



ROBOTICS

Operating manual

OmniCore



Trace back information:
Workspace 25B version a9
Checked in 2025-07-03
Skribenta version 5.6.019

Operating manual

OmniCore

Robotware 7.19

Document ID: 3HAC065036-001

Revision: W

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

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

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

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

Keep for future reference.

Additional copies of this manual may be obtained from ABB.

Original instructions.

Table of contents

| | |
|--|-----------|
| Overview of this manual | 9 |
| Product documentation | 15 |
| 1 Introduction to OmniCore | 17 |
| 1.1 About this section | 17 |
| 1.2 The OmniCore controller | 18 |
| 1.3 The FlexPendant | 20 |
| 1.4 RobotStudio | 26 |
| 1.5 Directory structure on OmniCore | 27 |
| 2 Navigating and handling the FlexPendant | 29 |
| 2.1 Overview | 29 |
| 2.2 The user interface | 30 |
| 2.2.1 FlexPendant touch screen | 30 |
| 2.2.2 Status bar | 33 |
| 2.2.2.1 Introduction | 33 |
| 2.2.2.2 Messages window | 34 |
| 2.2.2.3 QuickSet | 35 |
| 2.2.3 FlexPendant applications | 45 |
| 2.3 Personalizing the FlexPendant | 51 |
| 2.3.1 Introduction | 51 |
| 2.3.2 System information | 52 |
| 2.3.3 Configuring the identity settings | 53 |
| 2.3.4 Configuring date and time | 54 |
| 2.3.5 Configuring the interface language | 56 |
| 2.3.6 Configuring the default view during system events | 57 |
| 2.3.7 Configuring the Joystick | 58 |
| 2.3.8 Configuring the display of event logs | 59 |
| 2.3.9 Configuring the display of Controller ID and System Name | 60 |
| 2.3.10 Configuring the default paths | 61 |
| 2.3.11 Configuring the programmable keys | 62 |
| 2.3.12 Adapting the FlexPendant for left-handed users | 64 |
| 2.4 Basic procedures | 65 |
| 2.4.1 Using the soft keyboard | 65 |
| 2.4.2 Messages on the FlexPendant | 67 |
| 2.4.3 Capturing screenshots | 68 |
| 2.4.4 Filtering data | 69 |
| 2.4.5 Searching the settings | 70 |
| 2.4.6 Granting access for RobotStudio | 71 |
| 2.4.7 Logging on and off | 72 |
| 2.4.8 Changing the user password | 74 |
| 2.4.9 Calibrating the touchscreen | 75 |
| 2.5 Updating the applications | 76 |
| 3 OmniCore controller operating modes | 77 |
| 3.1 Introduction | 77 |
| 3.2 Changing operating modes | 78 |
| 3.3 Locking and unlocking operating modes | 79 |
| 4 Calibration | 83 |
| 4.1 Introduction | 83 |
| 4.2 How to check if the robot needs calibration | 84 |
| 5 Jogging | 87 |
| 5.1 Introduction to jogging | 87 |

| | | |
|----------|---|------------|
| 5.2 | Coordinate systems for jogging | 90 |
| 5.3 | Basic settings for jogging | 95 |
| 5.4 | Reading the exact position | 97 |
| 5.5 | Incremental movement for precise positioning | 99 |
| 5.6 | Restrictions to jogging | 100 |
| 5.7 | Lead-through | 101 |
| 5.8 | Motion supervision | 104 |
| 5.9 | Align tool | 106 |
| 5.10 | Working with SmartGripper | 107 |
| 5.10.1 | Introduction | 107 |
| 5.10.2 | Gripper configuration | 111 |
| 5.10.3 | Smart Gripper function | 112 |
| 6 | Programming and testing | 113 |
| 6.1 | Introduction | 113 |
| 6.2 | Before you start programming | 114 |
| 6.3 | Programming concept | 116 |
| 6.3.1 | Handling of programs | 116 |
| 6.3.2 | Handling of modules | 121 |
| 6.3.3 | Handling of routines | 126 |
| 6.3.4 | Handling of instructions | 134 |
| 6.3.5 | Example: Add movement instructions | 142 |
| 6.3.6 | Program and motion pointers | 144 |
| 6.4 | Debugging the program | 145 |
| 6.5 | Data types | 147 |
| 6.5.1 | Edit data in specific tasks, modules, or routines | 147 |
| 6.5.2 | Creating new data instance | 148 |
| 6.5.3 | Editing data instances | 150 |
| 6.6 | Tools | 152 |
| 6.6.1 | What is a tool? | 152 |
| 6.6.2 | What is the tool center point? | 154 |
| 6.6.3 | Creating a tool | 156 |
| 6.6.4 | Copying a tool | 158 |
| 6.6.5 | Defining the tool frame | 159 |
| 6.6.6 | Editing the tool data | 164 |
| 6.6.7 | Deleting a tool | 167 |
| 6.6.8 | Setup for stationary tools | 168 |
| 6.7 | Work objects | 170 |
| 6.7.1 | What is a work object? | 170 |
| 6.7.2 | Creating a work object | 171 |
| 6.7.3 | Copying a workobject | 172 |
| 6.7.4 | Defining a work object | 173 |
| 6.7.5 | Defining the work object coordinate system | 175 |
| 6.7.6 | Editing the work object data | 178 |
| 6.7.7 | Deleting a work object | 179 |
| 6.7.8 | Setup stationary work object | 180 |
| 6.8 | Payloads | 181 |
| 6.8.1 | Overview | 181 |
| 6.8.2 | Creating a payload | 182 |
| 6.8.3 | Copying a payload | 184 |
| 6.8.4 | Editing the payload data | 185 |
| 6.8.5 | Deleting a payload | 187 |
| 6.9 | Testing | 188 |
| 6.9.1 | Using the hold-to-run function | 188 |
| 6.9.2 | Running the program from a specific instruction | 190 |
| 6.9.3 | Running a specific routine | 191 |
| 6.9.4 | Stepping instruction by instruction | 192 |
| 6.10 | Service routines | 195 |
| 6.10.1 | Running a service routine | 195 |

| | | |
|-----------|--|------------|
| 6.10.2 | Connected Services Reset service routine | 198 |
| 6.10.3 | Battery shutdown service routine | 199 |
| 6.10.4 | Axis Calibration service routine | 200 |
| 6.10.5 | Calibration Pendulum, CalPendulum service routine | 201 |
| 6.10.6 | Service Information System, ServiceInfo service routine | 202 |
| 6.10.7 | LoadIdentify, load identification service routine | 203 |
| 6.10.8 | Brake check service routine | 215 |
| 6.10.9 | Cyclic Brake Check service routine | 222 |
| 6.10.10 | Wrist optimization service routine | 226 |
| 6.10.11 | SkipTaskExec service routine | 227 |
| 6.10.12 | Commutation service routine | 228 |
| 7 | Running in production | 229 |
| 7.1 | Introduction | 229 |
| 7.2 | Basic procedures | 230 |
| 7.2.1 | Starting programs | 230 |
| 7.2.2 | Stopping programs | 233 |
| 7.2.3 | Using multitasking programs | 234 |
| 7.2.4 | Returning the robot to the path | 237 |
| 7.2.5 | Using motion supervision and non motion execution | 238 |
| 7.3 | Managing Dashboards | 240 |
| 7.4 | Managing positions | 243 |
| 7.4.1 | Introduction | 243 |
| 7.4.2 | Teach position | 244 |
| 7.4.3 | Go To position | 249 |
| 7.4.4 | Working with displacements and offsets | 250 |
| 7.5 | Detaching and attaching a FlexPendant | 252 |
| 8 | Handling inputs and outputs, I/O | 255 |
| 8.1 | Introduction | 255 |
| 8.2 | Viewing signal lists | 256 |
| 8.3 | Setting signals as favorite signals | 257 |
| 8.4 | Simulating the signals and changing the signal values | 258 |
| 8.5 | I/O devices | 260 |
| 9 | Handling the event log | 263 |
| 9.1 | Introduction | 263 |
| 9.2 | Accessing the event log | 264 |
| 9.3 | Saving log entries | 265 |
| 9.4 | Clearing the log entries | 266 |
| 10 | Install, update, restart, and other configuration | 267 |
| 10.1 | Introduction | 267 |
| 10.2 | Install RobotWare system | 268 |
| 10.3 | Restart | 269 |
| 10.4 | Backup and restore systems | 271 |
| 10.4.1 | What is saved on backup? | 271 |
| 10.4.2 | Backup the system | 274 |
| 10.4.3 | Important when performing backups | 276 |
| 10.4.4 | Restore the system | 277 |
| 10.5 | Reset user data | 279 |
| 10.6 | FlexPendant logs | 281 |
| 10.7 | Update FlexPendant | 282 |
| 10.8 | Connection log | 284 |
| 10.9 | Connected Services log | 285 |
| 10.10 | Creating a system diagnostics file | 286 |
| 10.11 | Shutting down | 287 |

| | |
|-------|-----|
| Index | 289 |
|-------|-----|

Overview of this manual

About this manual

This manual contains instructions for operation of OmniCore controller based robots, primarily by using the FlexPendant.



Note

It is the responsibility of the integrator to conduct a risk assessment of the final application.

It is the responsibility of the integrator to provide safety and user guides for the robot system.



Note

Screenshots in this manual are generally intended to show a language version corresponding to the language of the manual. In some cases, a translated manual still uses English screenshots if the localized user interface was not available at the time of publishing the manual.

Usage

This manual should be used during operation.

Some actions that are more advanced, or not used in the daily operation, are described in *Operating manual - Integrator's guide OmniCore*.



Note

Before any work on or with the robot is performed, the safety information in the product manual for the controller and manipulator shall be read.

Who should read this manual?

This manual is intended for:

- operators
- product technicians
- service technicians
- robot programmers

Prerequisites

The reader should:

- Have read and understood the safety instructions in the product manuals for the robot.
- Be trained in robot operation.

Continues on next page

References

Documentation referred to in the manual, is listed in the table below.

| Document name | Document ID |
|---|----------------|
| Product manual - OmniCore E10 | 3HAC079399-001 |
| Product manual - OmniCore C30 | 3HAC060860-001 |
| Product manual - OmniCore C90XT Type A | 3HAC089065-001 |
| Product manual - OmniCore V250XT Type B | 3HAC087112-001 |
| Product manual - OmniCore V400XT | 3HAC081697-001 |
| User manual - FlexPendant | 3HAC093167-001 |
| Operating manual - Integrator's guide OmniCore | 3HAC065037-001 |
| Operating manual - Calibration Pendulum | 3HAC16578-1 |
| Operating manual - RobotStudio | 3HAC032104-001 |
| Application manual - Controller software OmniCore | 3HAC066554-001 |
| Application manual - MultiMove | 3HAC089689-001 |
| Application manual - Functional safety and SafeMove | 3HAC066559-001 |
| Technical reference manual - System parameters | 3HAC065041-001 |
| Operating manual - Calibration Pendulum | 3HAC16578-1 |



Tip

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

Revisions

| Revision | Description |
|----------|--|
| A | First edition. |
| B | <ul style="list-style-type: none">The safety information is moved to the product manuals for the controller and the manipulator.Updated the section Changing operating modes on page 78.Updated the section Detaching and attaching a FlexPendant on page 252. |
| C | <p>Released with RobotWare 7.0.1. The following updates are made in this revision:</p> <ul style="list-style-type: none">References to the <i>Hold-to run-button</i> is replaced with <i>thumb button</i> in the manual.Section <i>Service Information System service routine</i> updated with new counter: <i>moved distance</i>.Added information about recently used programs in the section Starting programs on page 230.Updated the section Creating new data instance on page 148. |

Continues on next page

| Revision | Description |
|----------|--|
| D | <p>Released with RobotWare 7.0.2. The following updates are made in this revision:</p> <ul style="list-style-type: none"> • FlexPendant terminology updated in entire manual. • Added information about SafeMove. • Updated information about queueing backups. • Added the section Locking and unlocking operating modes on page 79. • Added the section Go To position on page 249. • Updated the section QuickSet window tabs on page 36. • Updated the section Detaching and attaching a FlexPendant on page 252. • Updated the section Creating new data instance on page 148. • The directory of BC_config_IO.sys file is corrected in the section Description of the I/O setup on page 218. |
| E | <p>Released with RobotWare 7.1. The following updates are made in this revision:</p> <ul style="list-style-type: none"> • Removed information from Calibration chapter and inserted references to product manuals. • Added information about OmniCore C90XT in the section OmniCore C line on page 18. • Added information about Array data in various sections. • Updated the section Restart. • Added the section Working with SmartGripper on page 107. • Added the section Copying a data instance on page 151. • Added the section Managing positions on page 243. • Added the section Update position on page 137. • Added the section Capturing screenshots on page 68. • Added the section Debugging the program on page 145. • Added the section Updating the FlexPendant applications on page 282. |
| F | <p>Released with RobotWare 7.2. The following updates are made in this revision:</p> <ul style="list-style-type: none"> • Added the section Directory structure on OmniCore on page 27. • Added the section Shifting an instruction on page 140. • Added the section Motion supervision on page 104. • Added the section Incremental movement for precise positioning on page 99. • Added the section File Explorer on page 48. • Added information about task bar in the section FlexPendant touch screen on page 30. • A NOTE is added in section Reset user data that the TEMP folder is emptied at controller restart if RAPID and system parameters are reset. • A Footnote is added in the section FlexPendant applications on page 45 stating for what manipulators the lead-through functionality is applicable. |

Continues on next page

| Revision | Description |
|----------|--|
| G | Released with RobotWare 7.3. The following updates are made in this revision: <ul style="list-style-type: none"> Updated the section Configuring the interface language on page 56. Updated the section Configuring date and time on page 54. Updated information about grant in the section Detaching and attaching a FlexPendant on page 252. Updated the section File Explorer on page 48. Updated the section Locking and unlocking operating modes on page 79. |
| H | Released with RobotWare 7.4. The following updates are made in this revision: <ul style="list-style-type: none"> Added the section Configuring the default view during system events on page 57. Added the section Edit options on page 138. Added the section Connected Services log on page 285. Added the section Axis Calibration service routine on page 200. Added the section Wrist optimization service routine on page 226. Updated the section Connection log on page 284. Updated the section Handling inputs and outputs, I/O on page 255. Updated the section Editing data in specific tasks, modules, or routines on page 147. Information about OmniCore E line added in the section The OmniCore controller on page 18. |
| J | Released with RobotWare 7.5. The following updates are made in this revision: <ul style="list-style-type: none"> Added the section Adapting the FlexPendant for left-handed users on page 64. Added information for delta robots, see LoadIdentify, load identification service routine on page 203. Added information about Update Load in Jog Settings on page 38. Information about V line controller added in References on page 10 and The OmniCore controller on page 18. |
| K | Released with RobotWare 7.6. The following updates are made in this revision: <ul style="list-style-type: none"> Added information regarding Select Range in the section Edit options on page 138. Added the section SkipTaskExec service routine on page 227. Added the section Scenarios while editing a line that contains multiple instructions on page 141. Added information regarding QR code in the section Connected Services on page 42. Updated the section Lead-through on page 101. Updated the section Editing the values of a data instance on page 146. Updated the section Configuring date and time on page 54. Updated various sections to reflect the change of introducing Program Data as a new app. Minor changes in chapter Install, update, restart, and other configuration on page 267. Corrected name of UAS grant for locking the operating mode, see Locking and unlocking operating modes on page 79. |

Continues on next page

| Revision | Description |
|----------|---|
| L | Released with RobotWare 7.7. The following updates are made in this revision: <ul style="list-style-type: none"> Added the section Description of the icons in the QuickSet button on page 35. Added information about Align to a coordinate system on page 102 in the Lead-through section. Added information for 5-axis delta robots, see LoadIdentify, load identification service routine on page 203. Added a NOTE for the option Edit selection with keyboard in the section Edit options on page 138. Removed the information about <i>Pendulum Calibration</i> as that is not available on OmniCore. |
| M | Released with RobotWare 7.8. The following updates are made in this revision: <ul style="list-style-type: none"> Updated the section Updating the FlexPendant applications on page 282. Updated the section Modifying instructions on page 135. The name of the service routine <i>Bat_Shutdown</i> is changed to <i>BatteryShutDown</i>. |
| N | Released with RobotWare 7.10. The following updates are made in this revision: <ul style="list-style-type: none"> Added the section Configuring the display of Controller ID and System Name on page 60. Added the section Changing the user password on page 74. Information about OmniCore V250XT Type A added in References on page 10 and The OmniCore controller on page 18. Updated various sections to reflect the change of transferring the RAPID data editing functionality, from <i>Calibrate</i> to <i>Program Data</i>. Minor corrections in Service Information System, ServiceInfo service routine on page 202. |
| P | Released with RobotWare 7.12. The following updates are made in this revision: <ul style="list-style-type: none"> Added the section Configuring the default paths on page 61. Added a NOTE about locked accounts in the section Login procedure on page 72. Added a NOTE regarding adding the value in Mass field in the section How to create a tool on page 156. |
| Q | Released with RobotWare 7.13. The following updates are made in this revision: <ul style="list-style-type: none"> Information about OmniCore V250XT Type B added in References on page 10 and The OmniCore controller on page 18. Information about OmniCore V400XT added in References on page 10 and The OmniCore controller on page 18. ABB Connected Services is the new name for ABB Ability Connected Services. Added information that the restart type Reset System does not reset the topic <i>Communication</i> in the system parameter configuration. The functionality for non motion execution is now available in RobotWare. See Non motion execution on page 105. |

Continues on next page

| Revision | Description |
|----------|---|
| R | Released with RobotWare 7.14. The following updates are made in this revision: <ul style="list-style-type: none">• Added information about the <i>CalPendulum</i> service routine. This can be used only for IRB 1510ID, IRB 1520ID, IRB 2400, and IRB 4400. See Calibration Pendulum, CalPendulum service routine on page 201.• Added the section Calibrating the touchscreen on page 75.• Updated the section Moving the robot on page 145. |
| S | Released with RobotWare 7.15. The following updates are made in this revision: <ul style="list-style-type: none">• Updated the section Editing data in specific tasks, modules, or routines on page 147.• Added information about increment mode icons in the section Description of the icons in the QuickSet button on page 35.• Information about UAS settings and backups is updated in Restore the system on page 277, What is saved on backup? on page 271. |
| T | Released with RobotWare 7.16. The following updates are made in this revision: <ul style="list-style-type: none">• Updated the section Connected Services on page 42.• Updated the section Loading an existing module on page 121.• Updated the section Adding instructions on page 134.• Added the section Verifying the backup file before restore on page 277. |
| U | Released with RobotWare 7.17. The following updates are made in this revision: <ul style="list-style-type: none">• Information about execution tab added in section The execution tab on page 144.• Information about task selection panel added in section Switching between tasks on page 236.• Minor corrections. |
| V | Released with RobotWare 7.18. The following updates are made in this revision: <ul style="list-style-type: none">• New section: Shutting down on page 287.• Add new content in section Adding instructions on page 134. |
| W | Released with RobotWare 7.19. <ul style="list-style-type: none">• Added information about detaching the FlexPendant in manual mode, see Detaching and attaching a FlexPendant on page 252.• Added more information about RAPID programming language, see Naming convention on page 115.• Added information about submitting feedback, see Feedback on page 43.• Added the section Commutation service routine on page 228. |

Product documentation

Categories for user documentation from ABB Robotics

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



Tip

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

Product manuals

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

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

Technical reference manuals

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

Application manuals

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

An application manual generally contains information about:

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

Continues on next page

- Examples of how to use the application.

Operating manuals

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

1 Introduction to OmniCore

1.1 About this section

Overview

This section presents an overview of the FlexPendant, the OmniCore controller, and RobotStudio.

A robot consists of a robot controller, the FlexPendant, RobotStudio, and one or several manipulators or other mechanical units.

This manual describes a robot without options, not a robot system. However, in a few places, the manual gives an overview of how options are used or applied. Most options are described in detail in their respective application manual.

1 Introduction to OmniCore

1.2 The OmniCore controller

1.2 The OmniCore controller

Overview of OmniCore

The OmniCore controller contains all the functions needed to move and control the manipulator, and delivers flexibility, connectivity, and performance. The OmniCore controller gives ABB robots the ability to perform their tasks in a highly efficient manner and also increases the flexibility to incorporate the latest digital technologies. The controller comes with ABB's powerful operating system, RobotWare 7.

The controller can be equipped with additional offerings, such as industrial network protocols, vision solutions, and force control.

OmniCore A line

OmniCore A line is a series of cabinets for integration of, for example, application equipment with the same high degree of flexibility as the OmniCore controllers.

OmniCore E line

OmniCore E line is an ultra slim controller for confined spaces and high density lines within the OmniCore family, offering only the essential functions together with full motion performance and precision. The E line controller is designed for stand-alone and slave installations and has a very compact design with integrated hardware and RobotWare functions.

OmniCore E10

The OmniCore E10 controller has only one base configuration without electronic hardware options. For more information about the OmniCore E10 controller, see *Product manual - OmniCore E10*.

OmniCore C line

OmniCore C line is the compact line of controllers within the OmniCore family, offering significant size reduction and flexible integration possibilities without any compromise on performance or precision.

OmniCore C30

The OmniCore C30 controller offers a compact solution suitable for applications where there is less need for additional equipment inside. There are two variants available, used for different manipulators. For more information about the OmniCore C30 controller, see *Product manual - OmniCore C30* and *Product manual - OmniCore C30 Type A*.

OmniCore C90XT

The OmniCore C90XT controller offers a compact solution suitable for most applications with room for some additional equipment inside. For more information about the OmniCore C90XT controller, see *Product manual - OmniCore C90XT Type A*.

Continues on next page

OmniCore V line

OmniCore V line is a versatile and powerful controller with high degree of flexibility covering a wide range robot and applications. V line supports additional axes and provides flexible configuration opportunities.

OmniCore V250XT Type B

The OmniCore V250XT Type B controller offers a compact, yet flexible, solution for advanced applications and robots sizes up to IRB 77x0. The controller supports up to three additional drive units and has 15 liter optional space inside.

For more information about the OmniCore V250XT Type B controller, see *Product manual - OmniCore V250XT Type B*.

OmniCore V400XT

The OmniCore V400XT controller offers a compact, yet flexible, solution for advanced applications and robots sizes up to IRB 8700. The controller supports up to six additional drive units and has 50 liter optional space inside.

For more information about the OmniCore V400XT controller, see *Product manual - OmniCore V400XT*.

1.3 The FlexPendant

Introduction to the FlexPendant

The FlexPendant is a hand held operator unit that is used for many of the tasks when operating a robot: running programs, jogging the manipulator, modifying programs, and so on.

The FlexPendant is designed for continuous operation in harsh industrial environment. Its touchscreen is easy to clean and resistant to water, oil, and accidental welding splashes.

The FlexPendant consists of both hardware and software and is a complete computer in itself. It is connected to the robot controller by an integrated cable and connector.



Note

The FlexPendant for IRC5 is not compatible with OmniCore, and vice versa.

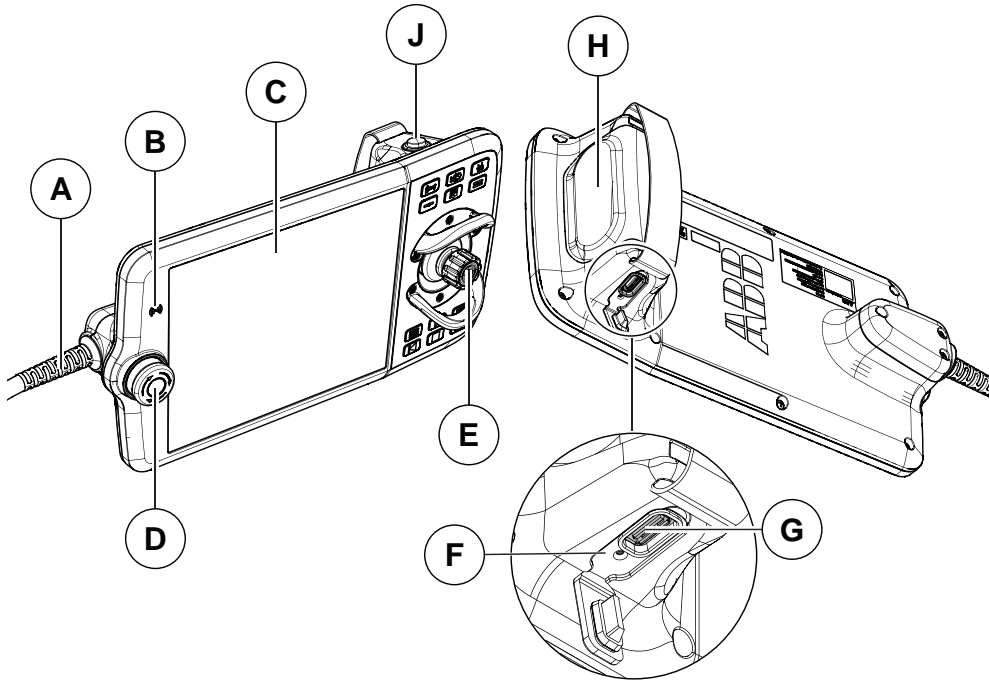


Note

If protective gloves are used, these must be compatible with touchscreens when using the FlexPendant.

Main parts

These are the main parts of the FlexPendant.



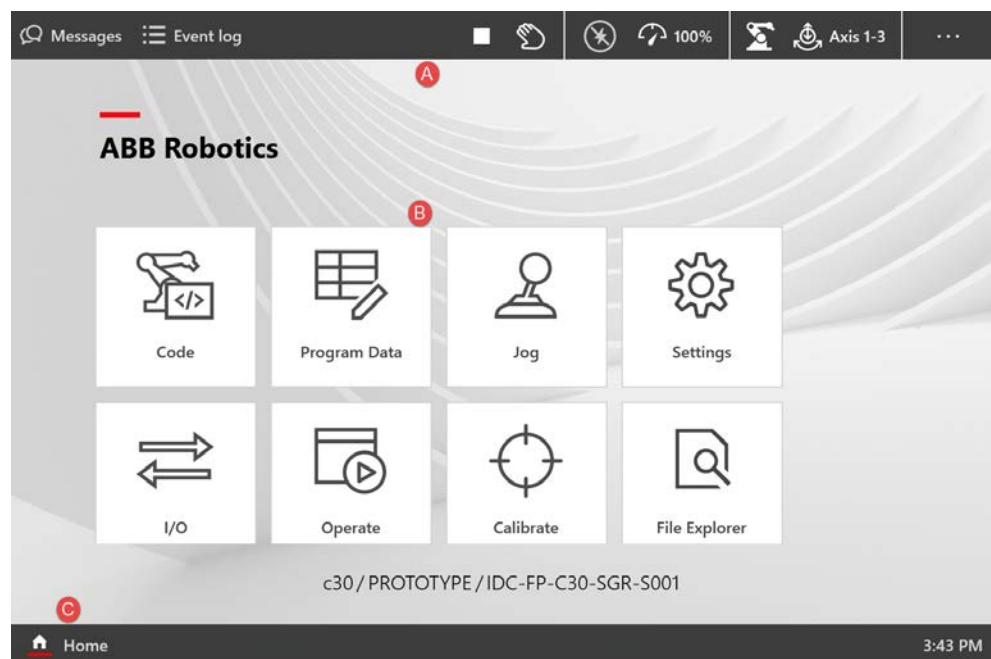
xx1700001891

| | |
|---|-----------|
| A | Connector |
|---|-----------|

Continues on next page

| | |
|---|--|
| B | RFID reader (functionality not yet implemented) |
| C | Touchscreen |
| D | Emergency stop device |
| E | Joystick |
| F | Reset button |
| G | USB port |
| H | Three-position enabling device. For details, see Three-position enabling device on page 23 |
| J | Thumb button. For details, see Thumb button on page 24 . |

Touchscreen



xx1800001181

| | | |
|---|--------------------|---|
| A | Status bar buttons | Allows you to navigate to operator messages, event logs, and QuickSet window. |
| B | Applications | The applications that are required for operating a robot system are available in the Home Screen. By default, the Home screen displays all the applications available to you. |
| C | Home button | From any window tap the Home button to navigate to the Home screen of FlexPendant. The Home screen view is also the default view of the FlexPendant during startup. |

Emergency stop device

On delivery, the emergency stop device on the FlexPendant is able to initiate the emergency stop function affecting the manipulator(s) and additional axis only.

Joystick

Use the joystick to move the manipulator. This is called jogging the robot. There are several settings for how the joystick will move the manipulator.

Continues on next page

1 Introduction to OmniCore

1.3 The FlexPendant *Continued*

Reset button

If the FlexPendant freezes during operation, press the reset button to restart the FlexPendant.

The reset button resets the FlexPendant, not the system on the controller.

USB port

Connect a USB memory to the USB port to read or save files. For example, to load and save programs and modules, save and restore backups, and so on. The USB memory name and drive letter (X:) is displayed in dialogs.

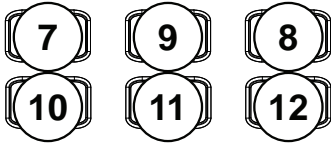
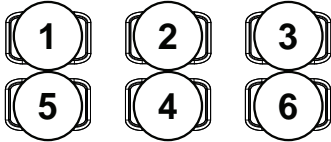
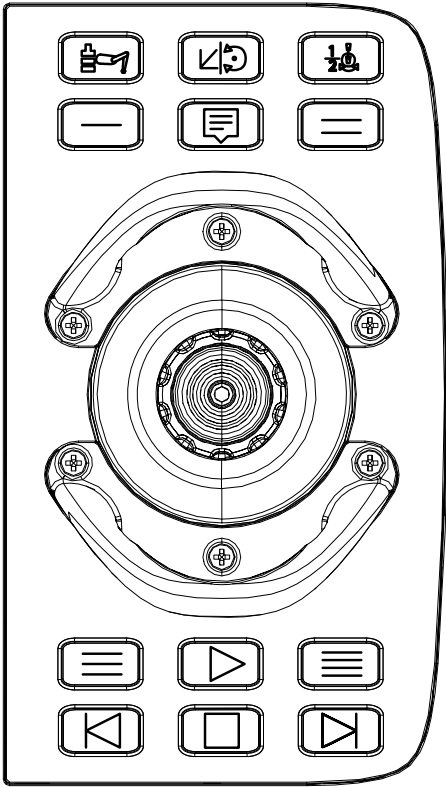


Note

Close the protective cap on the USB port when not used.

Hard buttons


The following hard buttons are available on the FlexPendant.



xx1700001892

| Label | Description |
|-------|---|
| 1 | Mechanical unit button. Allows you to select a mechanical unit. |
| 2 | Motion mode button 1. Allows you to toggle the motion mode between reorient and linear. |
| 3 | Motion mode button 2. Allows you to toggle the motion mode between axis 1-3 and axis 4-6. |

Continues on next page

| Label | Description |
|------------|---|
| 4 | <p>Messages button. Allows you to open the QuickSet window.</p> <p> Note</p> <p>Press the Messages button for a longer duration to capture a screenshot of the current screen.</p> <p>For more details, see Capturing screenshots on page 68.</p> |
| 5, 6, 7, 8 | <p>Programmable keys, 1 to 4.</p> <p>Programmable keys are hardware buttons on the FlexPendant that can be used for dedicated, specific functions set by the user.</p> |
| 9 | START button. Starts the program execution. |
| 10 | Step BACKWARD button. Executes one instruction backward. |
| 11 | STOP button. Stops the program execution. |
| 12 | Step FORWARD button. Executes one instruction forward. |

**Note**

The user interface of the panel in Virtual FlexPendant is slightly different. For more details, see [UI elements in Virtual FlexPendant panel on page 30](#).

Three-position enabling device

**CAUTION**

The person using the three-position enabling device is responsible to observe the safeguarded space for hazards due to robot motion and any other hazards related to the robot.

The three-position enabling device is located on the FlexPendant. When continuously held in center-enabled position, the three-position enabling device will permit robot motion and any hazards controlled by the robot. Release of or compression past the center-enabled position will stop the robot motion.

**CAUTION**

For safe use of the three-position enabling device, the following must be implemented:

- The three-position enabling device must never be rendered inoperational in any way.
- If there is a need to enter safeguarded space, always bring the FlexPendant. This is to enforce single point of control.

**CAUTION**

On the IRB 14050, the three-position enabling device is not active unless a valid SafeMove configuration is active in the controller.

Continues on next page

1 Introduction to OmniCore

1.3 The FlexPendant

Continued



Note

To enforce single-point of control from the FlexPendant, press and release the three-position enabling device twice.



Note

YuMi robots with SafeMove requires using the enabling device.

On YuMi robots without SafeMove the enabling device is disabled, hence, not used.

Thumb button

For robots used in collaborative application, the thumb button is used to enable the lead-through functionality.

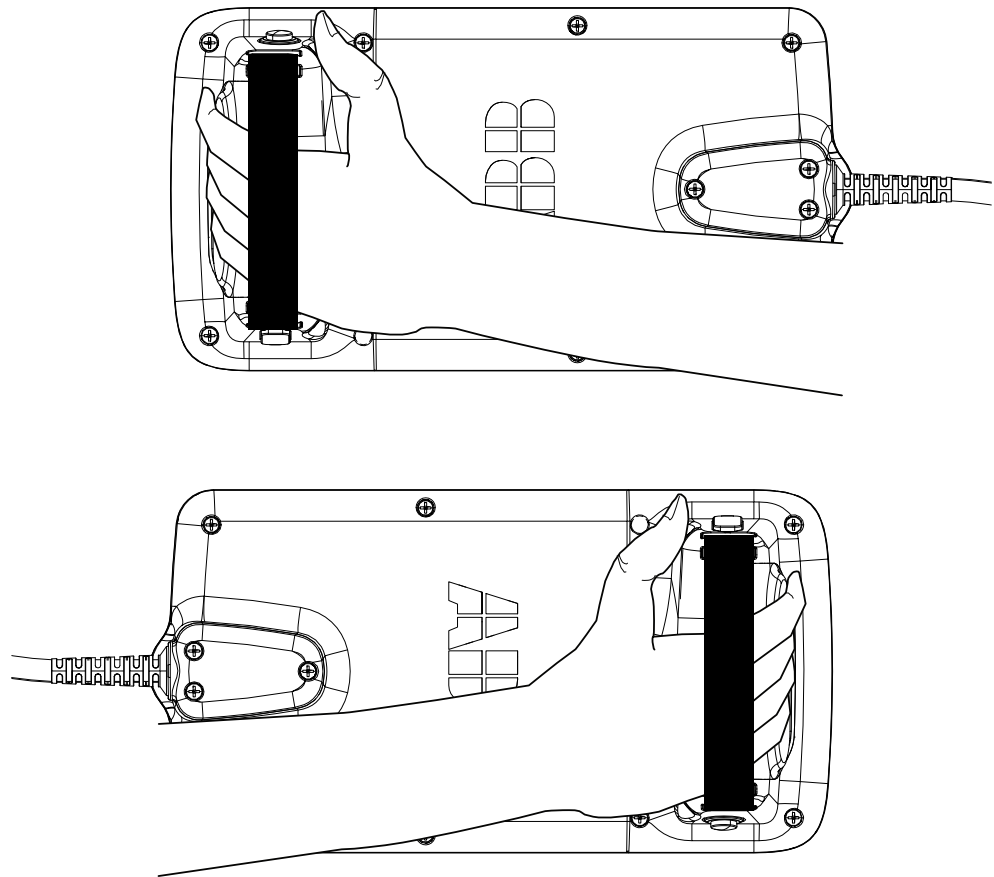
For robots supporting the mode manual full speed, the button is used as hold-to-run.

How to hold the FlexPendant

FlexPendant is typically operated while being held in the hand. The right-handed users use their left-hand to support the FlexPendant while their right-hand performs the operations on the touch screen. However, the left-handed users can easily adapt FlexPendant for their use.

For more details, see the section [Adapting the FlexPendant for left-handed users on page 64](#).

Continues on next page



xx1800000045

1.4 RobotStudio

Overview of RobotStudio

RobotStudio is an engineering tool for the configuration and programming of ABB robots, both real robots on the shop floor and virtual robots in a PC. To achieve true offline programming, RobotStudio utilizes ABB VirtualRobot™ Technology. RobotStudio has adopted the Microsoft Office Fluent User Interface. The Office Fluent UI is also used in Microsoft Office. As in Office, the features of RobotStudio are designed in a workflow-oriented way.

With add-ins, RobotStudio can be extended and customized to suit the specific needs. Add-ins are developed using the RobotStudio SDK. With the SDK, it is also possible to develop custom SmartComponents which exceed the functionality provided by RobotStudio's base components.

For more information, see *Operating manual - RobotStudio*.

RobotStudio for real controllers




RobotStudio allows, for example, the following operations when connected to a real controller:

- Installing and modifying RobotWare systems on controllers, using the **Modify Installation** function.
- Text-based programing and editing, using the **RAPID Editor**.
- File manager for the controller.
- Administrating the User Authorization System.
- Configuring system parameters.

1.5 Directory structure on OmniCore

Default base directory structure

The default base public directory structure on the OmniCore controller consists of the following directories:

| Directory | Description |
|-----------|---|
| HOME | <p>Intended for user files and for use by RAPID programs. Data stored in the <i>HOME</i> directory is included in the Backup and restore function.</p> <p> Note</p> <p>Add-in data should not be stored in this directory. The <i>ADDINDATA</i> directory is used for this purpose instead.</p> |
| DATA | Data stored in the <i>DATA</i> directory is <u>not</u> included in the Backup and restore function. |
| ADDINDATA | <p>The <i>ADDINDATA</i> directory contains a number of subdirectories used for the RobotWare add-ins.</p> <p>See <i>Application manual - RobotWare add-ins</i> for detailed information about the add-in directories.</p> |
| TEMP | <p>The <i>TEMP</i> directory is for the storage of temporary files.</p> <p> CAUTION</p> <p>The <i>TEMP</i> directory is cleaned on system reset and during system updates.</p> |
| BACKUP | The <i>BACKUP</i> directory is used for saving backups. |
| RAMDISK | <p>The <i>RAMDISK</i> directory, located in the RAM disk (nonpersistent), is used for high-performance logging.</p> <p> CAUTION</p> <p>Content is lost on each restart of the controller.</p> |
| PRODUCTS | The <i>PRODUCTS</i> directory is always read-only and contains the RobotWare installation along with the RobotWare add-ins. |

To make sure that all RAPID programs and URL-s in the Robot Web Service API work properly, make sure that you only use the locations specified above (using the predefined environment variables).



Note

RobotWare add-ins can add additional directories and files to the default base directory structure of your controller, so the actual default structure on your controller may have additional files and directories.

This page is intentionally left blank

2 Navigating and handling the FlexPendant

2.1 Overview

Introduction to this section

The content in this section applies to a robot and not a robot system. It is the responsibility of the integrator to provide a safety and users manual for the robot system.

This section provides information about basic parts of the user interface. The important elements for navigation is illustrated in [FlexPendant touch screen on page 30](#).

Handling and troubleshooting the FlexPendant

How to handle, clean, and troubleshoot the FlexPendant is described in the product manual for the controller.

Hardware and software options

This manual covers only the views of a basic RobotWare system. The details of the various options are explained in the application manuals.

2 Navigating and handling the FlexPendant

2.2.1 FlexPendant touch screen

2.2 The user interface

2.2.1 FlexPendant touch screen

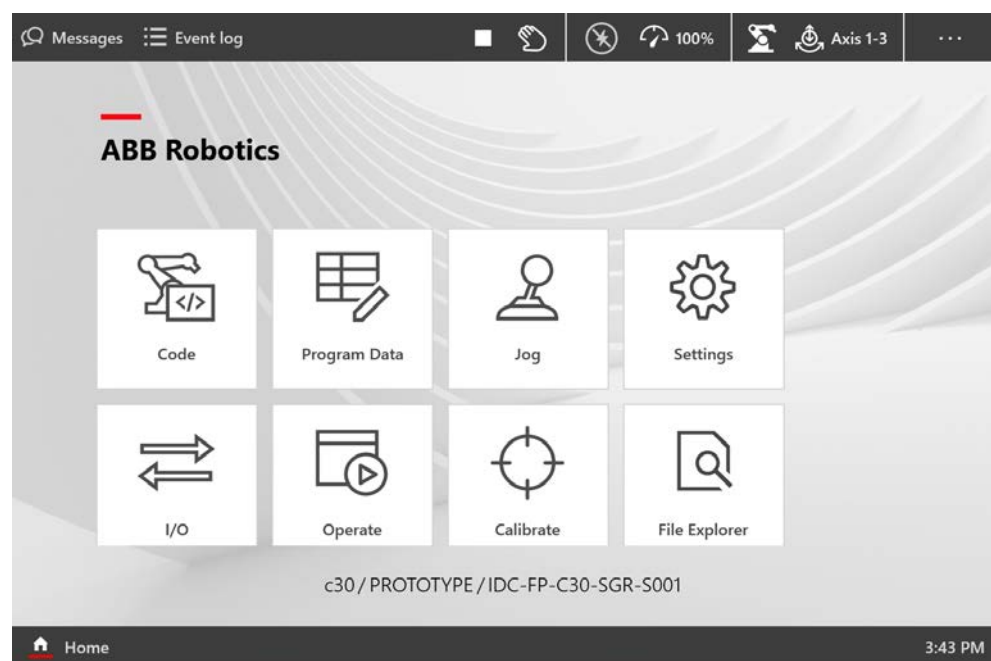
Overview

The FlexPendant touch screen consists of a status bar at the top, Task bar with Home button at the bottom, and also a set of applications.



Note

Virtual FlexPendant has an additional panel. For more details, see [UI elements in Virtual FlexPendant panel on page 30](#).



xx1900000917

The details about FlexPendant applications are described in the section [FlexPendant applications on page 45](#).

Active FlexPendant applications are displayed on the task bar at the bottom. To switch between the active applications, tap on the respective application in the task bar. To close an application, press and hold the application in the task bar and tap on the displayed close button.

UI elements in Virtual FlexPendant panel

The Virtual FlexPendant has some extra soft buttons and UI elements. These correspond to the hardware buttons on the FlexPendant. This section provides details about those soft buttons and UI elements available only in the panel of

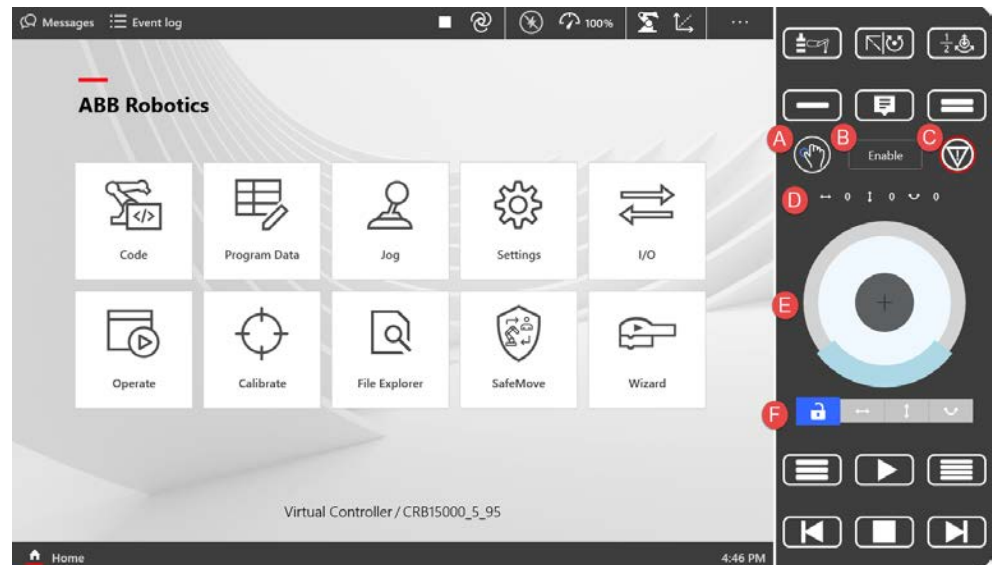
Continues on next page

2 Navigating and handling the FlexPendant

2.2.1 FlexPendant touch screen

Continued

Virtual Flexpendant. The details of the FlexPendant buttons and parts are available in the section [Main parts on page 20](#).



xx2100000288

| | | |
|---|-----------------------|---|
| A | Thumb button | Enables the lead-through functionality. |
| B | Enable/Release button | Provides the same function as the three-position enabling device. |
| C | Emergency stop button | Stops the system with an emergency stop. |
| D | X Y Z values | Displays instantaneously the difference in x,y,z axis values while jogging the robot using joystick. |
| E | Virtual Joystick | Move the central part of virtual joy stick to jog the robot along X and Y axes. Move the semi-circular bar along the circumference to jog the robot along Z axis. |
| F | Axes bar | Locks all axes or individual axis. |

Continues on next page

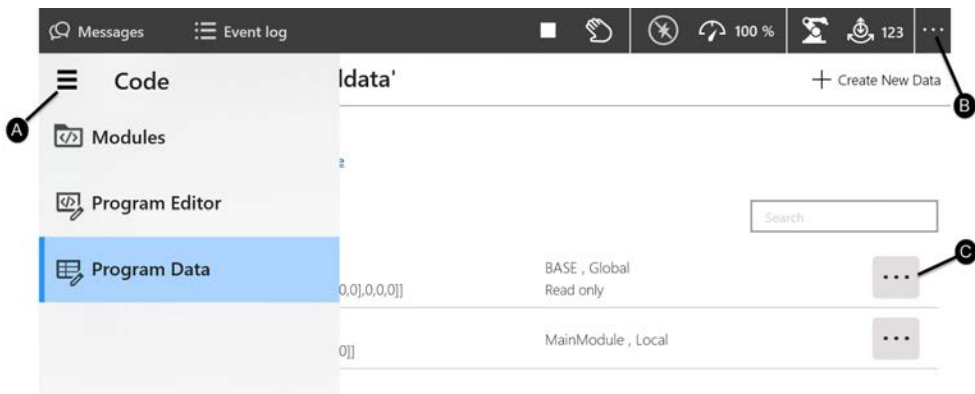
2 Navigating and handling the FlexPendant

2.2.1 FlexPendant touch screen

Continued

Menus

The following figure and table provides an overview of the menus that are available in the FlexPendant interface.



xx1900002308

| | | |
|---|-------------------------|---|
| A | Application menu button | The Application menu button is available when you select one of the applications on the FlexPendant start screen. The menu content reflects the selected application. |
| B | QuickSet menu button | The QuickSet menu button displays the QuickSet window. |
| C | Context menu button | The context menu button displays a menu with available options for a selected row. |

2.2.2 Status bar

2.2.2.1 Introduction

Overview

The following figure and table provides an overview of the user interface elements that you can access from the status bar of the FlexPendant.



xx1800000600

| Label | Buttons | Description |
|-------|-------------------|---|
| A | Messages button | Tapping the Messages button displays the operator messages window. The operator messages window displays the messages from robot programs. This usually happens when the program needs some kind of operator response in order to continue. For more details see Overview on page 34 |
| B | Event logs button | Tapping the Event logs button displays the event logs window. The event logs window displays all the event logs. The list allows you to filter the event log list according to the category. |
| C | QuickSet button | The icons on the right side of status bar represents the current status for important settings. The description of each of these icons is provided in the section Description of the icons in the QuickSet button on page 35 . Tapping on any icon in this region displays the QuickSet window along with the status details. For more details, see QuickSet window tabs on page 36 . |

2 Navigating and handling the FlexPendant

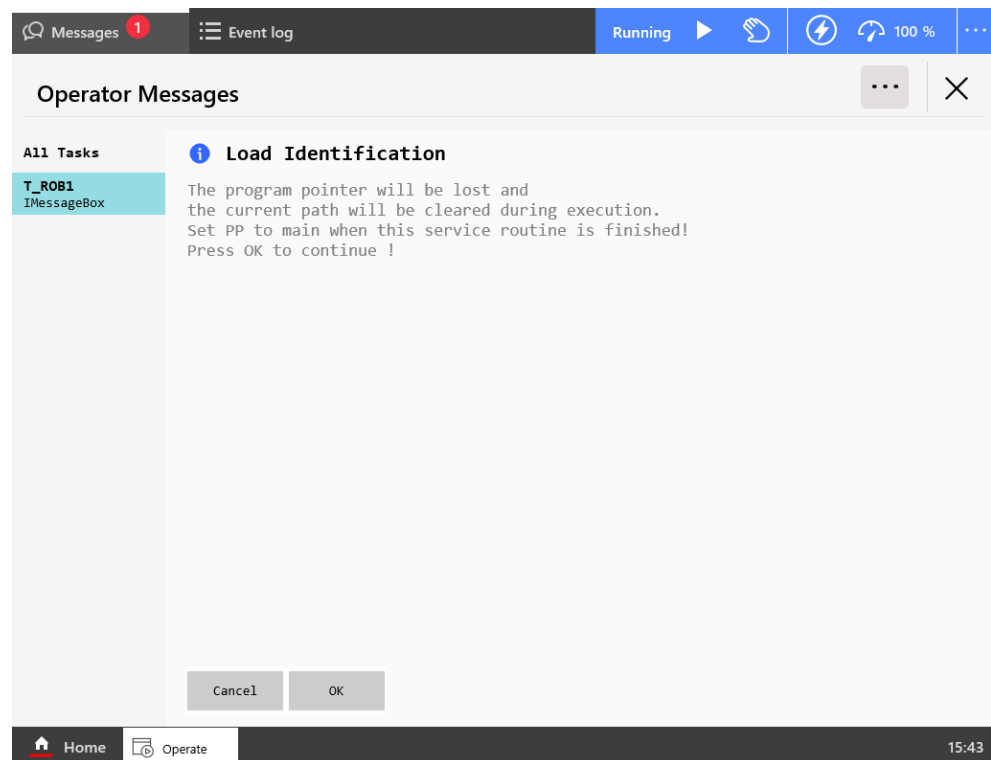
2.2.2.2 Messages window

2.2.2.2 Messages window

Overview

The operator messages window displays messages from the program. With *Multitasking* installed, all task messages are displayed in the same operator window. If a message requires action, for example a service routine message, then a separate window for that task is displayed.

The operator window is opened by tapping the **Messages** button on the status bar. The following illustration displays an example of an operator message:

















xx2000000866

2.2.2.3 QuickSet

Description of the icons in the QuickSet button

QuickSet button in the status bar has various icons which indicates the current status of various parameters. The following table provides a description of each of these icons:





| Section | Icon | Description |
|-------------------------|---|--|
| Operating status |  | Indicates that the program is running. |
| |  | Indicates that the program is stopped. |
| |  | Indicates that the operating mode is Auto. |
| |  | Indicates that the operating mode is Manual. |
| |  | Indicates that the operating mode is Manual Full Speed. |
| Motors and speed status |  | Indicates that the motors are ON. |
| |  | Indicates that the motors are OFF. |
| |  | Indicates the running speed of the robot. The value of speed can be between 0% to 100%. |
| Jog status |  | Indicates the active mechanical unit.  Note The Conveyor icon is displayed here if that mechanical unit is selected. |
| |  | Indicates that the jog mode is Axis 1-3.  Note The Axis 4-6 icon is displayed here if that jog mode is selected. |
| |  | Indicates that the jog mode is Linear. |
| |  | Indicates that the jog mode is Reorient. |

Continues on next page

2 Navigating and handling the FlexPendant

2.2.2.3 QuickSet

Continued

| Section | Icon | Description |
|-----------------------|---|------------------------|
| Increment mode status |  Small | Small movements |
| |  Medium | Medium movements |
| |  Large | Large movements |
| |  User | User defined movements |

QuickSet window tabs

Overview

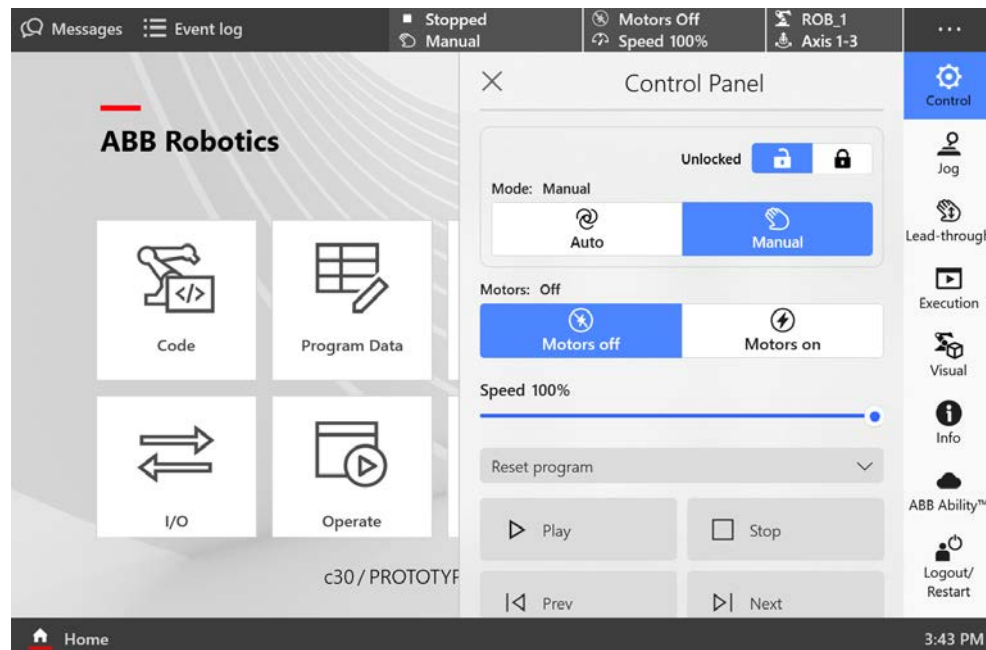
When you tap on the QuickSet button the QuickSet window is displayed. The QuickSet window has the following tabs:

- Control
- Jog
- Execution
- Visual
- Information
- Connected Services
- Logout/Restart




Continues on next page

Control Panel

The **Control** tab displays the **Control Panel**. The **Control Panel** allows you to change the operating modes and turn the motors on and off. It also displays the program control buttons.



xx1900000622

| Button/Section | Description |
|---------------------|---|
| Lock/Unlock buttons | Locks or unlocks the operating mode. For more details, see Locking and unlocking operating modes on page 79 . |
| Mode | <p>Changes the operating mode.</p> <p> (Auto mode): Select this mode to move the manipulator or run a program without human intervention. In this mode the safety function of the three-position enabling device is bypassed.</p> <p> (Manual mode): Select this mode to move the manipulator or run a program with human intervention. In this mode to make the motors on, select the Motors on button from the Motors section. You can also press the three-position enabling device to activate the motors.</p> <p> (Manual FS or Manual Full Speed mode): Select this mode to run the program at full speed while still having access to all the debugging functions available in the program editor.</p> |
| Motors | <p>Turns the motors on or off.</p> <p>Motors off: Turns the motors off.</p> <p>Motors on: Turns the motors on.</p> |
| Speed | Controls the speed of program execution. Drag the scroll bar to control the speed. The speed of 100% indicates that the program is running at full speed. |
| Reset Program | Sets the program pointer to main. |
| Cancel routine | Cancels the execution of a service routine. |

Continues on next page

2 Navigating and handling the FlexPendant

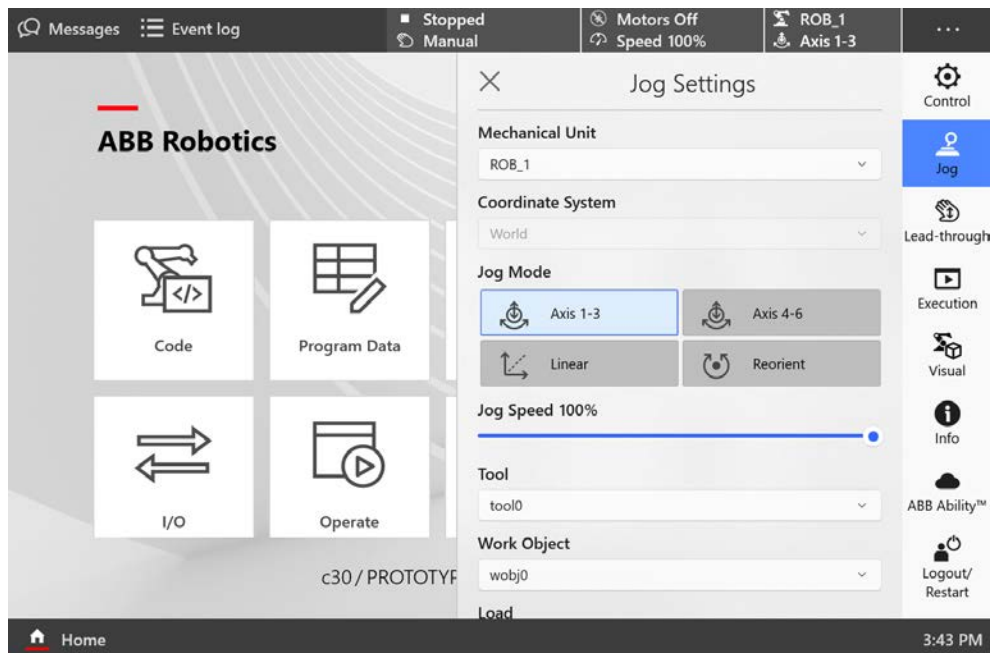
2.2.2.3 QuickSet

Continued

| Button/Section | Description |
|----------------|------------------------------------|
| Play | Starts the program execution. |
| Stop | Stops the program execution. |
| Prev | Executes one instruction backward. |
| Next | Executes one instruction forward. |

Jog Settings

The Jog tab displays the Jog Settings.



xx1900000623

| Section/List/Button | Description |
|---------------------|--|
| Mechanical Unit | Selects a mechanical unit from the list. |
| Coordinate System | Selects a coordinate system from the list. |
| Jog Mode | Selects a Jog mode from the section. |
| Jog Speed | Controls the jog speed. Drag the scroll bar to set the jog speed. The speed of 100% indicates that the jogging is at full speed. |
| Tool | Selects a tool from the list. |
| Work Object | Selects a work object from the list. |
| Load | Selects a load from the list. |
| Incremental mode | Jogs the robot in small steps, which enables very precise positioning. |

Lead-through

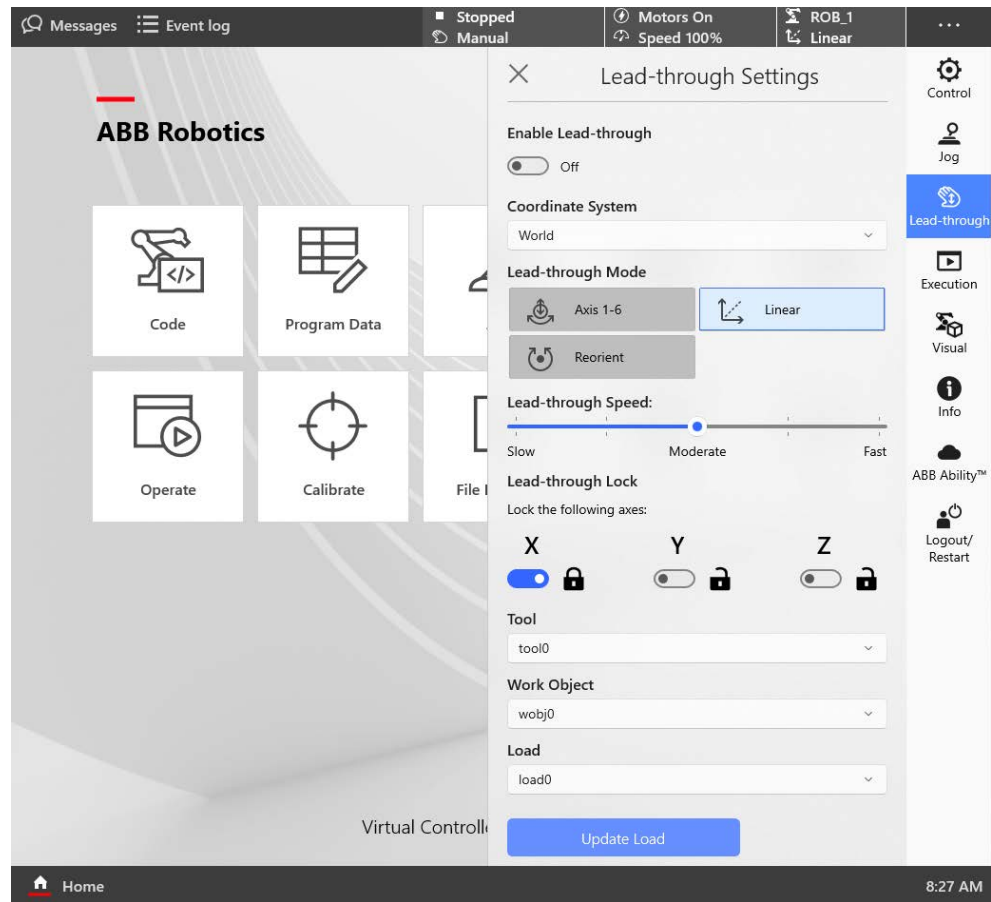
The Lead-through tab displays the settings required for the lead-through functionality.




Note

The Lead-through tab is available only for the CRB 15000 robot.

Continues on next page



xx2200000306


| Section/List/Button | Description |
|--------------------------------------|--|
| Enable Lead-through | Enables or disables the lead-through functionality. |
| Coordinate System | Selects a coordinate system from the list. |
| Lead-through Mode | Selects a Lead-through mode from the list. |
| Lead-through Speed scroll bar | Controls the Lead-through speed. Drag the scroll bar to set the Lead-through speed. The speed of 100% indicates that the Lead-through jogging is at full speed. |
| Lead-through Lock | <p>Lock or unlocks the movement of an axis by enabling or disabling the lock button next to the axis.</p> <p> Note</p> <p>The Lead-through lock section is disabled for the Axis 1-6 mode.</p> |
| Tool | Selects a tool from the list. |
| Work Object | Selects a work object from the list. |
| Load | Selects a load from the list. |

Continues on next page

2 Navigating and handling the FlexPendant

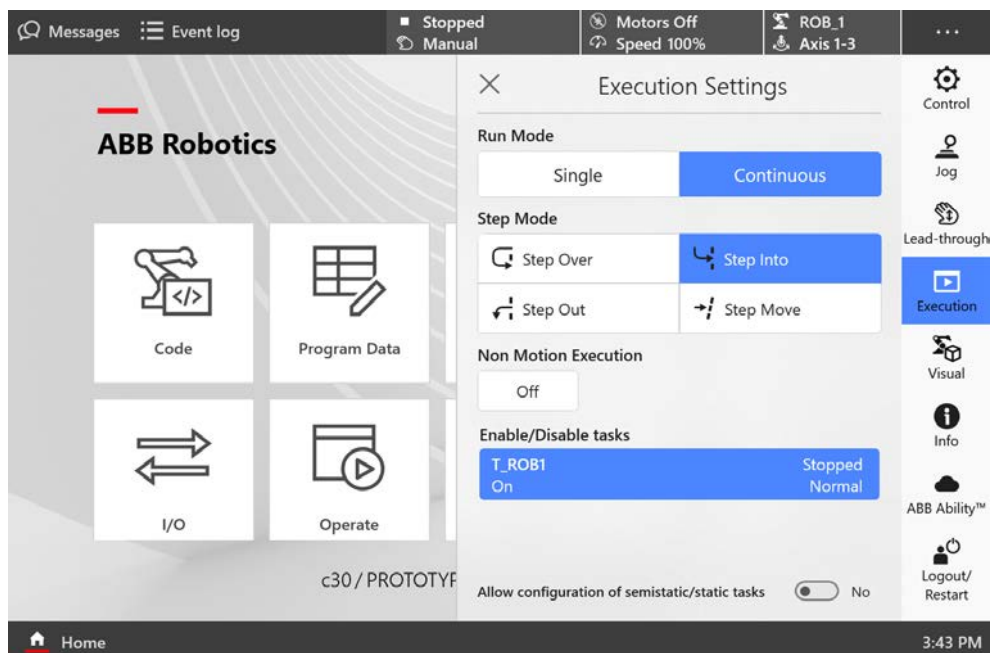
2.2.2.3 QuickSet

Continued

| Section/List/Button | Description |
|---------------------|--|
| Update Load | <p>Refreshes the mass of the current active load (the mass as measured by the controller).</p> <p>The Update Load button is enabled only if all the following conditions are valid:</p> <ul style="list-style-type: none">the robot is CRB15000.the logged in user has Edit RAPID code and Modify position grants.the Motors are on. <p> Note</p> <p>When you press the Update Load button, the lead-through function will be turned off (if it is active). Lead-through will be reactivated once the operation is complete(if it was active before).</p> |

Execution Settings

The Execution tab displays the Execution Settings.



xx1900000624

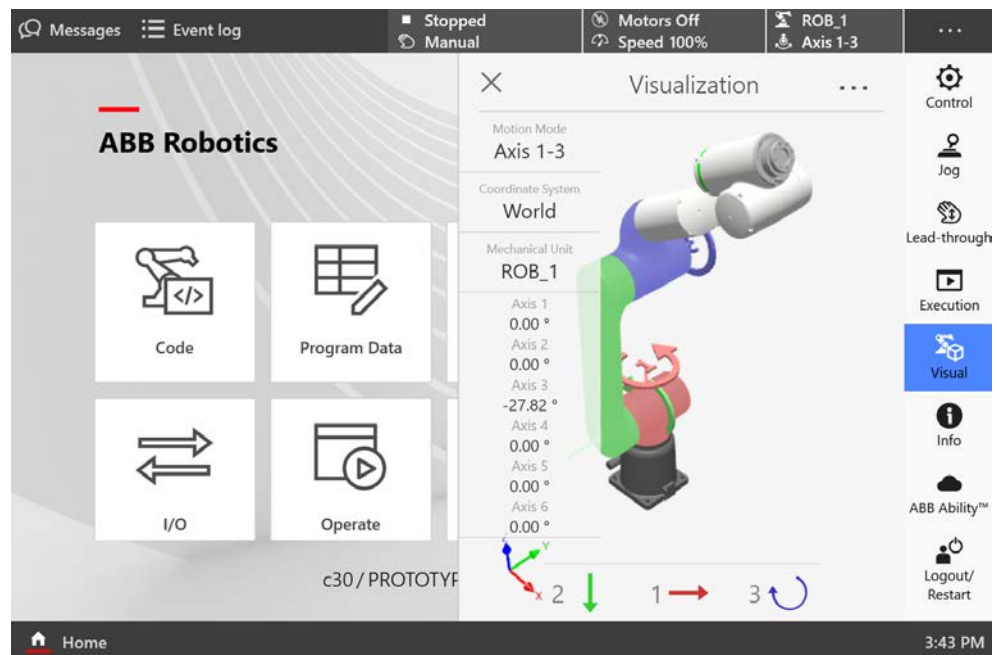
| Button/Section | Description |
|-----------------|--|
| Run Mode | <p>Selects the program run mode.</p> <ul style="list-style-type: none">Single: Runs one cycle and then stops the execution.Continuous: Runs continuously. |

Continues on next page

| Button/Section | Description |
|-----------------------------|--|
| Step Mode | <p>Selects the program step mode.</p> <ul style="list-style-type: none"> Step Over: The called routines are executed in one single step. Step Into: Steps into the called routines and executes them step-by-step. Step Out: Executes the remaining part of the current routine and then stops at the next instruction in the routine from which the current routine was called. Not possible to use in the Main routine. Step Move: Steps to the next move instruction. Stops before and after movement instructions, for example, to modify positions. |
| Non Motion Execution | This is used to run a RAPID program without the robot motion. |
| Enable/Disable tasks | This is used to enable or disable the selected task. |

Visualization

The Visual tab displays the visualization details based on the actual robot movement.



xx1900000625

| Section | Description |
|--------------------------|--|
| Motion Mode | Displays the selected motion mode. |
| Coordinate System | Displays the selected coordinate system. |
| Mechanical Unit | Displays the selected mechanical unit. |
| Axes | Displays the position of each axis. |
| Colored arrows | Displays the direction in which the jog stick needs to be moved for jogging the selected axis. |

Continues on next page

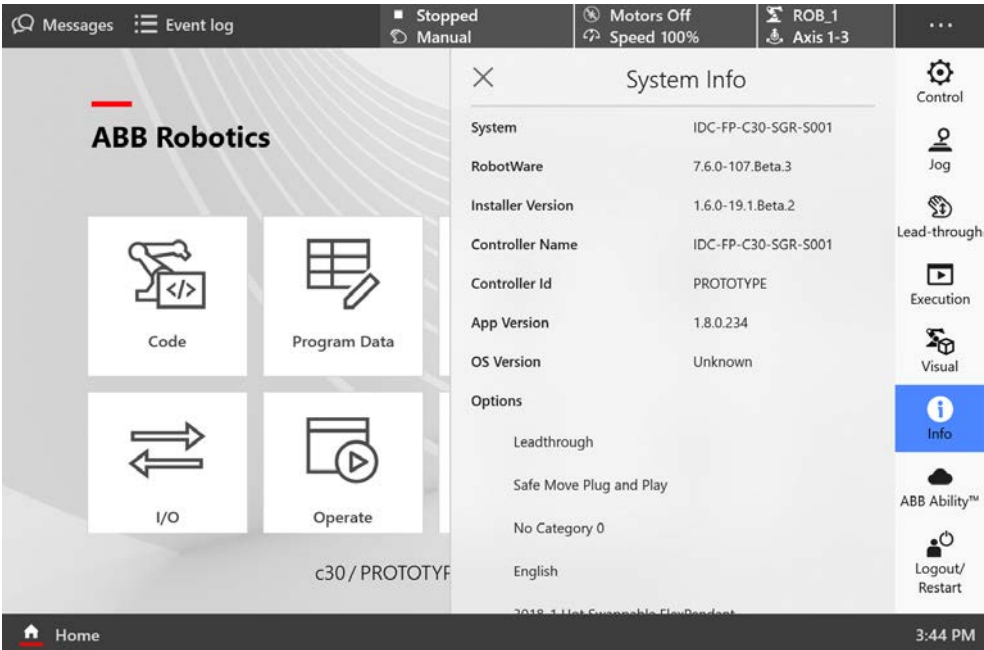
2 Navigating and handling the FlexPendant

2.2.2.3 QuickSet

Continued

Info

The Info tab displays the information about the system.



xx1900000626

| Section | Description |
|---------------|--|
| System | Displays the name of the system. |
| RobotWare | Displays the version number of the RobotWare selected in the system. |
| IP Address | Displays the IP address of the management port. |
| Controller Id | Displays the identity of the controller. |
| App version | Displays the version information of the FlexPendant application. |
| Options | Displays the options selected in the system. |

Connected Services

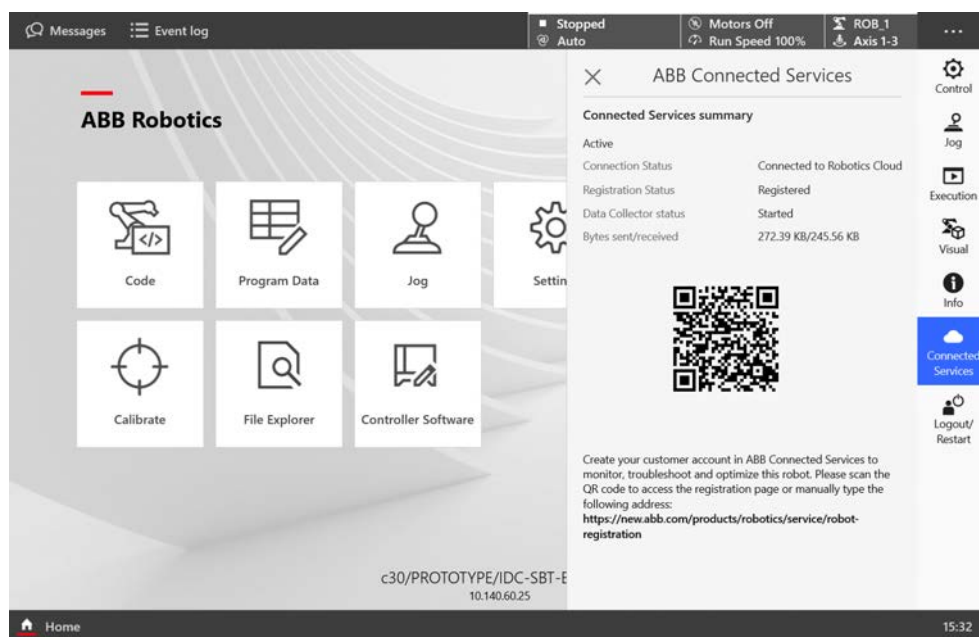


Note

ABB Connected Services is the new name for the functionality previously known as ABB Ability. During a period of time, both names will appear in and on our products.

The **Connected Services** tab displays information about the Connected Services in the **Connected Services Summary** section.

Continues on next page



xx2000000132

| Button/Section | Description |
|-----------------------|--|
| Status | Displays whether the connected service connection is active or not. |
| Connection Status | Displays the name of the server where Connected Services is connected. |
| Registration Status | Displays whether the Connected Services is registered in the server or not. |
| Data Collector Status | Displays whether the data collector is started or not. |
| Bytes sent/received | Displays the amount of bytes sent or received. |
| QR code | Scan the QR code to navigate to https://new.abb.com/products/robotics/service/connected-services , where you can contact ABB to create a customer account for ABB Connected Services. |

Feedback

The **Feedback** tab displays a QR code that can be used to submit feedback online about the interface.

Continues on next page

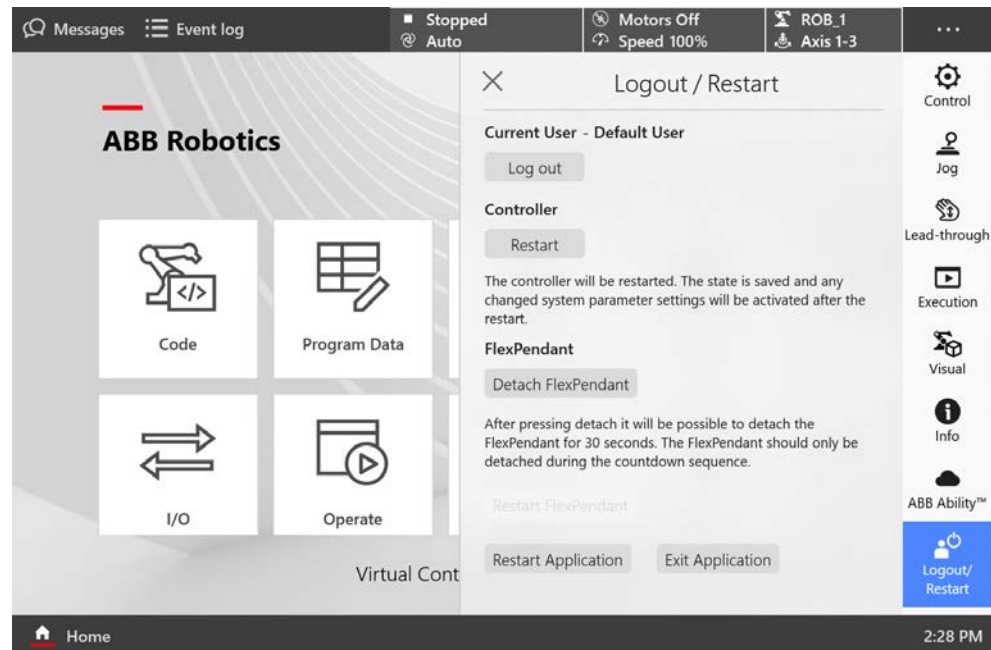
2 Navigating and handling the FlexPendant

2.2.2.3 QuickSet

Continued

Logout/Restart

The Logout/Restart tab allows you to disconnect the FlexPendant, restart the controller and FlexPendant, and so on.



xx1900000627

| Button/Section | Description |
|--------------------|---|
| Current User | Displays the name of the current logged in user. Log out: Logs out the current user from the FlexPendant. |
| Controller | Restart: This is used to restart the controller. |
| FlexPendant | Detach FlexPendant: This is used to detach the FlexPendant from the controller without stopping the current program execution. For more details, see Detaching and attaching a FlexPendant on page 252 . Restart FlexPendant: This is used to restart the FlexPendant. Restart Application: This is used to restart the FlexPendant application. Exit Application: This is used to exit the FlexPendant application. |
| Screen orientation | Rotate 180°: This is used to rotate the FlexPendant screen by 180°. This functionality is useful for left hand users. For more details, see the section Adapting the FlexPendant for left-handed users on page 64 . |

2.2.3 FlexPendant applications

The FlexPendant applications

The software on the FlexPendant contains applications for controlling the robot. There are different application packages depending on the options selected for the robot. The *Limited App Package* is always included, unless another app package is selected.

There are more applications available than those listed below. These can be specific for the selected products and options, for example, application software, or applications for controlling grippers and tools.

Code

The **Code** application is used to create new programs, modify existing programs, and so on.

| Feature | <i>Limited App Package</i> [3120-1] | <i>Essential App Package</i> [3120-2] | <i>Program Package</i> [3151-1] |
|---|-------------------------------------|---------------------------------------|---------------------------------|
| Create new programs, edit existing programs | | | ✓ |
| View and edit RAPID modules and RAPID routines | | | ✓ |
| Debug Options PP to main, cursor to program pointer, goto position, call routine, cancel routine, check program, view system data, next move instruction | | | ✓ |
| Teach position (ModPos) | | | ✓ |
| Check for syntactic and semantic error | | | ✓ |

If the option *Program Package* is not selected then programs must be created and edited using RobotStudio.

Program Data

The **Program Data** application is used to view and edit RAPID data.

| Feature | <i>Limited App Package</i> [3120-1] | <i>Essential App Package</i> [3120-2] | <i>Program Package</i> [3151-1] |
|---|-------------------------------------|---------------------------------------|---------------------------------|
| View and edit RAPID data (program data) | | | ✓ |
| Manage payload data | ✓ | ✓ | |
| Manage tool data | ✓ | ✓ | |
| Manage work object data | ✓ | ✓ | |

Continues on next page

2 Navigating and handling the FlexPendant

2.2.3 FlexPendant applications

Continued

Jog

The **Jog** application is used to jog the ABB industrial robot using an intuitive touch based user interface or using a joystick.

| Feature | Limited App Package [3120-1] | Essential App Package [3120-2] | Program Package [3151-1] |
|----------------------|------------------------------|--------------------------------|--------------------------|
| Joystick jog | ✓ | ✓ | |
| Touch jog | | ✓ | |
| Align tool | | ✓ | |
| Lead-through | ✓ ⁱ | ✓ ⁱ | |
| Jog supervision | ✓ | ✓ | |
| GoTo (jog to target) | | ✓ | |
| 3D visualization | ✓ | ✓ | |

ⁱ Only applicable for compatible manipulators, currently IRB 14050, CRB 15000 and CRB 1810.

Settings

The **Settings** application is used to configure the general settings of OmniCore controller and FlexPendant. Controller configuration includes Network, ABB Connected Services, Time and Language, Backup, Restore, System diagnostics and so on. FlexPendant configuration includes background settings and programmable keys.

| Feature | Limited App Package [3120-1] | Essential App Package [3120-2] | Program Package [3151-1] |
|--|------------------------------|--------------------------------|--------------------------|
| System About, hardware devices, software resources | ✓ | ✓ | |
| Network Status, WAN settings, DNS Client | ✓ | ✓ | |
| ABB Connected Services Status, Connected Services status, configure 3G/wired Status, Connected Services status, configure 4G/3G/wired | ✓ | ✓ | |
| Configure Connected Services | ✓ | ✓ | |
| Backup and Recovery Backup, restore, system diagnostics, restart, reset user data, RobotWare Installation Utilities | ✓ | ✓ | |
| Date & time | ✓ | ✓ | |
| Region & language | ✓ | ✓ | |
| Programmable keys | ✓ | ✓ | |

Continues on next page

I/O

The I/O application is used to manage the I/O signals. Signals are configured with system parameters.

| Feature | Limited App Package [3120-1] | Essential App Package [3120-2] | Program Package [3151-1] |
|--|------------------------------|--------------------------------|--------------------------|
| Show industrial networks | ✓ | ✓ | |
| View all I/O signals | ✓ | ✓ | |
| Display I/O signals with respect to category | ✓ | ✓ | |
| Filter signals | ✓ | ✓ | |
| Sort signals | ✓ | ✓ | |
| Set signals | ✓ | ✓ | |
| Bit values | ✓ | ✓ | |
| Navigate to device specific signals | ✓ | ✓ | |
| Identify device | ✓ | ✓ | |
| Scan EDS | ✓ | ✓ | |
| Activate and deactivate devices | ✓ | ✓ | |
| Start | ✓ | ✓ | |
| Scan | ✓ | ✓ | |
| Firmware upgrade | ✓ | ✓ | |

Operate

The **Operate** application is used to view the program code while the program is running. Controller data can be configured for viewing the data in the form of dashboards. Updates during production are shown here.

| Feature | Limited App Package [3120-1] | Essential App Package [3120-2] | Program Package [3151-1] |
|--|------------------------------|--------------------------------|--------------------------|
| View dashboards | | ✓ | |
| Configure dashboards | | ✓ | |
| Load and execute RAPID programs | ✓ | ✓ | |
| View loaded RAPID programs | ✓ | ✓ | |
| Teach position (ModPos) of robtargets in loaded RAPID programs | ✓ | ✓ | |
| Reset program pointer to Main | ✓ | ✓ | |
| Show program pointer position | ✓ | ✓ | |
| Show motion pointer position | ✓ | ✓ | |
| Execute service routines | ✓ | ✓ | |

Continues on next page

2 Navigating and handling the FlexPendant

2.2.3 FlexPendant applications

Continued

Calibrate

The **Calibrate** application is used for calibration and definition of frames for ABB robots.

| Feature | Limited App Package [3120-1] | Essential App Package [3120-2] | Program Package [3151-1] |
|---|------------------------------|--------------------------------|--------------------------|
| Mechanical unit calibration | ✓ | ✓ | |
| Update revolution counters | ✓ | ✓ | |
| Edit motor offset values | ✓ | ✓ | |
| Load motor offset values | ✓ | ✓ | |
| Fine calibration | ✓ | ✓ | |
| Robot memory | ✓ | ✓ | |
| Base frame calibration | ✓ | ✓ | |
| Execute calibration specific service routines | ✓ | ✓ | |

File Explorer

The File Explorer is a file manager, similar to Windows Explorer, with which you can view, rename, delete, or move files and folders on the controller or on a connected external USB memory.



Note

The file explorer supports operations on the following file formats: TXT, CFG, PNG, XML, ZIP, JPG, MOD, PGF, LOG, and MODX.

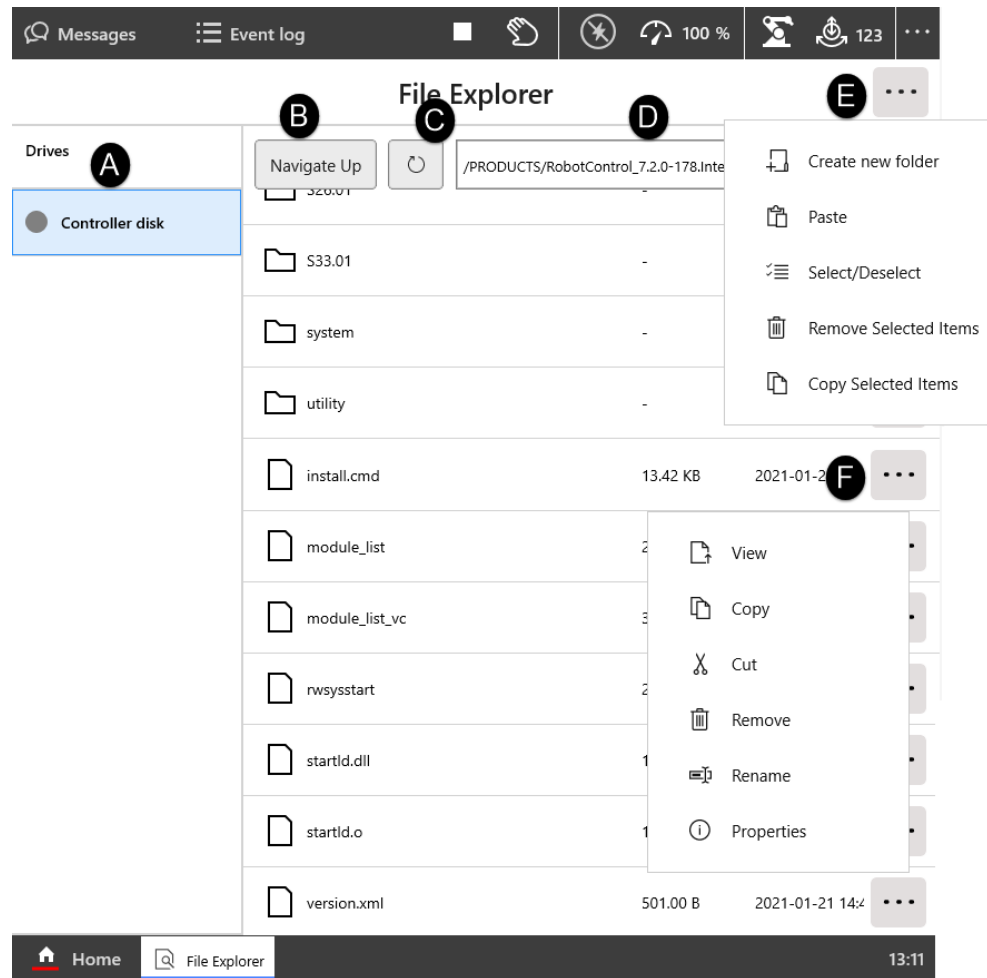
To manage files and folders, from the Home screen, open **File Explorer**. The file explorer window is displayed. The following image and table provides information regarding the functions available in the file explorer window.

Continues on next page

2 Navigating and handling the FlexPendant

2.2.3 FlexPendant applications

Continued



xx2100000050

| Label | Description |
|-------|---|
| A | Displays the available drives. If a USB memory is connected to the FlexPendant that is also displayed here. |
| B | Navigates to the folder up by one level. |
| C | Refreshes the files and folders. |
| D | Displays the path of the selected folder. |
| E | Displays the options available for a selected folder. <ul style="list-style-type: none">• Create new folder: Creates a new folder in the selected folder.• Paste: Pastes the copied files or folders in the selected folder.• Select/Deselect: Selects or clear the selection for a set of files and folders.• Remove Selected Items: Removes the selected items.• Copy Selected Items: Copies the selected items. |
| F | Displays the options available for a selected item. <ul style="list-style-type: none">• View: Allows you to view the selected text or picture files.• Copy: Copies the selected item.• Cut: Cuts the selected item.• Remove: Deletes the selected item.• Rename: Changes the name of the selected item.• Properties: Displays the properties of the selected item. |

Continues on next page

2 Navigating and handling the FlexPendant

2.2.3 FlexPendant applications

Continued



Note

The following grants are required for full access to controller disk:

- **Read access to controller disks**
- **Write access to controller disks**

Without the **Read and Write access to controller disks** grant you may get access to some folders in controller disk like `/TEMP` but not all of them.

While moving the file and folders following are the possible scenarios:

- Moving files and folders within the controller disk.
- Moving files and folders from controller to USB memory and vice versa.



Note

It is not possible to move or copy files and folders within a USB memory.

SafeMove

The application **SafeMove** is used to configure some parts of SafeMove. See *Application manual - Functional safety and SafeMove*. For full SafeMove configuration, see *Visual SafeMove* in RobotStudio.

Web apps

Web apps running on the FlexPendant use the EdgeHTML web engine by default. It is possible to change to Chromium web engine from **Settings > FlexPendant > Web App Mode**.

2.3 Personalizing the FlexPendant

2.3.1 Introduction

Overview

The FlexPendant can be personalized in a number of ways. This is described in this chapter.

2 Navigating and handling the FlexPendant

2.3.2 System information

2.3.2 System information

Information

The information about the selected hardware devices, software resources, services, and the features can be accessed from the **System** page of the **Settings** application.



Note

The only information that you can change from the System information page is to edit the robot system name. For more details, see [Configuring the identity settings on page 53](#).

Use the following procedure to access the system information:

- 1 On the start screen, tap **Settings**.
- 2 Tap **System**.

The **About** page displayed. On the left sidebar tap the **Hardware Devices** and **Software Resources** to access the respective information.

2.3.3 Configuring the identity settings

Configuring the identity settings

You can define a name for the robot system according to your requirement.

Use the following procedure to configure the name of the robot system.

- 1 On the start screen, tap **Settings**.
- 2 Tap **System**.
- 3 In the **Robot System Name** field tap **Edit**.
- 4 Type or update the name of the robot.
- 5 Tap **Apply**.

The changes are saved and the name of the robot system is updated.

2 Navigating and handling the FlexPendant

2.3.4 Configuring date and time

2.3.4 Configuring date and time

Overview

It is possible to configure the date and time from the FlexPendant.



Note

When you change the Date and time using the following procedure the controller clock is modified, not the FlexPendant clock.

Procedure

Use the following procedure to configure the date and time.

- 1 On the start screen, tap **Settings**.
- 2 Tap **Time & Language**.
- 3 On the left sidebar tap **Date & Time**.
- 4 Select **Network Time** or **Manual Time**.
 - **Network Time**: Select this for configuring the robot controller for automatic time synchronization using the NTP protocol of a time server. The time server is identified by its IP address or DNS name.
 - **Manual Time**: Select this if you do not have a time server that is reachable from the controller.
- 5 If you select **Network Time**, in the **Time Server Address** field type the URL or IP address of the time server.



Note

Tap the **Test** button next to the **Time Server Address** field to verify the time server address.

- 6 If you select **Manual Time**, in the **Select date** and **Select time** fields manually configure date and time.
- 7 In the **Time Format** section, select the format of the time as **24-hour** or **12-hour**.



Note

The default time format is **24-hour**.

- 8 Tap **Apply**.
The selected settings are saved.



Note

It is important to correctly define the Time Zone including the country and sub Zone, as this is used for different digital services.

Continues on next page



Note

You can set the format of the time as **24-hour** or **12-hour** from the **Time Format** section. The default time format is **24-hour**.

2 Navigating and handling the FlexPendant

2.3.5 Configuring the interface language

2.3.5 Configuring the interface language

Overview

The FlexPendant user interface is available in twenty languages. The default language is English. However, it is possible to change the interface language after logging in to the controller.



Note

When you switch to another language, all the buttons, menus, and dialogs will use the new language. But the interface language of RAPID instructions, variables, system parameters, and I/O signals is not affected.



Note

The number of interface languages that FlexPendant supports depends on the RobotWare version the system is using.

Configuring interface language after logging in to the controller

Use the following procedure to change the interface language:

- 1 On the start screen, tap **Settings**.
- 2 Tap **Time & Language**.
- 3 On the left sidebar, tap **Language**.
A list of supported user interface languages is displayed.
- 4 Tap on the desired language button.
- 5 Tap **Apply**.
The **Change language** confirmation window is displayed.
- 6 Tap **Yes**.
The FlexPendant is restarted and the interface language is changed to the selected language.

2.3.6 Configuring the default view during system events

Overview

This feature is used to configure a particular app view during start up or when switching to different operating modes.



Note

This configuration is disabled when the operating mode is **Auto**.

Procedure

Use the following procedure to configure the default view of FlexPendant for system events:

- 1 On the start screen, tap **Settings**.
- 2 Tap **FlexPendant**.
- 3 On the left sidebar, tap **Launch on System Event**.
The **Launch on System Event** page is displayed.
- 4 For a system event, tap the list and select a desired view.

Following system events are available :

- **Switching to Auto**: Select this option to define the FlexPendant view when switching to auto mode.
- **Switching to Manual**: Select this option to define the FlexPendant view when switching to manual mode.
- **Switching to Manual Full Speed**: Select this option to define the FlexPendant view when switching to manual full speed mode.
- **FlexPendant Startup**: Select this option to define the view when the FlexPendant starts.



Note

Tap **Clear** to remove the current selected view.

- 5 Tap **Save**.
The changes are saved.

2 Navigating and handling the FlexPendant

2.3.7 Configuring the Joystick

2.3.7 Configuring the Joystick

Overview

This feature is used to configure the desired response of the joystick while using the joystick jogging.

Procedure

Use the following procedure to configure the desired response of the joystick:

- 1 On the start screen, tap **Settings**.
- 2 Tap **FlexPendant**.
- 3 On the left sidebar, tap **Joystick Configuration**.
The **Joystick Response** page is displayed.
- 4 In the **Joystick mode** section, select a mode for the Joystick.
Following modes are available:
 - **Linear**: The jog speed is directly proportional to the deflection of the joystick.
 - **Progressive**: The jog speed is calculated from the deflection using a progressive function. For smaller deflections, the jog speed will be low and increases slowly. The speed then ramps up quickly when getting closer to 100% deflection. This makes it easier to jog the robot slowly.
- 5 Tap **Apply**.
The Joystick settings are saved.

2.3.8 Configuring the display of event logs

Overview

This feature is used to configure the display of event logs in FlexPendant.

Procedure

Use the following procedure to configure the event log display:

- 1 On the start screen, tap **Settings**.
- 2 Tap **FlexPendant**.
- 3 On the left sidebar, tap **Event Log**.
The **Event Log** page is displayed.
- 4 In the **Error Focus Handling** section, select an option for the display of event log messages.

Following are the available options:

- **Fewer interruptions:** Only the important event logs will be displayed.
- **Verbose:** All the event logs will be displayed.

The event logs settings are saved.

2 Navigating and handling the FlexPendant

2.3.9 Configuring the display of Controller ID and System Name

2.3.9 Configuring the display of Controller ID and System Name

Overview

This feature is used to change the display format of Controller ID and System Name in Home Screen.

Procedure

Use the following procedure to configure the display format of Controller ID and System Name:

- 1 On the start screen, tap **Settings**.
- 2 Tap **FlexPendant**.
- 3 On the left sidebar, tap **Controller Id and System Name**.
The **Controller Id and System Name** page is displayed.
- 4 In the **Select the preferred naming option** section, select an option for the display of Controller ID and System Name.

Following are the available options:

- **Both Controller Id and System Name (Default):** Displays both the Controller ID and System Name. This is the default display format.
- **Controller Id only:** Displays only the Controller ID.
- **System Name only:** Displays only the System Name.

The settings are saved.

2.3.10 Configuring the default paths

Overview

This feature is used for setting individual default paths for the following actions:

- Saving and loading RAPID modules.
- Saving and loading RAPID programs.

Procedure

Use the following procedure to configure the default paths:

- 1 On the start screen, tap **Settings**.
- 2 Tap **FlexPendant**.
- 3 On the left sidebar, tap **Default Path**.
The **Define default paths to the controller file system** page is displayed.
- 4 Tap **Browse** and select the required path for the following options:
 - **Select folder to set default path for RAPID modules:** Allows you to configure a default path for RAPID modules.
 - **Select folder to set default path for RAPID programs:** Allows you to configure a default path for RAPID programs.
- 5 Tap **Save**.
The default path settings are saved.

2 Navigating and handling the FlexPendant

2.3.11 Configuring the programmable keys

2.3.11 Configuring the programmable keys

Overview

Programmable keys are the four hardware buttons on the FlexPendant that can be used for dedicated, specific functions configured by the user.

The keys can be programmed to simplify programming or to test the programs. They can also be used to activate menus on the FlexPendant.



Note

It is not mandatory to configure all the programmable keys.

Procedure

Use the following procedure to configure the programmable keys.

- 1 On the start screen, tap **Settings**.
- 2 Tap **Personalization**.
- 3 On the sidebar tap **Programmable keys**.
- 4 Tap on a key.
- 5 Type a name for the selected key in the **Friendly Name** field.



Note

This field is optional.

- 6 Tap the **Type** list and select the type of action.
- 7 If the type **Input** is selected:
 - Tap to select an input signal from the **Digital Input** list.
 - Tap the **Allow in auto** list to select if the function is also allowed in automatic operating mode.



Note

An input cannot be set by using the programmable keys, its value can only be pulsed. The pulse will be the inverted value of the input.

- 8 If the type **Output** is selected:
 - Tap to select an output signal from the **Digital Outputs** list.
 - Tap the **Key Pressed** list and define how the signal should behave when the key is pressed.
 - Tap the **Allow in auto** list to select if the function is also allowed in automatic operating mode.

Following are the options available in the **Key Pressed** list.

- **Toggle**- switches the signal value from 0 to 1 or vice versa.
- **Set to 1** - sets the signal value to 1.
- **Set to 0** - sets the signal value to 0.

Continues on next page

- **Press/Release** - sets the signal value to 1 while the key is pressed (note that an inverted signal will be set to 0).
- **Pulse** - the signal value pulses once.

9 If the type **System** is selected:

- If **Toggle Increment Mode** is selected in the key **Pressed** list:
 - The programmable keys are defined to control toggling the increment mode on or off.
- If **Move PP to Main** is selected in the **Key Pressed List**:
 - The programmable keys are defined to move program point to main routine.
- Tap the **Allow in auto** list to select if the function is also allowed in automatic operating mode.

10 Tap **Apply**.

The selected key is configured.

11 Configure the other keys, if needed.

2 Navigating and handling the FlexPendant

2.3.12 Adapting the FlexPendant for left-handed users

2.3.12 Adapting the FlexPendant for left-handed users

Overview

Left-handed users, normally prefers their left-hand for the touch screen. The FlexPendant can easily be adapted to suit the needs of the left-handed users. They can easily rotate the screen through 180 degrees and use their right-hand to support the device.

Rotating the FlexPendant screen

Use the following procedure to adapt the FlexPendant to suit a left-handed user.

- 1 On the FlexPendant, tap QuickSet and select the **Logout/Restart** tab.
- 2 In the **Screen orientation** section, tap the **Rotate 180°** button.

The FlexPendant screen is rotated by 180 degrees and is adapted for the left-handed users.

What is affected?

The following settings are affected when adapting the FlexPendant for a left-handed user.

| Setting | Effect | Information |
|--|---|---|
| Jogging directions | The joystick directions are adjusted automatically. | The illustrations of jogging directions in the jogging menu are adjusted automatically. |
| Hardware buttons and programmable keys | No change. | See Hard buttons on page 22 . |
| Emergency stop | No change. | |
| Three-position enabling device | No change. | |

2.4 Basic procedures

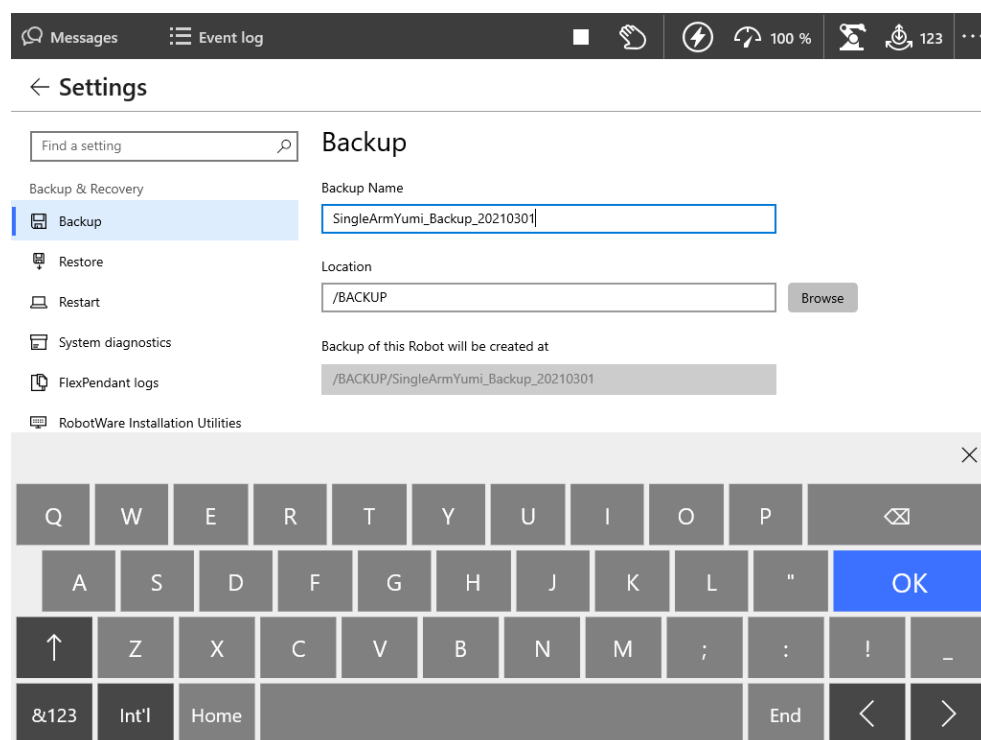
2.4.1 Using the soft keyboard

Soft keyboard

The soft keyboard is used frequently when operating the system, for example when entering file names or parameter values.

The soft keyboard works as an ordinary keyboard with which you can place the insertion point, type and correct typing errors. Tap letters, numbers and special characters to enter your text or values.

The following illustration shows the soft keyboard on the FlexPendant.




xx2000000922

Using international characters

To access international characters, tap the Int'l key on the soft keyboard. All western characters can be used, also in usernames and passwords.

Changing the insertion point

Tap the arrow keys to change the insertion point, for instance when correcting typing errors.


| If you need to move... | then tap... |
|------------------------|---|
| backward |  <p>xx2000000923</p> |

Continues on next page

2 Navigating and handling the FlexPendant

2.4.1 Using the soft keyboard

Continued

| If you need to move... | then tap... |
|------------------------|---|
| forward |  xx2000000924 |

Deleting

- 1 Tap the **Backspace** key (top right) to delete characters to the left of the insertion point.



xx2000000925

2.4.2 Messages on the FlexPendant

Overview

The FlexPendant displays messages from the system. These can be status messages, error messages, program messages, or requests for action from the user. Some require actions, and some are plain information.

Event log messages

The event log messages are messages from the RobotWare system about system status, events, or errors.

How to work with the event log messages is described in section [Handling the event log on page 263](#). All messages are also described in *Technical reference manual - Event logs for RobotWare 7*.

System messages

Some messages sent out by the system are not from the event log. They can come from other applications, such as RobotStudio.

To be able to change configurations and settings in the system from RobotStudio, the user must request write access. This generates a message on the FlexPendant where the operator can grant or deny access. The operator can at any time decide to withdraw the write access.

How to request access and work with RobotStudio is described in *Operating manual - RobotStudio*.

Program messages

RAPID programs can send out messages to the Operator window, for example, while running a service routine. See, [Overview on page 34](#).

How to generate program messages is described in *Technical reference manual - RAPID Instructions, Functions and Data types*.

2 Navigating and handling the FlexPendant

2.4.3 Capturing screenshots

2.4.3 Capturing screenshots

Overview

It is possible to capture the screenshot of a FlexPendant screen during operation. This function is useful when you want to explain about a FlexPendant screen, to report a FlexPendant issue, and so on.

Procedure

Use the following procedure to capture a FlexPendant screenshot:

- 1 Decide on the screen that you want to capture.
- 2 Press and hold the **Messages** hard button for a longer duration (around 1 second).

A toast notification is displayed and the screenshot of the current screen is saved to the temp folder on the controller disk.



Note

Use the File Transfer function in RobotStudio to access the image. The name of the file is in the format `FlexPendantScreen_yyyy-MM-dd_HH-MM-ss-ms.png`.



Note

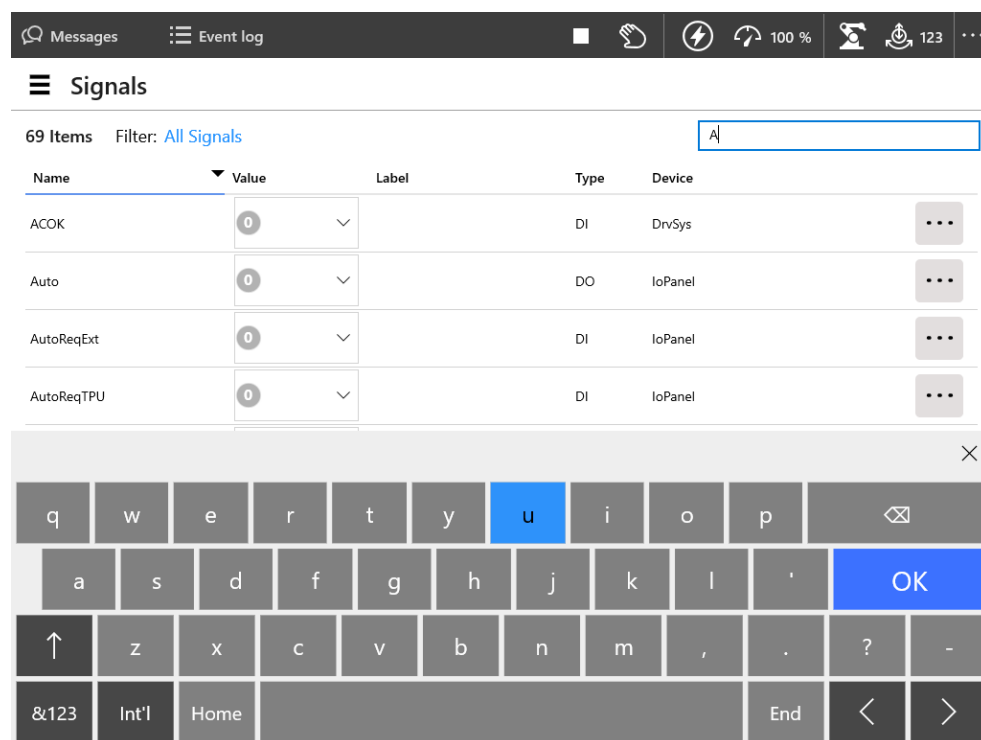
For information regarding the FlexPendant hard buttons, see FlexPendant [Main parts on page 20](#).

2.4.4 Filtering data

Filtering data

In several of the FlexPendant menus you can use filtering. This can be useful when you are looking at instances of a data type when there are many instances available. By filtering instances starting with a specific character for example, the number of items can be greatly reduced.

Depending on the type of data, you can filter data either alphabetically or numerically.



xx2000000926



Note

When filtering I/O signals there are more options than other types of data. For example, you can filter data by **Name** or **Type**

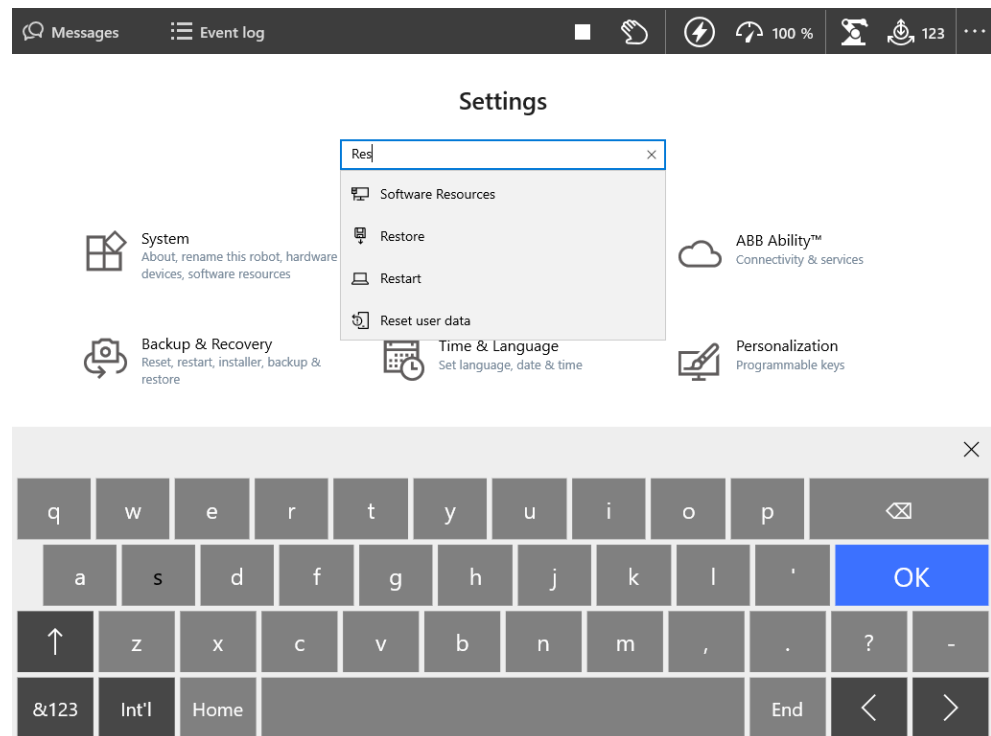
2 Navigating and handling the FlexPendant

2.4.5 Searching the settings

2.4.5 Searching the settings

Searching the settings

From the Settings application main page you can easily navigate to a particular settings by making use of the instant search feature. As you type few letters of a keyword in the settings search bar, the instant search feature starts listing the results in the search bar as shown in the following figure. Select the desired settings result to navigate to the corresponding settings window.



2.4.6 Granting access for RobotStudio

About write access on the controller

The controller accepts *only* one user with write access at a time. Users in RobotStudio can request write access to the system. If the system is running in manual mode, the request can be accepted or rejected on the FlexPendant.

Granting access for a RobotStudio user

The following procedure describes how to grant access for a RobotStudio user.

- 1 When a user in RobotStudio requests access, a message is displayed on the FlexPendant. Decide whether to grant or reject the access.
- 2 If you want to grant access, tap **Grant**. The user holds write access until he disconnects or until you reject the access.

If you want to reject access, tap **Deny**.



Note

If you have granted the access and want to revoke it, tap **Deny**.

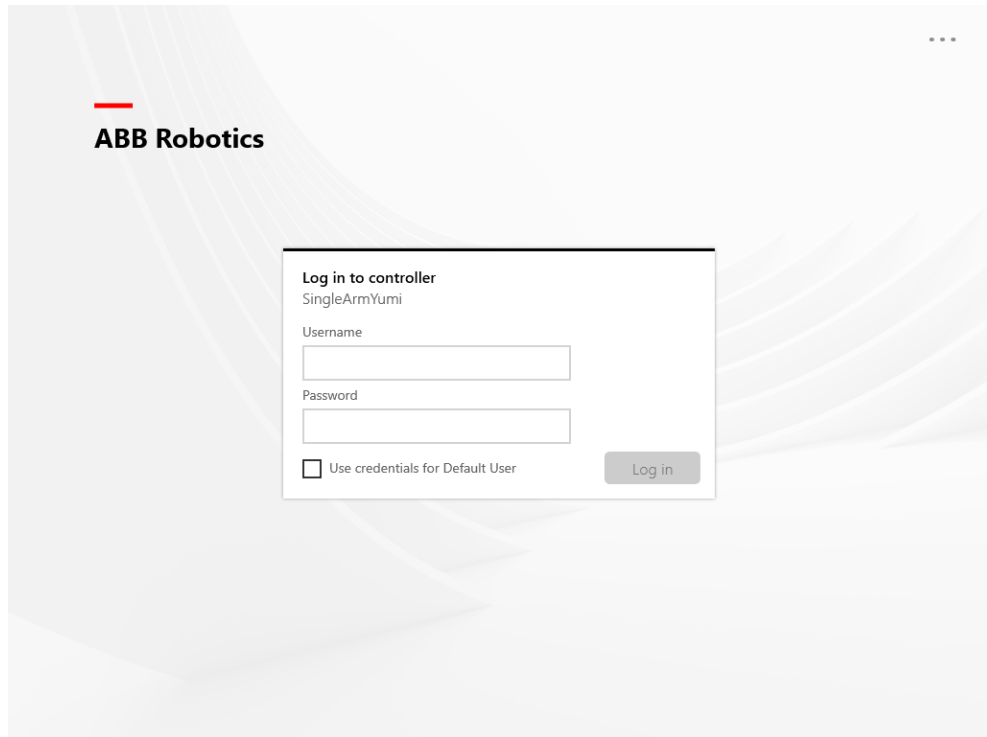
2 Navigating and handling the FlexPendant

2.4.7 Logging on and off

2.4.7 Logging on and off

Login procedure

Use the following procedure to log on to the controller, using the User Authorization System (UAS). For more details about User Authorization System, see *Operating manual - RobotStudio*.



xx2000000931

- 1 Tap on the **Username** field.
The soft keyboard is displayed.



Note

If you select the **Use credentials for Default User** checkbox then no password is required, and you are logged on automatically.

- 2 Type a valid username using the soft keyboard.
- 3 Tap on the **Password** field and type password for the selected user using the soft keyboard.
- 4 Tap **Log in**.
The selected user is logged in.

Continues on next page



Note

If you try to login with wrong password for more than 5 times in a row, the user account will be locked for two minutes. During that time all login attempts will be rejected. After two minutes you will get one new attempt for login and if that also fails another two minutes lockout is initiated. In case you forgot the password, contact system administrator for resetting the password.



Note

After a log off, the **Log in to controller** window is displayed.

Logout procedure

You can logout from the controller using one of the following procedures:

Using the QuickSet menu

- 1 Tap on the **QuickSet** button and select the **Logout/Restart** tab.
The Logout/Restart page is displayed.
- 2 In the **Current User** section tap **Logout**.
The current active user is logged off from the controller and the **Log in to controller** window is displayed.

Using the Settings application

- 1 On the start screen, tap **Settings**.
The Settings page is displayed.
- 2 At the bottom of the Settings page tap **Logout**.
The current active user is logged off from the controller and the **Log in to controller** window is displayed.

Handling users and authorization levels

Read more on how to add users or set the authorization in *Operating manual - RobotStudio*.

2 Navigating and handling the FlexPendant

2.4.8 Changing the user password

2.4.8 Changing the user password

Overview

It is possible to change the password of the logged in user from the FlexPendant.

Changing password procedure

Use the following procedure to change password of the logged in user.

- 1 On the start screen, tap **Settings**.
- 2 Tap **System**.
- 3 On the left sidebar, tap **Change User Password**.
The **Change User Password** window is displayed.
- 4 In the **Old Password** field type the current password.
- 5 In the **New Password** field type a new password.



Note

The number of characters in the password should be between 6 and 16.

- 6 In the **Confirm New Password** field type the same password that you typed in the **New Password** field.
- 7 Tap **Change Password**.
The new password is validated and saved.

2.4.9 Calibrating the touchscreen

Introduction

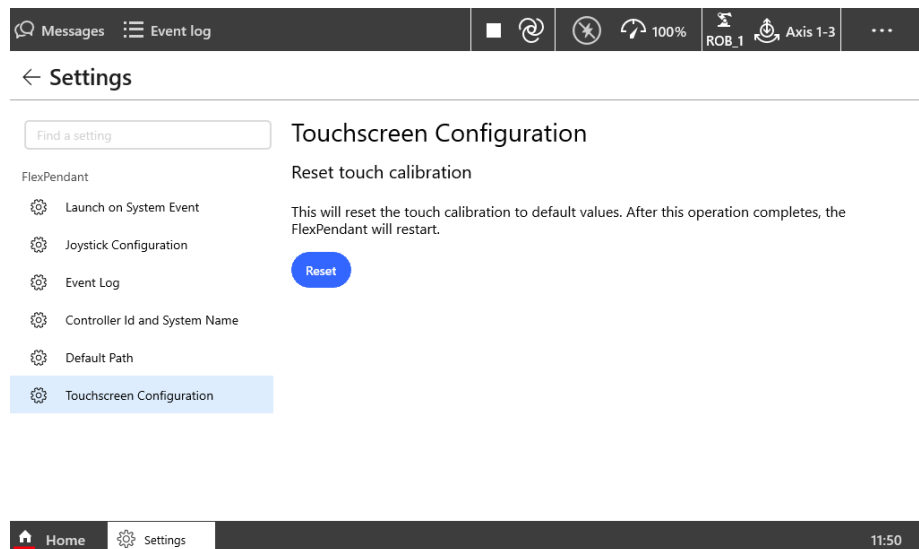
The FlexPendant touchscreen is factory calibrated and normally there is no need of recalibration. But in case such a requirement arises it is possible to recalibrate the FlexPendant touchscreen.

Calibrating the touchscreen

Use the following procedure to calibrate the touchscreen:

- 1 On the start screen, tap **Settings**.
- 2 Tap **FlexPendant**.
- 3 On the left sidebar, tap **Touchscreen Configuration**.

The **Touchscreen Configuration** page is displayed.



xx2400000291

- 4 Tap **Reset**.
The **Reset touch calibration** confirmation window is displayed.
- 5 Tap **OK**.
The touchscreen calibration values are reset to the default values and the FlexPendant is automatically restarted.

2 Navigating and handling the FlexPendant

2.5 Updating the applications

2.5 Updating the applications

Overview

The FlexPendant applications are updated using the **Update** option. For more details, see [Updating the FlexPendant applications on page 282](#).

3 OmniCore controller operating modes

3.1 Introduction

Overview

The OmniCore controller is delivered with the *Keyless Mode Selector* option. Using this you can change the operating modes from FlexPendant.

The following operating modes are available:

- Manual mode, also known as Manual reduced speed mode.
- Manual high speed mode (not available in USA or Canada).
- Automatic mode.

The operating modes are described in the product manual for the robot controller.



Note

This behavior of OmniCore controller is different compared to IRC5.



CAUTION

Suspended safeguards shall be returned to full functionality prior to selecting automatic mode.

Safety aspects of the Keyless Mode Selector



DANGER

Since the mode change and motors on operations are handled from the FlexPendant, it is physically possible to perform these operations within the safeguarded space. The user must always make sure to have safety equipment that is blocking automatic operation while being inside the safeguarded space, for example an auto stop connected to a gate.



Note

The Keyless Mode Selector is only a part of the robot. It is the responsibility of the integrator to do a risk assessment of the robot system.

3 OmniCore controller operating modes

3.2 Changing operating modes

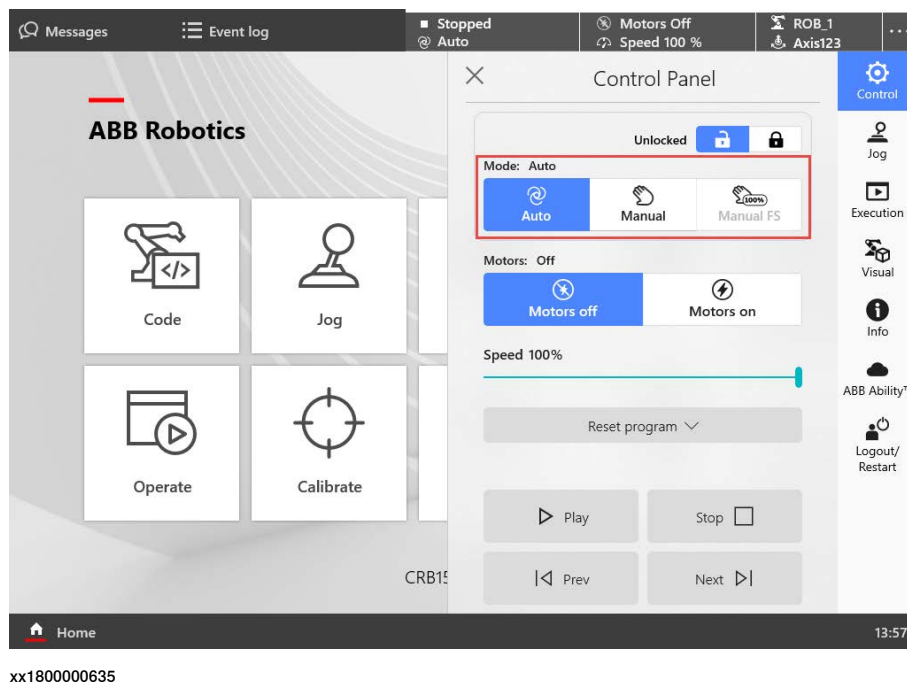
3.2 Changing operating modes

Procedure

Use the following procedure to change the operating mode on the robot controller:

- 1 On the FlexPendant status bar, tap the **QuickSet** button.

The **Control Panel** window is displayed.



- 2 Navigate to the **Mode** section. If the operating mode is locked, unlock it by following the procedure [Unlocking the operating mode on page 81](#).

- 3 Select the required operating mode.

When changing to automatic operating mode, a dialog is displayed with an acknowledge button. This does not apply to YuMi robots.

The operating mode is changed.



Note

OmniCore goes to motors OFF state when the operating mode is changed. This does not apply to YuMi robots.

For YuMi robots the motors are automatically ON after changing to auto mode.



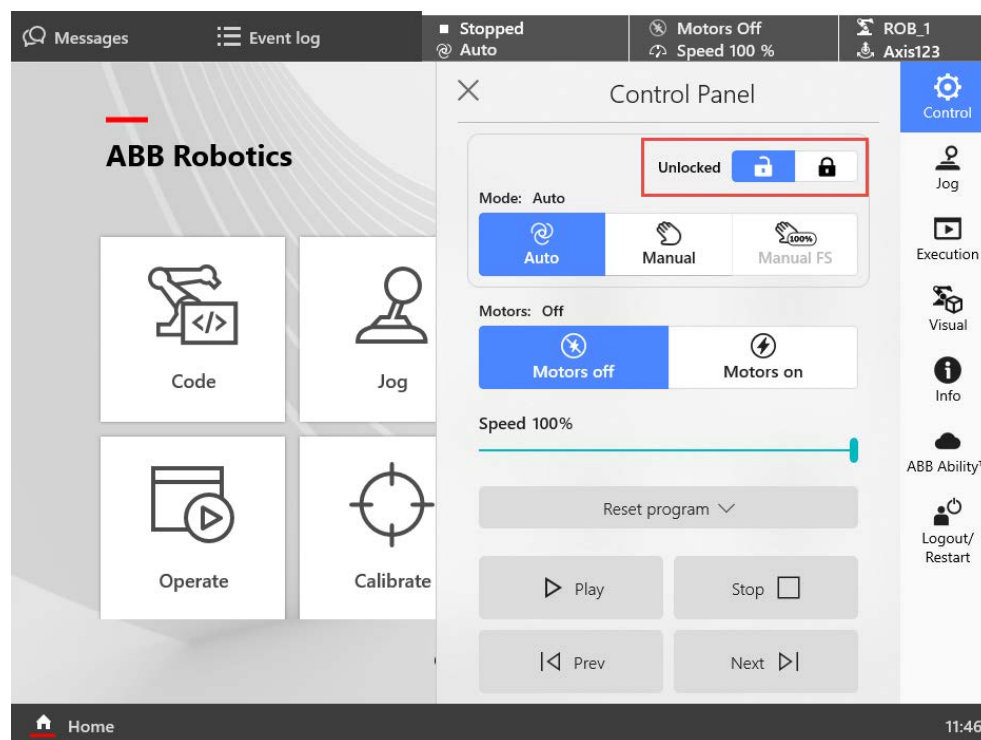
Note

The operating mode change works only in the sequence **Auto** <> **Manual** (Manual Reduced Speed) <> **Manual FS** (Manual Full speed). It is not possible to change the operating mode directly from **Auto** to **Manual FS**.

3.3 Locking and unlocking operating modes

Introduction

It is possible to restrict an unauthorized change in operating mode by locking it. This is done by configuring a temporary or permanent PIN-code.



xx2100000749

If a permanent PIN-code is not configured, then all users can configure a temporary PIN-code.

The permanent PIN-code can only be configured by a user with the grant **Lockable Mode Selector**.

Configuring a temporary PIN-code

Use the following procedure to configure a temporary PIN-code:

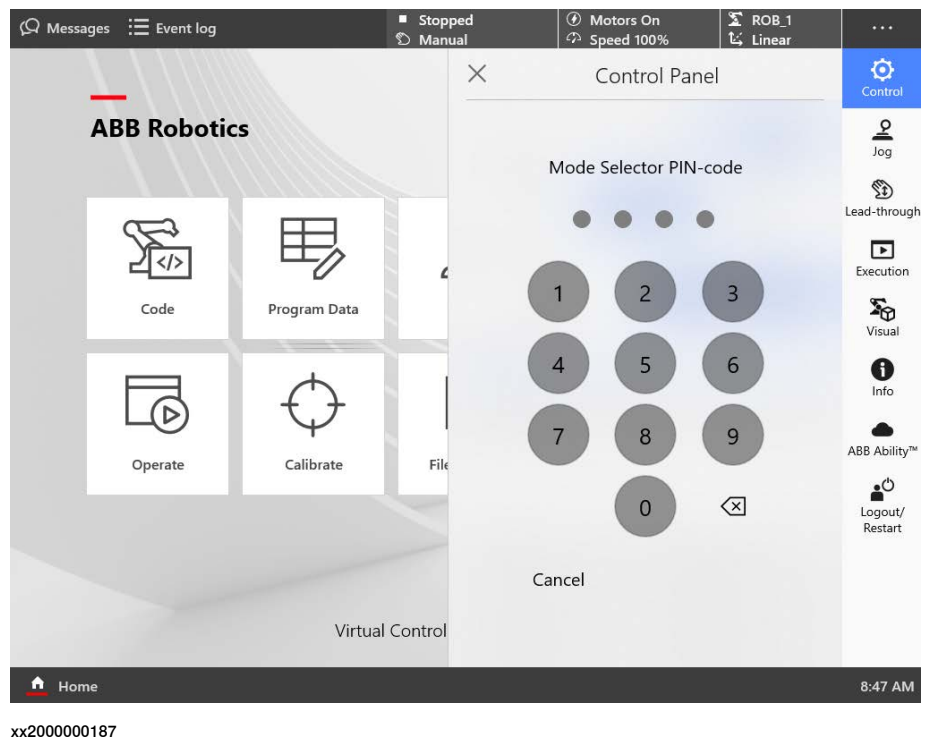
- 1 On the FlexPendant status bar, tap the **QuickSet** button.
The **Control Panel** window is displayed.
- 2 Tap the **Lock** button.
The **Mode Selector PIN-code** window is displayed.

Continues on next page

3 OmniCore controller operating modes

3.3 Locking and unlocking operating modes

Continued



3 Tap a 4 digit PIN-code on the number pad.

4 Tap OK.

The operating mode is locked.

Once a temporary PIN-code is enabled, the user must provide the correct PIN-code to change the operating mode. For more details, see [Unlocking the operating mode on page 81](#).



Note

Since the temporary PIN-code is reset every time the operator unlocks it, this procedure must be repeated every time to set a temporary PIN-code.

Configuring a permanent PIN-code

A user with the **Lockable Mode Selector** grant can configure a permanent PIN-code for other users to lock the operating mode.



Note

A user with the **Lockable Mode Selector** grant can unlock the Lock button at any time without providing a PIN-code.

Use the following procedure to configure a permanent PIN-code:

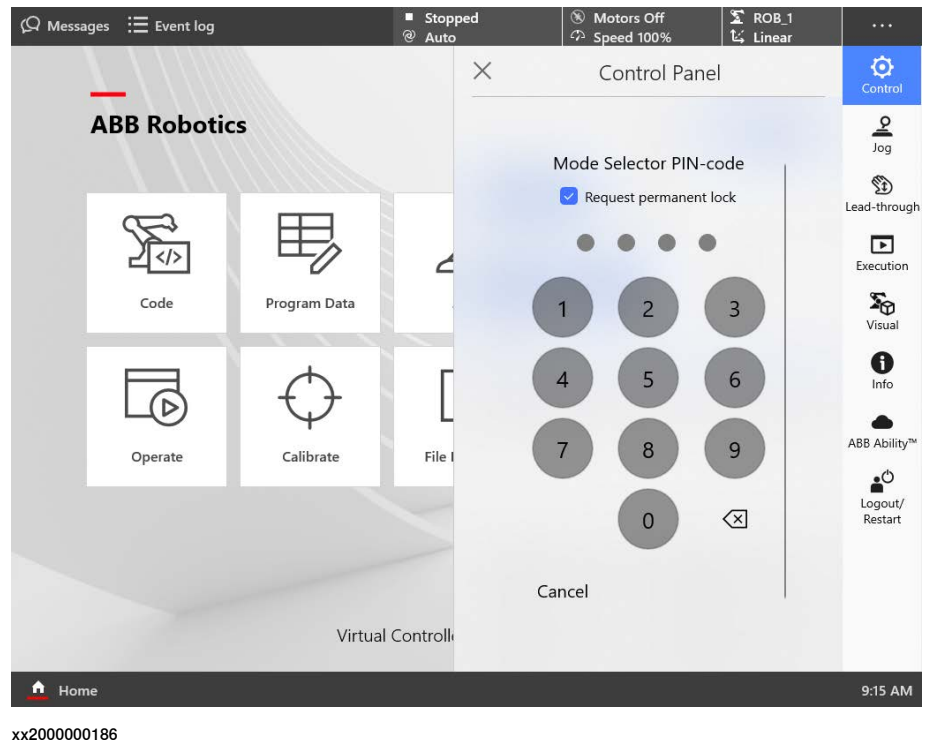
1 On the FlexPendant status bar, tap the **QuickSet** button.

The **Control Panel** window is displayed.

2 Tap the Lock button.

The **Mode Selector PIN-code** window is displayed.

Continues on next page



3 Select the **Request permanent lock** check box.

The **Request Permanent lock** check box is displayed only for those users with the **Lockable Mode Selector** grant enabled.

4 Tap a 4 digit PIN-code on the number pad.

5 Tap OK.

The permanent PIN-code is saved and is applicable for all the users.

Once a permanent PIN-code is enabled, the users must provide the correct PIN-code to change the operating mode. For more details, see [Unlocking the operating mode on page 81](#).



Note

For permanent PIN-code, after an unlock, if the operating mode is changed, the mode change will again be locked immediately.



Note

A saved permanent PIN-code is valid until a user with the **Lockable Mode Selector** grant resets the permanent PIN-code.

Unlocking the operating mode

If the operating mode is locked you can unlock it using a temporary or a permanent PIN-code.

Use the following procedure to unlock:

- 1 On the FlexPendant status bar, tap the **QuickSet** button.

Continues on next page

3 OmniCore controller operating modes

3.3 Locking and unlocking operating modes

Continued

The **Control Panel** window is displayed.

- 2 Tap the Unlock button.

The **Mode Selector PIN-code** window is displayed.

- 3 Tap the 4 digit PIN-code on the number pad.

For permanent PIN-code, the users will have one minute time to change the operating mode before it is locked again.

- 4 Tap **OK**.

The operating mode is unlocked.



Note

The temporary PIN-code is reset every time the user unlocks it.

4 Calibration

4.1 Introduction

Overview

This section provides an overview of the calibration information and about updating the revolution counter value for each axis, using the FlexPendant. Detailed information about calibration, revolution counter update, and so on can be found in the respective robot product manual.

4 Calibration

4.2 How to check if the robot needs calibration

4.2 How to check if the robot needs calibration

Check robot calibration status

Use the following procedure to check the calibration status of the robot:

- 1 On the start screen, tap **Calibrate**, and then **Calibration**.
- 2 The **Mechanical Unit** page is displayed.



Note

This step is required only if you are not already in the **Mechanical Unit** page when you open **Calibrate**.





Note


The **Mechanical Unit** page is displayed only if there are more than one mechanical unit available. Otherwise, the calibration summary page for the available mechanical unit is displayed.

- 3 Select the unit that needs to be calibrated from the **Mechanical Unit** list.
The calibration summary page for the selected mechanical unit is displayed.
The **Calibration Status** column displays the status of calibration for each axis.

What kind of calibration is needed?

| If the calibration status is... | then... |
|---------------------------------|--|
| Calibrated | Calibration is not needed. |
| Not calibrated | <p>the robot must be fine calibrated by a qualified service technician. Performing a fine calibration is described in the product manual for the robot.</p> <p> DANGER</p> <p>Do not attempt to perform the fine calibration procedure without proper training and tools. Doing so may result in the incorrect positioning that may cause injuries and property damage. Always consult a qualified service technician.</p> |
| Not updated | <p>update the revolution counters or perform the calibration.</p> <p> Note</p> <p>For IRB 14050 when you select update the revolution counters, you are recommended to perform the calibration.</p> <p>Updating the revolution counters is described in the product manual for the robot.</p> |

Continues on next page

| If the calibration status is... | then... |
|---------------------------------|--|
| Not commutated | <p>the robot must be fine calibrated by a qualified service technician. Performing a fine calibration is described in the product manual for the robot.</p> <p> DANGER</p> <p>Do not attempt to perform the fine calibration procedure without the proper training and tools. Doing so may result in incorrect positioning that may cause injuries and property damage. Always consult a qualified service technician.</p> |

This page is intentionally left blank

5 Jogging

5.1 Introduction to jogging

What is jogging?

Jogging is the process of manually positioning or moving the manipulator or additional axes in manual mode.

Prerequisites

You can jog the robot under the following conditions:

- The system has been started as detailed in this manual.
- Define the working range for the robots as well as for any other pieces of equipment working in the robot cell. The robot's working range is defined by system parameters. For more details, see *Technical reference manual - System parameters*.
- No programmed operation is running
- The controller is in manual mode.
- The three-position enabling device is pressed and the system is in Motors ON state.



Note

YuMi robots with SafeMove requires using the enabling device.
On YuMi robots without SafeMove the enabling device is disabled, hence, not used.



CAUTION

Before jogging, ensure the entire safeguarded space is visible to confirm that no person is present.



CAUTION

Before jogging, make sure that there are no obstacles that the robot can run into.

Limitations

There might be restrictions to how you can jog, see section [Restrictions to jogging on page 100](#).

Mechanical units that are configured with the safety function *Safe Disable of Drive Unit* cannot be jogged when they are disabled. See *Application manual - Functional safety and SafeMove*.

Continues on next page

5 Jogging

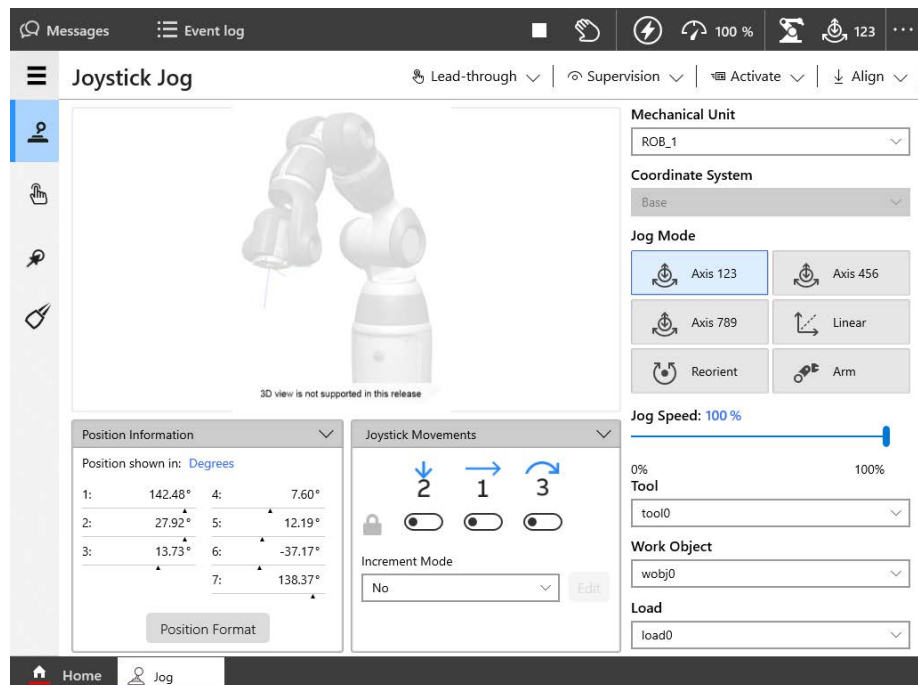
5.1 Introduction to jogging

Continued

Jogging the robot

Use the following procedure to jog the robot.

- 1 On the start screen, tap **Jog**, and then select **Joystick Jog** or **Touch Jog** from the menu.



xx2000001001

- 2 From the **Mechanical unit** list, select a mechanical unit.
- 3 From the **Motion mode** section, select an axis-set that need to be jogged. For example, to jog axis 2, select the axis set **Axis 1-3**.
- 4 Press and hold the three-position enabling device to enable the motors.
- 5 If you have selected **Joystick Jog**, refer the **Joystick Movements** section to understand the direction of the axis that you want to move and move the joystick.

If you have selected **Touch Jog**, tap on the axis that you want to move and drag it to the positive or negative direction.

The axis is moved according to the movement.

About motion modes and robots

The selected motion mode or coordinate system determines the way the robot moves.

In linear motion mode, the tool center point moves along straight lines in space, that is, it moves from "point A to point B". The tool center point moves in the direction of the selected coordinate system's axes.

Axis-by-axis motion mode moves one robot axis at a time. It is then hard to predict how the tool center point moves.

Continues on next page

About motion modes and additional axes

Additional axes can be jogged only axis-by-axis. An additional axis can either be designed for some kind of linear motion or for rotational (angular) motion. Linear motion is used in conveyers and rotational motion is used in many kinds of workpiece handlers.

Additional axes are not affected by the selected coordinate system.

5 Jogging

5.2 Coordinate systems for jogging

5.2 Coordinate systems for jogging

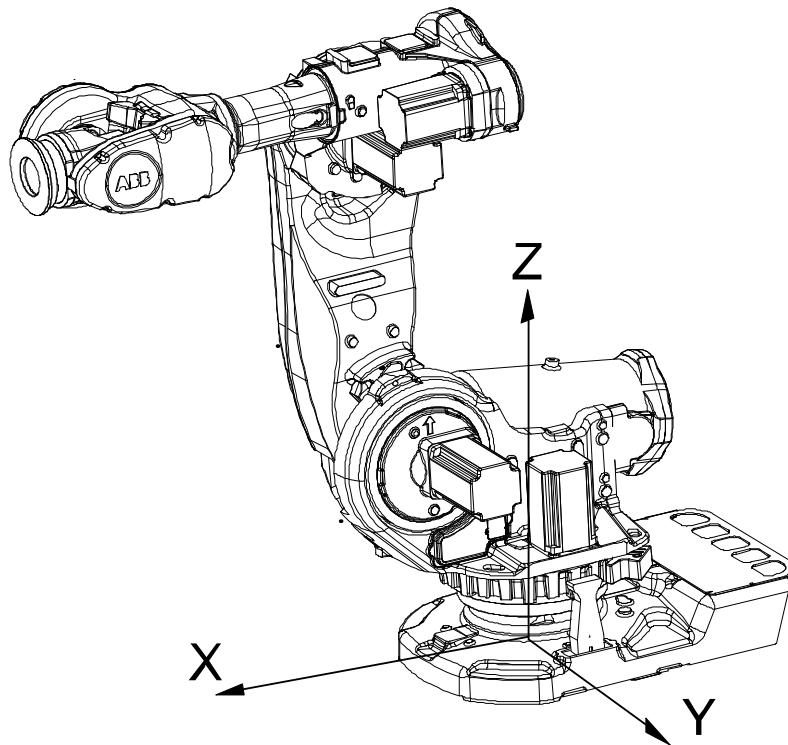
Coordinate systems

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.

- 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 **Wobj** (work object) coordinate system is related to the work piece and is often the best one for programming the robot.
- The **Tool** coordinate system defines the position of the tool the robot uses when reaching the programmed targets.
- The **World** coordinate system defines the robot cell. It is useful for jogging, general movements, and for handling stations and cells with several robots.

The base coordinate system



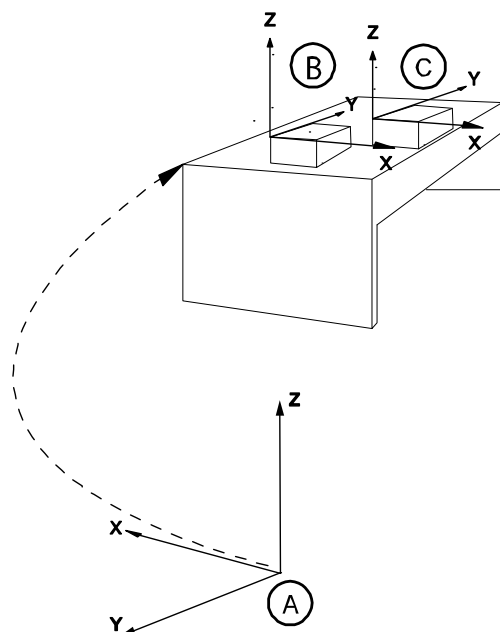
xx0300000495

The base coordinate system has its zero point at the base of the robot, which makes the movement predictable for fixed mounted robots. It is therefore useful for jogging a robot from one position to another. For programming a robot, other coordinate systems, like the work object coordinate system are often the better choices.

Continues on next page

When you are standing in front of a robot and jog in the base coordinate system, in a normally configured robot system, pulling the joystick towards you will move the robot along the X axis, while moving the joystick to the sides will move the robot along the Y axis. Twisting the joystick will move the robot along the Z axis.

The work object coordinate system



xx0600002738

| | |
|---|---------------------------------|
| A | World coordinate system |
| B | Work Object coordinate system 1 |
| C | Work Object coordinate system 2 |

The work object coordinate system corresponds to the work piece. It defines the placement of the work piece in relation to the world coordinate system (or any other coordinate system).

The work object coordinate system must be defined in two frames, the user frame (related to the world frame) and the object frame (related to the user frame).

A robot can have several work object coordinate systems, either for representing different work pieces or several copies of the same work piece at different locations.

It is in the work object coordinate system you create targets and paths when programming the robot. This gives the following advantages:

- When repositioning the work piece in the station you just change the position of the work object coordinate system and all paths are updated at once.
- Enables work on work pieces moved by external axes or conveyor tracks, since the entire work object with its paths can be moved.

Following are two examples of work object coordinate system in use:

- To determining the positions of a number of holes to be drilled along the edge of the work object.
- To create a weld between two walls in a box.

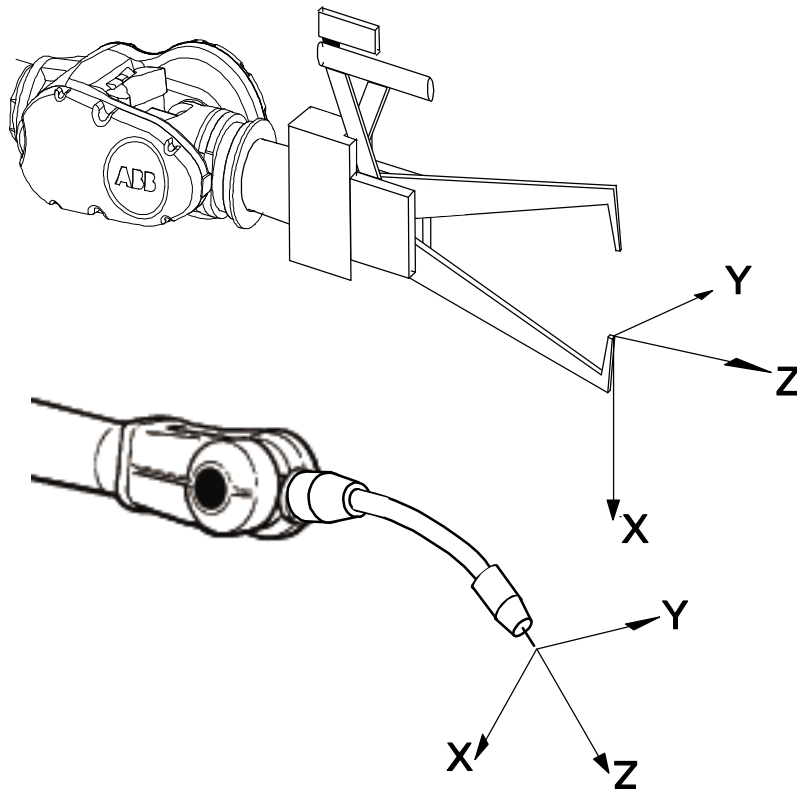
Continues on next page

5 Jogging

5.2 Coordinate systems for jogging

Continued

The tool coordinate system



en0300000497

The tool coordinate system has its zero position at the center point of the tool. It thereby defines the position and orientation of the tool. The tool coordinate system is often abbreviated TCPF (Tool Center Point Frame) and the center of the tool coordinate system is abbreviated TCP (Tool Center Point).

It is the TCP that moves to the programmed positions, when executing programs. This means that if you change the tool (and the tool coordinate system) the robot's movements will be changed so that the new TCP will reach the target.

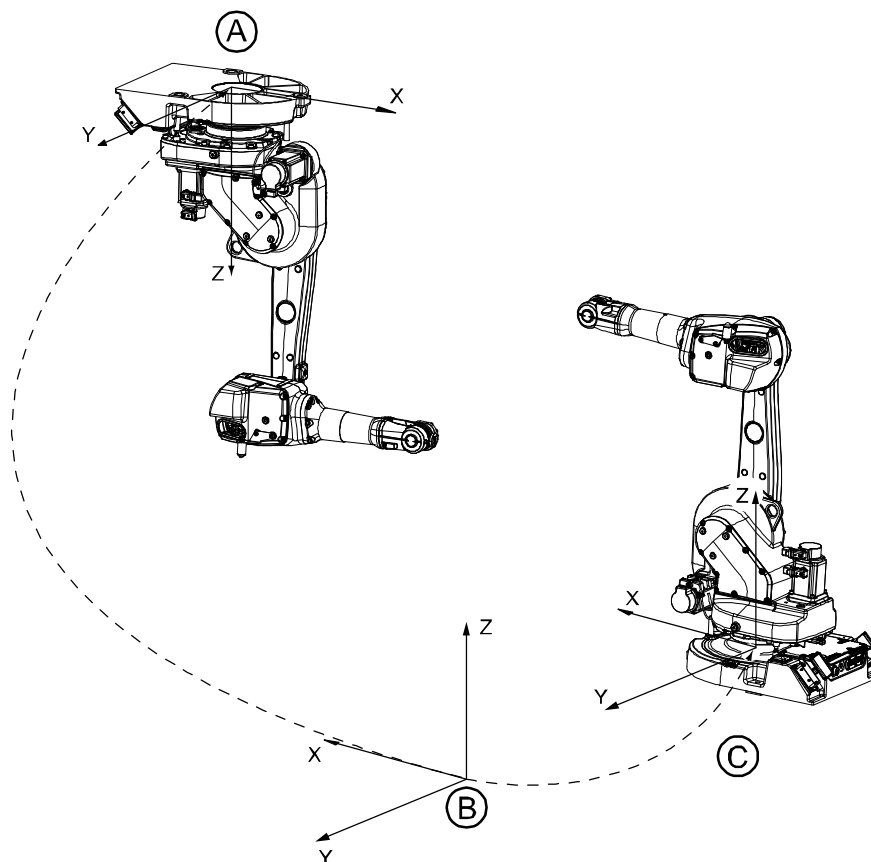
All robots have a predefined tool coordinate system, called `tool0`, located at the wrist of the robot. One or many new tool coordinate systems can then be defined as offsets from `tool0`.

When jogging a robot the tool coordinate system is useful when you don't want to change the orientation of the tool during the movement. For example, moving a saw blade without bending it.

Following is an example of tool coordinate system in use:

- Use the tool coordinate system when you need to program or adjust operations for threading, drilling, milling, or sawing.

Continues on next page

The world coordinate system

en0300000496

| | |
|---|------------------------------------|
| A | Base coordinate system for robot 1 |
| B | World coordinate |
| C | Base coordinate system for robot 2 |

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.

Examples of use

For example, you have two robots, one floor mounted and one inverted. The base coordinate system for the inverted robot would be upside down.

If you jog in the base coordinate system for the inverted robot, movements will be very difficult to predict. Choose the shared world coordinate system instead.

Illustration of axes and joystick directions

The axes of a generic six axis robot can be jogged manually using the joystick. Please check your plant or cell documentation to determine the physical orientation of any additional axes.

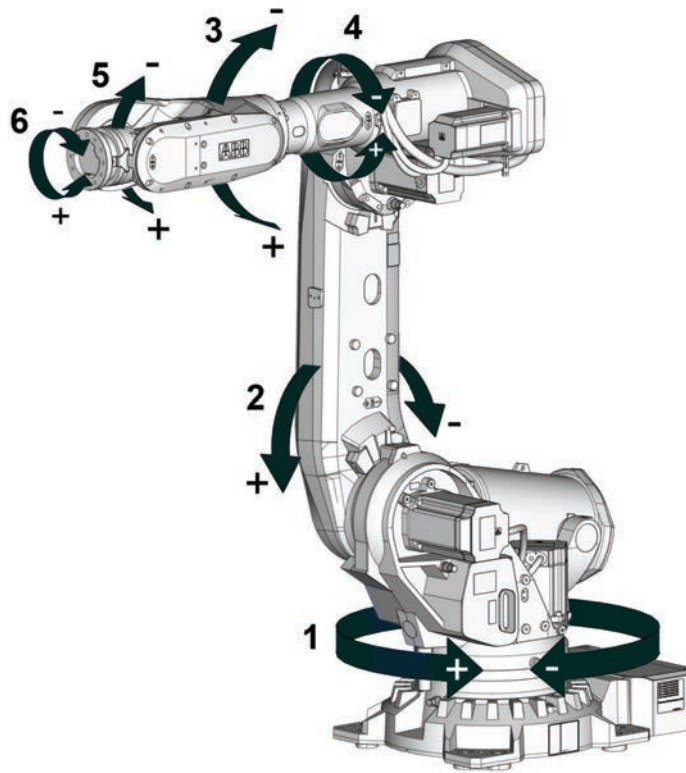
The illustration shows the movement patterns for each manipulator axis.

Continues on next page

5 Jogging

5.2 Coordinate systems for jogging

Continued



xx0300000520

5.3 Basic settings for jogging

Selecting mechanical unit for jogging

If your system has more than one robot, that is, additional robots or additional axes, then you need to select which mechanical unit to jog when using the joystick. Each mechanical unit that can be jogged is represented in the **Mechanical unit** list. The name of the unit is defined in the system configuration.



Note

The selected mechanical unit is active until you select another unit.

Mechanical units can be activated or deactivated with the **Activate** function in the Jogging window.

See your plant or cell documentation to see which mechanical units are available in your robot.



Note

Mechanical units that are configured with the safety function *Safe Disable of Drive Unit* cannot be jogged when they are disabled. See *Application manual - Functional safety and SafeMove*.

Selecting Motion mode

Navigate to the **Movement Types** section and tap on the movement type that you want to select.

Selecting tool, work object, and payload

It is important to choose the proper tool, work object, or payload. This is required when you create a program by jogging to the target positions. Failing to do this may result in overload errors or incorrect positioning either when you jog or when you run the program.

Select tool, workobject, or payload from the **Active tool**, **Active workobject**, or **Active Payload** list respectively according to the requirement.

Setting the tool orientation

Tools for arc welding, grinding, and dispensing must be oriented in a particular angle to the work piece to obtain the best result. You also need to set up the angle for drilling, milling or sawing.

In most cases you set the tool orientation when you have jogged the tool center point to a specific position such as the starting point for a tool operation. After you have set the tool orientation you continue to jog in linear motion to complete the path and the supposed operation.

The tool orientation is relative to the currently selected coordinate system. From a user perspective however this is not noticeable.

Select **Align Tool** from the menu.

Continues on next page

5 Jogging

5.3 Basic settings for jogging

Continued

Jogging axis by axis

Use axis by axis jogging when you need to:

- Move the mechanical unit out of a hazardous position.
- Move robot axes out of singularities.
- Position axes for fine calibration.

Use the following procedure to select axis group in the **Jogging** window.

- 1 On the start screen, tap **Jog**, and then select **Joystick Jog** or **Touch Jog** from the menu.
- 2 Navigate to the **Motion mode** section and select the axis set.
- 3 Press and hold the three-position enabling device to enable the motors.
- 4 Jog the axis according to your requirement.

The selected axis is moved according to your hand gestures.



CAUTION

The orientation of any mounted tool is affected by this procedure. If the resulting orientation is important, perform the procedure described in [Setting the tool orientation on page 95](#).

Incremental movement for precise positioning

Use incremental movement to jog the robot in small steps, which enables very precise positioning.

This means that once the increment setting is configured, the robot moves one step when you swipe on the graphical window. If you swipe for more time, a sequence of steps, (at a rate of 10 steps per second), is performed as long as you swipe the axis along the user interface.

You can use the jog app for precise movement.

Use the following procedure to select the incremental movement size using the **Jog** app.

- 1 On the start screen, tap **Jog**, and then select **Joystick Jog** from the menu.
- 2 Navigate to the **Motion mode** section and select the axis set.
- 3 In the **Joystick Movements** section, navigate to the **Increment Mode** list, and select the increment type.
- 4 Jog the selected axis.

The robot axis is moved according to the selected increment mode.

5.4 Reading the exact position

About positions and revolution counters

The exact position of the robot is determined using the position of the resolvers and counters that count the number of resolver revolutions. These are called revolution counters.

If the robot is correctly calibrated then the current position is automatically calculated at start.



CAUTION

If the positions are displayed in red text then the values from the revolution counters are lost and instead the values stored on the robot memory are displayed. Be careful when jogging the robot if the values are displayed in red text. Watch the robot closely and do not use the displayed values. If the mechanical unit is uncalibrated then the actual position can be very different from the position values stored by the robot memory. You must update the revolution counters before a program can be started. For more details about revolution counter update refer the respective robot product manual.



Note

If no positions are displayed then the mechanical unit is uncalibrated. Instead the text **Selected mechanical unit is not calibrated** is displayed.



Note

When updating the revolution counters, the ongoing RAPID instruction or function is interrupted, and the path is cleared.

How robot positions are displayed

Positions are always displayed as:

- The point in space expressed in the x, y, and z tool center point coordinates.
- The angular rotation of the tool center point expressed in Euler angles or as a quaternion.

How additional axes positions are displayed

When an additional axis is moved, only the axis position is displayed.

Linear axis positions are displayed in millimeters expressed as the distance to the calibration position.

Rotating axis positions are displayed in degrees expressed as the angle to the calibration position.

Reading the exact position

This procedure describes how to read the exact position.

- 1 On the start screen, tap **Jog**.

Continues on next page

5 Jogging

5.4 Reading the exact position

Continued

- 2 The position of each axis is displayed in the **Position Information** section.

Position format

The position can be displayed in different formats using the Jog application. To configure the position formats, in the **Position Information** section, tap **Position Format**.

The **Position Format** can be displayed relative to the following frames:

- World
- Base
- Workobject

Enable the **Display Orientation Values** option to configure the Orientation formats.

The **Orientation** format can be set to:

- Quaternion
- Euler angles

The **Presentation angle unit** can be set to:

- Degrees
- Radians

5.5 Incremental movement for precise positioning

Incremental movement

Use the increment mode to jog the robot in small steps, which enables very precise positioning.

This means that each time the joystick is deflected, the robot moves one step (increment). If the joystick is deflected for one or more seconds, a sequence of steps, (at a rate of 10 steps per second), will be performed as long as the joystick is deflected.

Default mode is no increment, then the robot moves continuously when the joystick is deflected.

Selecting the increment mode

To manage the incremental movement, open **Jog** and use the list available in the **Increment Mode** section.

Choose between **Small**, **Medium**, **Large** or **User** increments. Select the **User** option and tap **Edit** to define the values within in a certain range for each robot.

Incremental movement sizes

Choose between small, medium or large increments. You can also define your own increment movement sizes.

| Increment | Distance | Angular |
|-----------|----------|---------|
| Small | 0.05 mm | 0.005° |
| Medium | 1 mm | 0.02° |
| Large | 5 mm | 0.2° |
| User | | |

5 Jogging

5.6 Restrictions to jogging

5.6 Restrictions to jogging

Jog additional axes

Additional axes can be jogged only axis-by-axis. See *Application manual - Additional axes*.

Jog mechanical units that are not calibrated

If the mechanical unit is not calibrated the text **Unit not calibrated** will be displayed in the **Position** area of the **Jogging** window.

An uncalibrated mechanical unit can only be jogged axis-by-axis. Its working range will not be checked.

When the robot is not calibrated, incremental movement is restricted to one step per joystick deflection. A calibrated robot performs 10 steps/sec when deflecting the joystick.



CAUTION

Mechanical units whose working range is not controlled by the robot system can be moved to dangerous positions. Mechanical stops should be used and configured to avoid danger to equipment or personnel.

Jog robot axes in independent mode

It is not possible to jog axes in independent mode. You need to return the axes to normal mode in order to jog.

Jog while using world zones

With the option *World Zones* installed, defined zones will restrict the motion while you jog.

Jog with axis loads not set

If equipment is mounted on any of the robot axes, then axes loads must be set. Otherwise overload errors might occur when jogging.

How to set axis loads are described in the Product Manuals delivered with your robot.

Jog with tool or payload weights not set

If the weight of tools and payloads is not set, then overload errors might occur when jogging. Loads for additional axes controlled by specific software (dynamic models) can only be set in programming.

5.7 Lead-through

What is lead-through?

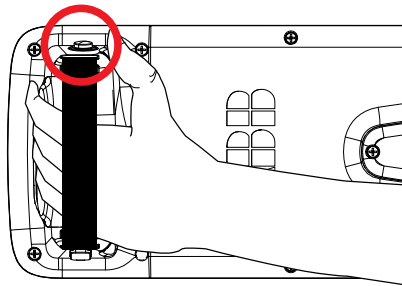
The lead-through functionality is available for robots designed for collaborative applications. If lead-through is available, this is shown on the FlexPendant.

Using lead-through, you can grab the robot arm and move it manually to a desired position, as an alternative to jogging.

Using lead-through

Use the following procedure to jog the robot using the lead-through functionality:

- 1 Enable lead-through in one of the following ways:
 - Press and hold the thumb button on the FlexPendant.



xx2100000331

- On the start screen, tap **Jog** and select the **Lead-through** menu.
- In the **QuickSet** menu, select the **Lead-through** tab.



Note

If the robot is in motors off state, it will automatically go to the motors on state when the lead-through is enabled.

- 2 In the **Jog Mode** section select a mode.
- 3 If required, in the **Lead-through lock** section use the lock button next to an axis to lock it.



Note

The **Lead-through lock** section is disabled for the **Axis 1-6** mode.

- 4 Gently pull the robot arm to the desired position.
The robot moves to the selected position. If the **Lead-through lock** option is selected, the robot moves in such a way that the movement is restricted in the locked direction.



Note

You can feel if an axis reaches its end position. Do not try to force the axis beyond this position.

Continues on next page

5 Jogging

5.7 Lead-through

Continued

- 5 If desired, save the position.



Note

The speed at which the robot moves when using the Lead-through functionality is managed using the horizontal scroll bar available in the **Lead-through Speed** section.



Note

If lead-through is enabled, it will be temporarily disabled during program execution and jogging. This means that it is possible to combine lead-through, jogging, and testing the RAPID program without having to disable the lead-through.



Note

When using lead-through, it is important that the load is correctly defined. If the load is heavier than defined, the effect will be the same as if you are pulling the robot arm downwards. If the load is lighter than the defined load, the effect will be the same as if you are pulling the robot arm upwards.

For the OmniCore, there is a button for updating/refreshing the load while lead-through is active.

For the OmniCore, if varying loads from cables and other disturbances are causing the robot to drift during lead-through, this can often be improved by setting the system parameter *Lead through load compensation* to *Always*. See *Technical reference manual - System parameters*, section *Motion*, type *Robot*.



Note

For Yumi robots with SafeMove, some different behaviors apply, see the product manual for the robot.

Align to a coordinate system

It is possible to align the robot to a coordinate system either in Auto or Manual mode from the lead-through page for the robot.

Use the following procedure to align the robot to a coordinate system:

- 1 In the Lead-through page select the a mode in the **Lead-through Mode** section.
- 2 In the **Align to coordinate system** section, select the required coordinate system.
- 3 Enable the motors.



Note

For collaborative robots, the motors are on by default unless extra safety options are selected in the system.

- 4 Tap and hold the **Press and Hold Align** button.

Continues on next page

The robot is aligned to the selected coordinate system.

Limitations

When using lead-through, the path planner is not updated until the program is resumed/restarted or jogging is used. For example, this means that World Zones supervision is not accessible when using lead-through.

5.8 Motion supervision

Overview

The controller software has the motion supervision functionality aiming at reducing collision impact forces on the robot. This helps protecting the robot and external equipment from severe damage if a collision occurs.

Motion supervision during program execution is always active, regardless which options are installed in the controller. When a collision is detected, the robot will immediately stop and relieve the residual forces by moving in reversed direction a short distance along its path. The program execution will stop with an error message. The robot remains in the Motors on state so that program execution can be resumed after the collision error message has been acknowledged.

Moreover, there is a software option called *Collision Detection*, which has extra features such as supervision during jogging. To find out if your system has this option installed, tap the **QuickSet** icon on the status bar, tap the **Info** tab, and look for the option **3107-1 Collision Detection**. For more information on *Collision Detection*, see *Application manual - Controller software OmniCore*.

Functions in RobotWare base

- *Path Supervision* is used (in automatic and manual full speed mode) to prevent the mechanical damage due to the robot running into an obstacle during program execution.
- *Non motion execution* is used to run a program without robot motion.

Functions in Collision Detection

A RobotWare system with *Collision Detection* option has the following additional functionalities:

- *Path Supervision* in manual mode and the possibility to tune supervision in all modes.
- *Jog Supervision* is used to prevent mechanical damage to the robot during jogging.
- RAPID instruction `MotionSup` is used to activate or deactivate collision detection and to tune sensitivity during program execution.

Configuring motion supervision



Note

The motion supervision configuration is allowed only in manual operating mode.

Use the following procedure to configure the motion supervision settings:

- 1 Open **Settings > Advanced**.
- 2 From the left panel select **Path supervision** or **Jog supervision**.
- 3 From the **Task Name** list select the task.
- 4 In the **Enable for selected task** option slide the bar and enable it.
- 5 In the **Sensitivity** section slide on the bar and select sensitivity.

Continues on next page

The selected sensitivity value is displayed near the section name.
The selected settings are automatically applied.

**Note**

All motion supervision must be set for each task separately.

Configuring Jog supervision using Jog application

Use the following procedure to configure Jog supervision from Jog application.

- 1 Open **Jog**.
- 2 Tap on **Supervision**.
The Jog supervision window is popped down.
- 3 From the **Task Name** list select the task.
- 4 In the **Enable for selected task** option slide the bar and enable it.
- 5 In the **Sensitivity** section slide on the bar and select sensitivity.
The selected sensitivity value is displayed near the section name.
The selected settings are automatically applied.

Non motion execution

Non motion execution enables you to run a RAPID program without robot motion. All other functions work normally; current cycle times, I/O, TCP speed calculation and so on.

Non motion execution can be used for program debugging or cycle time evaluation. It also represents a solution if you need to measure for example glue or paint consumption during a cycle.

When non motion execution is activated it can be executed in:

- manual mode
- manual full speed mode
- auto mode

Cycle times will be simulated according to the selected mode.

**Note**

Non motion execution can only be activated when the system is in Motors Off state.

**CAUTION**

Non motion execution is reset after a reboot. If you intend to run the program in non motion mode, do not restart without checking the status of **Non motion execution**. Starting the program incorrectly may cause serious injury or death, or damage the robot or other equipment.

5.9 Align tool

Overview

A tool can be aligned with a selected coordinate system.

When aligning a tool, the tool's z-axis is aligned to the nearest axis of the selected coordinate system. Therefore, it is recommended to first jog the tool so that it is close to the desired coordinates. When aligning a tool the tool's data is not changed.

Procedure

Use the following procedure to align a tool:

- 1 On the start screen, tap **Jog**.
- 2 On the command bar, tap **Align**.
The Align window is displayed. The current active tool and work object are displayed at the top of the window.
- 3 In the **Align to coordinate system** section, select the required coordinate system.
- 4 Press and hold the three-position enabling device to activate the motors.



Note

For collaborative robots, the motors are on by default unless extra safety options are selected in the system.

- 5 Continue holding the three-position enabling device and tap and hold the **Press and Hold Align** button.

The tool is aligned to the selected coordinate system.

5.10 Working with SmartGripper

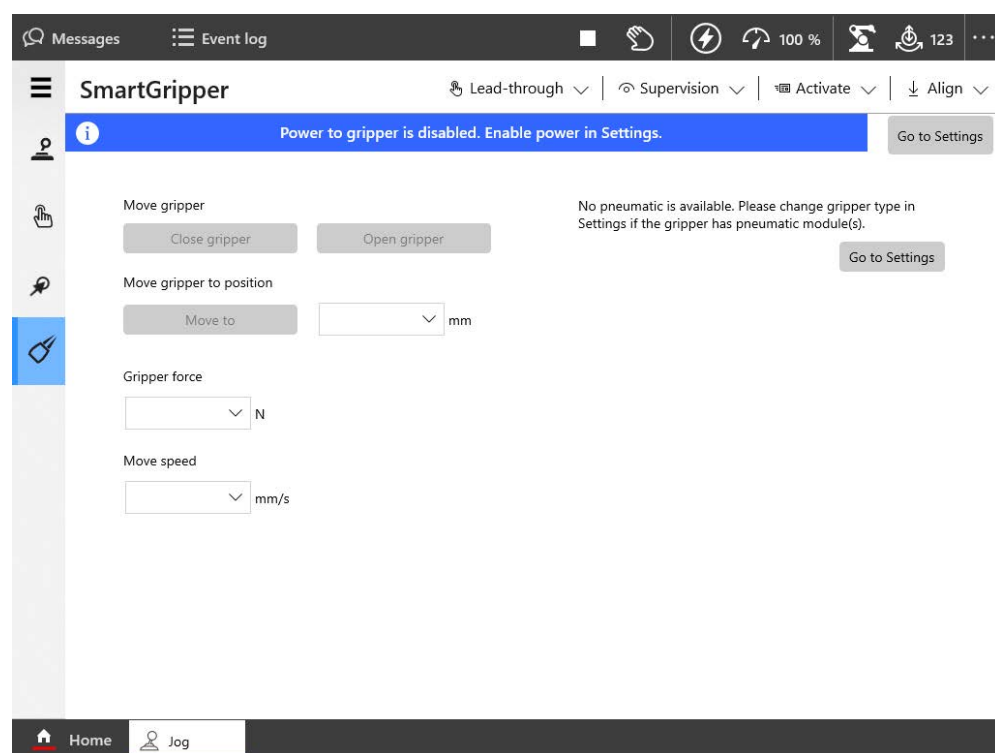
5.10.1 Introduction

Overview

SmartGripper is a RobotWare option that is used for facilitating the operation and programming of the ABB smart gripper. When you select the **SmartGripper Support** option in the **System Options** while creating a system using the **Modify Installation** function, the **SmartGripper** tab is displayed in the **Jog** application. Also, the settings for configuring the gripper is displayed in the **Settings** application. For more details about robot systems and the **Modify Installation** function, see *Operating manual - RobotStudio*.

SmartGripper - Jog user interface

The following figure and table describes the options available in the **SmartGripper** page when a gripper with the type vacuum is selected:



xx2000001338

| Group | Parameter | Description |
|--------------------------|--------------------------|--|
| Move gripper | Close gripper | Closes the gripper with the defined force and speed. |
| | Open gripper | Opens the gripper with the defined force and speed. |
| Move gripper to position | Move gripper to position | Sets a precise position value to move the gripper. |
| Gripper force | Force | Sets the gripping force for the gripper fingers. |

Continues on next page

5 Jogging

5.10.1 Introduction

Continued

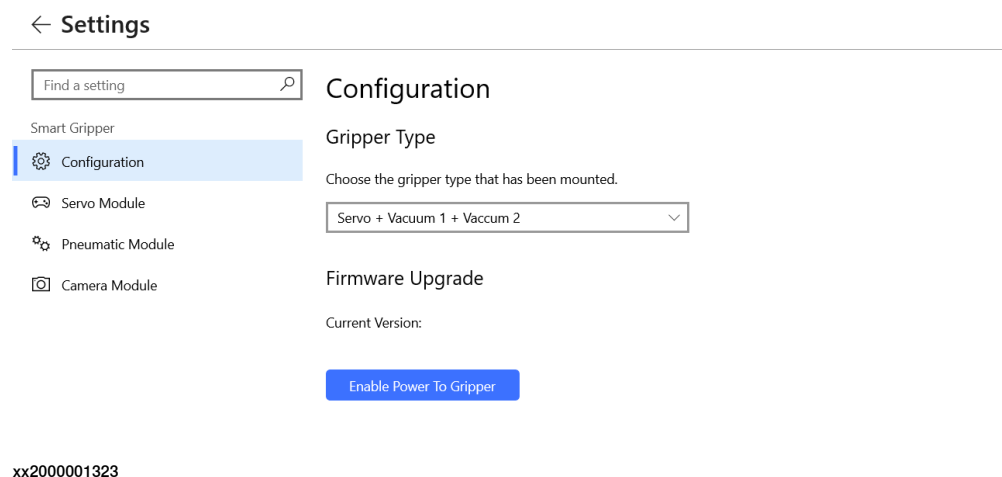
| Group | Parameter | Description |
|--------------|--------------|--|
| Move speed | Speed | Sets the movement speed for the gripper fingers. |
| Gas pressure | Gas pressure | Displays the current pressure of the gas. |
| Suction | On/Off | Sets the suction on or off. |
| Blow off | On/Off | Sets the blow-off on or off. |

SmartGripper - Settings user interface

Configuration page

When you open the SmartGripper settings, by default the **Configuration** page is displayed.

The following figure and table describes the options available in the **Configuration** page of **SmartGripper** settings.



| Group | Description |
|---------------------------------|---|
| Gripper Type | Allows you select the type of gripper. |
| Firmware upgrade | Displays the current version details of the gripper firmware. |
| Enable/Disable Power to Gripper | Enables/Disables the power to the gripper. |

Continues on next page

Servo module page

The following figure and table describes the options available in the **Servo module** page of the **SmartGripper** settings.

← Settings

Smart Gripper

Configuration

Servo Module

Pneumatic Module

Camera Module

Setup

Setup the gripper.

Calibrate

Speed (mm/s) 20

Force (N) 12

Command

Test the gripper using the following commands.

Jog + Jog - Stop

Grip + Grip - Move To 0 mm

Status

Calibration: **Non-calibrated**

Current Position: 0 mm

Current State: 0 - Ready

xx2000001328

| Group | Parameter | Description |
|---------|------------------------------|---|
| Setup | Calibrate | Calibrates the gripper at the current position. |
| | Speed | Sets the movement speed of the gripper fingers. |
| | Force | Sets the gripping force of the gripper fingers. |
| Command | Jog/Stop/Grip+/Grip-/Move to | <p>Allows you to manage the gripper finger movement.</p> <p>Note</p> <p>If the gripper is not calibrated, only the functions Jog and Stop can be used, and the functions Grip+, Grip- and Move To are disabled.</p> |
| Status | Calibration | Displays the calibration status of the gripper. |
| | Current position | Displays the current position of the gripper. |
| | Current state | Indicates the state of the gripper. For details about the gripper states, see <i>Returned value</i> in <i>IRB 14050 gripper Product manual</i> . |

Continues on next page

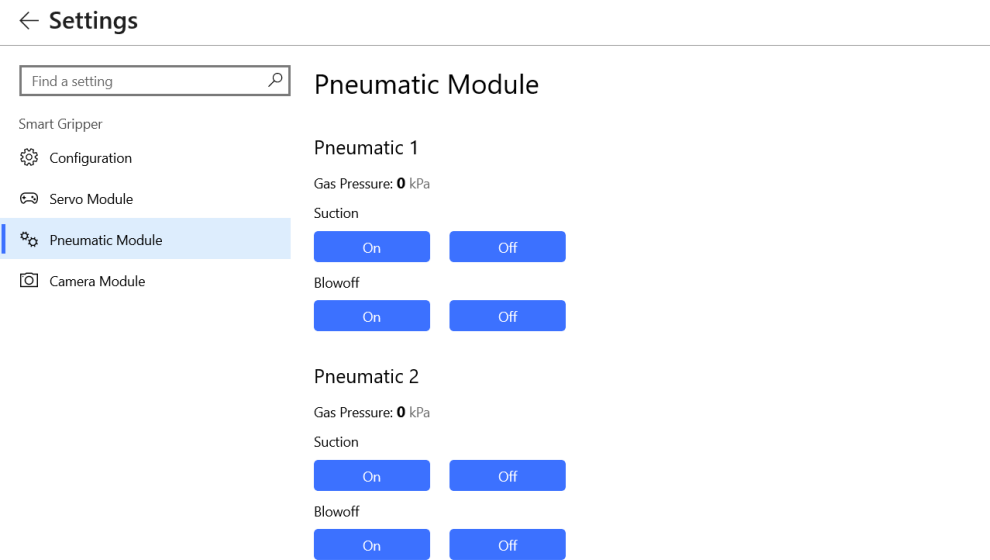
5 Jogging

5.10.1 Introduction
Continued

Pneumatic module page

The following figure describes the options available in the Pneumatic module page of **SmartGripper** settings. This is used for instructing the built-in valves to finish vacuum-sucking and blow-off operations.

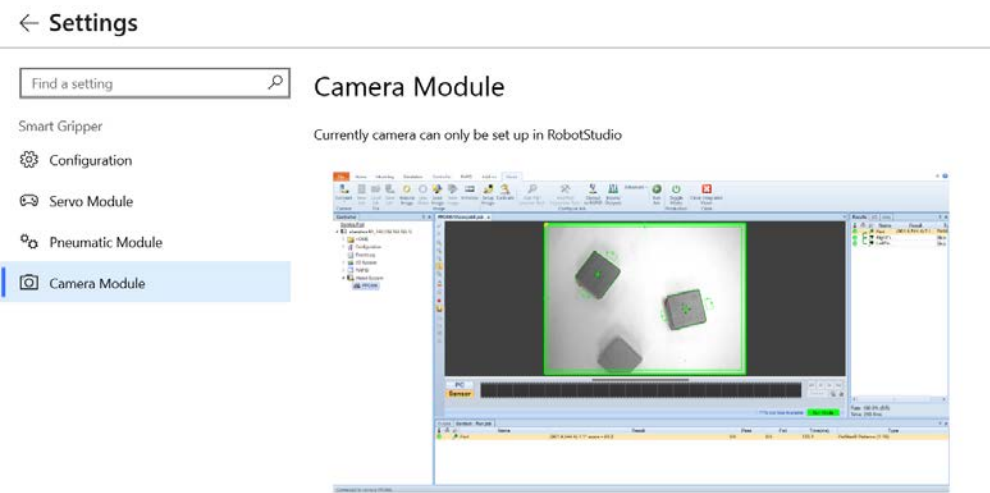
Two pneumatic block parts are available for different variants of the gripper. Suction and blow-off functions are exclusive to each other. That is, if one function is turned on, the other will be turned off.



xx2000001341

Camera module page

The following figure and table describes the options available in the **Camera module** page of the **SmartGripper** settings. This can be configured only from RobotStudio now. For more information of the camera module, see the *Application manual of Integrated Vision*.



xx2000001342

5.10.2 Gripper configuration

Configuring the power of the SmartGripper

The following requirements must be met before configuring the power:

- The gripper has been installed onto the robot arm correctly.
- The RobotWare is RobotWare 7.0 or later
- The **SmartGripper Support** option must be chosen while creating the system.
- The system is in manual operating mode.

Use the following procedure to configure the power of the **SmartGripper**:

- 1 On the start screen, tap **Settings**.
The **Settings** application home page is displayed.
- 2 Tap the **SmartGripper** icon.
The **SmartGripper** settings **Configuration** page is displayed.
- 3 Select the gripper type that has been mounted on the robot from the **Gripper Type** list.
- 4 Tap the **Enable Power of Gripper** button.
The gripper has been powered on and the communication is established.

Calibrating the SmartGripper

The following requirements must be met before calibrating the gripper:

- The gripper has been installed onto the robot arm correctly.
- The RobotWare is RobotWare 7.0 or later
- The **SmartGripper Support** option must be chosen while creating the system.
- The system is in manual operating mode.
- The gripper has been powered on and the communication is established.

Use the following procedure to calibrate the **SmartGripper**:

- 1 On the start screen, tap **Settings**.
The **Settings** application home page is displayed.
- 2 Tap **SmartGripper**.
The **SmartGripper** settings **Configuration** page is displayed.
- 3 On the left sidebar, tap **Servo Module**.
The **Servo Module** page is displayed.
- 4 In the **Command** section, tap and hold the **Grip -** button until the two fingers of the gripper are in direct contact with each other, which is the zero position used for calibration.
- 5 Tap the **Calibrate** button.
When the button color turns blue from gray, the gripper is calibrated and the status is updated.

5.10.3 Smart Gripper function

Procedure

Use the following procedure to work with the **SmartGripper** function:

- 1 On the start screen, tap **Jog**.
- 2 Tap **SmartGripper** tab.
The **SmartGripper** page is displayed.
- 3 If required, enter the value for the gripper force and gripper speed in the **Gripper force** and **Move speed** fields.
- 4 Tap **Open gripper** to open the gripper with the defined gripper speed.
- 5 Tap **Close gripper** to close the gripper with the defined gripper force and speed.
- 6 If required, enter the value in the **Move gripper to position** text box and tap **Move to** button to move the fingers to a specific position.
- 7 Tap the **On** or **Off** button in the **Suction** section to start or stop the suction tool picking up the objects.
- 8 Tap the **On** or **Off** button in the **Blow Off** section to start or stop the suction tool dropping down the objects.

6 Programming and testing

6.1 Introduction

Overview

This chapter provides you information about creating programs and testing those programs.

6 Programming and testing

6.2 Before you start programming

6.2 Before you start programming

Programming tools

You can use both the FlexPendant and RobotStudio for programming. The FlexPendant is best suited for modifying programs, such as positions and paths, while RobotStudio is preferred for more complex programming.

How to program using RobotStudio is described in *Operating manual - RobotStudio*.

Define tools, payloads, and work objects

Define tools, payloads and work objects before you start programming. You can always go back and define more objects later, but you should define your basic objects in advance.

See [Creating a tool on page 156](#).



WARNING

It is important to always define the actual tool load and, when used, the payload of the robot (for example, a gripped part). Incorrect definitions of load data can result in overloading of the robot mechanical structure. There is also a risk that the speed in manual reduced speed mode can be exceeded.

When incorrect load data is specified, it can often lead to the following consequences:

- The robot may not use its maximum capacity.
- Impaired path accuracy including a risk of overshooting.
- Risk of overloading the mechanical structure.

The controller continuously monitors the load and writes an event log if the load is higher than expected. This event log is saved and logged in the controller memory.

Define coordinate systems

Make sure the base and world coordinate systems have been set up properly during the installation of your robot system. Also make sure that additional axes have been set up.

Define tool and work object coordinate systems before you start programming. As you add more objects later you also need to define the corresponding coordinate systems.



Tip

For more details about the RAPID language and structure, see *Technical reference manual - RAPID Overview* and *Technical reference manual - RAPID Instructions, Functions and Data types*.

Continues on next page

Naming convention

The RAPID language is constructed using the standard ISO 8859-1 (Latin-1) character set. In addition, newline, tab, and form feed control characters are recognized. Everything such as names of modules, variables, folders, etc. must use symbols from the ISO 8859-1 character set.

Naming of files must follow the rules of the operating system on the computer.

UTF-8 symbols are supported only in the following places:

- Comments in RAPID code
- Content in RAPID strings

6 Programming and testing

6.3.1 Handling of programs

6.3 Programming concept

6.3.1 Handling of programs

Overview

This section provides information about the handling of robot programs. It describes how to:

- create a new program
- load an existing program
- save a program
- rename a program
- delete a program



Note

Each task must contain *one* program. The procedures in this section describes a single task system.

How to create a new program *when no program is available* is detailed in section [Creating a new program on page 117](#).

Program Editor

In FlexPendant the RAPID programs are created and edited using **Program Editor** in **Code**.

If you toggle between the **Program Editor** and another view and back again, the **Program Editor** will show the same part of the code as long as the program pointer has not been moved. If the program pointer is moved, the **Program Editor** shows the code at the position of the program pointer.

The same behavior applies to **Advanced View** in **Operate**.



Note

When you open **Code**, the name of the module where the program pointer is in is displayed in the **Program Pointer** location section.

About program files

The program is saved as a folder, named as the program, containing the actual program file, of type pgf.

When loading a program you open the program folder and select the pgf file.

When renaming a program you rename the program folder and the program file.

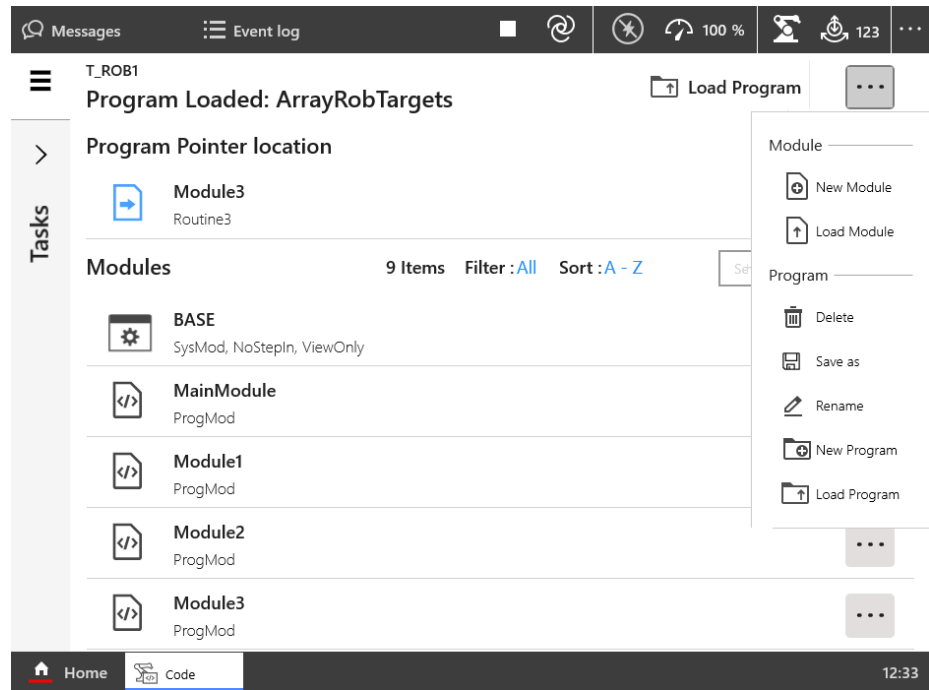
When saving a loaded program which is already saved to the hard disk, you must not open the existing program folder. Instead, you should save the program folder again and overwrite the old version, or rename the program.

Continues on next page

Creating a new program

This section describes how to create a new program.

- 1 On the start screen, tap **Code**, and then select **Modules** from the menu.
- 2 On the **Context** menu, tap **New Program**.



xx1900000214

If there was already a program loaded, a warning dialog appears:

- Tap **Save** to save the loaded program.
- Tap **Don't Save** to close loaded program without saving it, i.e. delete from program memory.
- Tap **Cancel** to leave the program loaded.

- 3 Tap **Main Module**.
- 4 Add instructions to the program.
For details regarding adding instructions, see [Adding instructions on page 134](#).
- 5 Tap **Check Program**.
- 6 Tap **Modules**.
- 7 On the **Context** menu, tap **Save as**.
- 8 Tap a name for the program in the **File Name** field.
- 9 Select the location for saving the new program file.
- 10 Tap **Save**.

The program is saved.

Once a program is created you can run the program. For details regarding running a program, see [Starting programs on page 230](#).

Saving a program

This section describes how to save a loaded program to the controller hard disk.

Continues on next page

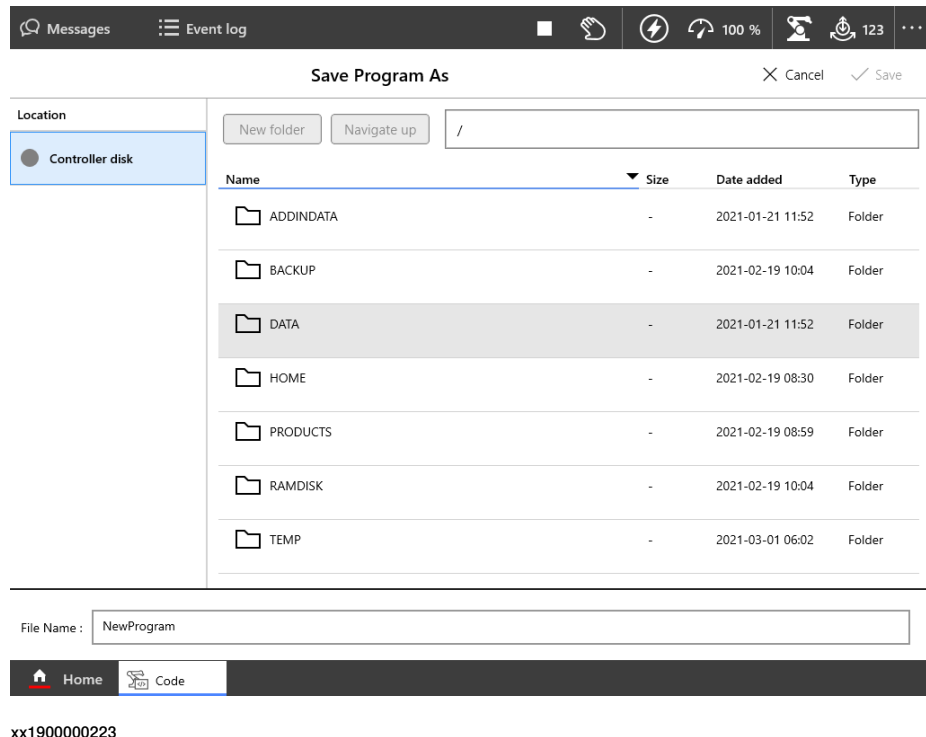
6 Programming and testing

6.3.1 Handling of programs

Continued

A loaded program is automatically saved in the program memory, but saving to the controller hard disk is an extra precaution.

- 1 On the start screen, tap **Code**, and then select **Modules** from the menu.
- 2 On the **Context menu**, tap **Save as**.
- 3 Use the suggested program name or type a new name in the **File Name** field.



- 4 Tap **Save**.
The program is saved.

Renaming a loaded program

This section describes how to rename a loaded program.

- 1 On the start screen, tap **Code**, and then select **Modules** from the menu.
- 2 On the **Context menu**, tap **Rename**.
- 3 In the **Enter New Name** field, type a new name for the selected program.

Continues on next page



xx1900000224

4 Tap **Apply**.

The program is renamed.

Deleting a program

This section describes how to delete a program.



Note

You can only delete a loaded program.

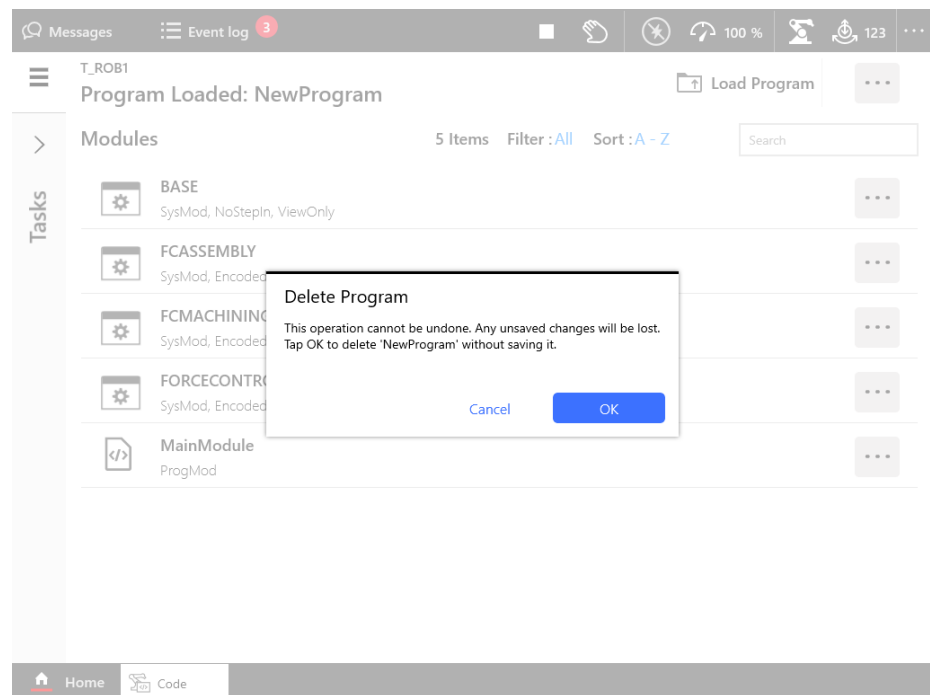
- 1 On the start screen, tap **Code**, and then select **Modules** from the menu.
- 2 On the **Context** menu, tap **Delete**.
- 3 In the **Delete Program** confirmation window, tap **OK** to delete, or **Cancel** to keep the program intact.

Continues on next page

6 Programming and testing

6.3.1 Handling of programs

Continued



xx1900000225

6.3.2 Handling of modules

Overview

This section details how to handle program modules. i.e.:

- create a new module
- load an existing module
- save a module
- rename a module
- delete a module

Creating a new module

This section describes how to create a new module.

- 1 On the start screen, tap **Code**, and then select **Modules** from the menu.
- 2 On the **Context menu**, tap **New Module**.
- 3 In the **Create New Module** window, enter a **Module Name**, and select if **Module Type** should be **Program** or **System**.



Note

How to later switch between these types is detailed in section [Changing type of module on page 124](#).

- 4 Tap **Apply**.

The module is created and displayed in the **Modules** section.

File format for modules



Note

In RobotWare 7.0 and earlier, the formats were `.mod` and `.sys`. When loading these in a RobotWare 7.1 controller or later using RobotStudio, they are automatically converted when saved. When saved, the new file extensions are `.sysx` and `.modx`. Note that the files must be converted, not just renamed.

To convert a file manually, the file must be saved as UTF-8 without BOM (Byte Order Mark).

Loading an existing module

Use the following procedure to load an existing module.

- 1 On the start screen, tap **Code**, and then select **Modules** from the menu.
- 2 On the **Context menu**, tap **Load Module**.
- 3 Navigate and select the module from the location where it is saved.
- 4 Tap **Load**.

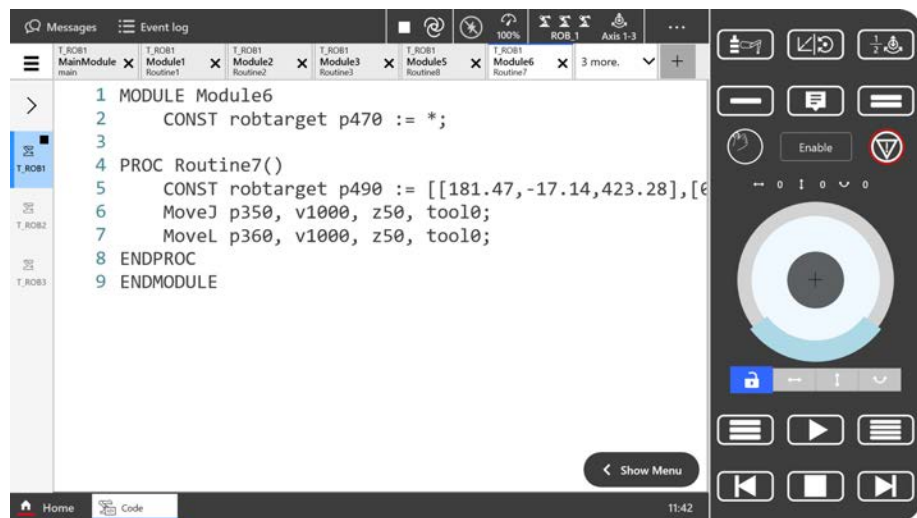
The selected module is loaded.

Continues on next page

6 Programming and testing

6.3.2 Handling of modules

Continued



xx2400001327



Note

When several modules are loaded, each module is displayed in a separate tab. If the number of loaded modules crosses a limit, the subsequent modules are hidden under the last displayed tab. A down arrow with information about the number of hidden modules is displayed on the last tab. Tap on it to access the hidden modules.



Note

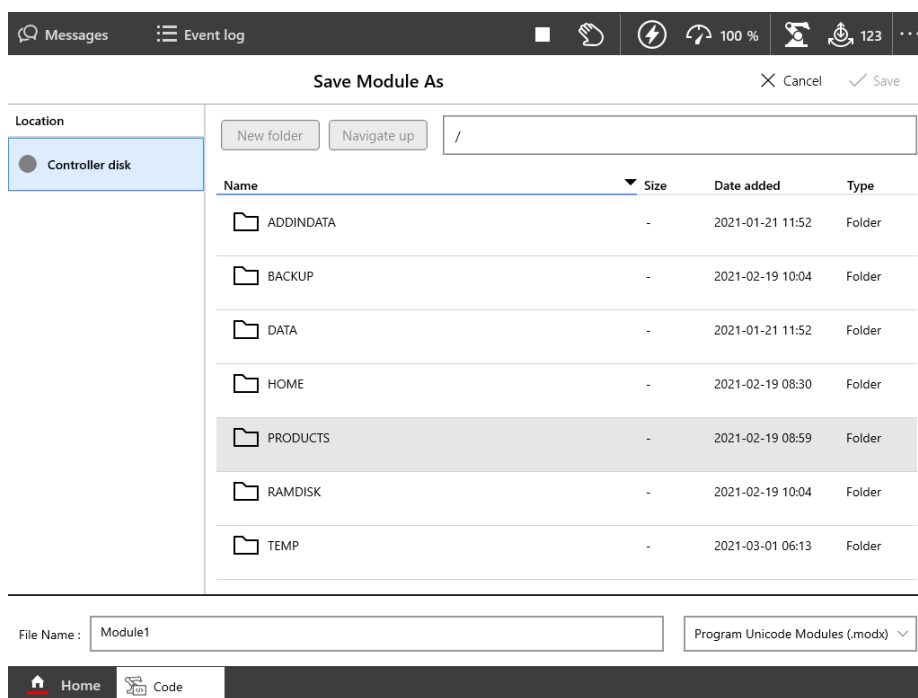
You can close all the hidden modules by selecting down arrow and tapping the **Close Modules Below** button.

Saving a module

This section describes how to save a module.

- 1 On the start screen, tap **Code**, and then select **Modules** from the menu.
- 2 Tap **Save as** on the **Context menu** for the module.
- 3 The **Save Module as** window is displayed:

Continues on next page



xx1900000229

Tap and select a location for saving the module.

Use the suggested module name or enter a **File Name**.

4 Tap **Save**.

The module is saved in the selected location.

Renaming a module

This section describes how to rename a module.

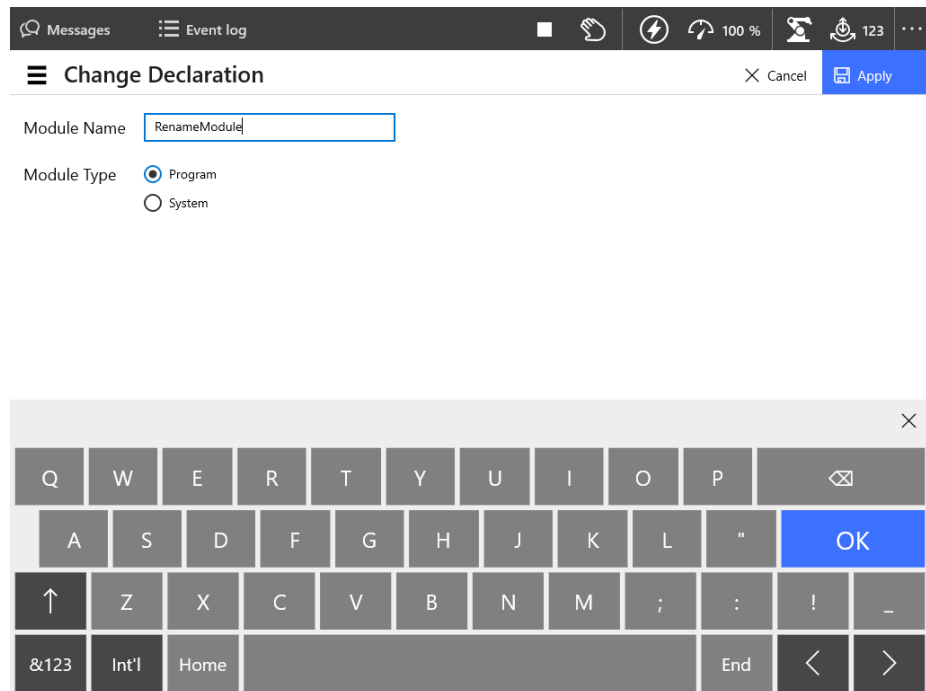
- 1 On the start screen, tap **Code**, and then select **Modules** from the menu.
- 2 Tap **Change Declaration** on the **Context** menu for the module.
The **Change Declaration** window is displayed.

Continues on next page

6 Programming and testing

6.3.2 Handling of modules

Continued



xx1900000230

- 3 In the **Module Name** field type a new name for the module and then tap **Apply**.

The module is renamed.

Changing type of module

This section describes how to change the type of module.

- 1 On the start screen, tap **Code**, and then select **Modules** from the menu.
- 2 Tap **Change Declaration** on the **Context menu** for the module.
The **Change Declaration** window is displayed.
- 3 In the **Module Type** list select a type and then tap **Apply**.

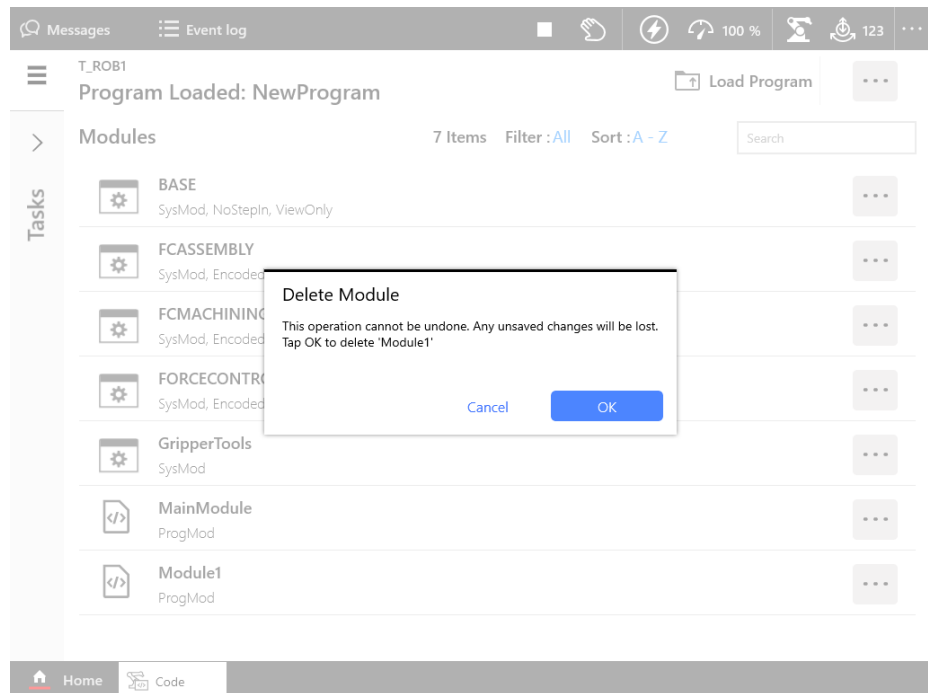
The module type is changed.

Deleting a module

This section describes how to delete a module from memory. If the module has been saved to disk, it will not be erased from the disk.

- 1 On the start screen, tap **Code**, and then select **Modules** from the menu.
- 2 Tap **Delete** on the **Context menu** for the module.
- 3 The **Delete Module** confirmation window is displayed:

Continues on next page



xx1900000231

4 Tap OK.

The selected module is deleted and removed from the module list.

6 Programming and testing

6.3.3 Handling of routines

6.3.3 Handling of routines

Overview

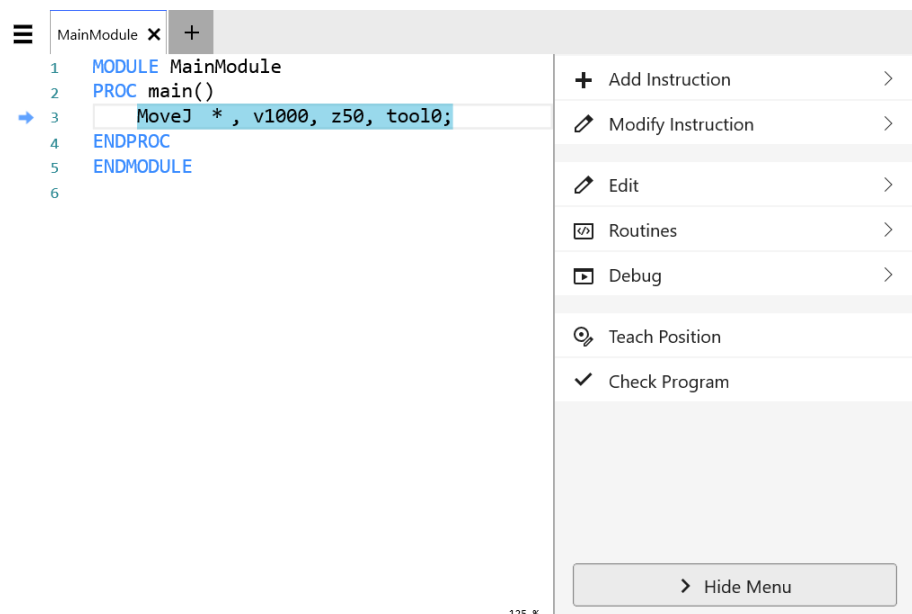
This section details how to handle program routines. i.e.:

- create a new routine
- create a copy of a routine
- change the declaration of a routine
- delete a routine

Creating a new routine

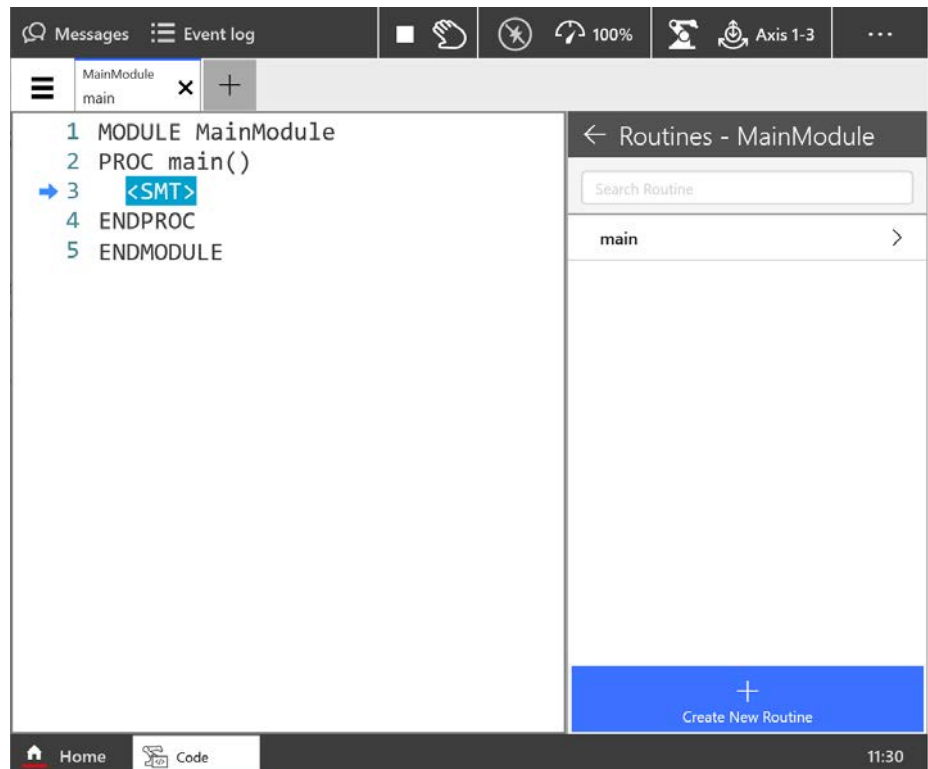
This section details how to create a new routine, set the declaration, and add it to a module.

- 1 On the start screen, tap **Code**, and then select **Program editor** from the menu.
- 2 Tap **Routines** in the menu to the right.



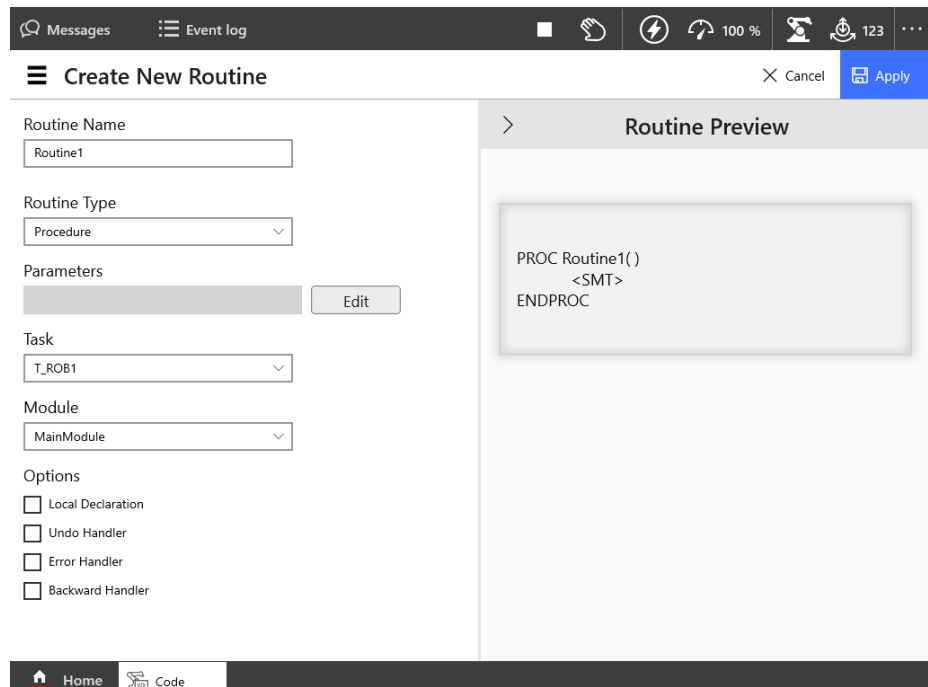
- 3 Tap **Create New Routine**.

Continues on next page



xx1900000251

4 The Create New Routine page is displayed:



xx1900000254

Complete the routine declaration by entering the following information:

- **Routine Name**

Continues on next page

6 Programming and testing

6.3.3 Handling of routines

Continued

- **Routine Type**

- Procedure: used for a normal routine without return value
- Function: used for a normal routine with return value
- Trap: used for an interrupt routine

- **Parameters**

Tap **Edit** to add parameters to the routine. See section [Defining parameters in routine on page 128](#) for more information.

- **Task**

- **Module**

- **Options**

- Local Declaration

Tap the checkbox to select **Local declaration** if the routine should be local.

A local routine can only be used in the selected module.

- Undo Handler
- Error Handler
- Backward Handler



Note

Use the **Preview** button to preview the values selected for the new routine.

5 Tap **Apply**

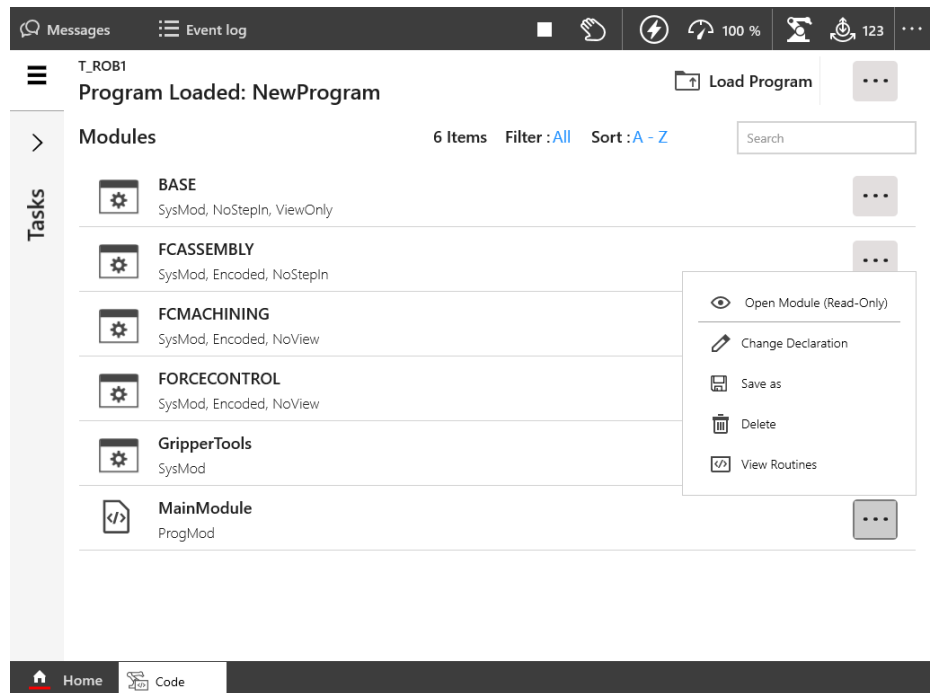
The new routine is created and displayed in the **Routines** list for the selected module.

Defining parameters in routine

This section describes how to define parameters in a routine.

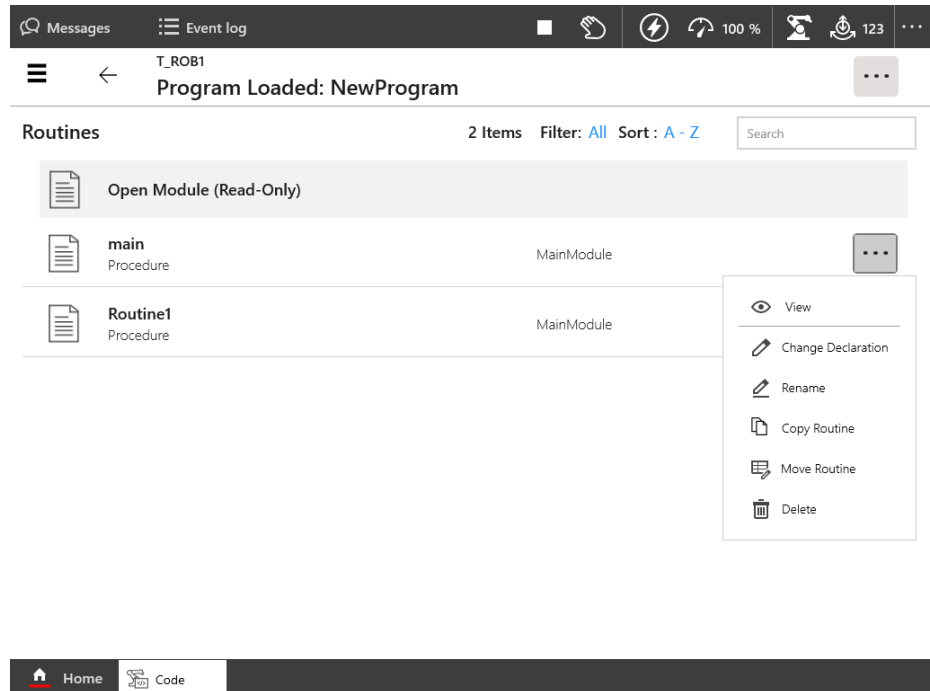
- 1 On the start screen, tap **Code**, and then select **Modules** from the menu.
- 2 Tap **View Routines** on the **Context menu** for the module.

Continues on next page



xx1900000228

3 Tap Change declaration on the Context menu for the routine.



xx1900000271

The **Change Declaration** page is displayed.

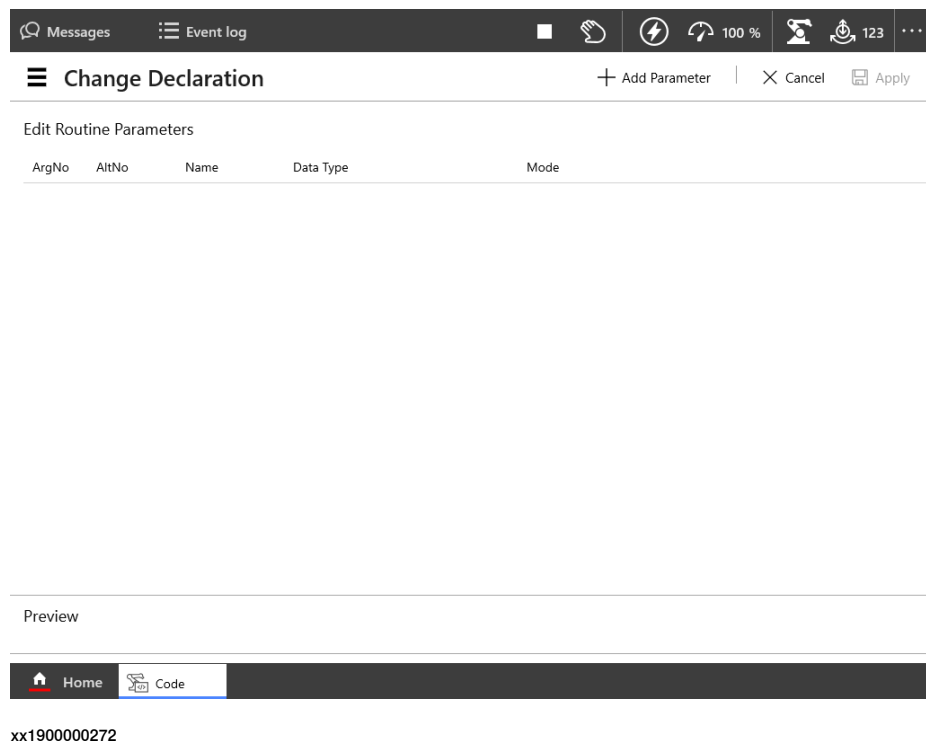
4 If no parameters are shown, tap the **Edit** button next to the **Parameters** field. The **Edit Routine Parameters** page is displayed.

Continues on next page

6 Programming and testing

6.3.3 Handling of routines

Continued



- 5 Tap **Add Parameter** and select **Mandatory Parameter** or **Optional Parameter** according to your requirement.

The selected parameter is added to the **Edit Routine Parameters** list.



Note

Select an optional parameter and tap **Add Parameter > Optional Mutual Parameter** to add a parameter that is mutually optional with another parameter.

Read more about routine parameters in the RAPID reference manuals.

- 6 Type a **Name** for the parameter and tap **Apply**.
- 7 The new parameter is displayed in the list. Tap to select a parameter. To edit values, tap the value.
- 8 Tap **Apply**.

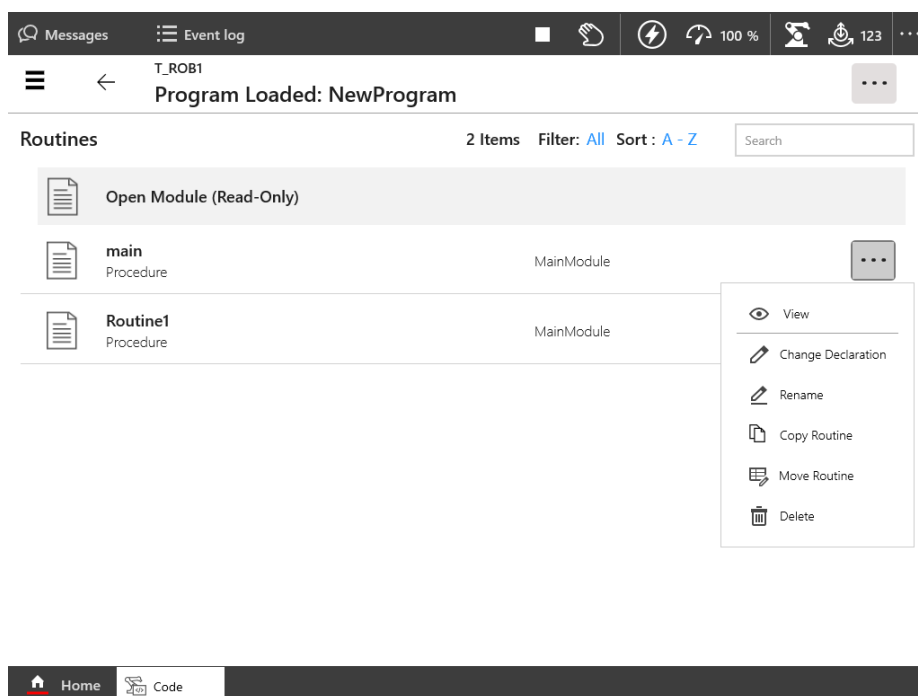
The selected parameters are added to the **Parameters** field in the routine declaration window.

Creating the copy of a routine

This section describes how to create a copy of a routine.

- 1 On the start screen, tap **Code**, and then select **Program Editor** from the menu.
- 2 Tap **View Routines** on the **Context menu** for the module.
The **Routines** window is displayed.

Continues on next page



xx1900000271

- 3 Tap **Copy Routine** on the **Context** menu for the routine.

The **Copy Routine** dialog is displayed.

- 4 Edit the name or other parameters according to your requirement.

- 5 Tap **Apply**.

A copy of the selected routine is created.

How to make all declarations is detailed in section [Creating a new routine on page 126](#).

Changing the declaration of a routine

This section describes how to change the declaration of a routine.

- 1 On the start screen, tap **Code**, and then select **Program Editor** from the menu.
 - 2 Tap **View Routines** on the **Context** menu for the module.
 - 3 Tap **Change Declaration** on the **Context** menu for the routine.
- The **Change Declaration** dialog is displayed.
- 4 Edit the values according to your requirement.
 - 5 Tap **Apply**.

The changes to the routine are saved.

How to make all declarations is detailed in section [Creating a new routine on page 126](#).

Continues on next page

6 Programming and testing

6.3.3 Handling of routines

Continued

Moving a routine

This section describes how to move a routine to another module.

- 1 On the start screen, tap **Code**, and then select **Program Editor** from the menu.
- 2 Tap **View Routines** on the **Context menu** for the module.
- 3 Tap **Move Routine** on the **Context menu** for the routine.

The **Move Routine** dialog is displayed.

The screenshot shows the 'Move Routine' dialog box. On the left, there are fields for 'Routine Name' (set to 'main'), 'Routine Type' (set to 'Procedure'), 'Parameters' (with an 'Edit' button), 'Task' (set to 'T_ROB1'), and 'Module' (set to 'GripperTools'). Below these are 'Options' with checkboxes for 'Local Declaration', 'Undo Handler', 'Error Handler', and 'Backward Handler'. On the right, the 'Routine Preview' section shows the code: `PROC main()
<SMT>
ENDPROC`. At the bottom of the dialog are 'Cancel' and 'Apply' buttons. The background shows the software's top bar with 'Messages' and 'Event log' tabs, and a bottom bar with 'Home' and 'Code' tabs.

- 4 Select the **Task** and **Module** to which the routine should be moved. Then tap **Apply**.

Renaming a routine

This section describes how to rename a routine.

- 1 On the start screen, tap **Code**, and then select **Program Editor** from the menu.
- 2 Tap **View Routines** on the **Context menu** for the module.
- 3 Tap **Rename** on the **Context menu** for the routine.
- 4 Type a new name for the routine in the **Enter New Name** field.
- 5 Tap **Apply**.

The selected routine is renamed.

Deleting a routine

This section describes how to delete a routine from memory.

- 1 On the start screen, tap **Code**, and then select **Program Editor** from the menu.

Continues on next page

- 2 Tap **View Routines** on the **Context menu** for the module.
- 3 Tap **Delete** on the **Context menu** for the routine.
The **Delete Routine** conformation window is displayed.
- 4 Tap **OK**.
The selected routine is deleted.

6.3.4 Handling of instructions

Instructions

A RAPID program consists of instructions. An instruction can, for example, move the robot, set an I/O signal, or display a message to the operator.

A large number of instructions are available, and these are listed in *Technical reference manual - RAPID Instructions, Functions and Data types*. The basic procedure for adding instructions are, however, identical.

Adding instructions

Use the following procedure to add instructions to a routine in a module. If no routine exists, this must be added first, see [Creating a new routine on page 126](#).

- 1 On the start screen, tap **Code**, and then select **Modules** from the menu.
- 2 Open the module on which instruction needs to be added.
- 3 Tap on the location where the instruction needs to be inserted.
- 4 Tap **Add Instruction**.

The **Add Instruction** panel is displayed.

- 5 Select an instruction from **Common** or **Groups** tab according to your requirement.

The selected instruction is displayed with its parameters.



Note

While adding or modifying a Move instruction, the editing of **ToPoint** field is disabled.

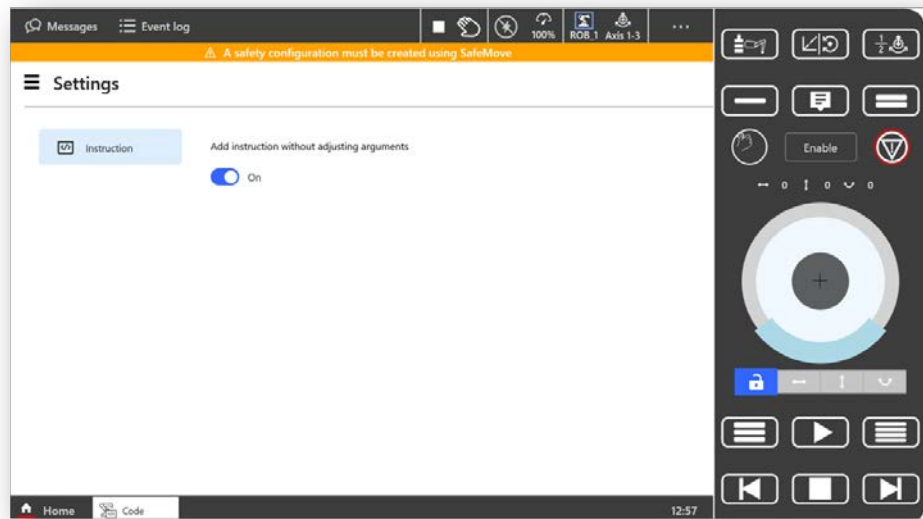
- 6 Modify the instruction parameters according to requirement and tap **Add**.
The selected instruction is inserted after the selected location on the module.
- 7 Tap **Check Program**.
The validity of the program is verified.

Enable Add instruction without adjusting arguments button

User can add the instruction directly without modifying the arguments in the routine. When **Add instruction without adjusting arguments** toggle button is enabled from

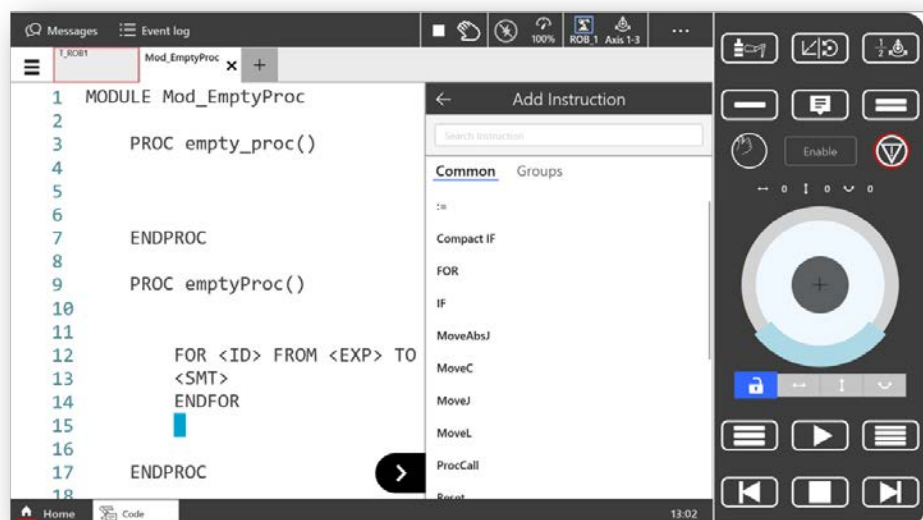
Continues on next page

the **Codes** → **Settings** → **Instruction** path, the selected instruction is not displayed in **Add Instruction** window, where you add the values for each instruction.



Add instruction at empty line in routine

User can now add the instructions in the empty lines in routine. The procedure to add the instruction remains the same by selecting the empty line, the selected instructions in **Add Instruction** is displayed, where you select and add the instructions.



Modifying instructions

Use the following procedure to modify an instruction:

- 1 On the start screen, tap **Code**, and then select **Modules** from the menu.
- 2 Open the module on which instruction need to be edited.
- 3 Tap on the instruction that needs to be edited.

Continues on next page

6 Programming and testing

6.3.4 Handling of instructions

Continued

4 **Tap Modify Instruction.**

The selected instruction is displayed with its arguments.

5 **Edit the instruction arguments.**



Note

Tap **Expression Editor** to open and edit the selected instruction in Expression Editor window.



Note

For the block instructions (FOR, IF, CompactIF, TEST, and WHILE), tap the menu next to an expression, and select **Modify Expression** to edit it.

6 **Tap Apply.**

The changes are updated in the selected instruction.

7 **Tap Check Program.**

The validity of the modified program is verified.

Editing an instruction argument that has array data

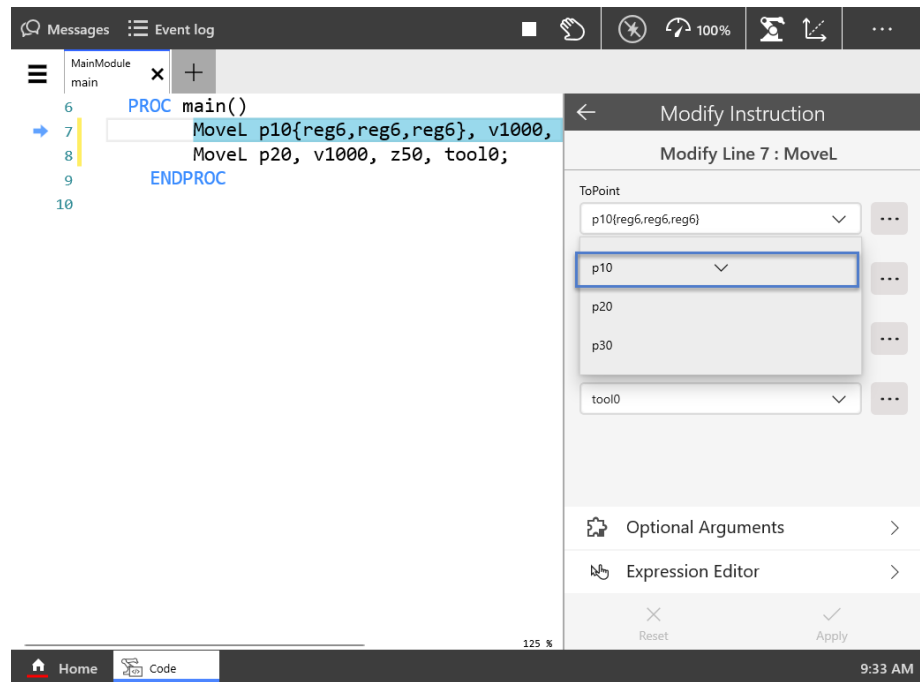
The programs that are created using RobotStudio may have instruction arguments that has array data. This section provides information regarding modifying an instruction argument that has array data:

1 **Tap on the instruction that needs to be edited.**

2 **Tap Modify Instruction.**

The selected instruction is displayed with its arguments. The argument item that has array data displays a down arrow next to it.

Continues on next page



xx2000001284

- 3 Tap on this arrow to expand the list.
The elements in the array are displayed.
- 4 Tap on an array element.
- 5 Make the required changes.
- 6 Tap **Apply**.
The changes are updated in the selected instruction.

Update position

When you edit an instruction that has a position argument, and in case you jog the robot to a different position, the change in position will not be automatically updated in position argument. To update the position argument with the current position values, the **Update Position** function is used.

Use the following procedure to update position while editing an instruction:

- 1 Tap on the instruction that has a position argument that needs to be modified.
- 2 Tap **Modify Instruction**.
- 3 Jog the robot to a new position according to your requirement.
- 4 Tap on the button available next to the position argument and select **Update Position**.
The **Update Position** window is displayed.
- 5 Tap **Update**.
The current robot position value is set in the selected position argument.
- 6 Tap **Apply**.
The selected instruction (with the updated position values) is updated.

Continues on next page

6 Programming and testing

6.3.4 Handling of instructions

Continued



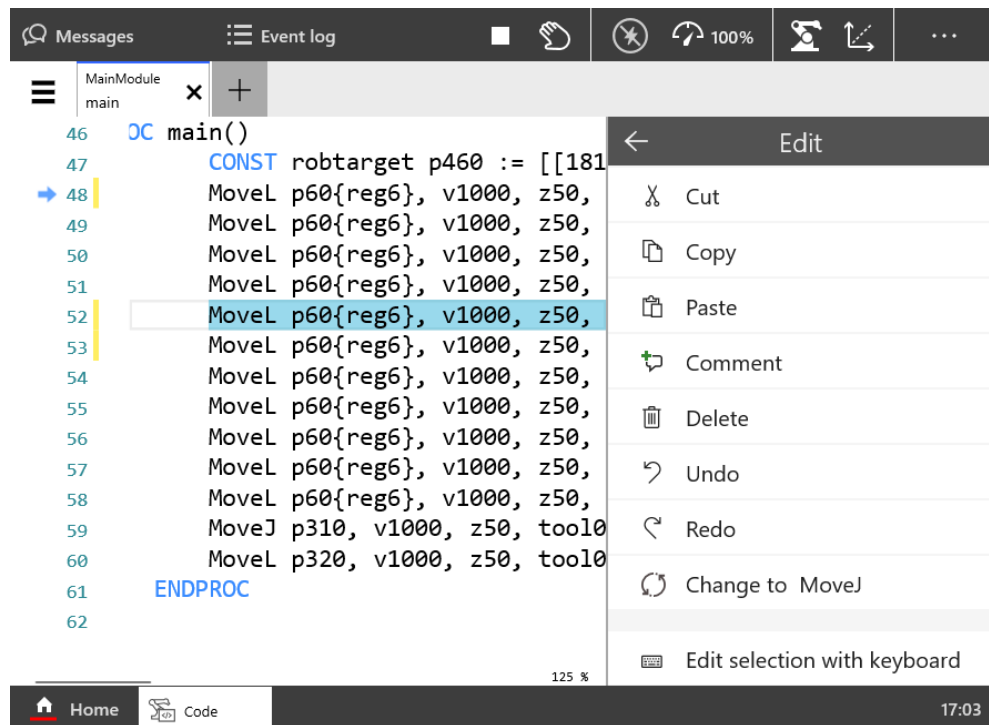
Note

The **Update Position** function is not available for array data position argument.

Edit options

To edit programs in **Program Editor** you can use edit options. The edit options are available in **Edit** under **Edit and Debug** section.



The following image and table provides information regarding edit options and its description.



xx2100000046

| Edit options | Description |
|----------------|--|
| Cut | Cuts a selected instruction and save it in clipboard. |
| Copy | Copies a selected instruction and save it in clipboard. |
| Paste | Pastes the copied instruction below a selected instruction. <div> Note If the selected instruction is the first line of the program, an option to paste the copied instruction Above or Below the first line is displayed.</div> |
| Comment | Comments a selected instruction. The commented instruction is skipped during the program execution. <div> Note Once you enable the Comment option in an instruction the name of the option is changed to Uncomment. This option which allows you to remove the Comment from the instruction.</div> |

Continues on next page

| Edit options | Description |
|---|---|
| Delete | Deletes a selected instruction. |
| Undo | Cancels the previous edit operation. You can undo a maximum of three steps. |
| Redo | Reverts the previous undo operation. You can redo a maximum of three steps.  Note Redo option is enabled only after an undo operation. |
| Change to <shift instruction name> | Changes the selected instruction to its shift instruction. |
| Select Range | For selecting a range of instructions for editing, tap on the first instruction of the range, tap Select Range , and then tap on the last instruction of the range. You can then perform various edit operations like Cut, Copy, Paste, and Delete for the selected range. |
| Edit selection with keyboard | Opens the selected instruction in advanced editor for editing.  Note For a block/conditional instruction it is possible to edit only a selected expression. |

Copying and pasting instructions or arguments

This section describes how to paste instructions or arguments.

- 1 On the start screen, tap **Code**, and then select **Modules** from the menu.
- 2 Open the module.
- 3 Tap on the instruction that needs to be copied.
- 4 Tap **Edit**.
The edit options window is displayed.
- 5 Tap **Copy**.
- 6 Tap and select the instruction after which you want to paste the copied instruction.
- 7 Tap **Paste**.

The copied instruction is pasted below the selected instruction.



Note

If the selected instruction is the first line of the program, an option to paste the copied instruction **Above** or **Below** the first line is displayed.

Commenting instruction rows

Use the following procedure to comment an instruction.

- 1 On the start screen, tap **Code**, and then select **Modules** from the menu.
- 2 Open the module.
- 3 Tap on the instruction that you want to comment.

Continues on next page

6 Programming and testing

6.3.4 Handling of instructions

Continued

4 Tap **Edit**.

The edit options window is displayed.

5 Tap **Comment**.

The selected instruction is commented and is skipped during the program execution.

Deleting an instruction

This section describes how to delete an instruction.

1 On the start screen, tap **Code**, and then select **Modules** from the menu.

2 Open the module.

3 Tap on the instruction that needs to be deleted.

4 Tap **Edit**.

The editing options window is displayed.

5 Tap **Delete**.

The selected instruction is deleted from the module.

Shifting an instruction

It is possible to change a selected instruction to its equivalent shift instruction.



Note

Parameter restrictions are enabled for certain instructions. Shifting to an equivalent instruction is not valid for such instructions.

Use the following procedure to shift an instruction.

1 On the start screen, tap **Code**, and then select **Modules** from the menu.

2 Open the module.

3 Tap on the instruction that you want to shift.

4 Tap **Edit**.

The editing options window is displayed.

5 Tap **Change to <shift instruction name>**.

The selected instruction is changed to its shift instruction.

Continues on next page

Scenarios while editing a line that contains multiple instructions

Following are some scenarios while editing a line that contains multiple instructions or comments:

- **A line that has an instruction and a comment:**
 - Selecting the instruction and then adding a new instruction *below the current line* moves the comment to the newly added instruction.
- **A line that has multiple instructions:**
 - Selecting the first instruction and then adding a new instruction *below the current line* adds the instruction after the selected instruction. It also moves the rest of the instructions to the new line.
 - Selecting the first instruction and then commenting or uncommenting comments/uncomments the whole line.
 - Selecting the second instruction or subsequent instructions and then commenting or uncommenting comments/uncomments all the instructions from the selection.

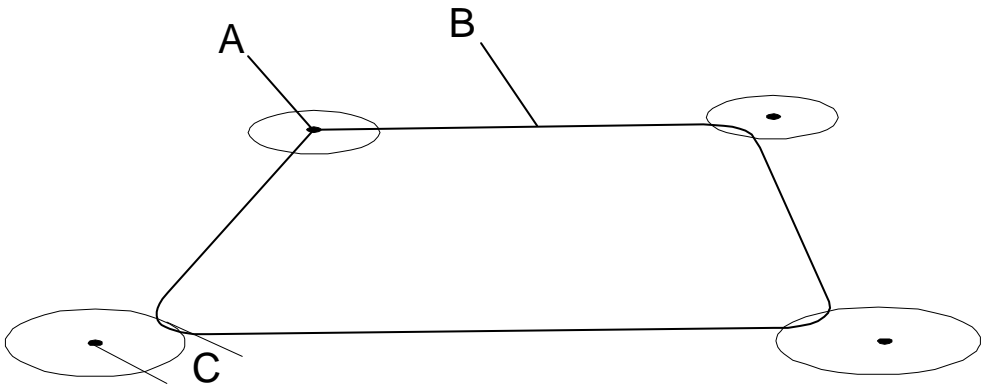
Edit selection with keyboard

It is possible to open a selected instruction in editor for editing.

6.3.5 Example: Add movement instructions

Overview

In this example you will create a simple program that makes the robot move in a square. You need four movement instructions to complete this program.



en0400000801

| | |
|---|--|
| A | First point |
| B | Robot movement Speed data v50 = speed 50mm/s |
| C | Zone z50 = (50mm) |

Add movement instructions

Use the following procedure to write a simple program that makes the robot move in a square:

- 1 On the start screen, tap **Code**, and then select **Modules** from the menu.
- 2 Open a module.
- 3 Jog the robot to the first point.
- 4 Tap **Add Instruction**.
- 5 Tap **MoveL**.
- 6 Tap **Add**.
- 7 Repeat the steps 4 to 7 and add the other three positions of the square.

The MoveL instructions for the selected points are added.



Note

For the first and last instruction, select the instruction and tap **Edit Instruction**. Then change the Zone to **Fine**.

- 8 Tap **Apply**.
The program is saved.

Result

Your program code should look like this:

```
Proc main()
```

Continues on next page

```
MoveL *, v50, fine, tool0;  
MoveL *, v50, z50, tool0;  
MoveL *, v50, z50, tool0;  
MoveL *, v50, z50, tool0;  
MoveL *, v50, fine, tool0;  
End Proc;
```

6 Programming and testing

6.3.6 Program and motion pointers

6.3.6 Program and motion pointers

The Program pointer

The Program Pointer (PP) indicates the instruction with which the program will start when you press the **Start**, **Forward**, or **Backward** buttons on the FlexPendant. Program execution continues from the instruction where the Program Pointer is placed. However, if the cursor is moved to another instruction when the program is stopped, the Program Pointer can be moved to the position of the cursor (or the cursor can be moved to the Program Pointer), and execution can be restarted from that position.

The Program Pointer is shown as an arrow to the left of a line in program code in **Program Editor of Code** and in **Advanced View of Operate**.

The Motion pointer

The Motion Pointer (MP) indicates the instruction that the robot is currently executing. This is normally one or more instructions after the Program Pointer, as the system executes and calculates the robot path faster than the robot moves.

The Motion Pointer is shown as a small robot to the left of the program code in **Program Editor of Code** and in **Advanced View of Operate**.

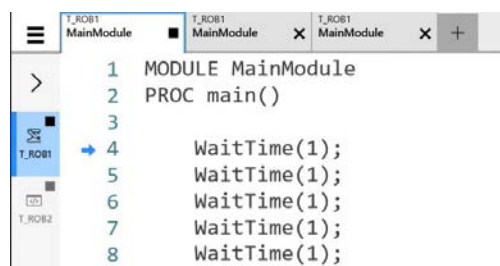
The cursor

The cursor indicates a complete instruction or any of the arguments.

The cursor is shown as blue highlighting of the program code in **Program Editor of Code**.

The execution tab

Apart from the text editor tabs in which you can modify the actual code, the **Code** application also contains an execution tab that displays the current position of the Program Pointer (PP). The execution tab, which is positioned at the far left of the top bar, allows the user to easily switch between different sections or views.



xx2400002051



Note

If no Program Pointer (PP) has been set, the execution tab displays the main routine.

6.4 Debugging the program

Introduction

The Debug option allows you to navigate through a RAPID program for understanding the execution flow, viewing the values of a data instance, and to run the routines.

Moving the pointers

The pointers helps you to navigate through a program. For more information about pointers, see [Program and motion pointers on page 144](#).

The following table provides the description of the options available under **Debug** in **Program Editor** for the movement of pointers:

| Pointer position | Description |
|------------------|---------------------------------------|
| PP to Main | Moves the program pointer to main |
| PP to Cursor | Moves the program pointer to cursor. |
| PP to Routine | Moves the program pointer to routine. |
| Cursor to PP | Moves the cursor to program pointer. |
| Cursor to MP | Moves the cursor to motion pointer. |

Moving the robot

While navigating through the program you may need to move the robot to a particular programmed position. The **Go To Position** option allows you to move the robot to a programmed position.



DANGER

When moving the robot automatically, the robot arm may move without warning. Make sure no personnel are in safeguarded space and that no objects are in the way between the current position and the programmed position.

Use the following procedure to move the robot to a particular programmed position:

- 1 On the program select a programmed position.
- 2 Tap **Debug**.
- 3 Tap **Go To Position**.
The **Go to position** window is displayed.
- 4 Press and hold the three-position enabling device and then tap and hold the **Go To** button.
The robot is moved from the current position to the selected programmed position.



Note

For collaborative robots, after selecting a programmed position, you need to just press and hold the **Go To** button to move the robot.

Continues on next page

6 Programming and testing

6.4 Debugging the program

Continued

Editing the values of a data instance

While navigating through a RAPID program you may need to edit or view the values of a data instance used in a RAPID program.

Use the following procedure to edit (or view) the values of a data instance in a program:

- 1 On the program select the data instance for which you want to edit the values.
- 2 Tap **Debug**.
- 3 Navigate to the **Others** section and tap **View Value**.
The values of the selected data instance are displayed.
- 4 Edit or view the values according to your requirement.



Note

For information regarding Viewing or editing values of a data instance that has array data, see [Editing an instruction argument that has array data on page 136](#).

- 5 Tap **Apply**.
The changes are saved.

Calling a routine

You can call a routine and check the execution flow from the Debug menu. It is possible to call the default service routines as well as the custom routines.

Use the following procedure to call a routine and check the execution flow:

- 1 In the **Debug** menu navigate to the **Others** section and tap **Call Routine**.
The **Call Routine** window is displayed. By default the service routines are displayed. Select the **All Routines** option to list all the routines.
- 2 Select a routine.
- 3 Tap **Go to**.
The routine code is displayed.



Note

For service routines the code is not displayed.

- 4 Tap **QuickSet** and run the routine.
- 5 Tap **Cancel Call Routine**.
The selected routine is cancelled.

6.5 Data types

6.5.1 Edit data in specific tasks, modules, or routines

Editing data in specific tasks, modules, or routines

It is possible to view, copy, modify, or delete selections of data types by selecting a specific scope.

Use the following procedure to view data instances in specific modules or routines:

- 1 On the start screen, tap **Program Data**.
- 2 Select the required scope. The following options are available:
 - **Built-In Data Only:** Displays all the data types used by the specific system.
 - **Current Execution:** Displays all the data types used in the current execution.

The following sub options are available. To access the sub options tap on **Change**.

- **Task:** Displays all the data types used by a specific task.
 - **Module:** Displays all the data types used by a specific module.
 - **Routine:** Displays all the data types used by a specific routine.
- 3 Select the required view. The data type based on the selected view is displayed.

The following view options are available:

- **Only used types:** Displays only the used data types.
- **All data types:** Displays all the data types.

- 4 Tap on a data type.

The items for the selected data type is displayed.

- 5 Tap on the context menu for an item.

The following options are available:

- **Edit Value:** Allows you to modify the values saved for the selected item.
- **Edit Declaration:** Allows you to modify the declaration elements.
- **Copy:** Creates a copy of the selected item with a new name.
- **Delete:** Deletes the selected item.



Note

For the data type `robtarget` the option **Update Position** is also displayed. This option helps you to update the position argument with the position values of the selected `robtarget`.

6.5.2 Creating new data instance

Creating new data instance

This section details how to create new data instances of data types.

- 1 On the start screen, tap **Program Data**.

The **Data Types** dialog is displayed.



Tip

Define what data to be displayed by selecting either **Only used types** or **All data types**.

- 2 Tap the data instance type to be created, for example, **bool**.
- 3 Tap the context menu, and select **Create New Data**.
The **Create New Data** dialog is displayed.
- 4 In the **Declaration** tab, complete the following fields for the new data type:

xx1900000351

- **Name:** Type a new name.
- **Scope:** Set the accessibility for the data instance from the following options:
 - **Global** - reachable by all the tasks.
 - **Local** - reachable within the module.
 - **Task** - reachable within the task.

Continues on next page

- **Storage type:** Set the type of memory used for the data instance from the following options:
 - **Persistent** to retain the data between sessions.
 - **Variable** if the data instance is variable.
 - **Constant** if the data instance is constant.
 - Tap the **Module** menu to select module.
 - Tap the **Routine** menu to select routine.
- 5 In the **Initial Value** tab, select the values according to the selected data type.
- 6 Tap **Apply**.

The new data instance is created.



Note

Creation of new data type is described in the manual *Operating manual - Integrator's guide OmniCore*.

6.5.3 Editing data instances

Overview

This section describes how to view data instances in the **Program Data** application. It also details how to edit, delete, change declaration of, copy, and define a data instance.

For the data types `tooldata`, `wobjdata` and `loaddata` also see sections [Tools on page 152](#), [Work objects on page 170](#) or [Payloads on page 181](#).

Viewing and editing data instances

This section details how to view the available instances of a data type.

- 1 On the start screen, tap **Program Data**.
- 2 Tap the data instance type for which the data instances need to be viewed or edited. For example, **tooldata**.

The data instances for the selected type are displayed.

- 3 Tap on the data instance that needs to be edited.
- 4 Tap the **Declaration** tab, **Initial Value**, and **Current Value** tab and edit the parameters according to your requirement.
- 5 Tap **Apply**.

The changes to the selected data instance are saved.



Note

If the value of a persistent variable is changed at any point in a running program, the **Code** will still show the old value until the program stops. The **Program Data**, however, always shows the current value of persistent variables. See *Persistent declaration* in the *Technical reference manual - RAPID Overview* for further information.

Deleting a data instance

Use the following procedure to delete a data instance.

- 1 On the start screen, tap **Program Data**.
- 2 Tap the data instance type for which you need to delete the data instance. For example, **tooldata**.

The data instances for the selected type are displayed.

- 3 Tap **Delete** on the context menu for the data instance that you want to delete.
- 4 Tap **Yes**.

The selected data instance is deleted.

Continues on next page



CAUTION

A deleted tool, work object, or payload cannot be recovered, and all related data will be lost. If the deleted tool, work object, or payload is referenced by any program, those programs need to be edited before executing.

If you delete a tool you cannot continue the program from the current position.

Copying a data instance

Use the following procedure to create the copy of a data instance.

- 1 On the start screen, tap **Program Data**.
- 2 Tap the data instance type for which you need to create a copy of the data instance. For example, **tooldata**.
- 3 Tap **Copy** on the context menu for the data instance that you want to copy.
A confirmation window is displayed.
- 4 Tap **OK**.
A copy of the selected data instance is created.



Note

The copied data instance has the same values as the original, but the name is unique.

6.6 Tools

6.6.1 What is a tool?

Tool

A tool is an object that can be mounted directly or indirectly on the robot turning disk or fitted in a fixed position within the robot working range.



Note

A fixture (jig) is not a tool.

All tools must be defined with a TCP (Tool Center Point).

Each tool that can be used by the robot must be measured and its data stored in order to achieve accurate positioning of the tool center point.



WARNING

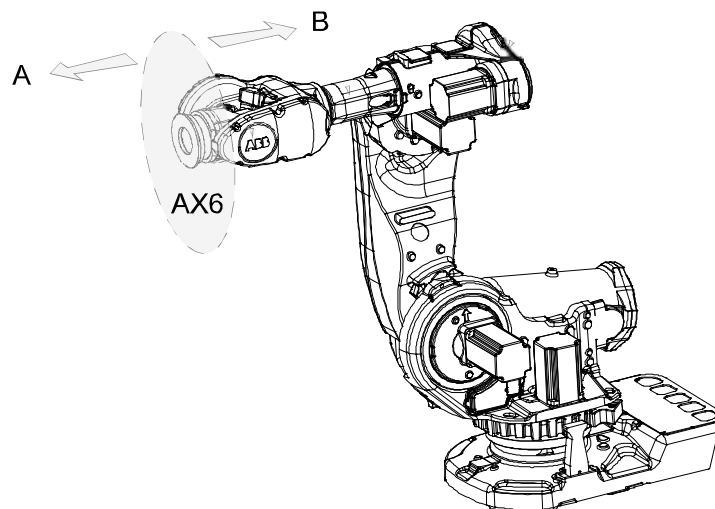
It is important to always define the actual tool load and, when used, the payload of the robot (for example, a gripped part). Incorrect definitions of load data can result in overloading of the robot mechanical structure. There is also a risk that the speed in manual reduced speed mode can be exceeded.

When incorrect load data is specified, it can often lead to the following consequences:

- The robot may not use its maximum capacity.
- Impaired path accuracy including a risk of overshooting.
- Risk of overloading the mechanical structure.

The controller continuously monitors the load and writes an event log if the load is higher than expected. This event log is saved and logged in the controller memory.

Illustration



en0400000803

| | |
|---|------------|
| A | Tool side |
| B | Robot side |

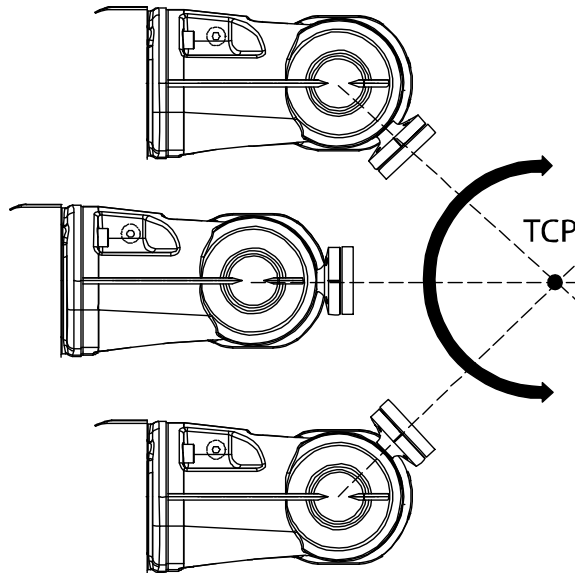
6 Programming and testing

6.6.2 What is the tool center point?

6.6.2 What is the tool center point?

Illustration

The illustration shows how the tool center point (TCP) is the point around which the orientation of the tool/manipulator wrist is being defined.



xx0300000604

Description

The tool center point (TCP) is the point in relation to which all robot positioning is defined. Usually the TCP is defined as relative to a position on the manipulator turning disk.



CAUTION

Incorrect settings for the TCP will result in incorrect speed. Always verify the speed after changing the settings.

The TCP will be jogged or moved to the programmed target position. The tool center point also constitutes the origin of the tool coordinate system.

The robot system can handle a number of TCP definitions, but only one can be active at any one time.

There are two basic types of TCPs: moveable or stationary.

Moving TCP

The vast majority of all applications deal with moving TCP, i.e. a TCP that moves in space along with the manipulator.

A typical moving TCP can be defined in relation to, for example the tip of a arc welding gun, the center of a spot welding gun, or the end of a grading tool.

Continues on next page

Stationary TCP

In some applications a stationary TCP is used, for example when a stationary spot welding gun is used. In such cases the TCP can be defined in relation to the stationary equipment instead of the moving manipulator.

6.6.3 Creating a tool

What happens when you create a tool?

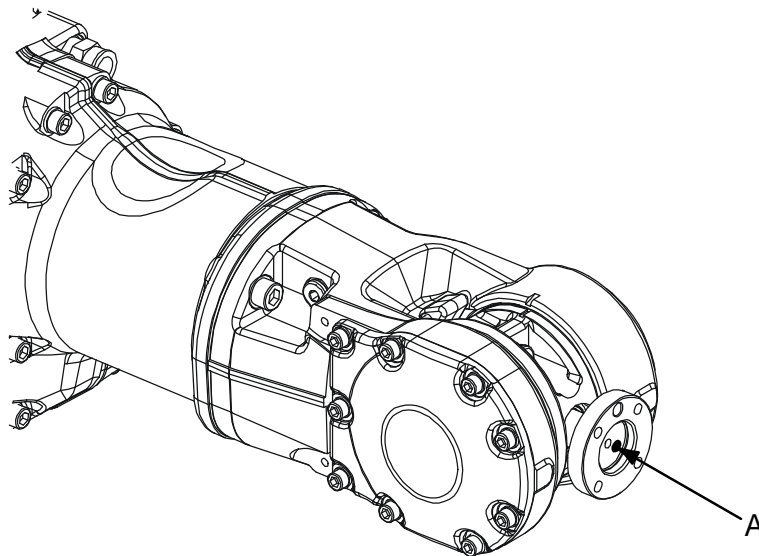
When you create a new tool a variable of the data type `tooldata` is created. The variable name will be the name of the tool. For more information on data types, see *Technical reference manual - RAPID Instructions, Functions and Data types*.

The new tool has initial default values for mass, frame, orientation etc., which must be defined before the tool can be used.

How to create a tool

The tool center point of the default tool (tool0) is in the center of the robot's mounting flange and shares the orientation of the robot base.

By creating a new tool you define another tool center point. For more information about tool center points, see [What is the tool center point? on page 154](#).



en0400000779

| | |
|---|-----------------------------------|
| A | Tool center point, TCP, for tool0 |
|---|-----------------------------------|

- 1 On the start screen, tap **Program Data**.
- 2 Select **tooldata** from the **Data Types** list.
- 3 Tap **Create New Data** in the menu to the right.
The **Create New Data** window is displayed.
- 4 Complete the tool information by typing or selecting information in the fields available under the **Declaration** and **Initial Value** tabs (see [Tool declaration settings on page 157](#)).



Note


In the **Mass** field under **Initial Value** tab, make sure to provide the mass of the attached tool in kg units.

- 5 Tap **Apply**.

Continues on next page

The tool is created.

Tool declaration settings

| If you want to change... | then... | Recommendation |
|-----------------------------------|--|--|
| the name of the tool | tap on the Name field and change name using the soft keyboard that appears. | Tools are automatically named <code>tool</code> followed by a running number, for example <code>tool110</code> or <code>tool121</code> . You are recommended to change this to something more descriptive such as gun, gripper or welder.  Note If you change the name of a tool after it is referenced in any program you must also change all occurrences of that tool. |
| the scope | select the preferred scope from the Scope drop-down list. | Tools should be global, if it should be available to all the modules in the program. |
| the storage type | select the value from the Storage Type drop-down list. | Tool variables must always be persistent. |
| the task | select the value from the Task drop-down list. | |
| the module | select the module in which this tool should be declared from the Module drop-down list. | |
| the routine | select the value from the Routine drop-down list. | |
| the size of the data array's axes | select the value from the Dimension drop-down list. | |



Note

The created tool is not useful until you define the tool data (TCP coordinates, orientation, weight, and so on). For more details, see [Editing the tool data on page 164](#) and [LoadIdentify, load identification service routine on page 203](#).

6.6.4 Copying a tool

Procedure

Use the following procedure to create the copy of a tool.

- 1 On the start screen, tap **Program Data**, and tap the **tooldata** data instance type.
The data of type tooldata are displayed.
- 2 Tap **Copy** on the context menu for the tooldata that you want to copy.
A confirmation window is displayed.
- 3 Tap **OK**.
A copy of the selected tooldata is created.



Note

The copied tooldata has the same values as the original, but the name is unique.

6.6.5 Defining the tool frame

Preparations

To define the tool frame, you first need a reference point in the world coordinate system. If you need to set the tool center point orientation, you also need to affix elongators to the tool.

You also need to decide which method to use for the tool frame definition.

Available methods

There are three different methods which can be used when defining the tool frame. All three require that you define the cartesian coordinates of the tool center point. What differs is how the orientation is defined.

| If you want to... | ...then select |
|---|-----------------------|
| set the orientation the same as the orientation of the robot's mounting plate | TCP (default orient.) |
| set the orientation in Z axis | TCP&Z |
| set the orientation in X and Z axes | TCP&Z,X |

How to define position and orientation

The following procedure describes how to select the method to be used when defining the tool frame:

- 1 On the start screen, tap **Program Data**, and then select **tooldata**.
The data of type 'tooldata' is displayed.
- 2 Tap on the context menu of the tool that you want to edit, and select **Define**.
The **Tool TCP Definition** window for the selected tool is displayed.

Continues on next page

6 Programming and testing

6.6.5 Defining the tool frame

Continued

Messages Event log 100 % 123

Tool TCP Definition

Define Position Define Orientation Result

Select number of points, modify the positions and tap next

Tool: tool1

Number of points
3

Point 1
Modified

Point 2
Not Modified

Point 3
Not Modified

| | |
|---------|---------|
| X | 0 mm |
| Y | 0 mm |
| Z | 0 mm |
| Rx | 0 deg |
| Ry | 0 deg |
| Rz | 0 deg |
| RobConf | 0,0,0,0 |

Modify > Next X Cancel

Home Operate Calibrate

xx1900000353

- 3 Select the number of approach points from the **Number of points** field. Usually 4 points are enough. If you choose more points to get a more accurate result, you should be equally careful when defining all of them.
- 4 Select a point, jog the robot to a required position, and then tap **Modify** to define the selected points. Repeat this step for all the points. See [How to proceed with tool frame definition on page 162](#).
- 5 Tap **Next**.
The Tool TCP Definition, Define Orientation window is displayed.

Continues on next page

Messages
Event log

Tool TCP Definition

Define Position
Define Orientation
Result

Select a method, modify the positions and tap next

Tool : tool1

Method
TCP & Z

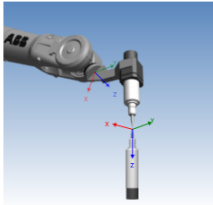
ReferencePoint
Not Modified

Elongator Z
Not Modified

Elongator X
Not Modified

Position for ReferencePoint

| | |
|---------|---------|
| X | 0 mm |
| Y | 0 mm |
| Z | 0 mm |
| Rx | 0 deg |
| Ry | 0 deg |
| Rz | 0 deg |
| RobConf | 0,0,0,0 |



Modify
Load Positions

Back
Next
Cancel

Home
Calibrate

10:06 AM

xx1900000354

- 6 Select the **Method** to be used.
- 7 Select a point and tap **Modify** to modify the positions.
- 8 Tap **Next**.
The **Tool TCP Definition, Calibration Result** window is displayed.
- 9 Tap **Finish** to save the calibration.

Continues on next page

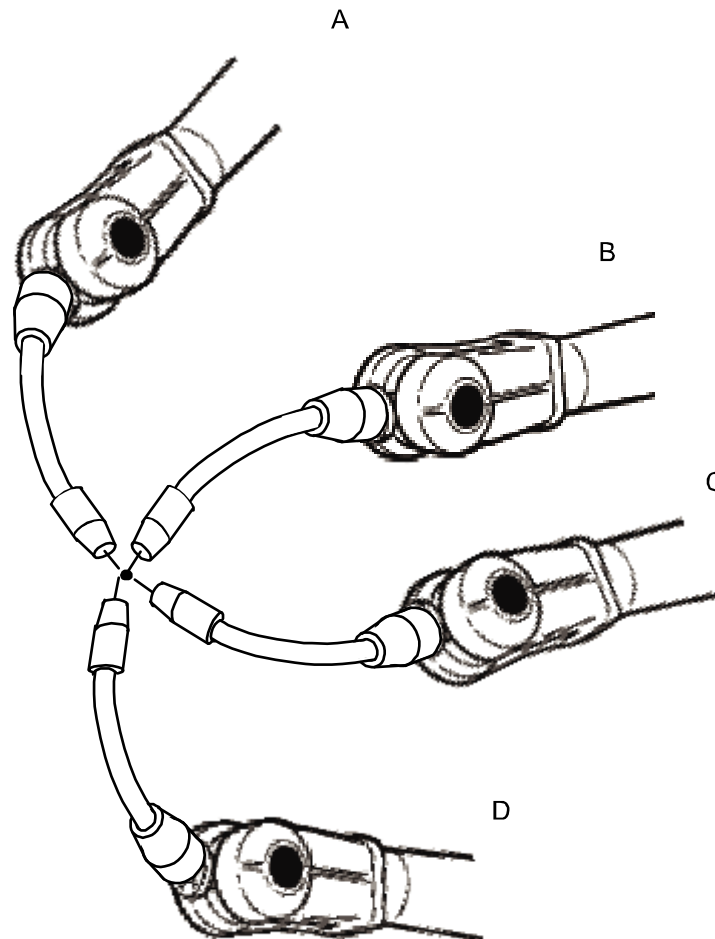
6 Programming and testing

6.6.5 Defining the tool frame

Continued

How to proceed with tool frame definition

This procedure describes how to define the tool center point in Cartesian coordinates.



en0400000906

- 1 Jog the robot to an appropriate position, A, for the first approach point. Use small increments to accurately position the tool tip as close to the reference point as possible.
- 2 Tap **Modify** to define the point.
- 3 Repeat step 1 and 2 for each approach point to be defined, positions B, C, and D.
Jog away from the fixed world point to achieve the best result. Just changing the tool orientation will not give as good a result.
- 4 If the method you are using is TCP&Z or TCP&Z,X orientation must be defined as well.
Follow the instructions in [How to define elongator points on page 163](#).
- 5 If, for some reason, you want to redo the calibration procedure described in step 1-4, tap **Cancel**.
- 6 Tap **Next**. The **Calculation Result** dialog box will now be displayed, asking you to cancel or to confirm the result before it is written to the controller.

Continues on next page

For further information see [Is the calculated result good enough? on page 163](#)

How to define elongator points

This procedure describes how to define the orientation of the tool frame by specifying the direction of the z and/or x axis. You need to do this only if you the tool orientation should differ from that of the robot base. The tool coordinate system by default resembles the coordinate system of tool0, as illustrated in [Measuring the tool center point on page 164](#).

- 1 Without changing the orientation of the tool, jog the robot so that the reference world point becomes a point on the desired positive axis of the rotated tool coordinate system.
- 2 Tap **Modify** to define the point.
- 3 Repeat step 1 and 2 for the second axis if it should be defined.

Is the calculated result good enough?

The **Calculation Result** dialog box displays the calculated result of the tool frame definition. You have to confirm that you accept the result before it can take effect in the controller. The alternative is to redo the frame definition in order to achieve a better result. The result **Mean Error** is the average distance of the approach points from the calculated TCP (tool center point). **Max Error** is the maximum error among all approach points.

It is hard to tell exactly what result is acceptable. It depends on the tool, robot type etc. you are using. Usually a mean error of a few tenths of a millimeter is a good result. If the positioning has been undertaken with reasonable accuracy the result will be okay.

As the robot is used as a measuring machine, the result is also dependent on where in the robot's working area the positioning has been done. Variation of the actual TCP up to a couple of millimeters (for large robots) can be found between definitions in different parts of the working area. The repeatability of any following TCP calibrations will thus increase if these are done close to the preceding ones. Note that the result is the optimal TCP for the robot in that working area, taking into account any discrepancies of the robot in the configuration at hand.



Tip

A common way to check that the tool frame has been correctly defined is to perform a reorientation test when the definition is ready. Select the reorient motion mode and the tool coordinate system and jog the robot. Verify that the tool tip stays very close to the selected reference point as the robot moves.

6.6.6 Editing the tool data

Tool data

Use the value settings to set the tool center point position and physical properties of the tool such as weight and center of gravity.

This can also be done automatically with the service routine `LoadIdentify`. See the sections [Running a service routine on page 195](#) and [LoadIdentify, load identification service routine on page 203](#).



CAUTION

If the tooldata is incorrectly defined there is a risk that the speed is higher than expected. This is particularly important in manual mode.

Editing the tool data

This section details how to edit the tool data.

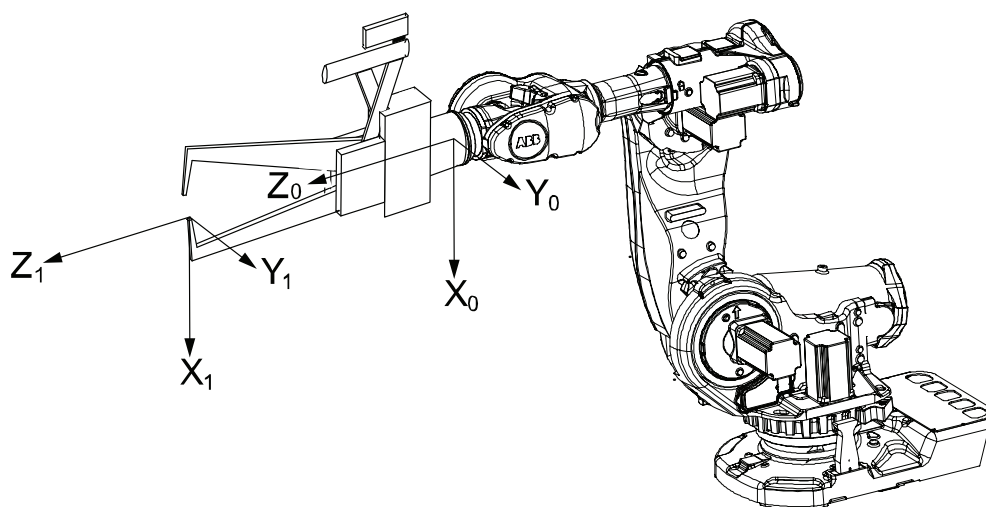
- 1 On the start screen, tap **Program Data**,
- 2 Select **tooldata** from the list of data types.
- 3 Tap on the menu button next to the tool that you want to edit.
The context menu is displayed.
- 4 Tap **Edit**.
The **Edit Data** page is displayed.
- 5 In the **Declaration**, **Initial Value**, and **Current Value** tabs edit the parameters according to your requirement.
- 6 Tap **Apply**.
The changes are saved.

Measuring the tool center point

The easiest way to define the tool center point, TCP, is usually to use the predefined method described in [Defining the tool frame on page 159](#). If you use this method, you do not have to write any values for the frame as these are supplied by the method.

If you already have the measurements of the tool, or for some reason want to measure them manually, the values can be entered in the tool data.

Continues on next page



en0400000881

| | |
|-------|--|
| X_0 | X axis for tool0 |
| Y_0 | Y axis for tool0 |
| Z_0 | Z axis for tool0 |
| X_1 | X axis for the tool you want to define |
| Y_1 | Y axis for the tool you want to define |
| Z_1 | Z axis for the tool you want to define |

- 1 Measure the distance from the center of the robot's mounting flange to the tool's center point along the X axis of tool0.
- 2 Measure the distance from the center of the robot's mounting flange to the tool's center point along the Y axis of tool0.
- 3 Measure the distance from the center of the robot's mounting flange to the tool's center point along the Z axis of tool0.

Editing the tool definition

| | Action | Instance | Unit |
|---|--|--|------|
| 1 | Enter the cartesian coordinates of the tool center point's position. | tframe.trans.x tframe.trans.y tframe.trans.z | [mm] |
| 2 | If necessary, enter the tool frame orientation. | tframe.rot.q1 tframe.rot.q2 tframe.rot.q3 tframe.rot.q4 | None |
| 3 | Enter the weight of the tool. | tload.mass | [kg] |
| 4 | If necessary, enter the tool's center of gravity. | tload.cog.x tload.cog.y tload.cog.z | [mm] |

Continues on next page

6 Programming and testing

6.6.6 Editing the tool data

Continued

| | Action | Instance | Unit |
|---|---|--|---------------------|
| 5 | If necessary, enter the orientation of the axis of moment | tload.aom.q1 tload.aom.q2 tload.aom.q3 tload.aom.q4 | None |
| 6 | If necessary, enter the tool's moment of inertia. | tload.ix tload.iy tload.iz | [kgm ²] |
| 7 | Tap Save to use the new values, Cancel to leave the definition unchanged. | | |

6.6.7 Deleting a tool

Deleting a tool

Use the following procedure to delete a tool:

- 1 On the start screen, tap **Program Data**,
- 2 Select **tooldata** from the list of data types.
- 3 Tap on the menu button next to the tool that you want to edit.
The context menu is displayed.
- 4 Tap **Delete**.
- 5 The **Delete Data** confirmation message is displayed.
- 6 Tap **Yes**.
The selected tool is deleted.



CAUTION

A deleted tool, work object, or payload cannot be recovered, and all related data will be lost. If the deleted tool, work object, or payload is referenced by any program, those programs need to be edited before executing.

If you delete a tool you cannot continue the program from the current position.

6.6.8 Setup for stationary tools

Stationary tools

Stationary tools are used, for instance, in applications that involve large machines such as cutters, presses and punch cutters. You may use stationary tools to perform any operation that would be difficult or inconvenient to perform with the tool on the robot.

With stationary tools, the robot holds the work object.

Make a tool stationary

This section describes how to make a tool stationary.

- 1 On the start screen, tap **Program Data**,
- 2 Select **tooldata** from the list of data types.
- 3 Tap on the menu button next to the tool that you want to edit.
The context menu is displayed.
- 4 Tap **Edit**.
The **Edit Data** page is displayed.
- 5 Tap the **Current Value** tab.
The data that defines the selected tool is displayed.
- 6 In the **robhold** field set the value to **FALSE**.
- 7 Tap **Apply**.
The selected tool is made stationary.

Make a work object robot held

This section describes how to make a work object robot held.

- 1 On the start screen, tap **Program Data**,
- 2 Select **wobjdata** from the list of data types.
- 3 Tap on the menu button next to the workobject that you want to edit.
The context menu is displayed.
- 4 Tap **Edit**.
The **Edit Data** page is displayed.
- 5 Tap the **Current Value** tab.
The data that defines the selected tool is displayed.
- 6 In the **robhold** field set the value to **TRUE**.
- 7 Tap **Apply**.
The changes are saved.

Set up the tool coordinate system

You use the same measurement methods to set up a stationary tool coordinate system as with tools mounted on the robot.

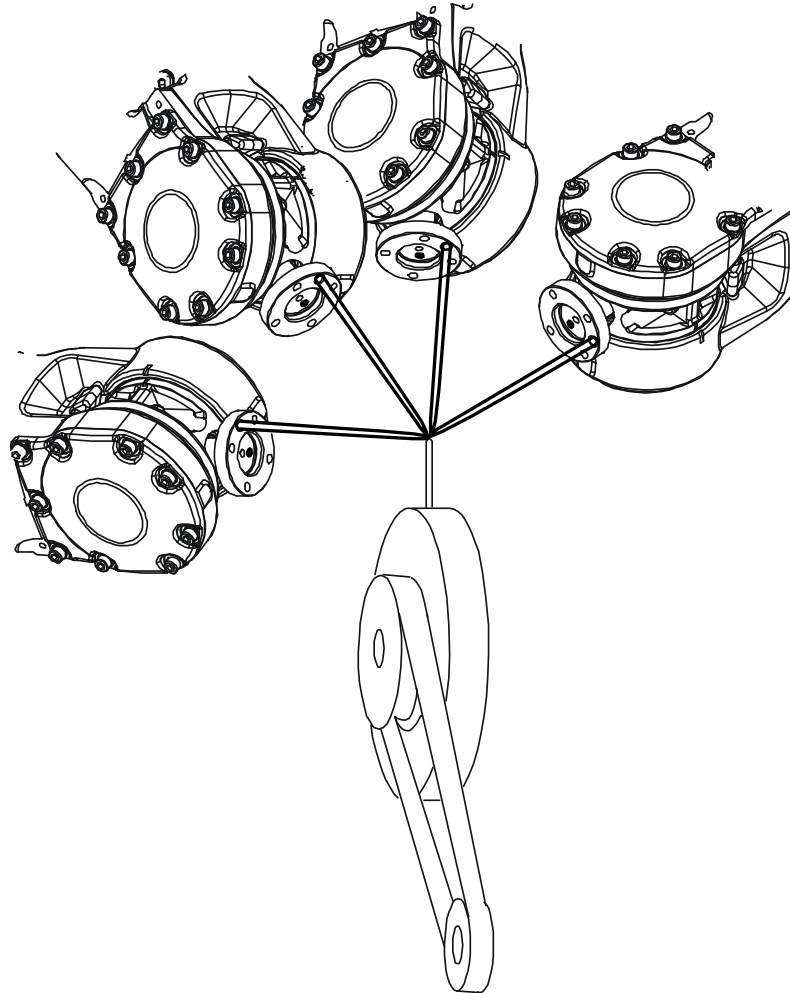
The world reference tip must, in this case, be attached to the robot. Define and use a tool with the reference tip's measurements when you create approach points.

Continues on next page

You also need to attach elongators to the stationary tool if you need to set up the orientation.

You should enter the reference tip's tool definition manually to minimize errors when calculating the stationary tool's coordinate system.

You may enter the stationary tool's definition manually.



en0400000990

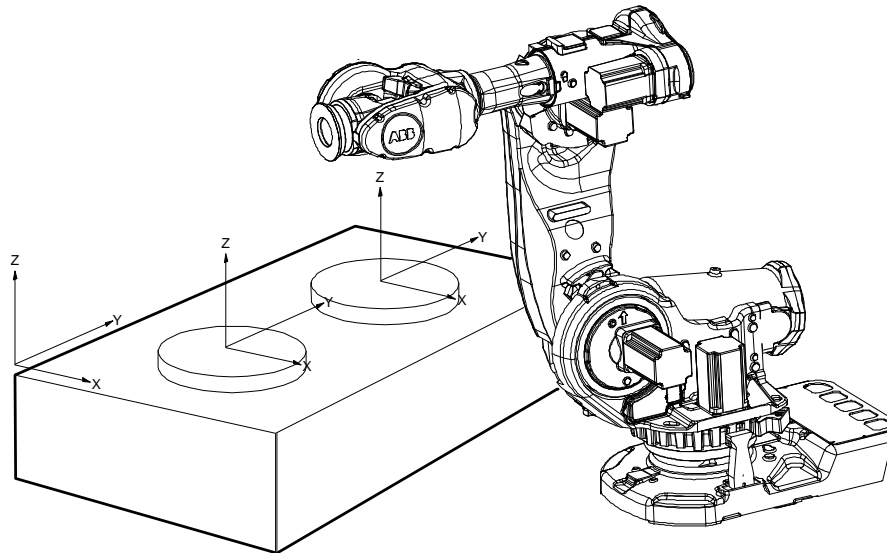
6 Programming and testing

6.7.1 What is a work object?

6.7 Work objects

6.7.1 What is a work object?

Illustration



en0400000819

Description

A work object is a coordinate system with specific properties attached to it. It is mainly used to simplify programming when editing programs due to displacements of specific tasks, objects processes etc.

The work object coordinate system must be defined in two frames, the user frame (related to the world frame) and the object frame (related to the user frame).

Work objects are often created to simplify jogging along the object's surfaces.

There might be several different work objects created so you must choose which one to use for jogging.

Payloads are important when working with grippers. In order to position and manipulate an object as accurate as possible its weight must be accounted for. You must choose which one to use for jogging.

6.7.2 Creating a work object

What happens when I create a work object?

A variable of the type `wobjdata` is created. The variable's name will be the name of the work object. For more information on data types, see *Technical reference manual - RAPID Instructions, Functions and Data types*.

This is detailed in section [What is a work object? on page 170](#).

Creating a work object

The work object's coordinate system is now identical with the world coordinate system.

To define the position and orientation of the work object's coordinate system, see [Work object declaration settings on page 171](#).

- 1 On the start screen, tap **Program Data**.
- 2 Select **wobjdata** from the menu.
The list of data of type `wobjdata` is displayed.
- 3 Tap **Create New Data** in the menu to the right.
- 4 In the **Declaration**, **Initial Value** tabs select the parameters according to your requirement.
- 5 Tap **Apply**
The work object is created.

Work object declaration settings

| If you want to change... | then... | Recommendation |
|--------------------------|--|--|
| the work object's name | enter a name in Name | Work objects are automatically named <code>wobj</code> followed by a running number, for example <code>wobj10</code> , <code>wobj27</code> . You should change this to something more descriptive. If you change the name of a work object after it is referenced in any program you must also change all occurrences of that work object. |
| the scope | select the scope of choice from the menu | Work objects should always be global to be available to all modules in the program. |
| the storage type | - | Work object variables must always be persistent. |
| the task | select the preferred task from the menu | |
| the module | select the module in which this work object should be declared from the menu | |

6.7.3 Copying a workobject

Procedure

Use the following procedure to create the copy of a workobject:

- 1 On the start screen, tap **Program Data**, and tap the **wobjdata** data instance type.
The data of type wobjdata are displayed.
- 2 Tap **Copy** on the context menu for the workobject that you want to copy.
A confirmation window is displayed.
- 3 Tap **OK**.
A copy of the selected wobjdata is created.



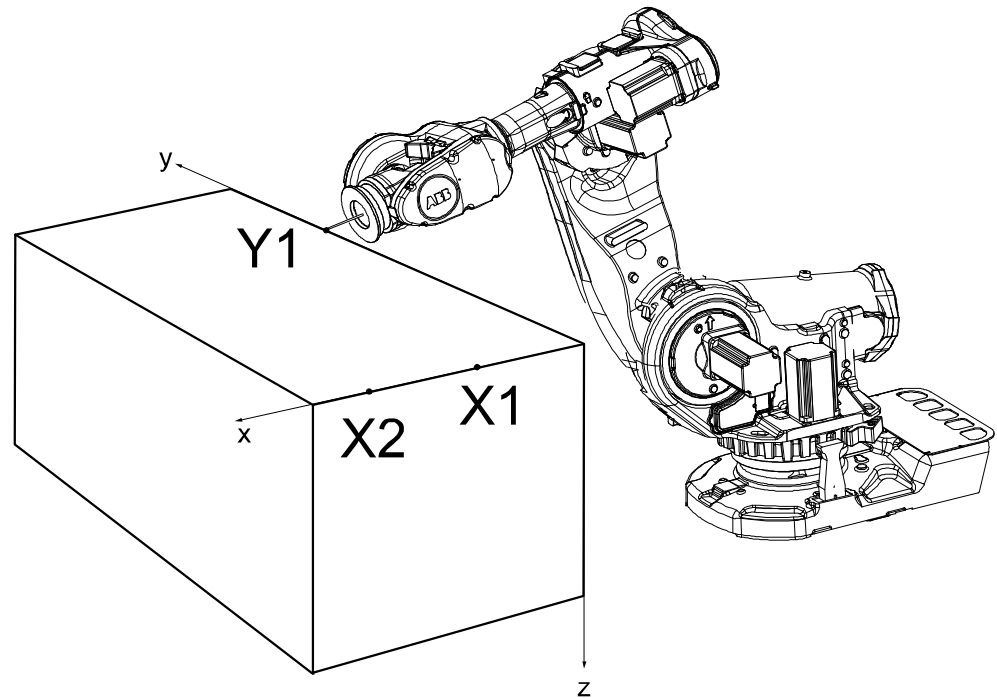
Note

The copied wobjdata has the same values as the original, but the name is unique.

6.7.4 Defining a work object

Overview

Defining a workobject means that the robot is used to point out the location of it. This is done by defining three positions, two on the x-axis and one on the y-axis. When defining a workobject you can use either the user frame or the object frame or both. The user select frame and the object frame usually coincides. If not, the object frame is displaced from the user frame.



en0400000887

Defining a work object

Use the following procedure to define a work object:

- 1 On the start screen, tap **Program Data**, and tap the **wobjdata** from the list of data types.
The data of type **wobjdata** are displayed.
- 2 Tap on the context menu next to the workobject that you want to define and select **Define**.
The **Workobject Definition** page is displayed. The **Define User frame** tab is displayed by default.
- 3 In the **User method** list select the method **User defined with 3 points**.
- 4 Select each point, jog the robot to the approach point, and tap **Modify**.
The message **Modified** is displayed for the selected point.
Repeat this step for each point.
- 5 Tap **Next**.
The **Define Object frame** tab is displayed.

Continues on next page

- 6 In the **Object Method** list select the **User defined with 3 points** method.
- 7 Select each point, jog the robot to the approach point, and tap **Modify**.
The message **Modified** is displayed for the selected point.
Repeat this step for each point.
- 8 Tap **Next**.
The **Results** tab is displayed.
- 9 Verify the calculation result. If any change is required tap the **Back** button and redefine the parameters.
- 10 Tap **Finish**.

6.7.5 Defining the work object coordinate system

Overview

Defining a work object means that the robot is used to point out the location of it. This is done by defining three positions, two on the x-axis and one on the y-axis. When defining a work object you can use either the user frame or the object frame or both. The user select frame and the object frame usually coincides. If not, the object frame is displaced from the user frame.

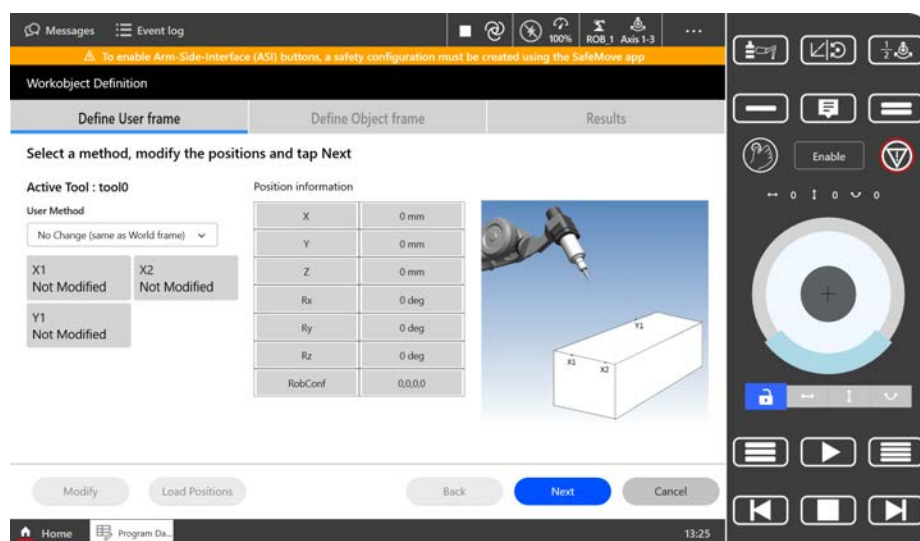
How to define the work object coordinate system

This procedure describes how to define the work object coordinate system. Note that this only works for a user created work object, not the default work object, wobj0.

- 1 On the start screen, tap **Program Data**, and then select **Workobject** from the list of data types.
- 2 Tap on the context menu next to the workobject that you want to define and select **Define**.

The **Workobject Definition** page is displayed. The **Define User frame** tab is displayed by default.

- 3 Select method from the **User method** drop down menu.



xx1900000451

- 4 Tap **Modify** to define the points. See [How to define the user frame on page 176](#).
- 5 Tap **Next**.
The **Workobject Definition, Define Object frame** window is displayed.
- 6 Select the **Object Methods** to be used.
- 7 Tap **Modify** to modify the positions. See [How to define the object frame on page 177](#).
- 8 Tap **Next**.
The **Workobject Definition, Calculation Result** window is displayed.

Continues on next page

6 Programming and testing

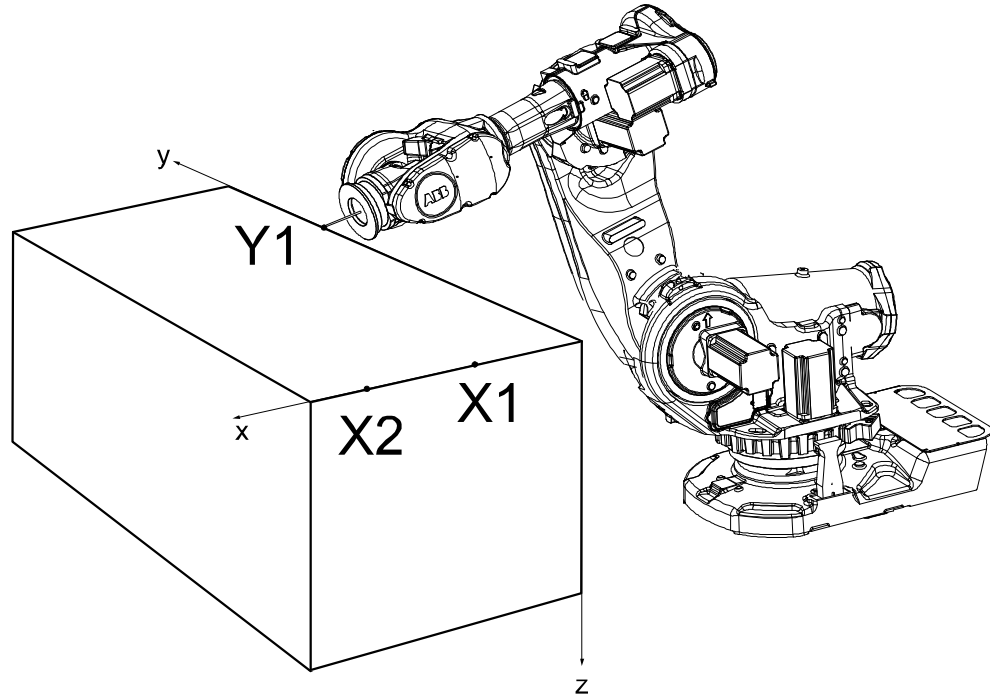
6.7.5 Defining the work object coordinate system

Continued

9 Tap **Finish** to save the calibration.

How to define the user frame

This section details how to define the user frame.



en0400000887

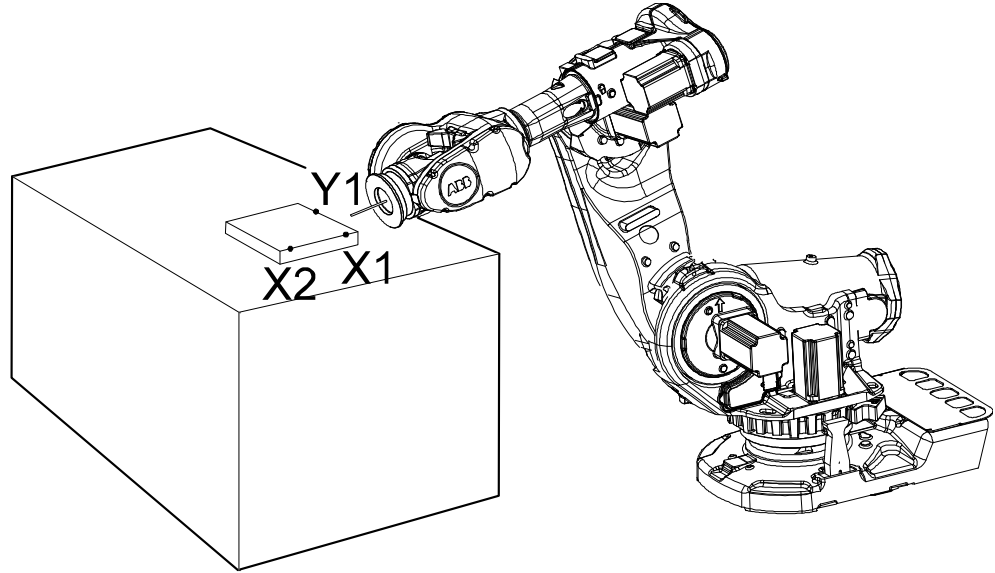
The x axis will go through points X1-X2, and the y axis through Y1.

- 1 In the **User method** drop down menu, select **User defined with 3 points**.
- 2 Press the three-position enabling device and jog the robot to the first (X1, X2 or Y1) point that you want to define.
Large distance between X1 and X2 is preferable for a more precise definition.
- 3 Select the point in the list.
- 4 Tap **Modify** to define the point.
- 5 Repeat steps 2 to 4 for the remaining points.

Continues on next page

How to define the object frame

This section describes how to define the object frame if you want to displace it from the user frame.



en0400000899

The x axis will go through points X1-X2, and the y axis through Y1.

- 1 In the **Object Methods** drop down menu, select **User defined with 3 points**.
- 2 See steps 2 to 4 in the description of [How to define the user frame on page 176](#).

6 Programming and testing

6.7.6 Editing the work object data

6.7.6 Editing the work object data

Overview

Use the work object data definition to set the position and rotation of the user and object frames.

How to display the work object data

- 1 On the start screen, tap **Program Data**, and then select **Workobject** from the list of data types.
- 2 Tap on the context menu next to the workobject that you want to edit and select **Edit**.
The **Edit Data** page is displayed.
- 3 Tap the **Declaration**, **Initial Value**, and **Current Value** tabs.
- 4 View the values according to your requirement.

How to set user and object frame values manually

The easiest way to set the work object and user coordinate systems position is to use the method described in [Defining the work object coordinate system on page 175](#). You can however edit the values manually using the following guide.

| Values | Instance | Unit |
|---|--|------|
| The cartesian coordinates of the position of the object frame | oframe.trans.x oframe.trans.y oframe.trans.z | mm |
| The object frame orientation | oframe.rot.q1 oframe.rot.q2 oframe.rot.q3 oframe.rot.q4 | - |
| The cartesian coordinates of the position of the user frame | uframe.trans.x uframe.trans.y uframe.trans.z | mm |
| The user frame orientation | uframe.rot.q1 uframe.rot.q2 uframe.rot.q3 uframe.rot.q4 | - |



Note

Editing work object data can also be done from the **Code** window.

6.7.7 Deleting a work object

Deleting a work object

Use the following procedure to delete a work object:

- 1 On the start screen, tap **Program Data**, and then select **Workobject** from the list of data types.

The **Data of type 'wobjdata'** page displays the available work objects.

- 2 Tap on the context menu next to the workobject that you want to delete and select **Delete**.

The **Delete Data** confirmation window is displayed.

- 3 Tap **Yes**.

The selected work object is deleted.

6.7.8 Setup stationary work object

Make a work object robot held

Use the following procedure to make a workobject stationery:

- 1 On the start screen, tap **Program Data**, and then select **Workobject** from the list of data types.

The **Data of type 'wobjdata'** page displays the list of available work objects.

- 2 Tap on the context menu next to the workobject that you want to make stationery and select **Edit**.

The **Edit Data** page is displayed.

- 3 Tap the **Current Value** tab.

- 4 Set the value in **robhold** field to **TRUE**.

- 5 Tap **Apply**

The changes are saved.

6.8 Payloads

6.8.1 Overview

Description

Payloads are important when working with grippers. In order to position and manipulate an object as accurate as possible its weight must be accounted for. You must choose which one to use for jogging.

6.8.2 Creating a payload

What happens when I create a payload?

When you create a payload, a variable of the type `loaddata` is created. The variable's name will be the name of the payload. For more information on data types, see *Technical reference manual - RAPID Instructions, Functions and Data types*.

Adding a new payload and setting data declaration

The payload's coordinate system will be set to the position, including orientation, of the world coordinate system.

- 1 On the start screen, tap **Program Data**, and then select **loaddata**.
The **loaddata** items are displayed.
- 2 Tap **Create New Data** in the menu to the right.
The **Create New Data** window is displayed.
- 3 Complete the payload information (see [Payload declaration settings on page 183](#)).
- 4 Tap **Apply**.
The payload is created.



WARNING

It is important to always define the actual tool load and, when used, the payload of the robot (for example, a gripped part). Incorrect definitions of load data can result in overloading of the robot mechanical structure. There is also a risk that the speed in manual reduced speed mode can be exceeded.

When incorrect load data is specified, it can often lead to the following consequences:

- The robot may not use its maximum capacity.
- Impaired path accuracy including a risk of overshooting.
- Risk of overloading the mechanical structure.

The controller continuously monitors the load and writes an event log if the load is higher than expected. This event log is saved and logged in the controller memory.

Payload declaration settings

| If you want to change... | ...then... | Recommendation |
|--------------------------|--|---|
| the payload's name | enter a name in Name | Payloads are automatically named <code>load</code> followed by a running number, for example <code>load10</code> , <code>load31</code> . You should change this to something more descriptive. If you change the name of a payload after it is referenced in any program you must also change all occurrences of that payload's name. |
| the scope | select the scope of choice from the menu | Payloads should always be global to be available to all modules in the program. |
| the storage type | - | Payload variables must always be persistent. |
| the task | select the preferred task from the menu | |
| the module | select the module in which this payload should be declared from the menu | - |

Setting the value for **ModalPayloadMode**

The **ModalPayloadMode** is defined in RobotStudio. See *Operating manual - RobotStudio*.

6.8.3 Copying a payload

Procedure

Use the following procedure to create the copy of a workobject:

- 1 On the start screen, tap **Program Data**, and tap the **loaddata** data instance type.
The data of type **loaddata** are displayed.
- 2 Tap **Copy** on the context menu for the **loaddata** that you want to copy.
A confirmation window is displayed.
- 3 Tap **OK**.
A copy of the selected **loaddata** is created.



Note

The copied **loaddata** has the same values as the original, but the name is unique.

6.8.4 Editing the payload data

Overview

Use the payload data to set physical properties of the payload such as weight and center of gravity.

This can also be done automatically with the service routine `LoadIdentify`. See the sections [Running a service routine on page 195](#) and [LoadIdentify, load identification service routine on page 203](#).

Displaying the payload definition

- 1 On the start screen, tap **Program Data**, and then select **loaddata** from the list of data types.
The **Data of type 'loaddata'** page displays the list of available loaddata.
- 2 Tap on the context menu next to the loaddata that you want to edit and select **Edit**.
- 3 Tap the payload for which you want to view the values.
The **Edit Data** window is displayed.
- 4 Tap the **Current Value** tab and view the values.

Changing the payload data

This procedure describes how to manually enter the payload data. This can also be done automatically by running the service routine `LoadIdentify`.

| | Action | Instance | Unit |
|---|--|--|---------------------|
| 1 | Enter the weight of the payload. | load.mass | [kg] |
| 2 | Enter the payload's center of gravity. | load.cog.x load.cog.y load.cog.z | [mm] |
| 3 | Enter the orientation of the axis of moment. | load.aom.q1 load.aom.q2 load.aom.q3 load.aom.q3 | |
| 4 | Enter the payload's moment of inertia. | ix iy iz | [kgm ²] |
| 5 | Tap Apply to use the new values, Cancel to leave the data unchanged. | - | - |

Using the `PayLoadsInWristCoords` parameter

By using the `PayLoadsInWristCoords` parameter, the loaddata for payloads can be specified relative to the wrist instead of the active TCP or work object. This can be useful if several tool or TCP or work objects (when tool is stationary) are used for one payload. In this case only one load identification is needed instead of one for each tool or TCP or work object. Thus it is possible to use the same payload loaddata for any robhold or stationary tool being active. This saves the time (for example, during commissioning).

Continues on next page

6 Programming and testing

6.8.4 Editing the payload data

Continued

For more information about `PayLoadsInWristCoords`, see *Technical reference manual - System parameters* and *Technical reference manual - RAPID Instructions, Functions and Data types*.

6.8.5 Deleting a payload

Deleting a payload

Use the following procedure to delete a payload:

- 1 On the start screen, tap **Program Data**, and then select **loaddata** from the list of data types.

The **Data of type 'loaddata'** page displays the list of available loaddata.

- 2 Tap on the context menu next to the loaddata that you want to delete and select **Delete**.

- 3 Tap **Delete** on the context menu for the payload that you want to delete.
The **Delete Data** confirmation window is displayed.

- 4 Tap **Yes**.

The selected payload is deleted.



CAUTION

A deleted tool, work object, or payload cannot be recovered, and all related data will be lost. If the deleted tool, work object, or payload is referenced by any program, those programs need to be edited before executing.

If you delete a tool you cannot continue the program from the current position.

6.9 Testing

6.9.1 Using the hold-to-run function

When to use the hold-to-run function

The hold-to-run function is used to run or step programs in manual full speed operating mode, in combination with the thumb button and the three-position enabling device. The location of the thumb button is illustrated in the section, FlexPendant [Main parts on page 20](#).

To run a program in manual full speed mode it is necessary, for safety reasons, to keep pressing both the three-position enabling device and the thumb button. This also applies when stepping through a program in manual full speed mode.



Note

For robots in collaborative applications, the motors are on by default. There is no need to press the three-position enabling device. Other safety measures apply if SafeMove is selected in the system.

The following table provides the relation between the controller operating mode and the hold-to-run function.

| Operating mode | Function |
|---------------------------|---|
| Manual full speed mode | Pressing the three-position enabling device the thumb button enables running a program. It may run continuously or step-by-step. Releasing the thumb button in this mode immediately stops the manipulator movement as well as program execution. When pressing it again, the execution is resumed from that position. |
| Manual reduced speed mode | Normally, hold-to-run has no effect in the manual reduced speed mode. However, it is possible to activate for manual reduced speed mode by changing a system parameter. |
| Automatic mode | Hold-to-run is not used in automatic mode. |

Using the hold-to-run function

Use the following procedure to use the hold-to-run function in manual full speed mode.

- 1 Open the program and set the program pointer to main.
- 2 Press the three-position enabling device to center-position.
- 3 Choose the execution mode by pressing one of the following buttons:
 - **Start** (for continuous program execution)
 - **Forward** (for step-by step program execution forwards)
 - **Backward** (for step-by step program execution backwards)
- 4 Press and hold the thumb button.

If **Start** was pressed, the program execution continues as long as you hold the thumb button.

Continues on next page

If **Forward** or **Backward** was pressed, the program is executed step-by-step by alternately releasing and pressing the **Forward/Backward** button.



Note

The thumb button must be pressed and held to continue running the program. If the button is released, the program execution will stop immediately.

- 5 If the three-position enabling device is released, intentionally or by accident, the procedure must be repeated to enable running.

6 Programming and testing

6.9.2 Running the program from a specific instruction

6.9.2 Running the program from a specific instruction

Overview

When starting a program the execution starts from the program pointer. To start from another instruction, move the program pointer to the cursor.



WARNING

When execution is started the robot will move to the first programmed position in the program. Make sure that the robot with TCP does not risk running into any obstacles.

Running the program from a specific instruction

- 1 Open **Code** and open the program.
- 2 Tap on the program step where you want to start, then tap **Debug** and then **PP to Cursor**.



DANGER

Make sure that no personnel are in the safeguarded space.

Before running the robot, read the safety information in the product manual for the controller.

- 3 Press the **Start** button on the FlexPendant.

6.9.3 Running a specific routine

Overview

When starting a program the execution starts from the program pointer. To start from another routine, move the program pointer to the routine.

Prerequisites

In order to run a specific routine the module with the routine must be loaded and the controller must be in manual stopped mode.

Running a specific routine

This procedure describes how to run a specific routine by moving the program pointer.

- 1 On the start screen, tap **Code**, and then select **Program Editor** from the menu.
- 2 Tap **Debug** and then **PP to Routine** to place the program pointer at the start of the routine.
- 3 Press the **Start** button on the FlexPendant.

6 Programming and testing

6.9.4 Stepping instruction by instruction

6.9.4 Stepping instruction by instruction

Overview

In all operating modes the program may be executed step by step forwards or backwards.

Stepping backwards is limited, see *Technical reference manual - RAPID Overview* for more details.

Select step mode

This section details how to select step mode. Stepping can be done in three ways; step in, step over, and motion step.

| | Action |
|---|--|
| 1 | Select step mode using the Quickset menu. For more details, see Execution Settings on page 40 . |

Stepping

This section details how to step forwards and backwards.

| If you want to step... | then press... |
|------------------------|--------------------------------|
| forward | Forward button on FlexPendant |
| backward | Backward button on FlexPendant |

Limitations of backward execution

There are some restrictions for the backward execution:

- When stepping backwards through a `MoveC` instruction, the execution does not stop in the circular point.
- It is not possible to step backwards out of a `IF`, `FOR`, `WHILE` and `TEST` statement.
- It is not possible to step backwards out of a routine when reaching the beginning of the routine.
- There are instructions affecting the motion that cannot be executed backwards (e.g. `ActUnit`, `ConfL` and `PDisPOn`). If attempting to execute these backwards, an alert box will inform you that this is not possible.

Backward execution behavior

When stepping forward through the program code, a program pointer indicates the next instruction to execute and a motion pointer indicates the move instruction that the robot is performing.

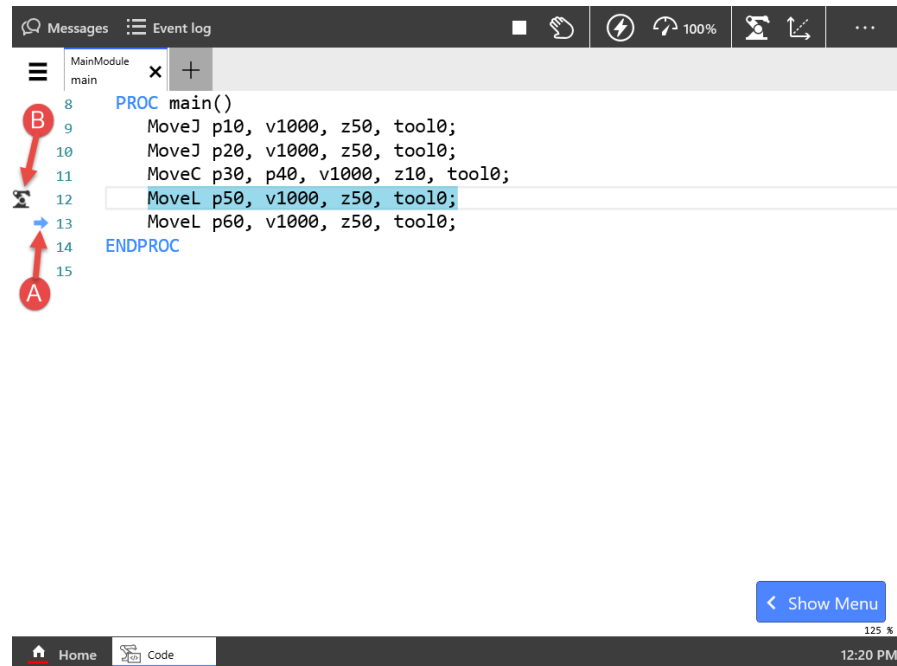
When stepping backward through the program code, the program pointer indicates the instruction above the motion pointer. When the program pointer indicates one move instruction and the motion pointer indicates another, the next backward movement will move to the target indicated by the program pointer, using the type of movement and speed indicated by the motion pointer.

Continues on next page

Example of backward execution

The following example illustrates the behavior when stepping backwards through move instructions. The program pointer and motion pointer helps you keep track of where the RAPID execution is and where the robot is.

MoveL, MoveJ, and MoveC are move instructions in RAPID, see *Technical reference manual - RAPID Instructions, Functions and Data types*.



xx1900000881

| | |
|---|-----------------|
| A | Program pointer |
| B | Motion pointer |

| When... | then... |
|--|---|
| stepping forward until the robot is in p50 | the motion pointer will indicate p50 and the program pointer will indicate the next move instruction (MoveL p60). |
| pressing the Backward button once | the robot will not move but the program pointer will move to the previous instruction (MoveC p30, p40). This indicates that this is the instruction that will be executed the next time Backward is pressed. |
| pressing the Backward button again | the robot will move to p40 linearly with the speed v1000. The target for this movement (p40) is taken from the MoveC instruction. The type of movement (linear) and the speed are taken from the instruction below (MoveL p50). The motion pointer will indicate p40 and the program pointer will move up to MoveL p20. |
| pressing the Backward button again | the robot will move circularly, through p30, to p20 with the speed v1000. The target p20 is taken from the instruction MoveL p20. The type of movement (circular), the circular point (p30) and the speed are taken from the MoveC instruction. The motion pointer will indicate p40 and the program pointer will move up to MoveL p10. |

Continues on next page

6 Programming and testing

6.9.4 Stepping instruction by instruction

Continued

| When... | then... |
|---|--|
| pressing the Backward button again | the robot will move linearly to p10 with the speed v1000. The motion pointer will indicate p10 and the program pointer will move up to <code>MoveJ p10</code> . |
| pressing the Forward button once | the robot will not move but the program pointer will move to the next instruction (<code>MoveL p20</code>). |
| pressing the Forward button again | the robot will move to p20 with the speed v1000. |

6.10 Service routines

6.10.1 Running a service routine

Service routines

Service routines perform a number of common services. The service routines available to you depends on your system setup and available options.

The service routines described in this manual are some of the commonly available.

Prerequisites

Service routines can only be started in manual reduced speed mode or in manual full speed mode.

The program must be stopped and there has to be a program pointer.

If the service routine contains parts that must be carried out in automatic mode, then the program pointer must not be moved manually before starting the service routine. The program pointer should be where the program flow was stopped.



WARNING

If a service routine is started in the middle of a stopped movement instruction (that is, before the end position is reached), then the movement will be resumed when the execution of the service routine starts.



CAUTION

Note that once a service routine has started, aborting it might not resume the system to its previous state, as the routine may have moved the robot arm.

Running a service routine

Use the following procedure to execute a service routine:



Note

Before running the service routine change the operating mode to **Manual** (manual reduced speed) mode or **Man FS** (manual full speed) mode.

- 1 On the start screen, tap **Operate**, and then select **Service Routines** from the menu.



Note

You can also find calibration related service routines in the **Calibrate** menu.

- 2 The available service routines are displayed.
- 3 Select the routine according to your requirement.
- 4 In the confirmation window tap **Yes**.

Continues on next page

6 Programming and testing

6.10.1 Running a service routine

Continued

- 5 Press the three-position enabling device to bring the controller to **Motors On** state.



Note

For YuMi robots you do not need to press the three-position enabling device since the motors are ON by default.

- 6 Press the **Play** button on the Control Panel.



Note

You can also press the **START** hard button on the FlexPendant.

- 7 The service routine is executed.



Note

After the execution of the routine, the program pointer is returned to where it was before.



CAUTION

Press **Cancel Call Rout** if you need to interrupt the routine before it has finished executing. Before resuming normal program flow, however, you must see to it that the robot is correctly positioned. If the interrupted routine has moved it, you will need to take actions to return the robot to its position. See [Returning the robot to the path on page 237](#) for further information.



WARNING

Do not execute a service routine in the middle of a move or a weld.

If you execute a service routine in the middle of a movement, the unfinished movements will be completed before the called routine is executed. This can result in an unwanted movement.

If possible, step and complete the interrupted movement before the service routine is called. Otherwise save the current movement by adding `StorePath` and `RestoPath` in the service routine. The movement will then be completed after the service routine has ended and the program starts again.

However, it is not possible to save more than one interrupted movement each time as wanted, if the service routine would be called from an error handler with `StorePath` and `RestoPath`.

Continues on next page

Limitations

Besides service routines, **Call Routine** applies to all routines with the following criteria:

- Must be a procedure with empty parameter list. This means not a function and not a trap routine.
- Must be in the task scope, not local. If the procedure is local in a module the scope is restricted to that module, and the procedure is not visible from the task level.
- Must be in a loaded module, not installed. (Check the system parameter **Installed** in the type *Automatic Loading of Modules* in the *Controller* topic.)

Related information

[Battery shutdown service routine on page 199.](#)

[LoadIdentify, load identification service routine on page 203.](#)

[Service Information System, ServiceInfo service routine on page 202.](#)

For more information about `StorePath` and `RestoPath`, see *Technical reference manual - RAPID Instructions, Functions, and Data types*.

6.10.2 Connected Services Reset service routine

Overview

It is possible to reset the software agent for Connected Services. When you reset, the software agent erases all its internal information including the registration information, the data collector script, and all the locally stored service information. The configuration will not be reset, but a new registration is required to reactivate the connected services.

Procedure

Use the following procedure to reset the software agent using FlexPendant:

- 1 On the start screen, tap **Operate**, and then select **Service Routines** from the menu.
- 2 Tap **Connected Services Reset**.
The **ConnectedServicesReset** window is displayed.
- 3 Tap **Yes**.
- 4 Press the **START** button on the FlexPendant.
A confirmation page is displayed with operator messages.
- 5 Tap **Reset**.
The software agent is reset.

6.10.3 Battery shutdown service routine

When to use this service routine

For SMB units with 2-pole battery contact, it is possible to shut down the battery backup of the serial measurement board to save battery power during transportation or storage. This is the *BatteryShutDown* service routine.

For SMB units with 3-pole contact, this function shall not be used since the power consumption is so low that it is not needed.



Tip

The service routine was earlier called *Bat_Shutdown*.

BatteryShutDown

When the system is powered on again, the function is reset. The revolution counters will be lost and need an update but the calibration values will remain.

The consumption in ordinary shutdowns is then approximately 1 mA. When using sleep mode the consumption is reduced to 0.3 mA. When the battery is nearly discharged, with less than 3 Ah left, an alert is given on the FlexPendant and the battery should be replaced.



Tip

Before starting the service routine *BatteryShutDown*, run the robot to its calibration position. This will make it easier to recover after the sleep mode.

Related information

How to start a service routine is described in [Running a service routine on page 195](#).

6.10.4 Axis Calibration service routine

Description of the service routine

Axis Calibration is a standard calibration method for calibration of ABB robots and is the most accurate method for the standard calibration.

The following routines are available for the Axis Calibration method:

- Fine calibration
- Update revolution counters
- Reference calibration

The calibration equipment for Axis Calibration is delivered as a toolkit.

The actual instructions of how to perform the calibration procedure and what to do at each step is given on the FlexPendant. You will be guided through the calibration procedure, step by step.

Related information

[*Running a service routine on page 195.*](#)

A more detailed description of the calibration method is given in the product manual for the respective manipulator.

6.10.5 Calibration Pendulum, CalPendulum service routine

Description of the service routine

For OmniCore, Calibration Pendulum is a standard calibration method for calibration of IRB 1510ID, IRB 1520ID, IRB 2400, and IRB 4400. No other robots can use this calibration method on OmniCore.

For OmniCore, the Calibration Pendulum service routine is started from the **Calibrate** app. See the product manual for the manipulator.

Two different routines are available for the Calibration Pendulum method:

- Calibration Pendulum II
- Reference calibration

The calibration equipment for Calibration Pendulum is delivered as a complete toolkit, including the *Operating manual - Calibration Pendulum*, which describes the method and the different routines further.

Related information

Calibration Pendulum is described in full in the manual *Operating manual - Calibration Pendulum*. Specific information for each robot is described in the robot's product manual.

6 Programming and testing

6.10.6 Service Information System, ServiceInfo service routine

6.10.6 Service Information System, ServiceInfo service routine

When to use this service routine

ServiceInfo is a service routine based on Service Information System, SIS, a software function which simplifies maintenance of the robot system. It supervises the operating time and mode of the robot, and alerts the operator when a maintenance activity is scheduled.

ServiceInfo

Maintenance is scheduled by setting the system parameters of the type *SIS Parameters*. All system parameters are described in *Technical reference manual - System parameters*.

More details about SIS is described in *Operating manual - Integrator's guide OmniCore*.

Supervised functions

The following counters are available:

- Calendar time counter
- Operation time counter
- Gearbox operation time counters
- Moved distance counter¹

¹ The moved distance value cannot be reset through the service routines.

Counters are reset when maintenance has been performed.



CAUTION

Resetting counters cannot be undone.

The counter status is displayed after running the ServiceInfo routine for maintenance. Status **OK** indicates that no service interval limit has been exceeded by that counter. Status **NOK** indicates that service interval limit has been exceeded by that counter.

Related information

[Running a service routine on page 195.](#)

6.10.7 LoadIdentify, load identification service routine

When to use this service routine

The service routine LoadIdentify is used to automatically identify the data of loads mounted on the robot. The data can also be entered manually, but this requires information that may be difficult to calculate.

To run LoadIdentify, there are a number of things to consider. These are described on the following pages. There is also information on error handling and limitations described in this chapter.



WARNING

It is important to always define the actual tool load and, when used, the payload of the robot (for example, a gripped part). Incorrect definitions of load data can result in overloading of the robot mechanical structure. There is also a risk that the speed in manual reduced speed mode can be exceeded.

When incorrect load data is specified, it can often lead to the following consequences:

- The robot may not use its maximum capacity.
- Impaired path accuracy including a risk of overshooting.
- Risk of overloading the mechanical structure.

The controller continuously monitors the load and writes an event log if the load is higher than expected. This event log is saved and logged in the controller memory.



Note

When using `LoadId` or `LoadIdentification` to identify the load of a tool or payload with unknown mass, the mass is estimated using the manipulator and the result can deviate from the actual mass. This is due to tolerances and variations between mechanical units. This does not necessarily mean that the identified payload or tool will cause issues with motion performance. If a very accurate value for the mass is required, it is recommended to weigh the tool or payload and use known mass in the identification.

Continues on next page

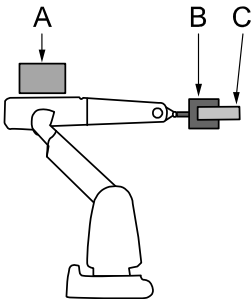
6 Programming and testing

6.10.7 LoadIdentify, load identification service routine

Continued

LoadIdentify

LoadIdentify can identify the tool load and the payload. The data that can be identified are mass, center of gravity, and moments of inertia.



en0500001535

| | |
|---|----------------|
| A | Upper arm load |
| B | Tool load |
| C | Payload |

Before running the load identification for the payload, make sure that the tool load data is correctly defined first, for example by running LoadIdentify for the tool.

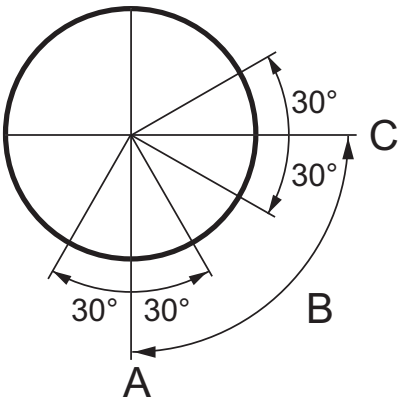
To identify the mass of B and C, axis 3 has to perform some movement. This means that to identify the mass, the upper arm load A must be known and correctly defined first.

To improve accuracy if the upper arm load A is mounted, input the known mass of B and C and choose the *known mass* method when identifying.

Configuration angles

To perform the identification the robot moves the load after a specific pattern and calculates the data. The axes that move are 3, 5 and 6. At the identification position, the motion for axis 3 is approximately ± 3 degrees and for axis 5 it is approximately ± 30 degrees. For axis 6 the motion is performed around two configuration points.

The optimum value for the configuration angle is either +90 degrees or -90 degrees.



en0500001537

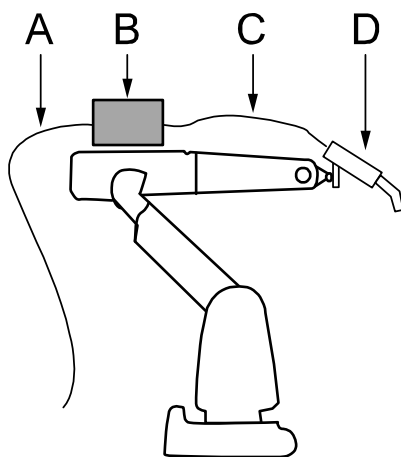
| | |
|---|----------------------------------|
| A | Configuration 1 (start position) |
| B | Configuration angle |

Continues on next page

| | |
|---|-----------------|
| C | Configuration 2 |
|---|-----------------|

LoadIdentify with arm loads mounted

The best way to perform load identification is to use a robot with no arm loads mounted. If this is not possible, good accuracy can still be achieved. Consider, for example, the robot in the figure below, which has arc welding equipment mounted on it.



en0500001536

| | |
|---|---------|
| A | Cable 1 |
| B | Load 1 |
| C | Cable 2 |
| D | Load 2 |

If we want to use load identification to find the data of load 2, the most important thing to remember is to make sure that the upper arm load is correctly defined, in particular its mass and center of gravity along the robot arm. The arm load includes everything that is mounted on the robot, except tool load and payload. In the figure above, cable 1, cable 2, and load 1 are included in the arm load, the total weight and center of gravity have to be calculated.

When performing the load identification, cable 2 should be disconnected since it will otherwise put an extra force on load 2. When identifying load 2 with such a force present, the result may differ considerably from the correct load. Ideally, cable 2 should be disconnected from load 2 and fastened on the upper arm. If this is not possible, the cable can also be disconnected at load 1 and fastened to the upper arm in such a way that the resulting force on load 2 is minimized.

Prerequisites for tool loads

Before running the LoadIdentify service routine for a tool load, make sure that:

- The tool is selected in the jogging menu.
- The tool is correctly mounted.
- Axis 6 is close to horizontal.
- The upper arm load is defined, if the tool mass is to be identified.
- The axes 3, 5, and 6 are not close to their corresponding working range limits.

Continues on next page

6 Programming and testing

6.10.7 LoadIdentify, load identification service routine

Continued

- The speed is set to 100%.
- The system is in manual mode.

Note that LoadIdentify cannot be used for `tool0`.

Prerequisites for payloads

Before running the LoadIdentify service routine for a payload, make sure that:

- The tool and payload are correctly mounted.
- Axis 6 is close to horizontal.
- The tool load is known (run LoadIdentify for the tool first).
- The upper arm load is defined, if the payload mass is to be identified.
- When using a moving TCP, the tool must be calibrated (TCP).
- When using a stationary TCP, the corresponding work object must be calibrated (user frame and object frame).
- The axes 3, 5, and 6 are not close to their corresponding working range limits.
- The speed is set to 100%.
- The system is in manual mode.

Note that LoadIdentify cannot be used for `load0`.

Running LoadIdentify

To start the load identification service routine you must have an active program in manual mode. Also the tool and payload that you want to identify must be defined and active in Jog.



Tip

Always run load identification with cold motors (no warm-up).

- 1 On the start screen, tap **Operate**, and then select **Service Routines** from the menu.
- 2 Tap LoadIdentify.
- 3 Press the three-position enabling device.
- 4 In the **Control Panel** tap **Play**. You can also press the **START** hard button on the FlexPendant.
The **Load Identification** operator message window is displayed.
- 5 Tap **OK** to confirm that current path will be cleared and that the program pointer will be lost.
- 6 In the operator message window, tap **Tool** or **PayLoad**.
- 7 Tap **OK** to confirm that the correct tool or payload is active in the jogging menu and that the tool or payload is correctly mounted.

Continues on next page



Note

If it is not correct, release the three-position enabling device and select the correct tool/payload in the **Jogging** menu. Then return to LoadIdentify, press the three-position enabling device, and press Start. Tap **Retry** and confirm that the new tool/payload is correct.

- 8 When identifying tool loads, confirm that the tool is active.
When identifying payloads, confirm that the payload's tool is active and calibrated.
- 9 When identifying payloads with stationary TCP, confirm that the correct work object is active and (preferably) calibrated. If it is correct, tap **OK** to confirm.
- 10 Select identification method. If you select the method where the mass is assumed to be known, remember that the tool/payload that you use must have the correct mass defined. Tap **OK** to confirm.
- 11 Select the configuration angle. The optimum is +90 or -90 degrees. If this is impossible, tap **Other** and set the angle. The minimum is +30 or -30 degrees.
- 12 If the robot is not in a correct position for load identification, you will be asked to jog one or more axes roughly to a specified position. When you have done this tap **OK** to confirm.

If the robot is still not in a correct position for load identification, the robot will slowly move to the correct position. Press **Move** to start the movement.



Note

Axis 1 to 3 must not be more than 10 degrees from proposed position.

- 13 The robot can go through the load identification movements slowly before performing the load identification (slow test). Tap **Yes** if you want a slow test and **No** to proceed to the identification.



Note

This is useful for ensuring that the robot will not hit anything during the identification. However, this will take a lot longer time.



Note

If the load identification is planned to be run in manual full speed, then the slow test is required before the actual measurement can start.

- 14 The setup for load identification is now complete. To start the motion, switch to Automatic mode and Motors On. Then tap **Move** to start the load identification movements.
- 15 When the identification is finished, switch back to manual mode, press the three-position enabling device and the Start button. Tap **OK** to confirm.
- 16 The result of the load identification is now presented on the FlexPendant.

Continues on next page

6 Programming and testing

6.10.7 LoadIdentify, load identification service routine

Continued

For robots that support the *Load diagram check* functionality, there is a message if the load is approved or not, and an **Analyze** button to view more information.

- 17 Tap **Yes** to update the selected tool or payload with the identified parameters.
Tap **No** to exit LoadIdentify without saving the parameters.

Load diagram check

For robots that support the *Load diagram check* functionality, the combination of the arm load, tool and payload, will be assessed against the load diagram. The ratings of total handling weight as well as of the center of gravity distance to the load diagram in Z and L directions will be provided both for wrist-up and wrist-down configurations.

There is a message if the load is approved or not, and an **Analyze** button to view more information.

- Load approved
- Load not approved
- Load approved only with wrist down

Running LoadIdentify with ModalPayloadMode deactivated

When the system parameter ModalPayloadMode is deactivated, set to 0, LoadIdentify will identify the tool load and the total load. It is no longer possible to define the payload.

With ModalPayloadMode deactivated it is possible to use the `\TLoad` argument in movement instructions. The `\TLoad` argument describes the total load used in the movement. The total load is the tool load together with the payload that the tool is carrying. If the `\TLoad` argument is used, then the `loaddata` in the current `tooldata` is not considered. For more information about ModalPayloadMode in movement instructions, see the section MoveL in *Technical reference manual - RAPID Instructions, Functions and Data types*.

To start the load identification service routine you must have an active program in manual mode and the tool and payload that you want to identify must be defined and active in the **Jogging** window.



Tip

Always run load identification with cold motors (no warm-up).

- 1 On the start screen, tap **Operate**, and then select **Service Routines** from the menu.
- 2 Tap LoadIdentify.
- 3 Press the three-position enabling device.
- 4 In the **Control Panel** tap **Play**. You can also press the **START** hard button on the FlexPendant.

The **Load Identification** operator message window is displayed.

Continues on next page

- 5 Tap **OK** to confirm that current path will be cleared and that the program pointer will be lost.
- 6 In the operator message window, tap **Tool** or **PayLoad**.
- 7 Tap **OK** to confirm that the correct tool or payload is active in the jogging menu and that the tool or payload is correctly mounted.



Note

If it is not correct, release the three-position enabling device and select the correct tool/payload in the **Jogging** menu. Then return to LoadIdentify, press the three-position enabling device, and press Start. Tap **Retry** and confirm that the new tool/payload is correct.

- 8 When identifying tool loads, confirm that the tool is active.
- 9 Select the identification method. If you select the method where the mass is assumed to be known, remember that the tool/total load that you use must have the correct mass defined. Tap **OK** to confirm.
- 10 Select the configuration angle. The optimum is +90 or -90 degrees. If this is impossible, tap **Other** and set the angle. The minimum is +30 or -30 degrees.
- 11 If the robot is not in correct position for load identification, you are asked to jog one or more axes roughly to a specified position. When you have done this tap **OK** to confirm.

If the robot is still not in a correct position for load identification, the robot will slowly move to the correct position. Press **Move** to start the movement.



Note

Axis 1 to 3 must not be more than 10 degrees from proposed position.

- 12 The robot can go through the load identification movements slowly before performing the load identification (slow test). Tap **Yes** if you want a slow test and **No** to proceed to the identification.



Note

This is useful for ensuring that the robot will not hit anything during the identification. However, this will take a lot longer time.



Note

If the load identification is planned to be run in manual full speed, then the slow test is required before the actual measurement can start.

- 13 The setup for load identification is now complete. To start the motion, switch to Automatic mode and Motors On. Then tap **Move** to start the load identification movements.
- 14 When the identification is finished, switch back to manual mode, press the three-position enabling device and the Start button. Tap **OK** to confirm.

Continues on next page

6 Programming and testing

6.10.7 LoadIdentify, load identification service routine

Continued

- 15 The result of the load identification is now presented on the FlexPendant. Tap **Tool** if you want to update the selected tool, tap **Loaddata** if you want to update the total load, or tap **No** if you want to quit without saving.
- 16 If **Loaddata** is selected it is possible to update the total load to either an existing or a new `loaddata` persistent variable.

Error handling

If the three-position enabling device is released during the load identification (before the movements start), the routine can always be restarted by pressing the three-position enabling device again and then pressing the Start button.

If an error should occur during the load identification movements, the routine must be restarted from the beginning. This is done automatically by pressing Start after confirming the error. To interrupt and leave the load identification procedure, tap **Cancel Call Routine** in **Program Editor** debug menu.

Limitations for LoadIdentify

Only tool loads and payloads can be identified with LoadIdentify. Thus arm loads cannot be identified.

If the load identification movements are interrupted by any kind of stop (program stop, emergency stop, etc.), the load identification must be restarted from the beginning. Confirm the error and press **Start** to automatically restart.

If the robot is stopped on a path with program stop and load identification is performed at the stop point, the path will be cleared. This means that no regain movement will be performed to return the robot back to the path.

The load identification ends with an EXIT instruction. That means that the program pointer is lost and must be set to main before starting any program execution.



Tip

The tool and/or payload data can be set manually if the load is small (10% or less of the maximum load) or symmetrical, for example if the tool load is symmetrical around axis 6.



Tip

If the mass of the tool or payload is unknown, the service routine LoadIdentify can in some cases identify a 0 kg mass. If the load is very small compared to the maximum load for the robot, then a 0 kg mass can be ok. Otherwise, try the following to identify the mass.

- Check that the arm loads are correctly defined and redo the identification.
- Find the weight of the load in some other way and perform a load identification with known mass to remove the dependency on arm loads.

Continues on next page

LoadIdentify for 4-axis palletizing and SCARA robots

When running LoadIdentify on a robot with 4 instead of 6 axes, there are some differences. In this description of the differences the robot type is assumed to be similar to IRB 260, IRB 460, IRB 660, IRB 760, or IRB 910SC.

The main differences are:

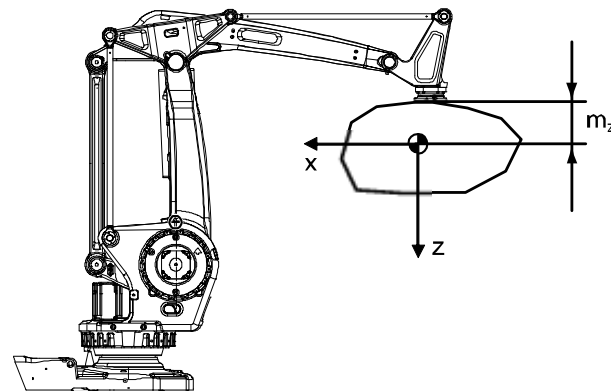
- The used axes are:
 - 1 (or 2 for some robot models)
 - 3 (for all robot models)
 - 6 (or 4 for some robot models)
- Because the first axis (axis 1 or 2) is used, the resulting movements can be large.
- Not all load parameters can be identified.

The first axis (axis 1 or 2) will move approximately ± 23 degrees from its current position. Therefore, the load can move a large distance during the identification. Axes 3 and 6 (or 4) will move as for 6-axis robots. The configuration angle for axis 6 (or 4) works exactly as for 6-axis robots.

Because there is not 6 axes, a 4-axis robot cannot identify all parameters of the load. The following parameters cannot be identified:

- I_x - The inertia around the x-axis.
- I_y - The inertia around the y-axis.
- m_z - The z-coordinate for the center of mass.

However, for this type of robot the above parameters have negligible effect on the motion performance. See the definition of the load coordinate system in the following figure.



xx0900000021



Tip

It is possible that the identification procedure fails to estimate the center of gravity if the measured torque data has too high variance. If this happens, it should still be possible to get good results by running the LoadIdentify routine again, preferably with another position of the last axis.

Continues on next page

LoadIdentify for 4-axis delta robots

When running LoadIdentify on a delta (picker/packer) robot with 4 axes, there are some differences. In this description of the differences the robot type is assumed to be similar to IRB 360, IRB 365, or IRB 390.

The main differences are:

- The used axes are:
 - 3 (for all robot models)
 - 4 (for all robot models)
- Not all load parameters can be identified.

Axis 4 will move approximately +60 degrees from its current position. Therefore, the load will move a large rotational distance during the identification.

Because there is not 6 axes, a 4-axis delta robot cannot identify all parameters of the load. In fact, only the following parameters are possible to identify:

- I_z , the inertia around the z-axis.
- m , the total mass.

However, for this type of robot the total mass and the inertia around the z-axis is enough to give good motion performance. All other `loaddata` parameters will be set to zero.

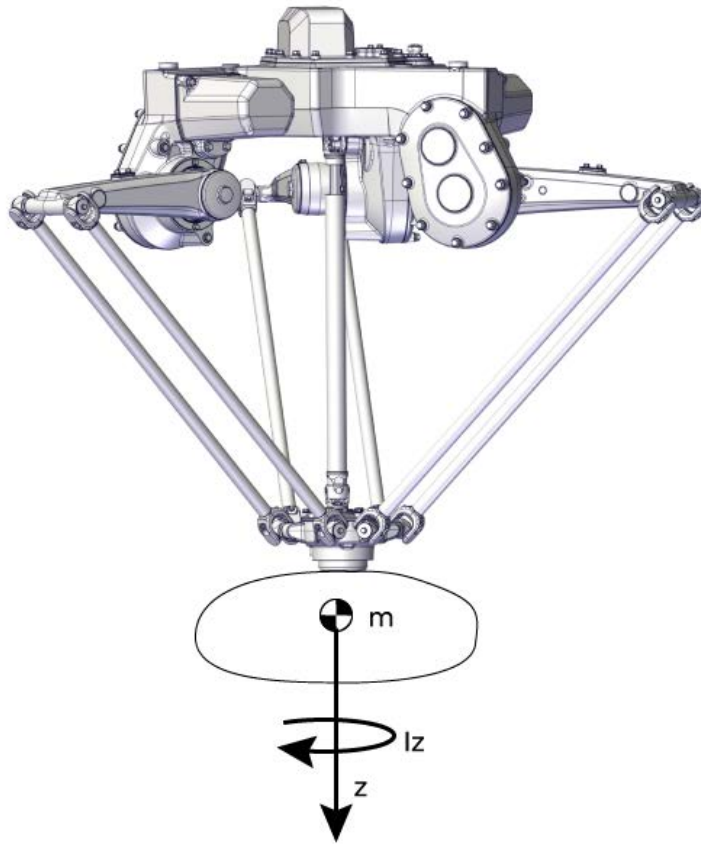


Note

An x,y-offset in the mass center (m_x, m_y) should not be manually entered since this information will be included in the identified I_z .

(However, if known, an offset m_z (z-coordinate for the center of mass) can be manually entered to slightly enhance motion performance.)

See the resulting `loaddata` from LoadIdentify for 4-axis delta robots in the following figure.



xx2100002581

LoadIdentify for 5-axis delta robots

When running LoadIdentify on a delta (picker/packer) robot with 5 axes, there are some differences. In this description of the differences the robot type is assumed to be similar to IRB 365 or IRB 390.

The main differences are:

- The used axes are:
 - 3 (for all robot models)
 - 4 (for all robot models)
 - 5 (for all robot models)
- Not all load parameters can be identified.

Axis 4 will move approximately +45 degrees from its current position. Therefore, the load will move a large rotational distance during the identification.

Axis 5 will move approximately +35 degrees from its current position. Therefore, the load will move a large rotational distance during the identification.

Instead of the usual 0-90 degree configuration movement, the axis 5 will move ± 45 degrees.

Because there is not 6 axes, a 5-axis delta robot cannot identify all parameters of the load. The following parameters are possible to identify:

- I_y , the inertia around the y-axis.

Continues on next page

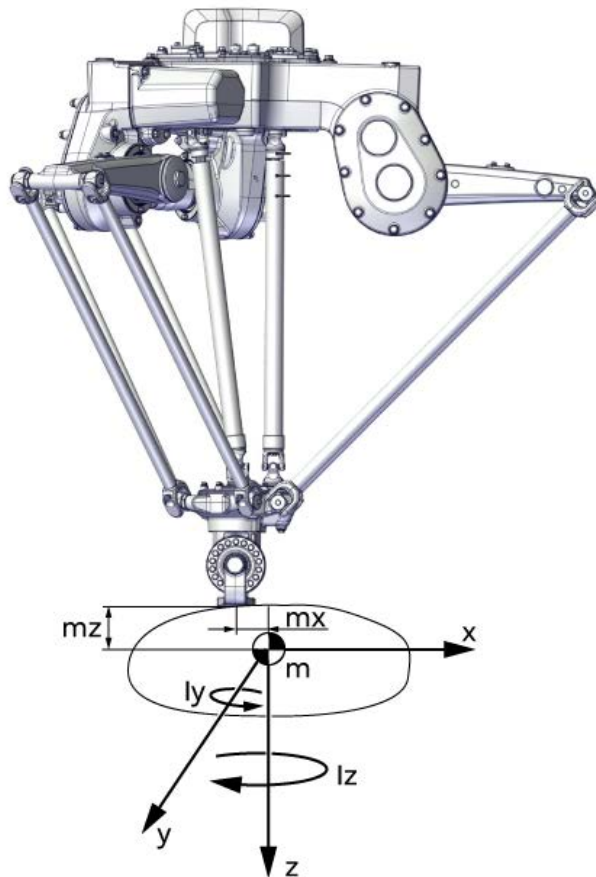
6 Programming and testing

6.10.7 LoadIdentify, load identification service routine

Continued

- I_z , the inertia around the z-axis.
- m_x , the x-coordinate for the center of mass.
- m_z , the z-coordinate for the center of mass.
- m , the total mass.

See the resulting `loaddata` from `LoadIdentify` for a 5-axis delta robot in the following figure.



xx2100002620

Related information

It is also possible to include `LoadIdentify` in a program by using RAPID instructions. See `LoadID` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

The product manual for the robot contain information on how and where to mount the loads.

How to define the system parameters for arm loads is described in *Technical reference manual - System parameters*.

6.10.8 Brake check service routine

Overview

The BrakeCheck service routine is used to verify whether the mechanical brakes work correctly.

The BrakeCheck service routine is included in the RobotWare installation if the controller does not have SafeMove option.



Note

If the controller has SafeMove option, the RobotWare installation includes *Cyclic Brake Check* service routine. For more details, see *Application manual - Functional safety and SafeMove*.

While running the BrakeCheck service routine the brakes are tested in consecutive order and each test takes 10-15 seconds.

Prerequisites for running the BrakeCheck service routine

Following are the prerequisites for running the BrakeCheck service routine:

- The robot and all additional axes must be moved to a safe and relaxed position (away from people, equipment and not too much stretched) before performing a brake check. Normally the robot moves only a few centimeters during the brake check.
- Move the robot to a stop point before performing a brake check.
- The brake check can be performed only at normal execution level (not from a trap routine, error handler, event routine, or store path level).

Exclude individual axes from the brake check

It is possible to exclude individual axes from the brake check. For this, set the value of system parameter `Deactivate Cyclic Brake Check for axis` to On. For more details, see [Configure system parameters on page 221](#).

Running the brake check

Following are the two ways to initiate a BrakeCheck service routine:

- Call the BrakeCheck service routine from FlexPendant. The controller must be in manual mode.
- Call the procedure BrakeCheck from the RAPID program.



WARNING

While the brake check routine is active, do not change the speed from the FlexPendant and do not use the instructions `VelSet`, `AccSet`, `SpeedRefresh`, or any other instruction that affects the motion performance in trap routines or event routines.

Continues on next page

6 Programming and testing

6.10.8 Brake check service routine

Continued



Note

The RAPID function `IsBrakeCheckActive` can be used to check if `BrakeCheck` is active.

Interrupt the brake check

It is not recommended, but it is possible to stop the execution while running a brake check.

If the brake check is interrupted, it will be resumed when the program execution starts again. The brake check can be resumed up to 3 times.

Brake maintenance

Brake maintenance is a feature in the brake check functionality.

The `BrakeCheck` routine automatically detects if maintenance of the mechanical brakes is needed and activates the *Brake maintenance* functionality during execution. *Brake maintenance* applies the brake and turns the motor shaft 1 radian five times, which gives a movement of the robot arm of less than 1 degree.

There are event logs that tell if *Brake maintenance* is needed, and if it has been run.

For more information see parameter *Brake Maintenance*, type *General Rapid*, topic *Controller*, in *Technical reference manual - System parameters*.

Event logs

When `BrakeCheck` is executed, the following event logs will be shown:

| Event log | Title |
|-----------|---------------------|
| 10272 | Brake Check Done |
| 10273 | Brake Check Started |

If there is a problem with one or several mechanical brakes, an event log that is describing which mechanical unit and which axis that has bad brakes will be shown:

| Event log | Title |
|-----------|---------------------------|
| 37234 | Brake Performance Warning |
| 37235 | Brake Performance Error |

Brake check for MultiMove systems



Note

Make sure that all mechanical units are standing still before starting a `BrakeCheck` routine.

One of the motion tasks call the routine `BrakeCheck` to perform brake check for all mechanical units in all tasks.

The brake check must not be performed while any tasks are in synchronized mode (coordinated movement). It is necessary to synchronize all motion tasks with `WaitSyncTask` instructions before and after the actual brake check. If running a

Continues on next page

movement instruction when one motion task is execution a `BrakeCheck`, you will have an error (41888) and all execution will stop. Instruction `ExitCycle` is also forbidden to use during an active `BrakeCheck`.

The RAPID function `IsBrakeCheckActive` can be used to check if there is an ongoing `BrakeCheck`.

Only one call to `BrakeCheck` can be done at a time. This is checked by the service routine and if more than one RAPID task or client try to execute the routine, you will have an error (41886).

Program example

```
T_ROB1
PERS tasks task_list{2} := [ ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync1;
VAR syncident sync2;
...
IF PLC_dil_DO_CBC = 1 THEN
    WaitSyncTask sync1, task_list;
    BrakeCheck;
    WaitSyncTask sync2, task_list;
ENDIF

T_ROB2
PERS tasks task_list{2} := [ ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync1;
VAR syncident sync2;
...
IF PLC_dil_DO_CBC = 1 THEN
    WaitSyncTask sync1, task_list;
    ! Wait for T_ROB1 to be ready with BrakeCheck
    WaitSyncTask sync2, task_list;
ENDIF
```

Brake check on additional axes

To be able to run brake check on additional axes, the parameter *Max Static Arm Torque* (in topic *Motion* and type *Brake*) needs to be calculated for the additional axis and entered into the configuration. Brake check uses this value when testing the brake at error-level.

The parameter should be the maximum static torque that the brake needs to withstand when the additional axis is positioned in maximum gravity. The following formula should be used:

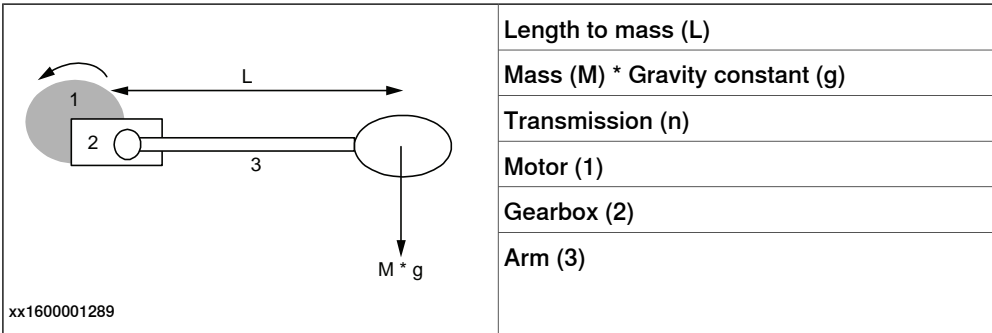
$$\text{Max Static Arm Torque} = (M * L * g) / n$$

Continues on next page

6 Programming and testing

6.10.8 Brake check service routine

Continued



To calculate the parameter for an axis that has no gravity, for example a track, the below formula may be used:

$$\text{Max Static Arm Torque} = T_{\text{brake min}}/1.35$$

T_{brake min} for ABB motor units can be found in the product specification for the specific motor unit, see *Product specification - Motor Units and Gear Units*.

For more information about parameter *Max Static Arm Torque*, see topic *Motion*, type *Brake* in *Technical reference manual - System parameters*.



Note

Note that the calculated value should be entered in [Nm] and calculated to the motor side.

Description of the I/O setup

Signal configuration

It is possible to configure digital output signals that reflect the status of the mechanical brakes in an open RAPID module. The digital output signals that can be configured are OK, WARNING, ERROR, and ACT (brake check active) for each drive module.

The signal configuration should be done in the RAPID module *BC_config_IO.sysx*, see [Description of the I/O setup on page 218](#).

The file *BC_config_IO.sys* can be found in directory */products/RobotControl_7.x.xxxx/utility/BrakeCheck/*, and must then be copied to the *HOME* directory of the active system.



Note

Remember to update the I/O configuration with the digital output signals.

In a MultiMove system you need to define an OK, WARNING, ERROR, and ACT digital output signals for each drive module.



Note

If the signals should keep their values after a power fail, the power fail settings in the system parameters must also be updated, see [Description of the I/O setup on page 218](#).

Continues on next page

Description of the BC_config_IO module

```
MODULE BC_config_IO(SYSMODULE,NOVIEW)
  PROC BC_config_IO_proc(VAR string user_io_names{*,*})
    !TPWrite "BC_config_IO_proc";
    ! Define your own signals. The signal
    ! names must be signals defined in EIO.cfg

    ! If 1 drive module
    user_io_names{1, 1}:="BCACT1";
    user_io_names{1, 2}:="BCOK1";
    user_io_names{1, 3}:="BCWAR1";
    user_io_names{1, 4}:="BCERR1";

    ! If 2 drive modules
    !user_io_names{2, 1}:="BCACT2";
    !user_io_names{2, 2}:="BCOK2";
    !user_io_names{2, 3}:="BCWAR2";
    !user_io_names{2, 4}:="BCERR2";

    ! If 3 drive modules
    !user_io_names{3, 1}:="BCACT3";
    !user_io_names{3, 2}:="BCOK3";
    !user_io_names{3, 3}:="BCWAR3";
    !user_io_names{3, 4}:="BCERR3";

    ! If 4 drive modules
    !user_io_names{4, 1}:="BCACT4";
    !user_io_names{4, 2}:="BCOK4";
    !user_io_names{4, 3}:="BCWAR4";
    !user_io_names{4, 4}:="BCERR4";
  ENDPROC
ENDMODULE
```

Description of the EIO.cfg file

```
EIO:CFG_1.0:6:1::
...
#
EIO_SIGNAL:

-Name "BCACT1" -SignalType "DO"

-Name "BCOK1" -SignalType "DO"

-Name "BCWAR1" -SignalType "DO"

-Name "BCERR1" -SignalType "DO"
```

Continues on next page

6 Programming and testing

6.10.8 Brake check service routine

Continued

Brake check signal description

Introduction

Description of different signal states for brake check in the BrakeCheck routine.
The signal names are according to [Description of the I/O setup on page 218](#).

Timing sequence for brake check signals

Description of which signals are set at different times during the BrakeCheck execution.

Beginning of brake check

The following signals are set in the beginning of the BrakeCheck execution.

| Signal | Set to |
|--------|--------|
| BCOK | 0 |
| BCACT | 1 |
| BCERR | 0 |
| BCWAR | 0 |

End of brake check

The following signals are set in the end of the BrakeCheck execution.

| Signal | BrakeCheck test OK Set to | BrakeCheck test WARNING Set to | BrakeCheck test ER- ROR Set to |
|--------|------------------------------|--------------------------------------|--------------------------------------|
| BCOK | 1 | 0 | 0 |
| BCERR | 0 | 0 | 1 |
| BCWAR | 0 | 1 | 0 |
| BCACT | 0 | 0 | 0 |

Program Pointer moved to Main after interrupted brake check

When the program pointer is moved to Main after an interrupted BrakeCheck execution, the following signals are set.

| Signal | Set to |
|--------|--------|
| BCOK | 0 |
| BCACT | 0 |

During the first brake check test

| Signal | Signal state |
|--------|--------------|
| BCOK | 0 |
| BCERR | 0 |
| BCWAR | 0 |
| BCACT | 1 |

Interrupted brake check test, program pointer still in BrakeCheck routine

| Signal | Signal state |
|--------|--------------|
| BCOK | 0 |

Continues on next page

| Signal | Signal state |
|--------|--------------|
| BCERR | 0 |
| BCWAR | 0 |
| BCACT | 1 |

Interrupted brake check test, program pointer moved from BrakeCheck routine

| Signal | Signal state |
|--------|--------------|
| BCOK | 0 |
| BCERR | 0 |
| BCWAR | 0 |
| BCACT | 0 |

Configure system parameters

About the system parameters

The configuration of system parameters required for a robot system should be made before running the brake check.



Note

The controller must be restarted after changing the system parameters.

Type Mechanical Unit

All mechanical units for additional axes that shall be supervised must have the parameters *Activate at Start Up* and *Deactivation Forbidden* set to On. (Supervised mechanical units must always be active.)

Type Arm

If an axis should be excluded from brake check, set the parameter *Deactivate Cyclic Brake Check for axis* to On.

Type Brake

If brake check is executed on an additional axis, a lowest safe brake torque must be defined. A 5% margin is added during the test for setting the fail limit. The parameter used is *Max Static Arm Torque* defined in on motor side. A warning limit is set with a higher torque value (depending on the brake).

6.10.9 Cyclic Brake Check service routine

Introduction

Cyclic Brake Check is a function that verifies that the brakes work correctly.

The *Cyclic Brake Check* service routine is included in the RobotWare installation if the controller has the option *SafeMove*. For details about working with *Cyclic Brake Check* and *SafeMove*, see *Application manual - Functional safety and SafeMove*.

Functionality

The Cyclic Brake Check is initiated by the robot controller or an external PLC. The robot moves to a safe position where the brakes are locked with servos engaged. The motors of the robot are then used to generate the torque. If any axis moves, the system is set in reduced speed mode. A new successful Cyclic Brake Check must be performed before the robot can be used again with normal speeds.

With a defined interval (brake cycle time), the robot must move to the safe position and perform a Cyclic Brake Check. If Cyclic Brake Check is not performed within the brake cycle time an error message is generated, and depending on configuration the robot will be set to reduced speed or keep its normal supervision levels. A predefined time (pre-warning time) warning appears on the FlexPendant before the brake cycle time has passed.

Pre-requisites for Cyclic Brake Check

Pre-requisites for Cyclic Brake Check:

- The robot and all the additional axes must be moved to a safe and relaxed position (away from people, equipment, and not too much stretched) before performing a brake check. Normally the robot moves only a few centimeters during the brake tests.
- Move the robot to a stop point before performing a Cyclic Brake Check.
- A Cyclic Brake Check can only be performed at normal execution level (not from a trap routine, error handler, event routine or store path level).
- Brakes are tested in consecutive order and each test takes about 10-15 seconds.
- Do not change the speed from the FlexPendant and do not use `VelSet`, `AccSet`, `SpeedRefresh`, or any other instruction that affects motion performance in trap routines or event routines while Cyclic Brake Check is active.



Note

The RAPID function `IsBrakeCheckActive` can be used to check if Cyclic Brake Check is active.

Continues on next page

Activate Cyclic Brake Check

Cyclic Brake Check can be initiated in the following way:

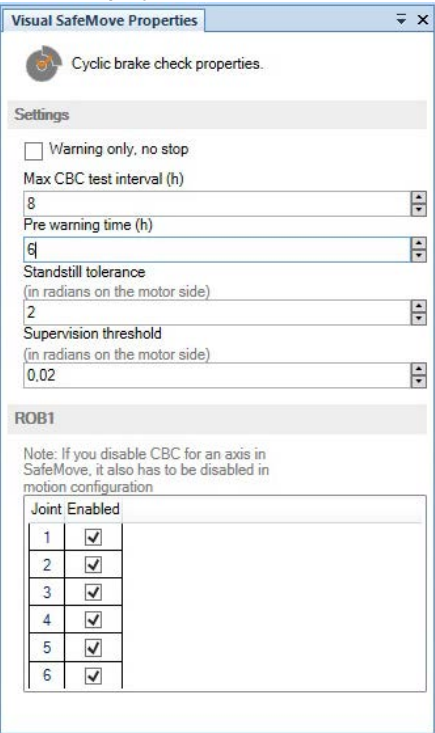


- Run the *Cyclic Brake Check* service routine from FlexPendant. The controller must be in manual mode.
- Run the RAPID program procedure `CyclicBrakeCheck`.

**Note**

Cyclic Brake Check cannot be dynamically activated or deactivated. If it is configured to be active, it is always active. That is, a constant supervision to verify the brake check has been performed within the configured time interval.

Cyclic Brake Check configuration

Use the following procedure to configure Cyclic Brake Check:


| | Action | Note/illustration |
|---|---|---|
| 1 | <p>In the Visual SafeMove tab ribbon, navigate to the Create group, and click on the Cyclic Brake Check button.</p> <p>The Cyclic brake check properties window is displayed.</p>  <p>xx1700001098</p> |  <p>xx1700001355</p>  <p>WARNING</p> <p>The Warning only, no stop check box should not be selected under any circumstances.</p> |
| 2 | <p>In Max CBC test interval (h), set the maximum allowed time (in hours) between the cyclic brake checks.</p> | <p>Note</p> <p>It is recommended to configure the maximum value in Max CBC test interval (h) as 8 hours.</p> |

Continues on next page

6 Programming and testing

6.10.9 Cyclic Brake Check service routine

Continued

| | Action | Note/illustration |
|---|--|--|
| 3 | In Pre warning time (h) , set how long before the end of the interval a warning should be displayed on the FlexPendant. | |
| 4 | Standstill tolerance is used for Stand Still Supervision during brake test. The motor is in regulation during brake test, and a small movement may be allowed. The size of the allowed movement is specified in Standstill tolerance (in radians on motor side). The default value is 2 radians. | Do not change the default value for Standstill tolerance . |
| 5 | Supervision threshold defines the threshold to verify that a brake check has been made. The default value is 0.02 radians. | Do not change the default value for Supervision threshold . |
| 6 | Make sure axes 1 to 6 are selected. If not, select the appropriate check box for the corresponding axis. |  WARNING An axis can be excluded only after an appropriate safety analysis has been performed. This must correspond with the axis that has the system parameter <i>Deactivate Cyclic Brake Check for axis</i> set to <i>On</i> . For the axes not included in SafeMove, deactivation of the axes must be done by setting the parameter <i>Deactivate Cyclic Brake Check for axis</i> to <i>On</i> through RobotStudio. |



Note

An error or warning message is logged for each axis with low brake torque. A status message is also logged for each complete brake cycle.

Example of RAPID program

A signal `diPSC1CBCPREWARN` is defined and cross connected to the safety signal `PSC1CBCPREWARN` in the I/O parameters. This cross connection is required because there is a restricted usage of safety signals in the RAPID program. The `PSC1CBCPREWARN` signal will be set to a logical high state when the prewarning time interval expires. It will be kept high until a successful brake check has been carried out. The status of the `diPSC1CBCPREWARN` signal is checked in the Main loop in the application program.

```
PROC main()  
  IF diPSC1CBCPREWARN=1 THEN  
    MoveAbsJ *, v1000, fine, tool1;  
    ! Call to the predefined service routine CyclicBrakeCheck  
    CyclicBrakeCheck;  
  ENDIF  
  ....  
ENDPROC
```

Brake maintenance

Brake maintenance is a feature in the Cyclic Brake Check functionality.

Continues on next page

Cyclic Brake Check automatically detects if maintenance of the mechanical brakes is needed and activates the *Brake maintenance* functionality during execution.

Brake maintenance applies the brake and turns the motor shaft 1 radian five times, which gives a movement of the robot arm of less than 1 degree.

There are event logs that tell if *Brake maintenance* is needed, and if it has been run.

For more information see parameter *Brake Maintenance*, type *General Rapid*, topic *Controller*, in *Technical reference manual - System parameters*.

Interrupted Cyclic Brake Check

It is not recommended, but it is possible to stop the execution while running a Cyclic Brake Check.

If the Cyclic Brake Check is interrupted, it will be resumed when the program execution starts again. The Cyclic Brake Check can be resumed up to 3 times.

If the Cyclic Brake Check is interrupted more than 3 times, a new cyclic brake check is required. Only reduced speed can be used until a new Cyclic Brake Check is performed.

6 Programming and testing

6.10.10 Wrist optimization service routine

6.10.10 Wrist optimization service routine

Overview

The wrist optimization service routine is used to improve the TCP reorientation performance. More information is available in the product manual for the manipulator.



Note

For advanced users, it is also possible to use the do the wrist optimization using the RAPID instruction `WristOpt`, see *Technical reference manual - RAPID Instructions, Functions and Data types*.

6.10.11 SkipTaskExec service routine

Overview

The `SkipTaskExec` service routine is used for debugging if there are problems with synchronizing tasks during Multitasking. It also makes it possible to run a program in automatic mode and exclude one task from the execution.

6 Programming and testing

6.10.12 Commutation service routine

6.10.12 Commutation service routine

When to use this service routine

The service routine **Commutation** aligns a motor's electrical signals with its physical position to ensure accurate torque generation and precise motion control.



CAUTION

If the motor is not properly commutated, it can rev up and break.

Commutation

The service routine sends a current through the motor, reads the angle (relation between motor and resolver), and writes these values to the configuration.



Note

The current must be defined in the system parameters, type *Motor Type*.

7 Running in production

7.1 Introduction

Overview

The content in this section applies to a robot and not a robot system. It is the responsibility of the integrator to provide a safety and users manual for the robot system.

7 Running in production

7.2.1 Starting programs

7.2 Basic procedures

7.2.1 Starting programs

Starting programs

Use the following procedure to start a program or to continue running a program that has been stopped.

- 1 Check that all the necessary preparations are done to the robot and make sure no obstacles exist within the safeguarded space.
- 2 Make sure no personnel are inside the safeguarded space.
- 3 On the FlexPendant, tap **QuickSet** and select a controller operating mode from the **Mode** section.



Note

For more details about controller operating modes, see [OmniCore controller operating modes on page 77](#).

- 4 In the **Motors** section, tap **Motors on**.

In Auto mode:

- On the status bar, tap the QuickSet button, tap the **Control** tab, and tap **Motors On** in the **Motors** section.

In manual mode and manual high speed mode:

- Press and hold the three-position enabling device.



Note

YuMi robots with SafeMove requires using the enabling device.

On YuMi robots without SafeMove the enabling device is disabled, hence, not used.

- 5 Open **Operate**.

The **Advanced View** page is displayed.

- 6 Tap and make a selection based on the following scenarios:

- If there is no loaded program:
 - a list of recent programs is displayed in the **Recently used programs** section. Select a recently used program to load it, and proceed to step [8](#).



Note

Tap the **Load Other Program** button to load a program that is not listed in the **Recently used programs** section.

- tap the **Load Program** button if there is no list of recently used programs.

Continues on next page

- If there is already a program loaded: On the command bar, select **Load Program** from the context menu.
 - Tap **Yes** to continue loading a new program.
 - Tap **No** to continue with the existing program and proceed to step 8.
- 7 Browse to the location, select a program file, and tap **Load**.
The selected program is loaded.
- 8 Tap **Advanced** and select **Reset Program Pointer to Main**.
The program pointer is set to main. For more information about program pointer, see [Program and motion pointers on page 144](#).
- 9 Tap **QuickSet** and tap on the **Play** button.
The program execution starts.



Note

You can also press the **Start** hard button on the FlexPendant to start the program.

Continue running after the program is changed

You can continue running a program even if it has been changed.

In automatic mode, a warning dialog may appear to avoid restarting the program if the consequences are unknown.

| If you... | then tap... |
|---|-------------|
| are sure the changes you have made are not in conflict with the current robot position and that the program can continue without danger to equipment or personnel | Yes |
| are unsure of the consequences your changes might have and want to investigate further | No |

Restart from the beginning

A program can be restarted from **Operate** or **Code**.



Note

Resetting the program pointer will reset the program pointer in all the normal tasks and the background tasks.

Use the following procedure to restart a program from **Operate**.

- 1 On the start screen, tap **Operate**, and then select **Advanced View** from the menu.
- 2 The selected program is loaded.
- 3 Tap **Advanced** and select **Reset Program Pointer to Main**.
The program pointer is set to main.

Continues on next page

7 Running in production

7.2.1 Starting programs

Continued

Use the following procedure to restart a program from **Code**.

- 1 On the start screen, tap **Code**, and then select **Program Editor** from the menu.
- 2 On the right menu tap **Debug**.
- 3 Select **PP to Main**

The program pointer is set to main.

Limitations

Only one program can be executed at a time, unless your system has the Multitasking option. If so several programs can be executed simultaneously.

If the robot system encounters program code errors while the program is running, it will stop the program and the error is logged in the event log.

7.2.2 Stopping programs

Stopping programs

Use the following procedure to stop a program:

- 1 Check that the ongoing operation is in such a state that it can be interrupted.
- 2 Make sure it is safe to stop the program.
- 3 Press the **Stop** button on the FlexPendant.

The program is stopped.



Note

If your robot system has the *Multitasking* option installed, see [Using multitasking programs on page 234](#).



DANGER

In case of an emergency use the emergency stop button instead of the stop button.

Stopping a program with the **Stop** button does not mean that the robot will stop moving immediately.

Stopping execution when using hold-to-run or step-by-step execution

When using hold-to-run or step-by-step execution, execution can be stopped according to the following.

| Mode | Action | Information |
|-----------------------------------|--|--|
| Operation <i>with</i> hold-to-run | Press the Stop button. | The hold-to-run function is described in section The FlexPendant on page 20 . |
| Step-by-step mode | The robot will stop after executing each instruction. Execute the next instruction by pressing the Forward button again. | If you press the STOP button while executing a move instruction, the robot will stop without completing the move. |

7 Running in production

7.2.3 Using multitasking programs

7.2.3 Using multitasking programs

Overview

In a system with the option *Multitasking* installed, you may have one or several programs running in parallel, for instance in a *MultiMove* cell with more than one robot where each robot has its own task and program (multitasking).

Manually set up tasks

Tasks need to be set up in order to run as planned. Normally, all tasks are set up on delivery. Setting up tasks is done by defining system parameters of the type *Controller*. For information about system parameters, see *Technical reference manual - System parameters*.

You need detailed information to set up tasks manually, see your plant or cell documentation for details.

How tasks are run

Tasks may be defined as Normal, Static, or Semistatic. Static and Semistatic tasks are automatically started as soon as a program is loaded into that task.

Normal tasks are started when you press the **Start** button of the FlexPendant, and stopped when you press the **Stop** button.

Load, run, and stop multitasking programs

Use the following procedure to load, run, and stop multitasking programs.

- 1 Make sure there is more than one task set up. This is done using system parameters, see *Technical reference manual - System parameters*.
- 2 Load programs to the respective task using **Operate** or **Code**. This is described in the section [Loading programs to a task on page 234](#).
- 3 If one or more task should be disabled, on the status bar, tap the QuickSet menu, select the Execution tab, and disable the required tasks in the **Enable/Disable tasks** section.
Disabling the tasks can be done only in manual mode.
- 4 Start program execution by pressing the **Start** button on the FlexPendant. All the active tasks are started.
- 5 Stop program execution by pressing the **Stop** button on the FlexPendant. All the active tasks are stopped.

Loading programs to a task

This section describes how to load a program to a task in a multitasking system. It is assumed that the tasks have been configured.

Continues on next page

Load a program in **Operate**

Use the following procedure to load a saved program in **Operate**:

- 1 On the start screen, tap **Operate**, and then select **Advanced View** from the menu.
 - If there is no loaded program: The recent programs are displayed in the **Recently used programs** section. Select a recently used program to open it. Otherwise, tap on the **Load Other Program** button.
 - If there is already a loaded program: On the command bar, select **Load Program** in the context menu.
 - Tap **Yes** to continue loading a new program.
 - Tap **No** to continue with the existing program.
- 2 From the **Location** section, select the device where the program is saved. The folders and files available in the selected device are displayed.
- 3 Navigate and choose the required program file.



Note

The supported file format is .pgf

- 4 Tap **Load**.
The selected program is loaded to the selected task.

Load a program in **Code**

Use the following procedure to load a saved program in **Code**:

- 1 On the start screen, tap **Code**, and then select **Modules** from the menu.
- 2 On the command bar, select **Load Program** in the context menu.
If there is already a program loaded, the **Load Program** window appears.
 - Tap **Save** to save the loaded program. Once the loaded program is saved the **Load Program** page is displayed.
 - Tap **Don't save** to close the loaded program without saving it, that is, delete from the program memory. Once the loaded program is closed the **Load Program** page is displayed.
 - Tap **Cancel** to keep the currently loaded program.
- 3 From the **Location** section select the device where the program is saved. The folders and files available in the selected device are displayed.
- 4 Navigate and choose the required program file.



Note

The supported file format is .pgf

- 5 Tap **Load**.
The selected program is loaded in **Program Editor**.

Continues on next page

7 Running in production

7.2.3 Using multitasking programs

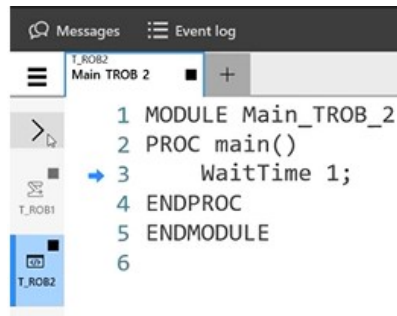
Continued

Viewing multitasking programs

In **Operate**, there is one tab for each task. To switch between viewing the different tasks, tap on the tabs.

Switching between tasks

In the **Program Editor/Code**, only one task is handled at a time. If there are multiple tasks, a task selection panel is available from which you can easily navigate between tasks.



xx2400002016

The selection panel icons indicate the type (motion/non-motion) and the status (started/stopped) of the task.



Note

Only normal tasks are displayed in the task selection panel. To view background tasks, expand the menu.

7.2.4 Returning the robot to the path

About paths and return regions

While a program is running, the robot or additional axis is considered to be *on path*, which means that it follows the desired sequence of positions.

If you stop the program the robot is still on path, unless you change its position. It is then considered to be *off path*. If the robot is stopped by an emergency or safety stop it may also be off path.

If the stopped robot is within the *path return region* you can start the program again, and the robot will return to the path and continue program execution.

Note that there is no way to predict the exact return movement for the robot.



Tip

The path return region is set with system parameters, see *Technical reference manual - System parameters, Type Path Return Region*.

Returning to path

Turning off the power to the robot motors often results in the robot slipping from its programmed path. This may occur after an uncontrolled emergency or safety stop. The allowed slip distance is configured with system parameters. The distance can be different depending on operating mode.



Note

On YuMi robots with SafeMove, axes 5 and 6 can drop slightly when releasing the enabling device, because there are no brakes on these axes. There is a risk that mounted tools or workpieces are damaged.

If the robot is not within the configured allowed distance, you may choose to let the robot return to the programmed path or continue to the next programmed point in the path. Then the program execution continues automatically in programmed speed.

- 1 Make sure there are no obstacles blocking the way and that payload and work objects are properly placed.
- 2 If necessary, put the system in automatic mode and press the Motors on button on the controller to activate the robot motors.
- 3 Press the Start button on the FlexPendant to continue execution from where it stopped. One of these things will happen:
 - The robot or axis slowly returns to the path and the execution continues.
 - The **Regain Request** dialog will be displayed.
- 4 If the **Regain Request** dialog is displayed, select the proper action.
 - Tap **Yes** to return to the path and continue the program.
 - Tap **No** to return to the next target position and continue the program.
 - Tap **Cancel** to cancel the program.

7 Running in production

7.2.5 Using motion supervision and non motion execution

7.2.5 Using motion supervision and non motion execution

Motion supervision

The controller software has the functionality aiming at reducing collision impact forces on the robot. This helps in protecting the robot and the external equipment from severe damage if a collision occurs.

Motion supervision during program execution is active by default and is always active, regardless which options are installed in the controller. When a collision is detected, the robot stops immediately and relieve the residual forces by moving a short distance in reversed direction along its path. The program execution stops with an error message. The robot remains in the Motors on state so that program execution can be resumed after the collision error message has been acknowledged.

Moreover, *Collision Detection* software option has extra features such as supervision during jogging. To find out if your system has this option installed, open the **Settings** app. Tap **System > About > Options** section which displays the available options.

Functions in Collision Detection

A RobotWare system with *Collision Detection* has the following additional functionality:

- *Path Supervision* used to prevent the mechanical damage due to the robot running into an obstacle during program execution.
- *Jog Supervision* used to prevent mechanical damage to the robot during jogging.



Note

All motion supervision must be set for each task separately.

Non motion execution

Non motion execution enables you to run a RAPID program without robot motion. All other functions like current cycle times, I/O, TCP speed calculation, and so on work normally.

Non motion execution can be used for program debugging or cycle time evaluation. It also represents a solution if you need to measure, for example, glue or paint consumption during a cycle.

When non motion execution is activated it can be executed in:

- manual mode
- manual full speed mode
- auto mode

Cycle times will be simulated according to the selected mode.

Continues on next page



Note

Non motion execution can only be activated when the system is in Motors Off state.



CAUTION

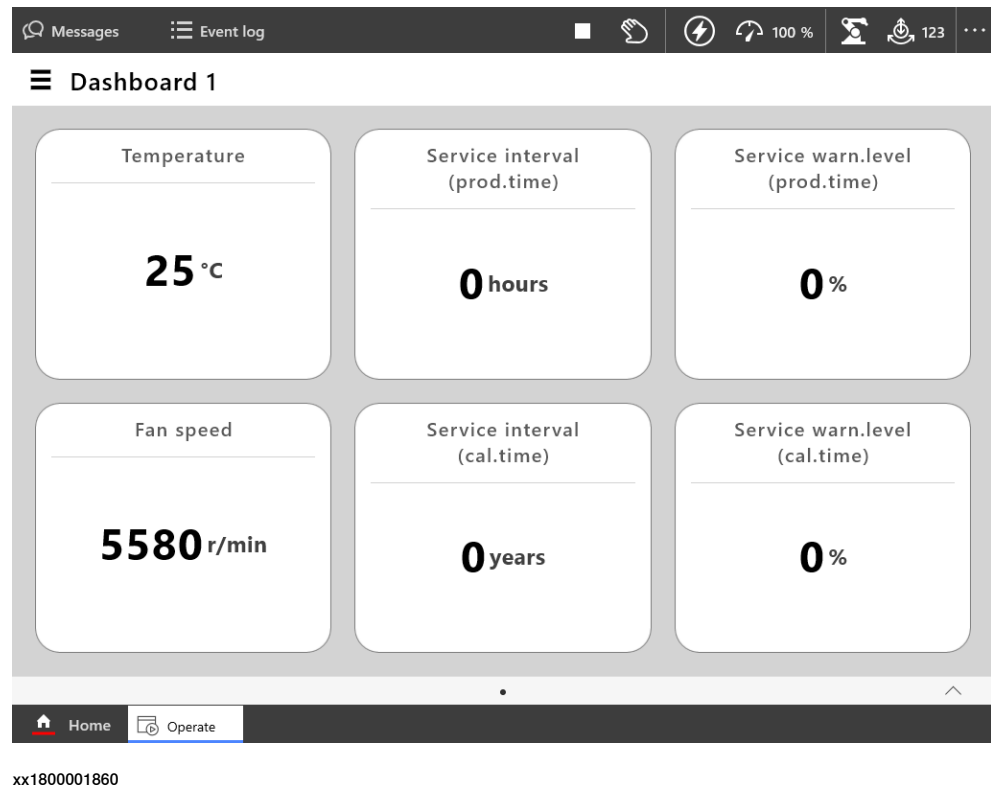
Non motion execution is reset after a reboot. If you intend to run the program in non motion mode, do not restart without checking the status of **Non motion execution**. Starting the program incorrectly may cause serious injury or death, or damages the robot or other equipment.

7.3 Managing Dashboards

Overview

Dashboards are used to display the controller parameters, RAPID data, live values from I/O signals, and so on.

The values are displayed through dashboards. The dashboards are made up of cards. Each card displays a value based on its defined properties. A dashboard can hold a maximum of 6 cards.



Defining cards

Use the following procedure to define a card:

- 1 On the start screen, tap **Operate**, and then select **Define Cards** from the menu.
- 2 Tap **Create New Card** on the context menu.
The **Add Card** page is displayed.
- 3 Type or select the values in the title, Label, Value, and Unit fields.
- 4 Tap **Apply**.

The new card is created and displayed on the **Define Cards** page.

Once a card is created you can associate the card to dashboards. For more details, see [Assigning cards to dashboard on page 241](#).

Continues on next page

Defining dashboards

Use the following procedure to define a dashboard:

- 1 On the start screen, tap **Operate**, and then select **Define Dashboards** from the menu.
- 2 Tap **Create New Dashboard** on the context menu.
The **Add Dashboard** page is displayed.
- 3 Type a name for the new dashboard in the **Title** field.
- 4 Tap **Apply**.

The new dashboard is created and displayed in the **Define Dashboards** and **Dashboards** page.

Once a dashboard is created you need to associate cards to it. For more details, see [Assigning cards to dashboard on page 241](#)

Assigning cards to dashboard

Use the following procedure to associate cards to a dashboard:

- 1 On the start screen, tap **Operate**, and then select **Define Dashboards** from the menu.
- 2 The **Define Dashboards** page is displayed. The current available dashboards are displayed on this page.
- 3 Tap on a dashboard for which cards needs to be associated.
The **Edit Dashboard** page is displayed.
- 4 Navigate to the **Associated Cards** section and tap **Associate**.
The **Select Card(s)** page is displayed with a list of available cards.
- 5 Select the cards that needs to be associated to the selected Dashboard.



Note

You can associate a maximum of 6 cards in each dashboard.

- 6 Tap **OK**.
The **Edit Dashboard** page is displayed with the list of selected dashboards.
- 7 Tap **Apply**.
The selected cards are associated with the Dashboard.

Assigning a card to multiple dashboards

You can assign a card to a multiple dashboards:

- 1 On the start screen, tap **Operate**, and then select **Define Cards** from the menu.
- 2 The **Define Cards** page is displayed. The current available cards are displayed on this page.
- 3 Tap on a card that needs to be associated with dashboards.
The **Edit Card** page is displayed.
- 4 Navigate to the **Associated Dashboards** section and tap **Associate**.
The **Select Dashboards** page is displayed with a list of available dashboards.

Continues on next page

7 Running in production

7.3 Managing Dashboards

Continued

- 5 Select the dashboards that needs to be associated with the selected card.



Note

You can associate a maximum of 6 cards to each dashboard. So if a dashboard already has 6 associated cards then you cannot associate the selected card to that dashboard.

- 6 Tap **OK**.

The **Edit Card** page is displayed with the list of associated dashboards.

- 7 Tap **Apply**.

The selected card is associated with the selected dashboards.

Deleting cards from dashboards

Use the following procedure to delete the associate cards from a dashboard:

- 1 On the start screen, tap **Operate**, and then select **Define Dashboards** from the menu.
- 2 The **Define Dashboards** page is displayed. The current available dashboards are displayed on this page.
- 3 Tap on a dashboard for which the associated cards needs to be deleted.
The **Edit Dashboard** page is displayed. The **Associated Cards** section displays the cards that are already associated to the selected dashboard.
- 4 Tap on the delete icon next to the card that you want remove from the selected dashboard.
The card is deleted from the **Associated Cards** section.
- 5 Tap **Apply**.
The changes are saved.

7.4 Managing positions

7.4.1 Introduction

Overview

A robot program usually contains programmed positions. Positions are instances of the data type `robtarg`. For details about creating a `robtarg` data instance, see [Creating new data instance on page 148](#). For more details about the RAPID variable `robtarg`, see *Technical reference manual - RAPID Instructions, Functions and Data types*.

The positions can be modified using the **Teach position** function where you jog the robot to a new position and save that position. A modified position value overwrites the original value. For more details, see [Teach position on page 244](#).

It is also possible to move the robot to a programmed position using the **GoTo** function. For more details, see [Go To position on page 249](#).



CAUTION

Changing programmed positions may significantly alter the robot's movement pattern.

Always make sure the changes are safe for both equipment and personnel.

7 Running in production

7.4.2 Teach position

7.4.2 Teach position

Introduction

When modifying instructions it is possible to jog directly to a new position and change the corresponding position argument of the instruction.

It is also possible to change the position values of the Program Data type `robtargt`.



Note

If a named position is modified, all the instructions using that position will be affected.



Note

The maximum movement or change in orientation, may be restricted by the system parameters (topic *Controller*, type *ModPos Settings*) in the system design. Please read your cell or plant documentation for details.

Prerequisite

Following are the prerequisite for modifying a programmed position:

- the system must be in manual mode
- the target position must have an initial value. For example:

```
CONST robtargt
p10:=[[515.00,0.00,712.00],[0.707107,0,0.707107,0],[0,0,0,0],
[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];CONST jointtarget
jpos10:=[[-0,-0,0,-0,-0,-0],
[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```



Note

The modified position values will normally be used when you restart the program. If the robot cannot use the values directly at start, a warning is displayed. Then the modified position will be used the next time than the position is used in the program.

Procedure

Teach position from the Program Editor main window

When modifying instructions, use the following procedure to jog directly to a new position and change the corresponding position argument of the instruction:

- 1 On the start screen, tap **Code**.
- 2 Tap the menu button and select **Program Editor**.
The RAPID program is displayed.
- 3 In the program click and select the position that needs to be updated.
- 4 Using joystick jog the robot to a new position.

Continues on next page

5 Tap Teach Position.

The **Teach Position** confirmation window is displayed.

6 Tap Teach.

The values of the selected position is updated with the current robot position values.

Teach Position from Program data

It is possible to change the position values by modifying the RAPID data type `robtarget`.

Use the following procedure to change the position values for the available positions:

1 On the start screen, tap Program Data.

The Program data types are displayed.

2 Select the data type `robtarget`.

The `robtarget` data instances (positions) are displayed.

3 Using joystick jog the robot to a new position.

4 Tap on the position that you want to modify.

The context menu is displayed.

5 Tap Teach Position.

A confirmation window is displayed.

6 Tap Teach.

The values of the selected position is updated with the current robot position values.



Note

Repeat steps 4 through 7 for the each position that you want to modify.

The programs that are created using RobotStudio may have array data position arguments. You cannot directly use the teach position function in such position arguments (`robtargets`). Use the following procedure to teach position for an array data `robtarget`:

1 Tap on an array data `robtarget`.

The elements available in the array are displayed.

2 Using joystick jog the robot to a new position.

3 Select the element.

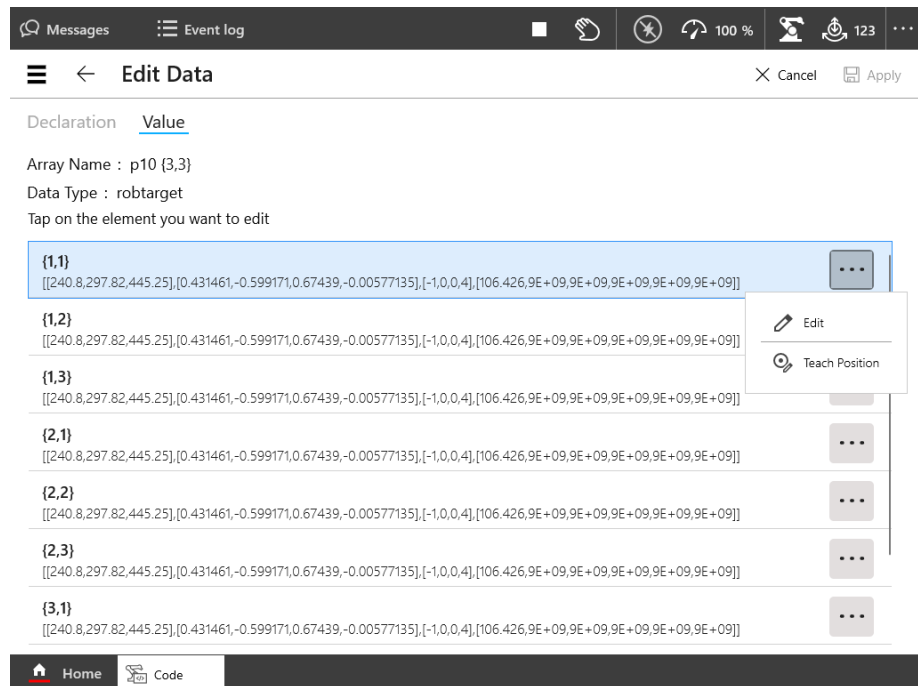
The context menu is displayed.

Continues on next page

7 Running in production

7.4.2 Teach position

Continued



xx2000001285

4 Tap Teach Position.

The Teach Position confirmation window is displayed.

5 Tap Teach.

The position values of the selected array element is updated with the current robot position values.

Teach position for an array data position argument

The programs that are created using RobotStudio may have position argument with array data. This section provides information regarding teaching position for a particular position array data item:

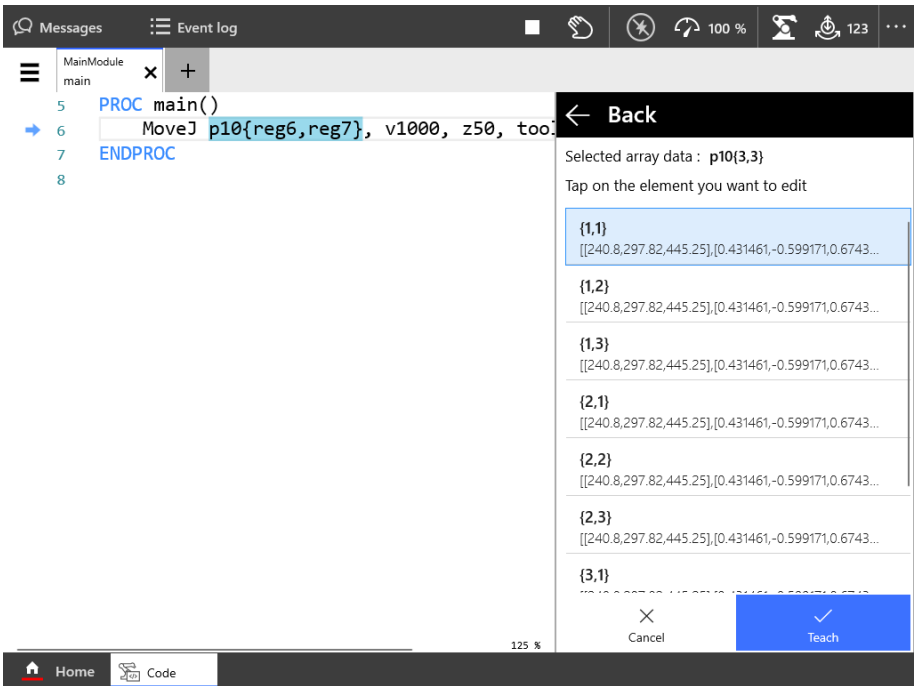
1 Tap on the position argument that has array data and that needs to be edited.

2 Tap Teach Position.

The elements available in the array are displayed.

3 Select the element for which the position needs to be modified and using joystick jog the robot to a new position.

Continues on next page



xx2000001296

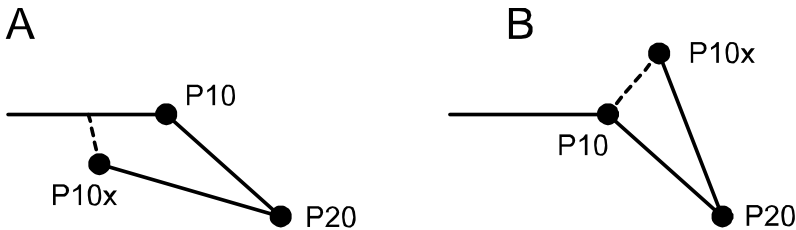
- 4 Tap Teach.
The Teach Position confirmation window is displayed.
- 5 Tap Teach.
The values of the selected position is updated with the current robot position values.

Examples

The following examples show how the planned path will be affected for different types of movement when modifying positions.

Linear movement

In example A the robot is stopped on path before reaching the position P10. The robot is jogged off path to the new position (P10x) and the position P10 is modified. In example B the robot is stopped on path in position P10. The robot is jogged off path to the new position (P10x) and the position P10 is modified.



xx0800000175

In both examples, when restarting the program the robot continues from the new P10 (that is now the same as P10x) directly to P20 without returning to the previous planned path (via the old P10).

Continues on next page

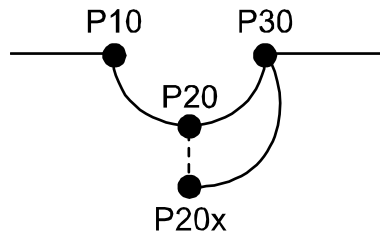
7 Running in production

7.4.2 Teach position

Continued

Circular movement

In this example the robot is stopped on path in position P20 (circle point) and then jogged to the new position P20x. The position P20 is modified.



xx0800000176

In single robot systems, the system is in unsynchronized mode: When restarting the program the robot continues directly from the new P20 (that is now the same as P20x) to P30 without returning to the previous planned path (via the old P20). The new planned path from P20 (P20x) to P30 is calculated using these two positions and position P10.

7.4.3 Go To position

Moving the robot to a programmed position

It is possible to automatically move the robot to a programmed position.



DANGER

When moving the robot automatically, the robot arm may move without warning. Make sure no personnel are in the safeguarded space and that no objects are in the way between the current position and the programmed position.

Use the following procedure to move the robot automatically to a programmed position:

- 1 On the start screen, tap **Jog**.
- 2 Tap the Menu button and select **Go To**.
A list of programmed positions (RobTargets and JointTargets) is displayed.
- 3 From the list, select a programmed position to which you want to move the robot.



Note

If you have many programmed positions you can use a filter to narrow down the visible positions. For details, see [Filtering data on page 69](#).

- 4 Press and hold the three-position enabling device and then tap and hold the **Go To** button.

The robot is moved from the current position to the selected programmed position.



Note

For collaborative robots, after selecting a programmed position, you need to just press and hold the **Go To** button to move the robot.

7 Running in production

7.4.4 Working with displacements and offsets

7.4.4 Working with displacements and offsets

About displacements

Sometimes, the same path is to be performed at several places on the same object, or on several work pieces located next to each other. To avoid having to reprogram all positions each time a displacement coordinate system can be defined.

This coordinate system can also be used in conjunction with searches, to compensate for differences in the positions of the individual parts.

The displacement coordinate system is defined based on the object coordinate system.

The displacement coordinate system is described in section [Coordinate systems for jogging on page 90](#).

Select displacement method

Depending on how, when, and how often you want to use displacements, the best method may vary.

Moving a work object

Moving a work object is suitable when you do not need to move or displace the work object very often.

This is detailed in section [Defining the work object coordinate system on page 175](#).

Displace a work object

A work object consists of a user frame and a object frame. You can move one or both of these frames. If you move both frames, then the whole work object is moved. It can be useful to displace the object frame from the user frame for instance when using one fixture for several work objects. Then you can keep the user frame and displace the object frame for the work objects.

See procedure *How to define object frame* in section [Defining the work object coordinate system on page 175](#).

Displace and rotate a work object

You may want to displace and rotate the object frame from the user frame if the displacement is not in just x, y, and z.

To displace in x, y, and z, you can use the same method as above. To rotate the work object, follow the procedure in section [Editing the work object data on page 178](#).

About offsets

Sometimes it is easier to define a position as an offset from a given position. If, for example, you know the exact dimensions of a work object, it will only be necessary to jog to one position.

The offset is programmed with the displacement distance in x, y, and z direction, in relation to the work object. For instance:

```
MoveL Offs(p10, 100, 50, 0), v50...
```

Define the offset for the position with the following expressions:

- 1 Original position / starting point

Continues on next page

- 2 Displacement in x direction
- 3 Displacement in y direction
- 4 Displacement in z direction

Examples

This example shows the move instructions with offsets to move the robot in a square (clockwise), starting at p10, with a 100 mm displacement in x and y.

```
MoveL p10, v50...  
MoveL Offs(p10, 100, 0, 0), v50...  
MoveL Offs(p10, 100, 100, 0), v50...  
MoveL Offs(p10, 0, 100, 0), v50...  
MoveL p10, v50...
```

Creating position offsets

Use the following procedure to change a position to become an offset position.

- 1 In the home screen open **Code**.
- 2 Tap the Menu button and select **Program Editor**.
- 3 Tap and select a position instruction to edit.
- 4 Tap **Modify Instruction**.
- 5 Tap **Expression Editor**.
- 6 Tap the **Functions** tab and select **offs**.
- 7 Tap to select each expression or argument, **<EXP>** or **<ARG>**, and then tap any of the desired available data or functions.
You can tap on **Change Data Type** and the other buttons available on the expression editor to fine tune the selected instruction.
- 8 Tap **Apply**.
The changes are saved.

7 Running in production

7.5 Detaching and attaching a FlexPendant

7.5 Detaching and attaching a FlexPendant

Introduction

With the option *Hot swappable FlexPendant [3018-1]* it is possible to detach and attach the FlexPendant from an OmniCore controller in automatic mode, without interrupting the ongoing process. The FlexPendant can also be detached and attached in manual reduced speed mode (requires RobotWare 7.19 or later).

Detaching the FlexPendant in manual mode on a controller running RobotWare 7.18 or earlier will result in an emergency stop.



Note

Detaching the FlexPendant is possible only if the logged in user has the **Detach the FlexPendant** grant.



CAUTION

Before detaching the FlexPendant, another emergency stop shall be available. How to configure emergency stops is described in the product manual for the controller, see [References on page 10](#).



CAUTION

When FlexPendant is detached, the status of other actuating controls shall be indicated clearly, for example, power on, fault detected, automatic operation. See [Configuring status indicators for operating mode on page 254](#).



CAUTION

A FlexPendant that is not connected to the robot must be stored out of sight so that it cannot be mistaken for being in use.



CAUTION

The FlexPendant connector shall only be used to connect the FlexPendant. For details, see the product manual for the respective robot controller.

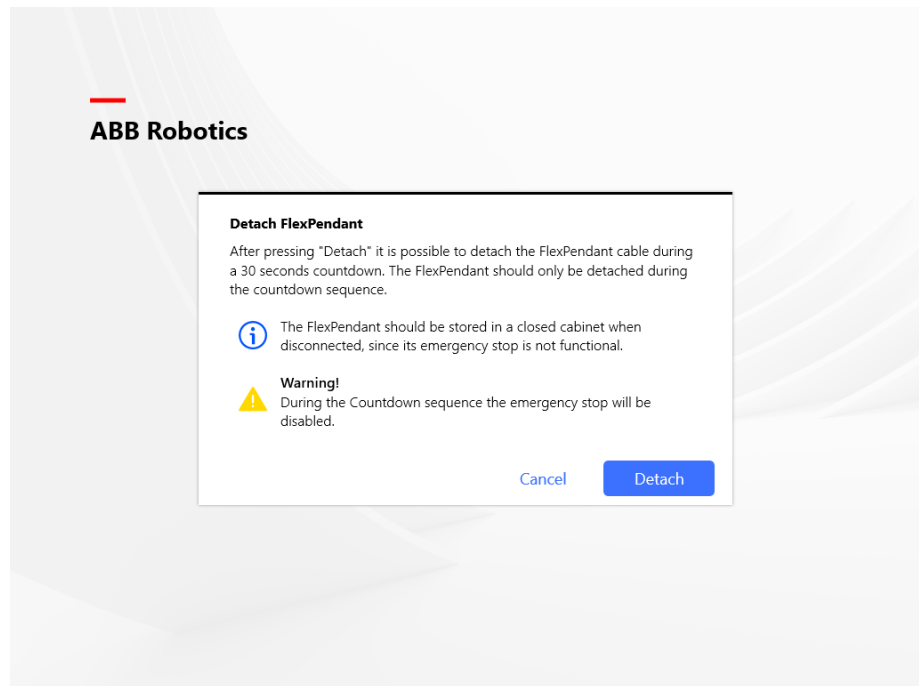
Detaching the FlexPendant

Use the following procedure to detach the FlexPendant.

- 1 On the status bar, tap the **QuickSet** button.
- 2 Tap the **Logout/Restart** tab.
- 3 In the **FlexPendant** section, tap **Detach FlexPendant**.

The **Detach FlexPendant** window is displayed.

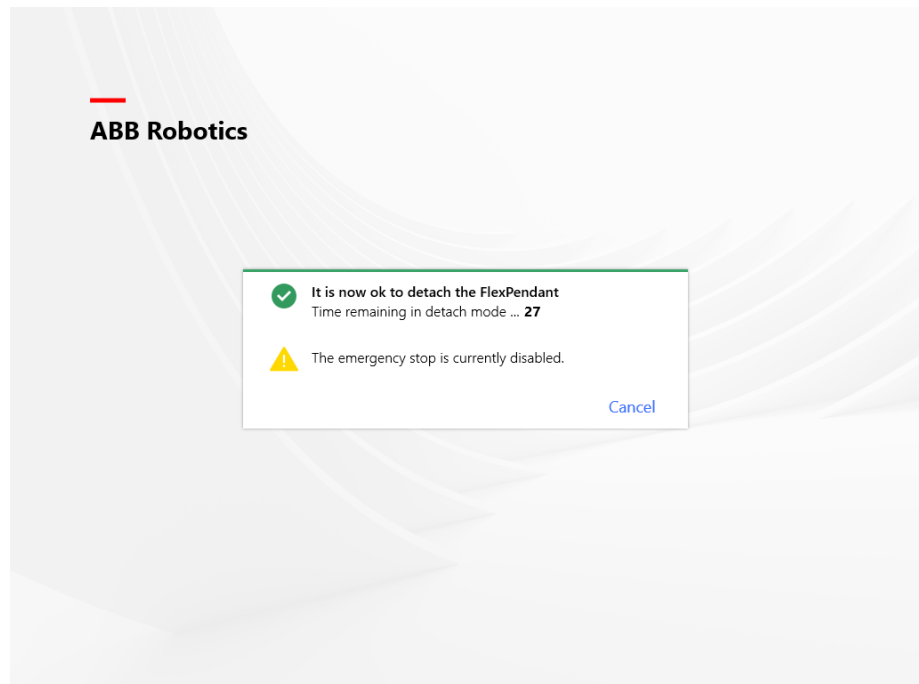
Continues on next page



xx1900000403

4 Tap **Detach**.

A popup window with 30 seconds countdown timer is displayed.



xx1900000404

5 When the countdown is progressing, detach the FlexPendant.

When detached, the FlexPendant will shut down.

Continues on next page

7 Running in production

7.5 Detaching and attaching a FlexPendant

Continued



Note

If the FlexPendant is not detached within 30 seconds, the process for detach of the FlexPendant is aborted.



WARNING

If the FlexPendant is detached after the 30 seconds countdown has passed, the controller will enter emergency stop state.

Attaching the FlexPendant



CAUTION

Always inspect the connector for dirt or damage before attaching. Clean or replace any damaged parts.

Attach the connector to the controller and tighten the locking ring or screws.



CAUTION

Make sure that the emergency stop device is not pressed in before attaching the FlexPendant.

Configuring status indicators for operating mode

When the FlexPendant is detached, the operating mode shall be visualized by other means. For example, the operating mode signals can be used.

| Signal name | Description | Bit value condition | From - To |
|-------------|---------------------------|---------------------|---------------------------------|
| ManRS | Manual reduced speed mode | 1 = Selected | Main computer - internal status |
| ManFS | Manual high speed mode | 1 = Selected | Main computer - internal status |
| Auto | Auto mode | 1 = Selected | Main computer - internal status |

8 Handling inputs and outputs, I/O

8.1 Introduction

Overview

The inputs and outputs signals can be managed using the I/O application or RobotStudio.

8 Handling inputs and outputs, I/O

8.2 Viewing signal lists

8.2 Viewing signal lists

Overview

The **Signals** page is used to view the input and output signals and their values.

Procedure

Use the following procedure to view a list of I/O signals:

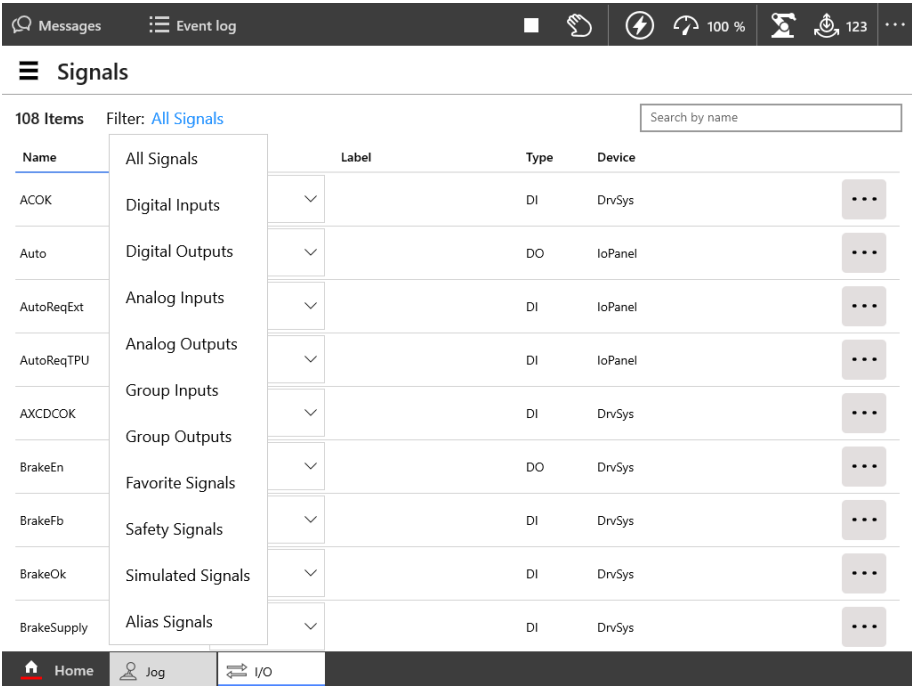
- 1 On the start screen, tap **I/O**, and then select **Signals** from the menu.
- 2 The **Signals** page is displayed.



Note

By default all the signals are displayed.

- 3 Tap on the **Filter** list.



xx1800001856

- 4 Select the category of the signal that you want to view.
The signals of the selected category are displayed.



Tip

You can use the search box to quickly filter and view a particular signal.

8.3 Setting signals as favorite signals

Overview

You can set few signals from various signal categories as favorite signals if you use it frequently.



Note

The signals can be set as favorite signals only when the operating mode is **Manual**.

Procedure

Use the following procedure to set signals as favorite signals:

- 1 Change the operating mode to **Manual**.
- 2 On the start screen, tap **I/O**, and then select **Favorite Signals** from the menu.
- 3 The **Favorite Signals** page is displayed and it displays all the signals.



Note

You can use the **Filter** list to display a selected category of signals.

- 4 Use the check box to select the signals that need to be set as favorite signals.
- 5 Tap **Apply**.

The selected signals are saved as favorite signals.




Tip

The favorite signals are listed under the **Favorite Signals** category on the **Signals** page. To view a selected category of signals, use the procedure [Viewing signal lists on page 256](#).

8.4 Simulating the signals and changing the signal values

Simulating the signals

A signal can be simulated.

**Note**

You cannot simulate a signal if its access level is set to **Read Only**.

MessagesEvent log

Signals

9 ItemsFilter: Favorite Signals

Search by name

| Name | Value | Label | Type | Device |
|----------------------|-------|-------|------|--------------------------|
| AI_All_Simulated | 7 | | AI | ABB_Scalable_IO_Analog |
| AI_Default_Simulated | 0 | | AI | ABB_Scalable_IO_Analog |
| AO_Default_Simulated | 0 | | AO | ABB_Scalable_IO_Analo... |
| BrakeOk | 0 | | DI | DrvSys |
| c1CntToEncStr | 1 | | DO | |
| c1Connected | 0 | | DO | |
| c1NullSpeed | 0 | | DO | |
| c1ObjectsInQ | 7 | | GO | |

HomeI/O


12:15

xx2100001551


Use the following procedure to simulate a signal:

- 1 On the start screen, tap **I/O**, and then select **Signals** from the menu.
- 2 Use the **Filter** list or use the search box to navigate to a particular signal.
- 3 Tap on the **Simulate** button available for the signal.

The signal is changed to a simulated signal.

**Note**

Once a signal is simulated the **Remove Simulation** button is displayed for the signal.

**Note**

If there is a cross connection created between two signals, changing the value of one signal automatically changes the value of the corresponding cross connected signal.

Continues on next page

Changing the signal values

The value of a signal can be changed.



Note

You can change the value of an input signal only if it is a simulated signal.

Use the following procedure to change the value of a signal:

- 1 On the start screen, tap **I/O**, and then select **Signals** from the menu.
- 2 Use the **Filter** list or use the search box to navigate to a particular signal.
- 3 To change the value of a signal:
 - For a digital signal, in the **Value** column tap on the toggle button and change the current value. **0 - False** and **1 - True** are the allowed values.
 - For an analog signal, in the **Value** column, tap on the number pad icon, type a value using the number pad within the allowed range, and then tap **OK**.

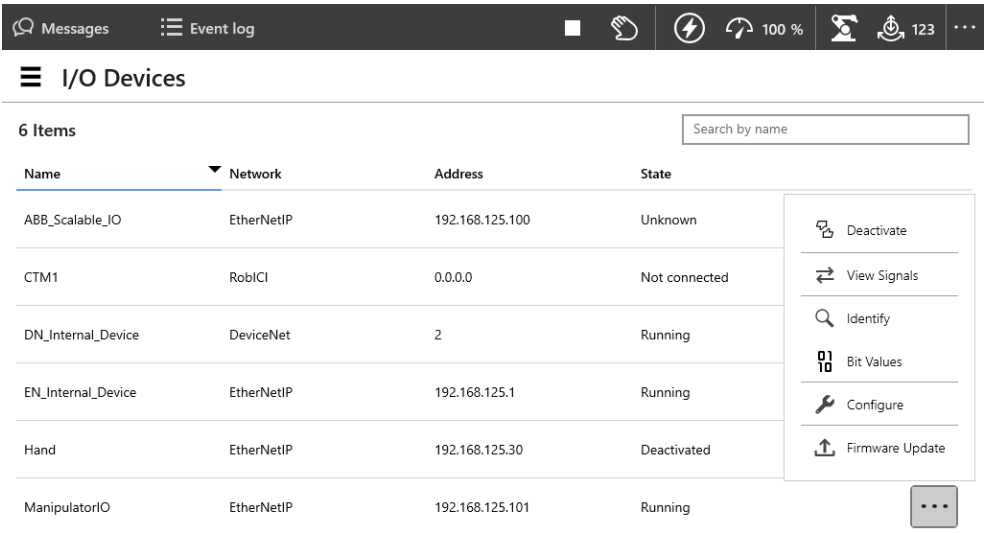
8 Handling inputs and outputs, I/O

8.5 I/O devices



8.5 I/O devices

Managing I/O devices


You can manage I/O devices from the **I/O Devices** page. To manage an I/O device, use the context menu that is displayed for the devices.



Following are the actions that you can perform using the context sensitive menu:

| Action | Output |
|----------------------------|--|
| Activate/Deactivate | Activates or deactivates the selected I/O device. |
| Bit Values | Displays the input and output bit values of the selected I/O device. |
| Signals | Displays the signals associated with the selected I/O device. |
| Actions | <div>Displays the MAC ID of the selected I/O device.</div> <div> Note</div> <div>Actions - Identify option is available only for the network type EtherNet IP.</div> |
| Configure | <div>Allows you to configure the selected I/O device.</div> <div> Note</div> <div>The Configure option is available only for the network type EtherNet IP.</div> |

Continues on next page

| Action | Output |
|-----------------|---|
| Firmware Update | <p>Allows you to upgrade the firmware.</p> <div> Note Actions - Firmware option is available only for the network type EtherNet IP.</div> |

Upgrading firmware

You can upgrade the firmware of I/O device of type EtherNet IP.

Use the following procedure to upgrade the firmware.

- 1 On the start screen, tap **I/O**, and then select **I/O Devices** from the menu.
- 2 Select **Firmware Update** in the context menu for an EtherNet IP I/O device.
The **I/O Modernization** window is displayed with the details of the firmware upgrade.



Note

The Firmware upgrade option is available only for the network type EtherNet IP.

- 3 Tap **Browse** and select the firmware from the location it is stored.
- 4 Tap **Upgrade**.
The firmware for the selected EtherNet IP I/O device is upgraded.

This page is intentionally left blank

9 Handling the event log

9.1 Introduction

Overview

Open the event log to:

- view the current event logs.
- study specific event log in detail.
- manage the log entries.

The log can be printed from RobotStudio.

9 Handling the event log

9.2 Accessing the event log

9.2 Accessing the event log

Open and close an event log

Use the following procedure to open the event log:

- 1 Tap the **Event log** icon on the status bar.

The **Event Log** page is displayed. It displays the event log list. By default the event logs from the Common domain (category) are displayed.

- 2 Select a category in the list to filter the event logs.
- 3 Tap a log entry.

The selected event log message is displayed.



Note

Tap the **Back** button to go back to the **Event Log** list.

9.3 Saving log entries

Overview

You should save log entries when:

- you need to clear the log but want to keep the current entries to be viewed later.
- you want to send log entries to support the solution of a problem.
- you want to keep log entries for future reference.



Note

The log can keep up to 20 entries in a category and up to 1000 entries in the all the events list. When the buffer is full the oldest entries are overwritten. There is no way to retrieve these overwritten log entries.

Save all log entries

Use the following procedure to save all the log entries.

- 1 Tap the **Event log** icon on the status bar.
- 2 Select **Save Log** from the context menu.
The **Save** window is displayed.
- 3 Select a location and select a folder to store the file.
- 4 In the **File name** text box, type a name for the file.
- 5 Tap **Save**.

The log entries are saved.

9 Handling the event log

9.4 Clearing the log entries

9.4 Clearing the log entries

Why should you clear the log entries?

Logs can be cleared or deleted to increase the available disk space. Deleting the log entries is often a good way to trace the faults since you can remove old or insignificant log entries not related to the problem that you are trying to solve.

Clear the log entries of a specific domain

Use the following procedure to clear all the log entries.

- 1 Tap the **Event log** icon on the status bar.
- 2 In the Domain list select the domain for which you want to clear the log entries.
- 3 On the command bar, select **Clear Domain** from the context menu.
A confirmation window is displayed.
- 4 Tap **OK**.
The log entries for the selected domain are deleted.

Clear all the log entries

Use the following procedure to clear all the log entries.

- 1 Tap the **Event log** icon on the status bar.
- 2 Select **Clear all** from the context menu.
A confirmation window is displayed.
- 3 Tap **OK**.
The log entries in all the domains are deleted.

10 Install, update, restart, and other configuration

10.1 Introduction

Overview

This chapter provides information about installing a RobotWare system, updating the FlexPendant applications, backup the system, and other configuration.

10 Install, update, restart, and other configuration

10.2 Install RobotWare system

10.2 Install RobotWare system

Overview

For information about how to install a new RobotWare system, or how to update an existing RobotWare system, see *Operating manual - Integrator's guide OmniCore*.

10.3 Restart

What happens with my RobotWare system?

The RobotWare system will be stopped.

All system parameters and programs will be saved to the controller mass memory.

During the restart process the system state will be resumed. Static and semistatic tasks will be started. Programs can be started from the point they were stopped.

Restarting this way will activate any configuration changes entered using RobotStudio.

Restart the controller

This section describes how to restart the controller and preserve the current system/parameters and RAPID programs.

- 1 On the start screen, tap **Settings**.
- 2 At the bottom of the **Settings** screen tap **Restart Controller**.

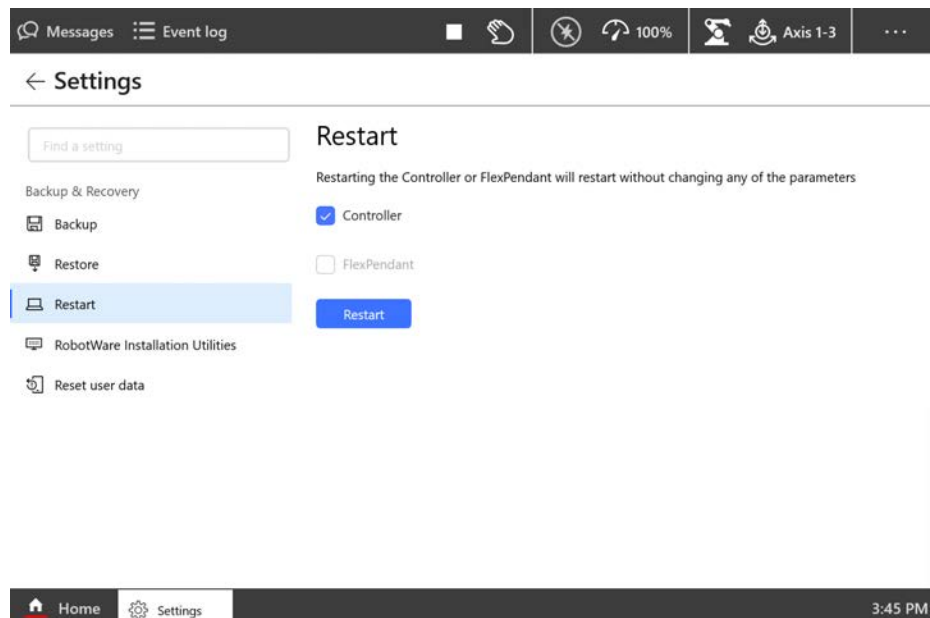
The controller is restarted.

Restart the controller and/or FlexPendant

This section describes how to restart the Controller or Flexpendant and preserve the current system/parameters and RAPID programs.

- 1 On the start screen, tap **Settings**, and then tap **Backup & Recovery**.
- 2 On the **Backup & Recovery** menu, tap **Restart**.

The restart dialog is displayed.



xx1900000123

Select **Controller** or **FlexPendant**.

- 3 Tap **Restart**.

The Controller and/or Flexpendant is restarted.

Continues on next page

10 Install, update, restart, and other configuration

10.3 Restart

Continued



Note

When the Controller is restarted the Flexpendant is also automatically restarted.

Restart Controller through QuickSet menu

Use the following procedure to restart the controller using the QuickSet menu:

- 1 Tap on the **QuickSet** button and select the **Logout/Restart** tab.
The Logout/Restart page is displayed.
- 2 In the **Controller** section tap **Restart**.
The **Restart** window is displayed.
- 3 Tap **OK**.
The controller is restarted.

Restart FlexPendant through QuickSet menu

There are certain cases under which only the FlexPendant needs to be restarted. For example, restart the FlexPendant while troubleshooting.



Note

If the FlexPendant freezes during operation, press the reset button to restart the FlexPendant. For more details, see [Reset button on page 22](#).

Use the following procedure to restart the FlexPendant using the QuickSet menu:

- 1 Tap on the **QuickSet** button and select **Logout/Restart**.
The **Logout/Restart** page is displayed.
- 2 In the **FlexPendant** section tap **Restart FlexPendant**.
The **Restart** window is displayed.
- 3 Tap **OK**.
The FlexPendant is restarted.

10.4 Backup and restore systems

10.4.1 What is saved on backup?

Introduction to backups

When creating a backup, or restoring a previously made backup, not all data is included.

What is saved?

The backup function saves all system parameters, system modules, and program modules in a context.

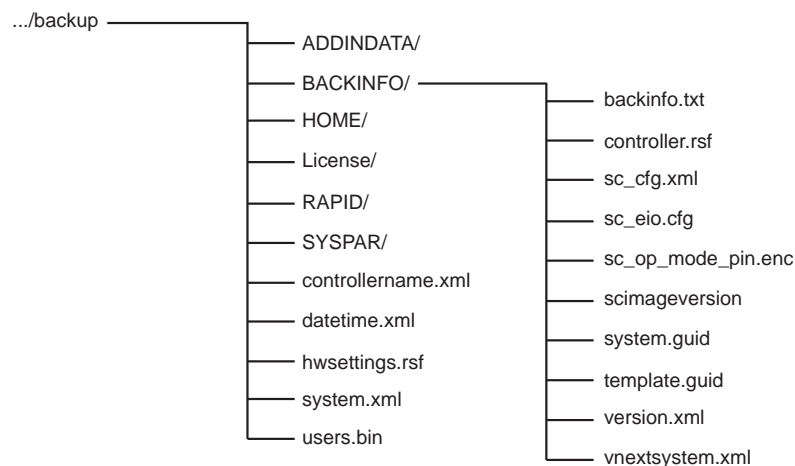
The data is saved in a directory specified by the user.

The directory is divided into the following five subdirectories:

- ADDINDATA
- BACKINFO
- HOME
- License
- RAPID
- SYSPAR

The following files are also saved in the ../backup (root directory):

- controllername.xml
- datetime.xml
- hwsettings.rsf contains the serial number of the controller and the controller type.
- system.xml contains information about installed SW and optional features selected in the RobotWare System.
- users.bin



xx1900000318

ADDINDATA

ADDINDATA contains a number of sub-directories used for the RobotWare add-ins.

Continues on next page

10 Install, update, restart, and other configuration

10.4.1 What is saved on backup?

Continued

BACKINFO

BACKINFO consists of the files *backinfo.txt*, *program.id*, and *system.guid*, *template.guid*, and *version.xml*.

- *backinfo.txt* is used when the system is restored. The backup must **never** be edited by the user!
- *controller.rsf* contains information about installed software and selected optional features in the backed up system.
- *sc_cfg.xml* is the safety configuration.
- *sc_eio.cfg* is the eio configuration connected to the safety configuration.
- *sc_op_mode_pin.enc* contains information on the pin code used for locking the operating mode.
- *scimageversion* contains information on which safety version the backup was created.
- *system.guid* is used to identify the unique system the backup was taken from.
- *system.guid* and/or *template.guid* is used in the restore to check that the backup is loaded to the correct system. If the *system.guid* and/or *template.guid* do not match, the user will be informed.
- *vnextsystem.xml* holds configuration data specific to the FlexPendant, such as programmable keys and start app on system event.

HOME

HOME is a copy of the files in the HOME directory.

License

License directory contains a copy of the license file that are used and valid only on the robot controller where the backup has been created. The license files can be used when creating a new RobotWare system in RobotStudio Installation Manager.

RAPID

RAPID consists of a subdirectory for each configured task. Each task has one directory for program modules and one for system modules. The module directory will keep all installed modules. More information on loading modules and programs is described in *Technical reference manual - System parameters*.

SYSPAR

SYSPAR contains the configuration files (that is, system parameters).

Limitations

It is not possible to move backups between controllers.

What is not saved?

A few things are not saved on backup, but can be useful to save separately:

- The current value of a PERS object in a installed module is not stored in a backup.
- Certificates and SW installed in RobotWare System are not included in the backup.

Continues on next page

Related information

Technical reference manual - System parameters.

Operating manual - RobotStudio.

10.4.2 Backup the system

When do I need this?

We recommend performing a backup:

- *Before* installing new RobotWare.
- *Before* making any major changes to instructions and/or parameters to make it possible to return to the previous setting.
- *After* making any major changes to instructions and/or parameters and testing the new settings to retain the new successful setting.

Backup the system

This section describes how to backup the system.

- 1 On the start screen, tap **Settings**, and then **Backup & Recovery**.
- 2 On the **Backup & Recovery** menu, tap **Backup**.

The backup dialog is displayed.

The screenshot shows the 'Backup' dialog within the 'Settings' app. On the left is a sidebar with options: Backup & Recovery, Backup (selected), Restore, Restart, System diagnostics, Start installer, and Reset user data. The main area is titled 'Backup' and contains the following fields: 'Backup Name' with the value 'Backup_20190117', 'Location' with the value '/user/BACKUP' and a 'Browse' button, and a preview 'Backup of this Robot will be created at' showing the path '/user/BACKUP/Backup_20190117'. Below these is an 'Options' section with a checkbox 'Backup to tar file'. At the bottom is a blue 'Backup' button. A note at the bottom of the dialog states: 'All modules and system parameters will be stored in a backup folder.'

xx1900000152

- 3 In the **Backup Name** field, type a name for the backup.



Note

By default, a backup name is suggested, but you can edit it. While renaming, ensure that the name does not start with a space.
If the backup name starts with a space, a warning dialog appears.



Note

The backup name can have different formats depending on the parameter setup. The name may contain the date, or the system name etc. See information about backups in section **Type System Input** in *Technical reference manual - System parameters* for detailed information.

Continues on next page

- 4 In the **Location** field, tap the **Browse** button and select a location for saving the backup.
- 5 Select the **Backup to tar file** checkbox if the backup need to be created as a TAR file.



Note

If the backup location is USB or another memory device, the **Backup to tar file** checkbox is selected by default and you cannot change it.

- 6 Tap **Backup**.

A backup of the modules and system parameters is created in the selected location.

Disable or queue backup

Backing up the system during production can interfere with the RAPID execution. To avoid that a backup is taken during critical process steps or sensitive robot movements, a system input (*Disable Backup*, type *System Input*) can be set during these critical steps. When the critical steps are done, the input should be reset to allow backups again.

If needed, the backup can be queued while *Disable Backup* is set, using the system parameter *General RAPID*, with action value *QueueBackup* set to *TRUE*. Then the backup will be queued until the signal is reset.

Disable Backup and *QueueBackup* are described in *Technical reference manual - System parameters*.

10.4.3 Important when performing backups

BACKUP directory

A local default backup directory, `BACKUP`, is automatically created by the system. We recommend using this directory for saving backups.

Never change the name of the `BACKUP` directory.

Never change the name of the actual backup to `BACKUP`, since this will cause interference with this directory.

When is backup possible?

A backup of a system can be performed during program execution, with a few limitations:

- Do not create backups while performing critical process steps or sensitive robot movements. This may affect the accuracy and performance of the movement. To make sure that no backup is requested, use a system input with the action value `Disable Backup` (type *System Input*). When the critical steps are done, the input should be reset to allow backups again.

If needed, the backup can be queued while `Disable Backup` is set, using the system parameter *General RAPID*, with action value `QueueBackup` set to `TRUE`. Then the backup will be queued until the signal is reset.

(Queueing functionality available from RobotWare 7.1.)

`Disable Backup` and `QueueBackup` are described in *Technical reference manual - System parameters*.

The system input signal can be set from `RAPID` for the parts of the code that are critical for disturbances.

What happens during backup?

During the backup process, background tasks continue to execute.

Large data amount

Since the `HOME` directory is included in the backup, large files contained in this folder will make the backup larger. To avoid this situation, you should either clean the `HOME` directory on regular basis removing the unnecessary files, or keep large files in some other location.

Faults during backup

If a fault occurs during the backup, for example full disk or power failure, the whole current backup is deleted to make sure that only valid fully saved backups are present on the disk.

10.4.4 Restore the system

When do I need this?

We recommend performing a restore:

- If you suspect that the program file is corrupt.
- If any changes made to the instructions and/or parameters settings did not prove successful, and you want to return to the previous settings.

During the restore, all system parameters are replaced and all the modules from the backup directory are loaded.

The Home directory is copied back to the new system's HOME directory during the restart.

Verifying the backup file before restore

While restoring a backup file, the following verifications are performed and if verification fails, a warning message is displayed:

- Verify whether the backup is modified.
 - For example, whether the files are edited, added, or removed from the backup. An event log is added for every change of the backup.
- Verify whether the backup comes from another system.
- Verify whether the backup has a different passphrase.



Note

Even though an event log is added for every failed verification, the backup can still be restored.

Procedure

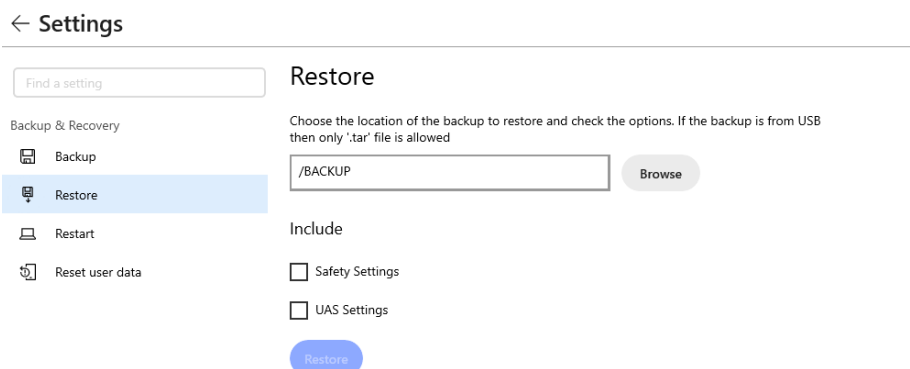
- 1 On the start screen, tap **Settings**, and then **Backup & Recovery**.
- 2 On the **Backup & Recovery** menu, tap **Restore**.
The **Restore** page is displayed.

Continues on next page

10 Install, update, restart, and other configuration

10.4.4 Restore the system

Continued



← Settings

Find a setting

Backup & Recovery

- Backup
- Restore**
- Restart
- Reset user data

Restore

Choose the location of the backup to restore and check the options. If the backup is from USB then only '.tar' file is allowed

/BACKUP Browse

Include

- ☐ Safety Settings
- ☐ UAS Settings

Restore

xx1900000153

- 3 Browse for the location where the backup is stored and select the backup file.



Note

If the backup location is USB or another memory device, the backup file is allowed only in the TAR format.

- 4 If required, select the following backup options:

- **Safety Settings**

Safety data will be included in the backup.

- **UAS Settings**

UAS data will be included in the backup.



Note

Not applicable for virtual controllers.



Note

Only allowed for users with the grant *Manage UAS settings*.

- 5 Tap **Restore**.

The restore is performed, and the system is restarted.

10.5 Reset user data

Overview

It is recommended to perform a user data reset:

- when you want to reset the RAPID program in the system.
- when you want to restore the system to its original state by resetting RAPID program and system parameters (except the topic *SIO/Communication*).
- when you want to remove the user configured safety settings and load the default safety settings.

Resetting RAPID program

After restart, the system state will be resumed except for manually loaded programs and modules. Static and semistatic tasks are started from the beginning, not from the state they had when the system was stopped.

Modules will be installed and loaded in accordance with the current configuration. System parameters will not be affected.

Resetting RAPID program and system parameters

After restart, the system returns to the default empty state and any changes done to system parameters and RAPID programs will be lost. Instead, system parameters and other settings are read from the originally installed system on delivery (except the topic *SIO/Communication*).



CAUTION

When the controller is restarted, the content in the **TEMP** folder in the controller is also emptied. To avoid problems, move any important content before resetting these parameters and restarting the controller.

Resetting safety settings

The user defined safety settings will be replaced by an empty default configuration and all safety information stored in the system will be erased. After the restart, a new safety configuration will be needed, synchronization between the safety controller and the robot controller needs to be performed and the safety configuration needs to be locked.

Procedure

Use the following procedure to reset the user data:

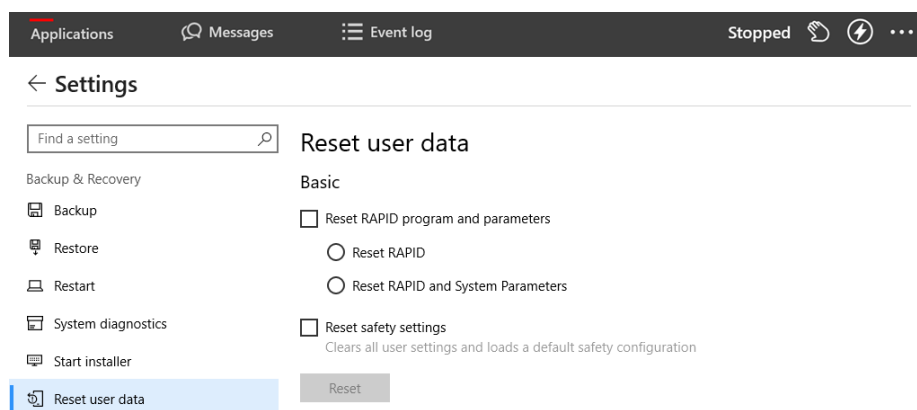
- 1 On the start screen, tap **Settings**, and then **Backup & Recovery**.
- 2 On the **Backup & Recovery** menu, tap **Reset user data**.
The **Reset user data** dialog is displayed.

Continues on next page

10 Install, update, restart, and other configuration

10.5 Reset user data

Continued



xx1900000124

3 Select one of the following options:

- **Reset RAPID:** To reset the RAPID data of the loaded system.
- **Reset RAPID and System Parameters:** To restore the system to its original state by resetting the RAPID and system parameters (except the topic *SIO/Communication*).



Note

If this option is selected, the content in the **TEMP** folder will be emptied when the controller is restarted.

- **Reset safety setting:** To remove the user configured safety settings and load the default safety settings.

4 Tap **Reset**.

A confirmation message is displayed.

5 Tap **OK**.

The controller is restarted and the system is updated according to the selected reset data settings.

10.6 FlexPendant logs

Overview

This option is used to save the FlexPendant logs for troubleshooting analysis.

Procedure

Use the following procedure to save the FlexPendant log:

- 1 On the start screen, tap **Settings**, and then select **Diagnostics** from the menu.
- 2 On the left sidebar tap **FlexPendant logs**.
The **FlexPendant logs** page is displayed.
- 3 If required, in the **File Name** field edit the name of the file.
- 4 To change the storage path, in the **Location** field tap **Browse** and select the required folder.
- 5 Tap **Save**.

The FlexPendant log is saved in the selected folder.

10 Install, update, restart, and other configuration

10.7 Update FlexPendant

10.7 Update FlexPendant

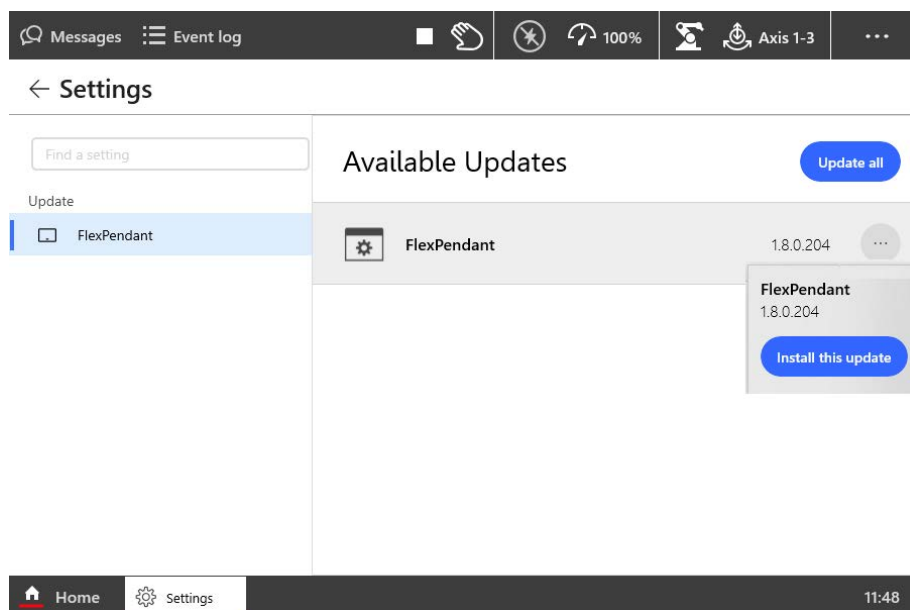
Updating the FlexPendant applications

If the installed system has updates for the FlexPendant applications, you can update it.

Updating the FlexPendant applications from Settings

Use the following procedure to update the applications from **Settings**:

- 1 On the start screen, tap **Settings**, and then select **Update** from the menu.
- 2 If there are updates, the update files are displayed in the **Available Updates** section.
- 3 Tap on the menu button next to an update.



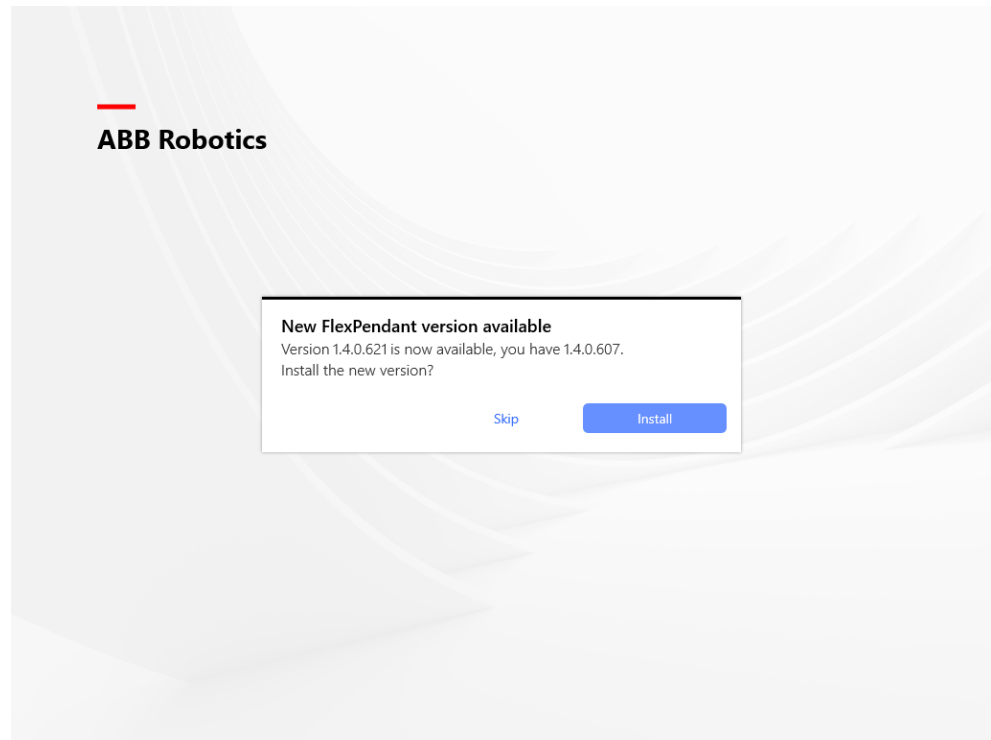
xx2000002046

- 4 Tap **Install this update**.
The selected update is installed and the FlexPendant is restarted.

Continues on next page

Updating the Flexpendant applications after a restart

After a restart if the FlexPendant detects a new version of applications in the system, it displays the following message. Tap **Install** to update the FlexPendant applications.



xx2000002047

10.8 Connection log

Overview

This option is used to save the current connection state of the connectivity module.



Note

Connection log is available only for Connected Services Gateway 3G and Wi-Fi but not for Wired.

Procedure

Use the following procedure to save the current connection state of the connectivity module:

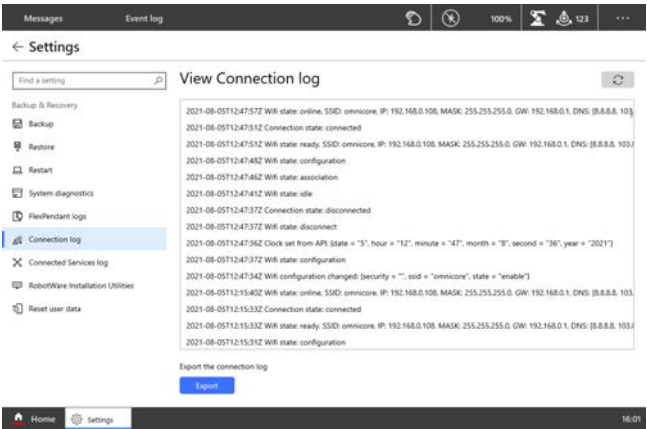
- 1 On the start screen, tap **Settings**, and then select **Diagnostics** from the menu.
- 2 On the left sidebar tap **Connection log**.

The **View Connection log** page is displayed with the available connection logs on a window.



Note

Tap on the refresh button to view the updated log.



xx1900000976

- 3 Tap **Export**.

The **Save Connection log** page is displayed.

- 4 If required, in the **File Name** field edit the name of the file.
- 5 To change the storage path, in the **Location** field tap **Browse** and select the required path.
- 6 Tap **Save**.

The current connection state of the connectivity module is saved in the selected location.

10.9 Connected Services log

Overview

This option is used to save the Connected Services log.



Note

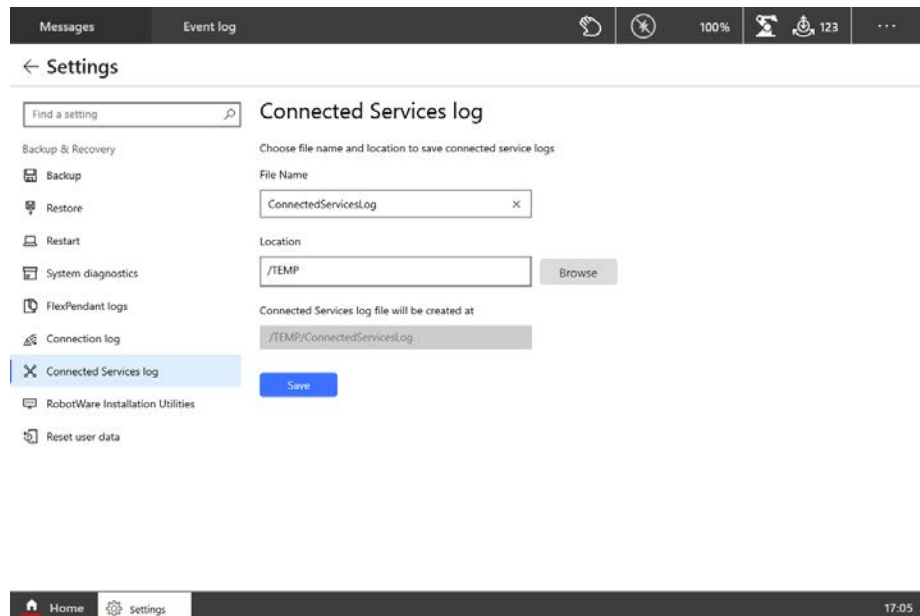
Connected Services log is available for all types of connectivity.

Procedure

Use the following procedure to save the Connected Services log:

- 1 On the start screen, tap **Settings**, and then select **Diagnostics** from the menu.
- 2 On the left sidebar tap **Connected Services log**.

The **Connected Services log** page is displayed.



xx2100001566

- 3 If required, in the **File Name** field edit the name of the file.
- 4 To change the storage path, in the **Location** field tap **Browse** and select the required path.
- 5 Tap **Save**.

The Connected Services log is saved in ZIP format in the selected path and a confirmation message is displayed.

- 6 Tap **OK**.

10 Install, update, restart, and other configuration

10.10 Creating a system diagnostics file

10.10 Creating a system diagnostics file

When do I need this?

The diagnostic file can be useful when contacting ABB technical support personnel for troubleshooting. The diagnostic file contains the setup and a number of test results from your system. For more information, see *Technical reference manual - Event logs for RobotWare 7*, section *Instructions, how to correct faults - Filling an error report*.

Creating a system diagnostics file

- 1 On the start screen, tap **Settings**, and then **Diagnostics**.
- 2 On the left sidebar tap **System diagnostics**.

The **System diagnostics** page is displayed.

The screenshot shows the 'System diagnostics' settings page. At the top, there is a navigation bar with 'Messages', 'Event log', and a 'Settings' button. Below this, the 'System diagnostics' page is displayed. It has a left sidebar with 'System diagnostics' and 'FlexPendant logs'. The main area contains a search bar, a description of the system diagnostics file, and fields for 'File Name' (set to 'SystemDiagnosticsData') and 'Location' (set to '/TEMP'). A 'Browse' button is next to the location field. Below these fields, it shows the path where the file will be created: '/TEMP/SystemDiagnosticsData'. A 'Save' button is at the bottom. The bottom of the screen shows a 'Home' button, a 'Settings' button, and a status bar with the time '4:23 PM'.

xx1900000151

Type a name for the file in the **File Name** section.

- 3 In the **Location** section, tap on the **Browse** button and select a folder to save the file.
- 4 Tap **Save**

A system diagnostics file is saved in the selected folder.

10.11 Shutting down

Shut down the controller

This section describes how to shut down the system and turn off power.

- 1 Stop all running programs.
- 2 Shut down the system in one of the following ways:
 - Shut down the system using the main power switch (On/Off switch).
 - Shut down the system from RobotStudio. In the **Controller** tab, right-click the controller and select **Restart > Shutdown**.



Note

When shutting down from RobotStudio, wait for 30 seconds after shutting down the system. It is then safe to turn off the main power switch.

- 3 The controller is shut down.
- 4 If you want to protect the FlexPendant you can unplug it and store it elsewhere when the system has shut down. See [Detaching and attaching a FlexPendant on page 252](#).

This page is intentionally left blank

Index

A

- applications
 - overview, 45
- applications for FlexPendant, 45
- approach points, 160
- apps, 21
- axes
 - illustration, 93
- Axis Calibration
 - service routine, 200

B

- backup, 271, 276
 - default file path, 61
 - directory, 276
 - system, 274
- backward button, 22
- backward execution
 - about, 192
 - limitations, 192
- battery shutdown
 - service routine, 199

C

- calculation result, 163
- calender time counter, 202
- Calibrate** application, 48
- calibrating
 - Axis Calibration, 200
 - CalPendulum, 201
 - LoadIdentify, 203
- calibration
 - status, 84
 - touchscreen, 75
- CalPendulum
 - service routine, 201
- characters
 - entering, 65
 - international, 65
- Code** application, 45
- commutation
 - service routine, 228
- connector, 20
- controller
 - restart, 269
- controller ID
 - configuring, 60
- coordinate systems
 - overview, 90
- cursor
 - about, 144

D

- data instance, 148
 - create, 148
- data types
 - creating new, 148
 - editing, 150
- date and time, 54
- debugging, 145
- default paths
 - setting, 61
- detaching FlexPendant, 252
- disable backup, 275

- disconnecting FlexPendant, 252
- displacements
 - about, 250
 - work object, 173, 175

E

- elongator points
 - define, 163
- emergency stop device
 - FlexPendant, 21
- enabling device, 20–21, 23
 - using, 188
- error messages, 67
- Essential App Package* [3120-2], 45
- execution tab, 144
- expressions
 - offs, 250
 - positions, 250

F

- File Explorer, 48
- files
 - programs, 116
- filtering
 - about, 69
 - data types, 69
 - files, 69
 - programs, 69
- FlexPendant
 - connecting, disconnecting, 252
 - hardware buttons, 22
 - how to hold, 24, 64
 - left-hander, 24
 - main parts, 20
 - overview, 20
 - screen, 33
- forward button, 22

H

- hard buttons, 22
- hold-to-run, 24, 233
 - using, 188

I

- I/O
 - changing values, 258
 - simulating, 258
- I/O, inputs and outputs, 256
- I/O application, 47
- incremental movement
 - definition, 96, 99
 - size settings, 99
- insertion point, change, 65
- instances
 - data types, 148
- instructions
 - backward execution, 192
 - copying and pasting, 139
 - copying arguments, 139
 - delete, 140
 - editing arguments, 135
 - handling of, 134
 - running from a specific, 190
- international characters, 65
- IsBrakeCheckActive, 216, 222
- ISO 8859-1, 115

J

- Jog** application, 46
- jogging
 - about, 87
 - additional axes, 100
 - axes in independent mode, 100
 - non calibrated mechanical units, 100
 - restrictions, 100
 - world zones, 100
- joystick, 20
 - using, 21
- joystick directions
 - illustration, 93

L

- lead-through, 101
- left-handed, 64
- Limited App Package* [3120-1], 45
- load, 102
- LoadIdentify
 - service routine, 203
- loads
 - identifying, 203
- locking operating mode, 79
- log off, 73
- log on, 72

M

- mechanical unit
 - selecting, 95
- modifying positions, 244
 - data instances, 150
 - overview, 243
- ModPos, 244
- module, 121
 - change type, 124
 - create, 121
 - delete, 124
 - file format, 121
 - load, 121
 - rename, 123
 - save, 122
- modules
 - handling of, 121
- Motion mode
 - selecting, 95
- motion pointer, MP, 193
- Motion Pointer, MP
 - about, 144
- multitasking programs
 - about, 234
 - loading, running and stopping, 234
 - task selection, 236
 - viewing, 236

O

- offsets
 - about, 250
 - creating, 251
 - description, 250
- Operate** application, 47
- operation time counter, 202
- operator messages, 33
- operator window, 34

P

- path

- returning to, 237
- path return region, 237
- payload, 102
 - create, 182
 - declaration settings, 183
- payload data
 - change, 185
 - edit, 185
- payload definition, 185
- payloads
 - deleting, 187
 - identifying, 203
 - selecting, 95
- personalizing, 51
- PIN code
 - permanent PIN code, 80
 - temporary PIN code, 79
- positions
 - about, 97
 - exact, 97
 - modifying, 243–244
 - moving to, 249
 - offset, 250
 - reading, 97
 - tuning, 243
- program, 116
 - delete, 119
 - editor, 116
 - files, 116
 - new, 117
 - rename, 118
 - save, 117
- program data
 - editing, 150
- Program Data** application, 45
- program directory, 116
- program execution start button, 23
- programmable buttons
 - editing, 23, 62
- programmable keys
 - editing, 23, 62
- Program Package* [3151-1], 45
- program pointer, PP, 193
- Program Pointer, PP
 - about, 144
- programs
 - default file path, 61
 - handling of, 116
 - multitasking, 234
 - starting, 230
 - step by step execution, 192
 - stopping, 233

Q

- queue backup, 275

R

- RAPID program
 - reset, 279
- reset button
 - location, 21
 - using, 22
- resolvers
 - about, 97
- restart, 269
 - controller, 269
 - QuickSet, 270

restore
 default file path, 61
 system, 277
 revolution counters
 about, 97
 battery shutdown, 199
 RobotStudio
 overview, 26
 routine, 126
 change declaration, 131
 copy, 130
 create, 126
 define parameters, 128
 delete, 132
 move, 132
 rename, 132
 routines
 handling of, 126
 running a specific, 191
 running service routines, 195
 run button, 23

S

safety setting
 reset, 279
 screenshot, 68
 search settings, 70
 service routines
 Axis Calibration, 200
 BatteryShutDown, 199
 CalPendulum, 201
 commutation, 228
 LoadIdentify, 203
 running, 195
 ServiceInfo, 202
 Settings
 search, 70
 Settings application, 46
 signals
 changing values, 258
 simulating, 258
 viewing, 256
 SIS, Service Information System
 counters, 202
 service routine, 202
 SkipTaskExec, 227
 SmartGripper
 configuring, 111–112
 SMB
 battery shutdown, 199
 soft keyboard, 65
 start button, 22
 stationary tool, 168
 step backward button, 22
 step by step execution, 192
 step forward button, 22
 stop button, 22
 symbols, 115
 system
 backup, 274
 restore, 277
 system diagnostics, 286
 create, 286
 system parameter
 reset, 279

T

targets
 modifying, 243–244
 moving to, 249
 tuning, 243
 tasks
 loading program to, 234
 normal, static, semistatic, 234
 setting up, 234
 starting and stopping, 234
 teach pendant
 detach, attach, 252
 teach position, 244
 three-position enabling device, 21, 23
 thumb button
 using, 24
 tool
 coordinate system, 168
 create, 156
 declaration settings, 157
 definition, 152
 edit definition, 165
 tool center point
 about, 156
 calculation result, 163
 define, 162
 defining, 163
 definition, 154
 measure, 164
 TCP, 156
 working area variations, 163
 tool data, 164
 edit, 164
 tool frame
 define, 159
 methods, 159
 reorientation test, 163
 tool orientation, 163
 setting, 95
 tool orientation, definition, 95
 tools
 aligning, 106
 deleting, 167
 identifying loads, 203
 make stationary, 168
 selecting, 95
 touchscreen
 calibration, 75
 TPU
 connecting, disconnecting, 252
 tuning
 positions, 243
 targets, 243

U

user data
 reset, 279
 UTF-8, 115

V

viewing messages in programs, 34
 view settings
 configuring, 57–59

W

work object
 coordinate system, 175

- create, 171
- declaration settings, 171
- definition, 170
- work object data
 - edit, 178
- work objects
 - defining, 173
 - deleting, 179
 - displacements, 173, 175
 - selecting, 95
- wrist optimization, 226
- write access
 - granting, 71
 - message, 67
 - rejecting, 71

**ABB AB****Robotics & Discrete Automation**

S-721 68 VÄSTERÅS, Sweden

Telephone +46 10-732 50 00

ABB AS**Robotics & Discrete Automation**

Nordlysvegen 7, N-4340 BRYNE, Norway

Box 265, N-4349 BRYNE, Norway

Telephone: +47 22 87 2000

ABB Engineering (Shanghai) Ltd.

Robotics & Discrete Automation

No. 4528 Kangxin Highway

PuDong New District

SHANGHAI 201315, China

Telephone: +86 21 6105 6666

ABB Inc.**Robotics & Discrete Automation**

1250 Brown Road

Auburn Hills, MI 48326

USA

Telephone: +1 248 391 9000

abb.com/robotics