ROBOTICS

Operating manual

IRC5 Integrator's guide

Trace back information:
Workspace 24A version a8
Checked in 2024-02-26
Skribenta version 5.5.019

Operating manual
IRC5 Integrator's guide
RobotWare 6.15.06

Document ID: 3HAC050940-001
Revision: P

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
Network security ... 13
Open source and 3rd party components ... 14
Product documentation ... 15

1 Welcome to IRC5 17

1.1 About this section ... 17
1.2 The IRC5 controller ... 18
1.3 The FlexPendant ... 19
1.4 RobotStudio Online ... 26
1.5 RobotStudio ... 28
1.6 When to use different jogging devices ... 29
1.7 Buttons and ports on the controller ... 32

2 Get started 35

2.1 About this chapter ... 35
2.2 Connections ... 36
2.2.1 Connecting a FlexPendant ... 36
2.2.2 Using the hot plug option ... 38
2.2.3 Set up the network connection ... 41
2.3 Action scenarios ... 43
2.3.1 About action scenarios ... 43
2.3.2 System start ... 44
2.3.3 Working with inputs and outputs ... 46
2.3.4 Backup and restore ... 47
2.3.5 Upgrading ... 48
2.3.6 Installing RobotWare add-ins ... 49

3 Programming 51

3.1 Before you start programming ... 51
3.2 The structure of a RAPID application ... 52
3.3 Programming ... 54
3.3.1 Handling of programs ... 54
3.3.2 Handling of modules ... 57
3.3.3 Handling of routines ... 60
3.4 Data types ... 65
3.4.1 Creating new data instance ... 65
3.5 Tools ... 67
3.5.1 Creating a tool ... 67
3.5.2 Defining the tool frame ... 70
3.5.3 Editing the tool data ... 74
3.5.4 Setup for stationary tools ... 77
3.6 Work objects ... 79
3.6.1 Creating a work object ... 79
3.6.2 Defining the work object coordinate system ... 80
3.6.3 Editing the work object data ... 84
3.6.4 Editing the work object declaration ... 85
3.7 Payloads ... 86
3.7.1 Creating a payload ... 86
3.7.2 Editing the payload data ... 88
3.8 Advanced programming ... 90
3.8.1 Mirroring a program, module, or routine ... 90
3.8.2 Editing instruction expressions and declarations ... 92
3.8.3 Hiding declarations in program code ... 95
3.8.4 Deleting programs from memory ... 96
3.8.5 Deleting programs from hard disk ... 98

3.8.6 Activating mechanical units 99
3.9 Hide RAPID arguments 100

4 Handling inputs and outputs, I/O 103

4.1 Basic procedures 103
4.1.1 Configuring I/O 103
4.1.2 Deactivating and activating I/O units 104
4.1.3 Alias I/O signals 105
4.2 Safety signals 107
4.2.1 Safety I/O signals 107

5 Systems 111

5.1 What is a system? 111
5.2 Memory and file handling 112
5.2.1 What is “the memory”? 112
5.2.2 File handling 113
5.2.3 Setting default paths 115
5.3 Restart procedures 116
5.3.1 Restart overview 116
5.3.2 Using the Boot Application 117
5.3.3 Restart and use the current system 121
5.3.4 Restart and start boot application 122
5.3.5 Restart and reset RAPID 123
5.3.6 Restart and reset system 124
5.3.7 Restart and revert to last auto saved 125
5.3.8 Reflashing firmware 126
5.4 Installed Systems 127
5.4.1 Managing Installed Systems 127
5.4.2 RobotWare startup error 129
5.5 Backup and restore systems 130
5.5.1 What is saved on backup? 130
5.5.2 Backup the system 133
5.5.3 Important when performing backups 135
5.5.4 Restore the system 137
5.6 Diagnostic files 140
5.6.1 Creating a diagnostic file 140
5.7 System configuration 141
5.7.1 Configuring system parameters 141

6 RobotWare installation concept 145

6.1 Introduction 145
6.2 Working with the repository 148
6.2.1 Recommended working procedure 148
6.2.2 Setting up the repository 150
6.2.3 Creating a base system 153
6.2.4 Creating an application system 155
6.2.5 Defining controllers 157
6.2.6 Creating an installation package 159
6.2.7 Creating an update package 162
6.2.8 Repository folders and file structure 164
6.3 Deploying installation and update packages 170
6.3.1 Overview 170
6.3.2 Installing a RobotWare system using Boot Application 171
6.3.3 Updating a RobotWare system 180
6.4 The recovery disk function 183
6.5 Limitations 185

7 Calibrating 187
7.1 Robot calibration ... 187
7.2 How to check if the robot needs calibration ... 188
7.3 Loading calibration data using the FlexPendant ... 189
7.4 Editing motor calibration offset ... 190
7.5 Serial measurement board memory ... 191
7.6 4 points XZ calibration ... 194

8 Descriptions of terms and concepts 197
8.1 What is the tool center point? ... 197
8.2 What is a work object? ... 199
8.3 What is a coordinate system? ... 200
8.4 What is mirroring? ... 206

9 Cyber security for IRC5 robot networks 213
9.1 Introduction ... 213
9.2 Network architecture and communication ... 215
9.3 Security analysis ... 218
9.4 IRC5 User Authorization System ... 219
9.5 Security policy ... 220
9.5.1 Introduction ... 220
9.5.2 General security requirements ... 221
9.5.3 ABB Robotics IRC5 product specific requirements ... 225
9.6 IRC5 application protocols ... 228
9.7 Network connections on the IRC5 main computer ... 232

Index 239

This page is intentionally left blank

Overview of this manual

About this manual

This manual contains advanced instructions for IRC5 based robot systems using a FlexPendant.

The daily operations are described in *Operating manual - IRC5 with FlexPendant*. This manual describes aspects for commissioning, as well as advanced instructions that are not used by the operator during ordinary operation.

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

Usage

This manual should be used during commissioning and when making changes to the robot system that are outside the scope of everyday operations. This manual needs to be complemented with *Operating manual - IRC5 with FlexPendant* that describes more common operations.

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

Who should read this manual?

This manual is intended for:

- integrators
- product technicians
- service technicians
- robot programmers

Prerequisites

The reader should:

- Be familiar with the concepts described in *Operating manual - Getting started, IRC5 and RobotStudio*.
- Be trained in robot operation.

References

Operating manual - IRC5 with FlexPendant	*3HAC050941-001*
Operating manual - Getting started, IRC5 and RobotStudio	*3HAC027097-001*
Product manual - IRC5 IRC5 with main computer DSQC 639.	*3HAC021313-001*
Product manual - IRC5 IRC5 with main computer DSQC1000 or later.	*3HAC047136-001*

Continues on next page

Continued

Product manual - IRC5 Panel Mounted Controller IRC5 with main computer DSQC 639.	*3HAC027707-001*
Product manual - IRC5 Panel Mounted Controller IRC5 with main computer DSQC1000 or later.	*3HAC047137-001*
Product manual - IRC5 Compact IRC5 with main computer DSQC 639.	*3HAC035738-001*
Product manual - IRC5 Compact IRC5 with main computer DSQC1000 or later.	*3HAC047138-001*
Operating manual - RobotStudio	*3HAC032104-001*
Operating manual - Service Information System	*3HAC050944-001*
Operating manual - Troubleshooting IRC5	*3HAC020738-001*
Technical reference manual - System parameters	*3HAC050948-001*
Technical reference manual - RAPID Overview	*3HAC050947-001*
Technical reference manual - RAPID Instructions, Functions and Data types	*3HAC050917-001*
Application manual - Additional axes and standalone controller	*3HAC051016-001*
Application manual - MultiMove	*3HAC050961-001*
Application manual - Controller software IRC5	*3HAC050798-001*
Application manual - DeviceNet Master/Slave	*3HAC050992-001*
Application manual - DeviceNet Anybus Slave	*3HAC050993-001*
Application manual - EtherNet/IP Anybus Adapter	*3HAC050997-001*
Application manual - EtherNet/IP Scanner/Adapter	*3HAC050998-001*
Application manual - PROFIBUS Anybus Device	*3HAC050965-001*
Application manual - PROFIBUS Controller	*3HAC050966-001*
Application manual - PROFIenergy Device	*3HAC050967-001*
Application manual - PROFINET Anybus Device	*3HAC050968-001*
Application manual - PROFINET Controller/Device	*3HAC065546-001*
Application manual - Functional safety and SafeMove2	*3HAC052610-001*
Operating manual - Calibration Pendulum	*3HAC16578-1*

Revisions

Revision	Description
-	Released with RobotWare 6.03.
A	Released with RobotWare 6.04. • Added section *Cyber security for IRC5 robot networks on page 213*.
B	Released with RobotWare 6.05. • Added the section *Using the `PayLoadsInWristCoords` parameter on page 88*. • Added the section *RobotWare installation concept on page 145*. • Added the section *Renaming a controller on page 119*. • Updated descriptions of stops.

Continues on next page

Revision	Description
C	Released with RobotWare 6.06. • Updated the section *Loading calibration data using the FlexPendant on page 189*. • Updated the section *Working with the repository on page 148*. • Updated the procedures in the section *Installing a RobotWare system using Boot Application on page 171*. • Updated the section *Backup and restore systems on page 130*. • Updated the section *4 points XZ calibration on page 194*. • Added the new section *RobotWare startup error on page 129*.
D	Released with RobotWare 6.07. • Updated the section *When is backup possible? on page 135* • Updated the section *Procedure - If the Controller settings does not matches with the settings in controller manifest file on page 177*. • Added information about Conveyor tracking module in the section, *Cyber security for IRC5 robot networks on page 213*. • Added SFTP to *IRC5 application protocols on page 228*. • Safety section restructured. • Clarified the limitations for Isolated Lan 3 in the section *Isolated LAN 3 or LAN 3 as part of the private network on page 233*.
E	Released with RobotWare 6.08. • Updated the section *Create system using installation package in boot server mode on page 161*. • Updated the section *What is saved on backup? on page 130* • Updated the section *The recovery disk function on page 183* with information regarding the Write Disk function.
F	Released with RobotWare 6.09. • Section *Handling of modules on page 57* updated. Screen shot for saving a module removed. • Updated information about update packages, see *Updating a RobotWare system on page 180*.
G	Released with RobotWare 6.10.01. "Cyber security" replaced by "Cybersecurity" in entire manual.
H	Released with RobotWare 6.11. • The safety information is moved to the product manuals for the controller and the manipulator. • Updated information about queueing backups. • Information added regarding UdpUc in section *IRC5 application protocols on page 228*. • Updated NOTE regarding IP addresses in section *Set up the network connection on page 41*. • Information added regarding installation media for Add-Ins added in section *RobotWare installation concept on page 145*. • Section *Creating an update package on page 162* updated with information regarding the downgrade option.
J	Released with RobotWare 6.12. • Minor corrections in section *System start on page 44*. • Note regarding anonymous FTP added in section *FTP on page 230*.
K	Released with RobotWare 6.13.02. • Added information regarding Integrated Vision in the section *IRC5 application protocols on page 228*.

Continues on next page

Continued

Revision	Description
L	Released with RobotWare 6.14.01. • Added information about a new version of the FlexPendant. • Removed information about T10.
M	Released with RobotWare 6.15. • Information about communication and application protocols updated in section *Network architecture and communication on page 215* and *IRC5 application protocols on page 228*.
N	Released with RobotWare 6.15.03. • Added information about TuneMaster in section *Network architecture and communication on page 215*.
P	Released with RobotWare 6.15.06. • Updated the section *IRC5 application protocols on page 228*. • 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.

Network security

Network security

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

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

Open source and 3rd party components

Open source and 3rd party components

ABB products use software provided by third parties, including open source software. The following copyright statements and licenses apply to various components that are distributed inside the ABB software. Each ABB product does not necessarily use all of the listed third party software components. Licensee must fully agree and comply with these license terms or the user is not entitled to use the product. Start using the ABB software means accepting also referred license terms. The third party license terms apply only to the respective software to which the license pertains, and the third party license terms do not apply to ABB products. With regard to programs provided under the GNU general public license and the GNU lesser general public license licensor will provide licensee on demand, a machine-readable copy of the corresponding source code. This offer is valid for a period of three years after delivery of the product.

ABB software is licensed under the ABB end user license agreement, which is provided separately.

RobotWare

For RobotWare, there is license information in the folder **licenses** in the RobotWare distribution package.

OpenSSL

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

This product includes software written by Tim Hudson (tjh@cryptsoft.com).

CTM

For OleOS, the Linux based operating system used on the conveyor tracking module (CTM), a list of copyright statements and licenses is available in the file **/etc/licenses.txt** located on the CTM board and accessible via the console port or by downloading the file over SFTP.

For the CTM application, a list of copyright statements and licenses is available in the file **/opt/ABB.com/ctm/licenses.txt** located on the CTM board and accessible via the console port or by downloading the file over SFTP.

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

Continued

- 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 Welcome to IRC5

1.1 About this section

Overview

This section presents an overview of the FlexPendant, the IRC5 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.

1.2 The IRC5 controller

The IRC5 controller

The IRC5 controller contains all functions needed to move and control the robot.

The standard IRC5 controller consists of a single cabinet. The controller is also available in a compact version, *IRC5 Compact*, and it can also be integrated in an external cabinet, *Panel Mounted Controller*.

When running more than one robot with one controller (MultiMove option), an extra drive module must be added for each additional robot. However, a single control module is used.

Related information

Product manual - IRC5, IRC5 of design M2004.

Product manual - IRC5, IRC5 of design 14.

Product manual - IRC5 Panel Mounted Controller, IRC5 of design M2004.

Product manual - IRC5 Panel Mounted Controller, IRC5 of design 14.

Product manual - IRC5 Compact, IRC5 of design M2004.

Product manual - IRC5 Compact, IRC5 of design 14.

Application manual - MultiMove.

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.

The hot plug button option makes it possible to disconnect the FlexPendant in automatic mode and continue running without it.

The FlexPendant is available in different versions, as the hardware has been updated over the years. The exact appearance on the graphics might therefore differ slightly from reality.

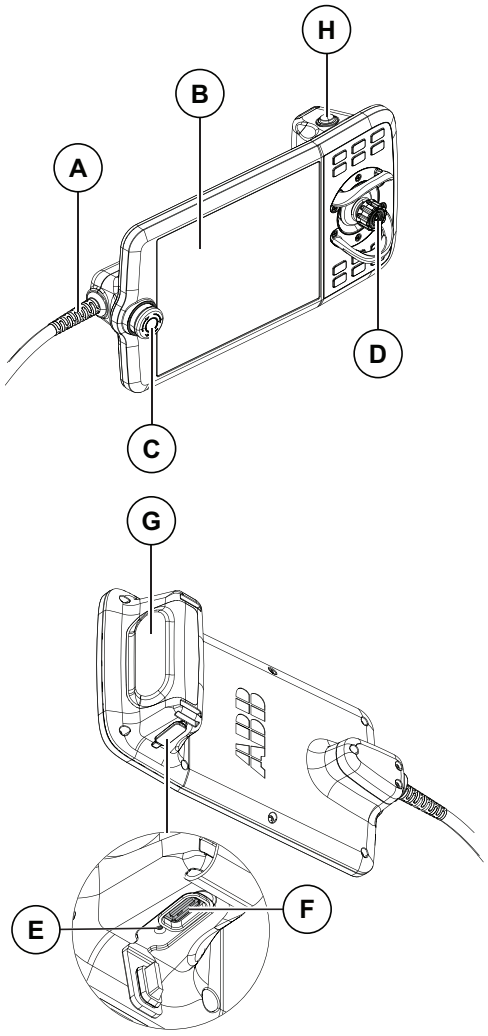
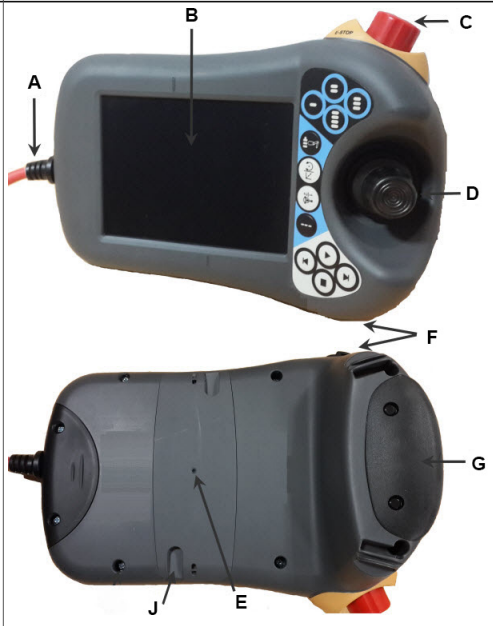
Note

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

Continues on next page

Main parts

These are the main parts of the FlexPendant.

FlexPendant with emergency stop button at the connector	FlexPendant with emergency stop button at the outer edge
 xx2200002054	 xx1400001636

A	Connector
B	Touchscreen
C	Emergency stop button
D	Joystick
E	Reset button
F	USB port
G	Three-position enabling device
H	Thumb button (Not available on all versions of FlexPendant.)
J	Stylus pen (Not available on all versions of FlexPendant.)

Continues on next page

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.

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. The USB memory is displayed as drive */USB:Removable* in dialogs and FlexPendant Explorer.

> **Note**
>
> Close the protective cap on the USB port when not used.

Stylus pen

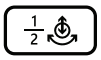
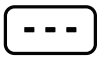
The stylus pen included with the FlexPendant is located on the back. Pull the small handle to release the pen.

Use the stylus pen to tap on the touch screen when using the FlexPendant. Do not use screw drivers or other sharp objects.

(Not available on all versions of FlexPendant.)

Hard buttons

The following hard buttons are available on the FlexPendant.

Button	Description
	Programmable keys, 1 - 4.
	Select mechanical unit.
	Toggle motion mode, reorient or linear.
	Toggle motion mode, axis 1-3 or axis 4-6.
	Toggle increments.
	**Step BACKWARD** button. Executes one instruction backward as button is pressed.
	**START** button. Starts program execution.
	**Step FORWARD** button. Executes one instruction forward as button is pressed.
	**STOP** button. Stops program execution.

Continues on next page

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.

Thumb button

The thumb button is only available on the FlexPendant with emergency stop located at the connector.

The thumb button is used for hold-to-run.

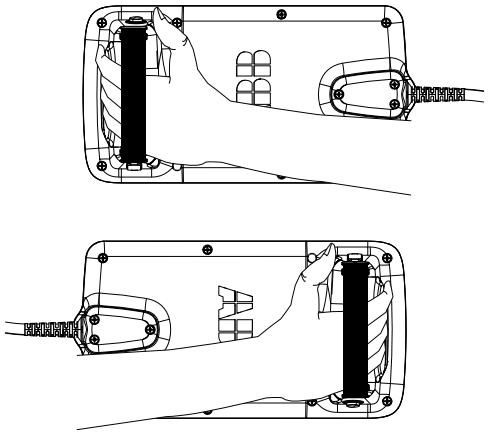
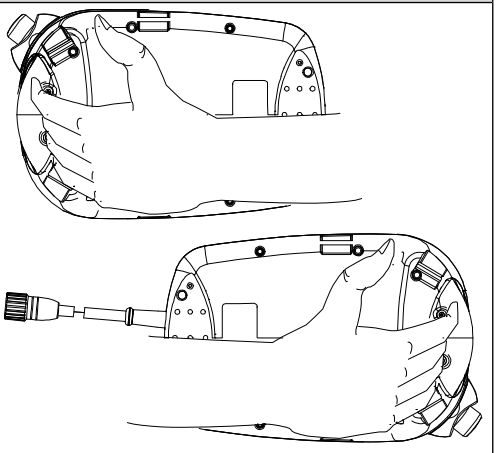
Hold-to-run is described in *Operating manual - IRC5 with FlexPendant*.

Continues on next page

How to hold the FlexPendant

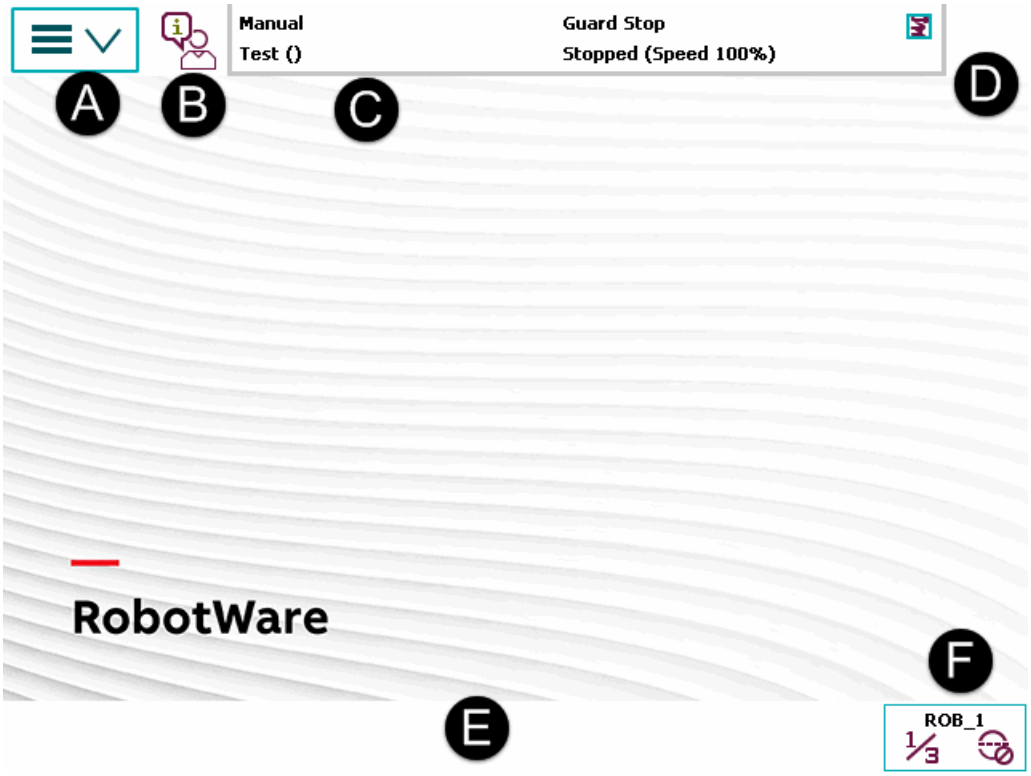
FlexPendant is typically operated while being held in the hand. The right-handed users use their left-hand to support the FlexPendant while their right-hand performs the operations on the touch screen. However, the left-handed users can easily adapt FlexPendant for their use.

For more details, see *Operating manual - IRC5 with FlexPendant*.

FlexPendant with emergency stop button at the connector	FlexPendant with emergency stop button at the outer edge
 xx1800000045	 en0400000913

Touchscreen elements

The illustration shows important elements of the FlexPendant touchscreen.



xx1400001446

Continues on next page

A	Main menu
B	Operator window
C	Status bar
D	Close button
E	Task bar
F	Quickset menu

Main menu

The following items can be selected from the Main menu:

- HotEdit
- Inputs and Outputs
- Jogging
- **Production Window**
- **Program Editor**
- **Program Data**
- **Backup and Restore**
- **Calibration**
- **Control Panel**
- **Event Log**
- FlexPendant **Explorer**
- **System Info**
- etc.

This is further described in section *The ABB Menu* in *Operating manual - IRC5 with FlexPendant*.

Operator window

The operator window displays messages from robot programs. This usually happens when the program needs some kind of operator response in order to continue. This is described in section *Operator window* in *Operating manual - IRC5 with FlexPendant*.

Status bar

The status bar displays important information about system status, such as operating mode, motors on/off, program state and so on. This is described in section *Status bar* in *Operating manual - IRC5 with FlexPendant*.

Close button

Tapping the close button closes the presently active view or application.

Task bar

You can open several views from the Main menu, but only work with one at a time. The task bar displays all open views and is used to switch between these.

Continues on next page

Quickset menu

The quickset menu provides settings for jogging and program execution. This is described in section T*he Quickset menu* in *Operating manual - IRC5 with FlexPendant*.

1.4 RobotStudio Online

Introduction to RobotStudio Online

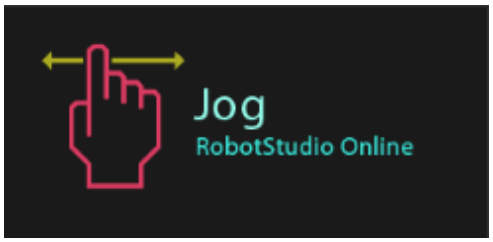
RobotStudio Online is a suite of **Windows Store** applications intended to run on **Windows 10** tablets. It provides functionality for the shop floor commissioning of robot systems.

You can run these apps on a tablet that communicates with the robot controller wirelessly. To enable certain functionality, such as entering manual mode and enabling power to the mechanical unit motors, you need a safety device that is connected to the robot using the same plug that alternatively is used to connect the FlexPendant.

The following RobotStudio Online apps are available in the Microsoft *Windows Store*:

Note

You must have **Windows 8.1** to run these Apps.

RobotStudio Online Apps	Description
 xx1400002047	**Manage** is a tool to manage IRC5 controllers on a network.
 xx1400002049	**Calibrate** is a tool for calibration and definition of frames with IRC5 controllers.
 xx1400002048	**Jog** is a tool for manual positioning (moving or jogging) with IRC5 controllers.

Continues on next page

RobotStudio Online Apps	Description
 xx1400002050	**Tune** is a tool for shop floor editing of RAPID programs with IRC5 controllers.
 xx1400002511	**Operate** is a tool used in production to view the program code.
 xx1500000832	**YuMi** is a tool for programming of the YuMi robot, IRB 14000 from ABB. It will help the users to get a fast introduction to robot programming using lead-thru and graphical programming.

1.5 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:

- Controller.Software.isRobotWare6Installing 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.

1.6 When to use different jogging devices

Overview

For operating and managing the robot, you can use any of the following:

- FlexPendant: Optimized for handling robot motions and ordinary operation
- RobotStudio: Optimized for configuration, programming and other tasks not related to the daily operation.
- RobotStudio Online Apps : Optimized for jogging, managing, working with the frames, calibration methods and RAPID programs available in the robot controller.

Start, restart and shut down the controller

To...	Use...
Start the controller	The power switch on the controller's front panel.
Restart the controller	The FlexPendant, RobotStudio, RobotStudio Online Apps or the power switch on the controller's front panel.
Shut down the controller	The power switch on the controller's front panel or the FlexPendant, tap **Restart**, then **Advanced**.
Shut down the main computer	The FlexPendant.

Run and control robot programs

To...	Use...
Jog a robot	The FlexPendant.
Start or stop a robot program	The FlexPendant, RobotStudio or RobotStudio Online Apps.
Start and stop background tasks	The FlexPendant, RobotStudio or RobotStudio Online Apps.

Communicate with the controller

To...	Use...
Acknowledge events	The FlexPendant or RobotStudio Online Apps.
View and save the controller's event logs	RobotStudio, FlexPendant or the RobotStudio Online Apps.
Back up the controller's software to files on the PC or a server	RobotStudio, FlexPendant or the RobotStudio Online Apps.
Back up the controller's software to files on the controller	The FlexPendant or RobotStudio Online Apps.
Transfer files between the controller and network drives	RobotStudio, FlexPendant or the RobotStudio Online Apps.

Continues on next page

Program a robot

To...	Use...
Create or edit robot programs in a flexible way. This is suitable for complex programs with a lot of logic, I/O signals or action instructions	RobotStudio to create the program's structure and most of the source code and the FlexPendant to store robot positions and make final adjustments to the program. When programming, RobotStudio provides the following advantages: • A text editor optimized for RAPID code, with auto-text and tool-tip information about instructions and parameters. • Program check with program error marking. • Close access to configuration and I/O editing.
Create or edit a robot program in a supportive way. This is suitable for programs that mostly consist of move instructions	The FlexPendant. When programming, the FlexPendant provides the following advantages: • Instruction pick lists • Program check and debug while writing • Possibility to create robot positions while programming
Add or edit robot positions	The FlexPendant with a combination of suitable RobotStudio Online Apps.
Modify robot positions	The FlexPendant with a combination of suitable RobotStudio Online Apps.

Configure the robot's system parameters

To...	Use...
Edit the system parameters of the running system	RobotStudio, FlexPendant or the RobotStudio Online Apps.
Save the robot's system parameters as configuration files	RobotStudio, FlexPendant or the RobotStudio Online Apps.
Load system parameters from configuration files to the running system	RobotStudio, FlexPendant or the RobotStudio Online Apps.
Load calibration data	RobotStudio, FlexPendant or the RobotStudio Online Apps.

Create, modify and install systems

To...	Use...
Create or modify a system	RobotStudio together with RobotWare and a valid RobotWare Key for systems based on RobotWare 5. RobotStudio together with RobotWare and license file for systems based on RobotWare 6.
Install a system on a controller	RobotStudio
Install a system on a controller from a USB memory	The FlexPendant.

Calibration

To...	Use...
Calibrate base frame etc.	The FlexPendant or the RobotStudio Online Apps.

Continues on next page

To...	Use...
Calibrate tools, work objects etc.	The FlexPendant or the RobotStudio Online Apps.

Related information

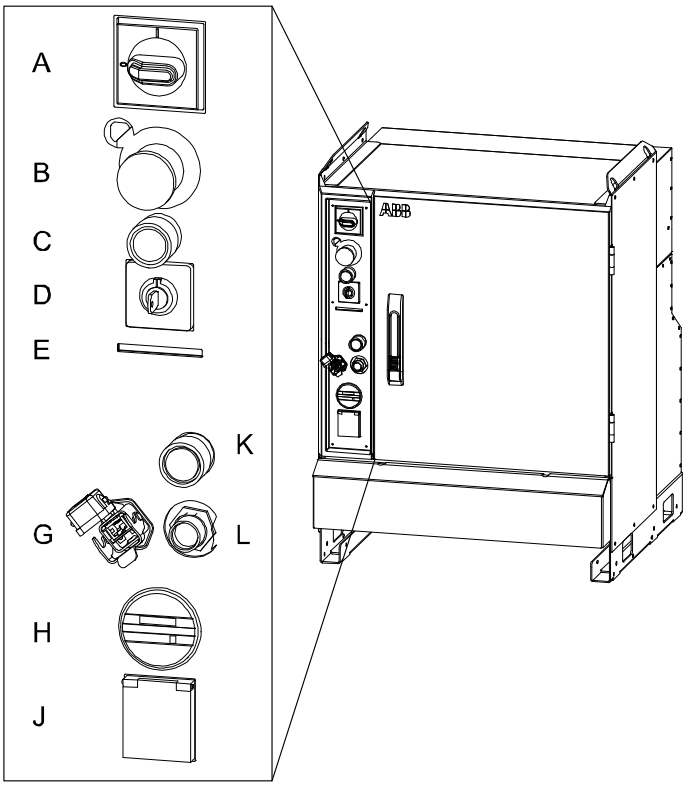
The table below specifies which manuals to read, when performing the various tasks referred to:

Recommended use...	for details, see manual...	Document number
FlexPendant	*Operating manual - IRC5 with FlexPendant*	*3HAC050941-001*
RobotStudio	*Operating manual - RobotStudio*	*3HAC032104-001*

1.7 Buttons and ports on the controller

Buttons and ports on the controller

These are the buttons and ports on an IRC5 controller. Some buttons and ports are options and might not be available on your controller. The buttons and ports look the same but the placing can differ depending on the controller model (IRC5 Standard, IRC5 Compact, or IRC5 Panel Mounted Controller) and if there is an external operator's panel.



xx0600002782

A	Main switch
B	Emergency stop
C	Motors on
D	Mode switch
E	Safety chain LEDs (option)
G	Service port for PC (option)
H	Duty time counter (option)
J	Service outlet 115/230 V, 200 W (option)
K	Hot plug button (option)
L	Connector for FlexPendant

Related information

Product manual - IRC5, IRC5 of design 14.

Continues on next page

Product manual - IRC5 Panel Mounted Controller, IRC5 of design 14.

Product manual - IRC5 Compact, IRC5 of design 14.

Operating manual - Troubleshooting IRC5.

This page is intentionally left blank

2 Get started

2.1 About this chapter

Overview

This chapter describes how to connect the FlexPendant to the controller and how to set up network connections. It also presents a number of often performed work tasks with the FlexPendant, described as action scenarios.

2.2 Connections

2.2.1 Connecting a FlexPendant

Location of FlexPendant connector

The FlexPendant connector is located on the operator's panel on the controller, or on an external operator's panel. The Panel Mounted Controller has a connector on the front.

Connecting a FlexPendant

CAUTION

Always inspect the connector for dirt or damage before connecting it to the controller. Clean or replace any damaged parts.

	Action	Information
1	Locate the FlexPendant socket connector on the controller or operator's panel.	The controller must be in manual mode. If your system has the option Hot plug, then you can also disconnect in auto mode. See section *Using the hot plug option on page 38*. The controller must be in manual mode.
2	Plug in the FlexPendant cable connector.	
3	Screw the connector lock ring firmly by turning it clockwise.	
4	The FlexPendant starts automatically when connected and verifies that it has the correct software installed. If an update is needed, this is shown.	*Updating the add-in FlexPendant SxTPU4 Software on page 36*

Updating the add-in FlexPendant SxTPU4 Software

Note

The add-in is only available for the FlexPendant with the emergency stop located at the connector. All other FlexPendant versions will automatically update their software via the controller (if needed).

The FlexPendant with the emergency stop located at the connector has an add-in that enables support for different RobotWare versions. This is the **FlexPendant SxTPU4 Software** add-in. The version of the add-in is shown during start-up.

When connecting the FlexPendant, the add-in verifies that it has support for the RobotWare version on the controller. If the RobotWare version is not supported by default, then the add-in requires an update. There are two method to update the FlexPendant add-in. The update is distributed as a software package.

- The update can be installed using a USB drive.
- If the update is available on the controller, then the FlexPendant will update itself when connecting it to the controller.

Continues on next page

Once the add-in is updated, the FlexPendant can be connected to other IRC5 controllers with the same RobotWare version without requiring additional updates.

Update using a USB drive

Use the following procedure to update the add-in using a USB drive.

1. Download the update from RobotStudio, in the tab **Add-Ins**.
2. Save the software package (.rspak) on a USB drive in the folder `SxTPU4`, located in the root folder.
3. With the FlexPendant connected to the controller, reset the FlexPendant with the USB drive connected.
4. The update starts automatically and takes approximately 3-4 minutes.

Update from the controller

Use the following procedure to update the add-in from the controller.

1. In RobotStudio, use **Installation Manager 6** to create or update a system on your controller. Add the product *FlexPendantSxTPU4Software*.
2. Connect the FlexPendant to the controller.
3. The update starts automatically and takes approximately 3-4 minutes.

Handling the FlexPendant cables

FlexPendant cables are allowed to be rolled up by hand with a minimum bending radius of 10 times the cable diameter. This also applies to the extension cable. For example, if the cable is 9.5 mm then it is allowed to roll it with a radius of 95 mm.

Extension cables are not allowed to be used in chains.

2.2.2 Using the hot plug option

Hot plug option

The hot plug option makes it possible to:

- Disconnect the FlexPendant from a system in automatic mode and thereby run the system without a FlexPendant connected.
- Temporarily connect and operate a FlexPendant without interrupting the application running on the system.

WARNING

Pressing the hot plug button disables the emergency stop button on the FlexPendant. Only press the hot plug button while connecting or disconnecting the FlexPendant.

WARNING

A disconnected FlexPendant cannot initiate a protective or emergency stop. It must be stored out of sight so that it cannot be mistaken for being in use.

Connect and disconnect the FlexPendant using the hot plug button

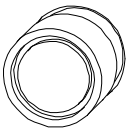
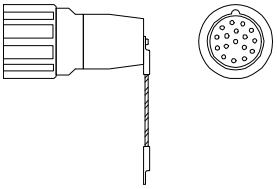
The following procedure describes how to connect or disconnect the FlexPendant on a system in automatic mode using the hot plug button option.

Note

Do not switch to manual mode (or manual full speed mode) while the system is running without the FlexPendant. The FlexPendant must be connected when you switch to automatic mode otherwise you cannot confirm the mode change.

	Action	Information
1	Make sure that the system is in automatic mode.	
2	Press and hold the hot plug button.	A red lamp inside the button indicates when pressed.

Continues on next page

	Action	Information
3	Keep pressing the hot plug button and at the same time, switch the jumper plug with the FlexPendant plug.	A  B  xx0600002784 A: Hot plug button B: FlexPendant connector  xx0600002796 Jumper plug
4	Release the hot plug button.	Make sure that the button is not stuck in the actuated position since this disables the FlexPendant emergency stop button.
5	If the connected FlexPendant does not have support for the RobotWare version running on the controller, then a dialog is shown that the add-in must be updated. See *Updating the add-in FlexPendant Sx-TPU4 Software on page 36*.	The three-position enabling device and emergency stop button are active even if the add-in dialog is shown.

Note

When the FlexPendant is disconnected, the jumper plug must be connected in its place.

Note

If the hot plug button is released while neither the jumper plug, nor the FlexPendant is connected, the robot movements will be stopped since the emergency stop chains are opened.

Continues on next page

2.2.2 Using the hot plug option
Continued

Limitations for messages on the FlexPendant

When using the hot plug option, the following limitations apply to messages on the FlexPendant:

Operator messages

Some applications may require input from the operator by using the FlexPendant (e.g. applications using RAPID instructions `TPReadNum`, `UIMsgBox`, etc.). If the application encounters such an operator message, program execution will wait. After connecting the FlexPendant you must then stop and start the program execution to be able to see and respond to these messages. They are not displayed automatically by just connecting the FlexPendant.

If possible, avoid using these types of instructions when programming systems that are using the hot plug button option.

Event log messages

When connecting the FlexPendant, event log messages can be viewed also for the period when the FlexPendant was disconnected, since these are stored on the controller.

2.2.3 Set up the network connection

When do I need to setup the network connection?

You need to setup the controller's network connection when the controller is connected to a network for the first time or when the network addressing scheme changes.

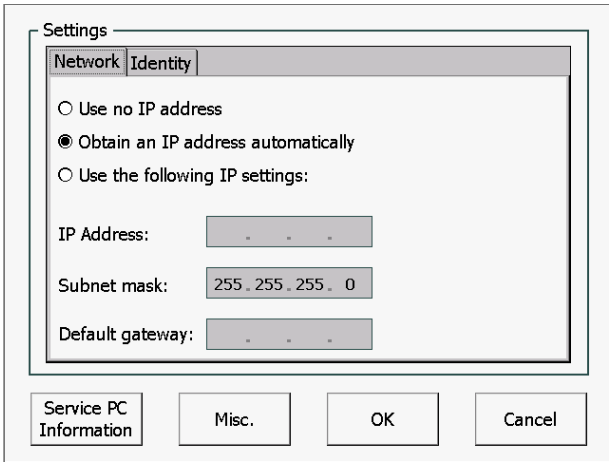
Preparations

If an IP address is to be obtained automatically, make sure there is a server running that supplies the network with IP addresses (a DHCP server). Otherwise you will not be able to access the controller via the controller network.

It is still possible to access the controller via the service PC connection.

Network connection dialog box

The illustration shows the network connection dialog box.



en0400000902

Set up the network connection

Regardless of how you choose to set up the network connections, the first steps are common:

	Action	Information
1	Perform a restart and select the option **Start Boot Application**.	**Start Boot Application** is described in section *Restart and start boot application on page 122*.
2	In the Boot Application, tap **Settings**. The network connection dialog is displayed.	
3	If you choose to use no IP address, then tap **Use no IP address**. Otherwise, proceed below!	In some cases it can be useful to disconnect the controller from the network, without disconnecting the network cable. Without IP address the controller cannot be accessed from other equipment on the same network.

Continues on next page

2.2.3 Set up the network connection

Continued

	Action	Information
4	If you choose to obtain an IP address automatically, then tap **Obtain an IP address automatically**. Otherwise, proceed below!	
5	If you choose to use a fixed IP address, tap **Use the following IP address**. Enter the IP address, subnet mask, and default gateway.	**Note** Make sure a valid address is used so there are no conflicts in the network. A conflict may cause other controllers to malfunction **Note** It is not recommended using leading zeros in dot-decimal notation of IP addresses. The numbers may wrongly be interpreted as octal numbers. Different behaviors on virtual and real controllers may be experienced.
6	Tap **OK** to save the new setting.	
7	In the Boot Application, tap **Select System** and select the system to restart.	How to select a system is detailed in section *Selecting a system on page 119*.
8	In the Boot Application, tap **Restart Controller** and tap **OK** to restart the controller with new settings.	**Note** You can verify the new settings by tapping **ABB -> System Info -> Controller properties -> Network connections -> LAN**

2.3 Action scenarios

2.3.1 About action scenarios

Overview

This chapter presents brief procedures, detailing a number of typical actions a typical user may perform. It also includes references to detailed information about the same topics.

The brief information given, is intended to be used directly by experienced users, while the references may be more adequate for novices and for training purposes.

Related information

Note that there may be more information available than the one referred to in the procedures.

Information about:

- a specific menu is described in *Operating manual - IRC5 with FlexPendant*.
- a specific button on the FlexPendant is described in *The FlexPendant on page 19*.
- a specific button is described in the product manual for the controller, for tasks performed using the controls on the controller cabinet.
- how to perform a specific task is described in *Operating manual - IRC5 with FlexPendant*, for example programming or running in production.

Related information can also be found in other manuals:

- *Operating manual - RobotStudio*
- Product manual for the controller

2.3.2 System start

Prerequisites before start

This procedure details the main steps required to start the system when the power has been switched off.

All information is based on the assumption that *working system software has already been installed* on the robot controller, as the case would be at first start directly after delivery.

Note that there may be more information available than the one referred to in the procedure.

System start

This procedure details all required steps to start the system for the first time. For everyday start, step 4 is normally the only required step.

	Action	Information
1	Install the robot equipment.	Mechanical installation and electrical connections between manipulator and controller is described in the *Product manual* of the robot and controller respectively.
2	Make sure the safety circuits of the system are properly connected to the robot cell or have jumper connections installed (if required).	How to connect the safety circuits is detailed in the robot's *Product manual*.
3	Connect the FlexPendant to the controller.	The FlexPendant and its major parts and functions are detailed in section *The FlexPendant on page 19* How to connect the FlexPendant to the controller is detailed in section *Connecting a FlexPendant on page 36*
4	Switch the power on.	Use the main switch on the controller.
5	If the controller or manipulator have been replaced with spare parts, make sure the calibration values, revolution counters and serial numbers are updated correctly.	Normally, only the revolution counters require updating, which is to be performed as detailed in the product manual for the robot. If required, transfer the calibration data from the serial measurement board as detailed in *Serial measurement board memory on page 191* for systems *without* the Absolute Accuracy option. If required, enter the calibration data as detailed in *Loading calibration data using the FlexPendant on page 189* for systems *with* the Absolute Accuracy option.
6	This step is only required if the robot system will be connected to a network. Perform a restart and start the boot application. The Boot Application is started.	Described in section *Restart and start boot application on page 122*.

Continues on next page

	Action	Information
7	This step is only required if the robot system will be connected to a network. Use the Boot Application to: • set the IP address of the controller cabinet • set the network connections • select the system • restart the system The system is restarted.	How to use the Boot Application is detailed in section *Using the Boot Application on page 117*. At this point, a single system is available.
8	Install RobotStudio on a PC.	Proceed as detailed in *Operating manual - RobotStudio*. RobotStudio is used to create a system to run on the controller, but at this point (prior to the first start) a system is already installed by the manufacturer.
9	Connect the controller to a PC (through the service port) or to the network (if used).	Proceed as detailed in section *Connecting a PC to the controller* in *Operating manual - RobotStudio*. Also see section *Set up the network connection on page 41*.
10	Start RobotStudio on the PC.	Proceed as detailed in *Operating manual - RobotStudio*.
11	Restart the controller.	
12	The robot system is now ready for operation.	

2.3.3 Working with inputs and outputs

Working with inputs and outputs

This procedure details the main steps required to set outputs, read inputs and configure I/O devices.

For more information about I/O, see section *Handling inputs and outputs, I/O on page 103*.

	Action	Information
1	Create a new I/O.	I/O signals are created using system parameters.
2	Before using any input or output, the system must be configured to enable the I/O functions.	Configuring the system is done when creating the system. How to do this is described in *Operating manual - RobotStudio*.
3	Set a value to a specific *digital output*, *analog output*, *digital input* or *analog input*.	Described in in section *Simulating and changing signal values* in *Operating manual - IRC5 with FlexPendant*.
4	Set safety signals.	Signal explanation is detailed in *Safety I/O signals on page 107*
5	Edit an I/O.	Described in in section *Simulating and changing signal values* in *Operating manual - IRC5 with FlexPendant*.

2.3.4 Backup and restore

Backup and restore

How to perform the backup is described in section *Backup the system on page 133*.

Re-introducing the previously saved memory contents from the backup into the robot controller is called *performing a restore*. How to perform the restore is described in section *Restore the system on page 137*.

Note that there may be more information available than the one referred to above.

2.3.5 Upgrading

Upgrading

This procedure details the main steps required to correctly upgrade the system. By upgrading we mean changing hardware, such as replacing circuit board with newer versions, as well as loading software with later releases.

Note that there may be more information available than the one referred to in the procedure.

Type of upgrade	Information
When replacing circuit boards such as buses, I/O boards, etc., with newer versions, the system will automatically reflash the unit.  xx0100000003 During reflashing, the system may restart several times, and it is vital not to shut down the system, or in any other way interrupt the automatic process.	
When upgrading the robot or controller mechanically, fitting instructions are normally delivered with the kit. If no such instruction are provided, useful information may be found in the Repair section of the *Product Manual* of the equipment in question.	
When upgrading the system software, the system must be changed in order to reflect the additions. A new license key may be required.	How to modify an existing system is detailed in section *How to Modify a System* in the *Operating manual - RobotStudio*. How to create a new system is detailed in section *Creating a new system* in the *Operating manual - RobotStudio*.

2.3.6 Installing RobotWare add-ins

Installing RobotWare add-ins

The main steps required to correctly install a RobotWare add-in is described in *Operating manual - RobotStudio*.

For instruction on building RobotWare add-ins, visit the ABB Robotics Developer Center web site at *http://developercenter.robotstudio.com* or see *Application manual - RobotWare Add-Ins*.

This page is intentionally left blank

3 Programming

3.1 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.

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.

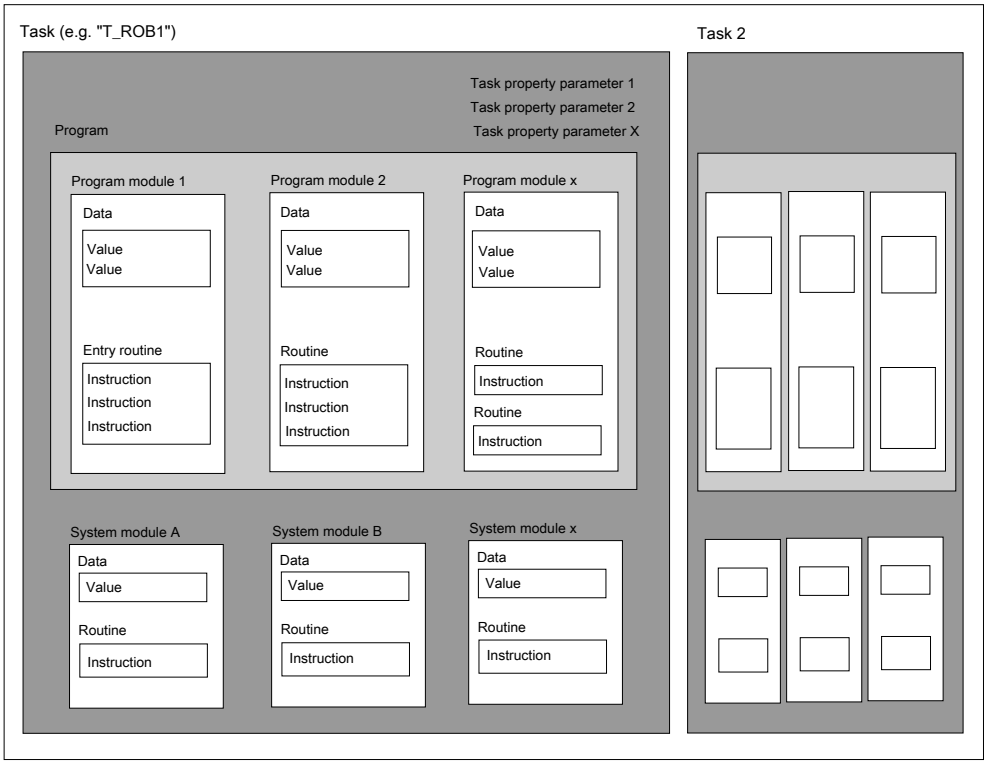
Define tool and work object coordinate systems before you start programming. As you add more objects later you also need to define the corresponding coordinate systems.

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*.

3.2 The structure of a RAPID application

Illustration of a RAPID application



en0300000576

Parts

Part	Function
Task	Each task usually contains a RAPID program and system modules aimed at performing a certain function, e.g. spot welding or manipulator movements. A RAPID application may contain one task. If you have the *Multitasking* option installed, then there can be more than one task. Read more about *Multitasking* in *Application manual - Controller software IRC5*.
Task property parameter	The task property parameters set certain properties for all task contents. Any program stored in a certain task, assumes the properties set for that task. The task property parameters are specified in *Technical reference manual - RAPID Overview*.
Program	Each program usually contains program modules with RAPID code for different purposes. Any program must have an entry routine defined to be executable.

Continues on next page

Part	Function
Program module	Each program module contains data and routines for a certain purpose. The program is divided into modules mainly to enhance overview and facilitate handling the program. Each module typically represents one particular robot action or similar. All program modules will be removed when deleting a program from the controller program memory. Program modules are usually written by the user.
Data	Data are values and definitions set in program or system modules. The data are referenced by the instructions in the same module or in a number of modules (availability depending on data type). Data type definitions are specified in the *Technical reference manual - RAPID Instructions, Functions and Data types*.
Routine	A routine contains sets of instructions, i.e. defines what the robot system actually does. A routine may also contain data required for the instructions.
Entry routine	A special type of routine, in English sometimes referred to as "main", defined as the program execution starting point. **Note** Each program **must** have an entry routine called "main", or it will not be executable. How to appoint a routine as entry routine is specified in *Technical reference manual - RAPID Overview*. The default name for main can be changed by the system parameter configurations, type *Task*. See *Technical reference manual - System parameters*.
Instruction	Each instruction is a request for a certain event to take place, e.g. "Run the manipulator TCP to a certain position" or "Set a specific digital output". The instructions, their syntax and function is thoroughly described in the *Technical reference manual - RAPID Instructions, Functions and Data types*.
System module	Each system module contains data and routines to perform a certain function. The program is divided into modules mainly to enhance overview and facilitate handling the program. Each module typically represents one particular robot action or similar. All system modules will be retained when "Delete program" is ordered. System modules are usually written by the robot manufacturer or line builder.

3.3 Programming

3.3.1 Handling of programs

Overview

This section details how to perform normal 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

Each task must contain *one* program, no more, no less. Note that the following procedures describe a single task system, i.e. only one task is available.

How to create a new program *when no program is available* is detailed in section *Creating a new program on page 54*.

About program files

When saving a program to the controller hard disk, it is by default saved to the directory HOME in the system's folder unless otherwise stated. How to set another default path is detailed in section *How to set default paths* in *Operating manual - IRC5 with FlexPendant*.

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.

Creating a new program

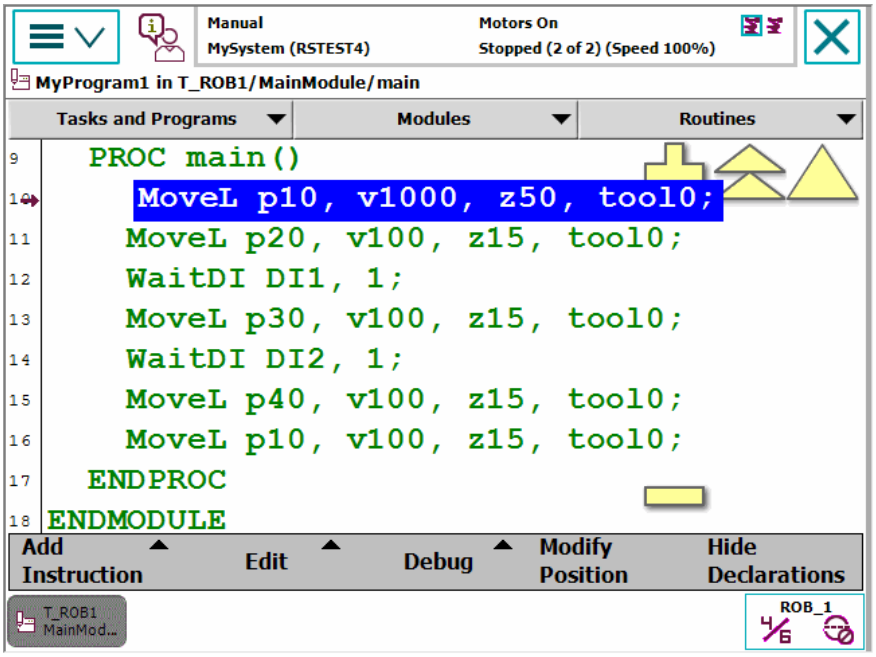
This section describes how to create a new program.

	Action
1	On the **ABB** menu, tap **Program Editor**.
2	Tap **Tasks and Programs**.
3	Tap **File**, then **New Program**. 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.
4	Continue by adding instructions, routines, or modules. A new program is created.

Continues on next page

Loading an existing program

This section describes how to load an existing program.

	Action
1	On the **ABB** menu, tap **Program Editor**.
2	Tap **Tasks and Programs**.
3	Tap **File**, then **Load Program**. 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 loaded program.
4	Use the file searching tool to locate the program file to be loaded (file type pgf). Then tap **OK**. The program is loaded and the program code is displayed.  en0400000699

Saving a program

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

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

	Action
1	On the **ABB** menu, tap **Program Editor**.
2	Tap **Tasks and Programs**.
3	Tap **File** and select **Save Program As...**.
4	Use the suggested program name or tap **...** to open the soft keyboard and enter a new name. Then tap **OK**.

Continues on next page

3.3.1 Handling of programs
Continued

Renaming a loaded program

This section describes how to rename a loaded program.

	Action
1	On the **ABB** menu, tap **Program Editor**.
2	Tap **Tasks and Programs**.
3	Tap **File** and select **Rename Program**. A soft keyboard is displayed.
4	Use the soft keyboard to enter the new name of the program. Then tap **OK**.

Deleting a program

This section describes how to delete a program.

	Action
1	On the **ABB** menu, tap **Program Editor**.
2	Tap **Tasks and Programs**.
3	Tap **File** and select **Delete Program**. A confirmation dialog is displayed.
4	Tap **OK** to delete, or **Cancel** to keep the program intact.

3.3.2 Handling of modules

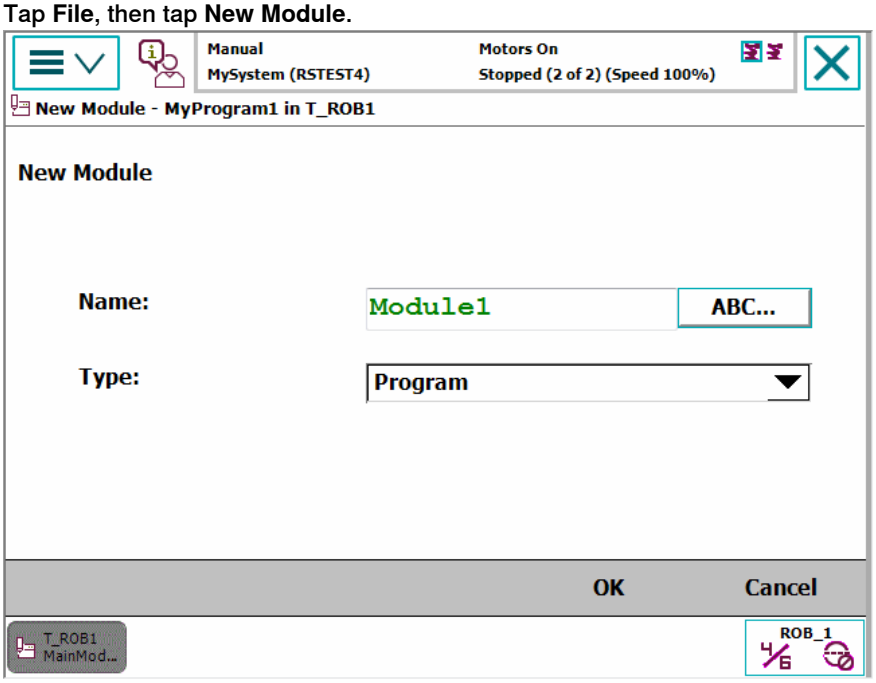
Overview

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

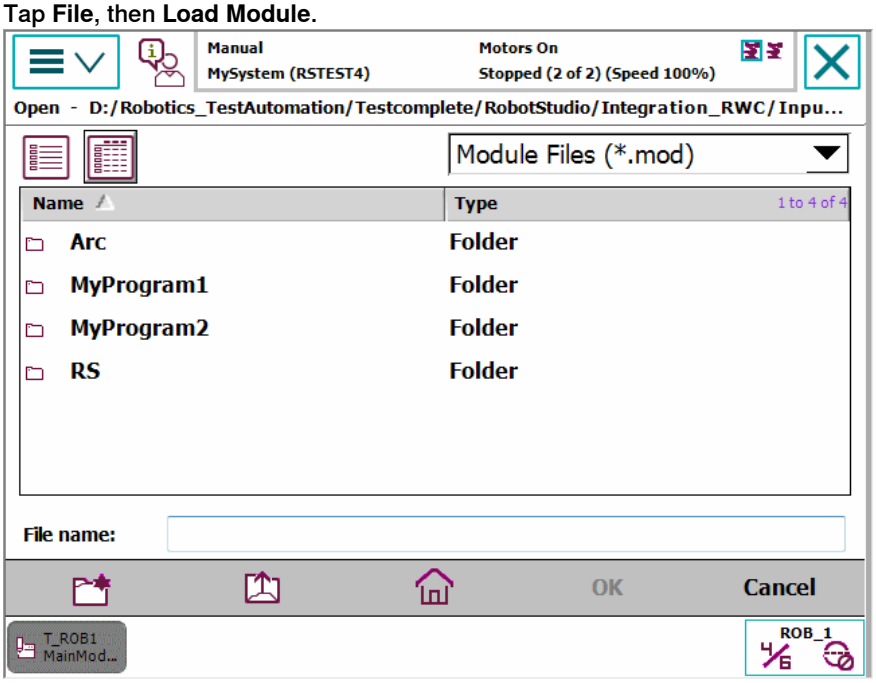
This section describes how to create a new module.

	Action
1	On the **ABB** menu, tap **Program Editor**.
2	Tap **Modules**.
3	Tap **File**, then tap **New Module**.  en0400000688
4	Tap **ABC...** and use the soft keyboard to enter the new module's name. Then tap **OK** to close the soft keyboard.
5	Select which type of module to be created: • Program • System Then tap **OK**. How to later switch between these types is detailed in section *Changing type of module on page 59*.

Continues on next page

Loading an existing module

This section describes how to load an existing module.

	Action
1	On the **ABB** menu, tap **Program Editor**.
2	Tap **Modules**.
3	Tap **File**, then **Load Module**.  en0400000689 Locate the module to be loaded.
4	Tap **OK** to load the selected module. The module is loaded.

Saving a module

This section describes how to save a module.

	Action
1	On the **ABB** menu, tap **Program Editor**.
2	Tap **Modules** and tap to select the module you want to load.
3	Tap **File**, then **Save Module As...**
4	Tap on the suggested file name and use the soft keyboard to enter the module's name. Then tap **OK**.
5	Use the file searching tool to locate where you want to save the module. Then tap **OK**. The module is saved.

Renaming a module

This section describes how to rename a module.

	Action
1	On the **ABB** menu, tap **Program Editor**.

Continues on next page

	Action
2	Tap **Modules**.
3	Tap **File**, then **Rename Module...** The soft keyboard is displayed.
4	Use the soft keyboard to enter the module's name. Then tap **OK**.

Changing type of module

This section describes how to change the type of module.

	Action
1	On the **ABB** menu, tap **Program Editor**.
2	Tap **Modules** and select the module to be changed.
3	Tap **File**, then **Change declaration...**
4	Tap **Type** and select module type.
5	Tap **OK**.

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.

	Action
1	On the **ABB** menu, tap **Program Editor**.
2	Tap **Modules** and tap to select the module you want to delete.
3	Tap **File**, then **Delete Module...** A dialog box is displayed.
4	Tap **OK** to delete the module without saving it. If you want to save the module first, tap **Cancel** and save the module first. How to save the module is detailed in section *Saving a module on page 58*.

3.3.3 Handling of routines

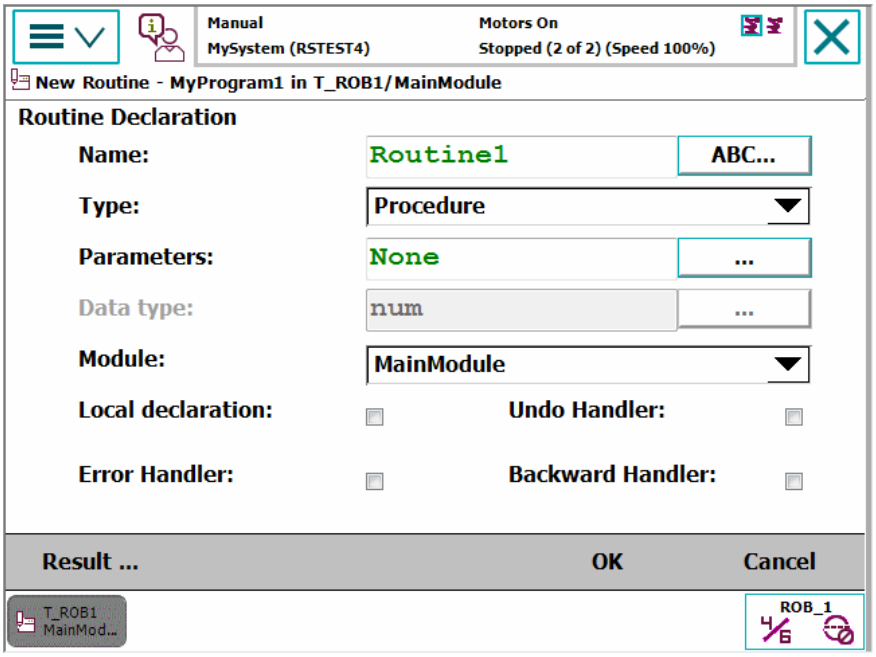
Overview

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.

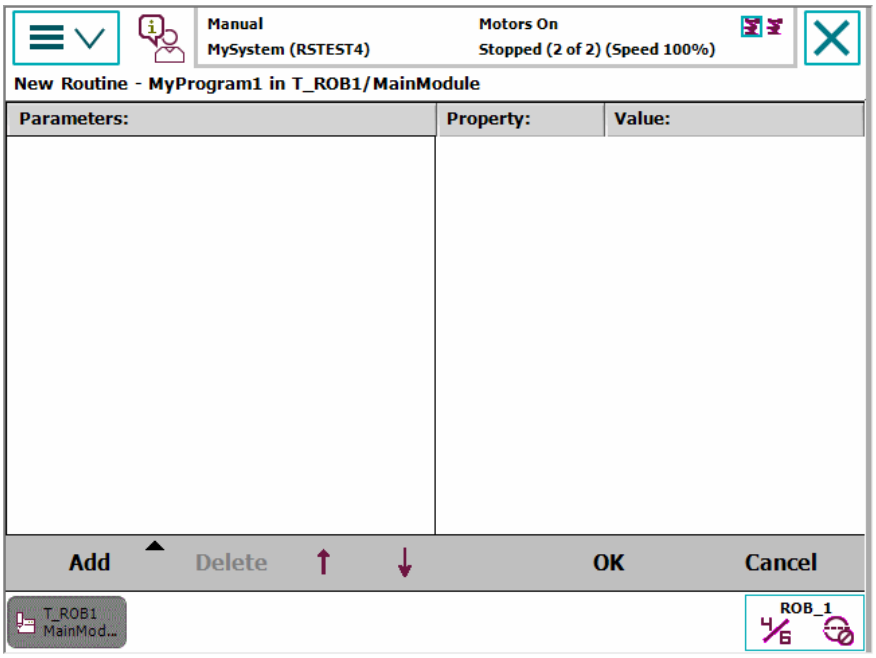
	Action
1	On the **ABB** menu, tap **Program Editor**.
2	Tap **Routines**.
3	Tap **File**, then **New Routine**. A new routine is created and displayed with default declaration values.  en0400000692
4	Tap **ABC...** and use the soft keyboard to enter the new routines' name. Then tap **OK**.
5	Select the type of routine: • Procedure: used for a normal routine without return value • Function: used for a normal routine with return value • Trap: used for an interrupt routine
6	Do you need to use any parameters? If YES; tap **...** and proceed as detailed in section *Defining parameters in routine on page 61*. If NO; proceed to the next step.
7	Select module to add the routine to.

Continues on next page

	Action
8	Tap the checkbox to select **Local declaration** if the routine should be local. A local routine can only be used in the selected module.
9	Tap **OK**.

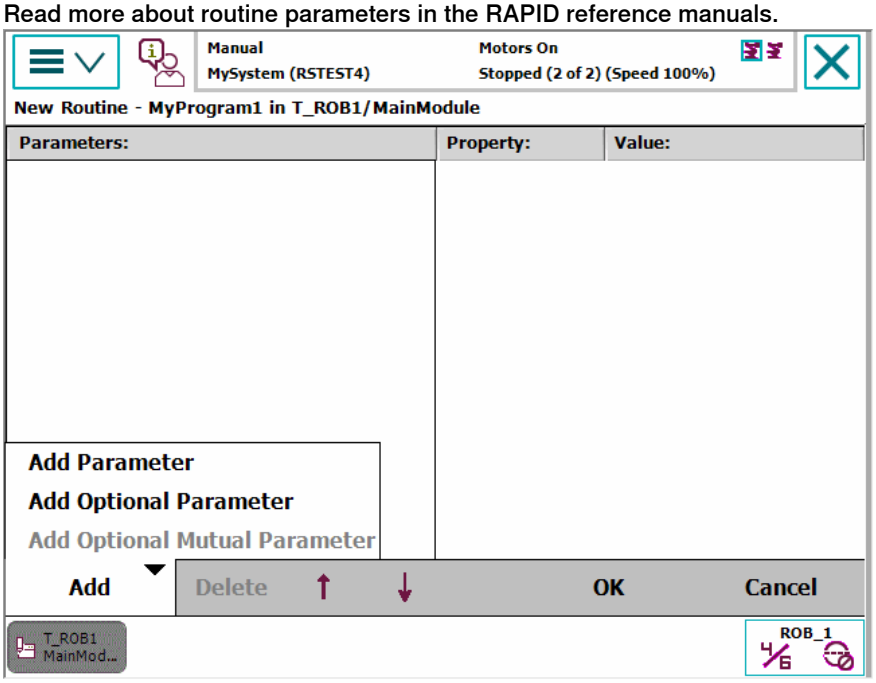
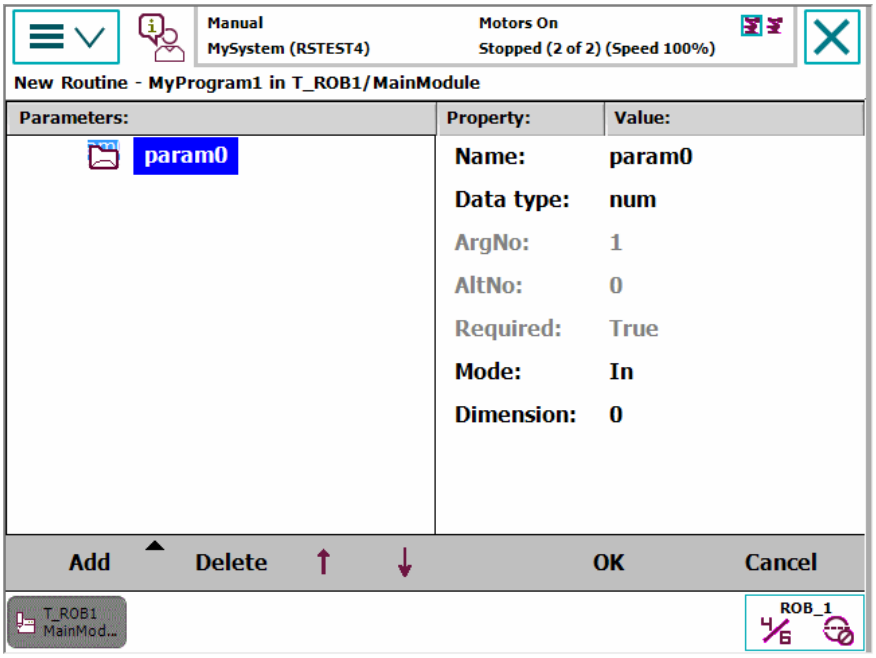
Defining parameters in routine

This section describes how to define parameters in a routine.

	Action
1	In the routine declaration, tap **...** to define parameters. A list of defined parameters is displayed.  en0400000693

Continues on next page

3.3.3 Handling of routines
Continued

	Action
2	If no parameters are shown, tap **Add** to add a new parameter. • **Add optional parameter** adds a parameter that is optional • **Add optional mutual parameter** adds a parameter that is mutually optional with another parameter Read more about routine parameters in the RAPID reference manuals.  en0400000695
3	Use the soft keyboard to enter the name of the new parameter and then tap **OK**. The new parameter is displayed in the list.  en0400000696
4	Tap to select a parameter. To edit values, tap the value.

Continues on next page

	Action
5	Tap **OK** to return to the routine declaration.

Creating a copy of a routine

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

	Action
1	On the **ABB** menu, tap **Program Editor**.
2	Tap **Routines**.
3	Highlight the routine by tapping it.
4	Tap **File**, then **Copy Routine**. The new routine is displayed. The name of the new routine is set to the same as the original with the suffix *Copy*.
5	Make any changes in the declarations for the new routine copy. Then tap **OK**. How to make all declarations is detailed in section *Creating a new routine on page 60*.

Changing the declaration of a routine

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

	Action
1	On the **ABB** menu, tap **Program Editor**.
2	Tap **Routines**.
3	Highlight the routine by tapping it.
4	Tap **File**, then **Change Declaration**
5	Change any declaration values for the routine. Then tap **OK**. Declaration settings are described in section *Creating a new routine on page 60*.

Moving a routine

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

	Action
1	On the **ABB** menu, tap **Program Editor**.
2	Tap **Routines**.
3	Highlight the routine by tapping it.
4	Tap **File**, then **Move Routine...**
5	Select task and module. Then tap **OK**.

Deleting a routine

This section describes how to delete a routine from memory.

	Action
1	On the **ABB** menu, tap **Program Editor**.
2	Tap **Routines**.
3	Highlight the routine by tapping it.
4	Tap **File**, then **Delete Routine...** A dialog box is displayed.

Continues on next page

3.3.3 Handling of routines
Continued

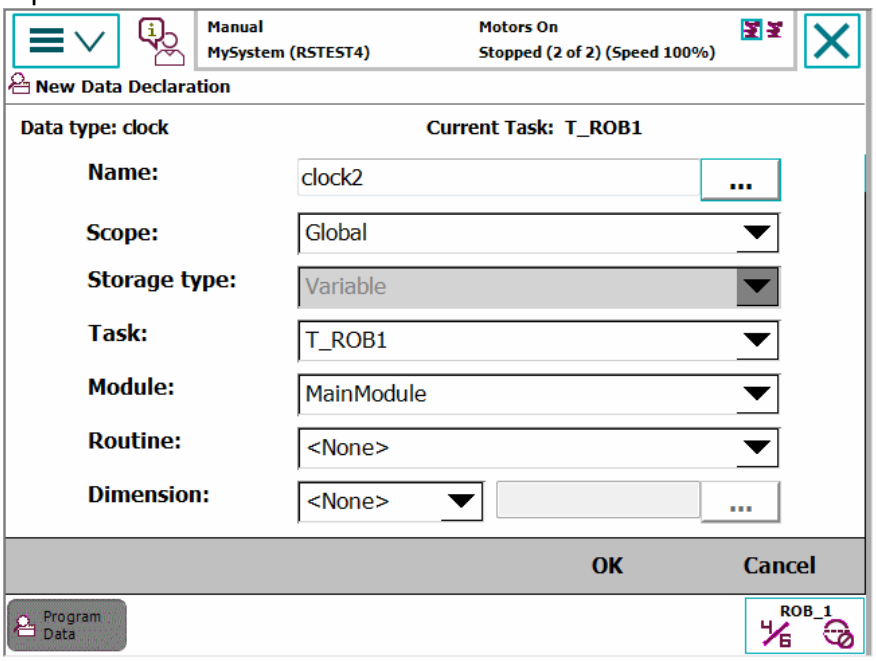
	Action
5	Tap: • **OK** to delete the routine without saving any changes made to it. • **Cancel** to revert without deleting the routine.

3.4 Data types

3.4.1 Creating new data instance

Creating new data instance

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

	Action
1	On the **ABB** menu, tap **Program Data**. A list of all available data types is displayed.
2	Tap the data instance type to be created, i.e. **bool** and then tap **Show data**. A list of all instances of the data type is displayed.
3	Tap **New**.  en0400000663
4	Tap **...** the right of **Name** to define the data instance's name.**Name**
5	Tap the **Scope** menu to set accessibility for the data instance. Select: • **Global** - reachable by all tasks • **Local** - reachable within the module • **Task** - reachable within the task
6	Tap the **Storage type** menu to select type of memory used for the data instance. Select: • **Persistent** if the data instance is persistent • **Variable** if the data instance is variable • **Constant** if the data instance is constant
7	Tap the **Module** menu to select module.
8	Tap the **Routine** menu to select routine.

Continues on next page

3.4.1 Creating new data instance

Continued

	Action
9	If you want to create an array of data instances, then tap the **Dimensions** menu and select the number of dimensions in the array, 1-3. • 1 • 2 • 3 • None Then tap **...** to set the **Size** of the array's axes.
10	Tap **OK**.

3.5 Tools

3.5.1 Creating a tool

What happens when you create a tool?

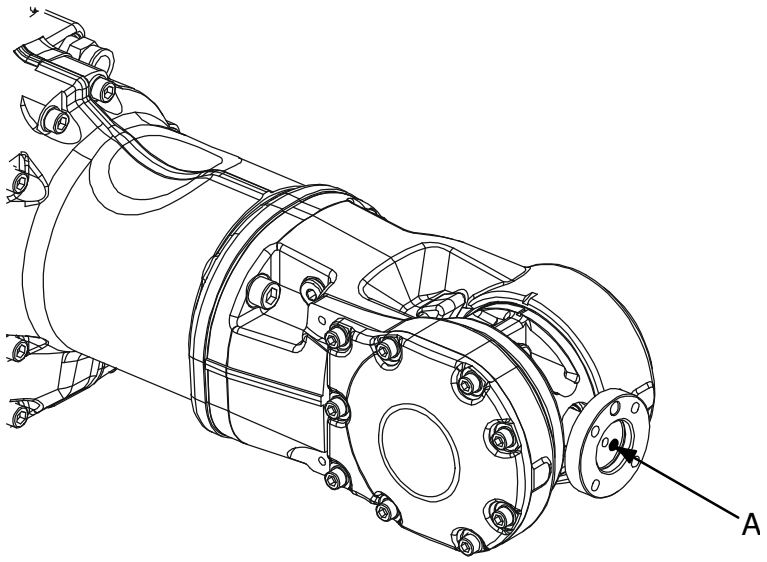
When you create a new tool a variable of the data type `tooldata` is created. The variable name will be the name of the tool. For more information on data types, see *Technical reference manual - RAPID Instructions, Functions and Data types*.

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 197*.



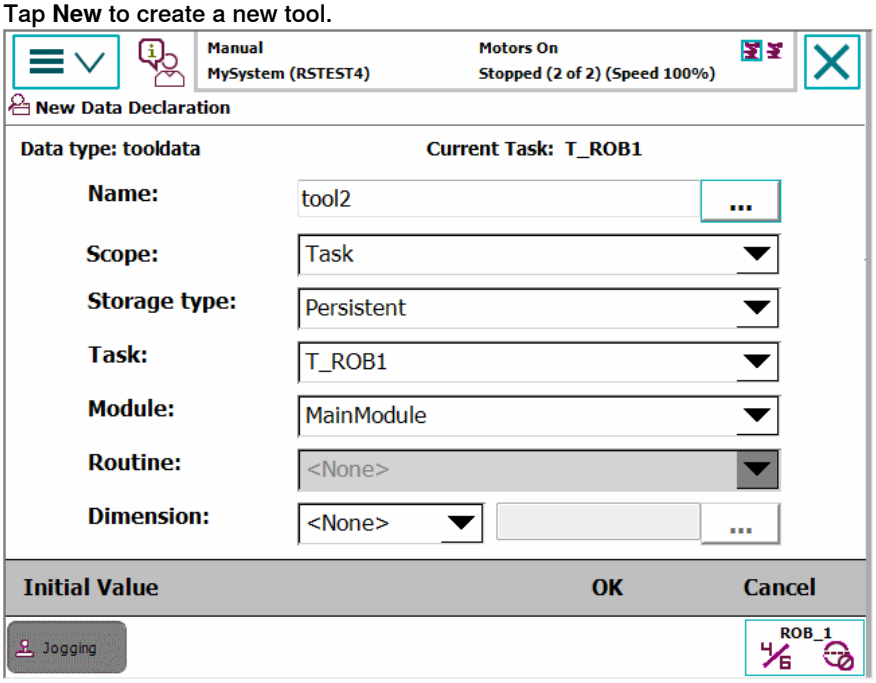
en0400000779

A	Tool center point, TCP, for tool0

	Action
1	On the **ABB** menu, tap **Jogging**.
2	Tap **Tool** to display the list of available tools.

Continues on next page

3.5.1 Creating a tool
Continued

	Action
3	Tap **New** to create a new tool.  en0300000544 Enter values for each field, see table below.
4	Tap **OK**.

Tool declaration settings

If you want to change...	then...	Recommendation
the name of the tool	tap ... button next to **Name**	Tools are automatically named `tool` followed by a running number, for example `tool10` or `tool21`. 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 menu	Tools should always be global, as to be available to all modules in the program.
the storage type	-	Tool variables must always be persistent.
the module	select the module in which this tool should be declared from the menu	

Continues on next page

If you want to change...	then...	Recommendation
the size of the data array's axes	tap ... button next to **Dimension**	

Note

The created tool is not useful until you have defined the tool data (TCP coordinates, orientation, weight etc.). See *Editing the tool data on page 74* and section about LoadIdentify in *Operating manual - IRC5 with FlexPendant*.

3.5.2 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.

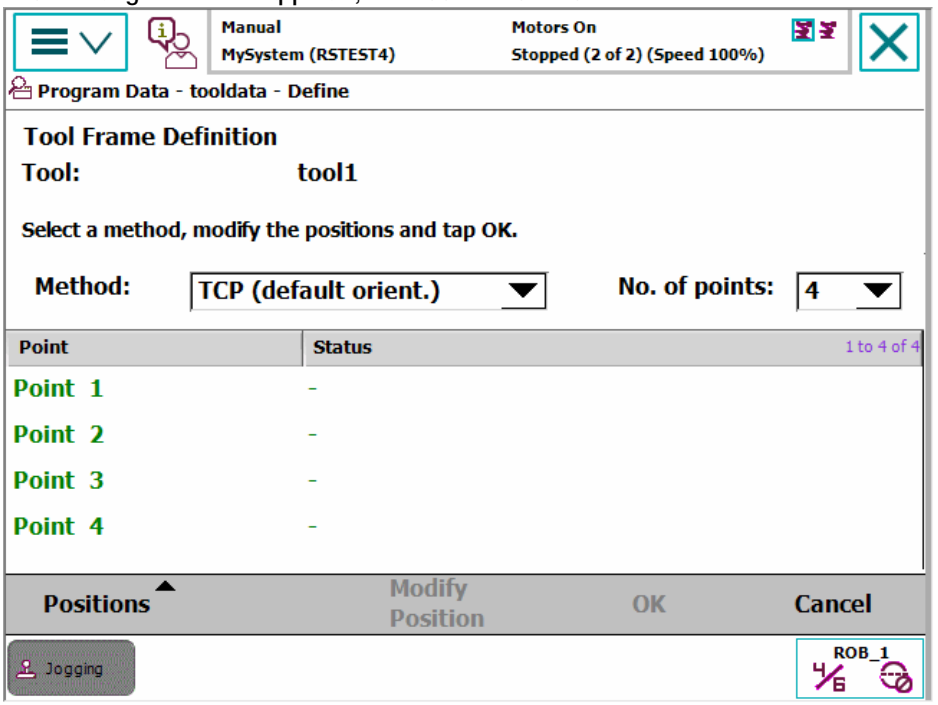
If you want to...	...then select
set the orientation the same as the orientation of the robot's mounting plate	**TCP (default orient.)**
set the orientation in Z axis	**TCP&Z**
set the orientation in X and Z axes	**TCP&Z,X**

How to select a method

This procedure describes how to select the method to be used when defining the tool frame.

	Action
1	On the **ABB** menu, tap **Jogging**.
2	Tap **Tool** to display a list of available tools.
3	Select the tool you want to define.
4	In the Edit menu, tap **Define**

Continues on next page

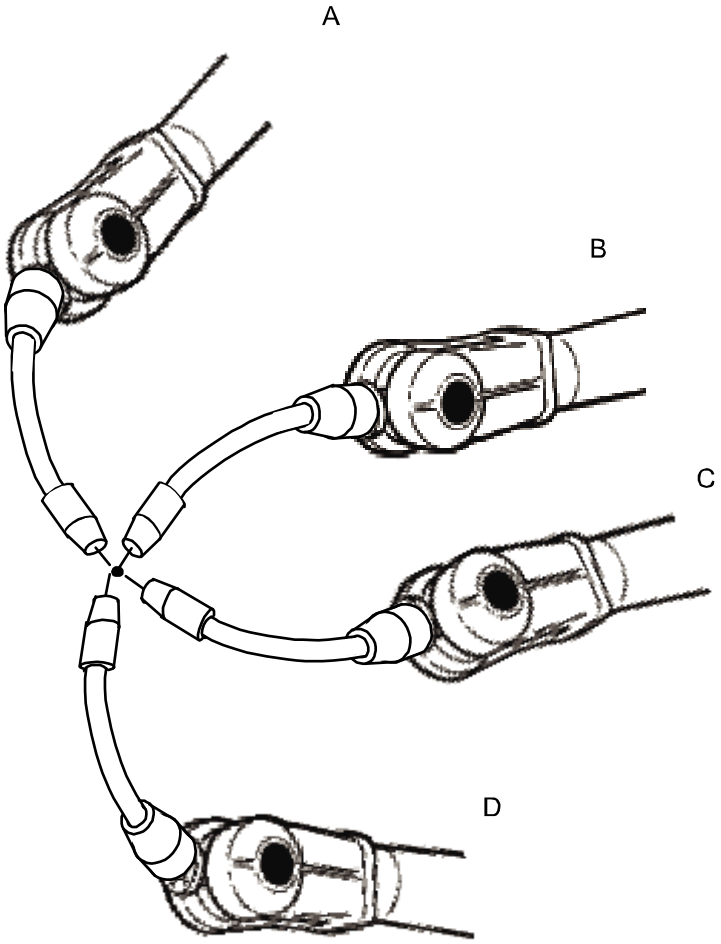
	Action
5	In the dialog box which appears, select the method to use.  en0600003147
6	Select the number of approach points to use. Usually 4 points is enough. If you choose more points to get a more accurate result, you should be equally careful when defining all of them.
7	See *How to proceed with tool frame definition on page 72* for information on how to gather positions and perform the tool frame definition.

Continues on next page

3.5.2 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

	Action	Information
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 Position** 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 73*.
5	If, for some reason, you want to redo the calibration procedure described in step 1-4, tap **Positions** and then **Reset All**.	

Continues on next page

	Action	Information
6	When all points are defined you can save them to file, which enables you to reuse them later. On the **Positions** menu, tap **Save**.	
7	Tap **OK**. 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.	For further information see *Is the calculated result good enough? on page 73*

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 75*.

	Action
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 Position** 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.

3.5.3 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 *Operating manual - IRC5 with FlexPendant*.

> **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.

Displaying the tool data

This section details how to display the tool data.

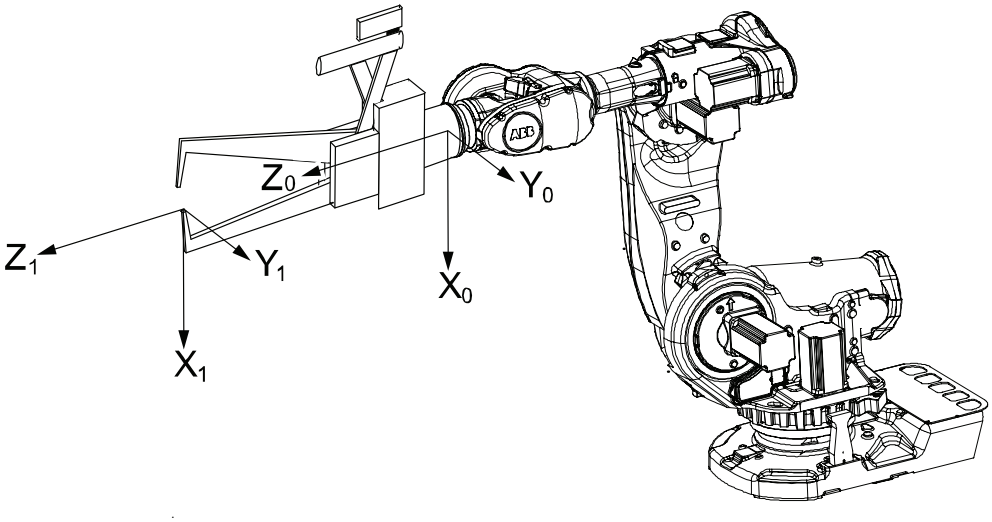
	Action
1	On the **ABB** menu, tap **Jogging**.
2	Tap **Tool** to display the list of available tools.
3	Tap the tool you want to edit, then tap **Edit**. A menu appears. • **Change Declaration** • **Change Value** • **Delete** • **Define**
4	In the menu, tap **Change Value**. The data that defines the tool appears. Green text indicates that the value can be changed.
5	Proceed with changing the data as described below.

Continues on next page

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 70*. 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.



en0400000881

X	X axis for tool0
Y	Y axis for tool0
Z	Z axis for tool0
X	X axis for the tool you want to define
Y	Y axis for the tool you want to define
Z	Z axis for the tool you want to define

	Action
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]

Continues on next page

3.5.3 Editing the tool data
Continued

	Action	Instance	Unit

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]

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 **OK** to use the new values, **Cancel** to leave the definition unchanged.		

3.5.4 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 stationery.

	Action
1	On the **ABB** menu, tap **Jogging**.
2	Tap **Tool** to display the list of available tools.
3	Tap the tool you want to edit, then tap **Edit**. A menu appears.
4	In the menu, tap **Change value**. The data that defines the tool appears.
5	Tap the instance `robhold`.
6	Tap **FALSE** to make this tool stationary.
7	Tap **OK** to use the new setup, **Cancel** to leave the tool unchanged.

Make a work object robot held

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

	Action
1	In the Jogging window, tap **Work object** to display the list of available work objects.
2	Tap the work object you want to edit, then tap **Edit**. A menu appears.
3	In the menu, tap **Change value**. The data that defines the work object appears.
4	Tap the instance `robhold`.
5	Tap **TRUE** to indicate that this work object is held by the robot.
6	Tap **OK** to use the new setup, **Cancel** to leave the work object unchanged.

Differences in coordinate system referencing

This section describes differences in coordinate system referencing.

The...	...normally references the...	...but now references the...
work object coordinate system	user coordinate system	user coordinate system (no change)
user coordinate system	world coordinate system	robot’s mounting plate
tool coordinate system	robot’s mounting plate	world coordinate system

Continues on next page

3.5.4 Setup for stationary tools
Continued

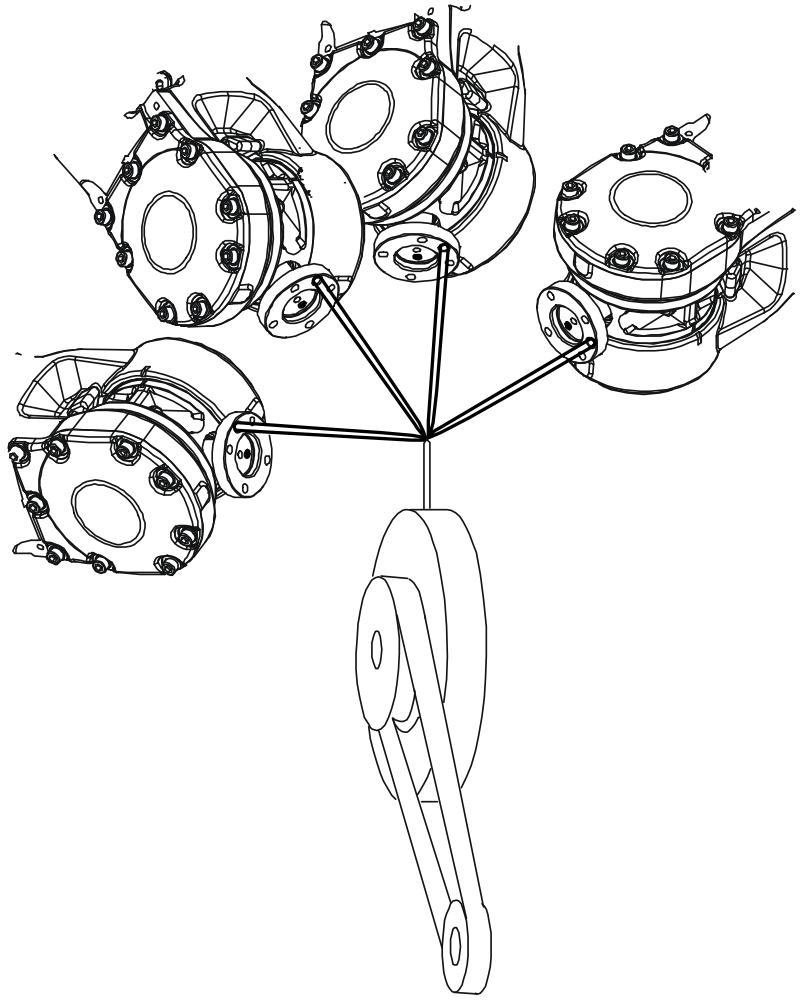
Set up the tool coordinate system

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. 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

3.6 Work objects

3.6.1 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 199*.

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 *Editing the work object declaration on page 85*.

	Action
1	On the **ABB** menu, tap **Jogging**.
2	Tap **Work Object** to display the list of available work objects.
3	Tap **New...** to create a new work object.
4	Tap **OK**.

Work object declaration settings

If you want to change...	then...	Recommendation
the work object's name	tap the ... button next to it	Work objects are automatically named `wobj` followed by a running number, for example `wobj10`, `wobj27`. 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 module	select the module in which this work object should be declared from the menu	

3.6.2 Defining the work object coordinate system

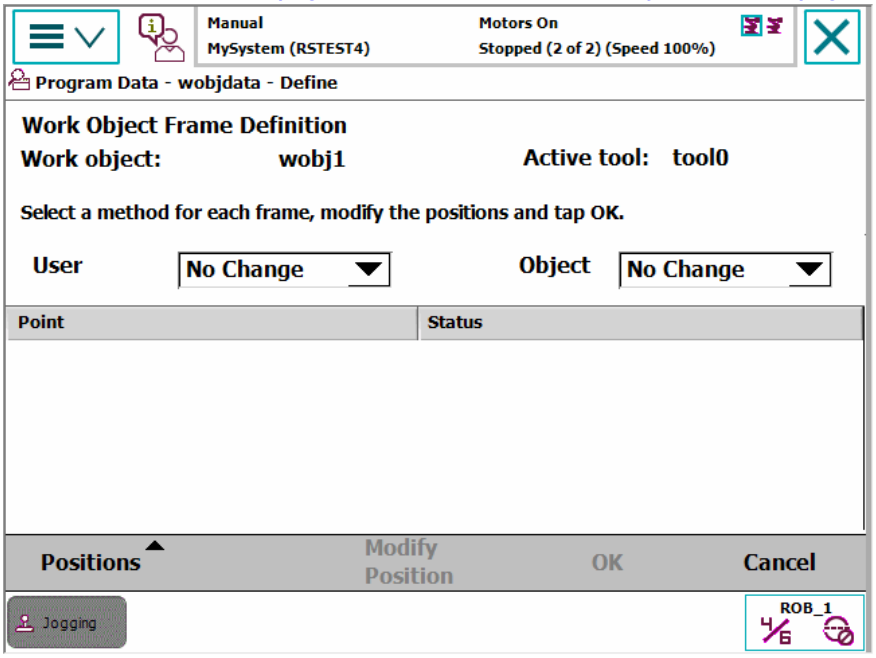
Overview

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 select method

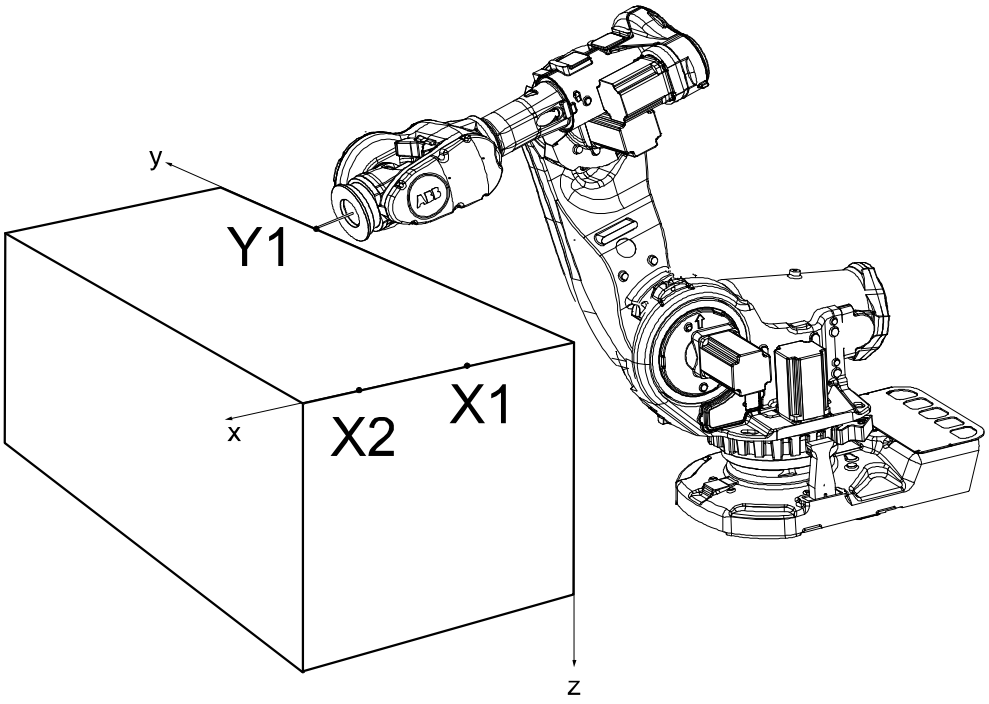
This procedure describes how to select method for defining either user frame or object frame or both. Note that this only works for a user created work object, not the default work object, wobj0. Defining work object can also be done from the **Program Data** window.

	Action
1	On the **ABB** menu, tap **Jogging**
2	Tap **Work object** to display the list of available work objects.
3	Tap the work object you want to define, then tap **Edit**.
4	In the menu, tap **Define.....**
5	Select method from the **User method** and/or the **Object method** menu. See *How to define the user frame on page 81* and *How to define the object frame on page 82* 

Continues on next page

How to define the user frame

This section details how to define the user frame.



en0400000887

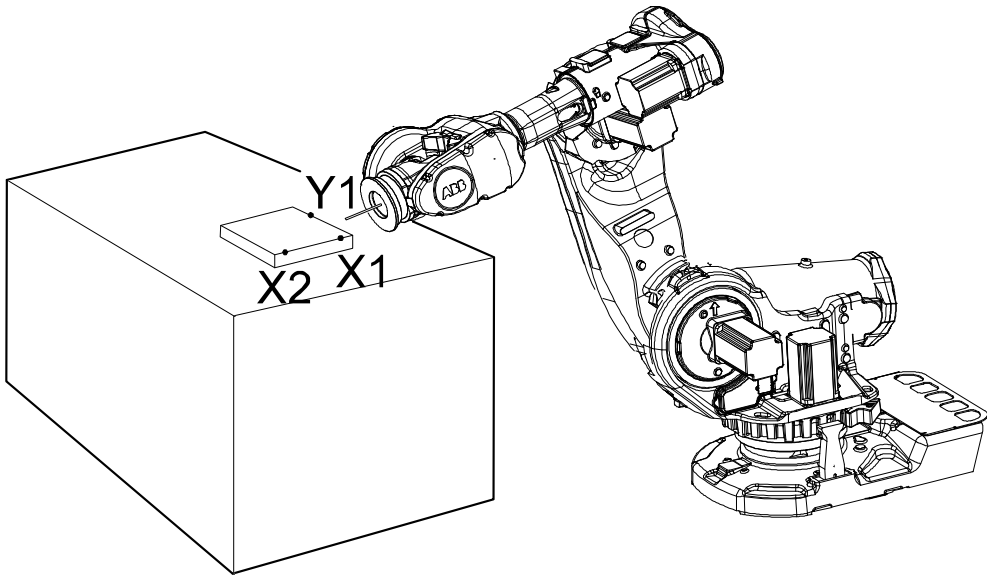
The x axis will go through points X1-X2, and the y axis through Y1.

	Action	Information
1	In the **User method** pop up menu, tap **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 Position** 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.

	Action
1	In the **Object method** pop up menu, tap **3 points**.
2	See steps 2 to 4 in the description of *How to define the user frame on page 81*.

How to save the defined positions

Normally the defined positions are only used as temporary positions by the controller to calculate the position of the work object and are then discarded. However, the positions can also be saved to a program module for later use or analysis.

When saving the positions a new program module is created where the positions are stored with predefined names given by the controller. The names of the positions can be changed afterwards, but when loading the positions it is recommended to use the predefined names.

Note

Only the positions (robtargets) are saved. Make sure to note which tool was used when modifying the defined positions.

	Action
1	When the work object frame definition is completed and all positions have been modified, tap **OK**.
2	In the **Save Modified Points** dialog, tap **Yes**.
3	Tap **ABC** to change the name of the program module, tap **OK** to accept the name.

Continues on next page

	Action
4	The names of the positions and the module is displayed in the **Save** dialog, tap **OK**.

How to load defined positions

In some cases it is not practical or possible to use the robot to define the positions. Then the positions can be defined or calculated elsewhere and loaded to the **Work Object Frame Definition** dialog.

Positions from any program module can be can be loaded, but is recommended to use the module from the **Save Modified Points** dialog with predefined position names given by the controller.

CAUTION

Make sure that the correct tool and work object is activated in the **Work Object Frame Definition** dialog before loading any positions.

	Action
1	In the **Work Object Frame Definition** dialog, tap **Positions** and **Load**.
2	Tap the module that holds the calibration points, tap **OK**.
3	If the controller finds all or any predefined positions in the module, the positions are automatically loaded to the correct user or object point. In the **Load** dialog, tap **OK**.
4	If some positions are missing or do not have the correct names, the controller cannot load the positions automatically so the user is asked to match the positions manually. Tap each point in the list to assign the positions manually from the drop down list. Tap **OK**.
5	If necessary, use **Modify Position** to define any remaining points that could not be loaded.

3.6.3 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

	Action
1	On the **ABB** menu, tap **Jogging**.
2	Tap **Work object** to display the list of available work objects.
3	Tap the work object you want to edit, then tap **Edit**.
4	Tap **Change Value.** The data that defines the work object appears.

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 80*. You can however edit the values manually using the guide below.

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 **Program Data** window.

3.6.4 Editing the work object declaration

Overview

Use the declaration to change how the work object variable can be used in the program's modules.

Displaying the work object declaration

	Action
1	On the **ABB** menu, tap **Jogging**.
2	Tap **Work object** to see the list of available work objects.
3	Tap the work object you want to edit, then tap **Edit**.
4	In the menu, tap **Change Declaration**.
5	The work object's declaration appears.
6	Edit the tool declaration as listed in section *Creating a work object on page 79*.

Note

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.

3.7 Payloads

3.7.1 Creating a payload

What happens when I create a payload?

A variable of the type `loaddata` is created. The variables 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 payloads coordinate system will be set to the position, including orientation, of the world coordinate system.

	Action
1	On the **ABB** menu tap **Jogging**.
2	Tap **Payload** or **Total Load** to display the list of available payloads. **Note** **Total Load** is displayed only when the value of **ModalPayLoadMode** is set to **0** and the mechanical units are TCP robots. See *Setting the value for ModalPayLoadMode on page 87*.
3	Tap **New** to create a new payload and enter the data. See *Payload declaration settings on page 87*.
4	Tap **OK**.

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.

Continues on next page

Payload declaration settings

If you want to change...	...then...	Recommendation
the payload's name	tap the ... button next to it	Payloads are automatically named `load` followed by a running number, for example `load10`, `load31`. 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 module	select the module in which this payload should be declared from the menu	-

Setting the value for ModalPayLoadMode

This procedure describes how to modify the value of **ModalPayLoadMode**:

1. On the **ABB** menu, tap **Control Panel** and then **Configuration**.
2. Select **Controller**.
3. Select the type **System Misc** and tap.
4. Select **ModalPayLoadMode** and then tap **Edit**.
5. Tap the parameter **Value** twice and set to **0**.
6. Click **OK**.
7. Tap **Yes** to the question **The changes will not take effect until the controller is warm started. Do you want to restart now?**.

3.7.2 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 *Operating manual - IRC5 with FlexPendant*.

Displaying the payload definition

	Action
1	On the **ABB** menu, tap **Jogging**.
2	Tap **Payload** to display the list of available payloads.
3	Tap the payload you want to edit, then tap **Edit**.
4	Tap **Change Value**. The data that defines the payload appears.

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 **OK** 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

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

3.8 Advanced programming

3.8.1 Mirroring a program, module, or routine

Mirroring

Mirroring creates a copy of a program, module, or routine in a specific mirror plane. The mirror function can be applied to any program, module, or routine.

Mirroring can be performed in two different ways:

- Default against the base frame coordinate system. The mirror operation will be performed across the xz-plane in the base frame coordinate system. All positions and work object frames that are used in an instruction in the selected program, module or routine are mirrored. The position orientation axes x and z will be mirrored.
- Advanced against a specific mirror frame. The mirror operation will be performed across the xy-plane in a specified work object frame, mirror frame. All positions in the selected program, module or routine are mirrored. If the work object argument in an instruction is another work object than specified in the mirror dialog, the work object in the instruction is used in the mirror operation. It is also possible to specify which axis in the position orientation that will be mirrored, x and z or y and z.

Mirroring is described in section *What is mirroring? on page 206*.

Mirroring a routine

This section describes how to mirror a routine.

	Action
1	In the **ABB** menu, tap **Program Editor**.
2	Tap **Edit** and tap **Mirror**.
3	To define the mirror. • Tap the **Module** menu to select in which module the routine to mirror is used. • Tap the **Routine** menu to select which routine you want to mirror. • Tap **...** to open the soft keyboard and enter the name for the new routine.
4	If you want to mirror in base frame then proceed to the next step. If you want to define another type of mirror then tap **Advanced options** and proceed as follows. To define the type of mirror: • Deselect the **Base Mirror** checkbox. • Tap **...** to the right of **Work object** to select the work object frame to which all positions which are to be mirrored are related to. • Tap **...** to the right of **Mirror frame** to select the mirror plane to which all positions will be mirrored. • Tap the **Axis to mirror** menu to specify how to mirror the position orientation. x means that x and z axes will be mirrored. y means that y and z axes will be mirrored. • Tap **OK** to save the advanced options.
5	Tap **OK**. A dialogue box is displayed.
6	Tap **Yes** to apply the selected mirror to the routine, or tap **No** to cancel.

Continues on next page

Mirroring a module or program

This section describes how to mirror a module or program.

	Action
1	In the **ABB** menu, tap **Program Editor**.
2	Tap **Edit** and tap **Mirror**.
3	To define the mirror. • Tap the **Module** menu to select module to mirror. • Tap **...** to open the soft keyboard and enter the name for the new module or program.
4	If you want to mirror in base frame then proceed to the next step. If you want to define another type of mirror then tap **Advanced options** and proceed as follows. To define the type of mirror: • Deselect the **Base Mirror** checkbox. • Tap **...** to the right of **Work object** to select the work object frame to which all positions which are to be mirrored are related to. • Tap **...** to the right of **Mirror frame** to select the mirror plane to which all positions will be mirrored. • Tap the **Axis to mirror** menu to specify how to mirror the position orientation. x means that x and z axes will be mirrored. y means that y and z axes will be mirrored. • Tap **OK** to save the advanced options.
5	Tap **OK**. A dialogue box is displayed.
6	Tap **Yes** to apply the selected mirror to the module, or tap **No** to cancel.

3.8.2 Editing instruction expressions and declarations

Expressions

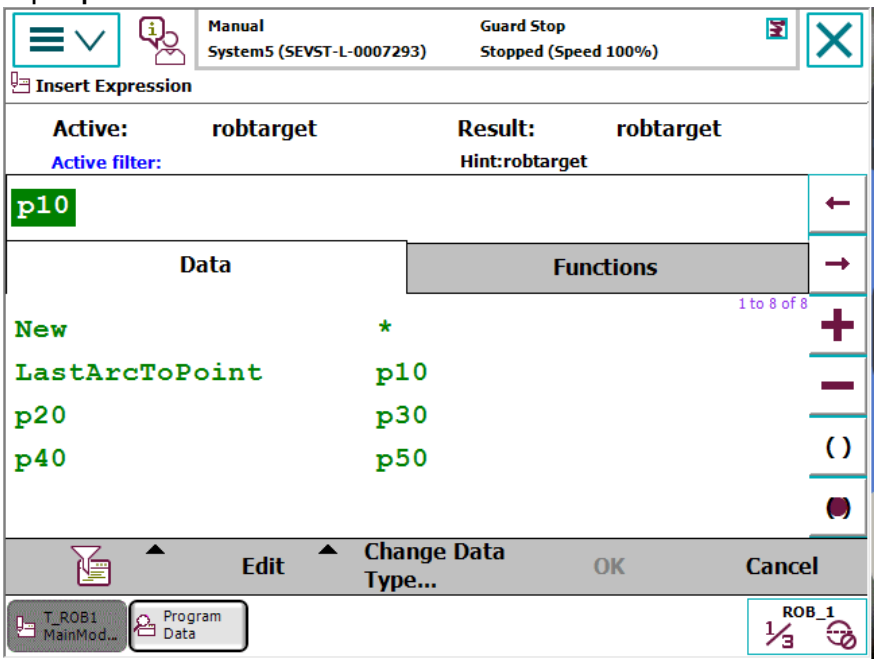
An expression specifies the evaluation of a value. It can be used, for example:

- as a condition in an IF instruction
- as an argument in an instruction
- as an argument in a function call

Read more in *Technical reference manual - RAPID Overview* and *Technical reference manual - RAPID Instructions, Functions and Data types*.

Inserting expressions

This procedure describes how to insert and edit expressions in instructions.

	Action
1	In the **Program Editor**, tap to select the instruction you want to edit and then tap **Edit**.
2	Tap **Change Selected** and tap to select the argument to change.
3	Tap **Expression**.  en0400000704
4	Edit the length of the expression by tapping the keys to the right: • Arrows: step backward and forward in the expression. • + to add expression. Tap the new expression to define it. • - to delete expression. • () to set a parenthesis around the highlighted expression. • (o) to delete a parenthesis.

Continues on next page

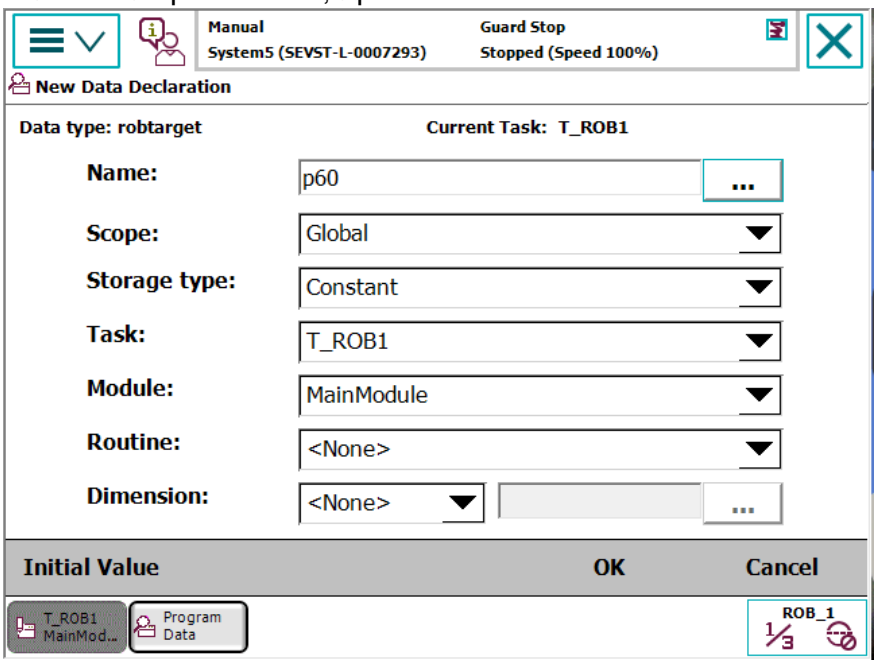
	Action
5	Tap: • **New** to create a new data declaration, i.e. adding a data declaration not previously used. This is detailed in section *Creating new data declarations on page 93*. • **View** to change views or change data type. This is detailed in section *Changing data type on page 94*. • **ABC** displays the soft keyboard.
6	Tap **OK** to save the expression.

Declarations and data types

When editing an expression, new data can be declared with the button **New**. More information about data declarations and how to edit them can be found in *Operating manual - IRC5 with FlexPendant*.

Creating new data declarations

This procedure describes how to create a new data declaration in an instruction expression.

	Action
1	In the Insert Expression view, tap **New**.  en0400000705
2	Tap and enter desired values: • **Initial value** to set the initial value. • **...** to display the soft keyboard and change the data type's name. • **Scope** • **Storage type** • **Module** • **Routine** • **Dimension** to set the size of an array if the data type should be an array. • If a value has been chosen for Dimension, tap **...** to set array size.

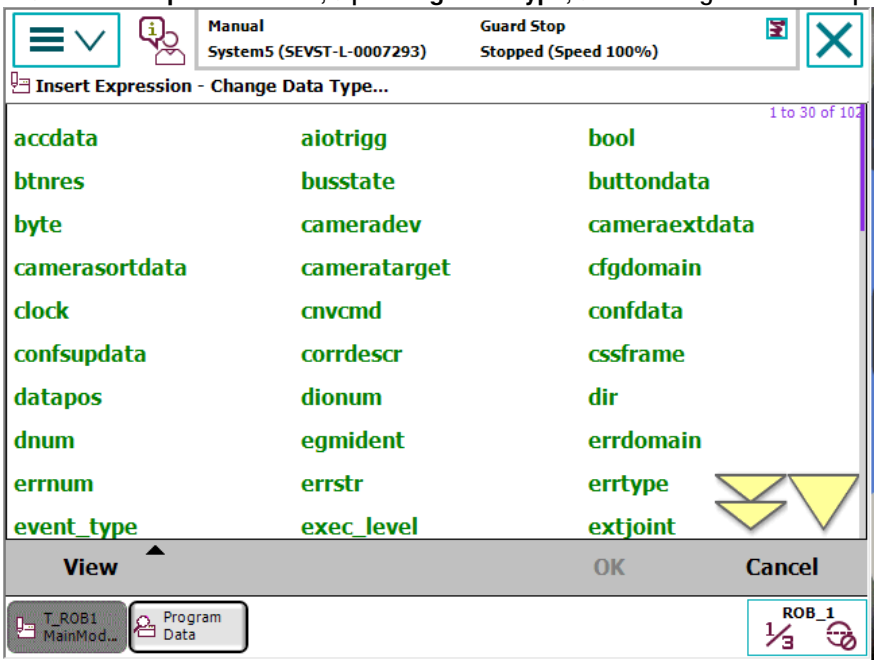
Continues on next page

3.8.2 Editing instruction expressions and declarations
Continued

	Action
3	After making all selections, tap **OK**. A dialog box is displayed, prompting you to allow resetting of the program pointer and applying all changes: • Tap **Yes** to proceed. • Tap **No** to return to the data type view without resetting of the program pointer or applying changes.

Changing data type

This section describes how to change data type.

	Action
1	In the **Insert Expression** view, tap **Change data type**, the following screen is displayed:  en0400000706
2	Tap to select the required data type and tap **OK**.

3.8.3 Hiding declarations in program code

Declarations

Program declarations can be hidden to make the program code easier to read.

Hiding declarations

This section describes how to hide or show declarations.

	Action
1	In the **ABB** menu, tap **Program Editor** to view a program.
2	Tap **Hide Declarations** to hide declarations. Tap **Show Declarations** to show declarations.

3.8.4 Deleting programs from memory

Overview

Deleting a program in a task does not erase the program from the controller mass memory but only from the program memory.

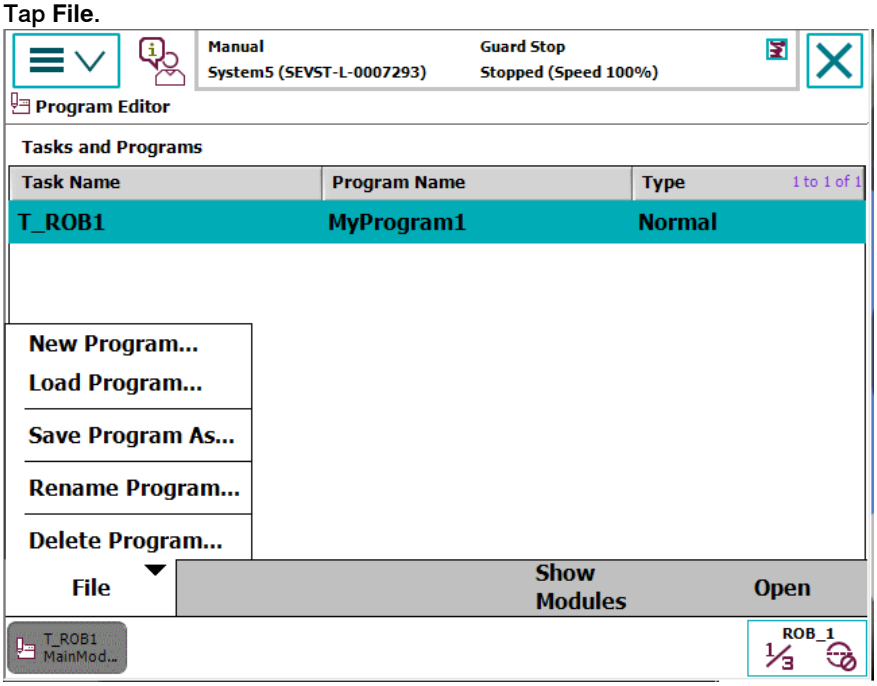
When you switch programs, the previously used program is deleted from the program memory, but not removed from the mass memory if it was saved there.

How to save your work is detailed in section *Handling of programs on page 54*.

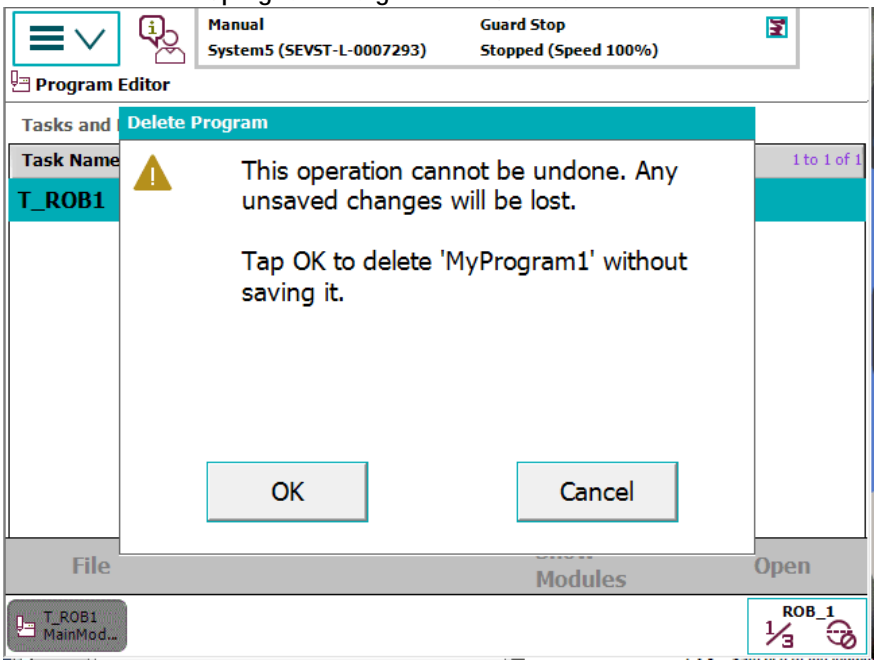
The different memories are described in section *What is “the memory”? on page 112*.

Deleting programs from memory

This section details how to delete programs from the program memory.

	Action
1	On the ABB menu tap **Program Editor**.
2	Tap **Tasks and Programs**.
3	Tap **File**.  en0400000678

Continues on next page

	Action
4	Tap **Delete Program...**. **WARNING!** Recent program changes will not be saved.  en0400000853
5	Tap **OK**. If you don't want to lose information about program changes then use **Save Program** before deleting the program. How to save your work is described in section *Handling of programs on page 54*.

3.8.5 Deleting programs from hard disk

Overview

Programs are deleted via **FlexPendant Explorer** or an FTP client. When deleting programs from the controller hard disk, the currently loaded program in the program memory is not affected.

The different memories are described in section *What is “the memory”? on page 112*.

Deleting programs with FlexPendant Explorer

Programs can be deleted using **FlexPendant Explorer**. For information about the **FlexPendant Explorer**, see section *Operating manual - IRC5 with FlexPendant*.

3.8.6 Activating mechanical units

Overview

A mechanical unit can be active or deactive. Only active units are run when executing a program. Deactivated units will not run. This may be useful when programming or testing a program.

A robot **cannot** be deactivated.

The Activate function does not affect jogging. To select mechanical unit for jogging, use the **Mechanical unit** property in the **Jogging** menu.

Activating mechanical units

This procedure describes how to activate a mechanical unit.

	Action	Information
1	On the **ABB** menu, tap **Jogging**.	
2	Make sure that the right mechanical unit is selected, then tap **Activate...**. To deactivate an active mechanical unit, tap **Deactivate**.	A robot cannot be deactivated.

Related information

Mechanical units can be active or deactive at start depending on the system setup, see *Technical reference manual - System parameters*, topic *Motion*.

3.9 Hide RAPID arguments

Overview

Instructions with a lot of arguments can be difficult to view on the FlexPendant. To easily view the program on the FlexPendant it is possible to hide the arguments (both optional and mandatory) in the Editing window and in the Production window.

The mandatory (required) arguments can be hidden only in the RAPID Editor/Production Window on the FlexPendant. When a RAPID routine or instruction is expanded all the mandatory arguments are displayed.

Usage

Introduction

To hide RAPID arguments you need to edit the RAPID metadata XML file (`rapid_edit_rules.xml`). The location of this file is $HOME directory of the controller.

The following is an example of RAPID metadata XML file.

```
<?xml version="1.0" encoding="utf-8"?>
<Rapid>
<Edit>
<Instruction name="Instr1">
<Argument name="ReqArg1" showeditor="true" />
<Argument name="ReqArg2" showeditor="false" />
<Argument name="OptArg1" show="false" showeditor="false" />
<Argument name="OptArg2" show="false" showeditor="true" />
<Argument name="OptArg3" show="true" showeditor="false" />
<Argument name="OptArg4" show="true" showeditor="true" />
</Instruction>
</Edit>
</Rapid>
```

Note

The purpose of hiding optional/mandatory arguments is ease-of-use. If no value is specified, by default the parameters are displayed.

Hide optional parameter in the argument window

To hide an optional parameter in the expanded view, set the "show" flag to "false" in the RAPID metadata xml file.

Hide optional parameter in the RAPID Editor and the Production window

To hide an optional parameter in editors, set the "showeditor" flag to "false" in the RAPID metadata xml file.

Hide mandatory parameter in the RAPID Editor and the Production window

To hide a mandatory (required) parameter in editors, set the "showeditor" flag to "false" in the RAPID metadata xml file.

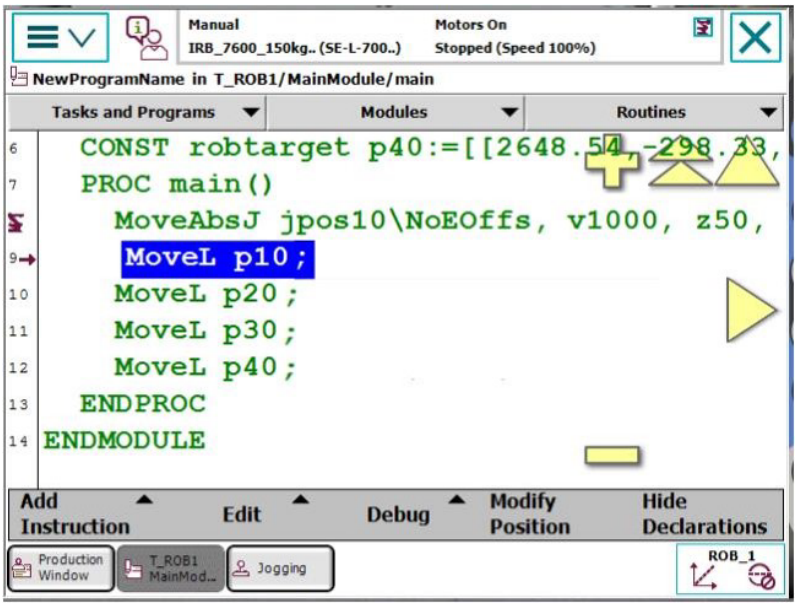
Continues on next page

Example - Edit `rapid_edit_rules.xml`

To hide arguments present in the MoveL instruction following is the xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<Rapid>
<Edit>
<Instruction name="MoveL">
<Argument name="Speed" showeditor="false" />
<Argument name="Zone" showeditor="false" />
<Argument name="Tool" showeditor="false" />
<Argument name=”WObj" showeditor="false" show=”true” />
<Argument name=”ID" showeditor="false" show=”false” />
</Instruction>
</Edit>
</Rapid>
```

Once this xml is configured only the robtargets are shown for MoveL as the other arguments are hidden from the XML file.

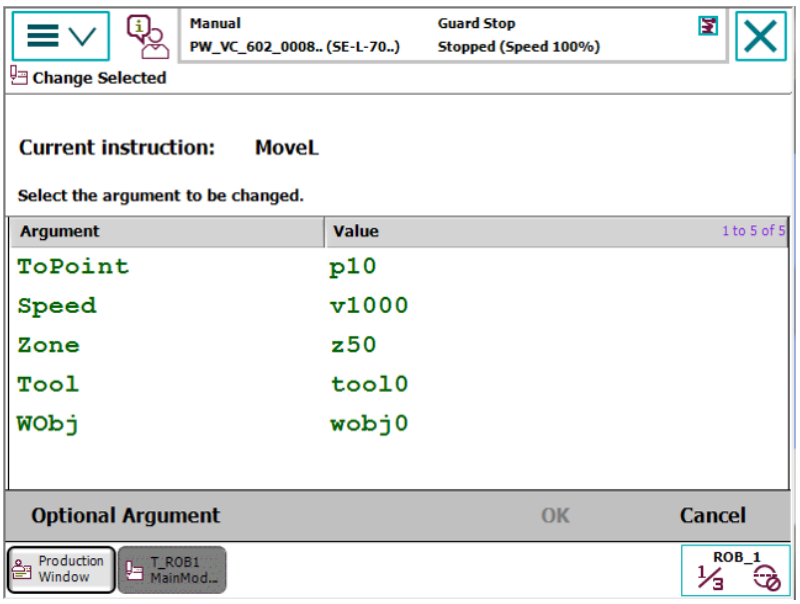


xx1500001585

Continues on next page

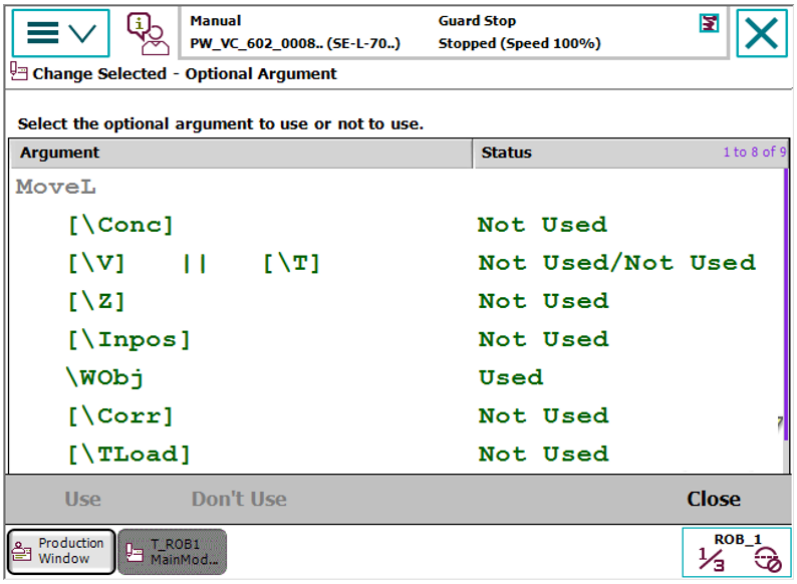
3.9 Hide RAPID arguments
Continued

When an instruction is expanded only arguments configured are displayed as shown in the following image.



xx1500001586

The optional argument ID is not displayed as shown in the following image, since its “show” attribute was set to “false”.



xx1500001587

4 Handling inputs and outputs, I/O

4.1 Basic procedures

4.1.1 Configuring I/O

Creating and editing buses, units and signals

The configuration of I/O is slightly different for different fieldbuses. How to create and edit buses, units and signals are described in the respective fieldbus manuals.

4.1.2 Deactivating and activating I/O units

Overview

Deactivating an I/O unit makes the controller ignore the unit. This can be useful during commissioning, for avoiding errors if the I/O unit is not connected to the controller yet. The signals configured on the unit will still be visible when it is deactivated, but the signal values will not be available. The controller will not attempt to send or receive any signals on a deactivated unit.

Activating the unit again will take it back to normal operation.

Deactivating and activating I/O units

This section describes how to activate I/O units.

	Action
1	On the **ABB** menu tap **Inputs and outputs**. The list of Most common I/O signals is displayed.
2	Tap **View** to change the selection of signals in the list. Select **I/O Units**.
3	Tap an I/O unit in the list.
4	Tap **Activate** or **Deactivate**.

Note

All signals on the I/O unit must have an access level that allows local clients (for instance the FlexPendant) to have write access. If not, then the unit cannot be activated or deactivated from local clients. The access level is set with system parameters for each signal, see the types *Signal* and *Access Level* in the topic *I/O*.

Note

The unit cannot be deactivated if the system parameter *Unit Trustlevel* is set to *0 (Required)*. *Unit Trustlevel* belongs to the type *Unit* in the topic *I/O*.

Related information

For information on how to configure an I/O unit (for instance to add and remove signals or to set the limits of the signal), see *Configuring system parameters on page 141*.

Technical reference manual - System parameters.

4.1.3 Alias I/O signals

Introduction

AliasIO is used to define a signal of any type with an alias name. After the AliasIO instruction is executed in the RAPID program, the Alias I/O signal can be viewed from the Alias I/O menu in the same way as the other signals from the View menu.

Creating new signal data

This section describes how to create new signal data instances:

	Action
1	On the **ABB** menu, tap **Program Data**.
2	Tap **View** and select **All Data Types**. A list of all available data types are displayed.
3	Select **signaldi** and tap **Show Data.**
4	Tap **New**. The New Data Declaration screen appears.
5	Tap to the right of **Name** and define the name of data instance. For example, **alias_di1**.
6	Tap the **Scope** menu and select **Global**.
7	Tap **OK**.
8	Repeat steps 1 to 7 to create **signaldo** data instance. For example, **alias_do1**.

Note

Consider the following example,

"VARsignaldoalias_do1;"

"AliasIOdo_1, alias_do1;"

VAR declaration must be done global in the module.

After declaring **signaldi** and **signaldo** and executing the instruction **AliasIO do_1, alias_do1** the **alias_do1** signal is displayed in the AliasI/O menu in the same way the ordinary signals are displayed in the **View** menu.

The **alias_do1** signal is active as long as the RAPID program is active and is displayed after the AliasIO instruction is executed.

Adding AliasIO

This section describes how to add AliasIO instructions and to view them on **Alias I/O** menu:

	Action
1	Load the system parameters file *Eio.cfg*. For more information on loading system parameters and adding parameters from a file, see *Configuring system parameters on page 141*.
2	Declare the RAPID variables of data type **signaldi** and **signaldo** i.e **alias_di1** and **alias_do1** respectively as described in *Creating new signal data on page 105*.
3	On the **ABB** menu, tap **Program Editor**.

Continues on next page

4.1.3 Alias I/O signals

Continued

	Action
4	Tap to highlight the instruction under which you want to add a new instruction.
5	Tap **Add Instruction**. A category of instructions is displayed. A large number of instructions, divided into several categories are available.
6	Tap **I/O** to display a list of the available categories. You can also tap **Previous**/**Next** at the bottom of the list of instructions to move to theprevious/next category.
7	Select **AliasIO**.
8	Select **signaldi** data type and click **OK**.
9	Tap **<EXP>** and select the argument value (for example, **di01_Box**) for the **From Signal** argument from the configuration file.
10	Tap **<EXP>** for the **To Signal** argument.
11	Tap **Edit** and select **Change data type**.
12	Select **signaldi** from the list and click **OK**.
13	Select the argument value (for example, **alias_di1**) for the **To Signal** argument as created from step 1.
14	Tap **Debug** and select **PP to Main** from the Program Editor.
15	Press the **Start** button on the FlexPendant to run the program.
16	On the **ABB** menu, tap **Inputs and Outputs**.
17	Tap **View** and select **Alias I/O** menu. The new data declaration created from the RAPID program should be available. If not, click **Refresh**.
18	Repeat steps 7 to 16 to add **signaldo** data type and view from **Alias I/O** menu.

Note

Currently only digital signals **signaldi** and **signaldo** are supported.

4.2 Safety signals

4.2.1 Safety I/O signals

General

In the IRC5 controller's basic and standard form, certain **I/O** signals are dedicated to specific safety functions. These are listed below with a brief description of each.

All signals can be viewed in the **I/O** menu on the FlexPendant.

Safety I/O signals

The list below contains the safety I/O signals as used by the standard system.

Signal name	Description	Bit value condition	From - To
ES1	Emergency stop, chain 1	1 = Chain closed	From panel board to main computer
ES2	Emergency stop, chain 2	1 = Chain closed	From panel board to main computer
SOFTESI	Soft Emergency stop	1 = Soft stop enabled	From panel board to main computer
EN1	Enabling device1&2, chain 1	1 = Enabled	From panel board to main computer
EN2	Enabling device1&2, chain 2	1 = Enabled	From panel board to main computer
AUTO1	Op mode selector, chain 1	1 = Auto selected	From panel board to main computer
AUTO2	Op mode selector, chain 2	1 = Auto selected	From panel board to main computer
MAN1	Op mode selector, chain 1	1 = MAN selected	From panel board to main computer
MANFS1	Op mode selector, chain 1	1 = Man. full speed selected	From panel board to main computer
MAN2	Op mode selector, chain 2	1 = MAN selected	From panel board to main computer
MANFS2	Op mode selector, chain 2	1 = Man. full speed selected	From panel board to main computer
USERDOOVLD	Over load, user DO	1 = Error, 0 = OK	From panel board to main computer
MONPB	Motors-on pushbutton	1 = Pushbutton pressed	From panel board to main computer
AS1	Auto stop, chain 1	1 = Chain closed	From panel board to main computer
AS2	Auto stop, chain 2	1 = Chain closed	From panel board to main computer
SOFTASI	Soft Auto stop	1 = Soft stop enabled	From panel board to main computer
GS1	General stop, chain 1	1 = Chain closed	From panel board to main computer

Continues on next page

4.2.1 Safety I/O signals

Continued

Signal name	Description	Bit value condition	From - To
GS2	General stop, chain 2	1 = Chain closed	From panel board to main computer
SOFTGSI	Soft General stop	1 = Soft stop enabled	From panel board to main computer
SS1	Superior stop, chain1	1 = Chain closed	From panel board to main computer
SS2	Superior stop, chain2	1 = Chain closed	From panel board to main computer
SOFTSSI	Soft Superior stop	1 = Soft stop enabled	From panel board to main computer
CH1	All switches in run chain 1 closed	1 = Chain closed	From panel board to main computer
CH2	All switches in run chain 2 closed	1 = Chain closed	From panel board to main computer
ENABLE1	Enable from MC (read back)	1 = Enable, 0 = break chain 1	From panel board to main computer
ENABLE2_1	Enable from AXC1	1 = Enable, 0 = break chain 2	From panel board to main computer
ENABLE2_2	Enable from AXC2	1 = Enable, 0 = break chain 2	From panel board to main computer
ENABLE2_3	Enable from AXC3	1 = Enable, 0 = break chain 2	From panel board to main computer
ENABLE2_4	Enable from AXC4	1 = Enable, 0 = break chain 2	From panel board to main computer
PANEL24OVLD	Overload, panel 24V	1 = Error, 0 = OK	From panel board to main computer
DRVOVLD	Overload, drive modules	1 = Error, 0 = OK	From panel board to main computer
DRV1LIM1	Read back of chain 1 after limit switches	1 = Chain 1 closed	From axis computer to main computer
DRV1LIM2	Read back of chain 2 after limit switches	1 = Chain 2 closed	From axis computer to main computer
DRV1K1	Read back of contactor K1, chain 1	1 = K1 closed	From axis computer to main computer
DRV1K2	Read back of contactor K2, chain 2	1 = K2 closed	From axis computer to main computer
DRV1EXTCONT	External contactors closed	1 = Contactors closed	From axis computer to main computer
DRV1TEST1	A dip in run chain 1 has been detected	Toggled	From axis computer to main computer
DRV1TEST2	A dip in run chain 2 has been detected	Toggled	From axis computer to main computer
SOFTESO	Soft Emergency stop	1 = Set soft E-stop	From main computer to panel board
SOFTASO	Soft Auto stop	1 = Set soft Auto stop	From main computer to panel board
SOFTGSO	Soft General stop	1 = Set soft General stop	From main computer to panel board

Continues on next page

Signal name	Description	Bit value condition	From - To
SOFTSSO	Soft Superior stop	1 = Set soft Sup. E-stop	From main computer to panel board
MOTLMP	Motors-on lamp	1 = Lamp on	From main computer to panel board
TESTEN1	Test of Enable1	1 = Start test	From main computer to panel board
DRV1CHAIN1	Signal to interlocking circuit	1 = Close chain 1	From main computer to axis computer 1
DRV1CHAIN2	Signal to interlocking circuit	1 = Close chain 2	From main computer to axis computer 1
DRV1BRAKE	Signal to brake-release coil	1 = Release brake	From main computer to axis computer 1

This page is intentionally left blank

5 Systems

5.1 What is a system?

The system

A system is the software that runs on a controller. It consists of the specific RobotWare parts for the robots connected to the controller, configuration files, and RAPID programs.

The RobotWare license

What parts of RobotWare (supported robot models, options, etc.) that is included in the system is determined by the RobotWare license.

When running a system on a real controller it has to be built with the license that was delivered with the robot.

For running a system on a virtual controller (e.g. for simulations in RobotStudio) either a license from a real robot or a virtual license can be used. Using a license from a real robot is a quick way to ensure that the system matches that robot. Using a virtual license provides possibility to simulate and evaluate any robot model with any configuration. A system built with a virtual license can however never be run on a real controller.

Empty system

A new system that only contains the RobotWare parts and the default configurations is called an empty system. When robot or process specific configurations are made, I/O signals are defined or RAPID programs are created, the system is no longer considered empty.

Loaded system and stored systems

The loaded system is the system that will run on the controller when it is started. A controller can only have one system loaded, but additional systems can be stored on the controller’s disk or any disk on the PC network.

It is when a system is loaded, either in a real controller or a virtual one, you normally edit its content, like RAPID programs and configurations. For stored systems, you can make some changes with the System Builder in RobotStudio, like adding and removing options and replacing whole configuration files.

5.2 Memory and file handling

5.2.1 What is “the memory”?

Overview

When using the term “memory”, different things may be implied:

- The controller mass memory unit (hard disk, flash disk, or other drive)
- The hard disk of some other unit connected to the same LAN as the robot system, serving as a storage for software.

Controller mass memory unit

This is the main mass storage unit of the controller, i.e. the controller mass memory. Depending on controller version, it may be a flash disk, hard drive, or other type and it may vary in size. It contains all necessary software for operating the robot, and is the unit on which RobotWare is installed.

Note

The default file system of **FlexPendant Explorer** is:

- /hd0a : file system on a hard disk (persistent)

LAN unit

This may be used as extra mass storage device if the one in the controller is not sufficient. It is not normally considered a part of the robot system.

5.2.2 File handling

File handling and storing

Backups, programs, and configurations etc. are saved as files in the robot system. These files are handled either in a specific FlexPendant application, such as the **Program Editor**, or using the **FlexPendant Explorer**.

Files can be stored on a number of different drives, or memory devices, such as:

- Controller mass memory unit
- USB device
- Network drive

These drives are all used in the same way and available in the FlexPendant Explorer or when saving or opening files using an application on the FlexPendant.

Note

The default file system of **FlexPendant Explorer** is:

- /hd0a : filesystem on a hard disk (persitent)

USB memory information

IRC5 is equipped with a USB port on the FlexPendant, see *The FlexPendant on page 19*. There are also USB ports on the main computer in the IRC5 controller.

A USB memory is normally detected by the system and ready to use within a few seconds from plugging in the hardware. A plugged in USB memory is automatically detected during system start.

It is possible to plug in and unplug a USB memory while the system is running. However, observing the following precautions will avoid problems:

- Do not unplug a USB memory immediately after plugging in. Wait at least five seconds, or until the memory has been detected by the system.
- Do not unplug a USB memory during file operations, such as saving or copying files. Many USB memories indicate ongoing operations with a flashing LED.
- Do not unplug a USB memory while the system is shutting down. Wait until shutdown is completed.

Please also note the following limitations with USB memories:

- There is no guarantee that all USB memories are supported.
- Some USB memories have a write protection switch. The system is not able to detect if a file operation failed due to the write protection switch.

Continues on next page

5.2.2 File handling
Continued

Limitations

The maximum length for a file name is 99 characters and the maximum length for a file path including the file name is 247 characters.

Note

Some additional options may have other restrictions on the length of file names and file paths. For more information see *Application manual - Controller software IRC5*.

Related information

Operating manual - Troubleshooting IRC5.

What is “the memory”? on page 112.

5.2.3 Setting default paths

Introduction to default paths

You can set individual default paths for some actions using the FlexPendant.

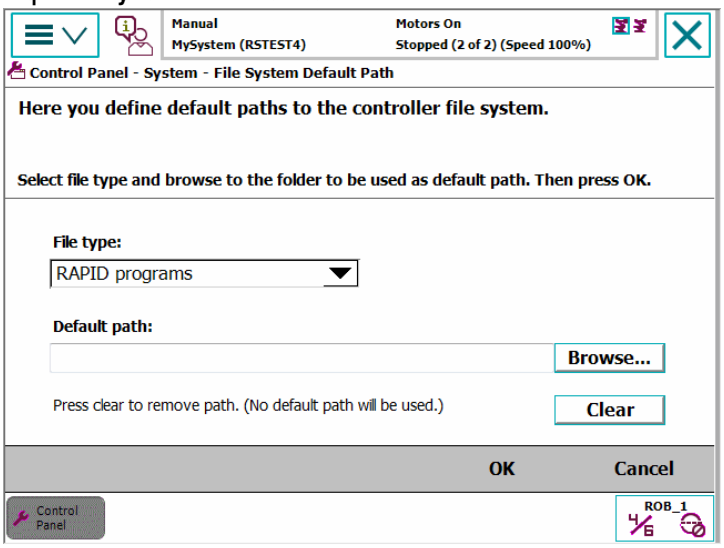
The following default paths can be set:

- Saving and loading RAPID programs.
- Saving and loading RAPID modules.
- Saving and storing configuration files.

This function is available if the user that is logged on is authorized. User authorization is handled via RobotStudio. See *Operating manual - RobotStudio*.

Setting default paths

Use this procedure to set a default path.

	Action
1	On the **ABB** menu, tap **Control Panel** and then **FlexPendant**.
2	Tap **File System Default Path**.  en0500002361
3	Tap the **File type** menu to choose type of default path: • RAPID programs • RAPID modules • Configurations files
4	Type the default path or tap **Browse**, to choose the desired location.
5	If required, any previously entered path can be removed by tapping **Clear**.
6	Tap **OK**.

5.3 Restart procedures

5.3.1 Restart overview

When do I need to restart a running controller?

ABB robot systems are designed to operate unattended for long times. There is no need to periodically restart functioning systems.

Restart the robot system when:

- new hardware has been installed.
- the robot system configuration files have been changed.
- a new system has been added and is to be used.
- a system failure (SYSFAIL) has occurred.

Restart types

A number of restart types are available:

Situation:	Restart type:	Detailed in section:
You want to restart and use the current system. All programs and configurations will be saved.	**Restart**	*Restart and use the current system on page 121*.
You want to restart and select another system. The Boot Application will be launched at start. **NOTE:** For system using RobotWare 5.14 or above, the required system can be directly selected, see *Managing Installed Systems on page 127*.	**Start Boot Application**	*Restart and start boot application on page 122*.
You want to delete all user loaded RAPID programs. **Warning!** *This can not be undone.*	**Reset RAPID**	*Restart and reset RAPID on page 123*.
You want to return to the default system settings. **Warning!***This will remove all user defined programs and configurations from memory and restart with default factory settings.*	**Reset system**	*Restart and reset system on page 124*.
The system has been restarted and you want to restart the current system using the image file (system data) from the most recent successful shut down.	**Revert to last auto saved**	*Restart and revert to last auto saved on page 125*.
You want to shut down and save the current system and shut down the main computer.	**Shutdown main computer**	See section *Shut down* in *Operating manual - IRC5 with FlexPendant*.

Related information

Operating manual - Troubleshooting IRC5.

5.3.2 Using the Boot Application

Boot Application

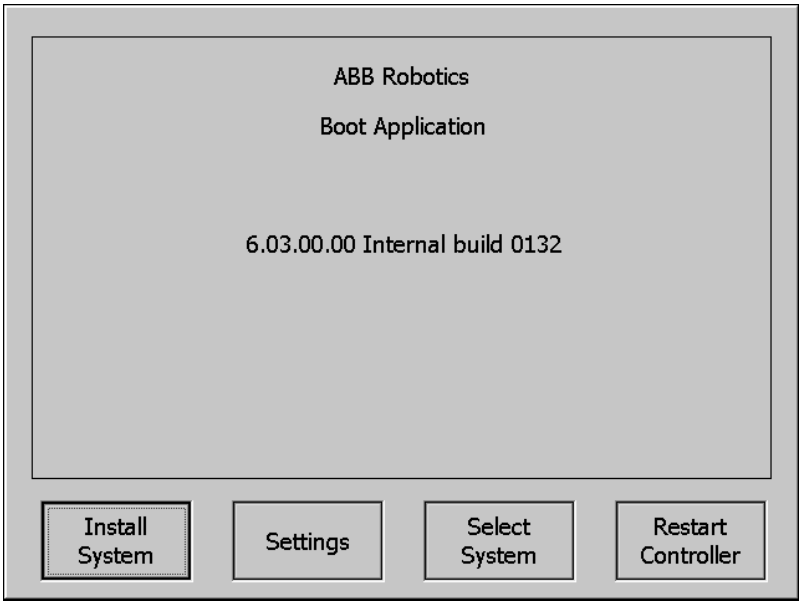
The Boot Application is primarily used to start the system when no RobotWare is installed, but can also be used for other purposes, such as changing the system to start. You can also use RobotStudio, see *Operating manual - RobotStudio*.

Purpose of the Boot Application

The Boot Application is installed at delivery and can be used to:

- Install systems.
- Set or check network settings.
- Rename controller
- Select a system/switch between systems from the mass storage memory.
- Load the system from USB memory units or network connections.

The illustration shows the Boot Application main screen. The buttons and functions available are described below.



xx1600000248

Installing a system

This procedure can take several minutes.

	Action	Information
1	Perform a restart and select the option **Start Boot Application**.	**Start Boot Application** is described in section *Restart and start boot application on page 122*.
2	Tap **Install System**.	
3	Connect a USB memory containing a system to the USB port and tap **Continue**.	USB ports are found on the FlexPendant and on the main computer in the controller.

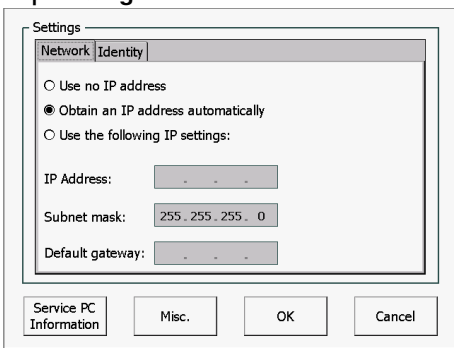
Continues on next page

5.3.2 Using the Boot Application
Continued

	Action	Information
4	Tap ... to the right of the Path text box to locate the system folder on the USB memory. Select a system folder and then tap **OK**.	
5	Tap **Continue** to start the copying.	
6	When the copying is complete, tap **OK**.	The USB memory can be disconnected at this point.
7	When asked to complete the installation, tap **OK**. The controller is now restarted with the system. The restart can take several minutes.	

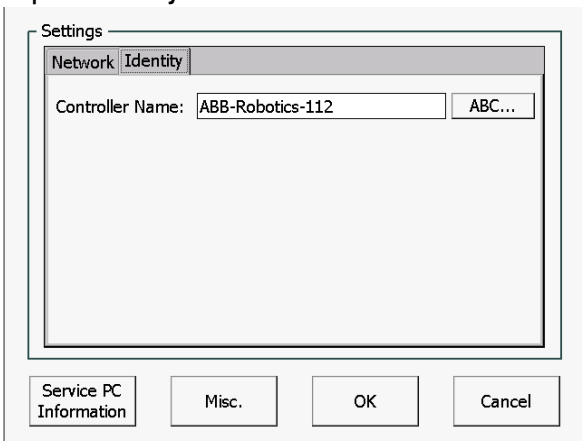
Boot Application settings

The Boot Application settings contain IP and network settings.

	Action	Information
1	Perform a restart and select the option **Start Boot Application**.	**Start Boot Application** is described in section *Restart and start boot application on page 122*.
2	Tap **Settings**.  en0400000902	
3	In the **Network** tab enter the following settings: • Use no IP address • Obtain IP address automatically • Use the following settings Use the numerical keyboard to enter the desired values.	These settings are detailed in section *Set up the network connection on page 41*.
4	Tap **Service PC information** to display network settings to be used when connection a service PC to the controller service port.	
5	Tap **Misc.** to display FlexPendant hardware and software versions.	

Continues on next page

Renaming a controller

	Action	Information
1	Perform a restart and select the **Start Boot Application** option.	**Start Boot Application** is described in section *Restart and start boot application on page 122*.
2	Tap **Settings**. The **Settings** window is displayed.	
3	Tap the **Identity** tab.  xx1700000331	
4	In the **Controller Name** field type a new name for the controller.	
5	Tap **OK**. The controller is renamed and the changes are saved.	

Selecting a system

	Action	Information
1	Perform a restart and select the option **Start Boot Application**.	**Start Boot Application** is described in section *Restart and start boot application on page 122*.
2	Tap **Select System**. A dialog box is displayed showing the available installed systems.	
3	Tap a system and then tap **Select**. The selected system is displayed in the box Selected System.	
4	Tap **Close**. A dialog box is shown urging you to restart to be able to use the selected system.	

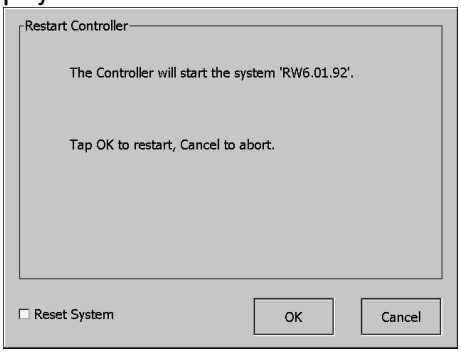
Restarting controller

	Action	Information
1	Restart the controller to the Boot Application mode.	For more details about Boot application mode, see *Restart and start boot application on page 122*.

Continues on next page

5.3.2 Using the Boot Application

Continued

	Action	Information
2	Tap **Restart System**. The **Restart Controller** window is displayed.  xx1500000343	To restart the controller and reset the selected system, select the **Reset System** checkbox.
3	Tap **OK**.	If the **Reset System** checkbox is selected, the controller is restarted with the default system settings. Otherwise the controller is restarted without any change in the system settings.

Related information

Operating manual - RobotStudio.

5.3.3 Restart and use the current system

What happens with my current system?

The current system will be stopped.

All system parameters and programs will be saved to an image file.

During the restart process the system's state will be resumed. Static and semistatic tasks will be started. Programs can be started from the point they where stopped.

Restarting this way will activate any configuration changes entered using RobotStudio.

Restart and use the current system

This section describes how to restart and use the current system.

	Action	Information
1	On the **ABB** menu, tap **Restart**. The restart dialog is displayed.	
2	Tap **Restart** to restart the controller using the current system.	To select another type of start, tap **Advanced**. Detailed information about advanced starts is given in *Restart overview on page 116*.

5.3.4 Restart and start boot application

What happens with my current system?

The current system will be stopped.

All system parameters and programs will be saved to an image file, so that the system state can be resumed later.

Restart and select another system

This section describes how to restart and select another system.

	Action	Information
1	Make sure the power to the controller cabinet is switched on.	
2	On the **ABB** menu, tap **Restart**. The restart dialog is displayed.	
3	Tap **Advanced...** to select restart method. The select restart method dialog is displayed.	
4	Tap **Start Boot Application**, then tap **Next**. A confirmation dialog is displayed.	
5	Tap **Start Boot Application** to restart the controller. The controller is restarted. After the start procedure the Boot Application is started.	
6	Use the Boot Application to select system.	How to use the Boot Application is detailed in *Using the Boot Application on page 117*.
7	Tap **Close**, then **OK** to return to the Boot Application.	
8	Tap **Restart** to restart the controller using the selected system.	

5.3.5 Restart and reset RAPID

What happens with my current system?

After restart the system's 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 set configuration. System parameters will not be affected.

Restart and delete programs and modules

This section describes how to restart and delete user loaded programs and modules.

	Action
1	On the **ABB** menu, tap **Restart**. The restart dialog is displayed.
2	Tap **Advanced...** to select restart method. The select restart method dialog is displayed.
3	Tap **Reset RAPID**, then tap **Next**. A dialog letting you confirm that you really want to restart is displayed.
4	Tap **Reset RAPID** to restart the controller. The controller is restarted using the current system. After the start procedure no programs or modules are open.

5.3.6 Restart and reset system

What happens to my current system?

After restart, the system's state will be resumed but any changes done to system parameters and other settings will be lost. Instead, system parameters and other settings are read from the originally installed system on delivery.

For example, this returns the system to the original factory system state.

Restart and return to default settings

This section describes how to restart and return to default settings.

	Action
1	On the **ABB** menu, tap **Restart**. The restart dialog is displayed.
2	Tap **Advanced...** to select restart method. The select restart method dialog is displayed.
3	Tap **Reset system**, then tap **Next**. A dialog letting you confirm that you really want to restart is displayed.
4	Tap **Reset system** to restart the controller. The controller is restarted using the current system. Changes to system parameters and other settings are lost.

5.3.7 Restart and revert to last auto saved

What happens with my current system?

After restart the system uses the backup of the image file from the last successful shut down. This means that all changes made to the system after that successful shut down are lost.

When to use revert to last auto saved

Revert to last auto saved should be used if the controller was shut down without successfully saving the image file and you want to restart the same system again. However, all changes made to the system since the last successful shut down are lost, for instance new programs, modified positions, or changes to system parameters.

If the system starts up with a corrupt or missing image file then the system is in system failure mode and an error message is displayed in the event log. The system must be restarted.

To restore the current system from the last successful shut down, then use **Revert to last auto saved**. An alternative is to use **Reset system** (resume the originally installed system at delivery).

Using revert to last auto saved when the controller is not in system failure mode due to a corrupt image file will be the same as a normal restart.

Restart from previously stored system data

This section describes how to restart from previously stored image file.

CAUTION

When restarting using **Revert to last auto saved**, all changes made to the system since the last successful shut down are lost and cannot be resumed.

	Action
1	On the **ABB** menu, tap **Restart**. The restart dialog is displayed.
2	Tap **Advanced...** to select restart method. The select restart method dialog is displayed.
3	Tap **Revert to last auto saved**, then tap **Next**.
4	Tap **Revert to last auto saved** to restart the controller. The controller is restarted using the image file from the most recent successful shut down.

Note

After loading a backup the program pointer will most likely not agree with the actual position of the robot.

Related information

Restart and reset system on page 124.

5.3.8 Reflashing firmware

Overview of reflashing

After replacing hardware units, such as axis computer, buses, etc., or installing newer versions of RobotWare, the system will automatically attempt reflashing the unit in order to maintain hardware/software compatibility if that is needed.

Reflashing is the process of loading appropriate firmware (hardware specific software) onto a specific unit running this software during operation.

If RobotWare is upgraded on the controller, then the FlexPendant will reflash, i.e. update to the new version, when connected.

The units currently using the reflash function are:

- Contactor interface board
- Drive units
- FlexPendant
- Profibus master
- Axis computer
- Panel board

Reflashing process

The automatic reflashing process, described below, must not be disturbed by switching off the controller while running:

	Event	Information
1	When the system is restarted, the system checks the hardware and firmwares used.	The result can be: • Hardware OK. • Hardware needs to be reflashed with new version of firmware. • Hardware cannot be used.
2	If reflashing of the firmware is required, the system restarts itself automatically while going to a specific *Update Mode*. All hardware that requires firmware update is reflashed in the same restart.	During the Update Mode, an attempt is made to download appropriate firmware to the hardware while a message is very briefly displayed on the FlexPendant.
3	Was the reflashing successful? If NO, an event log error message is logged.	A message is very briefly displayed on the FlexPendant and stored in the event log. The actual reflashing can take a few seconds or up to a few minutes, depending on the hardware to be reflashed.
4	After performing a successful reflash of all required hardware, the system performs a normal restart.	
5	Another check is made for any additional hardware/firmware mismatches.	
6	Was any additional mismatches found? If YES, the process is repeated once again. If NO, the process is complete.	If the reflashing fails twice, an error is logged.

5.4 Installed Systems

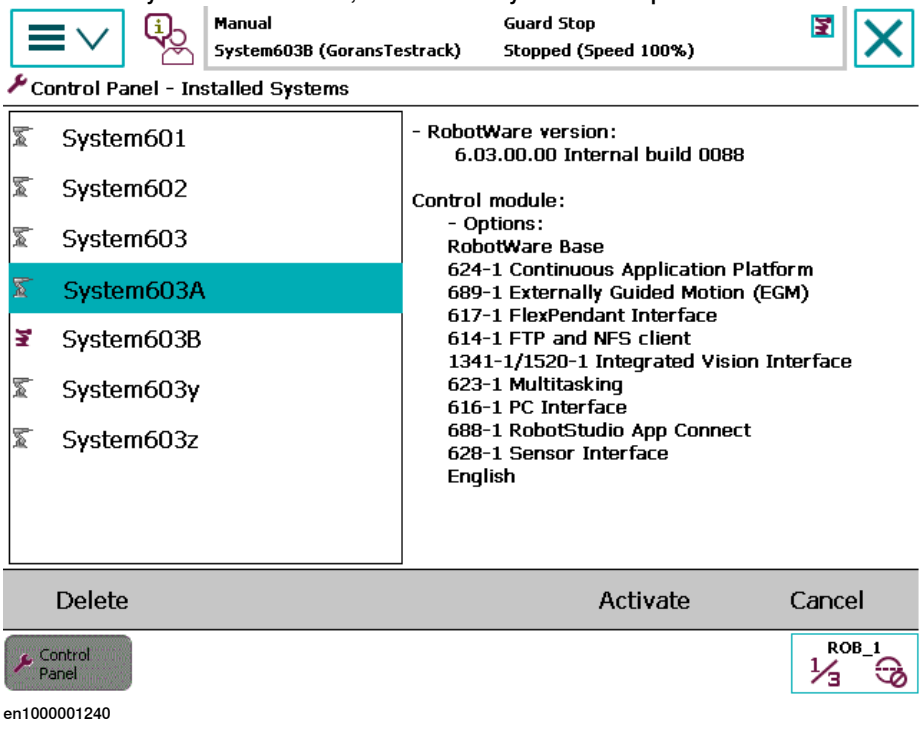
5.4.1 Managing Installed Systems

Overview

The Installed Systems is useful to switch between different systems installed. This feature in the FlexPendant allows the user to switch directly to different systems without doing a restart and starting the boot application.

Switching systems

This section describes switching directly to different systems.

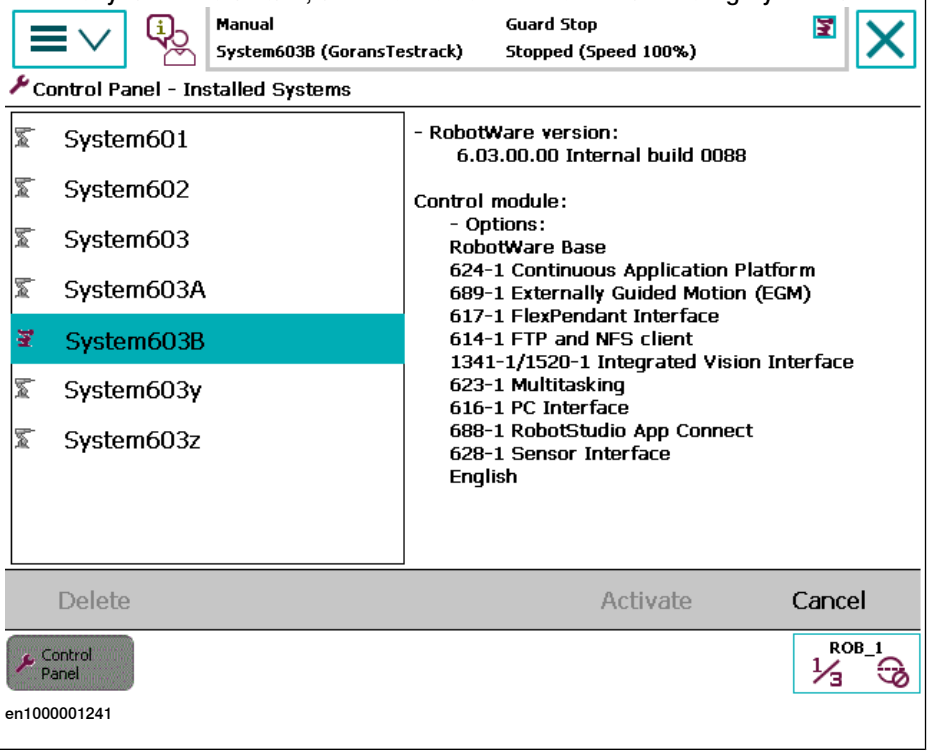
	Action
1	On the **ABB** menu, tap **Control Panel**
2	Tap **Installed systems** in **Control Panel**
3	View the list of installed systems. Click on the system that need to be activated and then tap **Activate**. To delete a system from the list, click on the system and tap **Delete**.  en1000001240

Continues on next page

5.4.1 Managing Installed Systems

Continued

	Action
4	While a system is activated, the **Activate** and **Delete** controls are greyed out.



Note

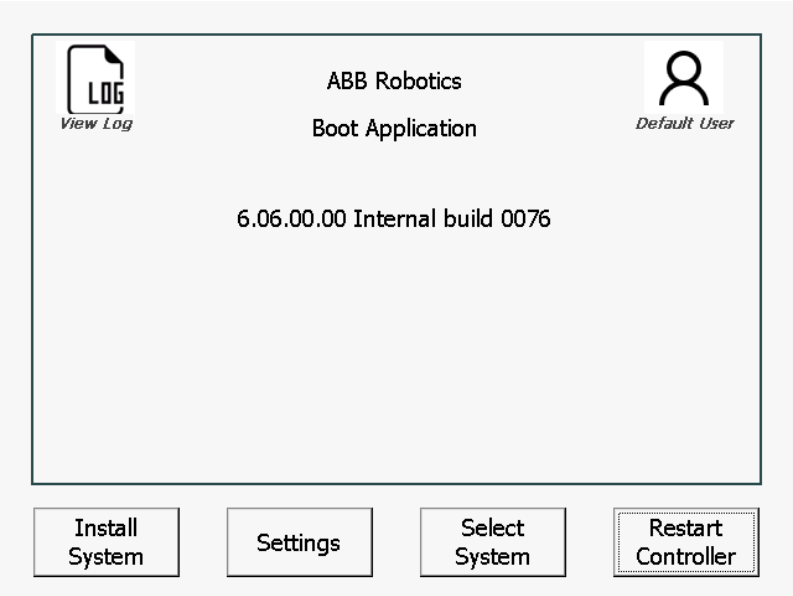
It is not possible to activate or delete an already active system.

5.4.2 RobotWare startup error

RobotWare startup error message

When the RobotWare system is unable to startup then the controller goes back to Boot Server mode, writes the error details in a log file, and display this log file on the FlexPendant. When you tap **OK**, the log file is closed and the **Boot Application** window is displayed. A link to this log file is displayed on the **Boot Application** window.

To again display the RobotWare startup error message, tap on the **View Log** icon.



xx1700001356

Note

The **View Log** icon is displayed on the **Boot Application** window only if there is an error log present.

5.5 Backup and restore systems

5.5.1 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. A default path can be set, see *Setting default paths on page 115*.

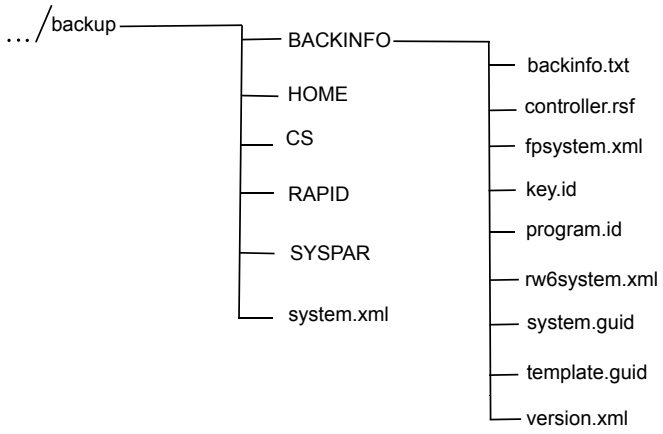
The directory is divided into the following five subdirectories:

- BACKINFO
- HOME
- CS
- RAPID
- SYSPAR

The file system.xml is also saved in the ../backup (root directory), it contains user settings.

Note

If the SafeMove option is installed, SafeMove files are also included in the system backup.



en0400000916

Continues on next page

BACKINFO

BACKINFO consists of the files *backinfo.txt*, *key.id*, *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 the options enabled in the backed up system.
- *fpsystem.xml* contains information on the settings for the FlexPendant.
- *key.id* and *program.id* files can be used to recreate a system, using RobotStudio, with the same options as the backed up system.
- *rw6system.xml* contains path information for *controller.rsf*, *system.guid*, and *key.id*.
- *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.
- *version.xml* contains detailed information about the RobotWare version.

CS

CS folder contains the following information:

- Date and time settings
- WLAN settings
- Controller name and Id
- UAS settings
- User account settings

HOME

HOME is a copy of the files in the HOME directory.

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).

What *is not* saved?

A few things are not saved on backup, but can be useful to save separately:

- The environment variable RELEASE: points out the current system pack. System modules loaded with RELEASE: as its path, are not stored in the backup.
- The current value of a PERS object in a installed module is not stored in a backup.

Continues on next page

5.5.1 What is saved on backup?
Continued

Related information

Technical reference manual - System parameters.

Operating manual - RobotStudio.

5.5.2 Backup the system

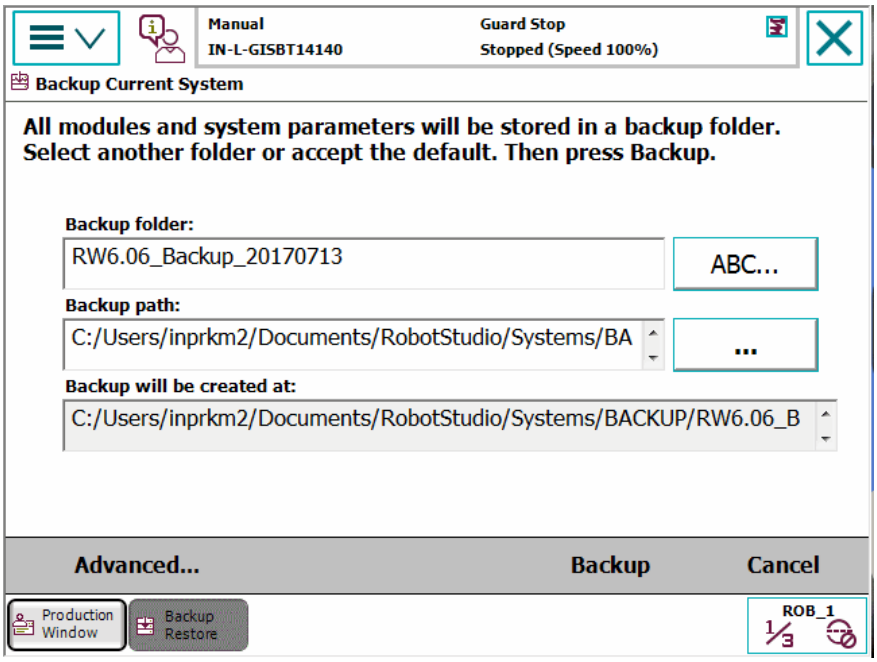
When do I need this?

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.

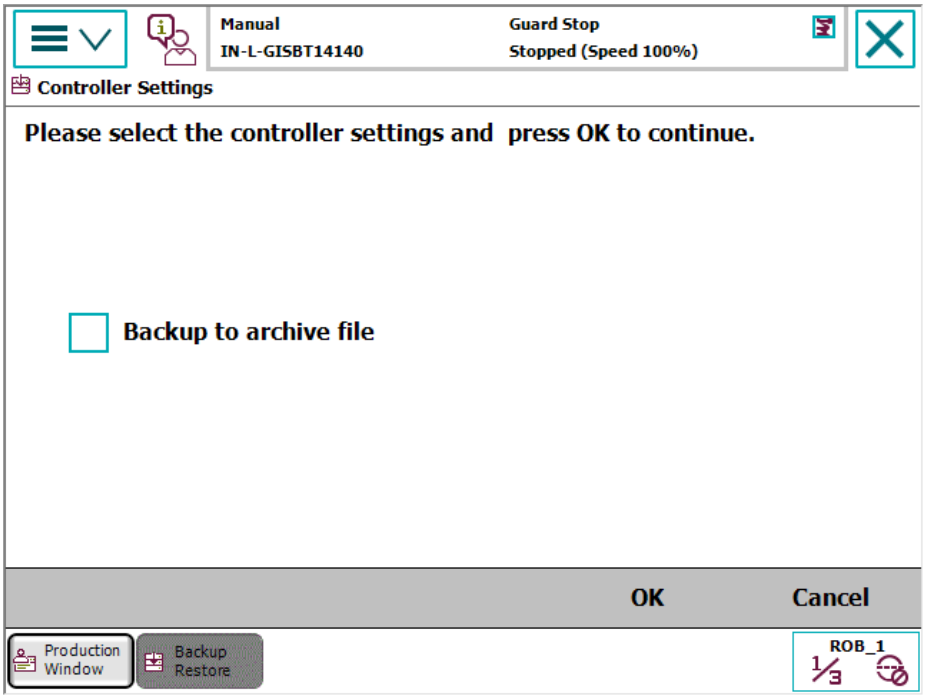
Backup the system

This section describes how to backup the system.

	Action
1	Tap the **ABB** menu and then tap **Backup and Restore**.
2	Tap **Backup Current System**. The **Backup Current System** window is displayed. If a default path has been defined as detailed in the section *Setting default paths on page 115*, this is displayed.  xx0300000441 **Note** • By default, a name for the Backup folder is created which can be renamed by the user later. • While renaming, ensure that the name does not start with a space. • If the folder name starts with a space, a warning dialog appears.

Continues on next page

5.5.2 Backup the system
Continued

	Action
3	Tap **Advanced...**. The **Controller Settings** window is displayed. **Note** Use this step to create the backup as a zip file. If this is not required navigate directly to the last step.  xx1700001302
4	Select the **Backup to archive file** checkbox.
5	Tap **OK**.
6	Is the displayed backup path the correct one? If YES: Tap **Backup** to perform the backup to the selected directory. A backup file named according to the current date is created. If NO: Tap **...** to the right of the backup path and select directory. Then tap **Backup**. A backup folder named according to the current date is created.

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*.

5.5.3 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.

Such backups are not copied to the directory HOME in following 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.

A default path can be created to any location on the network where the backup should be stored, see *Setting default paths on page 115*.

When is backup possible?

A backup of a system can be performed during program execution, with a few limitations:

- Start program, load program, load module, close program, and erase module cannot be done during backup in executing state. The RAPID instructions `Load` and `StartLoad` can, however, be used.
- 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 6.11.)

`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.

Duplicated modules?

No save operation is performed in the backup command. This implies that two revisions of the same module can exist in the backup, one from the program memory saved in Rapid\Task\Progmod\ directory and one from the HOME directory copied to the backup's home directory. Restoring such a backup will restore both revisions of the module, so the status remains unchanged.

Continues on next page

5.5.3 Important when performing backups
Continued

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.

5.5.4 Restore the system

When do I need this?

We recommend performing a restore:

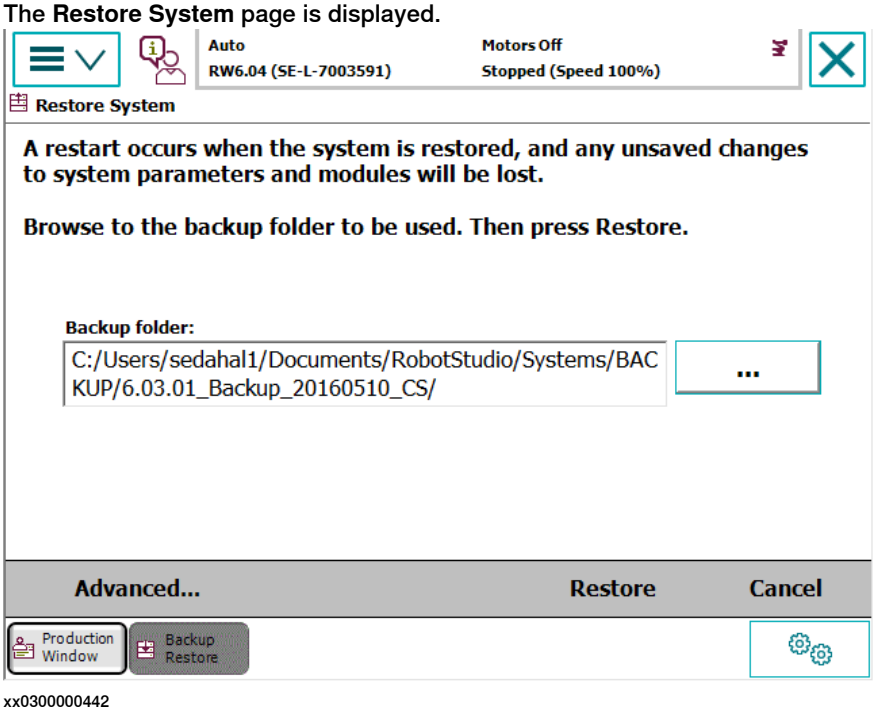
- If you suspect that the program file is corrupt.
- If any changes made to the instructions and/or parameters settings did not prove successful, and you want to return to the previous settings.

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.

Restore the system

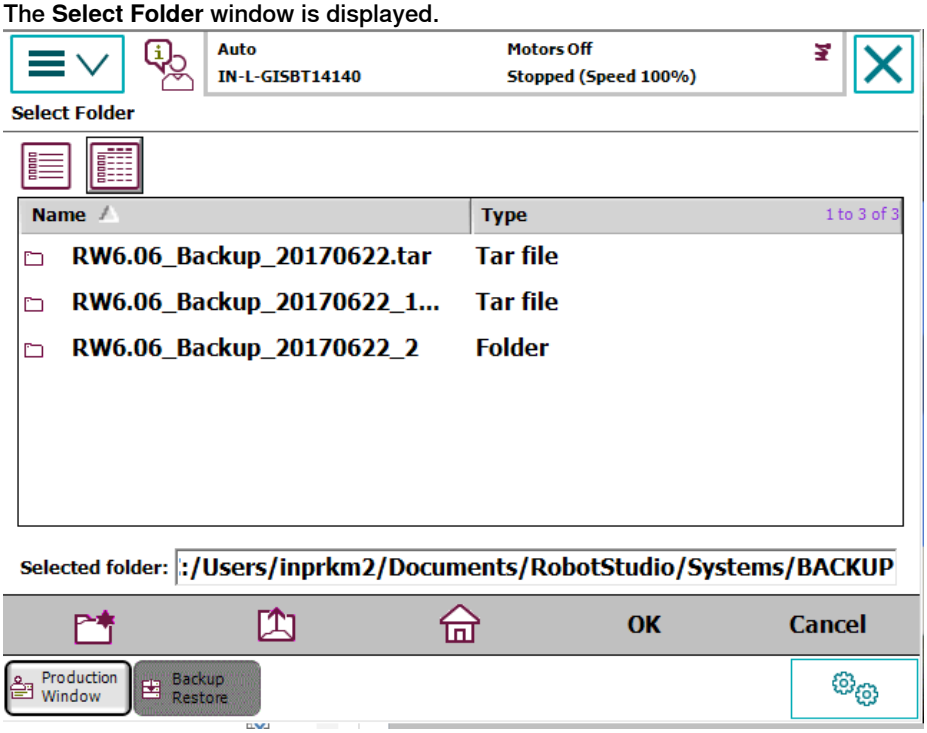
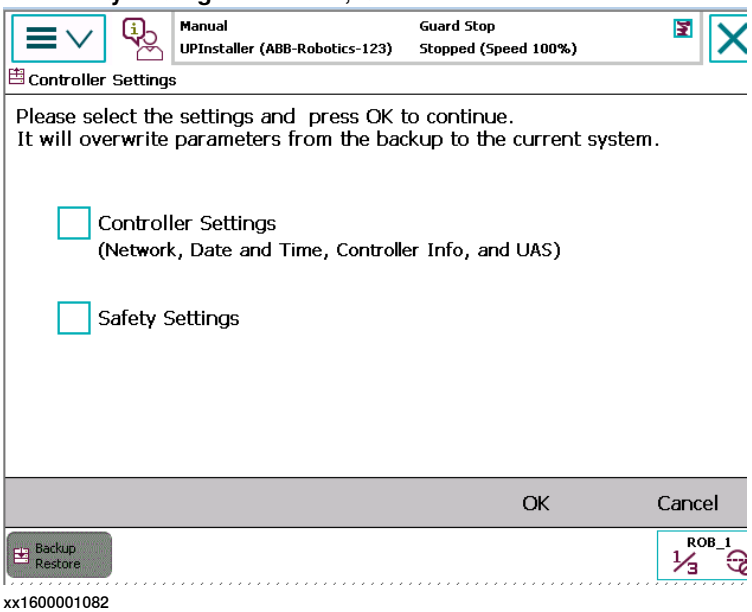
This section describes how to restore the system.

	Action
1	On the **ABB** menu, tap **Backup and Restore**.
2	Tap **Restore System**. The **Restore System** page is displayed.  xx0300000442 **Note** If a default path has been defined as explained in the section *Setting default paths on page 115*, that path is displayed in the **Backup folder** path.

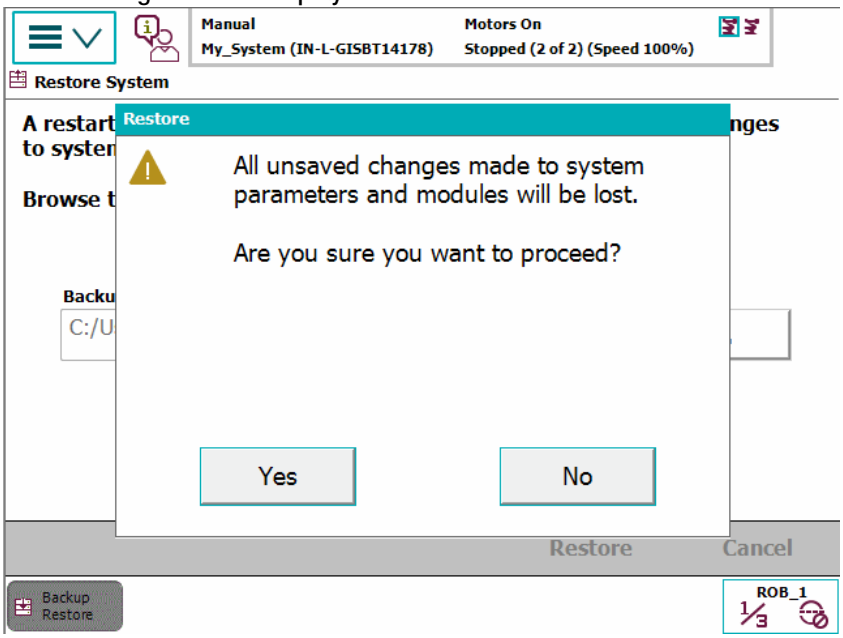
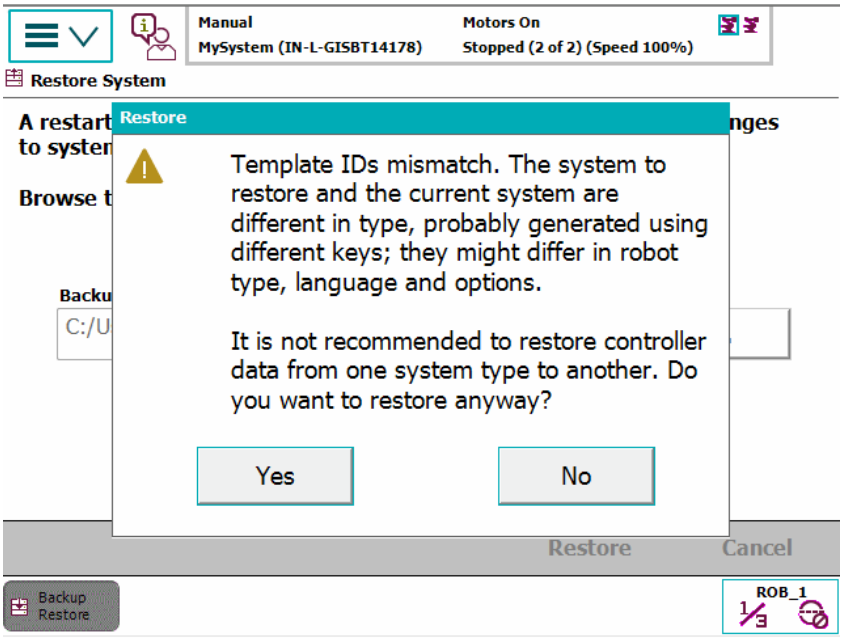
Continues on next page

5.5.4 Restore the system

Continued

	Action
3	To select the backup folder path or to select the TAR backup file format, tap .... The **Select Folder** window is displayed.  xx1700001303
4	Select the backup folder or select the backup file which is in TAR format.
5	Tap **OK**. The **Restore System** page is displayed again with the selected backup folder or file path details. If you want to replace the current controller and safety settings with the selected backup controller and safety settings, click **Advanced...**, select the **Controller Settings** and **Safety Settings** check box, and click **OK**.  xx1600001082

Continues on next page

	Action
6	Tap **Restore**. The following screen is displayed.  xx1400002325
7	Tap **Yes**. The restore is performed, and the system is restarted. **Note** If there is a mismatch between the backup and the current system, the following warning message is displayed.  xx1400002326

5.6 Diagnostic files

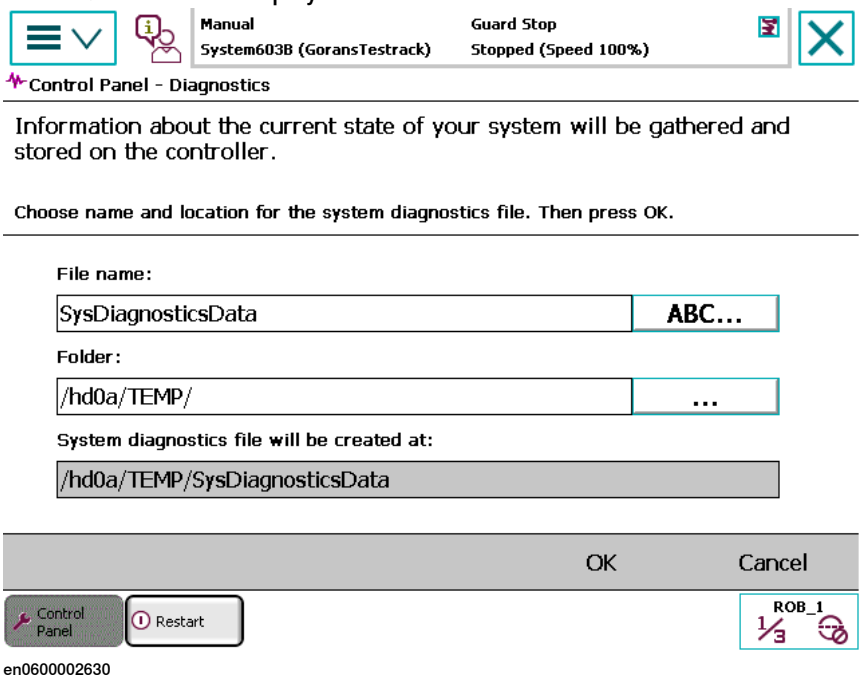
5.6.1 Creating a diagnostic file

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 *Operating manual - Troubleshooting IRC5*, section *Instructions, how to correct faults - Filling an error report*.

Create a diagnostic file

This section describes how to create a diagnostic file.

	Action
1.	On the **ABB** menu, tap **Control Panel**.
2.	Tap **Diagnostic**. A selection screen is displayed.  en0600002630
3.	Tap **ABC...** next to the File name to change the name of the diagnostic file.
4.	Tap **...** next to the Folder to change the destination for the file name.
5.	Tap **OK** to make a diagnostic file from the current system or tap **Cancel** to go back to the Control Panel.

5.7 System configuration

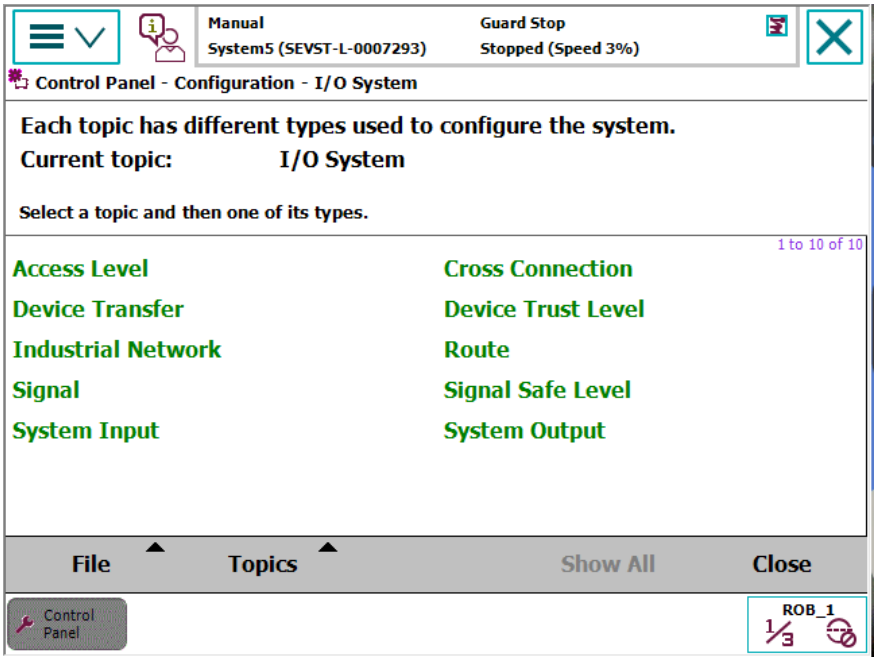
5.7.1 Configuring system parameters

About system parameters

System parameters define the system configuration. System parameters are edited using the FlexPendant or RobotStudio.

Viewing system parameters

This procedure describes how to view system parameter configurations.

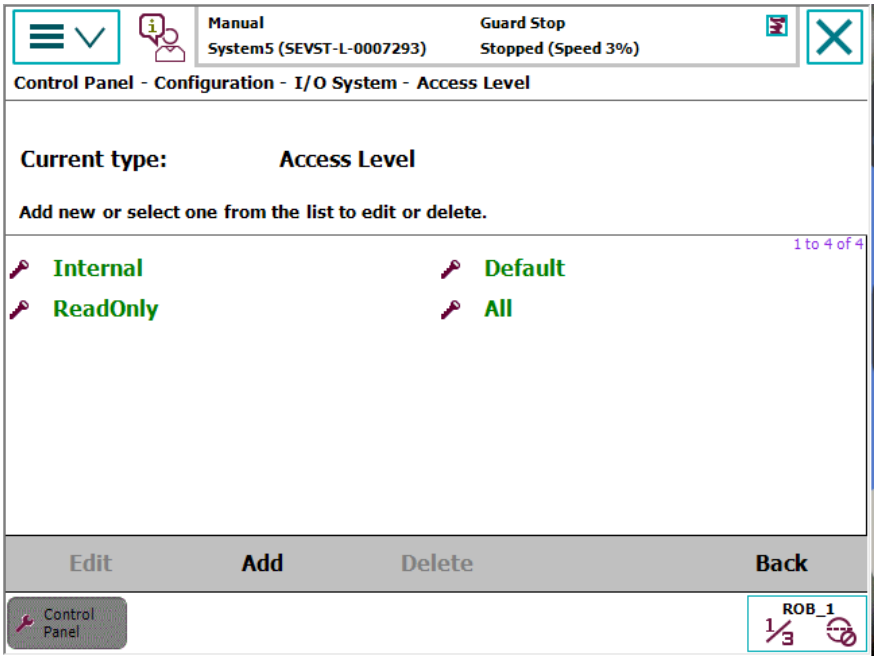
	Action
1	On the **ABB** menu, tap **Control Panel**.
2	Tap **Configuration**. A list of available types in the selected topic is displayed.  en0400001149
3	Tap **Topics** to select the topic. • Controller • Communication • I/O • Man-machine Communication • Motion
4	Tap **File** to save, load, or add new parameters from a file. Select folder and save or load. Proceed to section *Saving system parameter configurations on page 143*.
5	Tap to select a type and then tap **Show All**. To edit parameters proceed to section *Editing an instance on page 142*. To add instances proceed to section *Adding a new instance on page 142*.

Continues on next page

5.7.1 Configuring system parameters
Continued

Editing an instance

This section describes how to edit an instance of a system parameter type.

	Action
1	In the list of system parameter instances, tap to select an instance and then tap **Edit**. The selected instance is displayed.  en0400001151
2	Tap a parameter name or its value to edit the value. The way to edit values depend on the data type for the value, e.g. the soft keyboard is displayed for string or numerical values and dropdown menus are displayed for predefined values.
3	Tap **OK**.

Adding a new instance

This section describes how to add a new instance of a system parameter type.

	Action
1	In the list of system parameter instances, tap **Add**. A new instance with default values is displayed.
2	Tap the parameter name or its value to edit the value.
3	Tap **OK**.

Continues on next page

Saving system parameter configurations

This section describes how to save system parameter configurations. It is recommended to save the parameter configurations before making larger changes to the robot system. The parameters are saved automatically when performing backups.

	Action
1	In the list of types, tap the **File** menu and tap: • **Save As** to save the selected topic's parameter configurations. • **Save All As** to save all topics' parameter configurations.
2	Select directory where you want to save the parameters.
3	Tap **OK**.

Loading system parameters

This section describes how to load system parameter configuration and how to add parameters from a file.

	Action
1	In the list of types, tap the **File** menu and tap **Load Parameters**.
2	Select one of these actions, then tap **Load**: • Delete existing parameters before loading • Load parameters if no duplicates • Load parameters and replace duplicates. **Note** Configuration files and backups shall not be loaded into systems running an older RobotWare version than the one they were created in. Configuration files and backups are not guaranteed to be compatible between major releases of RobotWare and may need to be migrated after a RobotWare upgrade.
3	Select the directory and file where you want to load the parameters, then tap **OK**.

Related information

Technical reference manual - System parameters.

This page is intentionally left blank

6 RobotWare installation concept

6.1 Introduction

Overview of the installation concept

The installation of a new RobotWare system, or the update of an existing RobotWare system, can be managed in the following ways:

- Use Installation Manager to produce an installation or update package offline on a USB-stick, which later can be installed from the FlexPendant.
- Use Installation Manager to connect directly to the robot controller online over the network.

The following sections describe how to produce an installation or update package offline. For information on how to connect online over the network or using the Installation Manager in general, see *Operating manual - RobotStudio*

CAUTION

When selecting the robot in the **Installation Manager 6**, verify that the correct manipulator variant is selected.

Working with RobotWare system definitions

The repository editor supports more advanced way of working with RobotWare system definitions compared to the standard Installation Manager editor.

The repository supports hierarchies of system definitions where one system definition can inherit from another. The final system that is deployed on the controller is a result of combining the definitions from the repository.

The approach with a base system and an application system that is used in this chapter should be seen as an example, and not as a mandatory way of working. For example a subsystem of the application system can also be defined, where settings for a specific manipulator type is added.

Depending on the needs, it is possible to use completely different approaches in the way the repository is used.

Overview of the recovery function

If the SD memory card in the robot controller is damaged or in some other way made unusable, the Installation Manager disk recovery tool can be used to reformat or prepare a new ABB SD-card based on a backup of the system.

For more information, see *The recovery disk function on page 183*.

Clarifications

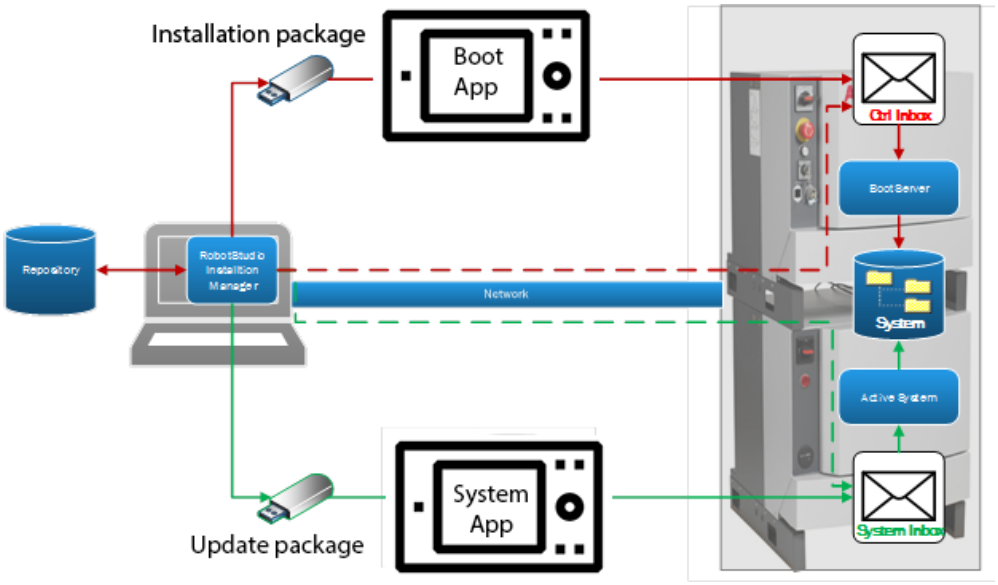
To distinguish between the software system running on the robot controller, which manages the manipulators and the whole setup of the controller and its mechanical units, the following definitions are used:

- RobotWare *system* - the software system running on the controller.
- Robot *system* - the controller and its mechanical units.

Continues on next page

Deployment packages and the repository

The repository is used by Installation Manager and is the storage where all files needed to create and modify RobotWare systems are placed. There are currently two types of deployment packages, the *Installation Package* and the *Update Package*.



xx1600001362

The installation package

The installation package is a set of files with a predefined format and directory structure that can be used to create RobotWare systems on one or more controllers.

The installation package created by the Installation Manager shall be stored on a USB-stick that can be inserted into the FlexPendant or the controller USB connector. The Boot Application on the FlexPendant copies the installation files for the selected system over to the controller inbox.

The Boot Server application on the robot controller uses the files in the controller inbox to create a RobotWare system on the memory card in the robot controller. When the installation is finished the installed RobotWare system is started.

The update package

The update package is a set of files with a predefined format and directory structure that can be used to update currently active RobotWare systems on one or more controllers.

The update package created by Installation Manager is stored on an USB-stick and inserted into the FlexPendant or the controller USB connector. The FlexPendant copies the selected update files over to the inbox of the currently active system.

The currently active system on the robot controller uses the files in its inbox to update itself. When the update is finished the updated active RobotWare system is started.

Prior to the update, a backup is taken of the active system and normally restored after the update.

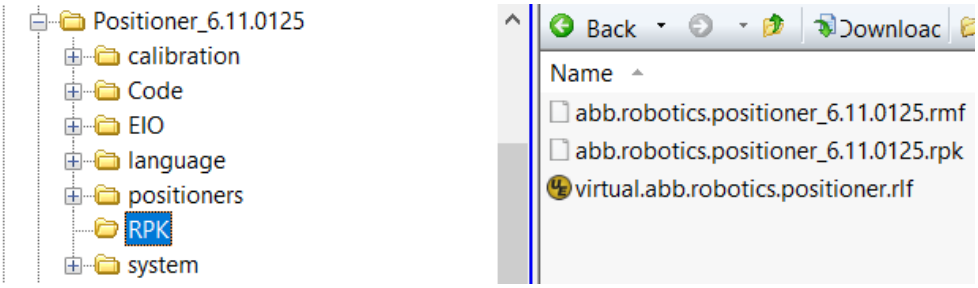
Continues on next page

Installation media

When creating RobotWare systems, access to the original product installation media (rmf, rpk files) is always needed. The media files may, for example, be useful when creating a virtual controller from a real controller ("go-offline"), or when re-creating a system installation.

All necessary files from the product media package for Add-In products are preserved on the real controller during system (re)installation and update. The files are stored in the Add-In product subdirectory named **RPK**, for example:

/hd0a/MySystem/PRODUCTS/Positioner_6.11.0125/RPK



xx2000000680

6.2 Working with the repository

6.2.1 Recommended working procedure

Basic steps

The following description of a workflow, from an empty repository to a complete repository for a plant in operation, is based on the assumption that the task is to commission and maintain a large number of robot systems. However, the recommendation is also applicable on smaller installations.

1. Setup the repository. Select a folder in the file system for the root of the repository.

   Installation Manager creates the following standard folders in this folder: *Controllers*, *Systems*, *Products*, *Licenses*, *Backups*, and *Additional Files*. Installation Manager also creates the *repository_manifest.xml* file which defines the repository.
2. Import products, licenses, backups, and additional files that are intended to be used when defining systems and controllers in the repository.

   Note

   Following are the details for selecting the folder or file:
   - **Products**: Navigate to the RobotWare folder location and select the `RobotWare_RPK_xxx` folder.
   - **License**: Navigate to the license file location and select the required license file.
   - **Backup**: Navigate to the backup folder location and select the backup folder.
   - **Additional files**: Navigate to the location and select the folder or file according to the requirement.
3. Identify the robot application types in the plant and define an application type system for each of the application types. These are the base systems in the repository.
4. Identify which manipulator types shall be used with which base system and then create an application system for each combination. These are the application systems in the repository.
5. Identify the robot systems (controllers) and create their definitions, for example, name and IP-addresses. Refer to the application system it should run.

Note

Step 4 can be started at the same time as step 1 and continue in parallel.

Continues on next page

Note

It is possible to structure products, backups, additional files as well as systems and controllers by using an arbitrary directory hierarchy inside of the each of the top level repository folders. This helps when managing a large number of items.

Note

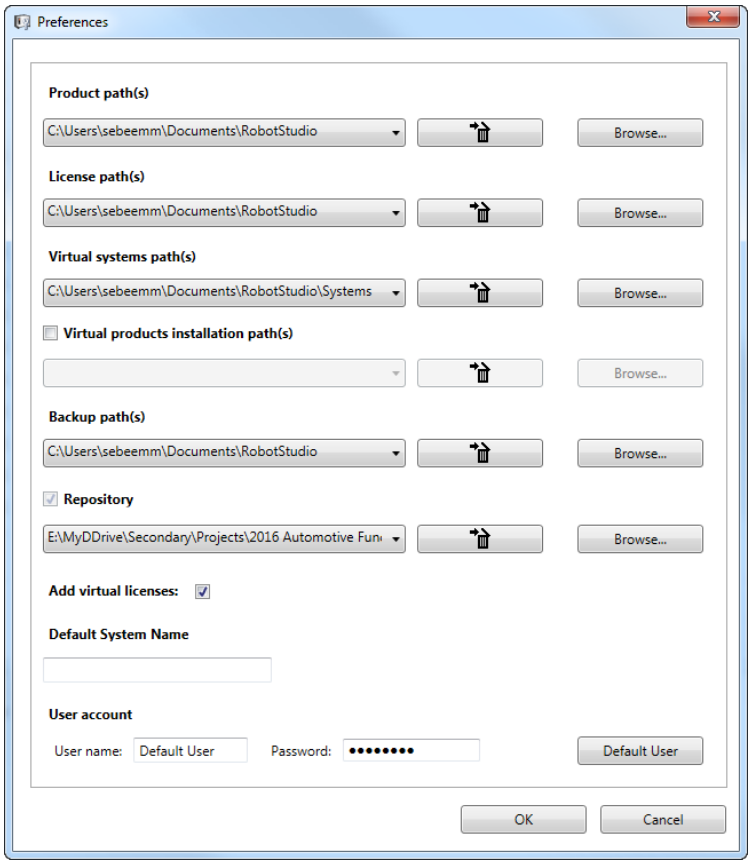
It is possible to create both installation packages as well as update packages to use for the installation. For more details, see *Creating an installation package on page 159* and *Creating an update package on page 162*.

6.2.2 Setting up the repository

Enable the repository functionality

To use the repository, it is first necessary to enable the repository functionality in Installation Manager.

Open the Installation Manager start page and click **Preferences**. The following dialog is displayed:



xx1600001443

Select the repository and browse to select the repository root folder.

Preparing the repository folder

Before the repository functionality can be used, the repository folder needs to be prepared.

The repository folder must have a *repository_manifest.xml* file, see *The repository_manifest.xml file on page 165*. The file is created automatically if it does not exist.

All category folders are created when a new folder is selected as repository folder, that is, *Controllers*, *Systems*, *Products*, *Licenses*, *Backups*, and *Additional Files*. A *repository_manifest.xml* file is also created to identify the repository.

Select the **Repository** tab in Installation Manager and start populating the categories **Products**, **Licenses**, **Backups**, and **Additional Files** using the **Add** button on the corresponding tab.

Continues on next page

Adding and removing products, licenses, and backups

Select the appropriate tab and use the **Add** or **Remove** buttons to add or remove products, licenses, and backups.

Note

Following are the details for selecting the folder or file:

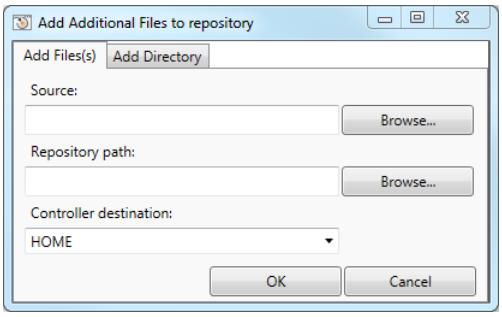
- **Products**: Navigate to the RobotWare folder location and select the `RobotWare_RPK_xxx` folder.
- **License**: Navigate to the license file location and select the required license file.
- **Backup**: Navigate to the backup folder location and select the backup folder.
- **Additional files**: Navigate to the location and select the folder or file according to the requirement.

When adding a product, license, or backup it will be copied into the selected corresponding folder, or subfolder, in the repository. Note that subfolders must be created manually when building a hierarchy.

When removing a product, license, or backup it will be deleted from the repository.

Adding and removing additional files

When adding an additional file to the repository it is also necessary to define where on the controller it shall be copied at installation time.



xx1600001444

First select the file to import in the repository using the **Browse** button for **Source**.

Then define where in the additional files folder the file shall be stored, i.e. in which folder under the additional file root folder using the browse button for **Repository path**. Default, when no destination is selected, is to store it directly under the additional files root folder.

The last step is to define where in the controller the file shall be stored upon installation. The controller destination uses the *HOME* directory as root and can

Continues on next page

6.2.2 Setting up the repository
Continued

only put additional files there. For example, *HOME/temp* puts the file in the temp folder.

UAS and safety files

Additional files can also contain folders with UAS files and safety controller configuration files. They are added by using the **Add Directory** tab of the dialog. Only a repository location can be specified for these files since the controller side destination is predefined and cannot be changed.

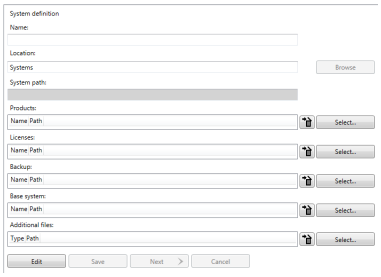
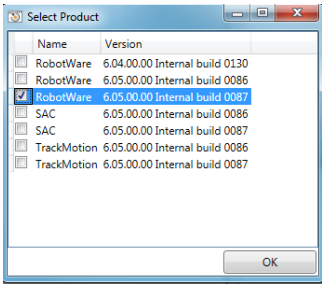
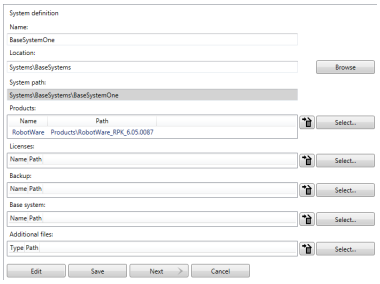
UAS files must be saved in a separate folder that is only allowed to contain UAS files. The same limitation is valid for safety files.

A safety files folder can be added as additional files to one or more systems or controllers. A UAS files folder can be added as additional files to controllers only.

6.2.3 Creating a base system

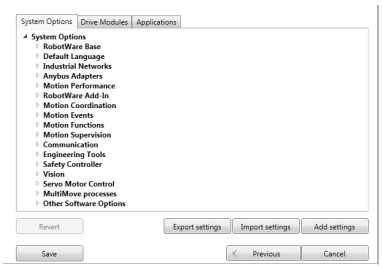
Description

The following procedure and example describes how to create a base system.

	Action	Note
1	Select the **Systems** tab in Installation Manager and click the **New** button to create a new system. The following right panel view is displayed.	 xx1600001445
2	Enter a name for the system in the **Name** text box.	The name *BaseSystemOne* is used in this example.
3	Click the **Browse** button for **Location** to select where to store the system. The system can either be placed directly under the *Systems* root folder or in any of its sub folders. Create and delete sub folders using the ordinary explorer functionality. **System path** displays the location of the system when its saved.	In this example the sub folder *Base Systems* has manually been created and selected via the **Browse** button. **System path** displays the location *Systems\Base Systems\BaseSystemOne*.
4	Click the **Select** button for **Products** to select which products to include in the system. A dialog shows the products that are available in the repository.	 xx1600001446 Only the RobotWare product is selected in this example.
5	Select **Licenses**, **Backups**, **Base system**, and **Additional files** in the same way as the products.	Base system is explained further in section *Description on page 155*, where this system is used as a base system.
6	Click the **Next** button.	 xx1600001447 No more selections are made for this example.

Continues on next page

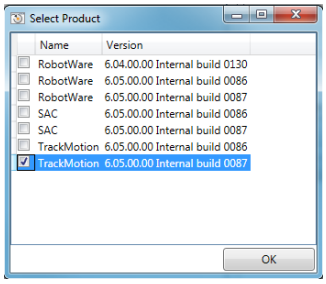
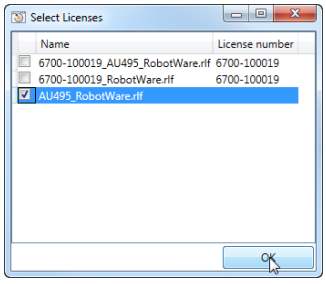
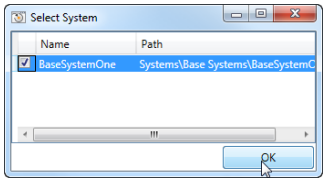
6.2.3 Creating a base system
Continued

	Action	Note
7	Select the options, robots, and applications to be included in the system. **Note** At this stage it is possible to add options without having a valid licence file specified. A valid license file is required when deploying the RobotWare system to a robot controller. All selections will be validated at installation time.	 xx1600001448
8	Click the **Save** button.	

6.2.4 Creating an application system

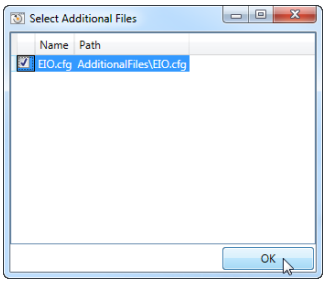
Description

The following procedure and example describes how to create an application system based on a base system.

	Action	Note
1	Select the **Systems** tab in Installation Manager and click the **New** button to create a new system.	
2	Enter a name for the system in the **Name** text box.	The name *AppSystemOne* is used in this example.
3	Click the **Browse** button for **Location** to select where to store the system. The system can either be placed directly under the *Systems* root folder or in any of its sub folders. Create and delete sub folders using the ordinary explorer functionality. **System path** displays the location of the system when its saved.	In this example the system is created directly under the *Systems* root folder. **System path** displays the location *Systems*.
4	Click the **Select** button for **Products** to select which products to include in the system. A dialog shows the products that are available in the repository.	 xx1600001449
5	Click the **Select** button for **Licenses**.	 xx1600001450
6	Click the **Select** button for **Backups** to include a backup.[i]	
7	Click the **Select** button for **Base system** to select which base system to include.	 xx1600001451 In this example the base system from the previous example is used, see*Description on page 153*.

Continues on next page

6.2.4 Creating an application system
Continued

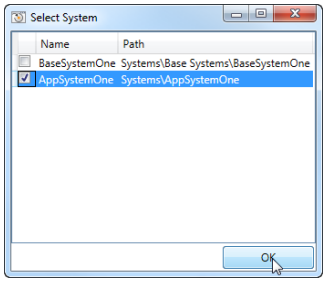
	Action	Note
8	Click the **Select** button for **Additional files** to include any additional files.	 xx1600001452 In this example a EIO.cfg-file is included.
9	Click the **Next** button and select the options, robots, and applications to be included in the system. **Note** Only options selected in the current system can be changed. Options selected in the referred base system are visible but can only be changed or deselected from the base system. Note that such a change will affect all systems using the base system.	
10	Click the **Save** button.	

i In the current RobotWare version there is no support for automatic restore of backup.

6.2.5 Defining controllers

Description

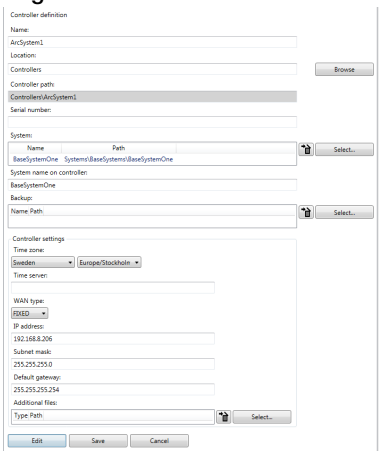
The following procedure and example describes how to apply an application system to a controller.

	Action	Note
1	Select the **Controllers** tab in Installation Manager and click the **New** button to create a new controller.	
2	Enter a name for the controller in the **Name** text box.	In this example the name *ArcSystem1* is used.
3	Click the **Browse** button for **Location** to select where to store the controller. The controller can either be placed directly under the *Controllers* root folder or in any of its sub folders. Create and delete sub folders using the ordinary explorer functionality.	In this example the sub folder *ArcLine* is created under the *Controllers* root folder.
4	Click the **Select** button for **System** to select which system to include.	 xx1600001453 In this example the application system *AppSystemOne* from the previous example is used, see*Description on page 155*.
5	Click the **Select** button for **Backups** to include a backup.	
6	Click the **Select** button for **Additional files** to include any additional file.	
7	Enter a serial number for the controller in the **Serial number** text box.	
8	Select the time zone settings or define a time server.	In this example the time zone *Sweden* and *Europe/Stockholm* is used.

Continues on next page

6.2.5 Defining controllers

Continued

	Action	Note
9	Define the communication settings.	In this example the following settings are used:  xx1600001454
10	Click the **Add** button to save and add the controller to the controllers collection.	

6.2.6 Creating an installation package

Overview

There are two variants of installation packages:

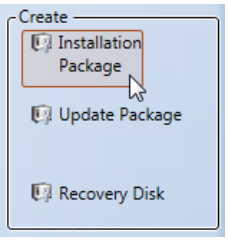
- The installation package defines the *controllers* on which the installation shall take place and systems to be installed.
- The installation package defines only the *systems*. The users select which system shall be installed at installation time.

Note

To be able to create an installation package with Installation Manager it is required that the RobotWare version that was released together with the current RobotStudio version is present. This is required regardless of which RobotWare version that is used by the deployed RobotWare systems.

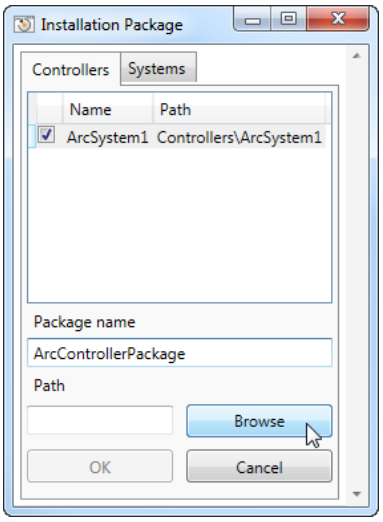
Controller installation package

The following procedure and example describes how to create a controller installation package.

	Action	Note
1	Select the **Create** panel in Installation Manager and click the **Installation Package** button to create a new installation package.	 xx1600001455
2	Select the **Controllers** tab.	
3	Select the controller in the **Select controllers** list.	In this example the controller *ArcSystem1* from the previous example is used, see*Description on page 157*.
4	Enter a name for the installation package in the **Package name** text box.	In this example the name *ArcControllerPackage* is used.

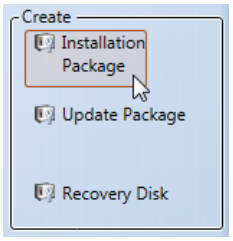
Continues on next page

6.2.6 Creating an installation package
Continued

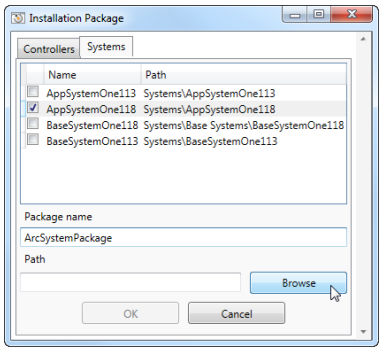
	Action	Note
5	Click the **Browse** button for **Path** to select where to store the controller package, for example on the USB-stick that is used to install the system on the controller.	 xx1600001456
6	Click the **OK** button to save and store the installation package.	The **OK** button is enabled when both **Package name** and **Path** have valid values.
7	The installation package is ready to be installed on the robot controller.	See *Deploying installation and update packages on page 170*.

System installation package

The following procedure and example describes how to create a system installation package.

	Action	Note
1	Select the **Create** panel in Installation Manager and click the **Installation Package** button to create a new installation package.	 xx1600001455
2	Select the **Systems** tab.	
3	Select the system in the **Select system** list.	In this example the application system *AppSystemOne* from the previous example is used, see*Description on page 155*.
4	Enter a name for the installation package in the **Package name** text box.	In this example the name *ArcSystemPackage* is used.

Continues on next page

	Action	Note
5	Click the **Browse** button for **Path** to select where to store the installation package, for example on the USB-stick that is used to install the system on the controller.	 xx1600001457
6	Click the **OK** button to save and store the installation package.	The **OK** button is enabled when both **Package name** and **Path** have valid values.
7	The installation package is ready to be installed on the robot controller.	See *Deploying installation and update packages on page 170*.

Create system using installation package in boot server mode

Use the following procedure to create a system using installation package in boot server mode:

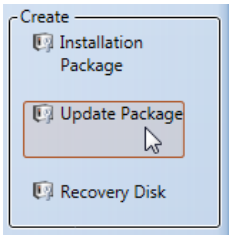
Step	Action
1	On FlexPendant tap **Restart** > **Advanced settings**. The **Advanced restart** window is displayed.
2	Tap **Start boot application**.
3	Tap **Next**. The **Restart** window is displayed.
4	Tap **Start Boot Application**.
5	Tap **Install System**. The **Install System** window is displayed.
6	Tap **Continue**. The field for selecting the installation package is displayed.
7	Browse and select the installation package from USB stick. **Note** If USB stick is not inserted, an error message is displayed.
8	Tap **OK**.
9	Tap **Continue**. The installation operation validates the installation package, copies the system, and displays the **Install System - Copy Complete** window.
10	Tap **Continue**. The system is created and the controller is restarted.

6.2.7 Creating an update package

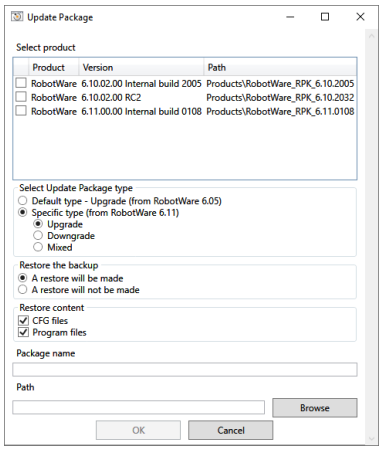
Update package creation

The following procedure and example describes how to create an update package.

Update packages are used to update currently active RobotWare systems on one or more controllers. Depending on the selected update package type, the package can either be used for upgrades, downgrades or a mix of the two.

	Action	Note
1	Select the **Create** panel in Installation Manager and click the **Update Package** button to create a new update package.	 xx1600001458
2	Select one or more products in the **Select product** list.	
3	In **Select Update Package type**, select if the package type should be *default* or *specific*. For specific package types, select if an upgrade or downgrade is to be performed, or a mix: • **Upgrade** - Replace products in a controller only when a newer product version is present in the package. • **Downgrade** – Replace products in a controller only when an older product version is present in the package. • **Mixed** – The products in the package will always replace the products in a controller if the versions differ. **Note** This only applies to products in the package that are currently used in the controller. Other products are ignored.	The **Default type** is available for RobotWare 6.05 and later. The **Specific type** is available for RobotWare 6.11 and later.
4	In **Restore the backup**, select if the backup should be restored at update.	The backup is automatically created from the FlexPendant during the update sequence and is located on the controller in the folder *hd0a/Backup*, see *Description on page 180*.
5	In **Restore content**, select which content of the backup that shall be restored.	
6	In field **Package name**, enter a name for the update package.	

Continues on next page

	Action	Note
7	Click **Browse** to select the **Path** indicating where the update package should be stored, for example on the USB-stick that is used to update the system on the controller.	 xx2000000675
8	Click **OK** to save and store the update package.	The **OK** button is enabled when the **Path** has been selected. The **OK** button is enabled when the product, or products, have been selected and the **Package name** and **Path** have valid values.
9	The update package is ready to be installed on the robot controller.	See *Description on page 180*.

6.2.8 Repository folders and file structure

File structure

The repository is a file structure with a root folder, the *<Repository name>* folder. The repository is defined by a *repository_manifest.xml* file.

The repository stores products, licenses, system definitions, controller definitions, backups, and additional files. Each category of files has it's own sub-folder under the root folder.

The repository root folder has six predefined category root folders:

Folder	Description
Products	Products can be organized in a hierarchical folder structure. This makes it possible to store several versions of products. Each product is represented by a folder. The recommended name of the folder is *<Product name>_<Product version>*. However the product is identified through the product manifest file (*.rmf*) in the folder. The same folder also contains all package files (*.rpk*) of the product **Note** It is not allowed to mix files from several products in the same folder.
Licenses	Each license is a file (*.rlf*).
Systems	Systems can be organized in a hierarchical folder structure. Each system is represented by a folder with its name. The system is defined by the *systemtemplate_manifest.xml* file in the folder.
Controllers	Controllers can be organized in a hierarchical folder structure. Each controller is represented by a folder with its name. The controller is defined by the *controller_manifest.xml* file in the folder.
Backups	Backups can be organized in a hierarchical folder structure. Each backup is represented by a folder with its name. The *BACKINFO* folder in the backup folder defines the backup. **Tip** The default name is *<System name>_BACKUP_<Date>*, but a better name may be *<Controller name>_<System name>_BACKUP_<Date>*, since the focus should be on the controller name. The system name may be identical on several controllers.
AdditionalFiles	Additional files can be organized in a hierarchical folder structure. Additional files can be selected to be included in a system or controller.

Continues on next page

The repository_manifest.xml file

The *repository_manifest.xml* file identifies the root folder of the repository. It can also hold information to redirect to category root folders outside the repository folder structure.

If the *repository_manifest.xml* file does not exist, then it is created by default with the following content:

```
<?xml version="1.0" encoding="utf-8"?>
<RepositoryManifest version="1.0">
  <Repository name="RSRepository">
    <Controllers path="Controllers" />
    <Products path="Products" />
    <Systems path="Systems" />
    <Backups path="Backups" />
    <AdditionalFiles path="AdditionalFiles" />
    <Licenses path="Licenses" />
  </Repository>
</RepositoryManifest>
```

XML-tag	Description
`RepositoryManifest`	The tag `RepositoryManifest` has the attribute `version` which defines the format of the manifest.
`Repository`	The tag `Repository` has the attribute `name` which defines the name of the repository. The folder of the repository shall have this name.
`Controllers`	The tag `Controllers` defines the controllers folder with the attribute `path`.
`Products`	The tag `Products` defines the products folder with the attribute `path`.
`Systems`	The tag `Systems` defines the systems folder with the attribute `path`.
`Backups`	The tag `Backups` defines the backups folder with the attribute `path`.
`Additional files`	The tag `Additional files` defines the additional files folder with the attribute `path`.
`Licenses`	The tag `Licenses` defines the backup licenses with the attribute `path`.

A path can be absolute or relative. An absolute path starts with a drive name such as C:\. A relative path is a path without a drive name or a slash as the first character in the path string.

The repository path, which is set in the installation manager preferences, is used as root path of all relative paths. It is recommended to use only relative paths if you intend to copy or distribute complete repositories.

Continues on next page

6.2.8 Repository folders and file structure
Continued

The systemtemplate_manifest.xml file

The `systemtemplate_manifest.xml` defines a system in the repository. It is possible to refer to a base system which makes it possible to simplify the build-up of a large number of systems.

Example of a *systemtemplate_manifest.xml* file of a base system:

```
<?xml version="1.0" encoding="utf-8"?>
<SystemTemplateManifest version="1.0">
  <System name="MyBaseSystem"
       path="Systems\BaseSystems\MyBaseSystem"/>
  <Products>
    <Product name="RobotWare" version="6.05.0000"
         path="Products\RobotWare_RPK_6.05.0000"/>
    <Product name="MyProduct" version="0.01.0135.00"
         path="Products\MyProduct_0.01.0135.00"/>
  </Products>
</SystemTemplateManifest>
```

Example of a *systemtemplate_manifest.xml* file of a system using the above base system:

```
<?xml version="1.0" encoding="utf-8"?>
<SystemTemplateManifest version="1.0">
  <System name="StudWelding1" path="Systems\StudWelding1"/>
  <BaseSystems>
    <BaseSystem name="MyBaseSystem"
         path="Systems\BaseSystems\MyBaseSystem"/>
  </BaseSystems>
  <Products>
    <Product name="POSITIONER" version="6.05.0000"
         path="Products\Positioner_RPK_6.05.0000"/>
  </Products>
  <Licenses>
    <File path="Licenses\MyLicense_RobotWare.rlf" />
  </Licenses>
  <Backup path="Backups\MyBackup" />
</SystemTemplateManifest>
```

XML-tag	Description
`SystemTemplateManifest`	The tag `SystemTemplateManifest` has the attribute `version` which defines the format of the manifest.
`System`	The tag `System` has the attribute `name` which defines the name of the system. The system folder shall have this name.
`BaseSystems`	The tag `BaseSystems` defines the base systems referred to. Currently the limitation is that only one base system can be referred to.
`BaseSystem`	The tag `BaseSystem` refers to another system definition and has the attribute `name` which defines the name of the base system, and the attribute `path` which defines the path to the base system. This feature can be used to create a hierarchy/inheritance of system definitions. Note that definitions from the base system are inherited/propagated to all subsystem definitions. Changing properties of a base system affects all other system definitions that directly or indirectly inherit from it.

Continues on next page

XML-tag	Description
`Products`	The tag `Products` defines the products in the system.
`Product`	The tag `Product` has the attribute `name` which defines the name of the product and the attribute `version` which defines the version of the product. The attribute `path` defines to the path to the product folder
`Licenses`	The tag `Licenses` defines the licenses in the system
`Backup`	The tag `Backup` defines the backup folder with the attribute `path`.
`AdditionalFiles`	The tag `AdditionalFiles` defines the additional files in the system.
`File`	The tag `File` has attribute `path` which defines the path for the file. The attribute `controllerPath` defines the folder on the controller where the file shall be copied at installation. **Note** In the current release this path must start with "HOME", i.e. all additional files must be placed under the system *HOME* folder structure.
`Dir`	The tag `Dir` has attribute `path` which defines the repository path for the directory. It also has attribute `type`, which defines the type of information contained in the directory.

Continues on next page

The controller_manifest.xml file

The *controller_manifest.xml* contains definition of a controller in the repository. A controller definition points out the system the controller shall run and also its specific controller settings. As an option it is also possible to define a backup which can be copied to the controller during the installation.

Example of a *controller_manifest.xml* with the following content:

```
<?xml version="1.0" encoding="utf-8"?>
<ControllerManifest version="1.0">
  <Controller name="MyController" serialNumber="6700-123456">
    <System name="StudWelding1" path="Systems\StudWelding1">
    <AdditionalFiles>
      <Dir path="AdditionalFiles\safety" type="SAFETY"/>
    </AdditionalFiles>
    <ControllerSettings>
      <DateTime timeServer="http://www.pool.ntp.org/zone/se "
           timeZone="Europe/Stockholm"/>
      <WAN type="FIXED" ip="192.168.9.36" subnetMask="255.255.255.0"
           defaultGateway="192.168.9.254"/>
      <UASSettings path="AdditionalFiles\UAS"/>
    </ControllerSettings>
    <Backup path="Backups\MyController_BACKUP"/>
  </Controller>
</ControllerManifest>
```

XML-tag	Description
`ControllerManifest`	The tag `ControllerManifest` has the attribute `version` which defines the format of the manifest.
`Controller`	The tag `Controller` has the attribute `name` which defines the name of the controller and the attribute `serialNumber` (optional) which identifies the serial number of the controller.
`System`	The tag `System` has the attribute `name` which defines the name of the system on the robot controller, and the attribute `path` which defines the path to the system relative to the repository root folder. **Note** It is recommended to keep the default, which is to use the same name of the system on the robot controller as in the repository.
`ControllerSettings`	The tag `ControllerSettings` defines the settings of the controller.
`DateTime`	The tag `DateTime` has the attribute `timeZone` which defines the time zone to use and the attribute `timeServer` which defines the URL to the time server to use for synchronization of the controller time. **Note** Note that the `timeZone` value must match the text in column TZ in file *zone.tab* under *RobotWareXXX/system/timezone*.

Continues on next page

XML-tag	Description
`WAN`	The tag `WAN` defines the network settings. It has the attribute `type`, with the values `"FIXED"`, `"DHCP"`, and `"None"`. The attributes `ip`, `defaultGateway`, and `subnetMask` define the connection. **Note** If the value `"FIXED"` is used then the attributes `ip` and `subnetMask` must not be empty. Value `"NONE"` means that no IP address is set.
`UASSettings`	The tag `UASSettings` has the attribute `path` which defines the catalog of the UAS files.
`Backup`	The tag `Backup` defines the backup to be copied to the */hd0a/BACKUP* folder after installation is ready.[i]
`AdditionalFiles`	The tag `Additional files` has the attribute `path` which defines the additional files folder, and the attribute `type` with the value `"SAFETY"`.

[i] In the current release there is no support for automatic restore of backup at the installation time.

6.3 Deploying installation and update packages

6.3.1 Overview

Introduction

The following two types of deployment packages can be generated from Installation Manager:

- Installation package - Used to install a new RobotWare system.
- Update package - Used to update an existing RobotWare system.

The deployment process copies the necessary files to the following locations on the controller SD-card.

- In the case of installation the files are copied to the controller inbox (*hd0a/Inbox/*).
- In the case of update the files are copied to the inbox owned by the currently active system.

6.3.2 Installing a RobotWare system using Boot Application

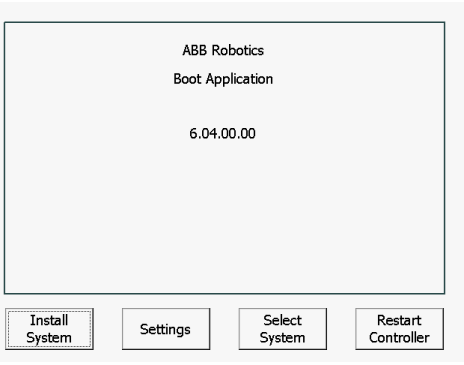
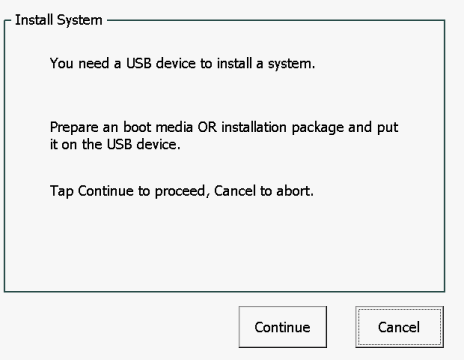
Procedure

Note

The user needs the following UAS grants to perform an installation:

- Administration of installed systems.

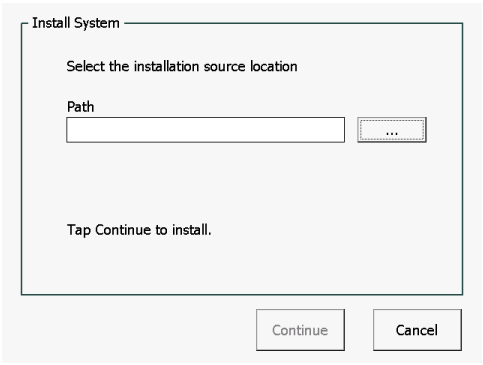
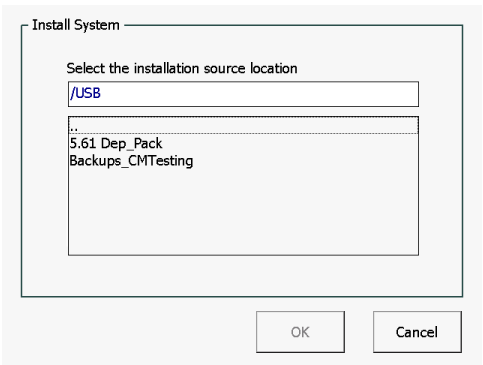
The following procedure provides the steps involved during the installation of the RobotWare system.

	Action	Illustration/Note
1	Start the controller in Boot Server mode.	 xx1600001361 **Note** If the installation package is stored elsewhere, make sure to copy the installation package to a USB-stick before starting the installation procedure.
2	Tap **Install System**. The **Install System** window is displayed.	
3	Insert the USB-stick with the installation package into the USB-port of the controller or FlexPendant and tap **Continue**.	 xx1700001434 **Note** At least 1 GB free memory on the Controller hard disk is required to install a system. If this is not available the installation cannot be continued and a message is displayed.

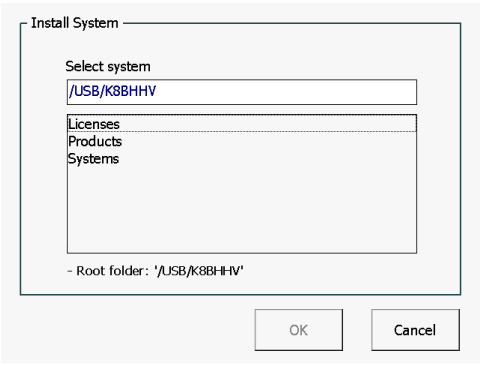
Continues on next page

6.3.2 Installing a RobotWare system using Boot Application

Continued

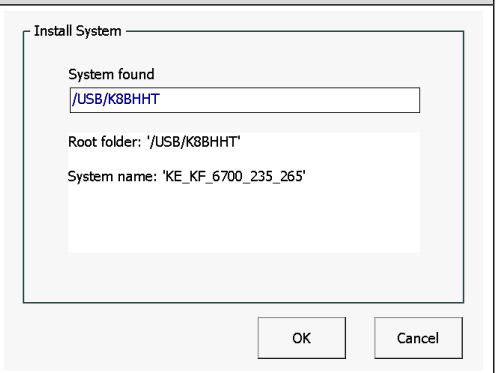
	Action	Illustration/Note
4	Select the installation package location using the browse button. The details of the selected installation package is displayed.	 xx1700001435 **Note** In case the deployment package contains an updated version of the Boot Application, then the Boot Application itself must first be upgraded and restarted before the installation can proceed. In this case, the user needs to repeat this step and select the desired installation package again after the restart.
5	Select the required package.	 xx1700001436

Continues on next page

	Action	Illustration/Note
6	A verification is started to find a match between the controller settings in the robot controller and the controller settings in the controller manifest file. Following are the possible scenarios that arise after this verification: • Controller manifest file is not available - In this case continue with the procedure mentioned in *Procedure - If there is no Controller manifest file on page 173*. • Controller settings (controller ID or name) matches with the controller manifest file settings - In this case continue with the procedure mentioned in *Procedure - If the Controller settings matches with the settings in the controller manifest file on page 175*. • Controller settings (controller ID or name) does not match with the controller manifest file settings- In this case continue with the procedure mentioned in *Procedure - If the Controller settings does not matches with the settings in controller manifest file on page 177*.	 xx1600001460 **Note** It is recommended to rename the controller first so that the name matches one of the controller entries in the prepared deployment package. The controller name can be modified from the **Settings** page of the Boot Application. If no controller definition is found in the manifest, then a system must be selected manually by browsing the installation package.

Procedure - If there is no Controller manifest file

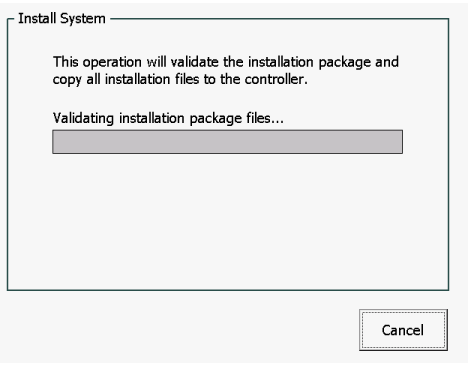
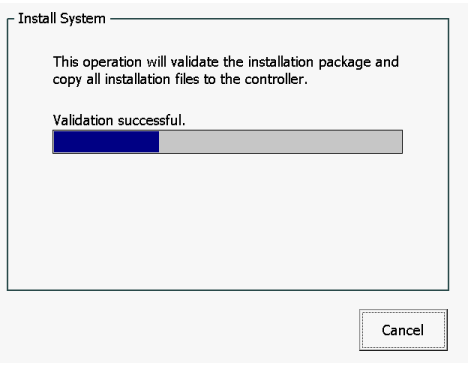
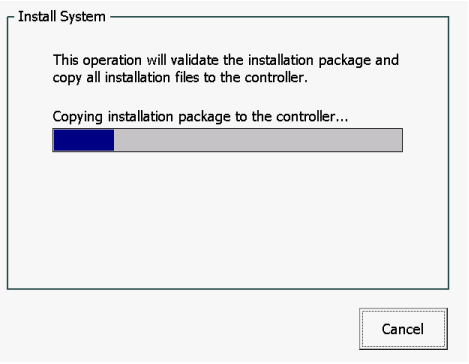
Use the following procedure to continue with the installation of manifest file is not available:

	Action	Illustration/Note
1	Select a system and tap **OK**.	 xx1600001461

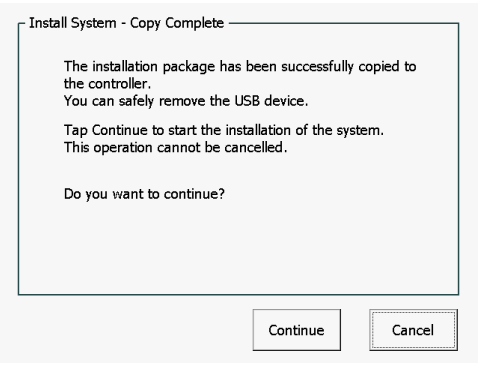
Continues on next page

6.3.2 Installing a RobotWare system using Boot Application

Continued

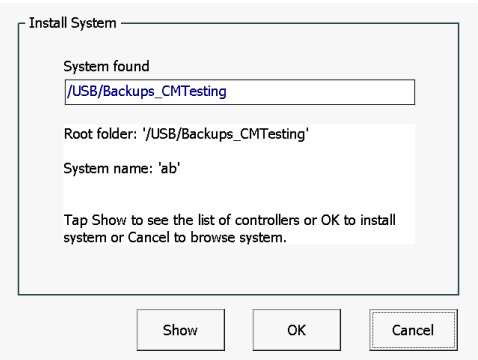
	Action	Illustration/Note
2	The validation of the installation package is started.	 xx1700001359
3	Once the validation is successful a message is displayed.	 xx1600001367 **Note** If the installation package is not valid, a message is displayed and the installation is terminated.
4	Copying of the installation package to the controller is started.	 xx1700001360

Continues on next page

	Action	Illustration/Note
5	Once the installation package is copied to the controller, the **Install System - Copy Complete** message is displayed.	 xx1600001368
6	Tap **Continue**. The system installation process is started. Once the installation is complete the controller is restarted with the new system.	

Procedure - If the Controller settings matches with the settings in the controller manifest file

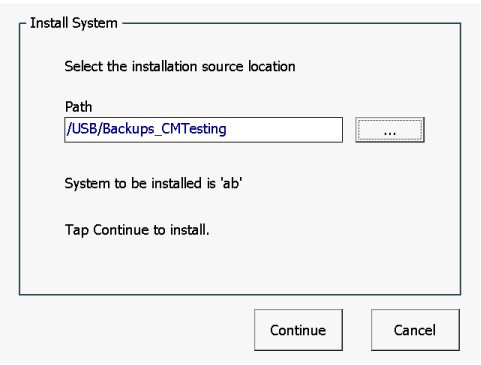
Use the following procedure to continue with the installation if the Controller settings (controller ID or name) matches with the settings in the controller manifest file:

	Action	Illustration/Note
1	If the Controller settings (controller ID or name) matches with the controller manifest file settings tap **OK**.	 xx1700001357 **Note** It is recommended to rename the controller first so that the name matches one of the controller entries in the prepared deployment package. The controller name can be modified from the **Settings** page of the Boot Application.

Continues on next page

6.3.2 Installing a RobotWare system using Boot Application

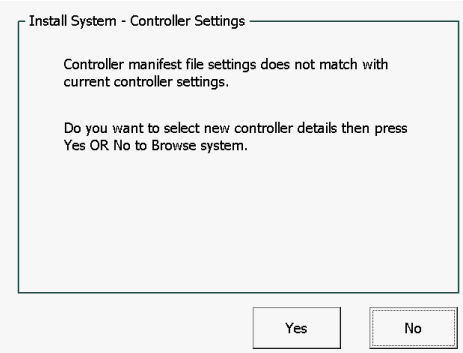
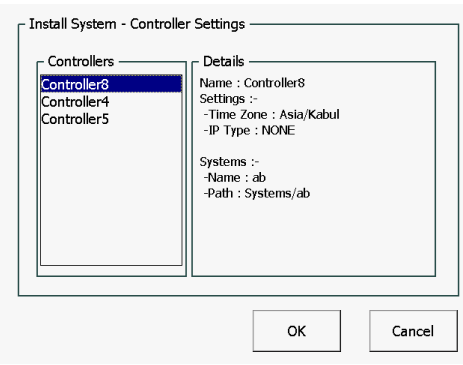
Continued

	Action	Illustration/Note
2	Tap **Continue**.	 xx1700001358
3	The validation of the installation package is started.	
4	Once the validation is successful a message is displayed.	**Note** If the installation package is not valid, a message is displayed and the installation is terminated.
5	Copying of the installation package to the controller is started.	
6	Once the installation package is copied to the controller, the **Install System - Copy Complete** message is displayed.	
7	Tap **Continue**. The system installation process is started. Once the installation is complete the controller is restarted with the new system.	

Continues on next page

Procedure - If the Controller settings does not matches with the settings in controller manifest file

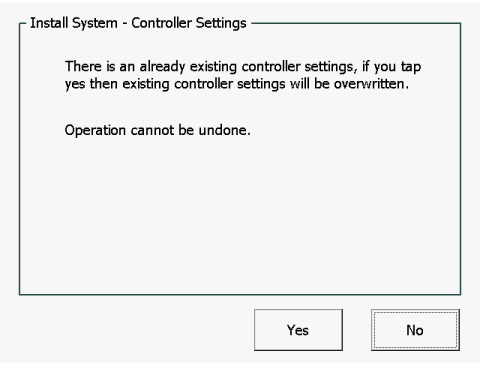
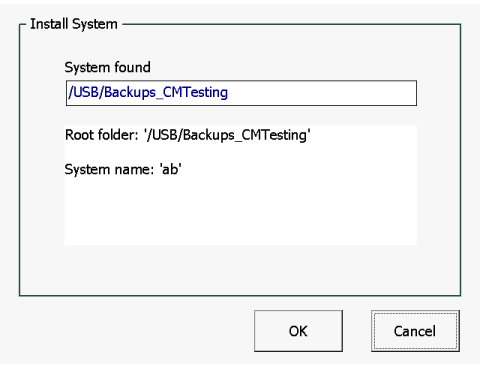
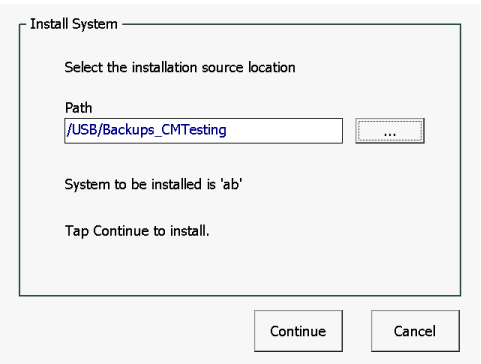
Use the following procedure to continue with the installation if the Controller settings (controller ID or name) does not match with the settings in controller manifest file:

	Action	Illustration/Note
1	If the Controller settings (controller ID or name) does not match with the settings in controller manifest file tap **Yes**. The **Install System-Controller Settings** window is displayed.	 xx1700001428 **Note** It is recommended to rename the controller first so that the name matches one of the controller entries in the prepared deployment package. The controller name can be modified from the **Settings** page of the Boot Application.
2	Select a controller from the **Controllers** list and tap **OK**. **Note** Only the controllers without a defined serial number are visible in the list.	 xx1700001429

Continues on next page

6.3.2 Installing a RobotWare system using Boot Application

Continued

	Action	Illustration/Note
3	Tap **Yes**.	 xx1700001430 **Note** Tapping **No** starts the procedure to create a new system.
4	Tap **OK**.	 xx1700001431
5	Tap **Continue**.	 xx1700001432
6	The validation of the installation package is started.	
7	Once the validation is successful a message is displayed.	**Note** If the installation package is not valid, a message is displayed and the installation is terminated.

Continues on next page

	Action	Illustration/Note
8	Copying of the installation package to the controller is started.	
9	Once the installation package is copied to the controller, the **Install System - Copy Complete** message is displayed.	
10	Tap **Continue**. The system installation process is started. Once the installation is complete the controller is restarted with the new system.	

6.3.3 Updating a RobotWare system

Description

The most frequent RobotWare system update use case is updating one or more products, for example, RobotWare Add-Ins. This is a frequent operation during the commissioning time, especially on large installations. To prepare an update, an update deployment package is created. It contains only the products that are upgraded.

The version of the product is used for defining the upgrade. The update mechanism compares the products already installed on the controller with those of the update package. A product can be updated only when fulfilling both of the following conditions:

- The product already exists on the controller.
- The product version of the update package is higher than the one which is already installed, in which the order of comparing the version number components is: first Major, then Minor, then Build, and finally Revision.

The update package can be made available in one or all of the following locations:

- System inbox
- Controller USB-port
- FlexPendant USB-port

Note

Note that update packages may be generated from a repository that contains no controller or system definitions, since update packages are generic and can be applied to any number of systems.

Note

To perform a RobotWare system update, the system must be in active state and the Controller must be in manual mode.

Note

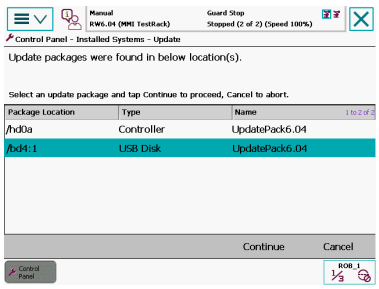
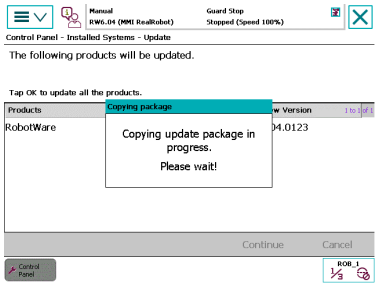
The user needs the following UAS grants to perform an update:

- Administration of installed systems.
- Backup and save.
- Update a RobotWare system.

The following procedure provides the steps involved during the update of the RobotWare system.

	Action	Note
1	Insert the USB-stick with the update package into a USB-port on the controller or the FlexPendant. If the update package is already available in the system inbox, start directly with step 2.	

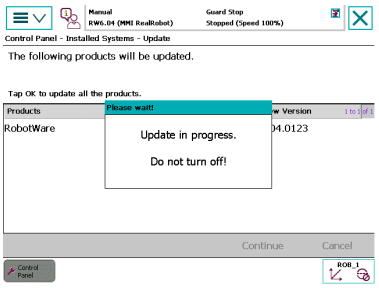
Continues on next page

	Action	Note
2	Tap **ABB Menu**, **Control Panel**, and **Installed Systems**. A search if the update system package is available at the system inbox, FlexPendant USB-port, and controller USB-port is done. Once the search is complete and an update package is found the **Update** button is displayed.	**Note** If the Update button is not displayed, check that the user has the required UAS grants.
3	Tap **Update**. The update packages found at various locations are displayed.	
4	Select the update package from the required location and tap **Continue**.	 xx1600001369
5	The validation of the update package is started. Once validated an update summary is displayed. Tap **Continue**.	**Note** If the update package is not valid an error message is displayed and the update package process is cancelled. **Note** A message is displayed if the system is already up to date.
6	The update package is copied to the currently active system's inbox. Once the files are copied a message is displayed. Tap **Continue**. The update confirmation message is displayed.	 xx1600001370
7	Tap **Yes**. The system backup message is displayed.	**Note** If **No** is selected, then the update package is saved in the controller inbox for later use.

Continues on next page

6.3.3 Updating a RobotWare system

Continued

	Action	Note
8	Tap **Continue**. The system update process is started. Once the update is done, the controller is restarted with the new updated system.	 xx1600001363 **Note** If the update fails an error message is displayed and the update process is terminated. **Note** During a successful update process, the controller is restarted automatically. If specified so by the processed update package, the backup created in the initial phase of the update process is automatically loaded back into the updated system, causing an additional restart.

6.4 The recovery disk function

Introduction

The recovery disk function is for recreating a complete installation of RobotWare and Add-Ins directly to a SD-card in the PC. The recovery disk function can be found under both the **Controller** tab and the **Repository** tab in Installation Manager.

Prerequisites

The recovery disk function uses information from a backup to recreate the RobotWare system that backup has been generated from.

For successful operation these prerequisites are needed:

- A backup for a system using RobotWare 6.0 or later, but not later than the version of the currently used RobotStudio.
- All products and Add-Ins used by the backup should be located on the pc (RPK:s and RMF:s).
- A SD-card reader.
- An original ABB Robotics SD-card.
- Administrative rights on the PC.

SD-card creation procedure

1. When the disk recovery function is started, the user will be asked to browse for the backup folder. Once a valid backup directory is selected, the **Write Disk** button is active.
2. Before the **Write Disk** process is started, you will be asked if you want to make a disk image for product defect report. If yes, define a file name and where the information should be stored and then click **Save**. All data will be copied from the disk to the selected location.

   Note

   If this function is used, RobotWare defect reports defining reasons for disk recovery can be sent without having to send in the actual disk.

3. The program will then try to locate all products and add-ins referenced in the selected backup. If one or more product cannot be found, a second directory browser will appear and the user will be asked to manually locate the product that could not be found. The program will remember the parent directory of the selected product, and include this directory in later searches.
4. When all products are located, the user will be asked to insert a SD-card into the computer, or if already present the user will first be asked to remove the card. This is to ensure that SD-cards are not overwritten by mistake.
5. Once the SD-card is inserted the user is prompted to confirm that the SD-card will be permanently overwritten. From this point on and until the operation is complete, the SD-card will be unusable in any other sense than to be used by the disk recovery function again.

Continues on next page

6 The operation is completed after about one minute, depending on the SD-card reader performance, and the user is asked to remove the SD-card.

Installation procedure

> **CAUTION**
>
> Always power off the robot controller before changing the SD-card.

1 Power off the robot controller.
2 Replace the SD-card in the controller with the recovery SD-card.
3 Power on the controller.
4 The controller completes the operation by resetting the system to default values.

   The text *Controller is resetting system* might appear on the FlexPendant depending on how long this operation takes.
5 The RobotWare system starts up.

   For RobotWare versions greater or equal to 6.04, the backup selected in the recovery disk creator is automatically restored.

The backup can be found with its original name in the following folder: */hd0a/BACKUP*.

Erase data from SD-card

The Clean Disk function will only work on an SD card inserted in the SD card reader of the PC. All data on the SD card will be deleted and a boot application will be installed.

> **CAUTION**
>
> The SD-card will be formatted and all information on it will be irreversibly deleted.
>
> If any license file shall be saved, please save a backup of the system which contains the license file on your PC.

1 Insert the SD-card to be emptied into the SD-card reader of the PC.
2 In the **Robot Recovery Disk Creator**, select **Clean Disk** and click on **Write Disk**.

The cleaning may take a few minutes (a progress bar shows the progress). When it is finished the SD-card can be removed from the PC and used in a robot controller. A new RobotWare system can now be installed on it.

6.5 Limitations

Introduction

This list of limitations and known issues will help to succeed using the repository, installation, update, and recovery in it's present state.

Repository limitations

- Each of the products in the *Products* directory must have its own folder where all files which belong to the product are stored. That is, sharing folders to store installation files that belong to different products is not supported.
- A system definition can only refer to one base system.
- Duplicate selections of products and backups are possible in a system, i.e. the same product or backup is selected in both a base system definition and a system referring to the base system definition. This can cause a problem if e.g. RobotWare is referred to in both the base system and the application system but with different versions.
- All options that exist in all products (licensed and non-licensed) are visible and available for selection when creating and editing a system definition. When installing the system, a check is made that the controller contains the necessary licenses to cover the selected options. It is possible to add licenses into the system definition in the repository and they will be included in the installation package, however they will not restrict the user selection possibilities when defining the system. The license check is always done during the installation and it is based on licenses that are already present on the controller together with licenses included in the installation package.
- Products and backups cannot be organized in a hierarchical folder structure.
- Currently the option selection tree structure is collapsed when opening the panel in the system definition. The preferred way would be to expand the tree to show the selected options upon entry.

Installation limitations

- Automatic restore of backups is not supported.
- Spaces and special characters in names of files and folders, for example % and &, are not supported. The installation will fail if such characters are used.
- It is recommended to use the USB slots on the controller rather than the USB slot on the FlexPendant since the copying is much faster.

Update limitations

- Update only handles update of products. It cannot add or delete products, neither can it downgrade products nor can it change the option selection.
- An update package must be located directly in the root folder of a USB stick. No other files than the update package files shall exist on the USB stick, since that may cause the update package to be rejected by the controller.
- It is recommended to use the USB slots on the controller rather than the USB slot on the FlexPendant since the copying is much faster.

Continues on next page

- If a user has insufficient UAS grants to perform an update the **Update** button will not be enabled. For description of the required grants, see *Deploying installation and update packages on page 170*.

Recovery limitations

- Only an original ABB Robotics SD-card can be used.
- Occasionally the recovery disk tool freezes when closing the window. Use the Windows Task Manager to close the application.
- Due to a limitation in windows, network drives might not be available for this function if the UAC function of Windows® is set to prompt. There are workarounds for this problem posted by Microsoft© and others on diverse internet-forums.
- To be able to manage installation compatibility this function can never make a recovery disk with a RobotWare of version higher than the version of the used RobotStudio.

7 Calibrating

7.1 Robot calibration

About robot calibration

The procedures for fine calibration of a robot and updating revolution counters are different for different robots. For instructions, see the product manual for the robot.

7.2 How to check if the robot needs calibration

Check robot calibration status

This section describes how to check the robot's calibration status.

	Action
1	On the **ABB** menu, tap **Calibration**.
2	In the list of mechanical units, check the calibration status.

What kind of calibration is needed?

If the calibration status is...	then...
Not calibrated	the robot must be calibrated by a qualified service technician.
Rev. counter update needed	You must update the revolution counters. How to update the revolution counters is described in the product manual for the robot.
Calibrated	No calibration is needed.

DANGER

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.

7.3 Loading calibration data using the FlexPendant

Overview

This section describes how to load calibration data for using the FlexPendant.

The calibration data is normally stored on the serial measurement board of each robot, regardless of whether the robot runs an absolute measurement system (*Absolute Accuracy* option is installed, *AbsAcc*) or not. This data is normally transferred automatically to the controller when the system is powered up, and in such cases no action is required by the operator.

Verify that the correct Serial Measurement Board (SMB) data has been loaded into the system as detailed below. In a MultiMove system, this procedure must be repeated for each robot.

Load calibration data

This table describes how to load the calibration data.

	Action
1	On the FlexPendant, tap the **ABB** menu, tap **Calibration**, and then select a mechanical unit.
2	Tap **Manual Method (Advanced)**. The calibration window for the selected mechanical unit is displayed.
3	Tap **Robot memory** and then tap **Show status**. The status for controller and robot memory is displayed.
4	If **Valid** is displayed under the headings **Controller Memory** and **Robot memory**, calibration data is correct. If not, the data (on the SMB board or in the controller) must be replaced with the correct data as detailed below: • If, for instance, the SMB board has been replaced, transfer the data from controller to SMB board. If the controller has been replaced, transfer the data from the SMB board to the controller. • Transfer data by tapping **Robot Memory**, **Update**, and then selecting which data to update.
5	After loading the calibration data, proceed with updating the revolution counters.

7.4 Editing motor calibration offset

Editing motor calibration offset

This procedure should be used when no specific file with motor calibration data is available, but only the numerical values. These values are normally found on a sticker on the rear of the robot.

Entering motor calibration values can be done in three ways:

- From a disk, using the FlexPendant (as detailed in section *Loading calibration data using the FlexPendant on page 189*).
- From a disk, using RobotStudio (as detailed in *Operating manual - RobotStudio*).
- Manually entering the values, using the FlexPendant (as detailed in section *Editing motor calibration offset on page 190*).

	Action	Information
1	On the **ABB** menu, tap **Calibration**.	
2	Tap to select mechanical unit and then tap **Calibration Parameters**.	
3	Tap **Edit motor calibration offset....** A dialog box is displayed, warning that updating the revolution counters may change programmed robot positions: • Tap **Yes** to proceed. • Tap **No** to cancel. Tapping **Yes** results in displaying a file selection view.	
4	Tap the axis to have its motor calibration offset edited. The offset value box is opened for that particular axis.	
5	Use the numerical keyboard to enter the value and then tap **OK**. After entering new offset values, a dialog box is displayed, urging you to restart the system to make use of the new values. Do a warm restart if required.	
6	After restarting, the contents of the calibration data in the controller cabinet and on the serial measurement board will differ. Update the calibration data.	Described in section *Serial measurement board memory on page 191*
7	Update the revolution counters.	Described in the product manual for the robot.

7.5 Serial measurement board memory

Serial measurement board (SMB)

The serial measurement board (SMB) primarily gathers resolver data from the robot's (or additional axes) motors. This data is used to measure the speed and position of each axis. Each SMB is capable of measuring up to 7 axes. It also stores a number of data pertaining to each robot.

This data is used by the controller and can be transferred between the SMB and the controller. Normally, the data is transferred automatically, but it can also be done manually.

The SMB data is affected when:

- The robot is replaced
- The SMB is replaced
- The controller (or its flash disk or mass memory unit) is replaced.
- Updating with new calibration data

The following data is stored on the SMB:

- Serial number for the mechanical unit
- Joint calibration data
- Absolute accuracy data
- SIS data (Service Information System)

Note that if the IRC5 controller is to be connected to a robot with an older SMB, not equipped with data storage capability, the SMB must be replaced.

SMB data update

If...	then...
the **flash disk** or mass memory or the complete **controller** is new or replaced by an unused spare part...	the data stored in the SMB is automatically copied to the controller memory.
the **SMB** is replaced by a new, unused, spare part SMB...	the data stored in the controller memory is automatically copied to the robot SMB memory.
the **flash disk** or the complete **controller** is replaced by a spare part, previously used in another system...	the data in the controller memory and the robot SMB memory is different. You must **update the controller memory manually** from the the robot SMB memory.
the **SMB** is replaced by a spare part SMB, previously used in another system...	the data in the controller memory and the robot SMB memory is different. You must first **clear the data in the new robot SMB memory**, and then **update the robot SMB memory** with the data from the controller memory.
new **calibration data** has been loaded via RobotStudio or using the FlexPendant and the system has been restarted...	the data in the controller memory and the robot SMB memory is different. You must **update the robot SMB memory manually** from the controller memory. Check that the new calibration values belong to a manipulator with the serial number defined in your system.

Continues on next page

View SMB data status

This section describes how to view the data status in the serial measurement board and the controller.

	Action
1	On the **ABB** menu, tap **Calibration** and select a mechanical unit.
2	Tap **Robot Memory** and then tap **Show status**. The data is displayed with status on the controller memory and on the robot SMB memory.

Update controller data from robot SMB memory

This section describes how to load data from the serial measurement board **to** the controller.

	Action	Information
1	On the **ABB** menu, tap **Calibration** and select a mechanical unit.	
2	Tap **Robot Memory** and then tap **Update**.	
3	Tap the button **Cabinet or manipulator has been exchanged**. A warning is displayed. Tap **Yes** to proceed or **No** to cancel.	It is vital that you load calibration data correctly.
4	The data is loaded. Tap **Yes** to acknowledge and restart the robot system.	The following data is updated: • Serial numbers for mechanical units • Calibration data • Absolute accuracy data • SIS data

Update data in robot SMB memory

This section describes how to update data on the serial measurement board **from** the controller. This is e.g. after calibration data has been loaded to the controller via RobotStudio or using the FlexPendant.

If the SMB already contains data, you must first clear the memory, see *Delete SMB data on page 193*.

	Action	Information
1	On the **ABB** menu, tap **Calibration** and select a mechanical unit.	
2	Tap **Robot Memory** and then tap **Update**.	
3	Tap the button **Serial measurement board has been replaced**. A warning is displayed. Tap **Yes** to proceed or **No** to cancel.	It is vital that you load calibration data correctly.
4	The data is updated.	

Continues on next page

Delete SMB data

This section describes how to delete the data stored on the robot SMB memory or the controller memory when creating spare parts.

	Action
1	On the **ABB** menu, tap **Calibration** and tap to select a mechanical unit.
2	Tap **Robot Memory** and then tap **Advanced**. The following functions are available: • Clear controller memory • Clear robot memory
3	Tap **Clear Controller Memory** if the controller should be replaced and used as a spare part. A list of the SMB data stored in the controller is displayed. Tap **Clear** to delete the controller memory for the selected robot. Repeat the procedure for all robots in the controller memory.
4	Tap **Clear Robot Memory** if the SMB should be replaced and used as a spare part. A list of the SMB data stored in the robot SMB memory is displayed. Tap **Clear** to delete the memory for the selected robot. Repeat the procedure for all robots using this SMB board.

Related information

Operating manual - RobotStudio.

Operating manual - Service Information System.

Application manual - Controller software IRC5

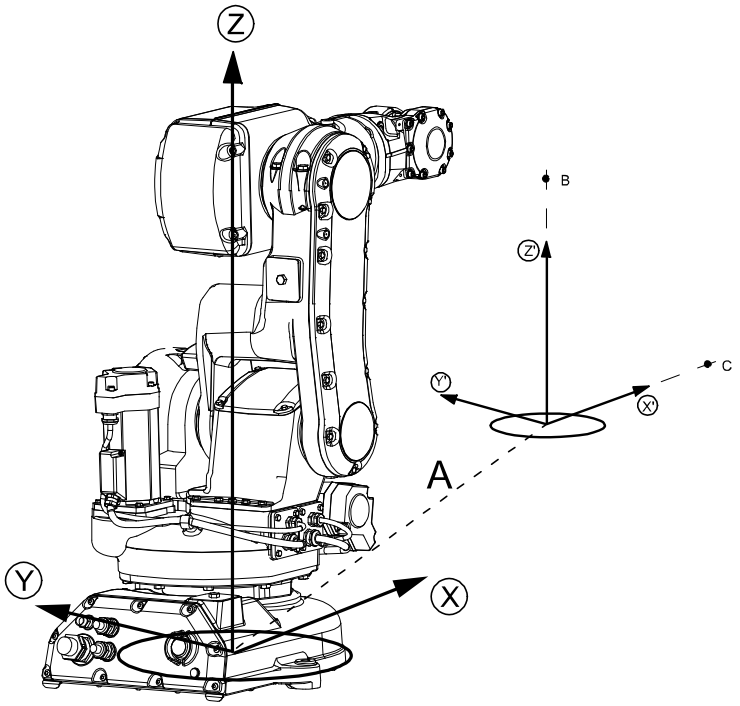
7.6 4 points XZ calibration

Base Frame calibration

This section describes the 4 points XZ calibration, in the Base Frame calibration options. Other calibration methods can be available in this menu depending on your installed options.

Overview

This section describes how to define the base frame using the 4 points XZ method. This method can move and rotate the base frame in relation to the world frame. Normally the base frame is centered and aligned with the world frame. Note that the base frame is fixed to the base of the robot.



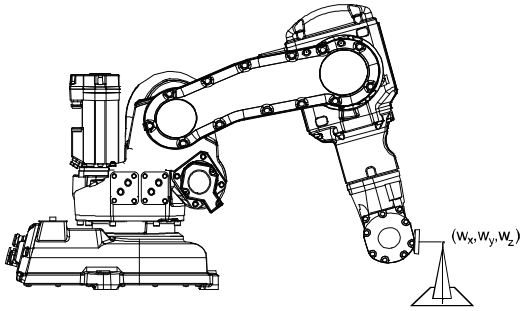
xx0400000782

A	Displacement distance between base frame and world frame
B	Elongator point Z’
C	Elongator point X’
X	X-axis in the base frame
Y	Y-axis in the base frame
Z	Z-axis in the base frame
X'	X-axis in the world frame
Y'	Y-axis in the world frame
Z'	Z-axis in the world frame

Continues on next page

Fixed reference Position

The calibration procedure requires that the tip of the tool is calibrated against a fixed reference position. The fixed position could be a manufactured World fixed tip device to facilitate finding the elongator points. The fixed reference position is the distance in meters (in (x,y,z)) between the fixed position and the world frame.



Calibrate_xx

Running 4 points XZ calibration

	Action	Information
1	On the **ABB** menu, tap **Calibration** and select a mechanical unit. Then tap **Base Frame**.	
2	Tap **4 points XZ...**.	
3	Set up a fixed reference position within the working range of the robot.	
4	Tap **...** to change reference point. Enter the coordinates of the fixed reference position (in meters). A numerical keyboard and boxes for X, Y and Z values are displayed.	
5	If the calibration positions exists in a file, follow the instructions below. Otherwise proceed to the next step • Tap **Positions** menu and then **Load** the file containing the values.	
6	Tap **Point 1** to highlight the line.	
7	Manually jog the robot to the previously fixed reference point.	
8	Tap **Modify position**. **Modified** is displayed on the status line.	
9	Re-orient the robot and again, run it to the reference point but from a different angle.	Repeat these steps until points 1, 2, 3, and 4 have been modified.
10	Tap **Elongator X** and manually run the robot to a position where the tool center point (TCP) touches an imaginary extension of the X-axis.	The imaginary X-axis is shown in the illustration above.
11	Tap **Modify position**. **Modified** is displayed on the status line.	Repeat these steps to modify **Elongator Z**.
12	To save the entered transformation data to a file, tap the **Positions** menu and then **Save**. Enter the name of the file and then tap **OK**.	

Continues on next page

7.6 4 points XZ calibration
Continued

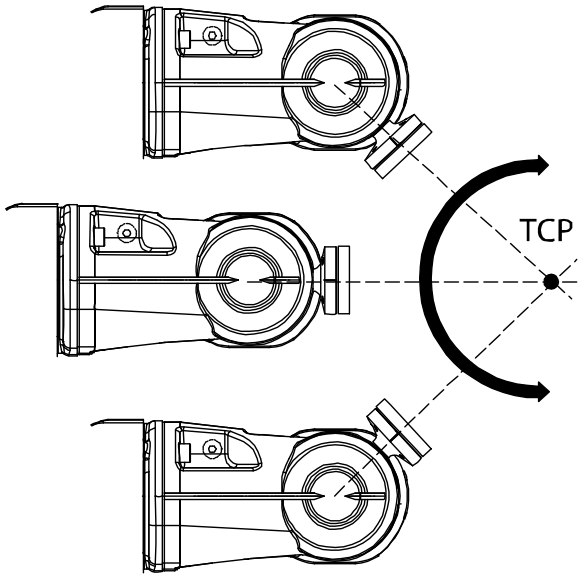
	Action	Information
13	To delete all entered transformation data, tap the **Positions** menu and then **Reset All**.	

8 Descriptions of terms and concepts

8.1 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.

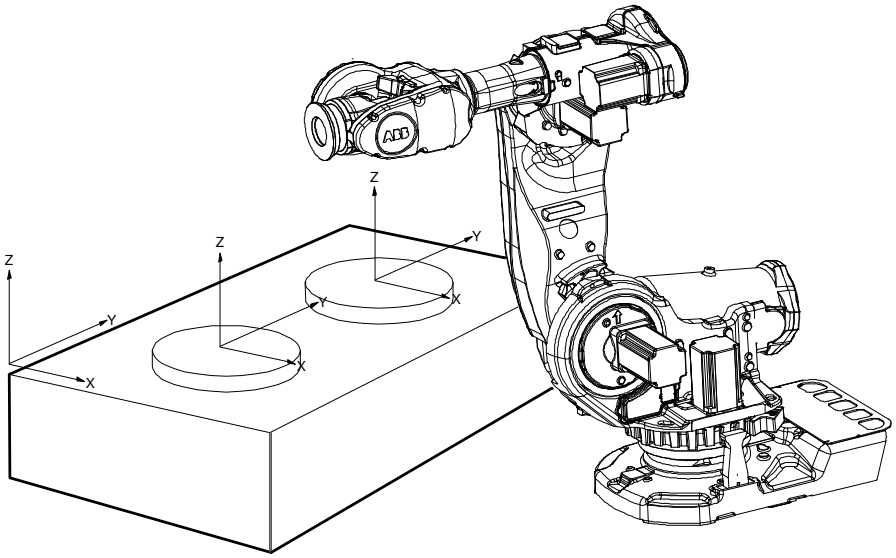
Continues on next page

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.

8.2 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.

8.3 What is a coordinate system?

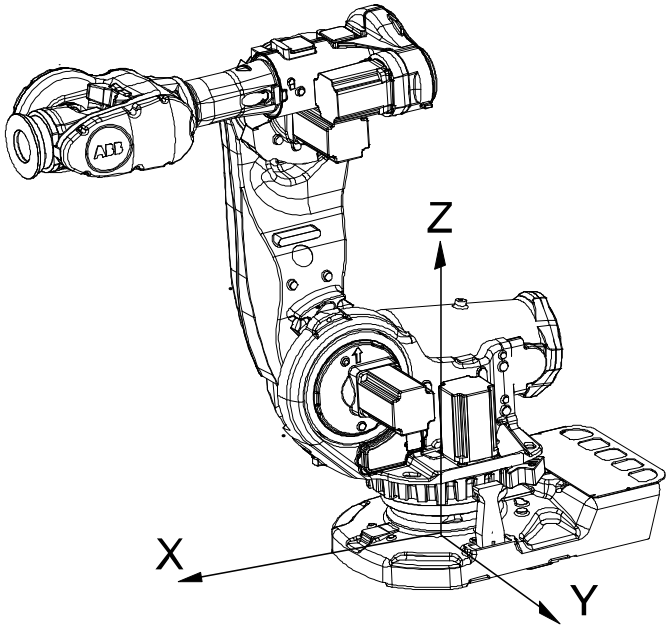
Overview

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. See *The base coordinate system on page 200* for more information.
- The *world coordinate system* that defines the robot cell, all other coordinate systems are related to the world coordinate system, either directly or indirectly. It is useful for jogging, general movements and for handling stations and cells with several robots or robots moved by external axes. See *The world coordinate system on page 201* for more information.
- The *user coordinate system* is useful for representing equipment that holds other coordinate systems, like work objects. See *The user coordinate system on page 202* for more information.
- The *work object coordinate system* is related to the work piece and is often the best one for programming the robot. See *The work object coordinate system on page 203* for more information.
- The *tool coordinate system* defines the position of the tool the robot uses when reaching the programmed targets. See *The tool coordinate system on page 204* for more information.

The base coordinate system



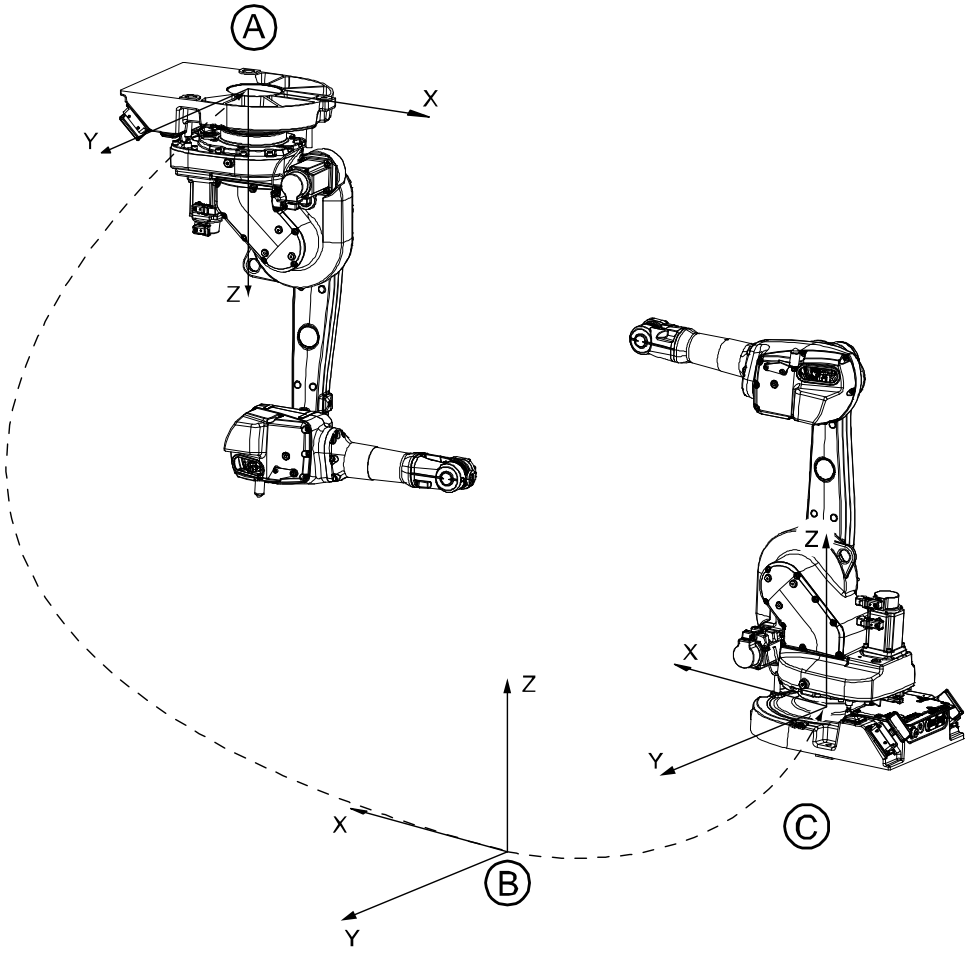
xx0300000495

Continues on next page

The base coordinate system has its zero point in the base of the robot, which makes movements 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 better choices. See *The work object coordinate system on page 203*for more information.

When you are standing in front of the 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 world coordinate system



en0300000496

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

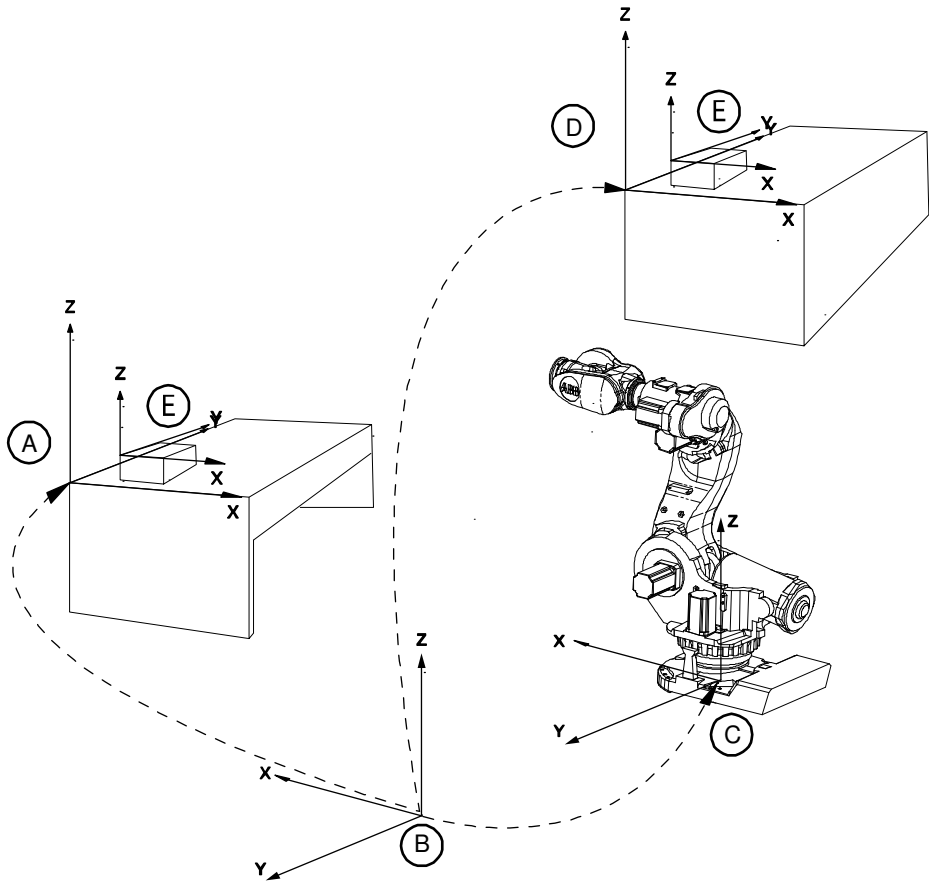
The world coordinate system has its zero point on a fixed position in the cell or station. This makes it useful for handling several robots or robots moved by external axes.

By default the world coordinate system coincides with the base coordinate system.

Continues on next page

8.3 What is a coordinate system?
Continued

The user coordinate system



en0400001225

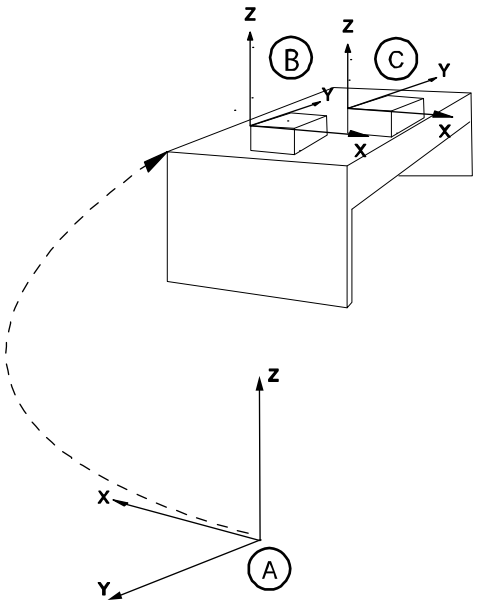
A	User coordinate system
B	World coordinate system
C	Base object coordinate system
D	Moved user coordinate system
E	Work object coordinate system, moved with user coordinate system

The user coordinate system can be used for representing equipment like fixtures, workbenches. This gives an extra level in the chain of related coordinate systems, which might be useful for handling equipment that hold work objects or other coordinate systems.

For information on how to define the user coordinate system, see information about the data type `wobjdata` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Continues on next page

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).

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.

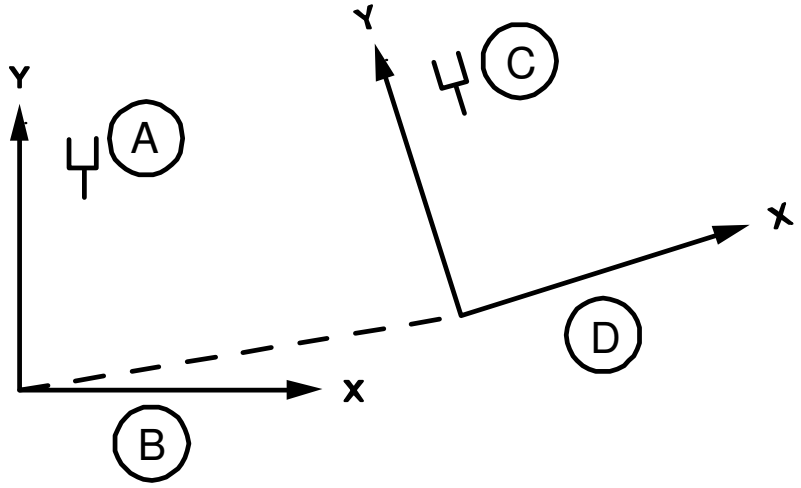
It is in work object coordinate systems you create targets and paths when programming the robot. This gives a lot of advantages:

- 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.

For information on how to define the work object coordinate system, see information about the data type `wobjdata` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Continues on next page

The displacement coordinate system



en0400001227

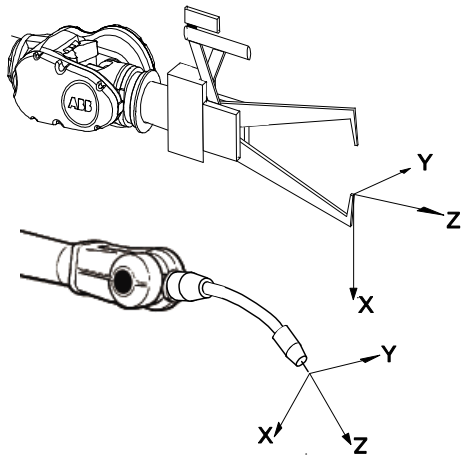
A	Original position
B	Object coordinate system
C	New position
D	Displacement coordinate system

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 work object coordinate system.

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

Continues on next page

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 the robot 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 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 instance moving a saw blade without bending it.

For information on how to define the tool coordinate system, see information about the data type `tooldata` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

8.4 What is mirroring?

Description

Mirroring creates a copy of a program, module, or routine in a specific mirror plane. The mirror function can be applied to any program, module, or routine.

Mirroring can be performed in two different ways:

- Default against the base frame coordinate system. The mirror operation will be performed across the xz-plane in the base frame coordinate system. All positions and work object frames that are used in an instruction in the selected program, module or routine are mirrored. The position orientation axes x and z will be mirrored.
- Advanced against a specific mirror frame. The mirror operation will be performed across the xy-plane in a specified work object frame, mirror frame. All positions in the selected program, module or routine are mirrored. If the work object argument in an instruction is another work object than specified in the mirror dialog, the work object in the instruction is used in the mirror operation. It is also possible to specify which axis in the position orientation that will be mirrored, x and z or y and z.

Note

The mirroring function recognizes the used workobject in all predefined motion instructions and in user made procedures with the same argument declaration:

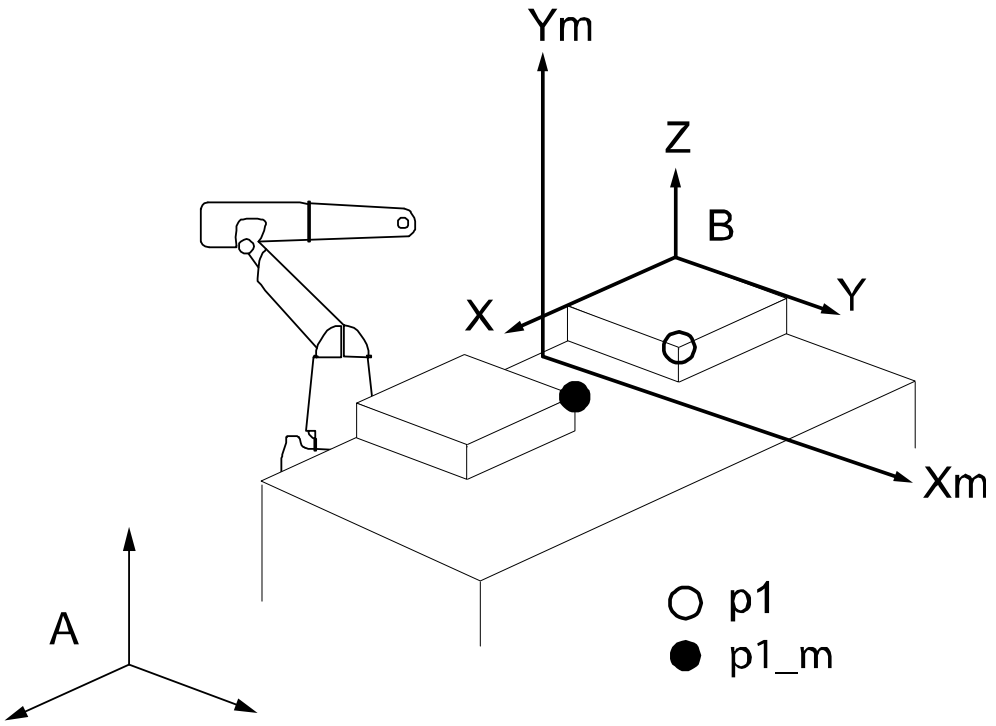
- an argument for the robtarget,
- an argument for the tooldata with name **'Tool'** and
- an optional argument for the wobjdata with the name **'Wobj'**.

The following descriptions of mirroring describes advanced mirroring.

Continues on next page

Mirror plane

The mirror function will mirror all positions (robtargets) in the mirror plane, i.e. the mirrored position will be located symmetrically on the other side of the plane, relative to the original position. The mirror plane is always the xy-plane of an object frame, used for mirroring. This object frame is defined by a work object data, e.g. with the name MIRROR_FRAME.



xx0600002815

Ym, Xm	Mirror plane
A	World frame
B	Work object frame
p1	Original point
p1_m	Mirrored point

Mirroring routines

Mirroring creates a copy of a routine with all positions (robtargets) mirrored in a specific mirror plane. In general, all data of the type robtarget used in the routine, both local and global, will be mirrored. It makes no difference whether the robtarget data is declared as a constant (which it should be), as a persistent, or as an ordinary variable. Any other data, e.g. of type pos, pose, orient, etc., will not be mirrored.

Mirroring data only affects the initialization value, i.e. any current value will be ignored. This means that if a robtarget variable has been defined without an init value, this variable will **not** be mirrored.

The new, mirrored routine will be given a new name (a default name is proposed). All stored data of type robtarget, used in the routine, will be mirrored and stored

Continues on next page

with a new name (the old name ending with “_m”). All immediate robtarget data, shown with an “*”, in movement instructions will also be mirrored.

Mirrored values and arguments

When mirroring a routine, the new routine is scanned for any local robtarget data, declared inside the routine with an init value. All init values of such data are mirrored. Then the new routine is scanned for statements with one or more arguments of type robtarget.

When such a statement is found, the following actions will take place:

- If the argument is programmed with a reference to a local variable or a constant, this argument will be ignored, since it has already been mirrored as described above.
- If the argument is programmed with an immediate robtarget data, shown with an asterisk“ *”, then this value will be mirrored directly.
- If the argument is programmed with a reference to a global variable, persistent or a constant, defined outside the routine with an init value, then a duplicate is created and stored in the module with a new name (the old name ending with “_m”). The init value of this new data is mirrored, and then the argument in the statement is changed to the new name. This means that the module data list will expand with a number of new mirrored robtarget data.

Error handlers or backward handlers in the routine are not mirrored.

Work object frame

All positions which are to be mirrored are related to a specific work object frame (B in figure above). This means that the coordinates of the robtarget data are expressed relative to this work object frame. Furthermore, the mirrored position will be related to the same work object frame.

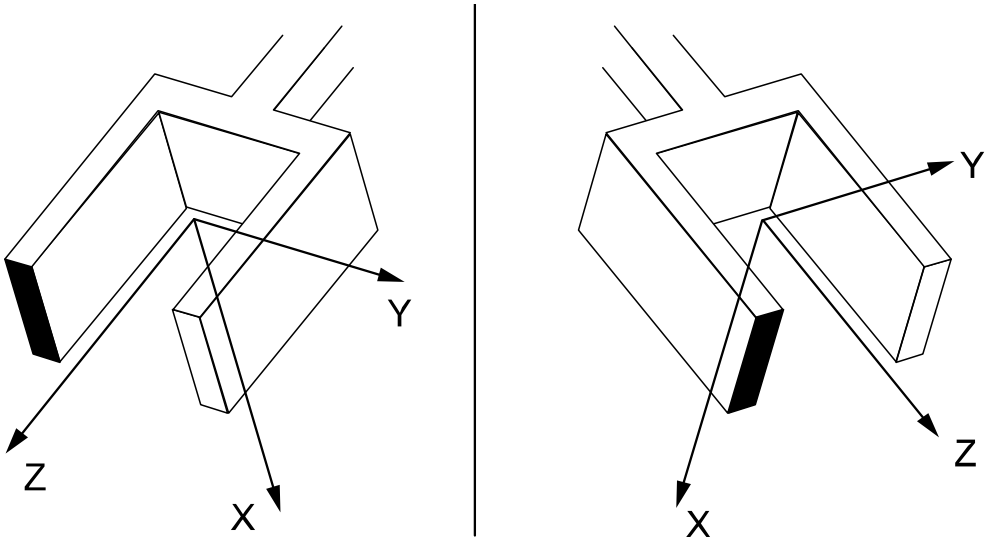
Before mirroring, this specific work object must be stated. This work object will be used as the reference frame for all variables that are to be mirrored.

Make sure to state the same work object as was originally used when defining the robtarget data, and which was used as a parameter in the movement instructions. If no work object was used, the wobj0 should be stated.

Continues on next page

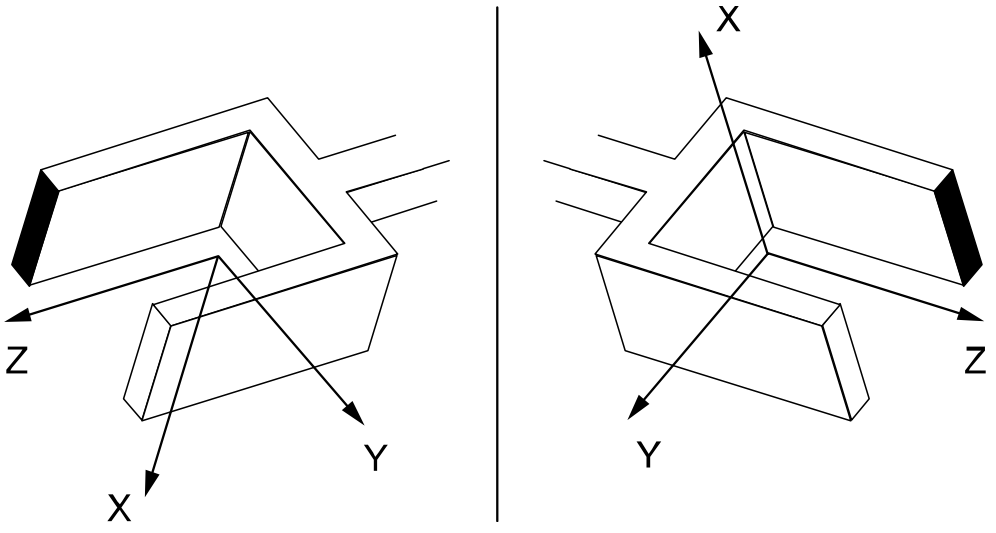
Orientation of mirrored positions

The orientation of the robtarget position is also mirrored. This mirroring of the orientation can be done in two different ways, where either the x and z axes are mirrored or the y and z axes. The method used, x or y axis (the z axis is always mirrored), is dependent on the tool used and how the tool coordinate system is defined.



xx0600002816

	Mirroring of x and z axes.



xx0600002817

	Mirroring of y and z axes.

Arm configurations

The arm configuration will not be mirrored, which means that after mirroring, it has to be carefully checked by executing the path in manual mode. If the arm configuration has to be changed, this must be done manually and the position corrected with a modpos command.

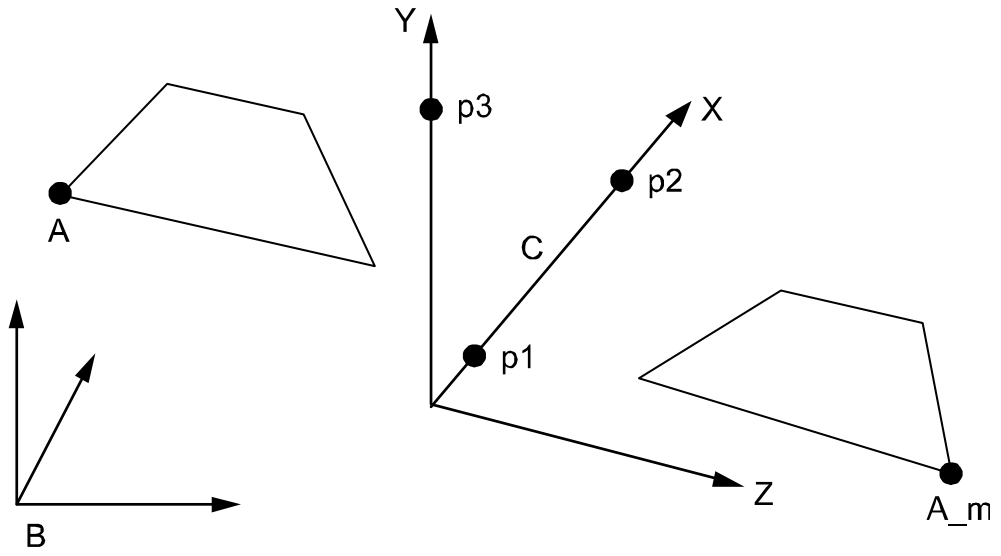
Continues on next page

8.4 What is mirroring?
Continued

Example 1: Mirroring with one robot

A mirrored copy of the routine `org` is to be created and stored with the name `mir`. All positions are related to the work object, wobj3. The mirror plane is known from three positions in the plane, p1, p2, and p3.

An original position in `org`, A, is mirrored to A_m.



xx0600002818

A	Original position
A_m	Mirrored position
B	Object frame wobj3
C	Mirror plane

To perform this mirroring, the mirror frame must first be defined. To do this, create a new work object and name it (e.g. `mirror`). Then, use the three points, p1 to p3, to define the object coordinate system by using the robot. This procedure is described in *Defining the work object coordinate system on page 80* .

After this, the routine, `org`, can be mirrored using wobj3 and mirror as input data.

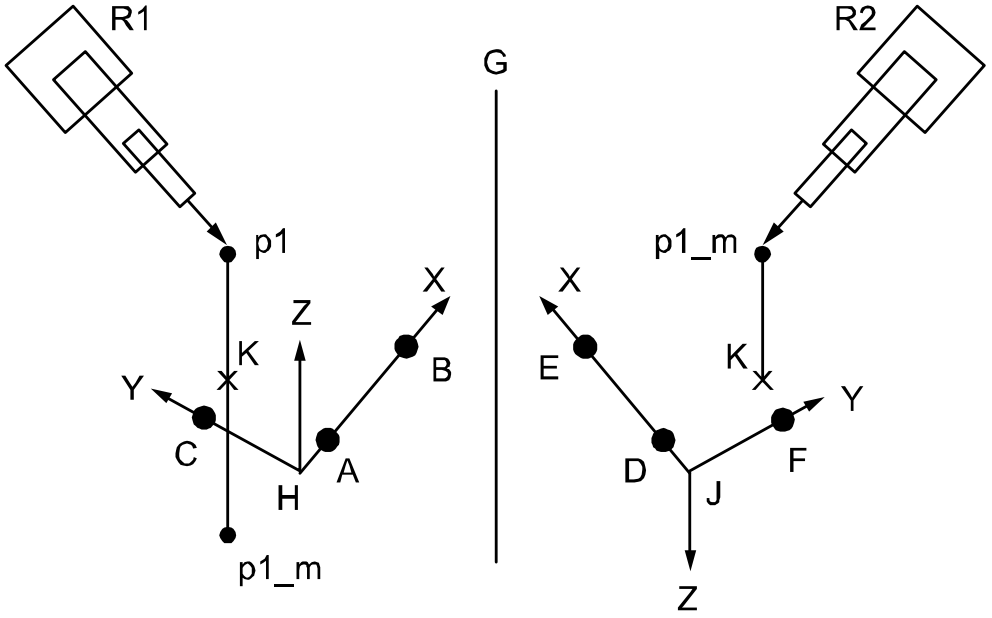
Example 2: Mirroring with two robots

The routine `org` was created on one robot and should be mirrored and used on another robot. Suppose that a spot welding robot, robot 1, is used for the left side of a car body. When the program for the left side is done, it should be mirrored and used again for the right side by robot 2.

The original program, `org`, is programmed relative to a work object, wobj1, which is defined with the help of three points, A, B and C on the left side of the car body. The mirrored program, `mir`, is to be related to a corresponding work object, wobj1, defined by the corresponding points D, E and F on the right side of the car body. Wobj1 for robot 2 is defined with robot 2.

Continues on next page

Note that since the points D, E, F are mirrored images of points A, B, and C, the wobj1 for robot 2 will also be mirrored. One of the consequences of this is that the z-axis will point downwards.



xx0600002819

R1	Robot 1
R2	Robot 2
G	Virtual mirror plane
H	wobj1 = mirror frame
J	wobj1 for robot 2
K	Projection of p1 in xy-plane
p1	Original position
p1_m	Mirrored position

After the work object, wobj1, has been defined, all programming is done in this frame. Then the program is mirrored using the same wobj1 frame as the mirroring frame. A position, p1, will be mirrored to the new position p1_m.

After this, the mirrored program is moved to robot 2, using the work object wobj1, as described above. This means that the mirrored position, p1_m, will be “turned up” as if it were mirrored in a “virtual” mirror plane between the two robots.

This page is intentionally left blank

9 Cyber security for IRC5 robot networks

9.1 Introduction

About this chapter

This chapter discusses security in a network installation with ABB IRC5 systems.

First, it gives an overview about IRC5 and other products typically deployed in a network installation and the communications between them. A security analysis identifies the most critical assets and security threats targeting them. To mitigate these security threats, every ABB IRC5 controller implements UAS (User Authorization System).

Security features, however, such as UAS and testing, are only components of an effective security strategy. It is equally important to define, implement, and maintain a security policy, which covers security processes, procedures, and mechanisms. This chapter lists requirements for such a security policy.

Section *IRC5 application protocols on page 228* contains descriptions of the communication protocols used by IRC5 products, which may be useful for configuring elements of the system security architecture, as well as a list of useful security tools.

Disclaimer

The intent of this chapter is to raise awareness about critical security threats and to provide guidance to address them as well as to inform how ABB is working on security assurance. However, due to the high number of different security risks and complex dependencies within actual installations, this document can neither cover all possible security risks, nor guarantee the success of the presented security mechanisms.

The benefits and risks of using open networking technology for robot controllers

ABB IRC5 products use standard Internet transport protocols, TCP and UDP. This means that IRC5 products can be connected to a normal network (TCP/IP/Ethernet) like any other computer or network product, which reduces costs and unifies network management. Furthermore, the interconnection of control systems and "office" systems, such as ERP, enable a wide range of new applications, which take advantage of such vertical integration from the shop floor up to the enterprise management. Section *Network architecture and communication on page 215* describes a typical IRC5 robot network.

However, the direct connection of control systems to the plant network also creates security risks (for example, malware infections (viruses, worms, Trojans), denial of service, disclosure of confidential data). Section *Security analysis on page 218* discusses these security threats in detail.

Continues on next page

Mitigating the risks through a comprehensive security policy and architecture

It is generally accepted that the security features of a product or system are only one part in a successful protection strategy. It is equally important to define, implement, and maintain an effective security policy, which covers (among other issues) risk analysis, procedures, responsibilities, and regular auditing. Section *Security policy on page 220* discusses general and IRC5 specific requirements for a security policy and shows, how such a security policy can be used to mitigate critical security threats targeting a robot control system. It is important to note though that security cannot be achieved by a one-time investment in a product or process but requires ongoing effort to operate and maintain.

9.2 Network architecture and communication

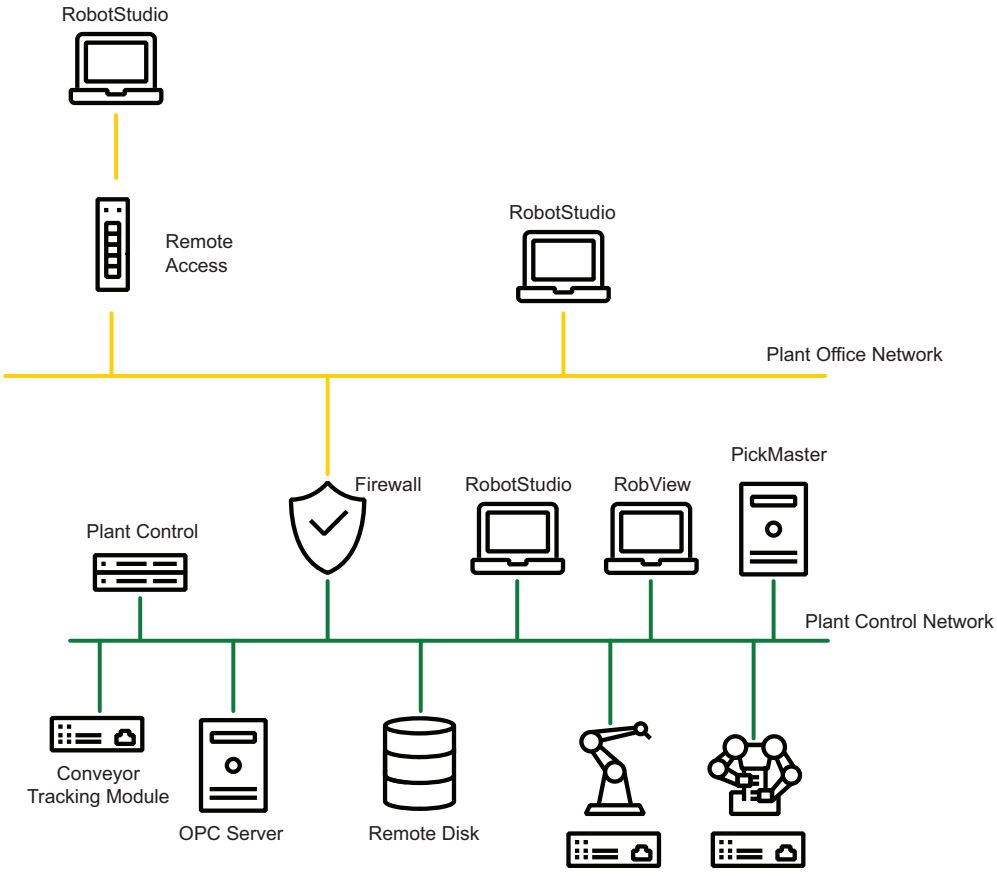
About this section

This section gives an overview about typical components of an IRC5 installation, which are attached to the network, and the communication between them. Furthermore, this section serves as basis for the threat analysis (*Security analysis on page 218*) and the requirements to a security policy (*Security policy on page 220*).

Simplified network with IRC5 products

The following figure shows a simplified network with IRC5 products. The intention of this picture is to show typical components, which may be part of a robot control system installation, and where they may be installed.

The objective of the shown network security architecture is to protect the plant control network from threats, which are originating in the plant office network and especially in remote networks and which are caused for example by viruses and worms. Therefore, it is strongly suggested to separate the plant control network and the plant office network with a protection device, such as a firewall.

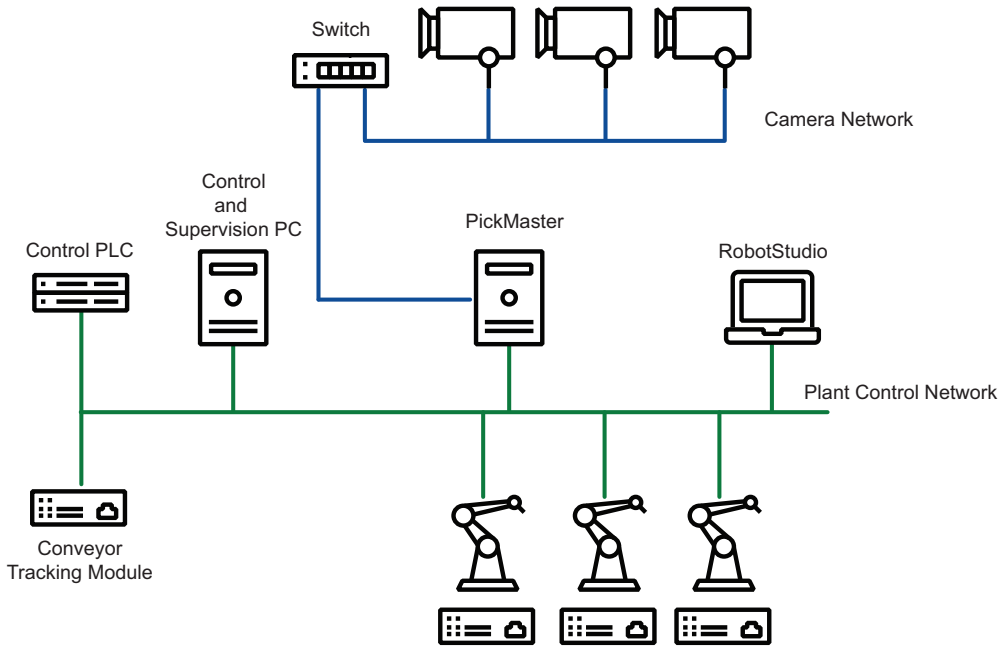


xx1600000613

Continues on next page

Network with PickMaster

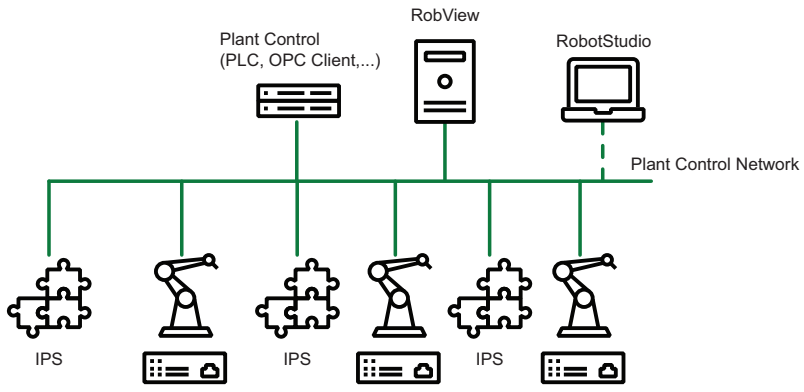
The Plant Control Network architecture details with the PickMaster product is slightly different and has other security aspects as shown in the following figure. In a PickMaster installation, the cameras are connected directly to the PC or via a switch to the PC over a separate network than the Plant Control Network.



xx1600000614

Network with IRC5P and paint robots

The following figure shows the details of the network in a Paint installation. In these installations, RobotStudio may be temporarily connected (dotted line) and each IRC5P control cabinet has 2 connections to the Plant Control Network.

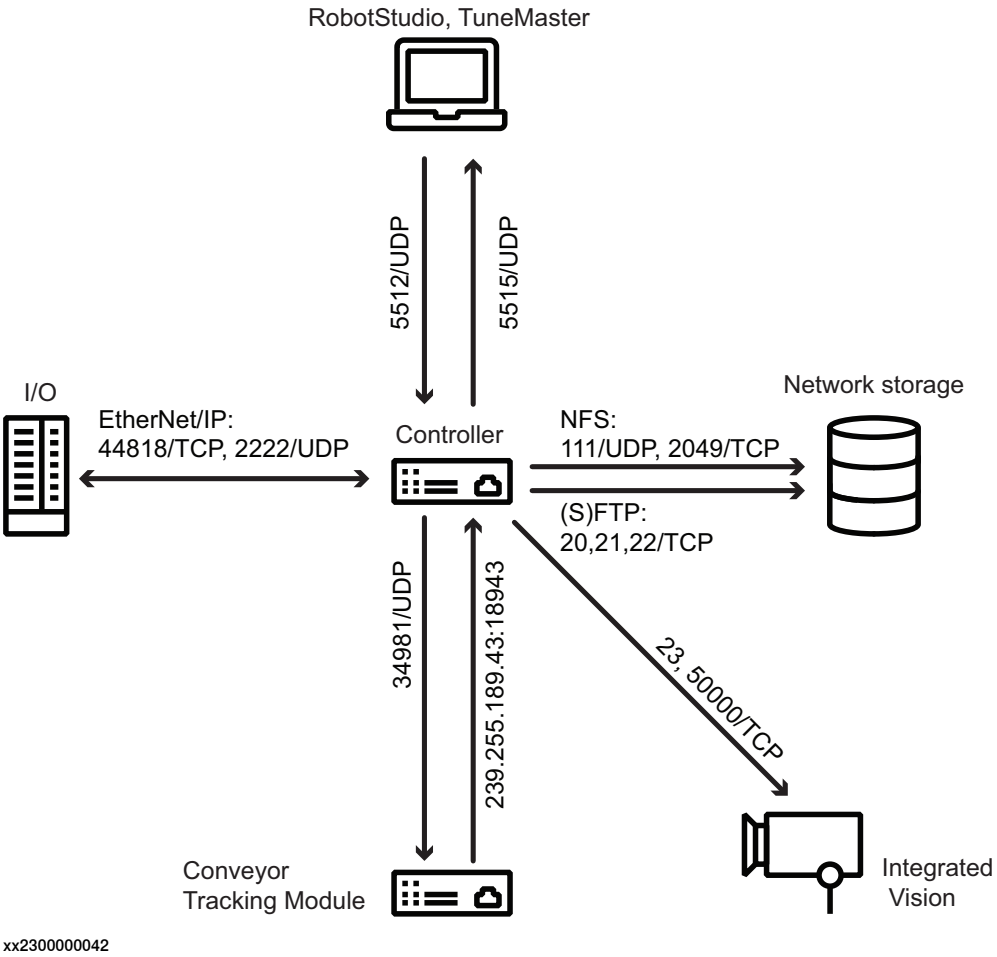


xx1600000615

Continues on next page

Communication protocols between IRC5 related products

The following figure shows the communication protocols between the IRC5 related products. For detailed information about the application protocols, see *IRC5 application protocols on page 228*.



9.3 Security analysis

Why do you need a security analysis?

Network architectures, in which the plant control network is attached to an office network, which in turn is connected to the Internet, potentially expose the plant control network to cyber attacks. Therefore, it is important to identify the critical security threats and to implement security mechanisms to prevent or at least mitigate them.

Critical assets to protect

The main assets to be protected are the main computer of the IRC5 robot controller, the Integrated Process System (IPS) controller, the RobView PC, the Conveyor Tracking Module, and the PickMaster PC, because any reduced availability or unauthorized access may cause significant financial loss due to damaged semi-finished goods or loss of production. Unauthorized changes to configuration files on the devices can have a direct impact on the correct functioning and availability of the controlled robots and processes.

OPC servers also represent critical assets, since they have direct access to the variables in the robot controllers. Further critical assets are the RobotStudio PCs, since they communicate directly with the IRC5 robot controllers and, if compromised, may be used as entry points to the devices.

Critical security threats

The following table summarizes critical security threats specific to ABB Robotics IRC5 products, the attacks causing them, and the targeted assets.

Threat	Attacks causing the threat	Targeted assets
Disturbance / DoS	Sending of large amount of data Upload of invalid RAPID programs Upload of invalid IPS configurations	IRC5 robot controller IPS controller Conveyor Tracking Module
Unauthorized access Unauthorized control	Brute force password attack Network Sniffing	IRC5 robot controller IPS controller in the IRC5P RobView PC RobotStudio PC PickMaster PC OPC Server Conveyor Tracking Module

Besides those presented threats, there are also a number of other generic security threats (for example, media, such as USB sticks, connected to workstations may introduce viruses or unprotected wireless access may open doors for hackers). Therefore, the requirements for a security policy described in section *Security policy on page 220* cover both generic and ABB Robotics IRC5 specific requirements to prevent/mitigate such security risks.

9.4 IRC5 User Authorization System

About the User Authorization System

The User Authorization System (UAS) limits which individuals (with specific roles) can perform which operations on the controller by defining the users and groups that can access the controller and the functionality. Depending on which group a user is assigned to, the user is granted access to certain functionality, while other functionality will be inaccessible. Which functionality the members of a certain group have access to, is controlled by assigning a list of grants to that group.

Two types of grants exist: controller grants and application grants. Controller grants are predefined by ABB, are validated by the robot controller, and apply to all tools and devices, which access the controller. Application grants may be added by application developers, may contain an additional value field, and are used and valid only within a specific application (for example, FlexPendant).

Mastership (write access)

The UAS implementation also supports the so called "mastership" locking concept, which can be used by a client to guarantee that as long as the client holds the mastership of the robot controller, no other client is allowed to send commands to the controller. As default, the mastership is held by the FlexPendant, but other clients can request write access.

Default User

At delivery, an ABB IRC5 system has a user named "Default User" that has a number of grants. After creating new user with specific grants, the Default User can be removed. If the Default User is active, but all grants are removed, there are still reading rights. So if you want to prevent unauthorized personnel from viewing any content on the IRC5 controller, deactivate the Default User.

Note

If the Default User is deactivated, it is necessary to login on the FlexPendant every time you use it.

How to use the User Authorization System

For information about how to configure and use UAS, see *Operating manual - RobotStudio*.

9.5 Security policy

9.5.1 Introduction

Overview

Often vulnerabilities are introduced into industrial control systems due to the lack of a well-defined, accurate, and enforced security policy. Therefore, the security policy plays an essential role in the reduction of vulnerabilities and the defense against and mitigation of security threats.

It is the responsibility of the owners of the control systems to define their security policy according to their specific requirements, such as: how to identify users (authentication), who is allowed to access what (authorization), and what should be audited regularly (audit). Once the security policy is defined, it has to be implemented and applied to all covered software, hardware, systems, data, networks, and personnel within the control system owner's organization resulting in a security architecture consisting of technical and procedural means. The security policy and its implementation in a security architecture have to be maintained continuously, since organizational changes, upcoming and evolved regulations, and new technologies have all an impact on the security policy. Therefore, security is not a one-time initiative, but an on-going process.

This section describes security requirements that should be addressed by the security policy of the control system owner. The proposed requirements are grouped into two categories: the first one is generic and the second one is specific to ABB Robotics IRC5 products. Note that the listed requirements are not exhaustive and that they should be tailored to the specific requirements, the size, and available resources of the control system owner's organization.

9.5.2 General security requirements

About general security requirements

This section only lists a number of things to consider in any computer network. It is not a complete list.

Physical security

All operations shall be tracked and servers, backup media, and other associated equipment shall be placed in locked machine rooms or cabinets. Access to these areas shall be restricted to dedicated employees. All access levels and responsibilities shall be explicitly documented.

Further physical security requirements:

- Physical data interfaces, such as CD and DVD drives, and USB ports, are locked or disabled.
- Network components, like switches and routers, shall be enclosed in locked cabinets.
- Tamper detection of unauthorized access (for example, inspecting the sealings)
- Only authorized personnel gets access to network components and cables.
- Unless necessary, wireless devices should not be connected to the plant control network. A rationale shall be documented for each exception.

Note

The information on SD card in the robot controller is not encrypted.

Anyone who has access to physical devices, such as SD cards and USB ports, can get access to information that may be sensitive.

Account management and network/system access

Account management and access control requirements define how user accounts and passwords are managed and how user rights and access to the network and the control system are controlled.

Procedures for account management (e.g., adding of a new account, revocation of an existing account, and changing or assigning of a new password) shall be documented. Employees, who are assigned to this function, shall be trained in those procedures.

Further requirements:

- Accounts are locked after a certain period of inactivity.
- Only authorized personnel get access to PCs and its interfaces, especially to PCs in the plant control network.
- Use the principle of "least privilege", i.e. use roles/grants with only the necessary rights to perform the task.
- Replace default accounts with personalized accounts.

Continues on next page

9.5.2 General security requirements
Continued

- Only controlled and protected computers are allowed to be connected to the plant control network.
- The definitions of authorized users, user groups, and access rights are continuously maintained to reflect properly the current authorities.
- Users lock the screen or log off before leaving the workplace.
- A password-protected screen-saver is activated after a certain idle time.
- After a specified number of consecutive log-on failures, a user account is locked for a certain time or until it is unlocked by a system administrator.

Passwords

A password policy should be in place. This may include the following rules:

- Passwords shall expire, forcing users to change passwords regularly.
- All password authentications within the network shall be encrypted.
- Only "strong" passwords are used.
- User and password lists are protected from unauthorized access.
- Passwords are not written down.
- Passwords are not shared between users.

Administration and patching of servers and workstations

All systems shall be kept updated according to the vendor's recommendations (e.g., new patches addressing critical vulnerabilities shall be applied as soon as they have been successfully tested and verified).

The hardening of operating systems and applications shall be regularly reviewed (e.g., unused ports shall be closed, unused services uninstalled, unneeded application features disabled, and demo or default application data moved or deleted).

Only authorized personnel are allowed to change system configuration and to install new software.

Virus protection

Anti-virus software shall be installed on every workstation and server, for which it is available. Anti-virus software and virus definitions shall be updated regularly.

All software to be installed on systems in the plant control network, shall be first checked for malware (viruses, worms) on a separate virus scanning PC, which is not connected to the plant control network.

E-mail

Internet and e-mail services may serve as carriers for viruses, worms, spyware, and other kind of malware to penetrate the plant control network. Therefore, systems in the plant control network shall not be allowed to access arbitrary Internet sites and e-mail services.

Continues on next page

Firewall

Firewalls that protect plant control networks shall provide stateful filtering and preferably offer application level support for the forwarded protocols (e.g., deep packet inspection). To protect plant control networks from flooding and denial-of-service attacks, the firewalls shall offer rate-limiting functionality.

If a firewall is used as VPN endpoint, it shall support state-of-the-art VPN protocols.

The firewalls shall be configured to allow only authorized traffic from dedicated source addresses and source ports to dedicated destination addresses and destination ports. Furthermore, the firewalls shall be locked down and need to be regularly maintained (i.e., patched, upgraded, and proper change management including regular audits for access rules).

Backup and recovery

All critical data shall be backed-up periodically and stored in a secure place. For ABB Robotics IRC5 products, backups should especially cover the data stored on the IRC5 controller including RAPID and configuration files as well as configurations on the Integrated Process System (IPS).

A good backup policy should also include the configurations and parameters from PickMaster, Conveyor Tracking Module, and RobView as well as the host Window PC configuration.

Note

An IRC5 backup contains unencrypted information. Make sure that all backup files are stored in a secure location.

Vulnerability scanning and risk assessment

In many IT environments, the use of vulnerability scanners, such as Nessus, NMAP, Metasploit, etc, are used to find and assess the potential vulnerabilities in IT equipment. These tools perform extensive probing and conduct a representative set of attacks on equipment. Because of the potential for disruption to the IRC5 controller, the recommended best-practice in the industry is not to perform these types of tests on production equipment but only on equipment in a controlled laboratory environment. Performing these types of scans and tests on the IRC5 controller while in production has the potential of disrupting the normal operation of the IRC5 and its communications to other devices on the network.

Cybersecurity procedures and policies

- Background checks, instructions and training of personnel and subcontractors.
- Guide on what is allowed to do, using which tools, by who, and when.
- Logging and cybersecurity monitoring methods in automation systems and networks.

Continues on next page

9.5.2 General security requirements
Continued

Maintenance and audits

Perform maintenance of the ongoing responsibilities and actions to ensure that the security policy is followed, kept up-to-date, and adapted to organizational changes.

Periodic testing and reviews of the security policy are required.

Disposal

Before disposal of any storage equipment (anything from an SD card to an IRC5 controller), make sure all sensitive information has been deleted.

Tip

To remove all data from the SD card, use the Clean Disk function (part of Recovery Disk function) in RobotStudio. See *Operating manual - RobotStudio*.

9.5.3 ABB Robotics IRC5 product specific requirements

Remote access/client

Remote access allows users to access company networks and systems from computers that are located outside of the protected company. In the context of an ABB robot control network, these are hosts running RobotStudio. Since the computers running RobotStudio may directly access systems in the plant control network, they extend the perimeter of the plant control network and may therefore create security risks.

To mitigate these risks, the following actions are suggested (in descending order of preference):

1. Avoid remote clients
2. Use remote terminal services, which are protected within a secure tunnel
3. Tunnel communication protocol(s) through a VPN and authenticate communication partners

Since remote clients represent security risks, they should be avoided whenever possible. Any remote connection shall be justified by business reasons.

Any remote host, which has been identified to require connection to the protected network, has to be hardened according to security host hardening best practices, which include, but are not limited to:

- Dedicated machine for remote access (i.e., not the same as for daily business)
- No other simultaneous network connections
- Only required services and processes are installed and running
- Only required network ports are active
- Restrictive access control
- Up-to-date patches / services packs / upgrades are applied
- Up-to-date anti malware software, such as a virus scanner, is running
- Regular maintenance intervals

Clients in Plant Office network

RobotStudio and other clients may also exist on hosts in the plant office network. Conveyor Tracking Module, RobView and PickMaster are not designed to be run outside the Plant boundary. Therefore, the same suggestions as for the remote clients (see *Remote access/client on page 225*) also apply to the clients in the plant office network.

Although, the plant office network is usually already protected by firewalls against the Internet and other networks, it still represents a security threat to the plant control network, since applications with high security risks, such as e-mail and Web browsers, are run within the plant office network. Therefore, a separation of the plant control network from all other networks, including the plant office network, using firewalls and preferably, also a DMZ, is strongly suggested.

The use of terminal services for access from clients in the plant office network to systems in the plant control network is, with respect to security, still preferred against the pure tunneling of communication protocols. However, since the risk

Continues on next page

caused by systems in the plant office network is lower than that caused by remote systems, tunneling may represents an acceptable alternative. It is still strongly suggested to deny any unprotected communication between the plant office network and the plant control network.

Clients in Plant Control network

The PickMaster and RobView clients in the Plant Control Network may keep a local cache of the IRC5 UAS credentials used to authenticate to the IRC5 controller. This is part of the functionality of these clients when presenting their own levels of users, groups, and grants specific to the operation of the client product.

Given that credentials are stored on the host Windows PC, these clients are important clients to secure and ensure that local and remote access is well controlled.

Conveyor Tracking Modules in plant control network

RobICI, SFTP, and IEEE1588/PTP v2 are used for the communication with Conveyor Tracking Modules. RobICI is an internal ABB protocol. For more details, see *IRC5 application protocols on page 228*.

IEEE1588/PTP v2 is used to improve conveyor tracking performance. SFTP is used for file transfer with RobotStudio clients.

To improve security and stability, it is recommended to block RobICI and IEEE1588/PTP v2 network traffic to Conveyor tracking Modules from outside the plant control network, for example, by using a firewall.

At delivery, a Conveyor Tracking Module has the following two preinstalled users with default passwords:

Name	Default password	Description
`abbadmin`	`abbadmin`	`abbadmin` is used for advanced maintenance and troubleshooting. For example, this user is required for firmware upgrades.
`ctmuser`	`ctmuser`	`ctmuser` is used for everyday maintenance and troubleshooting.

Note

It is recommended to change the default passwords to improve security.

WARNING

There is no way to recover a lost password. Make sure passwords are never lost or forgotten. Appropriate credentials are required to make any modifications to the CTM configuration and firmware.

RobAPI and Robot Web Services

RobAPI and Robot Web Services are used for the communication to the IRC5 robot controller. While RobAPI is an internal ABB protocol over TCP/IP, it also includes FTP traffic as well. For more details, see *RobAPI application protocol on page 230*.

Continues on next page

To improve security, RobAPI and Robot Web Services connections to IRC5 robot controllers from outside the plant control network should be additionally protected by an encrypted tunnel (VPN) between the client and the firewall (e.g., IPSec). Note that the use of a protected terminal server (see *Remote access/client on page 225*), which runs a RobotStudio instance which is installed inside the plant control network, would prevent the need for RobAPI and Robot Web Services packets to cross network boundaries.

UAS administration

IRC5 UAS provides access control to the controller (as described in section *IRC5 User Authorization System on page 219*). There are two requirements concerning UAS administration:

- In the factory configuration, UAS has a built-in "Default User" account that is assigned to the "Default" group, which holds administrative grants. Therefore, it is essential to change this association before a controller is deployed into a productive environment. The access rights of the Default User are changed by replacing its association to the Default group with another group, which has only a limited number of grants assigned. It is also possible to remove the Default User, in order to remove all rights to view information, but this means a login must always be performed on the FlexPendant.
- Since UAS can be disabled by a console command, it is important that only a limited number of authorized users have access to the console port in the cabinet of the controller. This can be realized by physical security mechanisms and procedures.

9.6 IRC5 application protocols

Overview

The IRC5 services and application protocols are presented in separate sections for default and configured/enabled protocols. The tables also define the network connections that may be used for each service/application. For detailed information about all network connections, see *Network connections on the IRC5 main computer on page 232*.

For more information about port number assignation, see *www.iana.org*.

Default services and application protocols

Service, or Application	Port number	Transport protocol	Network connection	Usage/Comments
DHCP server	68	UDP	LAN Service	Dynamic Host Configuration.
FTP server	20 21	TCP	WAN LAN Service	File transfer, software installation/upgrade.
NetScan	5512 5513 5514	UDP	WAN LAN Service	Detection of available IRC5 robot controllers on the network.
Robot Network Protocol (RNP/RobAPI)	5515	TCP	WAN LAN Service	Communication with IRC5 robot controller.
Robot Web Service	80	TCP	WAN LAN Service	Communication with IRC5 robot controller.

Configured/enabled services and application protocols

Service, or Application	Port number	Transport protocol	Network connection	Usage/Comments
ArcLink	Ports used are defined in the configuration for the UdpUc device.	UDP	WAN LAN Service	Support Lincoln ArcLinkXT power sources. Enabled by configuration.
Connected Services (HTTPS)	553	TCP	WAN	Secure connection to ABB Ability™ Cloud. Enabled by configuration.
DHCP client	67	UDP	WAN	Dynamic Host Configuration. Enabled by configuration.
DNS client	53	TCP, UDP	WAN	ABB Ability™ server name resolution. Enabled by configuration.
EGM (Google Protocol Buffers)	Ports used are defined in the configuration for the UdpUc device.	UDP	WAN LAN Service	Externally Guided Motion. Enabled by configuration. Requires option *689-1 Externally Guided Motion*.

Continues on next page

Service, or Application	Port number	Transport protocol	Network connection	Usage/Comments
EtherNet/IP messaging	44818	TCP UDP	WAN LAN	Enabled by configuration.
EtherNet/IP I/O	2222	TCP UDP	WAN LAN	Enabled by configuration.
FTP client	20 21	TCP	WAN LAN Service	Remote disk mounting with FTP. Enabled by configuration. Requires option *614-1 FTP SFTP and NFS client*.
ICI	233.253.77.88:12277 12288	UDP	LAN Service	I/O signals, command, and response. Enabled by configuration. Requires option *RobotWare Paint*.
IEEE1588/ PTP v1 or v2	319 320	UDP	WAN LAN Service	Time synchronization used by the RobICI protocol. Requires RobICI and the option *1550-1 Conv.Tracking unit Int.*, *1551-1 Conv.Tracking unit Ext.* or *RobotWare Paint*.
Integrated Vision Telnet client	23 50000 1069	TCP, UDP	LAN Service	Communication between controller and Cognex cameras. Enabled by configuration. Requires option *1341-1/1520-1 Integrated Vision Interface*.
NFS client	111 2049	TCP, UDP	WAN LAN Service	Remote disk mounting Enabled by configuration. Requires option *614-1 FTP SFTP and NFS client*.
NFS server	111 2049	TCP, UDP	LAN Service	Used by IPS to retrieve configuration files from robot controller. Requires option *RobotWare Paint*.
PROFINET RT	34962	UDP	WAN LAN	Enabled by configuration.
PROFINET RTM	34963	UDP	WAN LAN	Enabled by configuration.
PROFINET CM	34964	UDP	WAN LAN	Enabled by configuration.
PROFINET RPC	49152	UDP	WAN LAN	Enabled by configuration.
RobICI	239.255.189.43:18943 34981	TCP, UDP	WAN LAN Service	I/O signals, command, and response. Required for conveyor tracking module (CTM). Enabled by configuration. Requires option *1550-1 Conv.Tracking unit Int.* or *1551-1 Conv.Tracking unit Ext.*.

Continues on next page

Service, or Application	Port number	Transport protocol	Network connection	Usage/Comments
SFTP client	22	TCP	WAN LAN Service	Secure remote disk mounting with FTP over SSH. Enabled by configuration. Requires option *614-1 FTP SFTP and NFS client*.
SFTP server	22	TCP	WAN LAN Service	Secure File transfer – FTP over SSH. Enabled by configuration.

RobAPI application protocol

RobAPI uses three different protocols:

- NetScan for discovering industrial robot controllers available on the network
- RobAPI network protocol (RNP) for sending commands to and receiving events from robot controllers
- FTP for file transfer (see *FTP on page 230*)

NetScan is an application protocol built on UDP/IP for discovering industrial robot controllers available on the network and for collecting basic data. The data collected by NetScan includes version info and unique id of the software loaded on the controller.

The PC-side implementation of NetScan sends out requests, to which all active controllers respond. Both requests and responses are UDP broadcasts. The requests are sent to UDP port 5512, and the responses are sent to UDP port 5513.

PC-NetScan also supports the option to configure specific "non-local" IP-addresses of controllers, i.e., controllers not on the local subnet. This makes it possible to discover and connect to remote controllers, e.g., over the Internet. The PC-side NetScan sends out UDP unicast requests to port 5514, and the controller responds with UDP unicasts to the source address with the (dynamically allocated) source port.

The RobAPI network protocol (RNP) is an application protocol built on TCP/IP for communicating with industrial robot controllers over a network. The protocol uses by default TCP port 5515. The RobAPI network protocol is event-driven, i.e. RobAPI clients do not need to poll for I/O changes on the controller. Events are sent to the PC-side, whenever a change has occurred. However, only the PC establishes TCP connections.

The RobAPI network protocol has an are-you-alive (AYA) mechanism. With AYA it is possible for the robot controllers to detect unexpected disconnection of clients and for clients to detect unexpected disconnection of robot controllers. AYA messages are sent every 5 second from both the client and the server.

For file transfers, RobAPI uses standard FTP.

FTP

RobAPI Network Protocol uses FTP internally for file transfer. The FTP server of the IRC5 controller supports both active and passive FTP. However, if RobAPI is acting as FTP client, it uses only passive mode.

Continues on next page

FTP is also used independently of RobAPI, for file transfer and file system access to the Main Computer and IPS. This requires logging in with a user name and password that is defined for the controller, see *IRC5 User Authorization System on page 219*.

Note

The controller supports anonymous FTP login on the LAN/Service Port with any user name and password to access files on the controller.

The anonymous access is read only and the access is limited to certain directories located in the RobotWare installation.

Any attempt to read or write to any other location will be denied.

SFTP

The IRC5 robot controller acts as a SFTP server. The use of the server requires UAS grants for FTP Read or Write. Although SFTP could offer several logon options, the IRC5 robot controller only supports logon via user name and password.

SFTP is used internally for file transfer between Robot Studio and Conveyor Tracking Modules.

NFS

The Network File System (NFS) is the de facto standard for file sharing among UNIX hosts and also supported by Microsoft, e.g. Windows Services for UNIX (SFU). The IRC5 robot controller implements an NFS client. The supported NFS version is version 2 as defined in RFC 1094.

Telnet

Telenet is a network protocol based on TCP/IP to provide bi-directional text-oriented communications between a Telnet client and a Telnet server. By default, Telnet uses port 23.

RobICI

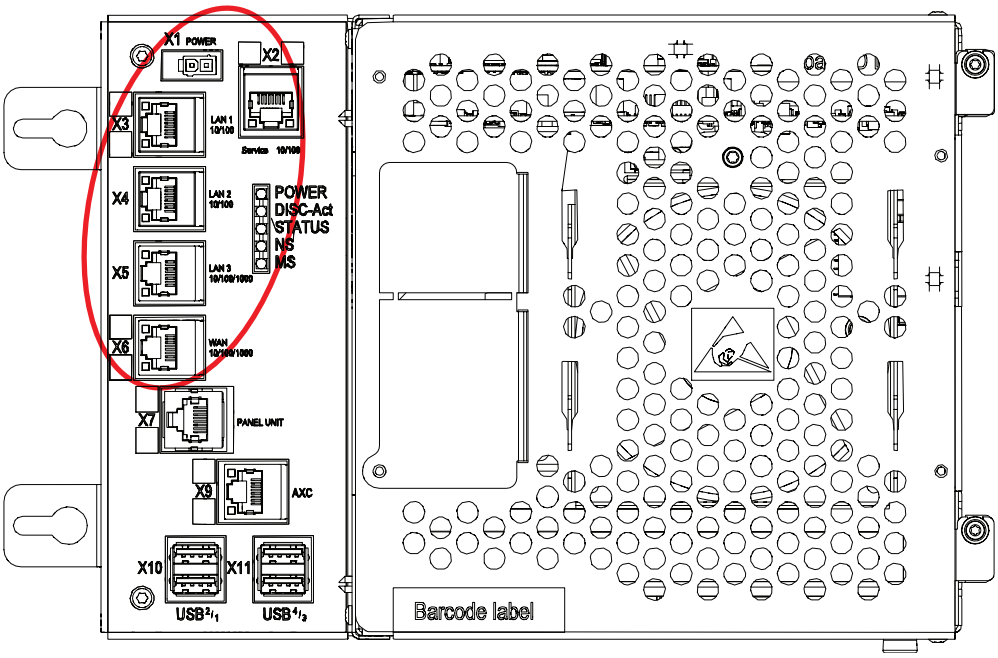
RobICI is an internal ABB application protocol that is used for high speed communication of I/O signals and other data between ABB products, for example, Conveyor Tracking Modules, IRC5 robot controllers and RobotStudio.

9.7 Network connections on the IRC5 main computer

Connections

The I/O network can be connected to one of the the Ethernet ports WAN, LAN 2, or LAN 3 on the main computer.

The following figure illustrates where the Ethernet port connectors, are placed on the main computer.



xx1500000391

Connector	Label	Description
X2	Service	Port to the robot's private network. Intended to be left empty so that service personnel can use it to connect to the computer unit.
X3	LAN 1	Port to the robot's private network. Normally used to connect the FlexPendant.
X4	LAN 2	Port to the robot's private network.
X5	LAN 3	By default LAN 3 is configured for an isolated LAN3 network. Can be reconfigured to be a part of the private network.
X6	WAN	Wide Area Network that can host a public industrial network.

Note

It is not supported to connect multiple ports of the main computer (X2 - X6) to the same external switch, unless static VLAN isolation is applied on the external switch.

Continues on next page

Intended use of WAN and LAN ports

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

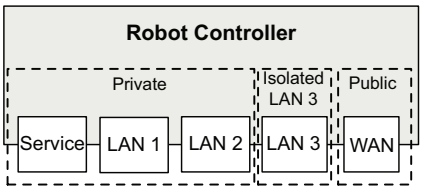
The LAN ports are intended for connecting network based process equipment to the controller, for example industrial networks, cameras, and welding equipment.

LAN 2 can only be used as a private network to the IRC5 controller.

Isolated LAN 3 or LAN 3 as part of the private network

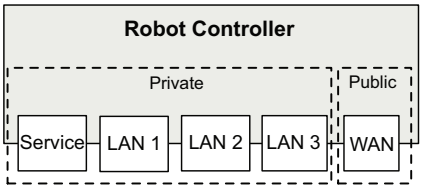
The default configuration is that LAN 3 is configured as an isolated network. This allows several robot controller to be connected to the same network, see *EtherNet/IP on dedicated industrial network on page 236*.

> **Note**
>
> The isolated LAN 3 cannot be used to connect to any HMI device (RobotStudio, Robot Web Services, or PC SDK client) since it does not support the protocol needed for communication.



xx1500000393

An alternative configuration is that LAN 3 is part of the private network. The ports Service, LAN 1, LAN 2, and LAN 3 then belong to the same network and act just as different ports on the same switch. This is configured by changing the system parameter *Interface*, in topic *Communication* and type *Static VLAN*, from "LAN 3" to "LAN". See *Technical reference manual - System parameters*.



xx1500000394

Continues on next page

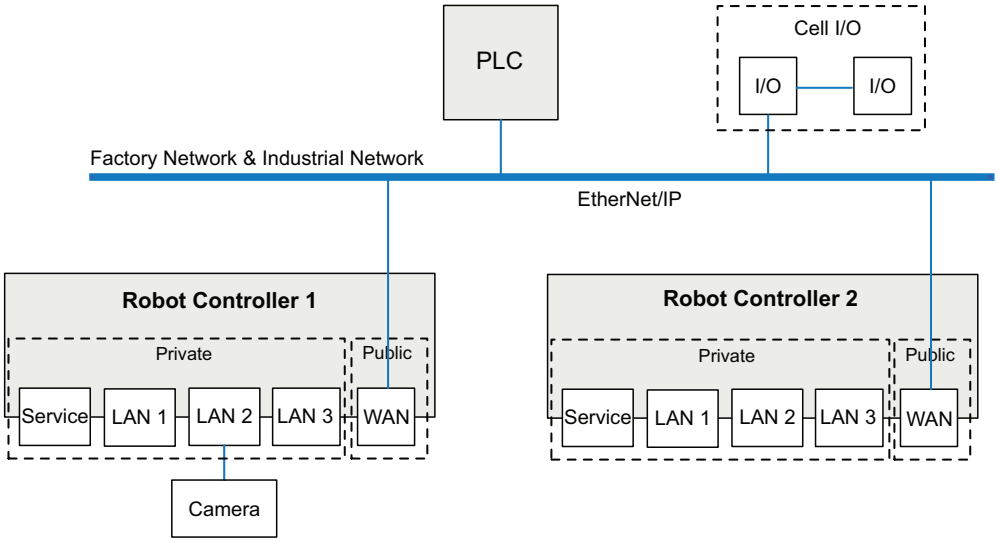
One EtherNet/IP network connected to the robot controller

If EtherNet/IP is used on the public network (WAN port) without an Anybus adapter, EtherNet/IP cannot be used on the private network. Equipment not using EtherNet/IP (for example a camera) can be connected to the private network. To use EtherNet/IP on both the public and private network, an Anybus adapter must be used. See *Using Anybus adapter to connect two EtherNet/IP networks on page 237*.

EtherNet/IP on factory network

When the WAN port is used for connecting to an industrial network, the traffic shares the same media as the factory network and will share bandwith with other non industrial network traffic.

The following figure illustrates the network when connecting a scanner and an adapter to the WAN port of the main computer:



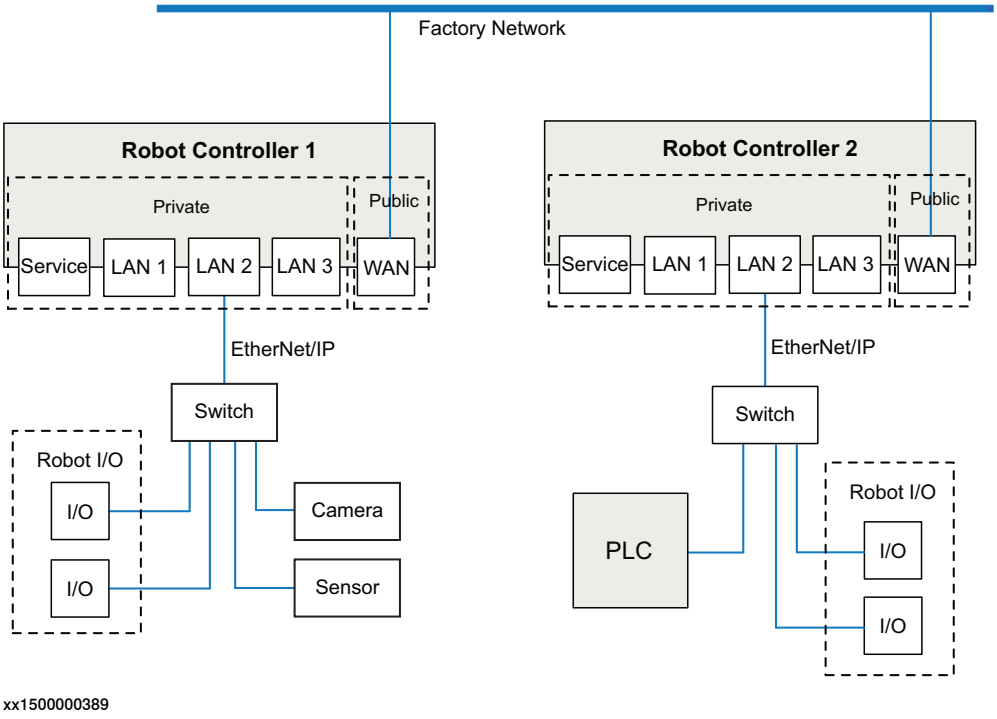
xx1500000387

Continues on next page

EtherNet/IP on private network

The private network can contain I/O, sensors, etc. for the robot controller. However, it is not possible to connect several robot controllers to the same private network.

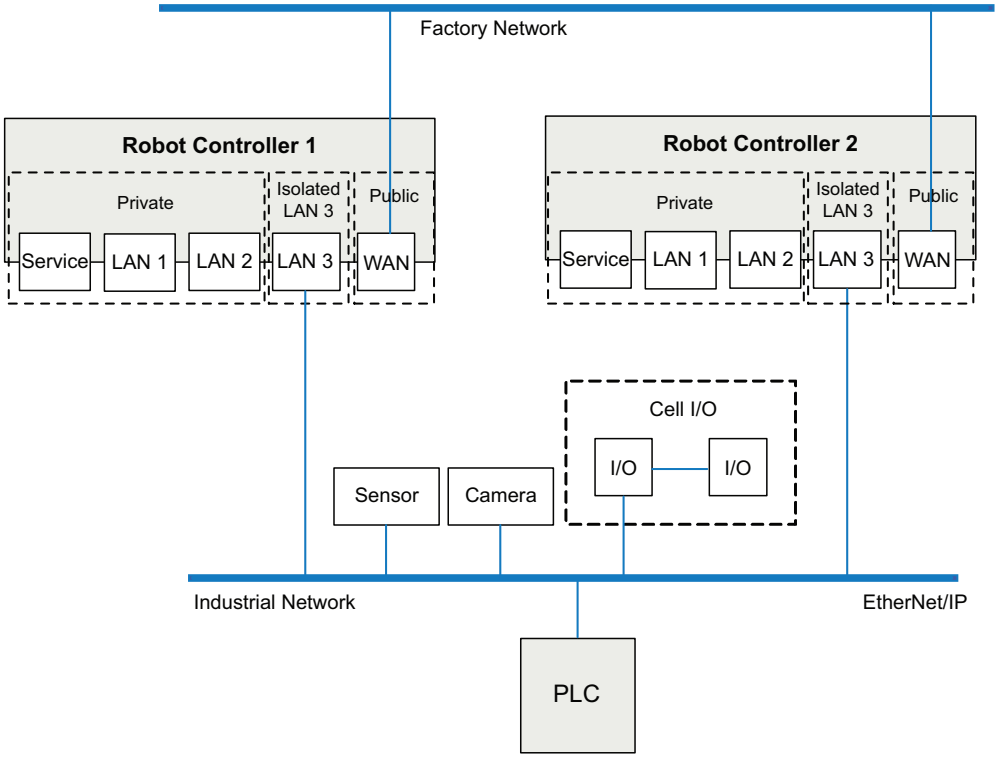
The following illustration shows two robot controllers with EtherNet/IP (and other IP traffic) on each private network. The factory network cannot communicate with the robot controller using EtherNet/IP.



Continues on next page

EtherNet/IP on dedicated industrial network

By connecting to the isolated LAN 3 port it is possible to connect several robot controllers to a dedicated industrial network.



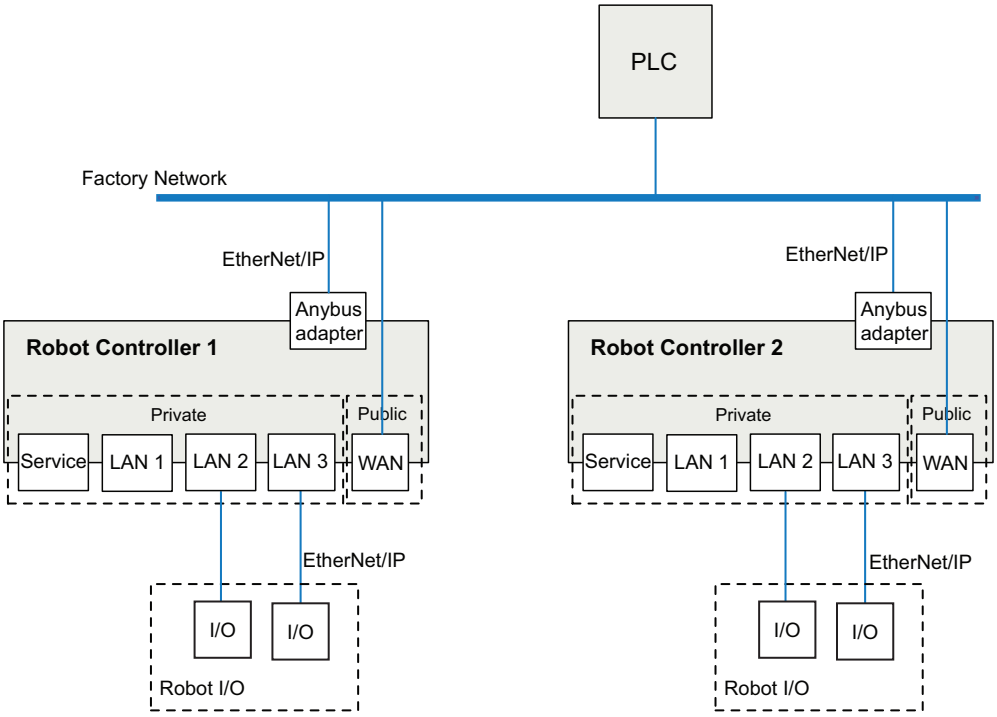
xx1500000388

Continues on next page

Using Anybus adapter to connect two EtherNet/IP networks

EtherNet/IP on shared factory network and private network

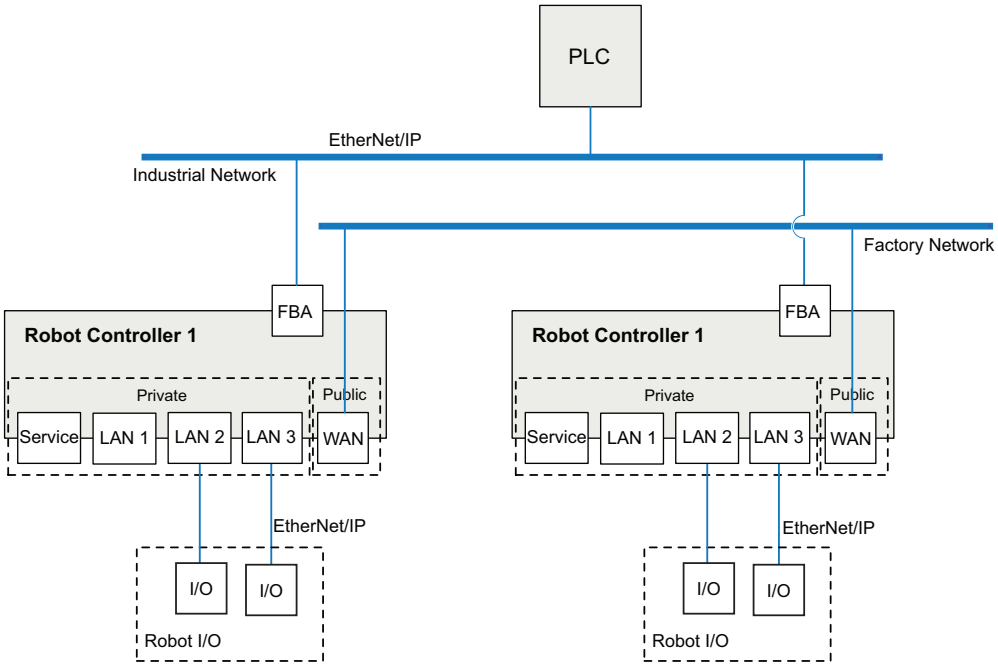
To be able to use EtherNet/IP on both the public and the private network, an Anybus adapter must be used. If the same factory network is used both for EtherNet/IP communication and other communication, both the Anybus adapter and the WAN port must be connected to the factory network. For information about the EtherNet/IP Anybus adapter, see *Application manual - EtherNet/IP Anybus Adapter*.



Continues on next page

EtherNet/IP on dedicated industrial network

If the EtherNet/IP communication shall be separated from other Ethernet communication, an Anybus adapter must be installed and connected to the public EtherNet/IP industrial network and the WAN port connected to the factory network. For information about the EtherNet/IP Anybus adapter, see *Application manual - EtherNet/IP Anybus Adapter*.



xx1500000392

Index

3
3rd party software, 14

A
ABB menu, 24
Absolute Accuracy
  data storage, 191
account management, 221
add-ins, 49
additional axes
  activating, 99
  deactivating, 99
anti-virus, 222
application protocols, 228
approach points, 71
ArcLink, 228

B
backup, 223
  default file path, 115
  directory, 135
  important, 135
  system, 133
  system parameters, 143
backward button, 21
boot application, 122
Boot Application
  restart, 117
  settings, 118
  using, 117
buttons
  controller, 32

C
cabinets, 18
calculation result, 73
calibration, 30
  4 points XZ, 194
  base frame, 194
  loading data, 189
  motor calibration offset, 190
  status, 188
clean disk, 184
close button, 24
communication protocols, 228
Connected Services, 228
connection
  FlexPendant, 36
  to network, 41
connections, 232
connector, 20
controller
  buttons, 32
  ports, 32
  single cabinet, 18
control tools, overview, 29
cyber security, 213

D
data instance, 65
data types
  changing type, 94
  creating new, 65
declarations
  hiding, 95
default paths
  setting, 115
Default User, 219
DHCP client, 228
DHCP server, 41, 228
disable backup, 134
displacements
  work object, 80
disposal, 224
DNS client, 228

E
EGM, 228
elongator points
  define, 73
e-mail, 222
emergency stop button
  FlexPendant, 20
enabling device, 20, 22
entry routine, 53
erase SD-card, 184
EtherNet/IP I/O, 229
EtherNet/IP messaging, 229
expressions
  editing, 92

F
files
  handling, 113
  programs, 54
firewall, 223
firmware
  reflashing, 126
flash disk drive, 112
FlexPendant
  connecting, 36
  connecting in operation, 38
  connector, 32
  disconnecting, 38
  hardware buttons, 21
  hot plug, 38
  how to hold, 23
  jumper plug, 38
  left-hander, 23
  main parts, 20
  overview, 19
  reflashing, 126
  screen, 23
forward button, 21
FTP, 228, 230
FTP client, 229

H
hard buttons, 21
hard disk drive, 112
hold-to-run, 22
hot plug, 38

I
I/O
  activating, 104
  deactivating, 104
  safety signals, 107
  units, 104
instances
  data types, 65

system parameters, 141
instructions
data types and declarations, 93
expressions, 92
IP address
fixed, 42
obtain automatically, 42
use no address, 41
IRC5 controller, 18
overview, 18
IRC5P, 216

J
joystick, 20
using, 21
jumper plug, 38

L
LAN unit, 112
licenses, 14

M
main computer, 232
main module, 53
Main routine, 53
main switch
controller, 32
mastership, 219
mechanical units
activating, 99
deactivating, 99
memory, 112
memory, what is?, 112
mirroring, 90
mirror plane, 207
routines, 207
what is, 206
modules
creating, 57
deleting, 59
handling of, 57
loading, 58
mirroring, 91
renaming, 58
saving, 58

N
NetScan, 228, 230
network architecture, 215
network connections
setting, 118
setting up, 41
network security, 13
NFS, 231
NFS client, 229
NFS server, 229

O
open source software, OSS, 14
operator window, 24
options
duty time counter, 32
hot plug button, 32, 38
IRC5 Compact, 18
IRC5 Panel Mounted Controller, 18
RobotWare add-ins, 49
safety chain LEDs, 32
service outlet, 32
service port, 32

P
paint application, 216
passwords, 222
payloads
creating, 86
declarations, 87
display definitions, 88
editing, 88
physical security, 221
PickMaster, 216
policy, 220
ports, 232
controller, 32
private network, 235
PROFINET CM, 229
PROFINET RPC, 229
PROFINET RT, 229
PROFINET RTM, 229
program a robot, 30
program directory, 54
program execution start button, 21
program memory, 96
program module, 53
programs
about files, 54
creating, 54
default file path, 115
deleting, 96, 98
handling of, 54
loading, 55
mirroring, 91
renaming, 56
saving, 55
protocols, 228

Q
queue backup, 134
quickset menu, 24

R
RAPID, structure, 52
RAPID application, 52
reflashing
axis computer, 126
contactor interface board, 126
drive unit, 126
firmware, 126
FlexPendant, 126
panel board, 126
profibus, 126
remote access, 225
remote client, 225
reset button
location, 20
using, 21
reset RAPID, 123
reset system, 124
restart, 121
controller, 119
overview, 116
reset RAPID, 123
reset system, 124
revert to last auto saved, 125
start boot application, 122

system, 119
restore
default file path, 115
system, 137
revert to last auto saved, 125
RNP, 228
RobAPI, 226, 230
RobICI, 229, 231
robots
activating, 99
deactivating, 99
RobotStudio
overview, 28
RobotStudio Oline Apps, 26
RobotStudio Online Apps
Calibrate, 26
Jog, 26
Manage, 26
Operate, 27
Tune, 27
YuMi, 27
RobotWare add-ins, 49
Robot Web Service, 228
Robot Web Services, 226
Routine, 53
routines
changing declarations, 63
copying, 63
creating, 60
defining parameters, 61
deleting, 63
handling of, 60
mirroring, 90
run button, 21

S
safety I/O signals, 107
serial numbers
data storage, 191
service port, 32
SFTP, 231
SFTP client, 230
SFTP server, 230
SIS, Service Information System
data storage, 191
SMB, 191
software licenses, 14
start boot application, 122
start button, 21
start up, 44
status bar, 24
step backward button, 21
step forward button, 21
stop button, 21
stylus pen
location, 20
using, 21
system
backup, 133
installing, 117
loaded, 111
overview, 111
renaming, 119
restarting, 119
restore, 137
return to default settings, 124
revert to last auto saved, 125
selecting, 119
start and select, 122
starting, 119
starting without software, 117
system, upgrading, 48
system module, 53
system parameters, 30
configuring, 141
instances, 141
saving, 143

T
task bar, 24
tasks, 52
Telnet, 231
three-position enabling device, 20, 22
thumb button
using, 22
tool, overview control tools, 29
tool center point
about, 67
calculation result, 73
define, 72
defining, 73
measuring, 75
TCP, 67
working area variations, 73
tool frame
defining, 70
methods, 70
reorientation test, 73
tool orientation, 73
tools
creating, 67
editing definitions, 75
editing tool data, 74
make stationary, 77
setting up tool coordinate system, 78
stationary, 77
Total Load, 86
touchscreen, 23

U
UAS, 219, 227
units
activating, 104
deactivating, 104
USB, 113
USB port
FlexPendant, 20
User Authentication System, 227
User Authorization System, 219

V
virus protection, 222

W
work objects
creating, 79
declarations, 79
defining coordinate system, 80
displacements, 80
editing declarations, 85
editing work objects data, 84
write access, 219

ABB AB
Robotics & Discrete Automation
S-721 68 VÄSTERÅS, Sweden
Telephone +46 10-732 50 00

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

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

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

abb.com/robotics

3HAC050940-001, Rev P, en