

ROBOTICS

Operating manual

Palletizing PowerPac



Trace back information: Workspace Main version a366 (not checked in) Published 2020-09-30 at 11:44:17 Skribenta version 5.3.075

Operating manual Palletizing PowerPac

RobotStudio 2020.2

Document ID: 3HAC042340-001 Revision: R

© Copyright 2012-2020 ABB. All rights reserved. Specifications subject to change without notice.

The information in this manual is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this manual.

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

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

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

Keep for future reference.

Additional copies of this manual may be obtained from ABB.

Original instructions.

© Copyright 2012-2020 ABB. All rights reserved. Specifications subject to change without notice.

Table of contents

Overv	/iew of the manual	9
Safet	у	13
Introduction 1		15
2.1 2.2 2.3 2.4 2.5	Overview About Palletizing PowerPac The palletizing process Terms and concepts Using a robot with 6 axes	15 16 18 22 25
Insta	lation	31
3.1	System requirements	31
Work	flow	35
4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 4.10 4.11 4.12 4.13 4.14	Introduction Preparing your controller for PickMaster How to add tool How to add feeders How to create or modify product data How to set pick setting How to set pick setting How to add job How to do simulation How to do simulation How to adjust cell layout to achieve reachability How to transfer the project How to start production How to Pack Palletizing Project Data to Controller How to UnPack Palletizing Project Data from Controller	35 36 37 38 39 40 41 42 43 44 45 46 47 48
		40 49
5.1 5.2 5.3	5.3.15 Download 5.3.16 Upload 5.3.17 Project overview 5.3.18 I/O interface 5.3.19 Project report 5.3.20 Pack data 5.3.21 UnPack data	101 102 104 106 107 110 111 114
	Safety Introc 2.1 2.2 2.3 2.4 2.5 Instal 3.1 Work 4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 4.10 4.11 4.12 4.13 4.14 User 5.1 5.2	2.2 About Palletizing Process 2.3 The palletizing process 2.4 Terms and concepts 2.5 Using a robot with 6 axes Installation

			Import PM5	117
				119
			Models IOPanel	
			Change viewpoint	
			Adjust position of object	144
			Help	145
	5.4	Layout	browser	
		5.4.1	Introduction	
		5.4.2	Station	
		5.4.3 5.4.4	Robot Tool SmartComponent	
		5.4.5	Feeder SmartComponent	
		5.4.6	Part	151
	5.5		mming browser	
		5.5.1	Introduction	
		5.5.2	Project	
		5.5.3	Controller	158 158
			5.5.3.2 Robot	164
		5.5.4	Tool	167
		5.5.5	Feeders	
			5.5.5.1 Context Menus from the Feeders Node	
			5.5.5.2 Feeder	
		5.5.6	5.5.5.3 Operation set Flows	
		5.5.0	5.5.6.1 Flows	
			5.5.6.2 Flow	
		5.5.7	Job	
~				
6	RAPI	D proar	am	205
6		D progr		205 205
6	6.1 6.2	Introdu		205
6	6.1	Introdu	iction ew Relationship between RAPID execution and PickMaster project	205 206 206
6	6.1 6.2	Introdu Overvie 6.2.1 6.2.2	iction ew Relationship between RAPID execution and PickMaster project RAPID modules overview	205 206 206 208
<u>6</u>	6.1	Introdu Overvie 6.2.1 6.2.2 Progra	iction ew Relationship between RAPID execution and PickMaster project RAPID modules overview m module templates	205 206 206 208 209
<u>6</u>	6.1 6.2	Introdu Overvie 6.2.1 6.2.2 Progra 6.3.1	Iction ew Relationship between RAPID execution and PickMaster project RAPID modules overview m module templates PmMain module	205 206 206 208 209 209
<u>6</u>	6.1 6.2	Introdu Overvia 6.2.1 6.2.2 Progra 6.3.1 6.3.2	Iction ew Relationship between RAPID execution and PickMaster project RAPID modules overview m module templates PmMain module PmUtility module	205 206 208 209 209 209 214
<u>6</u>	6.1 6.2	Introdu Overvie 6.2.1 6.2.2 Progra 6.3.1	Iction ew Relationship between RAPID execution and PickMaster project RAPID modules overview m module templates PmMain module PmUtility module PmProjMgr module	205 206 208 209 209 214 220
<u>6</u>	6.1 6.2	Introdu Overvie 6.2.1 6.2.2 Progra 6.3.1 6.3.2 6.3.3 6.3.4	Inction ew Relationship between RAPID execution and PickMaster project RAPID modules overview m module templates PmMain module PmUtility module PmProjMgr module PmProjServer module n modules	205 206 208 209 209 214 220 223 228
6	6.1 6.2 6.3	Introdu Overvia 6.2.1 6.2.2 Progra 6.3.1 6.3.2 6.3.3 6.3.4 System 6.4.1	Iction ew	205 206 208 209 209 214 220 223 228 228
<u>b</u>	6.1 6.2 6.3	Introdu Overvia 6.2.1 6.2.2 Progra 6.3.1 6.3.2 6.3.3 6.3.4 System 6.4.1 6.4.2	Inction	205 206 208 209 209 214 220 223 228 228 228 229
<u>b</u>	6.1 6.2 6.3	Introdu Overvia 6.2.1 6.2.2 Progra 6.3.1 6.3.2 6.3.3 6.3.4 System 6.4.1	Iction ew	205 206 208 209 209 214 220 223 228 228 228 229
<u>6</u>	6.1 6.2 6.3	Introdu Overvia 6.2.1 6.2.2 Progra 6.3.1 6.3.2 6.3.3 6.3.4 System 6.4.1 6.4.2	Inction	205 206 208 209 209 214 220 223 228 228 228 229
	6.1 6.2 6.3	Introdu Overvia 6.2.1 6.2.2 Progra 6.3.1 6.3.2 6.3.3 6.3.4 System 6.4.1 6.4.2 6.4.3 ime ope	Inction	205 206 208 209 214 220 223 228 228 228 228 229 230 235
	6.1 6.2 6.3 6.4	Introdu Overvie 6.2.1 6.2.2 Progra 6.3.1 6.3.2 6.3.3 6.3.4 System 6.4.1 6.4.2 6.4.3 ime ope Introdu FlexPe	Relationship between RAPID execution and PickMaster project	205 206 208 209 209 214 220 223 228 229 230 235 235 236
	6.1 6.2 6.3 6.4 Runti 7.1	Introdu Overvia 6.2.1 6.2.2 Progra 6.3.1 6.3.2 6.3.3 6.3.4 System 6.4.1 6.4.2 6.4.3 ime ope Introdu FlexPe 7.2.1	Relationship between RAPID execution and PickMaster project	205 206 208 209 209 214 220 223 228 229 230 235 235 236 236
	6.1 6.2 6.3 6.4 Runti 7.1	Introdu Overvie 6.2.1 6.2.2 Progra 6.3.1 6.3.2 6.3.3 6.3.4 System 6.4.1 6.4.2 6.4.3 ime ope Introdu FlexPe 7.2.1 7.2.2	Inction	205 206 208 209 209 214 220 223 228 228 229 230 235 235 235 236 236 238
	6.1 6.2 6.3 6.4 Runti 7.1	Introdu Overvie 6.2.1 6.2.2 Progra 6.3.1 6.3.2 6.3.3 6.3.4 System 6.4.1 6.4.2 6.4.3 ime ope Introdu FlexPe 7.2.1 7.2.2 7.2.3	Inction	205 206 208 209 209 214 220 223 228 228 229 230 235 235 235 236 236 238 238 239
	6.1 6.2 6.3 6.4 Runti 7.1	Introdu Overvie 6.2.1 6.2.2 Progra 6.3.1 6.3.2 6.3.3 6.3.4 System 6.4.1 6.4.2 6.4.3 ime ope Introdu FlexPe 7.2.1 7.2.2 7.2.3 7.2.4	Inction	205 206 208 209 209 214 220 223 228 229 230 235 235 236 236 238 238 239 246
	6.1 6.2 6.3 6.4 Runti 7.1	Introdu Overvie 6.2.1 6.2.2 Progra 6.3.1 6.3.2 6.3.3 6.3.4 System 6.4.1 6.4.2 6.4.3 ime ope Introdu FlexPe 7.2.1 7.2.2 7.2.3	Inction	205 206 208 209 209 214 220 223 228 229 230 235 235 236 236 238 238 239 246
	6.1 6.2 6.3 6.4 Runti 7.1	Introdu Overvia 6.2.1 6.2.2 Progra 6.3.1 6.3.2 6.3.3 6.3.4 System 6.4.1 6.4.2 6.4.3 ime ope Introdu FlexPe 7.2.1 7.2.2 7.2.3 7.2.4 7.2.5 7.2.6 I/O inte	Iction	205 206 208 209 214 220 223 228 229 230 235 235 236 238 238 238 239 246 252 255 259
	6.1 6.2 6.3 6.4 Runti 7.1 7.2	Introdu Overvia 6.2.1 6.2.2 Progra 6.3.1 6.3.2 6.3.3 6.3.4 System 6.4.1 6.4.2 6.4.3 ime ope Introdu FlexPe 7.2.1 7.2.2 7.2.3 7.2.4 7.2.5 7.2.6 I/O inte 7.3.1	Ictionew Relationship between RAPID execution and PickMaster project	205 206 208 209 214 220 223 228 229 230 235 236 236 238 239 246 252 255 259 259
	6.1 6.2 6.3 6.4 Runti 7.1 7.2	Introdu Overvia 6.2.1 6.2.2 Progra 6.3.1 6.3.2 6.3.3 6.3.4 System 6.4.1 6.4.2 6.4.3 ime ope 7.2.1 7.2.2 7.2.3 7.2.4 7.2.5 7.2.6 I/O inte 7.3.1 7.3.2	Ictionew Relationship between RAPID execution and PickMaster projectRAPID modules overview	205 206 208 209 214 220 223 228 229 230 235 236 235 236 238 239 246 252 255 259 259 260
	6.1 6.2 6.3 6.4 Runti 7.1 7.2	Introdu Overvia 6.2.1 6.2.2 Progra 6.3.1 6.3.2 6.3.3 6.3.4 System 6.4.1 6.4.2 6.4.3 ime ope Introdu FlexPe 7.2.1 7.2.2 7.2.3 7.2.4 7.2.5 7.2.6 I/O inte 7.3.1	Ictionew Relationship between RAPID execution and PickMaster project	205 206 208 209 214 220 223 228 229 230 235 236 236 238 239 246 252 255 259 259 259 260 262

		7.3.5	Timing diagrams for PLC communication	281
8	RAPI	D refere	ence information	287
<u> </u>				
	8.1		Iction to RAPID reference information	
	8.2	8.2.1	tions PmAckTarget - Acknowledge a target	
		8.2.1	PmAck larget - Acknowledge a larget PmCalcArmConf - Calculates the arm configuration	
		8.2.2	PmGetFlow - Get flow to execute	
		8.2.3	PmGetFlowInfo - Get information about a specific flow	293
		8.2.4 8.2.5	PmGetLastWa - Get last used work area	
		8.2.6	PmGetOperation - Get operation from a work area	
		8.2.7	PmGetProjectInfo - Get information about a specific project	
		8.2.7	PmGetWaByWobj - Get a reference to a work area using a work object data	
		8.2.9	PmGetWalnfo - Get information about a specific work area	
			PmSearchAdjust - Adjust number of remaining layers	
			PmSetLastWa - Set last used work area	
			PmSetRecoverAction - Set flow recover action	
			PmStartFlow - Starts a specific flow	
			PmStartProj - Start a PickMaster project	
			PmStopFlow - Stop a specific flow	
			PmStopProj - Stop a specific now PmStopProj - Stop current project	
	8.3		PmWaitProjStart - Wait for any active project	
	0.3	8.3.1	PmCalcIntermid - Calculate intermediate position	
		8.3.2	PmGetEvent - Get events for an action	
		o.s.z 8.3.3	PmGetPathHeight - Get a safe path height for an intermediate movement	323 205
		8.3.4	PmSetDefaultHeight - Set the default height	
		8.3.5		
		8.3.6	PmGetTarget - Get target	
		8.3.7	PmGetTgtAction - Get target action PmGetWaHeight - Get the height of a work area	
		8.3.8	PmGetWaName - Get the name of a work area	
	8.4		pes	
	0.4	8.4.1	pes pm_accdata - PickMaster acceleration/deceleration data	
		8.4.2	pm_acktype - PickMaster target acknowledge type	
		8.4.3	pm_actiondata - PickMaster action data	
		8.4.4	pm_actionhandle - PickMaster action handle	
		8.4.5	pm_actiontype - PickMaster action type	
		8.4.6	pm_eventdata - PickMaster event data	
		8.4.7	pm_eventtype - PickMaster event type	
		8.4.8	pm flowinfo - PickMaster flow information	
		8.4.9	pm_noduleinfo - PickMaster module information	
			pm_movetype - PickMaster movement type	
			pm_noverype - Pickmaster movement type	
		0.4.11 Q / 12	pm_onsetuata - Onset data pm_operationdata - PickMaster operation data	352
			pm_ophandle - PickMaster operation handle	
			pm_oprojectinfo - PickMaster project information	
			pm_projectimo - PickMaster project mormation	
			pm_redetinie - PickMaster redet mornation	
			pm_searchtype - PickMaster stack search type	
		Q / 1Q	pm_singareatype - PickMaster interpolation type around singular points	364
		Q / 10	pm_singarearype - PickMaster interpolation type around singular points	265
			pm_stoptype - PickMaster stop type	
			pm_stoptype - PickMaster stop type pm_targetdata - PickMaster target data	
		0.4.21 Q / 00	pm_targetbandle - PickMaster target bandle	271
		0.4.23 Q / 0/	pm_wadescr - PickMaster work area reference pm_wainfo - PickMaster Work Area information	312
		0.4.24	PII_wainio - Fichiviasiei work Area intorniation	5/3
9	Relat	ionship	s between PickMaster frames	377
	9.1	Structu	re of this chapter	377

9.2	Shape frame	378
9.3	Format frame versus shape frame	379
9.4	Pallet pattern frame versus shape frame	384
9.5	Format frame versus tool frame	385
9.6	Pallet pattern item versus tool frame	387
9.7	Format frame versus work object in format operation set	388
9.8	Pallet pattern frame versus work object in pallet pattern operation set	390
9.9	Tune frame versus work area frame	392
9.10	Tune frame versus work area frame versus displacement frame	395
Index		397

Overview of the manual

About this manual		
	This manual contains information and instructions for insta running Palletizing PowerPac.	lling, configuring, and
Usage		
	This manual should be used during installation and configu PowerPac. It describes Palletizing PowerPac and includes ste on how to perform the tasks from there.	•
Who should read th	is 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 wor	k
	 Good skills in the IRC5 robot controller and RAPID pr 	ogramming
		<u> </u>
References		
	References	Document ID
	Operating manual - IRC5 with FlexPendant	3HAC050941-001
	Operating manual - RobotStudio	3HAC032104-001
	Operating manual - Troubleshooting IRC5	3HAC020738-001
	Technical reference manual - RAPID Instructions, Functions and Data types	3HAC050917-001
	Technical reference manual - RAPID Overview	3HAC050947-001
	Technical reference manual - RAPID kernel	3HAC050946-001
	Technical reference manual - System parameters	3HAC050948-001
	Product manual - IRC5	3HAC021313-001
	Product manual - IRC5 Panel Mounted Controller	3HAC027707-001
	Product manual - IRB 260	3HAC026048-001
	Product manual - IRB 460	3HAC039842-001
	Product manual - IRB 660	3HAC025755-001
	Product manual - IRB 760	3HAC039838-001
	Product manual - IRB 700 Product specification - Palletizing PowerPac	3HAC041431-001
	riouuci specification - railetizing rowerrac	JI 1AC04 143 1-00 1

9

Overview of the manual

Continued

Other references

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

Revisions

Revision	Description		
-	First edition		
A	 Released with RobotStudio 5.14.03 The following updates are made: Updated the section <i>Pack data on page 111</i>. Updated the section <i>UnPack data on page 114</i>. Updated the section <i>Import PM5 on page 117</i>. 		
B Released with RobotStudio 5.15 The following updates are made: • Updated the section Import PM5 on page 117. • Added pmSystem_doMotorOnState and updated pmSystem_doMotorOn, see System outputs on page 271 • Updated new Operator interface on page 94 dialog box • Updated Flow recovery on page 246 • Updated timing diagrams in Timing diagrams for PLC constition on page 281. • Updated Approach/Depart, see Group operation set on page 281.			
С	 Released with RobotStudio 5.15.01 The following updates are made: Updated the section <i>PmMain module on page 209</i>. Updated the section <i>PmUtility module on page 214</i>. Updated the section <i>Public system module pmrcSys on page 230</i> Added the section <i>Delete projects on page 116</i>. Updated the program code in the section <i>PmProjMgr module on page 220</i>. 		
D	 Released with RobotStudio 5.60 The following updates are made: Added the section <i>IOPanel on page 142</i>. Added the section <i>Product flow sequence on page 21</i>. Updated the section <i>Extended I/O interface on page 271</i>. 		
E	 Released with RobotStudio 5.61 The following updates are made: Added the section Versions of Palletizing PowerPac on page 32 Updated the section General tab on page 121. 		
F	 Released with RobotStudio 6.0 The following updates are made: Updated the section Add tool on page 55. Updated the section Pallet pattern on page 72. Updated the section Download on page 102. 		
G	 Released with RobotStudio 6.01 The following updates are made: Updated the section <i>Operation set on page 179</i>. 		

Continued

Revision	Description
н	Released with RobotStudio 6.02
	The following updates are made:
	• Updated the section Using a robot with 6 axes on page 25.
	Updated the section <i>Tune Limitation on page 156</i> .
	Updated the section <i>Group operation set on page 180</i> .
	Updated the section <i>Operation editor on page 196</i> .
J	Released with RobotStudio 6.03
	The following updates are made:
	Updated the section <i>Feeder on page 170</i> .
	Updated the section <i>Operation set on page 179</i> .
К	Released with RobotStudio 6.04
	The following updates are made:
	 Updated the section PmCalcIntermid - Calculate intermediate position on page 319.
	 Updated the section <i>Preview Palletizing on page 90</i>.
	• Opdated the section <i>Fleview Failed2ing on page 90</i> .
L	Released with RobotStudio 6.05.
	Updated the section <i>Pallet pattern on page 72</i> .
М	Released with RobotStudio 6.06.
	Updated the section Layout Editor on page 75
	Updated the section <i>Product/Pallet/Sheet on page 59</i>
	Updated the section Operation editor on page 196
N	Released with RobotStudio 6.08.
	• Updated the section <i>Pattern/Stack operation set on page 183</i> .
Р	Released with RobotStudio 2019.1.
-	Updated the references.
0	Released with BobotStudio 2020.1.
Q	Added note in <i>Pick/Place</i> section.
	Updated <i>Operation Set</i> section.
D	· · ·
R	 Released with RobotStudio 2020.2. Updated the section <i>Scheduled Jobs on page 96</i>.
	 Updated the section Scheduled bobs on page 93

This page is intentionally left blank

1 Safety

Safety of personnel

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

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

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

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

Safety regulations

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

This page is intentionally left blank

2.1 Overview

2 Introduction

2.1 Overview

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 list of terms relating to PickMaster and terms which are specific to palletizing.

2.2 About Palletizing PowerPac

2.2 About Palletizing PowerPac

Overview

The Palletizing PowerPac (PzPP) is a RobotStudio solution for offline programming and simulation of Palletizing application. The PowerPac replaces the ABB's PickMaster 5 application.

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

The Palletizing PowerPac can be installed on a computer running Windows XP, Windows Vista, or Windows 7. RobotStudio should also be installed.

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

For minimum system requirements, see *Product specification - Palletizing PowerPac*. The Prepared for PickMaster option, together with the Palletizing PowerPac 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 120*.
- The Tool I/O Configuration for configuring the I/O connection between robot tool and controller. See *Edit Tool Signals on page 168*.
- The Build Feeder Configuration for configuring the feeder model properties, including hotspots. See *Create Feeder on page 139*.
- The Feeder Configuration for configuring the feeder properties, including signals, work objects, feeder CAD models, and so on. See *Feeder on page 170*.
- 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 59*.
- The Pallet Pattern Configuration for configuring the stack of products organized in different layouts. See *Pallet pattern on page 72*.
- The Layout Editor for creating new or modifying existing layouts. See *Layout Editor on page 75*.
- 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 79*.

2.2 About Palletizing PowerPac Continued

- The Operation Set Configuration for configuring the format operation set and pattern operation set. See *Operation set on page 179*.
- The Job Wizard Configuration for configuring a new job including pattern operation sets and/or format operation sets. See *Add job on page 83*.
- The Flow configuration for defining accessible feeders in runtime. See *Flow* on page 202.
- The Tuning for changing parameter values online from the FlexPendant. See *Tuning on page 255*.
- The Project Settings for defining project name, descriptions, restart options and tune limitations. See *Project Settings on page 154*.
- The Feeder Order Settings for defining feeder order in the cell layout. See the Feeder Order Settings page in *Controller Settings on page 158*.
- The Event Settings for defining signals that can report events from external devices. SeeThe Event Settings page in *Controller Settings on page 158*.
- The Message Settings for defining messages that can be reported from external devices. See The Message Settings page in *Controller Settings on page 158*.
- The Robot Settings for modifying motion limits and RAPID programs for each robot. See *Robot Settings on page 165*.
- 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 107.
- The Pack Data for storing the Project in controller. See Pack data on page 111.
- The UnPack Data for retrieving the stored project. See UnPack data on page 114.
- The Palletizing Library for saving products, layouts, I/O connections and messages. See *Library on page 119*.
- The Preview Palletizing function to go through each palletizing robot targets for user validation. See *Preview Palletizing on page 90*.
- The Simulation function to simulate the palletizing application. See *Simulation* on page 93.
- The Download and Upload function to synchronize programs between RobotStudio and robot controllers. See *Download on page 102* and *Upload on page 104*.

2 Introduction

2.3 The palletizing process

2.3 The palletizing process

Overview

The palletizing application aims at picking larger size objects from a fixed position and stacking them tightly together in another fixed position. An important parameter in 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. The lesser the space products require, lesser will be the transportation cost.

The palletizing cell

The following figure illustrates an example of a palletizing cell.



xx110000001

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 191*.

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.

All the goods produced in a factory pass through the palletizing cells before shipping to the 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.

2.3 The palletizing process Continued

Packing 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.

Speeding up the process

Introduction

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

To plan each layout

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

Infeeders, outfeeders, and logical devices

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

The robot can move between different stacks

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

The flow concept

Introduction

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.

2 Introduction

2.3 The palletizing process *Continued*

Execution of a palletizing job

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

- 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 Palletizing PowerPac 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.

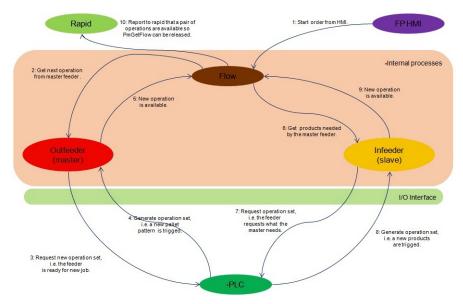
Running parallel jobs does not add any complexity to programming or operator interaction.

2.3 The palletizing process Continued

Product flow sequence

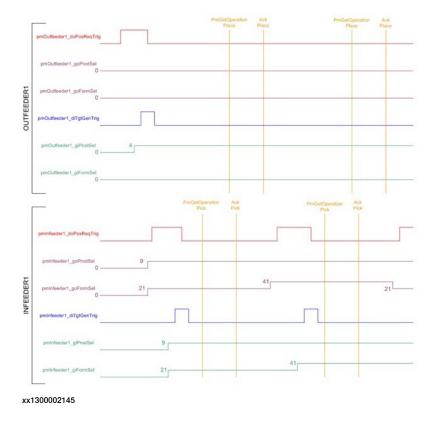
Following are the product flow sequence chart and the signal chart which are helpful while simulation.

Product flow sequence chart, 1 in -1 out



xx1300002144

Signal chart



Related information

Terms and concepts on page 22

2.4 Terms and concepts

2.4 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 18*

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

Term list

Term	Definition
Action	Description of one robot movement to get to/on/from a <i>target</i> . Every action can have a number of <i>events</i> . For more information, see <i>Relationship between RAPID execu-</i> <i>tion and PickMaster project on page 206</i> .
Activator	An I/O controlled part of a robot tool, normally a vacuum cup. For more information, see <i>Create Tool on page 120</i>
Event	A description of an event on the robot path, for example setting an I/O signal. For more information, see <i>Relationship between RAPID execu-</i> <i>tion and PickMaster project on page 206</i> .
Facing	Possibility to select one or more sides of an <i>item</i> that should be facing outwards in a <i>pallet pattern</i> .
Feeder	A PickMaster's representation of pick and place areas. Often referred to as station, infeeder, or outfeeder.
	It defines the signals needed to control the in- and outfeeding of groups and pallet patterns on the feeder. Palletizing jobs can only be started on master feeders, see <i>Flow on page 202</i> . 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 pat- tern operation sets. It also contains general robot movement data for the feeder, for example, safe positions.
	For more information, see the <i>Feeder on page 170</i> Configuration.
Flow	Logical directions of <i>items</i> being moved between pick and place stations when performing a palletizing/depalletizing job. A flow:
	 Contains one master station on which palletizing/depalletizing jobs can be started. The master station dictates the operation sequence for itself and its slave stations.
	 Contains one or more slave stations. The slave stations execute operations requested by the master station.
	For more information, see <i>Flow on page 202</i> .
Group	Describes a group of products that can be picked/placed by a robot tool in one operation
Group operation set	One operation that describes how to pick/place a group on a specific feeder. For more information, see the <i>Operation set</i> on page 179 Configuration .

2.4 Terms and concepts *Continued*

Term	Definition
Hotspot	A special frame on feeder model that can be used to attach a work object. It also describes the position relationship between an item and the work object.
Item	The generic term for a specific object to be picked or placed. An item can be a product (i.e. box or bag), pallet, or slip sheet. An item is based on the geometric definition of a shape. For more information, see the <i>Product/Pallet/Sheet on page 59</i> Configuration . Box: A rigid box with rectangle shape
	Bag: Same as box, but the bag has margins on length and width sides, specifying how much distance the bags can be overlapped with each other in a layout. Sheet: A thin sheet that is placed on a pallet before palletizing of products and between two layers to increase stability in the stack.
	Pallet: The actual wooden or plastic structure that the products are placed on.
Job	A job is a supported program (i.e. Operation Set) on a flow's master feeder.
Layout	Defines the arranged two dimensional pattern of the <i>shapes</i> in a layer.
Master Feeder	The master station of a flow
Operation	An operation describes what a robot shall do when entering a <i>work area</i> for a new pick or place. What to do is described as a list of <i>targets</i> .
	For more information, see <i>Relationship between RAPID execu-</i> <i>tion and PickMaster project on page 206.</i>
Pallet pattern	Defines a stack of <i>shapes</i> organized in different <i>layouts</i> . A layer in the stack can either be of <i>pallet</i> , <i>slip sheet</i> , or <i>product</i> type.
Pallet pattern operation set	A sequence of <i>operations</i> that describe how to pick/place a <i>pallet pattern</i> on a specific feeder.
Pick Setting, i.e. Format	Defines one item or a group of items that can be picked/placed by a robot tool in one operation.
	In PickMaster 5, it was called Format. For more information, see the <i>Pick setting on page 79</i> Config-
	uration.
Project	Description of a PickMaster palletizing process applied on a specific <i>line</i> . Several projects can be defined for each <i>line</i> .
	For more information, see <i>Project overview on page 106</i> .
Robot Target	A single robot target that robot goes to during picking and placing. An action contains one robot target.
Safe Target	A position that the robot always has to pass through when go- ing to or leaving a feeder
Slave Feeder	A slave station of a flow
Stack	An arranged pile of items consisting of a number of layers.
Target	A target describes a robot position used when performing an operation on a feeder. A target has a list of products carried by the robot tool and a list of actions that shall be ex- ecuted.Note the difference with robot target: a target can have a list of actions, and each action has one robot target.

Continues on next page

23

2 Introduction

2.4 Terms and concepts *Continued*

Term	Definition
Tool Function	A single function of a tool that requires input signals to control and will generate results as output signals.
Tool Scenario	One grouping of tool functions to use in all actions of a target during picking or placing.
Work area	The original term used in PickMaster 5 PC to describe feeders, now it is only used in RAPID functions and variables.
Zone	A vacuum tool function is divided into a number of zones. Each zone has one or more activators.

2.5 Using a robot with 6 axes

Overview

Even though running Palletizing PowerPac 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 Palletizing PowerPac 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 26.
- The wrist is tilted. See Wrist is tilted on page 27.
- The robot is bending backwards. See *Robot is bending backwards on page 28*.

For using a parallel rod robot with 6 axes together with Palletizing PowerPac, see *Parallel rod robot on page 29*.

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 180* and *Pattern/Stack operation set on page 183*.

2 Introduction

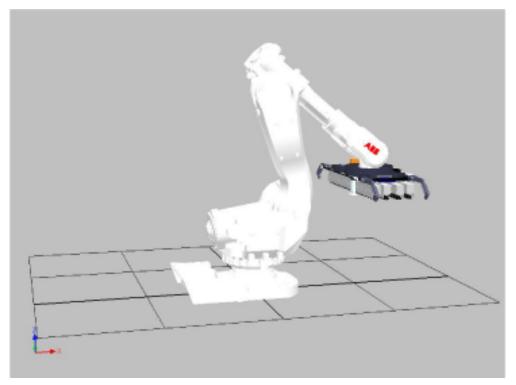
2.5 Using a robot with 6 axes *Continued*

Bending backwards robot

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

Axis 6 points down

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



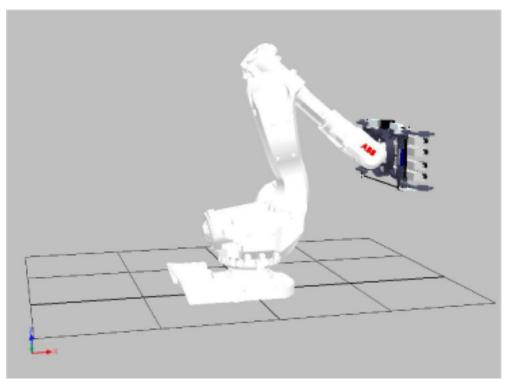
xx0700000354

Configuration	Description
Using SingArea \Wrist	Long linear movements causes varying tool orientation.
	Tool orientation deviates between targets when using SingArea / Wrist interpolation. The deviation is small for short movements but increases with longer distance between robot targets.
Using SingArea \LockAx4	While using a 6 axis robot, using this instruction locks the fourth axis during the motion between the targets.
Using SingArea \Off	See Manual modifications on page 28.

2.5 Using a robot with 6 axes *Continued*

Wrist is tilted

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



xx0700000355

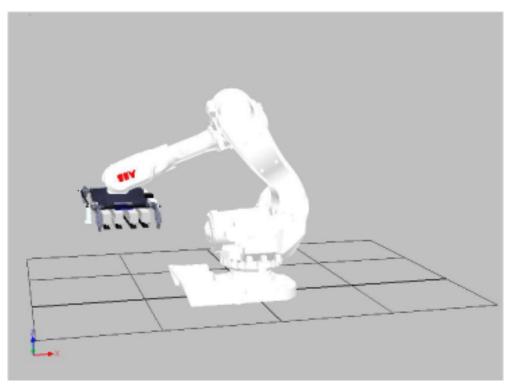
Configuration	Description	
Using SingArea \Wrist	Not recommended.	
Using SingArea \LockAx4	Not recommended.	
Using SingArea \Off	See Manual modifications on page 28.	

2 Introduction

2.5 Using a robot with 6 axes *Continued*

Robot is bending backwards

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



xx070000356

Configuration	Description	
Using SingArea \Wrist	Not recommended.	
Using SingArea \LockAx4	Not recommended.	
Using SingArea \Off	ea \Off See Manual modifications on page 28.	

Manual modifications

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

	Action	Note
1.	Use ConfL \Off. Update the system module <i>pmrcUser</i> , with the procedure PmDoAction from system module <i>pmrcSys</i> and rename the procedure to DoAction. Modify DoAction according to description in section <i>System module</i> <i>pmrcUser on page 29</i> .	Linear motion supervision must be turned off to allow movements with large axis reorientations (> 90°).
2.	Use a robot position for each work area with MoveJ. In PzPP browser, record a safe position for each feeder, and use MoveJ for each position. See Safe Targets Set WorkObject position on page 177.	First ensure the angles of axis 4 and 6 do <i>not</i> reach the physical limitation. Then ensure the robot has the correct arm configuration when operating on the work area.

2.5 Using a robot with 6 axes Continued

	Action	Note
3.	Use ConfJ \On. Turn on the configuration control for joint movements. Update the system module <i>pmrcUser</i> , with the procedure PmDoAction from system module <i>pmrcSys</i> , and rename the procedure to DoAction. Modify DoAction according to description in section <i>System module</i> <i>pmrcUser on page 29</i> .	First ensure the angles of axis 4 and 6 do <i>not</i> reach the physical limitation. Then ensure the robot has the correct arm configuration when operating on the work area.

System module pmrcUser

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

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

to DoAction instead of PmDoAction.

You find the complete program code for the system modules *pmrcUser* and *pmrcSys* in section *Program code on page 231*.

The fo	bllowing modifications must be done i	n the program code:
• Co	onfL\Off	
• Co	onfJ\On	
	Note	

Parallel rod robot

RAPID calls to PMCalcArmConf

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 26* and *Wrist is tilted on page 27* are possible.

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

All RAPID calls to the routine $\tt PmCalcArmConf$ must use the optional argument $\TypeB1$ instead of <code>\cf6</code>.

2 Introduction

2.5 Using a robot with 6 axes *Continued*

pmMain.mod

Modify the program module pmMain.mod, procedure Operate:

```
PmCalcArmConf
```

. . .

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

Related information

Group operation set on page 180. Pattern/Stack operation set on page 183. Public system module pmrcUser on page 229.

3 Installation

3.1 System requirements

Hardware and software requirements

The following are the prerequisites for installing Palletizing PowrPac:

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

Hardware requirements	Software requirements
CPU: 2.0 GHz or faster processor, recommen- ded is multicore processor	Microsoft Windows 7 (32 Bit or 64 bit) Microsoft Windows 10 (64 bit)
Memory: 1 GB RAM or more (More is recommended).	RobotStudio 5.60 or above
Available disk space: 5+ GB on the system disk, 250+ MB on the installation disk	RobotWare 5.60 or above
Graphics card: High performance OpenGL- compatible graphics card with the correspond- ing up-to-date drivers installed	.NET 4.7.2 Framework
Screen resolution: 1280 x 1024 pixels (Recom- mended)	
Colors: 256 or higher	
DPI: Normal size (96 dpi)	
Mouse: Three-button mouse	

Installing Palletizing PowerPac

Use this procedure to install the Palletizing PowerPac:

1 Browse to Palletizing PowerPac 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.



Ensure that you have RobotStudio installed on your computer before installing Palletizing PowerPac.

Installing a license

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

3 Installation

3.1 System requirements *Continued*

Versions of Palletizing PowerPac

The Palletizing PowerPac is available in the following two versions:

- Basic: The Basic version has limited features and is available free of cost.
- **Premium**: The Premium version has advanced features and is available on paying a subscription fee.



The Basic version requires a Station or Pack&Go file.

Following are the details of the features available in the two versions of Palletizing PowerPac.

Feature	Basic	Premium
Build Cell - Add Tool	Yes	Yes
Build Cell - Add Feeder	No	Yes
Product Data - Product/Pallet/Sheet - Pallet Patterns	Yes	Yes
Programming - Pick Setting - Add Job	Yes	Yes
Validation - Check Reachability - Preview Palletizing	Yes	Yes
Simulation - Start - Pause/Stop - Reset	No	Yes
Transfer - Add Controller - Download - Upload	Yes	Yes
Project Properties - Overview - IO Interface - Unpack data - Report	Yes	Yes
Project Properties - Pack Data - Delete Projects - Import PM5	No	Yes
Advanced - Library - Models	Νο	Yes

Continues on next page

3.1 System requirements *Continued*

Feature	Basic	Premium
Advanced - IOPanel	Yes	Yes
3D Tools and Help	Yes	Yes

The support for the different versions of Palletizing PowerPac depends on the RobotStudio version installed on your system as displayed in the following table:

RobotStudio version	Palletizing PowerPac Basic	Palletizing PowerPac Premium
RobotStudio Basic	Limited features	Not available
RobotStudio Premium	Limited features	Full features



Graphics is disabled in the Basic version of RobotStudio and Palletizing PowerPac.

This page is intentionally left blank

4.1 Introduction

4 Workflow

4.1 Introduction

The following is the recommended working procedure for Palletizing PowerPac with PickMaster.

4.2 Preparing your controller for PickMaster

4.2 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 Palletizing PowerPac, is needed to run PickMaster on an IRC5 robot controller.

Preparing your controller for PickMaster

Use this procedure to prepare the controller for PickMaster:

	Action	Note/See
1	Create and install a system for the robot controller using Ro- botStudio.	Operating manual - RobotStudio.
2	Setup RAPID tool data for the robot tool. Use RobotStudio or the FlexPendant.	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 pick- ing/placing and another one to be used when calibrat- ing work objects. If the robot tool supports stack search a separate tool data is created for that pur- pose. Technical reference manual - RAPID Instructions, Functions and Data types, section tooldata - Tool data.
3	Calibrate the work object for each feeder. Use the FlexPend- ant.	Technical reference manual - RAPID Instructions, Functions and Data types, section wobjdata - Work object data.Operating manual - IRC5 with FlexPend- ant.
4	Add tool related signals. Use RobotStudio or the FlexPend- ant.	See <i>Edit Tool Signals on page 168</i> for information about the signals.Operating manual - RobotStudio.Operating manual - IRC5 with FlexPendant.
5	Add feeder related signals. Use RobotStudio or the Flex- Pendant.	See <i>Edit detailed signals for feeder on page 173</i> for information about the signals.Operating manual - RobotStudio.Operating manual - IRC5 with FlexPendant.
6	Add event related signals if you want an external equip- ment to report errors and/or messages to the PickMaster application. Use RobotStudio or the FlexPendant.	The Controller Settings window .Operating manual - RobotStudio.Operating manual - IRC5 with FlexPend- ant.

4.3 How to add tool

4.3 How to add tool

On the **Palletizing** tab, click **Add Tool** to import a tool for the palletizing process. See *Add tool on page 55* for detailed description. 4.4 How to add feeders

4.4 How to add feeders

On the **Palletizing** 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 57* for detailed description.

4.5 How to create or modify product data

4.5 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 59* 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 72 for detailed description.

4.6 How to set pick setting

4.6 How to set pick setting

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

See *Pick setting on page 79* for detailed description.



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

4.7 How to add job

4.7 How to add job

On the **Palletizing** tab, click **Add a Job** to create a job for palletizing process. See *Add job on page 83* for detailed description. 4.8 How to do simulation

4.8 How to do simulation

On the **Palletizing** 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 *Simulation on page 93* for detailed description.

4.9 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.

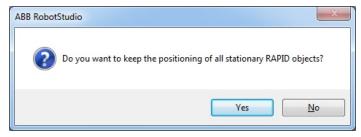
After you have changed robot position, a message box will show up to ask you whether to update the Task Frame:

ABB RobotStudio
Do you also want to move the Task Frame?
Yes <u>N</u> o

xx110000180

Choose Yes to update Task Frame, otherwise, the controller needs to be restarted.

Then a further message box will ask you to keep the positioning of all stationary RAPID variables (for example, work object):



xx110000181

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

4.10 How to transfer the project

4.10 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 101 and Download on page 102 for detailed description.

4.11 How to start production

4.11 How to start production

Introduction

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

Starting production

Use this procedure to start a PickMaster project.

	Action	See
1	Start the PickMaster FlexPendant interface.	
2	Select the project you want to run.	Opening a project on page 238.
3	Start the project.	Starting a project on page 239.
4	Start flows.	Starting a specific flow on page 240 and Starting and stopping all flows on page 242.

Related information

FlexPendant interface on page 236. Opening a project on page 238. Starting and stopping production on page 239. 4.12 How to Pack Palletizing Project Data to Controller

4.12 How to Pack Palletizing Project Data to Controller

On the **Palletizing** tab, in the **Project** group, click **Add Controller** to connect to online controllers. Click on **Pack Data** in **Project** group to transfer the palletizing project configured in RobotStudio Station to online or offline controllers.

For more information, see Add controller on page 101 and Pack data on page 111.

4.13 How to UnPack Palletizing Project Data from Controller

On the **Palletizing** tab, in the **Project** group, click **Add Controller** to connect to online controllers. Click on **UnPack Data** in **Project** group to transfer the palletizing project configured in RobotStudio Station from online or offline controller.

For more information, see Add controller on page 101 and UnPack data on page 114.

4.14 How to Import PickMaster5 Projects into Palletizing PowerPac

4.14 How to Import PickMaster5 Projects into Palletizing PowerPac

Palletizing tab, in the **Project** group, click on **Import PM5** to convert PickMaster5 project to palletizing project.

For more information, see Import PM5 on page 117.

5.1 Introduction

5 User interface

5.1 Introduction

This chapter describes how to navigate in Palletizing PowerPac. 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 Palletizing PowerPac.

5.2 General

5.2 General

Overview			
	Before starting Palletizing PowerPac, 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 Palletizing PowerPac sub-option.		
Loading a Station			
	You can go through following steps to create a station and a system, and start using Palletizing PowerPac:		
	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 – Palletizing PowerPac. 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.		

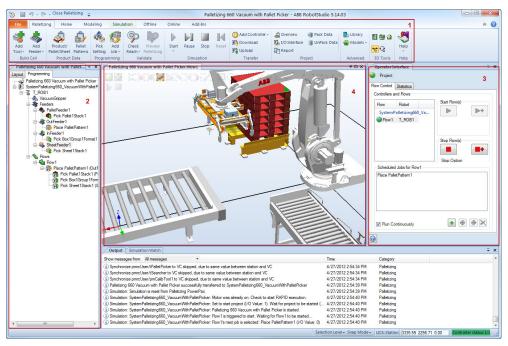


For other ways to create a station and a system, refer to Operating manual - RobotStudio.

5.2 General Continued

The User Interface

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



	Item	Description
1	Palletizing ribbon tab	Contains the general functions for palletizing application. When creating a new project, the work flow is usually from left to right.See <i>Ribbon on page 52</i> for detailed description.
2	Palletizing browser	Organizes the components of the station and project in a tree structure.See <i>Layout browser on page 146</i> for detailed description.
3	Tool window	Dialog boxes to edit certain project elements in browser and ribbon.
4	Graphics window	The graphics window is coordinated with the selection in browser elements and edited object in the tool window, showing context related temporary graphic objects.

5.3.1 Overview

5.3 Ribbon

5.3.1 Overview

The Palletizing PowerPac ribbon contains the following controls

- Building cell
- Creating pick and place programs
- Operating virtual controller
- Modifying Palletizing PowerPac data
- · Modeling tools and feeders
- Help information



xx110000003

Elements on the Palletizing ribbon-tab

Group	Button	Description
Build Cell	Add Tool	Add a tool to robot for palletizing/depalletiz- ing. See <i>Add tool on page 55</i> for detailed description.
	Add Feeder	Add feeders to the cell for palletizing/depal- letizing. At least two feeders are needed for a robot. See <i>Add feeder on page 57</i> for de- tailed description.
Product Data	Product/Pallet/Sheet	Create products, pallets or sheets to be palletized/depalletized. See <i>Product/Pallet/Sheet on page 59</i> for detailed description.
	Pallet Patterns	Create pattern related to the products to be palletized/depalletized. See <i>Pallet pattern on page 72</i> for detailed description.
Programming	Pick Setting	Define how robot tool picks the products, pallets or sheets. See <i>Pick setting on</i> <i>page 79</i> for detailed description.
	Add Job	Open Job Wizard to build a new palletiz- ing/de-palltizing job. See <i>Add job on page 83</i> for detailed description.
Validate	Check Reach	Check whether all of the picking robot tar- gets and placing robot targets are reachable.
	Settings on Check Reachability	Check whether the specified picking robot targets and placing robot targets are reachable. See <i>Check reach on page 88</i> for detailed description.
	Preview Palletizing	Preview all pick and place targets for all operation sets on feeders, by jumping tool or robot to the robot targets accordingly. See <i>Preview Palletizing on page 90</i> for de- tailed description.

Continues on next page

5.3.1 Overview Continued

Group	Button	Description
Simulation	Quick Start	Transfer program to virtual controller, open operator interface and start the simulation with the default jobs in one click. See <i>Oper-</i> <i>ator interface on page 94</i> for detailed descrip- tion.
	Operator Interface	Transfer program to virtual controller, open operator interface. Then start the simulation for the specified jobs manually. See <i>Operat-</i> <i>or interface on page 94</i> for detailed descrip- tion.
	Record Simulation as Viewer	Select this option to save the simulation as viewer file after the simulation is stopped. See <i>Record as viewer File on page 99</i> for detailed description.
	Speed Mode	Set simulation speed and there are three modes for it, such as Full, Customized and Low. See <i>Speed mode on page 100</i> for detailed description.
Transfer	Add Controller	Connect a controller with this station using RobotStudio Online functionality.
	Download	Download the program to connected control- lers. See <i>Download on page 102</i> to Control- lers for detailed description.
	Upload	Upload the program from the connected controller. See <i>Upload on page 104</i> from Controllers for detailed description.
Project	Overview	Displays the overview information for all operation sets in the project. See <i>Project</i> overview on page 106 for detailed description.
	I/O Interface	Displays the I/O information for this project. See I/O interface on page 107 for detailed description.
	Report	Generate a report for this project. See <i>Project report on page 110</i> for detailed description.
	Pack Data	Stores the Palletizing PowerPac project in the controller (Virutal or Real controller). See <i>Pack data on page 111</i> for detailed description.
	UnPack Data	Unpacks the stored Palletizing PowerPac project to the controller (Virutal or Real controller). See <i>UnPack data on page 114</i> for detailed description.
	Delete Projects	Deletes the saved projects from the Virtual or Online Controller. See <i>Delete projects on</i> <i>page 116</i> for more details.
Advanced	Library	Edit the default libraries for reuse. See <i>Library on page 119</i> for detailed description.
	Models	Guide user to design customized tool and feeder. See <i>Create Tool on page 120</i> and <i>Create Feeder on page 139</i> for detailed description.

Continues on next page

5.3.1 Overview *Continued*

Group	Button	Description
	IOPanel	Monitors and sets the signals used during palletizing process. See <i>IOPanel on page 142</i> .
3D Tools	View angles	Adjust the view point for the station from top, left, and front.
	Freehand move	Enable to adjust the position/orientation for the selected objects by manual dragging in 3D view.
Help	Help	Provides the user manual of Palletizing PowerPac.
	About	Provides the general information about Pal- letizing PowerPac.

5.3.2 Add tool

5.3.2 Add tool

1 On the **Palletizing** ribbon Tab, in the **Build Cell** group, click **Add Tool**.



2 Under Add Tool select the tool from ABB Grippers, User Grippers, or Browse Library to select a gripper. The Add Tool dialog appears.



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

Add Tool: VacuumGripper_1
Select Robot
Controller SysPPPExt
Robot T_ROB1
Signal Configuration
The signals are valid Gibbo Edit Signals
OK Add to Station Only Cancel

Item	Description
Controller	Select a controller in which the robot to attach the tool be- longs to.
Robot	Select a robot which will hold the tool.
Signal Configuration	Provide the match information about signals connection between the tool and the virtual controller. If the signal connection is not valid, the status light will be red, and you can check and modify with button Edit Signals . See <i>Edit</i> <i>detailed signals for feeder on page 173</i> for detailed descrip- tion.

- 3 Do one of the following:
 - Click **OK** 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.

5.3.2 Add tool *Continued*

- 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.
- 4 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.



When a custom tool is added the signals are mapped automatically to the free available signals based on their type.

5.3.3 Add feeder

5.3.3 Add feeder

1 On the **Palletizing** ribbon tab, in the **Build Cell** group, click **Add Feeder**.



2 Under Add Feeder select the Feeder from ABB Feeders, User Feeders, or Browse Library to select a feeder.



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

3 After a feeder library is selected, the following dialog box pops up.

🎻 Add Feeder	×
Select Robot Controller SysPPPExt	
Robot T_ROB1	•
Signal Configuration The signals are valid	Edit Signals
Feeder Type 🍣 💿 InFeeder	💐 🔘 OutFeeder
🍫 🔘 PalletFeeder	🍇 🔘 SheetFeeder
OK Add to	Station Only Cancel

Item	Description
Controller	Select a controller in which the robot to work with the feeder belongs to.
Robot	Select a robot in which the feeder is added.
Signal Configuration	Provide the match information about signals connection between the feeder and the virtual controller. If the signal connection is not valid, the status light is red, and you can check and modify with button Edit Signals . See <i>Edit detailed</i> <i>signals for feeder on page 173</i> for detailed description.

5.3.3 Add feeder *Continued*

- 4 Do one of the following:
 - Click **OK** to add the selected feeder to the robot. The signals are connected between the feeder and the virtual controller automatically.
 - 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.
- 5 After the feeder is added, a new feeder node is added under the **Feeders** node of the robot in the programming browser, and also a new feed node is added in the layout browser.

5.3.4 Product/Pallet/Sheet

5.3.4 Product/Pallet/Sheet

1 On the **Palletizing** ribbon tab, in the **Product Data** group, click Product/Pallet/Sheet.



2 In the **Product/Pallet/Sheet** dialog box, click on any of the four item types: Box, Bag, Pallet, or Sheet.

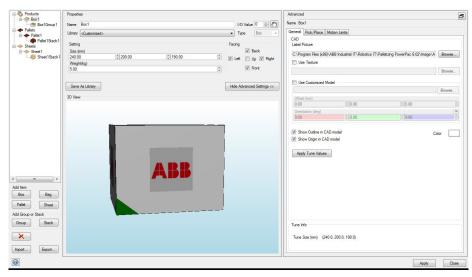


In a project, at least one box or one bag should be created.



Note

After an object is selected from the item list, its properties will be shown on the right of the UI.



xx110000009

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.

Palletizing PowerPac supports the handling of multiple columns of items in the group. This allows the user to create a group with multiple columns in the Products dialog. The creation of a multi column item group also means that the Layouts in the Operation sets have to be handled according to the number and arrangement of items in the group. This has been handled for a few groups.

5.3.4 Product/Pallet/Sheet Continued

> To use multiple columns, mention the number of columns to be picked by the gripper in the Product/Pallet/Sheet dialog box. Also, ensure that the pattern layout has a count that is divisible by the number of items in the format.



Note

Multi column support is handled only for the start corner X- Y-, the user is free to use other combinations but the layouts may require manual corrections.

Currently the Operation set generation is handled only for "Back" orientation for the BoxGroup (Item group).

It is recommended to manually check for collisions in the Operation set while the boxes are being placed as it is currently not possible to check for collisions.

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

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

5.3.4 Product/Pallet/Sheet Continued

Box Properties

This section describes how to edit the box properties.

Properties			
Name Box1			I/O Value 0 🚔 🖑 🕅
Library <pre>Custom></pre>			▼ Type Box ▼
Setting Size (mm) 240.00	▲ ▼ 200.00	▲ 300.00	Facing Back Left Up V Right
Weight (kg) 5.00			Front
3D View		\sim	
		Ann	

xx110000010

То	Do this
Rename a Box	In the Name text box, type a new name for the box. In default, a unique name can be generated automatically when a new box is created.
Set I/O value	In the I/O value number box, type a value to identify different products. Or click the button besides it, a new I/O value can be found automatically.
Select Box from Library	In the Library drop-down combo box, select an existing box to read in its values.
View the Item Type	In the Type combo box, the item type is shown.Its type cannot be changed after the item is created.
Set Box Size	In the Size boxes, type the dimensions for x, y and z.
Set Box Weight	In the Weight number box, type the weight (in kg).
Set Box Facing	Select the check boxes in Facing group to define which sides of the box to attach a label. See below 'Note' for detailed description.
Save the Box	Click <i>Save As Library</i> button, current box will be saved into the library for reusing.
Edit Advanced Settings for Box	Click <i>Show Advanced Settings</i> button, the advanced settings of the box will be shown in the right window and can be modi- fied manually.See <i>Advanced setting for Item on page 66</i> .
Show a Box	With the specified size and facing setting, the box can be shown in the 3D View.

61

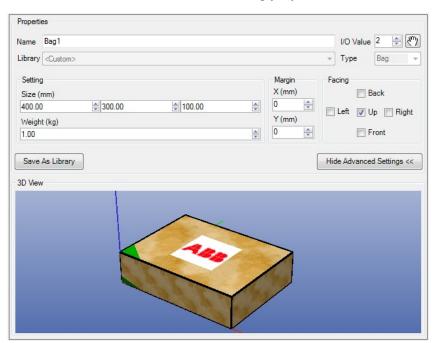
5.3.4 Product/Pallet/Sheet *Continued*



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 66* 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.

Bag Properties

This section describes how to edit the bag properties.

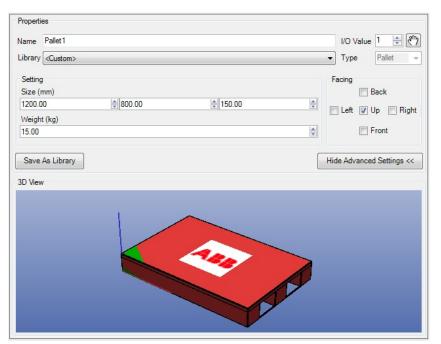


То	Do this
Rename a Bag	Same as Box
Set I/O value	Same as Box
Select Bag from Library	Same as Box
View the Item Type	Same as Box
Set Bag Size	Same as Box
Set Bag Weight	Same as Box
Set Bag Margin	In the Margin number boxes, type the margin distance which is allowed for overlapping between multiple bags 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

5.3.4 Product/Pallet/Sheet Continued

Pallet Properties

This section describes how to edit the pallet properties.

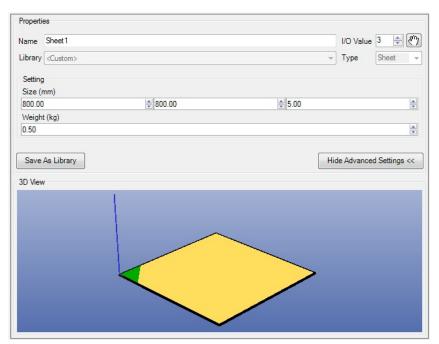


То	Do this
Rename a Pallet	Same as Box
Set I/O value	Same as Box
Select Pallet from Library	Same as Box
View the Item Type	Same as Box
Set Pallet Size	Same as Box
Set Pallet Weight	Same as Box
Set Pallet Facing	Same as Box
Save the Pallet	Same as Box
Edit Advanced Settings for Pallet	Same as Box
Show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

5.3.4 Product/Pallet/Sheet *Continued*

Sheet Properties

This section describes how to edit the Sheet properties.



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

5.3.4 Product/Pallet/Sheet Continued

Group Properties

Properties Box1Group1 Group Name Used Item Box1 Layout Orientation Front • 2 Row Count Column Count 3 I/O Value 2020 ÷ 87 Weight 30 kg 3D View AB ASS

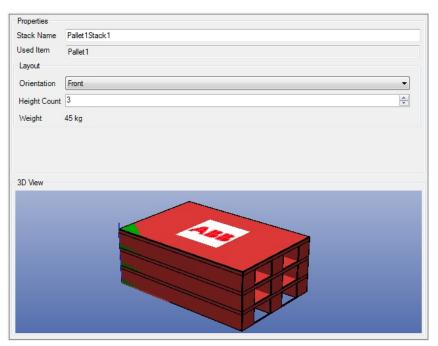
This section describes how to edit the Group properties.

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

5.3.4 Product/Pallet/Sheet *Continued*

Stack Properties

This section describes how to edit the stack properties.



xx110000015

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

Advanced setting for Item

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

5.3.4 Product/Pallet/Sheet Continued

General

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

eneral	Pick/Place	Motion Limits		
AD abel I	Picture			
:\Pro	ogram Files (x8	6)\ABB Industrial I	Robotics IT	Browse
Us	e Texture			
				Browse
Us	e Customized	Model		
				Browse
Provide states	t (mm)		La Desta	
0.00		0.00	0.00	
Orien	itation (deg)	0.00	0.00	
0.00			¥ 0.00	
✓ Sh	ow Outline in	CAD model	Co	lor
J Sh	ow Origin in C	AD model	00	
Арр	ly Tune Value	s		
	nfo			
Fune Ir				
		240.0, 200.0, 190.0		

xx110000016

То	Do this
Set Label Picture	Click the Browse button to select a custom picture as the product label.
Set Texture	Select the Use Texture c heck box to use the selected picture 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. If customized model is used changing the size of the item will not affect the model itself.
Show Outline	Select the Show Outline check box to display the outline of the CAD item as thin lines along the product edges.
Show Origin	Select the Show Origin check box to diaply the origin of the CAD item as green markers on the top and bottom surfaces.

Operating manual - Palletizing PowerPac 3HAC042340-001 Revision: R

Continues on next page

5.3.4 Product/Pallet/Sheet *Continued*

То	Do this
Set Color	Click Color to select the color of the CAD item.
	Note
	The color is hidden if texture is used.
Apply Tune Values	Click Apply Tune Values to set the tuned values as the default values for individual products.

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 186* for detailed information.

eneral Pick/Place Motion			
Vacuum Pick/Place Settings	Default		Tuned
Pick Time	0.010	▲	0.010
Place Time	0.010	▲ ▼	0.010
Vacuum Activation Time	0.100	×.	0.100 🚖
Vacuum Deactivation Time	0.000	A V	0.000

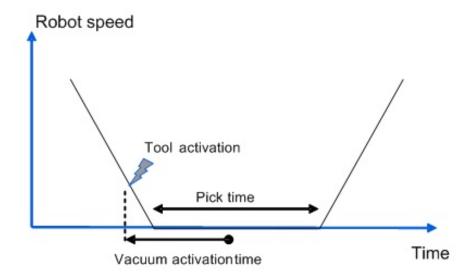
5.3.4 Product/Pallet/Sheet Continued

То	Do this			
Set Time Values for Vacu- um Activation and Deactiv- ation.	In the Vacuum Activation Time and Vacuum Deactivation Time number boxes, type the values.			
Adjust Time that Robot Stays at the Robot Target Position When Picking or Placing an Item.	In the Pick Time and Place Time number boxes, type the time in seconds.			
Set Drop Offset	In the Drop Offset number box, type the offset (in mm).It is useful for releasing bag from a certain height.			
	Note If Drop offset is greater than the product height, then default			
	approach direction is Drop. This is applicable only for bag. If drop offset is less than or equal to product height, then An-			
	gular Approach direction will be set automatically. If required user can change the approach direction from the edit option.			
	Note			
	If product is already used in any of the Operation Set, the fol- lowing pop-up message is displayed when there is change in the Drop Offset value.			
	Palletizing ×			
	Drop offset of following products are changed and they are used by OperationSets of place type: Bag1 (Place PalletPattern11)			
	Do you want to update the drop offsets in the related OperationSets accordingly?			
	Yes No Cancel			
	xx2000001086			
	 Yes - New Drop Offset value shall be applied to all the existing Operation Sets, and will be part of the product. New Operation sets created for this product shall reflect this new Drop Offset value. 			
	 No - New Drop Offset value shall NOT be applied to any of the existing Operation Sets. But it will be part of the product. New Operation sets created with this product, shall have the new Drop offset value. Cancel - New Drop Offset shall be reverted to existing 			
	Drop Offset value.			

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,

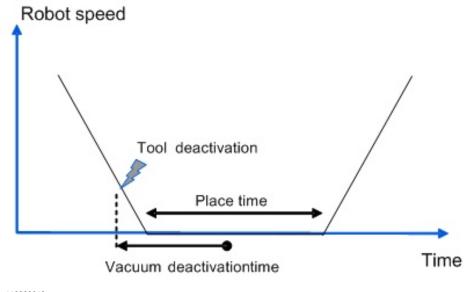
5.3.4 Product/Pallet/Sheet *Continued*

activation will take place after reaching the pick position. If the time is set to zero, activation will occur after half the pick time has passed.



xx110000018

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



5.3.4 Product/Pallet/Sheet Continued

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.

ame Box1		
ieneral Pick	c/Place Motion Limits	
General Mot	ion Limits Default	Tuned
Speed	5000.0	€ 5000.0 ÷ mm/s
Acc/Dec	10.0	10.0 🌩 m/s2
Rot Speed	500.0 500.0 deg/s	
	📝 Use Pick Approach	📝 Use Pick Depart
	Pick Approach	Pick Depart
Speed	5000.0	5000.0
Acc/Dec	10.0	10.0
Rot Speed	500.0	500.0
	Vuse Place Approach	✓ Use Place Depart
	Place Approach	Place Depart
Speed	5000.0	5000.0
Acc/Dec	10.0	10.0
Acc/Dec		

xx110000020

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



There is NO need to set approach or depart speed because these will be limited by the acceleration and deceleration limits. 5.3.5 Pallet pattern

5.3.5 Pallet pattern

Introduction

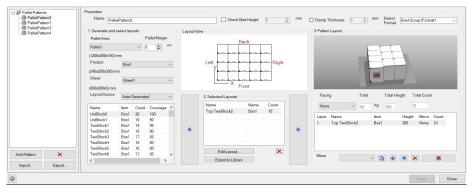
1 On the Palletizing ribbon tab, in the Product Data group, click Pallet Patterns.



2 The **Pallet Patterns** dialog box opens and you can create or import pallet pattern to be used in the station.



The items that are defined before using Product/Pallet/Sheet are listed in the pattern. In a project, at least one pallet pattern should be created. After one pallet pattern is selected in the **Pallet Patterns** list, its corresponding properties are displayed on the right side.



xx2000001085

The following table describes the details of the Pallet Pattern list section:

Item	Description
Add Pattern button	Creates a new pallet pattern with the selected items. For more information, see <i>Pallet Pattern Layout on page 73</i> .
➢ button	Deletes the selected pallet pattern from the pattern list.
Import button	Imports and creates pallet patterns from an XML file. The used items in the XML file will also be created in station.
Export button	Exports the existing pallet patterns into an XML file.

Item	Description
General properties	Name: Displays the name of the selected pallet pattern. You can also edit the name of the selected pallet pattern.
	Check Max Height:Verifies the selected maximum height.
	Clamp Thickness: Allows you to specify the thickness of the clamp static board. This thickness is used to create gaps to accommodate the static board of the clamp between the boxes inside the pattern. Each pattern is updated according to the location of the static board when the clamp gripper places the items.
	Select Format: Allows you to select the format that is used when a clamp gripper is configured. The format selection specifies the relative orientation of the items with the clamp gripper. This relative orientation is used to find the location of the static board while re-calculating the pallet patterns to accommodate the gaps.
1. Generate and select layouts section	Based on the selected pallet, product, sheet and inputted pallet margin, the possible layout is calculated. You can also select the layouts saved in the library.You can choose a layout and click the right arrow button to select into the 2. Selected Layouts section.
2. Selected Layouts section	The selected layouts can be modified, and/or used directly to create layers.The specified layout can also be saved to reuse.You can choose a selected layout and click the right arrow button to add into pattern layers.
3. Pattern Layers sec- tion	Displays the whole layers in a pallet pattern. For each layer you can also see the layout name, and adjust the mirror type.
Layout View section	Displays the 2D view of any selected layout in the above 3 lists of layouts.

The following table describes the details of the Properties dialog box:

Pallet Pattern Layout

This section describes defining the pallet pattern layout. A pallet pattern is built as a stack of item layouts. Layout contains only one type of item.

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



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 populated 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.

The layouts that are saved in the library are selected by choosing **From Library** in **Layout Source**. After you have selected a new sheet, product, or 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.

То	Do this
Access Library Layouts	In the Layout Source drop-down list, select From Library . This will show all layouts from the library converted to use the selected items.
Select Layouts to Use in the Pallet Pattern	In the Layout Source list, select the layouts of interest and click right arrow button to bring it to the Selected Layouts list.
Edit Layout	In the Selected Layouts list, select the layout and click Edit Layout button. This opens up the Layout Editor. See <i>Layout</i> <i>Editor on page 75</i> for detailed description.
Export to Library	In the Selected Layouts list, select the layout and click Export to Library
Show a Layout	In either the Layout Source or the Selected Layouts list, select the layout and it is displayed in the Layout View.
Remove Layout from Selec- ted List	In the Selected Layouts list, select the layout to remove and click the Delete button.

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.

То	Do this	
Add Layers	In the Selected Layouts list, select the layout you want to use and click the right arrow button to add to the Pattern Layers list.	
Remove Layers	In the Pattern Layers list, select the layer and click Click to remove every layer in the Pattern Layers list.	
Reorder Layers	In the Pattern Layers list, select the layer and click 🔮 🔮 to reorder the layers in the list.	
Show a Layer	In the Pattern Layers list, select the layer and it will be shown in the Layout View	
Show Pallet Pattern	All of the layers in the Pattern Layers list will be shown in the Pattern Layers View.	

Layout Editor

Introduction

Layout Editor is to 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 to a pallet pattern configuration for a project.

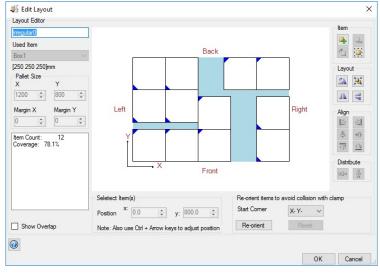
Depending on which layout to edit, you can start the Layout Editor in two ways:

- 1 From the Library, if you want to edit a layout that is included in the library.
 - In the library tree view, select a layout item to edit. For more information, see *Library on page 119*.
 - Click Edit.
 - The Layout Editor window appears and you can edit the layout.
- 2 From a **Pallet Pattern** dialog box, if you want to edit a layout that is included in a **Pallet Pattern**.
 - Select the pallet pattern that includes the layout to edit.
 - In the Selected Layouts list, select the layout to edit.

Continues on next page

- Click Edit Layout
- The Layout Editor window appears and you can edit the layout.

Illustration, Layout Editor



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

Item	Description	
Re-orientation	This feature is supported only in grippers with clamp. The Start Corner combo box contains all the four possible start corner options. On click of Re-orient button, the items are re-oriented to avoid moving clamp collision and the re-orientation is based on start corner, current orientation and the pick format selected from the pallet patterns window. When re-orientation is applied, the layout buttons, and context menu options are disabled. On click of Reset button, re-orientation functionality is reverted.	
	If a layout is re-oriented in a pallet pattern, then new targets need to be generated for all the operation sets that uses the same pallet pattern.	
Layout information	Shows the number of items in the layout and its coverage ratio. If there are any overlapping shapes, this will be noted here as well.	

Proceed with the item

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

Proceed with the layout

То	Do this
Mirror the layout in x direc- tion	Click Flip Horizontal.
Mirror the layout in y direc- tion	Click Flip Vertical.
Rotate the entire layout 180°	Click Rotate.
Center the layout	Click Center.

5.3.5 Pallet pattern *Continued*

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.



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

Related information

Library on page 119 Pallet pattern on page 72 Terms and concepts on page 22

5.3.6 Pick setting

5.3.6 Pick setting

Procedure

1 On the **Palletizing** ribbon tab, in the **Programming** group, click **Pick Setting** to define how the item group or a stack is held by a robot tool.



2 The item and tool orientations can be configured, along with 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.

Solution1:View1 Pick Setting* ×			-
Box1 G-@ Box1Group1	Name Box1Group1Format1	Controller PzPPVirtualSystem	Robot 🕖 T_ROB1 🗸
. SociGroup Formet ((T_ROBI)	Tool Lossion RemTool Function Match. Additional Events Specify the relative position between tool and products Aign position of Camp I . With Rem Tool Aignment Fions Offset fmm)	3D Vew	Tool: 45 kg Total: 50 kg
* X			Tool: 45 kg Total:

xx110000025

3

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

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



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

79

5.3.6 Pick setting *Continued*

Tool Location

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

Name	
Box1Group1Form	at 1
Tool Location	Item - Tool Function Match Additional Events
Align position	of Clamp1 Vith Item1
Tool Alignmer	Terrent Terrent
	Rotation (deg) Image: Constraint of the second

То	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 <i>Item - Tool Function Match on</i> <i>page 81</i> .
Tool Alignment	Tool alignment allows you to position the tool based on the selection relative to the object which is being picked. This is valid only in the case of clamp gripper. For the rest of the grippers the default position is center.
Adjust Offset and Rotation	Input the translation offset and rotation angles (or click the ro- tation 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.

5.3.6 Pick setting Continued

Item - Tool Function Match

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

ool Sc	enario	E	oxPickSo	cenario						
tem	Tool F	Tool Function								
em2 em3		Vacuum 1/Config 1(Default) Vacuum 1/Config 1(Default) Vacuum 1/Config 1(Default)								
Auto	Calculat	te								
] Auto			Zone3	Zone4	Zone5	Zone6	Zone7	Zone8	Zone9	Zone1
] Auto			Zone3	Zone4	Zone5	Zone6	Zone7	Zone8	Zone9	Zone1
	Zone1	Zone2								

xx110000027

То	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 <i>Tool events and</i> <i>scenarios on page 135</i> 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 to automatically calculate the de- fault item-zone matches.

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.

5.3.6 Pick setting *Continued*

See *Create Tool on page 120* for detailed information on picking sensor of different type of tool functions.



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

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.

Туре	Signal Name	Value	Preset Time	Movement
Set	GO Gripper	2	0	PickFirstApproach
Wait	GI_GripperOpen	1	ō	PickFirstApproach
Set	GO_Gripper	1	0	Pick
Wait	GI_GripperClosed	1	0	Pick
Set	GO Gripper	1	0	PickLastDepart
Wait	GI_GripperClosed	1	0	PickLastDepart

xx110000028

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

5.3.7 Add job

5.3.7 Add job

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.

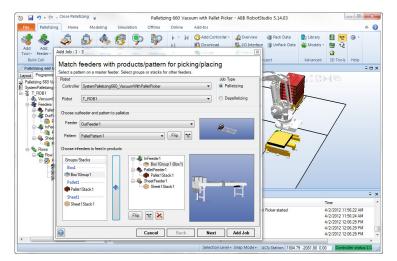
1 On the **Palletizing** ribbon tab, in the **Programming** group, click **Add Job** to open a wizard to create a new job for palletizing/depalletizing in three steps.



xx110000029

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

2 Distribute the roles of each feeders as described in the following table:



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

5.3.7 Add job *Continued*

То	Do this
Select Feeder as Master	Master feeder is usually the feeder where the palletizing/de- palletizing jobs will be started. See <i>Flow on page 202</i> for detailed information.
	In the Feeder selection box, select one feeder as the master. Note that feeders that are not valid to be a master (for example, those being slave feeders for existing flows) are not listed in the selection box.
	Once the master feeder is selected, the other feeders of the robot valid as slave will be listed in the tree view.
	If this feeder is not used as master by any existing flow, when the job creation is finished, a new flow will be created as well, including the newly generated job.
	If this feeder is a master of an existing flow, when the job is created, the existing flow will include the newly created job.
Select Pallet Pattern for the Job	In the Pattern selection box, select one pallet pattern. This pattern will be implemented in this job.In the main 3D view, the selected pallet pattern will be shown on the master feeder.Once the pallet pattern is selected, the necessary groups or stacks to build the pallet pattern will be listed in the list view (i.e. Groups and Stacks list).
Adjust Orientation for Selected Pallet Pattern	Click Flip button, the orientation of selected pallet pattern will be flipped 90 degrees on the master feeder.
Select Feeder to Feed in Products	Select one candidate feeder in the right list, select one group or stack in the left list, click Right Arrow button, the selected group or stack will be added under the selected candidate feeder. It means that this feeder will feed in (or out) products in this palletizing (or de-palletizing) job. Groups or stacks provided by existing operation sets on the feeder will also be listed.
Remove Feeder from Feeding in Products	In the right list, select one group or stack under a candidate feeder, click Delete button, the selected group or stack will be removed from the candidate feeder. It means that this selected group on this feeder will not play a role in this job.
Adjust Orientation for Selected Group or Stack	In the right list, select one group or stack under a candidate feeder, click Flip button, the orientation of selected group or stack will be flipped 90 degrees on this feeder.

3 Click **Next** button to continue to the next step.



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

4 Configure Operation Set for Group or Stack and then Configure Operation Set for Pallet Pattern is described in the following sections.

5.3.7 Add job Continued

Configure operation set for group or stack

After the group or stack is selected, the operation set is configured. 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.

🄞 🖬 🔊 • (≌ ↓ Close Palletizing 🖕	Palletizing 660 Vacuum with Pallet Picker - ABB Robot	Studio 5.14.03	×
File Palletizing Home Modeling Simulatio	n Offline Online Add-Ins	۵	0
Add Add	Image: Second controller Image:		
Tool - Feeder - Build Cell	8	roject Advanced 3D Tools Help	
Palletizing 660 V Configure operation set for	r group or stack	view advanced 30 roos nep	1 X
Layout Programmer Check pick setting selection for the groups of	r stacks.		-
Palletizing 660 Va Pick from infeeders	Name Pick Box1Group1Format1	Z Z Z Z OHO	
SystemPalletizing			
	Product Info Item Box1		
Feeders Pallet Pallet Pallet		7	
e OutF Pick Palet1Stack1	Group Box1Group1		
Ball InFer SheetFeeder1	Pick Setting		
Carl Pick Sheet 1Stack 1	Box1Group1Format1 (T_ROB1)		
Г. 🤹 F.			
E-Service Flows			
		/	
L			
			×
		Time	^
		t Picker started 4/2/2012 11:56:22 AM 4/2/2012 11:56:24 AM	
Check Reach	Preview Palletizing	4/2/2012 11:36:24 AM 4/2/2012 12:06:29 PM	
	Cancel Back Next Add Job	4/2/2012 12:06:29 PM	
	Cancer Dack Next Add Job	4/2/2012 12:06:29 PM	-
	Selection Level + Snap Mode +	UCS: Station 1804.79 -2081.80 0.00 Controller status: 1/1	
<u></u>			

То	Do this
Rename the Operation Set	In the Name text box, type a new name for the operation set. In default, a unique name can be generated automatically when a new operation set is created.
Select Pick Setting	Each group or stack should have at least one pick setting defined before. Based on the selected group or stack, one ap- propriate pick setting can be selected in the Pick Setting selec- tion box for this operation set.
Set Stack Search	In some cases before picking the items (usually for pallet or sheet), the position of the items should be searched.Select Stack Search check box, the robot will search the height of pallet or sheet before picking it.
Check Reach	Click Check Reach button to validate the reachability of all ro- bot targets related to picking/placing the group/stack.
	 The color of the icon next to the Check Reach button will show the check result: Green: all robot targets can be reached Red: some of robot targets cannot be reached.
	If it is red, then move mouse cursor over the icon 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. For more information, see <i>Check reach on page 88</i> .
	You can adjust feeder position, flip group/stack position, or move robot to make all targets reachable.

5.3.7 Add job *Continued*

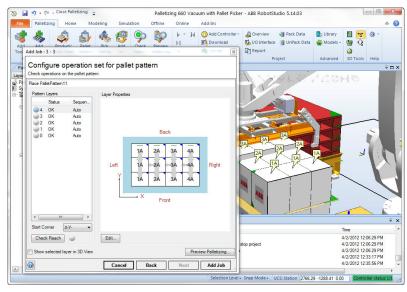
То	Do this
Preview Palletizing	Click Preview Palletizing button, a preview dialog will pop out to visualize all the robot targets in the main 3D view, which you can go through step-by-step. For more information, see <i>Preview Palletizing on page 90</i> .

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, and pick settings are selected for slave feeders, the operation set for the master feeder is configured. In general, if this job type is palletizing, the action for the operation set 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.



То	Do this
Rename the Operation Set	In the Name text box, type a new name for the operation set. In default, a unique name is generated automatically when a new operation set is created.
Edit the Pallet Pattern	Select one layer in the Pattern Layers list, this layer layout will be shown in Layer Properties view.
	Click Edit button, a dialog box will be shown to edit the selected layer. See <i>Layout Editor on page 75</i> for detailed description.
	Select Start Corner combo box, this layer will change its start corner according to the selection.

5.3.7 Add job Continued

То	Do this
Check Reach	Click Check Reach button, to validate the reachability of all robot targets related to picking/placing the pattern.
	The color of the icon next to the Check Reach button will show the check result:
	Green: all robot targets can be reached
	 Red: some of robot targets cannot be reached.
	If is is red, the unreachable items in the selected pattern will turn red in main 3D view. For more information, see <i>Check reach on page 88</i> .
	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 <i>Preview Palletizing on page 90</i> for detailed description.
Show Selected Layer in 3D View	Select Show Selected Layer in 3D View check box and select one layer in the Pattern Layers list, the selected layer will be shown in 3D View and layers above it will be invisible.

Click Finish button to complete the wizard.

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

5.3.8 Check reach

5.3.8 Check reach

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

1 On the Palletizing ribbon tab, in the Validate group, click Check Reach button, to validate the reachability of all robot targets related to picking/placing the pattern. If all robot targets are reachable, the icon displays green. Otherwise it turns red.



2 If there are unreachable targets, a warning dialog box pops up with the message 'Do you want to see details'. If you click Yes, a list appears on the right side with the reach status of all the items.



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

Check Reach		₹ ×
Checked Program	Source	Туре
Pick Box1Group1F	InFeeder1	Operation Set
Place PalletPattem	OutFeeder1	OperationSet
٠ III		•
Check Cl	heck All	Close

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

5.3.8 Check reach Continued

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

5.3.9 Preview Palletizing

5.3.9 Preview Palletizing

Previewing the Palletizing operation

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

To preview the Palletizing operation:

1 On the **Palletizing** ribbon tab, in the **Validate** group, click **Preview Palletizing**. The preview interface is displayed in the right window. All existing operation sets are listed in the Palletizing Programs group.

Preview Palletizin Palletizing Program	_			Ŧ
Operation set		Feeder	Robot	Controller
Pick Box1Group1	Fo	InFeed	T_ROB1	System48
Pick Pallet1Stack	:1	PalletF	T_ROB1	System48
Pick Sheet1Stack	c1	Sheet	T_ROB1	System48
Place PalletPatter	m11	OutFe	T_ROB1	System48
Preview				
- Show with	– Pla	y Control		
Tool	Sp	beed		Step /22
Am Configuration	-1,0),-1,0	-	Apply Config
Reset Configuration			-	Close
Indicators Show Robot	Targe	ts	Show T	Farget Description
0				Close

xx1600001365

2 Select an operation set in the list, and select preview preview palletizing options according to your requirement.

5.3.9 Preview Palletizing Continued

The following figure and table provides information regarding the preview palletizing options.

Show with	Play Control	
	Speed	Step /38
Tool		
	,Ų	0
🔽 Robot		I
Arm Configuration	-12,-3,1 🔹	Apply Config
Reset	•	Close
Configuration	Reset Current Target	
Indicators	Reset All Layers	

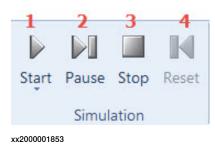
Field	Description
Play Control- Speed	You can drag the button in the speed track bar and adjust the preview play speed from the slowest to the fastest.
Play Control - Step	Type a step number in the box and press ENTER. The preview goes to the specified step.
Play controls	Click to play all preview steps one by one from the current step.
	Click to pause the preview step at the current step.
	Click to stop the preview step the first step.
	Click or the up arrow in Steps number box to navig- ate the previous step.
	Click or the down arrow in Steps number box to navigate to the next step.
Show with Tool	Select the Tool check box to display the tool during the preview process.
Show with Robot	Select Robot check box, the robot holding the tool jumps to each robot target during the preview process. If the robot target is not reachable, the robot will not move to this robot target.

5.3.9 Preview Palletizing *Continued*

Field	Description
Arm Configuration	The Arm Configuration list displays all the possible arm configurations for a selected target in the operation set. You can define Safe targets and Search targets with their respective configuration. Click Apply Config to apply the selected configuration to the current target.
	 Following are the known limitations of Arm Configuration option: Arm configurations for General Offset targets, Horizontal Approach/Depart targets, and Extra Z targets have by default the same configuration as the item placement target and cannot be edited.
	 If a Pallet pattern has the same layout pattern in all the layers, then the Arm configurations are same through all the layers. The arm configurations within a layer can be changed but cannot be changed layer to layer.
	Note
	When defining search targets you can select the reachable Arm configurations for Search targets and subsequent Box pick targets.
	Note
	The configurations are retained for small displacements in Feeder positions and Operation set displacements. For larger displacements Arm configurations may have to be redefined.
Reset Configuration	The Rest Configuration list allows you to reset the config- urations. Configurations could be reset for each target as well as for the entire operation set by selecting Reset All Layers.
	Note
	To reset the configuration of an individual operation, first select the operation set in the browser tree.
	Note
	When a target is reset, the configurations for the all the offset targets surrounding the target are reset.
Show Robot Targets	Select Show Robot Targets check box, the robot targets related to the operation set will be shown in main 3D view.
Show Target Description	Select Show Target Description check box, the description about the robot targets will be shown in main 3D view.

5.3.10 Simulation

5.3.10 Simulation



Click Start, the following three simulation steps will be executed automatically and the operator interface will be displayed in the right window.

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

 Click Pause to pause the simulation.

 On Pausing, the button changes to Step. The step allows you to advance the simulation step by step.

 Click Stop to end the simulation.

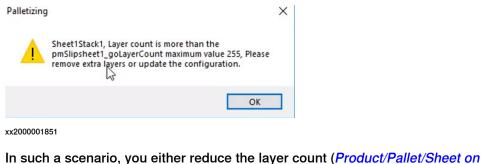
 Stop button is enabled once the simulation starts.

 Click Reset, the simulation environment will be reset.

 The products created during the simulation will be cleaned and the station is ready for next simulation.
 Reset button is enabled when simulation is not started.



If the layer count is more than the max value set, simulation stops running with an error message as following:

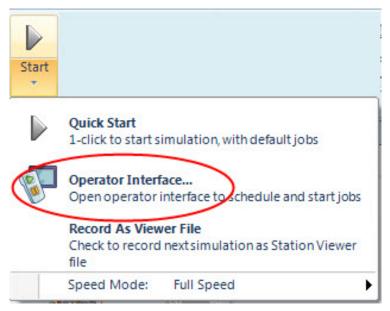


In such a scenario, you either reduce the layer count (*Product/Pallet/Sheet on page 59*) or increase the max value set in Configuration, refer *3HAC032104 Operating manual - RobotStudio*.

5.3.11 Operator interface

5.3.11 Operator interface

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



xx_120000014

The project status is shown on top of the interface:

Project status	Description	Status Light
Started	The project has been started successfully.	xx110000044
Starting	The project is in the starting process.	>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Stopped	The project has been stopped.	xx110000046
Error	Error occurs during the project execution.	>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

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

5.3.11 Operator interface Continued

Control page

Operator Interface Project		÷
Flow Control Statistics		
Controllers and Flows		
Flow Robot Controller660_PnG_Ph Flow1 T_ROB1		Start Flow(s)
		Stop Flow(s)
		Stop Option
		Finish Cycle \vee
Scheduled Jobs for Flow1 Pallet Patterns	Layer Coun	t Product Count
Place PalletPattern21	2	3
Run Continuously		♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥<
0		

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

xx2000001765

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

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

5.3.11 Operator interface *Continued*

Flow control

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

То	Do this
Start a Flow	Select one flow in the Controllers and Flows group whose status is
	Stopped, click click 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 dis- abled.
Stop a Flow	Select one flow in the Controllers and Flows group whose status is Run- ning, click 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 dis- abled.
Start All Flow(s)	Click b 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 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.

Scheduled Jobs

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

То	Do this
Add Job	In the Scheduled Jobs group, click \clubsuit 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 $ imes$ to delete the selected job 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.

5.3.11 Operator interface Continued

То	Do this
Pallet Patterns	In the Scheduled Jobs group, set the Layer Count and Product Count.
	Note
	The user can provide an option to run the simulation from a specific layer and product count.
	For example, if Layer Count=2 and Product Count=3, then the simulation will start from layer 2 and starts placing the product from 3 onwards.
	That is,
	layer 1 is bypassed
	 products are placed from 3 onwards in layer 2
	no change in the functionality of the remaining layers
Layer Count	In the Scheduled Jobs group, set the layer count for simulation.
Product Count	In the Scheduled Jobs group, set the product count for simulation.

5.3.11 Operator interface *Continued*

Statistics Tab

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

imulation	_			
Product	In	Out	Th	nroughput (/h)
Box1	6	3	16	572
Pallet1	0	0	0	
Bag1	0	0	0	
Sheet1	0	0	0	
Message	D-H11, D-		ìme (s)	Category
Place Pallet	Pattern 11: Dr	opping p 4	.2	PositionNotMate
Place Pallet Place Pallet	Pattern 11: Dr	opping p 4 opping p 4	.2 .2	Position Not Mate Position Not Mate
Place Pallet Place Pallet		opping p 4 opping p 4	.2	Position Not Ma
Place Pallet Place Pallet	Pattern 11: Dr	opping p 4 opping p 4	.2 .2	PositionNotM PositionNotM

xx110000055

When the simulation is started, the statistics information is continuously updated to display the palletizing process data.

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

5.3.12 Record as viewer File

5.3.12 Record as viewer File

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

5.3.13 Speed mode

5.3.13 Speed mode

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

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

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

5.3.14 Add controller



On the **Palletizing** ribbon tab, in the **Transfer** group, under **Add Controller** click one of the following:

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

For more information on Online functionality, See Operating manual - RobotStudio.

5.3.15 Download

5.3.15 Download



xx110000059

On the **Palletizing** ribbon tab, in the **Transfer** group, click **Download**, the download interface will be displayed. 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 to Onlin	e Controller(s)	₹	×
Download to Virtual	Controller(s)		
Virtual Controller	Online Controller	1	
SystemPalletizing660_\	/acuumWith System120(Virtual)		
- Online Controller Info-			
ID	61cce257-5e58-4fb2-b705-21f0c11c0504		
System Name	System120		
Controller Name	CN-L-6051016		
RW Version	5.14.0.0		
Available	True		
Virtual	True		
IP	127.0.0.1		
			_
	Download C	lose	

5.3.15 Download Continued

То	Do this
Download Program to On- line Controller	Clear the 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).
	Note
	The download to online controllers is task based. You can se- lect any particular task in the online controller to download from the station.
Download Program to Virtu- al Controller	Select Download to Virtual Controller check box, and click Download button to download the program to virtual control- ler(s).

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.

5.3.16 Upload

5.3.16 Upload

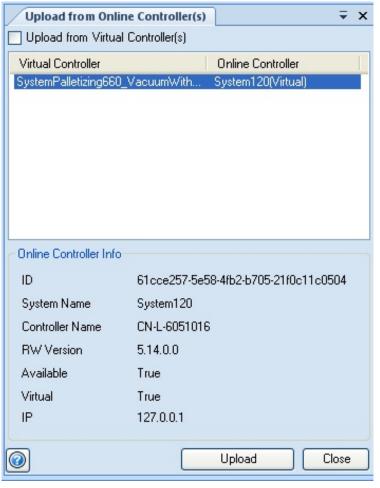


xx110000061

On the **Palletizing** ribbon tab, in the **Transfer** group, click **Upload** button in ribbon, the upload interface gets displayed. 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.



5.3.16 Upload Continued

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

5.3.17 Project overview

5.3.17 Project overview



xx110000063

On the **Palletizing** ribbon tab, in the **Project** group, click **Overview** button in ribbon to display the overview information for the project.

^r oject Name	Palletiz	ingDemo			Project Descr	iption			
Controller Overview	SysPP	PExt		•					
Operation Se	t	Feeder	Туре	Content	Formats	Products	Product I/O	Format I/O	
Pick Box1Gin Place PalletF		InFeeder1 OutFeeder1	Pick Place	Format Pattern	Box1Grou Box1Grou	Box1 Box1	0	0 N/A	
									Save As Report

xx110000064

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

5.3.18 I/O interface

5.3.18 I/O interface

🗓 I/O Interface

xx110000065

On the **Palletizing** ribbon tab, in the **Project** group, 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.

Products/Pallets/Sheets Generate Default Name I//0 Value Box1 0	Name				Generate Defaul
	Name				-
Box1 0		1/0 Value	Feeder	Robot	Controller
	Pick PPForPick	Û	InFeeder1	T_ROB1	SystemPalletizing
Pallet1 1	Place PalletPattern	10	OutFeeder1	T_ROB1	SystemPalletizing
Groups					
Name I/0 Value Parent Product					
Box1Group1 10 Box1 Box1Group2 20 Box1					
Box1Group2 20 Box1					
	<				
Nows	Feeders				
Generate Default					Generate Defau
Name I/0 Value Robot Controller	Name I/O	Value Rob	ot Ci	ontroller	
Flow1 0 T_ROB1 SystemPalletizingMultiPickPlace	InFeeder1 1	T_RC	JB1 Sy	stemPalletizin	gMultiPickPlace
	OutFeeder1 2	T_RC	JB1 Sy	stemPalletizin	gMultiPickPlace

5.3.18 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.

Values Project I/O Signals		Save as Repo
	for for remote project starting with PLC)	
Controllers		
Name	Туре	
SystemPalletizingMultiPickPlace(Virt	ual) Virtual	
Projects on System Palletizing MultiPic	(Place(Virtual)	
	0 Value	
TestMultiPickPlace 1		Import
		Export
		×
		Generate Defau

xx110000067

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

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

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

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

5.3.18 I/O interface Continued

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.

/0 Values Project I/0 Signals				Save as Rep
ist of all robot controller signals used in	the project			
Object	Туре	Usage/Purpose	Signal Name	
SystemPalletizingMultiPickPlace	Controller	Error Trigger	pmEvent_diTrigger	1
SystemPalletizingMultiPickPlace	Controller	Error Source	pmEvent giErrorSource	
vstemPalletizingMultiPickPlace	Controller	Error Message	pmEvent giMessage	
acuumGripper	Tool	giSCToolActivators	pmGripper1 goActivators	
acuumGripper	Tool	giOpenClaw	pmGripper1_goToolEvent1	
acuumGripper	Tool	goClawOpened	pmGripper1_giTooEvent1	
acuumGripper	Tool	giOpenSearcher	pmGripper1_goToolEvent2	
acuumGripper	Tool	doSearcherOpened	pmGripper1_diActivator1Opened	
acuumGripper	Tool	doSearcherClosed	pmGripper1_diActivator1Closed	
low1	Flow	Status	pmFlow1 goStatus	
nFeeder1	InFeeder	Position Request Trigger (DO)	pmInfeeder1_doPosRegTrig	
nFeeder1	InFeeder	Product Selection (GO)	pminfeeder1_goProdSel	
Feeder1	InFeeder	Format Selection (GO)	pminfeeder1_apFormSel	
nFeeder1	InFeeder	Requesting Master (GO)	pminfeeder1_goRegMaster	
nFeeder1	InFeeder	Target Generation Trigger (DI)	pmInfeeder1_diTatGenTria	
Feeder1	InFeeder	Product Selection (GI)	pmInfeeder1_giProdSel	
Feeder1	InFeeder	Format Selection (GI)	pminfeeder1 gFormSel	
nFeeder1	InFeeder	Start Laver Count (GI)	pmInfeeder1_giStartLaverCount	
nFeeder1	InFeeder	Start Product Count (GI)	pmInfeeder1_giStartProdCount	
Feeder1	InFeeder	Position Available (DO)	pmInfeeder1_doPosAvail	
nFeeder1	InFeeder	Operation Set Complete (DO)	pmInfeeder1_doDpSetCompl	
nFeeder1	InFeeder	Queue Empty (DO)	pmInfeeder1_doQueueEmpty	
nFeeder1	InFeeder	Execution State (GO)	pminfeeder1_goExecState	
Feeder1	InFeeder	Height State (GO)	pmInfeeder1_goHeightState	
Feeder1	InFeeder	Laver Count (GD)	pmInfeeder1_goLayerCount	
nFeeder1	InFeeder	Product Count (GO)	pmInfeeder1_goProductCount	
Feeder1	InFeeder	Stop Job (DI)	pminfeeder1_diStopJob	
nFeeder1	InFeeder	Redo Search (DI)	pmInfeeder1_diRedoSearch	
nFeeder1	InFeeder	Robot Execution (DI)	N/A	
lutFeeder1	OutFeeder	Position Request Trigger (DO)	pmOutfeeder1_doPosReaTria	
lutFeeder1	OutFeeder	Product Selection (GO)	pmOutfeeder1_goProdSel	
lutFeeder1	OutFeeder	Format Selection (GO)	pmOutfeeder1 goFormSel	
lutFeeder1	OutFeeder	Requesting Master (GO)	pmOutfeeder1_goRegMaster	
)utFeeder1	OutFeeder	Target Generation Trigger (DI)	pmOutfeeder1_diTatGenTrig	
JutFeeder1	OutFeeder	Product Selection (GI)	nmDutfeeder1_gProdSel	

xx110000068

5.3.19 Project report

5.3.19 Project report



xx110000069

On the **Palletizing** ribbon tab, in the **Project** group, 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.

🎻 Report					l		• X		
Box1Group1 E			ront	100%	•				
Pallet Pattern		ayer	Layout Name	Item	lte		Mirror		
					Co	unt			
PalletPattern1									
t	P Back Rig	1	Top UniBlock1	Box1		12	ClockWi0		
, , t	P Back Ric Front X	2	Top UniBlock1	Box1	22	12	Non		
Operation sets									
Name	Content	Feeder	r Pick Or P	lace	Robot	Con	troller		
Pick Box1Group1Format1	Box1Group1For mat1	Pick Box1Gi up1For			T_ROB 1	Sys	PPPExt		
						[Close		

xx110000070

5.3.20 Pack data

5.3.20 Pack data

1 In the **Project** group, click **Pack Data**, the **Pack Data** interface is displayed in the right window.

o Pack Data

xx1200000122

The available virtual controllers and online controllers are listed in the drop down menu.

2 Click Pack Data To.

5.3.20 Pack data *Continued*

The **Pack Data** creates a Pack and Go file of the RobotStudio Station in *.rspag* format and transfers it to the online/virtual controllers.

Pack Data to Cont	roller(s) $\overline{}$ ×
Virtual Controller	Online Controller
SystemPalletizing660	Click here to select
Include Libraries Online Controller Info ID	0
System Name	0
Controller Name	0
RobotWare Version	0
Available	0
Virtual	0
IP	0
0	Pack Data To Close

xx_120000013

То	Do this
Pack Data to Online Con- troller	After the online controller(s) is added, the available online controller(s) is available for selection in the drop down menu Online Controller . Click Pack Data To in tool window and select Online Controller(s).
Pack Data to Virtual Con- troller	Click Pack Data To in tool window and select Virtual Controller(s).

5.3.20 Pack data Continued

То	Do this
Include Libraries	Select Include Libraries present in the dialog to include them while packing data.



Note

If a similar palletizing project data exists in the controller, the data would be overwritten.

5.3.21 UnPack data

5.3.21 UnPack data

1 In the Project group, click UnPack Data .

📑 UnPack Data

xx1200000123

The UnPack Data interface is displayed in the right window.

/	UnPack Data from Controller(s)	Ŧ	×
	Select		
	Controller Type		
	Virtual Online		
	Controller Name		
	SystemPalletizing660_VacuumWithPalletPicker	•	
	Project Name		
	Palletizing 660 Vacuum with Pallet Picker	•	
	Pack and Go file		
	Click here to select	•]
(UnPack Clo	se	
α 1	120000013		

Controller Name: Corresponds to the selected controller **Project Name**: Corresponds to the Palletizing Project name

5.3.21 UnPack data Continued

Pack and Go file: Corresponds to the Pack and Go file created

2 Select either Virtual or Online as the controller type.



Note

Based on the selection the Virtual or Online controllers would be populated along with the available pack and go files.

- 3 Click and select the Pack and Go file.
- 4 Click UnPack Data.



- 1 The Pack and Go file can also be accessed from the controller system folder.
- 2 The system that is unpacked will not have the project information in the form of Pack and Go (.rspag). Pack data needs to be done inorder to have the same.

5.3.22 Delete projects

5.3.22 Delete projects

💋 Delete Projects 🛛

en1300000344

The controller (Virtual or Online Controller) contains reference of projects. The **Delete Projects** option allows you to delete the saved projects from the selected Controller.

To delete a project:

1 Click the Delete Projects button.

The Delete project from controller window is dispalyed.

ystemPalletizing460_ClampC	IneZone 🔻
Projects	
 Project MultiRobot Project 0 Palletizing 460 Clamp 2controller 1robot 	

en1300000343

- 2 Select the controller type from the **Select Controller Type** section. The saved projects in the selected controller are displayed.
- 3 Select the project(s) that you want to delete.
- 4 Click the **Delete** button.

The selected projects are removed from the selected controller.

5.3.23 Import PM5

5.3.23 Import PM5

1 In the Project group, click Import PM5,



ImportPM5 Project	X
Import Select PM5 Project File	
BTC_911_Tool_Configuration	on.pmproj
Controller Mapping	1
Virtual Controller	Pickmaster Controller
TestSys	Controller1
Task Mapping	
Virtual Task	Pickmaster Robot
T_ROB1(TestSys)	Robot1(Controller1)
	Import Close

xx1200000125

The Import PM5 interface is displayed in the right window.

- 2 Browse and select the PickMaster5 project to be imported (.pmproj).
- 3 Map the controllers in RobotStudio with the controllers in PickMaster5 project.
- 4 Map the tasks in RobotStudio with the tasks in PickMaster5 project.
- 5 Click Import.

Continues on next page

5.3.23 Import PM5 *Continued*

6 The PickMaster5 project is converted to Palletizing PowerPac project structure.

Note

- 1 The Import function will check for *.pmline* file before reading the PickMaster project.
- 2 Default feeders are used which are provided as part of Palletizing PowerPac.
- 3 Supports addition of Vaccumn gripper tool only
- 4 The values of Format offset, Format rotation and operation sets are read from the *.pmproj* file, the values are updated during to Palletizing PowerPac AutoCorrect.
- 5 User needs to update the workobject and tooldata values from the controller.
- 6 If ItemZoneMap is not manual, the zone mapping is re-calculated automatically and overwritten.

5.3.24 Library

5.3.24 Library

📴 Library

xx110000071

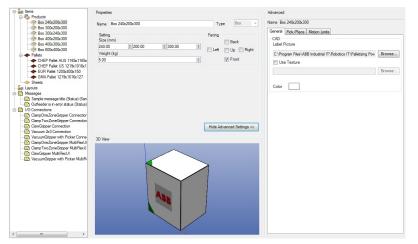
In the **Advanced** group, click **Library** to display the libraries used in the project. Library contains template products, pattern layouts, controller messages and tool I/O connections.

In Palletizing PowerPac, there are two places where library files are saved:

- Installation directory: C:\Program Files (x86)\ABB\Palletizing PowerPac 2019\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.



xx110000072

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

5.3.25 Models

5.3.25 Models

Create Tool



xx110000073

The **Create Tool** interface is used to create a Palletizing PowerPac 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.

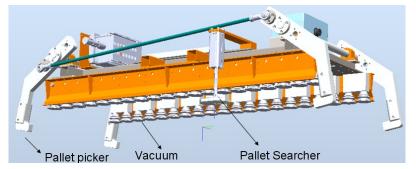
Tool Functions

Introduction

A Palletizing PowerPac 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 168* 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 Palletizing PowerPac contains the following three tool functions:



xx110000074

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

5.3.25 Models Continued

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

General tab

General Tool Functio	ns		
Tool Name	ClawGripper		
I/O Connection Templ	ate ClawGripper Conne	ection	•
Tool Height And We	ight		
UseTool Height	800		* *
Tool Weight	60.0		*
Center of Gravity (mm)			
0.00	0.00	🚔 10.00	▲ ▼
Inertia Ix, Iy, Iz (kgm²)			
1.11~	€ 0.33~	⊜ 0.81~	×
	Max Payload (Kg) TCP (Relative to T Position (mm) 0.00		
	Orientation (deg)		
	0.00	00.0	
+	K		

	Description
I/O Connection Template	Choose a default I/O template for this tool. The chosen template will be used as default when the tool is attached to a robot. The available templates are read from library.
Use Tool Height	The height of the tool. This value is considered as the default tool height. If this option is not selected, default height of the tool based on the tool functions is considered as the tool height.
Tool Weight	The weight of the tool. This value is applied to the related tool data in station when this tool is attached to a robot.
Center of Gravity (mm)	Center of gravity of the tool. This value is applied to the related tool data in station when this tool is attached to a robot.
Inertia Ix,Iy,Iz (kgm ²)	Inertia of the tool. This value is applied to the related tool data in station when this tool is attached to a robot.
Tooldata List	A list of tool data that are contained by this tool. Some tool functions of this tool may use one of the tool data. The tool data will be created or overridden accordingly in station when this tool is attached to a robot.
Name	Name of the tool data.
Max Payload (Kg)	Maximum payload for this tool data. In Pick Setting dialog, if the tool function using this tool data is specified to pick products more than this payload, a warning appears.
TCP (Relative to Tool Ori- gin)	The position and orientation of the tool data.

To add/edit/remove tool functions

There are currently 6 types of tool functions defined in Palletizing PowerPac. Click



xx110000077

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

Vacuum



xx110000078

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

The editing UI for Vacuum is as following:

Properties	Description						
General		You can change the tool function name, choose the tool data, and input a control signal name for tool SmartComponent.					
	Vacuum1		Tooldata	tVacuum	~		
	Control Signal (GI) giSCT oolActivators						
	xx110000080						
	Note						
	The control sig controller signa		tool Sma	artCompone	nt, and is no	t a robot	

5.3.25 Models *Continued*

Duranantia	Description
Properties	Description
Activators	An activator is a physical correspondent to the control signal, and controls one part of the tool, for example a vacuum cup. The figure shows a configuration example
	Vacuum Activators Zones Activators Zones Activators Image: Start Bit Bit Length A/D/(I) Act1 0 1 0/1 Act2 1 1 0/1 Act3 2 1 0/1 Act4 3 1 0/1 Act5 4 1 0/1 Act6 5 1 0/1 Act7 6 1 0/1 Act8 7 1 0/1 Act9 8 1 0/1
	Act1 Act1 Act1 Position (Relative to TCP) (mm)
	0.000 -540.000 0.000
	Size (x,y,z) (mm) 280.000 \$100.000 \$20.000 \$
	 xx110000081 To add a new activator, click . To remove an activator, click . 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 125.

5.3.25 Models Continued

uration is eral conf shows a	a colle s a setu igurati	ection up of z	of acti ones t	vators			es.			
(· · · · ·	eeiing	uratior		one a	tool ca	an hav	e. A	tool	while a con can have s e. The figur	sev-
Vacuum Ac	tivators Z	Zones								
~ Configurati	ons									
Vacuum1/	Config1(De	efault)								
- Configurati	on									
Config1										
Activator/Z	one Mapp	ing								
	Zone1	Zone2	Zone3	Zone4	Zone5	Zone6	Zone	7 🔨		
Act1										
Act2		 Image: A start of the start of								
Act3			 Image: A start of the start of					=		
Act4				 Image: A start of the start of						
Act5					 Image: A start of the start of					
Act6						 Image: A start of the start of				
Act7							 Image: A set of the set of the			
Act8								~		
						-				
Add Zor	ne					De	elete Zo	ne		
To add o In the Co click on a To add o To incluc ator row	r remo onfigura a config r remo le an a and th	ations guratic ve a zc ctivato e zone	group on and one, cl or into e colur	; To so selec ick "Ao a zono nn.	et a co t Set a dd Zor e, sele	onfigur as Defa ne" and oct the	ratior ault (d "De chec	n as Con elete ck b	default, rig figuration; e Zone" butt ox at the ac	ton.
	Vacuum Ac Configurati Configurati Config1 Activator/Z Act1 Act2 Act3 Act4 Act5 Act6 Act7 Act8 Act6 Act7 Act8 Act8 Act8 Act8 Act8 Act8 Act8 Act8	Vacuum Activators Configurations Vacuum1/Config1(Du Configuration Configuration Config1 Activator/Zone Mapp Act1 Act2 Act3 Act4 Act5 Act6 Act7 Act8 Act7 Act8 Configuration Configuration Configuration Act2 Act3 Act4 Act5 Act6 Act7 Act8 To add or remoo In the Configuration Click on a configuration To add or remoo To include an a ator row and th	Vacuum Activators Zones Configurations Vacuum1/Config1(Default) Configuration Configuration Configuration Configuration Configuration Configuration Configuration Configuration Configuration Configuration Activator/Zone Mapping Zone1 Act1 Zone1 Act2 Zone2 Act3 Act3 Act4 Act4 Act5 Act6 Act7 Act6 Act7 Act7 Act8 Zone xx110000084 To add or remove a configuration To add or remove a zon To include an activato To include an activato ator row and the zone	Vacuum Activators Zones Configurations Vacuum1/Config1(Default) Configuration Configuration Configuration Configuration Configuration Configuration Configuration Configuration Activator/Zone Mapping Activator/Zone Mapping Activator Activator Activator	Configurations Vacuum1/Config1(Default) Configuration Config1 Activator/Zone Mapping Zone1 Zone2 Act1 Act2 Act3 Act4 Act5 Act6 Act7 Act8 Act7 Act8 Act7 Act8 Configurations group; To stations group; To stations group; To stations group; To station and selection To add or remove a zone, click "Acta to include an activator into a zone ator row and the zone column.	Vacuum Activators Zones Configurations Vacuum1/Config1[Default] Configuration Configuration Configuration Configuration Configuration Configuration Configuration Configuration Configuration Configuration Activator/Zone Mapping Activator/Zone Mapping Activator/Zone Mapping Activator/Zone Mapping Act1 V Act2 V Act3 V Act4 V Act5 V Act6 V Act7 V Act8 V Act7 V Act8 V Act9 V Act1 V Act6 V Act7 V Act8 V Vacuum V Act7 V Act8 V Cone V Xx110000084 V To add or remove a zone, click "Add Zor To add or remove a zone, click "Add Zor T	Vacuum Activators Zones Configurations Vacuum1/Config1(Default) Configuration Configuration Configuration Configuration Configuration Configuration Configuration Configuration Configuration Configuration Activator/Zone Mapping Configuration Activator/Zone Mapping Configuration Act1 V Act2 V Act3 V Act4 V Act5 V Act6 V Act7 V Act6 V Act7 V Act7 V Act6 V Act7 V Act7 V Act7 V Act7 V Act8 V Configurations group; To set a configuration In the Configurations group; To set a configuration Click on a configuration and select Set as Def. To add or remove a zone, click "Add Zone" and To include an activator into a zone, select the	Vacuum Activators Zones Configurations Vacuum1/Config1(Default) Configuration Configuration Configuration Configuration Configuration Configuration Configuration Configuration Configuration Configuration Activator/Zone Mapping Configuration Act1 V Act2 V Act3 V Act4 V Act5 V Act6 V Act7 V Act8 V Act7 V Act7 V Act8 V Act9 V Act1 V Act6 V Act7 V Act7 V Act8 V Vacuum Vac	Vacuum Activators Zones Configurations Image: Configuration (Config1) Activator/Zone Mapping Image: Configuration (Config1) Activator Image: Configuration (Config1) In the Configurations group; To set a configuration as click on a configuration and select Set as Default Configuration (Config1) To add or remove a zone, click "Add Zone" and "Delete (Configuration (Config1) To include an activator into a zone, select the check b ator row and the zone column.	Vacuum Activators Zones Configurations Image: Configuration in the configuratin the configuration in the configuration in th

Activators properties

The following table describes the settings of the activators:

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

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

Clamp



xx110000086

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

Properties	Description				
Open/Close Clamp	Opening and closing of clamp can be controlled by DI or GI signals, and when the open or close state is reached, it can set DO or GO signals to certain values. Also a mechanical movement in 3D view may be included to visualize the openin and closing.				
	Open Clamp Close Clamp Open Hook Close Hook Picking Sensor (Simulation)				
	Signal Name Value Type gi gi QpenClamp 1 Set				
	C Status Signals (Dutput)				
	Signal Name Value Type goClampOpened 1 Set				
	Joint Positions (for simulation) Move Duration 0.20 🗢 s				
	J1 0.000 180.00 < >				
	✓ J2 0.000 85.000 < > J3 < 0.000				
	xx110000087				
	Take the above setting for Open Clamp as example:To add/edit/remove a signal to trigger the open movement, click				
	🛨 , 📴 , or 🔀.				
	In the Control Signals (Input) group. You can specify the signat type (DI/DO/GI/GO), the signal set value, the pulse value and				
	In the Control Signals (Input) group. You can specify the signat type (DI/DO/GI/GO), the signal set value, the pulse value and length. See <i>Edit signal control on page 134</i> for detailed inform ation. To add/edit/remove a signal to be triggered when open move				
	 In the Control Signals (Input) group. You can specify the signat type (DI/DO/GI/GO), the signal set value, the pulse value and length. See <i>Edit signal control on page 134</i> 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 <i>Edit signal control on page 134</i> for detailed information of setting signal and value. 				
	 In the Control Signals (Input) group. You can specify the signat type (DI/DO/GI/GO), the signal set value, the pulse value and length. See <i>Edit signal control on page 134</i> 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 <i>Edit signal control on page 134</i> for detailed information of the signal control on page 134 for detailed information of the signal control on pag				
	 In the Control Signals (Input) group. You can specify the signat type (DI/DO/GI/GO), the signal set value, the pulse value and length. See <i>Edit signal control on page 134</i> 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 <i>Edit signal control on page 134</i> for detailed information of setting signal and value. To add a joint movement for opening of clamp, select the conresponding joint and move the slider bar to the opened position You can also click the slider bar and input the joint position directly. To edit the time that the joint takes to move to the open position, input a time value (s) in Move Duration number box. 				
	 In the Control Signals (Input) group. You can specify the signat type (DI/DO/GI/GO), the signal set value, the pulse value and length. See <i>Edit signal control on page 134</i> 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 <i>Edit signal control on page 134</i> for detailed information of setting signal and value. To add a joint movement for opening of clamp, select the conresponding 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 				

The editing UI of Clamp is as following:

5.3.25 Models Continued

Properties	Description					
Picking Sensor (Simula- tion)	The picking sensor is mainly for simulation use. It defines an area (usually within the clamp tool) that will intersect with the products when robot moves to the picking position.					
	In simulation, the products that intersect with this area during the closing of the clamp will be picked up by the clamp.					
	Thus users need to set the position and size of this area with					
	appropriate values so that this area will intersect with the products that are "intended" to be picked when the tool is at picking positions.					
	Open Hook Close Hook Picking Sensor (Simulation)					
	Clamp Sensor (SmartComponent)					
	Position (Relative to TCP) (mm)					
	0.000 🗘 0.000 🗘 -210.000 📚					
	Orientation (deg)					
	0.000 © 0.000 ©					
	Size (x,y,z) (mm) 300.000 💲 50.000 🗘 400.000 🚭					
	Pick Mode CenterBottom The gripper will pick the products from product center bottom as					
	During editing, the picking sensor area will also be shown in 3D view as a gray box:					
	xx110000092					

Properties	Description
	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

Claw



xx110000093

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

The editing UI of Claw is as following:

Properties	Description
Open/Close Claw	Opening and closing of claw is similar to that of clamp. See Open/Close Clamp to see how to edit control signals and status signals, and joint movements.
	Open Claw Close Claw Picking Sensor (Simulation) Control Signals (Input) Signal Name Value Type
	giūpenClaw 1 Set
	Status Signals (Output) Signal Name Value Type goClawOpened 1 Set
	goClawOpened 1 Set
	Joint Positions (for simulation) Move Duration 0.20 s
	✓ J1 0.000 45.00 < > J2 0.000 100.00 < >
	xx110000094

5.3.25 Models *Continued*

Properties	Description	
Picking Sensor (Simula- tion)	The picking sensor for claw is the same See Picking Sensor (Simulation) for clar ation.	•
	Open Claw Close Claw Picking Sensor (Simulation)	
	Clamp Sensor (SmartComponent)	
	Position (Relative to TCP) (mm)	
	0.000 0.000 -120.000	*
	Orientation (deg)	
	0.000 😂 0.000 😂 0.000	*
	Size (x,y,z) (mm)	
	50.000 🗢 400.000 🗢 250.000	×
	Pick Mode CenterBottom	*
	The gripper will pick the products from product center bo default (e.g. claw, clamp type)	ottom as

Search



xx110000096

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

The editing UI of Search is as following:

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

5.3.25 Models Continued

Properties	Description
Open/Loosen/Close Searcher	A typical sequence of search process includes: open searcher, loosen search, wait until search sensor touches product and stop signal triggers, close searcher. The steps related to joint movements are described as Open Seacher, Loosen Searcher, and Close Searcher.
	Open Searcher Loosen Searcher Close Searcher Search Sensor (Sii < >
	Open/extrude the search sensor, to prepare for searching
	Control Signals (Input)
	Signal Name Value Type giOpenSearcher 1 Set
	giOpenSearcher 1 Set
	Status Signals (Output)
	Signal Name Value Type •
	doSearcherOpened 1 Set doSearcherClosed 0 Set
	Joint Positions (for simulation)
	Move Duration 0.10 s
	J1 0.000 100.0 < >
	J2 -88.00 0.000 60.000 < >
	J3 -88.00 0.000 60.000 < >
	J4 0.000 100.00 < >
	xx110000098
	The editing each movement is same as Open/Close Clamp. See Open/Close Clamp for detailed information.

5.3.25 Models *Continued*

Properties	Descriptio	n					
Search Sensor (Simulation)) The editing of search sensor is the same as that of Clamp, except that for search, Pick Mode information is not needed, since search is not used for picking.						
	Note that the position of the sensor should be set carefully so						
	that the height of the lowest position of the sensor area is the						
			search TCP pos				
			sor SmartCompo				
	pallet/sheet stack earlier than the TCP, and robot can not cal- culate the height value correctly, thus missing the product						
	picking after	erwards.	freedy, thus hits	sing the pi	ouuor		
			rch Sensor (Simulation)	4 >			
	Position (Relative						
		0.000	-10.000	*			
	Orientation (deg)		¥ 10.000	×			
	0.000	0.000	0.000	×			
	Size (x,y,z) (mm)						
	60.000	\$0.000	20.000	×			
	xx110000099						
	During odit	ing the ecerch	sensor area will	l also he sl	nown in		
		a gray box:					
		a gray box:					
		a gray box:	1046				
		a gray box:	10HB				
		a gray box:	18HB				
		a gray box:	18HB				
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					
		a gray box:					

Sensor



5.3.25 Models Continued

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

The editing UI of Sensor is as following:

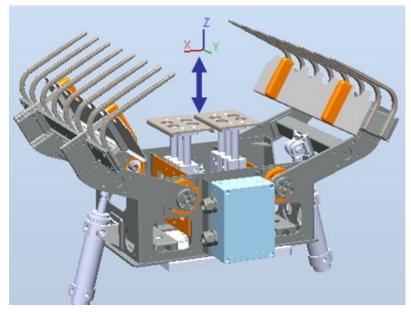
Properties	Description
General	A DI signal can be specified to activate the sensor, and a DO signal should be specified to be triggered when product is detected.
	Activate Sensor (DI)
	Signal diStartPartCheck 🔽 1 📚
	Sensor Out (DO) Signal doPartChecked Sensor (Simulation) Position (Relative to TCP) (mm) 0.000 0.0000 0.000 0.000 0.0000 0.0000 0.000 0.0000 0.0
	0.000 C0.000 C0.000 C0.000 C
	10.000 20.000
	xx110000102

Unit mover



xx110000103

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



xx110000104

Properties	Description
Open/Close Setting	Opening and closing of a Unit Mover is similar to that of clamp. See Open/Close Clamp to see how to edit control signals and status signals, and joint movements.
	Open Setting Close Setting
	Control Signals (Input)
	Signal Name Value Type
	giOpenFixer 1 Set
	Status Signals (Output)
	Signal Name Value Type
	goFixerOpened 1 Set
	Joint Positions (for simulation)
	Move Duration 0.00 s
	□ J1 0.000 < >
	✓ J2 0.000 100.0 < >
	xx110000105

The editing UI of unit mover is as following:

Edit signal control

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

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

🎻 Edit Signal Con	trol X
Signal giOpenClaw	· •
Type DigitalGroup Set or Pulse	plnput 🔻
Set	⊚ Pulse
Set Value	1
Preset Time (s)	0.000
Pulse Reset Value	0
Pulse Length (s)	0.100
0	K Cancel

xx110000106

Properties	Description
Signal	To add a signal control on a new SmartComponent signal, click on the "Signal" list box to change it to input mode, and type the name for the new signal. Then choose the signal type at "Type" list box. The new signal can then be used other signal controls.
	To add a signal control on an already created signal, click on the arrow besides the "Signal" list box, and choose one. In this case you will not be able to change the signal type by the "Type" list box.
Туре	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.

Tool events and scenarios

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

The tool scenarios describe which group of tool functions to use to pick different type of products. For example, the ABB Vacuum tool contains two picking tool functions: vacuum and pallet picker. When the tool is picking boxes, only vacuum is used; and when it is picking pallets, only pallet picker is used. Thus two scenarios can be created, each only including the vacuum function and the pallet picker function respectively. See *Pick setting on page 79* 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 168*). 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.

ol Function Ev	vents Tool Scenario)S			
lit I/O signals	to trigger or wait whe	en a function is used during pic	king, placi	ng, or searching.	
Contained Too	Functions				
6					
1		Image: A start and a start			
Claw1	Fixe	er			
	ce for Claw 1(Picking	and placing events)			
Pick Place	e				
Туре	Signal Name	Controller Signal	Value	Preset Time	Moveme
Set	giOpenClaw	pmGripper1_goToolEvent1	1	0.5	Pick First
Wait	goClawOpened	pmGripper1_giToolEvent1	1	0	Pick First
	giOpenClaw	pmGripper1_goToolEvent1	2	0	Pick
Wait	goClawOpened	pmGripper1_giToolEvent1	2	0	Pick
4		m			Þ
					P.
+	$\boldsymbol{\times}$				
				1000	

xx110000107_

Edit Tool E	vents			∓ ×
Tool Function	Events Tool Scenarios			
searching.	ls to trigger or wait when ool Functions	a function	is used during	picking, placing, or
Claw1	Fixer			
	ence forClaw1(Picking an	id placing e	events)	
Туре	Signal Name	Value	Preset Time	Movement
Set Wait	giOpenClaw goClawOpened	1 1	0.5 0	PickFirstApproach PickFirstApproach
Set Wait	giOpenClaw goClawOpened	2 2	0 0	Pick Pick
•				
				Apply Close

xx110000107

Properties	Description
Contained Tool Functions	List of contained tool functions.
	Each contained tool function should have a sequence of control events.
Event Sequence	The list of events for the selected tool function, in picking and placing
Туре	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.

Properties	Description
Movement	The movement phase (i.e. action) of a pick/place target where this event is used.
	Each pick/place target contains at least 1 Approach point, 1 pick/place point, and 1 Depart point. Depending on the ap- proach/depart setting in operation sets, one pick/place target may contain more than 1 Approach points and 1 Depart points.
	(See <i>Operation set on page 179</i> for detailed information on operation, target, and actions)
	Accordingly, in picking and placing, there are 6 movements defined respectively:
	Pick First Approach ->Pick Last Approach->Pick-> Pick First Depart ->Pick Last Depart
	Place First Approach ->Place Last Approach->Pick-> Place First Depart ->Place Last Depart
	First Approach means the first approach point of a pick (or place) operation.
	Last Approach means the last approach point of a pick (or place) operation. If only 1 approach point was specified originally, an extra point will be added at the same location as the last approach.
	Pick (or Place) means the actual point where pick (or place) happens.
	First Depart means the first depart point of a pick (or place) operation. If only 1 depart point was specified originally, an extra point will be added at the same location as the first depart.
	Last Depart means the last depart point of a pick (or place) operation.
Add, edit, or delete an event	To add/edit/remove an event, click <table-cell-rows> , 📝 , or 🔀 under the event list.</table-cell-rows>

Edit Tool Events				∓ x
Tool Function Events	Tool Scenarios			
Add scenarios to group	o different tool fun	ctions that work together		
ToolScenario1				
Pick/Place Scenario				
ToolScenario1				
Functions for Pickin	g:	Functions for Placing	:	
Pick First A Pick Last A Pick Last A V Pick Pick Last A V Pick First D Pick First D Pick Last D	pproach epart	Place First A Place Last A Place Last A Place Last A Place First D Place First D Place Last D	pproach epart	
Check to include to	ol functions to use	in each movement		
			Apply	Close

xx110000108

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

Create Feeder



The Create Feeder interface is used to create a Palletizing PowerPac 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.

eeder Name	Conveyor_400			
efault Feeder Type	InFeeder			~
Hotspots on this mod	el			
Origin	Hotspot_Le	ft		
Hotspot_Left Hotspot_Right	Name	Hotspot_L	eft	
hotopot_hight	Reference	World		~
	Position (mn	1)		
	0.00	\$ 0.00~	\$ 775.00	\$
	Orientation	(deg)		
	0.00	\$ 0.00	0.00	\$
	Height	775		\$
	WorkObje	ot Alignment (F	Relative to Produ	uct) —
	Rotation	90		~
	Alignment	Right		~
• 🗙	5			

xx110000110

A Palletizing PowerPac 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.

Properties	Description
Default Type	The default type is used when this SmartComponent model is added for a robot, and a feeder is to be created. The default feeder type (i.e., InFeeder, OutFeeder, PalletFeeder, or SheetFeeder) will use the type defined here, but user can still choose to use another type. See <i>Add feeder on page 57</i> for detailed information.
Hotspots	The list of hotspots contained by this feeder.
General	General information: name, position, and orientation of the hotspot frame.

Duanantiaa	Description
Properties	Description
WorkObject/Product align- ment	The group specifies the location of the work object and which quadrant the items are picked from or placed to. Adjust the Alignment and Rotation parameters to select a location suitable for calibration of the work object.
	Alignment: Specifies if the work object origin is located to the right or to the left of the items.
	Rotation: Specifies the orientation of the work object coordinate system.
	When you change the values for Alignment and Rotation, a dummy box and three coordinate arrows will be shown at the hotpot in 3D view, visualizing the relationship between item and work object. Note that the item's front is attached with label.
	AN A
	746032714844 m
	xx110000111



vork object sho

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.

5.3.26 IOPanel

5.3.26 IOPanel

Overview

IOPanel helps to monitor and set the signals used during a palletizing process. This can be used both for VC and RC. It helps to understand the problems during commission and it works as a tool for training classes. It also helps to understand the handshake between the PLC and the controller and also how to control the system using the flow and job signals.

User interface

0 1 15			
elect Controller System15			
nmon Infeeder Outfeeder StackSheets			
Project Control	Gripper Control	Event Messages	Default Height Control
pmProject_diStart 🗸 🥚	pmGripper1_goActivators 👻 0 👘	pmEvent_giMessage	pmProject_diSetDefaultHeigh 👻 🔿
pmProject_diStop	pmGripper1_doToolEvent1	pmEvent_giErrorSource	pmProject_giDefHeightWaSt 👻 0
pmProject_giSelection	pmGripper1_goToolEvent1 2	pmEvent_diTrigger 0	pmProject_giDefaultHeight 👻 0
pmProject_goCurrent	pmGripper1_diToolEvent1		
pmProject_goStatus 🗸 0 🜲	pmGripper1_giToolEvent1 v 2		
	pmGripper1_goSearchActiva 👻 0 👘		
	pmGripper1_diSearchStop 👻 🔘		
Flow Control	System Signals		
Flow Control		nan Svetem do Auto On 🔻	mSvetem diMetersOn
	pmSystem_doCycleOn	pmSystem_doAutoOn 🔹 🍊	pmSystem_diMotorsOn
pmFlow_diStart	pmSystem_doCycleOn pmSystem_doMotorOn	pmSystem_doAutoOn pmSystem_diLoadStat1	pmSystem_diStart
pmFlow_diStart pmFlow_diStop	pmSystem_doCycleOn pmSystem_doMetorOn pmSystem_doRunchOk pmSystem_doRunchOk 	pmSystem_diLoadStart1	pmSystem_diStart pmSystem_diStop
pmFlow_dStart • pmFlow_dStop • pmFlow_gSelection • 0 ÷	pmSystem_doCycleOn pmSystem_doMotorOn		pmSystem_diStart
pmFlow_dStart pmFlow_dStop pmFlow_gSelection pmFlow_gStopOption 1	pmSystem_doCycleOn pmSystem_doMetorOn pmSystem_doRunchOk pmSystem_doRunchOk 	pmSystem_diLoadStart1	pmSystem_diStart pmSystem_diStop
[mRlow_dStat ▼ [mRlow_dStop ▼ [mRlow_gSelection ▼ [mRlow_gStopOpton 1 [mRlow_gStatus 0	pmSystem_doCycleOn pmSystem_doMetorOn pmSystem_doRunchOk pmSystem_doRunchOk 	pmSystem_diLoadStart1	pmSystem_diStart pmSystem_diStop
pmRow_dStat • pmRow_dStop • pmRow_gSelection • pmRow_gStopOption • 1 ÷ pmRow_gStatus • 0 ÷ pmRow_gostutus • 0 ÷	pmSystem_doCycleOn pmSystem_doMetorOn pmSystem_doRunchOk pmSystem_doRunchOk 	pmSystem_diLoadStart1	pmSystem_diStart pmSystem_diStop
pmRow_dStat • pmRow_dStop • pmRow_gSelection • pmRow_gSopOption • pmRow_gSopOption • pmRow_gSopOption • pmRow_gSopOption • pmRow_gSopOption • pmRow_gSopOption • pmRow_goSopOption • pmRow_goSopOption • pmRow_goSopOption • 0 • pmRow_goSopOption • 0 • pmRow_goSopOption •	pmSystem_doCycleOn pmSystem_doMetorOn pmSystem_doRunchOk pmSystem_doRunchOk 	pmSystem_diLoadStart1	pmSystem_diStart pmSystem_diStop
pmRow_dStat • pmRow_dStop • pmRow_gSelection • pmRow_gSopOption • pmRow_gSopOption • pmRow_gSopOption • pmRow_gSopOption • pmRow_gSopOption • pmRow_gostatus • pmRow_gostatus • pmRow_gostatus • pmRow_gostatus • pmRov •	pmSystem_doCycleOn pmSystem_doMetorOn pmSystem_doRunchOk pmSystem_doRunchOk 	pmSystem_diLoadStart1	pmSystem_diStart pmSystem_diStop

In the Advanced group, click IOPanel. The IOPanel window is displayed.

xx1300002143

The IOPanel window has the following four tabs:

- The **Common** tab allows you to monitor the project signals, gripper signals, event message signals, default height signals, flow signals, and system signals.
- The Infeeder tab allows you to monitor the infeeder signals.
- The Outfeeder tab allows you to monitorthe outfeeder signals.
- The Stack Sheets tab allows you to monitor the pallet and stack sheet signals.

By default, the appropriate signals are selected for each each types. If required, the signal can be changed and monitored. Set the signal by clicking on the LED. The LED color changes to red. If it is a group signal then type the numeric value corresponding to the signal.

5.3.27 Change viewpoint

5.3.27 Change viewpoint

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

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

5.3.28 Adjust position of object

5.3.28 Adjust position of object

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

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

5.3.29 Help

5.3.29 Help



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

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

5.4.1 Introduction

5.4 Layout browser

5.4.1 Introduction

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

PalletizingDemo			Ŧ	×
Layout	Programming			
<u> </u>	Layout (Palletiz	zingDemo)		
	S IRB660_1	80_31501		
IRBP_A500_D1450-H700_M2009_F			01	
	🖓 VacuumGr	ipper		
Conveyor_400				
	Outfeeder	1200_800		
		-		

lcon	Node	Description
∑ xx110100119	Station	The station node
x x110100120	Robot	The robot in the station.
ŵ xx110000119	ΤοοΙ	A tool SmartComponent in the station. This may or may not be attached to a robot
	Feeder	A feeder SmartComponent in the station. This may or may not be added into a robot.
✓ xx110100121	Part	A physical object in RobotStudio. Parts with geometric information are made up of one or more 2D or 3D entities. Parts without geometric information are empty.

5.4.2 Station

5.4.2 Station

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

Context Menus from the Station Node

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

5.4.3 Robot

5.4.3 Robot

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

Context Menus from the Robot Node

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

5.4.4 Tool SmartComponent

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

Context Menus from the Tool Node

Context Menu	Description
Edit Tool Functions	Edit tool functions for this tool.See <i>Tool Functions on page 120</i> for detailed description
Edit Tool Events	Edit tool events for this tool.See <i>Tool events and scenarios on page 135</i> for detailed description
Joint Jog	Jog the tool's joint. Only available when a mechanism is used.
Disconnect Library	Disconnect the library
Save As Library	Save the library as rslib file
Attach to / Detach from	If the tool has not been attached to the robot, click Attach to menu and select the right robot, this tool will attach to the se- lected robot. If the tool has been attached to a robot, click De- tach from menu, this tool will detach from the robot.
Examine	Adjust an appropriate viewpoint for the tool.
Rename	Change the tool's name.
Visible	Show or hide the tool.
Delete	Delete the tool from station.

5.4.5 Feeder SmartComponent

5.4.5 Feeder SmartComponent

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

Context Menus from the Feeder Node

Context Menu	Description
Edit Feeder Hotspots	Edit hotspots for this feeder.See <i>Create Feeder on page 139</i> for detailed description
Disconnect Library	Disconnect the library
Save As Library	
Attach to / Attached Status	If the feeder has not been attached to the robot, click Attach to menu and select the right robot, this feeder will attach to the selected robot. If the tool has been attached to a robot, which robot is attached to will display in this context menu.
Set Position	Set feeder's position.
Examine	Adjust an appropriate viewpoint for the feeder.
Rename	Change the feeder's name.
Visible	Show or hide the feeder.
Delete	Delete the feeder from station.

5.4.6 Part

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

Context Menus from the Part Node

Context Menu	Description
Build into a Tool	Build this part as a tool Smart Component, which can be used by the robot. See <i>Create Tool on page 120</i> for detailed descrip- tion
Build into a Feeder	Build this part as a feeder Smart Component, which can be used by the robot. See <i>Create Feeder on page 139</i> for detailed description.
Set Position	Set part's position.
Examine	Adjust an appropriate viewpoint for the part.
Rename	Change the part's name.
Visible	Show or hide the part.
Delete	Delete the part from station.

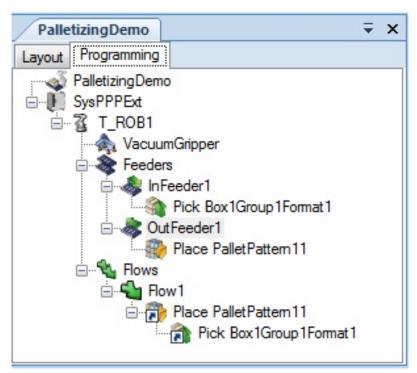
5.5.1 Introduction

5.5 Programming browser

5.5.1 Introduction

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.



lcon	Node	Description
∛ xx110000122	Project	The project node
III xx110000123	Controller	A controller that exists in the station
置 xx110000124	Robot	A robot (motion task) that exists in the parent controller
♠ xx110000125	ΤοοΙ	The tool used by the parent robot. It references a tool SmartComponent.
≫ xx110000126	Feeders	The collection of feeders for the parent robot
* xx110000127	Feeder	A feeder that is used by the robot. It references a feeder SmartComponent.

5.5.1 Introduction Continued

lcon	Node	Description
	Operation Set	An Operation Set program that picks(places) from(to) the parent feeder. Depending on the type of item it handles (box, bag, pallet, or sheet), and whether it is picking or placing, the icon is different.
% xx110000129	Flows	The collection of flows for the parent robot
u xx110000130	Flow	A flow that exists for the parent robot
	Job (Master Operation Set)	A supported job for the parent flow. It actually references to an Operation Set of the flow's master feeder.
	Slave Operation Set	Reference to an Operation Set on the slave feeders that are related to the master job.

5.5.2 Project

5.5.2 Project

Context Menus from the Project Node

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

Context Menu	Description
Settings	Edit settings for the project. The following section gives you a more detailed description on Project Settings.
Rename	Change the Project's name.

Project Settings

Introduction

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

🏐 Settings		x	
Settings Settings Settings Settings General Restart Options Tune Limitation Name PalletizingDemo Description This is a demo description for project.			
	ОКСа	ancel	

xx110000133

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

5.5.2 Project Continued

Restart Options

🐐 Settings	x
Settings General Restart Options Tune Limitation Select which restart options should be visible on the FlexPendant when recovering after an error. To hide a restart option on the FlexPendant, clear the option's check box. Continue Pick-Place Restart Layer Next Pallet	
OK Cance	<u>ا</u> ا

xx110000134

Select which restart options should be visible on the FlexPendant when recovering after an error. All options are selected as default. To hide a restart option on the FlexPendant, clear the option's check box. See *Flow on page 202*.

Continue Pick-Place is always available.

5.5.2 Project *Continued*

Tune Limitation

ect0	Settings					
zPP_RC1	General Restart Option	ns Tune Limitat	tion			
T_ROB1	Reset Tune Value	Apply Tune	Values]		
	Feeder Workobject					
	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	x,y,z (mm)				gle Z (deg)
	Lower Tune Limit	0.00	0.00	0.00	▲ 0	<u>*</u>
	Upper Tune Limit	x,y,z (mm)				gle Z (deg)
	Upper Tune Limit	0.00	0.00	0.00	÷ 0	* *
	Item					
			Lower T	une Limit	Uppe	r Tune Limit
	Speed (%)		100	-	10	0 🚖
	Rot Speed (%)		100	*	10	0 ≑
	Acc/Dec (%)		100	-	10	0
	Pick Time (%)		100	*	10	
	Place Time (%)		100		10	
	Vacuum Activation Tim	ne (s)	0	-	0	* *
	Vacuum Deactivation	Time (s)	0	-	0	*
	Size Z (%)		100	*	10	0

xx110000135

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 235*.

Feeder WorkObject Group

Click Reset Tune Value to reset all the tuned values.

Click Apply Tune Value to assign the tuned values as the default values.

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 180*.
- 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.

5.5.2 Project Continued

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

The following table describes the parameters of the Feeder WorkObject group

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 66*.
- 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:

Item	Description
Speed	Specifies maximum speed for an item.
Rot Speed	Specifies maximum orientation speed for an item.
Acc/Dec	Specifies maximum acceleration/deceleration for an item.
Pick Time	Specifies the time that the robot stays at the target position when picking an item.
Place Time	Specifies the time that the robot stays at the target position when placing an item.
Vacuum Activation Time	Specifies the time for vacuum activation.
Vacuum Deactivation Time	Specifies the time for vacuum deactivation.
Size (z)	Specifies the height of an item.

5.5.3.1 Controller

5.5.3 Controller

5.5.3.1 Controller

Context Menus from the Controller Node

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

Context Menu	Description
Settings	Edit settings for the controller.The following section gives you a more detailed description on Controller Settings.

Controller Settings

Introduction

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

🗳 Settings		×
PalletizingDemo ⊖. SysPPPEd T_ROB1	Settings Name SysPPPExt General Events Feeder Order Messages Image: Pulse Feeder DO Signals (old style) Pulse Length: (50 ~ 2000 ms) (old style) 500 Image: ms	
		OK Cancel

5.5.3.1 Controller Continued

Item	Description
Pulse Feeder DO Signals	If selected, the Position request trigger signals and the Opera- tion set complete signals will be pulsed instead of set by the controller. It is not recommended to use "Pulse Feeder DO Signals" since a pulse may be missed out to be detected by the PLC.
Pulse Length	Here you specify the general pulse length that will be used, if the controller pulses an output signal . The pulse length can be set in the range of 50 ms to 2000 ms.

Events

🏐 Settings			X
PalletizingDemo	Event Signals Trigger (DI): Error Source (GI): Messages (GI): Source Settings Specifies which bi source.	eeder Order Messages pmEvent_diTrigger pmEvent_giErrorSource pmEvent_giMessage t of the Error Source (GI) signa s can be set when triggering an	
	Source SysPPPExt T_ROB1 InFeeder1/T_ROE OutFeeder1/T_RO		Bit 1 2 3 4 Add
			OK Cancel

xx110000137

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

Event Signals Group

Item	Description
Trigger (DI)	Specifies which digital input signal will be used to trigger an event. When this signal goes high, the event will be reported to the PickMaster process. This signal must be used together with the Error Source (GI) and/or Messages (GI) signal.

5.5.3.1 Controller Continued

Item	Description
Error Source (GI)	A group input signal representing the source of an error. If the signal is defined it indicates where the error has occurred (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 <i>Source Settings Group</i> .
Messages (GI)	A group input signal representing an event message. If the signal is defined it specifies which message will appear on the FlexPendant when the Trigger (DI) signal is set. You define the messages in the line view of PickMaster, see <i>Messages on page 162</i> .

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 246*.

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

If the source points out a feeder, then the feeder will be set in error state when the Trigger (DI) peaks. The current state of a feeder is indicated by the FlexPendant interface, see *Viewing flow status on page 242*. It is also indicated by the feeder I/O signal Execution state (GO), see *Feeder on page 170*.

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 278.

5.5.3.1 Controller Continued

Feeder order

⊟ ·· PalletizingDemo ⊟ ·· <mark>SysPPPExt</mark> 	Settings Name SysPPPExt			
	General Events Feeder Ord	· · · · · · · · · · · · · · · · · · ·		
	Define the physical order of the in which order the robot will p through its working range		en making long moveme	ents
	Feeder	Order	Robot	
	PM_HOME InFeeder1 OutFeeder1	-1 -1 -1	T_ROB1 T_ROB1 T_ROB1	

xx110000138

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

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

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

161

5.5.3.1 Controller *Continued*

Messages

🗳 Settings	
PalletizingDemo	Settings Name SysPPPExt General Events Feeder Order Messages Messages Max Title Length: 0/60 Max Message Length (Title & Description): 0/195
	Category Value Title Description Description:
	Import Export Selected Export All

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

5.5.3.1 Controller Continued

Item	Description
	Click the Message text group and type the description for the selected message.

5.5.3.2 Robot

5.5.3.2 Robot

Context Menus from the Robot Node

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

Context Menu	Description
Attach Tool	If there is no tool attached to the robot, this context menu will display. It is to add a tool to the robot.
	See Add tool on page 55 for detailed description.
Settings	Edit settings for the robot. The following section gives you a more detailed description on Robot Settings.
Jump Home	Jump the robot to the home position.
Set Position	Set robot's position.
Examine	Adjust an appropriate viewpoint for the robot.
Visible	Show or hide the robot.

5.5.3.2 Robot Continued

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.

a Settings	1 m m	×
	Settings Controller PzPP_660_6.02 Robot T_ROB1 Motion Motion Limits (when robot is not holding products) Speed (mm/s)	Set to Full Speed
	Image: Constraint of the second start RAPID Image: Constraint of the second start result of the second start res and start result of the second start result of the	Add Edit Export Browse
		OK Cancel

Item	Description
Controller	The name of the controller which the robot belong to.
Robot	The name of the robot.
Speed	The speed of the robot.
Acceleration/Deceleration	The speed can be set to full or low with the right button. The acceleration/deceleration speed of the robot.
Rotation Speed	The rotation speed of the robot.
Load and Start RAPID	If you select this check box, you can press the start button on the FlexPendant and the RAPID module will be loaded to the robot controller. The check box is selected by default.
Activate PickWare Log	If you select this check box, the log files for system debug purpose are generated in the Home folder of the robot system.

5.5.3.2 Robot *Continued*

Item	Description
Add	To add a RAPID module to the project: 1 Click Add. 2 The Load RAPID dialog box appears, and you can select which RAPID module to add.
	3 In the Load RAPID dialog box, click OK.
Delete	To remove a RAPID module from the project, select a module and click .
Edit	To edit a RAPID module, select a module and click Edit.
Export	To save a module with a new name, select a module and click Export.
Browse	Select software to edit the RAPID. In default, the software is notepad.

5.5.4 Tool

Context Menus from the Tool Node

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

Context Menu	Description
Edit Tool Signals	Edit signals for this tool. See <i>Edit Tool Signals on page 168</i> Error! Reference source not found. for detailed description
Edit Tool Functions	Edit tool functions for this tool. See <i>Tool Functions on page 120</i> for detailed description
Edit Tool Events	Edit tool events for this tool. See <i>Tool events and scenarios on page 135</i> for detailed descrip- tion
Joint Jog	Jog the tool's joint.
Detach from	If the tool has been attached to a robot, click Detach from menu, this tool will detach from the robot. And this node will be deleted either.
Examine	Adjust an appropriate viewpoint for the tool.
Rename	Change the tool's name.
Visible	Show or hide the tool.

5.5.4 Tool *Continued*

Edit Tool Signals

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

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

🐳 Edit I/O Signals for Tool		
Tool I/O Connection for T_ROB1 Choose controller I/O to connect with tool I/O ClampOneZoneGripper Connection Save As Library		
Tool I/O	Controller I/O	
Tool I/O Controller I/O @ giOpenCl pmGripper1_goToolEvent1 @ goClamp pmGripper1_giToolEvent1 @ giOpenHo pmGripper1_goToolEvent2 @ goHookO pmGripper1_giToolEvent2		
OK Cancel		

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

5.5.5.1 Context Menus from the Feeders Node

5.5.5 Feeders

5.5.5.1 Context Menus from the Feeders Node

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

Context Menu	Description
	It is to add a feeder into the station.See <i>Add feeder on page 57</i> for detailed description.

5.5.5.2 Feeder

5.5.5.2 Feeder

Context Menus from the Feeder Node

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

Context Menu	Description
Edit	Edit general settings for this feeder. See <i>Edit feeder on page 171</i> for detailed description
Edit Feeder Hotspots	Edit hotspots for this feeder. See <i>Create Feeder on page 139</i> for detailed description
Add Operation Set - Group	Add a group operation set. See <i>Operation set on page 179</i> for detailed description. Available when the feeder does not have any operation set or already has group operation sets.
Add Operation Set – Pat- tern/Stack	Add a pattern or stack operation set. See <i>Operation set on</i> page 179 for detailed description Available when the feeder does not have any operation set or already has pattern/stack operation sets.
Safe Targets	Define the safe positions that the robot will pass before entering and after leaving the selected feeder. You can define several safe positions to let the robot follow a path when entering and leaving the feeder. Please note the order of the selected safe positions! The last safe position in the list is the last that the robot will pass before reaching the feeder and the first that the robot will pass when leaving the feeder. The last safe position in the list will also define the reference arm and wrist configur- ation of the robot when operating the feeder. See <u>Safe targets on page 175</u> for detailed description
Robot Path Height	Define some height attributes that are considered when the robot is moving to, from or over the feeder. The Robot path height settings affect the height of intermediate movements, which means the settings affect the output of the RAPID instruc- tion PmGetPathHeight that is used by MoveInterMid in the PmUtility module. See <i>Robot path height on page 176</i> for de- tailed description
Set WorkObject Position	Set WorkObject position for the feeder. See Set WorkObject position on page 177 for detailed description
Examine	Adjust an appropriate viewpoint for the feeder.
Rename	Change the feeder's name.
Visible	Show or hide the feeder.
Delete	Delete the feeder from station.

5.5.5.2 Feeder Continued

Edit feeder

There are two pages to edit general settings for feeder.

Edit Feeder: InFeeder1	∓ x		
Name	Туре		
InFeeder1	InFeeder 💌		
Work Object to Attach	Signal Index		
pmInfeeder1_WObj 🔹	1		
Tuned Offset (0, 0, 0) mm, (0) deg			
Feeder Model Feeder I/O			
Model (SmartComponent)			
Conveyor_400 -			
Hotspot to Attach Workobject			
Hotspot_Right	•		
Stop Job Timeout (ms) 5000	×		
	ly Close		

xx110000142

Item	Description
Name	The name of the feeder.
Туре	 The feeder type and the following types are available: Infeeder Outfeeder PalletFeeder SheetFeeder
Work Object to Attach	Specify to which work object this feeder is connected. To change work object, select a work object from the drop- down box. Ten predefined work objects are listed in the work object drop- down box. The work object can be selected also in offline mode.
Signal Index	Specify which default signals that shall be suggested for the feeder when selecting Use Default I/O checkbox. It is combined with the feeder type.
Stop Job Timeout	Define the maximum time a pending position request will be active after a stop job has been started. It is used only by the slave feeders. Should be set to a time that is longer than the time it takes to generate new targets on the feeder. For ex- ample, longer than the maximum product infeed time.

Continues on next page

5.5.5.2 Feeder *Continued*

Item	Description
Model (SmartComponent)	Select which model (SmartComponent) is the source of the feeder.
Hotspot to Attach Workobject	Select which hotspot is attached with the work object.

Edit Feeder: OutFeeder1	X	
Name	Туре	
OutFeeder1	OutFeeder 👻	
Work Object to Attach	Signal Index	
pmOutfeeder1_WObj	1	
Tuned Offset (0, 0, 0) mm, (0) deg	
Feeder Model Feeder I/O		
✓ Use Default I/O	Edit Signals	
Simulated		
Feeder Used as Master		
🔲 Feeder Used as Slave		
Feeder Used neither as Ma	aster nor as Slave	
Stop Job Timeout (ms)	5000 🚖	
Арр	ly Close	

Item	Description
Use Default I/O	See <i>Edit detailed signals for feeder on page 173</i> for detailed description.
Edit Signals…	Edit detailed signals for the feeder.Click this button, a dialog will popup. In this dialog, the signal configuration can be adjusted.See <i>Edit detailed signals for feeder on page 173</i> for detailed description.
Simulated	See <i>Edit detailed signals for feeder on page 173</i> for a detailed description.
Feeder Used as Master	See <i>Edit detailed signals for feeder on page 173</i> for a detailed description.
Feeder Used as Slave	See <i>Edit detailed signals for feeder on page 173</i> for a detailed description.

5.5.5.2 Feeder Continued

Item	Description
Feeder Used neither as Master nor as Slave	When the feeder is not used in any job this check box is selected by default.
Signal Status	Provides the match information about signals connection between the feeder and the virtual controller. If the signal connection is not valid, the status light is red, otherwise it is green.

Edit detailed signals for feeder

Use Default I/O	Target Generation	Status
Feeder Used as Master	Simulated	Position Available (DO)
Feeder Used as Slave	Target Generation Trigger (DI)	pmOutfeeder1 doPosAvail
Feeder Used neither as Master nor as Slave	pmOutfeeder1_diTgtGenTrig 👻	Operation Set Complete (DO)
_	Product Selection (GI)	pmOutfeeder1 doOpSetCompl
Position Request	pmOutfeeder1_giProdSel 🔹	
Position Request Trigger (DO)	Format Selection (GI)	Queue Empty (DO)
pmOutfeeder1_doPosReqTrig 🗸		pmOutfeeder1_doQueueEmpty
Product Selection (GO)		Execution State (GO)
-	Start Layer Count (GI)	pmOutfeeder1_goExecState
Format Selection (GO)	pmOutfeeder1_giStartLayerCi 👻	Height State (GO)
Tolma: Selection (CO)	Start Product Count (GI)	pmOutfeeder1 goHeightState
	pmOutfeeder1_giStartProdCo 👻	
Requesting Master (GO)		Layer Count (GO)
•		pmOutfeeder1_goLayerCount
Robot Control		Product Count (GO)
Stop Job (DI)		pmOutfeeder1_goProductCount
pmOutfeeder1_diStopJob		
Redo Search (DI)		
Robot Execution (DI)		
•		

xx110000144

Item	Description
Use Default I/O	When selected, PickMaster will, depending on the selection of Master or Slave, suggest default signals in Position request, Target generation, and Status. The names of the default signals will be based on the selection in feeder type and default signal index. Clear the checkbox if you want to edit the signal selection manually. See Default signals.
Feeder Used as Master	Specify this feeder is used as master. Filter that simplifies setting up a master feeder. When selected, only signals and parameters that can be used by a master feeder are enabled for updates.
Feeder Used as Slave	Specify this feeder is used as slave. Filter that simplifies setting up a slave feeder. When selected, only signals and parameters that can be used by a slave feeder are enabled for updates.
Feeder Used neither as Master nor as Slave	Specify this feeder is used neither as master nor as slave. When selected, signals and parameters are not enabled.

Operating manual - Palletizing PowerPac 3HAC042340-001 Revision: R

Continues on next page

5.5.5.2 Feeder Continued

Item	Description
Position Request Group	Define the signals used by the PickMaster process when re- questing new operation sets. For a detailed description of the signals, see <i>Basic I/O interface on page 262</i> .
Target Generation Group	Define the signals used by a PLC to indicate that new targets are available for picking or placing. For a detailed description of the signals, see <i>Basic I/O interface on page 262</i> .
Simulated	When the check box is selected, the configured target genera- tion signals are not used on the robot controller. Instead, Target generation will be directly cross connected to the output of the Position request signals.
	Use simulation on all slave feeders to test run a palletizing project without having implemented the communication with a PLC.
	Palletizing jobs can be started from the FlexPendant or the I/O interface. If only one job, that is, operation set, has been defined for a master feeder, simulation can be used to auto start the job in a continuous mode when the flow has been started.
Status Group	Define different signals to monitor the status of the PickMaster process. For a detailed description of the signals, see <i>Basic I/O interface on page 262</i> .
Robot Control Group	Define different signals to affect the PickMaster process. For a detailed description of the signals, see <i>Basic I/O interface on page 262</i> .

\mathbf{r} 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 179 Configuration for more information.

5.5.5.2 Feeder Continued

Safe targets

Edit Safe Targe	ets: InFeeder1		₹ :
Name	Move Ty	Zone	
SafePos_1	MoveJ	z20	
SafePos_2	MoveJ	z20	
•			P
×			•
Safe Target			
Name			
Ivame	SafePos_1		
Configuration	0,0,0,0		• 🕙
Location Exter	nal Axes		
Reference Wor	kObject		
pmInfeeder1_V	VObj		
Position (mm)			
-252.24~	-174.32~	뉮 339.0 <mark>4</mark> 1	- 🔺
Orientation (de	g)		
-180.00	0.00~		
Read Robot P	os tSearcher		
	_ • •		~
	Apply		Close

xx110000145

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

Continues on next page

5.5.5.2 Feeder *Continued*

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

Robot path height

The default robot path 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 height or Height of latest finished operation set**. For an infeeder, the default height is normally set to **Full height**, **Empty height**, or **Height of latest finished operation set** depending on how the products are fed into the working range of the robot.

Edit Path Height: In	Feeder1	x
Safety Height Type)	4
Height of latest	finished operation set	
Empty height	300.00	(mm)
Full height	300.00	(mm)
Auto generate f	full height	
Safety Offset (mm)	0.00	(mm)
	Apply	Close

5.5.5.2 Feeder Continued

Item	Description	
Height of latest finished operation set	Height of latest finished operation set is the final height of the latest run operation set. The default height can be tempor arily updated in runtime, for example, set to Empty after unload ing 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.	
	Note	
	It is possible to update the default height in runtime to save cycle time without decreasing the margins for collisions, espe- cially if the project consists of many feeders and flows.	
Empty height	Define the empty height of the feeder in the work object frame (mm).	
Full height	Defines the full height of the feeder in the work object frame (mm). Select the Auto generate full height check box to generate the value as the maximum value between height of the configured operation sets and the safe targets defined (if any).	
Safety Offset	An offset that is always added to the expected height of the feeder.	

Set WorkObject position

Edit Feeder Position: InFeeder1 = ×	
Work Object pmInfeeder1_WObj	
Work Object Position	
Reference World	
Position (mm)	
1000.00 🔄 1000.00 🔄 775.00 🚔	
Orientation (deg)	
0.00 0.00 0.00	
3 Apply Close	

То	Do this
Change the Reference Co- ordinate System	Select the Reference coordinate system you want to use to position the WorkObject used for the feeder.
Edit WorkObject Position	Click in one of Position boxes, and then click the position in the graphics window to transfer the values to the Position boxes.Or type a new value in the Position boxes to specify the position of the WorkObject.

5.5.5.2 Feeder *Continued*

То	Do this
Edit WorkObject Orienta- tion	Type a new value in the Orientation boxes to specify the orient- ation of the WorkObject. Or click and to adjust the orientation for the WorkObject.
Check Reach for the WorkObject	Click 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.

5.5.5.3 Operation set

5.5.5.3 Operation set

Overview

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 product groups are to be picked or placed.

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 170* Configuration.

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

Context Menus from the operation Set

Context Menu	Description
Edit	Open the editing dialog for the operation set.See <i>Group oper-</i> <i>ation set on page 180</i> and <i>Pattern/Stack operation set on</i> <i>page 183</i> for detailed information
Locate Pick Setting	Locate the referenced Pick Setting. The Pick Setting interface will be opened and the referenced pick setting will be selected. Available for group and stack operation set.
Locate Pallet Pattern	Locate the referenced pattern. The pattern interface will be opened and the referenced pallet pattern will be selected.Avail- able for pattern operation set.
Locate Stack	Locate the referenced item stack. The Product/Pallet/Sheet interface will be opened and the referenced item stack will be selected.Available for stack operation set.
Preview Palletizing	Preview each pick (place) target in the operation set.See <i>Preview Palletizing on page 90</i> for detailed description.
Rename	Change the operation set's name.
Delete	Delete the Operation Set from the parent feeder.

5.5.5.3 Operation set *Continued*

Group operation set

Edit: Pick Box1Group1Fo	rmat1 🗧 🛪
Pick Box1Group1Format1	
Pick Setting Selection	
Product	Group
Box1 -	Box1Group1 -
Pick Setting	
Box1Group1Format1 (T_RO	B1) 🔻 🕕
Validation	
Preview Palletizing	Check Reach 🥥
Settings	
General Approach/Depart	Search
Pick or Place	
Pick	Place
Preparation Time (Simulatio	n)
Infeeding time	1.0 s
Displacement	
Offset (mm)	
0.00	0.00
	Apply Close

Item	Description
Pick Setting Selection	Select the item, item group and the pick setting for that item group to use. Click 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 op- eration set. See <i>Preview Palletizing on page 90</i> 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.

Item	Description
Approach/Depart	Define the robot path to follow when picking or placing the iten group. It enables configuration of Approach and Depart movements which are necessary for specific grippers like Jav Gripper. It will be useful where the gripper has to make hori- zontal movements to avoid collisions to neighboring equip- ment/components inside the cell. It is basically an improvemen to the movement of robots as well as the pick and place direc tions. You can pick and place the items (Bag,Box,Pallet,Shee and so on) based on the selection of Approach and Depart. 1 Select the Add Horizontal Approach and Add Horizonta Depart, then add the approach and depart value.
	Edit: Pick Box1Group1Format1
	Pick Box1Group1Format1
	Pick Setting Selection
	Product Group
	Box1 Box1Group1
	Pick Setting
	Box1Group1Format1 (T_ROB1)
	Validation
	Preview Palletizing Check Reach
	Settings General Approach/Depart Search
	General Approach Depart Search
	General Offset 100
	V End Approach in z Only
	V Start Depart in z Only 100
	Add Horizontal Approach mm
	Horizontal Approach mm
	Front v 200 mm mm
	Add Horizontal Depart
	Horizontal Depart
	Back 💌 200 🚖 mm
	Use Concurrency
	Use Am Config Monitoring
	Use SingArea SingArea Wrist 👻
	Apply Cose
	xx110000155
	XX110000100

5.5.5.3 Operation set *Continued*

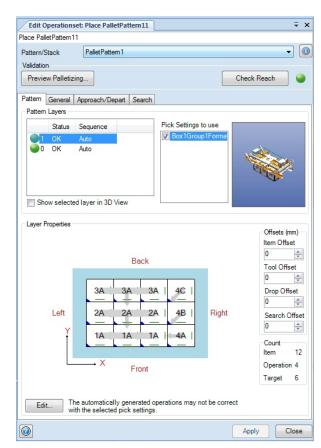
Item	Description				
Search		earch only if the operation set is Pick . h, select the Activate Stack Search			
	General Approach/Depart Search				
	Activate Stack Search				
	C Settings				
	Speed (mm/s)				
	100.00				
	Offset (mm) Stop H	leight (mm)			
	0.00	\$			
	Search Tool Function				
	Search1	~			
	xx110000156 For more information, s	see Stack search on page 191.			

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

Item	Description	
Speed	Defines the TCP speed of the robot when it searches for the format.	
Offset	Defines an offset of the distance between the expected position of the format, and the starting position of the robot's search movement. The total search offset will also include the product height of the format.	
Stop height	Defines the TCP height above the work object where the robot will stop the search movement (if search stop never occurs).	
Search Tool Function	Defines which search tool function to use for searching.See <i>Tool Functions on page 120</i> on description on search tool function	

Pattern/Stack operation set

Introduction



xx110000157

Item	Description
Pattern/Stack	Set the name of the Operation Set.
	Select the pallet pattern or item stack to use.
	Click to see item count, weight, total height, and the re- lated product selection signal from the online PLC when the targets have been generated.
Validation	Use Preview Palletizing function to view each target of the op- eration set. See <i>Preview Palletizing on page 90</i> for detailed information.
	Use Check Reach to quickly check reach-ability of all the tar- gets, which will be indicated by the status light beside the check button and before each layer. Also see <i>Check reach on page 88</i> for detailed information.

183

5.5.5.3 Operation set *Continued*

ltem	Description				
Pattern tab	This section lists and shows information about all layers in the pallet pattern and the pick settings to be used for this pallet pattern.				
	In the Pick Settings to use list box, check the ones that you want to include in the operation set. It is important to select them in such a way that it is possible to complete each layer.				
	Note				
	When a pick setting for an item group with more than one column is used, the algorithm to calculate the operations and target positions may be wrong. Warning information will be shown in 2D layout view and user should manually check the sequence and use Preview Palletizing to validate.				
	To edit layer offsets, select the layer and edit the valuesTo edit the operations for a layer, double-click on the layer, or click on the layer and then click Edit.Manually edited layers operations will be specifically marked as "Manual", otherwise, they will be marked as "Auto". See <i>Operation editor on page 196</i> for more information.The display shows how the items are grouped to- gether in different operations. For more information, see Display information.				
General tab	Pattern General Approach/Depart Search				
	I/O Value 0				
	Pick O Place				
	Preparation Time (Simulation) Clear time 0.1 📚 s				
	Start Corner				
	O X+ Y-				
	Offset (mm)				
	0.000 🗢 0.000 🗢 Z Rot (deg)				
	0.00 Flip Drag in 3D View				
	xx110000158				
	I/O Value: Set the I/O value of the Operation Set. Also see I/C <i>interface on page 107</i> for detailed information.				
	Pick or Place : Indicates how does the robot access the item group				
	Preparation Time (Simulation): If picking, it describes the time it takes for the patt ern/stack group to show up on the in-feeder before robot goes to pick; if placing, it describes the time it takes for the pattern/stack to be moved away on the out-feeder after robot has placed it. A robot can only continue to work on the feeder after the previous pattern/stack is moved away.				
	Start Corner: Defines the corner of the pallet pattern where the robot should start.				
	Displacement: Describes the displacement between the item group and the feeder's work object. To rotate the frame 90 degrees, click the Flip button, and offsets and angle will be updated to keep chosen alignment. Or you can also click "Drag in 3D View" and use Freehand Move to move the item CAD models directly.				

Item	Description				
Approach/Depart tab	Define the robot path to follow when picking or placing the item group. It enables configuration of Approach and Depart movements				
Search	To enable the stack search, select the Activate check box. For information about the configuration for the stack search, see Stack search on page	ion parameters			
	To configure the Place Search operation select the Place option in the General tab. Once you select the place option, the Place Search configuration options are enabled on the Search tab. For information about the configuration parameters for the place search, see <i>Place search on page 192</i> . Note				
	Place search is used only for Boxes and Bags.				
	Edit: Place PalletPattern 11	∓ ×			
	Place PalletPattern11 Pattern/Stack PalletPattern1	~ 0			
	Validation Preview Palletizing Check F	Reach			
	Pattern General Approach/Depart Search				
	Activate Stack Search				
	Settings Speed (mm/s) Frequency				
	100.00				
	Offset (mm) Stop Height (mm) 200.00 ↓ 0.00 ↓				
	Search Tool Function				
	Search1 V				
	Place Search				
	Post-Place Pre-Place				
	Search start layer				
	3 ~				
	Search on item (within layer)				
	First Item/group V				
	Apply	Close			
	xx110000160				

Display information

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

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.



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

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

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

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



Item offset is se

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

Edit: Place PalletPattern11			1	
Place PalletPattem11				
Pattem/Stack PalletPa	attem1			- 0
Validation		_		_
Preview Palletizing			Check Reach	
Pattern General Approach/	Depart Search			
General Offset	100	*	mm	
End Approach in z Only	100	×	mm	
Start Depart in z Only	100	A V	mm	
Add Horizontal Approach				
Front 🔻	50 🚖 mm			
Add Horizontal Depart Horizontal Depart Right Use Concurrency	50 💌 mm			
Use Arm Config Monitoring	1			
Use SingArea SingArea Off	, 			
			Apply	Close

xx110000159

Item	Description
General offset	Defines the general pick and place approach/depart height. It also defines the movement safety distance to keep between robot held items and their nearby items in a pallet pattern.
	See the robot movement tables on the following pages for a detailed approach and depart movement description when picking and placing items.
End approach in z only	Select this check box when you want the robot to move straight down just before it reaches the target. In the Offset box, type the extra approach distance.
Start depart in z only	Select this check box when you want the robot to move straight up when it leaves the target. In the Offset box, type the extra depart distance.

5.5.5.3 Operation set *Continued*

Item	Description		
Add Horizontal Approach	Select this check box when you want to add an additional ap- proach point before the pick target, so that the tool will move to the pick target from side instead of above. In Direction selec- tion box, select the horizontal direction; and in the Offset box, type the horizontal offset.		
Add Horizontal Depart	To move the gripper horizontally away from the object of in- terest with the specified horizontal depart value. Directions of movements can also be configured.		
Use Concurrency	When the check box is selected, the \Conc instruction is added to the RAPID move instruction. It allows the subsequent instructions to be executed while the robot is moving to avoid unwanted stop.		
Use Arm Config Monitoring	Select this check box to enable the Act . ArmConfMon Boolean variable.		
Use SingArea	You can select one of the following 3 options for SingArea. SingArea Off - The tool orientation is not allowed to differ. 		
	 Sing Area Wrist- The tool orientation is allowed to differ somewhat to avoid wrist singularity. 		
	• SingArea LockAx4 - The orientation is such that the 4th axis is locked, that is, 6 axis robot behaves like a 4 axis robot.		
	By default the SingArea is set to SingArea Off.		
	Note		
	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 following table 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.

Туре	Direction	End ap- proach	Start depart in Z only	Position count in Z only	Movement
any	none	no	no	3	
					$\left(\begin{array}{c} 1 \\ 2 \end{array} \right)^{3} $ offs
					xx110000161

5.5.5.3 Operation set Continued

Туре	Direction	End ap- proach	Start depart in Z only	Position count in Z only	Movement
pick	any	no	no	3	1 2 0 offs xx110000162
place	any	no	no	3	1 offs xx110000163
pick	any	no	yes	4	offs dept 2 offs xx110000164
place	any	yes	no	4	offs appr { 2 } fs offs xx110000165
pick	any	yes	yes	5	offs appr 3 offs dept xx110000166

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.

End approach in Z	Start depart in Z only	Special case	Position count	Movement
no	no	no *	5	1 2 3 offs offs xx110000167
no	no	last place	4	item offs offs item offs affs + tool offs offs xx110000168
no	no	first pick	4	offs + tool offs { 1 } 1 } 1 + tem offs offs offs offs offs offs offs off
yes	no	no *	6	h + item offs offs xx110000170
no	yes	no *	6	h+ item offs offs 1 1 1 1 1 5 1 1 5 1 1 1 1 1 1 1 1 1 1 1 1 1
yes	yes	no *	7	h+ item offs offs appr + + item offs offs dept txx110000172

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



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 135*, *Advanced setting for Item on page 66*, and the *Pick setting on page 79* 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 173*.

Stack search

Item	Description
Activate stack search	Select the Activate check box.
	When stack search is activated, you can enter values in the text boxes.
Speed	Defines the TCP speed of the robot when it searches for the next layer.
	To set speed, in the Speed text box, type the value the robot should use when searching.
Offset	Defines an offset of the distance between the next layer and the starting position of the robot's search movement. The total search offset will also include the product height of the next layer and the current layer's layer search offset (defined in the Layers section).Total search offset = offset + product height + layer search offset.
	To set offset, in the Offset text box, type the offset value from the assumed to most layer where the robot should start the search procedure.
Stop height	Defines the TCP height above the work object where the robot will stop the search movement (if search stop never occurs).
	To set stop height, in the Stop height text box, type the offset value from the work area where the robot should stop searching.

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

5.5.5.3 Operation set *Continued*

Item	Description
Frequency	Defines the frequency to use stack search. The value 1 means every layer, 2 means every second layer, 3 means every third layer, and so on.
	Set the frequency, in the Frequency box, type or select the layer frequency for which the stack search will be done. 1 means every layer, 2 means every second layer, and so on.
Search Tool Function	Defines which search tool function to use for searching. See on <i>Tool Functions on page 120</i> for details.

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

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

Place search

The following table describes the configuration parameters for Place Search.

Parameters	Description
Post Place	Executes the search operation after making a placement.
Pre-Place	Executes the search operation before making a placement.
Search start layer	Allows you to select from which layer the Place Search opera- tion should start. The frequency of search applies from the selected of search start layer.
Search on item (within layer)	Allows you to select whether First item/placement from the layout or the last item/placement from the layout should be used for the search operation.

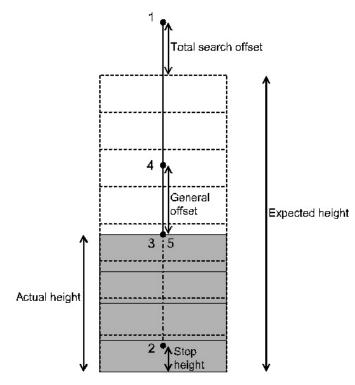
Following are the different scenarios arise during Place search:

Search on item/when to search	Post-Place	Pre-Place
First Item/placement	Post-place search is performed on the first item and the result is used for the remaining items of the current layer.	before the first item is placed
Last item/placement	Post-place search is performed on the last item of the current layer and the result is used in the next layer.	Pre-place search is performed on the last item of the current layer and this result is not use- ful since the layer is already completed.

Search tool	
	With stack search activated, a special search tool is activated and a search movement is started before the first item in the first layer is picked. The search movement starts an offset from the expected height of the pallet pattern/format and continues until a sensor input indicates that the top layer is reached. If there is no indication from the sensor input, the search movement continues until a stop height is reached. At search stop, the height of the pallet pattern (or format) is updated with the current height of the search tool. Then, movement is started to the approach position of the next pick position. The search tool is deactivated before the movement to the pick position is started.
Pallet pattern	
	If a pallet pattern is used, picking will continue with lower layers until the pallet is empty. It is also possible to configure new searches for lower layers to improve the picking accuracy. Further it is possible to order a new search from the top of the pallet pattern at any time before the next layer is picked by using an input signal. See Redo search signal in the illustration, in section <i>Edit detailed signals</i> <i>for feeder on page 173</i> .
Configuration of stac	k 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 sear	rch tool
	You do the configuration of the search tool function in the <i>Search tool on page 193</i> window.

Stack search target sequence, example

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



xx110000173

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

Check reach

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

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

In current version, intermediate positions are NOT checked.

Result	Description
Status light	The status light will be Gray, Green, and Red, indicating reach status of unknown, all reachable, and containing unreachable targets. Each layer also has a status light indicating its containing tar- gets.
	Validation Preview Palletizing Check Reach
	Pattern General Approach/Depart Search Pattern Layers Status Sequence 4 OK Auto 3 OK Auto 2 OK Auto 1 OK Auto 1 OK Auto 2 Show selected layer in 3D View
3D graphical display	In 3D view, if a layer contains unreachable positions, the related items with these positions are drawn with red color.
	xx110000175
Graphic display in Layer, layout display	When selecting a layer containing unreachable positions from the layers list, all items containing unreachable positions will be drawn in red Back
	Left IA 2A 3A 4A IA 2A 3A 4A IA 2A 3A 4A Front

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

5.5.5.3 Operation set *Continued*

Operation editor

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

ieneral Inform								Operations		
Top TwoBlock24 (Box2)(None)			New	🔇 🔇	Remove Last					
Туре:	Place 41				Box2Group1For	mat2 (T_ROB1)				
tems:					Fill	Remove All				
Operations: Targets:	ations: 41 ets: 41			Retain Man	ual Changes					
									3D View	
			E	Bac	k			_		
	-	384	18	-	-	-	9			
	*		-	30	-	30	30			
	-		-	38	-	-	238	D: 11		
Left	-	-	3	*	-	30	345	Right		
Y	21	"	7	3	7	-	-			
ť	14	3	A		tta e	ADA	253.			
						h di		•		
								oach dir ▶		
					Eve		ера	t dir ≯		
irection		-					-			
Approac	ch									
Horizont	tal Ap	pro	ach							
Horizont	tal De	epar	t							
										OK Cancel

xx110000177_

Note

User can select multiple operations/targets, and execute the functions available in context menu, except **Tool Event**.

A layer normally requires several operations, each using a specific format. The order of the operations is important since this is the order the robot will use to access the formats. Click on **Edit Layer** button in layer properties to edit the operations. The **Edit Layer** tab is displayed.

То	Do this
Remove an operation	Click Remove to remove the last existing operation.
	Note
	You can remove only the last operation, and new operations can only be added last. This means that to change operations, you may have to remove existing operations first.

Continues on next page

То	Do this
Remove all operations	Click Remove All to remove all existing operations.
Add an operation	Click New to add a new operation. The selected format to use for the operation is shown in the Display area.
Select format	Select the format to use for the new operation in the Format list.
Define format location	Move and rotate the format using the mouse. To define that an item in the format should be placed at a specific location in the layer, make the areas overlap and double-click on the intersec- tion. To define if an item should be placed separately or togeth- er with its neighbor, right-click and select Group item or Separ- ate item.
Auto generate operations	Click Fill to auto generate the operations for the rest of the layer.
Show the tool	Select this check box to display the location of the tool while dragging a format in the display area.
Approach direction	To change the general approach direction of a placed target, right-click and select Approach or select the target and use the mouse wheel to change the direction. The horizontal depart direction is indicated by grey arrows.
Horizontal Approach direc- tion	This is enabled only when the Add Horizontal Approach check box is selected in the Approach/Depart tab, and the direction is set to User Defined. If the check box is selected and the appropriate direction is chosen in this tab, then all the targets follow this direction. If the check box is not selected, the option is disabled. To change the horizontal approach direction of an individual target, right-click on the target and select Horizontal Approach direction and choose the appropriate direction. The horizontal approach direction is indicated by red arrows. By default red arrow will not be added to the operations. User has to edit each operation by right clicking on it and selecting the required Horizontal Approach direction.
	Angular Approach direction is not shown for operations for which Horizontal Approach direction is defined.

5.5.5.3 Operation set *Continued*

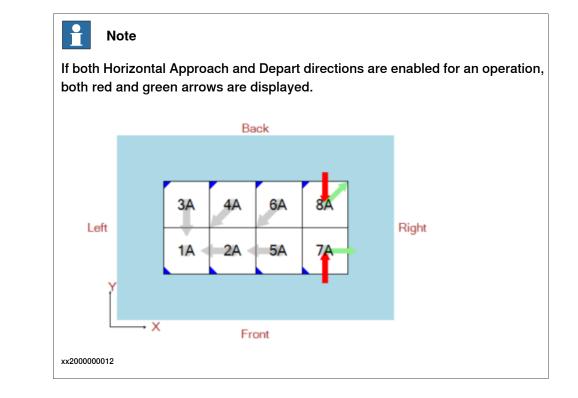
То	Do this
Horizontal Depart direction	
Tool Events	To set Tool Events for each item, right-click on the item and select Tool Events . The events are stored for each item. If there is no change to the Tool Events at the item level, then the Tool Events defined at the Tool level shall be used for the corresponding item.
Retain Manual Changes	Select this check box to retain the manual changes made to the pallet pattern even if you change the layout from the pallet pattern editor.



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

Note

Layers with edited operations is not 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.



5.5.6.1 Flows

5.5.6 Flows

5.5.6.1 Flows

Context Menus from the Flows

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

Context Menu	Description
Add Flow	It is to add a new flow into the robot.
	See Add Flow on page 200 for detailed description.

Add Flow

General	
Name	1/0 Value
Flow1	0
Description	
Input your description here	
Priority] Auto Start
1 🗘 🔽] Early Request
PalletFeeder1	~
Feeders Master Feeder	
PalletFeeder1	~
Slave Feeders	
 ✓ OutFeeder1 ✓ InFeeder1 ✓ SheetFeeder1 	

5.5.6.1 Flows Continued

Item	Description
Name	The name of the flow as it will be shown.
	The name of the flow will also appear on the PickMaster Flex- Pendant 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 star- ted/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.
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 inter- rupted 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.

5.5.6.2 Flow

5.5.6.2 Flow

Context Menus from the Flow

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

Context Menu	Description
Edit	Edit the selected flow configuration.See <i>Add Flow on page 200</i> for detailed description.
Add Job	Starts the Add Job wizard.See <i>Add job on page 83</i> for detailed description.
Locate Master Feeder	Locate the master feeder of this flow. The feeder node will be selected in the Programming browser.
Rename	Change the flow's name.
Delete	Delete the flow from the robot.

5.5.7 Job

5.5.7 Job

Context Menus from the Job

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

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

This page is intentionally left blank

6.1 Introduction

6 RAPID program

6.1 Introduction

Structure of this chapter

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

6 RAPID program

6.2.1 Relationship between RAPID execution and PickMaster project

6.2 Overview

6.2.1 Relationship between RAPID execution and PickMaster project

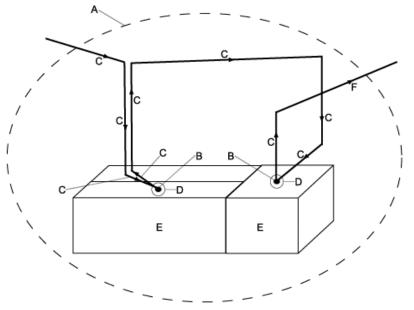
Overview	
	The RAPID program templates for a robot executes a pick-and-place cycle for one of the configured flows in every loop. The selection of flow is made in priority order among the flows which currently are ready to execute, that is, having targets generated for the next operation on both the infeeder and the outfeeder. The robot movements and the I/O events on the infeeder and outfeeder are performed according to the configured operation sets and formats.
Intermediate pos	sitions
	The RAPID program templates inludes functionality to generate safe intermediate positions for the movements between infeeders, outfeeders and the home position.
Operation, targe	t, action, event and product
Introduction	
	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 354.
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 368.
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 339.
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:



xx0600003002

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

6 RAPID program

6.2.2 RAPID modules overview

6.2.2 RAPID modules overview

Overview	
	To run a PickMaster project you need RAPID program modules and system modules, which are described in this section.
RAPID template mo	odules
	 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 <i>Palletizing PowerPac</i> , 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 <i>Prepared for PickMaster</i> , together with the sub-option
	Palletizing PowerPac, 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 procedure mainly used to set all modal data used in the moves.
	 pmrcBase, which is a system module that contains encrypted non-view procedures and variables used in the process.

6.3.1 PmMain module

6.3 Program module templates

6.3.1 PmMain module

Overview	This postion departies the relations and variables in the Departure we dolly. The
	This section describes the routines and variables in the <i>PmMain</i> 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	
	The routine is re-executed for every new pick and place cycle. The error handler is used to recover an error when running a pick and place cycle including a stack search on the master work area.
	To recover PM_ERR_JOB_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 on the master work area after an empty stack is detected.
	Generate a new job on the master work area.
	 4 Recover the error in the error handler using RETRY. As a result the robot wil start the new job and a new pick and place cycle is started.
Program code	
Ū	PROC Main()
	IF FirstMainLoop THEN
	MoveHomePos;
	FirstMainLoop:=FALSE;
	ENDIF PmWaitProjStart;
	OperateSequence;
	ERROR
	IF ERRNO = PM_ERR_JOB_EMPTY THEN
	RETRY;
	ENDIF
	ENDPROC
Related information	
	Main routine in Technical reference manual - RAPID kernel

Main routine in Technical reference manual - RAPID kernel.

6.3.1 PmMain module *Continued*

Procedure OperateSequence on page 210. PmWaitProjStart - Wait for any active project on page 318. PmGetWaByWobj - Get a reference to a work area using a work object data on page 302.

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 error handler is used to handle some errors when running an operation including a stack search.

Error code	Description
PM_ERR_PALLET_REDUCED	The detected stack height was lower than expected. The error is recovered by fetching next operation using RETRY.
PM_ERR_PALLET_EMPTY	The detected stack height was zero on a slave work area.
PM_ERR_JOB_EMPTY	The detected stack height was zero on a master work area. The error is raised and recovered in the Main routine.

To recover **PM_ERR_PALLET_EMPTY**:

- 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 position request.
- 4 Set variable MultiOperation to TRUE to avoid an intermediate movement before starting a new search.
- 5 Recover the error in the error handler using RETRY. As a result the robot will search the new stack from the top.

Program code ProC OperateSequence() PmGetFlow waInFeeder,waOutFeeder; Operate waInFeeder; Operate waOutFeeder; ERROR TEST ERRNO CASE PM_ERR_PALLET_REDUCED: ! Number of remaining layers on pallet was updated after stack search. ! Operate the same work area again to access the new current layer. MultiOperation:=TRUE; RETRY; CASE PM_ERR_PALLET_EMPTY:

6 RAPID program

6.3.1 PmMain module Continued

```
! The pallet stack was found empty during stack search.
MultiOperation:=TRUE;
RETRY;
CASE PM_ERR_JOB_EMPTY:
! The pallet stack was found empty during stack search on master.
MultiOperation:=FALSE;
RAISE;
ENDTEST
ENDPROC
```

Related information

Procedure Operate on page 211.

Procedure Operat	e
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	
U	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
	<pre>PmCalcArmConf Act.RobTgt,Tgt.TargetTool,Tgt.TargetWobj</pre>
	IF FirstTgtInOp AND (NOT MultiOperation)THEN
	MoveInterMid WorkArea, Tgt, Act, PmSafetyOffsetZ
	\MaxAngle:=MaxToolAngle\MinAngle:=MinToolAngle;
	ENDIF
	DoAction WorkArea,Tqt,Act;
	SetLastPos WorkArea,Tgt,Act;
	ENDWHILE

Continues on next page

6 RAPID program

6.3.1 PmMain module *Continued*

```
ENDWHILE
MultiOperation:=FALSE;
ERROR
TEST ERRNO
CASE PM_ERR_PALLET_REDUCED:
RAISE;
CASE PM_ERR_PALLET_EMPTY:
RAISE;
ENDTEST
ENDPROC
```

Related information

Procedure PmDoAction on page 230Procedure SetLastPos on page 215.Procedure MoveInterMid on page 216.pm_wadescr - PickMaster work area reference on page 372.pm_operationdata - PickMaster operation data on page 354.pm_targetdata - PickMaster target data on page 368.pm_actiondata - PickMaster action data on page 339.PmGetOperation - Get operation from a work area on page 298.PmCalcArmConf - Calculates the arm configuration on page 290.PmGetTarget - Get target on page 330.PmGetTgtAction - Get target action on page 332.PmSearchAdjust - Adjust number of remaining layers on page 305

Variables waInFeeder1 and waOutFeeder1

Usage	
5	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;
	<pre>PmGetFlow waInFeeder, waOutFeeder;</pre>
	Operate waInFeeder1;
	Operate waOutFeeder1;
Related information	
	pm_wadescr - PickMaster work area reference on page 372.
Constants MaxTool	Angle and MinToolAngle
Usage	
	The constants are used to set the maximum and minimum allowed angles for the

tool.

6.3.1 PmMain module Continued

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	<pre>PmCalcArmConf Tgt.RobTgtPoint, Tgt.TargetTool, Tgt.TargetWobj \cf6 \MaxAngle:=MaxToolAngle \MinAngle:=MinToolAngle \SingAreaType:=Act.SingAreaType;</pre>
Related information	pm_wadescr - PickMaster work area reference on page 372.
	PmCalcArmConf - Calculates the arm configuration on page 290.

6.3.2 PmUtility module

6.3.2 PmUtility module

Overview		
	This section describes the routines and variables in the <i>PmUtility</i> module. The	
	module contains support for the home and intermediate position.	
Procedure Move	HomePos	
Usage		
	This procedure is used to move the robot to the home position.	
Description		
	This routine uses MoveInterMid to move the robot to a well-defined home position with a safe path height when passing intermediate work areas on the way. The routine will wait for the project to be started before the movement is executed. The default installed work area PM_HOME is used as the work area to go to. Finally, the home position is set as the last work area, thus making it the starting point for the next intermediate movement which also will get a safe path height.	
Program code		
-	PROC MoveHomePos()	
	CONST num RetractDist:=50;	
	VAR pm_targetdata Tgt;	
	VAR pm_actiondata Act;	
	VAR pm_wadescr HomeWorkArea;	
	! Get the weight from current tool and frame from tool0	
	<pre>PmLastTool:=CTool();</pre>	
	<pre>PmLastTool.tframe:=tool0.tframe;</pre>	
	PmLastWobj:=CWObj();	
	<pre>PmLastRobTgt:=CRobT(\Tool:=PmLastTool\Wobj:=PmLastWobj);</pre>	
	Act.Speed:=v500; Act.Accel.AccLim:=FALSE;	
	Act.Accel.AccMax:=100;	
	Act.Accel.DecelLim:=FALSE;	
	Act.Accel.DecelMax:=100;	
	Act.Accel.Acc:=100;	
	Act.Accel.Ramp:=100;	
	Act.RobTgt:=CalcRobT(HomePos,tool0);	
	Tqt.TargetTool:=PmLastTool;	
	Tgt.TargetWobj:=pm_home_WObj;	
	<pre>PmLastRobTgt.trans.z:=PmLastRobTgt.trans.z+RetractDist;</pre>	
	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;	
	<pre>PmLastTool:=Tgt.TargetTool;</pre>	
	PmSetLastWa HomeWorkArea;	

6.3.2 PmUtility module Continued

Related information	
	PmWaitProjStart - Wait for any active project on page 318.
	PmGetWaByWobj - Get a reference to a work area using a work object data on
	page 302.
	PmSetLastWa - Set last used work area on page 307.
Procedure SetLast	Pos
Usage	
-	This procedure is used to store the last position, tool, work object, and work area.
Description	
	The stored position, tool, work object, and work area are used when calculating the intermediate position.
Arguments	
	WorkArea
	Datatype: pm_wadescr
	Last work area that was used.
	Tgt
	Datatype:pm_targetdata
	Last target data that was used.
	Act
	Datatype: pm_actiondata
	Last action data that was used.
	Last action data that was used.
Program code	
	PROC SetLastPos(VAR pm_wadescr WorkArea, VAR pm_targetdata Tgt, VAR pm_actiondata Act)
	VAR pm_actionata Act; VAR robtarget temp;
	temp:=LastRobTgt;
	LastRobTgt:=Act.RobTgt;
	LastWobj:=Tgt.TargetWobj;
	LastTool:=Tgt.TargetTool;
	PmSetLastWa WorkArea;
	IF Act.ArmConfMon = FALSE THEN
	<pre>LastRobTgt.robconf:=temp.robconf; ENDIF</pre>
	ENDIF
Related information	pm_actiondata - PickMaster action data on page 339.
	pm_targetdata - PickMaster target data on page 368.
	pm_wadescr - PickMaster work area reference on page 372.
	PmSetLastWa - Set last used work area on page 307.

ENDPROC

6 RAPID program

6.3.2 PmUtility module *Continued*

Procedure MoveInterMid		
Usage		
	This procedure is used to move the robot from a starting point (for example another work area or the home position) towards a new operation on the next work area with a safe path height when passing intermediate work areas on the way from the starting point.	
Description		
	This procedure uses the last stored position (from the SetLastPos procedure) and a new operation on the next work area to calculate three consecutive intermediate positions by using the PmCalcIntermid routine. PmGetPathHeight is used to find the minimum height for a safe travel towards the work area.	
Arguments		
	WorkArea	
	Datatype: pm_wadescr	
	Work area to go to.	
	Tgt	
	Datatype:pm_targetdata	
	The next target to go to.	
	Act	
	Datatype:pm_actiondata	
	The next action to perform.	
	SafetyOffsetZ	
	Datatype: num	
	An additional safety offset that is added to the minimum path height.	
	MaxAngle	
	Datatype: num	
	The maximum allowed tool angle.	
	MinAngle	
	Datatype: num	
	The minimum allowed tool angle.	
	MoveToEndPoint	
	Datatype: switch	
	Finish with zone or fine point.	
Program code		
	PROC MoveInterMid(VAR pm_wadescr WorkArea, VAR pm_targetdata Tgt, VAR pm_actiondata Act,num SafetyOffsetZ,\num MaxAngle,\num MinAngle,\switch MoveToEndPoint)	
	CONST num IntermidPart1:=0.1;	
	CONST num IntermidPart2:=0.5; CONST num IntermidPart3:=0.9;	
	VAR robtarget InterMid1;	

6.3.2 PmUtility module Continued

```
VAR robtarget InterMid2;
VAR robtarget InterMid3;
VAR jointtarget FromJointTgt;
VAR jointtarget ToJointTgt;
VAR robtarget FromRobTgt;
VAR robtarget ToRobTgt;
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.GripLenEmptyZ+SafetyOffsetZ;
ELSE
! MinZ with product in tool
MinZ:=PmGetPathHeight(LastWorkArea,WorkArea\UseSafePosition)+
     Tgt.GripLenLoadedZ+SafetyOffsetZ;
ENDIF
! Check z value also for the start and end target.
! Use tool and workobject from target tool.
TempTool:=Tgt.TargetTool;
TempWobj:=Tgt.TargetWobj;
! Set start and end target values.
FromJointTgt:=CalcJointT(PmLastRobTgt,PmLastTool\Wobj:=PmLastWobj);
ToJointTgt:=CalcJointT(Act.RobTgt,TempTool\Wobj:=TempWobj);
FromRobTgt:=CalcRobT(FromJointTgt,tool0);
ToRobTgt:=CalcRobT(ToJointTgt,tool0);
! Compare z value for start and end targets, set a new higher value
     for z if needed.
IF (FromRobTgt.trans.z<ToRobTgt.trans.z) AND</pre>
     (MinZ<ToRobTqt.trans.z)THEN
MinZ:=ToRobTgt.trans.z;
ELSEIF (ToRobTgt.trans.z<FromRobTgt.trans.z) AND
     (MinZ<FromRobTgt.trans.z) THEN
MinZ:=FromRobTgt.trans.z;
ENDIF
ConfJ\On;
! Use the frame from tool0 and the load from target tool
TempTool:=Tgt.TargetTool;
TempTool.tframe:=tool0.tframe;
! Set Acceleration
PathAccLim Act.Accel.AccLim\AccMax:=Act.Accel.AccMax,
     Act.Accel.DecelLim\DecelMax:=Act.Accel.DecelMax;
AccSet Act.Accel.Acc, Act.Accel.Ramp;
```

6.3.2 PmUtility module *Continued*

```
! Using PmDoMove3 instead of MoveJ.
! PmDoMove3 will automatically avoid using too many consecutive
     concurrent (\Conc) movements.
! Travel distance: 10%
InterMid1:=PmCalcIntermid(PmLastRobTgt,PmLastTool,PmLastWobj,
     Act.RobTgt,Tgt.TargetTool,Tgt.TargetWobj,IntermidPart1\
     MaxAngle?MaxAngle\MinAngle?MinAngle\AngleLimAx6\
     MinZ:=MinZ\FromWa:=LastWorkArea\ToWa:=WorkArea);
PmDoMove3
     PM_MOVE_JOINT\Conc, InterMid1, Act.Speed, z200, TempTool, wobj0;
! Calculate intermediate targets two and three
InterMid2:=PmCalcIntermid(PmLastRobTgt,PmLastTool,
     PmLastWobj,Act.RobTgt,Tgt.TargetTool,Tgt.TargetWobj,
     IntermidPart2\MaxAngle?MaxAngle\MinAngle?MinAngle\
     AngleLimAx6\MinZ:=MinZ\FromWa:=LastWorkArea\ToWa:=WorkArea);
InterMid3:=PmCalcIntermid(PmLastRobTgt,PmLastTool,
     PmLastWobj,Act.RobTgt,Tgt.TargetTool,Tgt.TargetWobj,
     IntermidPart3\MaxAngle?MaxAngle\MinAngle?MinAngle\
     AngleLimAx6\MinZ:=MinZ\FromWa:=LastWorkArea\ToWa:=WorkArea);
! Travel distance: 50%
PmDoMove3
     PM_MOVE_JOINT\Conc,InterMid2,Act.Speed,z200,TempTool,wobj0;
! Travel distance: 90%
IF Present(MoveToEndPoint) THEN
PmDoMove3
     PM_MOVE_JOINT\Conc,Act.RobTgt,Act.Speed,fine,TempTool,wobj0;
ELSE
PmDoMove3
     PM_MOVE_JOINT\Conc,InterMid3,Act.Speed,z200,TempTool,wobj0;
ENDIF
ENDPROC
```

Related information

pm_actiondata - PickMaster action data on page 339.
pm_targetdata - PickMaster target data on page 368.
pm_wadescr - PickMaster work area reference on page 372.
PmGetLastWa - Get last used work area on page 297.
PmGetPathHeight - Get a safe path height for an intermediate movement on page 325.
PmCalcArmConf - Calculates the arm configuration on page 290.

Variable HomePos

Usage

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

Continues on next page	
218	

6.3.2 PmUtility module Continued

Description	
	The home position must be modified for custom purposes.
Program code	
	LOCAL PERS jointtarget HomePos:=[[0,0,0,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
	MoveAbsJ HomePos,HomeSpeed,fine,TempTool;
Related information	
	Technical reference manual - RAPID Instructions, Functions and Data types, section robtarget - Position data.

6.3.3 PmProjMgr module

6.3.3 PmProjMgr module

Overview	
	The PmProjMgr module can be used if needed. It is prepared to be used in PickMaster I/O interface for starting a project and loading the modules needed for the main palletizing loop. The I/O signals used in this module are the same as in the configuration that comes with the installation of PickMaster. This module is compatible with the modules PmMain and PmUtility.
	This section describes:
	Procedure Main
	• Trap TrapProjectStopped
Procedure Main	This section describes the main routine in the <i>PmProjMgr</i> module.
Usage	
J.	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	
	<pre>PROC main() VAR pm_projectinfo ProjInfo; IF FirstProjMgrLoop THEN FirstProjMgrLoop:=FALSE;</pre>
Continuos on nort	<pre>StartLoadRun:=TRUE;</pre>
Continues on next	paye

6.3.3 PmProjMgr module Continued

```
IF PM_PROJECT_STATUS=PM_PROJECT_STOPPED OR
     PM_PROJECT_STATUS=PM_PROJECT_STOPPING OR
     PM_PROJECT_STATUS=PM_PROJECT_ERROR THEN
 IF PM_PROJECT_STATUS=PM_PROJECT_STOPPING THEN
    WaitUntil PM_PROJECT_STATUS=PM_PROJECT_STOPPED OR
         PM_PROJECT_STATUS=PM_PROJECT_ERROR;
ENDIF
! Wait for start project order from PLC
WaitDI pmProject_diStart,1;
! Check which project to be started
ProjectSelection:=pmProject_giSelection;
! Get info from select project
PmGetProjectInfo ProjectSelection,ProjInfo;
ProjectInfo:=ProjInfo;
! Start the selected project
IF StartLoadRun THEN
 WaitTestAndSet ProjectStart;
 IF PM_PROJECT_STATUS<>PM_PROJECT_RUNNING THEN
 PmStartProj ProjectInfo.Name\Signal:=pmProject_goStatus;
 ENDIF
 ProjectStart:=FALSE;
 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;
 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
```

6.3.3 PmProjMgr module *Continued*

StartLoadRun:=FALSE; ENDIF ENDPROC

Related information

Main routine in *Technical reference manual - RAPID kernel*. Late binding in *Technical reference manual - RAPID kernel*. *PmWaitProjStart - Wait for any active project on page 318*. *PmStartProj - Start a PickMaster project on page 314*.

Trap TrapProjectStopped

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

Usage

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

Description

TrapProjectStoped executes the following:

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

Program code

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

Related information

Technical reference manual - RAPID kernel.

6.3.4 PmProjServer module

6.3.4 PmProjServer module

Overview	
	The <i>PmProjServer</i> 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 restart of the controller if the signals are not found.
Procedure Main	This section describes the main routine in the <i>PmProjServer</i> 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	
	PROC main()
	VAR bool SignalsExist:=TRUE;
	! Connect all alias signals with the configured signals
	SignalsExist:=ConnectAliasSignals();
	WHILE SignalsExist=FALSE DO
	! Loop forever
	WaitTime 1000;
	ENDWHILE WHILE TRUE DO
	! Wait for project to be running.
	PmWaitProjStart;
	! Connect all traps with its interrupts
	ConnectTraps;
	! Wait for stop project order
	WaitUntil PM_PROJECT_STATUS=PM_PROJECT_STOPPED;
	<pre>SetGO alias_goCurrentProject, 0;</pre>
	! Disconnect all traps from its interrupts
	DeleteTraps;
	ENDWHILE
	ENDPROC

6.3.4 PmProjServer module *Continued*

Trap TrapSetRecoverAction		
• -	This section describes the trap that executes when the set recover action signal	
	has been pulsed.	
Description		
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 Argl;	
	VAR errstr Arg2;	
	VAR errstr Arg3;	
	VAR errstr Arg4;	
	VAR bool UseMasterWa:=FALSE;	
	<pre>FlowSelection:=alias_giSelectionFlow;</pre>	
	RecoverAction:=alias_giRecoverAction;	
	WaSelection:=alias_giWaRecoverSelection;	
	! Get info from selected flow	
	<pre>PmGetFlowInfo FlowSelection,FlowInfo;</pre>	
	IF RecoverAction=PM_RECOVER_REDO_LAST_PICK THEN	
	! Set recover action	
	<pre>PmSetRecoverAction FlowInfo.Name,RecoverAction\EventId:=EvtId</pre>	
	ELSE	
	! Get info from selected Work Area	
	PmGetWaInfo WaSelection,WaInfo;	
	IF UseMasterWa = TRUE THEN	
	! Set recover action - If no workarea, use the master workarea	
	for the flow	
	<pre>PmSetRecoverAction FlowInfo.Name\Workarea:=FlowInfo.MasterWa,</pre>	
	RecoverAction\EventId:=EvtId	
	\Argument1:=Arg1\Argument2:=Arg2\Argument3:=Arg3\Argument4:=Arg4;	
	UseMasterWa:=FALSE;	
	ELSE	
	! Set recover action	
	PmSetRecoverAction	
	<pre>FlowInfo.Name\Workarea:=WaInfo.Workarea,RecoverAction</pre>	
	\EventId:=EvtId\Argument1:=Arg1\Argument2:=Arg2	
	\Argument3:=Arg3\Argument4:=Arg4;	
	ENDIF	
	ENDIF	

6.3.4 PmProjServer module Continued

```
IF (EvtId<2398) AND (EvtId>2392) THEN
   PmErrorLog EvtId,FlowInfo.Name,Arg1,Arg2,Arg3,Arg4
         \EventType:=TYPE_WARN;
 ENDIF
ERROR
 ! Continue supervision on recoverable errors
 IF ERRNO=PM_ERR_FLOW_NOT_FOUND THEN
   IF RecoverAction = PM_RECOVER_CONTINUE_OPERATION THEN
     UseMasterWa:=TRUE;
     TRYNEXT;
   ELSE
     RETURN;
   ENDIF
 ELSEIF ERRNO=PM_ERR_WA_NOT_FOUND THEN
   RETURN;
 ELSEIF ERRNO=PM_ERR_NO_RUNNING_PROJECT THEN
   RETURN;
 ELSEIF ERRNO=PM_ERR_REDO_LAST_PICK_REJECTED THEN
   RETURN;
 ELSEIF ERRNO=PM_ERR_WORKAREA_EXPECTED THEN
   RETURN;
 ELSEIF ERRNO=PM_ERR_NOT_VALID_RECOVER_ACTION THEN
   RETURN;
 ELSE
   RETURN;
 ENDIF
ENDTRAP
```

Trap TrapStartFlow

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

Description

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

Program code

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

225

6.3.4 PmProjServer module *Continued*

	FICTTE FORMA-DM FOR MOANC FIAM CTATE THEN
	ELSEIF ERRNO=PM_ERR_WRONG_FLOW_STATE THEN
	RETURN;
	ELSE
	RETURN;
	ENDIF
	ENDTRAP
Trap TrapStopE	low
	This section describes the trap that executes when the stop flow signal has beer
	pulsed.
Description	
Description	A CL signal defines which flow should be standed and another CL signal defines
	A GI signal defines which flow should be stopped and another GI signal defines
	the stop behavior. The flow name received from <i>PmGetFlowInfo</i> is used to stop
	the flow.
Program code	
U	TRAP TrapStopFlow
	VAR pm_flowinfo FlowInfo;
	VAR num FlowSelection;
	VAR num StopOption;
	<pre>FlowSelection:=alias_giSelectionFlow;</pre>
	<pre>StopOption:=alias_giStopOptionFlow;</pre>
	! Get info from selected flow
	<pre>PmGetFlowInfo FlowSelection,FlowInfo;</pre>
	! Stop the selected flow
	PmStopFlow FlowInfo.Name,StopOption;
	ERROR
	! Continue supervision on recoverable errors
	IF ERRNO=PM_ERR_FLOW_NOT_FOUND THEN
	RETURN;
	ELSEIF ERRNO=PM_ERR_NO_RUNNING_PROJECT THEN
	RETURN;
	ELSE
	RETURN;
	ENDIF
	ENDTRAP

Trap TrapProjectStopped

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

Description

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

Program code

TRAP TrapProjectStopped
 PmStopProj;
ENDTRAP

Continues	on	next	page
226			

6.3.4 PmProjServer module *Continued*

Related information

PmGetFlowInfo - Get information about a specific flow on page 295.
PmStartFlow - Starts a specific flow on page 312.
PmStopFlow - Stop a specific flow on page 315.
PmStopProj - Stop current project on page 317.

6.4.1 System modules, overview

6.4 System modules

6.4.1 System modules, overview

Description

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

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

6.4.2 Public system module pmrcUser

6.4.2 Public system module pmrcUser

Description	
	The <i>pmrcUser</i> 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 TrapDIToc	lEvents
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. <i>WarningTime</i> 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;
	<pre>PmCheckToolEventInputSignals warningTime, pollRate;</pre>
	ENDTRAP

Trap TrapGIToolEvents

Usage

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

6.4.3 Public system module pmrcSys

6.4.3 Public system module pmrcSys

Description		
	base functionality. This m code is open and editable	ains instructions and data that are a part of the PickMaster nodule is declared as NOSTEPIN, which means that the but it is not possible to step into the routines. The n be removed for debug purposes.
		in a backup. If modifications are needed in this module, ons and move them to <i>pmrcUser</i> .
Procedure PmDoAc	tion	
Usage		
	Execute an action, that is events and motion setting	, a single robot movement having preconfigured path gs.
Basic examples		
		rocedure PmDoAction is seen in the program code for
	the <i>Procedure</i> Operate C	on page 211, in the PmMain module.
Arguments		
	WorkArea	
	Datatype: pm_wadescr	
	Contains a reference to the	ne work area to use.
	Tgt	
	Datatype: pm_targetdat	ca
	The target data used for t	he move.
	Act	
	Datatype: pm_actiondat	ta
	The action data used for t	the move.
Error handling		
_	-	e errors can be generated. The errors can be handled in system variable ERRNO will be set to:
	Error code	Des
	PM_ERR_PALLET_REDUCED	The detected stack height was lower than expected.
	PM_ERR_PALLET_EMPTY	The detected stack height was zero on a slave work area.
	PM_ERR_JOB_EMPTY	The detected height was zero on a master work area.
Syntax		·
Cymax	PmDoAction	
		ression (IN) of pm_wadescr > ','
		pression (IN) of pm_targetdata > ',' pression (IN) of pm_actiondata > ';'
	[ACL `•=`] < exp	TESSION (IN) OF DWTactiondarg > ''

6.4.3 Public system module pmrcSys Continued

Program code	
	See the program code by browsing the pmrcSys module on the robot.
Procedure PmDoMo	ove3
Usage	
	This procedure calls the selected move Instruction (MoveL, MoveJ, TriggL or TriggJ) using the selected parameters.
Basic examples	
	A basic example of the procedure ${\tt PmDoMove3}$ is seen in the program code for the
	<i>Procedure MoveInterMid on page 216</i> , in the PmUtility module.
Arguments	
	Move
	Datatype: pm_movetype
	The type of movement that is used. Supported types are MoveJ, MoveL, TriggL and TriggJ.
	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.

Continues on next page

6.4.3 Public system module pmrcSys *Continued*

	Τ6
	Datatype: triggdata
	Variable that refers to trigger conditions and trigger activity.
	Τ7
	Datatype: triggdata
	Variable that refers to trigger conditions and trigger activity.
	Т8
	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.
	ΤοοΙ
	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	
Flogram code	See the program code by browsing the $pmrcSys$ module on the robot.
Variable pm_home	•_Wobj
Usage	
	This variable is used for the default installed work area <code>PM_HOME</code> 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 LastRo	bTgt, LastWobj and LastTool
Usage	
osuge	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	
232	Operating manual - Palletizing PowerPac
	3HAC042340-001 Revision: R © Copyright 2012-2020 ABB_All rights reserved

6.4.3 Public system module pmrcSys *Continued*

Program code

See usage of the variables in the program code by browsing the ${\tt pmUtility}$ module.

This page is intentionally left blank

7.1 Introduction

7 Runtime operation

7.1 Introduction

Structure of this chapter

This chapter describes the runtime operating interface to PickMaster.

The runtime operating interface consists of three parts:

- PickMaster FlexPendant interface. A graphic operator's interface.
- PickMaster I/O interface. Used by a PLC.
- PickMaster RAPID interface. The RAPID interface can be customized to receive and handle requests not covered by the FlexPendant or the I/O interface. For a complete description of program modules, functions, procedures, and data specific for palletizing see *RAPID program on page 205*, and *RAPID reference information on page 287*.

Prerequisites

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

7 Runtime operation

7.2.1 Introduction to PickMaster FlexPendant interface

7.2 FlexPendant interface

7.2.1 Introduction to PickMaster FlexPendant interface

PickMaster FlexPendant interface

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

The PickMaster FlexPendant interface covers the following four areas:

- Open Project
- Production
- Process Signals
- Tune

Illustration

The illustration shows the PickMaster main menu.



xx0600002995

Open Project

Open Project displays a list of all the PickMaster projects that have been downloaded to the robot controller. A project must be opened before it can be started from the production window. The status bar always contains information about the loaded project as well as the current location within the window hierarchy. See *Opening a project on page 238*.

Production

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

Continues on next page	
236	

7.2.1 Introduction to PickMaster FlexPendant interface Continued

See Starting and stopping production on page 239.

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 <i>Zone frame on page 385</i> , and <i>Activators properties on page 125</i> .
	See also Process Signals on page 252.
Tune	
	Tune is used to change the parameter values online while running the PickMaster

Tune is used to change the parameter values online while running the PickMaster application. See *Tuning on page 255*. 7.2.2 Opening a project

7.2.2 Opening a project

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. For more information, see *Project on page 154*.

	Action	Note
1.	On the PickMaster main menu, tap Open Project .	The Open Project button is only accessible if the current PickMaster project is stopped.
2.	Tap a project.	When a project is selected, the OK button appears. To update the list of projects, tap Refresh .
3.	To open the project and return to the main menu, tap OK.	
4.	To view information about the project, tap View project info.	The window shows information about line path, transfer data, PickMaster version, user, computer, and a description of the project.
5.	To close the window, tap Cancel.	

The following figure shows the Open Project window.

🤾 Vi	🥇 Virtual FlexPendant			
		al _2034_DefaultProj(SEV)	Motors On Stopped (Speed 100%)	X
Pick	Master - Open Project	- MultiFlow-Demo: Stop	oped	
Nar	ne	Description		
2	MultiFlow-Demo	PZ-Demo - 3 Flows	5	
2	Flow-Demo	PZ-Demo - Single I	Flow	
8	Coffee	Palletizing Coffee		
⊢				
	Refresh View j info	project	ОК	Cancel
5	PickMaster			

xx0600002996

Related information

Starting and stopping production on page 239. Process Signals on page 252.

7.2.3 Starting and stopping production

7.2.3 Starting and stopping production

Overview

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

Illustration, Production window

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

		to opJobTest (SEVST-L-00036	Motors On 532) Running (Speed 100%)	⁷ X
Pick	Master - Production	n - BTC_1601_Long_te	erm_RC_mode: Running	
Flo	w	Flow status	Active job	Job status
1	Flow1	Running		Idle
4	Flow2	Running	Large boxes	Running
	Flow control	Dop Start All	Stop All	Stop
	Project	View Option	-	Close
	Production Window	r 🔁 1/0 🧮 1/0) 🤞 Jogging	

xx0700000229

Status	The status of the flow, which can be Stopped , Running , or Error . The status can also indicate type of stop in progress, for example, Stopping after cycle , Stopping after layer , and Stopping after pallet .
Active job	The currently running job on the flow, that is, the name of the active master operation set.

JobThe current job status of the flow, which can be Idle (no job isstatusrunning), Running or Stopping (a job has been ordered to stop).

Starting a project

	Action	Note
1.	On the Production menu, tap Project.	
2.	Tap Start.	A warning appears if the system is in motors off state.

7 Runtime operation

7.2.3 Starting and stopping production *Continued*

Stopping a project

	Action	Note
1.	On the Production menu, tap Project.	
2.	Tap Stop.	A warning appears.
	To proceed, tap Yes. To cancel, tap No.	

Restarting RAPID

The following procedure describes how to restart the program execution. This is the same function as pushing the hardware button *Start* on the IRC5 FlexPendant.

	Action	Note
1.	On the Production menu, tap Project .	
2.	Tap Restart RAPID.	The Restart RAPID button is only available when program is paused.

Starting a specific flow

Starting the flow will enable the starting and execution of a palletizing job.



The PickMaster project must be started before a flow can be started.

	Action
1.	In the list of defined flows, tap on the flow to start. The Start Flow button becomes available.
2.	Tap Start Flow.

Stopping a specific flow

Stopping the flow pause the execution of a running palletizing job. When the flow is restarted, the execution of the job will continue. A flow stop option can be selected to specify how and when the flow shall be stopped.

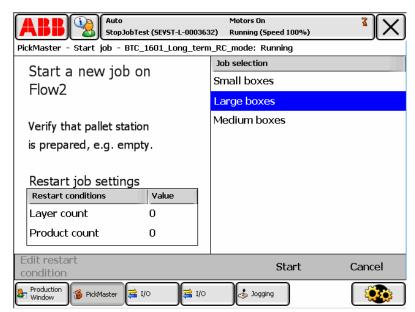
	Action
1.	In the list of defined flows, tap on the flow to stop. The Stop Flow button becomes available.
2.	Tap Stop Flow.

Starting a specific job

	Note
--	------

The flow must be running before a job can be started.

	Action
1.	In the list of defined flows, tap on the flow to start a job on. The Start Job button appears.
2.	Tap Start Job. The Start Job window appears.
3.	Select job and press start.



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

en1000000858

Job selection	A list of palletizing jobs defined for the master work area to select from.
Restart conditions	A list of parameters that needs to be specified when an unfinished job shall be restarted, for example, a half finished pallet pattern. When starting a new job, do not update these parameters. NOTE: A restart is not possible if the last operation was a partially completed multi drop operation. In that case, some products has to be manually removed from the stack before starting, for example, a removal of the top layer.
Layer count	Specifies number of available full layers, including defined pallet and slip sheets.
Product count	Specifies number of available products in the top layer. If the top layer is full, product count shall be set to zero.

Editing restart conditions

	Action
1.	Select restart condition.
2.	Press Edit restart condition menu.
3.	Enter appropriate value on the displayed numerical pad and press OK.

Stopping a specific job

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

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.



The flow must be running before a job can be stopped.

	Action
1.	In the list of defined flows, tap on the flow to stop the job on. The Stop Job button becomes available.
2.	Tap Stop Job. A pop-up window appears.
3.	In the pop-up window, confirm that the job shall be stopped.

Starting and stopping all flows

То	Do this
Start all flows	Tap Start All Flows.
Stop all flows	Stop All Flows.

Viewing flow status

	Action
1.	In the list of flows, tap on a specific flow. In the Production window, the Status command on the View menu becomes available.
2.	On the View menu, tap Status and select a work area.
3.	To close the Status Information window, tap Status on the View menu.

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.

۷ 🥈	irtual	FlexPendant				X
	B	• 5_0	nual 09_2034_DefaultP		s On ng (Speed 100%)	X
Pic	kMas	ter - Production	- MultiFlow-De	emo: Running		
Flo	W		Status		Stop Option	
1	M	Status Infor	mation: Med	ium Coffe ——		
4	S	Outfeed		Status:	Running	
				Operation Set:		
1	Li			Item Count:	0	
⊢						
				Status:	Running	
		Infeed		Operation Set:	Raining	
				Item Count:	0	
		L]
	Pro	oject 🔷	View	Stop Options		Close
5	PickMa	aster				

xx0700000228

Work	The name of the work area as specified in the PickMaster software.
Area	For further details, see <i>Feeder on page 170</i> .
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.
ltem	A counter for the accumulated number of items that has been picked
Count	or placed on the work area since the project start.

Viewing messages

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

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

🥇 Vir	tual FlexP	endant	(
	BB		Manual 5_09_2034_[)efaultProj(5E¥)	Motors On Running (Sp	eed 100%)	G	<u>IX</u>
Ever	nt Log								
Coe	te	Title					Date & Ti	me	
	2398	Messa	ige				2007-8-3	30 6:44:	21
	2399	PickM	aster Flow	Stoppe	d Imme	ediately	2007-8-3	30 6:2:5	7 📗
	_	_			_				
							Clear	C	ж
F 🚳	PickMaster								

xx0700000482

Code	The code of the message.
Title	The title of the message.
Date	The date and time of the generated message.
Clear	Clears the list of messages. Note that the messages are <i>not</i> deleted. If the production window is closed and then reopened, the cleared messages appear again.
ОК	Closes the window.

For details about message configuration, see *Messages on page 162*.

Changing or viewing flow stop options

	Action
1.	In the list of flows, tap on the flow to change the stop option for. The Flow Stop Op- tions button on the Production menu becomes available.
2.	Tap Flow Stop Options and select a stop option. The currently selected stop option is checked.

There are four different ways to stop each product flow from the FlexPendant. Each stop option is described according to what will happen when a flow is stopped with the specific stop option and after tapping **StopFlow**.

Flow stop option	Description
Finish cycle	The robot will continue palletizing until the current pick/place cycle of the job is finished.
Finish layer	The robot will continue palletizing until the current layer of the job is finished.
Finish job	The robot will continue palletizing until the current job is finished.

Flow stop option	Description
Stop immediately	The flow is stopped and a flow recovery action must be selected when restarting.
	If the flow is the current active flow, the robot will stop immediately, without finishing a started cycle. A warning symbol appears next to the flow, indicating that a flow recovery action must be selected, and the status changes to Error.
	If another flow shares one of the slave work areas, a warning symbol will appear also next to that flow when this slave is requested, but the flow status does not change.
	If the stopped flow is not the current active flow, the robot will continue palletizing using the remaining running flows.

A StopFlow command can be cancelled using Undo. However, this is not possible with flow stop option "Stop Immediately". To cancel a requested stop action:

• On the Stop Options menu, tap Undo.

Related information

Opening a project on page 238. Process Signals on page 252. Feeder on page 170. Events on page 159. Robot Settings on page 165. 7.2.4 Flow recovery

7.2.4 Flow recovery

Overview

If a work area is in error state, (for example, as a result of using event signals **Error Source** and **Trigger**), a warning symbol appears next to the flows that are affected.

If a flow is stopped in the error state, a warning symbol appears on the **Start Flow** button. For example, this occurs if the flow is stopped immediately or if the flow needs to access a work area in the error state. The warning symbol indicates that the flow must be recovered when started. The following section describes how to recover the flow in such a case.

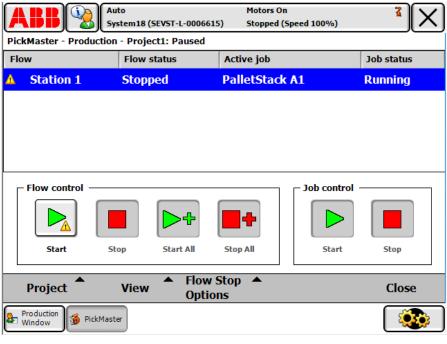
Recovering a specific flow

Overview

	Action	Note
1	In the Production window, tap Start Flow.	See Illustration, Production window with a flow to recover on page 247.
2	The Restart options window appears. Select one of the displayed recovery options and tap OK.	You get three options to continue. See <i>Illustra-</i> <i>tion, Restart options window on page 248.</i>
3	The Production window appears and there is a note symbol on the Start Flow button. Tap Start Flow .	See Illustration, Production window after se- lecting a recovery option on page 249.
4	A dialog box containing information about the selected option appears. Verify the status as specified in the message, and then tap OK.	After tapping OK, the flow is in production again. If the RAPID program has stopped, it will re- start. See <i>Illustration, Confirm restart informa-</i> <i>tion on page 250</i>

Illustration, Production window with a flow to recover

The following figure illustrates the **Production** window when a flow has been forced to stop due to an error.



xx1300000753

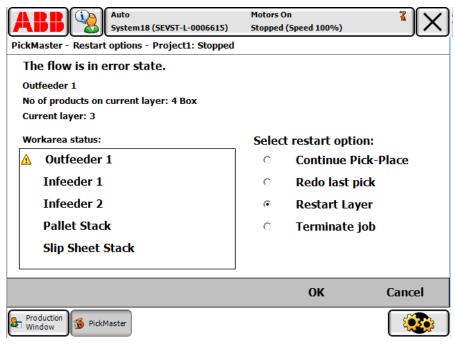
Start Flow

The warning symbol indicates that a flow has been forced to stop by an error in a work area or with the stop option **Stop immediately**. When the **Start Flow** button is pressed, the **Restart options** window appears.

7 Runtime operation

7.2.4 Flow recovery *Continued*

Illustration, Restart options window



xx1300000754

The work areas in error state are marked with a warning symbol. Select a suitable recover action and press **OK** to confirm.

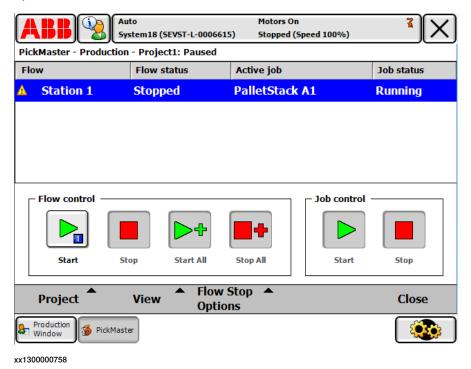
Recover action	Description
Continue Pick-Place	The flow continues from where it was stopped.
Redo last pick	The flow repeats the last pick operation. Redo last pick is en- abled in the gap only after the first product is picked and before the first product is placed.
Restart layer	The flow restarts the job from the beginning of the current layer, starting with the picking.
Terminate job	The flow terminates the current job, cancel the remaining oper- ations, and enable the starting of the next job.



The robot moves directly to the pick work area after a restart with flow recovery when using **Redo last pick**, **Restart layer**, or **Terminate job**. Any passed safe position is not 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.

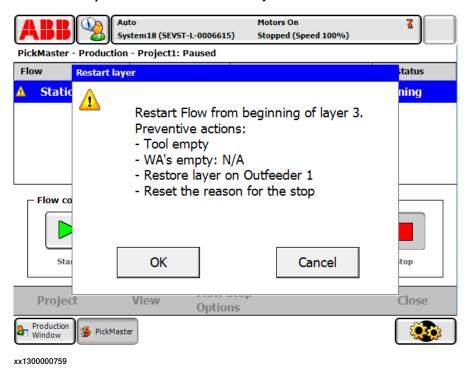


Start Flow

The info symbol indicates that a flow recovery action has been selected for the flow. Tap **Start** to display more information about the selected action, expected number of products in robot tool and on work areas, and so on.

Illustration, Confirm restart information

The following figure illustrates the information window to confirm while starting the flow after selecting a flow recovery action. Tap OK to immediately run the flow start. The error state for the flow and the work areas are reset. Tap **Cancel** to open the **Restart options** window from where you can select a new restart 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



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

Illustration, Restart other flows after error event

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

	/		
	Manual	Motors On	
	Doc(SEVST-L-0003202)	Running (Speed 100%)	
All tasks	T_ROB1 UIMessageBox		
Resta:	rt after error eve	ent!	
Verify that t	gram has been stoppe he cause of the stop xecution will contin	is removed before s	tarting.
			ок
Production Window	DB1 : Jain PickMaster		
xx0700000503			

Related information

Starting and stopping production on page 239

7.2.5 Process Signals

7.2.5 Process Signals

Introduction

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

- The I/O signals that are connected to a work area.
- The zones and the activators that build up the tool configuration.
- · The event signals.

Viewing the work area signals

	Action	Note
1.	On the PickMaster main menu, tap Pro- cess Signals.	
2.	Tap View and select Work Areas.	See <i>Feeder on page 170</i> .
3.	From the Work Areas list, tap one work area.	

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

ickMaster - Process Signals Work Areas	Name	Value	Type
Slip & Pallet	doSlipfeedTrig	0	DO
Medium Infeed	goSlipfeedProd	0	GO
C Medium Outfeed	goSlipfeedForm	0	GO
Small Outfeed	doSlipfeedTrig	0	DO
Large Outfeed	goSlipfeedProd	0	GO
Small & Large Infee	d goSlipfeedForm	0	GO
	doSlipfeedPosA	0	DO
	doSlipfeedQI	0	DO
	doSlipfeedBC	0	DO
View Close			

xx0700000226

Viewing the tool configuration

	Action
1.	On the PickMaster main menu, tap Process Signals.
2.	Tap View and select Tool.
3.	From the Tool list, tap one tool configuration.

7.2.5 Process Signals Continued

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

Auto Motors On Doc(SEVST-L-0002266) Stopped (Speed 100%) PickMaster - Process Signals - Coffee: Stopped					
Multigripper					
Tool Configurations	Nan	ne	Value	Туре	
🐲 Normal		Zone1	Deactive	Zone	
🐲 Small		Zone2	Deactive	Zone	
🐲 Large	(Zone3	Deactive	Zone	
	۵	Zone4	Deactive	Zone	
	•	Activator2	Deactive	Activator	
	•	Activator1	Deactive	Activator	
	•	Activator4	Deactive	Activator	
	•	Activator3	Deactive	Activator	
View Close					
🥳 PickMaster					

xx0700000225

Viewing events

	Action	Note
1.	On the PickMaster main menu, tap Process Signals .	
2.	Tap View and select Events.	See Events on page 159.
3.	From the Events list, tap one controller.	

7.2.5 Process Signals *Continued*

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

🥇 5_09_2034_DefaultProj				
ABB Manual 5_09_20	Motor: 34_DefaultProj(SEV) Stoppe	s Off ed (Speed 100%)	X	
PickMaster - Process Signals	- MultiFlow-Demo: Stopped			
Events	Name	Value	Туре	
IRC5	diErrorTrigger	0	DI	
	giErrorSourse	0	GI	
	giErrorMessage	0	GI	
View			Close	
S PickMaster				
7				

xx0700000407

Related information

Opening a project on page 238.

Starting and stopping production on page 239.

7.2.6 Tuning

7.2.6 Tuning

Introduction

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

Illustrations, Tune window

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

ltem

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

AULO Test_260 (Moto SEVST-L-0005370) Stopp	rs On ped (Speed 10	D%)	<pre>X</pre>		
PickMaster - Tune - myProject:	PickMaster - Tune - myProject: Stopped					
Item	Property	Current	Default	Unit		
Small Coffee	Speed	5000	5000	mm/s		
Medium Coffe	Ori speed	360	360	degrees/s		
Large Coffee	Acceleration	5	5	m/s2		
Pallet	Pick time	0,01	0,01	s		
Slipsheet	Place time	0,01	0,01	s		
	Size z	80	80	mm		
	Pick activation	0,1	0,1	<u>*</u>		
View Reset a	all Tune value			Close		
5 PickMaster						

xx0700000408

Item	The name of an item as specified in the PickMaster project. See
	Product/Pallet/Sheet on page 59.
Property	The property of an item. For description of the parameters, see

riopenty	The property of an item. For description of the parameters, s
	table below.

- Current Specifies the actual current value of the property.
- DefaultSpecifies the default value of the property as it was in the
PickMaster project when it was downloaded to the controller.
- **Reset All** All items and work areas are set to their respective default value.

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

Parameter	Description	
Speed	Specifies the maximum speed for an item.	
Ori speed	Specifies the maximum orientation speed for an item.	

7.2.6 Tuning Continued

Parameter	Description	
Acceleration	Specifies the maximum acceleration/deceleration for an item.	
Pick time	Specifies the time the robot stays at the target position when picking an item.	
Place time	Specifies the time the robot stays at the target position when placing an item.	
Size Z	Specifies the height of the item. When updating the item height, the height of pick and place positions for next items to be handled will be affected accordingly.	
	Always when updating the item height, the operator is asked if the new value shall be applied also to previously placed layers. If the answer is yes, product place positions will be affected by height updates of all previously placed items in lower layers. If the answer is no, place positions will only be affected by the height updates of new items to be placed. Pick positions will never be affected.	
Pick Activation	Specifies the time for pick activation.	
Place Activation	Specifies the time for place activation.	

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.

		tors On nning (Speed 1	00%)	X	
PickMaster - Tune - MultiFlow-Demo: Running					
Work Area	Property	Value	Default	Unit	
Infeed	Disp offs x	10	0	mm	
Outfeed	Disp offs y	0	0	mm	
Slip feed	Disp offs z	0,8	0	mm	
Infeed 2	Disprotz	0	0	degrees	
Outfeed 2					
Outfeed 3					
	•				
View Reset A	All Tune Value			Close	
🥳 PickMaster					

xx0700000409

Work area	The name of a work area specified in the PickMaster project. See <i>Feeder on page 170</i> .
Property	The property of a work area. For description of the parameters, see table below.
Value	Specifies the actual current value of the property.
Default	Specifies the default value of the property as it was in the PickMaster project when it was downloaded to the controller.

7.2.6 Tuning Continued

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

Parameter	Description
Disp offs x	Displacement of the work area in x-direction relative the work object.
Disp offs y	Displacement of the work area in y-direction relative the work object.
Disp offs z	Displacement of the work area in z-direction relative the work object.
Disprotz	Displacement angle of the work area in the z-direction.

How to proceed

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

Tuning an item

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

The following figure illustrates tuning of an item.

🥉 Virtual FlexPendant			<u>_ ×</u>
	ual 9_2034_DefaultProj(SE¥)	Motors On Running (Speed 100%)	X
PickMaster - Tune - Mult	tiFlow-Demo: Running		
Slipsheet			
Speed			
	5000 mm	/s 🖕	
Increment:	1 mm/s	•	
Limits:	5000 - 500	00 mm/s	
Default:	5000		
		Apply	Cancel
DickMaster			

xx0700000411

257

7.2.6 Tuning Continued

Tuning a work area

	Action	Note
1.	On the Tune menu, tap View .	
2.	Tap Work Area.	
3.	In the Work Area list, tap the work area to tune.	
4.	In the Property list, tap the work area property to tune.	
5.	On the Tune menu, tap Tune Value .	The Work Area Value window appears. See illustration below this procedure.
6.	In the Increment drop-down combo box, select the size of increment.	The increment specifies the value that will be added to/subtracted from the item property for each time you tap the plus or minus but- ton.
7.	Tap Apply .	

The following figure illustrates tuning of a work area.

🤞 Yirtual FlexPendant				
Manual 5_09_2034_DefaultProj(SEV)		Motors On Running (Speed 100%)	X	
PickMaster - Tune - Mult	tiFlow-Demo: Running			
Slip & Pallet				
Disp offs x	[
	0 mm			
Increment:	0.1 mm	•		
Limits:	0 - 0	mm		
Default:	0			
		Apply	Cancel	
🥳 PickMaster				

xx0700000410

7.3.1 Overview

7.3 I/O interface

7.3.1 Overview

PickMaster I/O interface

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

- The basic I/O interface defines work area specific signals. It covers the minimum I/O configuration needed to run a PickMaster project.
- The extended I/O interface adds optional functionality, for example reporting error events, starting projects and flows, and so on. You can use the extended I/O interface if needed.

7.3.2 Default signals

7.3.2 Default signals

Introduction to default signals

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

Configuration and setup

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

Additional signals are required if you use more than:

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

Use RobotStudio for:

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

Description of default signals

The following default signals are installed.

Work areas

Default signals are defined for eight work areas. Signal name prefixes:

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

Controller system actions

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

pmSystem

Project handling

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

pmProject

Continues on next page
260

7.3.2 Default signals Continued

Flows

Default signals are defined for flows. Signal name prefixes:

- pmFlow
- pmFlow1
- pmFlow2
- pmFlow3
- pmFlow4

Grippers

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

• pmGripper1

Event reporting

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

pmEvent

7.3.3 Basic I/O interface

7.3.3 Basic I/O interface

General

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

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

I/O signals must be setup for each work area. Some of the signals must be used and others can be used if needed.

How to use the work area signals depends on if the work area will be used as a master or slave.

Master work areas

Overview

The following signals must be configured:

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

The following signals must be setup if simulated target generation is selected for the work area. See Feeder on page 170.

Position request trigger signal (DO)



Note

The signal is mandatory only 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.

7.3.3 Basic I/O interface Continued

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 a job is completed.
- When a job is stopped before completing.
- When an operation set is finished using the robot execution signal.
- As a result of a flow recovery action.

A new job/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.

7.3.3 Basic I/O interface *Continued*

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.

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 an operation set have been placed/picked and the robot has departed from the work area after the last place/pick operation.
- If the job is stopped (e.g. before being completed)
- If the current job is finished before being completed by resetting the robot execution signal.

The signal is reset when the target generation trigger signal is pulsed. If the target generation signal is set very early, that is, just before or just after operation set is complete, the signal will be reset after a time corresponding to the configured pulse length. For more information on pulse length configuration, see *Controller Settings* on page 158.

Default signal, example: pmOutfeeder1_doOpSetCompl.

7.3.3 Basic I/O interface Continued

Execution state (GO)

The signal is optional.

The signal indicates the runtime state of the work area.

I/O value	Description	
0	A project using this work area is not running.	
1	Work area is running.	
2	Work area has an error. Flow recovery is required to recover from the error.	
3	Work area has a response error, that is, the PLC has generated wrong targets. A generation of correct targets is required to recover from the error.	

Default signal, example: *pmOutfeeder1_goExecState*.

Height state (GO)

The signal is optional.

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

I/O value	Description	
0	Full height. The height is equal to the setting of Full height , see <i>Robot path height on page 176</i> .	
1	Active height. An operation set is executed and the height is updated after each pick or place until the operation set is completed.	
2	Latest height. The height is equal to the final height of the latest run operation set.	
3	Empty height. The height is equal to the setting of Empty height , see <i>Robot path height on page 176</i> .	
4	Value. The height is set to a value with the RAPID function PmSetDefaultHeight.	

Default signal, example: pmOutfeeder1_goHeightState.

Layer count (GO)

The signal is optional.

The signal value indicates the number of full layers on the work area. Default signal, example: *pmOutfeeder1_goLayerCount*.

Product count (GO)

The signal is optional.

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

Default signal, example: pmOutfeeder1_goProductCount.

Stop job (DI)

The signal is optional.

7.3.3 Basic I/O interface *Continued*

	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 <i>Feeder on page 170</i> .
	If the job is waiting on a slave that is running out of products, stop job can be used to finish the job without running any further pick-place cycles. The job will become stopped after the stop job timeout has passed.
	Default signal, example: pmOutfeeder1_diStopJob.
Robot execution (DI)
Υ.	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.
	3 , 1 , P - - - - - - - - - -
Redo search (DI)	
Redo search (DI)	The signal is optional.
Redo search (DI)	
Redo search (DI)	The signal is optional.
Redo search (DI)	The signal is optional. The signal is used with operation sets having stack search activated.
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.
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
	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.
	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.
	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. The following signals must be setup:
	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. The following signals must be setup: • Target generation trigger signal (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. The following signals must be setup: • Target generation trigger signal (DI) • Target generation product selection (GI) ²
	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: <i>pmOutfeeder1_diRedoSearch</i> . The following signals must be setup: • Target generation trigger signal (DI) • Target generation product selection (GI) ² • Target generation format selection (GI) ³
	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: <i>pmOutfeeder1_diRedoSearch</i> . 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)
	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: <i>pmOutfeeder1_diRedoSearch</i> . 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 trigger signal (DO)

7.3.3 Basic I/O interface Continued

- Position request product selection (GO)²
- Position request format selection (GO)³
- 2) Only mandatory if there is more than one item.
- 3) Only mandatory if there is more than one format for the same item.

Position request trigger signal (DO)

The signal is mandatory.

The signal is set by the controller when one of the corresponding master work areas requests a format on this slave work area from the PLC. The requested format must be defined as an operation set in the 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 173*.

Default signal, example: pmInfeeder1_goReqMaster.

Target generation trigger signal (DI)

The signal is mandatory unless simulated target generation is used.

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

Default signal, example: *pmInfeeder1_diTgtGenTrig*.

Target generation product selection (GI)

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

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

Default signal, example: *pmInfeeder1_giProdSel*.

Continues on next page

7.3.3 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: *pmlnfeeder1_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.

7.3.3 Basic I/O interface Continued

The signal is set:

- After all products in an operation set have been placed/picked and the robot has departed from the work area after the last place/pick operation.
- If the current operation set is finished before being completed by resetting the robot execution signal.

The signal is reset when the target generation trigger signal is pulsed. If the target generation signal is set very early, that is, just before or just after operation set is complete, the signal will be reset after a time corresponding to the configured pulse length. For more information on pulse length configuration, see *Controller Settings on page 158*.

Default signal, example: pmInfeeder1_doOpSetCompl.

Execution state (GO)

The signal is optional.

The signal indicates the runtime state of the work area.

I/O value	Description
0	A project using this work area is not running.
1	Work area is running.
2	Work area has an error. Flow recovery is required to recover from the error.
3	Work area has a response error, that is, the PLC has generated wrong targets.
	A generation of correct targets is required to recover from the error.

Default signal, example: *pmInfeeder1_goExecState*.

Height state (GO)

The signal is optional.

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

I/O value	Description
0	Full height. The height is equal to the setting of Full height , see <i>Robot path height on page 176</i> .
1	Active height. An operation set is being executed and the height is updated after each pick or place until the operation set is completed.
2	Latest height. The height is equal to the final height of the latest run operation set.
3	Empty height. The height is equal to the setting of Empty height, see <i>Robot path height on page 176</i> .
4	Value. The height is set to a value with the RAPID function PmSetDefaultHeight.

Default signal, example: pmInfeeder1_goHeightState.

Layer count (GO)

The signal is optional.

7.3.3	Basic	I/O	interface
Conti	nued		

	The signal value indicates the number of full layers on the work area. For example, for a pallet stack or a slip sheet stack.
	Default signal, example: pmIntfeeder1_goLayerCount.
Product count (GO)	
	The signal is optional.
	The signal value indicates the number of products in the top layer. If the top layer is full, the signal is set to zero.
	Default signal, example: pmInfeeder1_goProductCount.
Stop job (DI)	
	The signal has no function and is not required.
Robot execution (DI)	
	The signal is optional.
	The signal is used to control whether the robot is allowed to approach the work area or not. If the signal is reset, the RAPID execution will not pass the instruction PmGetTarget until the signal is set. After an operation is performed by the robot, the signal must be reset and then set again to allow the robot to approach the work area next time.
	The signal can also be used to finish the current operation set before all operations have been completed. If the signal is reset, the remaining targets will be removed.
	Default signal, example: pmInfeeder1_diRobotExec.
Redo search (DI)	
. ,	The signal is optional.
	The signal is used with operation sets having stack search activated. If the signal is pulsed, next operation will start with a search movement from the top of the stack. The signal can be used after adding new items on a stack. To affect the next approach to the work area, the signal must be pulsed before the robot receives the operation in RAPID.
	Default signal, example: pmInfeeder1_diRedoSearch.

7.3.4 Extended I/O interface

7.3.4 Extended I/O interface

Controller system handling

Introduction

A number of default system signals are installed to handle the controller system, for example to set the controller in motors on state.

See *Technical reference manual - System parameters*, the topic *I/O*, for descriptions of system inputs and outputs.

System inputs

System input	Description
pmSystem_diLoadStart1	Load and start the PmProjMgr module for motion task T_ROB1.
pmSystem_diLoadX	Load the PmProjMgr module for motion task T_ROBX.
pmSystem_diStart	Start RAPID execution.
pmSystem_diStop	Stop RAPID execution.
pmSystem_diStartMain	Start RAPID execution from Main.
pmSystem_diMotorsOn	Set motors on.
pmSystem_diResetEstop	Confirm reset of emergency stop.

System outputs

System output	Description	
pmSystem_doLoadReadyX	A robot program (for example, PmProjMgr) is loaded for T_ROBX.	
pmSystem_doCycleOn	Robot program is executing.	
pmSystem_doMotorOnState	Set while in Motors on State. Otherwise, its reset.	
pmSystem_doMotorOn	Set while in Motors on State. If the controller is in guard stop, the output starts pulsing with a frequency of 1 sec. If the controller is not calibrated or the revolution counter is not updated, the output will pulsate even faster in manual mode.	
pmSystem_doRunchOk	Run chain is closed.	
pmSystem_doEmStop	Emergency stop state.	
pmSystem_doAutoOn	Automatic mode is used.	

Project handling

Overview

It is possible to start, halt, restart, stop, and supervise projects.

Default signals

The following default signals must be used for project handling:

Project signal	Description
pmProject_goCurrent	The current project.
pmProject_goStatus	The status of current project.

7.3.4 Extended I/O interface *Continued*

Project signal	Description
pmProject_diStop	Stop current project.
pmProject_diStart	Start selected project.
pmProject_giSelection	Project selector.
pmProject_diSetDefaultheight	Set default height for a work area.
pmProject_giDefaultHeight	Default height selector.
pmProject_giDefHeightWaSel	Work area selector for setting the default height.
pmProject_diNoWait	PickWare internal use.
pmProject_diNoWait	PickWare internal use.

Project status values

The current status of the project is reflected by the signal *pmProject_goStatus*.

I/O value	Description
0	Project is stopped.
1	Project is stopping.
2	Project is starting.
3	Project is running.
5	Project in error state.

Default height values

The following selections are supported when setting the default height.

I/O value	Description
0	Full
1	Latest
2	Empty
3	Value
4	Standard, that is, as configured in the Robot Path Height . See <i>Robot path height on page 176</i> .

Starting a project for a single robot system

	Action
1	Ensure that the previous project is stopped, that is he signal ${\tt pmProject_goStatus}$ is 0.
2	Switch to motors on state using the controller system signals. See <i>Technical reference manual - System parameters</i> , the topic <i>I/O</i> .
3	Pulse the system input pmSystem_diLoadStart1 to load and start the PmProjMgr module.
4	Wait until the robot program is started, that is when <i>pmSystem_doCycleOn</i> is set.
5	Set pmProject_giSelection to select the project to run. The I/O value for the project is set in the Project I/O value editor. See <i>Project Manager</i> on page 108.
6	Pulse pmProject_diStart to start the project.

7.3.4 Extended I/O interface Continued

	Action
7	Wait until the project has started, that is when pmProject_goStatus goes to 3.

Starting a project for a multimove system

If the used RobotWare version is older than 6.02, a few manual preparations are needed. A system input must be defined for all the ¹motion tasks, having the following arguments:

Name	Value
Signal name	pmSystem_diLoadX
Action	Load
Argument1	PmProjMgr.mod
Argument2	T_ROBX
Argument3	N/A
Argument4	N/A

	Action
1	Ensure that the previous project is stopped, that is the signal pmProject_goStatus is 0.
2	Switch to motors on state using the controller system signals. For more information, see $Topic$ I/O in Technical reference manual - System parameters.
3	Pulse the signal pmSystem_diLoadX, to load PmProjMgr module for motion task T_ROBX.
4	Wait until the module is loaded, that is when pmSystem_doLoadReadyX is set.
5	Repeat steps 3 and 4 for the next motion task (for example, T_ROB2) until all motion tasks are loaded.
6	Pulse the signal pmSystem_diStart, to start rapid execution for all motion tasks (this will also cause all pmSystem_doLoadReadyX to reset).
7	Wait until the robot programs has started, that is when <code>pmSystem_doCycleOn</code> is set.
8	Set pmProject_giSelection to select the project to run. The I/O value for the project is set in the Project I/O value editor. For more information, see <i>Project Manager on page 108</i> .
9	Pulse pmProject_diStart to start the project.
10	Wait until the project is started, that is, when pmProject_goStatus is 3.

Stopping robot program - halting project

	Action
1	Pulse pmSystem_diStop to stop the robot program.
2	Wait until the robot program is stopped, which occurs when <code>pmSystem_doCycleOn</code> is reset.

¹ A system input is preconfigured for T_ROB1. This instance can be copied and modified to create system inputs for T_ROB2, T_ROB3 and so on.

7.3.4 Extended I/O interface *Continued*

Restarting robot program - restarting project

	Action
1	Switch to motors on state using the controller system signals. See <i>Technical reference manual - System parameters</i> , the topic <i>I/O</i> .
2	Pulse <i>pmSystem_diStart</i> to start the robot program.
3	Wait until the robot program is started, which occurs when <i>pmSystem_doCycleOn</i> is set.

Stopping current project

	Action
1	Pulse <i>pmSystem_diStop</i> to stop the robot program.
2	Wait until the robot program is stopped, which occurs when <i>pmSystem_doCycleOn</i> is reset.
3	Pulse pmProject_diStop to stop the project.
4	Wait until project is stopped, which occurs when <i>pmProject_goStatus</i> goes to 0.

Set a new default height for a work area

Setting a new default height is a possibility to save cycle time without decreasing the margins for collisions, especially if the project consists of many work areas and flows.

For an outfeeder the default height can be set to *Empty* after a finished stack has been unloaded. This may allow the robot to make lower intermediate movements when passing over the outfeeder next time and thus saving cycle time.

For an infeeder the default height can be set to *Full* before a new stack is loaded. This will force the robot to make intermediate movements with enough height when passing over the work area.

The new default height is active until new targets have been generated (or after a new default height is set).

	Action
1	Set pmProject_giDefaultHeight to select the new default height.
2	Set <i>pmProject_giDefHeightWaSel</i> to select the work area. The I/O value of the work area is configured in the Work Area I/O Settings editor. See <i>Edit detailed signals for feeder on page 173</i> .
3	Pulse pmProject_diSetDefaultHeight.
4	Wait until the default height is updated, that is, the work area height state GO signal is updated to reflect the change. Note!
	If the current height state is 1, which means active height, the height state will not be updated until the last target of the current operation set is picked/placed.

7.3.4 Extended I/O interface Continued

Flow handling

It is possible to start, stop, and supervise, and recover flows.

Default signals

The following default signals must be used for flow handling:

Flow signal	Description
pmFlow_diStart	Start selected flow.
pmFlow_diStop	Stop selected flow.
pmFlow_diRecover	Recover the selected flow with the selected recover action on the selected work area.
pmFlow_giSelection	Flow selector.
pmFlow_giStopOption	Set stop option.
pmFlow_giRecoverAction	Set recover action.
pmFlow_giWaRecoverSelection	Select work area for recover action.
pmFlowX_goStatus	Status of flow X (X=1,2,3,).

Flow status values

The statuses of the flows are reflected by the *pmFlowX_goStatus* signals.

The status signal for the flow is setup in the Flow editor. See Flow on page 202.

If a flow goes to error state, you can do flow recovery from the I/O Interface or the PickMaster FlexPendant interface.

I/O value	Description
0	Flow is stopped.
1	Flow is running.
2	Flow is stopping after current pick place cycle.
3	Flow is stopping after current layer.
4	Flow is stopping after current pallet/operation set.
5	Flow in error state.

Flow stop options

Stop option is specified with the signal *pmFlow_giStopOption*.

I/O value	Description
0	Stop immediately.
1	Finish cycle.
2	Finish layer.
3	Finish pallet/operation set.

Flow recover actions

A flow recover action is specified with the signal *pmFlow_giRecoverAction*.

I/O value	Description
1	Continue
2	Restart layer

7.3.4 Extended I/O interface *Continued*

I/O value	Description
3	Terminate job/Next pallet
4	Redo last pick

Starting or restarting a flow

	Action
1	Ensure that current project is running. The <i>pmProject_goStatus</i> signal is set to 3.
2	Set <i>pmFlow_giSelection</i> to select flow. The I/O value for the flow is set in the Flow editor . See <i>Flow on page 202</i> .
3	Pulse pmFlow_diStart to start the project.
4	Wait until the flow is started, which occurs when the flow's status signal, <i>pm-FlowX_goStatus</i> , goes to 1.

Stopping a 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 shall be stopped.

	Action
1	Set <i>pmFlow_giSelection</i> to select flow. The I/O value for the flow is set in the Flow editor . See <i>Flow on page 202</i> .
2	Set <i>pmFlow_giStopOption</i> to select stop option.
3	Pulse <i>pmFlow_diStop</i> to stop the project.
4	For stop options 1-3, wait until the flow is stopped, which occurs when the flow's status signal, <i>pmFlowX_goStatus</i> , goes to 0. For stop option 0, wait until the flow changes to error state, which occurs when the flow's status signal goes to 5.

Stopping a flow immediately

When stopping a flow with stop option 0, it stops immediately, a flow recover action, for example Continue, must be specified before restarting the flow.

If this is the only running 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.

Recovering a flow

If a flow has entered error state, the *pmFlowX_goStatus* signal is set to 5. Use this procedure to recover the flow.

	Action
1	Set <i>pmFlow_giSelection</i> to select flow. The I/O value for the flow is configured in the Flow editor . See <i>Flow on page 202</i> .
2	Set <i>pmFlow_giRecoverAction</i> to select recover action.

7.3.4 Extended I/O interface Continued

	Action
3	Set <i>pmFlow_giWaRecoverSelection</i> to select which work area the recover action shall be applied for.
	The signal must not be set for the recover actions <i>Continue</i> and <i>Redo last pick</i> . 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 Settings editor. For more details, see <i>Events on page 159</i> .
4	Pulse <i>pmFlow_diRecover</i> to recover the flow.
5	An elog message is 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 <i>PmSetRecoverAction - Set flow recover action on page 308</i>).
	The flow is now prepared for being restarted.
	Note Note
	The error state of the flow indicated by ${\tt pmFlowX_goStatus}$ will not change until the flow is restarted.

Stopping an infeeder

Stopping the robot from moving to an infeeder, that is a slave work area. The robot program will stop if the robot just has started a movement to the infeeder. In all other situations, the robot will continue to run all possible pick and place cycles that does not include the infeeder. If a flow comes to a point where it needs the infeeder in order to continue palletizing, the flow will stop in error state. But the robot may still continue to run pick and place cycles for other running flows. If the infeeder is redundant within a flow, that is there exists another infeeder that can supply the same units, the flow never stops (unless all redundant infeeders have been stopped).

	Action
1	Set <i>pmEvent_giErrorSource</i> to select the infeeder to be stopped. For more information to setup event signals for the robot controller, see <i>Events on page 159</i> .
2	Pulse <i>pmEvent_diTrigger</i> to generate an error on the infeeder.
3	Wait until the infeeder is in error state, which occurs when the execution state signal, <i>pmInfeeder1_goExecState</i> goes to 2.
	Note

A multiple selection of infeeders is possible as in the preceding sequence.

Restarting an infeeder

Restarting an infeeder that is stopped. The infeeder can be restarted when the flow is executing or from a state where flows have been forced to stop. Afterwards, the robot will be able to pick from the infeeder again.

	Action
1	Select a flow that has reached error state, that is $pmFlowX_goStatus = 5$, that needs a stopped infeeder. If there are no such flows, select any flow that uses the infeeder. Set $pmFlow_giSelection$ to the selected flow. The I/O value for the flow is configured in the Flow editor. For more information, see <i>Flow on page 202</i> .

Continues on next page

7.3.4 Extended I/O interface *Continued*

	Action
2	Set pmFlow_giRecoverAction to 1 (= Continue).
3	Set <i>pmFlow_giWaRecoverSelection</i> to select the infeeder. The I/O value of the work area is configured in the Work Area I/O Settings editor. For more details, see <i>Events on page 159</i> .
4	Pulse <i>pmFlow_diRecover</i> to set recover action for the flow.
5	Pulse <i>pmFlow_diStart</i> to apply the recover action and start the selected flow.
6	Wait until the flow is running, <i>pmFlowX_goStatus</i> = 1, and the infeeder is running, <i>pmInfeeder1_goExecState</i> = 1.
	1 Note
	The flow may stay in error state, if it uses another infeeder that has been stopped. To get the flow running, a restart will be needed for that infeeder too.
7	If there is another flow that has an error state that needs the previously stopped infeed- er, continue from Action 1 to restart the flow.
1	Note
	hile recovering a flow using other restart actions and work area selections than escribed above, all infeeders used by the flow will be restarted.

Event and error reporting

Default signals

It is possible to report events and errors for work areas, robots, and controllers that affect the runtime operation. See *Events on page 159*.

The following default signals can be used for event and error reporting.

Signal	Description
pmEvent_diTrigger	Generate an event.
pmEvent_giErrorSource	Select error source(s).
pmEvent_giMessage	Select elog message.

Example, Report an error for a work area and log an elog message

	Action
1	Set the proper bit for <i>pmEvent_giErrorSource</i> to select a work area. The bit representing the work area is set in the Event settings tab of the Controller Properties . See <i>Events on page 159</i> .
2	Set <i>pmEvent_giMessage</i> to select a message. Values that represent different messages are set in the Message Settings , see section <i>Messages on page 162</i> .
3	Pulse <i>pmEvent_diTrigger</i> to generate the error and the elog message.
4	Wait until the work area enters error state, which occurs when the execution state signal of the work area, <i>pmFlowX_goExecState</i> , gets the value 2.

7.3.4 Extended I/O interface Continued

Example, Log an elog message

	Action
1	Set the proper bit for <i>pmEvent_giErrorSource</i> to select a controller. The bit that represents the controller is set in the Event settings tab of the Controller Properties . See <i>Events on page 159</i> .
2	Set <i>pmEvent_giMessage</i> to select a message. Values that represent different messages are set in the Message Settings , see section <i>Messages on page 162</i>
3	Pulse <i>pmEvent_diTrigger</i> to generate the elog message.

Robot tool

The control of the robot tool in runtime operation is integrated in the PickMaster RAPID interface and defined by the project configuration.

Available default signals for tools:

Signal	Description
pmGripper1_goActivators	Activators control
pmGripper1_doActivator1Open	Activator control
pmGripper1_doActivator1Close	Activator control
pmGripper1_doActivator2Open	Activator control
pmGripper1_doActivator2Close	Activator control
pmGripper1_doActivator3Open	Activator control
pmGripper1_doActivator3Close	Activator control
pmGripper1_doActivator4Open	Activator control
pmGripper1_doActivator4Close	Activator control
pmGripper1_diActivator1Opened	Activator status
pmGripper1_diActivator1Closed	Activator status
pmGripper1_diActivator2Opened	Activator status
pmGripper1_diActivator2Closed	Activator status
pmGripper1_diActivator3Opened	Activator status
pmGripper1_diActivator3Closed	Activator status
pmGripper1_diActivator4Opened	Activator status
pmGripper1_diActivator4Closed	Activator status
pmGripper1_diPartCheck1	Part check status
pmGripper1_diPartCheck2	Part check status
pmGripper1_diPartCheck3	Part check status
pmGripper1_diPartCheck4	Part check status
pmGripper1_diPartCheck5	Part check status
pmGripper1_giPartCheck1	Part check status
pmGripper1_goSearchActivate	Search tool activation.
pmGripper1_diSearchStop	Stack search stop trigger.
pmGripper1_goToolEvent1	GO tool event.

7.3.4 Extended I/O interface *Continued*

Signal	Description
pmGripper1_goToolEvent2	GO tool event.
pmGripper1_goToolEvent3	GO tool event.
pmGripper1_goToolEvent4	GO tool event.
pmGripper1_goToolEvent5	GO tool event.
pmGripper1_doToolEvent1	DO tool event.
pmGripper1_diToolEvent1	DI tool event.
pmGripper1_giToolEvent1	GI tool event.
pmGripper1_giToolEvent2	
pmGripper1_giToolEvent3	GI tool event.
pmGripper1_giToolEvent4	GI tool event.
pmGripper1_giToolEvent5	GI tool event.

7.3.5 Timing diagrams for PLC communication

7.3.5 Timing diagrams for PLC communication

Introduction

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.

Event	Description		
Operate infeeder	The RAPID execution has executed the RAPID procedure Operate for the infeeder.		
Pick	The robot has reached the pick position and picked up a complete format in the tool.		
Operate outfeeder	The RAPID execution has executed the RAPID procedure Operate for the outfeeder.		
Place	The robot has reached the place position and placed a complete format on the work area.		
Pallet pattern unloaded	A finished pallet pattern leaves the working range of the robot when being transferred from an outfeeder.		
Operation set complete	An operation set is complete when the robot has picked/placed the last product in a job/operation set and then departed from the last pick/place position. The doOpSetCompl signal is set a short time after the precedent pick/place event when the robot reaches the last depart target action.		

The following events in the palletizing process can be found in the timing diagrams:

See examples:

- Example minimum process control on a running flow on page 282.
- Example robot execution control on page 283.
- Example height control of a running flow on page 284.
- Example flow control on page 285.

7.3.5 Timing diagrams for PLC communication *Continued*

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.

A doPosRe <u>qTrig</u>	c		D	c 		□]
diTgtGenTrig	1					
В						
doPosAv <u>ail</u>	h		r	–∟		
doQueue <u>Empty</u>				_i		
doOpSetCompl						1
goExecState	1 (running)					
goHeightState 0(full)]						0 (full)
E	G	н	G		н	
doPosReqTrig	— i	Ir	1		1	
diTgtGen <u>Trig</u>			~			
F						
doPosAvail	f\					
doQueueEmpty			1 1-	_		
doOpSetCompl						
goExecState						
goHeightState	unning) 1 (act	ive)		(active)	0 (fi	111)
0	(full)	0 (fu	II)			

- 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

7.3.5 Timing diagrams for PLC communication *Continued*

Example robot execution control

Task: Pick single items from infeeder and place pallet pattern with two items on outfeeder, control the robot access to work areas by using the robot execution signal.

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

A doPosReqTrig		c I		D C		р
diTgtGenTrig	i					T
diRobotExe <u>c</u>		_	_			+
B doPosAvail		0				
doQueueE <u>mpty</u>						
doOpSetCo <u>mpl</u>						
goExecState	1 (running)	-	_		_	
goHeightState 0 (full)	1 (active)		_			0 (full)
E doPosReqTrig	G		н µ	G	н	
diTgtGenTr <u>ig</u>		_				
diRobotExe <u>c</u>		_			$+ \bot$	
F						
doPosAvail		_		h		
doQueueEmp <u>ty</u>						
doOpSetCompl					<u> </u>	
goExecState 1 (run	nning)				-	1
goHeightState	1 (a	ctive)	0 (full)	active)	0 (full)	

bl

- 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

7.3.5 Timing diagrams for PLC communication *Continued*

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 diSetDefault <u>Height</u> giHeightSelection giDefHeight <u>WaSel</u>				B 2 (Set empty) Select master
C E E E E E E E E E E E E E E E E E E E		F E	F	В
D doPosAvail doQueueEmpty doOpSetCompl goExecState 1 (running) goHeightState 3 (empty) 1 (active)			2 (3 (empty)
doPosReqTrig diTgtGenTrig		J		
H doPosAvail doQueueEmpty doOpSetCompl goExecState				
goHeightState 0 (full)	ve) 0 (full)	1 (active)	1 0 (full)	

Α	Project control, process control
---	----------------------------------

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

7.3.5 Timing diagrams for PLC communication *Continued*

Example flow control

Task: Start a flow, pick single items from infeeder and place pallet pattern on outfeeder, control the robot access to work areas by using the robot execution signal, stop the flow with stop option finish cycle, cancel the rest of the job by resetting the robot execution signal from the outfeeder.

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

A diStart □					
diStop			1 (fin	sh cycle)	
diStopOption		_			Ber avala)
goStatus <u>1 (r</u> 0 (stopp <u>ed)</u>	unning)			2 (stopping a	0 (stopped)
B doPosReqTrig		D I	E D		E
diTgtGenTrig diRobotExec	1				
C doPosAv <u>ail</u> doQueue <u>Empty</u> doOpSetCompl					
goHeightState 0 (full)	1 (active)				0 (full)
F doPosReqTrig diTgtGenTrig diRobotExec		J			
G doPosAv <u>ail</u> doQueue <u>Empty</u>					
doOpSet <u>Compl</u> goHeight <u>State</u> 0 (fi		ctive)	0 (full)		

- 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

This page is intentionally left blank

8.1 Introduction to RAPID reference information

8 **RAPID** reference information

8.1 Introduction to RAPID reference information

Structure of this chapter

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

8 **RAPID** reference information

8.2.1 PmAckTarget - Acknowledge a target

8.2 Instructions

8.2.1 PmAckTarget - Acknowledge a target

Usage	PmAckTarget is used to acknowledge a target.				
	TREATURE IS USED to doknowledge a larget.				
Basic examples					
	IF status = OK THEN				
	PmAckTarget Wa, Target, PM_ACK;				
	ELSE				
	-	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 ${\tt Status}$, can be one of the following:				
	Constant	Description			
	PM_ACK	The target is acknowledged as used.			
	PM_NACK	The target is acknowledged as not used.			
	PM_LOST	If the target is acknowledged as lost.			

Syntax

PmAckTarget

[Wa ':='] < expression (IN) of pm_wadescr > ','
[Target ':='] < expression (IN) of pm_targetdata > ','
[Status ':='] < expression (IN) of pm_acktype > ';'

Related information

For information about	See
The data type pm_wadescr	pm_wadescr - PickMaster work area refer- ence on page 372.

Continues on next page

8.2.1 PmAckTarget - Acknowledge a target *Continued*

For information about	See
The data type pm_targetdata	pm_targetdata - PickMaster target data on page 368.
The data type pm_acktype	pm_acktype - PickMaster target acknowledge type on page 338.

8.2.2 PmCalcArmConf - Calculates the arm configuration

8.2.2 PmCalcArmConf - Calculates the arm configuration

Usage	
	PmCalcArmConf is used to calculate a suitable arm configuration for a robtarget, that is the robconf component of the robtarget. Some switches can be selected to optimize the resulting arm configuration, for example for a robot of a certain type. A maximum and minimum angle can be set up for one axis. The resulting arm configuration will also depend on the initial settings of robconf.
Basic example	<pre>PmCalcArmConf RobTgtPoint,TargetTool,TargetWobj\cf6\MaxAngle:=180 \MinAngle:=-180\SingAreaType:=Act.SingAreaType;</pre>
Arguments	PmCalcArmConf RobTgt Tool Wobj [\cf1] [\cf4] [\cf6] [\TypeB1] [\MaxAngle] [\MinAngle] [\SingAreatype]
RobTgt	
	Robot target
	Data type: robtarget
	The robot target whose arm configuration will be calculated.
Tool	
	Tool
	Data type: tooldata
	The tool used for calculation of the robot arm configuration.
Wobj	
	Work object
	Data type: wobjdata
	The work object (coordinate system) to which the robot position is related.
[\cf1]	
	Data type: switch
	An arm configuration is calculated where the axis 1 angle is limited by the arguments MaxAngle and MinAngle. A solution closer to +45 degrees is preferred for axis 1. A solution close to the input arm configuration is preferred for the other axes.
[\cf4]	
- · -	Data type: switch
	An arm configuration is calculated where the axis 4 angle is limited by the arguments MaxAngle and MinAngle. A solution closer to +45 degrees is preferred for axis 4. A solution close to the input arm configuration is preferred for the other axes.
[\cf6]	
	Data type: switch
	An arm configuration suitable for a 4 axes palletizer robot or a 6 axes bending backwards robot is calculated. The axis 6 angle is limited by the arguments
Continues on next	t page

	٤	3.2.2 PmCalcArmConf - Calculates the arm configuration <i>Continued</i>
		e. A solution closer to +45 degrees is preferred for axis input arm configuration is preferred for the other axes.
[\TypeB1]		
	Data type: switch	
	is limited by the argume	table for a parallel rod robot is calculated. The axis 6 angle nts MaxAngle and MinAngle. A solution closer to +45 axis 6. A solution close to the input arm configuration is xes.
[\MaxAngle]		
	Maximum angle	
	Data type: num	
	-	for one axis. Which axis is decided by the selection of
	cfl, cf4, cf6 or TypeB	1.
[\MinAngle]		
	Minimum angle	
	Data type: num	
	-	for one axis. Which axis is decided by the selection of ${\tt cfl}$,
	cf4,cf6 or TypeB1.	
[\SingAreaType]		
	Interpolation mode	h/po
	• •	on mode to be used with this robtarget. The argument is uration with the LockAx4 interpolation mode.
Error handling		
	-	le errors can be generated. The errors can be handled in system variable ERRNO is set to:
	Error code	Description
	PM_ERR_CALCCONF	Failed to calculate arm configuration. A highly complex arm configuration may cause this error.
	PM_ERR_AXLIM	Failed to calculate axis limit. The axis angle cannot be calcu- lated due to angle limitations.
	PM_ERR_LIM_VALUE	Wrong limitation value. The coordinate is not possible to calculate.
Syntax		
	Instruction	
	-	< 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 >]
```

8.2.2 PmCalcArmConf - Calculates the arm configuration *Continued*

```
[ '\' SingAreaType ':=' < expression (IN) of pm_singareatype >]
    ';'
```

For information about	See
The data type confdata	Technical reference manual - RAPID Instruc- tions, Functions and Data types, section Data types.
The data type robtarget	<i>Technical reference manual - RAPID Instruc- tions, Functions and Data types, section Data types.</i>

8.2.3 PmGetFlow - Get flow to execute

8.2.3 PmGetFlow - Get flow to execute

Usage	
	PmGetFlow is used to wait until any flow reports that it is ready to be executed. The instruction will return two work area references to the work areas that are ready to be executed within a flow. If several flows are ready to be executed, the process behind the instruction will return references of the highest prioritized flow. If the time-out time is not used, the instruction is blocking until any flow is ready to be executed.
Basic examples	
	A basic example of the instruction PMGetFlow is illustrated below.
	PROC OperateSequence()
	<pre>PmGetFlow waInFeeder, waOutFeeder;</pre>
	Operate waInFeeder;
	Operate waOutFeeder; ENDPROC
Arguments	
-	<pre>PmGetFlow PickWa PlaceWa [\MaxTime] [\TimeFlag]</pre>
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 <code>PM_ERR_TIMEOUT</code> . 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.

8.2.3 PmGetFlow - Get flow to execute *Continued*

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 tim 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 tim is exceeded, otherwise it will be set to FALSE.	
More examples		
	More examples of how to use the instruction PmGetFlow are illustrated below.	
	PROC OperateSequence()	
	<pre>PmGetFlow waInFeeder, waOutFeeder \MaxTime:=6 \TimeFlag:=bTimeout;</pre>	
	IF NOT bTimeout THEN	
	Operate waInFeeder;	
	Operate waOutFeeder;	
	ELSE	
	<pre>p1 := CRobT(\Tool:=tool0 \WObj:=wobj0);</pre>	
	MoveL RelTool(p1,100,0,0), v100, fine, tool0;	
	MoveL pl, v100, fine, tool0;	
	ENDIF	
	ENDPROC	

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable ERRNO will be set to:

Error code	Description
PM_ERR_TIMEOUT	No flow was ready to be executed within the time-out time.

Syntax

```
PmGetFlow
[ PickWa ':=' ] < expression (VAR) of pm_wadescr > ','
[ PlaceWa ':=' ] < expression (VAR) of pm_wadescr >
[ '\' MaxTime ':=' < expression (IN) of num > ',']
[ '\' TimeFlag ':=' < variable (VAR) of bool >] ';'
```

For information about	See
The data type pm_wadescr	pm_wadescr - PickMaster work area reference on page 372
The Robot Flow Configuration	Flow on page 202

8.2.4 PmGetFlowInfo - Get information about a specific flow

Usage	
-	PmGetFlowInfo gets information about a flow. The flow must be in the started
	project.
Basic examples	
	A basic example of the instruction PmGetFlowInfo is illustrated below.
	TRAP TrapStartFlow
	VAR pm_flowinfo FlowInfo;
	VAR num FlowSelection;
	! Get info from selected flow
	<pre>PmGetFlowInfo FlowSelection,FlowInfo;</pre>
	ENDTRAP
Arguments	
-	PmGetFlowInfo SelectionNumber Name FlowInfo
SelectionNumber	
	Data type: num
	The number that maps a specific flow with its signal value.
Jame	
	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 fails with a recoverable error if the flow cannot be found. All other
	errors are considered to be fatal.
More examples	
	Another example of how to use the instruction PMGetFlowInfo is illustrated below
	TRAP TrapStartFlow
	VAR pm_flowinfo FlowInfo;
	VAR num FlowSelection;
	<pre>FlowSelection:=1;</pre>
	! Get info from selected flow
	<pre>PmGetFlowInfo FlowSelection,FlowInfo;</pre>
	! 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;
	Continues on next pag

8.2.4 PmGetFlowInfo - Get information about a specific flow *Continued*

ENDIF ENDTRAP

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable ERRNO will be set to:

Error code	Description
PM_ERR_FLOW_NOT_FOUND	No flow was found with this selection number or name.
PM_ERR_NO_RUNNING_PROJECT	No running project.

Syntax

```
PmGetFlowInfo
```

```
[SelectionNumber ':=' ] < expression (IN) of num > ','
[Name ':=' ] < expression (IN) of string > ','
[FlowInfo ':=' ] < expression (VAR) of pm_flowinfo > ';'
```

For information about	See
	pm_flowinfo - PickMaster flow information on page 347.

8.2.5 PmGetLastWa - Get last used work area

8.2.5 PmGetLastWa - Get last used work area

Usage	
	PmGetLastWa gets the last used work area. The work area must previous have
	been set by the instruction PmSetLastWa.
Basic examples	
	A basic example of the instruction PmGetLastWa is illustrated below.
	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	<u> </u>
	All errors are considered to be fatal.
Syntax	
	PmGetLastWa
	[Workarea ':='] < expression (VAR) of pm_wadescr > ';'

For information about	See
The data type pm_wadescr	pm_wadescr - PickMaster work area refer- ence on page 372
The instruction PmSetLastWa	PmSetLastWa - Set last used work area on page 307
The instruction PmGetPathHeight	PmGetPathHeight - Get a safe path height for an intermediate movement on page 325

8.2.6 PmGetOperation - Get operation from a work area

8.2.6 PmGetOperation - Get operation from a work area

Usage		
	PmGetOperation is	used to get operation data from a work area.
Basic example		
·	720.753],	<pre>wInfeeder2 := [FALSE,TRUE,"", [[2180.65,1430.22,- [0.00104,0.00130,0.00039,1.000000]], [1,0,0,0]]];</pre>
	VAR pm_wadescr	
	VAR pm_operatio	
		vInfeeder2, PickWa;
	PmGetOperation	
	Get operation data to	r the work area using work object data wInfeeder2.
Arguments	PmGetOperation	Wa Operation [\MaxTime] [\TimeFlag]
Wa		
	Data type: pm_wades	cr
	Contains a reference	to a work area.
Operation		
-	Data type: pm_opera	tiondata
	Operation data that is	s fetched from a work area.
[\MaxTime]		
[(Haniime]	Maximum Time	
	Data type: num	
	runs out before the co	of waiting time permitted, expressed in seconds. If this time ondition is met, the error handler will be called, if there is one, M_ERR_TIMEOUT. If there is no error handler, the execution
[\TimeFlag]		
	Timeout Flag	
	Data type: num	
	waiting time runs out the instruction, it is no	that contains the value TRUE if the maximum permitted before the condition is met. If this argument is included in ot considered an error if the maximum time runs out. This f the MaxTime argument is not included in the instruction.
Error handling	•	rable errors can be generated. The errors can be handled in The system variable ERRNO will be set to:
	Error code	Description
	PM_ERR_TIMEOUT	Nopm_operationdata could be fetched within the time out time.
		No pit_operationata could be letened within the time out time.

8.2.6 PmGetOperation - Get operation from a work area Continued

Syntax

PmGetOperation

```
[ Wa ':=' ] < expression (IN) of pm_wadescr > ','
[ Operation ':=' ] < expression (INOUT) of pm_operationdata >
[ '\' MaxTime ':=' < expression (IN) of num >]
[ '\' TimeFlag ':=' < variable (VAR) of bool >] ';'
```

For information about	See
The data type pm_wadescr	pm_wadescr - PickMaster work area refer- ence on page 372.
The data type pm_operationdata	pm_operationdata - PickMaster operation data on page 354.

8.2.7 PmGetProjectInfo - Get information about a specific project

8.2.7 PmGetProjectInfo - Get information about a specific project

Usage	PmGetProjectInfo gets information about a project. The project must be transferred to the controller.
Basic examples	A basic example of the instruction PmGetProjectInfo is illustrated below. 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 <i>Project</i> <i>Manager on page 108</i> .
Name	Data type: string The name of a transferred project.
ProjectInfo	Data type: pm_projectinfo Variable that holds the information about the project.
Program execution	The program will fail with a recoverable error if the project cannot be found. All other errorsare considered to be fatal.
More examples	Another example of how to use the instruction PMGetProjectInfo is illustrated below. 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;

```
8.2.7 PmGetProjectInfo - Get information about a specific project 
Continued
```

```
WHILE TRUE DO
  ! Execute the main routine in the selected project.
    %"PmMain:Main"%;
ENDWHILE
ERROR
    IF ERRNO=PM_ERR_PROJ_NOT_FOUND THEN
        ! There is no project mapped to the selection value
        TRYNEXT;
    ENDIF
ENDPROC
```

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable ERRNO will be set to:

Error code	Description
PM_ERR_PROJ_NOT_FOUND	No project was found with this selection number or name.

Syntax

PmGetProjectInfo

```
[SelectionNumber ':=' ] < expression (IN) of num> ','
[Name ':=' ] < expression (IN) of string > ','
[ProjectInfo ':=' ] < expression (VAR) of pm_ projectinfo > ';'
```

For information about	See
	pm_projectinfo - PickMaster project informa- tion on page 357.

8.2.8 PmGetWaByWobj - Get a reference to a work area using a work object data

8.2.8 PmGetWaByWobj - Get a reference to a work area using a work object data

			
Usage	DmCatNa RyMabi acts the ref	erence 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 wInfeed 220.753], [0.001	er1 := [FALSE,TRUE,"",[[2180.65,1430.22,-	
		1.00000]],[[0,0,0],[1,0,0,0]]];	
	VAR pm_wadescr PickWa	;	
	PmGetWaByWobj wInfeed	erl, 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 ref work object.	fer to the work area that corresponds to the provided	
Error handling			
-	•	rors can be generated. The errors can be handled in em variable ERRNO will be set to:	
	Error code	Description	
	PM_ERR_WOBJ	No work area has reference to the work object data used.	
Syntax			
,	PmGetWaByWobj		
		sistent (PERS) of wobjdata > ','	
	[Wa ':='] < expre	<pre>ssion (INOUT) of pm_wadescr > ';'</pre>	
Related information			
	For information about	See	
	The data type pm_wadescr	pm_wadescr - PickMaster work area reference on page 372	
	The data type wobjdata	Technical reference manual - RAPID Instructions, Func-	

tions and Data types, section Data types

8.2.9 PmGetWaInfo - Get information about a specific work area

8.2.9 PmGetWaInfo - Get information about a specific work area

Jsage		
	-	a about a work area. The work area must be in the n can be used in an external user interface or as a for from a selection number
Basic examples		
	Basic examples of the instruction	on PmGetWaInfo are illustrated below.
	VAR pm_wainfo WaInfo;	
	VAR num WaSelection:=1;	
	! Get info from selecte	
	PmGetWaInfo WaSelectior	n,WaInfo;
rguments		
-	PmGetWaInfo SelectionNu	umber WorkArea FlowInfo
electionNumber		
	Data type: num	
	The number that maps a specif	ic work area with its signal value.
orkArea		
01111100	Data type: pm_wadescr	
	A valid descriptor in a started project. The descriptor could be collected from	
	PmGetFlow or PmGetWaByWo	
lowInfo		
	Data type: pm_wainfo	
	Variable that holds the information	tion about the work area.
Program execution		
Togram execution	The program will fail with a reco	overable error if the work area cannot be found. A
	other errors are considered to l	
rror handling		
	The following recoverable error	rs can be generated. The errors can be handled i
	-	n variable ERRNO will be set to:
	Error code	Description
	PM_ERR_WA_NOT_FOUND	No work area was found with this selection number on name.
	PM_ERR_NO_RUNNING_PROJECT	No running project.
ivntax		
Syntax	PmGetWaInfo	
Syntax	PmGetWaInfo] < expression (IN) of num > ','

8.2.9 PmGetWaInfo - Get information about a specific work area *Continued*

For information about	See
The data type pm_wainfo	pm_wainfo - PickMaster Work Area informa- tion on page 373
The data type pm_wadescr	pm_wadescr - PickMaster work area refer- ence on page 372
The instruction PmGetFlow	PmGetFlow - Get flow to execute on page 293
The instruction PmGetWaByWobj	PmGetWaByWobj - Get a reference to a work area using a work object data on page 302

8.2.10 PmSearchAdjust - Adjust number of remaining layers

rches. ssed
rches.
ches.
ches.
ches.
of the
of the
rection
maining picking

ledge of the search target and fetching next operation.

8.2.10 PmSearchAdjust - Adjust number of remaining layers *Continued*

Error code	Description
PM_ERR_PALLET_EMPTY	The detected height of the pallet pattern or format indicates missing parts. The error is recovered through acknowledge of the search target, trigger the Redo Search signal for the work area and fetching next operation.

Limitations

The instruction may only be used after an action containing a SearchL movement has been fetched with PmGetTgtAction and before the corresponding target has been acknowledged.

Predefined data

The search type, used in argument SearchType can be one of the following:

Constant	Value	Description
PM_SEARCH_X	0	Search was performed in the x direction of the work object.
PM_SEARCH_Y	1	Search was performed in the y direction of the work object.
PM_SEARCH_Z	2	Search was performed in the z direction of the work object.

Syntax

PmSearchAdjust

[WorkArea ':='] < expression (IN) of pm_wadescr > ','

```
[ SearchType ':=' ] < expression (IN) of pm_searchtype > ','
```

[SearchPos ':='] < expression (IN) of num > ';'

For information about	See
Stack search	Stack search on page 191.
Format frame versus work object in format operation set	Format frame versus work object in format operation set on page 388.
Pallet pattern versus work object in pallet pattern operation set	Pallet pattern frame versus work object in pallet pattern operation set on page 390.
The data type pm_wadescr	pm_wadescr - PickMaster work area refer- ence on page 372.
The data type pm_searchtype	pm_searchtype - PickMaster stack search type on page 363.

8.2.11 PmSetLastWa - Set last used work area

8.2.11 PmSetLastWa - Set last used work area

Usage	
	PmSetLastWa sets the last used work area. Use the instruction PmGetLastWa to
	get the work area.
Basic examples	
	Basic examples of the instruction instruction are illustrated below.
	VAR pm_wadescr WorkArea;
	! Set last used work area
	PmSetLastWa WorkArea;
Arguments	
-	PmSetLastWa Workarea
Workarea	
	Data type: pm_wadescr
	A descriptor to the last used work area.
Program execution	
-	All errors are considered to be fatal.
Syntax	
-	PmSetLastWa
	[Workarea ':='] < expression (VAR) of pm_wadescr > ';'

For information about	See
The data type pm_wadescr	pm_wadescr - PickMaster work area refer- ence on page 372
The instruction PmGetLastWa	PmGetLastWa - Get last used work area on page 297
The instruction PmGetPathHeight	PmGetPathHeight - Get a safe path height for an intermediate movement on page 325

8.2.12 PmSetRecoverAction - Set flow recover action

8.2.12 PmSetRecoverAction - Set flow recover action

Usage	PmSetRecoverAction sets flow recover action before starting a flow in error state. This instruction must be used before starting a flow with use of the IO interface if the flow is in error state. The instruction also returns information that can be used in an event log message. This message describes circumstances for restarting
	with selected recover action.
Basic examples	
•	Basic examples of the instruction PmSetRecoverAction are illustrated below.
	VAR pm_flowinfo FlowInfo;
	<pre>PmSetRecoverAction FlowInfo.Name,PM_RECOVER_REDO_LAST_PICK;</pre>
Arguments	
Ū	<pre>PmSetRecoverAction Name \WorkArea RecoverAction \EventId \Argument1 \Argument2 \Argument3 \Argument4</pre>
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. This is mandatory if using recover action Restart layer or Terminate job/Next pallet, where it is recommended to select the master work area. This is not required for recover action Continue and Redo last pick.
RecoverAction	
	Data type: num
	The recover action that is performed at next flow start.
[\EventId]	
	Data type: num
	The event message number in process domain that creates a message for the chosen recover action.
[\Argument1]	
	Data type: errstr
	The first argument to the event log message, one space if not used.
[\Argument2]	
	Data type: errstr
	The sencond argument to the event log message, one space if not used.
[\Argument3]	
[\ALGuments]	Data type: errstr

8.2.12 PmSetRecoverAction - Set flow recover action Continued

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

[\Argument4]

Data type: errstr

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

Program execution

The program will fail with a recoverable error with recover action:

- PM_RECOVER_REDO_LAYER or PM_RECOVER_NEXT_PALLET without a valid work area.
- A recover action not in range.
- PM_RECOVER_REDO_LAST_PICK when nothing is picked.

All other errors are considered to be fatal.

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable ERRNO will be set to:

Error code	Description
PM_ERR_WORKAREA_EXPECTED	This recover action demands a work area.
PM_ERR_NOT_VALID_RECOVER_ACTION	The recover action is not one of the suppor- ted.
PM_ERR_REDO_LAST_PICK_REJECTED	The redo last pick recover action is rejected, no products picked.
PM_ERR_NO_RUNNING_PROJECT	No running project.

Predefined messages

There are predefined event messages in the process domain for describing what to do before the flow start with the chosen recover action.

```
<Message number="2393">
 Flow recover with redo last pick
 The Flow <Flow name> will redo last unfinished operation at next
       flow start.
 Verify that:
   The tool is empty
   Products from last operation are restored on <WorkArea name>
   The reason for the stop is solved.
</Message>
<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>
```

309

8.2.12 PmSetRecoverAction - Set flow recover action *Continued*

```
<WorkArea name/Number of products/Layer number>
</Message>
<Message number="2395">
 Flow recover with restart layer
 The Flow <Flow name> will restart from beginning of layer <layer
       number> on WorkArea <WorkArea name> at next flow start.
 Verify that:
   The reason for the stop is solved
   The tool is empty
   Following WorkAreas are empty:
      <WorkArea name>
     <WorkArea name>
</Message>
<Message number="2396">
 Flow recover with next pallet
 The Flow <Flow name> will restart from beginning of next pallet
       on WorkArea <WorkArea name> at next flow start.
 Verify that:
   The reason for the stop is solved.
   The tool is empty
   Following WorkAreas are empty:
     <WorkArea name>
      <WorkArea name>
     <WorkArea name>
</Message>
<Message number="2397">
 Flow recover with redo last pick
 The Flow <Flow name> will redo last unfinished operation at next
       flow start.
 Verify that:
   The tool is empty
   New products can be supplied on <WorkArea name>
   The reason for the stop is solved.
</Message>
```

Predefined data

Constant	Value	Description
PM_RECOVER_CONTINUE_OPERATION	1	The pick-place operation will contin- ue from where it was stopped.
PM_RECOVER_REDO_LAYER	2	The pick-place operation repeats last layer.
PM_RECOVER_NEXT_PALLET	3	The current job is terminated and the flow is prepared to continue with the next pallet.
PM_RECOVER_REDO_LAST_PICK	4	The pick-place operation repeats the last pick operation. This recover ac- tion is valid only when the robot has picked up items that has not been placed yet.

8.2.12 PmSetRecoverAction - Set flow recover action Continued

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 > ','
```

For information about	See
The instruction PmStartFlow	PmStartFlow - Starts a specific flow on page 312

8.2.13 PmStartFlow - Starts a specific flow

8.2.13 PmStartFlow - Starts a specific flow

Usage			
	PmStartFlow starts a flow. The flow mu	ust be in t	he started project.
Basic examples			
	A basic example of the instruction ${\tt PmSt}$	artFlow	is illustrated below.
	TRAP TrapStartFlow		
	VAR pm_flowinfo FlowInfo;		
	VAR num FlowSelection;		
	! Get info from selected flow PmGetFlowInfo FlowSelection,		
	! Start the selected flow	FIOWINIO	,
	PmStartFlow FlowInfo.Name;		
	ENDTRAP		
Arguments			
	PmStartFlow Name		
Jame			
	Data type: string		
	Variable that refers to a flow in a started	proiect.	
Program execution			
Program execution			project is running. All other
Program execution	The program will fail with a recoverable	error if no	
Program execution		error if no	
Program execution Error handling	The program will fail with a recoverable	error if no	
	The program will fail with a recoverable	error if no s wrong f	low name.
	The program will fail with a recoverable errors are considered to be fatal, such a	error if no s wrong f generate	low name. d. The errors can be handled i
	The program will fail with a recoverable errors are considered to be fatal, such a The following recoverable errors can be	error if no s wrong f generate	low name. d. The errors can be handled i will be set to:
	The program will fail with a recoverable errors are considered to be fatal, such a The following recoverable errors can be an ERROR handler. The system variable	error if no s wrong f generate ERRNO Descripti	low name. d. The errors can be handled i will be set to:
	The program will fail with a recoverable errors are considered to be fatal, such a The following recoverable errors can be an ERROR handler. The system variable Error code	error if no s wrong f generate ERRNO Descripti No runnir	low name. d. The errors can be handled i will be set to: on ng project. flow in error state without setting
Error handling	The program will fail with a recoverable errors are considered to be fatal, such a The following recoverable errors can be an ERROR handler. The system variable Error code PM_ERR_NO_RUNNING_PROJECT	error if no s wrong f generate ERRNO Descripti No runnir Starting a	low name. d. The errors can be handled i will be set to: on ng project. flow in error state without setting
	The program will fail with a recoverable errors are considered to be fatal, such a The following recoverable errors can be an ERROR handler. The system variable Error code PM_ERR_NO_RUNNING_PROJECT PM_ERR_WRONG_FLOW_STATE	error if no s wrong f generate ERRNO Descripti No runnir Starting a a recover	low name. d. The errors can be handled i will be set to: on ng project. flow in error state without setting action.
Error handling	The program will fail with a recoverable errors are considered to be fatal, such a The following recoverable errors can be an ERROR handler. The system variable Error code PM_ERR_NO_RUNNING_PROJECT PM_ERR_WRONG_FLOW_STATE There are predefined constants for the f	error if no s wrong f generate ERRNO Descripti No runnir Starting a a recover	low name. d. The errors can be handled i will be set to: on ng project. flow in error state without setting action.
Error handling	The program will fail with a recoverable errors are considered to be fatal, such a The following recoverable errors can be an ERROR handler. The system variable Error code PM_ERR_NO_RUNNING_PROJECT PM_ERR_WRONG_FLOW_STATE	error if no s wrong f generate ERRNO Descripti No runnir Starting a a recover	low name. d. The errors can be handled i will be set to: on ng project. flow in error state without setting action. s. The constants are used for low editor. See <i>Flow on page 20</i> .
Error handling	The program will fail with a recoverable errors are considered to be fatal, such a The following recoverable errors can be an ERROR handler. The system variable Error code PM_ERR_NO_RUNNING_PROJECT PM_ERR_WRONG_FLOW_STATE There are predefined constants for the f setting values on flow status signals, confi	error if no s wrong f generate ERRNO Descripti No runnir Starting a a recover	low name. d. The errors can be handled i will be set to: on ng project. flow in error state without setting action.
Error handling	The program will fail with a recoverable errors are considered to be fatal, such a The following recoverable errors can be an ERROR handler. The system variable Error code PM_ERR_NO_RUNNING_PROJECT PM_ERR_WRONG_FLOW_STATE There are predefined constants for the f setting values on flow status signals, confi	error if no s wrong f generate ERRNO Descripti No runnin Starting a a recover	low name. d. The errors can be handled i will be set to: on ng project. flow in error state without setting action. s. The constants are used for low editor. See <i>Flow on page 20</i>
Error handling	The program will fail with a recoverable errors are considered to be fatal, such a The following recoverable errors can be an ERROR handler. The system variable Error code PM_ERR_NO_RUNNING_PROJECT PM_ERR_WRONG_FLOW_STATE There are predefined constants for the f setting values on flow status signals, configuration	error if no s wrong f generate ERRNO Descripti No runnir Starting a a recover	d. The errors can be handled i will be set to: on ng project. flow in error state without setting action. s. The constants are used for low editor. See <i>Flow on page 20</i> . Description
Error handling	The program will fail with a recoverable errors are considered to be fatal, such a errors are considered to be fatal, such a a the following recoverable errors can be an ERROR handler. The system variable error code error code error code error code error erro	error if no s wrong fi generate e ERRNO Descripti No runnir Starting a a recover	low name. d. The errors can be handled i will be set to: on ng project. flow in error state without setting action. s. The constants are used for low editor. See <i>Flow on page 20</i> Description The flow is stopped

8.2.13 PmStartFlow - Starts a specific flow Continued

Constant	Value	Description
PM_FLOW_STOPPING_AFTER_PALLET	4	The flow is stopping after current pallet is finished
PM_FLOW_ERROR	5	The flow is in error state

Syntax

PmStartFlow

[FlowSelector ':='] < expression (IN) of string > ';'

For information about	See
The instruction PmGetFlowInfo	PmGetFlowInfo - Get information about a specific flow on page 295.
The instruction PmStopFlow	PmStopFlow - Stop a specific flow on page 315.
The instruction PmSetRecoverAction	PmSetRecoverAction - Set flow recover ac- tion on page 308
The data type pm_flowinfo	pm_flowinfo - PickMaster flow information on page 347.

8.2.14 PmStartProj - Start a PickMaster project

8.2.14 PmStartProj - Start a PickMaster project

Usage	Dmetart Droj etarte a Dia	kMaster pro	iect When this instruction is executed th
	PmStartProj starts a PickMaster project. When this instruction is executed, th project setup is read and all PickMaster internal parts of the project are initialized. The time of execution depends on the size of the project.		
Basic example			
	PmStartProj "MyPMP:		DJECT_STARTING THEN
			US=PM_PROJECT_RUNNING;
Arguments	PmStartProj Name [Signall	
		(biginai)	
Name	Data type: string		
	The name of the project to	start	
	The name of the project to	Juli.	
[\Signal]			
	Data type: signalgo The signal that shows the		
Predefined data			
Predefined data	There are predefined consused for setting values on	stants for the Signal (proje	e status of the project. Those constants an ect status signal) and the installed persiste DJECT_STATUS can be accessed from RAP
Predefined data	There are predefined consused for setting values on variable PM_PROJECT_STA	stants for the Signal (proje	e status of the project. Those constants a ect status signal) and the installed persiste
Predefined data	There are predefined consused for setting values on variable PM_PROJECT_STAprogram.	stants for the Signal (proje TUS. PM_PRC	e status of the project. Those constants a ect status signal) and the installed persiste DJECT_STATUS can be accessed from RAP
Predefined data	There are predefined consused for setting values on variable PM_PROJECT_STA program.	stants for the Signal (proje TUS. PM_PRC	e status of the project. Those constants and ect status signal) and the installed persiste DJECT_STATUS can be accessed from RAP
Predefined data	There are predefined consused for setting values on variable PM_PROJECT_STAprogram.	stants for the Signal (proje TUS. PM_PRC Value 0	e status of the project. Those constants an ect status signal) and the installed persiste DJECT_STATUS can be accessed from RAP Description The project is stopped.
Predefined data	There are predefined consused for setting values on variable PM_PROJECT_STAprogram. Constant PM_PROJECT_STOPPED PM_PROJECT_STOPPING	stants for the Signal (proje TUS. PM_PRC Value 0 1	e status of the project. Those constants an ect status signal) and the installed persiste DJECT_STATUS can be accessed from RAP Description The project is stopped. The project is about to stop.
Predefined data	There are predefined consused for setting values on variable PM_PROJECT_STAprogram. Constant PM_PROJECT_STOPPED PM_PROJECT_STOPPING PM_PROJECT_STARTING	stants for the Signal (proje TUS. PM_PRC Value 0 1 2	e status of the project. Those constants an ect status signal) and the installed persiste DJECT_STATUS can be accessed from RAP Description The project is stopped. The project is about to stop. The project is starting up.
	There are predefined consused for setting values on variable PM_PROJECT_STAprogram. Constant PM_PROJECT_STOPPED PM_PROJECT_STOPPING PM_PROJECT_STARTING PM_PROJECT_RUNNING PM_PROJECT_ERROR	stants for the Signal (proje TUS. PM_PRC 0 1 2 3 4 4	e status of the project. Those constants at ect status signal) and the installed persiste DJECT_STATUS can be accessed from RAP Description The project is stopped. The project is about to stop. The project is starting up. The project is running.
Syntax	There are predefined consused for setting values on variable PM_PROJECT_STAprogram. Constant PM_PROJECT_STOPPED PM_PROJECT_STOPPING PM_PROJECT_STARTING PM_PROJECT_RUNNING PM_PROJECT_ERROR PMStartProj ['\' Signal ':=	stants for the Signal (proje TUS. PM_PRC 0 1 2 3 4 4	e status of the project. Those constants an ect status signal) and the installed persiste DJECT_STATUS can be accessed from RAP Description The project is stopped. The project is about to stop. The project is starting up. The project is running. The project is in error state.
Predefined data Syntax	There are predefined consused for setting values on variable PM_PROJECT_STAprogram. Constant PM_PROJECT_STOPPED PM_PROJECT_STOPPING PM_PROJECT_STARTING PM_PROJECT_RUNNING PM_PROJECT_ERROR PMStartProj ['\' Signal ':=	stants for the Signal (proje TUS. PM_PRC 0 1 2 3 4 4	e status of the project. Those constants a ect status signal) and the installed persiste DJECT_STATUS can be accessed from RAP Description The project is stopped. The project is about to stop. The project is starting up. The project is running. The project is in error state.

For information about	See
The instruction PmStopProj.	PmStopProj - Stop current project on page 317.

8.2.15 PmStopFlow - Stop a specific flow

Usage		
	PmStopFlow stops a flow. The flow mu	st be in the started project.
Basic examples		
	A basic example of the instruction PmSt	copFlow is illustrated below.
	TRAP TrapStopFlow	
	VAR pm_flowinfo FlowInfo;	
	VAR num FlowSelection;	
	VAR num StopOption;	
	! Get info from selected flo	W
	PmGetFlowInfo FlowSelection,	FlowInfo;
	! Stop the selected flow	
	PmStopFlow FlowInfo.Name, St	copOption;
	ENDTRAP	
Arguments		
	PmStopFlow Name StopOption	
Name		
	Data type: string	
	Variable that refers to a flow in a started	d project.
StopOption		
	Data type: num	
	Variable that specifies different stop be	havior.
Program execution		
· · · g · · · · · · · · · · · · · ·	The program will fail with a recoverable	error if no project is running All other
	errors are considered to be fatal, such a	
	StopOption.	ao mong non namo or mong valuo on
	500000000000000000000000000000000000000	
Error handling		
	The following recoverable errors can be	e generated. The errors can be handled ir
	an ERROR handler. The system variable	0
	Error code	Description
	PM_ERR_NO_RUNNING_PROJECT	No running project.

8.2.15 PmStopFlow - Stop a specific flow

8.2.15 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 202*.

Constant	Value	Description
PM_FLOW_STOPPED	0	The flow is stopped
PM_FLOW_RUNNING	1	The flow is running
PM_FLOW_STOPPING_AFTER_CYCLE	2	The flow is stopping after current cycle is finished
PM_FLOW_STOPPING_AFTER_LAYER	3	The flow is stopping after current layer is finished
PM_FLOW_STOPPING_AFTER_PALLET	4	The flow is stopping after current pallet is finished
PM_FLOW_ERROR	5	The flow is in error state

StopOption constants

The following constants are predefined for StopOption.

Constant	Value	Description
PM_FLOW_STOP_IMMEDIATELY	0	The flow will stop immediately and a flow recovery action must be selected before restarting
PM_FLOW_FINISH_CYCLE	1	The flow will finish current cycle before stopping
PM_FLOW_FINISH_LAYER	2	The flow will finish current layer before stopping
PM_FLOW_FINISH_PALLET	3	The flow will finish current pallet before stopping

Syntax

PmStopFlow

```
[Name ':=' ] < expression (IN) of string > ','
[StopOption ':=' ] < expression (IN) of num > ';'
```

For information about	See
The instruction PmGetFlowInfo	PmGetProjectInfo - Get information about a specific project on page 300.
The instruction PmStartFlow	PmStartFlow - Starts a specific flow on page 312.
The data type pm_flowinfo	pm_flowinfo - PickMaster flow information on page 347.

8.2.16 PmStopProj - Stop current project

8.2.16 PmStopProj - Stop current project

Usage			
	PmStopProj stops the active	PickMaster pro	oject.
Basic example			
	PmStopProj;		
Syntax			
	PmStopProj;		
Related information			
	For information about	See	

For information about	See
The instruction PmStartProj	PmStartProj - Start a PickMaster project on page 314.

8.2.17 PmWaitProjStart - Wait for any active project

8.2.17 PmWaitProjStart - Wait for any active project

Usage			
5	be used with a timeout; t	the instruction th	ny project is running. The instruction can ien waits the timeout time before giving ntil any project is started from any client.
Basic example	PmWaitProjStart \	Marrima ·- E·	
			econds to see if the project is started
Arguments	PmWaitProjStart [\MaxTime] [\Ti	meFlag]
[\MaxTime]			
	Maximum Time		
	Data type: num		
	runs out before the cond	ition is met, the e	mitted, expressed in seconds. If this time error handler will be called, if there is one, If there is no error handler, the execution
[\TimeFlag]			
	Timeout Flag		
	Data type: bool		
	waiting time runs out bef the instruction, it is not c	fore the conditio considered an er	alue TRUE if the maximum permitted n is met. If this argument is included in ror if the maximum time runs out. This ment is not included in the instruction.
Error handling			
C C	The following recoverabl an ERROR handler. The		generated. The errors can be handled in ERRNO will be set to:
	Error code	Description	
	PM_ERR_TIMEOUT	The project was	not started within the time out time
Syntax			
Oymax	PmWaitProjStart		
			.on (IN) of num >] .e (VAR) of bool >] ';'
Related information			
	For information about		See
	The instruction Destant Da		PmStartProj - Start a PickMaster project on

For information about	See
The instruction PmStartProj	PmStartProj - Start a PickMaster project on page 314.

8.3 Functions

8.3.1 PmCalcIntermid - Calculate intermediate position

Usage	
	This function calculates an intermediate position between two targets. If no limitations are set, the calculated position is a part of all axis movements.
Basic example	InterMid:=PmCalcIntermid(p10, tool2, wobj2, p20, tool1, wobj1, 0.4
	\MaxAngle:=180 \MinAngle:= -180 \AngleLimAx6 \MaxY:=2200 \MinY:=-2200 \MinZ:=670 \FromWa:= LastWorkArea \ToWa:=WorkArea);
	<pre>MoveJ InterMid,v1000,z200,tool0\Wobj:=wobj0;</pre>
Return value	
	Data type: robtarget
	The function will return a robtarget expressed in wobj0 and tool0.
Arguments	
	<pre>PmCalcIntermid (StartRobTgt StartTool StartWobj EndRobTgt EndTool EndWobj InterMidPart [\MaxAngle] [\MinAngle] [\AngleLimAx1] [\AngleLimAx4] [\AngleLimAx6] [\MaxX] [\MinX] [\MaxY] [\MinY] [\MaxZ] [\MinZ] [\MaxRadius] [\MinRadius] [\FromWa] [\ToWa] [\LimitRobBase] [\LimitWorld])</pre>
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 tuma usa bida ba
	Data type: wobjdata
	The work object that is used at the end point.
	Continues on payt page

8.3.1 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]	
[(AIIGTEITIIAAO]	Data type: num
	Limit angle on axis 6.
[\MaxX]	
	Data type: num
	Maximum allowed X-value on intermediate position.
[\MinX]	
	Data type: num
	Minimum allowed X-value on intermediate position.
[\MaxY]	
	Data type: num
	Maximum allowed Y-value on intermediate position.
[\MinY]	
	Data type: num
	Minimum allowed Y-value on intermediate position.
[\MaxZ]	
	Data type: num
	Maximum allowed Z-value on intermediate position.
[\MinZ]	Data type: num
	Minimum allowed Z-value on intermediate position.
Continues on next pa	-
320	Operating manual - Palletizing PowerPac

3HAC042340-001 Revision: R

8.3.1 PmCalcIntermid - Calculate intermediate position Continued

[\MaxRadius]		
	Data type: num	
	Maximum allowed radius	of the intermediate position in the XY plane.
	Note	
	If MaxRadius is specifie	ed, limitations on X and Y will not be considered.
[\MinRadius]	_	
	Data type: num	
	Minimum allowed radius	of the intermediate position in the XY plane.
	Note	
	If MinRadius is specifie	ed, limitations on X and Y will not be considered.
[\LimitRobBase]		
	Data type: switch	
	Limitations on X, Y and Z	are defined in the base frame of the robot.
[\LimitWorld]		
	Data type: switch	
		are defined in the world frame, that is, wobj0. If this switch tions will be made in the robot base frame as default.
[\FromWa]		
	Data type: pm_wadescr	
	movement. The presence	ea the robot was operating before the intermediate e of the reference will not affect the result of the function. ed to improve error handling of the function.
[\ToWa]		
	Data type: pm_wadescr	
		ork area the robot will operate. The presence of the he result of the function. The reference is only used to f the function.
Error handling		
	•	e errors can be generated. The errors can be handled in system variable ERRNO will be set to:
	Error code	Description
	PM_ERR_CALCCONF	Failed to calculate arm configuration. A highly complex arm configuration can cause this error.
	PM_ERR_AXLIM	Failed to calculate axis limit. The axis angle cannot be calcu- lated because of the angle limitations.
	PM_ERR_LIM_VALUE	Wrong limitation value. The coordinate is not possible to calcu- late.

Continues on next page

8.3.1 PmCalcIntermid - Calculate intermediate position *Continued*

Error code	Description
PM_ERR_PART_VAL	The value of the InterMidPart is not valid.

Syntax

PmCalcIntermid '('
[StartRobTgt ':='] < expression (IN) of robtarget > ','
[StartTool ':='] < expression (IN) of tooldata > ','
[StartWobj ':='] < expression (IN) of wobjdata > ','
[EndRobTgt ':='] < expression (IN) of robtarget > ','
[EndTool ':='] < expression (IN) of tooldata > ','
[EndWobj ':='] < expression (IN) of wobjdata > ','
[InterMidPart ':='] < expression (IN) of num >
['\' MaxAngle ':=' < expression (IN) of num >
['\' MinAngle ':=' < expression (IN) of num >]
[['\'AngleLimAx1] ['\'AngleLimAx4] ['\'AngleLimAx6]
]
['\' MaxX := < expression (IN) of num >]
['\' MinX := < expression (IN) of num >]
$[' \ MaxY := < expression (IN) of num >]$
['\' MinY := < expression (IN) of num >]
['\' MaxZ := < expression (IN) of num >]
['\' MinZ := < expression (IN) of num >]
['\' MaxRadius := < expression (IN) of num >]
['\' MinRadius := < expression (IN) of num >]
['\' FromWa := < expression (IN) of pm_wadescr >]
['\' ToWa := < expression (IN) of pm_wadescr >]')'

A function with a return value of the data type robtarget.

8.3.2 PmGetEvent - Get events for an action

8.3.2 PmGetEvent - Get events for an action

Usage	
	PmGetEvent is used to get an event for an action on a work area.
Basic examples	
	VAR pm_eventdata Event;
	<pre>ArrSize := Dim(TriggArr,1);</pre>
	WHILE PmGetEvent(Wa, Tgt.TargetHandle, Act.ActionHandle, Event) AND i <= ArrSize DO
	TEST Event.Type
	CASE PM_EVENT_PROC:
	<pre>TriggEquip TriggArr{i}, Event.Dist, Event.Time,</pre>
	CASE PM_EVENT_DO:
	GetDataVal Event.SignalName,doSignal;
	<pre>TriggEquip TriggArr{i}, Event.Dist, Event.Time,</pre>
	CASE PM_EVENT_GO:
	GetDataVal Event.SignalName,goSignal;
	<pre>TriggEquip TriggArr{i}, Event.Dist, Event.Time,</pre>
	ENDTEST
	Incr i;
	ENDWHILE
	TEST Act.NumOfEvents
	CASE 0:
	MoveL Tgt.RobTgtPoint, Act.Speed, Act.Zone, curr_Tool \WObj:=curr_WObj;
	CASE 1:
	<pre>TriggL Tgt.RobTgtPoint, Act.Speed, TriggArr{1}, Act.Zone, curr_Tool \WObj:=curr_WObj;</pre>
	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)
la	
la	
	Data type: pm_wadescr
	Contains a reference to a work area.
CargetHandle	
	Data type: pm_targethandle
	Contains a reference to a target.
	Continues on next pag

8.3.2 PmGetEvent - Get events for an action *Continued*

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.

For information about	See
The data type pm_wadescr	pm_wadescr - PickMaster work area reference on page 372.
The data type pm_targethandle	pm_targethandle - PickMaster target handle on page 371.
The data type pm_actionhandle	pm_actionhandle - PickMaster action handle on page 342.
The data type pm_eventdata	pm_eventdata - PickMaster event data on page 344.
The instruction TriggEquip	Technical reference manual - RAPID Instructions, Functions and Data types, section Instructions.
The instruction TriggL	Technical reference manual - RAPID Instructions, Functions and Data types, section Instructions.

8.3.3 PmGetPathHeight - Get a safe path height for an intermediate movement

8.3.3 PmGetPathHeight - Get a safe path height for an intermediate movement

Usage	
	PmGetPathHeight returns a safe lowest path height for an intermediate movement between two work areas. None, one or both of the work areas may be defined as the home position of the robot.
	The function returns the maximum height in wobj0 (mm) of the two work areas and their intermediate work areas. The height of a work area is either the <i>active height</i> of the products or the <i>default height</i> , depending on the <i>height state</i> of the work area, see section <i>Height state (GO) on page 265</i> . 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 <i>Feeder order on page 161</i> 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 <i>PmSetDefaultHeight - Set the default height on page 328</i> .
Basic examples	
	Basic examples of the function PmGetPathHeight are illustrated below.
	VAR pm_wadescr waInFeeder;
	VAR pm_wadescr waOutFeeder; VAR num MinZ;
	PmGetFlow waInFeeder,waOutFeeder;
	! Calculate MinZ
	<pre>MinZ:=PmGetPathHeight(waInFeeder,waOutFeeder);</pre>
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 <i>Procedure MoveHomePos on page 214</i> .
ToWa	
20110	Data type: pm_wadescr
	The work area the robot will move to. May also be selected as the home position,
	that is a default installed work area connected to the predefined wobjdata pm_home_Wobj. See section <i>Procedure MoveHomePos on page 214</i> .

325

8.3.3 PmGetPathHeight - Get a safe path height for an intermediate movement *Continued*

[\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 175*.

[\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 176*.

Program execution All errors are considered to be fatal.

More examples

Example 1	
I	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 InterMidl;
	VAR num MinZ;
	PmGetLastWa LastWorkArea;
	! Calculate MinZ
	MinZ:=PmGetPathHeight(LastWorkArea,WorkAreaUseSafePosition)
	+Tgt.TargetTool.tframe.trans.z+SafetyHeight;
	! Use the frame from tool0 and the load from target tool
	TempTool:=Tgt.TargetTool;
	<pre>TempTool.tframe:=tool0.tframe;! Travel distance: 50%</pre>
	InterMidl:=PmCalcIntermid(LastRobTgt,LastTool,LastWobj,
	<pre>Act.RobTgt,Tgt.TargetTool,Tgt.TargetWobj,</pre>
	IntermidPart1\MaxAngle?MaxAngle\MinAngle?MinAngle
	\AngleLimAx6\MinZ:=MinZ);
	<pre>MoveJ\Conc,InterMid1,Act.Speed,z200,TempTool\Wobj:=wobj0;</pre>
E	ENDPROC

Syntax

PmGetPathHeight

```
[FromWa ':=' ] < expression (VAR) of pm_wadescr > ','
[ToWa ':=' ] < expression (VAR) of pm_wadescr > ','
['\' UseSafePosition ] ','
['\' UseDefaultHeight ] ';'
```

Related information

For information about	See
The data type pm_wadescr	pm_wadescr - PickMaster work area refer- ence on page 372
The instruction PmSetLastWa	PmSetLastWa - Set last used work area on page 307

Continues on next page

8.3.3 PmGetPathHeight - Get a safe path height for an intermediate movement *Continued*

For information about	See
The instruction PmGetLastWa	PmGetLastWa - Get last used work area on page 297

8.3.4 PmSetDefaultHeight - Set the default height

8.3.4 PmSetDefaultHeight - Set the default height

Usage	
	PmSetDefaultHeight updates the default height for a work area. It returns the new default height (mm).The height state GO signal of the work area is updated to reflect the change, see <i>Height state (GO) on page 265</i> . 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 <i>Empty</i> 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 <i>Full</i> before a new stack is loaded. This will force the robot to make intermediate movements with enough height when passing over the work area.
	The new default height is active until new targets have been generated (or after new default height is set).
Basic examples	
	Basic examples of the function PmSetDefaultHeight are illustrated below.
Example 1	
•	PERS wobjdata wobjOutfeeder :=
	<pre>[FALSE,TRUE,"",[[2180.65,-1430.22,-220.753],[0.00104,-0.00130,0.00039,1.00000]],[[0,0,0], [1,0,0,0]];</pre>
	VAR pm_wadescr OutWa;
	VAR num NewDefHeight;
	PmGetWaByWobj wobjOutfeeder, OutWa;
	NewDefHeight:=PmSetDefaultHeight OutWa \Empty;
Example 2	
Example 2	PERS wobjdata wobjOutfeeder :=
	<pre>[FALSE,TRUE,"",[[2180.65,-1430.22,-220.753],[0.00104,-0.00130,0.00039,1.00000]],[[0,0,0], [1,0,0,0]];</pre>
	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.
Arguments	
Arguments	PmSetDefaultHeight (Workarea [\Standard] [\Empty] [\Full] [\Latest] [\Value])
Continues on next	page

8.3.4 PmSetDefaultHeight - Set the default height *Continued*

Workarea	
	Data type: pm_wadescr
	The work area.
[\Standard]	
	Data type: switch
	Set the default height as configured in the Robot Path Height see <i>Robot path height on page 176</i> .
[\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.

8.3.5 PmGetTarget - Get target

8.3.5 PmGetTarget - Get target

Usage	
	PmGetTarget is used to get a target for an operation on a work area. If the optional argument OpHandle is left out, the function will return the next target without regard
	to the operation it belongs to.
Basic example	
	PmGetOperation Wa, Op;
	WHILE PmGetTarget(Wa, \OpHandle:=Op.OpHandle, Tgt) DO
	WHILE PmGetTgtAction(Wa, Tgt.TargetHandle, Act) DO
	ENDWHILE
	ENDWHILE
Return value	
	Data type: bool
	The function will return TRUE as long as a new $pm_targetdata$ can be delivered.
Arguments	
	<pre>PmGetTarget (Wa [\OpHandle] Targets [\MaxTime] [\TimeFlag])</pre>
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.
Targat	
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

8.3.5 PmGetTarget - Get target Continued

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

Following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable ERRNO will be set to:

Error code	Description
PM_ERR_TIMEOUT	No ${\tt pm_targetdata}$ could be fetched within the time out time.
PM_ERR_OPERATION_LOST	The pm_operationdata is not valid, probably because of a pulse on the robot execution signal.

Syntax

```
PmGetTarget '('
```

```
[ Wa ':=' ] < expression (IN) of pm_wadescr >
['\' OpHandle ':=' < expression (IN) of pm_ophandle >] ','
[ Target ':=' ] < expression (INOUT) of pm_targetdata >
[ '\' MaxTime ':=' < expression (IN) of num >]
[ '\' TimeFlag ':=' < variable (VAR) of bool >] ')'
```

A function with a return value of the data type bool name.

For information about	See
The data type pm_wadescr	pm_wadescr - PickMaster work area refer- ence on page 372.
The data type pm_ophandle	pm_ophandle - PickMaster operation handle on page 356.
The data type pm_targetdata	pm_targetdata - PickMaster target data on page 368.

8.3.6 PmGetTgtAction - Get target action

8.3.6 PmGetTgtAction - Get target action

Usage	PmGetTgtAction is used t	o get an action for a target on a work area.
Basic examples		
Dasic examples	PmGetOperation Wa, C	; q(
	-	\OpHandle:=Op.OpHandle, Tgt) DO
	WHILE PmGetTgtActi	on(Wa, Tgt.TargetHandle, Act) DO
	curr_WObj := Tgt	.TargetWobj;
	curr_Tool := Tgt	.TargetTool;
	MoveL Tgt.RobTgt	Point, Act.Speed, Act.Zone, curr_Tool
	\WObj:=curr	r_WObj;
	ENDWHILE	
	ENDWHILE	
Return value		
	Data type: bool	
	The function will return TRU for the target handle.	E as long as a new pm_actiondata can be delivered
Arguments		
	PmGetTgtAction (Wa	TargetHandle Action)
Va		
	Data type: pm_wadescr	
	Contains a reference to a w	ork area
TargetHandle		
	Data type: pm_targethand	lle
	Contains a reference to a target.	
Action		
ACTION		
	Data type: pm_actiondata	
	Action data that is fetched f	rom a work area.
Error handling		
	The following recoverable e	rror can be generated. The error can be handled in ar
	-	n variable ERRNO will be set to:
	Error code	Description
	PM_ERR_OPERATION_LOST	The $pm_{operation}$ data is not valid, probably because of a pulse on the robot execution signal.
Syntax		
,	PmGetTgtAction '('	
	[Wa ':='] < expr	ression (IN) of pm_wadescr > ','
		'] < expression (IN) of pm_targethandle > ',
	[Action ':='] <	expression (INOUT) of pm_actiondata > ')'
Continues on next	page	
	r	

3HAC042340-001 Revision: R

8.3.6 PmGetTgtAction - Get target action Continued

A function with a return value of the data type bool name.

For information about	See
The data type pm_wadescr	pm_wadescr - PickMaster work area refer- ence on page 372.
The data type pm_targethandle	pm_targethandle - PickMaster target handle on page 371.
The data type pm_actiondata	pm_actiondata - PickMaster action data on page 339.

8.3.7 PmGetWaHeight - Get the height of a work area

8.3.7 PmGetWaHeight - Get the height of a work area

Usage	PmGetWaHeight gets the current stack height of a specified work area.
	Fildetwallergitt gets the current stack height of a specifica 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.
[\UseSafePositio	n]
	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
	pm_wadescr - PickMaster work area refer- ence on page 372.

8.3.8 PmGetWaName - Get the name of a work area

Usage	
	PmGetWaName gets the name of a specified work area.
Basic examples	
	A basic example of the function PmGetWaName is illustrated below.
	VAR string waname;
	<pre>waname:=PmGetWaName(WorkArea);</pre>
Return value	
	Data type: string
	The function will return the name of the work area.
Arguments	
	PmGetWaName (WorkArea)
WorkArea	
	Work Area
	Data type: pm_wadescr
	Contains a reference to a work area.
Syntax	
-	PmGetWaName '('
	[WorkArea ':='] < expression (VAR) of pm_wadescr >')'
	A function with a return value of the data type string.

8.3.8 PmGetWaName - Get the name of a work area

8.4.1 pm_accdata - PickMaster acceleration/deceleration data

8.4 Data types

8.4.1 pm_accdata - PickMaster acceleration/deceleration data

Usage	
5	${\tt pm_accdata}$ 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	
	maximum acceleration
	Data type: num
	The absolute value of the acceleration limitation in m/s ² .
	Only used when acclim is TRUE.
	Used as argument AccMax in PathAccLim.
decellim	
	deceleration limit
	Data type: bool
	TRUE if there is to be a limitation of the deceleration, FALSE otherwise.

8.4.1 pm_accdata - PickMaster acceleration/deceleration data Continued

Used as argument DecelLim in PathAccLim.

decelmax

maximum deceleration Data type: num The absolute value of the deceleration limitation in m/s². Used only when decellim is TRUE. Used as argument DecelMax in PathAccLim.

Examples

```
VAR pm_actiondata Act;
VAR num my_accmax;
WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) DO
 my_accmax := Act.accel.accmax;
...
ENDWHILE
```

Limitations

The pm_accdata members can only be set by the instruction PmGetTgtAction.

>

Structure

<	dataobject of pm_accdata
	< acc of num >
	< ramp of num >
	< acclim of bool >
	< accmax of num >
	< decellim of bool >
	< decelmax of num >

For information about	See
The data type pm_actiondata	pm_actiondata - PickMaster action data on page 339.
The function PmGetTgtAction	PmGetTgtAction - Get target action on page 332.
The instruction PathAccLim	Technical reference manual - RAPID Instruc- tions, Functions and Data types, section In- structions.
The instruction AccSet	Technical reference manual - RAPID Instruc- tions, Functions and Data types, section In- structions.

8.4.2 pm_acktype - PickMaster target acknowledge type

8.4.2 pm_acktype - PickMaster target acknowledge type

Usage			
	pm_acktype	is used to repre	sent an integer with a symbolic constant for different
	types of acknowledgements.		
Description			
	Apm_acktyp	e is used to de	ide which type of acknowledgement should be used.
Example			
	PmAckTar	get WorkArea,	WorkArea, PM_ACK;
Predefined data			
	Constant	Value	Comment
	PM_ACK	301	The target is acknowledged as used
	PM_NACK	302	The target is acknowledged as not used
	PM_LOST	303	The target is acknowledged as lost
		1	
Characteristics			
Characteristics	pm_acktype characteristics		type for num and consequently inherits its

For information about	See
The instruction PmAckTarget	PmAckTarget - Acknowledge a target on page 288.

8.4.3 pm_actiondata - PickMaster action data

Usage	
	pm_actiondata specifies an action for a target.
Description	
	Properties for one target action.
Components	
RobTgt	
	Data type: robtarget
	Specifies the position of the robot and external axes.
Туре	
	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 <i>Technical reference manual - RAPID Instructions, Functions and Data types</i> .
SingAreaType	
	Data type: pm_singareatype
	Specifies type of interpolation mode.
Accel	
	Acceleration and deceleration data
	Data type: pm_accdata
	Restrict the robot's acceleration and deceleration.
Search	
	Data type: pm_searchdata
	Defines search type, search stop type and search signal to be used with a search movement.
Speed	
	Data type: speeddata

8.4.3 pm_actiondata - PickMaster action data

8.4.3 pm_actiondata - PickMaster action data Continued 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 > < RobTqt 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 >

For information about	See
	pm_movetype - PickMaster movement type on page 351.

8.4.3 pm_actiondata - PickMaster action data *Continued*

For information about	See
The data type pm_accdata	pm_accdata - PickMaster acceleration/deceleration data on page 336.
The data type pm_searchdata	pm_searchdata - PickMaster search data on page 361
The function PmGetTgtAction	PmGetTgtAction - Get target action on page 332.
The data type zonedata	Technical reference manual - RAPID Instructions, Functions and Data types, section Data types.
The data type speeddata	Technical reference manual - RAPID Instructions, Functions and Data types, section Data types
The instruction ConfL	Technical reference manual - RAPID Instructions, Functions and Data types, section Instructions.
Concurrent program execution	Technical reference manual - RAPID overview, section Synchronisation with logical instructions.

8.4.4 pm_actionhandle - PickMaster action handle

8.4.4 pm_actionhandle - PickMaster action handle

Usage		
	pm_actionhandle is used to stor	e data about a target action.
Description		
	Data of the type pm_actionhandl	e contains a reference to an action.
Examples		
•	PmGetOperation WorkArea, C	p;
	WHILE PmGetTarget(WorkArea	\OpHandle:=Op.OpHandle, Tgt) DO
	WHILE PmGetTgtAction(Wor	kArea, Tgt.TargetHandle, Act) DO
	WHILE PmGetEvent(WorkA	rea, Tgt.TargetHandle, Act.ActionHandle,
	Event) DO	
	ENDWHILE	
	ENDWHILE	
	ENDWHILE	
Limitations		
	Describe the limitations for the data script-text.	a type. Names of data types are written in
Characteristics		a type. Names of data types are written in
Characteristics		
Characteristics Related informatic	script-text. pm_actionhandle is a non-value	
	script-text. pm_actionhandle is a non-value	
	script-text. pm_actionhandle is a non-value on	data type.
	script-text. pm_actionhandle is a non-value on For information about	data type. See pm_actiondata - PickMaster action data on
	script-text. pm_actionhandle is a non-value on For information about The data type pm_actiondata	data type. See pm_actiondata - PickMaster action data on page 339.

8.4.5 pm_actiontype - PickMaster action type

8.4.5 pm_actiontype - PickMaster action type

Usage	
	pm_actiontype is used to represent an integer with a symbolic constant for
	different types of actions.
Description	
	A pm_actiontype is used to decide which payload should be used for the next movement.
Example	
	TEST Tgt.Type
	CASE PM_APPROACH_POS:
	<pre>curr_Load := Tgt.AppProdsLoad;</pre>
	CASE PM_TARGET_POS:
	<pre>curr_Load := Tgt.AppProdsLoad;</pre>
	<pre>curr_StopPoint:=Tgt.StopPointData;</pre>
	CASE PM_DEPART_POS:
	<pre>curr_Load := Tgt.DepProdsLoad;</pre>
	ENDTEST
	GripLoad curr_Load;

Constant	Value	Comment
PM_APPROACH_POS	200	The action is a part of the approach movement
PM_TARGET_POS	201	The action moves to the target position
PM_DEPART_POS	202	The action is a part of the depart movement

Characteristics

 ${\tt pm_actiontype}$ is an alias type for ${\tt num}$ and thus inherits its characteristics.

For information about	See
The data type pm_actiondata	pm_actiondata - PickMaster action data on page 339.
The instruction GripLoad	Technical reference manual - RAPID Instruc- tions, Functions and Data types, section In- structions.

8.4.6 pm_eventdata - PickMaster event data

8.4.6 pm_eventdata - PickMaster event data

Usage	
	pm_eventdata is used to specify an event .
Description	
	The components in pm_eventdata are used in the instruction TriggEquip. The different components decide what kind of trigg should be used for a specific action. For example, it describes when a Gripper should be closed or opened.
Components	
type	
	event type
	Data type: pm_eventtype
	Type of event. Used to separate how to set up triggers with instruction TriggEquip. Types that are allowed are PM_EVENT_PROC, PM_EVENT_DO, and PM_EVENT_GO.
time	
	Data type: num
	Input argument EquipLag in instruction TriggEquip.
dist	
	distance
	Data type: num
	Input argument Distance in instruction TriggEquip.
procid	
	process id
	Data type: num
	Input argument ProcID in instruction TriggEquip.
signalname	
	Data type: string
	Input argument DOp or GOp in instruction TriggEquip.
Atime	
	Data type: num
	Reserved for future use.
value	
	Data type: num
	Input argument SetValue in instruction TriggEquip. Should not be used if a PickMaster project uses longer group signals than 23 bits.
Dvalue	
	Data type: dnum
Quartia	
Continues on nex	xī page

8.4.6 pm_eventdata - PickMaster event data Continued

Input argument SetDvalue in instruction TriggEquip. Normally used instead of value since the numerical resolution is higher. Required for PickMaster projects using long group signals, that is longer than 23 bits and up to 32 bits.

Examples

```
VAR pm_eventdata Event;
ArrSize := Dim(TriggArr,1);
WHILE PmGetEvent(WorkArea, Tgt.TargetHandle, Act.ActionHandle,
     Event) AND i <= ArrSize DO
 TEST Event.Type
   CASE PM_EVENT_PROC:
     TriggEquip TriggArr{i}, Event.Dist, Event.Time,
           \ProcID:=Event.ProcId, Event.Value;
   CASE PM_EVENT_DO:
     GetDataVal Event.SignalName,doSignal;
     TriggEquip TriggArr{i}, Event.Dist, Event.Time,
           \DOp:=doSignal, Event.Value;
   CASE PM_EVENT_GO:
     GetDataVal Event.SignalName,goSignal;
     TriggEquip TriggArr{i}, Event.Dist, Event.Time,
           \GOp:=goSignal, Event.Value;
 ENDTEST
 Incr i;
ENDWHILE
```

Limitations

The event data members can only be set by the instruction PmGetEvent.

Structure

< dataobject of pm_eventdata > < type of pm_eventtype > < time of num > < dist of num > < procid of num > < signalname of string > < value of num >

For information about	See
The data type pm_eventtype	pm_eventtype - PickMaster event type on page 346.
The function PmGetEvent	<i>PmGetEvent - Get events for an action on page 323.</i>
The instruction TriggEquip	Technical reference manual - RAPID Instruc- tions, Functions and Data types, section In- structions.
The instruction TriggL	Technical reference manual - RAPID Instruc- tions, Functions and Data types, section In- structions.

8.4.7 pm_eventtype - PickMaster event type

8.4.7 pm_eventtype - PickMaster event type

Usage	
	pm_eventtype is used to represent an integer with a symbolic constant for different
	type of events.
Description	
	A ${\tt pm_eventtype}$ is used to decide which type of trigg event should be used.
Examples	
	TEST Event.Type
	CASE PM_EVENT_PROC:
	<pre>TriggEquip TriggArr{i}, Event.Dist, Event.Time,</pre>
	CASE PM_EVENT_DO:
	GetDataVal Event.SignalName,doSignal;
	<pre>TriggEquip TriggArr{i}, Event.Dist, Event.Time, \DOp:=doSignal, Event.Value;</pre>
	CASE PM_EVENT_GO:
	GetDataVal Event.SignalName,goSignal;
	<pre>TriggEquip TriggArr{i}, Event.Dist, Event.Time, \GOp:=goSignal, Event.Value;</pre>
	ENDTEST

Predefined data

Constant	Value	Comment
PM_EVENT_PROC	220	Process with identity Procld should receive the event. (For internal use)
PM_EVENT_DO	221	Digital output signal is changed.
PM_EVENT_GO	222	Digital group output signal is changed.

Characteristics

pm_eventtype is an alias type for num and thus inherits its characteristics.

For information about	See
The data type pm_eventdata	pm_eventdata - PickMaster event data on page 344.
The instruction TriggEquip.	Technical reference manual - RAPID Instruc- tions, Functions and Data types, section In- structions.

8.4.8 pm_flowinfo - PickMaster flow information

Usage	
-	pm_flowinfo holds information about a flow.
Description	
	The flow information holds data and references to everything that is needed for starting a flow or viewing user information about a specific flow.
Components	
Name	
	Data type: string
	The name of the flow.
SelectionNumber	
	Data type: num
	The number that connects a flow with its I/O signal value. See Flow on page 20
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	
	VAR pm_flowinfo FlowInfo;
	<pre>PmGetFlowInfo FlowSelection,FlowInfo;</pre>
Limitations	
	The flow information members can only be set by the instruction ${\tt PmGetFlowInf}$
Structure	
	< dataobject of pm_flowinfo >
	< Name of string >
	< SelectionNumber of num > < MasterWa of pm_wadescr >
	< Masterwa of pm_wadescr > < SlaveWorkAreas of pm_slavewainfo >
	< NumberOfSlaveWA of num >
	Continues on next pa
Operating manual - Pa	Iletizing PowerPac

8.4.8 pm_flowinfo - PickMaster flow information

8.4.8 pm_flowinfo - PickMaster flow information *Continued*

For information about	See
The instruction PmGetFlowInfo	PmGetFlowInfo - Get information about a specific flow on page 295.
The data type pm_wadescr	pm_wadescr - PickMaster work area refer- ence on page 372.
The data type pm_slavewainfo	pm_slavewainfo - PickMaster slave work area information on page 365.

Usage	
	<pre>pm_moduleinfo holds information about the RAPID modules that can be loaded into a task.</pre>
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 <i>Robot Settings on page 165</i> .
Components	
-	There are 10 components, named in a number series, ModName1 to ModName10.
ModNamel	
	Data type: string
	The name of the file containing a RAPID module.
	5
• • •	
ModName10	
	Data type: string
	The name of the file containing a RAPID module.
Examples	
	VAR pm_projectinfo ProjInfo;
	PmGetProjectInfo ProjectSelection, ProjInfo;
	Load \Dynamic,"HOME:"\File:=ProjInfo.Robot1.ModuleNames.ModName1;
Limitations	
	The operation data members can only be set by the instruction PmGetProjectInfo
	There can be maximum ten modules.
Structure	
ondotaro	< dataobject of pm_moduleinfo >
	< ModName1 of string >
	< ModName2 of string >
	< ModName3 of string >
	< ModName4 of string >
	< ModName5 of string >
	< ModName6 of string >
	< ModName6 of string > < ModName7 of string >
	_
	< ModName7 of string >

8.4.9 pm_moduleinfo - PickMaster module information

8.4.9 pm_moduleinfo - PickMaster module information *Continued*

For information about	See
The instruction PmGetProjectInfo	PmGetProjectInfo - Get information about a specific project on page 300.
The data type pm_robotinfo	pm_robotinfo - PickMaster robot information on page 359.

8.4.10 pm_movetype - PickMaster movement type

8.4.10 pm_movetype - PickMaster movement type

Jsage			
•	pm_movetype is us	ed to represent	an integer with a symbolic constant for differ
	type of movements	-	
Description			
	The pm_movetype be used.	is used to decid	le which type of movement instruction shou
xample			
	TEST Move		
	CASE PM_MOV	/E_LIN:	
			Zone, Tool\WObj:=WObj;
	CASE PM_MOV		
		Point, Speed,	Zone, Tool\WObj:=WObj;
	ENDTEST		
Predefined data			
	Constant	Value	Comment
	Constant PM_MOVE_JOINT	Value 240	Comment The action is a joint movement
haractoristics	PM_MOVE_JOINT	240	The action is a joint movement
Characteristics	PM_MOVE_JOINT PM_MOVE_LIN	240 241	The action is a joint movement The action is a linear movement
Characteristics	PM_MOVE_JOINT PM_MOVE_LIN pm_movetype is ar	240 241	The action is a joint movement
Characteristics	PM_MOVE_JOINT PM_MOVE_LIN	240 241	The action is a joint movement The action is a linear movement
	PM_MOVE_JOINT PM_MOVE_LIN pm_movetype is ar characteristics.	240 241	The action is a joint movement The action is a linear movement
Characteristics Related informat	PM_MOVE_JOINT PM_MOVE_LIN pm_movetype is ar characteristics.	240 241	The action is a joint movement The action is a linear movement
	PM_MOVE_JOINT PM_MOVE_LIN pm_movetype is ar characteristics.	240 241 n alias data type	The action is a joint movement The action is a linear movement for num and consequently inherits its
	PM_MOVE_JOINT PM_MOVE_LIN pm_movetype is ar characteristics. tion For information abo	240 241 n alias data type ut	The action is a joint movement The action is a linear movement for num and consequently inherits its See pm_actiondata - PickMaster action data of the section

8.4.11 pm_offsetdata - Offset data

8.4.11 pm_offsetdata - Offset data

Jsage	
	pm_offsetdata describes an offset position.
escription	
	The $pm_offsetdata$ is used to displace a robot position in x, y, and z direction
	and rotation around the axis.
Components	
- 2	
	Data type: num
	The displacement in the x-direction, in the object coordinate system.
7	
	Data type: num
	The displacement in the y-direction, in the object coordinate system.
7	
Ζ	Data type: num
	The displacement in the z-direction, in the object coordinate system.
îx.	Data type: num
	The rotation in degrees around the x-axis of the tool coordinate system.
сy	
	Data type: num
	The rotation in degrees around the y-axis of the tool coordinate system.
CZ	
	Data type: num
	The rotation in degrees around the z-axis of the tool coordinate system.
Examples	
-	VAR pm_actiondata Act;
	VAR num x_offset;
	WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) DO x offset := Act.offset.x
	_
	ENDWHILE
imitations	The offect data members can only be get by the instruction product with the instruction product with the set of the set o
	The offset data members can only be set by the instruction PmGetTgtAction.
Structure	
	< dataobject of pm_offsetdata >
	< x of num >
Continues on nex	
352	Operating manual - Palletizing PowerPa 3HAC042340-001 Revision:

8.4.11 pm_offsetdata - Offset data Continued

< y of num > < z of num > < rx of num > < ry of num > < rz of num >

For information about	See
The data type pm_actiondata	pm_actiondata - PickMaster action data on page 339.
The function PmGetTgtAction	PmGetTgtAction - Get target action on page 332.
The function Offs	<i>Technical reference manual - RAPID Instruc- tions, Functions and Data types, section Functions.</i>

8.4.12 pm_operationdata - PickMaster operation data

8.4.12 pm_operationdata - PickMaster operation data

Usage	
	pm_operationdata is used to specify an operation on a work area.
Description	
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	
SCEITEITUIIDET	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.
Structure	
	< dataobject of pm_operationdata >
	< operationnumber of num >
	< layernumber of num >
Continues on next p	age
354	Operating manual - Palletizing PowerPac
	© Copyright 2012-2020 ABB. All rights reserved

8.4.12 pm_operationdata - PickMaster operation data Continued

- < scenenumber of num >
- < formatid of num >
- < numoftargets of num >
- < ophandle of pm_ophandle >

For information about	See
The instruction PmGetOperation	PmGetOperation - Get operation from a work area on page 298.
The format id value	The Pick setting on page 79 Configuration.

8.4.13 pm_ophandle - PickMaster operation handle

8.4.13 pm_ophandle - PickMaster operation handle

Usage		
	pm_ophandle is used to store data about an opera	tion.
Description		
	Data of the type $pm_ophandle$ contains a reference	e to an operation.
Examples		
	PmGetOperation WorkArea, Op;	
	WHILE PmGetTarget(WorkArea \OpHandle:=Op	.OpHandle, Tgt) DO
	ENDWHILE	
Characteristics		
	pm_ophandle is a non-value data type .	
Related informatio	n	
	For information about See	

For information about	See
The data type pm_operationdata	pm_operationdata - PickMaster operation data on page 354.
The instruction PmGetOperation	<i>PmGetOperation - Get operation from a work area on page 298.</i>

8.4.14 pm_projectinfo - PickMaster project information

Usage	
Usage	pm_projectinfo holds information about a project.
Description	
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	
ProjectDescript	ion
	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 <i>Project Manager</i> on page 108.
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.

8.4.14 pm_projectinfo - PickMaster project information *Continued*

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 >	

For information about	See
The instruction PmGetProjectInfo	PmGetProjectInfo - Get information about a specific project on page 300.
The data type pm_robotinfo	pm_robotinfo - PickMaster robot information on page 359.

8.4.15 pm_robotinfo - PickMaster robot information

Usage	
-	pm_robotinfo holds information about a RAPID task connected to a robot.
Description	
	The robot information holds data and references to everything that is needed fo 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	
	VAR pm_projectinfo ProjInfo;
	<pre>PmGetProjectInfo ProjectSelection,ProjInfo;</pre>
	IF projInfo.Robot1.LoadRapid THEN
	LoadAllModules projInfo.Robot1.ModuleNames, projInfo.Name, projInfo.Robot1.TaskName;
	ENDIF
Limitations	
	The robot information members can only be set by the instruction
	PmGetProjectInfo.
Structure	
	< dataobject of pm_robotinfo >
	< Active of bool >
	< Name of string >
	< TaskName of string >
	< ModuleNames of pm_moduleinfo >
	Continues on next pag
Operating manual -	Palletizing PowerPac 34

8.4.15 pm_robotinfo - PickMaster robot information

8.4.15 pm_robotinfo - PickMaster robot information *Continued*

For information about	See
The instruction PmGetProjectInfo	PmGetProjectInfo - Get information about a specific project on page 300.
The data type pm_projectinfo	pm_projectinfo - PickMaster project informa- tion on page 357.
The data type pm_moduleinfo	pm_moduleinfo - PickMaster module inform- ation on page 349.

8.4.16 pm_searchdata - PickMaster search data

Jsage	
	pm_searchdata is used to store data used for search movements.
escription	
	pm_searchdata is a part of pm_actiondata and is used as input argument to procedure DoSearch. It defines search specific data.
components	
earchtype	
	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	
iotriggtype	
lotriggtype	movement.
	movement. Data type: pm_iotriggtype I/O trigger type. Defines how the search signal is triggered, e.g trigger on both
	movement. Data type: pm_iotriggtype I/O trigger type. Defines how the search signal is triggered, e.g trigger on both
	movement. Data type: pm_iotriggtype I/O trigger type. Defines how the search signal is triggered, e.g trigger on both flanks.
	<pre>movement. Data type: pm_iotriggtype I/O trigger type. Defines how the search signal is triggered, e.g trigger on both flanks. VAR pm_actiondata Act; VAR pm_stoptype my_stoptype; WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) DO</pre>
	movement. Data type: pm_iotriggtype I/O trigger type. Defines how the search signal is triggered, e.g trigger on both flanks. VAR pm_actiondata Act; VAR pm_stoptype my_stoptype;
	<pre>movement. Data type: pm_iotriggtype I/O trigger type. Defines how the search signal is triggered, e.g trigger on both flanks. VAR pm_actiondata Act; VAR pm_stoptype my_stoptype; WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) DO</pre>
Example	<pre>movement. Data type: pm_iotriggtype I/O trigger type. Defines how the search signal is triggered, e.g trigger on both flanks. VAR pm_actiondata Act; VAR pm_stoptype my_stoptype; WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) DO my_stoptype := Act.search.stoptype;</pre>
Example	<pre>movement. Data type: pm_iotriggtype I/O trigger type. Defines how the search signal is triggered, e.g trigger on both flanks. VAR pm_actiondata Act; VAR pm_stoptype my_stoptype; WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) D0 my_stoptype := Act.search.stoptype; ENDWHILE</pre>
Example Limitations	<pre>movement. Data type: pm_iotriggtype I/O trigger type. Defines how the search signal is triggered, e.g trigger on both flanks. VAR pm_actiondata Act; VAR pm_stoptype my_stoptype; WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) D0 my_stoptype := Act.search.stoptype; ENDWHILE</pre>
Example Limitations	<pre>movement. Data type: pm_iotriggtype I/O trigger type. Defines how the search signal is triggered, e.g trigger on both flanks. VAR pm_actiondata Act; VAR pm_stoptype my_stoptype; WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) D0 my_stoptype := Act.search.stoptype; ENDWHILE</pre>
Example Limitations	movement. Data type: pm_iotriggtype I/O trigger type. Defines how the search signal is triggered, e.g trigger on both flanks. VAR pm_actiondata Act; VAR pm_stoptype my_stoptype; WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) D0 my_stoptype := Act.search.stoptype; ENDWHILE The pm_searchdata members can only be set by the instruction PmGetTgtAction
Example Limitations	<pre>movement. Data type: pm_iotriggtype I/O trigger type. Defines how the search signal is triggered, e.g trigger on both flanks. VAR pm_actiondata Act; VAR pm_stoptype my_stoptype; WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) D0 my_stoptype := Act.search.stoptype; ENDWHILE The pm_searchdata members can only be set by the instruction PmGetTgtAction <dataobject of="" pm_searchdata=""></dataobject></pre>
iotriggtype Example Limitations	<pre>movement. Data type: pm_iotriggtype I/O trigger type. Defines how the search signal is triggered, e.g trigger on both flanks. VAR pm_actiondata Act; VAR pm_stoptype my_stoptype; WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) DO my_stoptype := Act.search.stoptype; ENDWHILE The pm_searchdata members can only be set by the instruction PmGetTgtAction <dataobject of="" pm_searchdata=""> < searchtype of pm_searchtype ></dataobject></pre>

8.4.16 pm_searchdata - PickMaster search data

8 **RAPID** reference information

8.4.16 pm_searchdata - PickMaster search data *Continued*

Related information

For information about	See
	pm_actiondata - PickMaster action data on page 339.
The function PmGetTgtAction	PmGetTgtAction - Get target action on page 332.

8.4.17 pm_searchtype - PickMaster stack search type

Usage			
	pm_searchtype is	s used to represe	nt an integer with a symbolic constant for
	different types of s	tack search types	5.
Description			
	Apm_searchtype	e is used to speci	iy which type of stack search type is used.
Example			
	PmSearchAdju	st PickWA, PM_S	SEARCH_Z, 570;
Predefined data			
	Constant	Value	Comment
	PM_SEARCH_X	0	Stack search in the x direction
	PM_SEARCH_Y	1	Stack search in the y direction
	PM_SEARCH_Z	2	Stack search in the z direction
Characteristics			
onaracteristics	pm searchtype is	s an alias data tvr	be for num and consequently inherits its
	characteristics.		
Related information	on		
	For information abo	out	See
	The instruction PmSearchAdjust		PmSearchAdjust - Adjust number of remain

8.4.18 pm_singareatype - PickMaster interpolation type around singular points

8.4.18 pm_singareatype - PickMaster interpolation type around singular points

Usage	
	pm_singareatype is used to represent an integer with a symbolic constant for different types of interpolation around singular points.
Description	
	The pm_singareatype is used to decide how the robot is to move in the proximity of singular points.
Examples	
	IF Act.SingAreaType = PM_SING_AREA_OFF THEN
	SingArea\Off;
	ELSEIF Act.SingAreaType = PM_SING_AREA_WRI THEN
	SingArea\Wrist;
	ELSE
	SingArea\LockAx4;

Constant	Value	Description
PM_SING_AREA_OFF	260	The tool orientation is not allowed to differ.
PM_SING_AREA_WRI	261	The tool orientation is allowed to differ somewhat to avoid wrist singularity.
PM_SING_AREA_LOCKAX4	262	The orientation is such that the 4th axis is locked, that is, 6 axis robot behaves like a 4 axis robot.

Characteristics

pm_singareatype is an alias type for num and thus inherits its characteristics.

Related information

For information about	See
The data type pm_actiondata	pm_actiondata - PickMaster action data on page 339.
The instruction SingArea	Technical reference manual - RAPID Instruc- tions, Functions and Data types, section In- structions.

8.4.19 pm_slavewainfo - PickMaster slave work area information

8.4.19 pm_slavewainfo - PickMaster slave work area information

Usage	
	${\tt pm_slavewainfo}$ 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 ${\tt pm_wadescr}$. 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, SlaveWal to SlaveWal9.
SlaveWal	
	Data type: pm_wadescr
	The descriptor to a work area.
SlaveWa19	
	Data type: pm_wadescr
	The descriptor to a work area.
Examples	
	VAR pm_flowinfo FlowInfo;
	VAR num height;
	VAR num FlowSelection;
	<pre>PmGetFlowInfo FlowSelection,FlowInfo;</pre>
	height:=PmGetWaHeight(FlowInfo.SlaveWorkAreas.SlaveWal);
Limitations	
Limitations	The slave work area information members can only be set by the instruction
Limitations	The slave work area information members can only be set by the instruction PmGetFlowInfo.
	PmGetFlowInfo.
	<pre>PmGetFlowInfo.</pre>
	<pre>PmGetFlowInfo.</pre>
	<pre>PmGetFlowInfo. < dataobject of pm_slavewainfo ></pre>
	<pre>PmGetFlowInfo. < dataobject of pm_slavewainfo > < SlaveWa1 of pm_wadescr > < SlaveWa2 of pm_wadescr > < SlaveWa3 of pm_wadescr > < SlaveWa4 of pm_wadescr > < SlaveWa5 of pm_wadescr > </pre>
	<pre>PmGetFlowInfo. </pre> <pre>< dataobject of pm_slavewainfo > <pre>< SlaveWa1 of pm_wadescr > <pre>< SlaveWa2 of pm_wadescr > <pre>< SlaveWa3 of pm_wadescr > <pre>< SlaveWa4 of pm_wadescr > <pre>< SlaveWa5 of pm_wadescr > <pre>< SlaveWa6 of pm_wadescr ></pre></pre></pre></pre></pre></pre></pre>
	<pre>PmGetFlowInfo. < dataobject of pm_slavewainfo ></pre>
	<pre>PmGetFlowInfo. < dataobject of pm_slavewainfo > <</pre>
	<pre>PmGetFlowInfo. < dataobject of pm_slavewainfo > < SlaveWal of pm_wadescr > SlaveWa2 of pm_wadescr > SlaveWa3 of pm_wadescr > SlaveWa4 of pm_wadescr > SlaveWa5 of pm_wadescr > SlaveWa6 of pm_wadescr > SlaveWa6 of pm_wadescr > SlaveWa7 of pm_wadescr > SlaveWa8 of pm_wadescr > SlaveWa9 of pm_wadescr > SlaveWa</pre>
	<pre>PmGetFlowInfo. < dataobject of pm_slavewainfo > <</pre>
	<pre>PmGetFlowInfo. </pre> <pre>< dataobject of pm_slavewainfo > <pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre></pre>
Limitations	<pre>PmGetFlowInfo. < dataobject of pm_slavewainfo > <</pre>
	<pre>PmGetFlowInfo. </pre> <pre>< dataobject of pm_slavewainfo > <pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre></pre>

8 **RAPID** reference information

8.4.19 pm_slavewainfo - PickMaster slave work area information *Continued*

< SlaveWal5 of pm_wadescr >
< SlaveWal6 of pm_wadescr >
< SlaveWal7 of pm_wadescr >
< SlaveWal8 of pm_wadescr >
< SlaveWal9 of pm_wadescr >

Related information

For information about	See
The instruction PmGetProjectInfo	PmGetProjectInfo - Get information about a specific project on page 300.
The data type pm_flowinfo	pm_flowinfo - PickMaster flow information on page 347.

8.4.20 pm_stoptype - PickMaster stop type

8.4.20 pm_stoptype - PickMaster stop type

Usage	
-	$pm_stoptype$ is used to represent an integer with a symbolic constant for different
	types of stop methods.
Description	
	A ${\tt pm_stoptype}$ 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;
	<pre>StopType := SearchData.StopType;</pre>
	TEST StopType
	CASE PM_STOP_NOT_USED:
	CASE PM_STOP:
	CASE PM_SSTOP:
	CASE PM_PSTOP:

Predefined data

Constant	Value	Comment
PM_STOP_NOT_USED	0	No stop is being used
PM_STOP	1	Robot stiff stop
PM_PSTOP	2	Robot program stop
PM_SSTOP	3	Robot soft stop

Characteristics

 $\tt pm_stoptype$ is an alias data type for num and consequently inherits its characteristics.

Related information

For information about	See
The data type pm_searchdata	pm_searchdata - PickMaster search data on page 361.

8.4.21 pm_targetdata - PickMaster target data

8.4.21 pm_targetdata - PickMaster target data

Usage	
obuge	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	
Seeling and a set	Data type: stoppointdata
	Defines the stoppointdata to be used when moving to the position of the target.
NumOfAppProds	
	number of approach products
	Data type: num

8.4.21 pm_targetdata - PickMaster target data Continued

AppProdsLoad	
	ask product load
	ach product load
-	/pe:loaddata
Define	s the load attached to the robot tool when approaching the target position.
NumOfDepProds	
numbe	er of depart products
Data ty	/pe: num
The nu	Imber of products held by the robot tool when departing from the target
positio	n.
DepProdsLoad	
	product load
•	/pe:loaddata
-	s the load attached to the robot when departing from the target position.
Denne	s the load attached to the lobot when departing norm the target position.
NumOfActions	
numbe	er of actions
Data ty	/pe: num
Specif	ies the number of target actions.
TargetHandle	
	/pe:pm_targethandle
A refe	rence to the target from where this $pm_targetdata$ was retrieved.
	5
ProductHeight	
-	/pe: num
	eight of the product to place or pick. It is not necessary a product in the tool
for this	s target.
Examples	
•	GetOperation WorkArea, Op;
WH	IILE $PmGetTarget(WorkArea \\OpHandle:=Op.OpHandle, Tgt) DO$
HIN BIN	IDWHILE
Limitations	
Limitations	peration data members can only be set by the instruction PmGetTarget.
Limitations The op	peration data members can only be set by the instruction PmGetTarget.
Limitations The op Structure	
Limitations The op Structure	dataobject of pm_targetdata>
Limitations The op Structure	
Limitations The op Structure	dataobject of pm_targetdata> < TargetNumber of num >
Limitations The op Structure	<pre>dataobject of pm_targetdata> < TargetNumber of num > < OperationNumber of num > < LayerNumber of num > < SceneNumber of num ></pre>
Limitations The op Structure	<pre>dataobject of pm_targetdata> < TargetNumber of num > < OperationNumber of num > < LayerNumber of num > < SceneNumber of num > < RobTgtPoint of robtarget ></pre>
Limitations The op Structure	<pre>dataobject of pm_targetdata> < TargetNumber of num > < OperationNumber of num > < LayerNumber of num > < SceneNumber of num ></pre>

8 **RAPID** reference information

8.4.21 pm_targetdata - PickMaster target data *Continued*

- < StopPointData of stoppointdata >
- < NumOfAppProds of num >
- < AppProdsLoad of loaddata >
- < NumOfDepProds of num >
- < DepProdsLoad of loaddata >
- < NumOfActions of num >
- < TargetHandle of pm_targethandle >
- < ProductHeight of num >

Related information

For information about	See
The instruction PmGetTarget	PmGetTarget - Get target on page 330.
The data type robtarget	Technical reference manual - RAPID Instruc- tions, Functions and Data types, section Data types.
The data type tooldata	Technical reference manual - RAPID Instruc- tions, Functions and Data types, section Data types.
The data type wobjdata	Technical reference manual - RAPID Instruc- tions, Functions and Data types, section Data types.
The data type stoppointdata	Technical reference manual - RAPID Instruc- tions, Functions and Data types, section Data types.
The data type loaddata	Technical reference manual - RAPID Instruc- tions, Functions and Data types, section Data types.

8.4.22 pm_targethandle - PickMaster target handle

8.4.22 pm_targethandle - PickMaster target handle

Jsage	pm_targethandle is used to stor	e data about a target.			
Description					
	Data of the type pm_targethandl	e contains a reference to a target.			
Example					
	PmGetOperation WorkArea, Op;				
	WHILE PmGetTarget(WorkArea \OpHandle:=Op.OpHandle, Tgt) DO				
	WHILE PmGetTgtAction(WorkArea, Tgt.TargetHandle, Act) DO				
	ENDWHILE				
	ENDWHILE				
Characteristics					
	pm_targethandle is a non-value	data type.			
Related informatio	n				
	For information about	See			
	The data type pm_targetdata	pm_targetdata - PickMaster target data on page 368.			
	The function PmGetTarget	PmGetTarget - Get target on page 330.			

For information about	See
The data type pm_targetdata	pm_targetdata - PickMaster target data on page 368.
The function PmGetTarget	PmGetTarget - Get target on page 330.
The function PmGetTgtAction	PmGetTgtAction - Get target action on page 332.

8.4.23 pm_wadescr - PickMaster work area reference

8.4.23 pm_wadescr - PickMaster work area reference

Usage		
-	pm_wadescr is used to store data	about a work area.
Description		
	Data of the type pm_wadescr cont	ains a reference to a work area.
Examples		
	PERS wobjdata wInfeeder1	:=
	[FALSE, TRUE, "", [[218	0.65,-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 t y	ype.
Related information	on	
	For information about	See
	The function PmGetWaByWobj	PmGetWaByWobj - Get a reference to a work area using a work object data on page 302.

Usage	
	pm_wainfo holds information about a work area.
Description	
	The work area information holds data and references that can be used for viewing user information about the specific work area.
Components	
Name	
	Data type: string
	The name of the work area.
SelectionNumber	
	Data type: num
	The number that connects a work area with its I/O signal value. See I/O values on page 107.
Workarea	
	Data type: pm_wadescr
	The work area descriptor.
TaskNo	
	Data type: num
	The RAPID task number this work area is connected to. Only valid after the first call to PmGetTarget for this work area.
Order	
	Data type: num
	The configured order for this work area. See <i>Feeder order on page 161</i> .
DefaultHeight	
	Data type: num
	The configured default height. See <i>Robot path height on page 176</i> .
DiTgtGenTrig	
	Data type: string
	The name of the configured target generation trigger signal.
DoPosRegTrig	
	Data type: string
	The name of the configured position request trigger signal.
GoProdSel	
	Data type: string
	The name of the configured target generation product selection signal.

8.4.24 pm_wainfo - PickMaster Work Area information

8 **RAPID** reference information

8.4.24 pm_wainfo - PickMaster Work Area information *Continued*

374	Operating manual - Palletizing PowerPac
Continues on next pa	
Examples	VAR pm_wainfo WaInfo; VAR num WaSelection:=1;
	The number of accumulated picked or placed products since project start for this work area.
AccumulatedNumOf	Data type: num
	The number of accumulated acknowledged targets since project start for this work area.
AccumulatedAckTa	nrget Data type: num
	The name of the configured execution state signal.
DoExecState	Data type: string
DoEvocetoto	The name of the configured operation set complete signal.
DoOpSetCompl	Data type: string
	The name of the configured position available signal.
DoPosAvail	Data type: string
	The name of the configured queue empty signal.
DoQueueEmpty	Data type: string
	The name of the configured redo search signal.
DiRedoSearch	Data type: string
	The name of the configured robot execution signal.
DiRobotExec	Data type: string
	The name of the configured position request format selection signal.
GiFormSel	Data type: string
	The name of the configured position request product selection signal.
GiProdSel	Data type: string
	The name of the configured target generation format selection signal.
GoFormSel	Data type: string
CoFormCol	

3HAC042340-001 Revision: R

8 **RAPID** reference information

8.4.24 pm_wainfo - PickMaster Work Area information Continued

PmGetWaInfo WaSelection,WaInfo;

Limitations	
	The work area information members can only be set by the instructior
	PmGetWaInfo.
Structure	
	< dataobject of pm_wainfo >
	< Name of string >
	< SelectionNumber of num >
	< WorkArea of pm_wadescr >
	< TaskNo of num >
	< Order of num >
	< DefaultHeight of num >
	< DiTgtGenTrig of string >
	< DoPosReqTrig of string >
	< GoProdSel of string >
	< GoFormSel of string >
	< GiProdSel of string >
	< GiFormSel of string >
	< DiRobotExec of string >
	< DiRedoSearch of string >
	< DoQueueEmpty of string >
	< DoPosAvail of string >
	< DoOpSetCompl of string >
	< DoExecState of string >
	< AccumulatedAckTarget of num >
	< AccumulatedNumOfProd of num >

Related information

For information about	See
The instruction PmGetWaInfo	PmGetWaInfo - Get information about a spe- cific work area on page 303
The data type pm_wadescr	pm_wadescr - PickMaster work area refer- ence on page 372

This page is intentionally left blank

9.1 Structure of this chapter

9 Relationships between PickMaster frames

9.1 Structure of this chapter

This chapter describes the different frames used by PickMaster and how they relate to each other. Each section describes the relationship between two (or several) frames.

9 Relationships between PickMaster frames

9.2 Shape frame

9.2 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.

9.3 Format frame versus shape frame

9.3 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 79* 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.

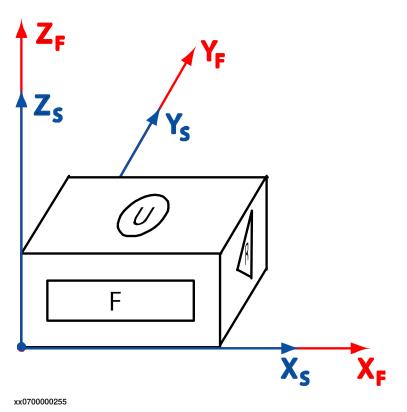
The relationship between the format frame and the shape frames will also depend on the X size, Y size, and Z size of the shape defined in the **Product/Pallet/Sheet Configuration** and the tuning of item size which is displayed in the same dialog.

Orientation explanation

Orientation defines which side of the item to place in the negative y-direction.

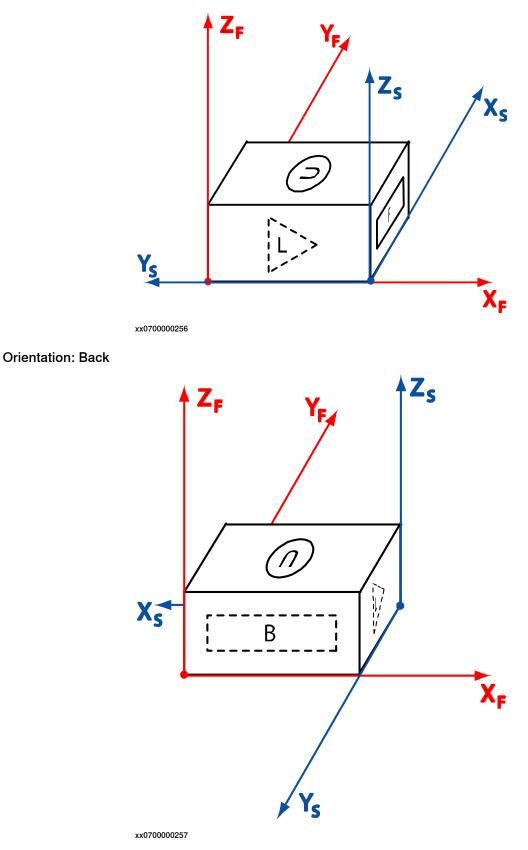
The following examples show some different orientations.

Orientation: Front



$\mathbf{X}_{F}, \mathbf{Y}_{F}, \mathbf{Z}_{F}$	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
В	Back side of the shape
U	Upper side of the shape

Orientation: Left

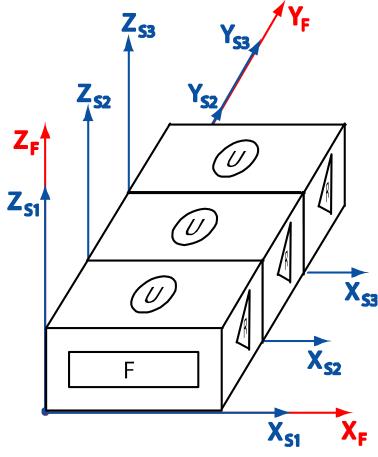


Row explanation

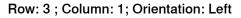
Row defines number of shapes in the y-direction of the format frame.

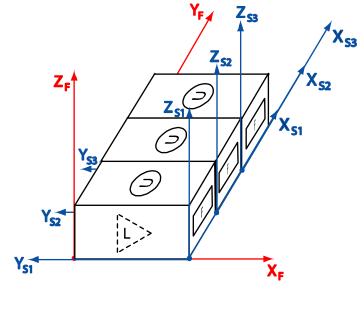
The following examples show the combination of row, column, and orientation.

Row: 3 ; Column: 1; Orientation: Front



xx0700000258



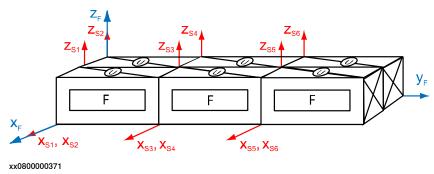


xx0700000259

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 79* configuration.

9 Relationships between PickMaster frames

9.4 Pallet pattern frame versus shape frame

9.4 Pallet pattern frame versus shape frame

Use a palletizing area shape

The pallet pattern frame coincides with the shape frame of the palletizing area shape, which is selected in the **Layout Configuration** (only shapes with **Form** set to **Pallet** can be selected).

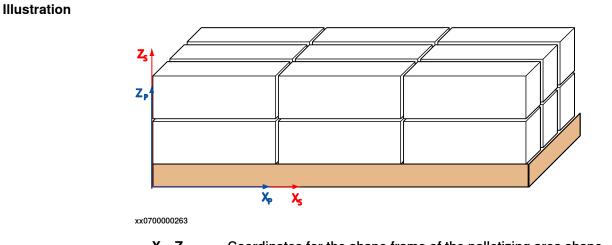
The palletizing area shape can be used as pallet in the pallet pattern but this is not mandatory.

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

Where to see the shape frame and pallet pattern

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



X _S , Z _S	Coordinates for the shape frame of the palletizing area shape
X_P, Z_P	Coordinates for the pallet pattern frame

9.5 Format frame versus tool frame

Illustration Too 7 8 9 10 2 3 4 5 6 11 12 Item frame Zone frame Tool frame Z z Format ν frame ltem Item 2 Item 3 X, z 🕇 Offset frame xx070000264 Format frame The format frame is defined in the Item Group Configuration window, under Product/Pallet/Sheet window, shown in the 3D view. Item frame The offset frame is selected by setting Aligned item in the Pick Setting Configuration window, under Tool location. The offset frame is positioned on the top of the selected item and has the same orientation as the format frame. Offset frame The offset frame has an offset relative to the item frame. This offset is selected by setting Offset in the Pick Setting Configuration window, under Tool location. The offset frame z-direction is always opposite to the item frame. The orientation of the offset frame's xy-plane relative to the item frame can be selected in steps of 90 degrees in the Pick Setting Configuration window, under Tool location. Tool frame The zone frame cannot be directly viewed in the **Pick Setting Configuration** window. Instead, the location of the tool frame is displayed. The tool frame position is different from the zone frame but the orientation is the same. Zone frame The zone frame is selected by setting aligned tool configuration and zone in the Pick Setting Configuration window, under Tool location. The zone frame has an offset relative to the tool frame. The location of the zone frames and the corresponding offsets are displayed in the Tool Function(Vacuum) window, under the Zones tab. The offset can be changed by selecting Activator in Zones, or by setting the activator x, y, z in the Tool Function(Vacuum) window, under Activators tab.

9.5 Format frame versus tool frame

9 Relationships between PickMaster frames

9.5 Format frame versus tool frame *Continued*

Zone frame and offset frame

When a format is being picked by a robot, a zone frame coincides with an offset frame.

Where to see the tool frame

The location of the tool frame relative to a format can be viewed in the **Pick Setting Configuration** window. See the *Pick setting on page 79* configuration.

9.6 Pallet pattern item versus tool frame

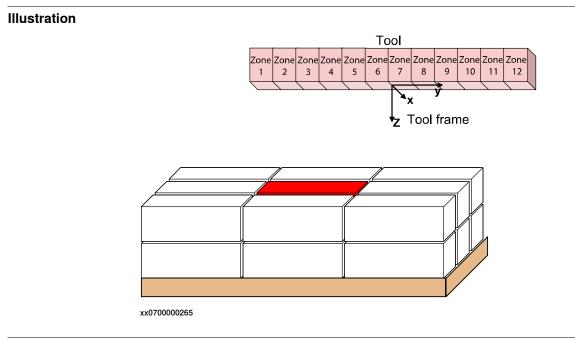
9.6 Pallet pattern item versus tool frame

Tool location

The location of the tool relative to a specific item when picking/placing it, is automatically solved by PickMaster. The location of the tool is highly dependent on the selected **Formats to use** in the **Pallet Pattern Operation Set Configuration**. Every used format creates a number of possible positions of the tool relative to the item. The solution must also consider that every layer must be completely finished before starting with the next layer in the pallet pattern.

The autogenerated solution can be modified in detail in the **Operation Editor**, which is accessible from the **Pallet Pattern Operation Set Configuration**. For each item in every layer is it possible to modify:

- The pick/place order within the layer.
- The format to be used.
- The item in the selected format to be used.
- The orientation of the item may be flipped 180 degrees.
- · Single access or (if possible) grouped access with adjacent items.



Where to see the tool frame

The location of the tool frame relative to an item in a pallet pattern can be viewed in the **Pallet Pattern Operation Set Configuration** window. See *Pattern/Stack operation set on page 183*.

Related information

The Operation editor on page 196.

9 Relationships between PickMaster frames

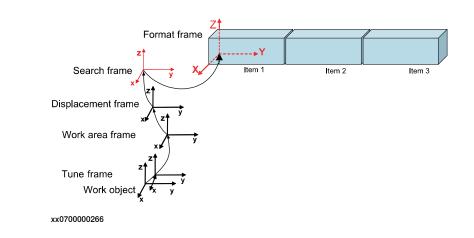
9.7 Format frame versus work object in format operation set

9.7 Format frame versus work object in format operation set

Overview

The expected location of a format relative to the work object while being picked or placed depends on a chain of frames, starting with the work object and ending with the format frame.

Illustration



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 Palletizing PowerPac, the position of the work object is aligned with the selected hotspot of the feeder model (Feeder hotpot configuration window).

Tune frame

The location of the tune frame relative the work object is defined by the current online tuning of the work area. By default the tune frame coincides with the work object. The tune frame location may be changed from the PickMaster RC online tuning by updating the following work area properties: **Disp offs x**, **Disp offs y**, **Disp offs z**, and **Disp angle z**. The tune frame is first displaced from the work object with the Disp offs vector. Then the tune frame is rotated around the displaced origo in the z-direction with the Disp angle z.

Work area frame

The location of the work area frame relative to the tune frame is defined by the Alignment and Rotation settings of the selected hotspot used by the feeder.

Displacement frame

The location of the displacement frame relative to the work area frame is defined by the settings of **Displacement frame**: x, y, z and z angle in the **Group Operation Set Configuration** window.

Continues	on	next	page
-----------	----	------	------

9.7 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.

9 Relationships between PickMaster frames

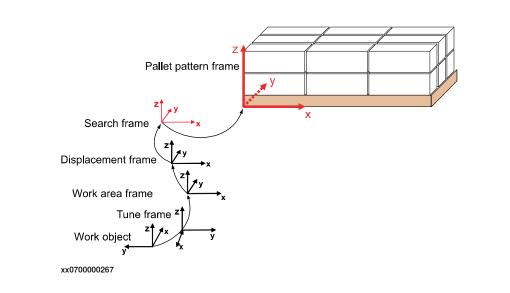
9.8 Pallet pattern frame versus work object in pallet pattern operation set

9.8 Pallet pattern frame versus work object in pallet pattern operation set

Overview

The expected location of a pallet pattern frame relative to the work object while being picked or placed depends on a chain of frames, starting with the work object and ending with the pallet pattern frame.

Illustration



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 Palletizing PowerPac, 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.

9.8 Pallet pattern frame versus work object in pallet pattern operation set *Continued*

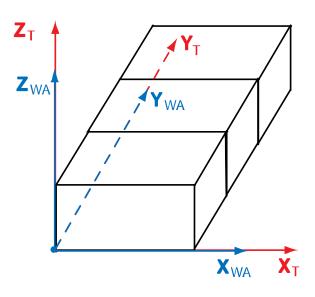
Displacement fram	ne
	The location of the displacement frame relative to the work area frame is defined by the settings of Displacement frame : x, y, z and z angle in the Pallet Pattern Operation Set Configuration window.
Search frame	
	Search frame has an offset relative to the displacement frame. The 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 v	vork 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 locatior is only valid if the tune frame location is zero (=default) and if the search frame offset is zero.

9.9 Tune frame versus work area frame

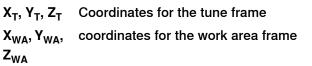
9.9 Tune frame versus work area frame

Formats and pallet patterns		
	The relationship between tune frame and work area frame is valid for both formats and pallet patterns.	
Alignment		
	Alignment defines which side of the work area frame the tune frame is found on, left or right.	
	The offset of the work area frame is zero with left alignment. With right alignment, the offset becomes equal to the x-size of the format expressed in the format frame.	
Orientation		
	Orientation defines the orientation of the tune frame in the z-direction relative to	
	the work area frame: 0, 90, 180 or 270 degrees.	
Examples		
	The displayed positions of the format in these examples assumes no displacement frame offset/reorientation and zero search offset.	

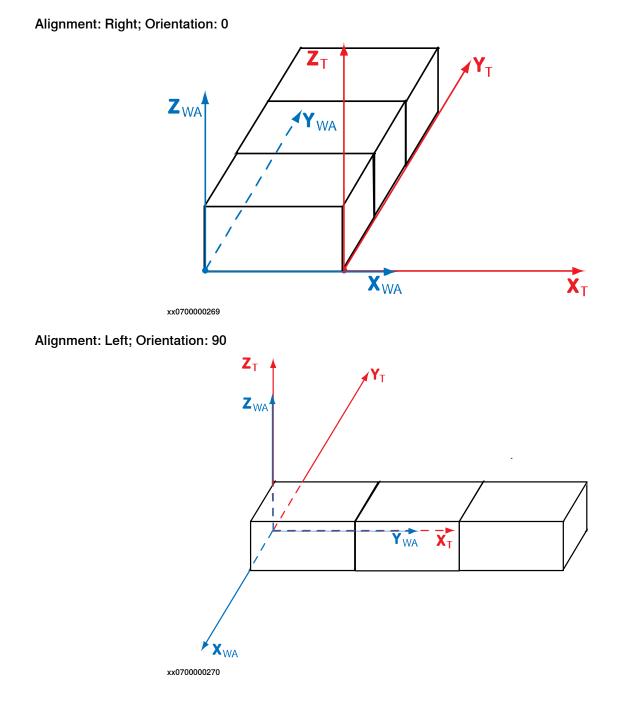
Alignment: Left; Orientation: 0



xx0700000268

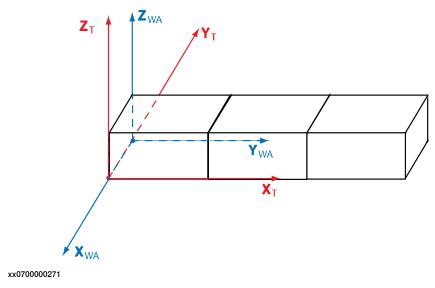


9.9 Tune frame versus work area frame *Continued*



9.9 Tune frame versus work area frame *Continued*

Alignment: Right; Orientation: 90



9.10 Tune frame versus work area frame versus displacement frame

9.10 Tune frame versus work area frame versus displacement frame

Displacement frame settings

Displacement frame settings are made in the Group Operation Set Configuration window for formats and in the Pattern Operation Set Configuration window for pallet patterns.

x, **y**, and **z** defines an offset of the displacement frame relative to the work area frame but in the direction of the tune frame. **z** angle defines a z-rotation of the displacement frame relative the displaced origin.

Example

This example shows the coordinates for the tune frame, the work area frame and the displacement frame. It also shows how the items in the format are placed.

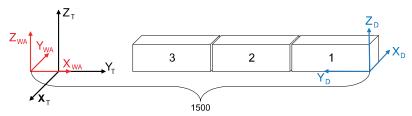
The following settings are made for the work area frame in the **Group Operation Set Configuration** window:

Setting	Value
Alignment	Right
Orientation	270

The following settings are made for the displacement frame in the **Pattern Operation Set Configuration** window:

Setting	Value
x	0
У	1500
z	0
z angle	90°

The displayed position of the format in this example assumes zero search frame offset.



xx0700000272

 X_T, Y_T, Z_T Coordinates for the tune frame

 X_{WA}, Y_{WA} , Coordinates for the work area frame

Z_{WA}

 X_D, Y_D, Z_D Coordinates for the displacement frame

This page is intentionally left blank

Index

Α

actions definition, 22 activators definition, 22

С

continue pick-place, 248 controller system handling, 271

D

default signals configuration, 260 controller system actions, 260 controller system handling, 271 error reporting, 278 event reporting, 261, 278 flow handling, 275 flows, 261 grippers, 261 installed, 260 project handling, 260, 271 work areas, 260 displacement frame, 388, 390, 395

Ε

errors flow recovery, 246 Production window, 247 events definition, 22 flow recovery, 246 Production window, 247

F

facing definition, 22 FlexPendant flow recovery, 246 opening project, 238 process signals, 252 restart options, 248 starting production, 239 tuning, 255 flows default signals, 275 definition, 22 handling, 275 recover actions, 275 recovering, 276 restarting, 276 starting, 276 status values, 275 stop options, 275 stopping, 276 stopping immediately, 276 format frame, 379, 385, 388

| |/O

basic I/O interface, 262 extended I/O interface, 271 signal descriptions, 262, 271 item frame, 385 items definition, 23 layouts definition, 23 line definition, 23 Ν next pallet, 248 0 operations definition, 23 Ρ palletizing process, 18 pallet pattern definition, 23 pallet pattern frame, 384, 390 pallet pattern operation sets definition, 23 pm_accdata, 336 pm acktype, 338 pm_actiondata, 339 pm_actionhandle, 342 pm actiontype, 343 pm_eventdata, 344 pm_eventtype, 346 pm_flowinfo, 347 pm_moduleinfo, 349 pm_movetype, 351 pm offsetdata, 352 pm operationdata, 354 pm_ophandle, 356 pm_projectinfo, 357 pm_robotinfo, 359 pm_searchdata, 361 pm_searchtype, 363 pm_singareatype, 364 pm_slavewainfo, 365 pm_stoptype, 367 pm targetdata, 368 pm_targethandle, 371 pm_wadescr, 372 pm_wainfo, 373 PmAckTarget, 288 PmCalcArmConf, 290 PmCalcIntermid, 319 PmGetEvent, 323 PmGetFlow, 293 PmGetFlowInfo, 295 PmGetOperation, 298 PmGetPathHeight, 325 PmGetProjectInfo, 300 PmGetQueByWobj, 302 PmGetTarget, 330 PmGetTgtAction, 332 PmGetWaHeight, 334 PmGetWalnfo, 303 PmGetWaName, 335 PmSearchAdjust, 305 PmSetRecoverAction, 308 PmStartFlow, 312 PmStartProj, 314

PmStopFlow, 315

PmStopProj, 317 PmWaitProjStart, 318 position sources definition, 23 process signals, 252 production starting overview, 45 starting with FlexPendant, 239 products packing, 19 project opening on FlexPendant, 238 projects default signals, 271 definition, 23 handling, 271 restarting, 274 starting, 272-273 stopping current, 274 stopping program, 273

R

RAPID action, 206 **DIToolEvent trap, 229** DoMove3 procedure, 231 event, 206 HomePos variable, 218 LastRobTgt variable, 232 LastTool variable, 232 Main procedure, 209, 220 MaxToolAngle constant, 212 MinToolAngle constant, 212 MoveHomePos procedure, 214 MoveInterMid procedure, 216 OperateLoop procedure, 210 Operate procedure, 211 operation, 206 pm_accdata data type, 336 pm_acktype data type, 338 pm actiondata data type, 339 pm_actionhandle data type, 342 pm_actiontype data type, 343 pm_eventdata data type, 344 pm_eventtype data type, 346 pm_flowinfo data type, 347 pm_moduleinfo data type, 349 pm_movetype data type, 351 pm_offsetdata data type, 352 pm operationdata data type, 354 pm_ophandle data type, 356 pm projectinfo data type, 357 pm_robotinfo data type, 359 pm_searchdata data type, 361 pm_searchtype data type, 363 pm_singareatype data type, 364 pm_slavewainfo data type, 365 pm_stoptype data type, 367 pm_targetdata data type, 368 pm_targethandle data type, 371 pm wadescr data type, 372 PmAckTarget instruction, 288 PmCalcArmConf instruction, 290

PmCalcIntermid function, 319 PmDoAction procedure, 212, 230 PmGetEvent function, 323 PmGetFlowInfo instruction, 295 **PmGetFlow instruction**, 293 PmGetLastWa, 297 PmGetOperation instruction, 298 PmGetPathHeight, 325 PmGetProjectInfo instruction, 300 PmGetQueByWobj instruction, 302 PmGetTarget function, 330 PmGetTgtAction function, 332 PmGetWaHeight function, 334 PmGetWalnfo, 303 PmGetWaName function, 335 PmSearchAdjust instruction, 305 PmSetDefaultHeight, 328 PmSetLastWa, 307 PmSetRecoverAction, 308 PmStartFlow instruction, 312 PmStartProj instruction, 314 **PmStopFlow instruction**, 315 PmStopProj instruction, 317 PmWaitProjStart instruction, 318 product, 207 SetLastPos procedure, 215 system modules, 208 target, 206 templates, 208 TrapGIToolEvents trap, 229 TrapProjectStopped, 222, 226 TrapSetRecoverAction, 224 TrapStartFlow, 225 TrapStopFlow, 226 walnFeeder1 variable, 212 waOutFeeder1, 212 redo last pick, 248 restart layer, 248 restart options, 248 robots bending backwards, 26 parallel rod, 29 using 6 axes, 25

S

safety, 13 search frame, 388, 390 shape frame, 378–379, 384 slip sheets using, 19

Т

target frame, 385 timing diagrams, 281 tool frame, 385, 387 tuning FlexPendant, 255

W

work area frame, 388, 390, 392, 395 work object, 388, 390, 392, 395

Ζ

zone frame, 385



ABB AB, Robotics Robotics and Motion S-721 68 VÄSTERÅS, Sweden Telephone +46 (0) 21 344 400

ABB AS, Robotics Robotics and Motion Nordlysvegen 7, N-4340 BRYNE, Norway Box 265, N-4349 BRYNE, Norway Telephone: +47 22 87 2000

ABB Engineering (Shanghai) Ltd. Robotics and Motion No. 4528 Kangxin Highway PuDong District SHANGHAI 201319, China Telephone: +86 21 6105 6666

ABB Inc. Robotics and Motion 1250 Brown Road Auburn Hills, MI 48326 USA Telephone: +1 248 391 9000

abb.com/robotics