OMRON

Automation Software Sysmac Studio

Vision & Robot Integrated Simulation Calibration Parameter Technology Introduction Guide

SYSMAC-SE2□□□
SYSMAC-RA401L
NJ501-4□□□
R88D-KN□-ECT
FH-1□□□
FH-3□□□

Technology Introduction Guide

Y213-E1-01

SYSMAC always in control

Introduction

Thank you for purchasing the Automation Software Sysmac Studio.
The Automation Software Sysmac Studio Vision & Robot Integrated Simulation Calibration Parameter Technology Introduction Guide (hereinafter called this guide) describes the calibration parameters calculated by the Vision & Robot integrated simulation that is one of simulations of the Sysmac Studio.

This guide does not contain safety information and other details that are required for actual use. Thoroughly read and understand the manuals for all of the devices that are used in this guide to ensure that the system is used safely. Review the entire contents of these materials, including all safety precautions, precautions for safe use, and precautions for correct use.

Refer to the *Startup Guide for Vision & Robot Integrated Simulation* (Cat. No. Y128) for the operating procedures of the Vision & Robot integrated simulation.

Intended Audience

This guide is intended for the following personnel, who must also have knowledge of electrical systems (an electrical engineer or the equivalent).

- Personnel in charge of introducing FA systems.
- Personnel in charge of designing FA systems.
- Personnel in charge of managing FA systems and facilities.

Applicable Products

This guide covers the following product.

- Automation Software Sysmac Studio

Terms and Conditions Agreement

Automation Software Sysmac Studio

1. WARRANTY
   - The warranty period for the Software is one year from the date of purchase, unless otherwise specifically agreed.
   - If the User discovers defect of the Software (substantial non-conformity with the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504)), and return it to OMRON within the above warranty period, OMRON will replace the Software without charge by offering media or download from OMRON's website. And if the User discovers defect of media which is attributable to OMRON and return it to OMRON within the above warranty period, OMRON will replace defective media without charge. If OMRON is unable to replace defective media or correct the Software, the liability of OMRON and the User's remedy shall be limited to the refund of the license fee paid to OMRON for the Software.

2. LIMITATION OF LIABILITY
   - THE ABOVE WARRANTY SHALL CONSTITUTE THE USER'S SOLE AND EXCLUSIVE REMEDIES AGAINST OMRON AND THERE ARE NO OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTY OF MERCHANTABILITY OR FITNESS FOR PARTICULAR PURPOSE. IN NO EVENT, OMRON WILL BE LIABLE FOR ANY LOST PROFITS OR OTHER INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF USE OF THE SOFTWARE.
   - OMRON SHALL HAVE NO LIABILITY FOR DEFECT OF THE SOFTWARE BASED ON MODIFICATION OR ALTERNATION TO THE SOFTWARE BY THE USER OR ANY THIRD PARTY.
   - OMRON SHALL HAVE NO LIABILITY FOR SOFTWARE DEVELOPED BY THE USER OR ANY THIRD PARTY BASED ON THE SOFTWARE OR ANY CONSEQUENCE THEREOF.

3. APPLICABLE CONDITIONS

   USER SHALL NOT USE THE SOFTWARE FOR THE PURPOSE THAT IS NOT PROVIDED IN THE *SYSMAC STUDIO VERSION 1 OPERATION MANUAL* (CAT. NO. W504).

4. CHANGE IN SPECIFICATION

   The software specifications and accessories may be changed at any time based on improvements and other reasons.

5. EXTENT OF SERVICE

   The license fee of the Software does not include service costs, such as dispatching technical staff.

6. ERRORS AND OMISSIONS

The information in this guide has been carefully checked and is believed to be accurate; however, no responsibility is assumed for clerical, typographical, or proofreading errors, or omissions.

Precautions

- When building a system, check the specifications for all devices and equipment that will make up the system and make sure that the OMRON products are used well within their rated specifications and performances. Safety measures, such as safety circuits, must be implemented in order to minimize the risks in the event of a malfunction.
- Thoroughly read and understand the manuals for all devices and equipment that will make up the system to ensure that the system is used safely. Review the entire contents of these manuals, including all safety precautions, precautions for safe use, and precautions for correct use.
- Confirm all regulations, standards, and restrictions that the system must adhere to.

Trademarks

- Sysmac and SYSMAC are trademarks or registered trademarks of OMRON Corporation in Japan and other countries for OMRON factory automation products.
- Windows, Windows 98, Windows XP, Windows Vista, and Windows 7 are either registered trademarks or trademarks of Microsoft Corporation in the USA and other countries.
- Intel, Intel Logo, and Intel Atom are trademarks of Intel Corporation in the U.S. and/or other countries.
- EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.
- The SD logo is a trademark of SD-3C, LLC.

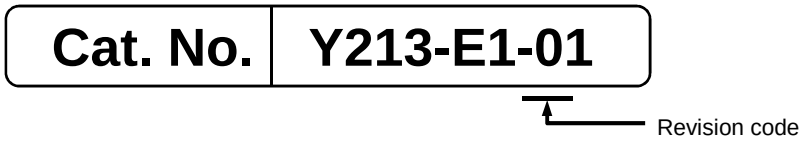
Other company names and product names in this Guide are the trademarks or registered trademarks of their respective companies.

Software Licenses and Copyrights

This product incorporates certain third party software. The license and copyright information associated with this software is available at http://www.fa.omron.co.jp/nj_info_e/.

Revision History

A manual revision code appears as a suffix to the catalog number on the front and back covers of the manual.



Revision code	Date	Revised content
A	December 2015	Original production

CONTENTS

MEMO

1

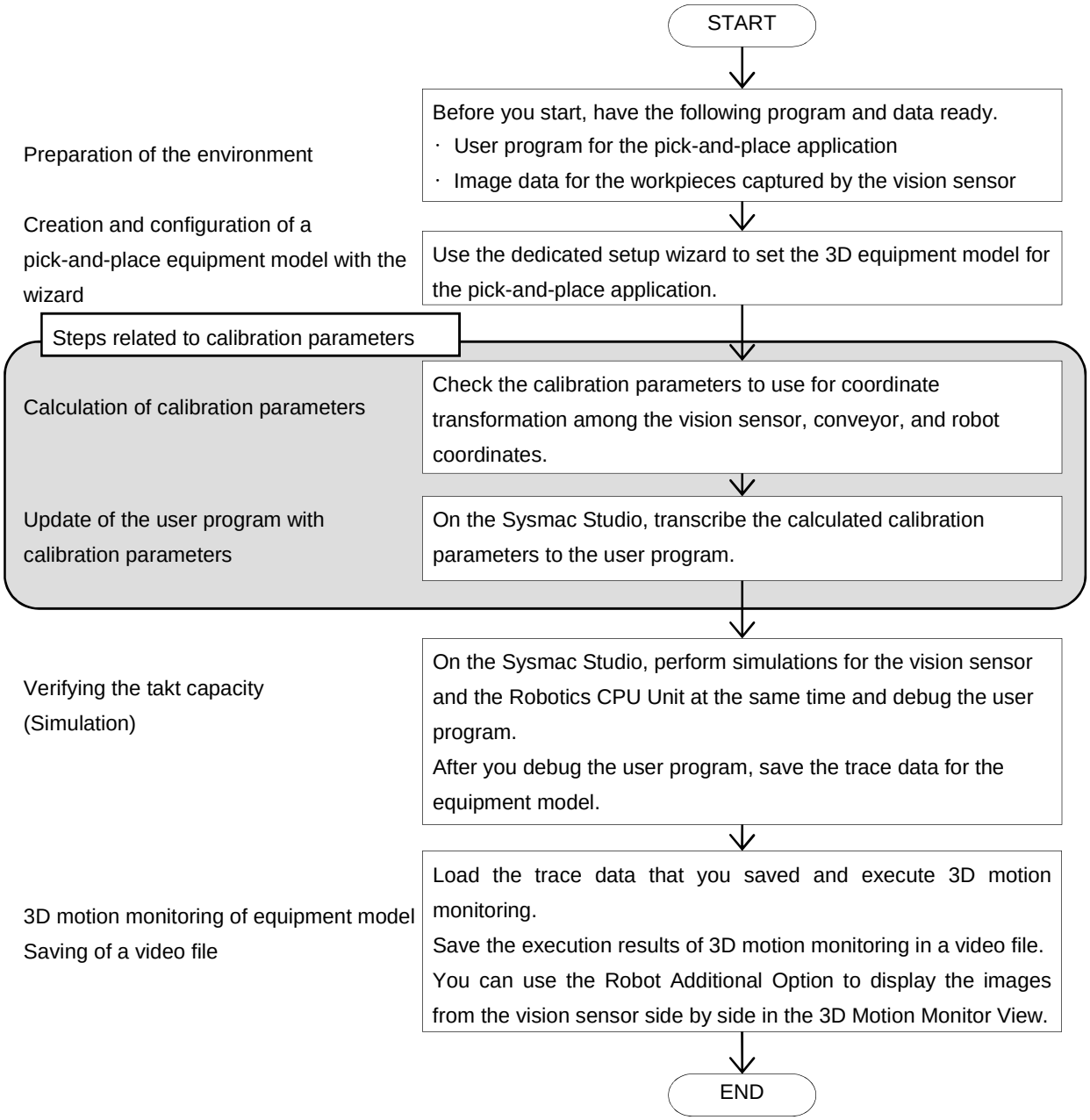
Overview

This section describes the overview of the calibration parameters of the Vision & Robot integrated simulation.

1-1 Overview

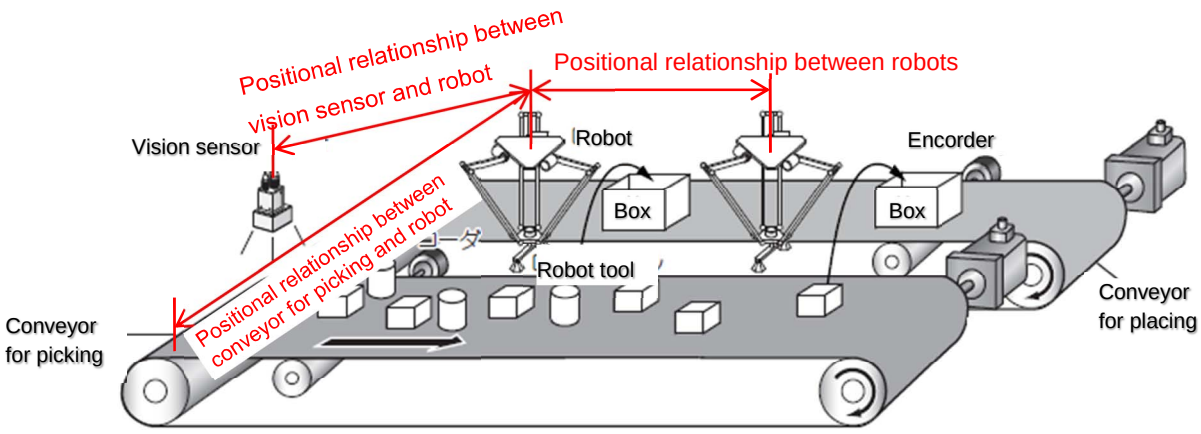
This guide describes the calibration parameters calculated by the Vision & Robot integrated simulation (hereinafter called integrated simulation) that is one of simulations of the Sysmac Studio.

Use this guide for "Calculation of calibration parameters" and "Update of the user program with calibration parameters" in the integrated simulation flow chart.



1-2 Calibration Parameters

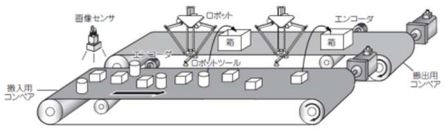
Calibration parameters represent the data of the 3D equipment model for the pick-and-place application, including the positional relationship between the 3D machine models for vision sensors, conveyors, and robots and directions and parameter settings of the 3D machine models.



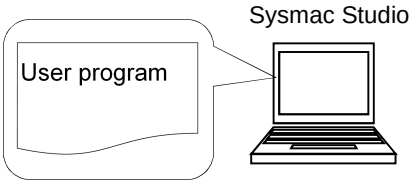
In the actual device environment, calibration parameters can be obtained by operating physical sensors, conveyors, and robots and measuring. In the integrated simulation, calibration parameters are calculated when the 3D equipment model is set using the dedicated setup wizard.

Update of the user program for a pick-and-place application with the calibration parameters allows you to simulate the operation of the pick-and-place application in 3D.

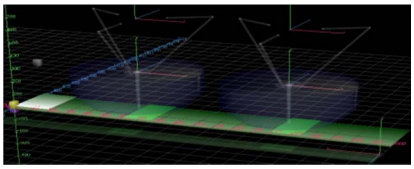
Actual device environment



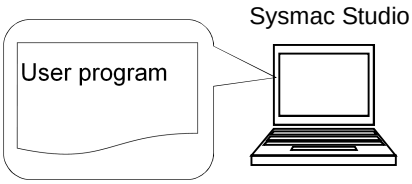
Parameters are obtained by actual operation and measurement.



Integrated simulation

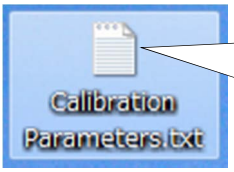
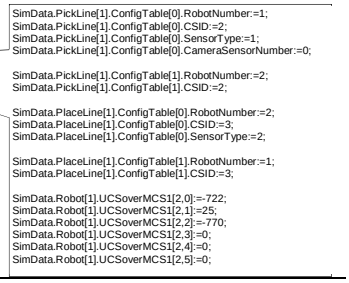


Integrated simulation calculates parameters.



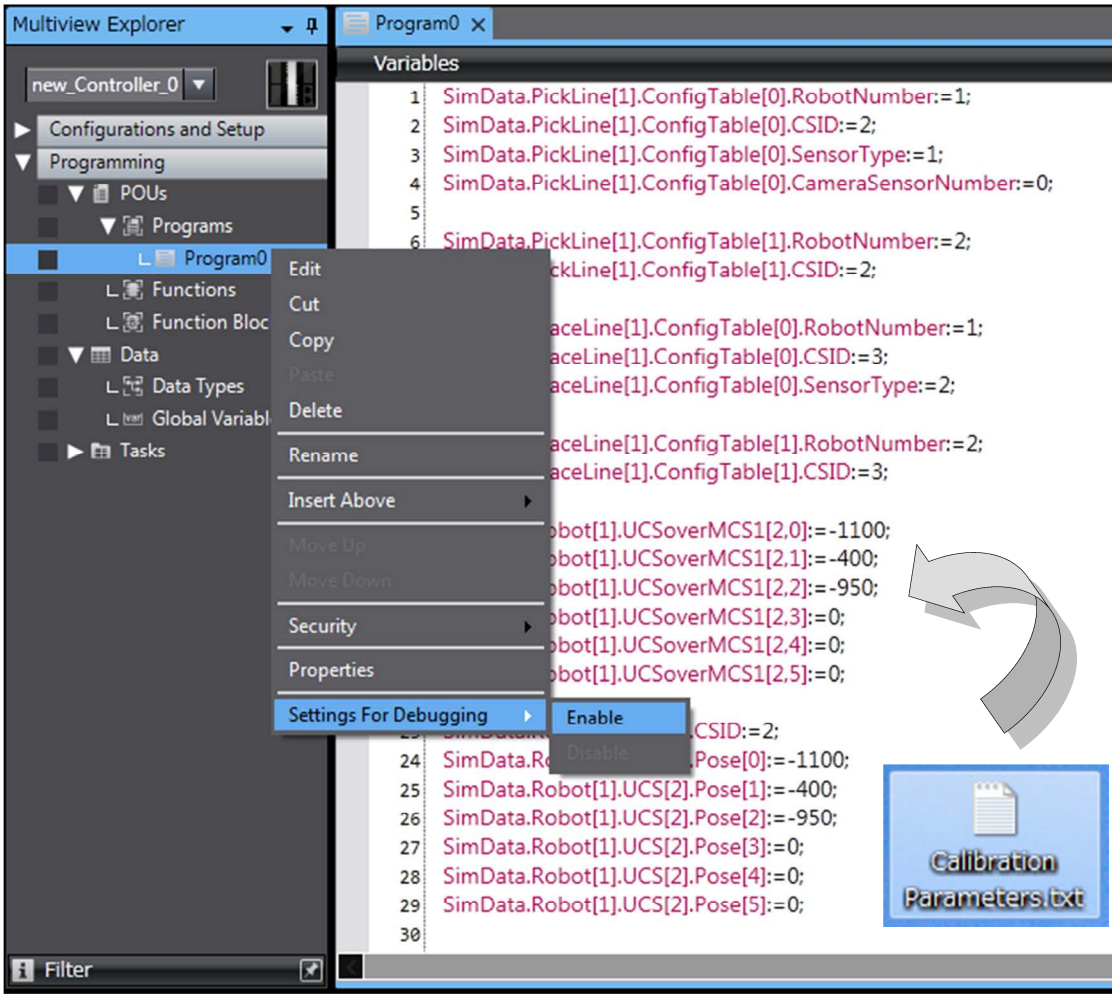
Outputting Calibration Parameter Data

After setting all the 3D machine models in the integrated simulation, click the **Parameter Output (ST program)** Button in the dedicated setup wizard or click the **Output Calibration Parameters** Button in the 3D Motion Monitor View to output the calibration parameters of the 3D equipment model to a text file (*.txt).

After setting with the setup wizard, click Parameter Output (ST Program)... or click  in the 3D Motion Monitor View	 Calibration parameters are saved to a text file. 

Using Calibration Parameter Data

Create a debug program in ST and transcribe the calibration parameters saved in the text file to it.



MEMO

2

Variables and Data Types

This section describes the variables to which the calibration parameters calculated by the Vision & Robot integrated simulation are assigned and the data types of the variables.

2-1 Variables

The calibration parameters calculated by the integrated simulation are output to a text file (*.txt) and used as the program in ST to assign the data to variables

Define the following variables as local or global variables to use the calculated calibration parameters in the user program.

Name	Data type	Description
SimData	sSimData	Calibration parameters calculated by Vision & Robot integrated simulation are assigned to this variable.

2-2 Data Types (Structures)

The data types of the variables to which calibration parameters are assigned are structures. A structure can be the member of another structure.

Define the following structure variables to which calibration parameters are assigned.

- sSimData
- sSimRobot
- sSimOpeRange
- sSimLine
- sLineConfigTable
- sSimVision

sSimData Specifications

The following table shows the specifications of the structure sSimData.

Name	Data type	Description
sSimData	STRUCT	
Robot	ARRAY[0..8] OF sSimRobot	Data related to robots
PickLine	ARRAY[0..16] OF sSimLine	Data related to conveyor for picking
PlaceLine	ARRAY[0..16] OF sSimLine	Data related to conveyor for placing
Vision	ARRAY[0..7] OF sSimVision	Data related to vision sensors

sSimRobot Specifications

The following table shows the specifications of the structure sSimRobot.

Name	Data type	Description
sSimRobot	STRUCT	
UCSoverMCS1	ARRAY[0..15,0..5] OF LREAL	UCS data relative to MCS1
UCS	ARRAY[0..15] OF _sMC_COORD_REF	Data set for UCS
TCS	ARRAY[0..16] OF _sMC_TOOL_COORD_REF	Data set for TCS
KinTransform	_sMC_KIN_REF	Data set for robot kinematics
Workspace	_sMC_WORKSPACE_REF	Data set for robot workspace
PickRange	ARRAY[0..16] OF sSimOpeRange	Data related to start and missing lines of conveyor for picking
PlaceRange	ARRAY[0..16] OF sSimOpeRange	Data related to start and missing lines of conveyor for placing

sSimOpeRange Specifications

The following table shows the specifications of the structure sSimOpeRange

Name	Data type	Description
sSimOpeRange	STRUCT	
StartLineP1	ARRAY[0..2] OF LREAL	X, Y, and Z coordinates of end point 1 of start line
StartLineP2	ARRAY[0..2] OF LREAL	X, Y, and Z coordinates of end point 2 of start line
MissingLineP1	ARRAY[0..2] OF LREAL	X, Y, and Z coordinates of end point 1 of missing line
MissingLineP2	ARRAY[0..2] OF LREAL	X, Y, and Z coordinates of end point 2 of missing line

sSimLine Specifications

The following table shows the specifications of the structure sSimLine.

Name	Data type	Description
sSimLine	STRUCT	
AxisNumber	UINT	Axis number of axis that is assigned to 3D machine model for conveyor for picking or placing
UnitVector	ARRAY[0..2] OF LREAL	Unit vectors of conveyor for picking or placing
CameraSensor Origin	ARRAY[0..5] OF LREAL	Coordinates of origin of vision sensor on conveyor for picking or ON/OFF sensor on conveyor for placing relative to MCS1
MeasureInterval	LREAL	For conveyor for picking, interval at which vision sensor captures images. For conveyor for placing, size of box
MeasureInterval Offset	LREAL	For conveyor for picking, not used. For conveyor for placing, interval between two adjacent boxes.
LineLength	LREAL	Length of conveyor for picking or placing
ConfigTable	ARRAY[0..7] OF sLineConfigTable	Data related to configuration of conveyor for picking or placing

sLineConfigTable Specifications

The following table shows the specifications of the structure sLineConfigTable.

Name	Data type	Description
sLineConfigTable	STRUCT	
RobotNumber	UINT	Robot number
CSID	UINT	CSID of robot
SensorType	UINT	Type of sensor on conveyor for picking or placing 0: Not used 1: Vision sensor 2: ON/OFF sensor
CameraSensor Number	UINT	Line number of vision sensor on conveyor for picking. Not used for conveyor for placing.

sSimVision Specifications

The following table shows the specifications of the structure sSimVision

Name	Data type	Description
sSimVision	STRUCT	
Origin	ARRAY[0..2] OF LREAL	Coordinates of origin of vision sensor relative to MCS1
Dx	LREAL	Travel distance of conveyor relative to MCS1 (X direction)
Dy	LREAL	Travel distance of conveyor relative to MCS1 (Y direction)
Length	LREAL	Length of camera view (X view)
Width	LREAL	Width of camera view (Y view)
TiltAngle	LREAL	Orientation of camera view

3

Parameters

This section describes the data of the calibration parameters.

3-1 Data Related to Robots

This section describes the calibration parameters related to the robot.

A program in ST to assign the following calibration parameter data is output for each robot.

Calibration parameter data	
SimData.Robot[1].UCSoverMCS1[α,0]	Data of the first conveyor for picking
SimData.Robot[1].UCSoverMCS1[α,1]	
SimData.Robot[1].UCSoverMCS1[α,2]	
SimData.Robot[1].UCSoverMCS1[α,3]	
SimData.Robot[1].UCSoverMCS1[α,4]	
SimData.Robot[1].UCSoverMCS1[α,5]	
SimData.Robot[1].UCS[α].CSID	
SimData.Robot[1].UCS[α].Pose[0]	
SimData.Robot[1].UCS[α].Pose[1]	
SimData.Robot[1].UCS[α].Pose[2]	
SimData.Robot[1].UCS[α].Pose[3]	
SimData.Robot[1].UCS[α].Pose[4]	
SimData.Robot[1].UCS[α].Pose[5]	
SimData.Robot[1].UCSoverMCS1[β,0]	Data of the second and following conveyors for picking or placing. Data for each line is output.
SimData.Robot[1].UCSoverMCS1[β,1]	
SimData.Robot[1].UCSoverMCS1[β,2]	
SimData.Robot[1].UCSoverMCS1[β,3]	
SimData.Robot[1].UCSoverMCS1[β,4]	
SimData.Robot[1].UCSoverMCS1[β,5]	
SimData.Robot[1].UCS[β].CSID	
SimData.Robot[1].UCS[β].Pose[0]	
SimData.Robot[1].UCS[β].Pose[1]	
SimData.Robot[1].UCS[β].Pose[2]	
SimData.Robot[1].UCS[β].Pose[3]	
SimData.Robot[1].UCS[β].Pose[4]	
SimData.Robot[1].UCS[β].Pose[5]	
⋮	
⋮	
SimData.Robot[1].UCSoverMCS1[γ,0]	
SimData.Robot[1].UCSoverMCS1[γ,1]	
SimData.Robot[1].UCSoverMCS1[γ,2]	
SimData.Robot[1].UCSoverMCS1[γ,3]	
SimData.Robot[1].UCSoverMCS1[γ,4]	
SimData.Robot[1].UCSoverMCS1[γ,5]	
SimData.Robot[1].UCS[γ].CSID	
SimData.Robot[1].UCS[γ].Pose[0]	
SimData.Robot[1].UCS[γ].Pose[1]	
SimData.Robot[1].UCS[γ].Pose[2]	
SimData.Robot[1].UCS[γ].Pose[3]	
SimData.Robot[1].UCS[γ].Pose[4]	
SimData.Robot[1].UCS[γ].Pose[5]	

Calibration parameter data	
SimData.Robot[1].TCS[1].ToolID	Data of the first robot tool
SimData.Robot[1].TCS[1].Pose[0]	
SimData.Robot[1].TCS[1].Pose[1]	
SimData.Robot[1].TCS[1].Pose[2]	
SimData.Robot[1].TCS[1].Pose[3]	
SimData.Robot[1].TCS[1].Pose[4]	
SimData.Robot[1].TCS[1].Pose[5]	
SimData.Robot[1].TCS[2].ToolID	Data of the second and following robot tools. Data for each robot tool is output.
SimData.Robot[1].TCS[2].Pose[0]	
SimData.Robot[1].TCS[2].Pose[1]	
SimData.Robot[1].TCS[2].Pose[2]	
SimData.Robot[1].TCS[2].Pose[3]	
SimData.Robot[1].TCS[2].Pose[4]	
SimData.Robot[1].TCS[2].Pose[5]	
⋮	
SimData.Robot[1].TCS[16].ToolID	
SimData.Robot[1].TCS[16].Pose[0]	
SimData.Robot[1].TCS[16].Pose[1]	
SimData.Robot[1].TCS[16].Pose[2]	
SimData.Robot[1].TCS[16].Pose[3]	
SimData.Robot[1].TCS[16].Pose[4]	
SimData.Robot[1].TCS[16].Pose[5]	
SimData.Robot[1].KinTransform.KinType	
SimData.Robot[1].KinTransform.KinParam[0]	
SimData.Robot[1].KinTransform.KinParam[1]	
SimData.Robot[1].KinTransform.KinParam[2]	
SimData.Robot[1].KinTransform.KinParam[3]	
SimData.Robot[1].KinTransform.ExpansionParam[0]	
SimData.Robot[1].Workspace.WorkspaceType	
SimData.Robot[1].Workspace.WorkspaceParam[0]	
SimData.Robot[1].Workspace.WorkspaceParam[1]	
SimData.Robot[1].Workspace.WorkspaceParam[2]	
SimData.Robot[1].Workspace.WorkspaceParam[3]	
SimData.Robot[1].Workspace.WorkspaceParam[4]	

Calibration parameter data	
SimData.Robot[1].PickRange[1].StartLineP1[0]	Data of the first conveyor for picking
SimData.Robot[1].PickRange[1].StartLineP1[1]	
SimData.Robot[1].PickRange[1].StartLineP1[2]	
SimData.Robot[1].PickRange[1].StartLineP2[0]	
SimData.Robot[1].PickRange[1].StartLineP2[1]	
SimData.Robot[1].PickRange[1].StartLineP2[2]	
SimData.Robot[1].PickRange[1].MissingLineP1[0]	
SimData.Robot[1].PickRange[1].MissingLineP1[1]	
SimData.Robot[1].PickRange[1].MissingLineP1[2]	
SimData.Robot[1].PickRange[1].MissingLineP2[0]	
SimData.Robot[1].PickRange[1].MissingLineP2[1]	
SimData.Robot[1].PickRange[1].MissingLineP2[2]	
SimData.Robot[1].PickRange[2].StartLineP1[0]	Data of the second and following conveyors for picking. Data for each conveyor for picking is output.
SimData.Robot[1].PickRange[2].StartLineP1[1]	
SimData.Robot[1].PickRange[2].StartLineP1[2]	
SimData.Robot[1].PickRange[2].StartLineP2[0]	
SimData.Robot[1].PickRange[2].StartLineP2[1]	
SimData.Robot[1].PickRange[2].StartLineP2[2]	
SimData.Robot[1].PickRange[2].MissingLineP1[0]	
SimData.Robot[1].PickRange[2].MissingLineP1[1]	
SimData.Robot[1].PickRange[2].MissingLineP1[2]	
SimData.Robot[1].PickRange[2].MissingLineP2[0]	
SimData.Robot[1].PickRange[2].MissingLineP2[1]	
SimData.Robot[1].PickRange[2].MissingLineP2[2]	
⋮	
SimData.Robot[1].PickRange[16].StartLineP1[0]	
SimData.Robot[1].PickRange[16].StartLineP1[1]	
SimData.Robot[1].PickRange[16].StartLineP1[2]	
SimData.Robot[1].PickRange[16].StartLineP2[0]	
SimData.Robot[1].PickRange[16].StartLineP2[1]	
SimData.Robot[1].PickRange[16].StartLineP2[2]	
SimData.Robot[1].PickRange[16].MissingLineP1[0]	
SimData.Robot[1].PickRange[16].MissingLineP1[1]	
SimData.Robot[1].PickRange[16].MissingLineP1[2]	
SimData.Robot[1].PickRange[16].MissingLineP2[0]	
SimData.Robot[1].PickRange[16].MissingLineP2[1]	
SimData.Robot[1].PickRange[16].MissingLineP2[2]	
SimData.Robot[1].PlaceRange[1].StartLineP1[0]	Data of the first conveyor for placing
SimData.Robot[1].PlaceRange[1].StartLineP1[1]	
SimData.Robot[1].PlaceRange[1].StartLineP1[2]	
SimData.Robot[1].PlaceRange[1].StartLineP2[0]	
SimData.Robot[1].PlaceRange[1].StartLineP2[1]	
SimData.Robot[1].PlaceRange[1].StartLineP2[2]	
SimData.Robot[1].PlaceRange[1].MissingLineP1[0]	
SimData.Robot[1].PlaceRange[1].MissingLineP1[1]	
SimData.Robot[1].PlaceRange[1].MissingLineP1[2]	
SimData.Robot[1].PlaceRange[1].MissingLineP2[0]	
SimData.Robot[1].PlaceRange[1].MissingLineP2[1]	
SimData.Robot[1].PlaceRange[1].MissingLineP2[2]	

Calibration parameter data	
SimData.Robot[1].PlaceRange[2].StartLineP1[0]	Data of the second and following conveyors for placing. Data for each conveyor for placing is output.
SimData.Robot[1].PlaceRange[2].StartLineP1[1]	
SimData.Robot[1].PlaceRange[2].StartLineP1[2]	
SimData.Robot[1].PlaceRange[2].StartLineP2[0]	
SimData.Robot[1].PlaceRange[2].StartLineP2[1]	
SimData.Robot[1].PlaceRange[2].StartLineP2[2]	
SimData.Robot[1].PlaceRange[2].MissingLineP1[0]	
SimData.Robot[1].PlaceRange[2].MissingLineP1[1]	
SimData.Robot[1].PlaceRange[2].MissingLineP1[2]	
SimData.Robot[1].PlaceRange[2].MissingLineP2[0]	
SimData.Robot[1].PlaceRange[2].MissingLineP2[1]	
SimData.Robot[1].PlaceRange[2].MissingLineP2[2]	
⋮	
SimData.Robot[1].PlaceRange[16].StartLineP1[0]	
SimData.Robot[1].PlaceRange[16].StartLineP1[1]	
SimData.Robot[1].PlaceRange[16].StartLineP1[2]	
SimData.Robot[1].PlaceRange[16].StartLineP2[0]	
SimData.Robot[1].PlaceRange[16].StartLineP2[1]	
SimData.Robot[1].PlaceRange[16].StartLineP2[2]	
SimData.Robot[1].PlaceRange[16].MissingLineP1[0]	
SimData.Robot[1].PlaceRange[16].MissingLineP1[1]	
SimData.Robot[1].PlaceRange[16].MissingLineP1[2]	
SimData.Robot[1].PlaceRange[16].MissingLineP2[0]	
SimData.Robot[1].PlaceRange[16].MissingLineP2[1]	
SimData.Robot[1].PlaceRange[16].MissingLineP2[2]	

* You can specify a number between 0 and 15 for α, β, or γ.

The data of the second and following robots is output as follows:

SimData.Robot[2].***** : Data of the second robot
SimData.Robot[3].***** : Data of the third robot
SimData.Robot[4].***** : Data of the fourth robot
⋮

Details of SimData.Robot[1].UCSoverMCS1[α,0-5]

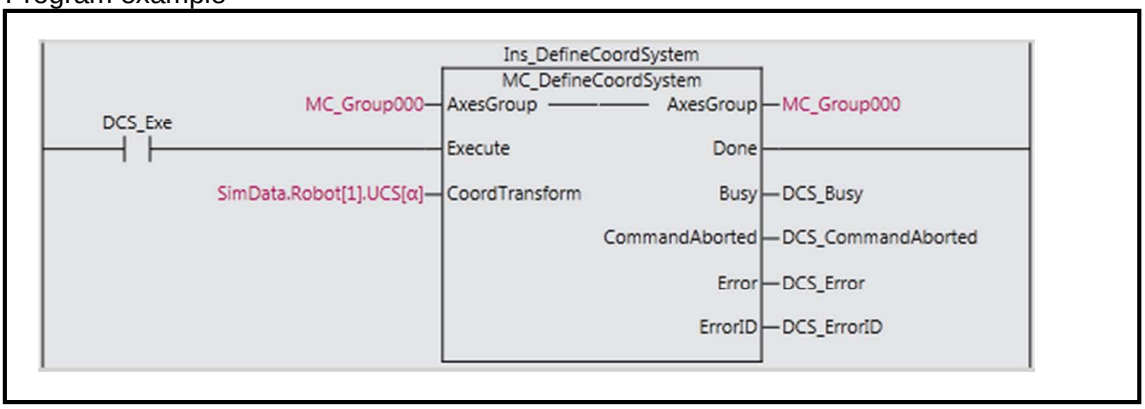
The values of the UCS relative to the MCS1 are assigned.
Use these variables to transform the position in the UCS into a position in the MCS1 and vice versa.

The program example for transforming is skipped.

Details of SimData.Robot[1].UCS[α].CSID
SimData.Robot[1].UCS[α].Pose[0-5]

The values to define the user coordinate system (UCS) are assigned.
Set the user coordinate system (UCS) by using the MC_DefineCoordSystem (Define Coordinate) instruction.

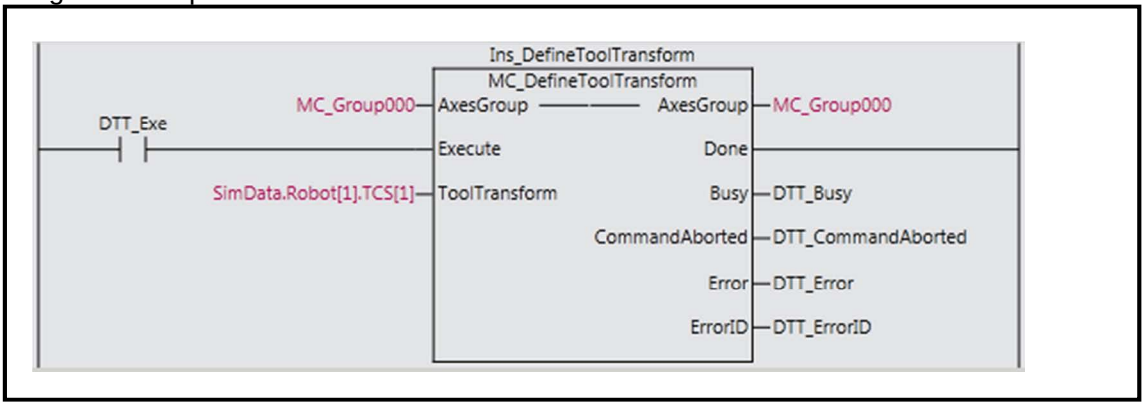
Program example



Details of SimData.Robot[1].TCS[1].ToolID
SimData.Robot[1].TCS[1].Pose[0-5]

The values to define the tool coordinate system (TCS) are assigned.
Set the tool coordinate system (TCS) by using the MC_DefineToolTransform (Define Tool Coordinate) instruction.

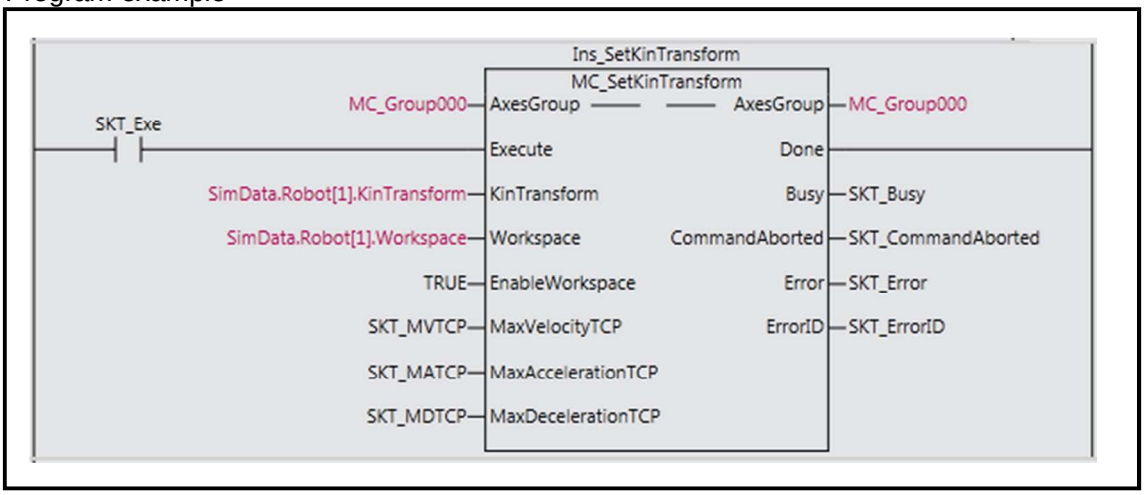
Program example



Details of SimData.Robot[1].KinTransform.KinType
SimData.Robot[1].KinTransform.KinParam[0-3]
SimData.Robot[1].KinTransform.ExpansionParam[0]

The values (parameters) to set the kinematics are assigned.
Set the parameters for kinematics transformation by using the MC_SetKinTransform (Set Kinematics Transformation) instruction.

Program example



Details of SimData.Robot[1].Workspace.WorkspaceType
SimData.Robot[1].Workspace.WorkspaceParam[0-4]

The values (parameters) to set the workspace to check are assigned.
Set the workspace by using the MC_SetKinTransform (Set Kinematics Transformation) instruction.

Refer to the *Details of SimData.Robot[1].KinTransform* for the program example.

Details of SimData.Robot[1].PickRange[1].StartLineP1[0-2] SimData.Robot[1].PickRange[1].StartLineP2[0-2]

The values to define the start line (coordinate values of two end points) are assigned.
The coordinate values are based on the machine coordinate systems (MCS) of each robot.

- SimData.Robot[1].PickRange[1].StartLineP1[0]: X coordinate of the first end point
- SimData.Robot[1].PickRange[1].StartLineP1[1]: Y coordinate of the first end point
- SimData.Robot[1].PickRange[1].StartLineP1[2]: Z coordinate of the first end point
- SimData.Robot[1].PickRange[1].StartLineP2[0]: X coordinate of the second end point
- SimData.Robot[1].PickRange[1].StartLineP2[1]: Y coordinate of the second end point
- SimData.Robot[1].PickRange[1].StartLineP2[2]: Z coordinate of the second end point

Create a program to start the robot to pick when the current position of a workpiece to pick passes through the straight line defined by two end points (start line).
When using the current position of a workpiece to pick in the user coordinate system or machine coordinate system of Robot 1 (MCS1), transform the values of these variables into the values in the same coordinate system as the current position of the workpiece.

The program example is skipped.

Details of SimData.Robot[1].PickRange[1].MissingLineP1[0-2] SimData.Robot[1].PickRange[1].MissingLineP2[0-2]

The values to define the missing line (coordinate values of two end points) of the conveyor for picking are assigned.
The coordinate values are based on the machine coordinate systems (MCS) of each robot.

- SimData.Robot[1].PickRange[1].MissingLineP1[0]: X coordinate of the first end point
- SimData.Robot[1].PickRange[1].MissingLineP1[1]: Y coordinate of the first end point
- SimData.Robot[1].PickRange[1].MissingLineP1[2]: Z coordinate of the first end point
- SimData.Robot[1].PickRange[1].MissingLineP2[0]: X coordinate of the second end point
- SimData.Robot[1].PickRange[1].MissingLineP2[1]: Y coordinate of the second end point
- SimData.Robot[1].PickRange[1].MissingLineP2[2]: Z coordinate of the second end point

Create a program to stop the robot to pick when a picked workpiece passes through the straight line defined by two end points (missing line).
When using the current position of the picked workpiece in the user coordinate system or machine coordinate system of Robot 1 (MCS1), transform the values of these variables into the values in the same coordinate system as the current position of the workpiece.

The program example is skipped.

Details of SimData.Robot[1].PlaceRange[1].StartLineP1[0-2] SimData.Robot[1].PlaceRange[1].StartLineP2[0-2]

The values to define the start line (coordinate values of two end points) are assigned.
The coordinate values are based on the machine coordinate systems (MCS) of each robot.

- SimData.Robot[1].PlaceRange[1].StartLineP1[0]: X coordinate of the first end point
- SimData.Robot[1].PlaceRange[1].StartLineP1[1]: Y coordinate of the first end point
- SimData.Robot[1].PlaceRange[1].StartLineP1[2]: Z coordinate of the first end point
- SimData.Robot[1].PlaceRange[1].StartLineP2[0]: X coordinate of the second end point
- SimData.Robot[1].PlaceRange[1].StartLineP2[1]: Y coordinate of the second end point
- SimData.Robot[1].PlaceRange[1].StartLineP2[2]: Z coordinate of the second end point

Create a program to start the robot to place when a border or box on the conveyor for placing passes through the straight line defined by two end points (start line).
When using the position to place a workpiece in the user coordinate system or machine coordinate system of Robot 1 (MCS1), transform the values of these variables into the values in the same coordinate system as the position to place the workpiece.

The program example is skipped.

Details of SimData.Robot[1].PlaceRange[1].MissingLineP1[0-2] SimData.Robot[1].PlaceRange[1].MissingLineP2[0-2]

The values to define the missing line (coordinate values of two end points) of the conveyor for picking are assigned.
The coordinate values are based on the machine coordinate systems (MCS) of each robot.

- SimData.Robot[1].PlaceRange[1].MissingLineP1[0]:X coordinate of the first end point
- SimData.Robot[1].PlaceRange[1].MissingLineP1[1]:Y coordinate of the first end point
- SimData.Robot[1].PlaceRange[1].MissingLineP1[2]:Z coordinate of the first end point
- SimData.Robot[1].PlaceRange[1].MissingLineP2[0]:X coordinate of the second end point
- SimData.Robot[1].PlaceRange[1].MissingLineP2[1]:Y coordinate of the second end point
- SimData.Robot[1].PlaceRange[1].MissingLineP2[2]:Z coordinate of the second end point

Create a program to stop the robot to place when a border or box on the conveyor for placing passes through the straight line defined by two end points (missing line).
When using the position to place a workpiece in the user coordinate system or machine coordinate system of Robot 1 (MCS1), transform the values of these variables into the values in the same coordinate system as the position to place the workpiece.

The program example is skipped.

3-2 Data Related to Conveyors for Picking

This section describes the calibration parameters related to the conveyor for picking.

A program in ST to assign the following calibration parameter data is output for each conveyor for picking.

Calibration parameter data	
SimData.PickLine[1].AxisNumber	
SimData.PickLine[1].UnitVector[0] SimData.PickLine[1].UnitVector[1] SimData.PickLine[1].UnitVector[2]	
SimData.PickLine[1].CameraSensorOrigin[0] SimData.PickLine[1].CameraSensorOrigin[1] SimData.PickLine[1].CameraSensorOrigin[2] SimData.PickLine[1].CameraSensorOrigin[3] SimData.PickLine[1].CameraSensorOrigin[4] SimData.PickLine[1].CameraSensorOrigin[5]	
SimData.PickLine[1].MeasureInterval	
SimData.PickLine[1].MeasureIntervalOffset	
SimData.PickLine[1].LineLength	
SimData.PickLine[1].ConfigTable[0].RobotNumber SimData.PickLine[1].ConfigTable[0].CSID SimData.PickLine[1].ConfigTable[0].SensorType SimData.PickLine[1].ConfigTable[0].CameraSensorNumber	Data of the first robot
SimData.PickLine[1].ConfigTable[1].RobotNumber SimData.PickLine[1].ConfigTable[1].CSID	Data of the second and following robots Data for each robot is output
SimData.PickLine[1].ConfigTable[2].RobotNumber SimData.PickLine[1].ConfigTable[2].CSID	
⋮	
SimData.PickLine[1].ConfigTable[7].RobotNumber SimData.PickLine[1].ConfigTable[7].CSID	

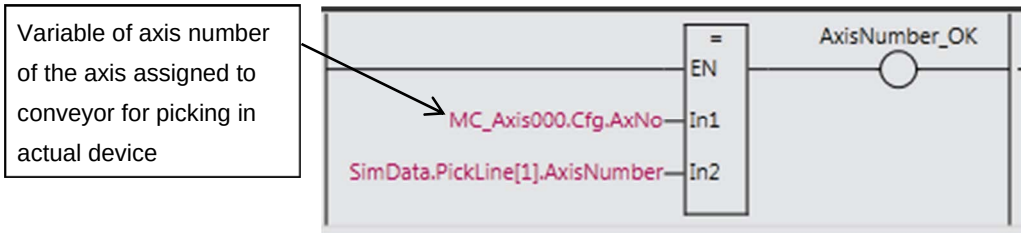
The data of the second and following conveyors for picking are output as follows:

SimData.PickLine[2].***** : Data of the second conveyor for picking
SimData.PickLine[3].***** : Data of the third conveyor for picking
SimData.PickLine[4].***** : Data of the fourth conveyor for picking
⋮

Details of SimData.PickLine[1].AxisNumber

The axis number of the axis that is assigned to the 3D machine model for the conveyor for picking is assigned.
Create an interlock program to check if the axis number obtained from the variable of the axis assigned to the conveyor for picking in the actual device environment is the same as this variable.

Program example



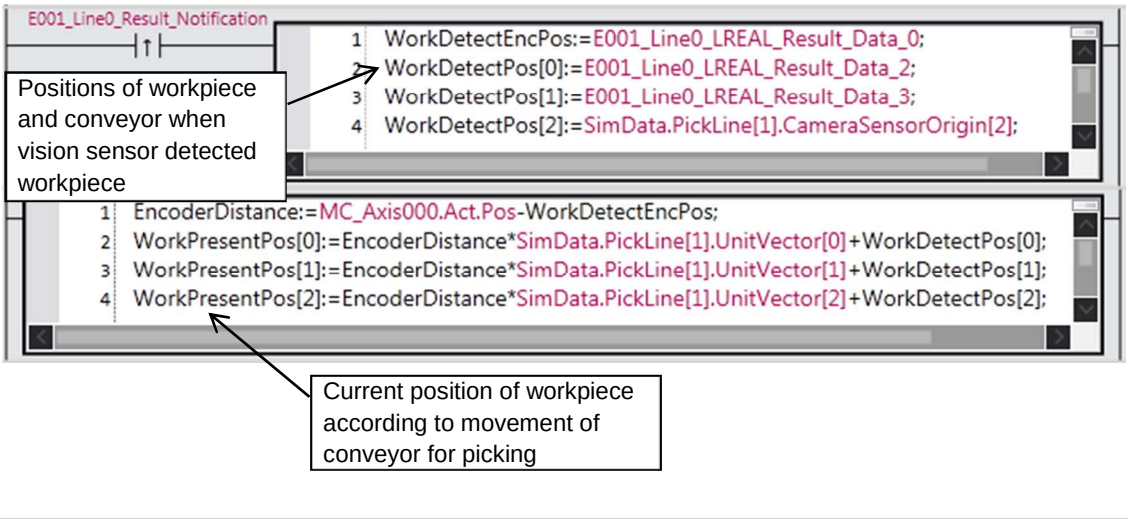
Details of SimData.PickLine[1].UnitVector[0-2]

The unit vector of the conveyor for picking (travel direction of the conveyor for picking relative to MCS1) is assigned.

- SimData.PickLine[1].UnitVector[0] : X-axis component of unit vector
- SimData.PickLine[1].UnitVector[1] : Y-axis component of unit vector
- SimData.PickLine[1].UnitVector[2] : Z-axis component of unit vector

Create a program to calculate the current position of a workpiece from the workpiece position detected by the vision sensor, travel distance of the conveyor for picking, and this variable.

Program example



Details of SimData.PickLine[1].CameraSensorOrigin[0-5]

Coordinates of the origin of the vision sensor on the conveyor for picking relative to MCS1 are assigned.

- SimData.PickLine[1].CameraSensorOrigin[0] : X coordinate
- SimData.PickLine[1].CameraSensorOrigin[1] : Y coordinate
- SimData.PickLine[1].CameraSensorOrigin[2] : Z coordinate
- SimData.PickLine[1].CameraSensorOrigin[3] : Rx coordinate
- SimData.PickLine[1].CameraSensorOrigin[4] : Ry coordinate
- SimData.PickLine[1].CameraSensorOrigin[5] : Rz coordinate

Create a program to use the z coordinate of this variable when the z coordinate of the workpiece position detected by the vision sensor is determined.

Refer to the *Details of SimData.PickLine[1].UnitVector[0-2]* for the program example.

Details of SimData.PickLine[1].MeasureInterval

The interval at which the vision sensor captures images is assigned.
Create a program to turn ON the measurement trigger for the vision sensor whenever the current position of the conveyor for picking increases by the value of this variable.

Program example

```
1 IF MC_Axis000.Act.Pos>=TriggerEncPos+SimData.PickLine[1].MeasureInterval THEN
2     E001_Line0_Trigger:=TRUE;
3     TriggerEncPos:=MC_Axis000.Act.Pos;
4 END_IF;
```

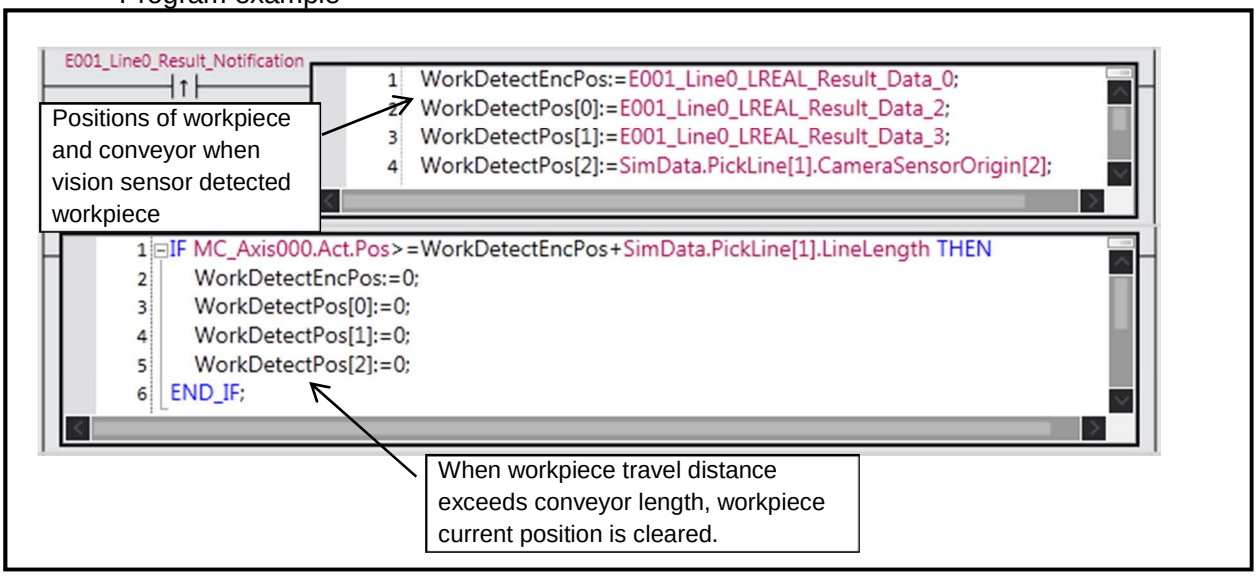
Details of SimData.PickLine[1].MeasureIntervalOffset

This calibration parameter is used for the conveyor for placing and not used for the conveyor for picking.

Details of SimData.PickLine[1].LineLength

The length of the conveyor for picking is assigned.
Create a program to finish the calculation of the current position of the workpiece detected by the vision sensor when the travel distance of the detected workpiece exceeds the value of this variable.

Program example



Details of SimData.PickLine[1].ConfigTable[0].RobotNumber

The robot number of the first robot on the conveyor for picking is assigned.
No program for this variable is required.

Details of SimData.PickLine[1].ConfigTable[0].CSID

The CSID of the first robot on the conveyor for picking is assigned.
No program for this variable is required.

Details of SimData.PickLine[1].ConfigTable[0].SensorType

The type of the sensor on the conveyor for picking is assigned.
1: Vision sensor is always assigned.
No program for this variable is required.

Details of SimData.PickLine[1].ConfigTable[0].CameraSensorNumber

The sensor number of the sensor on the conveyor for picking (line number of the vision sensor) is assigned.
No program for this variable is required.

3-3 Data Related to Conveyors for Placing

This section describes the calibration parameters related to the conveyor for placing.

A program in ST to assign the following calibration parameter data is output for each conveyor for placing.

Calibration parameter data	
SimData.PlaceLine[1].AxisNumber	
SimData.PlaceLine[1].UnitVector[0] SimData.PlaceLine[1].UnitVector[1] SimData.PlaceLine[1].UnitVector[2]	
SimData.PlaceLine[1].CameraSensorOrigin[0] SimData.PlaceLine[1].CameraSensorOrigin[1] SimData.PlaceLine[1].CameraSensorOrigin[2] SimData.PlaceLine[1].CameraSensorOrigin[3] SimData.PlaceLine[1].CameraSensorOrigin[4] SimData.PlaceLine[1].CameraSensorOrigin[5]	
SimData.PlaceLine[1].MeasureInterval	
SimData.PlaceLine[1].MeasureIntervalOffset	
SimData.PlaceLine[1].LineLength	
SimData.PlaceLine[1].ConfigTable[0].RobotNumber SimData.PlaceLine[1].ConfigTable[0].CSID SimData.PlaceLine[1].ConfigTable[0].SensorType SimData.PlaceLine[1].ConfigTable[0].CameraSensorNumber	Data of the first robot
SimData.PlaceLine[1].ConfigTable[1].RobotNumber SimData.PlaceLine[1].ConfigTable[1].CSID ⋮	Data of the second and following robots Data for each robot is output
SimData.PlaceLine[1].ConfigTable[7].RobotNumber SimData.PlaceLine[1].ConfigTable[7].CSID	

The data of the second and following conveyors for placing are output as follows:

SimData.PlaceLine[2].***** : Data of the second conveyor for placing
SimData.PlaceLine[3].***** : Data of the third conveyor for placing
SimData.PlaceLine[4].***** : Data of the fourth conveyor for placing
⋮

The following data of the conveyor for placing is the same data as the conveyor for picking. Refer to the *3-2 Data Related to Conveyor for Picking* for the details of the data and program examples.

- SimData.PlaceLine[1].AxisNumber
- SimData.PlaceLine[1].UnitVector[0]
- SimData.PlaceLine[1].UnitVector[1]
- SimData.PlaceLine[1].UnitVector[2]
- SimData.PlaceLine[1].CameraSensorOrigin[0]
- SimData.PlaceLine[1].CameraSensorOrigin[1]
- SimData.PlaceLine[1].CameraSensorOrigin[2]
- SimData.PlaceLine[1].CameraSensorOrigin[3]
- SimData.PlaceLine[1].CameraSensorOrigin[4]
- SimData.PlaceLine[1].CameraSensorOrigin[5]
- SimData.PlaceLine[1].LineLength
- SimData.PlaceLine[1].ConfigTable[0].RobotNumber
- SimData.PlaceLine[1].ConfigTable[0].CSID
- SimData.PlaceLine[1].ConfigTable[1].RobotNumber
- SimData.PlaceLine[1].ConfigTable[1].CSID

  |

  |

  |
- SimData.PlaceLine[1].ConfigTable[7].RobotNumber
- SimData.PlaceLine[1].ConfigTable[7].CSID

The following data is different from the data of the conveyor for picking. The details of the data and program examples are described later.

- SimData.PlaceLine[1].MeasureInterval
- SimData.PlaceLine[1].MeasureIntervalOffset
- SimData.PlaceLine[1].ConfigTable[0].SensorType

The following data is not output for the conveyor for placing.

- SimData.PlaceLine[1].ConfigTable[0].CameraSensorNumber

Details of SimData.PlaceLine[1].MeasureInterval

When the type of the conveyor for placing is Conveyor for placing (with borders), the value of Border line: Interval is assigned. When the type is Conveyor for placing (Box), the value of Box: Length is assigned.
Create a program to turn ON the measurement trigger for the vision sensor whenever the current position of the conveyor for placing increases by the total value of this variable and the SimData.PlaceLine[1].MeasureIntervalOffset variable.

Program example

```
1 IF MC_Axis001.Act.Pos>=TriggerEncPos+SimData.PlaceLine[1].MeasureInterval
2     + SimData.PlaceLine[1].MeasureIntervalOffset THEN
3     PlaceLine_Trigger:=TRUE;
4     TriggerEncPos:=MC_Axis001.Act.Pos;
5 END_IF;
```

Details of SimData.PlaceLine[1].MeasureIntervalOffset

When the type of the conveyor for placing is Conveyor for placing (with borders), a fixed value is assigned. When the type is Conveyor for placing (Box), the value of Box: Interval is assigned.
Create a program to turn ON the measurement trigger for the vision sensor whenever the current position of the conveyor for placing increases by the total value of this variable and the SimData.PlaceLine[1].MeasureInterval variable.

Refer to the *Details of SimData.PlaceLine[1].MeasureInterval* for the program example.

Details of SimData.PlaceLine[1].ConfigTable[0].SensorType

The type of the sensor on the conveyor for placing is assigned.
2: ON/OFF sensor is always assigned.
No program for this variable is required.

3-4 Data Related to Vision Sensors

This section describes the calibration parameters related to the vision sensor.

A program in ST to assign the following calibration parameter data is output for each sensor.

Calibration parameter data
SimData.Vision[0].Origin[0]
SimData.Vision[0].Origin[1]
SimData.Vision[0].Origin[2]
SimData.Vision[0].Dx
SimData.Vision[0].Dy
SimData.Vision[0].Length
SimData.Vision[0].Width
SimData.Vision[0].TiltAngle

The data of the second and following vision sensors is output as follows:

SimData.Vision[1].***** : Data of the second vision sensor
SimData.Vision[2].***** : Data of the third vision sensor
SimData.Vision[3].***** : Data of the fourth vision sensor
|
|
|

Set the calibration parameter data related to the vision sensor to the processing unit data using the communication command of the vision sensor.
A communication command macro example of vision sensor and program example for the CPU unit are described later.

Details of SimData.Vision[0].Origin[0-2]

Coordinates of the origin of the vision sensor relative to the MCS1 are assigned.

- SimData.Vision[0].Origin[0] : X coordinate
- SimData.Vision[0].Origin[1] : Y coordinate
- SimData.Vision[0].Origin[2] : Z coordinate

Details of SimData.Vision[0].Dx SimData.Vision[0].Dy

The unit travel distance of the conveyor relative to the MCS1 is assigned.

- SimData.Vision[0].Dx : X-axis component of unit travel distance
- SimData.Vision[0].Dy : Y-axis component of unit travel distance

Details of SimData.Vision[0].Length SimData.Vision[0].Width

The size (length and width) of the vision sensor view is assigned.

- SimData.Vision[0].Length : Length of vision Sensor view (X-direction view)
- SimData.Vision[0].Width : Width of vision Sensor view (Y-direction view)

Details of SimData.Vision[0].TiltAngle

The orientation of the vision sensor view is assigned.

3-4-1 Communication Command Macro Example of Vision Sensor

This section describes the example of communication command macro to set the calibration parameters sent from the CPU unit to the processing unit data of the vision sensor.

Command name	CMD0128
Function name	FUNC_0128
BUSY ON	ON

```
Dim READ_DATA&(7)
Dim BUF&(1)

COMMUNICATION_AREA& = 0
CHANNEL_OFFSET& = 10
CHANNEL_COUNT& = 16

PI# = 3.14159265358979

ERROR& = 0

GetSystemData "IoModule", "ioIdent3", IO_MODULE$
If UCase$(IO_MODULE$) = "ETHERCAT" Then

  GetSystemData IO_MODULE$, "_userInputDataSize", IO_SIZE&

  If IO_SIZE& = -1 Then
    GetSystemData IO_MODULE$, "userInputDataSize0", IO_SIZE&
  EndIf

  If IO_SIZE& <> 1048592 Then
    RESPONSECODE& = -1
    RESPONSEVALUE&(0) = 0
    Return
  EndIf

Else
  RESPONSECODE& = -1
  RESPONSEVALUE&(0) = 0
  Return
EndIf

If ARGUMENTVALUE#(0) = 0 Then

  I& = 0

  Do
    Try
      GetImageWindow I&, TMP_X&, TMP_Y&, TMP_WIDTH&, TMP_HEIGHT&, TMP_UNIT_NO&,
TMP_SUB_NO&, TMP_MAG&, TMP_ORIGN_X&, TMP_ORIGN_Y&, TMP_UPDATE&, TMP_VISIBLE&
      SetImageWindow I&, TMP_X&, TMP_Y&, TMP_WIDTH&, TMP_HEIGHT&, TMP_UNIT_NO&,
TMP_SUB_NO&, TMP_MAG&, TMP_ORIGN_X&, TMP_ORIGN_Y&, 0, TMP_VISIBLE&
    Catch
      Exit Do
    End Try

    I& = I& + 1
  Loop While (1)
```

```
   RESPONSECODE& = 0
   RESPONSEVALUE&(0) = 0

Elseif ARGUMENTVALUE#(0) = 1 Then

   FOV_ORIGN_X# = 0
   FOV_ORIGN_Y# = 0

   CONVEYOR_MOVE_X# = 0
   CONVEYOR_MOVE_Y# = 0

   FOV_WIDTH# = 0
   FOV_HEIGHT# = 0

   FOV_ANGLE# = 0

   Dim ERROR_FLAG&(4)
   For I& = 0 To 4
      ERROR_FLAG&(I&) = 1
   Next

   RESPONSECODE& = 0
   RESPONSEVALUE&(0) = 0

Elseif ARGUMENTVALUE#(0) = 2 Then

   ReadPlcMemory IO_MODULE$, COMMUNICATION_AREA&, CHANNEL_OFFSET&,
CHANNEL_COUNT&, READ_DATA&()

   BUF&(0) = READ_DATA&(4)
   BUF&(1) = READ_DATA&(5)
   GetPlcData IO_MODULE$, BUF&(), 0, 8, BUF#
   FOV_ORIGN_X# = BUF#

   BUF&(0) = READ_DATA&(6)
   BUF&(1) = READ_DATA&(7)
   GetPlcData IO_MODULE$, BUF&(), 0, 8, BUF#
   FOV_ORIGN_Y# = BUF#

   ERROR_FLAG&(0) = 0

   RESPONSECODE& = 0
   RESPONSEVALUE&(0) = 0

Elseif ARGUMENTVALUE#(0) = 3 Then

   ReadPlcMemory IO_MODULE$, COMMUNICATION_AREA&, CHANNEL_OFFSET&,
CHANNEL_COUNT&, READ_DATA&()

   BUF&(0) = READ_DATA&(4)
   BUF&(1) = READ_DATA&(5)
   GetPlcData IO_MODULE$, BUF&(), 0, 8, BUF#
   CONVEYOR_MOVE_X# = BUF#

   BUF&(0) = READ_DATA&(6)
   BUF&(1) = READ_DATA&(7)
   GetPlcData IO_MODULE$, BUF&(), 0, 8, BUF#
   CONVEYOR_MOVE_Y# = BUF#
```

```
   ERROR_FLAG&(1) = 0

   RESPONSECODE& = 0
   RESPONSEVALUE&(0) = 0

Elseif ARGUMENTVALUE#(0) = 4 Then

   ReadPlcMemory IO_MODULE$, COMMUNICATION_AREA&, CHANNEL_OFFSET&,
CHANNEL_COUNT&, READ_DATA&()

   BUF&(0) = READ_DATA&(4)
   BUF&(1) = READ_DATA&(5)
   GetPlcData IO_MODULE$, BUF&(), 0, 8, BUF#
   FOV_WIDTH# = BUF#

   BUF&(0) = READ_DATA&(6)
   BUF&(1) = READ_DATA&(7)
   GetPlcData IO_MODULE$, BUF&(), 0, 8, BUF#
   FOV_HEIGHT# = BUF#

   ERROR_FLAG&(2) = 0

   RESPONSECODE& = 0
   RESPONSEVALUE&(0) = 0

Elseif ARGUMENTVALUE#(0) = 5 Then

   ReadPlcMemory IO_MODULE$, COMMUNICATION_AREA&, CHANNEL_OFFSET&,
CHANNEL_COUNT&, READ_DATA&()

   BUF&(0) = READ_DATA&(4)
   BUF&(1) = READ_DATA&(5)
   GetPlcData IO_MODULE$, BUF&(), 0, 8, BUF#
   FOV_ANGLE# = BUF#

   ERROR_FLAG&(3) = 0

   RESPONSECODE& = 0
   RESPONSEVALUE&(0) = 0

Elseif ARGUMENTVALUE#(0) = 6 Then

   If (ERROR_FLAG&(2) = 0) Then
      If (FOV_WIDTH# <= 0) OR (FOV_HEIGHT# <= 0) Then
         For I& = 0 To 4
            ERROR& = ERROR& + ERROR_FLAG&(I&) * 2 ^ I&
         Next

         RESPONSECODE& = -1
         RESPONSEVALUE&(0) = ERROR&
         Return
      EndIf
   EndIf

   If (ERROR_FLAG&(3) = 0) Then

      If 0 <= FOV_ANGLE# AND FOV_ANGLE# <= 3 Then
         If CONVEYOR_MOVE_X# <> 0 Then
```

```
        ANGLE# = Atn(CONVEYOR_MOVE_Y#/ CONVEYOR_MOVE_X#) * 180 / PI# + FOV_ANGLE# * 90
        If CONVEYOR_MOVE_X# < 0 Then
        ANGLE# = ANGLE# + 180
        EndIf
        If CONVEYOR_MOVE_X# > 0 AND CONVEYOR_MOVE_Y# < 0 Then
        ANGLE# = ANGLE# + 360
        EndIf
      Else
        ANGLE# = CONVEYOR_MOVE_Y# * 90 + FOV_ANGLE# * 90
        If CONVEYOR_MOVE_Y# < 0 Then
        ANGLE# = ANGLE# + 360
        EndIf
      EndIf
    Else
      For I& = 0 To 4
        ERROR& = ERROR& + ERROR_FLAG&(I&) * 2 ^ I&
      Next

      RESPONSECODE& = -1
      RESPONSEVALUE&(0) = ERROR&
      Return
    EndIf

    If ANGLE# > 180 Then
      RADIAN# = (ANGLE# - 360) / 180 * PI#
    Elseif ANGLE# <= -180 Then
      ANGLE# =
      RADIAN# = (ANGLE# + 360) / 180 * PI#
    Else
      RADIAN# = ANGLE# / 180 * PI#
    EndIf

    RBT_X0# = FOV_ORIGN_X#
    RBT_X1# = FOV_ORIGN_X# + Cos(RADIAN#) * FOV_WIDTH#
    RBT_X2# = FOV_ORIGN_X# - Sin(RADIAN#) * (-FOV_HEIGHT#)
    RBT_X3# = FOV_ORIGN_X# + Cos(RADIAN#) * FOV_WIDTH# -Sin(RADIAN#) * (-FOV_HEIGHT#)
    RBT_Y0# = FOV_ORIGN_Y#
    RBT_Y1# = FOV_ORIGN_Y# + Sin(RADIAN#) * FOV_WIDTH#
    RBT_Y2# = FOV_ORIGN_Y# + Cos(RADIAN#) * (-FOV_HEIGHT#)
    RBT_Y3# = FOV_ORIGN_Y# + Sin(RADIAN#) * FOV_WIDTH# + Cos(RADIAN#) * (-FOV_HEIGHT#)

  EndIf

  I& = 0
  COUNTER& = 0
  CAM_SIZEX& = 0
  CAM_SIZEY& = 0
  RET& = 0

  Do
    If UnitItemIdent$(I&) = "ConveyorCalibration" Then

      COUNTER& = COUNTER& + 1
      Try
        GetImageSize I&, 0, CAM_SIZEX&, CAM_SIZEY&
      Catch
        RET& = -1
      End Try
```

```
        If (CAM_SIZEX& <= 0) OR (CAM_SIZEY& <= 0) OR (RET& = -1) Then

          For I& = 0 To 4
          ERROR& = ERROR& + ERROR_FLAG&(I&) * 2 ^ I&
          Next

          RESPONSECODE& = -1
          RESPONSEVALUE&(0) = ERROR&
          Return
        Else
          Exit Do
        EndIf

    Elseif UnitItemIdent$(I&) = "" Then
      Exit Do
    EndIf

    I& = I& + 1
  Loop While (1)

  If COUNTER& = 0 Then
    For I& = 0 To 4
      ERROR& = ERROR& + ERROR_FLAG&(I&) * 2 ^ I&
    Next

    RESPONSECODE& = -1
    RESPONSEVALUE&(0) = ERROR&
    Return
  EndIf

  ERROR_FLAG&(4) = 0

  If (ERROR_FLAG&(0) = 1) OR (ERROR_FLAG&(1) = 1) OR (ERROR_FLAG&(2) = 1) OR
(ERROR_FLAG&(3) = 1) Then
    For I& = 0 To 4
      ERROR& = ERROR& + ERROR_FLAG&(I&) * 2 ^ I&
    Next

    RESPONSECODE& = -1
    RESPONSEVALUE&(0) = ERROR&
    Return
  EndIf

  I& = 0

  Do
    If UnitItemIdent$(I&) = "ConveyorCalibration" Then

      SetUnitData I&, "samplingPointNum", 4
      SetUnitData I&, "calcMethod", 1

      SetUnitData I&, "cameraX0", 0
      SetUnitData I&, "cameraX1", CAM_SIZEX&
      SetUnitData I&, "cameraX2", 0
      SetUnitData I&, "cameraX3", CAM_SIZEX&
      SetUnitData I&, "cameraY0", 0
      SetUnitData I&, "cameraY1", 0
      SetUnitData I&, "cameraY2", CAM_SIZEY&
      SetUnitData I&, "cameraY3", CAM_SIZEY&
```

```
        SetUnitData I&, "robotX0", RBT_X0#
        SetUnitData I&, "robotX1", RBT_X1#
        SetUnitData I&, "robotX2", RBT_X2#
        SetUnitData I&, "robotX3", RBT_X3#
        SetUnitData I&, "robotY0", RBT_Y0#
        SetUnitData I&, "robotY1", RBT_Y1#
        SetUnitData I&, "robotY2", RBT_Y2#
        SetUnitData I&, "robotY3", RBT_Y3#

        SetUnitData I&, "cameraEncCount", 0
        SetUnitData I&, "robotEncCount1", 0
        SetUnitData I&, "robotEncCount2", 0

        SetUnitData I&, "encDistX", CONVEYOR_MOVE_X#
        SetUnitData I&, "encDistY", CONVEYOR_MOVE_Y#

        GetUnitData I&, "calcConvCalibPara", RESULT&

      Elseif UnitItemIdent$(I&) = "" Then
        Exit Do
      EndIf

      I& = I& + 1

    Loop While (1)

    RESPONSECODE& = 0
    RESPONSEVALUE&(0) = 0

EndIf
```

3-4-2 Program Example for CPU Unit

This section describes the program example to send the calibration parameters related to the vision sensor.
In this case, the scene data for vision sensor to perform calibration was already created and the scene was switched.

Global variables

Name	Data type	AT specification *1	Comment
E001_Line0_Command_Code	DWORD	ECAT://node#1/Line0 Command Code	Command code
E001_Line0_Command_Parameter_0	DINT	ECAT://node#1/Line0 Command Parameter 0	Command parameter 0
E001_Line0_Command_Parameter_1	DINT	ECAT://node#1/Line0 Command Parameter 1	Command parameter 1
E001_Line0_Command_Parameter_2	DINT	ECAT://node#1/Line0 Command Parameter 2	Command parameter 2
E001_Line0_Command_Parameter_3	DINT	ECAT://node#1/Line0 Command Parameter 3	Command parameter 3
E001_Line0_Command_Ready	BOOL	ECAT://node#1/Line0 Command Ready	Ready for command
E001_Line0_Command_Completion	BOOL	ECAT://node#1/Line0 Command Completion	Command completed
E001_Line0_Command_Request	BOOL	ECAT://node#1/Line0 Command Request	Executes a command
E001_Line0_Command_Code_Echo_Back	DWORD	ECAT://node#1/Line0 Command Code Echo Back	Command code echo back
E001_Line0_Trigger_Ready	BOOL	ECAT://node#1/Line0 Trigger Ready	Trigger input ready state
E001_Line0_Trigger	BOOL	ECAT://node#1/Line0 Trigger	Triggers measurement
E001_Line0_Busy	BOOL	ECAT://node#1/Line0 Busy	Processing in progress
E001_Line0_Trigger_Ack	BOOL	ECAT://node#1/Line0 Trigger Ack	Trigger acknowledged state
E001_Line0_User_Input_Area_4	LREAL	ECAT://node#1/Line0 User Input Area 4	User output area 4
E001_Line0_User_Input_Area_5	LREAL	ECAT://node#1/Line0 User Input Area 5	User output area 5
SimData	$SimData		Calibration parameter

*1 AT specification when the vision sensor is connected to the node address 1.

Internal variables

Name	Data type	Initial value	Comment
Trigger	BOOL	FALSE	Execution conditions
DoSetPara	BOOL	FALSE	Executing
Stage	INT	0	State transition

```
// Start sequence when Trigger changes to TRUE
IF Trigger AND NOT(DoSetPara) THEN
    DoSetPara:=TRUE;
    Stage:=1;
END_IF;

CASE Stage OF
    1:    // Change image mode
        E001_Line0_Command_Code:=DWORD#00000128;
        E001_Line0_Command_Parameter_0:=0;
        E001_Line0_Command_Parameter_1:=0;
        E001_Line0_Command_Parameter_2:=0;
        E001_Line0_Command_Parameter_3:=0;
        IF E001_Line0_Command_Ready AND NOT(E001_Line0_Command_Completion) THEN
            E001_Line0_Command_Request:=TRUE;
        END_IF;

        IF E001_Line0_Command_Completion THEN
            E001_Line0_Command_Request:=FALSE;
            IF E001_Line0_Command_Code_Echo_Back=E001_Line0_Command_Code THEN
                Stage:=2;
            END_IF;
        END_IF;

    2:    // Execute measurement
        IF E001_Line0_Trigger_Ready THEN
            E001_Line0_Trigger:=TRUE;
        END_IF;
        IF E001_Line0_Trigger_Ack THEN
            E001_Line0_Trigger:=FALSE;
            Stage:=3;
        END_IF;

    3:    // Start calibration
        E001_Line0_Command_Parameter_0:=1;
        IF E001_Line0_Command_Ready AND NOT(E001_Line0_Command_Completion) THEN
            E001_Line0_Command_Request:=TRUE;
        END_IF;

        IF E001_Line0_Command_Completion THEN
            E001_Line0_Command_Request:=FALSE;
            IF E001_Line0_Command_Code_Echo_Back=E001_Line0_Command_Code THEN
                Stage:=4;
            END_IF;
        END_IF;

    4:    // Set coordinate of origin of vision sensor relative to MCS1
        E001_Line0_Command_Parameter_0:=2;
        E001_Line0_User_Input_Area_4:=SimData.Vision[0].Origin[0];
        E001_Line0_User_Input_Area_5:=SimData.Vision[0].Origin[1];
        IF E001_Line0_Command_Ready AND NOT(E001_Line0_Command_Completion) THEN
            E001_Line0_Command_Request:=TRUE;
        END_IF;

        IF E001_Line0_Command_Completion THEN
            E001_Line0_Command_Request:=FALSE;
            IF E001_Line0_Command_Code_Echo_Back=E001_Line0_Command_Code THEN
                Stage:=5;
            END_IF;
        END_IF;

    5:    // Set unit travel distance of conveyor relative to MCS1
        E001_Line0_Command_Parameter_0:=3;
        E001_Line0_User_Input_Area_4:=SimData.Vision[0].Dx;
        E001_Line0_User_Input_Area_5:=SimData.Vision[0].Dy;
        IF E001_Line0_Command_Ready AND NOT(E001_Line0_Command_Completion) THEN
            E001_Line0_Command_Request:=TRUE;
        END_IF;

        IF E001_Line0_Command_Completion THEN
            E001_Line0_Command_Request:=FALSE;
            IF E001_Line0_Command_Code_Echo_Back=E001_Line0_Command_Code THEN
```

```
            Stage:=6;
        END_IF;
    END_IF;

6:      // Set size (length and width) of vision sensor view
    E001_Line0_Command_Parameter_0:=4;
    E001_Line0_User_Input_Area_4:=SimData.Vision[0].Length;
    E001_Line0_User_Input_Area_5:=SimData.Vision[0].Width;
    IF E001_Line0_Command_Ready AND NOT(E001_Line0_Command_Completion) THEN
        E001_Line0_Command_Request:=TRUE;
    END_IF;

    IF E001_Line0_Command_Completion THEN
        E001_Line0_Command_Request:=FALSE;
        IF E001_Line0_Command_Code_Echo_Back=E001_Line0_Command_Code THEN
            Stage:=7;
        END_IF;
    END_IF;

7:      // Set orientation of vision sensor view
    E001_Line0_Command_Parameter_0:=5;
    E001_Line0_User_Input_Area_4:=SimData.Vision[0].TiltAngle/90;
    IF E001_Line0_User_Input_Area_4=-1 THEN
        E001_Line0_User_Input_Area_4:=3;
    END_IF;
    IF E001_Line0_Command_Ready AND NOT(E001_Line0_Command_Completion) THEN
        E001_Line0_Command_Request:=TRUE;
    END_IF;

    IF E001_Line0_Command_Completion THEN
        E001_Line0_Command_Request:=FALSE;
        IF E001_Line0_Command_Code_Echo_Back=E001_Line0_Command_Code THEN
            Stage:=8;
        END_IF;
    END_IF;

8:      // Execute calibration
    E001_Line0_Command_Parameter_0:=6;
    IF E001_Line0_Command_Ready AND NOT(E001_Line0_Command_Completion) THEN
        E001_Line0_Command_Request:=TRUE;
    END_IF;

    IF E001_Line0_Command_Completion THEN
        E001_Line0_Command_Request:=FALSE;
        IF E001_Line0_Command_Code_Echo_Back=E001_Line0_Command_Code THEN
            Stage:=0;
        END_IF;
    END_IF;

0:
    IF NOT(Trigger) THEN
        DoSetPara:=FALSE;
    END_IF;

END_CASE;
```

4

Coordinate Systems

This section describes the coordinate systems in the pick-and-place application used for the Vision & Robot integrated simulation.

4-1 Overview of Coordinate Systems

This section describes the general coordinate systems used in the pick-and-place application.

Coordinate Systems Used by NJ Robotics CPU Unit

Coordinate system	Description
Axis coordinate system (ACS)	The axis coordinate system (ACS) is a rotation or linear motion coordinate system specified for each axis. This system is also called the Joint Coordinate System or the Link Coordinate System. Each axis has one coordinate system.
Machine coordinate system (MCS)	The machine coordinate system (MCS) is a Cartesian coordinate system specified for each robot. Use the position of the robot TCP viewed from the origin of the machine coordinate system to set the target position for robot operation. Each robot (one axes group) has one machine coordinate system.
User coordinate system (UCS)	The user coordinate system is a Cartesian coordinate system defined by the user. Use the rotation and the position relative to the origin of the machine coordinate system to set the origin position (Tx, Ty, Tz, Rx, Ry, Rz) of the user coordinate systems. The rotation is also called Pose. You can set up to 16 user coordinate systems for each robot.
Tool coordinate system (TCS)	The tool coordinate system is a Cartesian coordinate system set by the user according to the tool to be used. The reference position of the tool coordinate system is the position of the tool TCP. You can set up to 16 tool coordinate systems for each robot, and select from the total of 17 systems that include the default tool coordinate system.

Refer to the N*J-series NJ Robotics CPU Unit USER'S MANUAL* (Cat. No. W539) for details on each coordinate system.

Coordinate Systems Used by Vision Sensor

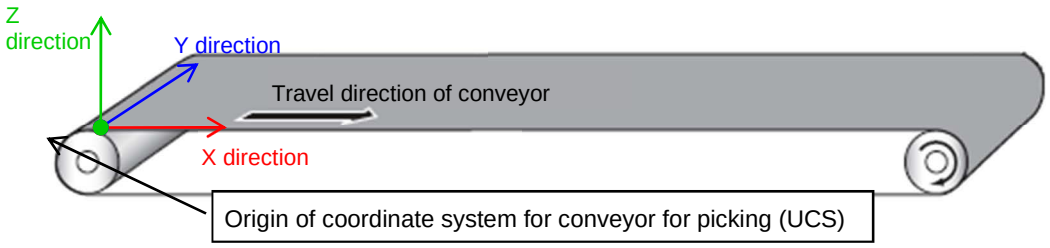
Coordinate system	Description
Camera coordinate system	The origin is at the upper-left corner of the captured image. The horizontal axis is the X coordinate, and the vertical axis is the Y coordinate. The unit is pixel.

4-2 Coordinate Systems for Integrated Simulation

This section describes the coordinate systems for the integrated simulation that are used to calculate calibration parameters and to simulate the pick-and-place application.

4-2-1 Coordinate System for Conveyor for Picking

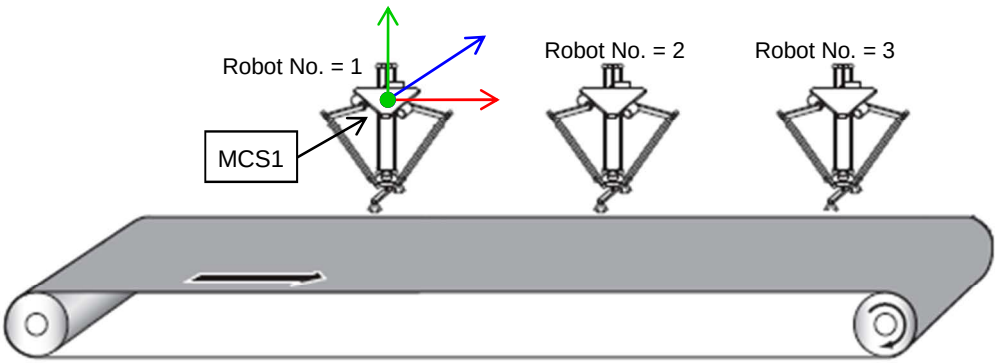
The user coordinate system (UCS) is used for the conveyor for picking.
The origin of the coordinate system for the conveyor for picking (UCS) is at the upper end of the conveyor for picking.



In the integrated simulation, workpieces on the conveyor for picking move along the X axis of the coordinate system for the conveyor for picking (UCS) according to the travel distance of the conveyor (travel distance of the axis assigned to the conveyor for picking). They do not move along the Y and X axis.

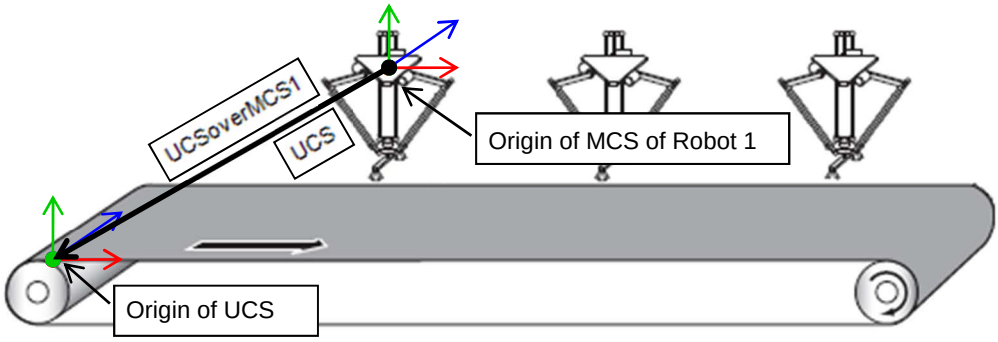
4-2-2 Relationship between Conveyor for Picking and Robots

Robots on the conveyor for picking are numbered from 1 in ascending order from the upper end of the conveyor for picking.
The machine coordinate system (MCS) for the robot located at the uppermost position of the conveyor is called "MCS1".
The robot with the robot number=X will hereinafter be called "Robot x".

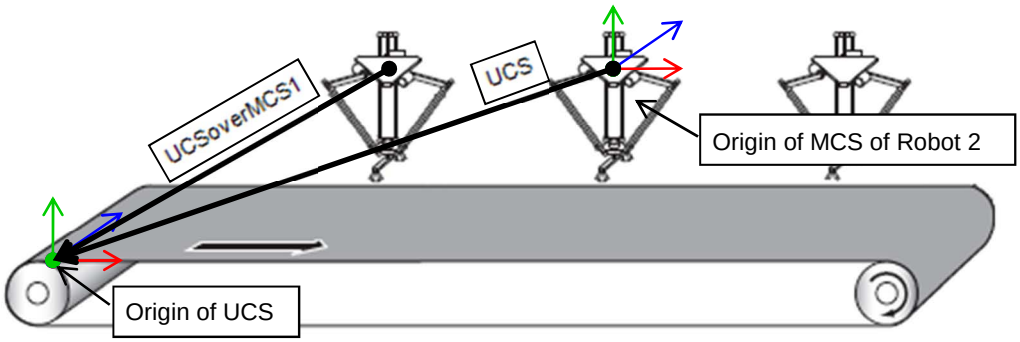


When multiple robots are used for the conveyor for picking, the positional relationship between the conveyor for picking and robots can be calculated from the UCS of Robot 1 (UCS relative to the MCS1) and the UCS of each robot.
The following figures show the positional relationship between the UCS relative to the MCS1 (UCSoverMCS1) and the UCS of each robot.

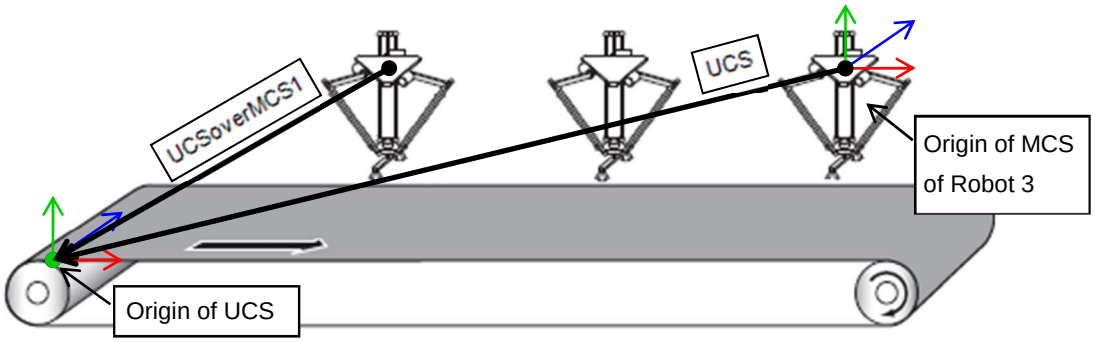
Relationship between UCSoverMCS1 and UCS of Robot 1



Relationship between UCSoverMCS1 and UCS of Robot 2



Relationship between UCSoverMCS1 and UCS of Robot 3



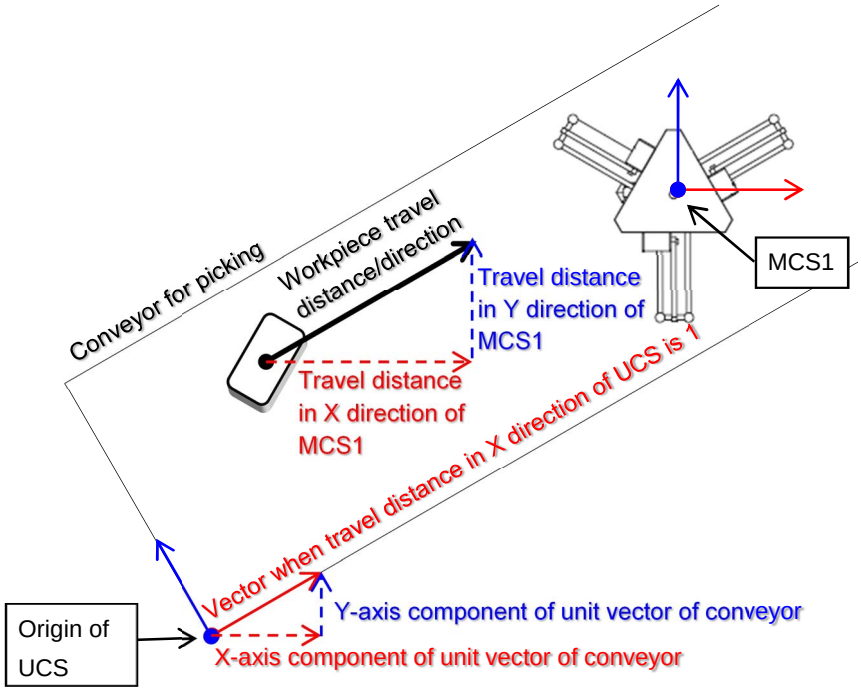
When all the rotation parameters (Rx, Ry, Rz) of the UCS of Robot 1 are set to 0, workpieces on the conveyor for picking move only along the X axis of the MCS1. The travel distance in the X direction of the MCS1 is the same as the travel distance of the conveyor for picking.

When one or more rotation parameters (Rx, Ry, Rz) of the UCS of Robot 1 are set to other than 0, workpieces on the conveyor for picking move not only along the X axis but also other axes of the MCS1.

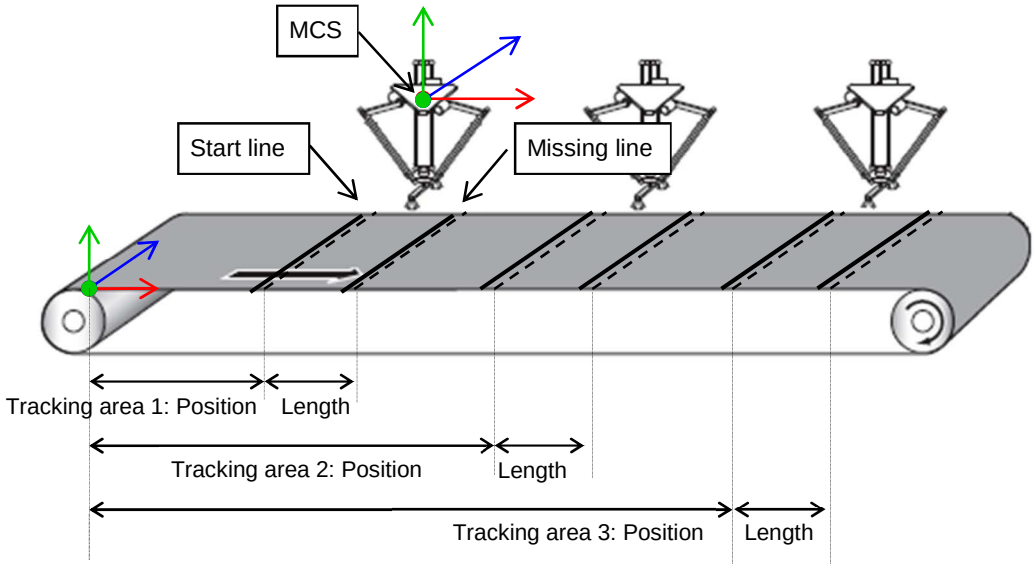
In both cases, the travel distance of the workpiece on the conveyor for picking relative to the machine coordinate system for Robot 1 (MCS1) can be obtained from the following equation.

Travel distance in X direction relative to MCS1
= Conveyor travel distance * X-axis component of unit vector of conveyor
Travel distance in Y direction relative to MCS1
= Conveyor travel distance * Y-axis component of unit vector of conveyor
Travel distance in Z direction relative to MCS1
= Conveyor travel distance * Z-axis component of unit vector of conveyor

Workpiece travel distance on conveyor for picking relative to MCS1 when Rz of UCS is 30°



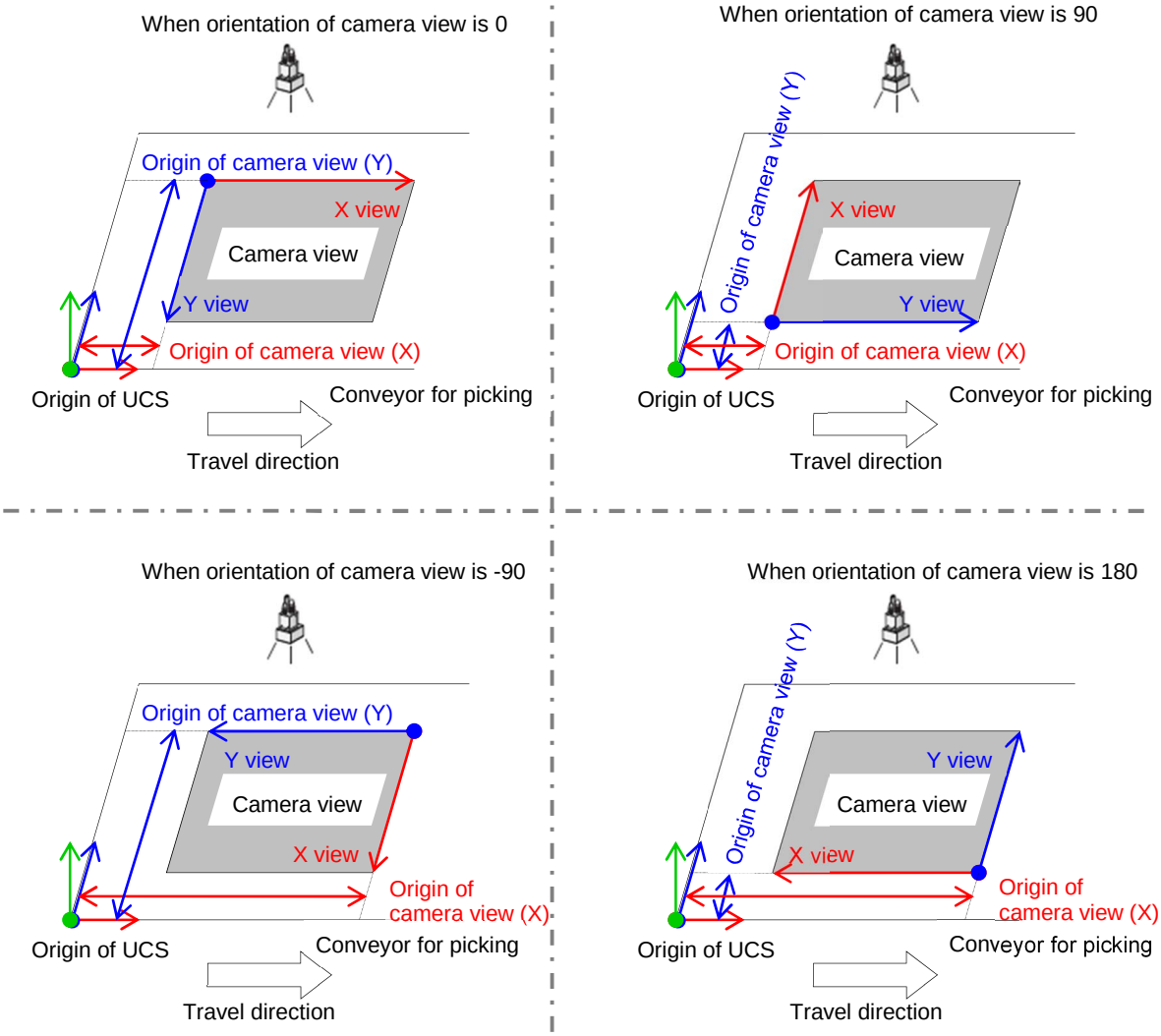
The tracking area set on the conveyor for picking is transformed into the start and missing lines in the machine coordinate system (MCS) of the robot to output as the calibration parameters.



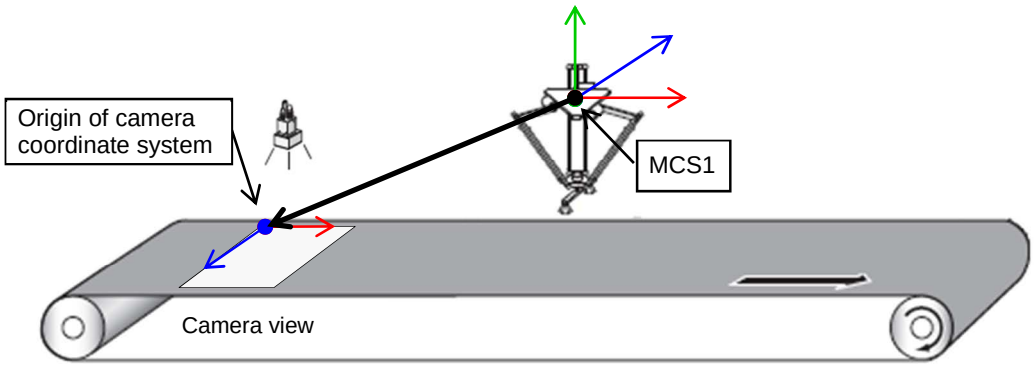
4-2-3 Relationship between Conveyor for Picking and Camera View of Vision Sensor

The coordinate system for the camera view of the vision sensor (camera coordinate system) is on the X-Y plane of the coordinate system for the conveyor for picking (UCS).
The orientation of the camera view represents the orientation (rotation angle) of the camera view in the X direction relative to the X direction of the coordinate system for the conveyor for picking (UCS).

The following figure shows the relationship between the coordinate system for the conveyor for picking and camera coordinate system.



The origin of the camera view of the vision sensor (origin of the camera coordinate system) is transformed into the machine coordinate system of Robot 1 (MCS1) to output as the calibration parameters.

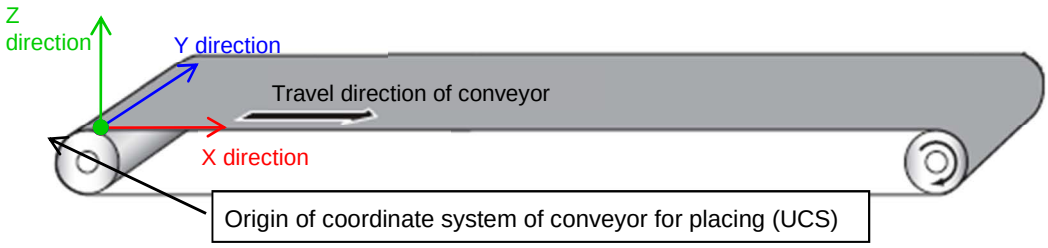


Send these calibration parameters to the vision sensor by using the communication command macro.
Create the scene to output the workpiece position after transforming it into the machine coordinate system of Robot 1 (MCS1) using these parameters.

In the integrated simulation, the workpiece positions output by the vision sensor are displayed in the machine coordinate system of Robot 1 (MCS1) and used for simulation. If the workpiece positions are output in the coordinate system other than the machine coordinate system of Robot 1 (MCS1), the integrated simulation will not work properly.

4-2-4 Coordinate System for Conveyor for Placing

The user coordinate system (UCS) is used for the conveyor for placing as well as for the conveyor for picking.
The origin of the coordinate system of the conveyor for placing (UCS) is at the upper end of the conveyor for placing.

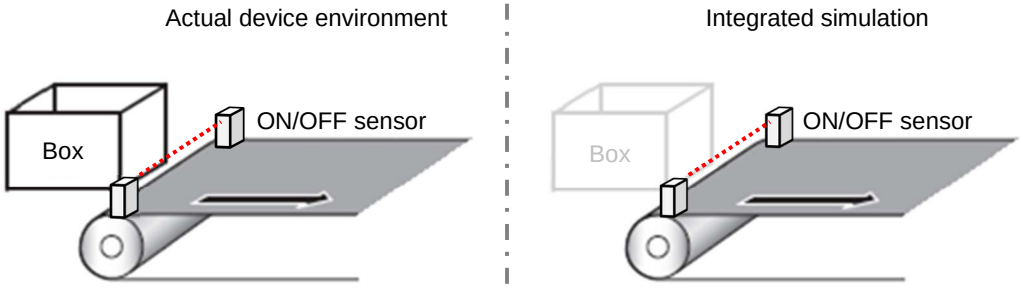


In the integrated simulation, borders or boxes on the conveyor for placing move along the X axis of the coordinate system of the conveyor for placing (UCS) according to the travel distance of the conveyor (travel distance of the axis assigned to the conveyor for placing). They do not move along the Y and X axis.

4-2-5 ON/OFF Sensor on Conveyor for Placing

The ON/OFF sensor is located at the origin of the coordinate system for the conveyor for placing (UCS).
This sensor is assumed to detect borders or boxes on the conveyor for placing in the actual device environment.

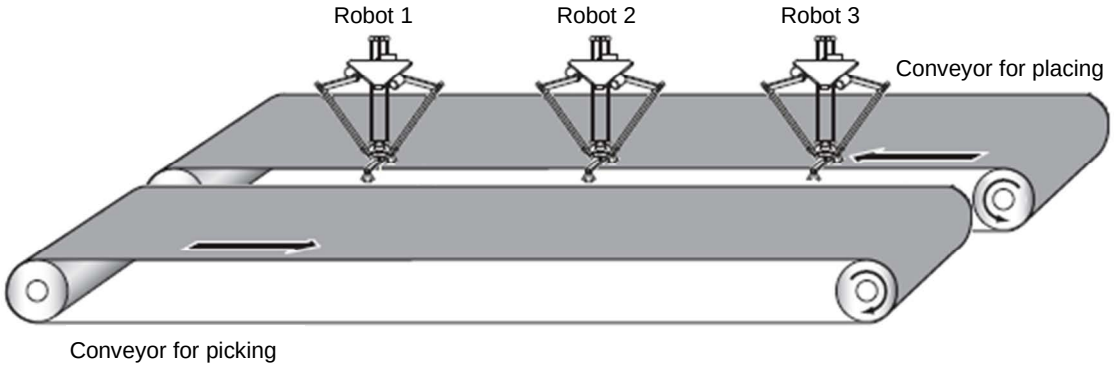
In the actual device environment, the sensor is turned ON and OFF by detecting the border or box on the conveyor. In the integrated simulation, the user program is required to control ON and OFF.



In the integrated simulation, the border or box is displayed on the conveyor for placing to simulate when the ON/OFF sensor on the conveyor is turned ON.

4-2-6 Relationship between Conveyor for Placing and Robots

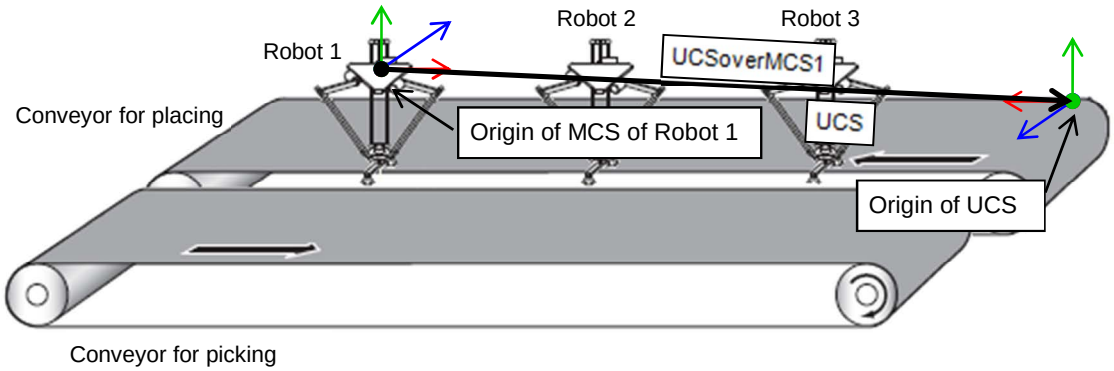
As robots on the conveyor for placing are numbered based on the conveyor for picking, the number of the robot located at the uppermost position of the conveyor for placing may not be 1.



When the conveyors for picking and placing are installed as shown above, the robots on the conveyor for placing are numbered in descending order from the upper end of the conveyor for placing. This means that the number of the robot located at the lowermost position is 1.

In the case shown above, the following figure shows the positional relationship between the coordinate system of the conveyor for placing (UCS) and the machine coordinate system of Robot 1 (MCS1)

UCSoverMCS1 and UCS of Robot 1



MEMO

OMRON Corporation **Industrial Automation Company**
Tokyo, JAPAN
Contact: www.ia.omron.com

Regional Headquarters
OMRON EUROPE B.V.
Wegalaan 67-69, 2132 JD Hoofddorp
The Netherlands
Tel: (31)2356-81-300/Fax: (31)2356-81-388

OMRON ELECTRONICS LLC
2895 Greenspoint Parkway, Suite 200
Hoffman Estates, IL 60169 U.S.A
Tel: (1) 847-843-7900/Fax: (1) 847-843-7787

OMRON ASIA PACIFIC PTE. LTD.
No. 438A Alexandra Road # 05-05/08 (Lobby 2),
Alexandra Technopark,
Singapore 119967
Tel: (65) 6835-3011/Fax: (65) 6835-2711

OMRON (CHINA) CO., LTD.
Room 2211, Bank of China Tower,
200 Yin Cheng Zhong Road,
PuDong New Area, Shanghai, 200120, China
Tel: (86) 21-5037-2222/Fax: (86) 21-5037-2200

Authorized Distributor:

Cat. No. Y213-E1-01 1215 (1215)