ABB Robotics

Application manual
PickMaster 5

Power and productivity
for a better world™

Trace back information:
Workspace R11-2 version a2
Checked in 2011-09-23
Skribenta version 759

Application manual

PickMaster 5

Version 5.14

Document ID: 3HAC025829-001

Revision: G

ABB AB
Robotics Products
SE-721 68 Västerås
Sweden

Table of contents

Overview of the manual

About this manual

This manual contains information and instructions for installing, configuring, and running PickMaster 5.

Usage

This manual should be used during installation and configuration of PickMaster 5. It describes PickMaster 5 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:

- System integrators
- ABB service engineers
- End customers

Prerequisites

The reader should:

- Be familiar with PickMaster.
- Have experience of installation and configuration work.
- Good skills in the IRC5 robot controller and RAPID programming.

Organization of chapters

The manual is organized in the following chapters:

Chapter	Contents
1Welcome to PickMaster	Introduction to PickMaster 5 and the *Line and Project* concept and to the palletizing process. A terminology list, including definitions of specific terms for this manual.
2Getting started	A step-by-step procedure describing the work flow, from installing the hardware and software to setting up the project for an application. Description of possible required modifications when using a robot with 6 axes.
3Hardware and software	How to install PickMaster on a PC and how to get and install a license key for PickMaster. Computer requirements are described.
4Navigate and handle PickMaster	Dialog boxes and other parts of PickMaster; their content and functions as well as how to access them.
5Configuration	Step-by-step procedures that describes the working procedure for a line and a project configuration.
6RAPID program	How RAPID programs are used in PickMaster. Program examples for palletizing applications.
7Runtime operation	How PickMaster palletizing projects are started, controlled, and supervised on a robot controller.
8RAPID reference information	Descriptions of RAPID instructions, functions and data types, that are specific for PickMaster.
9Relationships between PickMaster frames	Describes the different frames used by PickMaster and how they relate to each other.

Continues on next page

Continued

References

Document references

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

Other references

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

Revisions

Revision	Description
-	First edition
A	Added with the following new functionality: • Flow recovery • Tuning
B	Updated PickMaster version.
C	Updated with PickMaster 5.03 and 5.04. New functionality descriptions: • Reachability check. • PickMaster I/O interface, including default signals. • Redo last pick. • Quick start palletizing project. • Copy and paste project related objects, that is shape item, pallet pattern, position source, and format. • Multiple rows in tool configuration.

Continues on next page

Revision	Description
D	Updated with PickMaster 5.10. New functionality descriptions: • New licensing procedure. • Tuning of product height from FlexPendant interface. • Flow recovery from the I/O interface. • Moving between work areas in a safe way can be done using `PmGetPathHeight`. • New RAPID instructions: `PmGetLastWa`, `PmGetWaInfo`, `PmSetLastWa`, `PmSetRecoverAction` • New RAPID functions: `PmGetPathHeight` • New RAPID data types: `pm_wainfo` • New system module: *pmrcSys* Updates in RAPID reference descriptions.
E	Updated with PickMaster 5.12. The following updates are made: • Updated pictures and descriptions in sections *The Controller Properties window on page 66*, *The Work Area Configuration on page 73The Item Configuration on page 105* , *The Operation Set Configuration on page 128* and *The Position Source Configuration on page 124*. • Removed all information about event log files in section *The IRC5 Configuration on page 63*. • Signal descriptions updated in section *The Controller Properties window on page 66*. • Added valid values for the **Priority** setting in section *The General part on page 146*. • Modifications in section *Relationship between RAPID execution and PickMaster project on page 174*. • Terminology: `Queue` replaced by `WorkArea` in the RAPID program examples. • The *Public system module* section updated and divided into the sections *Public system module pmrcUser on page 196* and *Public system module pmrcSys on page 197*. • New sections added: *Public system module pmrcSys on page 197*, *PmSetDefaultHeight - Set the default height on page 302*, *Timing diagrams for PLC communication on page 256*. • Other updated sections *PmMain module on page 177*, *Basic I/O interface on page 240*, *Extended I/O interface on page 249* and *PmGetPathHeight - Get a safe path height for an intermediate movement on page 299*.

Continues on next page

Revision	Description
F	Updated with PickMaster 5.13 The following updates are made: - Updated *Computer requirements on page 38* - Updated figures and descriptions in section *The Work Area Configuration on page 73*, *The Item Configuration on page 105*, *The Operation Set Configuration on page 128*, *The Tune Limitations on page 158*,*The Flow Configuration on page 145Starting and stopping production on page 216*, *Flow recovery on page 223* and *Tuning on page 232*. - Removed description in **Setting up a flow** in section *Setting up the Project on page 167*. - Updated the program code in the Procedure `MoveHomePos` in section *PmUtility module on page 181*. - Updated content in sections *Basic I/O interface on page 240*. - Reference Information is moved to *Overview of the manual on page 9*
G	Updated with PickMaster 5.14.02 The following updates are made: - Removed the section **Activating the trial license** from the **Hardware and Software** chapter. - Updated the image in the section *Illustration, Tune Limitations on page 159*.

Product documentation, M2004

Categories for manipulator documentation

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

All documents listed can be ordered from ABB on a DVD. The documents listed are valid for M2004 manipulator systems.

Product manuals

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

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

Technical reference manuals

The technical reference manuals describe the manipulator software in general and contain relevant reference information.

- **RAPID Overview**: An overview of the RAPID programming language.
- **RAPID Instructions, Functions and Data types**: Description and syntax for all RAPID instructions, functions, and data types.
- **RAPID Kernel**: A formal description of the RAPID programming language.
- **System parameters**: Description of system parameters and configuration workflows.

Application manuals

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

An application manual generally contains information about:

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

Continues on next page

- How to use the application.
- Examples of how to use the application.

Operating manuals

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

The group of manuals includes (among others):

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

Safety

Safety of personnel

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

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

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

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

Safety regulations

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

This page is intentionally left blank

1 Welcome to PickMaster

1.1 Introduction

Structure of this chapter

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

- The *Line* and *Project* concept.
- A description of the palletizing process.
- A terminology list containing PickMaster and specific palletizing terms.

1.2 What is PickMaster?

Overview

PickMaster 5 is PC software for controlling ABB robots in palletizing applications via the IRC5 robot controller. It is designed to handle one or more cells in the production.

PickMaster is a modular product, which can be customized to your special needs:

- *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.
- *PickMaster PC*allows you to make the configuration for a palletizing application and process.

Prerequisites

PickMaster can be installed on a computer running Windows XP, Windows Vista, or Windows 7. 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.

PickMaster functionality

The main entries to the functionality of PickMaster are the Line and Project views.

PickMaster contains:

- The *PickMaster Library* for saving shapes, layouts and tools. See *The PickMaster Library on page 51*.
- The *IRC5 Configuration* for configuring the IRC5 robot controller. See *The IRC5 Configuration on page 63*.
- The *Work Area Configuration* for configuring the work area. See *The Work Area Configuration on page 73*.
- The *Tool Configuration* for configuring the robot tool. See *The **Tool Configuration** on page 81*.
- The *Event Settings* for defining signals that can report events from external devices. See *The Controller Properties window on page 66*.
- The *Shape Configuration* for configuring the product shape. See *The Shape Configuration on page 102*.
- The *Item Configuration* for configuring the item that will be picked and placed by the robot. See *The Item Configuration on page 105*.
- The *Format Configuration* for configuring the item and tool orientation, as well as the tool zones to activate when gripping the items. See *The Format Configuration on page 110*.
- The *Pallet Pattern Configuration* for configuring the stack of shapes organized in different layouts. See *The Pallet Pattern Configuration on page 115*.
- The *Layout Editor* for creating new or modifying existing layouts. See *The Layout Editor on page 120*.

Continues on next page

- The *Position Source Configuration* for configuring what to pick or place on which work area. See *The Position Source Configuration on page 124*.
- The *Operation Set Configuration* for configuring the format operation set and pattern operation set. See *The Operation Set Configuration on page 128*.
- The *Message Settings* for defining messages that can be reported from external devices. See *The Message Settings on page 149*.
- The *Robot Settings* for modifying safe positions for each work area. See *The Robot Settings on page 154*.
- The *Flow configuration* for defining accessible work areas in runtime. See *The Flow Configuration on page 145*.
- The *Tuning* for changing parameter values online from the FlexPendant. See *The Import Tune on page 156*.
- The *Project I/O value* editor for assigning I/O values to PickMaster projects. See *Setting up Project I/O values on page 54*.
- The *Flow I/O settings* editor for assigning I/O settings to flows. See *The Flow I/O settings editor on page 148*.

1.3 The Line and Project concept

Overview

PickMaster consists of two different parts that need to be set up separately, a line and a project. The line contains all the physical components, such as robots and work areas, whereas the project contains the items to be picked, pallet patterns, operations, and work flows.

Why a line and a project?

The purpose of the line and project concept is to keep all components that will not be changed very often in a separate line file. In general, a line will be created and configured once and for all. There may be several existing lines, but mostly only one is used. Many different projects can be created using the same line.

What is saved to a line?

Physical components and configurations saved to a line:

- Controllers.
- Robots.
- Tool configurations.
- Work areas.

What is saved to a project?

Items and configurations saved to a project:

- Items to pick and place.
- Position sources defining what, where and when to pick and place.
- RAPID programs for the robot controller.
- Robot settings.
- Project related messages.
- Pallet patterns.
- Flow configurations.
- Tune data.

Limitations

It is not possible to have more than one line open at the same time. This means that even though it is possible to have several projects open at the same time, they must all be configured to use the same line.

Related information

The Line view on page 57.

Setting up the line on page 164.

The Project view on page 95.

Setting up the project on page 167.

1.4 The palletizing process

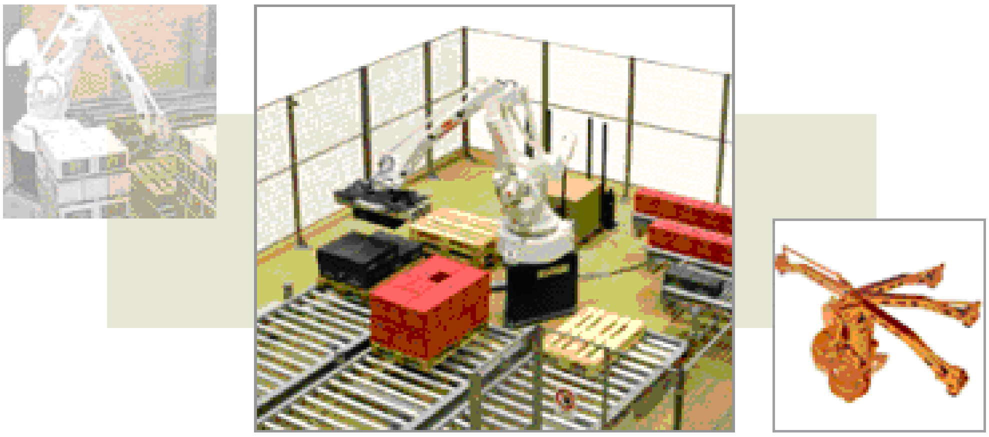
Overview

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.



xx0700000262

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

- Picking and placing one or more products.
- Picking and placing slip sheets from a slip sheet stack station to pallet stations. This task is optional.
- Picking 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 131*.

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.

Continues on next page

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.

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 format 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 format 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

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

Terminology on page 24.

1.5 Terminology

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 21*.

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 174*.
Activator	An I/O controlled part of a robot tool, normally a vacuum cup. For more information, see *Activators tab on page 87*.
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 174*.
Facing	Possibility to select one or more sides of an *item* that should be facing outwards in a *pallet pattern*.
Flow	Logical directions of *items* being moved between pick and place stations when performing a palletizing/depalletizing job. A flow: • Contains one master station on which palletizing/depalletizing jobs can be started. The master station dictates the operation sequence for itself and its slave stations. • Contains one or more slave stations. The slave stations execute operations requested by the master station. For more information, see *The Flow Configuration on page 145*.
Format	Defines one *item* or a group of *items* that can be picked/placed by a robot tool in one *operation*. For more information, see *The Format Configuration on page 110*.
Format operation set	One *operation* that describes how to pick/place a *format* on a specific *work area*. For more information, see *The Operation Set Configuration on page 128*.
Item	The generic term for a specific object to be picked or placed in a PickMaster application. An item can be a *product*, *pallet*, or *slip sheet*. An item is based on the geometric definition of a *shape*. For more information, see *The Item Configuration on page 105*.
Layout	Defines the arranged two dimensional pattern of the *shapes* in a layer.
Line	Description of static objects in a PickMaster installation, for example robots, *work areas*, robot tool. For more information, see *The Line view on page 57*.

Continues on next page

Continued

Term	Definition
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 174*.
Pallet	The actual wooden or plastic structure that the products are placed on.
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 *work area*.
Position source	Holds all *format operation sets* and all *pallet pattern operation sets* defined for a specific work area. Contains general robot movement data for the work area, for example safe positions.
Project	Description of a PickMaster palletizing process applied on a specific *line*. Several projects can be defined for each *line*. For more information, see *The Project view on page 95*.
Shape	Geometric description of an *item*.
Slip sheet	A thin sheet that is placed on a pallet before palletizing of products and between two layers to increase stability in the *stack*.
Stack	An arranged pile of items consisting of a number of layers.
Target	A target describes a robot position used when performing an *operation* on a *work area*. A target has a list of *products* carried by the robot tool and a list of *actions* that shall be executed.
Work area	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 *formats* and *pallet patterns* on the *work area*. Palletizing jobs can only be started on master work areas, see *Flow*. The feeding on a slave *work area* is dictated by one or more master *work areas*. For more information, see *The Work Area Configuration on page 73*.
Zone	A robot tool is divided into number of zones. Each zone has one or more *activators*.

This page is intentionally left blank

2 Getting started

2.1 Introduction

Structure of this chapter

This chapter describes how to get started with the PickMaster and includes:

- Hardware and software overview
- Step-by-step working procedure, which includes steps from how to get started with the installation of the PickMaster software till how to run a PickMaster project. The step-by-step working procedure summarizes all steps in a recommended order. For details, you are referred to other step-by-step procedures describing each step.
- A description of the quick start project included in the installation.

2.2 Hardware and software overview

Overview

This section gives an overview of the necessary hardware and software to run PickMaster.

Computer that runs PickMaster

To be able to run PickMaster you need a computer with some specific requirements, see *Computer requirements on page 38*.

For working with PickMaster it is necessary to connect the computer to the robot controller. There are also network settings for the computer that must be connected to the robot controller. See *Network settings on page 39*.

PickMaster software

The PickMaster computer software is delivered online, with all the files needed to install PickMaster and some additional software that may be useful. See *Additional software on page 40*.

Robot controller software, RobotWare

PickMaster 5 only supports the IRC5 robot controller. The RobotWare software is preinstalled on the robot controller and is also supplied on a DVD together with the robot controller. See *Robot controller software on page 44*.

Related information

Computer requirements on page 38.

Network settings on page 39.

Installing PickMaster on page 40.

Robot controller software on page 44.

2.3 Working procedure

Step-by-step overview

This is an overview of how to work with PickMaster.

	Action	See
1	Install the PickMaster software.	*Installing PickMaster on page 40*.
2	Request and install the license key.	*Activating a license key automatically over the Internet on page 42*, or *Activating a PickMaster license manually on page 43*.
3	Prepare your controller for PickMaster.	*Preparing your controller for PickMaster on page 163*.
4	Set up the *Line*.	*Working procedure for setup on page 162*.
5	Set up the *Project*.	*Working procedure for setup on page 162*.
6	Run the project.	*Starting production on page 172*.

Tip

To get started quickly you can use the quick start project included in the PickMaster installation.For more information, see *Quick starting a new PickMaster palletizing project on page 36*.

Note

To run PickMaster 5 applications on a robot with 6 axes might require some manual modifications depending on the robot configuration that is used. For more information, see *Using a robot with 6 axes on page 30*.

Related information

Getting and installing a license key for PickMaster on page 41.

Setting up the line on page 164.

Setting up the project on page 167.

Using a robot with 6 axes on page 30.

2.4 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:

- Axis 6 is pointing down. See *Axis 6 points down on page 31*.
- The wrist is tilted. See *Wrist is tilted on page 32*.
- The robot is bending backwards. See *Robot is bending backwards on page 33*.

For using a parallel rod robot with 6 axes together with PickMaster 5, see *Parallel rod robot on page 34*.

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 *The Format Operation Set Configuration on page 129* and *The Pallet Pattern Operation Set Configuration on page 130*.

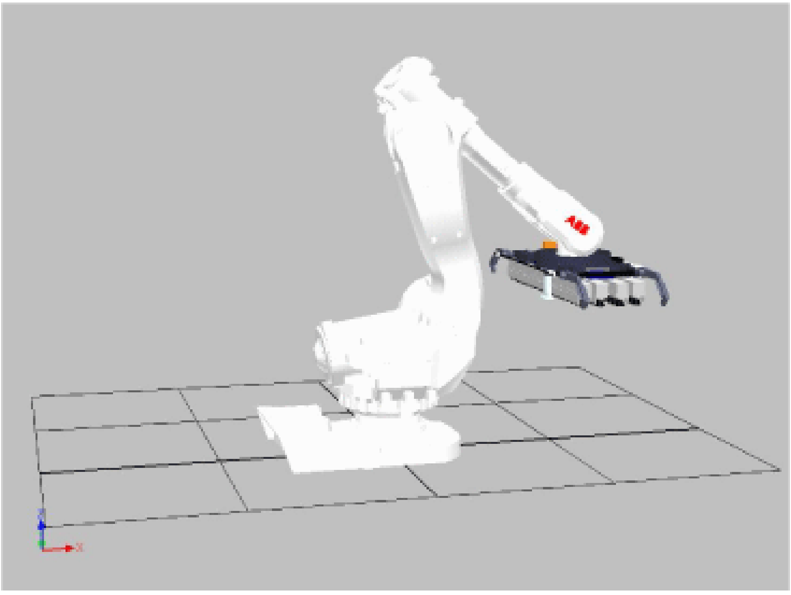
Continues on next page

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

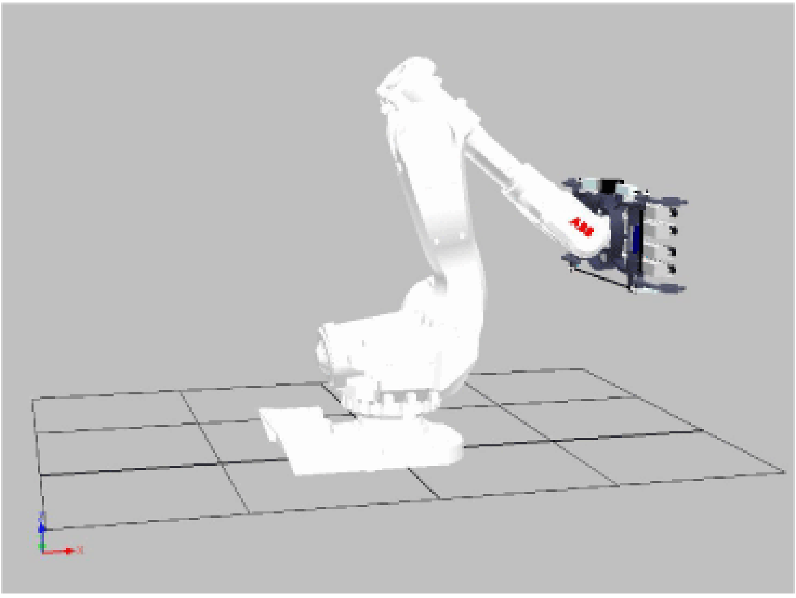
Configuration	Description
Using **SingArea \Wrist**	Long linear movements causes varying tool orientation. Tool orientation deviates between targets when using **SingArea /Wrist** interpolation. The deviation is small for short movements but increases with longer distance between robot targets.
Using **SingArea \Off**	See *Manual modifications on page 33*.

Continues on next page

Continued

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

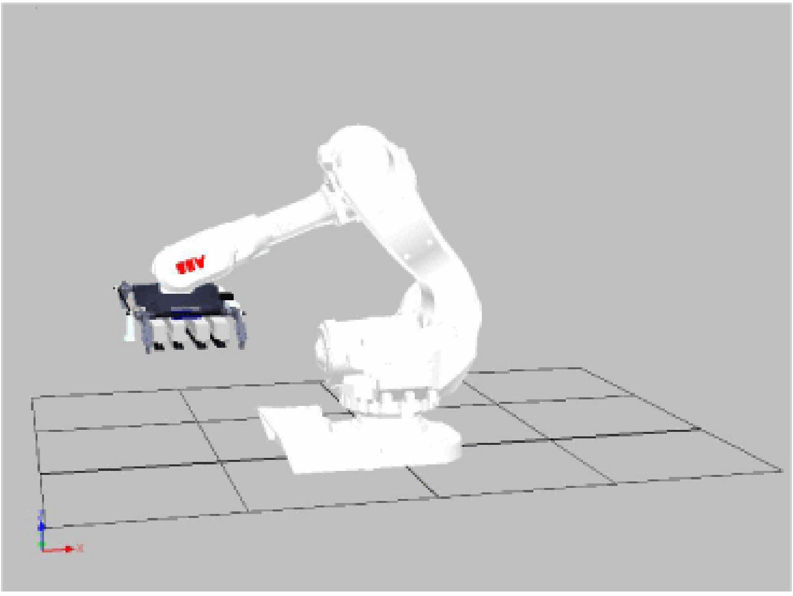
Configuration	Description
Using **SingArea \Wrist**	Not recommended.
Using **SingArea \Off**	See *Manual modifications on page 33*.

Continues on next page

Continued

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 33*.

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 34*.	Linear motion supervision must be turned off to allow movements with large axis reorientations (> 90°).
2.	Use a robot position for each work area with `MoveJ`. In the line configuration, record a position for each work area. In the project configuration, select a robot position and `MoveJ` for each position source.	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.

Continues on next page

Continued

	Action	Note
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 34*.	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 198*.

Note

The following modifications must be done in the program code:

- `ConfL\Off`
- `ConfJ\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 31* and *Wrist is tilted on page 32* 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`.

Continues on next page

Continued

pmMain.mod

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

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

Related information

The Format Operation Set Configuration on page 129.

The Pallet Pattern Operation Set Configuration on page 130.

Public system module pmrcUser on page 196.

2.5 Quick starting a new PickMaster palletizing project

Introduction to the quick start project

There is a quick start project included in the PickMaster installation so you can get a PickMaster project up and running in short time. The purpose is to make it easy for new users to PickMaster to get started and an overview of the functionality. The project is designed for simulation and overview.

This quick start project consists of two flows palletizing boxes on two outfeeders. The pallet stack is shared between the two flows. The robot is mounted on a pedestal, approximately one meter above the floor.

The project can easily be modified to fit other configurations.

About folders and files

The quick start project is located in the PickMaster folder, in a folder called **QuickStart**. The PickMaster installation folder is usually located under ProgramFiles\ABB Industrial IT\Robotics IT\

Create a new folder on your computer and name it **QuickStart**, this folder is known as your *local***QuickStart** folder. This is where you save the quick start project. You can also save the robot system here. If you use Windows Vista you should not create the local **QuickStart** folder in the ProgramFiles folder.

Quick starting a new palletizing project

Use this procedure to quick start a new PickMaster palletizing project.

	Action
1	Start RobotStudio.
2	Use the **System Builder** to create a system. Use an IRB660 and select the option *Prepared for PickMaster* with the sub-option *PickMaster 5*.
3	Start your system in RobotStudio.
4	Copy the files in folder **QuickStart** in the PickMaster installation to your local **QuickStart** folder.
5	Remove the write-protection on all files in the local **QuickStart** folder.
6	Start the FlexPendant and load the module *pmrcUser660.sys* from your local **QuickStart** folder. The module contains functioning tool and work object definitions.
7	Start PickMaster.
8	Import the line *QuickStartLine.pmline* from your local **QuickStart** folder.
9	Connect the line to the running system.
10	Close the line and open the project *QuickStartProject.pmproj* from your local **QuickStart** folder.
11	Transfer the project to the controller.
12	Set the controller in motors on state.
13	Open the project *QuickStartProject* from the PickMaster FlexPendant interface.
14	Start the project and flows from the PickMaster FlexPendant interface.

3 Hardware and software

3.1 Introduction

Structure of this chapter

This chapter describes how to install PickMaster and the requirements of the hardware. The following is described:

- The PC requirements and setup.
- How to install PickMaster on your computer.
- How to request and install a license for the PC software.
- Basic controller information.

3.2 Computer requirements

The computer

The following is required of the computer that will run PickMaster:

Parts	Minimum requirement
Screen resolution	1280*1024 pixels
Operating system	Windows XP SP2

3.3 Network settings

Overview

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.

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.

3.4 Installing PickMaster

General

The computer software is delivered from your local ABB office. The package contains all the software needed to install PickMaster and some additional software that may be useful.

Prerequisites

You need administrator rights on the computer.

Note

Any old versions of PickMaster must be uninstalled before you can install the new version of PickMaster.

Installing PickMaster

	Action	Note
1	Log on with a user account that has administrator rights.	
2	Double-click the file *setup.exe*, which is located in the folder PickMaster.	This opens an installation guide, which will guide you through the rest of the installation.
3	Follow the instructions in the wizard.	

Additional software

The PickMaster package contains the following additional software that is not supported by ABB:

Software	Description
Adobe Acrobat Reader	Reads the .pdf files. Can be found in the folder *AdobeReader*.

3.5 Getting and installing a license key for PickMaster

3.5.1 License key

Introduction to licensing

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

Why a license?

Without a license the functionality in PickMaster is very limited.

Without an installed license:

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

Information about the current license

To get information about the current license:

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

3.5.2 Activating a license key automatically over the Internet

Activating a license key

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

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

3.5.3 Activating a PickMaster license manually

Introduction to manual activation

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

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

Activating a license manually

Use this procedure to activate a PickMaster license manually.

	Action
1	To start the licensing application either use: • PickMaster, on the **Tools** menu, click **Licensing**. • **Windows Start** menu, point to **Programs**, **ABB Industrial IT**, **Robotics IT**, **PickMaster** and click **Licensing**.
2	In the licensing application, click **PickMaster License Activation Wizard**.
3	Under **Automatic Activation**, select **Step 1: Create a license request file** and click **Next**.
4	Enter your 25 character Activation Key (xxxxx-xxxxx-xxxxx-xxxxx-xxxxx) and click **Next**.
5	Click **Save Request**.
6	Type a name for a license request file (*.licreqx), browse to a suitable folder, and click **Save**.
7	Click **Finish**.
8	Use a removable medium, such as a USB device, to transfer the license request file to a computer with an Internet connection.
9	On the computer with Internet connection, start an Internet browser and go to http://www101.abb.com/manualactivation and follow the instructions to activate your license manually. You will be instructed to browse for the saved license request file. The result will be a license file (*.bin) that you must save.
10	Transfer the license file to the PickMaster PC.
11	On the PickMaster computer, start the licensing application.
12	Under **Automatic Activation**, select **Step 3: Install a license file (*.bin)** and click **Next**.
13	Follow the wizard instructions. The PickMaster license will now be activated and the PickMaster installation ready to use.

3.6 Robot controller software

RobotWare software

To run PickMaster projects on an IRC5 robot controller the *RobotWare option Prepared for PickMaster*, which includes the *PickMaster 5* sub-option, is required.

Related information

References on page 10.

4 Navigate and handle PickMaster

4.1 Introduction

Navigate in PickMaster

This chapter describes how to navigate in PickMaster. 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 window layout provides an overview of the menus, toolbars, and windows in PickMaster.

4.2 General

4.2.1 The user interface

The user interface

The PickMaster program is built as a studio environment where all functionality can be accessed from within the same program. See *The main window on page 47*.

The Line view and the Project view

The line view and the project view are the central areas of the program. All objects are created and configured from here.

Pop-up menus

Every object in the line and project views, as well as the views themself, can show a pop-up menu with a set of commands that can be performed on that object. This menu is shown by clicking on the object with the right mouse button. The pop-up menu on the view itself is used to create the various objects.

Common commands available on most pop-up menus:

- Renaming objects.
- Showing or hiding object names.
- Removing objects.

Double-clicking an object performs the most common command for that object. Some commands are also available from the main menu bar.

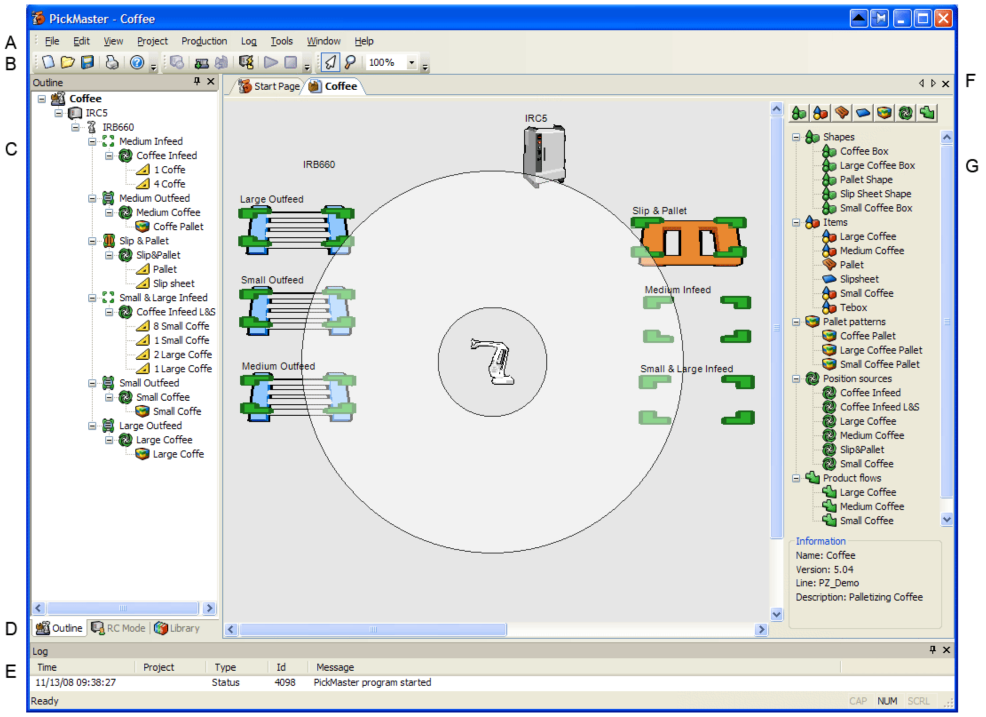
Orientation of objects

The PickMaster objects can be arranged arbitrarily. The orientation of the objects has no impact on the function but it is convenient to arrange the objects according to the physical line.

4.2.2 Layout of the main window

The main window

The main window contains elements like menu bar, toolbar, and browsers.



Name: Coffee
Version: 5.04
Line: PZ_Demo
Description: Palletizing Coffee

```
Time               Project   Type     Id     Message
11/13/08 09:38:27            Status   4098   PickMaster program started
```

xx0600002795

Parts

Part	Description
A	The Menu bar contains menus with commands.
B	The Toolbar contains buttons for quick access to commands in the menus.
C	The Tree view contains views. From here you access different functions and select which part to work with. The same functions are accessible from the workspace.
D	The Tree view tabs show which tree views you have open.
E	The Log window shows status for the PickMaster program.
F	The Windows tabs show which windows you have open in the workspace.
G	The Workspace displays the active window, for example the start page, a project view, or the line view.

Related information

Tree view on page 48.

The PickMaster Library on page 51.

4.2.3 Tree view

The content

The tree view area contains different views, which are selected by clicking the following tabs at the bottom of the area:

- The *Outline* tab: If the line is open, the *Outline* tab shows the objects in the current line and their connections. If one or several projects are open, the *Outline* tab shows the objects in the open projects and their connections.
- The *RC Mode* shows the available controllers on the network and available projects on each controller. It also shows if there exists an I/O value mapping of projects on each controller.
- The *Library* shows saved and stored configurations of shapes, layouts, tools, and messages.

Related information

The Line view on page 57.

The Project view on page 95.

The PickMaster Library on page 51.

4.2.4 Optional settings for PickMaster

General

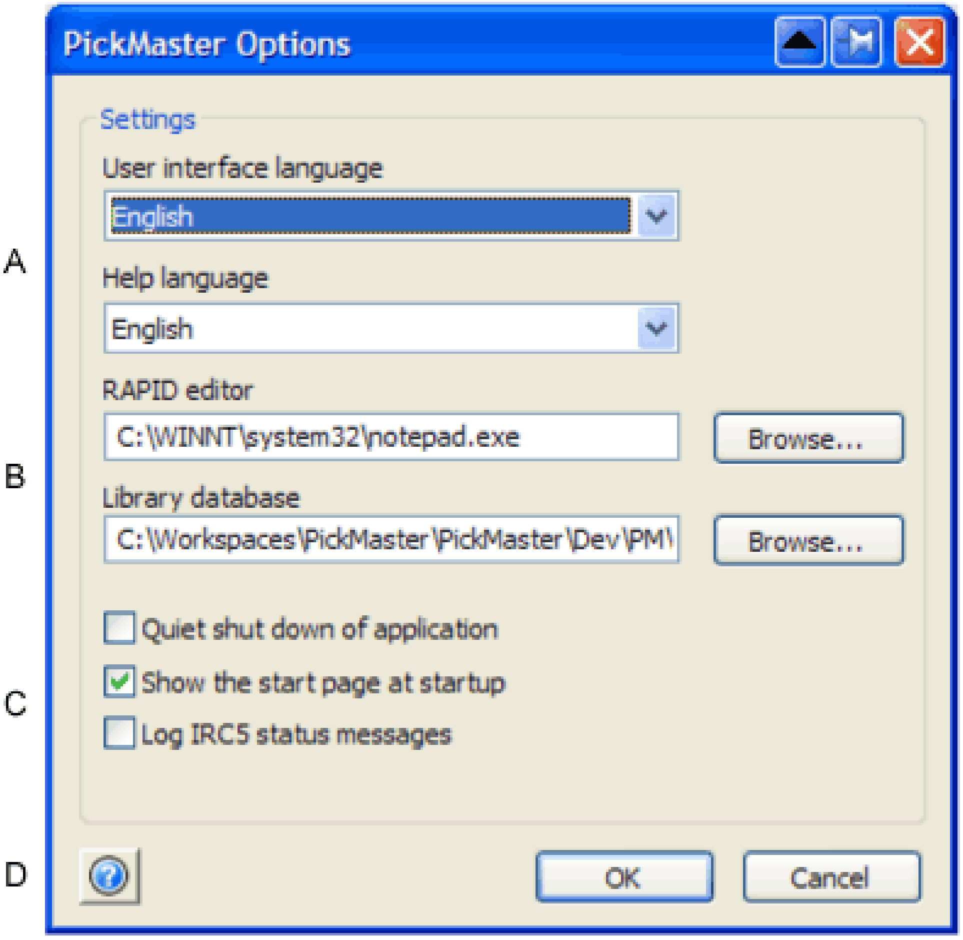
The **PickMaster Options** dialog box helps you to customize PickMaster to your own needs.

Start the PickMaster Options dialog box

To start the PickMaster Options:

1 On the **Tools** menu, click **Options**.

Illustration, PickMaster Options



PickMaster Options

Settings

User interface language: English

Help language: English

RAPID editor: C:\WINNT\system32\notepad.exe

Library database: C:\Workspaces\PickMaster\PickMaster\Dev\PM\

□ Quiet shut down of application

☑ Show the start page at startup

□ Log IRC5 status messages

xx0500002543

Parts

A	The **User interface language** combo box specifies the user interface languages. The **Help language** combo box specifies the language of the online help file.
B	The **RAPID editor** text box specifies which editor to use when editing RAPID programs. The **Library database** text box specifies the resource data base, which is used for saving data, for example, about robot tools or palletizing layouts.

Continues on next page

Continued

C	The **Quiet shut down of application** check box specifies if the PickMaster program will shut down or not even if a project is running. The check box is cleared by default. This means that you get a warning and prevention action if you try to shut down the PickMaster program while a project is running. Before the program can be closed, you must stop all the projects. If the check box is selected, PickMaster will stop any running project and shut down within 15 seconds. The **Show the start page at startup** check box specifies if the start page will appear or not when starting PickMaster. The **Log IRC5 status messages** check box specifies if status, and warning and error messages for the controller will be displayed or not in the log area. If the check box is cleared, only warning and error messages are displayed.
D	The **Help** button opens the context sensitive help for the PickMaster Options dialog box. The **OK** button closes the PickMaster Options dialog box and applies any changes made to it. The **Cancel** button closes the PickMaster Options dialog box and cancels any changes made to it.

4.2.5 The PickMaster Library

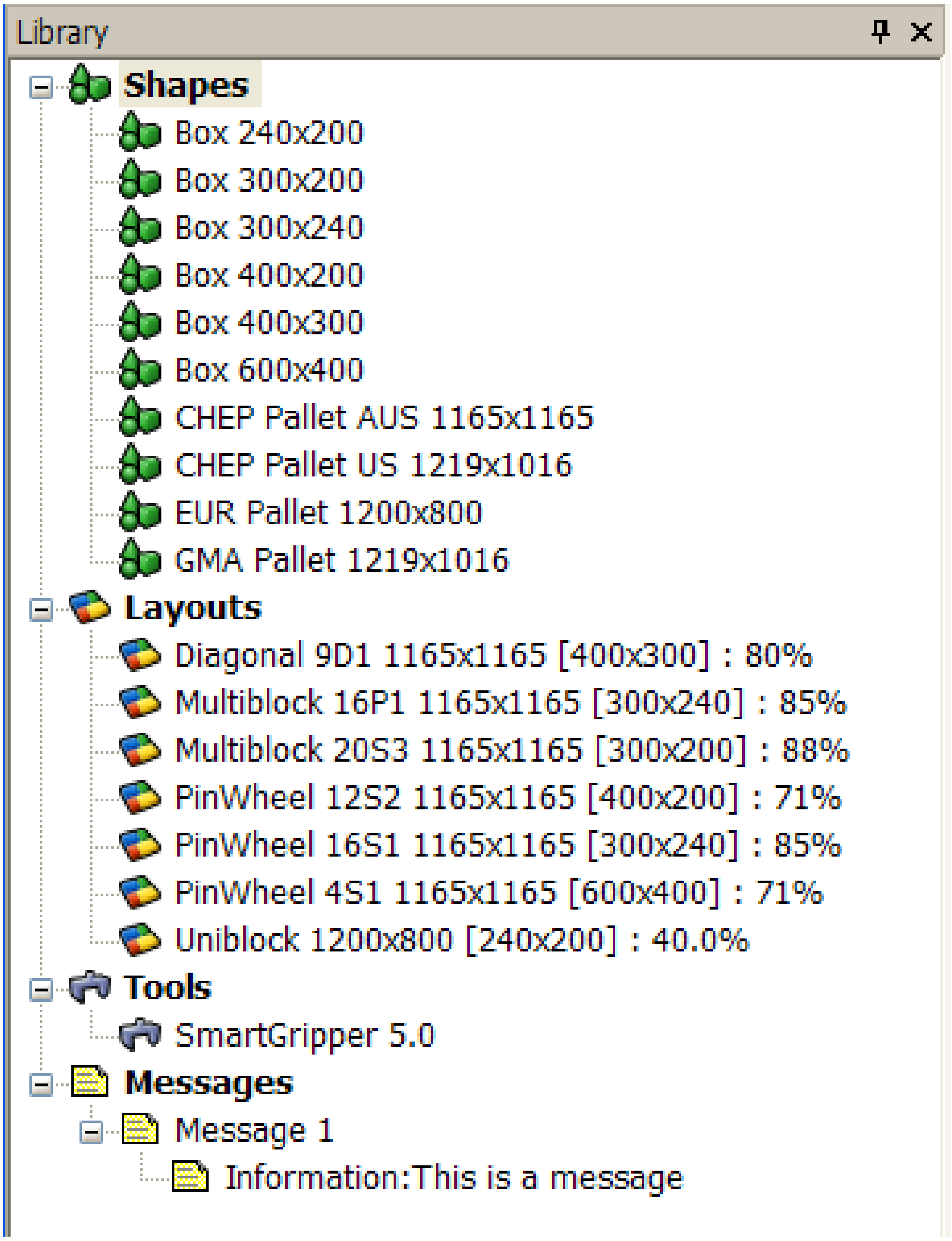
Overview

The Library is a database where you save specific PickMaster configurations. You can retrieve shape, layout, and tool configurations stored in the library from any line or project. The library is accessible from a tree view.

Continues on next page

Illustration, Library

The library tree view lets you interact with the library.



Library

- Shapes
  - Box 240x200
  - Box 300x200
  - Box 300x240
  - Box 400x200
  - Box 400x300
  - Box 600x400
  - CHEP Pallet AUS 1165x1165
  - CHEP Pallet US 1219x1016
  - EUR Pallet 1200x800
  - GMA Pallet 1219x1016
- Layouts
  - Diagonal 9D1 1165x1165 [400x300] : 80%
  - Multiblock 16P1 1165x1165 [300x240] : 85%
  - Multiblock 20S3 1165x1165 [300x200] : 88%
  - PinWheel 12S2 1165x1165 [400x200] : 71%
  - PinWheel 16S1 1165x1165 [300x240] : 85%
  - PinWheel 4S1 1165x1165 [600x400] : 71%
  - Uniblock 1200x800 [240x200] : 40.0%
- Tools
  - SmartGripper 5.0
- Messages
  - Message 1
    - Information:This is a message

xx0600002762

Shapes	Lists item shapes.
Layouts	Lists different layouts of shapes in a layer.
Tools	Lists robot tools.
Messages	Lists different messages.

Continues on next page

Available command for library groups

The following command is available for the library groups. Right-click to select command.

Command	Description
Add	Add a new item.

Available commands for items

The following commands are available for all items in the library tree view. Right-click to select command.

Command	Description
Edit	Edit an item. Opens a new dialog box.
Rename	Rename an item.
Delete	Delete an item. A dialog box appears where you confirm the deletion. A deletion cannot be undone.
Duplicate	Duplicate an item. A new item with the same settings is added.

Limitations

The library is only accessible from PickMaster.

Related information

The Layout Editor on page 120.

*The **Tool Configuration** on page 81*.

Terminology on page 24.

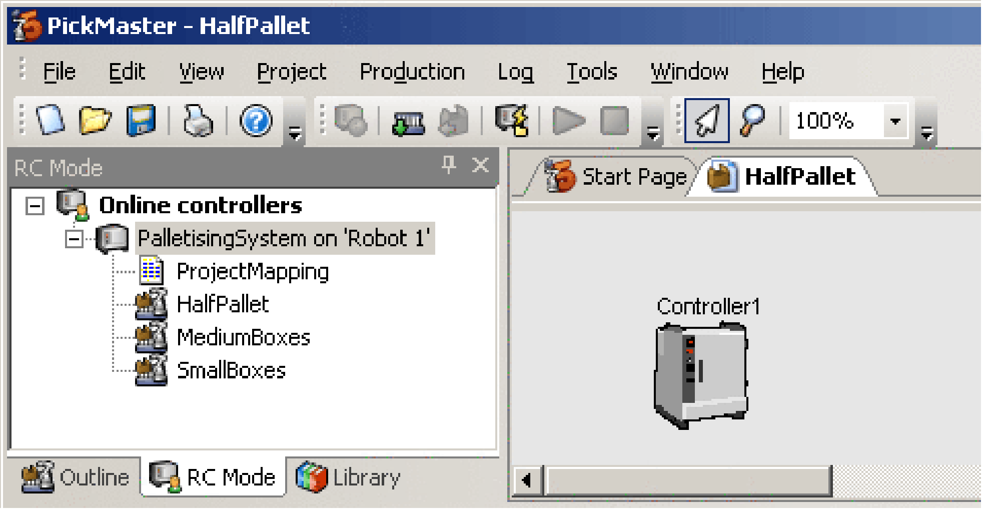
4.2.6 Setting up Project I/O values

Overview

If you want to start PickMaster 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.

If a project I/O value configuration has been downloaded to a controller, a *ProjectMapping* object can be seen at the top of the controller's project list in the **RC Mode** tab.



en0800000395

Prerequisites

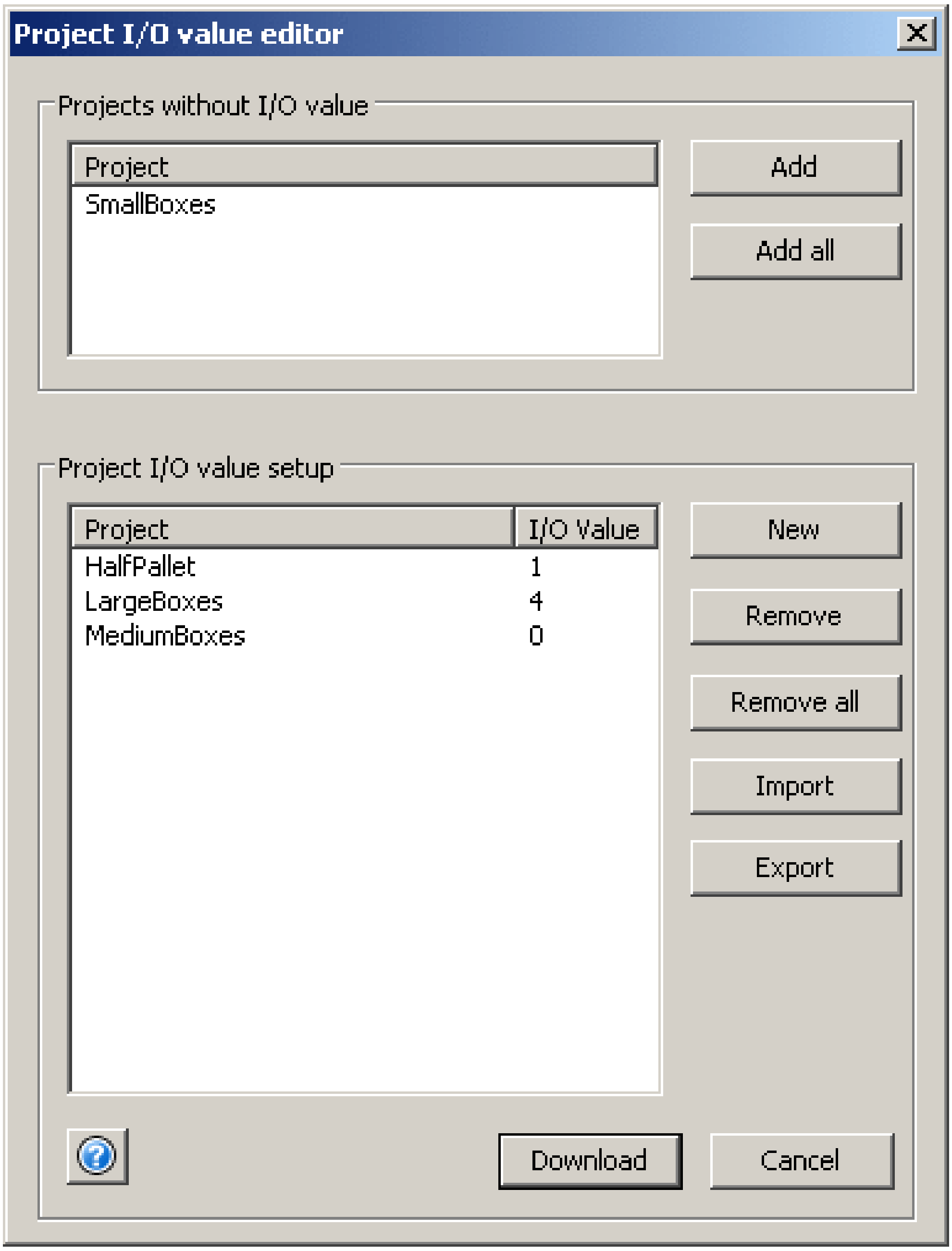
The controller must be online when you configure project I/O values.

Starting the Project I/O value editor

	Action
1	Select the **RC Mode** tab in the tree view.
2	Right-click the controller and select **Edit project I/O values**. The project I/O values are uploaded from the controller and the **Project I/O value editor** dialog appears.

Continues on next page

Illustration, Project I/O value editor



Project
SmallBoxes

Project	I/O Value
HalfPallet	1
LargeBoxes	4
MediumBoxes	0

en0800000373

Projects without I/O value	Here you see projects available on the controller that not has been assigned I/O values.
Project I/O value setup	Here you see projects that have been assigned I/O values. Some of these projects might be available on the controller. Others might not be available (that is, created or imported projects or projects that have been deleted from the controller). The project names and I/O values can be edited. The project names and I/O values must be unique.

Continues on next page

Projects without I/O values

Part	Description
Add	Add the project to the **Project I/O value setup** list. The lowest possible unique I/O value is automatically set for the added project.
Add all	Add all projects to the **Project I/O value setup** list. The lowest possible unique I/O values are automatically set for the added projects.

Project I/O value setup

Part	Description
New	Create a new project in the list. The project will get a unique default name and a unique I/O value.
Remove	Remove the project from the list. If the project is available on the controller it will show in the **Projects without I/O values** list. The removed project I/O value can now be used by another project.
Remove all	Remove all projects in the list. The projects that are available on the controller will show in the **Projects without I/O values** list.
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.
Download	Exit the dialog and download the project I/O value setup to the controller.
Cancel	Exit the dialog without downloading the I/O value setup to the controller.

4.3 Line

4.3.1 The Line view

Overview

To be able to run a project with PickMaster, a line must be defined. A line describes the configuration of the physical line. As long as the equipment in the physical line has not changed, there is no need to edit the line.

A line is saved as a *.pmline* file.

Prerequisites

To be able to work with a line, the line file must be registered and all projects must be closed.

Registering the Line file

To work with a line the line file must be registered on the computer from which you run the project. When you create a new line it is automatically registered, but if for example, you want to copy a line file from another computer, you need to register the line before you can start working with it. To register the line file, you have to import it.

Open the Line view

To...	On the File menu, click...
create a new line	**New Line**
edit an existing line	**Edit Line**
import a line copied from another computer	**Import Line**

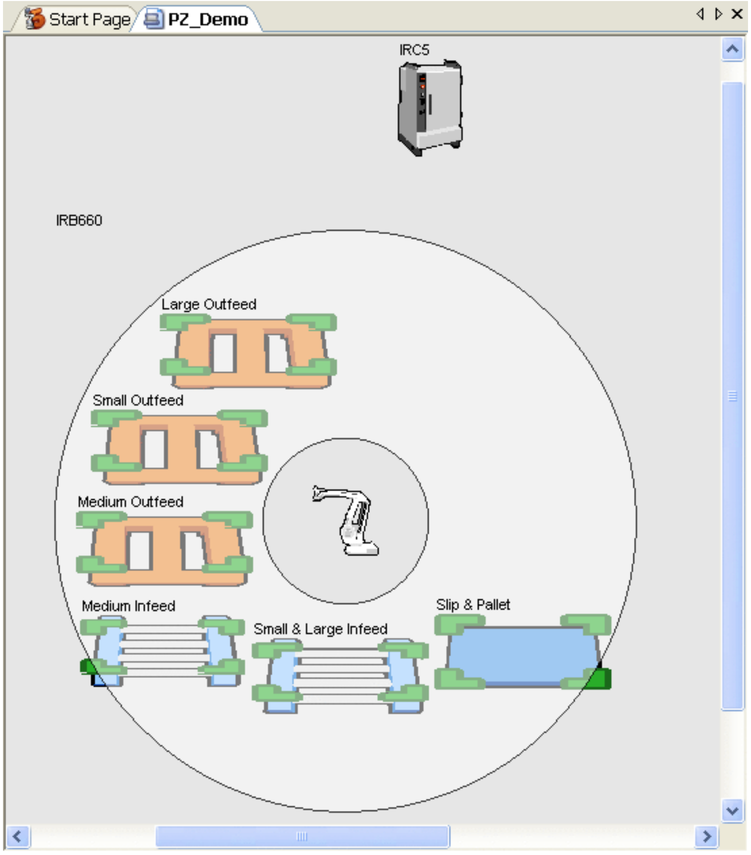
When a project is open

To...	On the File menu, click...
open the corresponding line for editing.	**Open Current Line**
return to the project	**Open Previous Projects**

Continues on next page

Illustration, Line

The PickMaster workspace lets you interact with the line.



xx0600002820

Robot controller	The robot controller belongs to the PickMaster line and can only be created and removed from the line.
Robot	The robot belongs to the PickMaster line and can only be created and removed from the line.
Work area	The work area belongs to the PickMaster line and can only be created and removed from the line.
Tool	The tool belongs to the PickMaster line and can only be created and removed from the line

How to proceed

This section describes how to use the different commands in the Line view.

Add a controller to the line

	Action
1.	In the Line view, right-click in the workspace and click **New IRC5 Controller**. Or On the **Line** menu, click **New IRC5 Controller**.

Continues on next page

	Action
2.	The **Select IRC5 Controller** dialog box appears. You can select an online or a template controller: • Online controller; select the **Controller Selection** check box and select a controller. • Template controller; select the **Template controller** check box and select a **Robot** check box. For more information, ***The Select IRC5 Controller** on page 61*.
3.	Click **OK**. A controller graphic appears on the workspace of the Line view.

All controllers that are available on the network appear in the **Online controllers** list. If the controller you intend to use is not available on the network, you can create a template controller. You can also select the number of robots your template controller will use.

Add a robot to the line

The robots are closely connected to the controller. To create a new robot the controller must include robots that are not already in the project.

	Action
1.	In the Line view, right-click the controller and click **Edit**.
2.	In the **Robot** list, select a **Robot** check box to select a new robot. If all check boxes are selected, you must first create a new controller and then create a new robot.

Add a tool to the line

The tool has a one-to-one relationship to the robot - that is, a robot always has one tool. You can edit the tool, change it by importing a tool from the PickMaster Library or create a new one.

	Action	Note
1.	In the **Line** view, right-click the robot and point to **Tool**.	
2.	To edit the tool, click **Edit**. To create a new tool, click **New**. To import a tool from the PickMaster Library, click **Import**.	When creating a new tool or importing one from the PickMaster Library, the current tool will be removed.

Add a work area to the line

The work area is created from a robot. To create a work area there must be at least one robot in the line. The created work area is automatically connected to the robot from which it was created, but this can be changed later on.

	Action
1.	In the Line view, right-click a robot.
2.	Click **New Work Area**.

Related information

The Line and Project concept on page 20.

The IRC5 Configuration on page 63.

Continues on next page

Setting up the Line on page 164.

The ***Tool Configuration*** *on page 81*.

The Work Area Configuration on page 73.

4.3.2 The Select IRC5 Controller

Overview

This section describes how to use the **Select IRC5 Controller** window, and select which controller is to be used.

Start the Select IRC5 Controller window

You start the **Select IRC5 Controller** window in different ways depending on if you intend to add a new or select an existing controller.

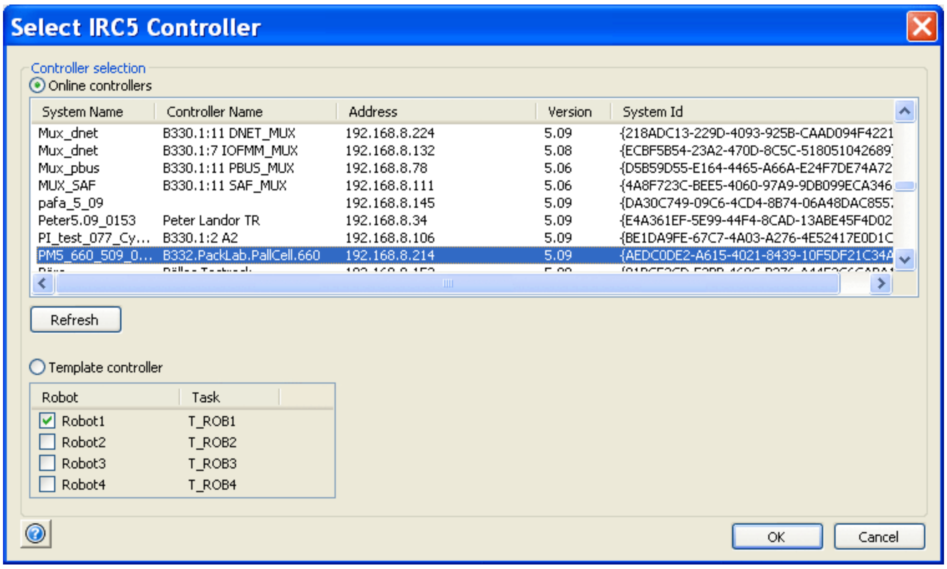
To add a new controller

	Action	See
1.	In the Line view, right-click in the workspace and select **New IRC5 Controller**.	
2.	The **Select IRC5 Controller** window appears, and you can select an online or a template controller.	*Add a controller to the line on page 58*.

To select an existing controller

	Action	See
1.	In the Line view, right-click on an existing controller and click **Edit**.	
2.	The **IRC5 Configuration** window appears. To start the **Select IRC5 Controller** window, click **Change**.	*The IRC5 Configuration on page 63*.

Illustration, Select IRC5 Controller dialog box



● Online controllers

System Name	Controller Name	Address	Version	System Id
Mux_dnet	B330.1:11 DNET_MUX	192.168.8.224	5.09	{218ADC13-229D-4093-925B-CAAD094F4221
Mux_dnet	B330.1:7 IOFMM_MUX	192.168.8.132	5.08	{ECBF5B54-23A2-470D-8C5C-518051042689
Mux_pbus	B330.1:11 PBUS_MUX	192.168.8.78	5.06	{D5B59D55-E164-4465-A66A-E24F7DE74A72
MUX_SAF	B330.1:11 SAF_MUX	192.168.8.111	5.06	{4A8F723C-BEE5-4060-97A9-9DB099ECA346
pafa_5_09		192.168.8.145	5.09	{DA30C749-09C6-4CD4-8B74-06A48DAC855
Peter5.09_0153	Peter Landor TR	192.168.8.34	5.09	{E4A361EF-5E99-44F4-8CAD-13ABE45F4D02
PI_test_077_Cy...	B330.1:2 A2	192.168.8.106	5.09	{BE1DA9FE-67C7-4A03-A276-4E52417E0D1C
PM5_660_509_0...	B332.PackLab.PallCell.660	192.168.8.214	5.09	{AEDC0DE2-A615-4021-8439-10F5DF21C34A

○ Template controller

Robot	Task
☑ Robot1	T_ROB1
□ Robot2	T_ROB2
□ Robot3	T_ROB3
□ Robot4	T_ROB4

xx0600002998

Online controllers	Lists all available controllers on the network.
Template controller	Used to create a template controller.

Continues on next page

How to proceed

This section describes how to proceed in the **Select IRC5 Controller** window.

Select a controller

In the **Select IRC5 Controller** window you select which controller to use. All controllers that are available on the network will automatically pop up in the online controllers list.

To...	Do this
Select an existing controller	Select the **Online controllers** check box. • From the controllers list, select the desired controller
Create a template controller	Select the **Template controller** check box. • Select the number of robots that the template controller will use. • If needed, change task name for each robot.

Related information

The IRC5 Configuration on page 63.

4.3.3 The IRC5 Configuration

Overview

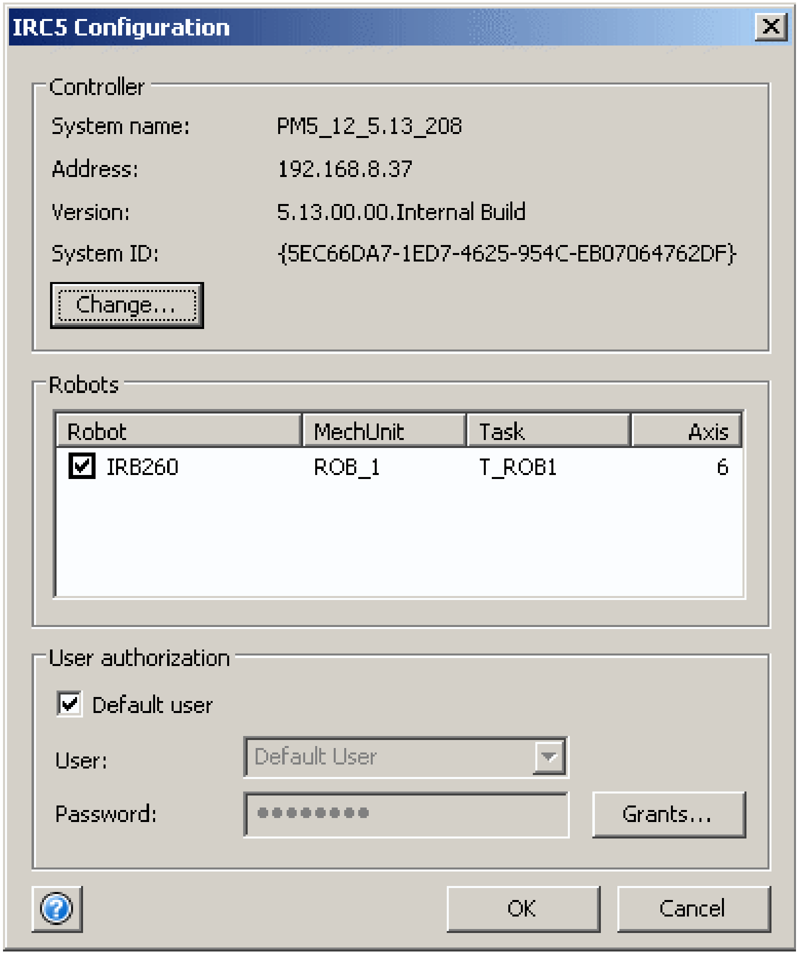
In the **IRC5 Configuration** window you make the configuration for the IRC5 robot controller.

Start the IRC5 Configuration

In the workspace of the Line view

- double-click the controller graphic, or
- right-click the controller graphic and click **Edit**.

Illustration, IRC5 Configuration



IRC5 Configuration

Controller

System name: PM5_12_5.13_208

Address: 192.168.8.37

Version: 5.13.00.00.Internal Build

System ID: {5EC66DA7-1ED7-4625-954C-EB07064762DF}

Robots

Robot	MechUnit	Task	Axis
☑ IRB260	ROB_1	T_ROB1	6

User authorization

☑ Default user

User: Default User

Password: ●●●●●●●●

xx0600002821

Controller	Here you get information about the selected controller. The available information is the system name, IP address, the RobotWare version and the system ID.
Robots	Here you get information about available robots for the controller.
User authorization	Here you select which user authorization rights to use when logging on to the controller. Certain grants are needed to be able to run PickMaster. For more information about grants, see *Operating manual - RobotStudio*.

Continues on next page

How to proceed

This section describes how to proceed in the **IRC5 Configuration** window.

Change a controller

To transfer a project to the controller, the correct controller must be selected. If you have selected the wrong controller, you must change it.

Note

In the following cases you have to select the correct controller:

- If you have selected a controller which you did not intend to select.
- If the line is created on a template controller.
- If RobotWare has been updated on the controller after the controller was created.

To change a controller:

	Action
1.	In the **IRC5 Configuration** dialog box, click **Change**.
2.	The **Select IRC5 Controller** dialog box appears. Select the **Controller Selection** check box and select a controller. **Note** If the selected controller has fewer robots than were used in the previously used controller, or, if the task names are different, the **New Task For Existing Robot** dialog box will appear. In the drop-down combo box, select which of the existing robots that should be mapped to the selected controller and click **OK**.
3.	In the **Select IRC5 Configuration** dialog box, click **OK**.

Select which robot to use

To...	Do this
Select which robot to use	In the **Robot** list, select a **Robot** check box.
Rename a robot	In the **Robot** list, click the name of the robot to rename and type a new name.

Change user

To...	Do this
Change user	In the **User authorization** area 1 Clear the **Default user** check box. 2 In the **User** drop-down combo box, select a user. 3 In the **Password** text box, type the password.
See the grants of the selected user	Click **Grants**.

Note

If you have not changed the user on the controller, you can use the default user, which has the required grants.

Continues on next page

Continued

Related information

The Line view on page 57.

4.3.4 The Controller Properties window

Overview of the Controller Properties window

In the **Controller Properties** window you define controller related values, that is, general settings, event settings and work area order settings.

The general settings define some PickMaster process related controller properties.

The event setttings define an I/O signal interface that can be used by a PLC to report errors for PickMaster related objects, for example work areas, to the controller. An event setting can also be used to generate tailored elog messages.

The work area order settings define the physical order of the work areas, that is, in which order the robot will pass the work areas when making long movements.

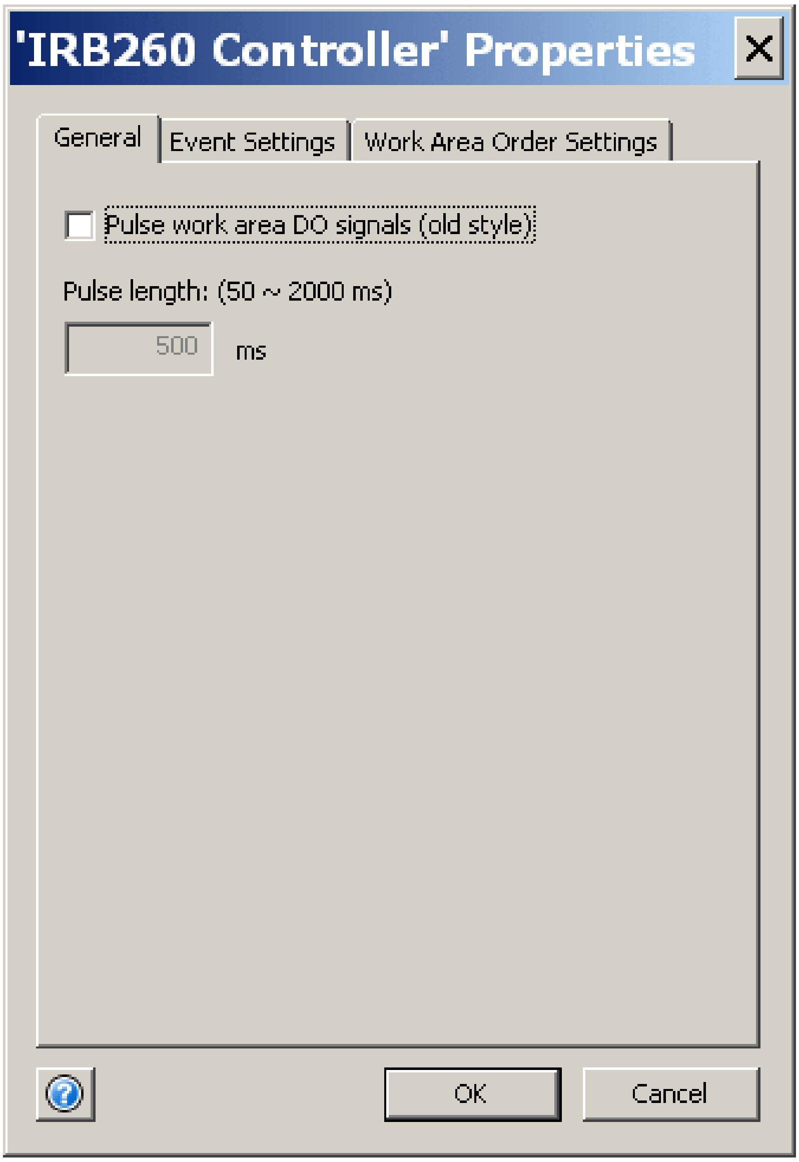
Start the Controller Properties

	Action	Info
1	In the workspace of the Line view, right-click the controller graphic and click **Properties**.	
2	The **Controller Properties** window appears and you can edit the values.	See *Illustration, Event Settings on page 68*.

Continues on next page

Continued

Illustration, General Settings



'IRB260 Controller' Properties

□ Pulse work area DO signals (old style)

Pulse length: (50 ~ 2000 ms)

500 ms

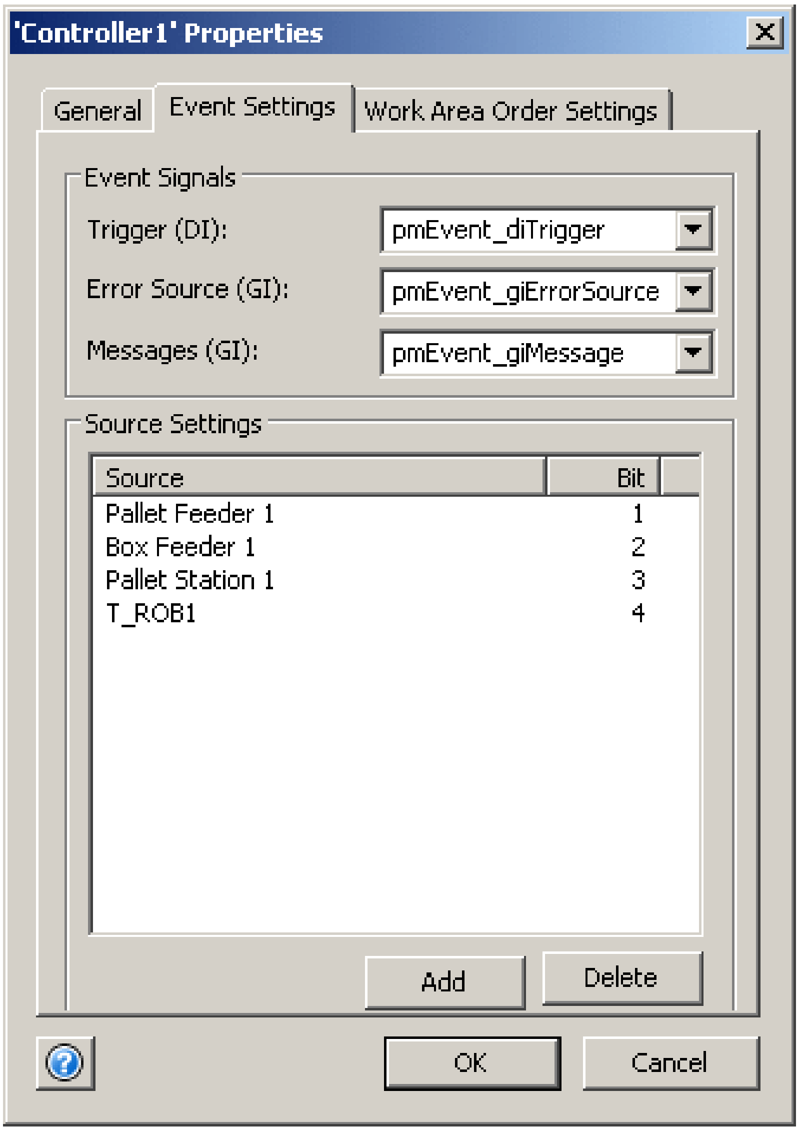
xx0700000401

Pulse work area 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 work area DO signals** is selected. The pulse length can be set in the range of 50 ms to 2000 ms.

Continues on next page

Continued

Illustration, Event Settings



Trigger (DI): pmEvent_diTrigger

Error Source (GI): pmEvent_giErrorSource

Messages (GI): pmEvent_giMessage

Source	Bit
Pallet Feeder 1	1
Box Feeder 1	2
Pallet Station 1	3
T_ROB1	4

en0800000284

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

The Signals part

Part	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.

Continues on next page

Part	Description
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 (work area, 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 *The Source Settings part on page 69*.
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. You define the messages in the line view of PickMaster, see *The Message Settings on page 149*.

The Source Settings part

The **Source Settings** part needs to be configured if the **Error Source (GI)** signal in the **Signals** part is used. See *Flow recovery on page 223*.

Part	Description
Source	Specifies the work area, 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 work area, then the work area will be set in error state when the **Trigger (DI)** peaks. The current state of a work area is indicated by the FlexPendant interface, see *Viewing flow status on page 219*. It is also indicated by the work area I/O signal Execution state (GO), see *The Work Area Configuration on page 73*.

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

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

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

If you plan to use *Redo last pick* then we recommend that you set the place work area in error state rather than the pick work area. Setting the pick work area 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.

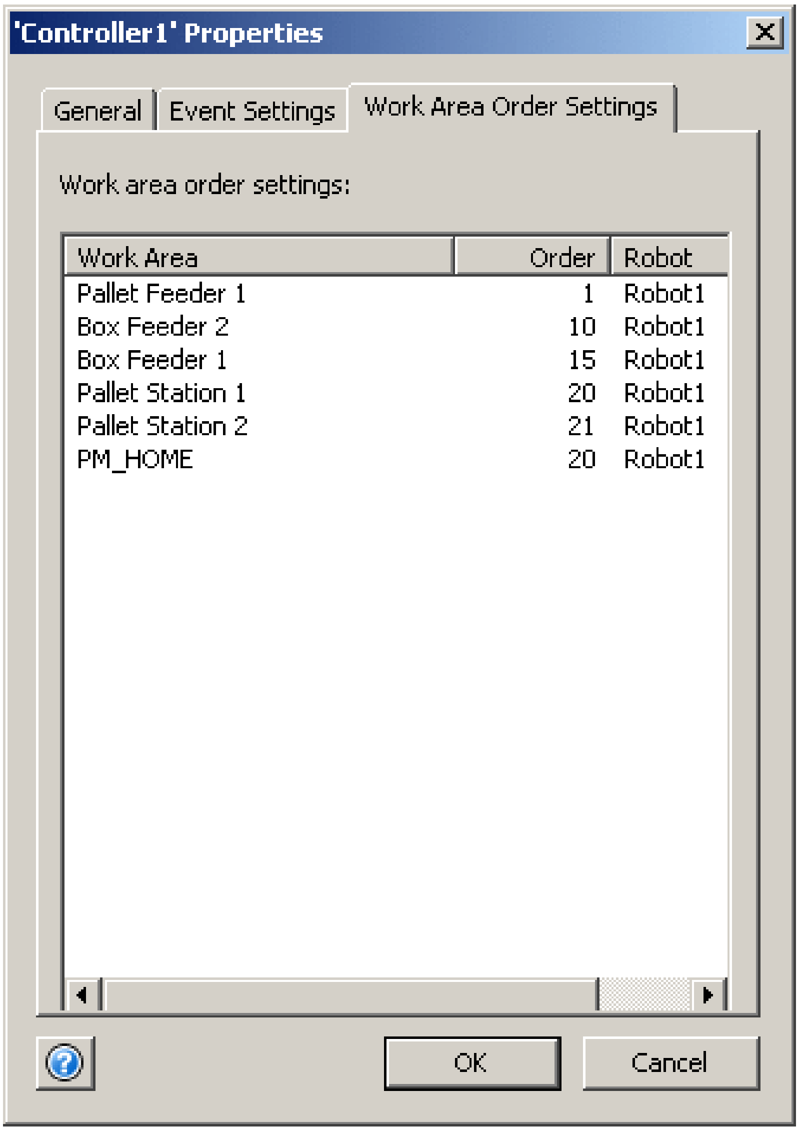
Flow recovery can be used from the FlexPendant interface or the I/O interface to recover from errors.

Example - Usage of signals

For examples, see *Event and error reporting on page 254*.

Continues on next page

Illustration, Work Area Order Settings



Work area order settings:

Work Area	Order	Robot
Pallet Feeder 1	1	Robot1
Box Feeder 2	10	Robot1
Box Feeder 1	15	Robot1
Pallet Station 1	20	Robot1
Pallet Station 2	21	Robot1
PM_HOME	20	Robot1

en0900000134

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

The work area order settings are used to define which work areas that will be considered when planning the path height of an intermediate movement between two work areas. Also, when moving to or from the home position, the work area order settings will affect the path height. Each work area and the home position are given an order number. When planning an intermediate movement, the order number of the start work area and the end work area (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 work areas to be considered when planning the path height. The order numbers can be freely chosen and two work areas can have the same order.

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

4.3.5 The New Task For Existing Robot

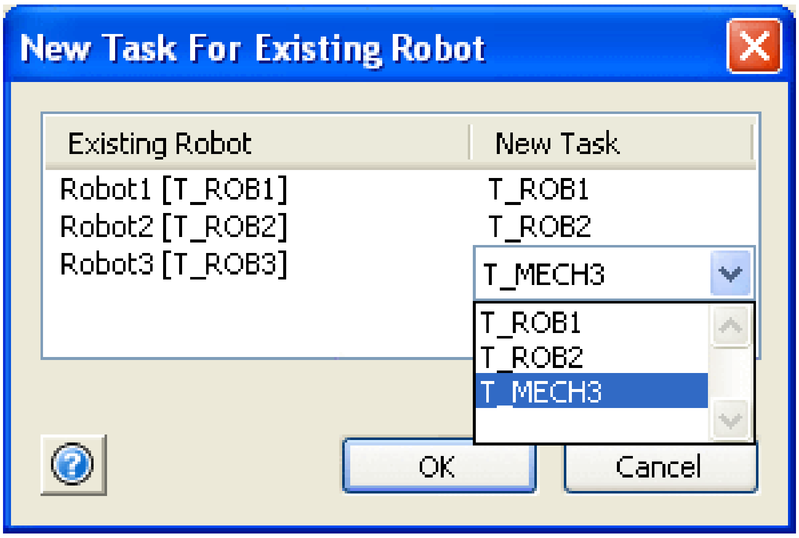
Overview

The **New Task For Existing Robot** window helps you create a map between your robots and the new controller. If the new controller has fewer mechanical units than were used in the old controller, or if the task names are different, you must select which of the existing robots should be mapped to which new task. If this is the case, the **New Task For Existing Robot** window will appear to help map the existing robots to the new controller. If there are robots with the correct task name on the new controller, the mapping will be automatic.

Note

The New Task For Existing Robot window cannot be started by the user. It will automatically appear when changing the controller and there is a mismatch in the task names in the old and new controller.

Illustration, New Task For Existing Robot



New Task For Existing Robot

Existing Robot	New Task
Robot1 [T_ROB1]	T_ROB1
Robot2 [T_ROB2]	T_ROB2
Robot3 [T_ROB3]	T_MECH3

xx0600003001

Existing Robot	List of all robots connected to the old controller and the old task name for each robot within brackets.
New Task	The name of the new task for the robot.

Continues on next page

How to proceed

This section describes how to proceed in the **New Task For Existing Robot** window.

Select new tasks for the old robots

When the **New Task For Existing Robot** window appears there is always a suggested mapping of the tasks.

To...	Do this
Change the selected task for a robot	Click the **New Task** column for the robot to be changed, and select a task from the drop down list.

Note

Each task can only be mapped to one robot.

If you do not select a new task for a robot, that robot will be removed from the line.

4.3.6 The Work Area Configuration

Overview

In the **Work Area Configuration** window you configure existing work areas. A work area must always be related to a robot.

Start the Work Area Configuration

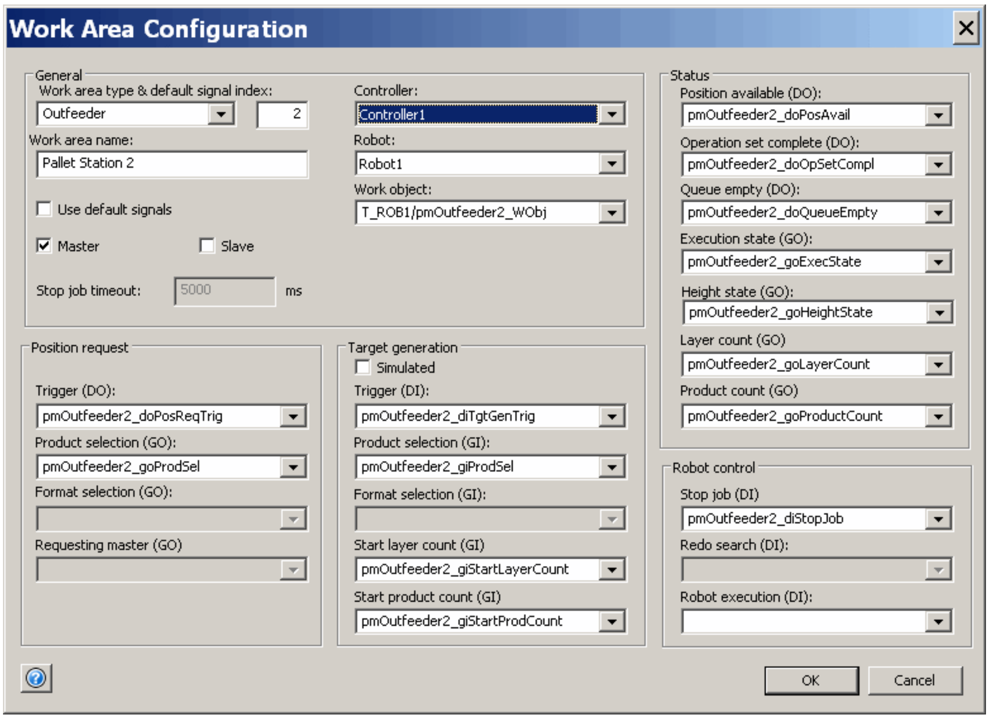
	Action	Info
1	In the workspace right-click the work area symbol and click **Edit**, or double-click the work area symbol.	
2	The **Work Area Configuration** window appears and you can edit the work area.	See *Illustration, Work Area Configuration on page 74*.

Continues on next page

Continued

Illustration, Work Area Configuration

The **Work Area Configuration** window consists of five parts:



Work Area Configuration

General

Work area type & default signal index: Outfeeder 2

Work area name: Pallet Station 2

☐ Use default signals

☑ Master ☐ Slave

Stop job timeout: 5000 ms

Controller: Controller1

Robot: Robot1

Work object: T_ROB1/pmOutfeeder2_WObj

Status

Position available (DO): pmOutfeeder2_doPosAvail

Operation set complete (DO): pmOutfeeder2_doOpSetCompl

Queue empty (DO): pmOutfeeder2_doQueueEmpty

Execution state (GO): pmOutfeeder2_goExecState

Height state (GO): pmOutfeeder2_goHeightState

Layer count (GO): pmOutfeeder2_goLayerCount

Product count (GO): pmOutfeeder2_goProductCount

Position request

Trigger (DO): pmOutfeeder2_doPosReqTrig

Product selection (GO): pmOutfeeder2_goProdSel

Format selection (GO):

Requesting master (GO):

Target generation

☐ Simulated

Trigger (DI): pmOutfeeder2_diTgtGenTrig

Product selection (GI): pmOutfeeder2_giProdSel

Format selection (GI):

Start layer count (GI): pmOutfeeder2_giStartLayerCount

Start product count (GI): pmOutfeeder2_giStartProdCount

Robot control

Stop job (DI): pmOutfeeder2_diStopJob

Redo search (DI):

Robot execution (DI):

xx0600002978

General	This part contains the name of the work area and which controller, robot, and work object it is connected to. For detailed descriptions, see *The General part on page 74*.
Position request	This part contains the trigger and selection signals, see *The Position request part on page 75*.
Target generation	This part contains the trigger and selection signals. For detailed descriptions, see *The Target generation part on page 76*.
Status	This part contains the status signals. For detailed descriptions, see *The Status part on page 76*.
Robot control	This part contains the robot execution signal. For detailed descriptions, see *The Robot control part on page 76*.

The General part

Part	Description
Work area type & default signal index	The work area type and default signal index specify which default signals that shall be suggested for the work area when selecting **Use default signals** checkbox. The following work area types are available: • Infeeder • Outfeeder • Pallet • Slipsheet If no visualization type is selected for this work area, PickMaster will suggest a visualization type based on the selected work area type when closing this dialog.

Continues on next page

Continued

Part	Description
Work area name	The name of the work area as it will be shown. **Note** The name will also appear on the PickMaster FlexPendant interface.
Master	Filter that simplifies setting up a **master work area**. When selected, only signals and parameters that can be used by a master work area are enabled for updates.
Slave	Filter that simplifies setting up a **slave work area**. When selected, only signals and parameters that can be used by a slave work area are enabled for updates.
Use default signals	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 **Work area type and default signal index**. Clear the checkbox if you want to edit the signal selection manually. See *Default signals on page 238*.
Controller	Specifies to which IRC5 controller the work area is connected. To change the controller, select the desired controller from the **Controller** drop-down box.
Robot	Specifies to which robot the work area is related. To change the robot, select the desired robot from the **Robot** drop-down box.
Work object	Specifies to which work object this work area is connected. To change work object, select a work object from the **Work object** 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.
Stop job timeout	Defines the maximum time a pending position request will be active after a stop job has been started. It is used only by the slave work areas. Should be set to a time that is longer than the time it takes to generate new targets on the work area. For example, longer than the maximum product infeed time.

The Position request part

This part defines the signals used by the PickMaster process when requesting new operation sets. For a detailed description of the signals, see *Basic I/O interface on page 240*.

Continues on next page

The Target generation part

This part defines 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 on page 240*.

Part	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 work areas 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 work area, simulation can be used to auto start the job in a continuos mode when the flow has been started.

The Status part

This part defines different signals to monitor the status of the PickMaster process. For a detailed description of the signals, see *Basic I/O interface on page 240*.

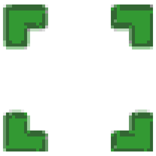
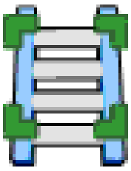
The Robot control part

Defines different signals to affect the PickMaster process. For a detailed description of the signals, see *Basic I/O interface on page 240*.

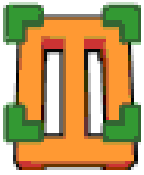
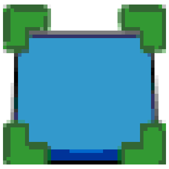
Visualization

You can configure the visualization of the work area to represent a roll conveyor, a slip sheet or a pallet.

When saving the configuration of a new created work area, PickMaster will suggest a visualization image for the work area based on the work area type.

Work area type	Visualization image
Infeeder	 xx0800000403
Outfeeder	 xx0800000404

Continues on next page

Work area type	Visualization image
Pallet	 xx0800000405
Slip sheet	 xx0800000406

To change the visualization:

1 In the workspace, right-click the work area symbol and point to **Visualization**. Select the desired visualization.

Additional information

Note

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

Tip

Work objects and signals are much easier and more convenient to select if the configuration is performed with the controller available on the network. Thus you only need to use the drop-down list to select the desired work object or signal.

Related information

Preparing your controller for PickMaster on page 163.

4.3.7 The Work Area I/O Settings

Overview

In the **Work Area I/O Settings Editor** a unique I/O value can be assigned for each work area. The I/O value is used when performing a flow recovery action from a PLC instead of using the FlexPendant interface.

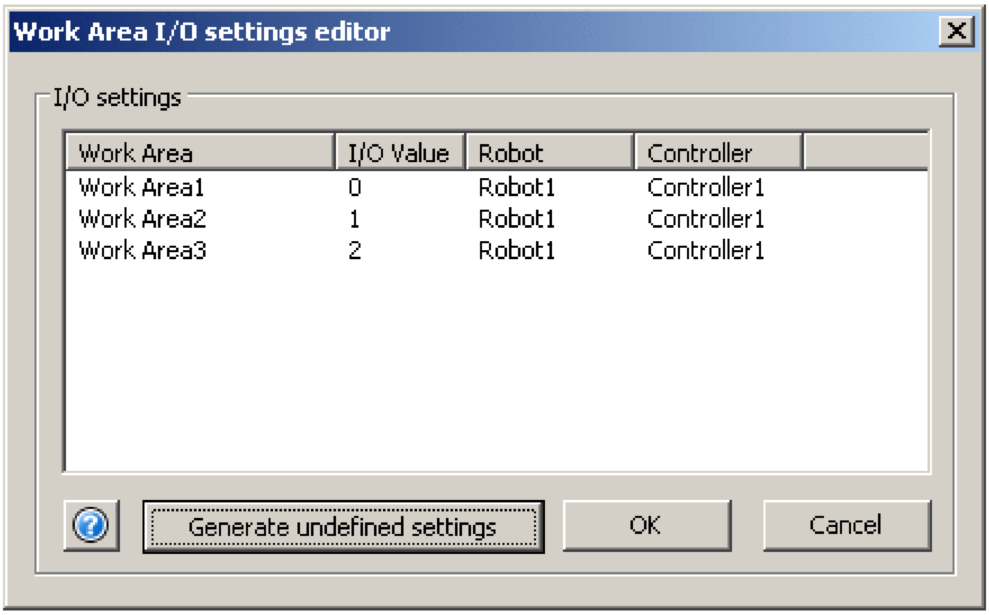
For example when an error has occurred for a flow, an appropriate I/O value is set for the GI signal *pmFlow_giWaRecoverSelection*. The I/O value selects the work area that the recover action shall be applied for.

See *Flow handling on page 251*.

Start the Work Area I/O Settings Editor

	Action
1	On the **Line** menu, select **Work Area I/O Settings**. The **Work Area I/O Settings Editor** appears and you can edit I/O values.

Illustration, Work Area I/O Settings Editor



Work Area	I/O Value	Robot	Controller
Work Area1	0	Robot1	Controller1
Work Area2	1	Robot1	Controller1
Work Area3	2	Robot1	Controller1

en0900000136

I/O Value	Click a value to edit it. Default I/O value is -1 and indicates that an I/O value has not yet been assigned for the work area. All I/O values must be unique!
Generate un-defined settings	Click to automatically assign a unique I/O value to each work area which not yet has been assigned a value (that is, I/O value = -1). Lowest possible free I/O values will be set.

4.3.8 The Work Object Orientation

Overview

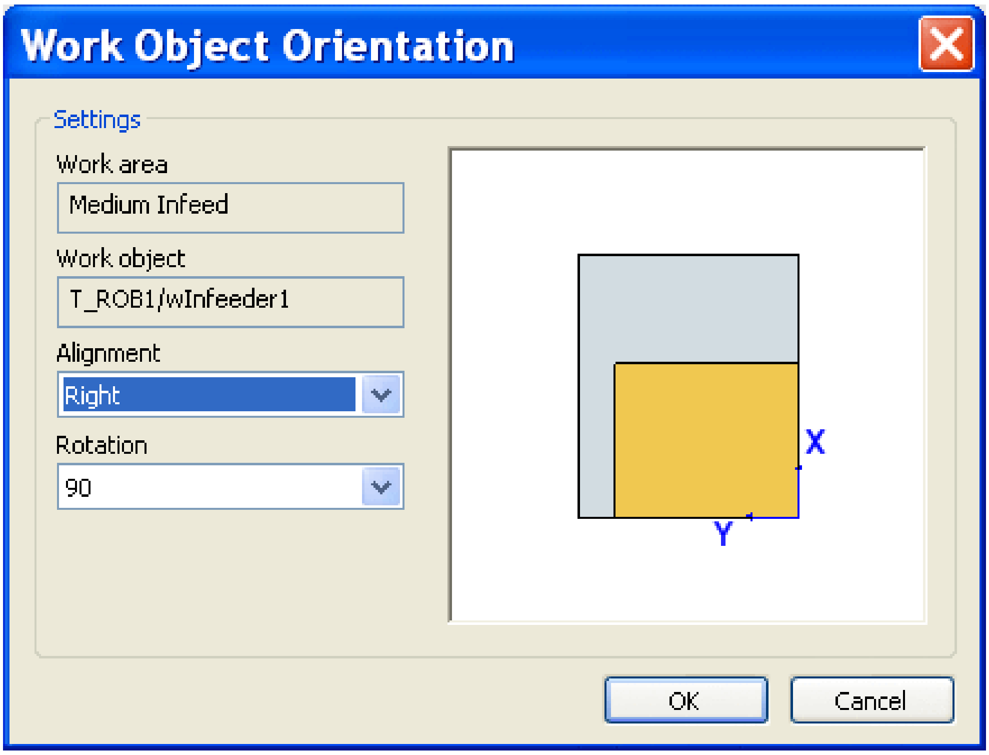
In the **Work Object Orientation** window you define where in the work area the work object is calibrated. The defined location of the work object origin will be shown in several windows when configuring a project.

Start the Work Object Orientation

	Action
1	In the workspace right-click the work area symbol and click **Orientation**.
2	The **Work Object Orientation** window appears and you can edit the work object orientation.

Illustration, Work Object Orientation

The **Work Object Orientation** window contains a few options and a display showing a schematic image of a work area. The image shows 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.



Work Object Orientation

Settings

Work area: Medium Infeed

Work object: T_ROB1/wInfeeder1

Alignment: Right

Rotation: 90

en0600003246

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.

Continues on next page

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 work area used for picking, the work object calibration is especially important. If the picking location of the tool must be adjusted, then update the format location of the tool. Avoid adjusting the displacement frame or tuning the location of the work area, since this might reduce the accuracy when items are placed.

For a palletizing work area or other work area where products are placed, the work object calibration can be adjusted with the displacement frame or by tuning the location of the work area to modify the place location of items.

Related information

The Work Area Configuration on page 73.

The Operation Set Configuration on page 128.

4.3.9 The Tool Configuration

Overview

In the **Tool Configuration** window you make the configuration for the robot tool. The tool configuration is a representation of the robot tool and is included in the line configuration.

This section describes:

- How to start the **Tool Configuration**.
- The different functions in the **Tool Configuration**.
- How to import and export a tool.

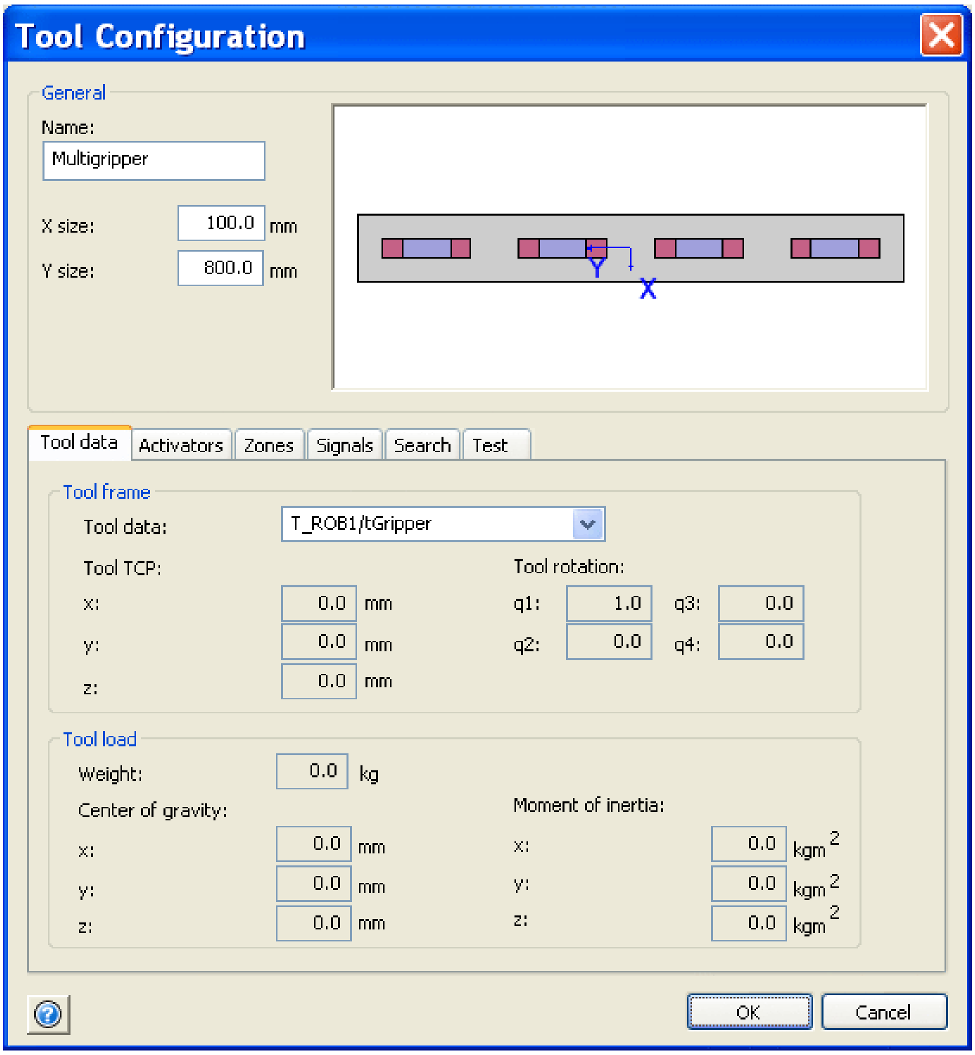
Start the Tool Configuration

	Action	Info
1	In the workspace, right-click the robot symbol. Select **Tool** and click **Edit**. or In the Library tree view, right-click the tool to edit and select **Edit**.	You can also import a tool from the PickMaster library, see *Import a tool from the library to the robot on page 90*.
2	The **Tool Configuration** window appears and you can edit the tool.	See *Illustration, Tool Configuration on page 82*.

Continues on next page

Illustration, Tool Configuration

The **Tool Configuration** window consists of two main parts, a general part and a tab view.



Name: Multigripper

X size: 100.0 mm

Y size: 800.0 mm

Tool data: T_ROB1/tGripper

Tool TCP:

x: 0.0 mm

y: 0.0 mm

z: 0.0 mm

Tool rotation:

q1: 1.0

q2: 0.0

q3: 0.0

q4: 0.0

Weight: 0.0 kg

Center of gravity:

x: 0.0 mm

y: 0.0 mm

z: 0.0 mm

Moment of inertia:

x: 0.0 kgm 2

y: 0.0 kgm 2

z: 0.0 kgm 2

xx0600002749

General	The general part of the Tool Configuration window contains the name, the size of the tool and an area that shows the tool. For detailed descriptions of the different parts, see *The general part on page 82*.
Tabbed pages	The tabbed pages consist of the six tabs **Tool data**, **Activators**, **Zones**, **Signals**, **Search** and **Test**. For detailed descriptions of the contents in the different tabbed pages, see *The tabbed pages on page 83*.

The general part

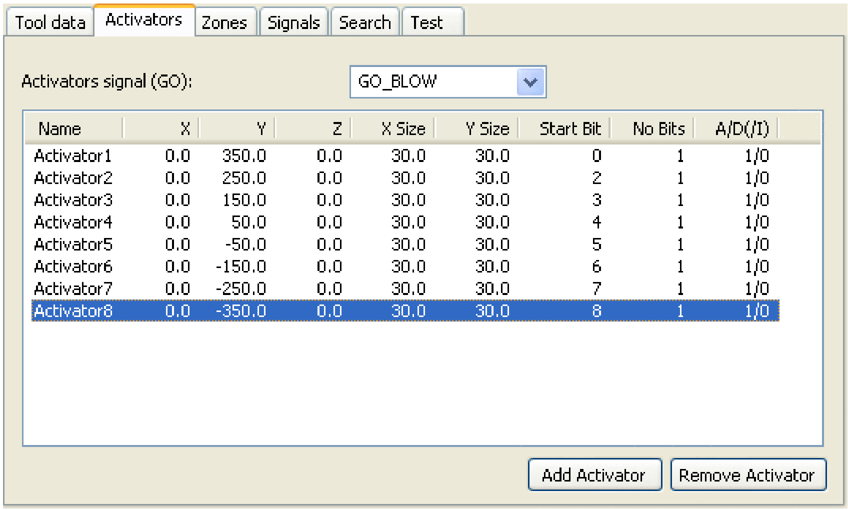
Parts	Description
Name	The name of the tool.
X Size	The outer dimension of the tool in the x-direction. Only used to create the graphical view of the tool.
Y Size	The outer dimension of the tool in the y-direction. Only used to create the graphical view of the tool.

Continues on next page

Continued

Parts	Description
Display for the tool	The display shows an overview of the tool, viewed from the robot wrist location with the z-axis pointing away from the observer. The origin represents the TCP (Tool Center Point) defined in the corresponding tool data on the controller.

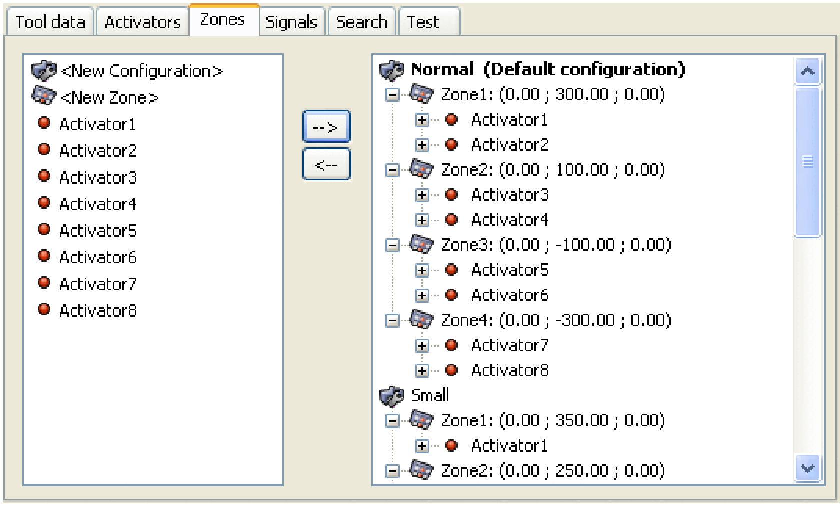
The tabbed pages

<table>
<tr><th>Tabbed page</th><th>Description</th></tr>
<tr><td><b>Tool data</b></td><td>Here you define the corresponding tool data on the controller.
In the <b>Tool data</b> drop-down combo box you select which tool data variable to use. You can select an existing tool from the controller if it is connected or type a tool data. The tool data is read from the controller.
For an illustration of the <b>Tool data</b> page, see <i>Illustration, Tool Configuration on page 82</i>.
<b>Note</b>
It is important that the selected or typed tool data exists on the controller when the project is running because the tool data will be used in every robot movement.</td></tr>
<tr><td><b>Activators</b></td><td>Here you set the values for the activators and select the signal that will control the gripping of the tool.
An activator is a physical correspondent to a DO/GO (digital output/group output) and controls one part of the tool, for example a vacuum cup or a mechanical gripper.
The figure shows a configuration example.


Activators signal (GO): GO_BLOW
<table>
<tr><th>Name</th><th>X</th><th>Y</th><th>Z</th><th>X Size</th><th>Y Size</th><th>Start Bit</th><th>No Bits</th><th>A/D(/I)</th></tr>
<tr><td>Activator1</td><td>0.0</td><td>350.0</td><td>0.0</td><td>30.0</td><td>30.0</td><td>0</td><td>1</td><td>1/0</td></tr>
<tr><td>Activator2</td><td>0.0</td><td>250.0</td><td>0.0</td><td>30.0</td><td>30.0</td><td>2</td><td>1</td><td>1/0</td></tr>
<tr><td>Activator3</td><td>0.0</td><td>150.0</td><td>0.0</td><td>30.0</td><td>30.0</td><td>3</td><td>1</td><td>1/0</td></tr>
<tr><td>Activator4</td><td>0.0</td><td>50.0</td><td>0.0</td><td>30.0</td><td>30.0</td><td>4</td><td>1</td><td>1/0</td></tr>
<tr><td>Activator5</td><td>0.0</td><td>-50.0</td><td>0.0</td><td>30.0</td><td>30.0</td><td>5</td><td>1</td><td>1/0</td></tr>
<tr><td>Activator6</td><td>0.0</td><td>-150.0</td><td>0.0</td><td>30.0</td><td>30.0</td><td>6</td><td>1</td><td>1/0</td></tr>
<tr><td>Activator7</td><td>0.0</td><td>-250.0</td><td>0.0</td><td>30.0</td><td>30.0</td><td>7</td><td>1</td><td>1/0</td></tr>
<tr><td>Activator8</td><td>0.0</td><td>-350.0</td><td>0.0</td><td>30.0</td><td>30.0</td><td>8</td><td>1</td><td>1/0</td></tr>
</table>
xx0700000291
In the <b>Activators signal (GO)</b> drop-down combo box you select which group output signal that will control the gripping of the tool.
From release RobotWare 5.11.01, there is a default activator signal in PickMaster RC software. The signal is called <i>pmGripper1_goActivators</i>. You can select this signal as Activator signal even if the controller is not online.
To add a new activator, click the <b>Add Activator</b> button.
To remove an activator, click the <b>Remove Activator</b> button.
To edit an activator setting, click the value to edit and edit the value.
For descriptions of the activator settings, see <i>Activators tab on page 87</i>.</td></tr>
</table>

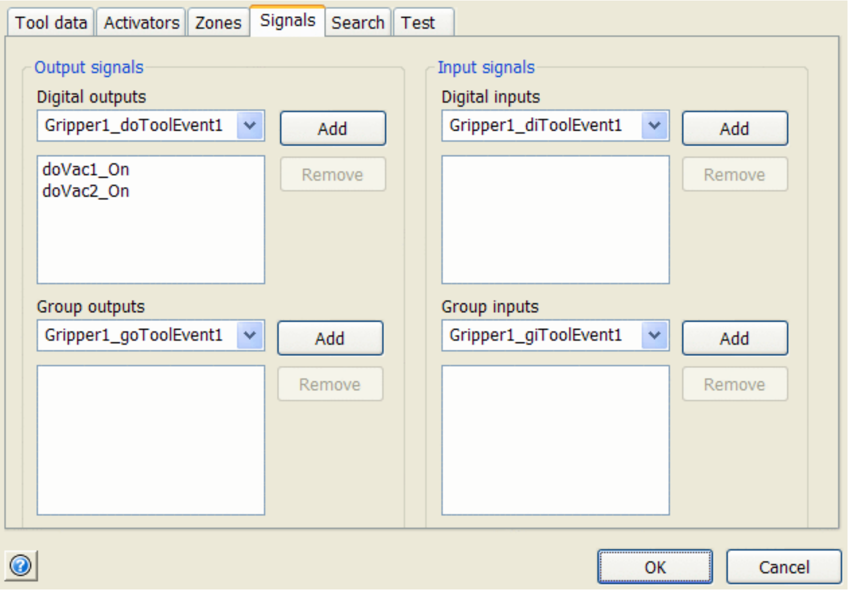
Continues on next page

Continued

Tabbed page	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 robot tool can have. A tool can have several configurations but only one at a time can be active. The figure shows a configuration example.  Tool data Activators Zones Signals Search Test --> <-- <New Configuration> <New Zone> Activator1 Activator2 Activator3 Activator4 Activator5 Activator6 Activator7 Activator8 **Normal (Default configuration)** Zone1: (0.00 ; 300.00 ; 0.00) Activator1 Activator2 Zone2: (0.00 ; 100.00 ; 0.00) Activator3 Activator4 Zone3: (0.00 ; -100.00 ; 0.00) Activator5 Activator6 Zone4: (0.00 ; -300.00 ; 0.00) Activator7 Activator8 Small Zone1: (0.00 ; 350.00 ; 0.00) Activator1 Zone2: (0.00 ; 250.00 ; 0.00) xx0600002754 The activators visible in the list to the left in the figure are the activators that have been defined in the Activators view. The TCP (Tool Center Point) shown for each zone will automatically be generated as the center position of all included activators. It is recommended to set the same z-value to all activators within one zone. For configuration procedures, see *Zones tab on page 87*. **Note** Activators that are not added to any zone will be in deactivated state. For details of the states, A/D/I, see *Activators tab on page 87*. **Note** The TCP that will actually be used during a pick and place operation is calculated for each movement. The calculated TCP depends both on what is held in the gripper and the zones to use for each specific robot target.

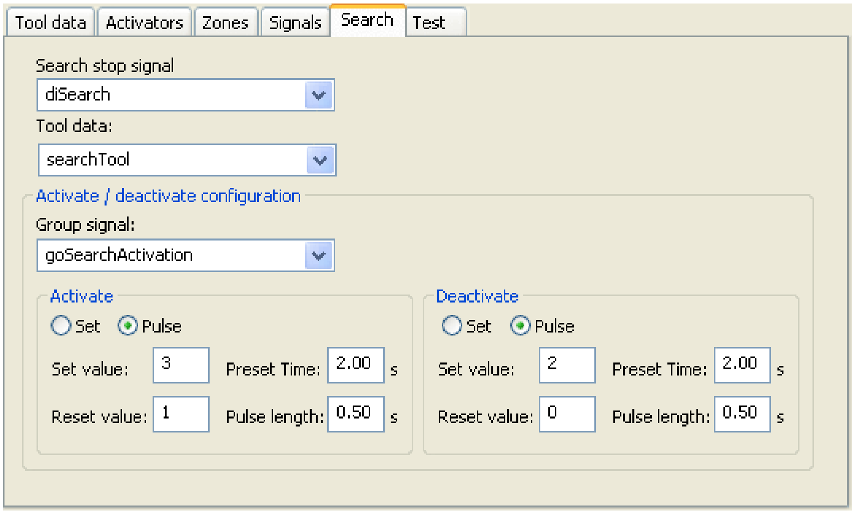
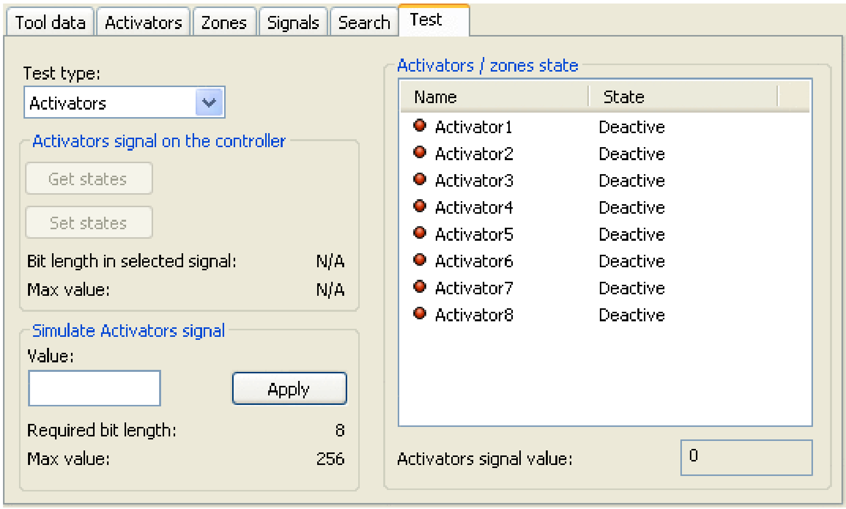
Continues on next page

Continued

Tabbed page	Description
Signals	Sometimes it is *not* enough to use the tool zones to pick or place items. Instead there might be a separate gripper on the tool controlled by specific output and input signals. On this page you can add tool specific I/O signals of digital input, digital output, or group output types. These signals can later be used in the project to define how to pick or place a specific format. The figure below shows a configuration example. The signals added in the **Signals** tab have previously been defined on the controller. In this example, the **Group outputs** signal is defined for controlling (opening/closing) an I/O driven gripper device, and the **Digital inputs** signals are used for checking status of the gripper device (whether it is open or not). For more information about how to define signals on the controller, see *Technical reference manual - System parameters*. From release RobotWare 5.11.01, there are default tool event signals in PickMaster RC software. These signals are available for selection even if the controller is not online.  Tool data Activators Zones Signals Search Test Output signals Digital outputs Add Remove Group outputs Add Remove Input signals Digital inputs Add Remove Group inputs Add Remove OK Cancel Digital outputs: Gripper1_doToolEvent1 doVac1_On doVac2_On Group outputs: Gripper1_goToolEvent1 Digital inputs: Gripper1_diToolEvent1 Group inputs: Gripper1_giToolEvent1 xx0700000294 For more information about the signal page, see *Signals tab on page 88*. For more information about how the signals are used, see *The Format Configuration on page 110*.

Continues on next page

Continued

Tabbed page	Description
Search	Here you define the configuration for the search tool. The figure shows a configuration example.  Tool data Activators Zones Signals Search Test Search stop signal: diSearch Tool data: searchTool Activate / deactivate configuration Group signal: goSearchActivation Activate: ○ Set ● Pulse Set value: 3 Preset Time: 2.00 s Reset value: 1 Pulse length: 0.50 s Deactivate: ○ Set ● Pulse Set value: 2 Preset Time: 2.00 s Reset value: 0 Pulse length: 0.50 s xx0700000292 In the **Search stop signal** drop-down combo box you choose which signal to be activated when the search tool hits an object in its path. In the **Tool data** drop-down combo box you choose which tool to use for the search operation. In the **Activate/Deactivate configuration**, you set the configuration for extending and withdrawing the search tool. If the **Set** option button is selected, the set value is sent at preset time before the robot reaches the target T0. If the **Pulse** option button is selected, the reset value is sent after pulse duration. For more information, see *Search tab on page 89*. From release RobotWare 5.11.01, there is a default search stop signal and a default group signal for activate/deactivate configuration in PickMaster RC software. These signals are available for selection even if the controller is not online.
Test	Here you can test the functionality of your tool. The figure shows a configuration example.  Tool data Activators Zones Signals Search Test Get states Set states Apply Test type: Activators Activators signal on the controller Bit length in selected signal: N/A Max value: N/A Simulate Activators signal Value: Required bit length: 8 Max value: 256 Activators / zones state Name State ● Activator1 Deactive ● Activator2 Deactive ● Activator3 Deactive ● Activator4 Deactive ● Activator5 Deactive ● Activator6 Deactive ● Activator7 Deactive ● Activator8 Deactive Activators signal value: 0 xx0700000293 For descriptions of the different parts in the Test tab, see *Test tab on page 89*.

Continues on next page

Activators tab

The following table describes the settings of the activators:

Setting	Description
Name	The activator's name, that is the reference the user has to the activator.
X	The center position of the activator related to the TCP.
Y	The center position of the activator related to the TCP.
Z	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.
x-size	The size of the activator in x-direction.
y-size	The size of the activator in y-direction.
Start bit	Defines the first bit field where the activator is connected to the Activators signal (GO).
No Bits	The number of bit fields used for the activator.
A/D/I	Defines the Active, Deactivated, and Idle 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: • Active state, which is set for an activator when it holds an item. • Deactivated state, which is set for an activator when it releases an item. • Idle 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. **Note** If *No bits* is set to the value 1, the Idle state is not used since only two states can be given with one bit.

Note

If you have defined activators that are connected to the same I/O, a warning appears. Adjust the values for the settings *Start bit* and *No Bits*.

Note

The bits of the Activators signal (GO) that are not used by an activator must be set to the value 0 - that is, every bit of the Activators signal (GO) that should be controlled by PickMaster must be defined as an activator.

Zones tab

The following table describes how to proceed with the zone configuration:

To...	Do this
Add a new configuration	Drag the **<New Configuration>** symbol from the list on the left side to the list on the right side. Or Select the **<New Configuration>** symbol in the list on the left side and click the **-->** button.

Continues on next page

To...	Do this
Select a default configuration	Click the **Configuration** symbol in the list on the right side.
Rename a configuration	Click the **Configuration** symbol in the list on the right side and press F2.
Add a new Zone to a configuration	Drag the **<New Zone>** symbol from the list on the left side to the desired configuration in the list on the right side. Or In the list on the right side, select the configuration where the new zone should be added. Then select the **<New Zone>** symbol in the list on the left side and click the **-->** button.
Rename a Zone	Click the **Zone** symbol and press F2.
Add an activator to a zone	Drag the desired activator from the list on the left side to the desired zone in the list on the right side. Or In the list on the right side, select the zone where the new activator should be added. Then select an activator from the list on the left side and click the **-->** button. (The area that shows the tool in the Tool Configuration window will show the new setup.)
Remove an item from a configuration	Drag the item from the list on the right side to the list on the left side. Or In the list on the right side, select the item to be removed and click the **-->** button.

Note

When you click **OK**, an information message can appear saying there is no proper configuration. To select a default configuration, click on the desired **Configuration** symbol in the Zones view.

Note

If you remove a zone, the included activators will also be removed from the configuration. If you remove the configuration, the included zones and activators will also be removed.

Signals tab

The following table describes how to proceed with the tool signals.

To...	Do this
Add a signal	Depending on the type of signal you want to add, 1 Select the **Digital outputs**, **Group outputs** or **Digital inputs** combo box 2 Select the signal to add or enter a signal name 3 Click the corresponding **Add** button
Remove a signal	Select the signal to remove and click the corresponding **Remove** button.

Continues on next page

Search tab

The following table describes how to proceed with the search tool configuration.

	Action
1.	In the **Search stop signal** combo box, select the search stop input signal. The signal is used to indicate that the top layer has been detected during the search movement.
2.	In the **Tool data** combo box, select the search tool.
3.	If needed, set up the activate/deactivate configuration as described in the following steps.
4.	In the **Group signal** combo box, select the group I/O signal to activate/deactivate. The signal is used to activate the search tool with an I/O event before the search movement starts. The signal is also used to deactivate the search tool with an I/O event just before approaching the next target position after a search movement is completed. If the signal is not defined, activation and deactivation I/O events will not occur when performing stack search.
5.	Select the type of search activation I/O event, that is, Set or Pulse: 1 In the **Set value** and **Preset Time** text boxes, type the desired values. 2 If Pulse is selected, type also the desired values in the **Reset value** and **Pulse length** text boxes. (Pulse length is the duration time of the pulse.)
6.	Select the type of search deactivation I/O event, that is Set or Pulse: 1 In the **Set value** and **Preset time** text boxes, type the desired values. 2 If Pulse is selected, type also the desired values in the **Reset value** and **Pulse length** text boxes.

Test tab

The following table describes the different parts of the Test tab.

Part of the Test tab	Description
The **Test type** drop-down combo box	Here you choose if you are going to test activators or zones.
Activators/zones state	In the list box all activators/zones in the activated configuration (selected in the **Zone** tab) are listed together with their state. To edit the state of an activator/zone: 1 In the list box, click the state of an activator/zone. 2 In the drop-down combo box, select a state. The **Activators signal value** shows the current state in the list box translated to the numerical value of the activators signal. The values will depend on the values that were set to the activators in the Activators tab, see *Activators tab on page 87*.
Activators signal on the controller	This part includes the following buttons: • The **Get states** button, which you click to read the activators signal from the connected controller. The **Activators/zones state** will be updated to that value. • The **Set states** button, which you click to set the activators signal on the controller to the value of the **Activators/zones state**.

Continues on next page

Continued

Part of the Test tab	Description
Simulate Activators signal	Here you can simulate a retrieval of the activators signal value with a value that you select. This can be useful when no robot is connected. To proceed: 1 In the **Value** text box, type a value for the activators signal. 2 Click **Apply**. The states in the **Activators/zones** state list will be updated with the simulated value. **Note** When using two bits activators there will be undefined values. The reason is that there are three different states for an activator, but a two bits activator gives four different values. An activator with undefined state will be set to an appropriate state. Accordingly, the **Activators signal value** and the **Value** text box will have different values.

Import and export a tool

There is a tool object for each robot in the line. Tools can also be stored in the PickMaster Library. You can import a tool from and export a tool to the library.

CAUTION

When you import a tool from the library, you overwrite the existing robot tool. To import a tool and then run the robot, you must be aware that you will run the robot with the imported tool.

Import a tool from the library to the robot

When you want to use a tool that is included in the PickMaster library, you import a tool to the robot.

To import a tool:

	Action
1.	In the workspace, right-click the robot. Select **Tool** and click **Import**.
2.	In the **Import Tool from Library** dialog box, select a tool from the drop-down combo box. You can select from the tools that are included in the PickMaster library.
3.	Click **OK**.
4.	A dialog box appears, asking if you want to overwrite the current tool. Click **Yes**.

Export a tool from the robot to the library

When you want to save a tool configuration, you export the tool to the PickMaster library. A copy of the exported tool will appear in the library, without overwriting any tool.

To export a tool:

	Action
1.	In the workspace, right-click the robot. Select **Tool** and click **Export**.

Continues on next page

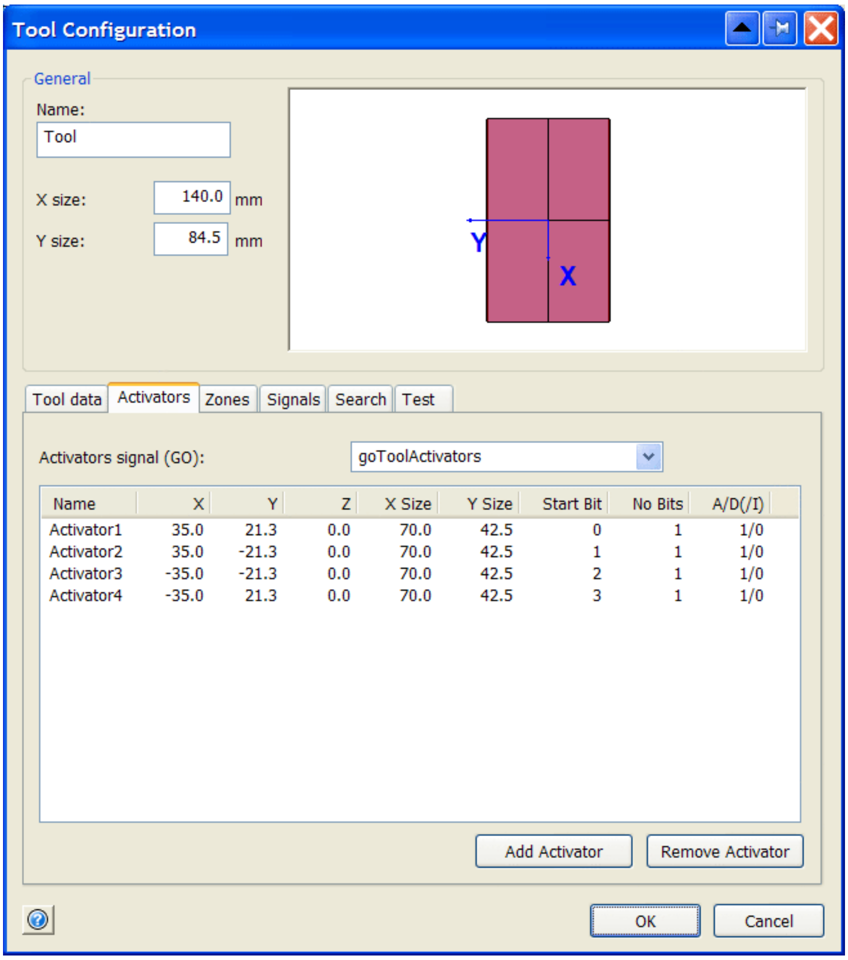
Continued

	Action
2.	A copy of the exported tool appears in the Tools group of the PickMaster library. To rename the tool, right-click the tool and click **Rename**.

2D tool

PickMaster 5 supports using tools with two-dimensional activator zones for picking items grouped in multiple rows.

The illustrations below show an example of tool configuration for picking 2x2 items.



Tool Configuration

General

Name: Tool

X size: 140.0 mm

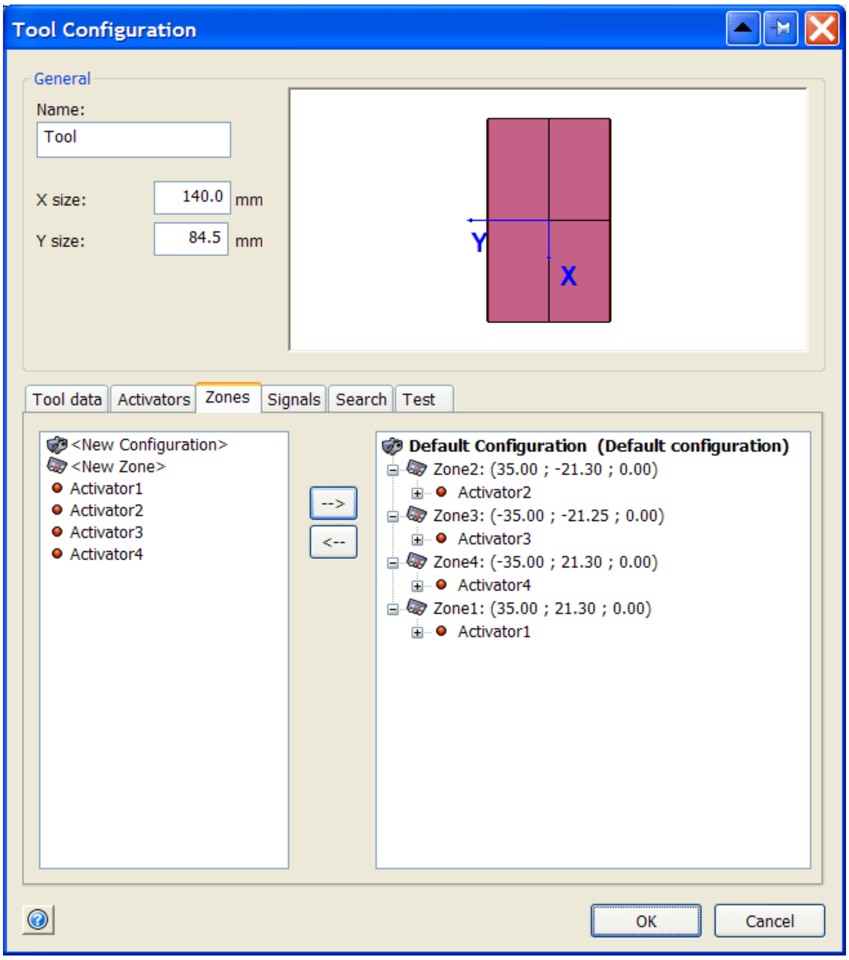
Y size: 84.5 mm

Activators signal (GO): goToolActivators

Name	X	Y	Z	X Size	Y Size	Start Bit	No Bits	A/D(/I)
Activator1	35.0	21.3	0.0	70.0	42.5	0	1	1/0
Activator2	35.0	-21.3	0.0	70.0	42.5	1	1	1/0
Activator3	-35.0	-21.3	0.0	70.0	42.5	2	1	1/0
Activator4	-35.0	21.3	0.0	70.0	42.5	3	1	1/0

en0800000285

Continues on next page



Name: Tool

X size: 140.0 mm

Y size: 84.5 mm

- <New Configuration>
- <New Zone>
- Activator1
- Activator2
- Activator3
- Activator4

Default Configuration (Default configuration)
- Zone2: (35.00 ; -21.30 ; 0.00)
  - Activator2
- Zone3: (-35.00 ; -21.25 ; 0.00)
  - Activator3
- Zone4: (-35.00 ; 21.30 ; 0.00)
  - Activator4
- Zone1: (35.00 ; 21.30 ; 0.00)
  - Activator1

en0800000286

Related information

The Line view on page 57.

The PickMaster Library on page 51.

The Line view on page 57.

Terminology on page 24.

4.3.10 The Positions and Robot Configuration

Overview

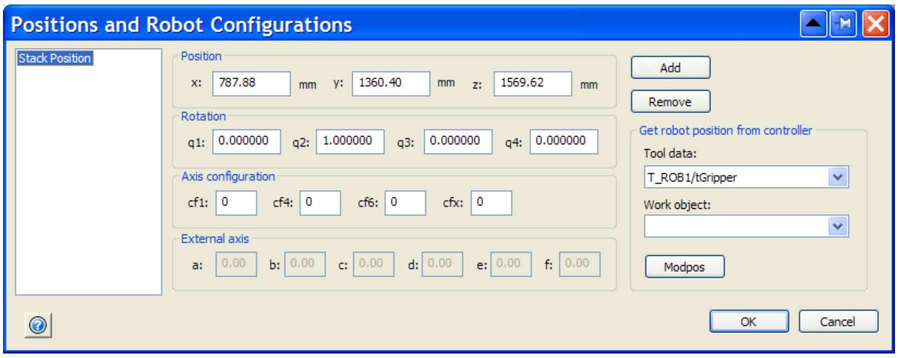
This section describes the **Positions and Robot Configuration** window where you create positions that can be used as work area safe positions, and make 6 axes robots to use the correct configuration. For details, see *Work area safe positions on page 94*.

The positions are created in the PickMaster line, whereas the usage of the positions is initiated in the PickMaster project.

Start the Positions and Robot Configuration

	Action
1	In the workspace of the Line view, right-click the work area symbol and select **Positions**.
2	The **Positions and Robot Configuration** dialog box appears. Now you can create and edit positions.

Illustration, The Positions and Robot Configuration



Stack Position

x: 787.88 mm y: 1360.40 mm z: 1569.62 mm

q1: 0.000000 q2: 1.000000 q3: 0.000000 q4: 0.000000

cf1: 0 cf4: 0 cf6: 0 cfx: 0

a: 0.00 b: 0.00 c: 0.00 d: 0.00 e: 0.00 f: 0.00

Tool data: T_ROB1/tGripper

Work object:

xx0700000249

List	This is a list that contains the created positions.
Position **Rotation** **Axis configuration** **External axis**	Here you type all the data that is necessary to describe the robot position.
Add and **Remove** buttons	You click the **Add** button to create a new work area safe position. The position will appear in the list on the left side. You click the **Remove** button to remove the work area safe position that is selected in the list on the left side.
Get robot position from controller	The data about **Position**, **Rotation**, **Axis configuration**, and **External axis** can be read from the robot if it is available on the network. (Then you should jog the robot to the desired position with the desired configuration.) To get the correct position, both tool data and work object information are needed. The **Tool data** drop-down combo box shows the configured tool for the robot, and the **Work object** drop-down combo box shows the configured work object for the current work area. If no tool is configured or the configured work object is not valid on the controller, the **Modpos** button will *not* be accessible. When you click **Modpos**, all the data will be read from the robot for the position selected in the list. The **Modpos** button will be accessible only when the robot is available and if a position is selected from the list.

Continues on next page

How to proceed

This section describes how to proceed with the work area safe positions.

Using a work area safe position

	Action	Note
1.	Create the positions in the work area in the PickMaster line.	See *Illustration, The **Positions and Robot Configuration** on page 93*.
2.	Select one of the positions in the position source in the PickMaster project.	A position is selected in the **Robot position** area in the **Position Source Configuration** window. See *Illustration, Position Source Configuration on page 125*.
3.	Select if the robot should use `MoveJ` or `MoveL` for the position.	`MoveJ` is the default selection.

Note

When a position is selected for a position source, the robot will always pass this point when going to or leaving the work area connected to the position source. This will be done for all operation sets in the position source.

Work area safe positions

A work area safe position is a position that the robot always has to pass through when going to or leaving a work area. The main purpose of such position is that there is some kind of obstacle that forces the robot to approach from a special point to avoid a collision, for example, when pallets are placed in a rack. Another reason for using work area safe positions is six axes robots. Since the robot configuration must be set for the position, there is the possibility to ensure that the robot has the correct configuration when approaching the work area.

Related information

The Line view on page 57.

The Project view on page 95.

The Work Area Configuration on page 73.

The Position Source Configuration on page 124.

4.4 Project

4.4.1 The Project view

Overview

To be able to run a project with PickMaster, a line must be defined. A project describes the packaging process and it is in the project that you configure the products and the product flow.

A project is saved as a *.pmproj* file.

Prerequisites

To be able to work with a project, the line that the project is built on must be registered on the computer. See *Registering the Line file on page 57*.

When the line is registered, close it and open the project, and then the line will automatically be opened.

Open the Project view

To...	On the File menu, click...
create a new project	**New Project**
edit an existing project	**Open Project**

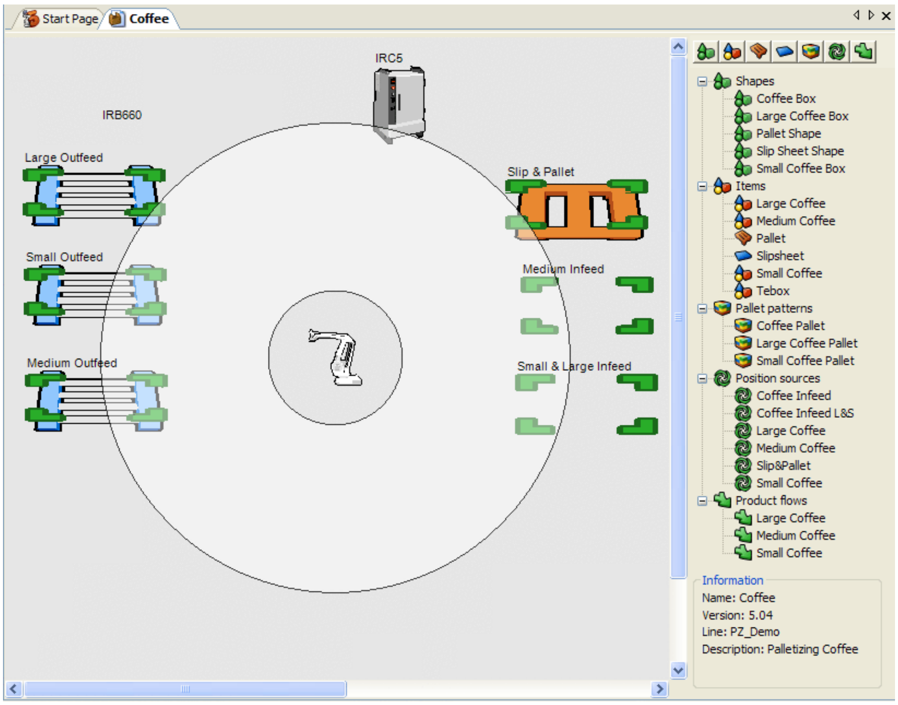
When a project is open

To...	On the File menu, click...
open the corresponding line for editing	**Open Current Line**
return to the project	**Open Previous Projects**

Continues on next page

Illustration, Project

The PickMaster workspace lets you interact with the project.



Information

Name: Coffee

Version: 5.04

Line: PZ_Demo

Description: Palletizing Coffee

xx0600002985

Project panel	The project panel is located on the right side of the Project window. Here you create shapes, items (products, pallets, and slip sheets), pallet patterns, position sources, and product flows.

How to proceed

This section describes how to use the different commands in the Project view.

Add objects in the project panel

Object to create	Action
Shape	In the **Project panel**, right-click the **Shapes** bar and click **New Shape**. Or On the **Project panel** bar, click the **Shapes** button.
Product	In the **Project panel**, right-click the **Items** bar and click **New Product**. Or On the **Project panel** bar, click the **Product** button.
Pallet	In the **Project panel**, right-click the **Items** bar and click **New Pallet**. Or On the **Project panel** bar, click the **Pallet** button.
Slip sheet	In the **Project panel**, right-click the **Items** bar and click **New Slip Sheet**. Or On the **Project panel** bar, click the **Slip sheet** button.

Continues on next page

Continued

Object to create	Action
Format	How to create formats is described in section *The Format Configuration on page 110*.
Pallet pattern	In the **Project panel**, right-click the **Pallet patterns** bar and click **New Pallet Pattern**, **Import**, or click **Import From Pallet Toolkit**. See *Import and export pallet patterns on page 98* below. Or On the **Project panel** bar, click the **Pallet pattern** button.
Position source	In the **Project panel**, right-click the **Positions sources** bar and click **New Position Source**. See *The Position Source Configuration on page 124*. Or On the **Project panel** bar, click the **Position Source** button.
Flow	In the **Project panel**, right-click the **Flow** bar and click **New Flow**. See *The Flow Configuration on page 145*. Or On the **Project panel** bar, click the **Flow** button.

Copy and paste PickMaster project objects

It is possible to copy and paste the following objects in a PickMaster project:

- Shape
- Item (Product/Pallet/SlipSheet)
- Pallet Pattern
- Position Source

Objects can only be copied and pasted into their parent node, for example a Shape cannot be pasted under node Item or Pallet patterns.

When copying and pasting an Item (Product/Pallet/SlipSheet), all formats belonging to the item will also be pasted into the new Item. Formats can also be copied and pasted. For more information, see *Proceed with Organize Formats on page 114*.

The following rules applies to copying and pasting a Position Source:

- A Position Source can only be pasted on a work area that has no associated Position Source. Otherwise an error message is displayed and no new Position Source is created.
- All operation sets belonging to the Position Source will be copied to the new Position Source.
- In the Position Source Configuration dialog, Operation Set can be copied and pasted into the same Position Source.

To...	Do this
Copy an object	In the **Project panel**, right-click an object and select **Copy**. Or In the **Project panel**, select an object and press CTRL + C.
Paste an object	In the **Project panel**, right-click an object container and select **Paste**. Or In the **Project panel**, select an object container and press CTRL + V.

Continues on next page

Continued

Import and export pallet patterns

Pallet patterns generated in PickMaster can be exported as separate files with the *.pmpall* extension. When exporting a pallet pattern, all the shapes used in the pattern will also be included in the file. When a pallet pattern is imported, any shapes used in a pallet pattern that does not exist in the project will be added as well.

To...	Do this
Export a pallet pattern	In the **Project panel**, right-click the pallet pattern to export and click **Export**. Select where to save the file.
Import a pallet pattern	In the **Project panel**, right-click the **Pallet patterns** bar and click **Import**. Select the pallet pattern file to import.

Import pallet patterns from Pallet Toolkit

Pallet patterns can also be imported from the Pallet Toolkit RAPID modules. When importing a Pallet Toolkit project, PickMaster will import all shapes, products, pallets, slip sheets and the pallet pattern from the RAPID module. If there already is a shape in the PickMaster project with the same shape found in the Pallet Toolkit RAPID module, no new shape will be created in the PickMaster project. Instead, the shape already present in the PickMaster project will be used when creating the items.

To...	Do this
Import a pallet pattern from Pallet Toolkit	In the **Project panel**, right-click the **Pallet patterns** bar and click **Import From Pallet Toolkit**. Select the RAPID module file to import.

Note

PickMaster can only import the pallet pattern from a Pallet toolkit project. Pallet operations cannot be imported.

Further, PickMaster can only import pallet patterns from single-drop pallet toolkit projects. Multi-drop projects cannot be imported.

Add a description to the project

	Action
1.	In the Project view, right-click in the workspace and click **Properties**. Or On the **Project** menu, click **Properties**.
2.	The **Project Information dialog** box appears. In the **Description** text box, type a description of the project.

Transfer the project to the controller

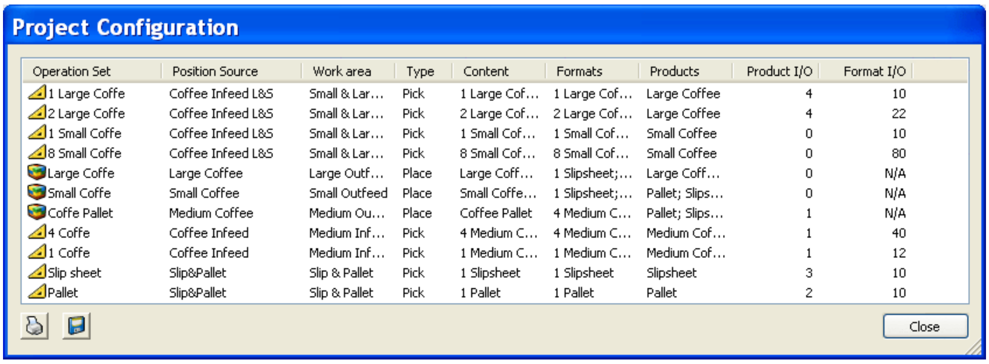
To transfer a project to the controller, proceed in one of the following ways:

- In the Project view, right-click in the workspace and click **Transfer**, or
- On the **Project** menu, click **Transfer**.

Continues on next page

Handle the Project Configuration dialog box

The Project Configuration dialog box shows each configured operation set and its corresponding product and format I/O values. It is also possible to print the list and save to a comma delimited or tab delimited text file.



Project Configuration

Operation Set	Position Source	Work area	Type	Content	Formats	Products	Product I/O	Format I/O
1 Large Coffe	Coffee Infeed L&S	Small & Lar...	Pick	1 Large Cof...	1 Large Cof...	Large Coffee	4	10
2 Large Coffe	Coffee Infeed L&S	Small & Lar...	Pick	2 Large Cof...	2 Large Cof...	Large Coffee	4	22
1 Small Coffe	Coffee Infeed L&S	Small & Lar...	Pick	1 Small Cof...	1 Small Cof...	Small Coffee	0	10
8 Small Coffe	Coffee Infeed L&S	Small & Lar...	Pick	8 Small Cof...	8 Small Cof...	Small Coffee	0	80
Large Coffe	Large Coffee	Large Outf...	Place	Large Coff...	1 Slipsheet;...	Large Coff...	0	N/A
Small Coffe	Small Coffee	Small Outfeed	Place	Small Coffe...	1 Slipsheet;...	Pallet; Slips...	0	N/A
Coffe Pallet	Medium Coffee	Medium Ou...	Place	Coffee Pallet	4 Medium C...	Pallet; Slips...	1	N/A
4 Coffe	Coffee Infeed	Medium Inf...	Pick	4 Medium C...	4 Medium C...	Medium Cof...	1	40
1 Coffe	Coffee Infeed	Medium Inf...	Pick	1 Medium C...	1 Medium C...	Medium Cof...	1	12
Slip sheet	Slip&Pallet	Slip & Pallet	Pick	1 Slipsheet	1 Slipsheet	Slipsheet	3	10
Pallet	Slip&Pallet	Slip & Pallet	Pick	1 Pallet	1 Pallet	Pallet	2	10

xx0700000016

To...	Do this...
Open the Project Configuration dialog box	On the **Project** menu, click **Configuration**. Or Right-click in the workspace and click **Configuration**.
Print the list	Click the print button in the lower left corner.
Save the list	Click the save button in the lower left corner. Select location, file name and file type (comma delimited or tab delimited text file), and click **OK**.
Sort columns	Click the column header of the column you intend to sort. To sort in the other direction, click again.

Related information

The Line and Project concept on page 20.

Setting up the Project on page 167.

*The **Tool Configuration** on page 81*.

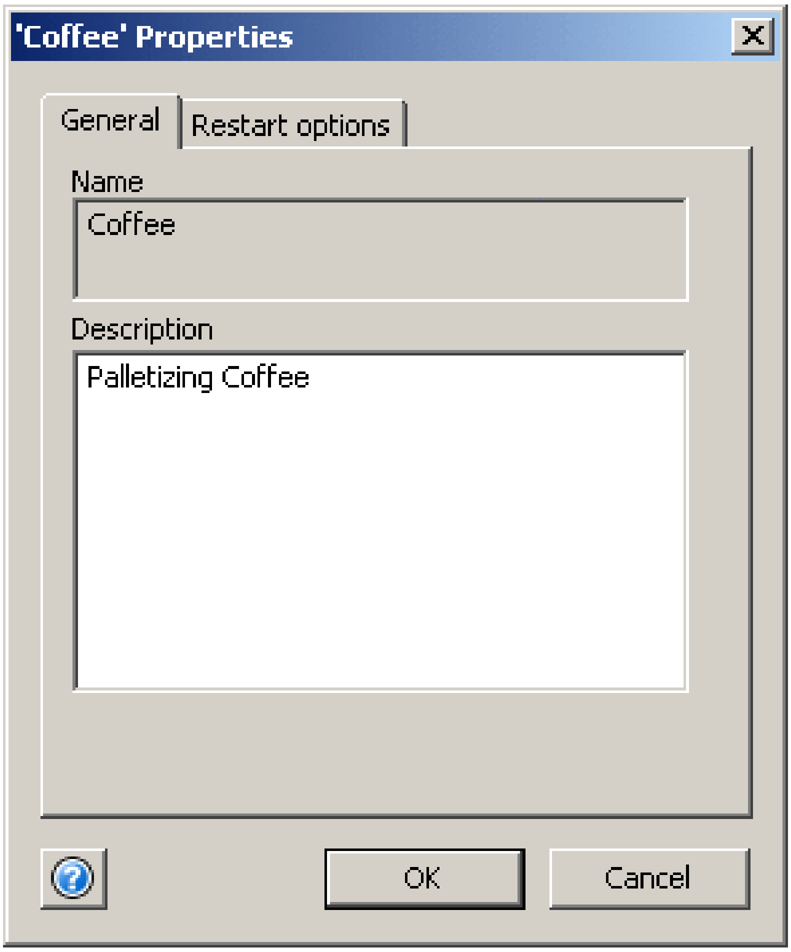
4.4.2 The Project Properties

Overview

In the Project Properties window you add a project description and define which restart options should be available on the FlexPendant.

Illustration, Project Properties

Enter a project description in the **Description** box.



'Coffee' Properties

Name: Coffee

Description: Palletizing Coffee

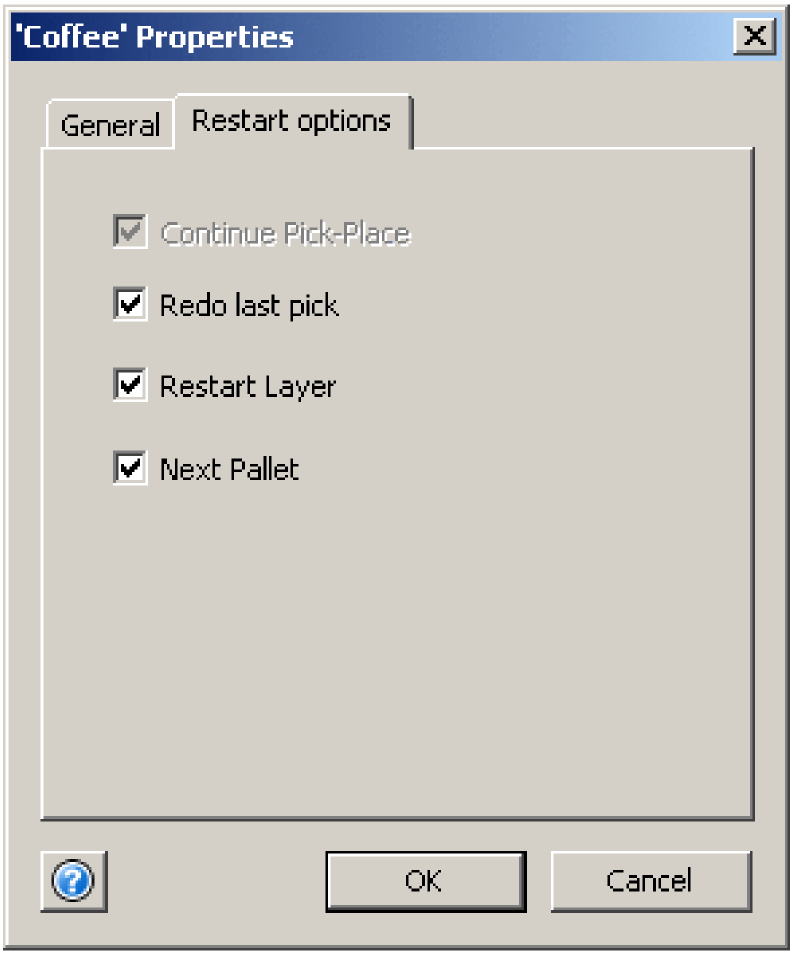
en0800000393

Continues on next page

Illustration, Restart options

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 recovery on page 223*.

Continue Pick-Place is always available.



☑ Continue Pick-Place

☑ Redo last pick

☑ Restart Layer

☑ Next Pallet

en0800000387

4.4.3 The Shape Configuration

Overview

In the **Shape Configuration** window you create or modify existing shapes. A shape defines the physical size and form of a project item and can be saved either in the PickMaster Library or in the project.

Start the Shape Configuration

Depending on which shape to edit, you can start the Shape Configuration in two ways:

- From the PickMaster Library, if you want to edit a shape that is included in the library, or
- From the project window, if you want to edit a shape that is included in a project.

Start from the PickMaster Library

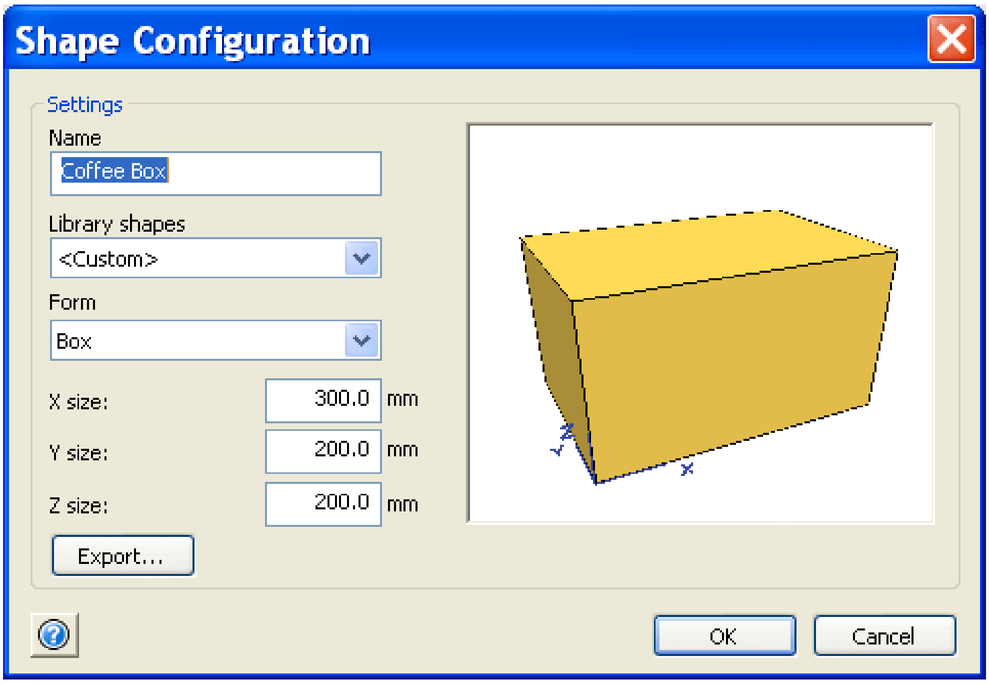
	Action
1.	In the PickMaster Library tree view, select the shape to edit and right-click.
2.	Click **Edit**.
3.	The **Shape Configuration** window appears and you can edit the shape.

Start from the project

	Action
1.	In the PickMaster project panel, select the shape to edit and right-click.
2.	Click **Edit**.
3.	The **Shape Configuration** window appears and you can edit the shape.

Continues on next page

Illustration, Shape Configuration


Shape Configuration

Settings

Name: Coffee Box

Library shapes: <Custom>

Form: Box

X size: 300.0 mm

Y size: 200.0 mm

Z size: 200.0 mm

xx0600002882

Settings	Here you name the shape and set its physical size and form.

How to proceed

This section describes how to proceed in the Shape Configuration window.

Define the size of the shape

The size of the shape can either be set manually or copied from an existing shape in the PickMaster Library. The form of the shape defines the look of the shape and which item types that can use the shape. Products can only use shapes with box forms, pallets use pallet forms and slip sheets use shapes with sheet forms.

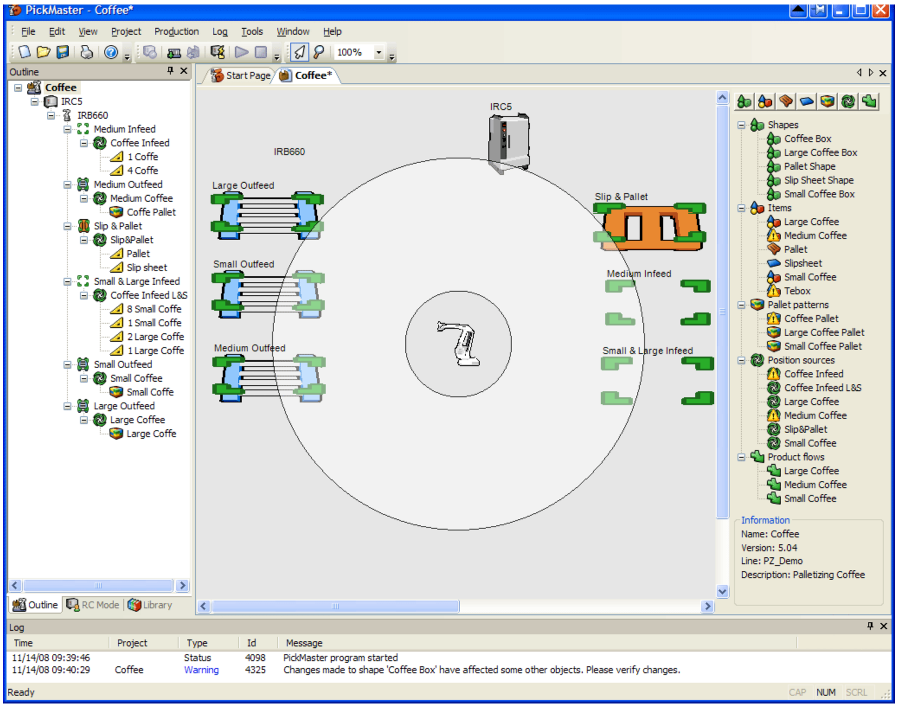
To...	Do this
Set the size manually	In the **X size**, **Y size** and **Z size** boxes, type the dimensions.
Copy size from library shape	In the **Library shapes** drop-down combo box, select the shape to copy.
Change form of the shape	In the **Form** drop-down combo box, select the form type to use.

Note

When the size of a shape has been modified, all products, formats, pallet patterns that use the shape and all Position Sources that use the affected products, formats, or pallet patterns will be shown with warning icons. A warning will also be displayed in PickMaster log window, see the following illustration.

Continues on next page

To remove all warning icons, open the configuration for each object and apply changes.



Name: Coffee
Version: 5.04
Line: PZ_Demo
Description: Palletizing Coffee

```
Time               Project   Type      Id     Message
11/14/08 09:39:46            Status    4098   PickMaster program started
11/14/08 09:40:29  Coffee    Warning   4325   Changes made to shape 'Coffee Box' have affected some other objects. Please verify changes.
```

xx0800000361

Export a shape to the PickMaster Library

If you are working with a shape in the project, it is possible to save that shape in the PickMaster Library for future access from other projects.

To...	Do this
Export the shape	Click **Export**.

Related information

The Item Configuration on page 105.

The PickMaster Library on page 51.

4.4.4 The Item Configuration

Overview

A PickMaster item is the general term for an object that can be picked and placed by the robot. An item can either be a product, a slip sheet, or a pallet pattern. The size of an item cannot be set. Instead, it refers to a specific shape in the same project. Several items can refer to the same shape and if the size of the shape is changed, the size of the item will also change. Specific item-related settings are configured in the **Item Configuration** window.

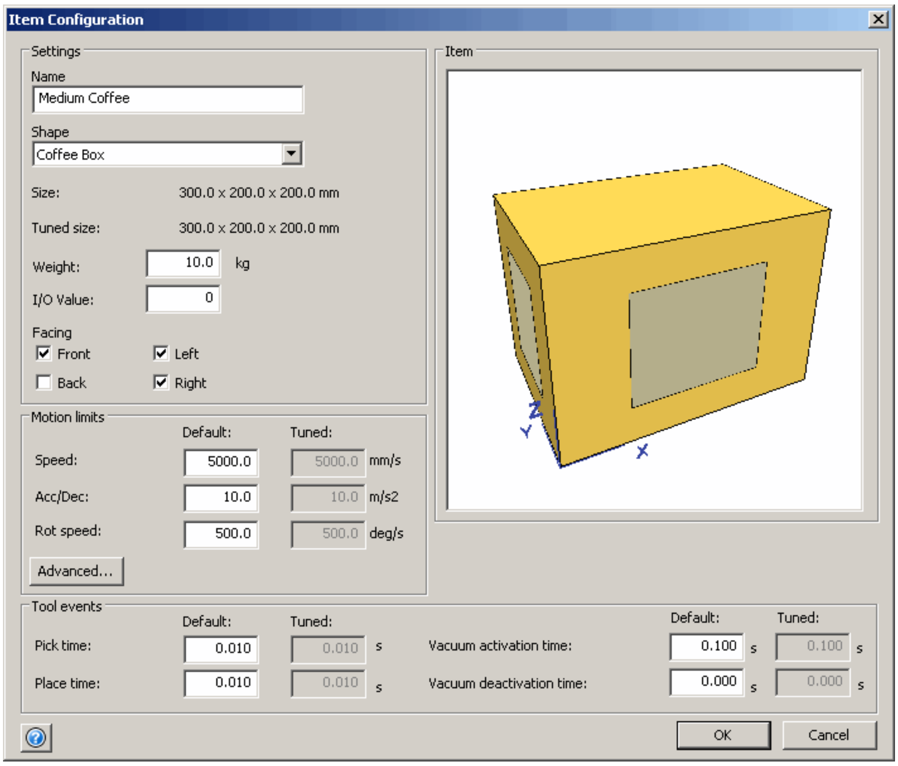
Start the Item Configuration

In the project panel

- Click the item you want to edit, or
- Right-click the item you want to edit and click **Edit**.

Continues on next page

Illustration, Item Configuration



Item Configuration

Settings

Name: Medium Coffee

Shape: Coffee Box

Size: 300.0 x 200.0 x 200.0 mm

Tuned size: 300.0 x 200.0 x 200.0 mm

Weight: 10.0 kg

I/O Value: 0

Facing

☑ Front ☑ Left

☐ Back ☑ Right

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

Tool events

	Default:	Tuned:			Default:	Tuned:	
Pick time:	0.010	0.010	s	Vacuum activation time:	0.100	0.100	s
Place time:	0.010	0.010	s	Vacuum deactivation time:	0.000	0.000	s

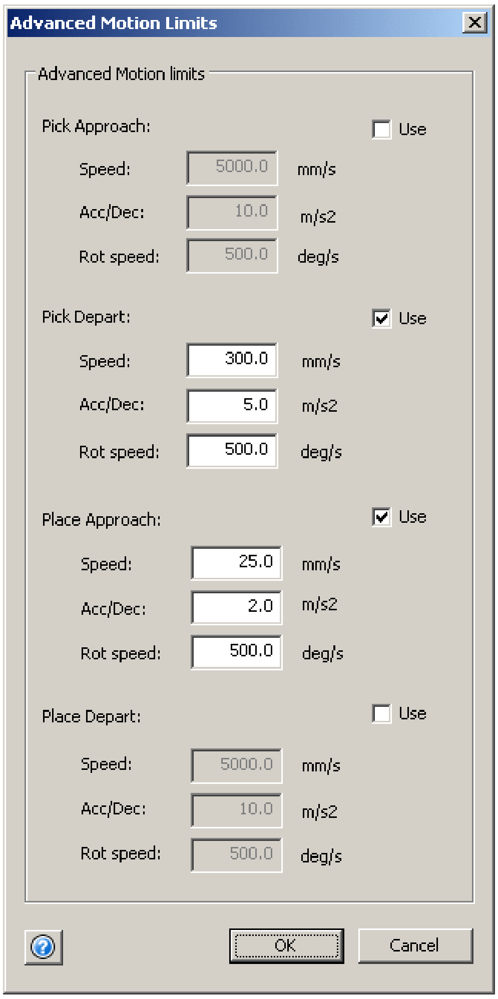
en1000000880

Settings	Here you define what shape to use for the item, the item weight and the facing. The display shows a 3-D view of the item. You can also view the tuned item size.
Motion limits	In this section you can define the default speed and acceleration limits for the item. You can also view the tuned speed and acceleration of the item. To edit separate motion configurations for the actions (Pick Approach, Pick Depart, Place Approach and Place Depart), click the **Advanced** button. Then select the **Use** check box of the action to use and edit. If the **Use** check box is cleared, the default settings will be used for this action.
Tool events	Here you set the timing of tool events to pick and place the item as efficient as possible. You can also view the tuned picking and placing time.

Continues on next page

Continued

Illustration, Advanced Motion Limits



Advanced Motion Limits

Advanced Motion limits

Pick Approach: □ Use

Speed: 5000.0 mm/s

Acc/Dec: 10.0 m/s2

Rot speed: 500.0 deg/s

Pick Depart: ☑ Use

Speed: 300.0 mm/s

Acc/Dec: 5.0 m/s2

Rot speed: 500.0 deg/s

Place Approach: ☑ Use

Speed: 25.0 mm/s

Acc/Dec: 2.0 m/s2

Rot speed: 500.0 deg/s

Place Depart: □ Use

Speed: 5000.0 mm/s

Acc/Dec: 10.0 m/s2

Rot speed: 500.0 deg/s

xx1000000168

How to proceed

This section describes how to proceed in the **Item Configuration** window.

Proceed with basic appearance

To...	Do this
Set item shape	In the **Shape** drop-down combo box, select the shape to edit.
Set item weight	In the **Weight** text box, type the weight (in kg).
Set I/O value	In the **I/O value** text box, type a value to identify different products.
Set item facing	Select the check boxes to define which sides of the product that will have a label. (Only applicable for products.)

Continues on next page

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. 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 limit motions

To...	Do this
Set maximum speed	In the **Speed** text box, type the maximum allowed speed.
Set acceleration/deceleration	In the **Acc/Dec** text box, type the acceleration value (in mm/s). The same value will be used for both acceleration and deceleration.
Set maximum rotation speed	In the **Rot Speed** text box, type the maximum allowed rotation speed (in deg/s).

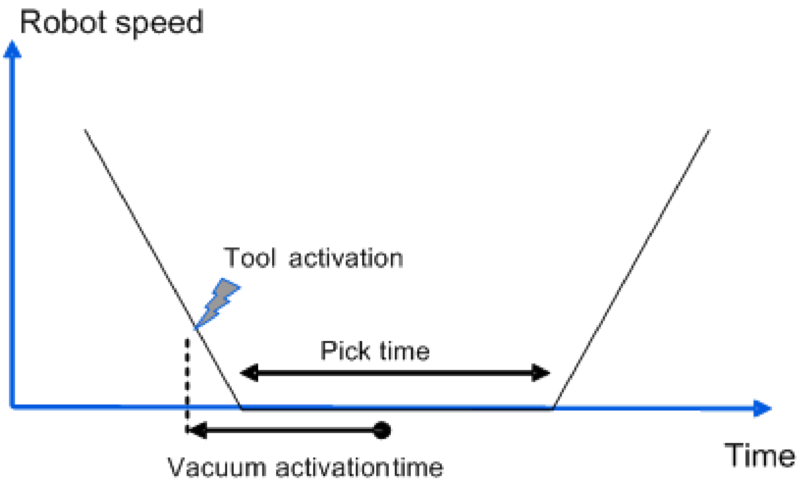
Note

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

Proceed with tool events

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.

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.

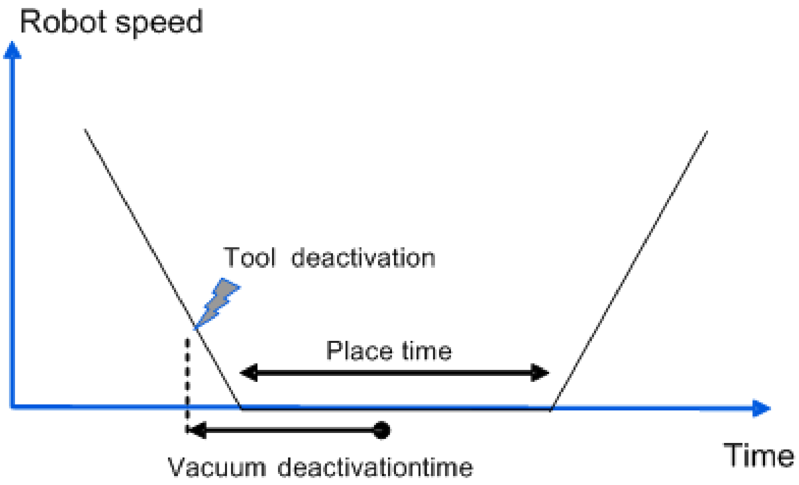


en1000000856

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

Continues on next page

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.



en1000000857

To...	Do this
Set time values for vacuum activation and deactivation.	In the **Vacuum activation time** and **Vacuum deactivation time** text boxes, type the values.
Adjust the time the robot stays at the target position when picking or placing an item.	In the **Pick time** and **Place time** text boxes, type the time in seconds.

Related information

The Shape Configuration on page 102.

4.4.5 The Format Configuration

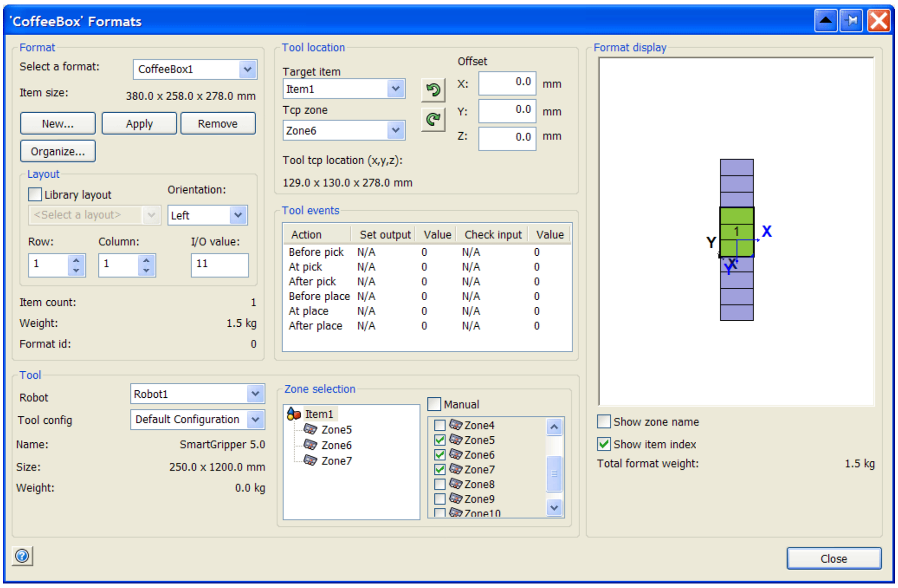
Overview

A PickMaster format defines how a group of items are located in and held by a robot tool. The item and tool orientations must be configured, as well as the tool zones to activate when gripping the items. Furthermore, tool I/O signal events can be defined to control specific tool functionality.

Start the Format Configuration

In the Project Panel, right-click an item and select **Formats**.

Illustration, Format Configuration



'CoffeeBox' Formats

Format

Select a format: CoffeeBox1

Item size: 380.0 x 258.0 x 278.0 mm

Layout

□ Library layout

<Select a layout>

Orientation: Left

Row: 1

Column: 1

I/O value: 11

Item count: 1

Weight: 1.5 kg

Format id: 0

Tool location

Target item: Item1

Tcp zone: Zone6

Offset

X: 0.0 mm

Y: 0.0 mm

Z: 0.0 mm

Tool tcp location (x,y,z):

129.0 x 130.0 x 278.0 mm

Tool events

Action	Set output	Value	Check input	Value
Before pick	N/A	0	N/A	0
At pick	N/A	0	N/A	0
After pick	N/A	0	N/A	0
Before place	N/A	0	N/A	0
At place	N/A	0	N/A	0
After place	N/A	0	N/A	0

Tool

Robot: Robot1

Tool config: Default Configuration

Name: SmartGripper 5.0

Size: 250.0 x 1200.0 mm

Weight: 0.0 kg

Zone selection

Item1
- Zone5
- Zone6
- Zone7

□ Manual

□ Zone4
☑ Zone5
☑ Zone6
☑ Zone7
□ Zone8
□ Zone9
□ Zone10

Format display

□ Show zone name

☑ Show item index

Total format weight: 1.5 kg

xx0600002984

Format	Select the specific format that you want to modify. You can also create, delete, and save the changes to the format. This is the general information about the format, the size, the weight, the orientation and so on. The layout information of the Format is also displayed here.
Tool	Specify the robot and its tool configuration that accesses the format and the zones used to hold specific items.
Tool location	Define the position of the robot tool when gripping the selected items.
Tool events	Define I/O events required for some formats, to pick and place the items.
Format Display	This is an overview of the format and an illustration of the selected items and the location of the tool. **Total format weight** shows the total weight of the format configuration, including the items' weight and tool weight.

Continues on next page

How to proceed

This section describes how to proceed in the **Format Configuration** window.

Proceed with format configuration

To...	Do this
Select a format	In the **Select a format** menu, select the name of the format. Information about the selected format will be displayed.
Create a new format	Click **New** to create a new format. A default name of the new format will be displayed in the **Select a format** menu.
Remove a format	Select the format to remove in the **Select a format** menu and then click **Remove**.
Apply changes on a format	Click **Apply** to save changes made to the selected format.
Organize formats	Click **Organize** to start the **Organize Formats** window, see *Proceed with Organize Formats on page 114*.

Proceed with layout configuration

To...	Do this
Select item orientation	In the **Orientation** combo box, select the item orientation to use. The selection defines the side of the item to place in negative y direction, see *Illustration, Item Configuration on page 106*. A blue marker shows the origin of each item.
Define item count	The number of items in the format is defined by number of item rows and item columns. Type a number in the **Row** and **Column** text boxes. The maximum number of rows/columns for each format is 20. The total number of items will be displayed in **Item count**. The **Format display** will show how items are related to each other. All items are listed in **Zone selection**.
Define I/O value	The I/O value is a combined value and is calculated from the orientation of the format and the number of rows and columns of occurrences of the shape in the format. The I/O value represents the different ways a format can arrive on a conveyor. You can also define your own I/O value.
Use a user-defined format	It is possible to use a user-defined layout for a format. This layout must exist in the PickMaster Library. Select the **Library layout** check box, then select a layout from the PickMaster Library in the layout menu.
Create a 2D format	Create a 2D Format by defining how many rows and how many columns the format should contain. The Format display will show how items are related to each other.

Note

The *Format id* in the **Format** part of the **Format Configuration** window is a unique value representing the format. This value can later be used in a RAPID program module to identify a format in an operation.

Continues on next page

Proceed with tool location

The tool is positioned by matching an item with a zone. The tool will be placed with the center point of the selected zone at the center point of the corresponding item.

To...	Do this
Define the tool location	1 In the **Target item** combo box, select an item. 2 In the **TCP zone** combo box, select a zone.
Adjust tool location	In the **X offset** and **Y offset** text boxes, type the value for the distance to move the tool.
Rotate the tool	Use the arrow buttons to rotate the tool either clockwise or counter-clockwise.

Note

The zone names and item indices can be shown in the Format display by selecting the corresponding check boxes. Thus is it easier to figure out which item and zone to match.

Proceed with tool zones

You must define which tool zones to use for each item in the format. By default, the zones are selected automatically and appear in the tree view in the **Zones** tab of the **Tool Configuration** window. When selecting an item, zones that are used for this item will be activated in the **Format** display.

If the selected zones do not meet the requirements, you can manually select the zones to use with a specific item.

	Action
1.	In **Zone selection**, select the **Manual** check box. Based on the layout of the format, PickMaster will calculate the total number of items for this format. All items in this format will be listed.
2.	In the item tree view, click the item you want to adjust.
3.	In the zones combo box, select the zones to use for the selected item.

Note

When you manually select zones for an item and if the selected zone is outside the item, a warning message appears below the graphical display.

Proceed with tool events

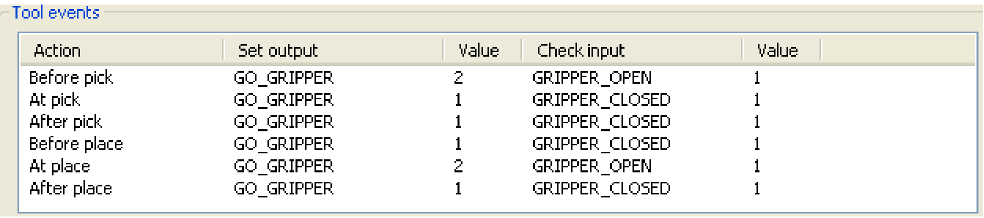
For some formats, the items cannot be picked or placed by only using the tool zones - for example, when picking a pallet using a specific *I/O driven gripper* instead of vacuum zones. In such cases it is possible to set digital or group output signals and wait for digital input signals at various positions when picking or placing a format.

These I/O events can be set for each pick or place operation at first action, target action and last action. The signals that are available in the **Tool events** list are the ones defined in the tool configuration. See *The **Tool Configuration** on page 81*. You

Continues on next page

Continued

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.



Tool events

Action	Set output	Value	Check input	Value
Before pick	GO_GRIPPER	2	GRIPPER_OPEN	1
At pick	GO_GRIPPER	1	GRIPPER_CLOSED	1
After pick	GO_GRIPPER	1	GRIPPER_CLOSED	1
Before place	GO_GRIPPER	1	GRIPPER_CLOSED	1
At place	GO_GRIPPER	2	GRIPPER_OPEN	1
After place	GO_GRIPPER	1	GRIPPER_CLOSED	1

xx0700000303

Action	Describes when the event occurs. The specific times are set in the **Item Configuration** dialog box, see *Proceed with tool events on page 108*.
Set output	The output signal that is set for a specific action.
Value	The value of the output signal.
Check input	The 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.
Value	The value of the input signal.

To...	Do this
Set an output signal	1 In the **Action** column, find the action to set the output signal for. 2 In the **Set output** column, select the signal. 3 In the **Value** column, type the value to set. Be careful to set the value for the correct row.
Check a digital input signal	1 In the **Action** column, find the action to check the input signal for. 2 In the **Check input** column, select the signal. 3 In the **Value** column, select the value to wait for. Be careful to set the value for the correct row.

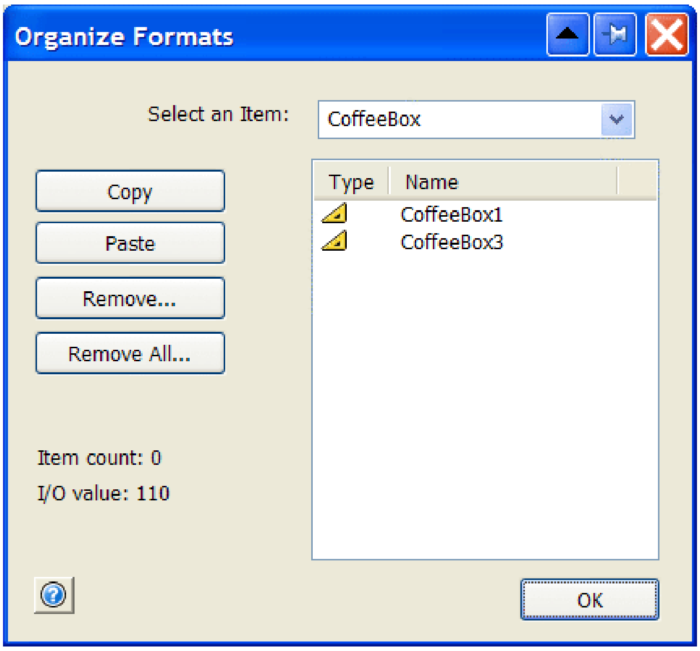
Note

If no zones should be activated when accessing the format, all zones should be manually cleared.

Continues on next page

Proceed with Organize Formats

As other PickMaster project object, the user is also able to copy/paste formats in an Item or between Items. This can be done in Organize Formats window which is launched when the user pressing the Organize button in the Format window.



en0800000287

To...	Do this
View all formats in an item	Select an item in the **Select an Item** menu. All formats belonging to the item will be displayed.
Rename a format	Select the name you want to change and type the new name.
Remove one or more formats	Select one or more formats and click **Remove**.
Remove all formats	Click **Remove All** to remove all formats in the item.
Copy formats	Select the formats you want to copy and click **Copy**. You can also right-click and select **Copy** or use CTRL + C.
Paste formats in the same item	First copy the formats. Then, click anywhere in the format list and click **Paste**.
Paste formats in another item	First copy the formats. Then, 1 In the drop-down list, select the item to which you want to add the copies. 2 Click anywhere in the format list, and click **Paste**. If the items are using different shapes, a warning message will be displayed. **Note!** It is not possible to paste formats into an item using different types of shapes, that is a format belonging to an item of shape Product cannot be pasted into an item of shape Pallet/SlipSheet.

Related information

*The **Tool Configuration** on page 81*.

4.4.6 The Pallet Pattern Configuration

Overview

A 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, depending on the type of shape used for that layer.

A pallet pattern is only defined with shapes and can consequently be used with different products. When configuring an operation set for a pallet pattern you define which item to use for a specific shape. For more information, see *The Pallet Pattern Operation Set Configuration on page 130*.

Start the Pallet Pattern Configuration

In the project panel

- Click the pallet pattern you want to edit, or
- Right-click the pallet pattern you want to edit and click **Edit**.

Illustrations, Pallet Pattern Configuration

The pallet pattern configuration consists of two dialog boxes, the **Layout Configuration** and the **Pallet Pattern Configuration**. In the **Layout Configuration** dialog box you select the layouts you want to use in your pallet pattern and in the **Pallet Pattern Configuration** dialog box you define how to position and order the selected layouts.

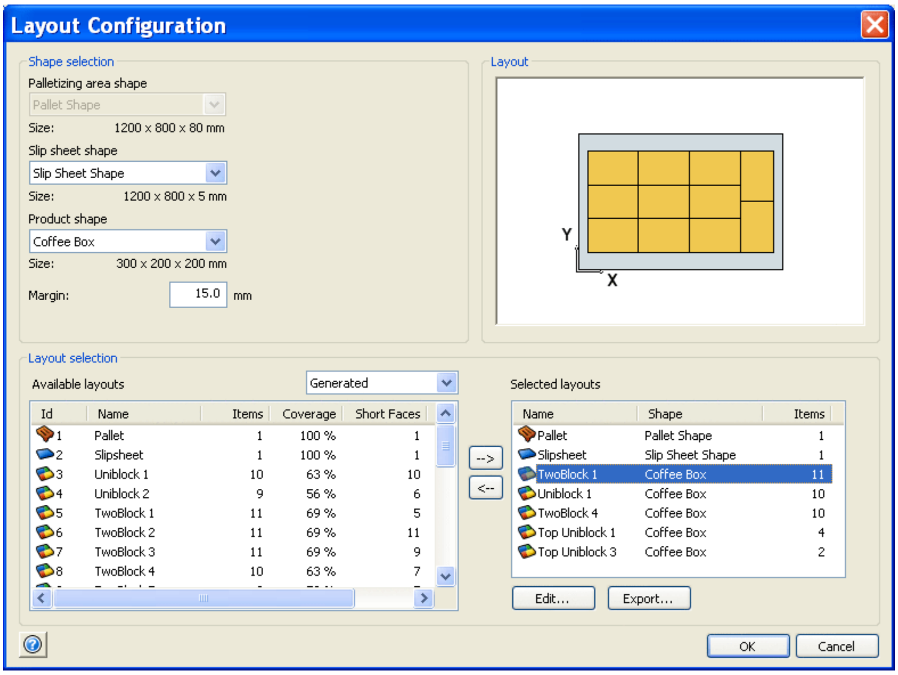
Note

When a pallet pattern is edited for the first time, the **Layout Configuration** appears to let you define the layouts to use in the pallet pattern. But when editing a pallet pattern that is already configured, the **Pallet Pattern Configuration** appears. To open the **Layout Configuration** dialog box, click the **Setup** button in the **Pallet Pattern Configuration** dialog box.

Continues on next page

Continued

The Layout Configuration



Palletizing area shape: Pallet Shape
Size: 1200 x 800 x 80 mm

Slip sheet shape: Slip Sheet Shape
Size: 1200 x 800 x 5 mm

Product shape: Coffee Box
Size: 300 x 200 x 200 mm

Margin: 15.0 mm

Available layouts: Generated

Id	Name	Items	Coverage	Short Faces
1	Pallet	1	100 %	1
2	Slipsheet	1	100 %	1
3	Uniblock 1	10	63 %	10
4	Uniblock 2	9	56 %	6
5	TwoBlock 1	11	69 %	5
6	TwoBlock 2	11	69 %	11
7	TwoBlock 3	11	69 %	9
8	TwoBlock 4	10	63 %	7

Selected layouts

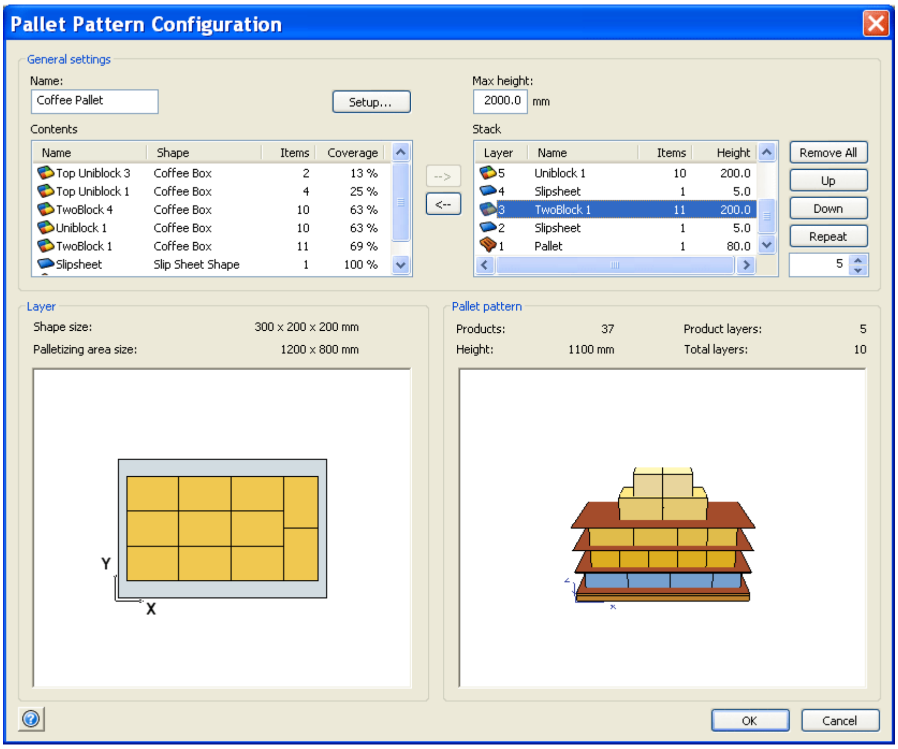
Name	Shape	Items
Pallet	Pallet Shape	1
Slipsheet	Slip Sheet Shape	1
TwoBlock 1	Coffee Box	11
Uniblock 1	Coffee Box	10
TwoBlock 4	Coffee Box	10
Top Uniblock 1	Coffee Box	4
Top Uniblock 3	Coffee Box	2

xx0600002992

Shape selection	Here you select the shapes to use in the pallet pattern.
Layout	Shows an overview of the selected layout.
Layout selection	In this area you define what layouts should be used in the pallet pattern. You can select from calculated layouts or layouts saved in the library or edit your own layouts.

Continues on next page

The Pallet Pattern Configuration



Pallet Pattern Configuration

General settings

Name: Coffee Pallet

Max height: 2000.0 mm

Contents

Name	Shape	Items	Coverage
Top Uniblock 3	Coffee Box	2	13 %
Top Uniblock 1	Coffee Box	4	25 %
TwoBlock 4	Coffee Box	10	63 %
Uniblock 1	Coffee Box	10	63 %
TwoBlock 1	Coffee Box	11	69 %
Slipsheet	Slip Sheet Shape	1	100 %

Stack

Layer	Name	Items	Height
5	Uniblock 1	10	200.0
4	Slipsheet	1	5.0
3	TwoBlock 1	11	200.0
2	Slipsheet	1	5.0
1	Pallet	1	80.0

Layer

Shape size: 300 x 200 x 200 mm

Palletizing area size: 1200 x 800 mm

Pallet pattern

Products: 37

Height: 1100 mm

Product layers: 5

Total layers: 10

xx0600002993

General settings	Here you define how to stack your layouts with various shapes to get the pallet pattern you want.
Layer	Shows an overview of the selected layer.
Pallet pattern	Shows an overview of resulting pallet pattern.

To access the Layout Configuration dialog box:

In the **Pallet Pattern Configuration** dialog box, click **Setup**.

How to proceed

This section describes how to proceed with the pallet pattern configuration.

Proceed with defining layouts

A pallet pattern is built as a stack of shape layouts. Each layout may only contain one type of shape and you must select shapes for palletizing area, slip sheet and product. For information about how to create shapes, see *The Shape Configuration on page 102*. Layouts are defined in the **Layout Configuration** dialog box.

To...	Do this
Select shapes	In the combo boxes in the **Shape selection** part, select the shapes to use.
Set layout margin	In the **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 shape.

Continues on next page

Note

You must always select a shape for the palletizing area, even if no pallet is to be included in the pallet pattern. The pallet shape defines the maximum size of each layout. The pallet shape cannot be changed once it is used but slip sheet and product shapes can be changed to build a pallet pattern with different shapes.

When the appropriate shapes are selected, the **Available layouts** list will be propagated with the generated layouts. The layouts are generated for the selected shapes 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 PickMaster Library.

After you have selected a new slip sheet, product shape, or given a new margin, new layouts are generated based on the new information. The new layouts are listed in the **Available layouts** list. All layouts in the **Selected layouts** list still use the old shape and margin.

To...	Do this
Access library shapes	In the combo box at **Available layouts**, select **Library**. This will show all layouts from the library converted to use the selected shapes.
Select layouts to use in the pallet pattern	In the **Available layouts** list, select the layouts of interest and either use the right arrow button or drag them to the **Selected layouts** list.
Edit a layout	In the **Selected layouts** list, select the layout and click **Edit**. This opens up the **Layout Editor**, see *The Layout Editor on page 120*.
Export a layout to the library	In the **Selected layouts** list, select the layout and click **Export**.
Show a layout	In either the **Available layouts** or the **Selected layouts** lists, select the layout you want to view.
Rename a layout	In the **Selected layouts** list, double-click on the layout name and type the new name. The name can also be changed in the **Layout Editor**.
Remove layout from selected list	In the **Selected layouts** list, select the layouts to remove and either click the left arrow button or drag the layouts to the **Available layouts** list.
Copy and paste a layout	In the **Selected layouts** list, select the layout you want to copy. Right-click and select **Copy**. Then click anywhere in the list. Right-click and select **Paste**.

Continues on next page

Proceed with defining pallet pattern layers

When the layouts to use in your pallet pattern are defined, you continue with setting the order and number of layers in the **Pallet Pattern Configuration** dialog box. The resulting pallet pattern is always shown in the Pallet pattern section together with the stack details.

To...	Do this
Define the stack	In the **Contents** list, select the layouts you want to use and either click the right arrow button or drag the layouts to the **Stack** list.
Remove layers	In the **Stack** list, select the layers you want to remove and either click the left arrow button or drag the layers to the **Contents** list. You can also click the **Remove All** button to remove every layer in the stack.
Reorder layers	In the **Stack** list, select the layers you want to move and click the **Up** or **Down** button to reorder the layers in the stack.
Duplicate layers	In the **Stack** list, select the layers you want to duplicate and type the number of copies to add to the top of the stack. Click the **Repeat** button to perform the operation.
Limit stack height	In the **Max** height text box, enter the maximum allowed stack height (in mm). When adding layouts to the stack either from the **Contents** list or by using the duplication function, you are warned if the total stack height will exceed the given limit.
Show a layer	In **Stack layer** list, select the layout you want to view and that layer will be shown in the **Layer** section.
Edit the selected layouts	Click the **Setup** button to open the **Layout Configuration**.

Related information

The Item Configuration on page 105.

The Layout Editor on page 120.

The PickMaster Library on page 51.

4.4.7 The 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 PickMaster 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:

- From the PickMaster Library, if you want to edit a layout that is included in the library, or
- From a pallet pattern configuration, if you want to edit a layout that is included in a pallet pattern.

Start from the PickMaster Library

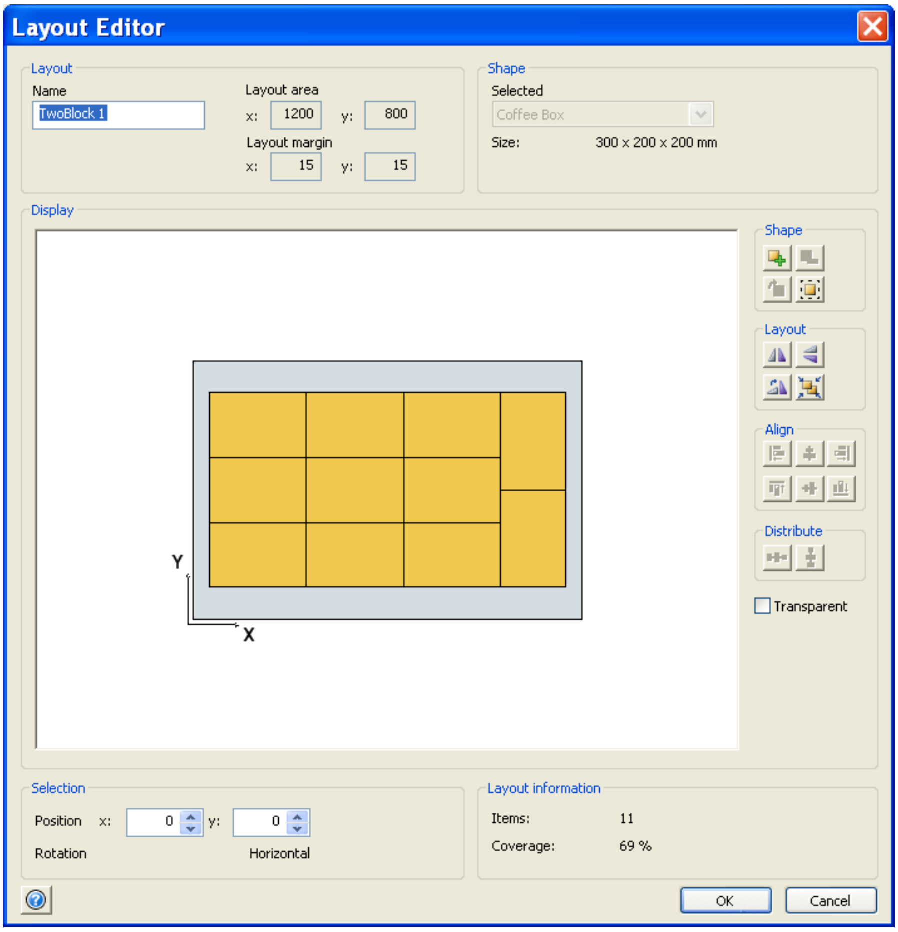
	Action
1.	In the PickMaster library tree view, select a layout item to edit and right-click.
2.	Click **Edit**.
3.	The **Layout Editor** window appears and you can edit the layout.

Start from a pallet pattern configuration

	Action
1.	Double-click the **Pallet pattern** that includes the layout to edit.
2.	The **Pallet Pattern Configuration** dialog box appears. Click **Setup**.
3.	The **Layout Configuration** dialog box appears. In the **Selected layouts** list, select the layout to edit.
4.	Click **Edit**.
5.	The **Layout Editor** window appears and you can edit the layout.

Continues on next page

Illustration, Layout Editor


Layout Editor

Layout
Name: TwoBlock 1
Layout area x: 1200 y: 800
Layout margin x: 15 y: 15

Shape
Selected: Coffee Box
Size: 300 x 200 x 200 mm

Selection
Position x: 0 y: 0
Rotation: Horizontal

Layout information
Items: 11
Coverage: 69 %

xx0600002781

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.
Shape	The drop-down combo box shows the selected shape. If you add a new shape, it will be added to the layout. If there are no shapes in the layout, you select which shape to add. When you edit a library layout you can change the selected shape to any of the available shapes in the library. When you edit a layout in a pallet pattern, you cannot change the shape. First you change the shape in the **Layout Configuration** dialog box, and then edit the shape. Size is given in x, y and z size of the current shape.
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. For descriptions of how to use the buttons, see *How to proceed on page 122*. The **Transparent** check box provides an option to show the shapes half-transparent, which makes it easier to find overlapping items.
Selection	Shows the position and orientation of the selected item. You can adjust the position by editing the values in the text boxes.
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.

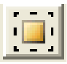
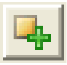
Continues on next page

Note

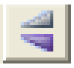
The last selected item is marked with a dim dotted rectangle and is called the *marked item*. It is always the position values of the *marked item* that appear in the **Selection** area. The *marked item* is also important when using the Align and Distribute buttons. See *The Layout Editor on page 120*.

How to proceed

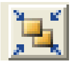
Proceed with the shape

To...	Do this
Select multiple items	Press and hold the CTRL key while you click the items to select.
Select all items	Click **Select All**, or perform a select all operation (CTRL+A).  xx0600002814
Add a new item	Click **Add**.  xx0600002811
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. xx0600002812
Rotate the selected item/items	Click **Rotate**. xx0600002813

Proceed with the layout

To...	Do this
Mirror the layout in x direction	Click **Flip Horizontal**. xx0600002807
Mirror the layout in y direction	Click **Flip Vertical**.  xx0600002808
Rotate the entire layout 180°	Click **Rotate**. xx0600002809

Continues on next page

To...	Do this
Center the layout	Click **Center**.  xx0600002810

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.

Note

Clicking **Rotate** for square layouts will rotate the layout 90°.

Tip

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

Related information

The PickMaster Library on page 51.

Setting up a pallet pattern on page 168.

Terminology on page 24.

4.4.8 The Position Source Configuration

Overview

The position source is a central object in a project. It defines what to pick or place on which work area. There must be one position source defined for each work area to use in the project. The position source holds a number of operation sets, which, in turn, defines how the robot should access a format or pallet pattern. For more information about operation sets, see *The Operation Set Configuration on page 128*.

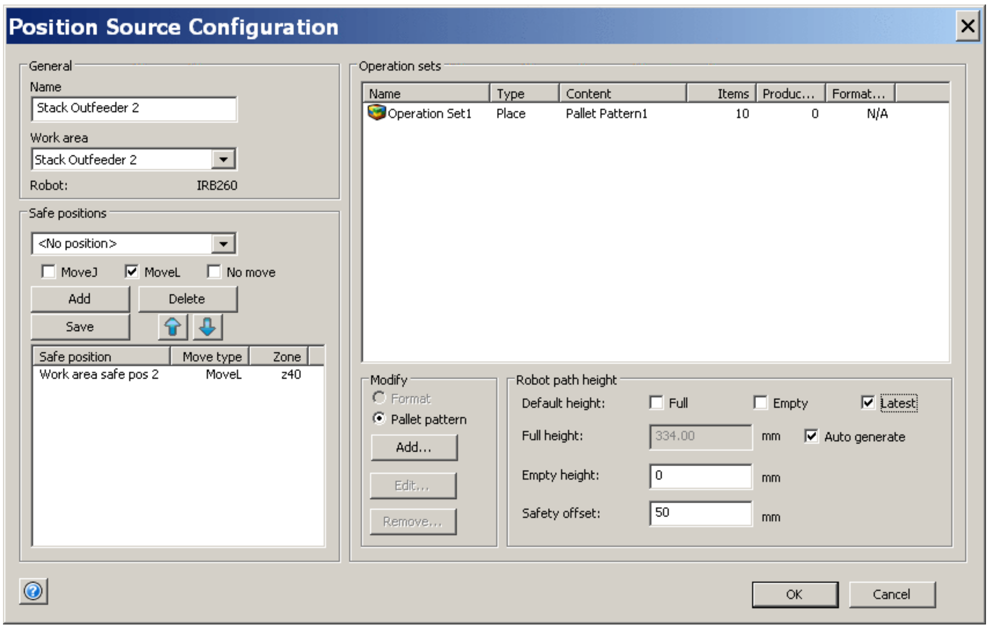
Start the Position Source Configuration

In the project panel

- double-click a position source to edit, or
- right-click a position source and click **Edit**.

Continues on next page

Illustration, Position Source Configuration



Position Source Configuration

General

Name: Stack Outfeeder 2

Work area: Stack Outfeeder 2

Robot: IRB260

Safe positions

<No position>

☐ MoveJ ☑ MoveL ☐ No move

Safe position	Move type	Zone
Work area safe pos 2	MoveL	z40

Operation sets

Name	Type	Content	Items	Produc...	Format...
Operation Set1	Place	Pallet Pattern1	10	0	N/A

Modify

○ Format
● Pallet pattern

Robot path height

Default height: ☐ Full ☐ Empty ☑ Latest

Full height: 334.00 mm ☑ Auto generate

Empty height: 0 mm

Safety offset: 50 mm

xx0600002994

General	Select the work area to generate positions for the operation sets.
Robot position	Define the safe positions that the robot will pass before entering and after leaving the selected work area. You can define several safe positions to let the robot follow a path when entering and leaving the work area. 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 work area and the first that the robot will pass when leaving the work area. The last safe position in the list will also define the reference arm and wrist configuration of the robot when operating the work area.
Operation sets	Create and edit operation sets.
Robot path height	Defines some height attributes that are considered when the robot is moving to, from or over the work area. 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. For description about the configuration parameters, see *Robot path height, configuration parameters on page 125*.

Robot path height, configuration parameters

Parameter	Description
Default height	The default height is the expected height of the work area 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 `PmSetDefaultHeight`. 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 work areas and flows.

Continues on next page

Parameter	Description
Full height	Defines the full height of the work area 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	Defines the empty height of the work area in the work object frame (mm).
Safety offset	An offset that is always added to the expected height of the work area.

How to proceed

This section describes how to proceed in the **Position Source Configuration** window.

Proceed with position source configuration

To...	Do this
Select work area	In the **Work area** drop-down combo box, select the work area to generate position for.
Select robot position	See *Proceed with safe positions on page 126*.
Create an operation set	In the **New** section 1 Select either **Format** or **Pallet Pattern** depending on the type of operation set you want to create. 2 Click **Add**.
Edit an operation set	Select the operation set to edit in the list, and click **Edit**.
Remove an operation set	Select the operation set to remove in the list, and click **Remove**.

Note

When the Target Generation Selection signal is not set on the selected work area, there can only be one operation set in the position source. See *The Operation Set Configuration on page 128* for more information.

Proceed with safe positions

To...	Do this
Select a safe position	In the **Robot position** drop-down list, select the safe positions that should be used. Used safe positions will not show in the drop-down list.
Select a move type	Click **MoveJ**, **MoveL** or **No move** to select move type. **No move** means the robot will not pass the safe position. However, if a no move safe position is last in the list, it will become the reference arm and wrist configuration of the robot when operating the work area.
Add a safe position to the list	Select a safe position from the drop-down list and select move type. Click **Add** to add the selected safe position to the list.
Delete a safe position from the list	Select the safe position and click **Delete** to remove it from the list. The removed safe position will appear in the drop-down list again.
Change move type for a safe position in the list	Select the safe position in the list and click the move type value. Select move type in the drop-down list.
Change the order of the safe positions	Select a safe position and then click the arrow buttons up or down.

Continues on next page

Continued

To...	Do this
Save the selected safe positions	Click **Save** to save the changes made to all safe positions.

Related information

The Format Configuration on page 110.

The Work Area Configuration on page 73.

The Pallet Pattern Configuration on page 115.

The Operation Set Configuration on page 128.

4.4.9 The Operation Set Configuration

Overview

An operation set defines how positions are generated for a specific work area. There are two types of operation sets, the **Format Operation Set** defining how to pick or place a format and the **Pallet Pattern Operation Set** defining how to pick or place a complete pallet pattern. Operation sets are created from a position source and consequently locked to a work area and a robot with its tool.

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

Operation sets defined for slave work areas specifies how requested formats are to be picked or placed.

Start the Operation Set Configuration

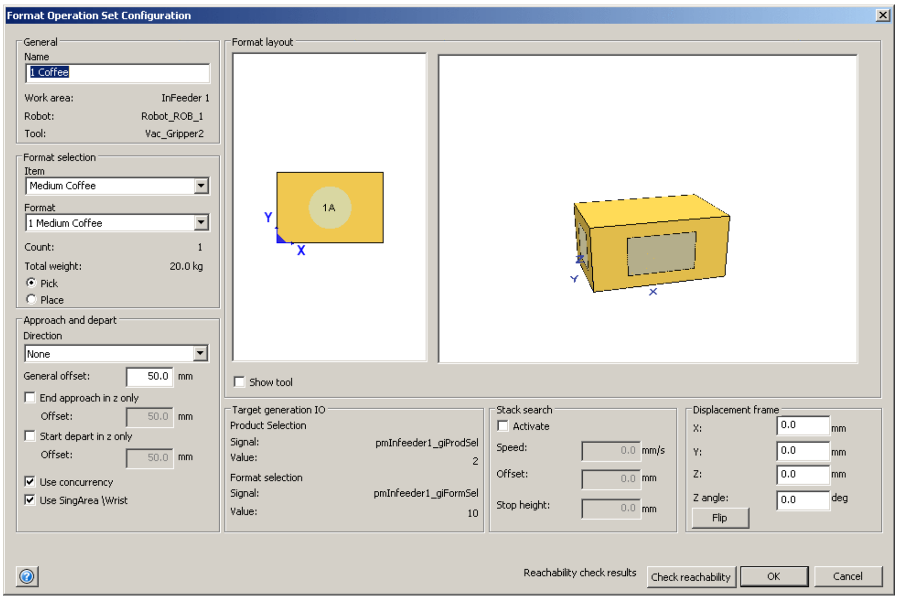
Operation sets are owned by position sources and are accessed from there. For more information about how to access operation sets, see *The Position Source Configuration on page 124*.

Continues on next page

Illustrations, Operation Set Configuration

The format and pallet pattern operation sets are configured in different dialog boxes, but they have several settings in common.

The Format Operation Set Configuration



General
Name: 1 Coffee
Work area: InFeeder 1
Robot: Robot_ROB_1
Tool: Vac_Gripper2

Format selection
Item: Medium Coffee
Format: 1 Medium Coffee
Count: 1
Total weight: 20.0 kg
● Pick
○ Place

Approach and depart
Direction: None
General offset: 50.0 mm
□ End approach in z only
Offset: 50.0 mm
□ Start depart in z only
Offset: 50.0 mm
☑ Use concurrency
☑ Use SingArea \Wrist

□ Show tool

Target generation IO
Product Selection
Signal: pmInfeeder1_giProdSel
Value: 2
Format selection
Signal: pmInfeeder1_giFormSel
Value: 10

Stack search
□ Activate
Speed: 0.0 mm/s
Offset: 0.0 mm
Stop height: 0.0 mm

Displacement frame
X: 0.0 mm
Y: 0.0 mm
Z: 0.0 mm
Z angle: 0.0 deg

Reachability check results

xx0600003005

General	Set the name of the operation set and view robot information.
Format selection	Select the item and format to use and how to access the format. This section shows also the item count and the total weight of the format.
Approach and depart	Define the robot path to follow when picking or placing the format.
Format layout	Displays an overview of the format. The two-dimensional display shows how the items are grouped together in different operations. For more information, see *Display information on page 131*. To view how the robot tool accesses the format, select the **Show Tool** check box.
Target generation I/O	Displays the input signals from the online PLC when the targets have been generated.
Stack search	Select the stack search if you have clicked the **Pick** button in the **Format selection**. To select the stack search, select the **Activate** check box. For descriptions of the configuration parameters for the stack search, see table below.
Displacement frame	If needed, select to view how the robot tool accesses the various formats in a layer. To rotate the frame 90 degrees, click the **Flip** button.Offsets and angle will be updated to keep chosen alignment.
Check reachability	See *Proceed with Check reachability on page 140*.

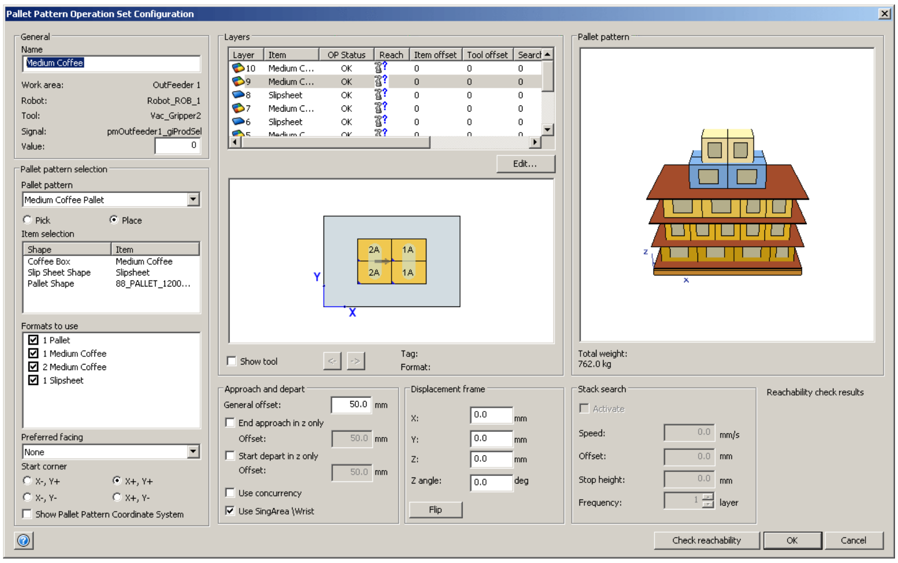
The following table describes the configuration parameters for the stack search.

Speed	Defines the TCP speed of the robot when it searches for the format.
Offset	Defines an offset of the distance between the expected position of the format, and the starting position of the robot's search movement. The total search offset will also include the product height of the format.

Continues on next page

Stop height	Defines the TCP height above the work object where the robot will stop the search movement (if search stop never occurs).

The Pallet Pattern Operation Set Configuration



Pallet Pattern Operation Set Configuration

General

Name: Medium Coffee

Work area: OutFeeder 1

Robot: Robot_ROB_1

Tool: Vac_Gripper2

Signal: pmOutfeeder1_giProdSel

Value: 0

Pallet pattern selection

Pallet pattern: Medium Coffee Pallet

○ Pick ● Place

Item selection

Shape	Item
Coffee Box	Medium Coffee
Slip Sheet Shape	Slipsheet
Pallet Shape	88_PALLET_1200...

Formats to use

☑ 1 Pallet
☑ 1 Medium Coffee
☑ 2 Medium Coffee
☑ 1 Slipsheet

Preferred facing: None

Start corner

○ X-, Y+ ● X+, Y+

○ X-, Y- ○ X+, Y-

□ Show Pallet Pattern Coordinate System

Layers

Layer	Item	OP Status	Reach	Item offset	Tool offset	Search
10	Medium C...	OK	?	0	0	0
9	Medium C...	OK	?	0	0	0
8	Slipsheet	OK	?	0	0	0
7	Medium C...	OK	?	0	0	0
6	Slipsheet	OK	?	0	0	0
5	Medium C...	OK	?	0	0	0

Edit...

□ Show tool <- ->

Tag:

Format:

Pallet pattern

Total weight: 762.0 kg

Approach and depart

General offset: 50.0 mm

□ End approach in z only

Offset: 50.0 mm

□ Start depart in z only

Offset: 50.0 mm

□ Use concurrency

☑ Use SingArea \Wrist

Displacement frame

X: 0.0 mm

Y: 0.0 mm

Z: 0.0 mm

Z angle: 0.0 deg

Flip

Stack search

□ Activate

Speed: 0.0 mm/s

Offset: 0.0 mm

Stop height: 0.0 mm

Frequency: 1 layer

Reachability check results

Check reachability OK Cancel

xx0600003006

General	Set the name and I/O value of the Operation Set. Note that the **I/O value** text box is only available if the **Product selection (GO)** signal is defined in the **Target Generation group** in the **Work Area Configuration** window.
Pallet pattern selection	Select the pallet pattern to use and how to access it. To select a different item for the shape, click the item and a drop-down box appears. In the drop-down box, select the item to use. This section shows also the total weight of the pallet pattern.
Layers	This section lists and shows information about all layers in the pallet pattern. To edit layer offsets, click directly on the column and edit the displayed value. To edit the operations for a layer, click on the layer and then click **Edit**. Any edited layers will be specifically marked. See *The Operation Editor on page 143* for more information. The display shows how the items are grouped together in different operations. For more information, see *Display information on page 131*. By selecting the **Show tool** check box you can view how the robot tool accesses the various formats in the layer.
Pallet pattern	Shows an illustration overview of the pallet pattern.
Approach and depart	Define the robot path to follow when picking or placing the formats in the pallet pattern.
Displacement frame	If needed, select to view how the robot tool accesses the various formats in a layer. To rotate the frame 90 degrees, click the **Flip** button.Offsets and angle will be updated to keep chosen alignment.
Stack search	You can select the stack search if you have clicked the **Pick** button in the **Pallet pattern selection**. To select the stack search, select the **Activate** check box. For descriptions of the configuration parameters for the stack search, see later in this section. See also *Stack search on page 131* for more information.
Check reachability	See *Proceed with Check reachability on page 140*.

Continues on next page

The following table describes the configuration parameters for the stack search.

Speed	Defines the TCP speed of the robot when it searches for the next layer.
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.
Stop height	Defines the TCP height above the work object where the robot will stop the search movement (if search stop never occurs).
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.

One offset is available to adjust the search offset for each layer. Additional search offset is used if stack search is configured for the layer.

Use a positive value to increase the search approach height from the defined search offset. Use a negative value to decrease the approach height. There is no limitation on search offsets but the expected height of the layer when starting the search movement defines a lower limit in runtime.

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 format operation sets there will only be one operation with one target, but for pallet patterns the items will be automatically grouped into several operations.

The three-dimensional display shows an overview of the resulting operation set, including the facing of all the items.

Stack search

With the stack search function it is possible to configure runtime calibrations of the stack height for a pallet pattern or format.

The function is useful to:

- Handle picking from pallet patterns where the initial number of layers varies from time to time - for example, pallet stations or slip sheet stations.
- Handle picking from pallet patterns or formats where the height of the layer/layers varies a lot from the configured height.

Continues on next page

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/format 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 pattern (or format) 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 *Illustration, Work Area Configuration on page 74*.

Configuration of stack search

You do the configuration of the stack search in the **Pallet Pattern Operation Set Configuration** and **Format Operation Set configuration** dialog boxes. See *The Pallet Pattern Operation Set Configuration on page 130* and *The Format Operation Set Configuration on page 129*.

Configuration of search tool

You do the configuration of the search tool in the **Search** tab in the **Tool Configuration** window. See *Search tab on page 89*.

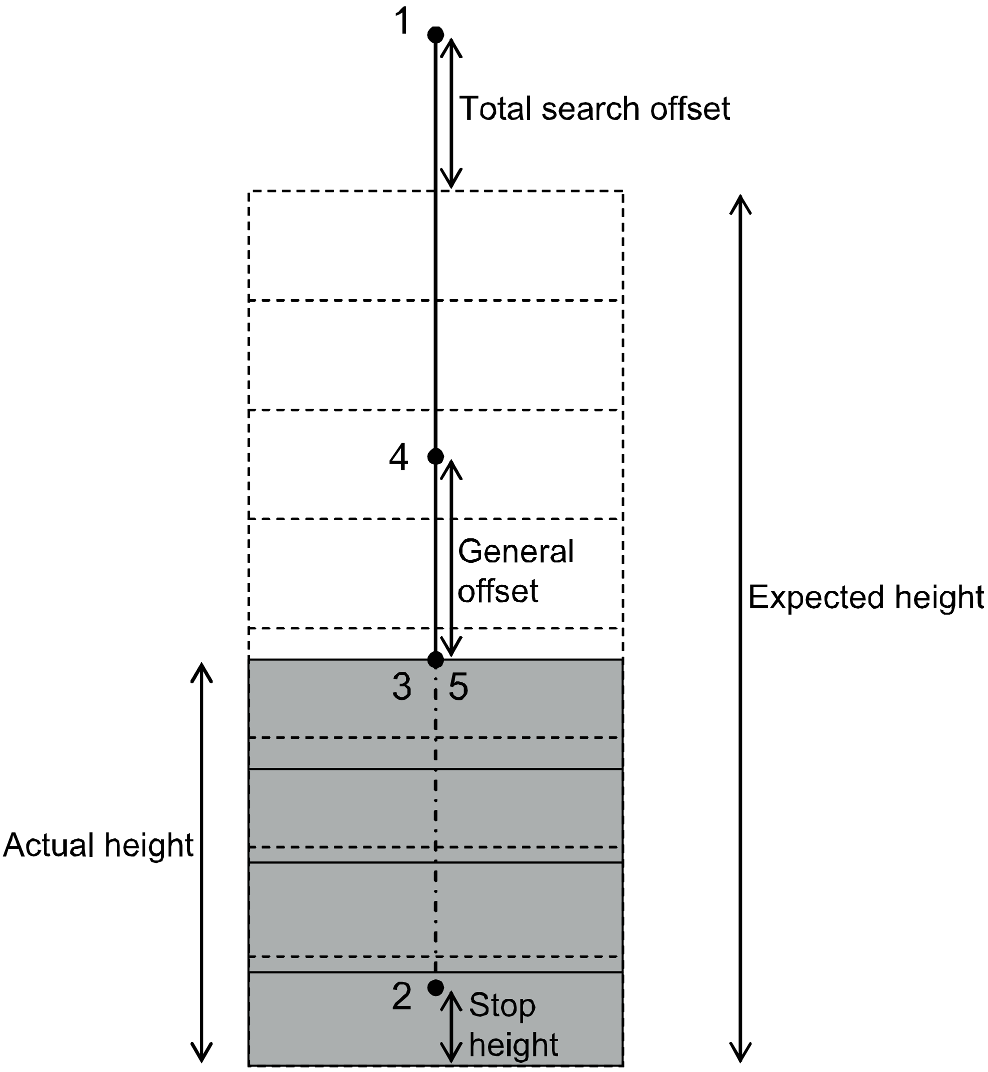
To configure the search tool, you have to:

- Select `tooldata` that has a TCP offset corresponding to the location of the top layer at search stop. This `tooldata` will only be used (without any tcp offset modification) when moving to the layer start position of the search movement and when performing the search movement.
- Select the sensor input signal for search stop.
- Select the output signals for activation and deactivation of the search tool.

Continues on next page

Stack search target sequence, example

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



xx0600003459

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.
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.

Continues on next page

How to proceed

This section describes how to proceed in the Operation Set Configuration windows.

Proceed with Format selection

This information is only applicable to format operation sets.

To...	Do this
Select a format	In the **Item** drop-down box, select the format to use in the operation set. In the **Format** drop-down box, select the format to use in the current operation set. Only formats not used in another operation set in this position source are listed. If you select a format with 2D layout or Library layout, then operations will automatically be generated for each layer. A message will be displayed under the **Layers** list in the **Pallet Pattern Operation Set Configuration** that the generated operations might not be correct or well optimized. All operations should be modified manually.
Select type	Select either the **Pick** or **Place** option button depending on the type of operation set you are configuring.

Proceed with Pallet pattern selection

This information is only applicable to pallet pattern operation sets.

Since a pallet pattern is a collection of shapes, you must select which item to use for every shape included in the pallet pattern. When the robot accesses a pallet pattern, it must be done using predefined formats. Therefore, you must also select the formats to use in the operation set.

The list in the Layers section shows information on all the layers in the pallet pattern. Select the layer you want to view in the two-dimensional display.

To...	Do this...
Select a pallet pattern	In the **Pallet pattern** combo box, select the pallet pattern to use in the operation set.
Select type	Select either the **Pick** or **Place** option button depending on the type of operation set you are configuring.
Select items	In the **Item selection** list box 1 Click on the item name next to the shape you want to configure. 2 Select the item from the combo box that is shown in the item column.
Select formats	In the **Formats to use** list box, check the formats that you want to include in the operation set. It is important to select formats in such a way that it is possible to complete each layer.
Define facing	In the **Preferred facing** combo box, select the side of the pallet pattern where you want to maximize item facing. The default setting is *none* and usually that generates the most efficient operations.
Select start corner	In the **Start corner** section, select the option button that corresponds to the corner of the pallet pattern where the robot should start. To display the pallet pattern coordinate system, select the Show Pallet Pattern Coordinate System check box.

Continues on next page

Note

The resulting operation set is continuously updated and displayed when any of the settings are changed. Layers that are edited are marked and will not be changed.

Proceed with offsets for each layer

You can configure adjustments of the general offset and the search offset for individual layers in the Layers 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.

To...	Do this
Modify any of the offset values	Select the offset you want to edit and type a new value. Press Enter to apply the change.

Proceed with the Approach and depart

In this section you define how the robot should move when accessing formats in the operation set. The movement will consist of 3 to 7 positions, depending on the selected settings. The position in which the robot actually picks or places the format is known as the target. Positions preceding the target are called approach and positions following the target are called depart.

Option	Description
Direction	Only available for format operation sets. Defines the direction in the x-y plane of the format when the robot is approaching a target for placing a format or departing a target after picking a format.

Continues on next page

Continued

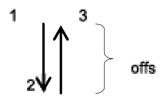
Option	Description
General offset	Defines the general pick and place approach/depart height. It also defines the movement safety distance to keep between robot held items and their nearby items in a pallet pattern. See the robot movement tables on the following pages for a detailed approach and depart movement description when picking and placing items.
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** text 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** text box, type the extra depart distance.
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: • The wrist axes will never wind up during a long sequence of only linear movements. • The 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

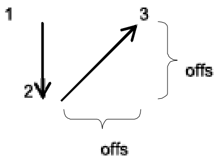
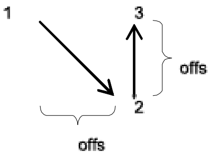
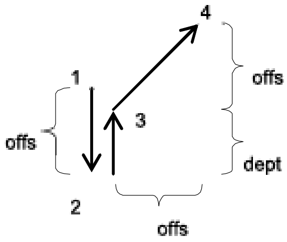
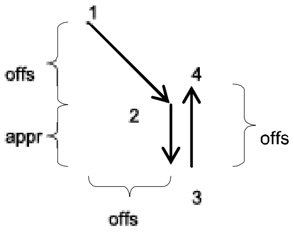
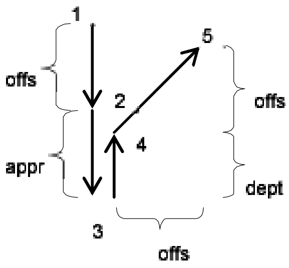
For more information about the concurrency and SingArea settings, see *Technical reference manual - RAPID Instructions, Functions and Data types*.

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

- General offset is *offs*.
- End approach in z is *appr*.
- Start depart in z is *dept*.

Type	Direction	End approach	Start depart in z only	Position count in z only	Movement
any	none	no	no	3	 en0600003007

Continues on next page

Type	Direction	End approach	Start depart in z only	Position count in z only	Movement
pick	any	no	no	3	 en0600003008
place	any	no	no	3	 en0600003009
pick	any	no	yes	4	 en0600003010
place	any	yes	no	4	 en0600003011
pick	any	yes	yes	5	 1 5 offs 2 offs 4 appr dept 3 offs en0600003012

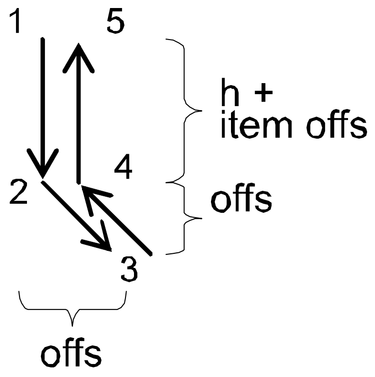
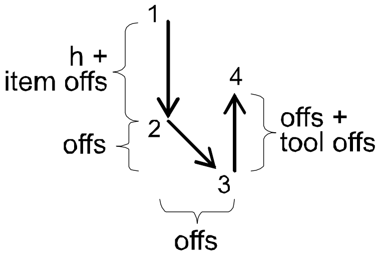
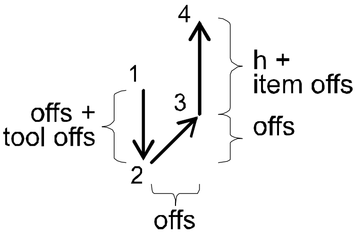
For pallet pattern operation sets the movements are generated a little bit differently. Since multiple picking or placing is supported, the height of the current item must be

Continues on next page

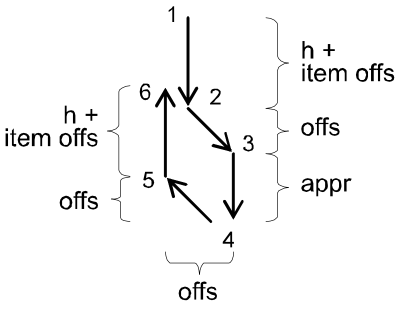
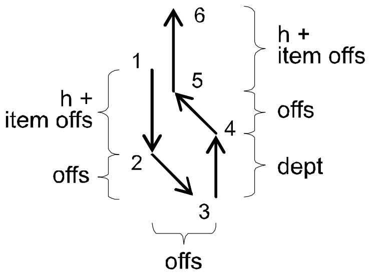
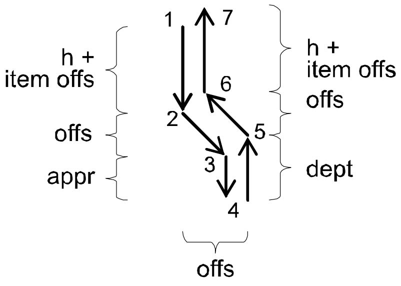
taken into account. The special cases, when the last item is placed as well as when the first item is picked, are handled separately.

The table below illustrates the resulting movement for a pallet pattern operation set with different settings. Use the following assumptions:

- General offset is *offs*.
- End approach in z is *appr*.
- Start depart in z is *dept*.
- Layer item offset is *item offs*.
- Layer tool offset is *tool offs*.
- Item height is *h*.

End approach in z only	Start depart in z only	Special case	Position count	Movement
no	no	no *	5	 en0600003013
no	no	last place	4	 1 h + item offs 4 2 offs + tool offs offs 3 offs en0600003014
no	no	first pick	4	 4 h + item offs 1 3 offs + tool offs offs 2 offs en0600003015

Continues on next page

End approach in z only	Start depart in z only	Special case	Position count	Movement
yes	no	no *	6	 1 2 3 4 5 6 h + item offs offs appr h + item offs offs offs en0600003016
no	yes	no *	6	 6 1 5 4 2 3 h + item offs offs dept h + item offs offs offs en0600003017
yes	yes	no *	7	 1 7 6 2 5 3 4 h + item offs offs dept h + item offs offs appr offs en0600003018

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

Note

The statuses of the robot 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 *The **Tool Configuration** on page 81*, *The Item Configuration on page 105*, and *The Format Configuration on page 110*.

Continues on next page

Proceed with the I/O interface

Whenever the target generation signal is set for a work area, the positions in an operation set are sent to that work area. If different operation sets are required in a position source, the target generation selection signal must be set for the corresponding work area 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.

To...	Do this
Set I/O value	In the **Value** text box, type the I/O value for the operation set.

If the target generation selection signal is not configured for the corresponding work area, the I/O value cannot be changed and only one operation set is allowed for the position source. When using several operation sets in the same position source, all operation sets must have unique I/O values. For more information about the target generation signals, see *The Work Area Configuration on page 73*.

Proceed with the Display Options

Display options are only available for pallet pattern operation sets.

To...	Do this
Show/hide the tool	Select the **Show tool** check box to show the tool in the two-dimensional display and clear it to hide the tool.
Show tool location	Use the left and right arrow buttons to show the tool location for the various targets in the layer.

Proceed with Stack search

To...	Do this
Activate stack search	Select the **Activate** check box. When stack search is activated, you can enter values in the text boxes.
Set speed	In the **Speed** text box, type the value the robot should use when searching.
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.
Set stop height	In the **Stop height** text box, type the offset value from the work area where the robot should stop searching.
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.

Proceed with Check reachability

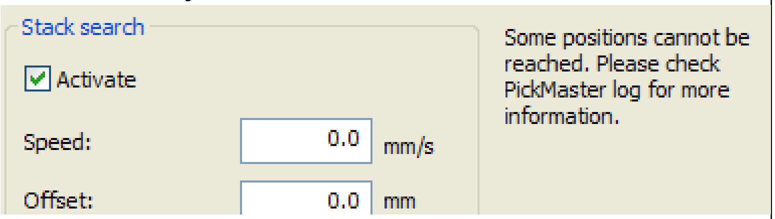
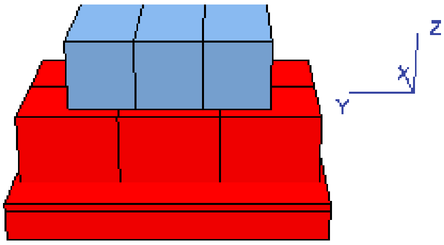
Click **Check reachability** to verify if a all positions can be reached by the robot. All robot positions are checked for each layer and each operation, including safe positions, approach positions, pick/place positions, and stack search specific positions. Imported tune data can affect the result. The check reachability function is implemented for **Pallet pattern operation set** and **Format operation set**. In current version, intermediate positions cannot be checked.

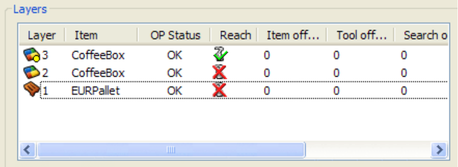
Continues on next page

The following must be fulfilled to use the reachability check:

- PickMaster is connected to the robot controller.
- Work object data is configured on controller and in PickMaster Line.
- Tool data (and search tool data if stack search is activated) is configured on controller and in PickMaster Line.
- The OP Status (Operation status) for all layer are approved.

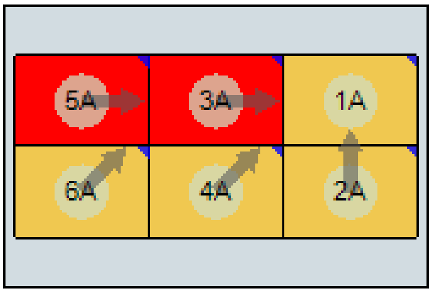
The result of the reachability check function is displayed in different ways.

<table>
<tr><th>Result</th><th>Description</th></tr>
<tr><td>Text message</td><td>A warning message will be displayed in the lower right corner of the window, <b>Reachability check results</b>.

Stack search
☑ Activate
Speed: 0.0 mm/s
Offset: 0.0 mm
Some positions cannot be reached. Please check PickMaster log for more information.
en0800000365</td></tr>
<tr><td>Graphic display in Pallet pattern</td><td>All layers that contain unreachable positions are drawn with red color. The current selected layer is still drawn with the selected layer color.


en0800000366</td></tr>
<tr><td>Graphic display in Layers, layer list</td><td>Layers containing unreachable positions are drawn with a red cross over the robot icon in Reach status. If all positions are reachable the robot icon is indicated with green.

Layers
<table>
<tr><th>Layer</th><th>Item</th><th>OP Status</th><th>Reach</th><th>Item off...</th><th>Tool off...</th><th>Search o</th></tr>
<tr><td>3</td><td>CoffeeBox</td><td>OK</td><td>✓</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>2</td><td>CoffeeBox</td><td>OK</td><td>✗</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>EURPallet</td><td>OK</td><td>✗</td><td>0</td><td>0</td><td>0</td></tr>
</table>
en0800000367</td></tr>
</table>

Continues on next page

Continued

Result	Description
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.  en0800000368

Related information

The Format Configuration on page 110.

The Pallet Pattern Configuration on page 115.

The Position Source Configuration on page 124.

*The **Tool Configuration** on page 81*.

The Operation Editor on page 143.

Handle the Project Configuration dialog box on page 99.

4.4.10 The Operation Editor

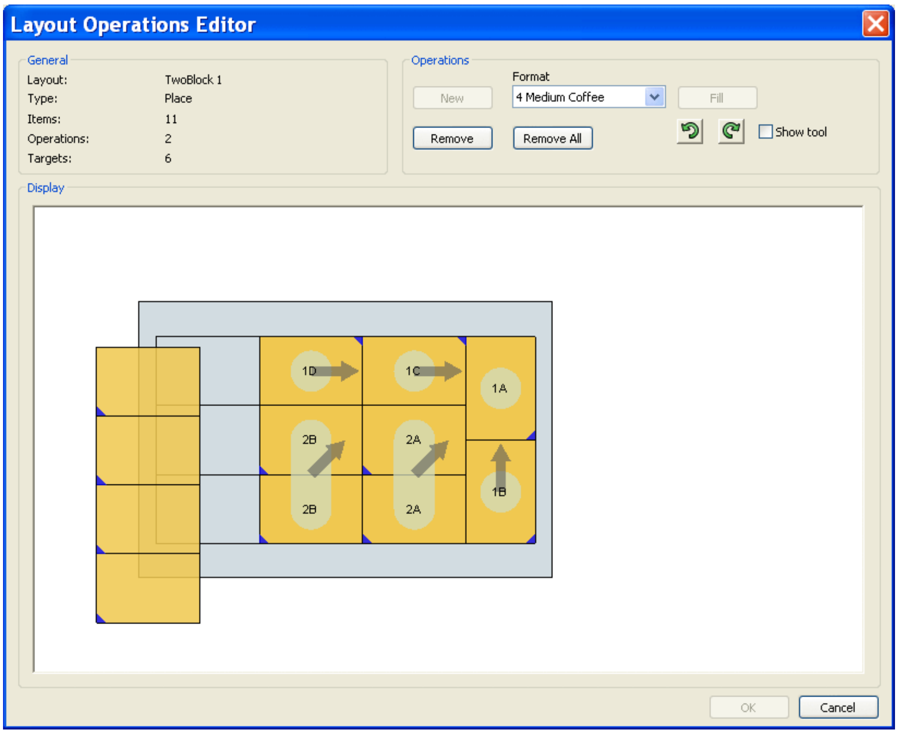
Overview

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

Start the Operation Editor

The **Operation Editor** is only accessible from a Pattern Operation Set. In the **Pallet Pattern Operation Set Configuration** window, select the layer to change and click **Edit**.

Illustration, Operation Editor



Layout Operations Editor

General

Layout: TwoBlock 1

Type: Place

Items: 11

Operations: 2

Targets: 6

Operations

Format: 4 Medium Coffee

□ Show tool

en0600003254

How to proceed

This section describes how to proceed with the operation editor.

A layer normally requires several operations, each using a specific format. The order of the operations is important since this is the order the robot will use to access the formats.

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.

Continues on next page

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 format to use for the operation is shown in the **Display** area.
Select format	Select the format to use for the new operation in the **Format** list.
Define format location	Move and rotate the format using the mouse. To define that an item in the format should be placed at a specific location in the layer, make the areas overlap and double-click on the intersection. To define if an item should be placed separately or together with its neighbor, right-click and select **Group item** or **Separate item**.
Auto generate operations	Click **Fill** to auto generate the operations for the rest of the layer.
Show the tool	Select the **Show tool** check box to show the location of the tool while dragging a format in the display area.
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 format and change the access directions.

Note

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

Related information

The Operation Set Configuration on page 128.

4.4.11 The Flow Configuration

Overview

A flow defines which slave work areas that must be used to supply the formats and products needed by a master work area. The selected slaves must be able to supply all formats used by the jobs defined on the master work area.

Several slaves may supply the same format (redundant supply).

Several flows can be defined in order to run jobs in parallel on different master work areas.

One slave work area may be shared among multiple flows.

In runtime execution, the sequence of format requests to the different slaves in the palletizing system is affected by the selection of master jobs to run and their operation sequence. It is also dependent on redundant formats, multiple flows, flow priorities, shared slaves, infeed speeds, robot performance, error handling, and so on.

Limitations

A flow must be complete. All formats used by the master opeation sets/jobs must be served by any slave before the settings are accepted.

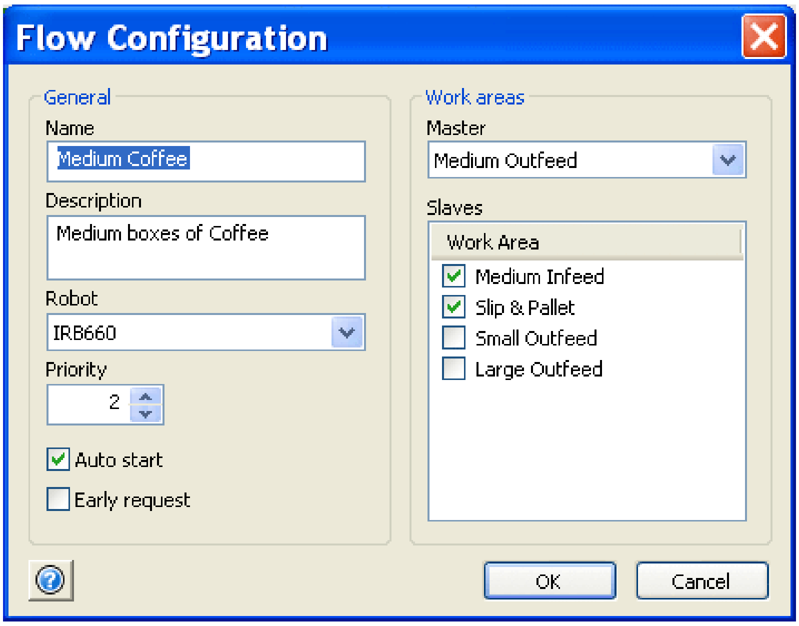
Start the Robot Flow Configuration

In the project panel

- Double-click the flow to edit, or
- Right-click the flow to edit and click **Edit**

Continues on next page

Illustration, Robot Flow Configuration



General

Name: Medium Coffee

Description: Medium boxes of Coffee

Robot: IRB660

Priority: 2

☑ Auto start

□ Early request

Work areas

Master: Medium Outfeed

Slaves

Work Area
☑ Medium Infeed
☑ Slip & Pallet
□ Small Outfeed
□ Large Outfeed

xx0700000023

General	Specify the name of the flow, which robot to use, and the priority of the flow. Define if the flow will start automatically. For detailed description, see *The General part on page 146*.
Work areas	Specify which work area that will be used in the flow.

The General part

Part	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.
Description	A brief description of the flow.
Robot	Specify which robot the work area is related to. To change the robot, select the desired robot from the **Robot** drop-down combo box.
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** if the flow should start when the project starts.

Continues on next page

Part	Description
Early request	Select **Early request** if the next request for new products to the slave work area (used as a slave) is made before the operation is executed on the master work area (used as a master). If this option is not selected, the next request for new products to the slave work areas is made after the operation is executed on the master work area. ℹ **Note** In practice, when palletizing and the outfeeder work area 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 work area 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.

The Work areas part

Part	Description
Master	Select the work area on which palletizing jobs are started. Often the work area where the pallet pattern is built is used as a master. A work area used as a master in a flow cannot be involved in any other flows.
Slaves	Select the work areas used as slaves (slave work areas). These are the work areas that will serve the master work area with products. The slave work areas are often pallet stacks, slipsheet stacks, and conveyor feeders. Slave work areas can be shared by multiple flows.

4.4.12 The Flow I/O settings editor

Overview

In the **Flow I/O Settings Editor** it is possible to configure:

- 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.
- A unique GO status signal for each flow. The status signal is used to monitor the runtime status of the flow.

Prerequisites

The controller must be online, otherwise only free default status signals can be selected, for example *pmFlow1_goStatus*, *pmFlow2_goStatus*.

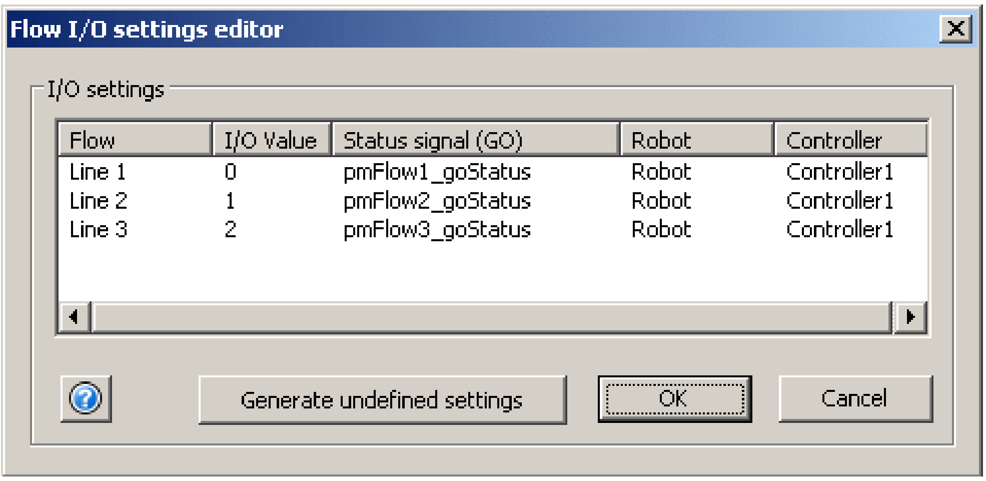
The status signals must have a minimum bit length of three to represent the flow states.

Starting the Flow I/O settings editor

	Action
1	In the project panel, right-click the **Product flows** tree symbol.
2	Select **Edit I/O settings**.
3	To edit values, click on a value and select from the drop-down list.

Illustration, Flow I/O settings editor

The list displays all project flows, their assigned I/O values and status signals. If the I/O value is undefined for a flow (that is not configured), the value will be -1 and the status signal field will be empty.



Flow	I/O Value	Status signal (GO)	Robot	Controller
Line 1	0	pmFlow1_goStatus	Robot	Controller1
Line 2	1	pmFlow2_goStatus	Robot	Controller1
Line 3	2	pmFlow3_goStatus	Robot	Controller1

en0800000374

Generate undefined settings	Generates default I/O settings for each undefined I/O value (that is, value is -1) or status signal (that is, value is empty). Lowest possible free I/O value will be set and default status signals.

4.4.13 The Message Settings

Overview

The **Message Settings** window is used for configuration of project-related messages and to import/export messages from/to the PickMaster Library.

In the **Message Settings** window you can define messages and connect them to values. These values are read from the **Messages (GI)** signal at the same time as there is a pulse on **Trigger (DI)** signal. These signals are configured in the **Controller properties** window in the PickMaster Line. In case of an event or an error that needs more information from the operator, the **Messages (GI)** signal is set to a value and the **Trigger (DI)** signal is pulsed. The message belonging to the value will appear on the FlexPendant, in the event log. This functionality can only be used if the event signals **Trigger (DI)** and **Messages (GI)** have previously been set up in the **Controller properties** window. For details about how to set up event signals, see *The Controller Properties window on page 66*. For information about how to view the messages, see *Viewing messages on page 220*.

Note

For each message the only valid values are between *1* and *2*$^{\text{number of bits}}$*-1*. For example, if the message signal consists of 7 bits the maximum value for this signal is 127 (2^7-1). The number of bits are defined by the **Messages (GI**) signal.

Note

The **Trigger (DI)** signal triggers both messages and events to the PickMaster process. If no message should be generated when triggering an event, the **Message (GI)** signal should be set to value 0.

Note

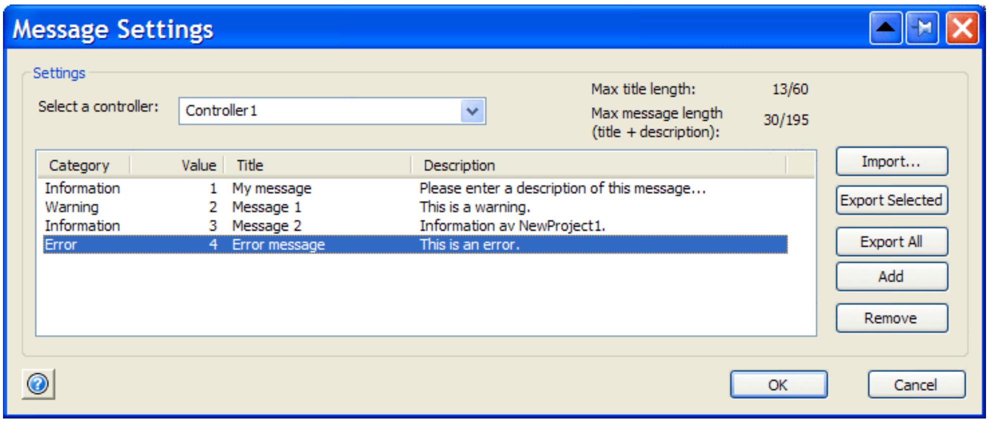
The message is only used for sending information from a superior system to the user. There is no relationship between a **Messages (GI)** signal and the **Error Source (GI)** signal, other than that they are triggered from the same **Trigger (DI)** signal.

Start the Message Settings

	Action
1	Open the Project view.
2	In the workspace, right-click and select **Message Settings**. Or On the **Project** menu, click **Message Settings**.
3	The **Message Settings** window appears and you can configure messages for the project.

Continues on next page

Illustration, Message Settings



Message Settings

Settings

Select a controller: Controller1

Max title length: 13/60

Max message length (title + description): 30/195

Category	Value	Title	Description
Information	1	My message	Please enter a description of this message...
Warning	2	Message 1	This is a warning.
Information	3	Message 2	Information av NewProject1.
Error	4	Error message	This is an error.

xx0700000447

Select a controller	Select a controller that the project should connect to.
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 **Add** to add new message. If the controller is connected, the first available value will be given to the new message.
Remove	Click **Remove** to remove 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.
Message	Click the **Message** column and type the message.

Note

If no value or value 0 is given to the message, the message will not be saved.

Note

To save change(s) to a message, you must click **OK** in the **Message Settings** window.

Continues on next page

Continued

Related information

The Import Messages on page 152.

The Controller Properties window on page 66.

4.4.14 The Import Messages

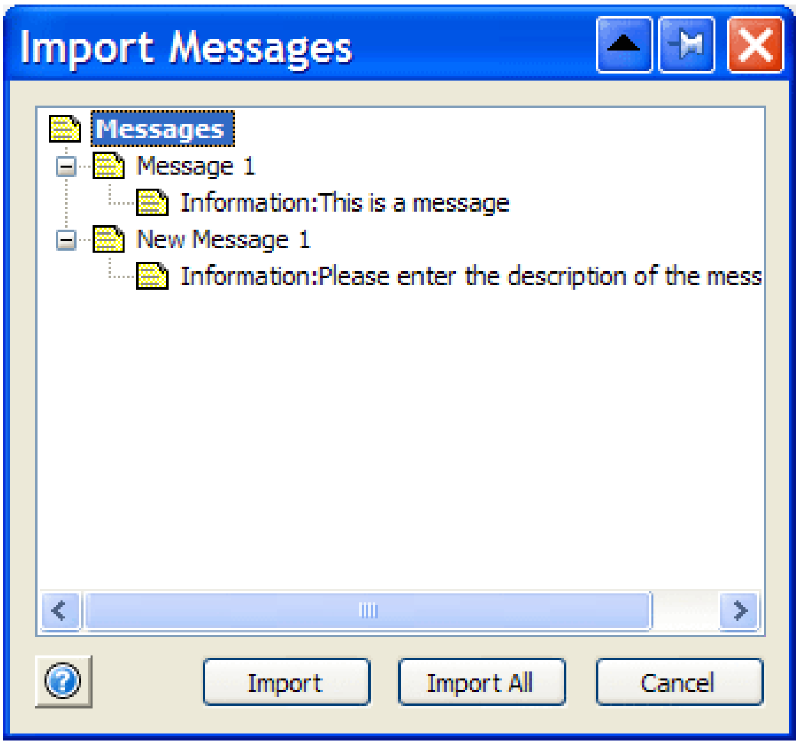
Overview

The **Import Messages** window is used to import messages from PickMaster library to a PickMaster project.

Start the Import Messages

	Action
1	In a PickMaster project, open **Message Settings** window. See *Start the **Message Settings** on page 149*.
2	In the **Message Settings** window, click **Import**.
3	The **Import Messages** window appears and you can import messages to the project.

Illustration, Import Messages


Import Messages
Messages
Message 1
Information:This is a message
New Message 1
Information:Please enter the description of the mess
Import
Import All
Cancel

xx0700000446

How to proceed

This section describes how to proceed with the **Import Messages** window.

To...	Do this
Import one message	In the **Messages** tree, select the required message to import and click **Import**. The selected message will be added to the message list in the **Message Settings** window. If the controller is connected, the first available value will be given to the message.

Continues on next page

To...	Do this
Import all messages	Click **Import All**. All messages will be added to the message list in the **Message Settings** window. If the controller is connected, an available value will be given to each message

Related information

The Message Settings on page 149.

4.4.15 The Robot Settings

Overview

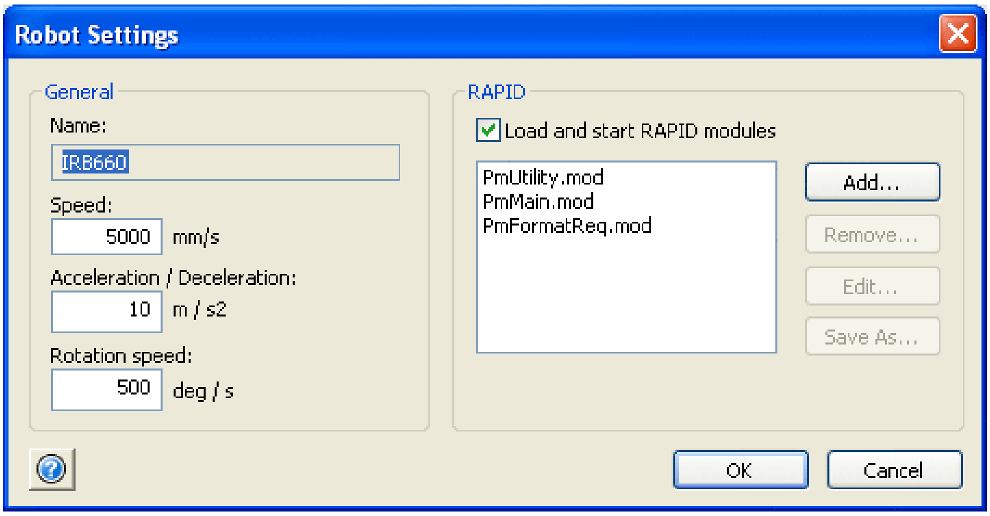
The **Robot Settings** window is used for modifying the motion settings of the robot and RAPID modules to be loaded to the controller.

Start the Robot Settings

	Action
1	Open the Project view.
2	In the workspace, right-click the robot symbol and click **Settings**.
3	The **Robot Settings** window appears and you can edit the robot settings.

Illustration, Robot Settings

The **Robot Settings** window consists of two main parts, a general and a RAPID part.



Robot Settings

General

Name: IRB660

Speed: 5000 mm/s

Acceleration / Deceleration: 10 m / s2

Rotation speed: 500 deg / s

RAPID

☑ Load and start RAPID modules

PmUtility.mod
PmMain.mod
PmFormatReq.mod

xx0600002918

General	Name and speed values for the robot.
RAPID	Commands to manage the RAPID modules.

The General part

Parts	Description
Name	The name of the robot.
Speed	The speed of the robot.
Acceleration/Deceleration	The acceleration/deceleration speed of the robot.
Rotation speed	The rotation speed of the robot.

Continues on next page

The RAPID part

Parts	Description
Load and start RAPID modules	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 Program** dialog box appears, and you can select which RAPID module to add. 3 In the **Robot Settings** dialog box, click **OK**.
Remove	To remove a RAPID module from the project, select a module and click **Remove**.
Edit	To edit a RAPID module, select a module and click **Edit**.
Save As	To save a module with a new name, select a module and click **Save As**.

Related information

Setting up the Project on page 167.

4.4.16 The Import Tune

Overview

The **Import Tune** window is used to import new values from the controller into the PickMaster project. When the values of the parameters have been tuned, and thus edited online, only the new values will remain on the controller.

For information about how to tune the parameters online, see *Runtime operation on page 211*.

Note

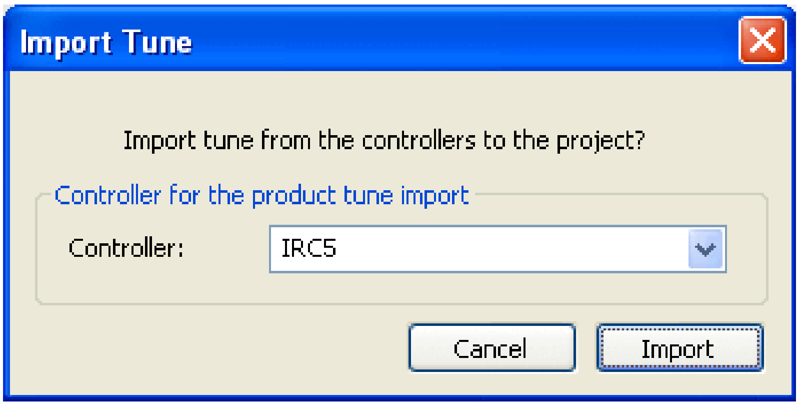
When you use the **Import Tune** function, the current tune values of the parameters in the PickMaster project will be overwritten by the tune values currently residing on the selected controller.

Start the Import Tune

	Action
1	Open the Project view.
2	In the workspace, right-click and click **Import Tune**. or In the **Project** menu, click **Import Tune**.
3	The **Import Tune** window appears and you can import the tune values from the controller.

Illustration, Import Tune

In the Import Tune window you select from which controller to import the tuned values for an item into the PickMaster project.



Import Tune

Import tune from the controllers to the project?

Controller for the product tune import

Controller: IRC5

xx0700000403

Note

The tuned values for a work area are automatically imported from the controller it belongs to.

Continues on next page

How to proceed

To...	Do this
Import tune values	In the **Controller** drop-down combo box, select the controller and click **Import**.

Note

If a project contains more than one controller, the tune values for the items can differ on the different controllers. Therefore you must select from which controller to import the item tune. The work area tune is imported from the controller, which it belongs to.

Related information

The Tune Limitations on page 158.

Tuning on page 232.

4.4.17 The Tune Limitations

Overview

The **Tune Limitations** window can be used to set the minimum and maximum limits for how much each parameter can be tuned online.

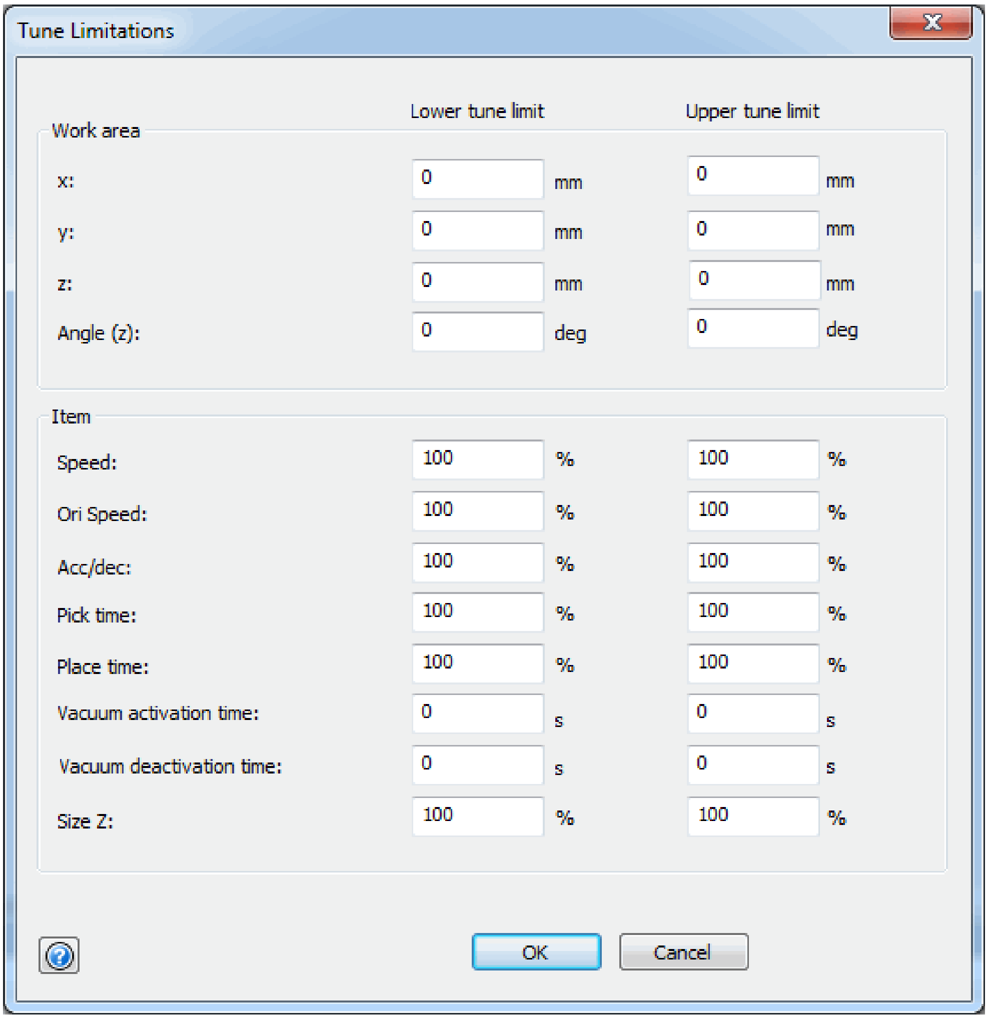
For more information about how to tune the parameters online, see *Runtime operation on page 211*.

Start the Tune Limitations

	Action
1	Open the Project view.
2	In the **Project** menu, click **Tune Limitations**.
3	The **Tune Limitations** window appears and you can specify the tune limits for different parameters.

Continues on next page

Illustration, Tune Limitations



	Lower tune limit		Upper tune limit	
Work area				
x:	0	mm	0	mm
y:	0	mm	0	mm
z:	0	mm	0	mm
Angle (z):	0	deg	0	deg
Item				
Speed:	100	%	100	%
Ori Speed:	100	%	100	%
Acc/dec:	100	%	100	%
Pick time:	100	%	100	%
Place time:	100	%	100	%
Vacuum activation time:	0	s	0	s
Vacuum deactivation time:	0	s	0	s
Size Z:	100	%	100	%

xx0700000412

Work area	Work area related parameters. For detailed descriptions of the different parts, see *The Work area part on page 159*.
Item	Item related parameters. For detailed descriptions of the different parts, see *The Item part on page 160*.

The Work area part

In the **Work area** part the limits are specified using the same unit as the parameter. The x, y, z, and z angle defines the displacement of the work area 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

- The default value of **Z angle** in the **Format Operation Set Configuration** window is 25 degrees. See *The Format Operation Set Configuration on page 129*.
- The **Lower tune limit** in the **Tune Limitations** window is set to 6 degrees. See *Illustration, Tune Limitations on page 159*. This means you can tune the z-angle

Continues on next page

to maximum 6 degrees less than the default value, that is, you can tune the **Angle (z)** to a minimum of 19 degrees.

- The **Upper tune limit** in the **Tune Limitations** window is set to 6 degrees. See *Illustration, Tune Limitations on page 159*. 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.

Parameters

The following table describes the parameters of the **Work area** part.

Part	Description
x	Displacement of the work area in x-direction relative the work object.
y	Displacement of the work area in y-direction relative the work object.
z	Displacement of the work area in z-direction relative the work object.
Angle (z)	Displacement angle of the work area in the z-direction.

The Item part

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

Example

- The default speed in the **Item Configuration** window is 1000 mm/s. This is referred to as 100%. See *The Item Configuration on page 105*.
- The **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. See *Illustration, Tune Limitations on page 159*.
- The **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. See *Illustration, Tune Limitations on page 159*.

Parameters

The following table describes the parameters of the **Item** part:

Part	Description
Speed	Specifies maximum speed for an item.
Ori speed	Specifies maximum orientation speed for an item.
Acc/Dec	Specifies maximum acceleration/deceleration for an item.
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.

Related information

The Import Tune on page 156.

5 Configuring PickMaster

5.1 Introduction

Structure of this chapter

This chapter explains how to configure PickMaster. Step-by-step procedures describes the working procedures for the configuration.

5.2 Setting up a new application

Introduction

This section describes the recommended working procedure when you start creating a completely new application. The working procedure helps you understand the dependency between the different objects. Of course, it is possible to change or expand your application later on.

A good approach when creating a new application is to start with the basic functionality and when that works as expected, expand the application.

The line and the project

PickMaster consists of two different parts, the *line* and the *project*. See *The Line and Project concept on page 20*. The project uses a line as a base, which means that you must start to create a line. Once you have set up your line, you can create as many projects as you like on that line. There is no need to create a new line or modify the existing line unless you change the hardware.

Working procedure for setup

The following procedure describes how to do the complete setup:

	Action	See
1	Create a line.	*Setting up the Line on page 164*.
2	Create a project.	*Setting up the Project on page 167*.
3	Transfer the project to the controller.	*Transferring the project on page 171*.

Tip

The setup of a line and project is easier if the controller is available on the network. The setup will also be easier if all signals, work objects, and tool data are set up correctly on the controller.

5.3 Preparing your controller for PickMaster

Introduction

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.

Prerequisites

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 FlexPendant.	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 work area. 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.	*The **Tool Configuration** on page 81* for information about the signals. *Operating manual - RobotStudio*. *Operating manual - IRC5 with FlexPendant*.
5	Add work area related signals. Use RobotStudio or the FlexPendant.	*The Work Area Configuration on page 73* for information about the signals. *Operating manual - RobotStudio*. *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 Properties window on page 66*. *Operating manual - RobotStudio*. *Operating manual - IRC5 with FlexPendant*.

Related information

*The **Tool Configuration** on page 81*.

The Work Area Configuration on page 73.

The Controller Properties window on page 66.

5.4 Setting up the Line

Introduction

The line describes the hardware and the electrical wiring of the actual production site. The controllers, with their robots, take the central place in the line. All other line components are created from the robot they belong to.

This section includes:

- The working procedure for setting up the line.
- Detailed procedures of each step in setting up the line.

Setting up the line

	Action	See
1	Set up the controller and the robot.	*Setting up a controller and robot on page 164.*
2	Set up the work areas.	*Setting up a work area on page 164.*
3	Set up the tool.	*Setting up a tool on page 165.*
4	Set up the event settings, if needed.	*Setting up an event setting on page 166.*

Setting up a controller and robot

Use this procedure to set up the robot controller.

	Action	See
1	Select which controller to use.	*Add a controller to the line on page 58.* *Select which robot to use on page 64.*
2	Configure the controller and select which robot to use.	*The IRC5 Configuration on page 63.*

Note

If the controller you intend to use is unavailable, you have the following two choices.

- If there is another controller available with the same configuration, you can use that controller to set up your project.
- If there is no appropriate controller available, you must create a template controller.

In either case you must change to the desired controller before transferring the project.

Setting up a work area

The work area is PickMaster's representation of the pick and place areas, and it connects the work object with the signals needed to control the picking and placing. One work area is needed for each infeeder, outfeeder, pallet stack and slip sheet stack.

Continues on next page

Continued

Use this procedure to set up the work area.

	Action	See
1	Define the work object on the controller.	*Preparing your controller for PickMaster on page 163*.
2	Set up the signals on the controller.	*Preparing your controller for PickMaster on page 163*.
3	Select the work object to use.	*The Work Area Configuration on page 73*.
4	Select the signals.	*The Work Area Configuration on page 73*.

Note

If the controller is available, you can choose between the possible signals and work objects. If the controller is unavailable, you have to type in the correct signal or work object name.

Tip

The work objects must be defined and loaded in RAPID to be available. You can define a work object by using the FlexPendant. See *Operating manual - IRC5 with FlexPendant*, section *Creating a work object*.

It is recommended you declare all needed work objects as global in a system module on the controller, for example *pmrcUser* module. See *Public system module pmrcUser on page 196*.

Setting up a tool

All robots have a default tool, which needs to be modified in most applications.

Use this procedure to set up the tool.

	Action	See
1	Define the tool data on the controller.	*Technical reference manual - RAPID overview*
2	Select tool data.	*The* ***Tool Configuration*** *on page 81*. *The tabbed pages on page 83*.
3	Set up activators.	*Activators tab on page 87*.
4	Set up zones.	*Zones tab on page 87*.
5	Set up signals.	*Signals tab on page 88*
6	Set up search parameters.	*Search tab on page 89*
7	Test the tool.	*Test tab on page 89*.

Tip

The tool data must be defined and active in RAPID to be available. It is recommended you declare all needed tool data as global in a system module on the controller, for example *pmrcUser* module. See *Public system module pmrcUser on page 196*.

Continues on next page

Tip

If you intend to use the same tool for many robots, you only need to define it once. You export it to the PickMaster Library and then you can import it to other robots. See *The **Tool Configuration** on page 81*.

Setting up an event setting

Use this procedure describes to define which errors and/or messages will be reported to the PickMaster application from an external equipment, for example a PLC.

	Action	See
1	Set up the signals on the controller.	*Operating manual - RobotStudio*. *Operating manual - IRC5 with FlexPendant*.
2	Select the signals.	*The Controller Properties window on page 66*.
3	Associate sources to the bits in the **Error Source (GI)** signal.	*The Controller Properties window on page 66*.
4	Define the messages for the events.	*The Message Settings on page 149*.

Note

If the controller is available, you can choose between the possible signals and work objects. If the controller is unavailable, you have to type the correct signal or work object name.

Related information

The Line view on page 57.

The IRC5 Configuration on page 63.

Terminology on page 24.

Public system module pmrcUser on page 196.

The Controller Properties window on page 66.

The Message Settings on page 149.

5.5 Setting up the Project

Introduction

The project describes the products to be picked and how they are handled in your application.

This section includes:

- The working procedure for setting up the project.
- Descriptions of project components and detailed procedures of steps in setting up the project.

Setting up the project

Use this procedure to set up the project.

	Action	See
1	Create shapes.	*Start the Shape Configuration on page 102*. *How to proceed on page 103*.
2	Create pallets.	*Project panel components on page 168*.
3	Create products.	*Project panel components on page 168*.
4	Create slip sheets (if needed).	*Project panel components on page 168*.
5	Create formats.	*Project panel components on page 168*.
6	Create pallet patterns.	*Setting up a pallet pattern on page 168*.
7	Create position sources.	*Setting up a position source on page 169*.
8	Create flows.	*Setting up a flow on page 169*.
9	Create messages.	*The Message Settings on page 149*.
10	Add and edit RAPID modules.	*Adding and editing RAPID modules on page 169*.
11	Create tuning limitations.	*The Tune Limitations on page 158*.

Continues on next page

Project panel components

The following table describes the project panel components:

Shape	The shape describes the geometrical size of the items handled in PickMaster. A shape can be used for many items that have the same size but there must exist a shape with the right size for each product, pallet, and slip sheet you want to create. **Note** The form of the shape decides if the shape should be used for products, pallets, or slip sheets. **Note** A palletizing project needs at least one pallet shape even if the application does not use pallets. The area of the pallet is needed to create the layouts. **Tip** If you intend to use the same shape in different projects, you can save it to the PickMaster Library. See *The PickMaster Library on page 51*.
Pallet	To create a pallet there must be at least one shape of the form *Pallet*.
Product	The product describes the product that should be palletized.
Slip sheet	It is not necessary to create a slip sheet to have a complete palletizing project. The slip sheet only needs to be included if your application uses slip sheets. To create a slip sheet there must be at least one shape of the form *Sheet*.
Format	The format describes the relation between the tool and the items. You need to create at least one format for each product, pallet or slip sheet that you want the robot to handle in your application. You may need more than one format for your products. There must be a combination of format such as the total number of boxes in the formats and the number of boxes in each layer evens out.
Pallet pattern	The pallet pattern describes how the products should be stacked on the pallet, and consists of a number of layers which all have a layout.
Position sources	The position source is the central part of the project and it • Creates the positions for the robot and consequently defines the actual picking and placing of products. • Can use both formats and pallet patterns to create the positions.
Flow	A flow describes what operations should be performed. There must be at least one flow defined in a project.

See also *Terminology on page 24*.

Setting up a pallet pattern

Use this procedure to set up the pallet pattern.

	Action	See
1	Select the pallet, product, and slip sheet shapes to use. Proceed with defining shapes on page 97.	*Define the size of the shape on page 103*
2	Select the layouts to use.	*Proceed with defining layouts on page 117*.
3	Edit the layouts, if needed.	*How to proceed on page 122*.

Continues on next page

	Action	See
4	Add and order the selected layouts.	*Proceed with defining pallet pattern layers on page 119*.

💡 **Tip**

You can save a layout in the PickMaster Library and use it in other pallet patterns and projects.

Setting up a position source

Use this procedure to set up the position source.

	Action	See
1	In the **Position Source Configuration** dialog box, select if you will use format or pallet pattern.	*How to proceed on page 126*.
2	Configure the operation set.	*The Operation Set Configuration on page 128*.

Setting up a flow

Use this procedure to set up the flow.

	Action	See
1	In the **Flow Configuration** dialog box, select the master work area.	*The Flow Configuration on page 145*
2	In the **Slaves - Work Area** check boxes, select the work areas that will be used by the master work area.	*The Flow Configuration on page 145*
3	In the **Name** text box, type the name, and in the **Description** text box, type a short description of the flow. Both name and description will appear on the PickMaster FlexPendant interface during run time.	*Opening a project on page 214*
4	In the **Robot** combo box, select the robot that should execute the flow.	*The Flow Configuration on page 145*
5	In the **Priority** combo box, select/type the flow priority. This action is optional.	*The Flow Configuration on page 145*
6	Select the **Auto start** check box if you want to use auto start.	*The Flow Configuration on page 145*
7	Select the **Early request** check box if you want to use early request.	*The Flow Configuration on page 145*

Adding and editing RAPID modules

To be able to run the project, RAPID modules are needed. There are template RAPID modules that can be used, which allow you to customize your project.

Use this procedure to add RAPID modules.

	Action	See
1	Import the RAPID template to use.	*The RAPID part on page 155*.

Continues on next page

Continued

Related information

The Shape Configuration on page 102.

The Layout Editor on page 120.

Terminology on page 24.

5.6 Transferring the project

Introduction

When your project is ready you need to transfer it to the controller.

Prerequisites

To transfer the project, the selected controller must be available on the network.

Before transferring the project, it is checked for errors and to ensure that the configuration is valid. Error messages in the PickMaster log provide detailed information about any errors.

Transferring a project

To transfer a project to the controller:

1. On the **Project** menu, click **Transfer**, or in the toolbar, click **Transfer Project**.

Note

The project verification of line and project when transferred to the controller does not include verification of RAPID variables, work area, and/or tool data that are typed manually. If entered manually, the variables must be declared as global persistent variables in RAPID.

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 the **Import Tune** function before the downloading.

5.7 Starting production

Introduction

To run a PickMaster project you can either use the PickMaster FlexPendant interface or the I/O interface.

Starting production with FlexPendant

Use this procedure to start a PickMaster project from the FlexPendant.

	Action	See
1	Start the PickMaster FlexPendant interface.	
2	Select the project you want to run.	*Opening a project on page 214*.
3	Start the project.	*Starting a project on page 216*.
4	Start flows.	*Starting a specific flow on page 217* and *Starting and stopping all flows on page 219*.
5	Start jobs	*Starting a specific job on page 217*

Starting production with I/O interface

Use this procedure to start a PickMaster project from the I/O interface.

	Action	See
1	Select and start the project you want to run.	*Starting a project on page 250*.
2	Start flows.	*Starting or restarting a flow on page 252* .
3	Start jobs	*Target generation trigger signal (DI) on page 240* and *Target generation product selection (GI) on page 240*

Note

You can also combine the FlexPendant interface and the I/O interface to start production, for example, starting projects and flows from the FlexPendant and starting palletizing jobs using the I/O interface.

Flows can also be configured as auto started. For more information, see *Setting up the Project on page 167*.

6 RAPID program

6.1 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.

6.2 Overview

6.2.1 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 328*.

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 341*.

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 313*.

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.

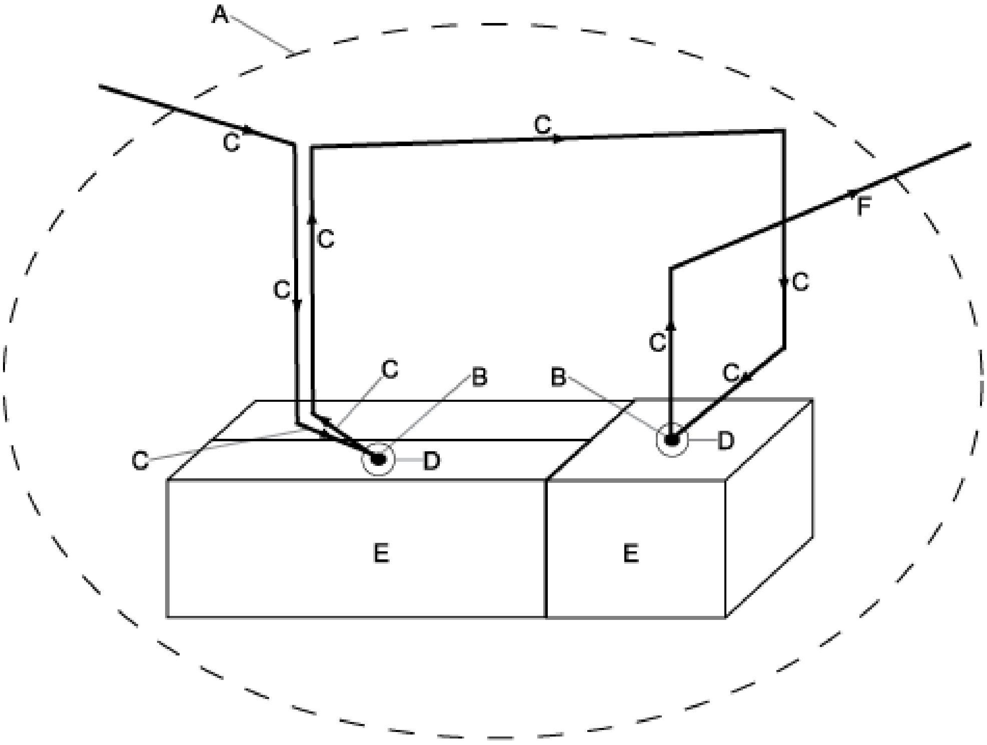
Product

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

Continues on next page

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

6.2.2 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:

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

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

- *pmrcUser*, which is a system module that contains tool and work object declarations and traps for checking I/O values.
- *pmrcSys*, which is a system module that contains open non-view procedures mainly used to set all modal data used in the moves.
- *pmrcBase*, which is a system module that contains encrypted non-view procedures and variables used in the process.

6.3 Program module templates

6.3.1 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:

- `Main`
- `OperateSequence`
- `Operate`

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 177.

PmWaitProjStart - Wait for any active project on page 292.

PmGetWaByWobj - Get a reference to a work area using a work object data on page 276.

Procedure `OperateSequence`

Usage

This routine performs one cycle sequence.

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.

Continues on next page

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 178.

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
```

Related information

Procedure `PmDoAction` on page 197

Procedure `SetLastPos` on page 182.

Procedure `MoveInterMid` on page 183.

pm_wadescr - PickMaster work area reference on page 345.

Continues on next page

pm_operationdata - PickMaster operation data on page 328.

pm_targetdata - PickMaster target data on page 341.

pm_actiondata - PickMaster action data on page 313.

PmGetOperation - Get operation from a work area on page 272.

PmCalcArmConf - Calculates the arm configuration on page 264.

PmGetTarget - Get target on page 304.

PmGetTgtAction - Get target action on page 306.

PmSearchAdjust - Adjust number of remaining layers on page 279

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 345.

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;
```

Related information

pm_wadescr - PickMaster work area reference on page 345.

PmCalcArmConf - Calculates the arm configuration on page 264.

6.3.2 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.

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 292.

Continues on next page

PmGetWaByWobj - Get a reference to a work area using a work object data on page 276.

PmSetLastWa - Set last used work area on page 281.

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.

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 313.

pm_targetdata - PickMaster target data on page 341.

pm_wadescr - PickMaster work area reference on page 345.

PmSetLastWa - Set last used work area on page 281.

Continues on next page

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.

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

Continued

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+SafetyOffsetZ;
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
```

Continues on next page

```
      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 313.

pm_targetdata - PickMaster target data on page 341.

pm_wadescr - PickMaster work area reference on page 345.

PmGetLastWa - Get last used work area on page 271.

PmGetPathHeight - Get a safe path height for an intermediate movement on page 299.

PmCalcArmConf - Calculates the arm configuration on page 264.

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.

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*.

6.3.3 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:

- Procedure `Main`
- Trap `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:

- Setting up a TRAP that supervises the stop project signal. The RAPID execution is interrupted and continues from Main.
- Waiting for project start signal.
- Reading project selection signal. The mapping between project and its signal value must have been transferred.
- Starting selected project.
- Setting current project signal to the value of the selected project.
- Loading the modules in the project. The modules are only loaded if it is specified in the configuration and if they are not already loaded.
- Executing 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.

Continues on next page

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;
  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
```

Continues on next page

Continued

```
    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;
      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 292.

PmStartProj - Start a PickMaster project on page 288.

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:

- Stopping robot movement.
- Clearing the robot path.
- Resetting stop move state.
- Continuing execution from main.

Continues on next page

Program code

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

Related information

Technical reference manual - RAPID kernel.

6.3.4 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:

- Connecting all alias signals with the configured signals.
- Waiting for a project to start.
- Connecting traps to project stop, flow start, and flow stop traps.
- Waiting for the project to stop.
- Disconnecting all traps.

Continues on next page

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;
  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.

Continues on next page

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;
  ! 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
```

Continues on next page

```
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.

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
```

Continues on next page

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;
    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 269.

PmStartFlow - Starts a specific flow on page 286.

PmStopFlow - Stop a specific flow on page 289.

PmStopProj - Stop current project on page 291.

6.4 System modules

6.4.1 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:

- `pmrcUser` (open)
- `pmrcSys` (open)
- `pmrcBase` (encrypted).

6.4.2 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 the tool event DI signal is not set at the requested robot position. The RAPID execution and robot movement is interrupted until the specified signal value is set. The trap is prepared with some debug information that can be used if there are problems with signal settings. Just uncomment the instruction `TPWrite`.

Program code

```
TRAP TrapDIToolEvents
  VAR num loopCounter:=0;
  VAR bool found:=FALSE;
  VAR num waitMaxTime:=5;

  WHILE found=FALSE DO
    Incr loopCounter;
    IF INTNO=pm_IntNos{loopCounter} THEN
      ! Uncomment following line for information about signal and
           value
      ! TPWrite "Signal "+pm_SignalNames{loopCounter}+" is waiting
           for IO value "\Num:=pm_diValues{loopCounter};
      WaitDI pm_diSignals{loopCounter},pm_diValues{loopCounter}
           \MaxTime:=waitMaxTime;
      StartMove;
      found:=TRUE;
    ENDIF
  ENDWHILE
ERROR
  IF ERRNO=ERR_WAIT_MAXTIME THEN
    SkipWarn;
    waitMaxTime:=3600;
    PmErrorLog 2364, ERRSTR_TASK, pm_SignalNames{loopCounter},
         ValToStr(pm_diValues{loopCounter}), ERRSTR_CONTEXT,
         ValToStr(waitMaxTime)\ErrorHandler:=TRUE;
    RETRY;
  ENDIF
ENDTRAP
```

Trap `TrapGIToolEvents`

Usage

This trap works in the same way as `TrapDIToolEvents` but with GI signals.

6.4.3 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.

Continues on next page

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_DO:
        Incr NumOfTriggEvents;
        GetDataVal Event.SignalName,doSignal;
        IF Act.type = PM_TARGET_POS AND Tgt.StopPointData.type =
             2 THEN
          TriggEventTime := Event.time - Tgt.stopPointData.stoptime
               / 2;
          IF TriggEventTime >= 0 THEN
            TriggIO TriggArr{NumOfTriggEvents}, TriggEventTime,
                 \Time, \DOp:=doSignal,SetDvalue:=Event.DValue;
          ELSE
            TriggIO TriggArr{NumOfTriggEvents}, 0, \Time,
                 \DOp:=doSignal, SetDvalue:=Event.DValue
                 \DODelay:=-TriggEventTime;
          ENDIF
        ELSE
          TriggEquip TriggArr{i}, Event.Dist, Event.Time,
               \DOp:=doSignal, SetDvalue:=Event.DValue;
        ENDIF
        ...
      CASE PM_EVENT_WAIT_DI:
        GetDataVal Event.SignalName,diSignal;
        diValue:=Event.Value;
        bCheckDI:=TRUE;
      ...
    ENDTEST
  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
```

Continues on next page

```
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;
  ENDIF
  ...
  CASE 3:
    PmDoMove1 Act.Move, Act.UseConc, Act.RobTgt,
         Offs(Tgt.RobTgtPoint,Act.Offset.x,Act.Offset.y,Act.Offset.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
```

Continues on next page

Continued

```
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
```

Related information

pm_wadescr - PickMaster work area reference on page 345.

pm_targetdata - PickMaster target data on page 341.

pm_actiondata - PickMaster action data on page 313.

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.

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.

Continues on next page

Continued

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

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.

Continues on next page

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, 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, Zone, Inpos, Tool, WObj;
  ELSE
    PmDoMove2 Move, ToPoint, Speed \T1?T1 \T2?T2 \T3?T3 \T4?T4
         \T5?T5 \T6?T6, Zone, Inpos, Tool, WObj;
  ENDIF
ERROR
  RAISE;
ENDPROC
```

Related information

pm_movetype - PickMaster movement type on page 325.

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.

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

Continues on next page

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

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, 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, Zone \Inpos:=Inpos, Tool, WObj;
  ELSE
    PmDoMove3 Move \Conc?Conc, ToPoint, Speed \T1?T1 \T2?T2 \T3?T3
         \T4?T4 \T5?T5 \T6?T6, 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.

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.

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, 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, Zone \Inpos?Inpos, Tool \WObj:=WObj;
    CASE PM_TRIGG_JOINT:
      TriggJ \Conc?Conc, ToPoint, Speed, T1, \T2?T2 \T3?T3 \T4?T4
           \T5?T5 \T6?T6, 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 325.

Continues on next page

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.

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, 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.

Continues on next page

Continued

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:
      SearchL \Sup, diSearchSignal \Flanks, SearchPoint, ToPoint,
           Speed, Tool \WObj:=WObj;
    CASE PM_STOP:
      SearchL \Stop, diSearchSignal \Flanks, SearchPoint, ToPoint,
           Speed, Tool \WObj:=WObj;
    CASE PM_SSTOP:
      SearchL \SStop, diSearchSignal \Flanks, SearchPoint, ToPoint,
           Speed, Tool \WObj:=WObj;
    CASE PM_PSTOP:
      SearchL \PStop, 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
```

Continues on next page

Continued

```
ERROR
  TEST ERRNO
    CASE ERR_WHLSEARCH:
      SearchError:=TRUE;
      SearchPoint:=ToPoint;
      TRYNEXT;
    CASE PM_ERR_PALLET_REDUCED:
      IF SearchError = TRUE THEN
        ...
        ...
        RAISE ERR_WHLSEARCH;
      ELSE
        RAISE;
      ENDIF
    CASE PM_ERR_PALLET_EMPTY:
      RAISE;
    DEFAULT:
      RaiseToUser \BreakOff;
  ENDTEST
ENDPROC
```

Related information

pm_movetype - PickMaster movement type on page 325

pm_searchdata - PickMaster search data on page 335

pm_searchtype - PickMaster stack search type on page 336

pm_stoptype - PickMaster stop type on page 340

PmSearchAdjust - Adjust number of remaining layers on page 279

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

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;
...
```

7 Runtime operation

7.1 Introduction

Structure of this chapter

This chapter describes the runtime operating interface to PickMaster.

The runtime operating interface consists of three parts:

- PickMaster FlexPendant interface. A graphic operator's interface.
- PickMaster I/O interface. Used by a PLC.
- PickMaster 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 173*, and *RAPID reference information on page 261*.

Prerequisites

The runtime operating interface is only available with the option *Prepared for PickMaster*, with the sub-option *PickMaster 5*.

7.2 FlexPendant interface

7.2.1 Introduction to PickMaster FlexPendant interface

PickMaster FlexPendant interface

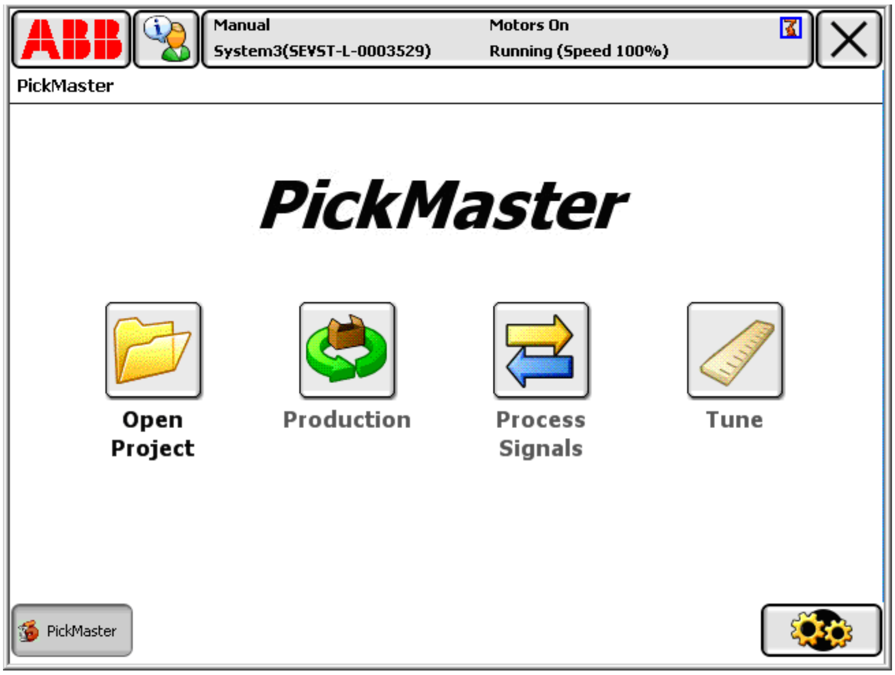
The PickMaster FlexPendant interface is available from the ABB menu on the FlexPendant. It is a graphical user interface designed to control and/or supervise the palletizing process.

The PickMaster FlexPendant interface covers the following four areas:

- Open Project
- Production
- Process Signals
- Tune

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 214*.

Production

Production is used by the operator to start and stop and monitor the palletizing process.

Continues on next page

See *Starting and stopping production on page 216*.

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 *Zones tab on page 87* and *Activators tab on page 87*.

See also *Process Signals on page 229*.

Tune

Tune is used to change the parameter values online while running the PickMaster application.

See *Tuning on page 232*.

7.2.2 Opening a project

Introduction

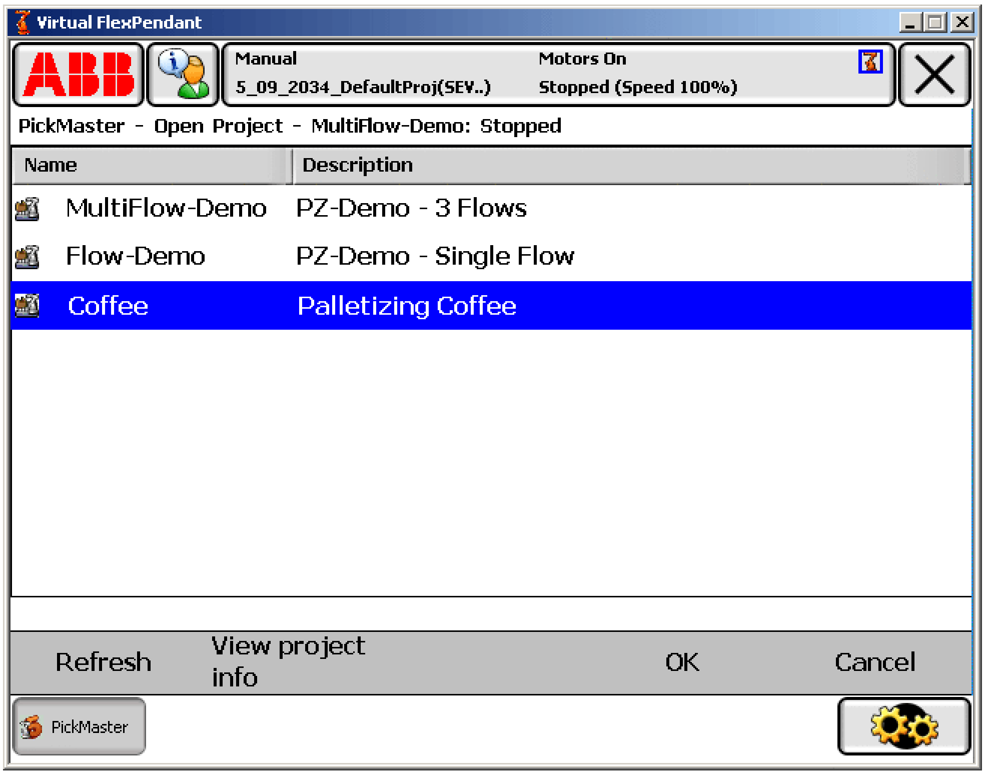
This section describes how to open a PickMaster project from the Open Project window. A list of all the available projects on the controller appears together with a description, if provided. To provide a description, see *Add a description to the project on page 98*.

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.


Virtual FlexPendant
Manual
5_09_2034_DefaultProj(SEV..)
Motors On
Stopped (Speed 100%)
PickMaster - Open Project - MultiFlow-Demo: Stopped

Name	Description
MultiFlow-Demo	PZ-Demo - 3 Flows
Flow-Demo	PZ-Demo - Single Flow
Coffee	Palletizing Coffee

xx0600002996

Continues on next page

Continued

Related information

Starting and stopping production on page 216.

Process Signals on page 229.

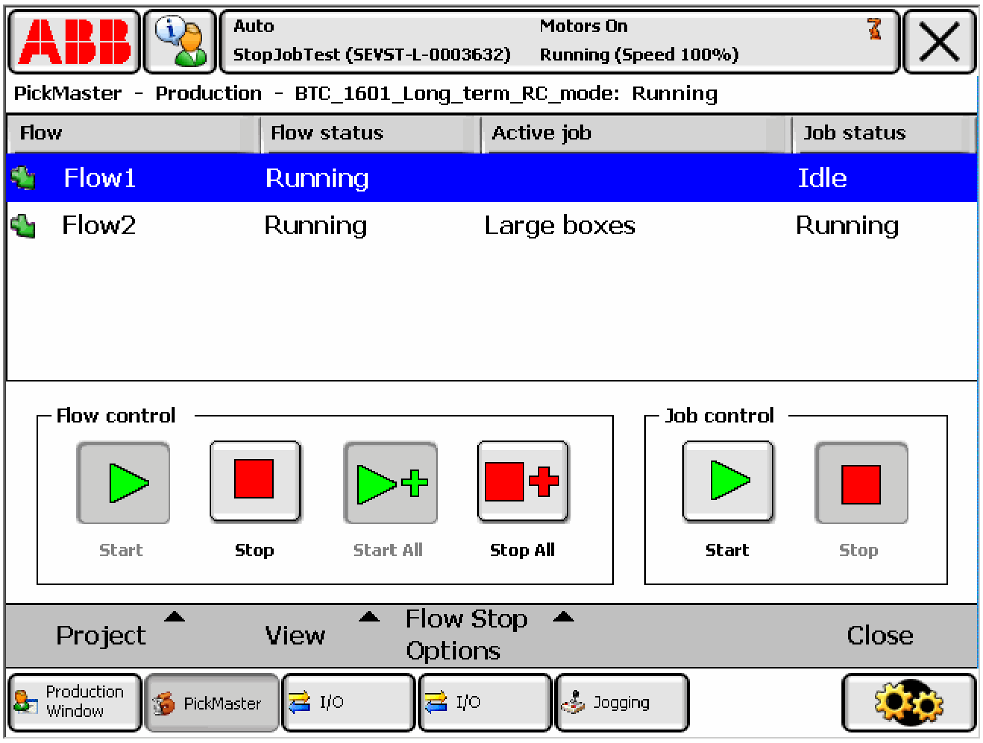
7.2.3 Starting and stopping production

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

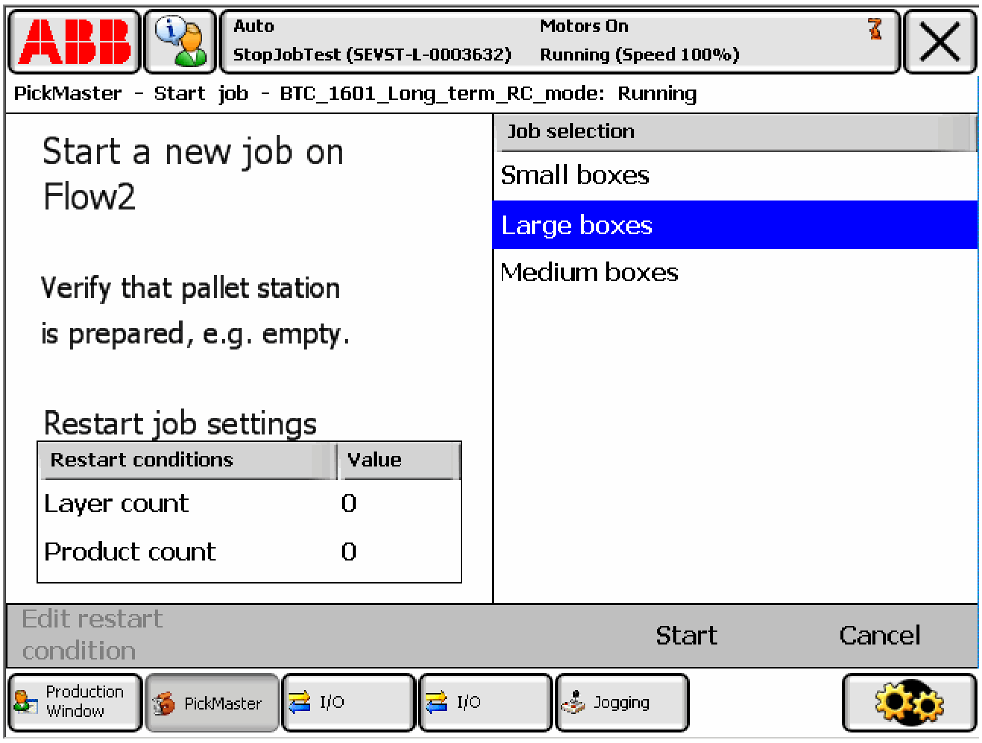
Note

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.

Continues on next page

The following illustration shows an example of the Start job window.



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**.

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.

Continues on next page

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

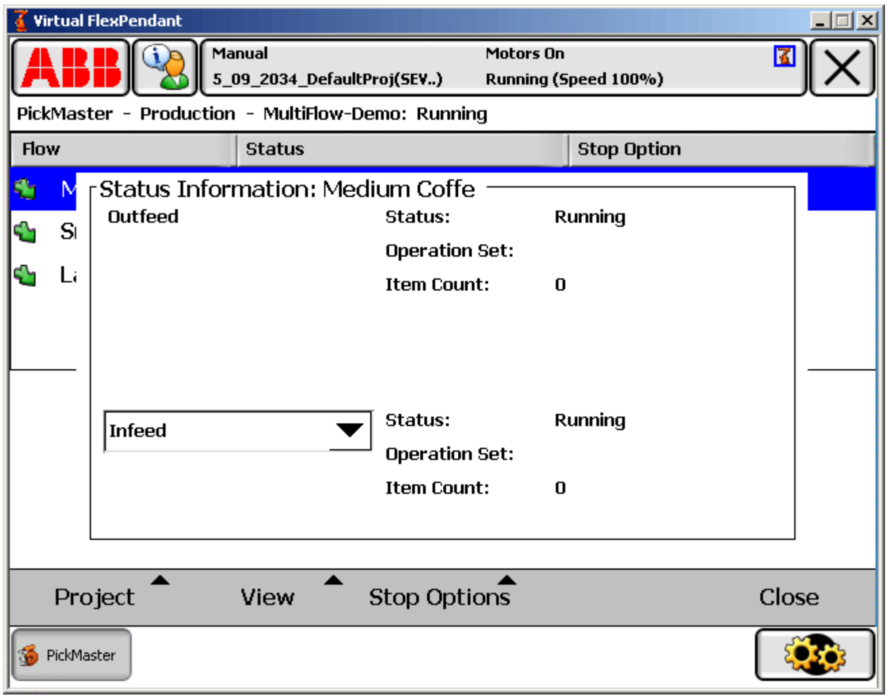
To...	Do this
Start all flows	Tap **Start All Flows**.
Stop all flows	Stop **All Flows**.

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

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.



Status Information: Medium Coffe

Outfeed

Status: Running

Operation Set:

Item Count: 0

Infeed

Status: Running

Operation Set:

Item Count: 0

xx0700000228

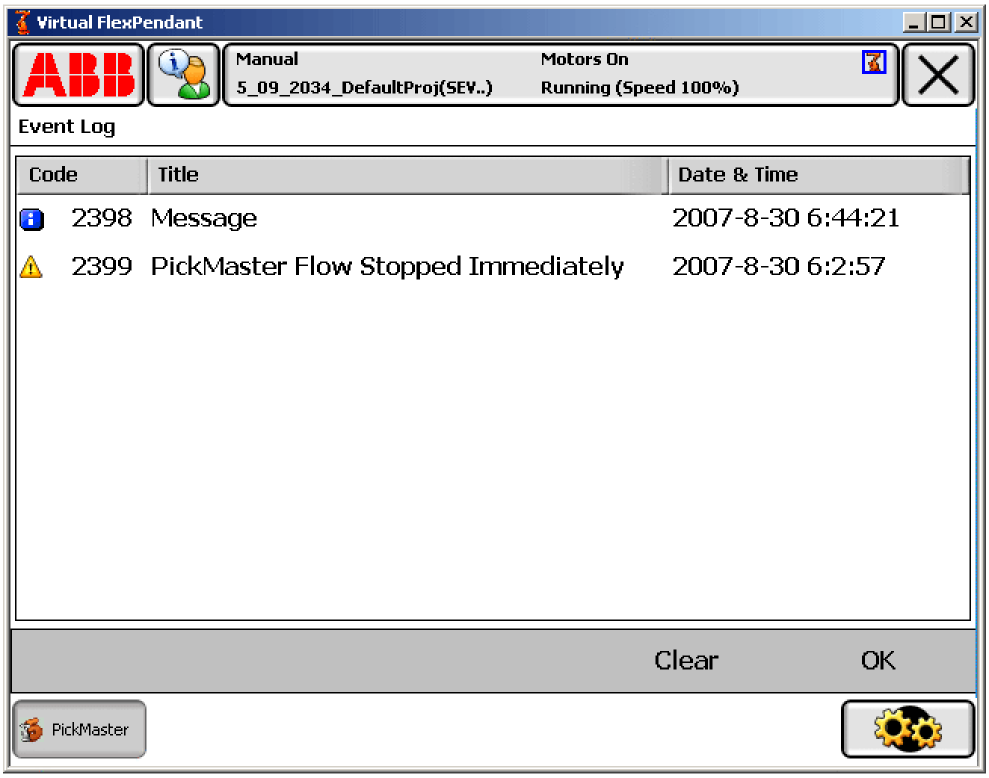
Work Area	The name of the work area as specified in the PickMaster software. For further details, see *The Work Area Configuration on page 73*.
Status	Provides status information about the work area, which can be **Running** or **Error**.
Operation Set	Specifies the last accessed operation set on the 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 *The Message Settings on page 149*.

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.

Continues on next page

Flow stop option	Description
Finish layer	The robot will continue palletizing until the current layer of the job is finished.
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:

- On the **Stop Options** menu, tap **Undo**.

Related information

Opening a project on page 214.

Process Signals on page 229.

The Work Area Configuration on page 73.

The Controller Properties window on page 66.

The Robot Settings on page 154.

The IRC5 Configuration on page 63.

7.2.4 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.

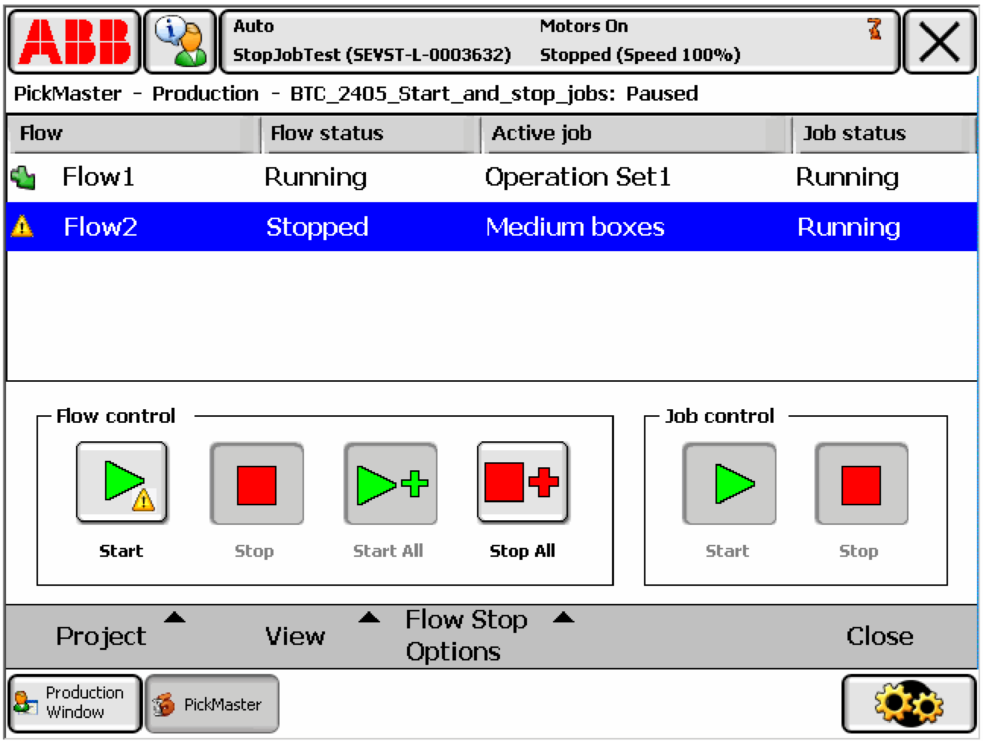
Recovering a specific flow

	Action	Note
1.	In the **Production** window, tap **Start Flow**.	See *Illustration, Production window with a flow to recover on page 224*.
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 225*.
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 226*.
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 227*

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

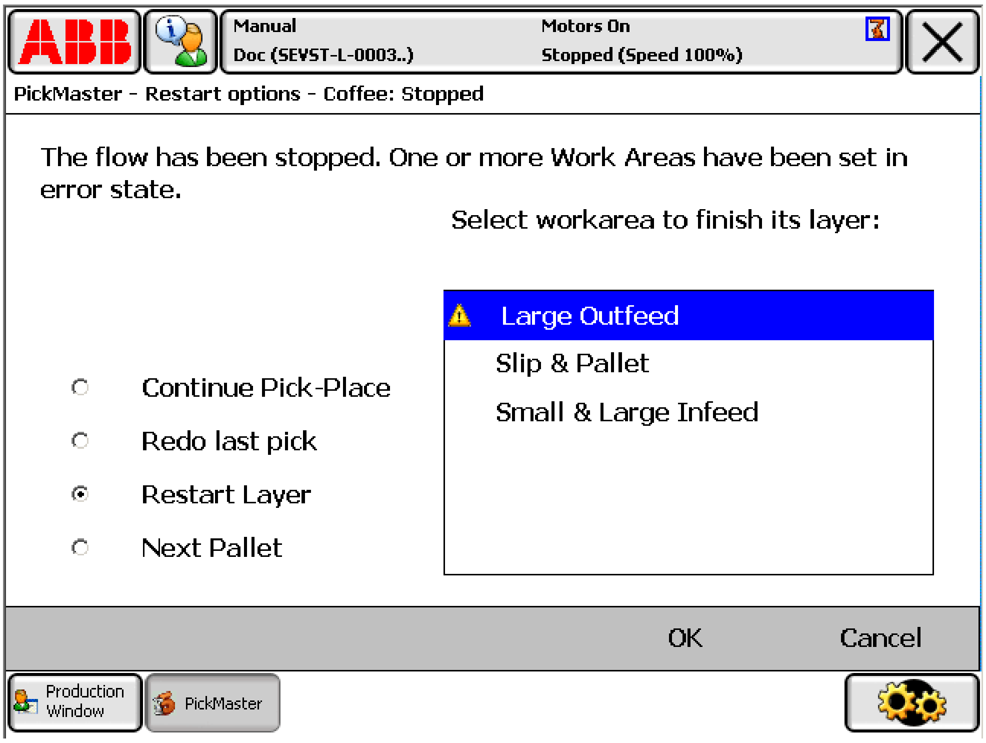
Flow	Flow status	Active job	Job status
Flow1	Running	Operation Set1	Running
⚠ Flow2	Stopped	Medium boxes	Running

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**.

Continues on next page

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

Continue Pick-Place	The flow will continue from where it was stopped. Before continuing with the working procedure, you need to verify that: • The reason for the error has been handled. • The 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: • The reason for the error has been handled. • The tool is empty. • The 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: • The reason for the error has been handled. • The tool is empty. • The correct format can be supplied in the work areas. • The layer that will be repeated is restored.

Continues on next page

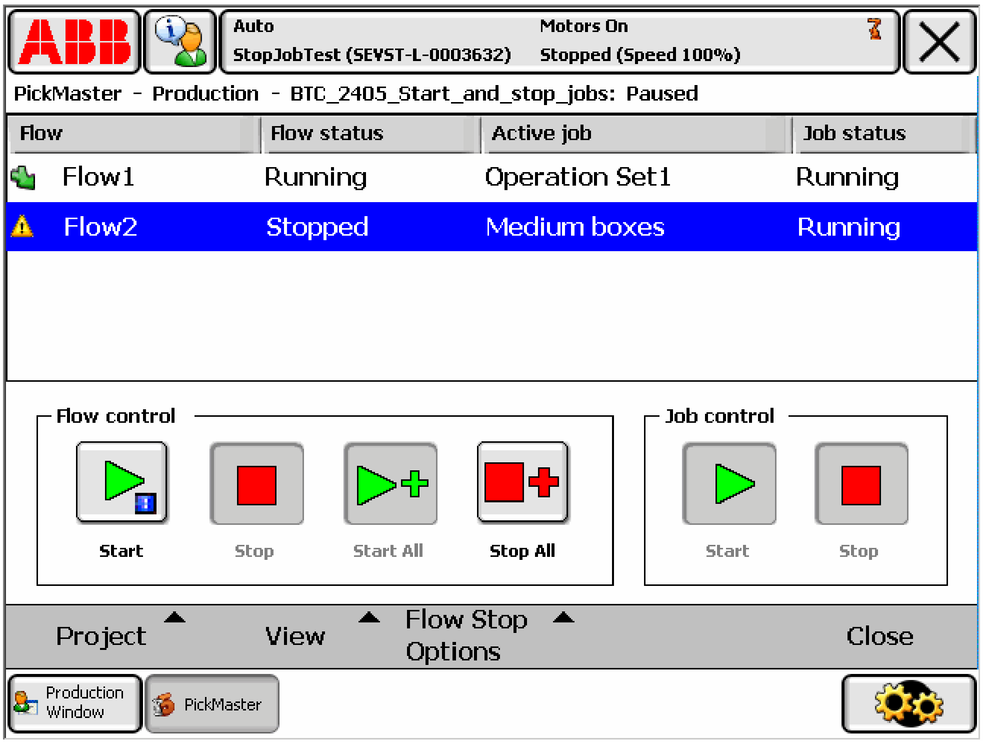
Next pallet	The work areas set in error state are marked with a warning symbol. Select the work area for the pallet you intend to restart. We recommended selecting the master work area. Before continuing with the working procedure, you need to verify that: • The reason for the error has been handled. • The tool is empty. • The 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.

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

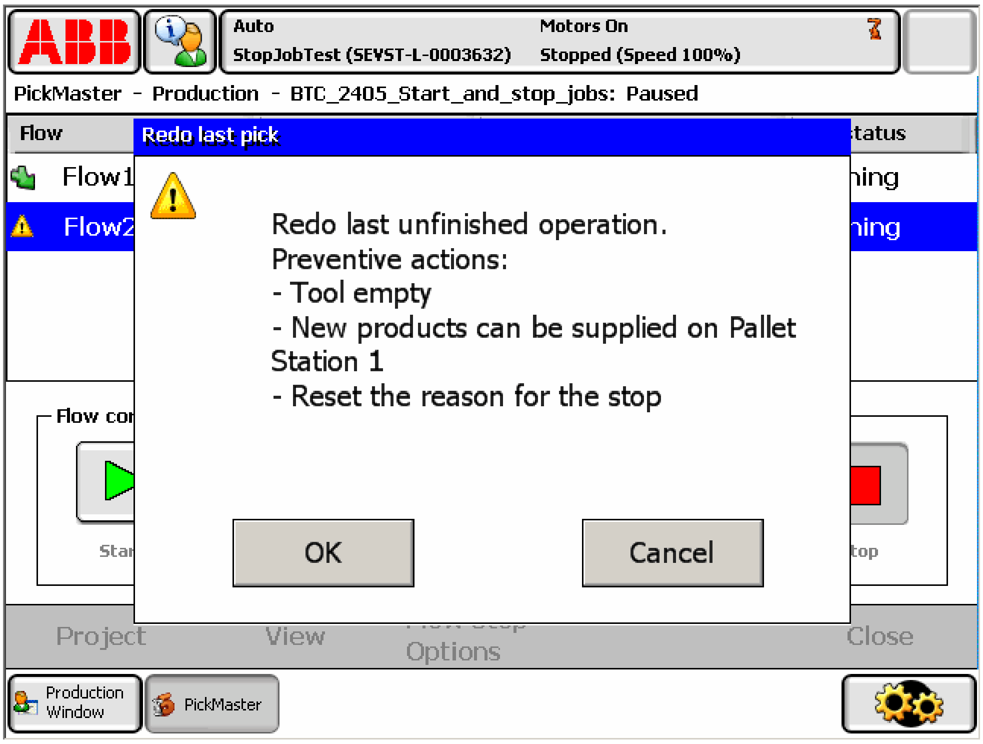
en1000000988

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 unfinished operation.
Preventive actions:
- Tool empty
- New products can be supplied on Pallet Station 1
- Reset the reason for the stop

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

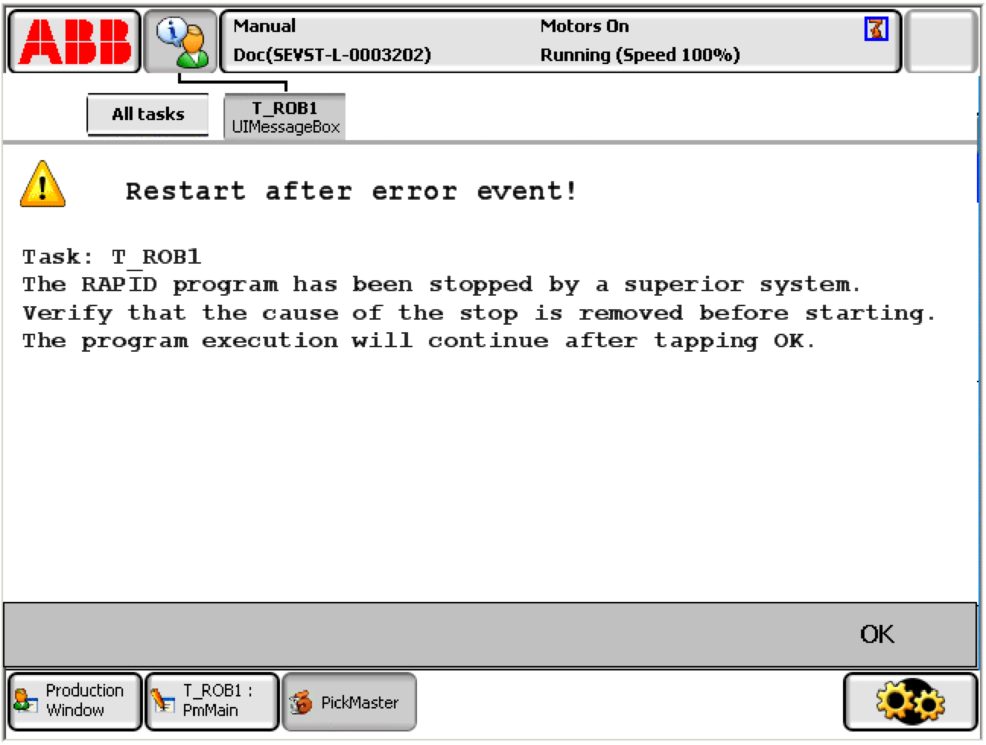
WARNING

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`.

Continues on next page

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 216

7.2.5 Process Signals

Introduction

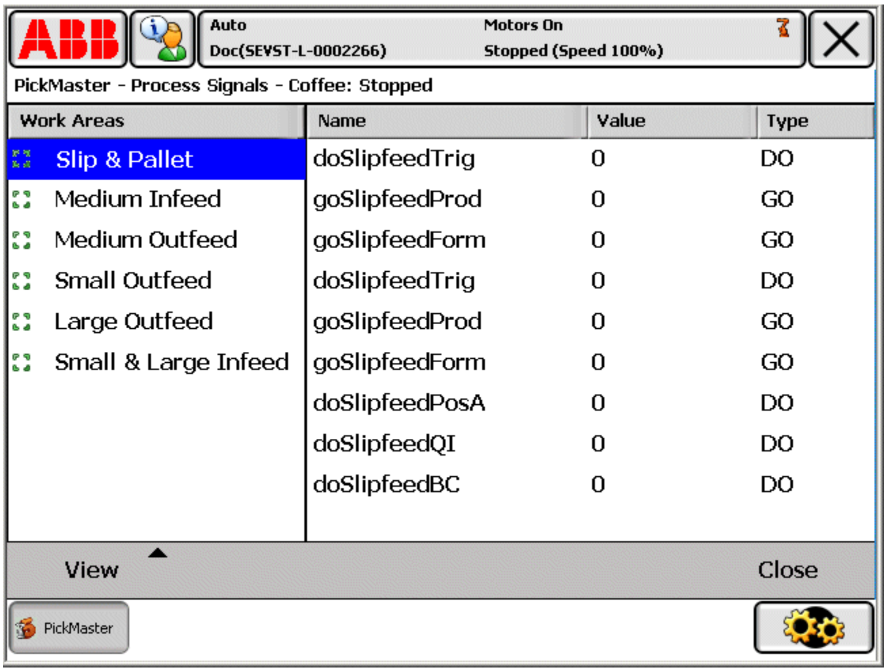
This section describes how to use the **Process Signals** window to manually control and view:

- The I/O signals that are connected to a work area.
- The zones and the activators that build up the tool configuration.
- The 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 *The Work Area Configuration on page 73*.
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.



Auto
Doc(SEVST-L-0002266)
Motors On
Stopped (Speed 100%)

PickMaster - Process Signals - Coffee: Stopped

Work Areas	Name	Value	Type
Slip & Pallet	doSlipfeedTrig	0	DO
Medium Infeed	goSlipfeedProd	0	GO
Medium Outfeed	goSlipfeedForm	0	GO
Small Outfeed	doSlipfeedTrig	0	DO
Large Outfeed	goSlipfeedProd	0	GO
Small & Large Infeed	goSlipfeedForm	0	GO
	doSlipfeedPosA	0	DO
	doSlipfeedQI	0	DO
	doSlipfeedBC	0	DO

xx0700000226

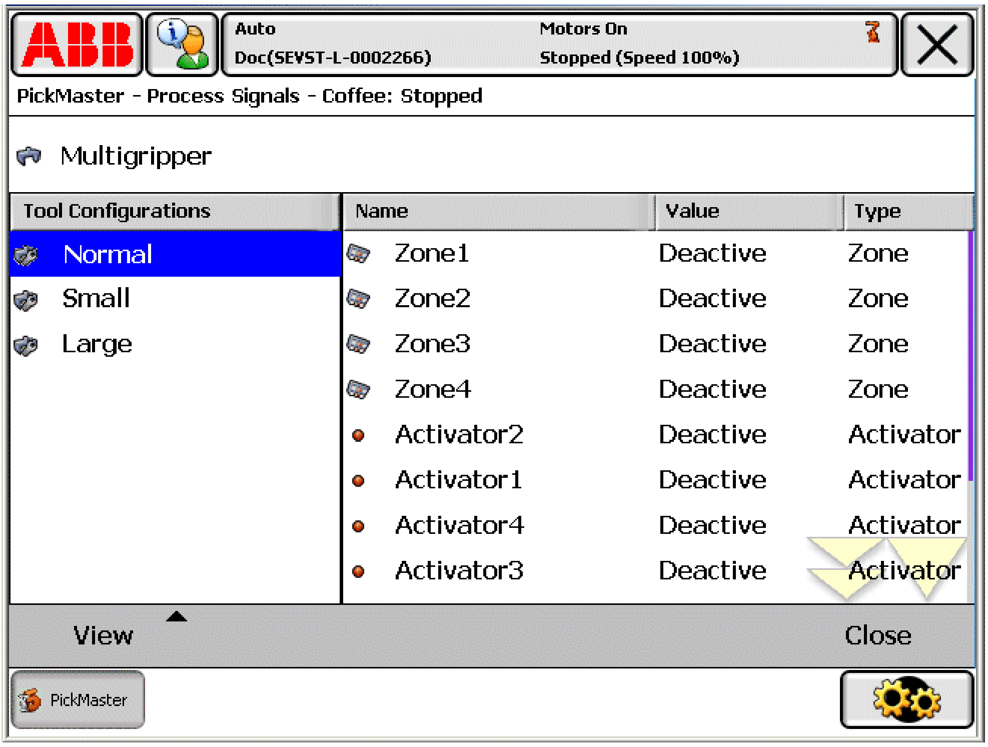
Viewing the tool configuration

	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

Continued

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

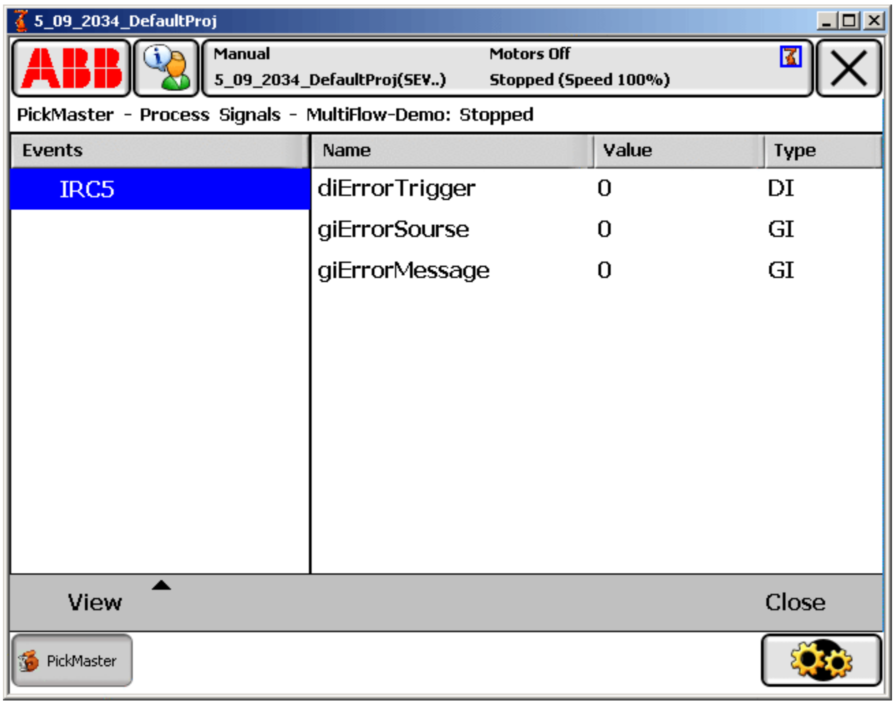
xx0700000225

Viewing events

	Action	Note
1.	On the **PickMaster** main menu, tap **Process Signals**.	
2.	Tap **View** and select **Events**.	See *The Controller Properties window on page 66*.
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
IRC5

Name	Value	Type
diErrorTrigger	0	DI
giErrorSourse	0	GI
giErrorMessage	0	GI

xx0700000407

Related information

Opening a project on page 214.

Starting and stopping production on page 216.

7.2.6 Tuning

Introduction

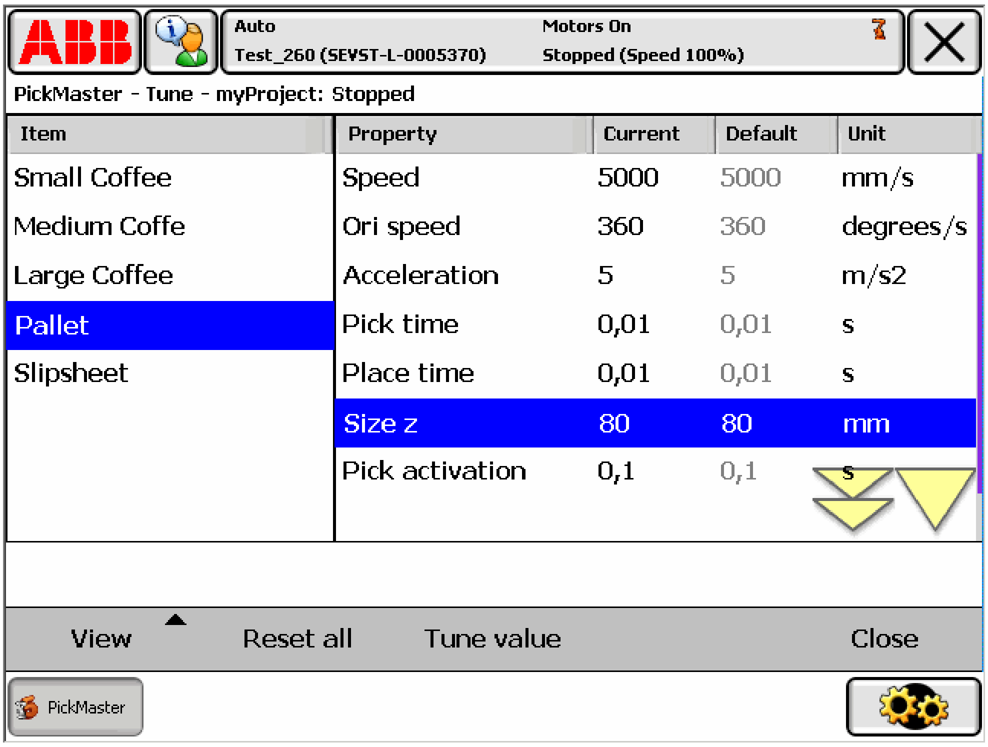
This section describes how to tune the parameter values online in the **Tune** window. A parameter can be tuned at any time for a selected project, for example while the project is running. Parameter tune updates affects the received data of the next calls to Pickmaster RAPID instructions (for example `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.



Auto
Test_260 (SEVST-L-0005370)
Motors On
Stopped (Speed 100%)

PickMaster - Tune - myProject: Stopped

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 *The Item Configuration on page 105*.
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.

Continues on next page

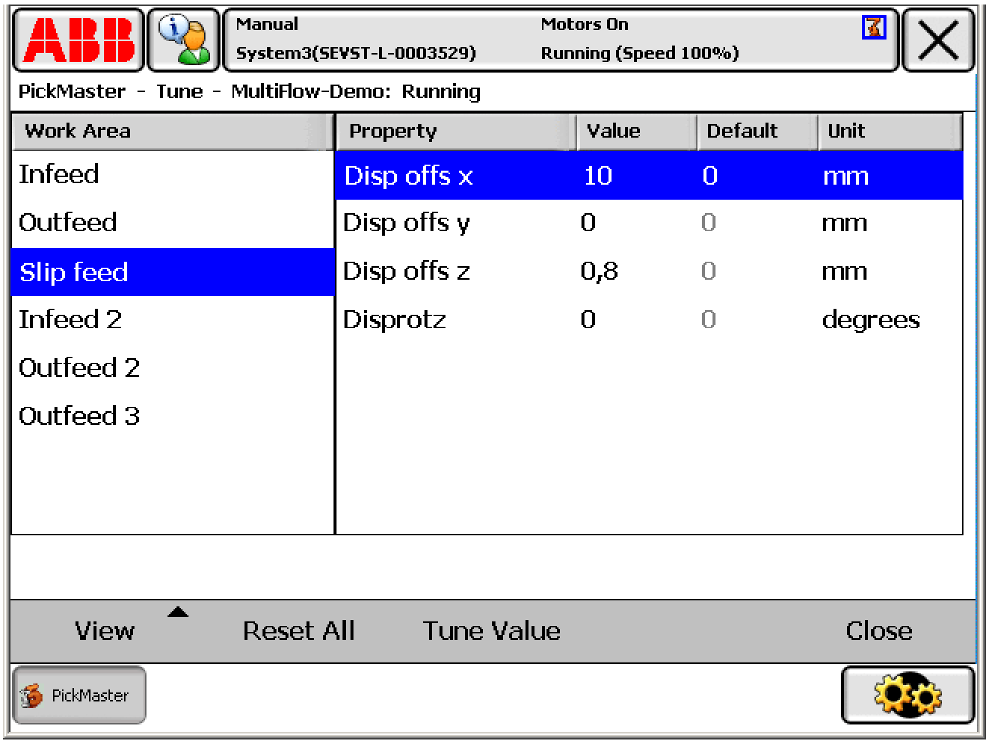
The following table describes the **Property** parameters:

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.
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.

Continues on next page

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.



Manual
System3(SEVST-L-0003529)

Motors On
Running (Speed 100%)

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 *The Work Area Configuration on page 73*.
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.
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.

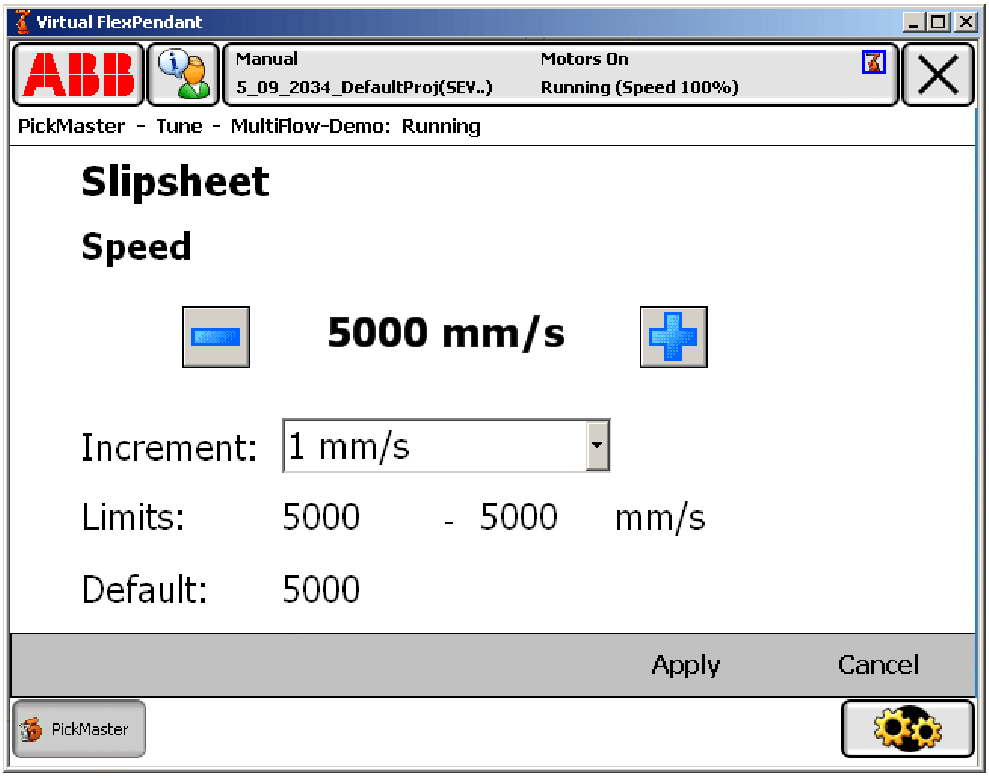
Tuning an item

	Action	Note
1.	On the **Tune** menu, tap **View**.	
2.	Tap **Item**.	

Continues on next page

	Action	Note
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.



PickMaster - Tune - MultiFlow-Demo: Running

Slipsheet

Speed

5000 mm/s

Increment: 1 mm/s

Limits: 5000 - 5000 mm/s

Default: 5000

xx0700000411

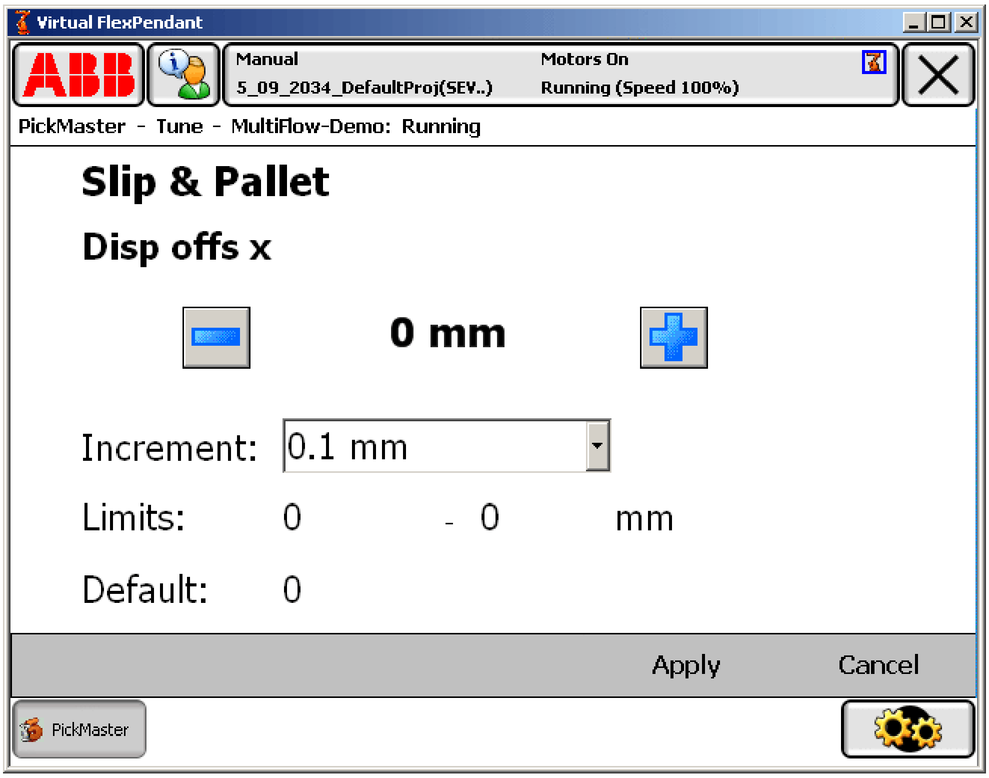
Tuning a work area

	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.	
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.

Continues on next page

	Action	Note
7.	Tap **Apply**.	

The following figure illustrates tuning of a work area.


Virtual FlexPendant
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

7.3 I/O interface

7.3.1 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:

- The basic I/O interface defines work area specific signals. It covers the minimum I/O configuration needed to run a PickMaster project.
- The 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.

7.3.2 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:

- 8 work areas
- 1 robots
- 1 controllers
- 4 flows

Use RobotStudio for:

- Renaming, reconfiguring, or removing default signals.
- Adding 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:

- *pmInfeeder1*
- *pmInfeeder2*
- *pmInfeeder3*
- *pmInfeeder4*
- *pmOutfeeder1*
- *pmOutfeeder2*
- *pmOutfeeder3*
- *pmOutfeeder4*
- *pmSlipsheet1*
- *pmPallet1*

Controller system actions

Default signals are defined for controller system actions. Signal name prefix:

- *pmSystem*

Project handling

Default signals are defined for project handling. Signal name prefix:

- *pmProject*

Continues on next page

Flows

Default signals are defined for flows. Signal name prefixes:

- *pmFlow*
- *pmFlow1*
- *pmFlow2*
- *pmFlow3*
- *pmFlow4*

Grippers

Default signals are defined for a gripper. Signal name prefix:

- *pmGripper1*

Event reporting

Default signals are defined for event reporting. Signal name prefix:

- *pmEvent*

7.3.3 Basic I/O interface

General

The I/O signals are used by a PLC to:

- Start and stop jobs on master work areas.
- Control and supervise the flow of products on work areas.
- Control and supervise the robot execution on work areas.
- Control 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:

- Target generation trigger signal (DI)
- Target generation product selection (GI)[1]
- Position request trigger signal (DO)[1]

The following signals must be setup if simulated target generation is selected for the work area. See *The Work Area Configuration on page 73*.

- Position request trigger signal (DO)

1) The signal is only mandatory if there is more than one operation set in the position source.

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 work area position source. 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:

- Work area is empty and ready to receive products to build a new pallet.
- Work 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*.

Continues on next page

Target generation format selection (GI)

The signal has no function and is not required.

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:

- When the corresponding flow is started.
- When an operation set is completed on the work area.
- When an operation set is finished using the robot execution signal.
- As 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:

- A new operation set is generated by the PLC.

Continues on next page

- Any operation, except the last one of an operation set, is performed by the robot.

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:

- A new operation set is generated and the robot execution signal is set by the PLC.
- Any 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

- Any operation is received in RAPID.
- The 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*.

Height state (GO)

The signal is optional.

Continues on next page

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 sections *Illustration, Position Source Configuration on page 125* and *Robot path height, configuration parameters on page 125*.
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 sections *Illustration, Position Source Configuration on page 125* and *Robot path height, configuration parameters on page 125*.
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 *The Work Area Configuration on page 73*.

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*.

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

Continues on next page

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:

- Target generation trigger signal (DI)
- Target generation product selection (GI)[2]
- Target generation format selection (GI)[3]
- Position request trigger signal (DO)
- Position request product selection (GO)[2]
- Position request format selection (GO)[3]

The following signals must be setup if simulated target generation is selected for the work area. See *The Work Area Configuration on page 73*.

- Position request trigger signal (DO)
- Position request product selection (GO)[2]
- Position request format selection (GO)[3]

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 position source. 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*.

Continues on next page

Continued

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 *The Work Area I/O Settings on page 78*.

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*.

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.

Continues on next page

Continued

If the *Robot execution* signal not is defined, *Position available* is set after:

- A new operation set is generated by the PLC.
- Any 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:

- A new operation set is generated and the *Robot execution* signal is set by the PLC.
- Any 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:

- Any operation is received in RAPID.
- The *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*.

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*.

Continues on next page

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 sections *Illustration, Position Source Configuration on page 125* and *Robot path height, configuration parameters on page 125*.
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 sections *Illustration, Position Source Configuration on page 125* and *Robot path height, configuration parameters on page 125*.
4	Value. The height is set to a value with the RAPID function Pm-SetDefaultHeight.

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.

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.

Continues on next page

Continued

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*.

7.3.4 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.

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 **Position Source Configuration**. See sections *Illustration, Position Source Configuration on page 125* and *Robot path height, configuration parameters on page 125*.

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 *Setting up Project I/O values on page 54*.
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.

Continues on next page

Continued

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*.
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 section *The Work Area I/O Settings on page 78*.
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.

Continues on next page

Flow signal	Description
pmFlow_diStop	Stop selected flow.
pmFlow_diRecover	Recover the selected flow with the selected recover action on the selected work area.
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 signals for all flows are setup in the **Flow I/O settings editor**. See *The Flow I/O settings editor on page 148*.

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.

Continues on next page

	Action
2	Set *pmFlow_giSelection* to select flow. The I/O value for the flow is set in the **Flow I/O settings editor**. See. *The Flow I/O settings editor on page 148*
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.

Stopping a flow

	Action
1	Set *pmFlow_giSelection* to select flow. The I/O value for the flow is set in the **Flow I/O settings editor**. See *The Flow I/O settings editor on page 148*.
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 *The Controller Properties window on page 66*, 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 I/O settings editor**. See *The Flow I/O settings editor on page 148*.
2	Set *pmFlow_giRecoverAction* to select recover action.

Continues on next page

	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 *The Work Area I/O Settings on page 78*.
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 282*). 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.

Event and error reporting

It is possible to report events and errors for work areas, robots, and controllers that affect the runtime operation. See *The Controller Properties window on page 66*.

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 section *The Controller Properties window on page 66*.
2	Set *pmEvent_giMessage* to select a message. Values that represent different messages are set in the **Message Settings**, see section *The Message Settings on page 149*.
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 section *The Controller Properties window on page 66*.
2	Set *pmEvent_giMessage* to select a message. Values that represent different messages are set in the **Message Settings**, see section *The Message Settings on page 149*
3	Pulse *pmEvent_diTrigger* to generate the elog message.

Continues on next page

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	Zone activation.
pmGripper1_goSearchActivate	Search tool activation.
pmGripper1_diSearchStop	Stack search stop trigger.
pmGripper1_goToolEvent1	GO tool event.
pmGripper1_doToolEvent1	DO tool event.
pmGripper1_diToolEvent1	DI tool event.
pmGripper1_giToolEvent1	GI tool event.

7.3.5 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:

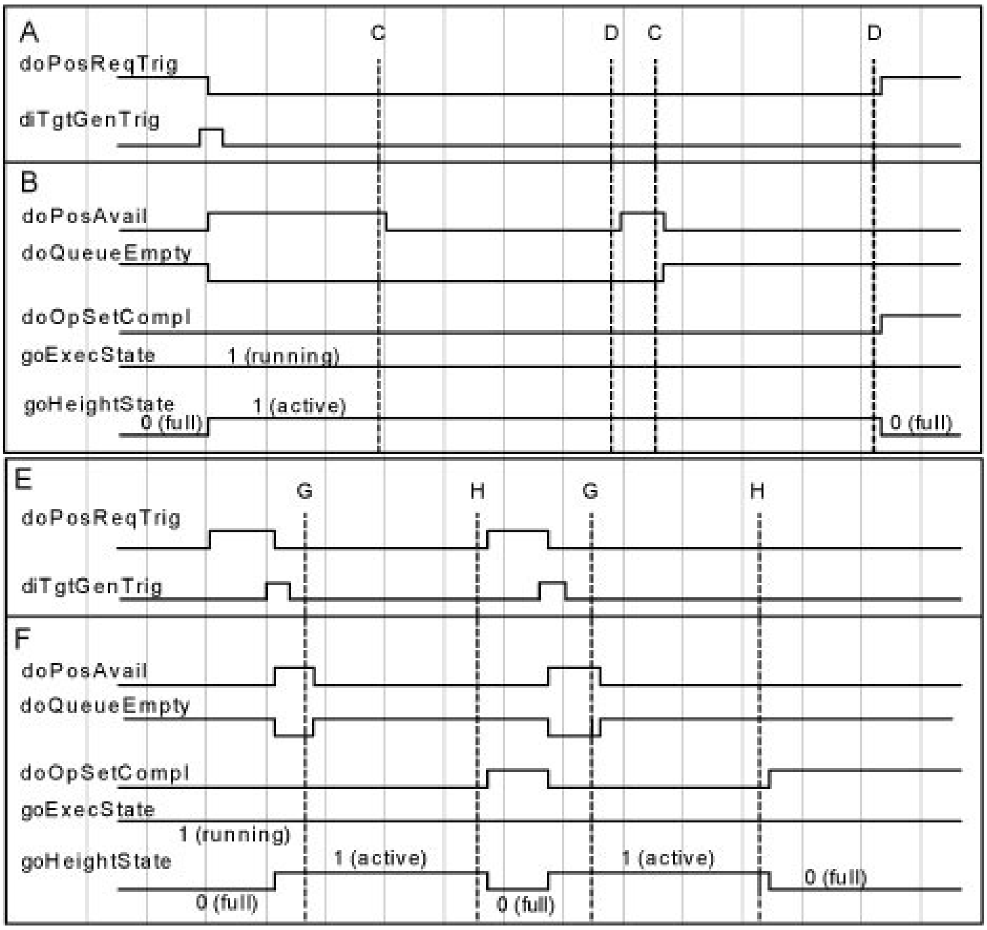
- *Example minimum process control on a running flow on page 256*.
- *Example robot execution control on page 257*.
- *Example height control of a running flow on page 258*.
- *Example flow control on page 259*.

Example minimum process control on a running flow

Task: Pick single items from infeeder and place pallet pattern with two items on outfeeder.

Continues on next page

Settings: Early request, Use concurrency, non-pulsed controller mode.



en1000000195

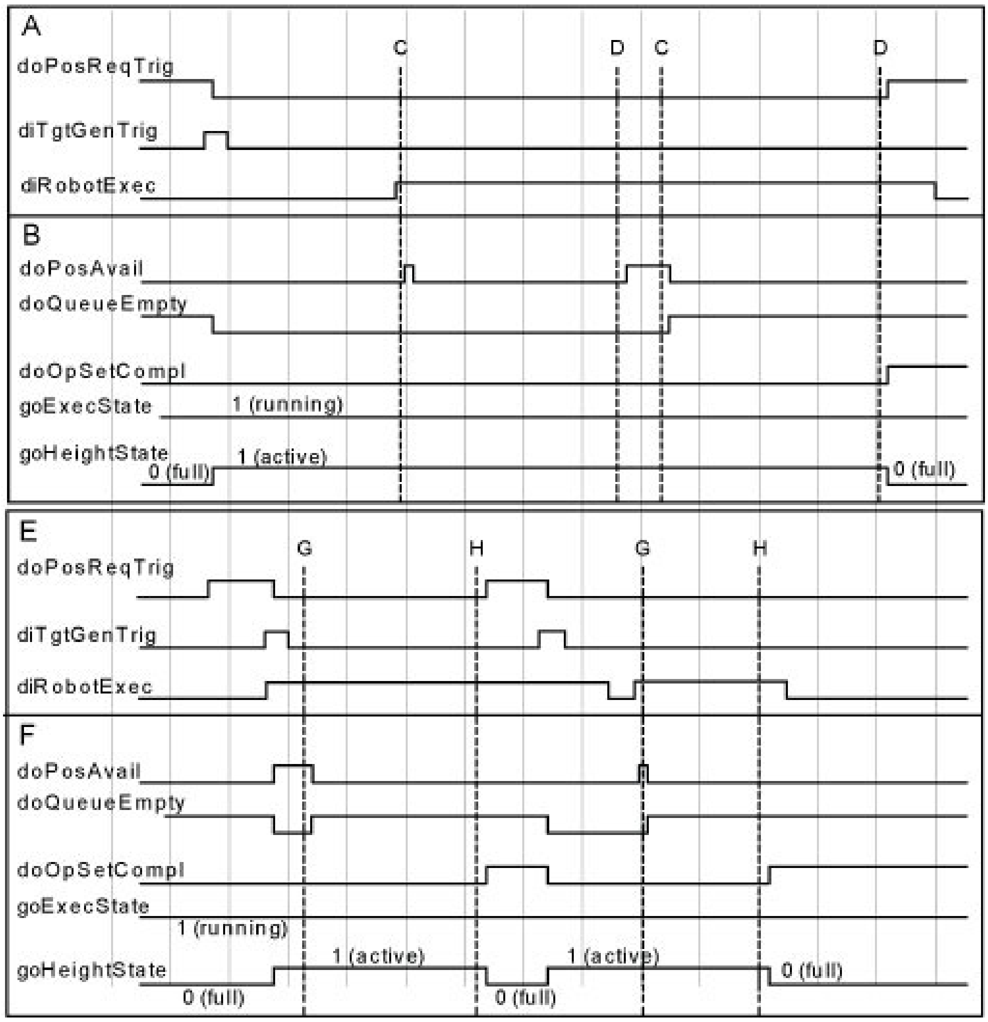
A	Master outfeeder, process control
B	Master outfeeder, process status
C	Operate outfeeder
D	Place
E	Slave infeeder, process control
F	Slave infeeder, process status
G	Operate infeeder
H	Pick

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.

Continues on next page

Settings: Early request, Use concurrency, non-pulsed controller mode, default height *Full* on both infeeder and outfeeder.



en1000000198

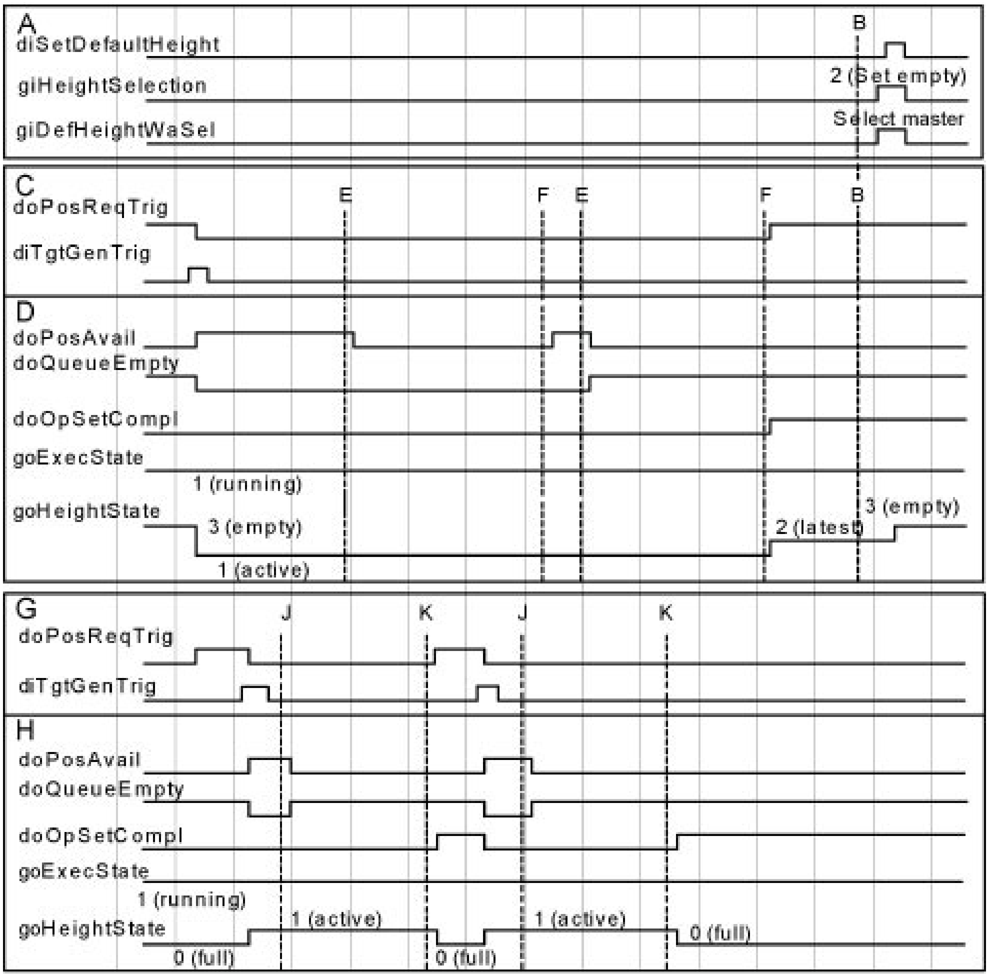
A	Master outfeeder, process control
B	Master outfeeder, process status
C	Operate outfeeder
D	Place
E	Slave infeeder, process control
F	Slave infeeder, process status
G	Operate infeeder
H	Pick

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.

Continues on next page

Settings: Early request, Use concurrency, non-pulsed controller mode, default height *Full* on infeeder and *Latest* on outfeeder.



en1000000202

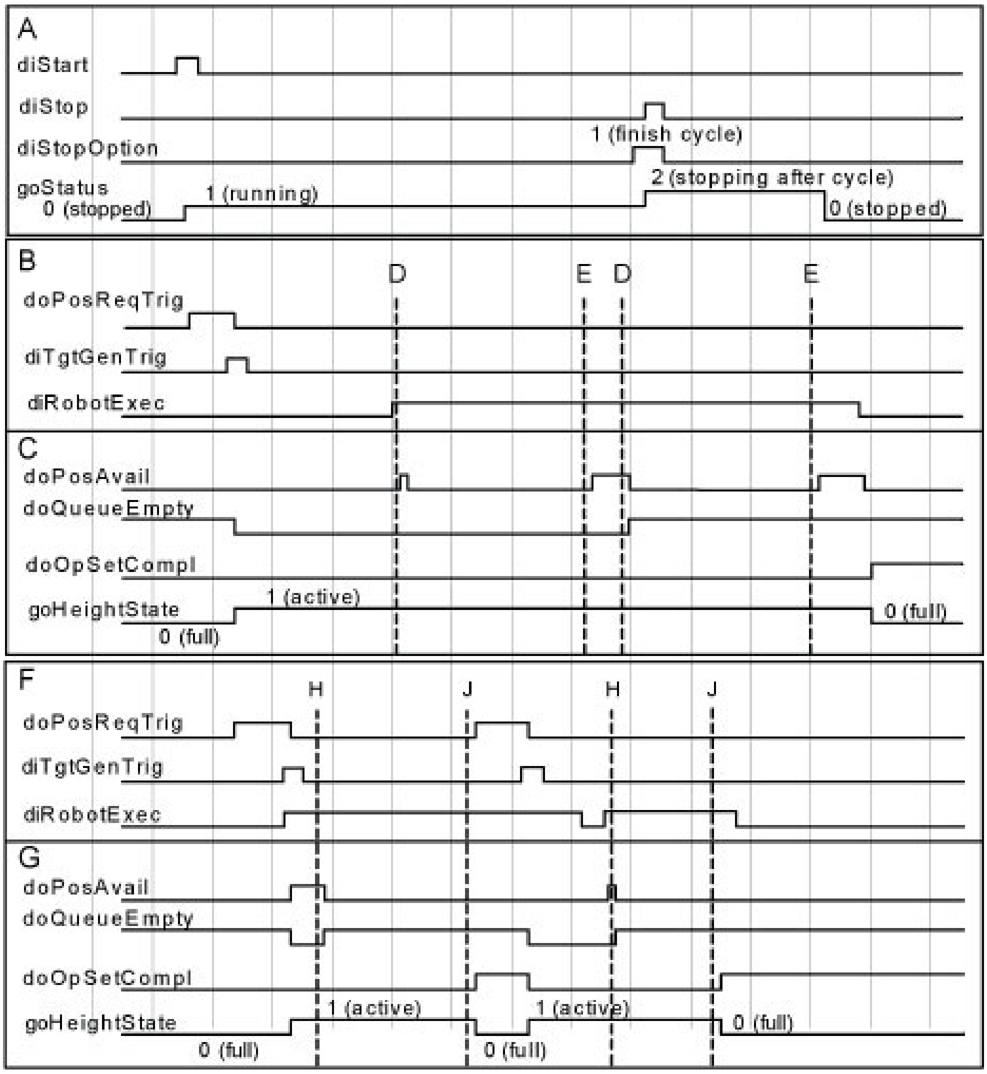
A	Project control, process control
B	Pallet pattern unloaded from outfeeder
C	Master outfeeder, process control
D	Master outfeeder, process status
E	Operate outfeeder
F	Place
G	Slave infeeder, process control
H	Slave infeeder, process status
J	Operate infeeder
K	Pick

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.

Continues on next page

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

8 RAPID reference information

8.1 Introduction to RAPID reference information

Structure of this chapter

This chapter describes the RAPID instructions, functions, and data types that are specific for PickMaster.

8.2 Instructions

8.2.1 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 > ';'
```

Related information

For information about	See
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 345.*

Continues on next page

Continued

For information about	See
The data type `pm_targetdata`	*pm_targetdata - PickMaster target data on page 341*.
The data type `pm_acktype`	*pm_acktype - PickMaster target acknowledge type on page 312.*

8.2.2 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.

[\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

Continues on next page

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 >] ';'
```

Related information

For information about	See
The data type `confdata`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Data types*.

Continues on next page

For information about	See
The data type `robtarget`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Data types*.

8.2.3 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 268*.

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 345*
The **Robot Flow Configuration**	*The Flow Configuration on page 145*

8.2.4 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

Continued

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 321*.

8.2.5 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 345*
The instruction PmSetLastWa	*PmSetLastWa - Set last used work area on page 281*
The instruction PmGetPathHeight	*PmGetPathHeight - Get a safe path height for an intermediate movement on page 299*

8.2.6 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

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 345*.
The data type pm_operationdata	*pm_operationdata - PickMaster operation data on page 328*.

8.2.7 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 275*.

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 *Setting up Project I/O values on page 54*.

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

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 331*.

8.2.8 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 345*
The data type wobjdata	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Data types*

8.2.9 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

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 346*
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 345*
The instruction `PmGetFlow`	*PmGetFlow - Get flow to execute on page 267*
The instruction `PmGetWaByWobj`	*PmGetWaByWobj - Get a reference to a work area using a work object data on page 276*

8.2.10 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

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 131*.
Format frame versus work object in format operation set	*Format frame versus work object in format operation set on page 359*.
Pallet pattern versus work object in pallet pattern operation set	*Pallet pattern frame versus work object in pallet pattern operation set on page 361*.
The data type pm_wadescr	*pm_wadescr - PickMaster work area reference on page 345*.
The data type pm_searchtype	*pm_searchtype - PickMaster stack search type on page 336*.

8.2.11 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 345*
The instruction PmGetLastWa	*PmGetLastWa - Get last used work area on page 271*
The instruction PmGetPathHeight	*PmGetPathHeight - Get a safe path height for an intermediate movement on page 299*

8.2.12 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

Continues on next page

Continued

The third argument to the event log message, one space if not used.

`[\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:

- `PM_RECOVER_REDO_LAYER` or `PM_RECOVER_NEXT_PALLET` without a valid work area.
- A recover action not in range.
- `PM_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

Continued

```
<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>
```

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.

Continues on next page

Constant	Value	Description
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 286*

8.2.13 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 I/O settings editor. See *The Flow I/O settings editor on page 148*.

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

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 269*.
The instruction `PmStopFlow`	*PmStopFlow - Stop a specific flow on page 289*.
The instruction `PmSetRecoverAction`	*PmSetRecoverAction - Set flow recover action on page 282*
The data type `pm_flowinfo`	*pm_flowinfo - PickMaster flow information on page 321*.

8.2.14 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 291*.

8.2.15 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 *The Flow I/O settings editor on page 148*.

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 274*.
The instruction `PmStartFlow`	*PmStartFlow - Starts a specific flow on page 286*.
The data type `pm_flowinfo`	*pm_flowinfo - PickMaster flow information on page 321*.

8.2.16 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 288*.

8.2.17 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 288.*

8.3 Functions

8.3.1 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

`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

`[\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`.

8.3.2 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.

TargetHandle

Data type: pm_targethandle

Contains a reference to a target.

Continues on next page

`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 345.*
The data type `pm_targethandle`	*pm_targethandle - PickMaster target handle on page 344.*
The data type `pm_actionhandle`	*pm_actionhandle - PickMaster action handle on page 316.*
The data type `pm_eventdata`	*pm_eventdata - PickMaster event data on page 318.*
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*.

8.3.3 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 242*. 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 **Work Area Order Settings** tab, see section *The Controller Properties window on page 66*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 302*.

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 181*.

`ToWa`

Data type: `pm_wadescr`

Continues on next page

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 181*.

`[\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 *The Position Source Configuration on page 124*.

`[\UseDefaultHeight]`

Data type: `switch`

Consider the default heights for work areas having an active height. For more information on how to configure safe positions, see *The Position Source Configuration on page 124*.

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 345*
The instruction `PmSetLastWa`	*PmSetLastWa - Set last used work area on page 281*
The instruction `PmGetLastWa`	*PmGetLastWa - Get last used work area on page 271*

8.3.4 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 242*. 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

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 **Position Source Configuration** for the work area. See *The Position Source Configuration on page 124*.

`[\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 **Position Source Configuration** will always be added to the specified height.

8.3.5 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,

Continues on next page

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

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 345*.
The data type `pm_ophandle`	*pm_ophandle - PickMaster operation handle on page 330*.
The data type `pm_targetdata`	*pm_targetdata - PickMaster target data on page 341*.

8.3.6 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.

Syntax

```
PmGetTgtAction '('
  [ Wa ':=' ] < expression (IN) of pm_wadescr > ','
  [ TargetHandle ':=' ] < expression (IN) of pm_targethandle > ','
  [ Action ':=' ] < expression (INOUT) of pm_actiondata > ')'
```

Continues on next page

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 345*.
The data type `pm_targethandle`	*pm_targethandle - PickMaster target handle on page 344*.
The data type `pm_actiondata`	*pm_actiondata - PickMaster action data on page 313*.

8.3.7 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 345.*

8.3.8 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.

8.4 Data types

8.4.1 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.

Continues on next page

Used as argument `DecelLim` in `PathAccLim`.

`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 313.*
The function `PmGetTgtAction`	*PmGetTgtAction - Get target action on page 306.*
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*.

8.4.2 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 262*.

8.4.3 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.

`Speed`

Data type: `speeddata`

Continues on next page

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 >
```

Related information

For information about	See
The data type `pm_movetype`	*pm_movetype - PickMaster movement type on page 325.*

Continues on next page

Continued

For information about	See
The data type `pm_accdata`	*pm_accdata - PickMaster acceleration/deceleration data on page 310*.
The data type `pm_searchdata`	*pm_searchdata - PickMaster search data on page 335*
The function `PmGetTgtAction`	*PmGetTgtAction - Get target action on page 306*.
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*.

8.4.4 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 313*.
The function PmGetTarget	*PmGetTarget - Get target on page 304*.
The function PmGetTgtAction	*PmGetTgtAction - Get target action on page 306*.
The function PmGetEvent	*PmGetEvent - Get events for an action on page 297*.

8.4.5 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 313.*
The instruction GripLoad	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Instructions*.

8.4.6 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.

`Dvalue`

Data type: `dnum`

Continues on next page

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 <= 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
```

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 320*.
The function `PmGetEvent`	*PmGetEvent - Get events for an action on page 297*.
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*.

8.4.7 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 318*.
The instruction TriggEquip.	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Instructions*.

8.4.8 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 *The Flow I/O settings editor on page 148*.

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 269.*
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 345.*
The data type `pm_slavewainfo`	*pm_slavewainfo - PickMaster slave work area information on page 338.*

8.4.9 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 *The Robot Settings on page 154*.

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 274*.
The data type `pm_robotinfo`	*pm_robotinfo - PickMaster robot information on page 333*.

8.4.10 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 313*.
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*.

8.4.11 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

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 313*.
The function `PmGetTgtAction`	*PmGetTgtAction - Get target action on page 306*.
The function `Offs`	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Functions*.

8.4.12 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 **Format 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 272*.
The format id value	*The Format Configuration on page 110*.

8.4.13 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 328.*
The instruction PmGetOperation	*PmGetOperation - Get operation from a work area on page 272.*

8.4.14 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 *Setting up Project I/O values on page 54*.

`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;
```

Limitations

The operation data members can only be set by the instruction `PmGetProjectInfo`.

Continues on next page

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 274.*
The data type pm_robotinfo	*pm_robotinfo - PickMaster robot information on page 333.*

8.4.15 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`.

Structure

```
< dataobject of pm_robotinfo >
  < Active of bool >
  < Name of string >
  < TaskName of string >
  < ModuleNames of pm_moduleinfo >
```

Continues on next page

Continued

Related information

For information about	See
The instruction `PmGetProjectInfo`	*PmGetProjectInfo - Get information about a specific project on page 274*.
The data type `pm_projectinfo`	*pm_projectinfo - PickMaster project information on page 331*.
The data type `pm_moduleinfo`	*pm_moduleinfo - PickMaster module information on page 323*.

8.4.16 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.

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 >
```

Related information

For information about	See
The data type pm_actiondata	*pm_actiondata - PickMaster action data on page 313.*
The function PmGetTgtAction	*PmGetTgtAction - Get target action on page 306.*
The instruction DoSearch	*Procedure PmDoSearch on page 206.*

8.4.17 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 279.*

8.4.18 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 313.*
The instruction SingArea	*Technical reference manual - RAPID Instructions, Functions and Data types*, section *Instructions*.

8.4.19 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

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 274*.
The data type `pm_flowinfo`	*pm_flowinfo - PickMaster flow information on page 321*.

8.4.20 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 206*.
The data type pm_searchdata	*pm_searchdata - PickMaster search data on page 335*.

8.4.21 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.

`AppProdsLoad`

approach product load

Continues on next page

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

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 304*.
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*.

8.4.22 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 341.*
The function `PmGetTarget`	*PmGetTarget - Get target on page 304.*
The function `PmGetTgtAction`	*PmGetTgtAction - Get target action on page 306.*

8.4.23 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 276.*

8.4.24 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 *The Work Area I/O Settings on page 78*.

`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 *The Work Area Configuration on page 73*.

`DefaultHeight`

Data type: `num`

The configured default height. See *The Position Source Configuration on page 124*.

`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

`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.

Continues on next page

Examples

Example 1

```
VAR pm_wainfo WaInfo;
VAR num WaSelection:=1;
PmGetWaInfo WaSelection,WaInfo;
```

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 277*
The data type `pm_wadescr`	*pm_wadescr - PickMaster work area reference on page 345*

9 Relationships between PickMaster frames

9.1 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.

9.2 Shape frame

Shape size

The size of the shape is specified as **X size**, **Y size** and **Z size** in the **Shape Configuration** window.

Visualization of the shape frame

The shape frame is shown in the **Shape Configuration** window, but without reference to any other frame.

In the windows **Format Configuration**, **Pallet Pattern Operation Set Configuration** and **Format Set Operation Configuration** the origin of the shape frame is marked as a blue corner on the shape.

9.3 Format frame versus shape frame

Format frame settings

The relationship between a format frame and shape frames are defined in the **Format Configuration** by the settings of **Orientation**, **Row**, and **Column**. See *The Format Configuration on page 110*.

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.

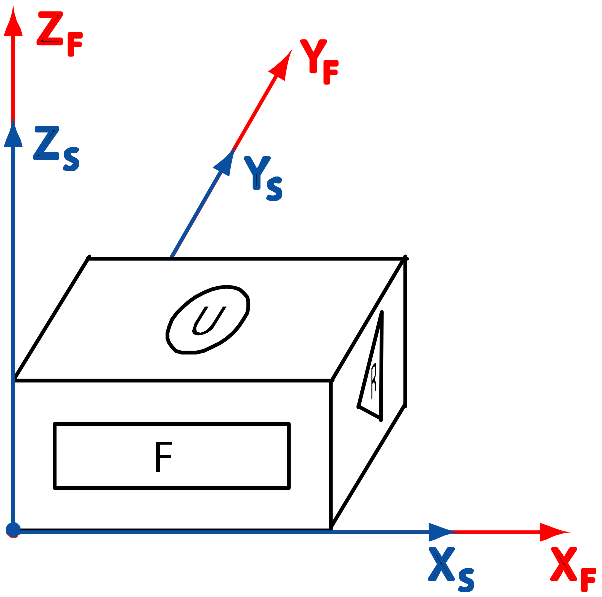
The relationship between the format frame and the shape frames will also depend on the X size, Y size, and Z size of the shape defined in the **Shape Configuration** and the tuning of item size which is displayed in the same dialog.

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

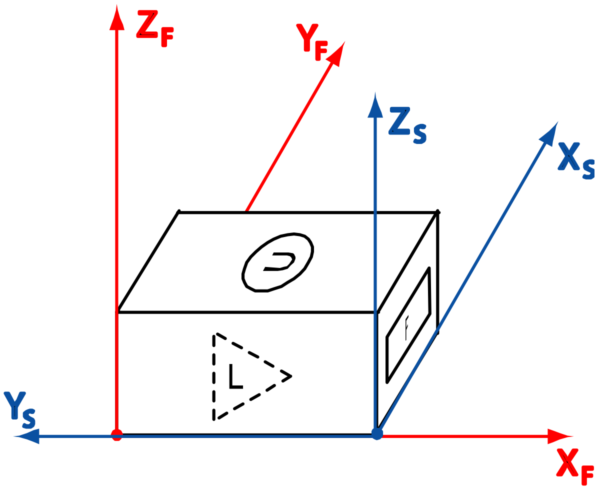


xx0700000255

X	Coordinates for the format frame
X	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

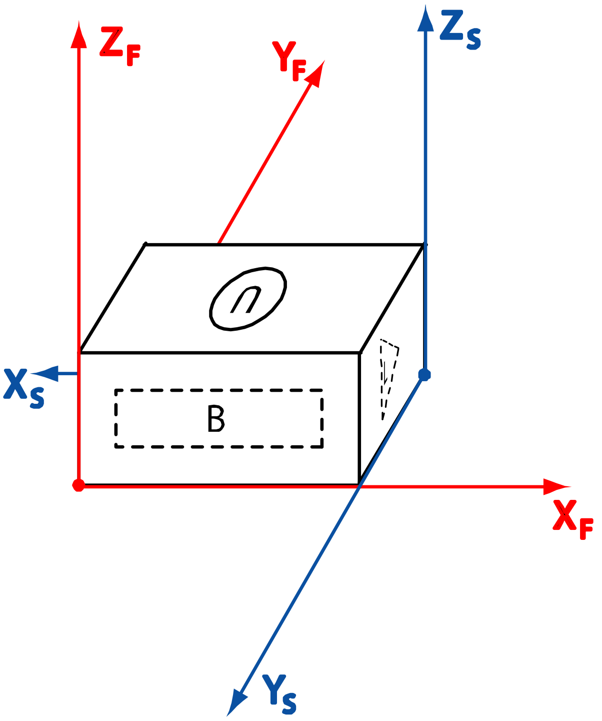
Continues on next page

Orientation: Left



xx0700000256

Orientation: Back



xx0700000257

Continues on next page

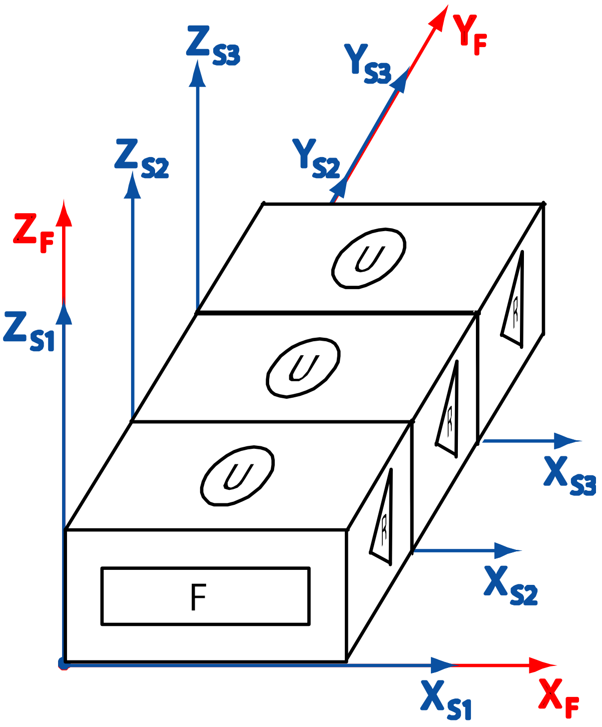
Continued

Row explanation

Row defines number of shapes in the y-direction of the format frame.

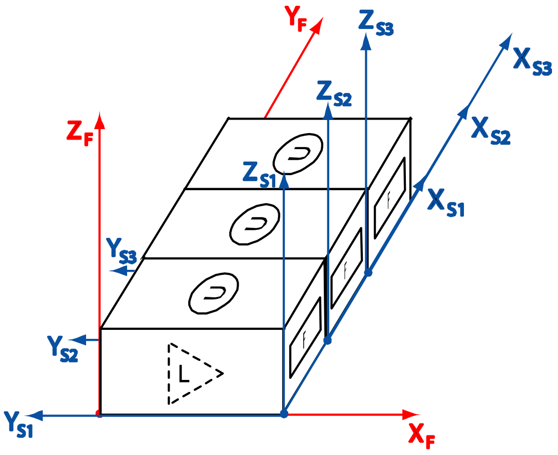
The following examples show the combination of row, column, and orientation.

Row: 3 ; Column: 1; Orientation: Front



xx0700000258

Row: 3 ; Column: 1; Orientation: Left



xx0700000259

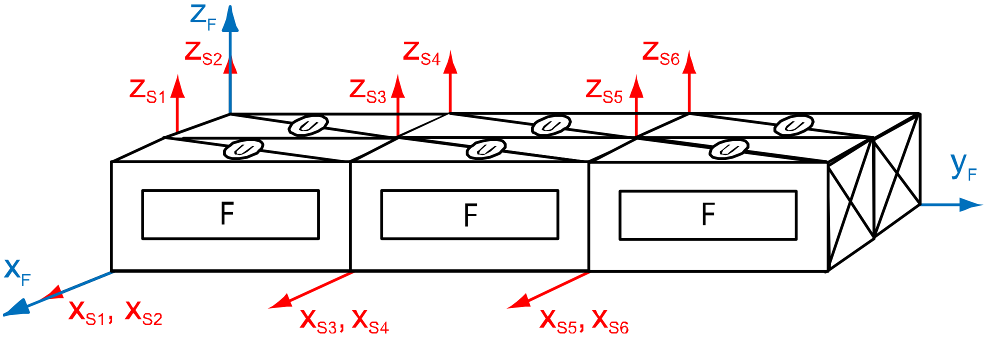
Continues on next page

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 **Format Configuration** window. For more information, see *The Format Configuration on page 110*.

9.4 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.

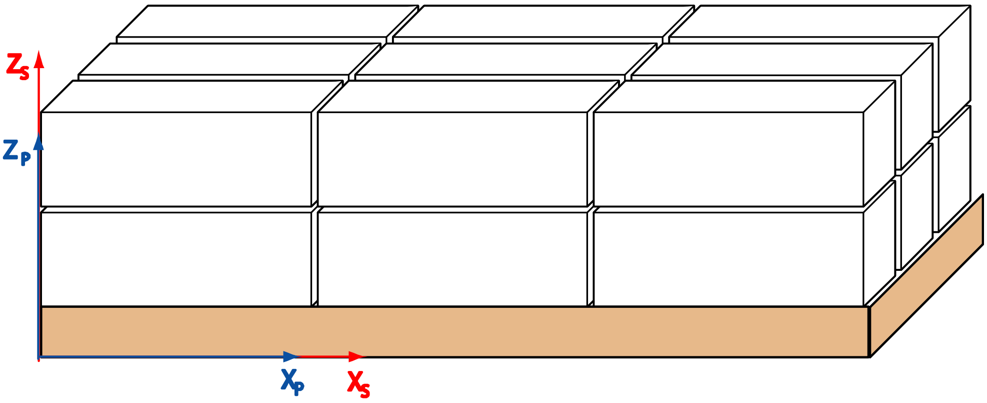
The palletizing area shape sets the x-y-size of the pallet pattern. Only box shapes and sheet shapes that are smaller than or have the same size as the palletizing area shape will be available for use within the pallet pattern.

Where to see the shape frame and pallet pattern

The location of the shape frame can be viewed in the **Shape Configuration** window.

The location of the pallet pattern frame can be viewed in the **Layout Configuration** or the **Pallet Pattern Configuration** window.

Illustration

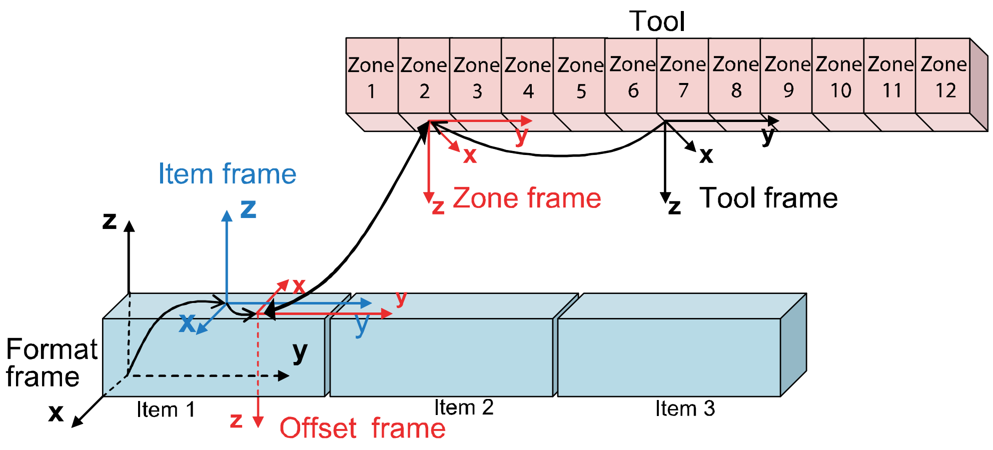


xx0700000263

X	Coordinates for the shape frame of the palletizing area shape
X	Coordinates for the pallet pattern frame

9.5 Format frame versus tool frame

Illustration



xx0700000264

Format frame

The format frame is defined in the **Format Configuration** window, under **General**.

Item frame

The offset frame is selected by setting **Target item** in the **Format 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 **Format 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 **Format Configuration** window, under **Tool location**.

Tool frame

The zone frame cannot be directly viewed in the **Format 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 **Tool configuration** in the **Format Configuration** window, under **Tool**, and by setting **Tcp zone** in the **Format 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 Configuration** 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 Configuration** window, under **Activators** tab.

Continues on next page

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 **Format Configuration** window. See *The Format Configuration on page 110*.

9.6 Pallet pattern item versus tool frame

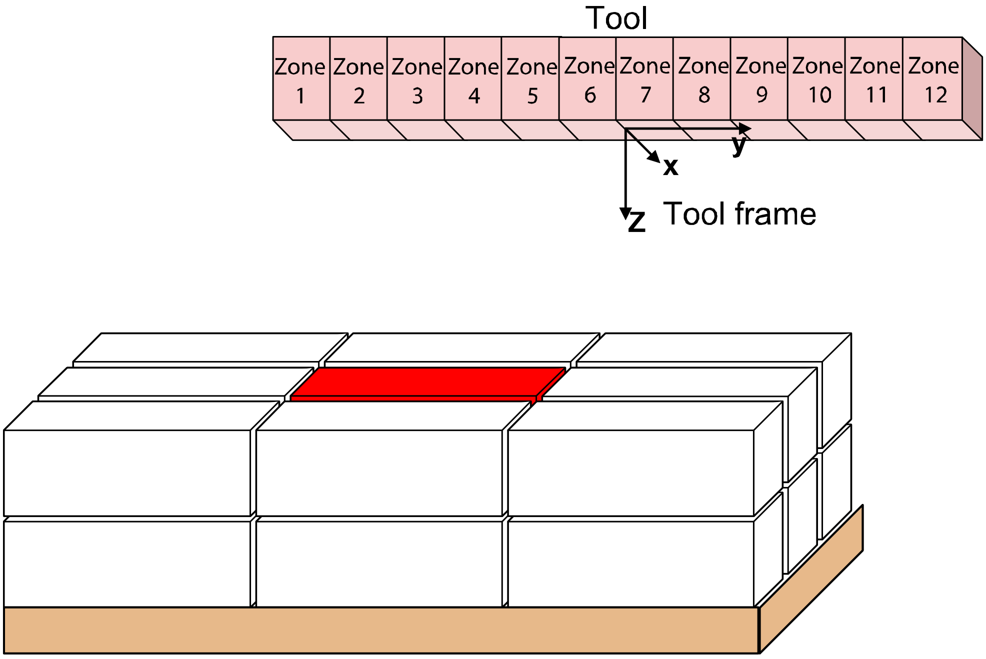
Tool location

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:

- The pick/place order within the layer.
- The format to be used.
- The item in the selected format to be used.
- The orientation of the item may be flipped 180 degrees.
- Single 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 *The Pallet Pattern Operation Set Configuration on page 130*.

Related information

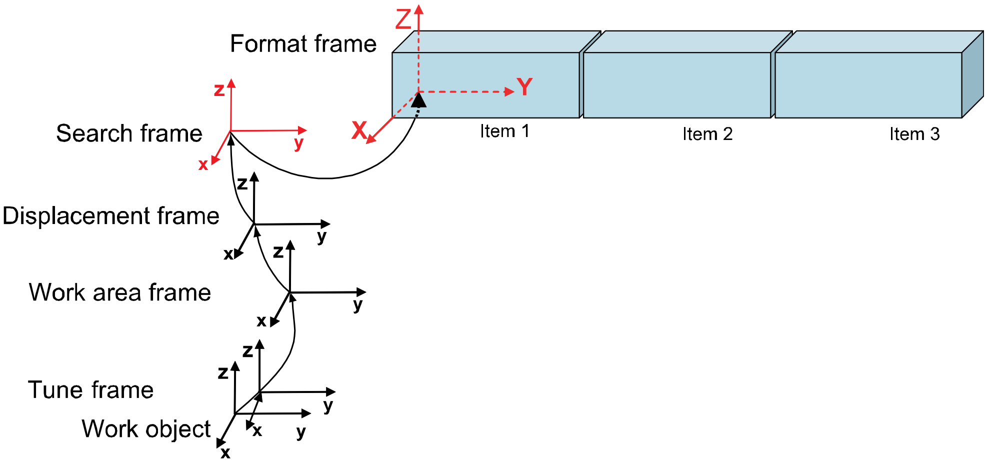
The Operation Editor on page 143

9.7 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 **Work Area 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.

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 **Work Object Orientation** settings of **Alignment** and **Rotation**.

Displacement frame

The location of the displacement frame relative the work area frame is defined by the settings of **Displacement frame**: **x**, **y**, **z** and **z angle** in the **Format 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 **Work Object Orientation** window. However, the location is only valid if the following are zero: tune frame location, displacement frame offset, reorientation, and search frame.

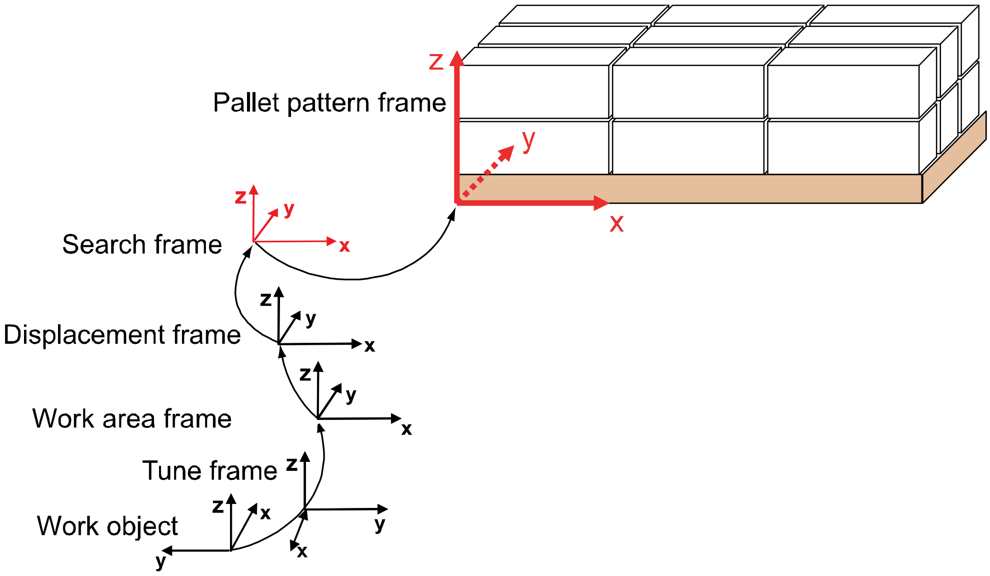
The location of the work object relative to a specific format can be viewed in the **Format Operation Set Configuration** window. However, the location is only valid if the tune frame location is zero (=default) and if the search frame offset is zero.

9.8 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 **Work Area 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.

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 **Work Object Orientation** settings of **Alignment** and **Rotation**.

Continues on next page

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 **Work Object Orientation** 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.

9.9 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.

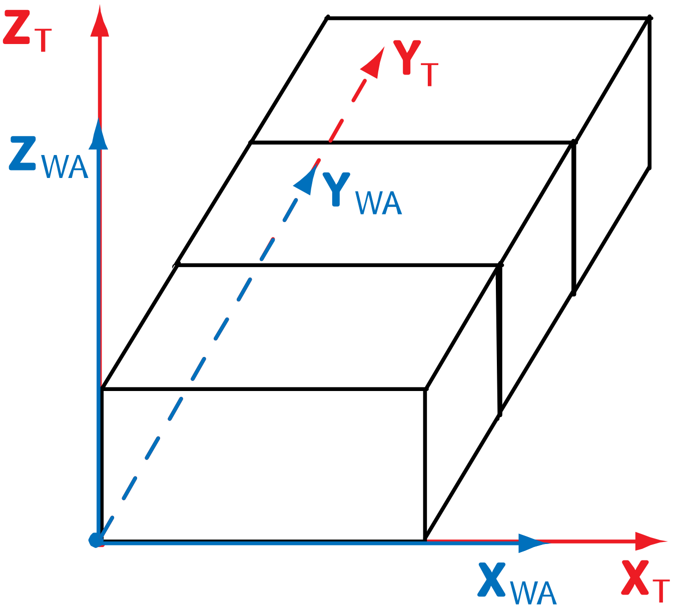
Orientation

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



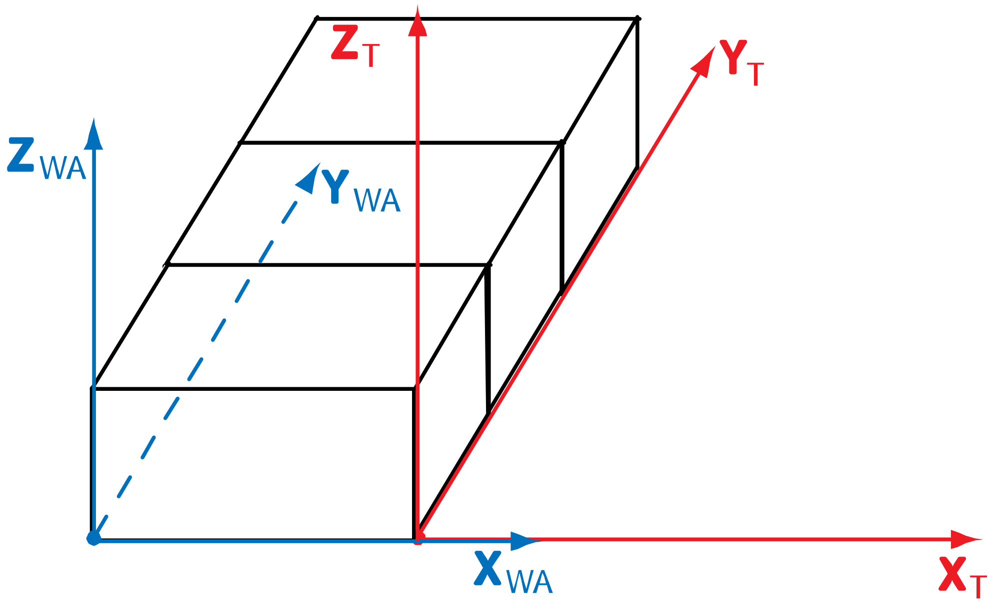
xx0700000268

X	Coordinates for the tune frame
X	coordinates for the work area frame

Continues on next page

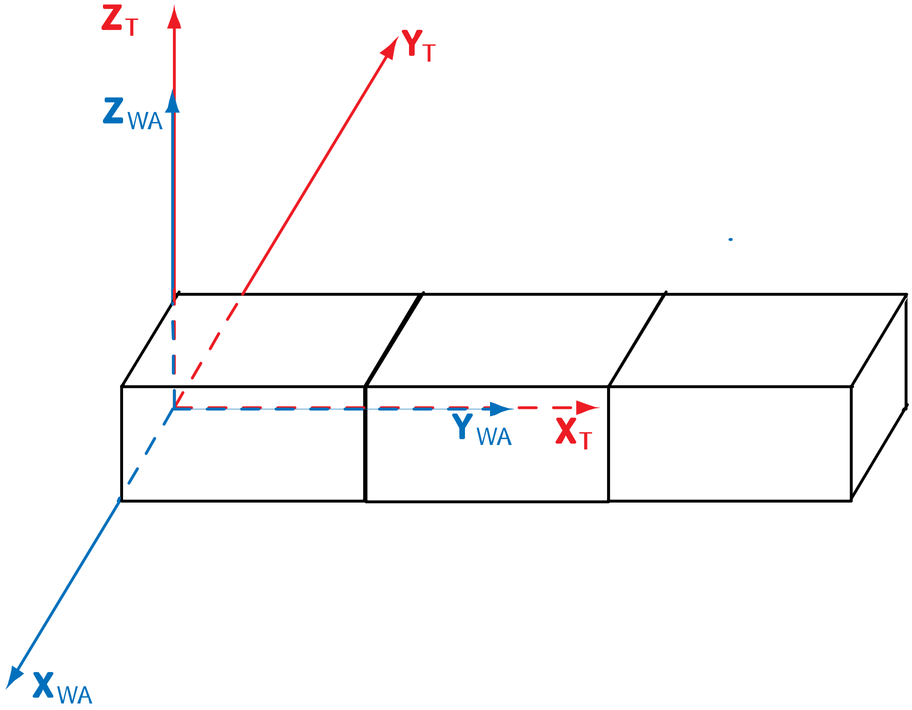
Continued

Alignment: Right; Orientation: 0



xx0700000269

Alignment: Left; Orientation: 90

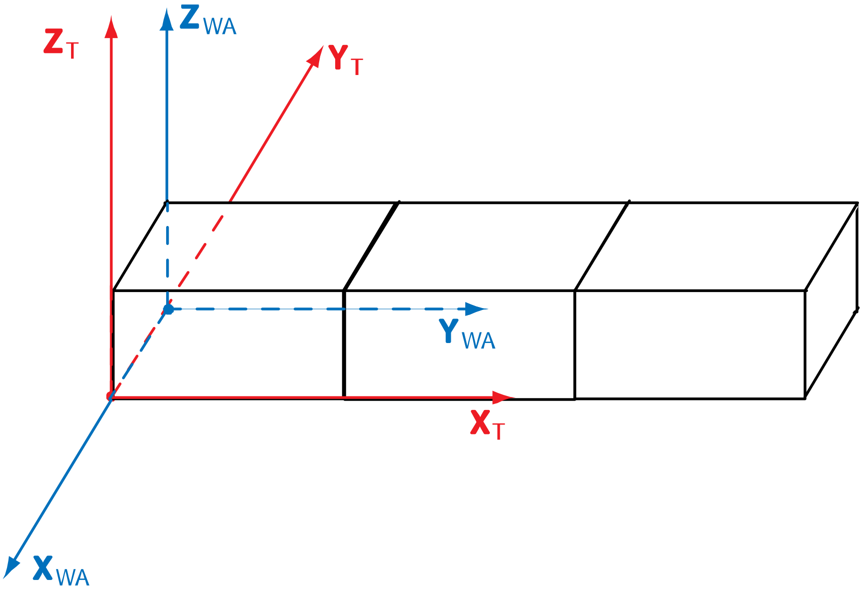


xx0700000270

Continues on next page

Continued

Alignment: Right; Orientation: 90



xx0700000271

9.10 Tune frame versus work area frame versus displacement frame

Displacement frame settings

Displacement frame settings are made in the **Format 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.

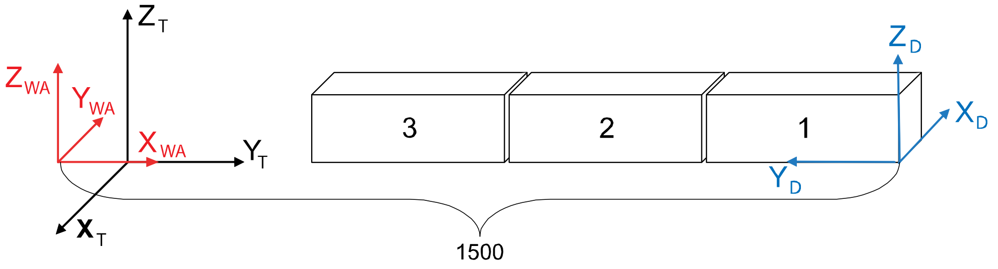
The following settings are made for the work area frame in the **Format Operation Set Configuration** window:

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	Coordinates for the tune frame
X	Coordinates for the work area frame
X	Coordinates for the displacement frame

Index

Contact us

ABB AB
Discrete Automation and Motion
Robotics
S-721 68 VÄSTERÅS
SWEDEN
Telephone +46 (0) 21 344 400
www.abb.com