ROBOTICS

Operating manual

RobotStudio

Trace back information:
Workspace 24A version a4
Checked in 2024-02-22
Skribenta version 5.5.019

Operating manual

RobotStudio

2024.1

Document ID: 3HAC032104-001
Revision: AS

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.

Table of contents

Overview of this manual 9
Product documentation 12
Safety 14
Network security 15

1 Getting Started 17

1.1 What is RobotStudio? 17
1.2 System Requirements 19
1.3 How to activate RobotStudio? 23
1.4 Connecting a PC to the controller 31
1.5 Network Settings 36
1.6 Managing user rights and write access on a controller 38
1.6.1 Managing user rights and write access on an IRC5 controller 38
1.6.2 Managing user rights and write access on an OmniCore controller 41
1.7 Manage user interface using mouse 43
1.8 Libraries, geometries and CAD files 53
1.9 How to install RobotWare and Add-Ins 56

2 Building Stations 59

2.1 Understanding stations and projects 59
2.2 Preparing the computer for hosting RobotStudio 61
2.3 Creating the station 65
2.4 Importing robots and related components 66
2.5 Creating a virtual controller 68
2.6 Synchronizing to virtual controller to create a RAPID program 70
2.7 Configuring station with robot and positioner 71
2.8 Configuring station with robot and Track motion 72
2.9 Configuring Conveyor tracking 73
2.10 Configuring external axis 76
2.11 Programming MultiMove systems 79
2.12 Replacing robot in a station 85
2.13 Back up a system 86

3 Programming robots in the 3D environment 89

3.1 Understanding offline programming 89
3.2 Robot axis configurations 95
3.3 Creating a workobject 98
3.4 Creating a path with targets and move instructions 99
3.5 Creating a path from an edge or curve 100
3.5.1 AutoPath 100
3.6 Creating a collision free path between two targets or move instructions 102
3.7 Configuring a stationary tool 104
3.8 Define arm configurations for the targets 105
3.9 Testing positions and motions 106
3.10 Generating the RAPID program 107
3.11 Modifying target orientations 108

4 Understanding RAPID editor 109

4.1 Working with RAPID editor 109
4.2 RAPID editor intellisense 111
4.3 Manage RAPID modules and programs 113
4.4 Adding code snippets 116
4.5 Inserting instructions from the list 118
4.6 Editing standalone files and backups 119
4.7 RAPID Data Editor 121

4.8 Creating custom instructions using Instruction Template 122
4.9 RAPID Path Editor 125
4.10 Applying and verifying the edits 127

5 Testing and Debugging RAPID 129

5.1 Debugging a task 129
5.2 Understanding program pointer 131
5.3 Working with RAPID breakpoints 133
5.4 Navigate through your program using RAPID Call Stack window 134
5.5 Using RAPID Watch window for debugging your RAPID code 135

6 Simulating programs 137

6.1 Playing Simulation 137
6.1.1 Simulation Setup 137
6.1.2 Simulation Control 139
6.2 Detecting Collision 140
6.3 Collision Avoidance 144
6.4 I/O Simulation 149
6.5 Simulation time measurement 152
6.6 Understanding TCP trace 153

7 Advanced RobotStudio simulations 155

7.1 Understanding SmartComponents 155
7.1.1 Smart Component 155
7.1.2 Basic Smart Components 157
7.2 The Infeeder example 181
7.3 Cable Simulation using Physics 192
7.4 Physics joints 194
7.5 Virtual commissioning using the SIMIT Smart Component 195
7.6 Virtual commissioning using the OPC UA Client Smart Component 204

8 Deploying and Sharing 213

8.1 Saving and loading RAPID programs and modules 213
8.2 Sharing a station 214
8.3 Capturing the selected screen 216
8.4 Recording a simulation 217
8.5 Creating a 3D animation of your simulation 218
8.6 Deploying a RAPID program to a robot controller 219

9 Installing robot controller software 223

9.1 Installation 223
9.1.1 About Installation 223
9.1.2 Using Modify Installation for RobotWare 7 224
9.1.3 Using Installation Manager for RobotWare 6 233
9.1.3.1 Startup and settings 233
9.1.3.2 Building a new robot controller 234
9.1.3.3 Modifying a robot controller 236
9.1.3.4 Copying a robot controller 238
9.1.3.5 Creating a robot controller from backup 240
9.1.3.6 Renaming a robot controller 242
9.2 Using System Builder for managing RobotWare 5 243
9.2.1 About System Builder 243
9.2.2 Viewing system properties 244
9.2.3 Building a new system 245
9.2.4 Modifying a system 249
9.2.5 Copying a system 253
9.2.6 Creating a system from backup 254
9.2.7 Downloading a system to a controller 255

9.2.8 Examples using the System Builder when offline 256
9.2.8.1 A system with support for one robot and one positioner external axis 256
9.2.8.2 Options settings for systems with positioners 259
9.3 A MultiMove system with two coordinated robots 260
9.3.1 Creating a coordinated system using System Builder 260
9.3.2 Creating a coordinated system using Installation Manager 262

10 Working with system parameters 263

10.1 System parameters 263
10.2 Adding instances 266
10.3 Copying an instance 267
10.4 Deleting an instance 268
10.5 Save one configuration file 269
10.6 Saving several configuration files 270
10.7 Loading a configuration file 271

11 Monitoring robot signals 273

11.1 Signal analyzer 273
11.2 Monitored signals 274
11.3 Recording signals 277
11.4 Recordings 278

12 Jobs 281

13 Screenmaker 287

13.1 Introduction to ScreenMaker 287
13.2 Development environment 290
13.3 Working with ScreenMaker 295
13.3.1 Managing projects 295
13.3.2 Application variables 313
13.3.3 Data binding 314
13.3.4 ScreenMaker Doctor 316
13.4 Frequently asked questions 319
13.5 Tutorial 322

14 RobotStudio® Cloud 331

15 Home tab 337

15.1 Virtual controller 337
15.2 Target 339
15.3 Path 343
15.4 Other 346
15.5 Virtual Reality 349

16 Modeling tab 351

16.1 Import Geometry 351
16.2 Create Mechanism 352
16.3 Create Tool 357

17 Simulation tab 359

17.1 Station Logic 359
17.2 TCP Trace 366

18 Controller tab 367

18.1 Add Controller 367
18.2 Authenticate 371
18.3 Events 380

18.4 Configuration 381
18.5 FlexPendant Viewer 387
18.6 Properties 388
18.7 Installation 393
18.8 Conveyor Tracking 402
18.9 Motion Configuration 409
18.10 Task Frames 411
18.11 Go Offline 412
18.12 Create Relation 413
18.13 Online monitor 415

19 RAPID tab **417**

19.1 Synchronize 417
19.2 Adjust Robtargets 420

20 Add-Ins tab **423**

20.1 Gearbox Heat Estimation 423

A Options **427**

B Terminology **435**

C Technical support **453**

Index **455**

Overview of this manual

About this manual

RobotStudio is a PC application for modeling, offline programming, and simulation of robot cells. This manual describes how to create, program and simulate robot cells and stations using RobotStudio. This manual also explains the terms and concepts related to both offline and online programming.

Usage

This manual should be used when working with the offline or online functions of RobotStudio.

Who should read this manual?

This manual is intended for RobotStudio users, proposal engineers, mechanical designers, offline programmers, robot technicians, service technicians, PLC programmers, Robot programmers, and Robot System integrators.

Prerequisites

The reader should have basic knowledge of:

- Robot programming
- Generic Windows handling
- 3D CAD programs

Organization of chapters

The operating manual is structured into the following chapters:

Chapter		Contents
1	Getting Started	Contains installation, activation and licensing information, introduction to feature levels, network settings and user rights.
2	Building Stations	Contains information on how to prepare PC for hosting RobotStudio, introduction on project and stations and building stations. This includes importing and configuring the equipment to be simulated, as well as testing the reachability for finding the optimal station layout.
3	Programming robots in the 3D environment	Describes how to create robot movements, I/O signals, process instructions and logics in a RAPID program for the robots. It also describes how to run and test the program.
4	Understanding RAPID editor	Describes how to simulate and validate robot programs.
5	Testing and Debugging RAPID	Describes how to debug a task, introduces program pointer, RAPID breakpoints. RAPID Call Stack window and using RAPID Watch window for debugging code.
6	Simulating programs	Describes simulation setup, playing simulation, collision detection, collision avoidance and I/O simulation.

Continues on next page

Continued

Chapter		Contents
7	Advanced simulation using smart components and physics	Introduces advance features such as Smart Components, physics simulation and virtual commissioning using SIMIT.
8	Deploying and Sharing	Describes how to pack and distribute a station.
9	Installing robot controller software	Contains instructions to work with System builder and Installation Manager.
10	Working with system parameters	Introduces system parameters and contains instruction for managing instances and configuration files
11	Monitoring robot signals	Introduces signal analyzer and contains instructions for managing signal recordings.
12	Jobs	Introduces the Jobs feature and contains instructions for managing this feature.
13	Screenmaker	Describes the ScreenMaker development tool, how to manage projects in ScreenMaker and various menus and commands used in the application.

References

Reference	Document ID
Operating manual - OmniCore	*3HAC065036-001*
Technical reference manual - RAPID Overview	*3HAC065040-001*
Technical reference manual - System parameters	*3HAC065041-001*
Application manual - MultiMove	*3HAC089689-001*
Application manual - Conveyor tracking	*3HAC066561-001*
Application manual - Functional safety and SafeMove	*3HAC066559-001*
Application manual - Electronic Position Switches	*3HAC050996-001*
Application manual - Integrated Vision	*3HAC067707-001*
Application manual - Controller software OmniCore	*3HAC066554-001*

Revisions

Revision	Description
A	First revision, called RobotStudio 2008, released for Partner Days. The entire manual has been adapted to the new GUI, in which RobotStudio ^{Online} has been integrated.
B	Released with RobotStudio 5.12.
C	Released with RobotStudio 5.13.
D	Released with RobotStudio 5.13.02.
E	Released with RobotStudio 5.14.
F	Released with RobotStudio 5.14.02.
G	Released with RobotStudio 5.14.02.01.
H	Released with RobotStudio 5.14.03.
J	Released with RobotStudio 5.15.

Continues on next page

Revision	Description
K	Released with RobotStudio 5.15.01.
L	Released with RobotStudio 5.60.
M	Released with RobotStudio 5.61.
N	Released with RobotStudio 6.0.
P	Released with RobotStudio 6.01.
Q	Released with RobotStudio 6.02.
R	Released with RobotStudio 6.03
S	Released with RobotStudio 6.03.01.
T	Released with RobotStudio 6.04.
U	Released with RobotStudio 6.05.
V	Released with RobotStudio 6.06.
W	Released with RobotStudio 6.07.
X	Released with RobotStudio 6.08.
Y	Released with RobotStudio 2019.1.
Z	Released with RobotStudio 2019.2.
AA	Released with RobotStudio 2019.3.
AB	Released with RobotStudio 2019.5.
AC	Released with RobotStudio 2020.1.
AD	Released with RobotStudio 2020.2.
AE	Released with RobotStudio 2020.3.
AF	Released with RobotStudio 2020.4.
AG	Released with RobotStudio 2021.1.
AH	Released with RobotStudio 2021.2.
AI	Released with RobotStudio 2021.3.
AJ	Released with RobotStudio 2021.4.
AK	Released with RobotStudio 2022.1.
AL	Released with RobotStudio 2022.2.
AM	Released with RobotStudio 2022.3.
AN	Released with RobotStudio 2022.3.1.
AO	Released with RobotStudio 2023.1
AP	Released with RobotStudio 2023.2
AQ	Released with RobotStudio 2023.3
AR	Released with RobotStudio 2023.4
AS	Released with RobotStudio 2024.1

Product documentation

Categories for user documentation from ABB Robotics

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

Tip

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

Product manuals

Manipulators, controllers, DressPack, and most other hardware is delivered with a **Product manual** that generally contains:

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

Technical reference manuals

The technical reference manuals describe reference information for robotics products, for example lubrication, the RAPID language, and system parameters.

Application manuals

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

An application manual generally contains information about:

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

Continues on next page

- Examples of how to use the application.

Operating manuals

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

Safety

Safety of personnel

A robot is heavy and extremely powerful regardless of its speed. A pause or long stop in movement can be followed by a fast hazardous movement. Even if a pattern of movement is predicted, a change in operation can be triggered by an external signal resulting in an unexpected movement.

Therefore, it is important that all safety regulations are followed when entering safeguarded space.

WARNING

Program changes should always be validated and tested before entering production, to protect humans and property. Ensure it is possible to stop the robot with a protective stop device.

Safety regulations

Before beginning work with the robot, make sure you are familiar with the safety regulations described in the manual *Safety manual for robot - Manipulator and IRC5 or OmniCore controller*.

Network security

Network security

This product is designed to be connected to and to communicate information and data via a network interface. It is your sole responsibility to provide, and continuously ensure, a secure connection between the product and to your network or any other network (as the case may be).

You shall establish and maintain any appropriate measures (such as, but not limited to, the installation of firewalls, application of authentication measures, encryption of data, installation of anti-virus programs, etc) to protect the product, the network, its system and the interface against any kind of security breaches, unauthorized access, interference, intrusion, leakage and/or theft of data or information. ABB Ltd and its entities are not liable for damage and/or loss related to such security breaches, any unauthorized access, interference, intrusion, leakage and/or theft of data or information.

This page is intentionally left blank

1 Getting Started

1.1 What is RobotStudio?

Overview

RobotStudio is an engineering tool for configuring and programming ABB robots, both physical robots on the shop floor and virtual robots in a PC. Use this application for modeling, offline programming, and simulation of robot cells. Its advanced modeling and simulation features help in visualizing multi robot control, safety features, 3D vision, and remote robot supervision.

RobotStudio's built-in programming environment allows *online* and *offline* programming of robot controllers. In online mode, it is connected to a *robot controller* and in offline mode, it is connected to a *virtual controller* that emulates a robot controller in a PC.

RobotStudio is downloadable from http://new.abb.com/products/robotics/robotstudio/downloads. The 30 day basic(trial) version is free and offers full functionalities including CAD Converters. CAD Converters are not part of the Premium license and it requires additional options to be purchased. The Premium version offers complete functionalities and requires activation. To purchase a Premium license, contact your local ABB Robotics sales representative at www.abb.com/contacts.

RobotStudio feature levels

RobotStudio features are categorized into basic and premium levels. Administrator privileges on the PC is mandatory to install RobotStudio.

- **Basic** - Offers selected RobotStudio functionality to configure, program, and run a *virtual controller.* It also includes online features for programming, configuring, and monitoring a *robot controller* connected over Ethernet and it does not require activation. In the Basic Functionality mode, which is a reduced functionality mode, RobotStudio allows the usage of basic features only for both robot and virtual controllers. No existing files or *stations* are affected in this mode.
- **Premium** - Offers full RobotStudio functionality for *offline programming* and simulation of multiple robots. The Premium level includes features of the basic level and it requires activation. To purchase a Premium license, contact your local ABB Robotics sales representative at *www.abb.com/contacts*.

Continues on next page

Details of Premium and Basic features

The following table lists the features provided with Basic and Premium *licenses*.

Feature	Basic	Premium
Necessary features for commissioning a real or virtual robot[1], such as: • System Builder • Event Log Viewer • Configuration Editor • RAPID Editor • Backup / Restore • I/O Window	Yes	Yes
Productivity features, such as: • *RAPID* Data Editor • RAPID Compare • Adjust *robtargets* • RAPID Watch • RAPID *Breakpoints* • Signal Analyzer • *MultiMove* tool • ScreenMaker[1,2] • Jobs		Yes
Elementary offline features, such as: • Open *station*[4] • Unpack & Work[4] • Run Simulation[4] • Go *Offline* • Robot jogging tools • Gearbox heat prediction • *ABB Library* of robots	Yes	Yes
Advanced offline features, such as: • Graphical programming • Save station • *Pack & Go* • Import / Export *Geometry* • Import Library • Create Export Viewer and movies • Transfer • AutoPath • Collision Free Path • 3D operations		Yes
Add-Ins[3]	Yes	Yes

1. Requires the RobotWare feature **RobotStudio Connect** *for OmniCore (or* **PC-Interface for IRC5***) on the robot controller to enable WAN communication. This option is not needed for connection via the Management port or for virtual controller.*

2. Requires the RobotWare feature **FlexPendant Interface** *on the IRC5 robot controller. This feature is not available in OmniCore. ScreenMaker is not available for OmniCore.*

3. Add-ins which does not use the Stations API can be loaded in Basic mode.

4. Smart Components included in the station that are using the Station API or having physics behaviour will not be simulated in basic mode.

1.2 System Requirements

Overview

Before installing RobotStudio, ensure that the computer meets the following hardware and software requirements.

Hardware

High-performance desktop or laptop workstation, with the following requirements:

Part	Requirement
CPU	2.0GHz or faster processor, multiple cores recommended
Memory	8 GB minimum. 16 GB or more if working with heavy CAD models.
Disk	10+ GB free space, solid state drive (SSD) recommended.
Graphics card	High-performance, DirectX 11 compatible, gaming graphics card from any of the leading vendors. For the Advanced lightning mode Direct3D feature level 10_1 or higher is required.
Display settings	1920 x 1080 pixels or higher resolution is recommended.
dots per inch (dpi)	Only Normal size supported for Integrated Vision.
Mouse	Three-button mouse.
3D Mouse [optional]	Any 3D mouse from 3DConnexion. See http://www.3dconnexion.com.

RobotStudio supports *RobotWare* version 5.07 up to the latest released version, including revisions. See RobotStudio Release Notes for any compatibility limitations.

You can connect RobotStudio to a *robot controller* either through its service port or over Ethernet.

- To connect RobotStudio over Ethernet:
  - For IRC5 controllers the RobotWare option **616-1 PC Interface** is required.
  - For OmniCore controllers the RobotWare option **3119-1 RobotStudio Connect** is required.
- To run ScreenMaker or FlexPendant SDK applications on an IRC5 controller, the RobotWare option **617-1 FlexPendant Interface** is required.

Software

Operating system	Description
Windows 10 Anniversary Update or later	64-bit edition

It is recommended to run Windows updates to get the latest updates to Windows before installing and running RobotStudio. Windows Firewall can block certain features that are necessary to run RobotStudio, which must be unblocked as required. You can view and edit the state of a program at **Start** > **Control Panel** >

Continues on next page

Windows Firewall. For more information on Windows Firewall, *visit www.microsoft.com*.

Firewall settings

The firewall settings are applicable to robot and virtual controllers. The following table describes the necessary firewall configurations:

Status	Name	Action	Direction	Protocol	Remote Address	Local Service	Remote Service	Application
	RobNetScanHost	Allow	Out	UDP/IP	Any	Any	5512,5514	RobNetScanHost.exe
	IRC5Controller	Allow	In	UDP/IP	Any	5513	Any	RobNetScanHost.exe
	RobComCtrlServer	Allow	Out	TCP/IP	Any	Any	5515	RobComCtrlServer.exe
	RobotFTP	Allow	Out	TCP/IP	Any	Any	FTP(21)	Any
	RobotStudio	Allow	Out	HTTP	Any	Any	80	RobotStudio.exe
	RobotStudio	Allow	Out	HTTPS	Any	Any	443	RobotStudio.exe
	RobICI (CTM) Discovery	Allow	In	UDP	Any	18943	Any	RobotStudio.exe
	RobICI (CTM) Communication	Allow	Out	UDP	Any	Any	34981	RobotStudio.exe
	RobICI (CTM) Communication	Allow	Out	TCP	Any	Any	34981	RobotStudio.exe
	SFTP/SSH (CTM Configuration/Backup)	Allow	Out	TCP	Any	Any	22	RobotStudio.exe

The following table describes the necessary firewall configurations for the RobotWare option, Integrated Vision:

Status	Name	Action	Direction	Protocol	Remote Address	Local Service	Remote Service	Application
	Telnet	Allow	Out	TCP/IP	Any	Any	23	RobotStudio.exe
	In-Sight Protocol	Allow	Out	TCP/IP	Any	Any	1069	RobotStudio.exe
	In-Sight Discovery	Allow	In/Out	UDP/IP	Any	1069	1069	RobotStudio.exe
	Upgrade port (PC only)	Allow	Out	TCP/IP	Any	Any	1212	RobotStudio.exe
	DataChannel	Allow	Out	TCP/IP	Any	Any	50000	RobotStudio.exe

Continues on next page

Note

RobotStudio uses the current Internet Options, HTTP, and proxy settings.

The following table describes the necessary firewall configurations for SLP Distributor.

Status	Name	Action	Direction	Protocol	Remote Address	Local Service	Remote Service	Application
	Software Potential Distributor (for the web interface)	Allow	In	TCP/IP	Any	2468	Any	Slps.Distributor.Host.exe
	Software Potential Distributor (licensing)	Allow	In	TCP/IP	Any	8731	Any	Slps.Distributor.Host.exe
	Software Potential Distributor (activation)	Allow	Out	TCP/IP	Any	Any	443	Slps.Distributor.Host.exe

Services and processes

The following table describes the Windows Services.

Name	Application	Type	Startup Type	Account	Description
ABB Industrial Robot Communication Server	RobComCtrlServer.exe	Service	Manual	Local System account	Provides support for communicating with ABB robot controllers.
ABB Industrial Robot Discovery Server	RobNetScanHost.exe	Service	Manual	Local System account	Provides support for discovery of ABB robot controllers.

SLP Distributor is not part of the RobotStudio installation. It is optional and must be installed on a dedicated server. Install the SLP Distributor server on a dedicated PC accessible from the PCs where RobotStudio is going to be used.

The following table details SLP Distributor.

Name	Application	Type	Startup Type	Account	Description
Software Potential Distributor	Slps.Distributor.Host.exe	Service	Automatic	Local System account	Allocates Pooled Resources to Software Potential-protected applications.

The following table describes the applications that RobotStudio starts.

Application	Description
7z.exe	Compression/decompression command line tool.
cfree_server.exe	Used for collision free path planning.
CadConverter.exe	Converts between CAD formats.
comp.exe	ABB compression command line tool.
ConvexHullBuilder.exe	Creates convex hulls for collision avoidance and collision free path planning.

Continues on next page

Application	Description
decomp.exe	ABB decompression command line tool.
InstallationManager.exe	Creates and modifies RobotWare 6.x systems.
InstallationManager7.exe	Creates and modifies RobotWare 7.x systems.
PdfConverter.exe	Used by Visual SafeMove to create PDF files.
PoissonRecon.exe	Reconstructs meshes from point clouds. Used for swept volume generation.
RobotDiskRecovery.Exe	Creates robot recovery disks.
RobotStudio.Installer.exe	Performs actions that require elevation, such as initialization of license storage.
RobVC.exe	The ABB virtual controller.
RSSystemInfo.exe	RobotStudio Support Tool
RobotStudio.FleetManagement.JobRunner.exe	Executes jobs that communicate with multiple robot controllers.
RunJob.exe	A command line tool for executing jobs using *RobotStudio.FleetManagement.JobRunner.exe.*
ScreenMaker.exe	Creates FlexPendant user interfaces for RobotWare 5-6.x systems.
SystemBuilder.exe	Creates RobotWare 5 systems.
tar.exe	Creates or extracts files from archives. Used for backups.
Virtual FlexPendant.exe	Virtual Flexpendant for RobotWare 5-6.x systems.
Vrchost64.exe	The 64-bit ABB virtual controller.

1.3 How to activate RobotStudio?

Activation of RobotStudio license

Activation of RobotStudio installation is a procedure for validating the RobotStudio *license*. To continue using the application with all of its features, it must be activated. RobotStudio product activation is based on the anti-piracy technology and is designed to verify that software products are legitimately licensed. Activation works by verifying that the *activation key* is not in use on more personal computers than are permitted by the software license.

When you start RobotStudio for the first time after installation, it prompts for the 25-digit activation key (xxxxx-xxxxx-xxxxx-xxxxx-xxxxx). The software performs in the Basic Functionality mode in the absence of a valid activation key. A successful activation entails the user with valid licenses for the features covered by the subscription.

Installing the trial license

Use the following procedure to request for the the 30 day trial version. An active Internet connection is required for this procedure.

1. Download and install RobotStudio from *http://new.abb.com/products/robotics/robotstudio/downloads*.
2. Start RobotStudio. On the **File** tab, click the **Help** section.
3. Under **Support**, click **Manage Licenses**. The **Options** dialog appears with the **Licensing** options.
4. Under **Licensing**, click **Activation Wizard** to view RobotStudio license options.
5. Select the **I want to request a trial license** option, click the **Next** button.

   RobotStudio installation dialog opens and starts to install the application, click **Finish** once the installation completes.

   RobotStudio will connect to the cloud service and request a trial license. A trial license can be requested once per PC.

Type of licenses

Type of license	Description
Multi-user	Allows you to centralize license management by installing *licenses* on a single server rather than on individual client machines. A Multi-user license allows a number of users on the same TCP/IP network to share access to product licenses. The server administers the licenses to the clients as required. A Multi-user license allows several clients to use the software. Multi-user licensing in RobotStudio uses the SLP Distributor server as the licensing server. Multi-user licenses are currently available only for authorized value providers and schools. The School and Value provider licenses fall into the multi-user category. SLP Distributor is installed as a Windows Service on a network server and manages concurrent licensing of RobotStudio. In addition to the licensing service endpoint, SLP distributor also provides a web interface for administration of services such as activating licenses, viewing usages and so on.
Standalone	Allows a single user to install and use RobotStudio on a single computer.

Continues on next page

Type of license	Description
Commuter	Allows a RobotStudio user to work offline from the multi-user license server. You can check out a license from the server for a specified number of days. During this period the checked out license is unavailable to other users. The commuter license is made available for other clients only when it is manually checked in back to the server.

Understanding the activation key

When RobotStudio starts for the first time after installation, it prompts for the 25-digit activation key (xxxxx-xxxxx-xxxxx-xxxxx-xxxxx). The activation key for a standalone license must be activated inside RobotStudio and the activation key for a multi-user license must be activated in the SLP Distributor on the license server. Read the *Subscription confirmation mail* that ABB sends before activating RobotStudio installation.

Multi-user license logging

SLP distributor provides option for logging the Multi-user license usage. License usage can be logged in the distributor such that it can be manually retrieved for subsequent analysis. This feature is not enabled by default and must be enabled by uncommenting the `<add key="Slps.Distributor.Service.EnableUsageLogging" value="true">` setting in the `Slps.Distributor.Services.dll.config` file located in *C:\Program Files (x86)\ABB\SLP.Distributor.Host\Services*.

Note

Windows may prevent the editing of `Slps.Distributor.Services.dll.config` file in its original location. Hence, copy the file to the user directory for editing before replacing the original file with the edited file.

When enabled, Distributor logging may have a minor impact on the performance of the server and it keeps a sliding window of 45 daily logs, with a maximum size of 45MB. The usage logs are available in a text file in *\Slp.Distributor.Host\Services\Usage*.

Activation Wizard

The Activation Wizard tool provides activation options during RobotStudio installation. It provides two activation modes, automatic activation over Internet or manual activation. When RobotStudio starts for the first time after installation, the wizard starts automatically and prompts for the *activation key*. RobotStudio can be activated after installation or later using the wizard. Use the following steps to start the Activation Wizard tool.

1. Click the **File** tab, and then click the **Help** section.
2. Under **Support**, click **Manage Licenses**. The **Options** dialog appears with the **Licensing** options.
3. Under **Licensing**, click **Activation Wizard** to view RobotStudio license options.

Continues on next page

What are Installation Options?

RobotStudio offers three Installation options, *Minimal*, *Custom* and *Complete*. Activation is not required for Minimal installation, or for Basic functionality modes of *Complete* or *Custom* installation.

- *Minimal* - Installs features to program, configure, and monitor a *robot controller* connected over Ethernet.
- *Custom* - Installs user-customized features and excludes selected robot *libraries*.
- *Complete* - Installs complete RobotStudio and includes additional features of Basic and Premium functionalities.

Activating standalone license without Internet access

On a computer with Internet access, RobotStudio gets activated automatically. Automatic activation requires a working Internet connection and a valid *activation key* that has not exceeded the allowed number of installations. In the absence of Internet access, the product must be activated manually. Restart RobotStudio after activation.

1. Create a license request file by selecting the option **Step 1:Create a license request file**.

   Proceed through the wizard, enter the *activation key* and save the license request file to your computer.
2. Use removable storage, such as a USB stick, to transfer the file to a computer with an Internet connection. On that computer, open a web browser, go to *http://manualactivation.e.abb.com/* and follow the instructions given.

   A license key file gets created, save this file and transfer it to the computer hosting the installation.
3. Restart the activation wizard and go through the steps until the *Activate a Standalone License* page.
4. Under *Manual Activation*, select the option **Step 3:Install a license file**.

   Proceed through the wizard, select the license key file when requested. On completion, RobotStudio is activated and ready for use.

RobotStudio must be restarted after activation.

Activating Multi-user license

Install the SLP Distributor server on a dedicated PC accessible from the PCs where RobotStudio is going to be used.

1. Install the SLP Distributor server from the *SLP Distributor* directory of the RobotStudio distribution. The SLP Distributor server is installed as a service that starts automatically with Windows. It requires two open TCP ports, by default 2468 (for the web interface) and 8731 (for licensing). The installer

Continues on next page

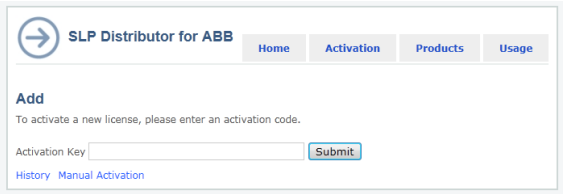
optionally opens these ports in the standard Windows firewall, but any third-party firewall must be configured manually by the system administrator.

Note

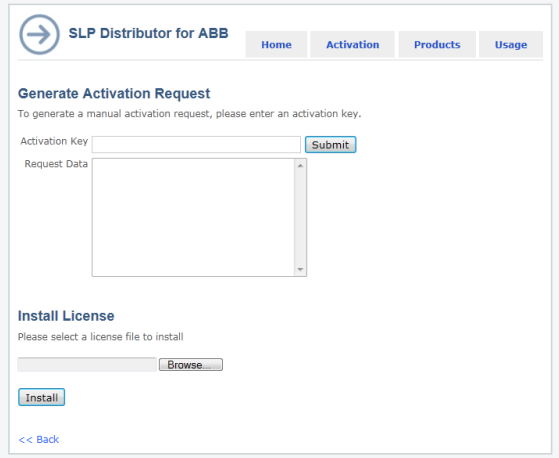
Refer to the latest RobotStudio Release Notes for more information on the installer requirements.

2 Activate the licenses for Multi-user licensing .

Once the SLP server is online, you can access its web interface at *http://<server>:2468/web*. The following table shows how to use the server's web interface.

To...	Use...
Activate a Multi-user license automatically (for PCs with Internet connection)	The **Activation** tab. Type in the **Activation Key** provided by ABB, and then click **Submit**. The number of concurrent users that get activated depends on the *activation key*provided.  xx1300000052

Continues on next page

To...	Use...
Activate a Multi-user license manually (for PCs *without* Internet connection)	The **Activation** tab. a Click **Manual Activation**. b Type in your activation key provided by ABB, and then click **Submit**. c Save the file to a removable storage, such as a USB stick, then transfer the file to a machine with an Internet connection. On that machine, open a web browser and browse to *http://manualactivation.e.abb.com/* and follow the instructions given. A license key file gets generated and will be saved. This file will be returned to the machine which hosts the installation. d After receiving the license file, click **Browse** to upload and install the license file. Your Multi-user license is now activated.  xx1300000051
View the installed licenses	In the **Home** tab, under **Dashboard**, click **Details**. Alternatively, click the **Products** tab. The **Product details for RobotStudio** page opens which shows the installed licenses details.
View the usage of licenses	On the **Home** tab, under **Dashboard**, click **Usage**. Alternatively, click the **Usage** tab. The *Current usage of RobotStudio* page opens in which the following details are tabulated. • Licenses which are currently allocated • Client to which each license is allocated to • Number of remaining licenses available for use Each table row corresponds to one client system.

3 Set up the client for Multi-user licensing.

Use RobotStudio Activation Wizard in the client system for setting up Multi-user license. Use this procedure to set up Multi-user license for a client system.

	Action
1	On the **File** tab, click **Options** and go to **General:Licensing**.

Continues on next page

	Action
2	On the *Licensing* page to the right, click **Activation wizard** to start the Activation Wizard.
3	In the Activation Wizard, on the *Activate RobotStudio* page, choose the option **I want to specify a Multi-user license server and manage server license**, and then click **Next**. You will proceed to the *License Server* page.
4	Specify the name or IP address of the License Server, and then click **Finish**. If Windows UAC is enabled, a confirmation dialog appears. This prompts to restart RobotStudio to start using the specified server. To go to the SLP Distributor server web interface, click the **Open the server dashboard** link. **Note** To apply the changes restart RobotStudio.

Note

For Multi-user Licensing to work, the client system should be online with the server.

Tip

Multi-user licenses are displayed as *Multi-user* in the **View Installed Licenses** link of the *Licensing* page.

Activating commuter licenses

A commuter license key can be checked out if a PC has to be disconnected from the network. Normally, the PC possessing a multi-user license key must remain connected through the network to the License Server.

A commuter license key allows the PC to be disconnected from the network. The commuter license expires when the check out time expires. You may check in the license after use, and it becomes immediately available for other users.

It is not possible to check out specific features in the license. All features in a license are included when it is checked out. Use the activation wizard to check in/check out a commuter license.

	Action
1	On the **File** menu, click **Options** and select **General: Licensing**.
2	On the *Licensing* page to the right, click **Activation wizard** to start the Activation Wizard.
3	In the Activation Wizard, on the *Activate RobotStudio* page, choose **I want to check out or check in a commuter license** and click **Next**. The *Commuter License* page opens.

Continues on next page

	Action
4	In the *Commuter License* page, select one of the following options as per the requirement: • **Check out a commuter license** - In the **Check out days** box specify the number of days for keeping the license. This option is disabled if a commuter license has already been checked out. • **Check in a commuter license** - Choose this option to return the currently checked out license to the server. This option is enabled only if a commuter license is already checked out. If so, the expiry date and time of the license gets displayed.
5	Click **Finish** to complete the check in/check out.

Tip

Multi-user licenses that are checked-out as commuter licenses will be displayed as Floating (checked out) in the *View Installed Licenses* link of the *Licensing* page.

Verifying RobotStudio activation

1. On the **File** tab, click **Options**, and then select the **Licensing** section.
2. Click **View Installed License Keys** to see the status of the current license. On successful activation, valid licenses for the features covered by the subscription gets displayed here.

Activating RobotStudio® Cloud

RobotStudio Premium license includes the RobotStudio® Cloud subscription. This must be activated directly from the **Activation Wizard**.

Note

To activate RobotStudio® Cloud subscription, user must have an account in the myABB Business Portal, *www.abb.com/myABB*.

Activating the RobotStudio® Cloud subscription

1. Click the **File** tab, and then click the **Help** section.
2. Under **Support**, click **Manage Licenses**. The **Options** dialog appears with the **Licensing** options.
3. If you have not activated your RobotStudio license, click the **Activation Wizard** and follow the instructions.
4. If your RobotStudio license includes the RobotStudio® Cloud subscription, a **Sign In** button will be visible in RobotStudio.
5. Click the **Sign In** button, and enter your credentials in the **Sign in to your account** dialog.

   If the signing in is successful, the RobotStudio *Subscription key* and your *username(email address)* gets displayed.
6. Click the **Activate** button.

   If the activation is successful, you can see the **Activated** icon and the **View My Cloud Projects** link. Click this link to open RobotStudio® Cloud.

Continues on next page

Alternatively, **Activation Wizard** can be accessed using the following option.

- From the **Options** page, click **Options:General:Licensing** and then click the **Activation Wizard** button.

Demo Stations

RobotStudio provides a set of demo *stations* to help users. Users can open these stations for understanding the basic structure of a generic station. These stations are stored as *pack&go* files and can be downloaded. An active Internet connection is required for downloading the demo stations.

To open a trial station:

1. Start RobotStudio.
2. On the **File** tab, click **Open**, under **Samples** click **Demo Stations** to view available demo stations.
3. Click a demo station to unpack and open. When you open a demo station for the first time, it will be downloaded.

RobotStudio notifications

RobotStudio's in-app notifications notify users about the new updates. These notifications include updates for RobotStudio and add-ins, new and updated simulation models and license information. User can open these notification pop-ups by clicking the Notification icon in the title bar. The icon has a blue dot if there are any new (unread) notifications.

1.4 Connecting a PC to the controller

Overview

There are two ways of physically connecting a PC to the controller, to the service port or to the factory network port.

The service port

The service port is intended for service engineers and programmers connecting directly to the controller with a PC. The service port is configured with a fixed IP-address, which is the same for all controllers and cannot be changed, and has a DHCP server that automatically assigns an IP-address to the connected PC.

The factory network port

The factory network port is intended for connecting the controller to a network. The network settings can be configured with any IP-address, typically provided by the network administrator.

Limitations

Note

The maximum number of connected network clients is:

- LAN port: 3
- Service port: 1
- FlexPendant: 1

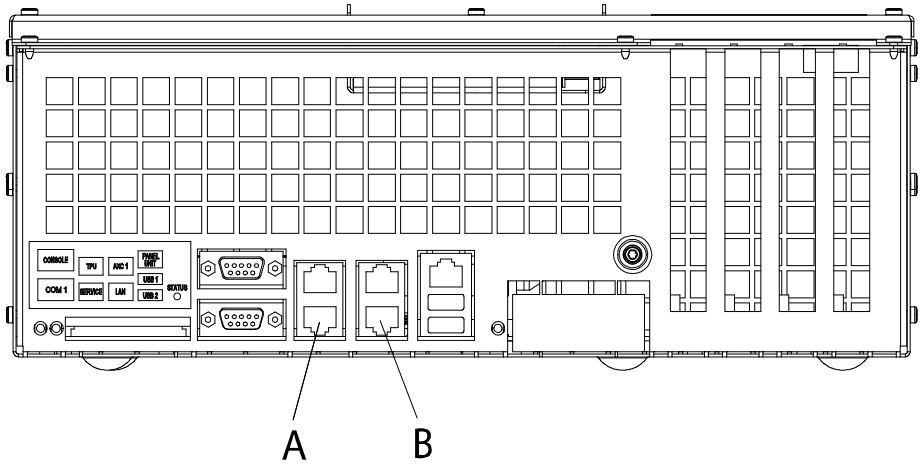
The maximum number of applications running on the same PC which is connected to one controller has no built-in maximum.

However, for a controller with RW 7 *UAS* limits the number of users to 123. For a controller with RW 6, *UAS* limits the number of users to 100. The maximum number of concurrently connected FTP clients is 4.

Continues on next page

Ports on the computer unit DSQC 639

The illustration below shows the two main ports on the computer unit DSQC 639, the service port and the LAN port.



connecti

A	Service port on the computer unit (connected to the service port on the controller front through a cable).
B	LAN port on the computer unit (connects to the factory network).

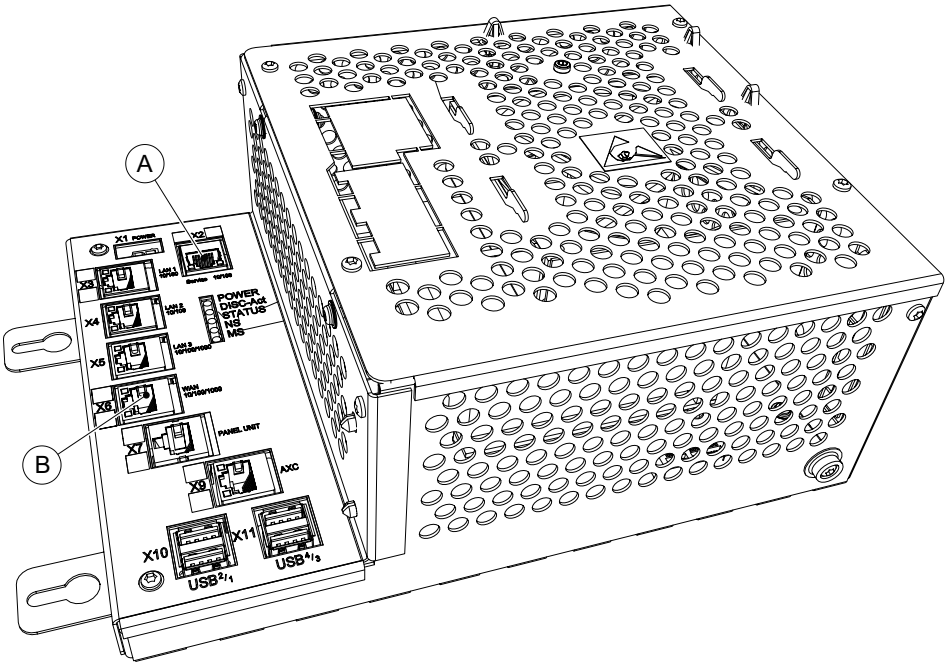
Note

The LAN port is the only public network interface to the controller, typically connected to the factory network with a public IP-address provided by the network administrator.

Continues on next page

Ports on the computer unit DSQC1000/DSQC1018/DSQC1024

The illustration below shows the two main ports on the computer unit DSQC1000/DSQC1018/DSQC1024, the service port and the WAN port.



xx1300000609

A	Service port on the computer unit (connected to the service port on the controller front through a cable).
B	WAN port on the computer unit (connects to the factory network).

Note

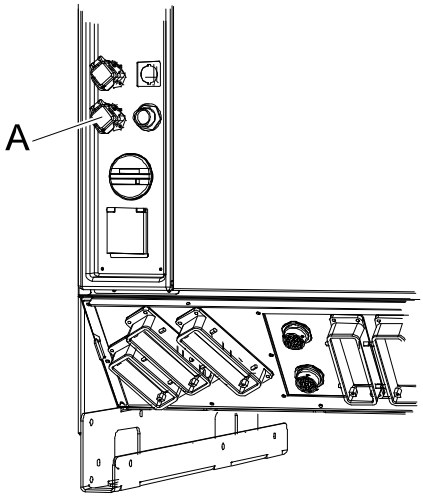
The WAN port is the only public network interface to the controller, typically connected to the factory network with a public IP-address provided by the network administrator. LAN1, LAN2, and LAN3 can only be configured as private networks to the IRC5 controller.

Connecting a PC to the controller

	Action	Note
1	Make sure that the network setting on the PC to be connected is correct. **When connecting to the service port:** • The PC must be set to *Obtain an IP address automatically* or set as described in **Service PC Information** in the **Boot Application** on the FlexPendant. **When connecting to the factory network port:** • The network settings for the PC depend on the network configuration setup by the network administrator.	Refer to the system documentation for your PC, depending on the operating system you are running.

Continues on next page

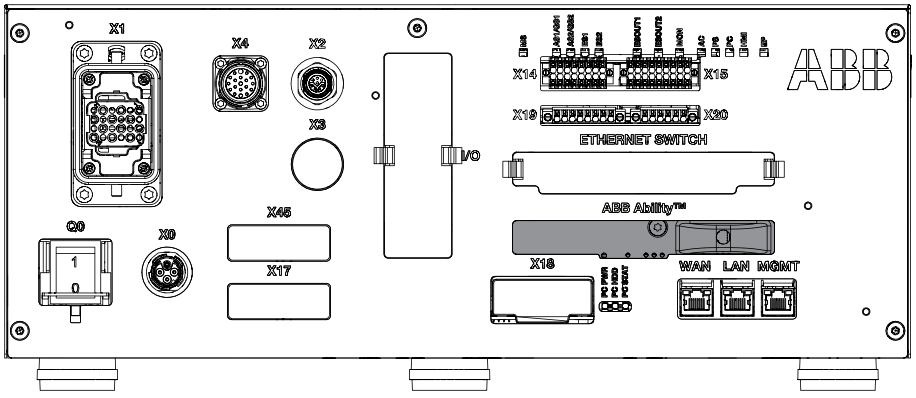
1.4 Connecting a PC to the controller
Continued

	Action	Note
2	Connect a network cable to the network port of your PC.	
3	**When connecting to the service port:** • Connect the network cable to the service port on the controller, or to the service port on the computer unit. **When connecting to the factory network port:** • Connect the network cable to the factory network port on the computer unit.	 connectb A Service port on the controller

Connecting RobotStudio to OmniCore

Connectors in OmniCore

The following diagram shows the connectors on the front panel of the OmniCore controller.



xx1800000823

MGMT port

The MGMT port(management port) is intended for service engineers and programmers connecting directly to the controller with a PC.

The management port is configured with the fixed IP address 192.168.125.1, which is the same for all controllers and cannot be changed, and has a DHCP server that automatically assigns an IP address to the connected PC.

Note

Do not connect another DHCP server to this port.

Continues on next page

WAN port

The WAN port is a public network interface to the controller, typically connected to the factory network with a public IP address provided by the network administrator.

The WAN port can be configured with fixed IP address, or DHCP, from **RobotStudio** or the FlexPendant. By default the IP address is blank.

Some network services, like FTP and RobotStudio, are enabled by default. Other services are enabled by the respective RobotWare application.

Note

The WAN port cannot use any of the following IP addresses which are allocated for other functions on the controller:

- 192.168.125.0 - 255
- 192.168.126.0 - 255
- 192.168.127.0 - 255

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

LAN port

The I/O Network is needed when an Industrial Ethernet network must be isolated from the Public Network. The LAN port is connected to the controller's I/O Network and is intended for connecting the robot controller to a factory wide industrial network isolated from WAN.

A factory wide I/O Network should be connected to the WAN port on the controller, or to the LAN port if the I/O network needs to be isolated from the network already connected to WAN.

1.5 Network Settings

Prerequisites

The PC can be connected to the controller through an Ethernet network in the following ways:

- Local network connection
- Service port connection
- Remote network connection

Local network connection

You can connect your PC to the same Ethernet network that the controller is connected to. When the PC and the controller are connected correctly and to the same subnet, the controller will be automatically detected by RobotStudio. The network settings for the PC depend on the network configuration. Contact the network administrator for setting up the PC.

Service port connection

When connecting to the controller's service port, obtain an IP address for the PC automatically, or specify a fixed IP address. Contact the network administrator for setting up the service port connection.

Automatic IP address

The controller's service port has a DHCP server that will automatically give the PC an IP address if it is configured for this. For detailed information, see the Windows help on configuring TCP/IP.

Fixed IP address

Instead of obtaining an IP address automatically, you can also specify a fixed IP address on the PC you connect to the controller.

Use the following settings for a fixed IP address:

Property	Value
IP address	192.168.125.2
Subnet mask	255.255.255.0

For detailed information about how to set up the PC network connection, see Windows help on configuring TCP/IP.

Note

Obtaining an IP address automatically might fail if the PC already has an IP address from another controller or Ethernet device. To ensure the accuracy of the IP address if the PC was connected to an Ethernet device, do one of the following:

- Restart the PC before connecting to the controller.
- Run the command `ipconfig /renew` from the command prompt after connecting the PC to the controller.

Continues on next page

Remote network connection

To enable connection to the controller on a remote subnet or over the local network, the relevant network traffic must be allowed through any firewall between the PC and the controller. The firewall must be configured to accept the following TCP/IP traffic from the PC to the controller:

- UDP port 5514 (unicast)
- TCP port 5515
- Passive FTP

All TCP and UPD connections to remote controllers are initiated by the PC, that is, the controller only responds on the given source port and address.

Connecting to the controller

1. Make sure the PC is connected to the controller's service port and that the controller is running.
2. On the **File** menu, click **Online** and then select **One Click Connect**.

   The **Controller** tab opens.
3. Click **Add Controller**.
4. Click **Request Write access**.

If the controller is in mode	Then
Auto	You will now get *Write Access* if it is available.
Manual	A message box on the FlexPendant will allow you to grant remote *Write Access* to RobotStudio.

1.6 Managing user rights and write access on a controller

1.6.1 Managing user rights and write access on an IRC5 controller

Overview

User Authorization System (UAS) restricts user access to the controller data and functionalites. These functionalities are categorized and protected by UAS grants. There are two types of grants; controller grants and application grants. Controller grants are predefined and provided by *RobotWare*. Application grants are defined by RobotWare add-ins. These grants are managed using the **UAS Administration Tool**.

UAS grants are viewable using the UAS grant viewer. The *UAS Grant Viewer* page displays information about the grants of the current user. In the **Authenticate** menu, click **UAS Grant Viewer** to open the viewer.

Group

Group is a collection of grants that represents user roles. The available user roles are administrator, programmer, operator and user defined. User inherits the grants of the group it is associated to.

All the controllers have a preset group and preset user named *Default Group* and *Default User* respectively. The *Default User* has an open password *robotics*. The *Default Group and User* cannot be removed and the password cannot be changed. However, the user with the user grant *Manage UAS settings* can modify the controller grants and application grants of the default user.

You can deactivate the *Default User* except for RobotWare 6.04 and earlier. Before deactivating the default user, it is recommended to define at least one user with the grant *Manage UAS settings* so as to continue managing users and groups.

Write access

Write access is required to change data on a controller. The controller accepts a single user with write access at a time. RobotStudio users can request write access to the system. If the system is running in manual mode, the request for write access is accepted or rejected on the FlexPendant. User loses write access if the mode changes from manual to automatic, or vice versa. If the controller is in manual mode, then the write access can be revoked from the FlexPendant.

Adding a user to the administers group

In addition to the *Default Group*, certain predefined user groups are available in the *robot controller*. The predefined groups are, *Administrator*, *Operator*, *Service* and *Programmer*. The Administrator group has the controller grant *Full access* enabled.

1. On the **Controller** tab, click **Add controller** and then click **Add Controller..** and then select the controller from the **Add Controller** dialog.
2. On the **Controller** tab, click **Request Write Access**.
3. Click **Authenticate** and then click **Edit User Accounts**.

   UAS Administration Tool opens.

Continues on next page

4 On the **Users** tab, click **Add**. The **Add new user** dialog opens.
5 In the **User Name** and **Password** boxes, enter suitable values. Click **OK**.
The new user gets added to the **Users on this controller** list.

> **Note**
>
> Valid characters that can be used for creating passwords are *abcdefghijklmnopqrstuvwxyzA BCDEFGHIJKLMNOPQRSTUVWXYZ_-1234567890,;.:!#%^()=?*@${[]}£*, space is also a valid character.

6 Select the user, and then from the User's groups, click the **Administrator** check box.
7 Click **OK**. The new user gets added to the **Administrator** group.

Use the same steps to create users for various groups.

> **Note**
>
> To view the Controller/Application grants assigned to a particular group, in the **UAS Administration Tool**, on the **Groups** tab, select the group and then select the particular category of grant.

Creating a new user group

1 In the **UAS Administration Tool**, click the **Groups** tab.
2 On the **Groups** tab, click **Add**. The **Add new group** dialog opens.
3 Enter the required details and click **OK**.
The new group gets added.

Modifying an existing user group

1 In the **UAS Administration Tool**, click the **Groups** tab.
2 On the **Groups** tab, Select the group and then click **Edit**. Enter the required changes and click **OK**.

Creating a new user

1 In the **UAS Administration Tool**, click the **Users** tab, and then click **Add**. The **Add new user** dialog opens.
2 In the **User Name** and **Password** boxes, enter suitable values. Click **OK**.
The new user gets added to the **Users on this controller** list.
3 Select the user, and then from the **User's groups**, click the group to which the user must be added.
4 Click **OK**.
5 Click **OK**. The new user gets added to the **Administrator** group.

Modifying an existing user

1 In the **UAS Administration Tool**, click the **Users** tab.

Continues on next page

1.6.1 Managing user rights and write access on an IRC5 controller
Continued

2 On the **User** tab, select the group from **User's groups** and then select the required user.
3 Click **Edit**. Enter the required changes and click **OK**.

1.6.2 Managing user rights and write access on an OmniCore controller

Overview

The OmniCore controller is delivered with a preset user named *Default User*. This user is assigned certain grants by default and it belongs to the role *Operator*. If a new user is created with specific grants, the *Default User* can be removed. An active *Default User* has *read only* rights to the controller data even if all grants are removed. Hence, to prevent any unauthorized access to OmniCore controller data, the *Default User* must be deleted.

The OmniCore controller is delivered with a default configured user, named *Admin*. All UAS grants are assigned against this user, such as, adding, removing and modifying users. The *Admin* user belongs to the role *Administrator* by default. You can deactivate the *Admin* user. Before deactivating default users, it is recommended to define at least one user with the grant *Manage UAS settings* so as to continue managing users and roles.

Adding a user to the Administrators group

The predefined user roles available in the *robot controller* are *Administrator* and *Operator*. For the Administrator role the controller grant *UAS_ADMINISTRATION* is enabled by default.

1. On the **Controller** tab, click **Add controller** and then click **Add Controller..** and then select the controller from the **Add Controller** dialog.
2. On the **Controller** tab, click **Request Write Access**.
3. Click **Authenticate** and then click **Login as Different User**. The **Login** dialog opens, enter the default credentials **User Name** and password as *Admin* and *robotics* respectively and click **Login**.
4. Click **Authenticate** and then click **Edit User Accounts**.

   The **Edit User Accounts** window opens.
5. On the **Users** tab, click **Add user**.
6. Enter suitable values in the fields as required and then under **Roles** select the **Administrator** check box. Click **Apply**.

   The new user gets added to the **Users on this controller** list.

Use the same steps to create users for various roles.

Creating a new user role

1. In the **Edit User Accounts**, click the **Roles** tab.
2. On the **Roles** tab, click **Add Role**.
3. Enter the required details and click **Apply**.

   The new role gets added to the selected user.

Modifying an existing user role

1. In the **Edit User Accounts**, click the **Roles** tab.
2. On the **Roles** tab, select the role and then click **Edit User**. Enter the required changes and click **Apply**.

Continues on next page

1.6.2 Managing user rights and write access on an OmniCore controller
Continued

Creating a new user

1 In the **Edit User Accounts**, click the **Users** tab, and then click **Add User**.
2 In the **User Name** and **Password** boxes, enter suitable values. Select the required roles and then click **Apply**.

The new user gets added to the **Users on this controller** list with the selected roles.

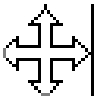
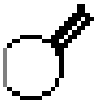
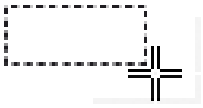
Note

Valid characters that can be used for creating passwords are *abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ_-1234567890,;.:!#%/\()=?*@${[]}£*, space is also a valid character.

1.7 Manage user interface using mouse

Navigating the graphics window using the mouse

The table below shows how to navigate the graphics window using the mouse:

To	Use the keyboard /mouse combination	Description
Select items  selectio	 left-cli	Just click the item to select. To select multiple items, press CTRL key while clicking new items.
Rotate the station  rotate	**CTRL + SHIFT +**  left-cli	Press CTRL + SHIFT + the left mouse button while dragging the mouse to rotate the station. With a 3-button mouse you can use the middle and right buttons, instead of the keyboard combination.
Pan the station  pan	**CTRL +**  left-cli	Press CTRL + the left mouse button while dragging the mouse to pan the station.
Zoom the station  zoom	**CTRL +**  right-cl	Press CTRL + the right mouse button while dragging the mouse to the left to zoom out. Dragging to the right zooms in. With a 3-button mouse you can also use the middle button, instead of the keyboard combination.
Zoom using window  window_z	**SHIFT +**  right-cl	Press SHIFT + the right mouse button while dragging the mouse across the area to zoom into.
Select using window  window_s	**SHIFT +**  left-cli	Press SHIFT + the left mouse button while dragging the mouse across the area to select all items that match the current selection level.

Using a 3D mouse

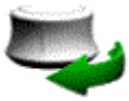
The 3Dconnexion 3D mouse has a pressure-sensitive controller cap designed to flex in all directions. The direction of movement are push, pull, twist, or tilt the cap to pan, zoom, and rotate the current view. A 3D mouse is used along with a regular mouse. Connect a 3D mouse to the RobotStudio environment to interact with the graphical environment.

It is possible to connect the programmable buttons of the 3D mouse to the commonly used RobotStudio commands by assigning the commands to custom

Continues on next page

keyboard shortcuts. The custom keyboard shortcuts are configured with the same user interface as the Quick Access Toolbar. After configuring the keyboard shortcuts in RobotStudio, connect the programmable buttons to the keyboard shortcuts in the 3D mouse applications control panel. For more information, refer the 3D mouse user manual .

The 3D mouse can move in six axes as mentioned in the following table.

Individual Axis	Axes	Description
 xx1500000297	Pan Right/Left	Moves the model right and left.
 xx1500000299	Zoom	Zoom the model in and out.
 xx1500000298	Pan Up/Down	Moves the model up and down.
 xx1500000301	Spin	Rotate around vertical axis.
 xx1500000300	Tilt	Tilts the model forwards and backwards.
 xx1500000302	Roll	Rolls the model sideways.

It is possible to freely rotate the view while using a SpaceMouse without the *up* direction getting locked to the positive z-axis. To disable this behavior, in **Advanced Settings**, uncheck the *sideways rotation* option.

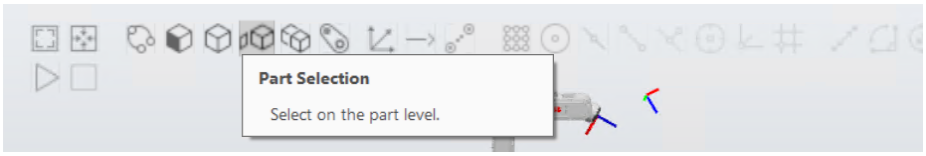
Selection levels and snap modes

Every item in a station can be moved to achieve the required layout using suitable selection levels. RobotStudio provides a set of selection levels which can be used to select part or specific part of an object. These selection levels are curve, surface, entity, part, mechanism, group, target/frame and path. The target/frame and path selection can be combined with other selection levels.

Continues on next page

Open a demo station while exploring the selection levels.

1 In the **Graphics** window, click the **Part Selection** icon. Hover the mouse over the icon to view the ToolTip which contains the name and purpose of the icon.



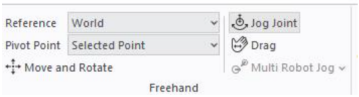
xx1900000134

2 In the **Graphics** window, click the **Snap Object** icon. This is a multi-snap mode, snapping to the closest center, edge or corner.

xx1900000137

Click a part in the **Graphics** view, the entire object will be highlighted. You can also see the pick point as a white star that has snapped to the closest center/edge/corner.

3 On the **Home** tab, in the **Freehand** group, click the **Jog Joint** button and then select any joint on the robot.



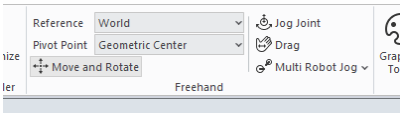
xx1900000136

By pressing the left mouse button on the joint in the **Graphics** window, the robot can be jogged in any direction.

Moving and Rotating Objects

Use the **Move and Rotate** tool to move and rotate objects and to jog the robot by moving its TCP.

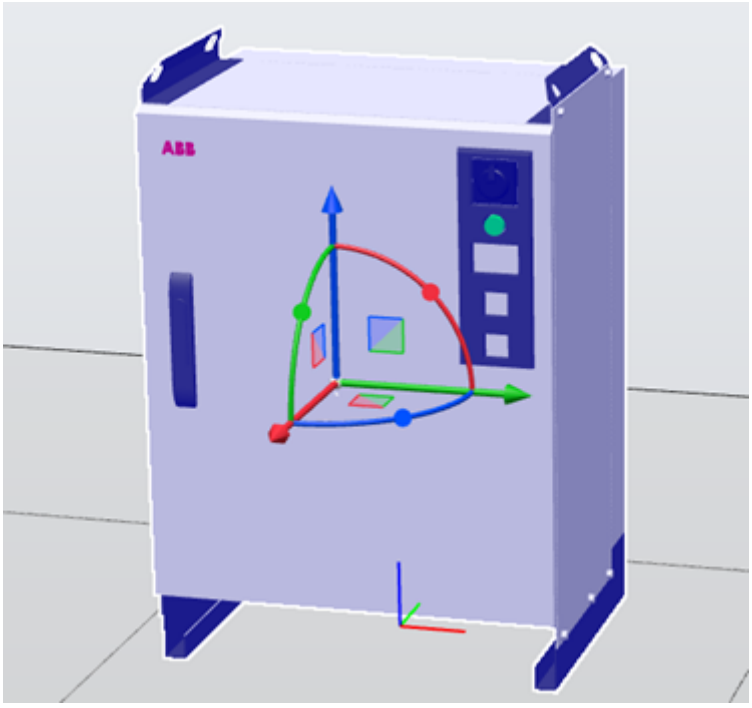
Select an object and then on the **Home** tab, in the **Freehand** group, click the **Move and Rotate** tool.



xx1900000135

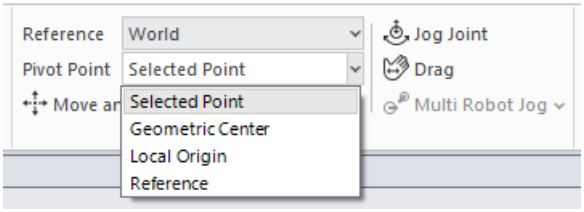
Continues on next page

The **Move and Rotate** tool with the movement handles will be displayed.



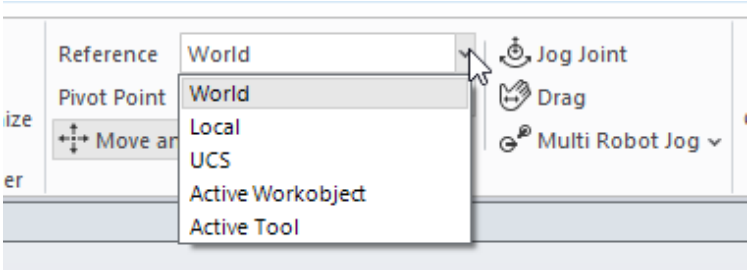
xx2300001765

The position of the tool can be altered by selecting a pivot point.



xx2300001766

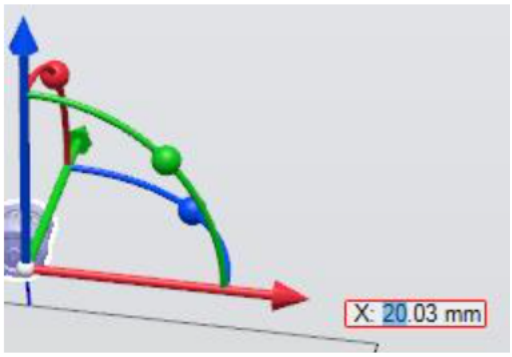
The orientation of the tool's axes can be defined by the selected reference.



xx2300001767

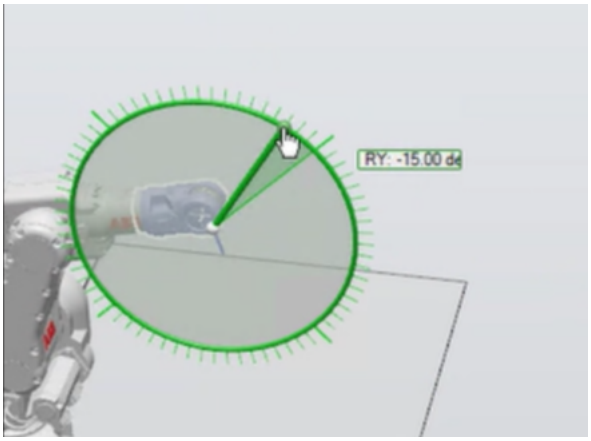
Continues on next page

When moving or rotating an object, the offset from the starting position appears aside the tool, where, precise values for the movement or rotation can be set.



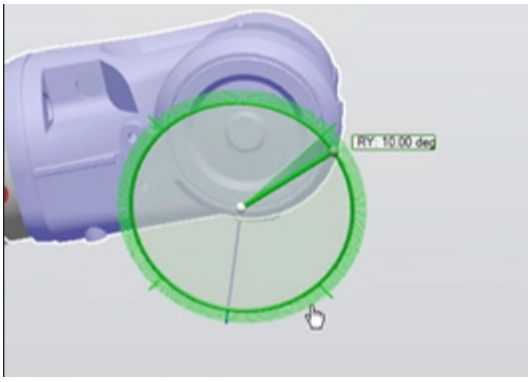
xx2300001781

Use the drop-down menu to show position relative to the selected reference.



xx2300001782

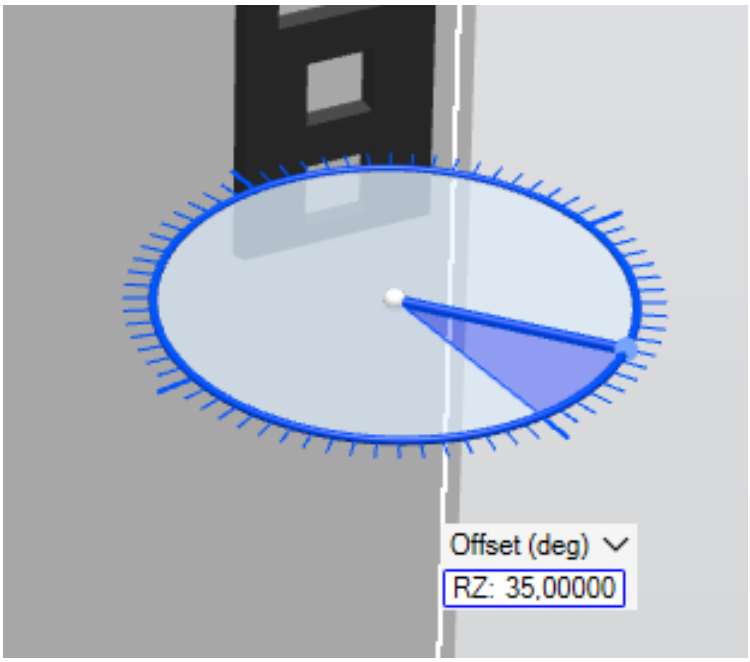
While dragging an object, press and hold the CTRL button to enable movement in increments. Press CTRL + SHIFT to enable fine increments. The increment size depends on how close you have zoomed in on the object.



xx2300001783

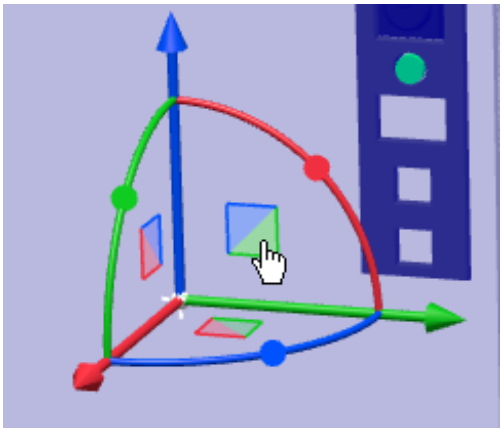
Continues on next page

Rotation increments are 5 degrees and fine increments are 1 degree.



xx2300001815

As displayed in the following image, click and drag the icon to move the object along the required plane.

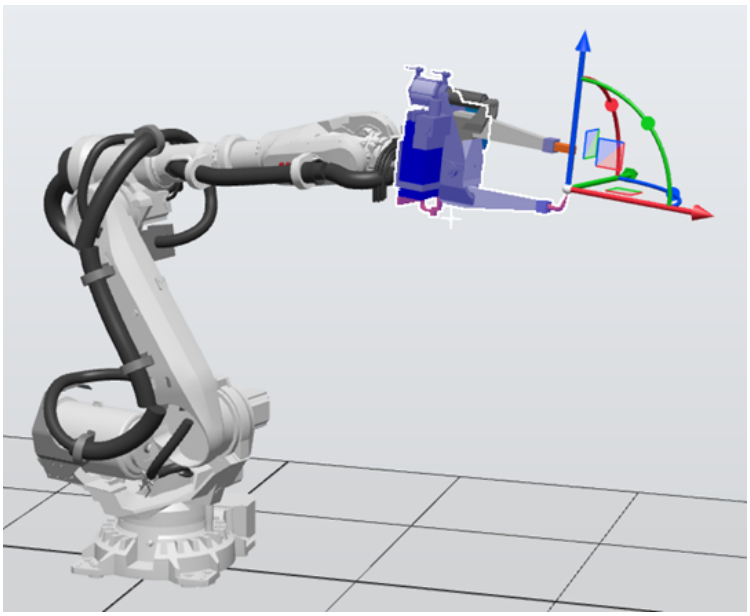


xx2300001878

Continues on next page

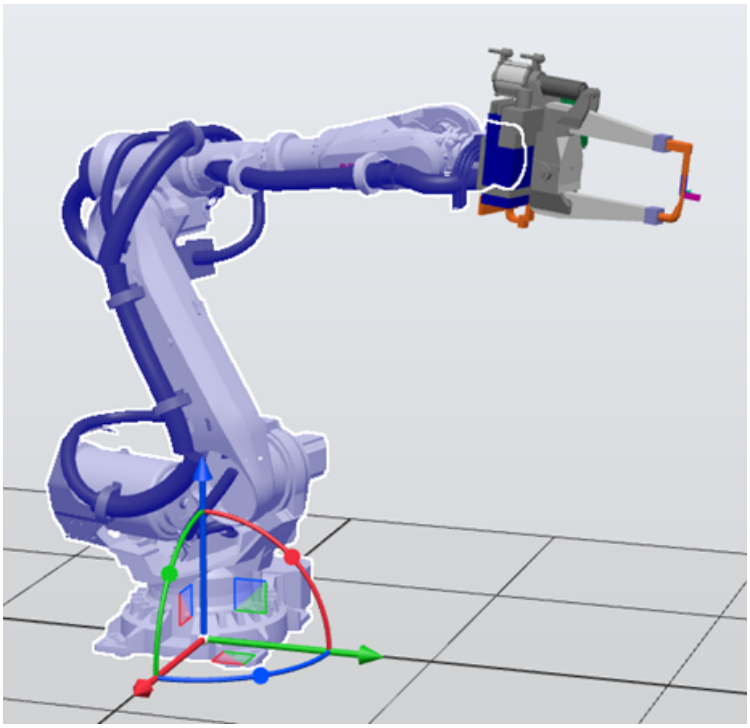
Jogging a robot

Combine selection level **Part** with the **Move and Rotate** tool and then click on the robot base to move the robot, click anywhere else on the robot to move the robot TCP.



xx2300001879

With the **Mechanism** level selected, click anywhere on the robot to move the robot.



xx2300001880

Continues on next page

Selecting an item in the graphics window

To select items in the **Graphics** window, follow these steps:

1. At the top of the **Graphics** window, click the desired selection level icon.
2. Optionally, click the desired snap mode icon for the part of the item you wish to select.
3. In the **Graphics** window, click the item. The selected item will be highlighted.

Deep rectangle selection

The Deep rectangle selection (default selection mode) is enabled if you press and hold the `SHIFT` key and draw a rectangle in the 3D view using the mouse. This mode selects objects covered by the selection rectangle regardless of its visibility.

Deep rectangle selection

To enable the deep rectangle selection (default selection mode), press and hold the `SHIFT` key and drag the mouse diagonally over the objects to select. This mode selects objects covered by the selection rectangle regardless of its visibility. In this selection mode, you can select multiple items in the **Graphics** window.

Shallow rectangle selection

To enable shallow rectangle selection, press the `SHIFT + S` keys and draw a rectangle in the 3D view using the mouse. In this mode, it is possible to select the currently visible object.

Selecting an item in the browsers

To select items in a browser, do the following:

1. Click the item. The selected item will be highlighted in the browser.

Multiple selection of items in the browsers

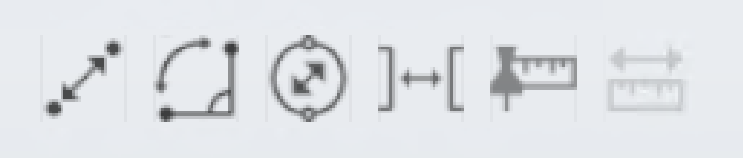
To select multiple items in a browser, follow these steps:

1. Make sure that all the items to be selected are of the same type and located in the same branch of the hierarchical structure; otherwise, the items will not be operable.
2. Do one of the following:
   - To select adjacent items: In the browser, hold down the **SHIFT** key and click the first and then the last item. The list of items will be highlighted.
   - To select separate items: In the browser, hold down the **CTRL** key and click the items you want to select. The items will be highlighted.

Continues on next page

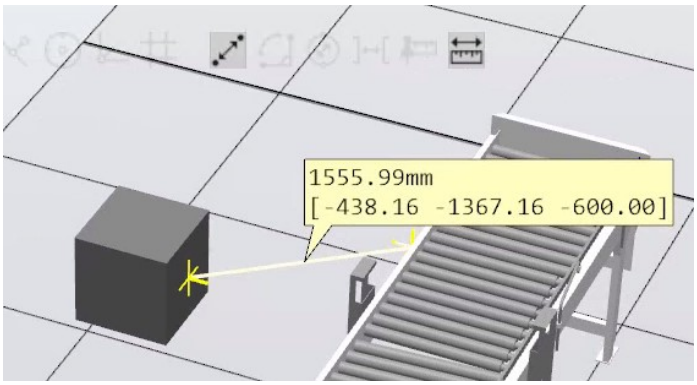
Measurement tools

RobotStudio provides the following measurement tools for gauging the linear and angular distances between various objects in the Graphics window. Hover the mouse over these icons to see the name and purpose of these tools.



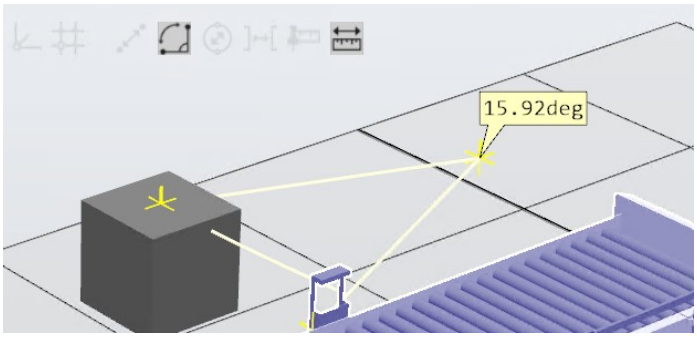
xx2000002621

- In the **Graphics** window, click the **Point to Point** icon to enable this tool. With the icon selected, snap two objects in the Graphics window to view the distance between these objects in millimeters.



xx2000002622

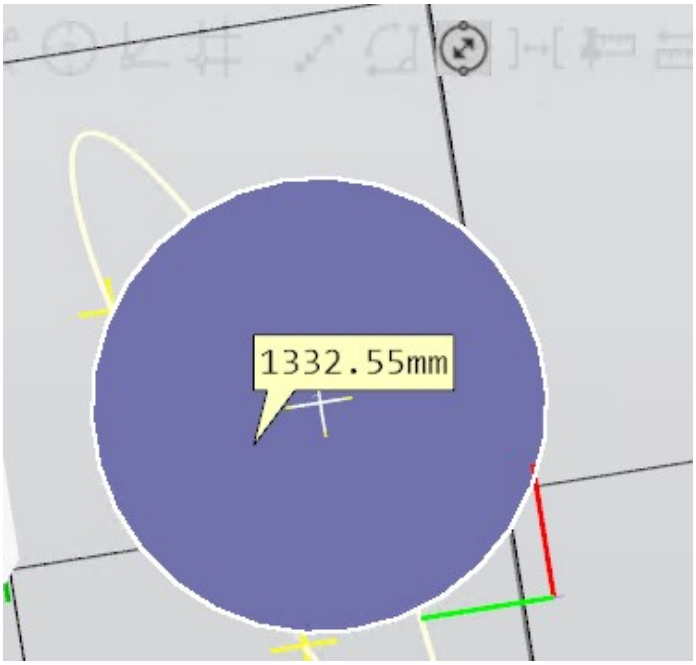
- In the **Graphics** window, click the **Angle** icon to enable this tool. With the icon selected, snap three points to create two lines to measure the angle between them in degrees.



xx2000002623

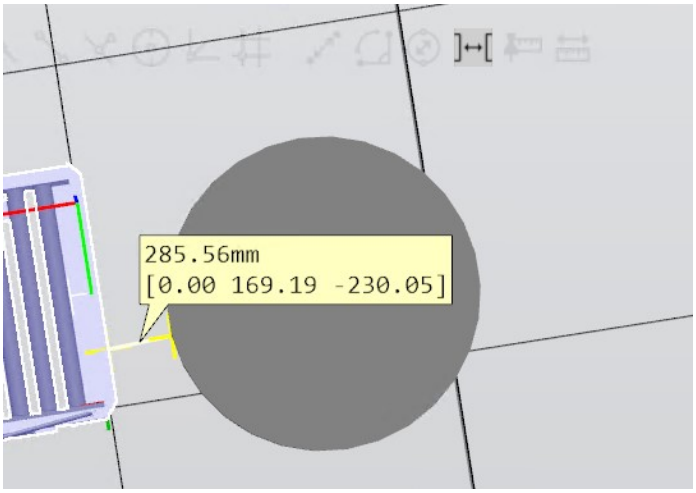
Continues on next page

- In the **Graphics** window, click the **Diameter** icon to enable this tool. With the icon selected, snap three points on the circumference of the surface to find the diameter in millimeters.



xx2000002624

- In the **Graphics** window, click the **Minimum Distance** icon to enable this tool. With the icon selected, click on two objects to find the minimum distance between them in millimeters.



xx2000002660

- The **Keep measurements** and the **Track moving objects** tools are used along with other tools, when **Keep measurements** icon is selected, RobotStudio retains the mark up from the previous measurement.

  When **Track moving objects** icon is selected, the mark up gets updated dynamically to reflect the movement of objects in the **Graphics** window.

1.8 Libraries, geometries and CAD files

Overview

Models of *work pieces* and equipment are added to RobotStudio for simulating the *station* and for further programming. RobotStudio provides *ABB library* which contains models of ABB robots and related equipment as *library files* or *geometries*. *User libraries* are imported as geometries to RobotStudio. These files can be created in RobotStudio.

Difference between geometries and libraries

External files are imported to a *station* as *user library* or *geometries*. Geometries are CAD files, when imported, these files are copied to a station, hence, the size of the station file increases.

When *user library* files are imported, a link gets created from the station file to the corresponding library file, hence the size of the station file remains the same.

Components of a geometry

A geometry consisting of several parts is called the *Component Group* in RobotStudio. 3D *Geometries* of a station are displayed in the **Layout** browser, under **Components** as *parts*. Parts contain bodies. Bodies contain faces (surfaces) or *curves*.

Importing and converting CAD files

Use the import function to import *geometries* from single CAD files. On the **Home** tab, click **Import Geometry** and select the CAD file to import.

If you want to re-use the CAD file from other stations or for future use, you can save it as a library file. Library files can be imported using the **Import Library** function on the **Home** tab.

Exporting geometry

Use the **Export Geometry** function to export component groups, parts, stations or mechanism links to CAD file. Right-click an open group, station, part or mechanism link and then select **Export Geometry** to access this command. The supported export formats vary for each entity, which can be selected while exporting.

Supported 3D formats

The native 3D CAD format of RobotStudio is SAT, it also supports other formats for which you need an option. The CAD support in RobotStudio is provided by the software component ACIS. The following table lists the supported formats and the corresponding options.

Format	File extensions	Option required	Read	Write
3DStudio	.3ds	-		
ACIS	.sat, .sab, .asat, .asab	-	X	X
CATIA V4	.model, .exp, .session	CATIA	X	

Continues on next page

Format	File extensions	Option required	Read	Write
CATIA V5/V6	.CATPart, .CAT-Product, .CGR, .3DXML	CATIA	X	X
COLLADA 1.4.1	.dae	-		
DirectX writes 2.0	.x	-		
DXF/DWG	.dxf, .dwg	AutoCAD	X	
FBX	.fbx	-		X
IGES	.igs, .iges	IGES	X	X
Inventor	.ipt, .iam	Inventor	X	
JT	.jt	JT	X	
LDraw	.ldr, .ldraw, .mpd	-	X	
NX	.prt	NX	X	
OBJ	.obj	-		
Parasolid	.x_t, .xmt_txt, .x_b, .xmt_bin	Parasolid	X	
Pro/E / Creo	.prt, .prt.*, .asm, .asm.*	Creo	X	
Solid Edge	.par, .asm, .psm	SolidEdge	X	
SolidWorks	.sldprt, .sldasm	SolidWorks	X	
STEP	stp, step, p21	STEP	X	
STL, ASCII STL supported (binary STL not supported)	stl	-		
VDA-FS	vda, vdafs	VDA-FS	X	
VRML, VRML2 (VRML1 not supported)	wrl, vrml, vrml2	-	X	
glTF	.glb	-		X

Note

Refer to the RobotStudio Product Specification for the version information of various supported CAD formats.

To import these files into RobotStudio, use the **Import Geometry** function.

Mathematical versus graphical geometries

A *geometry* in a CAD file always has an underlying mathematical representation. Its graphical representation, displayed in the *Graphics* window, is generated from the mathematical representation when the geometry is imported to RobotStudio, after which the geometry is referred to as *part*.

For this type of geometry, it is possible to set the detail level of the graphical representation. Setting the detail level reduces the *file size* and *rendering time* for

Continues on next page

large models and improves the visual display for small models. The detail level only affects the visual display; *path*s and *curves* created from the model will be accurate both with coarse and fine settings.

A *part* can also be imported from a file that defines its graphical representation; in this case, there is no underlying mathematical representation. Some of the functions in RobotStudio, such as *snap mode* and *creation of curves from the geometry*, will not work with this type of part. To customize the detail level settings, on the **File** tab, click **Options** and then select **Options:Graphics:Geometry**.

Options:Graphics:Geometry

Detail Level	Specify the level of detail required when importing geometries. Select **Fine**, **Medium** or **Coarse** as required.

1.9 How to install RobotWare and Add-Ins

Installing RobotWare

Use Modify Installation to create and modify systems with *RobotWare* versions 7. Use Installation Manager 6 to create and modify systems with RobotWare versions 6.0. Use System Builder to create and modify systems based on earlier versions of RobotWare.

Installing Add-Ins

RobotStudio *add-ins* are available in the **Gallery** window. The recommended method is to install software is from RobotStudio.

Installing Add-Ins from Gallery in RobotStudio

1. Start RobotStudio and open the **Add-Ins** tab. The **Gallery** window is displayed.
2. In the **Gallery** window, use the **Search** function or **Common tags** to filter the available add-ins.
3. Select the add-in to be installed. Additional information is displayed in the window to the right.
4. Select **Version** and click **Add**. The add-in gets installed.

> **Note**
>
> The default is the latest version.

Installing add-ins manually

Add-ins and additional content can be installed manually from the hard disk.

1. Locate the *.rspak* file in the user directory.
2. Start RobotStudio and open the **Add-Ins** tab. Select **Install Package** from the menu bar.
3. Navigate to the required *.rspak* file in the user directory and click **Open**. The add-in gets installed.

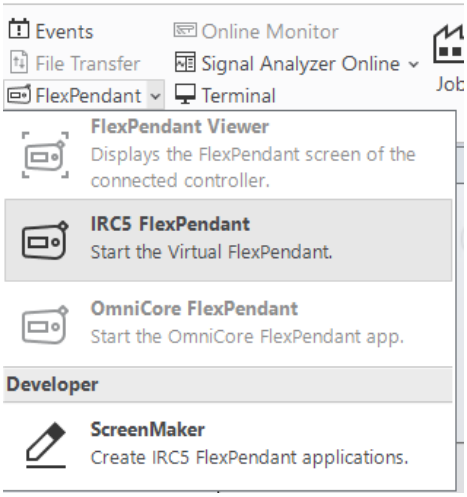
Installing the Virtual FlexPendant

RobotWare 7.0, Microsoft Windows 10 and Microsoft App Installer for Windows 10 must be available in the host PC for carrying out the following procedure.

1. In the **Add-ins** tab, in the **Add-ins** browser, expand the RobotWare 7.0 node. Right-click the **FlexPendant Apps** node and then click **Install** from the context menu.

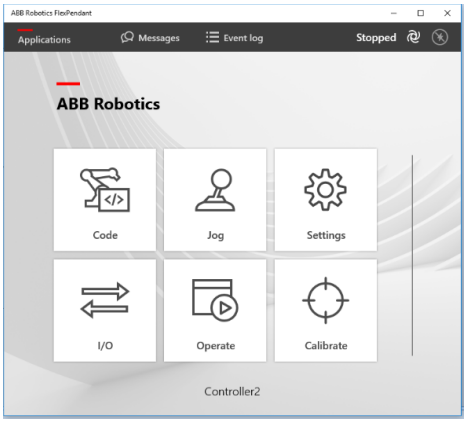
Continues on next page

2 On successful installation, the following option gets added to the tool bar.



xx1900001281

3 Select the FlexPendant option to open Virtual FlexPendant.



xx1900001282

This page is intentionally left blank

2 Building Stations

2.1 Understanding stations and projects

Station

Station document is a *.rsstn* file that contains data of the robot cell. It stores information about various *station components*, *station logic* including *smart components* that controls external components, 3D graphics, CAD data, and data on the graphical part of the robot program. Station document is linked to the *virtual controller* that runs the robot in a *station*. virtual controller data is external to the station document.

Project

Projects add structure to the station data. It contains folders for structuring station data so as to keep related data together. Project contains the station, that is, the station document is part of the project structure. By default, RobotStudio provides project folders. In the default project folder structure, files of similar type are stored in folders. The project structure is the recommended way for storing station files. In the project structure there are dedicated folders for, Components, Controller Data, Station, User Files, Virtual Controllers.

For a new installation of RobotStudio, the project folder contains the following subfolders.

- Components: This folder contains user library components that belong to the project.
- Controller Data: This folder contains auto-generated backups of virtual controllers, updated when the project is saved.
- Station: This folder contains graphics, geometry and other components referred to by the station.
- User Files: This folder contains files or directories that a user want to add to the project. Some examples are CAD files that belong to the project, RAPID programs that do not belong to a virtual controller or user generated backups.
- Virtual Controllers: This folder contains virtual controller generated files and folders.
- Project.rsproj : The project file *<project name>.rsproj* that makes the folder as a project. Use this file to open the project.

Continues on next page

Pack and Go

The *Pack and Go* file is a single file that packages the station data along with the related *virtual controllers* for archiving and for sharing station data with other users.

Note

A *station* file that includes references to the virtual controllers cannot be moved to a new location on the local disk. To move a station file to a different location, create a *Pack & Go* file of the station from the original location and then move this file to the new location and then unpack the file.

2.2 Preparing the computer for hosting RobotStudio

Overview

Laptops that are configured with switchable graphics adapters can engage higher performance graphics adapter for 3D applications, and the energy-efficient integrated graphics adapter for less demanding tasks. For a laptop with switchable graphics adapters, ensure that RobotStudio uses the adapter that handles the high performance discrete graphics. To get the best user experience, it is recommended to install the latest display drivers available.

Before installing display drivers, check the Windows® Device Manager and verify that both graphics adapters appear in the list of hardware devices and that they are enabled. When a driver is not installed for the device, either of the graphics adapters may appear under *Other Devices* as a *Generic Video Controller*. Some vendors of the graphics adapters provide configuration software for creating application specific profiles. It is recommended to create a RobotStudio specific application profile that uses the high-performance graphics adapter.

A second option is to modify the application configuration file, *RobotStudio.exe.config* such that RobotStudio uses a specific graphics adapter. This file is located in the installation folder of RobotStudio, *C:\Program Files (x86)\ABB\RobotStudio x.x*. It contains a certain line that controls the type of graphics adapter to use. Uncomment the following line to set a specific graphics adapter for 3D graphics, remove the initial "<!--" "-->" in the beginning and at the end of the line.

```
<!-- <add key="GraphicsDeviceType" value="Discrete"/> -->
```

Valid values are *Discrete*, *Integrated*, *Warp*, or *Default*.

- Discrete : configures the use of the high performance graphics adapter.
- Integrated : configures the energy efficient graphics adapter to be used.
- Default : allows the laptop to choose, but enables logging of graphics adapter information to the *Output* window.
- Warp(Windows Advanced Rasterization Platform) : controls software rendering, for example, use CPU instead of GPU.

Continues on next page

A third option which may or may not be applicable to the laptop, is to configure the graphics adapter to use in the BIOS, refer the user documentation of the laptop for details.

Note

Ensure to meet the following requirements while selecting the user name for a new virtual controller.

- Virtual Controllers with RobotWare lower than 7.3 do not support non-Latin1 characters in the Windows filepaths.
- If such characters are used in the Windows user name, for example, *å, ä, ö or ü*, which is part of the path to the virtual controller, the virtual controller will not start.
- Available workarounds are to change the Windows user name to remove such characters, or save the virtual controller to a path which does not contain these characters.

Document folders

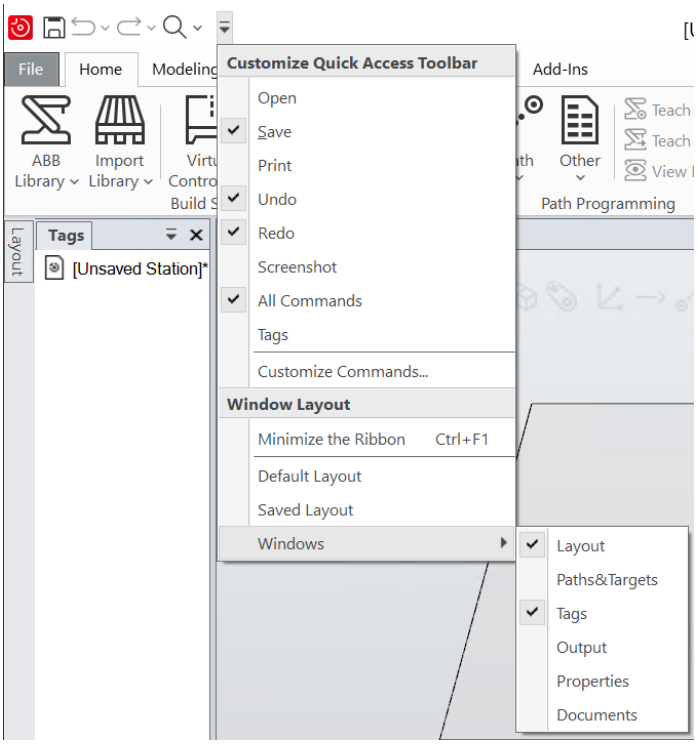
ABB Library contains geometries of robots and other equipments. Creating a user library and adding user geometry to those galleries allows you to directly access those *libraries* and *geometries* from the **Import Library** and **Import Geometry** galleries in RobotStudio. You can create folders with such libraries and then add references to make them appear in the user interface galleries.

Use the following steps to create a gallery for frequently used documents.

1. Start **RobotStudio**.
2. On the **File** tab, click **New** and then double-click **Empty Station** to open a new empty station.

Continues on next page

The **Documents** window is part of the default layout. If you are unable to find the **Documents** window, on the **Quick Access** menu, click **Windows** and then click **Documents**.



xx1800003487

3 In the **Documents** window, click **Locations**. The **Document Locations** dialog box opens.

4 In the **Document Locations** dialog box, click **Add Location** and then **File System**. The **File System** dialogue box opens.

5 In the **File System** dialog box, enter the required details and click **OK**. Library files from the selected folder will be available in the **Import Library** option.

Repeat the same step and add library for the **Import Geometry** option.

After adding these locations, it is possible to access the selected files directly from RobotStudio. Any file saved to those locations they will be automatically added to the galleries.

Use the **Search** function in the **Documents** window to search for a document by its name. The result will appear on the **Documents** window. Double-click the found item(s) to import it to the station. Alternatively, use the **Browse** function to browse to all available locations created in document locations. These functions provide easy and quick access to your documents.

Setting the folder structure and autosave options

It is recommended to set the autosave option to enable RobotStudio to save the current changes or progress in the program. This reduces the risk of data loss during any unexpected interrupts such as crash, freeze or user error.

Continues on next page

To access the Autosave option, on the **File** tab, click **Options** and then select **Options:General:Autosave**.

Autosave

Enable autosave of RAPID	This check-box is selected by default. *RAPID* programs are saved automatically in every 30 seconds.
Enable autosave of station	Unsaved stations are saved automatically at the interval specified in the **minute interval** box.
Enable automatic backup of station files	Takes multiple backup of station files as specified in the **Number of backups** list and saves it in a sub-folder of the corresponding Stations folder (StationBackups). Requires a *Project*.
Enable automatic backup of controllers in project	Select this option to backup the *virtual controllers* of a project when saving the station. The backups are stored in the *Backups* folder of the corresponding project.

Assign default locations for storing the user data, projects and so on, by selecting **Options:General:Files & Folders**.

Files & Folders

User Documents location	Shows the default path to the project folder.
Local projects location	Shows the default path to the project folder.
...	To browse to the project folder, click the browse button.
Automatically create document subfolders	Select this check box to enable the creation of individual subfolders for document types.
minute interval	Specify the interval between the savings when using *Autosave* in this box.
Document Locations	Launches the **Document Locations** dialog box.
Clear Recent Stations and Controllers	Clears the list of recently accessed stations and controllers.
Additional *distribution package* location	RobotWare 6 and related RobotWare add-ins media pools are distributed as distribution packages. For RobotStudio to find them, they need to be located in a specific folder. If the folder is not specified, the default location is used. On a Windows installation with English language, the default folder is *C:\User\<user name>\AppData\Local\ABB\DistributionPackages*. The location can be customized by entering a search path here.
Download packages to this location	Select this check box to download *distribution packages* to the user defined location instead of the default folder.
Unpacked *RobotWare* Location	Shows the default path to the unpacked RobotWare folder.
Media Pool for RobotWare 5.x	This is where RobotStudio searches for RobotWare 5.xx media pools.

2.3 Creating the station

Overview

The following steps define the workflow for building a new *station*.

1. Create an empty station.
2. Import a robot model.
3. Add *positioners* and track motion.
4. Create a *virtual controller*.
5. Import a tool and attach it to the robot.
6. Create a *workobject*.
7. Define paths and targets to create a robot program graphically.
8. *Synchronize* to RAPID to create the RAPID program.

Creating a project with an optional virtual controller

1. Click the **File** tab. The RobotStudio Backstage view appears, click **New**.
2. Under **Project**, click **Project**.
3. Enter the name of the *project* in the **Name** box and then browse and select the destination folder in the **Location** box. The default location of the project is `C:\User\<user name>\Documents\RobotStudio\Projects`.

   The included virtual controller the name of the project.

   > **Note**
   >
   > The project folder helps in keeping a well-structured folder system that is easy to navigate through.

4. Click **Create**.

   The new project gets created. RobotStudio saves this project by default.

2.4 Importing robots and related components

Importing a robot model

1. In the **Home** tab, click **ABB Library** and select a robot model. Models that are not installed are indicated by a download icon and will be downloaded automatically.
2. Select the desired robot variant in the dialog and click **OK**.

   The selected robot model is displayed in the graphic window.

A robot which is not connected to a controller cannot be programmed, hence configure a *virtual controller* for the robot.

RobotStudio Models

Before working offline, it is recommended to download the required simulation models and Virtual Controller templates of robot models, positioners, tracks and tools.

1. On the **Add-Ins** tab, in the **Packages** group, click **Gallery**.
2. Under **Gallery**, click the **RobotStudio Models** tab to view all available robot models, positioners, tracks and tools.
3. Select the required packages and click the **Add Selected** button.

   Use the **Add All** button to download all available robot models.

All downloaded models will be available in ABB Library and can be selected when creating a new Virtual Controller. Models that are not downloaded are indicated by a download icon.

Importing and attaching a tool

A *tool* is a RobotStudio object that operates on the work piece, for example, an arc weld gun or a gripper. For achieving correct motions in robot programs, parameters of the tool must be specified in the *tool data*. The most essential part of the *tool data* is the *TCP*, which is the position of the tool center point relative to the wrist of the robot (which is the same as the default tool, tool0).

1. To import a tool, in the **Home** tab, click **Import Library** and then click **Equipment** and select a tool.

   The tool gets imported to the station and placed at the origin of the *world coordinate system*, thus hidden inside the robot. When imported, the tool gets added to the **Layout** browser, but will not be attached to the robot. Tool must be attached to a robot to synchronize its movements with the robot.
2. To attach a tool to the robot, inside the **Layout** browser, right-click the tool and then click **Attach to** and select the robot model.

   Tool can be attached by dragging and dropping it on the robot in the **Layout** browser.
3. In the **Update position** dialog box, click **Yes**.

Continues on next page

Importing a library

Use this procedure for importing library files to a *station*:

1. On the **Home** tab, click **Import Library** and select various component libraries.
2. Click **User Library** to select the user defined *libraries*.
3. Click **Equipment** to import predefined ABB *libraries*.
4. Click **Project Library** to select the predefined *Projects*.
5. Click **Locations** to open the Document Locations window.
6. Click **Browse for Library** to select the saved library files.

2.5 Creating a virtual controller

A robot which is not connected to a controller cannot be programed, hence configure a *virtual controller* for the robot.

1. In the **Home** tab, Click **Virtual Controller**.
2. Click **From Layout** to bring up the first page of the wizard.
3. In the **Name** box, enter the name of the virtual controller. The location of the *virtual controller* will be displayed in the **Location** box.
4. Click **Next**.
5. In the **Mechanisms** box, select the *mechanisms* that you want to include in the *virtual controller*. Add mechanisms such as tracks or *positioners* to your station before creating a virtual controller.

   Depending on the mechanisms that are added, an appropriate RobotWare version is selected automatically.

   For OmniCore controllers, the controller variant can be selected among the variants supported by the robot. The controller variant can affect motion performance of the robot.
6. Click **Next**.

   The wizard now prompts a mapping of the *mechanisms* to a specific motion *task*, in accordance with the following rules:
   - Only one *TCP* robot is allowed per task.
   - Up to six motion tasks may be added, but only four *TCP* robots can be used, and they must be assigned to the first four tasks.
   - The number of tasks may not exceed the number of mechanisms.
   - If the *system* contains one *TCP* robot and one external axis, they will be assigned to the same task. It is, however, possible to add a new task and assign the external axis to it.
   - If the system contains more than one *TCP* robot, any external axes will be assigned to a separate task. It is, however, possible to move them to other tasks.
   - The number of *external axes* in a task is limited by the number of available drive modules in the cabinet (one for large robots, two for medium, three for small).

     If only one *mechanism* was selected in the previous page, this page will not be shown.

     Tasks can be added and removed using the respective buttons; mechanisms can be moved up or down using the respective arrows. To map the mechanisms to tasks, follow this step:
7. Optionally, make any edits in the mapping, and then click **Next**.

   The **Controller Options** page appears.

Continues on next page

8 On the **Controller Options** page, you have the option to align Task Frame(s) with the corresponding *Base Frame*(s).
   - For single robot system, select the checkbox to align task frame with base frame.
   - For *MultiMove* Independent system, select the check box to align task frame with base frame for each robot.
   - For MultiMove Coordinated system, select the robot from the drop down list and select the check box to align task frame with base frame for the selected robot.

9 Verify the summary and then click **Finish**.

   If the virtual controller contains more than one robot, the number of tasks and the base frame positions of the mechanism should be verified in the **Motion Configuration** window.

Note

To create a system from layout, all *mechanisms* such as robots, *track motions* and *positioners*, must be saved as libraries.

2.6 Synchronizing to virtual controller to create a RAPID program

Movements of the robot can be programmed using *RAPID*. A robot which is not part of a task in a controller cannot be programed, hence configure a *virtual controller* for the robot before *synchronizing* to RAPID.

1. With the station open, in the **RAPID** tab, click **Synchronize**.
2. From the options, click **Synchronize to RAPID** to match objects in the station to the RAPID code.

   In the **Controller** browser, expand the tree view to find the RAPID node. Click RAPID node to view the RAPID files.

2.7 Configuring station with robot and positioner

You can configure a station with robot and *positioner* using the **Virtual Controller** button.
To configure a station with a robot and positioner:

1. Click **From Layout** to bring up the first page of the wizard.
2. In the **Name** box, enter the name of the virtual controller. The location of the virtual controller will be displayed in the **Location** box.
3. In the **RobotWare** list, select the version of *RobotWare* you want to use.
4. Click **Next**.
5. In the **Mechanisms** box, select the Positioner that you want to include in the *virtual controller*.
6. Click **Next**.

   Tasks can be added and removed using the respective buttons; positioner can be moved up or down using the respective arrows. To map the positioner to tasks, follow this step:
7. Optionally, make any edits in the mapping, and then click **Next**.

   The **Controller Options** page appears.
8. In the **Controller Options** page, you have the option to align Task Frame(s) with the corresponding *Base Frame*(s).
   - For single virtual controller, select the checkbox to align task frame with base frame.
   - For *MultiMove* Independent system, select the check box to align task frame with base frame for each robot.
   - For MultiMove Coordinated system, select the robot from the drop down list and select the check box to align task frame with base frame for the selected robot.
9. Verify the summary and then click **Finish**.

   If the virtual controller contains more than one robot, the number of tasks and the baseframe positions of the positioner should be verified in the **Motion Configuration** window.

Note

To create a system from layout, all *mechanisms* such as robots, *track motions* and *positioners*, must be saved as libraries.

2.8 Configuring station with robot and Track motion

You can configure a *station* with robot and *track motion* using the **Virtual Controller** button.
To configure a station with a robot and track motion:

1. Click **From Layout** to bring up the first page of the wizard.
2. In the **Name** box, enter the name of the system. The location of the system will be displayed in the Location box.
3. In the RobotWare list, select the version of *RobotWare* you want to use.
4. Click **Next**.
5. In the Mechanisms box, select the track motion that you want to include in the *virtual controller*.
6. Click **Next**.

   Tasks can be added and removed using the respective buttons; track motion can be moved up or down using the respective arrows. To map the track motion to tasks, follow this step:
7. Optionally, make any edits in the mapping, and then click **Next**.

   The System Option page appears.
8. On the **Controller Options** page, you have the option to align Task Frame(s) with the corresponding *Base Frame*(s).
   - For single virtual controller, select the check-box to align task frame with base frame
   - For *MultiMove* Independent system, select the check-box to align task frame with base frame for each robot.
   - For MultiMove Coordinated system, select the robot from the drop down list and select the check-box to align task frame with base frame for the selected robot.
9. Verify the summary and then click **Finish**.

   If the virtual controller contains more than one robot, the number of tasks and the *base frame* positions of the *track motion* should be verified in the **Motion Configuration** window.

Note

To create a system from layout, all *mechanisms* such as robots, *track motions* and *positioners*, must be saved as libraries.

2.9 Configuring Conveyor tracking

Overview

Configuring a conveyor tracking involves three steps; creating a conveyor, creating a virtual controller with the option **606-1 Conveyor Tracking** and then creating a connection between *virtual controller* and the conveyor.

Creating a Conveyor

1. In the **Modelling** tab, in the **Mechanism** group, click **Create Conveyor**. The **Create Conveyor** browser opens.
2. From the **Conveyor Geometry** list, select a geometry. To add a geometry to the station, click **Import Geometry** and then select a *geometry*.
3. In the **Reference Frame**, enter the base frame values (*Position* and *Orientation*) relative to the World/*local origin* of the selected graphic component.

   The Reference Frame defines the position where objects appear on the conveyor.
4. From the **Type** list, select the type of conveyor.

   Note

   Only linear conveyors are supported.
5. In the **Conveyor length** box, enter the length of the conveyor. The **Create Mechanism** gets enabled.
6. Click **Create** to create the conveyor.
7. Build a new virtual controller using the **Virtual Controller** button.

   In the **Controller Options** pane, scroll down to **Motion Coordination**, under **Conveyor Control Options** select **606-1 Conveyor Tracking** and then select one or more of the following options as required:
   - **1552-1 Tracking Unit Interface**
   - **709-1 DeviceNet Master/Slave**
   - **Conveyor Tracking on PIB**
8. Add the new *virtual controller* to the *station*.

Creating a new Virtual Controller

1. In the **Home** tab, Click **Virtual Controller**.
2. Click **From Layout** to bring up the first page of the wizard.
3. In the **Name** box, enter the name of the virtual controller. The location of the *virtual controller* will be displayed in the **Location** box.
4. In the RobotWare list, select the version of *RobotWare* you want to use.
5. Click **Next**.
6. In the **Mechanisms** box, select the *mechanisms* that you want to include in the *virtual controller*.

Continues on next page

7 Click **Next**.

The wizard now prompts a mapping of the mechanisms to a specific motion *task*, in accordance with the following rules:

- Only one *TCP* robot is allowed per task.
- Up to six motion tasks may be added, but only four *TCP* robots can be used, and they must be assigned to the first four tasks.
- The number of tasks may not exceed the number of mechanisms.
- If the virtual controller contains one TCP robot and one external axis, they will be assigned to the same task. It is, however, possible to add a new task and assign the *external axis* to it.
- If the *virtual controller* contains more than one TCP robot, any external axes will be assigned to a separate task. It is, however, possible to move them to other tasks.
- The number of *external axes* in a task is limited by the number of available *drive modules* in the cabinet (one for large robots, two for medium, three for small).

  If only one *mechanism* was selected in the previous page, this page will not be shown.

  Tasks can be added and removed using the respective buttons; mechanisms can be moved up or down using the respective arrows. To map the mechanisms to tasks, follow this step:

8 Optionally, make any edits in the mapping, and then click **Next**.

The *System Option* page opens. Select the **606-1 Conveyor Tracking** option and then select one or more of the following options as required:

- **1552-1 Tracking Unit Interface**
- **709-1 DeviceNet Master/Slave**
- **Conveyor Tracking on PIB**

9 On the *System Option* page, you have the option to align *Task Frame*(s) with the corresponding *Base Frame*(s).

10 Verify the summary and then click **Finish**.

If the *virtual controller* contains more than one robot, the number of tasks and the *base frame* positions of the mechanism should be verified in the **Motion Configuration** window.

Create connection between virtual controller and conveyor

1 In the **Modeling** tab, click **Create Connection**.

2 In the **Create Connection** tab, select the conveyor library in the **Conveyor** list and then select the *mechanical unit*.

3 Set a suitable offset (base frame of the conveyor). This offset defines the location of the *base frame* of the conveyor mechanical unit in relation to the conveyor reference frame.

4 Under the **Connection Window**, set appropriate values for **Minimum** and **Maximum distances**, and **Start Window Width**.

Continues on next page

5 Under **Base Frames**, select the following options:
   - Select **Station Values** to update the *virtual controller* with the *station* layout values.
   - Select **Align Task Frame** to move the *task frame* to the connection (to align it with the base frame). The conveyor base frame will become zero.
   - Select **Use Controller Values** to update the station layout and the task frame to match the *virtual controller* values.

6 Click **Create**.

   Delete the existing connection and create a new connection if you select another *workobject* to be used for conveyor tracking.

2.10 Configuring external axis

Create a system with external axes

1 Import the required robots, *positioners*, and track libraries into the RobotStudio *station*.

If a robot and track are selected, attach the robot to the track.

2 Create a *virtual controller* from layout.

> **Note**
>
> To create a virtual controller with IRBT4004, IRBT6004, or IRBT7004, the TrackMotion mediapool Type A must be installed.

Supported external axes configurations for RobotWare 5

The following table shows a combination of different external axes configurations:

Combination	Positioner type							
	A	B	C	D	K	L	2xL	R
One IRB (Positioner in same task)	Y	Y	Y	Y	Y	Y	Y	Y
One IRB (Positioner in separate task)	Y	Y	Y	Y	Y	Y	Y	Y
Two IRB (Positioner in separate task)	Y	Y	Y	Y	Y	Y	N	Y
One IRB on Track Motion (Positioner in same task)	Y	N	N	N	YX	Y	Y	N
One IRB on Track Motion (Positioner in separate task)	N	N	N	N	N	N	Y	N

- Y - Combination is supported
- N - Combination is not supported
- YX - Combination is supported and manual mapping of mechanical units and joints required

> **Note**
>
> Creating a Virtual Controller from Layout only supports tracks of type RTT and IRBTx003 in combination with *positioners*. i.e. IRBTx004 is not supported in combination with the *positioners*.

Continues on next page

Supported RobotWare 6 configurations for positioners, motor units, gear units and track motions

The following table shows various *RobotWare* 6 configurations:

	Single system	MultiMove system	Track motion (non-dynamic model)*	Track motion (dynamic model)**	MU/MTD 1	MU/MTD 2	MU/MTD 3	IRBP L 1	IRBP L 2	IRBP A 1	IRBP A 2	IRBP C	IRBP B	IRBP D	IRBP K	IRBP R	No drives
Track	X		X														1
	X		X		X												2
	X		X					X									2
	X		X					X	X								2
	X		X							X							3
	X		X							X	X						3
	X		X									X					2
	X		X		X												2
	X		X		X			X									3
	X		X		X			X	X								3
	X		X		X							X					3
	X		X		X	X											3
MU/MTD + IRBP	X				X												1
	X				X	X											2
	X				X	X	X										3
	X				X			X									2
	X				X			X	X								3
	X				X					X							2
	X				X					X	X						3
	X				X							X					2
	X				X	X		X									3
	X				X	X		X	X								3
	X				X	X						X					3
IRBP only	X							X									1
	X							X	X								2
	X									X							1
	X									X	X						2
	X											X					1
	X												X				3
	X													X			3

Continues on next page

	Single system	MultiMove system	Track motion (non-dynamic model)*	Track motion (dynamic model)**	MU/MTD 1	MU/MTD 2	MU/MTD 3	IRBP L 1	IRBP L 2	IRBP A 1	IRBP A 2	IRBP C	IRBP B	IRBP D	IRBP K	IRBP R	No drives
	X														X		3
	X															X	3

* Track motion that does not use a dynamic model, applicable for RTT.

**RobotWare *add-in* that contains a dynamic model for track motion is used, applicable for IRBT 4004, 6004, 7004 and 2005.

Note

You can configure dynamic (IRBTx004 and IRBT2005) tracks only for T_ROB1 and T_ROB2. This is limited by RobotWare add-in for track motion.

MU does not support *MultiMove* in Virtual Controller From Layout. Dynamic tracks cannot be combined with MU.

Manual mapping of mechanical units and joints

If the *virtual controller* contains more than one mechanical unit, the number of *tasks* and *base frame* positions of the *mechanism* should be verified in the virtual controller configuration.

1. On the **Controller** tab, in the **Virtual Controller** group, click **Motion Configuration**.

   This opens the **System Configuration** dialog.
2. Select the robot from the node in the hierarchical tree.

   The property page of this node contains controls for mapping and setting axes and joints.
3. Click **Change** to open a dialog box.
4. Manually map the mechanical unit and mechanism joints. Click **Apply**.
5. Modify the base frame positions of the *mechanical unit*.

2.11 Programming MultiMove systems

Overview

The purpose of MultiMove is to let one controller handle several robots. This does not only save on hardware costs, it also allows advanced coordination between different robots and other mechanical units.

Here are some examples of applications

- Several robots can work on the same moving work object.
- One robot can move a work object while other robots work on it.
- Several robots can cooperate to lift heavy objects.

> **Note**
>
> MultiMove was available as an integrated feature in RobotStudio till version 2019.5. For later versions of RobotStudio, this feature is provided as an add-in which must be installed from **Add-Ins Gallery**.

Workflow for setting MultiMove

In *MultiMove* systems, a single robot or *positioner* holds the work piece and other robots operate on it.

1. Select the robots and *paths* to use in the program.
2. Execute motion instructions along the paths.
3. Tune motion behavior, such as tolerances and constraints for *TCP* motion.
4. Generate the *tasks* for the robots.

For detailed information about *MultiMove* in *RobotWare* systems and *RAPID* programs, see *MultiMove application manual*.

Selecting robots and paths

This procedure is for selecting the robots and paths in the station that shall be used for the *MultiMove* program. All robots for the MultiMove program must belong to the same *system*.

1. On the **Home** tab, click **MultiMove**. Click the **Setup** tab below the MutliMove work area.
2. In the work area, click the **System config** bar for expanding the system config section.
3. In the **Select System** box, select the system that contains the robots to program.

   The robots of the selected system are now displayed in the **System grid** below the **Select system** box.
4. For each robot that is used in the program, select the check box in the **Enable** column.
5. For each robot that is used in the program, specify whether it carries the *tool* or the work piece using the options in the **Type** column.

Continues on next page

6 In the work area, click the **Path config** bar for expanding the path config section.
7 Select the **Enable** check box for the tool robot and click the **expand** button. This displays the paths of the robot.
8 Select the order of the paths to execute by specifying them in right order using **Path name** column.
9 For each path that shall be included in the program, select the check box in the **Enable** column.
10 In the **Start Position** section, select the robot in the **Select Robot that other should jump to** box, click **Apply**.
11 When you have set up the robots and paths, continue testing the MultiMove and then tune the motion properties, if necessary.

Executing motion instructions along paths

This procedure is for setting the robot's start position and testing the resulting movements along the *path* sequence.

1 Jog the robots to good start position.
2 On the **Home** tab, click **MultiMove**. Click the **Test** tab at the bottom of the *MultiMove* tab.
3 Optionally, select the **Stop at end** check box to make the simulation stop after moving along the *paths*. Clearing this check box makes the simulation continue in a loop until you click **Pause**.
4 Click **Play** to simulate the motion along the paths based on the current start position.

Pause the simulation to use the following options to fine tune multimove paths.

Action	Description
Examine the robots' positions for critical targets.	Click **Pause** and then use the arrow buttons to move to one *target* at a time.
Jog the robots to new start positions.	Change in positions require new paths and configurations. In most cases, positions near the robots' joint limits shall be avoided.
Go to the **Motion Behavior** tab and remove constraints.	The default setting for the motion properties is free from constraints. If this has been changed, constraints might exist that limit motion more than necessary.

Tuning the motion behavior

Overview

Tuning the motion behavior means to set up rules for the robot's movements, for example, constraints on the *position* or *orientation* of the *tool*. Generally, the MultiMove program will obtain the smoothest motion with the fastest cycle and process times with as few constraints as possible.

Continues on next page

Motion Behavior tab

This is used for specifying constraints and rules for how the robots shall move relative to each other. The default setting is no particular constraints, which results in the fewest joint movements. However, changing the motion behavior might be useful for:

- Locking the orientation or position of the tool.
- Optimizing cycle time or reachability by allowing tolerances.
- Avoiding collisions or singularity by restricting joint motion.

Both Joint Influence and *TCP* Constraints restrict the robot's movements. Changes in these settings might result in lower performance or situations where it is impossible to find proper solutions. The weight values for Joint Weights and TCP Constraints set how much the setting for each joint or TCP direction shall affect the robots relative to each other. It is the difference between the weight values that matters, not the absolute values. If contradicting behaviors have been set, the one with the lowest weight value will win.

Tool Tolerance, instead of restricting, enables more motion. Therefore, tolerances may improve cycle and process times and enhance the reachability of the robots. Tolerances, too, have weight value; here is set how much the robots shall use the tolerance. A low value indicates that the tolerance will be used a lot, while a high value means that the robots will try to avoid using the tolerance.

The joint influence controls the balance of how much the robots will use their joints. Decreasing the weight value for one axis will restrict the motion for this axis, while increasing it will promote motion on this axis relative to alternative axes.

The TCP constraints control the *position* and *orientation* of the *tool*. Enabling a TCP constraint will decrease the motion of the tool and increase the motion of the work piece.

The tool tolerances control the allowed deviation between the tool and the work piece. By default, tolerances are not enabled, which means that no deviation is allowed. Enabling a tolerance, if applicable, might improve motion performance. For example, if the tool is symmetric around its Z axis, you can enable the Rz tolerance without affecting the accuracy of the generated paths.

The tool offset sets a fixed distance between the tool and the paths.

Joint Influence	**Select Robot**	Select the robot's joints to constrain in this box.
	Joints for Robot	Displays the robot's joints and their constraint weights. Each joint is presented in its own row.
	Axis	Displays which axis the constraint affects.
	Influence	Specify how much the motion for the axis is constrained. 0 means a locked axis, while 100 means no constraint relative to default constraint values.
TCP Constraints	**Active TCP**	This grid displays the position and rotations of the TCP together with their constraint weights.

Continues on next page

	Enable	Select this check box to activate the constraint for this TCP pose.
	Pose	Displays the TCP pose that is affected by the constraint.
	Value	Specify the pose value to constrain at. Either type the value, or click the Pick from TCP button to use the values of the current TCP position.
	Influence	Specify how much the motion for the TCP value is constrained. 0 means a locked TCP at this pose, while 100 means no constraint relative to default constraint values.
Tool Tolerance	Enable	Select this check box to activate the tolerance for this tool pose.
	Pose	Displays the tool pose that is affected by the constraint.
	Value	Specify the pose value to apply the tolerance around.
	Influence	Specify the size of the tolerance. 0 means no deviation is allowed, while 100 means all deviations are allowed.
Tool Offset	**Enable**	Select this check box to activate the offset for this tool pose.
	Pose	Displays the tool pose that is affected by the offset setting.
	Offset	Specify the value of the offset here.

Modifying the joint influences

The joint influence controls the balance of how much the robots will use their joints. Decreasing the weight value for one axis will restrict the motion for this axis, while increasing it will promote motion on this axis relative to alternate axes.

1. On the **Home** tab, in the **Path Programming** group, click **MultiMove**. The **MultiMove** pane opens.
2. In the **MultiMove** pane, click **Motion Behavior**.
3. Expand the **Joint Influence** group by clicking its title bar.
4. In the **Select Robot** box, select the robot whose joint influence you want to modify.

   The weight values for the robot axes are now displayed in the grid.
5. For each axis whose motion you want to restrict or promote, adjust the Weight value. A lower value restricts, and a higher value promotes, motion on that axis.

Modifying the TCP constraints

The joint influence controls the balance of how much the robots will use their joints. Decreasing the weight value for one axis will restrict the motion for this axis, while increasing it will promote motion on this axis relative to alternative axes.

1. On the **Home** tab, in the **Path Programming** group, click **MultiMove**. The **MultiMove** pane opens.

Continues on next page

2 In the **MultiMove** pane, click **Motion Behavior**.
3 Expand the **TCP Constraints** group by clicking its title bar.
The directions and rotations in which you can constrain the TCP's motion are now displayed in the grid
4 For each pose you want to constrain, select the **Enable** check box and specify the constraint values (location in the TCP coordinate system). To use the values from the current TCP position, click **Pick from TCP**.
5 Optionally, adjust the **Weight** value for the constraint. A low value results in a harder constraint, while a high value allows a larger deviation.

Modifying the tool tolerance

The joint influence controls the balance of how much the robots will use their joints. Decreasing the weight value for one axis will restrict the motion for this axis, while increasing it will promote motion on this axis relative to alternative axes.

1 On the **Home** tab, in the **Path Programming** group, click **MultiMove**. The **MultiMove** pane opens.
2 In the **MultiMove** pane, click **Motion Behavior**.
3 Expand the **Tool Tolerance** group by clicking its title bar.
The directions and rotations in which you can enable tolerances are now displayed in the grid.
4 For each offset you want to set, select the **Enable** check box.
5 In the **Value** column, specify the allowed deviation.
6 Optionally, adjust the **Weight** value for the tolerance. A low value increases the use of the tolerance, while a high value promotes motion that do not use the tolerance.

Modifying the tool offset

The tool offset sets a fixed distance between the *tool* and the *paths*.

1 On the **Home** tab, in the **Path Programming** group, click **MultiMove**. The **MultiMove** pane opens.
2 In the **MultiMove** pane, click **Motion Behavior**.
3 Expand the **Tool Offset** group by clicking its title bar.
The directions and rotations in which you can set offsets are now displayed in the grid.
4 For each offset you want to set, select the **Enable** check box.
5 In the **Offset** column, specify the offset distance.

Creating paths

After testing the MultiMove program, convert the temporary move instructions that MultiMove function uses to ordinary *paths*. Use the following steps to create paths for the *MultiMove* program.

1 On the **Home** tab, in the **Path Programming** group, click **MultiMove**. The **MultiMove** pane opens.
2 In the **MultiMove** pane, click **Create Paths** tab.
3 Click the title bar to expand the **Settings** group.

Continues on next page

4 Optionally, change the settings in the following boxes:

Box	Description
Start ID	Specify the first ID number for the *synchronization* of the instructions for the robots.
ID step index	Specify the increment between ID numbers.
Sync ident prefix	Specify a prefix for the syncident variable, which connects the sync instructions in the tasks for the tool robot and the work piece robot with each other.
Task list prefix	Specify a prefix for the tasklist variable, which identifies the tasks for the tool robot and the work piece robot to synchronize .

5 Click the title bar to expand the **WP Robot Settings** group and then check the settings in the following boxes:

Box	Description
WP Workobject	Specify the *workobject* to which the targets generated for the workpiece robot shall belong.
WP TCP	Specify which *tooldata* the work piece shall use when reaching its targets.
Path prefix	Specify a prefix for the generated *paths*.
Target prefix	Specify a prefix for the generated *targets*.

6 Click the title bar to expand the **Generate path** group and then click **Create Paths**.

Setting up MultiMove without using Multimove functions

In addition to using the functions that calculate and create optimized MulitMove paths, you can program MulitMove manaully using a combination of the ordinary programming tools in RobotStudio and a set of tools specific for MultiMove programming.

The main actions for programming MulitMove manually are outlined below. Not all actions might be necessary, but the order in which they shall be carried out depends on the contents of the *station* and your goals.

Action	Description
Creating Tasklists and Syncidents	This data specifies the tasks and paths that shall be synchronized with each other.
Adding and updating ID arguments to the instructions to *synchronize*	To add IDs to the intructions, you can use one of the following methods: Using the *Recalculate ID tool* to add and update IDs for instructions in paths that already are synchronized. Using the Convert path to MultiMove path tool to add IDs to instructions in paths that have not yet been synchronized.
Adding and adjusting Sync instructions to the paths.	Add `SyncMoveon/Off` or `WaitSyncTask` instructions to the paths to *synchronization* and set their tasklist and Syncident parameters.
Teaching MultiMove instructions	It is also possible to jog all robots to the desired positions and then teach instructions to new synchronized paths.

2.12 Replacing robot in a station

1 In the **Layout** browser, under **Mechanisms**, right-click the robot and then click **Replace Robot**.

  The **Replace Robot** window opens.

2 In the **Name** list, select a robot model to replace the current robot.

3 Click **OK**.

> **Note**
>
> Robots with multiple mechanisms are not supported, for example, IRB 14000.

2.13 Back up a system

Overview

When backing up a system you copy all the data needed to restore the system to its current state:

- Information about software and options installed on the system.
- System's home directory and all its content.
- All robot programs and modules in the system.
- All configuration and calibration data of the system.

Prerequisites

To backup a system you must have:

- Write access to the controller
- Logged on to the controller with appropriate grants.

Creating a Backup

To create a backup, follow these steps:

1. In the **Controller** browser, select the system you want to backup from the browser.
2. Right-click and select **Create Backup**.

   The **Create Backup** dialog box appears.
3. Enter a new backup name and specify a location for the backup, or keep the default ones.

   You can create backup either in the PC or in the controller disk using the following options, **Browse File System** or **Browse Controller**.
   - To create a backup in the PC, select **Browse File System** and then select the destination folder.
   - To create a backup in the controller disk, select **Browse Controller** and then select the destination folder in the controller disk.

     Note

     Do not select **Home** folder when you specify the destination folder for backup.
4. Select the checkbox **Backup to archive file** to archive the backup as a `.tar` file. The backup can be archived in both the local PC and the controller disk.

   Note

   For RobotWare versions prior to 6.06, the backup is archived as a `.zip` file in the local PC alone.
5. Click **OK**.

   The progress of the backup is displayed in the Output window.

Continues on next page

Note

Do not create backups while performing critical or sensitive robot movements.This may affect the accuracy and performance of the movement. To make sure that no backup is requested while in critical areas, use a system input with the action value `Disable Backup`.

The system input signal can be set from RAPID for the parts of the code that are critical for disturbances or remove the setting otherwise. For more information, see Technical reference manual - System parameters.

Backup folder

When the backup is complete you will have a folder with the name of the backup in the specified location. This folder contains a set of subfolders which together comprise the backup.

CAUTION

If the contents of the Backup folder are changed, then it will not be possible to restore the system from backup.

Subfolders	Description
BACKINFO	Contains information necessary for re-creating the system's software and options from the mediapool.
HOME	Contains a copy of the system’s home directory content.
RAPID	Contains one subfolder for each task in the system’s program memory. Each of these task folders contains separate folders for program modules and system modules.
SYSPAR	Contains the system's configuration files.

Note

The contents of the PIB board of a IRC5P system (a controller system for painting) will not be included with the regular RobotStudio backup. Please use the backup function of the FlexPaintPendant to include the PIB content.

This page is intentionally left blank

3 Programming robots in the 3D environment

3.1 Understanding offline programming

Overview

This section provides an introduction to the coordinate systems used mostly for offline programming. In RobotStudio, you can either use the coordinate systems or the user-defined coordinated systems for co-relating elements and objects in the *station*.

Hierarchy

The coordinate systems are co-related hierarchically. The origin of each coordinate system is defined as a position in one of its ancestries. The following are the descriptions of the commonly used coordinate systems.

Tool Center Point Coordinate system

The tool center point coordinate system, also called TCP, is the center point of the *tool*. You can define different TCPs for one robot. All robots have one predefined TCP at the robot's tool mounting point, called *tool0*.

When a program runs, the robot moves the TCP to the programmed position.

RobotStudio World Coordinate system

The RobotStudio world coordinate system represents the entire station or robot cell. This is the top of the hierarchy to which all other coordinate systems are related.

Base Frame (BF)

The base coordinate system is called the Base Frame (BF). Each robot in the station, both in RobotStudio and the real world has a base coordinate system which is always located at the base of the robot.

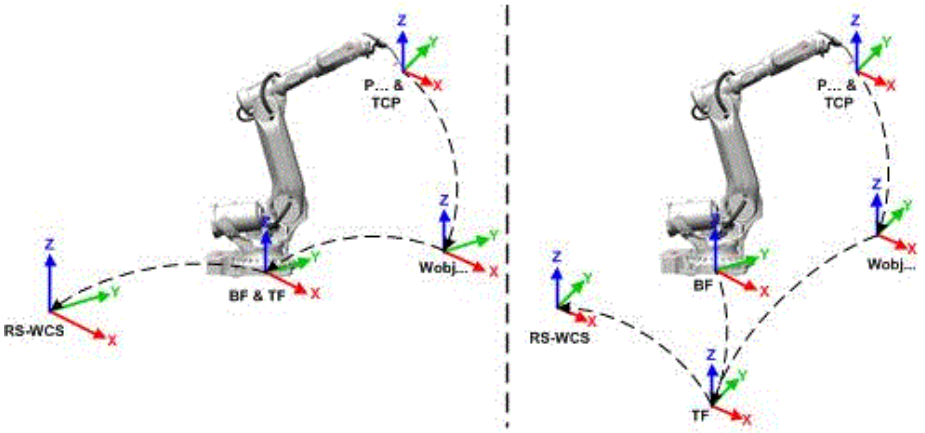
Task Frame (TF)

The task frame represents the origin of the robot controller world coordinate system in RobotStudio.

The following picture illustrates the difference between the base frame and the task frame.

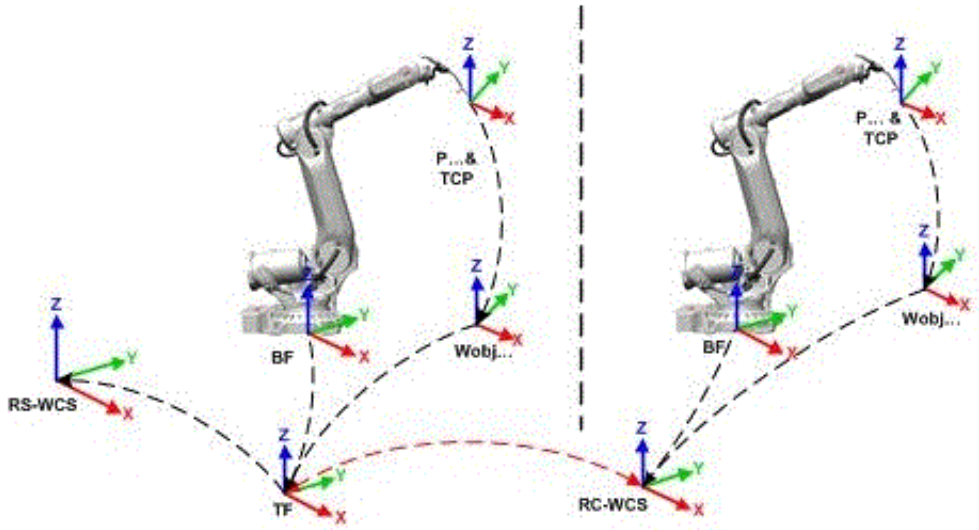
Continues on next page

In the picture to the left, the task frame is located at the same position as the robot base frame. In the picture to the right, the task frame has been moved to another position.



en1000001303

The following picture illustrates how a task frame in RobotStudio is mapped to the robot controller coordinate system in the real world. For example, on the shop floor.



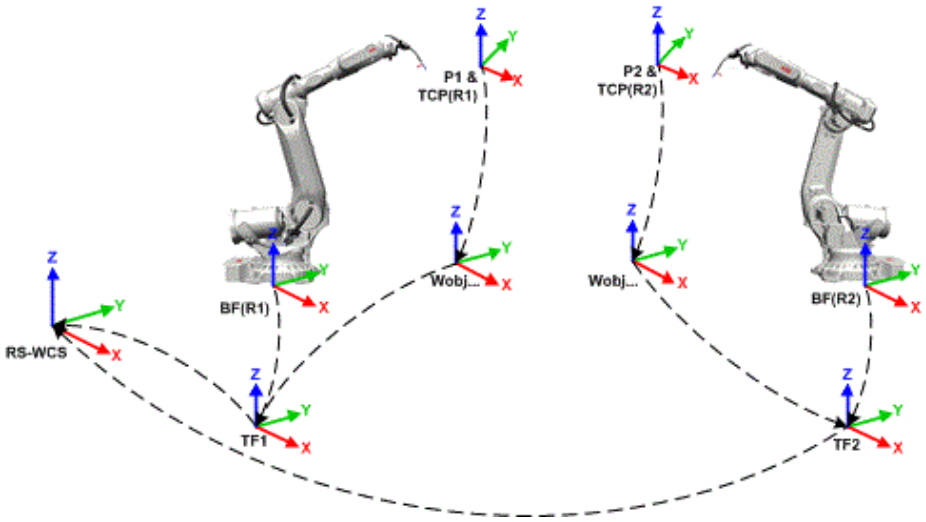
xx1900001116

RS-WCS	World coordinate system in RobotStudio
RC-WCS	World coordinate system as defined in the robot controller. It corresponds to the task frame of RobotStudio.
BF	Robot Base Frame
TCP	Tool Center Point
P	Robot target
TF	Task Frame
Wobj	Workobject

Continues on next page

Stations with multiple robot systems

For a single robot system, RobotStudio's task frame corresponds to the robot controller *world coordinate system*. When several controllers are present in the station, the *task frame* allows the connected robots to work in different coordinate systems. That is, the robots can be located independent of each other by defining different task frames for each robot.



en1000001442

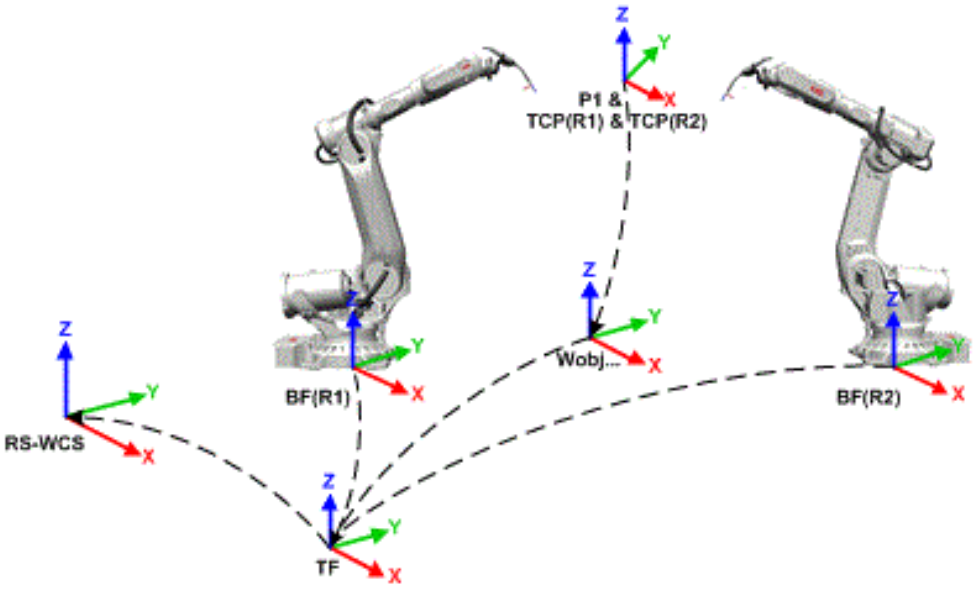
RS-WCS	World coordinate system in RobotStudio
TCP(R1)	Tool Center Point of robot 1
TCP(R2)	Tool Center Point of robot 2
BF(R1)	Base Frame of robot system 1
BF(R2)	Base Frame of robot system 2
P1	Robot target 1
P2	Robot target 2
TF1	Task Frame of robot system 1
TF2	Task Frame of robot system 2
Wobj	Workobject

Continues on next page

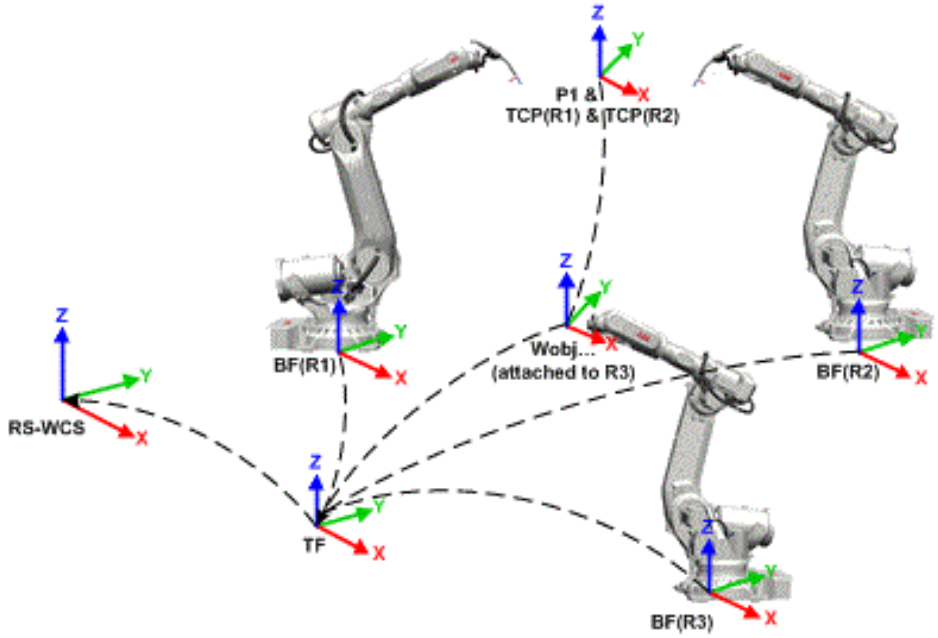
MultiMove Coordinated systems

The *MultiMove* functions helps you create and optimize programs for MultiMove systems where one robot or *positioner* holds the work piece and other robots operate on it.

When using a robot system with the RobotWare option *MultiMove Coordinated*, it is important that the robots are working in the same coordinate system. As such, RobotStudio do not allow *task frames* of the controller to be separated.



en1000001305



en1000001306

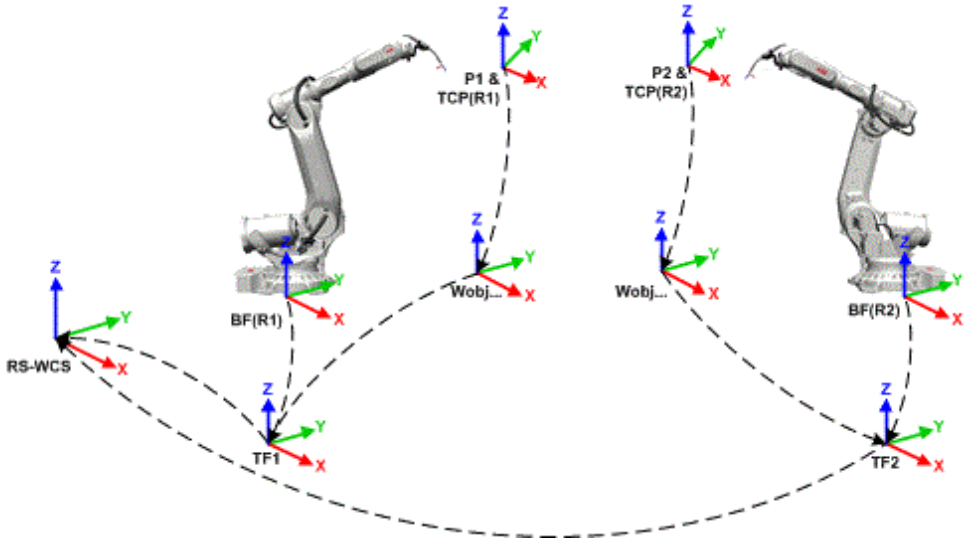
RS-WCS	World coordinate system in RobotStudio
TCP(R1)	Tool Center Point of robot 1

Continues on next page

TCP(R2)	Tool Center Point of robot 2
BF(R1)	Base Frame of robot 1
BF(R2)	Base Frame of robot 2
BF(R3)	Base Frame of robot 3
P1	Robot target 1
TF	Task Frame
Wobj	Workobject

MultiMove Independent systems

For a *virtual controller* with the RobotWare option *MultiMove Independent*, robots operate simultaneously and independently while being controlled by one controller. Even though there is only one robot controller *world coordinate system*, robots often work in separate coordinate systems. To allow this setup in RobotStudio, the *task frames* for the robots can be separated and positioned independent of each other.



en1000001308

RS-WCS	World coordinate system in RobotStudio
TCP(R1)	Tool Center Point of robot 1
TCP(R2)	Tool Center Point of robot 2
BF(R1)	Base Frame of robot 1
BF(R2)	Base Frame of robot 2
P1	Robot target 1
P2	Robot target 2
TF1	Task Frame 1
TF2	Task Frame 2
Wobj	Workobject

Continues on next page

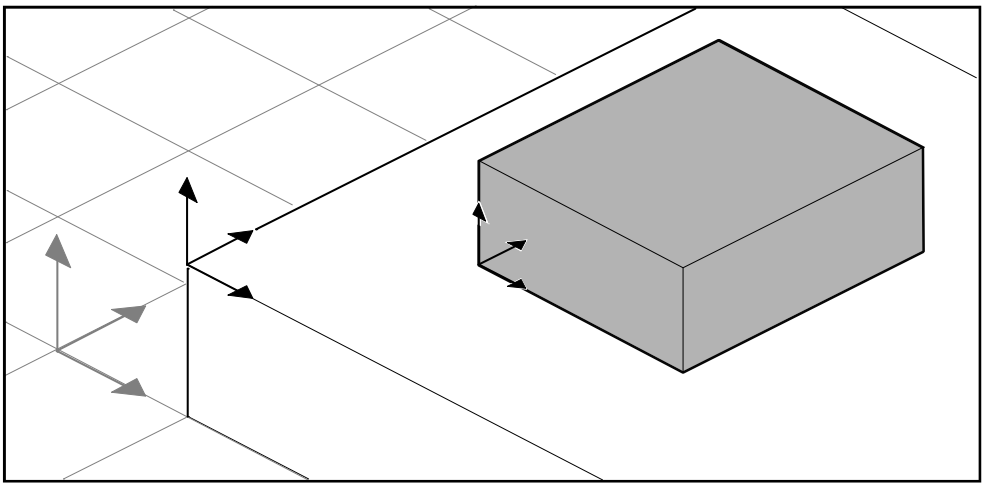
Workobject coordinate system

The *workobject* normally represents the physical work piece. It is composed of two coordinate systems: the *User frame* and the *Object frame*, where the latter is a child to the former. When programming a robot, all *targets* (positions) are related to the object frame of a *workobject*. If no other workobject is specified, the targets will be related to the default *Wobj0*, which always coincides with the world coordinate system of the robot.

Using workobjects provide the chance to easily adjust robot programs with an offset, if the location of the work piece has been changed. Thus, workobjects can be used for calibrating *offline* programs. If the placement of the fixture or work piece relative to the robot in the real station does not completely match the placement in the offline *station*, adjust the position of the workobject.

Workobjects are also used for coordinated motion. If a workobject is attached to a *mechanical unit* (and the system uses the option for coordinated motion), the robot will find the *targets* in the workobject even when the mechanical unit moves the workobject.

In the picture below the grey coordinate system is the *world coordinate system*, and the black ones are the object frame and the user frame of the workobject. Here the user frame is positioned at the table or fixture and the object frame at the work piece.



xx0500001519

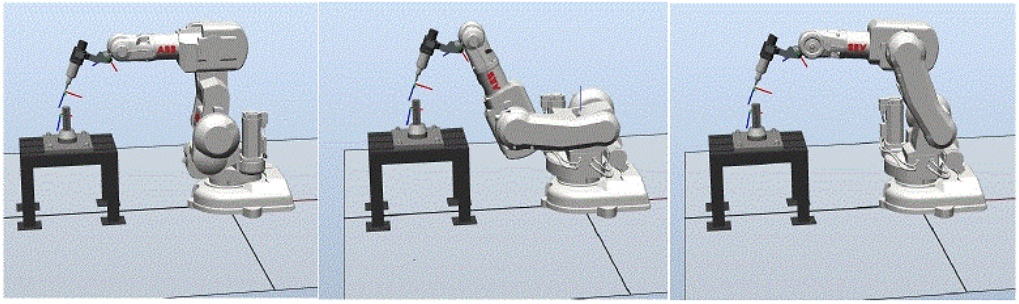
User coordinate systems

User coordinate systems are used for creating reference points in the *station*. For example, you can create user coordinate systems at strategic points in the station to simplify programming.

3.2 Robot axis configurations

Axis configurations

Targets are defined and stored as coordinates in a *workobject* coordinate system. When the controller calculates the position of the robot axes for reaching the target, it will often find more than one possible solution to configuring the robot axes.



configur

To distinguish between the different configurations, all targets have a configuration value that specifies the quadrant in which each axis shall be located.

Storing axis configurations in targets

For targets that are taught after jogging the robot to the position, the used configuration will be stored in the *target*.

Targets created by specifying or calculating *positions* and *orientations* get a default configuration value (0,0,0,0), which might not be valid for reaching the target.

Common problems related to robot axis configurations

It is most likely that targets created by other ways than jogging cannot be reached at their default configuration.

Even if all targets in a path have reachable configurations, you might come across problems when running the path if the robot cannot move from one configuration to the other. This is likely to occur where an axis shifts greater than 90 degrees during linear or circular movements.

Common solutions for configuration problems

To resolve the problems described above, assign a valid configuration to each target and verify that the robot can move along each *path*. You can also turn configuration control off, that is, ignore the stored configurations and let the robot find working configurations at runtime. This must be performed carefully to avoid unexpected results.

In the absence of working configurations, reposition the work piece, reorient *targets* (if acceptable for the process) or add an *external axis* that either moves the work piece or the robot for increasing reachability.

Continues on next page

How configurations are denoted

The robot's axis configurations are denoted by a series of four integers, specifying in which quadrant of a full revolution significant axes are located. The quadrants are numbered from zero for positive (counterclockwise) rotation and from -1 for negative (clockwise) rotation.

For a linear axis, the integer specifies the range (in meters) from the neutral position in which the axis is located.

A configuration for a six-axis industrial robot (like IRB 140) may look like:

```
[0-121]
```

The first integer (0) specifies the position of axis 1: somewhere in the first positive quadrant (between 0 and 90 degrees rotation).

The second integer (-1) specifies the position of axis 4: somewhere in the first negative quadrant (between 0 and -90 degrees rotation).

The third integer (2) specifies the position of axis 6: somewhere in the third positive quadrant (between 180 and 270 degrees rotation).

The fourth integer (1) specifies the position of axis x, a virtual axis used for specifying the wrist center in relation to other axes.

Configuration control

When executing a robot program, you can choose whether to control configuration values. If configuration control is turned off, configuration values stored with the targets are ignored, and the robot will use the configuration closest its current configuration for reaching the *target*. If turned on, it will only use the specified configuration for reaching the targets.

Configuration control can be turned off and on for joint and linear movements independently and is controlled by the *ConfJ* and *ConfL* action instructions.

Turning configuration control off

Running a program without configuration control may result in different configurations each time a cycle is executed: When the robot returns to the start position after completing a cycle, it may choose a different configuration than the original.

For programs with linear move instructions this might cause a situation where the robot gets closer and closer its joint limits and eventually will not be able to reach the target.

For programs with joint move instructions this might cause sweeping, unpredictable movements.

Turning configuration control on

Running a program with configuration control forces the robot to use the configurations stored with the *targets*. This results in predictable cycles and predictable motions. In some situations, however, like when the robot moves to a target from an unknown position, using configuration control may limit the robot's reachability.

When programming *offline*, you must assign a configuration to each target if the program shall be executed with configuration control.

Continues on next page

Visualizing the spatial hierarchy of an object

Spatial hierarchy refers to the hierarchy of transformations in the offline programming environment. To visualize this, On the **Home** tab, in **Graphics Tools**, click **Show**/**Hide** and then select **Parent Frame** to set it **Active**. With the **Parent Frame** in **Active** state, the spatial hierarchy of the selected object with its immediate parent will be shown with an arrow in the direction of the spatial parent. The following table shows the spatial hierarchy between objects.

Object	Spatial parent
target	workobject object frame
workobject object frame	workobject user frame
workobject user frame	task frame (if not moved by a mechanism
workobject user frame	mechanism flange (if moved by a mechanism)
tooldata	robot flange (if held by robot)
tooldata	task frame (if not held by robot)
mechanism	task frame
task frame	world

3.3 Creating a workobject

1. On the **Home** tab, in the **Path Programming** group, click **Other** and select **Create Workobject**.

   The **Create Workobject** dialog box appears.
2. In the **Misc Data** group, enter the values for the new *workobject*.
3. In the **User Frame** group, do one of the following:
   - Set the position of the user frame by entering values for the **Position x,y,z** and the **Rotation rx, ry, rz** for the workobject by clicking in the **Values** box.
   - Select the user frame by using the **Frame by points** dialog box.
4. In the **Object Frame** group, reposition the object frame relative to the user frame by using the following steps:
   - Set the position of the object frame by selecting values for **Position x, y, z** by clicking in the **Values** box.
   - For the **Rotation rx, ry, rz**, select **RPY (Euler XYZ)** or **Quaternion**, and enter the rotation values in the **Values** dialog box.
   - Select the object frame by using the **Frame by points** dialog box.
5. In the **Sync Properties** group, enter the values for the new workobject.
6. Click **Create**. The workobject gets created and displayed under the **WorkObjects and Targets** node under the robot node in the **Paths&Targets** browser.

3.4 Creating a path with targets and move instructions

Creating an empty path

1. In the **Paths&Targets** browser, select the motion task in which you want to create the *path*.
2. In the **Home** tab, from the **Path Programming** group, click **Path** and then click **Empty Path**.
3. Right-click a workobject and select **Set as active**.
4. To set the correct motion properties for the *targets*, select the active process in the **Change Active Process** box in the **Elements** toolbar.
5. If the active template is set to **MoveAbsJoint**, then:
   - A target that is dragged into a path will be converted into a **jointtarget** (recognized by a different icon on in the browser).
   - One target can not be used as different types, for example, **MoveJoint**, but must be deleted and re-created.
   - When the target has been *synchronized* with the *virtual controller*, the joint target values will be calculated and inserted in the *RAPID* program.

   It is possible to call a procedure from another procedure. In the **Paths & Targets** browser, click **Paths & Procedures** to view the folder structure. You can move procedures between folders using a drag-and-drop operation.

Add to Path

Creating a move instruction based on an existing target

1. Select the *target* for which to create the move instruction.
2. Right click on the selected target. and click **Add to Path**.

   The move instruction will appear under the *path* node as a reference to the original *target*.

Align TCP to UCS

Use the following procedure to align the TCP to UCS.

1. In the **Layout** browser, right-click a robot. The context menu opens.
2. In the context menu, click **Align TCP**. The orientation of the active tool will be aligned to the UCS.

   Here the axis closest to the UCS will be aligned and the remaining axes are oriented accordingly.

3.5 Creating a path from an edge or curve

3.5.1 AutoPath

Overview

AutoPath helps in generating accurate paths (linear and circular) based on CAD *geometry*.

Prerequisites

A geometric object with edges, curves, or both must be available.

Creating a path automatically

Use the AutoPath feature to create paths from *curves* or along the edges of a surface. To create a path along a surface, use selection level *Surface*, and to create a path along a curve, use selection level *Curve*. When using Selection level surface, the closest edge of the selection will be picked for inclusion in the *path*. An edge can only be selected if connected to the last selected edge.

When using selection level Curve, all edges along the curve will be added to the list. When using selection level Surface, all edges on a surface can be added by holding the SHIFT button while selecting an edge. The *Approach* and *Travel* directions as defined in RobotStudio options are used to define the *orientation* of the created *targets*.

Use this procedure to automatically generate a path.

1. In the **Home** tab, click **Path** and select **AutoPath**.

   The AutoPath tool appears.
2. Select the edge or curve of the geometric object for which you want to create a path.

   The selection is listed as edges in the **tool** window.

   Note

   - If in a geometric object, you select curve (instead of an edge), then all the points that result in the selected curve gets added as edges to the list in the **Graphic** window.
   - Always select continuous edges.
3. Click **Remove** to delete the recently added edge from the **Graphic** window.

   Note

   To change the order of the selected edges, select the **Reverse** check box.

Continues on next page

4 You can set the following **Approximation Parameters**:

Select or enter values in	to
MinDist	Set the minimum distance between the generated points. That is, points closer than the minimum distance are filtered.
Tolerance	Set the maximum deviation from the geometric description allowed for the generated points.
MaxRadius	Determines how large a circle radius has to be before considering the circumference as a line. That is, a line can be considered as a circle with infinite radius.
Linear	Generate a linear move instruction for each target.
Circular	Generate circular move instructions where the selected edges describe circular segments.
Constant	Generate points with a Constant distance.
End Offset	Set the specified offset away from the last target.
Start Offset	Sets the specified offset away from the first target.

The **Reference Surface** box shows the side of the object that is taken as normal for creating the *path*.

Click **More** to set the following parameters:

Select or enter values in	to
Approach	Generate a new target at a specified distance from the first target.
Depart	Generates a new target at a specified distance from the last target.

5 Click **Create** to automatically generate a new path.

A new path is created and move instructions are inserted for the generated *targets* as set in the Approximation parameters.

Note

The targets are created in the active *workobject*.

6 Click **Close**.

3.6 Creating a collision free path between two targets or move instructions

Overview

Use the Collision Free Path feature to create a path between two or more targets, joint targets or move instructions avoiding obstacles in the environment. The path will be generated with *MoveAbsJ* instructions and joint targets. This feature is available only with RobotStudio premium license.

In the **Home** tab, from the **Path Programming** group, click **Path** and then click **Collision Free Path**, the **Collision Free Path** tool window opens.

Note

External track motion is not considered when creating collision free paths.

Option	Description
Single Path	Creates a single collision free path connecting all targets.
Multiple Paths	Creates collision free paths from one target to many targets.
Reverse	Reverses the order of targets and thus reverses the direction of the generated path(s). If **Multiple Paths** mode is selected, this creates collision free paths from many targets to one target.
Optimize order Visible when **Single Path** is selected.	Optimizes the order of via targets to minimize the length of the collision free path.
Optimize zones Visible when **Single Path** is selected.	Enlarges the zones of the via targets as much as possible. Zones are always optimized for additional targets that are added between via targets.
Return to start Visible when **Multiple Paths** is selected.	Makes the generated paths return to the start position.
Add	When the **Add** mode button is enabled, targets can be added by selecting them in the **Paths&Targets** browser or in the 3D graphics view Targets can be rearranged in the **Collision Free Path** tool window using a drag-and-drop operation. Similarly, targets can be added from the **Paths&Targets** browser to the **Collision Free Path** tool window using a drag-and-drop operation.
Clear All	Clears all targets.
Minimum distance to obstacles	Specify the minimum allowed distance between robot/tool/load and obstacles in the environment along the path.
Minimum robot self-collision distance	Set the minimum allowed distance between tool, load and robot links.
Speed	Select the speed of the TCP in the generated instructions.

- The tool window can be opened by selecting two or more targets, joint targets or move instructions and by selecting **Create Collision Free Path...** from the context menu.

Continues on next page

- Objects can be excluded from the collision free path planning by deselecting **Include in Collision Free Path Planning** in the **Path Planning** context menu. All objects except sensors are included by default.
- The tool attached to the robot, and objects attached to the tool, are included in the collision free path planning by default.
- The path will be generated in the task, and with the work object and tool, that are selected in the **Settings** group in the **Home** tab.
- When an object (component) is selected in the **Layout** browser, its collision geometry will be highlighted in the **Graphics** window. When a group is selected, all objects in that group get their corresponding collision geometry highlighted.

  Similarly, when you click on the collision geometry of an object in the **Graphics** window, the corresponding object gets selected in the **Layout** browser, which helps in identifying the object to which the collision geometry belongs to.
- **Include in Collision Free Path Planning**:

  Collision Geometry Detail Level: Provides options to select the level of detail on the collision geometry for collision free path planning calculations, the options are **Low**, **Medium**, **High** and **Custom**.
  - **Low**: Generates a collision geometry with only a single convex hull (fast).
  - **Medium**: Generates a collision geometry that contains multiple convex hulls, but fewer than the **High** option. This option provides a balance between accuracy and performance compared to the **Low** and **High** options.
  - **High**: Generates collision geometry with high detail level (slow).
  - **Custom**: Generates a collision geometry based on the **Number of convex hulls** and **Resolution** parameters. A higher **Resolution** value enhances the accuracy of the collision geometry. Generating collision geometry with a higher resolution is a compute-intensive feature, which can slow down the collision geometry generation.
- **Show Collision Geometries**: All collision geometries that are selected to be part of the path planning will be generated and displayed.
- **Use Group's Collision Geometry**: When this option is selected, a single collision geometry is generated for a Component Group instead of a collision geometry for each contained part. This is useful to improve collision free path planning performance when a Component Group contains many parts close to each other.
- **Visualizing Collision Free Path results**: When the collision free path creation fails, error messages are listed in the **Collision Free Path** tab in the **Output** window. Hover over a collision related error message, the tooltip displays a probable workaround to avoid the error. Double-click a collision error message or right-click and then click **Visualize**, robot moves to the target and the collision geometry gets highlighted. When the target is out of reach, double-click or right-click the error item to visualize the errors.

3.7 Configuring a stationary tool

Stationary tool is a device that stands in a fixed location. The robot manipulator picks up the work piece and brings it to the device to perform specific tasks, such as gluing, grinding or welding. In this configuration, the robot holds the *workobject* and hence is attached to the robot whereas the *tool* is set as stationary.

1. On the **Home** tab, in the **Path Programming** group, click **Other** and select **Create Workobject**.

   The **Create Workobject** dialog box appears.
2. In the **Misc Data** group, enter the values for the new workobject.
   - Set **Robot holds workobject** to *True*.
   - In **Moved by mechanical unit**, select the robot, for example, *ROB_1*.
   - Set **Programmed** to *False*.

   If the workobject is attached to the robot using the **Attach to** option, then these settings get applied automatically.
3. On the **Home** tab, in the **Path Programming** group, click **Other** and select **Create Tooldata**.

   The **Create Tooldata** dialog box appears.
4. In the **Misc Data** group, enter the values for the new *tooldata*.

   Set **Robot holds tool** to *False*.

3.8 Define arm configurations for the targets

Auto Configuration

The Auto Configuration function runs through the *path* and optimizes the configurations with respect to their preceding *target*. There are two options, either all of the configurations can be optimized, or only the configurations for linear and circular move instructions. Use the *Configurations tool* on the individual joint targets to change the configuration of intermediate joint move instructions.

In the **Paths&Targets** browser, right-click a path, and select **Auto Configuration**.

The robot now steps through each *target* in the path and sets the configurations.

Auto Configuration for a dual-arm robot

A normal robot arm has six axis or degrees of freedom of movement whereas a dual-arm robot has seven degrees of movement. Hence by configuring its arm angles, a dual-arm robot can adapt several ways to reach a *target*. You can set the arm angle while applying auto configuration.

3.9 Testing positions and motions

Jumping to a target

1. In the **Paths&Targets** browser, browse to the target to jump to through the **Controller**, **Tasks** and **WorkObjects** nodes.
2. Click **Jump to target**.

   If the *target* has a valid configuration for the robot axes stored, the active *TCP* of the robot will immediately be positioned at the target. If no valid configuration is stored, right click on the target and select configuration the **Configuration** dialog box is displayed.
3. In the **Configuration** dialog box, select a suitable configuration solution and click **Apply**. The selected configuration is now stored with the *target*.

> **Note**
>
> Deactivate the configuration check while using **Jump to target**. The robot uses the configuration solution that is closest to the current position for reaching the target.

Copying TCP target values

It is possible to copy the current coordinates of the active TCP to the Windows clipboard. These coordinates are copied in relation to the active work object as a RAPID robtarget declaration. Values of the active external axis are included, for example, while using this feature to copy the current TCP of a robot placed on a track, the external axis values are also included in the robtarget declaration.

To use this feature, in the **Layout** browser, right-click the object and then click **Copy TCP as robtarget**.

Copying position as jointtarget

The current joint values of the robot and external axis can be copied to the Windows clipboard as a RAPID jointtarget declaration.

To use this feature, in the **Layout** browser, right-click the object and then click **Copy Pose as jointtarget**. The copied values can be pasted from the clipboard to a RAPID module as jointtarget text.

3.10 Generating the RAPID program

Synchronizing to the station

1. On the **RAPID** tab, in the **Access** group, click the arrow next to the **Synchronize** icon, and then click **Synchronize to Station**.
2. Select the paths to be *synchronized* to the *station* from the list.
3. Click **OK**.

   The message **Synchronization to Station completed** is displayed in the **Output** window.

RAPID can be synchronized from the files of the HOME folder as an alternative:

RAPID procedures without parameters only can be synchronized to RobotStudio.

- In the **Controller** browser, under **Home** folder, right-click any file and then click **Synchronize to Station** to synchronize data and procedures of the selected file into the station.
- In the **Controller** browser, under **Home** folder, right-click any file and then click **Synchronize to File** to synchronize data and procedures from the station into the selected file.

  Before selecting this option, you must synchronize the selected file to the station. Using the **Synchronize to File** option can conflict with the synchronize to RAPID option.

3.11 Modifying target orientations

1. In the **Paths&Targets** browser, right-click a target and select **Modify Target**.
2. Select **Set Position**, the **SetPosition** browser opens.
3. From the **Reference** list, select a *coordinate* system. This will be used as a reference for *position* and orientation values.
4. Select **RPY angles (Euler ZYX)**, and enter the new *orientation* values.
5. Click **Apply**.
6. In the **RAPID** tab, click **Synchronize** and then select **Sychronize to RAPID** to update the *RAPID* program.

Replicating position and orientation to multiple objects

It is possible to copy the position and orientation of any object in a station and apply the same to another object. The position and orientation of the object can be copied either relative to the world coordinate system or relative to its parent's position and orientation. The same reference will be used when applying the position and orientation to another object.

1. In the Layout browser, right-click an object, from the context menu click **Position** and then click **Copy Position and Orientation**.

   This option is also available in the **Paths&Targets** browser, right-click the object and then click **Copy Position and Orientation** from the context menu.
2. Click **Relative World** or **Relative Parent** as required.
3. Select the object(s) where position or orientation must be applied, click **Apply Position and Orientation** .
4. Select the required **Position and Orientation**, **Position** or **Orientation** as required to apply the copied values to the selected object.

Note

This option is not available for path and face.

4 Understanding RAPID editor

4.1 Working with RAPID editor

Overview

The *RAPID* tab provides tools and functionalities for creating, editing, and managing RAPID programs. You can manage RAPID programs which are *online* on a *robot controller*, *offline* on a *virtual controller*, or standalone programs. Use RAPID editor to view and edit programs loaded into a robot controller, both robot and virtual. The integrated RAPID editor is useful for editing all robot *tasks* other than robot motion. Use RAPID editor to edit the RAPID code of the RAPID *modules*.

General RAPID Editor features

The following are the general features of the RAPID Editor:

- **Read-only documents** - If the document is read-only (for example, write access unavailable), then the background of the editor area will be light gray instead of the normal white. Typing in an editor that is in the read-only state results in a dialog asking you whether RobotStudio should acquire write access.
- **Context-sensitive help** - Pressing F1 when the cursor is on a RAPID programming construct, such as an instruction, opens the related section in the RAPID reference manual, instead of the main RobotStudio help.
- **Auto-indent cursor on return** - When you press Enter, the cursor is automatically indented by the appropriate amount on the following line. For example, after typing a PROC header, pressing ENTER will indent the cursor one tab (or the corresponding number of spaces, depending on settings).
- **Completion list** - When you type in code in the editor, a pop-menu which lists possible code suggestion maybe displayed depending on the kind of RAPID code construct being written. The suggestions listed also depend on where in the document the cursor is.

  Pressing comma (,), semi-colon (;), colon (:), equal sign (=), Spacebar, Tab, or Enter keys automatically inserts the selected item. Press Esc to cancel the list.
- **Collapsible regions** - All regions of the code can be collapsed. In the **RAPID** tab, click **Outlining** to access this functionality. You can either collapse or expand all regions or can select a particular region to collapse or expand.
- **Zooming in and out** - In the RAPID editor you can zoom in and zoom out of the code display. Click the plus (+) and minus (-) buttons at the top right corner of the RAPID editor window to zoom in and zoom out.

> **Tip**
>
> The Zoom in Zoom out feature is also present in the *RAPID Tasks*, *Rapid Editor*, *Configuration Editor*, *Event viewer*, and *I/O* windows.

Continues on next page

- **Cut, copy, paste and drag and drop** - These standard commands for clipboard handling of text are supported.
- **Comment and Uncomment** - You can use the keyboard shortcut (Ctrl+Q) to comment and (Ctrl+Shift+Q) to uncomment the selected block of RAPID code. These commands are also available in the context menu.
- **Undo and redo** - Standard commands for undo and redo operations are supported.
- **Selection modes** - You can select text by character, row and column.
- **Line numbers** - Line numbers for the RAPID code lines are displayed in the left margin of the editor.
- **Keyboard shortcuts** - For keyboard shortcuts in the RAPID Editor.

Compare

Use the compare function to compare between folders, files, controller and editor versions of files. Unlike common text editors, the comparison function in RobotStudio is designed to include RAPID specific advanced filtering wherein you can choose to exclude Backinfo, PERS variables, comments and so on. The following filters are available:

- The **BackInfo** option excludes the timestamp of the *Backinfo.txt* file during comparison.
- The **PERS variables** option excludes changes in PERS variables during comparison. A persistent variable retains the last assigned value during restart by writing the value in the RAPID code. Hence the results of comparison can still show differences even though there are no changes in code. Select this option to narrow down comparison to view code changes alone.
- The **Comments**,**Character case** and **White space** options exclude changes in them during comparison.

4.2 RAPID editor intellisense

Overview

RAPID editor provides various code editing features and commands for performing actions on the code.

Syntax highlighting

Text is highlighted in different colors depending on their token classification (such as keyword, identifier and so on). You can configure these colors in the *File* tab, under *Options:Robotics:Text Editor*. In addition to token classification, the editor also shows different colors for built-in and installed identifiers (such as MoveL) and also for identifiers declared in user code.

Quick-Info tooltips

When you hover the mouse pointer over a symbol (such as a data declaration or procedure call), a tooltip is displayed describing the symbol. For many built-in symbols (such as MoveJ) a short description is also displayed. For symbols corresponding to a data declaration, the current value is also displayed.

Auto-completion

After typing or completing a procedure call (such as MoveJ), pressing the Tab key will fill in all required parameters. Note that this is only available for certain built-in procedures, such as those listed in the *Insert Instruction* menu.

Argument information

While typing in procedure calls and function calls, tooltips showing argument information are displayed.

Error highlighting

Red squiggly lines appear under errors in the code. All syntax errors and a subset of semantic errors are indicated in this manner.

Quick Find

Enter the search string in the **Quick find** box and then press Enter or F3. If an instance is found, it is highlighted. Press F3 again to search for the next instance.

Go to line

Enter a line number in the **Go to line** box and press Enter. The cursor moves to the corresponding line in the RAPID editor. When multiple tasks are open in the editor, and if you select the procedure name in the list box, the cursor moves to the task that contains the specified name.

Jump To

The Jump To list has an item for each routine and data declaration in the program module. Click an item to move to its location in the code.

Continues on next page

Find or Replace

Click **Find/Replace** to open the **Find/Replace** dialog. This dialog provides standard find/replace functionality and the following options.

- Enter the string to search in the **Find what** list.
- In the **Look in** list, select the option to specify the location to look for. The various options are **Current Document**, **Current System**, **Current Task**, or a folder in your PC (you can browse to a folder to specify it).

The **Search Results** window displays the results of the find operation. Double-click a search result to go to the corresponding instance in the RAPID editor. If the instance is from a module which is not in the RAPID editor, then the module opens in the editor.

Go To Definition

The **Go To Definition** command is enabled for an identifier in the RAPID Editor context menu if the source code for the corresponding symbol definition is available.

Click **Go To Definition** to move the cursor to (and select) the corresponding symbol definition. This action detects symbol definitions such as routine declarations, data declarations and record definitions.

Navigate forward/backward

Use the **Navigate Backward** button to move to previous locations of the RAPID modules being edited, and the **Navigate Forward** button to return to more recent locations.

Find Unused References

Click **Find unused references in Task** to see all data declarations in the task of the active module document that are not used anywhere. The results are shown in the *Search Results* window. Click **Find unused references in Module** to see unused data declarations in the current module.

Find All References

The **Find All References** command is enabled for identifiers in the editor code.

For a given identifier, click **Find All References** to search through the entire task for uses of the same identifier (including its definition). Note that this is not just a string search. It takes RAPID scoping rules into account. For PERS and syncident data, this function searches the other tasks for a matching global symbol and return the uses of those.

4.3 Manage RAPID modules and programs

Overview

RobotStudio allows editing of standalone modules, that are in memory of the controller or that are available in the disk. When connected to *virtual* or *robot controller*, these standalone files can be edited in the *Home* folder. RAPID code of the controller is structured into *modules*. A module contains several routines of type procedure, function or trap. Modules are of two types system and program.

System module contains code related to robot installation, such as, surrounding equipment, calibration equipment, feeders and service routines. Program module contains RAPID code related to a particular process or parts that the robots are working on. Program modules of a particular task constitute a RAPID program which is handled as a unit. A program, when saved to disk saves each module as independent to each other along with a header file (*pgf) containing references of these modules.

Creating a standalone RAPID module

1 On the **File** tab, click **File**.
2 Click **New**, then under **Files**, click **RAPID Module File**.

  Three types of RAPID *modules* are available, blank, main and system.
3 Under **RAPID Module File**, click the required option, a module with the default name *New Module* opens in the RAPID editor.

Creating a RAPID program from virtual controller

Synchronizing the *station* with virtual controller creates the RAPID program in the RAPID editor. Synchronizing ensures that the RAPID program in the virtual controller corresponds to the programs in RobotStudio. You can synchronize both from RobotStudio to the *virtual controller* and vise versa.

In a RobotStudio station, robot *positions* and movements are defined by *targets* and move instructions in *paths*. These correspond to data declarations and RAPID instructions in the *modules* of the RAPID program. By synchronizing the station to the virtual controller, you create RAPID code out of the data in the station. By synchronizing the virtual controller to the station, you create paths and targets out of the RAPID program in the virtual controller.

Synchronizing the station to the virtual controller updates the RAPID program of the virtual controller with the latest changes in the station. This is useful to do before performing a simulation, saving a program to files on the PC or copying or loading RobotWare systems.

1 On the **RAPID** tab, in the **Access** group, click the arrow next to the **Synchronize** icon, and then click **Synchronize to RAPID**.
2 Select the elements to be synchronized from the list.
3 Click **OK**.

  The message **Synchronization to RAPID completed** is displayed in the Output window.

Continues on next page

> **Note**
>
> This function is also present in the **Controller** group on the **Home** tab.

Loading a RAPID program

You can load a RAPID program to a robot controller either from the PC disk or from the controller disk. For a *virtual controller* you can only load a program from the PC disk.

1. On the **RAPID** tab, in the **Controller** group, click **Program** icon and then select **Load Program** to load a RAPID program from the virtual controller and select **Load Program from Controller** to load a RAPID program from the controller.

   Alternatively, in the **Controller** browser, right-click the active *task* under the *station*, and click **Load Program**.
2. In the **Open** dialog box that appears, browse to the location of the program to be loaded to your station and click **Open**.

Saving a program

Save a RAPID program either in the *virtual controller* or in the controller disk. You need write access for saving a program.

1. On the **RAPID** tab, in the **Controller** group, click **Program** icon and then click **Save Program As** to save the RAPID program in the system and click **Save Program to Controller** to save a program in the controller disk.

   Alternatively, in the **Controller** browser, right-click the active *task* under the *station*, and select **Save Program As**.
2. In the *Save As* dialog box that appears, browse to the location where you want to save your program, and click **Save**.

Renaming a program

Write access is required for renaming a program.

1. On the **RAPID** tab, in the **Controller** group, click **Program** icon and then click **Rename Program**.

   Alternatively, in the **Controller** browser, right-click the active *task* under the *station*, and select **Rename Program**.
2. In the *Rename* dialog box that appears, enter a new name for your program, and click **Ok**.

Deleting a program

Write access is required for deleting a program.

1. On the **RAPID** tab, in the **Controller** group, click **Program** and select **Delete Program**.

   A confirmation dialog is displayed.
2. Click **Yes**.

   The selected program is deleted.

Continues on next page

To delete the entire program under a*task* in a *station*, in the **Controller** group, click **Program** and then click **Delete Program**.

Alternatively, in the **Controller** browser, right-click the task under the station and then click **Delete Program**.

4.4 Adding code snippets

Adding code snippets

Code Snippets are pieces of code which you can insert into the RAPID Editor. To view and select a code snippet, in the **Insert** group, click **Snippet**.

The list which appears show two kinds of code snippets:

- Predefined code snippets
- User defined code snippets

The following are the predefined code snippets in RobotStudio:

- Array of num, 2x2x4
- Array of num, 2x4
- Array of num, 2x4x2
- Array of num, 4x2
- Module header
- Procedure with parameters
- Procedure with error handler
- Robtarget declaration
- Tooldata declaration
- Workobject declaration

You can also create your own code snippets or save a section of existing code from the RAPID editor as a code snippet. Such user created code snippets are also listed along with the predefined snippets. To save a section of existing code, from the RAPID editor, as a code snippet:

1. Select the code that must be saved as a snippet.
2. In the **Insert** group, click the arrow next to the **Snippet** icon, and then click **Save Selection as Snippet**.

   The *Save As* dialog box appears. Specify a name for the snippet and save it. The RobotStudio **.snippet* files are saved in the following folder.
   C:\<Documents and Settings>\<user name>\RobotStudio\Code Snippets

> **Note**
>
> The folder *<Documents and Settings>* may be configured with different names, for example, *Data*. It may also be translated on localized versions of Windows.
>
> Snippets can also be edited in an XML editor such as Microsoft Visual Studio. For information on creating customized code snippets, see *http://msdn.microsoft.com/*.

Inserting a code snippet in a RAPID module

1. Click the **File** tab.
2. Click **Open**, then under **Local**, click **Open** and select the required RAPID file. The file opens in the RAPID editor.
3. Place the cursor in the required line, then, in the **Insert** group click **Snippet**.

Continues on next page

4 From the drop down menu, select the required option, for example, **Procedure with parameters**. The following code snippet gets inserted in the file.

```
PROC myProcedure(\switch doThis | switch doThat, INOUT num
     numRepeats, PERS num dataList{*})
ENDPROC
```

4.5 Inserting instructions from the list

Inserting instructions in a RAPID module

To insert a predefined instruction into the code:

1. Place the cursor at the required point in the RAPID code.
2. In the **Insert** group, click **Instruction**.

   A list of pre-defined instructions gets displayed.

   The instruction is inserted into the code where the cursor is placed. RobotStudio generates and inserts default arguments to the instruction, using similar rules as the FlexPendant.

A large number of instructions that are divided into several categories are available by default. Most common instructions are listed under the default category, *Common*. You can create three personalized lists using the system parameters of the type *Most Common Instruction* in the topic *Man-machine Communication*. The system parameters are described in Technical reference manual - System parameters

Example

1. Click the **File** tab. The backstage view opens.
2. Click **Open**, then under **Local**, click **Open** then select the required RAPID file. The file opens in the RAPID editor.
3. Place the cursor in the required line, then, in the **Insert** group click **Instruction**.
4. From the drop down menu, select the required option, for example, **Common > MoveAbsJ**. The following instruction gets inserted in the file.

```
MoveAbsJ <ARG>\NoEOffs,v1000,z50,tool0\WObj:=wobj0;
```

4.6 Editing standalone files and backups

Starting the RAPID Editor

To open a RAPID module in the RAPID editor, in the **Controller** browser, right-click on a RAPID module, and then click **RAPID Editor**.

The RAPID code of the module opens in the editor window.

Tip

You can view the graphical layout, without closing the editor, by clicking the graphics window tab.

Managing RAPID files

1 In the Files browser, right-click the **File** node and then click **Open**. The **Open File** dialog box appears,

2 In the **Open File** dialog browse and open system module (**.sys*), RAPID modules (**.mod*), and Configuration files (**.cfg*) from the PC or on a network.

Note

The content of the *HOME* folder of the connected *virtual* or *robot controller* is visible in the **Controller** browser. Both RAPID and configuration files can be edited as text files.

For virtual controllers, RAPID modules can be *synchronized* to the graphics environment of the *station* using the context menu of the RAPID module file. To synchronize any changes back to the RAPID module, use the command **Synchronize to file**.

3 A RAPID or system module file opens in the RAPID editor. The system parameters file (**.cfg*) opens in a notepad-like editor.

4 Click the **Save** button on the Quick Access Toolbar to save the changes.

Note

The RAPID editor displays syntax errors in standalone RAPID module when the variable declarations exist in another *module*.

Managing system backups

Right-click **Backup** and click **Browse**, to select and open system backups.

The structure of the backup is reflected in the **Files browser** under the *Backups* node. There is one node for each *task* defined in the *virtual controller*. The RAPID modules of each task are shown as its child nodes in the tree view. The editor will find data declared in other *modules* and mark the code as being syntactically and semantically correct.

The contents of the *HOME* folder are displayed in a separate folder. RAPID modules of the *HOME* folder are edited in the standalone mode, that is, which means that

Continues on next page

the editor will not find data declared in other modules. The reason is that the editor cannot know in which context (task) the module should be treated. The *SYSPAR* folder will show the configuration files.

Note

There is no syntax check or intellisense for editing configuration files.

4.7 RAPID Data Editor

RAPID Data Editor overview

The RAPID Data Editor allows direct access to RAPID data values to view and edit.

To open the RAPID Data Editor, on the **RAPID** tab go to the **Controller** browser, right-click a RAPID module, and then click **RAPID Data Editor**. This opens the Data window which shows the data declarations in that particular *module*.

Data declarations are grouped according to their data types. All data declarations belonging to a data type are shown in a table below it. Each row corresponds to a data declaration and shows the contents of the declaration.

Using the RAPID Data Editor

- Editing the values of a row opens the changed value in the RAPID Editor window. The new value is shown in both the Data editor and also the RAPID editor. This means that the changes made in the RAPID data editor are seen in the RAPID editor, and vice-versa.

  Tip

  An asterisk (*) on the window tab indicates unsaved changes.

- Select multiple cells and edit them together.
- Create, edit or delete a data declaration from the RAPID Data Editor.
- To delete a data declaration, select the row and click the **Delete** button beside it.
- To add a new declaration, click **New Declaration** next to the required data type. This adds a new row to the table below it having some default properties and values, which can be edited. However, you cannot add a data type declaration that is not already present in the *module*. In such cases, you must add the declaration manually to the module using the RAPID Editor.
- To view the orientation of *robtargets* in angles, select the **Show quaternions as RPY angles** checkbox in the RAPID Data Editor. *Orientations* can be represented in angles and quaternions, set the default representation in the RobotStudio Options.

Note

The RAPID Data Editor only shows data declarations that contain editable values.

4.8 Creating custom instructions using Instruction Template

Overview

RobotStudio provides certain set of default instructions. To create instructions for custom applications such as paint, dispensing etc, the Instruction Template feature can be used. You can manually define the instruction templates for RAPID motion instructions other than the default instructions such as MoveL, and MoveC. Custom instructions created using the instruction template feature can be saved as template files and shared with other PCs.

The instruction templates can be saved in the *C:\Users\<user name>\Documents\RobotStudio\Instruction Templates* in XML format and reused later. These files can be used in the same way as any other pre-defined XML files that are imported and used for robot controllers with appropriate RobotWare options.

Creating instruction template

1 Create a custom procedure which accepts arguments such as *robtarget*, *tooldata*, and an optional *wobjdata* as move instructions.

The procedure can include additional arguments such as *speeddata* and *zonedata* or other process related parameters.

Example 1

```
PROC MyMoveL(robtarget ToPoint, PERS tooldata Tool,\wobjdata
     Wobj)
MoveL ToPoint,v100,fine,Tool,\Wobj:=wobj0;
ENDPROC
```

Example 2

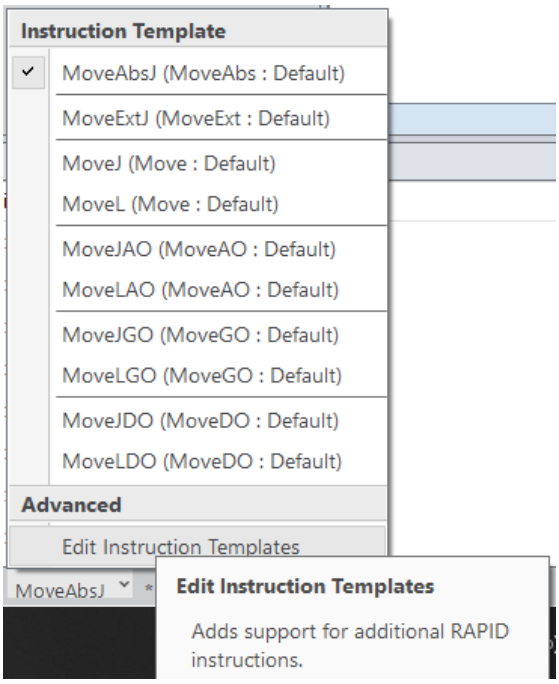
```
PROC MyMoveJ(robtarget ToPoint, PERS tooldata Tool,\wobjdata
     Wobj)
MoveJ ToPoint,v100,fine,Tool,\Wobj:=wobj0;
ENDPROC
```

Note

It is helpful to use the standard naming convention for arguments and instructions. The last letter of the instruction defines the type of the instruction, for example, MoveL implies linear movement, similarly, MoveC and MoveJ implies, circular and joint movements respectively. Following naming conventions helps RobotStudio to interpret the instructions faster, which reduces user input while defining the instruction template. The robtarget must be named *ToPoint*, incase of circular instruction the via point must be named *ViaPoint* according to the naming convention.

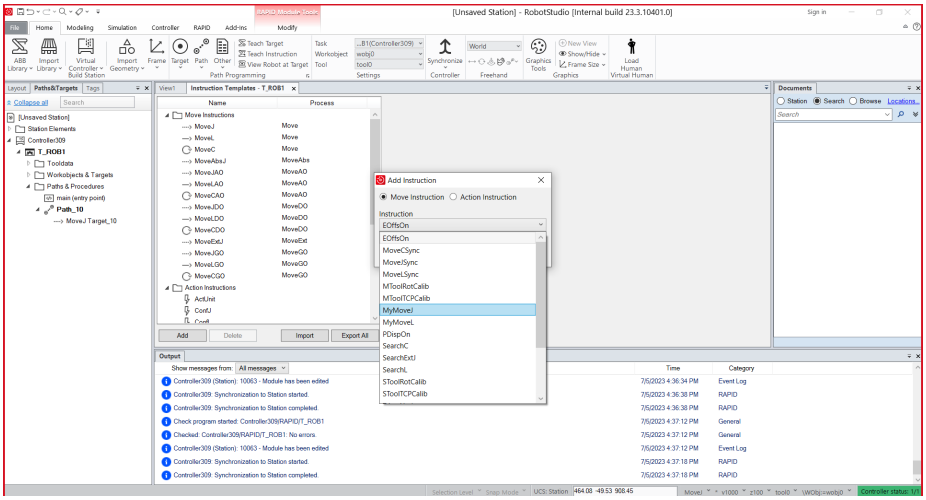
Continues on next page

2 Open Instruction Template window.



xx1900001270

3 In the Instruction Template window, select the instruction and then click the **Add** button.



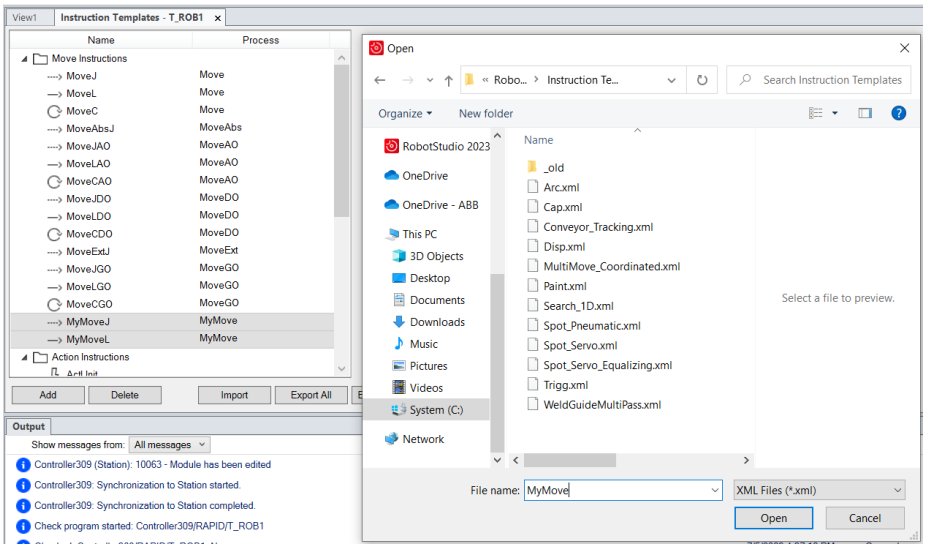
xx1900001271

Repeat this step for all instructions. The instructions are grouped according to the process name (for example, *MyMove*). Adjustments to the *Motion Type*, *Point Mapping* and default parameters are possible at this phase as required.

Motion type defines how RobotStudio traces the path preview. *Point Mapping* defines how RobotStudio interprets robtarget.

Continues on next page

4 Select the new instruction templates and then click **Export Selection** button and then the **Save** button to save the file as XML in the Instruction Template folder.



xx1900001273

The instruction template file must be exported for sharing the file with other users or for using it in the RAPID Path editor. To export the file, use the **Export Selection** button. Import this file to the PC for using it in a station.

4.9 RAPID Path Editor

Overview

Use RAPID Path editor to graphically view and edit programs loaded into a controller. The RAPID program is displayed in a 3D structure in the RAPID Path Editor that allows editing of multiple *robtargets*. The edits made in the graphical RAPID program gets updated in the corresponding RAPID Editor and the RAPID Data Editor simultaneously.

Using the RAPID Path Editor

In the **RAPID** tab, right-click the **Module** or the **Path** to open the **RAPID Path Editor and the Properties** window. Select any target in the RAPID Editor and the corresponding visualization gets displayed in the RAPID Path editor. Any change in the position of a target using the freehand tool in the RAPID Path Editor, gets reflected in the corresponding RAPID program. Also its properties change to the new *position* of the *target*.

- Editing target definition: Use **Properties** tab to edit target definition such as position, orientation and *external axes*.
- Editing move instruction: Edit a move instruction using the **Properties** tab that includes the values for arguments.
- Editing a Jointtarget: Edit the joint values in a Jointtarget using the external Axes in the **Properties** tab. However, a jointtarget has no corresponding graphical representation.

Changes to the graphical representation of the RAPID program gets reflected in the **Properties** tab and vice-versa.

Using the Path Editor tab

Use the **RAPID Path Editor** tab to import a work piece or a *tool* to the graphical representation of the RAPID program.

- Importing a work piece: To visualize a work piece in the Path editor, import it from a *geometry* or library file.
- To set the position of the imported work piece, click **Set position** or choose a *wobj* from the list of *wobj* defined in the program.
- Importing a tool: Add a tool to the visualization by using the tool option. Import a tool from *ABB library* or *user library* (`.rslib`).

Creating custom move instructions in the path editor.

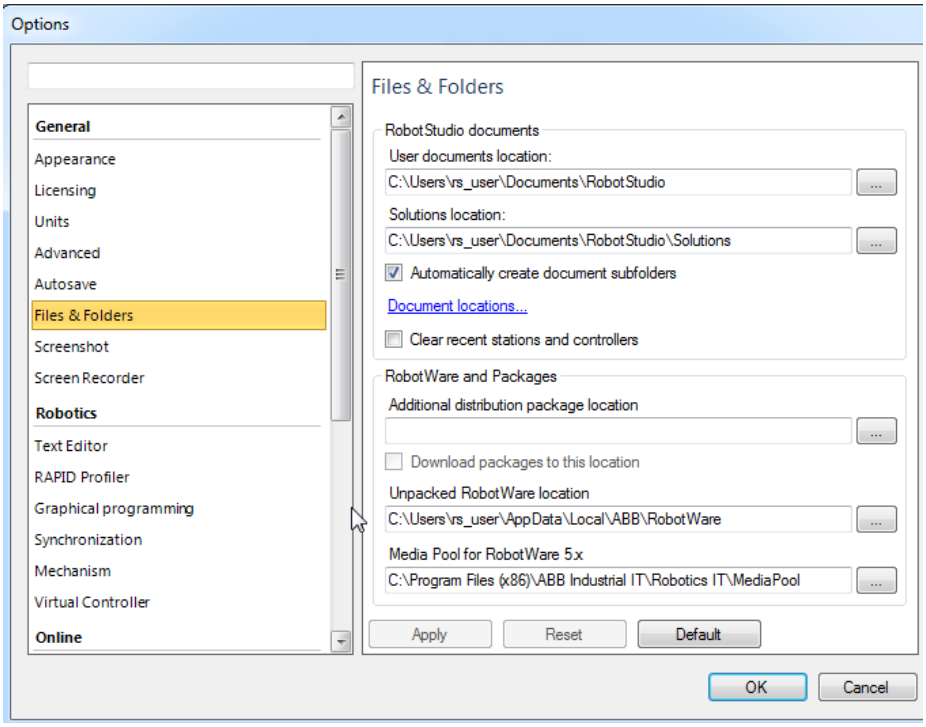
RAPID Path Editor supports generic move instructions. It is possible to view and create such move instructions in the 3D environment of a robot station and in the RAPID Path Editor. The following procedure describes how to view an instruction template file in the RAPID Path Editor.

The required instruction template file must be imported to the host PC before starting this procedure.

1 Copy the required instruction template file to the *C:\Users\<user name>\Documents\RobotStudio\Instruction Templates* folder in the computer.

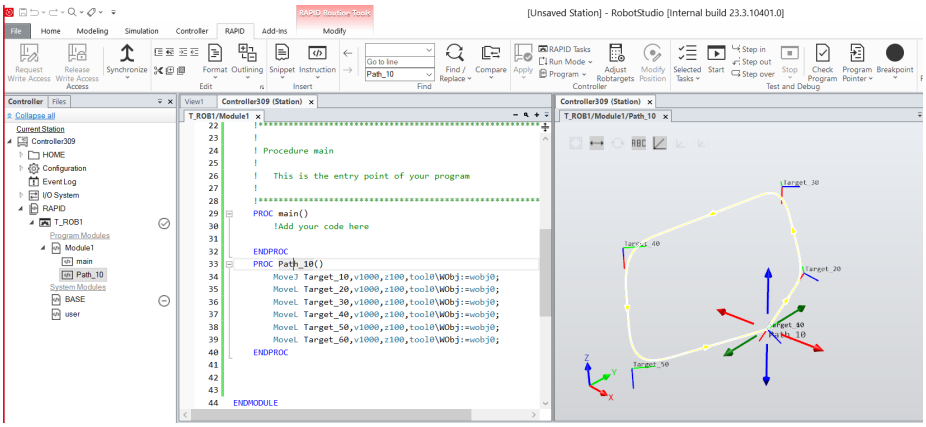
Continues on next page

The search path depends on how the RobotStudio documents folder is configured.



xx1900001274

2 Open the RAPID Path Editor to view and edit the custom move instruction in RobotStudio like a standard move instruction.



xx1900001275

4.10 Applying and verifying the edits

Applying and verifying the edits

To apply the changes made in the editor to the *virtual controller* and also to check the program, In **RAPID** tab, go to the **Controller** group, and click the arrow next to the **Apply** icon, perform any of the following steps:

- To apply only the changes made in the *module*, currently displayed in the editor, click **Apply Changes**.

  Alternatively, click the **Apply** icon
- To apply the changes made in all modified modules, click **Apply All**.

> **Note**
>
> The Apply commands are enabled only for unsaved changes. RobotStudio commits the changes without losing the program pointer. If this is not possible, you will be asked if it is OK to lose the program pointer.

To verify the syntactic and semantic correctness of the modules, in the **Test and Debug** group, click **Check Program**.

Synchronizing to station

Synchronize the *virtual controller* to *station* for applying the changes performed on the RAPID program to the virtual station. This applies to virtual robots only.

> **Note**
>
> **Synchronize to station** is available only for the virtual controller.

1. On the **RAPID** tab, in the **Access** group, click the arrow next to the **Synchronize** icon, and then click **Synchronize to Station**.
2. Select the paths to be synchronized to the station from the list.
3. Click **OK**.

   The message **Synchronization to Station completed** is displayed in the **Output** window.

RAPID can be synchronized from the files of the *HOME* folder as an alternative:

- In the **Controller** browser, under **Home** folder, right-click any file and then click **Synchronize to Station** to synchronize data and procedures of the selected file into the station.
- In the **Controller** browser, under **Home** folder, right-click any file and then click **Synchronize to File** to synchronize data and procedures from the station into the selected file.

  Before selecting this option, you must synchronize the selected file to the station. Using the **Synchronize to File** option can conflict with the synchronize to RAPID option.

Continues on next page

Limitations

- *Robtargets* that are local to a procedure are not supported by **Synchronize to Station**. Only robtargets that are local to a module are supported.
- RobotStudio does not fully support instructions using *Offs* or *RelTool* functions. These are synchronized and will appear in the element browser, but commands such as *View Tool at Target* and *Locate Target* will not work. *Targets* used in the instructions will not be visible in graphics. However, instructions can be programmed and edited using the RAPID Editor and can be simulated using the virtual controller.
- RobotStudio does not support RAPID programs containing arrays of *tooldata*, robtargets and *workobjects*. These programs will not be synchronized to the *station*.
- Workobjects and tooldata that are shared between several *tasks* must be specified in RAPID with its full value for each task when programming offline with RobotStudio. This will trigger a warning *Initial value for PERS not updated* in the controller event log. You may ignore this warning. However, ensure that the RAPID variable definitions are the same in all tasks, otherwise you may get unexpected behavior.

5 Testing and Debugging RAPID

5.1 Debugging a task

Overview

Debugging involves locating and correcting code errors in a *task*. Debugging is part of the code testing and is an integral part of the entire software development. RAPID editor provides several debugging features for testing and perfecting your code. Some of them are introducing *breakpoints* for tracking a particular instruction, using call stack window to monitor program pointer and so on.

Selecting, Starting and stopping a task for RobotWare 6.xx

A task can be selected, started, and stopped from the RAPID editor:

- In the **Controller** group, click the **Selected tasks** button to view the list of tasks. You can select the required foreground task(Normal) or background task (Semistatic/Static) from the list.
- In the **Controller** group, click the **Start** button to start executing the selected task.
- In the **Controller** group, click the **Stop** button to stop executing the selected task.

You can monitor the status of execution in the **Output** window.

Task execution state for RobotWare 5.xx and 6.xx

A task can be activated, started, and stopped from the **Controller** browser with the following limitations:

- Motors must be in *ON* state for the *Start* operation.
- Need *Write access* to the controller and the grants *Execute program* or *Full access*. While performing *Start* or *Stop* operations in Auto, the *Remote Start Stop in Auto* grant is required.
- Normal tasks alone can be activated and deactivated. Background tasks are activated automatically.
- Background tasks of type *Static* and *Semistatic* with the TrustLevel set to NoSafety only can be started and stopped.

  For detailed information about the different TrustLevel values, see the *Technical reference manual - system parameters*.

The following table shows cases where task execution state cannot be changed.

Activating, starting and stopping tasks in RobotWare 5.xx

To activate a task, right-click the task in the **Controller** browser and then turn on the **Active** command. If the prerequisites are met, you can operate the task, such as start and stop the task, move the Program Pointer to main and set the run mode.

Continues on next page

To start a task, right-click the task in the **Controller** browser and then click **Start Task**. You can start Normal tasks, but you can only start a *Static* or *Semistatic* task if the TrustLevel is set to NoSafety.

> **CAUTION**
>
> When starting a task, the manipulator axes may move very quickly and sometimes in unexpected ways! Make sure no personnel is near the manipulator arm!

To stop a task, right-click the task in the Controller browser and then click **Stop Task**. You can stop Normal tasks, but to stop a *Static* or *Semistatic* task the TrustLevel must be set to NoSafety. This method of starting and stopping tasks is applicable only for RobotWare version **5.15.xx** and below.

5.2 Understanding program pointer

Overview

RAPID instructions are executed in the controller for carrying out various operations. These instructions are grouped into various tasks and these tasks are further grouped into *motion* and *non-motion* tasks. A *non-motion* task affects a logical part of the controller operation such as handling of signals, whereas a motion task controls the movement of a mechanical equipment, for example, a manipulator.

A task is a collection of RAPID instructions that are executed line by line. Every task has a program pointer. During execution, program pointer points to the current line of code in a task. The program pointer indicates the instruction with which the program will start execution or rather the program execution continues from the instruction where the program pointer is located.

The motion pointer indicates the instruction of the *motion* task that the robot is currently executing. A *motion* task has both motion pointer and program pointer. Program pointer is ahead of the motion pointer for preparing the controller for motion execution. A motion pointer is normally located at one or more instructions after the program pointer. This allows the system to execute and calculate the robot path faster before the robot moves. The motion pointer is represented as a small robot icon placed to the left margin in the RAPID Editor.

A task can further be classified into, Normal, *Static* and *Semistatic*. A motion task must always be Normal. But a non-motion task can be Normal, Static or Semistatic. Execution of Static and Semistatic tasks start when the controller is powered on. But a Normal task start only when the **Play** button is pressed. In a Semistatic task, when the controller restarts, the program execution starts from the beginning of the task. But the program execution of a Static task starts from the last position of the program pointer before restart.

How the program pointer helps

During program execution, the program pointer points to the line of code that is currently executing. The function **Follow Program Pointer** keeps the program pointer visible during program execution by automatically scrolling the RAPID editor window according to the movements of the program pointer. To enable the function, in the **Test and Debug** group on the **RAPID** tab, click the arrow next to the program pointer icon and then select **Follow Program Pointer**.

Note

During program execution, you can view the movement of program pointer across all open modules. Hence, keep all required modules open.

The other commands in the Program Pointer menu are:

- Go To Program Pointer – To show the current location of the program pointer in the RAPID editor
- Go To Motion Pointer – To show the current location of the motion pointer in the RAPID editor

Continues on next page

- To set the program pointer at a particular line of code or code segment and then start program execution from that point, use the **Set Program Pointer** options. You can choose from the following options:
  - Set Program Pointer to Main in all tasks
  - Set Program Pointer to Cursor
  - Set Program Pointer to Routine

Maintaining the program pointer

The RAPID code can only be edited when the controller is not running, that is when it is in state *Ready* or *Stopped*. In *Ready* state the program pointer is not set, but in *Stopped* state the program pointer is set to a specific location of the program. For limited changes to the RAPID code of a controller in *Stopped* state, the current location of the program pointer can be maintained. After such an edit, you can resume program execution from where it was without having to reset the program pointer.

Note

If the edit is too large for the program pointer to be maintained then a warning message is displayed to convey this.

The program pointer cannot be maintained, for example, when editing the line of code at which the program pointer is located. Editing that line of code results in resetting the program pointer. In effect, the program will start from the beginning when the controller is started after the edit.

WARNING

After resetting program pointer, when the program execution starts, the robot can move along the shortest path from its current location to the first point of the program.

5.3 Working with RAPID breakpoints

Using breakpoints

Breakpoints are used to stop program execution at a certain point or line of code. Set breakpoints to stop program execution to monitor the state of code variables or to view the call stack. You can set a breakpoint in the source code by clicking in the left margin of a file. When you run this code, execution stops whenever the breakpoint is hit, that is, before the code on that line is executed. Press the **Start** button to resume execution.

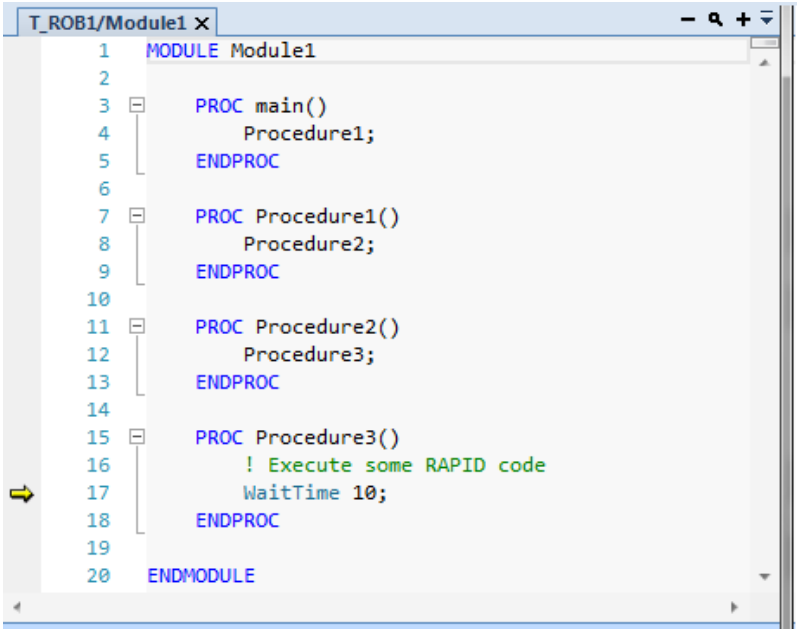
When a RAPID program is run from RobotStudio using the **Play** button, the program execution includes breakpoints and stops execution at breakpoints. Breakpoint functionality is available only in RobotStudio. When the program is run from FlexPendant, breakpoints are ignored. Breakpoints can be viewed in the **RAPID Breakpoints** window. Double-click a certain breakpoint to view the particular line of code.

5.4 Navigate through your program using RAPID Call Stack window

Using the Call Stack window

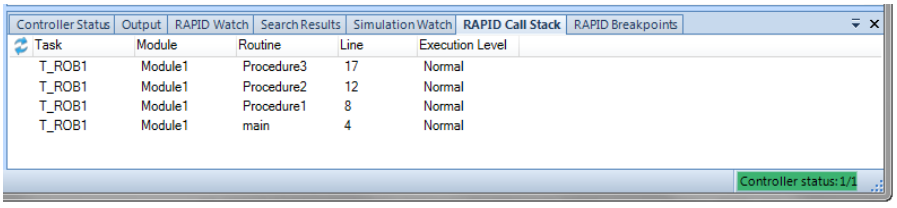
A call stack stores information about the active subroutines of a RAPID program. Use the **Call Stack** window to view the routine call currently on the stack. This feature is useful especially for tracking program pointer during the execution of nested routines.

The following example shows a nested routine where `Procedure 1` calls `Procedure 2` which then calls `Procedure 3` and so on. The **Call Stack** window tracks program pointer and shows the trail of execution.



xx1800000961

The **Call Stack** window displays the name of the task, module, routine and line number that shows the trail of program pointer during program execution. Call Stack will not be refreshed automatically during program execution, and it gets refreshed automatically when the execution completes. To refresh call stack during program execution, click the **Refresh** button.



xx1800000878

5.5 Using RAPID Watch window for debugging your RAPID code

Viewing variables and I/O signals

The RAPID Watch window displays details such as name, value, datatype and system name of selected variables and I/O signals during program execution. You can view and edit RAPID data of variables in the **RAPID watch** window, during program execution and after the controller stops. I/O signals can only be viewed and not edited in the **RAPID watch** window. To view a variable or I/O signal in the **RAPID watch** window, it must be added to the window. In the RAPID editor, right-click the required variable or I/O signal, and then click **Add Watch**.

By default, during program execution, values of variables are automatically refreshed in the **RAPID watch** window every 2 seconds. You can also manually refresh these values. To enable or disable automatic refresh, in the context menu, select or clear the **Auto Refresh** command. To do a manual refresh, in the context menu, click **Refresh** (keyboard shortcut F5).

Note

`CONST` variables cannot be edited. Variables and signals added to the **RAPID watch** window gets removed when RobotStudio closes.

This page is intentionally left blank

6 Simulating programs

6.1 Playing Simulation

6.1.1 Simulation Setup

Overview

The Simulation Setup dialog box is used to perform the following two main tasks.

- Setting up the sequence and *entry point* in the robot program.
- Creating simulation scenarios for different simulated objects.

You can create simulation scenarios containing different simulated objects and connect each scenario with a predefined state to ensure that the correct state is applied to all project objects before running the scenario. If you want to simulate a specific part or segment of the cell where not all simulated objects in the cell are included, you can set up a new scenario and add only the objects needed for simulation.

Prerequisites

To set up a simulation, the following conditions must be met:

- At least one *path* must have been created in the *station*.
- The paths to be simulated must be synchronized to the *virtual controller*.

Setup simulation pane

From this pane, you can perform the combined task of configuring the program sequence and program execution such as *entry point*, and running the execution mode.

The **Setup simulation** pane consists of the following:

Option	Description
Active Simulation scenario	Lists all active station scenarios. • Add: Click to add a new scenario. • Remove: Click to delete the selected scenario. • Rename: Click to rename the selected scenario.
Initial state	Initial state of the simulation.
Manage states	Opens the **Station Logic** pane.
Simulated objects	Displays all objects that can be part of a simulation. Objects that utilize simulation time can be part of a simulation. For example, *virtual controllers* and *Smart Components*. When you create a new scenario, all objects are selected by default.
Virtual time mode	• Time slice: This option makes RobotStudio always use the *time slice mode*. • Free run: This option makes RobotStudio always use the *free run mode*.

Setting up a simulation

1 Click **Simulation Setup** to bring up the **Simulation Setup** pane.

Continues on next page

6.1.1 Simulation Setup
Continued

2 Select the tasks to be active during simulation in the **Select Active Tasks** box.
3 Select the run mode as either **Continuous** or **Single Cycle**.
4 From the **Simulated Objects** list select the task.
5 Select the entry point from the **Entry point** list.
6 Click **Edit** to open the RAPID program where the user can edit the procedure.

Creating simulation scenarios

Verify the productivity of certain solutions, check collision in robot cell, ensure that robot program is free of any motion error.

1 Click **Simulation Setup** to bring up the **Setup Simulation** pane.
2 Under **Active Simulation Scenario** ,
   - Click **Add** to create a new scenario in the **Simulated objects** box.
   - Click **Remove** to delete the selected scenario from the **Simulated objects** box.
3 Select a saved state for the scenario from the **Initial state** list.

6.1.2 Simulation Control

Running a simulation

1 In the **Simulation Control** group,

Click...	to...
Play/Resume	start and resume the simulation. • The **Pause** button is enabled once you start the simulation. • The **Play** button is changed to **Resume** once you pause the simulation. • Click **Resume** to resume the simulation.
Play and select **Record to Viewer**	Start the simulation and to record it to a *Export Viewer*. The **Save As** dialog box appears where the simulation is saved.
Pause/Step	Pause and step the simulation. • The **Pause** button is changed to **Step** once you start the simulation. • Click **Step** to run the simulation in steps. You can set the simulation time step.
Reset	Reset the simulation to its initial state.

Note

When running a simulation in time slice mode, all breakpoints set in the RAPID editor windows will be deactivated temporarily.

Note

During the simulation play, if a user does not want more objects to be added to the conveyor, in **Layout** browser, expand the conveyor node, then right-click the object source and clear the **Enabled** option.

Resetting simulation

1 In the **Simulation Control** group, click **Reset** to reset the simulation.

2 Click **Reset** and select **Save Current state** to store states of objects and *virtual controllers* to be used in a simulation scenario.

3 Click **Reset** and select **Manage states** to start *Station Logic*.

6.2 Detecting Collision

Overview

With RobotStudio you can detect and log collisions between objects in the *station*. The basic concepts of *collision detection* are explained below.

Collision sets

A *collision set* contains two groups, *Objects A* and *Objects B*, in which you place the objects to detect any collisions between them. When any object in *Objects A* collides with any object in *Objects B*, the collision is displayed in the graphical view and logged in the **Output** window. You can have several collision sets in the station, but each collision set can only contain two groups.

A common use of collision sets is to create one collision set for each robot in the *station*. For each collision set you then put the robot and its tool in one group and all objects you do not want it to collide with in the other. If a robot has several tools, or holds other objects, you can either add these to the robot's group as well or create specific collision sets for these setups.

Each collision set can be activated and deactivated separately.

Collisions and near-misses

In addition to collisions, the collision detection can also watch for *near-misses*, which is when an object in *Objects A* comes within a specified distance from an object in *Objects B*.

Recommendations for collision detection

In general, the following principles are recommended to facilitate *collision detection*:

- Use as small *collision sets* as possible, split large parts and collect only relevant parts in the collision sets.
- Enable coarse detail level while importing *geometry*.
- Limit the use of *near-miss*.
- Enable last collision detection, if the results are acceptable.

To customize the collision settings, on the **File** tab, click **Options** and then select **Options:Simulation:Collision**.

Options:Simulation:Collision

Perform collision detection	Select if collision detection is to be performed during simulation or always. Default value: always.
Pause/stop simulation at collision	Select this check box if you want the simulation to stop at a collision or at a near miss. Default value: cleared.
Log collisions to Output window	Select this check box if you want the collisions to be logged to the output window. Default value: selected.
Log collisions to file:	Select this check box if you want to log the collisions to a file. Browse for the file to log in by clicking the browse button. Default value: cleared.

Continues on next page

Enable fast collision detection	Select this check box to enhance the performance by detecting collisions between geometrical bounding boxes instead of geometrical triangles. This might result in falsely reported collisions, since the triangles are the true geometry and the bounding boxes always are larger. All true collisions will, however, be reported. The larger the object, the greater the number of false collisions that are likely to be detected.
View	Click this button to open the log file specified in the file box in Notepad.
Clear	Click this button to delete the log file specified in the file box.
...	Click this button to browse for the file in which you want to log the collisions.

Creating a collision set

1 Click **Create Collision Set** to create a collision set in the **Layout** browser.

2 Expand the collision set and then drag one of the objects to the **ObjectsA** node to check for collisions.

If you have several objects you want to check for collisions with objects in the **ObjectsB** node, for example, the tool and the robot, drag all of them to the **ObjectsA** node.

3 Drag the objects to the **ObjectsB** node to check for collisions.

If you have several objects you want to check for collisions with objects in the **ObjectsA** node, for example, the work piece and the fixture, drag all of them to he **ObjectsB** node.

Tip

Selecting a collision set or one of its groups (*Objects A* or *Objects B*) highlights the corresponding objects in the graphical window and the browser. Use this feature to quickly check what objects have been added to a collision set or to one of its groups.

Results of creating a collision set

After you have created a *collision set*, RobotStudio will check the positions of all objects and detect when any object in *ObjectsA* collides with any object in *ObjectsB*. Activation of detection and display of collisions depend on how the collision detection is set up.

If the collision set is active, RobotStudio will check the positions of the objects in the groups, and indicate any collision between them according to the current color settings.

Collision detection

Collision detection checks whether robots or other moving parts collide with equipment in the *station*. In complex stations, you can use several *collision sets* for detecting collisions between several groups of objects.

After *collision detection* has been set up, it does not need to be started, but automatically detects collisions according to the setup.

Continues on next page

Setting the objects for collision detection

To set the objects for *collision detection*, follow these steps:

1. Make sure that the objects for collision detection are placed correctly in collision sets.
2. Make sure that the *collision set* for the objects is activated, which is indicated by an icon in the **Layout** browser:

Icon	Description
 xx050033	Active. Collisions between objects in this set will be detected.
 xx050007	Not active. Collisions between objects in this set will not be detected.

   To activate or deactivate collision sets, continue with the following steps:
3. Right-click the collision set to change and then click **Modify Collision set** to bring up a dialog box.
4. Select or clear the **Active** check box and then click **Apply**.

Setting near-miss detection

Near-misses occur when objects in collision sets are close to colliding. Each collision set has its own near-miss settings. For setting near-miss detection, follow these steps:

1. In the **Layout** browser, right-click the collision set to change and then click **Modify Collision set** to bring up a dialog box.
2. In the **Near miss** box, specify the maximum distance between the objects to be considered a near-miss and then click **Apply**.

Setting logging options

In addition to the graphical display of collisions, you can also log the collisions to the **Output** window or a separate log file:

1. On the **File** menu, click **Options**, under **Simulation**, click **Collision**.
2. On the **Navigation** pane to the left, select **Simulation: Collision**.
3. On the *Collision* page to the right, select **Log collisions to Output window** check box.

   The collision log is displayed in the **Output** window.
4. On the *Collision* page to the right, select **Log collisions to file** check box and enter the name and *path* to the log file in the box.

   A separate file for logging collisions is created below the check box.

Modifying a collision set

To modify a collision set, follow these steps:

1. Right-click the collision set and then select **Modify Collision set** from the context menu. The **Modify Collision set** dialog box opens.

Continues on next page

2 Select or enter the required values in various fields provided in the dialog box.
3 Click **Apply**.

The **Modify Collision set** dialog box provides the following options:

Options	Description
Active	Collisions between objects in this set will be detected.
Near miss(mm)	Specifies the maximum distance between the objects to be considered a near miss.
Higlight colliding	Lets the user select the colliding object (part, body, or surface) that must be highlighted when two objects collide. It also creates a temporary markup at the point of collision or near miss.
Collision color	Displays the collision in the selected color.
Near miss color	Displays the near miss in the selected color.
Show markup at collision	Shows markup at collision or near miss.
Detect collisions between invisible objects	Detect collisions even if the objects are invisible.

6.3 Collision Avoidance

Overview

The *Collision Avoidance* function monitors the *geometries* of the robot and its *work envelope* and stops the robot from a possible collision. The static geometry surrounding the robot can also be included in the configuration. This is useful where object positions are dynamically created during runtime by cameras or sensors. The predicted collision can be visualized in the RobotStudio Online Monitor. Collision Avoidance is active during jogging and program execution.

The Collision Prediction supports convex geometries such as points, line segments, and convex polygons. Non-convex objects must be split into smaller parts that can be approximated. The Convex Hull has two parameters for controlling the complexity of the collision model, **Max outside tolerance** and **Max inside tolerance**. The **Max outside tolerance** allows inclusion of a bigger approximated object than the original geometry. The **Max inside tolerance** allows the approximated object to be smaller than the original geometry.

Note

A premium license of RobotStudio is required to load a geometrical object of type **.SAT*. The corresponding CAD converter option is required for other formats. Only polygon models can be loaded in the Basic version.

CAUTION

Collision Avoidance shall not be used for safety of personnel.

Activating Collision Avoidance

This feature can be activated from the **Controller** tab.

- In the **Controller** tab, in the **Configuration** group, click **Collision Avoidance** and select **Activate Collision Avoidance**.
- Alternatively, in the **Controller** browser, right-click any controller and from the context menu, click **Collision Avoidance** and select **Activate Collision Avoidance**.

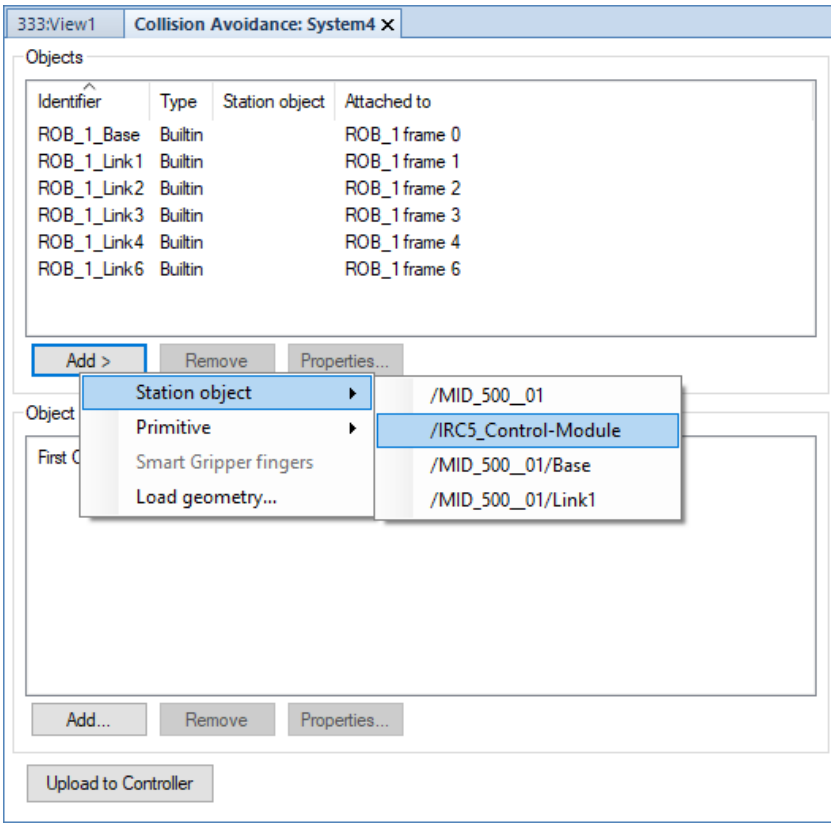
Configuring collision avoidance

1. In the **Configuration** group, select **Collision Avoidance** > **Configure**.

   The **Collision Avoidance** window appears.
2. Under **Objects** group, click **Add**, and select **Station object**, **Primitive**, or **Load geometry...** from the drop-down list.

Continues on next page

This option allows you to create collision models for predicting collision.



xx1800002593

Select the option	To
Station object	Add an existing object or modify its properties
Primitive	Add an object and modify its properties
Smart Gripper fingers	Add smart gripper fingers (only applicable for YuMi)
Load geometry	Add a CAD geometry and modify its properties

3 The **Collision Object Properties** dialog box opens, set and modify the **Properties**, **Convex Hull**, and **Position** of the object.

4 Under **Properties**, additional I/O features can be selected and configured.

Activation signal	Any collision object can be configured with an activation signal that controls whether the object is *Active* or not. This signal is useful while modelling several tools where only one tool is *Active* at a time. It can also be useful for modelling objects that may be present in the robot cell, such as a pallet. The signal must correspond to a digital input.
Stop-active signal	Any non-moving collision object can be configured with a stop-active signal. This signal determines whether the stopping functionality of the zone is *Active* or not. If the signal is low, then the robot will not stop when it comes in contact with this zone/collision object. The signal must correspond to a digital input.

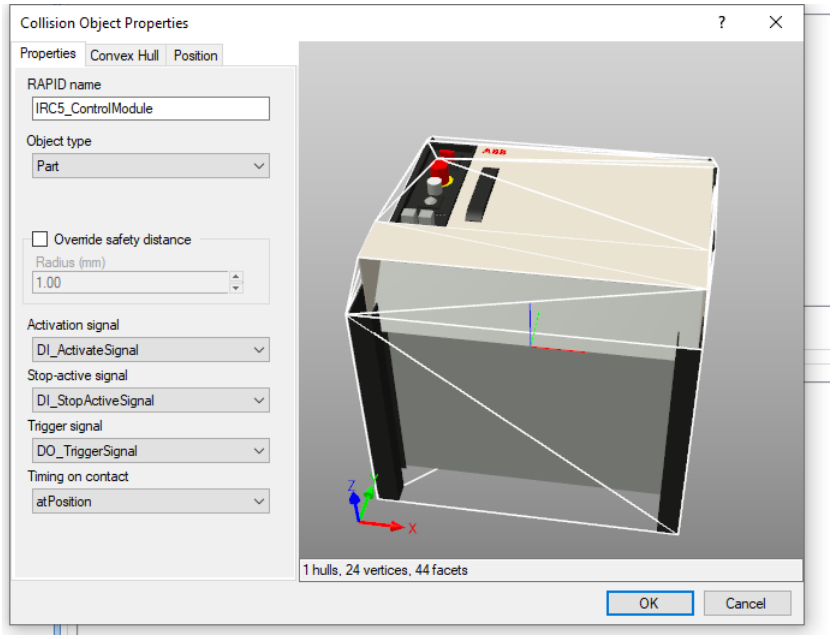
Continues on next page

Trigger signal	Any non-moving collision object can be configured with a trigger signal. The value of the trigger signal indicates the robots that are in contact with the collision object at a time. The value of a trigger signal should be interpreted as a bit pattern, where bit *k* is high if robot *k* is in contact with the collision object. Trigger signals can be used to implement safe workspace sharing between multiple robots. The user can specify the timing behavior of the signal in a number of ways. The signal must map to a group signal in a MultiMove system, otherwise it can be a digital output.

5 Click **OK**, to add the object to the **Objects** list.

Note

A maximum of 10 objects can be added.

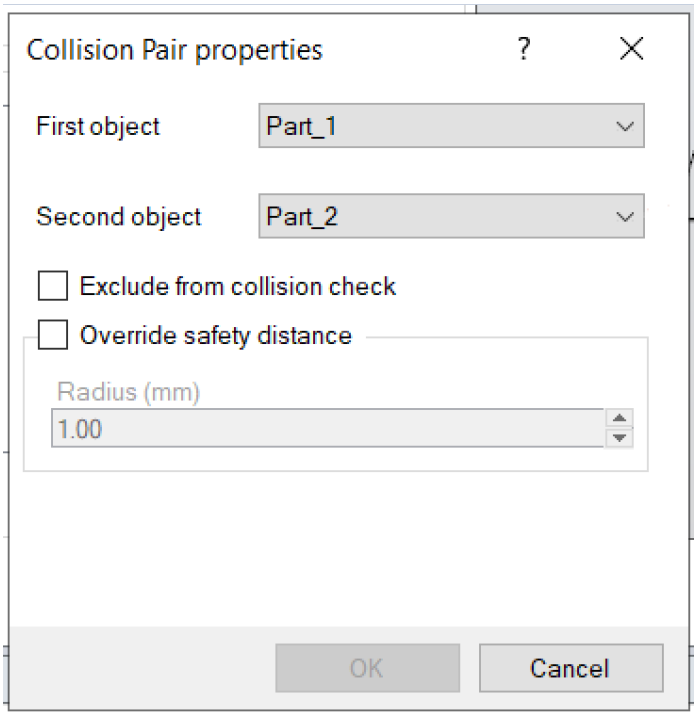


xx1800002592

To easily configure multiple objects, they can be paired.

Continues on next page

6 Under **Object Pairs** group, click **Add**, the **Collision Pair properties** dialog box opens.



xx1800002594

7 Select the objects to be paired for collision avoidance from their respective drop-down lists.

8 Select the **Exclude from collision check** checkbox, to exclude the paired object from the collision check.

9 Select the **Override safety distance** checkbox, to override the preset safety distance.

10 Click **Ok**, to pair the objects and add it to the **Object Pairs** list.

11 Click **Upload to Controller**, to upload the configuration to the robot controller.

Using the **File Transfer** feature a collision avoidance file can be transferred from the **HOME** folder of the *virtual controller* to the robot controller.

Limitations

- For RobotWare 6, *Collision Avoidance* is a function included in the option *Collision Detection*.
- Paint robots, IRB 6620LX, and delta robots are not supported.
- *Collision Avoidance* cannot be used in manual mode together with responsive jogging. The system parameter *Jog Mode* must be changed to *Standard*.
- Only stationary/non-moving objects can be configured with a trigger signal. A trigger signal must correspond to a group signal. Furthermore, each collision object must have its own trigger signal.
- There is no support for applications that do corrections to the path, such as conveyor tracking, WeldGuide, Force Control, SoftMove, SoftAct etc.

Continues on next page

- The *Collision Avoidance* functionality between 2 robots (or more) can only be achieved when using a MultiMove system.

6.4 I/O Simulation

Setting I/O signals using the I/O Simulator

The I/O Simulator is used to view and edit I/O signals that are involved in the simulation. Using the I/O simulator window, you can view and manually set existing signals, create lists during program execution, and simulate or manipulate signals.

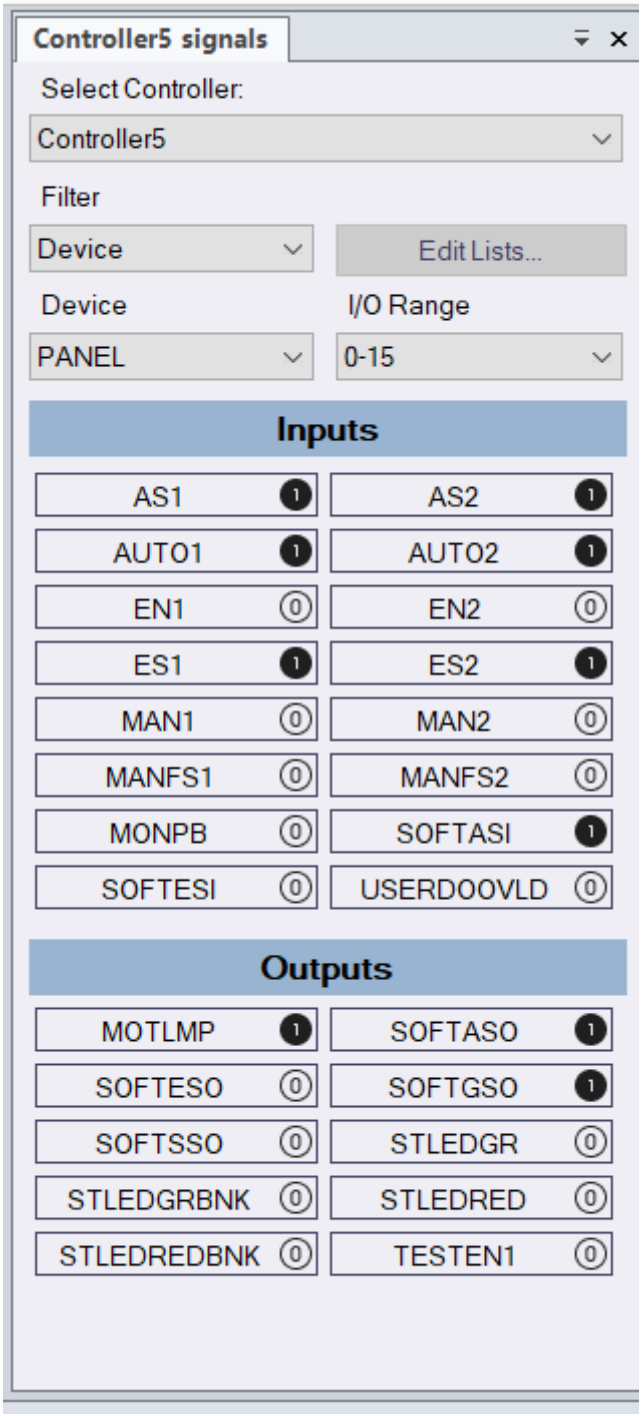
1. In Monitor group, click **I/O Simulator**. This opens the I/O simulator.
2. If the *station* contains several *virtual controllers*, select the appropriate one in the **Select Controller** list.
3. In the **Filter** list and **I/O Range** list, make selections that display the signals to set. Depending on the filter used, you might also set a filter specification.
4. To change the value of a digital I/O signal, click it.

   To change the value of an analog signal, type the new value in the value box.

Continues on next page

The I/O Simulator window

The I/O simulator window displays the signals for one *virtual controller* at a time, in groups of 16 signals. For handling large sets of signals, you can filter which signals to display and also create custom lists with favorite signals for quick access.



xx2000000008

Part	Description
1	**Select System**. Select the required virtual controller to view the signals.

Continues on next page

Part	Description
2	**Filter type**. Select the type of filter to use.
3	**Filter Specification**. Select the filter for limiting the signal display. For example, if *Board* is set as filter type, then select the board to view its signals.
4	**Inputs**. Displays all input signals that pass the applied filter. If more than 16 signals pass, only 16 signals at a time are displayed. Then use the **I/O range** list to select the signals to view.
5	**Outputs** Displays all output signals that pass the applied filter. If more than 16 signals pass, only 16 signals at a time are displayed. Then use the **I/O range** list to select the signals to view.
6	**Edit Lists**. Click this button to create or edit lists of favorite signals.
7	**I/O Range**. When more than 16 signals pass the filter, use this list to select the range of signals to display.

Types of signal filters

Filter	Description
Board	Displays all signals on a specific board. To select a board, use the **Filter Specification** list.
Group	Dispalys a group input or group output signal. To select a group, use the **Filter Specification** list.
User List	Displays all signals in a favorite list. To select a list, use the **Filter Specification** list.
Digital Inputs	Displays all digital input signals.
Digital Outputs	Displays all digital output signals.
Analog Inputs	Displays all analog input signals.
Analog Outputs	Displays all analog output signals.

Signal icons

 value 1	Digital signal with value 1.
 value zero	Digital signal with value 0.
 cross connec	The cross in the upper right corner indicates that the signals are a cross-connection.
 inverted	The -1 in the upper right corner indicates that the signal is inverted.
 value box	Value box for groups or analog signals.

6.5 Simulation time measurement

Stopwatch for measuring process time

The Stopwatch feature is used for measuring the time taken between two trigger points in a simulation, and also for the simulation as a whole. The two trigger points are called the Start Trigger and the End Trigger. When a stopwatch is setup, the timer starts when the Start Trigger occurs, and stops when the End Trigger occurs.

The status bar shows the simulation time during the simulation.

Setting up a Stopwatch

1. On the **Simulation** tab, in the **Monitor** group, click **Stopwatch**.

   The Stopwatch settings dialog appears.
2. Click Add button, and specify a **Name** for the stopwatch.
3. Select a **Start Trigger** and an **End Trigger** for the stopwatch.

   The following parameters are listed for selection as triggers:
   - Simulation Start
   - Simulation Stop
   - Target Changed

     Additionally, specify the mechanical Unit and the target.
   - I/O Value

     Additionally, specify the source *mechanical unit* from where the signal comes, the type of I/O signal and the value of the signal.

6.6 Understanding TCP trace

Overview

TCP trace displays the movement of the robot in a simulation, it traces the movement of the TCP. This data provides an overview of certain parameters. A typical example is using TCP trace to monitor TCP speed. The TCP trace can be colored by the speed of the robot to get the qualitative overview of the variation in speed throughout robot motion. This data can be used to find issues in speed calculations and for optimizing the program.

Tracing continuous signals

1. On the **Simulation** tab, in the **Monitor** group, click **TCP Trace**. The **TCP trace** browser opens.
2. Select the robot form the **Robot** drop down option.
3. Select **Enable TCP Trace** check box to enable tracing.
4. Select **Color by signal** check box and then click **..** button. The **Select Signal** dialog opens.
5. Open the **Mechanical Units** node and then select **Speed In Current Wobj** in the **TCP** node and click **OK**.
6. Select the **Use color scale** box, and enter the values in the **From** and **To** boxes.

   Note

   The **Use color scale** defines how the trace shall be colored. As the signal changes between the values defined in the **From** and **To** boxes, the color of the trace also varies according to the color scale.

7. In the **Simulation Control** group, click **Play** to view the color of the trace when the speed signal is used in the simulation.

Tracing discrete signals

1. In the **TCP Trace** browser, select **Enable TCP Trace** check box to enable tracing.
2. Select **Color by signal** check box and then click **..** button. The **Select Signal** dialog opens.
3. Open the **I/O System** node and then select **DRV1K1** in the **DRV_1** node and click **OK**.
4. Select the **Use secondary color:** box and then select high/low in the **when signal is** drop-down.

   Note

   The color assigned to the trace gets displayed when the signal value meets the specified conditions.

Continues on next page

5 In the **Simulation Control** group, click **Play** to view the trace with the change in signal strength.

Visualizing events along trace

1 In the **TCP Trace** browser, select **Show events** check box, and then click **Select events** to open the list of events.
2 In **Select events** dialog, select the events that must be monitored. Click **OK**.
3 In the **Simulation Control** group, click **Play**. The selected events gets displayed along trace as *markups* during simulation.

Visualizing TCP stop positions along the path

TCP stop position indicates the final stopping position of the TCP when a *category 0 stop* or *category 1 stop* occurs.

1 In the **TCP Trace** browser, select **Enable TCP Trace** and **Show stop position** check boxes, and then select category 0 stop or category 1 stop and color.
2 In the **Simulation Control** group, click **Play**, the category 0 or category 1 stop positions are displayed along the path.

Here the TCP trace gets marked in the primary color and the TCP stop positions in the selected color. At regular intervals along the path, programmed TCP positions are connected to the corresponding TCP stop positions with straight lines.

WARNING

The measurement and calculation of overall stopping performance for a robot must be tested with its correct load, speed, and tools, in its actual environment, before the robot is taken into production, see *ISO 13855:2010*.

7 Advanced RobotStudio simulations

7.1 Understanding SmartComponents

7.1.1 Smart Component

Overview

Smart Components are RobotStudio objects with built-in properties and logic for simulating components that are not part of the *virtual controller*. RobotStudio, by default, offers a set of Base Smart Components for basic motion, signal logic, arithmetic, parametric modeling, sensors and so on.

Base Smart Components can be used to build user defined Smart Components with more complex properties. Some examples are gripper motion, objects moving on conveyors, logic and so on. Smart Components can be saved as library files for reuse.

Purpose of Smart Components

Smart Components provide a graphical programming interface for creating complex components that can be part of stations and simulations. RobotStudio provides Base Smart Components that are required for all possible simulation scenarios. If the property of a certain component is complex and it cannot be simulated by using the Base Smart Components, then code behind can be used. With code behind, it is possible for a developer to program a .NET assembly within the component to customize the *Smart Component*.

Note

Refer http://developercenter.robotstudio.com/robotstudio for more details.

Accessing the Base Smart Components

The Smart Component Editor allows you to create, edit, and aggregate Smart Components using a graphical user interface. On the **Modelling** tab, in the **Create** group, click **Smart Component** or right click on the smart component present on the context menu and select **Edit Component**. The **Smart Component Editor** window appears.

It is possible to protect a Smart Component from being edited. To create a *protected smart component*, right-click the smart component, and then click **Protected**. You can also optionally specify a password that will be required to unlock the component for edits.

Note

Information about Base Smart Components are listed against the selection in the Smart Component editor.

Continues on next page

7.1.1 Smart Component
Continued

Building blocks of a Smart Component

Properties and Bindings: Properties represent the collection of various parameters that define a *Smart Component*. The property of a Smart Component is significant while designing complex Smart Components from Base Smart Components. Consider the example of the Smart Component, *Line Sensor*, which detects if any object intersects a line between two points.

Properties of *Line Sensor* that are considered in the following example are *Start point, End Point* and *Sensed Part*. Here, the Start and end points set the distance where the *Line Sensor* is in *Active* state. The *Sensed Part* decides which object the *Line Sensor* detects, for example, a box that moves on a belt conveyor.

When the *Line Sensor* senses the box , the property *Sensed Part* detects the box. If you want the box to be picked by another Smart Component, for example, a *Gripper*, the property of the *Line Sensor* must be connected to the *Gripper* such that it is ready to grip the box. This is achieved using property bindings. Property bindings connect the value of one property to the value of another property.

Signals: Signals are the property of the Smart Component that has a value, type and direction (input/output). These signals are analogous to the I/O signals on a robot controller. During simulation, the signal values control the responses of the Smart Component. Connections create a tunnel for information to move from the signal of one component to the a signal of another component.

It is possible to rename Smart Component signals and properties in the **Station Logic Design** tab. To rename a signal, right-click the signal and then click **Properties**, the **Edit** dialog opens, type-in the changes and click **OK**.

Asset: These are other files embedded into a Smart Component for perfecting its execution, such as external data files or 3D models.

State: State refers to the parameter values of various components in the Smart Component at a given time. These parameters are, I/O Signal value, Property value and so on.

Grouping signals and properties of Smart Components

Properties and signals of Smart Components can be organized and combined into groups for a filtered view. This feature is useful when working with complex Smart Components consisting of many properties and signals where similar properties or signals can be categorized into groups.

Use the following procedure to create a group and add properties to the group.

1. In the **Smart Component Editor**, click the **Properties and Bindings** tab and then click **Add Dynamic Property**.
2. In the **Add Dynamic Property** window, enter a valid name in the **Group** box and then click **OK**. A new group gets created in the Smart Component **Properties** browser.
3. To add an existing property to a group, with the property selected, click **Edit** and then in the edit dialog, select the Group and click **OK**.

In the **Smart Component Editor**, click the **Signals and Connectors** tab and then follow the same procedure to create a group for signals and to add signals to groups.

7.1.2 Basic Smart Components

Overview

The base components represent a complete set of basic building block components. They can be used to build user defined Smart Components with more complex behavior.

Signals and Properties

LogicGate

The signal Output is set by the logical operation specified in Operator of the two signals InputAand InputB, with the delay specified in Delay.

Properties	Description
Operator	The logical operator to use. The following lists the various operators: • AND • OR • XOR • NOT • NOP
Delay	Time to delay the output signal.

Signals	Description
InputA	The first input.
InputB	The second input.
Output	The result of the logic operation.

LogicExpression

Evaluates a logic expression.

Properties	Description
Expression	Expression to evaluate. Supports logic operators AND, OR, NOT, XOR. Input signals are automatically added for other identifiers.

Signals	Description
Result	Contains the result of the evaluation.

LogicMux

Output is set according to: Output = (Input A * NOT Selector) + (Input B * Selector)

Signals	Description
Selector	Set to 0 to select first input, 1 to select second input.
InputA	Specifies the first input.
InputB	Specifies the second input.
Output	Specifies the result of the operation.

Continues on next page

LogicSplit

The LogicSplit takes Input and sets OutputHigh to the same as Input, and OutputLow as the inverse of Input.

PulseHigh sends a pulse when Input is set to high, and PulseLow sends a pulse when Input is set to low.

Signals	Description
Input	Specifies the input signal.
OutputHigh	Goes high (1) when input is 1.
OutputLow	Goes high (1) when input is 0.
PulseHigh	Sends pulse when input is set to high.
PulseLow	Sends pulse when input is set to low.

LogicSRLatch

TheLogicSRLatch has one stable state.

- When Set=1, Output=1 and InvOutput=0
- When Reset=1, Output=0 and InvOutput=1

Signals	Description
Set	Sets the output signal.
Reset	Resets the output signal.
Output	Specifies output signal.
InvOutput	Specifies Inverse output signal.

Converter

Converts between property values and signal values.

Properties	Description
AnalogProperty	Converts to AnalogOutput.
DigitalProperty	Converts to DigitalOutput.
GroupProperty	Converts to GroupOutput.
BooleanProperty	Converts from DigitalInput and to DigitalOutput.

Signals	Description
DigitalInput	Converts to DigitalProperty.
DigitalOutput	Converted from DigitalProperty.
AnalogInput	Converts to AnalogProperty.
AnalogOutput	Converted from AnalogProperty.
GroupInput	Converts to GroupProperty.
GroupOutput	Converted from GroupProperty.

VectorConverter

Converts between Vector3 and X, Y,and Z values.

Properties	Description
X	Specifies the X-value of Vector.

Continues on next page

Properties	Description
Y	Specifies the Y-value of Vector.
Z	Specifies the Z-value of Vector
Vector	Specifies the vector value.

Expression

The Expression consists of numeric literals (including PI), parentheses, mathematical operators +,-,*,/,^ (power) and mathematical functions sin, cos, sqrt, atan, abs. Any other strings are interpreted as variables, which are added as additional properties. The result is displayed in Result.

Signals	Description
Expression	Specifies the expression to evaluate.
Result	Specifies the result of evaluation.

Comparer

The Comparer compares First value with Second value, using Operator. Output is set to 1 if the condition is met.

Properties	Description
ValueA	Specifies the first value.
ValueB	Specifies the second value.
Operator	Specifies the comparison operator. The following lists the various operators: • == • != • >= • < • <=

Signals	Description
Output	Goes high (1) if the comparison evaluates to true.

Counter

Count is increased when the input signal Increase is set, and decreased when the input signal Decrease is set. Count is reset when the input signal Reset is set.

Properties	Description
Count	Specifies the current count.

Signals	Description
Increase	Set to high (1) to increase the count.
Decrease	Set to high (1) to decrease the count.
Reset	Set to high (1) to reset the count to zero.

Continues on next page

7.1.2 Basic Smart Components
Continued

Repeater

Pulses Output signal Count number of times.

Properties	Description
Count	Number of times to pulse Output.

Signals	Description
Execute	Set to high (1) to pulse Output Count times.
Output	Output pulse.

Timer

The Timer pulses the Output signal based on the given interval.

If Repeat is unchecked, one pulse will be triggered after the time specified in Interval. Otherwise, the pulse will be repeated at the interval given by Interval.

Properties	Description
StartTime	Specifies the time to pass before the first pulse.
Interval	Specifies the simulation time between the pulses.
Repeat	Specifies if the signal should be pulsed repeatedly or only once.
Current time	Specifies the current simulation time.

Signals	Description
Active	Set to high (1) to activate the timer.
Output	Goes high (1) and then low (0) at the specified interval.
Reset	Set to high (1) to reset the current time.

StopWatch

The StopWatch measures time during simulation (TotalTime). A new lap can be started by triggering the Lap input signal. LapTime shows the current lap time. The time is only measured when Active is set to 1. The times are reset when the input signal Reset is set.

Properties	Description
TotalTime	Specifies the accumulated time.
LapTime	Specifies the current lap time.
AutoReset	If true, TotalTime and LapTime will be set to 0 when the simulation starts.

Properties	Description
TotalTime	Outputs the accumulated time.
LapTime	Outputs the lap time.
AutoReset	Resets the stopwatch when the simulation is started.

Signals	Description
Active	Set to high (1) to activate the stopwatch.
Reset	Set to high (1) to reset the stopwatch.
Lap	Set to high (1) to start a new lap

Continues on next page

MultiTimer

The Timer pulses digital signals at specified times during simulation.

Properties	Description
Count	Specifies the number of signals.
CurrentTime	Outputs the current time.
Time1	Time when the corresponding output is pulsed.

Signals	Description
Active	Set to high (1) to activate the timer.
Reset	Set to high (1) to reset the current time.
Output1	Goes high (1) and then low (0) at the specified time.

StringFormatter

Formats a string from input properties.

Properties	Description
Format	Format string. Supports variables like {id:type}, where type can be d (double), i (int), s (string), o (object).
Result	Formatted string.

Parametric Primitives

ParametricBox

The ParametricBox generates a box with dimensions specified by length, width, and height.

Properties	Description
SizeX	Specifies the length of the box in the X-axis direction.
SizeY	Specifies the length of the box in the Y-axis direction.
SizeZ	Specifies the length of the box in the Z-axis direction
GeneratedPart	Specifies the generated part.
KeepGeometry	False to remove the geometry data from the generated part. This can make other components such as Source execute faster.

Signals	Description
Update	Set to high (1) to update the generated part.

ParametricCylinder

The ParametricCylinder generates a cylinder with the dimensions given by Radius and Height.

Properties	Description
Radius	Specifies the radius of the cylinder.
Height	Specifies the height of the cylinder.
GeneratedPart	Specifies the generated part.

Continues on next page

Properties	Description
KeepGeometry	False to remove the geometry data from the generated part. This can make other components such as Source execute faster.

Signals	Description
Update	Set to high (1) to update the generated part.

ParametricLine

The ParametricLine generates a line with a given end point or a given length. If either of them is changed, the other one will be updated accordingly.

Properties	Description
EndPoint	Specifies the end point for the line.
Length	Specifies the length of the line.
GeneratedPart	Specifies the generated part.
GeneratedWire	Specifies the generated wire object.
KeepGeometry	False to remove the geometry data from the generated part. This can make other components such as Source execute faster.

Signals	Description
Update	Set to high (1) to update the generated part.

ParametricCircle

The ParametricCircle generates a circle with a given radius.

Properties	Description
Radius	Specifies the radius of the circle.
GeneratedPart	Specifies the generated part.
GeneratedWire	Specifies the generated wire object.
KeepGeometry	False to remove the geometry data from the generated part. This can make other components such as Source execute faster

Signals	Description
Update	Set to high (1) to update the generated part.

LinearExtrusion

The LinearExtrusion extrudes SourceFace or SourceWire along the vector given by Projection.

Properties	Description
SourceFace	Specifies the face to extrude.
SourceWire	Specifies the wire to extrude.
Projection	Specifies the vector to extrude along.
GeneratedPart	Specifies the generated part.

Continues on next page

Properties	Description
KeepGeometry	False to remove the geometry data from the generated part. This can make other components such as Source execute faster.

Signals	Description
Update	Set to high (1) to update the generated part.

LinearRepeater

The LinearRepeater creates a number of copies of Source, with the spacing and direction given by Offset.

Properties	Description
Source	Specifies the object to copy.
Offset	Specifies the distance between copies.
Distance	Specifies the distance between the copies.
Count	Specifies the number of copies to create.

MatrixRepeater

The MatrixRepeater creates a specified number of copies in three dimensions, with the specified spacing of the object in Source.

Properties	Description
Source	Specifies the object to copy.
CountX	Specifies the number of copies in the X-axis direction.
CountY	Specifies the number of copies in the Y-axis direction.
CountZ	Specifies the number of copies in the Z-axis direction.
OffsetX	Specifies the offset between the copies in the X-axis direction.
OffsetY	Specifies the offset between the copies in the Y-axis direction.
OffsetZ	Specifies the offset between the copies in the Z-axis direction.

CircularRepeater

The CircularRepeater creates a number of given copies of Source around the center of the SmartComponent with a given DeltaAngle.

Properties	Description
Source	Specifies the object to copy.
Count	Specifies the number of copies to create.
Radius	Specifies the radius of the circle.
DeltaAngle	Specifies the angle between the copies.

Continues on next page

Sensors

CollisionSensor

The CollisionSensor detects collisions and near miss events between the First object and the Second object. If one of the objects is not specified, the other will be checked against the entire station. When the Active signal is high and a collision or a near miss event occurs and the component is active, the SensorOut signal is set and the parts that participate in the collision or near miss event are reported in the first colliding part and second colliding partof the Property editor.

Properties	Description
Object1	The first object to check for collisions.
Object2	The second object to check for collisions.
NearMiss	Specifies the near miss distance.
Part1	The part of First object that has a collision.
Part2	The part of Second object that has a collision.
CollisionType	• None • Near miss • Collision

Signals	Description
Active	Set to high (1) to activate the sensor.
SensorOut	Goes high (1) when a collision or near miss occurs.

LineSensor

The LineSensor defines a line by the Start, End, and Radius. When an Active signal is high, the sensor detects objects that intersect the line. Intersecting objects are displayed in the ClosestPart property and the point on the intersecting part that is closest to the line sensors start point is displayed in the ClosestPoint property. When intersection occurs the SensorOut output signal is set.

Properties	Description
Start	Specifies the start point.
End	Specifies the end point.
Radius	Specifies the radius.
SensedPart	Specifies the part that intersects the line sensor. If several parts intersect, then the part that is closest to the Start point is listed.
SensedPoint	Specifies the point on the intersecting part, closest to the Start point.
Length	Specifies the distance between the sensor start and end points. The Length property can be used to adjust the length of the sensor after the direction has been specified with the start and end points. When the Length property is changed, the end point is updated accordingly along the direction of the sensor automatically.
Tag	When a tag is specified, the sensor detects only objects with the same tag.

Continues on next page

Signals	Description
Active	Set to 1 to activate the sensor.
SensorOut	Goes high (1) when an object intersects the line.

PlaneSensor

The PlaneSensor defines a plane by Origin, Axis1, and Axis2. When the Active input signal is set the sensor detects objects that intersect this plane. Intersecting objects are displayed in the SensedPart property and when intersection occurs the SensorOut output signal is set.

Properties	Description
Origin	Specifies the origin of the plane.
Axis1	Specifies the first axis of the plane.
Axis2	Specifies the second axis of the plane.
SensedPart	Specifies the part that intersects the PlaneSensor. If several parts intersect, then the one listed first in the Layout browser is selected.
Tag	When a tag is specified, the sensor detects only objects with the same tag.

Signals	Description
Active	Set to high (1) to activate the sensor.
SensorOut	Goes high (1) when an object intersects the plane.

VolumeSensor

The VolumeSensor detects objects that are inside or partly inside a box-shaped volume. The volume is defined by a Corner Point, the Length, Height, and Width of the sides and the orientation angles.

Properties	Description
CornerPoint	Specifies the local origin of the box.
Orientation	Specifies the orientation (Euler ZYX) relative to Reference.
Length	Specifies the length of the box.
Width	Specifies the width of the box.
Height	Specifies the height of the box.
Percentage	The percentage of the volume to react on. Set to 0 to react on all objects.
PartialHit	Allow an object to be sensed if only a part of it is inside the volume sensor.
SensedPart	The last object that either entered or left the volume.
SensedParts	The objects sensed in the volume.
VolumeSensed	The total volume sensed.
Tag	When a tag is specified, the sensor detects only objects with the same tag.

Signals	Description
Active	Set to high (1) to activate the sensor.

Continues on next page

7.1.2 Basic Smart Components
Continued

Signals	Description
SensorOut	Goes high (1) when an object is detected.

PositionSensor

The PositionSensor monitors the position and orientation of an object.

The position and orientation of an object is updated only during the simulation.

Properties	Description
Object	Specifies the object to monitor.
Reference	Specifies the reference coordinate system (Parent or Global).
ReferenceObject	Specifies the reference object, if Reference is set to Object.
Position	Specifies the position of the object relative to Reference.
Orientation	Specifies the orientation (Euler ZYX) relative to Reference.

Signals	Description
SensorOut	Goes high (1) when Position or Orientation have changed.

ClosestObject

The ClosestObject defines a Reference object or a Reference point. When the Execute signal is set, the component finds the ClosestObject, ClosestPart, and the Distance to the reference object, or to the reference point if the reference object is undefined. If RootObject is defined, the search is limited to that object and its descendants. When finished and the corresponding properties are updated the Executed signal is set.

Properties	Description
ReferenceObject	Specifies the object to get the closest object to.
ReferencePoint	Specifies the point to get the closest object to.
RootObject	Specifies the object whose children to search. Empty means entire station.
ClosestObject	Specifies the object closest to Reference object or Reference point.
ClosestPart	Specifies the part closest to Reference object or Reference point.
Distance	Specifies the distance between the Reference object and the Closest object.

Signals	Description
Execute	Set to high (1) to set find the closest object.
Executed	Goes high (1) when the operation is completed.

JointSensor

Monitors mechanism joint values during simulation

Properties	Description
Mechanism	Specifies the mechanism to monitor.

Continues on next page

Signals	Description
Update	Set to high (1) to update the joint values.

GetParent

The GetParent returns the parent object of the input object. The executed signal is triggered if a parent is found.

Properties	Description
Child	Specifies the object to whose parent is to be found.
Parent	Specifies the parent of the child

Note

The **Child** list for **Properties:GetParent** does not show every part or object in the station. However, if you do not find the required part or object in the list, then add it from the browser or graphic window by clicking it.

Actions

Attacher

The Attacher will attach Child to Parent when the Execute signal is set. If the Parent is a mechanism, the Flange to attach to must also be specified. When the input Execute is set, the child object is attached to the parent object. If Mount is checked, the child will also be mounted on the parent, with the Offset and Orientation specified. The output Executed will be set when finished.

Properties	Description
Parent	Specifies the object to attach to.
Flange	Specifies the Index of mechanism flange to attach to.
Child	Specifies the object to attach.
Mount	If true, the object to attach mounts on the attachment parent.
Offset	Specifies the position relative to the attachment parent when using Mount.
Orientation	Specifies the orientation relative to the attachment parent when using Mount.

Signals	Description
Execute	Set to high (1) to create the attachment.
Executed	Goes high (1) when the operation is complete.

Detacher

The Detacher will detach the Child from the object it is attached to when the Execute signal is set. If Keep position is checked, the position will be kept. Otherwise the child is positioned relative to its parent. When finished, the Executed signal will be set.

Properties	Description
Child	Specifies the object to detach.

Continues on next page

Properties	Description
KeepPosition	If false, the attached object is returned to its original position.

Signals	Description
Execute	Set to high (1) to remove the attachment.
Executed	Goes high (1) when the operation is complete.

Source

The Source property of the source component indicates the object that should be cloned when the Execute input signal is received. The parent of the cloned objects is specified by the Parent property and a reference to the cloned object is specified by the Copy property. The output signal Executed signifies that the clone is complete.

Properties	Description
Source	Specifies the object to copy.
Copy	Specifies the copied object.
Parent	Specifies the parent to the copy. If not specified, the copy gets the same parent as the source.
Position	Specifies the position of the copy relative its parent.
Orientation	Specifies the orientation of the copy relative its parent.
Transient	Marks the copy as transient if created during simulation. Such copies are not added to the undo queue and are automatically deleted when the simulation is stopped. This is used to avoid increased memory consumption during simulation.
PhysicsBehavior	Specifies the physics behavior of the copy.

Signals	Description
Execute	Set to high (1) to create the copy.
Executed	Goes high (1) when the operation is complete.

Sink

The Sink deletes the object referenced by the Object property. Deletion happens when the Execute input signal is received. The Executed output signal is set when the deletion is finished.

Properties	Description
Object	Specifies the object to remove.

Signals	Description
Execute	Set to high (1) to remove the object.
Executed	Goes high (1) when the operation is complete.

Continues on next page

Show

When the Execute signal is set, the object referenced in Object appears. When finished, Executed signal will be set.

Properties	Description
Object	Specifies the object to show.

Signals	Description
Execute	Set to True to show the object.
Executed	Sends a pulse when completed.

Hide

When the Execute signal is set, the object referenced in Object will be hidden. When finished, Executed signal will be set.

Properties	Description
Object	Specifies the object to hide.

Signals	Description
Execute	Set to True to hide the object.
Executed	Sends a pulse when completed.

SetParent

Sets the parent of a graphic component.

Properties	Description
Child	The graphic component whose parent is to be set.
Parent	New parent.
KeepTransform	To keep the position and orientation of the child object.

Signals	Description
Execute	Set to high (1) to move the child object to the new parent.

Manipulators

LinearMover

The LinearMover moves the object referenced in the Object property with a speed given by the Speed property in the direction given by the Direction property. The motion starts when the Execute input signal is set and stops when Execute is reset.

Properties	Description
Object	Specifies the object to move.
Direction	Specifies the direction to move the object.
Speed	Specifies the speed of movement.
Reference	Specifies the coordinate system in which values are specified. It can be Global, Local, or Object.
ReferenceObject	Specifies the reference object, if Reference is set to Object.

Continues on next page

Signals	Description
Execute	Set to high (1) to start moving the object.

LinearMover2

The LinearMover moves an object a specified distance.

Properties	Description
Object	Specifies the object to move.
Direction	Specifies the direction to move the object.
Distance	Specifies the distance to move the object.
Duration	Specifies the time for the movement.
Reference	Specifies the coordinate system in which values are specified. It can be Global, Local, or Object.
ReferenceObject	Specifies the Reference object.

Signals	Description
Execute	Set to high (1) to start the movement.
Executing	Goes high (1) during the movement.
Executed	Goes high (1) when the movement is completed.

Rotator

The Rotator rotates the object referenced in the Object property with an angular speed given by the Speed property. The axis of rotation is given by CenterPoint and Axis. The motion starts when the Execute input signal is set and stops when the Execute is reset.

Properties	Description
Object	Specifies the object to rotate.
CenterPoint	Specifies the point to rotate around.
Axis	Specifies the axis of the rotation.
Speed	Specifies the speed of the rotation.
Reference	Specifies the coordinate system in which values are specified. It can be Global, Local, or Object.
ReferenceObject	Specifies the object which are relative to CenterPoint and Axis, if Reference is set to Object.

Signals	Description
Execute	Set to high (1) to start rotating the object.

Rotator2

The Rotator rotates an object a specified angle around an axis.

Properties	Description
Object	Specifies the object to rotate.
CenterPoint	Specifies the point to rotate around.
Axis	Specifies the axis of the rotation.
Angle	Specifies the angle to rotate

Continues on next page

Properties	Description
Duration	Specifies the time for the movement.
Reference	Specifies the coordinate system in which values are specified. It can be Global, Local, or Object.
ReferenceObject	Specifies the Reference object.

Signals	Description
Execute	Set to high (1) to start the movement.
Executing	Goes high (1) during the movement.
Executed	Goes high (1) when the movement is completed.

PoseMover

The PoseMover has a Mechanism, a Pose, and Duration as properties. When the Execute input signal is set the mechanism joint values are moved to the given pose. When the pose is reached the Executed output signal is set.

Properties	Description
Mechanism	Specifies the mechanism to move to a pose.
Pose	Specifies the Index of the pose to move to.
Duration	Specifies the time for the mechanism to move to the pose.

Signals	Description
Execute	Set to True, to start or resume moving the mechanism.
Pause	Pauses the movement.
Cancel	Cancels the movement.
Executed	Pulses high when the mechanism has reached the pose.
Executing	Goes high during the movement.
Paused	Goes high when paused.

JointMover

The JointMover has a Mechanism, a set of Joint Values and a Duration as properties. When the Execute input signal is set the mechanism joint values are moved to the given pose. When the pose is reached the Executed output signal is set. The GetCurrent signal retrieves the current joint values of the mechanism.

Properties	Description
Mechanism	Specifies the mechanism to move to a pose.
Relative	Specifies if J1-Jx are relative to the start values, rather than absolute joint values.
Duration	Specifies the time for the mechanism to move to the pose.

Signals	Description
GetCurrent	Retrieve current joint values.
Execute	Set to True to start moving the mechanism.
Pause	Pauses the movement
Cancel	Cancels the movement

Continues on next page

Signals	Description
Executed	Pulses high when the mechanism has reached the pose.
Executing	Goes high during the movement.
Paused	Goes high when paused.

Positioner

The Positioner takes an Object, Position, and Orientation as properties. When the Execute signal is set the object is repositioned in the given position relative to the Reference. When finished the Executed output is set.

Properties	Description
Object	Specifies the object to position.
Position	Specifies the new position of the object.
Orientation	Specifies the new orientation of the object.
Reference	Specifies the coordinate system in which values are specified. It can be Global, Local, or Object.
ReferenceObject	Specifies the object which are relative to Position and Orientation, if Reference is set to Object.

Signals	Description
Execute	Set to high (1) to set the position.
Executed	Goes high (1) when the operation is completed.

MoveAlongCurve

Moves an object along a geometric curve (using a constant offset).

Properties	Description
Object	Specifies the object to move.
WirePart	Specifies the part that contains wire(s) to move along.
Speed	Specifies the speed.
KeepOrientation	Set to true to keep the orientation of the object.
StartDistance	Where along the curve to start the movement.
ResetPosition	
StartPosition	

Signals	Description
Execute	Set to high (1) to start the movement.
Pause	Set to high (1) to cancel the movement.
Cancel	Goes high (1) during the movement.
Executing	Goes high (1) during the movement.
Paused	Goes high (1) when the movement is paused.
Executed	Goes high (1) when the movement is completed.

Continues on next page

Controller

RapidVariable

Sets or gets the value of a RAPID variable.

Properties	Description
DataType	RAPID datatype of the variable to get or set (bool, num, dnum, string, pos, orient, or pose)
Controller	The virtual controller that contains the variable.
Task	The RAPID task that contains the variable.
Module	The RAPID module in which the variable is defined.
Variable	The name of the RAPID variable
Value	The variable value and the type depends on DataType.

Signals	Description
Get	Set to high (1) to get the value.
Set	Set to high (1) to set the value.
Executed	Is pulsed when the value is updated.

Physics

PhysicsControl

Controls the physics related properties of parts or component groups.

Properties	Description
Object	Specifies the object to control.
Behavior	Determines the behavior of the object in the physics simulation.
Surface Velocity	Sets the surface velocity.

Signals	Description
Enabled	Set to high (1) to enable physics behavior.
SurfaceVelocityEnabled	Set to high (1) to enable surface velocity.

PhysicsJointControl

Controls properties of a physics joint.

Properties	Description
MotorSpeed	Sets the speed of a motorized joint.
Joint	The joint to control.

Signals	Description
MotorEnabled	Enables or disables the joint motor.

Continues on next page

PLC

OpcUaClient

An OPC UA client.

Properties	Description
ServerAddress	The IP address and the port number to the OPC UA server.
UseSecurity	Enables encrypted connection and requires a trusted server certificate.
AutoConnect	Enables automatic connection when loading the station and after a connection failure.
Blocking	When enabled the simulation will wait for the OPC UA communication to complete in each time step.
Authentication	Enables choosing authentication type for connecting to the OPC UA server.

Following signals are available as commands in the context menu.

Commands	Description
Connect	Connect to the OPC UA server.
Disconnect	Disconnect from the OPC UA server.
ChangeUser	Connect to the OPC UA server with a different user.
Configure	Configure the mapping between OPC UA server nodes and RobotStudio signals.
ImportConfiguration	Import the mapping between OPC UA server nodes and RobotStudio signals.
ExportConfiguration	Export the mapping between OPC UA server nodes and RobotStudio signals.

SIMITConnection

Siemens SIMIT connection via Shared Memory.

Properties	Description
SharedMemoryName	SIMIT Shared Memory name.

Following signals are available as commands in the context menu.

Commands	Description
Connect	Connect to SIMIT.
Disconnect	Disconnect from SIMIT.

Virtual Reality

VrHandController

A component that moves with a hand controller in VR

Properties	Description
Hand	Left or right hand.
TrackTip	Selects if the tip or the base of the hand controller is tracked.

Continues on next page

Signals	Description
TriggerDown	Goes to high (1) when the primary trigger is pressed.

VrSession

Adds custom buttons to a VR menu pane, and signals when the user exits or enters VR

Properties	Description
NumCommands	Number of commands to add.

Signals	Description
SessionStarted	Pulsed when the user starts VR.
SessionEnded	Pulsed when the user exits VR.

VrTeleporter

Teleports the VR user to the location of the component

Signals	Description
Execute	Set to high (1) to execute the teleport.

Other

Queue

The Queue represents a FIFO (first in, first out) queue. The object in Back is added to the queue when the signal Enqueue is set. The front object of the queue is shown in Front. The object in Front is removed from the queue when the signal Dequeue is set. If there are more objects in the queue, the next object is shown in Front. All objects in the queue are removed from the queue when the signal Clear is set.

If a transformer component (such as LinearMover) has a queue component as its Object, it will transform the contents of the queue, rather than the queue itself.

Properties	Description
Back	Specifies the object to enqueue.
Front	Specifies the first object in queue.
NumberOfObjects	Specifies the number of objects in the queue.

Signals	Description
Enqueue	Adds the object in Back to the end of the queue.
Dequeue	Removes the object in Front from the queue.
Clear	Removes all objects from the queue.
Delete	Removes the object in Front from the queue and from the station.
DeleteAll	Clears the queue and removes all objects from the station

Continues on next page

7.1.2 Basic Smart Components
Continued

ObjectComparer

Sets a digital signal as a result of an object comparison. Determines if ObjectA is the same as ObjectB.

Properties	Description
ObjectA	Specifies the first object to compare.
ObjectB	Specifies the second object to compare.

Signals	Description
Output	Goes high (1) if the objects are equal.

GraphicSwitch

Switches between two parts, either by clicking on the visible part in the graphics or by setting and resetting the input signal.

Properties	Description
PartHigh	Displayed when the signal is high.
PartLow	Displayed when the signal is low.

Signals	Description
Input	Input signal.
Output	Output signal high (1) when PartHigh is displayed, and low (0) when PartLow is displayed.

Highlighter

The Highlighter temporarily sets the color of the Object to the RGB-values specified in Color. The color is blended with the original color of the objects as defined by Opacity When the signal Active is reset, Object gets its original colors.

Properties	Description
Object	Specifies the object to highlight.
Color	Specifies the RGB-values of the highlight color.
Opacity	Specifies the amount to blend with the object's original color (0-255).

Signals	Description
Active	Set to high (1) to change the color, low (0) to restore the original color.

MoveToViewPoint

Moves to the selected viewpoint in the given time, when the input signal Execute is set. The output signal Executed is set when the operation is completed.

Properties	Description
Viewpoint	Specifies the viewpoint to move to.
Time	Specifies the time to complete the operation.

Signals	Description
Execute	Set to high (1) to start the operation.
Executed	Goes high (1) when the operation is completed.

Continues on next page

Logger

Prints a message in the output window.

Properties	Description
Format	Format string. Supports variables like {id:type}, where type can be d (double), i (int), s (string), o (object)
Message	Formatted message.
Severity	Message severity: 0 (Information), 1 (Warning), 2 (Error).

Signals	Description
Execute	Set to high (1) to print the message.

SoundPlayer

Plays the sound specified by Sound Asset when the input signal Execute is set. The asset must be a .wav file

Properties	Description
SoundAsset	Specifies the sound file that should be played. Must be a .wav file.
Loop	Set to true to loop the sound.

Signals	Description
Execute	Set to high (1) to play the sound.
Executed	Goes high (1) when the sound is played.
Stop	Set to high (1) to stop playing.

Random

Random generates a random number between Min and Max in Value when Execute is triggered.

Properties	Description
Min	Specifies minimum value.
Max	Specifies maximum value.
Value	Specifies a random number between Min and Max.

Signals	Description
Execute	Set to high (1) to generate a new random number.
Executed	Goes high (1) when the operation is completed.

StopSimulation

Stop a running simulation when the input signal Execute is set.

Signals	Description
Execute	Set to high (1) to stop the simulation.

Continues on next page

7.1.2 Basic Smart Components
Continued

TraceTCP

Enables or disables TCP trace for a robot.

Properties	Description
Robot	Specifies the robot to trace.

Signals	Description
Enabled	Set to high (1) to enable TCP trace.
Clear	Set to high (1) to clear the TCP trace.

SimulationEvents

Pulses signals when the simulation starts and stops.

Signals	Description
SimulationStarted	Pulsed when the simulation starts.
SimulationStopped	Pulsed when the simulation stops.

LightControl

Controls a light source.

Properties	Description
Light	Specifies the light source.
Color	Sets the color of the light.
CastShadows	Enables the light to cast shadows.
AmbientIntensity	Sets the ambient intensity of the light.
DiffuseIntensity	Sets the diffuse intensity of the light.
HighlightIntensity	Sets the specular intensity of the light.
SpotAngle	Sets the angle of the spotlight cone.
Range	Sets the maximum range of the light.

Signals	Description
Enabled	Enables or disables the light source.

MarkupControl

Controls properties of graphical Markup.

Properties	Description
Markup	Specifies the Markup to control.
Text	Specifies text on the markup.
Visible	True if the markup is visible.
Position	Specifies the position of the markup arrow.
BackColor	Specifies the background color of the markup.
ForeColor	Specifies the text color of the markup.
FontSize	Specifies the text size of the markup.
Topmost	If true the markup will not be obscured by other objects.

Continues on next page

Signals	Description
GetValues	Set to high (1) retrieve property values for the selected Markup.

ApplicationWindowPanel

Controls a light source.

Properties	Description
ApplicationName	Specifies the name of the application executable to capture.
ApplicationTitle	Title of the main window to capture.
Width	Panel width.
Height	Panel height.
ClipLeft	Pixels to clip from the left edge.
ClipRight	Pixels to clip from the right edge.
ClipTope	Pixels to clip from the top edge.
ClipBottom	Pixels to clip from the bottom edge.

ColorTable

Stores a list of colors.

Properties	Description
NumColors	Specifies the number of colors in the list.
SelectedColorIndex	The index of the currently selected color in the list.
SelectedColor	The currently selected color.
Color0	
Color1	

ConveyorControl

Controls a Conveyor with I/O signals.

Properties	Description
Conveyor	Specifies the Conveyor to control.

Signals	Description
Stop	Stops the conveyor.
Velocity	Sets the velocity of the conveyor.
Acceleration	Sets the acceleration of the conveyor

DataTable

Stores a list of objects.

Properties	Description
DataType	The item data type. Supports Numeric, Text, Color and Object.
NumItems	The number of items in the list.
SelectedIndex	The index of the currently selected item in the list.
SelectedItem	The value of the currently selected item

Continues on next page

7.1.2 Basic Smart Components
Continued

ExecuteCommand

Executes a RobotStudio command.

Properties	Description
CommandID	Specifies the ID of the command to execute.

Signals	Description
Execute	Executes the command, if it is enabled.

PaintApplicator

Applies paint to a part.

Properties	Description
Part	Specifies the Part to be painted.
Color	Specifies the paint color.
ShowPreviewCone	True if a preview paint cone should be displayed.
Strength	The amount of paint to add each time step.
Angle	Angle of the paint cone.
Range	Range (max distance) of the paint cone.
Width	Maximum width of the covered area.
Height	Maximum height of the covered area.

Signals	Description
Enabled	Set to high to enable painting during simulation.
Clear	Clears all paint.

7.2 The Infeeder example

Overview

A typical application of Smart Components is in the material handling scenario, for example, simulating an infeeder. You can create dynamic objects which move in a straight line until they arrive at a picking position. A gripper attached to a robot picks the object and places it. When an object is removed from the picking position, a new object gets created automatically. You can save the components associated with an infeeder as a Smart Component library for reuse.

To access the *pack&go* files for this infeeder station, click **File** tab, click **Open** and then under **Samples** click Demo Stations. Click the tiles, *SC Infeeder example* and *SC Infeeder finished*. Open the *SC Infeeder example* before starting this example.

> **Note**
>
> RobotStudio provides several sample stations to help users. To download these stations click **File** tab, click **Open** and then click **Samples**.

Prerequisites

All basic components required for material handling must be imported to the *station*. These components are a box, belt conveyor, robot, stop, and pallet.

> **Note**
>
> The station in this example is Physics enabled.

Workflow

1. Create a station with a robot, belt conveyor, robot, stop, and pallet. Add the required I/O signals to the *virtual controller*.
2. Create the Infeeder Smart Component, this involves creating a work piece, moving work pieces on the conveyor, placing a line sensor on the Stop, setting up I/O connections between the Smart Components using bindings and then saving this Smart Component as a library file.
3. Create a Smart Component for the Gripper.
4. Create a Smart Component for the pallet.
5. Setup I/O connections and create bindings between the Smart Components and the virtual controller.
6. Connect all Smart Components of the *station* in the *Station logic*.

Generating work pieces

1. On the **Modeling** tab, in the **Create** group, click the **Smart Component** button. The Smart Component Editor opens and a new Smart Component gets created in the **Layout** browser.
2. In the **Layout** browser, double-click the Smart Component and change the name to *SC_Infeeder*.

Continues on next page

3 In the **Smart Component** editor, click the **Compose** tab and then click the **Add component** button.

4 Click the **Actions** gallery and then click **Source**. The **Source** gets added as a child component.

> **Note**
>
> This *Smart Component* creates copies of the selected GraphicComponent, for example, box.

5 Right-click **Source** and then click **Properties**.

6 In the **Properties** browser, select **Source** as box and **Physics Behavior** as **Dynamic** to allow the gravity of the box to control the response of the box during the simulation.

7 Select **Transient** so that the boxes are generated only during simulation and they are deleted automatically when the simulation stops.

8 Click **Apply**.

While creating Smart Components, to minimize the risk of making mistakes, it must be tested as often as possible.

To test the Smart Component, start the simulation and then, click **Execute**. New box gets generated.

Moving work pieces on the conveyor

1 In the **Layout** browser, drag and drop the **Belt Conveyor** on the *SC_Infeeder*.

2 Right-click the **Belt Conveyor** and from the context menu, select **Physics** and then set the behavior to **Fixed**.

If you want to test the Smart Component at this point, start the simulation and then, click **Execute** in the **Source: Properties** browser. The box gets created and it will be placed on the conveyor. Stop the simulation before continuing the procedure.

3 Right-click the **Belt Conveyor** and from the context menu, select **Physics** and then click **Surface Velocity** to enable surface velocity and to set the direction of movement and speed of the belt.

4 In the **Layout** browser, right-click **Source** and then click **Properties**.

5 To test the Smart Component, start the simulation and then, click **Execute**.

You can see the box moving on the conveyor and moving through the Stop and Pallet.

6 To prevent boxes from moving through the Stop and the Pallet, in the **Layout** browser, right-click the Stop and from the context menu, select **Physics** and then set the behavior to **Fixed**. Repeat the same step for the Pallet.

Placing the Line Sensor on the Stop

1 Right-click the Stop and from the context menu select **Modify** and then ensure that the **Detectable by Sensors** check box is not selected.

Continues on next page

If the **Detectable by Sensors** is selected then the sensor will detect the Stop as an object. In that case the sensor will not detect any other object.

2 In the Smart Component editor, click **Add component** and from the context menu, click **Sensors** and then click **LineSensor**. The **LineSensor** gets added to the Smart Component.

3 Place the **Line Sensor** at the center of the Stop using the the snap mode **Snap Center**.

4 In the **Layout** browser, right-click **LineSensor** and then click **Properties**. Enter the length and direction of the sensor and click **Apply**.

Line Sensor has two signals *Active (Digital) - Set to 1 to activate the sensor* and *SensorOut (Digital) - Goes high (1) when an object intersects the line*. When the Line Sensor detects a part the property *SensedPart* gets populated.

5 It is possible to view how these signals and properties get activated during simulation:

a In the **Properties: LineSensor** browser, click the down arrow next to the close button and select **Tear Off** to separate **Properties: LineSensor** from the **Layout** browser.

b Start the simulation and click **Execute** in the **Properties: Source** browser. The box gets created and starts to move on the conveyor. Observe the change in the signal and property values of the LineSensor in the **Properties: LineSensor** window. When the Line Sensor detects the box, the **Sensed Part** field gets populated with the box object and the **SensorOut** signal becomes active.

Configuring the I/O signals for the SC_Infeeder Smart Component

The digital output from the Infeeder Smart Component can be connected to the digital input of the *virtual controller*.

1 In the Smart Component editor, click the **Signals and Connections** tab and then click **Add I/O Signals**.

2 In the **Add I/O Signals** window, select the **Type of Signal** as *DigitalOutput* and enter a suitable name for the **Signal Base Name**, for this example, type in, *doBoxInPos* and click **OK**.

3 Add a connection between the Line Sensor's *SensorOut* property to the output signal of the *SC_Infeeder*.

In the Smart Component editor, click the **Signals and Connections** tab and then click **Add I/O Connection**.

4 In the **Add I/O Connection** window, Select the **Source Object** as *Line Sensor*, **Source Signal** as *SensorOut*, **Target Object** as *SC_Infeeder* and **Target Signal or Property** as *doBoxInPos* and click **OK** .

The logic behind this binding is that when the *LineSensor* senses the box, the *SensorOut* signal becomes active and this sends an I/O signal to the SC_Infeeder's output signal *doBoxInPos*. This output can eventually be connected to the virtual controller such that the robot picks the box when the *LineSensor* senses the box or when the box is in the picking position and is ready to be picked.

Continues on next page

5 In the **Smart Component** editor, click **Add component** and from the context menu, select **Signals and Properties** and then add **LogicGate**.

6 In the **Properties: LogicGate**, in the Operator drop-down, select **NOT**.

The Smart Component *LogicGate* performs logical operations on digital signals. The logic gate used in this example is a **NOT** gate or the inverter. For an inverter the output is exactly the opposite of the input, so if the input is 0, the output is 1 and vice versa.

7 In the **Smart Component** editor, click the **Signals and Connections** tab and then click **Add I/O Connection**.

8 In the **Add I/O Connection** window, perform the following connections.

Source Object	Source Signal	Target Object	Target Signal or Property	Description
LineSensor	SensorOut	LogicGate [NOT]	InputA	When the *LineSensor* senses the box, its output signal is high. The robot gets the *doBoxInPos* signal and it picks the box. Then the *LineSensor* does not have any object to sense, hence the *SensorOut* signal becomes zero, which is the InputA to the *LogicGate*. The *LogicGate* eventually inverts the signal and sends the Output to the Source.
LogicGate [NOT]	Output	Source	Execute	The Source upon receiving the signal from *LogicGate*, triggers *Execute* signal to create a new box.

This binding will ensure steady inflow of objects on the conveyor.

9 Save the Smart Component SC_Infeeder as a library for reuse.

Picking and placing work pieces with a robot

Overview

The *Vacuum Tool* Smart Component requires the following components.

- CAD geometry of the *Vacuum Tool*.
- Attacher Smart Component
- Detacher Smart Component
- Logical Gate (NOT)
- LogicSRLatch

Creating the Vacuum Tool Smart Component

1 On the **Modeling** tab, in the **Create** group, click the **Smart Component** button. The Smart Component editor opens and a new Smart Component gets created in the Layout browser.

2 In the **Layout** browser, double-click the Smart Component and change the name to *SC_VacuumTool*.

3 Add the CAD geometry of the *Vacuum Tool* to the new Smart Component.

4 In the **Smart Component** editor, in the **Compose** tab, right-click the Vacuum Tool and select **Set as Role**.

Continues on next page

With this option, the *tooldata* of the *Vacuum Tool* will be visible when it is attached to the robot.

5 Right-click the *Vacuum Tool* and from the context menu select **Modify** and then ensure that the **Detectable by Sensors** check box is not selected.

If the **Detectable by Sensors** is selected then the sensor will detect the *Vacuum Tool* as an object. In that case the sensor will not detect any other object.

6 In the Smart Component editor, click **Add component** and from the context menu, click **Sensors** and then click **LineSensor**. The **LineSensor** gets added to the Smart Component.

7 Place the *LineSensor* at the center of the *Vacuum Tool* using the the snap mode **Snap Center**.

8 In the **Layout** browser, right-click *LineSensor* and then click **Properties**. Enter the length and direction of the sensor and click **Apply**.

It is recommended to extend the length of the *LineSensor* beyond the vacuum cup of the *Vacuum Tool*.

9 Configure I/O signals for the Smart Component to establish communication between the robot and the *Vacuum Tool*.

Name	Signal Type	Value	Description
diAttach	DigitalInput	0	This signal gets active when the robot sends the *doVacuum* signal to the *Vacuum Tool*.
doAttached	DigitalOutput	0	This signal gets active when the *Vacuum Tool* picks the box, or when the *LineSensor* in the *Vacuum Tool* detects the box.

Adding the Attacher and Detacher

1 In the Smart Component editor, click **Add component** and from the context menu, click **Actions** and then click **Attacher**.

Repeat the same step and add the **Detacher** Smart Component. These Smart Components are added to attach an object to the *Vacuum Tool* and then to detach the same object.

2 In the **Properties: Attacher**, in the **Parent** drop-down, select the **Vacuum Tool(SC_VacuumTool)**.

The Attacher must pick any object the Line Sensor senses, to achieve the same, add the following binding.

Source Object	Source Property	Target Object	Target Property or Signal	Description
LineSensor_2	SensedPart	Attacher	Child	The *Sensed Part* property of the *LineSensor* is connected as the child of the *Attacher*, hence it picks any object that the *Sensed Part* senses.
Attacher	Child	Detacher	Child	The *Detacher*'s child property is connected to the *Attacher*'s child property such that the *Detatcher* detaches the object the attacher is attached to.

Continues on next page

Configuring the I/O signals

1 In the **Smart Component** editor, click **Add component** and from the context menu, select **Signals and Properties** and then add **LogicGate**.

Repeat the same step to add the *LogicalSRLatch*.

2 In the **Properties: LogicGate**, in the Operator drop-down, select **NOT**.

The Smart Component *LogicGate* performs logical operations on digital signals. The logic gate used in this example is a **NOT** gate or the inverter. For an inverter the output is exactly the opposite of the input, so if the input is 0, the output is 1 and vice versa.

3 Add the following signals and connections.

Source Object	Source Signal	Target Object	Target Signal or Property	Description
SC_VacuumTool	diAttach	LineSensor_2	Active	When the input signal *diAttach* of the SC_VacuumTool is 1, then the *Line Sensor* gets activated.
LineSensor_2	SensorOut	Attacher	Execute	When the *LineSensor* senses the object, the *SensorOut* signal goes high. Then the *Attacher* sends the *Execute* signal to the Smart Component to attach the object.
SC_VacuumTool	diAttach	LogicGate2[NOT]	InputA	When the sensor senses the object the *diAttach* is high, this signal is send to the *Logical gate2[NOT]* which inverts the signal.
LogicGate2[NOT]	Output	Detacher	Execute	The inverted signal is send to the *Detacher* which then activates the *Execute* signal. The purpose of this connection is to make the *Detacher* to detach the object.
Attacher	Executed	LogicalSRLatch	Set	When the attacher attaches an object it sends the Set signal to the *LogicalSRLatch*.
Detacher	Executed	LogicalSRLatch	Reset	When the detacher detaches the object it sends the *Executed* signal to the *LogicalSRLatch*. SR latch has two inputs, set and reset. The S input is used to produce high on the output signal. The R input is used to produce low on the output signal. An SR latch always holds the steady output. The purpose of using a SR Latch is to provide a steady output form the *SC_Vacuum Tool*. The output signals from the Attacher and Detacher are infrequent. Using SR Latch ensures a steady output from the *SC_VacuumTool*.

Continues on next page

Source Object	Source Signal	Target Object	Target Signal or Property	Description
LogicalSRLatch	Output	SC_VacuumTool	doAttached	The output signal from the Latch is send to the output signal of the *SC_VacuumTool*.

4 Save the Smart Component as a library for reuse.

Deleting work pieces from a station

1 On the **Modelling** tab, in the **Create** group, click the **Smart Component** button. The **Smart Component** editor opens.

2 In the **Layout** browser, double-click the Smart Component and change the name to *SC_OutPallet*.

Drag and drop the CAD *geometry* of the pallet to add it to the *SC_OutPallet*.

3 Right-click the Pallet and from the context menu select **Modify** and then ensure that the **Detectable by Sensors** check box is not selected.

If the **Detectable by Sensors** is selected then the sensor will detect the Pallet as an object. In that case the sensor will not detect any other object.

4 In the **Smart Component** editor, click **Add component** and from the context menu, click **Sensors** and then click **PlaneSensor**. The **PlaneSensor** gets added to the Smart Component.

5 In the **Layout** browser, right-click the **PlaneSensor** and then click **Properties**. Select the origin and axes such that the *Plane Sensor* covers the entire surface of the pallet. Click **Apply**. *Plane Sensor* detects any object that intersects the defined surface.

The *Plane Sensor* has two signals *Active (Digital) - Set to 1 to activate the sensor* and *SensorOut (Digital) - Goes high (1) when an object intersects the line*. When the Plane Sensor detects a part the property *SensedPart* gets populated.

6 Click the **Actions** gallery and then click **Sink**.

Note

This Smart Component removes the selected GraphicComponent from the view, for example, box.

7 Add the following binding between the Plane Sensor and the Sink.

In the **Smart Component** editor, click the **Properties and Bindings** tab and then click **Add Binding**.

Source Object	Source Property	Target Object	Target Property or Signal	Description
PlaneSensor	SensedPart	Sink	Object	This binding connects the *SensedPart* property of the *Plane Sensor* to the Object property of the *Sink* such that it deletes the sensed part.

Continues on next page

If the *Plane Sensor* and *Sink* Smart Components are connected at this stage, the box gets removed as soon as the *Plane Sensor* detects it. But it is possible to add a time delay between the sensor sensing the box and the box getting deleted.

8 In the **Smart Component** editor, click **Add component** and from the context menu, select **Signals and Properties** and then add **LogicGate**.

9 In the **Properties: LogicGate**, select the **NOP Operator** and set the time **Delay** in seconds.

The Smart Component *LogicGate* performs logical operations on digital signals. The logic gate used in this example is a **NOP** gate, it delays the subsequent operation by the selected time.

10 In the **Smart Component** editor, click the **Signals and Connections** tab and then click **Add I/O Connection**.

11 In the **Add I/O Connection** window, perform the following connections.

Source Object	Source Signal	Target Object	Target Signal or Property	Description
PlaneSensor	SensorOut	LogicGate [NOP]	InputA	When the *Plane Sensor* senses the box, its output signal is high, the *SensorOut* signal is sent to the *LogicGate*. The *LogicGate* delays the *Execute* property of the Sink.
LogicGate [NOP]	Output	Sink	Execute	The Sink upon receiving the signal from *LogicGate*, triggers *Execute* signal to delete the box.

12 Save the Smart Component as a library for reuse.

Configuring the station logic for the Infeeder

Adding and exposing I/O signals in the virtual controller

Before configuring the *station logic* of the Infeeder, add the required I/O signals to the controller, for example, *diBoxInPos*, *diVacuum* and *doVacuum*.

1 In the **Controller** browser, expand the **Configuration** node and then double-click **I/O System**.

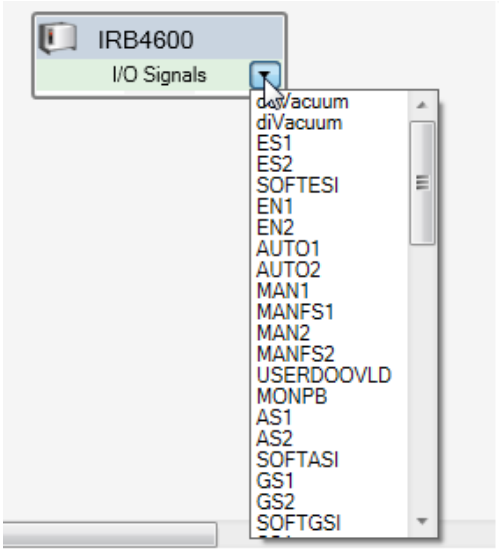
2 In the **Configuration-I/O System** window, under **Type**, right-click **Signal** and then add the required signals in the **Instance Editor**.

The controller must be restarted for executing the changes.

The signals added to the *virtual controller* using the **Instance editor** must be exposed in the **Design** view of the Station logic window for creating connection graphically between various station elements and the virtual controller.

Continues on next page

In the **Station Logic** window, click the **Design** tab and then click the arrow in the virtual controller and then select the signal that must be exposed.



xx1800002954

Configuring Station Logic

1. On the **Simulation** tab, in the **Configure** group, click **Station Logic**. The **Station Logic** window opens. This window displays all the components that are part of the *station*.
2. Click the **Signals and Connections** tab, and then click **Add I/O Connection**. In the **Add I/O Connections** window, create the following connections.

Source Object	Source Signal	Target Object	Target Signal or Property	Description
SC_InFeeder	doBoxInPos	IRB4600	diBoxInPos	When the *Line Sensor* detects the box, the Infeeder sends the output signal *diBoxInPos* to the virtual controller.
IRB4600	doVacuum	SC_VacuumTool	diAttach	The virtual controller sends the *doVacuum* signal to the *SC_VaccumTool*. The *SC_VaccumTool* simulates the vacuum and attaches the part.
SC_VacuumTool	doAttached	IRB4600	diVacuum	The *SC_VacuumTool* grips the object and sends the signal to the virtual controller.

Tip

Alternatively, these signals can be configured in the **Design** view of the **Station Logic** window.

3. Use the following piece of code for simulating the Infeeder.

```
MODULE Module1
```

Continues on next page

```
CONST robtarget
     pPick:=[[300.023,150,209.481],[0,-0.707106781,0.707106781,0],[0,0,0,0],
[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

!***********************************************************
!
! Procedure main
!
! Smart Component example, Pick and Place application.
!
!***********************************************************
PROC main()
WHILE TRUE DO
PickPart;
ENDWHILE
ENDPROC

PROC PickPart()

!** The robot moves to a wait position 200 mm above the pick
     position. **

MoveJ Offs(pPick,0,0,200),v500,z1,tVacuum\WObj:=wobjInFeeder;

!** The robot waits for a box to pick at the infeeder stop.
     **

WaitDI diBoxInPos,1;

!** The robot goes to the pick position. **

MoveL pPick,v100,fine,tVacuum\WObj:=wobjInFeeder;

!** To attach the box, the robot turns on the digital output
     signal "doVacuum" which is connected to the
Smart Component "SC_VacuumTool". **

SetDO doVacuum,1;

!** The robot waits for the digital input signal "diVacuum"
     to go high, which comes from "SC_VacuumTool"
and indicates that the box is attached. **

WaitDI diVacuum,1;

!** The robot moves up from the infeeder. **

MoveL Offs(pPick,0,0,200),v500,z1,tVacuum\WObj:=wobjInFeeder;

!** The robot moves to the drop position above the pallet. **
```

Continues on next page

```
MoveL
     Offs(pPick,0,-800,200),v500,fine,tVacuum\WObj:=wobjInFeeder;

!** To detach the box, the robot turns off the digital output
     signal "doVacuum" which is connected to the
Smart Component "SC_VacuumTool". **

SetDO doVacuum,0;

!** The robot waits for the digital input signal "diVacuum"
     to goes low, which comes from "SC_VacuumTool" and
indicates that the box is detached. **

WaitDI diVacuum,0;

!** The wait time simulates the time it takes for a real vacuum
     gripper to loose the vacuum. **

WaitTime 0.5;
ENDPROC
ENDMODULE
```

7.3 Cable Simulation using Physics

Overview

In a station, the cable connecting robots undergo extensive wear and tear which reduces its life span. RobotStudio's physics enabled cable simulation helps in selecting the right material of the cable and accurately designing the length, radius and mounting height of the cable to optimize its performance.

Creating a cable

1. In the **Modelling** tab, in the **Create** group, click **Cable** to create a cable, the **Create Cable** pane opens, set cable properties like length, radius, material and so on here.

   You can also create a cable from the **Physics** contextual tab.
2. Click in the station/robot to add the start and end points of the cable .
3. Select a point on the cable, and drag it to add a control point. You can also set the control point in the **Create Cable** pane. The control point can be any **Free Point** or an **Attached Point**. You can attach the cable to any objects using the **Attached to** list box in the **Create Cable** pane.
4. Click **Create**, the new cable gets displayed in the **Layout** browser.

Setting the properties of materials

Select one of the following options to set/edit the material of cable.

- In the **Physics** contextual tab, click **Material**. The **Material Properties** pane opens.

  You can choose the material either from the standard material list or can use custom material. Click the **Edit Materials** option to edit material properties.
- In the **Modelling** browser, right-click a part and then click **Physics\Material\ Material Properties** to open the **Physics Material** pane.

  You can select your material either from the standard material list or can use custom material. Click the **Edit Materials** option to edit material properties.

Modifying length of a cable

1. In the **Layout** browser, right-click cable, then click **Modify Cable**. The **Modify Cable** pane opens.
2. Edit the required parameter and click **Apply**.

Click options **Shorter** or **Longer** to stretch or skew the cable.

Applying Behavior to objects

Use the behavior feature to set various motion related characteristics to a RobotStudio object. A physics object follows the rules of physics during simulation. In the **Layout** browser, right-click a part and then select **Physics** to set various behavior options. The following settings are available.

- Inactive: This object will not interact with other objects during simulation.
- Fixed: The position of the object remains fixed during simulation.
- Kinematic: During simulation, RobotStudio controls the motion of the object.

Continues on next page

- Dynamic: During simulation the object follows rules of physics while in motion.

Similarly any joint created using the **Physics Joint** options will follow rules of physics. These options are **Rotational joint**, **Prismatic joint**, **Ball joint** and **Lock joint**.

Material selection in Physics simulation

RobotStudio includes a number of default materials for adding their material properties in the simulation. You can create a copy of one of the default materials and then manipulate its properties to influence the behavior of items in the simulation.

Property	Description
Density	Mass per volume.
Young's modulus	A mechanical property that measures the stiffness of a solid material.
Poisson's ratio	Poisson's ratio is a measure of the contraction that happens when an object is stretched.
Coefficient of restitution	Ratio of the final to initial relative velocity between two objects after they collide. It normally ranges from 0 to 1, where 1 would be a perfectly elastic collision.
Roughness (friction)	Friction is the force resisting the relative motion of solid surfaces sliding against each other.
Cable properites	Cables are internally modelled as solid bodies attached with springs. These springs can create a damping effect on the cable and its stiffness may vary. In addition to the standard properties, cables are affected by the spring damping (contact damping) and stiffness (Young's modulus)

Collision geometry in physics modelling

Tuning the motion behavior of a robot for simulation depends a lot on the physical properties of the participants in the simulation. Properties such as mass, density, friction between the surfaces, and effects of gravity can deviate the predicted behavior of objects in the *station*. RobotStudio uses collision geometry of objects for collision calculations. Creating collision geometry is a computationally intensive feature for the PC as it uses more CPU resources for collision calculations. Hence it is recommended that static objects in the station must be excluded in collision calculations.

RobotStudio uses *collision geometry* in physics simulations and regular *geometry* for collision detection.

7.4 Physics joints

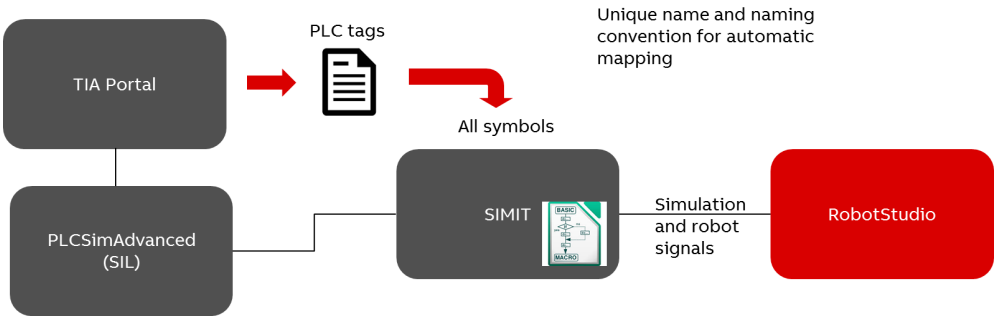
Joints define the geometrical relationship between one rigid body relative to the world or, two or more rigid bodies relative to each other, that should be met under certain conditions. Each joint is build upon elementary joints or *links* that defines the geometrical relationship. Acting upon these elementary joints can be secondary order joints such as motors, limits or locks. The available joints in RobotStudio are rotational, *Prismatic* , *Cylindrical*, *ball* and *Lock* joints.

7.5 Virtual commissioning using the SIMIT Smart Component

Overview

The SIMIT *SmartComponent* enables signal communication between SIEMENS SIMIT Simulation Platform and RobotStudio to perform virtual commissioning. SIMIT uses Shared Memory to communicate with RobotStudio. This shared memory is created in SIMIT. Input signals are written by SIMIT to the memory area and output signals are read by SIMIT from the memory area. The schematic overview of the setup:

> **Note**
>
> Refer to the SIMITConnection Smart Component section to know its Properties and Signals.



xx1900000807

Prerequisites

RobotStudio 2019 and SIMIT ULTIMATE 9.1 and higher versions must be installed on the same computer.

Signals/Symbols

The SIMITConnection Smart Component recognizes three *symbol* types based on a naming convention. These symbols are Robot, Station and Process. Robot symbols are robot controller signals. Station symbols (non-robot signals) control the connection between the PLC and various *station components* such as sensors and actuators during simulation. Process symbols are internal signals that reside only in SIMIT and are not passed to RobotStudio.

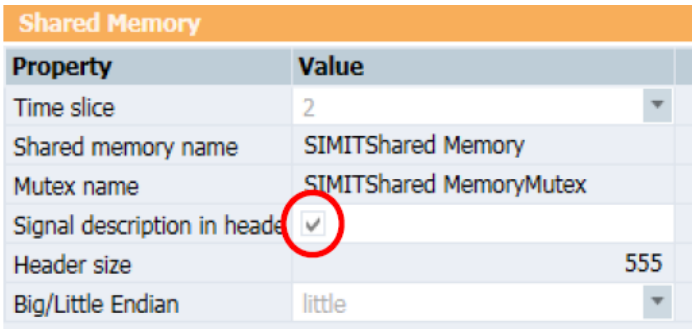
Workflow

Use the following workflow to configure, start and execute the exchange of signals between SIMIT and RobotStudio.

1. Configure TIA project in SIMIT. While configuring *symbols*, ensure that each symbol has a unique name in the TIA project.
2. Export the symbols from TIA portal as an Excel symbol document.
3. Import the symbol document to SIMIT.
4. A coupling gets created between SIMIT and the PLC simulator (PLCSim, *PLCSimAdvanced*, OPC or other programs).

Continues on next page

5 A shared memory coupling gets created to be used from RobotStudio.
6 The Process symbol only resides in SIMIT and must be handled through SIMIT macros, UI and so on.
7 Import the Robot and Station symbols into the shared memory and perform the following setting.



xx1900000805

- **Time slice**: Set the cycle at which the coupling exchanges data. Time slice 2 is the default, corresponding to a cycle of 100 ms.
- **Shared Memory name**: Enter the name with which the shared memory area can be addressed.
- **Signal description in header**: Choose whether or not SIMIT will create an extended header area. Signal description in header must be checked.
- **Big/Little Endian**: Little endian denotes that the least significant byte (the lowest memory address) is stored first. This must be selected.

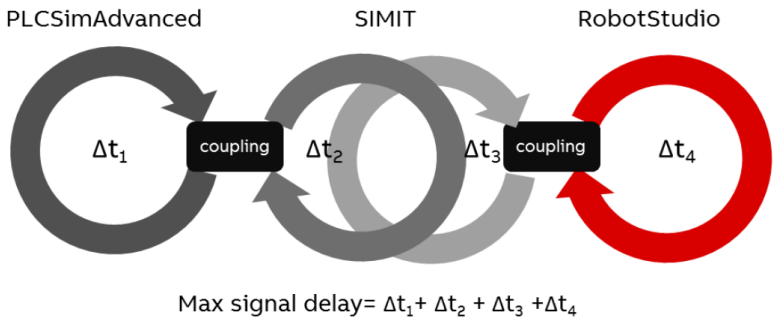
8 Start the simulation in SIMIT.
9 Connect the SIMITConnection *SmartComponent* in RobotStudio.
   a In RobotStudio, on the **Simulation** tab, in the **Configure** group, click **Station logic**.
   b In the **Station Logic** window, click **Add component**, then select the **SIMIT Connection** Smart Component under **PLC**.
   c In the **Layout** browser, right-click **SIMITConnection**, and click **Properties**, enter the Shared memory name. Since RobotStudio and SIMIT connects through the shared memory, it is important that the shared memory name must be identical in both SIMIT and in RobotStudio.
   d Start the SIMIT simulation, switch to RobotStudio and in the **Layout** browser, right-click **SIMIT Connection** and then click **Connect** to establish the connection between SIMIT and RobotStudio.
10 Start simulation in RobotStudio.

Note

Only virtual controllers in the project (or station) can be used by the SIMITConnection Smart Component. It is not possible to use real robot controllers or virtual controllers that are not part of the project (or station).

Continues on next page

Time synchronization



xx1900000808

The maximum signal delay is the sum of the *cycle time* of individual products. After the cycle the coupling exchanges data. This delay will affect the total cycle time of the production cell. Usually connection gets established between the PLC and the devices of the cell such that the signal delay will be duplicated since it is a set and wait signal before continuing the simulation. This significantly affects the total simulated cycle time.

Retrieving robot joint values from RobotStudio to SIMIT

Naming convention for joint signals

It is possible to retrieve the joint values from RobotStudio to the SIMIT shared memory. These values get updated at every 24 milliseconds in the simulation (12ms for picker robots). When simulation is not running, RobotStudio updates joint values to SIMIT only twice per second.

The joint signals (data type REAL) must abide to the following naming conventions in the shared memory. The joint symbol must be named either *Joint[joint_index]symbol_id* or *Joint[system_name][mech_unit][joint_index]symbol_id*.

Joint symbol naming convention	Description
Joint[joint_index]symbol_id	• *[joint_index]* is an integer 1-7. By this convention only one robot with a single mechanical unit can be used. • *symbol_id* is an optional arbitrary identifier that can be used to give a descriptive name to the *joint*. The *symbol_id* is an optioned field which can be excluded. For example, to retrieve data from axis 3 of a single robot in RobotStudio, the symbol name in SIMIT is *[Joint[3]]*.

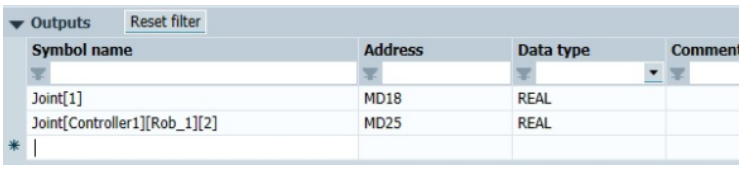
Continues on next page

Joint symbol naming convention	Description
Joint[system_name][mech_unit][joint_index]symbol_id	• *[system_name]* is a unique id of the robot system, therefore joint values from several robots can be retrieved from RobotStudio. • *[mech_unit]* is the mechanical unit of the robot. • *[joint_index]* is an integer 1-7. • *symbol_id* is an optional arbitrary identifier that can be used to give a descriptive name to the joint. The *symbol_id* is an optional field which can be excluded. For example, to retrieve data from axis 3 of mechanical unit *ROB_1* of the robot with system name *System1* in RobotStudio, the symbol name in SIMIT is *Joint[System1][ROB_1][3]*.

Procedure

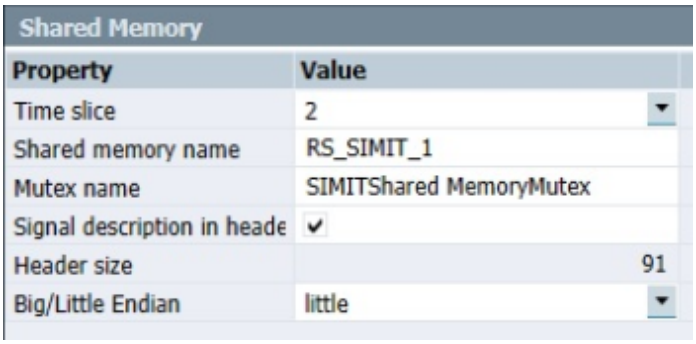
This procedure requires an active *virtual controller*.

1. Create a project in SIMIT, In the **Project** tree view, right-click **Coupling** and then select **Shared Memory**.
2. In the **Output** window, enter the names of joint symbols. The joint symbol names used in this example follows the required naming convention.



xx1900000810

3. In the **Shared Memory** window, perform the following setting.

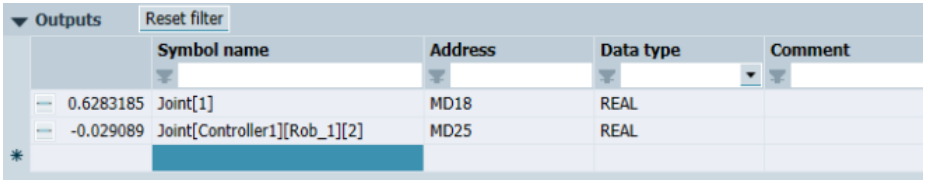


xx1900000809

4. In RobotStudio, on the **Simulation** tab, in the **Configure** group, click **Station logic**.
5. In the **Station Logic** window, click **Add component**, then select the **SIMIT Connection** Smart Component under **PLC**.
6. In the **Layout** browser, right-click **SIMIT Connection**, and click **Properties**, enter the Shared memory name. Since RobotStudio and SIMIT connects through the shared memory, it is important that the shared memory name must be identical in both SIMIT and RobotStudio, for example, *RS_SIMIT_1*.

Continues on next page

7 Start the SIMIT simulation, switch to RobotStudio and in the **Layout** browser, right-click **SIMIT Connection** and then click **Connect** to establish the connection between SIMIT and RobotStudio. On successful connection, the joint values get displayed in SIMIT.

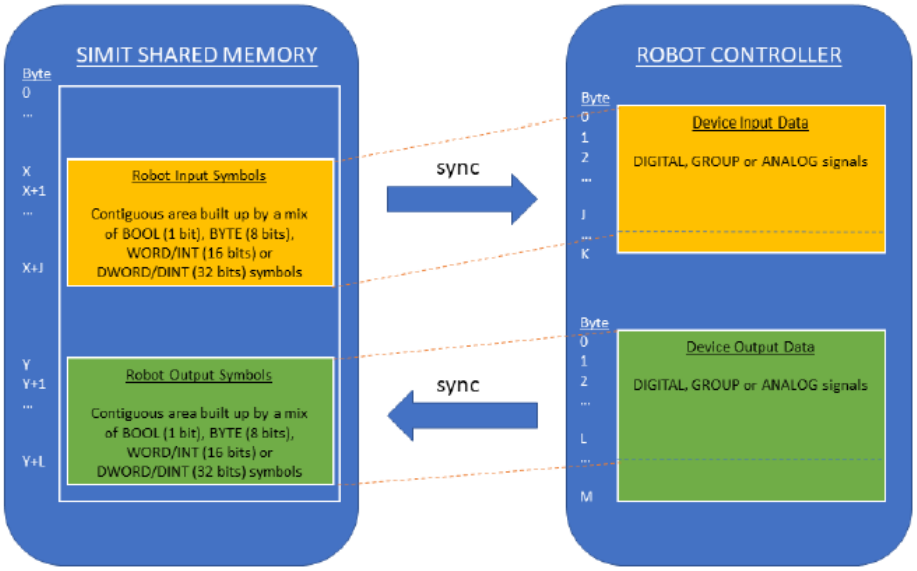


xx1900000811

Coupling robot signals with SIMIT

Naming convention for robot signals

The SIMIT *SmartComponent* can synchronize a set of SIMIT symbols with the input or output data of a device in the robot controller EIO configuration based on the following conditions.



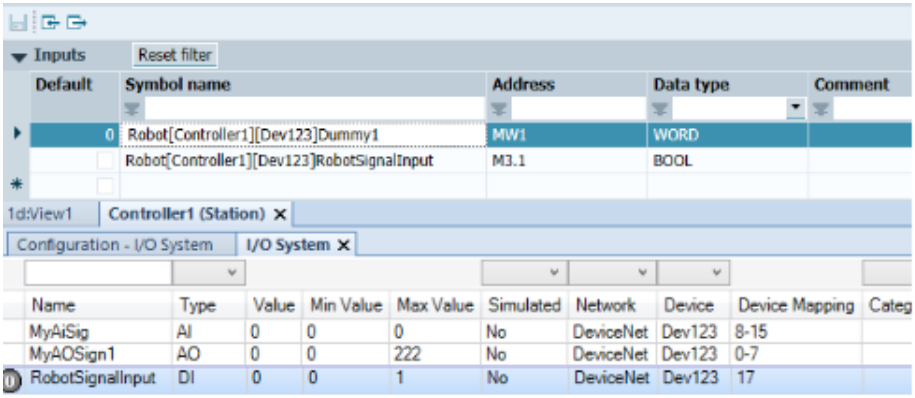
xx1900000832

- The robot input or output symbols for a specific EIO device must contain contiguous shared memory.

  In the following example a digital input signal with Device Mapping 17 is represented in SIMIT. To ensure continuity in the shared memory, a dummy symbol of datatype WORD is used here. A symbol of type WORD is represented by 16 bits in SIMIT. The shared memory area mapped to the device *[Dev123]* starts at the first byte, for example, "1" in *MW1*. The *Robot[Controller1][Dev123]Dummy1* symbol occupies two bytes. That is why

Continues on next page

the *RobotSignalInput* is mapped to the first bit of the third byte *M3.1*. This will ensure the absence of memory gaps in the shared memory.



xx1900000837

- The robot input or output symbol area can start at any address in the shared memory, but always map it to the beginning of the EIO device input or output data.
- The start and end of a robot symbol area is determined by the first and last robot symbol for a specific robot system and EIO device.
- The size of the robot input or output symbol area in the shared memory must be less than or equal to the size of the input or output data area of the EIO device.
- The robot *symbols* can belong to any of the datatypes: BOOL (1 bit), BYTE (8 bits), WORD/INT (16 bits) or DWORD/DINT (32 bits). If a BOOL symbol is used, all bits in the byte must be assigned to a BOOL symbol, unless it is the last byte where the last bits can be left unassigned. REAL type is not supported for robot symbols.
- The robot symbols in SIMIT need not correspond to (or must have the same size as) the individual I/O signals in the robot controller. Examples:
  - One BYTE (8 bits) robot symbol could map to the device data for 8 digital I/O signals in the robot controller.
  - 2 WORD (16 bits each) robot symbols could map to a 32-bit group signal in the robot controller.

  It is possible to create robot symbols that exactly correspond to a specific symbol, for example, a *BOOL* (1 bit) robot symbol could be created to correspond to a digital I/O signal in the robot controller.

Continues on next page

Robot symbol naming convention	Description
Robot[system_name][device_name]symbol_id	• The *[system_name]* is a unique id of the robot system, which makes it possible to connect signals from several robots to the PLC. • The *[device_name]* is the name of the device in the robot controller EIO configuration. • The *[symbol_id]* is an arbitrary identifier to make the different robot symbols unique in SIMIT. For example, the first input data symbol of the PLC EIO device in the robot system *System1* could have the symbol name *Robot[System1][PLC]_IN1* in SIMIT.

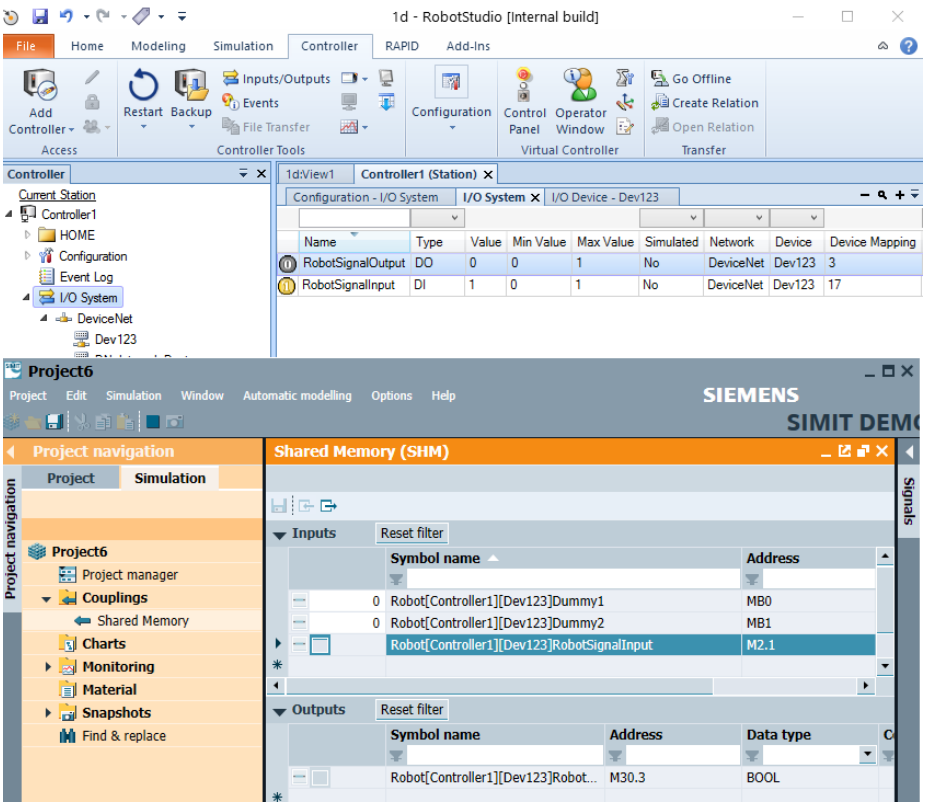
Procedure

This example requires an active virtual controller with a DeviceNet device.

1 In the **Configuration Editor**, create I/O signals for the Device(or any device), for example, *RobotSignalOutput* and *RobotSignalinput*.
2 In RobotStudio, on the **Simulation** tab, in the **Configure** group, click **Station logic**.
3 In the **Station Logic** window, click **Add component**, then select the **SIMIT Connection** Smart Component under **PLC**.
4 In the **Layout** browser, right-click **SIMITConnection**, and click **Properties**, enter the Shared memory name. Since RobotStudio and SIMIT connects through the shared memory, it is important that the shared memory name must be identical in both SIMIT and in RobotStudio, for example, *RS_SIMIT_1*.
5 Switch to SIMIT, add **Inputs** and **Outputs** symbols as per the required naming convention.
6 Start the SIMIT simulation, switch to RobotStudio and in the **Layout** browser, right-click **SIMITConnection** and then click **Connect** to establish the connection between SIMIT and RobotStudio. On successful connection, the

Continues on next page

I/O signals in SIMIT get auto-connected to the SIMITConnection *Smart Component*.



xx1900000831

Coupling station signals with SIMIT

Naming convention for station signals

During simulation, station signals control the connection between the PLC and various station elements such as sensors and actuators. *Smart Components* replicate these station elements in RobotStudio and expose I/O signals that must be connected to the PLC. Station signals can use any of the symbol types in SIMIT. BOOL symbols are assigned to digital signals in RobotStudio, REAL symbols changes to analog signals, similarly BYTE, WORD, INT, DWORD and DINT symbols get converted to group signals.

To automatically map the PLC and other Smart Components in the Station Logic, the signal names must be identical in PLC and in the SmartComponent. This unique name helps RobotStudio to find the correct Smart Component and map the corresponding *symbol* in the PLC. If there are multiple components using the same signal name, the user must perform a manual mapping in the *Station Logic*.

Procedure

This example requires an active *virtual controller*.

1 In RobotStudio, on the **Simulation** tab, in the **Configure** group, click **Station logic**.

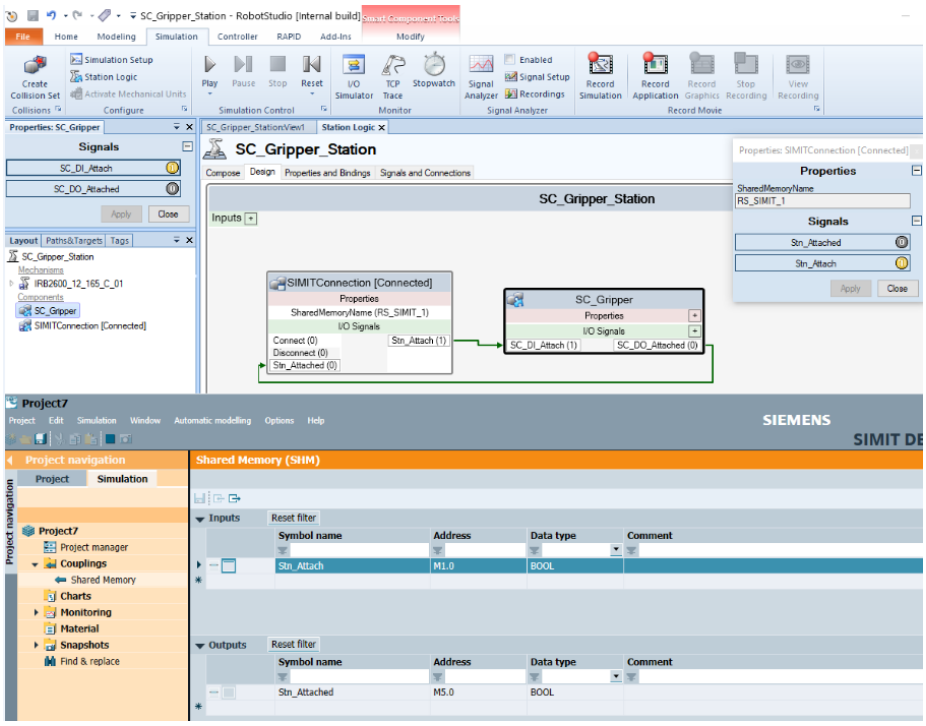
Continues on next page

2 In the **Station Logic** window, click **Add component**, then select the **SIMIT Connection** Smart Component under **PLC**.

3 In the **Layout** browser, right-click **SIMIT Connection**, and click **Properties**, enter the Shared memory name. Since RobotStudio and SIMIT connects through the shared memory, it is important that the shared memory name must be identical in both SIMIT and RobotStudio, for example, *RS_SIMIT_1*.

4 Switch to SIMIT and add I/O signals under **Inputs** and **Outputs** for example, *Stn_Attach* and *Stn_Attached*.

5 Start the SIMIT simulation, switch to RobotStudio and in the **Layout** browser, right-click **SIMIT Connection** and then click **Connect** to establish the connection between SIMIT and RobotStudio. On successful connection, the I/O signals in SIMIT get auto-connected to the SIMITConnection smart component.



xx1900000830

7.6 Virtual commissioning using the OPC UA Client Smart Component

Overview

The OPC UA Client *Smart Component* enables signal communication between RobotStudio and other simulation environments that implement an OPC UA server such as B&R Automation Studio.

OPC UA Smart Component can be used for virtual commissioning of robot cells that include PLCs. To achieve communication between the two simulation environments, station and robot signals need to be mapped to the OPC UA Nodes in the PLC.

> **Note**
>
> Refer to the OpcUaClient Smart Component section to know its Properties and Signals.

Prerequisites

A simulation environment that implements an OPC UA Server, such as B&R Automation Studio.

Mapping of signals to OPC UA Nodes

OPC UA Nodes

An *OPC UA* Node is an entity representing information on an OPC UA Server. Nodes are identified by a unique *NodeId*. The *NodeId* consists of 3 parts; *Namespace*, *Identifier Type*, and *Identifier*.

An OPC UA Server exposes Nodes which can be read or written by OPC UA clients. The OPC UA Client Smart Component maps RobotStudio signals to OPC UA Nodes of the *Variable NodeClass*. For more information on the *Variable NodeClass*, refer to the *OPC UA Specifications, OPC Unified Architecture, Part 3: Address Space Model*.

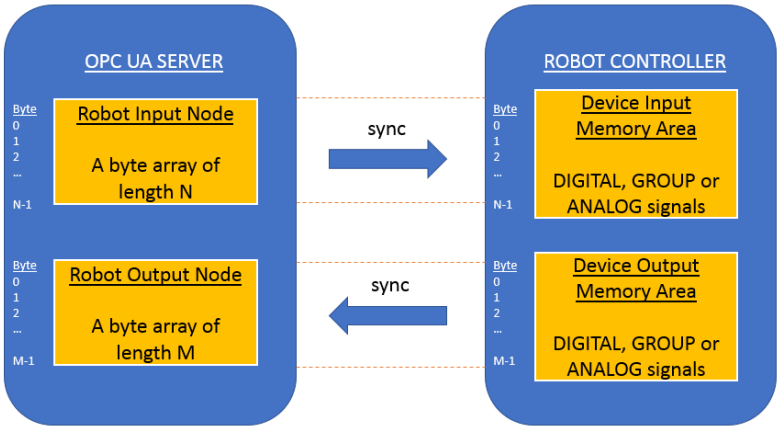
Robot signals

Robot signals are I/O signals that are mapped to the input or output memory area of a device in the *virtual controller*. The OPC UA Smart Component synchronizes the entire input or output memory area of the device to an OPC UA Node, hence all signals on the device are synchronized together without the need to map individual signals. Robot signals are not visible on the OPC UA Client Smart Component.

- The OPC UA Node must be a byte array of the same length as the I/O memory area of the device. The data type of the OPC UA Node must be a subtype of Byte (Identifier 3) with the *ValueRank* set to 1 (OneDimension) and the *ArrayDimensions[0]* set to the same value as *Input Size* or *Output Size* of the I/O device. Refer to the *Technical reference manual - System parameters* for more information about *Input Size* and *Output Size* parameters of the device.

Continues on next page

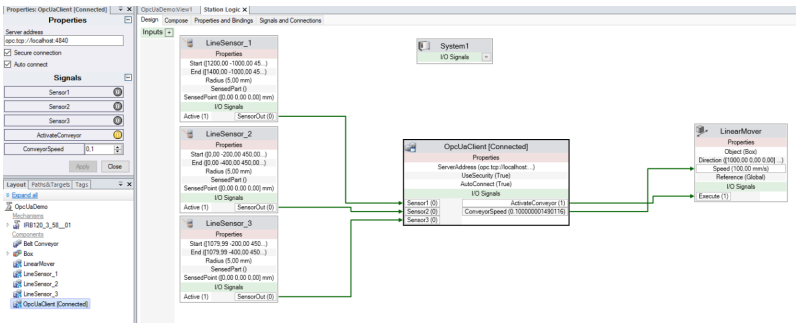
- The bits of the OPC UA Node must have the same meaning as the bits in the device mapping on the robot controller. For example, a digital signal with device mapping 17 in the controller corresponds to bit 1 of byte 2 (zero indexed) in the OPC UA node.



xx2100000249

Station signals

Station signals are non-robot signals that connect the PLC and various *station components* such as sensors and actuators during virtual commissioning. The OPC UA Client Smart Component synchronizes the station signals between RobotStudio and the PLC. These signals are visible on the OPC UA Client Smart Component in the **Station Logic** window.



xx2100000289

In the above figure, *Sensor1*, *Sensor2*, *Sensor3* and so on, are station signals synchronized by the OPC UA Client Smart Component.

Station signals can be mapped to a specific subset of the OPC UA data types as shown in the following table.

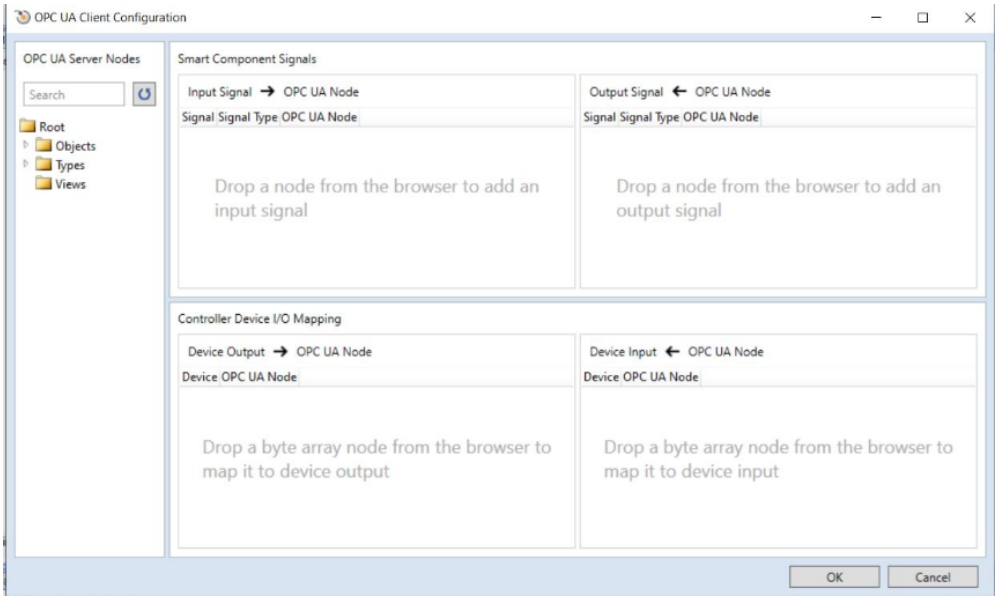
Type of station signal	OPC UA data type
Digital	*Boolean*
Analog	*Float*, *Double*

Continues on next page

Type of station signal	OPC UA data type
Group	*SByte*, *Byte*, *Int16*, *UInt16*, *Int32*, *UInt32*, *Int64* or *UInt64* **Note** A *Group* signal in RobotStudio is a 32 bit unsigned integer. A mismatch in bit sizes can result in overflow while reading or writing signals between RobotStudio and the OPC UA server.

OPC UA Client Configuration dialog

Use the OPC UA Client Configuration dialog to configure the mapping between nodes in the OPC UA Server and RobotStudio signals. In the **Station Logic** window, right-click the **OpcUaClient** Smart Component and then click **Configure...** to open this dialog.



xx2100000510

Item	Description
OPC UA Server Nodes browser	Displays the nodes from the connected OPC UA server in a tree structure. The **OpcUaClient** Smart Component must be connected to the OPC UA server to view the nodes.
Smart Component Signals	Station (Smart Component) signals can be mapped to nodes with data types *Boolean*, *Float*, *Double* or any integer type. In the **OPC UA Server Nodes** browser, hover over the node to view its details in the tooltip. Add the required OPC UA node for configuring to the **Input Signal** or **Output Signal** lists from the **OPC UA Server Nodes** browser using the drag-and-drop operation or by using the options **Map to input signal** or **Map to output signal** from the context menu.

Continues on next page

Item	Description
Controller Device I/O Mapping	The input/output of a device in the Robot Controller can be mapped to a byte array in the OPC UA Node here. In the **OPC UA Server Nodes** browser, hover over the node to view its details in the tooltip. Add the required OPC UA node for configuring to the **Device Output** or **Device Input** lists from the **OPC UA Server Nodes** browser using the drag-and-drop operation or by using the options **Map to controller device input** or **Map to controller device output** from the context menu.

Configuration file

The purpose of the configuration file (*.csv) is to export or import the mapping between the *OPC UA* Server nodes and RobotStudio signals. These files can be viewed with a text editor like Notepad or a spreadsheet program like Microsoft Excel. The configuration file stores data in a tabular format as follows.

Data field	Description
NamespaceUri	Specifies the *Namespace URI* of a Node on the OPC UA server.
IdentifierType	Specifies the type of the *Identifier* (any of the values *Numeric*, *String*, *Guid* or *Opaque*).
Identifier	The *Identifier* part of a *NodeId*.
ReadWrite	Specifies if the OPC UA Node value shall be read or written (possible values *Read* or *Write*). For station signals, the value *Read* creates an output signal on the Smart Component while *Write* creates an input signal. For robot signals, *Read* specifies that the data read from the OPC UA Node will be written to the robot input signals (to the device input memory area), while *Write* specifies that that robot output signals (from the device output memory area) will be read and written to the OPC UA Node.
Signal	The name of the signal that will be created in the Smart Component. Must be empty when mapping robot signals to an OPC UA Node.
SignalType	The type of the signal that will be created in the Smart Component (possible values *Digital*, *Analog* or *Group*). Must be empty when mapping robots signals to an OPC UA Node.
Controller	Specifies the name of the virtual controller. Must be empty for station signals.
Device	The name of a device on a bus in the virtual controller specified by Controller. Used to map an OPC UA Node to robot signals. Must be empty for station signals.

Properties of OPC UA Client Smart Component

Property	Description
Server address	Address of the OPC UA Server, for example, *opc.tcp://ipaddress:portnumber* • *ipaddress* refers to the IP address of the OPC UA server. • *portnumber* refers to the port number that identifies the OPC UA server.
Secure connection	The **Secure connection** check-box is selected by default and ensures that the OPC UA server certificate is trusted and that the connection is encrypted.

Continues on next page

Property	Description
Auto connect	Select this check-box to ensure automatic connection of the client to the specified **Server address** when loading the station and for reconnecting to the server after a connection failure.

Server/Client Certificates

The Smart Component generates a client certificate automatically. A server security certificate must be accepted when connecting the OPC UA Client Smart Component to an OPC UA Server. This process ensures a secure connection between the client and server. When **Secure connection** is enabled, the communication between the server and client will be encrypted using their certificates.

Note

The OPC UA Client Smart Component supports anonymous and username authentication when it connects to the OPC UA server, the credentials for username authentication can be stored and retrieved from the Windows Credentials Manager.

The OPC UA client certificate that is used to connect to the server is stored in: *%localappdata%\ABB\RobotStudio\OPC UA Certificates\own\certs*

Note

The OPC UA Server must be configured to trust the OPC UA client certificate.

Time synchronization

During simulation the signals are synchronized every time-step, that is, 24 milliseconds by default, which can be altered using the RobotStudio Option **Simulation:Clock**. When simulation is not running, the signals are synchronized every 500 milliseconds.

Workflow

1. Set up a PLC project in any simulation environment with an OPC UA server.
2. Connect the OPC UA Client *Smart Component* in RobotStudio.
   a. In RobotStudio, on the **Simulation** tab, in the **Configure** group, click **Station logic**.
   b. In the **Station Logic** window, click **Add component**, then select the **OpcUaClient** Smart Component under **PLC**.
   c. In the **Layout** browser, right-click **OpcUaClient**, and click **Properties**, enter the OPC UA Server URL in **Server address**.

      The **Secure connection** check-box will be selected by default. Select the **Auto connect** check-box to automatically connect the client to the specified **Server address** when loading the station or for reconnecting to the server after a connection failure.
   d. Connect to the OPC UA Server by right-clicking the Smart Component and click **Connect**.

Continues on next page

e Accept the security certificate to establish connection with the OPC UA Server.

3 Configure OPC UA nodes and RobotStudio signals in the **OPC UA Client Configuration** dialog.

4 Start simulation.

Coupling robot signals with OPC UA nodes

Prerequisites

This following example requires an active virtual controller with a PROFINET device with I/O memory area size of 64 bytes. This example also uses a B&R Automation Studio project with a simulated PLC and configured OPC UA server.

Procedure

1 In the **Configuration Editor** in RobotStudio, create I/O signals for the device, for example *RobotSignalOutput* and *RobotSignalInput*.

2 In **RobotStudio**, on the **Simulation** tab, in the **Configure** group, click **Station Logic**.

3 In **Station Logic**, click **Add component**, then select the **OpcUaClient** Smart Component under **PLC**.

4 Switch to B&R Automation Studio, add variables corresponding to the I/O signals, for example *RobotSignalOutput* and *RobotSignalInput*.

5 Add two byte array variables with the same size as the robot controller device's memory area, for example *RobotOutput* and *RobotInput* of type *USINT[0..63]*. .

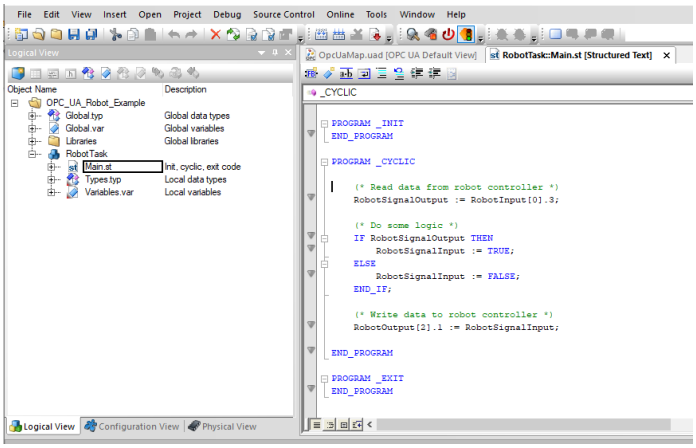
> **Note**
>
> Output signal from the the robot is the input to the PLC, so *RobotInput* must be of the same size as the device output memory area and *RobotOutput* must be of the same size as the device input memory area.

6 Configure *RobotInput* and *RobotOutput* variables to be available as nodes in the OPC UA server.

7 In the PLC program read from the *RobotInput* variable into the variables corresponding to the robot output signals, for example *RobotSignalOutput*.

Continues on next page

Write the variables corresponding to the robot input signals to the *RobotOutput* variable.



xx2100000238

8 Build the configuration in Automation Studio and transfer it to the PLC runtime, and start the runtime.

9 Switch to RobotStudio, right-click the **OpcUaClient** and click **Properties**. Enter the **Server address** and select **Auto connect** to connect to the OPC UA server.

10 In the **Layout** browser, right-click **OpcUaClient** and click **Configure...** to open the **OPC UA Client Configuration** dialog.

11 In the **OPC UA Server Nodes** browser, find the *RobotInput* node and drag it to the **Device Output**. In the **Device** drop-down, click the PROFINET device.

Similarly, drag the *RobotOutput* node to the **Device Input** and then click the PROFINET device in the **Device** drop-down.

On successful configuration, the signals will be continuously synchronized between PLC simulation in Automation Studio and virtual controller in RobotStudio.

Coupling Station signals

Prerequisites

This example requires a B&R Automation Studio project with a simulated PLC and configured OPC UA server.

Procedure

1 In RobotStudio, on the **Simulation** tab, in the **Configure** group, click **Station Logic**.

2 In the **Station Logic** window, click **Add component**, then select the **OpcUaClient** Smart Component under **PLC**.

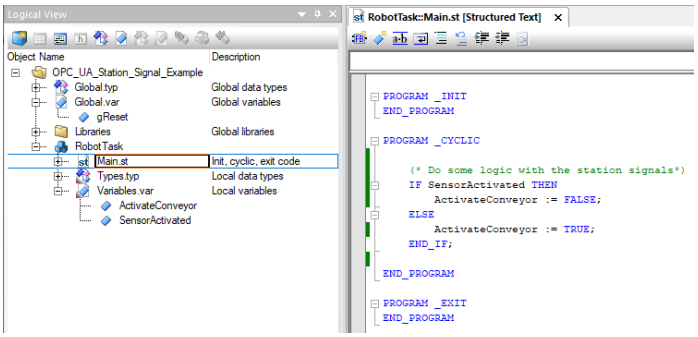
3 Right-click the **OpcUaClient** and click **Properties**. Enter the **Server address** and select **Auto connect** to connect to the OPC UA server.

4 Switch to B&R Automation Studio, add variables corresponding to the I/O signals as required, for example two Boolean variables *SensorActivated* and

Continues on next page

ActivateConveyor. Configure the variables to be available as nodes in the OPC UA server.

5 In the PLC program read and write from the variables as needed.



xx2100000244

6 Build the configuration in Automation Studio and transfer it to the PLC runtime. Start the runtime.

7 In the **Layout** browser, right-click **OpcUaClient** and click **Configure...** to open the **OPC UA Client Configuration** dialog.

8 In the **OPC UA Server Nodes** browser, find the *SensorActivated* node and drag it to the **Input Signal** list. Similarly, drag the *ActivateConveyor* node to the **Output Signal** list. Click **OK**, the new signals appear on the **OpcUaClient** Smart Component.

9 Connect the I/O signals on the OPC UA Client Smart Component to other Smart Components in RobotStudio. On successful connection, the I/O signals are continuously synchronized between the PLC simulation and RobotStudio.

Exporting and importing configuration file to the Smart Component

An **OpcUaClient** configuration can be exported from a station and imported to other stations.

- To export a configuration file, in the **Layout** browser, right-click the **OpcUaClient** Smart Component and click **Export configuration...** and save the configuration file.
- To import a configuration file, in the **Layout** browser, right-click the **OpcUaClient** Smart Component and click **Import configuration...**, browse to the required folder and select the configuration file.

  The signals from the configuration file will be displayed as inputs and outputs on the **OpcUaClient** Smart Component.

> **Note**
>
> The imported configuration file overwrites the existing configuration.

This page is intentionally left blank

8 Deploying and Sharing

8.1 Saving and loading RAPID programs and modules

Overview

One of the purposes of creating simulations in RobotStudio is to create a robot program that can be transferred to the *robot controller*. RAPID programs are stored in the *virtual controllers* of the respective *station*. You can save entire programs or specific modules.

These programs can be shared with destination controllers in three different ways, save program as a file in the host computer and transfer these files to the destination controllers, create a backup and then restore the file in the destination controller or transfer the file by using the transfer function.

8.2 Sharing a station

Overview

Use the **Pack & Go** feature to create a package (*.rspag) of an active station. The **Unpack & Work** feature can be used to unpack the *Pack & Go* file on another computer. The Pack & Go file excludes *RobotWare*, but RobotWare *add-ins* that are part of the station are included in the Pack&Go file. The required RobotWare must be installed in the target computer, but any required RobotWare add-ins are included/distributed with the Pack&Go file.

The *Pack & Go* format is the recommended mode for sharing the RobotStudio *stations*.

Packing a station

1. On the **File** tab, under **Share**, click **Pack and Go**. The **Pack & Go** dialog opens.
2. Enter the name of the package and then browse and select the location of the package.
3. Optionally, select the **Password protect the package** check box.
4. Type in the password in the **Password** box to protect your package.
5. Click **OK**.

Unpacking a station

1. On the **File** tab, click **Open** and browse the folder and select the *Pack&Go* file, the **Unpack & Work** wizard opens.
2. In the **Welcome to the Unpack & Work Wizard** page, click **Next**.

   Note

   A password-protected Pack&Go file asks for a password for loading the station.

3. In the **Select package** page, click **Browse** and then select the **Pack & Go file to unpack** and the **Target folder**. Click **Next**.
4. In the **Library handling** page select the target library. Two options are available, **Load files from local PC** or **Load files from Pack & Go**. Click the option to select the location for loading the required files, and click **Next**.

   Note

   The station file in the Pack& Go uses library file (*.rslib). If the PC has a later version of the library file, then select the option **Load files from local PC** for loading the PC version of the library file.

5. In the **Virtual Controller** page, select the **RobotWare version** and then click **Locations** to access the **RobotWare Add-in** and Media pool folders. Optionally, select the check box to automatically restore backup. Click **Next**.
6. In the **Ready to unpack** page, review the information and then click **Finish**.

Continues on next page

7 In the **Unpack & Work finished** page, review the results and then click **Close**.

Note

During an unpack operation, if you select the **Copy configuration files to SYSPAR folder** option, the configuration (CFG) files of the back up virtual controller inside the *Pack& Go* file gets copied to the SYSPAR folder of the new *virtual controller*. This is to avoid any loss of configuration data during an I- start. Select this option for complex configurations like the paint system.

8.3 Capturing the selected screen

Overview

Screen capture entails two functions useful for demonstrations and training purposes:

- The Screenshot function which allows you to capture an image of the application.
- The Record Movie function which allows you to make a recording of your work in RobotStudio, either of the entire GUI or just the **Graphics** window.

Taking screenshots

Use Screenshot feature to capture an image of the entire application or an active document window such as the **Graphics** window. Screenshot can be taken by using the keyboard shortcut *CTRL + B* or by using the **Screenshot** button on the Quick Access Toolbar.

To enable the Screenshot command in the Quick Access toolbar:

1. Click the **Quick Access Toolbar**, and then click **Customize Commands** from the drop down menu. Options:General:Screenshot.

   The Customize Commands and Quick Access Toolbar dialog opens.
2. In the **Show Commands from** box , select **Other Commands** and then select **Screenshot**.
3. Click the **Add>>** button to add this option to the **Quick Access Toolbar** and then click **Close**.

   The Screenshot icon gets added to the toolbar.
4. On the Quick Access Toolbar, click **Screenshot**, the screenshot gets saved to `C:\Users\<user name>\Pictures` folder.

Recording movies

Overview

RobotStudio provides three options for recording movies, **Record Simulation**, **Record Application** and **Record Graphics**. The **Record Application** option records the entire user interface of RobotStudio including the curser and mouse-clicks of the simulation. These recordings are used for demonstrating certain features. The **Record Simulation** option is similar to the **Record Application**, but records in virtual time.

Recording the screen

1. On the **Simulation** tab, in the **Record Movie** group, click **Record application** to capture the entire application window, or **Record graphics** to capture just the **Graphics** window.
2. After recording, click **Stop Recording**.
3. Click **View Recording** to playback the latest capture.

8.4 Recording a simulation

Records the graphics view only and it uses the virtual time for recording the frames. The advantage here is that the recording will not be affected by the PC configuration or the load on the PC at the time of recording as the frames are recorded with the virtual timestamp.

1. In the **Record Movie** group, click **Record Simulation** to record the next simulation to a video clip.
2. After recording, click **Stop Recording**.

   The simulation is saved in a default location which is displayed in the **Output** window.
3. Click **View Recording** to playback the recording.

8.5 Creating a 3D animation of your simulation

Overview

The Project Viewer can playback a station in 3D on computers that do not have RobotStudio installed. It packages the station file together with files needed to view the station in 3D. It can also play recorded simulations.

Prerequisites

These prerequisites are applicable only when the project viewer is saved in **.exe* format.

- Visual C++ Redistributable for Visual Studio 2015-2022.
- .NET Framework 4.8 must be installed on the playback computer.

Creating and loading a project viewer

Use the following steps to create and save the project viewer as a file to the local PC and then open this file in a 3D Viewer.

1
- To create a project viewer without simulation.

  On the **File** tab, click **Share** and then click **Export Viewer** .
- To create a project viewer with simulation.

  On the **Simulation** tab, in the Simulation Control group, click **Export Viewer**.

2 The **Export Viewer** dialog opens. In the **Location** group, select **This PC**.

3 Click the **...** button to browse and select a folder for saving the project.

4 Select the file format from the **Save as type** list. This project viewer can be saved in **.exe*,**.glb* (gITF file) or *.rsstnv (Station Viewer file).

5 Click **Create**.

Simulation control buttons are enabled when the Project Viewer contains a recorded simulation. When a simulation starts, the movements and visibility of objects are recorded. This recording can be included in the Project Viewer.

6 To load Project Viewer, double-click the package file. The results are displayed in the **Output** window and the embedded project file is automatically loaded and presented in a 3D viewer.

> **Note**
>
> The project viewer inherits the default settings of RobotStudio, to customize these settings on the **File** tab, click **Options** to edit the settings.

8.6 Deploying a RAPID program to a robot controller

Overview

Use transfer function to transfer *RAPID* programs that are created *offline* in a virtual controller to a robot controller in the shop floor or to another virtual controller. As part of the transfer function, you can also compare the data present in the *virtual controller* with that present in the *robot controller* and then select the data to be transferred.

Relations for transfer of data

To transfer data, a **Relation** must be defined between the two controllers. A Relation defines the rules for transferring data between two controllers.

Creating a Relation

When two controllers are listed in the Controller browser, a Relation can be created between them. To create a Relation:

1. On the **Controller** tab, in the **Transfer** group, click **Create Relation**.

   The **Create Relation** dialog box opens.
2. Enter a **Relation Name** for the relation.
3. Specify the **First Controller**, from the list.

   The First Controller, also called the Source, owns the data being transferred.
4. Specify the **Second Controller**, from the list. This can either be a *robot controller* or another *virtual controller*.

   The Second Controller, also called the Target, receives the data being transferred.
5. Click **OK**.

   The relation between the controllers is now created.

The *Relation* dialog box opens, here you can configure and execute the transfer. Relations of a controller are listed under the Relations Node in the Controller browser.

> **Note**
>
> The properties of the relation are saved in a XML file under INTERNAL in the system folder of the first controller (source).

Transferring data

You can configure the details of the transfer of data and also execute the transfer, in the *Relation* dialog box.

Continues on next page

To open the *Relation* dialog box, double-click a relation. Alternatively, select a relation in the **Controller** browser, and then in the **Transfer** group, click **Open Relation**.

Configuring the transfer

Before executing a transfer, configure the data to be transferred, under the *Transfer Configuration* heading. Use the following guidelines to configure transfer.

- Use the check boxes in the *Included* column to include or exclude the corresponding items shown in the tree structure. All items in a module that are included will be transferred. Other non-listed items of a module such as comments, records and so on will be automatically included in the transfer.
- The *Action* column shows a preview of the result of the transfer operation, based on the items included or excluded.
- If a module exists both in the source and the target controllers, and the *Action* column shows *Update*, then click **Compare** in the Analyze column. This opens the *Compare* box which shows two versions of the *module* in different panes. The affected lines are highlighted and you can also step through the changes. You can choose one of the following options for the comparison:
  - **Source with target** - Compares the source module with the target module
  - **Source with result** - Compares the source module with the module that will be the result of the transfer operation
- BASE (module), wobjdata and tooldata are excluded by default.
- wobjdata wobj0, tooldata tool0, and loaddata load0 of the BASE module are unavailable for inclusion.

A task can be transferred only if:

- Write access to the target controller is present (must be manually retrieved).
- Tasks are not running.
- Program execution is in the stopped state.

Executing the transfer

Under the *Transfer* heading, the Source and Target *modules* are shown along with the arrow showing the direction of the transfer. You can change the direction of the transfer by clicking **Change Direction**. This also switches the source and target modules.

To execute the transfer, click **Transfer now**. A dialog showing a summary of the transfer appears. Click **Yes** to complete the transfer. The result of the transfer is displayed for each module in the **Output** window.

The **Transfer now** button is disabled if:

- None of the included *tasks* can be transferred.
- Write access is required but not held.

Continues on next page

Note

If one of the several modules fail, then the following error message is displayed.

```
Module xxx.zzz has failed. Do you want to continue?
```

This page is intentionally left blank

9 Installing robot controller software

9.1 Installation

9.1.1 About Installation

Overview

This section describes how to create, modify, and copy systems to run on *robot* and *virtual controllers* using the Installation feature.

> **Note**
>
> Use Modify Installation to create and modify systems with *RobotWare* versions 7. Use Installation Manager 6 to create and modify systems with RobotWare versions 6.0. Use System Builder to create and modify systems based on earlier versions of RobotWare.

About real and virtual systems and license files

When using real license files to create a robot controller, the license files contain the options that the user has ordered, and in most cases no additional configuration is required. Real *license files* can both be used to create *robot* and *virtual controllers*.

When using virtual license files to create a virtual controller, all options and robot models are available, which is useful for evaluation purposes, but requires more configuration while creating the virtual controller. Virtual license files can only be used to create virtual controllers.

Products

Product is the collective name for the different software such as RobotWare, RobotWare add-ins, third party software and so on. Products are either free or licensed, licensed products require a valid license file.

Deployment packages and the repository

Modify Installation can be used to create an installation or update package offline on a USB-stick, which later can be installed from the FlexPendant. The repository is the storage where all files needed to create and modify *RobotWare systems* are placed.

Prerequisites

The following are the prerequisites for creating a system:

- A *RobotWare license* file for the system, when creating a system to run on a robot controller. The license file is delivered with the controller.
- A virtual license file for creating a system for virtual use. All *products* are delivered with a virtual license file.
- Installing on a *robot controller* requires a connection from the computer to the service or Ethernet port of the controller.

9.1.2 Using Modify Installation for RobotWare 7

Updating an existing RobotWare system

Description

The most frequent RobotWare system update use case is updating one or more software, for example, RobotWare and add-ins. This is a frequent operation during the commissioning time, especially on large installations.

> **Note**
>
> To perform a RobotWare system update, the controller must be in the RobotWare system mode.

System update changes the configuration of the currently installed RobotWare system. There are different types of configuration changes, such as:

- Adding or removing licenses
- Upgrading, removing installed software or adding new software
- Activating or deactivating optional features

Before performing a system update, it is recommended to:

- create a backup of the system (user data) and store it on an external storage media.
- create a snapshot of the current system state.

Upgrading a software in the RobotWare system

The following procedure provides the steps involved during the update of the RobotWare system.

> **CAUTION**
>
> Do not turn off the controller while system update is in progress. Doing this may in worst case lead to data corruption in the RobotWare system, in which case it needs to be reinstalled.

1. Access the **Modify Installation** view in RobotStudio.
2. Select **Software > Included**.
3. The **Included Software** window displays the software that is included in the current RobotWare system.
4. Select the product that should be upgraded and tap **Update**.
5. In the **Update Software** window, select the software version to be used and tap **OK**.
6. The **Summary** tab shows an overview of all the changes.

Continues on next page

7 Continue to modify the system, or select **Apply**/**Apply and reset** to confirm and save the changes.

> **Note**
>
> The **Modify Installation** dialog will be closed during the controller update.
>
> When the update process is finished, check the event log for information about the update results. A successful update will be indicated in the event log, and if the update has failed, one or more error logs will be generated.

Adding/removing software

The following procedure provides the steps involved during the update of the RobotWare system.

> **CAUTION**
>
> Do not turn off the controller while system update is in progress. Doing this may in worst case lead to data corruption in the RobotWare system, in which case it needs to be reinstalled.

1 Access the **Modify Installation** view in RobotStudio.
2 Select **Software** > **Included**.
3 The **Included Software** window displays the software that is included in the current RobotWare system. Select one of the following:
   - Select the product box for the software that should be added to the system.
   - Deselect the product box to remove the product from the system.

   > **Note**
   >
   > Products may have dependences to certain versions of other products. A product may only be removed if all products that are dependent on it are removed as well.

4 The **Summary** tab shows an overview of all the changes.
5 Continue to modify the system, or select **Apply**/**Apply and reset** to confirm and save the changes.

> **Note**
>
> The **Modify Installation** dialog will be closed during the controller update.
>
> When the update process is finished, check the event log for information about the update results. A successful update will be indicated in the event log, and if the update has failed, one or more error logs will be generated.

Continues on next page

9.1.2 Using Modify Installation for RobotWare 7
Continued

Adding/removing add-in packages

The following procedure provides the steps involved during the update of the RobotWare system.

> **CAUTION**
>
> Do not turn off the controller while system update is in progress. Doing this may in worst case lead to data corruption in the RobotWare system, in which case it needs to be reinstalled or recovered from a snapshot.

1. Access the **Modify Installation** view in RobotStudio.
2. Select one of the following:
   - To add add-in packages, select **Software** > **Available** and tap **Include**.
   - To remove add-in packages, select **Software** > **Included** and tap **Remove**.

     > **Note**
     >
     > Products may have dependences to certain versions of other products. A product may only be removed if all products that are dependent on it are removed as well.

     > **Note**
     >
     > RobotWare is mandatory and cannot be removed from the system.
3. The **Summary** tab shows an overview of all the changes.
4. Continue to modify the system, or select **Apply**/**Apply and reset** to confirm and save the changes.

   > **Note**
   >
   > The **Modify Installation** dialog will be closed during the controller update.
   >
   > When the update process is finished, check the event log for information about the update results. A successful update will be indicated in the event log, and if the update has failed, one or more error logs will be generated.

Changing the software installation order when adding/removing RobotWare add-ins

When adding and removing RobotWare add-ins to/from the system, sometimes it is necessary to manually adjust the installation and initialization order or the included add-ins.

1. Access the **Modify Installation** view in RobotStudio.
2. Select **Software** > **Included**.
3. In the **Included Software** window, tap the **Installation order** button to open the **Change Installation Order** window. Select a product and use the up and down arrows to change the installation order. Select **Done**.
4. The **Summary** tab indicates that the installation order has been updated.

Continues on next page

5 Continue to modify the system, or select **Apply**/**Apply and reset** to confirm and save the changes.

> **Note**
>
> The **Modify Installation** dialog will be closed during the controller update.
>
> When the update process is finished, check the event log for information about the update results. A successful update will be indicated in the event log, and if the update has failed, one or more error logs will be generated.

Working with option selections

Overview

The following categories of system features can be updated:

- System options
- Controllers
- Robots
- FlexPendant

> **Note**
>
> Some features extend, showing more options upon selection. For example, in the group controller variant, you get the option of choosing variant type only when a controller first is selected. The additional drive units work similarly, some are unavailable until you select a different drive system type. This means options can be locked behind selections.

Turning options on/off

> **CAUTION**
>
> Do not turn off the controller while system update is in progress. Doing this may in worst case lead to data corruption in the RobotWare system, in which case it needs to be reinstalled.

1 Access the **Modify Installation** view in RobotStudio.
2 Select the tab **Options**.
3 Select the option category to be updated, and the corresponding **Options** that should be activated/deactivated for the system.

> **Note**
>
> Linked options will be selected automatically.
>
> Conflicting options cannot be selected.

4 The **Summary** tab shows an overview of all the changes.

Continues on next page

5 Continue to modify the system, or select **Apply**/**Apply and reset** to confirm and save the changes.

> **Note**
>
> The **Modify Installation** dialog will be closed during the controller update.
>
> When the update process is finished, check the event log for information about the update results. A successful update will be indicated in the event log, and if the update has failed, one or more error logs will be generated.

Adding licenses to enable additional option access

> **CAUTION**
>
> Do not turn off the controller while system update is in progress. Doing this may in worst case lead to data corruption in the RobotWare system, in which case it needs to be reinstalled.

1 Access the **Modify Installation** view in RobotStudio.
2 Select the tab **Options**.
3 Select **Edit** to access the **Edit License files** window. Select one of the following:
   - Select **Add** to browse for a new license to be added.
   - Select an existing license and tap **Remove**.
4 The **Summary** tab shows an overview of all changes.
5 Continue to modify the system, or select **Apply**/**Apply and reset** to confirm and save the changes.

> **Note**
>
> The **Modify Installation** dialog will be closed during the controller update.
>
> When the update process is finished, check the event log for information about the update results. A successful update will be indicated in the event log, and if the update has failed, one or more error logs will be generated.

Exporting and importing option selections

> **CAUTION**
>
> Do not turn off the controller while system update is in progress. Doing this may in worst case lead to data corruption in the RobotWare system, in which case it needs to be reinstalled.

1 Access the **Modify Installation** view in RobotStudio.
2 Select the tab **Options**.

Continues on next page

3 Select one of the following:
   - Select **Export** and browse to the location where the exported option selections should be saved. Select **Save**.

     The current option selections will be saved to an RSF file that can be imported or added to other systems.
   - Select **Import** and browse to the location of the configuration file, and then select **Open**.

     Note

     All current selections will first be cleared.
   - Select **Add** and browse to the location of the configuration file, and then select **Open**.

     Note

     Existing selections are kept, and options that are not currently selected will be added.

4 Continue to modify the system, or select **Apply**/**Apply and reset** to confirm and save the changes.

Note

The **Modify Installation** dialog will be closed during the controller update.

When the update process is finished, check the event log for information about the update results. A successful update will be indicated in the event log, and if the update has failed, one or more error logs will be generated.

Drive system types

The following matrix describes the existing drive system types and some examples of compatible products:

Product		Power									
Manipulator	**Controller**	2.5kVA-310V	2.5kVA-370V	3.0kVA-370V	7.0kVA-370V	3.0kVA-370V	480VA-24V	1.2kVA-48V	1.5kVA-48V	13kVA-650V	7.5kVA-650V
IRB 1600 or smaller	C30	A1									
	C90XT										
	E10		B2								
IRB 14050	C30						C6				
CRB 15000	C30							D7			
CRB 15000	C30								D10		
IRB 2600	V250XT				E4	E5					
	V400XT										

Continues on next page

9.1.2 Using Modify Installation for RobotWare 7
Continued

Product		Power									
IRB 4600 or larger	V250XT									E8	E9
	V400XT										

Installing a new RobotWare system

Description

Before installing a new RobotWare system on the controller, it is required to:

- create a virtual controller.
- create an installation package.

Create a virtual controller

1 Start RobotStudio.
2 Select **Add Controller** > **Connect to Controller** in the **Controller** ribbon.
3 In the **Connect to Controller** window, select the **Virtual Controllers** tab.
4 Select **New Controller**.
5 In the **New Virtual Controller** dialog, select option **Create New** and complete the following:
   - **Name**

     Give the new system a valid name. If you enter an invalid name you will not be able to proceed.

     Note

     The system name can contain between 1 to 55 characters. Allowed characters are "A–Z", "a–z", "0–9", and "-" (hyphen). Hyphen "-" is only allowed between characters.

   - **Location**
   - **Robot model**
   - **RobotWare**
   - **Controller**

Note

Selecting option **Create from backup** can be used to create a system based on the configuration found in the selected Backup. This means that the same set of SW products (RobotWare and add-ins), licenses and options will be used.

Note, however that the software referred to by the Backup is not included in the Backup itself and must be previously downloaded to your computer by using the RobotStudio **Add-Ins** page.

Note also that this procedure will not automatically include RAPID programs and system parameters to your new system. If needed, they can be loaded to the new system by restoring the Backup once the new system is installed and started.

Continues on next page

6 Select **OK** to continue.
7 Continue with creating an installation package.

Creating a new installation package

Overview

The installation package is a software package that consists of predefined directory structure and number of files, used for purpose of re-deploying RobotWare System on a robot controller. The installation package is created in RobotStudio and is deployed on the controller using RobotWare Installation Utilities on the FlexPendant.

RobotWare Installation Utilities is a small package of installation related utilities that is always present on each robot controller and cannot be removed. It is used to deploy and re-deploy RobotWare system which is the operating system of the robot controller. When in RobotWare Installation Utilities mode, the robot cannot be moved using the FlexPendant and robot programs cannot be written or executed.

Prerequisites

The following prerequisites must be met before you can start creating an installation package:

- Latest version of RobotStudio must be installed.
- License files for products to be installed must be available. Licenses are included in the RobotWare system at purchase, but can also be retrieved from a backup of the RobotWare system currently deployed on the controller, or exported from the controller via RobotWare Installation Utilities.

  Note

  Virtual licenses can also be used.

- Product versions to be installed must be available in RobotStudio or in a custom location.

  These versions can be made available by selecting a RobotWare distribution package (.rspak file) from RobotStudio (tap **Install Package** in the **Add-Ins** tab). All products that are installed this way, have matched versions and correct dependencies to each other.
- A virtual controller must be created.

Create installation package

1 Start RobotStudio.
2 Select **Add Controller** > **Connect to Controller** in the **Controller** ribbon.
3 In the **Connect to Controller** window, select the controller and tap **OK**.
4 Request write access.
5 Launch the **Modify Installation** dialog from the **Controller** ribbon.
6 Select the tab **Software**.

Continues on next page

7 Select **Create Package** to create an installation package based on the virtual controller configuration.

> **Note**
>
> If the virtual system has been built using virtual licenses, these will not be included in the installation package.
>
> If virtual licenses are used, the selected feature configuration will be matched against the real licenses present in the controller and the installation will stop if some licenses are missing. This situation can be avoided if real licenses from the controller are exported and imported into the virtual system when it is built.

8 In the **Create Installation Package** dialog, define the following:

- **Package Name**

  Enter a name for the installation package.
- **Location**

  Browse and select the output folder (for example, a USB-stick) for the installation package.

Select **OK**.

9 The window **Installation Package created** is displayed. The installation package for the selected system has been created. Select **OK**.

10 Continue with installing the package on the controller.

9.1.3 Using Installation Manager for RobotWare 6

9.1.3.1 Startup and settings

Starting Installation Manager

On the **Controller** tab, in the **Configuration** group click **Modify Installation** to start the **Installation Manager** application.

This window provides two options. Select **Network** to create systems for *robot controllers* and **Virtual** to create systems for *virtual controllers*.

Setting application preferences

Before creating a system using Installation Manager, it is recommended to set the path to the desired location where *products*, *licenses*, and backups are located and where virtual systems will be created.

1. On the **Controller** tab, in the **Configuration** group click **Modify Installation**.
2. In the **Installation Manager** window, click **Preferences**. The **Preferences** window opens.
3. Browse and select folders for **Product path(s)**, **License path(s)**, **Virtual systems path(s)**, and **Backup path(s)** in respective lists.

   The **User name** and **Password** boxes are populated with the default credentials provided to you with your RobotStudio license. These credentials are applicable only for a robot controller.
4. In the **Default System Name** box, enter the default system name. When you create a new system, this name will be assigned by default.
5. Click **OK** to set the preferences.

Advanced users can select the location to install virtual controllers. Select the **Virtual products installation path(s)** check box and then click **Browse** to select the folder. You can see the selected path in the drop-down list box. Clear the check box to enable the default path.

Settings file

Settings file contains the selected options. When Installation Manager connects to the robot controller, it reads options from the *Settings* file. Any change in the options are mapped in the file.

Use the **Settings** buttons to perform the following tasks with the settings file.

Buttons	Description
Export settings	Click this button to export the current settings of the robot controller.
Import settings	Click this button to import settings to the robot controller. The current settings of the system will be cleared before performing this operation.
Add settings	Click this button to add settings to the current setup of the robot controller.
Revert	Click this button to revert to the current settings of the robot controller.

9.1.3.2 Building a new robot controller

Creating a new system for a robot controller

1. In the **Installation Manager** window, click **Controllers**, and then click the **Network** tab.

   The **Network** tab shows the name and IP address of all the available controllers on the LAN network and/or any controller attached through the service port.
2. Select your controller in the controller list and click **Open**. Installation Manager fetches information from the controller.
3. Click **New**. The **Create New** pane appears.
4. In the **Create New** pane, in the **Name** box, enter the name of the new system.
5. Click **Next**. The **Products** tab gets selected.
6. Click **Add**, the **Select Product** window opens. Select the product manifest file and click **OK**.

   If you want to add more products such as Add-ins, click **Add** again and select the product. To find a product that is not in the list, click **Browse** and then select the file from the particular folder.

   > **Note**
   >
   > The product install order shows the order in which products and Add-ins are installed on the controller. Installing products in the required sequence is important when products are dependent on each other. The product order is assigned to products automatically based on the sequence in which the products must be installed on the controller. This is independent of the order in which products are added to the system.
7. Click **Next**. The **Licenses** tab gets selected.
8. Click **Add**, the **Select License** window opens. Select the license file and click **OK**.

   Repeat the step to add multiple license files to your system.
9. Click **Next**, the **Options** tab gets selected. This pane shows the **System Options**, **Drive Modules** and **Applications**. Here you are able to customize your options.
10. Click **Next**, the **Confirmation** tab gets selected and shows an overview of the system options.
11. Click **Apply**, the system gets created on the controller.

Once the installation completes, a **Restart Controller** dialog appears, click **Yes** to restart the controller. Click **No** to manually restart controller later, the controller

Continues on next page

stores the new system or the changed system and these changes will take effect during the next restart.

> **Note**
>
> Changing the *RobotWare* version needs *BootServer* update. Controller must be restarted for BootServer update. Hence, the controller will be restarted automatically when you change the RobotWare.

Creating a new system for a virtual controller

1. In the **Installation Manager** window, click **Controllers**, and then click the **Virtual** tab.
2. Click **New**. The **Create New** pane opens.
3. In the **Create New** pane, in the **Name** box, enter the name of the new system.
4. Click **Next**. The **Products** tab gets selected.
5. Click **Add**, the **Select Product** window opens. Select the product manifest file and click **OK**.

   If you want to add more products such as *Add-ins*, click **Add** again and select the product. To find a product that is not in the list, click **Browse** and then select the file from the particular folder.

   > **Note**
   >
   > The product install order shows the order in which products and add-ins are installed on the controller. Installing products in the required sequence is important when products are dependent on each other. The product order is assigned to products automatically based on the sequence in which the products must be installed on the controller. This is independent of the order in which products are added to the system.

6. Click **Next**. The **Licenses** tab gets selected.
7. Click **Add**, the **Select License** window opens. Select the license file and click **OK**.

   Repeat the same step to add multiple license files to your system.
8. Click **Next**, the **Options** tab gets selected. This pane shows the **System Options**, **Drive Modules** and **Applications**. Here you are able to customize your options.
9. Click **Next**, the **Confirmation** tab gets selected and shows an overview of the system options.
10. Click **Apply**, the system gets created.

9.1.3.3 Modifying a robot controller

Modifying a system for a robot controller

1. In the **Installation Manager** window, select **Controllers** and then select the **Network** tab.
2. Select your controller in the controller list and click **Open**. Installation Manager fetches information from the controller.
3. Select the particular system that you want to modify.

   The **Overview** pane displays the system options of the selected system.

   > **Note**
   >
   > To be able to modify a system it must first be activated. Select the system you want to modify and press the activate button.

4. Click **Next**. The **Products** tab gets selected. All products and add-ins that were part of the selected system will be displayed here.
   - To upgrade/downgrade a product select the product and click **Replace**.
   - To remove a product select the product and click **Remove**.

     > **Note**
     >
     > It is not possible to remove the RobotWare product.

5. Click **Next**. The **Licence** tab gets selected. The license details of the selected system will be displayed here. Here you are able to add/remove licenses.
6. Click **Next**. The **Options** tab gets selected. Here you are able to customize your options.
7. Click **Next**, the **Confirmation** tab gets selected and shows an overview of the system options.
8. Click **Apply** for the changes to take place.

Once the installation completes, a **Restart Controller** dialog appears, click **Yes** to restart the controller. Click **No** to manually restart controller later, the controller stores the new system or changed system and these changes will take effect during the next restart.

> **Note**
>
> Any change in the *RobotWare* version needs a *BootServer* update, which requires a controller restart. The controller will be restarted automatically.

Modifying a system for virtual controller

1. In the **Installation Manager** window, select **Controllers** and then select the **Virtual**. Here you are able to view the list of all virtual systems.
2. Select the particular system that you want to modify.

   The **Overview** pane will display the system options of the selected system.

Continues on next page

3 Click **Next**. The **Products** tab gets selected. All products and add-ins that were part of the selected system will be displayed here.
   - To upgrade/downgrade a product select the product and click **Replace**.
   - To remove a product select the product and click **Remove**.

   Note

   It is not possible to remove the *RobotWare* product.

4 Click **Next**. The **Licence** tab opens. The license details of the selected system will be displayed here. Here you are able to add/remove licenses.
5 Click **Next**. The **Options** tab opens. This pane shows the **System Options**, **Drive Modules** and **Applications**. Here you are able to customize your options.
6 Click **Next**, the **Confirmation** tab gets selected and shows an overview of the system options.
7 Click **Apply** for the changes to take place.

Deleting a system from a robot controller

1 In the **Installation Manager** window, select **Controllers** and then select the **Network** tab.
2 Select your controller in the controller list and click **Open**. Installation Manager fetches information from the controller.
3 Select the system you want to delete and then click **Remove**.

   The selected system gets deleted.

Note

The active system cannot be deleted. First activate a different system, or deactivate the system by using the restart mode **Start Boot Application**.

Deleting a system from virtual controller

1 In the **Installation Manager** window, select **Controllers** and then select the **Virtual** or **Network** tab. Here you are able to view the list of all virtual systems.
2 Select the system you want to delete and then click **Remove**.

   The selected system gets deleted.

Note

When you create a system for *virtual controller*, corresponding products are installed in the user APPDATA folder, and many virtual controller systems point to these *products*. Hence, products are not deleted while deleting a system from virtual controller.

9.1.3.4 Copying a robot controller

Copying a system from a virtual controller to a robot controller

To be able to copy a *virtual controller* to a *robot controller*, the virtual controller must have been created with real license files.

1. In the **Installation Manager** window, select **Controllers** and then select the **Network** tab.
2. Select your controller in the controller list and click **Modify**. Installation Manager fetches information from the controller.
3. Click **New**. The **Create New** pane opens.
4. Enter the name of the virtual controller in the **Name** box, and then click the **virtual system** option under **Create new from**.
5. Click **Select**, the **Select Virtual System** pane opens, select the particular system and then click **OK**.
6. Click **Next**. The **Products** tab gets selected. All products and add-ins that were part of the selected virtual controller will be displayed here.
7. Click **Next**. The **Licenses** tab gets selected. The license details of the selected virtual controller will be displayed here.
8. Click **Next.** The **Options** tab gets selected. Here you are able to select/deselect options to customize them.
9. Click **Next**, the **Confirmation** tab gets selected and shows an overview of the system options.
10. Click **Apply**, the system gets created.

Once the installation completes, a **Restart Controller** dialog appears, click **Yes** to restart the controller. Click **No** to manually restart controller later, the controller stores the new or changed virtual controller and these changes will take effect during the next restart.

Copying a virtual controller

1. In the **Installation Manager** window, select **Controllers** and then select the **Virtual** tab. Here you are able to view the list of all virtual systems.
2. Click **New**, the **Create New** pane opens.
3. Enter the name of the virtual system in the **Name** box, and then click the **virtual system** option under **Create**.
4. Click **Select**, the **Select Virtual System** pane opens, select the particular system and then click **OK**.
5. Click **Next**. The **Products** tab gets selected. All products and add-ins that were part of the selected system will be displayed here.
6. Click **Next**. The **Licence** tab gets selected. The license details of the selected system will be displayed here.
7. Click **Next**. The **Options** tab gets selected. Here you are able to select/deselect options to customize them.
8. Click **Next**, the **Confirmation** tab gets selected and shows an overview of the system options.

Continues on next page

9 Click **Apply** for the changes to take place.

9.1.3.5 Creating a robot controller from backup

Creating a robot controller from backup for a robot controller

1. In the **Installation Manager** window, select **Controllers** and then select **Network** tab.
2. Select your controller in the controller list and click **Open**. Installation Manager fetches information from the controller.
3. Click **New**, the **Create New** pane opens.
4. Enter the name of the robot controller in the **Name** box, and then click **backup** option under **Create**.
5. Click **Select**, the **Select Backup** pane opens, you can select the particular backup system and then click **OK**. If the right RobotWare already exists, then the version will be selected.

   Note

   In the folder hierarchy of the backup path, the name of the valid backup system folder that must be selected will be set in bold. Similarly, names of all valid backup systems will be marked in bold in the folder hierarchy. User must select one of the valid backup folders for further system creation.

6. Click **Next**. The **Products** tab gets selected. All products and add-ins that were part of the backup will be displayed here. You can add new/other product(s) and/or replace RobotWare version if needed.
7. Click **Next**. The **Licenses** tab gets selected. The license details of the backup will be displayed here. You are able to view the license(s) from the backup. Here you can add new/more licenses.
8. Click **Next** the **Options** tab gets selected, select/deselect options to customize them.
9. Click **Next**, the **Confirmation** tab gets selected and shows an overview of the system options.
10. Click **Apply**, the system gets created.

    Once the installation completes, a **Restart Controller** dialog appears, click **Yes** to restart the controller. Click **No** to manually restart controller later, the controller stores the new system or changed system and these changes will take effect during the next restart.

Creating a robot controller from backup for a virtual controller

1. In the **Installation Manager** window, select **Controllers** and then select the **Virtual** tab. Here you are able to view the list of all virtual systems.
2. Click **New**, the **Create New** pane opens.
3. Enter the name of the system in the **Name** box, and then click **backup** option under **Create**.
4. Click **Select**, the **Select Backup** pane opens, you can select the particular backup system and then click **OK**.

Continues on next page

If the right RobotWare already exists, then the version will be selected. If the RobotWare does not exist, click **Replace** to select the RobotWare.

> **Note**
>
> In the folder hierarchy of the backup path, the name of the valid backup system folder that must be selected will be set in bold. Similarly, names of all valid backups will be marked in bold in the folder hierarchy. User must select one of the valid backup folders for further system creation.

5 Click **Next**. The **Products** tab gets selected.

All *products* and *add-ins* that were part of the backups will be displayed here. You can add new/other product(s) and/or replace RobotWare if needed.

6 Click **Next**. The **Licenses** tab gets selected. Here you are able to view the license details of the backup.

7 Click **Next**, the **Options** tab gets selected, select/deselect options to customize them.

8 Click **Next**, the **Confirmation** tab gets selected and shows an overview of the system options.

9 Click **Apply**, the virtual controller gets created.

9.1.3.6 Renaming a robot controller

Renaming a robot controller

You can rename a robot controller without re-installation.

1. In the **Installation Manager** window, select **Controllers** and then select **Network** tab.
2. Select the active system which must be renamed.
3. Click **Rename**. The **Rename System** dialog appears.
4. In the **Enter the new name for the system** box, type in the new name.
5. Click **Rename**.

9.2 Using System Builder for managing RobotWare 5

9.2.1 About System Builder

Overview

This section describes how you create, build, modify and copy systems to run on virtual and robot controllers. These systems may even be converted to boot media and downloaded to a robot controller.

The system points out the robot models and options to use; it also stores configurations and programs for the robots. Therefore, it is good practice to use a unique system for each station even if the stations use the same basic setup. Otherwise, changes in one station may accidently overwrite data used in another station.

Note

Use System Builder to create and modify systems based on RobotWare 5.xx. Use Installation Manager to create and modify systems with RobotWare versions 6.0 and later.

About virtual and real systems

The system you run on virtual controllers can either be a real system built on real RobotWare keys or a virtual system built on virtual keys.

When using real systems, the RobotWare keys define which options and robot models shall be used, thus helping you to configure the system correctly. Real systems can be run both on virtual controllers and real IRC5 controllers.

When using virtual keys, all options and robot models are available, which is useful for evaluation purposes, but requires more configuration when creating the system. Systems built on virtual keys can only be run on virtual controllers.

Prerequisites

Creating a system entails applying a predefined template to a station, reusing an existing system or letting RobotStudio propose a system based on a layout.

To create a system, the following conditions must be met:

- The RobotWare media pool must be installed on your PC.
- You must have a RobotWare key for the system, if creating a system to run on a robot controller. The RobotWare key is a license key that determines which robot models to use and which RobotWare options to run on the controller. The license key is delivered with the controller.
- If you want to create a system for virtual use only, you can use a virtual key instead. Virtual keys are generated by the wizard. When using virtual keys, you select the robot models and options to use in the *Modify Options* section of the wizard.
- Downloading to the robot controller requires a direct connection from your computer to the service or Ethernet port of the controller.

9.2.2 Viewing system properties

Overview

All systems you create with the System Builder are stored locally on your computer. It is recommended that you store them in one or more dedicated system directories.

Viewing system properties

To view system properties and add comments, follow these steps:

1. In the **System Builder** dialog box, select a system from the **Systems** box.

   If necessary, in the **System directory** list, you can navigate to the folder in which your systems are stored.
2. The system properties are then displayed in the **System Properties** box. Optionally, type a comment in the **Comments** box, and click **Save**.

9.2.3 Building a new system

Overview

The **New Controller System Wizard**, used for building a new system, is launched from the System Builder.

Starting the wizard

To start the wizard, follow these steps:

1. Click **System Builder** to bring up a dialog box.
2. In the **Actions** group, click **Create New**. This starts the wizard.
3. Read the information on the welcome page and click **Next**.

Specifying the name and location

To determine where on your computer to store the system you are creating, follow these steps:

1. In the **Name** box, enter a name for the system you are creating.
2. In the **Path** box, enter the path to the system directory in which you will store the system.

   You can also click the **Browse** button and browse to the system directory.
3. Click **Next**.

Entering the RobotWare keys

The RobotWare keys determine which RobotWare versions and parts to use in the system.

Creating a system to run on either IRC5 controllers or virtual controllers requires at least two keys: one for the controller module and one for each drive module in the cabinet. The keys are delivered together with the controller.

For creating a system to run on virtual controller only (for example, in Virtual IRC5), you can use virtual keys. Virtual keys give access to all options and robot models, but limits the use of the system to virtual controllers only.

To enter the key for the controller module, follow these steps:

1. In the **Controller Key** box, enter the controller key. You can also click **Browse** and browse to the key file. If creating a system for virtual use only, select the **Virtual Key** check box, and the controller key will be generated by the wizard.
2. In the **Media Pool** box, enter the path to the media pool. You can also click **Browse** and browse to the folder system.
3. In the **RobotWare Version** list, select which version of the RobotWare you want to use. Only RobotWare versions that are valid for the used key are available.
4. Click **Next**.

Continues on next page

9.2.3 Building a new system
Continued

Entering the drive keys

To enter the keys for the drive modules:

1. In the **Drive Key** box, enter the key for the drive module. You can also click the **Browse** button and browse to the key file. If you used a virtual controller key, a virtual drive key is already generated by the wizard.
2. Click the right arrow button next to the **Drive Key** box. The key now appears in the **Added drive keys** list.

   For real systems the drive key determines the connected robot model. For virtual systems you select the robot model in the *Modify Options* page. The default model is IRB140.
3. If you have a MultiMove system, repeat steps 1 and 2 for each drive key to add.

   If you have a MultiMove system, make sure that the keys are numbered in the same way as their corresponding drive modules are connected to the controller module. Use the up and down arrows to rearrange the drive keys, if necessary.
4. If you want to create the system as it is now, click **Finish**.

   If you want to modify options, or add options, parameter data or additional files to the home directory, click **Next**.

Adding additional options

Here you can add options, such as external axes and dispense applications, that are not included in the basic system. Options require a license key and must be first imported to the media pool. To add additional options, follow these steps:

1. In the **Key** box, enter the option key. You can also click the **Browse** button and browse to the option's key file.
2. Click the **Arrow** button.

   The option that the key unlocks is now displayed in the **Added Options** list.

   Note

   If several versions of an additional option exists, only the latest version can be selected. To use an older version, remove the other versions of the additional option from the Mediapool.

   System Builder can modify a system only when all referenced additional options and RobotWare mediapool are placed in the same folder. It is not possible to modify a system that uses a mediapool embedded in a Pack and Go file. You must copy the mediapool to a common mediapool folder and create a system from backup.
3. Repeat steps 1 and 2 for all options you want to include.
4. Choose whether you want to create the system as it is now, or to continue with the wizard.

   If you want to create the system as it is now, click **Finish**.

Continues on next page

If you want to modify options, or add parameter data or additional files to the home directory, click **Next**.

Modifying options

Here you can set up and configure the options in your system. For virtual robot systems, you also select the robot models to use. To modify any options, follow these steps:

1. In the **Option** tree, expand the option folders to the level where you find the option you want to modify.

   Only the options unlocked by the used keys are available.
2. Modify the option.
3. Repeat steps 1 and 2 for all options you want to modify.
4. Choose whether you want to create the system as it is now, or to continue with the wizard.

   If you want to create the system as it is now, click **Finish**.

   If you want to add parameter data or additional files to the home directory, click **Next**.

Adding parameter data

Parameter data is stored in the parameter data files (.cfg files). Each parameter topic has its own parameter file. You can add only one parameter file for each topic. To add parameter data, follow these steps:

1. In the **Parameter data** box, enter the path to the folder for the parameter data files. You can also click the **Browse** button and browse to the folder.
2. In the list of parameter data files, select the file you want to include and press the **Arrow** button. Repeat for all files you want to include.

   The included parameter data files will now appear in the **Added parameter data files** list.

   Repeat steps 1 and 2 for each parameter data file you want to add.
3. Choose whether you want to create the system as it is now, or to continue with the wizard.

   If you want to create the system as it is now, click **Finish**.

   If you want to add additional files to the home directory, click **Next**.

Adding files to the home directory

You can add any type of file to the system’s home directory. When the system is loaded to a controller, these files will also be loaded. To add files to the system's home directory, follow these steps:

1. In the **Files** box, enter the path to the folder for the files you want to include. You can also click the **Browse** button and browse to the folder.
2. In the list of files, select the file to add and click the **Arrow** button. Repeat for all files you want to add.

   The added files will now appear in the **Added files** list.

Continues on next page

9.2.3 Building a new system
Continued

3 Choose whether you want to create the system as it is now, or to continue with the wizard.

If you want to create the system as it is now, click **Finish**.

If you want to read a summary before you create the system, click **Next**.

Completing the New Controller System Wizard

To complete the wizard, follow these steps:

1 Read the system summary.

2 If the system is OK, click **Finish**.

If the system is not OK, click **Back** and make modifications or corrections.

9.2.4 Modifying a system

Overview

The **Modify Controller System Wizard**, used to modify existing systems, is launched from the System Builder. The wizard helps you with tasks like changing robots, adding and removing external axes and other options. A system that is running must be first shut down before modification.

Starting the wizard

To start the wizard when creating a new station:

1. If the system is currently running, on the **Controller** menu, point to **Shutdown** and then click **Shutdown**.
2. On the **Controller** menu, click **System Builder** to bring up a dialog box.
3. In the **System directory** list, enter or browse to the system directory. Select a system from the list beneath, review the system properties and add and save any comments.
4. In the **Actions** group, click **Modify**. This starts the wizard.
5. Read the information on the welcome page and click **Next**.

Modifying the program revision

The RobotWare versions that are available for the system are determined by the controller key. The key is essential to the system and cannot be modified.

To use another RobotWare version than the available ones, create a new system with another key.

To optionally modify the program revision, follow the appropriate step or steps:

1. To keep the current RobotWare version, select **Yes** and then click **Next**.
2. To replace the current RobotWare version, Select **No, replace it**.
3. In the **Media pool** box, enter the path to the media pool. You can also click the **Browse** button and browse to the folder.
4. In the **New program revision** box, select which version of RobotWare you want to use. Only RobotWare versions that are valid for the RobotWare key are available.
5. Click **Next**.

Adding or removing drive keys

The drive key corresponds to the drive modules in your controller. For MultiMove systems, you have one drive module (and one key) for each robot. The keys for your system are delivered together with the controller.

The system is created with a virtual robot controller key, virtual drive keys are generated by the wizard. When you have added one virtual drive key for each robot, you select which robot to use for each key on the *Modify Options* page.

To optionally add or remove the keys for the drive modules, follow these steps:

1. To add a key for a drive module, enter the key in the **Enter Drive Key** box. You can also click the **Browse** button and browse to the key file.

Continues on next page

2 Click the right arrow button. The key now appears in the **Added drive key** list.

   If you have a MultiMove system, repeat steps 1 and 2 for each drive key to add.

3 To remove a drive module, select the corresponding key in the **Added drive key** list and click **Remove drive key**.

   If you have a MultiMove system, repeat step 3 for each drive key to remove.

4 If you have a MultiMove system, make sure that the keys are numbered in the same way as their corresponding drive modules are connected to the controller module. Use the up and down arrows to rearrange the drive keys, if necessary.

5 Choose whether you want to create the system as it is now, or to continue with the wizard.

   If you want to create the system as it is now, click **Finish**.

   If you want to modify options, parameter data or add files to or remove files from the home directory, click **Next**.

Adding or removing additional options

To optionally add or remove additional options:

1 To add an add-in, in the **Enter Key** box, enter the option key. You can also click the **Browse** button and browse to the option's key file.

2 Click the **Arrow** button.

   The option that the key unlocks is now displayed in the **Added Options** list.

> **Note**
>
> If several versions of an additional option exists, only the latest version can be selected. To use an older version, remove the other versions of the additional option from the Mediapool.

3 Repeat steps 1 and 2 for all options you want to include.

4 To remove an add-in, In the additional options, in the Added options list, select the add-in you want to remove.

5 Click **Remove**.

6 Choose whether you want to create the system as it is now, or to continue with the wizard.

   If you want to create the system as it is now, click **Finish**.

   If you want to modify parameter data or add files to or remove files from the home directory, click **Next**.

Modifying options

To optionally modify any options, follow these steps:

1 In the **Option** tree, expand the option folders to the level where you find the option you want to modify.

   Only the options unlocked by the used keys are available.

Continues on next page

2 Modify the option.
3 Repeat steps 1 and 2 for all options you want to modify.
4 Choose whether you want to create the system as it is now, or to continue with the wizard.

   If you want to create the system as it is now, click **Finish**.

   If you want to modify parameter data or add files to or remove files from the home directory, click **Next**.

Adding or removing parameter data

Parameter data is stored in the parameter data files (.cfg files). Each parameter topic has its own parameter file. You can add only one parameter file for each topic. To add or remove parameter data, follow these steps:

1 To add parameter data, in the **Parameter data** box, enter the path to the folder for the parameter data files. You can also click the **Browse** button and browse to the folder.
2 In the list of parameter data files, select the file you want to include and press the **Arrow** button. Repeat for all files you want to include.

   The included parameter data files will now appear in the **Added parameter data files** list.

   Repeat steps 1 and 2 for each parameter data file you want to add.
3 To remove parameter data, in the **Added parameter data files** list, select the parameter data file to remove.
4 Click **Remove**.
5 Choose whether you want to create the system as it is now, or to continue with the wizard.

   If you want to create the system as it is now, click **Finish**.

   If you want to add to or remove files from the home directory, click **Next**.

Add files to or remove files from the home directory

You can add any type of file to the system’s home directory, or remove files from it. When the system is loaded to a controller, these files will also be loaded. To optionally add files to or remove files from the system's home directory, follow these steps:

1 To add files, in the **Files** box, enter the path to the folder for the files you want to include. You can also click the **Browse** button and browse to the folder.
2 In the list of files, select the file to add and click the **Arrow** button. Repeat for all files you want to add.

   The added files will now appear in the **Added files** list.
3 To remove files, in the **Added files** list, select the file to remove.
4 Click **Remove**.
5 Choose whether you want to create the system as it is now, or to continue with the wizard.

   If you want to create the system as it is now, click **Finish**.

Continues on next page

9.2.4 Modifying a system
Continued

If you want to read a summary before you create the system, click **Next**.

Complete the Modify Controller System wizard

To complete the wizard, follow these steps:

1 Read the system summary.
2 If the system is OK, click **Finish**.
If the system is not OK, click **Back** and make modifications or corrections.

Result

Modifications will take effect when the wizard is completed.

If the system has been downloaded to a controller, it must be downloaded again before the modifications will take effect on the controller.

If the system is used by a virtual controller, restart the controller using the restart mode **Reset controller** for the changes to take effect.

Deleting a system

To delete a system, follow this steps:

1 From the **System Builder** dialog box, select the system and then click **Delete**.

9.2.5 Copying a system

Copy a system

To copy a system, follow these steps:

1. From the **System Builder** dialog box, select the system and then click **Copy** to bring up a dialog box.
2. Enter a name for the new system and a path, and then click **OK**.

9.2.6 Creating a system from backup

Overview

The **Create System from Backup Wizard**, which creates a new system from a controller system backup, is launched from the System Builder. In addition, you can change the program revision and options.

Starting the wizard

To start the wizard, follow these steps:

1. From the **System Builder** dialog box, click **Create from Backup**. This starts the wizard.
2. Read the information on the welcome page and click **Next**.

Specifying the name and location

To specify the destination folder, follow these steps:

1. In the **Name** box, enter a name for the system you are creating.
2. In the **Path** box, enter the path to the system directory in which you will store the system.

   You can also click the **Browse** button and browse to the system directory.
3. Click **Next**.

Locating the backup

To locate a system from backup, follow these steps:

1. In the **Backup folder** box, enter the path to the backup folder. Alternatively, click the **Browse** button to browse to it. Click **Next**.
2. In the **Media Pool** box, enter the path to the media pool containing the appropriate RobotWare program. Confirm the backup information that now appears in the wizard. Click **Next**.

9.2.7 Downloading a system to a controller

Overview

All systems you access from the System Builder are stored on your computer. If you wish to run a system on a robot controller, you must first load it to the controller, which then requires a restart.

Load a system

To load a system to a controller, follow these steps:

1 From the System Builder dialog box, select a system and then click **Download to Controller** to bring up a dialog box.

> **Note**
>
> Systems with incompatible hardware versions will not be displayed in the **Download to Controller** dialog box.

2 Specify the Destination Controller for the system.

You can select by using the...	if...
Select controller from list option	the controller has been detected automatically.
Specify IP address or controller name option	your PC and the robot is connected to the same network. You can only use the controller name in DHCP networks.
Use service port option	your PC is directly connected to the controller's service port.

3 Optionally, click **Test Connection** to confirm that the connection between the computer and the Controller is OK.

4 Click **Load**.

5 Answer **Yes** to the question **Do you want to restart the controller now?**

Yes	The controller restarts immediately and the downloaded system starts automatically.
No	The controller does not restart immediately. To start using the downloaded system, you have to: a Restart the controller using the restart mode **Start Boot Application**. b Select the system manually.
Cancel	The downloaded system is removed from the controller.

9.2.8 Examples using the System Builder when offline

9.2.8.1 A system with support for one robot and one positioner external axis

Overview

In this example we will use the System Builder to create an offline system to use in a new RobotStudio station with one IRB1600 robot and one IRBP 250D positioner external axis.

Prerequisites

When creating systems for positioner external axes, you need the media pool and the license key file for that specific positioner. In this example we will use a media pool and license key file for a demo positioner.

Paths to files and folders assume that RobotStudio and the RobotWare media pool have been installed at their default locations on Windows XP. If not, adjust the paths accordingly.

Starting the New Controller System Wizard

To create a system like the one described above, follow these steps:

1. Click **System Builder** to bring up a dialog box.
2. In the dialog box, click **Create New** to bring up the **New Controller System Wizard**.
3. Read the welcome text, and click **Next** to continue to the next page.

Entering the controller key

1. Select the **Virtual key** check box. A virtual controller key now appears in the Controller Key box. In this example we will use the default media pool and RobotWare version.
2. Click **Next** to continue to the next page.

Entering drive keys

1. Click the **Right Arrow** button next to the **Enter Drive key** box to create one drive key for the robot.
2. Click **Next** to continue to the next page.

Adding options

This is where we point out the key file for the positioner.

1. Next to the **Enter key** box, click the browse button and select the key file.

Continues on next page

In this example, browse to and select the file *extkey.kxt* in the folder *C:\Program Files\ABB Industrial IT\Robotics IT\MediaPool\3HEA-000-00022.01*.

> **Tip**
>
> In the *MediaPool* folder media pools for several standard positioners are installed. They are named by the positioner's article number, with a suffix that indicates if it is configured for single-robot or MultiMove systems.

2 Click the *Right Arrow* button next to the **Enter key** box to add the key for the positioner.

3 Click **Next** and continue to the next page of the wizard.

Modifying options

When creating robot systems from real robot keys, the key sets the options. But since we are using a virtual key, we have to set the options manually. To set the options necessary for a positioner, follow these steps:

1 Scroll down to the **RobotWare** / **Hardware** group and select the **709-x DeviceNet** check box.

This option is for the communication between the controller and the track external axis.

2 Scroll down to the **DriveModule1** / **Drive module application** group and expand the **ABB Standard manipulator** option. Select the **IRB 1600** option.

This option sets the robot to an IRB 1600-5/1.2.

3 Scroll down to the **DriveModule1** > **Drive module configuration** group; select the **Drive System 04 1600/2400/260** option; expand the **Additional axes drive module** group and select the **R2C2 Add drive** option.

a Expand the **Drive type in position Z4** group and select the **753-1 Drive C in pos Z4** option

b Expand the **Drive type in position Y4** group and select the**754-1 Drive C in pos Y4** option

c Expand the **Drive type in position X4** group and select the**755-1 Drive C in pos X4** option

Continues on next page

9.2.8.1 A system with support for one robot and one positioner external axis
Continued

This option adds drive modules for the positioner axes.

Note

When using the latest drive system, do the following:

Scroll down to the **DriveModule1** > **Drive module configuration** group; select the **Drive System 09 120/140/1400/1600 Compact** option; expand the **Power supply configuration** group and select **1-Phase Power supply** or **3-Phase Power supply** (as applicable) > **Additional axes drive module** > **Additional drive**

a Expand the **Drive type in position X3** group and select the **Drive ADU-790A in position X3** option

b Expand the **Drive type in position Y3** group and select the **Drive ADU-790A in position Y3** option

c Expand the **Drive type in position Z3** group and select the **Drive ADU-790A in position Z3** option

4 Click **Finish** and the system will be created. When starting the system in a RobotStudio station, you have to set up the system to load a model for the positioner and to get the motions to work properly.

9.2.8.2 Options settings for systems with positioners

Overview

This is an overview of the RobotWare options to set when creating a system for positioner external axes. Note that besides setting the RobotWare options, you must add an additonal option key for the positioner.

Media pools and option keys for the positioners

If you have the media pool and option key for your positioner, you can use these files.

If not, media pools for standard positioners are installed with RobotStudio. The path to these media pools in a default installation is: *C.\program files\ABB Industrial IT\Robotics IT\MediaPool*. In this folder a media pool for each positioner is located. These are named by the article number of the positioner, with a suffix that indicates if it is configured for a single-robot or a MultiMove system.

In the **Additional option** page of the **System Builder**, you should add the option for the positioner by opening the mediapool folder for the positioner to add and selecting the *extkey.kxt* file.

Options for positioners in single-robot systems

When adding a positioner to a single-robot system, the positioner will be added to the same task as the robot. Below, the options to set on the **Modify Options** page of the **System Builder** for such a system are listed:

- **RobotWare** > **Hardware** > **709-x DeviceNet** > **709-1 Master/Slave Single**
- Optionally, for using the system with ArcWare also add **RobotWare** > **Application arc** > **633-1 Arc**
- **DriveModule 1** > **Drive module configuration** > **Drive System 04 1600/2400/260** > **RC2C Add drive** > **753-1 Drive C in pos Z4** > **754-2 Drive T in pos Y4** > **755-3 Drive U in pos X4**

Options for positioners in MultiMove robot systems

When adding a positioner to a MultiMove robot system, the positioner shall be added to a task of its own (thus you also have to add a drive key for the positioner). Below, the options to set on the **Modify Options** page of the **System Builder** for such a system are listed:

- **RobotWare** > **Hardware** > **709-x DeviceNet** > **709-1 Master/Slave Single**
- **RobotWare** > **Motion coordinated part 1** > **604-1 MultiMove Coordinated** Optionally, expand the MultiMove Coordinated option and select process options for the robots.
- Optionally, for using the system with ArcWare, add **RobotWare** > **Application Arc** > **633-1 Arc**
- **DriveModule 1** > **Drive module configuration** > **Drive System 04 1600/2400/260** > **RC2C Add drive** > **753-1 Drive C in pos Z4** > **754-2 Drive T in pos Y4** > **755-3 Drive U in pos X4**. For the other drive modules, no additional axes should be configured.

9.3 A MultiMove system with two coordinated robots

9.3.1 Creating a coordinated system using System Builder

Overview

In this example we will use the System Builder to create a coordinated offline system with one IRB2400 and one IRB1600 robot to use in a new RobotStudio station.

Starting the New Controller System Wizard

To create a system like the one described above, follow these steps:

1. Click **System Builder** to bring up the dialog box.
2. In the dialog box, click **Create New** to bring up the **New Controller System Wizard**.
3. Read the welcome text, and click **Next** to continue to the next page.

Entering the name and path

1. In the **Name** box, enter the name of the system. The name must not contain blank spaces or non-ASCII characters.

   In this example, name the system *MyMultiMove*.
2. In the **Path** box, enter the path for the folder to save the system in, or click the **Browse** button to browse to the folder or create a new one.

   In this example, save the system in *C:\Program Files\ABB\RobotStudio\ABB Library\Training Systems*.
3. Click **Next** to continue to the next page.

Entering the controller key

1. Select the **Virtual key** check box. A virtual controller key now appears in the Controller Key box. In this example we will use the default media pool and RobotWare version.
2. Click **Next** to continue to the next page.

Entering drive keys

1. Click the **Right Arrow** button next to the **Enter Drive key** box twice to create one drive key for each robot.
2. Click **Next** to continue to the next page.

Adding options

This system does not require any additional option keys. Click **Next** and continue to the next page of the wizard.

Modifying options

When creating robot systems from robot keys, the key sets the options. But since we are using a virtual key, we have to set the options manually.

Continues on next page

When creating a system for several manipulators (up to four), you must include either of the RobotWare options **MultiMove Independent**, or **MultiMove Coordinated** for the related motion tasks to start.

> **Note**
>
> It is recommended to use the **System From Layout** function when you create robot systems for RobotStudio. Then the MultiMove option gets added automatically.

To set the options necessary for a MultiMove, follow these steps:

1. Scroll down to the **RobotWare** / **Motion Coordination part 1** group and select the **MultiMove Coordinated** check box.
2. Scroll down to the **RobotWare**/ **Engineering Tools** group and select the **Multitasking** check box.

   > **Note**
   >
   > The option **Advanced RAPID** is included in RobotWare- operating system from RobotWare 5.60 and later.

3. Scroll down to the **DriveModule1** / **Drive module application** group and expand the **ABB Standard manipulator** option. Select the **IRB 2400 Type A** option, manipulator variant **IRB 2400L Type A**.
4. Scroll down to the **DriveModule2** / **Drive module application** group and expand the **ABB Standard manipulator** option. Select the **IRB 1600** option, manipulator variant **IRB 1600-5/1.2**.
5. Click **Finish** and the system will be created.

9.3.2 Creating a coordinated system using Installation Manager

Overview

In this example we will use the Installation Manager to create a coordinated *offline* system to use in a new RobotStudio *station*.

Creating a coordinated system

To create a system like the one described above, follow these steps:

1. In **Installation Manager**, in the **Controllers** page select a system and click **Next**.
2. In the **Product** page, click **Add**, the **Select Product** dialog box is displayed.
3. Select the required product and click **OK**.
4. Select the added product and click **Next**.
5. In the **Licenses** page, select the required license and click **Next**.
6. In the **Options** page, under the **System Options** tab, select the **Motion Coordination** > **Multimove Options** > **604-1 MultiMove Coordinated** check box. The **Engineering Tools** option will auto-expand.
7. Under the **Engineering Tools** option, select the required check boxes and click **Next**.
8. In the **Confirmation** page, review the system configuration and click **Apply**, to create the system.

10 Working with system parameters

10.1 System parameters

Overview

The controller configuration is a collection of six topics, each describing a configuration area of the controller. A controller is configured at the factory as per the *RobotWare* options that are selected at the time of delivery. The factory default configuration is altered only during an update or any alteration processes. The configuration parameters can be saved as text files *(*.cfg)* that lists the values of system parameters. If the parameter is assigned the default value, then it will not be listed in the configuration file.

When creating a backup of the controller, the configuration files will be stored in the *SYSPAR* folder of the backup file structure. The configuration files will be loaded into the controller memory when the backup is restored.

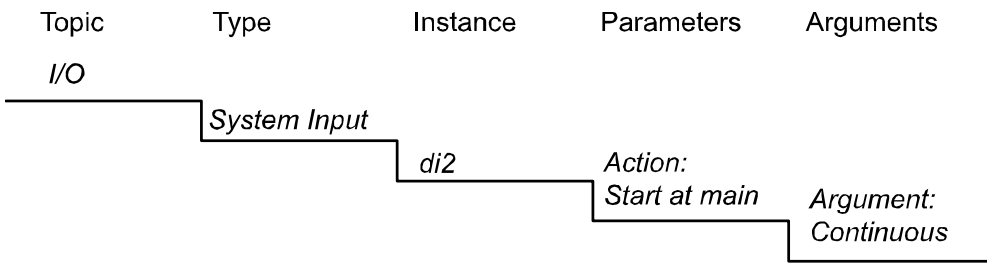
The controller system folder contains the *SYSPAR* folder in `...\MySystem\SYSPAR\` location. The configuration files in this folder gets loaded when the controller resets. Note that changes to the configuration and *RAPID* will be discarded during a controller reset. Configuration changes which are related to the installation and independent of the program being executed can be loaded from the *SYSPAR* folder. Examples of folders that can be loaded are configuration of background *tasks* and the corresponding RAPID *modules*.

Topic:	Configuration area:	Configuration file:
Communication	Communication protocols and devices	SIO.cfg
Controller	Safety and RAPID specific functions	SYS.cfg
I/O	I/O boards and signals	EIO.cfg
Man-machine communication	Functions to simplify working with the *virtual controller*	MMC.cfg
Motion	The robot and *external axes*	MOC.cfg
Process	Process specific tools and equipment	PROC.cfg

A topic is a collection of system parameters of the same type. It represents a configuration area of the controller. A separate configuration file is saved for each topic, it can also be generated while creating a backup. Type holds the parameter definition. The parameter values are normally predefined at delivery. The values are restricted to data type, and sometimes to be within an interval. An instance is a user defined variable of the selected type. In some cases system parameters, depending on their values, are further structured into subparameters, also called

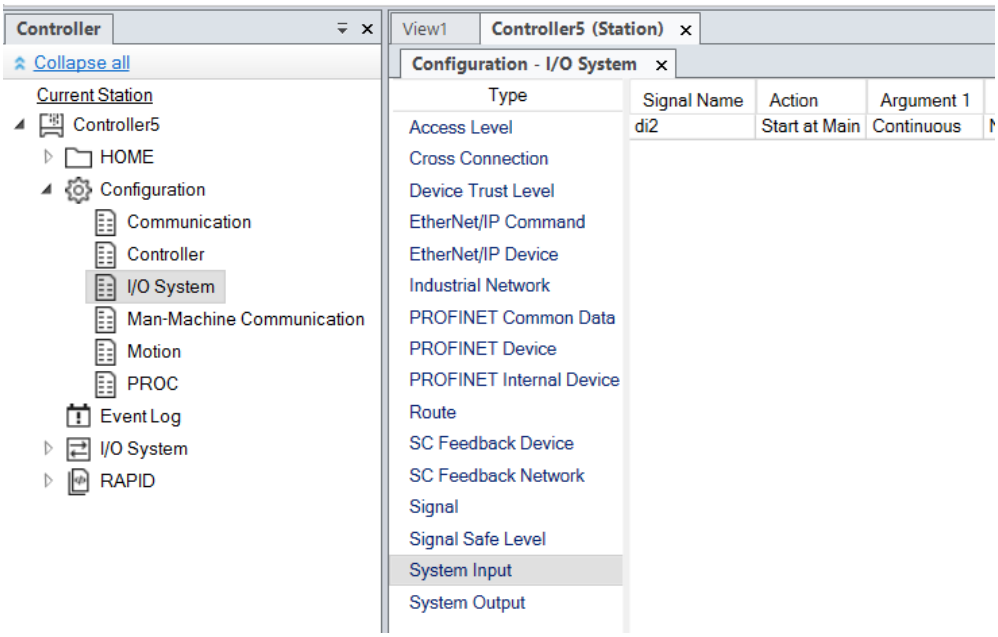
Continues on next page

arguments or action values. The following image shows the details of the I/O signal, *di2*.



en0800000183

System parameters are configured using RobotStudio or the FlexPendant.



xx1800003052

Viewing configurations

1. To view the topics of a controller, from the **Controller** tab, expand the **Configuration** node of the controller.

   All topics are now displayed as child nodes of the Configuration node.
2. To view the types and instances of a topic, double-click the required topic node.

   The **Configuration Editor** opens and lists all types of the topic in the **Type name** list. In the **Instance** list, rows display the selected type and columns display parameter values of the instances.
3. To view detailed parameter information of an instance, double-click the instance.

   The instance editor displays the current value, restrictions and limits of each parameter in the instance.

Continues on next page

Note

Virtual networks, devices and signals contained in the EIO.cfg file can be viewed the I/O System node in the Controller browser.

10.2 Adding instances

Use the **Configuration Editor** to select a configuration type and to create a new instance of the type, for example, adding a new instance of the type *Signal* creates a new signal in the virtual controller.

1. In the **Controller** tab, expand the **Controller** and the **Configuration** node and double-click the topic that contains the type to add an instance. This opens the Configuration Editor.
2. In the **Type name** list, select the type to add an instance.
3. Double-click the type, the corresponding window opens to the right hand side.
4. Right-click any element, and click **Edit Tasks**.
5. On the **Controller** menu, point to **Configuration** and click **Add type** (the word type is replaced by the type that was selected earlier).

   Right-click anywhere in the configuration editor and then select **Add type** from the shortcut menu.

   A new instance with default values gets added and displayed in the **Instance Editor** window.
6. Click **OK** to save the new instance.

   The values in the new instance are now validated. If the values are valid, the instance gets saved, else a notification on the incorrect parameter values gets displayed. Certain changes take effect only after a controller restart.

10.3 Copying an instance

1. In the **Controller** tab, expand the **Controller** and the **Configuration** node and double-click the topic that contains the instance to copy. This opens the Configuration Editor.
2. In the **Type name** list of the Configuration Editor, select the type to copy an instance.
3. Right--click the selected row, and then click **Copy** and change the name of the instance. Click **OK**.
4. To select multiple instances, in the **Instance** list, select several instances to copy.

   The parameter values of all selected instances must be identical, else the default values will be absent in the parameters of the new instances.
5. On the **Controller** menu, point to **Configuration** and click **Copy Type** (the word type is replaced by the type that was selected earlier).

   Right-click the instance to copy and then select **Copy Type** from the shortcut menu.

   A new instance with the same values as the one that was copied gets added and displayed in the **Instance Editor** window.
6. Change the name of the instance and click **OK** to save the new instance.

   The values in the new instance are now validated. If the values are valid, the instance gets saved, else a notification on the incorrect parameter values gets displayed. Certain changes take effect only after a controller restart.

10.4 Deleting an instance

1. In the **Controller** tab, expand the **Controller** and the **Configuration** node and double-click the topic that contains the type of which you want to delete an instance. This opens the Configuration Editor.
2. In the **Type name** list of the Configuration Editor, select the type of which you want to delete an instance.
3. In the **Instance** list, select the instance to delete.
4. On the **Controller** menu, point to **Configuration** and then click **Delete type** (the word type is replaced by the type you selected previously).

   You can also right-click the instance to delete and then select **Delete type** from the shortcut menu.
5. A message box is displayed, asking if you want to delete or keep the instance. Click **Yes** to confirm that you want to delete it.

   The values in the new instance are now validated. If the values are valid, the instance gets saved, else a notification on the incorrect parameter values gets displayed. Certain changes take effect only after a controller restart.

10.5 Save one configuration file

The system parameters of a configuration topic can be saved to a configuration file, stored on the PC or any of its network drives.

The configuration files can then be loaded into a controller. They are thereby useful as backups, or for transferring configurations from one controller to another.

1. In the **Controller** tab, expand the **Configuration** node and select the topic to save to a file.
2. On the **Controller** menu, point to **Configuration** and select **Save Parameters**.

   You can also right-click the topic and then select **Save System Parameters** from the shortcut menu.
3. In the **Save As** dialog box, browse for the folder to save the file in.
4. Click **Save**.

10.6 Saving several configuration files

1. In the **Controller** tab, select the **Configuration** node.
2. On the **Controller** menu, point to **Configuration** and click **Save System Parameters**.

   You can also right-click the configuration node and then click **Save System Parameters**.
3. In the **Save System Parameters** dialog box, select the topics to save to files. Then click **Save**.
4. In the **Browse for Folder** dialog box, browse for the folder to save the files in, and then click **OK**.

   The selected topics will now be saved as configuration files with default names in the specified folder.

10.7 Loading a configuration file

A configuration file contains the system parameters of a configuration topic. They are thereby useful as backups, or for transferring configurations from one controller to another. When loading a configuration file to a controller, it must be of the same major version as the controller. For instance, you cannot load configuration files from an S4 system to an IRC5 controller.

1. In the **Controller** tab, select the **Configuration** node.
2. On the **Controller** menu, point to **Configuration** and select **Load Parameters**.

   Alternatively, right-click the configuration node and then select **Load Parameters** from the context menu. This opens the **Select mode** dialog box.
3. In the **Select mode** dialog box, select the following options:
   - Select **Delete existing parameters before loading** to replace the entire configuration of the topic with the one in the configuration file.
   - Click **Load parameters if no duplicates** to add new parameters from the configuration file to the topic, without modifying the existing ones.
   - Click **Load parameters and replace duplicates** to add new parameters from the configuration file to the topic and update the existing ones with values from the configuration file. Parameters that exist only in the controller and not in the configuration file will remain.
4. Click **Open** and browse to the configuration file to load. Then click **Open** again.
5. In the information box, click **OK** to load the parameters from the configuration file.
6. After loading the configuration file, close the **Select mode** dialog box.

   Certain changes take effect only after a controller restart.

This page is intentionally left blank

11 Monitoring robot signals

11.1 Signal analyzer

Overview

Signal Analyzer is used to record robot signals from *robot* or *virtual controllers*. The version of Signal Analyzer used for robot controllers is called Signal Analyzer Online.

- To open Signal Analyzer, on the **Simulation** tab, in the **Signal Analyzer** group, click **Signal Analyzer**.
- To open Signal Analyzer Online, on the **Controller** tab, in the **Controller Tools** group, click **Signal Analyzer Online**.

Significance of signal analyzer

The data from the Signal Analyzer can be used to fine-tune robot behavior for optimizing robot performance during production. For example, if the robot is working with a continuous process where constant speed is the critical parameter, then the Signal Analyzer can be used to monitor the speed signals from the robot motion system. Constant monitoring helps in identifying any speed fluctuations which can be traced to the corresponding line in the RAPID code that causes the speed fluctuations. After identifying the root cause of the fluctuations, the robot program can be edited to rectify these issues.

Another example where Signal Analyzer can be used is cable simulation, where the physics signals monitor the cable tension, length and twist. These signals can be used for identifying the wear and tear of cables. Based on these signal data, the cable (routing, length) and the program can be altered to reduce wear.

11.2 Monitored signals

Signal analyzer can only monitor a group of selected signals. This selection varies for robot controller and virtual controller. The following table provides the list of signals for robot and virtual controllers that can be monitored in signal analyzer.

Category	Available signals
Controller Signals	Total Motor Power
	Total Power Consumption
EventLog	All domains
I/O System	All signals
Joint	J1-J6
	Near Limit
Target	Fine Point
	Target Changed[1] , *Tool* Changed, *Workobject* Changed
TCP	Maximum Linear Acceleration in World
	Linear Acceleration In World
	Orientation Q1-Q4 Current Workobject
	Orientation Speed in Current Workobject
	Pos X, Y, Z in Current Workobject
	Robot Configuration cf1, cf4, cf6, cfx
	Speed in Current Workobject
Stress	Maximum Stress Index[3]
	Stress Index J1[2]
	Stress Index J2
	Stress Index J3
	Stress Index J4
	Stress Index J5
	Stress Index J6
Smart Components	All signals
Physics	Cable tension, cable length and cable twist (only available for stations with cables)
Stop distance estimation	All signals, available only for virtual controllers.
Devices (polled)	CPU temperature and RAPID Memory.

Continues on next page

Category	Available signals
1. The Target Changed event data do not represent the current target, instead it represents the subsequent target. *2. Signal for assessing the amount of stress that the joints are subjected to. Monitoring this signal helps in comparative analyses between different options of cycles to reduce the mean value of the signal rather than removing individual maximum values. This signal can also be used to compare the same cycle on different robots. This signal has no unit of measurement and depends on how hard the robot is run relative to the gear on the axis. For a stationary axis, the Stress Index signal value is zero. The maximum value of Stress Index signal is 110.* *3. The Maximum Stress Index signal has the value of the Stress Index Jx signal with the highest value at each point in time.*	

Stop distance estimation

These signals show the stop distance of the robot during a *category 0 stop* or *category 1 stop*. For example, if the stop distance signal for joint J1 is 7 degrees at time=48 ms, it means that J1 will move 7 degrees in positive direction if the robot is stopped at t=48 ms.

WARNING

The measurement and calculation of overall stopping performance for a robot must be tested with its correct load, speed, and tools, in its actual environment, before the robot is taken into production, see *ISO 13855:2010*.

Total Motor Power

The Total Motor Power signal shows the total instantaneous power for each *joint*. It may be positive or negative.

The instantaneous power for a specific joint is positive when it accelerates and negative when it decelerates. If one joint is accelerating at the same time as another joint is decelerating, then the negative energy from the decelerating joint is reused by the accelerating joint. If the sum of the instantaneous power of all joints is negative then the power surplus is either fed back to grid or burned off in the bleeder.

For a virtual robot the signal is based on a nominal robot during typical conditions, for a robot the signal is based on the torque for that particular robot in the actual conditions. For a robot the value of the motor power signal depends on various factors, for example, the temperature of the robot and the length of the cables.

Note

Total Motor Power signal represents the power consumed by the mechanical robot arm and not the power that is fed into the controller cabinet from the power network. The power used by the controller cabinet is excluded.

Continues on next page

Total Motor Energy

The Total Motor Energy signal is the integration of the power over time. The Total Motor Energy signal includes negative Total Motor Power signal when the controller supports refeeding of power.

Note

- For OmniCore controllers with version earlier than 7.10, Total Motor Energy signal is not decreased in RobotStudio even if there is a power surplus.
- For OmniCore controllers with version 7.10 or later, Total Motor Energy signal is decreased in RobotStudio when there is a power surplus.

Purpose of signals

The purpose of the Total Motor Power and Total Motor Energy signals is to provide an estimate of the power and energy used by the robots. For virtual robots, these signals can be used to identify peaks in the power usage to enable the robot programmer to adjust the robot program with the aim to reduce the power consumption. For robots, the signals can be used to compare the power usage of different robot individuals running the same robot program, to see if any robot differs significantly from the others. Any such deviation may indicate that the robot needs maintenance.

Near Limit

Near Limit checks the distance to the closest limit for each *joint*. If any joint is less than 20 degrees from a limit, the Near Limit signal will show the current value. Otherwise, the value of the signal will be constant at 20 degrees. If more than one joint is below 20 degrees from a limit, then the one that is closest will be looked at.

11.3 Recording signals

Recording signals for a virtual controller

Before starting signal analyzer, choose the signals that must be recorded in the **Signal Setup** window.

1. Load a *station* with a *virtual controller*.
2. In the **Simulation** tab, in the **Signal Analyzer** group, click **Signal Setup**.

   The **Signal Setup** window appears.
3. In the **Signals** view, select the signals to be recorded during simulation.

   The selected signals are added to the **Selection** view.
4. The signal recording can be set to start with the simulation. To enable signal recording during simulation, in the **Signal Analyzer** group, select **Enabled**.
5. The signal data from each signal recording session is saved. To view the signal recording, in the **Signal Analyzer** group, click **Recordings**.

To add the joint position signals of mechanical units to the signal recording, select **Signal Setup** and then select **Quick add position signals**. Use the **Recording Playback** feature to view the recorded signals.

Recording signals for a robot controller

RobotStudio must be connected to a robot controller for using the following procedure.

1. On the **Controller** tab, in the **Controller Tools** group, click **Signal Analyzer Online** and then click **Signal Setup**. The **Signal Setup** window opens.
2. In the **Signals** view, select the signals that must be recorded during simulation.

   The selected signals are added to the **Selection** view.
3. Start the simulation, and then in the **Controller Tools** group, click **Signal Analyzer Online** and then click **Start Recording**.
4. To stop recording, in the **Controller Tools** group, click **Signal Analyzer Online** and then click **Stop Recording**.
5. The signal data from each signal recording session is saved. To view the signal recording, in the **Signal Analyzer** group, click **Recordings**.

11.4 Recordings

Overview

The saved signal recordings of the current station are stored in the following locations.

- In the RobotStudio documents folder:

  `C:\Users\<user name>\Documents\RobotStudio\SignalAnalyzer\Stations` for a virtual controller and `C:\Users\<user name>\Documents\RobotStudio\SignalAnalyzer\Online` for a robot controller.
- In the Project folder for the respective project:

  `C:\Users\<user name>\Documents\RobotStudio\Projects\<Project folder>\SignalAnalyzer`.

The signal recordings can be exported in the following formats:

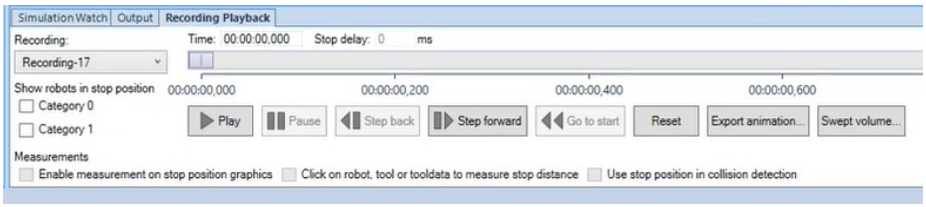
- RobotStudio Signal Recording (**.rssigdata*).
- Excel Workbook (**.xlsx*).
- Text (Tab delimited) (**.txt*).

The signal recordings can be imported in the RobotStudio Signal Recording (**.rssigdata*) format.

Using the recording play back feature

It is possible to record and view the joint position signals of mechanical units. Use the following procedure to record and then play back the signal recording of joint position signals.

1. To add the joint position signals of mechanical units to the signal recording, in the **Signal Analyzer** group, click **Signal Setup** and then click **Quick add position signals**.
2. In the **Simulation Control** group, Click **Play** to record the simulation.
3. In the **Signal Analyzer** group, click **Playback**. The **Recording Playback** tab gets displayed.



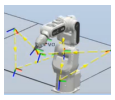
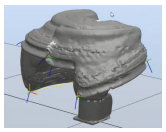
xx2000002027

4. In the **Recording:** list, select the required file and then click **Play** to view the recording.

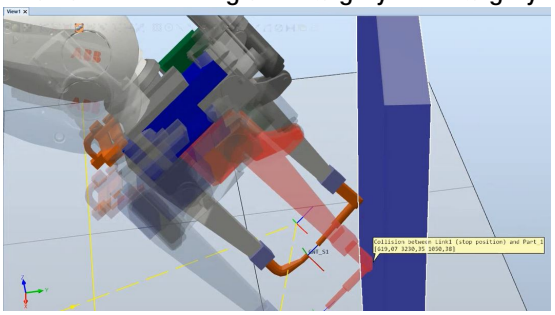
   Use the buttons in the **Recording Playback** tab to navigate through the recording during play back.

Continues on next page

Use the following check-boxes to enhance play back visualization.

Option	Description
Category0	View a semitransparent robot that moves along the *category 0 stop* positions.
Category1	View a semitransparent robot that moves along the *category 1 stop* positions.
Export Animation	Export a recording with joint signals to a 3D animation in *glTF (.glb)* format. SafeMove zones and geometries can be included if Visual SafeMove is open.
Swept Volume	Creates a new part based on the current recording that represents the approximate total volume of the robot and tool movement along a path. This will help in assessing collisions between the environment and the volume. These parts can be exported in polygon formats such as, glTF, STL, VRML and so on, but CAD formats are not supported. Part / Swept volume  xx2100002564  xx2100002565 The following options are available: **Construct from interpolated positions**: Creates a point cloud with extra points added to compensate for the discrete nature of the recording, then use an algorithm to construct a volumetric polygon mesh from the point cloud. The resultant volume will be rough and contains significantly less polygons and avoids discretization holes. **Combine discrete positions**: Creates a single part combining the robot geometry in each discrete position of the recording. The part is processed to remove redundant polygons. The resultant volume will be bigger(greater memory usage) than the volume created using the **Construct from interpolated positions** option. While creating volume using this option, discrete positions only are included, hence, the result can contain holes especially when the speed is high. **Note** The RAM usage can be significantly high for longer recordings.
Enable measurement on stop position graphics	Enable measurement on the semitransparent robot.
Click on robot, tool or tooldata to measure stop distance	Measure the distance between a selected point on the robot and the corresponding stop position on the semitransparent robot.

Continues on next page

Option	Description
Use stop position in collision detection	Detect collision using the Category0 or Category 1 stop positions.  xx2000002209

Accessing the RAPID code from signal recording

Signal recordings are analyzed to find the root-cause of any unexpected robot events. From signal analyzer, it is possible to open the RAPID module that causes this robot event. Use the following procedure to access the RAPID code from signal analyzer. The *Target Changed* event must be recorded for using the following procedure.

Note

The *Target Changed* event data do not represent the current target, instead it represents the subsequent target.

1. In the **Recordings** window, right-click the signal recording and then click **Open**.

   The **Signal Analyzer** window opens with the selected signal recording.
2. Scroll down and access the **Events** table, and click the required RAPID event to view the **Current Target** details.
3. Click the **Location** link to open the respective RAPID module.

12 Jobs

Understanding jobs

Overview

RobotStudio is designed to work with one *robot controller* at a time. Use Jobs function to perform certain actions on a large population of robot controllers. Fleet or device set is a collection of selected set of controllers in a network that are identified by the IP Address or the DNS name. A Job is defined by a Device list and an Action. Action will be carried out for the selected controllers in the Device list.

The default set of actions are designed to monitor and to collect data from the controllers in the device list. This data can be analyzed for error detection and further rectification for maintaining uniform configuration across all controllers in a fleet.

Jobs tab

The Jobs feature is managed from the **Jobs** contextual tab. On the **Controller** tab, in the **Controller Tools** group, click **Jobs** to open the **Jobs** Contextual tab.

The **Jobs** contextual tab contains the following groups.

Groups	Description
Device Lists	A user-defined group of robot controllers for applying a job. Device lists can be reused between jobs.
Jobs	A command group containing various job options.

Device Lists group

The Device Lists group contains the following commands and controls.

Commands	Description
New Device List	Opens a new empty device list window for creating a new group of robot controllers.
Open Device List	Opens a previously saved device list to edit or review.
Save Device List	Saves the contents of an active device list window to disk in *.xlsx* format. These files can be edited in Microsoft Excel without changing the grid structure.
Scan Subnet	Populates the active device list window with all controllers that Netscan finds on the local subnet.

Jobs group

The Jobs group contains the following commands and controls.

Commands	Description
New job	Opens a new Job window.
Templates	Provides options, **Save Job** and **Edit Job Templates**. • **Save Job** : Saves the job specification as an *.xml* file. • **Edit Job Templates**: Allows you to edit an existing template file.
Verify	Verifies the status of the group of robot controllers.
Execute	Executes the user-selected action.

Continues on next page

Continued

Commands	Description
Pause	Temporarily stops an active action.
Resume	Resumes the paused action.
Cancel	Aborts the active job.

Device List window

You can define the population of robot controllers that the job should be applied to, using the **Device List** window. This window contains the metadata of all robot controllers that are part of the group.

- Network Address: This field is mandatory. IP address or DNS name of the controller. The controllers can be distributed on multiple subnets. You can add controllers that cannot be directly found by Netscan from the current location.
- Controller Name: This is an optional field. This value is used to verify the controller name that can be identified by the network address.
- System Name: This is an optional field. This value is used to verify the system name running on the controller that can be identified by the network address.
- Group: This is an optional field. This value is used to filter out a subset of robot controllers from a list while executing jobs.
- Subgroup: This is an optional field. This value is used to filter out a subset of robot controllers from a list while executing jobs.
- Comments: This is an optional field. You can type in comments in this field. If you add a controller to the list using the **Scan Subnet** command, then the system displays **Found by Netscan** message in this field.

Allow Execution State Running

You can not execute a job when the controller's program execution is in the *Running* state. This precaution is to avoid production getting disturbed when robots are performing sensitive *path* following applications such as laser cutting or arc welding.

Jobs such as backup or search *RAPID* data can put load on the controller. Select the **Allow Execution State Running** check box available in the **Device Selection** area while executing these jobs.

Allow System Failure state

When the **Allow System Failure state** check-box is selected, the job will be executed even if the connected controller is in system failure state. When this check-box is not selected, the jobs will not be executed if the controller is in system failure state.

Verify identity of OmniCore controllers

Refer to *Operating manual - Integrator's guide OmniCore* for details about certificate handling in OmniCore controllers. When the **Verify identity of OmniCore controllers** check-box is selected, the security certificate will be verified before executing the job.

Continues on next page

Supported actions

Using the Jobs feature, you can perform the following actions on a group of robot controllers. You must select the required action in the **Selected Action** list in the **Jobs** window. Some of these actions require the following additional data.

Action	Description
Backup	• Backup Path: User-defined destination folder for saving the backup file, this file contains controller specific backup folders with a date and time stamp in the format Backup_{Date}_{Time}. • Backup name: The name of the backup file in the format {SystemName}_{Date}. The name template can be modified according to the user needs. The strings in curly brackets will be replaced by the current values. **Note** The complete list of options to use for creating the file and folder names are {NetworkAddress}, {ControllerName}, {SystemName}, {SerialNumber}, {Comments}, {Group}, {Date} and {Time}.
Backup Program Modules	Creates backup of program modules with a date and time stamp in the format Backup_{Date}_{Time}. You can specify the name and destination folder of the backup program module in the **Backup name** and **Backup Path** fields.
Distribute Update Package	Copies the selected distribution package to the controller.
Update UAS	Updates the password of UAS file of the selected controller from the device list.
Set Time	Threshold (seconds): User-defined threshold time in seconds. The threshold defines the allowed time difference.
Verify Time	Reads the time for each controller and compares it with the local PC time.
Set Time Server and Time Zone	Select this action to set the time zone of the NTP server. Enter the name of the server in the **NTP Server** box and select the required time zone in the **Time Zone** list.
Save Event Logs	Reads the specified event logs and saves it to the specified location on the PC.
Search Event Logs	Searches the event logs for a specified type (All, Warnings and Errors, Errors) and upto an optional time(in days). You can also specify the range for the error codes to include in the search.

Continues on next page

Continued

Action	Description
Read Single Data	You can read RAPID Data, I/O Signal values, Configuration parameters and device information with this function. • For RAPID Data, you must specify the URL of the RAPID instance as `Task/Module/Data`or only `Task/Data`, for example, `T_ROB1/Module1/myToolData`, or `T_ROB1/myToolData`. The result will be the value of the instance. • For I/O Signals, you must specify the name of the signal, for example, *mySignal*. The result will be the signal value. • For configuration parameters, you must specify the URL to the instance attribute in the form `DOMAIN/TYPE/InstanceName/AttributeName`, for example `MOC/ARM_LOAD/r1_load_4/mass` or `EIO/EIO_SIGNAL/diMySignal/access`. • Select the **Devices** option, to read device information like *Main Computer Fan Speed*, *Main Computer Module Temperature*, *CPU Temperature* and *Free RAPID Memory(MB)* by selecting the required option in the **Copy From the device browser**. To read other properties, copy the device property ID from the **Device Browser**. Right-click the desired property and select **Copy device property ID** from the context menu. Paste the ID into the text field above.
Search RAPID Data	Searches for RAPID instances that matches the specified search patterns. You can restrict the search to *tasks*, *modules*, data types and names of record fields that match the specified pattern.
Search RAPID Text	Searches for lines that contain the specified text string. You can restrict the search to tasks or modules that match a certain name pattern.
Write File or Directory	Writes the selected file or directory to the specified target directory.
Read File or Directory	Reads the selected file or directory from *HOME* folder or from a task.
System Information	Reads the options, languages and media versions of the controllers.
Run External Tool	Invokes an external executable. • External Tool Path: Location of the folder where the external tool is placed. • Arguments: User specified arguments which the external tool passes, for example {SystemName}, { Network address},{Group} and so on. • Timeout(s): Specifies command time out period.
Compare Folder	Compares two folders and generates a report with the differences. These reports are available in two file formats, *Excel* and *xml*. These reports are not exclusive to Compare Folder.
Save System Diagnostics	Saves the system diagnostics file that contains the information about the current state of the controller.
Save Assessment Data	Saves the assessment data (Assessment_Data _{Date}_{Time}) of the device to the selected location. Assessment data contains usage statistics of a robot such as Production Time, Duty factor, Program speed, Cycle time, Workload stress, Operating hours and so on. This data is used for predictive maintenance of the robot (manipulator).

Jobs with several actions

- Use the **Add** button to create a job with several Actions.
- Use the **Up** and **Down** buttons to change the execution order. There will be one sheet per Action in the resulting Excel report.

Continues on next page

- Use the **Remove** button to remove the selected Action.
- The **Save** function, **Verify** and **Execute** are not specific to the multi-action Job.

Creating a device list

1. In the **Device Lists** group, click **New Device** List. The **Unnamed Device List** window opens.

   This device list can be saved in *.xlsx* format.
2. Enter the required details such as **Network Address**, **Controller Name** and so on. In the **Device Lists** group, click **Save Device List**.
3. To create a device list with all available devices in the network, click **Scan Subnet** the **Device Lists** group.

Creating a new job

1. On the **Controller** tab, in the **Controller Tools** group, click **Jobs**. The **Jobs** contextual tab opens.
2. In the **Device Lists** group, click **New Device** List. The **Unnamed Device List** window opens.

   This device list can be saved in *.xlsx* format.
3. In the **Device Selection** area, select a device list in the **Device List**. The **Group Filter** list will be populated if the selected list contains data in the **Group** field.
4. Enter valid credentials in the **Username** and **Password** boxes, or select **Default Credentials** if the **Default User** has sufficient grants to perform the selected action.

   The specified user must be available and this user should have sufficient grants for all controllers.
5. In the **Action** list, click the action that you want to perform. Depending on the selected action, additional action specific data might be required.

   To configure these actions, you must provide the required data.
6. In the **Jobs** group, click **Verify**/**Execute** to perform the selected action.

   Once the action gets completed, a report and a log file are created. You can open this report (in *.xlsx* format) from the **History browser**. The log file is used for troubleshooting and support.

Reuse a job template

Jobs are saved as xml files. These files can later be edited in RoboStudio. Job files must be placed in `C:\Users\rs_user\Documents\RobotStudio\JobTemplates` to be able to edit them. The files located in this location will be listed in the **Templates** drop down.

1. In the **Jobs** group, click **Templates**. Jobs available in the `C:\Users\rs_user\Documents\RobotStudio\JobTemplates` location will be listed here.
2. Click the particular job file to open.

Continues on next page

3 Customize the file and then click **Templates** and then click **Save Job** and save the file.

Running job from the Command Prompt window

Use the following steps to run a job or a batch of jobs from the Command Prompt window.

Step	Action
1	Create a Job and save it with a suitable name, for example *Job1*. For a default installation of RobotStudio, the job gets saved in `C:\Users\<user name>\Documents\RobotStudio\JobTemplates` folder as a **.xml* file.
2	Open **Notepad** and type in `"C:\Program Files (x86)\ABB\RobotStudio YYYY\Bin\Addins\FleetManagement\runjob.exe" "C:\Users\<user name>\Documents\RobotStudio\JobTemplates\Job1.xml" /defaultcredentials .` The example assumes that RobotStudio is installed in the default location. A specific user name and password can be supplied with the options `/user:<user name>` and `/password:<password>`.
3	Save the **.txt* file with the **.cmd* extension.
4	Double-click the **.cmd* file to run the job. Log files and reports get generated and is available in the **Jobs browser**.

Note

You can schedule jobs using the Windows built-in Task Scheduler.

The following table provides details of various command-line arguments along with their descriptions.

Arguments	Description
`/user:<user> /password:<pw>`	Run job with a specific username and password.
`/defaultCredentials`	Run job using the default user credentials.
`/logDirectory:<path>`	Directory for log and report files.
`/disableCertificateVerification`	Disable RobotWare 7 certificate verification.
`/allowExecStateRunning`	Allow execution state Running.

13 Screenmaker

13.1 Introduction to ScreenMaker

What is ScreenMaker?

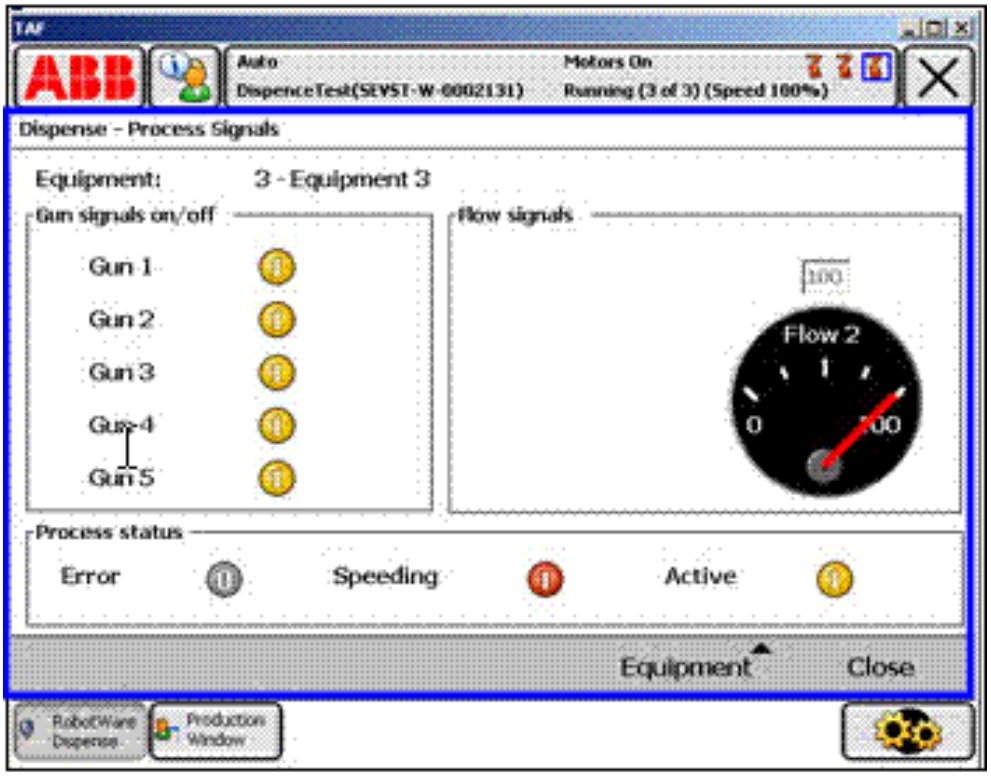
ScreenMaker is a tool in RobotStudio for developing custom screens. It is used to create customized FlexPendant GUIs without learning Visual Studio development environment and .NET programming.

> **Note**
>
> ScreenMaker is only relevant for systems using IRC5 controllers.

Why use ScreenMaker?

A customized operator interface on the factory floor is the key to a simple robotic system. A well-designed custom operator interface presents the right amount of information at the right time and in the right format to the user.

GUI concepts



xx0800000226

A GUI makes it easier for people to work with industrial robots by presenting a visual front end to the internal workings of a robotic system. For FlexPendant GUI applications, the graphical interface consists of a number of screens, each occupying the user window area (the blue box in the figure above) of the FlexPendant touch screen. A FlexPendant screen is then composed of a number

Continues on next page

of smaller graphical components in a design layout. Typical controls (sometimes referred as widgets or graphic components) include buttons, menus, images, and text fields.

A user interacts with a GUI application by:

- Clicking a button
- Selecting from a menu
- Typing a text in a text box
- Scrolling

An action such as clicking a button is called an event. Whenever an action is performed, an event is sent to the GUI application. The exact content of an event is solely dependent on the graphic component itself. Different components trigger different types of events. The GUI application responds to the events in the order generated by the user. This is called event-driven programming, since the main flow of a GUI application is dictated by events rather than being sequential from start to finish. Due to the unpredictability of the user's actions, one major task in developing a robust GUI application is to ensure that it works correctly no matter what the user does. Of course, a GUI application can, and actually does, ignore events that are irrelevant.

The event handler holds sets of actions to be executed after an event occurs. Similar to trap routines in the RAPID program, the event handler allows the implementation of application-specific logic, such as running a RAPID program, opening a gripper, processing logic or calculating.

In summary, from a developer's point of view, a GUI consists of at least two parts:

- *the view part*: layout and configuration of controls
- *the process part*: event handlers that respond to events

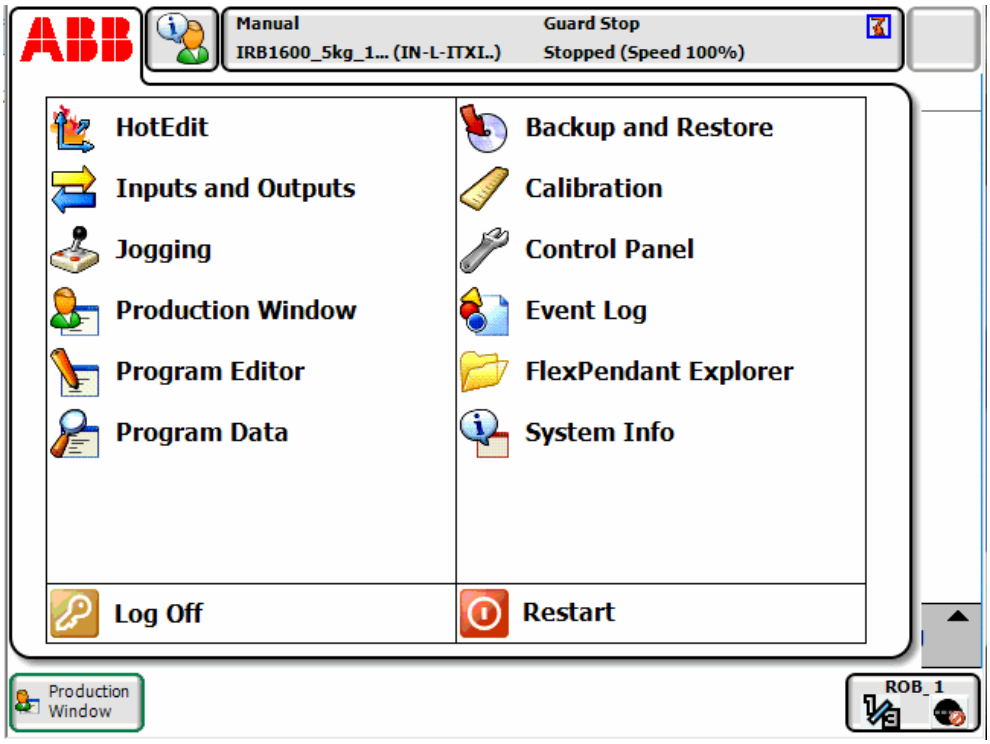
Modern GUI development environments often provide a form designer, a (What You See Is What You Get) WYSIWYG tool to allow the user to select, position and configure the widgets. As for event handlers, typically the developer must use a special programming language recommended by the development environment.

Note

ScreenMaker does not support Undo/Redo operations.

Continues on next page

FlexPendant concepts



xx0800000228

Running Windows CE, the ABB FlexPendant has limited CPU power and memory compared to a PC. A custom GUI application must therefore be placed in the designated folders on the controller hard drive before being loaded. Once loaded, it can be found in the ABB menu as seen in the figure above. Click the menu item to launch the GUI application.

As the robot controller is the one actually controlling the robot and its peripheral equipment by executing a RAPID program, a GUI application needs to communicate with the RAPID program server to read and write RAPID variables and set or reset I/O signals.

It is essential for RAPID programmers to understand that there are two different levels controlling a work cell: an event-driven GUI application running on the FlexPendant, and a sequential RAPID program running in the controller. These reside on different CPUs and use different operating systems, so communication and coordination are important and must be carefully designed.

Limitations

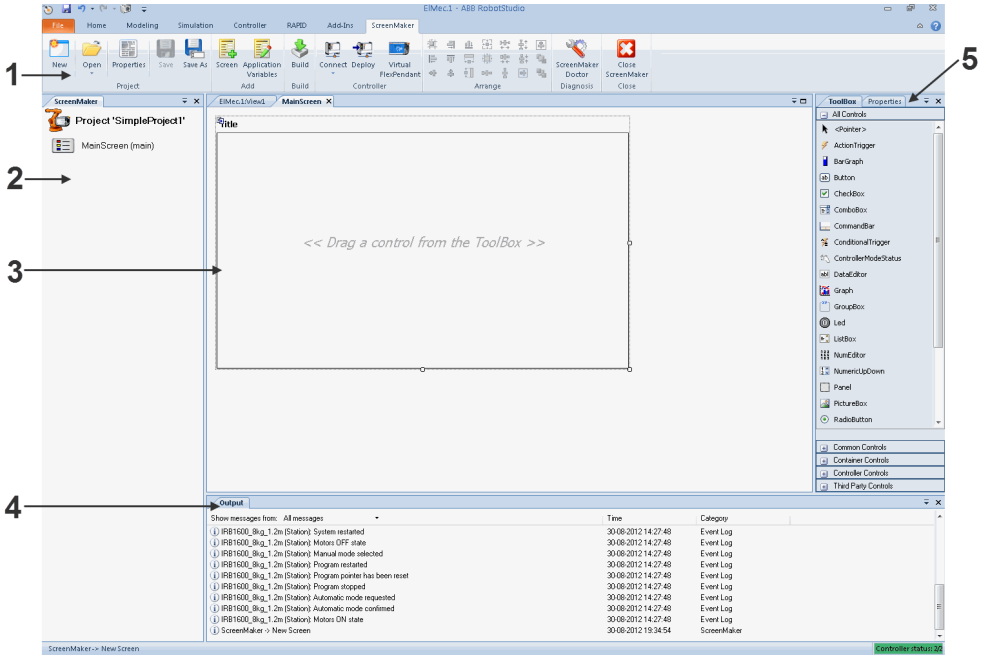
ScreenMaker supports English language when building the application in RobotStudio. ScreenMaker Designer does not provide a localization tool. Therefore, applications created with ScreenMaker display the same text specified at the design time, regardless of the choice of language on the FlexPendant.

When Asian languages are used (Chinese, Japanese, Korean), these screens display accurately only when the FlexPendant language matches with the the ScreenMaker language. Otherwise empty markers will be displayed instead of text characters.

13.2 Development environment

Overview

This section presents an overview of the ScreenMaker development environment for creating user screens.



en0900000584

	Parts	Description
1	Ribbon	Displays group of icons organized in a logical sequence of functions.
2	Project explorer	Shows the active screen project and lists the screens that are defined in the project.
3	Design area	Layout to design the screen with the available controls.
4	Output window	Displays information about the events that occur during ScreenMaker development.
5	ToolBox / Properties	Displays a list of available controls. Contains the available properties and events of the selected control(s). The value of the properties can either be a fixed value or a link to an IRC5 data or an Application Variable.

Ribbon

The ScreenMaker tab contains groups of commands organized in a logical sequence of functions that facilitates the user in managing ScreenMaker projects. The tab consists of the following groups:

Group	Functions used for
Project	Managing a ScreenMaker project. See Managing ScreenMaker Projects.

Continues on next page

Group	Functions used for
Add	Adding screen and application variables. See Managing screens and Managing Application Variables.
Build	Building a project. See Building a project.
Controller	Connecting and deploying to the controller. See Connecting to controller and Deploying to controller. Also for opening the **Virtual FlexPendant**.
Arrange	Re-sizing and positioning the controls on the design area.
Diagnosis	Detecting problems in the project and providing a diagnostic solution.
Close	Closing a project.

Arrange

This toolbar displays icons for resizing and positioning controls on the design area.

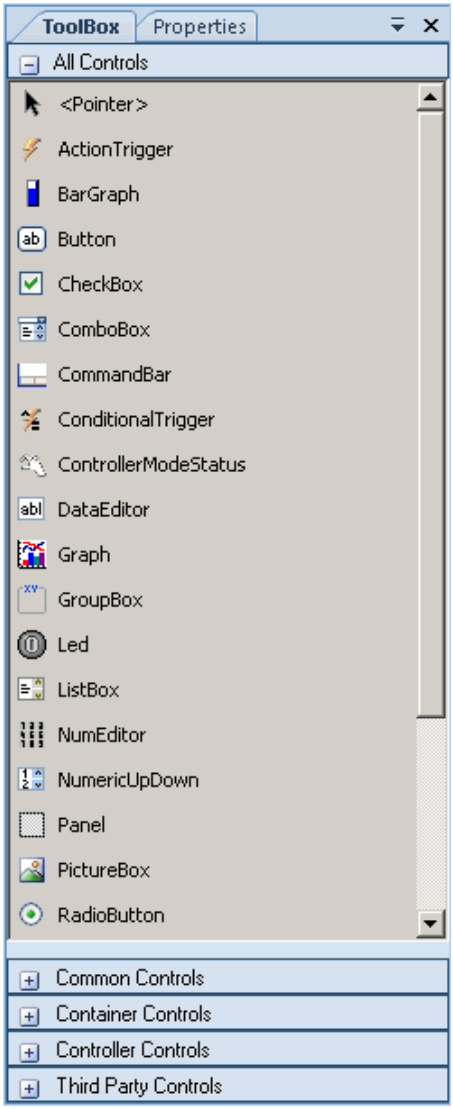
The icons are enabled once you select a control or group of controls on the design area.

en0900000592

Continues on next page

ToolBox

ToolBox acts a container for holding all the available controls that can be placed on a screen.



en0900000407

The following table displays the GUI controls that can be dragged to the design area.

Control	Description
ActionTrigger	Allows to run a list of actions when either a signal or rapid data changes.
BarGraph	Represents an analog value in a bar.
Button	Represents a control that can be clicked. Provides a simple way to trigger an event, and is commonly used to execute commands. It is labeled either with text or an image.
CheckBox	Allows multiple selections from a number of options. They are displayed as a square box with white space (for unselected) or as a tick mark (for selected).

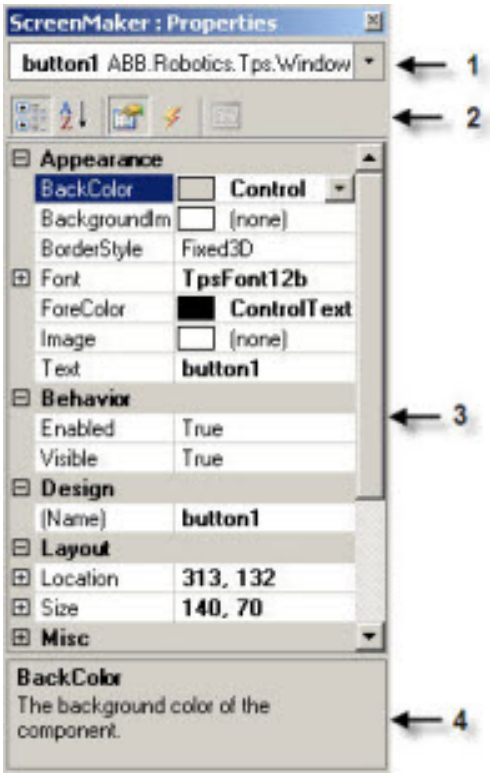
Continues on next page

Control	Description
ComboBox	Represents a control that enables to select items from a list. Combination of a drop-down list and a textbox. It allows you to either type a value directly into the control or choose from the list of existing options. It is not possible to add I/O signals to the combobox/listbox control.
CommandBar	Provides a menu system for a ScreenForm.
ConditionalTrigger	Allows to define conditions while defining action triggers. An action is triggered, if there is any change in value of the data bound.
ControllerModeStatus	Displays the mode of the Controller (Auto - Manual).
DataEditor	Represents a text box control that can be used to edit the data.
Graph	Represents a control that plots data with lines or bars.
GroupBox	Represents a Windows control that displays a frame around a group of controls with an optional caption. Is a container used to group a set of graphic components. It usually has a title at the top.
LED	Displays a two states value, like a Digital Signal.
ListBox	Represents a control to display a list of items. Allows the user to select one or more items from a list contained within a static, multiple line text box.
NumEditor	Represents a text box control that can be used to edit a number. When the user clicks it, a Numpad is opened. It is not recommended to add a NumEditor in a container control.
NumericUpDown	Represents a spin box that displays numeric values.
Panel	Used to group collection of controls.
PictureBox	Represents a picture box control that displays images.
RadioButton	Allows to select only one of a predefined set of options.
RapidExecutionStatus	Displays the execution status of the Controller Rapid Domain (Running - Auto).
RunRoutineButton	Represents a Windows button control that calls a RapidRoutine when clicked.
Switch	Displays and lets change a two states value, like a Digital Output Signal.
TabControl	Manages a set of tab pages.
TpsLabel	Very commonly used widget that displays text, a label is usually static, that is, it has nointeractivity. A label generally identifies a nearby text box or other graphic component.
VariantButton	Used to change the values of RAPID variables or Application variables.

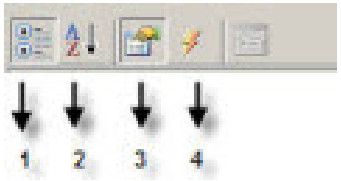
Continues on next page

Properties window

A control is characterized by its properties and events. Properties describe the appearance and behavior of the component, while events describe the ways in which a control notifies its internal state change to others. By changing the value of a property, the controls have a different look and feel, or exhibit different behavior.



en0900000408

	Element	Description
1	Graphical component name panel	Displays the selected component, and lists the available components of the active design screen.
2	Properties window toolbar	 en0900000409 1 Organizes table panel in categories 2 Organizes table panel alphabetically 3 Displays Properties in table panel 4 Displays Events in table panel
3	Table panel	Displays all the properties or events in two-columns. The first column shows the property or event name, the second shows the value of the property or name of the event handler.
4	Information panel	Display information about a property or event.

13.3 Working with ScreenMaker

13.3.1 Managing projects

Overview

This section describes how to manage projects in ScreenMaker. A complete cycle includes creating, saving, building, connecting, and deploying a ScreenMaker project.

You can manage a project (create, delete, load, or save) either from the ScreenMaker ribbon or the context menu.

Creating a new project

ScreenMaker does not support Unicode characters. Hence, do not use these characters when you create a ScreenMaker project.

Use the following procedure to create a new project:

1 Click **New** from the ScreenMaker ribbon or right-click **Project** context menu and select **New Project.**

The **New ScreenMaker Project** dialog box appears.

> **Note**
>
> You can create a new project either from *ScreenMaker installed templates* or *ScreenMaker custom templates*.

2 To create a new project from the *ScreenMaker installed templates*,
   a Click *Simple Project*.
   b Enter a name and specify the location for the new project. By default, the new project is saved on *C:\My Documents\RobotStudio\My ScreenMaker Projects.*
   c Click **OK**.
   d A screen *MainScreen(main)* is added in the tree view.

3 To create a new project from the **ScreenMaker custom templates**,
   a Click **Basic**, *Standard*, or **Extended**.
   b Enter a name and specify the location for the new project. By default, the new project is saved on **C:\My Documents\RobotStudio\My ScreenMaker Projects.**
   c Click **OK**.

> **Note**
>
> - If you select the template **Basic**, a project with two screens are created.
> - If you select the template **Standard**, a project with four screens are created.
> - If you select the template **Extended**, a project with six screens are created.

Continues on next page

13.3.1 Managing projects
Continued

Loading a project or template

Use this procedure to load an existing project or an existing template:

1. Click *Open* from the ScreenMaker ribbon or right-click **Project** context menu and select **Open Project.**

   The **Open Screen Project File** dialog box appears.

   WARNING

   A warning message appears when you open an existing ScreenMaker project where the FlexPendant SDK version is different from the version the project was created.

2. Browse to the location of the project file or template file to be loaded and click **Open.**

Note

You can also load an existing project using a quick access method.

1. Click **Recent** from the ScreenMaker ribbon or right-click **Project** context menu and select **Recent Projects**.
2. Select the project file from the list of most recently opened projects.

Saving a project

To save a project or template, follow this step:

- Click **Save** from the ScreenMaker ribbon or right-click **Project** context menu and select **Save**.

To save the existing project or template with a new name, follow this step:

- Click **SaveAs** from the ScreenMaker ribbon or right-click **Project** context menu and select **SaveAs**.

Note

- Project files are saved with the extension **.smk*.
- Template files are saved with the extension **.smt*.

SaveAs FlexPendant Project

To save the ScreenMaker project as a FlexPendant project, in the project context menu, click **SaveAs FlexPendant Project**.

The project is saved with the extension ***.csproj** which can be opened using Microsoft Visual Studio 2008.

Continues on next page

Designing screens

This section describes adding, copying, renaming, deleting, and editing a screen.

Overview

The Form designer is a tool to edit or design a screen. It allows you to design the screen with the required controls and the design area resembles a FlexPendant screen.

Editing a screen

To edit a screen, follow these steps:

1. Drag a control from the toolbox and drop it on the design area.

   The **Properties** window displays all the properties of the control.
2. Select the control and resize or reposition for configuration.

   Note

   You can either select a single control or multiple controls:
   - Single control : Left-click the control on the design area or select the control from the list in the Properties window.
   - Multiple controls: Left-click on the design area, drag the mouse and create a window selecting all the controls.
3. Click the smart tag on the upper right corner of the control to perform the basic tasks of configuration.

   Note

   You can perform additional configuration by editing the attributes in the Properties window.

Using ScreenMaker controls

This section describes building the GUIs using the following controls from the ToolBox.

ActionTrigger

An action trigger initiates an event, such as making a hidden object visible when an action is performed using a control. It allows to run a list of actions when the property value changes. The property value can be bound to a signal, rapid data, or application variable.

ActionTrigger control can also be used to invoke the application from RAPID.

Use this procedure to add an ActionTrigger control:

	Action
1	Drag an *ActionTrigger* control from the **ToolBox** on to the design area.

Continues on next page

	Action
2	You can modify the name, set the default value and configure data binding value for a ActionTrigger control. • Set the values of a property in the *Properties window*. • You can set the trigger event for an ActionTrigger to any of the event handler created either from a control or from an Events Manager option. • Configure the data binding values using Configuring Data Binding. • Set the application variables using the Managing Application Variables.

Note

An action is not triggered when the screen is launched for the first time, but is triggered when there is a difference in the bound value at any point of time. This functionality is supported only in RobotWare 5.12.02 or higher.

Example: Consider a signal being bound to the value property. The value of the signal changes at runtime on performing a specific action. The event handler configured for ActionTrigger control gets triggered based on this signal value change.

TpsLabel

TpsLabel is a standard Windows label that displays a descriptive text.

Use this procedure to add a TpsLabel control:

Step	Action
1	Drag a **TpsLabel** control from the **ToolBox** on to the design area.
2	You can set the values, setup events, configure data binding values and set the application values for a TpsLabel control. • Set the values of a property in the *Properties window*. • Set up the events , see *Setup Events*. • To configure the data binding values, see *Configuring Data Binding*. • To set the application variables, see *Managing Application Variables*.
3	You can set the option Allow Multiple States to true and change the property. 1 Click Allow Multiple States. The StatesEditor dialog box appears. 2 Click the check-box Allow Multi-States, select the properties to change from Properties For States and click OK.

The controls Button, PictureBox, and TpsLabel support AllowMultipleStates. For more information on how to use AllowMultipleStates, see *Picture object and changing images due to I/O*.

Panel

Panel is used to group a collection of controls.

Use this procedure to add a Panel control:

Step	Action
1	Drag a Panel control from the ToolBox on to the design area.
2	You can add a group of controls to a panel.

Continues on next page

Step	Action
3	You can modify the name, set the default value and binding value for a Panel control. • To set the values of a property, see *Properties window*. • To set up the events, see *Setup Events* . • To configure the data binding values, see *Configuring Data Binding*. • To set the application variables, see *Managing Application Variables*.

Note

Currently only EventHandler, CancelEventHandlers, and MouseEventArgs are supported.

ControllerModeStatus

ControllerModeStatus displays the mode of the controller (Auto - Manual).

Use this procedure to add a ControllerModeStatus control:

Step	Action
1	Drag a ControllerModeStatuscontrol from the ToolBox on to the design area.
2	You can set the values, setup events, configure data binding values, and set the application variables for a ControllerModeStatus control. • To set the values of a property, see *Properties window*. • To set up the events, see *Setup Events*. • To configure the data binding values, see *Configuring Data Binding*. • To set the application variables, see *Managing Application Variables*.
3	You can select the image to be displayed when the controller is in Auto mode and in Manual mode. • Click AutoImage in the Properties window and browse to select the image to be displayed in Auto mode. • Click ManualImage in the Properties window and browse to select the image to be displayed in Manual mode.

RapidExecutionStatus

RapidExecutionStatus displays the execution status of the Controller Rapid Domain (Running - Auto).

Use this procedure to add a RapidExecutionStatus control:

Step	Action
1	Drag a RapidExecutionStatus control from the ToolBox on to the design area.
2	You can set the values, setup events, configure data binding values, and set the application values for a RapidExecutionStatus control. • To set the values of a property, see *Properties window*. • To set up the events, see Setup Events. • To configure the data binding values, see *Configuring Data Binding*. • To set the application variables, see *Managing Application Variables*.
3	You can select the image to be displayed when the Program is running and is stopped. • Click RunningImage in the Properties window and browse to select the image to be displayed when the Program is running. • Click StoppedImage in the Properties window and browse to select the image to be displayed when the Program is stopped.

Continues on next page

RunRoutineButton

RunRoutineButton represents a Windows button that calls a RapidRoutine when clicked.

> **Note**
>
> To call a routine containing movements, you are not recommended to use the RunRoutine Button control. Instead use a normal button control to call a Trap routine. In the Trap routine, use instructions such as StopMove, StorePath, RestorePath and StartMove to control the movements of the robot.

Use this procedure to add a RunRoutineButton control:

Step	Action
1	Drag a RunRoutineButton control from the ToolBox on to the design area.
2	Click the smart tag on the RunRoutineButton and select one of the following RunRoutineButtonTasks. • Define Actions before calling Routine • Select Routine to call • Define Actions after calling Routine
3	Click Define Actions before calling Routine to define an action/event before calling the routine. The Events Panel dialog box appears.
4	Click Define Actions after calling Routine to define an action/event after calling the routine. The Events Panel dialog box appears.
5	Click Select Routine to call. The Controller Object Binding dialog box appears.
6	In the Properties window, set the value for the following properties: • RoutineToCall - Set the routine to be called. Indicates the RAPID Routine that will be called when this button is pressed. • AllowInAuto - Set to True or False. Indicates if the routine could be called in the Auto mode. • TextAlign - Set to MiddleLeft and MiddleCenter. Indicates the text alignment. Note the following restrictions: • You cannot bind RunRoutineButton to built-in Service routines. • Only user defined procedures with no arguments can be bound. • Set the PP to task before performing action through RunRoutineButton.

CommandBar

CommandBar allows you to add menu items in a controlled and organized order.

Use this procedure to add menu items to the CommandBar control:

Step	Action
1	Drag a CommandBar control from the ToolBox on to the design area. The CommandBar appear at the bottom of the screen.
2	Click the smart tag on the CommandBar and select Add/Remove Items. The MenuItem Collection Editor window appears.

Continues on next page

Step	Action
3	Click Add. A new menu item is added and its properties are displayed which can be edited. Note that while editing the menu item, ensure that the property Text is filled. If not, nothing appears on the CommandBar.
4	To remove the menu item, select the menu item and click Remove.
5	Click Close to close the MenuItem Collection Editor window.

To add an event to a menu item, for example *menuItem1* on the command bar, use this procedure:

Step	Action
1	Go to the Properties window and select *menuItem1* from the drop-down list.
2	Click Events icon and then double-click the Click event. This opens the Events Panel dialog for the Click event.
3	Click Add Action from the Events Panel dialog. This opens a sub-list of actions.
4	Click an action from the sub-list of actions to add it to *menuItem1*'s Click event.

Note

ScreenMaker does not support the FlexPendant controls feature to add sub menu items on CommandBar.

VariantButton

The VariantButton control is a simple button control with additional features and properties. Using this control, you can change the values of RAPID or Application variables.

Use this procedure to add the VariantButton control:

Step	Action
1	Drag a VariantButton control from the ToolBox on to the design area.
2	You can perform the following VariantButton tasks from the SmartTag: • Define Actions before value change • Define Actions after value change
3	You can set the following VariantButton specific properties from the Properties window: • Select Increment or Decrement from Behavior drop down. The default behavior of VariantButton is Increment. • Select StepRate and set the rate at which the value must be varied. • Select DataType to which the value should be bound and set the value property of the selected datatype. Supports only the RAPID datatypes, **Num** and **Dnum**. For more information on data binding, see *Configuring Data Binding*.
4	You can also perform the following common tasks from the Properties windows: • Set BackColor, ForeColor, Location, and Size of the control. • Select True or False from the Visible dropdown to hide or unhide the control. • Select True or False from the Enabled drop down to enable or disable the control.

Continues on next page

13.3.1 Managing projects
Continued

ConditionalTrigger

The ConditionalTrigger button defines the condition while defining action triggers. An action will be triggered if there is a change in the value of the data bound.

Use this prcodedure to add the ConditionalTrigger control:

Step	Action
1	Drag a ConditionalTrigger control from the ToolBox on to the design area.
2	You can set the following ConditionalTrigger properties from the Properties window: • Select the condition to execute from the Condition drop down. The following are the supported conditions AND, OR, XOR, NOT, and EQUAL. • Select True or False from the Enabled drop down to enable or disable the control. • Select LHS and RHS and bind the data value to Controller Object or Application Variable.

Defining Events

Event handler is a set of actions to be executed after an event occurs.

To set up an event, follow these steps:

1 Select the control for which the event handler is to be defined.
2 Open the **Events Panel** dialog box in any one of the following ways:
   - Double-click the control.
   - Right-click the control, select **Events Manager**, click **Add** enter the name, and click **OK** and close.
   - Click smart tag and select the task from the list.
   - In the **Properties** window, click **Events** icon and select the desired event from the list.
3 Click **Add Action** to add an action from a predefined list of actions.

   The following table lists the set of predefined actions:

Screens	• Open Screen • Close Screen
Signals	• Set a Digital Signal • Invert a Digital Signal • Pulse a Digital Signal • Read a Signal • Write a Signal • Reset a Digital Signal
RapidData	• Read a Rapid Data • Write a Rapid Data
Application Variable	• Read and Write
Advanced	• Call another Action list • Call .NET method • Call Custom Action • Call FP Standard View

4 Select the action from the left window and perform the following:
   - Click **Delete** to delete the action.
   - Click **Move Up** or **Move Down** to change the order of execution of actions.

Continues on next page

5 Click **OK**

Deleting an event handler

To delete a user created event handler, do the following:

1 Right-click the control, select **Events Manager**. The **Events Manager** dialog box appears.
2 Select the event handler to be deleted from the list and click **Delete**.

Advanced Actions

Call another Action List

Existing event handlers from Events Manager can be reused by other controls while defining actions for event. You can call another event handler from an existing event handler.

In the following example, **listbox1_SelectedIndexChanged** event handler is called from **comboBox1_SelectionIndexChanged** event handler.

Select the *Show warning message before performing actions* check box to have a warning displayed before you can perform these actions.

Call .NET Method

You can import the dlls and add references to the *Advanced* tab of the **Project Properties** dialog box.

Once the references are defined, .NET methods appear in the *Project Properties* dialog box and can be included in the *Actions* list which will be executed on performing the desired action.

The .NET assembly supports only public static methods.

Double click the method and bind the return value to the application variable.

Binding can be done only to the application variable.

Note

ScreenMaker allows you to call static methods of the public classes defined in another DLL. This DLL is usually a class library or a control library. It has the following limitations and the user should be aware of them while using .Net DLLs.

- DLL's references must be in the same directory in order to load the DLL.
- ScreenMaker provides access only to the static methods which contain basic data types such as string, int, double, boolean, object.

The following procedure provides information on creating a .NET assembly. This assembly can be added as a reference to ScreenMaker Project and for performing certain computations which are not directly possible using ScreenMaker or to call methods of FlexPendant or PCSDK.

Use Visual Studio 2010 or above to create a .NET assembly.

1 Create a new project with Class Library as your template.
2 Create public static methods like the following.

```
namespace SMDotNetMethods
{
public class Methods
```

Continues on next page

13.3.1 Managing projects
Continued

```
{
/// <summary>
/// Inverts a boolean value
/// </summary>
/// <param name ="Value">input boolean value</param>
/// <returns>inverted boolean value</returns>
public static bool InvertBool(bool value)
{
return (value == false);
}

/// <summary>
/// Increments a numerical value
/// </summary>
/// <param name="value">value to be incremented</param>
/// <returns>incremented value</returns>
public static double Increment(double value)
{
return (value + 1);
}
}
}
```

3 Build the project.
4 Use the assembly generated from this Class Library project.
5 Add it as a reference to the ScreenMaker project.

Call Custom Action

You can add an user control to the *ScreenMaker toolbox* and call a custom method for that control by defining it in the *ScreenMaker.dll.config file*.

Call Custom Action supports only the Graph control.

Call FP Standard View

Standard FlexPendant screens can be opened on any action performed on the control. The standard FlexPendant screens include Rapid Editor, Rapid Data, LogOff, Jogging, Backup and Restore.

For example, on button1_click, Rapid Editor view is opened.

Editing the property value

You can edit the property value of a control from the *Properties window* in three ways:

1 By typing the numerics, strings and text. For example, Location, Size, Name etc.
2 By selecting the predefined values from the list. For example, BackColor, Font etc.
3 By entering the values in the dialog box. For example, Enabled, States, BaseValue etc.

Continues on next page

Deleting an event handler

To delete a user created event handler, do the following:

1. Right-click the control, select **Events Manager**. The **Events Manager** dialog box appears.
2. Select the event handler to be deleted from the list and click **Delete**.

Modifying Project properties

Project properties define the properties of the ScreenMaker project, including how the GUI is loaded and displayed in the FlexPendant.

Use this procedure to modify the project properties:

1. Right-click **Project** context menu and select **Properties**.

   The **Project Properties** dialog box appears.
2. In the **Display** tab under **Caption**, enter the text in the **Caption of the Application** field to edit the caption.

   The updated caption appears in the **ABB Menu** on the right side.
3. In the **Display** tab under **ABB Menu**, select the following options,

Option	Description
Left	application is visible to the left in the ABB Menu.
Right	application is visible to the right in the ABB Menu.
None	application is not visible at all in the ABB Menu.

> **Note**
>
> The applications that uses the option **None** cannot be run on RobotWare releases earlier than 5.11.01.

4. In the **Display** tab under **ABB Menu**, browse and select the **ABB menu image**.
5. In the **Display** tab under **TaskBar**, browse and select the **TaskBar image**.

> **Note**
>
> By default, the **Use Default Image** and **Use Menu Image** checkbox is enabled and the deafult image *tpu-Operator32.gif* is selected.

6. In the **Display** tab under **Startup** , select **Automatic** to load the screen automatically at the Startup.

> **Note**
>
> By default, the start up type is **Manual**.

7. In the **Advanced** tab under **Run Settings**, select **Launch virtual FlexPendant after deploying** checkbox.

Continues on next page

The virtual FlexPendant will be launched after deploying the ScreenMaker project to the virtual controller.

> **Note**
>
> This feature is not applicable if connected to a robot controller.

8 In the **Project Properties** dialog, select the **General** tab to view the project properties which includes, **Name**, **Assembly**, **Version**, and **Path**.

Version displays the specific versions of Controller and FlexPendant SDK that the ScreenMaker project uses.

Connecting to controller

Use this procedure to connect to both robot controller and virtual controller:

1 Click **Connect** from the ScreenMaker ribbon or right-click **Project** context menu and select **Connect**.

The **Select a Robot Controller** dialog box appears.

> **Note**
>
> Click the **Connect** dropdown from the ScreenMaker ribbon to directly connect to the controller.

2 Click **Refresh** to find a list of all the available controllers.

> **Note**
>
> By default, the currently connected controller is highlighted and has a small icon before the row as an indicator.

3 Select the controller to be connected from the list and click *Connect*.

The connection status is displayed in the Project tree view.

To remove the connection with the controller, click **Disconnect** from the Project context menu.

Building a project

The result from building the ScreenMaker project is a set of files including DLL file and images. The ScreenMaker project can be compiled into binary format (.dll) that can be deployed on a FlexPendant.

Use this procedure to build a project:

1 Click **Build** from the ScreenMaker ribbon or right-click Project context menu and select **Build**.

The result is displayed in the output window.

Continues on next page

Deploying to controller

Use this procedure to deploy a ScreenMaker project to a robot controller or virtual controller:

1. Connect to the controller you want to deploy to.
2. Click **Deploy** from the ScreenMaker ribbon or right-click Project context menu and select **Deploy Screen to Controller**.

   The **Download** dialog box appears displaying the progress of download. It disappears once the download is successful.

   The **TpsViewxxxxxx.dll** file is downloaded.
3. Restart the controller.

> **Note**
>
> - If a robot controller is used, you can reboot the FlexPendant by moving its joystick three times to the right, once to the left, and once towards you.
> - If a virtual controller is used, you can reboot the FlexPendant by closing the virtual FlexPendant window.

Closing a project

To close a project, follow this step:

- Right-click **Project** context menu and select **Close Project**.

Closing ScreenMaker

To close ScreenMaker, follow this step:

- Click **Close ScreenMaker** from the ScreenMaker ribbon.

Managing ScreenMaker Widgets

What is a widget

A widget is a visual building block, containing an information arrangement, which represents an aspect of a robot application. It is a reusable and sharable user interface building block which can help speed up the development of screens.

A ScreenMaker widget is similar in function to the widgets used in computer programming. The widget is an element of a graphical user interface (GUI) that displays an information arrangement which is changeable by the user. The widgets, when combined in an application, hold data processed by the application and the available interactions on this data.

Widget Workflow

Widget created from ScreenMaker can be used in ScreenMaker application and in Production Screen application.

The following are the steps required to create a Widget in ScreenMaker.

1. Start RobotStudio.
2. Launch ScreenMaker.
3. Create a new **Widget Project** or open an existing widget project.

Continues on next page

13.3.1 Managing projects
Continued

For information on how to create a new widget project.

4 Connect to a robot or a virtual controller, as required.
5 If required, change the widget properties, using the **Widget Properties** dialog box.

   For information on the Widget Properties dialog box.

6 Drag-and-drop the necessary user interface components, as you would in a normal ScreenMaker project.
7 Link the user interface properties to the IRC5 data or to the application variables
8 Build the widget project. The widget component is created and saved in `...\Documents\RobotStudio\Widget Components` folder.

Sample use case

Consider a case where you want to design a production screen which can do the following:

- Display a graph
- Show alarms
- Show status of the controller

To achieve this:

1 Create a new widget project in ScreenMaker and names it as, for example, GraphWidget.
2 Drags-and-drop the graph control and other necessary controls on the widget form.
3 Connect to a robot controller or virtual controller, as required.
4 Bind the controls to the controller data.
5 Use the widget properties dialog box, to change the size of the widget.
6 Build the project.
7 Download the output to the production screen.

You can then repeat the above steps to create widgets either in the same or in different projects based on your need to show the alarms and the status of the controller.

Creating a ScreenMaker widget project

1 On the ScreenMaker tab, click **New**. Alternatively, in the project context menu, click **New Project**.

   The **New ScreenMaker Project** dialog box appears.

2 Under Widget Templates, click **Widget**.
3 Specify a name for the widget project.

   ScreenMaker widgets projects are by default stored in the `...\Documents\RobotStudio\Widget Projects` folder.

4 Click **OK**.

Continues on next page

The widget project along with a screen **MainScreen(main)**, appears in the tree view. The widget project has **.wzp** file name extension. Widgets also appear in the **Toolbox**.

Note

- You can open one widget project at any time. Close an open widget project before opening a new one.
- A widget project has only one screen, the main screen, on which the widgets are designed. All controls defined on a widget are considered as one widget.
- Widgets are loaded into the toolbox from a folder that contains the widget component DLLs, from Additional Options folder under MediaPool. If you delete the widget components from these locations (`...\Documents\RobotStudio\Widget Components`), then the widgets will not appear in the Toolbox.

Creating Production Screen Widget

ScreenMaker helps the user to create two types of widgets, Production Screen widget and Standard widget. Controls in a widget can be bound to rapid or signal data.

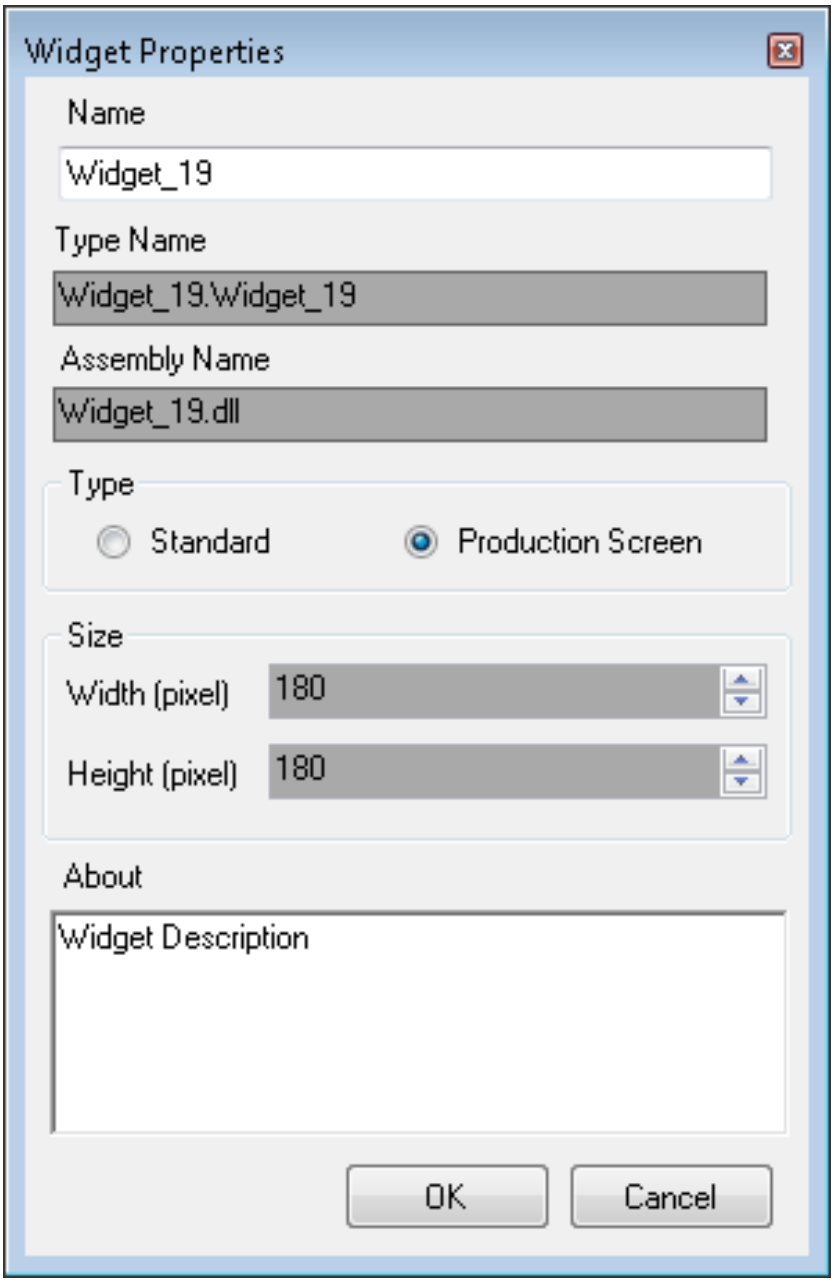
The Production Screen option is a framework for creating a customized GUI that can be used to present process data and status and execute FlexPendant applications.

To run widgets on the Production Screen, FlexPendant Interface option must be selected. Use the following procedure to create the Production Screen widget.

1. In the **Screenmaker** ribbon, select **New**. The **New Project** dialog opens.
2. Select **Widget Template** to create a new widget project.
3. Drag and drop controls to the widget.
4. Select **Widget Properties**, **Widget Properties** dialog opens.
5. Under **Type**, click **Production Screen** and click **OK**.

Continues on next page

6 Build the project.



xx1400000275

The *ProductionSetup.xml* file must be updated with widget details to view the widget that was created in the Production Screen. You can find *ProductionSetup.xml* under `$System\HOME\ProdScr` and Widget components under *$System\HOME\ProdScr\tps*.

An example of widget detail is provided here:

```
<Widget>
<Name>Widget_9</Name>
<Page>1</Page>
<Assembly>Widget_9.dll</Assembly>
<Type>Widget_9.Widget_9</Type>
```

Continues on next page

```
<Position>
<X>1</X>
<Y>2<Y>
</Position>
<ZIndex>1</ZIndex>
<Bindings>
<Binding PropertyName ="led1.Value" BindingType="SIGNAL"
     DataName="MOTLMP" />
<Binding PropertyName ="button1.Text" BindingType="RAPID"
     DataName="T_ROB1/BASE/wobj0" />
</Bindings>
</Widget>
```

The production screen provides the flexibility to modify bindings of the widget. This is provided under Bindings tag as shown here:

```
<Bindings>
<Binding PropertyName ="led1.Value" BindingType="SIGNAL"
     DataName="MOTLMP" />
<Binding PropertyName ="button1.Text" BindingType="RAPID"
     DataName="T_ROB1/BASE/wobj0" />
</Bindings>
```

Specifying widget properties

To specify the properties of a widget project, right-click a widget project, and then click **Properties**. The Widget Properties dialog box appears.

You set and modify the following in the properties for the widget project:

- Name of the project
- Size of the widget - x,y (in mm)
- Select the type of Widget
  - Production Screen: The Widget can be used with Production Screen environment
  - ScreenMaker: The Widget can be used with ScreenMaker applications

Modifying Binding Information of Widget

Use this option to modify the binding information of widget. When a widget is built from the Widget Project, an xml file is created. This xml contains widget details and binding information. This entry must be vailable in the *Production.xml* file to work with the Production Screen Environment.

```
<Bindings>
<Binding PropertyName ="meter1.Value" BindingType="IO"
     DataName="aoMeterSignal" />
<Binding PropertyName ="meter1.Title" BindingType="RAPID"
     DataName="Flow1Title" />
</Bindings>
```

It is possible to create, use and modify the bindings of a widget created from ScreenMaker and to view the results in Production Screen and in ScreenMaker application environment.

Continues on next page

13.3.1 Managing projects
Continued

Building and Deploying

The output of the Widget Project is a single Widget Component dll file, for example, *TpsViewMyWidget.dll*. The widgets built from the Widget Project are used in the ScreenMaker project. Widgets cannot be deployed to the controller from ScreenMaker. If Widgets are used in ScreenMaker projects, it gets deployed.

When the ScreenMaker project which uses a widget is built, the widget component is added as a reference to the project.

When the ScreenMaker project output is deployed to the controller, the referenced widget components are also copied to the system *HOME* folder.

13.3.2 Application variables

Overview

Application Variables are variables defined inside a ScreenMaker application. An application variable is similar to a RAPID variable. It supports the data types supported by RAPID, such as num, dnum, string, tooldata, wobjdata, and so on.

An application variable's definition includes its name, data type and initial value. During the execution of the ScreenMaker application, an application variable has a persistent value. It can store values coming from controller data or can be used to write values to controller data. Therefore, it is like an intermediate persistent variable which is used during RAPID execution along with other RAPID variables.

Managing application variables

To create, delete, and rename an application variable, follow these steps:

1. On the **ScreenMaker** tab, in the **Add** group, click **Application Variables**.

   Alternatively, in the **ScreenMaker** browser, right-click the project, and then click **Application Variables**.

   The **Project Application Variables** dialog box appears.
2. Click **Add** and define the name, type and value of the new variable.
3. Select the variable, click **Delete** to delete a variable.
4. Select the variable, click **Rename**, enter the new name and click **OK** to rename a variable.
5. Click **Close**.

You can view the application variables related to a project listed in the **Project Application Variables** dialog box. To filter and view the variables according to their data types, use the **Type** list.

13.3.3 Data binding

Overview

Data binding is the mechanism that links a GUI property with an external data source such that whenever the data source is updated, the GUI property will be updated automatically and vice versa.

There are two ways of linking the data with the GUI properties:

- Controller object data binding
- Application variable data binding

Configuring data binding

Data binding can be configured using the Properties window.

Using Properties window

1 On the design area, select the control.
2 In the Properties window, locate the row from the table for binding the value.
3 Select the property and click the list to display the Binding menu.

Click...	to...
Remove actual binding	remove the existing data binding.
Bind to a Controller object	select available data in the controller for binding.
Bind to an Application variable	select available data in the project *Application Variables* for binding.
Bind to an Array	select available RAPID array in the controller for binding.

Configuring data binding for different controls

Binding to an array can be done with the following controls:

Control	Description
DataEditor	The default index value is 1. DataEditor is designed in such a way that the default value of the Rapid array starts with 1 and not 0.
ComboBox and ListBox	The default index value is -1. You can enter the appropriate index value but cannot bind to a controller object or an application variable. Note the following: • You can limit the number of items to be displayed in the ComboBox and ListBox of an array. • While using a ComboBox, a RAPID index starts with 1 (1 specifies the first element) and the ComboBox index starts with 0(0 specifies the first index). • When adding items to the ListBox or ComboBox control, it is not possible to add I/O signals.

Controller object data binding

Controller object data binding lets you to select the data in the controller for binding.

Use the following procedure to set up a binding with controller objects.

Open the Controller Object Binding dialog box and follow these steps:

1 In the **Type of Object** group, select either **Rapid data** or **Signal data**.

Continues on next page

2 In the **Shared** group, select **Built-in data only** to access shared **Rapid data**.

When you select **Built-in data only**, the option **Signal data** and the text box **Module** are disabled.

3 If you have selected **Rapid data**, then in the **Scope** group, you can select a task and module from the list.

When you select **Signal data**, the **Scope** group is disabled.

4 In the **See** list, select the desired data.

CAUTION

Unwanted toggle of I/O Signals & RAPID data when using data binding to Enabled properties.

When you bind the Enabled property to a Controller object, and a dialog (like the confirm Going to Auto) is displayed on top, then your screen will be disabled and all the Enabled properties of all its controls will be set to false and all bound Controller objects with it.

To avoid this behavior it is recommended to bind Enabled properties to I/O Signals with Access Level: ReadOnly as this results in a one way binding.

Note

ScreenMaker supports binding to only constant and persistant variables. The variables must not be declared LOCAL. TASK PERS is supported.

For example, the following binding is supported:

```
PERS num n1:=0;
TASK PERS num n2:=0;
CONST num n3:=0;
```

The following binding is not supported:

```
LOCAL PERS num n1:=0;
VAR num n1:=0
```

Note

A data bound RAPID array should be declared as PERS, it will not work as CONST.

Application variable data binding

Use the following procedure to set up a binding with project application variables.

Open the Application Variables Bind Form dialog box and follow these steps:

1 Select an application variable and the field to connect.

2 Click **Setup Variables** to manage the variables.

The **Project Application Variables** dialog box appears.

3 Click **OK**.

13.3.4 ScreenMaker Doctor

Overview

ScreenMaker Doctor is a diagnostic solution to detect problems in the ScreenMaker project. It helps analyze the project and fix errors such as:

- Unused events
- Broken references, application variables, signals, modules, and Rapid data
- RunRoutine issue

Using ScreenMaker Doctor

Use this procedure to launch ScreenMaker Doctor, detect and report issues, and to view causes and solutions:

1. In the ScreenMaker ribbon, click **ScreenMaker Doctor**.

   The **ScreenMaker Doctor** Wizard opens.
2. Click **Next**.

   The wizard starts detecting issues and are reported as **Completed Checks**. The detected issues are categorized as:
   - Broken References
   - Unused Events
   - Broken ApplicationVariables
   - Broken Signals
   - Broken Modules
   - Broken RapidData
   - RunRoutine issue
   - Broken Routine
   - Other Dependencies
3. Click **View Causes and Solutions** to generate a report.

   The left hand side of the report displays issues under each category and the right hand side of the report displays the Probable Causes and Solutions for the issues.

   To check for issues again using the same instance, click **Re-Detect Issues**.

> **Note**
>
> In order to detect the signal data and RAPID, ScreenMaker project should be connected to the controller.

Errors fixed by ScreenMaker Doctor

The following sections show you how errors, which can be fixed by ScreenMaker Doctor, may manifest.

Unused Events

The following sequence of actions will result in creating unused events.

1. Create a ScreenMaker project.

Continues on next page

2 Define events for the controls.
3 Define the events *Button1_Click* and *Button2_Click* for the controls *Button1*and *Button2* respectively.
4 Delete the control *Button1*. The event *Button1_Click* will still exist. An unused event is created.

You can run ScreenMaker Doctor to detect and fix this error.

Broken Reference

The following sequence of actions will result in creating broken references.

1 Create a ScreenMaker project.
2 Define events for the controls.
3 Define the events *Button1_Click* and *Button2_Click* for the controls *Button1*and *Button2* respectively.
4 Define action *ScreenOpen - Screen2* for the event *Button1_Click*.
5 Delete or rename the screen. A broken reference is created.

You can run ScreenMaker Doctor to detect and fix this error.

Broken Application Variables

The following sequence of actions will result in creating broken application variables.

1 Create a ScreenMaker project.
2 Add an Application variable to the project.
3 Rename or delete the Application variable. No error is reported.

An error is reported during the run time due to the broken application variable.

You can run ScreenMaker Doctor to detect and fix this error.

Broken Rapid Data/Signals

If rapid data is bound but not found in the controller connected in the ScreenMaker project, then perform the following procedure:

1 Create a ScreenMaker project.
2 Connect to a controller.
3 Bind the properties of the controls with controller data.
4 Build the project and deploy it to the controller.

   The application works.
5 Connect the ScreenMaker project to another controller and deploy the same project.

   The application produces errors in the FlexPendant.
6 Run ScreenMaker Doctor. It detects that RapidData is not found in the controller, thereby suggesting to define the same.

Broken Modules

If modules are bound but not found in the controller connected in the ScreenMaker project, then perform the following procedure:

1 Create a ScreenMaker project.
2 Connect to a controller.
3 Bind the properties of the controls with controller data.

Continues on next page

13.3.4 ScreenMaker Doctor
Continued

4 Build the project and deploy to controller.

   The application works.

5 Connect the ScreenMaker project to another controller and deploy the same.

   The application produces errors in the FlexPendant.

6 Run ScreenMaker Doctor.

   It detects that the module in which the rapid data was defined is not found in the controller, thereby suggesting to define the same. ScreenMaker doctor also detects Hidden modules.

RunRoutine Issue

A check is made whether *ScreenMaker.sys* file is loaded on the controller or not. An issue is detected if the system module is not loaded.

You can run ScreenMaker Doctor to detect and fix this error.

A *System.NullReferenceException* appears if *ScreenMaker.sys* entry is not available in *SYS.CFG* file of robot system. To overcome this issue, add the following entry under *CAB_TASKS_MODULES* in the *SYS.CFG* file and save and load the modified file back into the robot system and then restart the robot system.

```
File "RELEASE:/options/gtpusdk/ScreenMaker.sys" -ModName
"ScreenMaker"\ -AllTask -Hidden
```

13.4 Frequently asked questions

How to deploy manually to a Virtual Controller

If for any reason you wish to manually by-pass the Deploy button in RobotStudio and the virtual controller, the following information describes what files are to be moved.

Location of output files

The files that contain the FlexPendant application from ScreenMaker are found (for example) in the **bin** directory under the **My ScreenMaker Projects** located in the **My documents** directory of the user.

For example, **My Documents\My ScreenMaker Projects\SCM_Example\bin** where SCM_Example is the example ScreenMaker project.

The files in the **bin** directory are to be copied to a location where the Virtual FlexPendant can read them during the start of the FlexPendant.

Location where the Virtual FlexPendant reads the files

The recommended location for manually copying the ScreenMaker output files is the location of the virtual controller system.

If the system is created manually from **System Builder**, it is located in the **My Documents** directory.

For example, **My Documents\IRB4400_60_SCM_Example\HOME** where IRB4400_60_SCM_Example is the example controller system.

If the system is created by a Pack and Go and then restored, it is located in the **RobotStudio\Systems** folder.

For example,
MyDocuments\RobotStudio\Systems\IRB4400_60_SCM_Example\HOME where IRB4400_60_SCM_Example is the example controller system.

Copy files

Copy the files from the ScreenMaker output to the Home directory of the virtual controller system.

Restart the Virtual FlexPendant and the new application will be loaded.

Picture object and changing images due to I/O

The typical user objective is to have an image that changes when an I/O signal changes, this is common for a digital input to affect the state on the FlexPendant.

Actions

This is accomplished by adding an image and allowing the image to have multiple states.

Set AllowMultipleState to TRUE and set the Image state.

Create two states and add images for each state.

The Value property is extremely important. If binding to a digital input then there are two states for the input, 0 and 1. Set the Value property to the value of the bound variable 0 and 1 for digital input. It is also possible to bind to RAPID variables and have multiple states and values for the values in the RAPID variable.

Continues on next page

Set the SelectedStateValue property to bind to a controller object.

How to get radio buttons to show state when entering

The objective is to have two radio buttons that controls one digital output. When the screen is loaded, the buttons should show the current state of the output.

Actions

Create a group or a panel and place the two radio buttons on the group or panel.

For button1, set the property default value to **True** and bind the property to the value of the controller digital output signal.

For button2, do not do any changes.

When the screen is loaded, the state of the two radio buttons is established correctly.

What is RAPID array

A RAPID array is a variable that contains more than one value. An index is used to indicate one of the values.

Sample RAPID array

Consider the following RAPID code.

```
VAR string part{3} := ["Shaft", "Pipe", "Cylinder"];
```

Here, ‘part’ is a RAPID array which consists of three values. The index of the array in part ranges from 1 to 3.

The index of a RAPID array should not be negative and should start with 1.

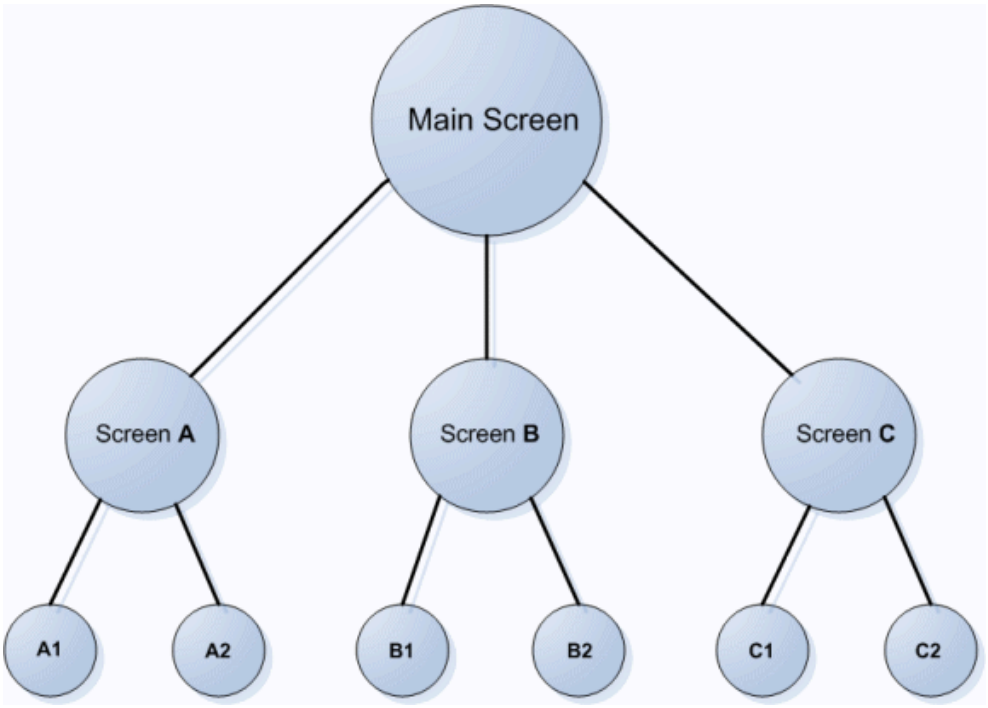
Screen navigation

Screen navigation in ScreenMaker follows a tree structure.

Consider the following example:

- To open screen **A1**, you first have to open **Screen A**
- To navigate from screen **A1** to screen **B1**, you first have to close screen **A1** and then **Screen A** and navigate from **Main Screen** through **Screen B** to screen **B1**.
- Similarly, to navigate from screen **B1** to screen **C1**, you first have to close screen **B1** and **Screen B** and then navigate from **Main Screen** through **Screen C** to screen **C1**.

Continues on next page



en0900000645

13.5 Tutorial

Overview

This chapter is designed as a tutorial to take you through the steps involved in designing a FlexArc Operator Panel.

The FlexArc Operator Panel is a simple arc welding cell, where the robots can perform the following three different jobs.

Job	Description
Produce	Welding the part
Service	Service the welding gun
Bull's Eye	Calibrate with bull's eye

The FlexArc Operator Panel displays the following graphic elements:

- Controller Status (controller mode auto or manual and the RAPID execution status)
- Part Status (number of produced parts, the average cycle time per part, and a Reset button)
- Robot jobs (Produce, Service, and Bull's Eye) and Robot locations (Robot at home position, service location, calibration location, and part location)
- Start and Stop buttons.

Designing the FlexArc operator panel

Use this procedure to design the FlexArc operator panel:

	Action	Information
1	Create a system for the FlexArc operator panel.	Select the following options: • FlexPendant Interface • PC Interface
2	Load **EIO.cfg** and **MainModule.mod** files.	Select the following options, By default: • For *Windows XP*, the files can be found at *C:\Documents and Settings\<user name>\My Documents\RobotStudio\My ScreenMaker Projects\Tutorial*

Continues on next page

<table>
<tr><th></th><th>Action</th><th>Information</th></tr>
<tr><td>3</td><td>The following signals are created after loading EIO.cfg file
<table>
<tr><th>IO</th><th>Type</th><th>Description</th><th>Connection</th></tr>
<tr><td>DI_RobotAtHome</td><td>DI</td><td>Indicates robot at home position</td><td>DI_RobotAtHome = DO_SIMHOME</td></tr>
<tr><td>DI_RobotAtBullseye</td><td>DI</td><td>Indicates robot at bull's eye position</td><td>DI_RobotAtBullseye = DO_SIMBULLS</td></tr>
<tr><td>DI_RobotAtService</td><td>DI</td><td>Indicates robot at service position</td><td>DI_RobotAtService = DO_SIMSERVICE</td></tr>
<tr><td>DI_PRODUCE</td><td>DI</td><td>Indicates robot is producing part</td><td>DI_PRODUCE = DO_PRODUCE</td></tr>
<tr><td>DO_SIMHOME</td><td>DO</td><td>Simulate robot at home</td><td></td></tr>
<tr><td>DO_SIMBULLS</td><td>DO</td><td>Simulate robot at bull's eye</td><td></td></tr>
<tr><td>DO_SIMSERVICE</td><td>DO</td><td>Simulate robot at service</td><td></td></tr>
<tr><td>DO_PRODUCE</td><td>DO</td><td>Simulate robot is producing part</td><td></td></tr>
<tr><td>GI_JOB</td><td>GI</td><td>The code of ordered job</td><td>GI_JOB = GO_JOB</td></tr>
<tr><td>GO_JOB</td><td>GO</td><td>Simulate job order</td><td></td></tr>
</table></td><td></td></tr>
<tr><td>4</td><td>Create an empty station in RobotStudio with the system created in the previous step.</td><td></td></tr>
<tr><td>5</td><td>Launch ScreenMaker from RobotStudio.</td><td></td></tr>
<tr><td>6</td><td>Create a new ScreenMaker project.</td><td>Select the following options:
1 Enter the project name as FlexArcGUI, and save it in the default location, C:\Users\<user name>\Documents\RobotStudio\My ScreenMaker Projects\Tutorial.
2 A new tab MainScreenis added to the Design Surface.</td></tr>
<tr><td>7</td><td>Configure the Project properties.</td><td>To customize how the GUI should appear on the FlexPendant, modify the Project properties.</td></tr>
<tr><td>8</td><td>Connect to the controller.</td><td>The result appears in the output window.</td></tr>
</table>

Continues on next page

<table>
<tr><th></th><th>Action</th><th>Information</th></tr>
<tr><td>9</td><td>Create application variables (temporary variables) and configure them with the following data:
<table>
<tr><th>Name</th><th>Type</th><th>Value</th></tr>
<tr><td>MyResetValue</td><td>Num</td><td>0</td></tr>
<tr><td>JobProduce</td><td>Num</td><td>1</td></tr>
<tr><td>JobIdle</td><td>Num</td><td>0</td></tr>
<tr><td>JobBulls</td><td>Num</td><td>2</td></tr>
<tr><td>JobService</td><td>Num</td><td>3</td></tr>
</table></td><td></td></tr>
<tr><td>10</td><td>Design the Main Screen.</td><td></td></tr>
<tr><td>11</td><td>Build and Deploy the project.</td><td></td></tr>
<tr><td>12</td><td>Open virtual FlexPendant and test the GUI.</td><td>• In RobotStudio, press Ctrl+F5 to launch the virtual FlexPendant.
• Click FlexArc operator panel to launch the GUI.
Note
Ensure that you switch the controller to Auto mode and start the RAPID execution.</td></tr>
</table>

Introduction to designing the screen

A major effort in the GUI project development is designing screens. The Form designer in the ScreenMaker allows you to drag controls from the toolbox to the design surface. Using the Properties window, you can resize, position, label, color, and configure the controls.

Designing FlexArc Operator Panel screen

Use this procedure to design the FlexArc Operator Panel screen:

1 Drag a GroupBox control from the General category; place it on the design surface and set the following values in the Properties window.

Property	Value
Location	14,45
Size	150,100
Title	Controller Status
BackColor	LightGray

2 Drag another GroupBox control from the General category; place it on the design surface and set the following values in the Properties window.

Property	Value
Location	14,170
Size	150,204

Continues on next page

Property	Value
Title	Part Status
BackColor	LightGray

3 Drag a ControllerModeStatus control from the Controller Data category; place it in the *Controller Status* groupbox created and set the following values in the Properties window:

Property	Value
Location	19,40
Size	44,44
BackColor	LightGray

4 Drag a RapidExecutionStatus control from the ControllerData category; place it in the *Controller Status* groupbox created and set the following values in the Properties window:

Property	Value
Location	80,40
Size	44,44
BackColor	LightGray

5 Drag a TpsLabel control from the General category; place it in the *Part Status* groupbox created and set the following values in the Properties window:

Property	Value
Location	16,30
Size	131,20
Text	Parts Produced
BackColor	LightGray
Font	TpsFont10

6 Drag a NumEditor control from the ControllerData category; place it in the *Parts Status* groupbox created and set the following values in the Properties window:

Property	Value
Location	16,56
Size	116,23
Value	Link to RAPID variable *partsReady* defined in the module *MainModule*.

7 Drag another TpsLabel control from the General category; place it in the *Part Status* groupbox created and set the following values in the Properties window:

Property	Value
Location	16,89

Continues on next page

Property	Value
Size	131,20
Text	Cycle time/part
BackColor	LightGray
Font	TpsFont10

8 Drag another NumEditor control from the General category; place it in the *Part Status* groupbox created and set the following values in the Properties window:

Property	Value
Location	16,115
Size	116,23
Value	Link to RAPID variable *cycleTime* defined in the module *MainModule*.

9 Drag a Button control from the General category; place it in the *Part Status* group box created and set the following values in the Properties window:

Property	Value
Location	33,154
Size	85,34
Text	Reset

Perform the following for the **Reset** button in the *Part Status* group:

Step	Action
1	Double-click the button **Reset**. The **Events Panel** dialog box appears which is used to define the actions for Events.
2	In the **Events Panel** dialog box, click **Add Action**; point to **Rapid Data** and select **Write a Rapid Data**. The **Action Parameters** dialog box appears; assign Rapid data to the following value and click **OK**. • **T_ROB1.MainModule.partsReady** to **MyResetValue.Value** Similarly, assign Rapid data to the following value and click **OK**. • **T_ROB1.MainModule.cycleTime** to **MyResetValue.Value** Two actions of similar type are needed to perform the **Reset** action. One is to reset Rapid variable **partsReady** to 0, the other is to reset Rapid variable **cycleTime** to 0.

10 Drag a PictureBox control from the General category; place it on the design surface and set the following values in the Properties window:

Property	Value
Location	177,28
Size	284,359
SizeMode	StretchImage
Image	FlexArcCell.GIF

Continues on next page

Note

You can find the graphic (.GIF) files at *C:\MyDocuments\RobotStudio\My ScreenMaker Projects\Tutorial\Images*.

11 Drag another PictureBox control from the General category; place it on the design surface and set the following values in the Properties window:

Property	Value
Location	237,31
Size	48,48
SizeMode	StretchImage
Image	RobotAtHome.GIF
AllowMultipleStates	True Select **Image** property from the **StatesEditor** dialog box.
SlectedStateValue	DI_RobotAtHome
States	Link State{0} to *RobotAtHome_gray.GIF* Link State{1} to *RobotAtHome.GIF*

Note

Add **AllowMultipleStates** option to the PictureBox control. The objective is to have an image that changes when an I/O signal changes.

For more information on how to use **AllowMultipleStates** for PictureBox control, see *Picture object and changing images due to I/O*.

12 Drag a Button control from the General category; place it on the design surface and set the following values in the Properties window:

Property	Value
Location	486,66
Size	116,105
Text	Start
Font	TpsFont20b
BackColor	LimeGreen
Enabled	Link to DI_RobotAtHome

Perform the following for the **Start** button:

Step	Action
1	Double-click the button **Start** or click the **Smart tag** and select *Define Actions when clicked*. The **Events Panel** dialog box appears which is used to define the actions for Events.
2	In the **Events Panel** dialog box, click **Add Action**; point to **Rapid Data** and select **Write a Rapid Data**. The **Action Parameters** dialog box appears.
3	In the **Action Parameters** dialog box, assign Rapid data to the following value and click **OK**. • **T_ROB1.MainModule.JobProduce** to **Job**.

Continues on next page

13 Drag a Button control from the General category; place it on the design surface and set the following values in the Properties window:

Property	Value
Location	486,226
Size	116,105
Text	Stop
Font	TpsFont20b
BackColor	LimeGreen
Enabled	Link to DI_PRODUCE

Perform the following for the **Stop** button:

Step	Action
1	Double-click the button **Stop** or click the **Smart tag** and select *Define Actions when clicked*. The **Events Panel** dialog box appears which is used to define the actions for Events.
2	In the **Events Panel** dialog box, click **Add Action**; point to **Rapid Data** and select **Write a Rapid Data**. The **Action Parameters** dialog box appears.
3	In the **Action Parameters** dialog box, assign Rapid data to the following value and click **OK**. • **T_ROB1.MainModule.JobIdle** to **JobIdle**

14 Drag a Button control from the General category; place it on the design surface and set the following values in the Properties window:

Property	Value
Location	274,246
Size	111,47
Text	Bull’s Eye
Font	TpsFont14b
Enabled	Link to DI_RobotAtHome
AllowMultipleStates	True Select **BackColor** property from the **StatesEditor** dialog box
SelectedStates	DI_RobotAtBull'sEye
States	Link State{0} to *Red* Link State{1} to *Green*

Perform the following for the **Bull’s Eye** button:

Step	Action
1	Double-click the button **Bull’s Eye** or click the **Smart tag** and select *Define Actions when clicked*. The **Events Panel** dialog box appears which is used to define the actions for Events.
2	In the **Events Panel** dialog box, click **Add Action**; point to **Rapid Data** and select **Write a Rapid Data**. The **Action Parameters** dialog box appears.
3	In the **Action Parameters** dialog box, assign Rapid data to the following value and click **OK**. • **T_ROB1.MainModule.JobBulls** to **JobBulls**

Continues on next page

15 Drag a Button control from the General category; place it on the design surface and set the following values in the Properties window:

Property	Value
Location	274,324
Size	111,47
Text	Service
Font	TpsFont14b
Enabled	Link to DI_RobotAtHome
AllowMultipleStates	True Select **BackColor** property from the **StatesEditor** dialog box
SelectedStates	DI_RobotAtService
States	Link State{0} to *Red* Link State{1} to *Green*

Perform the following for the **Service** button:

Step	Action
1	Double-click the button **Service** or click the **Smart tag** and select *Define Actions when clicked*. The **Events Panel** dialog box appears which is used to define the actions for Events.
2	In the **Events Panel** dialog box, click **Add Action**; point to **Rapid Data** and select **Write a Rapid Data**. The **Action Parameters** dialog box appears.
3	In the **Action Parameters** dialog box, assign Rapid data to the following value and click **OK**. • **T_ROB1.MainModule.JobService** to **JobService**

Building and deploying the project

1 From the ScreenMaker ribbon, click Build.

2 From the ScreenMaker ribbon, click Deploy.

3 In RobotStudio, press **Ctrl+F5** to launch the Virtual Flexpendant and click FlexArc Operator Panel to open the GUI.

Note

Ensure that you start the RAPID execution and switch the controller into Auto mode.

This page is intentionally left blank

14 RobotStudio®Cloud

Overview

Use this feature to:

- Edit, visualize and verify robot programs.
- Store and manage RobotStudio projects.
- Track and manage changes to projects using Version Control.
- Stay connected to your project repository and edit from anywhere.
- Share projects and collaborate with others.

Getting started

Activating the license key

RobotStudio Premium license includes the RobotStudio Cloud subscription. This must be activated from the **Activation Wizard**.

To activate RobotStudio Cloud subscription, user must have an account in the myABB Business Portal, *www.abb.com/myABB*.

Activating the RobotStudio Cloud subscription

1. Click the **File** tab, and then click the **Help** section.
2. Under **Support**, click **Manage Licenses**. The **Options** dialog appears with the **Licensing** options.
3. If you have not activated your RobotStudio license, click the **Activation Wizard** and follow the instructions.
4. If your RobotStudio license includes the RobotStudio Cloud subscription, a **Sign In** button will be visible in RobotStudio in the last step of the **Activation Wizard** and on the **Options:General:Licensing** page.
5. Click the **Sign In** button, and enter your credentials in the **Sign in to your account** dialog.

   If the signing in is successful, the RobotStudio *Subscription key* and your *username(email address)* gets displayed.
6. Click the **Activate** button.

   If the activation is successful, you can see the **Activated** icon and the **View My Cloud Projects** link. Click this link to open RobotStudio Cloud.

Opening RobotStudio Cloud web page

1. In RobotStudio, on the top right corner, above the ribbon, click on your user name, then click **View My Cloud Projects** to open RobotStudio Cloud in the web browser.
2. You'll be asked for credentials before signing in to the site. Type in your credentials and press ENTER.

Creating a Cloud project

1. Start RobotStudio.
2. Click the **File** tab to open the **Backstage** view.

Continues on next page

Continued

3 On the left navigation pane, click **New**.
4 Under **Project**, click **Project**.
5 In the **Project** pane, in the **Name** box, type the name for the new project.
6 Under **Location**, select **RobotStudio Cloud**.
7 Click **Create**. A new project gets created in RobotStudio Cloud.

> **Note**
>
> To start from scratch, create an empty project and customize. Or, for practice using RobotStudio features try including Robot and Virtual Controllers with robot model and RobotWare from the options provided.

Opening a Cloud project in RobotStudio

1 In the **Backstage** view, click the project tile, to view the project details on the right navigation pane.
2 Under **Branches**, click the **Open** button to open the project in RobotStudio.

   By default the Main opens, to open a specific branch, select the branch and click the **Open** button.

Opening a Cloud project in a web browser

1 In the **Backstage** view, click the project tile, to view the project details on the right navigation pane.
2 On the right navigation pane, click the **View Online** link, to open the RobotStudio Cloud web page in the web browser.

   You'll be asked for credentials every time before signing in to the site. Type in your credentials and press ENTER to view the project in RobotStudio Cloud.

Uploading an existing project

1 Open an existing project in RobotStudio.
2 Click the **File** tab to open the **Backstage** view.
3 On the left navigation pane, click **Share** to see the available options.
4 Click **Upload to RobotStudio Cloud**. The Upload to RobotStudio Cloud pane opens.
5 Type-in the Project Name and commit message in the fields provided, click **Upload**.

   A copy of the project gets created and uploaded to RobotStudio Cloud.

Copying a project

1 On the File menu, click **Save Project As** to open the **Save MyProject As** dialog.
2 Select the check boxes **RobotStudio Cloud** or **This PC** to save the copy of the project to the respective location.

Continues on next page

3 If you choose to save the copy of the project locally, browse and select the required folder and click **OK**.

The entire Project structure including Virtual Controllers, library components and user files inside the Project folder will be copied to the new location. The user can select to open the copy or keep the current Project open.

Version Control

RobotStudio's version control functionality allows you to manage and keep track of changes made to your RobotStudio project. It also enables you to work on parallel versions of a RobotStudio project by creating branches.

Accessing the RobotStudio Cloud portal requires a working Internet connection and valid credentials.

What is a branch?

A branch is a copy of the RobotStudio project where you can explore changes without affecting the original files. It allows you to work in parallel and independently from other branches. Branches are useful to:

- Share and collaborate on projects with colleagues and stakeholders.
- Isolate experiments and test different versions of a robot program or cell layout.

Any new project in RobotStudio has a Main branch. This is the primary or default branch in RobotStudio.

Create a branch

1 In RobotStudio, click the **File** tab to open the **Backstage** view.
2 On the left navigation pane, click **Open** and then click **RobotStudio Cloud** to see all available cloud projects.
3 Click the project tile, to view the project details on the right navigation pane.
4 On the right navigation pane, under **My Branches**, click **New**.
5 Type the name of the branch in the **Create New Branch** dialog.

If you click **Create** , a new branch gets created, you can see the name of the branch under **My Branches**. Select this branch and click **Open** to view the project under this branch.

If you click **Create & Open**, the project opens under the newly created branch in RobotStudio.

Committing changes

All development updates can be saved using the **Commit** button. Commit has two options, **Commit** and **Review and Commit**.

- To commit the latest updates, on the top right corner, above the ribbon, click **Commit**. The **Commit Changes** dialog opens, type-in the required descriptions and click the **Commit** button to save the changes.

  Alternatively, **Commit** can also be accessed from the **File** menu, On the **File** menu, click the **Commit** option.

Continues on next page

Continued

- If a branch has uncommitted changes from the last edit or when the cloud version of the branch has been edited, use the **Review and Commit** option. On the top right corner, above the ribbon, click **Commit** and then click **Review and Commit**. The **Review Changes** window displays the module, with the latest changes and the last committed file side by side. Review the updates and click the **Commit** button.

> **Note**
>
> The **Save Project** option saves the project on the local disk, while **Commit** stores a version-controlled commit in the Cloud.

Merging changes from a branch to main

The branch must be updated with the latest changes in the Main before starting the merge operation.

1. In the **Backstage** view, click the project tile, to view the project details on the right navigation pane.
2. On the right navigation pane, under **Branches**, click **Merge**.

   Alternatively, the **Merge** button can also be accessed as follows:

   On the left navigation pane, click **Info**, the **Project Info** pane opens, under **Branch Information** click **Merge**.

   The **Merge** button will only be active if there are changes in the branch.
3. The **Merge** dialog opens in a web browser, click the **Merge** button. On successful merge, the branch gets deleted and main will be updated to reflect the changes in the branch.

Getting updates to a branch from main

1. Hover the mouse over the project tile and click **View Project** to open the **Version Control** page.
2. On the **Main** navigation pane, under **My Branches**, click the branch that must be updated.

   If there are changes in the **Main** branch, the **Update** button will be enabled.
3. Click the **Update** button, the **Update from Main** dialog opens in a web browser detailing the changes that will be merged into the branch from the **Main** branch.
4. Click the **Update** button to accept the changes in the **Main** to the branch.

   If the branch has committed updates, the *getting updates to a branch from main* operation can trigger conflicts.

Resolving conflicts

All conflicting changes must be resolved before the branch is updated.

1. If there are conflicting changes in the branch, when you click the **Update** button, the **Resolve Conflicts** window opens.

   This dialog shows the changes in both main and branch.

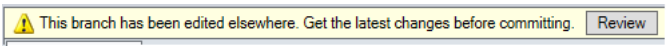
Continues on next page

2 Select the check boxes next to the highlighted changes, your selections will be available in the **Output** box.

Review and customize the contents in the **Output** box and click **Update**.

Getting the latest changes to the local copy of the branch

When the current branch in RobotStudio has new changes in RobotStudio Cloud, user gets the following notification.



xx2200001839

These changes can be reviewed and subsequently updated to the local copy.

1 Click the **Review** button in the notification. The **Get Changes** dialog opens.
2 Click the **Get Changes** button to get the changes to the local copy.

Making changes to the local copy of the branch when the branch has been changed in the RobotStudio Cloud can trigger conflicts, which must be resolved before getting the latest changes.

1 Click the **Review** button in the notification. The **Get Changes** dialog opens

When there are conflicting changes, the **Get Changes** dialog displays the **Resolve** button.

2 Click the **Resolve** button, the **Resolve Conflicts** window opens.

This window shows the changes in both **Latest version from RobotStudio Cloud** and **Your uncommitted changes**.

3 Select the check boxes next to the highlighted changes, your selections will be available in the **Output** box. You can choose to edit manually to resolve conflicts.

The **Apply** button will not be enabled till all conflicts are resolved.

4 Click **Apply** and then **Get Changes** to get the changes to the local copy of the branch.

Delete a branch

1 In RobotStudio, click the **File** tab to open the **Backstage** view.
2 On the left navigation pane, click **Open** and then click **RobotStudio Cloud** to see all available cloud projects.
3 Click the project tile, to view the project details on the right navigation pane.
4 On the right navigation pane, under **My Branches**, select the branch and click **Delete**.

Creating and exporting project viewer

Use the following steps to create and save the project viewer as a file to RobotStudio Cloud. The viewer can be shared by generating a link.

1
- To create a project viewer without simulation.

  On the **File** tab, click **Share** and then click **Export Viewer** .
- To create a project viewer with simulation.

Continues on next page

Continued

On the **Simulation** tab, in the Simulation Control group, click **Export Viewer**.

2 The **Export Viewer** dialog opens. In the **Location** group, select **RobotStudio Cloud**.

The **RobotStudio Cloud** option is only available for a Cloud project where the user has signed in.

3 Select **Create link to share file** for generating a link for sharing the project viewer file with other Cloud users.

Type-in description in the **Additional description** to describe project.

If **Create link to share file** is selected, the resulting link can be copied to the clipboard or sent in an e-mail.

4 Click **Create**. This project viewer will be created in the **.glb* (gITF file) format and uploaded to RobotStudio Cloud.

Simulation control buttons are enabled when the Project Viewer contains a recorded simulation. When recording a simulation the user can confirm that the simulation was successful before the viewer is created .

Project viewers and other attachments can be managed in the **Attachments** view in the RobotStudio Cloud web application.

15 Home tab

15.1 Virtual controller

About this button

With the **Virtual Controller** button, you can either create a *virtual controller* from layout or template, choose an existing virtual controller, or select a virtual controller from a robot gallery.

Creating a virtual controller from layout

To create virtual controller using layout follow below procedure:

1. Click **From Layout** to bring up the first page of the wizard.
2. In the **Name** box, enter the name of the virtual controller. The location of the virtual controller will be displayed in the **Location** box.
3. In the **RobotWare** drop down, select the version of *RobotWare* you want to use.
4. Click **Next**.
5. In the **Mechanisms** box, select the *mechanisms* that you want to include in the virtual controller.
6. Click **Next**.

   If only one mechanism was selected in the previous page, this page will not be shown.

   Tasks can be added and removed using the respective buttons; mechanisms can be moved up or down using the respective arrows.
7. Optionally, make any edits in the mapping, and then click **Next**.

   The **Controller Option** page appears.
8. On the **Controller Option** page, you have the option to align *Task Frame*(s) with the corresponding *Base Frame*(s).
   - For single robot system, select the check-box to align task frame with base frame.
   - For *MultiMove* Independent system, select the check box to align task frame with base frame for each robot.
   - For MultiMove Coordinated system, select the robot from the drop down list and select the check box to align task frame with base frame for the selected robot.
9. Verify the summary and then click **Finish**.

   If the system contains more than one robot, the number of tasks and the base frame *positions* of the mechanism should be verified in the **Motion Configuration** window.

> **Note**
>
> To create a system from layout, all your mechanisms such as robots, track motions and positioners, must be saved as libraries.

Continues on next page

Creating a New Virtual Controller

1. Click **New Controller** to bring up a dialog box.
2. Under the **Controller** group, enter the name of the controller in the **Name** box.
3. Select **Create new** and then select the required robot model from the **Robot Model** list for downloading to create a controller. If the selected model is not downloaded already, then user will be notified to download.
4. Based on the robot model, select the **RobotWare** version and **Variant**.

   While selecting Robot model that uses **RobotWare** version 7, **Controller** also must be selected.
5. To create from backup, select **Create from backup** and then browse to select the required backup file. You can also select the RobotWare version followed by the RobotWare Add-in version. Select the **Restore backup** check box to restore backup on the new controller.
6. In the **Mechanisms** group, select whether to **Import from library** or to **Use existing station mechanisms**.

Adding an existing virtual controller

1. Click **Existing Controller** to bring up a dialog box.
2. In the **Location** list, select a folder.
3. In the **Virtual Controller** list, select a controller.

   Optionally, to modify or create a new the virtual controller, click on **Manage** *link*, select **Modify Installation** from the list and then select the controller.
4. In the **Options** group, select whether to import libraries or to use the existing station libraries.

   User can tick the **Reset controller (I-start)** check-box to reset the virtual controller before adding it to the station.
5. Click **OK**.

15.2 Target

Overview

You can create multiple targets at once, either enter the coordinates manually or click in the Graphics window to pick the desired positions.

You can align the target *orientations* with the surface of an adjacent CAD *part* by selecting the check box for aligning the surface to the closest part.

Type-in the target name prefix and select the task and workobject where the targets should go, and optionally select the *path* where the corresponding move instruction must be created. The selected instruction template will be used for the created move instruction.

Creating a target

1. In the **Layout** browser, select the workobject in which you want to create the target and set it as **Active**.
2. Click **Create Target** to bring up a dialog box.
3. Select the **Reference** *coordinate system* you want to use to position the target:

If you want to position the target	Select
absolute in the world coordinate system of the station	**World**
relative to the position of the active workobject	**workobject**
in a user-defined coordinate system	**UCS**

4. In the **Points** box, click **Add New** and then click the desired position in the graphics window to set the position of the target. You can also enter the values in the **Coordinates boxes** and click **Add**.
5. Enter the **Orientation** for the target. A preliminary cross will be shown in the graphics window at the selected position. Adjust the position, if necessary. To create the target, click **Create**.
6. If you want to change the workobject for which the target is to be created, expand the **Create Target** dialog box by clicking the **More** button. In the **WorkObject** list, select the workobject in which you want to create the target.
7. If you want to change the target name from the default name, expand the **Create Target** dialog box by clicking the **More** button and entering the new name in the **Target name** box.
8. Click **Create**. The target will appear in the browser and in the graphics window.

Continues on next page

Note

The created target will not get any configuration for the robot axes. To add the configuration values to the target, use either **Auto Configuration** or the **Configurations** dialog box.

If using external axes, the position of all activated external axes will be stored in the target.

Creating a jointtarget

Joint target defines each individual axis position, for both the robot and the external axes.

1. Click **Create Jointtarget** to bring up a dialog box.
2. If you want to change the default name of the jointtarget, enter the new name in the **Name** box.
3. In the **Axes Values** group, do as follows:
   - For the **Robot axes**, click the **Values** box and then click the down arrow. The **Joint Values** dialog box will be displayed. Enter the joint values in the boxes and click **Accept**.
   - For the **External axes**, click the **Values** box and then click the down arrow. The **Joint Values** dialog box will be displayed. Enter the joint values in the boxes and click **Accept**.
4. Click **Create**. The jointtarget will appear in the browser and in the graphics window.

Note

JointTargets for external axis are not visualized in the graphical window.

The Create Jointtarget dialog box

Name	Specify the name of the jointtarget.
Robot axes	Click the **Values** list, enter the values in the **Joint values** dialog box and click **Accept**.
External axes	Click the **Values** list, enter the values in the **Joint values** dialog box and click **Accept**.
Storage Type	Select the **Storage TypeTASK PERS** if you intend to use the jointtarget in multimove mode. The storage type of a data object determines how it is stored in memory and how it can be used.The available types are **CONST,VAR,PERS,TASK PERS**.
Module name	Select the module in which you want to declare the jointtarget.

Overview

Targets on Edge creates targets and move instructions along the edges of the geometric surface by selecting target points in the graphics window. Each point on a geometric edge has certain properties that can be used to position robot targets relative to the edge.

Continues on next page

Creating targets on edge

1 On the **Home** tab, click **Target** and select **Create Targets on Edge**.

The **Targets on Edge** dialog box appears.

> **Note**
>
> The selection mode in graphics window is automatically set to **Surface**, and the snap mode is set to **Edge**.

2 Click on the surface of the *body* or part to create target points.

The closest point on the adjacent edge is calculated and added to the list box on as target points Point 1, Point 2 ....

> **Note**
>
> When an edge is shared between two surfaces, the normal and tangent directions depend on the surface selected.

3 Use the following variables to specify how a target is related to a point on the edge.

Select...	to...
Vertical offset	specify the distance from the edge to the target in the surface normal direction.
Lateral offset	specify the distance from the edge to the target perpendicular to the edge tangent.
Approach angle	specify the angle between the (inverse) surface normal and the approach vector of the target.
Reverse travel direction	specify if the travel vector of the target is parallel or inversely parallel to the edge tangent.

> **Note**
>
> For each target point, a preview of the approach and travel vectors are displayed as arrows and as a sphere representing the point on the edge in the graphics window. The preview of the arrows are updated dynamically once the variables are modified.

4 Click **Remove** to remove the target points from the list box.

5 Click **More** to expand the **Create Targets on Edge** dialog box and to choose the following advanced options:

Use...	to..
Target name	change the target name from the default name to a new user defined name
Task	select the task for which to add targets. By deafult, active task in the station is selected.
Workobject	select the workobject for which you want to create the targets on edge.

Continues on next page

Use...	to..
Insert Move Instructions in	create Move instructions in addition to targets, which will be added to the selected path procedure. The active process definition and process template will be used.

6 Click **Create**.

The target points and Move instructions (if any) are created and are displayed in the Output window and graphics window.

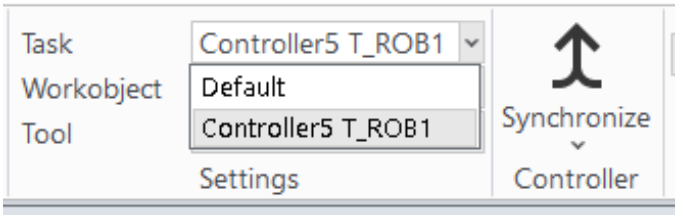
15.3 Path

Empty path

Use the below procedure to create an empty path:

1. On the **Home** tab, click **Path**.
2. Click **Empty Path**.
3. In the **Paths&Targets** browser, the new empty path is created in the folder of the active task.

> **Note**
>
> The active task is as per the **Task** selected in the combobox.



xx1900001906

AutoPath

Use the AutoPath feature to generate accurate paths (linear and circular) based on CAD *geometry*. You need to have a geometric object with edges, *curves*, or both.

AutoPath feature can create paths from curves or along the edges of a surface. To create a path along a surface use selection level Surface, and to create a path along a curve, use selection level Curve. When using Selection level Surface, the closest edge of the selection will be picked for inclusion in the path. An edge can only be selected if connected to the last selected edge.

When using Selection level Curve, the selected edge will be added to the list. If the curve does not have any branches, all edges of the entire curve will be added to the list if holding the SHIFT button when selecting an edge. The Approach and Travel directions as defined in the RobotStudio options are used to define the orientation of the created targets.

Use this procedure to automatically generate a path.

1. In the **Home** tab, click **Path** and select **AutoPath**.

   The AutoPath tool appears. Select the **Create multiple paths from curves** check-box to create multiple paths from the selected curves.
2. Select the edge or curve of the geometric object for which you want to create a path.

Continues on next page

The selection is listed as edges in the tool window.

Note

- If in a geometric object, you select curve (instead of an edge), then all the points that result in the selected curve are added as edges to the list in the graphic window.
- Ensure you always select continuous edges.

3 Click **Remove** to delete the recently added edge from the graphic window.

Note

To change the order of the selected edges, select **Reverse** check box.

4 You can set the following **Approximation Parameters**:

Select or enter values in	to
MinDist	Set the minimum distance between the generated points. That is, points closer than the minimum distance are filtered.
Tolerance	Set the maximum deviation from the geometric description allowed for the generated points.
MaxRadius	Determines how large a circle radius has to be before considering the circumference as a line. That is, a line can be considered as a circle with infinite radius.
Linear	Generate a linear move instruction for each target.
Circular	Generate circular move instructions where the selected edges describe circular segments.
Constant	Generate points with a Constant distance.
End Offset	Set the specified offset away from the last target.
Start Offset	Sets the specified offset away from the first target.

The **Reference Surface** box shows the side of the object that is taken as normal for creating the path.

Click **More** to set the following parameters:

Select or enter values in	to
Approach	Generate a new target at a specified distance from the first target.
Depart	Generates a new target at a specified distance from the last target.

5 Click **Create** to automatically generate a new path.

A new path is created and move instructions are inserted for the generated targets as set in the Approximation parameters.

Note

The targets are created in the active workobject.

Continues on next page

6 Click **Close**.

15.4 Other

Create Workobject

Overview

A workobject is a local coordinate system that indicates the reference position (and orientation) of a work piece. The workobject contains two *frames*, the user frame and the object frame. The user frame (user coordinate system) is one of the two frames of a workobject, is defined relative to the controller world coordinate system. The object frame is defined relative to the user frame. To create a workobject, the following parameters must be specified in the **Create Workobject** dialog.

Name	Specify the name of the workobject.
Robot holds workobject	Select whether the workobject is to be held by the robot. If you select **True**, the robot will hold the workobject. The tool can then either be stationary or held by another robot.
Moved by mechanical unit	Select the *mechanical unit* that moves the workobject. This option is applicable only if **Programmed** is set to **False**.
Programmed	Select **True** if the workobject is to use a fixed coordinate system, and **False** if a movable (that is, external axes) will be used.
Position x, y, z	Click in one of these boxes, and then click the position in the graphics window to transfer the values to the **Position** boxes.
Rotation rx, ry, rz	Specify the rotation of the workobject in the UCS.
Frame by points	Specify the frame position of the user frame.
Position x, y, z	Click in one of these boxes, and then click the position in the graphics window to transfer the values to the **Position** boxes.
Rotation rx, ry, rz	Specify the rotation of the workobject.
Frame by points	Specify the frame position of the object frame.
Storage type	Select **PERS** or **TASK PERS**. Select the **Storage TypeTASK PERS** if you intend to use the workobject in multimove mode.
Task	Specify the task of the sync properties.
Module	Select the module in which to declare the workobject.

Creating a workobject

1 On the **Home** tab, in the **Path Programming** group, click **Other** and select **Create Workobject**.

  The **Create Workobject** dialog box appears.

2 In the **Misc Data** group, enter the values for the new workobject.

3 In the **User Frame** group, do one of the following:

- Set the position of the user frame by entering values for the **Position x,y,z** and the **Rotation rx, ry, rz** for the workobject by clicking in the **Values** box.
- Select the user frame by using the **Frame by points** dialog box.

Continues on next page

4 In the **Object Frame** group you can reposition the object frame relative to the user frame by doing any of the following:
   - Set the position of the object frame by selecting values for **Position x, y, z** by clicking in the **Values** box.
   - For the **Rotation rx, ry, rz**, select **RPY (Euler ZYX)** or **Quaternion**, and enter the rotation values in the **Values** dialog box.
   - Select the object frame by using the **Frame by points** dialog box.
5 In the **Sync Properties** group, enter the values for the new workobject.
6 Click **Create**. The workobject will be created and displayed under the **Targets** node under the robot node in the **Paths&Targets** browser.

Creating tooldata

1 In the **Layout** browser, make sure the robot in which to create the tooldata is set as the active task.
2 On the **Home** tab, in the **Path Programming** group, click **Other**, and then click **Create Tooldata**.

   The *Create Tooldata* dialog box opens.
3 In the **Misc Data** group:
   - Enter the **Name** of the tool.
   - Select whether the tool is to be held by the robot in the **Robot holds tool** list.
4 In the **Tool Frame** group:
   - Define the **Position x, y, z** of the tool.
   - Enter the **Rotation rx, ry, rz** of the tool.
5 In the **Load Data** group:
   - Enter the **Weight** of the tool.
   - Enter the **Center of gravity** of the tool.
   - Enter the **Inertia** of the tool.
6 In the **Sync Properties** group:
   - In the **Storage type** list, select **PERS** or **TASK PERS**. Select **TASK PERS** if you intend to use the tooldata in MultiMove mode.
   - In the **Task** list, select the robot in which the task to be done using the tooldata.
   - In the **Module** list, select the module in which to declare the tooldata.
7 Click **Create**. The tooldata appears as a coordinate system in the graphics window.

Continues on next page

Create Action Instruction

Overview

An action instruction is an instruction other than a move instruction that can, for example, set parameters, or activate or deactivate equipment and functions. The action instructions available in RobotStudio are limited to those commonly used for controlling the robot's motions. For inserting other action instructions or another kind of RAPID code in the program, use the RAPID Editor.

The following table lists the action instructions that are available by default. It is possible to add the corresponding instruction template using the instruction template tool.

Action instruction	Description
ConfL On/Off	*ConfL* specifies whether to monitor the robot's configurations during linear movements. When *ConfL* is set to Off, the robot may use another configuration than the programed one for reaching the target during program execution.
ConfJ On/Off	*ConfJ* specifies whether to monitor the robot's configurations during joint movements. When *ConfJ* is set to Off, the robot may use another configuration than the programed one for reaching the target during program execution.
Actunit *UnitName*	*Actunit* activates the mechanical unit specified by *UnitName*.
DeactUnit *UnitName*	*Deactunit* deactivates the mechanical unit specified by *UnitName*.

Creating an action instruction

1 In the **Paths&Targets** browser, select where to insert the action instruction.

   To insert the action instruction at the beginning of a path then select the path.

   To insert the action instruction after another instruction then select the proceeding instruction.

2 On the **Home** tab, in the **Path Programming** group, click **Other**, and then click **Create Action Instruction**.

   The **Create Action Instruction** dialog box appears.

3 In the **Task** list, select the robot to which action instruction to be done.

4 In the **Path** list, select the path.

5 From the *Instruction Templates* list, select the action instruction to create.

6 Optionally, modify instruction arguments in the **Instruction Arguments** grid.

7 Click **Create**.

15.5 Virtual Reality

Introduction

The Virtual Reality (VR) button gets enabled on the **Home** tab when you connect a VR headset to the PC. RobotStudio and its Export Viewer works with HTC VIVE, HTC VIVE Cosmos, Oculus Rift, Meta/Oculus Quest 2, Oculus Rift S, Valve Index and Samsung HMD Odyssey (Windows Mixed Reality). These devices have motion tracking sensors for location and orientation tracking of the headset and for the hand controls. The hand controls can be used to interact with the virtual reality environment.

Note

Before activating Virtual Reality in RobotStudio, it is recommended to test the VR headset with the samples from the supplier to ensure its functionality.

Prerequisites

- A high-performance gaming PC that matches the specifications of the VR headset requirements. Check the supplier web page for installation details.
- Microsoft Windows 10 operating system.
- It is recommended to ensure that the physical space is obstacle-free. However, even in a limited space, the teleport function can be used to move around in the VR environment.

Programming

The VR function enables robots to be lead through programmed in a safe way which is otherwise not possible. You can teach movements to the robot by simply moving the robot around.

Navigation modes in the VR environment

The main navigation modes in the VR environment are:

- Physically moving around
- Snap move/turn
- Teleportation
- Drag navigation

Physically moving around

Take short steps or rotate yourself in the real world to make small adjustments to your viewpoint.

Snap move/turn

Press the left hand controller joystick/trackpad forward or backward to snap move in that direction in short steps. Press the joystick/trackpad left or right to snap turn in that direction.

Continues on next page

Teleportation

To achieve teleportation, press the **Grip** button on the left hand controller. Pressing the **Grip** button displays a beam, press the **Trigger** button to teleport to the yellow spot where the beam hits the floor. To cancel the teleportation, just release the left **Grip** button.

To move the teleportation target to a higher or lower level, with the left **Grip** button pressed, press the left joystick/trackpad forward or backward.

Drag navigation

To move and rotate the VR environment around a point that is centered between the left and right hand controllers, press the left and right **Grip** buttons simultaneously. Use the same buttons to move up/down, forward/backward, right/left and rotate in the VR environment.

16 Modeling tab

16.1 Import Geometry

Overview

Use Import geometry to add 3D parts to the station. The User Geometry gallery will display 3D geometry files in the Geometry folder of the user document location. If you have a project, the Project Geometry gallery will display 3D geometry files in the User Files folder of the current project. Additional galleries can be specified with the Locations option. To import general files, use the Browse for Geometry option. This dialog provides the following additional options.

- Convert CAD geometry to single part: Converts an assembly containing multiple parts to a single part.
- Surface model(render both sides of surfaces): Render both sides of surfaces.
- Import hidden/no-show entities: Entities specified as hidden/no show in the geometry file will be imported and made visible.
- Link to Geometry(remember source location): The file path of the imported geometry file will be remembered to allow geometry to be updated at a later occasion. Right click a part in Layout browser and click Update Linked geometry.
- Duplicate instanced geometry: If the geometry file contains entities which have multiple instances, each instance will be imported as separate entities.

Mathematical versus graphical geometries

A geometry in a CAD file always has an underlying mathematical representation. Its graphical representation, displayed in the graphics window, is generated from the mathematical representation when the geometry is imported to RobotStudio, after which the geometry is referred to as a part.

For this kind of geometry, you can set the detail level of the graphical representation, thus reducing the file size and rendering time for large models and improving the visual display for small models you might want to zoom in on. The detail level only affects the visual display; paths and curves created from the model will be accurate both with coarse and fine settings.

A part can also be imported from a file that simply defines its graphical representation; in this case, there is no underlying mathematical representation. Some of the functions in RobotStudio, such as snap mode and creation of curves from the geometry, will not work with this kind of part.

16.2 Create Mechanism

Overview

It is possible to create *mechanisms* to simulate external objects such as, turn tables, grippers, *positioners*. The following list describes some of the mechanisms:

- Robot is a mechanism that has a TCP and can be controlled by a *virtual controller*.
- Device is a mechanism which does not have a TCP.
- External axes is mechanism that does not have a TCP but can be controlled by a virtual controller, like track motion or a workpiece positioner or a gantry. An external axis can move a robot, for example, track motion.
- Tool is a mechanism that has TCP which can be moved by a robot.

> **Note**
>
> A mechanism must include two links.

Create a new mechanism

1. Click **Create Mechanism.**

   The Create Mechanism window is opened.
2. In the **Mechanism Model Name** box, enter a mechanism name.
3. From the **Mechanism Type** list, select a mechanism type.
4. In the tree structure, right-click **Links**, and then click **Add Link** to bring up the **Create Link** dialog box.

   A suggested name appears in the **Link Name** box.
5. In the **Selected Component** list, select a component (which will be highlighted in the graphics window) and click the arrow button to add the component to the components list box.

   The **Selected Component** list then automatically selects the next component, if any more are available. Add these, as required.

   > **Note**
   >
   > Components that are component of a library or mechanism cannot be selected.
6. Select a component in the components list box, enter any values in the **Selected Components** group boxes, and then click **Apply to component**.

   Repeat for each component, as required.
7. Click **OK**.
8. In the tree structure, right-click **Joints**, and then click **Add Joint** to bring up the **Create Joint** dialog box.

   A suggested name appears in the **Joint Name** box.

Continues on next page

9 Complete the **Create Joint** dialog box, and then click **OK**.

> **Note**
>
> The joint axes must be mapped to complete the joint values.

10 In the tree structure, right-click **Frame/Tool Data**, and then click **Add Frame/Tool** to bring up the **Create Frame/Tool** dialog box.

A suggested name appears in the **Frame/Tool Data name** box.

11 Complete the **Create Frame/Tool** dialog box, and then click **OK**.

The validity criteria for the **Frame/Tool** node are as follows:

12 In the tree structure, right-click **Calibration**, and then click **Add Calibration** to bring up the **Create Calibration** dialog box.

13 Complete the **Create Calibration** dialog box, and then click **OK**.

14 In the tree structure, right-click **Dependency**, and then click **Add Dependency** to bring up the **Create Dependency** dialog box.

15 Complete the **Create Dependency** dialog box, and then click **OK**.

16 If all nodes are valid, compile the mechanism.

Compiling a mechanism

When compiling, a new mechanism, created in the create mode of the Mechanism Modeler, is added to the station with the default name "Mechanism_" followed by an index number.

When compiling, an existing editable mechanism, modified in the modify mode of the Mechanism Modeler, is saved without any poses, joint mapping ot transition times.

To compile a mechanism, follow these steps:

1 To compile a new or edited mechanism, click **Compile Mechanism**.

The mechanism is inserted into the active station. The link parts are cloned with new names, but the corresponding links will update their part references. When the Mechanism modeler is closed, these cloned parts will be removed.

2 The Mechanism Modeler now switches to modify mode. To complete the mechanism, see below.

Completing or modifying a mechanism

To complete the modeling of a mechanism, follow these steps:

1 If the values in the **Joint Mapping** group are correct, click **Set**.

2 Configure the Poses grid. To add a pose, click **Add** and then complete the **Create Pose** dialog box. Click **Apply**, followed by **OK**.

- To add a pose, click **Add** and then complete the **Create Pose** dialog box. Click **Apply**, followed by **OK**.
- To edit a pose, select it in the grid, click **Edit**, and then complete the **Modify Pose** dialog box. Click **OK**.
- To remove a pose, select it in the grid and then click **Remove**.

Continues on next page

3 Click **Edit Transition Times** to edit transition times.

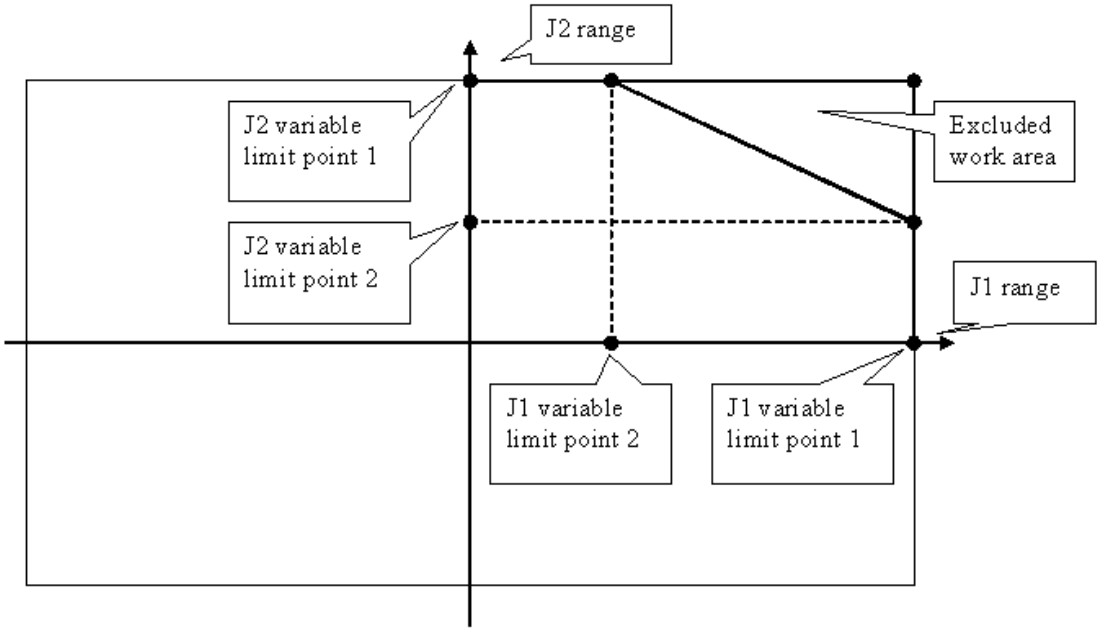
4 Click **Close**.

The Create Mechanism dialog box

Mechanism Model Name	Specifies the model name of the mechanism.
Mechanism Type	Specifies the mechanism type.
Tree structure	The components of the mechanism in a tree structure. The tree structure will not be visible unless the mechanism is editable. Each node (link, joint, frame, calibration and dependency) can be edited in its own dialog box, see below.
Compile Mechanism	Click this button to compile the mechanism. This button will not be visible unless the mechanism is editable and the mechanism model name is valid.

Using variable joint limits for interdependent joints

The Variable limit type can be assigned to joint limits as way of delimiting the range of motion for interdependent joints (area of movement as depicted on a graph).

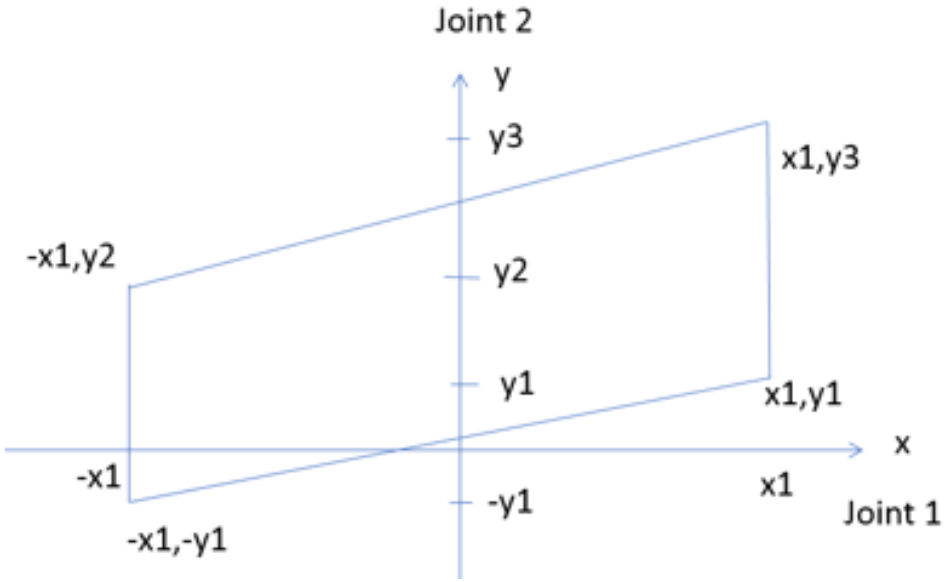


xx060012

Consider two joints Joint 1 and 2 as depicted below. Joint 2 can move between y1 and y4 when Joint 1 is at x1 degrees, and between y3 to y2 when Joint 1 is at x2

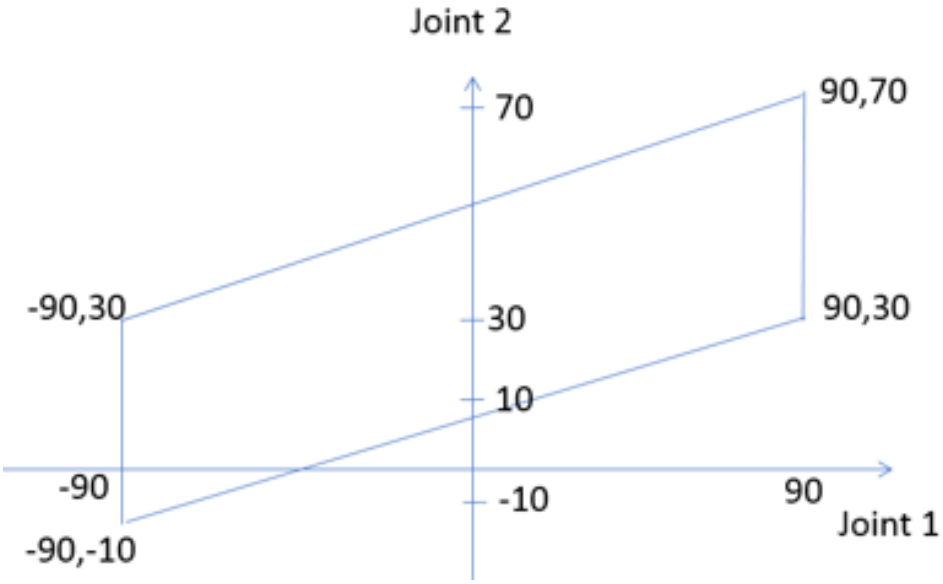
Continues on next page

degrees. That is, an area is formed by 4 points for each joint pair (Joint1 value: Joint2 value): x1: y1, x2: y2, x3: y3 and x4: y4



xx2100000521

Use the following example to visualize variable joint limits. For example, Joint 2 has a range of motion between -10 to +30 when Joint 1 is at -90 degrees and between +10 to +70 when Joint 1 is at +90 degrees



xx2100000522

An area will be formed by 4 points for each joint pair (Joint1:Joint2): -90:30, 90:70, 90:10 and -90, -10. In the mechanism modeler, add the following values for the variable limits in the following order J1: -90, 90, 90, -90 and J2: 30, 70, 10, -10.

Continues on next page

The Modify Mechanism dialog box

The **Modify Mechanism** dialog box contains the objects found in the **Create mechanism** dialog box, as well as the following:

Joint Mapping	These boxes handle the joint mapping of the mechanism. When editing, the mechanism must be disconnected from its library. The values must be integers from 1 – 6 in ascending order.
Set	Click this button to set the joint mapping.
Poses	Displays the poses and their joint values. Selecting a pose will move the mechanism to it in the graphics window.
Add	Click this button to bring up the **Create Pose** dialog box for adding a pose.
Edit	Click this button to bring up the **Modify Pose** dialog box for editing a selected pose. A SyncPose cannot be edited unless the mechanism is disconnected from its library.
Remove	Click this button to remove the selected pose. A single SyncPose cannot be removed.
Set Transition Times	Click this button to edit the transition times.

16.3 Create Tool

Creating a tool

You can create a robot hold tool by using the **Create Tool Wizard**. The wizard allows you to easily create a tool from an existing part or by using a dummy part to represent a tool. To create a tool complete with tooldata, follow these steps:

1 Click **Create Tool**.

2 In the **Tool Name** box, enter a tool name and choose one of the following options:

Option	Action
Use Existing	Select one of the existing parts from the list. The selected part will represent the tool graphics. The selected part must be a single part. Parts with attachments cannot be selected.
Use Dummy	A cone will be created to represent the tool.

3 Continue entering the **Mass** of the tool, the **Center of Gravity** and the **Moment of Inertia Ix, Iy, Iz**, if these values are known.

> **Note**
>
> If you do not know the correct values, the tool can still be used for programming motions, but this data must be corrected before running the program on real robots or measuring cycle times.

> **Tip**
>
> If the tool is built from materials with a similar density, you can find the center of gravity by clicking the tool model using the **Center of gravity** snap mode.

4 Click **Next**.

5 In the **TCP Name** box, enter a name for the Tool Center Point (TCP).

> **Note**
>
> The default name is the same as the name of the tool. If creating several TCPs for one tool, each TCP must have a unique name.

6 Enter the position of the TCP relative to the world coordinate system, which represents the tool mounting point, by any of the methods below:

Method	Description
Read values from existing target or frame	Click in the **Values from Target/Frame** box, then select the frame either in the graphics window or the **Paths&Targets** browser.

Continues on next page

Method	Description
Enter position and orientation manually.	In the **Position** and **Orientation** boxes, type the values. If **Use Dummy Part** is selected, the position value can not be 0,0,0. At least one coordinate has to be > 0 in order for a cone to be created.

7 Click the arrow right button to transfer the values to the **TCP(s):** box.

If the tool shall have several TCPs, repeat steps 5 to 7 for each TCP.

8 Click **Done**.

The tool is created and appears in the **Layout** browser and in the graphics window.

Creating tooldata for an existing geometry

Ensure to select the robot in which tooldata is created. To create tooldata for an existing geometry, follow these steps:

1 Click **Create Tool** and select **Use Existing** and the imported tool from the list.

2 Enter the requested data in the boxes in the **Create Tool Wizard**.

3 Attach the tool by dragging it to the robot.

What to do next

To make the tool ready to use, do one of the following:

- To make the robot hold the tool, attach the tool to the robot.
- In the graphics window, check the position and orientation of the TCP. If it is incorrect, modify the values in the tool frame part of the tooldata.
- To simplify future usage of the created tool, save it as a library. On the **File** menu, click **Save As Library**. Browse to the folder where you want to store the tool component, enter a name for the tool component and click **Save**.

17 Simulation tab

17.1 Station Logic

Overview

The *station logic* is a typical connection between an IO signal/properties over different items/objects of the station and an output signal/properties from a *virtual controller*. The convenient way to connect different items/objects of a station is to use design view.

Example: In a simulation module, where the gripper is controlled by IO signal you can connect the output signal of the virtual controller to the gripper through the station logic.

Note

The properties are connected through binding and the connection are displayed in red color.

The signals are connected through connections and the connection are displayed through green color.

The Station Logic editor consists of the following tabs:

Tab	Description
Compose	In this option, you can add and edit the *smart components* and all the smart components are listed in child components. The state of the component can be saved to be restored later. The state contains selected modifiable aspects of the component and its child components at the time when the state was saved. • Child components: Lists all smart components. • Saved States: The state of the components can be saved. • Assets: Allows you to browse and select any file as an asset related to RobotStudio
Design	Its a graphical view where all the selected objects/items are listed. Its a most convenient way to connect different items/objects of a station
Properties and Bindings	Displays all object/item properties.
Signals and Connections	Displays all object/item signals.

Connecting smart components in the Design view

In this procedure you can connect the I/O signal/properties over different items/objects of the station and an output signal/properties using **Design view** tab.

1 Click on the **Simulation** tab and select **Station Logic** option.

The **Station Logic** window is displayed.

2 To add a smart component, press "**Add component**" link and select it from the drop down menu.

The object is listed under **Child Components** option.

Continues on next page

3 Go to the **Design** tab.

All the **Smart Components** of the station are listed in **Design** view.

4 To connect two smart components, select and hold the property or I/O signal from the source component, slow drag and drop on the other property or I/O signal of the target component.

The connectivity line is displayed to show that connection is created.

Compose tab

Overview

The Compose tab consists of the following:

- Child components
- Saved States
- Assets

Child components

It is a list box that displays all the objects contained by the component. Objects connected to a library have an overlay that indicates that the objects are locked. Smart Components are displayed first followed by other type of objects.

The following commands are displayed in child components:

Command	Description
Add component	Adds a child object to the component from the list. You can select a built-in base Smart Component, a new empty Smart Component, a library from file or a geometric part from file. Base components are organized as sub-menus based on the usage. For example, *Signals and Properties*, *Sensors*, *Actions* and so on. Recently used base components are listed at the top.
Edit parent	Sets the context of the Editor to the parent of the component that is currently being edited.
Disconnect from library	Disconnects the selected object from library, allowing it to be edited.
Export to XML	Opens a dialog box where you can export and save the component definition along with its properties as an **.rsxml* file

Right-click on the selected object to display the following context menu items:

Item	Description
Edit	Sets the context of the Editor to the selected child object.
Delete	Deletes the child object.
Show in Browser	Indicates if the object should be displayed in the Layout browser.
Set as Role	Sets the object as the **Role** of the component. The Smart Component will inherit certain characteristics of the Role object. For example, attaching a component with a tool as Role to a robot will cause a ToolData to be created.
Properties	Opens the Property editor dialog box for the object.

Continues on next page

Saved States

The *state* of the component can be saved to be restored later. The following commands are available:

Command	Description
Save Current State	Opens the **Save Current State** dialog box.
Restore Selected State	Restores the component to the selected state.
Details	Opens a window that displays detailed information about the selected state.
Delete	Deletes the selected state.

Save Current State

1. In the **Name** text box, enter a name for the state. If a state with the same name already exists, you will be asked to overwrite the existing state.
2. In the **Description** text box, enter the description for the state.
3. In the **Values to save**, select the value to be saved.
4. Select the check box to save the state of all child components.

> **Note**
>
> When working on a station level,
> - In the **Values to save**, you can also select certain *Virtual Controller* values in the saved states.
> - You need not select the option **Recursive** as the state of the station is always saved.

It is possible to include the following states of selected objects while saving and restoring the *state* of a station or a Smart Component. To save these states, in **Values to save**, select the corresponding check-box.

- Property values
- Joint values
- Visibility
- *Physics behavior*
- Position, orientation and attachment

In addition to these object states, following controller states can also be saved.

- Variable values
- I/O signal values

Assets

The assets contained in the component are displayed as grid.

The following commands are available:

Command	Description
Add Asset	Opens a dialog box and allows you to browse and select any file as an asset.

Continues on next page

Command	Description
Update All Assets	Replaces the data of all the assets with the data of the corresponding file on the disk. If the file is not available, a warning message is displayed in the output window.
View	Opens the selected asset in the associated program.
Save	Opens a dialog box and allows you to save the selected asset.
Delete	Deletes the selected asset.

Note

The text resources (descriptions) for properties and signals are stored in an asset called *Resources.<language-id>.xml*. If this is deleted, the texts for that language will be empty and the default (English) will be used. The default language when authoring a component is always English, regardless of the application language.

Design View

Its a graphical view where all the selected objects/items are listed. Its a most convenient way to connect different items/objects of a station.

Command	Description
Show Bindings	Displays the property connections.
Show Connections	Displays the signal connections.
Show unused	Displays all unused objects which are not connected.
Zoom	You can zoom the size of the object as required using the slider or for default size click on **Auto Arrange** option.

Properties and Bindings tab

Overview

The Properties and Bindings tab consists of the following:

- Property Bindings

Property Bindings

The property bindings contained in the component are displayed in a grid.

The following commands are available:

Command	Description
Add Binding	Opens the **Add Binding** dialog box.
Add Expression Binding	Opens the **Add Expression Binding** dialog box.
Edit	Opens the Edit Binding or Edit Expression Binding dialog box, depending on the type of binding selected.
Delete	Deletes the selected binding.

Add or Edit Binding

The **Add Binding** dialog box allows you to create or edit a property binding.

Continues on next page

The following options are available:

Control	Description
Source Object	Specifies the owner of the source property.
Source Property	Specifies the source of the binding.
Target Object	Specifies the owner of the target property.
Target Property or Signal	Specifies the target of the binding. Only properties of the same type as the source property type are listed.
Allow cyclic binding	Allows the target property to be set two times in the same context, which otherwise generates an error. The target list box, besides dynamic properties also displays some common properties such as object transform that can only be used as target and not as source.

Add or Edit Expression Binding

The **Add Expression Binding** dialog box allows you to specify a mathematical expression as the source of a property binding.

The following controls are available:

Control	Description
Expression	Specifies the mathematical expressions. The following lists the allowable mathematical expressions: • **Allowed operators**: +, - (unary and binary) *, /, ^ (power), Sin(), Cos(), Sqrt(), Atan() and Abs(). • **Allowed operands**: Numeric constants, PI, and Numeric dynamic properties on the current smart component and any child smart components. The text box has the IntelliSense-like funtionality which allows you to select from the available properties. If the expression entered in the text box is invalid, an error icon is displayed.
Target Object	Specifies the owner of the target property.
Target Property	Specifies the target of the binding. Only numeric properties are listed.

Signals and Connections tab

Overview

The Signals and Connections tab consists of the following:

- I/O Signals
- I/O Connections

I/O Signals

The **I/O Signals** contained in the component are displayed in a grid.

The following commands are available:

Command	Description
Add I/O Signals	Opens the **Add I/O Signals** dialog box.
Expose Child Signal	Opens the **Expose Child Signal** dialog box.
Edit	Opens the **Edit Signal** dialog box.

Continues on next page

Command	Description
Delete	Deletes the selected signal.

Add or Edit I/O signals

The **Add I/O Signals** dialog box allows you to edit an I/O signal, or add one or more I/O signals to the component.

The following controls are available:

Control	Description
Type of Signal	Specifies the type and direction of the signal. The following are the available types of signals: • Digital • Analog • Group
Signal Base Name	Specifies the name of the signal. The name must contain an alphanumeric character and start with a letter (a-z or A-Z). If more than one signal is created, numeric suffixes specified by Start Index and Step are added to the names.
Signal Value	Specifies the initial value of the signal.
Auto-reset	Specifies that a digital signal should have transient behavior. This applies to digital signals only. Indicates that the signal value is automatically reset to 0.
Number of Signals	Specifies the number of signals to create.
Start Index	Specifies the first suffix when creating multiple signals.
Step	Specifies the suffix interval when creating multiple signals.
Minimum	Specifies the minimum value for an analog signal. This applies to Analog signal only.
Maximum	Specifies the maximum value for an analog signal. This applies to Analog signal only.
Hidden	Indicates if the property should not be visible in GUI such as the Property Editor and I/O Simulator.
Read only	Indicates if the property value should be possible to modify in GUI such as the Property Editor and I/O Simulator.

Note

When editing an existing signal, only the **Signal Base Name** and **Signal Value** can be modified, while all other controls are locked.

If the input is valid,**OK** is enabled allowing you to create or update the signal. If not, an error icon is displayed.

Expose Child signal

The **Expose Child Signal** dialog box allows you to add a new I/O signal that is connected to a signal in a child object.

Continues on next page

The following controls are available:

Control	Description
Signal Name	Specifies the name of the signal to be created. By default, it is the same as the name of the selected child signal.
Child Object	Specifies the object for which to expose a signal.
Child Signal	Specifies the child signal.

I/O Connections

The **I/O Connections** contained in the component are displayed in a grid.

The following controls are available:

Control	Description
Add I/O Connection	Opens the **Add I/O Connection** dialog box.
Edit	Opens the **Edit I/O Connection** dialog box.
Delete	Deletes the selected connection.

Add or Edit I/O Connection

The **Add I/O Connection** dialog box allows you to create an I/O connection or edit an existing connection.

The following controls are available:

Control	Description
Source Object	Specifies the owner of the source signal.
Source Signal	Specifies the source of the connection. The source must either be an output from a child component, or an input to the current component.
Target Object	Specifies the owner of the target signal.
Target Signal or Property	Specifies the target of the connection. The target must be of the same type as the source, and either an input to a child component or an output from the current component.
Allow cyclic connection	Allows the target signal to be set two times in the same context, which would otherwise generate an error.

17.2 TCP Trace

The TCP Trace tab

Enable TCP Trace	Select this check box to activate tracing of the TCP path for the selected robot. **Note** For TCP trace to work correctly, ensure that the workobjects and tools used by the program are synchronized to the station.
Follow moving Workobjects	Select this check box to activate tracing of moving workobject.
Clear trace at simulation start	Select this check box to remove the current trace when simulation starts.
Primary color	You can set the color of the trace here.
Color by signal	Select this check box to assign a particular color to the TCP path of the selected signal.
Use color scale	Select this button to define how the trace shall be colored. As the signal changes between the values defined in the **From** and **To** boxes, the color of the trace also varies according to the color scale.
Use secondary color	You can assign a color to the trace which gets displayed when the signal value meets the specified conditions.
Show events	Select this check box to view events along the trace.
Clear TCP traces	Click this button to remove the current trace from the graphics window.
Show stop position	Select *category 0 stop* or *category 1 stop* and color to visualize the corresponding stop positions along trace.

18 Controller tab

18.1 Add Controller

Overview

Use the **Add Controller** feature for connecting to a robot controller that is connected to the management port or to virtual controllers.

> **Note**
>
> RobotWare options must be selected while connecting RobotStudio to robot controller over Ethernet (LAN).
>
> - For controllers with RobotWare 7, the required *RobotWare* option is **3119-1 RobotStudio Connect**.
> - For controllers with RobotWare 6, the required *RobotWare* option is **616-1 PC Interface**.
>
> This feature is not required when connecting through the management port.

On the **Controller** tab, click the arrow next to the **Add Controller** icon, and then click one of the following options as per the requirement:

- **One Click Connect**: for connecting to a robot controller that is connected to the management port.
- **Connect to Controller**: for connecting to a robot controller from the network, a virtual controller, or a controller from a device list.

Local and remote clients

Users can add and connect to a robot controller as a local or remote client. When logged on as a local client, the user will be able to get write access in manual mode and will be granted permission to perform operations that are usually allowed only from the FlexPendant, for example, starting execution, moving the program pointer and so on.

When a controller is in manual mode, requests for write access by remote clients must be approved by a local client, such as the FlexPendant. To approve the write access, the enabling device on the FlexPendant has to be pressed a number of

Continues on next page

times. Once logged on as a local client, write access in manual mode is granted without approval.

> **Note**
>
> - Local login is useful when the controller is in manual mode. When the controller is in *auto* mode, write access is granted for clients without local login.
> - Virtual controllers that are part of a station do not require write access (RobotStudio acts as local client). Whereas the write access and *Login as local client* for virtual controllers that are not part of a station are similar to that of the robot controllers.
> - OmniCore supports *Login as local client* only when it is connected over management port.

One Click Connect

Prerequisites for using the **One Click Connect** feature.

- Connect the PC to the management port .
- Ensure correct network settings on the PC, enable DHCP or provide static IP address.

Use the following steps to connect to a robot controller.

1. On the **Controller** tab, click the arrow next to the **Add Controller** icon, and then click **One Click Connect**.
2. The **Login** dialog opens, enter valid credentials and click **OK**.
3. On successful login, in the **Controller** browser, under **Management Port**, the connected robot controller will be visible.

Connecting to a robot controller on the network

1. In the **Controller** tab, click **Add Controller** and then click **Connect to Controller**.

   The **Connect to Controller** window opens.
2. In the **Connect to Controller** window, click the **Network** tab.

   By default, all available robot controllers on the network are listed here. To view the virtual controllers on the network, click the **Show virtual controllers on the network** check-box.
3. To connect to a specific controller on the network, type-in its IP address in the **Locate Remote Controller** box, and then click **Locate**.
4. Select the controller and click **OK**.
5. The **Login** dialog opens, enter valid credentials and click **OK**.

Connecting to a controller in Low Bandwidth mode

The Low Bandwidth mode is useful when connecting RobotStudio to a controller over a network connection with limited bandwidth. In Low Bandwidth mode, the controller communication is considerably reduced when compared with the regular connection mode.

Continues on next page

When a controller is connected in Low Bandwidth mode:

- the signal analyzer gets disabled.
- system requires manual refresh to update the I/O Viewer, RAPID Watch window, RAPID Program Pointer, Online Monitor and FlexPendant Viewer in RobotStudio with the current controller status, hence manual buttons are added to initiate the refresh.
- semantic check in the RAPID editor will be restricted.

Use the following steps to connect to a controller in Low Bandwidth mode:

> **Note**
>
> RobotWare version installed in the controller and in the connecting PC must be same.

1. In the **Connect to Controller** window, click the **Network** tab.
2. Select the required controller and then click the **Low bandwidth** check-box and click **OK**.
3. The **Login** dialog opens, enter valid credentials and click **OK**.

Starting a virtual controller

1. In the **Controller** tab, click **Add Controller** and then click **Connect to Controller**.

   The **Connect to Controller** window opens.
2. In the **Connect to Controller** window, click the **Virtual Controllers** tab.
3. Click the **...** button, the **Select Folder** dialog opens, select the required folder and click **Select Folder**.

   All virtual controllers from the selected folder will be displayed. The **Start Controller** check-box is selected by default.

   > **Note**
   >
   > When the **Start Controller** check-box is not selected, a virtual controller (inactive state) will be added in the **Controller** browser.
4. Select the required controller and click **OK**.

Local login: User can log in to a virtual or robot controller as local or a remote client. In **Local client**, the user will be logged on as a local user where the program execution can be started in manual mode, in addition, program pointer can be set in manual mode.

The opposite of a local user is a remote user, which is the default user of RobotStudio. The privileges of a remote user is restricted when the controller is in manual mode. When compared to a local user, a remote user cannot start program execution or set the program pointer.

Continues on next page

Handle Write Access Automatically: Selecting this check box initiates write access in the selected controller automatically.

Creating a new virtual controller

1 In the **Virtual Controllers** tab, click the **New Controller** button. The **New Virtual Controller** dialog opens.
2 To create a new virtual controller:
   a Click **Create new**, and then select the required **Robot model**, **RobotWare** and **Controller**.
   b Select **Customize options** check-box and then click **OK**, to open the **Change Options** window to add options.
   c Click **OK**. The new controller gets added to the list of **Virtual Controllers**.
3 To create a virtual controller from backup:
   a Click **Create from backup**, then under **Select backup**, click **....** The **Select Folder** window opens.
   b Select the backup, and then click **Select Folder**, to view the details of the backup (name, RobotWare and so on) in the **New Virtual Controller** window.
   c Click **OK**. The new controller gets added to the list of **Virtual Controllers**

Deleting a virtual controller

1 In the **Virtual Controllers** tab, from the list of virtual controllers, select the controller, and then click the **Delete Controller** button.
2 The **Delete Virtual Controller** dialog opens, click **OK** to delete the selected virtual controller.

Connecting to a controller from the device list

A device list contains a selected set of controllers on a network that are identified by IP Address or DNS name. Device list can be saved to disk in *.xlsx* format. It is possible to connect to a controller from the device list.

1 In the **Connect to Controller** window, click the **Device List** tab.

   The most recently saved device list with the controllers get displayed.
2 Click the **...** button, the **Select a device list** dialog opens, select the device list and click **Open**.
3 All controllers that are part of the list gets displayed. Click the controller to connect and then click **OK**. The authentication dialog opens, type-in the credentials and click **Login**.

Click **Manage Device Lists** to open the **Jobs** tab.

18.2 Authenticate

Overview

Introduction

The data, functionality, and commands on a controller are protected by a User Authorization system (also called UAS). The UAS restricts the parts of the system the user has access to. Different users can have different access grants.

You can perform the following functions from the **Authenticate** menu:

- Login as a Different User
- Log off
- Log off all controllers
- Edit User Accounts
- UAS Grant Viewer

Login as a Different User

1. In the **Authenticate** menu, click **Login as a Different User**. The **Add new user** dialog box appears.
2. In the **User Name** box, enter the user name you want to log on as.
3. In the **Password** box, enter the password for the user name you are logging on as.
4. Click **OK**.

Log off

In the **Authenticate** menu, click **Log off** to log the user off from the controller.

Login off all controllers

In the **Authenticate** menu, click **Log off** to log the user off from all the controllers.

Managing user rights and write access on an IRC5 controller

Overview

User Authorization System (UAS) restricts user access to the controller data and functionalites. These functionalities are categorized and protected by UAS grants. There are two types of grants; controller grants and application grants. Controller grants are predefined and provided by *RobotWare*. Application grants are defined by RobotWare add-ins. These grants are managed using the **UAS Administration Tool**.

UAS grants are viewable using the UAS grant viewer. The *UAS Grant Viewer* page displays information about the grants of the current user. In the **Authenticate** menu, click **UAS Grant Viewer** to open the viewer.

Group

Group is a collection of grants that represents user roles. The available user roles are administrator, programmer, operator and user defined. User inherits the grants of the group it is associated to.

All the controllers have a preset group and preset user named *Default Group* and *Default User* respectively. The *Default User* has an open password *robotics*. The

Continues on next page

Default Group and User cannot be removed and the password cannot be changed. However, the user with the user grant *Manage UAS settings* can modify the controller grants and application grants of the default user.

You can deactivate the *Default User* except for RobotWare 6.04 and earlier. Before deactivating the default user, it is recommended to define at least one user with the grant *Manage UAS settings* so as to continue managing users and groups.

Write access

Write access is required to change data on a controller. The controller accepts a single user with write access at a time. RobotStudio users can request write access to the system. If the system is running in manual mode, the request for write access is accepted or rejected on the FlexPendant. User loses write access if the mode changes from manual to automatic, or vice versa. If the controller is in manual mode, then the write access can be revoked from the FlexPendant.

Adding a user to the administers group

In addition to the *Default Group*, certain predefined user groups are available in the *robot controller*. The predefined groups are, *Administrator*, *Operator*, *Service* and *Programmer*. The Administrator group has the controller grant *Full access* enabled.

1. On the **Controller** tab, click **Add controller** and then click **Add Controller..** and then select the controller from the **Add Controller** dialog.
2. On the **Controller** tab, click **Request Write Access**.
3. Click **Authenticate** and then click **Edit User Accounts**.

   UAS Administration Tool opens.
4. On the **Users** tab, click **Add**. The **Add new user** dialog opens.
5. In the **User Name** and **Password** boxes, enter suitable values. Click **OK**.

   The new user gets added to the **Users on this controller** list.
6. Select the user, and then from the User's groups, click the **Administrator** check box.
7. Click **OK**. The new user gets added to the **Administrator** group.

Use the same steps to create users for various groups.

> **Note**
>
> To view the Controller/Application grants assigned to a particular group, in the **UAS Administration Tool**, on the **Groups** tab, select the group and then select the particular category of grant.

Creating a new user group

1. In the **UAS Administration Tool**, click the **Groups** tab.
2. On the **Groups** tab, click **Add**. The **Add new group** dialog opens.
3. Enter the required details and click **OK**.

   The new group gets added.

Modifying an existing user group

1. In the **UAS Administration Tool**, click the **Groups** tab.

Continues on next page

2 On the **Groups** tab, Select the group and then click **Edit**. Enter the required changes and click **OK**.

Creating a new user

1 In the **UAS Administration Tool**, click the **Users** tab, and then click **Add**. The **Add new user** dialog opens.
2 In the **User Name** and **Password** boxes, enter suitable values. Click **OK**.
The new user gets added to the **Users on this controller** list.
3 Select the user, and then from the **User's groups**, click the group to which the user must be added.
4 Click **OK**. The new user gets added to the **Administrator** group.

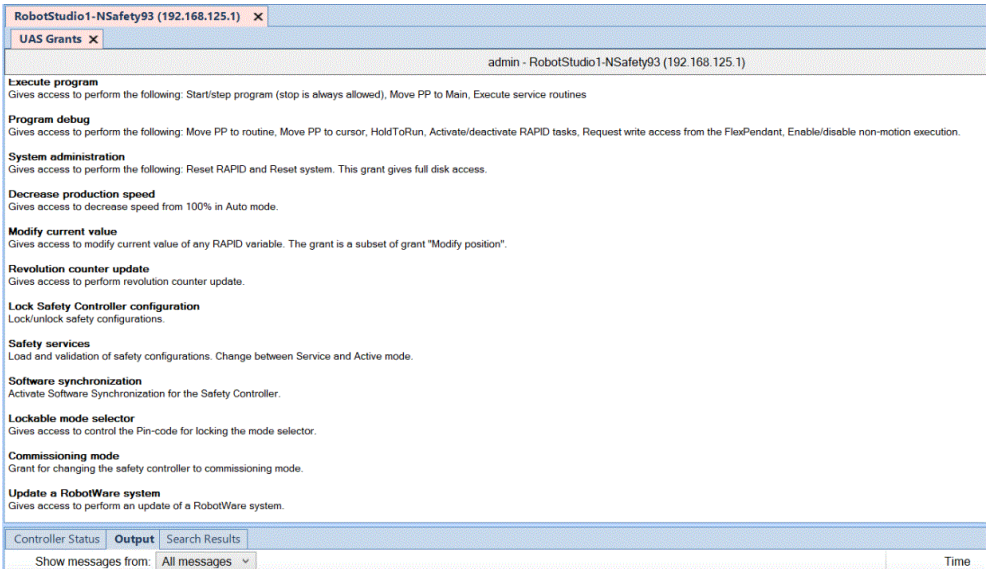
Modifying an existing user

1 In the **UAS Administration Tool**, click the **Users** tab.
2 On the **User** tab, select the group from **User's groups** and then select the required user.
3 Click **Edit**. Enter the required changes and click **OK**.

UAS Grant Viewer

The UAS Grant Viewer page displays information about the grants provided to the user currently logged in and the groups owning them.

1 In the **Authenticate** menu, click **UAS Grant Viewer**. The **UAS Grants** window appears.



en0900000852

Examples of common actions to perform

Action	Necessary grants
Rename the controller (A restart of the controller is necessary)	Modify controller properties Remote restart
Change system parameters and load configuration files	Modify configuration Remote restart

Continues on next page

18.2 Authenticate
Continued

Action	Necessary grants
Install a new system	Administration of installed system
Perform a backup (A restart of the controller is necessary)	Backup and save Remote restart
Restore a backup (A restart of the controller is necessary)	Restore a backup Remote restart
Load/delete modules	Load program
Create new module	Load program
Edit code in RAPID modules	Edit RAPID code
Save modules and programs to disk	Backup and save
Start program execution from Task Window	Execute program
Create a new I/O signal, that is, add a new instance of the type Signal (A restart of the controller is necessary)	Modify configuration Remote restart
Set the value of an I/O signal	I/O write access
Access to controller disks from File Transfer window	Read access to controller disks Write access to controller disks

Controller grants

Full access	This grant includes all controller grants, also new grants added in future RobotWare versions. The grant does not include any application grants or the *Safety Controller configuration* grant.
Manage UAS settings	Gives access to read and write the UAS configuration, that is to read, add, remove and modify UAS users and groups.
Execute program	Gives access to perform the following: • Start/step program (stop is always allowed) • Move PP to Main • Execute service routines
Perform ModPos and HotEdit	Gives access to perform the following: • Modify or teach positions in RAPID code (ModPos) • During execution modify positions in RAPID code as single points or as a path (HotEdit) • Restore ModPos/HotEdit positions to original • Modify current value of any RAPID variable
Modify current value	Gives access to modify current value of any RAPID variable. This grant is a subset of the grant *Perform ModPos and HotEdit*.
I/O write access	Gives access to perform the following: • Set I/O signal value • Set signal as simulated and remove simulation • Set the device and industrial network as enabled/disabled
Backup and save	Gives access to perform a backup and to save modules, programs and configuration files. The grant gives full FTP access to the current systems BACKUP and TEMP directory.
Restore a backup	Gives access to restore backup and restart the controller using the restart mode **Revert to last auto saved**.

Continues on next page

Modify configuration	Gives access to modify the configuration database, that is to load configuration files, change system parameter values and add/delete instances.
Load program	Gives access to load/delete modules and programs.
Remote restart	Gives access to perform restart and shutdown from a remote location. No grant is required to perform restart via a local device, as for example the FlexPendant.
Edit RAPID code	Gives access to perform the following: • Modify code in existing RAPID modules • Frame calibration (tool, workobj) • Commit ModPos/HotEdit positions to current values • Rename program
Program debug	Gives access to perform the following: • Move PP to routine • Move PP to cursor • HoldToRun • Activate/deactivate RAPID tasks • Request write access from the FlexPendant • Enable/disable non-motion execution
Decrease production speed	Gives access to decrease speed from 100% in Auto mode. This grant is not required if speed is already below 100%, or controller is in Manual mode.
Key-less mode selector	Unlock key-less mode selector.
Calibration	Gives access to perform the following: • Fine calibrate mechanical unit • Calibrate *base frame* • Update/clear SMB data Frame calibration (tool, wobj) requires the grant *Edit RAPID code*. Manual offset of mechanical unit calibration data and loading new calibration data from file require the grant *Modify configuration*.
Administration of installed systems	Gives access to perform the following: • Install new system • Reset RAPID • Reset system • Start Boot Application • Select System • Install system from device This grant gives full FTP access, that is, the grant gives the same rights as *Read access to controller disks* and *Write access to controller disks*.
Read access to controller disks	Gives external read access to controller disks. This grant is only valid for explicit disk access, for example with an FTP client or the File Manager of RoboStudio. It is possible, for example, to load a program from /hd0a without this grant.
Write access to controller disks	Gives external write access to controller disks. This grant is only valid for explicit disk access, for example with an FTP client or the File Manager of RoboStudio. It is possible, for example, to save a program to the controller disk or perform a backup without this grant.
Modify controller properties	Gives access to set controller name, controller ID and system clock.

Continues on next page

Delete log	Gives access to delete messages in the controller Event Log.
Revolution counter update	Gives access to update the revolution counter.
Safety Controller configuration	Gives access to perform a configuration of the Safety Controller. This is valid only for the PSC-option and is not included in the *Full access* grant.

Application grants

Access to the ABB menu on FlexPendant	Value **true** gives access to the ABB menu on the FlexPendant. This is the default value if a user does not have the grant. Value **false** means that the user cannot access the ABB menu when the controller is in Auto mode. The grant has no effect in Manual mode.
Log off FlexPendant user when switching to Auto mode	A user having this grant is automatically logged off from the FlexPendant when switching from Manual mode to Auto mode.

Managing user rights and write access on an OmniCore controller

Overview

The OmniCore controller is delivered with a preset user named *Default User*. This user is assigned certain grants by default and it belongs to the role *Operator*. If a new user is created with specific grants, the *Default User* can be removed. An active *Default User* has *read only* rights to the controller data even if all grants are removed. Hence, to prevent any unauthorized access to OmniCore controller data, the *Default User* must be deleted.

The OmniCore controller is delivered with a default configured user, named *Admin*. All UAS grants are assigned against this user, such as, adding, removing and modifying users. The *Admin* user belongs to the role *Administrator* by default. You can deactivate the *Admin* user. Before deactivating default users, it is recommended to define at least one user with the grant *Manage UAS settings* so as to continue managing users and roles.

Controller grants

Manage UAS settings	Gives access to read and write the UAS configuration, that is to read, add, remove and modify UAS users and groups.
Modify system parameters	Gives access to modify the system parameters, that is load system parameter files, change system parameter values and add/delete instances.
Backup and save	Gives access to perform a backup and to save modules, programs and system parameter files. This grant gives read/write access to the BACKUP folder.
Modify current value	Gives access to modify current value of any RAPID variable. This grant is a subset of the grant *Perform ModPos and HotEdit*.
Modify controller properties	Gives access to set controller name, system clock and WAN IP configuration.
Modify network security properties	Gives access to set network security settings, such as firewall configuration and syslog server.

Continues on next page

Delete log	Gives access to delete messages in the controller Event Log.
Read access to controller disks	Gives external read access to controller disks. This grant is only valid for explicit disk access, for example with the File Manager of RoboStudio or with RWS file service. **Note** • TEMP folder is always possible to read even without this grant. • This grant doesn't give access to the BACKUP folder. • This grant doesn't give access to remote mounted devices.
Write access to controller disks	Gives external read and write access to controller disks. This grant is only valid for explicit disk access, for example with the File Manager of RoboStudio or with RWS file service. **Note** • It is always possible to write to the TEMP folder even without this grant. • This grant doesn't give access to the BACKUP folder. • This grant doesn't give access to remote mounted devices.
IO write access	Gives access to perform the following: • Set I/O signal value • Set signal as simulated and remove simulation • Set I/O unit and bus as enabled/disabled.
Remote restart	Gives access to perform system restart and main computer shutdown from a remote location. No grant is required to restart the system through a local device, for example, the FlexPendant.
Restore a backup	Gives access to restore backup. This grant gives read access to the BACKUP folder
System administration	Gives access to Reset RAPID and to Reset system. This grant gives full disk access.
Edit RAPID code	Gives access to perform the following: • Modify code in existing RAPID modules • Rename program.
Load program	Gives access to load/delete modules and program.
Modify position	Gives access to perform the following: • Modify positions in RAPID code • Modify current value of any RAPID variable.
Modify current value	Gives access to modify current value of any RAPID variable. The grant is a subset of grant **Modify position**.
Execute program	Gives access to perform the following: • Start/step program (stop is always allowed) • Move Program Pointer to Main • Execute service routines • Set run mode and cycle

Continues on next page

Program debug	Gives access to perform the following: • Move Program Pointer to routine, • Move Program Pointer to cursor • HoldToRun • Activate/deactivate RAPID tasks • Request write access from the FlexPendant • Enable/disable non-motion execution.
Calibration	Gives access to perform the following: • Fine calibrate mechanical unit • Calibrate base frame • Update/clear SMB data. **Note** Frame calibration (tool, wobj) requires the grant **Edit RAPID code**. Manual offset of mechanical unit calibration data and loading a new calibration data from file requires the grant **Modify system parameters**.
Revolution counter update	Gives access to perform revolution counter update.
Decrease production speed	Gives access to decrease speed from 100% in Auto mode.
Lock Safety Controller configuration	Lock/unlock safety configurations.
Safety services	Load and validate safety configurations. Changes between Service and Active mode.
Software synchronization	Activate Software Synchronization for the Safety Controller.
Commissioning mode	Grant for changing the safety controller to commissioning mode.
Lockable mode selector	Gives access to control the Pin-code for locking the mode selector.
Update a RobotWare system	Gives access to perform an update of a RobotWare system.
Remote login	A user with this grant can request FlexPendant to login as another user.
Remote Start and Stop in Auto	A remote user with this grant can start and stop program in Auto mode.
Read files on remote mounted devices	A user with this grant have access to read files on a remote mounted device.
Read and write files on remote mounted devices	A user with this grant have access to read and write files on a remote mounted device.
Detach the FlexPendant	A user with this grant can detach the FlexPendant in automatic mode without causing any stops. The hot swappable FlexPendant option is required.
Log off FlexPendant user when switching to Auto	A user with this grant is automatically logged off from the FlexPendant when switching from Manual mode to Auto mode.

Note

Application grants are not supported in OmniCore systems.

Continues on next page

Adding a user to the Administrators group

The predefined user roles available in the *robot controller* are *Administrator* and *Operator*. For the Administrator role the controller grant *UAS_ADMINISTRATION* is enabled by default.

1. On the **Controller** tab, click **Add controller** and then click **Add Controller..** and then select the controller from the **Add Controller** dialog.
2. On the **Controller** tab, click **Request Write Access**.
3. Click **Authenticate** and then click **Login as Different User**. The **Login** dialog opens, enter the default credentials **User Name** and password as *Admin* and *robotics* respectively and click **Login**.
4. Click **Authenticate** and then click **Edit User Accounts**.

   The **Edit User Accounts** window opens.
5. On the **Users** tab, click **Add user**.
6. Enter suitable values in the fields as required and then under **Roles** select the **Administrator** check box. Click **Apply**.

   The new user gets added to the **Users on this controller** list.

Use the same steps to create users for various roles.

Creating a new user role

1. In the **Edit User Accounts**, click the **Roles** tab.
2. On the **Roles** tab, click **Add Role**.
3. Enter the required details and click **Apply**.

   The new role gets added to the selected user.

Modifying an existing user role

1. In the **Edit User Accounts**, click the **Roles** tab.
2. On the **Roles** tab, select the role and then click **Edit User**. Enter the required changes and click **Apply**.

Creating a new user

1. In the **Edit User Accounts**, click the **Users** tab, and then click **Add User**.
2. In the **User Name** and **Password** boxes, enter suitable values. Select the required roles and then click **Apply**.

   The new user gets added to the **Users on this controller** list with the selected roles.

18.3 Events

Overview

You can view events in the **Event Log**. The severity of each event is indicated by its background color; blue for information, yellow for warning and red for an error which needs to be corrected to proceed.

On the **Controller** tab in the Controller Tools group, click **Events** to view the **Event Log**.

Types of Events

Name of the Event	Description
Operational events	Events dealing with handling of the system.
System events	Events dealing with system functions, system states, and so on.
Hardware events	Events dealing with system hardware, manipulators as well as controller hardware.
Program events	Events dealing with RAPID instructions, data, and so on.
Motion events	Events dealing with the control of the manipulator movements and positioning.
I/O events	Events dealing with inputs and outputs, data buses, and so on.
User events	Events defined by the user.
Functional safety events	Events related to functional safety.
Process events	Application specific events, arc welding, spot welding, and so on.
Configuration events	Events dealing with the configuration of the system.
Connected Service events	Connected Service Embedded event logs which are generated during starting, registering, unregistering, losing connectivity, and so on.

18.4 Configuration

Configuration

From the Configuration you can view and edit the system parameters in a controller. The Configuration has a direct communication with the controller. This means that changes you make are applied to the controller as soon as you complete the action.

With the Configuration Editor, the following actions can be performed for each topic:

- view types, instances, and parameters
- edit instances and its parameters
- copy and paste instances within a type
- add and delete instances

Layout of the Configuration

The Configuration consists of the Type name list and the Instance list.

The **Type name** list displays all available configuration types for the selected topic. The list of type names can not be edited.

The **Instance** list displays all system parameters of the selected type in the **Type name** list. Each row in the list is an instance of the type. The columns show the parameters and their values.

The Configuration editor includes the following:

- Configuration topics
  - Communication
  - Controller
  - I/O
  - Man-machine communication
  - Motion
  - PROC
- Add Signals
- I/O Engineering Tool

Add Signals

This is a special window to add several I/O signals at the same time. You need to have write access to the controller to be able to open the add signal window.

Type of Signal	Defines the type of signal.
Signal Base Name	Defines the name for one or more signals.
Assigned to Device	Defines the device to which the signal belongs.
Signal Identification Label	Optionally, offers filtering and sorting based on this category.
Number of Signals	Defines the number of signals to add in a range.
Start Index	Defines the index (number) to start the range with.
Step	Defines the number which the index should increase with.
Device Mapping Start	Defines the bits in the I/O memory map of the assigned unit to which the signal is mapped.

Continues on next page

Category	Optionally, offer filtering and sorting based on this category.
Access Level	Defines the write access to I/O signals for categories of I/O controlling clients connected to the robot controller. This field is enabled only when Advance check-box is selected. Not necessarily Write access. Options are Default, ReadOnly and All.
Default Value	Specifies the I/O signal value to be used at the start.
Invert Physical Value	Applies an inversion between the physical value of the signal and its logical representation in the system.

Viewing configurations

1 To view the topics of a controller, from the **Controller** tab, expand the **Configuration** node for the controller.

All topics in are now displayed as child nodes to the Configuration node.

2 To view the types and instances of a topic, double-click the topic node for the topic to view.

The Configuration Editor is now opened, listing all types of the topic in the **Type name** list. In the **Instance** list, each instance of the selected type in the **Type name** list is displayed as a row. The parameter values of the instances are displayed in the columns of the instance list.

3 To view detailed parameter information for an instance, double-click the instance.

The instance editor now displays the current value, restrictions, and limits of each parameter in the instance.

Editing parameters

You can either edit the parameters of one single instance, or you can edit several instances at the same time.

1 In the **Controller** tab, expand the **Controller** and the **Configuration** node and double-click the topic that contains the instances to edit.

This opens the Configuration Editor.

2 In the **Type name** list of the Configuration Editor, select the type that the instance to edit belongs to.

The instances of the type are now displayed in the Instance list of the Configuration Editor.

3 In the **Instance** list, select the instances to edit and press the Enter Key. To select several instances at once, hold down the SHIFT or CTRL key while selecting.

Alternatively, right-click an instance and then click **Edit**.

The Instance Editor is now displayed.

4 In the Parameter list of the Instance Editor, select the parameter to edit and change the value of the parameter in the **Value** box.

When editing several instances at one time, the parameter values you specify will be applied to all instances. For parameters that you do not specify any new value, each instance will keep its existing value for that parameter.

Continues on next page

5 Click **OK** to apply the changes to the configuration database in the controller.

For many parameters, the changes will not take effect until the controller is restarted. If your changes require a restart, you will be notified of this.

You have now updated the controller's system parameters. If you are going to make several changes, you can wait with the restart until all changes are done.

Adding instances

With the Configuration Editor, you can select a type and create a new instance of it. For example, adding a new instance of the type Signal creates a new signal in the system.

1 In the **Controller** tab, expand the **Controller** and the **Configuration** node and double-click the topic that contains the type of which you want to add an instance.

This opens the Configuration Editor.

2 In the **Type name** list of the Configuration Editor, select the type of which you want to add an instance.

3 Right-click anywhere in the configuration editor or on the Type node and then select **New Type** from the context menu.

A new instance with default values is added and displayed in the **Instance Editor** window.

4 If required, edit the values.

5 Click **OK** to save the new instance.

The values in the new instance are now validated. If the values are valid, the instance is saved. Otherwise, you will be notified of which parameter values to correct.

For many instances, the changes will not take effect until the controller is restarted. If your changes require a restart, you will be notified of this.

You have now updated the controller's system parameters. If the changes require a restart of the controller, the changes will not take effect until you do this. If you are going to make several changes, you can wait with the restart until all changes are done.

Copying an instance

1 In the **Controller** tab, expand the **Controller** and the **Configuration** node and double-click the topic that contains the instance to copy.

This opens the Configuration Editor.

2 In the **Type name** list of the Configuration Editor, select the type of which you want to copy an instance.

3 In the **Instance** list, select the instance to copy.

4 Right-click the instance to copy and then select **Copy Type** from the context menu.

5 Change the name of the new instance. If required, also edit the other parameter values.

Continues on next page

6 Click **OK** to save the new instance.

The values in the new instance are now validated. If the values are valid, the instance is created. Otherwise, you will be notified of which parameter values to correct.

For many instances, the changes will not take effect until the controller is restarted. If your changes require a restart, you will be notified of this.

You have now updated the controller's system parameters. If the changes require a restart of the controller, the changes will not take effect until you do this. If you are going to make several changes, you can wait with the restart until all changes are done.

Deleting an instance

1 In the **Controller** tab, expand the **Controller** and the **Configuration** node and double-click the topic that contains the type of which you want to delete an instance.

This opens the Configuration Editor.

2 In the **Type name** list of the Configuration Editor, select the type of which you want to delete an instance.

3 In the **Instance** list, select the instance to delete.

4 Right-click the instance to delete and then select **Delete Type** from the context menu.

5 A message box is displayed, asking if you want to delete or keep the instance. Click **Yes** to confirm that you want to delete it.

For many instances, the changes will not take effect until the controller is restarted. If your changes require a restart, you will be notified of this.

You have now updated the controller's system parameters. If the changes require a restart of the controller, the changes will not take effect until you do this. If you are going to make several changes, you can wait with the restart until all changes are done.

Save a configuration file

The system parameters of a configuration topic can be saved to a configuration file, stored on the PC or any of its network drives.

The configuration files can then be loaded into a controller. They are thereby useful as backups, or for transferring configurations from one controller to another.

1 In the **Controller** tab, expand the **Configuration** node and select the topic to save to a file.

2 In the **Controller** tab click **Save Parameters**.

You can also right-click the topic and then select **Save Parameters** from the context menu.

3 In the **Save As** dialog box, browse for the folder to save the file in.

4 Click **Save**.

Saving several configuration files

1 In the **Controller** tab, select the **Configuration** node, not any of its types.

Continues on next page

2 In the **Controller** tab click **Save System Parameters**.

  You can also right-click the Configuration node and then click **Save Parameters**.

3 In the **Save System Parameters** dialog box, select the topics to save to files. Then click **Save**.

4 In the **Browse for Folder** dialog box, browse for the folder to save the files in, and then click **OK**.

  The selected topics will now be saved as configuration files with default names in the specified folder.

Loading a configuration file

A configuration file contains the system parameters of a configuration topic. They are thereby useful as backups, or for transferring configurations from one controller to another.

When loading a configuration file on a controller, it must be of the same major version as the controller. For instance, you cannot load configuration files from an IRC 5 system to an OmniCore controller.

1 In the **Controller** tab, select the **Configuration** node.

2 In the **Controller** tab click Load Parameters.

  You can also right-click the configuration node and then select **Load Parameters** from the context menu.

  This opens the **Open** dialog box.

3 In the **Open** dialog, browse to the configuration file to load.

4 In the **Open** dialog box, select how you want to combine the instances in the configuration file to load with the existing in the controller:

If you want to	then
replace the entire topic with the contents in the file.	select **Delete the existing topic and load instances**
add new instances from the configuration file to the topic, without modifying the existing ones.	select **Load if no existing instances**
add new instances from the file to the topic and replace the existing ones with values from the file. Instances that only exist in the controller and not in the configuration file will not be changed at all. Instances that exist in the controller will first be deleted and then loaded with values from the file, that is, it will get default values for parameters not listed in the file.	click **Load and replace existing instances**

5 Click **Open** to load.

6 In the information box, click **OK** to confirm that you want to load the instances from the configuration file.

If a restart of the controller is necessary for the changes to take effect, you will be notified of this.

Continues on next page

Layout of the Instance editor

The Instance Editor window lists the parameters and their values of the selected instance.

In the **Value** column you can view and edit the value of the parameters.

When you click a row, the lower section of the Instance Editor window displays the type of parameter, restrictions for the parameter value and other conditions for the parameter.

18.5 FlexPendant Viewer

Overview

FlexPendant Viewer is an add-in to RobotStudio that retrieves and displays a screenshot from the FlexPendant. The screenshot is generated automatically at the moment of the request.

Prerequisites

The controller you want to retrieve screen shots from must be added to your robot view.

A FlexPendant must be connected to the controller. If no FlexPendant is currently connected (option *Hot plug* is installed and the jumper plug is used) then no screen shot can be retrieved.

Using FlexPendant Viewer

1. Make sure you are connected to the controller.
2. In the **Controller Tools** group, click the arrow next to the **FlexPendant** icon, and then click **FlexPendant Viewer**.

   A screen shot will be displayed in the workspace.
3. To reload the screen shot, click **Reload** in the workspace.
4. To set an automatic reload period for the screen shot, click on the menu **Tools**, point to **FlexPendant Viewer** and click **Configure**.

   Set the desired reload period and select the check-box **Activated**. Then click **OK**.

Results on the controller

The screenshot will automatically be saved as a file on the controller. When a new request is sent, a new screenshot is generated and saved, overwriting the previous file.

No message will be displayed on the FlexPendant.

18.6 Properties

Overview

Select this option to view and edit various properties of the connected robot controller.

Renaming the controller

The controller name is an identification of the controller that is independent of the system or the software running on the controller. Unlike the controller ID, the controller name does not have to be unique for each controller.

> **Note**
>
> The controller name must be written with characters from the ISO 8859-1 (Latin 1) character set.

1. In the **Configuration** group, click **Properties**, and then click **Rename**.

   The **Rename Controller** dialog box appears.
2. Enter the new name of the controller in the dialog box.
3. Click **OK**.

   The new name will be activated when the controller is restarted.

   You will be prompted to either click **Yes** to restart the controller immediately or click **No** to restart later.

Setting the controller date and time

You can either set the date and time to the same as the network time server, or you can specify the date and time manually.

Use this procedure to set the controller date and time:

1. In the **Configuration** group, click **Properties**, and then click **Date and Time**.

   The **Set Date and Time** dialog box appears.
2. This dialog provides two options: **Network Time** and **Manual Time**.
   - Select **Network Time** and in the **Time server address** box, enter the IP address of the network time server.
   - Select **Manual time** and then set the **Date** and **Time** in the boxes provided. You can select the required time zone in the **Time zone** list.

Setting the Controller ID

The Controller ID is by default set to the serial number of the controller. The Controller ID is a unique identifier for the controller and should not be changed. However, if the hard disk of the controller is replaced, the ID will be lost and you must set it back to the serial number of the controller.

> **Note**
>
> You must **Request Write Access** to the controller before setting the controller ID.

Continues on next page

1 In the **Configuration** group, click **Properties**, and then click **Controller ID**.

The **Set Controller ID** dialog box appears.

2 Enter the Controller ID and then click **OK**.

> **Note**
>
> Use only characters from the ISO 8859-1 (Latin 1) character set and no more than 40 characters.

Configuring the public network interface of the controller

> **Note**
>
> While connecting RobotStudio to the public network, IP address must be set only on the MGMT port for RobotWare 7.

1 On the **Controller** tab, in the **Configuration** group, click **Properties** and click **Network settings** and then click **Public Network**.

2 Select **Obtain an IP address automatically** to set the controller to receive the IP address from the network's DHCP server.

OR

Select **Use the following IP address** and then enter the required **IP address**, **Subnet Mask** and **Default Gateway** boxes to manually set the IP address of the controller.

3 Click the **Port Speed (Mbps)** drop-down to select the port speed, this selection is only available for RobotWare 7.1 onwards.

Port speed has three options, Auto, 10, 100. When the port speed is set to **Auto**, the network switch and the network interface automatically finds the highest available supported speed and adapts to this port speed. The adapted maximum port speed will be displayed.

4 It is possible to set the DNS server address for the controller, write access to the controller is required to use the following procedure.

Select **Automatically get DNS server address** to set the controller to receive the DNS server address automatically.

OR

Select the **Use the following server addresses** and then enter the **Preferred DNS server** and **Alternate DNS server** and click **Apply**.

Configuring the I/O network of the controller

The following procedure is applicable for RobotWare 7.4 onwards.

1 On the **Controller** tab, in the **Configuration** group, click **Properties** and click **Network settings** and then click **I/O Network**.

2 Enter suitable values as required in the **IP Address** and **Subnet Mask**.

The value of the **Default Gateway** is retrieved from the public network interface of the controller, hence this is a read-only field.

Continues on next page

3 Select the **Port Speed (Mbps)**, and click **Apply**.

The actual port speed on the network interface may differ from the port speed selected in the **Port Speed (Mbps)** if the network switch does not support the selected speed. When the port speed is set to **Auto**, the network switch and the network interface automatically finds the highest available supported speed and adapts to this port speed.

Viewing controller and system properties

You can view the following properties for a controller and its running system.

Controller Properties	System Properties
Boot Application	Control Module
Controller ID	Drive Module #1
Controller Name	Serial Number
Installed Systems	System Name
Network Connections	

1 In the **Configuration** group, click **Properties**, and then click **Controller and System Properties**.

The **Controller and System Properties** window appears.

2 In the tree view at the left of the window, browse to the node for which you want to view the properties.

The properties of the selected object are displayed in the Properties list to the right of the window.

Viewing the Device Browser

The Device Browser displays the properties and trends of the various hardware and software devices in a robot controller. To open the Device Browser, in the **Configuration** group, click **Properties**, and then click **Device Browser**.

Displaying the properties of a device

In the tree view, browse to the node for which you want to view the properties and then click it. The properties of the selected object, along with their corresponding values, are listed to the right of the tree view.

Updating the tree view

Press **F5**, to update the tree view.

Alternatively, right-click inside the tree view pane, and then click **Refresh**.

Displaying a trend

Select a device in the tree view and then double-click any property, that has a numerical value, in the right-hand panel. This opens a trend view. The trend view collects data at a rate of one sample per second.

Hiding, stopping, starting or clearing a trend

Right-click anywhere on the trend view and then click the required command.

Continues on next page

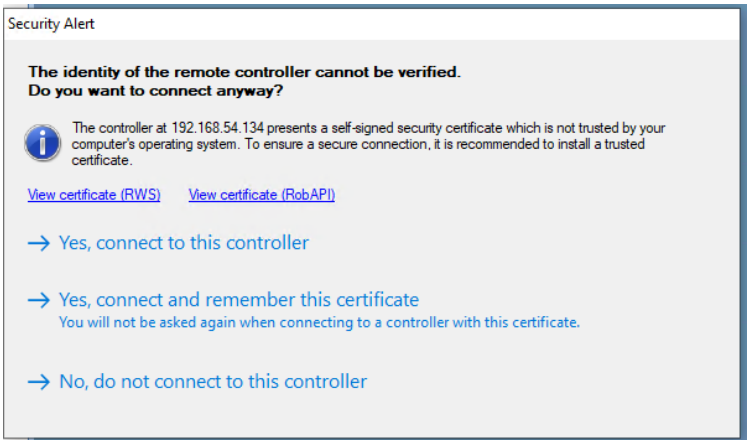
Saving system diagnostics

You can create a System Diagnostics data file from RobotStudio.

To save a System Diagnostics data file to your PC, in the **Configuration** group, click **Properties**, and then click **Save system diagnostics**.

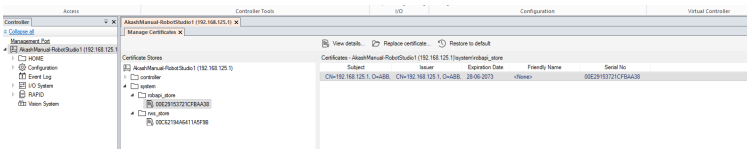
Manage Certificates

When connecting to an OmniCore controller, the communication is encrypted using certificates on the controller. By default, these are self-signed certificates that must be trusted by the RobotStudio user. In this case, the following dialog will be displayed.



xx1900001786

To ensure a trusted connection, it is recommended to replace the certificates of the controller with the certificates that are trusted by the PCs, which will be connected to the controller. RobotStudio supports only PEM (.pem) certificate. Refer *Operating manual - Integrator's guide OmniCore* for more information on certificate handling.

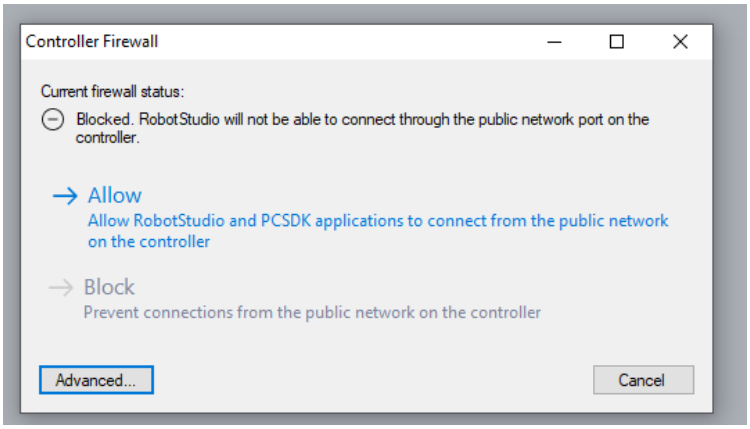


xx1900001787

Continues on next page

Configure firewall

Use this feature to set up the firewall to allow network connections. When you choose **Allow**, it configures the firewall to allow connections through the public network port on the controller. To prevent connections press **Block**. **Advanced** button opens the **Configuration editor** to manually edit the configuration file. By default, the firewall will be blocked. This must be configured to allow network connection through WAN.



xx1900001788

18.7 Installation

Updating an existing RobotWare system

Description

The most frequent RobotWare system update use case is updating one or more software, for example, RobotWare and add-ins. This is a frequent operation during the commissioning time, especially on large installations.

> **Note**
>
> To perform a RobotWare system update, the controller must be in the RobotWare system mode.

System update changes the configuration of the currently installed RobotWare system. There are different types of configuration changes, such as:

- Adding or removing licenses
- Upgrading, removing installed software or adding new software
- Activating or deactivating optional features

Before performing a system update, it is recommended to:

- create a backup of the system (user data) and store it on an external storage media.
- create a snapshot of the current system state.

Upgrading a software in the RobotWare system

The following procedure provides the steps involved during the update of the RobotWare system.

> **CAUTION**
>
> Do not turn off the controller while system update is in progress. Doing this may in worst case lead to data corruption in the RobotWare system, in which case it needs to be reinstalled.

1. Access the **Modify Installation** view in RobotStudio.
2. Select **Software** > **Included**.
3. The **Included Software** window displays the software that is included in the current RobotWare system.
4. Select the product that should be upgraded and tap **Update**.
5. In the **Update Software** window, select the software version to be used and tap **OK**.
6. The **Summary** tab shows an overview of all the changes.

Continues on next page

7 Continue to modify the system, or select **Apply**/**Apply and reset** to confirm and save the changes.

> **Note**
>
> The **Modify Installation** dialog will be closed during the controller update.
>
> When the update process is finished, check the event log for information about the update results. A successful update will be indicated in the event log, and if the update has failed, one or more error logs will be generated.

Adding/removing software

The following procedure provides the steps involved during the update of the RobotWare system.

> **CAUTION**
>
> Do not turn off the controller while system update is in progress. Doing this may in worst case lead to data corruption in the RobotWare system, in which case it needs to be reinstalled.

1 Access the **Modify Installation** view in RobotStudio.
2 Select **Software** > **Included**.
3 The **Included Software** window displays the software that is included in the current RobotWare system. Select one of the following:
   - Select the product box for the software that should be added to the system.
   - Deselect the product box to remove the product from the system.

   > **Note**
   >
   > Products may have dependences to certain versions of other products. A product may only be removed if all products that are dependent on it are removed as well.

4 The **Summary** tab shows an overview of all the changes.
5 Continue to modify the system, or select **Apply**/**Apply and reset** to confirm and save the changes.

> **Note**
>
> The **Modify Installation** dialog will be closed during the controller update.
>
> When the update process is finished, check the event log for information about the update results. A successful update will be indicated in the event log, and if the update has failed, one or more error logs will be generated.

Continues on next page

Adding/removing add-in packages

The following procedure provides the steps involved during the update of the RobotWare system.

> **CAUTION**
>
> Do not turn off the controller while system update is in progress. Doing this may in worst case lead to data corruption in the RobotWare system, in which case it needs to be reinstalled or recovered from a snapshot.

1. Access the **Modify Installation** view in RobotStudio.
2. Select one of the following:
   - To add add-in packages, select **Software** > **Available** and tap **Include**.
   - To remove add-in packages, select **Software** > **Included** and tap **Remove**.

     > **Note**
     >
     > Products may have dependences to certain versions of other products. A product may only be removed if all products that are dependent on it are removed as well.

     > **Note**
     >
     > RobotWare is mandatory and cannot be removed from the system.
3. The **Summary** tab shows an overview of all the changes.
4. Continue to modify the system, or select **Apply**/**Apply and reset** to confirm and save the changes.

   > **Note**
   >
   > The **Modify Installation** dialog will be closed during the controller update.
   >
   > When the update process is finished, check the event log for information about the update results. A successful update will be indicated in the event log, and if the update has failed, one or more error logs will be generated.

Changing the software installation order when adding/removing RobotWare add-ins

When adding and removing RobotWare add-ins to/from the system, sometimes it is necessary to manually adjust the installation and initialization order or the included add-ins.

1. Access the **Modify Installation** view in RobotStudio.
2. Select **Software** > **Included**.
3. In the **Included Software** window, tap the **Installation order** button to open the **Change Installation Order** window. Select a product and use the up and down arrows to change the installation order. Select **Done**.
4. The **Summary** tab indicates that the installation order has been updated.

Continues on next page

5 Continue to modify the system, or select **Apply**/**Apply and reset** to confirm and save the changes.

> **Note**
>
> The **Modify Installation** dialog will be closed during the controller update.
>
> When the update process is finished, check the event log for information about the update results. A successful update will be indicated in the event log, and if the update has failed, one or more error logs will be generated.

Working with option selections

Overview

The following categories of system features can be updated:

- System options
- Controllers
- Robots
- FlexPendant

> **Note**
>
> Some features extend, showing more options upon selection. For example, in the group controller variant, you get the option of choosing variant type only when a controller first is selected. The additional drive units work similarly, some are unavailable until you select a different drive system type. This means options can be locked behind selections.

Turning options on/off

> **CAUTION**
>
> Do not turn off the controller while system update is in progress. Doing this may in worst case lead to data corruption in the RobotWare system, in which case it needs to be reinstalled.

1 Access the **Modify Installation** view in RobotStudio.

2 Select the tab **Options**.

3 Select the option category to be updated, and the corresponding **Options** that should be activated/deactivated for the system.

> **Note**
>
> Linked options will be selected automatically.
>
> Conflicting options cannot be selected.

4 The **Summary** tab shows an overview of all the changes.

Continues on next page

5 Continue to modify the system, or select **Apply**/**Apply and reset** to confirm and save the changes.

> **Note**
>
> The **Modify Installation** dialog will be closed during the controller update.
>
> When the update process is finished, check the event log for information about the update results. A successful update will be indicated in the event log, and if the update has failed, one or more error logs will be generated.

Adding licenses to enable additional option access

> **CAUTION**
>
> Do not turn off the controller while system update is in progress. Doing this may in worst case lead to data corruption in the RobotWare system, in which case it needs to be reinstalled.

1 Access the **Modify Installation** view in RobotStudio.
2 Select the tab **Options**.
3 Select **Edit** to access the **Edit License files** window. Select one of the following:
   - Select **Add** to browse for a new license to be added.
   - Select an existing license and tap **Remove**.
4 The **Summary** tab shows an overview of all changes.
5 Continue to modify the system, or select **Apply**/**Apply and reset** to confirm and save the changes.

> **Note**
>
> The **Modify Installation** dialog will be closed during the controller update.
>
> When the update process is finished, check the event log for information about the update results. A successful update will be indicated in the event log, and if the update has failed, one or more error logs will be generated.

Exporting and importing option selections

> **CAUTION**
>
> Do not turn off the controller while system update is in progress. Doing this may in worst case lead to data corruption in the RobotWare system, in which case it needs to be reinstalled.

1 Access the **Modify Installation** view in RobotStudio.
2 Select the tab **Options**.

Continues on next page

3 Select one of the following:

- Select **Export** and browse to the location where the exported option selections should be saved. Select **Save**.

  The current option selections will be saved to an RSF file that can be imported or added to other systems.
- Select **Import** and browse to the location of the configuration file, and then select **Open**.

  Note

  All current selections will first be cleared.
- Select **Add** and browse to the location of the configuration file, and then select **Open**.

  Note

  Existing selections are kept, and options that are not currently selected will be added.

4 Continue to modify the system, or select **Apply**/**Apply and reset** to confirm and save the changes.

Note

The **Modify Installation** dialog will be closed during the controller update.

When the update process is finished, check the event log for information about the update results. A successful update will be indicated in the event log, and if the update has failed, one or more error logs will be generated.

Drive system types

The following matrix describes the existing drive system types and some examples of compatible products:

Product		Power									
Manipulator	**Controller**	2.5kVA-310V	2.5kVA-370V	3.0kVA-370V	7.0kVA-370V	3.0kVA-370V	480VA-24V	1.2kVA-48V	1.5kVA-48V	13kVA-650V	7.5kVA-650V
IRB 1600 or smaller	C30	A1									
	C90XT										
	E10		B2								
IRB 14050	C30						C6				
CRB 15000	C30							D7			
CRB 15000	C30								D10		
IRB 2600	V250XT				E4	E5					
	V400XT										

Continues on next page

Product		Power									
IRB 4600 or larger	V250XT									E8	E9
	V400XT										

Installing a new RobotWare system

Description

Before installing a new RobotWare system on the controller, it is required to:

- create a virtual controller.
- create an installation package.

Create a virtual controller

1 Start RobotStudio.
2 Select **Add Controller** > **Connect to Controller** in the **Controller** ribbon.
3 In the **Connect to Controller** window, select the **Virtual Controllers** tab.
4 Select **New Controller**.
5 In the **New Virtual Controller** dialog, select option **Create New** and complete the following:
   - **Name**

     Give the new system a valid name. If you enter an invalid name you will not be able to proceed.

     Note

     The system name can contain between 1 to 55 characters. Allowed characters are "A–Z", "a–z", "0–9", and "-" (hyphen). Hyphen "-" is only allowed between characters.

   - **Location**
   - **Robot model**
   - **RobotWare**
   - **Controller**

Note

Selecting option **Create from backup** can be used to create a system based on the configuration found in the selected Backup. This means that the same set of SW products (RobotWare and add-ins), licenses and options will be used.

Note, however that the software referred to by the Backup is not included in the Backup itself and must be previously downloaded to your computer by using the RobotStudio **Add-Ins** page.

Note also that this procedure will not automatically include RAPID programs and system parameters to your new system. If needed, they can be loaded to the new system by restoring the Backup once the new system is installed and started.

Continues on next page

6 Select **OK** to continue.

7 Continue with creating an installation package.

Creating a new installation package

Overview

The installation package is a software package that consists of predefined directory structure and number of files, used for purpose of re-deploying RobotWare System on a robot controller. The installation package is created in RobotStudio and is deployed on the controller using RobotWare Installation Utilities on the FlexPendant.

RobotWare Installation Utilities is a small package of installation related utilities that is always present on each robot controller and cannot be removed. It is used to deploy and re-deploy RobotWare system which is the operating system of the robot controller. When in RobotWare Installation Utilities mode, the robot cannot be moved using the FlexPendant and robot programs cannot be written or executed.

Prerequisites

The following prerequisites must be met before you can start creating an installation package:

- Latest version of RobotStudio must be installed.
- License files for products to be installed must be available. Licenses are included in the RobotWare system at purchase, but can also be retrieved from a backup of the RobotWare system currently deployed on the controller, or exported from the controller via RobotWare Installation Utilities.

  Note

  Virtual licenses can also be used.

- Product versions to be installed must be available in RobotStudio or in a custom location.

  These versions can be made available by selecting a RobotWare distribution package (.rspak file) from RobotStudio (tap **Install Package** in the **Add-Ins** tab). All products that are installed this way, have matched versions and correct dependencies to each other.
- A virtual controller must be created.

Create installation package

1 Start RobotStudio.

2 Select **Add Controller** > **Connect to Controller** in the **Controller** ribbon.

3 In the **Connect to Controller** window, select the controller and tap **OK**.

4 Request write access.

5 Launch the **Modify Installation** dialog from the **Controller** ribbon.

6 Select the tab **Software**.

Continues on next page

7 Select **Create Package** to create an installation package based on the virtual controller configuration.

> **Note**
>
> If the virtual system has been built using virtual licenses, these will not be included in the installation package.
>
> If virtual licenses are used, the selected feature configuration will be matched against the real licenses present in the controller and the installation will stop if some licenses are missing. This situation can be avoided if real licenses from the controller are exported and imported into the virtual system when it is built.

8 In the **Create Installation Package** dialog, define the following:

- **Package Name**

  Enter a name for the installation package.
- **Location**

  Browse and select the output folder (for example, a USB-stick) for the installation package.

Select **OK**.

9 The window **Installation Package created** is displayed. The installation package for the selected system has been created. Select **OK**.

10 Continue with installing the package on the controller.

18.8 Conveyor Tracking

Introduction

The Conveyor Tracking tab contains a tree view browser displaying the connected DSQC2000 units and other related objects. It is used as a user interface to configure settings and monitor live signals of the DSQC2000 tracking units.

Note

While in the tree view browser or any of the related object dialogues, pressing F1 will open the *Application manual - Conveyor tracking*.

To open the Conveyor Tracking tab:

	Action
1	Click **Conveyor Tracking** from the **Configuration** group, in the **Controller** tab. The **Conveyor Tracking** tab opens.

Connecting and adding a CTM

Connecting a CTM

One of the following methods can be used to connect the CTM to the computer:

1 **Connecting the CTM through a WAN port**

This method is recommended only after a fixed IP address is configured for the CTM - WAN interface.

Connect the computer with a fixed IP address to the same WAN network, through a network switch or a direct connection. Configure the computer's IP address with the CTM's subnet.

Example:

CTM - WAN interface	192.168.8.xx
Computer network interface	192.168.8.246

The default factory setting for WAN is **Obtain an IP address automatically** (DHCP client). To connect a WAN with this setting a DHCP server is required on the network.

2 **Connecting through a LAN port**

This method is recommended for the initial setup of a CTM with default factory settings. It can also be used for debugging, when there is a connection problem with the CTM.

Configure the fixed IP address of the computer with the subnet of the CTM – LAN interface. The LAN interface has IP address 192.168.126.200.

Example:

CTM - LAN interface	192.168.126.200
Computer network interface	192.168.126.246

Continues on next page

Adding a CTM

To add a CTM, a connection must be established between the computer and the CTM:

	Action
1	Click **Add CTM**. The **Add CTM** dialog box displays all the detected CTMs.
2	Select the CTM from the list and click **OK**. The CTM is added to the tree view object browser.

Identifying a CTM in RobotStudio

If RobotStudio is connected to a network with multiple CTMs, it may become difficult to identify a particular CTM. This can be avoided by a direct connection via LAN port.

To simplify identification, it is recommended to assign a unique and descriptive name for every CTM.

One of the following methods can also be used to identify a CTM:

- Select the device in the tree view browser, its corresponding green discovery LED will start blinking on the CTM module.
- Press the white SW1 button on the CTM module once, a green dot will start blinking on the corresponding CTM symbol in the tree view browser.

> **WARNING**
>
> Do not press the SW1 button while the CTM is restarting, this will reset the CTM to default factory settings and any updates or upgrades will be lost.

Network settings

The WAN port of the CTM is connected to the robot controllers through an Ethernet network. To establish communication between the robot controllers and the CTM, a fixed IP address must be assigned to the CTM. The IP address should be located on the same subnet as the network interface of the connected robot controllers.

To change the network settings:

	Action
1	Right-click the CTM in the tree view browser, select **Network settings**, and then click **WAN interface**.
2	Enter the **IP address**, the **Subnet mask**, and the **Default gateway** (optional) and then click **OK**.

Authenticate

A CTM has the following two predefined users:

User	Default password	Description
abbadmin	abbadmin	It is an advanced user and is used for advanced maintenance and troubleshooting.

Continues on next page

User	Default password	Description
ctmuser	ctmuser	It is a normal user and is used for everyday maintenance and troubleshooting.

Note

Login as an advanced user to upgrade the firmware.

To use some features of a CTM, a login is required. If you are using the default credentials, login is automatic to the Conveyor Tracking tab. Otherwise, authentication is requested whenever required.

Note

It is recommended to change the password to improve security.

Appropriate credentials are required to make any modifications to the CTM configuration and firmware.

WARNING

You cannot recover a lost password, so ensure the password is not lost or forgotten.

To login as a user:

	Action
1	Right-click the CTM and click **Authenticate** > **Login as a Different User**.
2	Enter the required credentials and click **Login**.

To log off the current user:

	Action
1	Right-click the CTM and click **Authenticate** > **Log Off**.

To change the password of the current user:

	Action
1	Right-click the CTM and click **Authenticate** > **Change password**.
2	Change the password and click **OK**.

Rename

You can rename the CTM, the encoders, and the sensors.

To rename any object:

	Action
1	Right-click the object in the tree view browser and then click **Rename**. The **Rename** dialog box opens.
2	Enter a new name for the object and then click **OK**. The new names are stored in the CTM and will be displayed in RobotStudio, once it is restarted.

Continues on next page

Configuring an encoder

You can configure some settings of an encoder, for example, Speed filter.

To configure an encoder:

	Action
1	Right-click the encoder in the tree view browser and then click **Configure encoder**. The **Configure encoder** tab opens.
2	Modify the settings and click **Apply**.

An encoder can be configured as used or not used. This change in configuration will affect how the encoder is displayed in the tree view browser. To tune the encoder speed filtering, you can define the **Low-pass filter cut off frequency** in the **Speed filter** section.

Configuring a sensor

You can configure some settings of a sensor, for example, Sync Separation.

To configure a sensor:

	Action
1	Right-click the sensor in the tree view browser and then click **Configure sensor**. The **Configure sensor** dialog box opens.
2	Modify the settings and click **Apply**.

Sync separation is used to filter an unstable signal in the sync input signal. It defines the minimum encoder distance (counts) between two sync pulses from the sensor. When the actual distance is shorter than this value, then the second sync pulse is ignored.

The sensor type can be configured as **I/O Sensor**, **Camera**, or **Not used**. This change in configuration will affect how the sensor is displayed in the tree view browser.

Note

It is recommended to set the same sync separation value for all the encoders.

The camera pulse width defines the pulse length (ms) used with the camera trigger signal.

Note

For high speed conveyors, the pulse length must be shorter than the time between consecutive camera images, so that:

Camera pulse width < (Trigger distance in mm / Max conveyor speed in mm/s) / 1000

Restart

Overview

A restart or reboot of a CTM can be performed using the tree view browser.

Continues on next page

Restart

A restart is required after modifying the settings of the CTM.

To restart the CTM:

	Action
1	Right-click the desired CTM in the tree view browser and then click **Restart**.

Reboot

A reboot is slower than a restart, it corresponds to the restart after cycling the power.

To reboot the CTM:

	Action
1	Right-click the desired CTM in the tree view browser and then click **Reboot**.

Signals

There are 3 types of live signals: encoder, sensor, and other. You can sort and filter the signals by various properties:

Signal property	Description
Name	Name of the signal
Type	Signal value type, for example, float
Value	Signal live value
Unit	Signal value unit, for example, Hz
Category	Signal category, for example, Public or Internal
Sensor	Sensor number (1-8)
Encoder	Encoder number (1-4)
Function	Signal name used in the robot controller, for example, *TrigVis*
Label	Connector on the CTM, for example, X11
Description	Information on what the signal represents

List all signals

To list all the signals:

	Action
1	Right-click the desired CTM in the tree view browser and then click **Signals**.

List all encoder signals

To list all the encoder signals:

	Action
1	Right-click the desired encoder in the tree view browser and then click **Signals**.

List all sensor signals

To list all the sensor signals:

	Action
1	Right-click the desired sensor in the tree view browser and then click **Signals**.

Continues on next page

Backup and Restore

You can save a backup of the CTM settings. The backup will contain:

- Data about the CTM configurations, including encoder and sensor parameters and new names. While restoring the backup, these settings are applied to the target CTM
- Network settings
- Firmware version information

Creating a backup of the CTM

Keeping a backup is recommended to simplify the task of replacing the CTM with a new unit.

To create a backup of the CTM:

	Action
1	Right-click the CTM in the tree view browser and then select **Create Backup…** . The **Create Backup from CTM** dialog box opens.
2	Enter a **Backup Name**.
3	Enter or select a save **Location** and click **OK**. A backup of the CTM is created in the entered location.

Restoring a backup

To restore a backup of the CTM:

	Action
1	Right-click the CTM in the tree view browser and then select **Restore Backup….** The **Restore from Backup to CTM** dialog box opens.
2	Enter or browse to the required **Location**, select backup folder and click **Open**. The backup of the CTM saved in the folder is displayed in the list of the **Available backups** section.
3	Select the required backup from the list and click **OK**. The CTM's settings will be restored according to the backup, after a restart of the CTM.

Firmware upgrade

A firmware upgrade is normally provided by ABB, as a .cab file.

WARNING

It is recommended to backup the CTM configurations, as a firmware upgrade may result in the factory default network settings.

To upgrade the firmware of the CTM:

	Action
1	Login as an advanced user to upgrade the firmware.
2	Right-click the CTM and then click **Firmware Upgrade**. The **Firmware Upgrade** dialog box opens. It displays the current version of the firmware.

Continues on next page

	Action
3	Click **Browse**, navigate to the .cab file and then click **Open**. The **Verify Software** dialog box opens. It displays the publisher of the file.
4	If the publisher is trusted, click **Yes**. The **Firmware Upgrade** dialog box opens. It displays the new firmware version.
5	Verify it and click **Upgrade**, to download it. The CTM will restart after the download and installation.

After a firmware upgrade, you may have to restore the network settings of the CTM.

18.9 Motion Configuration

Overview

The **Motion Configuration** window contains functions for making and viewing advanced system configurations, such as changing controller and baseframe positions, calibrating and setting up external axes.

> **CAUTION**
>
> Editing the system may result in corrupted systems or unexpected robot behaviors. Be sure to understand the effects of the changes before proceeding.

The mechanism folder node

The property page of this node contains controls for mapping and setting axis and joints. It is from this page you set up external axes.

The mechanism library node

The property page of this node contains controls for changing the *baseframe* of the robot or mechanism. Here, too, you specify whether the baseframe is moved by another *mechanism* (coordinated motion), like a track *external axis*.

Updating the baseframe position

1. Move the *mechanical unit* (robot or external axis) to its new location using the ordinary tools for moving and placing objects.
2. In the **Controller** browser, select the controller for the mechanical unit.
3. On the **Controller** tab, in the **Virtual Controller** group, click **Motion Configuration**.

   This opens the **Motion Configuration** dialog.
4. Select the node for the mechanical unit in the hierarchical tree. The baseframe property sheet for the robot is now displayed.
5. Select the baseframe position values to use after restarting the robot.

Select	To
Controller Values	Reset all changes to the baseframe made since the last time the *virtual controller* was started.
Stored Station Values	Reset all changes made to the baseframe since the last time the station was saved. Optionally, you can enter new values in the baseframe coordinate boxes (relative to the controller world coordinate system).
Use Current Station Values	Read and use the current location of the baseframe. Optionally, you can enter new values in the baseframe coordinate boxes (relative to the controller world coordinate system).

6. Click **OK**.

Continues on next page

Calibrating the baseframe position

The station with a virtual controller that includes a workpiece *positioner* which is based on backup from a robot controller. RobotStudio needs to take the calibration of the workpiece positioner into account, otherwise the positioner will be misplaced. The calibration parameter for a mechanism can be used to take the calibration of the real positioner into account.

The positioner mechanism is placed according to the *base frame* values of the virtual controller. When a virtual controller is created from backup, in which, the positioner was calibrated online using four points method, and if the positioner was not in its sync position at the first calibration point, then the positioner mechanism will not be aligned with the system's *task frame*.

You can calibrate base frame values to realign the positioner mechanism with the system's task frame.

1. In the **controller** tab, click **Motion Configuration**. The **Motion Configuration** dialog opens.
2. In the **Motion Configuration** window, select the positioner mechanism and then click **Calibrate**.

   The **Base Frame Calibration Position** dialog opens.
3. From the **Motion Configuration** dialog, copy the **Orientation** values under **Base Frame** and paste these values in the boxes under **Orientation** in the **Base Frame Calibration Position** dialog.
4. Click **Apply** and then click **OK**. You must restart the controller when prompted. The positioner mechanism will be aligned with the system's task frame now.

   Any object attached to the positioner will take the orientation of the positioner.

18.10 Task Frames

Modifying Task frame

1. On the **Controller** tab, in the **Virtual Controller** group, click **Task Frames.**

   The **Modify Task Frames** dialog box appears.
2. Set the reference to **World**, **UCS**, or **Local**.
3. Edit the position and orientation of task frames in the **Task Frames** coordinate box.
4. Click **Apply.**

   To the question, *Do you also want to move the Base Frame(s)?*
   - Click **Yes** to move the base frame, but keeps its relative placement to the *task frame*.

     The related robot will move in relation with the station world, but since the robot moves with the task frame, the robot base frame values remain unchanged.
   - Click **No**. The following question appears **Do you want to update the controller configuration and restart?**. Click **Yes** to restart the controller and update the base frame configuration of the connected virtual controller.

     The related robot will not move in relation with the station world. Here the task frame has moved, hence the base frame values will be recalculated. The controller must be restarted for the configuration changes to take place.

> **Note**
>
> If there are any stationary RAPID objects (tooldata, workobjects) connected to the robot, the following question appears **Do you want to keep the positioning of all stationary RAPID objects?**
>
> - Click **Yes** to keep all the stationary RAPID objects in their global coordinates.
>
>   Stationary workobject and *tooldata* will remain constant in relation to the station world but their values will be recalculated.
> - Click **No** to move all the stationary RAPID objects along with the base frame (same coordinates relative to base frame).
>
>   Stationary workobject and tooldata will move with the robot and its values will not be recalculated.

18.11 Go Offline

Overview

The main purpose of this feature is to create a new station with a *VC* similar to the connected *robot controller*. This helps a robot technician to work offline even when the robot controller is disconnected.

Using Go Offline

1. Connect the PC to a robot controller.
2. On the **Controller** tab, click **Request Write Access.**
3. Click **Go Offline.**

   The **Go Offline** dialog box is displayed.
4. Enter a name in the **Virtual Controller Name** field and browse for the location to save the system.
5. Select the *RobotWare* version followed by the RobotWare Add-in version and click **OK**.

   A new station gets created with a virtual controller which has the same configuration as the robot controller.

Note

A pre-requisite is that any RobotWare add-ins used by the system must be available on the PC. A Relation is automatically created between the virtual controller and the robot controller.

18.12 Create Relation

Overview

The transfer function allows easy transfer of offline-created RAPID programs to the real robot on the shop floor. This means that you can transfer data from a *virtual controller* (which is offline) to a *robot controller* (which is online). As part of the transfer function you can also compare the data present in the virtual controller with that present in the robot controller and then select which data to transfer.

You can also use the transfer function to transfer data from a virtual controller to another virtual controller.

Relations for transfer of data

To transfer data, you must first set up a **Relation** between the two controllers. A Relation defines the rules for the transfer of data between the two controllers.

Creating a Relation

When you have two controllers listed in the Controller browser, you can create a Relation between them. To create a Relation:

1. On the **Controller** tab, in the **Transfer** group, click **Create Relation**.

   The Create Relation dialog box is displayed.
2. Enter a **Relation Name** for the relation.
3. Specify the **First Controller**, from the list.

   The First Controller, also called the Source, owns the data being transferred.
4. Specify the **Second Controller**, from the list. This can either be a robot controller or another virtual controller.

   The Second Controller, also called the Target, receives the data being transferred.
5. Click **OK**.

   The relation between the controllers is now created.

After this, the *Relation* dialog box opens, using which you can configure and execute the transfer. Relations of a controller are listed under its Relations node in the Controller browser.

> **Note**
>
> The properties of the relation are saved in a XML file under INTERNAL in the owner controller's system folder.

Transferring data

You can configure the details of the transfer of data and also execute the transfer, in the *Relation* dialog box.

Continues on next page

18.12 Create Relation
Continued

To open the *Relation* dialog box, double-click a relation. Alternatively, select a relation in the **Controller** browser, and then in the **Transfer** group, click **Open Relation**.

Configuring the transfer

Before executing a transfer, you can configure the data to be transferred, under the *Transfer Configuration* heading. Configure using these guidelines:

- Use the check boxes in the *Included* column to include or exclude the corresponding items shown in the tree structure. All items in a module that are included will be transferred. Other non-listed items of a module such as comments, records and so on will be automatically included in the transfer.
- The *Action* column shows a preview of the transfer's result, based on the items you include or exclude.
- If a module exists both in the source and the target controllers, and the *Action* column shows *Update*, then click **Compare** in the Analyze column. This opens the *Compare* box which shows two versions of the module in different panes. The affected lines are highlighted and you can also step through the changes. You can choose one of the following options for the comparison:
  - **Source with target** - Compares the source module with the target module
  - **Source with result** - Compares the source module with the module that will be the result of the transfer operation
- BASE (module), wobjdata and tooldata are excluded by default.
- wobjdata wobj0, tooldata tool0, and loaddata load0 of the BASE module are unavailable for inclusion.

A task can be transferred only if:

- Write access to the target controller is present (must be manually retrieved).
- Tasks are not running.
- Program execution is in the stopped state.

Executing the transfer

After you have configured the transfer, you can execute it.

Under the *Transfer* heading, the Source and Target modules are shown along with the arrow showing the direction of the transfer. You can change the direction of the transfer by clicking **Change Direction**. This also switches the source and target modules.

To execute the transfer click **Transfer now**. A dialog showing a summary of the transfer appears. Click **Yes** to complete the transfer. The result of the transfer is displayed for each module in the output window.

The **Transfer now** button is disabled if:

- None of the included tasks can be transferred.
- Write access is required but not held.

18.13 Online monitor

Overview

This feature allows you to remotely monitor the robot connected to a real controller. It displays a 3D layout of the connected robot controller and enhances user's current perception of reality by adding motion visualization augmentation.

Note

The Online Monitor shows TCP robots and TCP robots with track. When connecting the Online Monitor to a virtual controller, the motion is shown only if the virtual controller is using Free-Run mode, not the Time Slice mode.

Using Online Monitor

The following procedure describes the Online Monitor feature in RobotStudio:

1 Connect the PC to a controller and add the controller.

2 Click **Online Monitor**.

The 3D view of the mechanical units of the controller system is displayed in the graphics window.

Note

The robot view is refreshed every second with the current joint values of all the mechanical units.

Gravity parameters in online monitor

Online Monitor displays the orientation of a robot according to its gravity parameters. The Gravity Alpha, Gravity Beta and Gamma Rotation parameters define the rotation of the robot around the X, Y and Z axis in the world coordinate system. The Online Monitor orients the robot in the Graphical view according to the gravity parameters.

These parameters describe how the robot is oriented in relation with the floor or ground, whether it is suspended (mounted on the ceiling), shelf mounted (mounted on the wall), or mounted on the regular floor. If the robot is configured to be mounted on the ceiling, it will be displayed upside down in the Online Monitor. You can set these parameters in the **Motion configuration file**.

For more information about gravity parameters, see *Technical reference manual - System parameters*

Indication of TCP

A cone is automatically created to indicate the active tool data being used. The cone has its base in the robot wrist and its tip at the location of the tool data.

Kinematic Limitations

When the Kinematic Limitation button is enabled, the graphical 3D viewer indicates whether the robot is at a joint limit or at a singularity.

Continues on next page

For joint limits, the corresponding link is highlighted in yellow to indicate a warning and in red to indicate an error. The tolerance limits are defined in RobotStudio Options - Online - Online Monitor.

For singularity, a markup indicates when the axis 5 is close to singularity. The singularity level is also defined in RobotStudio Options.

Visualizing Safety Zones in Online Monitor

This feature lets you to visualize the current status of the manipulators in a robot system and provides an augmented reality of the robot cell. It allows you to visualize a failure scenario, for example, an unplanned stopping of a robot. To give the user an idea of the physical layout and the safety zone that has caused this stopping of the robot, safety zones are visualized in online monitor. When the robot enters a restricted zone, the safemove supervision feature stops the robot.

Features

- A **Show Safety Zones** button is available in the Online Monitor for each manipulator in the system, for example, four buttons in a MultiMove system with four manipulators.
- The name of each tool zone and the corresponding manipulator is shown as a markup, for example, Rob1 STZ1, …, Rob4 STZ8, Rob1 MTZ1, …, Rob4 MTZ8 and so on.
- Zones that are defined as **Allow inside** are visualized as a green semitransparent hollow shape.
- Zones that are defined as **Allow outside** are visualized as a red semitransparent solid shape.
- A message is displayed in the output window if no STZ or MTZ is defined for the manipulator.
- The controller event log message **20468 SC STZ violation** is displayed in the output window if it is present in the controller event log.

> **Note**
>
> You can open one SafeMove Configurator at a time, even though several controllers may be connected. If the SafeMove Configurator is opened for one controller, real or virtual, the icon gets disabled for the other controllers.

19 RAPID tab

19.1 Synchronize

Overview

To synchronize is to make sure that the RAPID program in the *virtual controller* corresponds to the program in the station. You can synchronize from the station to the virtual controller and vice versa.

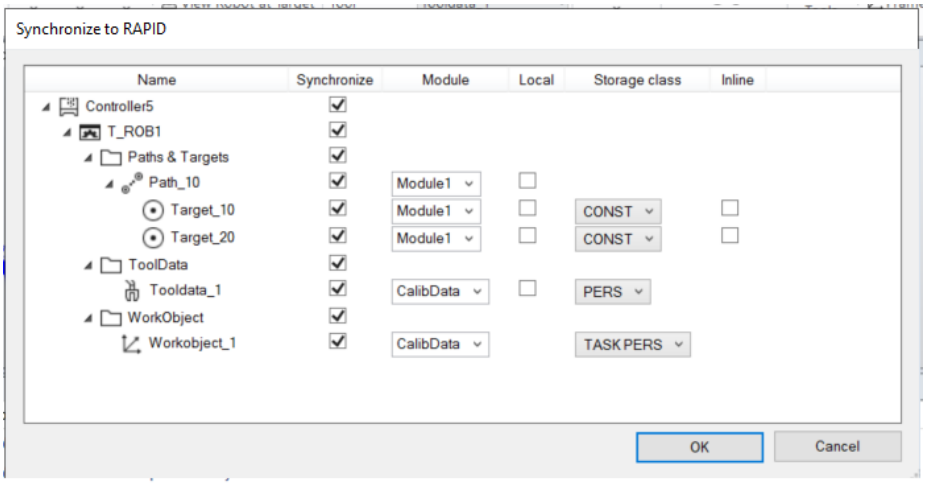
In a station, a robot's position corresponds to the targets and its movements are defined by move instructions in the robot path. These correspond to data declarations and instructions in the modules of the RAPID program.

Synchronizing to RAPID

This operation updates the RAPID program of the virtual controller to reflect the latest changes in the station. Perform this operation before running a simulation, saving a program to file and before taking a backup of a virtual controller.

1 To synchronize, click the arrow next to the **Synchronize** icon, and then click **Synchronize to RAPID**.

2 Select the elements to be synchronized from the list.



xx1900001535

Element	Description
Module	Specifies the target module of the virtual controller, if it does not exist , it will be created. Select from the list of modules or type in the name of the new module.
Local	Specifies whether the data must be created as **LOCAL**. Adds the keywork **LOCAL** before the data declaration and creates data **LOCAL** to the module.
Storage class	Specifies the storage type of the data declaration.
Inline	Specifies whether the data declaration must be declared as inline or named. In this case the data is declared in the instruction itself.

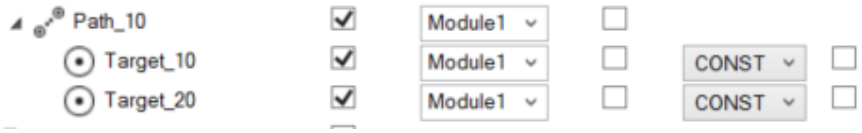
3 Click **OK**.

Continues on next page

The message **Synchronization to RAPID completed** is displayed in the Output window.

Note

To see the properties of targets , the containing path must be expanded.



xx1900001536

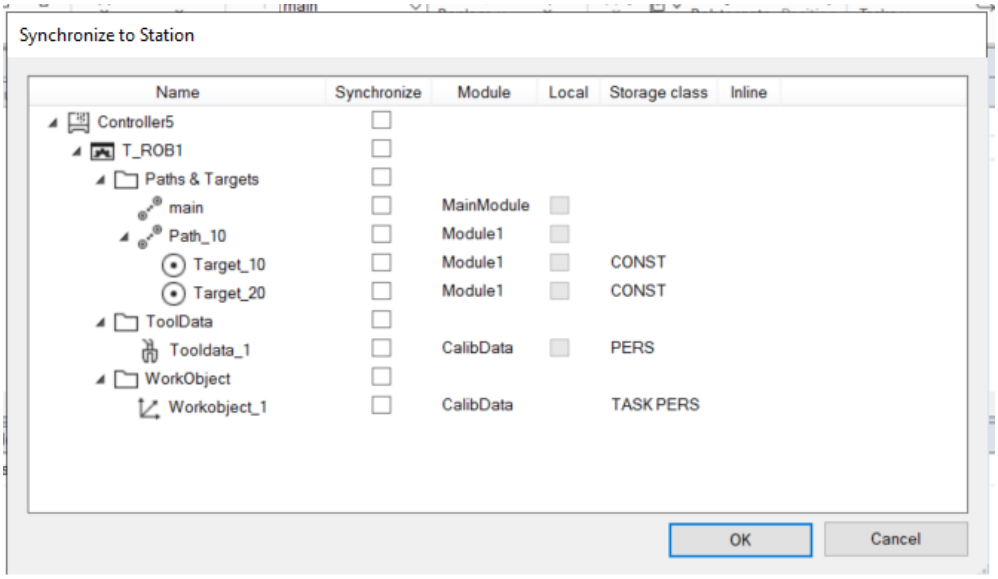
Any reference to workobjects or tooldata from an instruction will be synchronized even if they are not selected.

Synchronizing to the station

This operation updates the station to reflect the latest changes of the RAPID code in the virtual controller. Perform this operation when you start a virtual controller with RAPID program, when you have loaded a program or module from a file, or after editing a program.

To synchronize to station, click the arrow next to the Synchronize icon, and then click Synchronize to Station.

Select the objects to be synchronized to the station from the list. It is not possible to change the properties in this step. For changing any of the following properties the RAPID code must be edited.



xx1900001534

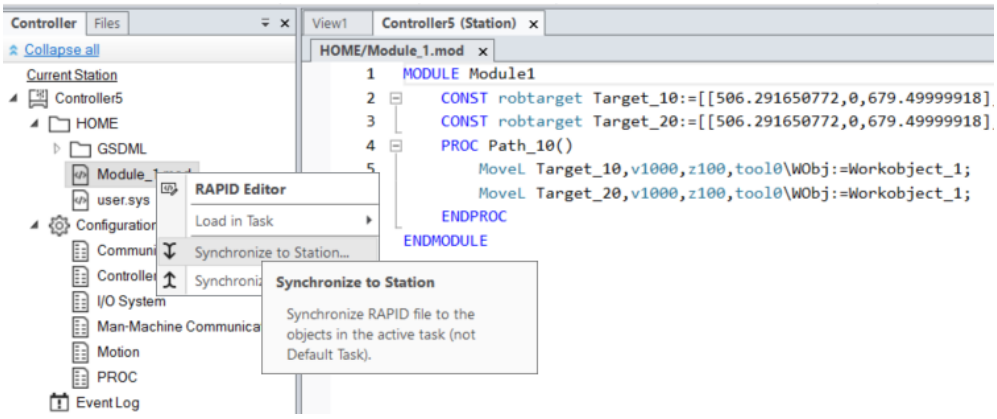
Click **OK**. The message Synchronization to Station completed is displayed in the Output window.

Continues on next page

Synchronize to and from files

RAPID can be synchronized from the files of the **HOME** folder as an alternative. This option is available **only** in the context menu of the selected file.

- In the **Controller** browser, under **Home** folder, right-click any RAPID module and then click **Synchronize to Station** to synchronize the selected file to the station.
- In the **Controller** browser, under **Home** folder, right-click a RAPID module and then click **Synchronize to File** to match the station into the selected file.



xx1900001533

Limitations

- Robtargets that are local to a procedure are not supported by **Synchronize to Station**. Only robtargets that are local to a module are supported.
- RobotStudio does not fully support instructions using *Offs* or *RelTool* functions. These are synchronized and will appear in the element browser, but commands such as *View Tool at Target* and *Locate Target* will not work. Targets used in the instructions will not be visible in graphics. However, instructions can be programmed and edited using the RAPID Editor and can be simulated using the virtual controller.
- RobotStudio does not support RAPID programs containing arrays of tooldata, robtargets and workobjects. These programs will not be synchronized to the station.
- Workobjects and tooldata that are shared between several tasks must be specified in RAPID with its full value for each task when programming offline with RobotStudio. This will trigger a warning *Initial value for PERS not updated* in the controller event log. You may ignore this warning. However, you must carefully ensure that the RAPID variable definitions are the same in all tasks, otherwise you may get unexpected behavior.

19.2 Adjust Robtargets

Overview

The Adjust Robtargets feature helps in recalculating and changing the robtarget data (tooldata and workobject data) while maintaining the joint angles of the robot. The robtarget data related to the specified source tooldata and workobject will be adjusted for usage with the new tooldata and workobject. In the **RAPID** tab, in the **Controller** group, click **Adjust Robtargets** to access this feature.

Prerequisites

- You should have a controller (virtual or robot) running with one or more modules containing procedures with a sequence of move instructions expressed with a defined tool and workobject.
- You should have RobotStudio Premium license to use this feature.
- The **Execute** button of the feature Adjust Robtargets will be enabled only if the selected tool data or workobject data have the same properties, such as robhold, ufprog, ufmec and so on.

> **Note**
>
> Arrays, event records, and offsets are not supported. Relative tool is also not supported. Circular move instruction (MoveC) is supported.

Using Adjust Robtargets

> **Note**
>
> Take a backup copy of your modules before adjusting your robtargets.

The following procedure describes the Adjust Robtargets feature in RobotStudio:

1. On the **RAPID** tab, in the **Controller** browser, select a RAPID task or module under the RAPID icon. Then click **Adjust Robtargets** on the **RAPID** tab.

   Alternatively, right-click the RAPID task or module in the **Controller** browser, and then click **Adjust Robtargets** in the context menu.

   The Robtarget Adjust dialog box appears.

   > **Note**
   >
   > You can access **Adjust Robtargets** from the **Controller** tab also. Right-click the RAPID task or module in the **Controller** browser and then click **Adjust Robtargets** in the context menu.

2. If the module you want to adjust is selected, then go to Step 4. Otherwise continue with the next step.

Continues on next page

3 Select a task from the **Task** drop-down list and module from the **Module** drop-down list.

> **Note**
>
> In the **Module** drop-down list, you can either select a particular module or **<ALL>** to update.

4 Select the source robtarget data (that is, the data defined in the selected task) from **Old tooldata** and **Old wobjdata** drop-down list.

5 Select the destination robtarget data (that is, new tooldata and workobject) from **New tooldata** and **New wobjdata** drop-down list.

6 Click **Execute**.

The **Execute** button is enabled only if source robtarget data (that is, old tooldata and workobject) and destination robtarget data (that is, new tooldata and workobject) are different.

The module searches for move instructions that use the specified old tooldata or workobject and recalculates the robtarget data for the new tooldata and workobject.

For example,

1 Select "tool0" as the source tool and "wobj0" as the source workobject.

2 Select "toolb" as the new tool and "wobjb"as the new workobject.

3 Click Execute.

Robtargets of "tool0" and "wobj0" will be replaced with re-calculated robtargets which correspond to the same robot configuration (all joint angles will be the same), and with the new tool "toolb" and "wobjb". Note that both the tooldata and the wobjdata are replaced independently.

Update instruction

By default, the **Update instruction** check box is selected. This means that move instructions using the specified source (old) tooldata and workobject will be updated to use the target (new) tooldata and workobject in addition to recalculating the robtargets.

If the **Update instruction** check box is cleared, the robtargets will be recalculated, but the move instructions will not be updated. They will still use the source tooldata and workobject.

This feature is useful after the calibration of tooldata and workobject. After calibration, you might still want to use the old names of the tooldata and workobject, but update their values and recalculate the robtargets accordingly. The following sample procedure illustrates how this can be accomplished.

Sample procedure

Prerequisite: RAPID module with robtargets and move instructions that use uncalibrated tooldata tool1, and workobject wobj1.

1 Calibrate your tooldata tool1, and workobject wobj1. Store the new values in tool1_calib and wobj1_calib, respectively. Keep the old values of the

Continues on next page

uncalibrated tooldata and workobject in tool1, and wobj1. An error message gets displayed for an invalid selection of tooldata or workobject data.

2 Open the Adjust robtargets tool and clear the **Update instruction** check box. Select your RAPID module, enter tool1, and wobj1 as your old tooldata and wobjdata, and tool1_calib, and wobj1_calib as the new tooldata and wobjdata, respectively.

3 Click **Execute**, and apply the changes to the controller from the RAPID Editor

4 In the RAPID Editor, rename your tooldata tool1 to tool1_uncalib, and tool1_calib to tool1 and apply the changes to the controller. Also, perform the same for wobj1.

Now, your robtargets are updated to match the calibrated values of tool1 and wobj1.

Limitations

- If a robtarget is used more than once but with different tools or workobjects, then a message *Target is referenced* is displayed in the output window.
- The adjust robtargets function operates on a module level and does not update any referenced targets defined in other modules. It ignores the scope of the robtargets when the referenced targets are local to a procedure. In this case, any targets with the same name in the module scope will also be updated.
- Adjust Robtargets function works for modules with semantic errors, the editor excludes the particular line containing the error and continues with program execution. But Syntax errors stop program execution.

20 Add-Ins tab

20.1 Gearbox Heat Estimation

Overview

The Gearbox Heat Estimation tool helps to estimate heat problems in the gearboxes. When the temperature is above a predefined value, you can adjust the cycle to reduce the temperature or order a fan that can cool down the gear.

Robots with compact gearboxes have a risk of getting overheated under certain circumstances. The gearbox temperature is supervised by *Service Information System* (SIS). SIS is a software function within the robot controller, that simplifies its maintenance. It supervises the operating time and mode of the robot, and alerts the operator when a maintenance activity must be scheduled. It also supervises large robots from damaging the motors during high load operations with a safety shutdown.

The temperature supervision is based on an algorithm that estimates the stationary temperature of the gearboxes and motors of the robot. The algorithm estimates the heat based on the character of the robot motion and also the room temperature. Intensive motion (high average speed and/or high average torque and/or short wait time) will increase the heat of the gearbox and motors.

To avoid overheating, SIS stops the robot if the temperature becomes too high. For large robots, there is an option to add a cooling fan to axis 1, 2, and sometimes axis 3, to allow the robot to run even with a heavy duty program.

Note

Gearbox Heat Estimation is not supported for Tool and External axis. When a *virtual controller* has more than one robot, only one robot will have estimations calculated. The other robots will only display 0% chance of overheating.

Prerequisites

1. RobotStudio 5.14.02 or later.
2. RobotWare 5.14.01 or later.
3. RobotStudio station with controller having a programmed cycle that includes payload for the robot.

Estimating the gearbox heat

Use the following procedure for estimating the heat generated by the robot:

1. Create a new station or open a saved station. The Gearbox heat button is now visible in the **Add-Ins** tab.
2. In the **Add-Ins** tab, click Gearbox Heat. The **Gearbox Heat Estimation** window opens.

Continues on next page

3 In the **Add-Ins** tab, select **Enabled** to enable the **Gearbox Heat Estimation** tool.

> **Note**
>
> For a manipulator without compact gear, **Gearbox Heat Estimation** is disabled.

4 Run a simulation.

> **Note**
>
> For RobotStudio Basic, the **Play** button in the **Simulation** tab will be disabled. As such, you will be unable to run the simulation from the **Simulation** tab. In such a scenario, use the **Play** button which will now be visible in the **Gearbox Heat Estimation** tab window to run the simulation.

> **Note**
>
> The data is recorded during the simulation only if the **Gearbox Heat** tab is visible.

5 In Cycles, define the behavior of the cycle for estimating the heat generated by the robot:

- **Continuous**: Select this option to calculate the estimations continuously, that is, without waiting time between two consecutive cycles.
- **Number of cycles per hour**: Select this option to manually define the number of cycles per hour used for calculation.
- **Waiting time between cycles (sec)**: Select this option to define the waiting between cycles. Enter the waiting time in seconds.

6 In **Ambient Temperature**, define the ambient temperature.

- Use the slider to change the temperature.
- Select **Use temperature from controller(s)** to reset the the ambient temperature and read the temperature from the robot configuration as specified in the parameter *Motion->SIS Parameter -> r1_sis_param->Robot temperature*.

  > **Note**
  >
  > The temperature value used while configuring the robot in the real environment must be used for ambient temperature calculations.

7 Calculate the result in either of the following ways:

- In the **Recordings** section, either double-click a recording or select a recording and click **Calculate**.

Continues on next page

- In the **System** section, either double-click a controller or select a controller and click **Calculate**.

Note

- The **Recordings** section displays the recordings to be analyzed when **Gearbox Heat Estimation** is enabled.

The results are displayed for each joint and with fans for the joints that can have fans installed as an option.

Note

The following factors influence the heat accumulated:

- Axis speed
- Payload
- Room temperature (ambient temperature)
- Waiting time (to allow robot to cool down)

Note

The calculated energy is displayed as different heat levels:

- **Green**: Indicates no heat problem
- **Orange**: Indicates it is recommended to install a fan.
- **Red**: Indicates a fan must be installed.
- **Grey**: Indicates it is not possible to calculate the possible energy level for this joint.
- **Not available**: Indicates the joints that cannot have a fan installed.

Note

Recommended action is displayed along with the warning level for each joint.

- **Joint**: Represents the joint.
- **Without fan**: Displays the percentage of heat levels calculated to the corresponding joint without fan.
- **With fan**: Displays the percentage of heat levels calculated to the corresponding joint with fan.
- **Action**: Displays the recommended action.

This page is intentionally left blank

A Options

Common buttons

Apply	Click this button to save all options in the current page.
Reset	Click this button to reset to the settings you had before this session all values that you have changed on the current page.
Default	Click this button to reset to their default values all settings on the current page.

Options:General:Appearance

Select application language	Select the language to be used. RobotStudio is available in the following seven languages: English, French, German, Spanish, Italian, Japanese, and Chinese (simplified).
Select color theme	Select the color to be used.
Default scale for zoomable windows	Defines the default scale to use for windows that are zoomable, for example, RAPID Editor, RAPID Data Editor and Configuration Editor.
Show ScreenTips	Select this check box to view ScreenTips.
Display Position Edit boxes with Red/Green/Blue background	Select the check box if you want to display the position boxes in the modify dialog boxes with colored background. Default value: selected.
Group related document windows under one tab	Select this check box to group related document window under one tab. Modifying this option requires a restart for the changes to take effect.
Restore hidden dialogs and messages	Select this check box to restore dialogs or messages which you may have hidden while using RobotStudio.

Options:General:Licensing

Disable licensing	Reverts to Basic mode to use features that do not require activation.
View installed licenses	Click to view the licenses listed by feature, version, type, expiration date and status.
Activation Wizard	Click to activate RobotStudio license.
RobotStudio user experience program • I want to help improve RobotStudio • I do not want to participate right now	For RobotStudio Basic users, it is mandatory to participate in the user experience report. For RobotStudio Premium users, you can choose whether or not to participate in the user experience report .

Options:General:Units

Quantity	Select the quantity for which you want to change the units.
Unit	Select the unit for the quantity.
Display decimals	Enter the number of decimals that you want to be displayed.
Edit decimals	Enter the number of decimals that you want when modifying.

Continues on next page

Continued

Default orientation format • RPY angles (Euler ZYX) • Quaternions	Specifies the default format to use for orientations.

Options:General:Advanced

Number of undo/redo steps	The number of operations that can be undone or redone. Lowering this value can decrease memory usage.
Warn about running Virtual Controller processes on startup	Warns of orphaned VC processes.
Show acknowledge dialog box when deleting objects	Warns when deleting objects.
Show acknowledge dialog box when deleting targets and corresponding move instructions	Warns when deleting targets and move instructions.
Bring the output window to front if an error message is displayed	Select this check box to bring the output window to front if an error message is displayed

Options:General:Autosave

Enable autosave of RAPID	This check-box is selected by default. RAPID programs are saved automatically in every 30 seconds.
Enable autosave of station	Unsaved stations are saved automatically at the interval specified in the **minute interval** box.

Options:General:Files & Folders

User Documents location	Shows the default path to the project folder.
Local projects location	Shows the default path to the project folder.
...	To browse to the project folder, click the browse button.
Automatically create document subfolders	Select this check box to enable the creation of individual subfolders for document types.
minute interval	Specify the interval between the savings when using Autosave in this box.
Document Locations	Launches the Document Locations dialog box.
Clear Recent Stations and Controllers	Clears the list of recently accessed stations and controllers.
Additional distribution package location	RobotWare 6 and related RobotWare add-ins mediapools are distributed as distribution packages. For RobotStudio to find them, they need to be located in a specific folder. If the folder is not specified, the default location is used. On a Windows installation with English language, the default folder is C:\User\<user name>\AppData\Local\ABB\DistributionPackages. The location can be customized by entering a search path here.
Download packages to this location	Select this check box to dowload distribution packages to the user defined location instead of the default folder.

Continues on next page

Unpacked RobotWare Location	Shows the default path to the unpacked RobotWare folder.
Media Pool for RobotWare 5.x	This is where RobotStudio searches for RobotWare 5.xx mediapools.

Options:General:Screenshot

Entire application window	Select this option to capture the entire application.
Active document window	Select this option to capture the active document window, typically the graphics window.
Copy to clipboard	Select this check box to save the captured image to the system clipboard.
Save to file	Select this check box to save the captured image to file.
Location	Specify the location of the image file. The default location is the "My Pictures" system folder.
...	Browse for the location.
File name	Specify the name of the image file. The default name is "RobotStudio" to which is added a date.
The file suffix list	Select the desired file format. The default format is JPG.

Options:General:Screen Recorder

Framerate	Specify the framerate in frames per second.
Start recording after	Select this option to start recording after the specified time.
Stop recording after	Select this option to stop recording after the specified time.
Include mouse cursor	Select this option to include the mouse cursor for the functions **Record Application** and **Record Graphics**.
Resolution - Same as window	Select this option to use the same resolution as in the graphics window.
Resolution - Limit resolution	Select this option to scale down the resolution as per the **Maximum Width** and **Maximum Height** you specify.
Maximum width	Specify the maximum width in pixels.
Maximum height	Specify the maximum height in pixels.
Video compression	Select the video compression format. Note that DivX format is not supported.
Location	Specifies the location of the videos.
File name	Enter a file name to save the output file in an MP4 format.

Options:Robotics:Text Editor

Show line numbers	Select this check-box to view line numbers in the RAPID editor
Show ruler	Select this check-box to show the ruler in the RAPID editor
Show whitespace	Select this check-box to show whitespace characters in the RAPID editor
Word wrap	Select this check-box if you want to wrap long lines.
Convert tabs to spaces	Select this check-box to convert tabs to spaces in the RAPID editor

Continues on next page

Continued

Tab size	Specify the number of spaces for a Tab press.
RAPID Text styles	Specify the appearance of the various text classes.
Text color	Specifies the text color of the RAPID editor.
Background color	Specifies the background color of the RAPID editor.
Bold	Select this check-box for bold-face fonts in the RAPID editor.
Italic	Select this check-box for italicized fonts in the RAPID editor.
Follow program pointer by default	Select this check-box to enable the program pointer by default.

Options:Robotics:Graphical Programming

Show dialog when warning for globally defined workobjects	Select this check box if you want RobotStudio to display a warning when there are workobjects with the same name that have been declared as in other tasks. Default value: selected.
Show synchronize dialog box after loading program /module	Select this check box if you want the synchronize dialog box to be displayed when you have loaded a program or a module. Default value: selected.
Show notification that default data is used	Select this check box if you want to be notified that *wobj0* and/or *tool0* is active and will be used in the current action. Default value: selected.
Set as active when creating tooldata	Select this check box if you want newly created tooldata to be set as active. Default value: selected.
Set as active when creating workobjects	Select this check box if you want newly created workobjects to be set as active. Default value: selected.
AutoPath	Specify the maximum gap allowed when creating an AutoPath.

Options:Robotics:Synchronization

Use default synchronization locations	Converting data, such as target to Workobject, shall use the default behavior for synchronization locations. Default value: selected.
Show default synchronization locations notification	Notifies of the behavior above. Default value: selected.
Declaration default locations	Specify the locations for corresponding objects when synchronizing to the VC.

Options:Robotics:Mechanism

Approach Vector	Select the approach vector. Default value: Z.
Travel Vector	Select the travel vector. Default value: X.
When jumping to a target or move instruction with undefined configuration • **Show dialog for setting the configuration** • **Use the configuration closest to the current one**	Select the required option to enable configuration to either allow the user to set the configuration or to select a configuration closest to the current one when it jumps to target or move instructions. **Show dialog for setting the configuration** is selected by default.

Continues on next page

Options:Robotics:Virtual Controller

Always on top	Select this check box if you want to have the virtual FlexPendant always on top. Default value: selected.
Enable transparency	Select this check box if you want parts of the virtual FlexPendant to be transparent. Default value: selected.
Logging	After restarting the controller, • Select this check box to log the console output to "console.log" in the controller directory • Select this check box to log the console ouput to a console window
Automatically open virtual Operator Window	Select this check box to automatically open the virtual Operator Window. Default value: Enabled.

Options:Online:Authentication

Recent Users	Lists the recent users.
Remove/Remove All	Click these buttons to remove one or all recent users, respectively.
Enable Automatic Logoff	Select the check box if you want to log off automatically.
Timeout	Determines the length of the session before being automatically logged off.
Displays languages for controller texts such as event logs. • RobotStudio language • FlexPendant language	Select the application that controls the language of the controller event logs.

Options:Online:Online Monitor

Update Rate (s)	Specifies the update interval.
Revolute Joint Limits	Sets the revolution limit for joints.
Linear Joint Limits	Sets the linear limit for joints.
Singularities	Sets the singularities.

Options:Online:Jobs

Max number of devices processed in parallel	Specifies the number of devices for which a job is executed in parallel.
Directory for log files and report files	Specifies the directory for log/report files.

Options:Graphics:Appearance

Anti-aliasing	Move the slider to control the multisampling level used to smooth jagged edges. The available options are hardware dependent. RobotStudio must be re-started for this setting to take effect.
Font	Specifies the font used in markups.
Advanced lighting	Select the check box to enable advanced lighting by default.

Continues on next page

Continued

Perspective	Click this option to view the perspective view of the object by default.
Orthographic	Click this option to view the orthographic view of the object by default.
Custom background color	Click the colored rectangle to change default background color.
Show floor	Select the check box if you want the floor (at z=0) to be displayed by default. Change the floor color by clicking the colored rectangle. Default values: selected.
Transparent	Select the check box if you want the floor to be transparent by default. Default values: selected.
Show UCS Grid	Select the check box if you want the UCS grid to be displayed. Default value: selected.
Grid Space	Change the UCS grid space in the X and y coordinate directions by entering the requested value in the box. Default value: 1000 mm (or equivalent in other units).
Show UCS coordinate system	Select the check box if you want the UCS coordinate system to be displayed. Default value: selected.
Show world coordinate system	Select the check box if you want the coordinate systems to be displayed. Default value: selected.
Show navigation and selection buttons	Select this check box to have the navigation and selection buttons on the graphics window.

The settings you make take effect when creating a new station or when selecting **Default View Settings** from the **Settings** menu of the **View** tab of the Graphics Tools ribbon.

Options:Graphics:Performance

Rendering detail level	Select if the detail level is to be Auto, Fine, Medium or Coarse. Default value: Auto.
Render both sides of surfaces	Select the check box if you want to ignore the back-facing triangles. Default value: selected. Culling back-facing triangles improves the graphics performance but may give unexpected display if surfaces in models are not faced correctly.
Cull objects smaller than	Select the size in pixels under which objects will be disregarded. Default value: 2 pixels.

The settings you make here are generic for all objects in RobotStudio. With the **Graphic Appearance** dialog box you can, however, override some of these settings for single objects.

Options:Graphics:Behavior

Navigation	Select a navigational activity and then specify the mouse buttons to be used for the selected navigational activity.
Navigation sensitivity	Select the navigation sensitivity when using the mouse movements or navigation buttons by clicking the bar and dragging it into position. Default value: 1.
Automatically adjust view center distance	Select to automatically adjust the view center distance when rotating or zooming a 3D view.

Continues on next page

Selection radius (pixels)	Change the selection radius (that is, how close the mouse cursor click must be to an item to be selected) by entering the requested pixel value in the box. Default value: 5.
Selection highlight color	Click the colored rectangle to change the highlight color.
Selection preview	Select the check box to enable temporarily highlighting of items that may be selected when the mouse cursor passes over them. Default value: selected.
Show local coordinate system for selected objects	Select the check box to show the local coordinate system for the selected objects. Default value: selected.

Options:Graphics:Geometry

Detail Level	Specify the level of detail required when importing geometries. Select **Fine**, **Medium** or **Coarse** as required.

Options:Graphics:Stereo/VR*

Mirror Output	Displays the image available in the VR glasses in the Graphics view.
Quality	Move the slider to adjust the quality of the image to an acceptable level of lag.
Disable Anti-Aliasing	This option is disabled by default for better performance.

Options:Simulation:Clock

Simulation speed	Sets the simulation speed relative to real time. You can define the simulation speed to a maximum of 200%
As fast as possible	Select this check box to run the simulation as fast as possible. When you select this option, the simulation speed slider is disabled.
Simulation timestep	Specifies the simulation timestep.
Run time slice in parallel for multiple controllers	When simulating a large number of controllers (such as ten controllers), this option may increase performance by utilizing multiple CPU cores. This option is hardware dependent and hence may give different results depending on the computer used.

Options:Simulation:Collision

Perform collision detection	Select if collision detection is to be performed during simulation or always. Default value: always.
Pause/stop simulation at collision	Select this check box if you want the simulation to stop at a collision or at a near miss. Default value: cleared.
Log collisions to Output window	Select this check box if you want the collisions to be logged to the output window. Default value: selected.
Log collisions to file:	Select this check box if you want to log the collisions to a file. Browse for the file to log in by clicking the browse button. Default value: cleared.

Continues on next page

Continued

Enable fast collision detection	Select this check box to enhance the performance by detecting collisions between geometrical bounding boxes instead of geometrical triangles. This might result in falsely reported collisions, since the triangles are the true geometry and the bounding boxes always are larger. All true collisions will, however, be reported. The larger the object, the greater the number of false collisions that are likely to be detected.
View	Click this button to open the log file specified in the file box in Notepad.
Clear	Click this button to delete the log file specified in the file box.
...	Click this button to browse for the file in which you want to log the collisions.

Options:Simulation:Physics

Collision Geometry detail level	Set the slider to set the physics collision geometry either to faster or to a more accurate state.

B Terminology

A

ABB Library

The default repository of downloaded robots, positioners, tracks and their respective galleries.

Add-In

A software program that expands the capabilities of RobotStudio or RobotWare. Creating Add-Ins is the recommended way for third party developers to add new features into RobotWare or RobotStudio.

A RobotWare Add-In contains RAPID modules and configuration files that holds the code for loading the add-in and configuring it at start up. The Add-In may also include .xml files with event log messages in different languages. Add-Ins can be packaged using the RobotWare Add-In Packaging tool. You can download the tool from http://www.abb.com/abblibrary/DownloadCenter/

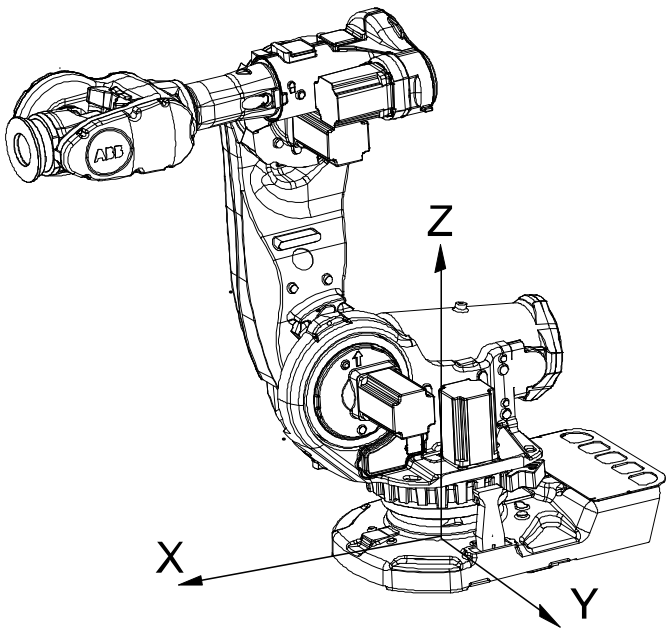
Activation key

A 25-character string that ABB sends by e-mail during RobotStudio purchase. This key is used for activating RobotStudio standalone license during manual activation in the absence of Internet connection.

B

Base frame

The base coordinate system is called the Base Frame (BF). The base frame for a robot is located at the center of its foot. It describes the location of the robot in relation to the world coordinate system.



xx0300000495

Continues on next page

Continued

Ball joint

Defined by a point that allows free rotation but no translation.

Body

A body is a shape, which can be a solid, surface, or curve.

Boot Server

It is a software installed in the controller along with RobotWare. The Boot Application on the FlexPendant copies the installation files for the selected system to the controller inbox. The Boot Server application on the robot controller uses these files in the controller inbox to create a RobotWare system on the memory card in the robot controller. Once the installation finishes, the selected RobotWare system starts.

Breakpoint

A breakpoint is an intentional stop placed for debugging in a RAPID program. A breakpoint is a signal that tells the debugger to temporarily stop the RAPID program at a certain point. When execution stops at a breakpoint, the program is in the break mode. Entering the break mode does not stop the program, you can resume program execution at any time.

C

Category 0 stop

Stop by immediate removal of power to the actuators. Mechanical brakes are applied. A robot that is stopped with a category 0 stop does not follow its programmed path while decelerating.

Category 1 stop

Controlled stop with power available to the actuators to achieve the stop. Power is removed from the actuators when the stop is achieved. A robot that is stopped with a category 1 stop follows its programmed path while decelerating.

Checksum

A checksum is an auto-generated unique mandatory text string of 64 characters appended to the safety configuration file of a controller. This string captures any changes that may happen to the file during transmission or installation. It is applied to the safety configuration file that Visual SafeMove creates.

Clip Plane

A clip plane is an imaginary infinite plane that cuts through geometric objects in the station. It allows the user to temporarily cut parts of the model away to help visualize the interior of a geometry or mesh. Objects on one side of the plane are visible while objects on the other side are invisible. A station can contain multiple clip planes, but each graphics view can only have one active clip plane.

Collision Geometry

Collision geometry is the simplified shape of an object where sharp edges and uneven surfaces of complex geometries are removed for easiness in collision calculations. The simplified collision geometry is used in physics simulations. RobotStudio uses regular geometry for collision detection.

Continues on next page

Collision detection

Useful to identify intersections between 3D objects in a station during programming phase and to detect any probable collisions with robot. It helps in modifying robot programs accordingly and avoid collisions on the shop floor at run-time. There is a performance penalty in using collision detection as it is a compute-intensive feature and demands computer resources extensively for collision calculations. Therefore, it is recommended that it must include parts that can potentially collide and exclude others during collision calculations.

Collision Set

A collision set is a pair of object sets that are checked for mutual collisions. A common use of collision sets is to create one for each robot in the station. Add robot and its tool in the first object set and objects against which you want to check collision in the other. Each collision set can be activated and deactivated separately.

Curve

A curve is a wire body like a line, circle, arc, polygon, polyline, or spline.

Cylindrical joint

Defined by a line and is a combination of prismatic and rotational joint.

Coordinate system

A coordinate system specifies the position and orientation of an object in the 3D space using three coordinates x, y and z. Orientation of an object can be specified either by using three angles or quadrants. RobotStudio allows using the following coordinate systems to define the orientation and placement of components. A coordinate system defines a plane or space by axes from a fixed point called the origin. Robot targets and positions are located by measurements along the axes of coordinate systems. A robot uses several coordinate systems, each suitable for specific types of jogging or programming. RobotStudio uses the following coordinate systems, World, Local, UCS, Active Work object, Active Tool.

- The base coordinate system is located at the base of the robot. It is the easiest one for just moving the robot from one position to another.
- The work object coordinate system is related to the work piece and is often the best one for programming the robot.
- The tool coordinate system or the Tool Center Point frame(TCP) defines the position of the tool the robot uses while reaching the programmed targets.
- The world coordinate system defines the robot cell, all other coordinate systems are related to the world coordinate system, either directly or indirectly. The world coordinate system has its zero point on a fixed position in the cell or station. This makes it useful for handling several robots or robots moved by external axes.

By default, the world coordinate system coincides with the base coordinate system. The user coordinate system is useful for representing equipment that holds other coordinate systems, like work objects.

Continues on next page

Continued

Cycle time

Simulations are calculated cyclically. The cycle time specifies the time frame in which the calculations are to be performed and data is to be exchanged.

D

Drive module

Houses power supply and the drive units of the robot and additional motors. If you have external axis in the system the corresponding drive modules must be in place.

Distribution Package

Distribution Package is the basic unit for the distribution and installation. Package Components are the smallest non-divisible unit of distribution containing a version and type, for example, a RobotStudio add-in. The contents of the Distribution Package may be installed on an embedded device such as the robot controller. RobotWare 6 and related RobotWare add-ins media pools are packaged and distributed in specific folders called distribution packages.

On a Windows installation with English language, the default folder is `C:\User\<user name>\AppData\Local\ABB\DistributionPackages`, this location can be customized. A distribution package may consist of one or more products. When distributed as one file, the suffix of the file is .rspak. Use the Install Package command in the Add-Ins page to install a distribution package.

Locations of distribution package

In RobotStudio 6, RobotWare and related packages are referred as application data. A distribution package is available in the following locations.

- ProgramData: used when the appdata is shared among users on the computer. If RobotWare is installed with RobotStudio, the path is `%ProgramData%\ABB Industrial IT\Robotics IT\DistributionPackages\`.
- LocalAppData: used if a package or manifest is installed by a particular user. If RobotWare is installed with RobotStudio, the path is `Users\<username>\AppData\Local\ABB Industrial IT\Robotics IT\DistributionPackages`.
- Customized location (optional): may be used when several users share a package repository. For more information, see Additional distribution package location in RobotStudio Options:General:Files & Folders.

E

External Axis

Moving equipment that is controlled by the robot controller (in addition to the robot) is denoted as an external axis, for example, track motion, a positioner, and so on.

Entry point

Point where the program execution starts.

Continues on next page

F

Frame

Frame is the visual representation of a coordinate system in RobotStudio.

- Position of a component is represented with respect to World, Base and Work object frames.
- Orientation format is set to Quaternion or Euler angles.
- Position angle format is set to Angles.
- Presentation angle unit can be set to Degrees or Radians.

Face

Each surface of the body is called a face. Solid bodies are 3D objects, made up of faces. A true 3D solid is one body containing multiple faces.

Freerun mode

Controllers run independently of each other. The cycle time will be accurate, but the timing for setting signals and triggering events will be inaccurate.

G

Geometry

3D representation of real objects like box, cylinder and so on. CAD models of work pieces and custom equipment are imported as geometries to the station. Geometry consists of two layers; the mathematical representation of the curves and surfaces known as boundary representation (BReps), and the graphics layer, containing triangles that approximate the BReps. Graphics layer is used in collision detection and for visualizing the mathematical layer.

I

Instruction Templates

A RAPID instruction file (template) containing predefined argument values used to create new instructions. These templates can be created for RAPID instructions in the virtual controller.

J

Joint

A joint defines how links around are connected. The most common joint types are prismatic or linear. But there are also ball joints, cylindrical, and lock joints.

Continues on next page

Continued

L

License

RobotStudio features are activated using the activation key. Activation key is a 25-character key that is availed when you purchase RobotStudio. RobotWare options are enabled/unlocked by one or more RobotWare license files(.rlf). Several license files can be combined for one RobotWare. The license file is delivered with the robot. To extend the Virtual Controller with more RobotWare options, contact ABB. Only the RobotWare options that are made available/unlocked by the license file will be available for selection in the Installation Manager while building or modifying the RobotWare.

RobotWare license decides the parts of RobotWare (supported robot models, options and so on.) that must be part of the system. When running a system on a robot controller, it must be built with the license that was delivered with the robot. For running a virtual controller (for simulations in RobotStudio) either a license from a real robot or a virtual license can be used. Using a license from a real robot is a quick way to ensure that the virtual controller matches that robot. Using a virtual license provides possibility to simulate and evaluate any robot model with any configuration. A virtual controller built with a virtual license cannot be run on a robot controller.

Library files

Library files are standalone external reusable files that are added to a RobotStudio station. The ABB product range of robots are downloaded as library files. Library files contain geometrical data and RobotStudio specific data. For example, when a tool is saved as a library file, its tool data is saved along with the CAD data.

Local origin

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

Lock joint

Connects two objects and does not allow them to move in relation to another.

Link

A link is a mechanical part. Several links are connected through joints to form a manipulator (mechanism).

M

Mechanism

A mechanism is a graphical representation of a robot, tool, external axis, or device, various parts of a mechanism move along or around axes.

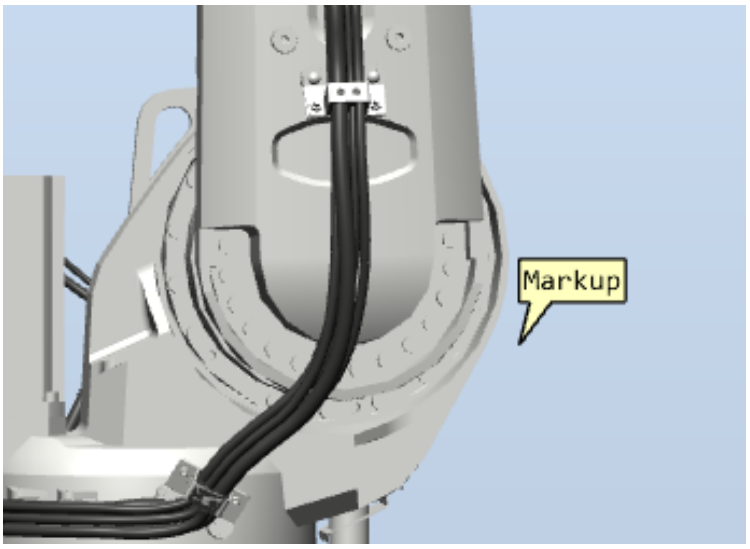
Continues on next page

MultiMove

Controls as many as 4 robots (36 axes) at a time. In a multimove system, common work objects are shared between robots which requires complex coordinated patterns. MultiMove also facilitates a dynamic switch between independent and coordinated motion.

Markup

A markup is a text box displayed in 3D. It is part of the station and appears as a text bubble pointing to a position in the **Graphics** window.



xx1900000306

Mechanical unit

A mechanical unit is the representation of a robot or one or more external axis in the robot controller, for example, robot, track motion and so on.

Module

RAPID code of the controller is structured into modules. A module contains several routines of type procedure, function or trap. Modules are of two types system and program.

N

Near-miss

Near-miss occurs when objects in collision sets are closer to each other but it avoids collision. Each collision set has its own near-miss settings. The near-miss feature can be used to add a margin to collisions and to gauge distance between the components in the station during movements.

O

Offline

User is disconnected from a robot controller and is working with a virtual controller.

Continues on next page

Continued

Orientation

The orientation of an object such as a line, plane or rigid body is it's placement in space. It is the imaginary rotation that is needed to move the object from a reference placement to its current placement. A rotation may not be enough to reach the current placement. It may be necessary to add an imaginary translation too. The location and orientation together fully describe how the object is placed in space. The above-mentioned imaginary rotation and translation may be thought to occur in any order, as the orientation of an object does not change when it translates, and its location does not change when it rotates.

Orientation formats available in RobotStudio are Quaternion, Euler angles and RPY angles.

Quaternion

A quaternion is a mathematical representation of orientation. They are points in space represented by their coordinates. A quaternion consists of four values between -1 and 1. The sum of its squares must be equal to one, that is, it has to be normalized (in which case it may be called unit quaternion).

Euler Angles

The term Euler implies that each angle is applied to the original coordinate system (before the rotations are applied). The angles describe orientations around different axes and in different order. The convention used in RobotStudio and for the IRC5 controller is Euler ZYX, which means the first values describe the angle to rotate around the z axis, the second value describes the orientation angle around the original Y-axis and the last value describes the orientation around original x-axis. There are also other conventions like Euler ZYZ, and Euler XYZ which ABB does not use.

RPY Angles

The RPY convention describes orientation with three angles, it is short for Roll, Pitch, Yaw. Any target orientation can be reached, starting from a known reference orientation, using a specific sequence of intrinsic rotations, whose magnitudes are the Euler angles of the target orientation. The difference compared to the Euler-convention is that each angle describes orientation around the new, rotated coordinate system. When rotating using the RPY convention, then the first angle describes orientation around x (same as Euler), but the second angle describes orientation around the y-axis of the rotated coordinate system (different from Euler), and the z-angle describes orientation around the rotated z axis. The RPY representation is equivalent of the Euler ZYX representation.

Offline programming

Robot programming using the virtual controller.

Open Platform Communications Unified Architecture (OPC UA)

It is a platform independent protocol and communicates data continuously among PLCs on the shop floor, RTUs in the field, HMI stations, and software applications on desktop PCs. Even when the hardware and software are from different vendors, OPC compliance makes continuous real-time communication possible.

Continues on next page

Online

User is connected directly to a robot controller through the network.

P

Path

A path is a sequence of targets with move instructions that the robot follows.

Part

Top node of a geometry is called a part.

Pack & go

Way to share RobotStudio stations by combining the station and virtual controllers packaged into one file.

Physics behavior

Depicts to what extend an object's physics behavior shall be simulated, any of the following types can be assigned:

- Dynamic: For a dynamic object, the physics simulation controls its motion which is affected by gravity. It can interact with other objects in the physics simulation.
- Kinematic: Here, an object interacts with other simulated objects, but its motion is controlled outside the physics simulation, for example, a robot whose movement is controlled by a robot controller is an object depicting kinematic behavior.
- Fixed: A fixed object takes part in the physics simulation but remains in a fixed position as if it has an infinite mass. Other objects can collide with it but there will not be any resultant motion.
- Inactive: Here, the object does not take part in the physics simulation, other objects move cross this object without colliding.

Positioner

A positioner is used to position a work piece for the robot to have better access. In arc welding, positioners are used to re-orient the work piece so that the weld is always done vertically due to gravity.

Position

Three coordinates that describe the x, and y and z- position of a point in a given coordinate system. In RobotStudio position of an object can be displayed relative to the reference coordinate systems World, Base and Work object.

Product

In the context of RobotWare 6, product is the collective name for software such as RobotWare, RobotWare add-ins, third party software and so on. Products are either free or licensed, licensed products require a valid license file.

Project

Projects add structure to the station data. It contains folders for structuring station data so as to keep related data together.

Continues on next page

Continued

Protected Smart Component

A Smart Component which is protected using a password from being edited.

Prismatic joint

Allow two connected links to move along a line that defines the joint.

PLCSIM Advanced

Virtual PLC.

R

Robotware keys

The RobotWare key is the license key that decides the robot models and RobotWare options to run on the controller. The license key is delivered with the controller. The RobotWare keys unlock the RobotWare options included in the system and determine the RobotWare version from which the RobotWare system will be built. For IRC5 systems, there are three types of RobotWare keys:

- The controller key, which specifies the controller and software options.
- The drive keys, which specify the robots in the system. The system has one drive key for each robot it uses.
- Add-ins specify additional options, like positioner external axes.

Using a virtual key, you can select any RobotWare option, but a RobotWare system created from a virtual key can only be used in a virtual environment such as RobotStudio.

RobotWare license

This license unlocks the RobotWare options, for example, robots and RobotWare options. To upgrade from RobotWare version 5.15 or earlier, replace the controller main computer and get RobotWare 6 licenses. Contact ABB Robotics service representative at www.abb.com/contacts.

RobotWare system

A set of software files that, when loaded into a controller, enables all functions, configurations, data, and programs controlling the robot. RobotWare systems are created in RobotStudio. These systems can be saved on a PC or on a control module. RobotWare systems can be edited by RobotStudio or the FlexPendant.

Robot Controller

A physical robot controller. It contains all functions needed to move and control the robot.

RobotWare

Set of software products used to configure a robot controller.

RAPID

Programming language for ABB robot controller.

Rail

A mechanism consisting of a linear axis with a carriage on which the robot is mounted.

Continues on next page

Robtarget

RobotStudio targets are translated to the RAPID data type robtarget during RAPID synchronization. It defines the position and orientation that the TCP shall reach. A robtarget defines a point in 3D space when it is associated with a work object. The position is defined based on the coordinate system of the work object, including any program displacement.

Routine

A well-defined part of a program for carrying out the intended task. Routines are either procedures, functions or traps.

Rotational joint

Defined by a line around which parts can rotate.

S

Station

A station is the 3D representation of the virtual robot cell. It is saved to a file with the extension *.rsstn.

Station logic

Station logic defines how smart components and virtual controllers of a station are connected.

State

State contains selected modifiable aspects of objects and its child objects which can be saved and restored when required.

Station Components

Physical objects such as robot, fixtures, tools, fences and so on, that are used to design an efficient and maintainable robotic cell are collectively referred to as station components.

Synchronization

The synchronization function converts targets, workobjects, tools and paths in the 3D environment to RAPID code in the virtual controller and vice versa.

Smart Component

Smart Component is a RobotStudio object (with or without a 3D graphical representation) whose properties are implemented by the code-behind or by aggregating other Smart Components. The base components that are availed with RobotStudio installation provides a complete set of basic building blocks. They can be used to build user defined Smart Components with more complex properties.

System

A set of software files that, when loaded into a controller, enables all functions, configurations, data and programs controlling the robot system. These systems can be saved on a PC or on a control module. RobotWare systems can be created and edited in RobotStudio or FlexPendant.

Continues on next page

Continued

Solid model

Solid model is a consistent set of principles for mathematical and computer modeling of 3D solids, for example, a robot mechanism. Solid models are 3D objects, made up of faces consisting of shapes like boxes, cones, cylinders, pyramids, or spheres.

Solids (Primitive solids)

Basic 3D shapes like boxes, cones, cylinders, spheres, wedges, pyramids, and donuts. Combine primitive shapes to create more complex solids.

STATIC task

A STATIC task gets restarted at the current position of the robot when the system was powered off.

SEMISTATIC task

A SEMISTATIC task gets restarted from the beginning whenever the power is turned on. A SEMISTATIC task will also initiate the restart sequence, reload modules specified in the system parameters if the module file is newer than the loaded module.

Station Viewer

It can playback a station in 3D without RobotStudio. It packages the station file together with the files needed to view the station in 3D. It can also play recorded simulations.

Simit

SIMIT is a simulation platform from Siemens for virtual commissioning of factory automation.

Symbol

A signal is identified by this name in SIMIT.

T

Tool

A tool is an object that can be mounted directly or indirectly on the robot turning.

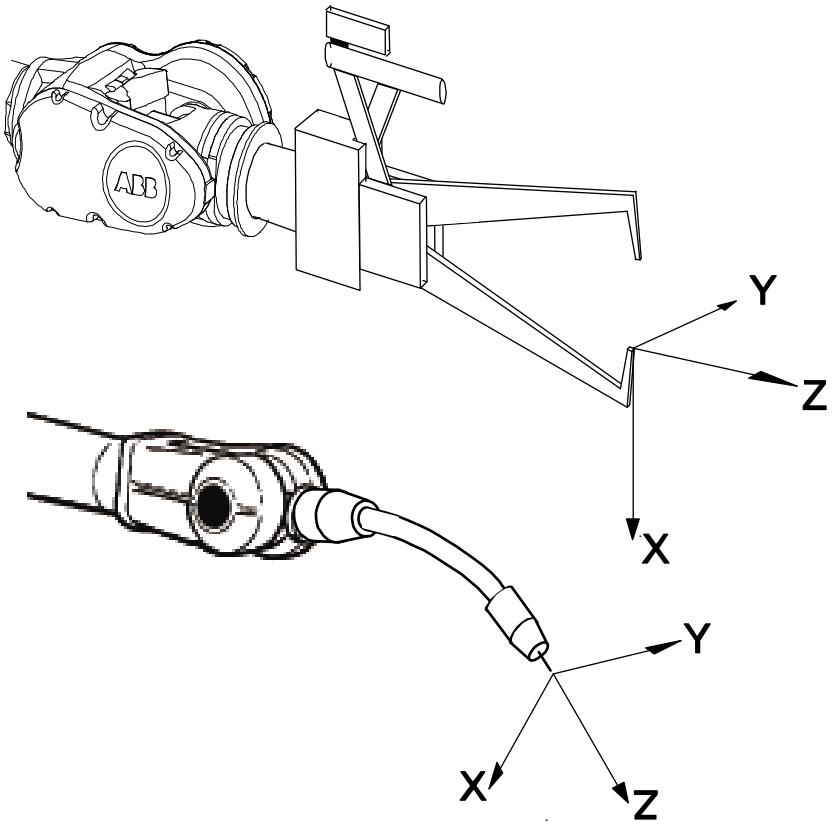
Tooldata

A tool is represented with a variable of the data type *tooldata*. Tooldata represents characteristics of a tool such as position and orientation of the TCP and the physical characteristics of the tool load.

Continues on next page

Tool Centre Point (TCP)

Refers to the point in relation to which robot's positioning is defined. It is the center point of the tool coordinate system that defines the position and orientation of the tool. TCP has its zero position at the center point of the tool. The tool center point also constitutes the origin of the tool coordinate system. Robot system can handle a number of TCP definitions, but only one can be active.



en0300000497

Task

A task is an activity or piece of work. RobotStudio tasks are either Normal, Static or Semistatic.

Task frame

Represents the origin of the robot controller world coordinate system in RobotStudio.

Track motion

A mechanism consisting of a linear axis with a carriage on which the robot is mounted. The track motion is used to give the robot improved reachability while working with large work pieces.

Target

Target signifies the position to which the robot is programmed to move. It is a RobotStudio object that contains the position and orientation of the point that the robot must reach. Position data is used to define the position in the move instructions to which the robot and additional axes will move.

Continues on next page

Continued

As the robot is able to achieve the same position in several different ways, the axis configuration is also specified. Target object contains values that shows position of the robot, orientation of the tool, axis configuration of the robot and position of the additional logical axes.

Time slice mode

During a simulation involving stations with one or several controllers, RobotStudio manages time using the time slice mode for controller synchronization. In this mode, each controller gets a single time slice for execution. When all the participants have completed execution of their respective allotted time slice, RobotStudio moves on to the next time slice.

U

User Authorization System

Defines the correct access level for each user, protects the system from unauthorized usage.

User library

Library files imported to RobotStudio.

V

Virtual controller

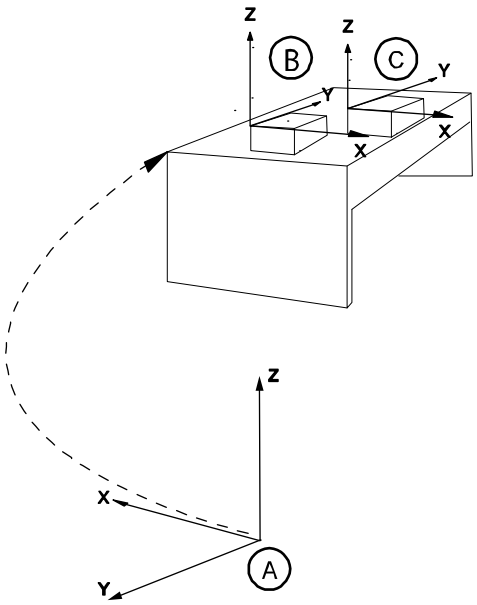
A software that emulates the robot controller on the PC. It is used for offline programming and simulation. Virtual controller replicates the RobotWare system.

Continues on next page

W

Work Object

A work object is a local coordinate system that indicates the reference position (and orientation) of a work piece. The work object coordinate system must be defined in two coordinate systems , the user coordinate system (related to the world coordinate system) and the object coordinate system (related to the user coordinate system). Work objects are often created to simplify jogging along the object's surfaces. Work objects should always be global to be available to all modules in the program.



xx0600002738

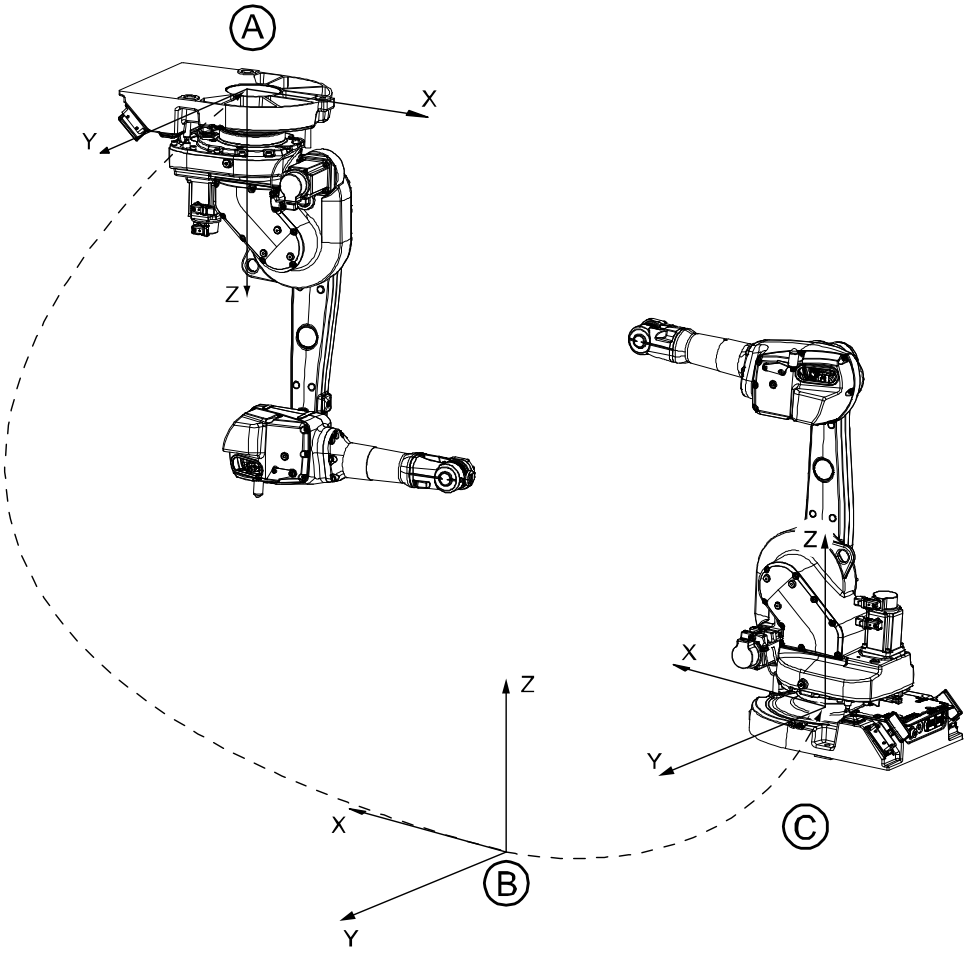
A	World coordinate system
B	Work Object coordinate system 1
C	Work Object coordinate system 2

Continues on next page

Continued

World coordinate system

The world coordinate system represents the entire station or the robot cell, all other coordinate systems are related to the world coordinate system, either directly or indirectly. The world coordinate system has its zero point on a fixed position in the cell or station. This makes it useful for handling several robots or robots moved by external axes. By default, the world coordinate system coincides with the base coordinate system.



en0300000496

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

Work Envelope

The defined area of space in which a robot can move is its work envelope. Work envelope for a robot is the maximum range of movement that can be visualized in 2D/3D graphics. Work envelope can be added to the station as a part, which can be saved in the station and exported as any geometry.

Continues on next page

Wobjdata

A work object is represented with a variable of the data type *wobjdata*. It describes the work object that the robot welds, processes, moves within, and so on.

This page is intentionally left blank

C Technical support

Overview

Contacting ABB

If you have any questions or problems with your RobotStudio installation, please contact your local ABB Robotics Service representative, see *http://www.abb.com/contacts*.

Have the following in mind

1. Running the latest version of RobotStudio ensures that it works properly and also includes improvements and new product functionality. ABB recommends that you update to the latest version of RobotStudio whenever a new version is available and before contacting ABB for support.
2. Provide a brief description explaining how to reproduce your problem.
3. Provide screenshots if applicable. (Use ALT + PRINT SCREEN to get an image of the active window instead of the entire screen.)
4. Generate a Full Scan with the **RobotStudio Support Tool** available next to RobotStudio in the Start menu. (Click **Start** > **Programs** > **ABB** > **RobotStudio** > **RobotStudio Support Tool**, then click on **Run Full Scan** and then click **Save Report**. Save this report and attach it with your problem description.
5. Provide us the following user information:
   a. name
   b. company
   c. contact information
   d. name of the operating system including the language details
   e. subscription ID of the purchased license
   f. Machine ID, see **Help** section of **File** tab

License support

For license-related questions, please contact the team responsible for license support directly at *softwarefactory_support@se.abb.com*

This page is intentionally left blank

Index

A
Add Controller, 368–369
Adjust Robtargets, 420
Application grants, 376

B
Backup, 86
  create backup, 86

C
collision
  sets, 140
configuration editor, 381
  instance editor, 386
configuration file, 384
Controller grants, 374, 376
  Backup and save, 374
  Calibration, 375
  Delete log, 376
  Edit RAPID code, 375
  Execute program, 374
  Full access, 374
  I/O write access, 374
  Manage UAS settings, 374, 376
  Modify configuration, 375
  Modify controller properties, 375
  Modify current value, 374, 376
  Program debug, 375
  Read access to controller disks, 375
  Safety Controller, 376
  Write access to controller disks, 375
controller world coordinate system, 91

D
detecting collision, 141
device browser, 390

G
graphics window, 43

L
LED, 293

M
Manage ScreenMaker project
  Close ScreenMaker, 307
Manage ScreenMaker Project
  Close project, 307
Managing ScreenMaker Projects
  Build project, 306
Modify project properties, 305

N
near-miss detection, 142
network security, 15
network settings
  firewall settings, 36
  local network connection, 36
  remote network connection, 36
  service port connection, 36

P
Properties
  Device Browser, 390
  Renaming the controller, 388
  Save System Diagnostics, 391
  Set controller ID, 388
  Set date and time, 388
  View controller and system properties, 390
Properties Window
  Event Help panel, 294
  Graphical Component Name panel, 294
  Properties window toolbar, 294
  Table panel, 294

R
Relation, 413
remote subnet, 37
Robot system button, 337
  adding an existing system, 338
  adding a template system, 338
  create system from layout, 337

S
safety, 14
station world coordiante system, 89
switch, 293
System Builder
  about virtual and real systems, 243
  download a system to controller, 255
System Configuration
  controller values, 409
  stored station values, 409
  used current station values, 409
system parameters
  editing parameters, 382

T
ToolBox
  ActionTrigger, 292
  BarGraph, 292
  CheckBox, 292
  ComboBox, 293
  CommandBar, 293
  ConditionalTrigger, 293
  ControllerModeStatus, 293
  DataEditor, 293
  Graph, 293
  GroupBox, 293
  ListBox, 293
  NumEditor, 293
  NumericUpDown, 293
  Panel, 293
  PictureBox, 293
  RapidExecutionStatus, 293
  RunRoutineButton, 293
  TabControl, 293
TpsLabel, 293

U
UCS, 94
user coordinate system, 94

V
VariantButton, 293

W
WorkObject, 94
world coordinate system, 89

ABB AB
Robotics & Discrete Automation
S-721 68 VÄSTERÅS, Sweden
Telephone +46 10-732 50 00

ABB AS
Robotics & Discrete Automation
Nordlysvegen 7, N-4340 BRYNE, Norway
Box 265, N-4349 BRYNE, Norway
Telephone: +47 22 87 2000

ABB Engineering (Shanghai) Ltd.
Robotics & Discrete Automation
No. 4528 Kangxin Highway
PuDong New District
SHANGHAI 201319, China
Telephone: +86 21 6105 6666

ABB Inc.
Robotics & Discrete Automation
1250 Brown Road
Auburn Hills, MI 48326
USA
Telephone: +1 248 391 9000

abb.com/robotics

3HAC032104-001, Rev AS, en