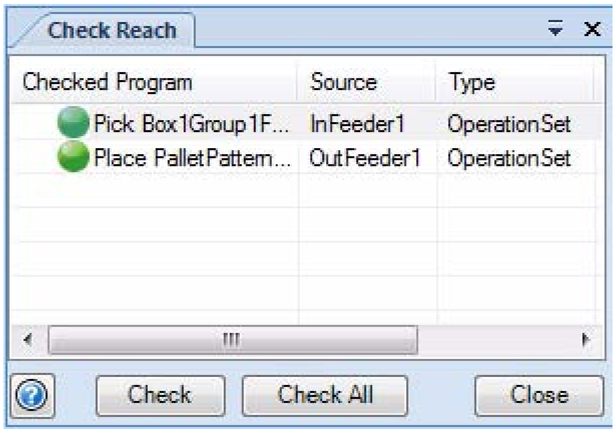
NOTE!

It is recommended that before downloading program and running simulation, all targets should be checked to be reachable.

Check Reachability

Click *Check Reach* button, all of robot targets will be checked altogether. If all robot targets are reachable, the button icon in ribbon will change to green. Otherwise it will turn to red. If there are unreachable targets, a dialog will pop up to ask whether to view the detailed information. If answer is “Yes”, an interface will show in the right window containing reach status of all the programs.

Click *Settings* on Check Reach button, the same interface as below will also show in the right window.



Checked Program	Source	Type
🟢 Pick Box1Group1F...	InFeeder1	OperationSet
🟢 Place PalletPattern...	OutFeeder1	OperationSet

xx110000033

To...	Do this
Check Selected Object	Select an object in the check list and click Check button, or double click the object, the robot targets contained by the object will be checked. The light status will change its color according to the check result. Green means that the robot targets can be reached while red means that some of robot targets cannot be reached.

Continues on next page

Continued

To...	Do this
Check All Robot Targets	Click Check All button, all objects in the list will be checked.
Show Unreachable Products	If some objects contain unreachable targets after checking, click on the object and the unreachable items in the selected object will turn red in main 3D view. Information of unreachable targets will also be shown in RobotStudio output window.

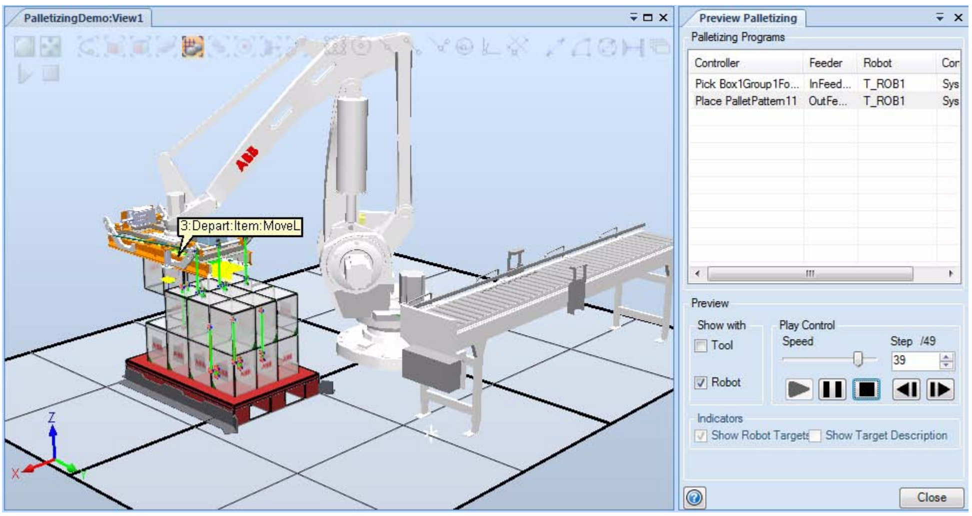
4.3.9. Preview Palletizing

Overview

xx110000034

For any operation set, the operation process can be previewed, i.e., each pick (place) robot target can be checked. The preview helps you to check the sequence of picking and placing, and check on possible collisions.

Click *Preview Palletizing* button, the preview interface will show in the right window. All existing operation sets are listed in the Palletizing Programs group.



Controller	Feeder	Robot	Cor
Pick Box1Group1Fo...	InFeed...	T_ROB1	Sys
Place PalletPattern11	OutFe...	T_ROB1	Sys

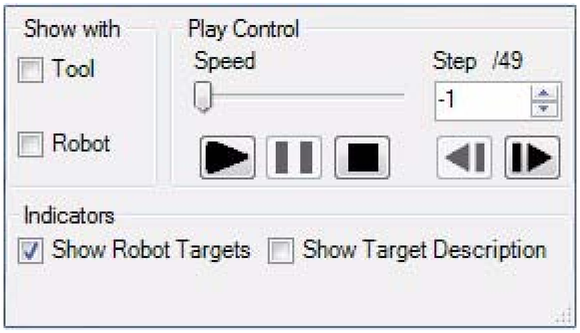
Step: 39

xx110000035

Select one operation set in the list, the preview panel will be activated.

Proceed with the Preview Panel

This section describes how to control the preview panel. All the preview result will show in main 3D view.



Step: -1

xx110000036

Continues on next page

Continued

To...	**Do this**
Control Preview Step	Click Play ▶ xx110000037 button, all preview steps will be played one by one from the current step. Click Pause ❚❚ xx110000037 button, the preview step will be paused at current step. Click Stop ■ xx110000037 button, the preview step will be initialized at the first step.Click Previous Step ◀❚ xx110000037 button or the up arrow in Steps number box, the preview will go to the previous step.Click Next Step ❚▶ xx110000037 button or the down arrow in Steps number box, the preview will go to the next step. Type in a step number in the Steps number box and press Enter on keyboard, the preview will go to the specified step.
Set Preview Speed	Drag the button in the Speed track bar, the preview play speed can be adjusted from the slowest to the fastest.
Preview with Tool	Select Tool check box, the tool will be shown during the preview process.
Preview with Robot	Select Robot check box, the robot holding the tool will jump to each robot target during the preview process.If the robot target is not reachable, the robot will not move to this robot target.
Show Robot Targets	Select Show Robot Targets check box, the robot targets related to the operation set will be shown in main 3D view.
Show Target Description	Select Show Target Description check box, the description about the robot targets will be shown in main 3D view.

4.3.10. Start Simulation

xx110000042

There are three main steps for simulation.

1. Download the project file to virtual controller.
2. Load project file into PickWare and start project.
3. Start flows and jobs.

Click Quick Start button, the three simulation steps will be executed automatically. And the operator interface will be displayed in the right window.

4.3.11. Operator Interface

Overview

xx110000043

Click *Operator Interface* button, the first two simulation steps will be executed. You can then start the flows and jobs with the buttons in this interface.

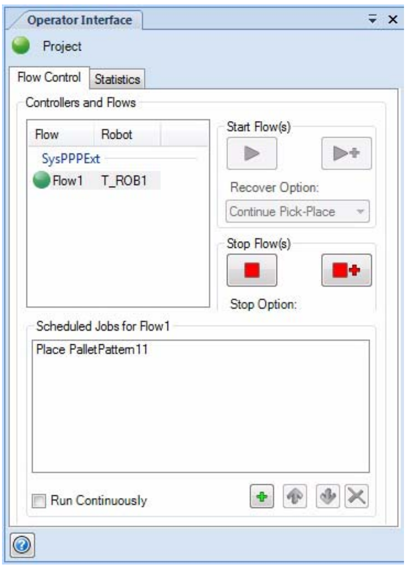
The project status is shown on top of the interface:

Project status	Description	Status Light
Started	The project has been started successfully.	🟢 xx110000044
Starting	The project is in the starting process.	🟡 xx110000045
Stopped	The project has been stopped.	⚪ xx110000046
Error	Error occurs during the project execution.	🔴 xx110000047

Below the project status, there are two pages in the operator interface:

Control Page

Control the starting and stopping of flows, and the job sequences for each flow



xx110000048

Continues on next page

Continued

All available flows are listed in the Controllers and Flows group. After the project is started, the status light will show different color to indicate flow status.

Project status	Description	Status Light
Running	The flow is running.	🟢 xx110000044
Stopping	The flow is in the stopping process.	🟡 xx110000045
Stopped	The flow has been stopped.	⚪ xx110000046
Error	Error occurs during the flow execution.	🔴 xx110000047

Proceed with the Flow Control

For the flows in the Controller and Flows group, their execution status can be controlled by the right buttons.

To...	Do this
Start a Flow	Select one flow in the Controllers and Flows group whose status is Stopped, click ▶ xx110000044 button to start the selected flow. If the selected flow is to be restarted, a flow recover option can be selected to specify how and when the flow must be restarted. If the status of the selected flow is not Stopped, this button will be disabled.
Stop a Flow	Select one flow in the Controllers and Flows group whose status is Running, click 🟥 xx110000045 button to stop the selected flow. A flow stop option can be selected to specify how and when the flow must be stopped. If the status of the selected flow is not Running, this button will be disabled.

Continues on next page

To...	Do this
Start All Flow(s)	Click ▶✚ xx110000046 button to start all flows. If some of flows are to be restarted, a flow recover option can be selected to specify how and when these flows must be restarted. This button is enabled when the status of some of the flows are Stopped.
Stop All Flow(s)	Click ■✚ xx110000047 button to stop all flows. A flow stop option can be selected to specify how and when the flow must be stopped. This button is enabled when the status of some of the flows are Running.

Proceed with the Scheduled Jobs

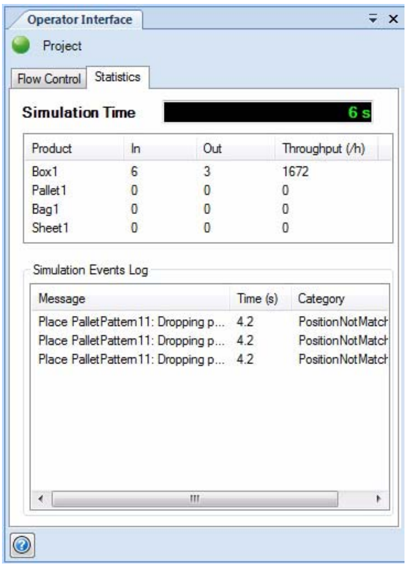
When one flow is selected, the scheduled jobs belonging to this flow will be refreshed in Scheduled Jobs group as well.

To...	Do this
Add Job	In the Scheduled Jobs group, click ✚ xx110000044 button and a dialog will pop up. The dialog shows all supported jobs of the flow. Select one job and click OK button, the selected job will be added into the list.
Delete a Job	In the Scheduled Jobs group, select one job and click ✕ xx110000045 button, the selected job will be deleted from the list.
Reorder Jobs	In the Scheduled Jobs group, select one job and click the Up or Down button to reorder the job in the list.
Repeat the Jobs	Select Run Continuously check box, the jobs in the list will be executed repeatedly.

Continues on next page

Statistics Page

Statistics Page displays the statistics information for the palletizing process, including cycle time, throughputs, and certain event logs.



Simulation Time 6 s

Product	In	Out	Throughput (/h)
Box1	6	3	1672
Pallet1	0	0	0
Bag1	0	0	0
Sheet1	0	0	0

Simulation Events Log

Message	Time (s)	Category
Place PalletPattern11: Dropping p...	4.2	PositionNotMatch
Place PalletPattern11: Dropping p...	4.2	PositionNotMatch
Place PalletPattern11: Dropping p...	4.2	PositionNotMatch

xx110000055

When the simulation is started, the statistics information will be updated in time to display the palletizing process data.

Item	Description
Simulation Time	Display the simulation time in second.
Product Data	Display how many products are inputted and outputted and the calculated data of throughput (based on outputted products and cycle time).
Simulation Event Log	Display the main event log for palletizing process.

4.3.12. Record as Viewer File

Check this option, to save the whole simulation as RobotStudio Viewer file after the simulation is stopped.

4.3.13. Speed Mode

There are three speed modes for the simulation process: Full, Customized and Low.

You can check and change the speed mode, before the simulation is started. Default mode is Low.

Note that the specified speed is effective when robot is moving without products. If robot is moving with products, the speed is limited by the motion limits of the product itself.

4.3.14. Stop Simulation

xx110000056

If the simulation is started, the Pause and Stop button will be enabled.

Click the button, the simulation will be paused. You can then step forward the simulation by continuously clicking the button.

You can click the Start button to change back and make the simulation run by itself.

4.3.15. Pause and Step Simulation

xx110000056

If the simulation is started, the Pause and Step button will be enabled.

Click the button, the simulation will be paused. You can then step forward the simulation by continuously clicking the button.

You can click the Start button to change back and make the simulation run by itself.

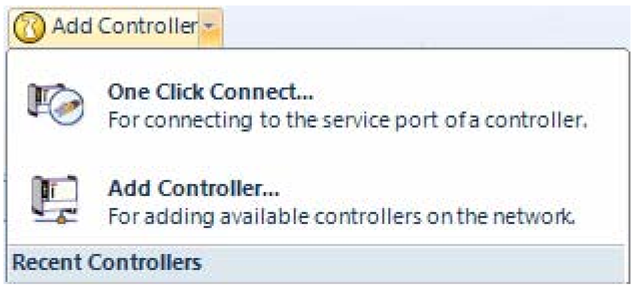
4.3.16. Reset Simulation

xx110000057

The Reset button is enabled whether simulation is started or not.

Click *Reset* button, the simulation environment will be reset, the products created during the simulation will be cleaned, and the station is ready for next simulation.

4.3.17. Add Controller


Add Controller
One Click Connect...
For connecting to the service port of a controller.
Add Controller...
For adding available controllers on the network.
Recent Controllers

xx110000058

There are two ways to connect to a controller:

?tAdd Controller - For adding available controllers to the network

?tOne Click Connect - For connecting to the service port of the controller

See Operating manual - RobotStudio for detailed information of Online functionality.

4.3.18. Download

xx110000059

Click *Download* button in ribbon, the download interface will display in the right window. And the available virtual controllers and online controllers will be listed in the table.

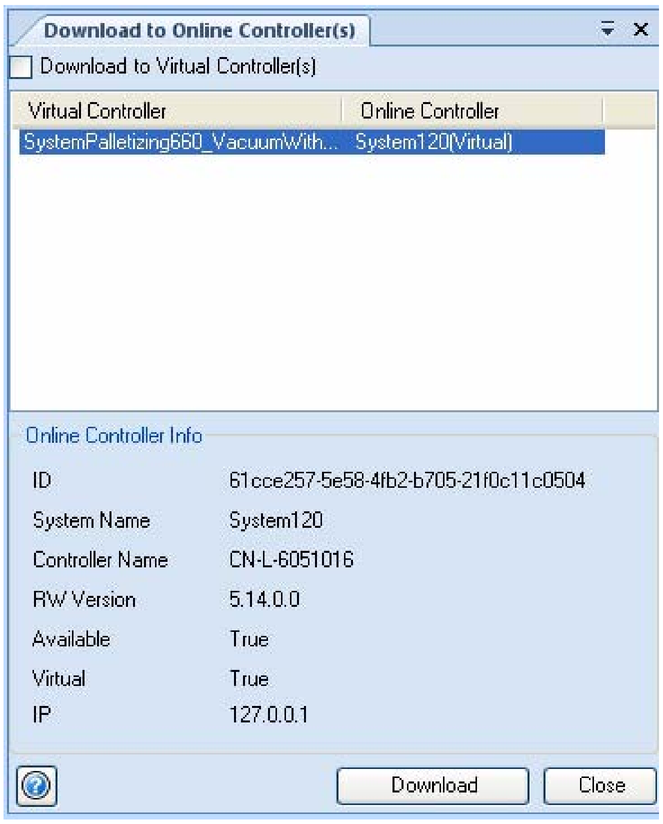
The download includes:

?tSynchronizing work object and tool data values, if downloading to virtual controller(s)

?tChecking the completeness of project setups

?tGenerating and transferring the configuration files

?tGenerating and transferring the tune file to Pickware



Download to Virtual Controller(s)

Virtual Controller	Online Controller
SystemPalletizing660_VacuumWith...	System120(Virtual)

Online Controller Info

ID	61cce257-5e58-4fb2-b705-21f0c11c0504
System Name	System120
Controller Name	CN-L-6051016
RW Version	5.14.0.0
Available	True
Virtual	True
IP	127.0.0.1

xx110000060

To...	Do this
Download Program to Online Controller	Unselect Download to Virtual Controller check box. After the online controller(s) is added as described above, the available online controller(s) will be available to select in the table's "Online Controller" column.Match each virtual controller with an online controller, and click Download button to download the program to the available online controller(s).
Download Program to Virtual Controller	Select Download to Virtual Controller check box, and click Download button to download the program to virtual controller(s).

Continues on next page

Before transferring the project, it is checked for errors to ensure that the configuration is valid.

NOTE!

The project verification when transferred to the controller does not include verification of the RAPID program.

NOTE!

It is not possible to transfer a project to a controller that is running the very same project. To update a project on a controller, the project must first be stopped.

NOTE!

If the same project already exists on the controller, the tuning will be overwritten. To preserve the tuning use function before the downloading.

4.3.19. Upload

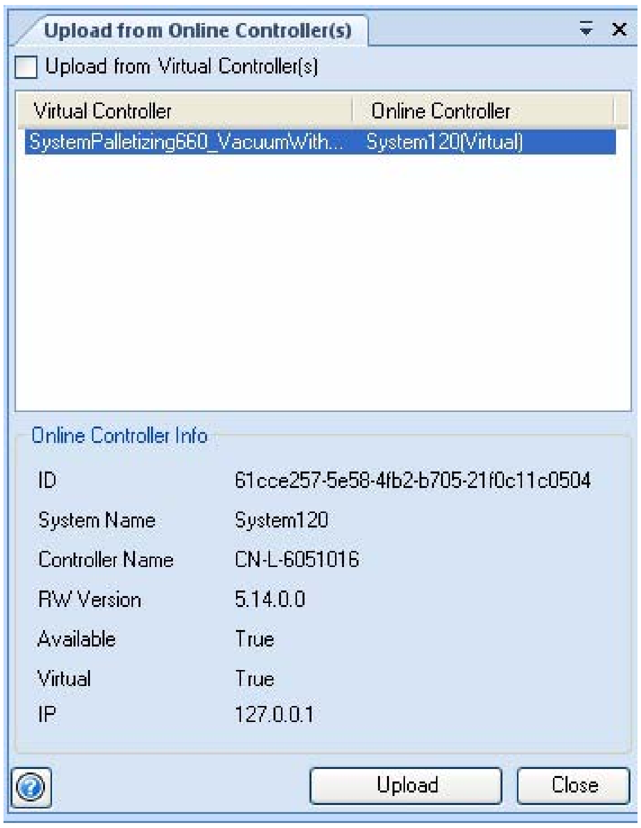
xx110000061

Click *Upload* button in ribbon, the upload interface will display in the right window. And the available virtual controllers and online controllers will be listed in the table.

The upload includes:

?tReading the work object and tool data values from controller: feeder positions and tool function positions in the project will be updated accordingly.

?tReading the tune value from Pickware: the product sizes (the graphical model), feeder tune offsets (thus operation set positions on the feeders) will be updated accordingly.


Upload from Online Controller(s)
Upload from Virtual Controller(s)
Upload
Close

Virtual Controller	Online Controller
SystemPalletizing660_VacuumWith...	System120(Virtual)

Online Controller Info

ID	61cce257-5e58-4fb2-b705-21f0c11c0504
System Name	System120
Controller Name	CN-L-6051016
RW Version	5.14.0.0
Available	True
Virtual	True
IP	127.0.0.1

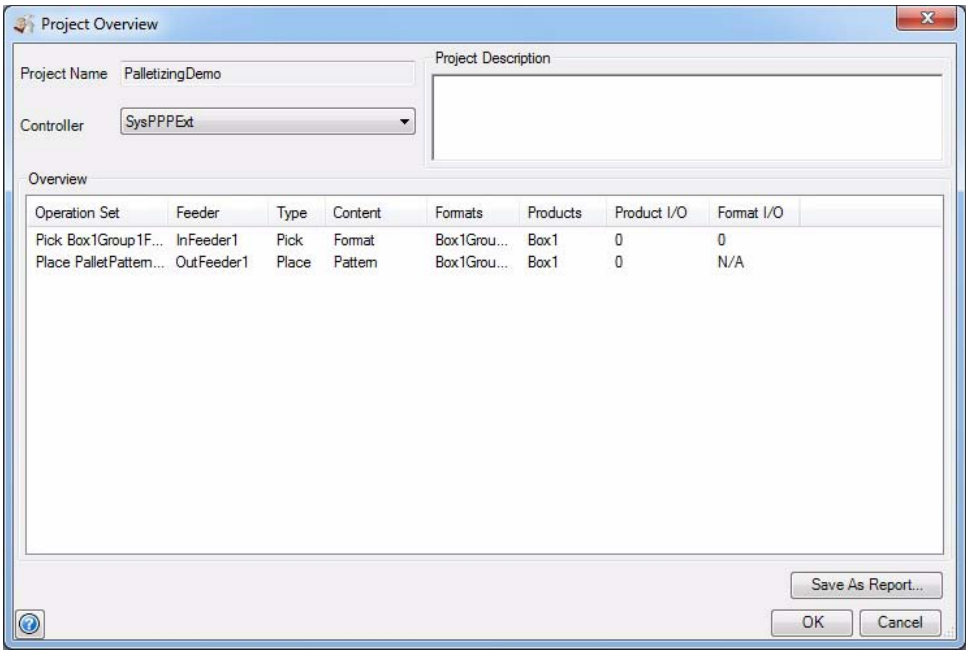
xx110000062

To...	Do this
Upload Program from Online Controller	Unselect Upload from Virtual Controller check box.After the online controller(s) is added as described above, the available online controller(s) will be available to select in the table's "Online Controller" column.Match each virtual controller with an online controller, and click Upload button to upload data from the available online controller(s).
Upload Program from Virtual Controller	Select Upload from Virtual Controller check box, and click Upload button to upload the program from virtual controller(s).

4.3.20. Project Overview

xx110000063

Click *Overview* button in ribbon to display the overview information for the project.



Project Name: PalletizingDemo

Controller: SysPPPExt

Project Description

Overview

Operation Set	Feeder	Type	Content	Formats	Products	Product I/O	Format I/O
Pick Box1Group1F...	InFeeder1	Pick	Format	Box1Grou...	Box1	0	0
Place PalletPattern...	OutFeeder1	Place	Pattern	Box1Grou...	Box1	0	N/A

xx110000064

The dialog shows each configured operation set and its corresponding product and group I/O values. You can also save this information into .txt file by clicking the “Save As Report” button.

4.3.21. I/O Interface

xx110000065

Click *I/O Interface* button in ribbon to display the I/O information for the project.

The interface is a one-stop place to view all information related with PLC for synchronizing with robot for palletizing process.

This includes all I/O values used in the projects (items and item groups, pattern operation sets, feeders, flows and projects), and all I/O signal names used in controllers, feeders, tools, and flows.

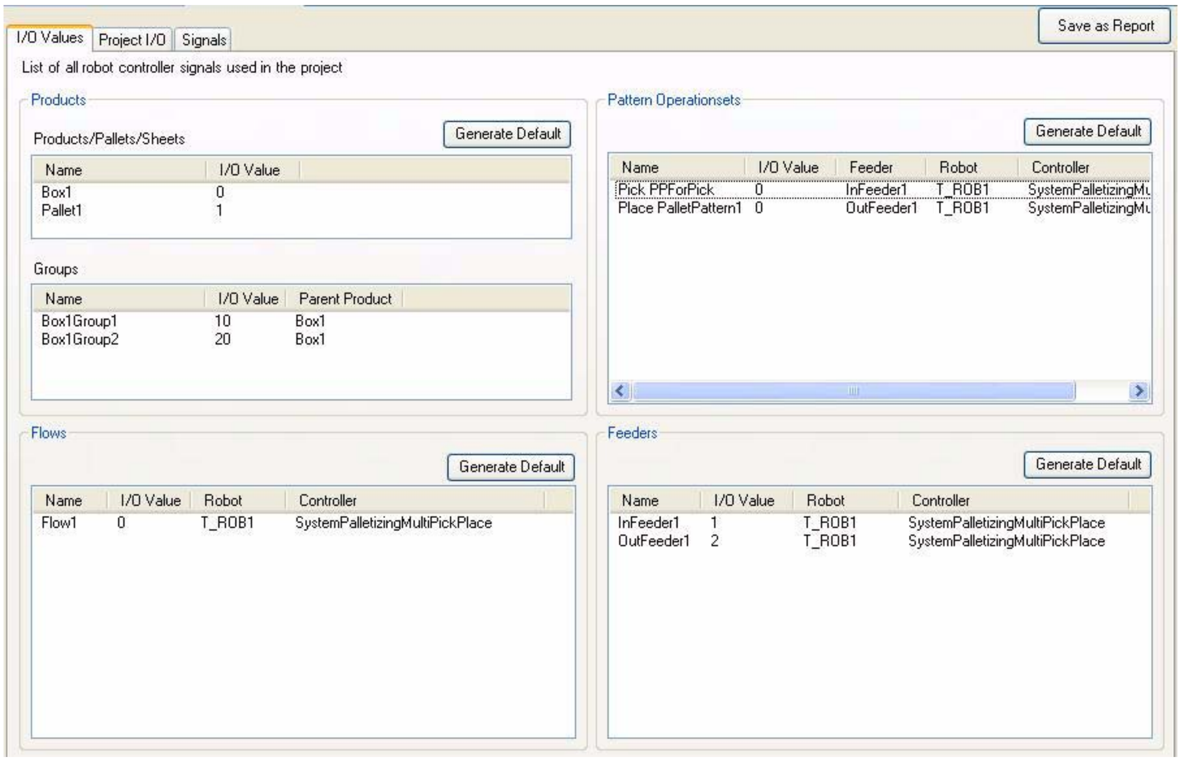
To save all the information into .txt file, click “Save As Report” button, input file name and click Save.

I/O values

To view and edit all I/O values used in the project.

For Products, Pattern operation sets, and Flows, you can also view and edit their I/O values in their editing interface respectively.

You can click “Generate Default” to generate default values for interested objects.



List of all robot controller signals used in the project

Products

Products/Pallets/Sheets

Name	I/O Value
Box1	0
Pallet1	1

Groups

Name	I/O Value	Parent Product
Box1Group1	10	Box1
Box1Group2	20	Box1

Pattern Operationsets

Name	I/O Value	Feeder	Robot	Controller
Pick PPForPick	0	InFeeder1	T_ROB1	SystemPalletizingMu
Place PalletPattern1	0	OutFeeder1	T_ROB1	SystemPalletizingMu

Flows

Name	I/O Value	Robot	Controller
Flow1	0	T_ROB1	SystemPalletizingMultiPickPlace

Feeders

Name	I/O Value	Robot	Controller
InFeeder1	1	T_ROB1	SystemPalletizingMultiPickPlace
OutFeeder1	2	T_ROB1	SystemPalletizingMultiPickPlace

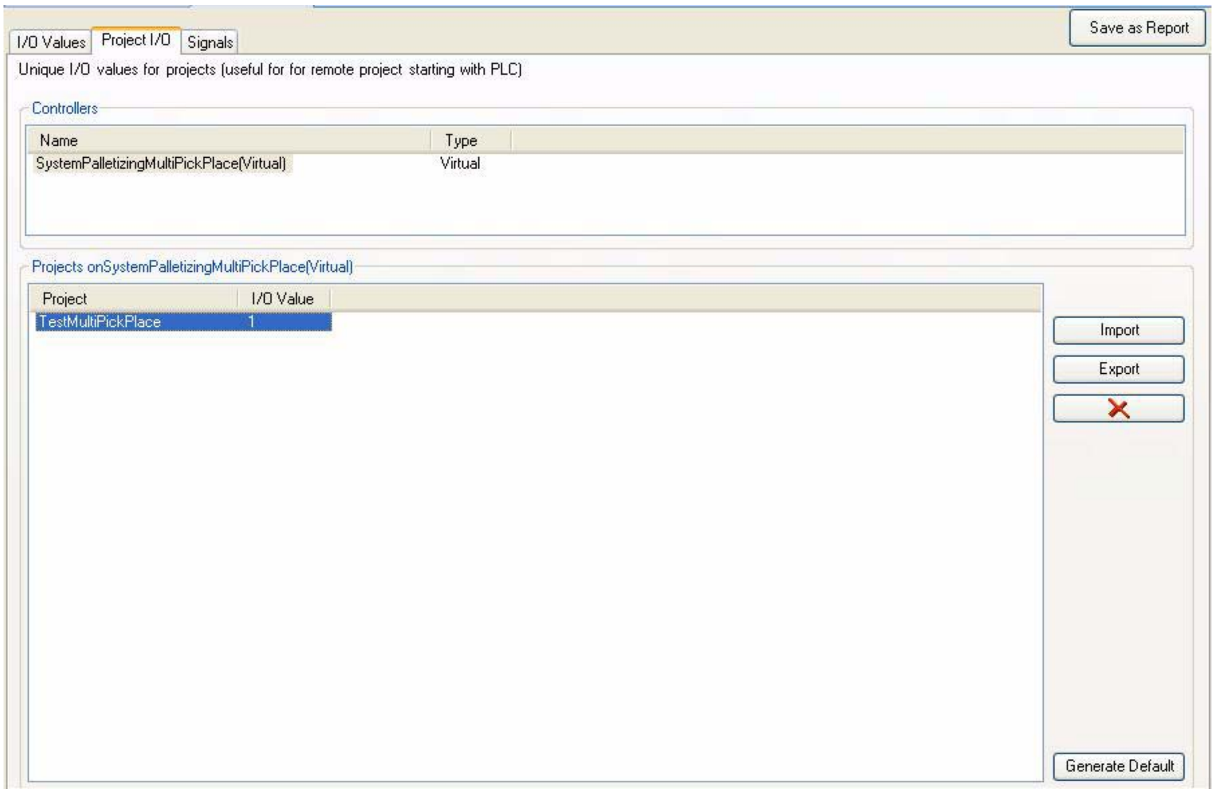
xx110000066

Continues on next page

Project Manager

If you want to start the projects using a PLC instead of the FlexPendant, then you must assign unique I/O value to each project.

The project I/O values are stored on the controller in the folder HOME:/Pickmaster/RC-Mode/ProjectMapping. A dedicated GI signal, pmProject_giSelection, specifies which project to start.



Unique I/O values for projects (useful for for remote project starting with PLC)

Name	Type
SystemPalletizingMultiPickPlace(Virtual)	Virtual

Projects onSystemPalletizingMultiPickPlace(Virtual)

Project	I/O Value
TestMultiPickPlace	1

xx110000067

The projects exist in the controller will be listed on the list view. And their I/O values will be also read from the project mapping file and shown together.

Projects with I/O value of -1 means these projects are not assigned in the mapping file and cannot be remotely started by PLC.

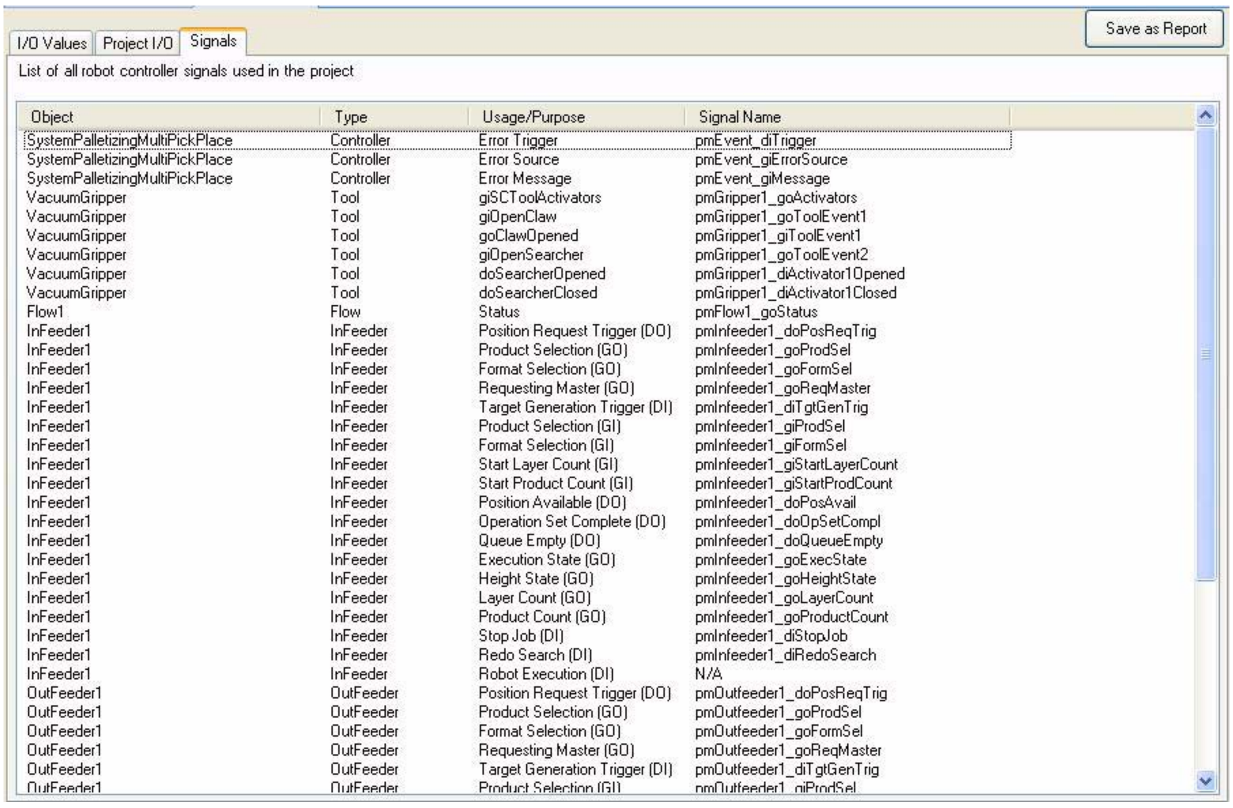
To change an I/O value for a project, you can click on the I/O value number and input a different value.

Item	Description
Generate Default	Generate default valid I/O values for project with unassigned values.
Export…	Export the project I/O value setup to an xml file.
Import	Import a project I/O value setup from an xml file. The current setup will be replaced.
Remove	Remove the project from the list. The removed project I/O value can now be used by another project.

Continues on next page

Signals

You can view all the signals used in the project, including the signal name, related object and its type, and the signal's usage/purpose.



List of all robot controller signals used in the project

Object	Type	Usage/Purpose	Signal Name
SystemPalletizingMultiPickPlace	Controller	Error Trigger	pmEvent_diTrigger
SystemPalletizingMultiPickPlace	Controller	Error Source	pmEvent_giErrorSource
SystemPalletizingMultiPickPlace	Controller	Error Message	pmEvent_giMessage
VacuumGripper	Tool	giSCToolActivators	pmGripper1_goActivators
VacuumGripper	Tool	giOpenClaw	pmGripper1_goToolEvent1
VacuumGripper	Tool	goClawOpened	pmGripper1_giToolEvent1
VacuumGripper	Tool	giOpenSearcher	pmGripper1_goToolEvent2
VacuumGripper	Tool	doSearcherOpened	pmGripper1_diActivator1Opened
VacuumGripper	Tool	doSearcherClosed	pmGripper1_diActivator1Closed
Flow1	Flow	Status	pmFlow1_goStatus
InFeeder1	InFeeder	Position Request Trigger (DO)	pmInfeeder1_doPosReqTrig
InFeeder1	InFeeder	Product Selection (GO)	pmInfeeder1_goProdSel
InFeeder1	InFeeder	Format Selection (GO)	pmInfeeder1_goFormSel
InFeeder1	InFeeder	Requesting Master (GO)	pmInfeeder1_goReqMaster
InFeeder1	InFeeder	Target Generation Trigger (DI)	pmInfeeder1_diTgtGenTrig
InFeeder1	InFeeder	Product Selection (GI)	pmInfeeder1_giProdSel
InFeeder1	InFeeder	Format Selection (GI)	pmInfeeder1_giFormSel
InFeeder1	InFeeder	Start Layer Count (GI)	pmInfeeder1_giStartLayerCount
InFeeder1	InFeeder	Start Product Count (GI)	pmInfeeder1_giStartProdCount
InFeeder1	InFeeder	Position Available (DO)	pmInfeeder1_doPosAvail
InFeeder1	InFeeder	Operation Set Complete (DO)	pmInfeeder1_doOpSetCompl
InFeeder1	InFeeder	Queue Empty (DO)	pmInfeeder1_doQueueEmpty
InFeeder1	InFeeder	Execution State (GO)	pmInfeeder1_goExecState
InFeeder1	InFeeder	Height State (GO)	pmInfeeder1_goHeightState
InFeeder1	InFeeder	Layer Count (GO)	pmInfeeder1_goLayerCount
InFeeder1	InFeeder	Product Count (GO)	pmInfeeder1_goProductCount
InFeeder1	InFeeder	Stop Job (DI)	pmInfeeder1_diStopJob
InFeeder1	InFeeder	Redo Search (DI)	pmInfeeder1_diRedoSearch
InFeeder1	InFeeder	Robot Execution (DI)	N/A
OutFeeder1	OutFeeder	Position Request Trigger (DO)	pmOutfeeder1_doPosReqTrig
OutFeeder1	OutFeeder	Product Selection (GO)	pmOutfeeder1_goProdSel
OutFeeder1	OutFeeder	Format Selection (GO)	pmOutfeeder1_goFormSel
OutFeeder1	OutFeeder	Requesting Master (GO)	pmOutfeeder1_goReqMaster
OutFeeder1	OutFeeder	Target Generation Trigger (DI)	pmOutfeeder1_diTgtGenTrig
OutFeeder1	OutFeeder	Product Selection (GI)	pmOutfeeder1_giProdSel

xx110000068

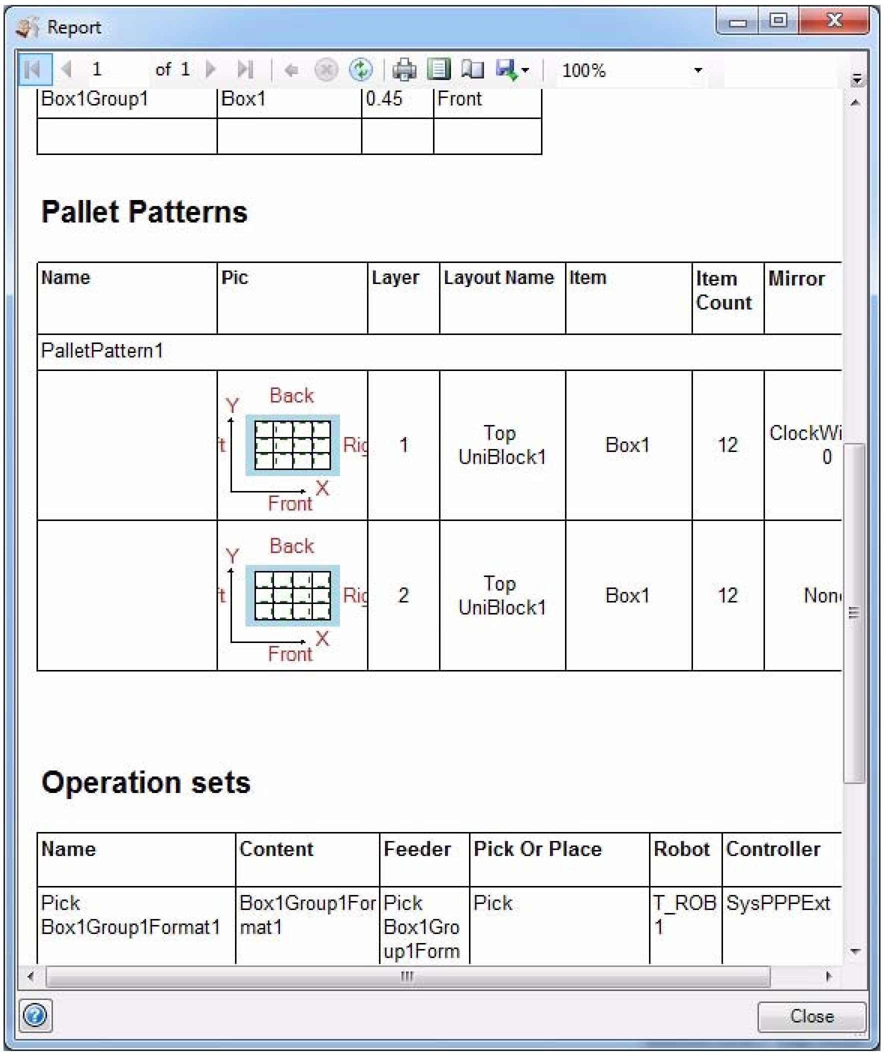
4.3.22. Project Report

xx110000069

Click *Report* button in ribbon to display the process report for the project.

The report function generates a report for all project objects, including item properties, pattern designs, operation sets, and flows.

You can save the report as a PDF or Excel file.



Box1Group1	Box1	0.45	Front

Pallet Patterns

Name	Pic	Layer	Layout Name	Item	Item Count	Mirror
PalletPattern1						
	Back / Y / t / Rig / X / Front	1	Top UniBlock1	Box1	12	ClockWi 0
	Back / Y / t / Rig / X / Front	2	Top UniBlock1	Box1	12	Non

Operation sets

Name	Content	Feeder	Pick Or Place	Robot	Controller
Pick Box1Group1Format1	Box1Group1Format1	Pick Box1Group1Form	Pick	T_ROB1	SysPPPExt

xx110000070

4.3.23. Library

xx110000071

Click *Library* button in ribbon to display the libraries used in the project.

Library contains template products, pattern layouts, controller messages and tool I/O connections.

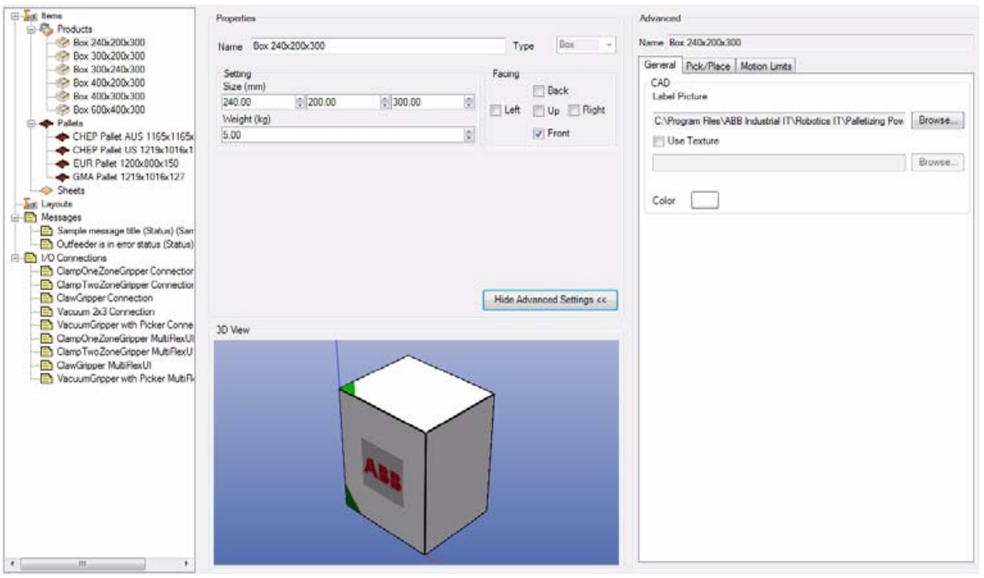
In PzPP, there are two places where library files are saved:

?tInstallation directory: for example: “C:\Program Files\ABB Industrial IT\Robotics IT\PzPP 5.14\Library”

?tUser directory: “…\My Documents\RobotStudio\Palletizing\Library\”

Library files from each location are retrieved and shown in the UI.

New libraries added are stored in the user library and only the library files from the user directory can be edited and saved.


Items
Products
Box 240x200x300
Box 300x200x300
Box 300x240x300
Box 400x200x300
Box 400x300x300
Box 600x400x300
Pallets
CHEP Pallet AUS 1165x1165x
CHEP Pallet US 1219x1016x1
EUR Pallet 1200x800x150
GMA Pallet 1219x1016x127
Sheets
Layouts
Messages
Sample message title (Status) (San
Outfeeder is in error status (Status)
I/O Connections
ClampOneZoneGripper Connectior
ClampTwoZoneGripper Connectior
ClawGripper Connection
Vacuum 2x3 Connection
VacuumGripper with Picker Conne
ClampOneZoneGripper MultiFlexUI
ClampTwoZoneGripper MultiFlexU
ClawGripper MultiFlexUI
VacuumGripper with Picker MultiFl
Properties
Hide Advanced Settings <<
3D View
Advanced
General
Pick/Place
Motion Limits
Browse...
Browse...

Properties
Name Box 240x200x300 Type Box
Setting
Size (mm) 240.00 200.00 300.00
Weight (kg) 5.00
Facing
Back [ ] Left [ ] Up [ ] Right [ ] [✓ Front]

Advanced
Name Box 240x200x300
CAD
Label Picture C:\Program Files\ABB Industrial IT\Robotics IT\Palletizing Pow
Use Texture [ ]
Color

xx110000072

Command	Description
Add an library item	Right click on a group node (i.e., Items, Layouts, Messages, Connections), and click Add context menu
Edit	Click on the node and edit on the right UI. Note only library items from user directory can be edited and saved. Otherwise, a message will be shown to user for warning.
Delete	Right click on a library item node, and click Delete context menu.
Copy an existing item	Right click on a library item node, and click Copy context menu.

4.3.24. Create Tool

4.3.24.1. Overview

xx110000073

The Create Tool interface is used to create a PzPP compatible tool SmartComponent model. The SmartComponent model can then be attached to the robot, or saved as RobotStudio library (.rslib) file, and be imported and reused in other palletizing projects.

4.3.24.2. Tool Functions

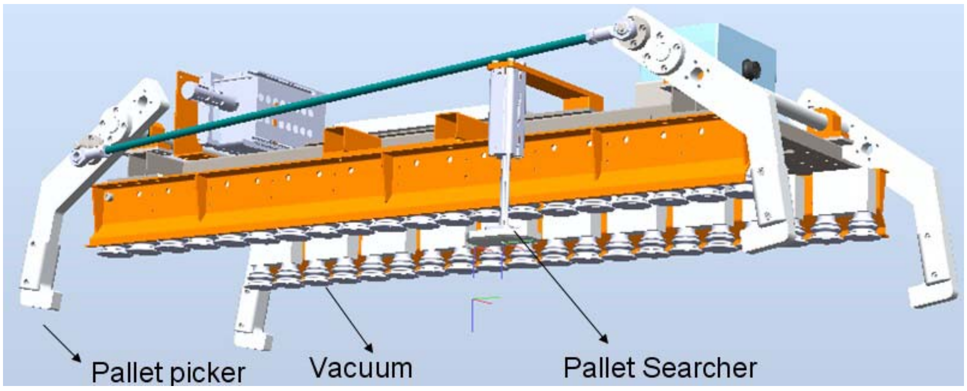
A PzPP compatible tool includes properties of:

?tTool weight, center of gravity, and inertia.

?tI/O connection template name: the I/O template that can be used as default for the tool. This information is used when the tool is attached to a robot. See *Edit Tool Signals on page 152* for detailed information.

?tTool function: A tool should contain at least one tool function. A tool function is a modular function of the tool that is controlled by certain I/O and optionally containing certain behaviors.

For example, the ABB Vacuum Tool in PzPP contains three tool functions:



xx110000074

1. Pallet pick function: a type of claw that is controlled by 2 DI signals for opening and closing, and and contains 2 DO signals to tell the opened and closed status. It is usually used to pick up a pallet.
2. Vacuum function: the vacuum is usually controlled by a GI signal to open certain groups of suction cups and close other groups of suction cups. It is usually used to pick boxes.
3. Searcher: the searcher is usually controlled by 2 DI signals for opening and closing, and and contains 2 DO signals to tell the opened and closed status. It is usually used to detect the actual height of a pallet stack. When the searcher is totally opened, and during the downward movement touches the pallet, one of the DO signal value is changed so robot will know the actually height of the pallet.

?tTool data: each tool function usually uses one tool data. Thus a tool may contain one or more tool data. These tool data, when the tool is attached to a robot, will be created in station.

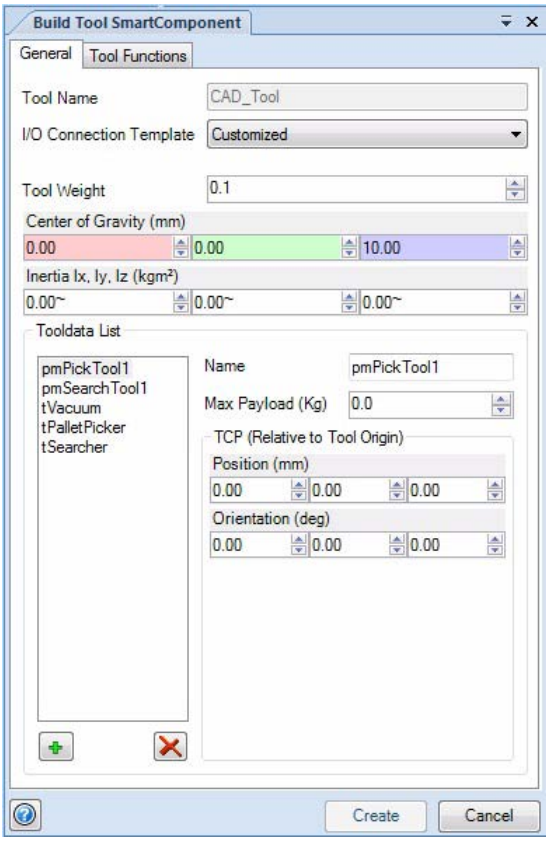
The following sections describe:

?tHow to edit tool weight, I/O template, tool data

?tHow to add/edit/remove different types of tool functions, and the related tool data

Continues on next page

General tab



Tool Name: CAD_Tool
I/O Connection Template: Customized
Tool Weight: 0.1
Center of Gravity (mm): 0.00, 0.00, 10.00
Inertia Ix, Iy, Iz (kgm²): 0.00~, 0.00~, 0.00~

Tooldata List:
- pmPickTool1
- pmSearchTool1
- tVacuum
- tPalletPicker
- tSearcher

Name: pmPickTool1
Max Payload (Kg): 0.0
TCP (Relative to Tool Origin)
Position (mm): 0.00, 0.00, 0.00
Orientation (deg): 0.00, 0.00, 0.00

xx110000075

	Description
I/O Connection template	Choose a default I/O template for this tool. The chosen template will be used as default when the tool is attached to a robot.The available templates are read from library.
Tool Weight	The weight of the tool. This value will be applied to the related tool data in station when this tool is attached to a robot.
Center of Gravity	Cog of the tool.This value will be applied to the related tool data in station when this tool is attached to a robot.
Inertia	Inertia of the tool.This value will be applied to the related tool data in station when this tool is attached to a robot.
Tool data list	A list of tool data that are contained by this tool.Some tool functions of this tool may use one of the tool data.The tool data will be created or overridden accordingly in station when this tool is attached to a robot.
Tool data Name	Name of the tool data.
Tool data Max payload	Maximum payload for this tool data.In Pick Setting dialog, if the tool function using this tool data is specified to pick products more than this payload, a warning will appear.
Tool data position and orientation	The position and orientation of the tool data.

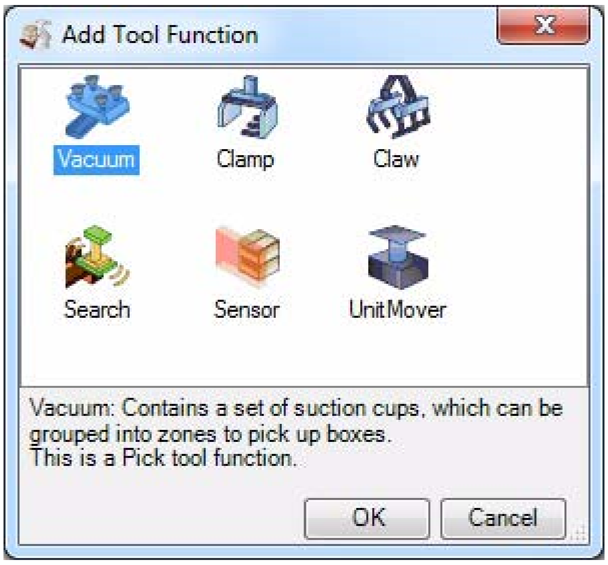
Continues on next page

To add/edit/remove tool functions

There are currently 6 types of tool functions defined in PzPP. Click the

xx110000076

button, and you can see:



Vacuum: Contains a set of suction cups, which can be grouped into zones to pick up boxes.
This is a Pick tool function.

xx110000077

Following sections describe each tool function, and their related UI interfaces

Vacuum

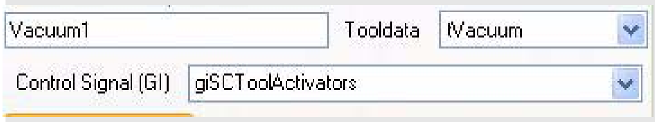
xx110000078

Continues on next page

Continued

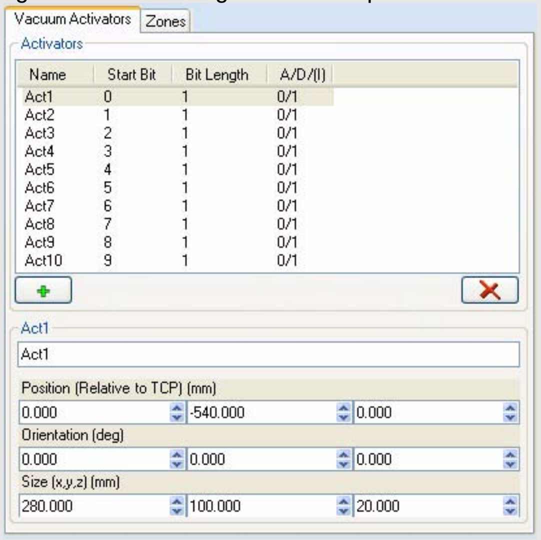
A Vacuum function contains one or more Activators, and several activators can be grouped into one zone and opened and closed together. It also contains one or more zone configurations that specify different way of grouping activators into zones.

The editing UI for Vacuum is as following:

Properties	**Description**
General	You can change the tool function name, choose the tool data, and input a control signal name for tool SmartComponent.  Vacuum1 Tooldata tVacuum Control Signal (GI) giSCToolActivators xx110000080 Note: The control signal is for the tool SmartComponent, and is not a robot controller signal.

Continues on next page

Continued

Properties	**Description**
Activators	An activator is a physical correspondent to the control signal, and controls one part of the tool, for example a vacuum cup. The figure shows a configuration example  xx110000081 To add a new activator, click xx110000082 To remove an activator, click xx110000083 To edit an activator setting, select the activator from the list and edit the value in the lower UI. For descriptions of the activator settings, see *Activators properties on page 108*.

Vacuum Activators | Zones

Activators

Name	Start Bit	Bit Length	A/D/(I)
Act1	0	1	0/1
Act2	1	1	0/1
Act3	2	1	0/1
Act4	3	1	0/1
Act5	4	1	0/1
Act6	5	1	0/1
Act7	6	1	0/1
Act8	7	1	0/1
Act9	8	1	0/1
Act10	9	1	0/1

Act1

Act1

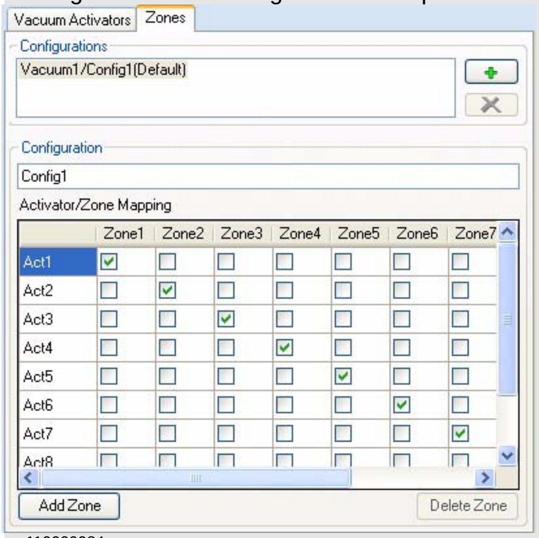
Position (Relative to TCP) (mm): 0.000 | -540.000 | 0.000

Orientation (deg): 0.000 | 0.000 | 0.000

Size (x,y,z) (mm): 280.000 | 100.000 | 20.000

Continues on next page

Continued

Properties	Description
Zones	Here you define the configurations of the zones. A zone is a collection of activators with the same state, while a configuration is a setup of zones that a tool can have. A tool can have several configurations but only one at a time can be active. The figure shows a configuration example  Configurations Vacuum1/Config1(Default) Configuration Config1 Activator/Zone Mapping | | Zone1 | Zone2 | Zone3 | Zone4 | Zone5 | Zone6 | Zone7 | | Act1 | ✓ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | | Act2 | ☐ | ✓ | ☐ | ☐ | ☐ | ☐ | ☐ | | Act3 | ☐ | ☐ | ✓ | ☐ | ☐ | ☐ | ☐ | | Act4 | ☐ | ☐ | ☐ | ✓ | ☐ | ☐ | ☐ | | Act5 | ☐ | ☐ | ☐ | ☐ | ✓ | ☐ | ☐ | | Act6 | ☐ | ☐ | ☐ | ☐ | ☐ | ✓ | ☐ | | Act7 | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ✓ | | Act8 | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | xx110000084 To add or remove a configuration, click xx110000082 or xx110000083 in the Configurations group; To set a configuration as default, right click on a configuration and select Set as Default Configu-ration; To add or remove a zone, click “Add Zone” and “Delete Zone” button. To include an activator into a zone, select the check box at the activator row and the zone column. The activators are defined at the Vacuum Activators page.

Activators properties

The following table describes the settings of the activators:

Setting	Description
Name	The activator’s name
Position	The center position of the activator related to the TCP. It is recommended to set the same z-value to all activators within one zone.
Orientation	The center orientation of the activator related to the TCP.
Size	The size of the activator.
Start bit	Defines the first bit field where the activator is connected to the Activators control signal.

Continues on next page

Continued

Setting	Description
Bit Length	The number of bit fields used for the activator.
A/D/I	Defines the Active (A), Deactivated (D), and Idle (I) states for the activator. The state for the activator is set by the values of the bits defined by the fields of the Start Bit and No. Bits. The different states: ?tActive state, which is set for an activator when it holds an item. ?tDeactivated state, which is set for an activator when it releases an item. ?tIdle state, which is set when no item is held by the activator. The values for the different states are given in the format A/D/I, that is Active/ Deactivated/Idle. To set a value, click the value of an activator and select the desired value from the drop-down combo box. If Bit Length is set to the value 1, the Idle state is not used since only two states can be given with one bit.

Clamp

xx110000086

Continues on next page

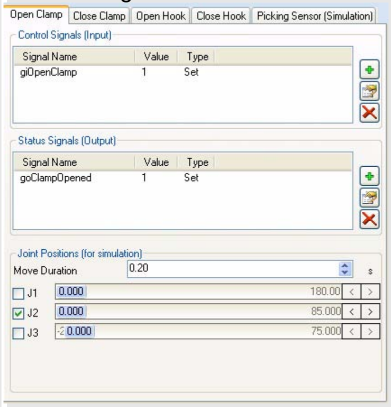
Continued

A Clamp function contains a clamp and a hook. They are usually independently controlled by different DI/GI signals.

The editing UI of Clamp is as following:

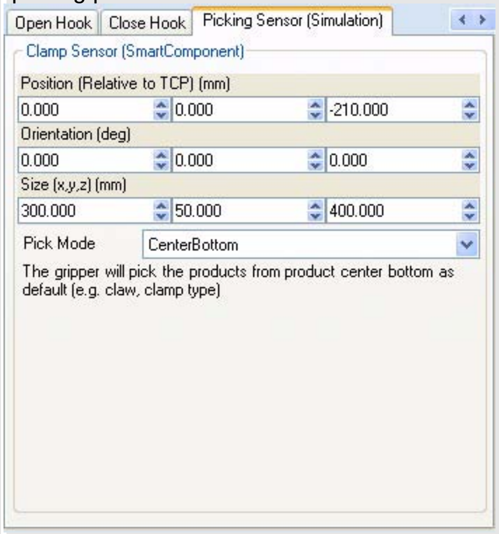
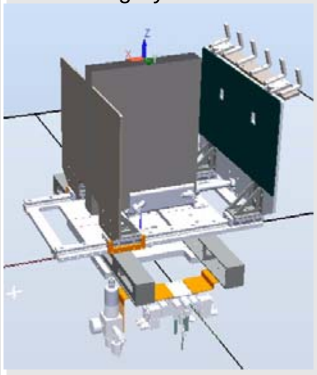
Continues on next page

Continued

Properties	Description
Open/Close Clamp	Opening and closing of clamp can be controlled by DI or GI signals, and when the open or close state is reached, it can set DO or GO signals to certain values.Also a mechanical movement in 3D view may be included to visualize the opening and closing.  Open Clamp · Close Clamp · Open Hook · Close Hook · Picking Sensor (Simulation) Control Signals (Input) Signal Name · Value · Type giOpenClamp · 1 · Set Status Signals (Output) Signal Name · Value · Type goClampOpened · 1 · Set Joint Positions (for simulation) Move Duration 0.20 s ☐ J1 0.000 180.00 ☑ J2 0.000 85.000 ☐ J3 -2 0.000 75.000 xx110000087 Take the above setting for Open Clamp as example:To add/edit/ remove a signal to trigger the open movement, click xx110000088 xx110000089 xx110000090 in the Control Signals (Input) group. You can specify the signal type (DI/DO/GI/GO), the signal set value, the pulse value and length. See *Edit Signal Control on page 118* for detailed information. To add/edit/remove a signal to be triggered when open movement is finished, click add/edit/remove in the Status Signals (Output) group. See *Edit Signal Control on page 118* for detailed information of setting signal and value. To add a joint movement for opening of clamp, select the corresponding joint and move the slider bar to the opened position. You can also click the slider bar and input the joint position directly. To edit the time that the joint takes to move to the open position, input a time value (s) in Move Duration number box. The above example specifies that: ?tWhen the input tool signal “giOpenClamp” goes to 1, the clamp will start to open. The open movement will take 0.2 seconds to move joint J2 to position 0. When the open movement is finished and the joint is at position 0, the output tool signal “goClampOpened” will be set to 1

Continues on next page

Continued

Properties	**Description**
Open/Close Hook	Same as Clamp
Picking Sensor (Simulation)	The picking sensor is mainly for simulation use. It defines an area (usually within the clamp tool) that will intersect with the products when robot moves to the picking position. In simulation, the products that intersect with this area during the closing of the clamp will be picked up by the clamp. Thus users need to set the position and size of this area with appropriate values so that this area will intersect with the products that are “intended” to be picked when the tool is at picking positions.  Open Hook / Close Hook / Picking Sensor (Simulation) Clamp Sensor (SmartComponent) Position (Relative to TCP) (mm): 0.000, 0.000, -210.000 Orientation (deg): 0.000, 0.000, 0.000 Size (x,y,z) (mm): 300.000, 50.000, 400.000 Pick Mode: CenterBottom The gripper will pick the products from product center bottom as default (e.g. claw, clamp type) xx110000091 During editing, the picking sensor area will also be shown in 3D view as a gray box:  xx110000092 Pick Mode: Pick mode defines whether the tool function should pick the products from top or bottom. This affects the default position of the tool relative to the products, defined in Pick Setting. Currently only Center Top and Center Bottom are supported: ?tCenter top: in default position, the center of the picking sensor area will be aligned with the center top of the product ?tCenter bottom: in default position, the center bottom of the picking sensor area will be aligned with the center bottom of the product

Continues on next page

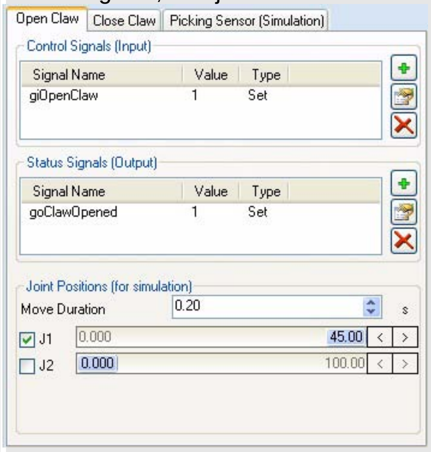
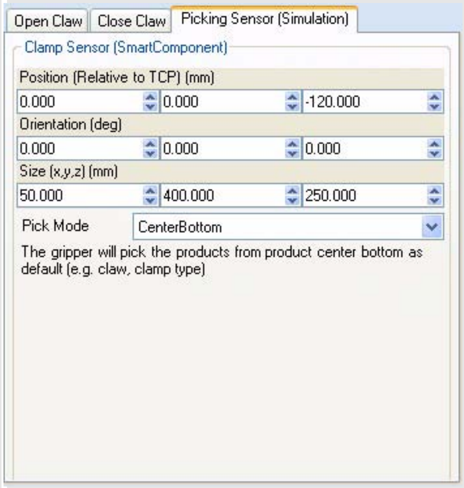
Continued

Claw

xx110000093

Simpler than clamp, a Claw function contains only a claw.

The editing UI of Claw is as following:

Properties	Description
Open/Close Claw	Opening and closing of claw is similar to that of clamp. See Open/Close Clamp to see how to edit control signals and status signals, and joint movements.  Open Claw | Close Claw | Picking Sensor (Simulation) Control Signals (Input) Signal Name: giOpenClaw, Value: 1, Type: Set Status Signals (Output) Signal Name: goClawOpened, Value: 1, Type: Set Joint Positions (for simulation) Move Duration 0.20 s J1 (checked) 0.000 45.00 J2 0.000 100.00 xx110000094
Picking Sensor (Simulation)	The picking sensor for claw is the same as that of Clamp. See Picking Sensor (Simulation) for clamp for detailed information.  Open Claw | Close Claw | Picking Sensor (Simulation) Clamp Sensor (SmartComponent) Position (Relative to TCP) (mm): 0.000, 0.000, -120.000 Orientation (deg): 0.000, 0.000, 0.000 Size (x,y,z) (mm): 50.000, 400.000, 250.000 Pick Mode: CenterBottom The gripper will pick the products from product center bottom as default (e.g. claw, clamp type) xx110000095

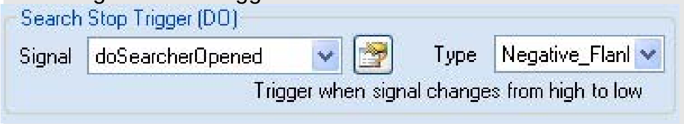
Continues on next page

Search

xx110000096

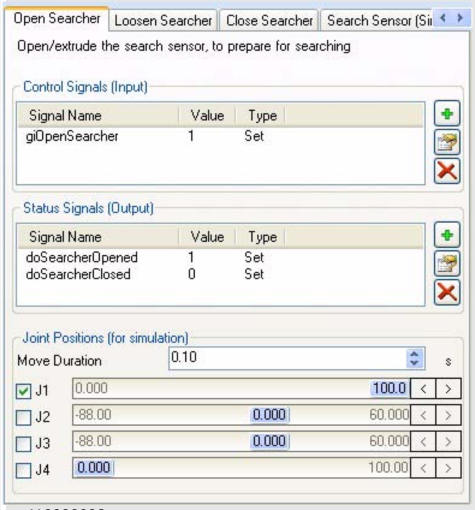
A search function contains a searcher movement, and search sensor. The movement contains the opening, the loosening, and the closing of searcher. And the search sensor describes in simulation where on the tool to detect product’s height (such as pallet stack).

The editing UI of Search is as following:

Properties	Description
General	The search stop trigger describes which tool signal to trigger when a product is detected by search sensor, and the flank type of the signal that is triggered.  Search Stop Trigger (DO) Signal: doSearcherOpened Type: Negative_Flanl Trigger when signal changes from high to low xx110000097 The trigger flank type describes what kind of signal value change indicates a search stop: ?tBoth Flanks: Trigger for any signal change ?tNegative Flank: Trigger when signal changes from high to low ?tPositive Flank: Trigger when signal change from low to high

Continues on next page

Continued

Properties	Description
Open/Loosen/Close Searcher	A typical sequence of search process includes: open searcher, loosen search, wait until search sensor touches product and stop signal triggers, close searcher. The steps related to joint movements are described as Open Seacher, Loosen Searcher, and Close Searcher.  xx110000098 The editing each movement is same as Open/Close Clamp. See Open/Close Clamp for detailed information.

Open Searcher | Loosen Searcher | Close Searcher | Search Sensor (Si

Open/extrude the search sensor, to prepare for searching

Control Signals (Input)

Signal Name	Value	Type
giOpenSearcher	1	Set

Status Signals (Output)

Signal Name	Value	Type
doSearcherOpened	1	Set
doSearcherClosed	0	Set

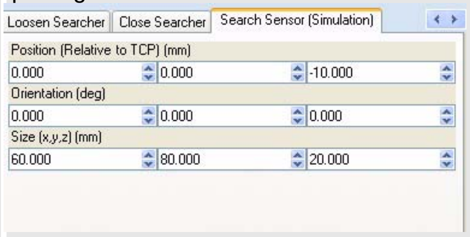
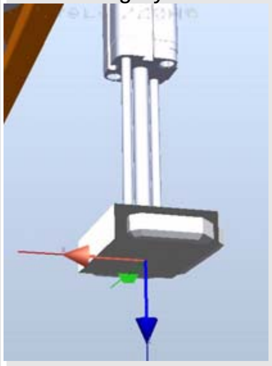
Joint Positions (for simulation)

Move Duration 0.10 s

✓ J1	0.000		100.0
J2	-88.00	0.000	60.000
J3	-88.00	0.000	60.000
J4	0.000		100.00

Continues on next page

Continued

Properties	Description
Search Sensor (Simulation)	The editing of search sensor is the same as that of Clamp, except that for search, Pick Mode information is not needed, since search is not used for picking. Note that the position of the sensor should be set carefully so that the height of the lowest position of the sensor area is the same as the height of the search TCP position. Otherwise, during simulation, the sensor SmartComponent may touch the pallet/sheet stack earlier than the TCP, and robot can not calculate the height value correctly, thus missing the product picking afterwards.  Loosen Searcher | Close Searcher | Search Sensor (Simulation) Position (Relative to TCP) (mm): 0.000, 0.000, -10.000 Orientation (deg): 0.000, 0.000, 0.000 Size (x,y,z) (mm): 60.000, 80.000, 20.000 xx110000099 During editing, the search sensor area will also be shown in 3D view as a gray box:  xx110000100

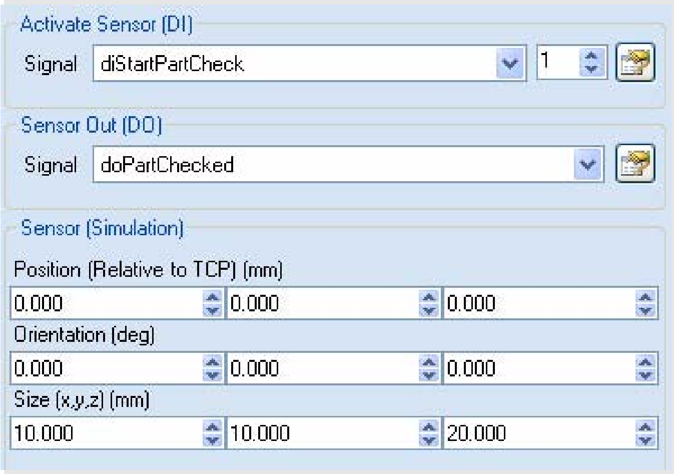
Sensor

xx110000101

Continues on next page

A sensor function is a simple function to add a product sensor and triggers certain DO when a product is detected.

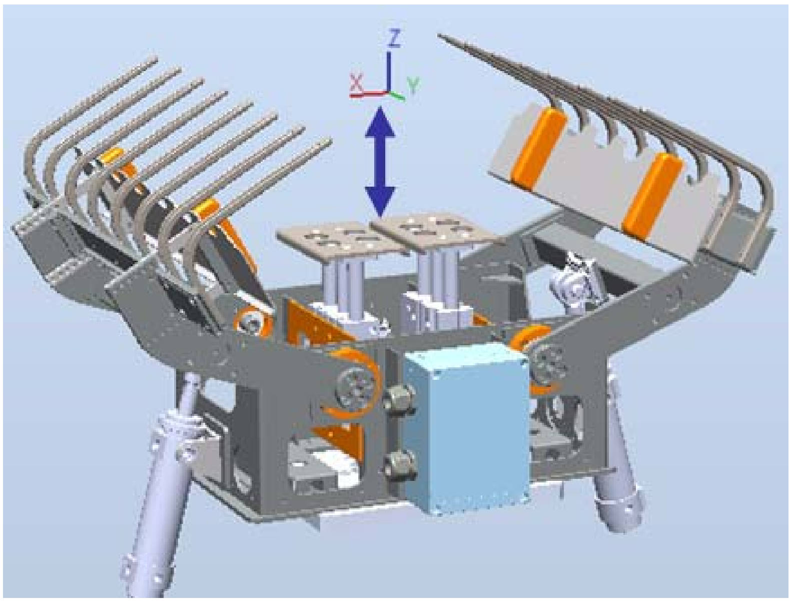
The editing UI of Sensor is as following:

Properties	Description
General	A DI signal can be specified to activate the sensor, and a DO signal should be specified to be triggered when product is detected.  Activate Sensor (DI) Signal diStartPartCheck 1; Sensor Out (DO) Signal doPartChecked; Sensor (Simulation) Position (Relative to TCP) (mm) 0.000 0.000 0.000; Orientation (deg) 0.000 0.000 0.000; Size (x,y,z) (mm) 10.000 10.000 20.000 xx110000102

Unit Mover

xx110000103

A unit mover function is a function to move some part of the tool which is not picking related. For example, a bag flatter/fixer in a claw tool is typical unit mover:

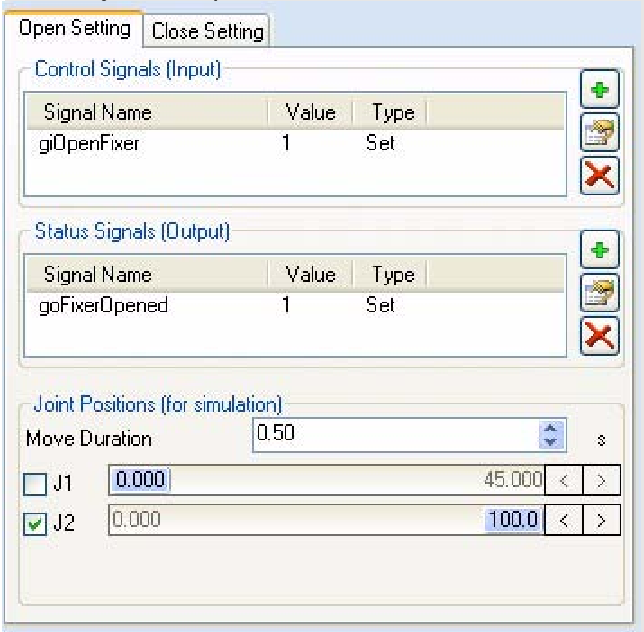


xx110000104

Continues on next page

Continued

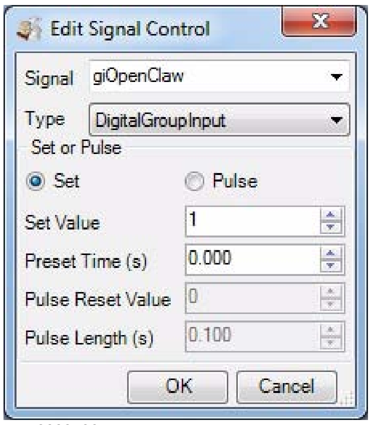
The editing UI of unit mover is as following:

Properties	Description
Open/Close Setting	Opening and closing of a Unit Mover is similar to that of clamp. See Open/Close Clamp to see how to edit control signals and status signals, and joint movements.  Open Setting / Close Setting — Control Signals (Input): Signal Name giOpenFixer, Value 1, Type Set — Status Signals (Output): Signal Name goFixerOpened, Value 1, Type Set — Joint Positions (for simulation): Move Duration 0.50 s; J1 0.000 … 45.000; J2 (checked) 0.000 … 100.0 xx110000105

Edit Signal Control

A signal control describes a signal configuration that can be used to trigger certain behavior (e.g. opening a clamp) or to show certain status (e.g. clamp is totally opened). It includes the information of the signal and the value for this signal to set to.

In Signal Control interface, you can specify such a value control for a signal.



Edit Signal Control

Signal: giOpenClaw

Type: DigitalGroupInput

Set or Pulse: [✓ Set] Pulse

Set Value: 1

Preset Time (s): 0.000

Pulse Reset Value: 0

Pulse Length (s): 0.100

xx110000106

Continues on next page

Continued

Properties	**Description**
Signal	To add a signal control on a new SmartComponent signal, click on the “Signal” list box to change it to input mode, and type the name for the new signal. Then choose the signal type at “Type” list box. The new signal can then be used other signal controls. To add a signal control on an already created signal, click on the arrow besides the “Signal” list box, and choose one. In this case you will not be able to change the signal type by the “Type” list box.
Type	Choose the type of the new signal. This is disabled if an already existing signal is selected.
Set or Pulse	This group specifies the value change property of the signal for set or pulse.If the Set option button is selected, the set value is sent at Preset Time before the robot reaches a certain target. If the Pulse option button is selected, the Reset Value is sent after Pulse Length.

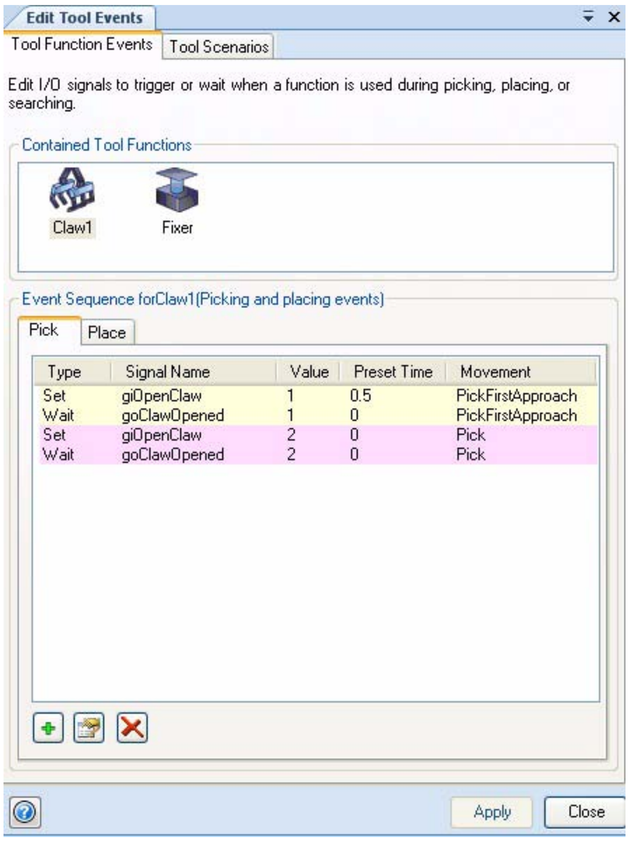
4.3.24.3. Tool Events and Scenarios

The tool events describe how the tool functions are controlled during picking and placing. It includes a list of events to control certain tool functions to open and close during different movement of picking and placing.

The tool scenarios describe which group of tool functions to use to pick different type of products. For example, the ABB Vacuum tool contains two picking tool functions: vacuum and pallet picker. When the tool is picking boxes, only vacuum is used; and when it is picking pallets, only pallet picker is used. Thus two scenarios can be created, each only including the vacuum function and the pallet picker function respectively. See *Pick Setting on page 68* on how tool scenarios are used.

When the tool is attached to the robot, a controller I/O signal will be specified to connect to each tool I/O signal (see *Edit Tool Signals on page 152*). Thus during picking/places of group and pattern operation sets, the controller signal and its value will be set or waited, according to what is specified for its connected tool I/O and value, in the tool event settings.

You can open the Edit Tool Events interface by right clicking on the tool node from browser's Layout tab or Programming tab.



Edit I/O signals to trigger or wait when a function is used during picking, placing, or searching.

Event Sequence forClaw1(Picking and placing events)

Type	Signal Name	Value	Preset Time	Movement
Set	giOpenClaw	1	0.5	PickFirstApproach
Wait	goClawOpened	1	0	PickFirstApproach
Set	giOpenClaw	2	0	Pick
Wait	goClawOpened	2	0	Pick

xx110000107

Continues on next page

Continued

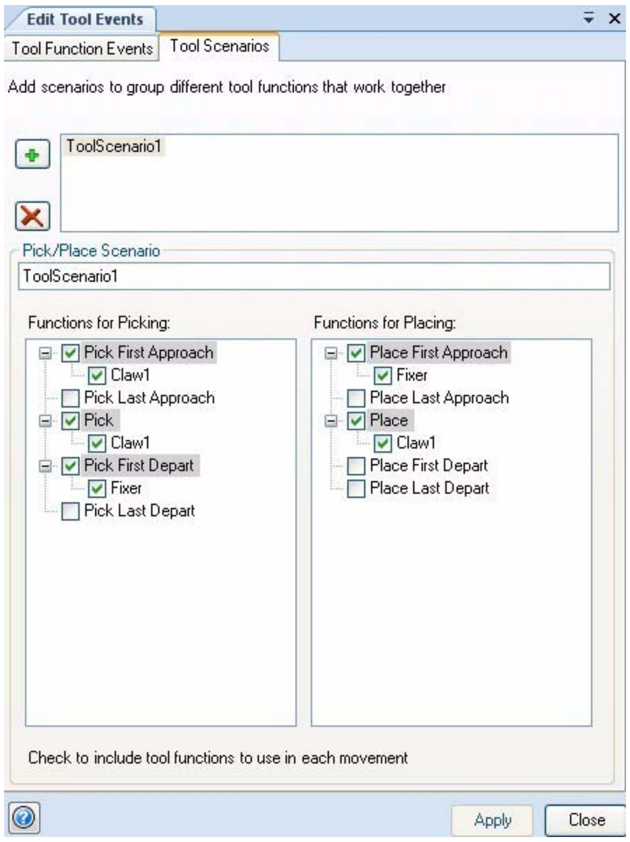
Properties	Description
Contained Tool Functions	List of contained tool functions. Each contained tool function should have a sequence of control events.
Event Sequence	The list of events for the selected tool function, in picking and placing
Type	Set or Wait. Set means the event is to set a signal to a certain value;Wait means the event is to start waiting until a signal value equals to a specified value.
Signal Name	The signal to be used in the event
Value	The set or wait value for the signal
Preset Time	The time to preset a signal to a certain value. Only valid for Set type of event.

Continues on next page

Properties	Description
Movement	The movement phase (i.e. action) of a pick/place target where this event is used. Each pick/place target contains at least 1 Approach point, 1 pick/place point, and 1 Depart point. Depending on the approach/depart setting in operation sets, one pick/place target may contain more than 1 Approach points and 1 Depart points. (See *Operation Set on page 163* for detailed information on operation, target, and actions) Accordingly, in picking and placing, there are 6 movements defined respectively: Pick First Approach ->Pick Last Approach->Pick-> Pick First Depart ->Pick Last Depart Place First Approach ->Place Last Approach->Pick-> Place First Depart ->Place Last Depart First Approach means the first approach point of a pick (or place) operation. Last Approach means the last approach point of a pick (or place) operation. If only 1 approach point was specified originally, an extra point will be added at the same location as the last approach. Pick (or Place) means the actual point where pick (or place) happens. First Depart means the first depart point of a pick (or place) operation. If only 1 depart point was specified originally, an extra point will be added at the same location as the first depart. Last Depart means the last depart point of a pick (or place) operation.
Add, edit or delete an event	To add/edit/remove an event, click ➕ xx110000088 ✎ xx110000089 ✖ xx110000090 under the event list.

Continues on next page

Continued



Add scenarios to group different tool functions that work together

ToolScenario1

Pick/Place Scenario

ToolScenario1

Functions for Picking:

- [✓] Pick First Approach
  - [✓] Claw1
- [ ] Pick Last Approach
- [✓] Pick
  - [✓] Claw1
- [✓] Pick First Depart
  - [✓] Fixer
- [ ] Pick Last Depart

Functions for Placing:

- [✓] Place First Approach
  - [✓] Fixer
- [ ] Place Last Approach
- [✓] Place
  - [✓] Claw1
- [ ] Place First Depart
- [ ] Place Last Depart

Check to include tool functions to use in each movement

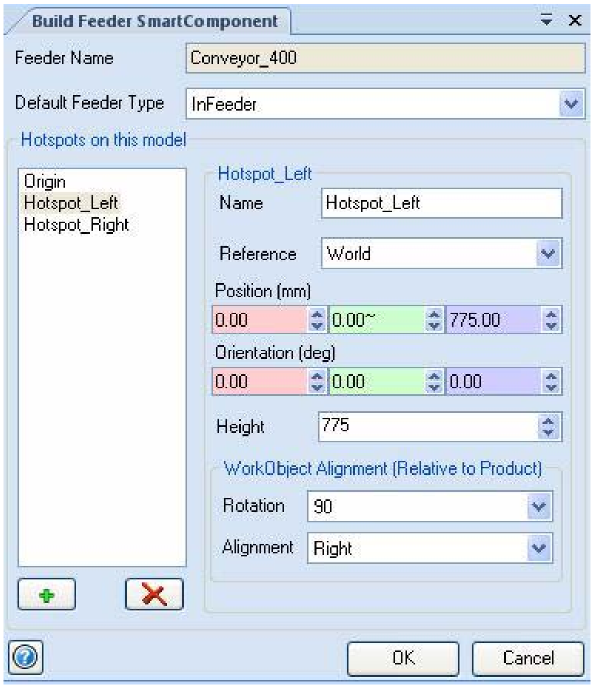
xx110000108

Properties	Description
Scenario List	List of contained tool scenarios. A tool should at least have one tool scenario.
Pick/Place Scenario	For each scenario, two tree views will be shown for Picking and Placing, including all the movements. The tool functions that contain events for the movement will be listed under the movement node. To include a tool function into a scenario at a specific movement, select the check box before the tool function node under the movement node.

4.3.25. Create Feeder

xx110000109

The Create Feeder interface is used to create a PzPP compatible feeder SmartComponent model. The SmartComponent model can then be added to the robot, or saved as RobotStudio library (.rslib) file, and be imported and reused in other palletizing projects.



Build Feeder SmartComponent

Feeder Name: Conveyor_400

Default Feeder Type: InFeeder

Hotspots on this model

- Origin
- Hotspot_Left
- Hotspot_Right

Hotspot_Left

Name: Hotspot_Left

Reference: World

Position (mm): 0.00 | 0.00~ | 775.00

Orientation (deg): 0.00 | 0.00 | 0.00

Height: 775

WorkObject Alignment (Relative to Product)

Rotation: 90

Alignment: Right

xx110000110

A PzPP compatible feeder includes properties of:

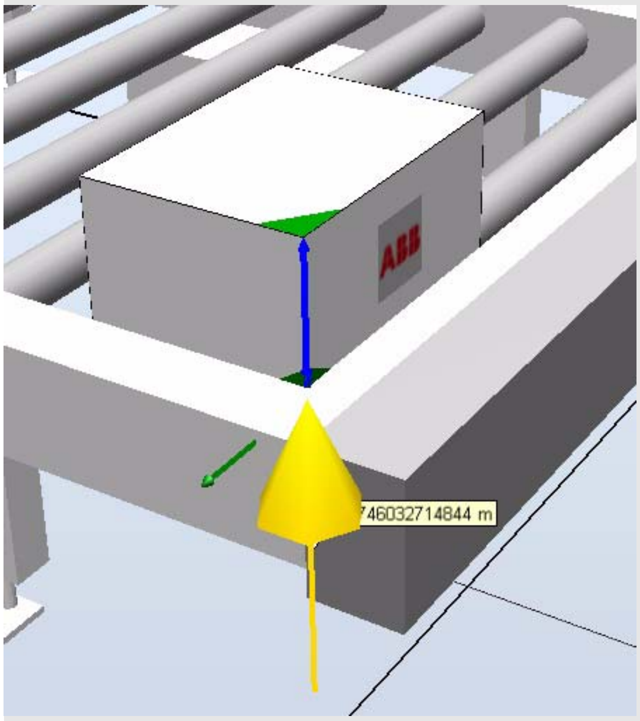
?tDefault feeder type: the default type that this feeder will be used.

?tHotspots of this feeder: a hotspot is a special frame attached with the feeder model that can be directly used to position a work object. A hotspot also describes how does the product will be shown up at this position.

Properties	Description
Default Type	The default type is used when this SmartComponent model is added for a robot, and a feeder is to be created. The default feeder type (i.e., InFeeder, OutFeeder, PalletFeeder, or Sheet-Feeder) will use the type defined here, but user can still choose to use another type. See *Add feeder on page 49* for detailed information.
Hotspots	The list of hotspots contained by this feeder.

Continues on next page

Continued

Properties	Description
General	General information: name, position, and orientation of the hotspot frame.
WorkObject/Product alignment	The group specifies the location of the work object and which quadrant the items are picked from or placed to. Adjust the Alignment and Rotation parameters to select a location suitable for calibration of the work object. Alignment: Specifies if the work object origin is located to the right or to the left of the items. Rotation: Specifies the orientation of the work object coordinate system. When you change the values for Alignment and Rotation, a dummy box and three coordinate arrows will be shown at the hotpot in 3D view, visualizing the relationship between item and work object. Note that the item’s front is attached with label.  xx110000111

NOTE!

The work object should be calibrated with its origin where items are fixed, that is along any guide and stop rails. For an infeeder or other feeders used for picking, the work object calibration is especially important. If the picking location of the tool must be adjusted, then update the tool location in Pick Setting. Avoid adjusting the displacement frame or tuning the location of the feeder, since this might reduce the accuracy when items are placed. For a palletizing feeder or other feeders where products are placed, the work object calibration can be adjusted with the displacement frame or by tuning the location of the feeder to modify the place location of items.

4.3.26. Change Viewpoint

A viewpoint stores the location and direction of a virtual camera in the 3D environment. It stores points of interest in a station and to create camera movements during simulation.

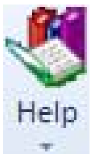
Icon	Description
xx110000112	To view the objects in the Top orientation.
xx110000113	To view the objects in the Front orientation.
xx110000114	To view the objects in the Right orientation.

4.3.27. Adjust Position of Object

The selected object in main 3D view can be adjusted with freehand.

Icon	Description
↔ xx110000115	To adjust the position of selected object in main 3D view.
⟳ xx110000116	To adjust the orientation of selected object in main 3D view.

4.3.28. Help



xx110000117

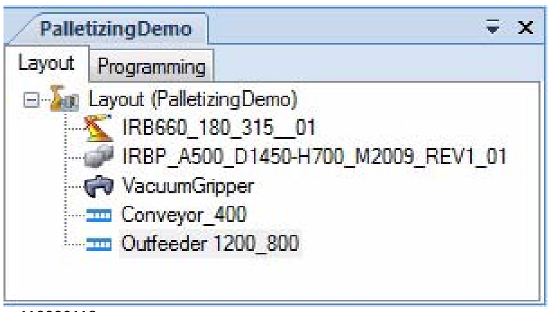
This group is related to displaying help documentation and the basic software information.

To...	Do this
Show Help File	Click Help button, a help file with chm format will display to give detailed information about this PowerPac.
Show Basic Information	Click About button, a dialog will display to show basic information about this PowerPac, such as version number, status of license, etc.

4.4 Layout Browser

4.4.1. Overview

The layout browser is a hierarchical display of physical items, such as robots and tools.



Layout (PalletizingDemo)
IRB660_180_315__01
IRBP_A500_D1450-H700_M2009_REV1_01
VacuumGripper
Conveyor_400
Outfeeder 1200_800

xx110000118

Icon	Node	Description
xx110000119	Station	The station node
xx110100120	Robot	The robot in the station.
xx110000119	Tool	A tool SmartComponent in the station. This may or may not be attached to a robot
xx110000120	Feeder	A feeder SmartComponent in the station. This may or may not be added into a robot.
xx110000121	Part	A physical object in RobotStudio. Parts with geometric information are made up of one or more 2D or 3D entities. Parts without geometric information are empty.

4.4.2. Station

You can access the context menus for the station node in the layout browser by right-clicking the station node.

Context Menus from the Station Node

Context Menu	Description
Import Model…	Import one model into this station, which should be a library component file for RobotStudio.

4.4.3. Robot

You can access the context menus for the robot node in the layout browser by right-clicking the robot node.

Context Menus from the Robot Node

Context Menu	Description
Jump Home	Jump the robot to the home position.
Mechanism Joint Jog	Jog the robot’s joint.
Set Position…	Set robot’s position.
Examine	Adjust an appropriate viewpoint for the robot.
Rename	Change the robot’s name.
Visible	Show or hide the robot.

4.4.4. Tool SmartComponent

You can access the context menus for the tool node in the layout browser by right-clicking the tool node.

Context Menus from the Tool Node

Context Menu	Description
Edit Tool Functions…	Edit tool functions for this tool.See *Tool Functions on page 103* for detailed description
Edit Tool Events…	Edit tool events for this tool.See *Tool Events and Scenarios on page 120* for detailed description
Joint Jog	Jog the tool's joint. Only available when a mechanism is used.
Disconnect Library	Disconnect the library
Save As Library	Save the library as rslib file
Attach to / Detach from	If the tool has not been attached to the robot, click Attach to menu and select the right robot, this tool will attach to the selected robot.If the tool has been attached to a robot, click Detach from menu, this tool will detach from the robot.
Examine	Adjust an appropriate viewpoint for the tool.
Rename	Change the tool's name.
Visible	Show or hide the tool.
Delete	Delete the tool from station.

4.4.5. Feeder SmartComponent

You can access the context menus for the feeder node in the layout browser by right-clicking the feeder node.

Context Menus from the Feeder Node

Context Menu	Description
Edit Feeder Hotspots…	Edit hotspots for this feeder.See *Create Feeder on page 124* for detailed description
Disconnect Library	Disconnect the library
Save As Library	Save the library as rslib file
Attach to / Attached Status	If the feeder has not been attached to the robot, click Attach to menu and select the right robot, this feeder will attach to the selected robot.If the tool has been attached to a robot, which robot is attached to will display in this context menu.
Set Position…	Set feeder’s position.
Examine	Adjust an appropriate viewpoint for the feeder.
Rename	Change the feeder’s name.
Visible	Show or hide the feeder.
Delete	Delete the feeder from station.

4.4.6. Part

You can access the context menus for the part node in the layout browser by right-clicking the part node.

Context Menus from the Part Node

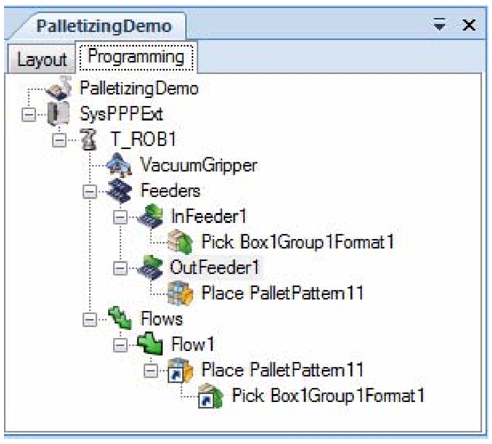
Context Menu	Description
Build into a Tool…	Build this part as a tool Smart Component, which can be used by the robot. See *Create Tool on page 102* for detailed description
Build into a Feeder…	Build this part as a feeder Smart Component, which can be used by the robot. See *Create Feeder on page 124* for detailed description.
Set Position…	Set part’s position.
Examine	Adjust an appropriate viewpoint for the part.
Rename	Change the part‘s name.
Visible	Show or hide the part.
Delete	Delete the part from station.

4.5 Programming Browser

4.5.1. Overview

The programming browser is a hierarchical display of program elements for each controller and robot.

Each virtual controller can subsequently run up to four robot nodes, here named T_ROB1.



xx110000121

Icon	Node	Description
xx110000122	Project	The project node
xx110000123	Controller	A controller that exists in the station
xx110000124	Robot	A robot (motion task) that exists in the parent controller
xx110000125	Tool	The tool used by the parent robot. It references a tool SmartComponent.
xx110000126	Feeders	The collection of feeders for the parent robot
xx110000127	Feeder	A feeder that is used by the robot. It references a feeder SmartComponent.
xx110000128	Operation Set	An Operation Set program that picks(places) from(to) the parent feeder. Depending on the type of item it handles (box, bag, pallet, or sheet), and whether it is picking or placing, the icon is different.

Continues on next page

Continued

Icon	**Node**	**Description**
xx110000129	Flows	The collection of flows for the parent robot
xx110000130	Flow	A flow that exists for the parent robot
xx110000131	Job (Master Operation Set)	A supported job for the parent flow. It actually references to an Operation Set of the flow's master feeder.
xx110000132	Slave Operation Set	Reference to an Operation Set on the slave feeders that are related to the master job.

4.5.2. Project

4.5.2.1. Introduction

You can access the context menus for the project node in the programming browser by right-clicking the project node.

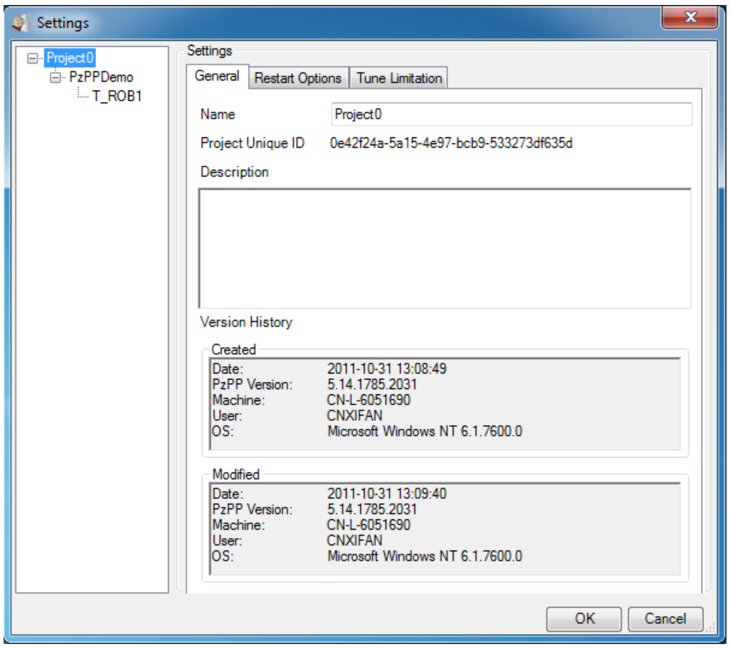
Context Menus from the Project Node

Context Menu	Description
Settings	Edit settings for the project. See *Project Settings on page 138* for detailed description
Rename	Change the Project's name.

4.5.2.2. Project Settings

This dialog is shared with project, controller and robot settings. The left tree view is a hierarchical display of project elements for each controller and robot.Select the project node and its settings will display in the right three pages.

General



Name: Project0

Project Unique ID: 0e42f24a-5a15-4e97-bcb9-533273df635d

Description

Version History

Created

```
Date:          2011-10-31 13:08:49
PzPP Version:  5.14.1785.2031
Machine:       CN-L-6051690
User:          CNXIFAN
OS:            Microsoft Windows NT 6.1.7600.0
```

Modified

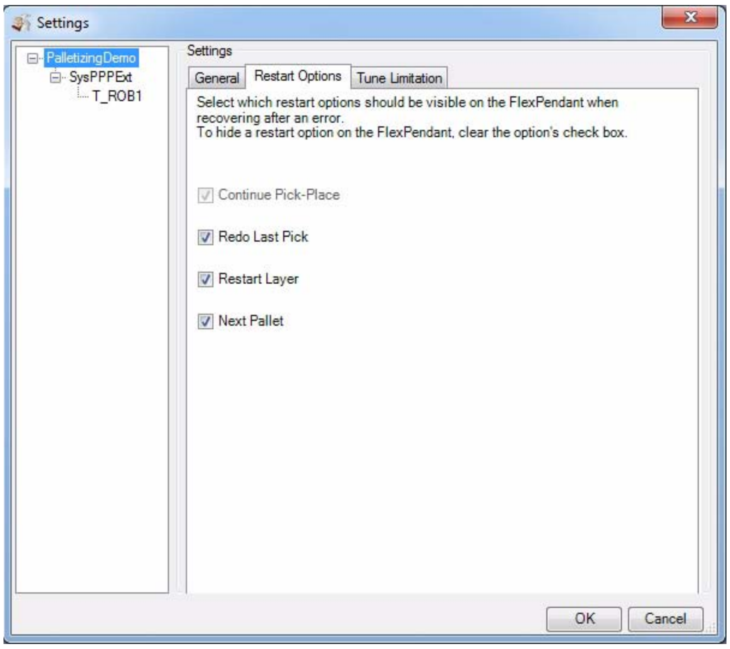
```
Date:          2011-10-31 13:09:40
PzPP Version:  5.14.1785.2031
Machine:       CN-L-6051690
User:          CNXIFAN
OS:            Microsoft Windows NT 6.1.7600.0
```

xx110000133

The project name and description can be edited in this page.

Continues on next page

Restart Options



Settings

- PalletizingDemo
  - SysPPPExt
    - T_ROB1

Settings

General | Restart Options | Tune Limitation

Select which restart options should be visible on the FlexPendant when recovering after an error.
To hide a restart option on the FlexPendant, clear the option's check box.

- [✓] Continue Pick-Place
- [✓] Redo Last Pick
- [✓] Restart Layer
- [✓] Next Pallet

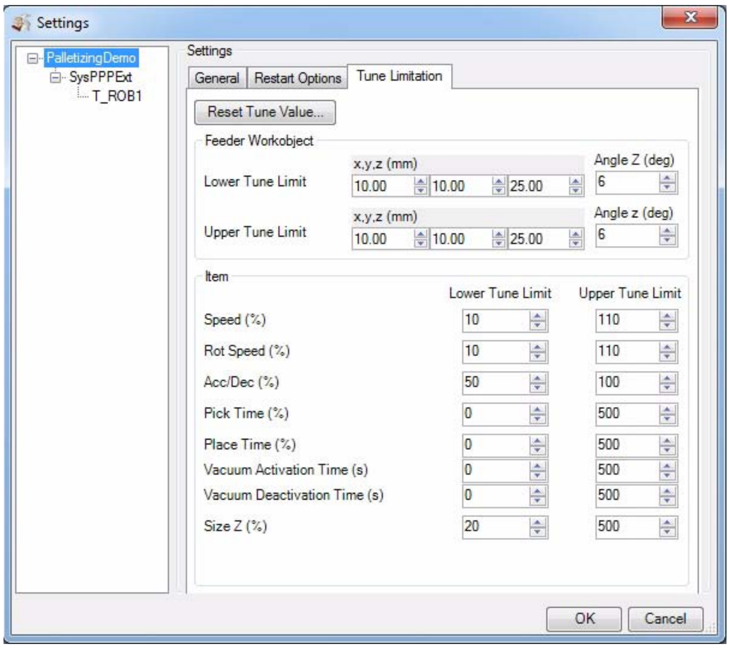
OK | Cancel

xx110000134

Select which restart options should be visible on the FlexPendant when recovering after an error. All options are selected as default. To hide a restart option on the FlexPendant, clear the option's check box. See *Flow on page 184*.

Continue Pick-Place is always available.

Tune Limitation



Settings

- PalletizingDemo
  - SysPPPExt
    - T_ROB1

Settings

General | Restart Options | Tune Limitation

Reset Tune Value...

Feeder Workobject

	x,y,z (mm)			Angle Z (deg)
Lower Tune Limit	10.00	10.00	25.00	6
Upper Tune Limit	10.00	10.00	25.00	6

Item

	Lower Tune Limit	Upper Tune Limit
Speed (%)	10	110
Rot Speed (%)	10	110
Acc/Dec (%)	50	100
Pick Time (%)	0	500
Place Time (%)	0	500
Vacuum Activation Time (s)	0	500
Vacuum Deactivation Time (s)	0	500
Size Z (%)	20	500

OK | Cancel

xx110000135

Continues on next page

It is used to set the minimum and maximum limits for how much each parameter can be tuned online. For more information about how to tune the parameters online, see *Runtime operation on page 229*.

Feeder WorkObject Group

Click *Reset Tune Value* button, all tune value can be reset.

In the Feeder WorkObject group the limits are specified using the same unit as the parameter. The x, y, z, and z angle defines the displacement of the feeder relative to the default value. The Lower tune limits define the maximum displacement from the default values in the negative direction. The Upper tune limits define the maximum displacement from the default values in the positive direction.

Example

?tAssume the default value of Z angle in the Group Operation Set Configuration window is 25 degrees. See *Group Operation Set on page 165* Configuration.

?tThe Lower tune limit in the Tune Limitations window is set to 6 degrees. This means you can tune the z-angle to maximum 6 degrees less than the default value, that is, you can tune the Angle (z) to a minimum of 19 degrees.

?tThe Upper tune limit in the Tune Limitations window is set to 6 degrees. This means you can tune the z-angle to maximum 6 degrees more than the default value, that is, you can tune the Angle (z) to a maximum of 31 degrees.

The following table describes the parameters of the Feeder WorkObject group

Item	Description
X	Displacement of the feeder in x-direction relative the work object.
Y	Displacement of the feeder in y-direction relative the work object.
Z	Displacement of the feeder in z-direction relative the work object.
Angle(Z)	Displacement angle of the feeder in the z-direction.

Item Group

In the Item group all limits are specified in percent (%) relative the default value, which is referred to as 100%.

Example

?tAssume the default speed in the Motion Limits window is 1000 mm/s. This is referred to as 100%. See *Advanced setting for Item on page 57*.

?tThe Lower tune limit in the Tune Limitations window is 10% of the default speed. Hence, you can tune the speed to a minimum of 100 mm/s.

?tThe Upper tune limit in the Tune Limitations window is 110% of the default speed. Hence, you can tune the speed to a maximum of 1100 mm/s.

The following table describes the parameters of the Item group:

Item	Description
Speed	Specifies maximum speed for an item.
Rot Speed	Specifies maximum orientation speed for an item.
Acc/Dec	Specifies maximum acceleration/deceleration for an item.

Continues on next page

Continued

Item	Description
Pick Time	Specifies the time that the robot stays at the target position when picking an item.
Place Time	Specifies the time that the robot stays at the target position when placing an item.
Vacuum Activation Time	Specifies the time for vacuum activation.
Vacuum Deactivation Time	Specifies the time for vacuum deactivation.
Size (z)	Specifies the height of an item.

4.5.3. Controller

4.5.3.1. Introduction

You can access the context menus for the controller node in the programming browser by right-clicking the controller node.

Context Menus from the Controller Node

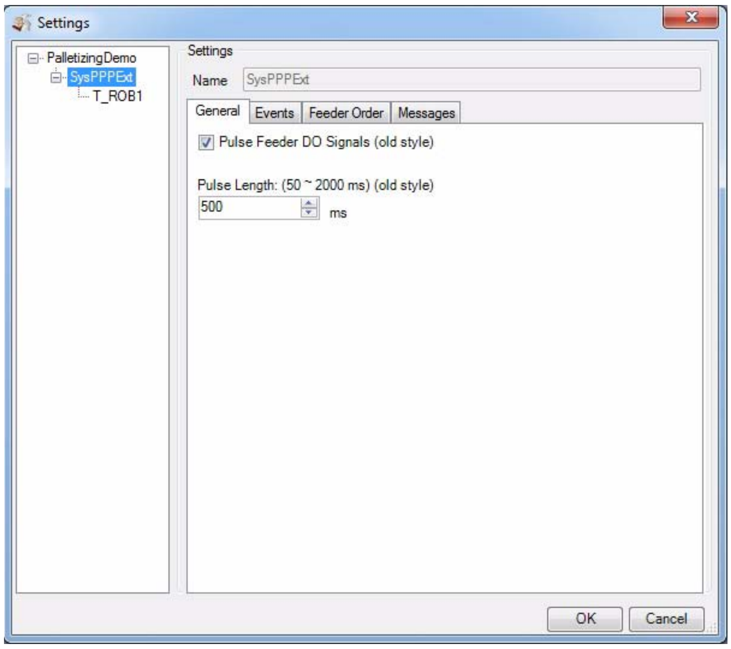
Context Menu	Description
Settings	Edit settings for the controller. See *Controller Settings on page 143* for detailed description

4.5.3.2. Controller Settings

This dialog is shared with project, controller and robot settings. The left tree view is a hierarchical display of project elements for each controller and robot.

Select the controller node and its settings will display in the right four pages.

General



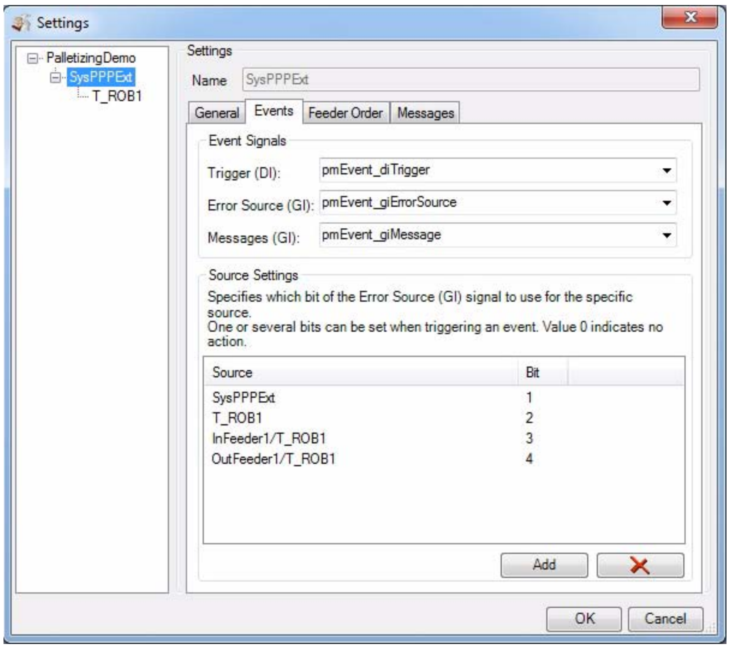
xx110000136

Item	Description
Pulse Feeder DO Signals	If set, the Position request trigger signals and the Operation set complete signals will be pulsed instead of set by the controller. It is not recommended to set the pulse signals.
Pulse Length	Here you specify the pulse length that will be used if Pulse Feeder DO signals is selected. The pulse length can be set in the range of 50 ms to 2000 ms.

Continues on next page

Continued

Events


Settings

PalletizingDemo
SysPPPExt
T_ROB1

Settings
Name: SysPPPExt

General | Events | Feeder Order | Messages

Event Signals
Trigger (DI): pmEvent_diTrigger
Error Source (GI): pmEvent_giErrorSource
Messages (GI): pmEvent_giMessage

Source Settings
Specifies which bit of the Error Source (GI) signal to use for the specific source.
One or several bits can be set when triggering an event. Value 0 indicates no action.

Source	Bit
SysPPPExt	1
T_ROB1	2
InFeeder1/T_ROB1	3
OutFeeder1/T_ROB1	4

xx110000137

Item	Description
Event Signals	This group contains the signals that are used by an external equipment to report a message or error to the PickMaster process.
Source Settings	This group contains the information on which source (feeder, robot and/or robot controller) is affected by a reported error.

Event Signals Group

Item	Description
Trigger (DI)	Specifies which digital input signal will be used to trigger an event. When this signal goes high, the event will be reported to the PickMaster process. This signal must be used together with the Error Source (GI) and/or Messages (GI) signal.
Error Source (GI)	A group input signal representing the source of an error. If the signal is defined it indicates where the error has occurred (feeder, robot and/or controller) when the Trigger (DI) signal is set, and thus where a flow recovery action should be performed to handle the error. For information about the source setting, see below *Source Settings Group*.
Messages (GI)	A group input signal representing an event message. If the signal is defined it specifies which message will appear on the FlexPendant when the Trigger (DI) signal is set.

Continues on next page

Continued

Source Settings Group

The Source Settings group needs to be configured if the Error Source (GI) signal in the Signals group is used. See *Flow recovery on page 240*.

Item	Description
Source	Specifies the feeder, robot and/or controller that can be affected by the reported error.
Bit	Specifies which bit of the Error Source (GI) signal to use for the specific source. One or several bits can be set when triggering an event. Value 0 indicates no action.

If the source points out a feeder, then the feeder will be set in error state when the Trigger (DI) peaks. The current state of a feeder is indicated by the FlexPendant interface, see *Viewing flow status on page 236*. It is also indicated by the feeder I/O signal Execution state (GO), see the *Feeder on page 154* Configuration .

A feeder in error state cannot be used in any pick or place operations.

A flow can continue to operate if there are alternative feeders to operate on. If there is no alternative feeder the flow will continue to operate until the feeder in error state is needed. The RAPID execution stops if there is only one flow.

If one feeder is shared between several flows all flows will stop when the feeder goes to error state. Setting one feeder in error state can result in error state on other feeders as well. The PickMaster process automatically sets affected feeders in error state.

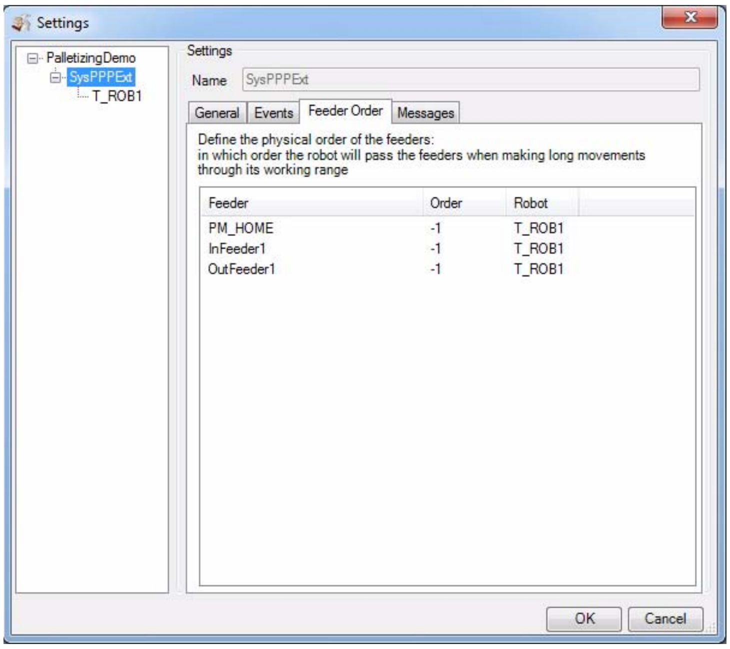
If you plan to use Redo last pick then we recommend that you set the place feeder in error state rather than the pick feeder. Setting the pick feeder in error state after the pick operation is completed will not stop the robot movement. In this case the robot will continue to deliver the faulty products.

Flow recovery can be used from the FlexPendant interface or the I/O interface to recover from errors.

For examples, see *Event and error reporting on page 270*.

Continues on next page

Feeder Order



Define the physical order of the feeders:
in which order the robot will pass the feeders when making long movements through its working range

Feeder	Order	Robot
PM_HOME	-1	T_ROB1
InFeeder1	-1	T_ROB1
OutFeeder1	-1	T_ROB1

xx110000138

The feeder order settings define the physical order of the feeders, that is, in which order the robot will pass the feeders when making long movements through its working range.

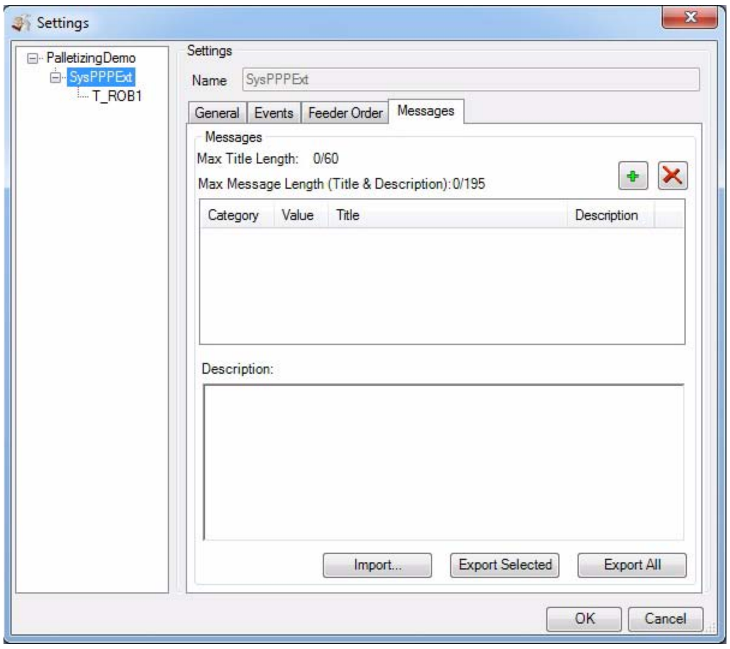
The feeder order settings are used to define which feeders that will be considered when planning the path height of an intermediate movement between two feeders. Also, when moving to or from the home position, the feeder order settings will affect the path height. Each feeder and the home position are given an order number. When planning an intermediate movement, the order number of the start feeder and the end feeder (where none, one or both of them may be the home position) will set the upper and lower order limits of order numbers for other feeders to be considered when planning the path height. The order numbers can be freely chosen and two feeders can have the same order.

PM_HOME is a default installed feeder with work object pm_homeWObj. The order number for this feeder should be where the home position is located.

Continues on next page

Continued

Messages



Name: SysPPPExt

Messages
Max Title Length: 0/60
Max Message Length (Title & Description): 0/195

Category	Value	Title	Description

Description:

xx110000139

Item	Description
Max Title Length	Displays the title length. There can be maximum 60 characters in the message title.
Max Message Length	Displays the total message length. Each message can contain maximum 195 characters, including the title.
Import	Click Import to import messages from PickMaster library.
Export Selected	Click Export Selected to export a message to PickMaster library.
Export All	Click Export All to export all messages to PickMaster library.
Add	Click ➕ xx110000088 to add a new message. If the controller is connected, the first available value will be given to the new message.
Delete	Click ❌ xx110000090 to delete a message.
Category	Click the Category column and select a category in the drop-down list. The category controls how the message is presented on the FlexPendant.
Value	Click the Value column and select a value or type a value in the drop-down list. The value is mandatory.
Title	Click the Title column and type the title.
Description	Click the Message text group and type the description for the selected message.

4.5.4. Robot

4.5.4.1. Introduction

You can access the context menus for the robot node in the programming browser by right-clicking the robot node.

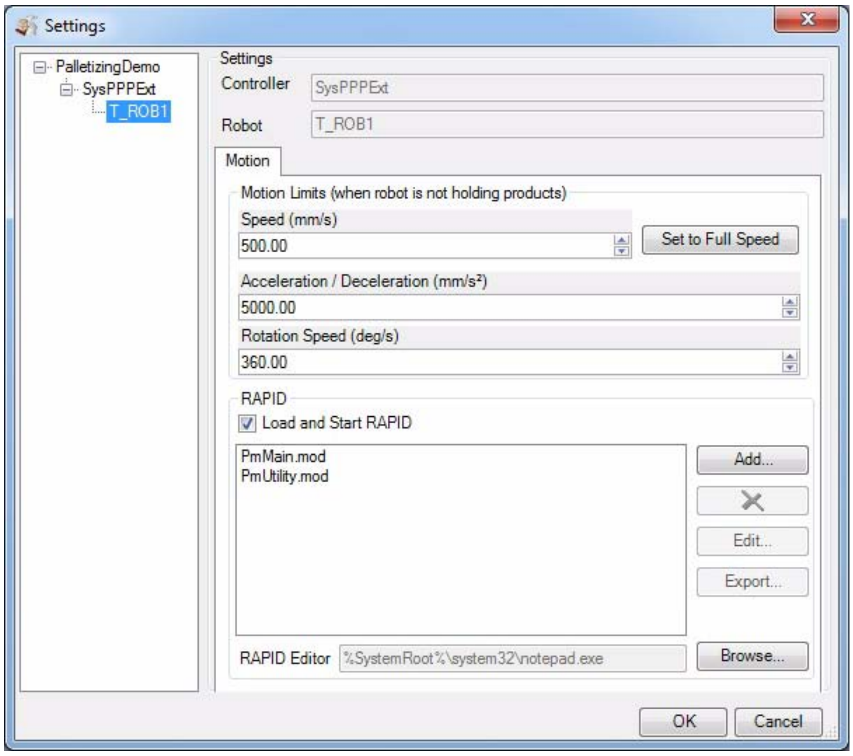
Context Menus from the Robot Node

Context Menu	Description
Attach Tool…	If there is no tool attached to the robot, this context menu will display. It is to add a tool to the robot. See *Add tool on page 47* for detailed description.
Settings	Edit settings for the robot. See *Robot Settings on page 149* for detailed description
Jump Home	Jump the robot to the home position.
Set Position…	Set robot's position.
Examine	Adjust an appropriate viewpoint for the robot.
Visible	Show or hide the robot.

4.5.4.2. Robot Settings

This dialog is shared with project, controller and robot settings. The left tree view is a hierarchical display of project elements for each controller and robot.

Select the robot node and its settings will display in the right page.



Settings
Controller: SysPPPExt
Robot: T_ROB1

Motion Limits (when robot is not holding products)
Speed (mm/s): 500.00
Acceleration / Deceleration (mm/s²): 5000.00
Rotation Speed (deg/s): 360.00

RAPID
☑ Load and Start RAPID
PmMain.mod
PmUtility.mod
RAPID Editor: %SystemRoot%\system32\notepad.exe

xx110000140

Item	Description
Controller	The name of the controller which the robot belong to.
Robot	The name of the robot.
Speed	The speed of the robot.
Acceleration/Deceleration	The speed can be set to full or low with the right button. The acceleration/deceleration speed of the robot.
Rotation Speed	The rotation speed of the robot.
Load and Start RAPID	If you select this check box, you can press the start button on the FlexPendant and the RAPID module will be loaded to the robot controller. The check box is selected by default.
Add	To add a RAPID module to the project: 1. Click Add. 2. The Load RAPID dialog box appears, and you can select which RAPID module to add. 3. In the Load RAPID dialog box, click OK.

Continues on next page

Continued

Item	Description
Delete	To remove a RAPID module from the project, select a module and click ✖ xx110000090 the button.
Edit	To edit a RAPID module, select a module and click Edit.
Export	To save a module with a new name, select a module and click Export.
Browse	Select software to edit the RAPID. In default, the software is notepad.

4.5.5. Tool

4.5.5.1. Introduction

You can access the context menus for the tool node in the programming browser by right-clicking the tool node.

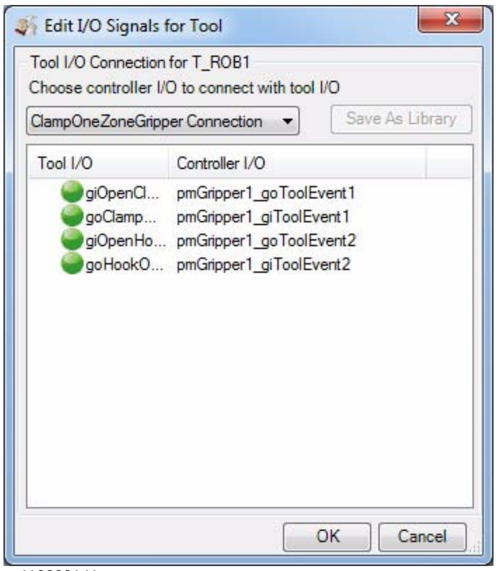
Context Menus from the Tool Node

Context Menu	Description
Edit Tool Signals	Edit signals for this tool. See *Edit Tool Signals on page 152* Error! Reference source not found. for detailed description
Edit Tool Functions…	Edit tool functions for this tool. See *Tool Functions on page 103* for detailed description
Edit Tool Events…	Edit tool events for this tool. See *Tool Events and Scenarios on page 120* for detailed description
Joint Jog	Jog the tool's joint.
Detach from	If the tool has been attached to a robot, click Detach from menu, this tool will detach from the robot. And this node will be deleted either.
Examine	Adjust an appropriate viewpoint for the tool.
Rename	Change the tool's name.
Visible	Show or hide the tool.

4.5.5.2. Edit Tool Signals

In Edit Tool Signals interface, you can view and edit the I/O connection between tool and controller.

A tool SmartComponent contains a set of input and output signals. In order for simulation to run, they should be connected with certain virtual controller I/O.


Edit I/O Signals for Tool
Tool I/O Connection for T_ROB1
Choose controller I/O to connect with tool I/O
ClampOneZoneGripper Connection
Save As Library
OK
Cancel

Tool I/O	Controller I/O
giOpenCl...	pmGripper1_goToolEvent1
goClamp...	pmGripper1_giToolEvent1
giOpenHo...	pmGripper1_goToolEvent2
goHookO...	pmGripper1_giToolEvent2

xx110000141

	Description
I/O Template List	A list of I/O connection templates.An I/O template contains a mapping between tool I/O signals and controller I/O signals.
Tool I/O	The list of tool input and output signals.
Controller I/O	The list of controller input and output signals.You can click on any controller signal name to select another signal. If you have made changes, the I/O template will change to “Customized”, indicating that the mapping is modified by you.
Save as Library	If the current template is “Custormized”, you are then able to save the current mapping into Library, for future reuse.

4.5.6. Feeders

Introduction

You can access the context menus for the feeders node in the programming browser by right-clicking the feeders node.

Context Menus from the Feeders Node

Context Menu	Description
Add Feeder…	It is to add a feeder into the station.See *Add feeder on page 49* for detailed description.

4.5.7. Feeder

4.5.7.1. Introduction

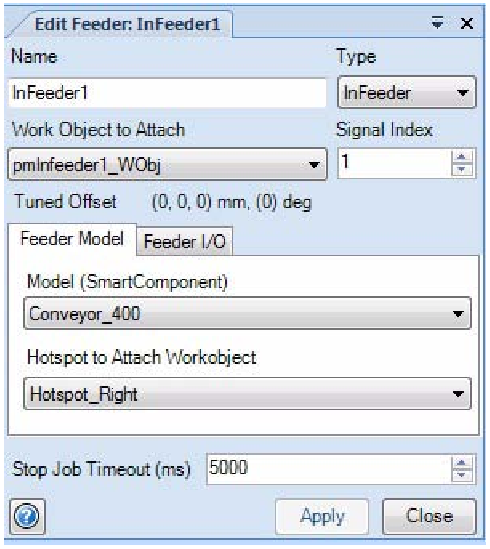
You can access the context menus for the feeder node in the programming browser by right-clicking the feeder node.

Context Menus from the Feeder Node

Context Menu	Description
Edit…	Edit general settings for this feeder. See *Edit Feeder on page 155* for detailed description
Edit Feeder Hotspots…	Edit hotspots for this feeder. See *Create Feeder on page 124* for detailed description
Add Operation Set - Group	Add a group operation set. See *Operation Set on page 163* for detailed description. Available when the feeder does not have any operation set or already has group operation sets.
Add Operation Set – Pattern/Stack	Add a pattern or stack operation set. See *Operation Set on page 163* for detailed description Available when the feeder does not have any operation set or already has pattern/stack operation sets.
Safe Targets…	Define the safe positions that the robot will pass before entering and after leaving the selected feeder. You can define several safe positions to let the robot follow a path when entering and leaving the feeder. Please note the order of the selected safe positions! The last safe position in the list is the last that the robot will pass before reaching the feeder and the first that the robot will pass when leaving the feeder. The last safe position in the list will also define the reference arm and wrist configuration of the robot when operating the feeder. See *Safe Targets on page 159* for detailed description
Robot Path Height…	Define some height attributes that are considered when the robot is moving to, from or over the feeder. The Robot path height settings affect the height of intermediate movements, which means the settings affect the output of the RAPID instruction PmGetPathHeight that is used by MoveInterMid in the PmUtility module. See *Robot Path Height on page 161* for detailed description
Set WorkObject Position…	Set WorkObject position for the feeder. See *Set WorkObject Position on page 162* for detailed description
Examine	Adjust an appropriate viewpoint for the feeder.
Rename	Change the feeder’s name.
Visible	Show or hide the feeder.
Delete	Delete the feeder from station.

4.5.7.2. Edit Feeder

There are two pages to edit general settings for feeder.

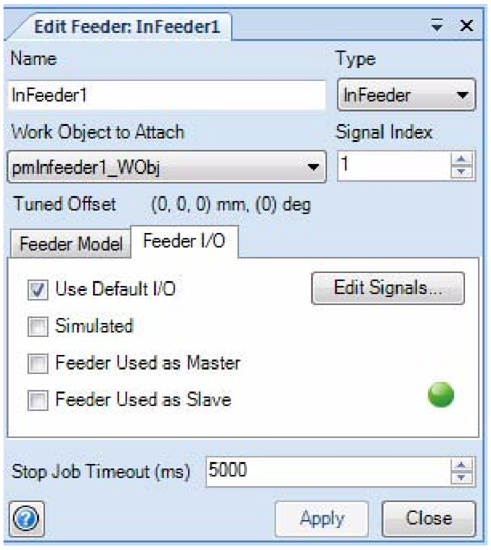


xx110000142

Item	Description
Name	The name of the feeder as it will be shown. **NOTE!** The name will also appear on the PickMaster FlexPendant interface.
Type	The feeder type and the following types are available: ?tInfeeder ?tOutfeeder ?tPalletFeeder ?tSheetFeeder
Work Object to Attach	Specify to which work object this feeder is connected. To change work object, select a work object from the drop-down box. Ten predefined work objects are listed in the work object drop-down box. The work object can be selected also in offline mode.
Signal Index	Specify which default signals that shall be suggested for the feeder when selecting Use Default I/O checkbox. It is combined with the feeder type.
Stop Job Timeout	Define the maximum time a pending position request will be active after a stop job has been started. It is used only by the slave feeders. Should be set to a time that is longer than the time it takes to generate new targets on the feeder. For example, longer than the maximum product infeed time.
Model (SmartComponent)	Select which model (SmartComponent) is the source of the feeder.
Hotspot to Attach Workobject	Select which hotspot is attached with the work object.

Continues on next page

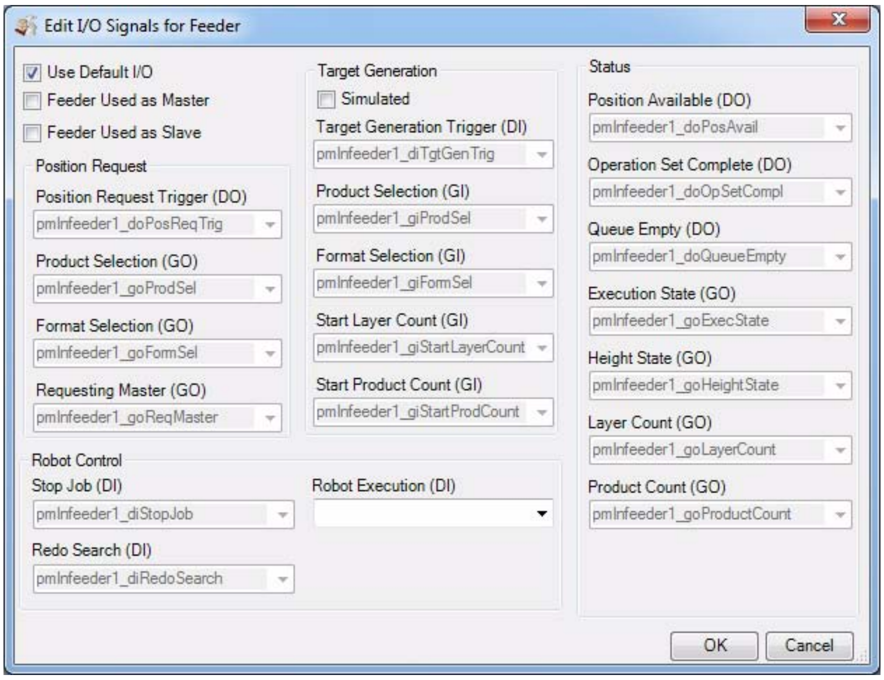
Continued



xx110000143

Item	Description
Use Default I/O	See *Edit Detailed Signals for Feeder on page 157* for detailed description.
Edit Signals…	Edit detailed signals for the feeder.Click this button, a dialog will popup. In this dialog, the signal configuration can be adjusted.See *Edit Detailed Signals for Feeder on page 157* for detailed description.
Simulated	See *Edit Detailed Signals for Feeder on page 157* for detailed description.
Feeder Used as Master	See *Edit Detailed Signals for Feeder on page 157* for detailed description.
Feeder Used as Slave	See *Edit Detailed Signals for Feeder on page 157* for detailed description.
Signal Status	Provide the match information about signals connection between the feeder and the virtual controller. If the signal connection is not valid, the status light will be red, otherwise its color is green.

4.5.7.3. Edit Detailed Signals for Feeder



[✓] Use Default I/O
[ ] Feeder Used as Master
[ ] Feeder Used as Slave

Position Request
Position Request Trigger (DO): pmInfeeder1_doPosReqTrig
Product Selection (GO): pmInfeeder1_goProdSel
Format Selection (GO): pmInfeeder1_goFormSel
Requesting Master (GO): pmInfeeder1_goReqMaster

Target Generation
[ ] Simulated
Target Generation Trigger (DI): pmInfeeder1_diTgtGenTrig
Product Selection (GI): pmInfeeder1_giProdSel
Format Selection (GI): pmInfeeder1_giFormSel
Start Layer Count (GI): pmInfeeder1_giStartLayerCount
Start Product Count (GI): pmInfeeder1_giStartProdCount

Status
Position Available (DO): pmInfeeder1_doPosAvail
Operation Set Complete (DO): pmInfeeder1_doOpSetCompl
Queue Empty (DO): pmInfeeder1_doQueueEmpty
Execution State (GO): pmInfeeder1_goExecState
Height State (GO): pmInfeeder1_goHeightState
Layer Count (GO): pmInfeeder1_goLayerCount
Product Count (GO): pmInfeeder1_goProductCount

Robot Control
Stop Job (DI): pmInfeeder1_diStopJob
Robot Execution (DI):
Redo Search (DI): pmInfeeder1_diRedoSearch

xx110000144

Item	Description
Use Default I/O	When selected, PickMaster will, depending on the selection of Master or Slave, suggest default signals in Position request, Target generation, and Status. The names of the default signals will be based on the selection in feeder type and default signal index. Clear the checkbox if you want to edit the signal selection manually. See Default signals .
Feeder Used as Master	Specify this feeder is used as master. Filter that simplifies setting up a master feeder. When selected, only signals and parameters that can be used by a master feeder are enabled for updates.
Feeder Used as Slave	Specify this feeder is used as slave. Filter that simplifies setting up a slave feeder. When selected, only signals and parameters that can be used by a slave feeder are enabled for updates.
Position Request Group	Define the signals used by the PickMaster process when requesting new operation sets. For a detailed description of the signals, see Basic I/O interface .
Target Generation Group	Define the signals used by a PLC to indicate that new targets are available for picking or placing. For a detailed description of the signals, see Basic I/O interface .

Continues on next page

Continued

Item	Description
Simulated	When the checkbox is selected, the configured target generation signals are not used on the robot controller. Instead, Target generation will be directly cross connected to the output of the Position request signals. Use simulation on all slave feeders to test run a palletizing project without having implemented the communication with a PLC. Palletizing jobs can be started from the FlexPendant or the I/O interface. If only one job, that is, operation set, has been defined for a master feeder, simulation can be used to auto start the job in a continuous mode when the flow has been started.
Status Group	Define different signals to monitor the status of the PickMaster process. For a detailed description of the signals, see Basic I/O interface .
Robot Control Group	Define different signals to affect the PickMaster process. For a detailed description of the signals, see Basic I/O interface .

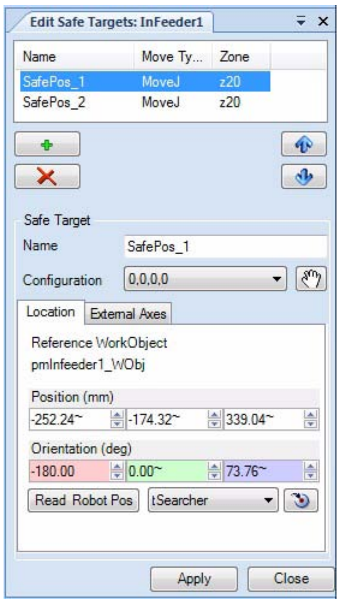
NOTE!

All the above-mentioned signals and their corresponding values are available in the Process signal view on the FlexPendant interface for PickMaster.

NOTE!

When the Target Generation Selection signal is not set on the selected feeder, there can only be one operation set in the feeder. See the *Operation Set on page 163* Configuration for more information.

4.5.7.4. Safe Targets



Edit Safe Targets: InFeeder1

Name	Move Ty...	Zone
SafePos_1	MoveJ	z20
SafePos_2	MoveJ	z20

Safe Target
Name: SafePos_1
Configuration: 0,0,0,0
Location | External Axes
Reference WorkObject: pmInfeeder1_WObj
Position (mm): -252.24~ | -174.32~ | 339.04~
Orientation (deg): -180.00 | 0.00~ | 73.76~
Read Robot Pos | tSearcher
Apply | Close

xx110000145

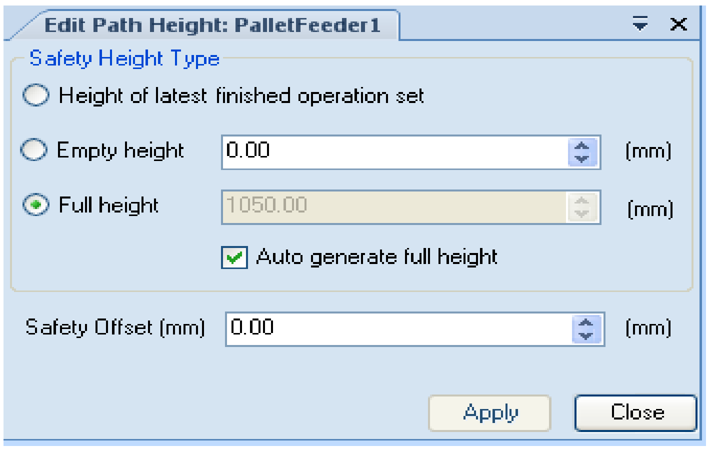
Item	Description
Add a Safe Target	Click button ✚ xx110000088 to add a new safe target in the list.
Delete a Safe Target	Select the safe target and click button ✖ xx110000090 to delete it from the list.
Reorder Safe Targets	Select the safe target and click ⇧ xx110000146 or ⇩ xx110000147 button to reorder the target s in the list.
Change Move Type for a Safe Target	Select the safe position in the list and click the move type value. Select move type in the drop-down list.
Rename a Safe Target	In the Name text box, type a new name for the safe target. In default, a unique name can be generated automatically when a new safe target is created.
Select Axis Configuration	Select one axis configuration for the selected safe target in the combo box. Also click the right button, the available configura-tions for the safe target can be calculated automatically.
Show the Reference WorkObject	The referenced workobject of the safe target is shown in the Location page.

Continues on next page

Continued

Item	**Description**
Edit Safe Target Position	Click in one of Position boxes, and then click the position in the graphics window to transfer the values to the Position boxes.Or type a new value in the Position boxes to specify the position of the safe target.
Edit Safe Target Orientation	Type a new value in the Orientation boxes to specify the orientation of the safe target.
Modify Safe Target Position and Orientation based on Current Robot Pose and Selected TCP	Select one available TCP in the combo box and click Read Robot Pos button, the selected safe target will be updated based on current robot pose.
Check Reach for the Safe Target	Select one safe target in the list and click xx110000148 button to check the robot can reach the safe target based on the selected TCP.
Edit External Axis for the Safe Target	If the robot system has external axis, in the External Axis page, the external axis value can be edited for the selected safe target.

4.5.7.5. Robot Path Height


Edit Path Height: PalletFeeder1

Safety Height Type

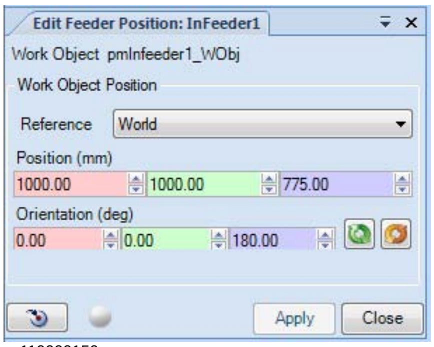
- Height of latest finished operation set
- Empty height: 0.00 (mm)
- [✓ Full height]: 1050.00 (mm)
- [✓] Auto generate full height

Safety Offset (mm): 0.00 (mm)

xx110000149

Item	Description
Default Height	The default height is the expected height of the feeder after an operation set has been completed. For an outfeeder the default height is normally set to Full or Latest. For an infeeder, the default height is normally set to Full, Empty or Latest depending on how the products are fed into the working range of the robot. Latest is the final height of the latest run operation set. The default height can be temporarily updated in runtime, for example set to Empty after unloading a completed stack from the working range of the robot, by using predefined signals in the extended I/O interface or the RAPID function PmSetDefault-Height. It is possible to update the default height in runtime to save cycle time without decreasing the margins for collisions, especially if the project consists of many feeders and flows.
Full Height	Define the full height of the feeder in the work object frame (mm). The Auto generate check box will generate the value as the maximum height of the configured operation sets.
Empty Height	Define the empty height of the feeder in the work object frame (mm).
Safety Offset	An offset that is always added to the expected height of the feeder.

4.5.7.6. Set WorkObject Position



Edit Feeder Position: InFeeder1

Work Object pmInfeeder1_WObj

Work Object Position

Reference: World

Position (mm): 1000.00 | 1000.00 | 775.00

Orientation (deg): 0.00 | 0.00 | 180.00

xx110000150

To...	Do this
Change the Reference Coordinate System	Select the Reference coordinate system you want to use to position the WorkObject used for the feeder.
Edit WorkObject Position	Click in one of Position boxes, and then click the position in the graphics window to transfer the values to the Position boxes.Or type a new value in the Position boxes to specify the position of the WorkObject.
Edit WorkObject Orientation	Type a new value in the Orientation boxes to specify the orientation of the WorkObject. Or click xx110000151 and xx110000152 to adjust the orientation for the WorkObject.
Check Reach for the WorkObject	Click xx110000148 button to check whether the robot can reach the WorkObject based on the active TCP. If it is reachable, the status light will be red, otherwise its color is green.

4.5.8. Operation Set

4.5.8.1. Introduction

An operation set defines how positions are generated for a specific feeder. There are two types of operation sets, the Group Operation Set defining how to pick or place a item group and the Pallet Pattern/Stack Operation Set defining how to pick or place a complete pallet pattern.

Operation sets are created from a feeder and consequently locked to the feeder and the robot with its tool.

Operation sets defined for master feeders are equivalent with palletizing jobs that can be started from a PLC or the FlexPendant interface.

Operation sets defined for slave feeders specify how requested product groups are to be picked or placed.

4.5.8.2. Start the Operation Set Configuration

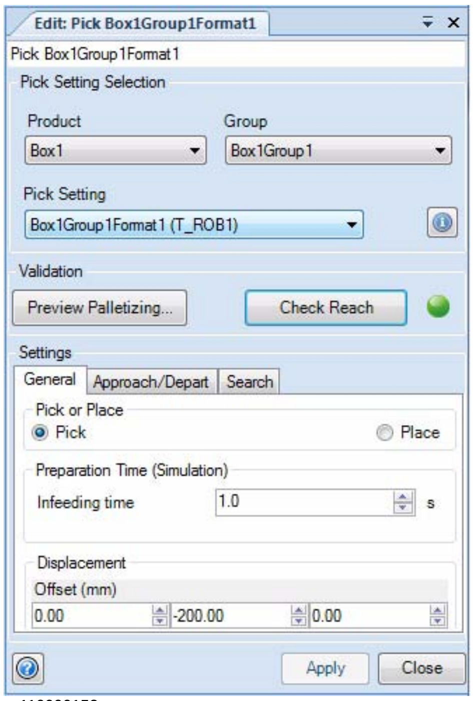
Operation sets are owned by feeders and are accessed from there. For more information about how to add operation sets, see the *Feeder on page 154* Configuration.

You can access the context menus for the operation set node in the programming browser by right-clicking the operation set node.

Context Menus from the Operation Set

Context Menu	Description
Edit…	Open the editing dialog for the operation set.See *Group Operation Set on page 165* and *Pattern/Stack Operation Set on page 167* for detailed information
Locate Pick Setting	Locate the referenced Pick Setting. The Pick Setting interface will be opened and the referenced pick setting will be selected. Available for group and stack operation set.
Locate Pallet Pattern	Locate the referenced pattern. The pattern interface will be opened and the referenced pallet pattern will be selected.Available for pattern operation set.
Locate Stack	Locate the referenced item stack. The Product/Pallet/Sheet interface will be opened and the referenced item stack will be selected.Available for stack operation set.
Preview Palletizing…	Preview each pick (place) target in the operation set.See *Preview Palletizing on page 80* for detailed description.
Rename	Change the operation set's name.
Delete	Delete the Operation Set from the parent feeder.

4.5.8.3. Group Operation Set



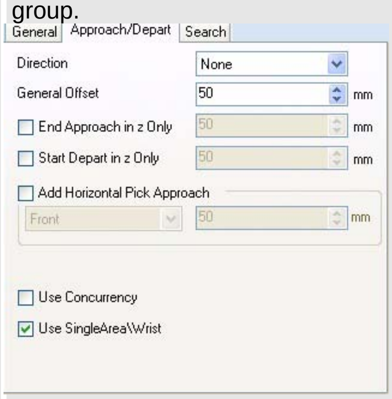
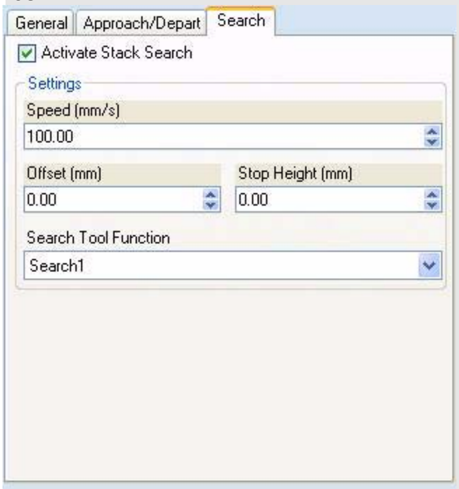
xx110000153

Item	Description
Pick Setting Selection	Select the item, item group and the pick setting for that item group to use. Click on the xx110000154 button to see item count, weight, Pick setting ID, and related input signals from the online PLC when the targets have been generated.
Validation	Use Preview Palletizing function to view each target of the operation set. See *Preview Palletizing on page 80* for detailed information.Use Check Reach to quickly check reach-ability of all the targets, which will be indicated by the status light.
General	Pick or Place: indicates how does the robot access the item group Preparation Time (Simulation): if picking, it describes the time it takes for the item group to show up on the in-feeder before robot goes to pick; if placing, it describes the time it takes for the item group to be moved away on the out-feeder after robot has placed it. A robot can only continue to work on the feeder after the previous items are moved away. Displacement: describes the displacement between the item group and the feeder’s work object. To rotate the frame 90 degrees, click the Flip button, and offsets and angle will be updated to keep chosen alignment. Or you can also click “Drag in 3D View” and use Freehand Move to move the item CAD models directly.

Continues on next page

4.5.8.3. Group Operation Set

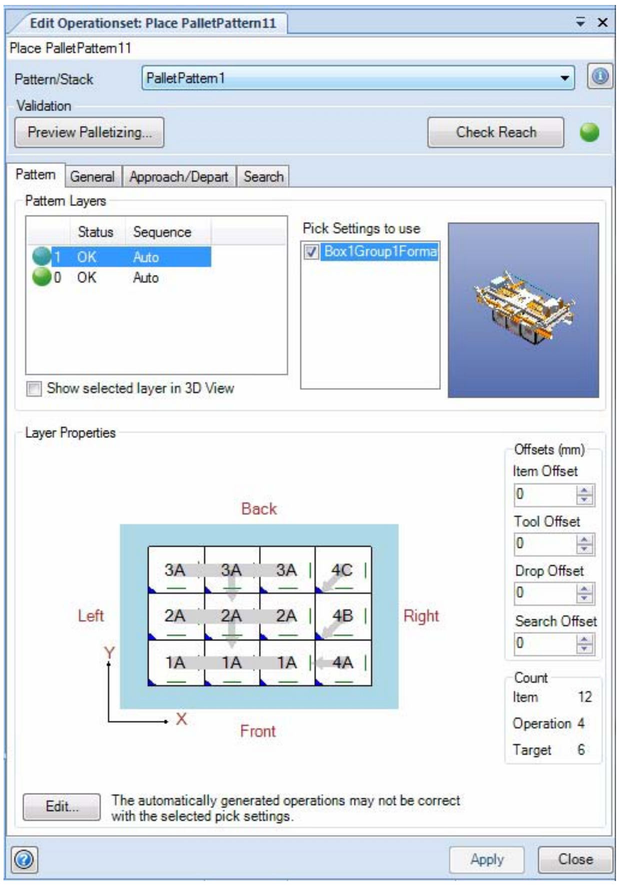
Continued

Item	Description
Approach/Depart	Define the robot path to follow when picking or placing the item group.  Direction: None; General Offset: 50 mm; ☐ End Approach in z Only: 50 mm; ☐ Start Depart in z Only: 50 mm; ☐ Add Horizontal Pick Approach: Front, 50 mm; ☐ Use Concurrency; ☑ Use SingleArea\Wrist xx110000155 Also see Approach for detailed information.
Search	You can enable stack search only if the operation set is Pick.To enable the stack search, select the Activate Stack Search check box.  ☑ Activate Stack Search; Settings: Speed (mm/s): 100.00; Offset (mm): 0.00; Stop Height (mm): 0.00; Search Tool Function: Search1 xx110000156 Also see *Stack search on page 175* for detailed information.

The following table describes the configuration parameters for the group operation set's stack search

Item	Description
Speed	Defines the TCP speed of the robot when it searches for the products.
Offset	Defines an offset of the distance between the expected position of the products, and the starting position of the robot's search movement. The total search offset will also include the product height of the group.
Stop height	Defines the TCP height above the work object where the robot will stop the search movement (if search stop never occurs).
Search Tool Function	Defines which search tool function to use for searching.See *Tool Functions on page 103* on description on search tool function

4.5.8.4. Pattern/Stack Operation Set



Place PalletPattern11

Pattern/Stack: PalletPattern1

Validation

Pattern Layers

	Status	Sequence
1	OK	Auto
0	OK	Auto

Show selected layer in 3D View

Pick Settings to use: ☑ Box1Group1Forma

Layer Properties

3A 3A 3A 4C
2A 2A 2A 4B
1A 1A 1A 4A

Offsets (mm)
Item Offset: 0
Tool Offset: 0
Drop Offset: 0
Search Offset: 0

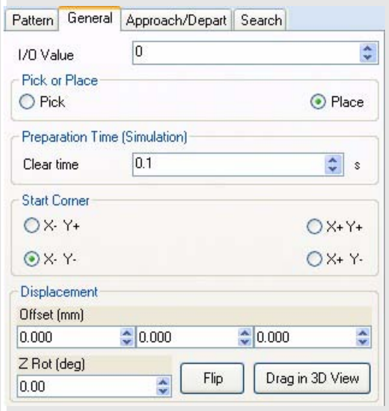
Count
Item 12
Operation 4
Target 6

The automatically generated operations may not be correct with the selected pick settings.

xx110000157

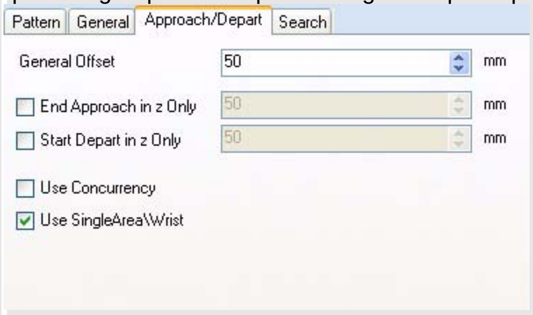
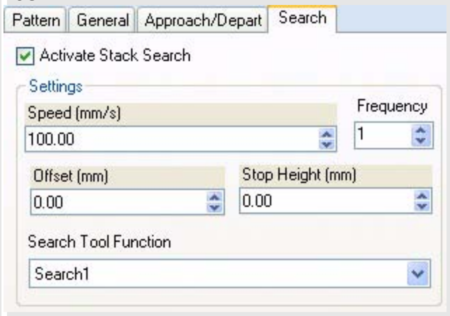
Item	Description
Pattern/Stack	Set the name of the Operation Set. Select the pallet pattern or item stack to use. Click on the xx110000154 button to see item count, weight, total height, and the related product selection signal from the online PLC when the targets have been generated.
Validation	Use Preview Palletizing function to view each target of the operation set. See *Preview Palletizing on page 80* for detailed information. Use Check Reach to quickly check reach-ability of all the targets, which will be indicated by the status light beside the check button and before each layer. Also see *Check Reach on page 177* for detailed information.

Continues on next page

Item	Description
Pattern tab	This section lists and shows information about all layers in the pallet pattern and the pick settings to be used for this pallet pattern. In the Pick Settings to use list box, check the ones that you want to include in the operation set. It is important to select them in such a way that it is possible to complete each layer. **NOTE!** when a pick setting for an item group with more than one column is used, the algorithm to calculate the operations and target positions may be wrong. Warning information will be shown in 2D layout view and user should manually check the sequence and use Preview Palletizing to validate. To edit layer offsets, select the layer and edit the valuesTo edit the operations for a layer, double-click on the layer, or click on the layer and then click Edit.Manually edited layers operations will be specifically marked as “Manual”, otherwise, they will be marked as “Auto”. See *Operation Editor on page 179* for more information.The display shows how the items are grouped together in different operations. For more information, see Display information.
General tab	 I/O Value: 0 Pick or Place: Pick [✓ Place] Preparation Time (Simulation): Clear time 0.1 s Start Corner: X- Y+ ; X+ Y+ ; [✓ X- Y-] ; X+ Y- Displacement: Offset (mm) 0.000 0.000 0.000 Z Rot (deg) 0.00 xx110000158 I/O Value: Set the I/O value of the Operation Set. Also see *I/O interface on page 174* for detailed information. Pick or Place: indicates how does the robot access the item group Preparation Time (Simulation): if picking, it describes the time it takes for the patt ern/stack group to show up on the in-feeder before robot goes to pick; if placing, it describes the time it takes for the pattern/stack to be moved away on the out-feeder after robot has placed it. A robot can only continue to work on the feeder after the previous pattern/stack is moved away. Start Corner: defines the corner of the pallet pattern where the robot should start. Displacement: describes the displacement between the item group and the feeder’s work object. To rotate the frame 90 degrees, click the Flip button, and offsets and angle will be updated to keep chosen alignment. Or you can also click “Drag in 3D View” and use Freehand Move to move the item CAD models directly.

Continues on next page

Continued

Item	Description
Approach/Depart tab	Define the robot path to follow when picking or placing the product group with the pick setting in the pallet pattern.  Pattern | General | Approach/Depart | Search; General Offset: 50 mm; ☐ End Approach in z Only: 50 mm; ☐ Start Depart in z Only: 50 mm; ☐ Use Concurrency; ☑ Use SingleArea\Wrist. xx110000159
Stack search	You can enable stack search only if the operation set is Pick.To enable the stack search, select the Activate Stack Search check box.  Pattern | General | Approach/Depart | Search; ☑ Activate Stack Search; Settings: Speed (mm/s): 100.00; Frequency: 1; Offset (mm): 0.00; Stop Height (mm): 0.00; Search Tool Function: Search1. xx110000160

Display information

The two-dimensional display shows how the items are grouped together in different operations. Each operation for a layer is given a number and every target within an operation is given a letter. As an example, 2A identifies the first target in the second operation and 4B identifies the second target in the fourth operation. Every item is marked with a tag notifying the operation and target number. Every target is also marked with an arrow showing the direction the robot will move in when picking or placing that target. For group operation sets there will only be one operation with one target, but for pallet patterns the items will be automatically grouped into several operations.

In 3D view, it also show an overview of the resulting operation set, including the facing of all the items.

Continues on next page

Offsets for each layer

You can configure adjustments of the general offsets, drop offset and the search offset for individual layers in the Layers Properties section.

NOTE!

This functionality is enabled in RobotWare 5.11. When using older RobotWare releases, then the configured layer offset values will have no effect.

Two offsets are available to adjust the general offset for each layer:

Layer offset	Used when
Item offset	Items are attached to the tool.
Tool offset	No items are attached to the tool.

Use positive values to approach/depart to/from the layer with increased height than defined by the general offset. Use negative values to approach/depart the layer with decreased height. There are no limitations on offsets but the product layer height defines a lower limit in runtime.

NOTE!

If the Item offset is set to a lower negative value than the general offset, there is a risk that carried items will collide with other items already situated in the top layer. Such an offset may however be useful if space or robot reach is limited in specific layers. To avoid collisions, the layout operations may be modified to achieve an optimal pick/place order and item orientations.

Drop offset describes the offset in Z direction above original targets. Usually used for bag palletizing where bags are dropped off from a certain height above the unfinished pallet pattern.

Approach and depart

In this section you define how the robot should move when accessing product groups with pick settings in the operation set. The movement will consist of 3 to 7 positions(and an additional horizontal approach position if Horizonal Pick Approach is activated), depending on the selected settings. The position in which the robot actually picks or places the products is known as the target. Positions preceding the target are called approach and positions following the target are called depart.

Item	Description
Direction	Only available for group operation sets. Defines the direction in the x-y plane of the product group when the robot is approaching a target for placing products or departing a target after picking products.
General offset	Defines the general pick and place approach/depart height. It also defines the movement safety distance to keep between robot held items and their nearby items in a pallet pattern. See the robot movement tables on the following pages for a detailed approach and depart movement description when picking and placing items.

Continues on next page

Continued

Item	Description
End approach in z only	Select this check box when you want the robot to move straight down just before it reaches the target. In the Offset box, type the extra approach distance.
Start depart in z only	Select this check box when you want the robot to move straight up when it leaves the target. In the Offset box, type the extra depart distance.
Add Horizontal Pick Approach	Select this check box when you want to add an additional approach point before the pick target, so that the tool will move to the pick target from side instead of above. In Direction selection box, select the horizontal direction; and in the Offset box, type the horizontal offset.
Concurrency	When the check box is selected, \Conc will be added to the RAPID move instruction. It lets subsequent instructions be executed while the robot is moving to avoid unwanted stop.
Use SingArea \Wrist	This option is selected by default. It is recommended to keep this option available for all 4 axes robots. When the check box is selected: ?tThe wrist axes will never wind up during a long sequence of only linear movements. ?tThe orientation of the tool will deviate between target positions during linear movements when approaching or departing from a work area. The orientation of the tool will never deviate during movements where products are picked or placed.

NOTE!

For more information about the concurrency and SingArea settings, see Technical reference manual - RAPID Instructions, Functions and Data types.

The table below illustrates the resulting robot movement for group operation sets using various settings. Use the following assumptions:

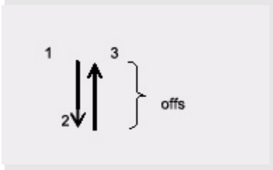
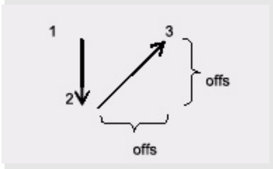
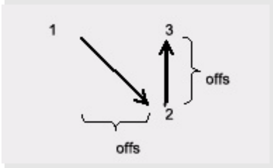
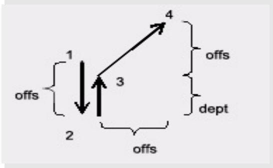
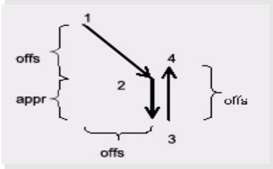
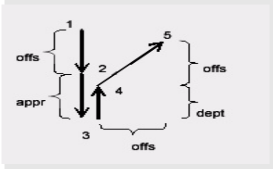
?tGeneral offset is offs.

?tEnd approach in z is appr.

?tStart depart in z is dept.

Continues on next page

Continued

Type	Direction	End approach	Start depart in Z only	Position count in Z only	Movement
any	none	no	no	3	 xx110000161
pick	any	no	no	3	 xx110000162
place	any	no	no	3	 xx110000163
pick	any	no	yes	4	 xx110000164
place	any	yes	no	4	 xx110000165
pick	any	yes	yes	5	 1 5 offs 2 offs 4 appr dept 3 offs xx110000166

Continues on next page

For pallet pattern operation sets the movements are generated a little bit differently. Since multiple picking or placing is supported, the height of the current item must be taken into account. The special cases, when the last item is placed as well as when the first item is picked, are handled separately.The table below illustrates the resulting movement for a pallet pattern operation set with different settings.

Use the following assumptions:

?tGeneral offset is offs.

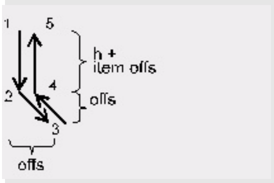
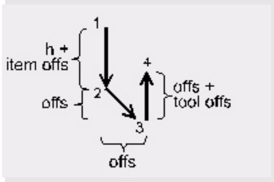
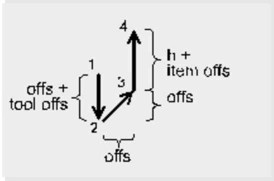
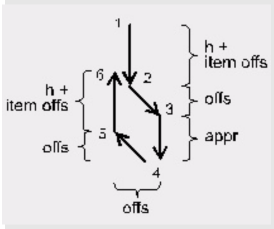
?tEnd approach in z is appr.

?tStart depart in z is dept.

?tLayer item offset is item offs.

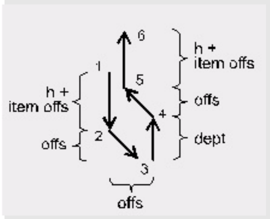
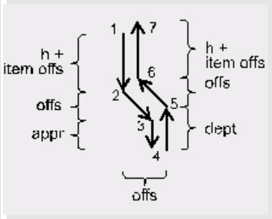
?tLayer tool offset is tool offs.

?tItem height is h.

End approach in Z only	Start depart in Z only	Special case	Position count	Movement
no	no	no *	5	 xx110000167
no	no	last place	4	 1 h + item offs 4 offs + tool offs 2 offs 3 offs xx110000168
no	no	first pick	4	 4 h + item offs 1 3 offs + tool offs offs 2 offs xx110000169
yes	no	no *	6	 1 h + item offs 6 2 h + item offs 3 offs 5 appr offs 4 offs xx110000170

Continues on next page

Continued

End approach in Z only	Start depart in Z only	Special case	Position count	Movement
no	yes	no *	6	 6 h + item offs 1 5 h + item offs offs 4 2 offs dept 3 offs xx110000171
yes	yes	no *	7	 1 7 h + item offs h + item offs 6 offs 2 offs 5 3 appr dept 4 offs xx110000172

```
*)Any pick or place except the first pick or the last place in the
     operation.
```

NOTE!

If using Vacuum tool function to pick the items, the statuses of the tool zones are changed during each operation. When picking items, the zones that are configured for those items are set to activate at the target position. When placing items, the corresponding tool zones are set to deactivate at the target position and then to idle at its last depart position.

For more information about tool events, see *Tool Events and Scenarios on page 120*, *Advanced setting for Item on page 57*, and the *Pick Setting on page 68* Advanced Events.

I/O interface

Whenever the target generation signal is set for a feeder, the positions in an operation set are sent to that feeder. If different operation sets are required in a feeder, the target generation selection signal must be set for corresponding feeder and all operation sets must be given unique I/O values. To select a specific operation set, you first set the correct I/O value for the target generation selection and then set the target generation trigger.

If the target generation selection signal is not configured for the corresponding feeder, only one operation set is allowed for the feeder. When using several operation sets in the same feeder, all operation sets must have unique I/O values. For more information about the target generation signals, see the *Edit Detailed Signals for Feeder on page 157*.

Continues on next page

Stack search

The following table describes the configuration parameters for the pattern/stack operation set's stack search.

Item	Description
Activate stack search	Select the Activate check box. When stack search is activated, you can enter values in the text boxes.
Speed	Defines the TCP speed of the robot when it searches for the next layer. To set speed, in the Speed text box, type the value the robot should use when searching.
Offset	Defines an offset of the distance between the next layer and the starting position of the robot's search movement. The total search offset will also include the product height of the next layer and the current layer's layer search offset (defined in the Layers section).Total search offset = offset + product height + layer search offset. To set offset, in the Offset text box, type the offset value from the assumed to most layer where the robot should start the search procedure.
Stop height	Defines the TCP height above the work object where the robot will stop the search movement (if search stop never occurs). To set stop height, in the Stop height text box, type the offset value from the work area where the robot should stop searching.
Frequency	Defines the frequency to use stack search. The value 1 means every layer, 2 means every second layer, 3 means every third layer, and so on. Set the frequency, in the Frequency box, type or select the layer frequency for which the stack search will be done. 1 means every layer, 2 means every second layer, and so on.
Search Tool Function	Defines which search tool function to use for searching. See *Search on page 114* on *Tool Functions on page 103* for details.

With the stack search function it is possible to configure runtime calibrations of the stack height for a pallet pattern, item stack, or product group. The function is useful to:

?tHandle picking from pallet patterns, or item stacks where the initial number of layers varies from time to time - for example, pallet stations or slip sheet stations.

?tHandle picking from pallet patterns, item stacks, or groups where the height of the layer/layers varies a lot from the configured height.

Search tool

With stack search activated, a special search tool is activated and a search movement is started before the first item in the first layer is picked. The search movement starts an offset from the expected height of the pallet pattern/stack/group and continues until a sensor input indicates that the top layer is reached. If there is no indication from the sensor input, the search movement continues until a stop height is reached. At search stop, the height of the pallet

Continues on next page

pattern/stack/group is updated with the current height of the search tool. Then, movement is started to the approach position of the next pick position. The search tool is deactivated before the movement to the pick position is started.

Pallet pattern

If a pallet pattern is used, picking will continue with lower layers until the pallet is empty. It is also possible to configure new searches for lower layers to improve the picking accuracy. Further it is possible to order a new search from the top of the pallet pattern at any time before the next layer is picked by using an input signal. See Redo search signal in the illustration, in section *Edit Detailed Signals for Feeder on page 157*.

Configuration of stack search

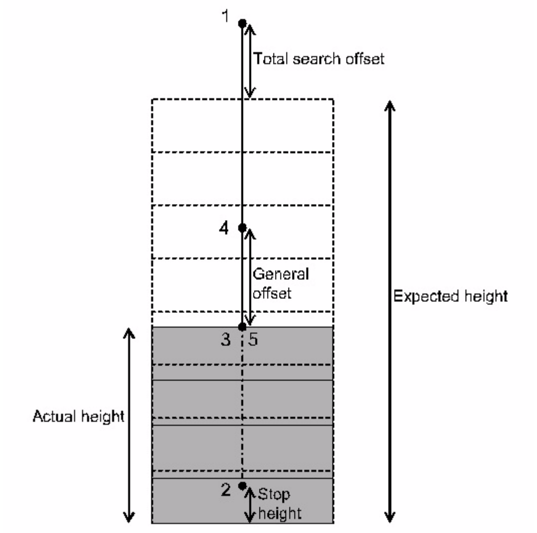
You do the configuration of the stack search in the Pallet Pattern Operation Set Configuration and Group Operation Set configuration dialog boxes.

Configuration of search tool

You do the configuration of the search tool function in the *Search on page 114*window.

Stack search target sequence, example

The following illustration shows an example of a stack search target sequence.



xx110000173

Sequence step	Description
1	Position of search tool at start of search movement. The search tool tcp will be placed above the center point of the stack. Activation of the search tool.
2	Target position for search movement with search tool. In this example never reached. To properly detect an empty stack, stop height should be set lower than half the height of the bottom layer (where the height of the bottom layer is defined as described in PmGetWaHeight).
3	Resulting search target. Top layer is reached with the search tool and search stop occurs. Height of the pallet pattern is updated.

Continues on next page

Continued

Sequence step	Description
4	Approach target of the next pick position with the normal tool configuration. Deactivation of the search tool.
5	Next pick position with the normal tool configuration.

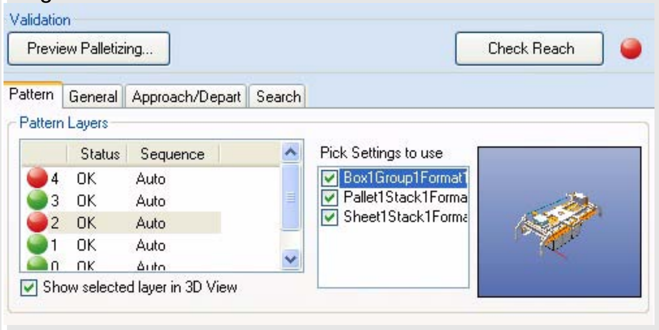
Check Reach

Click *Check Reach* to verify if all positions can be reached by the robot. All robot positions are checked for each layer and each operation, including safe positions, approach positions, pick/place positions, and stack search specific positions. Imported tune data can affect the result.

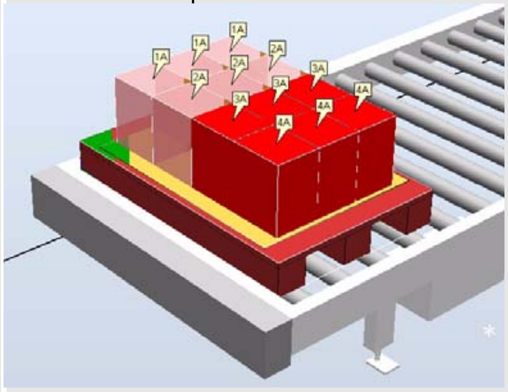
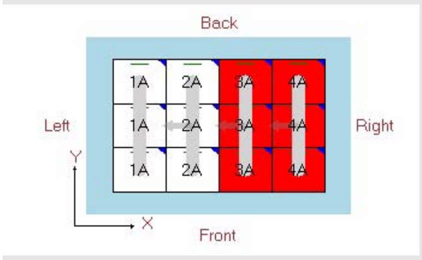
The check reachability function is implemented for Pallet pattern/stack/group operation set and Format operation set.

In current version, intermediate positions are NOT checked.

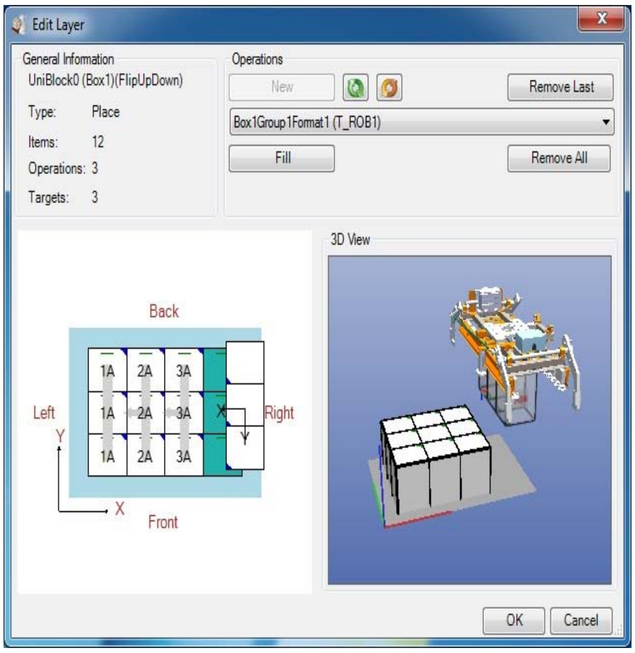
The result of the reachability check function is displayed in different ways

Result	Description
Status light	The status light will be Gray, Green, and Red, indicating reach status of unknown, all reachable, and containing unreachable targets. Each layer also has a status light indicating its containing targets.  Validation Preview Palletizing... Check Reach Pattern General Approach/Depart Search Pattern Layers Status Sequence 4 OK Auto 3 OK Auto 2 OK Auto 1 OK Auto 0 OK Auto Show selected layer in 3D View Pick Settings to use Box1Group1Format1 Pallet1Stack1Forma Sheet1Stack1Forma xx110000174

Continues on next page

Result	Description
3D graphical display	In 3D view, if a layer contains unreachable positions, the related items with these positions are drawn with red color.  xx110000175
Graphic display in Layer, layout display	When selecting a layer containing unreachable positions from the layers list, all items containing unreachable positions will be drawn in red  Back Left Right Front Y X 1A 2A 3A 4A 1A 2A 3A 4A 1A 2A 3A 4A xx110000176

4.5.8.5. Operation Editor



xx110000177

When configuring a pattern operation set, it is automatically given operations, which means that which product groups and pick settings to use and how to access them is defined for each layer. In some situations there might be requirements not fulfilled by auto-generated operations. In such cases you can adjust the operations using the Operation Editor.

A layer normally requires several operations, each using a specific pick setting. The order of the operations is important since this is the order the robot will use to access the product groups and pick settings.

NOTE!

It is only possible to remove the last operation, and new operations can only be added last. This means that to change operations, you may have to remove existing operations first.

To...	Do this
Remove an operation	Click Remove to remove the last existing operation.
Remove all operations	Click Remove All to remove all existing operations.
Add an operation	Click New to add a new operation. The selected pick setting to use for the operation is shown in the Display area.
Select pick setting	Select the pick setting to use for the new operation in the Pick Setting list.
Define product location	Move and rotate the products using the mouse. To define that an item in the group should be placed at a specific location in the layer, make the areas overlap and double-click on the intersection.

Continues on next page

Continued

To...	Do this
Auto generate operations	Click Fill to auto generate the operations for the rest of the layer.
Select access direction	To change the access direction of a placed target, right-click and select direction or select the target and use the mouse wheel to change the direction.

NOTE!

It is recommended that you use a mouse with a wheel when working with the Operation Editor. The mouse wheel can be used to rotate the products and change the access directions.

NOTE!

Layers with edited operations will not be updated in the operation set configuration when input such as start corner is changed. Remove all operations and click Fill to make the layer un-edited.

4.5.9. Flows

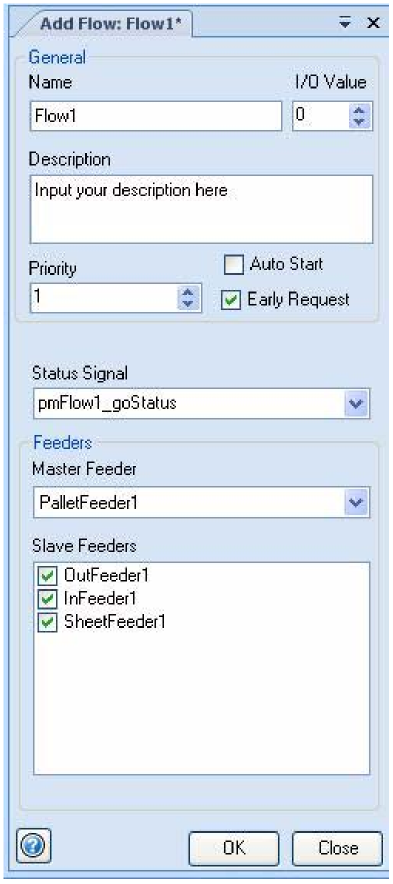
4.5.9.1. Introduction

You can access the context menus for the flows node in the programming browser by right-clicking the flows node.

Context Menus from the Flows

Context Menu	Description
Add Flow…	It is to add a new flow into the robot. See *Add Flow on page 182* for detailed description.

4.5.9.2. Add Flow



xx110000178

Item	Description
Name	The name of the flow as it will be shown. **NOTE!** The name of the flow will also appear on the PickMaster FlexPendant interface.
I/O Value	A unique I/O value for each flow. The I/O value is used when starting, stopping, or performing recovery of a flow from a PLC instead of using the FlexPendant. The I/O value of the GI signal, pmFlow_giSelection, specifies which flow to be started/stopped/recovered.
Description	A brief description of the flow.
Priority	Set individual priority between different flows, used to decide which flow to execute if several are ready to be executed. Valid values are between 0 and 32767(0 is the highest priority).
Auto Start	Select Auto Start check box if the flow should start when the project starts.

Continues on next page

Continued

Item	Description
Early Request	Select Early request check box if the next request for new products to the slave feeder (used as a slave) is made before the operation is executed on the master feeder (used as a master).If this option is not selected, the next request for new products to the slave feeders is made after the operation is executed on the master feeder. **NOTE!** In practice, when palletizing and the outfeeder feeder is the master, the next operation is requested on the infeeder right after the last products are picked up from it. The consequence can be that if something fails on the way to the feeder in which to place the products, it might not be possible to redo the interrupted operation since the next product is already requested in the infeeder, which may not be the same as the interrupted one.
Status Signal	A unique GO status signal for each flow. The status signal is used to monitor the runtime status of the flow.
Master Feeder	Select the feeder on which palletizing jobs are started. Often the feeder where the pallet pattern is built is used as a master. A feeder used as a master in a flow cannot be involved in any other flows.
Slave Feeders	Select the feeder s used as slaves (slave feeders). These are the feeder s that will serve the master feeder with products. The slave feeder s are often pallet stacks, slipsheet stacks, and conveyor feeders. Slave feeders can be shared by multiple flows.

4.5.10. Flow

4.5.10.1. Introduction

You can access the context menus for the flow node in the programming browser by right-clicking the flow node.

Context Menus from the Flow

Context Menu	Description
Edit…	Edit the selected flow configuration.See *Add Flow on page 182* for detailed description.
Add Job…	Starts the Add Job wizard.See *Add Job on page 73* for detailed description.
Locate Master Feeder	Locate the master feeder of this flow. The feeder node will be selected in the Programming browser.
Rename	Change the flow’s name.
Delete	Delete the flow from the robot.

4.5.11. Job

4.5.11.1. Introduction

You can access the context menus for the job node in the programming browser by right-clicking the job node.

Context Menus from the Job

Context Menu	Description
Locate Original Operation set	Locate the referenced operation set node in Programming browser.
Simulate Job	Starts simulation but only running the selected job.
Delete	Delete the referenced operation set of the master feeder.

4.5.11.1. Introduction

5 RAPID program

5.1. Introduction

Structure of this chapter

This chapter describes the RAPID program module templates and system modules. Program examples with detailed descriptions are also included. Windows and other parts of the user interface are described with regard to their content and how they are accessed.

5.2 Overview

5.2.1. Relationship between RAPID execution and PickMaster project

Overview

The RAPID program templates for a robot executes a pick-and-place cycle for one of the configured flows in every loop. The selection of flow is made in priority order among the flows which currently are ready to execute, that is, having targets generated for the next operation on both the infeeder and the outfeeder. The robot movements and the I/O events on the infeeder and outfeeder are performed according to the configured operation sets and formats.

Intermediate positions

The RAPID program templates inludes functionality to generate safe intermediate positions for the movements between infeeders, outfeeders and the home position.

Operation, target, action, event and product

To describe all movements in an Operation Set, their properties are divided into Operations, Targets and Actions, which are retrieved by the instructions `PmGetOperation`, `PmGetTarget` and `PmGetTgtAction`.

Operation

Everything that is done by the robot in one visit to work area. This includes multi-pick or multi-place of several products.

One Operation contains one or more Targets.

For details, see *pm_operationdata - PickMaster operation data on page 345*.

Target

The final position of every pick or place of one or more products. Also contains the work object and tool used for the whole path to and from the target positions.

One Target contains one or more Actions and product data.

For details, see *pm_targetdata - PickMaster target data on page 359*.

Action

One path segment on the way to and from a target. Every action is realized as a move instruction (`TriggL/MoveL`) in RAPID.

One Action contains one or more Events.

For details, see *pm_actiondata - PickMaster action data on page 329*.

Event

An event that occurs on the path. It can be a change of a signal value or an acknowledgement that a certain task has been performed and is realized through trigg data using `TriggL` in RAPID.

Product

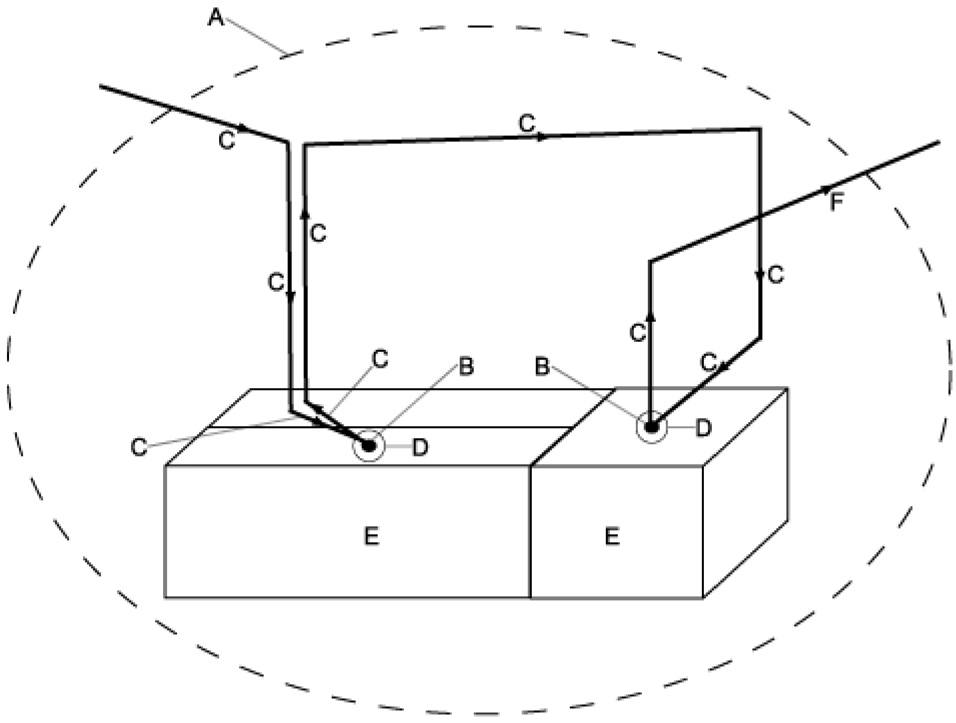
One or more Product(s) that is/are handled (picked/placed) by the robot at each Target.

Continues on next page

Continued

Illustration

A typical place operation of two products at two different angles will be realized as:



xx0600003002

A	Operation
B	Targets
C	Actions
D	Event (in this case *Turn off the vacuum*)
E	Products
F	To next InterMid position

5.2.2. RAPID modules overview

Overview

To run a PickMaster project you need RAPID program modules and system modules, which are described in this section.

RAPID template modules

The PickMaster installation includes the following two RAPID template modules:

?t*PmMain*, which is a program module that contains basic code to execute the operations in different work areas.

?t*PmUtility*, which is a program module that contains home positions and intermediate positions.

The RobotWare option *Prepared for PickMaster*, together with the sub-option *PickMaster 5*, includes the following two RAPID template modules:

?t*PmProjMgr*, which is a program module that contains basic code to execute the commands from PickMaster I/O interface.

?t*PmProjServer*, which is a program module that contains basic code to execute the commands from PickMaster I/O interface, executed in a semi-static RAPID task.

System modules

The RobotWare option *Prepared for PickMaster*, together with the sub-option *PickMaster 5*, includes the following three installed system modules:

?t*pmrcUser*, which is a system module that contains tool and work object declarations and traps for checking I/O values.

?t*pmrcSys*, which is a system module that contains open non-view procedures mainly used to set all modal data used in the moves.

?t*pmrcBase*, which is a system module that contains encrypted non-view procedures and variables used in the process.

5.3 Program module templates

5.3.1. PmMain module

Overview

This section describes the routines and variables in the *PmMain* module. The module contains the main procedure for the PickMaster RAPID execution, and it is where the program starts the execution.

This section describes the following procedures:

?tMain

?tOperateSequence

?tOperate

Procedure `Main`

Usage

This is the main procedure of the template and where the execution starts.

Description

In this routine, the initiation is done and the move to the home position.

Program code

```
PROC Main()
  IF FirstMainLoop THEN
     MoveHomePos;
     FirstMainLoop:=FALSE;
  ENDIF
  PmWaitProjStart;
  OperateSequence;
ENDPROC
```

Related information

Main routine in *Technical reference manual - RAPID kernel*.

Procedure `OperateSequence` on page 191.

PmWaitProjStart - Wait for any active project on page 308.

PmGetWaByWobj - Get a reference to a work area using a work object data on page 292.

Procedure `OperateSequence`

Usage

This routine performs one cycle sequence.

Continues on next page

Description

`OperateSequence` executes one cycle beginning with fetching a flow that is ready to be executed. Then the robot operates first on the infeeder and then on the outfeeder.

The default error handler handles raise errors from *PmSearchAdjust* when running stack search.

The `PM_ERR_PALLET_REDUCED` error, indicating that a number of layers was removed since the detected stack height was lower than expected, is recovered by default using `RETRY`. Next operation, which already is adjusted after the stack search, is fetched and executed. As a result, the object will be picked at the correct height with the correct operation.

The `PM_ERR_PALLET_EMPTY` error, indicating that objects are missing on the work area, is not recovered by default. To recover `PM_ERR_PALLET_EMPTY`, follow these directions:

1. Move back the robot in the negative search direction.
2. Eliminate the cause of the error, for example, fill upp with new pallets.
3. The position request DO signal is set after an empty stack is detected. Generate a new stack by setting the target generation signals according to the request.
4. Set variable `MultiOperation` to `TRUE` to avoid moving to an intermediate position.
5. Recover the error in the error handler using `RETRY`. As a result the robot will search the new stack from the top.

Program code

```
PROC OperateSequence()
  PmGetFlow waInFeeder, waOutFeeder;
  Operate waInFeeder;
  Operate waOutFeeder;
ERROR
  TEST ERRNO
     CASE PM_ERR_PALLET_REDUCED:
        ! Number of remaining layers on pallet was updated after
             stack search.
        ! Operate the same work area again to access the new current
             layer.
        MultiOperation:=TRUE;
        RETRY;
     CASE PM_ERR_PALLET_EMPTY:
        ! The pallet stack was found empty during stack search.
        MultiOperation:=FALSE;
        RAISE;
  ENDTEST
ENDPROC
```

Related information

Procedure `Operate` on page 192.

Procedure `Operate`

Usage

This procedure is used to execute an operation.

Continues on next page

Description

The procedure loops through all targets in an operation and through all actions in every target. It calls the `PmCalcArmConf`, which helps setting the arm configuration on every target. Before the very first target in the operation is executed, the robot will move to an intermediate position.

The default error handler handles raise errors from `PmSearchAdjust` when running stack search. The errors are raised to the calling routine, `OperateSequence`.

Arguments

WorkArea

Datatype: `pm_wadescr`

Contains a reference to a work area.

Program code

```
PROC Operate(VAR pm_wadescr WorkArea)
  VAR pm_operationdata Op;
  VAR pm_targetdata Tgt;
  VAR pm_actiondata Act;
  VAR bool FirstTgtInOp:=TRUE;

  PmGetOperation WorkArea, Op;
  WHILE PmGetTarget(WorkArea \OpHandle:=Op.OpHandle, Tgt) DO
    WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) DO
      PmCalcArmConf Act.RobTgt,Tgt.TargetTool,Tgt.TargetWobj
           \cf6\MaxAngle:=MaxToolAngle\MinAngle:=MinToolAngle;
      IF FirstTgtInOp AND (NOT MultiOperation)THEN
        MoveInterMid WorkArea,Tgt,Act,PmSafetyOffsetZ
             \MaxAngle:=MaxToolAngle\MinAngle:=MinToolAngle;
      ENDIF
      DoAction WorkArea,Tgt,Act;
      SetLastPos WorkArea,Tgt,Act;
    ENDWHILE
  ENDWHILE
  MultiOperation:=FALSE;

ERROR
  TEST ERRNO
    CASE PM_ERR_PALLET_REDUCED:
      RAISE;
    CASE PM_ERR_PALLET_EMPTY:
      RAISE;
  ENDTEST
ENDPROC
```

Continues on next page

Related information

Procedure `PmDoAction` on page 213

Procedure `SetLastPos` on page 197.

Procedure `MoveInterMid` on page 198.

pm_wadescr - PickMaster work area reference on page 363.

pm_operationdata - PickMaster operation data on page 345.

pm_targetdata - PickMaster target data on page 359.

pm_actiondata - PickMaster action data on page 329.

PmGetOperation - Get operation from a work area on page 288.

PmCalcArmConf - Calculates the arm configuration on page 280.

PmGetTarget - Get target on page 320.

PmGetTgtAction - Get target action on page 322.

PmSearchAdjust - Adjust number of remaining layers on page 295

Variables `waInFeeder1` and `waOutFeeder1`

Usage

The variables are used as work area descriptors for one infeeder and one outfeeder.

Description

The descriptors are handled to access the work areas when retrieving operations, targets and actions.

Program code

```
VAR pm_wadescr waInFeeder1;
VAR pm_wadescr waOutFeeder1;

PmGetFlow waInFeeder, waOutFeeder;
Operate waInFeeder1;
Operate waOutFeeder1;
```

Related information

pm_wadescr - PickMaster work area reference on page 363.

Constants `MaxToolAngle` and `MinToolAngle`

Usage

The constants are used to set the maximum and minimum allowed angles for the tool.

Description

The angle limitation is used to set the maximum and minimum angle on the tool axis 6. This may be changed because of the limitations in hoses and wires.

Program code

```
PmCalcArmConf Tgt.RobTgtPoint, Tgt.TargetTool, Tgt.TargetWobj
    \cf6 \MaxAngle:=MaxToolAngle \MinAngle:=MinToolAngle;
```

Continues on next page

Continued

Related information

pm_wadescr - PickMaster work area reference on page 363.

PmCalcArmConf - Calculates the arm configuration on page 280.

5.3.2. PmUtility module

Overview

This section describes the routines and variables in the *PmUtility* module. The module contains support for the home and intermediate position.

Procedure `MoveHomePos`

Usage

This procedure is used to move the robot to the home position.

Description

This routine uses `MoveInterMid` to move the robot to a well-defined home position with a safe path height when passing intermediate work areas on the way. The routine will wait for the project to be started before the movement is executed. The default installed work area `PM_HOME` is used as the work area to go to. Finally, the home position is set as the last work area, thus making it the starting point for the next intermediate movement which also will get a safe path height.

Continues on next page

Continued

Program code

```
PROC MoveHomePos()
  CONST num RetractDist:=50;
  VAR pm_targetdata Tgt;
  VAR pm_actiondata Act;
  VAR pm_wadescr HomeWorkArea;

  ! Get the weight from current tool and frame from tool0
  PmLastTool:=CTool();
  PmLastTool.tframe:=tool0.tframe;
  PmLastWobj:=CWObj();
  PmLastRobTgt:=CRobT(\Tool:=PmLastTool\Wobj:=PmLastWobj);

  Act.Speed:=v500;
  Act.RobTgt:=CalcRobT(HomePos,tool0);
  Tgt.TargetTool:=PmLastTool;
  Tgt.TargetWobj:=pm_home_WObj;

  PmLastRobTgt.trans.z:=PmLastRobTgt.trans.z+RetractDist;
  MoveLPmLastRobTgt,Act.Speed,fine,PmLastTool\Wobj:=PmLastWobj;

  PmWaitProjStart;
  PmGetWaByWobj pm_home_WObj,HomeWorkArea;
  MoveInterMid HomeWorkArea,Tgt,Act,
       PmSafetyHeight\MoveToEndPoint;
  PmLastRobTgt:=Act.RobTgt;
  PmLastWobj:=Tgt.TargetWobj;
  PmLastTool:=Tgt.TargetTool;
  PmSetLastWa HomeWorkArea;
ENDPROC
```

Related information

PmWaitProjStart - Wait for any active project on page 308.

PmGetWaByWobj - Get a reference to a work area using a work object data on page 292.

PmSetLastWa - Set last used work area on page 297.

Procedure `SetLastPos`

Usage

This procedure is used to store the last position, tool, work object, and work area.

Description

The stored position, tool, work object, and work area are used when calculating the intermediate position.

Continues on next page

Arguments

WorkArea

Datatype: `pm_wadescr`

Last work area that was used.

Tgt

Datatype: `pm_targetdata`

Last target data that was used.

Act

Datatype: `pm_actiondata`

Last action data that was used.

Program code

```
PROC SetLastPos(VAR pm_wadescr WorkArea, VAR pm_targetdata Tgt,
     VAR pm_actiondata Act)
  VAR robtarget temp;

  temp:=LastRobTgt;
  LastRobTgt:=Act.RobTgt;
  LastWobj:=Tgt.TargetWobj;
  LastTool:=Tgt.TargetTool;
  PmSetLastWa WorkArea;
  IF Act.ArmConfMon = FALSE THEN
     LastRobTgt.robconf:=temp.robconf;
  ENDIF
ENDPROC
```

Related information

pm_actiondata - PickMaster action data on page 329.

pm_targetdata - PickMaster target data on page 359.

pm_wadescr - PickMaster work area reference on page 363.

PmSetLastWa - Set last used work area on page 297.

Procedure `MoveInterMid`

Usage

This procedure is used to move the robot from a starting point (for example another work area or the home position) towards a new operation on the next work area with a safe path height when passing intermediate work areas on the way from the starting point.

Description

This procedure uses the last stored position (from the `SetLastPos` procedure) and a new operation on the next work area to calculate three consecutive intermediate positions by using the `PmCalcIntermid` routine. `PmGetPathHeight` is used to find the minimum height for a safe travel towards the work area.

Continues on next page

Continued

Arguments

WorkArea

Datatype: `pm_wadescr`

Work area to go to.

Tgt

Datatype: `pm_targetdata`

The next target to go to.

Act

Datatype: `pm_actiondata`

The next action to perform.

SafetyOffsetZ

Datatype: `num`

An additional safety offset that is added to the minimum path height.

MaxAngle

Datatype: `num`

The maximum allowed tool angle.

MinAngle

Datatype: `num`

The minimum allowed tool angle.

MoveToEndPoint

Datatype: `switch`

Finish with zone or fine point.

Continues on next page

Program code

```
PROC MoveInterMid(VAR pm_wadescr WorkArea,VAR pm_targetdata
     Tgt,VAR pm_actiondata Act,num SafetyOffsetZ,\num MaxAngle,
     \num MinAngle,\switch MoveToEndPoint)
  CONST num IntermidPart1:=0.1;
  CONST num IntermidPart2:=0.5;
  CONST num IntermidPart3:=0.9;
  VAR robtarget InterMid1;
  VAR robtarget InterMid2;
  VAR robtarget InterMid3;
  VAR num MinZ;
  VAR pm_wadescr LastWorkArea;
  PmGetLastWa LastWorkArea;
  ! Calculate MinZ. The z value of the tool and product is not
       considered in the calculation of min z in PmCalcIntermid.
  IF Tgt.NumOfAppProds=0 THEN
    ! MinZ without product in tool
    MinZ:=PmGetPathHeight(LastWorkArea,WorkArea\UseSafePosition)
         +Tgt.TargetTool.tframe.trans.z+SafetyOffsetZ;
  ELSE
    ! MinZ with product in tool
    MinZ:=PmGetPathHeight(LastWorkArea,WorkArea\UseSafePosition)
         +Tgt.TargetTool.tframe.trans.z+Tgt.ProductHeight+Safety
         OffsetZ;
```

Continues on next page

```
    ENDIF
    ConfJ\On;
    ! Use the frame from tool0 and the load from target tool
    TempTool:=Tgt.TargetTool;
    TempTool.tframe:=tool0.tframe;
    ! Travel distance: 10%
    InterMid1:=PmCalcIntermid(PmLastRobTgt,PmLastTool,PmLastWobj,
         Act.RobTgt,Tgt.TargetTool,Tgt.TargetWobj,IntermidPart1
         \MaxAngle?MaxAngle\MinAngle?MinAngle\AngleLimAx6
         \MinZ:=MinZ\FromWa:=LastWorkArea\ToWa:=WorkArea);
    MoveJ\Conc,InterMid1,Act.Speed,z200,TempTool\Wobj:=wobj0;
    ! Calculate intermediate targets two and three
    InterMid2:=PmCalcIntermid(PmLastRobTgt,PmLastTool,PmLastWobj,
         Act.RobTgt,Tgt.TargetTool,Tgt.TargetWobj,IntermidPart2
         \MaxAngle?MaxAngle\MinAngle?MinAngle\AngleLimAx6
         \MinZ:=MinZ\FromWa:=LastWorkArea\ToWa:=WorkArea);
    InterMid3:=PmCalcIntermid(PmLastRobTgt,PmLastTool,PmLastWobj,
         Act.RobTgt,Tgt.TargetTool,Tgt.TargetWobj,IntermidPart3
         \MaxAngle?MaxAngle\MinAngle?MinAngle\AngleLimAx6
         \MinZ:=MinZ\FromWa:=LastWorkArea\ToWa:=WorkArea);
    ! Travel distance: 50%
    ! No concurrency shall be used on the second internediate
         movement to avoid the limitation of having too many
         consecutive movements with concurrency.
    MoveJ InterMid2,Act.Speed,z200,TempTool\Wobj:=wobj0;
    ! Travel distance: 90%
    IF Present(MoveToEndPoint) THEN
      MoveJ\Conc,Act.RobTgt,Act.Speed,fine,TempTool\Wobj:=wobj0;
    ELSE
      MoveJ\Conc,InterMid3,Act.Speed,z200,TempTool\Wobj:=wobj0;
    ENDIF
  ENDPROC
```

Related information

pm_actiondata - PickMaster action data on page 329.

pm_targetdata - PickMaster target data on page 359.

pm_wadescr - PickMaster work area reference on page 363.

PmGetLastWa - Get last used work area on page 287.

PmGetPathHeight - Get a safe path height for an intermediate movement on page 315.

PmCalcArmConf - Calculates the arm configuration on page 280.

Variable `HomePos`

Usage

The variable is used to set the home position of the robot.

Description

The home position must be modified for custom purposes.

Continues on next page

Program code

```
LOCAL PERS jointtarget
     HomePos:=[[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
     09]];
...
MoveAbsJ HomePos,HomeSpeed,fine,TempTool;
```

Related information

Technical reference manual - RAPID Instructions, Functions and Data types, section *robtarget - Position data*.

5.3.3. PmProjMgr module

Overview

The *PmProjMgr* module can be used if needed. It is prepared to be used in PickMaster I/O interface for starting a project and loading the modules needed for the main palletizing loop. The I/O signals used in this module are the same as in the configuration that comes with the installation of PickMaster. This module is compatible with the modules *PmMain* and *PmUtility*.

This section describes:

?tProcedure `Main`

?tTrap `TrapProjectStopped`

Procedure `Main`

This section describes the main routine in the *PmProjMgr* module.

Usage

This is where the program starts the execution if the PickMaster I/O interface is used.

The error handler is simple but it is prepared so it can easily be complemented with more sophisticated error handling depending on your needs.

Description

PmProjMgr executes the following:

?tSetting up a TRAP that supervises the stop project signal. The RAPID execution is interrupted and continues from Main.

?tWaiting for project start signal.

?tReading project selection signal. The mapping between project and its signal value must have been transferred.

?tStarting selected project.

?tSetting current project signal to the value of the selected project.

?tLoading the modules in the project. The modules are only loaded if it is specified in the configuration and if they are not already loaded.

?tExecuting the main loop until the project is stopped. The main routine that is called is the same as if the project is started without the I/O interface.

Program code

```
PROC main()
  VAR pm_projectinfo ProjInfo;

  IF FirstProjMgrLoop THEN
    FirstProjMgrLoop:=FALSE;
    ! Project is stopping
    IDelete pmIntProjectStopping;
    CONNECT pmIntProjectStopping WITH TrapProjectStopped;
    ISignalDI\SingleSafe,pmProject_diStop,1,
         pmIntProjectStopping;
```

Continues on next page

```
ENDIF

! Activate the main loop
StartLoadRun:=TRUE;

IF PM_PROJECT_STATUS=PM_PROJECT_STOPPED OR
     PM_PROJECT_STATUS=PM_PROJECT_STOPPING OR
     PM_PROJECT_STATUS=PM_PROJECT_ERROR THEN
  IF PM_PROJECT_STATUS=PM_PROJECT_STOPPING THEN
    WaitUntil PM_PROJECT_STATUS=PM_PROJECT_STOPPED OR
         PM_PROJECT_STATUS=PM_PROJECT_ERROR;
ENDIF

! Wait for start project order from PLC
WaitDI pmProject_diStart,1;
! Check which project to be started
ProjectSelection:=pmProject_giSelection;
! Get info from select project
PmGetProjectInfo ProjectSelection,ProjInfo;
ProjectInfo:=ProjInfo;
! Start the selected project
IF StartLoadRun THEN
  PmStartProj ProjectInfo.Name\Signal:=pmProject_goStatus;
  SetGO pmProject_goCurrent, ProjectSelection;
ENDIF
ELSE ! STARTING OR RUNNING
  ! Wait for project to be running
  PmWaitProjStart;
ENDIF

! Load all program modules for the task
IF StartLoadRun THEN
  LoadAllModulesInTask ProjectInfo;
ENDIF

WHILE StartLoadRun DO
  ! Execute the main routine in the selected project
  %"PmMain:Main"%;
  IF PM_PROJECT_STATUS=PM_PROJECT_STOPPED OR
       PM_PROJECT_STATUS=PM_PROJECT_STOPPING OR
       PM_PROJECT_STATUS=PM_PROJECT_ERROR THEN
    StartLoadRun:=FALSE;
  ENDIF
ENDWHILE
ERROR
  IF ERRNO=PM_ERR_NO_TASK THEN
    ! ProjectInfo has no task configured for current task
    StartLoadRun:=FALSE;
```

Continues on next page

```
    ELSEIF ERRNO=PM_ERR_PROJ_NOT_FOUND THEN
      ! There is no project mapped to the selection value
      StartLoadRun:=FALSE;
      WaitDI pmProject_diStart,0;
      TRYNEXT;
    ELSEIF ERRNO=ERR_REFUNKPRC THEN
      ! There is no main routine in the loaded modules
      StartLoadRun:=FALSE;
    ENDIF
ENDPROC
```

Related information

Main routine in *Technical reference manual - RAPID kernel*.

Late binding in *Technical reference manual - RAPID kernel*.

PmWaitProjStart - Wait for any active project on page 308.

PmStartProj - Start a PickMaster project on page 304.

Trap `TrapProjectStopped`

This section describes the trap that is called when the project stopped signal is pulsed.

Usage

The trap will prevent that RAPID execution continues after an ordered stop of project. The execution continues at main but the loaded modules are not unloaded. Starting a new project will fail to load the new modules. This trap is not executed if the recommended stop sequence is followed.

Description

TrapProjectStoped executes the following:

?tStopping robot movement.

?tClearing the robot path.

?tResetting stop move state.

?tContinuing execution from main.

Program code

```
TRAP TrapProjectStopped
  StopMove\Quick;
  ClearPath;
  StartMove;
  FirstProjMgrLoop:=TRUE;
  WaitTime 2;
  ExitCycle;
ENDTRAP
```

Related information

Technical reference manual - RAPID kernel.

5.3.4. PmProjServer module

Overview

The *PmProjServer* module is used in the semi-static RAPID task `PM_PROJ_SUPERV`. It is prepared to be used in PickMaster I/O interface for starting and stopping flows. The I/O signals used in this module are the same as in the configuration that comes with the installation of PickMaster.

The I/O signals are mapped to alias signals to prevent errors if the signals are not configured in the controller. A warning is generated in the error log at each warm start of the controller if the signals are not found.

Procedure `Main`

This section describes the main routine in the *PmProjServer* module.

Usage

This is where the program starts the execution.

Description

Main routine executes the following:

?tConnecting all alias signals with the configured signals.

?tWaiting for a project to start.

?tConnecting traps to project stop, flow start, and flow stop traps.

?tWaiting for the project to stop.

?tDisconnecting all traps.

Program code

```
PROC main()
  VAR bool SignalsExist:=TRUE;

  ! Connect all alias signals with the configured signals
  SignalsExist:=ConnectAliasSignals();
  WHILE SignalsExist=FALSE DO
     ! Loop forever
     WaitTime 1000;
```

Continues on next page

```
  ENDWHILE

  WHILE TRUE DO
     ! Wait for project to be running.
     PmWaitProjStart;

     ! Connect all traps with its interrupts
     ConnectTraps;

     ! Wait for stop project order
     WaitUntil PM_PROJECT_STATUS=PM_PROJECT_STOPPED;
     SetGO alias_goCurrentProject, 0;

     ! Disconnect all traps from its interrupts
     DeleteTraps;
  ENDWHILE
ENDPROC
```

Trap `TrapSetRecoverAction`

This section describes the trap that executes when the set recover action signal has been pulsed.

Description

Before a flow that is in error state can be started a recover action has to be set. If using the I/O interface a flow, work area, and a recover action must have been set before the set recover action is pulsed. An event log messages is generated with information about conditions for a successful flow restart.

Program code

```
TRAP TrapSetRecoverAction
  VAR pm_flowinfo FlowInfo;
  VAR num FlowSelection;
  VAR pm_wainfo WaInfo;
  VAR num RecoverAction;
  VAR num WaSelection;
  VAR num EvtId;
  VAR errstr Arg1;
  VAR errstr Arg2;
  VAR errstr Arg3;
  VAR errstr Arg4;
  FlowSelection:=alias_giSelectionFlow;
  RecoverAction:=alias_giRecoverAction;
  WaSelection:=alias_giWaRecoverSelection;
```

Continues on next page

```
  ! Get info from selected flow
  PmGetFlowInfo FlowSelection,FlowInfo;
  IF RecoverAction=PM_RECOVER_CONTINUE_OPERATION OR
       RecoverAction=PM_RECOVER_REDO_LAST_PICK THEN
    ! Set recover action
    PmSetRecoverAction
         FlowInfo.Name,RecoverAction\EventId:=EvtId
         \Argument1:=Arg1\Argument2:=Arg2\Argument3:=Arg3
         \Argument4:=Arg4;
  ELSE
    ! Get info from selected Work Area
    PmGetWaInfo WaSelection,WaInfo;
    ! Set recover action
    PmSetRecoverAction
         FlowInfo.Name\Workarea:=WaInfo.Workarea,RecoverAction
         \EventId:=EvtId\Argument1:=Arg1\Argument2:=Arg2
         \Argument3:=Arg3\Argument4:=Arg4;
  ENDIF
  IF (EvtId<2398) AND (EvtId>2392) THEN
    PmErrorLog EvtId,FlowInfo.Name,Arg1,Arg2,Arg3,Arg4
         \EventType:=TYPE_WARN;
  ENDIF
ERROR
  ! Continue supervision on recoverable errors
  IF ERRNO=PM_ERR_FLOW_NOT_FOUND THEN
    RETURN;
  ELSEIF ERRNO=PM_ERR_WA_NOT_FOUND THEN
    RETURN;
  ELSEIF ERRNO=PM_ERR_NO_RUNNING_PROJECT THEN
    RETURN;
  ELSEIF ERRNO=PM_ERR_REDO_LAST_PICK_REJECTED THEN
    RETURN;
  ELSEIF ERRNO=PM_ERR_WORKAREA_EXPECTED THEN
    RETURN;
  ELSEIF ERRNO=PM_ERR_NOT_VALID_RECOVER_ACTION THEN
    RETURN;
  ELSE
    RETURN;
  ENDIF
ENDTRAP
```

Trap `TrapStartFlow`

This section describes the trap that executes when the start flow signal has been pulsed.

Description

A GI signal defines which flow should be started. The flow name received from *PmGetFlowInfo* is used to start the flow.

Continues on next page

Program code

```
TRAP TrapStartFlow
  VAR pm_flowinfo FlowInfo;
  VAR num FlowSelection;

  FlowSelection:=alias_giSelectionFlow;
  ! Get info from selected flow
  PmGetFlowInfo FlowSelection,FlowInfo;
  ! Start the selected flow
  PmStartFlow FlowInfo.Name;
  ERROR
     ! Continue supervision on recoverable errors
     IF ERRNO=PM_ERR_FLOW_NOT_FOUND THEN
        RETURN;
     ELSEIF ERRNO=PM_ERR_NO_RUNNING_PROJECT THEN
        RETURN;
     ELSEIF ERRNO=PM_ERR_WRONG_FLOW_STATE THEN
        RETURN;
     ELSE
        RETURN;
     ENDIF
ENDTRAP
```

Trap `TrapStopFlow`

This section describes the trap that executes when the stop flow signal has been pulsed.

Description

A GI signal defines which flow should be stopped and another GI signal defines the stop behavior. The flow name received from *PmGetFlowInfo* is used to stop the flow.

Program code

```
TRAP TrapStopFlow
  VAR pm_flowinfo FlowInfo;
  VAR num FlowSelection;
  VAR num StopOption;

  FlowSelection:=alias_giSelectionFlow;
  StopOption:=alias_giStopOptionFlow;
  ! Get info from selected flow
  PmGetFlowInfo FlowSelection,FlowInfo;
  ! Stop the selected flow
  PmStopFlow FlowInfo.Name,StopOption;
  ERROR
     ! Continue supervision on recoverable errors
     IF ERRNO=PM_ERR_FLOW_NOT_FOUND THEN
        RETURN;
     ELSEIF ERRNO=PM_ERR_NO_RUNNING_PROJECT THEN
        RETURN;
```

Continues on next page

Continued

```
            ELSE
               RETURN;
            ENDIF
        ENDTRAP
```

Trap `TrapProjectStopped`

This section describes the trap that executes if the stop project signal has been pulsed.

Description

`PmStopProj` is called to stop the current project. This trap will normally not be executed since the traps are disconnected at the same time as the stop project signal has been pulsed.

Program code

```
TRAP TrapProjectStopped
  PmStopProj;
ENDTRAP
```

Related information

PmGetFlowInfo - Get information about a specific flow on page 285.

PmStartFlow - Starts a specific flow on page 302.

PmStopFlow - Stop a specific flow on page 305.

PmStopProj - Stop current project on page 307.

5.4 System modules

5.4.1. System modules, overview

Description

An ABB IRC5 robot controller installed with the RobotWare option *Prepared for PickMaster*, together with the sub-option *PickMaster 5*, will always contain the following loaded system modules:

?`tpmrcUser` (open)

?`tpmrcSys` (open)

?`tpmrcBase` (encrypted).

5.4.2. Public system module pmrcUser

Description

The *pmrcUser* module contains declarations of work object data and tool data that can be used when setting up the line. Additional work objects and tools can be added here.

Trap `TrapDIToolEvents`

Usage

This trap is called if one DI signal is not set to the desired value at a specific robot position. The RAPID execution and robot movement is interrupted until all the specified DI and GI signal values for the robot position are set to their desired values.

WarningTime defines how long time the robot will wait for the signals to be set before a warning is logged.

PollTime defines how often the signals will be checked while waiting.

Program code

```
TRAP TrapDIToolEvents
  VAR num warningTime := 5;VAR num pollRate := 0.1;
      PmCheckToolEventInputSignals warningTime, pollRate;
ENDTRAP
```

Trap `TrapGIToolEvents`

Usage

This trap works in the same way as `TrapDIToolEvents` but is executed for GI signals.

5.4.3. Public system module pmrcSys

Description

The *pmrcSys* module contains instructions and data that are a part of the PickMaster base functionality. This module is declared as NOSTEPIN, which means that the code is open and editable but it is not possible to step into the routines. The NOSTEPIN statement can be removed for debug purposes.

This module is not saved in a backup. If modifications are needed in this module, then rename the instructions and move them to *pmrcUser*.

Procedure `PmDoAction`

Usage

This procedure prepares and executes every movement in the operation.

Description

This procedure sets up the path events and the modal data that is used in every motion.

Arguments

WorkArea

Datatype: `pm_wadescr`

Contains a reference to the work area to use.

Tgt

Datatype: `pm_targetdata`

The target data used for the move.

Act

Datatype: `pm_actiondata`

The action data used for the move.

Program code

```
PROC PmDoAction(VAR pm_wadescr WorkArea, VAR pm_targetdata Tgt,
     VAR pm_actiondata Act)
  ...
  WHILE PmGetEvent(WorkArea, Tgt.TargetHandle, Act.ActionHandle,
       Event) AND NumOfTriggEvents <= ArrSize DO
    TEST Event.Type
      ...
      CASE PM_EVENT_PROC:
      CASE PM_EVENT_DO:
      ...
      CASE PM_EVENT_GO:
      ...CASE PM_EVENT_WAIT_DI:
      ...CASE PM_EVENT_WAIT_GI:...ENDTEST
```

Continues on next page

```
ENDWHILE

TEST Act.Type
  CASE PM_APPROACH_POS:
    curr_Load := Tgt.AppProdsLoad;
  CASE PM_TARGET_POS:
    curr_Load := Tgt.AppProdsLoad;
    curr_StopPoint:=Tgt.StopPointData;
  CASE PM_DEPART_POS:
    curr_Load := Tgt.DepProdsLoad;
  DEFAULT:
    PmErrorLog 2357, ERRSTR_TASK, ValToStr(Act.Type),
         ERRSTR_CONTEXT, ERRSTR_UNUSED, ERRSTR_UNUSED;
ENDTEST
curr_WObj := Tgt.TargetWobj;
curr_Tool := Tgt.TargetTool;

PathAccLim Act.Accel.AccLim\AccMax:=Act.Accel.AccMax,
     Act.Accel.DecelLim\DecelMax:=Act.Accel.DecelMax;
AccSet Act.Accel.acc, Act.Accel.Ramp;
GripLoad curr_Load;
IF Act.ArmConfMon = TRUE THEN
  ConfL\On;
  ConfJ\On;
ELSE
  ConfL\Off;
  ConfJ\Off;
ENDIF
IF Act.SingAreaType = PM_SING_AREA_WRI THEN
  SingArea\Wrist;
ELSE
  SingArea\Off;
ENDIF

TEST NumOfTriggEvents
  CASE 0:IF Act.Move = PM_SEARCH_LIN THEN
    PmDoSearch WorkArea,Tgt.TargetHandle,Act.RobTgt,
         Act.Speed,Act.Search,curr_Tool,curr_WObj;
    PmAckTarget WorkArea,Tgt,PM_ACK;
  ELSE
    PmDoMove1 Act.Move,Act.UseConc,Act.RobTgt,Act.Speed,
         Act.Zone,curr_StopPoint,curr_Tool,curr_WObj;
```

Continues on next page

Continued

```
      ENDIF
      ...
      CASE 3:
         PmDoMove1 Act.Move, Act.UseConc, Act.RobTgt,
              Offs(Tgt.RobTgtPoint,Act.Offset.x,Act.Offset.y,Act.O
              ffset.z), Act.Speed \T1:=TriggArr{1} \T2:=TriggArr{2}
              \T3:=TriggArr{3}, Act.Zone, curr_StopPoint,
              curr_Tool, curr_WObj;
      ...
   ENDTEST
   IF bCheckDI = TRUE THEN
   WaitDI diSignal,diValue;
   ENDIF
 ERROR
   TEST ERRNO
      CASE ERR_SYM_ACCESS:
         IF Event.SignalName = "" THEN
            signalname := "<no signal>";
         ELSE
            signalname := Event.SignalName;
         ENDIF
            PmErrorLog 2354, ERRSTR_TASK, signalname,
                 ERRSTR_CONTEXT,          ERRSTR_UNUSED,
                 ERRSTR_UNUSED\ErrorHandler:=TRUE;
         RAISE;
      CASE PM_ERR_PALLET_REDUCED:
         PmAckTarget WorkArea,Tgt,PM_ACK;RAISE;
      CASE PM_ERR_PALLET_EMPTY:
         PmAckTarget WorkArea,Tgt,PM_ACK;RAISE;
      DEFAULT:
         RAISE;
   ENDTEST
 ENDPROC
 ...
```

Procedure `PmDoMove1`

Usage

This procedure evaluates whether concurrent moves are used.

Description

This procedure reads the boolean for the use of concurrent moves and from that sets the switch `\Conc` to the next move instruction.

Continues on next page

Arguments

Move

Datatype: `pm_movetype`

The type of movement that is used.

Conc

Datatype: `bool`

Tells if a concurrent move instruction is used.

ToPoint

Datatype: `robtarget`

The destination point of the movement.

Speed

Datatype: `speeddata`

The speed data that applies to movements. Speed data defines the velocity of the tool center point, the external axes and of the tool reorientation.

T1

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T2

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T3

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T4

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T5

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T6

Continues on next page

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T7

Datatype: triggdata

Variable that refers to trigger conditions and trigger activity.

T8

Datatype: triggdata

Variable that refers to trigger conditions and trigger activity.

Zone

Datatype: `zonedata`

Zone data for the movement. Zone data describes the size of the generated corner path.

Inpos

Datatype: `stoppointdata`

The setting of the dwell in the motion.

Tool

Datatype: `tooldata`

The tool in use when the robot moves. The tool center point is the point that is moved to the specified destination position.

WObj

Datatype: `wobjdata`

The work object (coordinate system) to which the robot position in the instruction is related.

Program code

```
PROC PmDoMove1(pm_movetype Move, bool Conc, robtarget ToPoint,
     speeddata Speed, \VAR triggdata T1, \VAR triggdata T2, \VAR
     triggdata T3, \VAR triggdata T4, \VAR triggdata T5, \VAR
     triggdata T6,\\VAR triggdata T7, \VAR triggdata T8, zonedata
     Zone, stoppointdata Inpos, PERS tooldata Tool, PERS wobjdata
     WObj)
  IF Conc = TRUE THEN
    PmDoMove2 Move\Conc, ToPoint, Speed \T1?T1 \T2?T2 \T3?T3
         \T4?T4 \T5?T5 \T6?T6\T7?T7 \T8?T8, Zone, Inpos, Tool,
         WObj;
  ELSE
    PmDoMove2 Move, ToPoint, Speed \T1?T1 \T2?T2 \T3?T3 \T4?T4
         \T5?T5 \T6?T6\T7?T7 \T8?T8, Zone, Inpos, Tool, WObj;
  ENDIF
ERROR
  RAISE;
ENDPROC
```

Continues on next page

Continued

Related information

pm_movetype - PickMaster movement type on page 342.

Procedure `PmDoMove2`

Usage

This procedure evaluates whether dwell is used.

Description

This procedure evaluates the `\Inpos` data to check whether a dwell is used or not.

Continues on next page

Continued

Arguments

Move

Datatype: `pm_movetype`

The type of movement that is used.

Conc

Datatype: `bool`

Tells if a concurrent move instruction is used.

ToPoint

Datatype: `robtarget`

The destination point of the movement.

Speed

Datatype: `speeddata`

The speed data that applies to movements. Speed data defines the velocity of the tool center point, the external axes and of the tool reorientation.

T1

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T2

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T3

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T4

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T5

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T6

Continues on next page

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T7

Datatype: triggdata

Variable that refers to trigger conditions and trigger activity.

T8

Datatype: triggdata

Variable that refers to trigger conditions and trigger activity.

Zone

Datatype: `zonedata`

Zone data for the movement. Zone data describes the size of the generated corner path.

Inpos

Datatype: `stoppointdata`

The setting of the dwell in the motion.

Tool

Datatype: `tooldata`

The tool in use when the robot moves. The tool center point is the point that is moved to the specified destination position.

WObj

Datatype: `wobjdata`

The work object (coordinate system) to which the robot position in the instruction is related.

Program code

```
PROC PmDoMove2(pm_movetype Move, \switch Conc, robtarget ToPoint,
     speeddata Speed, \VAR triggdata T1, \VAR triggdata T2, \VAR
     triggdata T3, \VAR triggdata T4, \VAR triggdata T5, \VAR
     triggdata T6, \VAR triggdata T7, \VAR triggdata T8, zonedata
     Zone, stoppointdata Inpos, PERS tooldata Tool, PERS wobjdata
     WObj)
  IF (Inpos.type=1) OR (Inpos.type=2) OR (Inpos.type=3) THEN
    PmDoMove3 Move \Conc?Conc, ToPoint, Speed \T1?T1 \T2?T2 \T3?T3
         \T4?T4 \T5?T5 \T6?T6,\T7?T7 \T8?T8 ,Zone \Inpos:=Inpos,
         Tool, WObj;
  ELSE
    PmDoMove3 Move \Conc?Conc, ToPoint, Speed \T1?T1 \T2?T2 \T3?T3
         \T4?T4 \T5?T5 \T6?T6, \T7?T7 \T8?T8, Zone, Tool, WObj;
  ENDIF
ERROR
  RaiseToUser \BreakOff;
ENDPROC
```

Continues on next page

Related information

The data type `stoppointdat` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Procedure `PmDoMove3`

Usage

This procedure evaluates the move type.

Description

This procedure evaluates the move type and calls the appropriate standard move instruction.

Continues on next page

Arguments

Move

Datatype: `pm_movetype`

The type of movement that is used.

Conc

Datatype: `bool`

Tells if a concurrent move instruction is used.

ToPoint

Datatype: `robtarget`

The destination point of the movement.

Speed

Datatype: `speeddata`

The speed data that applies to movements. Speed data defines the velocity of the tool center point, the external axes and of the tool reorientation.

T1

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T2

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T3

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T4

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T5

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T6

Continues on next page

Continued

Datatype: `triggdata`

Variable that refers to trigger conditions and trigger activity.

T7

Datatype: triggdata

Variable that refers to trigger conditions and trigger activity.

T8

Datatype: triggdata

Variable that refers to trigger conditions and trigger activity.

Zone

Datatype: `zonedata`

Zone data for the movement. Zone data describes the size of the generated corner path.

Inpos

Datatype: `stoppointdata`

The setting of the dwell in the motion.

Tool

Datatype: `tooldata`

The tool in use when the robot moves. The tool center point is the point that is moved to the specified destination position.

WObj

Datatype: `wobjdata`

The work object (coordinate system) to which the robot position in the instruction is related.

Program code

```
PROC PmDoMove3(pm_movetype Move, \switch Conc, robtarget ToPoint,
      speeddata Speed, \VAR triggdata T1, \VAR triggdata T2, \VAR
      triggdata T3, \VAR triggdata T4, \VAR triggdata T5, \VAR
      triggdata T6,\VAR triggdata T7, \VAR triggdata T8, zonedata
      Zone, \stoppointdata Inpos, PERS tooldata Tool, PERS wobjdata
      WObj)
  TEST Move
    CASE PM_MOVE_LIN:
      MoveL \Conc?Conc, ToPoint, Speed, Zone \Inpos?Inpos, Tool
            \WObj:=WObj;
    CASE PM_MOVE_JOINT:
      MoveJ \Conc?Conc, ToPoint, Speed, Zone \Inpos?Inpos, Tool
            \WObj:=WObj;
    CASE PM_TRIGG_LIN:
      TriggL \Conc?Conc, ToPoint, Speed, T1, \T2?T2 \T3?T3 \T4?T4
            \T5?T5 \T6?T6, \T7?T7 \T8?T8, Zone \Inpos?Inpos, Tool
            \WObj:=WObj;
```

Continues on next page

Continued

```
                CASE PM_TRIGG_JOINT:
                  TriggJ \Conc?Conc, ToPoint, Speed, T1, \T2?T2 \T3?T3 \T4?T4
                      \T5?T5 \T6?T6, \T7?T7 \T8?T8, Zone \Inpos?Inpos, Tool
                      \WObj:=WObj;
                DEFAULT:
                  PmErrorLog 2358, ERRSTR_TASK, ValToStr(Move),
                      ERRSTR_CONTEXT, ERRSTR_UNUSED, ERRSTR_UNUSED;
              ENDTEST
            ERROR
              RaiseToUser \BreakOff;
            ENDPROC
```

Related information

pm_movetype - PickMaster movement type on page 342.

The instruction `TriggL` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

The instruction`TriggJ` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

The instruction `MoveL`*Technical reference manual - RAPID Instructions, Functions and Data types*.

The instruction `MoveJ` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Procedure `PmDoSearch`

Usage

This procedure is used when performing a stack search.

Description

This procedure evaluates the search stop type, calls the `SearchL` instruction and evaluates the search result using the `PmSearchAdjust` instruction.

The procedure uses an error handler. If the `SearchL` routine returns `ERR_WHL_SEARCH`, the error handler will raise `PM_ERR_PALLET_EMPTY` if the search movement has reached below the expected height of the last layer. The error handler raises the `PmSearchAdjust` errors `PM_ERR_PALLET_REDUCED` and `PM_ERR_PALLET_EMPTY` which are recovered/recoverable in the *PmMain* module.

Continues on next page

Continued

Arguments

Move

Datatype: `pm_wadescr`

Contains a reference to the work area to use.

TargetHandle

Datatype: `pm_targethandle`

Contains a reference to the target handle.

ToPoint

Datatype: `robtarget`

The destination point of the search movement.

Speed

Datatype: `speeddata`

The speed data that applies to the search movement. Speed data defines the velocity of the tool center point, the additional axes, and of the tool reorientation.

SearchData

Datatype: `pm_searchdata`

Contains search type, search stop type, I/O trigger type and search signal.

Tool

Datatype: `tooldata`

Search tool. The tool in use when the robot makes the search movement. The tool center point is the point that is moved to the specified destination position.

WObj

Datatype: `wobjdata`

The work object (coordinate system) to which the robot position in the instruction is related.

Continues on next page

Program code

```
PROC PmDoSearch(VAR pm_wadescr WorkArea, VAR pm_targethandle
     TargetHandle, robtarget ToPoint, speeddata Speed,
     pm_searchdata SearchData, PERS tooldata Tool, PERS wobjdata
     WObj)
  VAR robtarget SearchPoint;
  VAR signaldi diSearchSignal;
  VAR bool SearchError:=FALSE;

  GetDataVal SearchData.SignalName,diSearchSignal;
  TEST SearchData.StopType
     CASE PM_STOP_NOT_USED:
     TEST SearchData.IOTriggType    CASE PM_BOTH_FLANKS:
        SearchL \Sup, diSearchSignal \Flanks, SearchPoint, ToPoint,
             Speed, Tool \WObj:=WObj;
        ...
     DEFAULT:
        PmErrorLog 2362, ERRSTR_TASK,
             ValToStr(SearchData.StopType), ERRSTR_CONTEXT,
             ERRSTR_UNUSED, ERRSTR_UNUSED;
  ENDTEST
  TEST SearchData.SearchType
     CASE PM_SEARCH_X:
        PmSearchAdjust WorkArea, SearchData.SearchType,
             SearchPoint.trans.x;
     CASE PM_SEARCH_Y:
        PmSearchAdjust WorkArea, SearchData.SearchType,
             SearchPoint.trans.y;
     CASE PM_SEARCH_Z:
        PmSearchAdjust WorkArea, SearchData.SearchType,
             SearchPoint.trans.z;
     DEFAULT:
        PmErrorLog 2361, ERRSTR_TASK,
             ValToStr(SearchData.SearchType), ERRSTR_CONTEXT,
             ERRSTR_UNUSED, ERRSTR_UNUSED;
  ENDTEST

ERROR
  TEST ERRNO
     CASE ERR_WHLSEARCH:
        SearchError:=TRUE;
        SearchPoint:=ToPoint;
        TRYNEXT;
     CASE PM_ERR_PALLET_REDUCED:
        IF SearchError = TRUE THEN
           ...
           ...
           RAISE ERR_WHLSEARCH;
```

Continues on next page

Continued

```
            ELSE
               RAISE;
            ENDIF
         CASE PM_ERR_PALLET_EMPTY:
            RAISE;
         DEFAULT:
            RaiseToUser \BreakOff;
      ENDTEST
   ENDPROC
```

Related information

pm_movetype - PickMaster movement type on page 342

pm_searchdata - PickMaster search data on page 352

pm_searchtype - PickMaster stack search type on page 354

pm_stoptype - PickMaster stop type on page 358

PmSearchAdjust - Adjust number of remaining layers on page 295

The instruction `SearchL`, see *Technical reference manual - RAPID Instructions, Functions and Data types*.

Variable `pm_home_Wobj`

Usage

This variable is used for the default installed work area PM_HOME to connect to an always existing work object.

Description

The variable is only used to get PM_HOME work area. It is a copy of the installed work object wobj0.

Variables `LastRobTgt`, `LastWobj` and `LastTool`

Usage

These variables are used to store the last position, work object, and tool.

Description

The variables are used to store the last position’s properties, to be able to calculate an intermediate position.

Continues on next page

Continued

Program code

```
InterMid:=PmCalcIntermid(PmLastRobTgt,PmLastTool,PmLastWobj,
     Act.RobTgt,Tgt.TargetTool,Tgt.TargetWobj, IntermidPart
     \MaxAngle:=MaxToolAngle\MinAngle:=MinToolAngle\AngleLimAx6
     \MinZ:=MinZ);
TASK PERS robtarget PmLastRobTgt:=[[0,0,0], [0,0,0,0],
     [0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
TASK PERS wobjdata PmLastWobj:=[FALSE, TRUE,
     "",[[0,0,0],[1,0,0,0]], [[0,0,0],[1,0,0,0]]];
TASK PERS PmLastTool:=[TRUE,
     [[0,0,0],[1,0,0,0]],[0.001,[0,0,0.001],[1,0,0,0],0,0,0]];
...
PmLastRobTgt:=Act.RobTgt;
PmLastWobj:=Tgt.TargetWobj;
PmLastTool:=Tgt.TargetTool;
...
```

6 Runtime operation

6.1. Introduction

Structure of this chapter

This chapter describes the runtime operating interface to PickMaster.

The runtime operating interface consists of three parts:

?tPickMaster FlexPendant interface. A graphic operator's interface.

?tPickMaster I/O interface. Used by a PLC.

?tPickMaster RAPID interface. The RAPID interface can be customized to receive and handle requests not covered by the FlexPendant or the I/O interface. For a complete description of program modules, functions, procedures, and data specific for palletizing see *RAPID program on page 187*, and *RAPID reference information on page 277*.

Prerequisites

The runtime operating interface is only available with the option *Prepared for PickMaster*, with the sub-option *PickMaster 5*.

6.2 FlexPendant interface

6.2.1. Introduction to PickMaster FlexPendant interface

PickMaster FlexPendant interface

The PickMaster FlexPendant interface is available from the ABB menu on the FlexPendant. It is a graphical user interface designed to control and/or supervise the palletizing process.

The PickMaster FlexPendant interface covers the following four areas:

?tOpen Project

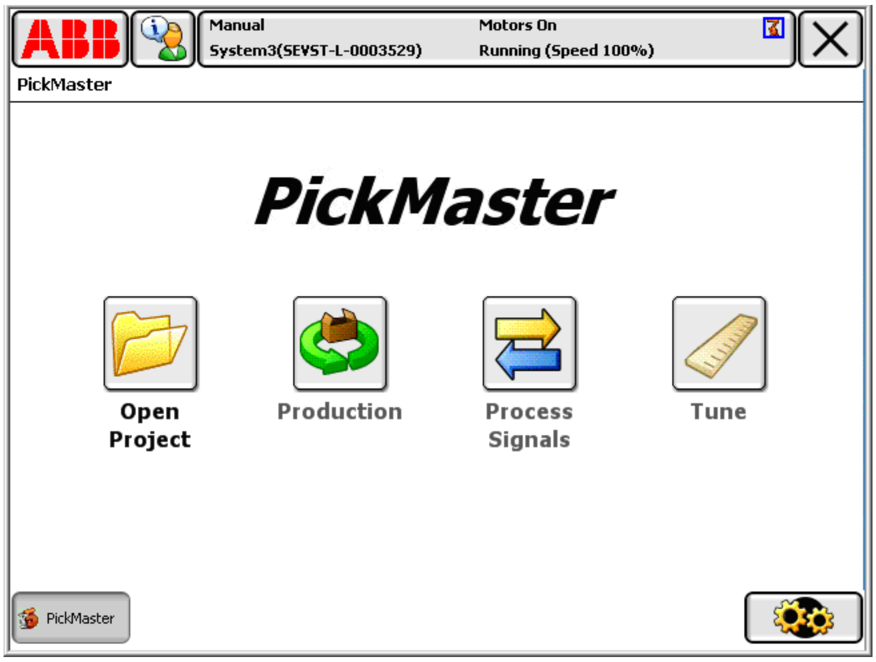
?tProduction

?tProcess Signals

?tTune

Illustration

The illustration shows the PickMaster main menu.



xx0600002995

Open Project

Open Project displays a list of all the PickMaster projects that have been downloaded to the robot controller. A project must be opened before it can be started from the production window. The status bar always contains information about the loaded project as well as the current location within the window hierarchy.

See *Opening a project on page 232*.

Production

Production is used by the operator to start and stop and monitor the palletizing process.

See *Starting and stopping production on page 233*.

Continues on next page

Continued

Process Signals

Process Signals presents a list of all the work areas, items, events, and the I/O signals that are connected to each work area. It is possible not only to view but also to set new signal values. This window also presents all the tool configurations that build up the tool together with the zones and the activators. For a detailed description of the tool configuration, see *Zone frame on page 374* and *Activators properties on page 108*.

See also *Process Signals on page 246*.

Tune

Tune is used to change the parameter values online while running the PickMaster application.

See *Tuning on page 249*.

6.2.2. Opening a project

Introduction

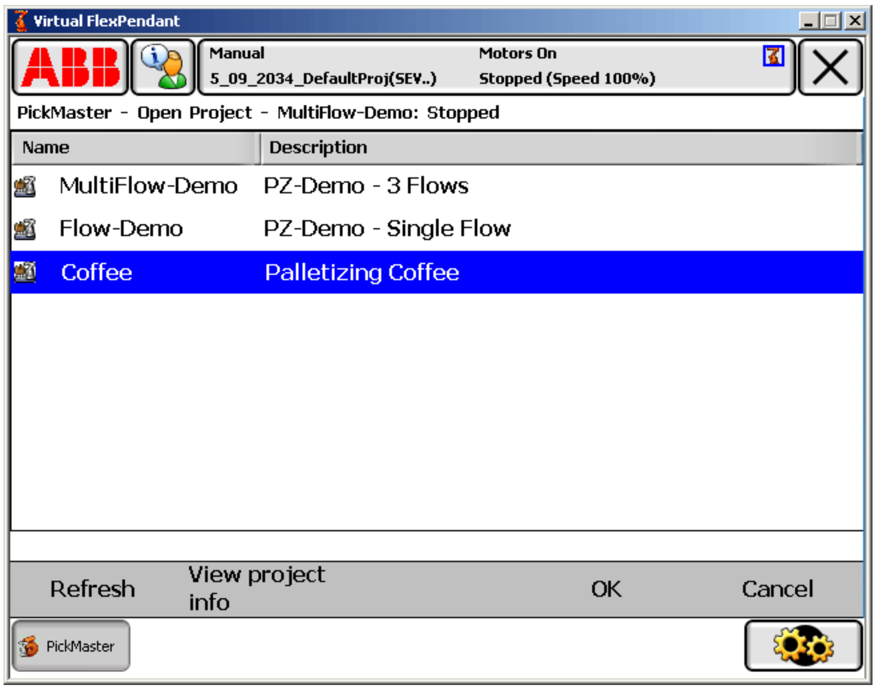
This section describes how to open a PickMaster project from the Open Project window. A list of all the available projects on the controller appears together with a description, if provided. To provide a description, see *Project Settings on page 138*.

Opening a project

	Action	Note
1.	On the **PickMaster** main menu, tap **Open Project**.	The **Open Project** button is only accessible if the current PickMaster project is stopped.
2.	Tap a project.	When a project is selected, the **OK** button appears. To update the list of projects, tap **Refresh**.
3.	To open the project and return to the main menu, tap **OK**.	
4.	To view information about the project, tap **View project info**.	The window shows information about line path, transfer data, PickMaster version, user, computer, and a description of the project.
5.	To close the window, tap **Cancel**.	

Illustration

The illustration shows the **Open Project** window.



PickMaster - Open Project - MultiFlow-Demo: Stopped

Name	Description
MultiFlow-Demo	PZ-Demo - 3 Flows
Flow-Demo	PZ-Demo - Single Flow
Coffee	Palletizing Coffee

xx0600002996

Related information

Starting and stopping production on page 233.

Process Signals on page 246.

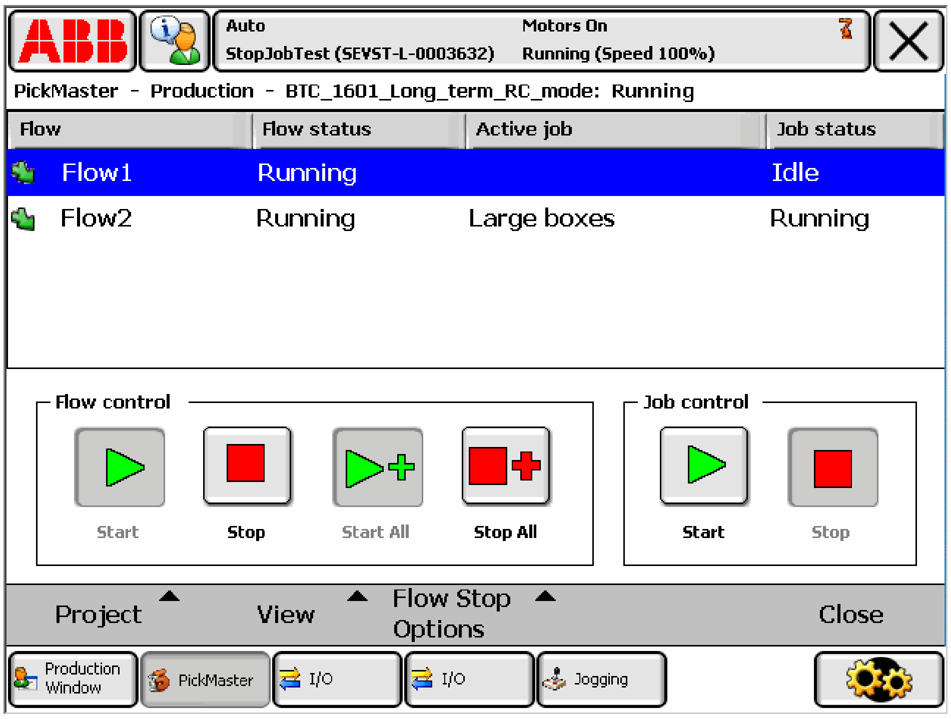
6.2.3. Starting and stopping production

Overview

This section describes how to start and stop the palletizing process from the Production window.

Illustration, Production window

The illustration shows an example of the Production window when three flows are defined in the PickMaster project.


Auto
StopJobTest (SEVST-L-0003632)
Motors On
Running (Speed 100%)

PickMaster - Production - BTC_1601_Long_term_RC_mode: Running

Flow	Flow status	Active job	Job status
Flow1	Running		Idle
Flow2	Running	Large boxes	Running

xx0700000229

Flow	The name of the flow as specified in the PickMaster project.
Status	The status of the flow, which can be **Stopped**, **Running**, or **Error**. The status can also indicate type of stop in progress, for example, **Stopping after cycle**, **Stopping after layer**, and **Stopping after pallet**.
Active job	The currently running job on the flow, that is, the name of the active master operation set.
Job status	The current job status of the flow, which can be **Idle** (no job is running), **Running** or **Stopping** (a job has been ordered to stop).

Starting a project

	Action	Note
1.	On the **Production** menu, tap **Project**.	
2.	Tap **Start**.	A warning appears if the system is in motors off state.

Continues on next page

Stopping a project

	Action	Note
1.	On the **Production** menu, tap **Project**.	
2.	Tap **Stop**. To proceed, tap **Yes**. To cancel, tap **No**.	A warning appears.

Restarting RAPID

The following procedure describes how to restart the program execution. This is the same function as pushing the hardware button *Start* on the IRC5 FlexPendant.

	Action	Note
1.	On the **Production** menu, tap **Project**.	
2.	Tap **Restart RAPID**.	The **Restart RAPID** button is only available when program is paused.

Starting a specific flow

Starting the flow will enable the starting and execution of a palletizing job.

-

NOTE!

The PickMaster project must be started before a flow can be started.

	Action
1.	In the list of defined flows, tap on the flow to start. The **Start Flow** button becomes available.
2.	Tap **Start Flow**.

Stopping a specific flow

Stopping the flow pause the execution of a running palletizing job. When the flow is restarted, the execution of the job will continue. A flow stop option can be selected to specify how and when the flow must be stopped. If no job is running, the flow will stop immediately.

	Action
1.	In the list of defined flows, tap on the flow to stop. The **Stop Flow** button becomes available.
2.	Tap **Stop Flow**.

Starting a specific job

-

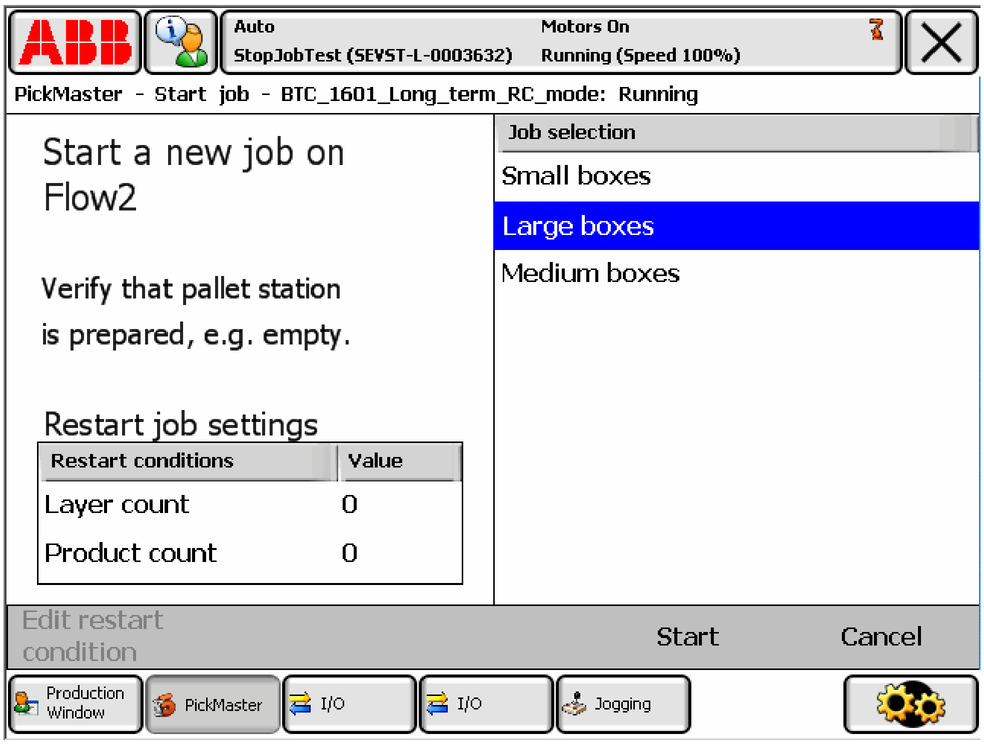
Continues on next page

NOTE!

The flow must be running before a job can be started.

	Action
1.	In the list of defined flows, tap on the flow to start a job on. The **Start Job** button appears.
2.	Tap **Start Job**. The Start Job window appears.
3.	Select job and press start.

The following illustration shows an example of the Start job window.



Auto StopJobTest (SEVST-L-0003632)
Motors On Running (Speed 100%)

PickMaster - Start job - BTC_1601_Long_term_RC_mode: Running

Start a new job on Flow2

Verify that pallet station is prepared, e.g. empty.

Restart job settings

Restart conditions	Value
Layer count	0
Product count	0

Job selection
Small boxes
Large boxes
Medium boxes

en1000000858

Job selection	A list of palletizing jobs defined for the master work area to select from.
Restart conditions	A list of parameters that needs to be specified when an unfinished job shall be restarted, for example, a half finished pallet pattern. When starting a new job, do not update these parameters. **NOTE:** A restart is not possible if the last operation was a partially completed multi drop operation. In that case, some products has to be manually removed from the stack before starting, for example, a removal of the top layer.
Layer count	Specifies number of available full layers, including defined pallet and slip sheets.
Product count	Specifies number of available products in the top layer. If the top layer is full, product count shall be set to zero.

Editing restart conditions

	Action
1.	Select **restart condition**.
2.	Press **Edit restart condition** menu.
3.	Enter appropriate value on the displayed numerical pad and press **OK**.

Continues on next page

Continued

Stopping a specific job

If stopping a job, it will be finished without being completed. The job will stop as soon as any currently ongoing or pending pick-places cycle has been completed or cancelled. Pending position requests on slaves will be cancelled if no targets are generated before the slave's stop job timeout has passed.

If the job is waiting on a slave that has been running out of products, stop job can be used to finish the job without running any further pick-place cycles. The job will become stopped after the stop job timeout has passed.

NOTE: The flow must be running before a job can be stopped.

	Action
1.	In the list of defined flows, tap on the flow to stop the job on. The **Stop Job** button becomes available.
2.	Tap **Stop Job**. A pop-up window appears.
3.	In the pop-up window, confirm that the job shall be stopped.

Starting and stopping all flows

To...	Do this
Start all flows	Tap **Start All Flows**.
Stop all flows	Stop **All Flows**.

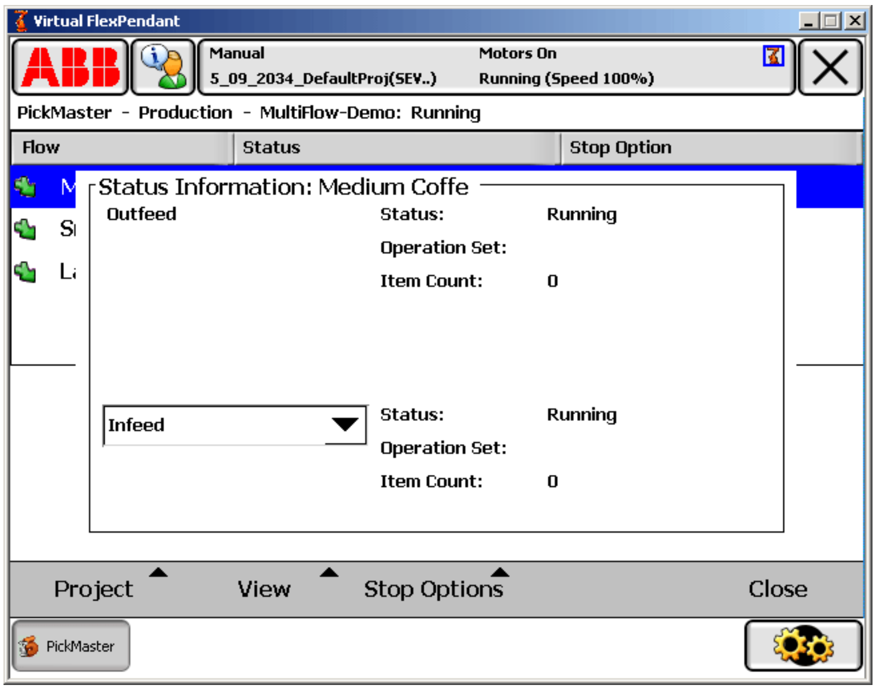
Viewing flow status

	Action
1.	In the list of flows, tap on a specific flow. In the Production window, the **Status** command on the **View** menu becomes available.
2.	On the **View** menu, tap **Status** and select a work area.
3.	To close the Status Information window, tap **Status** on the **View** menu.

Continues on next page

Continued

The following illustration shows an example of the **Production** window when the flow status information appears. In this example the project *Medium Coffee* includes the work areas Medium Outfeed and Medium Infeed.


Virtual FlexPendant

Manual
5_09_2034_DefaultProj(SEV..)
Motors On
Running (Speed 100%)

PickMaster - Production - MultiFlow-Demo: Running

Status Information: Medium Coffe

Outfeed	Status:	Running
	Operation Set:	
	Item Count:	0
Infeed	Status:	Running
	Operation Set:	
	Item Count:	0

xx0700000228

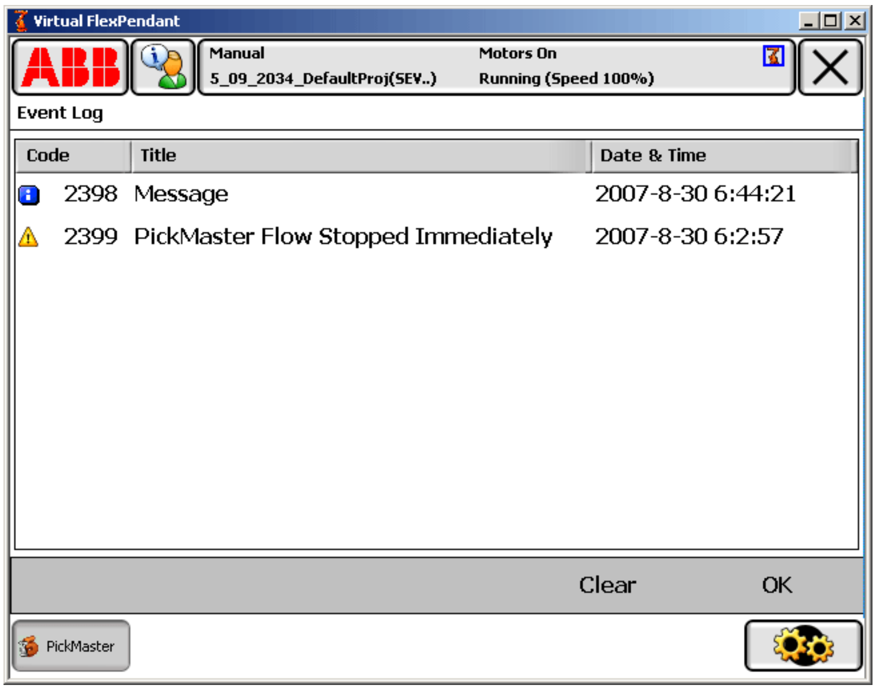
Work Area	The name of the work area as specified in the PickMaster software. For further details, see *Feeder on page 154*.
Status	Provides status information about the work area, which can be **Running** or **Error**.
Operation Set	Specifies the last accessed operation set on the work area.
Item Count	A counter for the accumulated number of items that has been picked or placed on the work area since the project start.

Viewing messages

	Action
1.	On the **View** menu, tap **Messages**.
2.	In the list of messages, tap a specific message. A message window appears.

Continues on next page

The following illustration shows an example of the message window where only messages concerning flow recovery are shown.



Code	Title	Date & Time
2398	Message	2007-8-30 6:44:21
2399	PickMaster Flow Stopped Immediately	2007-8-30 6:2:57

xx0700000482

Code	The code of the message.
Title	The title of the message.
Date	The date and time of the generated message.
Clear	Clears the list of messages. Note that the messages are *not* deleted. If the production window is closed and then reopened, the cleared messages appear again.
OK	Closes the window.

For details about message configuration, see *Messages on page 147*.

Changing or viewing flow stop options

	Action
1.	In the list of flows, tap on the flow to change the stop option for. The **Flow Stop Options** button on the **Production** menu becomes available.
2.	Tap **Flow Stop Options** and select a stop option. The currently selected stop option is checked.

There are four different ways to stop each product flow from the FlexPendant. Each stop option is described according to what will happen when a flow is stopped with the specific stop option and after tapping **StopFlow**.

Flow stop option	Description
Finish cycle	The robot will continue palletizing until the current pick/place cycle of the job is finished.
Finish layer	The robot will continue palletizing until the current layer of the job is finished.

Continues on next page

Continued

Flow stop option	Description
Finish job	The robot will continue palletizing until the current job is finished.
Stop immediately	The flow is stopped and a flow recovery action must be selected when restarting. If the flow is the current active flow, the robot will stop immediately, without finishing a started cycle. A warning symbol appears next to the flow, indicating that a flow recovery action must be selected, and the status changes to Stopped. If another flow shares one of the slave work areas, a warning symbol will appear also next to that flow when this slave is requested, but the flow status does not change. If the stopped flow is not the current active flow, the robot will continue palletizing using the remaining running flows.

A StopFlow command can be cancelled using Undo. However, this is not possible with flow stop option "Stop Immediately". To cancel a requested stop action:

?tOn the **Stop Options** menu, tap **Undo**.

Related information

Opening a project on page 232.

Process Signals on page 246.

Feeder on page 154.

Events on page 144.

Robot Settings on page 149.

6.2.4. Flow recovery

Overview

If a flow has been stopped by the event signals **Error Source** and **Trigger**, or with the stop option **Stop immediately**, a warning symbol appears next to the flows that are affected. When selecting a flow to be recovered, a warning symbol will appear on the **Start Flow** button in the **Production** window. This section describes how to resume a specific flow in such a case.

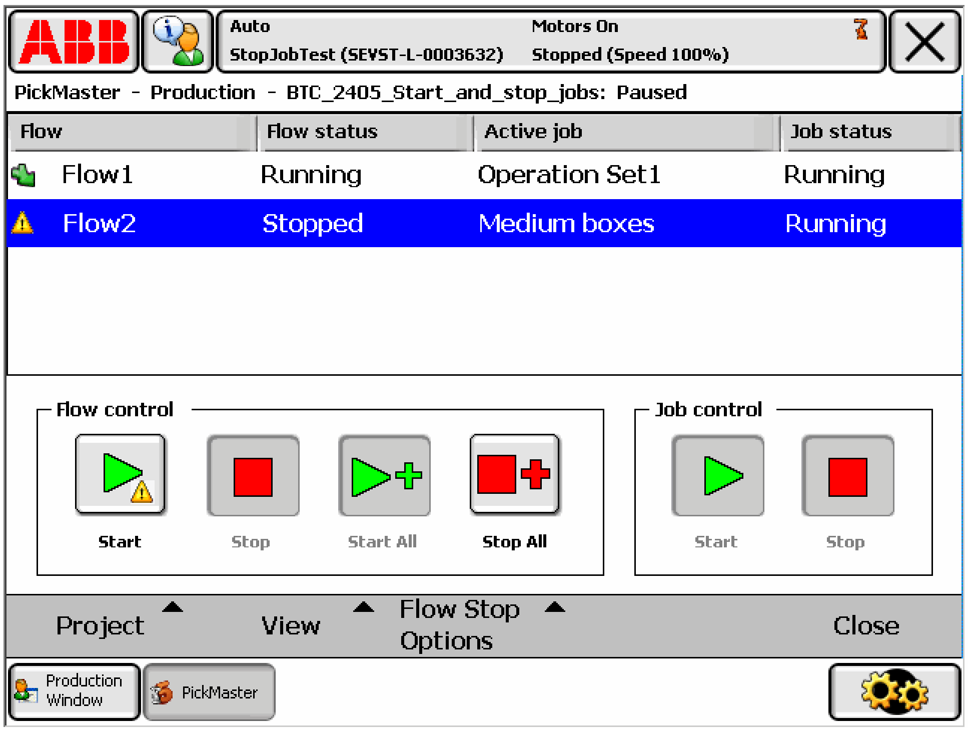
Recovering a specific flow

	Action	Note
1.	In the **Production** window, tap **Start Flow**.	See *Illustration, Production window with a flow to recover on page 241*.
2.	The **Restart options** window appears. Select one of the displayed recovery options and tap **OK**.	You get three options to continue. See *Illustration, Restart options window on page 241*.
3.	The **Production** window appears and there is a note symbol on the **Start Flow** button. Tap **Start Flow**.	See *Illustration, Production window after selecting a recovery option on page 243*.
4.	A dialog box containing information about the selected option appears. Verify the status as specified in the message, and then tap **OK**.	After tapping **OK**, the flow is in production again. If the RAPID program has stopped, it will restart. See *Illustration, Confirm restart information on page 244*

Continues on next page

Illustration, Production window with a flow to recover

The following figure illustrates the **Production** window when an error/event has occurred.



Auto StopJobTest (SEVST-L-0003632) Motors On Stopped (Speed 100%)

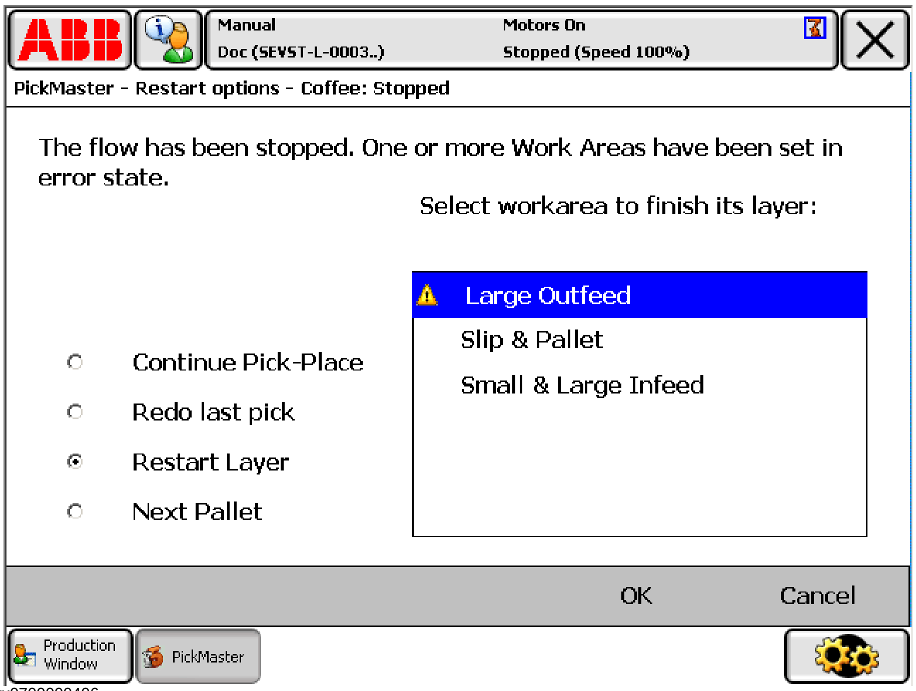
PickMaster - Production - BTC_2405_Start_and_stop_jobs: Paused

Flow	Flow status	Active job	Job status
Flow1	Running	Operation Set1	Running
Flow2	Stopped	Medium boxes	Running

xx0700000405

Start Flow	The warning symbol indicates that a flow has been stopped by an error in a work area or with the stop option **Stop immediately**.

Illustration, Restart options window



Manual Doc (SEVST-L-0003..) Motors On Stopped (Speed 100%)

PickMaster - Restart options - Coffee: Stopped

The flow has been stopped. One or more Work Areas have been set in error state.

Select workarea to finish its layer:

- Large Outfeed
- Slip & Pallet
- Small & Large Infeed

- ○ Continue Pick-Place
- ○ Redo last pick
- ● Restart Layer
- ○ Next Pallet

xx0700000406

Continues on next page

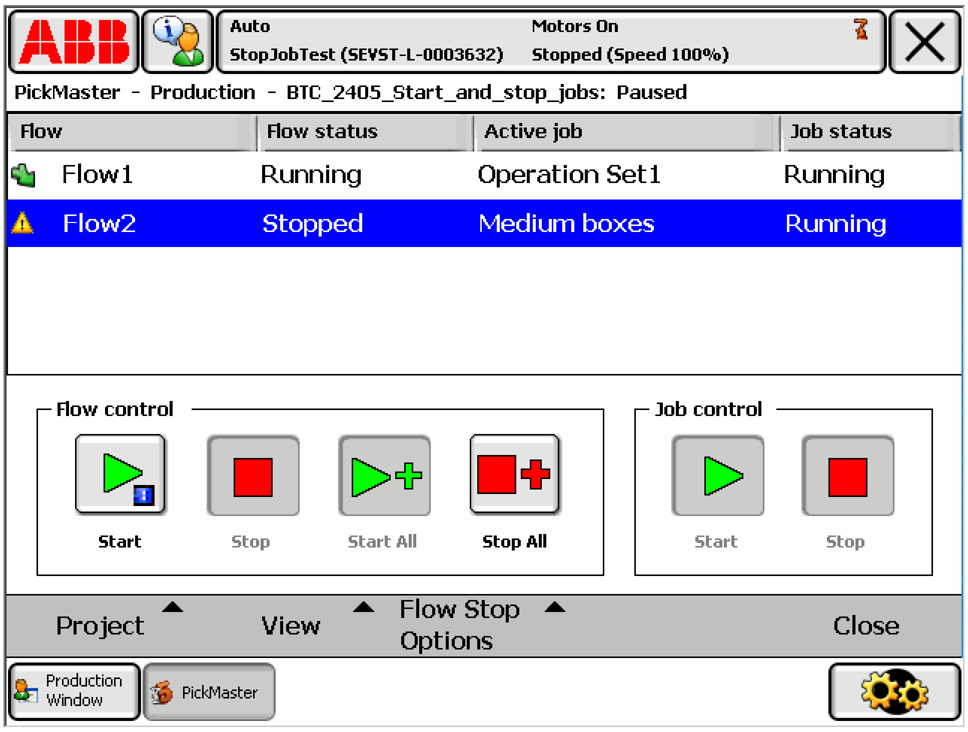
Continue Pick-Place	The flow will continue from where it was stopped. Before continuing with the working procedure, you need to verify that: ?tThe reason for the error has been handled. ?tThe tool and the work areas are holding the correct formats.
Redo last pick	The flow will repeat the last pick operation. Redo last pick is only enabled in the gap *after* the first product is picked and *before* the first product is placed. Before continuing you must verify that: ?tThe reason for the error has been handled. ?tThe tool is empty. ?tThe correct format can be supplied in the pick work area.
Restart layer	The work areas set in error state are marked with a warning symbol. Select the work area for the layer you intend to restart. It is recommended to select the master work area. Prior to continuing with the working procedure, you need to verify that: ?tThe reason for the error has been handled. ?tThe tool is empty. ?tThe correct format can be supplied in the work areas. ?tThe layer that will be repeated is restored.
Next pallet	The work areas set in error state are marked with a warning symbol. Select the work area for the pallet you intend to restart. We recommended selecting the master work area. Before continuing with the working procedure, you need to verify that: ?tThe reason for the error has been handled. ?tThe tool is empty. ?tThe correct format can be supplied in the work areas.

NOTE: The robot moves directly to the pick work area after a restart with flow recovery when using **Redo last pick**, **Restart layer**, or **Next pallet**. Any passed safe position will not be considered in the planned path. Ensure that the robot is jogged to a secured position.

Continues on next page

Illustration, Production window after selecting a recovery option

The following figure illustrates the **Production** window after selecting a recovery option for a flow.



Auto
StopJobTest (SEVST-L-0003632)
Motors On
Stopped (Speed 100%)

PickMaster - Production - BTC_2405_Start_and_stop_jobs: Paused

Flow	Flow status	Active job	Job status
Flow1	Running	Operation Set1	Running
Flow2	Stopped	Medium boxes	Running

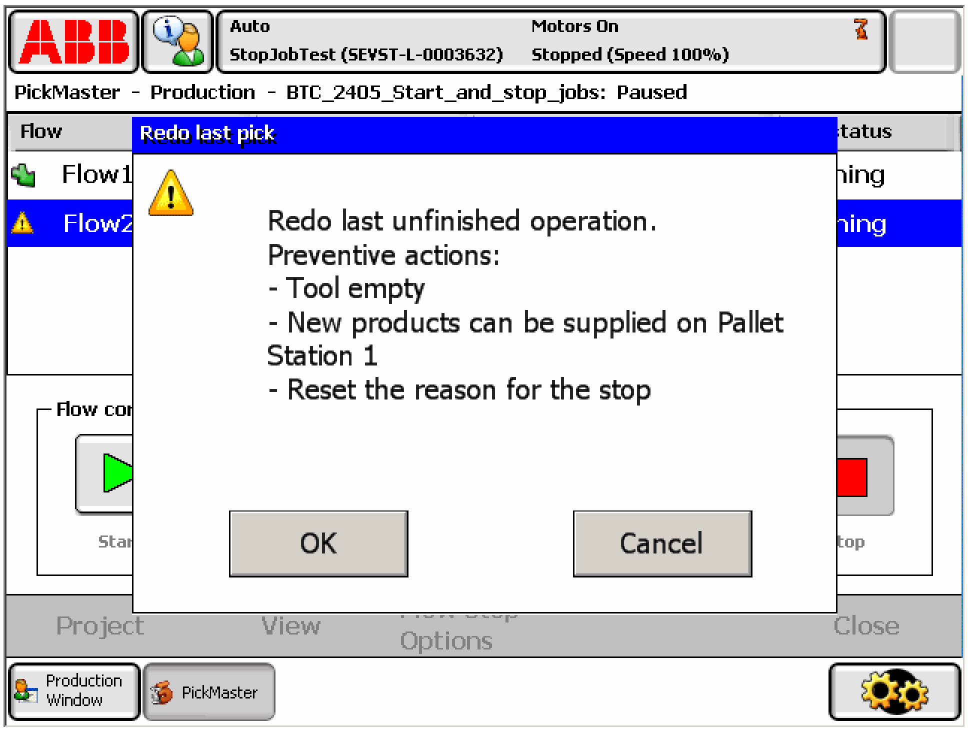
en1000000988

Start Flow	The info symbol indicates that a flow recovery action has been selected for the flow. More information about the selected action, expected number of products in robot tool and on work areas and so on are displayed when **start** is pressed.

Continues on next page

Illustration, Confirm restart information

The following figure illustrates the **information** window to confirm while starting the flow after selecting a flow recovery action.



Redo last pick

Redo last unfinished operation.
Preventive actions:
- Tool empty
- New products can be supplied on Pallet Station 1
- Reset the reason for the stop

OK
Cancel

en1000000989

Restarting other flows after error event

If a flow is stopped by an error event (or "Stop immediately") while the robot is executing the flow, the RAPID program will also stop.

In order to restart execution of the other flows without first resolving the error:

1. Verify that the robot tool is prepared for the next pick/place cycle, for example, empty
2. Jog the robot to a safe position from where execution of the next pick/place cycle can be started
3. Move PP to Main
4. Restart RAPID
5. Confirm a warning message on the flex pendant

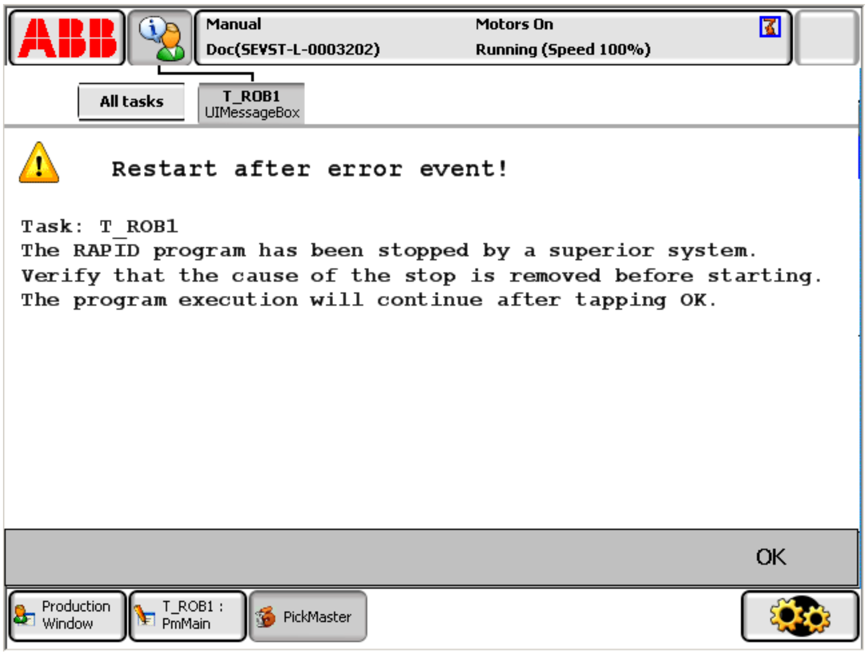
Continues on next page

WARNING!

If the PP is not moved to Main in step 3, the robot may start moving to an already fetched but not yet executed target. Execution will then stop on the next of the following functions: `PmGetEvent`, `PmGetTarget`, or `PmGetTgtAction`.

Illustration, Restart other flows after error event

The following figure illustrates the **warning** message which must be confirmed if RAPID is restarted after an error event of the executing flow.



Restart after error event!

Task: T_ROB1
The RAPID program has been stopped by a superior system.
Verify that the cause of the stop is removed before starting.
The program execution will continue after tapping OK.

xx0700000503

Related information

Starting and stopping production on page 233

6.2.5. Process Signals

Introduction

This section describes how to use the **Process Signals** window to manually control and view:

?tThe I/O signals that are connected to a work area.

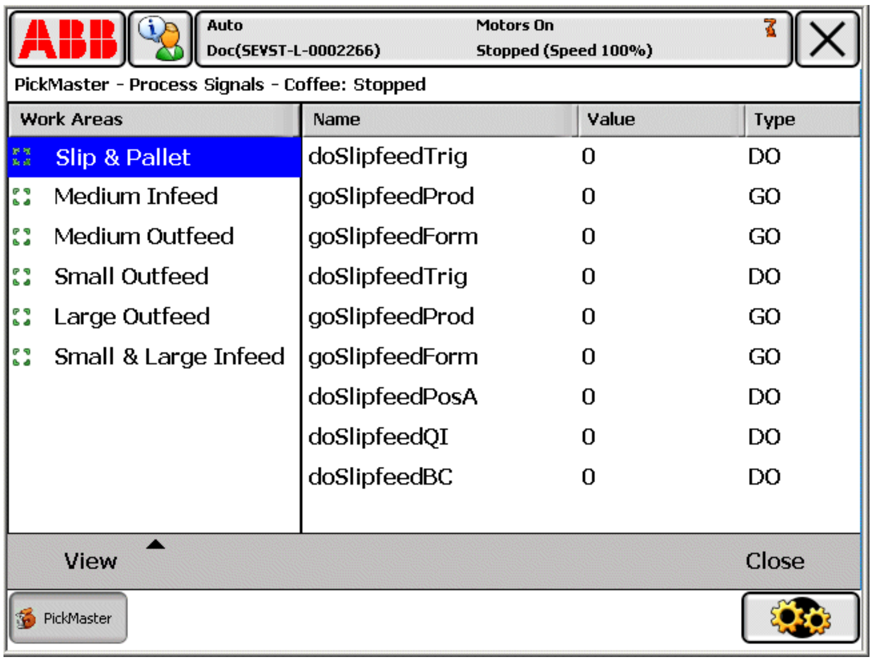
?tThe zones and the activators that build up the tool configuration.

?tThe event signals.

Viewing the work area signals

	Action	Note
1.	On the **PickMaster** main menu, tap **Process Signals**.	
2.	Tap **View** and select **Work Areas**.	See *Feeder on page 154*.
3.	From the **Work Areas** list, tap one work area.	

The following illustration shows an example of the **Process Signals** window when one work area is selected.



PickMaster - Process Signals - Coffee: Stopped

Work Areas
Slip & Pallet
Medium Infeed
Medium Outfeed
Small Outfeed
Large Outfeed
Small & Large Infeed

Name	Value	Type
doSlipfeedTrig	0	DO
goSlipfeedProd	0	GO
goSlipfeedForm	0	GO
doSlipfeedTrig	0	DO
goSlipfeedProd	0	GO
goSlipfeedForm	0	GO
doSlipfeedPosA	0	DO
doSlipfeedQI	0	DO
doSlipfeedBC	0	DO

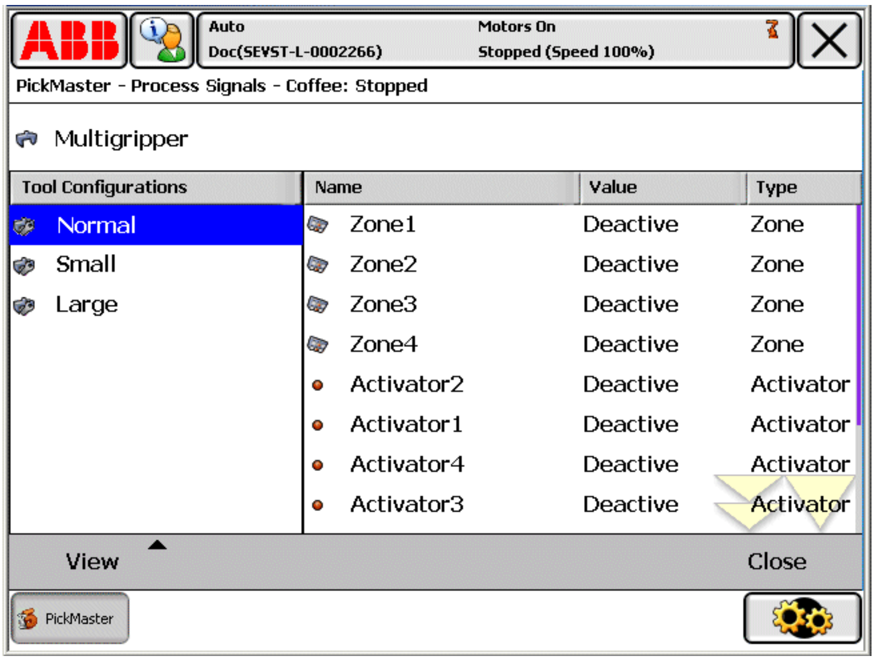
xx0700000226

Viewing the tool configuration

	Action
1.	On the **PickMaster** main menu, tap **Process Signals**.
2.	Tap **View** and select **Tool**.
3.	From the **Tool** list, tap one tool configuration.

Continues on next page

The following illustration shows an example of the **Process Signals** window when one tool configuration is selected.



Auto
Doc(SEVST-L-0002266)

Motors On
Stopped (Speed 100%)

PickMaster - Process Signals - Coffee: Stopped

Multigripper

Tool Configurations
Normal
Small
Large

Name	Value	Type
Zone1	Deactive	Zone
Zone2	Deactive	Zone
Zone3	Deactive	Zone
Zone4	Deactive	Zone
Activator2	Deactive	Activator
Activator1	Deactive	Activator
Activator4	Deactive	Activator
Activator3	Deactive	Activator

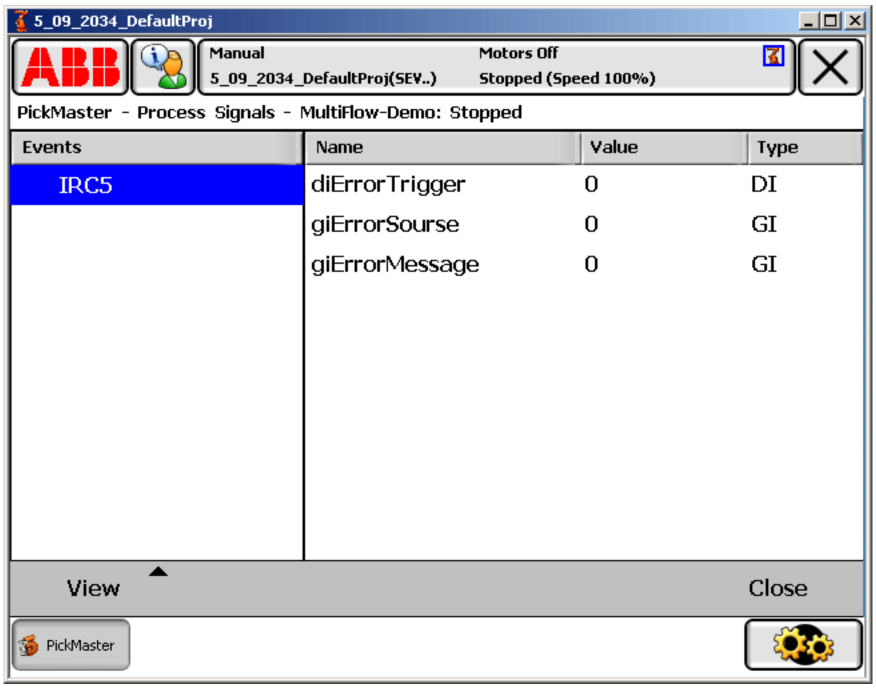
xx0700000225

Viewing events

	Action	Note
1.	On the **PickMaster** main menu, tap **Process Signals**.	
2.	Tap **View** and select **Events**.	See *Events on page 144*.
3.	From the **Events** list, tap one controller.	

Continues on next page

The following illustration shows an example of the **Process Signals** window when one controller is selected.



PickMaster - Process Signals - MultiFlow-Demo: Stopped

Events	Name	Value	Type
IRC5	diErrorTrigger	0	DI
	giErrorSourse	0	GI
	giErrorMessage	0	GI

xx0700000407

Related information

Opening a project on page 232.

Starting and stopping production on page 233.

6.2.6. Tuning

Introduction

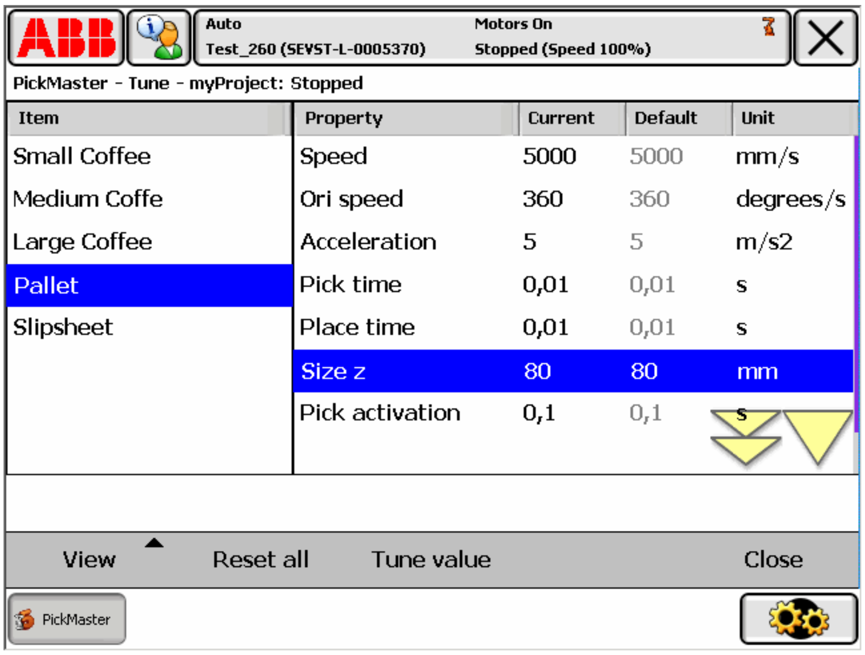
This section describes how to tune the parameter values online in the **Tune** window. A parameter can be tuned at any time for a selected project, for example while the project is running. Parameter tune updates affects the received data of the next calls to Pickmaster RAPID instructions (for example `PmGetTarget`).

Illustrations, Tune window

Below the FlexPendant interface illustrating tuning of an item and a work area.

Item

The illustration shows an example of the Tune window when five products are defined, and the properties that can be tuned for each product.



Item
Small Coffee
Medium Coffe
Large Coffee
Pallet
Slipsheet

Property	Current	Default	Unit
Speed	5000	5000	mm/s
Ori speed	360	360	degrees/s
Acceleration	5	5	m/s2
Pick time	0,01	0,01	s
Place time	0,01	0,01	s
Size z	80	80	mm
Pick activation	0,1	0,1	s

xx0700000408

Item	The name of an item as specified in the PickMaster project. See *Product/ Pallet/Sheet on page 51*.
Property	The property of an item. For description of the parameters, see table below.
Current	Specifies the actual current value of the property.
Default	Specifies the default value of the property as it was in the PickMaster project when it was downloaded to the controller.
Reset All	All items and work areas are set to their respective default value.

The following table describes the **Property** parameters:

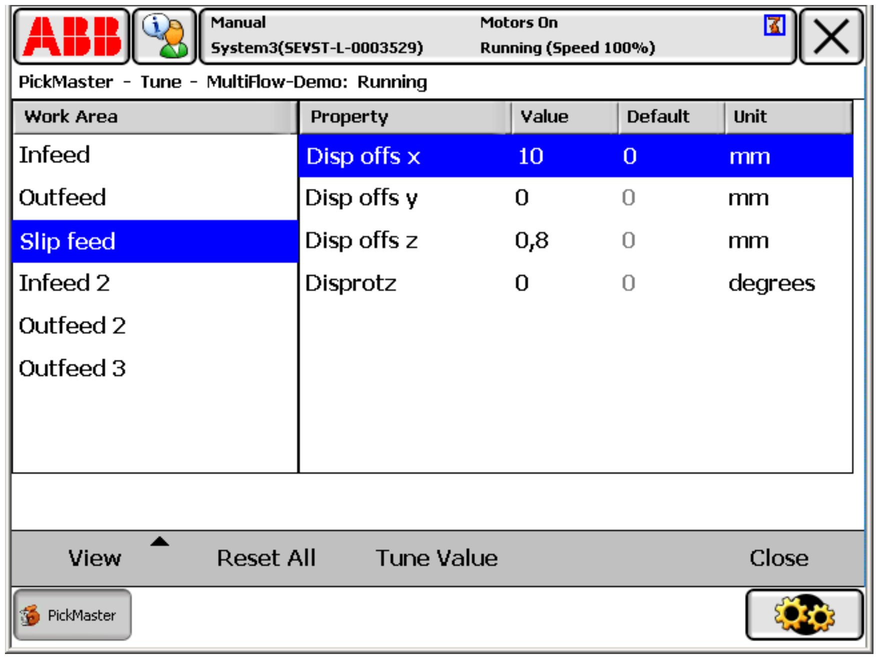
Parameter	Description
Speed	Specifies the maximum speed for an item.
Ori speed	Specifies the maximum orientation speed for an item.
Acceleration	Specifies the maximum acceleration/deceleration for an item.
Pick time	Specifies the time the robot stays at the target position when picking an item.

Continues on next page

Parameter	Description
Place time	Specifies the time the robot stays at the target position when placing an item.
Size Z	Specifies the height of the item. When updating the item height, the height of pick and place positions for next items to be handled will be affected accordingly. Always when updating the item height, the operator is asked if the new value shall be applied also to previously placed layers. If the answer is yes, product place positions will be affected by height updates of all previously placed items in lower layers. If the answer is no, place positions will only be affected by the height updates of new items to be placed. Pick positions will never be affected.
Pick Activation	Specifies the time for pick activation.
Place Activation	Specifies the time for place activation.

Work Area

The illustration shows an example of the Tune window when six work areas are defined, and the properties that can be tuned for each work area.



PickMaster - Tune - MultiFlow-Demo: Running

Work Area
Infeed
Outfeed
Slip feed
Infeed 2
Outfeed 2
Outfeed 3

Property	Value	Default	Unit
Disp offs x	10	0	mm
Disp offs y	0	0	mm
Disp offs z	0,8	0	mm
Disprotz	0	0	degrees

xx0700000409

Work area	The name of a work area specified in the PickMaster project. See *Feeder on page 154*.
Property	The property of a work area. For description of the parameters, see table below.
Value	Specifies the actual current value of the property.
Default	Specifies the default value of the property as it was in the PickMaster project when it was downloaded to the controller.

The following table describes the **Property** parameters:

Parameter	Description
Disp offs x	Displacement of the work area in x-direction relative the work object.
Disp offs y	Displacement of the work area in y-direction relative the work object.

Continues on next page

Parameter	Description
Disp offs z	Displacement of the work area in z-direction relative the work object.
Disprotz	Displacement angle of the work area in the z-direction.

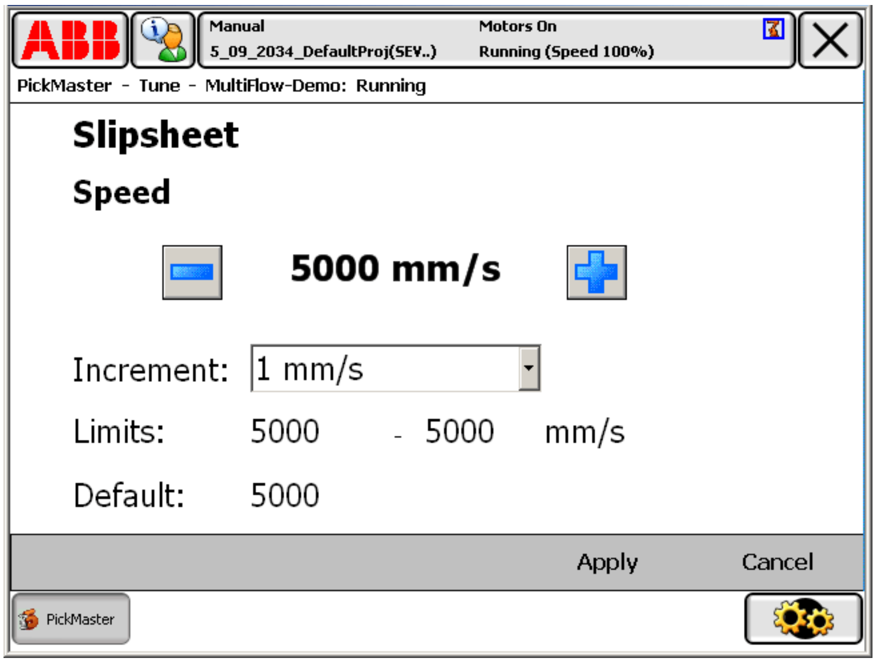
How to proceed

This section describes how to proceed with tuning of an item and a work area.

Tuning an item

	Action	Note
1.	On the **Tune** menu, tap **View**.	
2.	Tap **Item**.	
3.	In the **Item** list, tap the item to tune.	
4.	In the **Property** list, tap the item property to tune.	
5.	On the **Tune** menu, tap **Tune Value**.	The Item Tune Value window appears. See illustration below this procedure.
6.	In the **Increment** drop-down combo box, select the size of increment.	The increment specifies the value that will be added to/subtracted from the item property for each time you tap the **+** or **-** button.
7.	Tap **Apply**.	

The following figure illustrates tuning of an item.


Manual
5_09_2034_DefaultProj(SEV..)
Motors On
Running (Speed 100%)
PickMaster - Tune - MultiFlow-Demo: Running
Slipsheet
Speed
5000 mm/s
Increment: 1 mm/s
Limits: 5000 - 5000 mm/s
Default: 5000

xx0700000411

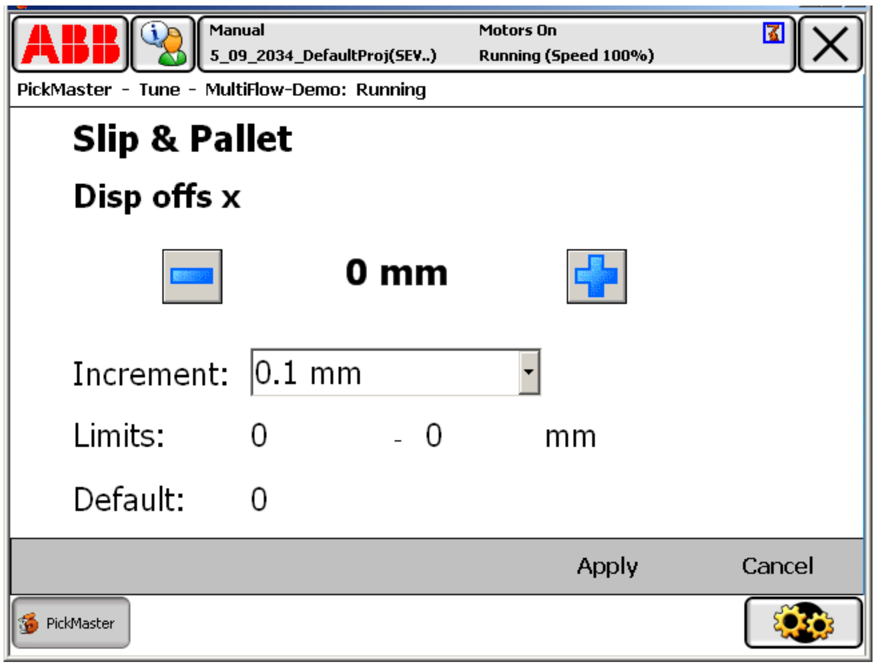
Tuning a work area

	Action	Note
1.	On the **Tune** menu, tap **View**.	
2.	Tap **Work Area**.	
3.	In the **Work Area** list, tap the work area to tune.	

Continues on next page

	Action	Note
4.	In the **Property** list, tap the work area property to tune.	
5.	On the **Tune** menu, tap **Tune Value**.	The **Work Area Value** window appears. See illustration below this procedure.
6.	In the **Increment** drop-down combo box, select the size of increment.	The increment specifies the value that will be added to/subtracted from the item property for each time you tap the plus or minus button.
7.	Tap **Apply**.	

The following figure illustrates tuning of a work area.


Manual
5_09_2034_DefaultProj(SEV..)
Motors On
Running (Speed 100%)
PickMaster - Tune - MultiFlow-Demo: Running

Slip & Pallet

Disp offs x

0 mm

Increment: 0.1 mm

Limits: 0 - 0 mm

Default: 0

xx0700000410

6.3 I/O interface

6.3.1. Overview

PickMaster I/O interface

The PickMaster I/O interface is used by external equipment, such as a PLC, to control and supervise the palletizing process. It consists of two parts:

?tThe basic I/O interface defines work area specific signals. It covers the minimum I/O configuration needed to run a PickMaster project.

?tThe extended I/O interface adds optional functionality, for example reporting error events, starting projects and flows, and so on. You can use the extended I/O interface if needed.

6.3.2. Default signals

Introduction to default signals

More than one hundred (100) default signals are installed on every controller with the option *Prepared for PickMaster*. The signals can be used when setting up PickMaster lines and projects.

Configuration and setup

The signals can be selected in the configuration dialogs. As default, the signals are mapped to simulated I/O units, *Pickmaster_Sim1*, *Pickmaster_Sim2*, and *Pickmaster_Sim3*. During commissioning, the signals can be mapped to physical I/O units.

Additional signals are required if you use more than:

?t8 work areas

?t1 robots

?t1 controllers

?t4 flows

Use RobotStudio for:

?tRenaming, reconfiguring, or removing default signals.

?tAdding additional signals.

Description of default signals

The following default signals are installed.

Work areas

Default signals are defined for eight work areas. Signal name prefixes:

?t*pmInfeeder1*

?t*pmInfeeder2*

?t*pmInfeeder3*

?t*pmInfeeder4*

?t*pmOutfeeder1*

?t*pmOutfeeder2*

?t*pmOutfeeder3*

?t*pmOutfeeder4*

?t*pmSlipsheet1*

?t*pmPallet1*

Continues on next page

Continued

Controller system actions

Default signals are defined for controller system actions. Signal name prefix:

?t*pmSystem*

Project handling

Default signals are defined for project handling. Signal name prefix:

?t*pmProject*

Flows

Default signals are defined for flows. Signal name prefixes:

?t*pmFlow*

?t*pmFlow1*

?t*pmFlow2*

?t*pmFlow3*

?t*pmFlow4*

Grippers

Default signals are defined for a gripper. Signal name prefix:

?t*pmGripper1*

Event reporting

Default signals are defined for event reporting. Signal name prefix:

?t*pmEvent*

6.3.3. Basic I/O interface

General

The I/O signals are used by a PLC to:

?tStart and stop jobs on master work areas.

?tControl and supervise the flow of products on work areas.

?tControl and supervise the robot execution on work areas.

?tControl and supervise the status and height of work areas.

I/O signals must be setup for each work area. Some of the signals must be used and others can be used if needed.

How to use the work area signals depends on if the work area will be used as a master or slave.

Master work areas

The following signals must be configured:

?tTarget generation trigger signal (DI)

?tTarget generation product selection (GI)[1]

?tPosition request trigger signal (DO)[1]

The following signals must be setup if simulated target generation is selected for the work area. See *Feeder on page 154*.

?tPosition request trigger signal (DO)

1) The signal is only mandatory if there is more than one operation set in the feeder.

Target generation trigger signal (DI)

The signal is mandatory unless simulated target generation is used.

The signal can also be skipped if palletizing jobs always are to be started using the FlexPendant interface.

A trigger pulse generates the start of a new palletizing/depalletizing job on that work area. A palletizing/depalletizing job is equivalent to one of the operation sets configured for the feeder. Before the signal is pulsed, the flow must be running, the position request trigger signal must be set and the work area must be prepared for the job to be started. For example:

?tWork area is empty and ready to receive products to build a new pallet.

?tWork area is loaded with a pallet to be depalletized.

Default signal, example: *pmOutfeeder1_diTgtGenTrig*.

Target generation product selection (GI)

The signal is mandatory if there is more than one operation set for the work area.

It is used to select among all palletizing jobs (that is, operation sets) configured for the work area. The signal is set to the product I/O value for the selected operation set. It must be set before the target generation trigger signal is pulsed.

Default signal, example: *pmOutfeeder1_giProdSel*.

Target generation format selection (GI)

The signal has no function and is not required.

Continues on next page

Continued

Target generation start layer count (GI)

The signal is optional. It is used when restarting an unfinished job. The signal is set to the number of full layers on the stack, including the pallet (if defined in the pallet pattern) and slip sheets. It must be set before the target generation trigger signal is pulsed.

Default signal, example: *pmOutfeeder1_giStartLayerCount*.

Target generation start product count (GI)

The signal is optional. It is used when restarting an unfinished job. The signal is set to the number of products on the top layer off the stack. If the top layer is full, the signal is set to zero. It must be set before the target generation trigger signal is pulsed.

Default signal, example: *pmOutfeeder1_giStartProdCount*.

Position request trigger signal (DO)

The signal is mandatory if simulated target generation is used and highly recommended otherwise.

The signal is set by the controller when it is ready to start a new palletizing job (that is, operation set)on that work area. This will happen:

?tWhen the corresponding flow is started.

?tWhen an operation set is completed on the work area.

?tWhen an operation set is finished using the robot execution signal.

?tAs a result of a flow recovery action.

A new operation set cannot be started until the signal is set and the work area is prepared for palletizing (the work area is empty and ready to receive new products from the slave infeeders) or depalletizing (the work area is loaded with a new pallet of products). The signal is reset when the target generation trigger signal is pulsed or if the flow is stopped.

Default signal, example: *pmOutfeeder1_doPosReqTrig*.

Position request product selection (GO)

The signal has no function and is not required.

Position request format selection (GO)

The signal has no function and is not required.

Position request requesting master (GO)

The signal has no function and is not required.

Position available (DO)

The signal is optional.

The signal indicates if target positions can be received in RAPID to be executed by the robot. Using the *Robot execution* signal affects the output of the *Position available* signal.

If the robot execution signal is not defined, *Position available* is set after:

?tA new operation set is generated by the PLC.

?tAny operation, except the last one of an operation set, is performed by the robot.

Continues on next page

If the robot execution signal is not defined, the *Position available* signal is reset after any operation is received in RAPID.

If the robot execution signal is defined, the *Position available* signal is set after:

?tA new operation set is generated and the robot execution signal is set by the PLC.

?tAny operation, except the last one of an operation set, is performed by the robot, the robot execution signal is reset and then set again by the PLC.

If the robot execution signal is defined, the *Position available* signal is reset after

?tAny operation is received in RAPID.

?tThe robot execution signal is reset to finish an uncompleted operation set.

Default signal, example: *pmOutfeeder1_doPosAvail*.

Queue empty (DO)

The signal is optional.

The signal indicates if there are targets generated that yet has not been received in RAPID.

After the last target is received for an operation set, the signal goes to one. Note, the signal will go high before the movements is finished, that is the last products might not have been placed/picked.

Default signal, example: *pmOutfeeder1_doQueueEmpty*.

Operation set complete (DO)

The signal is optional.

The signal is set after all products in the operation set have been placed/picked by the robot on/from the work area. The signal is reset when the target generation trigger signal is pulsed.

Default signal, example: *pmOutfeeder1_doOpSetCompl*.

Execution state (GO)

The signal is optional.

The signal indicates the runtime state of the work area.

I/O value	Description
0	A project using this work area is not running.
1	Work area is running.
2	Work area has an error. Flow recovery is required to recover from the error.
3	Work area has a response error, that is, the PLC has generated wrong targets. A generation of correct targets is required to recover from the error.

Default signal, example: *pmOutfeeder1_goExecState*.

Continues on next page

Continued

Height state (GO)

The signal is optional.

The signal indicates the current height of the work area. The signal must have a bit length of at least three to represent the five possible states.

I/O value	Description
0	Full height. The height is equal to the setting of **Full height**, see *Robot Path Height on page 161*.
1	Active height. An operation set is executed and the height is updated after each pick or place until the operation set is completed.
2	Latest height. The height is equal to the final height of the latest run operation set.
3	Empty height. The height is equal to the setting of **Empty height**, see *Robot Path Height on page 161*.
4	Value. The height is set to a value with the RAPID function `PmSetDefaultHeight`.

Default signal, example: *pmOutfeeder1_goHeightState*.

Layer count (GO)

The signal is optional.

The signal value indicates the number of full layers on the work area.

Default signal, example: *pmOutfeeder1_goLayerCount*.

Product count (GO)

The signal is optional.

The signal value indicates the number of products in the top layer. If the top layer is full, the signal is set to zero.

Default signal, example: *pmOutfeeder1_goProductCount*.

Stop job (DI)

The signal is optional.

The signal is used to stop the currently ongoing job before it is completed. The flow must be running before a job can be stopped. The job is stopped by pulsing the signal.

The job will stop as soon as any currently ongoing or pending pick-places cycle is completed or cancelled. Pending position requests on slaves will be cancelled if no targets are generated before the slave's stop job timeout has passed. For more information on stop job timeout, see *Feeder on page 154*.

If the job is waiting on a slave that is running out of products, stop job can be used to finish the job without running any further pick-place cycles. The job will become stopped after the stop job timeout has passed.

Default signal, example: *pmOutfeeder1_diStopJob*.

Continues on next page

Robot execution (DI)

The signal is optional.

The signal is used to control whether the robot is allowed to approach the work area or not. If the signal is reset, the RAPID execution will not pass the instruction `PmGetTarget` until the signal is set. After an operation is performed by the robot, the signal must be reset and then set again to allow the robot to approach the work area the next time.

The signal can also be used to finish the current operation set before all operations have been completed. If the signal is reset, the remaining targets will be removed.

Default signal, example: *pmOutfeeder1_diRobotExec*.

Redo search (DI)

The signal is optional.

The signal is used with operation sets having stack search activated.

Stack search is normally not used for a master work area.

If the signal is pulsed, the next operation will start with a search movement from the top of the stack. The signal can be used after adding new items on a stack. To affect the next approach to the work area, the signal must be pulsed before the robot receives the operation in RAPID.

Default signal, example: *pmOutfeeder1_diRedoSearch*.

Slave work areas

The following signals must be setup:

?tTarget generation trigger signal (DI)

?tTarget generation product selection (GI)^2

?tTarget generation format selection (GI)^3

?tPosition request trigger signal (DO)

?tPosition request product selection (GO)^2

?tPosition request format selection (GO)^3

The following signals must be setup if simulated target generation is selected for the work area. See *Feeder on page 154*.

?tPosition request trigger signal (DO)

?tPosition request product selection (GO)^2

?tPosition request format selection (GO)^3

Continues on next page

2) Only mandatory if there is more than one item.

3) Only mandatory if there is more than one format for the same item.

Position request trigger signal (DO)

The signal is mandatory.

The signal is set by the controller when one of the corresponding master work areas requests a format on this slave work area from the PLC. The requested format must be defined as an operation set in the feeder. The request occurs after the previous pick and place cycle for that flow has finished.

If the flow uses early request, then the request will occur in advance, before the robot has finished the previous cycle. Early request will decrease cycle times if the same flow is run in consecutive pick place cycles. The signal is reset when the target generation trigger signal is pulsed.

Default signal, example: *pmInfeeder1_doPosReqTrig*.

Position request product selection (GO)

The signal is mandatory if there is more than one item.

The signal I/O value specifies the requested product when the position request trigger signal is set.

Default signal, example: *pmInfeeder1_goProdSel*.

Position request format selection (GO)

The signal is mandatory if there is more than one format for the same item.

The signal I/O value specifies the requested format when the position request trigger signal is set.

Default signal, example: *pmInfeeder1_goFormSel*.

Position request requesting master (GO)

The signal is optional.

The signal value indicates the requesting master work area. The I/O value of the work area is configured in the Work Area I/O Settings editor. For more information on the Work Area I/O Settings, see *Edit Detailed Signals for Feeder on page 157*.

Default signal, example: *pmInfeeder1_goReqMaster.*

Target generation trigger signal (DI)

The signal is mandatory unless simulated target generation is used.

A trigger pulse indicates that a previously requested format is now available for the work area to be handled by the robot.

Default signal, example: *pmInfeeder1_diTgtGenTrig*.

Target generation product selection (GI)

The signal is mandatory if there is more than one item.

The signal specifies the product I/O value of the available format when the target generation trigger signal is pulsed.

Default signal, example: *pmInfeeder1_giProdSel*.

Continues on next page

Target generation format selection (GI)

The signal is mandatory if there is more than one format for the same item.

The signal specifies the format I/O value of the available format when the target generation trigger signal is pulsed.

Default signal, example: *pmInfeeder1_giFormSel*.

Target generation start layer count (GI)

The signal has no function and is not required.

Target generation start product count (GI)

The signal has no function and is not required.

Position available (DO)

The signal is optional.

The signal indicates if target positions can be received in RAPID to be executed by the robot. Using the *Robot execution* signal affects the output of the *Position available* signal.

If the *Robot execution* signal not is defined, *Position available* is set after:

?tA new operation set is generated by the PLC.

?tAny operation, except the last one of an operation set, is performed by the robot.

If the *Robot execution* signal is not defined, *Position available* is reset after any operation is received in RAPID.

If the *Robot execution* signal is defined, *Position available* is set after:

?tA new operation set is generated and the *Robot execution* signal is set by the PLC.

?tAny operation, except the last one of an operation set, is performed by the robot, the *Robot execution* signal is reset and then set again by the PLC.

If the *Robot execution* signal is defined, *Position available* is reset after:

?tAny operation is received in RAPID.

?tThe *Robot execution* signal is reset to finish an uncompleted operation set.

Default signal, example: *pmInfeeder1_doPosAvail*.

Queue empty (DO)

The signal is optional.

The signal indicates if there are targets generated that yet has not been received in RAPID.

After the last target is received for an operation set, the signal goes to one. Note, the signal will go high before the movements have finished, that is the last products might not yet have been picked/placed.

Default signal, example: *pmInfeeder1_doQueueEmpty*.

Operation set complete (DO)

The signal is optional.

The signal is set after all products in the operation set have been placed/picked by the robot on/from the work area. The signal is reset when the target generation trigger signal is pulsed.

Default signal, example: *pmInfeeder1_doOpSetCompl*.

Continues on next page

Execution state (GO)

The signal is optional.

The signal indicates the runtime state of the work area.

I/O value	Description
0	A project using this work area is not running.
1	Work area is running.
2	Work area has an error. Flow recovery is required to recover from the error.
3	Work area has a response error, that is, the PLC has generated wrong targets. A generation of correct targets is required to recover from the error.

Default signal, example: *pmInfeeder1_goExecState*.

Height state (GO)

The signal is optional.

The signal indicates the current height of the work area. The signal must have a bit length of at least three to represent the five possible states.

I/O value	Description
0	Full height. The height is equal to the setting of **Full height**, see *Robot Path Height on page 161*.
1	Active height. An operation set is being executed and the height is updated after each pick or place until the operation set is completed.
2	Latest height. The height is equal to the final height of the latest run operation set.
3	Empty height. The height is equal to the setting of **Empty height**, see *Robot Path Height on page 161*.
4	Value. The height is set to a value with the RAPID function PmSetDefaultHeight.

Default signal, example: *pmInfeeder1_goHeightState*.

Layer count (GO)

The signal is optional.

The signal value indicates the number of full layers on the work area. For example, for a pallet stack or a slip sheet stack.

Default signal, example: *pmIntfeeder1_goLayerCount*.

Product count (GO)

The signal is optional.

The signal value indicates the number of products in the top layer. If the top layer is full, the signal is set to zero.

Default signal, example: *pmInfeeder1_goProductCount*.

Stop job (DI)

The signal has no function and is not required.

Continues on next page

Robot execution (DI)

The signal is optional.

The signal is used to control whether the robot is allowed to approach the work area or not. If the signal is reset, the RAPID execution will not pass the instruction `PmGetTarget` until the signal is set. After an operation is performed by the robot, the signal must be reset and then set again to allow the robot to approach the work area next time.

The signal can also be used to finish the current operation set before all operations have been completed. If the signal is reset, the remaining targets will be removed.

Default signal, example: *pmInfeeder1_diRobotExec*.

Redo search (DI)

The signal is optional.

The signal is used with operation sets having stack search activated.If the signal is pulsed, next operation will start with a search movement from the top of the stack. The signal can be used after adding new items on a stack. To affect the next approach to the work area, the signal must be pulsed before the robot receives the operation in RAPID.

Default signal, example: *pmInfeeder1_diRedoSearch*.

6.3.4. Extended I/O interface

Controller system handling

A number of default system signals are installed to handle the controller system, for example to set the controller in motors on state.

See *Technical reference manual - System parameters*, the topic *I/O*, for descriptions of system inputs and outputs.

System inputs

System input	Description
pmSystem_diLoadStartX	Load and start the PmProjMgr module for motion task T_ROBX.
pmSystem_diStart	Start RAPID execution.
pmSystem_diStop	Stop RAPID execution.
pmSystem_diStartMain	Start RAPID execution from Main.
pmSystem_diMotorsOn	Set motors on.
pmSystem_diResetEstop	Confirm reset of emergency stop.

System outputs

System output	Description
pmSystem_doCycleOn	Robot program is executing.
pmSystem_doMotorOn	Motors on state.
pmSystem_doRunchOk	Run chain is closed.
pmSystem_doEmStop	Emergency stop state.
pmSystem_doAutoOn	Automatic mode is used.

Project handling

It is possible to start, halt, restart, stop, and supervise projects.

Default signals

The following default signals must be used for project handling:

Project signal	Description
pmProject_goCurrent	The current project.
pmProject_goStatus	The status of current project.
pmProject_diStop	Stop current project.
pmProject_diStart	Start selected project.
pmProject_giSelection	Project selector.
pmProject_diSetDefaultheight	Set default height for a work area.
pmProject_giDefaultHeight	Default height selector.
pmProject_giDefHeightWaSel	Work area selector for setting the default height.
pmProject_diNoWait	PickWare internal use.
pmProject_diNoWait	PickWare internal use.

Continues on next page

Project status values

The current status of the project is reflected by the signal *pmProject_goStatus*.

I/O value	Description
0	Project is stopped.
1	Project is stopping.
2	Project is starting.
3	Project is running.
5	Project in error state.

Default height values

The following selections are supported when setting the default height.

I/O value	Description
0	Full
1	Latest
2	Empty
3	Value
4	Standard, that is, as configured in the **Robot Path Height**. See *Robot Path Height on page 161*.

Starting a project

	Action
1	Ensure that the previous project is stopped, that is he signal *pmProject_goStatus* is 0.
2	Switch to motors on state using the controller system signals. See *Technical reference manual - System parameters*, the topic *I/O*.
3	Pulse the system input *pmSystem_diLoadStart1* to load and start the project manager module. In a MultiMove system, pulse the corresponding signals, *pmSystem_diLoadStartX*, to load and start the other motion tasks.
4	Wait until robot program has started, that is when *pmSystem_doCycleOn* is set.
5	Set *pmProject_giSelection* to select the project to run. The I/O value for the project is set in the **Project I/O value editor**. See *Project Manager on page 98*.
6	Pulse *pmProject_diStart* to start the project.
7	Wait until the project has started, that is when *pmProject_goStatus* goes to 3.

Stopping robot program - halting project

	Action
1	Pulse *pmSystem_diStop* to stop the robot program.
2	Wait until the robot program is stopped, which occurs when *pmSystem_doCycleOn* is reset.

Restarting robot program - restarting project

	Action
1	Switch to motors on state using the controller system signals. See *Technical reference manual - System parameters*, the topic *I/O*.

Continues on next page

	Action
2	Pulse *pmSystem_diStart* to start the robot program.
3	Wait until the robot program is started, which occurs when *pmSystem_doCycleOn* is set.

Stopping current project

	Action
1	Pulse *pmSystem_diStop* to stop the robot program.
2	Wait until the robot program is stopped, which occurs when *pmSystem_doCycleOn* is reset.
3	Pulse *pmProject_diStop* to stop the project.
4	Wait until project is stopped, which occurs when *pmProject_goStatus* goes to 0.

Set a new default height for a work area

Setting a new default height is a possibility to save cycle time without decreasing the margins for collisions, especially if the project consists of many work areas and flows.

For an outfeeder the default height can be set to *Empty* after a finished stack has been unloaded. This may allow the robot to make lower intermediate movements when passing over the outfeeder next time and thus saving cycle time.

For an infeeder the default height can be set to *Full* before a new stack is loaded. This will force the robot to make intermediate movements with enough height when passing over the work area.

The new default height is active until new targets have been generated (or after a new default height is set).

	Action
1	Set *pmProject_giDefaultHeight* to select the new default height.
2	Set *pmProject_giDefHeightWaSel* to select the work area. The I/O value of the work area is configured in the **Work Area I/O Settings** editor. See *Edit Detailed Signals for Feeder on page 157*.
3	Pulse *pmProject_diSetDefaultHeight*.
4	Wait until the default height is updated, that is, the work area height state GO signal is updated to reflect the change. **Note!** If the current height state is 1, which means active height, the height state will not be updated until the last target of the current operation set is picked/placed.

Flow handling

It is possible to start, stop, and supervise, and recover flows.

Default signals

The following default signals must be used for flow handling:

Flow signal	Description
pmFlow_diStart	Start selected flow.
pmFlow_diStop	Stop selected flow.
pmFlow_diRecover	Recover the selected flow with the selected recover action on the selected work area.

Continues on next page

Continued

Flow signal	**Description**
pmFlow_giSelection	Flow selector.
pmFlow_giStopOption	Set stop option.
pmFlow_giRecoverAction	Set recover action.
pmFlow_giWaRecoverSelection	Select work area for recover action.
pmFlowX_goStatus	Status of flow X (X=1,2,3,…).

Flow status values

The statuses of the flows are reflected by the *pmFlowX_goStatus* signals.

The status signal for the flow is setup in the **Flow editor**. See *Flow on page 184*.

If a flow goes to error state, you can do flow recovery from the I/O Interface or the PickMaster FlexPendant interface.

I/O value	**Description**
0	Flow is stopped.
1	Flow is running.
2	Flow is stopping after current pick place cycle.
3	Flow is stopping after current layer.
4	Flow is stopping after current pallet/operation set.
5	Flow in error state.

Flow stop options

Stop option is specified with the signal *pmFlow_giStopOption*.

I/O value	**Description**
1	Finish cycle.
2	Finish layer.
3	Finish pallet/operation set.

Flow recover actions

A flow recover action is specified with the signal *pmFlow_giRecoverAction*.

I/O value	**Description**
1	Continue.
2	Restart layer.
3	Next pallet.
4	Redo last pick.

Starting or restarting a flow

	Action
1	Ensure that current project is running. The *pmProject_goStatus* signal is set to 3.
2	Set *pmFlow_giSelection* to select flow. The I/O value for the flow is set in the **Flow editor**. See *Flow on page 184*.
3	Pulse *pmFlow_diStart* to start the project.
4	Wait until the flow is started, which occurs when the flow's status signal, *pmFlowX_goStatus*, goes to 1.

Continues on next page

Stopping a flow

	Action
1	Set *pmFlow_giSelection* to select flow. The I/O value for the flow is set in the **Flow editor**. See *Flow on page 184*.
2	Set *pmFlow_giStopOption* to select stop option.
3	Pulse *pmFlow_diStop* to stop the project.
4	Wait until the flow is stopped, which occurs when the flow’s status signal, *pmFlowX_goStatus*, goes to 0.

Stopping a flow immediately

When stopping a flow immediately it will go to error state and the corresponding palletizing job will be stopped.

A flow recover action must be specified before restarting the flow.

If this is the only flow or if the robot is currently working on this flow, then the robot will also stop immediately. However, if multiple flows are running, the robot can continue working on the other flows.

To force a stop of the robot, this sequence can be preceded by a *Stopping robot program - halting project* action. This will require a *Restarting robot program - restarting project* action after selecting recover action and restarting the flow.

	Action
1	Set *pmEvent_giErrorSource* to select the flow's master work area and all slave work areas. See *Events on page 144*, on how to setup event signals for the robot controller.
2	Pulse *pmEvent_diTrigger* to immediately generate an error on the flow and its work areas.
3	Wait until the flow is in error state, which occurs when the flow's status signal, *pmFlowX_goStatus*, goes to 5.

Recovering a flow

If a flow has entered error state, the *pmFlowX_goStatus* signal is set to 5. Use this procedure to recover the flow.

	Action
1	Set *pmFlow_giSelection* to select flow. The I/O value for the flow is configured in the **Flow editor**. See *Flow on page 184*.
2	Set *pmFlow_giRecoverAction* to select recover action.
3	Set *pmFlow_giWaRecoverSelection* to select which work area the recover action shall be applied for. The signal must not be set for the recover actions *Continue* and *Redo last pick*. The selected work area must be included in the selected flow, normally the master work area is selected. The I/O value of the work area is configured in the **Work Area I/O Settings editor**. See *Edit Detailed Signals for Feeder on page 157*.
4	Pulse *pmFlow_diRecover* to recover the flow.
5	An elog message will be logged from the PickMaster RAPID application. The message contains information on the expected state of the robot tool and work areas before restarting the flow (see *PmSetRecoverAction - Set flow recover action on page 298*). The flow is now prepared for being restarted. **Note:** the error state of the flow indicated by *pmFlowX_goStatus* will not change until the flow is restarted.

Continues on next page

Continued

Event and error reporting

It is possible to report events and errors for work areas, robots, and controllers that affect the runtime operation. See *Events on page 144*.

Default signals

The following default signals can be used for event and error reporting.

Signal	Description
pmEvent_diTrigger	Generate an event.
pmEvent_giErrorSource	Select error source(s).
pmEvent_giMessage	Select elog message.

Example, Report an error for a work area and log an elog message

	Action
1	Set the proper bit for *pmEvent_giErrorSource* to select a work area. The bit representing the work area is set in the **Event settings** tab of the **Controller Properties**. See *Events on page 144*.
2	Set *pmEvent_giMessage* to select a message. Values that represent different messages are set in the **Message Settings**, see section *Messages on page 147*.
3	Pulse *pmEvent_diTrigger* to generate the error and the elog message.
4	Wait until the work area enters error state, which occurs when the execution state signal of the work area, *pmFlowX_goExecState*, gets the value 2.

Example, Log an elog message

	Action
1	Set the proper bit for *pmEvent_giErrorSource* to select a controller. The bit that represents the controller is set in the **Event settings** tab of the **Controller Properties.** See *Events on page 144*.
2	Set *pmEvent_giMessage* to select a message. Values that represent different messages are set in the **Message Settings**, see section *Messages on page 147*
3	Pulse *pmEvent_diTrigger* to generate the elog message.

Robot tool

The control of the robot tool in runtime operation is integrated in the PickMaster RAPID interface and defined by the project configuration.

Available default signals for tools:

Signal	Description
pmGripper1_goActivators	Activators control
pmGripper1_doActivator1Open	Activator control
pmGripper1_doActivator1Close	Activator control
pmGripper1_doActivator2Open	Activator control
pmGripper1_doActivator2Close	Activator control
pmGripper1_doActivator3Open	Activator control
pmGripper1_doActivator3Close	Activator control
pmGripper1_doActivator4Open	Activator control
pmGripper1_doActivator4Close	Activator control
pmGripper1_diActivator1Opened	Activator status

Continues on next page

Continued

Signal	Description
pmGripper1_diActivator1Closed	Activator status
pmGripper1_diActivator2Opened	Activator status
pmGripper1_diActivator2Closed	Activator status
pmGripper1_diActivator3Opened	Activator status
pmGripper1_diActivator3Closed	Activator status
pmGripper1_diActivator4Opened	Activator status
pmGripper1_diActivator4Closed	Activator status
pmGripper1_diPartCheck1	Part check status
pmGripper1_diPartCheck2	Part check status
pmGripper1_diPartCheck3	Part check status
pmGripper1_diPartCheck4	Part check status
pmGripper1_diPartCheck5	Part check status
pmGripper1_giPartCheck1	Part check status
pmGripper1_goSearchActivate	Search tool activation.
pmGripper1_diSearchStop	Stack search stop trigger.
pmGripper1_goToolEvent1	GO tool event.
pmGripper1_goToolEvent2	GO tool event.
pmGripper1_goToolEvent3	GO tool event.
pmGripper1_goToolEvent4	GO tool event.
pmGripper1_goToolEvent5	GO tool event.
pmGripper1_doToolEvent1	DO tool event.
pmGripper1_diToolEvent1	DI tool event.
pmGripper1_giToolEvent1	GI tool event.
pmGripper1_giToolEvent2	GI tool event.
pmGripper1_giToolEvent3	GI tool event.
pmGripper1_giToolEvent4	GI tool event.
pmGripper1_giToolEvent5	GI tool event.

6.3.5. Timing diagrams for PLC communication

Introduction to timing diagrams

Each timing diagram shows a basic example on the I/O communication between the robot controller and the PLC when running a PickMaster project. The individual updates of different I/O signals are shown related to important events of the palletizing process, for example when a pickup of a format has been completed.

The following events in the palletizing process can be found in the timing diagrams:

Event	Description
Operate infeeder	The RAPID execution has executed the RAPID procedure `Operate` for the infeeder.
Pick	The robot has picked up a complete format in the tool, that is, finished an operation.
Operate outfeeder	The RAPID execution has executed the RAPID procedure `Operate` for the outfeeder.
Place	The robot has placed a complete format on the work area, that is, finished an operation.
Pallet pattern unloaded	A finished pallet pattern leaves the working range of the robot when being transferred from an outfeeder.

See examples:

?t*Example minimum process control on a running flow on page 273*.

?t*Example robot execution control on page 274*.

?t*Example height control of a running flow on page 275*.

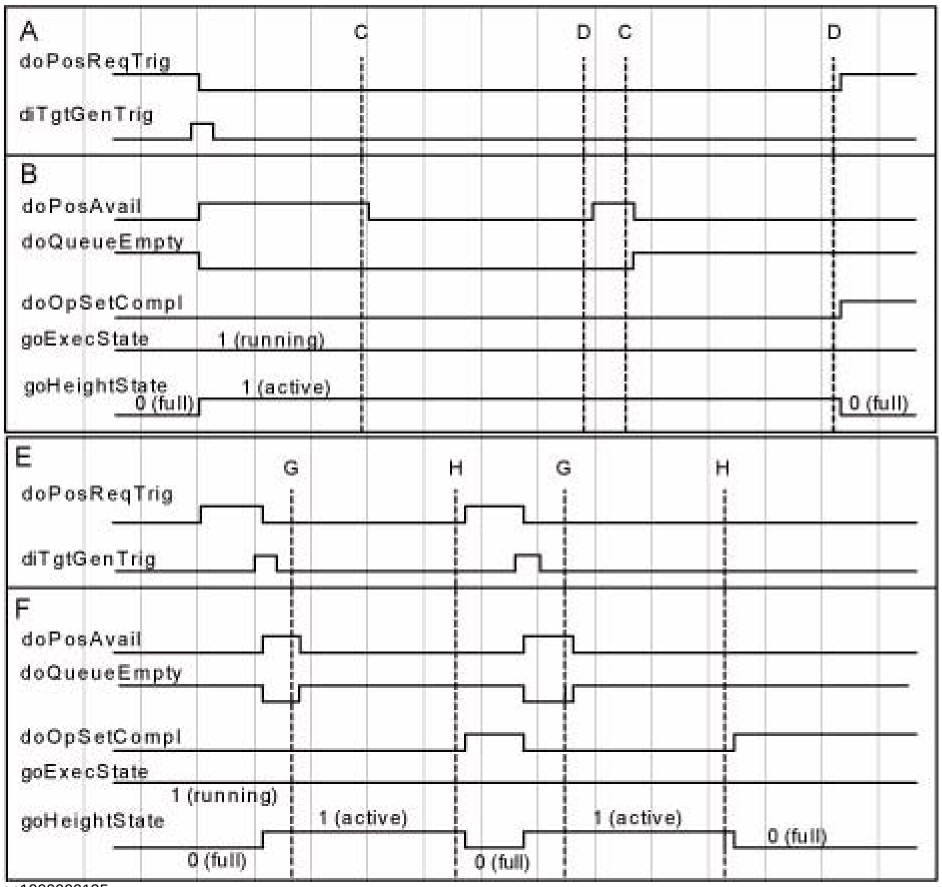
?t*Example flow control on page 276*.

Continues on next page

Example minimum process control on a running flow

Task: Pick single items from infeeder and place pallet pattern with two items on outfeeder.

Settings: Early request, Use concurrency, non-pulsed controller mode.



en1000000195

A	Master outfeeder, process control
B	Master outfeeder, process status
C	Operate outfeeder
D	Place
E	Slave infeeder, process control
F	Slave infeeder, process status
G	Operate infeeder
H	Pick

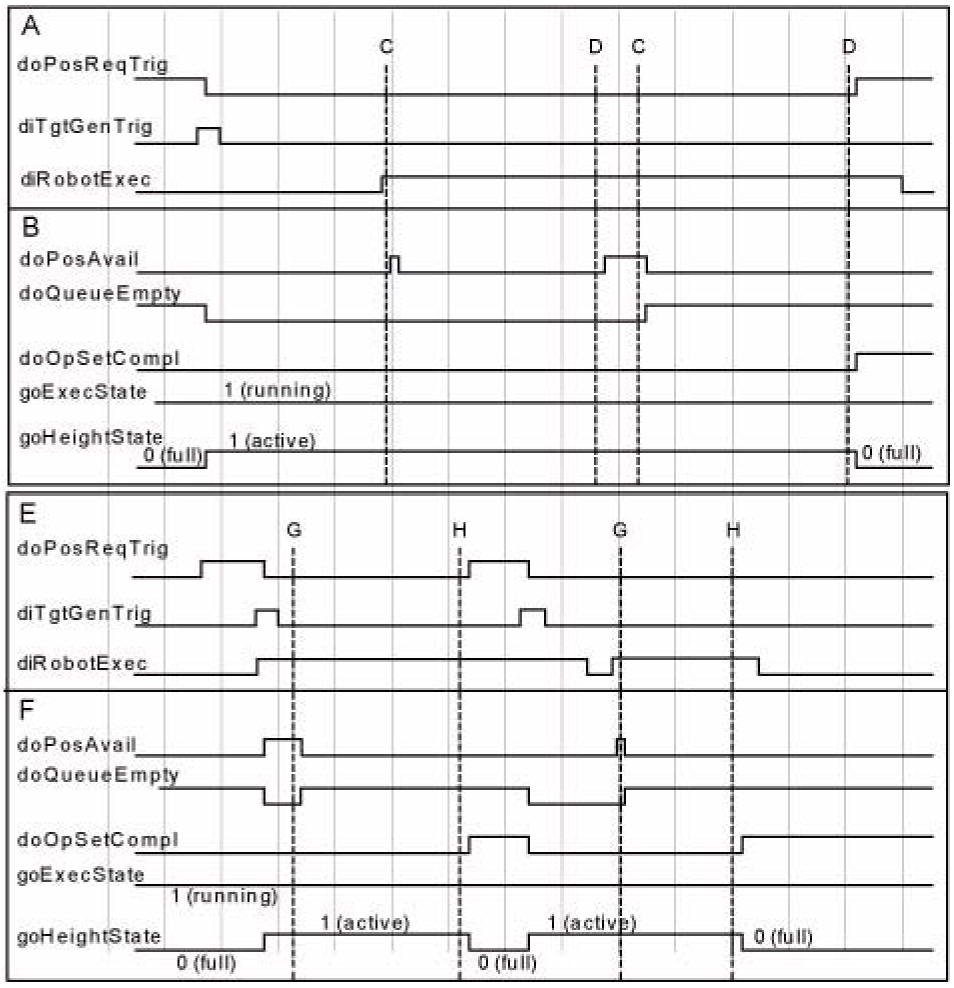
Continues on next page

Continued

Example robot execution control

Task: Pick single items from infeeder and place pallet pattern with two items on outfeeder, control the robot access to work areas.

Settings: Early request, Use concurrency, non-pulsed controller mode, default height *Full* on both infeeder and outfeeder.



en1000000198

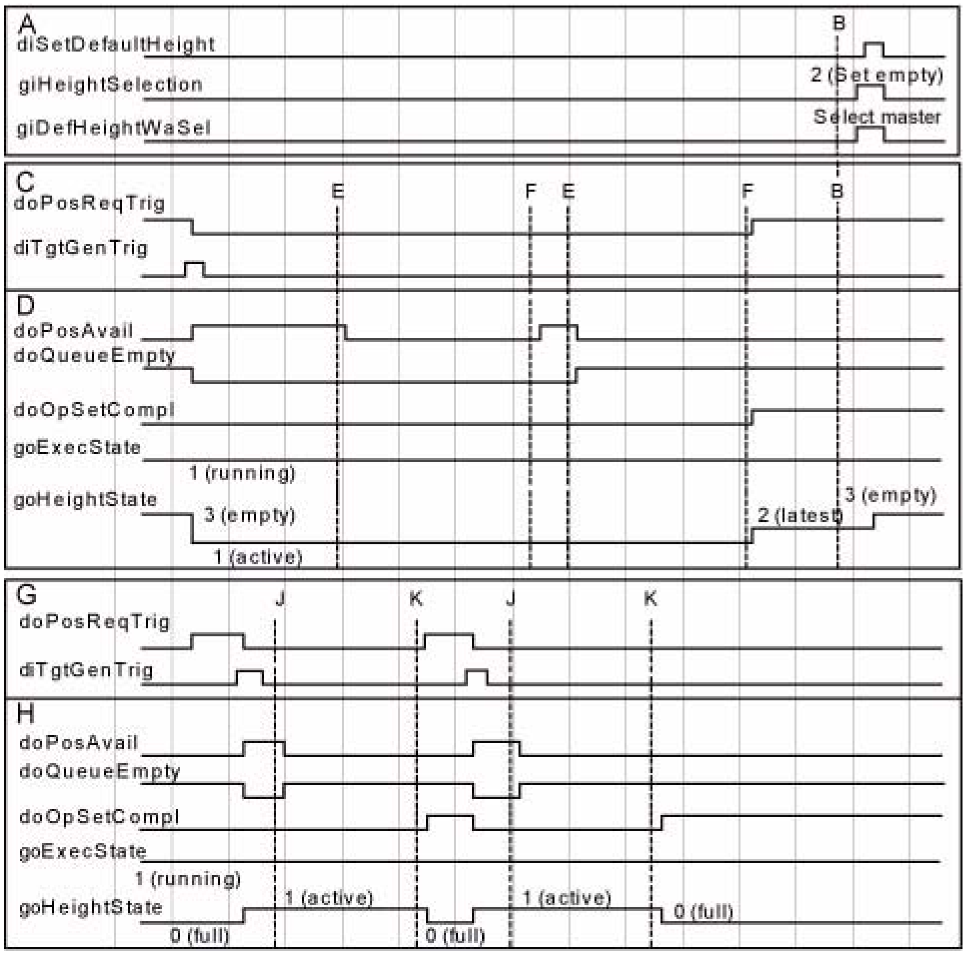
A	Master outfeeder, process control
B	Master outfeeder, process status
C	Operate outfeeder
D	Place
E	Slave infeeder, process control
F	Slave infeeder, process status
G	Operate infeeder
H	Pick

Continues on next page

Example height control of a running flow

Flow task: Pick single items from infeeder and place pallet pattern with two items on outfeeder. Control the height change of the outfeeder caused by unloading the pallet pattern to minimize the cycle time for other flows.

Settings: Early request, Use concurrency, non-pulsed controller mode, default height *Full* on infeeder and *Latest* on outfeeder.



en1000000202

A	Project control, process control
B	Pallet pattern unloaded from outfeeder
C	Master outfeeder, process control
D	Master outfeeder, process status
E	Operate outfeeder
F	Place
G	Slave infeeder, process control
H	Slave infeeder, process status
J	Operate infeeder
K	Pick

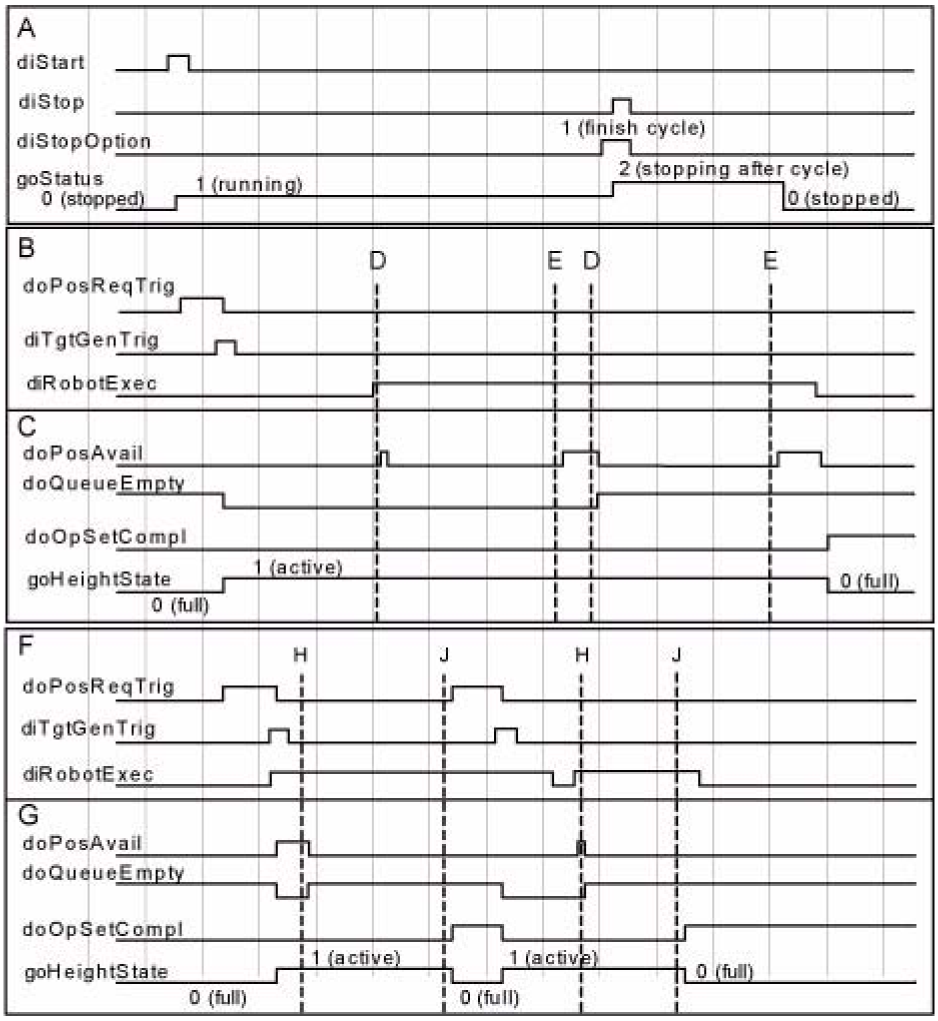
Continues on next page

Continued

Example flow control

Task: Start a flow, pick single items from infeeder and place pallet pattern on outfeeder, stop the flow after next cycle, evacuate the unfinished pallet pattern from the outfeeder.

Settings: Early request, Use concurrency, non-pulsed controller mode, default height *Full* on both infeeder and outfeeder.



en1000000189

A	Flow control, process control. In this example, the *giSelection* signal is constantly set to this flow.
B	Master outfeeder, process control
C	Master outfeeder, process status. In this example, the *goExecState* signal is constantly set to 1 (running).
D	Operate outfeeder
E	Place
F	Slave infeeder, process control
G	Slave infeeder, process status. In this example, the *goExecState* signal is constantly set to 1 (running).
H	Operate infeeder
J	Pick

7 RAPID reference information

7.1. Introduction to RAPID reference information

Structure of this chapter

This chapter describes the RAPID instructions, functions, and data types that are specific for PickMaster.

7.2 Instructions

7.2.1. PmAckTarget - Acknowledge a target

Usage

`PmAckTarget` is used to acknowledge a target.

Basic examples

```
IF status = OK THEN
  PmAckTarget Wa, Target, PM_ACK;
ELSE
  PmAckTarget Wa, Target, PM_NACK;
ENDIF
```

Arguments

```
PmAckTarget Wa Target Status
```

`Wa`

Data type: `pm_wadescr`

Contains a reference to a work area.

`Target`

Data type: `pm_targetdata`

The target that is acknowledged.

`Status`

Data type: `pm_acktype`

The acknowledge status.

Predefined data

The acknowledge status, used in argument `Status`, can be one of the following:

Constant	Description
PM_ACK	The target is acknowledged as used.
PM_NACK	The target is acknowledged as not used.
PM_LOST	If the target is acknowledged as lost.

Syntax

```
PmAckTarget
  [ Wa ':=' ]  < expression (IN) of pm_wadescr > ','
  [ Target ':=' ]  < expression (IN) of pm_targetdata > ','
  [ Status ':=' ]  < expression (IN) of pm_acktype > ';'
```

Continues on next page

Continued

Related information

For information about	See
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 363*.
The data type `pm_targetdata`	*pm_targetdata - PickMaster target data on page 359*.
The data type `pm_acktype`	*pm_acktype - PickMaster target acknowledge type on page 328*.

7.2.2. PmCalcArmConf - Calculates the arm configuration

Usage

`PmCalcArmConf` is used to calculate a suitable arm configuration for a `robtarget`, that is the `robconf` component of the robtarget. Some switches can be selected to optimize the resulting arm configuration, for example for a robot of a certain type. A maximum and minimum angle can be set up for one axis. The resulting arm configuration will also depend on the initial settings of `robconf`.

Basic example

```
PmCalcArmConf RobTgtPoint,TargetTool,TargetWobj\cf6\MaxAngle:=180
     \MinAngle:=-180;
```

Arguments

```
PmCalcArmConf RobTgt Tool Wobj [\cf1] | [\cf4] | [\cf6] | [\TypeB1]
     [\MaxAngle] [\MinAngle]
```

`RobTgt`

Robot target

Data type: `robtarget`

The robot target whose arm configuration will be calculated.

`Tool`

Tool

Data type: `tooldata`

The tool used for calculation of the robot arm configuration.

Wobj

Work object

Data type: `wobjdata`

The work object (coordinate system) to which the robot position is related.

`[\cf1]`

Data type: `switch`

An arm configuration is calculated where the axis 1 angle is limited by the arguments `MaxAngle` and `MinAngle`. A solution closer to +45 degrees is preferred for axis 1. A solution close to the input arm configuration is preferred for the other axes.

`[\cf4]`

Data type: `switch`

An arm configuration is calculated where the axis 4 angle is limited by the arguments `MaxAngle` and `MinAngle`. A solution closer to +45 degrees is preferred for axis 4. A solution close to the input arm configuration is preferred for the other axes.

Continues on next page

`[\cf6]`

Data type: `switch`

An arm configuration suitable for a 4 axes palletizer robot or a 6 axes bending backwards robot is calculated. The axis 6 angle is limited by the arguments `MaxAngle` and `MinAngle`. A solution closer to +45 degrees is preferred for axis 6. A solution close to the input arm configuration is preferred for the other axes.

`[\TypeB1]`

Data type: `switch`

An arm configuration suitable for a parallel rod robot is calculated. The axis 6 angle is limited by the arguments `MaxAngle` and `MinAngle`. A solution closer to +45 degrees is preferred for axis 6. A solution close to the input arm configuration is preferred for the other axes.

`[\MaxAngle]`

Maximum angle

Data type: num

Maximum angle allowed for one axis. Which axis is decided by the selection of `cf1`, `cf4`, `cf6` or `TypeB1`.

`[\MinAngle]`

Minimum angle

Data type: num

Minimum angle allowed for one axis. Which axis is decided by the selection of `cf1`, `cf4`, `cf6` or `TypeB1`.

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
PM_ERR_CALCCONF	Failed to calculate arm configuration. A highly complex arm configuration may cause this error.
PM_ERR_AXLIM	Failed to calculate axis limit. The axis angle cannot be calculated due to angle limitations.
PM_ERR_LIM_VALUE	Wrong limitation value. The coordinate is not possible to calculate.

Syntax

```
Instruction
  [ RobTgt ':=' ]  < expression (INOUT) of robtarget > ','
  [ Tool ':=' ]  < expression (IN) of tooldata > ','
  [ Wobj ':=' ]  < expression (IN) of wobjdata >
  [ '\' cf1 ] | [ '\'cf4 ] | [ '\'cf6 ] | [ '\'TypeB1 ]
  [ '\' MaxAngle ':=' < expression (IN) of num >]
  [ '\' MinAngle ':=' < expression (IN) of num >] ';'
```

Continues on next page

Related information

For information about	See
The data type `confdata`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Data types*.
The data type `robtarget`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Data types*.

7.2.3. PmGetFlow - Get flow to execute

Usage

`PmGetFlow` is used to wait until any flow reports that it is ready to be executed. The instruction will return two work area references to the work areas that are ready to be executed within a flow. If several flows are ready to be executed, the process behind the instruction will return references of the highest prioritized flow. If the time-out time is not used, the instruction is blocking until any flow is ready to be executed.

Basic examples

A basic example of the instruction `PMGetFlow` is illustrated below.

See also *More examples on page 284*.

Example 1

```
PROC OperateSequence()
  PmGetFlow waInFeeder, waOutFeeder;
  Operate waInFeeder;
  Operate waOutFeeder;
ENDPROC
```

Arguments

```
PmGetFlow PickWa PlaceWa [\MaxTime] [\TimeFlag]
```

`PickWa`

Data type: `pm_wadescr`

Variable that is updated to refer to the pick work area of the flow that is ready to be executed.

`PlaceWa`

Data type: `pm_wadescr`

Variable that is updated to refer to the place work area of the flow that is ready to be executed.

`[\MaxTime]`

Maximum Time

Data type: `num`

The maximum period of permitted waiting time, expressed in seconds. If this time runs out before the condition is met, the error handler will be called if there is one, with the error code `PM_ERR_TIMEOUT`. If there is no error handler, the execution will be stopped.

`[\TimeFlag]`

Timeout Flag

Data type: `bool`

The output argument that contains the value TRUE if the maximum permitted waiting time runs out before the condition is met. If this argument is included in the instruction, it is not considered an error if the maximum time runs out. This argument is ignored if the `MaxTime` argument is not included in the instruction.

Continues on next page

Program execution

If the programmed condition is not met when executing a `PmGetFlow` instruction, the robot will wait and the time will be supervised. If it exceeds the maximum time value, the program will continue if a `TimeFlag` is specified, or generate an error if it is not specified. If a `TimeFlag` is specified, this will be set to TRUE if the time is exceeded, otherwise it will be set to FALSE.

More examples

More examples of how to use the instruction `PmGetFlow` are illustrated below.

Example 1

```
PROC OperateSequence()
  PmGetFlow waInFeeder, waOutFeeder \MaxTime:=6
       \TimeFlag:=bTimeout;
  IF NOT bTimeout THEN
     Operate waInFeeder;
     Operate waOutFeeder;
  ELSE
     p1 := CRobT(\Tool:=tool0 \WObj:=wobj0);
     MoveL RelTool(p1,100,0,0), v100, fine, tool0;
     MoveL p1, v100, fine, tool0;
  ENDIF
ENDPROC
```

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
`PM_ERR_TIMEOUT`	No flow was ready to be executed within the time-out time.

Syntax

```
PmGetFlow
  [ PickWa ':=' ] < expression (VAR) of pm_wadescr > ','
  [ PlaceWa ':=' ] < expression (VAR) of pm_wadescr >
  [ '\' MaxTime ':=' < expression (IN) of num > ',']
  [ '\' TimeFlag ':=' < variable (VAR) of bool >] ';'
```

Related information

For information about	See
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 363*
The **Robot Flow Configuration**	*Flow on page 184*

7.2.4. PmGetFlowInfo - Get information about a specific flow

Usage

`PmGetFlowInfo` gets information about a flow. The flow must be in the started project.

Basic examples

A basic example of the instruction `PmGetFlowInfo` is illustrated below.

Example 1

```
TRAP TrapStartFlow
  VAR pm_flowinfo FlowInfo;
  VAR num FlowSelection;

  ! Get info from selected flow
  PmGetFlowInfo FlowSelection,FlowInfo;
ENDTRAP
```

Arguments

```
PmGetFlowInfo SelectionNumber | Name FlowInfo
```

`SelectionNumber`

Data type: `num`

The number that maps a specific flow with its signal value.

`Name`

Data type: `string`

The name of the flow in a started project.

`FlowInfo`

Data type: `pm_flowinfo`

Variable that holds the information about the flow.

Program execution

The program will fail with a recoverable error if the flow cannot be found. All other errorsare considered to be fatal.

Continues on next page

More examples

Another example of how to use the instruction `PMGetFlowInfo` is illustrated below.

Example 2

```
TRAP TrapStartFlow
  VAR pm_flowinfo FlowInfo;
  VAR num FlowSelection;

  FlowSelection:=1;
  ! Get info from selected flow
  PmGetFlowInfo FlowSelection,FlowInfo;
  ! Start the selected flow
  PmStartFlow FlowInfo.Name;
  TPWrite "Master work area = "+PmGetWaName (FlowInfo.MasterWa);
ERROR
  ! Continue supervision on recoverable errors
  IF ERRNO=PM_ERR_FLOW_NOT_FOUND THEN
     RETURN;
  ENDIF
ENDTRAP
```

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
`PM_ERR_FLOW_NOT_FOUND`	No flow was found with this selection number or name.
PM_ERR_NO_RUNNING_PROJECT	No running project.

Syntax

```
PmGetFlowInfo
  [SelectionNumber ':=' ] < expression (IN) of num > ','
  |[Name ':=' ] < expression (IN) of string > ','
  [FlowInfo ':=' ] < expression (VAR) of pm_flowinfo > ';'
```

Related information

For information about	See
The data type `pm_flowinfo`	*pm_flowinfo - PickMaster flow information on page 338*.

7.2.5. PmGetLastWa - Get last used work area

Usage

`PmGetLastWa` gets the last used work area. The work area must previous have been set by the instruction `PmSetLastWa`.

Basic examples

A basic example of the instruction `PmGetLastWa` is illustrated below.

Example 1

```
VAR pm_wadescr WorkArea;
! Get last used work area
PmGetLastWa WorkArea;
```

Arguments

```
PmGetLastWa Workarea
```

`Workarea`

Data type: `pm_wadescr`

A descriptor to the last set work area.

Program execution

All errors are considered to be fatal.

Syntax

```
PmGetLastWa
  [Workarea ':=' ] < expression (VAR) of pm_wadescr > ';'
```

Related information

For information about	See
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 363*
The instruction `PmSetLastWa`	*PmSetLastWa - Set last used work area on page 297*
The instruction `PmGetPathHeight`	*PmGetPathHeight - Get a safe path height for an intermediate movement on page 315*

7.2.6. PmGetOperation - Get operation from a work area

Usage

`PmGetOperation` is used to get operation data from a work area.

Basic example

```
PERS wobjdata wInfeeder2 := [FALSE,TRUE,"",
     [[2180.65,1430.22,-720.753],
     [0.00104,-0.00130,0.00039,1.00000]], [[0,0,0],[1,0,0,0]]];
VAR pm_wadescr PickWa;
VAR pm_operationdata Op;

PmGetWaByWobj wInfeeder2, PickWa;

PmGetOperation PickWa, Op;
```

Get operation data for the work area using work object data `wInfeeder2`.

Arguments

```
PmGetOperation Wa Operation [\MaxTime] [\TimeFlag]
```

`Wa`

Data type: `pm_wadescr`

Contains a reference to a work area.

`Operation`

Data type: `pm_operationdata`

Operation data that is fetched from a work area.

`[\MaxTime]`

Maximum Time

Data type: `num`

The maximum period of waiting time permitted, expressed in seconds. If this time runs out before the condition is met, the error handler will be called, if there is one, with the error code PM_ERR_TIMEOUT. If there is no error handler, the execution will be stopped.

`[\TimeFlag]`

Timeout Flag

Data type: `num`

The output argument that contains the value TRUE if the maximum permitted waiting time runs out before the condition is met. If this argument is included in the instruction, it is not considered an error if the maximum time runs out. This argument is ignored if the `MaxTime` argument is not included in the instruction.

Continues on next page

Continued

Error handling

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

Error code	Description
PM_ERR_TIMEOUT	No `pm_operationdata` could be fetched within the time out time.

Syntax

```
PmGetOperation
  [ Wa ':=' ]  < expression (IN) of pm_wadescr > ','
  [ Operation ':=' ]  < expression (INOUT) of pm_operationdata >
  [ '\' MaxTime ':=' < expression (IN) of num >]
  [ '\' TimeFlag ':=' < variable (VAR) of bool >] ';'
```

Related information

For information about	See
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 363*.
The data type `pm_operationdata`	*pm_operationdata - PickMaster operation data on page 345*.

7.2.7. PmGetProjectInfo - Get information about a specific project

Usage

`PmGetProjectInfo` gets information about a project. The project must be transferred to the controller.

Basic examples

A basic example of the instruction `PmGetProjectInfo` is illustrated below.

See also *More examples on page 291*.

Example 1

```
PROC main()
  ! Get info from select project
  PmGetProjectInfo ProjectSelection,ProjInfo;
ENDPROC
```

Arguments

```
PmGetProjectInfo SelectionNumber | Name ProjectInfo
```

`SelectionNumber`

Data type: `num`

The number that maps a transferred project with its signal value. See *Project Manager on page 98*.

`Name`

Data type: `string`

The name of a transferred project.

`ProjectInfo`

Data type: `pm_projectinfo`

Variable that holds the information about the project.

Program execution

The program will fail with a recoverable error if the project cannot be found. All other errorsare considered to be fatal.

Continues on next page

Continued

More examples

Another example of how to use the instruction `PMGetProjectInfo` is illustrated below.

Example 1

```
PROC main()
  VAR pm_projectinfo ProjInfo;
  VAR num ProjectSelection;

  ! Wait for start project order from PLC
  WaitDI pmProject_diStart,1;
  ! Check which project to be started
  ProjectSelection:=pmProject_giSelection;
  ! Get info from select project
  PmGetProjectInfo ProjectSelection,ProjInfo;
  ! Start the selected project
  PmStartProj ProjInfo.Name;

  WHILE TRUE DO
     ! Execute the main routine in the selected project.
     %"PmMain:Main"%;
  ENDWHILE
  ERROR
     IF ERRNO=PM_ERR_PROJ_NOT_FOUND THEN
        ! There is no project mapped to the selection value
        TRYNEXT;
     ENDIF
ENDPROC
```

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
`PM_ERR_PROJ_NOT_FOUND`	No project was found with this selection number or name.

Syntax

```
PmGetProjectInfo
  [SelectionNumber ':=' ] < expression (IN) of num> ','
  | [Name ':=' ] < expression (IN) of string > ','
  [ProjectInfo ':=' ] < expression (VAR) of pm_ projectinfo > ';'
```

Related information

For information about	See
The data type `pm_projectinfo`	*pm_projectinfo - PickMaster project information on page 348*.

7.2.8. PmGetWaByWobj - Get a reference to a work area using a work object data

Usage

`PmGetWaByWobj` gets the reference for a specified work area.

The arguments to the instruction is the work object data, that is to be connected to the work area, and the `pm_wadescr`.

Basic example

```
PERS wobjdata wInfeeder1 :=
     [FALSE,TRUE,"",[[2180.65,-1430.22,-220.753],
     [0.00104,-0.00130,0.00039,1.00000]],[[0,0,0],[1,0,0,0]]];
VAR pm_wadescr PickWa;
PmGetWaByWobj wInfeeder1, PickWa;
```

Arguments

```
PmGetWaByWobj WObj Wa
```

`WObj`

Work Object

Data type: `wobjdata`

The work object data that should be searched for in all work areas.

`Wa`

Data type: `pm_wadescr`

Variable that is updated to refer to the work area that corresponds to the provided work object.

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
PM_ERR_WOBJ	No work area has reference to the work object data used.

Syntax

```
PmGetWaByWobj
  [ Wobj ':=' ] < persistent (PERS) of wobjdata > ','
  [ Wa ':=' ] < expression (INOUT) of pm_wadescr > ';'
```

Related information

For information about	See
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 363*
The data type `wobjdata`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Data types*

7.2.9. PmGetWaInfo - Get information about a specific work area

Usage

`PmGetWaInfo` gets information about a work area. The work area must be in the started project. This information can be used in an external user interface or as a way to get a work area descriptor from a selection number. .

Basic examples

Basic examples of the instruction `PmGetWaInfo` are illustrated below.

Example 1

```
VAR pm_wainfo WaInfo;
VAR num WaSelection:=1;
! Get info from selected Work Area
PmGetWaInfo WaSelection,WaInfo;
```

Arguments

```
PmGetWaInfo SelectionNumber | WorkArea FlowInfo
```

`SelectionNumber`

Data type: `num`

The number that maps a specific work area with its signal value.

`WorkArea`

Data type: `pm_wadescr`

A valid descriptor in a started project. The descriptor could be collected from PmGetFlow or PmGetWaByWobj.

`FlowInfo`

Data type: `pm_wainfo`

Variable that holds the information about the work area.

Program execution

The program will fail with a recoverable error if the work area cannot be found. All other errors are considered to be fatal.

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
`PM_ERR_WA_NOT_FOUND`	No work area was found with this selection number or name.
`PM_ERR_NO_RUNNING_PROJECT`	No running project.

Continues on next page

Continued

Syntax

```
PmGetWaInfo
  [SelectionNumber ':=' ] < expression (IN) of num > ','
  |[Workarea ':=' ] < expression (VAR) of pm_wadescr > ','
  [WaInfo ':=' ] < expression (VAR) of pm_wainfo > ';'
```

Related information

Here you list related information and where to find it.

For information about	See
The data type `pm_wainfo`	*pm_wainfo - PickMaster Work Area information on page 364*
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 363*
The instruction `PmGetFlow`	*PmGetFlow - Get flow to execute on page 283*
The instruction `PmGetWaByWobj`	*PmGetWaByWobj - Get a reference to a work area using a work object data on page 292*

7.2.10. PmSearchAdjust - Adjust number of remaining layers

Usage

`PmSearchAdjust` is used after a stack search to adjust the number of remaining layers in a pallet pattern. It also updates the search frame to improve the picking accuracy for a pallet pattern or format.

Basic examples

Basic examples of the instruction `PmSearchAdjust` are illustrated below.

Example 1

```
VAR pm_wadescr PickWa;
VAR num PalletPatternHeightZ:=1097;

PmSearchAdjust PickWa, PM_SEARCH_Z, PalletPatternHeightZ;
```

A pallet pattern available at the specified work area is updated in the z-direction of the work object. The detected height of the pallet pattern is 1097 mm.

Example 2

```
VAR pm_wadescr PickWa;
VAR num FormatHeightZ:=154;

PmSearchAdjust PickWa, PM_SEARCH_Z, FormatHeightZ;
```

A format available at the specified work area is updated in the z-direction of the work object. The detected height of the format is 154 mm.

Arguments

```
PmSearchAdjust Workarea SearchType SearchPos
```

`WorkArea`

Data type: `pm_wadescr`

Contains a reference to a work area.

`SearchType`

Data type: `pm_searchtype`

Represents an integer with a symbolic constant for different types of searches.

`SearchPos`

Search Position

Data type: `num`

The detected size in mm of the pallet pattern or format. The size is expressed relative the work object.

Continues on next page

Continued

Error handling

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

Error code	Description
PM_ERR_PALLET_REDUCED	A number of layers were removed since the detected stack height was lower (at least half the product height lower) than expected. The error is recovered through acknowledge of the search target and fetching next operation.
PM_ERR_PALLET_EMPTY	The detected height of the pallet pattern or format indicates missing parts. The error is recovered through acknowledge of the search target, trigger the Redo Search signal for the work area and fetching next operation.

Limitations

The instruction may only be used after an action containing a `SearchL` movement has been fetched with `PmGetTgtAction` and before the corresponding target has been acknowledged.

Predefined data

The search type, used in argument `SearchType` can be one of the following:

Constant	Value	Description
PM_SEARCH_X	0	Search was performed in the x direction of the work object.
PM_SEARCH_Y	1	Search was performed in the y direction of the work object.
PM_SEARCH_Z	2	Search was performed in the z direction of the work object.

Syntax

```
PmSearchAdjust
  [ WorkArea ':=' ] < expression (IN) of pm_wadescr > ','
  [ SearchType ':=' ] < expression (IN) of pm_searchtype > ','
  [ SearchPos ':=' ] < expression (IN) of num > ';'
```

Related information

Here you list related information and where to find it.

For information about	See
Stack search	*Stack search on page 175*.
Format frame versus work object in format operation set	*Format frame versus work object in format operation set on page 377*.
Pallet pattern versus work object in pallet pattern operation set	*Pallet pattern frame versus work object in pallet pattern operation set on page 379*.
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 363*.
The data type `pm_searchtype`	*pm_searchtype - PickMaster stack search type on page 354*.

7.2.11. PmSetLastWa - Set last used work area

Usage

`PmSetLastWa` sets the last used work area. Use the instruction PmGetLastWa to get the work area.

Basic examples

Basic examples of the instruction `instruction` are illustrated below.

Example 1

```
VAR pm_wadescr WorkArea;
! Set last used work area
PmSetLastWa WorkArea;
```

Arguments

```
PmSetLastWa Workarea
```

`Workarea`

Data type: `pm_wadescr`

A descriptor to the last used work area.

Program execution

All errors are considered to be fatal.

Syntax

```
PmSetLastWa
  [Workarea ':=' ] < expression (VAR) of pm_wadescr > ';'
```

Related information

Here you list related information and where to find it.

For information about	See
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 363*
The instruction `PmGetLastWa`	*PmGetLastWa - Get last used work area on page 287*
The instruction `PmGetPathHeight`	*PmGetPathHeight - Get a safe path height for an intermediate movement on page 315*

7.2.12. PmSetRecoverAction - Set flow recover action

Usage

`PmSetRecoverAction` sets flow recover action before starting a flow in error state. This instruction must be used before starting a flow with use of the IO interface if the flow is in error state. The instruction also returns information that can be used in an event log message. This message describes circumstances for restarting with selected recover action.

Basic examples

Basic examples of the instruction `PmSetRecoverAction` are illustrated below.

Example 1

```
VAR pm_flowinfo FlowInfo;
PmSetRecoverAction FlowInfo.Name,PM_RECOVER_REDO_LAST_PICK;
```

Arguments

```
PmSetRecoverAction Name \WorkArea RecoverAction \EventId
     \Argument1 \Argument2 \Argument3 \Argument4
```

`Name`

Data type: `string`

The name of the flow to set recover action on.

`WorkArea`

Data type: `pm_wadescr`

The work area to perform recover action on. Mandatory if using recover action restart layer or next pallet. Not used for recover action continue and redo last pick. It is in most cases the master work area that should be chosen.

`RecoverAction`

Data type: `num`

The recover action that is performed at next flow start.

`[\EventId]`

Data type: `num`

The event message number in process domain that creates a message for the chosen recover action.

`[\Argument1]`

Data type: `errstr`

The first argument to the event log message, one space if not used.

`[\Argument2]`

Data type: `errstr`

The sencond argument to the event log message, one space if not used.

`[\Argument3]`

Data type: `errstr`

The third argument to the event log message, one space if not used.

Continues on next page

Continued

[\Argument4]

Data type: errstr

The fourth argument to the event log message, one space if not used.

Program execution

The program will fail with a recoverable error with recover action:

- ?tPM_RECOVER_REDO_LAYER or PM_RECOVER_NEXT_PALLET without a valid work area.
- ?tA recover action not in range.
- ?tPM_RECOVER_REDO_LAST_PICK when nothing is picked.

All other errors are considered to be fatal.

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
PM_ERR_WORKAREA_EXPECTED	This recover action demands a work area.
PM_ERR_NOT_VALID_RECOVER_ACTION	The recover action is not one of the supported.
PM_ERR_REDO_LAST_PICK_REJECTED	The redo last pick recover action is rejected, no products picked.
PM_ERR_NO_RUNNING_PROJECT	No running project.

Predefined messages

There are predefined event messages in the process domain for describing what to do before the flow start with the chosen recover action.

```
<Message number="2393">
  Flow recover with redo last pick
  The Flow <Flow name> will redo last unfinished operation at next
       flow start.
  Verify that:
     The tool is empty
     Products from last operation are restored on <WorkArea name>
     The reason for the stop is solved.
</Message>
```

Continues on next page

```
<Message number="2394">
  Flow recover with continue pick-place
  The Flow <Flow name> will restart from where it was stopped at
        next flow start.   Verify that the fault causing the stop
        has been handled.
  Verify expected number of products:
     Tool: <Number of products>
     WorkArea name/Number of products/Layer number
     <WorkArea name/Number of products/Layer number>
     <WorkArea name/Number of products/Layer number>
     <WorkArea name/Number of products/Layer number>
</Message>
<Message number="2395">
  Flow recover with restart layer
  The Flow <Flow name> will restart from beginning of layer <layer
        number> on WorkArea <WorkArea name> at next flow start.
  Verify that:
     The reason for the stop is solved
     The tool is empty
     Following WorkAreas are empty:
        <WorkArea name>
        <WorkArea name>
</Message>
<Message number="2396">
  Flow recover with next pallet
  The Flow <Flow name> will restart from beginning of next pallet
        on WorkArea <WorkArea name> at next flow start.
  Verify that:
     The reason for the stop is solved.
     The tool is empty
     Following WorkAreas are empty:
        <WorkArea name>
        <WorkArea name>
        <WorkArea name>
</Message>
<Message number="2397">
  Flow recover with redo last pick
  The Flow <Flow name> will redo last unfinished operation at next
        flow start.
  Verify that:
     The tool is empty
     New products can be supplied on <WorkArea name>
     The reason for the stop is solved.
</Message>
```

Continues on next page

Continued

Predefined data

Constant	Value	Description
PM_RECOVER_CONTINUE_OPERATION	1	The pick-place operation will continue from where it was stopped.
PM_RECOVER_REDO_LAYER	2	The pick-place operation repeats last layer.
PM_RECOVER_NEXT_PALLET	3	The pick-place operation continue with next pallet.
PM_RECOVER_REDO_LAST_PICK	4	The pick-place operation repeats last operation.

Syntax

```
PmSetRecoverAction
  [ Name ':=' ] < expression (IN) of string > ','
  [ '\' WorkArea ':=' ] < expression (VAR) of pm_wadescr > ','
  [ RecoverAction ':=' ] < expression (IN) of num > ','
  ['\' EventId ':=' ] < expression (IN) of num > ','
  ['\' Argument1 ':=' ] < expression (VAR) of errstr > ','
  ['\' Argument2 ':=' ] < expression (VAR) of errstr > ','
  ['\' Argument3 ':=' ] < expression (VAR) of errstr > ','
  ['\' Argument4 ':=' ] < expression (VAR) of errstr > ';'
```

Related information

Here you list related information and where to find it.

For information about	See
The instruction `PmStartFlow`	*PmStartFlow - Starts a specific flow on page 302*

7.2.13. PmStartFlow - Starts a specific flow

Usage

`PmStartFlow` starts a flow. The flow must be in the started project.

Basic examples

A basic example of the instruction `PmStartFlow` is illustrated below.

Example 1

```
TRAP TrapStartFlow
  VAR pm_flowinfo FlowInfo;
  VAR num FlowSelection;

  ! Get info from selected flow
  PmGetFlowInfo FlowSelection,FlowInfo;
  ! Start the selected flow
  PmStartFlow FlowInfo.Name;
ENDTRAP
```

Arguments

```
PmStartFlow Name
```

`Name`

Data type: `string`

Variable that refers to a flow in a started project.

Program execution

The program will fail with a recoverable error if no project is running. All other errors are considered to be fatal, such as wrong flow name.

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
`PM_ERR_NO_RUNNING_PROJECT`	No running project.
PM_ERR_WRONG_FLOW_STATE	Starting a flow in error state without setting a recover action.

Predefined data

There are predefined constants for the flow status. The constants are used for setting values on flow status signals, configured in flow editor. See *Flow on page 184*.

Constant	Value	Description
PM_FLOW_STOPPED	0	The flow is stopped
PM_FLOW_RUNNING	1	The flow is running
PM_FLOW_STOPPING_AFTER_CYCLE	2	The flow is stopping after current cycle is finished

Continues on next page

Continued

Constant	Value	Description
PM_FLOW_STOPPING_AFTER_LAYER	3	The flow is stopping after current layer is finished
PM_FLOW_STOPPING_AFTER_PALLET	4	The flow is stopping after current pallet is finished
PM_FLOW_ERROR	5	The flow is in error state

Syntax

```
PmStartFlow
  [FlowSelector ':=' ] < expression (IN) of string > ';'
```

Related information

For information about	See
The instruction `PmGetFlowInfo`	*PmGetFlowInfo - Get information about a specific flow on page 285*.
The instruction `PmStopFlow`	*PmStopFlow - Stop a specific flow on page 305*.
The instruction `PmSetRecoverAction`	*PmSetRecoverAction - Set flow recover action on page 298*
The data type `pm_flowinfo`	*pm_flowinfo - PickMaster flow information on page 338*.

7.2.14. PmStartProj - Start a PickMaster project

Usage

`PmStartProj` starts a PickMaster project. When this instruction is executed, the project setup is read and all PickMaster internal parts of the project are initialized. The time of execution depends on the size of the project.

Basic example

```
PmStartProj "MyPMProj";
  IF PM_PROJECT_STATUS=PM_PROJECT_STARTING THEN
    WaitUntil PM_PROJECT_STATUS=PM_PROJECT_RUNNING;
  ENDIF
```

Arguments

```
PmStartProj Name [\Signal]
```

`Name`

Data type: `string`

The name of the project to start.

`[\Signal]`

Data type: `signalgo`

The signal that shows the status of the project.

Predefined data

There are predefined constants for the status of the project. Those constants are used for setting values on Signal (project status signal) and the installed persistent variable `PM_PROJECT_STATUS`. `PM_PROJECT_STATUS` can be accessed from RAPID program.

Constant	Value	Description
PM_PROJECT_STOPPED	0	The project is stopped.
PM_PROJECT_STOPPING	1	The project is about to stop.
PM_PROJECT_STARTING	2	The project is starting up.
PM_PROJECT_RUNNING	3	The project is running.
PM_PROJECT_ERROR	4	The project is in error state.

Syntax

```
PmStartProj
  [ Name ':=' ] < expression (IN) of string > ','
  [ '\' Signal ':=' ] < expression (VAR) of signalgo > ';'
```

Related information

For information about	See
The instruction `PmStopProj`.	*PmStopProj - Stop current project on page 307*.

7.2.15. PmStopFlow - Stop a specific flow

Usage

`PmStopFlow` stops a flow. The flow must be in the started project.

Basic examples

A basic example of the instruction `PmStopFlow` is illustrated below.

Example 1

```
TRAP TrapStopFlow
  VAR pm_flowinfo FlowInfo;
  VAR num FlowSelection;
  VAR num StopOption;

  ! Get info from selected flow
  PmGetFlowInfo FlowSelection,FlowInfo;
  ! Stop the selected flow
  PmStopFlow FlowInfo.Name, StopOption;
ENDTRAP
```

Arguments

```
PmStopFlow Name StopOption
```

`Name`

Data type: `string`

Variable that refers to a flow in a started project.

`StopOption`

Data type: `num`

Variable that specifies different stop behavior.

Program execution

The program will fail with a recoverable error if no project is running. All other errors are considered to be fatal, such as wrong flow name or wrong value on `StopOption`.

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
`PM_ERR_NO_RUNNING_PROJECT`	No running project.
`PM_ERR_INVALID_FLOW_STOP_OPTION`	Invalid stop option.

Continues on next page

Predefined data

Flow status constants

There following constants are predefined for the status of the flow. Use the constants to set values on flow status signal, configured in flow I/O settings editor. See *Flow on page 184*.

Constant	Value	Description
PM_FLOW_STOPPED	0	The flow is stopped
PM_FLOW_RUNNING	1	The flow is running
PM_FLOW_STOPPING_AFTER_CYCLE	2	The flow is stopping after current cycle is finished
PM_FLOW_STOPPING_AFTER_LAYER	3	The flow is stopping after current layer is finished
PM_FLOW_STOPPING_AFTER_PALLET	4	The flow is stopping after current pallet is finished
PM_FLOW_ERROR	5	The flow is in error state

StopOption constants

The following constants are predefined for `StopOption`.

Constant	Value	Description
PM_FLOW_FINISH_CYCLE	1	The flow will finish current cycle before stopping
PM_FLOW_FINISH_LAYER	2	The flow will finish current layer before stopping
PM_FLOW_FINISH_PALLET	3	The flow will finish current pallet before stopping

Syntax

```
PmStopFlow
  [Name ':=' ] < expression (IN) of string > ','
  [StopOption ':=' ] < expression (IN) of num > ';'
```

Related information

For information about	See
The instruction `PmGetFlowInfo`	*PmGetProjectInfo - Get information about a specific project on page 290*.
The instruction `PmStartFlow`	*PmStartFlow - Starts a specific flow on page 302*.
The data type `pm_flowinfo`	*pm_flowinfo - PickMaster flow information on page 338*.

7.2.16. PmStopProj - Stop current project

Usage

`PmStopProj` stops the active PickMaster project.

Basic example

```
PmStopProj;
```

Syntax

```
PmStopProj;
```

Related information

For information about	See
The instruction `PmStartProj`	*PmStartProj - Start a PickMaster project on page 304*.

7.2.17. PmWaitProjStart - Wait for any active project

Usage

`PmWaitProjStart` is used to wait until any project is running. The instruction can be used with a timeout; the instruction then waits the timeout time before giving the answer. The instruction is blocking until any project is started from any client.

Basic example

```
PmWaitProjStart \MaxTime := 5;
```

Check if project is started. If not, wait 5 seconds to see if the project is started during that time.

Arguments

```
PmWaitProjStart [\MaxTime] [\TimeFlag]
```

`[\MaxTime]`

Maximum Time

Data type: `num`

The maximum period of waiting time permitted, expressed in seconds. If this time runs out before the condition is met, the error handler will be called, if there is one, with the error code PM_ERR_TIMEOUT. If there is no error handler, the execution will be stopped.

`[\TimeFlag]`

Timeout Flag

Data type: `bool`

The output argument that contains the value TRUE if the maximum permitted waiting time runs out before the condition is met. If this argument is included in the instruction, it is not considered an error if the maximum time runs out. This argument is ignored if the `MaxTime` argument is not included in the instruction.

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
PM_ERR_TIMEOUT	The project was not started within the time out time

Syntax

```
PmWaitProjStart
  [ '\' MaxTime ':=' < expression (IN) of num >]
  [ '\' TimeFlag ':=' < variable (VAR) of bool >] ';'
```

Related information

For information about	See
The instruction `PmStartProj`	*PmStartProj - Start a PickMaster project on page 304*.

7.3 Functions

7.3.1. PmCalcIntermid - Calculate intermediate position

Usage

This function calculates an intermediate position between two targets. If no limitations are set, the calculated position is a part of all axis movements.

Basic example

```
InterMid:=PmCalcIntermid(p10, tool2, wobj2, p20, tool1, wobj1, 0.4
     \MaxAngle:=180 \MinAngle:= -180 \AngleLimAx6 \MaxY:=2200
     \MinY:=-2200 \MinZ:=670 \FromWa:= LastWorkArea
     \ToWa:=WorkArea);
MoveJ InterMid,v1000,z200,tool0\Wobj:=wobj0;
```

Return value

Data type: `robtarget`

The function will return a `robtarget` expressed in `wobj0` and `tool0`.

Arguments

```
PmCalcIntermid (StartRobTgt StartTool StartWobj EndRobTgt EndTool
     EndWobj InterMidPart [\MaxAngle] [\MinAngle] [\AngleLimAx1]
     | [\AngleLimAx4] | [\AngleLimAx6] [\MaxX] [\MinX] [\MaxY]
     [\MinY] [\MaxZ] [\MinZ] [\FromWa] [\ToWa] [\LimitRobBase] |
     [\LimitWorld])
```

`StartRobTgt`

Data type: `robtarget`

The robot target from where the robot starts the move.

`StartTool`

Data type: `tooldata`

The tool that is used at the start point.

`StartWobj`

Data type: `wobjdata`

The work object that is used at the start point.

`EndRobTgt`

Data type: `robtarget`

The robot target where the move shall end.

`EndTool`

Data type: `tooldata`

The tool that is used at the end point.

`EndWobj`

Data type: `wobjdata`

The work object that is used at the end point.

Continues on next page

Continued

`InterMidPart`

Data type: `num`

The part of all the axis moves that is used as an intermediate position. If the intermediate position is in the middle of the start and end positions, the value shall be set to 0.5. The value must be between 0 and 1.

`[\MaxAngle]`

Data type: `num`

Maximum allowed axis angle on selected axis.

`[\MinAngle]`

Data type: `num`

Minimum allowed axis angle on selected axis.

`[\AngleLimAx1]`

Data type: `num`

Limit angle on axis 1.

`[\AngleLimAx4]`

Data type: `num`

Limit angle on axis 4.

`[\AngleLimAx6]`

Data type: `num`

Limit angle on axis 6.

`[\MaxX]`

Data type: `num`

Maximum allowed X-value on intermediate position.

`[\MinX]`

Data type: `num`

Minimum allowed X-value on intermediate position.

`[\MaxY]`

Data type: `num`

Maximum allowed Y-value on intermediate position.

`[\MinY]`

Data type: `num`

Minimum allowed Y-value on intermediate position.

`[\MaxZ]`

Data type: `num`

Maximum allowed Z-value on intermediate position.

`[\MinZ]`

Data type: `num`

Minimum allowed Z-value on intermediate position.

Continues on next page

Continued

`[\LimitRobBase]`

Data type: `switch`

Limitations on X, Y and Z are defined in the base frame of the robot.

`[\LimitWorld]`

Data type: `switch`

Limitations on X, Y and Z are defined in the world frame, that is, wobj0. If this switch is not selected, the limitations will be made in the robot base frame as default.

`[\FromWa]`

Data type: `pm_wadescr`

Reference to the work area the robot was operating before the intermediate movement. The presence of the reference will not affect the result of the function. The reference is only used to improve error handling of the function.

`[\ToWa]`

Data type: `pm_wadescr`

Reference to the next work area the robot will operate. The presence of the reference will not affect the result of the function. The reference is only used to improve error handling of the function.

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
PM_ERR_CALCCONF	Failed to calculate arm configuration. A highly complex arm configuration can cause this error.
PM_ERR_AXLIM	Failed to calculate axis limit. The axis angle cannot be calculated because of the angle limitations.
PM_ERR_LIM_VALUE	Wrong limitation value. The coordinate is not possible to calculate.
PM_ERR_PART_VAL	The value of the `InterMidPart` is not valid.

Continues on next page

Syntax

```
PmCalcIntermid '('
  [ StartRobTgt ':=' ]  < expression (IN) of robtarget > ','
  [ StartTool ':=' ]  < expression (IN) of tooldata > ','
  [ StartWobj ':=' ]  < expression (IN) of wobjdata > ','
  [ EndRobTgt ':=' ]  < expression (IN) of robtarget > ','
  [ EndTool ':=' ]  < expression (IN) of tooldata > ','
  [ EndWobj ':=' ]  < expression (IN) of wobjdata > ','
  [ InterMidPart ':=' ]  < expression (IN) of num >
  [ '\' MaxAngle ':=' < expression (IN) of num >
  [ '\' MinAngle ':=' < expression (IN) of num >]
  [ [ '\'AngleLimAx1 ] | [ '\'AngleLimAx4 ] | [ '\'AngleLimAx6 ] ]
  [ '\' MaxX := < expression (IN) of num >]
  [ '\' MinX := < expression (IN) of num >]
  [ '\' MaxY := < expression (IN) of num >]
  [ '\' MinY := < expression (IN) of num >]
  [ '\' MaxZ := < expression (IN) of num >]
  [ '\' MinZ := < expression (IN) of num >]
  [ '\' FromWa := < expression (IN) of pm_wadescr >]
  [ '\' ToWa := < expression (IN) of pm_wadescr >]')'
```

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

7.3.2. PmGetEvent - Get events for an action

Usage

`PmGetEvent` is used to get an event for an action on a work area.

Basic examples

```
VAR pm_eventdata Event;

ArrSize := Dim(TriggArr,1);
WHILE PmGetEvent(Wa, Tgt.TargetHandle, Act.ActionHandle, Event)
     AND i <= ArrSize DO
  TEST Event.Type
     CASE PM_EVENT_PROC:
        TriggEquip TriggArr{i}, Event.Dist, Event.Time,
              \ProcID:=Event.ProcId, Event.Value;
     CASE PM_EVENT_DO:
        GetDataVal Event.SignalName,doSignal;
        TriggEquip TriggArr{i}, Event.Dist, Event.Time,
              \DOp:=doSignal, Event.Value;
     CASE PM_EVENT_GO:
        GetDataVal Event.SignalName,goSignal;
        TriggEquip TriggArr{i}, Event.Dist, Event.Time,
              \GOp:=goSignal, Event.Value;
  ENDTEST
  Incr i;
ENDWHILE
TEST Act.NumOfEvents
  CASE 0:
     MoveL Tgt.RobTgtPoint, Act.Speed, Act.Zone, curr_Tool
           \WObj:=curr_WObj;
  CASE 1:
     TriggL Tgt.RobTgtPoint, Act.Speed, TriggArr{1}, Act.Zone,
           curr_Tool \WObj:=curr_WObj;
ENDTEST
```

Return value

Data type: `bool`

The function will return TRUE as long as a new `pm_eventdata` can be delivered for the action handle.

Arguments

```
PmGetEvent (Wa TargetHandle ActionHandle Event)
```

`Wa`

Data type: `pm_wadescr`

Contains a reference to a work area.

Continues on next page

`TargetHandle`

Data type: `pm_targethandle`

Contains a reference to a target.

`ActionHandle`

Data type: `pm_actionhandle`

Contains a reference to an action.

`Event`

Data type: `pm_eventdata`

Event data that is fetched from a work area.

Syntax

```
PmGetEvent '('
  [ Wa ':=' ]  < expression (IN) of pm_wadescr > ','
  [ TargetHandle ':=' ]  < expression (IN) of pm_targethandle > ','
  [ ActionHandle ':=' ]  < expression (IN) of pm_actionhandle > ','
  [ Event ':=' ]  < expression (INOUT) of pm_eventdata >')'
```

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

Related information

For information about	See
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 363*.
The data type `pm_targethandle`	*pm_targethandle - PickMaster target handle on page 362*.
The data type `pm_actionhandle`	*pm_actionhandle - PickMaster action handle on page 332*.
The data type `pm_eventdata`	*pm_eventdata - PickMaster event data on page 334*.
The instruction `TriggEquip`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Instructions*.
The instruction `TriggL`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Instructions*.

7.3.3. PmGetPathHeight - Get a safe path height for an intermediate movement

Usage

`PmGetPathHeight` returns a safe lowest path height for an intermediate movement between two work areas. None, one or both of the work areas may be defined as the home position of the robot.

The function returns the maximum height in wobj0 (mm) of the two work areas and their intermediate work areas. The height of a work area is either the *active height* of the products or the *default height*, depending on the *height state* of the work area, see section *Height state (GO) on page 259*. The home position does not have a height itself. The Safety offset is always added to the height of each work area.

The order of the work areas and the home position is defined in the **Feeder Order** tab, see section *Feeder Order on page 146*in the line configuration. Intermediate work areas are defined as those having an order value between the order values of the two work areas. The default height is used for inactive work areas, for example empty or full work areas waiting for new targets to be generated. The default height can be adjusted in runtime, for example to indicate that a finished stack has been unloaded. See *PmSetDefaultHeight - Set the default height on page 318*.

Basic examples

Basic examples of the function `PmGetPathHeight` are illustrated below.

Example 1

```
VAR pm_wadescr waInFeeder;
VAR pm_wadescr waOutFeeder;
VAR num MinZ;
PmGetFlow waInFeeder,waOutFeeder;
! Calculate MinZ
MinZ:=PmGetPathHeight(waInFeeder,waOutFeeder);
```

Return value

Data type: `num`

The function returns the lowest safe path height expressed in wobj0.

Arguments

```
PmGetPathHeight (FromWa ToWa [\UseSafePosition]
     [\UseDefaultHeight])
```

`FromWa`

Data type: `pm_wadescr`

The work area from where the robot shall move. May also be selected as the home position, that is, a default installed work area connected to the predefined wobjdata `pm_home_Wobj`. See section *Procedure `MoveHomePos` on page 196*.

Continues on next page

ToWa

Data type: pm_wadescr

The work area the robot will move to. May also be selected as the home position, that is a default installed work area connected to the predefined wobjdata pm_home_Wobj. See section *Procedure MoveHomePos on page 196*.

[\UseSafePosition]

Data type: switch

Consider the heights of used safe positions for work areas having an active height. For more information on how to configure safe positions, see *Safe Targets on page 159*.

[\UseDefaultHeight]

Data type: switch

Consider the default heights for work areas having an active height. For more information on how to configure default height, see *Robot Path Height on page 161*.

Program execution

All errors are considered to be fatal.

More examples

Example 1

```
PROC MoveInterMid(VAR pm_wadescr WorkArea, VAR pm_targetdata Tgt,
     VAR pm_actiondata Act,num SafetyHeight,\num MaxAngle,\num
     MinAngle,\switch MoveToEndPoint)
  CONST num IntermidPart1:=0.5;
  VAR robtarget InterMid1;
  VAR num MinZ;
  PmGetLastWa LastWorkArea;
  ! Calculate MinZ
  MinZ:=PmGetPathHeight(LastWorkArea,WorkArea\UseSafePosition)
       +Tgt.TargetTool.tframe.trans.z+SafetyHeight;
  ! Use the frame from tool0 and the load from target tool
  TempTool:=Tgt.TargetTool;
  TempTool.tframe:=tool0.tframe;! Travel distance: 50%
  InterMid1:=PmCalcIntermid(LastRobTgt,LastTool,LastWobj,
       Act.RobTgt,Tgt.TargetTool,Tgt.TargetWobj,
       IntermidPart1\MaxAngle?MaxAngle\MinAngle?MinAngle
       \AngleLimAx6\MinZ:=MinZ);
  MoveJ\Conc,InterMid1,Act.Speed,z200,TempTool\Wobj:=wobj0;
ENDPROC
```

Syntax

```
PmGetPathHeight
   [FromWa ':=' ] < expression (VAR) of pm_wadescr > ','
   [ToWa ':=' ] < expression (VAR) of pm_wadescr > ','
   ['\' UseSafePosition ] ','
   ['\' UseDefaultHeight ] ';'
```

Continues on next page

Continued

Related information

For information about	See
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 363*
The instruction `PmSetLastWa`	*PmSetLastWa - Set last used work area on page 297*
The instruction `PmGetLastWa`	*PmGetLastWa - Get last used work area on page 287*

7.3.4. PmSetDefaultHeight - Set the default height

Usage

`PmSetDefaultHeight` updates the default height for a work area. It returns the new default height (mm).The height state GO signal of the work area is updated to reflect the change, see *Height state (GO) on page 259*. The signal update occurs immediately if the work area does not have an active height.

Setting a new default height is a possibility to save cycle time without decreasing the margins for collisions, especially if the project consists of many work areas and flows.

For an outfeeder the default height can be set to *Empty* after a finished stack has been unloaded. This allows the robot to make lower intermediate movements when passing over the outfeeder next times and thus saving cycle time.

For an infeeder the default height can be set to *Full* before a new stack is loaded. This will force the robot to make intermediate movements with enough height when passing over the work area.

The new default height is active until new targets have been generated (or after new default height is set).

Basic examples

Basic examples of the function `PmSetDefaultHeight` are illustrated below.

Example 1

```
PERS wobjdata wobjOutfeeder :=
[FALSE,TRUE,"",[[2180.65,-1430.22,-220.753],[0.00104,-
     0.00130,0.00039,1.00000]],[[0,0,0], [1,0,0,0]]];
VAR pm_wadescr OutWa;
VAR num NewDefHeight;

PmGetWaByWobj wobjOutfeeder, OutWa;

NewDefHeight:=PmSetDefaultHeight OutWa \Empty;
```

Example 2

```
PERS wobjdata wobjOutfeeder :=
[FALSE,TRUE,"",[[2180.65,-1430.22,-220.753],[0.00104,-
     0.00130,0.00039,1.00000]],[[0,0,0], [1,0,0,0]]];
VAR pm_wadescr OutWa;
VAR num NewDefHeight;

PmGetWaByWobj wobjOutfeeder, OutWa;

NewDefHeight:=PmSetDefaultHeight OutWa \Value:=100;
```

Return value

Data type: `num`

The function returns the new default height.

Continues on next page

Continued

Arguments

```
PmSetDefaultHeight (Workarea [\Standard] | [\Empty] | [\Full] |
     [\Latest] | [\Value])
```

`Workarea`

Data type: `pm_wadescr`

The work area.

`[\Standard]`

Data type: `switch`

Set the default height as configured in the **Robot Path Height** see *Robot Path Height on page 161*.

`[\Empty]`

Data type: `switch`

Set the default height to *Empty*.

`[\Full]`

Data type: `switch`

Set the default height to *Full*.

`[\Latest]`

Data type: `switch`

Set the default height to the height of the latest completed operation set.

`[\Value]`

Data type: `num`

Set the default height to a specified value (mm).

NOTE!

The *Safety offset* defined in the **Robot Path Height** will always be added to the specified height.

7.3.5. PmGetTarget - Get target

Usage

`PmGetTarget` is used to get a target for an operation on a work area. If the optional argument `OpHandle` is left out, the function will return the next target without regard to the operation it belongs to.

Basic example

```
PmGetOperation Wa, Op;
WHILE PmGetTarget(Wa, \OpHandle:=Op.OpHandle, Tgt) DO
  WHILE PmGetTgtAction(Wa, Tgt.TargetHandle, Act) DO
     ...
  ENDWHILE
ENDWHILE
```

Return value

Data type: `bool`

The function will return TRUE as long as a new `pm_targetdata` can be delivered.

Arguments

```
PmGetTarget (Wa [\OpHandle] Targets [\MaxTime] [\TimeFlag])
```

`Wa`

Data type: `pm_wadescr`

Contains a reference to a work area.

`[\OpHandle]`

Data type: `pm_ophandle`

Contains a reference for an operation on a work area.

`Target`

Data type: `pm_targetdata`

Target data that is fetched from a work area.

`[\MaxTime]`

Maximum Time

Data type: `num`

The maximum period of waiting time permitted, expressed in seconds. If this time runs out before the condition is met, the error handler will be called, if there is one, with the error code PM_ERR_TIMEOUT. If there is no error handler, the execution will be stopped.

`[\TimeFlag]`

Timeout Flag

Data type: `bool`

The output argument that contains the value TRUE if the maximum permitted waiting time runs out before the condition is met. If this argument is included in the instruction, it is not considered an error if the maximum time runs out. This argument is ignored if the `MaxTime` argument is not included in the instruction.

Continues on next page

Continued

Error handling

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
PM_ERR_TIMEOUT	No `pm_targetdata` could be fetched within the time out time.
PM_ERR_OPERATION_LOST	The `pm_operationdata` is not valid, probably because of a pulse on the robot execution signal.

Syntax

```
PmGetTarget '('
  [ Wa ':=' ] < expression (IN) of pm_wadescr >
  ['\' OpHandle ':=' < expression (IN) of pm_ophandle >] ','
  [ Target ':=' ]  < expression (INOUT) of pm_targetdata >
  [ '\' MaxTime ':=' < expression (IN) of num >]
  [ '\' TimeFlag ':=' < variable (VAR) of bool >] ')'
```

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

Related information

For information about	See
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 363*.
The data type `pm_ophandle`	*pm_ophandle - PickMaster operation handle on page 347*.
The data type `pm_targetdata`	*pm_targetdata - PickMaster target data on page 359*.

7.3.6. PmGetTgtAction - Get target action

Usage

`PmGetTgtAction` is used to get an action for a target on a work area.

Basic examples

```
PmGetOperation Wa, Op;
WHILE PmGetTarget(Wa \OpHandle:=Op.OpHandle, Tgt) DO
  WHILE PmGetTgtAction(Wa, Tgt.TargetHandle, Act) DO
     curr_WObj := Tgt.TargetWobj;
     curr_Tool := Tgt.TargetTool;
     MoveL Tgt.RobTgtPoint, Act.Speed, Act.Zone, curr_Tool
           \WObj:=curr_WObj;
  ENDWHILE
ENDWHILE
```

Return value

Data type: `bool`

The function will return TRUE as long as a new `pm_actiondata` can be delivered for the target handle.

Arguments

```
PmGetTgtAction ( Wa TargetHandle Action )
```

`Wa`

Data type: `pm_wadescr`

Contains a reference to a work area.

`TargetHandle`

Data type: `pm_targethandle`

Contains a reference to a target.

`Action`

Data type: `pm_actiondata`

Action data that is fetched from a work area.

Error handling

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

Error code	Description
PM_ERR_OPERATION_LOST	The `pm_operationdata` is not valid, probably because of a pulse on the robot execution signal.

Continues on next page

Continued

Syntax

```
PmGetTgtAction '('
  [ Wa ':=' ]  < expression (IN) of pm_wadescr > ','
  [ TargetHandle ':=' ]  < expression (IN) of pm_targethandle > ','
  [ Action ':=' ]  < expression (INOUT) of pm_actiondata > ')'
```

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

Related information

For information about	See
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 363*.
The data type `pm_targethandle`	*pm_targethandle - PickMaster target handle on page 362*.
The data type `pm_actiondata`	*pm_actiondata - PickMaster action data on page 329*.

7.3.7. PmGetWaHeight - Get the height of a work area

Usage

`PmGetWaHeight` gets the current stack height of a specified work area.

Basic examples

```
height := PmGetWaHeight (PickWa);
```

Return value

Data type: `num`

The function returns the current height of the specified work area in mm. The height is equivalent to the z-coordinate of the current top layer expressed in the work object. If the stack is empty, zero is returned.

Arguments

```
PmGetWaHeight (Wa [\UseSafePosition])
```

`Wa`

Data type: `pm_wadescr`

Contains a reference to a work area.

`[\UseSafePosition]`

Data type: `switch`

If `UseSafePosition` is set, the height is equivalent to the maximum z-coordinate of used safe positions and the current top layer expressed in the work object.

Syntax

```
PmGetWaHeight '('
  [ Wa ':=' ]  < expression (IN) of pm_wadescr >')'
```

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

Related information

For information about	See
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 363*.

7.3.8. PmGetWaName - Get the name of a work area

Usage

`PmGetWaName` gets the name of a specified work area.

Basic examples

A basic example of the function `PmGetWaName` is illustrated below.

Example 1

```
VAR string waname;
waname:=PmGetWaName(WorkArea);
```

Return value

Data type: `string`

The function will return the name of the work area.

Arguments

```
PmGetWaName (WorkArea)
```

`WorkArea`

Work Area

Data type: `pm_wadescr`

Contains a reference to a work area.

Syntax

```
PmGetWaName '('
  [ WorkArea ':=' ] < expression (VAR) of pm_wadescr >')'
```

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

7.4 Data types

7.4.1. pm_accdata - PickMaster acceleration/deceleration data

Usage

`pm_accdata` is used to describe and restrict accelerations and decelerations.

Description

`pm_accdata` is a part of `pm_actiondata` and is used as input arguments to the instructions `PathAccLim` and `AccSet`. It restricts the robots acceleration and deceleration.

Components

`acc`

acceleration

Data type: `num`

Acceleration and deceleration as a percentage of the normal values. 100% corresponds to maximum acceleration. Maximum value: 100%. Input value lower than 20% gives 20% of maximum acceleration.

Used as argument `Acc` in `AccSet`.

`ramp`

Data type: `num`

The rate at which acceleration and deceleration increases as a percentage of the normal values (see instruction `AccSet` for more information).

Used as argument `Ramp` in `AccSet`.

`acclim`

acceleration limit

Data type: `bool`

TRUE if there is to be a limitation of the acceleration, FALSE otherwise.

Used as argument `AccLim` in `PathAccLim`.

`accmax`

max acceleration

Data type: `num`

The absolute value of the acceleration limitation in m/s^2.

Only used when `acclim` is TRUE.

Used as argument `AccMax` in `PathAccLim`.

`decellim`

deceleration limit

Data type: `bool`

TRUE if there is to be a limitation of the deceleration, FALSE otherwise.

Used as argument `DecelLim` in `PathAccLim`.

Continues on next page

Continued

`decelmax`

max deceleration

Data type: `num`

The absolute value of the deceleration limitation in m/s^2.

Only used when `decellim` is TRUE.

Used as argument `DecelMax` in `PathAccLim`.

Examples

```
VAR pm_actiondata Act;
VAR num my_accmax;

WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) DO
  my_accmax := Act.accel.accmax;
  ...
ENDWHILE
```

Limitations

The `pm_accdata` members can only be set by the instruction `PmGetTgtAction`.

Structure

```
< dataobject of pm_accdata >
  < acc of num >
  < ramp of num >
  < acclim of bool >
  < accmax of num >
  < decellim of bool >
  < decelmax of num >
```

Related information

For information about	See
The data type `pm_actiondata`	*pm_actiondata - PickMaster action data on page 329*.
The function `PmGetTgtAction`	*PmGetTgtAction - Get target action on page 322*.
The instruction `PathAccLim`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Instructions*.
The instruction `AccSet`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Instructions*.

7.4.2. pm_acktype - PickMaster target acknowledge type

Usage

`pm_acktype` is used to represent an integer with a symbolic constant for different types of acknowledgements.

Description

A `pm_acktype` is used to decide which type of acknowledgement should be used.

Example

```
PmAckTarget WorkArea, WorkArea, PM_ACK;
```

Predefined data

Constant	Value	Comment
PM_ACK	301	The target is acknowledged as used
PM_NACK	302	The target is acknowledged as not used
PM_LOST	303	The target is acknowledged as lost

Characteristics

`pm_acktype` is an alias data type for `num` and consequently inherits its characteristics.

Related information

For information about	See
The instruction `PmAckTarget`	*PmAckTarget - Acknowledge a target on page 278*.

7.4.3. pm_actiondata - PickMaster action data

Usage

`pm_actiondata` specifies an action for a target.

Description

Properties for one target action.

Components

`RobTgt`

Data type: `robtarget`

Specifies the position of the robot and external axes.

`Type`

Data type: `pm_actiontype`

Specifies type of action.

`MoveType`

Data type: `pm_movetype`

Specifies type of movement.

`ArmConfMon`

Data type: `bool`

Specifies if the robot’s configuration is monitored during the movement.

`UseConc`

Use concurrent

Data type: `bool`

Specifies if concurrent program execution is used or not. See *Technical reference manual - RAPID Instructions, Functions and Data types*.

`SingAreaType`

Data type: `pm_singareatype`

Specifies type of interpolation mode.

`Accel`

Acceleration and deceleration data

Data type: `pm_accdata`

Restrict the robot’s acceleration and deceleration.

`Search`

Data type: `pm_searchdata`

Defines search type, search stop type and search signal to be used with a search movement.

Continues on next page

`Speed`

Data type: `speeddata`

Specifies the movement speed. See *Technical reference manual - RAPID Instructions, Functions and Data types*.

`Zone`

Data type: `zonedata`

Specifies the corner path after the movement. See *Technical reference manual - RAPID Instructions, Functions and Data types*.

`NumOfEvents`

number of events

Data type: `num`

Specifies number of events.

`ActionHandle`

Data type: `pm_actionhandle`

A reference to the action where this `pm_actiondata` was retrieved.

Examples

```
VAR pm_actiondata Act;

PmGetOperation WorkArea, Op;
WHILE PmGetTarget(WorkArea \OpHandle:=Op.OpHandle, Tgt) DO
  WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) DO
     ...
  ENDWHILE
ENDWHILE
```

Limitations

The action data members can only be set by the instruction `PmGetTgtAction`.

Structure

```
< dataobject of pm_actiondata >
  < RobTgt of targetdata >
  < Type of pm_actiontype >
  < MoveType of pm_movetype >
  < ArmConfMon of bool >
  < UseConc of bool >
  < SingAreaType of pm_singareatype >
  < Accel of pm_accdata >
  < Search of pm_searchdata >
  < Speed of speeddata >
  < Zone of zonedata >
  < NumOfEvents of num >
  < ActionHandle of pm_actionhandle >
```

Continues on next page

Continued

Related information

For information about	See
The data type `pm_movetype`	*pm_movetype - PickMaster movement type on page 342*.
The data type `pm_accdata`	*pm_accdata - PickMaster acceleration/deceleration data on page 326*.
The data type `pm_searchdata`	*pm_searchdata - PickMaster search data on page 352*
The function `PmGetTgtAction`	*PmGetTgtAction - Get target action on page 322*.
The data type `zonedata`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Data types*.
The data type `speeddata`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Data types*
The instruction `ConfL`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Instructions*.
Concurrent program execution	*Technical reference manual - RAPID overview*, section *Synchronisation with logical instructions*.

7.4.4. pm_actionhandle - PickMaster action handle

Usage

`pm_actionhandle` is used to store data about a target action.

Description

Data of the type `pm_actionhandle` contains a reference to an action.

Examples

```
PmGetOperation WorkArea, Op;
WHILE PmGetTarget(WorkArea \OpHandle:=Op.OpHandle, Tgt) DO
  WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) DO
     WHILE PmGetEvent(WorkArea, Tgt.TargetHandle,
           Act.ActionHandle, Event) DO
        ...
     ENDWHILE
  ENDWHILE
ENDWHILE
```

Limitations

Describe the limitations for the data type. `Names of data types` are written in script-text.

Characteristics

`pm_actionhandle` is a non-value data type.

Related information

For information about	See
The data type `pm_actiondata`	*pm_actiondata - PickMaster action data on page 329*.
The function `PmGetTarget`	*PmGetTarget - Get target on page 320*.
The function `PmGetTgtAction`	*PmGetTgtAction - Get target action on page 322*.
The function `PmGetEvent`	*PmGetEvent - Get events for an action on page 313*.

7.4.5. pm_actiontype - PickMaster action type

Usage

`pm_actiontype` is used to represent an integer with a symbolic constant for different types of actions.

Description

A `pm_actiontype` is used to decide which payload should be used for the next movement.

Example

```
TEST Tgt.Type
  CASE PM_APPROACH_POS:
     curr_Load := Tgt.AppProdsLoad;
  CASE PM_TARGET_POS:
     curr_Load := Tgt.AppProdsLoad;
     curr_StopPoint:=Tgt.StopPointData;
  CASE PM_DEPART_POS:
     curr_Load := Tgt.DepProdsLoad;
ENDTEST
GripLoad curr_Load;
```

Predefined data

Constant	Value	Comment
PM_APPROACH_POS	200	The action is a part of the approach movement
PM_TARGET_POS	201	The action moves to the target position
PM_DEPART_POS	202	The action is a part of the depart movement

Characteristics

`pm_actiontype` is an alias type for `num` and thus inherits its characteristics.

Related information

For information about	See
The data type `pm_actiondata`	*pm_actiondata - PickMaster action data on page 329*.
The instruction `GripLoad`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Instructions*.

7.4.6. pm_eventdata - PickMaster event data

Usage

`pm_eventdata` is used to specify an event.

Description

The components in `pm_eventdata` are used in the instruction `TriggEquip`. The different components decide what kind of trigg should be used for a specific action. For example, it describes when a Gripper should be closed or opened.

Components

`type`

event type

Data type: `pm_eventtype`

Type of event. Used to separate how to set up triggers with instruction `TriggEquip`. Types that are allowed are `PM_EVENT_PROC`, `PM_EVENT_DO`, and `PM_EVENT_GO`.

`time`

Data type: `num`

Input argument `EquipLag` in instruction `TriggEquip`.

`dist`

distance

Data type: `num`

Input argument `Distance` in instruction `TriggEquip`.

`procid`

process id

Data type: `num`

Input argument `ProcID` in instruction `TriggEquip`.

`signalname`

Data type: `string`

Input argument `DOp` or `GOp` in instruction `TriggEquip`.

`Atime`

Data type: `num`

Reserved for future use.

`value`

Data type: `num`

Input argument `SetValue` in instruction `TriggEquip`. Should not be used if a PickMaster project uses longer group signals than 23 bits.

Continues on next page

Continued

`Dvalue`

Data type: `dnum`

Input argument `SetDvalue` in instruction `TriggEquip`. Normally used instead of `value` since the numerical resolution is higher. Required for PickMaster projects using long group signals, that is longer than 23 bits and up to 32 bits.

Examples

```
VAR pm_eventdata Event;

ArrSize := Dim(TriggArr,1);
WHILE PmGetEvent(WorkArea, Tgt.TargetHandle, Act.ActionHandle,
     Event) AND i <= ?rrSize DO
  TEST Event.Type
    CASE PM_EVENT_PROC:
      TriggEquip TriggArr{i}, Event.Dist, Event.Time,
           \ProcID:=Event.ProcId, Event.Value;
    CASE PM_EVENT_DO:
      GetDataVal Event.SignalName,doSignal;
      TriggEquip TriggArr{i}, Event.Dist, Event.Time,
           \DOp:=doSignal, Event.Value;
    CASE PM_EVENT_GO:
      GetDataVal Event.SignalName,goSignal;
      TriggEquip TriggArr{i}, Event.Dist, Event.Time,
           \GOp:=goSignal, Event.Value;
  ENDTEST
  Incr i;
ENDWHILE
```

Limitations

The event data members can only be set by the instruction `PmGetEvent`.

Structure

```
< dataobject of pm_eventdata >
  < type of pm_eventtype >
  < time of num >
  < dist of num >
  < procid of num >
  < signalname of string >
  < value of num >
```

Related information

For information about	See
The data type `pm_eventtype`	*pm_eventtype - PickMaster event type on page 337*.
The function `PmGetEvent`	*PmGetEvent - Get events for an action on page 313*.

Continues on next page

Continued

For information about	See
The instruction `TriggEquip`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Instructions*.
The instruction `TriggL`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Instructions*.

7.4.7. pm_eventtype - PickMaster event type

Usage

`pm_eventtype` is used to represent an integer with a symbolic constant for different type of events.

Description

A `pm_eventtype` is used to decide which type of trigg event should be used.

Examples

```
TEST Event.Type
  CASE PM_EVENT_PROC:
     TriggEquip TriggArr{i}, Event.Dist, Event.Time,
          \ProcID:=Event.ProcId, Event.Value;
  CASE PM_EVENT_DO:
     GetDataVal Event.SignalName,doSignal;
     TriggEquip TriggArr{i}, Event.Dist, Event.Time,
          \DOp:=doSignal, Event.Value;
  CASE PM_EVENT_GO:
     GetDataVal Event.SignalName,goSignal;
     TriggEquip TriggArr{i}, Event.Dist, Event.Time,
          \GOp:=goSignal, Event.Value;
ENDTEST
```

Predefined data

Constant	Value	Comment
PM_EVENT_PROC	220	Process with identity ProcId should receive the event. (For internal use)
PM_EVENT_DO	221	Digital output signal is changed.
PM_EVENT_GO	222	Digital group output signal is changed.

Characteristics

`pm_eventtype` is an alias type for `num` and thus inherits its characteristics.

Related information

For information about	See
The data type `pm_eventdata`	*pm_eventdata - PickMaster event data on page 334*.
The instruction `TriggEquip`.	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Instructions*.

7.4.8. pm_flowinfo - PickMaster flow information

Usage

`pm_flowinfo` holds information about a flow.

Description

The flow information holds data and references to everything that is needed for starting a flow or viewing user information about a specific flow.

Components

`Name`

Data type: `string`

The name of the flow.

`SelectionNumber`

Data type: `num`

The number that connects a flow with its I/O signal value. See *Flow on page 184*.

`SignalName`

Data type: `string`

The name of the configured status signal.

`MasterWa`

Data type: `pm_wadescr`

A work area descriptor to the master work area in the flow.

`SlaveWorkAreas`

Data type: `pm_slavewainfo`

A collection of slave work areas.

`NumberOfSlaveWA`

Data type: `num`

Number of slave work areas in the flow.

Examples

Example 1

```
VAR pm_flowinfo FlowInfo;
PmGetFlowInfo FlowSelection,FlowInfo;
```

Limitations

The flow information members can only be set by the instruction `PmGetFlowInfo`.

Continues on next page

Continued

Structure

```
< dataobject of pm_flowinfo >
  < Name of string >
  < SelectionNumber of num >
  < MasterWa of pm_wadescr >
  < SlaveWorkAreas of pm_slavewainfo >
  < NumberOfSlaveWA of num >
```

Related information

For information about	See
The instruction `PmGetFlowInfo`	*PmGetFlowInfo - Get information about a specific flow on page 285*.
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 363*.
The data type `pm_slavewainfo`	*pm_slavewainfo - PickMaster slave work area information on page 356*.

7.4.9. pm_moduleinfo - PickMaster module information

Usage

`pm_moduleinfo` holds information about the RAPID modules that can be loaded into a task.

Description

The module information contains names of the modules that can be loaded into a RAPID task after the project is started. The modules are selected in Robot settings and transferred with the project. See *Robot Settings on page 149*.

Components

There are 10 components, named in a number series, `ModName1` to `ModName10`.

`ModName1`

Data type: `string`

The name of the file containing a RAPID module.

`...`

`ModName10`

Data type: `string`

The name of the file containing a RAPID module.

Examples

Example 1

```
VAR pm_projectinfo ProjInfo;
PmGetProjectInfo ProjectSelection,ProjInfo;
Load \Dynamic,"HOME:"\File:=ProjInfo.Robot1.ModuleNames.ModName1;
```

Limitations

The operation data members can only be set by the instruction `PmGetProjectInfo`. There can be maximum ten modules.

Structure

```
< dataobject of pm_moduleinfo >
  < ModName1 of string >
  < ModName2 of string >
  < ModName3 of string >
  < ModName4 of string >
  < ModName5 of string >
  < ModName6 of string >
  < ModName7 of string >
  < ModName8 of string >
  < ModName9 of string >
  < ModName10 of string >
```

Continues on next page

Continued

Related information

For information about	See
The instruction `PmGetProjectInfo`	*PmGetProjectInfo - Get information about a specific project on page 290*.
The data type `pm_robotinfo`	*pm_robotinfo - PickMaster robot information on page 350*.

7.4.10. pm_movetype - PickMaster movement type

Usage

`pm_movetype` is used to represent an integer with a symbolic constant for different type of movements.

Description

The `pm_movetype` is used to decide which type of movement instruction should be used.

Example

```
TEST Move
  CASE PM_MOVE_LIN:
    MoveL ToPoint, Speed, Zone, Tool\WObj:=WObj;
  CASE PM_MOVE_JOINT:
    MoveJ ToPoint, Speed, Zone, Tool\WObj:=WObj;
ENDTEST
```

Predefined data

Constant	Value	Comment
PM_MOVE_JOINT	240	The action is a joint movement
PM_MOVE_LIN	241	The action is a linear movement

Characteristics

`pm_movetype` is an alias data type for `num` and consequently inherits its characteristics.

Related information

For information about	See
The data type `pm_actiondata`	*pm_actiondata - PickMaster action data on page 329*.
The instruction `MoveJ`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Instructions*.
The instruction `MoveL`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Instructions*.

7.4.11. pm_offsetdata - Offset data

Usage

`pm_offsetdata` describes an offset position.

Description

The `pm_offsetdata` is used to displace a robot position in x, y, and z direction and rotation around the axis.

Components

`x`

Data type: `num`

The displacement in the x-direction, in the object coordinate system.

`y`

Data type: `num`

The displacement in the y-direction, in the object coordinate system.

`z`

Data type: `num`

The displacement in the z-direction, in the object coordinate system.

`rx`

Data type: `num`

The rotation in degrees around the x-axis of the tool coordinate system.

`ry`

Data type: `num`

The rotation in degrees around the y-axis of the tool coordinate system.

`rz`

Data type: `num`

The rotation in degrees around the z-axis of the tool coordinate system.

Examples

```
VAR pm_actiondata Act;
VAR num x_offset;

WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) DO
  x_offset := Act.offset.x
  ...
ENDWHILE
```

Limitations

The offset data members can only be set by the instruction `PmGetTgtAction`.

Continues on next page

Continued

Structure

```
< dataobject of pm_offsetdata >
  < x of num >
  < y of num >
  < z of num >
  < rx of num >
  < ry of num >
  < rz of num >
```

Related information

For information about	See
The data type `pm_actiondata`	*pm_actiondata - PickMaster action data on page 329*.
The function `PmGetTgtAction`	*PmGetTgtAction - Get target action on page 322*.
The function `Offs`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Functions*.

7.4.12. pm_operationdata - PickMaster operation data

Usage

`pm_operationdata` is used to specify an operation on a work area.

Description

The operation data holds data and references to everything that is going to be done when the robot enters a work area.

Components

`operationnumber`

Data type: `num`

The operation number is the ordinal number of the operation in the layer.

`layernumber`

Data type: `num`

The layer number is the ordinal number of the layer in the operation set.

`scenenumber`

Data type: `num`

The scene number is the ordinal number of the current operation set.

`formatid`

Data type: `num`

The format id value is defined in the **Pick Setting Configuration** window.

`numoftargets`

number of targets

Data type: `num`

The number of targets that are handled by the operation.

`ophandle`

operation handle

Data type: `pm_ophandle`

The operation handle is a reference to the operation.

Examples

```
VAR pm_operationdata Op;
PmGetOperation WorkArea, Op;
```

Limitations

The operation data members can only be set by the instruction `PmGetOperation`.

Continues on next page

Structure

```
< dataobject of pm_operationdata >
  < operationnumber of num >
  < layernumber of num >
  < scenenumber of num >
  < formatid of num >
  < numoftargets of num >
  < ophandle of pm_ophandle >
```

Related information

For information about	See
The instruction `PmGetOperation`	*PmGetOperation - Get operation from a work area on page 288*.
The format id value	The *Pick Setting on page 68* Configuration.

7.4.13. pm_ophandle - PickMaster operation handle

Usage

`pm_ophandle` is used to store data about an operation.

Description

Data of the type `pm_ophandle` contains a reference to an operation.

Examples

```
PmGetOperation WorkArea, Op;
WHILE PmGetTarget(WorkArea \OpHandle:=Op.OpHandle, Tgt) DO
  ...
ENDWHILE
```

Characteristics

`pm_ophandle` is a non-value data type.

Related information

For information about	See
The data type `pm_operationdata`	*pm_operationdata - PickMaster operation data on page 345*.
The instruction `PmGetOperation`	*PmGetOperation - Get operation from a work area on page 288*.

7.4.14. pm_projectinfo - PickMaster project information

Usage

`pm_projectinfo` holds information about a project.

Description

The project information holds data and references to everything that is needed for starting a project or viewing user information about a specific project.

Components

`ProjectDescription`

Data type: `string`

The description used in project properties.

`Name`

Data type: `string`

The name of the project.

`SelectionNumber`

Data type: `num`

The number that connects a project with its I/O signal value. See *Project Manager on page 98*.

`Robot1`

Data type: `pm_robotinfo`

The information related to the first robot RAPID task.

`Robot2`

Data type: `pm_robotinfo`

The information related to the second robot RAPID task, used in a MultiMove system.

`...`

`Robot6`

Data type: `pm_robotinfo`

The information related to the sixth robot RAPID task, used in a MultiMove system.

`NumberOfFlows`

Data type: `num`

The number of flows handled by the project.

Examples

```
VAR pm_projectinfo ProjInfo;
PmGetProjectInfo ProjectSelection,ProjInfo;
```

Continues on next page

Continued

Limitations

The operation data members can only be set by the instruction `PmGetProjectInfo`.

Structure

```
< dataobject of pm_projectinfo >
  < ProjectDescription of string >
  < Name of string >
  < SelectionNumber of num >
  < Robot1 of pm_robotinfo >
  < Robot2 of pm_robotinfo >
  < Robot3 of pm_robotinfo >
  < Robot4 of pm_robotinfo >
  < Robot5 of pm_robotinfo >
  < Robot6 of pm_robotinfo >
  < NumberOfFlows of num >
```

Related information

For information about	See
The instruction `PmGetProjectInfo`	*PmGetProjectInfo - Get information about a specific project on page 290*.
The data type `pm_robotinfo`	*pm_robotinfo - PickMaster robot information on page 350*.

7.4.15. pm_robotinfo - PickMaster robot information

Usage

`pm_robotinfo` holds information about a RAPID task connected to a robot.

Description

The robot information holds data and references to everything that is needed for setting up the environment for a RAPID task after starting the project.

Components

`Active`

Data type: `bool`

Defines if valid data for this robot exists.

`Name`

Data type: `string`

The name of the robot.

`TaskName`

Data type: `string`

The name of the RAPID task connected to this robot.

`LoadRapid`

Data type: `bool`

Defines if the RAPID modules should be loaded or not after the project starts.

`ModuleNames`

Data type: `pm_moduleinfo`

The RAPID modules that should be loaded after the project starts.

Examples

Example 1

```
VAR pm_projectinfo ProjInfo;
PmGetProjectInfo ProjectSelection,ProjInfo;
IF projInfo.Robot1.LoadRapid THEN
  LoadAllModules projInfo.Robot1.ModuleNames, projInfo.Name,
       projInfo.Robot1.TaskName;
ENDIF
```

Limitations

The robot information members can only be set by the instruction `PmGetProjectInfo`.

Continues on next page

Continued

Structure

```
< dataobject of pm_robotinfo >
  < Active of bool >
  < Name of string >
  < TaskName of string >
  < ModuleNames of pm_moduleinfo >
```

Related information

For information about	See
The instruction `PmGetProjectInfo`	*PmGetProjectInfo - Get information about a specific project on page 290*.
The data type `pm_projectinfo`	*pm_projectinfo - PickMaster project information on page 348*.
The data type `pm_moduleinfo`	*pm_moduleinfo - PickMaster module information on page 340*.

7.4.16. pm_searchdata - PickMaster search data

Usage

`pm_searchdata` is used to store data used for search movements.

Description

`pm_searchdata` is a part of `pm_actiondata` and is used as input argument to procedure `DoSearch`. It defines search specific data.

Components

`searchtype`

Data type: `pm_searchtype`

Type of search (x, y or z). Is used as argument in `PmSearchAdjust`.

`stoptype`

Data type: `pm_stoptype`

Search stop type. Defines the robot stop method when performing a search movement.

`signalname`

Data type: `string`

Search signal name. The name of the signal to supervise during the search movement.

`iotriggtype`

Data type: `pm_iotriggtype`

I/O trigger type. Defines how the search signal is triggered, e.g trigger on both flanks.

Example

```
VAR pm_actiondata Act;
VAR pm_stoptype my_stoptype;

WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) DO
  my_stoptype := Act.search.stoptype;
  ...
ENDWHILE
```

Limitations

The `pm_searchdata` members can only be set by the instruction `PmGetTgtAction`.

Structure

```
<dataobject of pm_searchdata>
  < searchtype of pm_searchtype >
  < stoptype of pm_stoptype >
  < signalname of string >
  < iotriggtype of pm_iotriggtype >
```

Continues on next page

Continued

Related information

For information about	See
The data type `pm_actiondata`	*pm_actiondata - PickMaster action data on page 329*.
The function `PmGetTgtAction`	*PmGetTgtAction - Get target action on page 322*.
The instruction `DoSearch`	*Procedure `PmDoSearch` on page 224*.

7.4.17. pm_searchtype - PickMaster stack search type

Usage

`pm_searchtype` is used to represent an integer with a symbolic constant for different types of stack search types.

Description

A `pm_searchtype` is used to specify which type of stack search type is used.

Example

```
PmSearchAdjust PickWA, PM_SEARCH_Z, 570;
```

Predefined data

Constant	Value	Comment
PM_SEARCH_X	0	Stack search in the x direction
PM_SEARCH_Y	1	Stack search in the y direction
PM_SEARCH_Z	2	Stack search in the z direction

Characteristics

`pm_searchtype` is an alias data type for `num` and consequently inherits its characteristics.

Related information

For information about	See
The instruction `PmSearchAdjust`	*PmSearchAdjust - Adjust number of remaining layers on page 295*.

7.4.18. pm_singareatype - PickMaster interpolation type around singular points

Usage

`pm_singareatype` is used to represent an integer with a symbolic constant for different types of interpolation around singular points.

Description

The `pm_singareatype` is used to decide how the robot is to move in the proximity of singular points.

Examples

```
IF Act.SingAreaType = PM_SING_AREA_OFF THEN
  SingArea\Off;
ELSE
  SingArea\Wrist;
ENDIF
```

Predefined data

Constant	Value	Comment
PM_SING_AREA_OFF	260	The tool orientation is not allowed to differ.
PM_SING_AREA_WRI	261	The tool orientation is allowed to differ somewhat to avoid wrist singularity.

Characteristics

`pm_singareatype` is an alias type for `num` and thus inherits its characteristics.

Related information

For information about	See
The data type `pm_actiondata`	*pm_actiondata - PickMaster action data on page 329*.
The instruction `SingArea`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Instructions*.

7.4.19. pm_slavewainfo - PickMaster slave work area information

Usage

`pm_slavewainfo` holds information about the slave work areas in a flow.

Description

The slave work area information is a collection of work areas that can be used in any PickMaster function or instruction that demands `pm_wadescr`. The maximum number of slaves is 19. Current number of slaves is specified with `NumberOfSlaveWA` in `pm_flowinfo`.

Components

There are 19 components, named in a number series, `SlaveWa1` to `SlaveWa19`.

`SlaveWa1`

Data type: `pm_wadescr`

The descriptor to a work area.

`...`

`SlaveWa19`

Data type: `pm_wadescr`

The descriptor to a work area.

Examples

Example 1

```
VAR pm_flowinfo FlowInfo;
VAR num height;
VAR num FlowSelection;
PmGetFlowInfo FlowSelection,FlowInfo;
height:=PmGetWaHeight(FlowInfo.SlaveWorkAreas.SlaveWa1);
```

Limitations

The slave work area information members can only be set by the instruction `PmGetFlowInfo`.

Continues on next page

Continued

Structure

```
< dataobject of pm_slavewainfo >
  < SlaveWa1 of pm_wadescr >
  < SlaveWa2 of pm_wadescr >
  < SlaveWa3 of pm_wadescr >
  < SlaveWa4 of pm_wadescr >
  < SlaveWa5 of pm_wadescr >
  < SlaveWa6 of pm_wadescr >
  < SlaveWa7 of pm_wadescr >
  < SlaveWa8 of pm_wadescr >
  < SlaveWa9 of pm_wadescr >
  < SlaveWa10 of pm_wadescr >
  < SlaveWa11 of pm_wadescr >
  < SlaveWa12 of pm_wadescr >
  < SlaveWa13 of pm_wadescr >
  < SlaveWa14 of pm_wadescr >
  < SlaveWa15 of pm_wadescr >
  < SlaveWa16 of pm_wadescr >
  < SlaveWa17 of pm_wadescr >
  < SlaveWa18 of pm_wadescr >
  < SlaveWa19 of pm_wadescr >
```

Related information

For information about	See
The instruction `PmGetProjectInfo`	*PmGetProjectInfo - Get information about a specific project on page 290*.
The data type `pm_flowinfo`	*pm_flowinfo - PickMaster flow information on page 338*.

7.4.20. pm_stoptype - PickMaster stop type

Usage

`pm_stoptype` is used to represent an integer with a symbolic constant for different types of stop methods.

Description

A `pm_stoptype` is used to decide which type of robot stop method that shall be used.

`pm_stoptype` is a part of `pm_searchdata`.

Example

```
VAR pm_searchdata SearchData;
VAR signaldi diSearchSignal;
VAR pm_stoptype StopType;
...
GetDataVal SearchData.SignalName,diSearchSignal;
StopType := SearchData.StopType;
TEST StopType
  CASE PM_STOP_NOT_USED:
     ...
  CASE PM_STOP:
     ...
  CASE PM_SSTOP:
     ...
  CASE PM_PSTOP:
     ...
```

Predefined data

Constant	Value	Comment
PM_STOP_NOT_USED	0	No stop is being used
PM_STOP	1	Robot stiff stop
PM_PSTOP	2	Robot path stop
PM_SSTOP	3	Robot soft stop

Characteristics

`pm_stoptype` is an alias data type for num and consequently inherits its characteristics.

Related information

For information about	See
How `pm_stoptype` is used in the procedure `DoSearch`	*Procedure `PmDoSearch` on page 224*.
The data type pm_searchdata	*pm_searchdata - PickMaster search data on page 352*.

7.4.21. pm_targetdata - PickMaster target data

Usage

`pm_targetdata` specifies a target for a work area.

Description

The target data holds data and a reference to a target.

Components

`TargetNumber`

Data type: `num`

Defines the ordinal number of the target.

`OperationNumber`

Data type: `num`

Defines the ordinal number of the operation.

`LayerNumber`

Data type: `num`

Defines the ordinal number of the layer.

`SceneNumber`

Data type: `num`

Defines the ordinal number of the scene.

`RobTgtPoint`

Data type: `robtarget`

Defines the position of the robot and external axes.

`TargetTool`

Data type: `tooldata`

Defines the tool used with the target.

`TargetWobj`

Data type: `wobjdata`

Defines the work object used with the target.

`StopPointData`

Data type: `stoppointdata`

Defines the `stoppointdata` to be used when moving to the position of the target.

`NumOfAppProds`

number of approach products

Data type: `num`

The number of products held by the robot tool when approaching the target position.

Continues on next page

Continued

`AppProdsLoad`

approach product load

Data type: `loaddata`

Defines the load attached to the robot tool when approaching the target position.

`NumOfDepProds`

number of depart products

Data type: `num`

The number of products held by the robot tool when departing from the target position.

`DepProdsLoad`

depart product load

Data type: `loaddata`

Defines the load attached to the robot when departing from the target position.

`NumOfActions`

number of actions

Data type: `num`

Specifies the number of target actions.

`TargetHandle`

Data type: `pm_targethandle`

A reference to the target from where this `pm_targetdata` was retrieved.

`ProductHeight`

Data type: `num`

The height of the product to place or pick. It is not necessary a product in the tool for this target.

Examples

```
PmGetOperation WorkArea, Op;
WHILE PmGetTarget(WorkArea \OpHandle:=Op.OpHandle, Tgt) DO
  ...
ENDWHILE
```

Limitations

The operation data members can only be set by the instruction `PmGetTarget`.

Continues on next page

Continued

Structure

```
< dataobject of pm_targetdata>
  < TargetNumber of num >
  < OperationNumber of num >
  < LayerNumber of num >
  < SceneNumber of num >
  < RobTgtPoint of robtarget >
  < TargetTool of tooldata >
  < TargetWobj of wobjdata >
  < StopPointData of stoppointdata >
  < NumOfAppProds of num >
  < AppProdsLoad of loaddata >
  < NumOfDepProds of num >
  < DepProdsLoad of loaddata >
  < NumOfActions of num >
  < TargetHandle of pm_targethandle >
  < ProductHeight of num >
```

Related information

For information about	See
The instruction `PmGetTarget`	*PmGetTarget - Get target on page 320*.
The data type `robtarget`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Data types*.
The data type `tooldata`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Data types*.
The data type `wobjdata`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Data types*.
The data type `stoppointdata`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Data types*.
The data type `loaddata`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Data types*.

7.4.22. pm_targethandle - PickMaster target handle

Usage

`pm_targethandle` is used to store data about a target.

Description

Data of the type `pm_targethandle` contains a reference to a target.

Example

```
PmGetOperation WorkArea, Op;
WHILE PmGetTarget(WorkArea \OpHandle:=Op.OpHandle, Tgt) DO
  WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) DO
     ...
  ENDWHILE
ENDWHILE
```

Characteristics

`pm_targethandle` is a non-value data type.

Related information

For information about	See
The data type `pm_targetdata`	*pm_targetdata - PickMaster target data on page 359*.
The function `PmGetTarget`	*PmGetTarget - Get target on page 320*.
The function `PmGetTgtAction`	*PmGetTgtAction - Get target action on page 322*.

7.4.23. pm_wadescr - PickMaster work area reference

Usage

`pm_wadescr` is used to store data about a work area.

Description

Data of the type `pm_wadescr` contains a reference to a work area.

Examples

```
PERS wobjdata wInfeeder1 :=
     [FALSE,TRUE,"",[[2180.65,-1430.22,-220.753],
     [0.00104,-0.00130,0.00039,1.00000]],[[0,0,0], [1,0,0,0]]];
VAR pm_wadescr PickWa;
PmGetWaByWobj wInfeeder1, PickWa;
```

Characteristics

`pm_wadescr` is a non-value data type.

Related information

For information about	See
The function `PmGetWaByWobj`	*PmGetWaByWobj - Get a reference to a work area using a work object data on page 292*.

7.4.24. pm_wainfo - PickMaster Work Area information

Usage

`pm_wainfo` holds information about a work area.

Description

The work area information holds data and references that can be used for viewing user information about the specific work area.

Components

`Name`

Data type: `string`

The name of the work area.

`SelectionNumber`

Data type: `num`

The number that connects a work area with its I/O signal value. See *I/O values on page 97*.

`Workarea`

Data type: `pm_wadescr`

The work area descriptor.

`TaskNo`

Data type: `num`

The RAPID task number this work area is connected to. Only valid after the first call to `PmGetTarget` for this work area.

`Order`

Data type: `num`

The configured order for this work area. See *Feeder Order on page 146*.

`DefaultHeight`

Data type: `num`

The configured default height. See *Robot Path Height on page 161*.

`DiTgtGenTrig`

Data type: `string`

The name of the configured target generation trigger signal.

`DoPosRegTrig`

Data type: `string`

The name of the configured position request trigger signal.

`GoProdSel`

Data type: `string`

The name of the configured target generation product selection signal.

Continues on next page

Continued

`GoFormSel`

Data type: `string`

The name of the configured target generation format selection signal.

`GiProdSel`

Data type: `string`

The name of the configured position request product selection signal.

`GiFormSel`

Data type: `string`

The name of the configured position request format selection signal.

`DiRobotExec`

Data type: `string`

The name of the configured robot execution signal.

`DiRedoSearch`

Data type: `string`

The name of the configured redo search signal.

`DoQueueEmpty`

Data type: `string`

The name of the configured queue empty signal.

`DoPosAvail`

Data type: `string`

The name of the configured position available signal.

`DoOpSetCompl`

Data type: `string`

The name of the configured operation set complete signal.

`DoExecState`

Data type: `string`

The name of the configured execution state signal.

`AccumulatedAckTarget`

Data type: `num`

The number of accumulated acknowledged targets since project start for this work area.

`AccumulatedNumOfProd`

Data type: `num`

The number of accumulated picked or placed products since project start for this work area.

Examples

Example 1

```
VAR pm_wainfo WaInfo;
VAR num WaSelection:=1;
PmGetWaInfo WaSelection,WaInfo;
```

Continues on next page

Limitations

The work area information members can only be set by the instruction `PmGetWaInfo`.

Structure

```
< dataobject of pm_wainfo >
  < Name of string >
  < SelectionNumber of num >
  < WorkArea of pm_wadescr >
  < TaskNo of num >
  < Order of num >
  < DefaultHeight of num >
  < DiTgtGenTrig of string >
  < DoPosReqTrig of string >
  < GoProdSel of string >
  < GoFormSel of string >
  < GiProdSel of string >
  < GiFormSel of string >
  < DiRobotExec of string >
  < DiRedoSearch of string >
  < DoQueueEmpty of string >
  < DoPosAvail of string >
  < DoOpSetCompl of string >
  < DoExecState of string >
  < AccumulatedAckTarget of num >
  < AccumulatedNumOfProd of num >
```

Related information

For information about	See
The instruction `PmGetWaInfo`	*PmGetWaInfo - Get information about a specific work area on page 293*
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 363*

8 Relationships between PickMaster frames

8.1. Introduction

Structure of this chapter

This chapter describes the different frames used by PickMaster and how they relate to each other. Each section describes the relationship between two (or several) frames.

8.2. Shape frame

Shape size

The size of the shape is specified as **X size**, **Y size** and **Z size** in the **Product/Pallet/Sheet Configuration** window.

Visualization of the shape frame

The shape frame is shown in the **Product/Pallet/Sheet Configuration** window, but without reference to any other frame.

In the windows **Pick Setting Configuration**, **Pallet Pattern Operation Set Configuration** and **Group Operation Set Configuration** the origin of the shape frame is marked as a blue corner on the shape.

8.3. Format frame versus shape frame

Format frame settings

The relationship between a format frame and shape frames are defined in the **Pick Setting Configuration** by the settings of **Orientation**, **Row**, and **Column**. See the *Pick Setting on page 68* configuration.

Orientation defines the orientation relationship between the format frame and the shape frames. Possible values are **Front**, **Left**, **Back**, and **Right**.

Row defines number of shapes in the y-direction of the format frame.

Column defines number of shapes in the x-direction of the format frame.

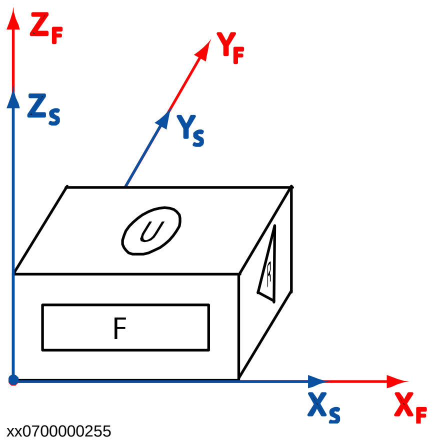
The relationship between the format frame and the shape frames will also depend on the X size, Y size, and Z size of the shape defined in the **Product/Pallet/Sheet Configuration** and the tuning of item size which is displayed in the same dialog.

Orientation explanation

Orientation defines which side of the item to place in the negative y-direction.

The following examples show some different orientations.

Orientation: Front



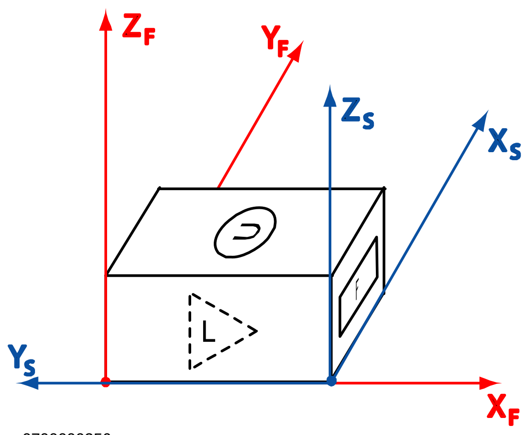
xx0700000255

X_F, Y_F, Z_F	Coordinates for the format frame
X_S, Y_S, Z_S	Coordinates for the shape frame
F	Front side of the shape
R	Right side of the shape
L	Left side of the shape
B	Back side of the shape
U	Upper side of the shape

Continues on next page

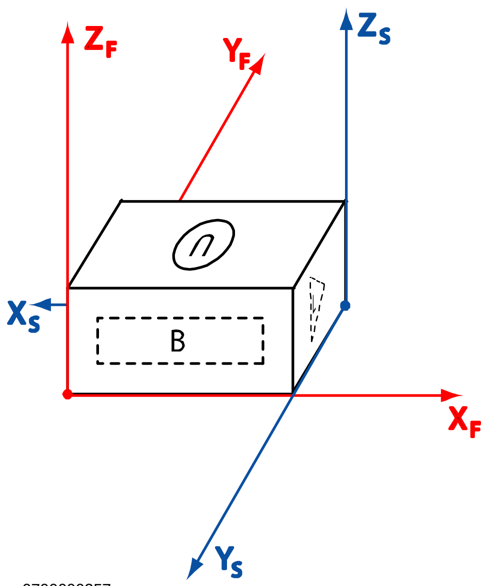
Continued

Orientation: Left



xx0700000256

Orientation: Back



xx0700000257

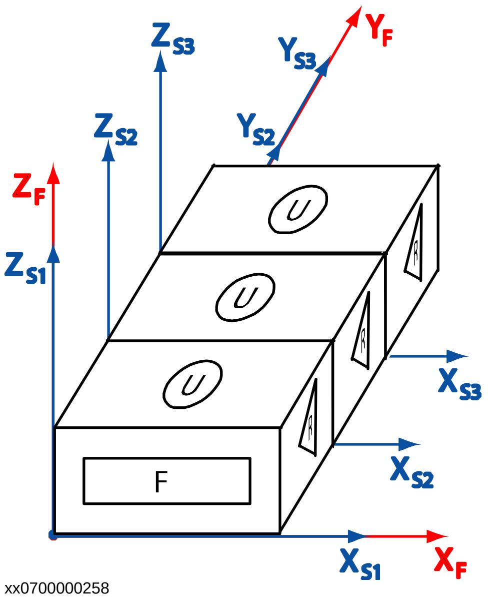
Continues on next page

Row explanation

Row defines number of shapes in the y-direction of the format frame.

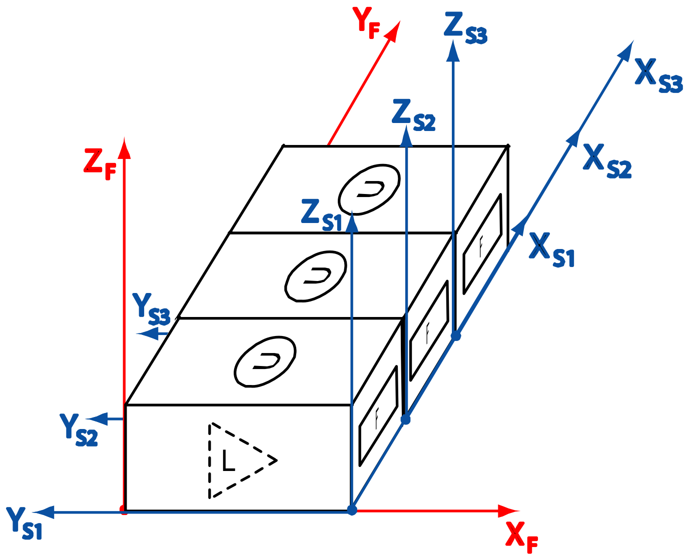
The following examples show the combination of row, column, and orientation.

Row: 3 ; Column: 1; Orientation: Front



xx0700000258

Row: 3 ; Column: 1; Orientation: Left



xx0700000259

Continues on next page

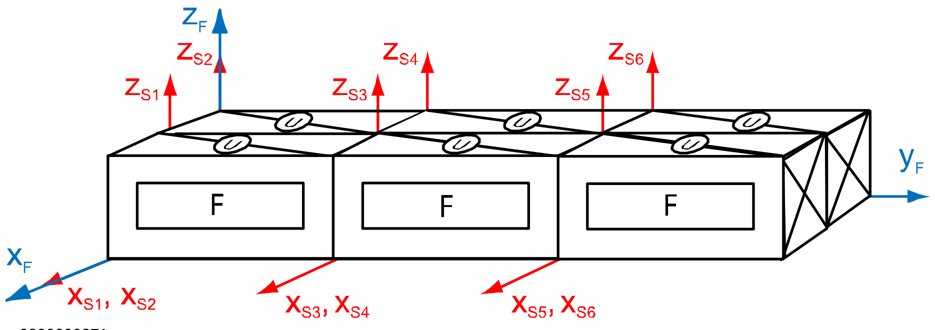
Continued

Column explanation

Column defines number of shapes in the x-direction of the format frame.

The following example shows the combination of row, column, and orientation.

Row: 3; Column: 2; Orientation: Front



xx0800000371

Visualization of the format frame

The format frame is shown in the **Pick Setting Configuration** window. For more information, see the *Pick Setting on page 68* configuration.

8.4. Pallet pattern frame versus shape frame

Use a palletizing area shape

The pallet pattern frame coincides with the shape frame of the palletizing area shape, which is selected in the **Layout Configuration** (only shapes with **Form** set to **Pallet** can be selected).

The palletizing area shape can be used as pallet in the pallet pattern but this is not mandatory.

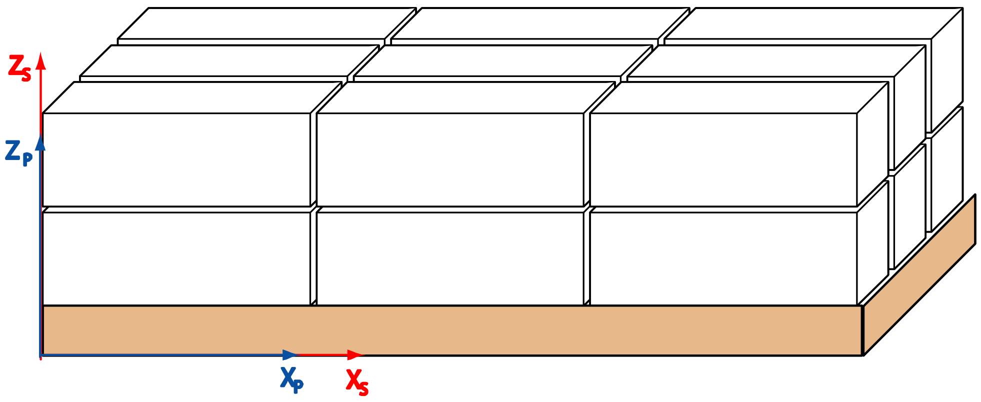
The palletizing area shape sets the x-y-size of the pallet pattern. Only box shapes and sheet shapes that are smaller than or have the same size as the palletizing area shape will be available for use within the pallet pattern.

Where to see the shape frame and pallet pattern

The location of the shape frame can be viewed in the **Product/Pallet/Sheet Configuration** window.

The location of the pallet pattern frame can be viewed in the **Layout Configuration** or the **Pallet Pattern Configuration** window.

Illustration

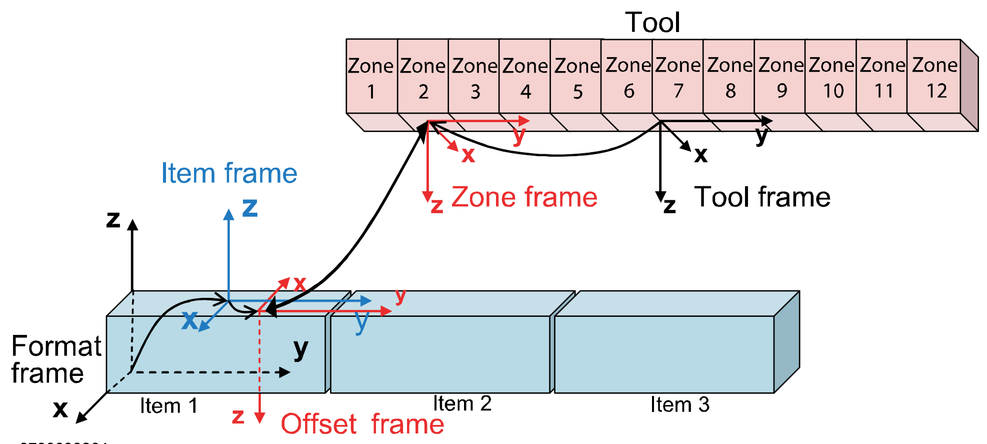


xx0700000263

X_S, Z_S	Coordinates for the shape frame of the palletizing area shape
X_P, Z_P	Coordinates for the pallet pattern frame

8.5. Format frame versus tool frame

Illustration



xx0700000264

Format frame

The format frame is defined in the **Item Group Configuration** window, under **Product/ Pallet/Sheet window, shown in the 3D view**.

Item frame

The offset frame is selected by setting **Aligned item** in the **Pick Setting Configuration** window, under **Tool location**. The offset frame is positioned on the top of the selected item and has the same orientation as the format frame.

Offset frame

The offset frame has an offset relative to the item frame. This offset is selected by setting **Offset** in the **Pick Setting Configuration** window, under **Tool location**. The offset frame z-direction is always opposite to the item frame. The orientation of the offset frame’s xy-plane relative to the item frame can be selected in steps of 90 degrees in the **Pick Setting Configuration** window, under **Tool location**.

Tool frame

The zone frame cannot be directly viewed in the **Pick Setting Configuration** window. Instead, the location of the tool frame is displayed. The tool frame position is different from the zone frame but the orientation is the same.

Zone frame

The zone frame is selected by setting aligned tool configuration and zone in the **Pick Setting Configuration** window, under **Tool location**. The zone frame has an offset relative to the tool frame. The location of the zone frames and the corresponding offsets are displayed in the **Tool Function(Vacuum)** window, under the **Zones** tab. The offset can be changed by selecting **Activator in Zones**, or by setting the activator x, y, z in the **Tool Function(Vacuum)** window, under **Activators** tab.

Continues on next page

Continued

Zone frame and offset frame

When a format is being picked by a robot, a zone frame coincides with an offset frame.

Where to see the tool frame

The location of the tool frame relative to a format can be viewed in the **Pick Setting Configuration** window. See the *Pick Setting on page 68* configuration.

8.6. Pallet pattern item versus tool frame

Tool location

The location of the tool relative to a specific item when picking/placing it, is automatically solved by PickMaster. The location of the tool is highly dependent on the selected **Formats to use** in the **Pallet Pattern Operation Set Configuration**. Every used format creates a number of possible positions of the tool relative to the item. The solution must also consider that every layer must be completely finished before starting with the next layer in the pallet pattern.

The autogenerated solution can be modified in detail in the **Operation Editor**, which is accessible from the **Pallet Pattern Operation Set Configuration**. For each item in every layer is it possible to modify:

?tThe pick/place order within the layer.

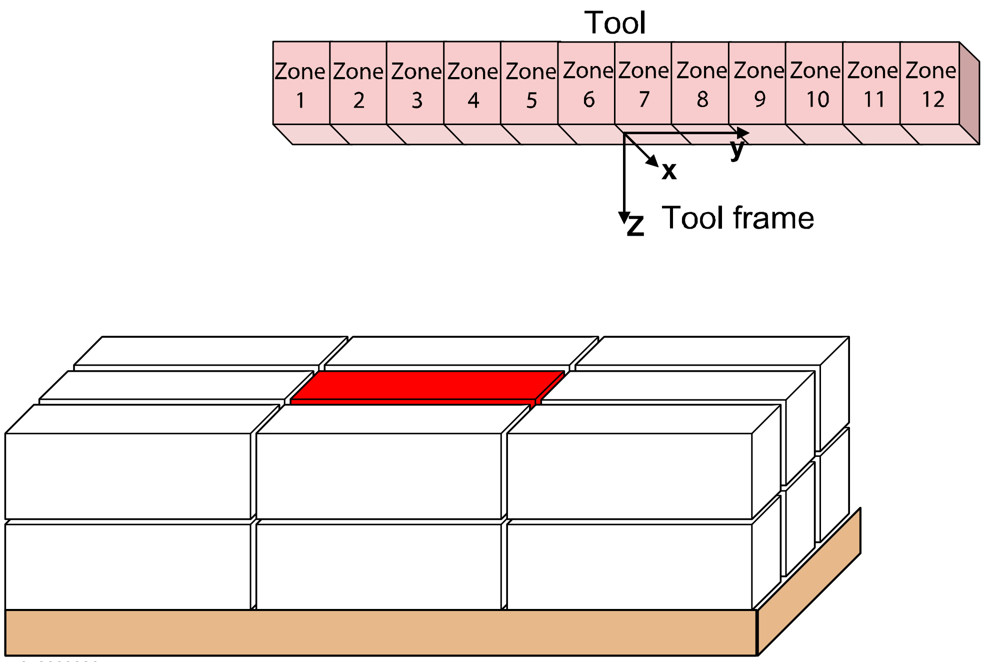
?tThe format to be used.

?tThe item in the selected format to be used.

?tThe orientation of the item may be flipped 180 degrees.

?tSingle access or (if possible) grouped access with adjacent items.

Illustration



xx0700000265

Where to see the tool frame

The location of the tool frame relative to an item in a pallet pattern can be viewed in the **Pallet Pattern Operation Set Configuration** window. See *Pattern/Stack Operation Set on page 167*.

Related information

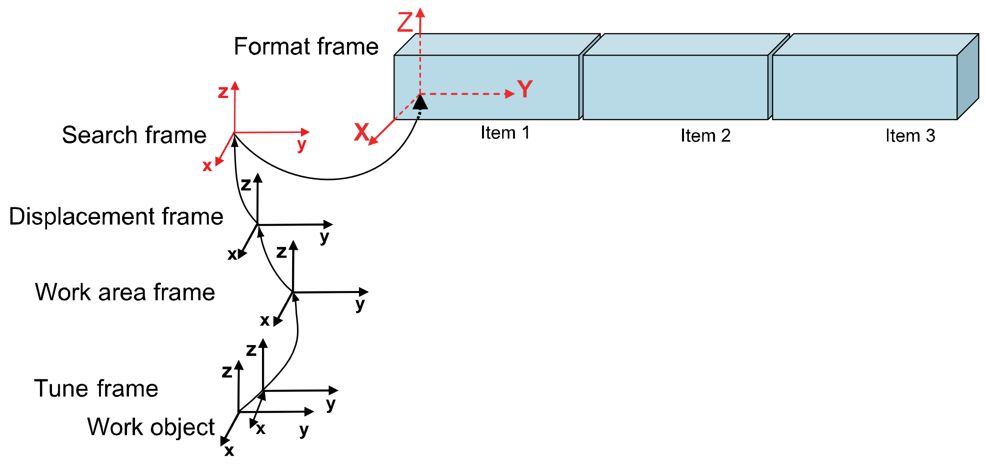
The *Operation Editor on page 179*.

8.7. Format frame versus work object in format operation set

Overview

The expected location of a format relative to the work object while being picked or placed depends on a chain of frames, starting with the work object and ending with the format frame.

Illustration



xx0700000266

Work object

The location of the work object is defined by the selection of **Work object** in the **Feeder Configuration** window.

The location of the work object is affected by all settings in the selected work object, including user frame and object frame settings. It is also affected by use of program displacement.

In PzPP, the position of the work object is aligned with the selected hotspot of the feeder model (Feeder hotpot configuration window).

Tune frame

The location of the tune frame relative the work object is defined by the current online tuning of the work area. By default the tune frame coincides with the work object. The tune frame location may be changed from the PickMaster RC online tuning by updating the following work area properties: **Disp offs x**, **Disp offs y**, **Disp offs z**, and **Disp angle z**. The tune frame is first displaced from the work object with the Disp offs vector. Then the tune frame is rotated around the displaced origo in the z-direction with the Disp angle z.

Work area frame

The location of the work area frame relative to the tune frame is defined by the Alignment and Rotation settings of the selected hotspot used by the feeder.

Displacement frame

The location of the displacement frame relative to the work area frame is defined by the settings of **Displacement frame**: **x**, **y**, **z** and **z angle** in the **Group Operation Set Configuration** window.

Continues on next page

Search frame

Search frame has an offset relative to the displacement frame. The offset is initially zero but is automatically updated after a stack search.

Format frame

At pick/place on a work area, the format frame coincides with the search frame.

Where to see the work object

The location of the work object relative to a general format can be viewed in the **Feeder Hotspot** window. However, the location is only valid if the following are zero: tune frame location, displacement frame offset, reorientation, and search frame.

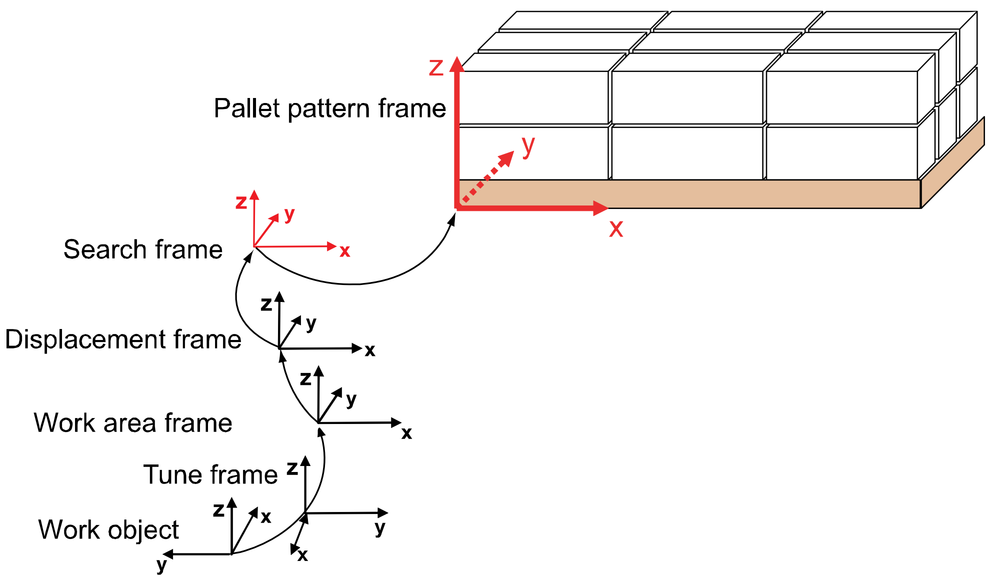
The location of the work object relative to a specific format can be viewed in the **Group Operation Set Configuration** window. However, the location is only valid if the tune frame location is zero (=default) and if the search frame offset is zero.

8.8. Pallet pattern frame versus work object in pallet pattern operation set

Overview

The expected location of a pallet pattern frame relative to the work object while being picked or placed depends on a chain of frames, starting with the work object and ending with the pallet pattern frame.

Illustration



xx0700000267

Work object

The location of the work object is defined by the selection of **Work object** in the **Feeder Configuration** window.

The location of the work object is affected by all settings in the selected work object, including user frame and object frame settings. It is also affected by use of program displacement.

In PzPP, the position of the work object is aligned with the selected hotspot of the feeder model (Feeder hotpot configuration window).

Tune frame

The location of the tune frame relative the work object is defined by the current online tuning of the work area. By default the tune frame coincides with the work object. The tune frame location may be changed from the PickMaster RC online tuning by updating the following work area properties: **Disp offs x**, **Disp offs y**, **Disp offs z**, and **Disp angle z**. The tune frame is first displaced from the work object with the Disp offs vector. Then the tune frame is rotated around the displaced origo in the z-direction with the Disp angle z.

Work area frame

The location of the work area frame relative to the tune frame is defined by the **Alignment and Orientation** settings of the selected **hotspot** used by the feeder.

Continues on next page

Continued

Displacement frame

The location of the displacement frame relative to the work area frame is defined by the settings of **Displacement frame**: **x**, **y**, **z** and **z angle** in the **Pallet Pattern Operation Set Configuration** window.

Search frame

Search frame has an offset relative to the displacement frame. The offset is initially zero but is automatically updated after a stack search.

Format frame

At pick/place on a work area, the pallet pattern frame coincides with the search frame.

Where to see the work object

The location of the work object relative to a general pallet pattern can be viewed in the **Feeder Hotspot** window. However, the location is only valid if the following are zero: the tune frame location, displacement frame offset, reorientation, and the search frame offset.

The location of the work object relative to a specific pallet pattern can be viewed in the **Pallet Pattern Operation Set Configuration** window. However, the location is only valid if the tune frame location is zero (=default) and if the search frame offset is zero.

8.9. Tune frame versus work area frame

Formats and pallet patterns

The relationship between tune frame and work area frame is valid for both formats and pallet patterns.

Alignment

Alignment defines which side of the work area frame the tune frame is found on, left or right.

The offset of the work area frame is zero with left alignment. With right alignment, the offset becomes equal to the x-size of the format expressed in the format frame.

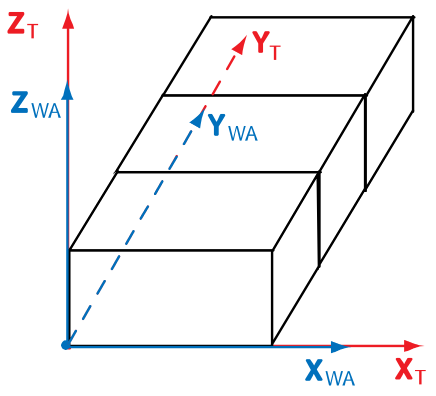
Orientation

Orientation defines the orientation of the tune frame in the z-direction relative to the work area frame: 0, 90, 180 or 270 degrees.

Examples

The displayed positions of the format in these examples assumes no displacement frame offset/reorientation and zero search offset.

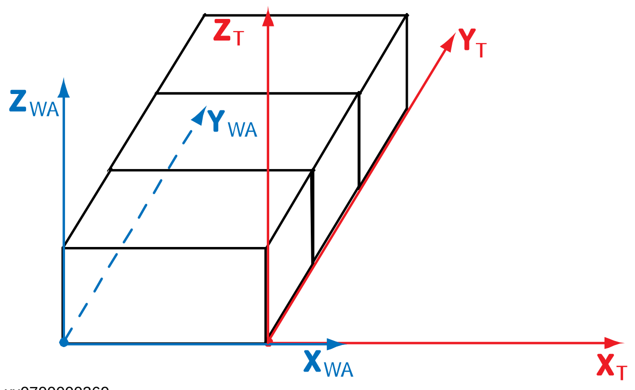
Alignment: Left; Orientation: 0



xx0700000268

X_T, Y_T, Z_T	Coordinates for the tune frame
X_{WA}, Y_{WA}, Z_{WA}	coordinates for the work area frame

Alignment: Right; Orientation: 0

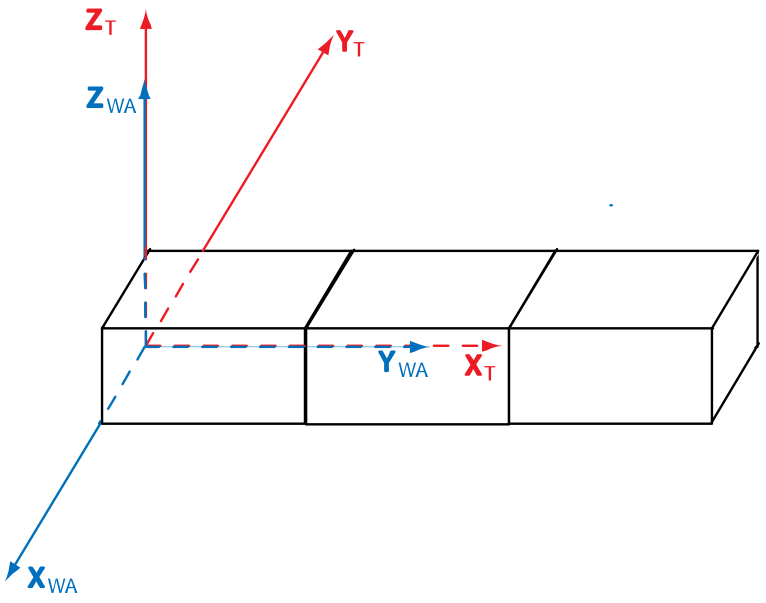


xx0700000269

Continues on next page

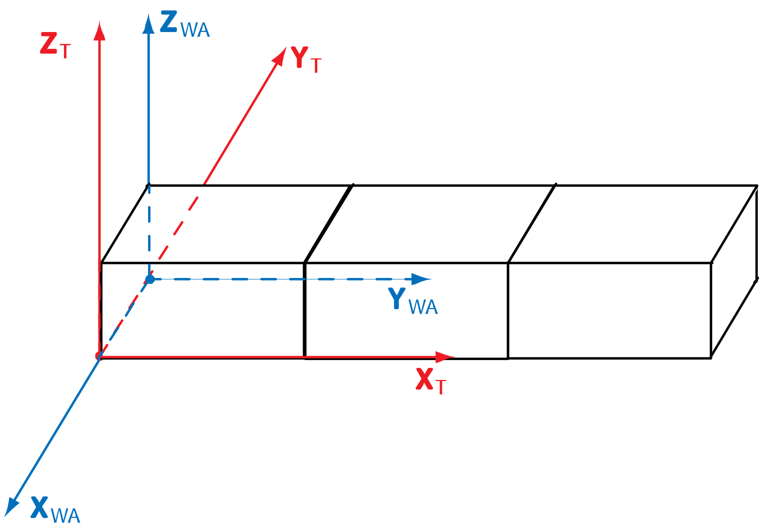
Continued

Alignment: Left; Orientation: 90



xx0700000270

Alignment: Right; Orientation: 90



xx0700000271

8.10. Tune frame versus work area frame versus displacement frame

Displacement frame settings

Displacement frame settings are made in the **Group Operation Set Configuration** window for formats and in the **Pattern Operation Set Configuration** window for pallet patterns.

x, **y**, and **z** defines an offset of the displacement frame relative to the work area frame but in the direction of the tune frame. **z angle** defines a z-rotation of the displacement frame relative the displaced origin.

Example

This example shows the coordinates for the tune frame, the work area frame and the displacement frame. It also shows how the items in the format are placed.

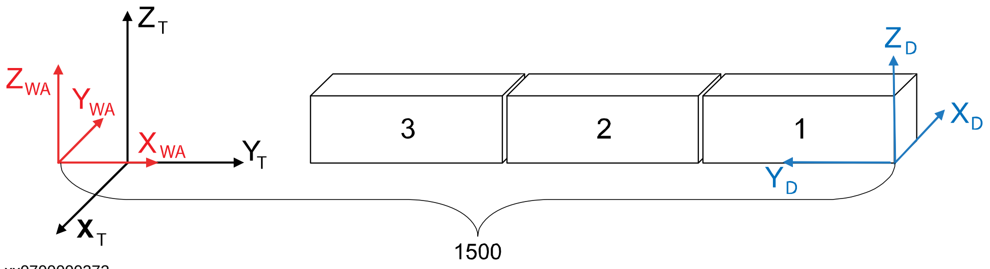
The following settings are made for the work area frame in the **Group Operation Set Configuration** window:

Setting	Value
Alignment	Right
Orientation	270

The following settings are made for the displacement frame in the **Pattern Operation Set Configuration** window:

Setting	Value
x	0
y	1500
z	0
z angle	90°

The displayed position of the format in this example assumes zero search frame offset.



xx0700000272

X_T, Y_T, Z_T	Coordinates for the tune frame
X_{WA}, Y_{WA}, Z_{WA}	Coordinates for the work area frame
X_D, Y_D, Z_D	Coordinates for the displacement frame