Rockwell Automation Application Content

Rockwell Automation Robotics Libraries

Reference Manual

Teach Tool Frame - Robot

raM_Robot_Opr_TeachToolFrame V2.0

January, 2024

Rockwell Automation

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

raM_Robot_Opr_TeachToolFrame:

Instruction calculates and stores a tool frame by touching a point in space at four distinct orientations.

Use when:

- Using a Device Handler for Robot Management.
- Manually jogging the robot to teach tool frames.
- The tool frame to be taught has the same orientation as the flange frame.
- The robot being used has at least five degrees of freedom.

Do NOT use when:

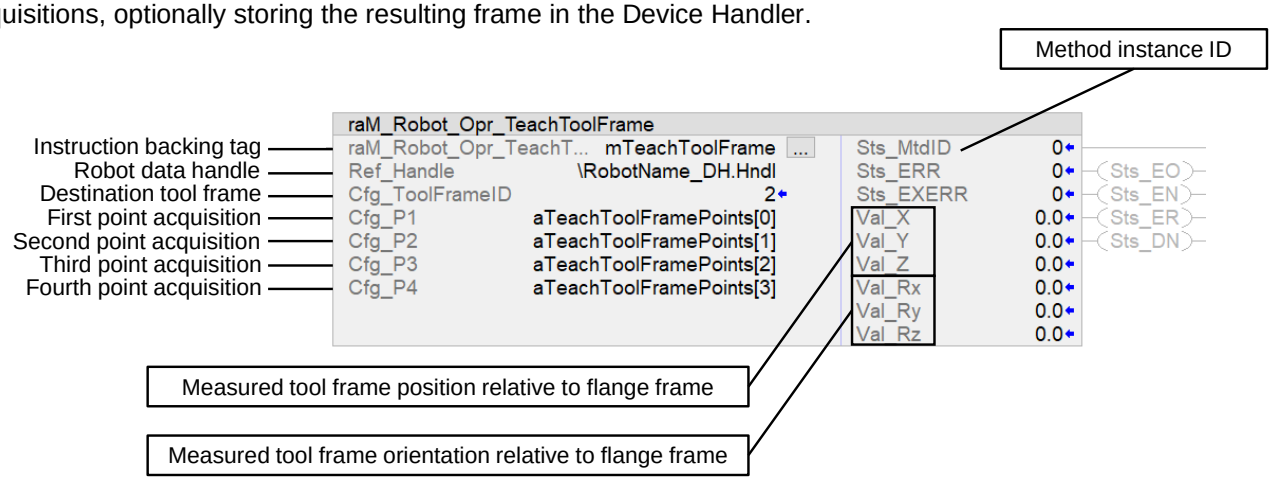
- Not using a Device Handler for Robot Management.
- The tool frame coordinates are available for manual configuration.
- Teaching a robot frame or a user frame.
- The tool frame to be taught has a different orientation than the flange frame.
- The robot being used has less than five degrees of freedom.

1.1 Prerequisites

- Device Handler for Robot
  - Rockwell Automation Robotics Libraries v2.0 →
- Studio 5000 – Logix Designer
  - v35.0 →
- Studio 5000 – Application Code Manager
  - v4.03.00 →

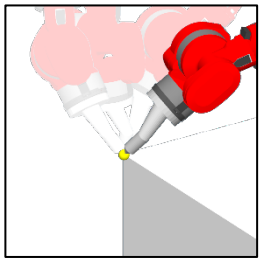
1.2 Functional Description

The instruction measures the coordinates of a tool frame relative to the flange frame from four point acquisitions, optionally storing the resulting frame in the Device Handler.

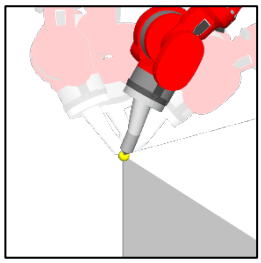


Each point acquisition results from touching a single point in space at for different orientations:

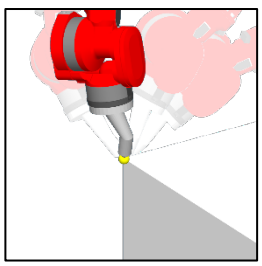
- P1 is obtained by touching the point with the tool tip (to be taught) at an arbitrary orientation.

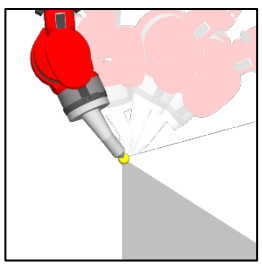


- P2 is obtained by touching the same point with the tool tip, but at a different orientation.



- Similarly, P3 and P4 are obtained by touching the same point with the tool tip at two new distinct orientations.





The four acquisitions must be supplied considering that the robot tool was manually jogged to each orientation in space. The inputs Cfg_P1, Cfg_P2, and Cfg_P3 define flange poses relative to the robot frame {R} when performing the acquisitions. This data can be readily retrieved from the Device Handler by copying the current value of the program public tag “Sts.Pose” when the robot is manually jogged to the point and kept standstill. The public tag “Sts.Pose” contains the current flange pose relative to the robot frame {R}.

The relevant frames, poses, and points of a complete three-point acquisition are depicted in Figure 1.

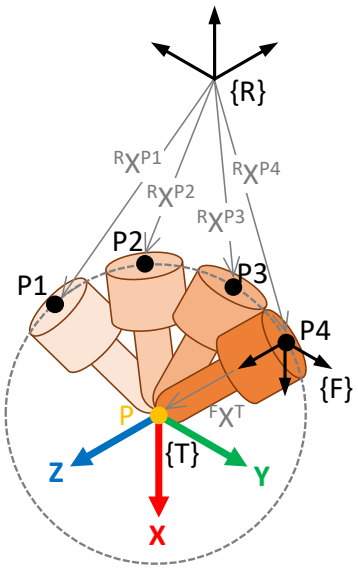


Figure 1 – Relationship between the four measurements of a fixed point P and the resulting tool frame {T}. Acquisitions P1, P2, P3, and P4 must be expressed as flange poses relative to robot frame {R} (${}^{R}X^{Pn}$). The calculated tool frame {T} is defined relative to the flange frame {F} with the transformation ${}^{F}X^{T}$. It is assumed that the flange and tool orientations are equal.

When the same point in space is touched with the tool tip at four different poses, the flange origin visits four points lying on a sphere. The measurement algorithm calculates the tool frame origin as the center of the sphere derived from P1, P2, P3, and P4. Acquisitions must be sufficiently apart from each other for the algorithm to work properly. The tool frame is assumed to have the same orientation as the flange frame.

The input Cfg_ToolFrameID enables the possibility of storing the calculated tool frame in the Device Handler with the desired ID.

The output user frame is formatted and provided as a tuple of six values (Val_X, Val_Y, Val_Z, Val_Rx, Val_Ry, and Val_Rz) that describe the position and orientation of the calculated tool frame relative to the flange frame {F}.

General Status Bit Behavior:

Note: Status bit not shown on the output side of the instruction are not used and will not exist in the instruction backing tag.

Status Bit	Description / Behavior
*.Sts_EO	• Enable Out indicated the status of the output line of the instruction. • If false (logically LO) any instruction on the ladder rung between the instruction and the neutral rail will not be energized. • If the instruction is removed from ladder scan either in a conditional subroutine, MCR zone, JMP/LBL etc., the bit will remain in its last evaluated state.
*.Sts_EN	• The rung-in condition of the ladder rung is true and the instruction is being evaluated. • If the instruction is removed from ladder scan either in a conditional subroutine, MCR zone, JMP/LBL etc., the bit will remain in its last evaluated state.
*.Sts_ER	• If the instruction experiences an internal error, the *. Sts_ER bit will be set. Error codes / Extended codes can be found by monitoring the backing tag *.Sts_ERR / *.Sts_EXERR members respectively. • If the instruction is removed from ladder scan either in a conditional subroutine, MCR zone, JMP/LBL etc., the bit will remain in its last evaluated state.
*.Sts_DN	• Used when the execution of the instruction completes within a single scan. • If the instruction is removed from ladder scan either in a conditional subroutine, MCR zone, JMP/LBL etc., the bit will remain in its last evaluated state.
*.Sts_IP	• Used to identify the instruction is in the process • If the instruction is removed from ladder scan either in a conditional subroutine, MCR zone, JMP/LBL etc., the bit will remain in its last evaluated state.
*.Sts_PC	• Used when the execution of the instruction requires more than a single scan to complete, and indicates the ‘process’ carried out by the instruction has successfully completed. • If the instruction is removed from ladder scan either in a conditional subroutine, MCR zone, JMP/LBL etc., the bit will remain in its last evaluated state.

1.3 Execution

- Level

1.3.1 Overview

Rung in condition transition response:

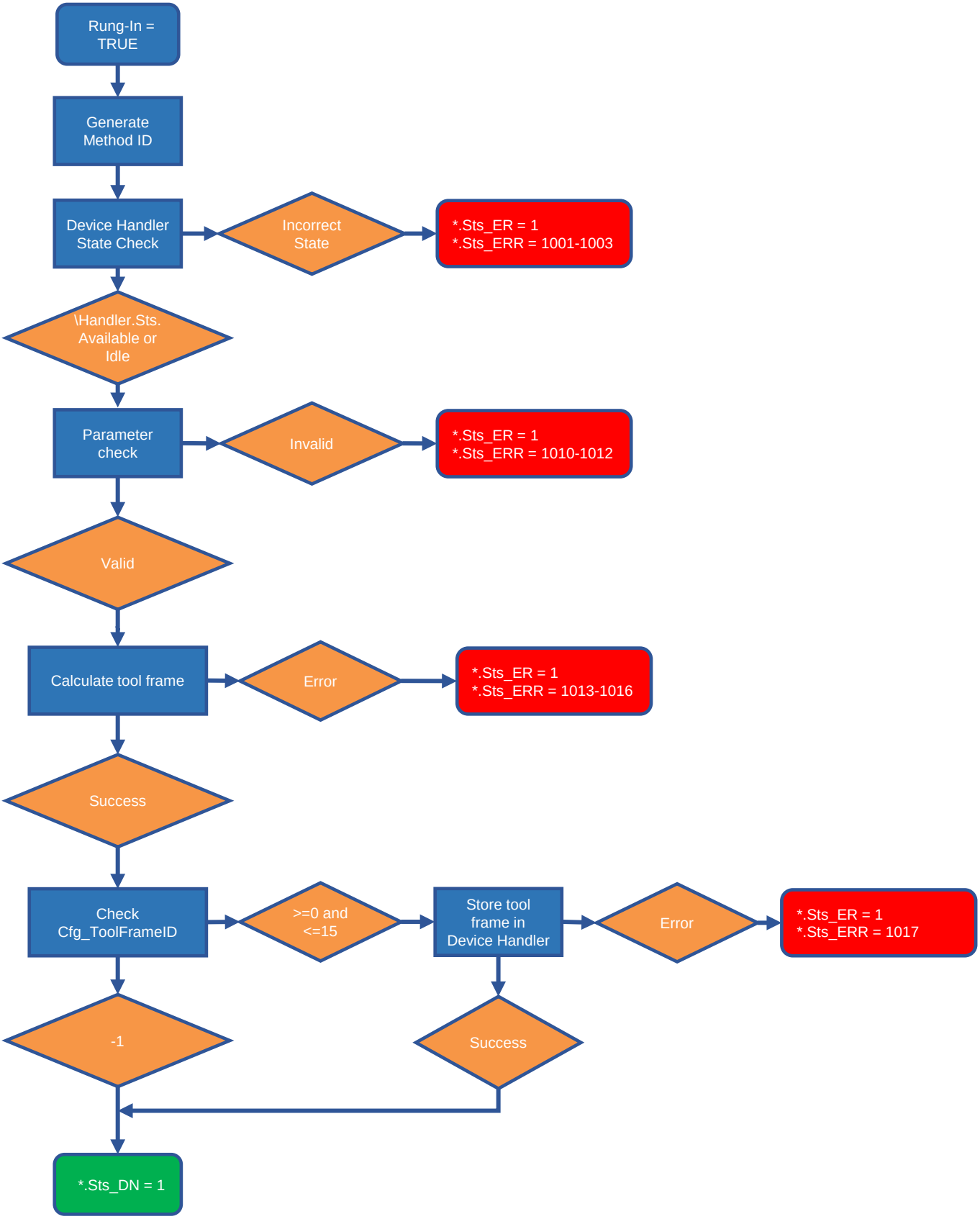
- False → True
  - Initialization
    - *.Sts_EO = 0
    - *.Sts_ER = 0
    - *.Sts_DN = 0
  - Running
    - *.Sts_EO = 1
    - *.Sts_EN = 1
      - Check parameters
      - Calculate tool frame origin relative to robot frame {R}
      - Change tool frame reference to flange frame {F}
      - Check calculated frame consistency
      - IF: No errors
        - THEN:
          - IF: Cfg_ToolFrameID <> -1
            - THEN: Store frame in Device Handler
          - *.Sts_DN = 1
      - IF: Error
        - THEN: *.Sts_DN = 0 and *.Sts_ER = 1

- True → False
  - *.Sts_EO = 0
  - *.Sts_EN = 0
    - IF: Error
      - THEN: *.Sts_ER = 1

1.3.2 Affected Device Handler Status

Status	Value

1.3.3 Execution Table



2 Instruction

2.1 Input Data

Input	Function / Description	DataType
Ref_Handle	Device Handler Data Structure	raM_UDT_Dvc_Robot_DataHndl
Cfg_ToolFrameID	Destination tool frame (-1: do not store; 0..15: tool frame id)	DINT
Cfg_P1	First flange pose relative to robot frame when touching reference point	raM_UDT_Robot_Opr_Frame
Cfg_P2	Second flange pose relative to robot frame when touching reference point	raM_UDT_Robot_Opr_Frame
Cfg_P3	Third flange pose relative to robot frame when touching reference point	raM_UDT_Robot_Opr_Frame
Cfg_P4	Fourth flange pose relative to robot frame when touching reference point	raM_UDT_Robot_Opr_Frame

2.2 Output Data

Output	Function / Description	DataType
Sts_EO	Instruction has enabled the rung output. Provides a visible indicator of the EnableOut system parameter for use during ladder instantiation	BOOL
Sts_EN	Instruction is Being Scanned - Rung In Condition = TRUE	BOOL
Sts_ER	Instruction is in Error - See Sts_ERR / Sts_EXERR for Additional Error Information	BOOL
Sts_ERR	Instruction Error Code - See Instruction Help for Code Definition	DINT
Sts_EXERR	Instruction Extended Error Code - See Instruction Help for Code Definition	DINT
Sts_MtdID	Method ID	DINT
Sts_DN	Instruction Execution has Completed	BOOL
Val_X	Tool frame X position relative to flange frame [mm]	REAL
Val_Y	Tool frame Y position relative to flange frame [mm]	REAL
Val_Z	Tool frame Z position relative to flange frame [mm]	REAL
Val_Rx	Tool frame Rx orientation relative to flange frame [deg]	REAL
Val_Ry	Tool frame Ry orientation relative to flange frame [deg]	REAL
Val_Rz	Tool frame Rz orientation relative to flange frame [deg]	REAL

2.3 Error Codes

Sts_ERR	Description
0	No errors present
1001	Device Handler is not running. Verify Ethernet modules and motion group state.
1002	Device Handler is faulted.
1003	Device Handler is not in a supported state. Device Handler must be in state Available. Verify Ethernet modules, motion group state and that the Device Handler has been configured.
1010	Cfg_ToolFrameID: Invalid tool id. Check if the value is between -1 and maximum frame number.
1011	Cfg_P1-P4: Invalid pose description. Acquisitions must be specified as flange poses relative to robot frame.
1012	Robot does not have the minimum degrees of freedom to teach a tool frame. Check if the robot has at least 5 DOF.
1013	Cfg_P1-P4: Acquisitions are not independent. Check if the four acquisitions are sufficiently apart from each other.
1014	Cfg_P1-P4: Invalid acquisitions. Make sure that the TCP touches the same point in every acquisition.
1015	Calculated frame is not right orthonormal.
1016	Error converting Euler angles.
1017	Error saving frame. See EXERR for Configure Frame error code.

Sts_EXERR	Description
< Number >	If an instruction error occurs internally, the value of the instruction*.ERR DINT will be placed in Sts_EXERR.

3 Application Code Manager

3.1 Definition Object: raM_Robot_Opr_TeachToolFrame

This object contains the AOI definition and used as linked library to implement object. This gives flexibility to choose to instantiate only definition and create custom implement code. User may also create their own implement library and link with this definition library object.

3.2 Implementation Object: raM_LD_Robot_TeachToolFrame

Implementation Language: Ladder
Content Type: Routine

This implement contains only a rung with an instance of the raM_Robot_Opr_TeachToolFrame object.

Parameter Name	Default Value	Instance Name	Definition	Description
RoutineName	_{ObjectName}	{RoutineName}	Routine	Name of the routine where the object will be placed
TagName	{ObjectName}	{TagName}	Tag	Instruction backing tag
StartBitTagName	Cmd_{ObjectName}	{StartBitTagName}	Local Tag	Tag name for start command enabling bit

Linked Library

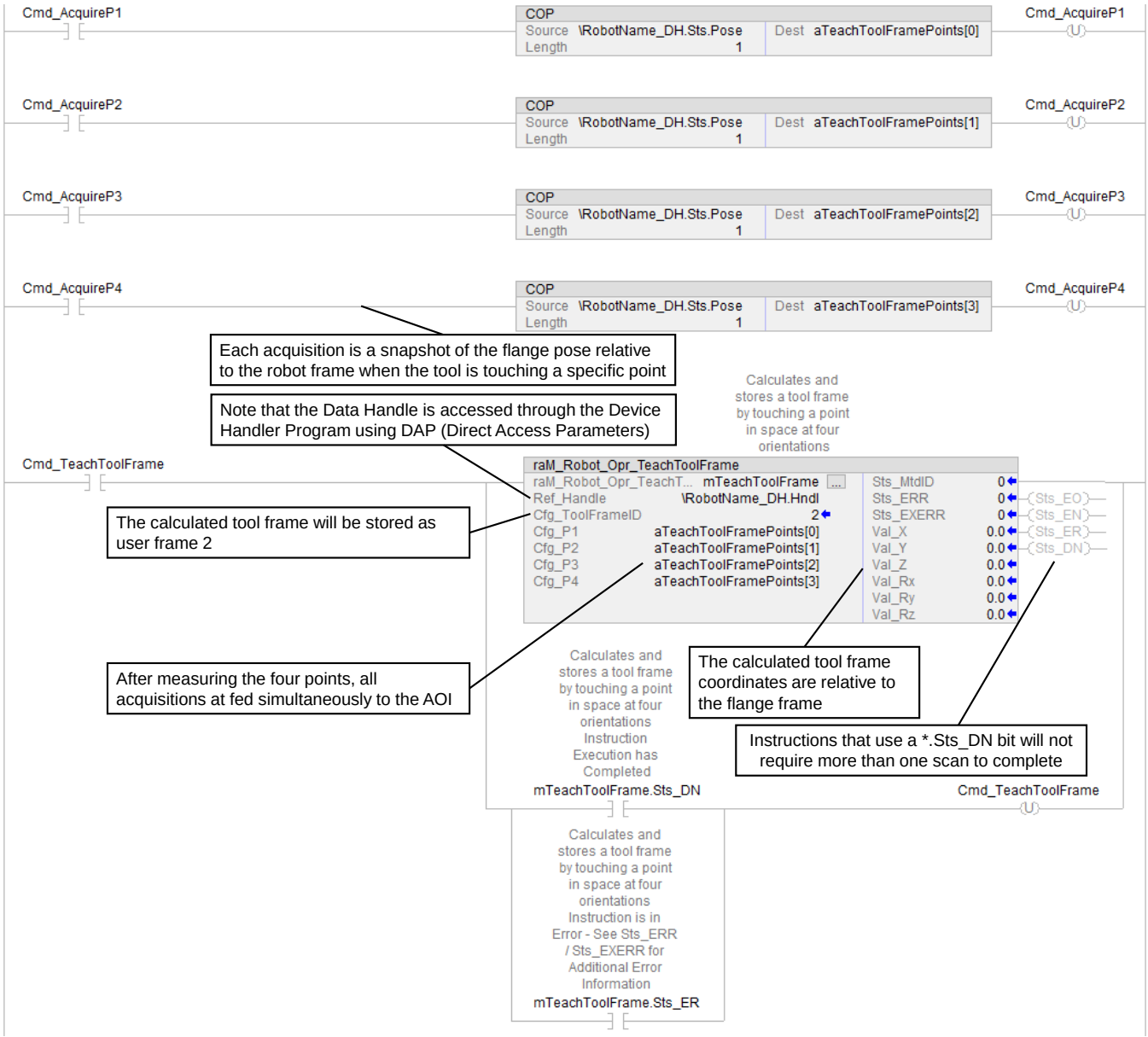
Link Name	Catalog Number	Revision	Solution	Category
RobotHandler	raM_Robot_Dvc_DeviceHandler	2	(RA-LIB) Robotics	Robot Handler
raM_Robot_Opr_TeachToolFrame	raM_Robot_Opr_TeachToolFrame	2	(RA-LIB) Robotics	Asset-Control

3.3 Attachments

Name	Description	File Name	Extraction path
V2_{LibraryName}	Reference Manual	RM-{LibraryName}.pdf	{ProjectName}\Documentation

4 Application

4.1 Using raM_Robot_Opr_TeachToolFrame



5 *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

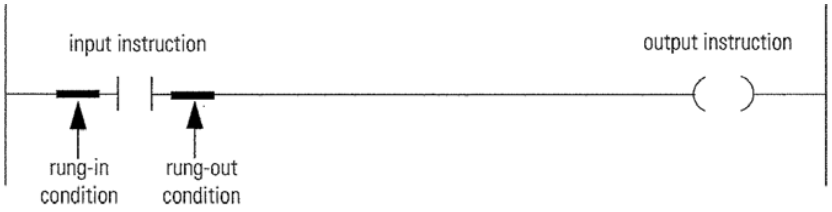
Instruction Execution - Edge and Continuous

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.