

# Rockwell Automation Application Content

## *Rockwell Automation Robotics Libraries*



## Reference Manual

### Teach User Frame - Robot

raM\_Robot\_Opr\_TeachUserFrame

V2.0

January, 2024

## 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

<b>Table of Contents .....</b>	<b>3</b>
<b>1 Overview.....</b>	<b>4</b>
1.1 Prerequisites .....	4
1.2 Functional Description.....	5
1.3 Execution .....	8
<b>2 Instruction.....</b>	<b>10</b>
2.1 Input Data.....	10
2.2 Output Data .....	10
2.3 Error Codes.....	11
<b>3 Application Code Manager.....</b>	<b>12</b>
3.1 Definition Object: raM_Robot_Opr_TeachUserFrame.....	12
3.2 Implementation Object: raM_LD_Robot_TeachUserFrame.....	12
3.3 Attachments.....	12
<b>4 Application.....</b>	<b>13</b>
4.1 Using raM_Robot_Opr_TeachUserFrame .....	13
<b>5 Appendix.....</b>	<b>14</b>
General .....	14
Common Information for All Instructions .....	14
Conventions and Related Terms .....	14

# 1 Overview

raM\_Robot\_Opr\_TeachUserFrame:

Instruction calculates and stores a user frame based on three points measured with the end-effector.

Use when:

- Using a Device Handler for Robot Management.
- Manually jogging the robot to teach user frames.

Do NOT use when:

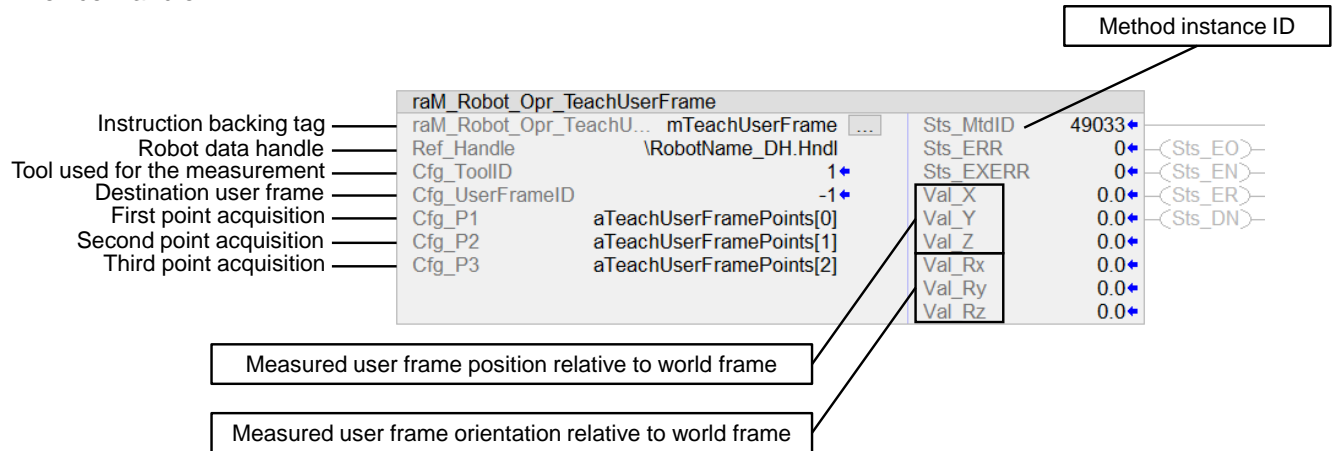
- Not using a Device Handler for Robot Management.
- The user frame coordinates are available for manual configuration.
- Teaching a robot frame or a tool frame.

## 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 →

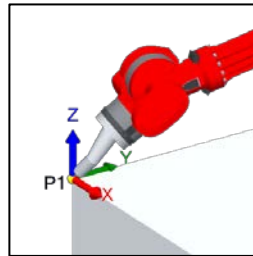
## 1.2 Functional Description

Instruction calculates a user frame from three point acquisitions, optionally storing the resulting frame in the Device Handler.

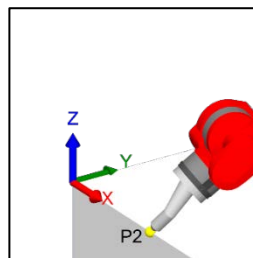


Each point acquisition has a specific purpose:

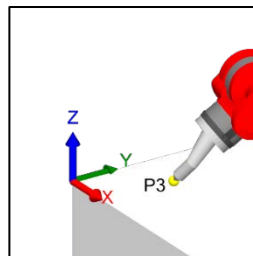
- P1 is obtained by touching the origin of the desired user frame.



- P2 is obtained by touching a point lying on the X axis of the desired user frame, with  $X > 0$ .



- P3 is obtained by touching a point lying on the XY plane of the desired user frame, with  $Y > 0$ .



The three acquisitions must be supplied considering that the robot tool was manually jogged to each point in space. The inputs Cfg\_ToolID, Cfg\_P1, Cfg\_P2, and Cfg\_P3 define the end-effector poses relative to the robot frame {R} when performing the acquisitions. Cfg\_P1-P3 can be interpreted in two different ways, depending on the supplied value for Cfg\_ToolID:

Cfg_ToolID	Cfg_P1-P3
-1	Tool poses relative to the robot frame {R} when measuring the three points. No offset will be applied. It is the user's responsibility to calculate/ensure that the acquisitions reflect the tool center point relative to the robot frame.
0..15	Flange poses relative to the robot frame {R} when measuring the three points. This arrangement is convenient because the 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 tool frame number Cfg_ToolID is added as an offset to each acquisition to account for the installed tool used for measuring the points.

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

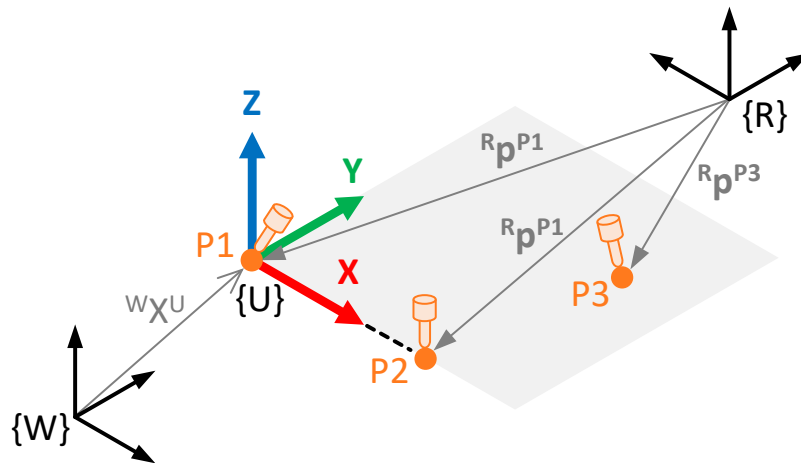


Figure 1 – Relationship between the three measurements and the resulting user frame {U}. Points P1, P2, and P3 must be expressed relative to robot frame {R} ( $R_{pPn}$ ). The calculated user frame is expressed relative to the world frame {W} ( $W_{Xu}$ ).

The following restrictions apply to the input measurements prior to evaluating the output user frame:

- Points P1 and P2 cannot be coincident.
- Points P1 and P3 cannot be coincident.
- Points P1, P2, and P3 cannot be collinear.

The input Cfg\_UserFrameID enables the possibility of storing the calculated user 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 user frame relative to the world frame {W}.

### 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	<ul style="list-style-type: none"><li>• Enable Out indicated the status of the output line of the instruction.</li><li>• If false (logically LO) any instruction on the ladder rung between the instruction and the neutral rail will not be energized.</li><li>• 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.</li></ul>
*.Sts_EN	<ul style="list-style-type: none"><li>• The rung-in condition of the ladder rung is true and the instruction is being evaluated.</li><li>• 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.</li></ul>
*.Sts_ER	<ul style="list-style-type: none"><li>• 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.</li><li>• 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.</li></ul>
*.Sts_DN	<ul style="list-style-type: none"><li>• Used when the execution of the instruction completes within a single scan.</li><li>• 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.</li></ul>
*.Sts_IP	<ul style="list-style-type: none"><li>• Used to identify the instruction is in the process</li><li>• 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.</li></ul>
*.Sts_PC	<ul style="list-style-type: none"><li>• 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.</li><li>• 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.</li></ul>

## 1.3 Execution

- Level

### 1.3.1 Overview

Rung in condition transition response:

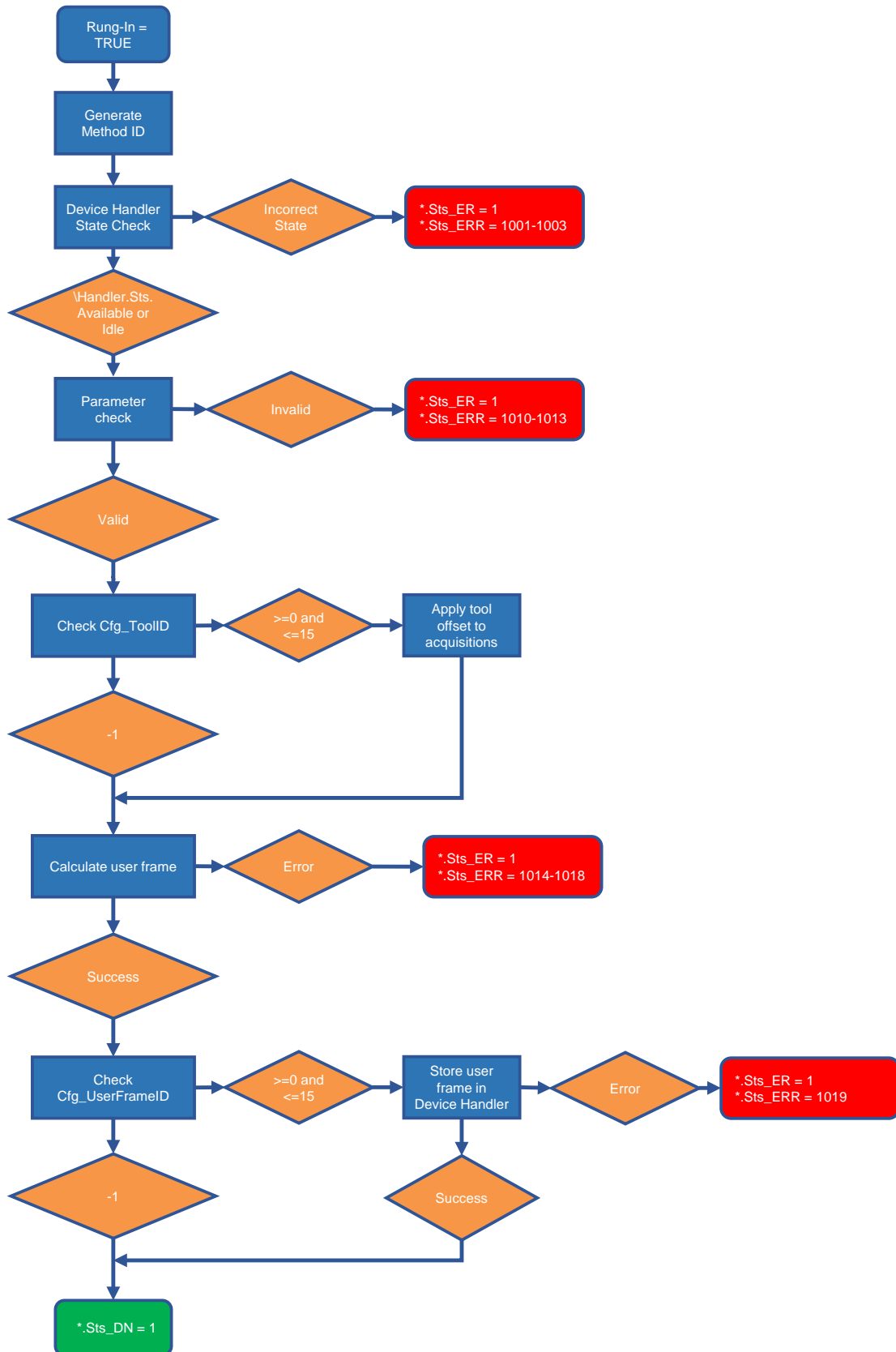
- False → True
  - Initialization
    - \*.Sts\_EO = 0
    - \*.Sts\_ER = 0
    - \*.Sts\_DN = 0
  - Running
    - \*.Sts\_EO = 1
    - \*.Sts\_EN = 1
      - Check parameters
      - IF: Cfg\_ToolID <> -1
        - THEN: Apply tool offset to acquisitions
      - Calculate user frame relative to robot frame {R}
      - Change user frame reference to world frame {W}
      - Check calculated frame consistency
      - IF: No errors
        - THEN:
          - IF: Cfg\_UserFrameID <> -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_ToolID	Tool offset to be added to the acquisitions if Cfg_P1-P3 are flange poses (-1: none; 0..15: tool id)	DINT
Cfg_UserFrameID	Destination user frame (-1: do not store; 0..15: user frame id)	DINT
Cfg_P1	Flange or tool pose relative to robot frame when touching point defining the frame's origin	raM_UDT_Robot_Opr_Frame
Cfg_P2	Flange or tool pose relative to robot frame when touching point on frame's X axis with X > 0	raM_UDT_Robot_Opr_Frame
Cfg_P3	Flange or tool pose relative to robot frame when touching point on XY plane with Y > 0	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	User frame X position relative to world frame [mm]	REAL
Val_Y	User frame Y position relative to world frame [mm]	REAL
Val_Z	User frame Z position relative to world frame [mm]	REAL
Val_Rx	User frame Rx orientation relative to world frame [deg]	REAL
Val_Ry	User frame Ry orientation relative to world frame [deg]	REAL
Val_Rz	User frame Rz orientation relative to world 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_ToolID: Invalid tool id. Check if the value is between -1 and maximum frame number.
1011	Cfg_ToolID: Tool frame not defined.
1012	Cfg_UserFrameID: Invalid user frame id. Check if the value is between -1 and maximum frame number.
1013	Points P1, P2, and P3 must be expressed relative to robot frame.
1014	Points P1 and P2 cannot be coincident.
1015	Points P1 and P3 cannot be coincident.
1016	Points P1, P2, and P3 cannot be collinear.
1017	Calculated frame is not right orthonormal.
1018	Error converting Euler angles.
1019	Error saving frame. See EXERR for raM_Robot_Opr_ConfigureFrame 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\_TeachUserFrame

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\_TeachUserFrame

Implementation Language: Ladder

Content Type: Routine

This implement contains only a rung with an instance of the raM\_Robot\_Opr\_TeachUserFrame 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}		Local Tag	Tag name for start command enabling bit

#### Linked Library

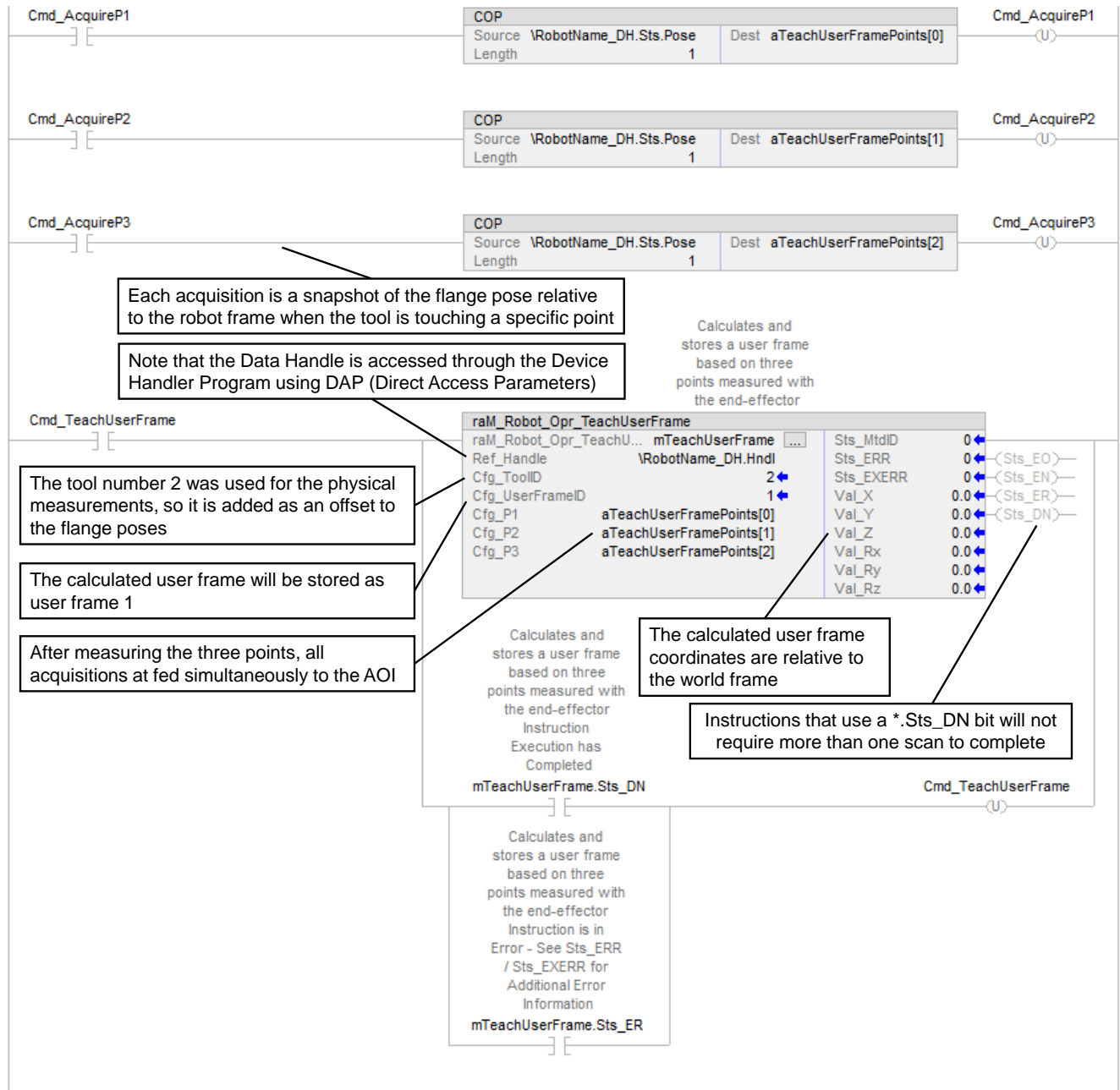
Link Name	Catalog Number	Revision	Solution	Category
RobotHandler	raM_Robot_Dvc_DeviceHandler	2	(RA-LIB) Robotics	Robot Handler
raM_Robot_Opr_TeachUserFrame	raM_Robot_Opr_TeachUserFrame	2	(RA-LIB) Robotics	Asset-Control

#### 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\_TeachUserFrame



## 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):

<b>This Term:</b>	<b>Means:</b>
<b>Set</b>	The bit is set to 1 (ON) A value is set to any non-zero number
<b>Clear</b>	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

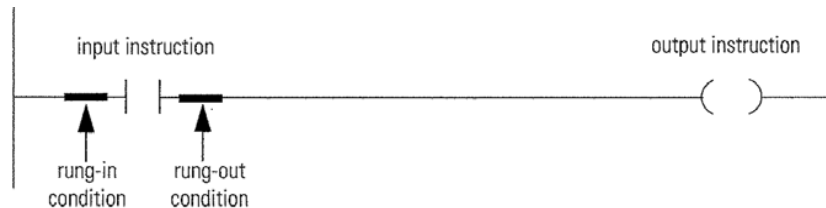
This manual uses Edge and Continuous to describe how an instruction is designed to be executed.

Method:	Description:
Edge	<ul style="list-style-type: none"><li>• Instruction Action is triggered by "rising edge" transition of the rung-in-condition</li></ul>
Continuous	<ul style="list-style-type: none"><li>• Instruction Action is triggered by input being at a level (in a state, usually 1)</li><li>• Opposite action is triggered by input being in opposite state (0)</li><li>• Instructions designed for continuous execution should typically be used on rungs without input conditions present allowing the instruction to be continuously scanned</li></ul>



## 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.

---