Rockwell Automation Application Content

Rockwell Automation Robotics Libraries

Reference Manual

HMI Application - Robot

Robot rOS v2.0

Rockwell Automation

December, 2023

Important User Information

Solid-state equipment has operational characteristics differing from those of electromechanical equipment. Safety Guidelines for the Application, Installation and Maintenance of Solid State Controls (publication SGI-1.1 available from your local Rockwell Automation sales office or online at http://literature.rockwellautomation.com) describes some important differences between solid-state equipment and hard-wired electromechanical devices. Because of this difference, and because of the wide variety of uses for solid-state equipment, all persons responsible for applying this equipment must satisfy themselves that each intended application of this equipment is acceptable.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of Rockwell Automation, Inc., is prohibited.

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.

WARNING 	Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.
IMPORTANT	Identifies information that is critical for successful application and understanding of the product.
ATTENTION 	Identifies information about practices or circumstances or death, property damage, or economic loss. Attentions avoid a hazard, and recognize the consequence.
SHOCK HAZARD 	Labels may be on or inside the equipment, that dangerous voltage may be present.
BURN HAZARD 	Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.

Table of Contents

1 Overview

This reference manual describes the overall structure and usability of the preconfigured FactoryTalk View ME application shipped along with Rockwell Automation Robotics Libraries v2.0.

Use when:

- Using a Device Handler for Robot Management.
- Testing Rockwell Automation Robotics Libraries.
- Developing a standardized FactoryTalk View ME application for robot control with noncomplex requirements.
- Controlling supported robot models and geometries.
- Load Path Add-On Instruction functionality is sufficient for programming robot moves.

Do NOT use when:

- Not using a Device Handler for Robot Management.
- Developing a custom FactoryTalk View ME application for robot control with complex requirements.
- Controlling unsupported robot models and geometries.
- A vendor-specific robot programming language or programming environment is required.

1.1 Prerequisites

- FactoryTalk View ME Station
  - v12.00.00 →
- FactoryTalk View Studio ME
  - v13.00.00 →
- Device Handler for Robot
  - Rockwell Automation Robotics Libraries v2.0 →
- Studio 5000 – Logix Designer
  - v35.0 →
- Studio 5000 – Application Code Manager
  - v4.03.00 →

1.2 Acronyms and Abbreviations

Name	Description
AOI	Add-On Instruction
CP-L	Continuous Path Linear
CP-W	Continuous Path Wrist
HMI	Human Machine Interface
PLC	Programmable Logic Controller
PTP	Point-to-Point
TCP	Tool Center Point
UI	User Interface
w.r.t.	with respect to

2 Safety

ATTENTION

Perform a risk assessment to make sure that all task and hazard combinations have been identified and addressed. The risk assessment can require additional circuitry to reduce the risk to a tolerable level. Safety circuits must consider safety distance calculations, which are not part of the scope of this document.

It is recommended that a formal Risk Assessment be performed on the robot as well as on the entire robotic cell during the design phase of the robotic application. It is also recommended to comply with ISO 10218-1 and 10218-2, "Robots and robotic devices – Safety requirements for industrial robots". (ANSI RIA R15.6 and CSA Z434-14 are national adoptions of ISO 10218.)

IMPORTANT

Before operating any robot using the HMI Application, it is highly recommended that the user connects the E-stop and Enable Pendant wiring from the 2711T terminal to Safety I/O and program their proper function in the Safety Task.

All MobileView 2711T terminals come equipped with an integrated safety-rated E-stop push button and a 3-position enable switch. These are not programmed or activated in the Rockwell Automation Robotics Libraries. It is highly recommended that the user connects these devices to Safety I/O and programs and validates their functions in the Safety Task prior to operating any robot using the HMI application.

Rockwell Automation offers a library of Safety Function Application Techniques that show detailed examples of how to implement and validate many common Safety Functions. Two specific Safety Function Application Techniques that apply to the MobileView 2711T are:

- SAFETY-AT125 "Enabling Switch with a Safety Controller and a POINT Guard I/O Module"
- SAFETY-AT036 "Emergency Stop Products: Safety Controller Connected to Dual-channel E-stops".

3 Basic Concepts

This Chapter has definitions for Robotics terms/concepts laying the foundations for this document and Rockwell Automation Robotics Libraries.

3.1 Robot System

A robot system is a cohesive group of hardware and software subsystems working in an integrated manner to perform robotic tasks. It usually directly maps to a single robot cell.

Within the context of Rockwell Automation Robotics Libraries, a robot system consists of one or more Rockwell Automation pendants, one or more Rockwell Automation robot controllers, and one or more Rockwell Automation Partner robots. Figure 1 depicts a possible configuration for such a system.

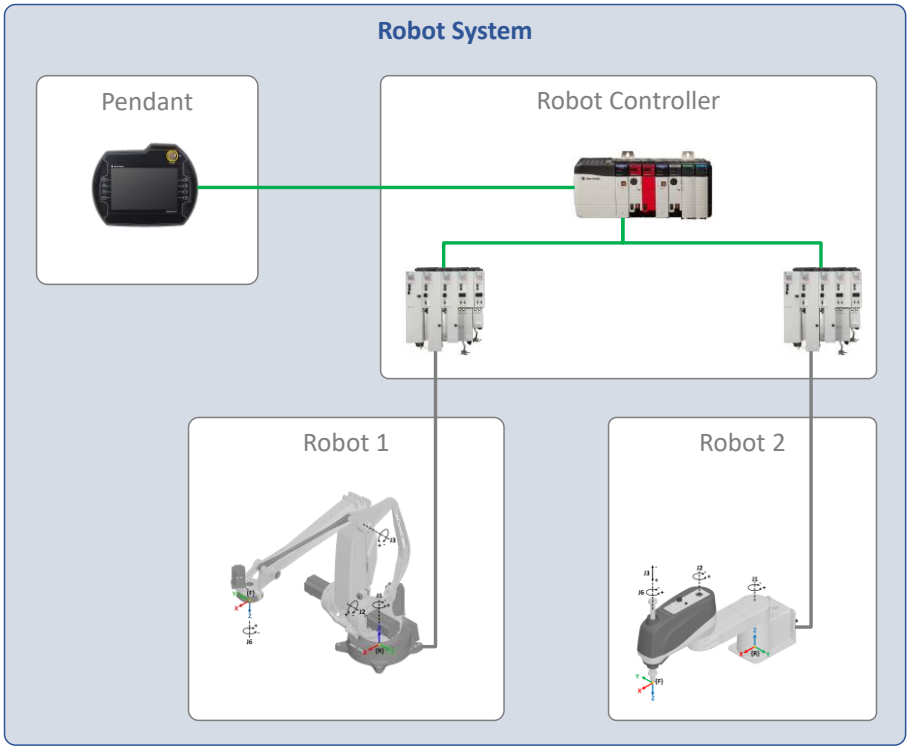


Figure 1- Example of a Rockwell Automation robot system.

3.2 Robot Controller

A robot controller is a collection of hardware and software modules controlling the motion of one or more robots.

Rockwell Automation implements an integrated robot controller with GuardLogix or Compact GuardLogix PLCs running Rockwell Automation Robotics Libraries content and Kinetix drives connected to one of more Partner robots.

3.3 Robot

A robot is an actuated mechanism composed of joints, actuators, links, and a tool. The following Rockwell Automation Partner robots are currently supported by the framework:

Manufacturer	Model	Geometry
Comau	NJ-40-2.5	Articulated Independent 6-Axis
Comau	NJ-60-2.2	Articulated Independent 6-Axis
Comau	Racer-3-0.63	Articulated Independent 6-Axis
Comau	Racer-5-0.63	Articulated Independent 6-Axis
Comau	Racer-5-0.80	Articulated Independent 6-Axis
Comau	Racer-7-1.4	Articulated Independent 6-Axis
Comau	PAL-180-3.1	Articulated Dependent 4-Axis
Comau	PAL-260-3.1	Articulated Dependent 4-Axis
Comau	Rebel-S6-0.45	SCARA Independent 4-Axis
Comau	Rebel-S6-0.60	SCARA Independent 4-Axis
Comau	Rebel-S6-0.60c	SCARA Independent 4-Axis, Ceiling mounted
Comau	Rebel-S6-0.75	SCARA Independent 4-Axis
Comau	Rebel-S6-0.75c	SCARA Independent 4-Axis, Ceiling mounted
Generic		Articulated Independent 6-Axis
Generic		Articulated Dependent 4-Axis
Generic		SCARA Independent 4-Axis
Generic		Delta 3-Axis
Generic		Delta 4-Axis
Generic		Delta 5-Axis

3.4 Pendant

The pendant, hereafter used interchangeably with the term HMI (Human Machine Interface), is a hand-held unit connected to the robot controller with which a robot can be monitored, programmed, and moved. The HMI application usability is the main subject of this document.

It is assumed that the HMI hardware is a MobileView 2711T terminal running the preconfigured FactoryTalk View ME application shipped with Rockwell Automation Robotics Libraries. We recommend using Second Generation MobileView terminals for best performance:

- 2711T-B1OI1N1
- 2711T-B1OI1N1-TC

The information in this manual is also worthwhile for users running the FactoryTalk View ME application on a Windows PC running FactoryTalk View ME Station. They should get similar usability experience, except for the hardware-specific functions provided by the MobileView terminal, e.g., the e-stop button and enabling switch.

3.5 Cartesian Space and Joint Space

Engineering disciplines often require approaching problems from different points of view. For instance, in Signal Processing, signals can be analyzed in both time and frequency domains. These domains are not exclusive – they complement each other.

Similarly, the Robotics discipline requires seeing robots from complementary domains. For example, Joint space involves the description of moves and state from the perspective of the robot joints; Cartesian space involves descriptions from the perspective of the 3D space the robot lives in. Robot software takes advantage of this duality in many control and monitoring functions – Joint/Cartesian jogging, forward/inverse kinematics calculations, Point-to-Point/Cartesian-Linear moves, etc.

3.6 Positions, Orientations, Poses, Frames, and Coordinate Systems

Points and vectors are handy mathematical entities for measuring the relative **position** between objects. However, position alone is insufficient to fully describe the configuration of a freely moving body in 3-D space. For instance, the body may be arbitrarily rotated around its center of mass while keeping the exact same position relative to a reference. This missing rotational aspect of the body is called **orientation**.

In summary, to fully define the location of a free body in three dimensional space, we need at least six independent coordinates: three for position and three for orientation. The term **pose** is often used in Robotics as a shorthand for this combination of position and orientation.

Poses only have meaning when expressed with respect to a reference **frame**. A frame is a geometric entity including a point (origin) and a set of three orthogonal unit vectors (axes). When frames are affixed to bodies in the robot and in the environment, they can be used for setting up spatial relationships, including the pose of robot parts, workpieces, and even desired move targets. Frames can be translated and rotated (transformed) just like points can be translated. Frames can also serve as a basis for Cartesian **coordinate systems**, meaning that the pose of a body can be described by a set of coordinates measured as translations and rotations relative to the frame axes.

The following Cartesian frames / coordinate systems are currently supported by the Rockwell Automation Robotics Libraries:

- World
- Robot
- Flange
- Tool
- User

Figure 2 presents a sample robot system with two robots and all the currently supported frames.

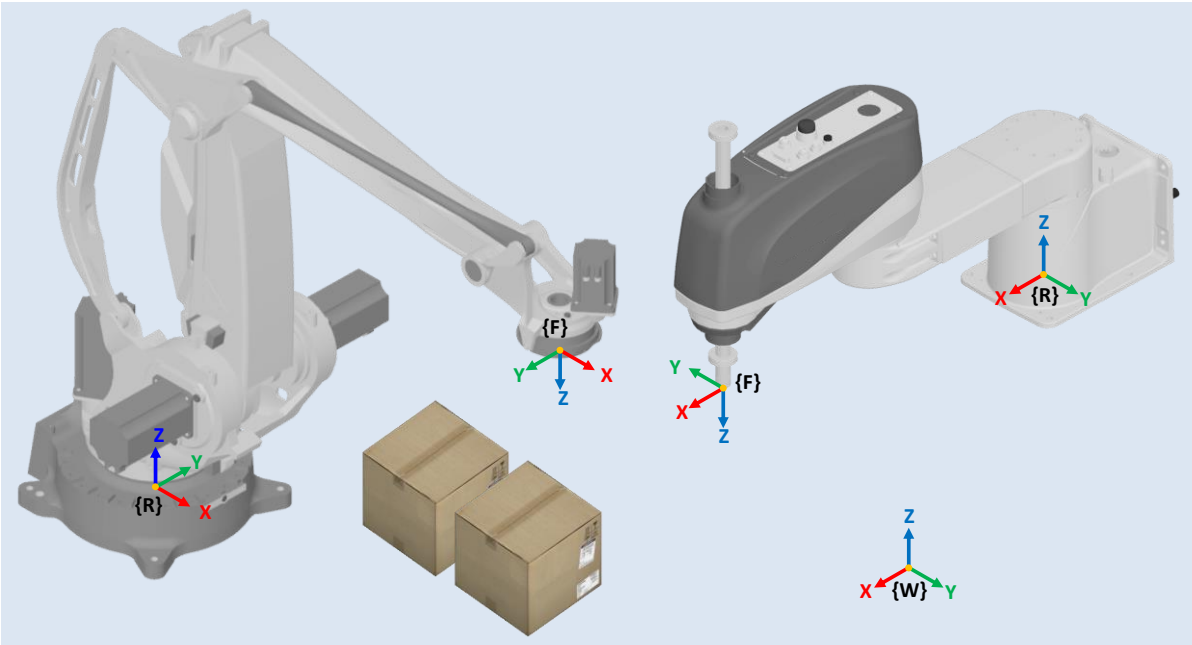


Figure 2- Supported frames in a sample two-robot system.

The **World** frame {W} is a permanently defined frame and serves as an inertial global reference / coordinate system for all the components of the robot system. It is useful for various operations, especially when there is more than one robot in the cell.

The **Robot** frame {R} is a frame located at the robot's base. It defines the pose of the robot base relative to **World** frame and is usually the easiest reference for simply moving the robot to another location. There is one **Robot** frame per controlled robot. By default, the **Robot** frames coincide with the **World** frame.

The **Flange** frame {F} is a frame located at the robot's flange center point. It defines the pose of the robot flange relative to **Robot** frame. There is one **Flange** frame per controlled robot.

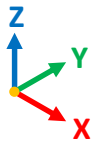
The **Tool** frame {T} is a frame located at the tip of the tool relative to the flange. There can be multiple **Tool** frames per controlled robot.

The **User** frame {U} is a customizable coordinate system that you define within the robot's base frame. It acts as a local reference point for programming robot movements and defining positions within a specific area of the robot's workspace. There can be multiple **User** frames per controlled robot.

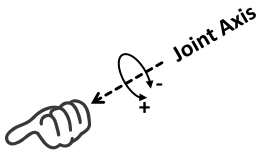
3.7 Conventions

The Rockwell Automation Robotics Libraries enforce the following conventions by design:

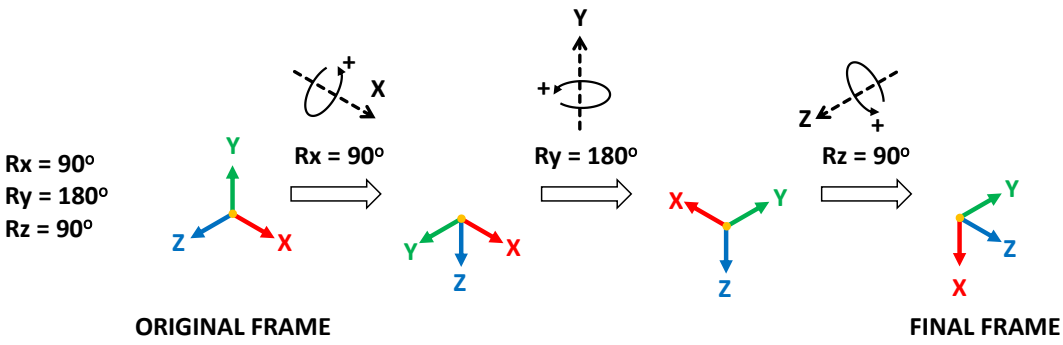
- Frames are right-handed. That means if the frame X-axis points to the right and the Y-axis "enters" this page, the Z-axis will point up:



- The positive direction for rotations obeys the right-hand rule:



- Orientation is represented by XYZ fixed-frame Euler angles, so the three angles (Rx, Ry, Rz) reflect a specific sequence:
    1. A rotation of Rx around the original frame X-axis, followed by
    2. A rotation of Ry around the original frame Y-axis, followed by
    3. A rotation of Rz around the original frame Z-axis.



4 rOS Program

4.1 Output Data

Output	Function / Description	DataType
PathPoints	Trajectory Target Point array	raM_UDT_Robot_Opr_PathPoint[30]
HMI_WorkZoneSts	rOS Configured Zone Status array	DINT[17]

5 Application Code Manager

5.1 Object: raM_Robot_rOS_HMI

Parameter Name	Default Value	Instance Name	Description
Program_Name	{ObjectName}	-	Program Name to be instantiated
GlobalDisplayTagName	MobileView1_Sts_CurrentDisplayNo	{ GlobalDisplayTagName }	

Configured HMI Content

HMI Content	Instance Name	Description
Launch Button	{ObjectName}_GrpName	Global Object configured callout instance

5.2 Attachments

Name	Description	File Name	Extraction path
V2_{LibraryName}	Reference Manual	RM-{LibraryName}.pdf	{ProjectName}\Documentation
V2_{LibraryName}	MobileView Application	RoboticsMobileView_v2_0.apa	{ProjectName}\Visualization\FTViewME\

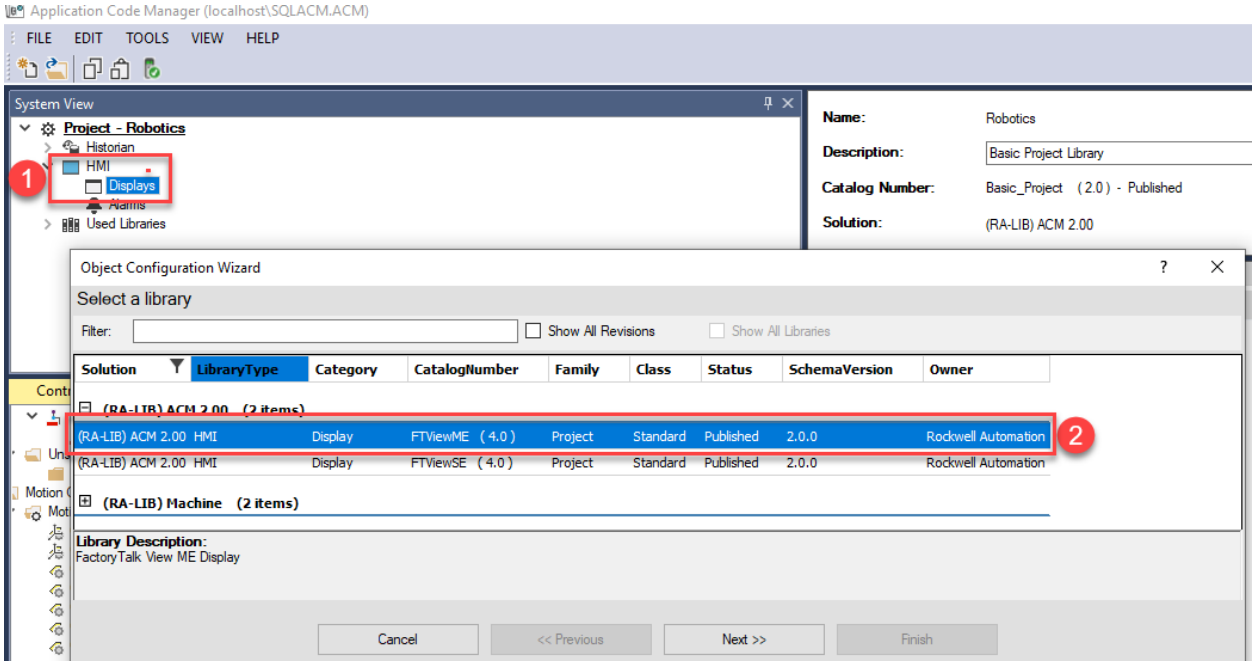
6 rOS Implementation

Robotics applications using MobileView pendant for viewing and controlling robots msut contain both a Robot Device Handler and a Robot OS program for each robot instance. ACM Implementation of raM_Robot_rOS_HMI library creates rOS program for each robot, links it to respective Robot Device Handler program

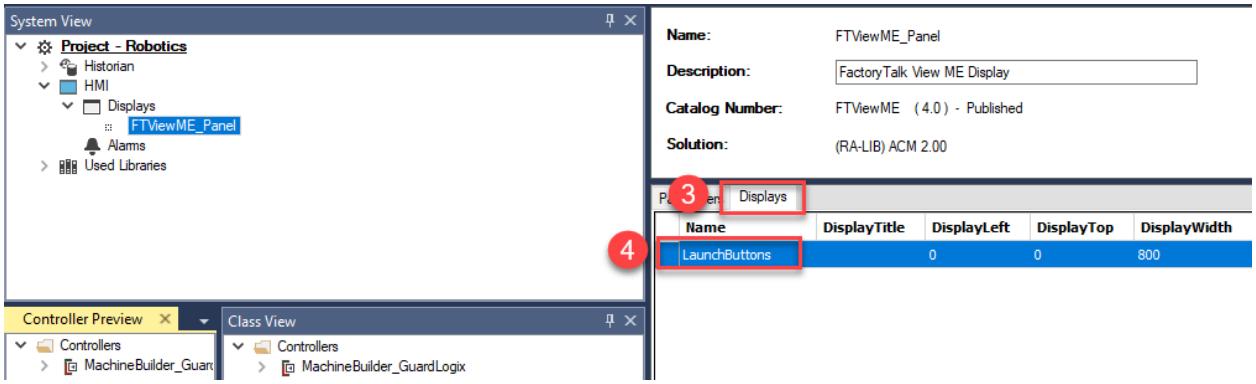
6.1 Adding Launch Buttons Display

Create a display in ACM project that will contain configured parametrized Robot Launch Buttons.

1. In ACM System View expand HMI, right-click on Displays and Add ACM 2.0 -> FTViewME display
2. Select ACM FTViewME Display



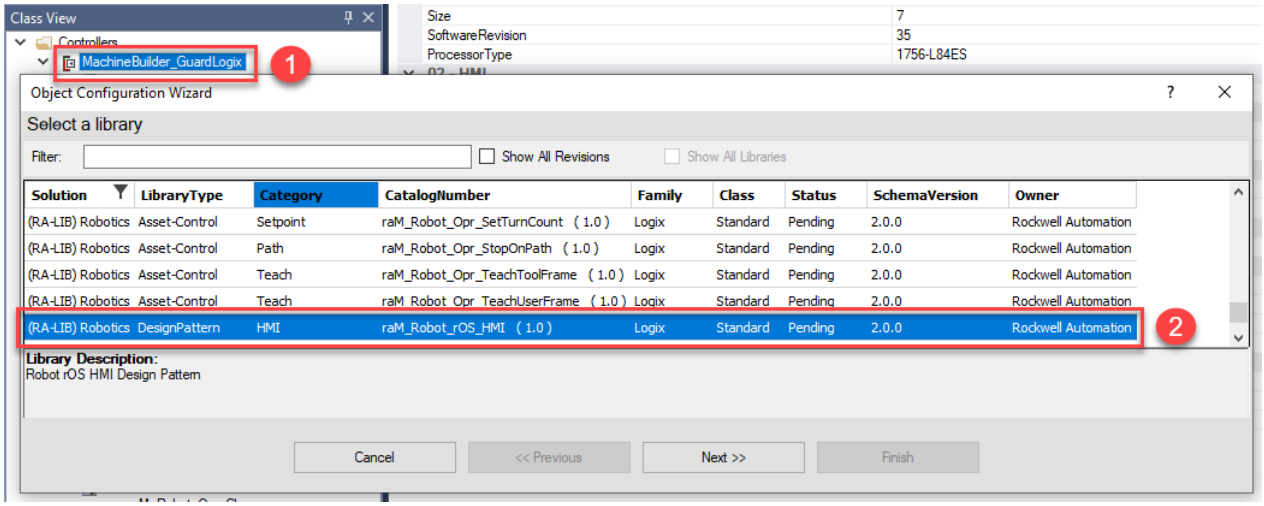
3. In Property View click on Displays Tab
4. Right-click on Name column to open drop-down then click on Add New display then name it LaunchButtons



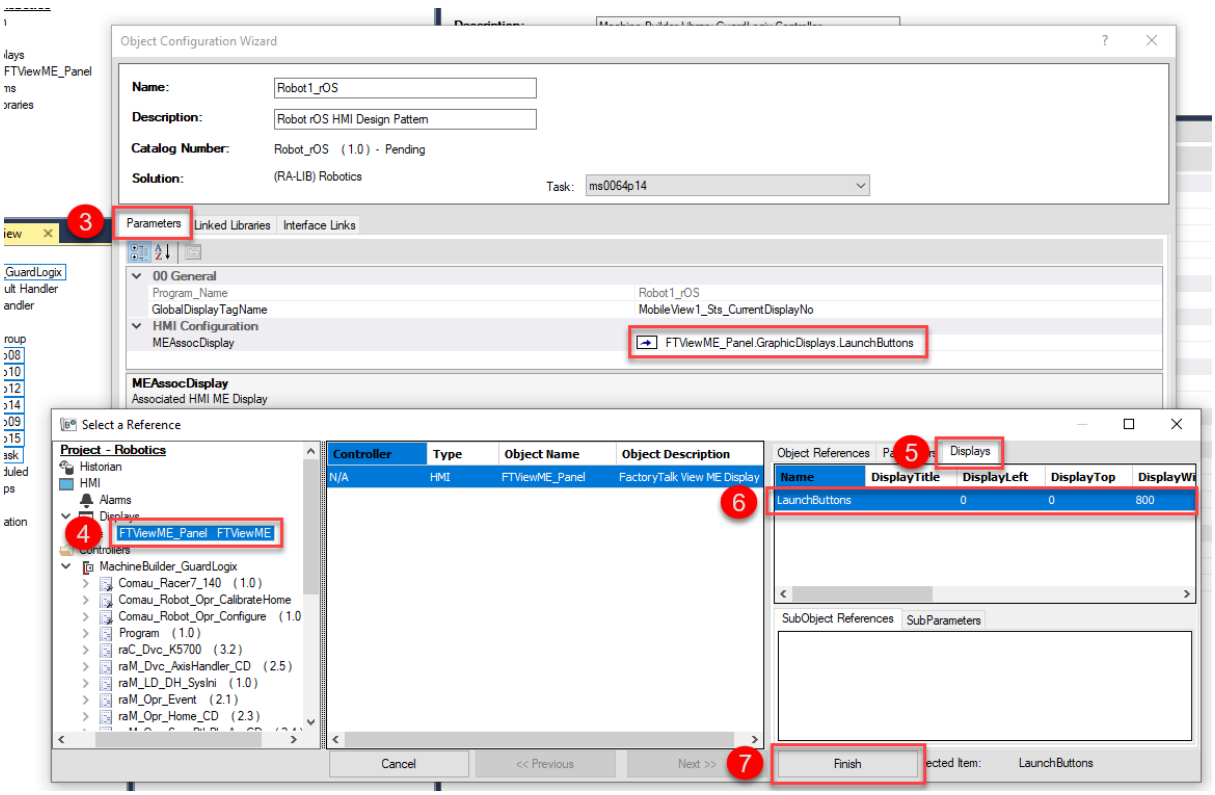
6.2 Adding raM_Robot_rOS_HMI Library

Add raM_Robot_rOS_HMI Library into Controller, connect it to Robot Handler

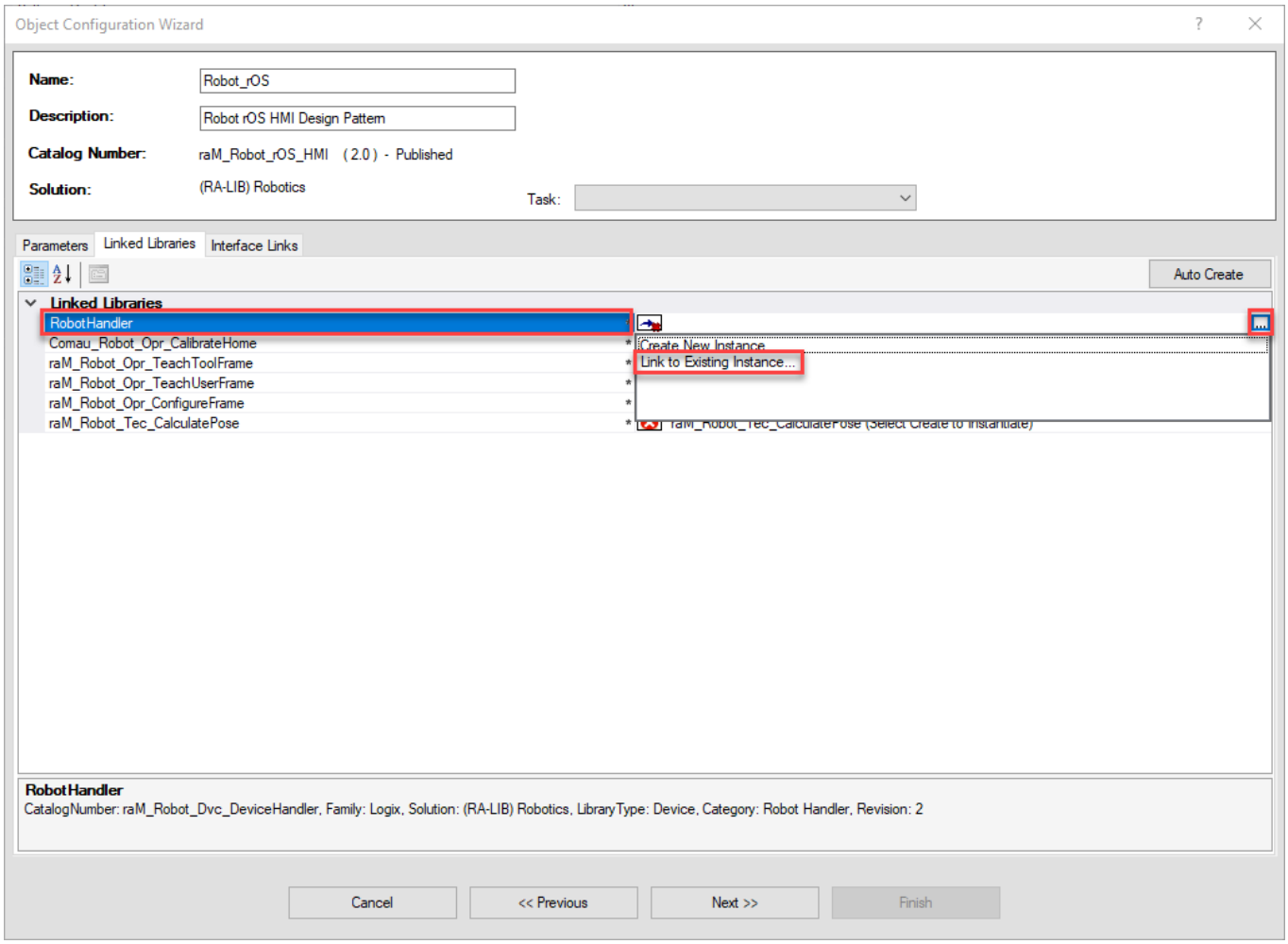
1. In ACM Class View, right-click on the controller to Add New
2. Select raM_Robot_rOS_HMI library object



3. In Parameters tab, click on MEAssocDisplay drop-down, then click on ellipsis button on the right to open Select a Reference popup
4. In Select Reference popup left window pane click on FTViewME_Panel
5. In same popup, right window pane, select Displays tab
6. Select the display that will hold Launch Buttons
7. Click Finish to close the popup window. This will populate the MEAssocDisplay configuration



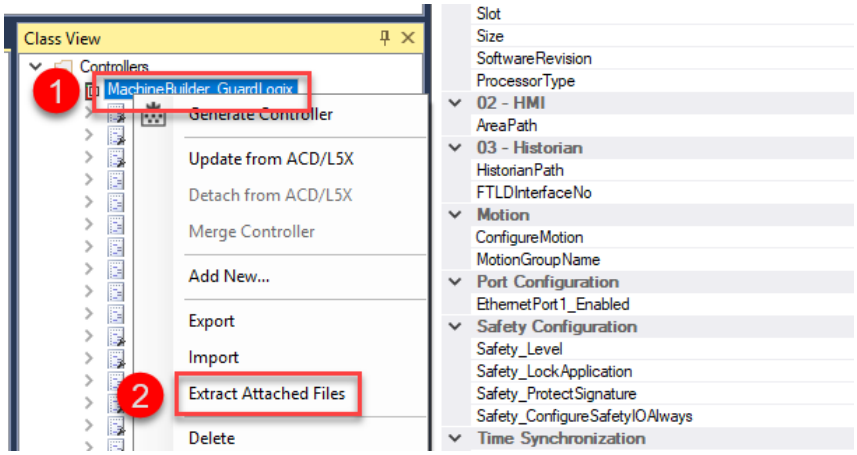
8. Click on Linked Libraries tab
9. Select a Robot Handler instance (new or previously configured)
10. Click on Auto Create to link dependent libraries
11. Click on Finish to complete configuration of raM_Robot_rOS_HMI library



6.3 Extracting Attachments

Extract attachments included with raM_Robot_rOS_HMI library

1. Right-click on Controller to open drop-down
2. Select Extract Attached Files

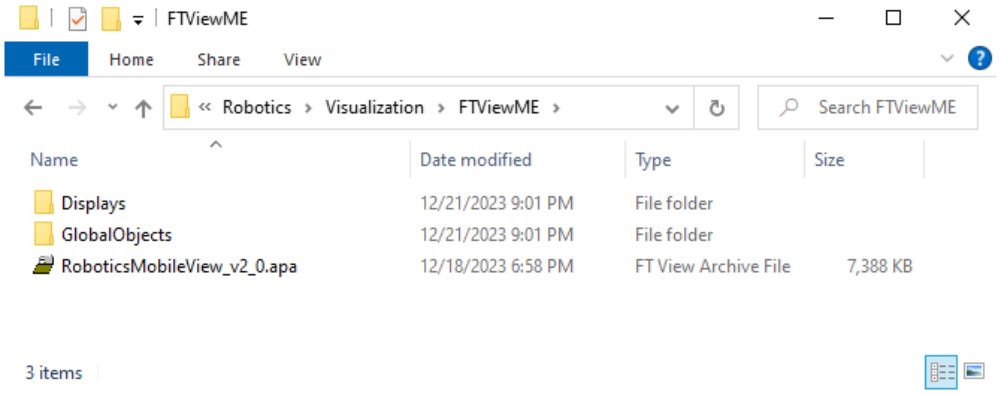


7 MobileView Application Configuration

Robotics MobileView application is a FactoryTalk View ME archive file that is attached to Robot_rOS library. The archive file is extracted as an attached file in the ACM project. The archived file needs to be restored as a FactoryTalk View project. In the restored project user must configure communication with controller, create launch buttons connected to each robot instance, and create runtime application file for MobileView.

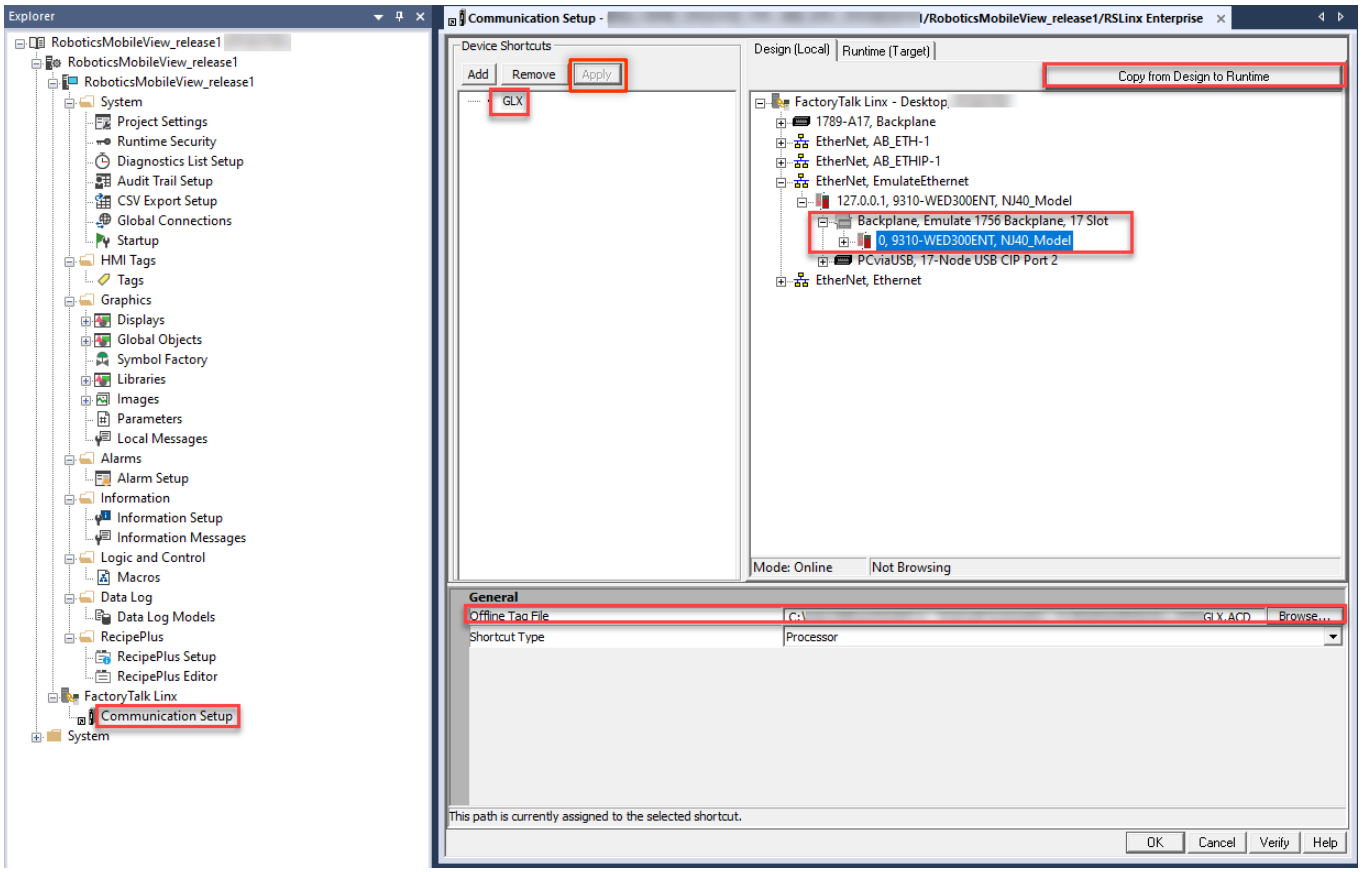
7.1 Restoring Robotics MobileView application

Restore Robotics MobileView application using RoboticsMobileView_v1_0.apa archive file from {Extract Folder}\Visualization\FTViewME\ folder generated by ACM. Double-click on RoboticsMobileView_v2_0 archive file to restore it.



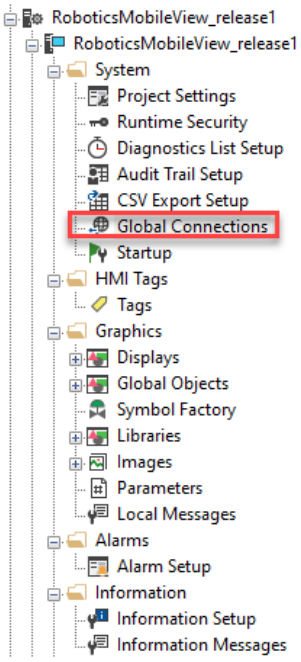
7.2 Configuring Communications

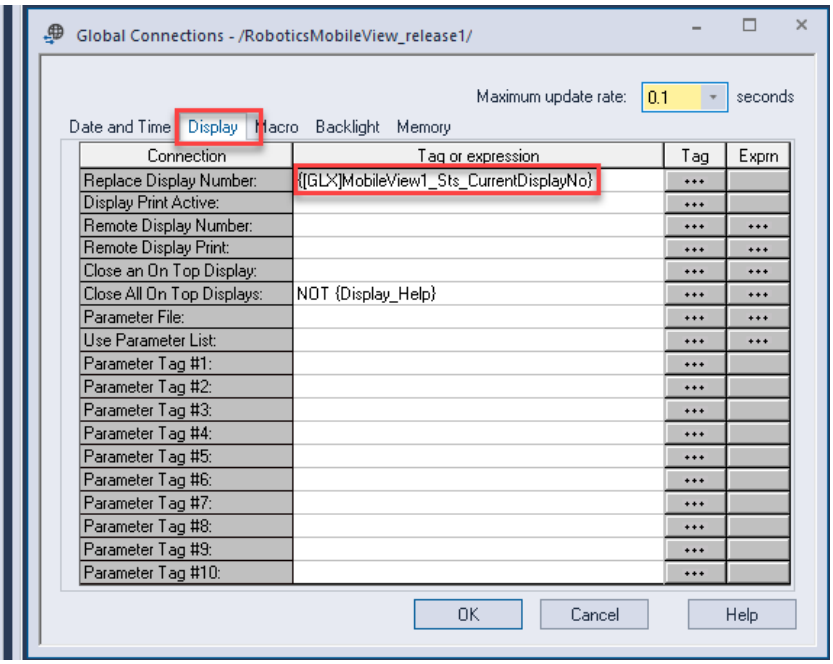
In FactoryTalk View ME, go to the **Explorer** window, and double-click the node **FactoryTalk Linx > Communication Setup**. Add a new shortcut matching the name assigned in ACM. Connect the shortcut to the controller by highlighting the controller and clicking **Apply**. (You may also optionally reference the **Offline Tag File** by browsing to the .ACD file created earlier and clicking **Apply**.) Click **Copy from Design to Runtime**, and then click **OK**.



7.3 Configuring Global Connections

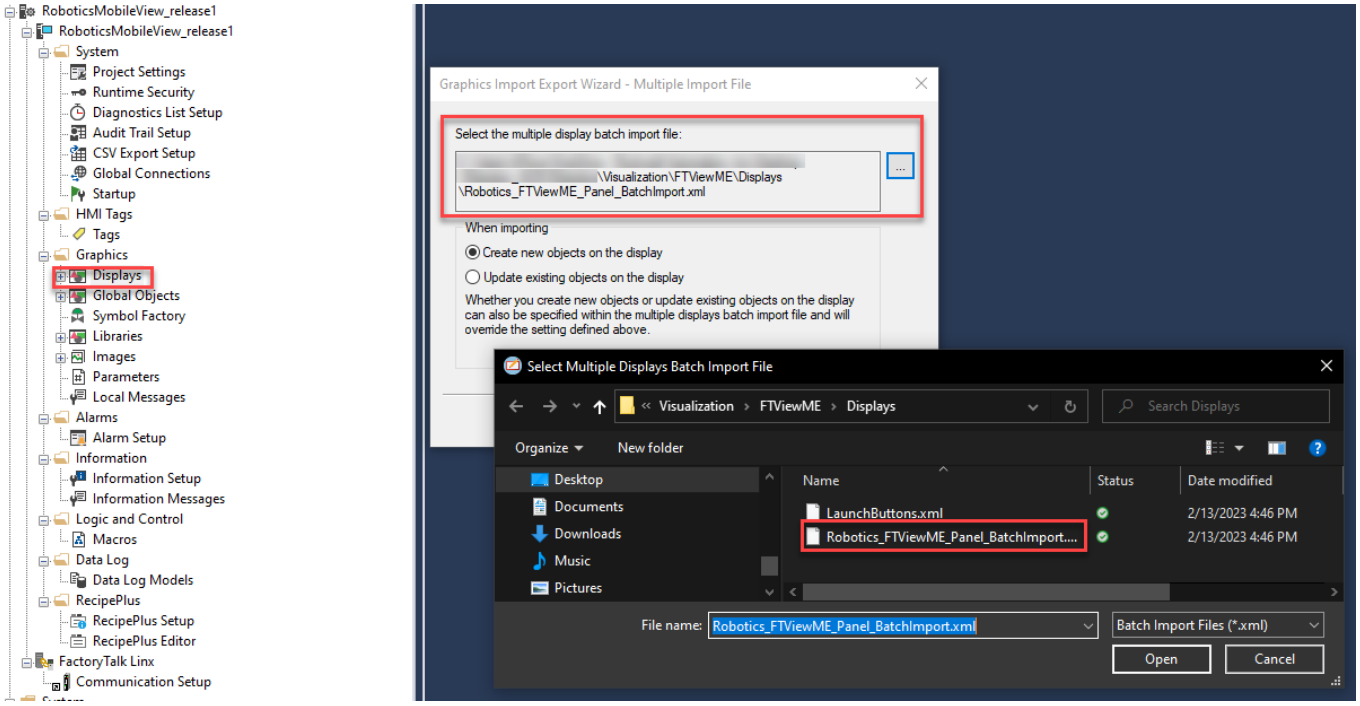
Open Global Connections and select Display Tab. Verify that Replace Display Number Connection tag {[*shortcut*]MobileView1_Sts_CurrentDisplayNo} uses correct shortcut and desired tagname from raM_Robot_rOS_HMI instance in Application Code Manager.



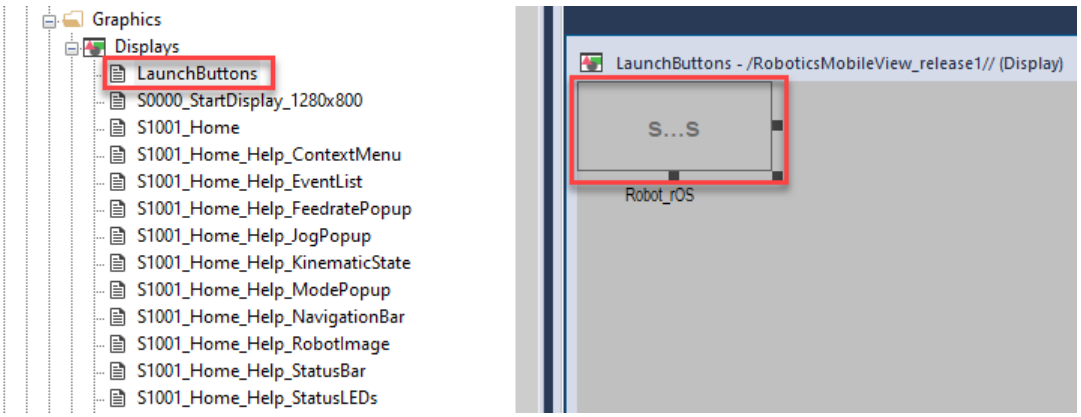


7.4 Inserting Launch Button from ACM generated display

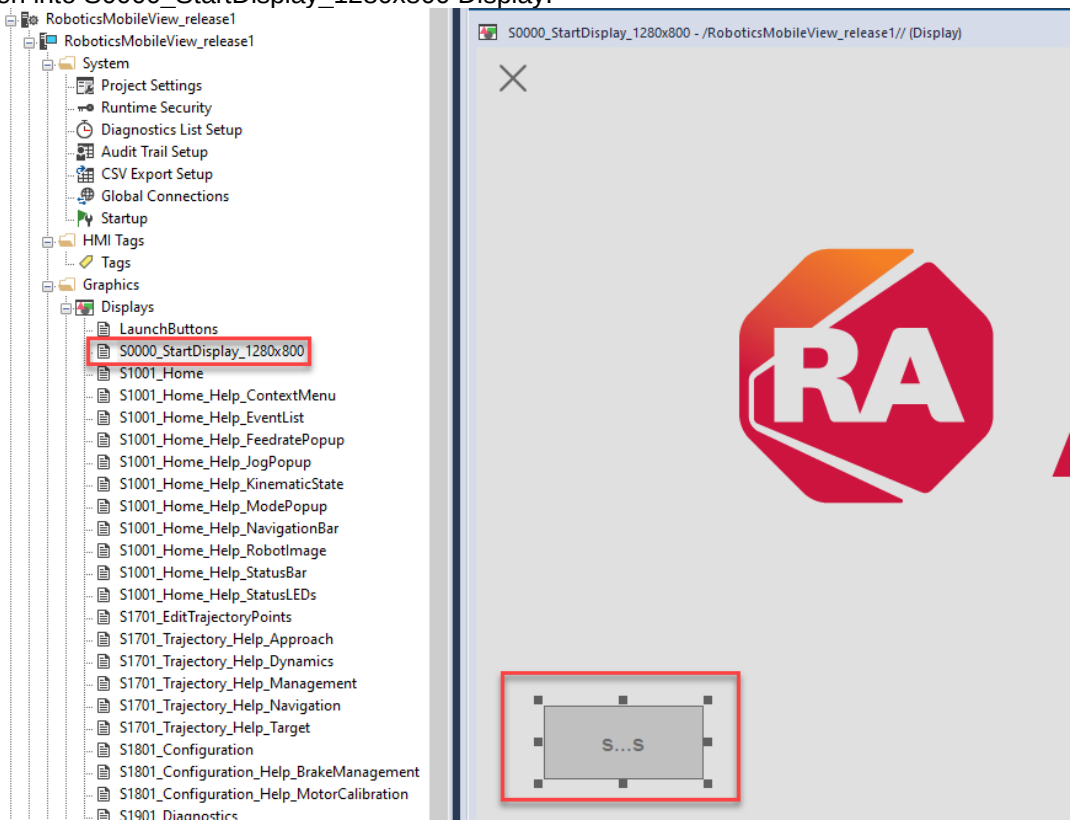
In FactoryTalk View, restore the RoboticsMobileView application. Using the Graphics Import Export Wizard, import Robotics_FTViewME_Panel_BatchImport.xml file.



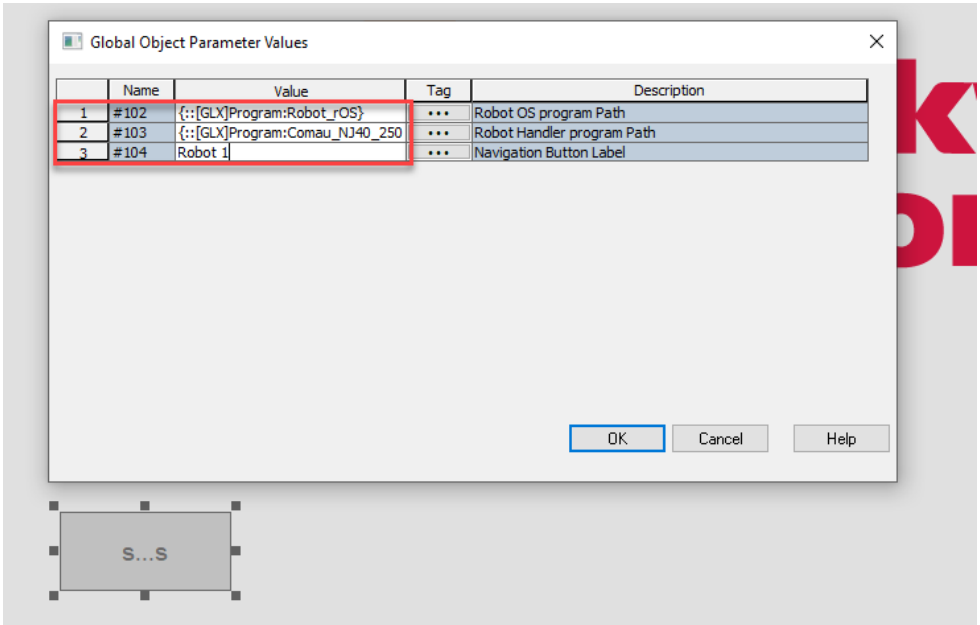
Copy the button from the imported **LaunchButtons** display.



Paste the button into S0000_StartDisplay_1280x800 Display.



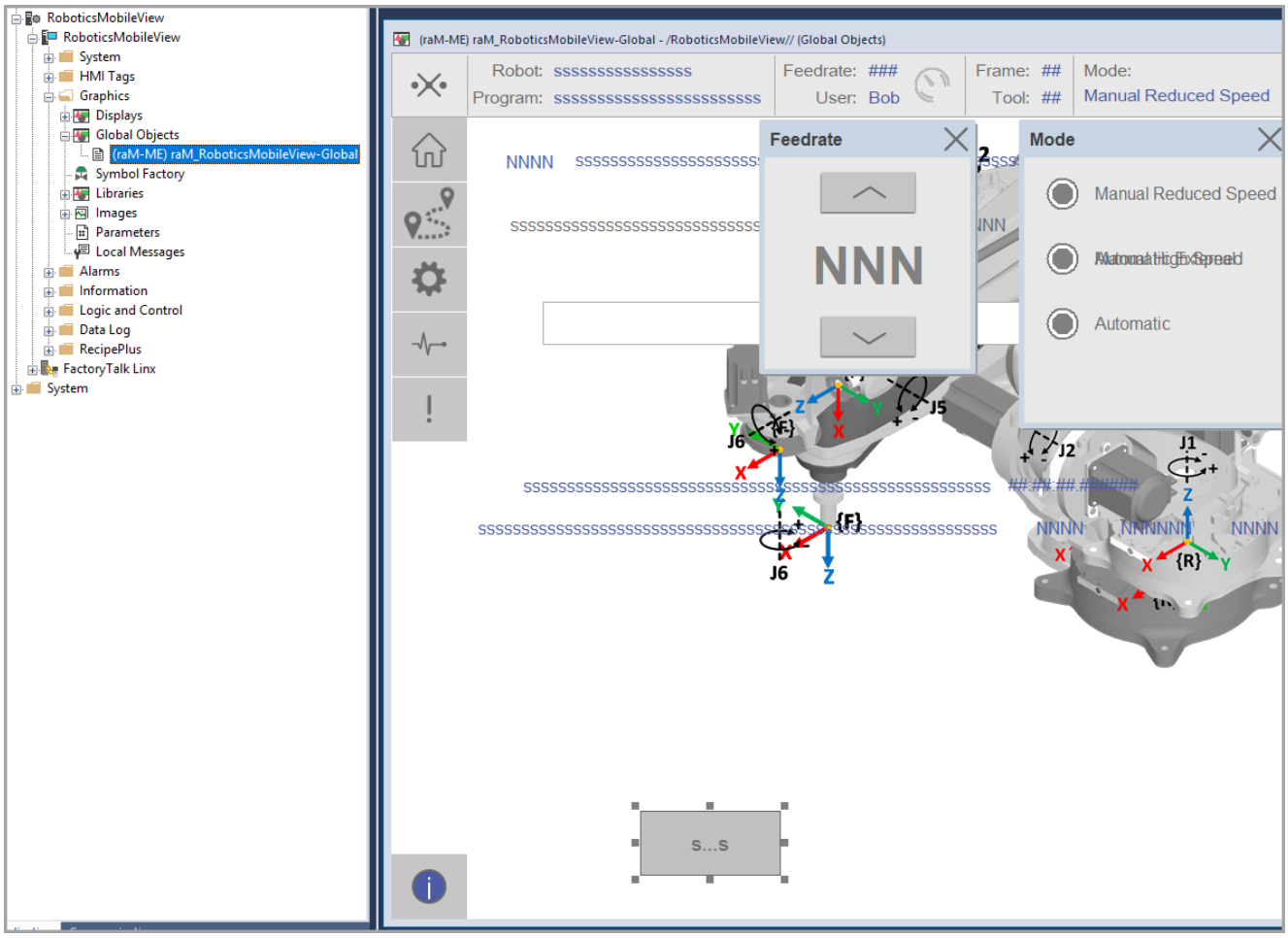
Right-click the button and select **Global Object Parameter Values**. Check that parameter values are correctly referencing the Robot OS and Device Handler programs.



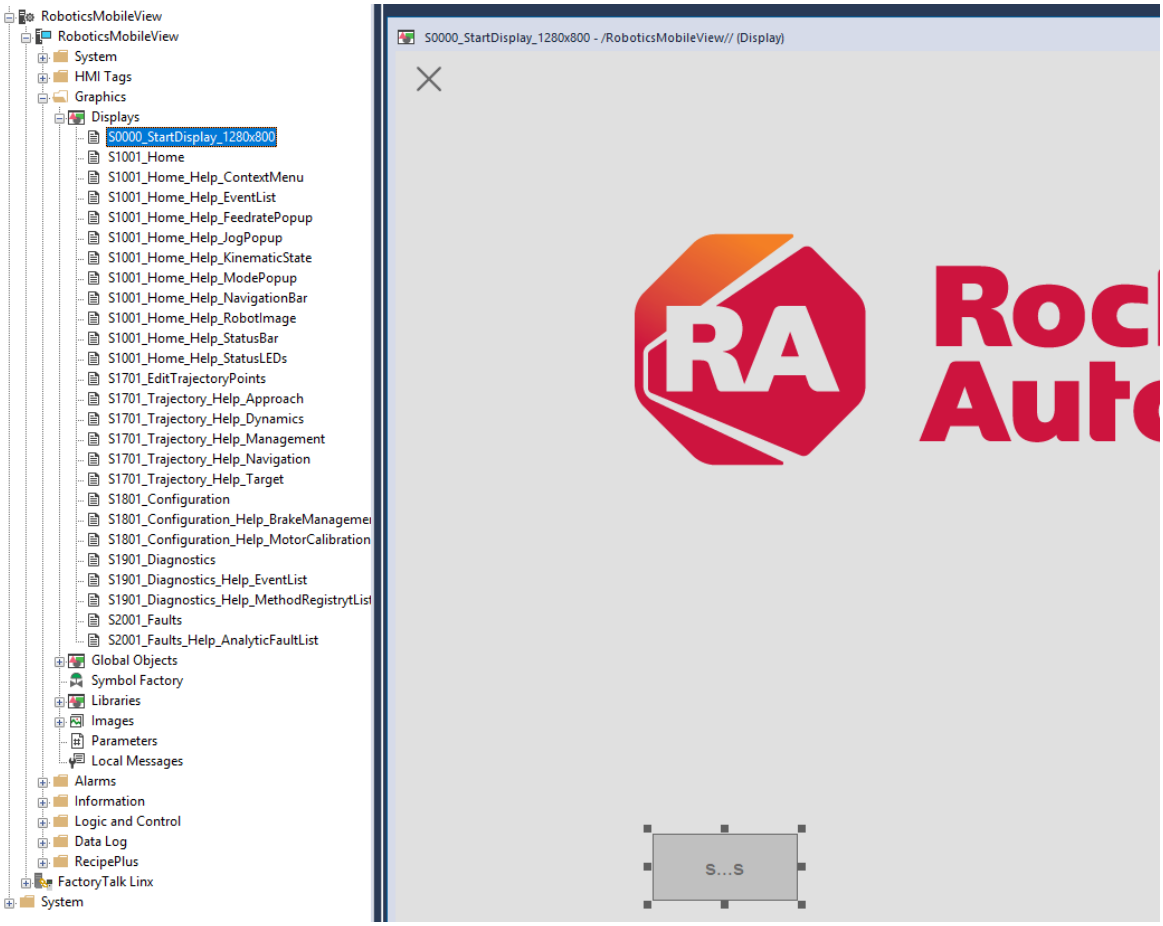
7.5 Configuring Launch Button Manually

This section needs to be executed only if the import of LaunchButtons display was not executed in previous section.

Copy the Launch button from Global Object file.

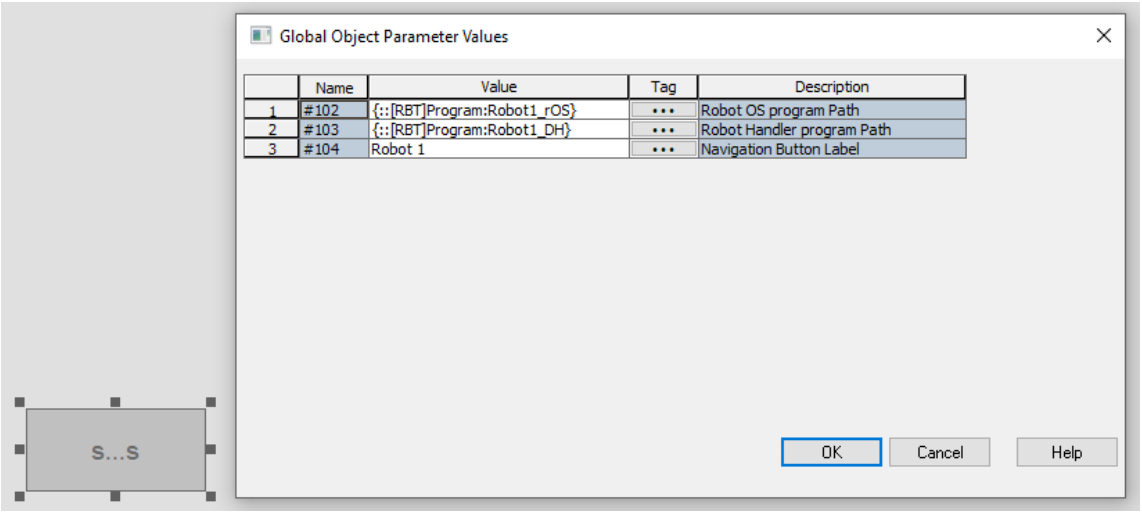


Paste the Launch button for each robot instance into **Start** display.



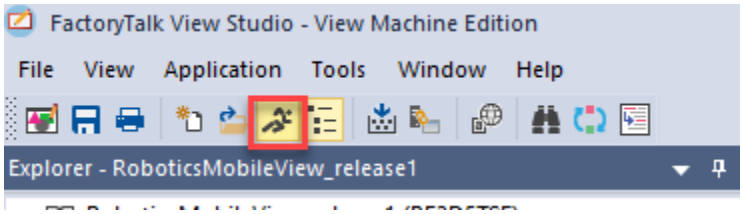
Configure the launch button Global Object Parameter Values

- #102 – Robot OS Program
- #103 – Robot DH Program
- #104 – Launch Button Label



7.6 Testing the Application

If Logix program has been downloaded into controller, test the application in FactoryTalk View.



If the application executed successfully during the test, create a runtime file and download it to the MobileView terminal.
Note: the runtime file version must match the FactoryTalk View Station version installed in MobileView.

8 Startup

This Chapter describes basic procedures and screens related to pendant initialization, including power-up, the **Start** screen, robot selection, and the robot's **Home** page. The installation and setup of the HMI hardware/software is out of the scope of this manual.

8.1 Powering up the Pendant

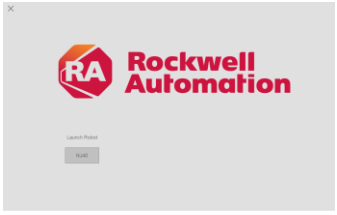
8.1.1 Purpose

If the HMI application is running on a MobileView 2711T terminal, it must execute a quick startup procedure prior to being ready for use.

8.1.2 Preconditions

- The pendant tethered cable is connected to the junction box and robot controller.
- FactoryTalk ME runtime application file has already been transferred and configured to run at startup.
- Robot controller is powered off.

8.1.3 Procedure

Step	Description
1	Energize the robot controller. The pendant will start automatically when 24V DC power is applied to the junction box.
2	When the MobileView 2711T terminal is powered up, a dark screen will be presented for a few seconds. 
3	Wait until the application **Start** screen is displayed. 

8.2 Start Screen

The **Start** screen (Figure 3) is the first screen shown after the pendant power-up routine is complete. It has buttons for robot selection and for closing the application.

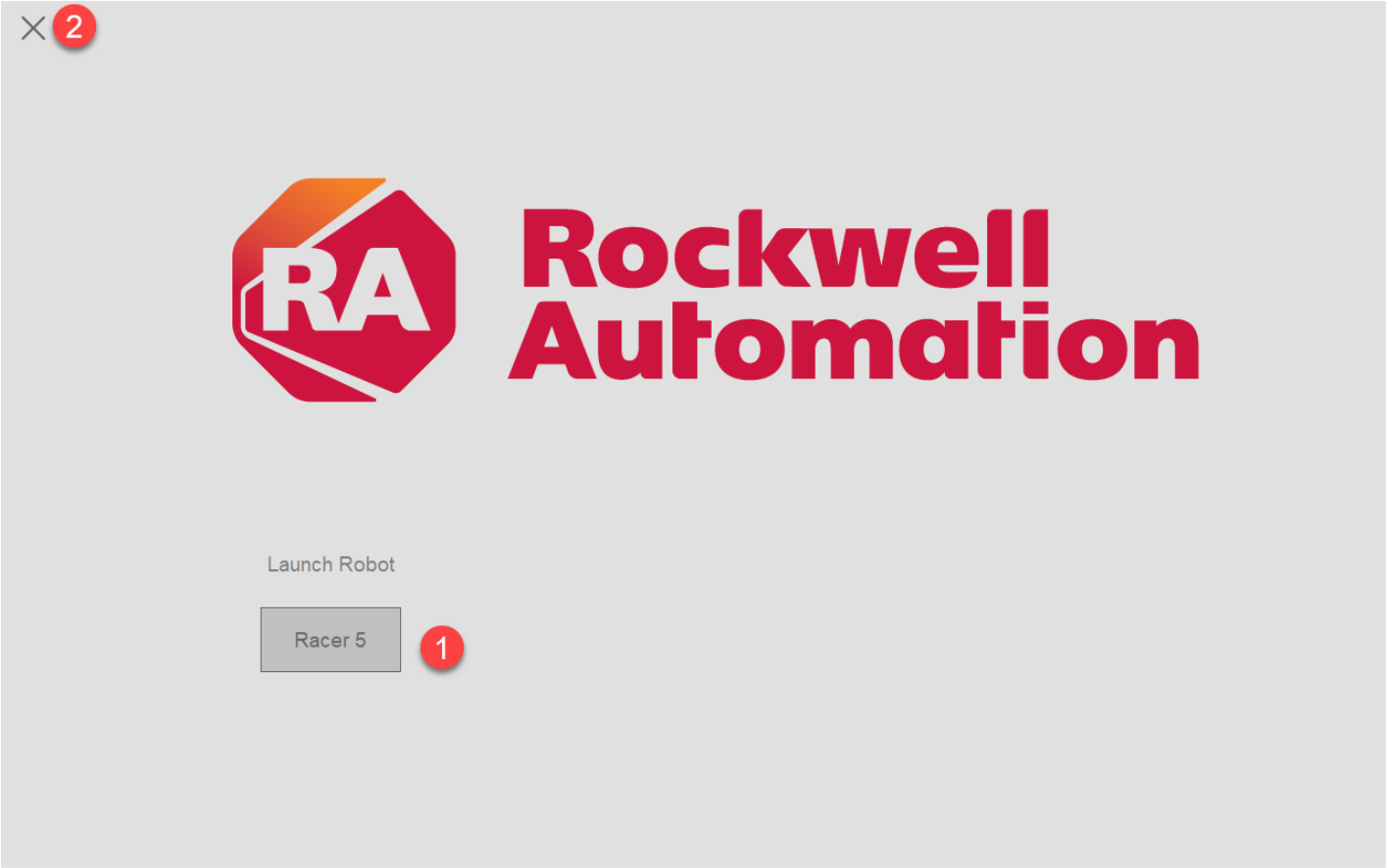


Figure 3- Start screen.

Item	Name	Description
1	Launch Robot button(s)	Launches the HMI screens for a specific robot. There is one dedicated button per available robot.
2	Close Application button	Closes the FactoryTalk View ME application.

8.3 Selecting a Robot

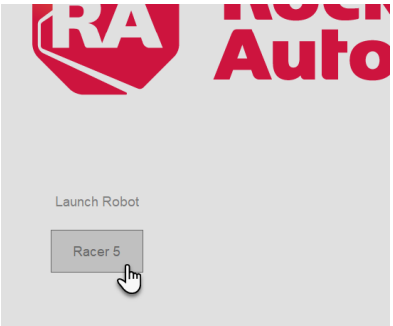
8.3.1 Purpose

If the robot system holds more than one robot, each robot will be provided with a specific set of HMI screens for dedicated monitoring and control. A robot must be selected before any operation can be conducted with the pendant.

8.3.2 Preconditions

- The pendant has just been powered up, or the previously selected robot has been exited.

8.3.3 Procedure

Step	Description
1	From the **Start** screen, click on the **Launch Robot** button matching the desired robot to select. 
2	The selected robot's **Home** page will be displayed.

8.4 Robot Screen

Figure 4 shows the typical appearance of an initial robot-specific screen. The display area has two permanent regions (**Status** bar and **Navigation** bar) and one dynamic region displaying the contents of the currently selected page.

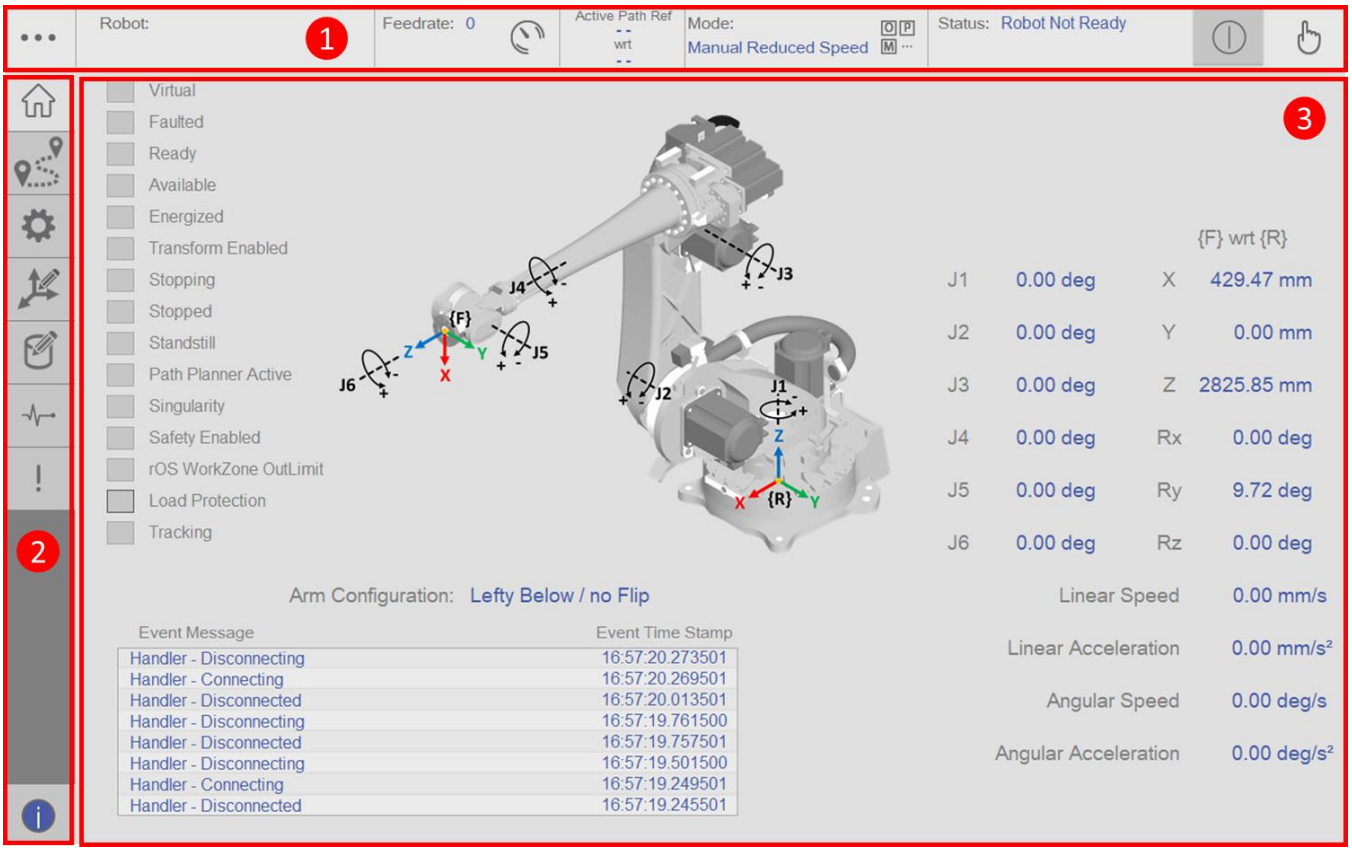


Figure 4 – Layout of the first robot-specific screen.

Item	Name	Description
1	Status bar	Permanent area providing access to general status information and commands that are always available, regardless of the currently selected screen.
2	Navigation bar	Permanent menu providing access to each screen and access to help callouts.
3	Page content	Dynamic area whose content is dependent on the currently selected screen.

The topmost area of the screen, called **Status** bar, is a permanent area holding controls and indicators that are always available regardless of the currently selected page. The status bar shows basic robot information such as name, current feedrate, current operating mode, and status. It also exposes commonly used functions like enabling the robot and jogging. These general operations are further detailed in Chapter 9.

The leftmost area of the screen, called **Navigation** bar (Figure 5), has buttons to navigate between the robot-specific pages.

Figure 5- Navigation bar.

Item	Name	Description
1	Home page button	Opens the **Home** page. Default page displayed when a new robot is selected.
2	Trajectory page button	Opens the **Trajectory** page.
3	Configuration page button	Opens the **Configuration** page.
4	Frame page button	Opens the **Frames Configuration** page
5	Work Zone page button	Opens the **Work Zone Configuration** page
6	Diagnostics page button	Opens the **Diagnostics** page.
7	Faults page button	Opens the **Faults** page.
8	Show Help Callouts button	Displays/hides clickable callouts on the selected page.

The **Navigation** bar also holds a button to toggle the visibility of help callouts distributed across the components of the active page. When clicked, a help callout opens a popup supplying quick HMI guidance on the associated UI element. An example of a help callout and help popup is depicted in Figure 6.

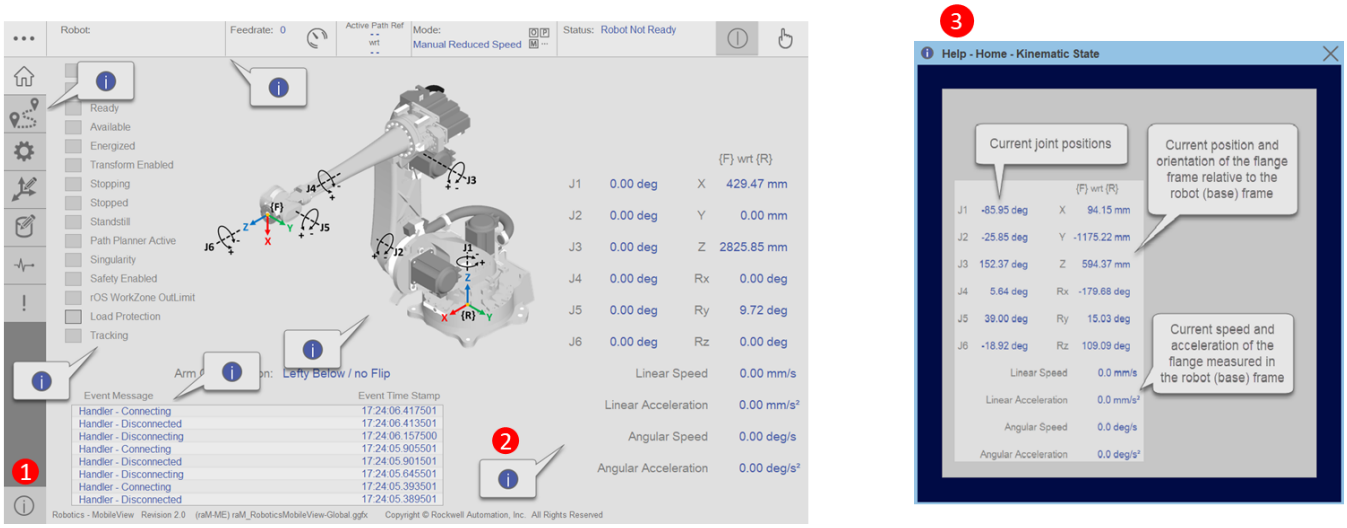


Figure 6- Example of help callout and associated popup.

Item	Name	Description
1	Open Help Callout button	When clicked, shows all the help callouts available for the current page.
2	Help callout	When clicked, opens a dedicated help popup.
3	Help popup	The help popup has quick tips about the UI element associated with the clicked callout.

8.5 Home Page

The **Home** page (Figure 7) has basic information related to the overall robot state, such as robot status, kinematics, and events. It can be used as a monitoring screen during the execution of programs or when the system is idle.

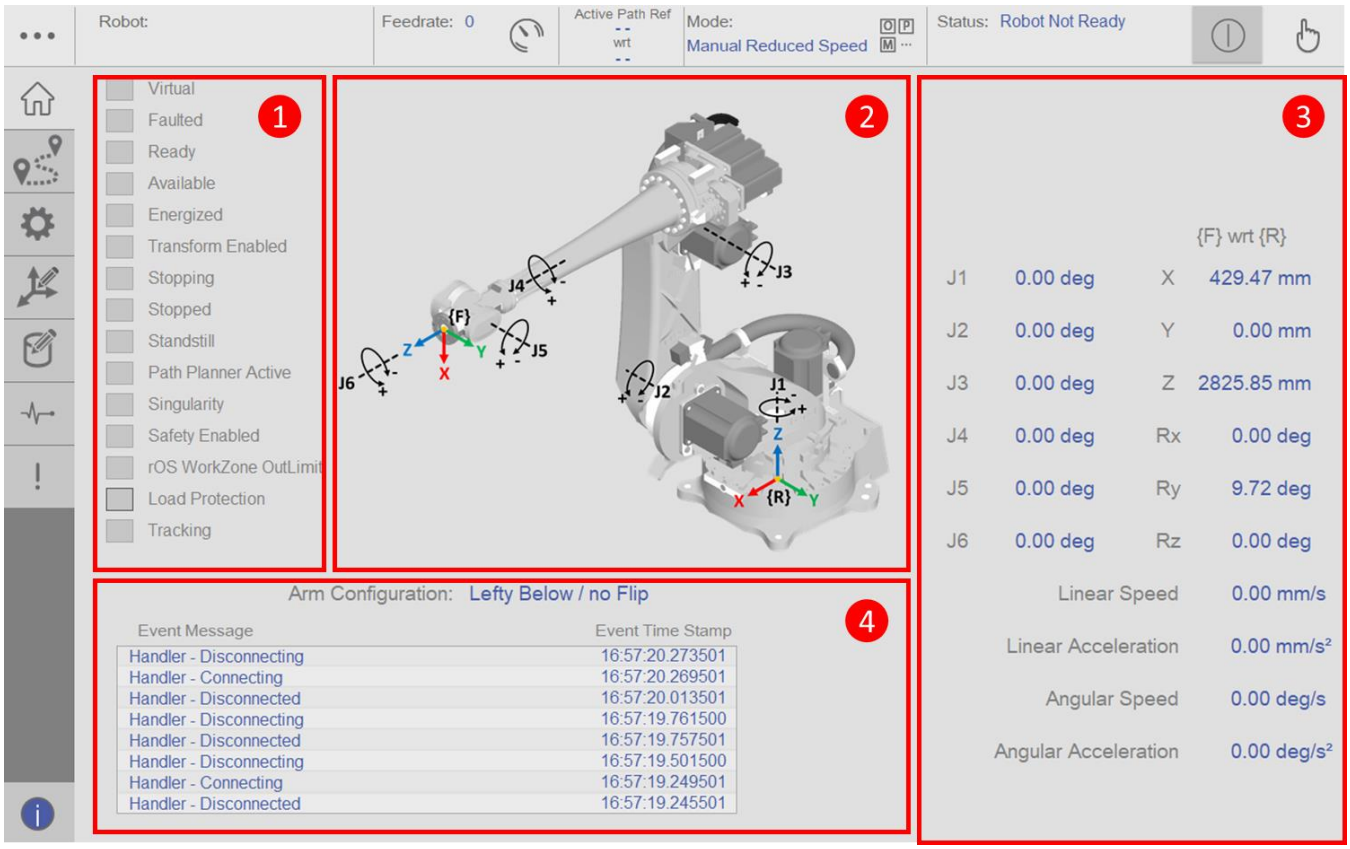


Figure 7- The Home page.

Item	Name	Description
1	Status LEDs group	Displays basic status of the robot and Device Handler.
2	Robot image	Displays the current robot geometry in use, basic frame conventions, and joint direction conventions.
3	Kinematic State group	Displays joint positions, Cartesian pose, Cartesian speeds, and Cartesian accelerations.
4	Event List group	Displays the latest timestamped events.

8.5.1 Status LEDs

The **Status LEDs** group is a set of Boolean indicators displaying the following robot statuses:

Status	Description
Virtual	The robot actuation system is emulated. No physical motors are being used.
Faulted	There is at least one fault active. Operator intervention is needed.
Ready	Hardware is ready.
Available	Robot is configured, communicating, and available for commands.
Energized	All motor axes are powered on.
Transform Enabled	Kinematic calculations are enabled.
Stopping	Path master is decelerating to a stop.
Stopped	Path master is stopped. Planner is inactive.
Standstill	All joint speeds are below a minimum threshold and thus assumed fully stopped.
Path Planner Active	Path points are being processed by the planner.
Singularity	A kinematic singularity is active. Cartesian moves are not allowed.
Safety Enabled	Safe Torque Off is not active and drives are capable of producing torque
rOS WorkZone OutLimit	One or more zones configured via rOS is out of the defined limit
Load Protection	The Load Protection feature is active and controlling feedrate
Tracking	Robot line tracking is active

These indicators are particularly important for operation since the statuses are used as permissives for many functions.

8.5.2 Robot Image

Each compatible robot geometry has a dedicated reference image on the **Home** page's center region depicting:

- The mechanical assembly.
- Joint names.
- Joint axes.
- Positive and negative joint directions.
- The placement for the **Robot** frame {R}.
- The placement for the **Flange** frame {F}.

Figure 8 shows an example of **Robot** image for the Comau NJ-40-2.5 robot. Note that, as a rule, the **Robot** frame Z-axis points up, joints are named J1, J2, …, Jn, and their positive directions follow the right-hand rule.

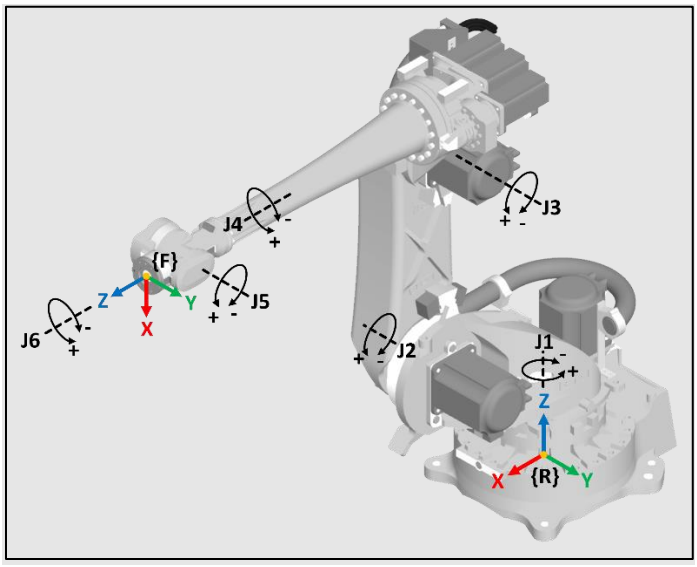


Figure 8- Robot image for the Comau NJ-40-2.5 robot.

IMPORTANT

The conventions for frames and joint directions do not necessarily follow the manufacturer datasheets. As Rockwell Automation Robotics Libraries offer a unified approach, all supported models must adapt to Rockwell Automation motion coordinate system conventions. This standardization might produce inconsistencies between the Robot image conventions and the physical markings on the mechanical unit.

IMPORTANT

The models depicted in the **Robot** image are usually NOT represented at the zero-angle joint configuration.

8.5.3 Kinematic State

The **Kinematic State** group (Figure 9) enables the visualization of Cartesian and Joint kinematic information related to the selected robot, including positions, speeds, and accelerations.

Figure 9- Kinematic State group.

Item	Name	Description
1	Joint Kinematic State group	Displays the robot joint positions.
2	Cartesian Kinematic State group	Displays the flange Cartesian pose, speed, and acceleration relative to **Robot** frame.

The following joint kinematics information is displayed:

Item	Unit	Description
J1…Jn	mm (prismatic joints) or deg (revolute joints)	Actual position of joint Jn relative to its home position.

The following Cartesian kinematics information is displayed:

Item	Unit	Description
X	mm	Actual position of the **Flange** frame measured as a translation along the X axis of the **Robot** frame.
Y	mm	Actual position of the **Flange** frame measured as a translation along the Y axis of the **Robot** frame.
Z	mm	Actual position of the **Flange** frame measured as a translation along the Z axis of the **Robot** frame.
Rx	deg	Actual orientation of the **Flange** frame measured as a rotation around the X axis of the **Robot** frame.
Ry	deg	Actual orientation of the **Flange** frame measured as a rotation around the Y axis of the **Robot** frame.
Rz	deg	Actual orientation of the **Flange** frame measured as a rotation around the Z axis of the **Robot** frame.
Linear Speed	mm/s	Actual linear speed of the **Flange** frame relative to **Robot** frame (magnitude of the velocity vector).
Linear Acceleration	mm/s^2	Actual linear acceleration of the **Flange** frame relative to **Robot** frame (magnitude of the acceleration vector).
Angular Speed	deg/s	Actual angular speed of the **Flange** frame relative to **Robot** frame (magnitude of the angular velocity vector).
Angular Acceleration	deg/s^2	Actual angular acceleration of the **Flange** frame relative to **Robot** frame (magnitude of the angular acceleration vector).

The Rx, Ry, and Rz are Euler angles following the XYZ fixed-frame rotation sequence. For more information on the conventions for representing orientations, see Section 3.7.

IMPORTANT If the robot is uncalibrated and/or transforms disabled, the displayed kinematic state is unreliable and should not be used.

8.5.4 Event List

The **Event** list provides basic timestamped information related to statuses, warnings, and faults registered by the system. The following event data is displayed in chronological order (latest first):

Column	Description
Event Message	Textual description of the event. It might hold just a message, or more detailed data such as the method that fired it.
Event Time Stamp	Time when the event occurred, following the “hour:minute:second:millisecond” format.

More detailed information about events can be retrieved from the **Diagnostics** page (Section 14.1) and **Faults** page (Section 0).

9 General Operations

This Chapter presents the general operations available through the **Status** bar.

9.1 Status Bar

The **Status** bar (Figure 10) is a permanent area of the display offering general control and monitoring functions, including robot information, robot selection, operating mode selection, feedrate selection, enabling/disabling, fault clearing, and jogging.

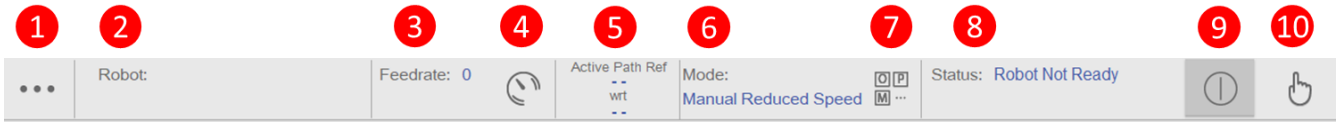


Figure 10- Status bar.

Item	Name	Description
1	Context Menu button	Opens a specific context menu for the selected screen.
2	Robot Name indicator	Displays the currently selected robot.
3	Current Feedrate indicator	Displays the current feedrate.
4	Feedrate Popup button	Opens the **Feedrate** popup and allows for feedrate changes.
5	Active Path Reference Frame	When a path is active, displays the current frame of reference
6	Current Mode indicator	Displays the current operating mode.
7	Mode Popup button	Opens the **Mode** popup and allows for operating mode changes.
8	Robot Status indicator	Displays the current robot status.
9	On / Off / Clear Faults buttons	Depending on current robot state, one of the following three buttons will be visible: Energizes the robot. De-energizes the robot. Acknowledges and clears faults.
10	Jog Popup button	Opens the **Jog** popup and allows the operator to jog the robot while in Manual Reduced Speed mode.

9.2 Unselecting a Robot

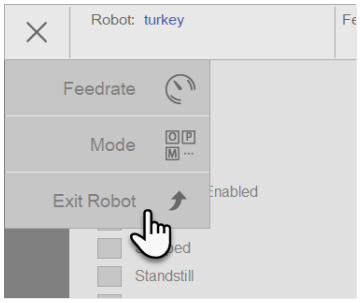
9.2.1 Purpose

The currently selected robot must be first unselected before a new robot can be launched.

9.2.2 Preconditions

- A robot has already been launched from the **Start** screen.

9.2.3 Procedure

Step	Description
1	From any screen, click on the • • • button on the top left corner to open the **Context** menu.
2	Select the **Exit Robot** option to go back to the **Start** screen. 
3	Select a new robot, following the robot selection procedure described in Section 8.3.

9.3 Energizing the Robot

9.3.1 Purpose

A robot must be energized to move. Thus, the bus and motors must be enabled before any motion task can be executed by the robot.

9.3.2 Preconditions

- Robot is not faulted.
- Robot is available.
- Robot is not energized.
- Power supply is ready.
- Safety input is enabled.
- Operating Mode is **Manual Reduced Speed** or **Manual High Speed**.

9.3.3 Procedure

Step	Description
1	From any screen, find the **Status** bar and click on the  button to energize the robot.
2	If the operation was successful, robot will be energized. Otherwise, energization faults will be raised by the controller.

9.4 De-energizing the Robot

9.4.1 Purpose

When robot motion is not intended, the bus and motors can be disabled.

9.4.2 Preconditions

- Robot is energized.
- Robot is not faulted.
- Robot is available.
- Operating mode is **Manual Reduced Speed** or **Manual High Speed**.

9.4.3 Procedure

Step	Description
1	From any screen, find the **Status** bar and click on the  button to de-energize the robot.
2	If the operation was successful, robot will be de-energized. Otherwise, de-energization faults will be raised by the controller.

9.5 Clearing Faults

9.5.1 Purpose

When the robot has any active faults, the available functionality is limited. Faults must be acknowledged and cleared before most operations can be conducted with the robot.

9.5.2 Preconditions

- Robot is faulted.
- Robot is available.
- Operating mode is **Manual Reduced Speed** or **Manual High Speed**.

9.5.3 Procedure

Step	Description
1	Locate and fix the problem that is causing the fault.
2	From any screen, find the **Status** bar and click on the  button to acknowledge and clear faults.
3	If the operation was successful, the active faults will be cleared. Otherwise, faults will remain active.

9.6 Adjusting the Feedrate

9.6.1 Purpose

The master axis used for scaling the motion profiles. The speed of this axis is known as the feedrate of the system. At a feedrate of 100%, the motion profiles are followed with the pre-defined settings corresponding to their units (mm/s, deg/s, etc). When changing the feedrate to 50%, the calculated profile will be used, but scaled by 1/2. This will result in half of the speed, acceleration, and deceleration. Consequently, adjusting the feedrate will affect how fast the robot will move when jogging and when executing programs. The robotics libraries support feedrates from 0% to 125%.

9.6.2 Interface

The **Feedrate** popup (Figure 11) is accessible from the **Status** bar and enables the adjustment of the current feedrate.

Figure 11- The Feedrate popup.

Item	Name	Description
1	Increment Feedrate button	Increments the feedrate setpoint each time the button is pressed.
2	Feedrate Setpoint indicator	Displays the feedrate setpoint.
3	Decrement Feedrate button	Decrements the feedrate setpoint each time the button is pressed.

9.6.3 Preconditions

- Robot is not faulted.
- Robot is available.
- Robot is energized.
- Operating mode is **Manual Reduced Speed** or **Manual High Speed**.

9.6.4 Procedure

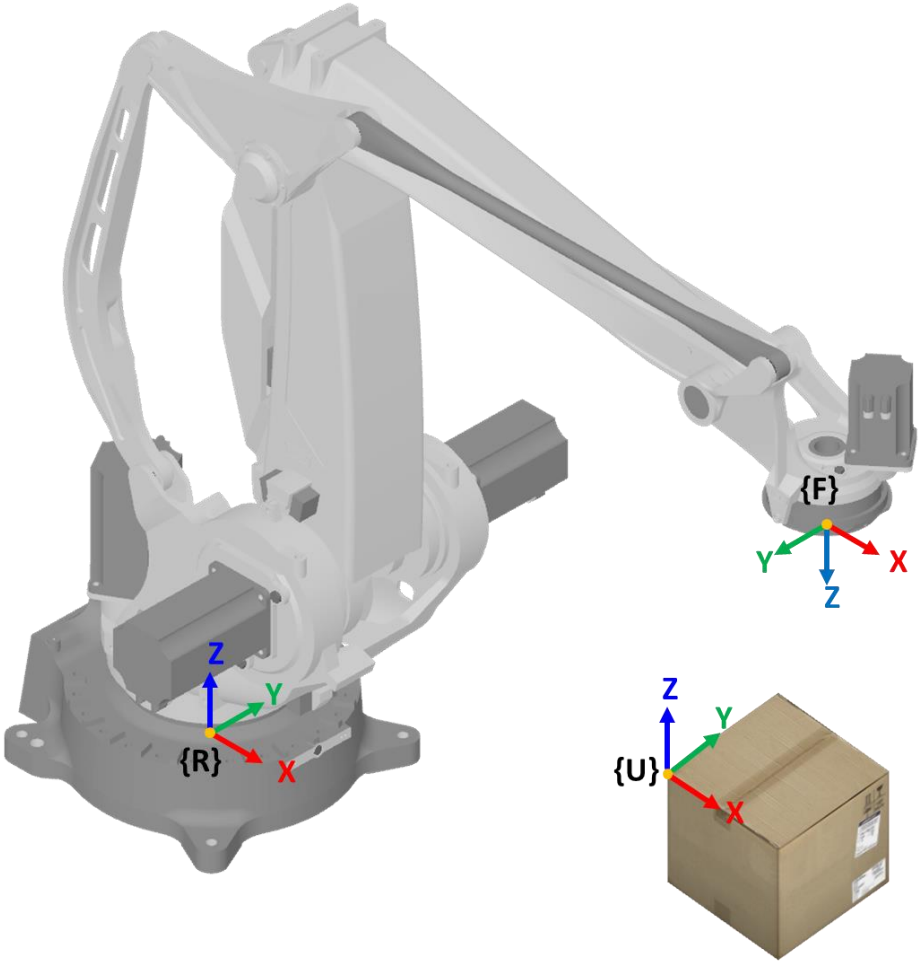
Step	Description
1	From any screen, find the **Status** bar and click on the  button to open the **Feedrate** popup.
2	Click on the **Increment Feedrate** and **Decrement Feedrate** buttons to adjust the feedrate setpoint.
3	Click on the  button to close the **Feedrate** popup.

Step	Description
4	Check if the **Current Feedrate** indicator on the **Status** bar matches the setpoint value. Feedrate: 0 If the operation was unsuccessful, feedrate faults will be raised by the controller. Feedrate may also be affected by the Load Protection feature and setpoints may not match actual feedrate.

9.7 Understanding the Active Path Frame Display

9.7.1 Purpose

When using Frames, it is important to understand which Frame you have selected as the robot will move with respect to that frame. Jogging the robot with the Robot Frame active will yield different results than jogging the robot with a User Frame active. This display shows you which Frame is active so when you jog the robot you understand its movements.



9.8 Selecting the Operating Mode

9.8.1 Purpose

The robot controller restricts the allowed functions of the robot with three different operating modes: **Manual Reduced Speed**, **Manual High Speed Mode**, and **Automatic External**.

The **Manual Reduced Speed** mode allows a robot to be manually commanded by the operator. Used for jogging, teaching, programming, program verification, and maintenance. Note that this is not a safety validated setting.

The **Manual High Speed** mode allows the operator to test a series of trajectories configured from the Trajectory Test screen. Path execution dynamics default to the Jog dynamics, however, they are switchable to operate with the programmed dynamics. This mode is used for program path verification and maintenance.

The Trajectory Test widget is accessible from the content menu button located on the trajectory test screen. This widget only tests trajectories created on the trajectory screen.

The **Automatic External** mode allows the execution of external robot programs without human intervention This mode disables the following HMI functionalities:

- Jogging.
- Setting the feedrate.
- Configuring the robot.

9.8.2 Interface

The **Mode** popup (Figure 12) is accessible from the **Status** bar and enables the switchover between the available operating modes.

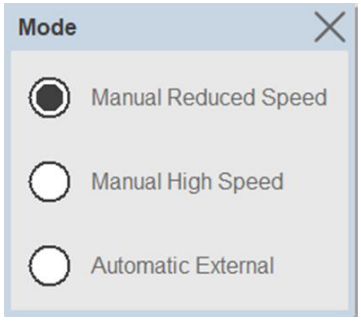


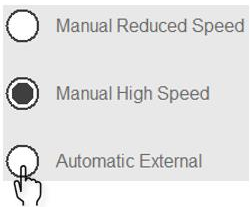
Figure 12- The Mode popup.

9.8.3 Preconditions

- Robot is stopped.

9.8.4 Procedure

Step	Description
1	From any screen, find the **Status** bar and click on the  button to open the **Mode** popup.

Step	Description
2	Select the desired operating mode. 
3	Click on the  button to close the **Mode** popup.

9.9 Jogging

Jogging a robot means using the pendant to manually position/move it in Cartesian or Joint space. One of the most important uses for jogging is to move the robot towards a target to be taught and later used when programming move sequences.

9.9.1 Jog Popup

The **Jog** popup (Figure 13) is accessible from the **Status** bar and centralizes all the functions needed to jog a robot.

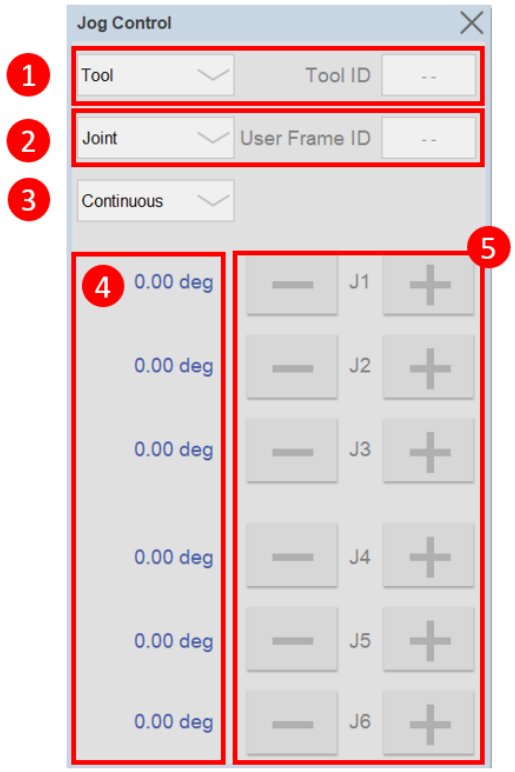


Figure 13- The Jog popup.

Item	Name	Description
1	Frame Selection dropdown & display	Selects the Frame to jog in (Tool or Flange). When Tool is selected, enter a Tool Frame ID of a previously defined tool frame.

Item	Name	Description
2	Jog Coordinate System dropdown	Selects the coordinate system frame for jogging commands. (World, Robot, Flange, Tool, User, Joint). When User is selected, enter a User Frame ID of a previously defined user frame.
3	Jog Increment dropdown	Selects the incremental jog step.
4	Jog Axis Positions group	Display the joint positions when joint jogging, or the flange or tool position and orientation relative to the current reference frame when Cartesian jogging.
5	Jog Keys group	Each key commands the associated axis to move when pressed and commands the associated axis to stop when released. The keys refer to either Cartesian axes or Joint axes, depending on the selected coordinate system.

IMPORTANT

A ⚠ symbol present in the Tool or User Frame ID box indicates that the Tool or User Frame selected has not been defined. For more information on the procedures for defining frames, see Chapter 12.

9.9.1.1 Jog Coordinate System Dropdown

The selectable coordinate systems used to jog a robot: **World, Robot, Flange, Tool, User, Joint**.

When joint jogging, each joint is moved independently in its positive or negative direction. Example: pressing the **J3+** jog key will command the joint J3 to move in the positive direction, while all other joints remain stopped.

When Cartesian jogging, the end-effector is moved relative to the positive or negative directions of a reference frame axis. Example: if the current jog coordinate system is **Robot**, pressing the **X-** jog key will translate the end-effector along the X-axis of the **Robot** frame, in the negative direction; pressing the **Ry+** jog key will rotate the end-effector around the Y-axis of the **Robot** frame, in the positive direction.

The possible values for jog coordinate systems selectable from the **Jog Coordinate System** dropdown in the **Jog** popup are:

Coordinate System	Description
World	Cartesian jogging - jog commands change the position and orientation of the flange or selected tool relative to **World** frame.
Robot	Cartesian jogging - jog commands change the position and orientation of the flange or selected tool relative to **Robot** frame.
Flange	Cartesian jogging - jog commands change the incremental position and orientation of the flange relative to either itself or the selected tool.
Tool	Cartesian jogging - jog commands change the incremental position and orientation of the selected tool relative to either itself or the flange.
User	Cartesian jogging - jog commands change the position and orientation of the flange or selected tool relative to the selected **User** frame.
Joint	Joint jogging - jog commands change the position of each joint individually.

For more information on frames and coordinate systems, see Section 3.6.

9.9.1.2 Jog Increment Dropdown

Incremental jogging is a feature that offers precise positioning by jogging the robot in pre-defined position steps. When an increment is set, the axis will move one step each time the jog key is pressed and held.

Once the set increment has been reached, the robot will stop. If the jog key is released while an increment is under execution, the motion will be immediately interrupted. A new increment is started if the jog key is pressed again.

The available jog increments selectable from the **Jog Increment** dropdown in the **Jog** popup are:

Jog Increment	Description
Continuous	Default value. Incremental jogging is deactivated. The robot moves continuously while the jog key is pressed.
100 mm / 10°	1 step = 100 mm or 10°.
10 mm / 5°	1 step = 10 mm or 5°.
1 mm / 1°	1 step = 1 mm or 1°.
0.1 mm / 0.1°	1 step = 0.1 mm or 0.1°.

9.9.1.3 Jog Axis Positions Group

The **Jog Axis Positions** group provides a convenient way of visualizing the axis positions while jogging. The values and units depend on the currently selected jog coordinate system.

When joint jogging, the following information is displayed:

Item	Unit	Description
J1…Jn	mm (prismatic joints) or deg (revolute joints)	Actual position of joint Jn relative to its home position.

When Cartesian jogging, the following information is displayed:

Item	Unit	Description
X	mm	Actual position of the flange origin measured as a translation along the X axis of the selected reference frame.
Y	mm	Actual position of the flange origin measured as a translation along the Y axis of the selected reference frame.
Z	mm	Actual position of the flange origin measured as a translation along the Z axis of the selected reference frame.
Rx	deg	Actual orientation of the flange measured as a rotation around the X axis of the selected reference frame.
Ry	deg	Actual orientation of the flange measured as a rotation around the Y axis of the selected reference frame.
Rz	deg	Actual orientation of the flange measured as a rotation around the Z axis of the selected reference frame.

The Rx, Ry, and Rz are Euler angles following the XYZ fixed-frame rotation sequence. For more information on the conventions for representing orientations, see Section 3.7.

ATTENTION

If the robot is uncalibrated and being jogged, the displayed positions are unreliable and should not be used. Watch the robot motion carefully under these circumstances. For more information on the procedures and cautions for calibration, see Section 11.2.

9.9.2 Joint Jogging

9.9.2.1 Purpose

Joint jogging enables manually controlling the position of each motor axis, reaching any point within the robot's envelope. The resulting motion of the joints can be easily predicted by inspecting the conventions for positive and negative joint directions. However, controlling the motion of the end-effector may be difficult – tasks like moving the tool

100 mm to the right with constant orientation are trial-and-error and imprecise. In some situations, jogging in joint space may be the quickest way of reaching a destination, or even the only available choice (e.g., when calibrating axes or moving through singularities).

In summary, joint jogging trades off sophisticated and intuitive end-effector motion for reliable and fast joint motion.

9.9.2.2 Preconditions

- Robot is not faulted.
- Robot is available.
- Robot is energized.
- The desired feedrate is selected. (>0)
- Current operating mode is **Manual Reduced Speed**.

9.9.2.3 Procedure

Step	Description
1	From any screen, find the **Status** bar and click on the  button to open the **Jog** popup.
2	Expand the **Jog Coordinate System** dropdown and select **Joint**.
3	Expand the **Jog Increment** dropdown and select the desired incremental jog step.
4	Press a **Plus (+)** jog key to move the associated joint axis in the positive direction. Press a **Minus (-)** jog key to move the associated joint axis in the negative direction. The joint axis will keep moving until a limit is reached or the jog key is released. Monitor the joint axis positions with the **Jog Axis Positions** group.
5	Click again on the  button to close the **Jog** popup.

IMPORTANT

An uncalibrated robot can only be jogged using the **Joint** coordinate system. For more information on the procedures and cautions for proper calibration, see Section 11.2.

9.9.3 Cartesian Jogging

9.9.3.1 Purpose

Cartesian jogging provides an intuitive way of translating and rotating the end-effector relative to the axes of a given reference frame. It is an appropriate choice for fine tasks like moving the end-effector 500 mm on a straight line while maintaining the same orientation. However, special attention must be paid to not cross singular configurations. Also, jogging in Cartesian space cannot be used when calibrating axes because the calculated positions are unreliable. In summary, Cartesian jogging trades off reliable and fast joint motion for sophisticated and intuitive end-effector motion.

The Robotics Libraries support Cartesian jogging in the **World, Robot, Flange, Tool, and User** frames.

9.9.3.2 Preconditions

- Robot is not faulted.
- Robot is available.

- Robot is energized.
- The desired feedrate is selected. (>0)
- Current operating mode is **Manual Reduced Speed**.
- Robot is calibrated.
- Transforms are enabled.

9.9.3.3 Procedure

Step	Description
1	From any screen, find the **Status** bar and click on the  button to open the **Jog** popup.
2	Expand the **Jog Coordinate System** dropdown and select the desired reference frame.
3	Expand the **Jog Increment** dropdown and select the desired incremental jog step.
4	Press a **Plus (+)** jog key to move relative to the associated Cartesian axis in the positive direction. Press a **Minus (-)** jog key to move relative to the associated Cartesian axis in the negative direction. The axis will keep moving until a limit is reached or the jog key is released. Monitor the axis positions with the **Jog Axis Positions** group.
5	Click again on the  button to close the **Jog** popup.

10 Programming

The current release of the HMI software supports basic robot programming through the explicit definition of user trajectories. The term trajectory used here as a synonym for an ordered collection of moves, i.e., a move sequence.

Each trajectory sample (move) is fully described by three aspects:

- where to go - a target (destination) relative to the previous target.
- what is the shape of the geometric path connecting the previous target to the next target.
- how fast the robot will move along the path.

The first aspect of the move definition, namely creating and storing the targets, constitutes a preliminary support for what is commonly referred to as “target teaching”.

10.1 Trajectory Page

The **Trajectory** page (Figure 14) contains all the necessary controls to create and edit move sequences.

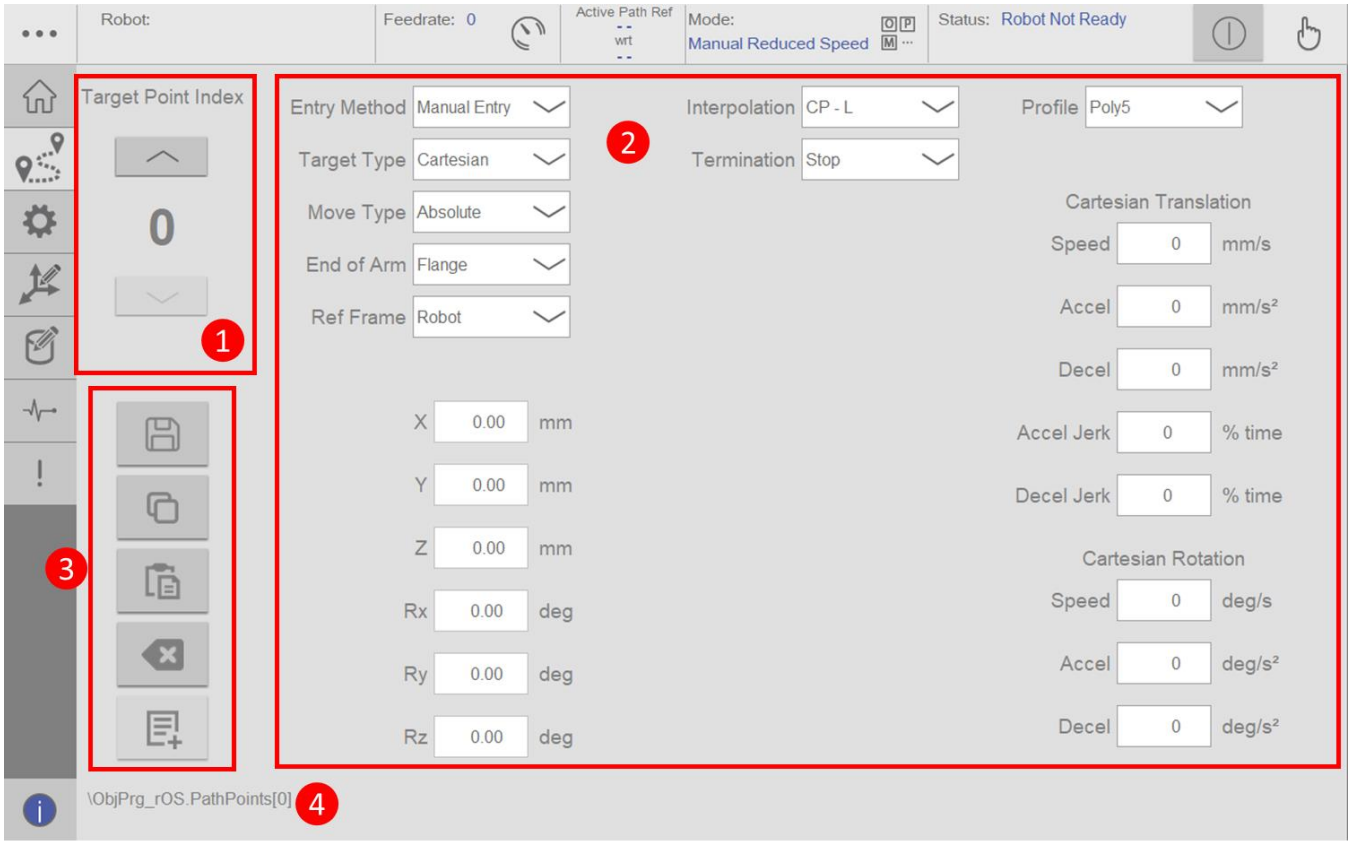


Figure 14- The Trajectory page.

Item	Name	Description
1	Trajectory Navigation group	Selects a specific move from the sequence.
2	Move Parameters group	Holds all the parameters for specifying a single move: the target (where), the path (how), and the timing (when).
3	Move Management group	Provides functions to save, clear, copy, and paste moves.

Item	Name	Description
4	Move Tag indicator	Displays the address of the PLC tag that can be used to programmatically access the selected trajectory.

10.1.1 Trajectory Navigation Group

The **Trajectory Navigation** group (Figure 15) enables the selection of individual moves from the sequence.

Figure 15- Trajectory Navigation group.

Item	Name	Description
1	Increment Move Index button	Navigates to the next move in the trajectory.
2	Current Move Index indicator	Displays the current move being edited (using zero-based indexing).
3	Decrement Move Index button	Navigates to the previous move in the trajectory.

10.1.2 Move Parameters Group

The **Move Parameters** group holds all the parameters controlling the definition of a move. For better clarity, this group can be logically divided into three separate parameter sets, according to which aspect of the move they relate to:

- move target (the destination).
- move path (the shape).
- move timing (the “dynamics”).

10.1.2.1 Move Target Parameters

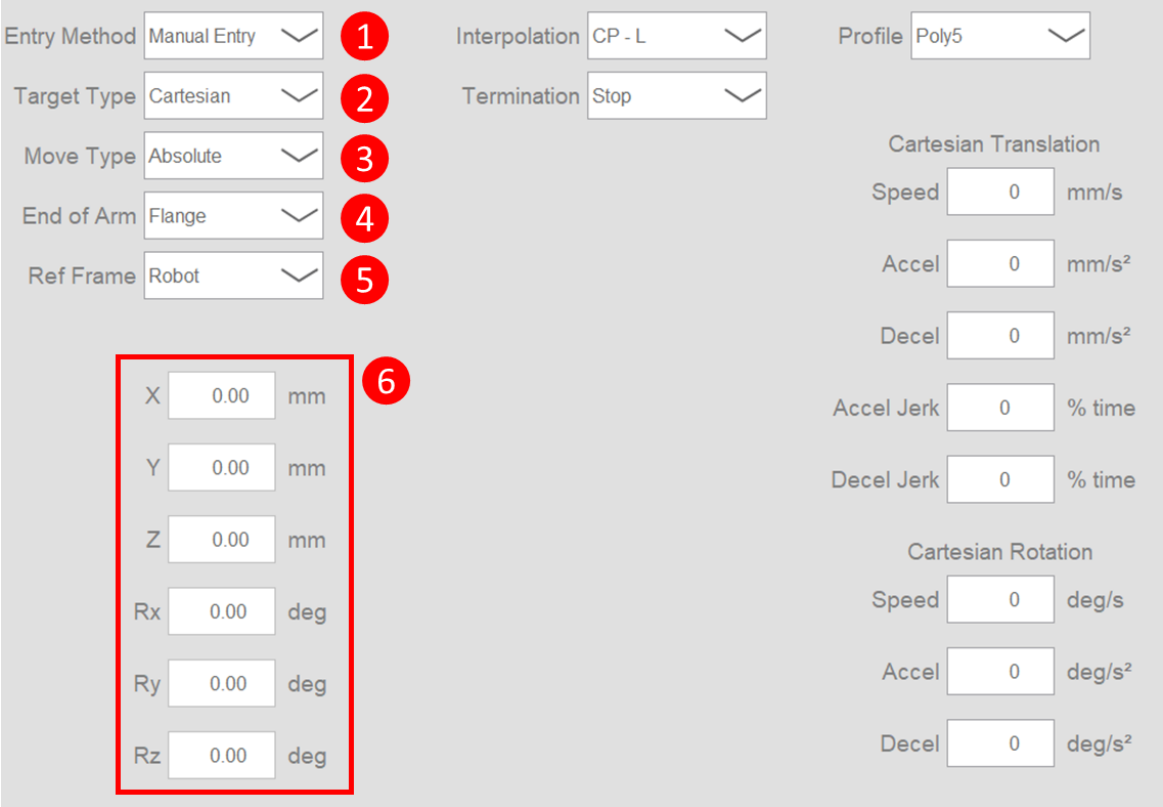
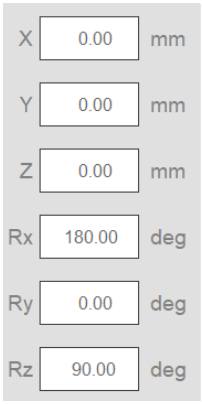
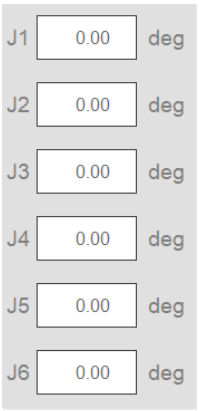


Figure 16- Move Target parameters.

Item	Name	Description
1	Entry Method dropdown	Selects the method for teaching the target. Current: when the **Insert Current** icon is pressed, the values in the **Target Coordinates** group will be filled with the current robot pose. Manual Entry: the values in the **Target Coordinates** group will be manually entered by the user.
2	Target Type dropdown	Selects whether the target is a Cartesian target or a Joint target. Joint: The target will be defined in joint space. The coordinates entered in the **Target Coordinates** group will be interpreted as joint positions. Cartesian: The target will be defined in Cartesian space. The coordinates entered in the **Target Coordinates** group will be interpreted as the pose of the flange or tool relative to the reference frame specified in the **Ref Frame** dropdown.

Item	Name	Description
3	Move Type dropdown	Selects the desired interpretation for the target coordinates. Absolute: The move coordinates defined in the **Target Coordinates** group will be interpreted as displacements from an absolute (zero) reference. For joint targets, the absolute reference is the joint home position; for Cartesian targets, the absolute reference is the user-defined reference frame specified in the **Ref Frame** dropdown. Incremental: The move coordinates defined in the **Target Coordinates** group will be interpreted as relative displacements from the current robot pose or previous target.
4	End of Arm dropdown	Specifies the end-effector frame. The available choices are **Flange** or **Tool.**
5	Ref Frame dropdown	Specifies the reference frame for Cartesian targets. The available frame choices are **World**, **Robot**, **Flange**, **Tool**, **User**.
6	Target Coordinates group	For Cartesian targets, the group will consist of pose coordinates. The number of fields visible here will be dependent on the robot geometry type:  For Joint targets, the group will consist of joint position coordinates. The number of fields visible here will be dependent on the robot geometry type: 

10.1.2.2 Move Path Parameters

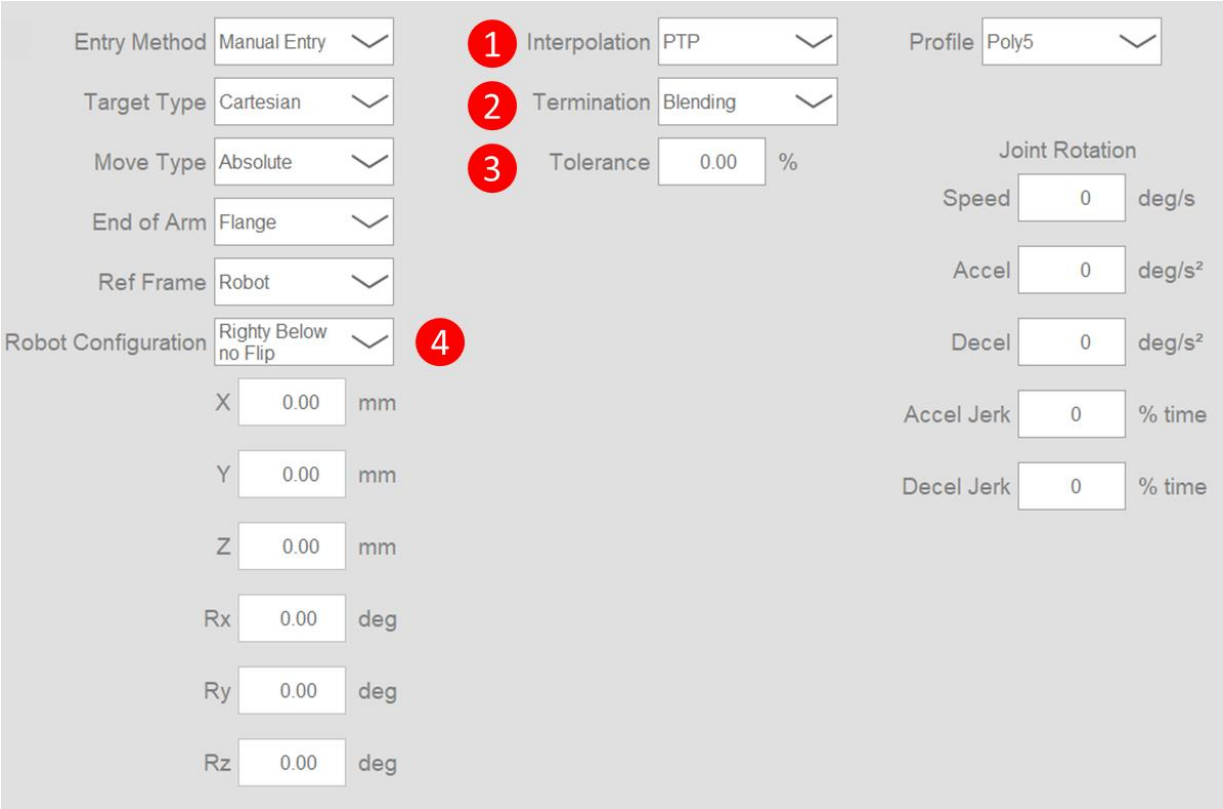


Figure 17- Move Path parameters.

Item	Name	Description
1	Interpolation dropdown	Selects the method for interpolating the path from the current robot pose or previous target to the new target. PTP (Point-to-Point): Used for general movements in which the destination is relevant, but the path shape is irrelevant. Motion planner will try to generate the fastest move to target in joint space. The resulting path in Cartesian space is unpredictable and generally not the shortest. Special attention must be paid to avoid collisions when moving in narrow spaces. All the joints will begin moving and stop moving at the same time, such that all axes adapt to the slowest axis. One advantage of PTP moves is their ability to move through singularities without issues. Only PTP moves are allowed when transforms are disabled. CP-L (Continuous Path - Linear): Used for fine movements in which both the destination and path shape are relevant. Robot will move to target following a predictable straight-line path in Cartesian space, while also controlling the end-effector orientation. The position and orientation are interpolated such that the final compound movement starts and ends at the same time. The resulting path is generally the shortest, but total move duration tends to be greater than PTP. Continuous path cartesian linear moves are usually appropriate for high precision tasks and limited spaces susceptible to collisions, but special attention must be paid to avoid moving through singularities. CP-W (Continuous Path - Wrist): Used for fine movements in which both the destination and path shape are relevant. This type of motion results in the end effector (tool) moving in a straight line while the wrist joints rotate to maintain a desired orientation. The resulting path is usually the shortest, but total move duration tends to be greater than PTP. Continuous path wrist moves are used when it's desired to move the tool in a straight line at a constant speed while the wrist joints rotate to maintain the desired tool orientation along the total path.

Item	Name	Description
2	Termination dropdown	Defines whether blending is used to connect two consecutive moves. Considering that the robot is at A, will move from A to target B (move AB), and finally from B to C (move BC); the dropdown offers two options to define the blending of moves AB and CD: Stop: Blending is disabled. The robot will stop at the target. The resulting motion will be a movement from A to B with a complete stop in point B, hence followed by the movement from B to C which will start from B with zero speed. This termination type is adequate for the first and last targets of a trajectory, and for any intermediate targets that must be accurately reached as part of a process requirement (e.g., picking parts, welding points, etc.). Note that disabling blending for all the intermediate targets without explicit requirements may introduce discontinuous trajectories and unnecessary acceleration/braking cycles. Blending: Blending is enabled. The robot will approach the target without stopping on it. The resulting motion will be a trajectory which doesn't pass through position B, and it won't stop during the transition from the first movement to the next. This results in a total movement (AB + BC) which can be significantly faster. To keep a smooth trajectory towards the target, the path planner will apply a blending radius according to a pre-defined tolerance (see parameter **Command Tolerance**). Not stopping at intermediate targets may reduce the total duration and improve the smoothness of the trajectory. Additionally, smoother moves tend to put less effort on the drives and transmissions, increasing the lifespan of the robot.
3	Command Tolerance control	Blending radius from previous target or current pose to next target. Used when **Termination Type** is Blending. This parameter can be used to control the amount of blending between consecutive moves and will reflect how much the trajectory is modified by blending. For CP-L or CP-W moves, the value is in mm and must be greater than zero. For PTP moves, value is in % of total move distance and must be between 0% and 100%.
4	Robot Configuration dropdown	Defines the preferred robot configuration from the set of potential solutions capable of reaching the target pose. In general, more than one set of joint positions may correspond to the same position and orientation of the tool in Cartesian space; hence, for PTP moves only, the robot configuration must be entered. Same: keep the previous target configuration. Remaining choices: combinations of lefty/righty, above/below, and flip/no-flip. The configurations of all supported robot geometries are described in the Rockwell Automation publication MOTION-UM002-EN-P.

IMPORTANT

The parameters **Target Type** and **Interpolation** are independent. That means a Cartesian target can be reached by either Point-to-Point (PTP), Continuous Path- Linear (CP-L), or Continuous Path- Wrist (CP-W) interpolated paths, and the same rule applies for joint targets. However, if a Cartesian target is specified for a PTP move, it will be first converted to a joint target, since PTP moves are always calculated in joint space. In a similar fashion, if a joint target is specified for a CP-L (or CP-W) move, it will be first converted to a Cartesian target, since CP-L and CP-W moves are always calculated in Cartesian space.

10.1.2.3 Move Timing Parameters

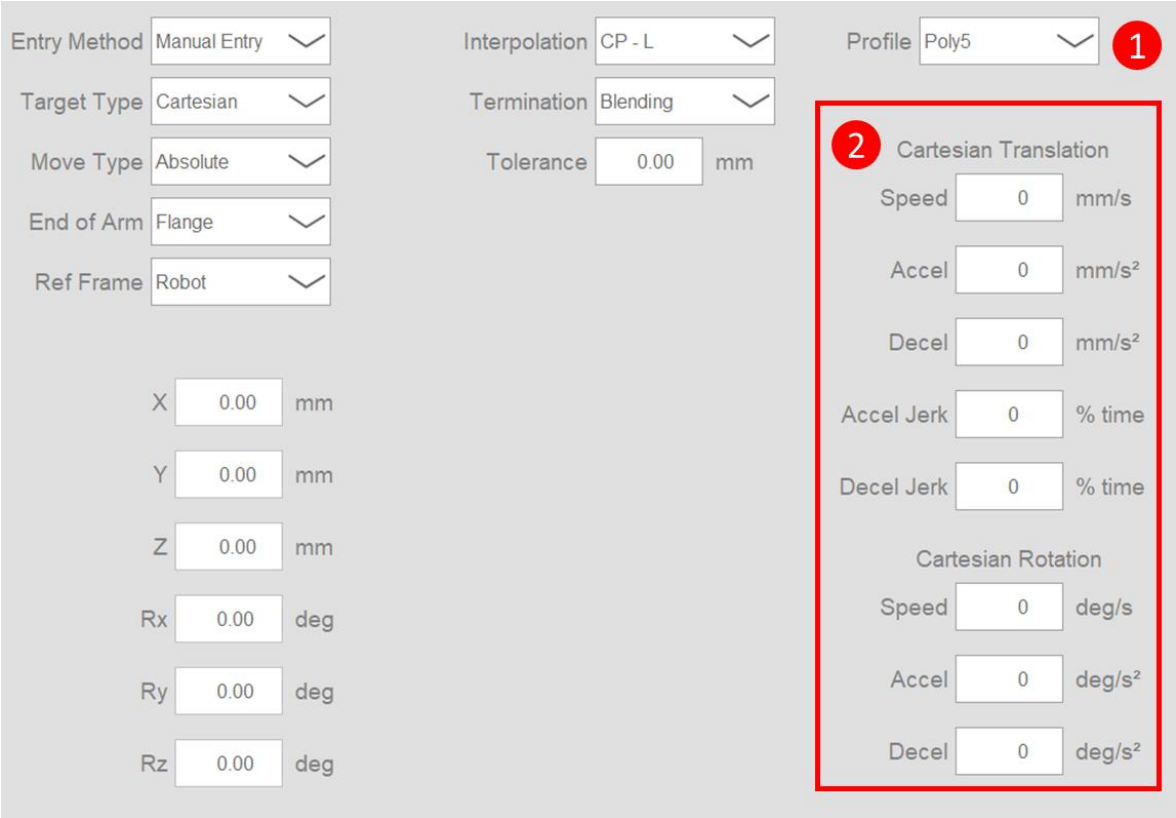
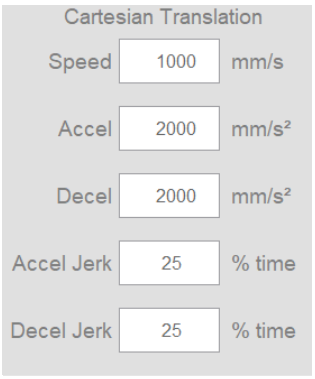
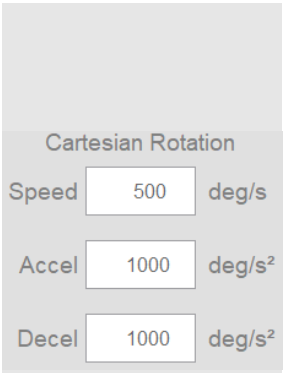
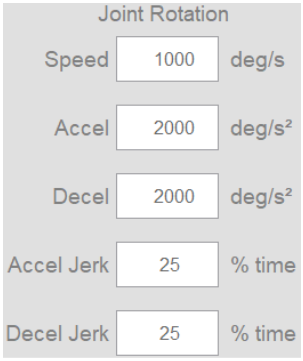


Figure 18- Move Timing parameters.

Item	Name	Description
1	Profile Type dropdown	Motion profile function to be used by the path planner. Cubic: A profile that provides basic / traditional motion. Can provide fast motion but is the least smooth. Poly5 (fifth-order polynomial): Another basic profile. Improves smoothness over S-Curve. ModSine (modified sine): Provides the best balance of move time, smoothness, and energy. Lowest energy consumption. Sine^2 (sine squared): Smoothest at the cost of slightly longer move times. **NOTE:** The above comparison statements assumes acceleration is an input and is kept constant while generating the profiles.

Item	Name	Description
2	Move Dynamics group	For Cartesian targets, group will consist of: • Cartesian Translation dynamics parameters: ○ Speed: end-effector linear speed limit ○ Accel: end-effector target linear acceleration ○ Decel: end-effector target linear deceleration ○ Accel Jerk: end-effector target linear acceleration jerk ○ Decel Jerk: end-effector target linear deceleration jerk • Cartesian Rotation dynamics parameters: ○ Speed: end-effector angular speed limit ○ Accel: end-effector target angular acceleration ○ Decel: end-effector target angular deceleration ○ Jerk: end-effector target angular jerk   For joint targets, group will consist of: • Joint Rotation dynamics parameters: ○ Speed: joint speed limit ○ Accel: joint target acceleration ○ Decel: joint target deceleration ○ Accel Jerk: joint target acceleration jerk ○ Decel Jerk: joint target deceleration jerk  **NOTE:** Values must be greater than zero for the first move. For all the remaining moves, zero means “keep previous value”. Jerk can be configured for any profile from 0-100% of time. Increasing this value trades off slightly slower move times for considerably lower jerk. Higher is better for smoothness and vibrations. - 100% Jerk: Default setting (recommended), with continuously varying acceleration. - 1-99% Jerk: Continuously varying acceleration at the beginning and end of the move with constant acceleration in the middle. - 0% Jerk: Constant acceleration, generating a Trapezoidal profile. Trapezoidal moves are not recommended as they impose high stress on electrical and mechanical components.

10.1.3 Move Management Group

The **Move Management** group (Figure 19) enables managing the currently selected move.

Figure 19- Move Management group.

Item	Name	Description
1	Save button	Keep the changes to all the current parameters of the selected move. Button will become enabled if any parameter is changed and not saved.
2	Copy button	Takes a snapshot of all the current parameters of the selected move.
3	Paste button	Writes all the currently copied parameters to the selected move.
4	Clear button	Resets all the parameters of the selected move to the default “zero” value.
5	Insert Current Pose Data	When **Entry Method** is Current, copies the current pose data to the target coordinate fields.

10.2 Defining Moves via Manual Entry

10.2.1 Purpose

When the move target coordinates are known beforehand, it can be entirely defined offline, without moving the robot.

10.2.2 Preconditions

- **Trajectory** page is opened.

10.2.3 Procedure

Step	Description
1	Click on the  and  buttons of the **Trajectory Navigation** group to select the desired move to be defined.
2	Expand the **Target Type** dropdown and select **Joint** or **Cartesian**.
3	Expand the **Entry Method** dropdown and select **Manual Entry**.
4	Expand the **Move Type** dropdown and select whether the target coordinates will be absolute or incremental.
5	Go to the **Target Coordinates** group and type the desired joint positions or Cartesian pose.

Step	Description
6	Expand the **Interpolation** dropdown and select the desired path interpolation method.
7	Expand the **Termination** dropdown and select whether blending is enabled or disabled.
8	If the selected **Termination** was **Blending**, click on the **Tolerance** control and select the desired blending radius.
9	If the selected **Target Type** was **Cartesian** and the selected **Interpolation** was **PTP,** expand the **Robot Configuration** dropdown and select the desired robot configuration.
10	Expand the **Profile** dropdown and select the desired motion profile function.
11	Go to the **Move Dynamics** group and configure all the desired joint or Cartesian translation and rotation dynamics parameters.
12	Click on the **Save** button to store all the parameters of the selected move.

10.3 Defining Moves via Jogging

10.3.1 Purpose

The most common method of programming robot moves is by manually jogging the robot to a target, “teaching” the current robot coordinates, and then use the taught coordinates as a destination for a move.

10.3.2 Preconditions

- **Trajectory** page is opened.
- Robot is not faulted.
- Robot is available.
- Robot is energized.
- Current operating mode is **Manual Reduced Speed**.

10.3.3 Procedure

Step	Description
1	Click on the  and  buttons of the **Trajectory Navigation** group to select the desired move to define.
2	Expand the **Target Type** dropdown and select **Joint** or **Cartesian**.
3	Expand the **Entry Method** dropdown and select **Current.** The **Insert Current Pose** icon will change shade and be enabled. 
4	Jog the robot to the desired target Jogging .
5	Click on the **Insert Current Pose** icon. The current robot position will be copied to the **Target Coordinates** group as a set of joint positions or a Cartesian pose.
6	Expand the **Interpolation** dropdown and select the desired path interpolation method.
7	Expand the **Termination** dropdown and select whether blending is enabled or disabled.
8	If the selected **Termination** was **Blending**, click on the **Tolerance** control and select the desired blending radius.
9	If the selected **Target Type** was **Cartesian** and the selected **Interpolation** was **PTP,** expand the **Robot Configuration** dropdown and select the desired robot configuration.

Step	Description
10	Expand the **Profile** dropdown and select the desired motion profile function.
11	Go to the **Move Dynamics** group and configure all the desired joint or Cartesian translation dynamics parameters.
12	Click on the **Save** button to store all the parameters of the selected move.

10.4 Copying Moves

10.4.1 Purpose

While defining a move sequence, it may be convenient to copy all the parameters of an existing move to another move, especially when only a few parameters differ.

10.4.2 Preconditions

- **Trajectory** page is opened.

10.4.3 Procedure

Step	Description
1	Click on the  and  buttons of the **Trajectory Navigation** group to select the desired move to copy parameters from.
2	Click on the **Copy** icon to copy all the parameters from the selected move. 
3	Click on the  and  buttons of the **Trajectory Navigation** group to select the desired move to paste parameters to.
4	Click on the **Paste** icon to get all the parameters from the copied move. Check if the move parameters reflect the correct values. 

10.5 Clearing Moves

10.5.1 Purpose

The quickest way to start a move definition from scratch is to select a move index and reset all the parameters to their default value.

10.5.2 Preconditions

- **Trajectory** page is opened.

10.5.3 Procedure

Step	Description
1	Click on the  and  buttons of the **Trajectory Navigation** group to select the desired move to clear.
2	Click on the **Clear Icon** remove the selected move from the trajectory. 

10.6 Testing and Executing Trajectories

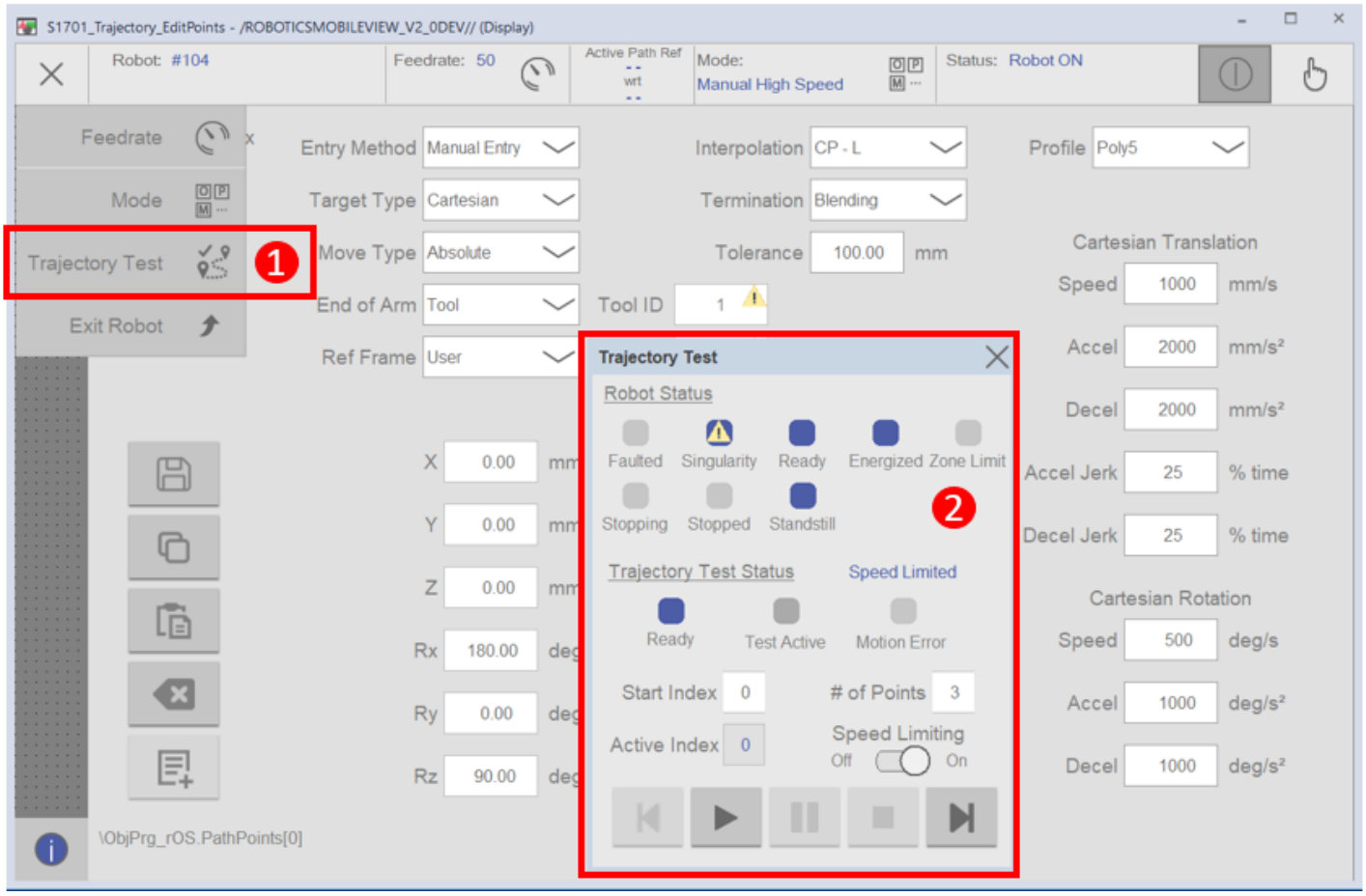
10.6.1 Trajectory Test

After completing the parameterization of the HMI-defined trajectory sequence, a **Trajectory Test** widget is accessible from the Trajectory screen that provides the ability to test individual targets or a series of targets along the defined trajectories.

For illustration purposes, this section provides a quick overview on the basic procedure to test trajectories defined with the **Trajectory** page.

10.6.1.1 Trajectory Test Widget

The Trajectory Test widget is accessible from the Context Menu button on the status bar. This control allows the user to test individual or a series of path points that have been configured on the Trajectory Edit page. It provides the ability to step through the path on a point-by-point basis or execute the entire series of path points. In addition, it provides the ability to temporarily modify any single point along the path and test the path prior to saving it as a permanent target in the path array tag.



Item	Name	Description
1	Trajectory Test Popup button	Displays the Trajectory Test popup.
2	Trajectory Test Popup	Trajectory Test interface.

10.6.1.2 Preconditions

- **Trajectory Edit** page is open.
- Moves have been defined using the **Trajectory Edit** page.
- Robot is not faulted.
- Robot is available.
- Robot is energized.
- Current operating mode is **Manual High Speed**.

10.6.1.3 Trajectory Test Pop-Up

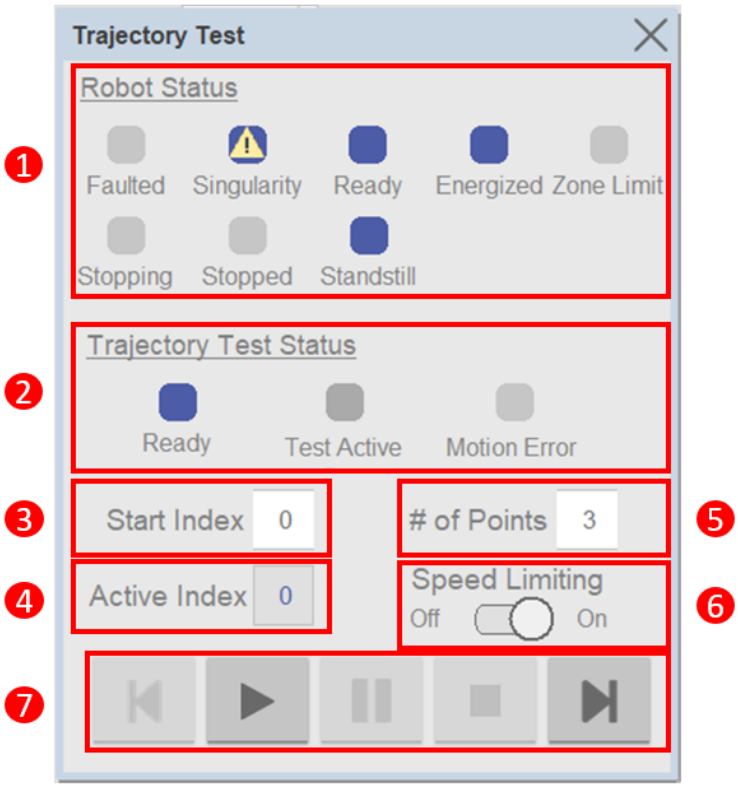


Figure 20 - Trajectory Test Pop-up

Item	Name	Description
1	Robot Status	Displays general robot status
2	Trajectory Test Status	Trajectory test execution status
3	Path Point Start Index	The initial defined point to begin path execution
4	Active Index	The path point currently loaded and being executed. A '-1' indicates no path points loaded.
5	# of Points	The number of path points to execute starting from (and including) the Start Index.
6	Speed Limiting	When enabled, limits the trajectory execution speed. When disabled, the trajectory executes with the programmed path dynamics.
7	Trajectory Test Control	Manages the execution of the path points

10.6.1.4 Trajectory Test Control

Figure 21 - Trajectory Test Control

Item	Name	Description
1	Execute Previous Path Point	When enabled, allows stepping back one step in the path sequence
2	Execute All Selected Path Points	Execute the entire path sequence from start index
3	Pause on Path	Pause on path. Select again to resume path sequence
4	Stop Sequence / Clear Fault	Terminate the path sequence. If the Trajectory Test fails, this control will reset the fault.
5	Execute Next Path Point	When enabled, allows stepping forward one step in the path sequence

10.6.1.5 Procedure

Step	Description
1	From the Trajectory Edit screen, configure your desired path points as shown on Trajectory Page
2	Ensure the robot is Available and not Faulted then switch the mode to **Manual High Speed** **Selecting the Operating Mode**
3	Select whether Speed Limiting is desired. Note: Speed Limiting enabled is the default condition anytime this popup is opened
4	Enter the desire **Start Index**. This will be the first target point in your configured trajectory.
5	Enter the **# of Points**. This will be the total number of points in your trajectory that you want to sequence through.
7	Select the Execute icon to cycle through your entire trajectory or the Step Forward (or Step Back) to execute a single step.

IMPORTANT

A trajectory test can be paused or stopped at any point in the sequence. Once paused, the trajectory can be resumed or stopped. After a stop, the sequence is cleared, and subsequent operations will start from the defined Start Index path point. If an error occurs during the test, the Motion Error indicator illuminates, and the error must be reset before attempting further trajectory test functions.

10.6.2 Trajectory Execute

After parameterization, the HMI-defined trajectory sequence is accessible as a public path array tag that can be consumed by user programs containing an instance of the **raM_Robot_Opr_LoadPath** instruction. The user is fully responsible for setting up the instruction instances, tag connections, and programming logic to carry out this use case. This is the same data array used by the **Trajectory Test** widget described previously.

For illustration purposes, this Section provides a quick overview on the basic required steps to program and execute trajectories defined with the **Trajectory** page.

10.6.2.1 Procedure

Step	Description
1	In Logix Designer, create a new task, a new program, and a new routine.
2	Add an instance of **raM_Robot_Opr_LoadPath** to the routine and set up the rung similarly to the rung depicted in Figure 22.
3	Connect the Device Handler **[RobotName]_DH.Hndl** tag to the input **Ref_Handle**.
4	Connect the tag **\ObjPrg_rOS.PathPoints** to the input **Ref_Path**. This is the tag containing the HMI-defined trajectory, also displayed on the bottom left area of the **Trajectory** page (**Move Tag** indicator).
5	Configure the start move index with the input **Cfg_StartIndex**, and the desired number of moves with the input **Cfg_NumberOfMoves**.
6	The trajectory will be executed when the rung-in condition of the instruction is true.

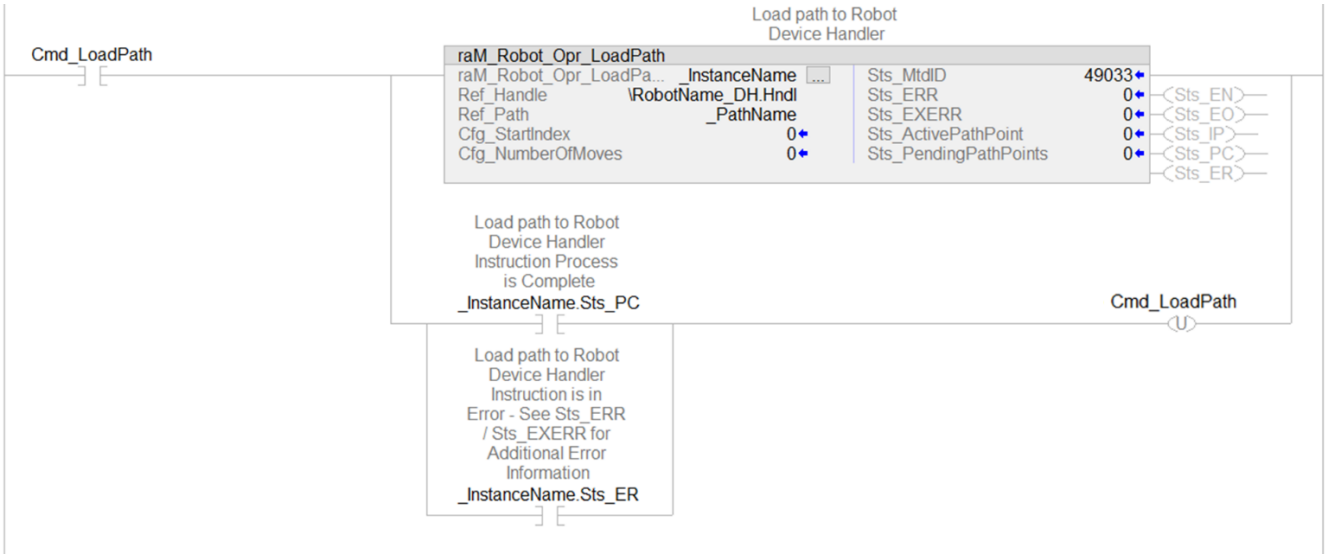


Figure 22 - Sample logic for executing HMI-defined trajectories

IMPORTANT

User code execution can only be performed with the mode set to **Automatic External** [Selecting the Operating Mode](#).

11 Configuration

This Chapter describes the **Configuration** page and related functions.

11.1 Configuration Page

The **Configuration** page (Figure 23) has dedicated controls for calibrating motors and controlling motor brakes.

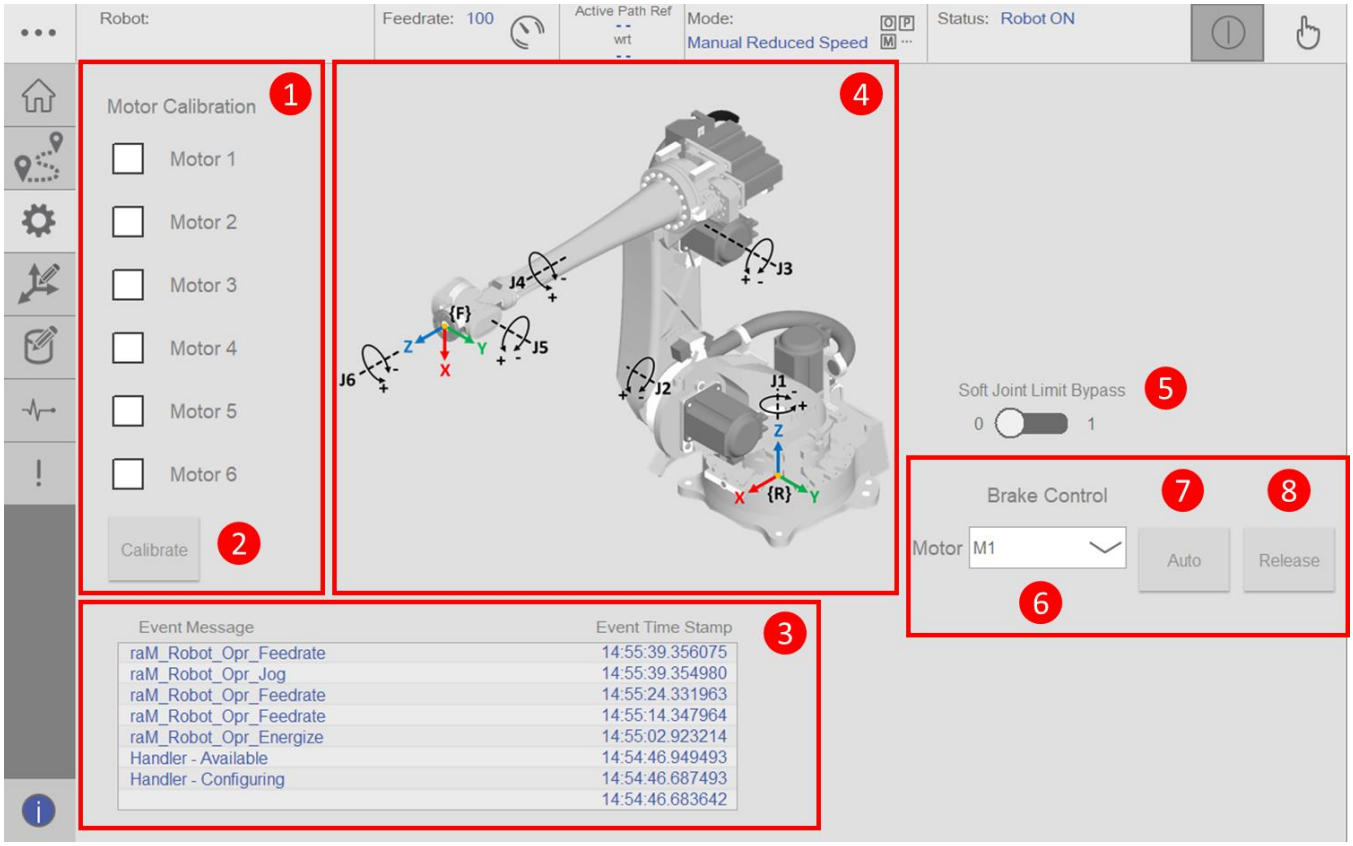


Figure 23- The Configuration page.

Item	Name	Description
1	Motor Calibration Selection group	Selects the motor axes to calibrate(home).
2	Calibrate Motors button	Starts the motor home calibration procedure.
3	Event list	Displays the latest timestamped events.
4	Robot image	Displays the current robot geometry in use, basic frames, and joint directions.
5	Soft Joint Limit Bypass switch	Selects whether joint soft limit checking is enabled or disabled.
6	Brake Selection dropdown	Selects the motor brake to be managed.
7	Auto Brake button	Sets the selected motor brake control mode to Automatic.
8	Release Brake button	Sets the selected motor brake control mode to Manual and commands it to release.

11.2 Calibrating Axes

11.2.1 Purpose

A robot must provide correct motor position feedback to be properly controlled. The procedure for establishing the home position for each axis and the strategies for keeping the positions valid across power cycles are manufacturer and technology dependent. It may involve aligning reference marks visually, moving towards limits, or other more sophisticated methods. Ultimately, the goal is to assure that “virtual” joint positions match the “real” joint positions.

When position sensors are incorrectly calibrated, the robot precision and accuracy are directly affected. If the robot is fully uncalibrated, bigger problems arise:

- The calculated joint positions become unreliable.
- Kinematic calculations cannot be performed, meaning that any Cartesian space command or state is invalid.
- There is no reliable way to compare actual joint positions against limits, so joint soft limit checks must be disabled. Visual inspection and mechanical stops are the only options.
- The only “safe” way of moving the robot is by incrementally moving each joint.

The calibration process may be required in many different cases, depending on the specific robot model, technology, and circumstances:

- When the robot is being commissioned.
- When the robot has been kept powered off for long periods.
- When the robot has been moved while powered off.
- When some actuator has been repaired or replaced.
- When the precision/accuracy of moves is clearly degraded and below requirements.
- When a mechanical joint limit has been reached at high speeds.
- When the robot is subjected to severe collisions.

The **Configuration** page offers a unified calibration procedure that applies to all the currently supported robot models. The process consists of manually lining up calibration marks, setting up the home calibration value for each axis, and finally triggering axis homing.

11.2.2 Preconditions

- **Configuration** page is opened.
- Robot is not faulted.
- Robot is available.
- Robot is not energized.
- Current operating mode is **Manual Reduced Speed**.

11.2.3 Procedure

Step	Description
1	Each axis to be calibrated must be pre-positioned so that the physical calibration marks on the axes are near alignment. Move the axes to their designated calibration positions using joint jogging or manual brake release.
2	Select the desired motor axes to calibrate on the **Motor Calibration Selection** group.
3	Press the **Calibrate Motors** button and wait for the calibration procedure to complete. If the calibration process was unsuccessful, faults will be listed in the **Event** list.

Although methods can be employed to detect that the position feedback is inconsistent, the user is ultimately responsible for assuring the calibration is valid. If there is any suspicion that any of the motors is uncalibrated, execute the calibration procedure again.

11.3 Managing Brakes

11.3.1 Purpose

Robot motors are equipped with electro mechanical brakes to reduce the likelihood of unintentional moves under power failure events or extended downtime. The brakes should be capable of applying sufficient holding torque on joints to maintain the robot standstill. Sometimes brakes are also designed for dynamically stopping a joint when an emergency occurs.

Motor brakes are engaged by default, because the fail-safe state is to automatically hold the robot when power is lost. To manually release the brake, an explicit command signal must be sent. This protection is especially important for gravity affected joints since heavy parts might fall and cause severe injuries and damage.

Although brakes should be usually controlled and kept in automatic mode, some emergency or special situations may require manual release. For example:

- If a severe collision occurred, and the robot is stuck and unable to be moved out of the collision area.
- During the calibration process, users may want to manually move the joints to line up their calibration marks. It is unlikely that this will be a reasonable operation for high-payload robots, although it may be a valid use case for small robots with less “gravity sensitive” joints (e.g., a SCARA).

The **Configuration** page offers controls for manually disengaging brakes and switching their control mode back to Automatic.

11.3.2 Preconditions

- **Configuration** page is opened.
- Robot is not faulted.
- Robot is available.
- Robot is not energized.
- Brakes are not being commanded by another internal instruction.
- Current operating mode is **Manual Reduced Speed**.

11.3.3 Procedure

Step	Description
1	Ensure the robot is supported externally before releasing the brake on any axis.
2	Expand the **Brake Selection** dropdown and select the desired motor brake to release.
3	Click on the **Release Brake** button to disengage the selected brake.
4	When the intended operation is complete, switch back the selected brake control mode to automatic by clicking on the **Auto Brake** button.

ATTENTION

In general, manual brake release should be regarded as an extremely careful operation. Motor brakes should be released one at a time, after properly securing the robot parts against free fall. If a motor brake is disengaged and the robot is not externally supported, heavy robot parts may suddenly drop and cause injury and/or equipment damage.

IMPORTANT

If brakes are manually released and the robot is re-energized, all motor brakes switch back to Automatic.

12 Frames

In robotics, frames play a crucial role in defining the robot's spatial awareness and movement. They act like coordinate systems, providing a reference point for the robot to understand its position, orientation, and the location of objects in its environment. Frames simplify robot programming by providing a structured and organized way to represent the robot's world, making programming tasks more intuitive and efficient. By specifying points and trajectories within different frames, programmers can instruct the robot to move its arm or navigate its environment when planning movements.

12.1 Frames Page

The **Frames** page has dedicated controls for creating and teaching different reference frames for the robot to operate in. Three frame types can be configured in this environment. The **Robot Frame** serves as the starting point for all other frames and is typically attached to the robot's base. The **Tool Frame** is attached to the end of the robot's arm. It defines the position and orientation of the tool relative to the robot's base. The **User Frame** is a frame that can be attached to any object or location in the robot's workspace. It's helpful for defining specific points of interest or aligning the robot with objects.

IMPORTANT

While the convenient HMI interface offers frame creation and configuration, advanced users can directly craft customized frames using the **raM_Robot_Opr_TeachToolFrame, raM_Robot_Opr_TeachUserFrame,** and **raM_Robot_Opr_SetFrame** instructions directly in their code. However, be aware that this code-based approach can overwrite frame data set up through the HMI.

12.1.1 Robot Frame

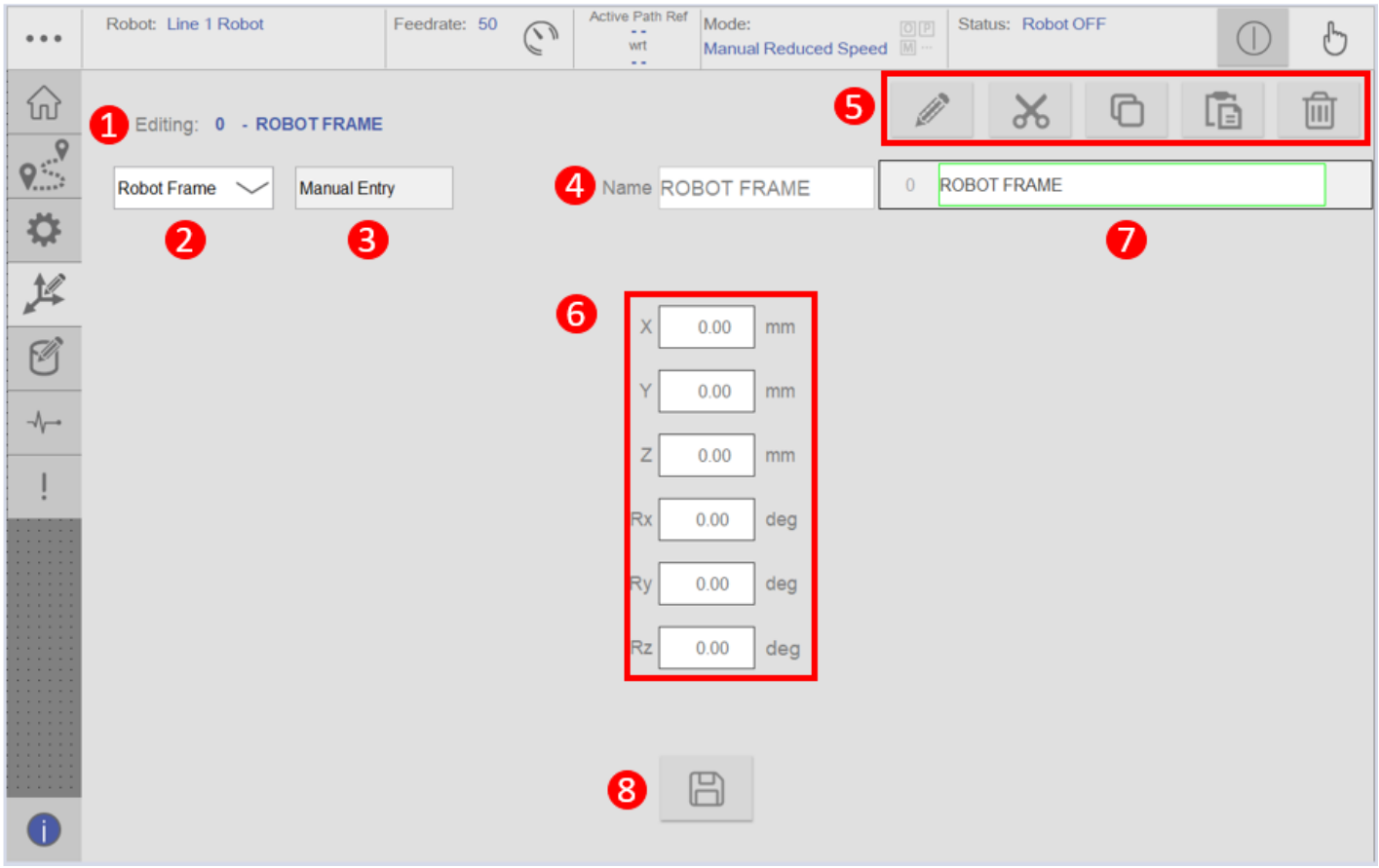


Figure 24- The Robot Frame.

Item	Name	Description
1	Frame Selected	Shows the Frame currently selected for editing
2	Frame Type Dropdown	Selects the type of Frame. Select Robot Frame
3	Frame Definition Dropdown	Selects how you will define the Frame: Manual entry or Teach. Currently for **Robot Frame**, manual entry is the only option
4	Frame Name Entry	Allows you to enter or change the name of the selected frame. **Robot Frame** is the default name
5	Frame Management Group	Provides functions to edit, cut, copy, paste, and delete Frames. Since there is only one **Robot Frame**, these options are diminished in this instance.
6	Frame Coordinates	Allows you to enter/edit the position and orientation of the Robot Frame.
7	Frame Storage Database	The Robot Frame only has one database entry
8	Save button	When editing the Robot Frame, saves the current coordinate data to the **Robot Frame** database.

12.1.1.1 Procedure

Step	Description
1	Select **Robot Frame** using the Frame Type dropdown
2	Manual Entry is the only option in the Frame Definition dropdown.
3	Type a name in the Frame Name entry box or leave as default
4	Type in the coordinates of the origin and orientation in the Frame Coordinates boxes
5	Press the Save button

12.1.2 User Frame

User Frames are defined either by manually entering the origin and orientation for the frame (if known) or by a 3-point acquisition procedure. A 3-point acquisition procedure is a common method for defining a user frame in robotics. It involves measuring the positions of three distinct points within the user frame and using these measurements to calculate the frame's origin and orientation.

12.1.2.1 User Frame Manual Entry

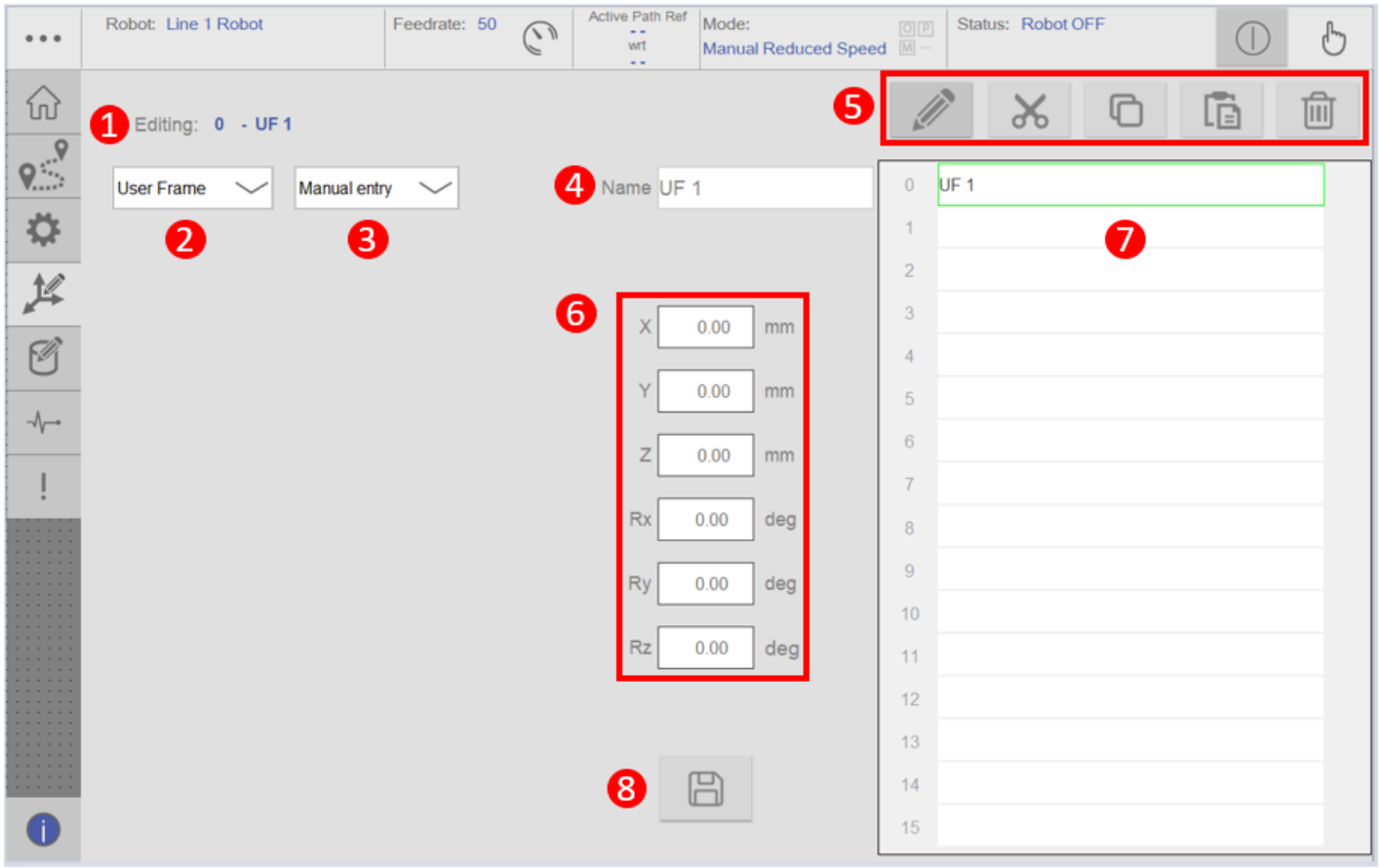


Figure 25 - User Frame Manual Entry

Item	Name	Description
1	Frame Selected	Shows the **User Frame** currently selected for editing
2	Frame Type Dropdown	Select the type of Frame. Select **User Frame**
3	Frame Definition Dropdown	Selects how you will define the Frame: Select **Manual entry**
4	Frame Name Entry	Allows you to enter or change the name of the selected **User frame**.
5	Frame Management Group	Provides functions to edit, cut, copy, paste, and delete Frames. Note that to edit the **User Frame** database, the robot must be Off.
6	Frame Coordinates	Allows you to enter/edit the origin and orientation of the **User Frame**.
7	Frame Storage Database	Up to 16 unique **User Frames** can be configured
8	Save button	When editing the **User Frame**, saves the current coordinate data to the **User Frame** database.

12.1.2.2 Procedure

Step	Description
1	Select **User Frame** using the Frame Type dropdown
2	Select Manual Entry using the Frame Definition dropdown.
3	Select an available Frame in the User Frame storage database table. The database number corresponds to the User Frame ID.
4	Type a name in the Frame Name entry box
5	Type in the coordinates of the origin and orientation in the Frame Coordinates boxes
6	Press the Save button

12.1.2.3 User Frame – 3 Point Acquisition

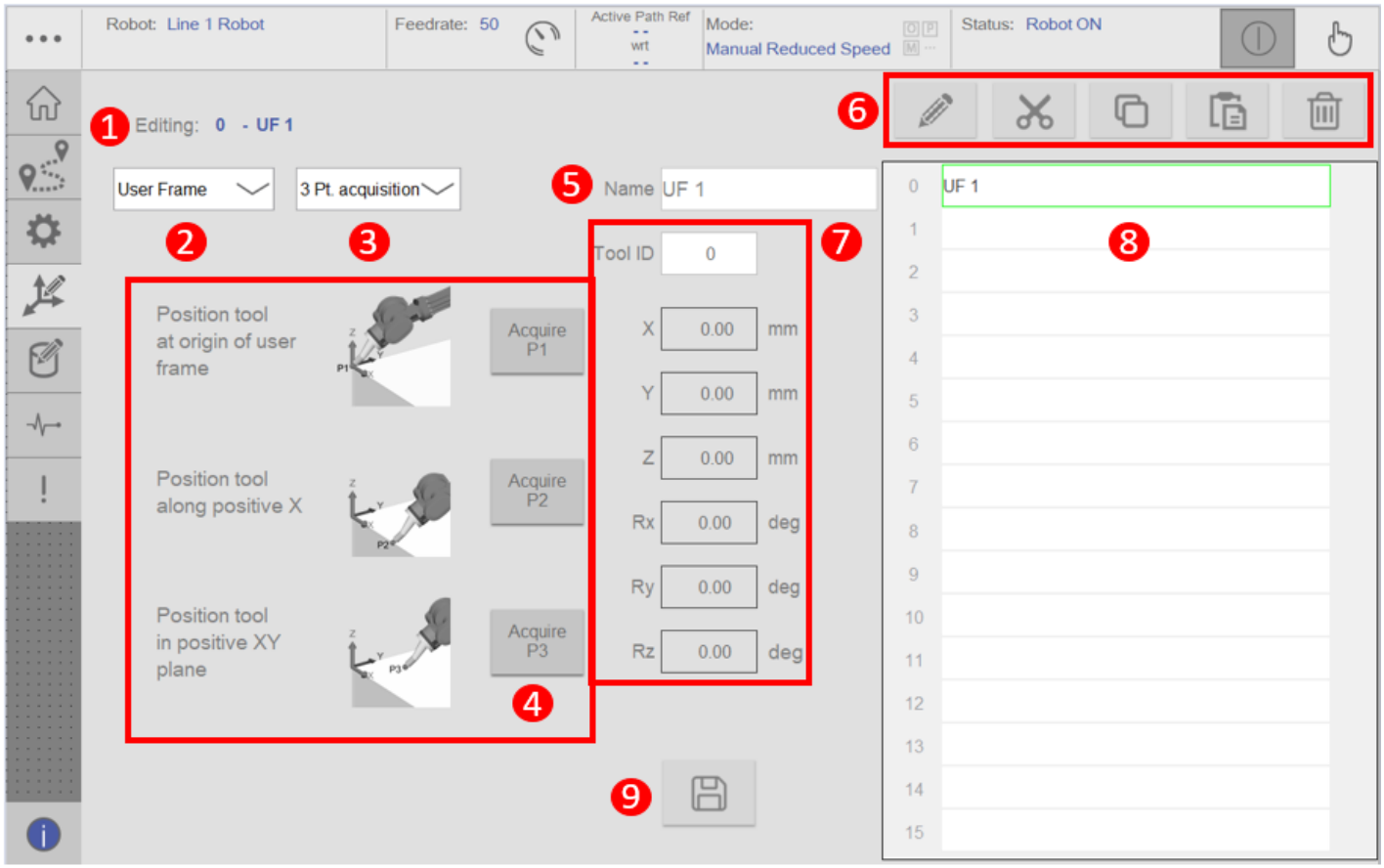


Figure 26 - User Frame 3 Point Teach

Item	Name	Description
1	Frame Selected	Shows the **User Frame** currently selected for editing
2	Frame Type Dropdown	Select the type of Frame. Select **User Frame**
3	Frame Definition Dropdown	Selects how you will define the Frame: Select **3 Pt. acquisition**
4	3 Point Teach Procedure	Sequence of required events to teach the **User Frame**
5	Frame Name Entry	Allows you to enter or change the name of the selected **User frame**.
6	Frame Management Group	Provides functions to edit, cut, copy, paste, and delete Frames. Note that to edit the **User Frame** database, the robot must be Off.
7	Frame Coordinates	In the 3-point acquisition mode, these fields are automatically populated with the calculated position and orientation using the configured tool after the three points are successfully acquired.
8	Frame Storage Database	Up to 16 unique **User Frames** can be configured
9	Save button	Once the **User Frame** position has been successfully calculated, saves the current coordinate data to the **User Frame** database.

IMPORTANT

A ⚠ symbol present in the Tool Frame ID box indicates that the Tool Frame selected has not been defined. For more information on the procedures for defining Tool Frames, see Section Tool Frame

IMPORTANT

A  symbol present on the Save icon indicates that the User Frame teach points were performed improperly and/or the robot is not in the correct state for saving the frame.

12.1.2.4 Procedure

Step	Description
1	Select User Frame using the Frame Type dropdown
2	Select **3 Pt. acquisition** using the Frame Definition dropdown.
3	Select an available Frame in the User Frame storage database table. The database number corresponds to the User Frame ID.
4	Type a name in the Frame Name entry box
5	Select the Tool being used to teach the Frame. A valid Tool ID must be used.
6	Jog the robot to the first position (P1) indicated by the picture on the screen. Press the **Acquire P1** button. Repeat the procedure as shown in the diagram for P2 and P3.
7	When the frame coordinates update, Press the Save button

12.1.3 Tool Frame

Tool Frames are defined either by manually entering the origin and orientation for the frame (if known) or by a 4-point acquisition procedure. A 4-point method is typically used for teaching the tool frame. This method utilizes four points located on the robot's tool (e.g., gripper, welding tool) to define its position and orientation relative to the robot's base frame.

12.1.3.1 Tool Frame Manual Entry

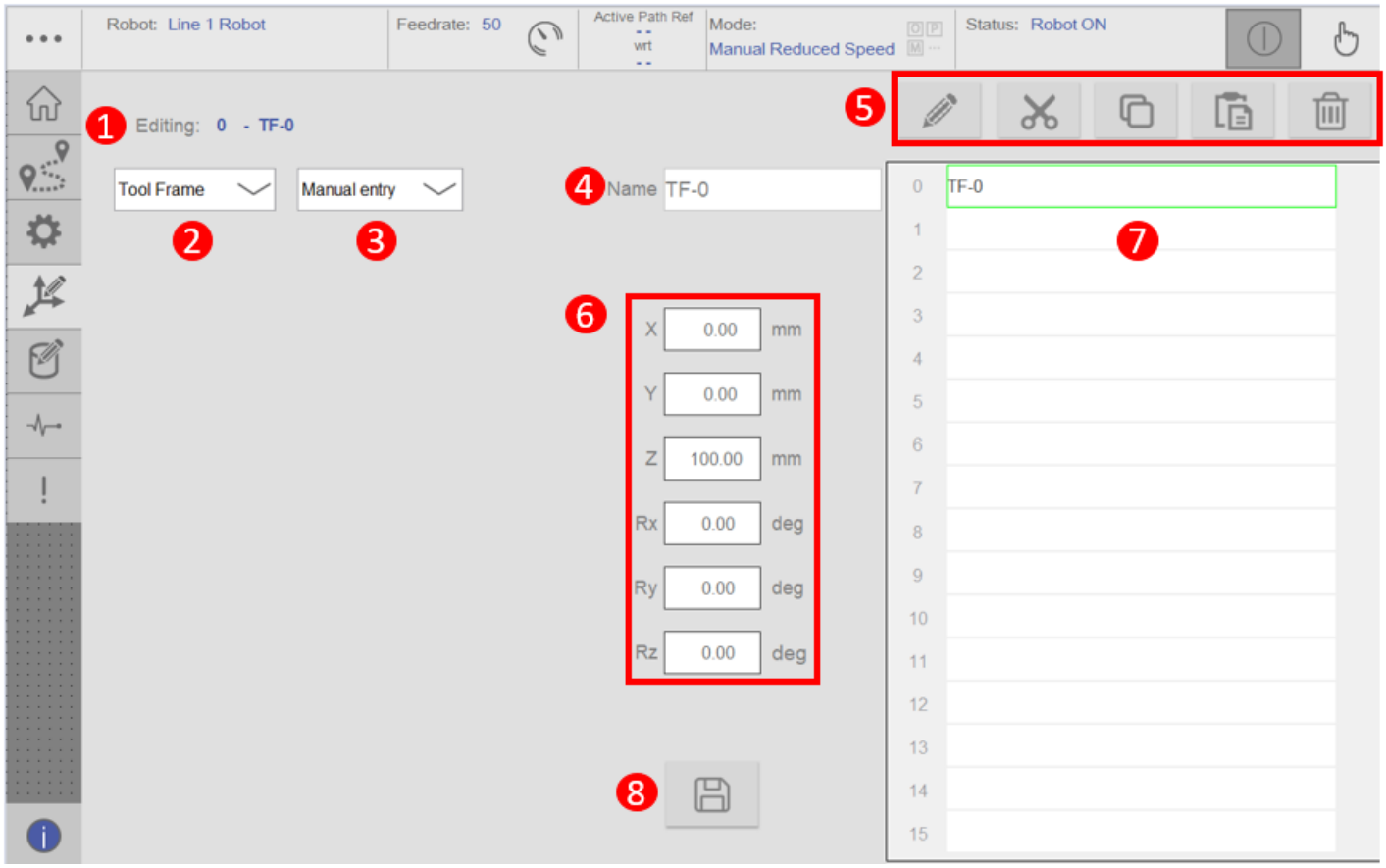


Figure 27 - Tool Frame Manual Entry

Item	Name	Description
1	Frame Selected	Shows the **Tool Frame** currently selected for editing
2	Frame Type Dropdown	Select the type of Frame. Select **Tool Frame**
3	Frame Definition Dropdown	Selects how you will define the Frame: Select **Manual entry**
4	Frame Name Entry	Allows you to enter or change the name of the selected **Tool frame**.
5	Frame Management Group	Provides functions to edit, cut, copy, paste, and delete Frames. Note that to edit the **Tool Frame** database, the robot must be Off.
6	Frame Coordinates	Allows you to enter/edit the origin and orientation of the **Tool Frame**.
7	Frame Storage Database	Up to 16 unique **Tool Frames** can be configured

Item	Name	Description
8	Save button	When editing the **Tool Frame**, saves the current coordinate data to the **Tool Frame** database.

12.1.3.2 Procedure

Step	Description
1	Select **Tool Frame** using the Frame Type dropdown
2	Select Manual Entry using the Frame Definition dropdown.
3	Select an available Frame in the **Tool Frame** storage database table. The database number corresponds to the Tool Frame ID.
4	Type a name in the Frame Name entry box
5	Type in the coordinates of the origin and orientation in the Frame Coordinates boxes
6	Press the Save button

12.1.3.3 Tool Frame – 4-Point Acquisition

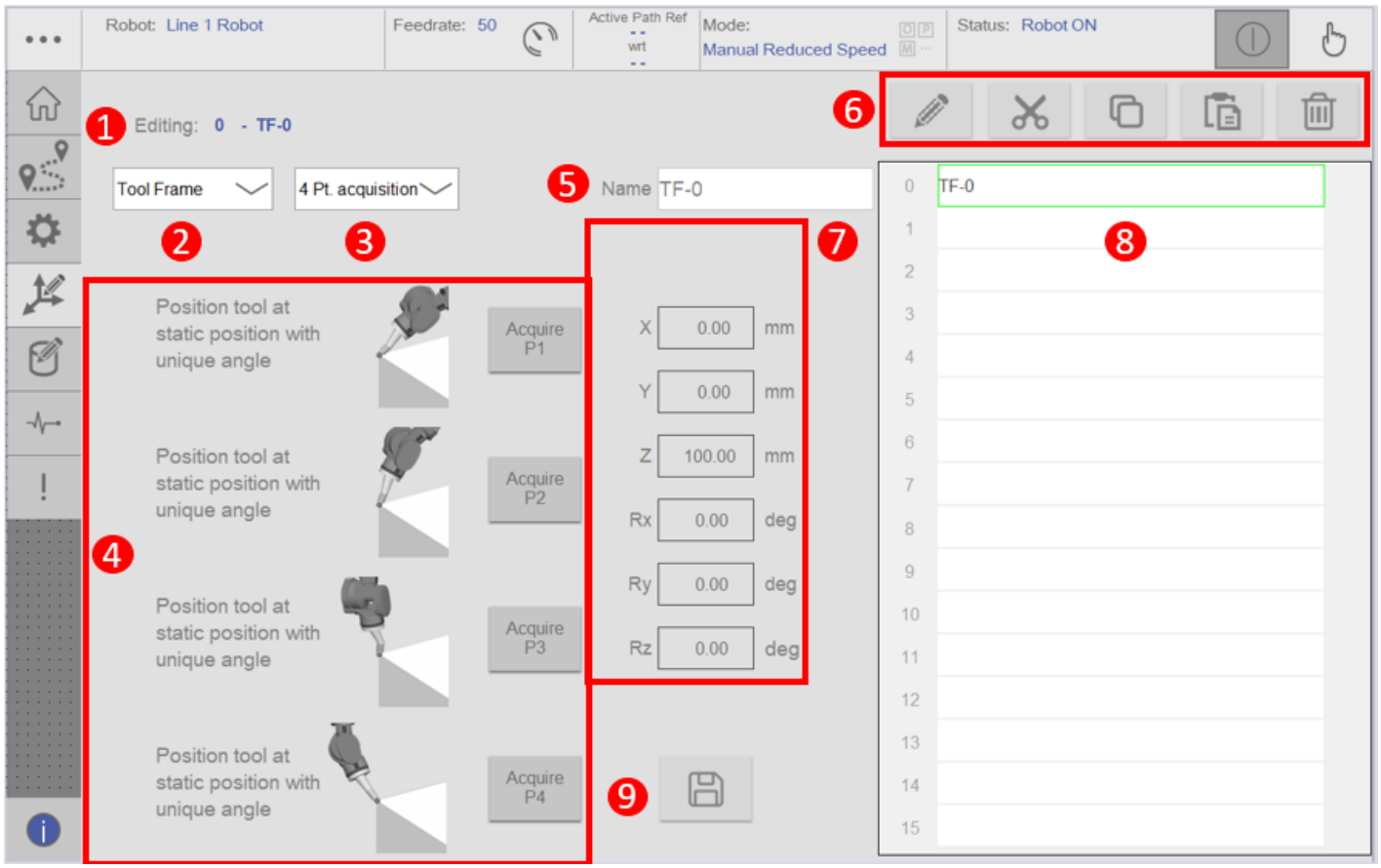


Figure 28 - Tool Frame 4 Point Teach

Item	Name	Description
1	Frame Selected	Shows the **Tool Frame** currently selected for editing

Item	Name	Description
2	Frame Type Dropdown	Select the type of Frame. Select **Tool Frame**
3	Frame Definition Dropdown	Selects how you will define the Frame: Select **4 Pt. acquisition**
4	4 Point Teach Procedure	Sequence of required events to teach the **Tool Frame**
5	Frame Name Entry	Allows you to enter or change the name of the selected **Tool frame**.
6	Frame Management Group	Provides functions to edit, cut, copy, paste, and delete Frames. Note that to edit the **Tool Frame** database, the robot must be Off.
7	Frame Coordinates	With the 4-point acquisition mode, these fields are automatically populated with the calculated position and orientation after the 4 points are successfully acquired.
8	Frame Storage Database	Up to 16 unique **Tool Frames** can be configured
9	Save Icon	Once the **Tool Frame** position has been successfully calculated, saves the current coordinate data to the **Tool Frame** database.

IMPORTANT A symbol present on the Save icon indicates that the Tool Frame teach points were performed improperly and/or the robot is not in the correct state for saving the frame.

13 Zone

Zones are crucial for ensuring robot safety and efficiency. Zones defined here provide indication that the tool or flange is venturing into unsafe areas or has the potential of colliding with objects. By defining and visualizing the work zone, programmers can optimize robot movements, plan collision-free paths, and ensure smooth operation within the cell.

IMPORTANT

While the convenient HMI interface offers zone creation and configuration, advanced users can directly craft additional zones using the **raM_Robot_Opr_ZoneBox, raM_Robot_Opr_ZoneCylinder, raM_Robot_Opr_ZoneSphere** instructions directly in their code. This code-based approach can be used to create independent zones without using or affecting the HMI generated zones.

13.1 Zone Configuration Page

The **Zone Configuration** page has dedicated controls for creating different Zones for the robot to operate in.

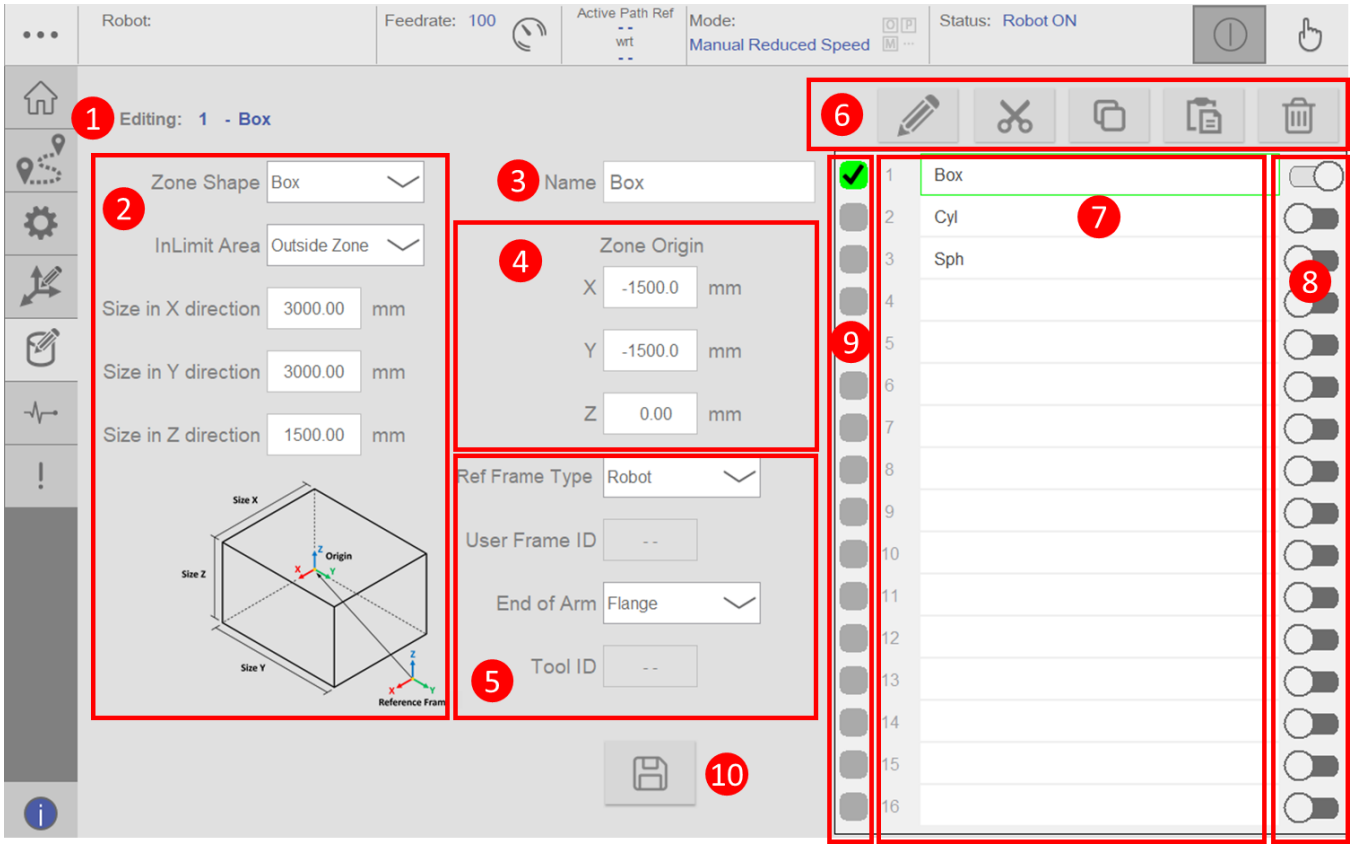


Figure 29- The Workzone page.

Item	Name	Description
1	Active Zone Display	Shows the zone currently selected.
2	Zone Definition Area	Allows you to define a zone for the robot to work within or outside of.
3	Zone Name Entry	Allows you to enter or change the name of the active zone in the zone list.

Item	Name	Description
4	Zone Origin Entry	Allows you to enter the origination point of the zone shape selected.
5	Frame Selection Group	Provides selection of the Frame being used to define the zone.
6	Zone Management Group	Provides functions to edit, cut, copy, paste, and delete defined zones.
7	Zone List	Shows all the defined Zones.
8	Enable/disable Zone Sliders	Enable/disable a zone or multiple zones.
9	Inside/Outside zone Display	Indicates whether the end of arm reference is currently inside or outside of the enabled zones.
10	Save button	When editing a zone, saves the current definition to the zone list

IMPORTANT

The end of arm tool referenced is in the defined InLimit Zone.

The end of arm tool referenced is NOT in the defined InLimit zone.

The zone is disabled.

The zone is improperly configured.

13.2 Defining a Zone

Three distinct zone geometries are currently supported: Box, Cylinder, and Sphere. Each type allows you to define an "InLimit" condition, which establishes whether the tool tip or flange is considered to be within or beyond the configured zone's boundaries. It is also possible to create complex zone areas by enabling multiple zones.

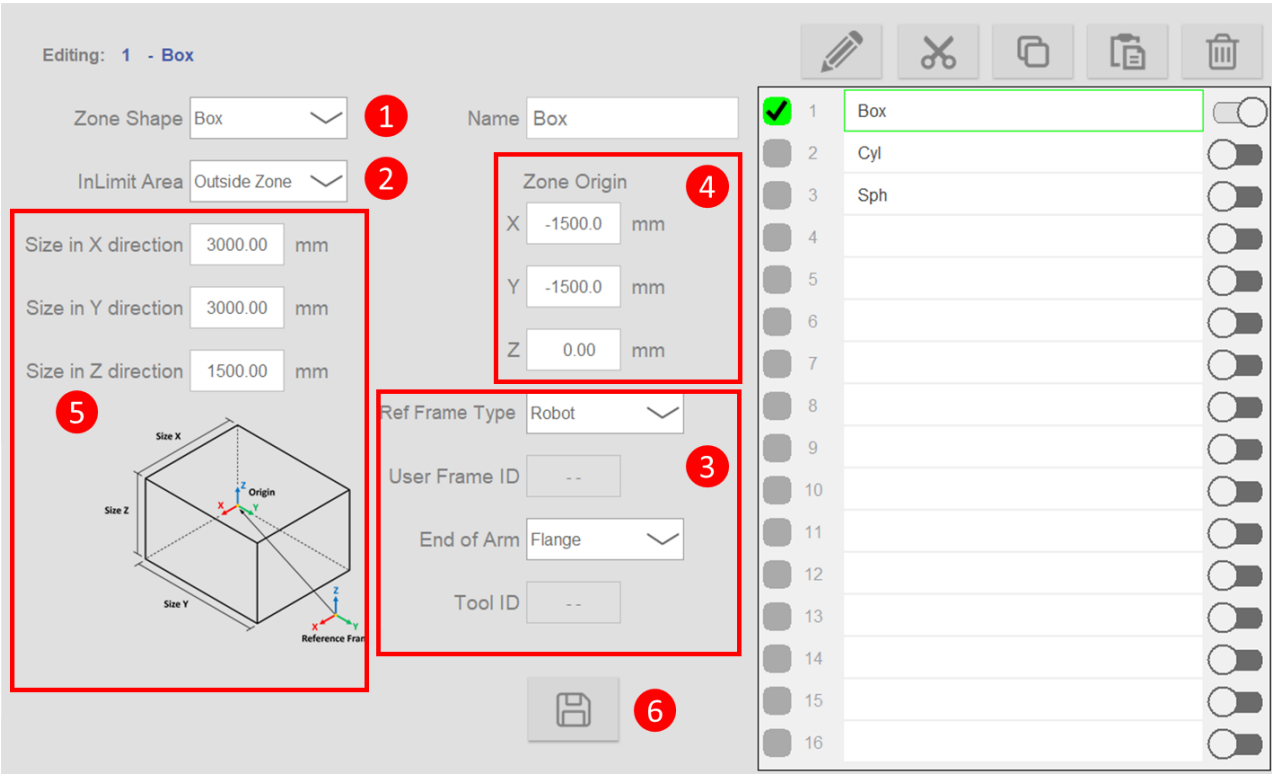


Figure 30- Work zone Definition

Item	Name	Description
1	Zone Shape Dropdown	Selects the shape of the zone; Box, Cylinder, or Sphere.
2	InLimit Area Dropdown	Select whether to indicate the robot is operating within or outside of the zone.
3	Reference Frame Area	Allows you to select the Frame and Tool to be used to define the zone. This selection will define where the zone will be referenced.
4	Zone Origin Area	Allows you to enter the origination point of the zone.
5	Zone Size Area	Allows you to enter the size of the zone using the zone Origin as the starting point.
6	Save Icon	When editing a zone, saves the current definition to the zone.

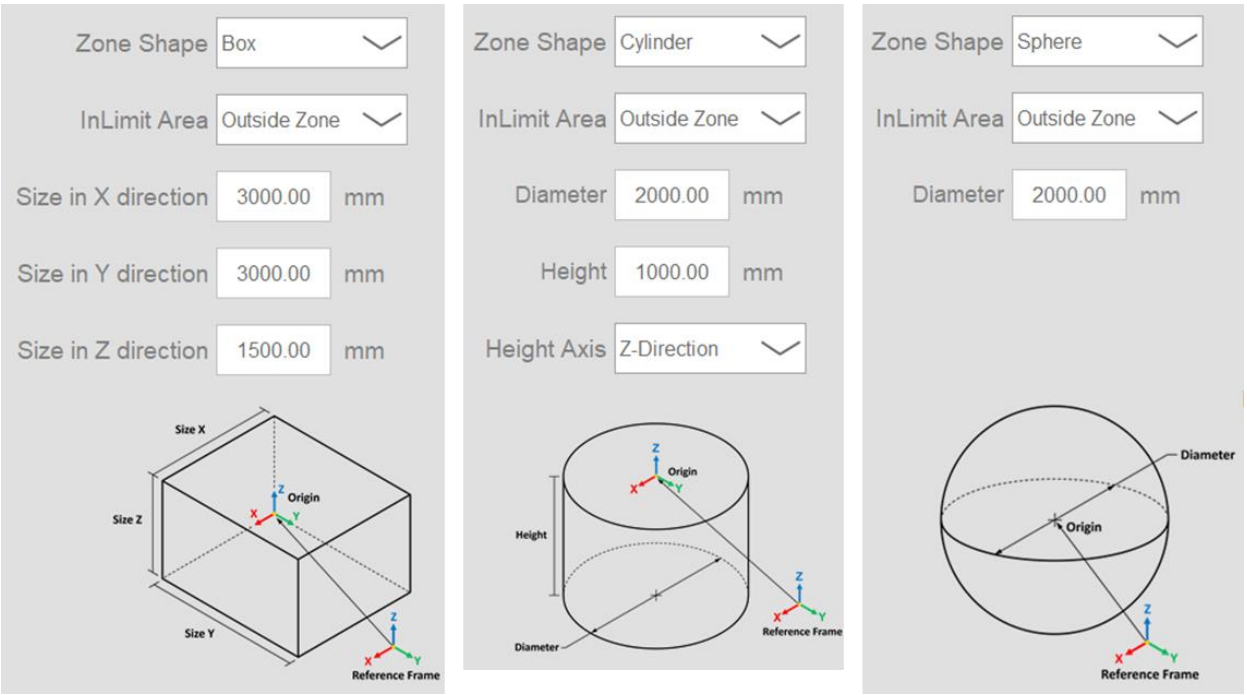


Figure 31- Work zone Size Areas for Box, Cylinder, and Sphere

13.2.1 Procedure

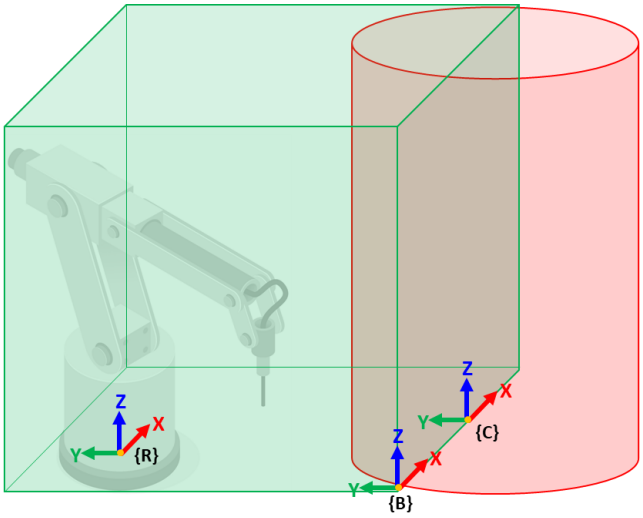
Step	Description
1	Select an available zone in the Zone Group.
2	Name the zone by typing a name in the zone Name entry box.
3	Select the shape of the zone to be defined using the zone Shape dropdown
4	Select whether you want the robot to indicate that it is inside or outside of this zone using the zone InLimit Area dropdown.
5	Select the Frame and Tool being used to define the zone.

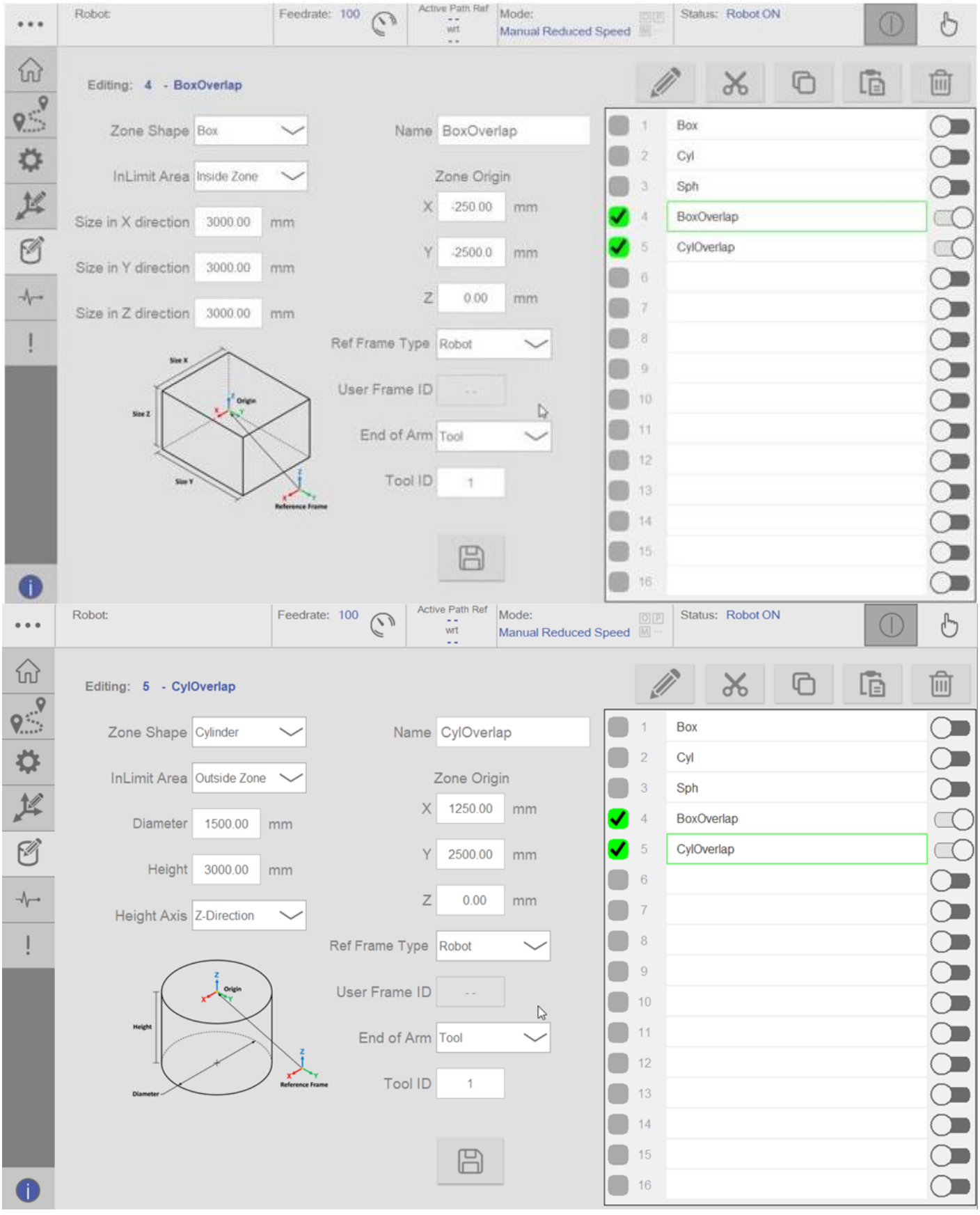
Step	Description
6	Define the origin of the zone as it relates to the origin of the Frame selected in step #5. In the example shown, the origin of the zone is 1.5m in the negative X direction and 1.5m in the negative Y direction from the Robot Frame selected in step #5.
7	Define the size of the zone. The entries here will be measured from the zone Origin point defined in step #6.
6	Press the Save icon

13.3 Creating Complex Work Envelopes Using Multiple Workzones

You can create complex work envelopes by defining and enabling multiple zones.
In this example, the robot tool needs to stay within the green square but must stay out of the red cylinder. So, we would need to define two overlapping zones:

1) A Box for the robot to stay within
2) A Cylinder for the robot to stay outside of.




Robot:
Feedrate: 100
Active Path Ref
wrt
Mode:
Manual Reduced Speed
Status: Robot ON
Editing: 4 - BoxOverlap
Zone Shape Box
InLimit Area Inside Zone
Size in X direction 3000.00 mm
Size in Y direction 3000.00 mm
Size in Z direction 3000.00 mm
Name BoxOverlap
Zone Origin
X -250.00 mm
Y -2500.0 mm
Z 0.00 mm
Ref Frame Type Robot
User Frame ID
End of Arm Tool
Tool ID 1
Size X
Size Z
Size Y
Origin
Reference Frame
1 Box
2 Cyl
3 Sph
4 BoxOverlap
5 CylOverlap
6
7
8
9
10
11
12
13
14
15
16
Robot:
Feedrate: 100
Active Path Ref
wrt
Mode:
Manual Reduced Speed
Status: Robot ON
Editing: 5 - CylOverlap
Zone Shape Cylinder
InLimit Area Outside Zone
Diameter 1500.00 mm
Height 3000.00 mm
Height Axis Z-Direction
Name CylOverlap
Zone Origin
X 1250.00 mm
Y 2500.00 mm
Z 0.00 mm
Ref Frame Type Robot
User Frame ID
End of Arm Tool
Tool ID 1
Origin
Height
Diameter
Reference Frame
1 Box
2 Cyl
3 Sph
4 BoxOverlap
5 CylOverlap
6
7
8
9
10
11
12
13
14
15
16

Note the Inside/Outside zone Display. The  icon indicates that the robotic tool is both inside the box zone and outside of the cylinder zone. If a zone is violated by the robot tool, the Inside/Outside zone display will display the  icon.

13.4 Using HMI Created Zones

Zones defined through the HMI generate status information that users can easily integrate into their code. This allows for custom annunciations or even robot motion interruption when movements enter or exit these zones. A public tag named **HMI_WorkZoneSts** is provided for this purpose. Each zone is assigned to a member of this DINT array. For example, consider a work zone configuration in position 1 of the zone table. The status information in **HMI_WorkZoneSts[1]** indicates the following:

HMI_WorkZoneSts[1].0 – HMI Zone 1 enable status
HMI_WorkZoneSts[1].1 – HMI Zone 1 within defined area
HMI_WorkZoneSts[1].2 – HMI Zone 1 outside defined area

14 Diagnostics

This Chapter briefly describes the contents of the **Diagnostics** and **Faults** pages.

14.1 Diagnostics Page

The **Diagnostics** page (Figure 24) has tables listing timestamped system events and registered methods for the selected robot.

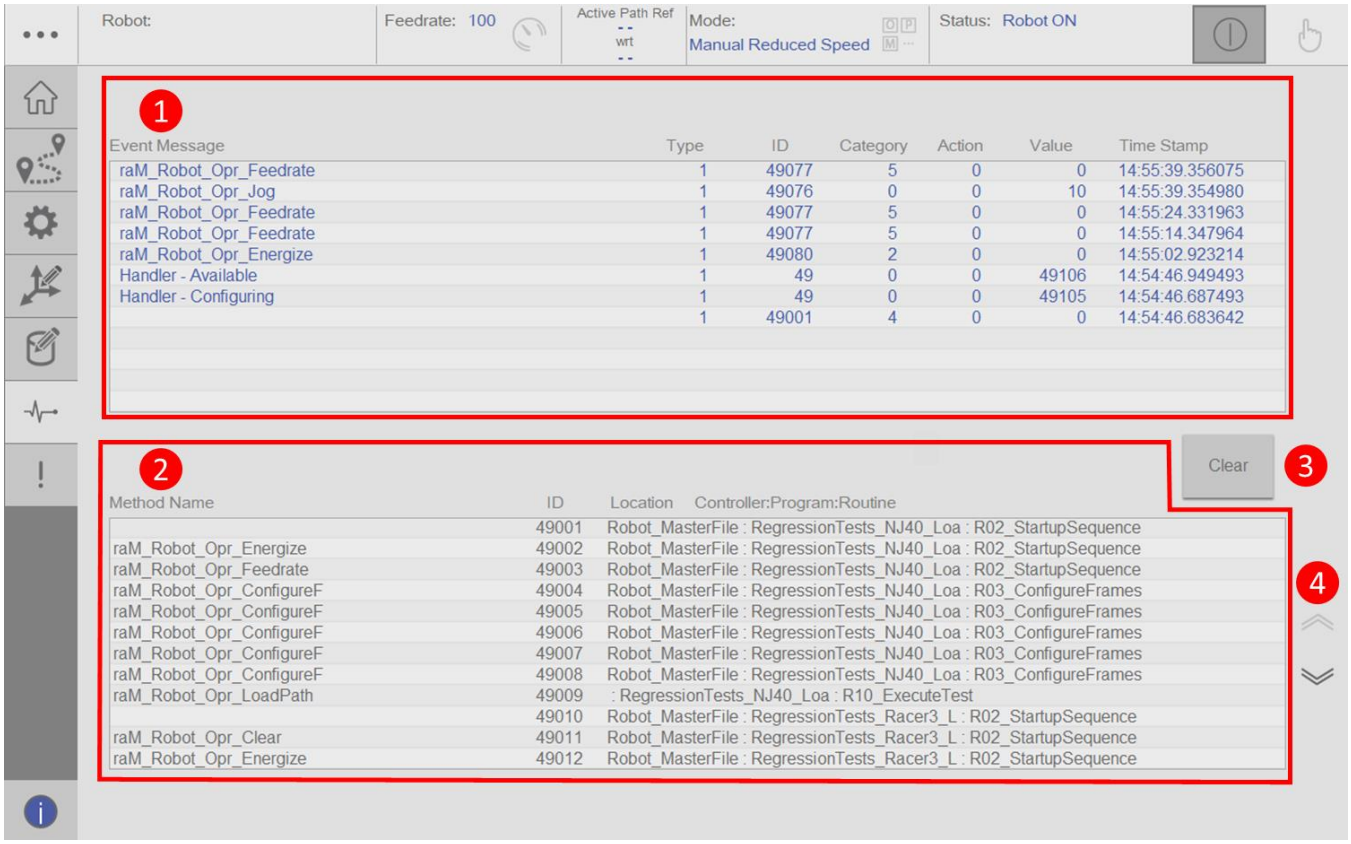


Figure 24- The Diagnostics page.

Item	Name	Description
1	Event list	Lists the system events in chronological order.
2	Method Registry list	Lists the registered methods in the robot controller and their locations.
3	Clear Events button	Clear all the events in the **Event** list.
4	Method Registry scrollbar	Scrolls up and down the **Method Registry** list.

14.1.1 Event List

The **Event** list provides basic timestamped information related to events generated by the system. The following event data is displayed in chronological order (latest first):

Column	Description
Event Message	Textual description of the event. It might contain just a message, or more detailed data such as the method that fired it.
Type	Event type identifier. 1: Status 2: Warning 3: Fault
ID	Either the method ID that caused the event, or event ID.
Action	If event is a fault, then method error code.
Time Stamp	Time when the event occurred, following the “hour:minute:second:millisecond” format.

14.1.2 Method Registry List

The **Method Registry** list shows all the methods belonging to the selected robot. The following method data is displayed:

Column	Description
Method Name	AOI name.
ID	Unique method ID assigned upon registration.
Location	Full path to where the method is instantiated, following the “controller:program:routine” format.

14.2 Faults Page

The **Faults** page (Figure 25) is a specialized version of the Event list in the **Diagnostics** page, showing only events of type “Fault”.

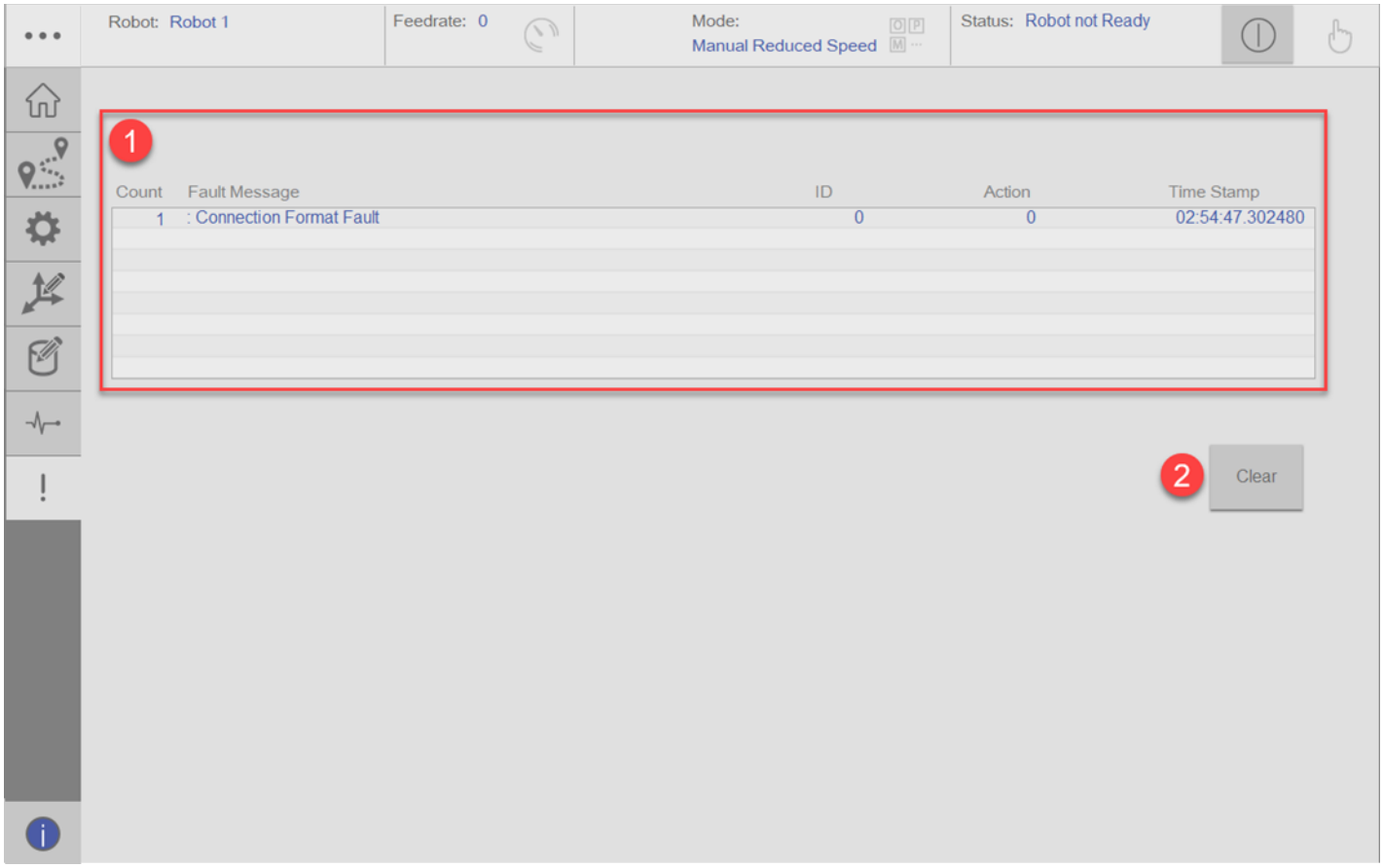


Figure 25- The Faults page.

Item	Name	Description
1	Fault list	Lists the system faults in chronological order.
2	Clear Faults button	Clear all the faults in the **Fault** list.

14.2.1 Fault List

The **Fault** list provides basic timestamped information related to faults generated by the system. The following fault data is displayed in chronological order (latest first):

Column	Description
Count	Number of occurrences of the fault since last reset.
Fault Message	Textual description of the fault.
ID	Either the method ID that caused the fault, or fault ID.
Action	Method error code.
Time Stamp	Time when the fault occurred, following the “hour:minute:second:millisecond” format.

15 Shut Down

This Chapter presents the final steps for softly shutting down the pendant whenever needed.

15.1 Shutting Down the Pendant

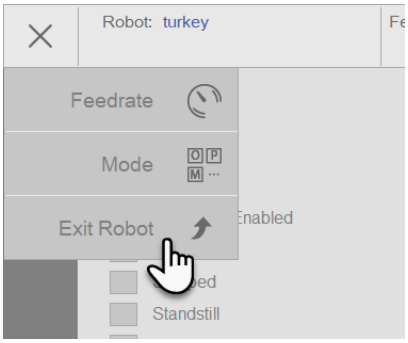
15.1.1 Purpose

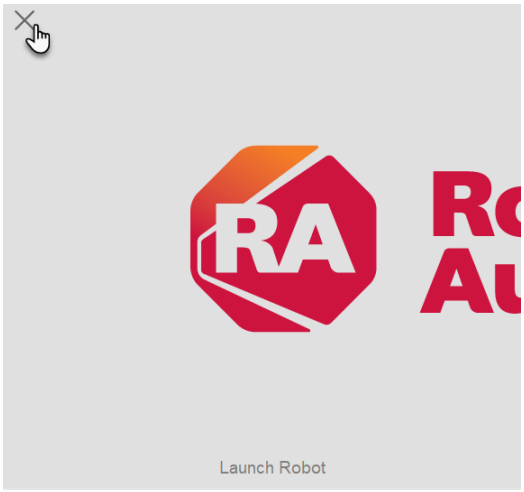
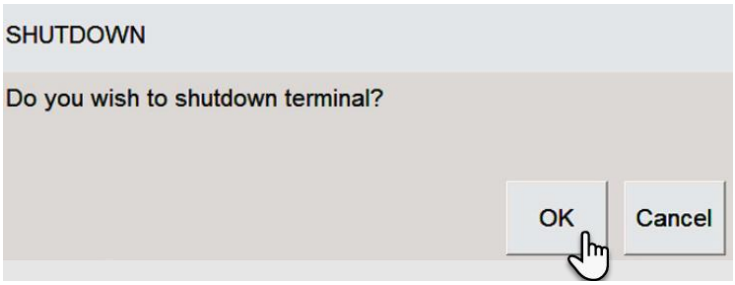
The pendant application and hardware can be shut down after all tasks have been completed and/or when maintenance work needs to be conducted.

15.1.2 Preconditions

- Pendant is powered up.
- A robot is selected.
- Robot is not energized.

15.1.3 Procedure

Step	Description
1	From any screen, click on the ••• button on the top left corner to open the **Context** menu.
2	Select the **Exit Robot** option to go back to the **Start** screen. 

Step	Description
3	On the **Start** screen, click on the **Close Application** button to close the FactoryTalk View ME application. 
4	Press  on the desktop taskbar to shut down the MobileView 2711T terminal.
5	Press the **OK** button on the confirmation dialog box to shut down the terminal. 

16 Appendix

General

This document provides a programmer with details on this OEM Building Block instruction for a Logix-based controller. You should already be familiar with how the Logix-based controller stores and processes data.

Novice programmers should read all the details about an instruction before using the instruction. Experienced programmers can refer to the instruction information to verify details.

IMPORTANT This OEM Building Block Instruction includes an Add-On Instruction for use with Version 24 or later of Studio 5000 Logix Designer.

Common Information for All Instructions

Rockwell Automation Building Blocks contain many common attributes or objects. Refer to the following reference materials for more information:

- Foundations of Modular Programming, **IA-RM001C-EN-P**

Conventions and Related Terms

Data - Set and Clear

This manual uses set and clear to define the status of bits (Booleans) and values (non-Booleans):

This Term:	Means:
Set	The bit is set to 1 (ON) A value is set to any non-zero number
Clear	The bit is cleared to 0 (OFF) All the bits in a value are cleared to 0

Signal Processing - Edge and Level

This manual uses Edge and Level to describe how bit (BOOL) Commands, Settings, Configurations and Inputs to this instruction are sent by other logic and processed by this instruction.

Send/Receive Method:	Description:
Edge	• Action is triggered by "rising edge" transition of input (0-1) • Separate inputs are provided for complementary functions (such as "enable" and "disable") • Sending logic SETS the bit (writes a 1) to initiate the action; this instruction CLEARS the bit (to 0) immediately, then acts on the request if possible • LD: use conditioned OTL (Latch) to send • ST: use conditional assignment [if (condition) then bit:=1;] to send • FBD: OREF writes a 1 or 0 every scan, should use Level, not Edge Edge triggering allows multiple senders per Command, Setting, Configuration or Input (many-to-one relationship)
Level	• Action ("enable") is triggered by input being at a level (in a state, usually 1) • Opposite action ("disable") is triggered by input being in opposite state (0) • Sending logic SETS the bit (writes a 1) or CLEARS the bit (writes a 0); this instruction does not change the bit • LD: use OTE (Energize) to send • ST: use unconditional assignment [bit:= expression_resulting_in_1_or_0;] or "if-then-else" logic [if (condition) then bit:= 1; else bit:= 0;] • FBD: use OREF to the input bit Level triggering allows only one sender can drive each Level

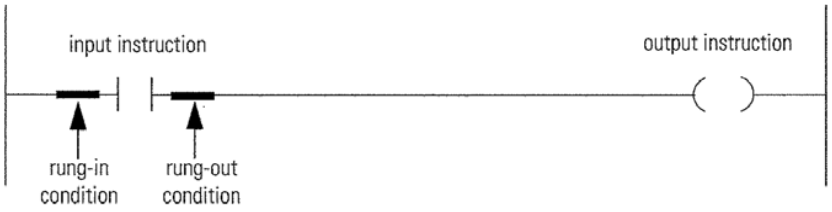
Instruction Execution - Edge and Continuous

This manual uses Edge and Continuous to describe how an instruction is designed to be executed.

Method:	Description:
Edge	• Instruction Action is triggered by "rising edge" transition of the rung-in-condition □
Continuous	• Instruction Action is triggered by input being at a level (in a state, usually 1) • Opposite action is triggered by input being in opposite state (0) • Instructions designed for continuous execution should typically be used on rungs without input conditions present allowing the instruction to be continuously scanned

Relay Ladder Rung Condition

The controller evaluates ladder instructions based on the rung condition preceding the instruction (rung-in condition). Based on the rung-in condition and the instruction, the controller sets the rung condition following the instruction (rung-out condition), which in turn, affects any subsequent instruction.



If the rung-in condition to an input instruction is true, the controller evaluates the instruction and sets the rung-out condition based on the results of the instruction. If the instruction evaluates to true, the rung-out condition is true; if the instruction evaluates to false, the rung-out condition is false.

IMPORTANT

The rung-in condition is reflected in the EnableIn parameter and determines how the system performs each Add-On Instruction. If the EnableIn signal is TRUE, the system performs the instruction's main logic routine. Conversely, if the EnableIn signal is FALSE, the system performs the instruction's EnableInFalse routine.

The instruction's main logic routine sets/clears the EnableOut parameter, which then determines the rung-out condition. The EnableInFalse routine cannot set the EnableOut parameter. If the rung-in condition is FALSE, then the EnableOut parameter and the rung-out condition will also be FALSE.

Pre-scan

On transition into RUN, the controller performs a pre-scan before the first scan. Pre-scan is a special scan of all routines in the controller. The controller scans all main routines and subroutines during pre-scan, but ignores jumps that could skip the execution of instructions. The controller performs all FOR loops and subroutine calls. If a subroutine is called more than once, it is performed each time it is called. The controller uses pre-scan of relay ladder instructions to reset non-retentive I/O and internal values.

During pre-scan, input values are not current and outputs are not written. The following conditions generate pre-scan:

- Transition from Program to Run mode.
- Automatically enter Run mode from a power-up condition.

Pre-scan does not occur for a program when:

- Program becomes scheduled while the controller is running.
- Program is unscheduled when the controller enters Run mode.

IMPORTANT The Pre-scan process performs the Process Add-On Instruction's logic routine as FALSE and then performs its Pre-scan routine as TRUE.