Trace back information:
Workspace OmniCore and R19C version a11
Checked in 2019-09-10
Skribenta version 5.3.012
The information in this manual is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this manual.

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

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

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

Keep for future reference.

Additional copies of this manual may be obtained from ABB.

Original instructions.

© Copyright 2019 ABB. All rights reserved.
Specifications subject to change without notice.
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? ....................................................................................................... 21
1.4 Connecting a PC to the controller ................................................................................................ 28
1.5 Network Settings .......................................................................................................................... 33
1.6 Managing user rights and write access on a controller ............................................................... 35
  1.6.1 Managing user rights and write access on an IRC5 controller ............................................... 35
  1.6.2 Managing user rights and write access on an Omnicore C30 controller .................................. 37
1.7 Manage user interface using mouse ............................................................................................. 39
1.8 Libraries, geometries and CAD files ........................................................................................ 43
1.9 How to install RobotWare and Add-Ins ....................................................................................... 46

2 Building Stations .......................................................................................................................... 49

2.1 Understanding stations and solutions .......................................................................................... 49
2.2 Preparing the computer for hosting RobotStudio ......................................................................... 51
2.3 Creating the station ....................................................................................................................... 54
2.4 Importing robots and related components .................................................................................... 55
2.5 Creating a virtual controller ......................................................................................................... 56
2.6 Synchronizing to virtual controller to create a RAPID program ............................................... 58
2.7 Configuring station with robot and positioner ............................................................................. 59
2.8 Configuring station with robot and Track motion ....................................................................... 60
2.9 Configuring Conveyor tracking .................................................................................................. 61
2.10 Configuring external axis .......................................................................................................... 64
2.11 Programming MultiMove systems ............................................................................................. 67
  2.11.1 Workflow for setting MultiMove ......................................................................................... 67
  2.11.2 Selecting robots and paths .................................................................................................... 68
  2.11.3 Executing motion instructions along paths .......................................................................... 69
  2.11.4 Tuning the motion behavior ............................................................................................... 70
  2.11.5 Creating paths ................................................................................................................... 73
  2.11.6 Setting up MultiMove without using Multimove functions .................................................. 74

3 Programming robots in the 3D environment .............................................................................. 75

3.1 Understanding offline programming ........................................................................................... 75
3.2 Robot axis configurations ........................................................................................................... 81
3.3 Creating a work object ................................................................................................................. 83
3.4 Creating a path with targets and move instructions ..................................................................... 84
3.5 Creating a path from an edge or curve ....................................................................................... 85
  3.5.1 AutoPath .............................................................................................................................. 85
3.6 Configuring a stationary tool ...................................................................................................... 87
3.7 Define arm configurations for the targets ................................................................................... 88
3.8 Testing positions and motions .................................................................................................... 89
3.9 Generating the RAPID program .................................................................................................. 90
3.10 Modifying target orientations ................................................................................................... 91

4 Understanding RAPID editor ...................................................................................................... 93

4.1 Working with RAPID editor ........................................................................................................ 93
4.2 RAPID editor intellisense ......................................................................................................... 95
4.3 Manage RAPID modules and programs .................................................................................... 97
4.4 Adding code snippets ................................................................................................................ 100
### 5 Testing and Debugging RAPID

5.1 Debugging a task .................................................. 113
5.2 Understanding program pointer ............................... 115
5.3 Working with RAPID breakpoints ........................... 117
5.4 Navigate through your program using RAPID Call Stack window ....................................... 118
5.5 Using RAPID Watch window for debugging your RAPID code ........................................ 119

### 6 Simulating programs

6.1 Playing Simulation .................................................. 121
6.1.1 Simulation Setup .................................................. 121
6.1.2 Simulation Control .............................................. 123
6.2 Detecting Collision .................................................. 124
6.3 Collision Avoidance ............................................... 128
6.4 I/O Simulation ...................................................... 132
6.5 Simulation time measurement ................................. 134
6.6 Understanding TCP trace ......................................... 135

### 7 Advanced simulation using smart components and physics

7.1 Understanding SmartComponents .......................... 137
7.1.1 Smart Component ............................................... 137
7.2 The Infeeder example ............................................. 139
7.3 Cable Simulation using Physics ................................. 150
7.4 Physics joints ...................................................... 152
7.5 Virtual commissioning using the SIMIT SmartComponent ........................................... 153

### 8 Deploying and Sharing

8.1 Saving and loading RAPID programs and modules .................................................. 161
8.2 Sharing a station .................................................... 162
8.3 Capturing the selected screen ................................. 164
8.4 Recording a simulation ........................................... 165
8.5 Creating a 3D animation of your simulation .............. 166
8.6 Deploying a RAPID program to a real controller .................................................. 167

### 9 Installing robot controller software

9.1 Using System Builder for managing RW 5 .................. 171
9.1.1 About System Builder ....................................... 171
9.1.2 Viewing system properties .................................. 173
9.1.3 Building a new system ....................................... 174
9.1.4 Modifying a system ........................................... 178
9.1.5 Copying a system .............................................. 182
9.1.6 Creating a system from backup ............................. 183
9.1.7 Downloading a system to a controller .................... 184
9.1.8 Examples using the System Builder when offline .................................................. 185
9.1.8.1 A system with support for one robot and one positioner external axis 185
9.1.8.2 Options settings for systems with positioners ........................................... 188
9.2 Using Installation manager for managing RW 6 and RW 7 ........................................ 189
9.2.1 Installation Manager ........................................... 189
9.2.1.1 About Installation Manager ........................................... 189
9.2.1.2 Startup and settings ............................................. 190
9.2.1.3 Building a new robot controller ........................................... 191
9.2.1.4 Modifying a robot controller ........................................... 193
# Table of contents

9.2.1.5 Copying a robot controller ............................................................ 195
9.2.1.6 Creating a robot controller from backup ................................. 197
9.2.1.7 Renaming a robot controller .................................................. 199
9.2.2 A MultiMove system with two coordinated robots .................... 200
9.2.2.1 Creating a coordinated system using System Builder .......... 200
9.2.2.2 Creating a coordinated system using Installation Manager .... 202

10 Working with system parameters ...................................................... 203
  10.1 System parameters ........................................................................... 203
  10.2 Adding instances ............................................................................ 205
  10.3 Copying an instance ........................................................................ 206
  10.4 Deleting an instance ....................................................................... 207
  10.5 Save one configuration file ............................................................ 208
  10.6 Saving several configuration files .................................................. 209
  10.7 Loading a configuration file ........................................................... 210

11 Monitoring robot signals ................................................................. 211
  11.1 Signal analyzer .............................................................................. 211
  11.2 Monitored signals ......................................................................... 212
  11.3 Recording signals ......................................................................... 214
  11.4 Recordings ...................................................................................... 215

12 Jobs ..................................................................................................... 217
  12.1 Understanding jobs ....................................................................... 217
  12.2 Creating a device list ..................................................................... 221
  12.3 Creating a new job ......................................................................... 222
  12.4 Reuse a job template ..................................................................... 223
  12.5 Scheduling jobs ............................................................................. 224

13 Screenmaker ....................................................................................... 225
  13.1 Introduction to ScreenMaker ......................................................... 225
  13.2 Development environment ............................................................ 228
  13.3 Working with ScreenMaker ......................................................... 233
    13.3.1 Managing projects ............................................................... 233
    13.3.2 Application variables ......................................................... 251
    13.3.3 Data binding ...................................................................... 252
    13.3.4 ScreenMaker Doctor .......................................................... 254
  13.4 Frequently asked questions .......................................................... 257
  13.5 Tutorial ......................................................................................... 260

A Options ............................................................................................... 269

B Terminology ....................................................................................... 277

C Technical support ............................................................................... 293

Index ..................................................................................................... 295
This page is intentionally left blank
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:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Getting Started</td>
</tr>
<tr>
<td></td>
<td>Contains installation, activation and licensing information, introduction to feature levels, network settings and user rights.</td>
</tr>
<tr>
<td>2</td>
<td>Building Stations</td>
</tr>
<tr>
<td></td>
<td>Contains information on how to prepare PC for hosting RobotStudio, introduction on solution 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.</td>
</tr>
<tr>
<td>3</td>
<td>Programming robots in the 3D environment</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>4</td>
<td>Understanding RAPID editor</td>
</tr>
<tr>
<td></td>
<td>Describes how to simulate and validate robot programs.</td>
</tr>
<tr>
<td>5</td>
<td>Testing and Debugging RAPID</td>
</tr>
<tr>
<td></td>
<td>Describes how to debug a task, introduces program pointer, RAPID breakpoints. RAPID Call Stack window and using RAPID Watch window for debugging code.</td>
</tr>
<tr>
<td>6</td>
<td>Simulating programs</td>
</tr>
<tr>
<td></td>
<td>Describes simulation setup, playing simulation, collision detection, collision avoidance and I/O simulation.</td>
</tr>
</tbody>
</table>

Continues on next page
Contents

Chapter | Contents
---|---
7 | Advanced simulation using smart components and physics
8 | Deploying and Sharing
9 | Installing robot controller software
10 | Working with system parameters
11 | Monitoring robot signals
12 | Jobs
13 | Screenmaker

References

<table>
<thead>
<tr>
<th>Reference</th>
<th>Document ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating manual - OmniCore</td>
<td>3HAC065036-001</td>
</tr>
<tr>
<td>Technical reference manual - RAPID Overview</td>
<td>3HAC065040-001</td>
</tr>
<tr>
<td>Technical reference manual - System parameters</td>
<td>3HAC065041-001</td>
</tr>
<tr>
<td>Application manual - Conveyor tracking</td>
<td>3HAC066561-001</td>
</tr>
<tr>
<td>Application manual - Electronic Position Switches</td>
<td>3HAC050996-001</td>
</tr>
<tr>
<td>Application manual - Integrated Vision</td>
<td>3HAC067707-001</td>
</tr>
<tr>
<td>Application manual - Controller software OmniCore</td>
<td>3HAC066554-001</td>
</tr>
</tbody>
</table>

Revisions

<table>
<thead>
<tr>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>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.</td>
</tr>
<tr>
<td>B</td>
<td>Released with RobotStudio 5.12.</td>
</tr>
<tr>
<td>C</td>
<td>Released with RobotStudio 5.13.</td>
</tr>
<tr>
<td>D</td>
<td>Released with RobotStudio 5.13.02.</td>
</tr>
<tr>
<td>E</td>
<td>Released with RobotStudio 5.14.</td>
</tr>
<tr>
<td>F</td>
<td>Released with RobotStudio 5.14.02.</td>
</tr>
<tr>
<td>G</td>
<td>Released with RobotStudio 5.14.02.01.</td>
</tr>
<tr>
<td>H</td>
<td>Released with RobotStudio 5.14.03.</td>
</tr>
<tr>
<td>J</td>
<td>Released with RobotStudio 5.15.</td>
</tr>
<tr>
<td>K</td>
<td>Released with RobotStudio 5.15.01.</td>
</tr>
<tr>
<td>L</td>
<td>Released with RobotStudio 5.60.</td>
</tr>
</tbody>
</table>

Continues on next page
### Overview of this manual

<table>
<thead>
<tr>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Released with RobotStudio 5.61.</td>
</tr>
<tr>
<td>N</td>
<td>Released with RobotStudio 6.0.</td>
</tr>
<tr>
<td>P</td>
<td>Released with RobotStudio 6.01.</td>
</tr>
<tr>
<td>Q</td>
<td>Released with RobotStudio 6.02.</td>
</tr>
<tr>
<td>R</td>
<td>Released with RobotStudio 6.03.</td>
</tr>
<tr>
<td>S</td>
<td>Released with RobotStudio 6.03.01.</td>
</tr>
<tr>
<td>T</td>
<td>Released with RobotStudio 6.04.</td>
</tr>
<tr>
<td>U</td>
<td>Released with RobotStudio 6.05.</td>
</tr>
<tr>
<td>V</td>
<td>Released with RobotStudio 6.06.</td>
</tr>
<tr>
<td>W</td>
<td>Released with RobotStudio 6.07.</td>
</tr>
<tr>
<td>X</td>
<td>Released with RobotStudio 6.08.</td>
</tr>
<tr>
<td>Y</td>
<td>Released with RobotStudio 2019.1.</td>
</tr>
<tr>
<td>Z</td>
<td>Released with RobotStudio 2019.2.</td>
</tr>
<tr>
<td>AA</td>
<td>Released with RobotStudio 2019.3.</td>
</tr>
</tbody>
</table>
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.


Product manuals

Manipulators, controllers, DressPack/SpotPack, 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.
• 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.
• 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.

Safety regulations

Before beginning work with the robot, make sure you are familiar with the safety regulations described in the manual Operating manual - General safety information.
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 damages and/or losses related to such security breaches, any unauthorized access, interference, intrusion, leakage and/or theft of data or information.
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 real controller and in offline mode, it is connected to a virtual controller that emulates a real 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 real 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 real 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.
## Details of Premium and Basic features

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

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

1. Requires the RobotWare option PC Interface on the real controller to enable LAN communication. This option is not needed for connection through the Service port or for virtual controller communication.

2. Requires the RobotWare option FlexPendant Interface on the robot controller system.
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:

<table>
<thead>
<tr>
<th>Part</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>2.0GHz or faster processor, multiple cores recommended</td>
</tr>
<tr>
<td>Memory</td>
<td>8 GB minimum. 16 GB or more if working with heavy CAD models.</td>
</tr>
<tr>
<td>Disk</td>
<td>10+ GB free space, solid state drive (SSD) recommended.</td>
</tr>
<tr>
<td>Graphics card</td>
<td>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.</td>
</tr>
<tr>
<td>Display settings</td>
<td>1920 x 1080 pixels or higher resolution is recommended.</td>
</tr>
<tr>
<td>dots per inch (dpi)</td>
<td>Only Normal size supported for Integrated Vision.</td>
</tr>
<tr>
<td>Mouse</td>
<td>Three-button mouse.</td>
</tr>
</tbody>
</table>

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

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

- To connect RobotStudio over Ethernet, enable the RobotWare option PC Interface.
- To run ScreenMaker or FlexPendant SDK applications on a robot controller, enable the RobotWare option FlexPendant Interface.

Software

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows 7SP1</td>
<td>64-bit edition</td>
</tr>
<tr>
<td>Windows 10 Anniversary Update or later</td>
<td>64-bit edition</td>
</tr>
</tbody>
</table>

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

Firewall settings

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

<table>
<thead>
<tr>
<th>Status</th>
<th>Name</th>
<th>Action</th>
<th>Direction</th>
<th>Protocol</th>
<th>Remote Address</th>
<th>Local Service</th>
<th>Remote Service</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>RobNetscan-Host</td>
<td>Allow</td>
<td>Out</td>
<td>UDP/IP</td>
<td>Any</td>
<td>Any</td>
<td>5512,5514</td>
<td></td>
<td>robnetscan-host.exe</td>
</tr>
<tr>
<td>IRSSController</td>
<td>Allow</td>
<td>In</td>
<td>UDP/IP</td>
<td>Any</td>
<td>5513</td>
<td></td>
<td></td>
<td>robnetscan-host.exe</td>
</tr>
<tr>
<td>RobComCtriServer</td>
<td>Allow</td>
<td>Out</td>
<td>TCP/IP</td>
<td>Any</td>
<td>Any</td>
<td>5515</td>
<td></td>
<td>robcomctriserver.exe</td>
</tr>
<tr>
<td>RobotFTP</td>
<td>Allow</td>
<td>Out</td>
<td>TCP/IP</td>
<td>Any</td>
<td>Any</td>
<td>FTP(21)</td>
<td></td>
<td>RobotStudio.exe</td>
</tr>
<tr>
<td>RobotStudio</td>
<td>Allow</td>
<td>Out</td>
<td>HTTP</td>
<td>Any</td>
<td>Any</td>
<td>80</td>
<td></td>
<td>RobotStudio.exe</td>
</tr>
<tr>
<td>RobotStudio</td>
<td>Allow</td>
<td>Out</td>
<td>HTTPS</td>
<td>Any</td>
<td>Any</td>
<td>443</td>
<td></td>
<td>RobotStudio.exe</td>
</tr>
</tbody>
</table>

**Note**

Port 443 is not required for controller communication, but it is required for external communication with ABB for availing services like upgrade notifications and online services.

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

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

**Note**

RobotStudio uses the current Internet Options, HTTP, and proxy settings to get the latest RobotStudio news. To view the latest RobotStudio news, go to the File tab and then click Help.
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 Microsoft 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 (xxxxxxxx-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.

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

<table>
<thead>
<tr>
<th>Type of license</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>Allows you to centralize license management by installing licenses on a single server rather than on individual client machines. A network 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 single network license allows several clients to use the software. Network licensing in RobotStudio uses the SLP Distributor server as the network licensing server. Network licenses are currently available only for authorized value providers and schools. 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.</td>
</tr>
<tr>
<td>Standalone</td>
<td>Allows a single user to install and use RobotStudio on a single computer.</td>
</tr>
</tbody>
</table>
1 Getting Started

1.3 How to activate RobotStudio?

Continued

<table>
<thead>
<tr>
<th>Type of license</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commuter</td>
<td>Allows a client system to work offline from the 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.</td>
</tr>
</tbody>
</table>

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 network and standalone licenses must be used carefully. If the activation key for standalone license is used for invoking network license, the activation fails. Hence carefully read the Subscription confirmation mail that ABB sends before activating RobotStudio installation.

Network license logging

SLP distributor provides option for logging the network 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 real 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 network license

1. Install the SLP Distributor server from the Utilities\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 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 network 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.

<table>
<thead>
<tr>
<th>To...</th>
<th>Use...</th>
</tr>
</thead>
</table>
| **Activate a network 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. |
| **Activate a network 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 `https://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. |
| **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. |

Your network license is now activated.

xx1300000051

xx1300000052

Continues on next page
### To... | Use...
---|---
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 network licensing.

Use **RobotStudio Activation Wizard** in the client system for setting up network license. Use this procedure to set up Network license for a client system.

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 On the <strong>File</strong> tab, click <strong>Options</strong> and go to **General:**Licensing.</td>
</tr>
<tr>
<td>2 On the <strong>Licensing</strong> page to the right, click <strong>Activation wizard</strong> to start the Activation Wizard.</td>
</tr>
</tbody>
</table>
| 3 In the Activation Wizard, on the **Activate RobotStudio** page, choose the option *I want to specify a network 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 Network Licensing to work, the client system should be online with the server.

**Tip**  
Network licenses are displayed as *Network* in the **View Installed Licenses** link of the **Licensing** page.
1 Getting Started

1.3 How to activate RobotStudio?

Continued

Activating commuter licenses

The commuter license key can be checked out if a workstation has to be
disconnected from the network. Normally, the workstation possessing a commuter
license key must remain connected through the network to the License Server.
The commuter license key is kept active on the workstation through a periodic
handshake between the workstation and license server. If the handshake terminates,
the key on the workstation expires and the application is notified through the license
manager.

A commuter license key does not require a handshake and thus the workstation
can be disconnected from the network. The commuter license in the client system
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. If a valid standalone license key is
available in the workstation, you cannot check out a commuter license key. Use
the activation wizard to along with the following procedure to check in/check out
a commuter license.

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 On the File menu, click Options and select General: Licensing.</td>
</tr>
<tr>
<td>2 On the Licensing page to the right, click Activation wizard to start the Activation Wizard.</td>
</tr>
<tr>
<td>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.</td>
</tr>
<tr>
<td>4 In the Commuter License page, select one of the following options as per the requirement:</td>
</tr>
<tr>
<td>• Check out a commuter license - In the Check out days box specify the number of days for keeping the license.</td>
</tr>
<tr>
<td>This option is disabled if a commuter license has already been checked out.</td>
</tr>
<tr>
<td>• Check in a commuter license - Choose this option to return the currently checked out license to the server.</td>
</tr>
<tr>
<td>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.</td>
</tr>
<tr>
<td>5 Click Finish to complete the check in/check out.</td>
</tr>
</tbody>
</table>

Tip

Network 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.
Demo Stations

RobotStudio provides a set of demo stations as part of the installation 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.

To open a trial station:

1. On the Start menu, select All Programs\ABB\RobotStudio X. x\RobotStudio.
2. In the File tab, click Open and start RobotStudio.
3. Browse to \Documents\RobotStudio\Stations.
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. The maximum number of concurrently connected FTP clients is 4.

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.
### 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.

**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.

---

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

<table>
<thead>
<tr>
<th>Action</th>
<th>Note</th>
</tr>
</thead>
</table>
| 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. |
| Connect a network cable to the network port of your PC.                |                                                                                                                                 |
| Connect the network cable to the service port on the controller, or to the service port on the computer unit. When connecting to the service port:  
  • Connect the network cable 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. |                                                                                                                                 |

[Diagram of the controller showing the service port labeled 'A Service port on the controller']
Connecting RobotStudio to OmniCore C30

Connectors in OmniCore C30

The following diagram shows the connectors on the front panel of the OmniCore C30 controller.

![Diagram of OmniCore C30 connectors]

**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.

**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.

Continues on next page
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.
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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP address</td>
<td>192.168.125.2</td>
</tr>
<tr>
<td>Subnet mask</td>
<td>255.255.255.0</td>
</tr>
</tbody>
</table>

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.
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.

<table>
<thead>
<tr>
<th>If the controller is in mode</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto</td>
<td>You will now get Write Access if it is available.</td>
</tr>
<tr>
<td>Manual</td>
<td>A message box on the FlexPendant will allow you to grant remote Write Access to RobotStudio.</td>
</tr>
</tbody>
</table>
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 functionalities. 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 administrators group

In addition to the *Default Group*, certain predefined user groups are available in the *real 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.
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.
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 C30 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 real 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.
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:

<table>
<thead>
<tr>
<th>To</th>
<th>Use the keyboard/mouse combination</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select items</td>
<td>left-cli</td>
<td>Just click the item to select. To select multiple items, press CTRL key while clicking new items.</td>
</tr>
<tr>
<td>Rotate the station</td>
<td>CTRL + SHIFT + left-cli</td>
<td>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.</td>
</tr>
<tr>
<td>Pan the station</td>
<td>CTRL + left-cli</td>
<td>Press CTRL + the left mouse button while dragging the mouse to pan the station.</td>
</tr>
<tr>
<td>Zoom the station</td>
<td>CTRL + right-cli</td>
<td>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.</td>
</tr>
<tr>
<td>Zoom using window</td>
<td>SHIFT + right-cli</td>
<td>Press SHIFT + the right mouse button while dragging the mouse across the area to zoom into.</td>
</tr>
<tr>
<td>Select using window</td>
<td>SHIFT + left-cli</td>
<td>Press SHIFT + the left mouse button while dragging the mouse across the area to select all items that match the current selection level.</td>
</tr>
</tbody>
</table>

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
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.

<table>
<thead>
<tr>
<th>Individual Axis</th>
<th>Axes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pan Right/Left</td>
<td>x1500000297</td>
<td>Moves the model right and left.</td>
</tr>
<tr>
<td>Zoom</td>
<td>x1500000299</td>
<td>Zoom the model in and out.</td>
</tr>
<tr>
<td>Pan Up/Down</td>
<td>x1500000298</td>
<td>Moves the model up and down.</td>
</tr>
<tr>
<td>Spin</td>
<td>x1500000301</td>
<td>Rotate around vertical axis.</td>
</tr>
<tr>
<td>Tilt</td>
<td>x1500000300</td>
<td>Tilts the model forwards and backwards.</td>
</tr>
<tr>
<td>Roll</td>
<td>x1500000302</td>
<td>Rolls the model sideways.</td>
</tr>
</tbody>
</table>

### 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.

Continued 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.

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

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 Move button.

A cross with arrows in the X, Y and Z directions now appear on the part. Drag the arrows to move the object.

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

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

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.
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 installation 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.

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.

<table>
<thead>
<tr>
<th>Format</th>
<th>File extensions</th>
<th>Option required</th>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DStudio</td>
<td>.3ds</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACIS</td>
<td>.sat, .sab, .sat, .asab</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CATIA V4</td>
<td>.model, .exp, .session</td>
<td>CATIA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CATIA V5/V6</td>
<td>.CATPart, .CAT-Product, .CGR, .3DXML</td>
<td>CATIA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLLADA 1.4.1</td>
<td>.dae</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DirectX writes 2.0</td>
<td>.x</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Continues on next page
### 1 Getting Started

#### 1.8 Libraries, geometries and CAD files

Continued

<table>
<thead>
<tr>
<th>Format</th>
<th>File extensions</th>
<th>Option required</th>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>DXF/DWG</td>
<td>.dxf, .dwg</td>
<td>AutoCAD</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>FBX</td>
<td>.fbx</td>
<td>-</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>IGES</td>
<td>.igs, .iges</td>
<td>IGES</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Inventor</td>
<td>.ipt, .iam</td>
<td>Inventor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JT</td>
<td>.jt</td>
<td>JT</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>LDraw</td>
<td>.ldr, .ldraw, .mpd</td>
<td>-</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>NX</td>
<td>.prt</td>
<td>NX</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>OBJ</td>
<td>.obj</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parasolid</td>
<td>.x_t, .xmt_txt,</td>
<td>Parasolid</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>.x_b, .xmt_bin</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pro/E / Creo</td>
<td>.prt, .prt.<em>, .asm, .asm.</em></td>
<td>Creo</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Solid Edge</td>
<td>.par, .asm, .psm</td>
<td>SolidEdge</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>SolidWorks</td>
<td>.sldprt, .sldasm</td>
<td>SolidWorks</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>STEP</td>
<td>stp, step, p21</td>
<td>STEP</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>STL, ASCII STL supported (binary STL not supported)</td>
<td>.stl</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VDA-FS</td>
<td>vda, vdafs</td>
<td>VDA-FS</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>VRML, VRML2 (VRML1 not supported)</td>
<td>wrl, vrml, vrml2</td>
<td>-</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Note**

Refer to the latest Release Notes 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 large models and improves the visual display for small models. 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 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

<table>
<thead>
<tr>
<th>Detail Level</th>
<th>Specify the level of detail required when importing geometries. Select Fine, Medium or Coarse as required.</th>
</tr>
</thead>
</table>
1.9 How to install RobotWare and Add-Ins

Installing RobotWare

RobotStudio *add-ins* are available in the RobotApp’s window. Access the RobotApps website to manually download and install additional content. The recommended method is to install software from RobotStudio.

1. Locate the `.rspak` file in the user directory.
2. Start RobotStudio and in the Add-Ins tab select Install Package.
3. Navigate to the required `.rspak` and open it. 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.
2. On successful installation, the following option gets added to the tool bar.
3 Select the FlexPendant option to open Virtual FlexPendant.
This page is intentionally left blank
2 Building Stations

2.1 Understanding stations and solutions

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.

Solution

Solutions add structure to the station data. It contains folders for structuring station data so as to keep related data together. Solution contains the station, that is, the station document is part of the solution structure. By default, RobotStudio provides solution folders. In the default solution folder structure, files of similar type are stored in folders. The solution structure is the recommended way for storing station files. In the solution structure there are dedicated folders for, stations, virtual controllers, libraries, geometry, RAPID programs, distribution packages, and backups.

For a new installation of RobotStudio, the solution folder contains the following subfolders:

<table>
<thead>
<tr>
<th>Name</th>
<th>Date modified</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backups</td>
<td>2019-05-20 07:06</td>
<td>File folder</td>
</tr>
<tr>
<td>Libraries</td>
<td>2019-05-20 06:57</td>
<td>File folder</td>
</tr>
<tr>
<td>SignalAnalyzer</td>
<td>2019-05-20 06:58</td>
<td>File folder</td>
</tr>
<tr>
<td>Stations</td>
<td>2019-05-20 11:57</td>
<td>File folder</td>
</tr>
<tr>
<td>Virtual Controllers</td>
<td>2019-05-20 06:57</td>
<td>File folder</td>
</tr>
<tr>
<td>Solution.rssln</td>
<td>2019-05-20 06:57</td>
<td>RobotStudio Solution file</td>
</tr>
</tbody>
</table>

- Backups: This folder holds controller backups during an autosave operation.
- Libraries: This folder holds solution related library files.
- Signal Analyzer: This folder holds recordings from the Signal Analyzer.
- Stations: This folder holds station files.
- Virtual Controllers: This folder holds virtual controller.
- Solution.rssln:
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 are 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.

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.

Document folders

ABB Library is part of the RobotStudio installation. ABB Library contains geometries of robots and other equipment. 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 Building Stations
2.2 Preparing the computer for hosting RobotStudio

Continued

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

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**.

![Image of File tab with New and Empty Station options]

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

### Autosave

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable autosave of RAPID</td>
<td>This check-box is selected by default. RAPID programs are saved automatically in every 30 seconds.</td>
</tr>
<tr>
<td>Enable autosave of station</td>
<td>Unsaved stations are saved automatically at the interval specified in the minute interval box.</td>
</tr>
<tr>
<td>Enable automatic backup of station files</td>
<td>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 Solution.</td>
</tr>
<tr>
<td>Enable automatic backup of controllers in solution</td>
<td>Select this option to backup the virtual controllers of a solution when saving the station. The backups are stored in the Backups folder of the corresponding solution.</td>
</tr>
</tbody>
</table>

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

### Files & Folders

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Documents location</td>
<td>Shows the default path to the project folder.</td>
</tr>
<tr>
<td>Solutions location</td>
<td>Shows the default path to the solutions folder.</td>
</tr>
<tr>
<td>...</td>
<td>To browse to the project folder, click the browse button.</td>
</tr>
<tr>
<td>Automatically create document subfolders</td>
<td>Select this check box to enable the creation of individual sub-folders for document types.</td>
</tr>
<tr>
<td>minute interval</td>
<td>Specify the interval between the savings when using Autosave in this box.</td>
</tr>
<tr>
<td>Document Locations</td>
<td>Launches the Document Locations dialog box.</td>
</tr>
<tr>
<td>Clear Recent Stations and Controllers</td>
<td>Clears the list of recently accessed stations and controllers.</td>
</tr>
<tr>
<td>Additional distribution package location</td>
<td>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&lt;user name&gt;\AppData\Local\ABB\Distribution-Packages. The location can be customized by entering a search path here.</td>
</tr>
<tr>
<td>Download packages to this location</td>
<td>Select this check box to download distribution packages to the user defined location instead of the default folder.</td>
</tr>
<tr>
<td>Unpacked RobotWare Location</td>
<td>Shows the default path to the unpacked RobotWare folder.</td>
</tr>
<tr>
<td>Media Pool for RobotWare 5.x</td>
<td>This is where RobotStudio searches for RobotWare 5.xx media pools.</td>
</tr>
</tbody>
</table>
2 Building Stations

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 work object.
7. Define paths and targets to create a robot program graphically.
8. Synchronize to RAPID to create the RAPID program.

Creating a solution with an empty station

1. Click the File tab. The RobotStudio Backstage view appears, click New.
2. Under Stations, click Solution with Empty Station.
3. Enter the name of the solution in the Solution Name box and then browse and select the destination folder in the Location box. The default location of the solution is C:\User\<user name>\Documents\RobotStudio\Solutions.

   The included station gets the name of the solution.

   **Note**

   The solution folder helps in keeping a well-structured folder system that is easy to navigate through.

4. Click Create.

   The new solution gets created. RobotStudio saves this solution 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. A robot which is not connected to a controller cannot be programmed, hence configure a virtual controller for the robot.

Note

While connecting a real controller to RobotStudio over the network, the controller presents a self-signed security certificate which must be trusted by the operating system of the host computer. To ensure a trusted connection, it is recommended to install a trusted certificate.

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.

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 Solutions Library to select the predefined solutions.
5. Click Locations to open the Document Locations window.
6. Click Browse for Library to select the saved library files.
A robot which is not connected to a controller cannot be programmed, 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. 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. Add mechanisms such as tracks or positioners to your station before creating a virtual controller.
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 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:

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

The Controller Options page appears.

9. 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.

Continues on next page
• 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.

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.

Note

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

2.6 Synchronizing to virtual controller to create a RAPID program

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 programmed, 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 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 **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
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:

1. Click Next.

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

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

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**.
2 Building Stations

2.10 Configuring external axis

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:

<table>
<thead>
<tr>
<th>Combination</th>
<th>Positioner type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>One IRB (Positioner in same task)</td>
<td>Y</td>
</tr>
<tr>
<td>One IRB (Positioner in separate task)</td>
<td>Y</td>
</tr>
<tr>
<td>Two IRB (Positioner in separate task)</td>
<td>Y</td>
</tr>
<tr>
<td>One IRB on Track Motion (Positioner in same task)</td>
<td>Y</td>
</tr>
<tr>
<td>One IRB on Track Motion (Positioner in separate task)</td>
<td>N</td>
</tr>
</tbody>
</table>

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

The following table shows various RobotWare 6 configurations:

<table>
<thead>
<tr>
<th></th>
<th>Single system</th>
<th>MultiMove system</th>
<th>Track motion (non-dynamic model)</th>
<th>Track motion (Dynamic model)</th>
<th>MU/MTD 1</th>
<th>MU/MTD 2</th>
<th>MU/MTD 3</th>
<th>IRBP L1</th>
<th>IRBP L2</th>
<th>IRBP A1</th>
<th>IRBP A2</th>
<th>IRBP C</th>
<th>IRBP B</th>
<th>IRBP K</th>
<th>IRBP R</th>
<th>No drives</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Track</strong></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MU/MTD + IRBP</strong></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>IRBP only</strong></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Continues on next page
2 Building Stations

2.10 Configuring external axis
Continued

| 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 Edit System.
   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

2.11.1 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.
2 Building Stations

2.11.2 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 MultiMove 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.

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. When you have set up the robots and paths, continue testing the MultiMove and then tune the motion properties, if necessary.
2.11.3 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.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examine the robots’ positions</td>
<td>Click <em>Pause</em> and then use the arrow buttons to move to one <em>target</em> at a time.</td>
</tr>
<tr>
<td>for critical targets.</td>
<td></td>
</tr>
<tr>
<td>Jog the robots to new start</td>
<td>Change in positions require new paths and configurations. In most cases,</td>
</tr>
<tr>
<td>positions.</td>
<td>positions near the robots’ joint limits shall be avoided.</td>
</tr>
<tr>
<td>Go to the <em>Motion Behavior</em></td>
<td>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.</td>
</tr>
<tr>
<td>tab and remove constraints.</td>
<td></td>
</tr>
</tbody>
</table>
2.11.4 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.

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.

<table>
<thead>
<tr>
<th>Joint Influence</th>
<th>Select Robot</th>
<th>Select the robot’s joints to constrain in this box.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joints for Robot</td>
<td>Displays the robot’s joints and their constraint weights. Each joint is presented in its own row.</td>
<td></td>
</tr>
</tbody>
</table>
### 2 Building Stations

#### 2.11.4 Tuning the motion behavior

Continued

**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.

<table>
<thead>
<tr>
<th><strong>TCP Constraints</strong></th>
<th><strong>Active TCP</strong></th>
<th>This grid displays the position and rotations of the TCP together with their constraint weights.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enable</strong></td>
<td>Select this check box to activate the constraint for this TCP pose.</td>
<td></td>
</tr>
<tr>
<td><strong>Pose</strong></td>
<td>Displays the TCP pose that is affected by the constraint.</td>
<td></td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td><strong>Influence</strong></td>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Tool Tolerance</strong></th>
<th><strong>Enable</strong></th>
<th>Select this check box to activate the tolerance for this tool pose.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pose</strong></td>
<td>Displays the tool pose that is affected by the constraint.</td>
<td></td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td>Specify the pose value to apply the tolerance around.</td>
<td></td>
</tr>
<tr>
<td><strong>Influence</strong></td>
<td>Specify the size of the tolerance. 0 means no deviation is allowed, while 100 means all deviations are allowed.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Tool Offset</strong></th>
<th><strong>Enable</strong></th>
<th>Select this check box to activate the offset for this tool pose.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pose</strong></td>
<td>Displays the tool pose that is affected by the offset setting.</td>
<td></td>
</tr>
<tr>
<td><strong>Offset</strong></td>
<td>Specify the value of the offset here.</td>
<td></td>
</tr>
</tbody>
</table>

**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 alternative axes.

1. On the Home tab, click the Motion Behavior tab.
2. Expand the Joint Influence group by clicking its title bar.
3. 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.
4. 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.

*Continues on next page*
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 Simulation tab, click the Motion Behavior tab.
2 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.
3 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.
4 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 Simulation tab, click the Motion Behavior tab.
2 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.
3 For each offset you want to set, select the Enable check box.
4 In the Value column, specify the allowed deviation.
5 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 Simulation tab, click the Motion Behavior tab.
2 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.
3 For each offset you want to set, select the Enable check box.
4 In the Offset column, specify the offset distance.
2.11.5 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, click Create Paths tab.
2. Expand the Settings group by clicking on its title bar.
3. Optionally, change the naming settings in the following boxes:

<table>
<thead>
<tr>
<th>Box</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start ID</td>
<td>Specify the first ID number for the synchronization of the instructions for the robots.</td>
</tr>
<tr>
<td>ID step index</td>
<td>Specify the increment between ID numbers.</td>
</tr>
<tr>
<td>Sync ident prefix</td>
<td>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.</td>
</tr>
<tr>
<td>Task list prefix</td>
<td>Specify a prefix for the tasklist variable, which identifies the tasks for the tool robot and the work piece robot to synchronize.</td>
</tr>
</tbody>
</table>

4. Expand the WP Robot Settings group by clicking on its title bar and then check the settings in the following boxes:

<table>
<thead>
<tr>
<th>Box</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WP Workobject</td>
<td>Specify the workobject to which the targets generated for the workpiece robot shall belong.</td>
</tr>
<tr>
<td>WP TCP</td>
<td>Specify which tooldata the workpiece shall use when reaching its targets.</td>
</tr>
<tr>
<td>Path prefix</td>
<td>Specify a prefix for the generated paths.</td>
</tr>
<tr>
<td>Target prefix</td>
<td>Specify a prefix for the generated targets.</td>
</tr>
</tbody>
</table>

5. Expand the Generate path group by clicking on its title bar and then click Create Paths.
Additional actions

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

The main actions for programming MultiMove 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.

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

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.

<table>
<thead>
<tr>
<th>RS-WCS</th>
<th>World coordinate system in RobotStudio</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC-WCS</td>
<td>World coordinate system as defined in the robot controller. It corresponds to the task frame of RobotStudio.</td>
</tr>
<tr>
<td>BF</td>
<td>Robot Base Frame</td>
</tr>
<tr>
<td>TCP</td>
<td>Tool Center Point</td>
</tr>
<tr>
<td>P</td>
<td>Robot target</td>
</tr>
<tr>
<td>TF</td>
<td>Task Frame</td>
</tr>
<tr>
<td>Wobj</td>
<td>Workobject</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>RS-WCS</th>
<th>World coordinate system in RobotStudio</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP(R1)</td>
<td>Tool Center Point of robot 1</td>
</tr>
<tr>
<td>TCP(R2)</td>
<td>Tool Center Point of robot 2</td>
</tr>
<tr>
<td>BF(R1)</td>
<td>Base Frame of robot system 1</td>
</tr>
<tr>
<td>BF(R2)</td>
<td>Base Frame of robot system 2</td>
</tr>
<tr>
<td>P1</td>
<td>Robot target 1</td>
</tr>
<tr>
<td>P2</td>
<td>Robot target 2</td>
</tr>
<tr>
<td>TF1</td>
<td>Task Frame of robot system 1</td>
</tr>
<tr>
<td>TF2</td>
<td>Task Frame of robot system 2</td>
</tr>
<tr>
<td>Wobj</td>
<td>Workobject</td>
</tr>
</tbody>
</table>

Continues on next page
MultiMove Coordinated systems

The *MultiMove* functions help you create and optimize programs for MultiMove systems where one robot or *positioner* holds the workpiece 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 does not allow *task frames* of the controller to be separated.

<table>
<thead>
<tr>
<th></th>
<th>World coordinate system in RobotStudio</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP(R1)</td>
<td>Tool Center Point of robot 1</td>
</tr>
</tbody>
</table>

Continues on next page
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.
Workobject coordinate system

The **work object** 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 **work object**. If no other work object is specified, the targets will be related to the default **Wobj0**, which always coincides with the world coordinate system of the robot.

Using work objects provide the chance to easily adjust robot programs with an offset, if the location of the work piece has been changed. Thus, work objects 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 work object.

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

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 work object. Here the user frame is positioned at the table or fixture and the object frame at the work piece.

![Coordinate System Diagram](image-url)

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 *work object* 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.

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.
3 Programming robots in the 3D environment

3.2 Robot axis configurations

Continued

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:

\[
[0, -1, 2, 1]
\]

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 \texttt{ConfJ} and \texttt{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 \texttt{offline}, you must assign a configuration to each target if the program shall be executed with configuration control.
3.3 Creating a work object

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 work object.
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 work object 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 XYX) 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 work object.
6. Click Create. The work object will be created and displayed under the Targets node under the robot node in the Paths&Targets browser.
3 Programming robots in the 3D environment

3.4 Creating a path with targets and move instructions

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 Simulation tab, from the Path Programming group, click Path and then click Empty Path.
3. To set the correct motion properties for the targets, select the active process in the Change Active Process box in the Elements toolbar.
4. If the active template is set to MoveAbsJoint, then:
   - A target that is dragged into a path will be converted into a joint target (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. From the Home menu, in the Path Programming group, select the type of move instruction to create.
3. Click Add to Path.

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

<table>
<thead>
<tr>
<th>Select or enter values in</th>
<th>to</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinDist</td>
<td>Set the minimum distance between the generated points. That is, points closer than the minimum distance are filtered.</td>
</tr>
<tr>
<td>Tolerance</td>
<td>Set the maximum deviation from the geometric description allowed for the generated points.</td>
</tr>
<tr>
<td>MaxRadius</td>
<td>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.</td>
</tr>
<tr>
<td>Linear</td>
<td>Generate a linear move instruction for each target.</td>
</tr>
<tr>
<td>Circular</td>
<td>Generate circular move instructions where the selected edges describe circular segments.</td>
</tr>
<tr>
<td>Constant</td>
<td>Generate points with a Constant distance</td>
</tr>
<tr>
<td>End Offset</td>
<td>Set the specified offset away from the last target.</td>
</tr>
<tr>
<td>Start Offset</td>
<td>Sets the specified offset away from the first target.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Select or enter values in</th>
<th>to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approach</td>
<td>Generate a new target at a specified distance from the first target.</td>
</tr>
<tr>
<td>Depart</td>
<td>Generates a new target at a specified distance from the last target.</td>
</tr>
</tbody>
</table>

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 *work object*.

6 Click **Close**.
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 work object 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 work object.
   - Set Robot holds workobject to True.
   - In Moved by mechanical unit, select the robot, for example, ROB_1.
   - Set Programmed to False.
   If the work object 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 Programming robots in the 3D environment

3.7 Define arm configurations for the targets

3.7 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, select Configurations and then 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.
3.8 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, the Select Robot configuration dialog box is displayed.

3 In the Select Robot 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.
3 Programming robots in the 3D environment

3.9 Generating the RAPID program

3.9 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.
3.10 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 Synchronize to RAPID to update the RAPID program.
This page is intentionally left blank
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 real controller, offline on a virtual controller, or standalone programs. Use RAPID editor to view and edit programs loaded into a robot controller, both real 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 may be 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 (:), equals 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
4 Understanding RAPID editor

4.1 Working with RAPID editor

Continued

- **Cut, copy, paste and drag and drop** - These standard commands for clipboard handling of text are supported.
- **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:RAPID 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
4 Understanding RAPID editor

4.2 RAPID editor intellisense

Continued

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 real 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.
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
Loading a RAPID program
You can load a RAPID program to a real 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.
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 Understanding RAPID editor

4.4 Adding code snippets

### 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/](http://msdn.microsoft.com/).

**Inserting a code snippet in a RAPID module**

1. Click the File tab.
2. 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 Snippet.

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

```rapid
PROC myProcedure(switch doThis | switch doThat, INOUT num numRepeats, PERS num dataList(*))

ENDPROC
```
4 Understanding RAPID editor

4.5 Inserting instructions from the list

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 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 real 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...
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\<username>\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 `robbarget`, `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
   ```rapid
   PROC MyMoveL(robbarget ToPoint, PERS tooldata Tool, wobjdata Wobj)
   MoveL ToPoint, v100, fine, Tool, Wobj:=wobj0;
   ENDPROC
   ```

   Example 2
   ```rapid
   PROC MyMoveJ(robbarget 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 `robbarget` must be named `ToPoint`, incase of circular instruction the `via point` must be named `ViaPoint` according to the naming convention.
2 Open Instruction Template window.

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

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.
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.

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 robot targets. The edits made in the graphical RAPID program gets updated in the corresponding RAPID Editor and the RAPID Data Editors 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\<username>\Documents\RobotStudio\Instruction Templates folder in the computer.
The search path depends on how the RobotStudio documents folder is configured.

2 Open the RAPID Path Editor to view and edit the custom move instruction in RobotStudio like a standard move instruction.
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, 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 robtargs 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, robtargs 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
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.
- 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 Testing and Debugging RAPID

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.

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.
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:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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 Setup Simulation pane.
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,

<table>
<thead>
<tr>
<th>Click...</th>
<th>to...</th>
</tr>
</thead>
</table>
| **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 Station 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 timestep. |
| **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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform collision detection</td>
<td>Select if collision detection is to be performed during simulation or always. Default value: always.</td>
</tr>
<tr>
<td>Pause/stop simulation at</td>
<td>Select this check box if you want the simulation to stop at a collision or at a near miss. Default value: cleared.</td>
</tr>
<tr>
<td>collision</td>
<td></td>
</tr>
<tr>
<td>Log collisions to Output</td>
<td>Select this check box if you want the collisions to be logged to the output window. Default value: selected.</td>
</tr>
<tr>
<td>window</td>
<td></td>
</tr>
<tr>
<td>Log collisions to file</td>
<td>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.</td>
</tr>
</tbody>
</table>
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 the 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:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🔴 <img src="image" alt="xx050033" /></td>
<td>Active. Collisions between objects in this set will be detected.</td>
</tr>
<tr>
<td>🟢 <img src="image" alt="xx050007" /></td>
<td>Not active. Collisions between objects in this set will not be detected.</td>
</tr>
</tbody>
</table>

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.
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:

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Collisions between objects in this set will be detected.</td>
</tr>
<tr>
<td>Near miss (mm)</td>
<td>Specifies the maximum distance between the objects to be considered a near miss.</td>
</tr>
<tr>
<td>Highlight colliding</td>
<td>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.</td>
</tr>
<tr>
<td>Collision color</td>
<td>Displays the collision in the selected color.</td>
</tr>
<tr>
<td>Near miss color</td>
<td>Displays the near miss in the selected color.</td>
</tr>
<tr>
<td>Show markup at collision</td>
<td>Shows markup at collision or near miss.</td>
</tr>
<tr>
<td>Detect collisions between invisible objects</td>
<td>Detect collisions even if the objects are invisible.</td>
</tr>
</tbody>
</table>
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.

In RobotWare 6, the option Collision Detection must be selected to enable this functionality.

This feature is available for all six and seven axis backwards bending robots, supported by the standard IRC5 controller.

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.

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

<table>
<thead>
<tr>
<th>Select the option</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station object</td>
<td>Add an existing object or modify its properties</td>
</tr>
<tr>
<td>Primitive</td>
<td>Add an object and modify its properties</td>
</tr>
<tr>
<td>Smart Gripper fingers</td>
<td>Add smart gripper fingers (only applicable for YuMi)</td>
</tr>
<tr>
<td>Load geometry</td>
<td>Add a CAD geometry and modify its properties</td>
</tr>
</tbody>
</table>

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

4 Click OK, to add the object to the Objects list.

Note

A maximum of 10 objects can be added.
To easily configure multiple objects, they can be paired.

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

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

7 Select the **Exclude from collision check** checkbox, to exclude the paired object from the collision check.
8. Select the **Override safety distance** checkbox, to override the preset safety distance.

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

10. Click **Upload to Controller**, to upload the configuration to the real controller.

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

---

### Limitations

**Collision Avoidance** is a function included in the option **Collision Detection**. **Collision Avoidance** can only be used by six and seven axis serial link robots (bending backwards). It is supported by robots with track motion and single axis **positioner** (L-type).

When jogging, **Collision Avoidance** will not be triggered if used together with responsive jogging. The system parameter **Jog Mode** must be changed to **Standard**.

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

!! **CAUTION**

**Collision Avoidance** shall not be used for safety of personnel.
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. 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 System 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.

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.

<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select System. Select the required virtual controller to view the signals.</td>
</tr>
<tr>
<td>2</td>
<td>Filter type. Select the type of filter to use.</td>
</tr>
</tbody>
</table>
### 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.

### 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.

### 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.

### Edit Lists
Click this button to create or edit lists of favorite signals.

### 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

<table>
<thead>
<tr>
<th>Filter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Board</td>
<td>Displays all signals on a specific board. To select a board, use the Filter Specification list.</td>
</tr>
<tr>
<td>Group</td>
<td>Displays a group input or group output signal. To select a group, use the Filter Specification list.</td>
</tr>
<tr>
<td>User List</td>
<td>Displays all signals in a favorite list. To select a list, use the Filter Specification list.</td>
</tr>
<tr>
<td>Digital Inputs</td>
<td>Displays all digital input signals.</td>
</tr>
<tr>
<td>Digital Outputs</td>
<td>Displays all digital output signals.</td>
</tr>
<tr>
<td>Analog Inputs</td>
<td>Displays all analog input signals.</td>
</tr>
<tr>
<td>Analog Outputs</td>
<td>Displays all analog output signals.</td>
</tr>
</tbody>
</table>

### 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 Simulating programs

6.5 Simulation time measurement

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. 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.
4. Click Add.
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 Monitoring group, click TCP Trace. The TCP trace browser opens.
2 Select Enable TCP Trace check box to enable tracing.
3 Select Color by signal check box and then click .. button. The Select Signal dialog opens.
4 Open the Mechanical Units node and then select Speed In Current Wobj in the TCP node and click OK.
5 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.

6 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 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.

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

6.6 Understanding TCP trace

Continued

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.
7 Advanced simulation using smart components and physics

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 select Edit Component from the context menu. 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
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.

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.
# 7 Advanced simulation using smart components and physics

## 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.

The pack&go files for this infeeder station is available in `C:\Program Files (x86)\ABB\RobotStudio\Samples\Stations`. This folder contains two pack&go files, `SC demo station finished.rspag` and `SC demo station start.rspag`. Open the `SC demo station start.rspag` before starting this example.

### 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`.
3. In the Smart Component editor, click the Compose tab and then click the Add component button.

Continues on next page
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.

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.

5 In the **Smart Component** editor, click **Add component** and from the context menu, select **Signals and Properties** and then add **LogicGate**.

Continues on next page
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.

<table>
<thead>
<tr>
<th>Source Object</th>
<th>Source Signal</th>
<th>Target Object</th>
<th>Target Signal or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LineSensor</td>
<td>SensorOut</td>
<td>LogicGate [NOT]</td>
<td>InputA</td>
<td>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.</td>
</tr>
<tr>
<td>LogicGate [NOT]</td>
<td>Output</td>
<td>Source</td>
<td>Execute</td>
<td>The Source upon receiving the signal from LogicGate, triggers Execute signal to create a new box.</td>
</tr>
</tbody>
</table>

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.

With this option, the tooldata of the Vacuum Tool will be visible when it is attached to the robot.

Continues on next page
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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Signal Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>diAttach</td>
<td>DigitalInput</td>
<td>0</td>
<td>This signal gets active when the robot sends the doVacuum signal to the Vacuum Tool.</td>
</tr>
<tr>
<td>doAttached</td>
<td>DigitalOutput</td>
<td>0</td>
<td>This signal gets active when the Vacuum Tool picks the box, or when the LineSensor in the Vacuum Tool detects the box.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Source Object</th>
<th>Source Property</th>
<th>Target Object</th>
<th>Target Property or Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LineSensor_2</td>
<td>Sensed-Part</td>
<td>Attacher</td>
<td>Child</td>
<td>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.</td>
</tr>
<tr>
<td>Attacher</td>
<td>Child</td>
<td>Detacher</td>
<td>Child</td>
<td>The Detacher's child property is connected to the Attacher's child property such that the Detacher detaches the object the attacker is attached to.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Source Object</th>
<th>Source Signal</th>
<th>Target Object</th>
<th>Target Signal or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC_VacuumTool</td>
<td>diAttach</td>
<td>LineSensor_2</td>
<td>Active</td>
<td>When the input signal diAttach of the SC_VacuumTool is 1, then the Line Sensor gets activated.</td>
</tr>
<tr>
<td>LineSensor_2</td>
<td>SensorOut</td>
<td>Attacher</td>
<td>Execute</td>
<td>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.</td>
</tr>
<tr>
<td>SC_VacuumTool</td>
<td>diAttach</td>
<td>LogicGate2[NOT]</td>
<td>InputA</td>
<td>When the sensor senses the object the diAttach is high, this signal is send to the LogicalGate2[NOT] which inverts the signal.</td>
</tr>
<tr>
<td>LogicGate2[NOT]</td>
<td>Output</td>
<td>Detacher</td>
<td>Execute</td>
<td>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.</td>
</tr>
<tr>
<td>Attacher</td>
<td>Executed</td>
<td>LogicalSRLatch</td>
<td>Set</td>
<td>When the attacher attaches an object it sends the Set signal to the LogicalSRLatch.</td>
</tr>
<tr>
<td>Detacher</td>
<td>Executed</td>
<td>LogicalSRLatch</td>
<td>Reset</td>
<td>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_VacuumTool. The output signals from the Attacher and Detacher are infrequent. Using SR Latch ensures a steady output from the SC_VacuumTool.</td>
</tr>
</tbody>
</table>
The output signal from the Latch is send to the output signal of the *SC_VacuumTool*.

<table>
<thead>
<tr>
<th>Source Object</th>
<th>Source Signal</th>
<th>Target Object</th>
<th>Target Signal or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogicalSR-Latch</td>
<td>Output</td>
<td>SC_VacuumTool</td>
<td>doAttached</td>
<td>The output signal from the Latch is send to the output signal of the <em>SC_VacuumTool</em>.</td>
</tr>
</tbody>
</table>

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*.

<table>
<thead>
<tr>
<th>Source Object</th>
<th>Source Property</th>
<th>Target Object</th>
<th>Target Property or Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlaneSensor</td>
<td>SensedPart</td>
<td>Sink</td>
<td>Object</td>
<td>This binding connects the <em>SensedPart</em> property of the <em>Plane Sensor</em> to the <em>Object</em> property of the <em>Sink</em> such that it deletes the sensed part.</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Source Object</th>
<th>Source Signal</th>
<th>Target Object</th>
<th>Target Signal or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlaneSensor</td>
<td>SensorOut</td>
<td>LogicGate</td>
<td>InputA</td>
<td>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.</td>
</tr>
<tr>
<td>LogicGate [NOP]</td>
<td>Output</td>
<td>Sink</td>
<td>Execute</td>
<td>The Sink upon receiving the signal from LogicGate, triggers Execute signal to delete the box.</td>
</tr>
</tbody>
</table>

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.
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.

![Image of IRB4600 I/O Signals]

### 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.

<table>
<thead>
<tr>
<th>Source Object</th>
<th>Source Signal</th>
<th>Target Object</th>
<th>Target Signal or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC_InFeeder</td>
<td>doBoxInPos</td>
<td>IRB4600</td>
<td>diBoxInPos</td>
<td>When the Line Sensor detects the box, the Infeeder sends the output signal <code>diBoxInPos</code> to the virtual controller.</td>
</tr>
<tr>
<td>IRB4600</td>
<td>doVacuum</td>
<td>SC_VacuumTool</td>
<td>diAttach</td>
<td>The virtual controller sends the <code>doVacuum</code> signal to the <code>SC_VacuumTool</code>. The <code>SC_VacuumTool</code> simulates the vacuum and attaches the part.</td>
</tr>
<tr>
<td>SC_VacuumTool</td>
<td>doAttached</td>
<td>IRB4600</td>
<td>diVacuum</td>
<td>The <code>SC_VacuumTool</code> grips the object and sends the signal to the virtual controller.</td>
</tr>
</tbody>
</table>

### 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*
7 Advanced simulation using smart components and physics

7.2 The Infeeder example

Continued

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 Advanced simulation using smart components and physics

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, in Rigid Bodies group, click Material. The Physics Material 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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Density</td>
<td>Mass per volume.</td>
</tr>
<tr>
<td>Young's modulus</td>
<td>A mechanical property that measures the stiffness of a solid material.</td>
</tr>
<tr>
<td>Poisson's ratio</td>
<td>Poisson's ratio is a measure of the contraction that happens when an object is stretched.</td>
</tr>
<tr>
<td>Coefficient of restitution</td>
<td>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.</td>
</tr>
<tr>
<td>Roughness (friction)</td>
<td>Friction is the force resisting the relative motion of solid surfaces sliding against each other.</td>
</tr>
<tr>
<td>Cable properties</td>
<td>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)</td>
</tr>
</tbody>
</table>

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.
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 built upon elementary joints or links that define 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 SmartComponent

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:

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).
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.

Continues on next page
7 Import the Robot and Station symbols into the shared memory and perform the following setting.

- **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.
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. 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}.

<table>
<thead>
<tr>
<th>Joint symbol naming convention</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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]]. |
| 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.

3. In the Shared Memory window, perform the following setting.

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.

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.
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.

- 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 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.
• 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.

<table>
<thead>
<tr>
<th>Robot symbol naming convention</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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**.

*Continues on next page*
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 I/O signals in SIMIT get auto-connected to the SIMITConnection Smart Component.

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.

Continues on next page
Procedure

This example requires an active virtual controller.

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 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.
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 real 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 Deploying and Sharing

8.2 Sharing a station

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 then browse to 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.

   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 Controller Systems 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 Deploying and Sharing

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 Deploying and Sharing

8.5 Creating a 3D animation of your simulation

8.5 Creating a 3D animation of your simulation

Overview

The Station 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

- The .NET Framework 4.5 must be installed on the playback computer.

Creating and loading a Station Viewer

1. On the File tab, click Share and select Save Station as Viewer
2. Select a suitable location and type in a filename, and click Save.
   - Select the option Show comments on startup and add text in the box. This comment gets displayed when the Station Viewer starts.
   - To save the simulation as Station Viewer, in the Simulation Control group, click the lower part of the Play button, and then select Record to Viewer. This simulation can be saved in *.exe, *.glb (gITF file) format.

   ![Note]
   Record to Viewer is enabled in Time Slice mode and it is disabled when the Free Run mode gets enabled.

3. To load a Station Viewer, double-click the package (.exe) file on the target computer.

   The results are displayed in the Output window and the embedded station file is automatically loaded and presented in a 3D view.

   ![Note]
   The Station Viewer inherits the default settings of RobotStudio, to customize these settings on the File tab, click Options to edit the settings.

Simulation

When you run a simulation, the movements and visibility of objects are recorded. This recording is optionally included in the Station Viewer. Simulation control buttons are enabled when the Station Viewer contains a recorded simulation.
8.6 Deploying a RAPID program to a real controller

Overview
Use transfer function to transfer RAPID programs that are created offline in a virtual controller to a real 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 real 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 real 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.
8 Deploying and Sharing

8.6 Deploying a RAPID program to a real controller

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, 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
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 Using System Builder for managing RW 5

9.1.1 About System Builder

Overview
This section describes how you create, build, modify and copy systems to run on virtual and real controllers. These systems may even be converted to boot media and downloaded to a real 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 accidentally 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 real 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.
9 Installing robot controller software

9.1.1 About System Builder

Continued

• Downloading to the real controller requires a direct connection from your computer to the service or Ethernet port of the controller.
9.1.2 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 Installing robot controller software

9.1.3 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.
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.
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
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 Installing robot controller software

9.1.4 Modifying a system

9.1.4 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
9 Installing robot controller software

9.1.4 Modifying a system

Continued

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
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 these steps:

1. From the System Builder dialog box, select the system and then click Delete.
9 Installing robot controller software

9.1.5 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.
9.1.6 Creating a system from backup

Overview

The Create System from Backup Wizard, which creates a new system from a controller system backup, is launched form 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.
9.1.7 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.

<table>
<thead>
<tr>
<th>You can select by using the...</th>
<th>if...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select controller from list option</td>
<td>the controller has been detected automatically.</td>
</tr>
<tr>
<td>Specify IP address or controller name option</td>
<td>your PC and the robot is connected to the same network. You can only use the controller name in DHCP networks.</td>
</tr>
<tr>
<td>Use service port option</td>
<td>your PC is directly connected to the controller's service port.</td>
</tr>
</tbody>
</table>

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?

<table>
<thead>
<tr>
<th>Yes</th>
<th>The controller restarts immediately and the downloaded system starts automatically.</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>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</td>
</tr>
<tr>
<td>Cancel</td>
<td>The downloaded system is removed from the controller.</td>
</tr>
</tbody>
</table>
9 Installing robot controller software

9.1.8 Examples using the System Builder when offline

9.1.8.1 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.
9 Installing robot controller software

9.1.8.1 A system with support for one robot and one positioner external axis

Continued

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
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 Installing robot controller software

9.1.8.2 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 additional 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.
9.2 Using Installation manager for managing RW 6 and RW 7

9.2.1 Installation Manager

9.2.1.1 About Installation Manager

Overview
This section describes how to create, modify, and copy systems to run on real and virtual controllers using the Installation Manager.

Note
Use Installation Manager to create and modify systems with RobotWare versions 6.0 and later. 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 real 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 real 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
Installation Manager 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 real 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 real controller requires a connection from the computer to the service or Ethernet port of the controller.
9 Installing robot controller software

9.2.1.2 Startup and settings

**Starting Installation Manager**

On the Controller tab, in the Configuration group click Installation Manager to start the Installation Manager application.

This window provides two options. Select Network to create systems for real 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 Installation Manager.
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 real 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.

<table>
<thead>
<tr>
<th>Buttons</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export settings</td>
<td>Click this button to export the current settings of the robot controller.</td>
</tr>
<tr>
<td>Import settings</td>
<td>Click this button to import settings to the robot controller. The current settings of the system will be cleared before performing this operation.</td>
</tr>
<tr>
<td>Add settings</td>
<td>Click this button to add settings to the current setup of the robot controller.</td>
</tr>
<tr>
<td>Revert</td>
<td>Click this button to revert to the current settings of the robot controller.</td>
</tr>
</tbody>
</table>
9.2.1.3 Building a new robot controller

Creating a new system for a real 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.


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.

   **Note**

   For RobotWare 7.0 systems, Add RobotOS, RobotControl and RobotWareInstallationutilities and click OK to create a package file, this package file can be used for multiple installations. RobotControl must be selected first.

   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.

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.

   **Note**

   For RobotWare 7.0 systems, the package gets created in the preferred folder.

   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 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**

For RobotWare 7.0 systems, Add *RobotOS*, *RobotControl* and *RobotWareInstallationUtilities* and click OK to create a package file, this package file can be used for multiple installations. *RobotControl* must be selected first.

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.

**Note**

For RobotWare 7.0 systems, the package gets created in the preferred folder.
9.2.1.4 Modifying a robot controller

Modifying a system for a real controller

1. In the Installtion 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 Installtion 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 real 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.
9.2.1.5 Copying a robot controller

Copying a system from a virtual controller to a real controller

To be able to copy a virtual controller to a real 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.
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 new from.
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.
9 Click Apply for the changes to take place.
9.2.1.6 Creating a robot controller from backup

Creating a robot controller from backup for a real 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 new from.
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 new from.
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.
9.2.1.7 Renaming a robot controller

You can rename a real 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 Installing robot controller software

9.2.2 A MultiMove system with two coordinated robots

9.2.2.1 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 checkbox. 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
1 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 real robot keys, the key sets the options. But since we are using a virtual key, we have to set the options manually.
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 Installing robot controller software

9.2.2.2 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.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Configuration area:</th>
<th>Configuration file:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>Communication protocols and devices</td>
<td>SIO.cfg</td>
</tr>
<tr>
<td>Controller</td>
<td>Safety and RAPID specific functions</td>
<td>SYS.cfg</td>
</tr>
<tr>
<td>I/O</td>
<td>I/O boards and signals</td>
<td>EIO.cfg</td>
</tr>
<tr>
<td>Man-machine</td>
<td>Functions to simplify working with the virtual controller</td>
<td>MMC.cfg</td>
</tr>
<tr>
<td>communication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motion</td>
<td>The robot and external axes</td>
<td>MOC.cfg</td>
</tr>
<tr>
<td>Process</td>
<td>Process specific tools and equipment</td>
<td>PROC.cfg</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Type</th>
<th>Instance</th>
<th>Parameters</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O</td>
<td>System Input</td>
<td>di2</td>
<td>Action: Start at main</td>
<td>Argument: Continuous</td>
</tr>
</tbody>
</table>

System parameters are configured using RobotStudio or the FlexPendant.

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.
10.2 Adding instances

Use the Configuration Editor to select a 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. 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.

4. 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 Working with system parameters

10.3 Copying an instance

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. In the Instance list, select one or several instances to copy. If you select several instances and they don't have the same value for all parameters, these parameters will have no default values in the new instances.

4. 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.

5. 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.
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 Working with system parameters

10.5 Save one configuration file

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.
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.
11 Monitoring robot signals

11.1 Signal analyzer

Overview

Signal Analyzer is used to record robot signals from real or virtual controllers. The version of Signal Analyzer used for real 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 Monitoring robot signals

11.2 Monitored signals

Signal analyzer can only monitor a group of selected signals. This selection varies for real and virtual controllers. The following table provides the list of signals for real and virtual controllers that can be monitored in signal analyzer.

<table>
<thead>
<tr>
<th>Category</th>
<th>Available signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller Signals</td>
<td>Total Motor Power</td>
</tr>
<tr>
<td></td>
<td>Total Power Consumption</td>
</tr>
<tr>
<td>EventLog</td>
<td>All domains</td>
</tr>
<tr>
<td>I/O System</td>
<td>All signals</td>
</tr>
<tr>
<td>Joint</td>
<td>J1-J6</td>
</tr>
<tr>
<td></td>
<td>Near Limit</td>
</tr>
<tr>
<td>Target</td>
<td>Fine Point</td>
</tr>
<tr>
<td></td>
<td>Target Changed, Tool Changed, Workobject Changed</td>
</tr>
<tr>
<td>TCP</td>
<td>Maximum Linear Acceleration in World</td>
</tr>
<tr>
<td></td>
<td>Orientation Q1-Q4 Current Workobject</td>
</tr>
<tr>
<td></td>
<td>Orientation Speed in Current Workobject</td>
</tr>
<tr>
<td></td>
<td>Pos X, Y, Z in Current Workobject</td>
</tr>
<tr>
<td></td>
<td>Robot Configuration cf1, cf4, cf6, cfx</td>
</tr>
<tr>
<td></td>
<td>Speed in Current Workobject</td>
</tr>
<tr>
<td>Smart Components</td>
<td>All signals</td>
</tr>
<tr>
<td>Physics</td>
<td>Cable tension, cable length and cable twist (only available for stations with cables)</td>
</tr>
<tr>
<td>Stop distance estimation</td>
<td>All signals, available only for virtual controllers.</td>
</tr>
<tr>
<td>Devices (polled)</td>
<td>CPU Load, CPU temperature and RAPID Memory.</td>
</tr>
</tbody>
</table>

Stop distance estimation

These signals show the stop distance of the robot during an emergency stop at a given time-step. For example, if the stop distance signal for joint J1 is 7 degrees at time t=48 ms, it means that J1 will move 7 degrees in positive direction if the robot is stopped at t=48 ms.

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 cannot be reused but it is burned off in the bleeder.

Continues on next page
For a virtual robot the signal is based on a nominal robot during typical conditions, for a real robot the signal is based on the torque for that particular robot in the actual conditions. For a real 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.

**Total Motor Energy**

The **Total Motor Energy** signal is the integration of the power over time.

**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 real 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 Monitoring robot signals

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.

Recording signals for a real controller
RobotStudio must be connected to a real 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\<username>\Documents\RobotStudio\SignalAnalyzer\Stations for a virtual controller and C:\Users\<username>\Documents\RobotStudio\SignalAnalyzer\Online for a real controller.
  ```

- In the Solution folder for the respective solution:
  
  ```
  C:\Users\<username>\Documents\RobotStudio\Solutions\<Solution 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.

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.

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.
This page is intentionally left blank
12 Jobs

12.1 Understanding jobs

Overview

RobotStudio online is designed to work with one *real 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.

<table>
<thead>
<tr>
<th>Groups</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Lists</td>
<td>A user-defined group of robot controllers for applying a job. Device lists can be reused between jobs.</td>
</tr>
<tr>
<td>Jobs</td>
<td>A command group containing various job options.</td>
</tr>
</tbody>
</table>

Device Lists group

The Device Lists group contains the following commands and controls.

<table>
<thead>
<tr>
<th>Commands</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Device List</td>
<td>Opens a new empty device list window for creating a new group of robot controllers.</td>
</tr>
<tr>
<td>Open Device List</td>
<td>Opens a previously saved device list to edit or review.</td>
</tr>
<tr>
<td>Save Device List</td>
<td>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.</td>
</tr>
<tr>
<td>Scan Subnet</td>
<td>Populates the active device list window with all controllers that Netscan finds on the local subnet.</td>
</tr>
</tbody>
</table>

Jobs group

The Jobs group contains the following commands and controls.

<table>
<thead>
<tr>
<th>Commands</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New job</td>
<td>Opens a new Job window.</td>
</tr>
</tbody>
</table>
| 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. |
12 Jobs

12.1 Understanding jobs

Continued

<table>
<thead>
<tr>
<th>Commands</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify</td>
<td>Verifies the status of the group of robot controllers.</td>
</tr>
<tr>
<td>Execute</td>
<td>Executes the user-selected action.</td>
</tr>
<tr>
<td>Pause</td>
<td>Temporarily stops an active action.</td>
</tr>
<tr>
<td>Resume</td>
<td>Resumes the paused action.</td>
</tr>
<tr>
<td>Cancel</td>
<td>Aborts the active job.</td>
</tr>
</tbody>
</table>

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.

Continues on next page
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.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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.

- **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 up to an optional time (in days). You can also specify the range for the error codes to include in the search.

- **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/TYP/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.

*Continues on next page*
12 Jobs

12.1 Understanding jobs

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search RAPID Data</td>
<td>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.</td>
</tr>
<tr>
<td>Search RAPID Text</td>
<td>Searches for lines that contain the specified text string. You can restrict the search to tasks or modules that match a certain name pattern.</td>
</tr>
<tr>
<td>Write File or Directory</td>
<td>Writes the selected file or directory to the specified target directory.</td>
</tr>
<tr>
<td>Read File or Directory</td>
<td>Reads the selected file or directory from HOME folder or from a task.</td>
</tr>
<tr>
<td>System Information</td>
<td>Reads the options, languages and media versions of the controllers.</td>
</tr>
<tr>
<td>Run External Tool</td>
<td>Invokes an external executable.</td>
</tr>
<tr>
<td></td>
<td>• External Tool Path: Location of the folder where the external tool is placed.</td>
</tr>
<tr>
<td></td>
<td>• Arguments: User specified arguments which the external tool passes, for example {SystemName}, {Network address},{Group} and so on.</td>
</tr>
<tr>
<td></td>
<td>• Timeout(s): Specifies command time out period.</td>
</tr>
<tr>
<td>Compare Folder</td>
<td>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.</td>
</tr>
<tr>
<td>Distribute Update Package</td>
<td>Copies the selected distribution package to the controller.</td>
</tr>
</tbody>
</table>

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.
- Use the Remove button to remove the selected Action.
- The Save function, Verify and Execute are not specific to the multi-action Job.
12.2 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.
12 Jobs

12.3 Creating a new job

12.3 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.
12.4 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.

3 Customize the file and then click Templates and then click Save Job and save the file.
12.5 Scheduling jobs

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.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create a Job and save it with a suitable name, for example Job1. For a default installation of RobotStudio, the job gets saved in <code>C:\Users\&lt;username&gt;\Documents\RobotStudio\JobTemplates</code> folder as a <code>.xml</code> file.</td>
</tr>
<tr>
<td>2</td>
<td>Open Notepad and type in <code>C:\Program Files (x86)\ABB\RobotStudio YY.xx\Bin\Addins\FleetManagement\runjob.exe C:\Users\&lt;username&gt;\Documents\RobotStudio\JobTemplates\Job1.xml/defaultcredentials</code>. The example assumes that RobotStudio is installed in the default location. A specific user name and password can be supplied with the options <code>/user:&lt;user name&gt;</code> and <code>/password:&lt;password&gt;</code>.</td>
</tr>
<tr>
<td>3</td>
<td>Save the <code>.txt</code> file with the <code>.cmd</code> extension.</td>
</tr>
<tr>
<td>4</td>
<td>Double-click the <code>.cmd</code> file to run the job. Log files and reports get generated and is available in the Jobs browser.</td>
</tr>
</tbody>
</table>

**Note**

You can schedule jobs using the Windows built-in Task Scheduler.
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.

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

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 of smaller graphical components in a design layout. Typical controls (sometimes referred as widgets or graphic components) include buttons, menus, images, and text fields.

Continues on next page
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
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.

<table>
<thead>
<tr>
<th>Parts</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Ribbon</td>
<td>Displays group of icons organized in a logical sequence of functions.</td>
</tr>
<tr>
<td>2 Project explorer</td>
<td>Shows the active screen project and lists the screens that are defined in the project. For more information,</td>
</tr>
<tr>
<td>3 Design area</td>
<td>Layout to design the screen with the available controls.</td>
</tr>
<tr>
<td>4 Output window</td>
<td>Displays information about the events that occur during ScreenMaker development.</td>
</tr>
<tr>
<td>5 ToolBox / Properties</td>
<td>Displays a list of available controls. For more information.. 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.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Group</th>
<th>Functions used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>Managing a ScreenMaker project. See Managing ScreenMaker Projects.</td>
</tr>
</tbody>
</table>
### Functions used for Group

<table>
<thead>
<tr>
<th>Group</th>
<th>Functions used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adding screen and application variables. See Managing screens and Managing Application Variables.</td>
</tr>
<tr>
<td>Build</td>
<td>Building a project. See Building a project.</td>
</tr>
<tr>
<td>Controller</td>
<td>Connecting and deploying to the controller. See Connecting to controller and Deploying to controller. Also for opening the Virtual FlexPendant.</td>
</tr>
<tr>
<td>Arrange</td>
<td>Re-sizing and positioning the controls on the design area.</td>
</tr>
<tr>
<td>Diagnosis</td>
<td>Detecting problems in the project and providing a diagnostic solution.</td>
</tr>
<tr>
<td>Close</td>
<td>Closing a project.</td>
</tr>
</tbody>
</table>

### 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.
ToolBox acts a container for holding all the available controls that can be placed on a screen.

The following table displays the GUI controls that can be dragged to the design area.

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActionTrigger</td>
<td>Allows to run a list of actions when either a signal or rapid data changes</td>
</tr>
<tr>
<td>BarGraph</td>
<td>Represents an analog value in a bar</td>
</tr>
<tr>
<td>Button</td>
<td>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.</td>
</tr>
<tr>
<td>CheckBox</td>
<td>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).</td>
</tr>
</tbody>
</table>

Continues on next page
<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComboBox</td>
<td>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.</td>
</tr>
<tr>
<td>CommandBar</td>
<td>Provides a menu system for a ScreenForm</td>
</tr>
<tr>
<td>ConditionalTrigger</td>
<td>Allows to define conditions while defining action triggers. An action is triggered, if there is any change in value of the data bound.</td>
</tr>
<tr>
<td>ControllerModeStatus</td>
<td>Displays the mode of the Controller (Auto - Manual)</td>
</tr>
<tr>
<td>DataEditor</td>
<td>Represents a text box control that can be used to edit the data.</td>
</tr>
<tr>
<td>Graph</td>
<td>Represents a control that plots data with lines or bars.</td>
</tr>
<tr>
<td>GroupBox</td>
<td>Represents a Windows control that displays a frame around a group of controls with an optional caption. It is a container used to group a set of graphic components. It usually has a title at the top.</td>
</tr>
<tr>
<td>LED</td>
<td>Displays a two states value, like a Digital Signal.</td>
</tr>
<tr>
<td>ListBox</td>
<td>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.</td>
</tr>
<tr>
<td>NumEditor</td>
<td>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.</td>
</tr>
<tr>
<td>NumericUpDown</td>
<td>Represents a spin box that displays numeric values.</td>
</tr>
<tr>
<td>Panel</td>
<td>Used to group collection of controls.</td>
</tr>
<tr>
<td>PictureBox</td>
<td>Represents a picture box control that displays images.</td>
</tr>
<tr>
<td>RadioButton</td>
<td>Allows to select only one of a predefined set of options.</td>
</tr>
<tr>
<td>RapidExecutionStatus</td>
<td>Displays the execution status of the Controller Rapid Domain (Running - Auto)</td>
</tr>
<tr>
<td>RunRoutineButton</td>
<td>Represents a Windows button control that calls a RapidRoutine when clicked</td>
</tr>
<tr>
<td>Switch</td>
<td>Displays and lets change a two states value, like a Digital Output Signal.</td>
</tr>
<tr>
<td>TabControl</td>
<td>Manages a set of tab pages.</td>
</tr>
<tr>
<td>TpsLabel</td>
<td>Very commonly used widget that displays text, a label is usually static, that is, it has no interactivity. A label generally identifies a nearby text box or other graphic component.</td>
</tr>
<tr>
<td>VariantButton</td>
<td>Used to change the values of RAPID variables or Application variables.</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Graphical component name panel</td>
<td>Displays the selected component, and lists the available components of the active design screen.</td>
</tr>
<tr>
<td>2 Properties window toolbar</td>
<td></td>
</tr>
<tr>
<td>1 Organizes table panel in categories</td>
<td></td>
</tr>
<tr>
<td>2 Organizes table panel alphabetically</td>
<td></td>
</tr>
<tr>
<td>3 Displays Properties in table panel</td>
<td></td>
</tr>
<tr>
<td>4 Displays Events in table panel</td>
<td></td>
</tr>
<tr>
<td>3 Table panel</td>
<td>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.</td>
</tr>
<tr>
<td>4 Information panel</td>
<td>Display information about a property or event.</td>
</tr>
</tbody>
</table>
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.
13 Screenmaker

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.
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:

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
### Action

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:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Drag a <strong>TpsLabel</strong> control from the <strong>ToolBox</strong> on to the design area.</td>
</tr>
<tr>
<td>2</td>
<td>You can set the values, setup events, configure data binding values and set the application values for a TpsLabel control.</td>
</tr>
<tr>
<td></td>
<td>- Set the values of a property in the <strong>Properties window</strong>.</td>
</tr>
<tr>
<td></td>
<td>- Set up the events, see <strong>Setup Events</strong>.</td>
</tr>
<tr>
<td></td>
<td>- To configure the data binding values, see <strong>Configuring Data Binding</strong>.</td>
</tr>
<tr>
<td></td>
<td>- To set the application variables, see <strong>Managing Application Variables</strong>.</td>
</tr>
<tr>
<td>3</td>
<td>You can set the option <strong>Allow Multiple States</strong> to true and change the property.</td>
</tr>
<tr>
<td></td>
<td>1 Click <strong>Allow Multiple States</strong>. The StatesEditor dialog box appears.</td>
</tr>
<tr>
<td></td>
<td>2 Click the check-box <strong>Allow Multi-States</strong>, select the properties to change from Properties For States and click OK.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Drag a Panel control from the <strong>ToolBox</strong> on to the design area.</td>
</tr>
<tr>
<td>2</td>
<td>You can add a group of controls to a panel.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Drag a ControllerModeStatus control from the ToolBox on to the design area.</td>
</tr>
</tbody>
</table>
| 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). This control is used.

Use this procedure to add a RapidExecutionStatus control:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Drag a RapidExecutionStatus control from the ToolBox on to the design area.</td>
</tr>
</tbody>
</table>
| 2    | You can set the values, setup events, configure data binding values, and set the application variables 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. |
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:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Drag a RunRoutineButton control from the ToolBox on to the design area.</td>
</tr>
</tbody>
</table>
| 2    | Click the smart tag on the RunRoutineButton and select one of the following Run-RoutineButtonTasks.  
      |   • 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:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 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. |
To add an event to a menu item, for example *menuItem1* on the command bar, use this procedure:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Go to the Properties window and select <em>menuItem1</em> from the drop-down list.</td>
</tr>
<tr>
<td>2</td>
<td>Click Events icon and then double-click the Click event. This opens the Events Panel dialog for the Click event.</td>
</tr>
<tr>
<td>3</td>
<td>Click Add Action from the Events Panel dialog. This opens a sub-list of actions.</td>
</tr>
<tr>
<td>4</td>
<td>Click an action from the sub-list of actions to add it to <em>menuItem1</em>'s Click event.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Drag a VariantButton control from the ToolBox on to the design area.</td>
</tr>
</tbody>
</table>
| 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. |
Conditional Trigger

The Conditional Trigger 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 procedure to add the Conditional Trigger control:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Drag a Conditional Trigger control from the ToolBox on to the design area.</td>
</tr>
</tbody>
</table>
| 2    | You can set the following Conditional Trigger 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:

<table>
<thead>
<tr>
<th>Screens</th>
<th>Open Screen</th>
</tr>
</thead>
</table>
| 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.
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.

```csharp
namespace SMDotNetMethods
{
    public class Methods
```
/// <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.
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,

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>application is visible to the left in the ABB Menu.</td>
</tr>
<tr>
<td>Right</td>
<td>application is visible to the right in the ABB Menu.</td>
</tr>
<tr>
<td>None</td>
<td>application is not visible at all in the ABB Menu.</td>
</tr>
</tbody>
</table>

   **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 default 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 real 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 real and virtual controllers:

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.
Deploying to controller

Use this procedure to deploy a ScreenMaker project to a real 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 real 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.
13 Screenmaker
13.3.1 Managing projects
Continued

For information on how to create a new widget project.
4 Connect to a real 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 real 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
246

Operating manual - RobotStudio
3HAC032104-001 Revision: AA
© Copyright 2019 ABB. All rights reserved.


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 and from RobotApps Repository. 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.

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>
</Widget>
```

Continues on next page
The production screen provides the flexibility to modify bindings of the widget. This is provided under Bindings tag as shown here:

```xml
<Bindings>
  <Binding PropertyName="led1.Value" BindingType="SIGNAL"
           DataName="MOTLMP" />
  <Binding PropertyName="button1.Text" BindingType="RAPID"
           DataName="T_ROB1/BASE/wobj0" />
</Bindings>
</Widget>
```

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 available in the Production.xml file to work with the Production Screen Environment.

```xml
<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.
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.

<table>
<thead>
<tr>
<th>Click...</th>
<th>to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove actual binding</td>
<td>remove the existing data binding.</td>
</tr>
<tr>
<td>Bind to a Controller object</td>
<td>select available data in the controller for binding.</td>
</tr>
<tr>
<td>Bind to an Application variable</td>
<td>select available data in the project Application Variables for binding.</td>
</tr>
<tr>
<td>Bind to an Array</td>
<td>select available RAPID array in the controller for binding.</td>
</tr>
</tbody>
</table>

Configuring data binding for different controls

Binding to an array can be done with the following controls:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataEditor</td>
<td>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.</td>
</tr>
</tbody>
</table>
| 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.
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:

```plaintext
PERS num n1:=0;
TASK PERS num n2:=0;
CONST num n3:=0;
```

The following binding is not supported:

```plaintext
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.
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
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.

Actions

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. 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.

```rapid
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.

**Continued on next page**
13 Screenmaker

13.4 Frequently asked questions

Continued
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.

<table>
<thead>
<tr>
<th>Job</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Produce</td>
<td>Welding the part</td>
</tr>
<tr>
<td>Service</td>
<td>Service the welding gun</td>
</tr>
<tr>
<td>Bull’s Eye</td>
<td>Calibrate with bull’s eye</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Action</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select the following options,</td>
</tr>
<tr>
<td></td>
<td>• FlexPendant Interface</td>
</tr>
<tr>
<td></td>
<td>• PC Interface</td>
</tr>
<tr>
<td>2</td>
<td>Load EIO.cfg and MainModule.mod files.</td>
</tr>
<tr>
<td></td>
<td>Select the following options,</td>
</tr>
<tr>
<td></td>
<td>By default:</td>
</tr>
<tr>
<td></td>
<td>• For Windows XP, the files can be found at C:\Documents and Settings&lt;user name&gt;\My Documents\RobotStudio\My ScreenMaker Projects\Tutorial</td>
</tr>
<tr>
<td></td>
<td>• For Windows 7, the files can be found at C:\Users&lt;user name&gt;\Documents\RobotStudio\My ScreenMaker Projects\Tutorial</td>
</tr>
</tbody>
</table>
### 13.5 Tutorial

#### Continued

<table>
<thead>
<tr>
<th>Action</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>The following signals are created after loading EIO.cfg file</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IO</th>
<th>Type</th>
<th>Description</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

| 4 | Create an empty station in RobotStudio with the system created in the previous step. |
| 5 | Launch ScreenMaker from RobotStudio. |
| 6 | Create a new ScreenMaker project. |
| 7 | Configure the Project properties. |
| 8 | Connect to the controller. |

Select the following options, 

1. Enter the project name as FlexArcGUI, and save it in the default location, C:\Users\<username>\Documents\RobotStudio\MyScreenMakerProjects\Tutorial. 
2. A new tab MainScreen is added to the Design Surface. 

To customize how the GUI should appear on the FlexPendant, modify the Project properties. 

The result appears in the output window.

Continues on next page
### Action 9
Create application variables (temporary variables) and configure them with the following data

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

### Action 10
Design the Main Screen.

### Action 11
Build and Deploy the project.

### Action 12
Open virtual FlexPendant and test the GUI

- 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.

---

**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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>14,45</td>
</tr>
<tr>
<td>Size</td>
<td>150,100</td>
</tr>
<tr>
<td>Title</td>
<td>Controller Status</td>
</tr>
<tr>
<td>BackColor</td>
<td>LightGray</td>
</tr>
</tbody>
</table>

2. Drag another GroupBox control from the General category; place it on the design surface and set the following values in the Properties window.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>14,170</td>
</tr>
<tr>
<td>Size</td>
<td>150,204</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>19,40</td>
</tr>
<tr>
<td>Size</td>
<td>44,44</td>
</tr>
<tr>
<td>BackColor</td>
<td>LightGray</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>80,40</td>
</tr>
<tr>
<td>Size</td>
<td>44,44</td>
</tr>
<tr>
<td>BackColor</td>
<td>LightGray</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>16,30</td>
</tr>
<tr>
<td>Size</td>
<td>131,20</td>
</tr>
<tr>
<td>Text</td>
<td>Parts Produced</td>
</tr>
<tr>
<td>BackColor</td>
<td>LightGray</td>
</tr>
<tr>
<td>Font</td>
<td>TpsFont10</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>16,56</td>
</tr>
<tr>
<td>Size</td>
<td>116,23</td>
</tr>
<tr>
<td>Value</td>
<td>Link to RAPID variable partsReady defined in the module MainModule.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>16,89</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>16,115</td>
</tr>
<tr>
<td>Size</td>
<td>116,23</td>
</tr>
<tr>
<td>Value</td>
<td>Link to RAPID variable <em>cycleTime</em> defined in the module <em>MainModule</em>.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>33,154</td>
</tr>
<tr>
<td>Size</td>
<td>85,34</td>
</tr>
<tr>
<td>Text</td>
<td>Reset</td>
</tr>
</tbody>
</table>

Perform the following for the Reset button in the *Part Status* group:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Double-click the button Reset. The Events Panel dialog box appears which is used to define the actions for Events.</td>
</tr>
</tbody>
</table>
| 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. |

Drag a PictureBox control from the General category; place it on the design surface and set the following values in the Properties window:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>177,28</td>
</tr>
<tr>
<td>Size</td>
<td>284,359</td>
</tr>
<tr>
<td>SizeMode</td>
<td>StretchImage</td>
</tr>
<tr>
<td>Image</td>
<td>FlexArcCell.GIF</td>
</tr>
</tbody>
</table>
11 Drag another PictureBox control from the General category; place it on the design surface and set the following values in the Properties window:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>237,31</td>
</tr>
<tr>
<td>Size</td>
<td>48,48</td>
</tr>
<tr>
<td>SizeMode</td>
<td>StretchImage</td>
</tr>
<tr>
<td>Image</td>
<td>RobotAtHome.GIF</td>
</tr>
<tr>
<td>AllowMultipleStates</td>
<td>True</td>
</tr>
<tr>
<td>SelectedStateValue</td>
<td>DI_RobotAtHome</td>
</tr>
<tr>
<td>States</td>
<td>Link State[0] to RobotAtHome_gray.GIF</td>
</tr>
<tr>
<td></td>
<td>Link State[1] to RobotAtHome.GIF</td>
</tr>
</tbody>
</table>

**Note**

You can find the graphic (.GIF) files at C:\MyDocuments\RobotStudio\My ScreenMaker Projects\Tutorial\Images.

12 Drag a Button control from the General category; place it on the design surface and set the following values in the Properties window:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>486,66</td>
</tr>
<tr>
<td>Size</td>
<td>116,105</td>
</tr>
<tr>
<td>Text</td>
<td>Start</td>
</tr>
<tr>
<td>Font</td>
<td>TpsFont20b</td>
</tr>
<tr>
<td>BackColor</td>
<td>LimeGreen</td>
</tr>
<tr>
<td>Enabled</td>
<td>Link to DI_RobotAtHome</td>
</tr>
</tbody>
</table>

Perform the following for the Start button:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>2</td>
<td>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.</td>
</tr>
<tr>
<td>3</td>
<td>In the Action Parameters dialog box, assign Rapid data to the following value and click OK. • T_ROB1.MainModule.JobProduce to Job.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>486,226</td>
</tr>
<tr>
<td>Size</td>
<td>116,105</td>
</tr>
<tr>
<td>Text</td>
<td>Stop</td>
</tr>
<tr>
<td>Font</td>
<td>TpsFont20b</td>
</tr>
<tr>
<td>BackColor</td>
<td>LimeGreen</td>
</tr>
<tr>
<td>Enabled</td>
<td>Link to DI_PRODUCE</td>
</tr>
</tbody>
</table>

Perform the following for the **Stop** button:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Double-click the button <strong>Stop</strong> or click the Smart tag and select <em>Define Actions when clicked</em>. The Events Panel dialog box appears which is used to define the actions for Events.</td>
</tr>
<tr>
<td>2</td>
<td>In the Events Panel dialog box, click Add Action; point to Rapid Data and select <em>Write a Rapid Data</em>. The Action Parameters dialog box appears.</td>
</tr>
</tbody>
</table>
| 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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>274,246</td>
</tr>
<tr>
<td>Size</td>
<td>111,47</td>
</tr>
<tr>
<td>Text</td>
<td>Bull’s Eye</td>
</tr>
<tr>
<td>Font</td>
<td>TpsFont14b</td>
</tr>
<tr>
<td>BackColor</td>
<td>Link to DI_RobotAtHome</td>
</tr>
<tr>
<td>Enabled</td>
<td>True</td>
</tr>
<tr>
<td>AllowMultipleStates</td>
<td>Select BackColor property from the StatesEditor dialog box</td>
</tr>
<tr>
<td>SelectedStates</td>
<td>DI_RobotAtBull’sEye</td>
</tr>
</tbody>
</table>
|States          | Link State[0] to **Red**  
  - Link State[1] to **Green** |

Perform the following for the **Bull’s Eye** button:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Double-click the button <strong>Bull’s Eye</strong> or click the Smart tag and select <em>Define Actions when clicked</em>. The Events Panel dialog box appears which is used to define the actions for Events.</td>
</tr>
<tr>
<td>2</td>
<td>In the Events Panel dialog box, click Add Action; point to Rapid Data and select <em>Write a Rapid Data</em>. The Action Parameters dialog box appears.</td>
</tr>
</tbody>
</table>
| 3    | In the Action Parameters dialog box, assign Rapid data to the following value and click OK.  
  - T_ROB1.MainModule.JobBulls to **JobBulls** |
Drag a Button control from the General category; place it on the design surface and set the following values in the Properties window:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>274,324</td>
</tr>
<tr>
<td>Size</td>
<td>111,47</td>
</tr>
<tr>
<td>Text</td>
<td>Service</td>
</tr>
<tr>
<td>Font</td>
<td>TpsFont14b</td>
</tr>
<tr>
<td>Enabled</td>
<td>Link to DI_RobotAtHome</td>
</tr>
<tr>
<td>AllowMultipleStates</td>
<td>True</td>
</tr>
<tr>
<td>SelectedStates</td>
<td>DI_RobotAtService</td>
</tr>
<tr>
<td>States</td>
<td>Link State[0] to Red</td>
</tr>
<tr>
<td></td>
<td>Link State[1] to Green</td>
</tr>
</tbody>
</table>

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
A Options

Common buttons

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply</td>
<td>Click this button to save all options in the current page.</td>
</tr>
<tr>
<td>Reset</td>
<td>Click this button to reset to the settings you had before this session all values that you have changed on the current page.</td>
</tr>
<tr>
<td>Default</td>
<td>Click this button to reset to their default values all settings on the current page.</td>
</tr>
</tbody>
</table>

Options: General: Appearance

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select application language</td>
<td>Select the language to be used. RobotStudio is available in the following seven languages: English, French, German, Spanish, Italian, Japanese, and Chinese (simplified).</td>
</tr>
<tr>
<td>Select color theme</td>
<td>Select the color to be used.</td>
</tr>
<tr>
<td>Default scale for zoomable windows</td>
<td>Defines the default scale to use for windows that are zoomable, for example, RAPID Editor, RAPID Data Editor and Configuration Editor.</td>
</tr>
<tr>
<td>Show ScreenTips</td>
<td>Select this check box to view ScreenTips.</td>
</tr>
<tr>
<td>Display Position Edit boxes with Red/Green/Blue background</td>
<td>Select the check box if you want to display the position boxes in the modify dialog boxes with colored background. Default value: selected.</td>
</tr>
<tr>
<td>Group related document windows under one tab</td>
<td>Select this check box to group related document window under one tab. Modifying this option requires a restart for the changes to take effect.</td>
</tr>
<tr>
<td>Restore hidden dialogs and messages</td>
<td>Select this check box to restore dialogs or messages which you may have hidden while using RobotStudio.</td>
</tr>
</tbody>
</table>

Options: General: Licensing

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable licensing</td>
<td>Reverts to Basic mode to use features that do not require activation.</td>
</tr>
<tr>
<td>View installed licenses</td>
<td>Click to view the licenses listed by feature, version, type, expiration date and status.</td>
</tr>
<tr>
<td>Activation Wizard</td>
<td>Click to activate RobotStudio license.</td>
</tr>
<tr>
<td>RobotStudio user experience program</td>
<td>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.</td>
</tr>
</tbody>
</table>

Options: General: Units

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity</td>
<td>Select the quantity for which you want to change the units.</td>
</tr>
<tr>
<td>Unit</td>
<td>Select the unit for the quantity.</td>
</tr>
<tr>
<td>Display decimals</td>
<td>Enter the number of decimals that you want to be displayed.</td>
</tr>
<tr>
<td>Edit decimals</td>
<td>Enter the number of decimals that you want when modifying.</td>
</tr>
</tbody>
</table>

Continues on next page
A Options

Continued

<table>
<thead>
<tr>
<th>Default orientation format</th>
<th>Specifies the default format to use for orientations.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• RPY angles (Euler ZYX)</td>
<td></td>
</tr>
<tr>
<td>• Quaternions</td>
<td></td>
</tr>
</tbody>
</table>

**Options: General: Advanced**

<table>
<thead>
<tr>
<th>Number of undo/redo steps</th>
<th>The number of operations that can be undone or redone. Lowering this value can decrease memory usage.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warn about running Virtual Controller processes on startup</td>
<td>Warns of orphaned VC processes.</td>
</tr>
<tr>
<td>Show acknowledge dialog box when deleting objects</td>
<td>Warns when deleting objects.</td>
</tr>
<tr>
<td>Show acknowledge dialog box when deleting targets and corresponding move instructions</td>
<td>Warns when deleting targets and move instructions.</td>
</tr>
<tr>
<td>Bring the output window to front if an error message is displayed</td>
<td>Select this check box to bring the output window to front if an error message is displayed</td>
</tr>
</tbody>
</table>

**Options: General: Autosave**

<table>
<thead>
<tr>
<th>Enable autosave of RAPID</th>
<th>This check-box is selected by default. RAPID programs are saved automatically in every 30 seconds.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable autosave of station</td>
<td>Unsaved stations are saved automatically at the interval specified in the minute interval box.</td>
</tr>
<tr>
<td>Enable automatic backup of station files</td>
<td>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 Solution.</td>
</tr>
<tr>
<td>Enable automatic backup of controllers in solution</td>
<td>Select this option to backup the virtual controllers of a solution when saving the station. The backups are stored in the Backups folder of the corresponding solution.</td>
</tr>
</tbody>
</table>

**Options: General: Files & Folders**

<table>
<thead>
<tr>
<th>User Documents location</th>
<th>Shows the default path to the project folder.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solutions location</td>
<td>Shows the default path to the solutions folder.</td>
</tr>
<tr>
<td>...</td>
<td>To browse to the project folder, click the browse button.</td>
</tr>
<tr>
<td>Automatically create document subfolders</td>
<td>Select this check box to enable the creation of individual sub-folders for document types.</td>
</tr>
<tr>
<td>minute interval</td>
<td>Specify the interval between the savings when using Autosave in this box.</td>
</tr>
<tr>
<td>Document Locations</td>
<td>Launches the Document Locations dialog box.</td>
</tr>
<tr>
<td>Clear Recent Stations and Controllers</td>
<td>Clears the list of recently accessed stations and controllers.</td>
</tr>
</tbody>
</table>
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\<username>\AppData\Local\ABB\Distribution-Packages`. The location can be customized by entering a search path here.

**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\<username>\AppData\Local\ABB\Distribution-Packages`. 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 mediapools.

### Options: General: Screenshot

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entire application window</strong></td>
<td>Select this option to capture the entire application.</td>
</tr>
<tr>
<td><strong>Active document window</strong></td>
<td>Select this option to capture the active document window, typically the graphics window.</td>
</tr>
<tr>
<td><strong>Copy to clipboard</strong></td>
<td>Select this check box to save the captured image to the system clipboard.</td>
</tr>
<tr>
<td><strong>Save to file</strong></td>
<td>Select this check box to save the captured image to file.</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td>Specify the location of the image file. The default location is the &quot;My Pictures&quot; system folder.</td>
</tr>
<tr>
<td><strong>...</strong></td>
<td>Browse for the location.</td>
</tr>
<tr>
<td><strong>File name</strong></td>
<td>Specify the name of the image file. The default name is &quot;RobotStudio&quot; to which is added a date.</td>
</tr>
<tr>
<td><strong>The file suffix list</strong></td>
<td>Select the desired file format. The default format is JPG.</td>
</tr>
</tbody>
</table>

### Options: General: Screen Recorder

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Framerate</strong></td>
<td>Specify the framerate in frames per second.</td>
</tr>
<tr>
<td><strong>Start recording after</strong></td>
<td>Select this option to start recording after the specified time.</td>
</tr>
<tr>
<td><strong>Stop recording after</strong></td>
<td>Select this option to stop recording after the specified time.</td>
</tr>
<tr>
<td><strong>Include mouse cursor</strong></td>
<td>Select this option to include the mouse cursor for the functions Record Application and Record Graphics.</td>
</tr>
<tr>
<td><strong>Resolution - Same as window</strong></td>
<td>Select this option to use the same resolution as in the graphics window.</td>
</tr>
<tr>
<td><strong>Resolution - Limit resolution</strong></td>
<td>Select this option to scale down the resolution as per the Maximum Width and Maximum Height you specify.</td>
</tr>
<tr>
<td><strong>Maximum width</strong></td>
<td>Specify the maximum width in pixels.</td>
</tr>
<tr>
<td><strong>Maximum height</strong></td>
<td>Specify the maximum height in pixels.</td>
</tr>
<tr>
<td><strong>Video compression</strong></td>
<td>Select the video compression format. Note that DivX format is not supported.</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td>Specifies the location of the videos.</td>
</tr>
<tr>
<td><strong>File name</strong></td>
<td>Enter a file name to save the output file in an MP4 format.</td>
</tr>
</tbody>
</table>

Continues on next page
### Options: Robotics: Text Editor

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show line numbers</td>
<td>Select this check-box to view line numbers in the RAPID editor</td>
</tr>
<tr>
<td>Show ruler</td>
<td>Select this check-box to show the ruler in the RAPID editor</td>
</tr>
<tr>
<td>Show whitespace</td>
<td>Select this check-box to show whitespace characters in the RAPID editor</td>
</tr>
<tr>
<td>Word wrap</td>
<td>Select this check-box if you want to wrap long lines.</td>
</tr>
<tr>
<td>Convert tabs to spaces</td>
<td>Select this check-box to convert tabs to spaces in the RAPID editor</td>
</tr>
<tr>
<td>Tab size</td>
<td>Specify the number of spaces for a Tab press.</td>
</tr>
<tr>
<td>RAPID Text styles</td>
<td>Specify the appearance of the various text classes.</td>
</tr>
<tr>
<td>Text color</td>
<td>Specifies the text color of the RAPID editor.</td>
</tr>
<tr>
<td>Background color</td>
<td>Specifies the background color of the RAPID editor.</td>
</tr>
<tr>
<td>Bold</td>
<td>Select this check-box for bold-face fonts in the RAPID editor</td>
</tr>
<tr>
<td>Italic</td>
<td>Select this check-box for italicized fonts in the RAPID editor</td>
</tr>
<tr>
<td>Follow program pointer by default</td>
<td>Select this check-box to enable the program pointer by default.</td>
</tr>
</tbody>
</table>

### Options: Robotics: RAPID Profiler

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default RAPID log file</td>
<td>Specify the name of the default RAPID log file.</td>
</tr>
<tr>
<td>Always ask for filename</td>
<td>Select this check box to manually specify the file name of the log file always.</td>
</tr>
<tr>
<td>Open analysis when logging is stopped</td>
<td>Select this check box to open the analysis after the log is made.</td>
</tr>
</tbody>
</table>

### Options: Robotics: Graphical Programming

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show dialog when warning for globally defined workobjects</td>
<td>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.</td>
</tr>
<tr>
<td>Show synchronize dialog box after loading program/module</td>
<td>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.</td>
</tr>
<tr>
<td>Show notification that default data is used</td>
<td>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.</td>
</tr>
<tr>
<td>Set as active when creating tooldata</td>
<td>Select this check box if you want newly created tooldata to be set as active. Default value: selected.</td>
</tr>
<tr>
<td>Set as active when creating workobjects</td>
<td>Select this check box if you want newly created workobjects to be set as active. Default value: selected.</td>
</tr>
<tr>
<td>AutoPath</td>
<td>Specify the maximum gap allowed when creating an AutoPath.</td>
</tr>
</tbody>
</table>

### Options: Robotics: Synchronization

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use default synchronization locations</td>
<td>Converting data, such as target to Workobject, shall use the default behavior for synchronization locations. Default value: selected.</td>
</tr>
<tr>
<td>Show default synchronization locations notification</td>
<td>Notifies of the behavior above. Default value: selected.</td>
</tr>
</tbody>
</table>

Continues on next page
**Declaration default locations**

Specify the locations for corresponding objects when synchronizing to the VC.

<table>
<thead>
<tr>
<th>Options: Robotics: Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approach Vector</td>
</tr>
<tr>
<td>Travel Vector</td>
</tr>
<tr>
<td>When jumping to a target or move instruction with undefined configuration</td>
</tr>
<tr>
<td>• Show dialog for setting the configuration</td>
</tr>
<tr>
<td>• Use the configuration closest to the current one</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Options: Robotics: Virtual Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always on top</td>
</tr>
<tr>
<td>Enable transparency</td>
</tr>
<tr>
<td>Logging</td>
</tr>
<tr>
<td>• Select this check box to log the console output to &quot;console.log&quot; in the controller directory</td>
</tr>
<tr>
<td>• Select this check box to log the console output to a console window</td>
</tr>
<tr>
<td>Automatically open virtual Operator Window</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Options: Online: Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recent Users</td>
</tr>
<tr>
<td>Remove/Remove All</td>
</tr>
<tr>
<td>Enable Automatic Logoff</td>
</tr>
<tr>
<td>Timeout</td>
</tr>
<tr>
<td>Displays languages for controller texts such as event logs.</td>
</tr>
<tr>
<td>• RobotStudio language</td>
</tr>
<tr>
<td>• FlexPendant language</td>
</tr>
<tr>
<td>Options: Online: Online Monitor</td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td>Update Rate (s)</td>
</tr>
<tr>
<td>Revolute Joint Limits</td>
</tr>
<tr>
<td>Linear Joint Limits</td>
</tr>
<tr>
<td>Singularities</td>
</tr>
</tbody>
</table>
**Options:** Online: Jobs

<table>
<thead>
<tr>
<th>Specification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max number of devices processed in parallel</td>
<td>Specifies the number of devices for which a job is executed in parallel.</td>
</tr>
<tr>
<td>Directory for log files and report files</td>
<td>Specifies the directory for log/report files.</td>
</tr>
</tbody>
</table>

**Options:** Graphics: Appearance

<table>
<thead>
<tr>
<th>Specification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anti-aliasing</td>
<td>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.</td>
</tr>
<tr>
<td>Font</td>
<td>Specifies the font used in markups.</td>
</tr>
<tr>
<td>Advanced lighting</td>
<td>Select the check box to enable advanced lighting by default.</td>
</tr>
<tr>
<td>Perspective</td>
<td>Click this option to view the perspective view of the object by default.</td>
</tr>
<tr>
<td>Orthographic</td>
<td>Click this option to view the orthographic view of the object by default.</td>
</tr>
<tr>
<td>Custom background color</td>
<td>Click the colored rectangle to change default background color.</td>
</tr>
<tr>
<td>Show floor</td>
<td>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.</td>
</tr>
<tr>
<td>Transparent</td>
<td>Select the check box if you want the floor to be transparent by default. Default values: selected.</td>
</tr>
<tr>
<td>Show UCS Grid</td>
<td>Select the check box if you want the UCS grid to be displayed. Default value: selected.</td>
</tr>
<tr>
<td>Grid Space</td>
<td>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).</td>
</tr>
<tr>
<td>Show UCS coordinate system</td>
<td>Select the check box if you want the UCS coordinate system to be displayed. Default value: selected.</td>
</tr>
<tr>
<td>Show world coordinate system</td>
<td>Select the check box if you want the coordinate systems to be displayed. Default value: selected.</td>
</tr>
<tr>
<td>Show navigation and selection buttons</td>
<td>Select this check box to have the navigation and selection buttons on the graphics window.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Specification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rendering detail level</td>
<td>Select if the detail level is to be Auto, Fine, Medium or Coarse. Default value: Auto.</td>
</tr>
<tr>
<td>Render both sides of surfaces</td>
<td>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.</td>
</tr>
<tr>
<td>Cull objects smaller than</td>
<td>Select the size in pixels under which objects will be disregarded. Default value: 2 pixels.</td>
</tr>
</tbody>
</table>

*Continues on next page*
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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigation</td>
<td>Select a navigational activity and then specify the mouse buttons to be used for the selected navigational activity.</td>
</tr>
<tr>
<td>Navigation sensitivity</td>
<td>Select the navigation sensitivity when using the mouse movements or navigation buttons by clicking the bar and dragging it into position. Default value: 1.</td>
</tr>
<tr>
<td>Automatically adjust view center distance</td>
<td>Select to automatically adjust the view center distance when rotating or zooming a 3D view.</td>
</tr>
<tr>
<td>Selection radius (pixels)</td>
<td>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.</td>
</tr>
<tr>
<td>Selection highlight color</td>
<td>Click the colored rectangle to change the highlight color.</td>
</tr>
<tr>
<td>Selection preview</td>
<td>Select the check box to enable temporarily highlighting of items that may be selected when the mouse cursor passes over them. Default value: selected.</td>
</tr>
<tr>
<td>Show local coordinate system for selected objects</td>
<td>Select the check box to show the local coordinate system for the selected objects. Default value: selected.</td>
</tr>
</tbody>
</table>

### Options: Graphics: Geometry

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail Level</td>
<td>Specify the level of detail required when importing geometries. Select Fine, Medium or Coarse as required.</td>
</tr>
</tbody>
</table>

### Options: Graphics: Stereo/VR*

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mirror Output</td>
<td>Displays the image available in the VR glasses in the Graphics view.</td>
</tr>
<tr>
<td>Quality</td>
<td>Move the slider to adjust the quality of the image to an acceptable level of lag.</td>
</tr>
<tr>
<td>Disable Anti-Aliasing</td>
<td>This option is disabled by default for better performance.</td>
</tr>
</tbody>
</table>

### Options: Simulation: Clock

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation speed</td>
<td>Sets the simulation speed relative to real time. You can define the simulation speed to a maximum of 200%.</td>
</tr>
<tr>
<td>As fast as possible</td>
<td>Select this check box to run the simulation as fast as possible. When you select this option, the simulation speed slider is disabled.</td>
</tr>
<tr>
<td>Simulation timestep</td>
<td>Specifies the simulation timestep.</td>
</tr>
<tr>
<td>Run time slice in parallel for multiple controllers</td>
<td>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.</td>
</tr>
</tbody>
</table>

Continues on next page
### Options: Simulation: Collision

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform collision detection</td>
<td>Select if collision detection is to be performed during simulation or always. Default value: always.</td>
</tr>
<tr>
<td>Pause/stop simulation at collision</td>
<td>Select this check box if you want the simulation to stop at a collision or at a near miss. Default value: cleared.</td>
</tr>
<tr>
<td>Log collisions to Output window</td>
<td>Select this check box if you want the collisions to be logged to the output window. Default value: selected.</td>
</tr>
<tr>
<td>Log collisions to file:</td>
<td>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.</td>
</tr>
<tr>
<td>Enable fast collision detection</td>
<td>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.</td>
</tr>
</tbody>
</table>

### Options: Simulation: Physics

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collision Geometry detail level</td>
<td>Set the slider to set the physics collision geometry either to faster or to a more accurate state.</td>
</tr>
</tbody>
</table>
B Terminology

A

ABB Library

The default repository of 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

Activation key is a 25-character key that ABB sends by e-mail during RobotStudio purchase. Use this key 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.

Continues on next page
B Terminology

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

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.

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.

Continues on next page
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.

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.
**B Terminology**

*Continued*

**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 `.rpack`. Use the Install Package command in the RobotApps 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` and **on a PC with a default installation of Windows 7, the path is C:\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` and **on a PC with a default installation of Windows 7, the path is C:\Users\<user name>\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.

**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.

*Continues on next page*
• 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.

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 real 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 real controller.

Library files

Library files are standalone external reusable files that are added to a RobotStudio station. The ABB product range of robots are installed as library files with RobotStudio, by default. 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.

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.

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.

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.

Offline

User is disconnected from a real controller and is working with a virtual controller.
Orientation

The orientation of an object such as a line, plane or rigid body is its 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 ZYX, 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.

**Online**

User is connected directly to a real controller through the network.

**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.

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.

Pack & go
Way to share RobotStudio stations by combining the station and virtual controllers packaged into one file.

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.

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.

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.
**B Terminology**

*Continued*

**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.

**Real 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 real 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.

**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.

*Continues on next page*
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, work objects, 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.

Solution
Solutions add structure to the station data. It contains folders for structuring station data so as to keep related data together.

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.

Continues on next page
B Terminology

Continued

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.

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.
B Terminology

Continued

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.

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 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 real controller on the PC. It is used for offline programming and simulation. Virtual controller replicates the RobotWare system.
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.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td>World coordinate system</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>Work Object coordinate system 1</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>Work Object coordinate system 2</td>
</tr>
</tbody>
</table>
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.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Base coordinate system for robot 1</td>
</tr>
<tr>
<td>B</td>
<td>World coordinate</td>
</tr>
<tr>
<td>C</td>
<td>Base coordinate system for robot 2</td>
</tr>
</tbody>
</table>

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 envelopes are only available for articulated robots, and not available for picker robots or external axes.
B Terminology

Continued

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.
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
Index

C
- collision sets, 124
- controller world coordinate system, 77

D
- detecting collision, 125

G
- graphics window, 39

L
- LED, 231

M
- Manage ScreenMaker project
  - Close ScreenMaker, 245
- Manage ScreenMaker Project
  - Close project, 245
- Managing ScreenMaker Projects
  - Build project, 244
  - Modify project properties, 243

N
- near-miss detection, 126
- network security, 15
- network settings
  - firewall settings, 33
  - local network connection, 33
  - remote network connection, 33
  - service port connection, 33

P
- Properties Window
  - Event Help panel, 232
  - Graphical Component Name panel, 232
  - Properties window toolbar, 232
  - Table panel, 232

R
- remote subnet, 34

S
- safety, 14
- station world coordinate system, 75
- switch, 231
- System Builder
  - about virtual and real systems, 171
  - download a system to controller, 184

T
- ToolBox
  - ActionTrigger, 230
  - BarGraph, 230
  - CheckBox, 230
  - ComboBox, 231
  - CommandBar, 231
  - ConditionalTrigger, 231
  - ControllerModeStatus, 231
  - DataEditor, 231
  - Graph, 231
  - GroupBox, 231
  - ListBox, 231
  - NumEditor, 231
  - NumericUpDown, 231
  - Panel, 231
  - PictureBox, 231
  - RapidExecutionStatus, 231
  - RunRoutineButton, 231
  - TabControl, 231
- TpsLabel, 231

U
- UCS, 80
- user coordinate system, 80

V
- VariantButton, 231

W
- WorkObject, 80
- world coordinate system, 75
ABB AB, Robotics
Robotics and Motion
S-721 68 VÄSTERÅS, Sweden
Telephone +46 (0) 21 344 400

ABB AS, Robotics
Robotics and Motion
Nordlysvegen 7, N-4340 BRYNE, Norway
Box 265, N-4349 BRYNE, Norway
Telephone: +47 22 87 2000

ABB Engineering (Shanghai) Ltd.
Robotics and Motion
No. 4528 Kangxin Highway
PuDong District
SHANGHAI 201319, China
Telephone: +86 21 6105 6666

ABB Inc.
Robotics and Motion
1250 Brown Road
Auburn Hills, MI 48326
USA
Telephone: +1 248 391 9000

abb.com/robotics

© Copyright 2019 ABB. All rights reserved.
Specifications subject to change without notice.