Operating manual
Palletizing PowerPac
RobotStudio 5.14.02

Document ID: 3HAC042340-001
Revision: -
The information in this manual is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this manual.

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

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

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

Additional copies of this manual may be obtained from ABB.

© Copyright 2011 ABB All rights reserved.

ABB AB
Robotics Products
SE-721 68 Västerås
Sweden
# Table of Contents

4.3.22 Project Report .................................................. 100
4.3.23 Library .......................................................... 101

**4.3.24 Create Tool** .................................................. 102
  4.3.24.1 Overview .................................................. 102
  4.3.24.2 Tool Functions ............................................ 103
  4.3.24.3 Tool Events and Scenarios ............................... 120
4.3.25 Create Feeder .................................................. 124
4.3.26 Change Viewpoint ............................................. 126
4.3.27 Adjust Position of Object ................................. 127
4.3.28 Help ........................................................... 128

**4.4 Layout Browser** .................................................. 129
  4.4.1 Overview ...................................................... 129
  4.4.2 Station ........................................................ 130
  4.4.3 Robot .......................................................... 131
  4.4.4 Tool SmartComponent ......................................... 132
  4.4.5 Feeder SmartComponent ...................................... 133
  4.4.6 Part ............................................................ 134

**4.5 Programming Browser** ......................................... 135
  4.5.1 Overview ...................................................... 135
  4.5.2 Project ........................................................ 137
    4.5.2.1 Introduction .............................................. 137
    4.5.2.2 Project Settings ......................................... 138
  4.5.3 Controller ..................................................... 142
    4.5.3.1 Introduction .............................................. 142
    4.5.3.2 Controller Settings ...................................... 143
  4.5.4 Robot .......................................................... 148
    4.5.4.1 Introduction .............................................. 148
    4.5.4.2 Robot Settings ............................................ 149
  4.5.5 Tool ............................................................ 151
    4.5.5.1 Introduction .............................................. 151
    4.5.5.2 Edit Tool Signals ........................................ 152
  4.5.6 Feeders ........................................................ 153
  4.5.7 Feeder .......................................................... 154
    4.5.7.1 Introduction .............................................. 154
    4.5.7.2 Edit Feeder ................................................ 155
    4.5.7.3 Edit Detailed Signals for Feeder ....................... 157
    4.5.7.4 Safe Targets .............................................. 159
    4.5.7.5 Robot Path Height ....................................... 161
    4.5.7.6 Set WorkObject Position ................................ 162
  4.5.8 Operation Set ................................................ 163
    4.5.8.1 Introduction .............................................. 163
    4.5.8.2 Start the Operation Set Configuration .................. 164
    4.5.8.3 Group Operation Set ...................................... 165
    4.5.8.4 Pattern/Stack Operation Set .............................. 167
    4.5.8.5 Operation Editor .......................................... 179
  4.5.9 Flows ........................................................... 181
    4.5.9.1 Introduction .............................................. 181
    4.5.9.2 Add Flow .................................................. 182
  4.5.10 Flow ........................................................... 184
    4.5.10.1 Introduction .............................................. 184
  4.5.11 Job ............................................................. 185
    4.5.11.1 Introduction .............................................. 185

5 RAPID program ....................................................... 187
  5.1 Introduction ..................................................... 187
# Table of Contents

## 5.2 Overview

- 5.2.1 Relationship between RAPID execution and PickMaster project .......................... 188
- 5.2.2 RAPID modules overview ......................................................................................... 190

## 5.3 Program module templates

- 5.3.1 PmMain module ........................................................................................................ 191
- 5.3.2 PmUtility module ...................................................................................................... 196
- 5.3.3 PmProjMgr module .................................................................................................. 203
- 5.3.4 PmProjServer module ............................................................................................... 206

## 5.4 System modules

- 5.4.1 System modules, overview ....................................................................................... 211
- 5.4.2 Public system module pmrcUser ................................................................................. 212
- 5.4.3 Public system module pmrcSys .................................................................................. 213

## 6 Runtime operation

- 6.1 Introduction .................................................................................................................. 229

## 6.2 FlexPendant interface

- 6.2.1 Introduction to PickMaster FlexPendant interface .................................................. 230
- 6.2.2 Opening a project ..................................................................................................... 232
- 6.2.3 Starting and stopping production ............................................................................. 233
- 6.2.4 Flow recovery .......................................................................................................... 240
- 6.2.5 Process Signals ........................................................................................................ 246
- 6.2.6 Tuning ....................................................................................................................... 249

## 6.3 I/O interface

- 6.3.1 Overview .................................................................................................................. 253
- 6.3.2 Default signals ......................................................................................................... 254
- 6.3.3 Basic I/O interface .................................................................................................... 256
- 6.3.4 Extended I/O interface ............................................................................................. 265
- 6.3.5 Timing diagrams for PLC communication ............................................................... 272

## 7 RAPID reference information

- 7.1 Introduction to RAPID reference information ............................................................... 277
- 7.2 Instructions

  - 7.2.1 PmAckTarget - Acknowledge a target ...................................................................... 278
  - 7.2.2 PmCalcArmConf - Calculates the arm configuration .............................................. 280
  - 7.2.3 PmGetFlow - Get flow to execute ............................................................................ 283
  - 7.2.4 PmGetFlowInfo - Get information about a specific flow ....................................... 285
  - 7.2.5 PmGetLastWa - Get last used work area ................................................................. 287
  - 7.2.6 PmGetOperation - Get operation from a work area .............................................. 288
  - 7.2.7 PmGetProjectInfo - Get information about a specific project ............................... 290
  - 7.2.8 PmGetWaByWobj - Get a reference to a work area using a work object data ............ 292
  - 7.2.9 PmGetWaInfo - Get information about a specific work area ................................... 293
  - 7.2.10 PmSearchAdjust - Adjust number of remaining layers ......................................... 295
  - 7.2.11 PmSetLastWa - Set last used work area ................................................................. 297
  - 7.2.12 PmSetRecoverAction - Set flow recover action ..................................................... 298
  - 7.2.13 PmStartFlow - Starts a specific flow ..................................................................... 302
  - 7.2.14 PmStartProj - Start a PickMaster project .............................................................. 304
  - 7.2.15 PmStopFlow - Stop a specific flow ....................................................................... 305
  - 7.2.16 PmStopProj - Stop current project ........................................................................ 307
  - 7.2.17 PmWaitProjStart - Wait for any active project ...................................................... 308

## 7.3 Functions

- 7.3.1 PmCalcIntermid - Calculate intermediate position ............................................... 309
- 7.3.2 PmGetEvent - Get events for an action ..................................................................... 313
- 7.3.3 PmGetPathHeight - Get a safe path height for an intermediate movement .......... 315
## 7.4 Data types

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.4.1 pm_accdata - PickMaster acceleration/deceleration data</td>
<td>326</td>
</tr>
<tr>
<td>7.4.2 pm_acktype - PickMaster target acknowledge type</td>
<td>328</td>
</tr>
<tr>
<td>7.4.3 pm_actiondata - PickMaster action data</td>
<td>329</td>
</tr>
<tr>
<td>7.4.4 pm_actionhandle - PickMaster action handle</td>
<td>332</td>
</tr>
<tr>
<td>7.4.5 pm_actiontype - PickMaster action type</td>
<td>333</td>
</tr>
<tr>
<td>7.4.6 pm_eventdata - PickMaster event data</td>
<td>334</td>
</tr>
<tr>
<td>7.4.7 pm_eventtype - PickMaster event type</td>
<td>337</td>
</tr>
<tr>
<td>7.4.8 pm_flowinfo - PickMaster flow information</td>
<td>338</td>
</tr>
<tr>
<td>7.4.9 pm_moduleinfo - PickMaster module information</td>
<td>340</td>
</tr>
<tr>
<td>7.4.10 pm_movetype - PickMaster movement type</td>
<td>342</td>
</tr>
<tr>
<td>7.4.11 pm_offsetdata - Offset data</td>
<td>343</td>
</tr>
<tr>
<td>7.4.12 pm_operationdata - PickMaster operation data</td>
<td>345</td>
</tr>
<tr>
<td>7.4.13 pm_ophandle - PickMaster operation handle</td>
<td>347</td>
</tr>
<tr>
<td>7.4.14 pm_projectinfo - PickMaster project information</td>
<td>348</td>
</tr>
<tr>
<td>7.4.15 pm_robotinfo - PickMaster robot information</td>
<td>350</td>
</tr>
<tr>
<td>7.4.16 pm_searchdata - PickMaster search data</td>
<td>352</td>
</tr>
<tr>
<td>7.4.17 pm_searchtype - PickMaster stack search type</td>
<td>354</td>
</tr>
<tr>
<td>7.4.18 pm_singareatype - PickMaster interpolation type around singular points</td>
<td>355</td>
</tr>
<tr>
<td>7.4.19 pm_slavewainfo - PickMaster slave work area information</td>
<td>356</td>
</tr>
<tr>
<td>7.4.20 pm_stoptype - PickMaster stop type</td>
<td>358</td>
</tr>
<tr>
<td>7.4.21 pm_targetdata - PickMaster target data</td>
<td>359</td>
</tr>
<tr>
<td>7.4.22 pm_targethandle - PickMaster target handle</td>
<td>362</td>
</tr>
<tr>
<td>7.4.23 pm_wadescr - PickMaster work area reference</td>
<td>363</td>
</tr>
<tr>
<td>7.4.24 pm_wainfo - PickMaster Work Area information</td>
<td>364</td>
</tr>
</tbody>
</table>

## 8 Relationships between PickMaster frames

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1 Introduction</td>
<td>367</td>
</tr>
<tr>
<td>8.2 Shape frame</td>
<td>368</td>
</tr>
<tr>
<td>8.3 Format frame versus shape frame</td>
<td>369</td>
</tr>
<tr>
<td>8.4 Pallet pattern frame versus shape frame</td>
<td>373</td>
</tr>
<tr>
<td>8.5 Format frame versus tool frame</td>
<td>374</td>
</tr>
<tr>
<td>8.6 Pallet pattern item versus tool frame</td>
<td>376</td>
</tr>
<tr>
<td>8.7 Format frame versus work object in format operation set</td>
<td>377</td>
</tr>
<tr>
<td>8.8 Pallet pattern frame versus work object in pallet pattern operation set</td>
<td>379</td>
</tr>
<tr>
<td>8.9 Tune frame versus work area frame</td>
<td>381</td>
</tr>
<tr>
<td>8.10 Tune frame versus work area frame versus displacement frame</td>
<td>383</td>
</tr>
</tbody>
</table>

Index 385
Overview of the manual

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

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

Who should read this manual?
This manual is mainly intended for:
- System integrators
- ABB engineers
- End customers

Prerequisites
The reader should:
- Have experience with RobotStudio
- 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:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
<td>Introduces Palletizing PowerPac, and palletizing process and terms that are used in following chapters.</td>
</tr>
<tr>
<td>2. Installation</td>
<td>How to install the software and pre-requisites for installation.</td>
</tr>
<tr>
<td>4. Navigating Palletizing PowerPac</td>
<td>Describes in detail each function of Palletizing PowerPac.</td>
</tr>
<tr>
<td>5. RAPID Program</td>
<td>How RAPID programs are used in PickMaster. Program examples for palletizing applications.</td>
</tr>
<tr>
<td>6. Runtime Operation</td>
<td>How PickMaster palletizing projects are started, controlled, and supervised on a robot controller.</td>
</tr>
<tr>
<td>7. RAPID Reference Information</td>
<td>Descriptions of RAPID instructions, functions and data types, that are specific for PickMaster.</td>
</tr>
<tr>
<td>8. Relationships Between Frames</td>
<td>Describes the different frames used by PickMaster and how they relate to each other.</td>
</tr>
</tbody>
</table>

Continues on next page
Overview of the manual

Continued

References

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

Other references

<table>
<thead>
<tr>
<th>References</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.robotstudio.com/forum/">http://www.robotstudio.com/forum/</a></td>
<td>PickMaster Support Forum</td>
</tr>
</tbody>
</table>

Revisions

<table>
<thead>
<tr>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>First edition</td>
</tr>
</tbody>
</table>
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.
1 Introduction

1.1. Overview

Structure of this chapter

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

- The Palletizing cell concept
- A description of the palletizing process
- A terminology list containing PickMaster and specific palletizing terms
1 Introduction

1.2. What is Palletizing PowerPac

Overview

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

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

- 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.
- PzPP allows you to make the configuration and simulation for a palletizing application and process.

Prerequisites

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

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

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

The PowerPac contains:

- The Build Tool Function and Tool Events Configuration for configuring a robot tool model. See Create Tool on page 102.
- The Tool I/O Configuration for configuring the I/O connection between robot tool and controller. See Edit Tool Signals on page 152.
- The Build Feeder Configuration for configuring the feeder model properties, including hotspots. See Create Feeder on page 124.
- The Feeder Configuration for configuring the feeder properties, including signals, work objects, feeder CAD models, etc. See Feeder on page 154.
- The Product/Pallet/Sheet Configuration for configuring the product/pallet/sheet that will be picked and placed by the robot. See Product/Pallet/Sheet on page 51.
- The Pallet Pattern Configuration for configuring the stack of products organized in different layouts. See Pallet pattern on page 62.
- The Layout Editor for creating new or modifying existing layouts. See Layout Editor on page 65.
- The Pick Setting Configuration for configuring the products and tool orientation, as well as the tool functions to activate when gripping the products. See Pick Setting on page 68.
- The Operation Set Configuration for configuring the group operation set and pattern operation set. See Operation Set on page 163.
- The Job Wizard Configuration for configuring a new job including pattern/stack operation sets and/or group operation sets. See Add Job on page 73.
- The Flow configuration for defining accessible feeders in runtime. See Flow on page 184.
- The Tuning for changing parameter values online from the FlexPendant. See Tuning on page 249.
- The Project Settings for defining project name, descriptions, restart options and tune limitations. See Project Settings on page 138.
- The Feeder Order Settings for defining feeder order in the cell layout. See the Feeder Order Settings page in Controller Settings on page 143.
- The Event Settings for defining signals that can report events from external devices. See The Event Settings page in Controller Settings on page 143.
- The Message Settings for defining messages that can be reported from external devices. See The Message Settings page in Controller Settings on page 143.
- The Robot Settings for modifying motion limits and RAPID programs for each robot. See Robot Settings on page 149.
- The I/O Interface editor for assigning I/O values to products, product groups, pattern operation sets, feeders, flows, and projects. See I/O Interface on page 97.
- The Palletizing Library for saving products, layouts, I/O connections and messages. See Library on page 101.
- The Preview Palletizing function to go through each palletizing robot targets for user validation. See Preview Palletizing on page 80.

Continues on next page
1 Introduction

1.2. What is Palletizing PowerPac

Continued

The Simulation function to simulate the palletizing application. See Start Simulation on page 82.

The Download and Upload function to synchronize programs between RobotStudio and robot controllers. See Download on page 93 and Upload on page 95.
1.3. The palletizing process

Overview

A palletizing application aims at picking larger size objects from a fixed position and stacking them tightly together in a second fixed position. An important parameter for the palletizing process is the speed of the process, that is the throughput of products in time and the efficiency in stacking the products in a stable configuration without taking up too much space. After the palletizing process the stacks are loaded into containers or trucks, and the less space the products require, the less transportation costs are involved.

The palletizing cell

The figure illustrates an example of a palletizing cell.

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 175.

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

The palletizing cells

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

Continues on next page
1 Introduction

1.3. The palletizing process

Continued

How to pack the products

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

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

How to speed up the process

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

To plan each layout

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

Infeeders, outfeeders, and logical devices

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

The robot can move between different stacks

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

1.3. The palletizing process

The Flow concept

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

Execution of a palletizing job

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

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

Sequence of palletizing jobs

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

Parallel palletizing jobs

The flow concept allows a robot running multiple palletizing jobs in parallel, using one palletizing station for each job. The jobs run independently of each other but can share common resources, for example one infeeder can feed products to several parallel jobs. The robot can switch between different jobs after each pick-place cycle. If a job not is ready to run after a pick-place cycle (for example the next box to be picked cannot be fed since an error has occurred on the infeeder) the robot continues to work with the other jobs until the error has been resolved. In this way, the productivity of the robot can be maximized. Running parallel jobs does not add any complexity to programming or operator interaction.

Related information

Terms and concepts on page 18
1 Introduction

1.4. Terms and concepts

1.4. Terms and concepts

About these terms

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

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

Term list

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

Continues on next page
1 Introduction

1.4. Terms and concepts

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

Continues on next page
1 Introduction

1.4. Terms and concepts

Continued

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>A target describes a robot position used when performing an operation on a feeder. A target has a list of products carried by the robot tool and a list of actions that shall be executed. Note the difference with robot target: a target can have a list of actions, and each action has one robot target.</td>
</tr>
<tr>
<td>Tool Function</td>
<td>A single function of a tool that requires input signals to control and will generate results as output signals.</td>
</tr>
<tr>
<td>Tool Scenario</td>
<td>One grouping of tool functions to use in all actions of a target during picking or placing.</td>
</tr>
<tr>
<td>Work area</td>
<td>The original term used in PickMaster 5 PC to describe feeders, now it is only used in RAPID functions and variables.</td>
</tr>
<tr>
<td>Zone</td>
<td>A vacuum tool function is divided into a number of zones. Each zone has one or more activators.</td>
</tr>
</tbody>
</table>
1.5. Using a robot with 6 axes

Overview

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

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

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

- Axis 6 is pointing down. See Axis 6 points down on page 21.
- The wrist is tilted. See Wrist is tilted on page 22.
- The robot is bending backwards. See Robot is bending backwards on page 23.

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

Configuration of SingArea

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

Bending backwards robot

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

Axis 6 points down

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

1.5. Using a robot with 6 axes

Continued

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using SingArea \Wrist</td>
<td>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.</td>
</tr>
<tr>
<td>Using SingArea \Off</td>
<td>See Manual modifications on page 23.</td>
</tr>
</tbody>
</table>

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.
1 Introduction

1.5. Using a robot with 6 axes

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.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using SingArea \Wrist</td>
<td>Not recommended.</td>
</tr>
<tr>
<td>Using SingArea \Off</td>
<td>See Manual modifications on page 23.</td>
</tr>
</tbody>
</table>

Manual modifications

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

<table>
<thead>
<tr>
<th>Action</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Use ConfL \Off. Update the system module pmrcUser, with the procedure PmDoAction from system module pmrcSys and rename the procedure to DoAction. Modify DoAction according to description in section System module pmrcUser on page 24.</td>
<td>Linear motion supervision must be turned off to allow movements with large axis reorientations (&gt; 90°).</td>
</tr>
</tbody>
</table>
1 Introduction

1.5. Using a robot with 6 axes

Continued

<table>
<thead>
<tr>
<th>Action</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Use a robot position for each work area with MoveJ.</td>
<td>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.</td>
</tr>
<tr>
<td>In PzPP browser, record a safe position for each feeder, and use MoveJ for each position. See Safe Targets Set WorkObject Position on page 162.</td>
<td></td>
</tr>
<tr>
<td>3. Use ConfJ \On.</td>
<td>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.</td>
</tr>
<tr>
<td>Turn on the configuration control for joint movements.</td>
<td></td>
</tr>
<tr>
<td>Update the system module pmrcUser,</td>
<td></td>
</tr>
<tr>
<td>with the procedure PmDoAction from system module pmrcSys, and rename the procedure to DoAction. Modify DoAction according to description in section System module pmrcUser on page 24.</td>
<td></td>
</tr>
<tr>
<td>You find the complete program code for the system modules pmrcUser and pmrcSys in section Program code on page 213.</td>
<td></td>
</tr>
</tbody>
</table>

System module pmrcUser

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

... 
IF Act.ArmConfMon = TRUE THEN
   ConfL\Off;
   ConfJ\On;
ELSE
   ConfL\Off;
   ConfJ\On;
ENDIF
... 
You find the complete program code for the system modules pmrcUser and pmrcSys in section Program code on page 213.
1.5. Using a robot with 6 axes

**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 21* and *Wrist is tilted on page 22* are possible.

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

**RAPID calls to PMCalcArmConf**

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

**pmMain.mod**

Modify the program module pmMain.mod, procedure Operate:

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

**Related information**

*Group Operation Set on page 165.*
*Pattern/Stack Operation Set on page 167.*
*Public system module pmrcUser on page 212.*
1 Introduction

1.5. Using a robot with 6 axes
# 2 Installation

## 2.1. Installation

### Overview

This section describes the installation process.

### Prerequisites

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

- A computer that meets or exceeds the system requirements.
- A log on account with administrator rights on the computer.
- RobotStudio installed on the computer.
- PzPP installation package.

### System requirements

High performance desktop or laptop workstation:

- CPU: 2.0 GHz or faster processor
- Memory: 1 GB system memory at minimum, 2 GB if running Windows Vista, stations with several robot systems, or large CAD-models.
- Free disk-space: 5+ GB free space
- Graphics card: High performance DirectX 9 or OpenGL-compatible graphics card with the corresponding up-to-date drivers installed
- Display settings: Screen resolution: 1280 x 1024 pixels or higher
- DPI: Normal size (96 dpi)

### Network setting

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

**Ethernet network**

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

**New local area network**

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

<table>
<thead>
<tr>
<th>IP addresses</th>
<th>192.168.1.X (where X is between 1 and 253)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gateway</td>
<td>192.168.1.254</td>
</tr>
<tr>
<td>Subnet mask</td>
<td>255.255.255.0</td>
</tr>
</tbody>
</table>

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

*Continues on next page*
2 Installation

2.1. Installation

Continued

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

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

Installing Palletizing PowerPac

To install the PzPP, follow these steps:

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

Installing a license

To install a license of RobotStudio and PowerPac, see Operating manual - RobotStudio.
3 Workflow for Palletizing PowerPac with PickMaster

3.1. Introduction

The following is a recommended working procedure for PzPP with PickMaster.
3 Workflow for Palletizing PowerPac with PickMaster

3.2. Preparing your controller for PickMaster

3.2. Preparing your controller for PickMaster

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

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

Preparing your controller for PickMaster
Use this procedure to prepare the controller for PickMaster:

<table>
<thead>
<tr>
<th>Action</th>
<th>Note/See</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Create and install a system for the robot controller using RobotStudio.</td>
<td>Operating manual - RobotStudio.</td>
</tr>
<tr>
<td>2 Setup RAPID tool data for the robot tool. Use RobotStudio or the FlexPendant.</td>
<td>The tool data will be created and downloaded in PzPP when certain tool is attached to the virtual robot. It is often convenient to define one tool data for picking/placing and another one to be used when calibrating work objects. If the robot tool supports stack search a separate tool data is created for that purpose. Technical reference manual - RAPID Instructions, Functions and Data types, section tooldata - Tool data.</td>
</tr>
<tr>
<td>3 Calibrate the work object for each feeder. Use the FlexPendant.</td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section wobjdata - Work object data. Operating manual - IRC5 with FlexPendant.</td>
</tr>
<tr>
<td>4 Add tool related signals. Use RobotStudio or the FlexPendant.</td>
<td>See Edit Tool Signals on page 152 for information about the signals. Operating manual - RobotStudio. Operating manual - IRC5 with FlexPendant.</td>
</tr>
<tr>
<td>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.</td>
<td>The Controller Settings window. Operating manual - RobotStudio. Operating manual - IRC5 with FlexPendant.</td>
</tr>
</tbody>
</table>
3.3. How to add tool

On the Palletizing ribbon-tab, click Add Tool to import a tool for the palletizing process.
See Add tool on page 47 for detailed description.
3.4. How to add feeders

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

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

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

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

See *Pallet pattern on page 62* for detailed description.
3.6. How to set pick setting

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

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

**NOTE!**

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

On the Palletizing ribbon-tab, click *Add a Job* to create a job for palletizing process. See *Add Job on page 73* for detailed description.
3.8. How to do simulation

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

See Start Simulation on page 82 for detailed description.
3.9. How to adjust cell layout to achieve reachability

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

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

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

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

Choose “Yes” to update Task Frame, otherwise, the controller needs to be warm started. Then a further message box will ask you to keep the positioning of all stationary RAPID variables (e.g. work object):

Choose “Yes” to keep all work object values unchanged, otherwise the work objects will move together with the robot.
3 Workflow for Palletizing PowerPac with PickMaster

3.10. How to transfer the project

3.10. How to transfer the project

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

See *Add Controller on page 92* and *Download on page 93* for detailed description.
3.11. How to start production

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

Starting production
Use this procedure to start a PickMaster project.

<table>
<thead>
<tr>
<th>Action</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Start the PickMaster FlexPendant interface.</td>
<td></td>
</tr>
<tr>
<td>2. Select the project you want to run.</td>
<td>Opening a project on page 232.</td>
</tr>
<tr>
<td>3. Start the project.</td>
<td>Starting a project on page 233.</td>
</tr>
<tr>
<td>4. Start flows.</td>
<td>Starting a specific flow on page 234 and</td>
</tr>
<tr>
<td></td>
<td>Starting and stopping all flows on page</td>
</tr>
<tr>
<td></td>
<td>236.</td>
</tr>
</tbody>
</table>

Related information

*FlexPendant interface on page 230.*
*Opening a project on page 232.*
*Starting and stopping production on page 233.*
3.12. How to send my project to get support from ABB

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

4 Navigating Palletizing PowerPac
4 Navigating Palletizing PowerPac

4.1. Introduction

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

Overview
Before you can start using PzPP, you must load a RobotStudio station.
In a station without system and robot, you are able to edit products, pallet patterns, create custom grippers and feeders, and edit libraries.
In a station with a system and at least one robot, you can continue to add tools and feeders to the robots, to configure flows and pick/place operation sets on feeders. The virtual controller (VC) associated with the robot must be loaded with Prepare for PickMaster option and PickMaster 5 sub-option.

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

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

The User Interface
The panes and windows of the user interface, described in the following figure, help you create a well-structured palletizing program.
### 4 Navigating Palletizing PowerPac

#### 4.2. General

*Continued*

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

4.3.1. Overview

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

Elements on the Palletizing ribbon-tab

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

Continues on next page
### 4.3.1. Overview

<table>
<thead>
<tr>
<th>Group</th>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation</td>
<td>Quick Start</td>
<td>Transfer program to virtual controller, open operator interface and start the simulation with the default jobs in one click. See <em>Operator Interface on page 83</em> for detailed description.</td>
</tr>
<tr>
<td></td>
<td>Operator Interface</td>
<td>Transfer program to virtual controller, open operator interface. Then start the simulation for the specified jobs manually. See <em>Operator Interface on page 83</em> for detailed description.</td>
</tr>
<tr>
<td></td>
<td>Record Simulation as Viewer</td>
<td>Select this option to save the simulation as viewer file after the simulation is stopped. See <em>Record as Viewer File on page 87</em> for detailed description.</td>
</tr>
<tr>
<td></td>
<td>Speed Mode</td>
<td>Set simulation speed and there are three modes for it, such as Full, Customized and Low. See <em>Speed Mode on page 86</em> for detailed description.</td>
</tr>
<tr>
<td>Transfer</td>
<td>Add Controller</td>
<td>Connect a controller with this station using RobotStudio Online functionality.</td>
</tr>
<tr>
<td></td>
<td>Download</td>
<td>Download the program to connected controllers. See <em>Download on page 93</em> to Controllers for detailed description.</td>
</tr>
<tr>
<td></td>
<td>Upload</td>
<td>Upload the program from the connected controller. See <em>Upload on page 95</em> from Controllers for detailed description.</td>
</tr>
<tr>
<td>Project</td>
<td>Overview</td>
<td>Show the overview information for all operation sets in the project. See <em>Project Overview on page 96</em> for detailed description.</td>
</tr>
<tr>
<td></td>
<td>I/O Interface</td>
<td>Show the I/O information for this project. See <em>I/O Interface on page 97</em> for detailed description.</td>
</tr>
<tr>
<td></td>
<td>Report</td>
<td>Generate a report for this project. See <em>Project Report on page 100</em> for detailed description.</td>
</tr>
<tr>
<td>Advanced</td>
<td>Library</td>
<td>Edit the default libraries for reuse. See <em>Library on page 101</em> for detailed description.</td>
</tr>
<tr>
<td></td>
<td>Cell Models</td>
<td>Guide user to design customized tool and feeder. See <em>Create Tool on page 102</em> and <em>Create Feeder on page 124</em> for detailed description.</td>
</tr>
<tr>
<td>3D Tools</td>
<td>View angles</td>
<td>Adjust the view point for the station from top, left and front.</td>
</tr>
<tr>
<td></td>
<td>Freehand move</td>
<td>Enable to adjust the position/orientation for the selected objects by manual dragging in 3D view.</td>
</tr>
<tr>
<td>Help</td>
<td>Help</td>
<td>Provide the user manual of PzPP.</td>
</tr>
<tr>
<td></td>
<td>About</td>
<td>Provide the general information about PzPP.</td>
</tr>
</tbody>
</table>
4.3.2. Add tool

**Overview**

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

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

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

**NOTE!**

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

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller</td>
<td>Select a controller in which the robot to attach the tool belongs to.</td>
</tr>
<tr>
<td>Robot</td>
<td>Select a robot which will hold the tool.</td>
</tr>
<tr>
<td>Signal Configuration</td>
<td>Provide the match information about signals connection between the tool and the virtual controller. If the signal connection is not valid, the status light will be red, and you can check and modify with button &quot;Edit Signals&quot;. See <em>Edit Detailed Signals for Feeder on page 157</em> for detailed description.</td>
</tr>
</tbody>
</table>
4 Navigating Palletizing PowerPac

4.3.2. Add tool

Continued

Use this procedure to edit parameters in the dialog box:

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

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

Overview

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

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

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

**NOTE!**

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

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller</td>
<td>Select a controller in which the robot to work with the feeder belongs to.</td>
</tr>
<tr>
<td>Robot</td>
<td>Select a robot in which the feeder is added.</td>
</tr>
<tr>
<td>Signal Configuration</td>
<td>Provide the match information about signals connection between the feeder and the virtual controller. If the signal connection is not valid, the status light will be red, and you can check and modify with button &quot;Edit Signals&quot;. See <em>Edit Detailed Signals for Feeder on page 157</em> for detailed description.</td>
</tr>
</tbody>
</table>
4 Navigating Palletizing PowerPac

4.3.3. Add feeder

Continued

Use this procedure to edit parameters in the dialog box:

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

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

Overview

It is to input customer’s products.

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

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

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

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

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

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

NOTE!

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

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

The interface looks as following:
After one object is selected in the item list, its properties will be shown on the right UI.

### Proceed with the Box Properties

This section describes how to edit the box properties.

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

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

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

**Proceed with the Bag Properties**

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

Proceed with the Pallet Properties

This section describes how to edit the pallet properties.

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

Continues on next page
4 Navigating Palletizing PowerPac

4.3.4. Product/Pallet/Sheet

Proceed with the Sheet Properties

This section describes how to edit the Sheet properties.

<table>
<thead>
<tr>
<th>To...</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Pallet Facing</td>
<td>Same as Box</td>
</tr>
<tr>
<td>Save the Pallet</td>
<td>Same as Box</td>
</tr>
<tr>
<td>Edit Advanced Settings for Pallet</td>
<td>Same as Box</td>
</tr>
<tr>
<td>Show a Pallet</td>
<td>Same as Box</td>
</tr>
</tbody>
</table>

This section describes how to edit the Sheet properties.

<table>
<thead>
<tr>
<th>To...</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rename a Sheet</td>
<td>Same as Box</td>
</tr>
<tr>
<td>Set I/O value</td>
<td>Same as Box</td>
</tr>
<tr>
<td>Select Sheet from Library</td>
<td>Same as Box</td>
</tr>
<tr>
<td>View the Item Type</td>
<td>Same as Box</td>
</tr>
<tr>
<td>Set Sheet Size</td>
<td>Same as Box</td>
</tr>
<tr>
<td>Set Sheet Weight</td>
<td>Same as Box</td>
</tr>
<tr>
<td>Set Sheet Facing</td>
<td>Same as Box</td>
</tr>
<tr>
<td>Save the Sheet</td>
<td>Same as Box</td>
</tr>
<tr>
<td>Edit Advanced Settings for Sheet</td>
<td>Same as Box</td>
</tr>
<tr>
<td>Show a Sheet</td>
<td>Same as Box</td>
</tr>
</tbody>
</table>
Proceed with the Group Properties

This section describes how to edit the Group properties.

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

This section describes how to edit the stack properties.

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

Advanced setting for Item

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

Continues on next page
4 Navigating Palletizing PowerPac

4.3.4. Product/Pallet/Sheet

Continued

Proceed with the General

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

![CAD Settings Diagram]

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

Continues on next page
Proceed with the Pick/Place

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

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

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

4.3.4. Product/Pallet/Sheet

Continued

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.

![Diagram of robot speed over time with tool activation and pick time]

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

![Diagram of robot speed over time with tool deactivation and place time]
Proceed with the Motion Limits

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

**To...**

<table>
<thead>
<tr>
<th>Set Maximum Speed</th>
<th>In the Speed number box, type the maximum allowed speed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Acceleration/Deceleration</td>
<td>In the Acc/Dec number box, type the acceleration value (in mm/s). The same value will be used for both acceleration and deceleration.</td>
</tr>
<tr>
<td>Set Maximum Rotation Speed</td>
<td>In the Rot Speed number box, type the maximum allowed rotation speed (in deg/s).</td>
</tr>
</tbody>
</table>

**NOTE!**

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

4.3.5. Pallet pattern

4.3.5.1. Introduction

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Pattern</td>
<td>Create a new pallet pattern with items created before. See <em>Proceed with the Pallet Pattern Layout on page 63</em> for detailed description.</td>
</tr>
<tr>
<td>Delete</td>
<td>Delete the selected pallet pattern in the pattern list.</td>
</tr>
<tr>
<td>Export</td>
<td>Export the existing pallet patterns into XML file.</td>
</tr>
<tr>
<td>Import</td>
<td>Import and create pallet patterns from XML file. The used items in the XML file will also be created in station.</td>
</tr>
</tbody>
</table>
After one pallet pattern is selected in the pattern list, its corresponding properties will be shown in the right.

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

**Proceed with the Pallet Pattern Layout**

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

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

**NOTE!**

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

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

4.3.5.1. Introduction

Continued

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

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

Proceed with the Pallet Pattern Layers

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

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

Overview

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

Start the Layout Editor

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

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

Start from the Library

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
</tr>
<tr>
<td>2.</td>
</tr>
<tr>
<td>3.</td>
</tr>
</tbody>
</table>

Start from a pallet pattern configuration

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
</tr>
<tr>
<td>2.</td>
</tr>
<tr>
<td>3.</td>
</tr>
<tr>
<td>4.</td>
</tr>
</tbody>
</table>
4 Navigating Palletizing PowerPac

4.3.5.2. Layout Editor

Continued

**Illustration, Layout Editor**

![Layout Editor](xx110000023)

<table>
<thead>
<tr>
<th><strong>Item</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Layout</td>
<td>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.</td>
</tr>
<tr>
<td>Item</td>
<td>The drop-down combo box shows the selected item. If you add a new item, it will be added to the layout. If there are no items in the layout, you select which item to add. When you edit a library layout you can change the selected item to any of the available items in the library. When you edit a layout in a pallet pattern, you cannot change the item.</td>
</tr>
<tr>
<td>Display</td>
<td>Shows the layout of a layer. Here you modify the layout by using a drag-and-drop operation or the buttons on the right side. The Show Overlap check box provides an option to show the overlapping boxes in red.</td>
</tr>
<tr>
<td>Selection</td>
<td>Shows the position and orientation of the selected item. You can adjust the position by editing the values in the text boxes, or by using CTRL + arrow keys.</td>
</tr>
<tr>
<td>Layout information</td>
<td>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.</td>
</tr>
</tbody>
</table>
4 Navigating Palletizing PowerPac

4.3.5.2. Layout Editor

Continued

Proceed with the item

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

Proceed with the layout

<table>
<thead>
<tr>
<th>To...</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mirror the layout in x direction</td>
<td>Click Flip Horizontal.</td>
</tr>
<tr>
<td>Mirror the layout in y direction</td>
<td>Click Flip Vertical.</td>
</tr>
<tr>
<td>Rotate the entire layout 180°</td>
<td>Click Rotate.</td>
</tr>
<tr>
<td>Center the layout</td>
<td>Click Center.</td>
</tr>
</tbody>
</table>

Align and distribute

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

TIP!

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

Related information

- Library on page 101
- Pallet pattern on page 62
- Terms and concepts on page 18
4.3.6. Pick Setting

Overview

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

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

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

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

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

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

4.3.6. Pick Setting

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

<table>
<thead>
<tr>
<th>Tool Location</th>
<th>Item - Tool Function Match</th>
<th>Additional Events</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

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

**Proceed with the Tool Additional Events**

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

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

![Additional Events Illustration](illustration.png)

<table>
<thead>
<tr>
<th>Type</th>
<th>Signal Name</th>
<th>Value</th>
<th>Pass Time</th>
<th>Movement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set</td>
<td>G0_Gripper</td>
<td>2</td>
<td>0</td>
<td>PickFirstApproach</td>
</tr>
<tr>
<td>Wait</td>
<td>G0_GripperOpen</td>
<td>1</td>
<td>0</td>
<td>PickFirstApproach</td>
</tr>
<tr>
<td>Set</td>
<td>G0_Gripper</td>
<td>1</td>
<td>0</td>
<td>Pick</td>
</tr>
<tr>
<td>Wait</td>
<td>G0_GripperClosed</td>
<td>1</td>
<td>0</td>
<td>Pick</td>
</tr>
<tr>
<td>Set</td>
<td>G0_Gripper</td>
<td>1</td>
<td>0</td>
<td>PickLastDepart</td>
</tr>
<tr>
<td>Wait</td>
<td>G0_GripperClosed</td>
<td>1</td>
<td>0</td>
<td>PickLastDepart</td>
</tr>
</tbody>
</table>

Type of the event:
- **?Set**: set an output signal to a value at a specific movement.
- **?Wait**: check the value of an input signal that indicates the actual status of the tool. In this example used for checking that the desired action indicated by the group output signal has been achieved.

**Signal Name**
The signal that is used in the event.
Continued

<table>
<thead>
<tr>
<th>Preset Time</th>
<th>The time in advance for the out signal to set to a value (only used for Set output type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movement</td>
<td>Describes when event occurs, i.e., first approach action, last approach action, target action and first depart action, or last depart action.</td>
</tr>
</tbody>
</table>
4.3.7. Add Job

Overview

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

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

<table>
<thead>
<tr>
<th>Page</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Match Feeders with Products/Pattern for Picking/Placing</td>
</tr>
<tr>
<td>2</td>
<td>Configure Operation Set for Group/Stack</td>
</tr>
<tr>
<td>3</td>
<td>Configure Operation Set for Pallet Pattern</td>
</tr>
</tbody>
</table>

Select a controller, robot, job type, specify a pallet pattern on a master feeder and select necessary groups/stacks on the other feeders.

Configure the operation set for selected groups/stacks in detail.

Configure the operation set for selected pallet pattern in detail.

Distribute roles for feeders

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

Continues on next page
### 4 Navigating Palletizing PowerPac

#### 4.3.7. Add Job

*Continued*

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

4.3.7. Add Job

Click Next button to continue to the next page.

TIP!

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

Configure operation set for group or stack

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

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

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

Continues on next page
4 Navigating Palletizing PowerPac

4.3.7. Add Job

Continued

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

Click Next button to continue to the next page.

**Configure operation set for pallet pattern**

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

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

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

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

Overview

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

**NOTE!**

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

Check Reachability

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

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

<table>
<thead>
<tr>
<th>Checked Program</th>
<th>Source</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pick Box1Group1F</td>
<td>InFeeder1</td>
<td>OperationSet</td>
</tr>
<tr>
<td>Place PalletPattern</td>
<td>OutFeeder1</td>
<td>OperationSet</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>To...</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check Selected Object</td>
<td>Select an object in the check list and click Check button, or double click the object, the robot targets contained by the object will be checked. The light status will change its color according to the check result. Green means that the robot targets can be reached while red means that some of robot targets cannot be reached.</td>
</tr>
</tbody>
</table>
### 4.3.8. Check Reach

*Continued*

<table>
<thead>
<tr>
<th>To...</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check All Robot Targets</td>
<td>Click Check All button, all objects in the list will be checked.</td>
</tr>
<tr>
<td>Show Unreachable Products</td>
<td>If some objects contain unreachable targets after checking, click on the object and the unreachable items in the selected object will turn red in main 3D view. Information of unreachable targets will also be shown in RobotStudio output window.</td>
</tr>
</tbody>
</table>
4 Navigating Palletizing PowerPac

4.3.9. Preview Palletizing

Overview

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

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

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

Proceed with the Preview Panel

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

**To...**

**Do this**

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

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

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

Overview

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

The project status is shown on top of the interface:

<table>
<thead>
<tr>
<th>Project status</th>
<th>Description</th>
<th>Status Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>Started</td>
<td>The project has been started successfully.</td>
<td>xx110000044</td>
</tr>
<tr>
<td>Starting</td>
<td>The project is in the starting process.</td>
<td>xx110000045</td>
</tr>
<tr>
<td>Stopped</td>
<td>The project has been stopped.</td>
<td>xx110000046</td>
</tr>
<tr>
<td>Error</td>
<td>Error occurs during the project execution.</td>
<td>xx110000047</td>
</tr>
</tbody>
</table>

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

Control Page

Control the starting and stopping of flows, and the job sequences for each flow
4 Navigating Palletizing PowerPac

4.3.11. Operator Interface

Continued

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

<table>
<thead>
<tr>
<th>Project status</th>
<th>Description</th>
<th>Status Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running</td>
<td>The flow is running.</td>
<td>xx110000044</td>
</tr>
<tr>
<td>Stopping</td>
<td>The flow is in the stopping process.</td>
<td>xx110000045</td>
</tr>
<tr>
<td>Stopped</td>
<td>The flow has been stopped.</td>
<td>xx110000046</td>
</tr>
<tr>
<td>Error</td>
<td>Error occurs during the flow execution.</td>
<td>xx110000047</td>
</tr>
</tbody>
</table>

Proceed with the Flow Control

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

<table>
<thead>
<tr>
<th>To...</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start a Flow</td>
<td>Select one flow in the Controllers and Flows group whose status is Stopped, click xx110000044 button to start the selected flow. If the selected flow is to be restarted, a flow recover option can be selected to specify how and when the flow must be restarted. If the status of the selected flow is not Stopped, this button will be disabled.</td>
</tr>
<tr>
<td>Stop a Flow</td>
<td>Select one flow in the Controllers and Flows group whose status is Running, click xx110000045 button to stop the selected flow. A flow stop option can be selected to specify how and when the flow must be stopped. If the status of the selected flow is not Running, this button will be disabled.</td>
</tr>
</tbody>
</table>
### 4 Navigating Palletizing PowerPac

#### 4.3.11. Operator Interface

**To...** | **Do this**
---|---
Start All Flow(s) | Click button to start all flows. If some of flows are to be restarted, a flow recover option can be selected to specify how and when these flows must be restarted. This button is enabled when the status of some of the flows are Stopped.

Stop All Flow(s) | Click button to stop all flows. A flow stop option can be selected to specify how and when the flow must be stopped. This button is enabled when the status of some of the flows are Running.

---

**Proceed with the Scheduled Jobs**

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

**To...** | **Do this**
---|---
Add Job | In the Scheduled Jobs group, click button and a dialog will pop up. The dialog shows all supported jobs of the flow. Select one job and click OK button, the selected job will be added into the list.

Delete a Job | In the Scheduled Jobs group, select one job and click button, the selected job will be deleted from the list.

Reorder Jobs | In the Scheduled Jobs group, select one job and click the Up or Down button to reorder the job in the list.

Repeat the Jobs | Select Run Continuously check box, the jobs in the list will be executed repeatedly.
Statistics Page displays the statistics information for the palletizing process, including cycle time, throughputs, and certain event logs.

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation Time</td>
<td>Display the simulation time in second.</td>
</tr>
<tr>
<td>Product Data</td>
<td>Display how many products are inputted and outputted and the calculated data of throughput (based on outputted products and cycle time).</td>
</tr>
<tr>
<td>Simulation Event Log</td>
<td>Display the main event log for palletizing process.</td>
</tr>
</tbody>
</table>
4.3.12. Record as Viewer File

Check this option, to save the whole simulation as RobotStudio Viewer file after the simulation is stopped.
There are three speed modes for the simulation process: Full, Customized and Low.
You can check and change the speed mode, before the simulation is started. Default mode is Low.
Note that the specified speed is effective when robot is moving without products. If robot is moving with products, the speed is limited by the motion limits of the product itself.
4.3.14. Stop Simulation

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

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

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

If the simulation is started, the Pause and Step button will be enabled.
Click the button, the simulation will be paused. You can then step forward the simulation by continuously clicking the button.
You can click the Start button to change back and make the simulation run by itself.
4.3.16. Reset Simulation

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

Click Reset button, the simulation environment will be reset, the products created during the simulation will be cleaned, and the station is ready for next simulation.
There are two ways to connect to a controller:

- **Add Controller** - For adding available controllers to the network
- **One Click Connect** - For connecting to the service port of the controller

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

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

The download includes:

- Synchronizing work object and tool data values, if downloading to virtual controller(s)
- Checking the completeness of project setups
- Generating and transferring the configuration files
- Generating and transferring the tune file to Pickware

![Download Interface](image)

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

Download Program to Virtual Controller | Select Download to Virtual Controller check box, and click Download button to download the program to virtual controller(s).

*Continues on next page*
4 Navigating Palletizing PowerPac

4.3.18. Download

Continued

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

**NOTE!**

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

**NOTE!**

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

**NOTE!**

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

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

The upload includes:

- Reading the work object and tool data values from controller: feeder positions and tool function positions in the project will be updated accordingly.
- Reading the tune value from Pickware: the product sizes (the graphical model), feeder tune offsets (thus operation set positions on the feeders) will be updated accordingly.

### Online Controller Info

<table>
<thead>
<tr>
<th>ID</th>
<th>61ce5257-5a53-4f2b-b735-21f10c110504</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Name</td>
<td>System20</td>
</tr>
<tr>
<td>Controller Name</td>
<td>ONL-6051.018</td>
</tr>
<tr>
<td>RW/Version</td>
<td>5.14.0.0</td>
</tr>
<tr>
<td>Available</td>
<td>True</td>
</tr>
<tr>
<td>Virtual</td>
<td>True</td>
</tr>
<tr>
<td>IP</td>
<td>127.0.0.1</td>
</tr>
</tbody>
</table>

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

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

The upload includes:

- Reading the work object and tool data values from controller: feeder positions and tool function positions in the project will be updated accordingly.
- Reading the tune value from Pickware: the product sizes (the graphical model), feeder tune offsets (thus operation set positions on the feeders) will be updated accordingly.

### Online Controller Info

<table>
<thead>
<tr>
<th>ID</th>
<th>61ce5257-5a53-4f2b-b735-21f10c110504</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Name</td>
<td>System20</td>
</tr>
<tr>
<td>Controller Name</td>
<td>ONL-6051.018</td>
</tr>
<tr>
<td>RW/Version</td>
<td>5.14.0.0</td>
</tr>
<tr>
<td>Available</td>
<td>True</td>
</tr>
<tr>
<td>Virtual</td>
<td>True</td>
</tr>
<tr>
<td>IP</td>
<td>127.0.0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>To...</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload Program from Online Controller</td>
<td>Unselect Upload from Virtual Controller check box. After the online controller(s) is added as described above, the available online controller(s) will be available to select in the table’s “Online Controller” column. Match each virtual controller with an online controller, and click Upload button to upload data from the available online controller(s).</td>
</tr>
<tr>
<td>Upload Program from Virtual Controller</td>
<td>Select Upload from Virtual Controller check box, and click Upload button to upload the program from virtual controller(s).</td>
</tr>
</tbody>
</table>
Click *Overview* button in ribbon to display the overview information for the project.

The dialog shows each configured operation set and its corresponding product and group I/O values. You can also save this information into .txt file by clicking the “Save As Report” button.
Click **I/O Interface** button in ribbon to display the I/O information for the project. The interface is a one-stop place to view all information related with PLC for synchronizing with robot for palletizing process.

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

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

I/O values

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

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

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

Continues on next page
4 Navigating Palletizing PowerPac

4.3.21. I/O Interface

Continued

Project Manager

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Object</th>
<th>Type</th>
<th>Usage/Purpose</th>
<th>Signal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Palletizing</td>
<td>Error Tag</td>
<td>pmTrigger</td>
<td></td>
</tr>
<tr>
<td>System Palletizing</td>
<td>Error Source</td>
<td>pmTriggerSource</td>
<td></td>
</tr>
<tr>
<td>System Palletizing</td>
<td>Error Message</td>
<td>pmTriggerMessage</td>
<td></td>
</tr>
<tr>
<td>Vacuum taker</td>
<td>Tool</td>
<td>gS1Xtaker</td>
<td>pmTrigger_LgPallets</td>
</tr>
<tr>
<td>Vacuum taker</td>
<td>Tool</td>
<td>gS1Xtaker2</td>
<td>pmTrigger_RgPallett</td>
</tr>
<tr>
<td>Vacuum taker</td>
<td>Tool</td>
<td>gS1Xtaker3</td>
<td>pmTrigger_DgPallett</td>
</tr>
<tr>
<td>Vacuum taker</td>
<td>Tool</td>
<td>gS1Xtaker4</td>
<td>pmTrigger_FgPallett</td>
</tr>
<tr>
<td>Vacuum taker</td>
<td>Tool</td>
<td>gS1Xtaker5</td>
<td>pmTrigger_RgPallett1</td>
</tr>
<tr>
<td>Vacuum taker</td>
<td>Tool</td>
<td>gS1Xtaker6</td>
<td>pmTrigger_DgPallett1</td>
</tr>
<tr>
<td>Floate</td>
<td>Status</td>
<td>pmFloat1</td>
<td></td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Position Request Trigger</td>
<td>pmInfEedr1_InfEedr1_Request</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Product Selection (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedR</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Form Selection (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Requesting Master (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedReq</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Target Generation trigger (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedTg2</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Product Selection (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Form Selection (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Start Layer Count (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Start Product Count (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Position Available (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Operation Set Complete (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Execute (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Execute State (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Height (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Layer Count (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Product Count (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Stop Job (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Ready Search (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>InfEedr1</td>
<td>InfEedr1</td>
<td>Robot Execution (ISO)</td>
<td>pmInfEedr1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>OutFeeder1</td>
<td>OutFeeder1</td>
<td>Position Request Trigger (ISO)</td>
<td>pmOutFeeder1_InfEedr1_Request</td>
</tr>
<tr>
<td>OutFeeder1</td>
<td>OutFeeder1</td>
<td>Product Selection (ISO)</td>
<td>pmOutFeeder1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>OutFeeder1</td>
<td>OutFeeder1</td>
<td>Form Selection (ISO)</td>
<td>pmOutFeeder1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>OutFeeder1</td>
<td>OutFeeder1</td>
<td>Requesting Master (ISO)</td>
<td>pmOutFeeder1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>OutFeeder1</td>
<td>OutFeeder1</td>
<td>Target Generation Trigger (ISO)</td>
<td>pmOutFeeder1_InfEedr1_FeedSel</td>
</tr>
<tr>
<td>OutFeeder1</td>
<td>OutFeeder1</td>
<td>Product Selection (ISO)</td>
<td>pmOutFeeder1_InfEedr1_FeedSel</td>
</tr>
</tbody>
</table>
Click Report button in ribbon to display the process report for the project.

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

You can save the report as a PDF or Excel file.
Click Library button in ribbon to display the libraries used in the project.

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

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

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

**User directory:** “…\My Documents\RobotStudio\Palletizing\Library\”

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

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

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

4.3.24.1. Overview

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

A PzPP compatible tool includes properties of:

- Tool weight, center of gravity, and inertia.
- I/O connection template name: the I/O template that can be used as default for the tool. This information is used when the tool is attached to a robot. See Edit Tool Signals on page 152 for detailed information.
- Tool function: A tool should contain at least one tool function. A tool function is a modular function of the tool that is controlled by certain I/O and optionally containing certain behaviors.

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

1. Pallet pick function: a type of claw that is controlled by 2 DI signals for opening and closing, and and contains 2 DO signals to tell the opened and closed status. It is usually used to pick up a pallet.

2. Vacuum function: the vacuum is usually controlled by a GI signal to open certain groups of suction cups and close other groups of suction cups. It is usually used to pick boxes.

3. Searcher: the searcher is usually controlled by 2 DI signals for opening and closing, and and contains 2 DO signals to tell the opened and closed status. It is usually used to detect the actual height of a pallet stack. When the searcher is totally opened, and during the downward movement touches the pallet, one of the DO signal value is changed so robot will know the actually height of the pallet.

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

The following sections describe:

- How to edit tool weight, I/O template, tool data
- How to add/edit/remove different types of tool functions, and the related tool data

Continues on next page
General tab

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

There are currently 6 types of tool functions defined in PzPP. Click the button, and you can see:

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

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

The editing UI for Vacuum is as following:

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>You can change the tool function name, choose the tool data, and input a control signal name for tool SmartComponent.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Vacuum1</td>
<td>ToolData: Vacuum</td>
</tr>
<tr>
<td>Control Signal (GI)</td>
<td>gSCToolActivators</td>
</tr>
</tbody>
</table>

Note: The control signal is for the tool SmartComponent, and is not a robot controller signal.
4 Navigating Palletizing PowerPac

4.3.24.2. Tool Functions

Continued

### Activators

An activator is a physical correspondent to the control signal, and controls one part of the tool, for example a vacuum cup. The figure shows a configuration example.

<table>
<thead>
<tr>
<th>Activator</th>
<th>Start Bit</th>
<th>Bit Length</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Act1</td>
<td>0</td>
<td>1</td>
<td>3/1</td>
</tr>
<tr>
<td>Act2</td>
<td>1</td>
<td>1</td>
<td>3/1</td>
</tr>
<tr>
<td>Act3</td>
<td>2</td>
<td>1</td>
<td>3/1</td>
</tr>
<tr>
<td>Act4</td>
<td>3</td>
<td>1</td>
<td>3/1</td>
</tr>
<tr>
<td>Act5</td>
<td>4</td>
<td>1</td>
<td>3/1</td>
</tr>
<tr>
<td>Act6</td>
<td>5</td>
<td>1</td>
<td>3/1</td>
</tr>
<tr>
<td>Act7</td>
<td>6</td>
<td>1</td>
<td>3/1</td>
</tr>
<tr>
<td>Act8</td>
<td>7</td>
<td>1</td>
<td>3/1</td>
</tr>
<tr>
<td>Act9</td>
<td>8</td>
<td>1</td>
<td>3/1</td>
</tr>
<tr>
<td>Act10</td>
<td>9</td>
<td>1</td>
<td>3/1</td>
</tr>
</tbody>
</table>

To add a new activator, click ![Add Activator](xx110000081)

To remove an activator, click ![Remove Activator](xx110000082)

To edit an activator setting, select the activator from the list and edit the value in the lower UI.

For descriptions of the activator settings, see Activators properties on page 108.
Here you define the configurations of the zones.
A zone is a collection of activators with the same state, while a configuration is a setup of zones that a tool can have. A tool can have several configurations but only one at a time can be active.
The figure shows a configuration example

```
Vacuum Activators
```

To add or remove a configuration, click
```
+ or
```

in the Configurations group; To set a configuration as default, right click on a configuration and select Set as Default Configuration;

To add or remove a zone, click "Add Zone" and "Delete Zone" button.

To include an activator into a zone, select the check box at the activator row and the zone column.

The activators are defined at the Vacuum Activators page.

### Activators properties

The following table describes the settings of the activators:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The activator’s name</td>
</tr>
<tr>
<td>Position</td>
<td>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.</td>
</tr>
<tr>
<td>Orientation</td>
<td>The center orientation of the activator related to the TCP.</td>
</tr>
<tr>
<td>Size</td>
<td>The size of the activator.</td>
</tr>
<tr>
<td>Start bit</td>
<td>Defines the first bit field where the activator is connected to the Activators control signal.</td>
</tr>
</tbody>
</table>
4 Navigating Palletizing PowerPac

4.3.24.2. Tool Functions

Continued

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Length</td>
<td>The number of bit fields used for the activator.</td>
</tr>
<tr>
<td>A/D/I</td>
<td>Defines the Active (A), Deactivated (D), and Idle (I) states for the activator. The state for the activator is set by the values of the bits defined by the fields of the Start Bit and No. Bits. The different states:</td>
</tr>
<tr>
<td></td>
<td>?Active state, which is set for an activator when it holds an item.</td>
</tr>
<tr>
<td></td>
<td>?Deactivated state, which is set for an activator when it releases an item.</td>
</tr>
<tr>
<td></td>
<td>?Idle state, which is set when no item is held by the activator.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>If Bit Length is set to the value 1, the Idle state is not used since only two states can be given with one bit.</td>
</tr>
</tbody>
</table>

Clamp

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

The editing UI of Clamp is as following:
### Properties

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opening and closing of clamp can be controlled by DI or GI signals, and when the open or close state is reached, it can set DO or GO signals to certain values. Also, a mechanical movement in 3D view may be included to visualize the opening and closing.</td>
</tr>
</tbody>
</table>

**Open/Close Clamp**

Opening and closing of clamp can be controlled by DI or GI signals, and when the open or close state is reached, it can set DO or GO signals to certain values. Also, a mechanical movement in 3D view may be included to visualize the opening and closing.

To add/edit/remove a signal to trigger the open movement, click in the Control Signals (Input) group. You can specify the signal type (DI/DO/GI/GO), the signal set value, the pulse value and length. See *Edit Signal Control on page 118* for detailed information.

To add/edit/remove a signal to be triggered when open movement is finished, click add/edit/remove in the Status Signals (Output) group. See *Edit Signal Control on page 118* for detailed information of setting signal and value.

To add a joint movement for opening of clamp, select the corresponding joint and move the slider bar to the opened position. You can also click the slider bar and input the joint position directly.

To edit the time that the joint takes to move to the open position, input a time value (s) in the Move Duration number box.

The above example specifies that:

\[ \text{When the input tool signal “giOpenClamp” goes to 1, the clamp will start to open. The open movement will take 0.2 seconds to move joint J2 to position 0. When the open movement is finished and the joint is at position 0, the output tool signal “goClampOpened” will be set to 1} \]
4 Navigating Palletizing PowerPac

4.3.24.2. Tool Functions

Continued

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open/Close Hook</td>
<td>Same as Clamp</td>
</tr>
<tr>
<td>Picking Sensor (Simulation)</td>
<td>The picking sensor is mainly for simulation use. It defines an area (usually within the clamp tool) that will intersect with the products when robot moves to the picking position. In simulation, the products that intersect with this area during the closing of the clamp will be picked up by the clamp. Thus users need to set the position and size of this area with appropriate values so that this area will intersect with the products that are “intended” to be picked when the tool is at picking positions.</td>
</tr>
</tbody>
</table>

During editing, the picking sensor area will also be shown in 3D view as a gray box:

Pick Mode: Pick mode defines whether the tool function should pick the products from top or bottom. This affects the default position of the tool relative to the products, defined in Pick Setting.

Currently only Center Top and Center Bottom are supported:
- Center top: in default position, the center of the picking sensor area will be aligned with the center top of the product
- Center bottom: in default position, the center bottom of the picking sensor area will be aligned with the center bottom of the product

Continues on next page
4 Navigating Palletizing PowerPac

4.3.24.2. Tool Functions

Continued

Claw

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

The editing UI of Claw is as following:

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open/Close Claw</td>
<td>Opening and closing of claw is similar to that of clamp. See Open/Close Clamp to see how to edit control signals and status signals, and joint movements.</td>
</tr>
<tr>
<td>Picking Sensor (Simulation)</td>
<td>The picking sensor for claw is the same as that of Clamp. See Picking Sensor (Simulation) for clamp for detailed information.</td>
</tr>
</tbody>
</table>
A search function contains a searcher movement, and search sensor. The movement contains the opening, the loosening, and the closing of searcher. And the search sensor describes in simulation where on the tool to detect product’s height (such as pallet stack).

The editing UI of Search is as following:

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>The search stop trigger describes which tool signal to trigger when a product is detected by search sensor, and the flank type of the signal that is triggered.</td>
</tr>
<tr>
<td></td>
<td><strong>Search Stop Trigger (D0)</strong></td>
</tr>
<tr>
<td>Signal</td>
<td>doSearchOpened</td>
</tr>
<tr>
<td></td>
<td>Trigger when signal changes from high to low</td>
</tr>
<tr>
<td>xx110000097</td>
<td>The trigger flank type describes what kind of signal value change indicates a search stop:</td>
</tr>
<tr>
<td></td>
<td>?Both Flanks: Trigger for any signal change</td>
</tr>
<tr>
<td></td>
<td>?Negative Flank: Trigger when signal changes from high to low</td>
</tr>
<tr>
<td></td>
<td>?Positive Flank: Trigger when signal change from low to high</td>
</tr>
</tbody>
</table>
A typical sequence of search process includes: open searcher, loosen search, wait until search sensor touches product and stop signal triggers, close searcher.

The steps related to joint movements are described as Open Searcher, Loosen Searcher, and Close Searcher.

The editing each movement is same as Open/Close Clamp. See Open/Close Clamp for detailed information.
4 Navigating Palletizing PowerPac

4.3.24.2. Tool Functions

Continued

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search Sensor (Simulation)</td>
<td>The editing of search sensor is the same as that of Clamp, except that for search, Pick Mode information is not needed, since search is not used for picking. Note that the position of the sensor should be set carefully so that the height of the lowest position of the sensor area is the same as the height of the search TCP position. Otherwise, during simulation, the sensor SmartComponent may touch the pallet/sheet stack earlier than the TCP, and robot can not calculate the height value correctly, thus missing the product picking afterwards.</td>
</tr>
</tbody>
</table>

During editing, the search sensor area will also be shown in 3D view as a gray box:
A sensor function is a simple function to add a product sensor and triggers certain DO when a product is detected.

The editing UI of Sensor is as following:

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>A DI signal can be specified to activate the sensor, and a DO signal should be specified to be triggered when product is detected.</td>
</tr>
<tr>
<td></td>
<td><strong>Activate Sensor (DI)</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Signal</strong> dStartPartCheck</td>
</tr>
<tr>
<td></td>
<td><strong>Sensor Out (DO)</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Signal</strong> doPartChecked</td>
</tr>
<tr>
<td></td>
<td><strong>Sensor (Simulator)</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Position (Relative to TCP) (mm)</strong></td>
</tr>
<tr>
<td></td>
<td>0.000 0.000 0.000</td>
</tr>
<tr>
<td></td>
<td><strong>Orientation (deg)</strong></td>
</tr>
<tr>
<td></td>
<td>0.000 0.000 0.000</td>
</tr>
<tr>
<td></td>
<td><strong>Size (x,y,z) (mm)</strong></td>
</tr>
<tr>
<td></td>
<td>10.000 10.000 20.000</td>
</tr>
</tbody>
</table>

**Unit Mover**

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

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open/Close Setting</td>
<td>Opening and closing of a Unit Mover is similar to that of clamp. See Open/Close Clamp to see how to edit control signals and status signals, and joint movements.</td>
</tr>
</tbody>
</table>

### Edit Signal Control

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Event Sequence (for claw) (Picking and placing events)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>Pick</td>
</tr>
<tr>
<td>Wait</td>
</tr>
<tr>
<td>Pick</td>
</tr>
<tr>
<td>Wait</td>
</tr>
</tbody>
</table>

Continues on next page
### Tool Events and Scenarios

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

The movement phase (i.e. action) of a pick/place target where this event is used.

Each pick/place target contains at least 1 Approach point, 1 pick/place point, and 1 Depart point. Depending on the approach/depart setting in operation sets, one pick/place target may contain more than 1 Approach points and 1 Depart points. (See Operation Set on page 163 for detailed information on operation, target, and actions)

Accordingly, in picking and placing, there are 6 movements defined respectively:

Pick First Approach -> Pick Last Approach -> Pick Last Depart
Pick First Depart -> Pick Last Depart
Place First Approach -> Place Last Approach -> Place Last Depart

First Approach means the first approach point of a pick (or place) operation.

Last Approach means the last approach point of a pick (or place) operation. If only 1 approach point was specified originally, an extra point will be added at the same location as the last approach.

Pick (or Place) means the actual point where pick (or place) happens.

First Depart means the first depart point of a pick (or place) operation. If only 1 depart point was specified originally, an extra point will be added at the same location as the first depart.

Last Depart means the last depart point of a pick (or place) operation.

Add, edit or delete an event

To add/edit/remove an event, click

Add, edit or delete an event

under the event list.
### Tool Events and Scenarios

**Scenario List**
- List of contained tool scenarios.
- A tool should at least have one tool scenario.

**Pick/Place Scenario**
- For each scenario, two tree views will be shown for Picking and Placing, including all the movements. The tool functions that contain events for the movement will be listed under the movement node.
- To include a tool function into a scenario at a specific movement, select the check box before the tool function node under the movement node.
The Create Feeder interface is used to create a PzPP compatible feeder SmartComponent model. The SmartComponent model can then be added to the robot, or saved as RobotStudio library (.rslib) file, and be imported and reused in other palletizing projects.

A PzPP compatible feeder includes properties of:

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

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Type</td>
<td>The default type is used when this SmartComponent model is added for a robot, and a feeder is to be created. The default feeder type (i.e., InFeeder, OutFeeder, PalletFeeder, or SheetFeeder) will use the type defined here, but user can still choose to use another type. See Add feeder on page 49 for detailed information.</td>
</tr>
<tr>
<td>Hotspots</td>
<td>The list of hotspots contained by this feeder.</td>
</tr>
</tbody>
</table>
NOTE!
The work object should be calibrated with its origin where items are fixed, that is along any guide and stop rails. For an infeeder or other feeders used for picking, the work object calibration is especially important. If the picking location of the tool must be adjusted, then update the tool location in Pick Setting. Avoid adjusting the displacement frame or tuning the location of the feeder, since this might reduce the accuracy when items are placed. For a palletizing feeder or other feeders where products are placed, the work object calibration can be adjusted with the displacement frame or by tuning the location of the feeder to modify the place location of items.
A viewpoint stores the location and direction of a virtual camera in the 3D environment. It stores points of interest in a station and to create camera movements during simulation.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx110000112</td>
<td>To view the objects in the Top orientation.</td>
</tr>
<tr>
<td>xx110000113</td>
<td>To view the objects in the Front orientation.</td>
</tr>
<tr>
<td>xx110000114</td>
<td>To view the objects in the Right orientation.</td>
</tr>
</tbody>
</table>
4.3.27. Adjust Position of Object

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

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="xx110000115" alt="Icon" /></td>
<td>To adjust the position of selected object in main 3D view.</td>
</tr>
<tr>
<td><img src="xx110000116" alt="Icon" /></td>
<td>To adjust the orientation of selected object in main 3D view.</td>
</tr>
</tbody>
</table>
This group is related to displaying help documentation and the basic software information.

<table>
<thead>
<tr>
<th>To...</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Help File</td>
<td>Click Help button, a help file with chm format will display to give detailed information about this PowerPac.</td>
</tr>
<tr>
<td>Show Basic Information</td>
<td>Click About button, a dialog will display to show basic information about this PowerPac, such as version number, status of license, etc.</td>
</tr>
</tbody>
</table>
4.4 Layout Browser

4.4.1. Overview

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

<table>
<thead>
<tr>
<th>Icon</th>
<th>Node</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Station Icon" /></td>
<td>Station</td>
<td>The station node</td>
</tr>
<tr>
<td><img src="image2.png" alt="Robot Icon" /></td>
<td>Robot</td>
<td>The robot in the station.</td>
</tr>
<tr>
<td><img src="image3.png" alt="Tool Icon" /></td>
<td>Tool</td>
<td>A tool SmartComponent in the station. This may or may not be attached to a robot</td>
</tr>
<tr>
<td><img src="image4.png" alt="Feeder Icon" /></td>
<td>Feeder</td>
<td>A feeder SmartComponent in the station. This may or may not be added into a robot.</td>
</tr>
<tr>
<td><img src="image5.png" alt="Part Icon" /></td>
<td>Part</td>
<td>A physical object in RobotStudio. Parts with geometric information are made up of one or more 2D or 3D entities. Parts without geometric information are empty.</td>
</tr>
</tbody>
</table>
4 Navigating Palletizing PowerPac

4.4.2. Station

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

<table>
<thead>
<tr>
<th>Context Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import Model…</td>
<td>Import one model into this station, which should be a library component file for RobotStudio.</td>
</tr>
</tbody>
</table>
4.4.3. Robot

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

<table>
<thead>
<tr>
<th>Context Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jump Home</td>
<td>Jump the robot to the home position.</td>
</tr>
<tr>
<td>Mechanism Joint Jog</td>
<td>Jog the robot's joint.</td>
</tr>
<tr>
<td>Set Position…</td>
<td>Set robot's position.</td>
</tr>
<tr>
<td>Examine</td>
<td>Adjust an appropriate viewpoint for the robot.</td>
</tr>
<tr>
<td>Rename</td>
<td>Change the robot's name.</td>
</tr>
<tr>
<td>Visible</td>
<td>Show or hide the robot.</td>
</tr>
</tbody>
</table>
4 Navigating Palletizing PowerPac

4.4.4. Tool SmartComponent

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

### Context Menus from the Tool Node

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

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

### Context Menus from the Feeder Node

<table>
<thead>
<tr>
<th>Context Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Feeder Hotspots…</td>
<td>Edit hotspots for this feeder. See Create Feeder on page 124 for detailed description</td>
</tr>
<tr>
<td>Disconnect Library</td>
<td>Disconnect the library</td>
</tr>
<tr>
<td>Save As Library</td>
<td>Save the library as rslib file</td>
</tr>
<tr>
<td>Attach to / Attached Status</td>
<td>If the feeder has not been attached to the robot, click Attach to menu and select the right robot, this feeder will attach to the selected robot. If the tool has been attached to a robot, which robot is attached to will display in this context menu.</td>
</tr>
<tr>
<td>Set Position…</td>
<td>Set feeder’s position.</td>
</tr>
<tr>
<td>Examine</td>
<td>Adjust an appropriate viewpoint for the feeder.</td>
</tr>
<tr>
<td>Rename</td>
<td>Change the feeder’s name.</td>
</tr>
<tr>
<td>Visible</td>
<td>Show or hide the feeder.</td>
</tr>
<tr>
<td>Delete</td>
<td>Delete the feeder from station.</td>
</tr>
</tbody>
</table>
You can access the context menus for the part node in the layout browser by right-clicking the part node.

### Context Menus from the Part Node

<table>
<thead>
<tr>
<th>Context Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build into a Tool…</td>
<td>Build this part as a tool Smart Component, which can be used by the robot.</td>
</tr>
<tr>
<td></td>
<td>See <em>Create Tool on page 102</em> for detailed description.</td>
</tr>
<tr>
<td>Build into a Feeder…</td>
<td>Build this part as a feeder Smart Component, which can be used by the robot.</td>
</tr>
<tr>
<td></td>
<td>See <em>Create Feeder on page 124</em> for detailed description.</td>
</tr>
<tr>
<td>Set Position…</td>
<td>Set part's position.</td>
</tr>
<tr>
<td>Examine</td>
<td>Adjust an appropriate viewpoint for the part.</td>
</tr>
<tr>
<td>Rename</td>
<td>Change the part's name.</td>
</tr>
<tr>
<td>Visible</td>
<td>Show or hide the part.</td>
</tr>
<tr>
<td>Delete</td>
<td>Delete the part from station.</td>
</tr>
</tbody>
</table>
4.5 Programming Browser

4.5.1. Overview

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

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

<table>
<thead>
<tr>
<th>Icon</th>
<th>Node</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Image" alt="xx1100000122" /></td>
<td>Project</td>
<td>The project node</td>
</tr>
<tr>
<td><img src="Image" alt="xx1100000123" /></td>
<td>Controller</td>
<td>A controller that exists in the station</td>
</tr>
<tr>
<td><img src="Image" alt="xx1100000124" /></td>
<td>Robot</td>
<td>A robot (motion task) that exists in the parent controller</td>
</tr>
<tr>
<td><img src="Image" alt="xx1100000125" /></td>
<td>Tool</td>
<td>The tool used by the parent robot. It references a tool SmartComponent.</td>
</tr>
<tr>
<td><img src="Image" alt="xx1100000126" /></td>
<td>Feeders</td>
<td>The collection of feeders for the parent robot</td>
</tr>
<tr>
<td><img src="Image" alt="xx1100000127" /></td>
<td>Feeder</td>
<td>A feeder that is used by the robot. It references a feeder SmartComponent.</td>
</tr>
<tr>
<td><img src="Image" alt="xx1100000128" /></td>
<td>Operation Set</td>
<td>An Operation Set program that picks places from(to) the parent feeder. Depending on the type of item it handles (box, bag, pallet, or sheet), and whether it is picking or placing, the icon is different.</td>
</tr>
</tbody>
</table>
4 Navigating Palletizing PowerPac

4.5.1. Overview

Continued

<table>
<thead>
<tr>
<th>Icon</th>
<th>Node</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx110000129</td>
<td>Flows</td>
<td>The collection of flows for the parent robot</td>
</tr>
<tr>
<td>xx110000130</td>
<td>Flow</td>
<td>A flow that exists for the parent robot</td>
</tr>
<tr>
<td>xx110000131</td>
<td>Job (Master Operation Set)</td>
<td>A supported job for the parent flow. It actually references to an Operation Set of the flow's master feeder.</td>
</tr>
<tr>
<td>xx110000132</td>
<td>Slave Operation Set</td>
<td>Reference to an Operation Set on the slave feeders that are related to the master job.</td>
</tr>
</tbody>
</table>
4.5.2. Project

4.5.2.1. Introduction

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

<table>
<thead>
<tr>
<th>Context Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Settings</td>
<td>Edit settings for the project. See Project Settings on page 138 for detailed description</td>
</tr>
<tr>
<td>Rename</td>
<td>Change the Project's name.</td>
</tr>
</tbody>
</table>
4.5.2.2. Project Settings

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

General

The project name and description can be edited in this page.
4 Navigating Palletizing PowerPac

4.5.2.2. Project Settings

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 on page 184.

Continue Pick-Place is always available.

Tune Limitation

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

**Feeder WorkObject Group**

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

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

**Example**

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

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

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

The following table describes the parameters of the Feeder WorkObject group:

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Displacement of the feeder in x-direction relative the work object.</td>
</tr>
<tr>
<td>Y</td>
<td>Displacement of the feeder in y-direction relative the work object.</td>
</tr>
<tr>
<td>Z</td>
<td>Displacement of the feeder in z-direction relative the work object.</td>
</tr>
<tr>
<td>Angle(Z)</td>
<td>Displacement angle of the feeder in the z-direction.</td>
</tr>
</tbody>
</table>

**Item Group**

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

**Example**

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

?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.

?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.

The following table describes the parameters of the Item group:

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Specifies maximum speed for an item.</td>
</tr>
<tr>
<td>Rot Speed</td>
<td>Specifies maximum orientation speed for an item.</td>
</tr>
<tr>
<td>Acc/Dec</td>
<td>Specifies maximum acceleration/deceleration for an item.</td>
</tr>
</tbody>
</table>
### 4.5.2.2. Project Settings

Continued

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pick Time</td>
<td>Specifies the time that the robot stays at the target position when picking an item.</td>
</tr>
<tr>
<td>Place Time</td>
<td>Specifies the time that the robot stays at the target position when placing an item.</td>
</tr>
<tr>
<td>Vacuum Activation Time</td>
<td>Specifies the time for vacuum activation.</td>
</tr>
<tr>
<td>Vacuum Deactivation Time</td>
<td>Specifies the time for vacuum deactivation.</td>
</tr>
<tr>
<td>Size (z)</td>
<td>Specifies the height of an item.</td>
</tr>
</tbody>
</table>
You can access the context menus for the controller node in the programming browser by right-clicking the controller node.

<table>
<thead>
<tr>
<th>Context Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Settings</td>
<td>Edit settings for the controller. See Controller Settings on page 143 for detailed description</td>
</tr>
</tbody>
</table>
4.5.3.2. Controller Settings

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

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

General

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulse Feeder DO Signals</td>
<td>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.</td>
</tr>
<tr>
<td>Pulse Length</td>
<td>Here you specify the pulse length that will be used if Pulse Feeder DO signals is selected. The pulse length can be set in the range of 50 ms to 2000 ms.</td>
</tr>
</tbody>
</table>
4 Navigating Palletizing PowerPac

4.5.3.2. Controller Settings

Continued

Events

![Screen capture of settings interface]

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

### Event Signals Group

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger (DI)</td>
<td>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.</td>
</tr>
<tr>
<td>Error Source (GI)</td>
<td>A group input signal representing the source of an error. If the signal is defined it indicates where the error has occurred (feeder, robot and/or controller) when the Trigger (DI) signal is set, and thus where a flow recovery action should be performed to handle the error. For information about the source setting, see below Source Settings Group.</td>
</tr>
<tr>
<td>Messages (GI)</td>
<td>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.</td>
</tr>
</tbody>
</table>
Source Settings Group

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Specifies the feeder, robot and/or controller that can be affected by the reported error.</td>
</tr>
<tr>
<td>Bit</td>
<td>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.</td>
</tr>
</tbody>
</table>

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

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

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

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

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

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

For examples, see Event and error reporting on page 270.
The feeder order settings define the physical order of the feeders, that is, in which order the robot will pass the feeders when making long movements through its working range.

The feeder order settings are used to define which feeders that will be considered when planning the path height of an intermediate movement between two feeders. Also, when moving to or from the home position, the feeder order settings will affect the path height.

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

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

4.5.3.2. Controller Settings

Messages

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

4.5.4.1. Introduction

4.5.4. Robot

4.5.4.1. Introduction

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

**Context Menus from the Robot Node**

<table>
<thead>
<tr>
<th>Context Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach Tool…</td>
<td>If there is no tool attached to the robot, this context menu will display. It is to add a tool to the robot. See <a href="#">Add tool on page 47</a> for detailed description.</td>
</tr>
<tr>
<td>Settings</td>
<td>Edit settings for the robot.</td>
</tr>
<tr>
<td>Jump Home</td>
<td>Jump the robot to the home position.</td>
</tr>
<tr>
<td>Set Position…</td>
<td>Set robot's position.</td>
</tr>
<tr>
<td>Examine</td>
<td>Adjust an appropriate viewpoint for the robot.</td>
</tr>
<tr>
<td>Visible</td>
<td>Show or hide the robot.</td>
</tr>
</tbody>
</table>
4.5.4.2. Robot Settings

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

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller</td>
<td>The name of the controller which the robot belong to.</td>
</tr>
<tr>
<td>Robot</td>
<td>The name of the robot.</td>
</tr>
<tr>
<td>Speed</td>
<td>The speed of the robot.</td>
</tr>
<tr>
<td>Acceleration/Deceleration</td>
<td>The speed can be set to full or low with the right button. The acceleration/deceleration speed of the robot.</td>
</tr>
<tr>
<td>Rotation Speed</td>
<td>The rotation speed of the robot.</td>
</tr>
<tr>
<td>Load and Start RAPID</td>
<td>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.</td>
</tr>
<tr>
<td>Add</td>
<td>To add a RAPID module to the project:</td>
</tr>
<tr>
<td></td>
<td>1. Click Add.</td>
</tr>
<tr>
<td></td>
<td>2. The Load RAPID dialog box appears, and you can select which RAPID module to add.</td>
</tr>
<tr>
<td></td>
<td>3. In the Load RAPID dialog box, click OK.</td>
</tr>
</tbody>
</table>

Continues on next page
4 Navigating Palletizing PowerPac

4.5.4.2. Robot Settings

Continued

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete</td>
<td>To remove a RAPID module from the project, select a module and click the button.</td>
</tr>
<tr>
<td>Edit</td>
<td>To edit a RAPID module, select a module and click Edit.</td>
</tr>
<tr>
<td>Export</td>
<td>To save a module with a new name, select a module and click Export.</td>
</tr>
<tr>
<td>Browse</td>
<td>Select software to edit the RAPID. In default, the software is notepad.</td>
</tr>
</tbody>
</table>
4.5.5. Tool

4.5.5.1. Introduction

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

<table>
<thead>
<tr>
<th>Context Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Tool Signals</td>
<td>Edit signals for this tool. See <em>Edit Tool Signals on page 152</em> for detailed description</td>
</tr>
<tr>
<td>Edit Tool Functions</td>
<td>Edit tool functions for this tool. See <em>Tool Functions on page 103</em> for detailed description</td>
</tr>
<tr>
<td>Edit Tool Events</td>
<td>Edit tool events for this tool. See <em>Tool Events and Scenarios on page 120</em> for detailed description</td>
</tr>
<tr>
<td>Joint Jog</td>
<td>Jog the tool’s joint.</td>
</tr>
<tr>
<td>Detach from</td>
<td>If the tool has been attached to a robot, click Detach from menu, this tool will detach from the robot. And this node will be deleted either.</td>
</tr>
<tr>
<td>Examine</td>
<td>Adjust an appropriate viewpoint for the tool.</td>
</tr>
<tr>
<td>Rename</td>
<td>Change the tool’s name.</td>
</tr>
<tr>
<td>Visible</td>
<td>Show or hide the tool.</td>
</tr>
</tbody>
</table>
4.5.5.2. Edit Tool Signals

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

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

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

Introduction

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

Context Menus from the Feeders Node

<table>
<thead>
<tr>
<th>Context Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Feeder…</td>
<td>It is to add a feeder into the station. See Add feeder on page 49 for detailed description.</td>
</tr>
</tbody>
</table>
4 Navigating Palletizing PowerPac

4.5.7.1. Introduction

4.5.7. Feeder

4.5.7.1. Introduction

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

### Context Menus from the Feeder Node

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

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

4.5.7.2. Edit Feeder

Continued

![Edit Feeder Screen](image)

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

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

Continues on next page
### 4 Navigating Palletizing PowerPac

#### 4.5.7.3. Edit Detailed Signals for Feeder

Continued

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

**NOTE!**

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

**NOTE!**

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a Safe Target</td>
<td>Click button</td>
</tr>
<tr>
<td></td>
<td><img src="xx110000088" alt="Add Safe Target" /> to add a new safe target in the list.</td>
</tr>
<tr>
<td>Delete a Safe Target</td>
<td>Select the safe target and click button</td>
</tr>
<tr>
<td></td>
<td><img src="xx110000090" alt="Delete Safe Target" /> to delete it from the list.</td>
</tr>
<tr>
<td>Reorder Safe Targets</td>
<td>Select the safe target and click</td>
</tr>
<tr>
<td></td>
<td><img src="xx110000146" alt="Reorder Safe Targets" /> or <img src="xx110000147" alt="Reorder Safe Targets" /> to reorder the target s in the list.</td>
</tr>
<tr>
<td>Change Move Type for a Safe</td>
<td>Select the safe position in the list and click the move type value.</td>
</tr>
<tr>
<td>Target</td>
<td>Select move type in the drop-down list.</td>
</tr>
<tr>
<td>Rename a Safe Target</td>
<td>In the Name text box, type a new name for the safe target. In default, a</td>
</tr>
<tr>
<td></td>
<td>unique name can be generated automatically when a new safe target is created.</td>
</tr>
<tr>
<td>Select Axis Configuration</td>
<td>Select one axis configuration for the selected safe target in the combo box.</td>
</tr>
<tr>
<td></td>
<td>Also click the right button, the available configurations for the safe target</td>
</tr>
<tr>
<td></td>
<td>can be calculated automatically.</td>
</tr>
<tr>
<td>Show the Reference WorkObject</td>
<td>The referenced workobject of the safe target is shown in the Location page.</td>
</tr>
</tbody>
</table>
4 Navigating Palletizing PowerPac

4.5.7.4. Safe Targets

Continued

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

#### Item | Description
--- | ---
Default Height | The default height is the expected height of the feeder after an operation set has been completed. For an outfeeder the default height is normally set to Full or Latest. For an infeeder, the default height is normally set to Full, Empty or Latest depending on how the products are fed into the working range of the robot. Latest is the final height of the latest run operation set. The default height can be temporarily updated in runtime, for example set to Empty after unloading a completed stack from the working range of the robot, by using predefined signals in the extended I/O interface or the RAPID function 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 feeders and flows.
Full Height | Define the full height of the feeder in the work object frame (mm). The Auto generate check box will generate the value as the maximum height of the configured operation sets.
Empty Height | Define the empty height of the feeder in the work object frame (mm).
Safety Offset | An offset that is always added to the expected height of the feeder.
### 4.5.7.6. Set WorkObject Position

<table>
<thead>
<tr>
<th>To do</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change the Reference Coordinate System</td>
<td>Select the Reference coordinate system you want to use to position the WorkObject used for the feeder.</td>
</tr>
<tr>
<td>Edit WorkObject Position</td>
<td>Click in one of Position boxes, and then click the position in the graphics window to transfer the values to the Position boxes. Or type a new value in the Position boxes to specify the position of the WorkObject.</td>
</tr>
<tr>
<td>Edit WorkObject Orientation</td>
<td>Type a new value in the Orientation boxes to specify the orientation of the WorkObject. Or click <img src="xx110000151" alt="Image" /> and <img src="xx110000152" alt="Image" /> to adjust the orientation for the WorkObject.</td>
</tr>
<tr>
<td>Check Reach for the WorkObject</td>
<td>Click <img src="xx110000148" alt="Image" /> button to check whether the robot can reach the WorkObject based on the active TCP. If it is reachable, the status light will be red, otherwise its color is green.</td>
</tr>
</tbody>
</table>
4.5.8. Operation Set

4.5.8.1. Introduction

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

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

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

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

4.5.8.2. Start the Operation Set Configuration

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

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

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

#### Item | Description
--- | ---
**Pick Setting Selection** | Select the item, item group and the pick setting for that item group to use. Click on the button to see item count, weight, Pick setting ID, and related input signals from the online PLC when the targets have been generated.

**Validation** | Use Preview Palletizing function to view each target of the operation set. See *Preview Palletizing on page 80* for detailed information. Use Check Reach to quickly check reach-ability of all the targets, which will be indicated by the status light.

**General** | Pick or Place: indicates how does the robot access the item group
- Preparation Time (Simulation): if picking, it describes the time it takes for the item group to show up on the in-feeder before robot goes to pick; if placing, it describes the time it takes for the item group to be moved away on the out-feeder after robot has placed it. A robot can only continue to work on the feeder after the previous items are moved away.
- Displacement: describes the displacement between the item group and the feeder’s work object. To rotate the frame 90 degrees, click the Flip button, and offsets and angle will be updated to keep chosen alignment. Or you can also click “Drag in 3D View” and use Freehand Move to move the item CAD models directly.

*Continues on next page*
The following table describes the configuration parameters for the group operation set’s stack search:

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Approach/Depart</strong></td>
<td>Define the robot path to follow when picking or placing the item group.</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td></td>
<td>Also see Approach for detailed information.</td>
</tr>
<tr>
<td><strong>Search</strong></td>
<td>You can enable stack search only if the operation set is Pick. To enable the stack search, select the Activate Stack Search check box.</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td></td>
<td>Also see Stack search on page 175 for detailed information.</td>
</tr>
</tbody>
</table>

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

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

**Item** | **Description**
--- | ---
Pattern/Stack | Set the name of the Operation Set. Select the pallet pattern or item stack to use. Click on the button to see item count, weight, total height, and the related product selection signal from the online PLC when the targets have been generated.

Validation | Use Preview Palletizing function to view each target of the operation set. See Preview Palletizing on page 80 for detailed information. Use Check Reach to quickly check reach-ability of all the targets, which will be indicated by the status light beside the check button and before each layer. Also see Check Reach on page 177 for detailed information.
### Pattern tab

This section lists and shows information about all layers in the pallet pattern and the pick settings to be used for this pallet pattern.

In the Pick Settings to use list box, check the ones that you want to include in the operation set. It is important to select them in such a way that it is possible to complete each layer.

**NOTE!**

When a pick setting for an item group with more than one column is used, the algorithm to calculate the operations and target positions may be wrong. Warning information will be shown in 2D layout view and user should manually check the sequence and use Preview Palletizing to validate.

To edit layer offsets, select the layer and edit the values. To edit the operations for a layer, double-click on the layer, or click on the layer and then click Edit. Manually edited layers operations will be specifically marked as "Manual", otherwise, they will be marked as "Auto". See *Operation Editor on page 179* for more information. The display shows how the items are grouped together in different operations. For more information, see *Display information*.

### General tab

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O Value</td>
<td>Set the I/O value of the Operation Set. Also see <em>I/O interface on page 174</em> for detailed information.</td>
</tr>
<tr>
<td>Pick or Place</td>
<td>indicates how does the robot access the item group</td>
</tr>
<tr>
<td>Preparation Time (Simulation)</td>
<td>if picking, it describes the time it takes for the pattern/stack group to show up on the in-feeder before robot goes to pick; if placing, it describes the time it takes for the pattern/stack to be moved away on the out-feeder after robot has placed it. A robot can only continue to work on the feeder after the previous pattern/stack is moved away.</td>
</tr>
<tr>
<td>Start Corner</td>
<td>defines the corner of the pallet pattern where the robot should start.</td>
</tr>
<tr>
<td>Displacement</td>
<td>describes the displacement between the item group and the feeder's work object. To rotate the frame 90 degrees, click the Flip button, and offsets and angle will be updated to keep chosen alignment. Or you can also click &quot;Drag in 3D View&quot; and use Freehand Move to move the item CAD models directly.</td>
</tr>
</tbody>
</table>

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

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

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

**NOTE!**

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

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

<table>
<thead>
<tr>
<th>Layer offset</th>
<th>Used when</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item offset</td>
<td>Items are attached to the tool.</td>
</tr>
<tr>
<td>Tool offset</td>
<td>No items are attached to the tool.</td>
</tr>
</tbody>
</table>

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

**NOTE!**

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

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

Approach and depart

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Direction</strong></td>
<td>Only available for group operation sets. Defines the direction in the x-y plane of the product group when the robot is approaching a target for placing products or departing a target after picking products.</td>
</tr>
<tr>
<td><strong>General offset</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>
### Item Description

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>End approach in z only</td>
<td>Select this check box when you want the robot to move straight down just before it reaches the target. In the Offset box, type the extra approach distance.</td>
</tr>
<tr>
<td>Start depart in z only</td>
<td>Select this check box when you want the robot to move straight up when it leaves the target. In the Offset box, type the extra depart distance.</td>
</tr>
<tr>
<td>Add Horizontal Pick Approach</td>
<td>Select this check box when you want to add an additional approach point before the pick target, so that the tool will move to the pick target from side instead of above. In Direction selection box, select the horizontal direction; and in the Offset box, type the horizontal offset.</td>
</tr>
<tr>
<td>Concurrency</td>
<td>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.</td>
</tr>
</tbody>
</table>
| 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 group 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.

*Continues on next page*
### 4.5.8.4. Pattern/Stack Operation Set

Continued

<table>
<thead>
<tr>
<th>Type</th>
<th>Direction</th>
<th>End approach</th>
<th>Start depart in Z only</th>
<th>Position count in Z only</th>
<th>Movement</th>
</tr>
</thead>
<tbody>
<tr>
<td>any</td>
<td>none</td>
<td>no</td>
<td>no</td>
<td>3</td>
<td><img src="xx110000161" alt="Diagram 1" /></td>
</tr>
<tr>
<td>pick</td>
<td>any</td>
<td>no</td>
<td>no</td>
<td>3</td>
<td><img src="xx110000162" alt="Diagram 2" /></td>
</tr>
<tr>
<td>place</td>
<td>any</td>
<td>no</td>
<td>no</td>
<td>3</td>
<td><img src="xx110000163" alt="Diagram 3" /></td>
</tr>
<tr>
<td>pick</td>
<td>any</td>
<td>no</td>
<td>yes</td>
<td>4</td>
<td><img src="xx110000164" alt="Diagram 4" /></td>
</tr>
<tr>
<td>place</td>
<td>any</td>
<td>yes</td>
<td>no</td>
<td>4</td>
<td><img src="xx110000165" alt="Diagram 5" /></td>
</tr>
<tr>
<td>pick</td>
<td>any</td>
<td>yes</td>
<td>yes</td>
<td>5</td>
<td><img src="xx110000166" alt="Diagram 6" /></td>
</tr>
</tbody>
</table>
For pallet pattern operation sets the movements are generated a little bit differently. Since multiple picking or placing is supported, the height of the current item must be taken into account. The special cases, when the last item is placed as well as when the first item is picked, are handled separately. The table below illustrates the resulting movement for a pallet pattern operation set with different settings.

Use the following assumptions:

- 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.

<table>
<thead>
<tr>
<th>End approach in Z only</th>
<th>Start depart in Z only</th>
<th>Special case</th>
<th>Position count</th>
<th>Movement</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>no</td>
<td>no *</td>
<td>5</td>
<td><img src="xx110000167" alt="Movement" /></td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td>last place</td>
<td>4</td>
<td><img src="xx110000168" alt="Movement" /></td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td>first pick</td>
<td>4</td>
<td><img src="xx110000169" alt="Movement" /></td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
<td>no *</td>
<td>6</td>
<td><img src="xx110000170" alt="Movement" /></td>
</tr>
</tbody>
</table>

Continues on next page
### 4 Navigating Palletizing PowerPac

#### 4.5.8.4. Pattern/Stack Operation Set

*Continued*

<table>
<thead>
<tr>
<th>End approach in Z only</th>
<th>Start depart in Z only</th>
<th>Special case</th>
<th>Position count</th>
<th>Movement</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>yes</td>
<td>no *</td>
<td>6</td>
<td><img src="xx100000171" alt="Diagram" /></td>
</tr>
<tr>
<td>yes</td>
<td>yes</td>
<td>no *</td>
<td>7</td>
<td><img src="xx100000172" alt="Diagram" /></td>
</tr>
</tbody>
</table>

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

**NOTE!**

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

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

### I/O interface

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

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

*Continues on next page*
Stack search

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

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

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

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

Search tool

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

Pallet pattern

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

Configuration of stack search

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

Configuration of search tool

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

Stack search target sequence, example

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

<table>
<thead>
<tr>
<th>Sequence step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>2</td>
<td>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).</td>
</tr>
<tr>
<td>3</td>
<td>Resulting search target. Top layer is reached with the search tool and search stop occurs. Height of the pallet pattern is updated.</td>
</tr>
</tbody>
</table>
Click **Check Reach** to verify if all positions can be reached by the robot. All robot positions are checked for each layer and each operation, including safe positions, approach positions, pick/place positions, and stack search specific positions. Imported tune data can affect the result.

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

In current version, intermediate positions are NOT checked.

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

<table>
<thead>
<tr>
<th>Result</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status light</td>
<td>The status light will be Gray, Green, and Red, indicating reach status of unknown, all reachable, and containing unreachable targets. Each layer also has a status light indicating its containing targets.</td>
</tr>
</tbody>
</table>

Continued
### Result | Description
--- | ---
3D graphical display | In 3D view, if a layer contains unreachable positions, the related items with these positions are drawn with red color.

![3D Graphical Display](image1)

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.

![Graphic Display in Layer](image2)
When configuring a pattern operation set, it is automatically given operations, which means that which product groups and pick settings to use and how to access them is defined for each layer. In some situations there might be requirements not fulfilled by auto-generated operations. In such cases you can adjust the operations using the Operation Editor.

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

**NOTE!**

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

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

Continues on next page
4.5.8.5. Operation Editor

Continued

<table>
<thead>
<tr>
<th>To...</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto generate operations</td>
<td>Click Fill to auto generate the operations for the rest of the layer.</td>
</tr>
<tr>
<td>Select access direction</td>
<td>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.</td>
</tr>
</tbody>
</table>

NOTE!

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

NOTE!

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

4.5.9.1. Introduction

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

**Context Menus from the Flows**

<table>
<thead>
<tr>
<th>Context Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Flow…</td>
<td>It is to add a new flow into the robot.</td>
</tr>
<tr>
<td></td>
<td>See <em>Add Flow on page 182</em> for detailed description.</td>
</tr>
</tbody>
</table>
4.5.9.2. Add Flow

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

Continues on next page
### Early Request
Select Early request check box if the next request for new products to the slave feeder (used as a slave) is made before the operation is executed on the master feeder (used as a master). If this option is not selected, the next request for new products to the slave feeders is made after the operation is executed on the master feeder.

**NOTE!**
In practice, when palletizing and the outfeeder feeder is the master, the next operation is requested on the infeeder right after the last products are picked up from it. The consequence can be that if something fails on the way to the feeder in which to place the products, it might not be possible to redo the interrupted operation since the next product is already requested in the infeeder, which may not be the same as the interrupted one.

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

### Master Feeder
Select the feeder on which palletizing jobs are started. Often the feeder where the pallet pattern is built is used as a master. A feeder used as a master in a flow cannot be involved in any other flows.

### Slave Feeders
Select the feeder(s) used as slaves (slave feeders). These are the feeder(s) that will serve the master feeder with products. The slave feeder(s) are often pallet stacks, slipsheet stacks, and conveyor feeders. Slave feeders can be shared by multiple flows.
4 Navigating Palletizing PowerPac

4.5.10. Flow

4.5.10.1. Introduction

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

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

4.5.11.1. Introduction

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

<table>
<thead>
<tr>
<th>Context Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locate Original Operation</td>
<td>Locate the referenced operation set node in Programming browser.</td>
</tr>
<tr>
<td>Simulate Job</td>
<td>Starts simulation but only running the selected job.</td>
</tr>
<tr>
<td>Delete</td>
<td>Delete the referenced operation set of the master feeder.</td>
</tr>
</tbody>
</table>
4 Navigating Palletizing PowerPac

4.5.11.1. Introduction
5 RAPID program

5.1. Introduction

Structure of this chapter

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

5.2.1. Relationship between RAPID execution and PickMaster project

5.2 Overview

5.2.1. Relationship between RAPID execution and PickMaster project

Overview

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

Intermediate positions

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

Operation, target, action, event and product

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

Operation

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

One Operation contains one or more Targets.

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

Target

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

One Target contains one or more Actions and product data.

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

Action

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

One Action contains one or more Events.

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

Event

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

Product

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

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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Operation</td>
</tr>
<tr>
<td>B</td>
<td>Targets</td>
</tr>
<tr>
<td>C</td>
<td>Actions</td>
</tr>
<tr>
<td>D</td>
<td>Event (in this case <em>Turn off the vacuum</em>)</td>
</tr>
<tr>
<td>E</td>
<td>Products</td>
</tr>
<tr>
<td>F</td>
<td>To next InterMid position</td>
</tr>
</tbody>
</table>
5 RAPID program

5.2.2. RAPID modules overview

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

RAPID template modules
The PickMaster installation includes the following two RAPID template modules:

\* 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.
5.3 Program module templates

5.3.1. PmMain module

Overview

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

This section describes the following procedures:

?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

Procedure OperateSequence on page 191.
PmWaitProjStart - Wait for any active project on page 308.
PmGetWaByWobj - Get a reference to a work area using a work object data on page 292.

Procedure OperateSequence

Usage

This routine performs one cycle sequence.
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 \texttt{PmSearchAdjust} when running stack search.

The \texttt{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 \texttt{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 \texttt{PM\_ERR\_PALLET\_EMPTY} error, indicating that objects are missing on the work area, is not recovered by default. To recover \texttt{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 up 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 \texttt{MultiOperation} to \texttt{TRUE} to avoid moving to an intermediate position.
5. Recover the error in the error handler using \texttt{RETRY}. As a result the robot will search the new stack from the top.

Program code

```plaintext
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 \textit{Operate on page 192.}

Procedure \textbf{Operate}

Usage

This procedure is used to execute an operation.
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

```rapid
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
        MultiOperation:=FALSE;
    ENDWHILE

    ERROR
    TEST ERRNO
        CASE PM_ERR_PALLET_REDUCED:
            RAISE;
        CASE PM_ERR_PALLET_EMPTY:
            RAISE;
    ENDT

ENDPROC
```

Continues on next page
5 RAPID program

5.3.1. PmMain module

Continued

Related information

Procedure PmDoAction on page 213
Procedure SetLastPos on page 197.
Procedure MoveInterMid on page 198.

pm_wadescr - PickMaster work area reference on page 363.
pm_operationdata - PickMaster operation data on page 345.
pm_targetdata - PickMaster target data on page 359.
pm_actiondata - PickMaster action data on page 329.
PmGetOperation - Get operation from a work area on page 288.
PmCalcArmConf - Calculates the arm configuration on page 280.
PmGetTarget - Get target on page 320.
PmGetTgtAction - Get target action on page 322.
PmSearchAdjust - Adjust number of remaining layers on page 295

Variables waInFeeder1 and waOutFeeder1

Usage

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

Description

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

Program code

VAR pm_wadescr waInFeeder1;
VAR pm_wadescr waOutFeeder1;
PmGetFlow waInFeeder, waOutFeeder;
Operate waInFeeder1;
Operate waOutFeeder1;

Related information

pm_wadescr - PickMaster work area reference on page 363.

Constants MaxToolAngle and MinToolAngle

Usage

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

Description

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

Program code

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

Continues on next page
5 RAPID program

5.3.1. PmMain module

Related information

*pm_wadescr* - PickMaster work area reference on page 363.

*PmCalcArmConf* - Calculates the arm configuration on page 280.

Continued
5 RAPID program

5.3.2. PmUtility module

5.3.2. PmUtility module

Overview

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

Procedure MoveHomePos

Usage

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

Description

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

5.3.2. PmUtility module

Continued

Program code

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

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

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

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

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

Related information

PmWaitProjStart - Wait for any active project on page 308.
PmGetWaByWobj - Get a reference to a work area using a work object data on page 292.
PmSetLastWa - Set last used work area on page 297.

Procedure SetLastPos

Usage

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

Description

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

Continues on next page
5 RAPID program

5.3.2. PmUtility module

Continued

Arguments

_WorkArea_

Datatype: pm_wadescr
Last work area that was used.

_Tgt_

Datatype: pm_targetdata
Last target data that was used.

_Act_

Datatype: pm_actiondata
Last action data that was used.

Program code

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

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

Related information

_pm_actiondata - PickMaster action data on page 329._
_pm_targetdata - PickMaster target data on page 359._
_pm_wadescr - PickMaster work area reference on page 363._
_PmSetLastWa - Set last used work area on page 297._

Procedure MoveInterMid

Usage

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

Description

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

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

Continues on next page
ENDIF
ConfJ\On;
! Use the frame from tool0 and the load from target tool
TempTool:=Tgt.TargetTool;
TempTool.tframe:=tool0.tframe;
! Travel distance: 10%
InterMid1:=PmCalcIntermid(PmLastRobTgt,PmLastTool,PmLastWobj,
\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,
\MaxAngle?MaxAngle\MinAngle?MinAngle\AngleLimAx6
\MinZ:=MinZ\FromWa:=LastWorkArea\ToWa:=WorkArea);
InterMid3:=PmCalcIntermid(PmLastRobTgt,PmLastTool,PmLastWobj,
\MaxAngle?MaxAngle\MinAngle?MinAngle\AngleLimAx6
\MinZ:=MinZ\FromWa:=LastWorkArea\ToWa:=WorkArea);

! Travel distance: 50%
! No concurrency shall be used on the second intermediate
! movement to avoid the limitation of having too many
! consecutive movements with concurrency.
MoveJ InterMid2,Act.Speed,z200,TempTool\Wobj:=wobj0;

! Travel distance: 90%
IF Present(MoveToEndPoint) THEN
MoveJ\Conc,Act.RobTgt,Act.Speed,fine,TempTool\Wobj:=wobj0;
ELSE
MoveJ\Conc,InterMid3,Act.Speed,z200,TempTool\Wobj:=wobj0;
ENDIF
ENDPROC

Related information

- pm_actiondata - PickMaster action data on page 329.
- pm_targetdata - PickMaster target data on page 359.
- pm_wadescr - PickMaster work area reference on page 363.
- PmGetLastWa - Get last used work area on page 287.
- PmGetPathHeight - Get a safe path height for an intermediate movement on page 315.
- PmCalcArmConf - Calculates the arm configuration on page 280.

**Variable** HomePos

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

**Description**
The home position must be modified for custom purposes.

Continues on next page
5 RAPID program

5.3.2. PmUtility module

Continued

Program code

```
LOCAL PERS jointtarget
    HomePos:=[[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];  

...  
MoveAbsJ HomePos,HomeSpeed,fine,TempTool;
```

Related information

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

Overview

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

This section describes:
- **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.

Program code

```rapid
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;
```
! 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;
ProjInfo:=ProjInfo;
! Start the selected project
IF StartLoadRun THEN
  PmStartProj ProjectInfo.Name\Signal:=pmProject_goStatus;
  SetGO pmProject_goCurrent, ProjectSelection;
ENDIF
ELSE ! STARTING OR RUNNING
  ! Wait for project to be running
  PmWaitProjStart;
ENDIF

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

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

Late binding in Technical reference manual - RAPID kernel.
PmWaitProjStart - Wait for any active project on page 308.
PmStartProj - Start a PickMaster project on page 304.

---

**Trap** TrapProjectStopped

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

**Usage**

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

**Description**

TrapProjectStopped executes the following:

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

**Program code**

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

**Related information**

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.

Program code

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

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

5.3.4. PmProjServer module

Continued

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

! Connect all traps with its interrupts
ConnectTraps;

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

! Disconnect all traps from its interrupts
DeleteTraps;
ENDWHILE
ENDPROC

Trap TrapSetRecoverAction

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

Description

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

Program code

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

5.3.4. PmProjServer module

Continued

Program code

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

Trap TrapStopFlow

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

Description

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

Program code

TRAP TrapStopFlow
    VAR pm_flowinfo FlowInfo;
    VAR num FlowSelection;
    VAR num StopOption;
    FlowSelection:=alias_giSelectionFlow;
    StopOption:=alias_giStopOptionFlow;
    ! Get info from selected flow
    PmGetFlowInfo FlowSelection,FlowInfo;
    ! Stop the selected flow
    PmStopFlow FlowInfo.Name,StopOption;
    ERROR
        ! Continue supervision on recoverable errors
        IF ERRNO=PM_ERR_FLOW_NOT_FOUND THEN
            RETURN;
        ELSEIF ERRNO=PM_ERR_NO_RUNNING_PROJECT THEN
            RETURN;
Continues on next page
5 RAPID program

5.3.4. PmProjServer module

Continued

    ELSE
    RETURN;
    ENDIF
    ENDTRAP

Trapped TrapProjectStopped

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

Description

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

Program code

    TRAP TrapProjectStopped
    PmStopProj;
    ENDTRAP

Related information

PmGetFlowInfo - Get information about a specific flow on page 285.
PmStartFlow - Starts a specific flow on page 302.
PmStopFlow - Stop a specific flow on page 305.
PmStopProj - Stop current project on page 307.
5.4 System modules

5.4.1. System modules, overview

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

- ?tpmrcUser (open)
- ?tpmrcSys (open)
- ?tpmrcBase (encrypted).
5 RAPID program

5.4.2. Public system module pmrcUser

5.4.2. Public system module pmrcUser

Description

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

Trap TrapDIToolEvents

Usage

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

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

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

Program code

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

Trap TrapGIToolEvents

Usage

This trap works in the same way as TrapDIToolEvents but is executed for GI signals.
5.4.3. Public system module pmrcSys

**Description**

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

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

**Procedure** PmDoAction

**Usage**

This procedure prepares and executes every movement in the operation.

**Description**

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

**Arguments**

*WorkArea*

Datatype: pm_wadescr

Contains a reference to the work area to use.

*Tgt*

Datatype: pm_targetdata

The target data used for the move.

*Act*

Datatype: pm_actiondata

The action data used for the move.

**Program code**

```
PROC PmDoAction(VAR pm_wadescr WorkArea, VAR pm_targetdata Tgt,
                 VAR pm_actiondata Act)

    ...

    WHILE PmGetEvent(WorkArea, Tgt.TargetHandle, Act.ActionHandle,
                      Event) AND NumOfTriggEvents <= ArrSize DO
        TEST Event.Type
        ...
        CASE PM_EVENT_PROC:
        CASE PM_EVENT_DO:
        ...
        CASE PM_EVENT_GO:
        ...
        CASE PM_EVENT_WAIT_DI:
        ...
        CASE PM_EVENT_WAIT_GI:
        ...
        ENDTST
```

Continues on next page
ENDWHILE

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

PathAccLim Act.Accel.AccLim\AccMax:=Act.Accel.AccMax,
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,
    PmAckTarget WorkArea,Tgt,PM_ACK;
  ELSE
    Act.Zone,curr_StopPoint,curr_Tool,curr_WObj;
  ENDIF
5 RAPID program

5.4.3. Public system module pmrcSys

Continued

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
ERROR
TEST ERRNO
CASE ERR_SYM_ACCESS:
  IF Event.SignalName = "" THEN
    signalname := "<no signal>";
  ELSE
    signalname := Event.SignalName;
  ENDIF
  PmErrorLog 2354, ERRSTR_TASK, signalname,
  ERRSTR_CONTEXT, ERRSTR_UNUSED, ERRSTR_UNUSED\ErrorHandler:=TRUE;
  RAISE;
CASE PM_ERR_PALLET_REDUCED:
  PmAckTarget WorkArea,Tgt,PM_ACK;RAISE;
CASE PM_ERR_PALLET_EMPTY:
  PmAckTarget WorkArea,Tgt,PM_ACK;RAISE;
DEFAULT:
  RAISE;
ENDTEST
ENDPROC
...

Procedure PmDoMove1

Usage

This procedure evaluates whether concurrent moves are used.

Description

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

Continues on next page
5 RAPID program

5.4.3. Public system module pmrcSys

Continued

Arguments

Move
Datatype: pm_movetype
The type of movement that is used.

Conc
Datatype: bool
Tells if a concurrent move instruction is used.

ToPoint
Datatype: robtarget
The destination point of the movement.

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

T1
Datatype: triggdata
Variable that refers to trigger conditions and trigger activity.

T2
Datatype: triggdata
Variable that refers to trigger conditions and trigger activity.

T3
Datatype: triggdata
Variable that refers to trigger conditions and trigger activity.

T4
Datatype: triggdata
Variable that refers to trigger conditions and trigger activity.

T5
Datatype: triggdata
Variable that refers to trigger conditions and trigger activity.

T6

Continues on next page
Datatype: `triggdata`
Variable that refers to trigger conditions and trigger activity.

T7
Datatype: `triggdata`
Variable that refers to trigger conditions and trigger activity.

T8
Datatype: `triggdata`
Variable that refers to trigger conditions and trigger activity.

Zone
Datatype: `zonedata`
Zone data for the movement. Zone data describes the size of the generated corner path.

Inpos
Datatype: `stoppointdata`
The setting of the dwell in the motion.

Tool
Datatype: `tooldata`
The tool in use when the robot moves. The tool center point is the point that is moved to the specified destination position.

WObj
Datatype: `wobjdata`
The work object (coordinate system) to which the robot position in the instruction is related.

Program code

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

Continues on next page
5 RAPID program

5.4.3. Public system module pmrcSys

Continued

Related information

*pm_movetype - PickMaster movement type on page 342.*

**Procedure** PmDoMove2

**Usage**

This procedure evaluates whether dwell is used.

**Description**

This procedure evaluates the \Inpos data to check whether a dwell is used or not.
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
Datatype: triggdata
Variable that refers to trigger conditions and trigger activity.
T7
Datatype: triggdata
Variable that refers to trigger conditions and trigger activity.
T8
Datatype: triggdata
Variable that refers to trigger conditions and trigger activity.
Zone
Datatype: zonedata
Zone data for the movement. Zone data describes the size of the generated corner path.

Inpos
Datatype: stoppointdata
The setting of the dwell in the motion.

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

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

Program code

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

Continues on next page
Related information

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

Procedure PmDoMove3

Usage

This procedure evaluates the move type.

Description

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

5.4.3. Public system module pmrcSys

Continued

Arguments

*Move*
**Datatype:** pm_movetype
The type of movement that is used.

*Conc*
**Datatype:** bool
Tells if a concurrent move instruction is used.

*ToPoint*
**Datatype:** robtarget
The destination point of the movement.

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

*T1*
**Datatype:** triggdata
Variable that refers to trigger conditions and trigger activity.

*T2*
**Datatype:** triggdata
Variable that refers to trigger conditions and trigger activity.

*T3*
**Datatype:** triggdata
Variable that refers to trigger conditions and trigger activity.

*T4*
**Datatype:** triggdata
Variable that refers to trigger conditions and trigger activity.

*T5*
**Datatype:** triggdata
Variable that refers to trigger conditions and trigger activity.

*T6*
Continues on next page
Datatype: \textit{triggdata}
Variable that refers to trigger conditions and trigger activity.

T7
Datatype: \textit{triggdata}
Variable that refers to trigger conditions and trigger activity.

T8
Datatype: \textit{triggdata}
Variable that refers to trigger conditions and trigger activity.

Zone
Datatype: \textit{zonedata}
Zone data for the movement. Zone data describes the size of the generated corner path.

Inpos
Datatype: \textit{stoppointdata}
The setting of the dwell in the motion.

Tool
Datatype: \textit{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: \textit{wobjdata}
The work object (coordinate system) to which the robot position in the instruction is related.

Program code

\begin{verbatim}
PROC PmDoMove3(pm_movetype Move, \switch Conc, robtarget ToPoint, speeddata Speed, \VAR triggdata T1, \VAR triggdata T2, \VAR triggdata T3, \VAR triggdata T4, \VAR triggdata T5, \VAR triggdata T6, \VAR triggdata T7, \VAR triggdata T8, zonedata Zone, \VAR stoppointdata Inpos, PERS tooldata Tool, PERS wobjdata WObj)
\quad TEST Move
\quad CASE PM\_MOVE\_LIN:
\quad \quad MoveL \Conc?Conc, ToPoint, Speed, Zone \Inpos?Inpos, Tool \WObj:=WObj;
\quad CASE PM\_MOVE\_JOINT:
\quad \quad MoveJ \Conc?Conc, ToPoint, Speed, Zone \Inpos?Inpos, Tool \WObj:=WObj;
\quad CASE PM\_TRIGG\_LIN:
\quad \quad TriggL \Conc?Conc, ToPoint, Speed, T1, T2?T2 \T3?T3 \T4?T4 \T5?T5 \T6?T6, T7?T7 \T8?T8, Zone \Inpos?Inpos, Tool \WObj:=WObj;
\end{verbatim}

Continues on next page
5 RAPID program

5.4.3. Public system module pmrcSys

Continued

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

Related information

pm_movetype - PickMaster movement type on page 342.
The instruction TriggL in Technical reference manual - RAPID Instructions, Functions and Data types.
The instruction TriggJ in Technical reference manual - RAPID Instructions, Functions and Data types.
The instruction MoveL in 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, I/O trigger type and search signal.

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

WObj
Datatype: wobjdata
The work object (coordinate system) to which the robot position in the instruction is related.
5 RAPID program

5.4.3. Public system module pmrcSys

Continued

Program code

PROC PmDoSearch(VAR pm_wadescr WorkArea, VAR pm_targethandle TargetHandle, robtarget ToPoint, speeddata Speed, pm_searchdata SearchData, PERS tooldata Tool, PERS wobjdata WObj)
VAR robtarget SearchPoint;
VAR signaldi diSearchSignal;
VAR bool SearchError:=FALSE;
GetDataVal SearchData.SignalName,diSearchSignal;
TEST SearchData.StopType
CASE PM_STOP_NOT_USED:
TEST SearchData.IOTriggType CASE PM_BOTH_FLANKS:
SearchL \Sup, diSearchSignal \Flanks, SearchPoint, ToPoint, Speed, Tool \WObj:=WObj;
...
DEFAULT:
PmErrorLog 2362, ERRSTR_TASK,
ValToStr(SearchData.StopType), ERRSTR_CONTEXT,
ERRSTR_UNUSED, ERRSTR_UNUSED;
ENDTEST
TEST SearchData.SearchType
CASE PM_SEARCH_X:
PmSearchAdjust WorkArea, SearchData.SearchType,
SearchPoint.trans.x;
CASE PM_SEARCH_Y:
PmSearchAdjust WorkArea, SearchData.SearchType,
SearchPoint.trans.y;
CASE PM_SEARCH_Z:
PmSearchAdjust WorkArea, SearchData.SearchType,
SearchPoint.trans.z;
DEFAULT:
PmErrorLog 2361, ERRSTR_TASK,
ValToStr(SearchData.SearchType), ERRSTR_CONTEXT,
ERRSTR_UNUSED, ERRSTR_UNUSED;
ENDTEST

ERROR

TEST ERRNO
CASE ERR_WHLSEARCH:
SearchError:=TRUE;
SearchPoint:=ToPoint;
TRYNEXT;
CASE PM_ERR_PALLET_REDUCED:
IF SearchError = TRUE THEN
...
... RAISE ERR_WHLSEARCH;
5 RAPID program

5.4.3. Public system module pmrcSys

Continued

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

Related information

- pm_movetype - PickMaster movement type on page 342
- pm_searchdata - PickMaster search data on page 352
- pm_searchtype - PickMaster stack search type on page 354
- pm_stoptype - PickMaster stop type on page 358
- PmSearchAdjust - Adjust number of remaining layers on page 295

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

Variable pm_home_Wobj

Usage

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

Description

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

Variables LastRobTgt, LastWobj and LastTool

Usage

These variables are used to store the last position, work object, and tool.

Description

The variables are used to store the last position’s properties, to be able to calculate an intermediate position.

Continues on next page
5 RAPID program

5.4.3. Public system module pmrcSys

Continued

Program code

InterMid:=PmCalcIntermid(PmLastRobTgt, PmLastTool, PmLastWobj, Act.RobTgt, Tgt.TargetTool, Tgt.TargetWobj, IntermidPart \MaxAngle:=MaxToolAngle\MinAngle:=MinToolAngle\AngleLimAx6 \MinZ:=MinZ);

TASK PERS robtarget PmLastRobTgt:=[[0,0,0], [0,0,0], [0,0,0,0], [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], [0.0, 0.0, 0.0], [0.0, 0.0, 0.0]];

... 
PmLastRobTgt:=Act.RobTgt;
PmLastWobj:=Tgt.TargetWobj;
PmLastTool:=Tgt.TargetTool;
...
6 Runtime operation

6.1. Introduction

Structure of this chapter

This chapter describes the runtime operating interface to PickMaster. The runtime operating interface consists of three parts:

- 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 187, and RAPID reference information on page 277.

Prerequisites

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

6.2 FlexPendant interface

6.2.1. Introduction to PickMaster FlexPendant interface

PickMaster FlexPendant interface

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

- Open Project
- Production
- Process Signals
- Tune

Illustration

The illustration shows the PickMaster main menu.

Open Project

Open Project displays a list of all the PickMaster projects that have been downloaded to the robot controller. A project must be opened before it can be started from the production window. The status bar always contains information about the loaded project as well as the current location within the window hierarchy.

See Opening a project on page 232.

Production

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

See Starting and stopping production on page 233.
6 Runtime operation

6.2.1. Introduction to PickMaster FlexPendant interface

Process Signals

**Process Signals** presents a list of all the work areas, items, events, and the I/O signals that are connected to each work area. It is possible not only to view but also to set new signal values. This window also presents all the tool configurations that build up the tool together with the zones and the activators. For a detailed description of the tool configuration, see *Zone frame on page 374* and *Activators properties on page 108*.

See also *Process Signals on page 246*.

Tune

**Tune** is used to change the parameter values online while running the PickMaster application. See *Tuning on page 249*.
6.2.2. Opening a project

Introduction

This section describes how to open a PickMaster project from the Open Project window. A list of all the available projects on the controller appears together with a description, if provided. To provide a description, see Project Settings on page 138.

Opening a project

<table>
<thead>
<tr>
<th>Action</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. On the PickMaster main menu, tap Open Project.</td>
<td>The Open Project button is only accessible if the current PickMaster project is stopped.</td>
</tr>
</tbody>
</table>
| 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.

Related information

Starting and stopping production on page 233.
Process Signals on page 246.
6.2.3. Starting and stopping production

Overview

This section describes how to start and stop the palletizing process from the Production window.

Illustration, Production window

The illustration shows an example of the Production window when three flows are defined in the PickMaster project.

Starting a project

<table>
<thead>
<tr>
<th>Action</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. On the Production menu, tap Project.</td>
<td></td>
</tr>
<tr>
<td>2. Tap Start.</td>
<td>A warning appears if the system is in motors off state.</td>
</tr>
</tbody>
</table>
6 Runtime operation

6.2.3. Starting and stopping production

Continued

Stopping a project

<table>
<thead>
<tr>
<th>Action</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. On the Production menu, tap Project.</td>
<td></td>
</tr>
<tr>
<td>2. Tap Stop. To proceed, tap Yes. To cancel, tap No.</td>
<td>A warning appears.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Action</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. On the Production menu, tap Project.</td>
<td></td>
</tr>
<tr>
<td>2. Tap Restart RAPID.</td>
<td>The Restart RAPID button is only available when program is paused.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Action</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. In the list of defined flows, tap on the flow to start. The Start Flow button becomes available.</td>
<td></td>
</tr>
<tr>
<td>2. Tap Start Flow.</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Action</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. In the list of defined flows, tap on the flow to stop. The Stop Flow button becomes available.</td>
<td></td>
</tr>
<tr>
<td>2. Tap Stop Flow.</td>
<td></td>
</tr>
</tbody>
</table>

Starting a specific job

Continues on next page
NOTE!
The flow must be running before a job can be started.

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. In the list of defined flows, tap on the flow to start a job on. The <strong>Start Job</strong> button appears.</td>
</tr>
<tr>
<td>2. Tap <strong>Start Job</strong>. The Start Job window appears.</td>
</tr>
</tbody>
</table>

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

**Start a new job on Flow2**

**Verify that pallet station is prepared, e.g. empty.**

**Restart job settings**

<table>
<thead>
<tr>
<th>Restart conditions</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer count</td>
<td>0</td>
</tr>
<tr>
<td>Product count</td>
<td>0</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Select <strong>restart condition</strong>.</td>
</tr>
<tr>
<td>2. Press <strong>Edit restart condition</strong> menu.</td>
</tr>
<tr>
<td>3. Enter appropriate value on the displayed numerical pad and press <strong>OK</strong>.</td>
</tr>
</tbody>
</table>

Continues on next page
6 Runtime operation

6.2.3. Starting and stopping production

Continued

Stopping a specific job

If stopping a job, it will be finished without being completed. The job will stop as soon as any currently ongoing or pending pick-places cycle has been completed or cancelled. Pending position requests on slaves will be cancelled if no targets are generated before the slave’s stop job timeout has passed.

If the job is waiting on a slave that has been running out of products, stop job can be used to finish the job without running any further pick-place cycles. The job will become stopped after the stop job timeout has passed.

**NOTE:** The flow must be running before a job can be stopped.

<table>
<thead>
<tr>
<th>Action</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. In the list of defined flows, tap on the flow to stop the job on. The <strong>Stop Job</strong> button becomes available.</td>
<td></td>
</tr>
<tr>
<td>2. Tap <strong>Stop Job</strong>. A pop-up window appears.</td>
<td></td>
</tr>
<tr>
<td>3. In the pop-up window, confirm that the job shall be stopped.</td>
<td></td>
</tr>
</tbody>
</table>

Starting and stopping all flows

<table>
<thead>
<tr>
<th>To...</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start all flows</td>
<td>Tap <strong>Start All Flows</strong>.</td>
</tr>
<tr>
<td>Stop all flows</td>
<td>Stop <strong>All Flows</strong>.</td>
</tr>
</tbody>
</table>

Viewing flow status

<table>
<thead>
<tr>
<th>Action</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. In the list of flows, tap on a specific flow. In the Production window, the <strong>Status</strong> command on the <strong>View</strong> menu becomes available.</td>
<td></td>
</tr>
<tr>
<td>2. On the <strong>View</strong> menu, tap <strong>Status</strong> and select a work area.</td>
<td></td>
</tr>
<tr>
<td>3. To close the Status Information window, tap <strong>Status</strong> on the <strong>View</strong> menu.</td>
<td></td>
</tr>
</tbody>
</table>
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.

**Work Area**

The name of the work area as specified in the PickMaster software. For further details, see Feeder on page 154.

**Status**

Provides status information about the work area, which can be Running or Error.

**Operation Set**

Specifies the last accessed operation set on the work area.

**Item Count**

A counter for the accumulated number of items that has been picked or placed on the work area since the project start.

**Viewing messages**

1. On the View menu, tap Messages.
2. In the list of messages, tap a specific message. A message window appears.

Continued on next page
The following illustration shows an example of the message window where only messages concerning flow recovery are shown.

<table>
<thead>
<tr>
<th>Code</th>
<th>Title</th>
<th>Date &amp; Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2398</td>
<td>Message</td>
<td>2007-8-30 6:44:21</td>
</tr>
<tr>
<td>2399</td>
<td>PickMaster Flow Stopped Immediately</td>
<td>2007-8-30 6:2:57</td>
</tr>
</tbody>
</table>

- **Code**: The code of the message.
- **Title**: The title of the message.
- **Date**: The date and time of the generated message.
- **Clear**: Clears the list of messages. Note that the messages are *not* deleted. If the production window is closed and then reopened, the cleared messages appear again.
- **OK**: Closes the window.

For details about message configuration, see *Messages on page 147*.

Changing or viewing flow stop options

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**.

<table>
<thead>
<tr>
<th>Flow stop option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finish cycle</td>
<td>The robot will continue palletizing until the current pick/place cycle of the job is finished.</td>
</tr>
<tr>
<td>Finish layer</td>
<td>The robot will continue palletizing until the current layer of the job is finished.</td>
</tr>
</tbody>
</table>
6 Runtime operation

6.2.3. Starting and stopping production

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.

<table>
<thead>
<tr>
<th>Flow stop option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finish job</td>
<td>The robot will continue palletizing until the current job is finished.</td>
</tr>
<tr>
<td>Stop immediately</td>
<td>The flow is stopped and a flow recovery action must be selected when restart.</td>
</tr>
<tr>
<td></td>
<td>If the flow is the current active flow, the robot will stop immediately,</td>
</tr>
<tr>
<td></td>
<td>without finishing a started cycle. A warning symbol appears next to the</td>
</tr>
<tr>
<td></td>
<td>flow, indicating that a flow recovery action must be selected, and the</td>
</tr>
<tr>
<td></td>
<td>status changes to Stopped.</td>
</tr>
<tr>
<td></td>
<td>If another flow shares one of the slave work areas, a warning symbol</td>
</tr>
<tr>
<td></td>
<td>will appear also next to that flow when this slave is requested, but the</td>
</tr>
<tr>
<td></td>
<td>flow status does not change.</td>
</tr>
<tr>
<td></td>
<td>If the stopped flow is not the current active flow, the robot will continue</td>
</tr>
<tr>
<td></td>
<td>palletizing using the remaining running flows.</td>
</tr>
</tbody>
</table>

Related information

*Opening a project on page 232.*
*Process Signals on page 246.*
*Feeder on page 154.*
*Events on page 144.*
*Robot Settings on page 149.*
6.2.4. Flow recovery

Overview

If a flow has been stopped by the event signals Error Source and Trigger, or with the stop option Stop immediately, a warning symbol appears next to the flows that are affected. When selecting a flow to be recovered, a warning symbol will appear on the Start Flow button in the Production window. This section describes how to resume a specific flow in such a case.

Recovering a specific flow

<table>
<thead>
<tr>
<th>Action</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. In the Production window, tap Start Flow.</td>
<td>See Illustration, Production window with a flow to recover on page 241.</td>
</tr>
<tr>
<td>2. The Restart options window appears. Select one of the displayed recovery options and tap OK.</td>
<td>You get three options to continue. See Illustration, Restart options window on page 241.</td>
</tr>
<tr>
<td>3. The Production window appears and there is a note symbol on the Start Flow button. Tap Start Flow.</td>
<td>See Illustration, Production window after selecting a recovery option on page 243.</td>
</tr>
<tr>
<td>4. A dialog box containing information about the selected option appears. Verify the status as specified in the message, and then tap OK.</td>
<td>After tapping OK, the flow is in production again. If the RAPID program has stopped, it will restart. See Illustration, Confirm restart information on page 244.</td>
</tr>
</tbody>
</table>
Illustration, Production window with a flow to recover

The following figure illustrates the Production window when an error/event has occurred.

Illustration, Restart options window

The flow has been stopped. One or more Work Areas have been set in error state.

Select workarea to finish its layer:

- Continue Pick-Place
- Redo last pick
- Restart Layer
- Next Pallet

Continues on next page
6 Runtime operation

6.2.4. Flow recovery

**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.

**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.

![Production window](image)

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.
Illustration, Confirm restart information

The following figure illustrates the **information** window to confirm while starting the flow after selecting a flow recovery action.

**Restarting other flows after error event**

If a flow is stopped by an error event (or "Stop immediately") while the robot is executing the flow, the RAPID program will also stop.

In order to restart execution of the other flows without first resolving the error:

1. Verify that the robot tool is prepared for the next pick/place cycle, for example, empty
2. Jog the robot to a safe position from where execution of the next pick/place cycle can be started
3. Move PP to Main
4. Restart RAPID
5. Confirm a warning message on the flex pendant

Continues on next page
WARNING!

If the PP is not moved to Main in step 3, the robot may start moving to an already fetched but not yet executed target. Execution will then stop on the next of the following functions:

- PmGetEvent
- PmGetTarget
- PmGetTgtAction

Illustration, Restart other flows after error event

The following figure illustrates the warning message which must be confirmed if RAPID is restarted after an error event of the executing flow.

Related information

Starting and stopping production on page 233
6.2.5. Process Signals

**Introduction**

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

- 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**

<table>
<thead>
<tr>
<th>Action</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Tap View and select Work Areas.</td>
<td>See Feeder on page 154.</td>
</tr>
<tr>
<td>3. From the Work Areas list, tap one work area.</td>
<td></td>
</tr>
</tbody>
</table>

The following illustration shows an example of the Process Signals window when one work area is selected.

![Process Signals Window Example](image)

**Viewing the tool configuration**

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Tap View and select Tool.</td>
</tr>
<tr>
<td>3. From the Tool list, tap one tool configuration.</td>
</tr>
</tbody>
</table>

Continues on next page
The following illustration shows an example of the Process Signals window when one tool configuration is selected.

### Viewing events

<table>
<thead>
<tr>
<th>Action</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Tap View and select Events.</td>
<td>See Events on page 144.</td>
</tr>
<tr>
<td>3. From the Events list, tap one controller.</td>
<td></td>
</tr>
</tbody>
</table>
## 6 Runtime operation

### 6.2.5. Process Signals

*Continued*

The following illustration shows an example of the **Process Signals** window when one controller is selected.

![Process Signals Window](image)

<table>
<thead>
<tr>
<th>Events</th>
<th>Name</th>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRC5</td>
<td>diErrorTrigger</td>
<td>0</td>
<td>DI</td>
</tr>
<tr>
<td></td>
<td>giErrorSource</td>
<td>0</td>
<td>GI</td>
</tr>
<tr>
<td></td>
<td>giErrorMessage</td>
<td>0</td>
<td>GI</td>
</tr>
</tbody>
</table>

**Related information**

*Opening a project on page 232.*

*Starting and stopping production on page 233.*
6.2.6. Tuning

Introduction

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

Illustrations, Tune window

Below the FlexPendant interface illustrating tuning of an item and a work area.

Item

The illustration shows an example of the Tune window when five products are defined, and the properties that can be tuned for each product.

<table>
<thead>
<tr>
<th>Item</th>
<th>Property</th>
<th>Current</th>
<th>Default</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small Coffee</td>
<td>Speed</td>
<td>5000</td>
<td>5000</td>
<td>mm/s</td>
</tr>
<tr>
<td>Medium Coffee</td>
<td>Ori speed</td>
<td>360</td>
<td>360</td>
<td>degrees/s</td>
</tr>
<tr>
<td>Large Coffee</td>
<td>Acceleration</td>
<td>5</td>
<td>5</td>
<td>m/s²</td>
</tr>
<tr>
<td>Pallet</td>
<td>Pick time</td>
<td>0,01</td>
<td>0,01</td>
<td>s</td>
</tr>
<tr>
<td>Slipsheet</td>
<td>Place time</td>
<td>0,01</td>
<td>0,01</td>
<td>s</td>
</tr>
<tr>
<td></td>
<td>Size z</td>
<td>80</td>
<td>80</td>
<td>mm</td>
</tr>
<tr>
<td></td>
<td>Pick activation</td>
<td>0,1</td>
<td>0,1</td>
<td></td>
</tr>
</tbody>
</table>

The following table describes the Property parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Specifies the maximum speed for an item.</td>
</tr>
<tr>
<td>Ori speed</td>
<td>Specifies the maximum orientation speed for an item.</td>
</tr>
<tr>
<td>Acceleration</td>
<td>Specifies the maximum acceleration/deceleration for an item.</td>
</tr>
<tr>
<td>Pick time</td>
<td>Specifies the time the robot stays at the target position when picking an item.</td>
</tr>
</tbody>
</table>

Continues on next page
6 Runtime operation

6.2.6. Tuning

Continued

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place time</td>
<td>Specifies the time the robot stays at the target position when placing an item.</td>
</tr>
<tr>
<td>Size Z</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Work Area</th>
<th>Property</th>
<th>Value</th>
<th>Default</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infeed</td>
<td>Disp offs x</td>
<td>10</td>
<td>0</td>
<td>mm</td>
</tr>
<tr>
<td>Outfeed</td>
<td>Disp offs y</td>
<td>0</td>
<td>0</td>
<td>mm</td>
</tr>
<tr>
<td>Slip feed</td>
<td>Disp offs z</td>
<td>0.8</td>
<td>0</td>
<td>mm</td>
</tr>
<tr>
<td>Infeed 2</td>
<td>Disprotz</td>
<td>0</td>
<td>0</td>
<td>degrees</td>
</tr>
<tr>
<td>Outfeed 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outfeed 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following table describes the Property parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disp offs x</td>
<td>Displacement of the work area in x-direction relative the work object.</td>
</tr>
<tr>
<td>Disp offs y</td>
<td>Displacement of the work area in y-direction relative the work object.</td>
</tr>
</tbody>
</table>
6 Runtime operation

6.2.6. Tuning

How to proceed

This section describes how to proceed with tuning of an item and a work area.

Tuning an item

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disp offs z</td>
<td>Displacement of the work area in z-direction relative the work object.</td>
</tr>
<tr>
<td>Disprotz</td>
<td>Displacement angle of the work area in the z-direction.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. On the Tune menu, tap View.</td>
<td></td>
</tr>
<tr>
<td>2. Tap Item.</td>
<td></td>
</tr>
<tr>
<td>3. In the Item list, tap the item to tune.</td>
<td></td>
</tr>
<tr>
<td>4. In the Property list, tap the item property to tune.</td>
<td></td>
</tr>
<tr>
<td>5. On the Tune menu, tap Tune Value.</td>
<td>The Item Tune Value window appears. See illustration below this procedure.</td>
</tr>
<tr>
<td>6. In the Increment drop-down combo box, select the size of increment.</td>
<td>The increment specifies the value that will be added to/subtracted from the item property for each time you tap the + or - button.</td>
</tr>
<tr>
<td>7. Tap Apply.</td>
<td></td>
</tr>
</tbody>
</table>

The following figure illustrates tuning of an item.

![Slipsheet Speed](image)

Tuning a work area

<table>
<thead>
<tr>
<th>Action</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. On the Tune menu, tap View.</td>
<td></td>
</tr>
<tr>
<td>2. Tap Work Area.</td>
<td></td>
</tr>
<tr>
<td>3. In the Work Area list, tap the work area to tune.</td>
<td></td>
</tr>
</tbody>
</table>

Continues on next page
6 Runtime operation

6.2.6. Tuning

Continued

<table>
<thead>
<tr>
<th>Action</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. In the <strong>Property</strong> list, tap the work area property to tune.</td>
<td></td>
</tr>
<tr>
<td>5. On the <strong>Tune</strong> menu, tap <strong>Tune Value</strong>. The <strong>Work Area Value</strong> window appears. See illustration below this procedure.</td>
<td></td>
</tr>
<tr>
<td>6. In the <strong>Increment</strong> 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.</td>
<td></td>
</tr>
<tr>
<td>7. Tap <strong>Apply</strong>.</td>
<td></td>
</tr>
</tbody>
</table>

The following figure illustrates tuning of a work area.

![Image of tuning interface](image)
6.3 I/O interface

6.3.1. Overview

PickMaster I/O interface

The PickMaster I/O interface is used by external equipment, such as a PLC, to control and supervise the palletizing process. It consists of two parts:

- 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.
6 Runtime operation

6.3.2. Default signals

6.3.2. Default signals

Introduction to default signals

More than one hundred (100) default signals are installed on every controller with the option Prepared for PickMaster. The signals can be used when setting up PickMaster lines and projects.

Configuration and setup

The signals can be selected in the configuration dialogs. As default, the signals are mapped to simulated I/O units, Pickmaster_Sim1, Pickmaster_Sim2, and Pickmaster_Sim3. During commissioning, the signals can be mapped to physical I/O units.

Additional signals are required if you use more than:

- 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

Continues on next page
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

Flows

Default signals are defined for flows. Signal name prefixes:

.panFlow
.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
6 Runtime operation

6.3.3. Basic I/O interface

6.3.3. 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)
- Position request trigger signal (DO)

The following signals must be setup if simulated target generation is selected for the work area. See Feeder on page 154.

- Position request trigger signal (DO)

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

Target generation trigger signal (DI)

The signal is mandatory unless simulated target generation is used.

The signal can also be skipped if palletizing jobs always are to be started using the FlexPendant interface.

A trigger pulse generates the start of a new palletizing/depalletizing job on that work area. A palletizing/depalletizing job is equivalent to one of the operation sets configured for the feeder. Before the signal is pulsed, the flow must be running, the position request trigger signal must be set and the work area must be prepared for the job to be started. For example:

- 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.

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

Continued on next page
If the robot execution signal is not defined, the *Position available* signal is reset after any operation is received in RAPID.

If the robot execution signal is defined, the *Position available* signal is set after:

- 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.

<table>
<thead>
<tr>
<th>I/O value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A project using this work area is not running.</td>
</tr>
<tr>
<td>1</td>
<td>Work area is running.</td>
</tr>
</tbody>
</table>
| 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`.

*Continued on next page*
6 Runtime operation

6.3.3. Basic I/O interface

Continued

Height state (GO)

The signal is optional.

The signal indicates the current height of the work area. The signal must have a bit length of at least three to represent the five possible states.

<table>
<thead>
<tr>
<th>I/O value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Full height. The height is equal to the setting of <strong>Full height</strong>, see <em>Robot Path Height on page 161</em>.</td>
</tr>
<tr>
<td>1</td>
<td>Active height. An operation set is executed and the height is updated after each pick or place until the operation set is completed.</td>
</tr>
<tr>
<td>2</td>
<td>Latest height. The height is equal to the final height of the latest run operation set.</td>
</tr>
<tr>
<td>3</td>
<td>Empty height. The height is equal to the setting of <strong>Empty height</strong>, see <em>Robot Path Height on page 161</em>.</td>
</tr>
<tr>
<td>4</td>
<td>Value. The height is set to a value with the RAPID function <strong>PmSetDefaultHeight</strong>.</td>
</tr>
</tbody>
</table>

Default signal, example: `pmOutfeeder1_goHeightState`.

Layer count (GO)

The signal is optional.

The signal value indicates the number of full layers on the work area.

Default signal, example: `pmOutfeeder1_goLayerCount`.

Product count (GO)

The signal is optional.

The signal value indicates the number of products in the top layer. If the top layer is full, the signal is set to zero.

Default signal, example: `pmOutfeeder1_goProductCount`.

Stop job (DI)

The signal is optional.

The signal is used to stop the currently ongoing job before it is completed. The flow must be running before a job can be stopped. The job is stopped by pulsing the signal.

The job will stop as soon as any currently ongoing or pending pick-places cycle is completed or cancelled. Pending position requests on slaves will be cancelled if no targets are generated before the slave's stop job timeout has passed. For more information on stop job timeout, see *Feeder on page 154*.

If the job is waiting on a slave that is running out of products, stop job can be used to finish the job without running any further pick-place cycles. The job will become stopped after the stop job timeout has passed.

Default signal, example: `pmOutfeeder1_diStopJob`.

Continues on next page
6 Runtime operation

6.3.3. Basic I/O interface

Continued

Robot execution (DI)

The signal is optional.
The signal is used to control whether the robot is allowed to approach the work area or not.
If the signal is reset, the RAPID execution will not pass the instruction PmGetTarget until the signal is set. After an operation is performed by the robot, the signal must be reset and then set again to allow the robot to approach the work area the next time.
The signal can also be used to finish the current operation set before all operations have been completed. If the signal is reset, the remaining targets will be removed.
Default signal, example: pmOutfeeder1_diRobotExec.

Redo search (DI)

The signal is optional.
The signal is used with operation sets having stack search activated.
Stack search is normally not used for a master work area.
If the signal is pulsed, the next operation will start with a search movement from the top of the stack. The signal can be used after adding new items on a stack. To affect the next approach to the work area, the signal must be pulsed before the robot receives the operation in RAPID.
Default signal, example: pmOutfeeder1_diRedoSearch.

Slave work areas

The following signals must be setup:

?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 Feeder on page 154.

?Position request trigger signal (DO)
?Position request product selection (GO)²
?Position request format selection (GO)³
Position request trigger signal (DO)

The signal is mandatory.

The signal is set by the controller when one of the corresponding master work areas requests a format on this slave work area from the PLC. The requested format must be defined as an operation set in the feeder. The request occurs after the previous pick and place cycle for that flow has finished.

If the flow uses early request, then the request will occur in advance, before the robot has finished the previous cycle. Early request will decrease cycle times if the same flow is run in consecutive pick place cycles. The signal is reset when the target generation trigger signal is pulsed.

Default signal, example: `pmInfeeder1_doPosReqTrig`.

Position request product selection (GO)

The signal is mandatory if there is more than one item.

The signal I/O value specifies the requested product when the position request trigger signal is set.

Default signal, example: `pmInfeeder1_goProdSel`.

Position request format selection (GO)

The signal is mandatory if there is more than one format for the same item.

The signal I/O value specifies the requested format when the position request trigger signal is set.

Default signal, example: `pmInfeeder1_goFormSel`.

Position request requesting master (GO)

The signal is optional.

The signal value indicates the requesting master work area. The I/O value of the work area is configured in the Work Area I/O Settings editor. For more information on the Work Area I/O Settings, see Edit Detailed Signals for Feeder on page 157.

Default signal, example: `pmInfeeder1_goReqMaster`.

Target generation trigger signal (DI)

The signal is mandatory unless simulated target generation is used.

A trigger pulse indicates that a previously requested format is now available for the work area to be handled by the robot.

Default signal, example: `pmInfeeder1_diTgtGenTrig`.

Target generation product selection (GI)

The signal is mandatory if there is more than one item.

The signal specifies the product I/O value of the available format when the target generation trigger signal is pulsed.

Default signal, example: `pmInfeeder1_giProdSel`.

Continued on next page
6 Runtime operation

6.3.3. Basic I/O interface

Continued

Target generation format selection (GI)
The signal is mandatory if there is more than one format for the same item.
The signal specifies the format I/O value of the available format when the target generation
trigger signal is pulsed.
Default signal, example: pmInfeeder1_giFormSel.

Target generation start layer count (GI)
The signal has no function and is not required.

Target generation start product count (GI)
The signal has no function and is not required.

Position available (DO)
The signal is optional.
The signal indicates if target positions can be received in RAPID to be executed by the robot.
Using the Robot execution signal affects the output of the Position available signal.
If the Robot execution signal not is defined, Position available is set after:
?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.

Continues on next page
Execution state (GO)

The signal is optional.

The signal indicates the runtime state of the work area.

<table>
<thead>
<tr>
<th>I/O value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A project using this work area is not running.</td>
</tr>
<tr>
<td>1</td>
<td>Work area is running.</td>
</tr>
<tr>
<td>2</td>
<td>Work area has an error. Flow recovery is required to recover from the error.</td>
</tr>
<tr>
<td>3</td>
<td>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.</td>
</tr>
</tbody>
</table>

Default signal, example: `pmInfeeder1_goExecState`.

Height state (GO)

The signal is optional.

The signal indicates the current height of the work area. The signal must have a bit length of at least three to represent the five possible states.

<table>
<thead>
<tr>
<th>I/O value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Full height. The height is equal to the setting of <strong>Full height</strong>, see <em>Robot Path Height on page 161</em>.</td>
</tr>
<tr>
<td>1</td>
<td>Active height. An operation set is being executed and the height is updated after each pick or place until the operation set is completed.</td>
</tr>
<tr>
<td>2</td>
<td>Latest height. The height is equal to the final height of the latest run operation set.</td>
</tr>
<tr>
<td>3</td>
<td>Empty height. The height is equal to the setting of <strong>Empty height</strong>, see <em>Robot Path Height on page 161</em>.</td>
</tr>
<tr>
<td>4</td>
<td>Value. The height is set to a value with the RAPID function PmSetDefaultHeight.</td>
</tr>
</tbody>
</table>

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: `pmInfeeder1_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.

Continued on next page
6 Runtime operation

6.3.3. Basic I/O interface

Continued

Robot execution (DI)

The signal is optional.
The signal is used to control whether the robot is allowed to approach the work area or not. If the signal is reset, the RAPID execution will not pass the instruction `PmGetTarget` until the signal is set. After an operation is performed by the robot, the signal must be reset and then set again to allow the robot to approach the work area next time.

The signal can also be used to finish the current operation set before all operations have been completed. If the signal is reset, the remaining targets will be removed.

Default signal, example: `pmInfeeder1_diRobotExec`.

Redo search (DI)

The signal is optional.
The signal is used with operation sets having stack search activated. If the signal is pulsed, next operation will start with a search movement from the top of the stack. The signal can be used after adding new items on a stack. To affect the next approach to the work area, the signal must be pulsed before the robot receives the operation in RAPID.

Default signal, example: `pmInfeeder1_diRedoSearch`. 
6.3.4. Extended I/O interface

**Controller system handling**

A number of default system signals are installed to handle the controller system, for example to set the controller in motors on state.

See *Technical reference manual - System parameters*, the topic I/O, for descriptions of system inputs and outputs.

**System inputs**

<table>
<thead>
<tr>
<th>System input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pmSystem_diLoadStartX</td>
<td>Load and start the PmProjMgr module for motion task T_ROBX.</td>
</tr>
<tr>
<td>pmSystem_diStart</td>
<td>Start RAPID execution.</td>
</tr>
<tr>
<td>pmSystem_diStop</td>
<td>Stop RAPID execution.</td>
</tr>
<tr>
<td>pmSystem_diStartMain</td>
<td>Start RAPID execution from Main.</td>
</tr>
<tr>
<td>pmSystem_diMotorsOn</td>
<td>Set motors on.</td>
</tr>
<tr>
<td>pmSystem_diResetEstop</td>
<td>Confirm reset of emergency stop.</td>
</tr>
</tbody>
</table>

**System outputs**

<table>
<thead>
<tr>
<th>System output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pmSystem_doCycleOn</td>
<td>Robot program is executing.</td>
</tr>
<tr>
<td>pmSystem_doMotorOn</td>
<td>Motors on state.</td>
</tr>
<tr>
<td>pmSystem_doRunchOk</td>
<td>Run chain is closed.</td>
</tr>
<tr>
<td>pmSystem_doEmStop</td>
<td>Emergency stop state.</td>
</tr>
<tr>
<td>pmSystem_doAutoOn</td>
<td>Automatic mode is used.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Project signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pmProject_goCurrent</td>
<td>The current project.</td>
</tr>
<tr>
<td>pmProject_goStatus</td>
<td>The status of current project.</td>
</tr>
<tr>
<td>pmProject_diStop</td>
<td>Stop current project.</td>
</tr>
<tr>
<td>pmProject_diStart</td>
<td>Start selected project.</td>
</tr>
<tr>
<td>pmProject_giSelection</td>
<td>Project selector.</td>
</tr>
<tr>
<td>pmProject_diSetDefaultHeight</td>
<td>Set default height for a work area.</td>
</tr>
<tr>
<td>pmProject_giDefaultHeight</td>
<td>Default height selector.</td>
</tr>
<tr>
<td>pmProject_giDefHeightWaSel</td>
<td>Work area selector for setting the default height.</td>
</tr>
<tr>
<td>pmProject_diNoWait</td>
<td>PickWare internal use.</td>
</tr>
<tr>
<td>pmProject_diNoWait</td>
<td>PickWare internal use.</td>
</tr>
</tbody>
</table>

*Continues on next page*
6 Runtime operation

6.3.4. Extended I/O interface

Continued

Project status values

The current status of the project is reflected by the signal \texttt{pmProject\_goStatus}.

<table>
<thead>
<tr>
<th>I/O value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Project is stopped.</td>
</tr>
<tr>
<td>1</td>
<td>Project is stopping.</td>
</tr>
<tr>
<td>2</td>
<td>Project is starting.</td>
</tr>
<tr>
<td>3</td>
<td>Project is running.</td>
</tr>
<tr>
<td>5</td>
<td>Project in error state.</td>
</tr>
</tbody>
</table>

Default height values

The following selections are supported when setting the default height.

<table>
<thead>
<tr>
<th>I/O value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Full</td>
</tr>
<tr>
<td>1</td>
<td>Latest</td>
</tr>
<tr>
<td>2</td>
<td>Empty</td>
</tr>
<tr>
<td>3</td>
<td>Value</td>
</tr>
<tr>
<td>4</td>
<td>Standard, that is, as configured in the \textit{Robot Path Height}. See \textit{Robot Path Height on page 161}.</td>
</tr>
</tbody>
</table>

Starting a project

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Ensure that the previous project is stopped, that is he signal \texttt{pmProject_goStatus} is 0.</td>
</tr>
<tr>
<td>2 Switch to motors on state using the controller system signals. See \textit{Technical reference manual - System parameters}, the topic I/O.</td>
</tr>
<tr>
<td>3 Pulse the system input \texttt{pmSystem_diLoadStart1} to load and start the project manager module. In a MultiMove system, pulse the corresponding signals, \texttt{pmSystem_diLoadStartX}, to load and start the other motion tasks.</td>
</tr>
<tr>
<td>4 Wait until robot program has started, that is when \texttt{pmSystem_doCycleOn} is set.</td>
</tr>
<tr>
<td>5 Set \texttt{pmProject_giSelection} to select the project to run. The I/O value for the project is set in the \textit{Project I/O value editor}. See \textit{Project Manager on page 98}.</td>
</tr>
<tr>
<td>6 Pulse \texttt{pmProject_diStart} to start the project.</td>
</tr>
<tr>
<td>7 Wait until the project has started, that is when \texttt{pmProject_goStatus} goes to 3.</td>
</tr>
</tbody>
</table>

Stopping robot program - halting project

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Pulse \texttt{pmSystem_diStop} to stop the robot program.</td>
</tr>
<tr>
<td>2 Wait until the robot program is stopped, which occurs when \texttt{pmSystem_doCycleOn} is reset.</td>
</tr>
</tbody>
</table>

Restarting robot program - restarting project

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Switch to motors on state using the controller system signals. See \textit{Technical reference manual - System parameters}, the topic I/O.</td>
</tr>
</tbody>
</table>

Continues on next page
Stopping current project

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Pulse pmSystem_diStop to stop the robot program.</td>
</tr>
<tr>
<td>2. Wait until the robot program is stopped, which occurs when pmSystem_doCycleOn is reset.</td>
</tr>
<tr>
<td>3. Pulse pmProject_diStop to stop the project.</td>
</tr>
<tr>
<td>4. Wait until project is stopped, which occurs when pmProject_goStatus goes to 0.</td>
</tr>
</tbody>
</table>

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).

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Set pmProject_giDefaultHeight to select the new default height.</td>
</tr>
<tr>
<td>2. Set pmProject_giDefHeightWaSel to select the work area. The I/O value of the work area is configured in the Work Area I/O Settings editor. See Edit Detailed Signals for Feeder on page 157.</td>
</tr>
<tr>
<td>3. Pulse pmProject_diSetDefaultHeight.</td>
</tr>
<tr>
<td>4. Wait until the default height is updated, that is, the work area height state GO signal is updated to reflect the change.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Flow signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pmFlow_diStart</td>
<td>Start selected flow.</td>
</tr>
<tr>
<td>pmFlow_diStop</td>
<td>Stop selected flow.</td>
</tr>
<tr>
<td>pmFlow_diRecover</td>
<td>Recover the selected flow with the selected recover action on the selected work area.</td>
</tr>
</tbody>
</table>
Flow status values

The statuses of the flows are reflected by the \textit{pmFlowX\_goStatus} signals. The status signal for the flow is setup in the Flow editor. See \textit{Flow on page 184}.

If a flow goes to error state, you can do flow recovery from the I/O Interface or the PickMaster FlexPendant interface.

<table>
<thead>
<tr>
<th>I/O value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Flow is stopped.</td>
</tr>
<tr>
<td>1</td>
<td>Flow is running.</td>
</tr>
<tr>
<td>2</td>
<td>Flow is stopping after current pick place cycle.</td>
</tr>
<tr>
<td>3</td>
<td>Flow is stopping after current layer.</td>
</tr>
<tr>
<td>4</td>
<td>Flow is stopping after current pallet/operation set.</td>
</tr>
<tr>
<td>5</td>
<td>Flow in error state.</td>
</tr>
</tbody>
</table>

Flow stop options

Stop option is specified with the signal \textit{pmFlow\_giStopOption}.

<table>
<thead>
<tr>
<th>I/O value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Finish cycle.</td>
</tr>
<tr>
<td>2</td>
<td>Finish layer.</td>
</tr>
<tr>
<td>3</td>
<td>Finish pallet/operation set.</td>
</tr>
</tbody>
</table>

Flow recover actions

A flow recover action is specified with the signal \textit{pmFlow\_giRecoverAction}.

<table>
<thead>
<tr>
<th>I/O value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Continue.</td>
</tr>
<tr>
<td>2</td>
<td>Restart layer.</td>
</tr>
<tr>
<td>3</td>
<td>Next pallet.</td>
</tr>
<tr>
<td>4</td>
<td>Redo last pick.</td>
</tr>
</tbody>
</table>

Starting or restarting a flow

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>
6 Runtime operation

6.3.4. Extended I/O interface

Continued

Stopping a flow

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Set <code>pmFlow_giSelection</code> to select flow. The I/O value for the flow is set in the Flow editor. See Flow on page 184.</td>
</tr>
<tr>
<td>2 Set <code>pmFlow_giStopOption</code> to select stop option.</td>
</tr>
<tr>
<td>3 Pulse <code>pmFlow_diStop</code> to stop the project.</td>
</tr>
<tr>
<td>4 Wait until the flow is stopped, which occurs when the flow's status signal, <code>pmFlowX_goStatus</code>, goes to 0.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Set <code>pmEvent_giErrorSource</code> to select the flow's master work area and all slave work areas. See Events on page 144, on how to setup event signals for the robot controller.</td>
</tr>
<tr>
<td>2 Pulse <code>pmEvent_diTrigger</code> to immediately generate an error on the flow and its work areas.</td>
</tr>
<tr>
<td>3 Wait until the flow is in error state, which occurs when the flow's status signal, <code>pmFlowX_goStatus</code>, goes to 5.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Set <code>pmFlow_giSelection</code> to select flow. The I/O value for the flow is configured in the Flow editor. See Flow on page 184.</td>
</tr>
<tr>
<td>2 Set <code>pmFlow_giRecoverAction</code> to select recover action.</td>
</tr>
<tr>
<td>3 Set <code>pmFlow_giWaRecoverSelection</code> to select which work area the recover action shall be applied for. The signal must not be set for the recover actions Continue and Redo last pick. The selected work area must be included in the selected flow, normally the master work area is selected. The I/O value of the work area is configured in the Work Area I/O Settings editor. See Edit Detailed Signals for Feeder on page 157.</td>
</tr>
<tr>
<td>4 Pulse <code>pmFlow_diRecover</code> to recover the flow.</td>
</tr>
<tr>
<td>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 <code>PmSetRecoverAction - Set flow recover action on page 298</code>). The flow is now prepared for being restarted. Note: the error state of the flow indicated by <code>pmFlowX_goStatus</code> will not change until the flow is restarted.</td>
</tr>
</tbody>
</table>

Continues on next page
6 Runtime operation

6.3.4. Extended I/O interface

Continued

Event and error reporting

It is possible to report events and errors for work areas, robots, and controllers that affect the runtime operation. See Events on page 144.

Default signals

The following default signals can be used for event and error reporting.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pmEvent_diTrigger</td>
<td>Generate an event.</td>
</tr>
<tr>
<td>pmEvent_giErrorSource</td>
<td>Select error source(s).</td>
</tr>
<tr>
<td>pmEvent_giMessage</td>
<td>Select elog message.</td>
</tr>
</tbody>
</table>

Example, Report an error for a work area and log an elog message

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Set the proper bit for pmEvent_giErrorSource to select a work area.</td>
</tr>
<tr>
<td>The bit representing the work area is set in the Event settings tab of</td>
</tr>
<tr>
<td>the Controller Properties. See Events on page 144.</td>
</tr>
<tr>
<td>2 Set pmEvent_giMessage to select a message. Values that represent</td>
</tr>
<tr>
<td>different messages are set in the Message Settings, see section</td>
</tr>
<tr>
<td>Messages on page 147.</td>
</tr>
<tr>
<td>3 Pulse pmEvent_diTrigger to generate the error and the elog message.</td>
</tr>
<tr>
<td>4 Wait until the work area enters error state, which occurs when the</td>
</tr>
<tr>
<td>execution state signal of the work area, pmFlowX_goExecState, gets the</td>
</tr>
<tr>
<td>value 2.</td>
</tr>
</tbody>
</table>

Example, Log an elog message

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Set the proper bit for pmEvent_giErrorSource to select a controller.</td>
</tr>
<tr>
<td>The bit that represents the controller is set in the Event settings</td>
</tr>
<tr>
<td>tab of the Controller Properties. See Events on page 144.</td>
</tr>
<tr>
<td>2 Set pmEvent_giMessage to select a message. Values that represent</td>
</tr>
<tr>
<td>different messages are set in the Message Settings, see section</td>
</tr>
<tr>
<td>Messages on page 147.</td>
</tr>
<tr>
<td>3 Pulse pmEvent_diTrigger to generate the elog message.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pmGripper1_goActivators</td>
<td>Activators control</td>
</tr>
<tr>
<td>pmGripper1_doActivator1Open</td>
<td>Activator control</td>
</tr>
<tr>
<td>pmGripper1_doActivator1Close</td>
<td>Activator control</td>
</tr>
<tr>
<td>pmGripper1_doActivator2Open</td>
<td>Activator control</td>
</tr>
<tr>
<td>pmGripper1_doActivator2Close</td>
<td>Activator control</td>
</tr>
<tr>
<td>pmGripper1_doActivator3Open</td>
<td>Activator control</td>
</tr>
<tr>
<td>pmGripper1_doActivator3Close</td>
<td>Activator control</td>
</tr>
<tr>
<td>pmGripper1_doActivator4Open</td>
<td>Activator control</td>
</tr>
<tr>
<td>pmGripper1_doActivator4Close</td>
<td>Activator control</td>
</tr>
<tr>
<td>pmGripper1_diActivator1Opened</td>
<td>Activator status</td>
</tr>
</tbody>
</table>

Continues on next page
### 6 Runtime operation

#### 6.3.4. Extended I/O interface

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pmGripper1_diActivator1Closed</td>
<td>Activator status</td>
</tr>
<tr>
<td>pmGripper1_diActivator2Opened</td>
<td>Activator status</td>
</tr>
<tr>
<td>pmGripper1_diActivator2Closed</td>
<td>Activator status</td>
</tr>
<tr>
<td>pmGripper1_diActivator3Opened</td>
<td>Activator status</td>
</tr>
<tr>
<td>pmGripper1_diActivator3Closed</td>
<td>Activator status</td>
</tr>
<tr>
<td>pmGripper1_diActivator4Opened</td>
<td>Activator status</td>
</tr>
<tr>
<td>pmGripper1_diActivator4Closed</td>
<td>Activator status</td>
</tr>
<tr>
<td>pmGripper1_diPartCheck1</td>
<td>Part check status</td>
</tr>
<tr>
<td>pmGripper1_diPartCheck2</td>
<td>Part check status</td>
</tr>
<tr>
<td>pmGripper1_diPartCheck3</td>
<td>Part check status</td>
</tr>
<tr>
<td>pmGripper1_diPartCheck4</td>
<td>Part check status</td>
</tr>
<tr>
<td>pmGripper1_diPartCheck5</td>
<td>Part check status</td>
</tr>
<tr>
<td>pmGripper1_giPartCheck1</td>
<td>Part check status</td>
</tr>
<tr>
<td>pmGripper1_goSearchActivate</td>
<td>Search tool activation.</td>
</tr>
<tr>
<td>pmGripper1_diSearchStop</td>
<td>Stack search stop trigger.</td>
</tr>
<tr>
<td>pmGripper1_goToolEvent1</td>
<td>GO tool event.</td>
</tr>
<tr>
<td>pmGripper1_goToolEvent2</td>
<td>GO tool event.</td>
</tr>
<tr>
<td>pmGripper1_goToolEvent3</td>
<td>GO tool event.</td>
</tr>
<tr>
<td>pmGripper1_goToolEvent4</td>
<td>GO tool event.</td>
</tr>
<tr>
<td>pmGripper1_goToolEvent5</td>
<td>GO tool event.</td>
</tr>
<tr>
<td>pmGripper1_doToolEvent1</td>
<td>DO tool event.</td>
</tr>
<tr>
<td>pmGripper1_diToolEvent1</td>
<td>DI tool event.</td>
</tr>
<tr>
<td>pmGripper1_giToolEvent1</td>
<td>GI tool event.</td>
</tr>
<tr>
<td>pmGripper1_giToolEvent2</td>
<td>GI tool event.</td>
</tr>
<tr>
<td>pmGripper1_giToolEvent3</td>
<td>GI tool event.</td>
</tr>
<tr>
<td>pmGripper1_giToolEvent4</td>
<td>GI tool event.</td>
</tr>
<tr>
<td>pmGripper1_giToolEvent5</td>
<td>GI tool event.</td>
</tr>
</tbody>
</table>

Continued
6 Runtime operation

6.3.5. Timing diagrams for PLC communication

6.3.5. Timing diagrams for PLC communication

Introduction to timing diagrams

Each timing diagram shows a basic example on the I/O communication between the robot controller and the PLC when running a PickMaster project. The individual updates of different I/O signals are shown related to important events of the palletizing process, for example when a pickup of a format has been completed.

The following events in the palletizing process can be found in the timing diagrams:

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operate infeeder</td>
<td>The RAPID execution has executed the RAPID procedure Operate for the infeeder.</td>
</tr>
<tr>
<td>Pick</td>
<td>The robot has picked up a complete format in the tool, that is, finished an operation.</td>
</tr>
<tr>
<td>Operate outfeeder</td>
<td>The RAPID execution has executed the RAPID procedure Operate for the outfeeder.</td>
</tr>
<tr>
<td>Place</td>
<td>The robot has placed a complete format on the work area, that is, finished an operation.</td>
</tr>
<tr>
<td>Pallet pattern unloaded</td>
<td>A finished pallet pattern leaves the working range of the robot when being transferred from an outfeeder.</td>
</tr>
</tbody>
</table>

See examples:

- [Example minimum process control on a running flow on page 273.](#)
- [Example robot execution control on page 274.](#)
- [Example height control of a running flow on page 275.](#)
- [Example flow control on page 276.](#)
### Example minimum process control on a running flow

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

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

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>doPosReqTrig</td>
<td>C</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>dtTgGenTrig</td>
<td></td>
<td></td>
<td>D</td>
</tr>
<tr>
<td>B</td>
<td>doPosAvail</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>doQueueEmpty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>doOpSetCompl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>goExecState</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>goHeightState</td>
<td></td>
<td>1 (running)</td>
<td>0 (full)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 (active)</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>doPosReqTrig</td>
<td>G</td>
<td>H</td>
<td>G</td>
</tr>
<tr>
<td></td>
<td>dtTgGenTrig</td>
<td></td>
<td></td>
<td>H</td>
</tr>
<tr>
<td>F</td>
<td>doPosAvail</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>doQueueEmpty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>doOpSetCompl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>goExecState</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>goHeightState</td>
<td></td>
<td>1 (running)</td>
<td>0 (full)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 (active)</td>
<td>0 (full)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 (active)</td>
<td>0 (full)</td>
</tr>
</tbody>
</table>

- **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

\[Continued\]
6 Runtime operation

6.3.5. Timing diagrams for PLC communication

Example robot execution control

Task: Pick single items from infeeder and place pallet pattern with two items on outfeeder, control the robot access to work areas.

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

<table>
<thead>
<tr>
<th></th>
<th>Master outfeeder, process control</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Master outfeeder, process status</td>
</tr>
<tr>
<td>C</td>
<td>Operate outfeeder</td>
</tr>
<tr>
<td>D</td>
<td>Place</td>
</tr>
<tr>
<td>E</td>
<td>Slave infeeder, process control</td>
</tr>
<tr>
<td>F</td>
<td>Slave infeeder, process status</td>
</tr>
<tr>
<td>G</td>
<td>Operate infeeder</td>
</tr>
<tr>
<td>H</td>
<td>Pick</td>
</tr>
</tbody>
</table>
Example height control of a running flow

Flow task: Pick single items from infeeder and place pallet pattern with two items on outfeeder. Control the height change of the outfeeder caused by unloading the pallet pattern to minimize the cycle time for other flows.

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

A. gSetDefaultHeight
   B. gHeightSelection
   C. doPosReqTrig
   D. doPosAvai, doQueueEmpty
   E. doOpSelCompl
   F. gHeightState
   G. doPosReqTrig
   H. doPosAvai, doQueueEmpty
   I. doOpSelCompl
   J. gHeightState
   K. gHeightState

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Project control, process control</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Pallet pattern unloaded from outfeeder</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Master outfeeder, process control</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Master outfeeder, process status</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Operate outfeeder</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Place</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Slave infeeder, process control</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Slave infeeder, process status</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Operate infeeder</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>Pick</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
6 Runtime operation

6.3.5. Timing diagrams for PLC communication

Continued

Example flow control

Task: Start a flow, pick single items from infeeder and place pallet pattern on outfeeder, stop the flow after next cycle, evacuate the unfinished pallet pattern from the outfeeder.

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

---

A Flow control, process control. In this example, the giSelection signal is constantly set to this flow.

B Master outfeeder, process control

C Master outfeeder, process status. In this example, the goExecState signal is constantly set to 1 (running).

D Operate outfeeder

E Place

F Slave infeeder, process control

G Slave infeeder, process status. In this example, the goExecState signal is constantly set to 1 (running).

H Operate infeeder

J Pick
7 RAPID reference information

7.1. Introduction to RAPID reference information

Structure of this chapter

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

7.2.1. PmAckTarget - Acknowledge a target

7.2 Instructions

7.2.1. PmAckTarget - Acknowledge a target

Usage

PmAckTarget is used to acknowledge a target.

Basic examples

IF status = OK THEN
  PmAckTarget Wa, Target, PM_ACK;
ELSE
  PmAckTarget Wa, Target, PM_NACK;
ENDIF

Arguments

PmAckTarget Wa Target Status

Wa

Data type: pm_wadescr
Contains a reference to a work area.

Target

Data type: pm_targetdata
The target that is acknowledged.

Status

Data type: pm_acktype
The acknowledge status.

Predefined data

The acknowledge status, used in argument Status, can be one of the following:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_ACK</td>
<td>The target is acknowledged as used.</td>
</tr>
<tr>
<td>PM_NACK</td>
<td>The target is acknowledged as not used.</td>
</tr>
<tr>
<td>PM_LOST</td>
<td>If the target is acknowledged as lost.</td>
</tr>
</tbody>
</table>

Syntax

PmAckTarget
[ Wa ':=' ] < expression (IN) of pm_wadescr > ','
[ Target ':=' ] < expression (IN) of pm_targetdata > ','
[ Status ':=' ] < expression (IN) of pm_acktype > ';'

Continues on next page
### Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type <strong>pm_wadescr</strong></td>
<td><em>pm_wadescr</em> - PickMaster work area reference on page 363.</td>
</tr>
<tr>
<td>The data type <strong>pm_targetdata</strong></td>
<td><em>pm_targetdata</em> - PickMaster target data on page 359.</td>
</tr>
<tr>
<td>The data type <strong>pm_acktype</strong></td>
<td><em>pm_acktype</em> - PickMaster target acknowledge type on page 328.</td>
</tr>
</tbody>
</table>
7.2.2. PmCalcArmConf - Calculates the arm configuration

**Usage**

PmCalcArmConf is used to calculate a suitable arm configuration for a robtarg, that is the robconf component of the robtarg. 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**

```
```

**RobTgt**

*Robot target*

Data type: robtarg

The robot target whose arm configuration will be calculated.

**Tool**

*Tool*

Data type: tooldata

The tool used for calculation of the robot arm configuration.

**Wobj**

*Work object*

Data type: wobjdata

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

[\cf1]

Data type: switch

An arm configuration is calculated where the axis 1 angle is limited by the arguments MaxAngle and MinAngle. A solution closer to +45 degrees is preferred for axis 1. A solution close to the input arm configuration is preferred for the other axes.

[\cf4]

Data type: switch

An arm configuration is calculated where the axis 4 angle is limited by the arguments MaxAngle and MinAngle. A solution closer to +45 degrees is preferred for axis 4. A solution close to the input arm configuration is preferred for the other axes.

*Continues on next page*
7 RAPID reference information

7.2.2. PmCalcArmConf - Calculates the arm configuration

Continued

[\cf6]

Data type: switch
An arm configuration suitable for a 4 axes palletizer robot or a 6 axes bending backwards robot is calculated. The axis 6 angle is limited by the arguments MaxAngle and MinAngle. A solution closer to +45 degrees is preferred for axis 6. A solution close to the input arm configuration is preferred for the other axes.

[\TypeB1]

Data type: switch
An arm configuration suitable for a parallel rod robot is calculated. The axis 6 angle is limited by the arguments MaxAngle and MinAngle. A solution closer to 45 degrees is preferred for axis 6. A solution close to the input arm configuration is preferred for the other axes.

[\MaxAngle]

Maximum angle
Data type: num
Maximum angle allowed for one axis. Which axis is decided by the selection of cf1, cf4, cf6 or TypeB1.

[\MinAngle]

Minimum angle
Data type: num
Minimum angle allowed for one axis. Which axis is decided by the selection of cf1, cf4, cf6 or TypeB1.

Error handling

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

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_ERR_CALCCONF</td>
<td>Failed to calculate arm configuration. A highly complex arm configuration may cause this error.</td>
</tr>
<tr>
<td>PM_ERR_AXLIM</td>
<td>Failed to calculate axis limit. The axis angle cannot be calculated due to angle limitations.</td>
</tr>
<tr>
<td>PM_ERR_LIM_VALUE</td>
<td>Wrong limitation value. The coordinate is not possible to calculate.</td>
</tr>
</tbody>
</table>

Syntax

Instruction

```
[ RobTgt ':=' ] < expression (INOUT) of robtarget > ',
[ Tool ':=' ] < expression (IN) of tooldata > ',
[ Wobj ':=' ] < expression (IN) of wobjdata >
[ '\cf1 ] | [ '\cf4 ] | [ '\cf6 ] | [ '\TypeB1 ]
[ '\MaxAngle ':=' < expression (IN) of num >]
[ '\MinAngle ':=' < expression (IN) of num >] ';
```

Continues on next page
7 RAPID reference information

7.2.2. PmCalcArmConf - Calculates the arm configuration

Continued

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type <strong>confdata</strong></td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Data types.</td>
</tr>
<tr>
<td>The data type <strong>robtarget</strong></td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Data types.</td>
</tr>
</tbody>
</table>
7 RAPID reference information

7.2.3. PmGetFlow - Get flow to execute

Usage

PmGetFlow is used to wait until any flow reports that it is ready to be executed. The instruction will return two work area references to the work areas that are ready to be executed within a flow. If several flows are ready to be executed, the process behind the instruction will return references of the highest prioritized flow. If the time-out time is not used, the instruction is blocking until any flow is ready to be executed.

Basic examples

A basic example of the instruction PmGetFlow is illustrated below.

See also More examples on page 284.

Example 1

PROC OperateSequence()
    PmGetFlow waInFeeder, waOutFeeder;
    Operate waInFeeder;
    Operate waOutFeeder;
ENDPROC

Arguments

PmGetFlow PickWa PlaceWa [MaxTime] [TimeFlag]

PickWa

Data type: pm_wadescr
Variable that is updated to refer to the pick work area of the flow that is ready to be executed.

PlaceWa

Data type: pm_wadescr
Variable that is updated to refer to the place work area of the flow that is ready to be executed.

[MaxTime]

Maximum Time

Data type: num
The maximum period of permitted waiting time, expressed in seconds. If this time runs out before the condition is met, the error handler will be called if there is one, with the error code PM_ERR_TIMEOUT. If there is no error handler, the execution will be stopped.

[TimeFlag]

Timeout Flag

Data type: bool
The output argument that contains the value TRUE if the maximum permitted waiting time runs out before the condition is met. If this argument is included in the instruction, it is not considered an error if the maximum time runs out. This argument is ignored if the MaxTime argument is not included in the instruction.
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

```rapid
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:

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_ERR_TIMEOUT</td>
<td>No flow was ready to be executed within the time-out time.</td>
</tr>
</tbody>
</table>

Syntax

```rapid
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

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type <code>pm_wadescr</code></td>
<td><code>pm_wadescr - PickMaster work area reference on page 363</code></td>
</tr>
<tr>
<td>The Robot Flow Configuration</td>
<td><code>Flow on page 184</code></td>
</tr>
</tbody>
</table>
7.2.4. PmGetFlowInfo - Get information about a specific flow

Usage

PmGetFlowInfo gets information about a flow. The flow must be in the started project.

Basic examples

A basic example of the instruction PmGetFlowInfo is illustrated below.

Example 1

TRAP TrapStartFlow
    VAR pm_flowinfo FlowInfo;
    VAR num FlowSelection;

    ! Get info from selected flow
    PmGetFlowInfo FlowSelection,FlowInfo;
ENDTRAP

Arguments

PmGetFlowInfo SelectionNumber | Name FlowInfo

SelectionNumber

Data type: num
The number that maps a specific flow with its signal value.

Name

Data type: string
The name of the flow in a started project.

FlowInfo

Data type: pm_flowinfo
Variable that holds the information about the flow.

Program execution

The program will fail with a recoverable error if the flow cannot be found. All other errors are considered to be fatal.
7 RAPID reference information

7.2.4. PmGetFlowInfo - Get information about a specific flow

Continued

More examples

Another example of how to use the instruction PmGetFlowInfo is illustrated below.

Example 2

```rapid
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:

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_ERR_FLOW_NOT_FOUND</td>
<td>No flow was found with this selection number or name.</td>
</tr>
<tr>
<td>PM_ERR_NO_RUNNING_PROJECT</td>
<td>No running project.</td>
</tr>
</tbody>
</table>

Syntax

```rapid
PmGetFlowInfo
    [SelectionNumber ":=" ] < expression (IN) of num > ','
    [[Name ":=" ] < expression (IN) of string > ','
    [FlowInfo ":=" ] < expression (VAR) of pm_flowinfo > ';
```

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type pm_flowinfo</td>
<td><a href="#">pm_flowinfo - PickMaster flow information on page 338.</a></td>
</tr>
</tbody>
</table>
7.2.5. PmGetLastWa - Get last used work area

Usage

PmGetLastWa gets the last used work area. The work area must previous have been set by the instruction PmSetLastWa.

Basic examples

A basic example of the instruction PmGetLastWa is illustrated below.

Example 1

VAR pm_wadescr WorkArea;
! Get last used work area
PmGetLastWa WorkArea;

Arguments

PmGetLastWa Workarea

Workarea

Data type: pm_wadescr
A descriptor to the last set work area.

Program execution

All errors are considered to be fatal.

Syntax

PmGetLastWa
[Workarea '=' ] < expression (VAR) of pm_wadescr > ';

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type pm_wadescr</td>
<td>pm_wadescr - PickMaster work area reference on page 363</td>
</tr>
<tr>
<td>The instruction PmSetLastWa</td>
<td>PmSetLastWa - Set last used work area on page 297</td>
</tr>
<tr>
<td>The instruction PmGetPathHeight</td>
<td>PmGetPathHeight - Get a safe path height for an intermediate movement on page 315</td>
</tr>
</tbody>
</table>
7 RAPID reference information

7.2.6. PmGetOperation - Get operation from a work area

Usage

PmGetOperation is used to get operation data from a work area.

Basic example

```
PERS wobjdata wInfeeder2 := [FALSE,TRUE,"", 
    [2180.65,1430.22,-720.753], 
    [0.00104,-0.00130,0.00039,1.00000]], [[0,0],[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:

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_ERR_TIMEOUT</td>
<td>No pm_operationdata could be fetched within the timeout time.</td>
</tr>
</tbody>
</table>

Syntax

```plaintext
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

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type pm_wadescr</td>
<td>pm_wadescr - PickMaster work area reference on page 363.</td>
</tr>
<tr>
<td>The data type pm_operationdata</td>
<td>pm_operationdata - PickMaster operation data on page 345.</td>
</tr>
</tbody>
</table>
7.2.7. PmGetProjectInfo - Get information about a specific project

Usage

PmGetProjectInfo gets information about a project. The project must be transferred to the controller.

Basic examples

A basic example of the instruction PmGetProjectInfo is illustrated below.

See also More examples on page 291.

Example 1

PROC main()
    ! Get info from select project
    PmGetProjectInfo ProjectSelection,ProjInfo;
ENDPROC

Arguments

PmGetProjectInfo SelectionNumber | Name ProjectInfo

SelectionNumber

Data type: num

The number that maps a transferred project with its signal value. See Project Manager on page 98.

Name

Data type: string

The name of a transferred project.

ProjectInfo

Data type: pm_projectinfo

Variable that holds the information about the project.

Program execution

The program will fail with a recoverable error if the project cannot be found. All other errors are considered to be fatal.
More examples

Another example of how to use the instruction `PMGetProjectInfo` is illustrated below.

Example 1

```rapid
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:

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_ERR_PROJ_NOT_FOUND</td>
<td>No project was found with this selection number or name.</td>
</tr>
</tbody>
</table>

Syntax

`PmGetProjectInfo`  
```
[SelectionNumber ':='] < expression (IN) of num> ','
| [Name ':='] < expression (IN) of string > ','
[ProjectInfo ':='] < expression (VAR) of pm_projectinfo > ';
```

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type <code>pm_projectinfo</code></td>
<td><code>pm_projectinfo - PickMaster project information on page 348.</code>,</td>
</tr>
</tbody>
</table>
7 RAPID reference information

7.2.8. PmGetWaByWobj - Get a reference to a work area using a work object data

Usage

PmGetWaByWobj gets the reference for a specified work area.
The arguments to the instruction is the work object data, that is to be connected to the work area, and the pm_wadescr.

Basic example

PERS wobjdata wInfeeder1 :=
    [FALSE, TRUE, '', [[2180.65, -1430.22, -220.753], [0.00104, -0.00130, 0.00039, 1.00000]], [[0, 0, 0], [1, 0, 0, 0]]];
VAR pm_wadescr PickWa;
PmGetWaByWobj wInfeeder1, PickWa;

Arguments

PmGetWaByWobj WObj Wa

WObj

Work Object
Data type: wobjdata
The work object data that should be searched for in all work areas.

Wa

Data type: pm_wadescr
Variable that is updated to refer to the work area that corresponds to the provided work object.

Error handling

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

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_ERR_WOBJ</td>
<td>No work area has reference to the work object data used.</td>
</tr>
</tbody>
</table>

Syntax

PmGetWaByWobj
    [ Wobj ' := ' ] < persistent (PERS) of wobjdata > ',,'
    [ Wa ' := ' ] < expression (INOUT) of pm_wadescr > ';;'

Related information

For information about | See
----------------------|---------------------------------|
The data type pm_wadescr | pm_wadescr - PickMaster work area reference on page 363
The data type wobjdata | Technical reference manual - RAPID instructions, Functions and Data types, section Data types
7.2.9. PmGetWaInfo - Get information about a specific work area

Usage

PmGetWaInfo gets information about a work area. The work area must be in the started project. This information can be used in an external user interface or as a way to get a work area descriptor from a selection number.

Basic examples

Basic examples of the instruction PmGetWaInfo are illustrated below.

Example 1

```plaintext
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:

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_ERR_WA_NOT_FOUND</td>
<td>No work area was found with this selection number or name.</td>
</tr>
<tr>
<td>PM_ERR_NO_RUNNING_PROJECT</td>
<td>No running project.</td>
</tr>
</tbody>
</table>

Continues on next page
7 RAPID reference information

7.2.9. PmGetWaInfo - Get information about a specific work area

Continued

Syntax

PmGetWaInfo

[SelectionNumber ':='] < expression (IN) of num > ','

[[Workarea ':='] < expression (VAR) of pm_wadescr > ','

[ WaInfo ':='] < expression (VAR) of pm_wainfo > ';

Related information

Here you list related information and where to find it.

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type pm_wainfo</td>
<td>pm_wainfo - PickMaster Work Area information on page 364</td>
</tr>
<tr>
<td>The data type pm_wadescr</td>
<td>pm_wadescr - PickMaster work area reference on page 363</td>
</tr>
<tr>
<td>The instruction PmGetFlow</td>
<td>PmGetFlow - Get flow to execute on page 283</td>
</tr>
<tr>
<td>The instruction PmGetWaByWobj</td>
<td>PmGetWaByWobj - Get a reference to a work area using a work object data on page 292</td>
</tr>
</tbody>
</table>
7.2.10. PmSearchAdjust - Adjust number of remaining layers

Usage

PmSearchAdjust is used after a stack search to adjust the number of remaining layers in a pallet pattern. It also updates the search frame to improve the picking accuracy for a pallet pattern or format.

Basic examples

Basic examples of the instruction PmSearchAdjust are illustrated below.

Example 1

VAR pm_wadescr PickWa;
VAR num PalletPatternHeightZ:=1097;

PmSearchAdjust PickWa, PM_SEARCH_Z, PalletPatternHeightZ;

A pallet pattern available at the specified work area is updated in the z-direction of the work object. The detected height of the pallet pattern is 1097 mm.

Example 2

VAR pm_wadescr PickWa;
VAR num FormatHeightZ:=154;

PmSearchAdjust PickWa, PM_SEARCH_Z, FormatHeightZ;

A format available at the specified work area is updated in the z-direction of the work object. The detected height of the format is 154 mm.

Arguments

PmSearchAdjust Workarea SearchType SearchPos

WorkArea

Data type: pm_wadescr
Contains a reference to a work area.

SearchType

Data type: pm_searchtype
Represents an integer with a symbolic constant for different types of searches.

SearchPos

Search Position

Data type: num
The detected size in mm of the pallet pattern or format. The size is expressed relative the work object.

Continues on next page
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:

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_ERR_PALLET_REDUCED</td>
<td>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.</td>
</tr>
<tr>
<td>PM_ERR_PALLET_EMPTY</td>
<td>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.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_SEARCH_X</td>
<td>0</td>
<td>Search was performed in the x direction of the work object.</td>
</tr>
<tr>
<td>PM_SEARCH_Y</td>
<td>1</td>
<td>Search was performed in the y direction of the work object.</td>
</tr>
<tr>
<td>PM_SEARCH_Z</td>
<td>2</td>
<td>Search was performed in the z direction of the work object.</td>
</tr>
</tbody>
</table>

Syntax

```plaintext
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.

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack search</td>
<td>Stack search on page 175.</td>
</tr>
<tr>
<td>Format frame versus work object in format operation set</td>
<td>Format frame versus work object in format operation set on page 377.</td>
</tr>
<tr>
<td>Pallet pattern versus work object in pallet pattern operation set</td>
<td>Pallet pattern frame versus work object in pallet pattern operation set on page 379.</td>
</tr>
<tr>
<td>The data type pm_wadescr</td>
<td>pm_wadescr - PickMaster work area reference on page 363.</td>
</tr>
<tr>
<td>The data type pm_searchtype</td>
<td>pm_searchtype - PickMaster stack search type on page 354.</td>
</tr>
</tbody>
</table>
7.2.11. PmSetLastWa - Set last used work area

Usage

PmSetLastWa sets the last used work area. Use the instruction PmGetLastWa to get the work area.

Basic examples

Basic examples of the instruction instruction are illustrated below.

Example 1

```plaintext
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

```plaintext
PmSetLastWa
[Workarea ':=' ] < expression (VAR) of pm_wadescr > ';'
```

Related information

Here you list related information and where to find it.

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type pm_wadescr</td>
<td>pm_wadescr - PickMaster work area reference on page 363</td>
</tr>
<tr>
<td>The instruction PmGetLastWa</td>
<td>PmGetLastWa - Get last used work area on page 287</td>
</tr>
<tr>
<td>The instruction PmGetPathHeight</td>
<td>PmGetPathHeight - Get a safe path height for an intermediate movement on page 315</td>
</tr>
</tbody>
</table>
7.2.12. PmSetRecoverAction - Set flow recover action

Usage

PmSetRecoverAction sets flow recover action before starting a flow in error state. This instruction must be used before starting a flow with use of the IO interface if the flow is in error state. The instruction also returns information that can be used in an event log message. This message describes circumstances for restarting with selected recover action.

Basic examples

Basic examples of the instruction PmSetRecoverAction are illustrated below.

Example 1

VAR pm_flowinfo FlowInfo;
PmSetRecoverAction FlowInfo.Name,PM_RECOVER_REDO_LAST_PICK;

Arguments

PmSetRecoverAction Name \WorkArea RecoverAction \EventId \Argument1 \Argument2 \Argument3 \Argument4

Name

Data type: string
The name of the flow to set recover action on.

WorkArea

Data type: pm_wadescr
The work area to perform recover action on. Mandatory if using recover action restart layer or next pallet. Not used for recover action continue and redo last pick. It is in most cases the master work area that should be chosen.

RecoverAction

Data type: num
The recover action that is performed at next flow start.

[\EventId]

Data type: num
The event message number in process domain that creates a message for the chosen recover action.

[\Argument1]

Data type: errstr
The first argument to the event log message, one space if not used.

[\Argument2]

Data type: errstr
The second argument to the event log message, one space if not used.

[\Argument3]

Data type: errstr
The third argument to the event log message, one space if not used.

Continues on next page
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:

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_ERR_WORKAREAEXPECTED</td>
<td>This recover action demands a work area.</td>
</tr>
<tr>
<td>PM_ERR_NOT_VALID_RECOVER_ACTION</td>
<td>The recover action is not one of the supported.</td>
</tr>
<tr>
<td>PM_ERR_REDO_LAST_PICK_REJECTED</td>
<td>The redo last pick recover action is rejected, no products picked.</td>
</tr>
<tr>
<td>PM_ERR_NO_RUNNING_PROJECT</td>
<td>No running project.</td>
</tr>
</tbody>
</table>

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>
7 RAPID reference information

7.2.12. PmSetRecoverAction - Set flow recover action

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>

Continues on next page
7 RAPID reference information

7.2.12. PmSetRecoverAction - Set flow recover action

Predefined data

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_RECOVER_CONTINUE_OPERATION</td>
<td>1</td>
<td>The pick-place operation will continue from where it was stopped.</td>
</tr>
<tr>
<td>PM_RECOVER_REDO_LAYER</td>
<td>2</td>
<td>The pick-place operation repeats last layer.</td>
</tr>
<tr>
<td>PM_RECOVER_NEXT_PALLET</td>
<td>3</td>
<td>The pick-place operation continue with next pallet.</td>
</tr>
<tr>
<td>PM_RECOVER_REDO_LAST_PICK</td>
<td>4</td>
<td>The pick-place operation repeats last operation.</td>
</tr>
</tbody>
</table>

Syntax

PmSetRecoverAction
[ Name ':='] < expression (IN) of string > ','
[ '\ WorkArea ':=' ] < expression (VAR) of pm_wadescr > ','
[ RecoverAction ':='] < expression (IN) of num > ','
[ '" EventId ':='] < expression (IN) of num > ','
[ '" Argument1 ':='] < expression (VAR) of errstr > ','
[ '" Argument2 ':='] < expression (VAR) of errstr > ','
[ '" Argument3 ':='] < expression (VAR) of errstr > ','
[ '" Argument4 ':='] < expression (VAR) of errstr > ';

Related information

Here you list related information and where to find it.

For information about | See
---|---
The instruction PmStartFlow | PmStartFlow - Starts a specific flow on page 302
### 7.2.13. PmStartFlow - Starts a specific flow

**Usage**

PmStartFlow starts a flow. The flow must be in the started project.

**Basic examples**

A basic example of the instruction `PmStartFlow` is illustrated below.

**Example 1**

```rapid
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:

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_ERR_NO_RUNNING_PROJECT</td>
<td>No running project.</td>
</tr>
<tr>
<td>PM_ERR_WRONG_FLOW_STATE</td>
<td>Starting a flow in error state without setting a recover action.</td>
</tr>
</tbody>
</table>

**Predefined data**

There are predefined constants for the flow status. The constants are used for setting values on flow status signals, configured in flow editor. See *Flow on page 184*.

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_FLOW_STOPPED</td>
<td>0</td>
<td>The flow is stopped</td>
</tr>
<tr>
<td>PM_FLOW_RUNNING</td>
<td>1</td>
<td>The flow is running</td>
</tr>
<tr>
<td>PM_FLOW_STOPPING_AFTER_CYCLE</td>
<td>2</td>
<td>The flow is stopping after current cycle is finished</td>
</tr>
</tbody>
</table>
7.2.13. PmStartFlow - Starts a specific flow

Syntax

```
PmStartFlow
[FlowSelector ':=' ] < expression (IN) of string > ';'
```

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The instruction PmGetFlowInfo</td>
<td>PmGetFlowInfo - Get information about a specific flow on page 285.</td>
</tr>
<tr>
<td>The instruction PmStopFlow</td>
<td>PmStopFlow - Stop a specific flow on page 305.</td>
</tr>
<tr>
<td>The instruction PmSetRecoverAction</td>
<td>PmSetRecoverAction - Set flow recovery action on page 298</td>
</tr>
<tr>
<td>The data type pm_flowinfo</td>
<td>pm_flowinfo - PickMaster flow information on page 338.</td>
</tr>
</tbody>
</table>

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 |
7 RAPID reference information

7.2.14. PmStartProj - Start a PickMaster project

Usage

PmStartProj starts a PickMaster project. When this instruction is executed, the project setup is read and all PickMaster internal parts of the project are initialized. The time of execution depends on the size of the project.

Basic example

```rapid
PmStartProj "MyPMProj";
IF PM_PROJECT_STATUS=PM_PROJECT_STARTING THEN
    WaitUntil PM_PROJECT_STATUS=PM_PROJECT_RUNNING;
ENDIF
```

Arguments

- **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.

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_PROJECT_STOPPED</td>
<td>0</td>
<td>The project is stopped.</td>
</tr>
<tr>
<td>PM_PROJECT_STOPPING</td>
<td>1</td>
<td>The project is about to stop.</td>
</tr>
<tr>
<td>PM_PROJECT_STARTING</td>
<td>2</td>
<td>The project is starting up.</td>
</tr>
<tr>
<td>PM_PROJECT_RUNNING</td>
<td>3</td>
<td>The project is running.</td>
</tr>
<tr>
<td>PM_PROJECT_ERROR</td>
<td>4</td>
<td>The project is in error state.</td>
</tr>
</tbody>
</table>

Syntax

```
PmStartProj
    [ Name ':=' ] < expression (IN) of string > ',
    [ '\ Signal ':=' ] < expression (VAR) of signalgo > ';'```

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The instruction PmStopProj.</td>
<td><a href="#">PmStopProj - Stop current project on page 307</a></td>
</tr>
</tbody>
</table>


7.2.15. PmStopFlow - Stop a specific flow

Usage

PmStopFlow stops a flow. The flow must be in the started project.

Basic examples

A basic example of the instruction PmStopFlow is illustrated below.

Example 1

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

  ! Get info from selected flow
  PmGetFlowInfo FlowSelection, FlowInfo;
  ! Stop the selected flow
  PmStopFlow FlowInfo.Name, StopOption;
ENDTRAP

Arguments

PmStopFlow Name StopOption

Name

Data type: string
Variable that refers to a flow in a started project.

StopOption

Data type: num
Variable that specifies different stop behavior.

Program execution

The program will fail with a recoverable error if no project is running. All other errors are considered to be fatal, such as wrong flow name or wrong value on StopOption.

Error handling

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

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_ERR_NO_RUNNING_PROJECT</td>
<td>No running project.</td>
</tr>
<tr>
<td>PM_ERR_INVALID_FLOW_STOP_OPTION</td>
<td>Invalid stop option.</td>
</tr>
</tbody>
</table>

Continues on next page
7 RAPID reference information

7.2.15. PmStopFlow - Stop a specific flow

Continued

Predefined data

Flow status constants
There following constants are predefined for the status of the flow. Use the constants to set values on flow status signal, configured in flow I/O settings editor. See Flow on page 184.

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_FLOW_STOPPED</td>
<td>0</td>
<td>The flow is stopped</td>
</tr>
<tr>
<td>PM_FLOW_RUNNING</td>
<td>1</td>
<td>The flow is running</td>
</tr>
<tr>
<td>PM_FLOW_STOPPING_AFTER_CYCLE</td>
<td>2</td>
<td>The flow is stopping after current cycle is finished</td>
</tr>
<tr>
<td>PM_FLOW_STOPPING_AFTER_LAYER</td>
<td>3</td>
<td>The flow is stopping after current layer is finished</td>
</tr>
<tr>
<td>PM_FLOW_STOPPING_AFTER_PALLET</td>
<td>4</td>
<td>The flow is stopping after current pallet is finished</td>
</tr>
<tr>
<td>PM_FLOW_ERROR</td>
<td>5</td>
<td>The flow is in error state</td>
</tr>
</tbody>
</table>

StopOption constants
The following constants are predefined for StopOption.

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_FLOW_FINISH_CYCLE</td>
<td>1</td>
<td>The flow will finish current cycle before stopping</td>
</tr>
<tr>
<td>PM_FLOW_FINISH_LAYER</td>
<td>2</td>
<td>The flow will finish current layer before stopping</td>
</tr>
<tr>
<td>PM_FLOW_FINISH_PALLET</td>
<td>3</td>
<td>The flow will finish current pallet before stopping</td>
</tr>
</tbody>
</table>

Syntax

PmStopFlow
[Name ':=' ] < expression (IN) of string > ','
[StopOption ':=' ] < expression (IN) of num > ';'

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The instruction PmGetFlowInfo</td>
<td>PmGetProjectInfo - Get information about a specific project on page 290.</td>
</tr>
<tr>
<td>The instruction PmStartFlow</td>
<td>PmStartFlow - Starts a specific flow on page 302.</td>
</tr>
<tr>
<td>The data type pm_flowinfo</td>
<td>pm_flowinfo - PickMaster flow information on page 338.</td>
</tr>
</tbody>
</table>
7.2.16. PmStopProj - Stop current project

Usage

PmStopProj stops the active PickMaster project.

Basic example

PmStopProj;

Syntax

PmStopProj;

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The instruction PmStartProj</td>
<td>PmStartProj - Start a PickMaster project on page 304.</td>
</tr>
</tbody>
</table>
7 RAPID reference information

7.2.17. PmWaitProjStart - Wait for any active project

Usage

PmWaitProjStart is used to wait until any project is running. The instruction can be used with a timeout; the instruction then waits the timeout time before giving the answer. The instruction is blocking until any project is started from any client.

Basic example

PmWaitProjStart \MaxTime := 5;
Check if project is started. If not, wait 5 seconds to see if the project is started during that time.

Arguments

PmWaitProjStart [\MaxTime] [\TimeFlag]

[\MaxTime]

Maximum Time
Data type: num
The maximum period of waiting time permitted, expressed in seconds. If this time runs out before the condition is met, the error handler will be called, if there is one, with the error code PM_ERR_TIMEOUT. If there is no error handler, the execution will be stopped.

[\TimeFlag]

Timeout Flag
Data type: bool
The output argument that contains the value TRUE if the maximum permitted waiting time runs out before the condition is met. If this argument is included in the instruction, it is not considered an error if the maximum time runs out. This argument is ignored if the MaxTime argument is not included in the instruction.

Error handling

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

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_ERR_TIMEOUT</td>
<td>The project was not started within the time out time</td>
</tr>
</tbody>
</table>

Syntax

PmWaitProjStart
  [ '"' MaxTime ':=< expression (IN) of num ]
  [ '"' TimeFlag ':=< variable (VAR) of bool ] ';

Related information

For information about | See
---------------------|--------------------------------------------------|
The instruction PmStartProj | PmStartProj - Start a PickMaster project on page 304.
7.3 Functions

7.3.1. PmCalcIntermid - Calculate intermediate position

Usage

This function calculates an intermediate position between two targets. If no limitations are set, the calculated position is a part of all axis movements.

Basic example

```
InterMid:=PmCalcIntermid(p10, tool2, wobj2, p20, tool1, wobj1, 0.4
\MaxAngle:=180 \MinAngle:= -180 \AngleLimAx6 \MaxY:=2200
\MinY:=2200 \MinZ:=670 \FromWa:= LastWorkArea
\ToWa:=WorkArea);
MoveJ InterMid, v1000, z200, tool0\Wobj:=wobj0;
```

Return value

Data type: robtarget

The function will return a robtarget expressed in wobj0 and tool0.

Arguments

```
PmCalcIntermid (StartRobTgt StartTool StartWobj EndRobTgt EndTool
EndWobj InterMidPart [\MaxAngle] [\MinAngle] [\AngleLimAx1]
```

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.
7 RAPID reference information

7.3.1. PmCalcIntermid - Calculate intermediate position

Continued

InterMidPart

Data type: num

The part of all the axis moves that is used as an intermediate position. If the intermediate position is in the middle of the start and end positions, the value shall be set to 0.5. The value must be between 0 and 1.

[\MaxAngle]

Data type: num

Maximum allowed axis angle on selected axis.

[\MinAngle]

Data type: num

Minimum allowed axis angle on selected axis.

[\AngleLimAx1]

Data type: num

Limit angle on axis 1.

[\AngleLimAx4]

Data type: num

Limit angle on axis 4.

[\AngleLimAx6]

Data type: num

Limit angle on axis 6.

[\MaxX]

Data type: num

Maximum allowed X-value on intermediate position.

[\MinX]

Data type: num

Minimum allowed X-value on intermediate position.

[\MaxY]

Data type: num

Maximum allowed Y-value on intermediate position.

[\MinY]

Data type: num

Minimum allowed Y-value on intermediate position.

[\MaxZ]

Data type: num

Maximum allowed Z-value on intermediate position.

[\MinZ]

Data type: num

Minimum allowed Z-value on intermediate position.
7 RAPID reference information

7.3.1. PmCalcIntermid - Calculate intermediate position

Continued

\[\text{LimitRobBase}\]

Data type: \textit{switch}

Limitations on X, Y and Z are defined in the base frame of the robot.

\[\text{LimitWorld}\]

Data type: \textit{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.

\[\text{FromWa}\]

Data type: \textit{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.

\[\text{ToWa}\]

Data type: \textit{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:

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_ERR_CALCCONF</td>
<td>Failed to calculate arm configuration. A highly complex arm configuration can cause this error.</td>
</tr>
<tr>
<td>PM_ERR_AXLIM</td>
<td>Failed to calculate axis limit. The axis angle cannot be calculated because of the angle limitations.</td>
</tr>
<tr>
<td>PM_ERR_LIM_VALUE</td>
<td>Wrong limitation value. The coordinate is not possible to calculate.</td>
</tr>
<tr>
<td>PM_ERR_PART_VAL</td>
<td>The value of the InterMidPart is not valid.</td>
</tr>
</tbody>
</table>

Continues on next page
7 RAPID reference information

7.3.1. PmCalcIntermid - Calculate intermediate position

Continued

Syntax

```c
PmCalcIntermid '('
  [ StartRobTgt ':=' ] < expression (IN) of robtarget > ','
  [ StartTool ':=' ] < expression (IN) of tooldata > ','
  [ StartWobj ':=' ] < expression (IN) of wobjdata > ','
  [ EndRobTgt ':=' ] < expression (IN) of robtarget > ','
  [ EndTool ':=' ] < expression (IN) of tooldata > ','
  [ EndWobj ':=' ] < expression (IN) of wobjdata > ','
  [ InterMidPart ':=' ] < expression (IN) of num >
  [ ' \', MaxAngle ':=' ] < expression (IN) of num >
  [ ' \', MinAngle ':=' ] < expression (IN) of num >
  [ [ ' \', AngleLimAx1 ] | [ ' \', AngleLimAx4 ] ]
  [ [ ' \', AngleLimAx6 ] ]
  [ ' \', MaxX := < expression (IN) of num >]
  [ ' \', MinX := < expression (IN) of num >]
  [ ' \', MaxY := < expression (IN) of num >]
  [ ' \', MinY := < expression (IN) of num >]
  [ ' \', MaxZ := < expression (IN) of num >]
  [ ' \', MinZ := < expression (IN) of num >]
  [ ' \', FromWa := < expression (IN) of pm_wadescr >]
  [ ' \', ToWa := < expression (IN) of pm_wadescr > ]')'
```

A function with a return value of the data type robtarget.
7.3.2. PmGetEvent - Get events for an action

Usage

PmGetEvent is used to get an event for an action on a work area.

Basic examples

```ris
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:
            WObj:=curr_WObj;
    CASE 1:
        TriggL Tgt.RobTgtPoint, Act.Speed, TriggArr{1}, Act.Zone,
            curr_Tool \WObj:=curr_WObj;
    ENDTEST
```

Return value

Data type: bool
The function will return TRUE as long as a new pm_eventdata can be delivered for the action handle.

Arguments

PmGetEvent (Wa TargetHandle ActionHandle Event)

Wa

Data type: pm_wadescr
Contains a reference to a work area.

Continues on next page
7 RAPID reference information

7.3.2. PmGetEvent - Get events for an action

Continued

**TargetHandle**

Data type: pm_targethandle
Contains a reference to a target.

**ActionHandle**

Data type: pm_actionhandle
Contains a reference to an action.

**Event**

Data type: pm_eventdata
Event data that is fetched from a work area.

**Syntax**

```
PmGetEvent (''
    [Wa ':-' ] < expression (IN) of pm_wadescr > '','
    [ TargetHandle ':-' ] < expression (IN) of pm_targethandle > '','
    [ ActionHandle ':-' ] < expression (IN) of pm_actionhandle > '','
    [ Event ':-' ] < expression (INOUT) of pm_eventdata >')'
```

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

**Related information**

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type pm_wadescr</td>
<td>pm_wadescr - PickMaster work area reference on page 363.</td>
</tr>
<tr>
<td>The data type pm_targethandle</td>
<td>pm_targethandle - PickMaster target handle on page 362.</td>
</tr>
<tr>
<td>The data type pm_actionhandle</td>
<td>pm_actionhandle - PickMaster action handle on page 332.</td>
</tr>
<tr>
<td>The data type pm_eventdata</td>
<td>pm_eventdata - PickMaster event data on page 334.</td>
</tr>
<tr>
<td>The instruction TriggEquip</td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Instructions.</td>
</tr>
<tr>
<td>The instruction TriggL</td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Instructions.</td>
</tr>
</tbody>
</table>
7.3.3. PmGetPathHeight - Get a safe path height for an intermediate movement

Usage

PmGetPathHeight returns a safe lowest path height for an intermediate movement between two work areas. None, one or both of the work areas may be defined as the home position of the robot.

The function returns the maximum height in wobj0 (mm) of the two work areas and their intermediate work areas. The height of a work area is either the active height of the products or the default height, depending on the height state of the work area, see section Height state (GO) on page 259. The home position does not have a height itself. The Safety offset is always added to the height of each work area.

The order of the work areas and the home position is defined in the Feeder Order tab, see section Feeder Order on page 146 in the line configuration. Intermediate work areas are defined as those having an order value between the order values of the two work areas. The default height is used for inactive work areas, for example empty or full work areas waiting for new targets to be generated. The default height can be adjusted in runtime, for example to indicate that a finished stack has been unloaded. See PmSetDefaultHeight - Set the default height on page 318.

Basic examples

Basic examples of the function PmGetPathHeight are illustrated below.

Example 1

```
VAR pm_wadescr waInFeeder;
VAR pm_wadescr waOutFeeder;
VAR num MinZ;
PmGetFlow waInFeeder,waOutFeeder;
! Calculate MinZ
MinZ:=PmGetPathHeight(waInFeeder,waOutFeeder);
```

Return value

Data type: num

The function returns the lowest safe path height expressed in wobj0.

Arguments

```
PmGetPathHeight (FromWa ToWa [\UseSafePosition] [\UseDefaultHeight])
```

FromWa

Data type: pm_wadescr

The work area from where the robot shall move. May also be selected as the home position, that is, a default installed work area connected to the predefined wobjdata pm_home_Wobj. See section Procedure MoveHomePos on page 196.
7  RAPID reference information

7.3.3. PmGetPathHeight - Get a safe path height for an intermediate movement

Continued

ToWa

Data type: pm_wadescr

The work area the robot will move to. May also be selected as the home position, that is a
default installed work area connected to the predefined wobjdatal pm_home_Wobj. See
section Procedure MoveHomePos on page 196.

[\UseSafePosition]

Data type: switch

Consider the heights of used safe positions for work areas having an active height. For more
information on how to configure safe positions, see Safe Targets on page 159.

[\UseDefaultHeight]

Data type: switch

Consider the default heights for work areas having an active height. For more information on
how to configure default height, see Robot Path Height on page 161.

Program execution

All errors are considered to be fatal.

More examples

Example 1

PROC MoveInterMid(VAR pm_wadescr WorkArea, VAR pm_targetdata Tgt,
   VAR pm_actiondata Act,num SafetyHeight,\num MaxAngle,\num
MinAngle,\switch MoveToEndPoint)
   CONST num IntermidPart1:=0.5;
   VAR robtarget InterMid1;
   VAR num MinZ;
   PmGetLastWa LastWorkArea;
   ! Calculate MinZ
   MinZ:=PmGetPathHeight(LastWorkArea,WorkArea\UseSafePosition)
       +Tgt.TargetTool.tframe.trans.z+SafetyHeight;
   ! Use the frame from tool0 and the load from target tool
   TempTool:=Tgt.TargetTool;
   TempTool.tframe:=tool0.tframe;! Travel distance: 50%
   InterMid1:=PmCalcIntermid(LastRobTgt,LastTool,LastWobj,
      Act.RobTgt,Tgt.TargetTool,Tgt.TargetWobj,
      IntermidPart1\MaxAngle?MaxAngle\MinAngle?MinAngle
       \AngleLimAx6\MinZ:=MinZ);
   Move\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
### 7.3.3. PmGetPathHeight - Get a safe path height for an intermediate movement

**Related information**

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type pm_wadescr</td>
<td>pm_wadescr - PickMaster work area reference on page 363</td>
</tr>
<tr>
<td>The instruction PmSetLastWa</td>
<td>PmSetLastWa - Set last used work area on page 297</td>
</tr>
<tr>
<td>The instruction PmGetLastWa</td>
<td>PmGetLastWa - Get last used work area on page 287</td>
</tr>
</tbody>
</table>
7.3.4. PmSetDefaultHeight - Set the default height

**Usage**

PmSetDefaultHeight updates the default height for a work area. It returns the new default height (mm). The height state GO signal of the work area is updated to reflect the change, see *Height state (GO) on page 259*. The signal update occurs immediately if the work area does not have an active height.

Setting a new default height is a possibility to save cycle time without decreasing the margins for collisions, especially if the project consists of many work areas and flows.

For an outfeeder the default height can be set to *Empty* after a finished stack has been unloaded. This allows the robot to make lower intermediate movements when passing over the outfeeder next times and thus saving cycle time.

For an infeeder the default height can be set to *Full* before a new stack is loaded. This will force the robot to make intermediate movements with enough height when passing over the work area.

The new default height is active until new targets have been generated (or after new default height is set).

**Basic examples**

Basic examples of the function PmSetDefaultHeight are illustrated below.

**Example 1**

```rapid
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**

```rapid
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 Robot Path Height see Robot Path Height on page 161.

[\Empty]

Data type: switch
Set the default height to Empty.

[\Full]

Data type: switch
Set the default height to Full.

[\Latest]

Data type: switch
Set the default height to the height of the latest completed operation set.

[\Value]

Data type: num
Set the default height to a specified value (mm).

NOTE!
The Safety offset defined in the Robot Path Height will always be added to the specified height.
7 RAPID reference information

7.3.5. PmGetTarget - Get target

**Usage**

PmGetTarget is used to get a target for an operation on a work area. If the optional argument OpHandle is left out, the function will return the next target without regard to the operation it belongs to.

**Basic example**

```plaintext
PmGetOperation Wa, Op;
WHILE PmGetTarget(Wa, \OpHandle:=Op.OpHandle, Tgt) DO
    WHILE PmGetTgtAction(Wa, Tgt.TargetHandle, Act) DO
        ...
    ENDFWHILE
ENDWHILE
```

**Return value**

Data type: bool
The function will return TRUE as long as a new pm_targetdata can be delivered.

**Arguments**

PmGetTarget (Wa [\OpHandle] Targets [\MaxTime] [\TimeFlag])

- **Wa**
  Data type: pm_wadescr
  Contains a reference to a work area.

- **OpHandle**
  Data type: pm_ophandle
  Contains a reference for an operation on a work area.

- **Target**
  Data type: pm_targetdata
  Target data that is fetched from a work area.

- **MaxTime**
  *Maximum Time*
  Data type: num
  The maximum period of waiting time permitted, expressed in seconds. If this time runs out before the condition is met, the error handler will be called, if there is one, with the error code PM_ERR_TIMEOUT. If there is no error handler, the execution will be stopped.

- **TimeFlag**
  *Timeout Flag*
  Data type: bool
  The output argument that contains the value TRUE if the maximum permitted waiting time runs out before the condition is met. If this argument is included in the instruction, it is not considered an error if the maximum time runs out. This argument is ignored if the MaxTime argument is not included in the instruction.
7 RAPID reference information

7.3.5. PmGetTarget - Get target

Continued

Error handling

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

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_ERR_TIMEOUT</td>
<td>No pm_targetdata could be fetched within the time out time.</td>
</tr>
<tr>
<td>PM_ERR_OPERATION_LOST</td>
<td>The pm_operationdata is not valid, probably because of a pulse on the robot execution signal.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type pm_wadescr</td>
<td>pm_wadescr - PickMaster work area reference on page 363.</td>
</tr>
<tr>
<td>The data type pm_ophandle</td>
<td>pm_ophandle - PickMaster operation handle on page 347.</td>
</tr>
<tr>
<td>The data type pm_targetdata</td>
<td>pm_targetdata - PickMaster target data on page 359.</td>
</tr>
</tbody>
</table>
7 RAPID reference information

7.3.6. PmGetTgtAction - Get target action

## Usage

PmGetTgtAction is used to get an action for a target on a work area.

### Basic examples

```
PmGetOperation Wa, Op;
WHILE PmGetTarget(Wa \OpHandle:=Op.OpHandle, Tgt) DO
  WHILE PmGetTgtAction(Wa, Tgt.TargetHandle, Act) DO
    curr_WObj := Tgt.TargetWobj;
    curr_Tool := Tgt.TargetTool;
         \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:

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_ERR_OPERATION_LOST</td>
<td>The <code>pm_operationdata</code> is not valid, probably because of a pulse on the robot execution signal.</td>
</tr>
</tbody>
</table>
7 RAPID reference information

7.3.6. PmGetTgtAction - Get target action

Continued

Syntax

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

A function with a return value of the data type bool name.

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type pm_wadescr</td>
<td>pm_wadescr - PickMaster work area reference on page 363.</td>
</tr>
<tr>
<td>The data type pm_targethandle</td>
<td>pm_targethandle - PickMaster target handle on page 362.</td>
</tr>
<tr>
<td>The data type pm_actiondata</td>
<td>pm_actiondata - PickMaster action data on page 329.</td>
</tr>
</tbody>
</table>
7.3.7. PmGetWaHeight - Get the height of a work area

Usage

PmGetWaHeight gets the current stack height of a specified work area.

Basic examples

height := PmGetWaHeight (PickWa);

Return value

Data type: num
The function returns the current height of the specified work area in mm. The height is equivalent to the z-coordinate of the current top layer expressed in the work object. If the stack is empty, zero is returned.

Arguments

PmGetWaHeight (Wa [UseSafePosition])

Wa

Data type: pm_wadescr
Contains a reference to a work area.

[UseSafePosition]

Data type: switch
If UseSafePosition is set, the height is equivalent to the maximum z-coordinate of used safe positions and the current top layer expressed in the work object.

Syntax

PmGetWaHeight '('
[ Wa '':=' ] < expression (IN) of pm_wadescr '>')'
A function with a return value of the data type num.

Related information

For information about | See
---|---
The data type pm_wadescr | pm_wadescr - PickMaster work area reference on page 363.
7.3.8. PmGetWaName - Get the name of a work area

Usage

PmGetWaName gets the name of a specified work area.

Basic examples

A basic example of the function PmGetWaName is illustrated below.

Example 1

```
VAR string waname;
waname:=PmGetWaName(WorkArea);
```

Return value

Data type: string
The function will return the name of the work area.

Arguments

PmGetWaName (WorkArea)

WorkArea

*Work Area*

Data type: pm_wadescr
Contains a reference to a work area.

Syntax

```
PmGetWaName '('
    [ WorkArea ':=' ] < expression (VAR) of pm_wadescr >')'
```

A function with a return value of the data type string.
7 RAPID reference information

7.4.1. pm_accdata - PickMaster acceleration/deceleration data

7.4 Data types

7.4.1. pm_accdata - PickMaster acceleration/deceleration data

Usage

pm_accdata is used to describe and restrict accelerations and decelerations.

Description

pm_accdata is a part of pm_actiondata and is used as input arguments to the instructions PathAccLim and AccSet. It restricts the robots acceleration and deceleration.

Components

acc

acceleration

Data type: num

Acceleration and deceleration as a percentage of the normal values. 100% corresponds to maximum acceleration. Maximum value: 100%. Input value lower than 20% gives 20% of maximum acceleration.

Used as argument Acc in AccSet.

ramp

Data type: num

The rate at which acceleration and deceleration increases as a percentage of the normal values (see instruction AccSet for more information).

Used as argument Ramp in AccSet.

acclim

acceleration limit

Data type: bool

TRUE if there is to be a limitation of the acceleration, FALSE otherwise.

Used as argument AccLim in PathAccLim.

accmax

max acceleration

Data type: num

The absolute value of the acceleration limitation in m/s².

Only used when acclim is TRUE.

Used as argument AccMax in PathAccLim.

decellim

deceleration limit

Data type: bool

TRUE if there is to be a limitation of the deceleration, FALSE otherwise.

Used as argument DecelLim in PathAccLim.
7 RAPID reference information

7.4.1. pm_accdata - PickMaster acceleration/deceleration data

Continued

decelmax

*max deceleration*

**Data type:** num

The absolute value of the deceleration limitation in m/s².

Only used when `decellim` is TRUE.

Used as argument `DecelMax` in `PathAccLim`.

**Examples**

```rapid
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**

```rapid
< dataobject of pm_accdata >
< acc of num >
< ramp of num >
< acclim of bool >
< accmax of num >
< decellim of bool >
< decelmax of num >
```

**Related information**

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type <code>pm_actiondata</code></td>
<td><code>pm_actiondata - PickMaster action data on page 329.</code></td>
</tr>
<tr>
<td>The function <code>PmGetTgtAction</code></td>
<td><code>PmGetTgtAction - Get target action on page 322.</code></td>
</tr>
<tr>
<td>The instruction <code>PathAccLim</code></td>
<td><code>Technical reference manual - RAPID Instructions, Functions and Data types, section Instructions.</code></td>
</tr>
<tr>
<td>The instruction <code>AccSet</code></td>
<td><code>Technical reference manual - RAPID Instructions, Functions and Data types, section Instructions.</code></td>
</tr>
</tbody>
</table>
7 RAPID reference information

7.4.2. pm_acktype - PickMaster target acknowledge type

7.4.2. pm_acktype - PickMaster target acknowledge type

Usage

pm_acktype is used to represent an integer with a symbolic constant for different types of acknowledgements.

Description

A pm_acktype is used to decide which type of acknowledgement should be used.

Example

PmAckTarget WorkArea, WorkArea, PM_ACK;

Predefined data

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_ACK</td>
<td>301</td>
<td>The target is acknowledged as used</td>
</tr>
<tr>
<td>PM_NACK</td>
<td>302</td>
<td>The target is acknowledged as not used</td>
</tr>
<tr>
<td>PM_LOST</td>
<td>303</td>
<td>The target is acknowledged as lost</td>
</tr>
</tbody>
</table>

Characteristics

pm_acktype is an alias data type for num and consequently inherits its characteristics.

Related information

For information about  

PmAckTarget

See

PmAckTarget - Acknowledge a target on page 278.
7.4.3. pm_actiondata - PickMaster action data

Usage

pm_actiondata specifies an action for a target.

Description

Properties for one target action.

Components

**RobTgt**

Data type: robtarget
Specifies the position of the robot and external axes.

**Type**

Data type: pm_actiontype
Specifies type of action.

**MoveType**

Data type: pm_movetype
Specifies type of movement.

**ArmConfMon**

Data type: bool
Specifies if the robot’s configuration is monitored during the movement.

**UseConc**

*Use concurrent*
Data type: bool
Specifies if concurrent program execution is used or not. See *Technical reference manual - RAPID Instructions, Functions and Data types*.

**SingAreaType**

Data type: pm_singareatype
Specifies type of interpolation mode.

**Accel**

*Acceleration and deceleration data*
Data type: pm_accdata
Restrict the robot’s acceleration and deceleration.

**Search**

Data type: pm_searchdata
Defines search type, search stop type and search signal to be used with a search movement.

Continues on next page
7 RAPID reference information

7.4.3. pm_actiondata - PickMaster action data

Continued

Speed

Data type: speeddata
Specifies the movement speed. See Technical reference manual - RAPID Instructions, Functions and Data types.

Zone

Data type: zonedata
Specifies the corner path after the movement. See Technical reference manual - RAPID Instructions, Functions and Data types.

NumOfEvents

number of events
Data type: num
Specifies number of events.

ActionHandle

Data type: pm_actionhandle
A reference to the action where this pm_actiondata was retrieved.

Examples

VAR pm_actiondata Act;

PmGetOperation WorkArea, Op;
WHILE PmGetTarget(WorkArea \OpHandle:=Op.OpHandle, Tgt) DO
  WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) DO
    ...
  ENDWHILE
ENDWHILE

Limitations

The action data members can only be set by the instruction PmGetTgtAction.

Structure

< dataobject of pm_actiondata >
< RobTgt of targetdata >
< Type of pm_actiontype >
< MoveType of pm_movetype >
< ArmConfMon of bool >
< UseConc of bool >
< SingAreaType of pm_singareatype >
< Accel of pm_accdata >
< Search of pm_searchdata >
< Speed of speeddata >
< Zone of zonedata >
< NumOfEvents of num >
< ActionHandle of pm_actionhandle >

Continues on next page
7 RAPID reference information

7.4.3. pm_actiondata - PickMaster action data

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type pm_movetype</td>
<td>pm_movetype - PickMaster movement type on page 342.</td>
</tr>
<tr>
<td>The data type pm_accd data</td>
<td>pm_accd data - PickMaster acceleration/deceleration data on page 326.</td>
</tr>
<tr>
<td>The data type pm_searchdata</td>
<td>pm_searchdata - PickMaster search data on page 352.</td>
</tr>
<tr>
<td>The function PmGetTgtAction</td>
<td>PmGetTgtAction - Get target action on page 322.</td>
</tr>
<tr>
<td>The data type zonedata</td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Data types.</td>
</tr>
<tr>
<td>The data type speeddata</td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Data types.</td>
</tr>
<tr>
<td>The instruction ConfL</td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Instructions.</td>
</tr>
<tr>
<td>Concurrent program execution</td>
<td>Technical reference manual - RAPID overview, section Synchronisation with logical instructions.</td>
</tr>
</tbody>
</table>
7 RAPID reference information

7.4.4. pm_actionhandle - PickMaster action handle

7.4.4. pm_actionhandle - PickMaster action handle

Usage

`pm_actionhandle` is used to store data about a target action.

Description

Data of the type `pm_actionhandle` contains a reference to an action.

Examples

```c
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
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

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type <code>pm_actiondata</code></td>
<td><code>pm_actiondata - PickMaster action data on page 329</code>.</td>
</tr>
<tr>
<td>The function <code>PmGetTarget</code></td>
<td><code>PmGetTarget - Get target on page 320</code></td>
</tr>
<tr>
<td>The function <code>PmGetTgtAction</code></td>
<td><code>PmGetTgtAction - Get target action on page 322</code>.</td>
</tr>
<tr>
<td>The function <code>PmGetEvent</code></td>
<td><code>PmGetEvent - Get events for an action on page 313</code>.</td>
</tr>
</tbody>
</table>
7.4.5. pm_actiontype - PickMaster action type

Usage

pm_actiontype is used to represent an integer with a symbolic constant for different types of actions.

Description

A pm_actiontype is used to decide which payload should be used for the next movement.

Example

```rapid
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

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_APPROACH_POS</td>
<td>200</td>
<td>The action is a part of the approach movement</td>
</tr>
<tr>
<td>PM_TARGET_POS</td>
<td>201</td>
<td>The action moves to the target position</td>
</tr>
<tr>
<td>PM_DEPART_POS</td>
<td>202</td>
<td>The action is a part of the depart movement</td>
</tr>
</tbody>
</table>

Characteristics

pm_actiontype is an alias type for num and thus inherits its characteristics.

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type pm_actiondata</td>
<td>pm_actiondata - PickMaster action data on page 329.</td>
</tr>
<tr>
<td>The instruction GripLoad</td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Instructions.</td>
</tr>
</tbody>
</table>
7.4.6. pm_eventdata - PickMaster event data

Usage

pm_eventdata is used to specify an event.

Description

The components in pm_eventdata are used in the instruction TriggEquip. The different components decide what kind of trigg should be used for a specific action. For example, it describes when a Gripper should be closed or opened.

Components

type

- *event type*
- Data type: pm_eventtype
- Type of event. Used to separate how to set up triggers with instruction TriggEquip. Types that are allowed are PM_EVENT_PROC, PM_EVENT_DO, and PM_EVENT.GO.

time

- Data type: num
- Input argument EquipLag in instruction TriggEquip.

dist

- *distance*
- Data type: num
- Input argument Distance in instruction TriggEquip.

procid

- *process id*
- Data type: num
- Input argument ProcID in instruction TriggEquip.

signalname

- Data type: string
- Input argument DOp or GOp in instruction TriggEquip.

Atime

- Data type: num
- Reserved for future use.

value

- Data type: num
- Input argument SetValue in instruction TriggEquip. Should not be used if a PickMaster project uses longer group signals than 23 bits.
Dvalue

Data type: dnum

Input argument SetDvalue in instruction TriggEquip. Normally used instead of value since the numerical resolution is higher. Required for PickMaster projects using long group signals, that is longer than 23 bits and up to 32 bits.

Examples

```
VAR pm_eventdata Event;

ArrSize := Dim(TriggArr,1);
WHILE PmGetEvent (WorkArea, Tgt.TargetHandle, Act.ActionHandle,
    Event) AND i <= ?rrSize DO
    TEST Event.Type
    CASE PM_EVENT_PROC:
        TriggEquip TriggArr{i}, Event.Dist, Event.Time,
        \ProcID:=Event.ProcId, Event.Value;
    CASE PM_EVENT_DO:
        GetDataVal Event.SignalName,doSignal;
        TriggEquip TriggArr{i}, Event.Dist, Event.Time,
        \DOp:=doSignal, Event.Value;
    CASE PM_EVENT_GO:
        GetDataVal Event.SignalName,goSignal;
        TriggEquip TriggArr{i}, Event.Dist, Event.Time,
        \GOp:=goSignal, Event.Value;
    ENDTEST
    Incr i;
ENDWHILE
```

Limitations

The event data members can only be set by the instruction PmGetEvent.

Structure

```
< dataobject of pm_eventdata >
< type of pm_eventtype >
< time of num >
< dist of num >
< procid of num >
< signalname of string >
< value of num >
```

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type pm_eventtype</td>
<td>pm_eventtype - PickMaster event type on page 337.</td>
</tr>
<tr>
<td>The function PmGetEvent</td>
<td>PmGetEvent - Get events for an action on page 313.</td>
</tr>
</tbody>
</table>
7 RAPID reference information

7.4.6. pm_eventdata - PickMaster event data

Continued

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The instruction TriggEquip</td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Instructions.</td>
</tr>
<tr>
<td>The instruction Triggl</td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Instructions.</td>
</tr>
</tbody>
</table>
7.4.7. pm_eventtype - PickMaster event type

Usage

pm_eventtype is used to represent an integer with a symbolic constant for different type of events.

Description

A pm_eventtype is used to decide which type of trigg event should be used.

Examples

```
TEST Event.Type
    CASE PM_EVENT_PROC:
        TriggEquip TriggArr{i}, Event.Dist, Event.Time,
        \ProcID:=Event.ProcId, Event.Value;
    CASE PM_EVENT_DO:
        GetDataVal Event.SignalName,doSignal;
        TriggEquip TriggArr{i}, Event.Dist, Event.Time,
        \DOp:=doSignal, Event.Value;
    CASE PM_EVENT_GO:
        GetDataVal Event.SignalName,goSignal;
        TriggEquip TriggArr{i}, Event.Dist, Event.Time,
        \GOp:=goSignal, Event.Value;
ENDTEST
```

Predefined data

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_EVENT_PROC</td>
<td>220</td>
<td>Process with identity ProcId should receive the event. (For internal use)</td>
</tr>
<tr>
<td>PM_EVENT_DO</td>
<td>221</td>
<td>Digital output signal is changed.</td>
</tr>
<tr>
<td>PM_EVENT_GO</td>
<td>222</td>
<td>Digital group output signal is changed.</td>
</tr>
</tbody>
</table>

Characteristics

pm_eventtype is an alias type for num and thus inherits its characteristics.

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type pm_eventdata</td>
<td>pm_eventdata - PickMaster event data on page 334.</td>
</tr>
<tr>
<td>The instruction TriggEquip.</td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Instructions.</td>
</tr>
</tbody>
</table>
7 RAPID reference information

7.4.8. pm_flowinfo - PickMaster flow information

Usage

pm_flowinfo holds information about a flow.

Description

The flow information holds data and references to everything that is needed for starting a flow or viewing user information about a specific flow.

Components

Name
Data type: string
The name of the flow.

SelectionNumber
Data type: num
The number that connects a flow with its I/O signal value. See Flow on page 184.

SignalName
Data type: string
The name of the configured status signal.

MasterWa
Data type: pm_wadescr
A work area descriptor to the master work area in the flow.

SlaveWorkAreas
Data type: pm_slavewainfo
A collection of slave work areas.

NumberOfSlaveWA
Data type: num
Number of slave work areas in the flow.

Examples

Example 1

VAR pm_flowinfo FlowInfo;
PmGetFlowInfo FlowSelection,FlowInfo;

Limitations

The flow information members can only be set by the instruction PmGetFlowInfo.

Continues on next page
7 RAPID reference information

7.4.8. pm_flowinfo - PickMaster flow information

**Structure**

< dataobject of pm_flowinfo >
< Name of string >
< SelectionNumber of num >
< MasterWa of pm_wadescr >
< SlaveWorkAreas of pm_slavewainfo >
< NumberOfSlaveWA of num >

**Related information**

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The instruction <strong>PmGetFlowInfo</strong></td>
<td><strong>PmGetFlowInfo - Get information about a specific flow on page 285.</strong></td>
</tr>
<tr>
<td>The data type <strong>pm_wadescr</strong></td>
<td><strong>pm_wadescr - PickMaster work area reference on page 363.</strong></td>
</tr>
<tr>
<td>The data type <strong>pm_slavewainfo</strong></td>
<td><strong>pm_slavewainfo - PickMaster slave work area information on page 356.</strong></td>
</tr>
</tbody>
</table>
7.4.9. pm_moduleinfo - PickMaster module information

Usage

\texttt{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 \textit{Robot Settings on page 149}.

Components

There are 10 components, named in a number series, \texttt{ModName1} to \texttt{ModName10}.

\textbf{ModName1}

Data type: \texttt{string}

The name of the file containing a RAPID module.

\textbf{...}

\textbf{ModName10}

Data type: \texttt{string}

The name of the file containing a RAPID module.

Examples

Example 1

\begin{verbatim}
VAR pm\_projectinfo ProjInfo;
PmGetProjectInfo ProjectSelection,ProjInfo;
Load \Dynamic,"\HOME:\File:=ProjInfo.Robot1.ModuleNames.ModName1;
\end{verbatim}

Limitations

The operation data members can only be set by the instruction \texttt{PmGetProjectInfo}. There can be maximum ten modules.

Structure

\begin{verbatim}
< 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 >
\end{verbatim}
### Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The instruction PmGetProjectInfo</td>
<td><em>PmGetProjectInfo - Get information about a specific project on page 290.</em></td>
</tr>
<tr>
<td>The data type pm_robotinfo</td>
<td><em>pm_robotinfo - PickMaster robot information on page 350.</em></td>
</tr>
</tbody>
</table>
7 RAPID reference information

7.4.10. pm_movetype - PickMaster movement type

7.4.10. pm_movetype - PickMaster movement type

Usage

`pm_movetype` is used to represent an integer with a symbolic constant for different type of movements.

Description

The `pm_movetype` is used to decide which type of movement instruction should be used.

Example

```plaintext
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

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_MOVE_JOINT</td>
<td>240</td>
<td>The action is a joint movement</td>
</tr>
<tr>
<td>PM_MOVE_LIN</td>
<td>241</td>
<td>The action is a linear movement</td>
</tr>
</tbody>
</table>

Characteristics

`pm_movetype` is an alias data type for `num` and consequently inherits its characteristics.

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type <code>pm_actiondata</code></td>
<td><code>pm_actiondata - PickMaster action data on page 329.</code></td>
</tr>
<tr>
<td>The instruction <code>MoveJ</code></td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Instructions.</td>
</tr>
<tr>
<td>The instruction <code>MoveL</code></td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Instructions.</td>
</tr>
</tbody>
</table>
7.4.11. pm_offsetdata - Offset data

Usage

pm_offsetdata describes an offset position.

Description

The pm_offsetdata is used to displace a robot position in x, y, and z direction and rotation around the axis.

Components

\( x \)

Data type: num
The displacement in the x-direction, in the object coordinate system.

\( y \)

Data type: num
The displacement in the y-direction, in the object coordinate system.

\( z \)

Data type: num
The displacement in the z-direction, in the object coordinate system.

\( rx \)

Data type: num
The rotation in degrees around the x-axis of the tool coordinate system.

\( ry \)

Data type: num
The rotation in degrees around the y-axis of the tool coordinate system.

\( rz \)

Data type: num
The rotation in degrees around the z-axis of the tool coordinate system.

Examples

```rapid
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.
7 RAPID reference information

7.4.11. pm_offsetdata - Offset data

Continued

Structure

< dataobject of pm_offsetdata >
< x of num >
< y of num >
< z of num >
< rx of num >
< ry of num >
< rz of num >

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type pm_actiondata</td>
<td>pm_actiondata - PickMaster action data on page 329.</td>
</tr>
<tr>
<td>The function PmGetTgtAction</td>
<td>PmGetTgtAction - Get target action on page 322.</td>
</tr>
<tr>
<td>The function Offs</td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Functions.</td>
</tr>
</tbody>
</table>
7.4.12. pm_operationdata - PickMaster operation data

Usage

pm_operationdata is used to specify an operation on a work area.

Description

The operation data holds data and references to everything that is going to be done when the robot enters a work area.

Components

operationnumber

Data type: num
The operation number is the ordinal number of the operation in the layer.

layernumber

Data type: num
The layer number is the ordinal number of the layer in the operation set.

scenenumber

Data type: num
The scene number is the ordinal number of the current operation set.

formatid

Data type: num
The format id value is defined in the Pick Setting Configuration window.

numoftargets

number of targets
Data type: num
The number of targets that are handled by the operation.

ophandle

operation handle
Data type: pm_ophandle
The operation handle is a reference to the operation.

Examples

VAR pm_operationdata Op;
PmGetOperation WorkArea, Op;

Limitations

The operation data members can only be set by the instruction PmGetOperation.
7 RAPID reference information

7.4.12. pm_operationdata - PickMaster operation data

Continued

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

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The instruction PmGetOperation</td>
<td>PmGetOperation - Get operation from a work area on page 288.</td>
</tr>
<tr>
<td>The format id value</td>
<td>The Pick Setting on page 68 Configuration.</td>
</tr>
</tbody>
</table>
7.4.13. pm_ophandle - PickMaster operation handle

Usage

pm_ophandle is used to store data about an operation.

Description

Data of the type pm_ophandle contains a reference to an operation.

Examples

```plaintext
PmGetOperation WorkArea, Op;
WHILE PmGetTarget(WorkArea OpHandle:=Op.OpHandle, Tgt) DO
  ...
ENDWHILE
```

Characteristics

pm_ophandle is a non-value data type.

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type pm_operationdata</td>
<td><a href="#">pm_operationdata - PickMaster operation data on page 345</a></td>
</tr>
<tr>
<td>The instruction PmGetOperation</td>
<td><a href="#">PmGetOperation - Get operation from a work area on page 288</a></td>
</tr>
</tbody>
</table>
7.4.14. pm_projectinfo - PickMaster project information

Usage

```
pm_projectinfo holds information about a project.
```

Description

The project information holds data and references to everything that is needed for starting a project or viewing user information about a specific project.

Components

ProjectDescription

Data type: string

The description used in project properties.

Name

Data type: string

The name of the project.

SelectionNumber

Data type: num

The number that connects a project with its I/O signal value. See Project Manager on page 98.

Robot1

Data type: pm_robotinfo

The information related to the first robot RAPID task.

Robot2

Data type: pm_robotinfo

The information related to the second robot RAPID task, used in a MultiMove system.

... 

Robot6

Data type: pm_robotinfo

The information related to the sixth robot RAPID task, used in a MultiMove system.

NumberOfFlows

Data type: num

The number of flows handled by the project.

Examples

```
VAR pm_projectinfo ProjInfo;
PmGetProjectInfo ProjectSelection,ProjInfo;
```
Limitations

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

Structure

```
< dataobject of pm_projectinfo >
  < ProjectDescription of string >
  < Name of string >
  < SelectionNumber of num >
  < Robot1 of pm_robotinfo >
  < Robot2 of pm_robotinfo >
  < Robot3 of pm_robotinfo >
  < Robot4 of pm_robotinfo >
  < Robot5 of pm_robotinfo >
  < Robot6 of pm_robotinfo >
  < NumberOfFlows of num >
```

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The instruction <code>PmGetProjectInfo</code></td>
<td><code>PmGetProjectInfo - Get information about a specific project on page 290.</code></td>
</tr>
<tr>
<td>The data type <code>pm_robotinfo</code></td>
<td><code>pm_robotinfo - PickMaster robot information on page 350.</code></td>
</tr>
</tbody>
</table>
7 RAPID reference information

7.4.15. pm_robotinfo - PickMaster robot information

Usage

pm_robotinfo holds information about a RAPID task connected to a robot.

Description

The robot information holds data and references to everything that is needed for setting up the environment for a RAPID task after starting the project.

Components

Active

Data type: bool
Defines if valid data for this robot exists.

Name

Data type: string
The name of the robot.

TaskName

Data type: string
The name of the RAPID task connected to this robot.

LoadRapid

Data type: bool
Defines if the RAPID modules should be loaded or not after the project starts.

ModuleNames

Data type: pm_moduleinfo
The RAPID modules that should be loaded after the project starts.

Examples

Example 1

VAR pm_projectinfo ProjInfo;
PmGetProjectInfo ProjectSelection,ProjInfo;
IF projInfo.Robot1.LoadRapid THEN
   LoadAllModules projInfo.Robot1.ModuleNames, projInfo.Name, projInfo.Robot1.TaskName;
ENDIF

Limitations

The robot information members can only be set by the instruction PmGetProjectInfo.

Continues on next page
7 RAPID reference information

7.4.15. pm_robotinfo - PickMaster robot information

Continued

Structure

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

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The instruction PmGetProjectInfo</td>
<td>PmGetProjectInfo - Get information about a specific project on page 290.</td>
</tr>
<tr>
<td>The data type pm_projectinfo</td>
<td>pm_projectinfo - PickMaster project information on page 348.</td>
</tr>
<tr>
<td>The data type pm_moduleinfo</td>
<td>pm_moduleinfo - PickMaster module information on page 340.</td>
</tr>
</tbody>
</table>
7 RAPID reference information

7.4.16. pm_searchdata - PickMaster search data

7.4.16. pm_searchdata - PickMaster search data

Usage

pm_searchdata is used to store data used for search movements.

Description

pm_searchdata is a part of pm_actiondata and is used as input argument to procedure DoSearch. It defines search specific data.

Components

searchtype

Data type: pm_searchtype
Type of search (x, y or z). Is used as argument in PmSearchAdjust.

stoptype

Data type: pm_stoptype
Search stop type. Defines the robot stop method when performing a search movement.

signalname

Data type: string
Search signal name. The name of the signal to supervise during the search movement.

iotriggtype

Data type: pm_iotriggtype
I/O trigger type. Defines how the search signal is triggered, e.g. trigger on both flanks.

Example

VAR pm_actiondata Act;
VAR pm_stoptype my_stoptype;

WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) DO
  my_stoptype := Act.search.stoptype;
  ...
ENDWHILE

Limitations

The pm_searchdata members can only be set by the instruction PmGetTgtAction.

Structure

<dataobject of pm_searchdata>
  <searchtype of pm_searchtype>
  <stoptype of pm_stoptype>
  <signalname of string>
  <iotriggtype of pm_iotriggtype>
Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type pm_actiondata</td>
<td>pm_actiondata - PickMaster action data on page 329.</td>
</tr>
<tr>
<td>The function PmGetTgtAction</td>
<td>PmGetTgtAction - Get target action on page 322.</td>
</tr>
<tr>
<td>The instruction DoSearch</td>
<td>Procedure PmDoSearch on page 224.</td>
</tr>
</tbody>
</table>
7 RAPID reference information

7.4.17. pm_searchtype - PickMaster stack search type

7.4.17. pm_searchtype - PickMaster stack search type

Usage

pm_searchtype is used to represent an integer with a symbolic constant for different types of stack search types.

Description

A pm_searchtype is used to specify which type of stack search type is used.

Example

PmSearchAdjust PickWA, PM_SEARCH_Z, 570;

Predefined data

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_SEARCH_X</td>
<td>0</td>
<td>Stack search in the x direction</td>
</tr>
<tr>
<td>PM_SEARCH_Y</td>
<td>1</td>
<td>Stack search in the y direction</td>
</tr>
<tr>
<td>PM_SEARCH_Z</td>
<td>2</td>
<td>Stack search in the z direction</td>
</tr>
</tbody>
</table>

Characteristics

pm_searchtype is an alias data type for num and consequently inherits its characteristics.

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The instruction PmSearchAdjust</td>
<td>PmSearchAdjust - Adjust number of remaining layers on page 295.</td>
</tr>
</tbody>
</table>
7.4.18. pm_singareatype - PickMaster interpolation type around singular points

Usage

`pm_singareatype` is used to represent an integer with a symbolic constant for different types of interpolation around singular points.

Description

The `pm_singareatype` is used to decide how the robot is to move in the proximity of singular points.

Examples

```plaintext
IF Act.SingAreaType = PM_SING_AREA_OFF THEN
   SingArea\Off;
ELSE
   SingArea\Wrist;
ENDIF
```

Predefined data

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_SING_AREA_OFF</td>
<td>260</td>
<td>The tool orientation is not allowed to differ.</td>
</tr>
<tr>
<td>PM_SING_AREA_WRI</td>
<td>261</td>
<td>The tool orientation is allowed to differ somewhat to avoid wrist singularity.</td>
</tr>
</tbody>
</table>

Characteristics

`pm_singareatype` is an alias type for `num` and thus inherits its characteristics.

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type <code>pm_actiondata</code></td>
<td><a href="#">pm_actiondata - PickMaster action data on page 329.</a></td>
</tr>
<tr>
<td>The instruction <code>SingArea</code></td>
<td><a href="#">Technical reference manual - RAPID Instructions, Functions and Data types, section Instructions.</a></td>
</tr>
</tbody>
</table>
7 RAPID reference information

7.4.19. pm_slavewainfo - PickMaster slave work area information

Usage

pm_slavewainfo holds information about the slave work areas in a flow.

Description

The slave work area information is a collection of work areas that can be used in any PickMaster function or instruction that demands pm_wadescr. The maximum number of slaves is 19. Current number of slaves is specified with NumberOfSlaveWA in pm_flowinfo.

Components

There are 19 components, named in a number series, SlaveWa1 to SlaveWa19.

SlaveWa1

Data type: pm_wadescr

The descriptor to a work area.

... 

SlaveWa19

Data type: pm_wadescr

The descriptor to a work area.

Examples

Example 1

VAR pm_flowinfo FlowInfo;
VAR num height;
VAR num FlowSelection;
PmGetFlowInfo FlowSelection,FlowInfo;
height:=PmGetWaHeight(FlowInfo.SlaveWorkAreas.SlaveWa1);

Limitations

The slave work area information members can only be set by the instruction PmGetFlowInfo.

Continues on next page
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

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The instruction PmGetProjectInfo</td>
<td>PmGetProjectInfo - Get information about a specific project on page 290.</td>
</tr>
<tr>
<td>The data type pm_flowinfo</td>
<td>pm_flowinfo - PickMaster flow information on page 338.</td>
</tr>
</tbody>
</table>
7 RAPID reference information

7.4.20. pm_stoptype - PickMaster stop type

7.4.20. pm_stoptype - PickMaster stop type

Usage

pm_stoptype is used to represent an integer with a symbolic constant for different types of stop methods.

Description

A pm_stoptype is used to decide which type of robot stop method that shall be used. pm_stoptype is a part of pm_searchdata.

Example

VAR pm_searchdata SearchData;
VAR signaldi diSearchSignal;
VAR pm_stoptype StopType;
...
GetDataVal SearchData.SignalName,diSearchSignal;
StopType := SearchData.StopType;
TEST StopType
  CASE PM_STOP_NOT_USED:
    ...
  CASE PM_STOP:
    ...
  CASE PM_PSTOP:
    ...
  CASE PM_PSTOP:
    ...

Predefined data

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_STOP_NOT_USED</td>
<td>0</td>
<td>No stop is being used</td>
</tr>
<tr>
<td>PM_STOP</td>
<td>1</td>
<td>Robot stiff stop</td>
</tr>
<tr>
<td>PM_PSTOP</td>
<td>2</td>
<td>Robot path stop</td>
</tr>
<tr>
<td>PM_SSTOP</td>
<td>3</td>
<td>Robot soft stop</td>
</tr>
</tbody>
</table>

Characteristics

pm_stoptype is an alias data type for num and consequently inherits its characteristics.

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>How pm_stoptype is used in the procedure DoSearch</td>
<td>Procedure PmDoSearch on page 224.</td>
</tr>
<tr>
<td>The data type pm_searchdata</td>
<td>pm_searchdata - PickMaster search data on page 352.</td>
</tr>
</tbody>
</table>
7.4.21. pm_targetdata - PickMaster target data

Usage

pm_targetdata specifies a target for a work area.

Description

The target data holds data and a reference to a target.

Components

TargetNumber

Data type: num
Defines the ordinal number of the target.

OperationNumber

Data type: num
Defines the ordinal number of the operation.

LayerNumber

Data type: num
Defines the ordinal number of the layer.

SceneNumber

Data type: num
Defines the ordinal number of the scene.

RobTgtPoint

Data type: robtarget
Defines the position of the robot and external axes.

TargetTool

Data type: tooldata
Defines the tool used with the target.

TargetWobj

Data type: wobjdata
Defines the work object used with the target.

StopPointData

Data type: stoppointdata
Defines the stoppointdata to be used when moving to the position of the target.

NumOfAppProds

Data type: num
The number of products held by the robot tool when approaching the target position.
7 RAPID reference information

7.4.21. pm_targetdata - PickMaster target data

Continued

AppProdsLoad

*approach product load*

Data type: loaddata

Defines the load attached to the robot tool when approaching the target position.

NumOfDepProds

*number of depart products*

Data type: num

The number of products held by the robot tool when departing from the target position.

DepProdsLoad

*depart product load*

Data type: loaddata

Defines the load attached to the robot when departing from the target position.

NumOfActions

*number of actions*

Data type: num

Specifies the number of target actions.

TargetHandle

Data type: pm_targethandle

A reference to the target from where this pm_targetdata was retrieved.

ProductHeight

Data type: num

The height of the product to place or pick. It is not necessary a product in the tool for this target.

Examples

```rapid
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.
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

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The instruction PmGetTarget</td>
<td>PmGetTarget - Get target on page 320.</td>
</tr>
<tr>
<td>The data type robtarget</td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Data types.</td>
</tr>
<tr>
<td>The data type tooldata</td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Data types.</td>
</tr>
<tr>
<td>The data type wobjdata</td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Data types.</td>
</tr>
<tr>
<td>The data type stoppointdata</td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Data types.</td>
</tr>
<tr>
<td>The data type loaddata</td>
<td>Technical reference manual - RAPID Instructions, Functions and Data types, section Data types.</td>
</tr>
</tbody>
</table>
7.4.22. pm_targethandle - PickMaster target handle

**Usage**

`pm_targethandle` is used to store data about a target.

**Description**

Data of the type `pm_targethandle` contains a reference to a target.

**Example**

```plaintext
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**

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The data type <code>pm_targetdata</code></td>
<td><code>pm_targetdata - PickMaster target data on page 359.</code></td>
</tr>
<tr>
<td>The function <code>PmGetTarget</code></td>
<td><code>PmGetTarget - Get target on page 320.</code></td>
</tr>
<tr>
<td>The function <code>PmGetTgtAction</code></td>
<td><code>PmGetTgtAction - Get target action on page 322.</code></td>
</tr>
</tbody>
</table>
7.4.23. pm_wadescr - PickMaster work area reference

Usage

pm_wadescr is used to store data about a work area.

Description

Data of the type pm_wadescr contains a reference to a work area.

Examples

PERS wobjdata wInfeeder1 :=
[FALSE, TRUE, ",", [[2180.65, -1430.22, -220.753],
  [0.00104, -0.00130, 0.00039, 1.00000]], [[0, 0, 0],
  [1, 0, 0, 0]]];

VAR pm_wadescr PickWa;
PmGetWaByWobj wInfeeder1, PickWa;

Characteristics

pm_wadescr is a non-value data type.

Related information

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The function PmGetWaByWobj</td>
<td>PmGetWaByWobj - Get a reference to a work area using a work object data on page 292.</td>
</tr>
</tbody>
</table>
# 7.4.24. pm_wainfo - PickMaster Work Area information

## Usage

```
pm_wainfo holds information about a work area.
```

## Description

The work area information holds data and references that can be used for viewing user information about the specific work area.

## Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type: string</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>string</td>
<td>The name of the work area.</td>
</tr>
<tr>
<td>SelectionNumber</td>
<td>num</td>
<td>The number that connects a work area with its I/O signal value. See I/O values on page 97.</td>
</tr>
<tr>
<td>Workarea</td>
<td>pm_wadescr</td>
<td>The work area descriptor.</td>
</tr>
<tr>
<td>TaskNo</td>
<td>num</td>
<td>The RAPID task number this work area is connected to. Only valid after the first call to PmGetTarget for this work area.</td>
</tr>
<tr>
<td>Order</td>
<td>num</td>
<td>The configured order for this work area. See Feeder Order on page 146.</td>
</tr>
<tr>
<td>DefaultHeight</td>
<td>num</td>
<td>The configured default height. See Robot Path Height on page 161.</td>
</tr>
<tr>
<td>DiTgtGenTrig</td>
<td>string</td>
<td>The name of the configured target generation trigger signal.</td>
</tr>
<tr>
<td>DoPosRegTrig</td>
<td>string</td>
<td>The name of the configured position request trigger signal.</td>
</tr>
<tr>
<td>GoProdSel</td>
<td>string</td>
<td>The name of the configured target generation product selection signal.</td>
</tr>
</tbody>
</table>

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.

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

```xml
<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

<table>
<thead>
<tr>
<th>For information about</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The instruction <code>PmGetWaInfo</code></td>
<td><code>PmGetWaInfo - Get information about a specific work area on page 293</code></td>
</tr>
<tr>
<td>The data type <code>pm_wadescr</code></td>
<td><code>pm_wadescr - PickMaster work area reference on page 363</code></td>
</tr>
</tbody>
</table>
8 Relationships between PickMaster frames

8.1. Introduction

Structure of this chapter
This chapter describes the different frames used by PickMaster and how they relate to each other. Each section describes the relationship between two (or several) frames.
8 Relationships between PickMaster frames

8.2. Shape frame

8.2. Shape frame

Shape size

The size of the shape is specified as X size, Y size and Z size in the Product/Pallet/Sheet Configuration window.

Visualization of the shape frame

The shape frame is shown in the Product/Pallet/Sheet Configuration window, but without reference to any other frame.

In the windows Pick Setting Configuration, Pallet Pattern Operation Set Configuration and Group Operation Set Configuration the origin of the shape frame is marked as a blue corner on the shape.
8 Relationships between PickMaster frames

8.3. Format frame versus shape frame

Format frame settings

The relationship between a format frame and shape frames are defined in the Pick Setting Configuration by the settings of Orientation, Row, and Column. See the Pick Setting on page 68 configuration.

Orientation defines the orientation relationship between the format frame and the shape frames. Possible values are Front, Left, Back, and Right.

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

Column defines number of shapes in the x-direction of the format frame.

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

Orientation explanation

Orientation defines which side of the item to place in the negative y-direction.

The following examples show some different orientations.

Orientation: Front

| \( X_F, Y_F, Z_F \) | Coordinates for the format frame |
| \( X_S, Y_S, Z_S \) | Coordinates for the shape frame |
| F | Front side of the shape |
| R | Right side of the shape |
| L | Left side of the shape |
| B | Back side of the shape |
| U | Upper side of the shape |

Continues on next page
8 Relationships between PickMaster frames

8.3. Format frame versus shape frame

Continued

Orientation: Left

Orientation: Back
8 Relationships between PickMaster frames

8.3. Format frame versus shape frame

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

![Diagram showing Row: 3; Column: 1; Orientation: Front]

Row: 3 ; Column: 1; Orientation: Left

![Diagram showing Row: 3; Column: 1; Orientation: Left]
8 Relationships between PickMaster frames

8.3. Format frame versus shape frame

Continued

**Column explanation**

Column defines number of shapes in the x-direction of the format frame.

The following example shows the combination of row, column, and orientation.

Row: 3; Column: 2; Orientation: Front

Visualization of the format frame

The format frame is shown in the Pick Setting Configuration window. For more information, see the Pick Setting on page 68 configuration.
8.4. Pallet pattern frame versus shape frame

Use a palletizing area shape

The pallet pattern frame coincides with the shape frame of the palletizing area shape, which is selected in the Layout Configuration (only shapes with Form set to Pallet can be selected).

The palletizing area shape can be used as pallet in the pallet pattern but this is not mandatory. The palletizing area shape sets the x-y-size of the pallet pattern. Only box shapes and sheet shapes that are smaller than or have the same size as the palletizing area shape will be available for use within the pallet pattern.

Where to see the shape frame and pallet pattern

The location of the shape frame can be viewed in the Product/Pallet/Sheet Configuration window.

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

Illustration

<table>
<thead>
<tr>
<th>XX0700000263</th>
<th>Coordinates for the shape frame of the palletizing area shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>XP, ZP</td>
<td>Coordinates for the pallet pattern frame</td>
</tr>
</tbody>
</table>
8 Relationships between PickMaster frames

8.5. Format frame versus tool frame

Illustration

Format frame

The format frame is defined in the Item Group Configuration window, under Product/Pallet/SHEET window, shown in the 3D view.

Item frame

The offset frame is selected by setting Aligned item in the Pick Setting Configuration window, under Tool location. The offset frame is positioned on the top of the selected item and has the same orientation as the format frame.

Offset frame

The offset frame has an offset relative to the item frame. This offset is selected by setting Offset in the Pick Setting Configuration window, under Tool location. The offset frame’s z-direction is always opposite to the item frame. The orientation of the offset frame’s xy-plane relative to the item frame can be selected in steps of 90 degrees in the Pick Setting Configuration window, under Tool location.

Tool frame

The zone frame cannot be directly viewed in the Pick Setting Configuration window. Instead, the location of the tool frame is displayed. The tool frame position is different from the zone frame but the orientation is the same.

Zone frame

The zone frame is selected by setting aligned tool configuration and zone in the Pick Setting Configuration window, under Tool location. The zone frame has an offset relative to the tool frame. The location of the zone frames and the corresponding offsets are displayed in the Tool Function(Vacuum) window, under the Zones tab. The offset can be changed by selecting Activator in Zones, or by setting the activator x, y, z in the Tool Function(Vacuum) window, under Activators tab.

Continues on next page
8.5. Format frame versus tool frame

Zone frame and offset frame
When a format is being picked by a robot, a zone frame coincides with an offset frame.

Where to see the tool frame
The location of the tool frame relative to a format can be viewed in the Pick Setting Configuration window. See the Pick Setting on page 68 configuration.
8.6. Pallet pattern item versus tool frame

**Tool location**

The location of the tool relative to a specific item when picking/placing it, is automatically solved by PickMaster. The location of the tool is highly dependent on the selected **Formats to use** in the Pallet Pattern Operation Set Configuration. Every used format creates a number of possible positions of the tool relative to the item. The solution must also consider that every layer must be completely finished before starting with the next layer in the pallet pattern.

The autogenerated solution can be modified in detail in the Operation Editor, which is accessible from the Pallet Pattern Operation Set Configuration. For each item in every layer is it possible to modify:

- 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**

The location of the tool frame relative to an item in a pallet pattern can be viewed in the Pallet Pattern Operation Set Configuration window. See Pattern/Stack Operation Set on page 167.

**Related information**

The Operation Editor on page 179.
8 Relationships between PickMaster frames

8.7. Format frame versus work object in format operation set

Overview

The expected location of a format relative to the work object while being picked or placed depends on a chain of frames, starting with the work object and ending with the format frame.

Illustration

Work object

The location of the work object is defined by the selection of Work object in the Feeder Configuration window.

The location of the work object is affected by all settings in the selected work object, including user frame and object frame settings. It is also affected by use of program displacement.

In PzPP, the position of the work object is aligned with the selected hotspot of the feeder model (Feeder hotspot configuration window).

Tune frame

The location of the tune frame relative to the work object is defined by the current online tuning of the work area. By default the tune frame coincides with the work object. The tune frame location may be changed from the PickMaster RC online tuning by updating the following work area properties: Disp offs x, Disp offs y, Disp offs z, and Disp angle z. The tune frame is first displaced from the work object with the Disp offs vector. Then the tune frame is rotated around the displaced origo in the z-direction with the Disp angle z.

Work area frame

The location of the work area frame relative to the tune frame is defined by the Alignment and Rotation settings of the selected hotspot used by the feeder.

Displacement frame

The location of the displacement frame relative to the work area frame is defined by the settings of Displacement frame: x, y, z and z angle in the Group Operation Set Configuration window.

Continues on next page
8 Relationships between PickMaster frames

8.7. Format frame versus work object in format operation set

Continued

Search frame
Search frame has an offset relative to the displacement frame. The offset is initially zero but is automatically updated after a stack search.

Format frame
At pick/place on a work area, the format frame coincides with the search frame.

Where to see the work object
The location of the work object relative to a general format can be viewed in the Feeder Hotspot window. However, the location is only valid if the following are zero: tune frame location, displacement frame offset, reorientation, and search frame.

The location of the work object relative to a specific format can be viewed in the Group Operation Set Configuration window. However, the location is only valid if the tune frame location is zero (=default) and if the search frame offset is zero.
8.8. Pallet pattern frame versus work object in pallet pattern operation set

Overview

The expected location of a pallet pattern frame relative to the work object while being picked or placed depends on a chain of frames, starting with the work object and ending with the pallet pattern frame.

Illustration

Work object

The location of the work object is defined by the selection of Work object in the Feeder Configuration window.

The location of the work object is affected by all settings in the selected work object, including user frame and object frame settings. It is also affected by use of program displacement.

In PzPP, the position of the work object is aligned with the selected hotspot of the feeder model (Feeder hotpot configuration window).

Tune frame

The location of the tune frame relative the work object is defined by the current online tuning of the work area. By default the tune frame coincides with the work object. The tune frame location may be changed from the PickMaster RC online tuning by updating the following work area properties: Disp offs x, Disp offs y, Disp offs z, and Disp angle z. The tune frame is first displaced from the work object with the Disp offs vector. Then the tune frame is rotated around the displaced origo in the z-direction with the Disp angle z.

Work area frame

The location of the work area frame relative to the tune frame is defined by the Alignment and Orientation settings of the selected hotspot used by the feeder.

Continues on next page
8 Relationships between PickMaster frames

8.8. Pallet pattern frame versus work object in pallet pattern operation set

Continued

Displacement frame

The location of the displacement frame relative to the work area frame is defined by the settings of Displacement frame: x, y, z and z angle in the Pallet Pattern Operation Set Configuration window.

Search frame

Search frame has an offset relative to the displacement frame. The offset is initially zero but is automatically updated after a stack search.

Format frame

At pick/place on a work area, the pallet pattern frame coincides with the search frame.

Where to see the work object

The location of the work object relative to a general pallet pattern can be viewed in the Feeder Hotspot window. However, the location is only valid if the following are zero: the tune frame location, displacement frame offset, reorientation, and the search frame offset.

The location of the work object relative to a specific pallet pattern can be viewed in the Pallet Pattern Operation Set Configuration window. However, the location is only valid if the tune frame location is zero (=default) and if the search frame offset is zero.
8.9. Tune frame versus work area frame

**Formats and pallet patterns**

The relationship between tune frame and work area frame is valid for both formats and pallet patterns.

**Alignment**

Alignment defines which side of the work area frame the tune frame is found on, left or right. The offset of the work area frame is zero with left alignment. With right alignment, the offset becomes equal to the x-size of the format expressed in the format frame.

**Orientation**

Orientation defines the orientation of the tune frame in the z-direction relative to the work area frame: 0, 90, 180 or 270 degrees.

**Examples**

The displayed positions of the format in these examples assumes no displacement frame offset/reorientation and zero search offset.

**Alignment: Left; Orientation: 0**

![Diagram](image1)

**Alignment: Right; Orientation: 0**

![Diagram](image2)

<table>
<thead>
<tr>
<th>$X_T$, $Y_T$, $Z_T$</th>
<th>Coordinates for the tune frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_{WA}$, $Y_{WA}$, $Z_{WA}$</td>
<td>coordinates for the work area frame</td>
</tr>
</tbody>
</table>
8 Relationships between PickMaster frames

8.9. Tune frame versus work area frame

Continued

Alignment: Left; Orientation: 90

Alignment: Right; Orientation: 90
8.10. Tune frame versus work area frame versus displacement frame

Displacement frame settings

Displacement frame settings are made in the Group Operation Set Configuration window for formats and in the Pattern Operation Set Configuration window for pallet patterns. 

\( x, y, \) and \( z \) defines an offset of the displacement frame relative to the work area frame but in the direction of the tune frame. \( z \) angle defines a \( z \)-rotation of the displacement frame relative the displaced origin.

Example

This example shows the coordinates for the tune frame, the work area frame and the displacement frame. It also shows how the items in the format are placed.

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

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment</td>
<td>Right</td>
</tr>
<tr>
<td>Orientation</td>
<td>270</td>
</tr>
</tbody>
</table>

The following settings are made for the displacement frame in the Pattern Operation Set Configuration window:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>0</td>
</tr>
<tr>
<td>( y )</td>
<td>1500</td>
</tr>
<tr>
<td>( z )</td>
<td>0</td>
</tr>
<tr>
<td>( z ) angle</td>
<td>90°</td>
</tr>
</tbody>
</table>

The displayed position of the format in this example assumes zero search frame offset.
8 Relationships between PickMaster frames

8.10. Tune frame versus work area frame versus displacement frame
waOutFeeder1 194
reachability 78
redo last pick 242
report 100
restart layer 242
restart options 241
robots
  bending backwards 21
  parallel rod 25
  using 6 axes 21

S
safety 9
search frame 377, 379
shape frame 368, 369, 373
sheet 51
simulation 82, 83, 87, 88, 89, 91
slip sheets
  using 16
smartcomponent 132, 133

T
target frame 374
timing diagrams 272
tool 47, 151
tool frame 374, 376
tuning
  FlexPendant 249

W
work area frame 377, 379, 381, 383
work object 377, 379, 381, 383
upload 95

Z
zone frame 374