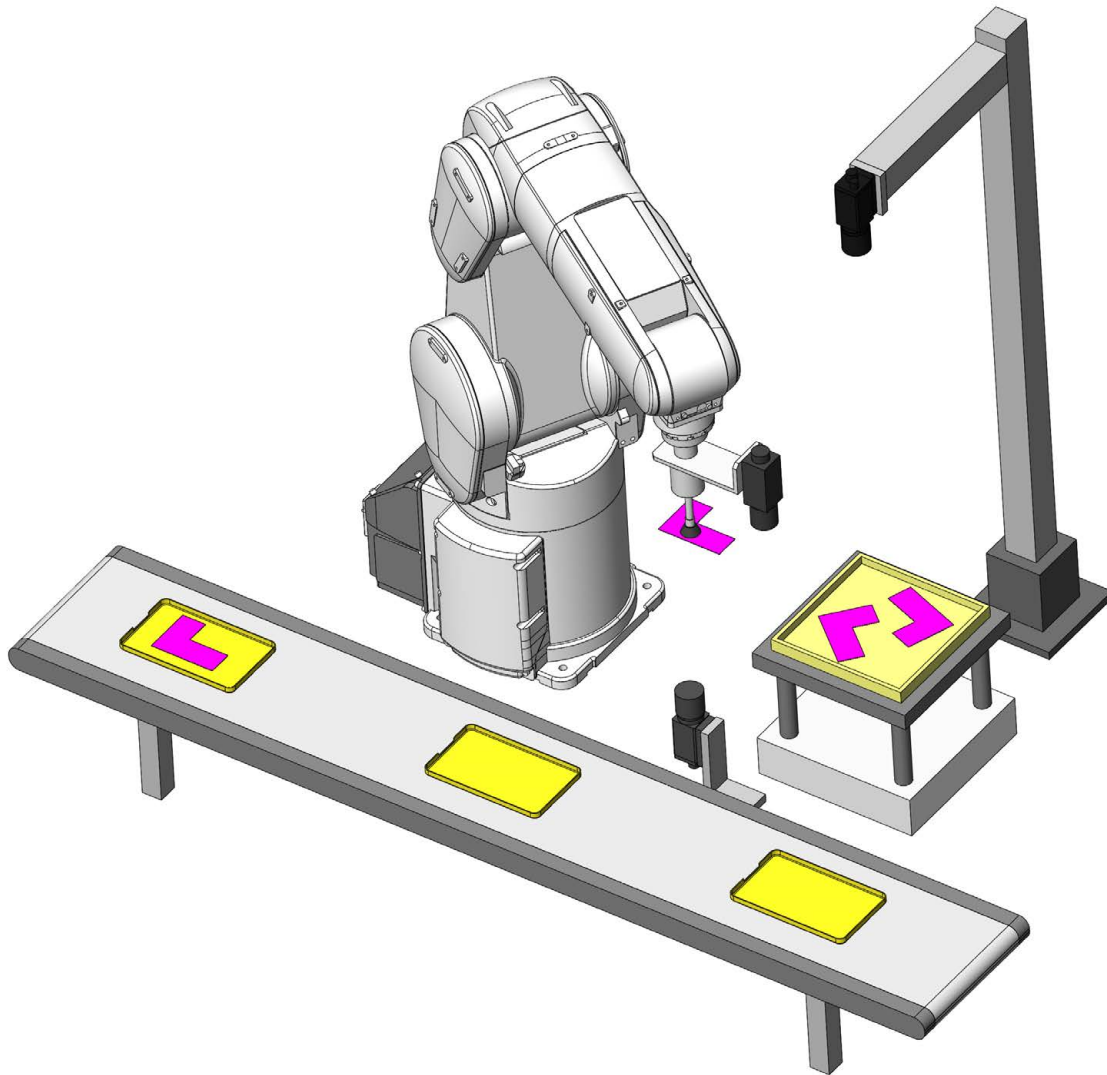


MELFA
Robot Seminar Textbook
2D Vision Sensors for Robots



MELFA
BFP-A3659-A

MELFA robot seminar schedule

Day 1		Day 2	
Agenda	Time	Agenda	Time
Seminar introduction Orientation ■ Fundamentals of 2D vision sensors System architecture ■ Fixed downward-facing camera Communication settings Lens adjustments	1 hr 20 min	■ Fixed downward-facing camera Automatic operation ■ Hand eye Tool settings	1 hr 20 min
Break	10 min	Break	10 min
■ Fixed downward-facing camera Tool settings Calibration	1 hr	■ Hand eye Calibration	1 hr
Lunch		Lunch	
■ Vision sensor commands Status variables ■ Fixed downward-facing camera Create identification jobs Teaching	1 hr 50 min	■ Hand eye Create identification jobs Teaching	1 hr 40 min
Break	10 min	Break	10 min
■ Fixed downward-facing camera Manual operation	1 hr 55 min	■ Hand eye Manual operation Automatic operation	1 hr 35 min
End		End	

SAFETY PRECAUTIONS

Always read the following precautions and the separate Safety Manual carefully before using this product, and take appropriate action when required.

CAUTION

Teaching of the robot should only be performed by individuals who have undergone special safety training.

(The same applies to maintenance work with the robot power ON.)

→ Conduct safety education.

CAUTION

Prepare work regulations indicating robot operation methods and procedures, and measures to be taken when errors occur or when rebooting robots. Observe these rules at all times.

(The same applies to maintenance work with the robot power ON.)

→ Prepare work regulations.

WARNING

Only teach the robot after first equipping the controller with a device capable of stopping operation immediately.

(The same applies to maintenance work with the robot power ON.)

→ Equip the controller with an Emergency Stop button.

CAUTION

Notify others that the robot is being taught by affixing a sign to the Start switch.

(The same applies to maintenance work with the robot power ON.)

→ Indicate that the robot is being taught.

DANGER

Install fences or enclosures around robots to prevent contact between robots and workers during operation.

→ Install safety fences.

CAUTION

Stipulate a specific signaling method to be used among related workers when starting operation.

→ Operation start signal

CAUTION

Shut off the power when maintaining the robot. Notify others that the robot is under maintenance by affixing a sign to the Start switch.

→ Indicate that maintenance work is being performed.

CAUTION

Before starting operation, conduct an inspection of robots, Emergency Stop buttons, and any other related devices to ensure that there are no abnormalities.

→ Inspection before starting operation

The following precautions are taken from the separate Safety Manual.
Refer to the Safety Manual for further details.

DANGER

Design interlocks such as individual device operation rights when operating the robot automatically with multiple control devices (GOTs, programmable controllers and push-button switches).

CAUTION

Only use robots within environments stipulated in the specifications.
Failure to observe this may result in decreased reliability or breakdown.
(Temperature, humidity, atmosphere, noise environment, etc.)

CAUTION

Only transport robots in the manner stipulated.
Failure to observe this may result in bodily injury or breakdown if the robot is dropped.

CAUTION

Caution Install and use the robot on a secure and stable platform.
Positional displacement or vibrations may occur if the robot is unstable.

CAUTION

Ensure that cables are kept as far apart from noise sources as much as possible.
Positional displacement or malfunction may occur if cables are close to noise sources.

CAUTION

Do not apply excessive force to connectors, or bend cables too much.
Failure to observe this may result in contact failures or wire damage.

CAUTION

Ensure that the weight of the workpiece, including the hand, does not exceed the rated load or permissible torque.
Failure to observe this may result in alarms or breakdown.

WARNING

Ensure that hands and tools are attached properly, and that workpieces are gripped securely.
Failure to observe this may result in bodily injury or property damage if objects are sent flying or released during operation.

WARNING

Ground the robot and controller properly.
Failure to observe this may result in malfunction due to noise, or even electric shock.

CAUTION

Ensure that the robot's operation status is displayed during operation.
Failure to display the robot's operating status may result in operators approaching the robot, potentially leading to incorrect operation.

WARNING

If performing teaching work inside the robot operation range, always ensure complete control over the robot beforehand. Failure to observe this may result in bodily injury or property damage if the robot is able to start with external commands.

CAUTION

Jog the robot with the speed set as low as possible, and never take your eyes off the robot. Failure to observe this may result in collision with workpieces or surrounding equipment.

CAUTION

Always check robot movement in step operation before commencing auto operation following program editing. Failure to observe this may result in collision with surrounding equipment due to programming mistakes, etc.

CAUTION

Ensure that the door of the safety fence locks or that the robot automatically stops if someone attempts to open it during auto operation. Failure to observe this may result in bodily injury.

CAUTION

Do not perform unauthorized modifications or use maintenance parts other than those stipulated. Failure to observe this may result in breakdown or malfunction.

WARNING

When moving the robot arm by hand, never insert hands or fingers into openings. Depending on the posture of the robot, hands or fingers may become jammed.

CAUTION

Do not stop the robot or perform an emergency stop by turning OFF the main power of the robot controller. Robot accuracy may be adversely affected if the main power of the robot controller is turned OFF during auto operation. Furthermore, the robot arm may collide with surrounding equipment if it falls or moves under its own inertia.

CAUTION

When rewriting internal robot controller information such as programs or parameters, do not turn OFF the main power of the robot controller. If the main power of the robot controller is turned OFF while rewriting programs or parameters during auto operation, internal robot controller information may be corrupted.

DANGER

Do not connect a Handy GOT when using this product's GOT direct connection function. The Handy GOT can operate the robot automatically regardless of whether the operation rights are enabled or disabled. This could lead to property damage or bodily injury.

DANGER

Do not connect a Handy GOT to a programmable controller when using iQ Platform compatible products with CR800-R or CR800-Q controllers. The Handy GOT can operate the robot automatically regardless of whether the operation rights are enabled or disabled. This could lead to property damage or bodily injury.

DANGER

Do not disconnect SSCNET III cables when either the multi CPU system or the servo amplifier is ON. Do not look directly at light emitted from the end of SSCNET III connectors or SSCNET III cables. Doing so may cause discomfort. (SSCNET III employs a Class 1 or equivalent light source as specified in JISC6802 and IEC60825-1.)

DANGER

Do not disconnect SSCNET III cables while the controller is ON. Do not look directly at light emitted from the end of SSCNET III connectors or SSCNET III cables. Doing so may cause discomfort.

(SSCNET III employs a Class 1 or equivalent light source as specified in JISC6802 and IEC60825-1.)

DANGER

Replace the caps of SSCNET III connectors after they have been disconnected to prevent them from being adversely affected by dust and foreign matter.

CAUTION

Take care not to wire devices incorrectly. Incorrect wiring may cause malfunctions such as the inability to terminate an emergency stop.

After wiring devices such as the teaching pendant Emergency Stop switch, door switch and any emergency stop devices prepared by the customer, ensure that they are functioning correctly to prevent accidents from occurring.

CAUTION

Not all commercial devices such as computers and network hubs will function correctly when connected to the controller's USB port. They may also be affected by temperatures and electronic noise found in FA environments.

When using commercial devices, protection against EMI (Electric-magnetic interference) and the use of ferrite cores etc. may be required to ensure devices function correctly. Mitsubishi Electric does not guarantee commercial devices will be compatible with our products. Neither will we carry out maintenance on them.

CAUTION

If needed, create a way to keep the robot system safe from unauthorized access from external devices connected to the network.

In addition, implement a firewall to keep the robot system safe from unauthorized access from external devices connected to the Internet.

■ Revision history

Date	Document No.	Revision
July 10, 2018	BFP-A3659	First edition
May 8, 2019	BFP-A3659-A	Added new information and revised existing pages. (Revised for In-Sight Explorer version 5.4.3 and RT ToolBox3 version 1.32J)

■ Important notices and disclaimers

- Distribution of this document, in-part or whole, without prior authorization is strictly prohibited.
- Information contained in this document is subject to change without notice.
- Specifications are based on the results of in-house standardized tests.
- This is an original Mitsubishi document.
- All trademarks/registered trademarks of company and product names mentioned in this document are the property of their respective owners.
- ® and ™ marks have been omitted in this document.

Contents

■ Introduction.....	1
■ Terminology.....	2
Chapter 1 - Fundamentals of 2D vision sensors	3
1.1 Specifications of 2D vision sensors	3
1.2 Specifications of supported robot controllers.....	3
1.3 Introduction to In-Sight Explorer	4
1.3.1 Installing In-Sight Explorer	4
1.3.2 In-Sight Explorer window layout.....	5
1.3.3 Application Steps.....	6
1.3.4 Frequently used operations.....	6
1.4 System architecture	7
1.4.1 D Type controllers	7
1.4.2 R Type/Q Type controllers	8
1.5 Types of vision sensors and their advantages and disadvantages	9
1.6 Steps required for automatic operation.....	12
1.6.1 Overview.....	12
1.6.2 Communication settings overview.....	13
1.6.3 Overview of programs used for each camera	13
Chapter 2 - Communication settings and lens adjustment.....	15
■ Steps required to configure the communication settings	15
2.1 Preparing the robot and vision sensor	16
2.2 Configuring computer network settings	16
2.3 Connecting the robot and camera (RT ToolBox3).....	18
2.3.1 Robot network settings.....	18
2.3.2 comRobot and camera connection settings.....	21
(1) Device parameter settings.....	21
(2) Parameter settings	23
2.4 Configuring camera communication settings (In-Sight Explorer)	24
2.4.1 Starting In-Sight Explorer	24
2.4.2 Connecting cameras	25
2.4.2.1 At least one camera detected	25
2.4.2.2 If no cameras are detected	26
2.4.2.3 Configuring system options.....	29
2.5 Adjusting the lens (focus and aperture).....	31
Chapter 3 - Fixed downward-facing camera	34
3.1 Principles behind the fixed downward-facing camera.....	34
3.1.1 Fundamentals of the fixed downward-facing camera	34
3.1.2 Setting a control point for calibration.....	36
3.1.3 Control point used for calibrating the fixed downward-facing camera	37
3.1.4 Calibration of the fixed downward-facing camera	38
(1) What is calibration?	38
(2) Typical calibration method.....	38
(3) N point calibration.....	39
(4) Fixed downward-facing camera: N point calibration method	39
3.1.5 Fundamentals of the fixed downward-facing camera	42
■ Process of adjusting the fixed downward-facing camera	44
3.2 Setting a control point used for calibration (Fixed downward-facing camera).....	45
3.2.1 Program for setting a control point used for calibration	45
◇ Program UBP.prg	45
◇ Program TLUP.prg	48
3.2.2 Setting the control point used for calibration	49
■ Process of setting the control point used for calibration	49
(1) Adding a user base program.....	51
(2) Executing the program "TLUP"	51
(3) Placing the calibration tool in the robot hand	52
(4) Positioning the pointed object on the platform	53
(5) Setting a control point used for calibration	53
(6) Checking the control point used for calibration	54

3.3 Calibration (Fixed downward-facing camera)	55
3.3.1 Calibration method	55
■ Calibration process	55
(1) Creating a new job	56
(2) Positioning calibration workpieces and connecting the camera	56
(3) Adjusting the lens	57
(4) Setting user-defined points	58
(5) Create N points	60
(6) Carry out N point calibration	62
3.4 Creating an identification job	65
■ Identification job creation process	65
(1) Creating a new job	65
(2) Configuring calibration file settings	66
(3) Setting the workpiece to be identified and the identification range	66
(4) Setting threshold and rotation tolerance values	68
(5) Configuring communication settings	69
(6) Selecting an identified pattern	70
(7) Saving the job	71
(8) Checking calibration	73
3.5 Setting up the relative position between the workpiece and the robot	75
3.5.1 Program for setting up the relative position between the workpiece and the robot	75
◇ Program UVS1.prg	75
◇ Program UVS2.prg	77
3.5.2 Setting up the relative position between the workpiece and the robot	80
■ Procedure	80
(1) Positioning the master workpiece used for adjustment	81
(2) Teaching the safe position (PHOME) and suction area on the master workpiece (PWK)	81
(3) Running the robot program "UVS1" automatically	82
(4) Teaching the destination position (PPUT)	83
3.6 Checking the robot movement using step operation and automatic operation	84
Chapter 4 - Setting up the hand eye	85
4.1 Overview of the hand eye	85
4.1.1 Setting a control point used for calibration	85
4.1.2 Control point for calibrating the hand eye	87
4.1.3 Calibrating the hand eye	88
4.1.4 Principal of how the robot identifies the workpiece using the hand eye	89
■ Process of setting up the hand eye	90
4.2 Setting the control point used for calibration using the hand eye	91
4.2.1 Program for setting the control point used for calibration	91
◇ Program UBP.prg	91
◇ Program HND1.prg	91
4.2.2 Setting the control point used for calibration	93
■ Process of setting the control point used for calibration	93
(1) Adding a user base program	95
(2) Connecting a camera	95
(3) Positioning the workpiece used for calibration	96
(4) Adjusting the lens	96
(5) Drawing two intersecting lines in In-Sight Explorer	97
(6) Setting the control point used for calibration	99
(7) Checking the control point used for calibration	100
4.3 Calibration using the hand eye	101
4.3.1 Calibration program	101
◇ Program HND2.prg	101
4.3.2 Calibration method	103
■ Calibration process	103
(1) Measuring the camera's FOV	105
(2) Setting user-defined points	107
(3) Entering the size of the camera's FOV	109
(4) Fine-tuning user-defined points	110
(5) Creating N points	111
(6) Carrying out N point calibration	113
(7) Setting an offset from the workpiece suction point to the imaging point	115

4.4 Creating an identification job.....	117
4.4.1 Program for creating an identification job.....	117
◇ Program HND3.prg.....	117
4.4.2 Method of creating an identification job.....	119
■ Process of creating the identification job.....	119
(1) Adjusting the suction and identification points of the master workpiece.....	119
(2) Creating a job.....	121
(3) Configuring calibration file settings.....	121
(4) Setting the workpiece to be identified and the identification range.....	122
(5) Setting threshold and rotation tolerance values.....	124
(6) Configuring communication settings.....	125
(7) Selecting an identified pattern.....	126
(8) Saving the job.....	127
4.5 Setting up the relative position between the workpiece and the robot.....	128
4.5.1 Program for setting up the relative position between the workpiece and the robot.....	128
◇ Program HND3.prg.....	128
4.5.2 Setting up the relative position between the workpiece and the robot.....	128
4.6 Checking the robot movement using step operation and automatic operation.....	129
4.6.1 Program for checking robot movement.....	129
◇ Program HND4.prg.....	129
4.6.2 Checking the movement of the robot.....	132
Appendix 1. Vectors and operations on vectors.....	135
Appendix 1.1 Robot position data and vectors.....	135
Appendix 1.2 Method of finding PH.....	136
Appendix 2. Vision sensor commands and status variables.....	137
Appendix 2.1 Vision sensor commands.....	137
NVOpen (Open network vision sensor port).....	138
NVClose (Disconnect network vision sensor).....	141
NVLoad (Load network vision sensor).....	142
NVRun (Run network vision sensor program).....	144
NVTrg (Network vision sensor trigger).....	146
EBRead (EasyBuilder read).....	148
EBWrite (EasyBuilder write).....	151
Appendix 2.2 Vision sensor status variables.....	153
M_NvOpen.....	154
Appendix 3. 2D vision sensor parameters.....	155
Appendix 4. Troubleshooting.....	157
Appendix 4.1 Error numbers.....	157
Appendix 4.2 2D vision sensor error code list.....	158

■ Introduction

In this seminar we will use Mitsubishi Electric VS80 series 2D vision sensors and study the fundamentals of systems that combine robots and vision sensors.



2D vision sensor (VS80 series)

Positioning is generally considered the most important thing when using robots.

First and foremost, we must understand that the pickup point and destination point are fixed. For robots, specialized jigs are needed for positioning and jigs can be expensive if complex workpieces are used. Frequent setup changes are a hassle and finding a place to store jigs is a problem. All of these things have adverse effects on productivity.

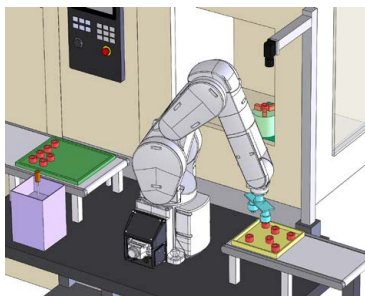
One solution to positioning workpieces is to position them using image recognition. Typically, the vision sensor decides whether the workpiece it has looked at matches the characteristics of a pre-registered jig. If it does, a predetermined output position is sent to the robot and the robot adjusts the orientation of the workpiece to match the jig.

Main applications

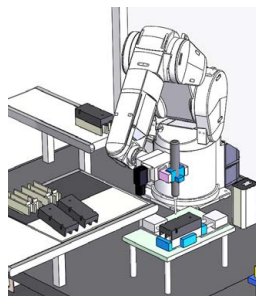
- Recognize and accurately pick randomly orientated workpieces from supply platforms and roughly aligned workpieces on plastic trays.
- Confirm the point of assembly and accurately place or assemble the workpiece at that point.
- Adjust the position of the hand if the workpiece is not being held properly.
- Read 2D codes and pick the product related to that code from the production line.
- Detect abnormalities.

Robots alone cannot inspect parts. For inquiries related to using vision sensors for flaw detection (scratches etc.), contact your local sales representative.

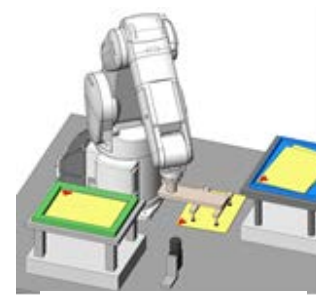
Examples



Bin picking



Assembly of parts



Code reading

■ Terminology

The following terms are used in this document.

Term	Definition
In-Sight Explorer (In-Sight Explorer for MELSENSOR)	Software used for setting up the vision sensor. (Created by Cognex Corp.) In-Sight Explorer is required to configure vision sensor settings and setup vision applications. To operate this software, a camera needs to be connected to the computer the software is installed on. * In this document, In-Sight Explorer for MELSENSOR is expressed as "In-Sight Explorer".
EasyBuilder®	EasyBuilder helps users program their vision system. (Created by Cognex Corp.)
PatMax®	A positioning tool which utilizes advanced geometric pattern verification technology and algorithms developed in-house by Cognex (U.S. patent granted). It is able to precisely identify the position of objects and is unaffected by shadows and changes in the angle or size of an object.
Job	Contains the program data of the 2D vision project. Jobs are needed to manage data and communications transmitted between the robot and 2D vision sensor(s). MELFA BASIC robot programs can control job access, imaging triggers and data acquisition of jobs created in In-Sight Explorer.
Calibration	Calibration is work that involves converting the coordinates of the vision sensor camera into robot coordinates.
Trigger	Captures images from the camera.
Hand eye	A camera attached to the end of the robot arm which is used for taking measurements and recognizing workpieces.
Fixed camera	A camera attached to a frame which is used for taking measurements and recognizing workpieces. Unlike the hand eye, fixed cameras cannot move.

* For further information on robot operations, robot programming, and vision sensors, refer to the PDF robot manual. The latest PDF robot manual can be downloaded from the Mitsubishi Electric FA website.
(Mitsubishi Electric FA website: www.MitsubishiElectric.co.jp/fa)

Related manuals

Term	Definition
Detailed explanations of functions and operations	Explanations of and how to use functions, MELFA BASIC commands, external I/O devices, and parameters.
Troubleshooting	Reasons for errors and how to deal with them when they arise.
RT ToolBox3/RT ToolBox3 mini Instruction manuals	Instruction manuals for comprehensive robot engineering assistance software (RT ToolBox3, RT ToolBox3 mini, RT ToolBox3 Pro).

Chapter 1 - Fundamentals of 2D vision sensors

1.1 Specifications of 2D vision sensors

There are two Mitsubishi Electric 2D vision sensors (MELSENSORS): the compact, wire-saving, stand-alone VS80 series, and the VS70 series which has in-built lights.

In this seminar we will use VS80 series vision sensors and learn the basics of systems that incorporate robots and vision sensors.



VS80 series



VS70 series

(1) MELSENSOR VS80 series

Model:	Resolution (pixels)		Processing power *1
	640×480	1600×1200	
VS80M-100-E	•		× 1
VS80M-200-E(ER)	•		× 1.5
VS80M-400-E(ER)	•		× 2
VS80M-202-E(ER)		•	× 1.5
VS80M-402-E(ER)		•	× 2

*1 Compared to the VS80M-100-E

(2) MELSENSOR VS70 series

Model:	Resolution (pixels)			Processing power *2
	640×480	800×600	1600×1200	
VS70M-600-E(ER)	•	•		× 1
VS70M-800-E(ER)	•	•		× 1.25
VS70M-802-E(ER)			•	× 1.25

*2 Compared to the VS70M-600-E

1.2 Specifications of supported robot controllers

Item	Specifications
Robot	FR series F series
Robot controller	CR800-R, Q, and D series CR750 series
No. of connectable robots and vision sensors	No. of cameras that can be connected to one robot controller: up to 7 No. of robots that can be connected to one vision system: up to 3
Software	RT ToolBox3: Ver.1.0 or above is recommended
Robot program language	MELFA BASIC. Contains dedicated vision sensor commands.

1.3 Introduction to In-Sight Explorer

In-Sight Explorer is software used for setting up vision sensors. (Cognex corp.)

In-Sight Explorer is required to configure VS series vision sensor settings and setup vision applications. In-Sight Explorer needed to be installed on a Windows PC and cameras need to be connected for it to operate.

1.3.1 Installing In-Sight Explorer

1. Download the In-Sight Explorer installer from the Mitsubishi Electric FA website.

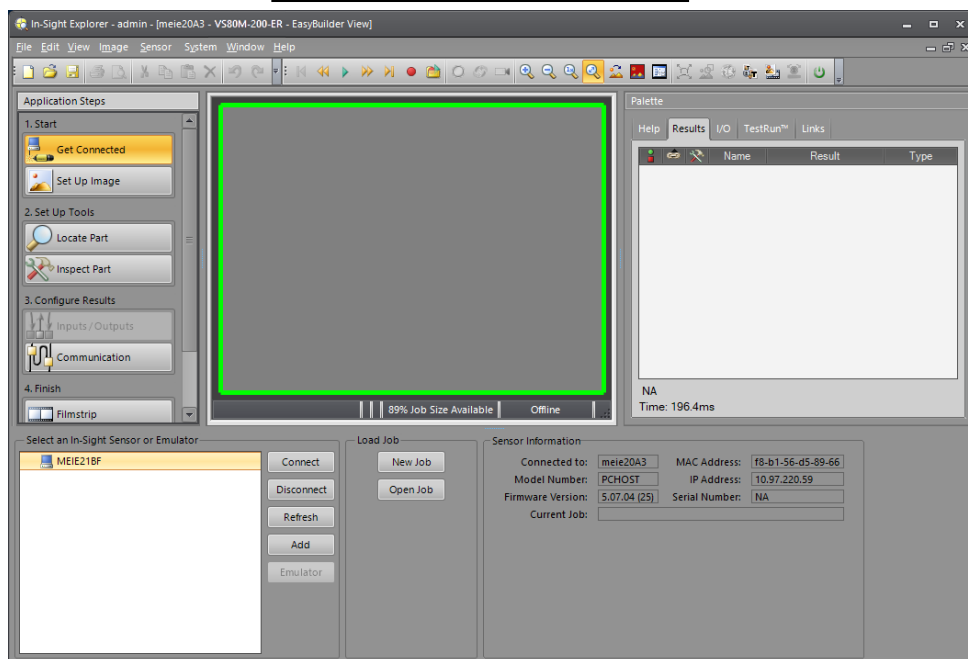
(<http://www.mitsubishielectric.co.jp/fa>)

2. Execute the downloaded file.

3. Open In-Sight Explorer. The Start screen (shown below) will appear.

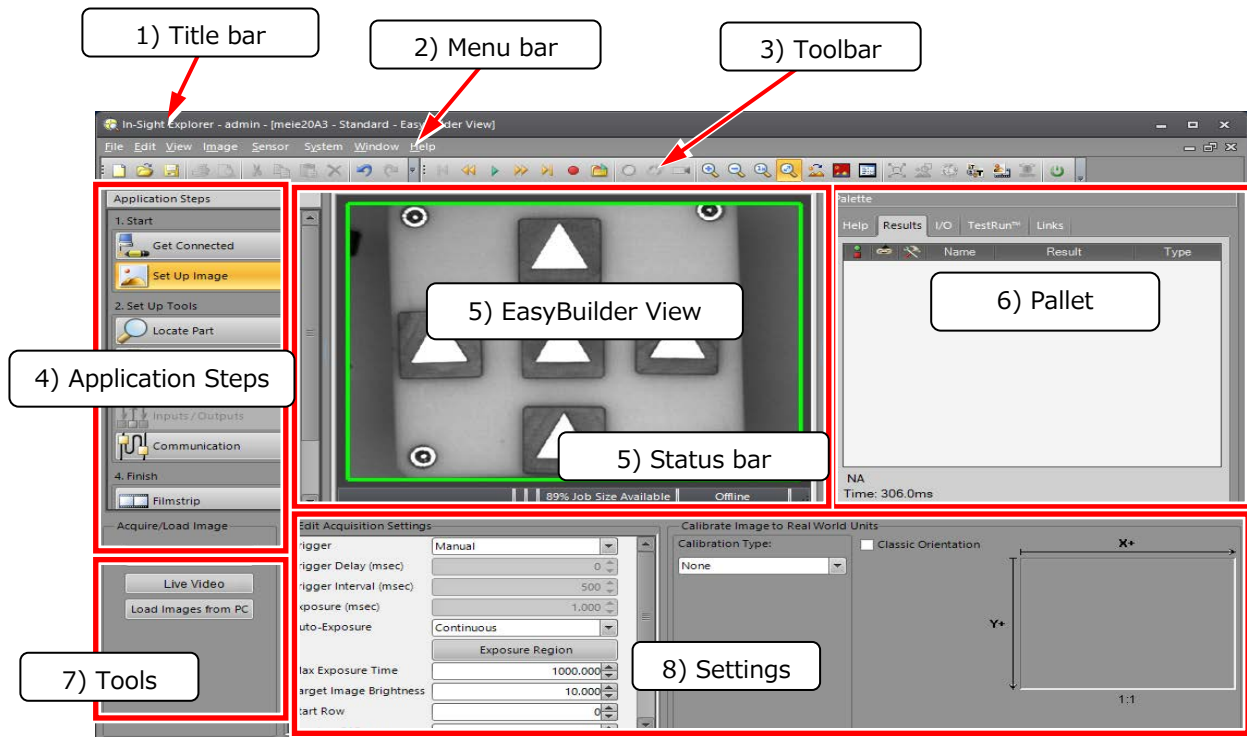
(Other options will become available once a vision sensor has been connected)

In-Sight Explorer Start screen



1.3.2 In-Sight Explorer window layout

The image below depicts the layout of In-Sight Explorer which is centered around an image.



1) Displays the name of the current job.

2) In-Sight Explorer menu bar

3) Three toolbars are available: the Standard toolbar, the Explorer toolbar, and the Job display toolbar.

4) Displays application steps in the order needed to make a job.

Clicking each step displays tools and settings in a pane below the Application Steps window.

Refer to Application Steps (1.3.3 Application Steps) for information on each step.

5) Displays the image from the vision sensor camera.

Pixels (coordinates) and the vision camera status (online/offline) are displayed in the bottom right status bar.

6) Displays information such as user-defined points, user-defined lines, and calibration results.

7) Displays tools and settings for each application step.

8) Displays information such as setting fields.

1.3.3 Application Steps

This page explains the application steps found in In-Sight Explorer.

Clicking on each application step displays applicable tools and settings for that step in the pane below.

Application Steps	Sub-steps	Description
Start	Get Connected	Used to connect vision sensors that are on the network.
	Set Up Image	Used to adjust image acquisition settings (trigger/imaging). The image acquisition method (trigger/live video etc.) and calibration type can be selected.
Set Up Tools	Locate Part	Used to define a workpiece reference point (feature) and set the area where the feature should be recognized.
	Inspect Part	An assortment of tools required for the vision sensor can be found here. Used to create things such as user-defined points, user-defined lines and carry out N point calibration.
Configure Results	Inputs/Outputs	Used to configure the connection settings of I/O modules and input/output operations.
	Communication	Used to configure communication to external devices and configure the settings of the data output from an identified pattern.
Finish	Filmstrip	Used to playback recorded images. Buffers job images sequentially and displays them. The filmstrip is located under EasyBuilder View.
	Save Job	Used to save the job while the vision sensor is offline. * Jobs are not automatically saved. Save your work periodically. Saved jobs do not contain communication and input/output settings.
	Run Job	Used to run the job.

1.3.4 Frequently used operations

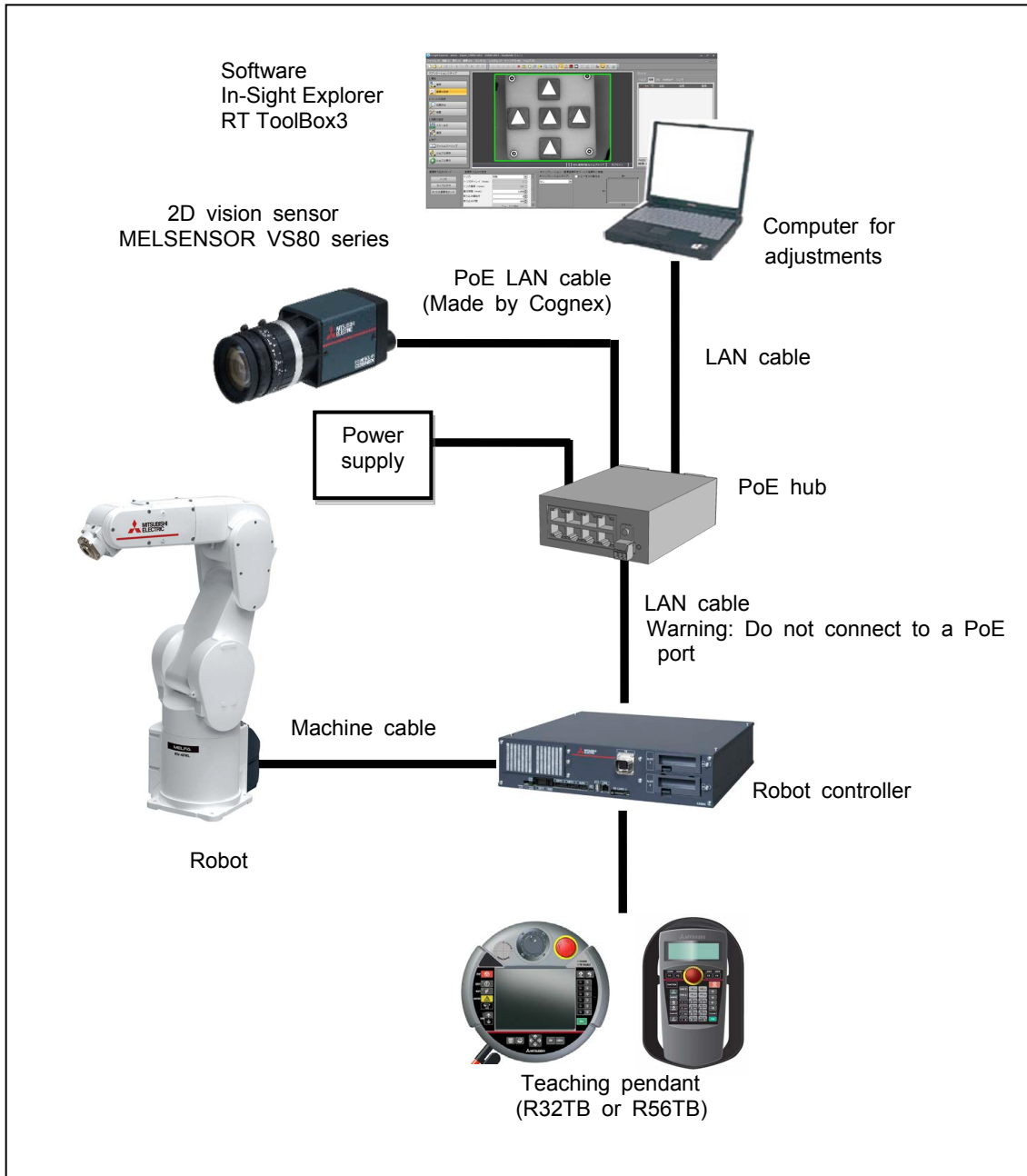
Operation	Description
Continuous trigger	Used to acquire images from the vision sensor. Periodically acquires and displays images. Intervals in which images are acquired can be specified.
Manual trigger	Used to acquire images from the vision sensor manually.
Live Video	Displays vision sensor images in real time. (Some operations in In-Sight Explorer become unavailable when Live Video mode is used.)
Load Images from PC	Used to load images from the computer.

1.4 System architecture

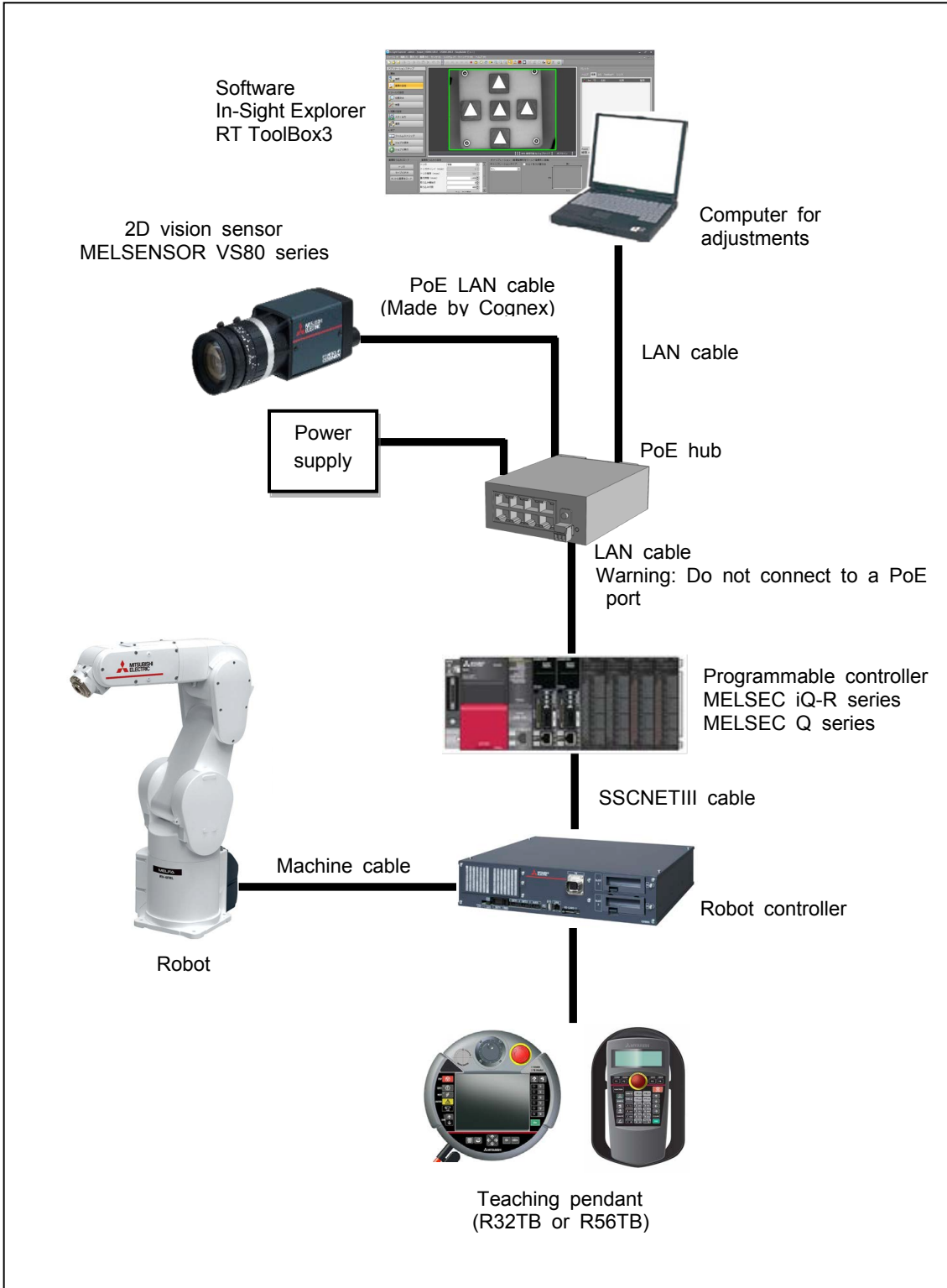
In order to use vision sensors, the devices in the diagram below are required.

The type of robot controller used affects what devices are needed and how they are connected to each other. Refer to the System architecture diagrams for each controller before connecting a camera to the robot controller.

1.4.1 D Type controllers



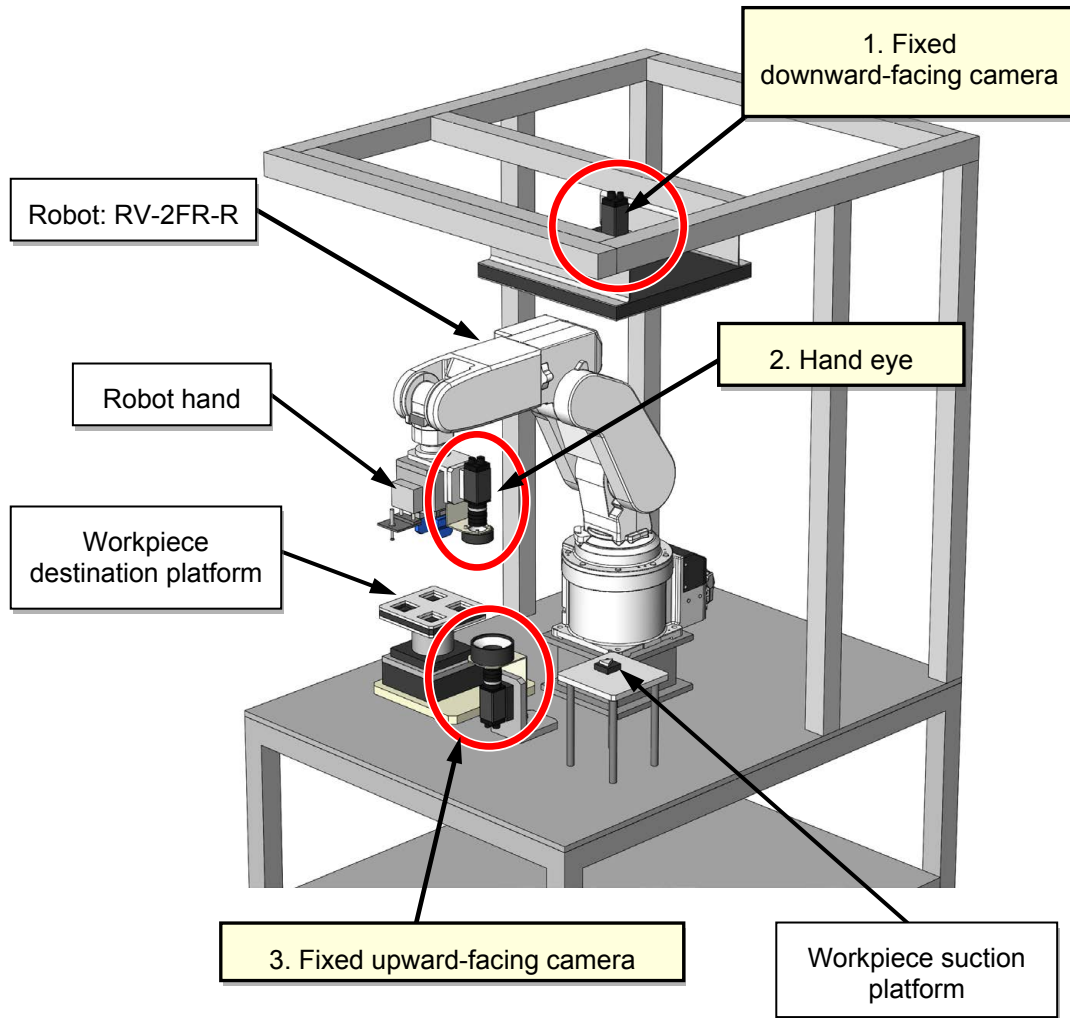
1.4.2 R Type/Q Type controllers



1.5 Types of vision sensors and their advantages and disadvantages

The example below shows three robot/vision sensor combinations.

- 1. Fixed downward-facing camera (set above the workpiece)
- 2. Hand eye (attached to the robot's hand)
- 3. Fixed upward-facing camera (set below the workpiece)

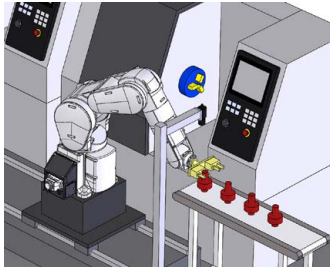


2D vision sensor cell

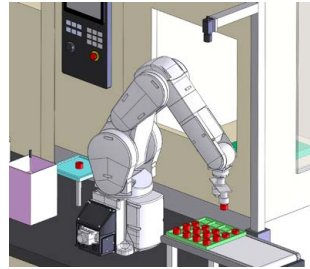
■ Advantages & disadvantages (○ = advantage / ▲ = disadvantage)

1. Fixed downward-facing camera

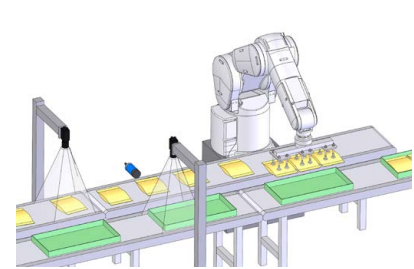
- Using a fixed downward-facing camera is the standard application of a vision sensor. Combining a vision sensor with a robot's multi-task function allows the vision sensor to search for another workpiece while the robot is outside its field of view (FOV), i.e., when the robot is transporting a workpiece. This helps improve cycle time.
- ▲ However, because the camera's position and FOV are fixed a different camera with a narrow FOV and a large resolution may be required for high-precision tasks. This leads to increased costs and limits to the types of systems that can be constructed.



Loading/unloading of processing machines



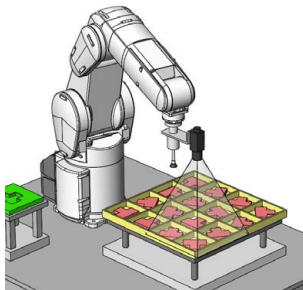
Pallet picking



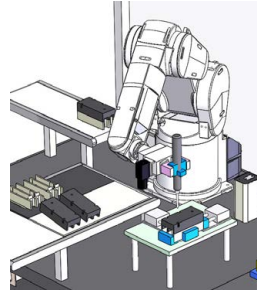
Conveyor tracking

2. Hand eye

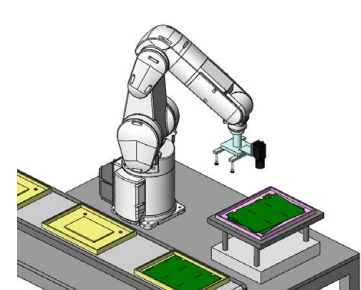
- Unlike a fixed downward-facing camera, the hand eye's FOV can be narrowed.
- This is because the hand eye's FOV is not restricted to the position where it picks up a workpiece and includes the full operating range of the robot.
- ▲ The downside of a hand eye is that it has a longer cycle time than a fixed downward-facing camera as it cannot do other things while processing identification data.



Pallet segmentation



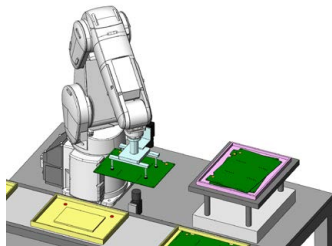
Screw tightening



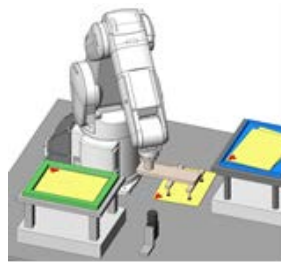
Workpiece positioning

3. Fixed upward-facing camera

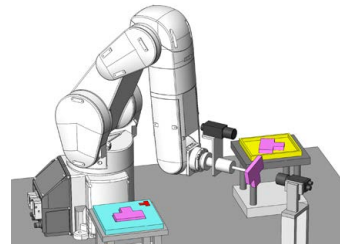
- A fixed upward-facing camera is generally used to help adjust the position of the workpiece the robot is holding.
 - In theory, it could be used for high precision work if the camera has a good view of the workpiece.
 - The camera could also be used if it does not matter if the hand deviates from its position when it grasps the workpiece.
- ▲ A fixed upward-facing camera, however, is rarely used on its own and it is assumed that it will be used with either one or both of the two cameras mentioned above. This means that system costs and camera processing times cannot be reduced.



Grasp position adjustment



Grasp position adjustment



Side camera

1.6 Steps required for automatic operation

1.6.1 Overview

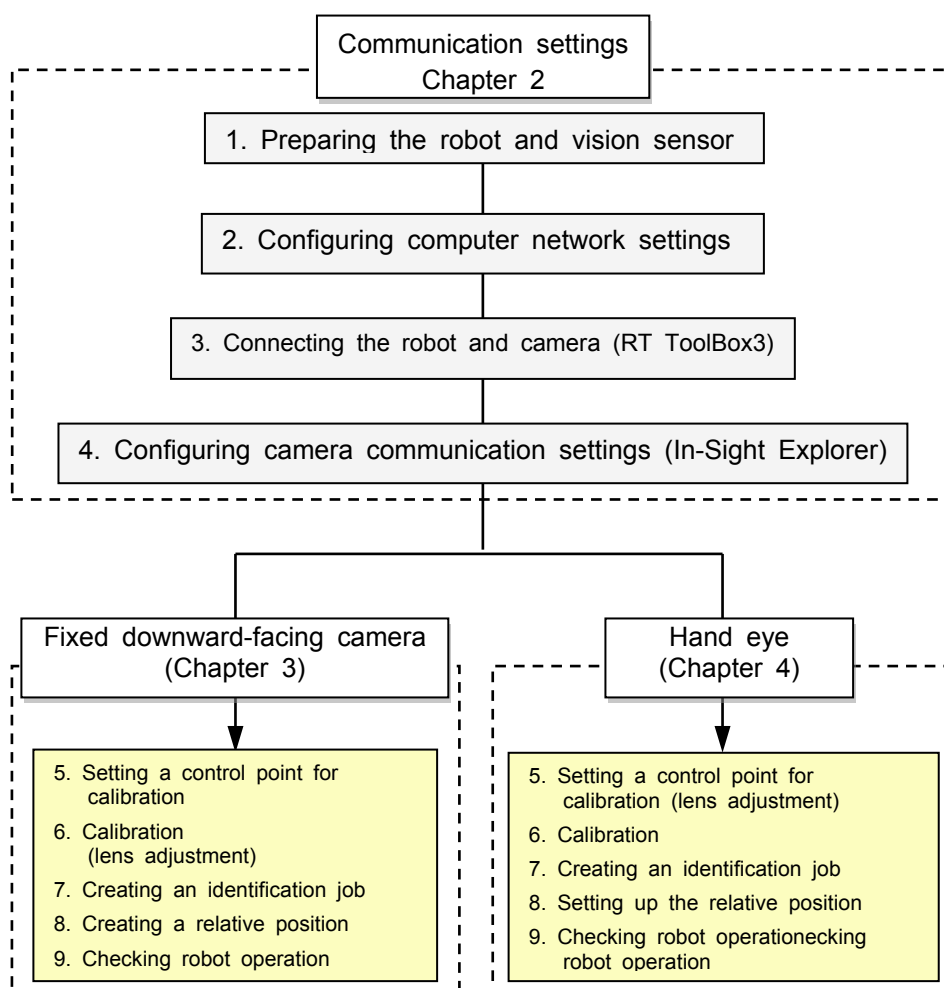
The following is an overview of the steps required to make automatic operation possible.

In this seminar we will learn about two types of camera; the fixed downward-facing camera and the hand eye.

- (1) Fixed downward-facing camera
- (2) Hand eye (connected to the robot hand)

Steps 1 to 4 of the communication settings are used for both the hand eye and the fixed downward-facing camera.

Steps 5 to 9 are the steps used to configure both cameras. Each camera is configured differently



1.6.2 Communication settings overview

Communication settings used in this seminar are shown below.

- Common settings

Robot IP address	Subnet mask	Port No.
192.168.1.20	255.255.255.0	23

- Camera specific settings

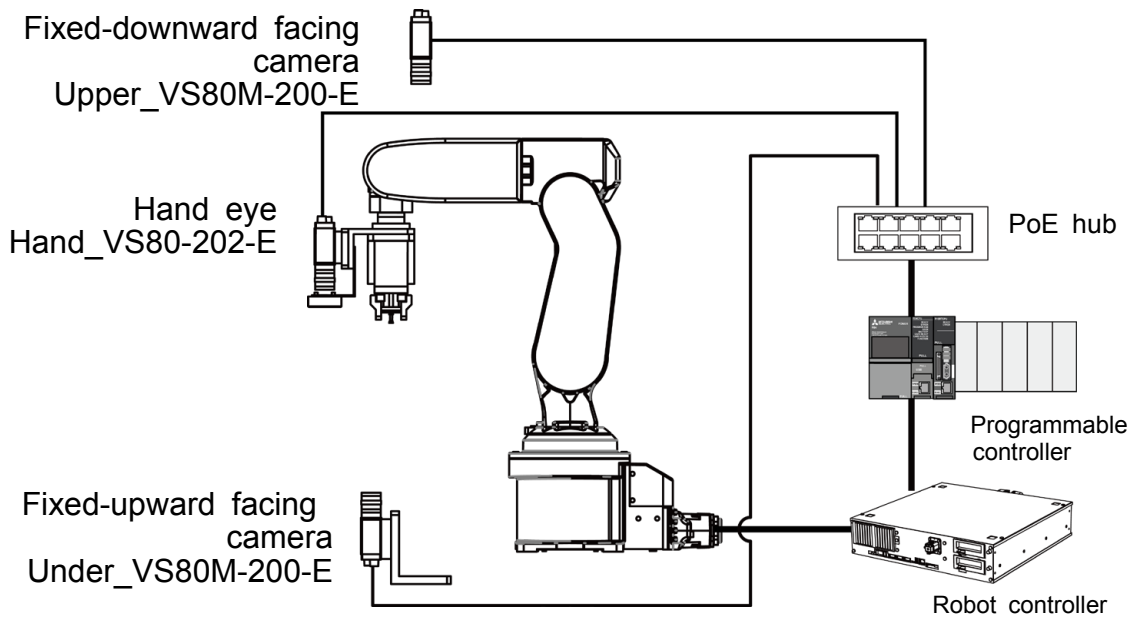
Camera type	Device	Camera IP address (*1)	Assigned to	Camera host name
Fixed downward-facing camera	OPT13	192.168.1.3	COM3	Upper_VS80M-200-E
Hand eye	OPT14	192.168.1.4	COM4	Hand_VS80-202-E
Fixed upward-facing camera	OPT15	192.168.1.5	COM5	Under_VS80-200-E

*1) The IP address of each camera needs to be set as the cameras will be connected to a network.

1.6.3 Overview of programs used for each camera

The programs used for each camera in this seminar are shown below.

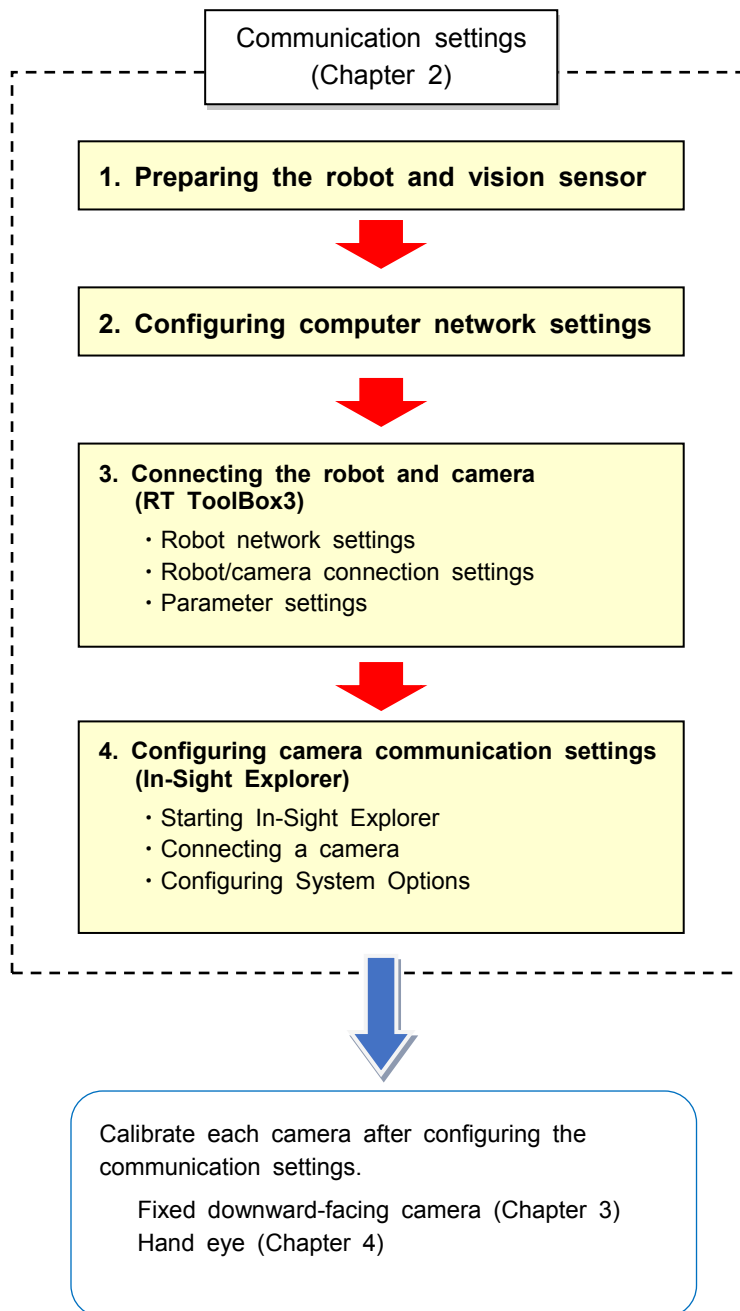
	Fixed downward-facing camera	Hand eye	Fixed upward-facing camera
External variables defined by the user	UBP		
Creation of a control point used for calibration	TLUP	HND1	TLLW
Calibration	-	HND2	-
Identification job creation	-	HND3	DVS1
Relative position creation (grasp position, position adjustment)	UVS1 UVS2	HND3	-
Automatic operation	UVS2	HND4	DVS2



Chapter 2 - Communication settings and lens adjustment

This chapter explains how to configure the communication settings.

■ Steps required to configure the communication settings



2.1 Preparing the robot and vision sensor

Referring to the system architecture figure, prepare the required devices and wire them.
(Refer to the system architecture figure "1.4 System architecture".)

2.2 Configuring computer network settings

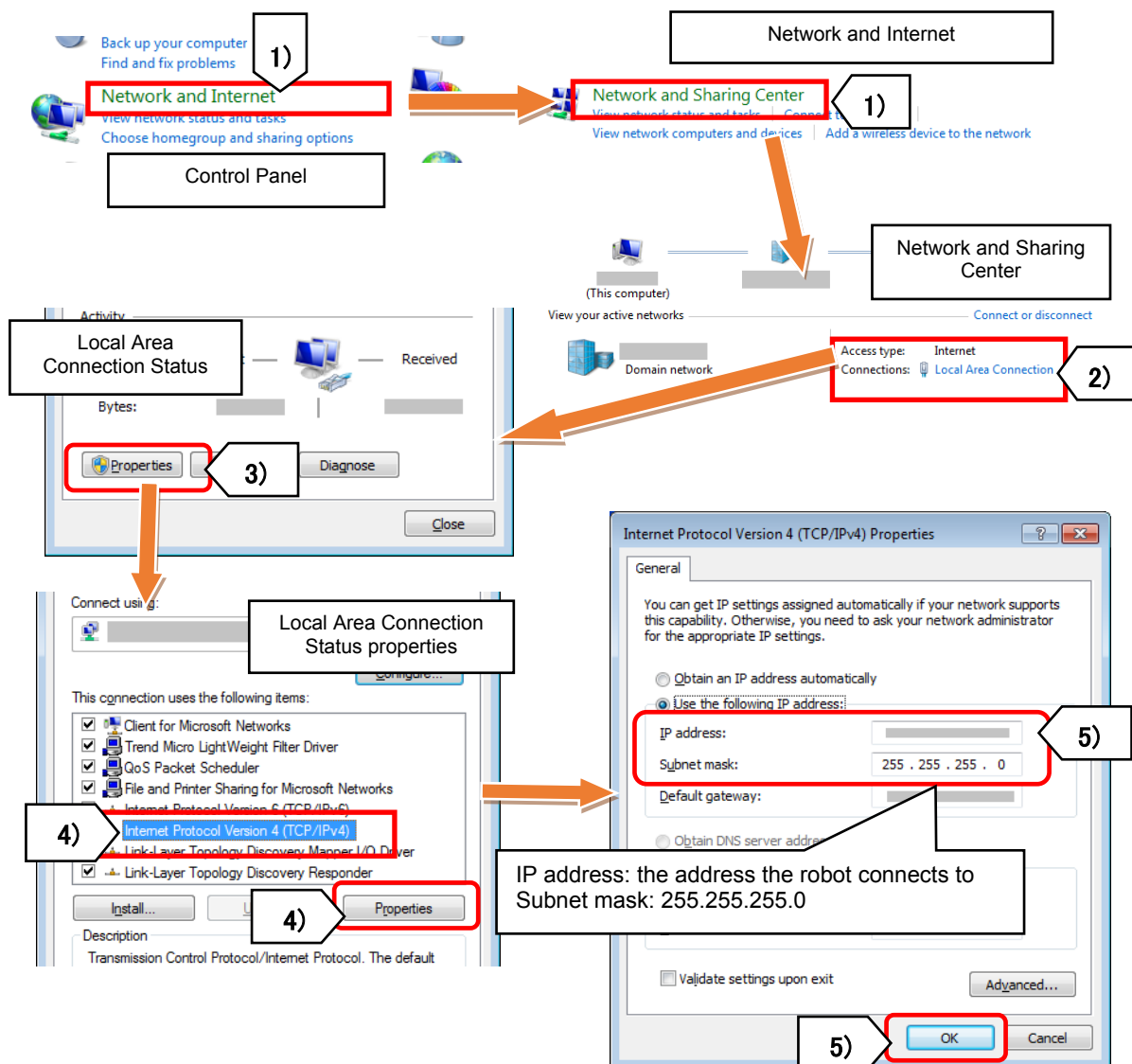
Configure the network settings of the computer that is running RT ToolBox3 and In-Sight Explorer.
The settings for the computers used in this seminar have already been configured. Follow the steps below when working outside of this seminar.

(1) Set the computer IP address (Procedure for Windows 7®.)

Set the IP address for the computer you are using.

- 1) Go to Control Panel → Network and Internet → Network and Sharing Center.
- 2) Click Local Area Connection under View your active networks.
- 3) Click Properties in the Local Area Connection Status window.
- 4) Select Internet Protocol Version 4 (TCP/IPv4), then click Properties.
- 5) Select Use the following IP address and input an IP address that matches the network that the robot will connect to.

In the Subnet mask field enter 255.255.255.0, then click OK and close the window.

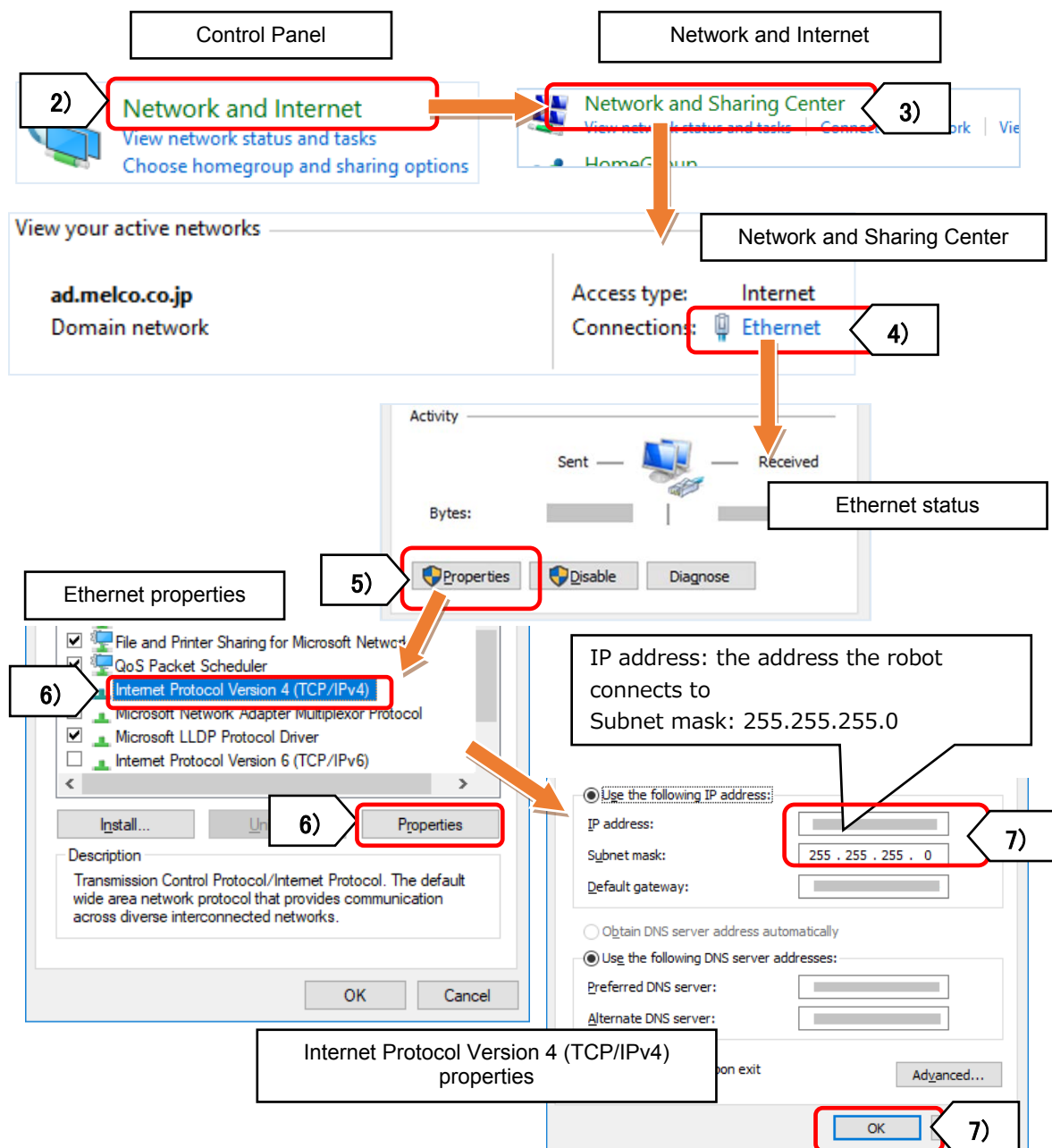


(1) Set the computer IP address (Procedure for Windows 10®.)

Set the IP address for the computer you are using.

- 1) Go to Start → Windows System → Control Panel.
- 2) In the Control Panel, click Network and Internet.
- 3) Then click Network and Sharing Center.
- 4) Double click Ethernet under View your active networks.
- 5) Click Properties in the Ethernet Status window.
- 6) Select Internet Protocol Version 4 (TCP/IPv4), then click Properties.
- 7) Select Use the following IP address and input an IP address that matches the network that the robot will connect to.

In the Subnet mask field enter 255.255.255.0, then click OK and close the window.



2.3 Connecting the robot and camera (RT ToolBox3)

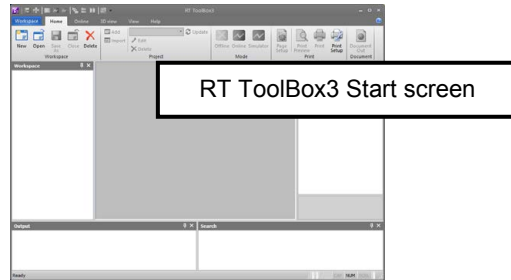
Using RT ToolBox3, we will configure the communication settings to connect the robot and camera.

2.3.1 Robot network settings

(1) Create a workspace.

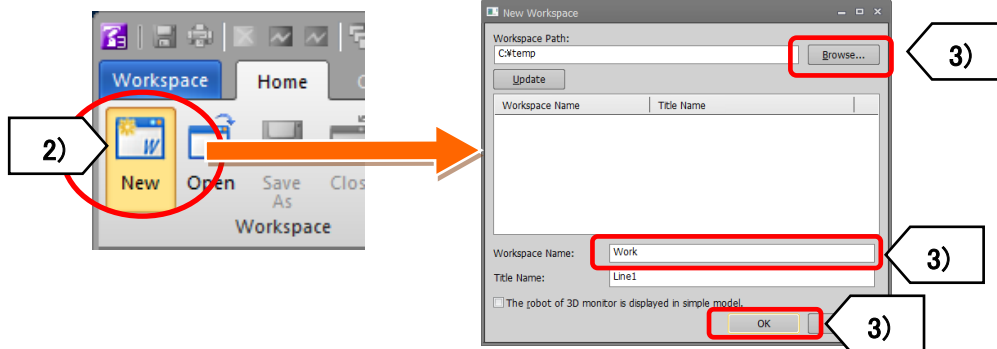
(1.1) Start RT ToolBox3 and create a new workspace.

1) Start RT ToolBox3.



2) Click New in the menu bar on the Home tab. The New Workspace window will appear.

3) Specify a place to save the workspace, then enter a workspace name and click OK.



(2) Step 1: Outline (project name and comment)

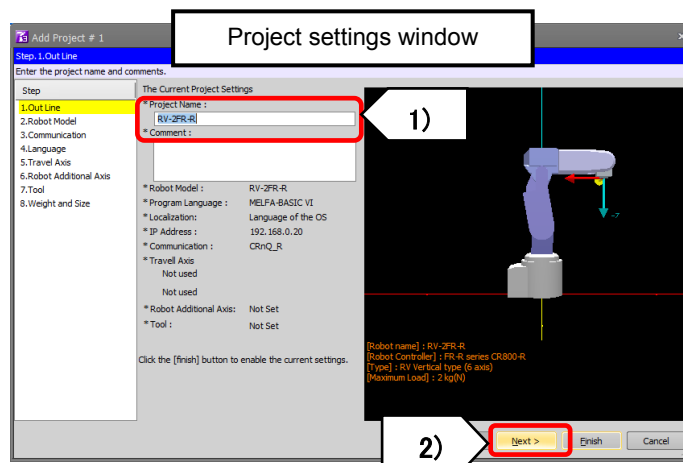
(2.1) Enter a project name and comment

The project window will appear. Here we will configure the project settings.

1) Enter a project name.

The project name will be set as either RC1 or RC2 by default (this can be left as it is).
Feel free to write any notes in the comment section.

2) Click Next >.



(3) Step 1: Outline (project name and comment)

(3.1) Select the robot and controller type.

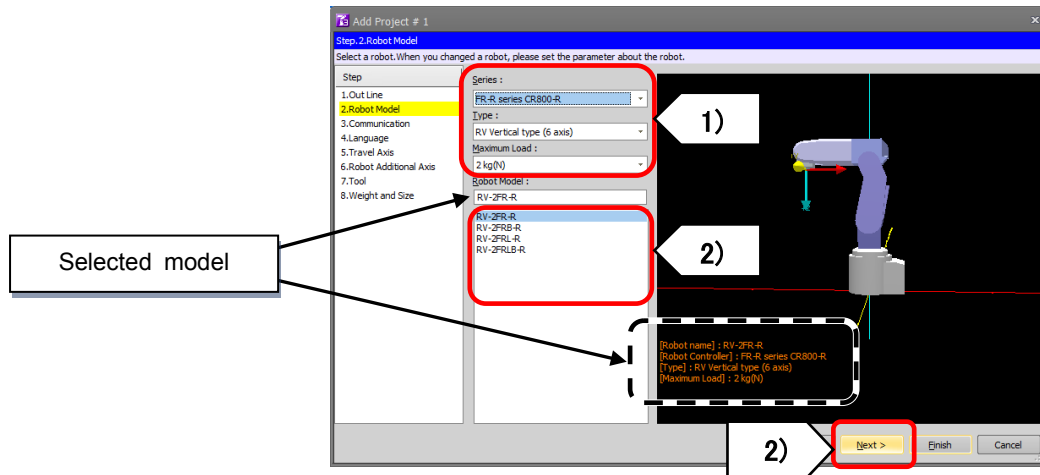
1) Select the robot and controller that will be used for the project.

The series, type and maximum load of the robot and controller can be narrowed down using the drop-down boxes.

2) Select the model from the list that appears below the drop-down boxes and click Next >.

The RV-2FR-R will be used in this seminar. Select the following from the drop-down boxes.

- Series: FR-R series CR800-R
- Type: RV Vertical type (6 axis)
- Maximum load: 2 kg
- Robot model: RV-2FR-R



(4) Step 3: Communication settings

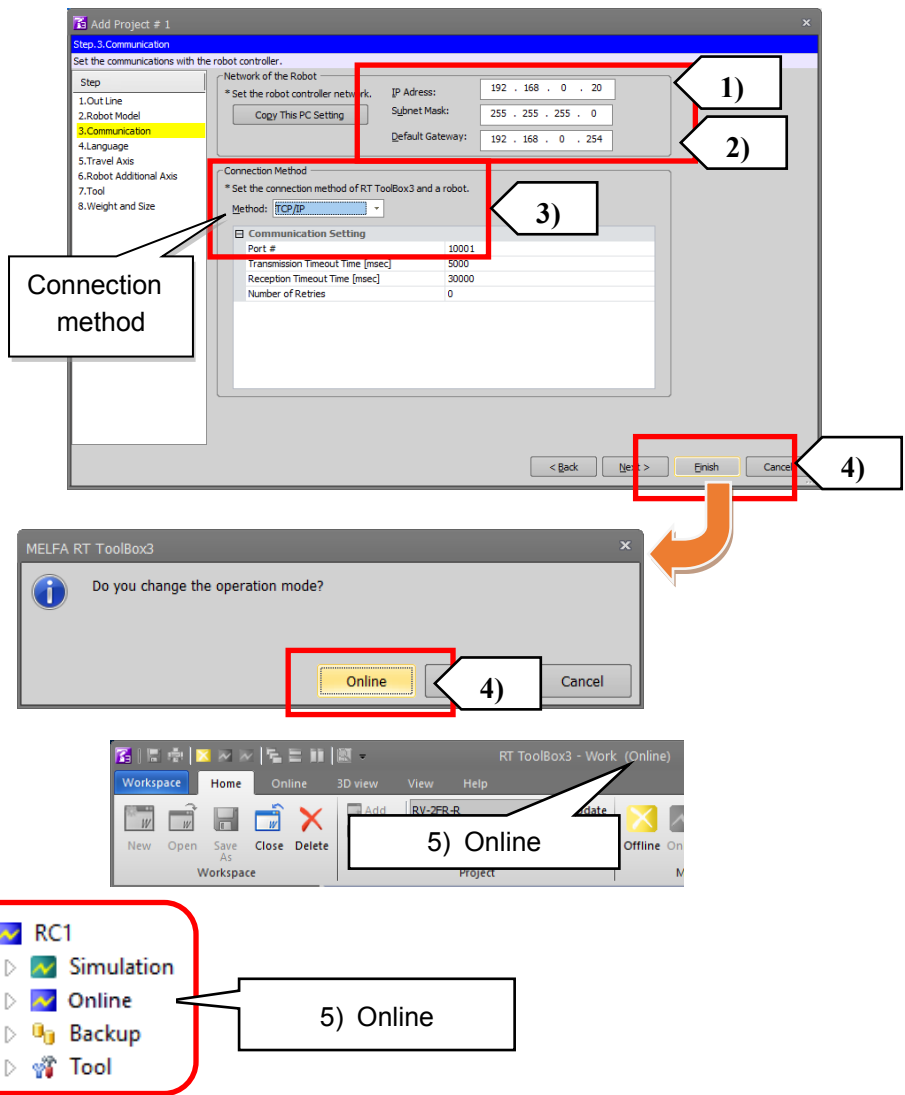
(4) Configure IP address and communication settings.

- 1) Enter the robot/controller IP address in the IP address field under Network of the robot.
- * Set the robot IP address to 192.168.1.20 for the purposes of this seminar. (*1)
- 2) Enter 255.255.255.0 in the Subnet Mask field.
- 3) Select TCP/IP for the connection method.

* Settings

Robot IP address*1	192.168.1.20
Subnet mask	255.255.255.0
Connection method	TCP/IP

- 4) This completes the project settings used in this seminar.
Click Finish and change the operation mode to Online.
- 5) Once the robot and robot controller are connected, Online will appear in the project tree.



*1 Regarding the robot/robot controller IP addresses:

- The default IP addresses for Q/R Type robots are 192.168.100.1. The default IP address for the D Type robot controller is 192.168.0.20.
- Use the parameter "NETIP" to check the IP address of the robot or robot controller.
- Reset the robot controller after changing an IP address.

2.3.2 comRobot and camera connection settings

Configure COM port and parameter communication settings to connect the robot and vision sensors. In this seminar, we will use the following settings for each vision sensor.

* Cameras used in this seminar and COM port communication settings

Camera type	Device	Camera IP address*1	Assigned to	Camera host name
Fixed downward-facing camera	OPT13	192.168.1.3	COM3	Upper_VS80M-200-E
Hand eye	OPT14	192.168.1.4	COM4	Hand_VS80-202-E
Fixed upward-facing camera	OPT15	192.168.1.5	COM5	Under_VS80-200-E

*1) Each camera requires its own IP address to connect to the network. Prepare an IP address to assign to each camera in advanced.

(1) Device parameter settings

(1.1) Select a device.

(The settings for the fixed downward-facing camera are used as an example in this section.)

- 1) In the project tree go to Online → Parameter → Communication Parameter → Ethernet.
- 2) Click Device & Line in the Ethernet window.
- 3) In the device list, double click on each device from OPT13 to OPT19.
(The Set button can be used instead of double clicking.) (OPT13 is used as an example)
- 4) The Device Parameter Setting window will appear.



Do not use OPT11 or OPT12. RT ToolBox3 uses OPT11 so the operation mode will switch to Offline if OPT11 is selected.

The screenshot illustrates the configuration process in three main stages:

- Project Tree:** Step 1) shows the navigation path: Online → Parameter → Communication Parameter → Ethernet.
- Ethernet Window:** Step 2) shows the 'Device & Line' menu item selected in the Ethernet window.
- Device List:** Step 3) shows the 'Device List' table with 'OPT13' selected. The table contains the following data:

Device	Mode	IP Address
OPT11	1: Server	192.168.0.2
OPT12	0: Client	192.168.0.3
OPT13	0: Client	192.168.0.4
OPT14	0: Client	192.168.0.5
OPT15	1: Server	192.168.0.6
OPT16	1: Server	192.168.0.7
OPT17	1: Server	192.168.0.8
- Device Parameter Setting Window:** This window is opened for 'OPT13' and shows the following settings:
 - Device: OPT13
 - Autconfiguration: None
 - Mode: (NETMODE(3)) 1: Server
 - Port #: (NETPORT(4)) 10003
 - Protocol: (CPRCE13) 0: No-procedure
 - Exit Code: (NETTERM(3)) 0: No-included
 - Packet Type: (CTERME13) 0: CR

(1.2) Configure device parameter settings

- 1) In the Device Parameter Setting window, select Network Vision Sensor (2D) from the drop-down box in the Autoconfiguration field.
- 2) Enter the IP address which you would like to assign to the camera selected in the IP address field. (E.g. 192.168.1.3)
- 3) The Port # is the same as what the camera port is set to. Make sure that Port # is set to 23 for this seminar.
- 4) Select any COM port from COM1 to COM8 from the Allocation drop-down box, then click OK. (COM3 is used as an example)

Network vision sensor (2D) settings

* For VS80 and VS70 series cameras, selecting Network vision sensor (2D) will automatically assign settings for fields other than IP address and Allocation.

(1.3) Configure the settings of the remaining cameras.

- 1) Follow Steps 1.1 and 1.2, then configure the settings as shown below.
- 2) In the Ethernet window, click Write to write the settings to the robot controller.
- 3) A dialog box with the message "Are you sure want to restart the robot controller?" will appear. Click Yes. The robot controller will restart.

The Ethernet window after settings have been made (example).

Settings of OPT13 to OPT15 complete

Settings of COM3 to COM5 complete

Device	Mode	IP Address	Port #	Protocol
OPT11	1: Server	192.168.0.2	10001	0: No-procedure
OPT12	1: Server	192.168.0.3	10002	0: No-procedure
OPT13	0: Client	192.168.1.3	23	2: Data Link
OPT14	0: Client	192.168.1.4	23	2: Data Link
OPT15	0: Client	192.168.1.5	23	2: Data Link
OPT16	1: Server	192.168.0.7	10006	0: No-procedure
OPT17	1: Server	192.168.0.8	10007	0: No-procedure
OPT18	1: Server	192.168.0.9	10008	0: No-procedure
OPT19	1: Server	192.168.0.10	10009	0: No-procedure

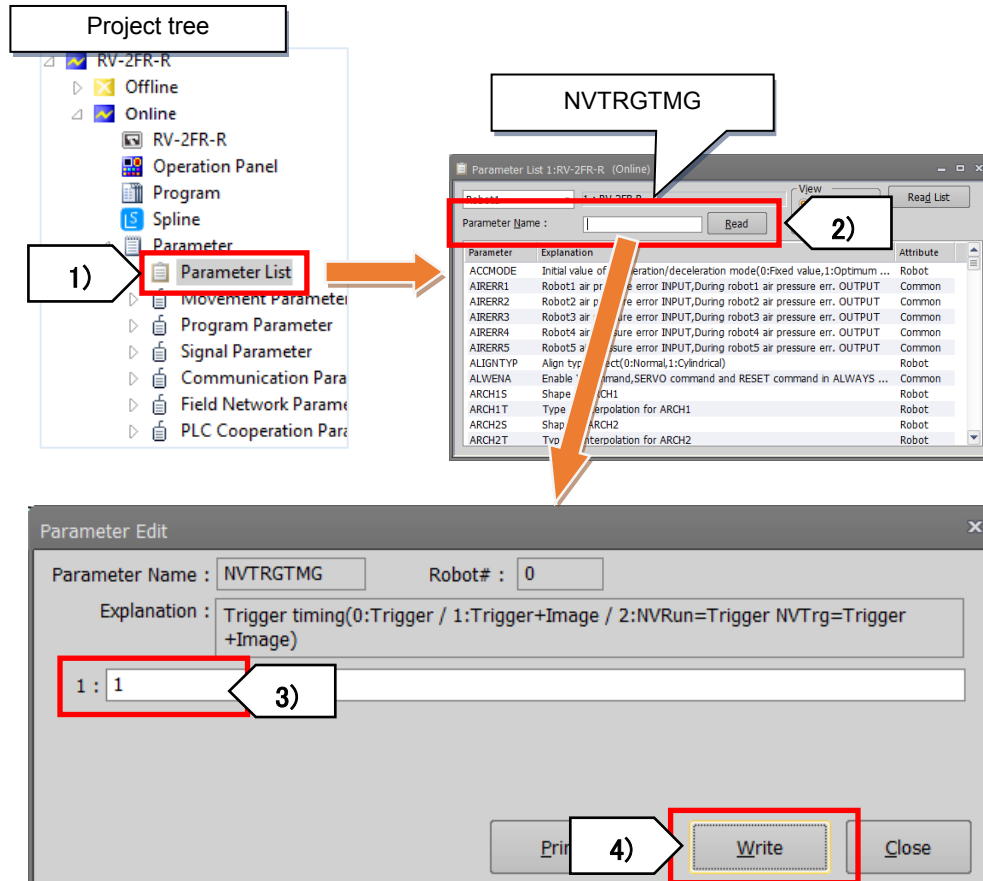
COM1: (No Selection) v
 COM2: (No Selection) v
 COM3: OPT13 v
 COM4: OPT14 v
 COM5: OPT15 v
 COM6: (No Selection) v
 COM7: (No Selection) v
 COM8: (No Selection) v

(2) Parameter settings

Parameters are set to call up imaging commands and acquire data after the camera has finished processing an image.

(2.1) Set "1" in the parameter "NVTRGTMG".

- 1) Go to Online → Parameter → Parameter List in the project tree.
- 2) Search for the parameter "NVTRGTMG".
- 3) Set "1" in the parameter "NVTRGTMG". (The default value is 2)
- 4) Click Write to write the parameter to the robot controller.
- 5) A dialog box with the message "Are you sure want to restart the robot controller?" will appear. Click Yes. The robot controller will restart.



* This completes the connection settings for the robot and camera.

Additional

Parameter NVTRGTMG

If parameter NVTRGTMG is set to the default value "2", the NVRun command*1 starts processing the next command without waiting for image processing to finish. Therefore, the results of previous identification data may be read if the EBRead command*2 is executed immediately.

*1 NvRun command: A command which specifies the job, which vision sensor to control and when to activate a trigger.

*2 EBRead command: A command which reads data from the camera.

For information on NVTRGTMG, refer to "Appendix 3 2D Vision sensor parameters".

2.4 Configuring camera communication settings (In-Sight Explorer)

In this step, we will configure the camera communication settings with In-Sight Explorer.

In order to configure vision sensor settings, In-Sight Explorer needs to be installed on the computer connected to the cameras.

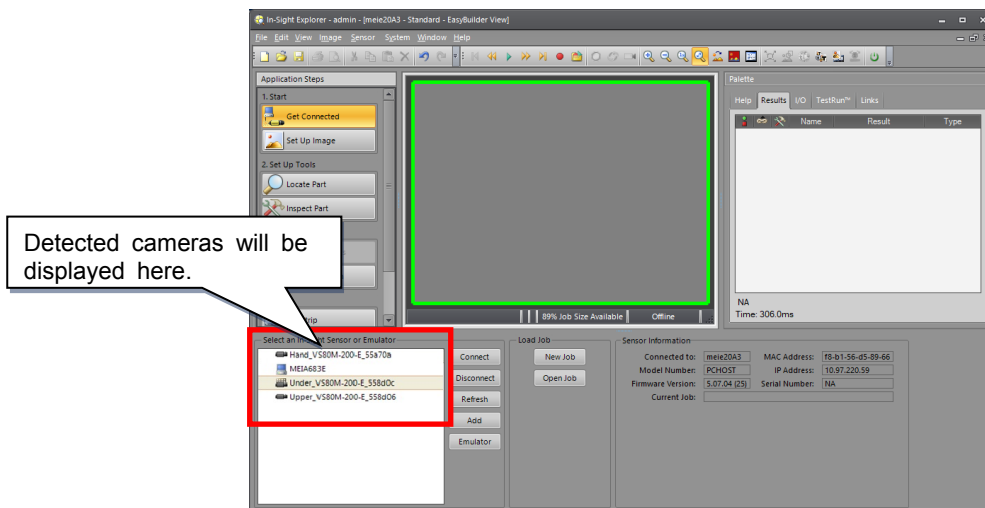
Refer to "1.3.1 Installing In-Sight Explorer" for information on how to install In-Sight Explorer.

2.4.1 Starting In-Sight Explorer

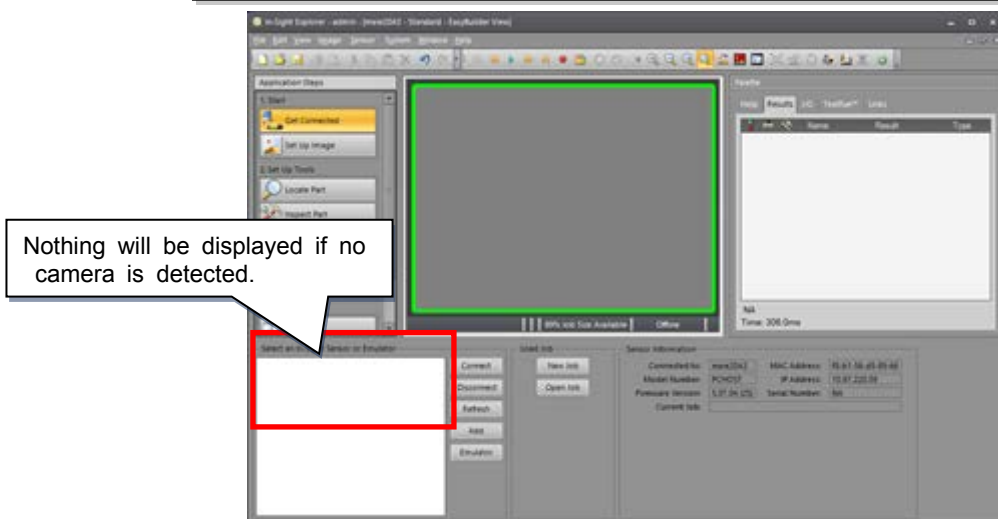
(1) Start In-Sight Explorer

1) There will be times when In-Sight Explorer detects cameras on start up and times when it does not.

At least one camera detected → Go to 2.4.2.1.



No cameras detected → Go to 2.4.2.2.



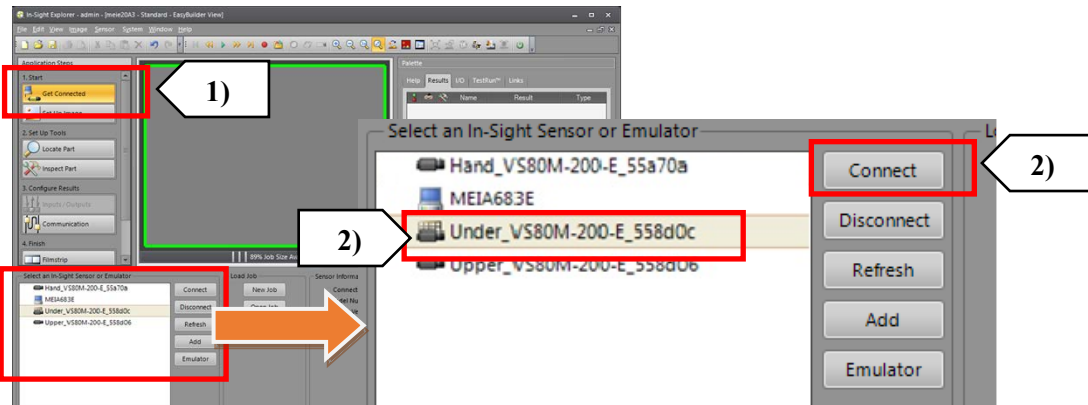
2.4.2 Connecting cameras

2.4.2.1 At least one camera detected

(1) Check the IP address.

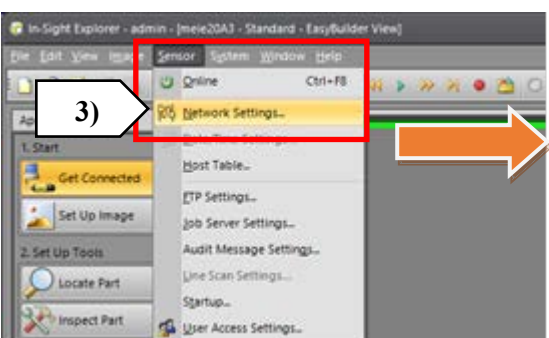
- 1) Select Get Connected from the Application Steps window.
- 2) A list of detected cameras will appear under Select an In-Sight Sensor or Emulator. Choose a camera from the list and click Connect.
 - * If no cameras appear here, follow the instructions in "2.4.2.2 If no cameras are detected".
- 3) Go to Sensor on the menu bar and select Network Settings. Ensure that the IP Address, Subnet Mask, and Telnet Port (Port No.) are the same as the settings that are stated in "2.3.2 Robot and camera connection settings". If there are any differences, adjust the settings as necessary, then click OK.

* Continue to Step 2.4.2.3 Configuring system options.

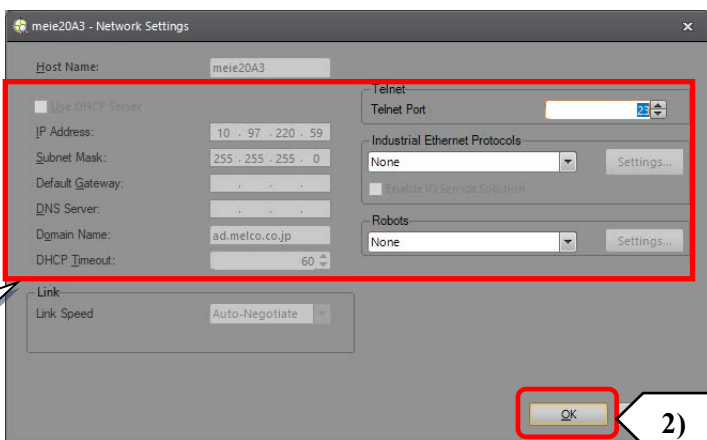


Select an In-Sight Sensor or Emulator

An image will be displayed once a camera is connected.



3)



meie20A3 - Network Settings

Host Name: meie20A3

Use DHCP Server

IP Address: 10 . 97 . 220 . 59

Subnet Mask: 255 . 255 . 255 . 0

Default Gateway: . . .

DNS Server: . . .

Dgmain Name: ad.melco.co.jp

DHCP Timeout: 60

Link Speed: Auto-Negotiate

Telnet: Telnet Port: []

Industrial Ethernet Protocols: None [Settings...]

Enable IP Sensor Solution

Robots: None [Settings...]

OK

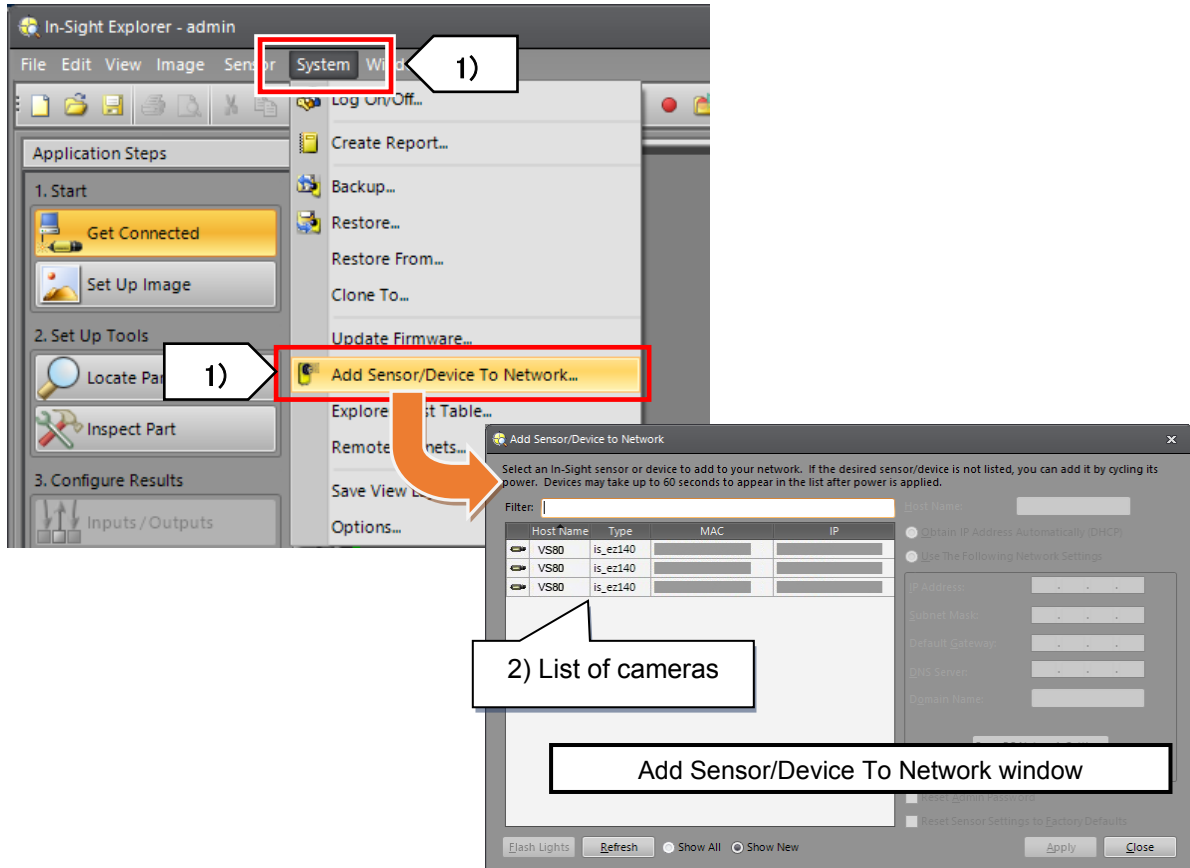
If the settings differ from the settings in "2.3.2 Robot and camera connection settings", adjust the settings to match the selected camera.

2.4.2.2 If no cameras are detected

If no cameras appear under Select an In-Sight Sensor or Emulator, they need to be found and recognized.

(1.1) Find the cameras.

- 1) Go to System on the menu bar and select Add Sensor/Device To Network.
- 2) The Add Sensor/Device To Network window will pop up and the cameras will soon be detected.
(The cameras will be detected, but not recognized by the robot at this point.)



(1.2) Configure the network settings so that the cameras are recognized by the robot.

1) Select the camera to configure its network settings.

The cameras can be identified by their Mac address.

2) A name can be set in the Host Name field to distinguish each camera. (Host names are optional.)

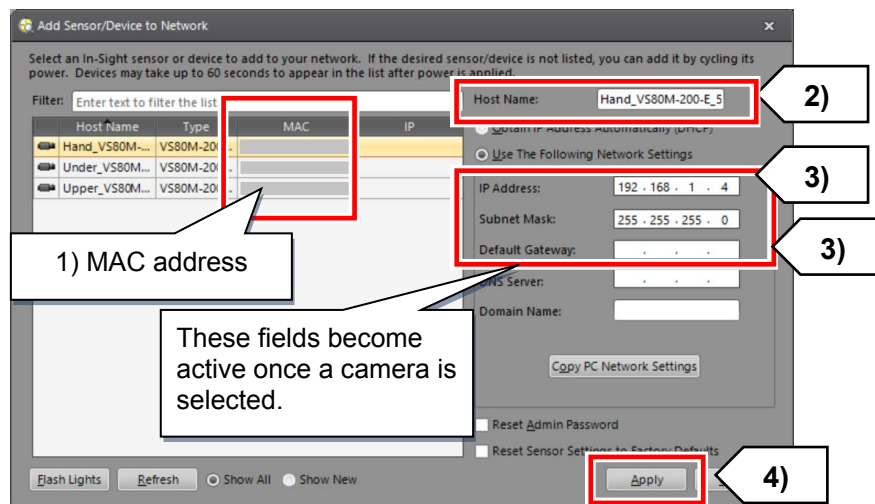
The following names are used in this seminar as examples.

Fixed downward-facing camera	Upper_VS80M-200-E
Hand eye	Hand_VS80-202-E
Fixed upward-facing camera	Under_VS80-200-E

3) Enter an IP address and Subnet mask.

Enter the same settings that are stated in "2.3.2 Robot and camera connection settings".

4) Click Apply.

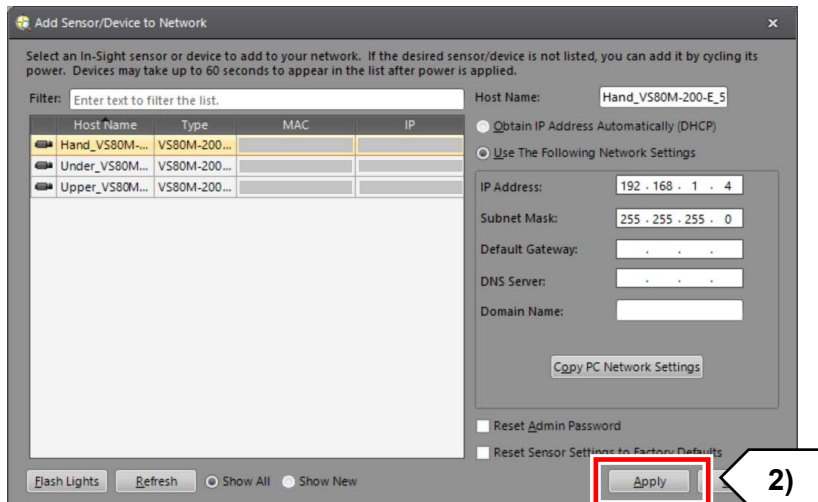


(1.3) Configure additional camera network settings.

1) Complete Step 1.2 for the other cameras to configure the network settings.

2) After configuring the settings, click Close.

The Add Sensor/Device To Network window will close, and then the cameras will restart.



(1.4) Connect the cameras.

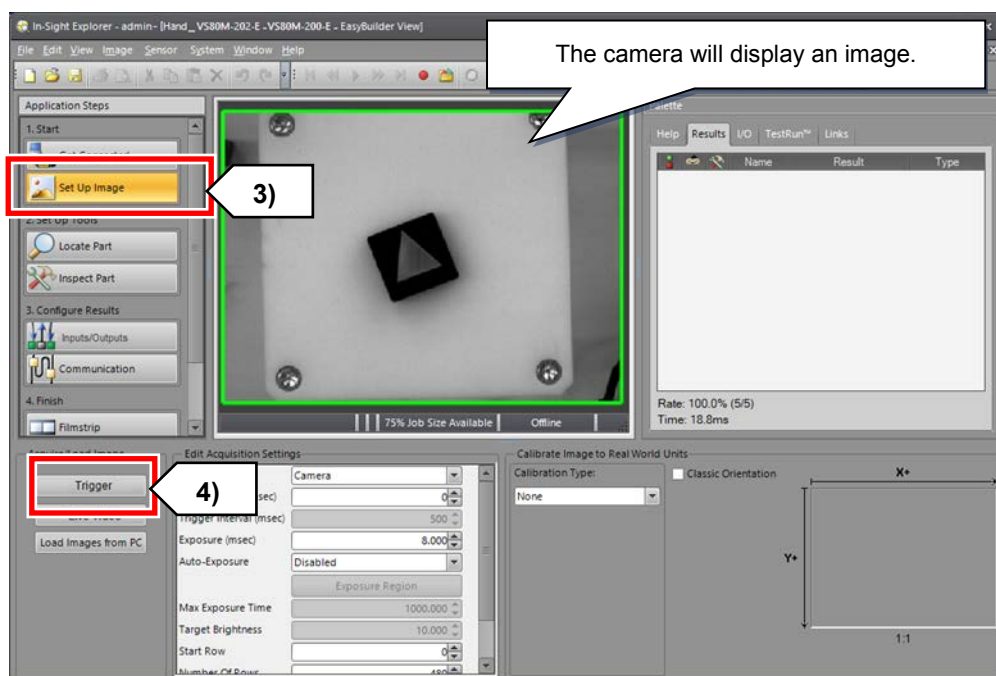
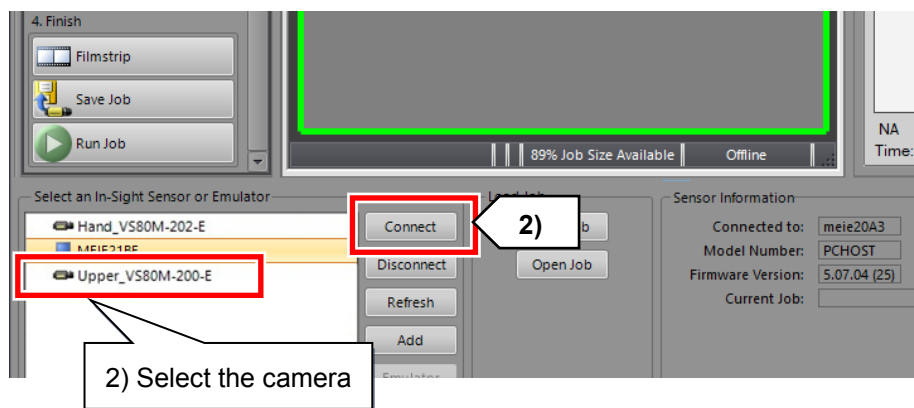
- 1) After the cameras have restarted, the names of the cameras will appear under Select an In-Sight Sensor or Emulator.
 - 2) Select the camera you want to use and click Connect to connect the camera.
 - 3) Select Set Up Image from the Application Steps window.
 - 4) Click Trigger under Acquire/Load Image to display the image of the selected camera.
- * If a camera cannot be connected, ensure that the IP address, Subnet Mask, and Telnet Port (Port No.) are the same as the settings that are stated in "2.3.2 Robot and camera connection settings".

How to check

Go to Sensor on the menu bar and select Network Settings.

If the IP Address, Subnet Mask, and Telnet Port (Port No.) are not the same as the settings that are stated in "2.3.2 Robot and camera connection settings", change them and click OK.

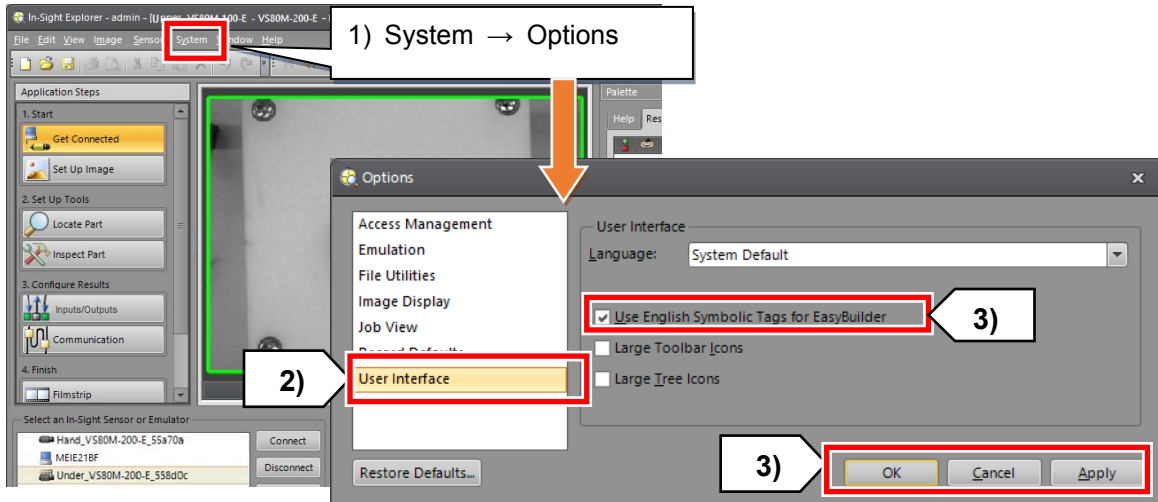
Reselect the camera, then click Connect and check that the camera is connected.



2.4.2.3 Configuring system options

(1) Configure system options

- 1) Go to System on the menu bar and select Options. The Options window will appear.
- 2) Select User Interface from the menu on the left.
- 3) Select the Use English Symbolic Tags for EasyBuilder check box. Click Apply then OK.



CAUTION If the Use English Symbolic Tags for EasyBuilder check box is not selected, data communication will become unstable when a program is executed.

* This completes the configuration of camera communication settings.

Additional

Common methods of acquiring

The following methods are the main methods used for acquiring images.

(1) Trigger

The camera acquires one image when the Trigger button is pressed.

When the trigger control is set to Manual, images can be acquired by pressing the F5 button.

Image acquisition settings can be configured in the Edit Acquisition Settings window, accessible by clicking on Set Up Image in the Application steps window.

Main trigger control methods

- Camera: Images are acquired upon the detection of a rising edge signal on the hardware input port of the vision system.
- Continuous: Images are acquired continuously using the intervals set in Trigger Intervals.
- External: Images are acquired on the command of an external device.
- Manual: Images are acquired when the function key (F5) is pressed.

(2) Live Video

Camera images are displayed in real time. Live view is also used when adjusting the lens.

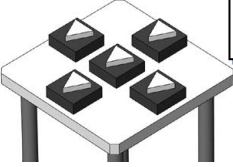
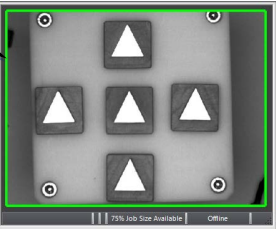
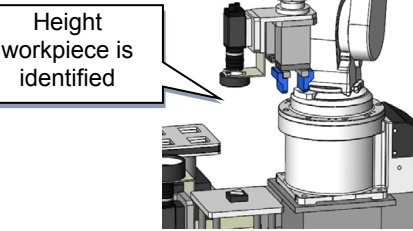

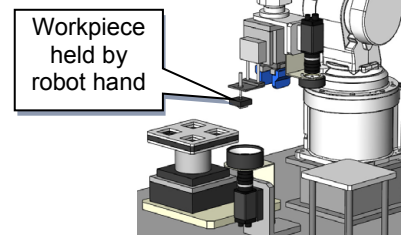

Notes

2.5 Adjusting the lens (focus and aperture)

This section explains how to adjust the focus and aperture (brightness) of the lens.

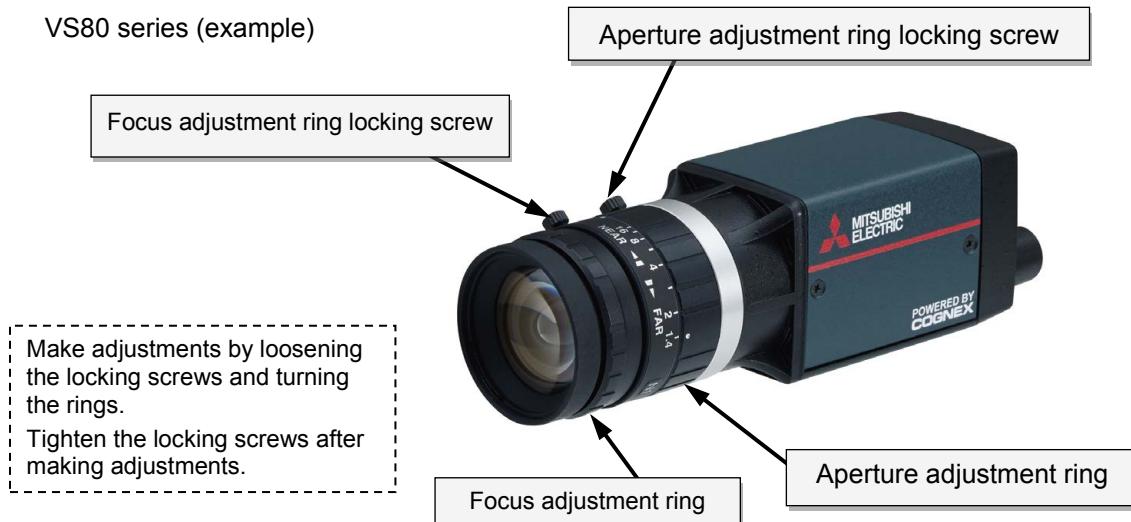
- Position the surface of the workpieces you want the camera to identify at a height that is within its field of view. Then adjust the focus and aperture.
- When adjusting the lens, ensure that the edges of the workpieces are clearly in focus.
- Lens adjustment methods and lens adjustment times differ depending on the camera type.

(1) Position the workpieces within the camera's FOV.

<p>Fixed downward-facing camera Place the workpieces in a position that fills most of the camera's FOV.</p>	 <p>Workpiece suction platform</p>	 <p>Identification range</p>
<p>Hand eye 1) Move the robot arm so that the hand camera is above where the workpiece is to be picked. Move the robot to a height which does not change the required FOV.</p>	 <p>Height workpiece is identified</p>	 <p>Identification range</p>
<p>Fixed upward-facing camera Pick up the workpiece with the robot hand. Move the robot to the height and position that is required for the camera to identify the workpiece.</p>	 <p>Workpiece held by robot hand</p>	

(2) Adjust the focus and aperture.

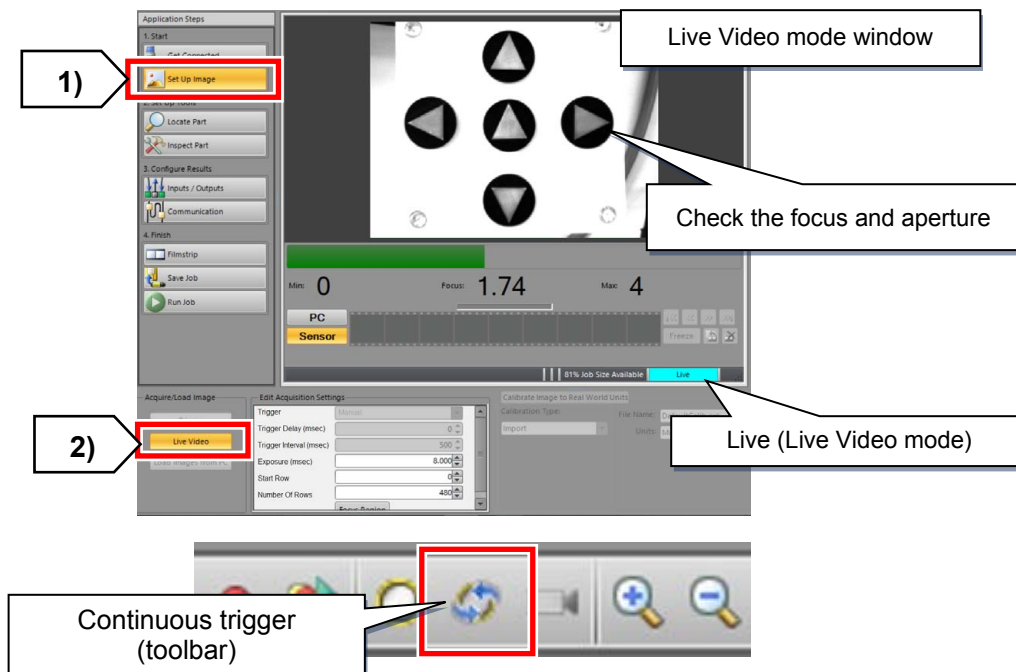
VS80 series (example)



* Lens adjustment methods differ depending on the camera type.
Read the manual thoroughly before using the product.

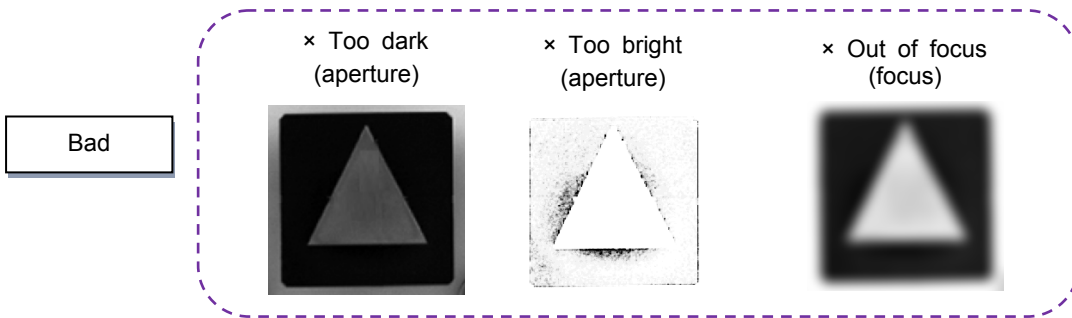
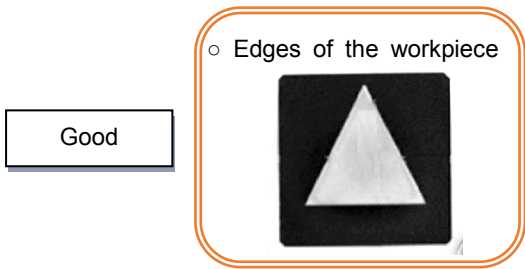
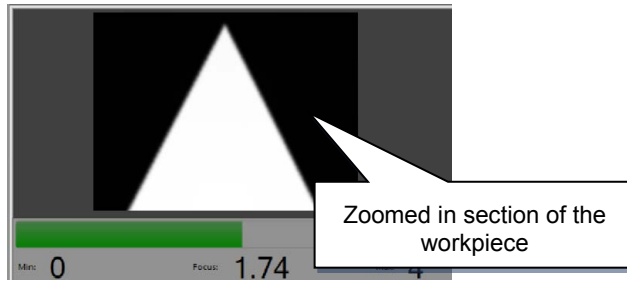
Example of adjusting the lens of fixed downward-facing camera

- 1) Select Set Up Image from the Application Steps window.
- 2) Click Live Video under Acquire/Load Image to switch to Live Video mode. Or select Repeating Trigger mode. (The icon is in the toolbar.)
- 4) Adjust the camera's focus and aperture while looking at EasyBuilder View so that the workpiece is clearly visible.



⚠ CAUTION

To adjust the focus and aperture, zoom in as much as possible in EasyBuilder View to see if the edges of the workpiece are clearly visible.



Chapter 3 - Fixed downward-facing camera

This chapter explains how to adjust the fixed downward-facing camera.

3.1 Principles behind the fixed downward-facing camera

3.1.1 Fundamentals of the fixed downward-facing camera

Fig. 3.1 shows the operations of a typical vision system.

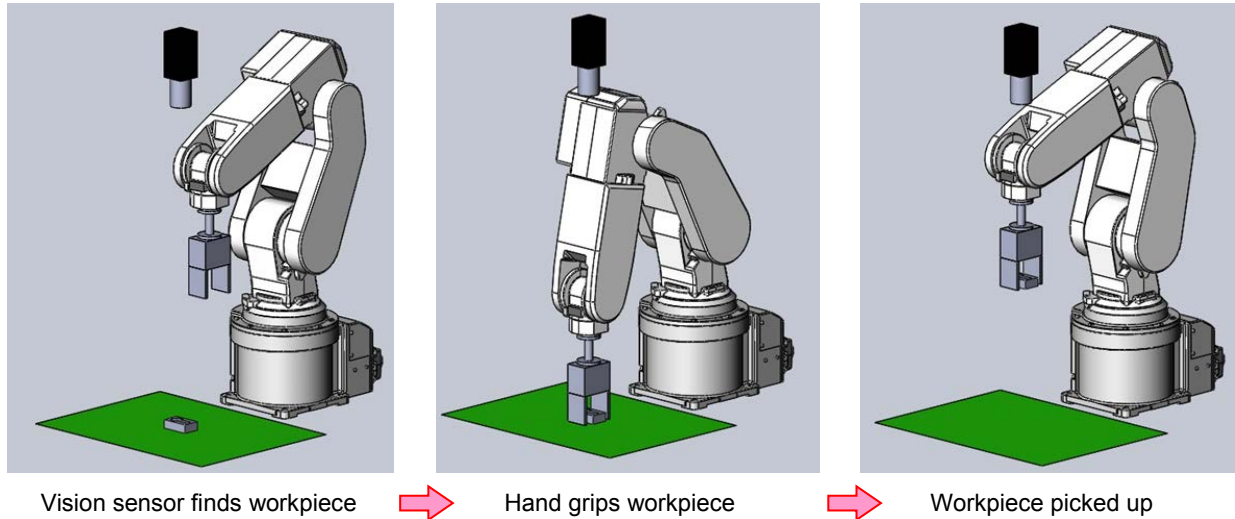


Fig. 3.1: Typical vision system

Fig. 3.1 shows that in a typical vision system, the robot needs a lot of information in advance to complete the operations (imaging/picking). The robot needs the following two pieces of information to perform the operations shown above.

- 1) Where is the position of the workpiece recognized by the vision sensor?
- 2) How should the robot approach the workpiece found by the vision sensor?

However, in regards to question 1, the robot does not know where the vision sensor is installed or where it is looking at.

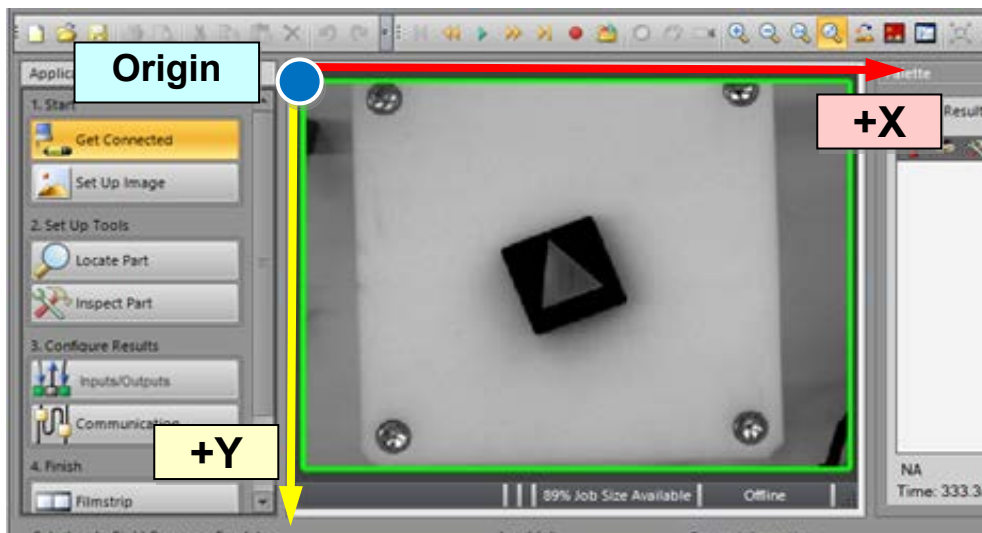


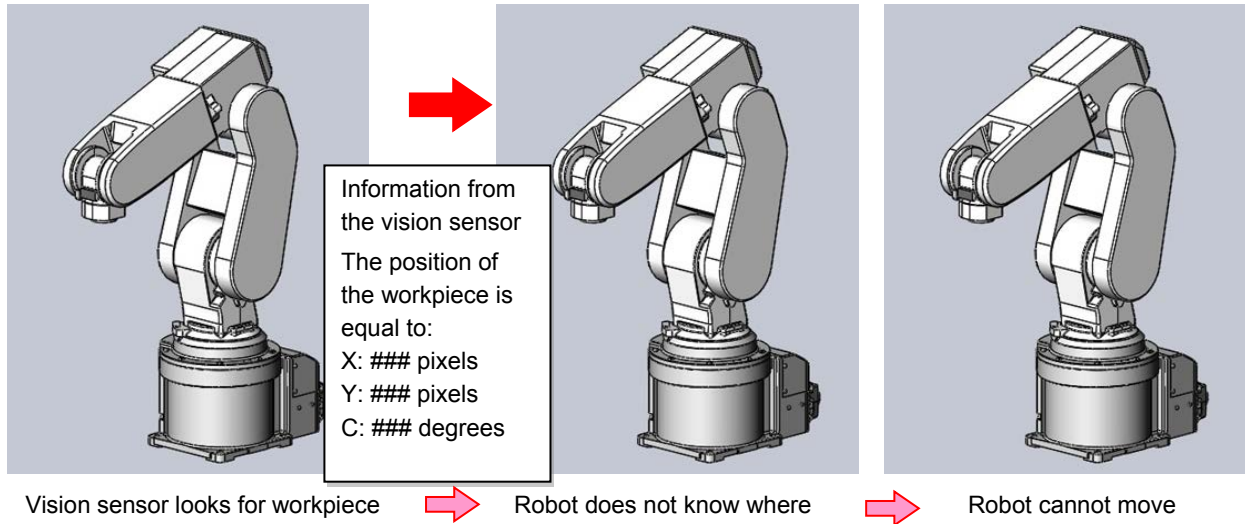
Fig 3.2: EasyBuilder View

The vision sensor uses In-Sight Explorer to identify the position of the workpiece within the FOV. However, the vision sensor does not know where the workpiece is in relation to the robot's origin.

Due to the following factors, the robot will not be able to operate even if the robot and vision sensor try work together*1.

*1 The vision sensor identifies the workpiece and the robot picks it up.

- The robot does not have any information other than its control point.
- The robot can only move using robot coordinates based on its own origin.



The following things need to be done in order to answer questions 1 and 2 on the previous page.

- Convert the position of the workpiece recognized by the vision sensor into robot coordinates.
- Teach the robot the approach position for the master workpiece which has been converted into robot coordinates.

Figure out the relative position of the workpiece in respect to the robot.

* Converting the position of the workpiece recognized by the vision sensor into robot coordinates is called "calibration".

3.1.2 Setting a control point for calibration

Fig. 3.3 shows a typical work environment.

As the robot does not know where the control point is, a pointed object (calibration tool) is put into the robot hand (Fig. 3.4) and the tip of the calibration tool is set as the control point. The teaching pendant will use the coordinates of the tip of the calibration tool to perform calibration accurately.

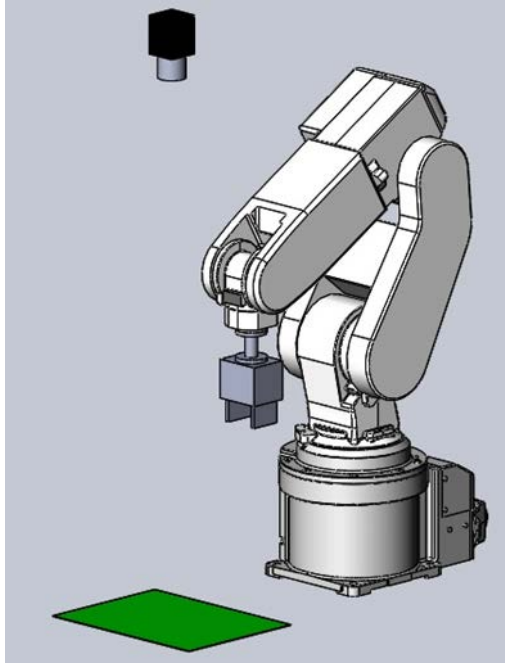


Fig. 3.3: Typical work environment

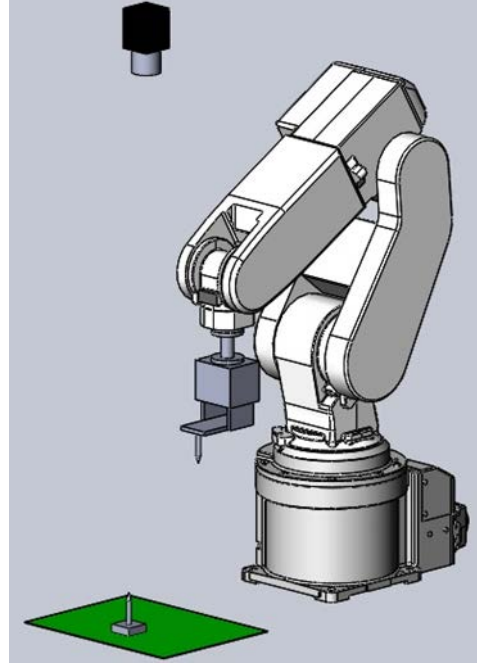


Fig. 3.4: Calibration tool

We will use the program "TLUP" to set a control point that is used for calibration.

(Refer to 3.2.1 Program for setting a control point used for calibration.)

Fig. 3.5 and Fig. 3.6 depict the teaching of P0 and P90 using TLUP.

A pointed object is also placed on the platform. When the robot is taught to rotate the tool's C axis 90° around the tip of the object on the platform, it calculates where that point is within its own tool coordinates. It then sets the coordinates as tool points even though they are only XY components.

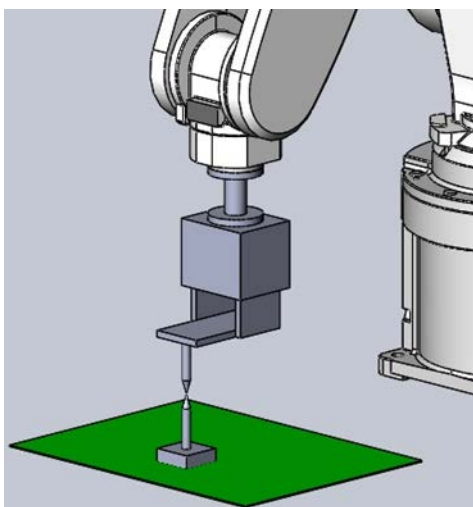


Fig. 3.5: Teaching of P0

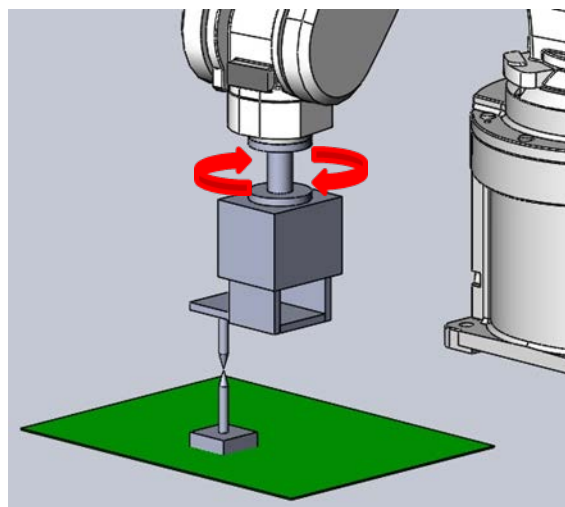


Fig. 3.6: Teaching of P90

Executing TLUP (control point setting program) sets the tip of the calibration tool in the robot hand as the control point. This control point is then used for calibration.

3.1.3 Control point used for calibrating the fixed downward-facing camera

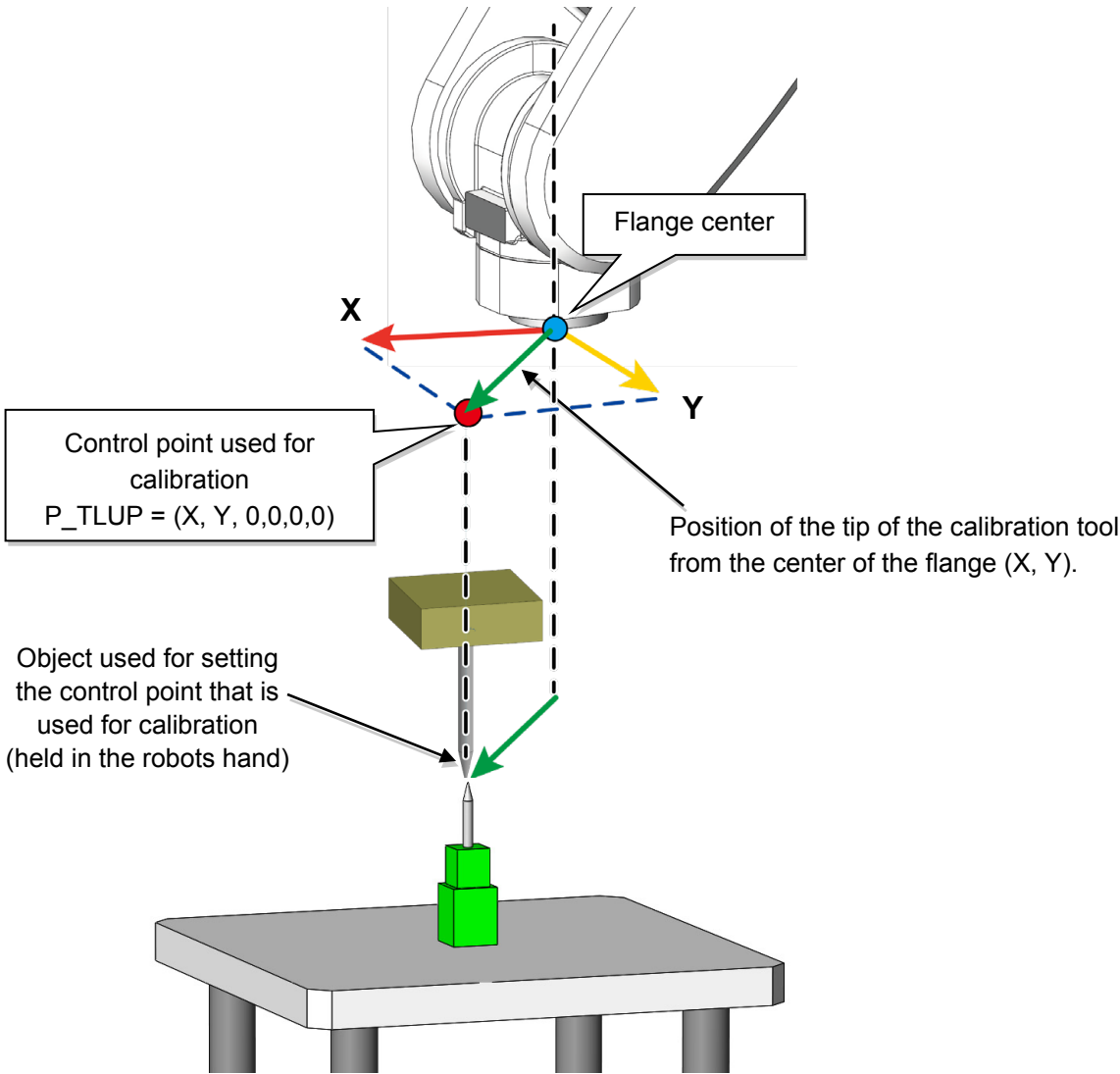


Fig. 3.7: Control point calculated using the fixed downward-facing

3.1.4 Calibration of the fixed downward-facing camera

(1) What is calibration?

Calibration is determining where the origin of vision sensor coordinates will be within the robot coordinates, which direction it is facing and what scale it is.

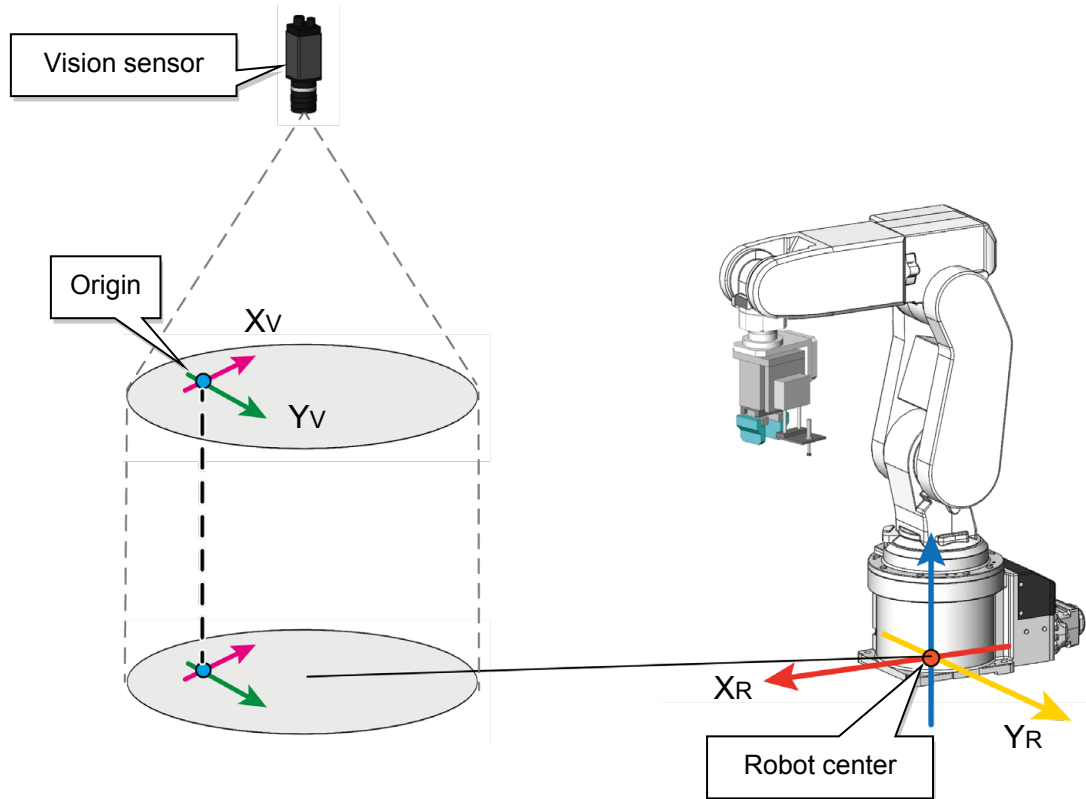


Fig. 3.8: Calibration of the fixed downward-facing camera

(2) Typical calibration method

As shown in Fig. 3.9, the positions of features within the vision sensors FOV are correlated with the positions of the features within the robot coordinate system. By correlating more than three points, we can determine where the vision sensor's origin point is within the robot coordinate system, which direction the camera is facing, and how many millimeters one pixel is equal to in the robot coordinate system. This is the fundamentals of calibration. (Fig. 3.9)

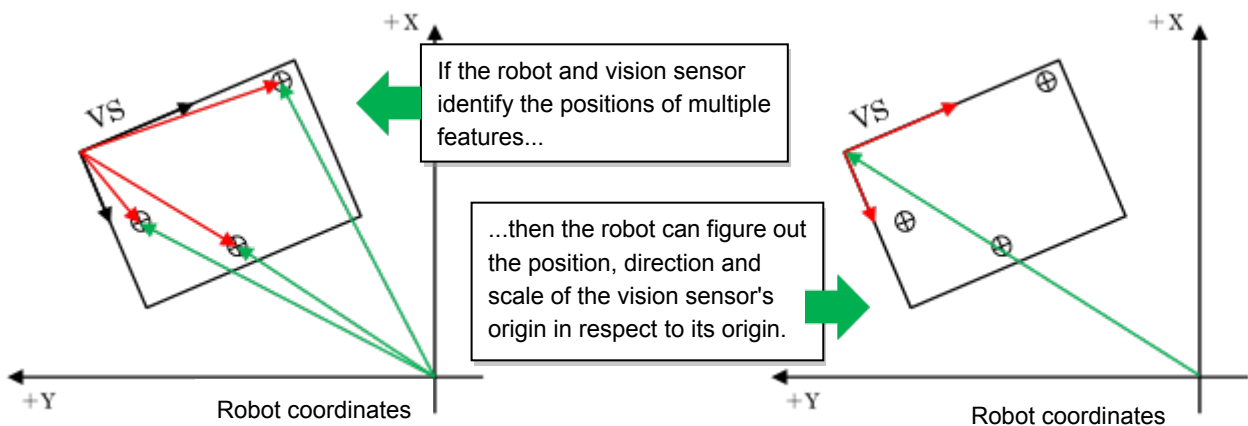


Fig. 3.9: Fundamentals of calibration

(3) N point calibration

In this seminar, we will perform calibration using the method of calibration called "N point calibration". N point calibration is a way of creating calibration data using multiple points which match in both the robot and screen coordinate systems.

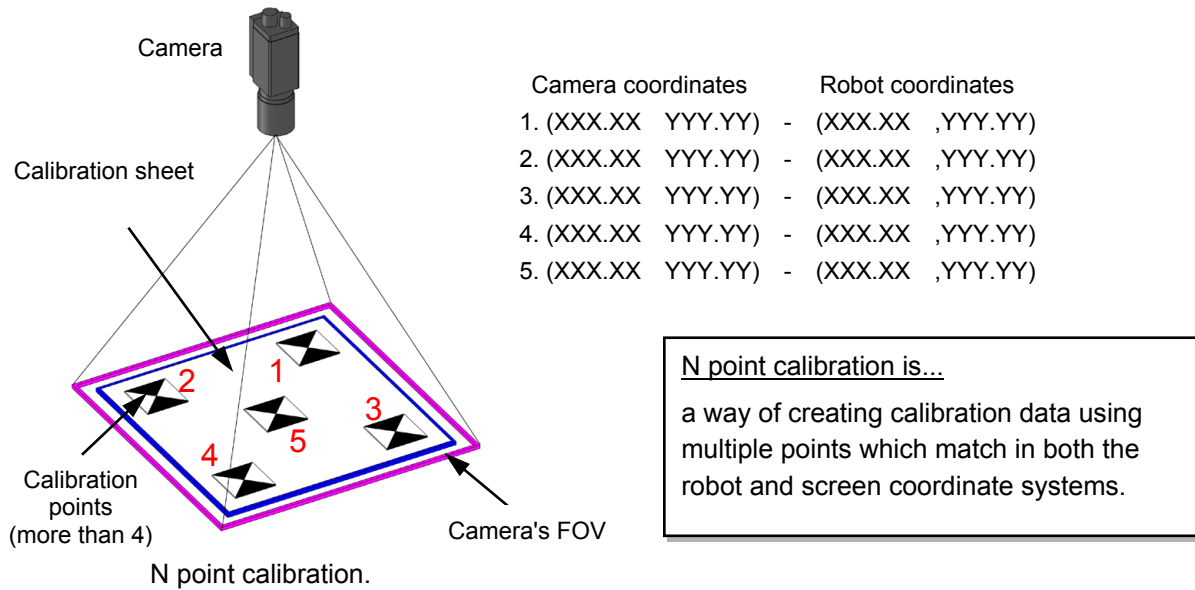


Fig. 3.10: N point calibration

(4) Fixed downward-facing camera: N point calibration method

- 1) Place workpieces with clearly distinguishable features within the camera's FOV at the height at which the workpiece is to be identified.
For example, the apex of the isosceles triangles shown in Fig. 3.11 is used as a feature which is checked against the vision sensor coordinate system.
- 2) When the workpieces (shown in Fig. 3.11A) are identified by the vision sensor, they will be displayed in EasyBuilder View as shown in Fig. 3.11B.
- 3) The coordinates of the feature are as shown in Fig. 3.11C. (Example: 324, 187)

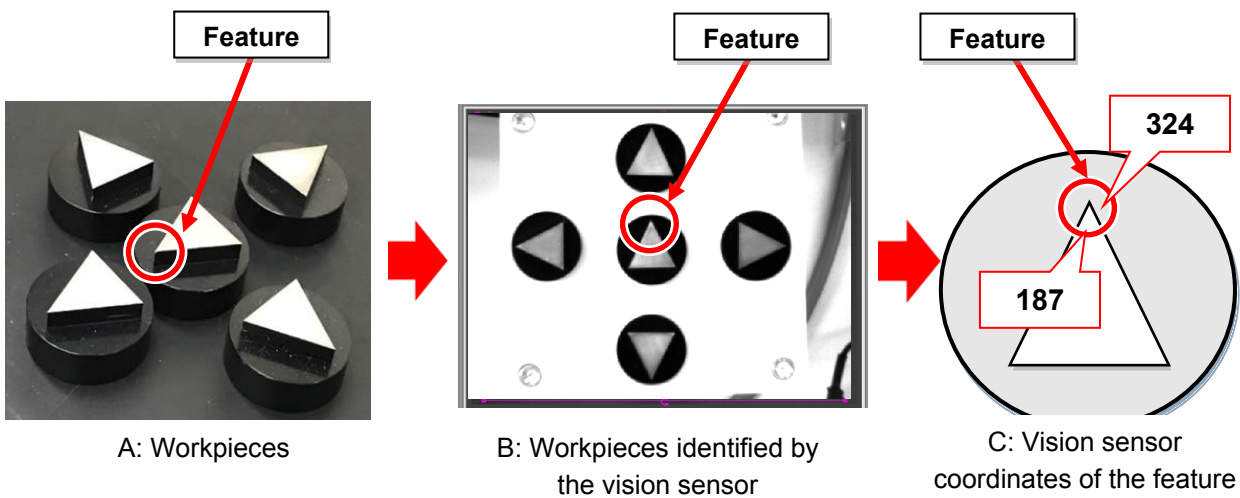


Fig. 3.11: Coordinates of the feature used for N point calibration

In case 1, the positions of the features match in both the robot and vision sensor coordinate systems so calibration can be performed correctly. However in case 2, calibration cannot be performed correctly because the positions of the features do not match. Therefore, the positions of the features within the vision sensor and robot coordinate systems need to match in order for calibration to be performed.

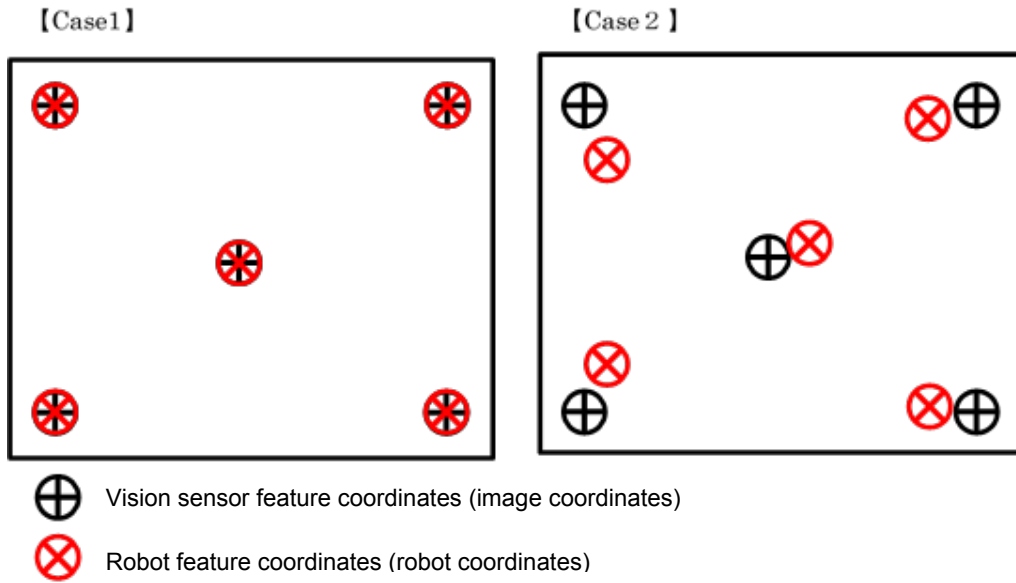


Fig. 3.12: Position of features does not match in robot and vision

In order to check the robot coordinates, move the robot arm to the position of the feature. Then read the robot's current position using the teaching pendant or RT ToolBox3. The current position will then become the position of the feature.

Fig. 3.13 shows a workpiece that has been placed within the camera's FOV and the robot arm being moved to the position of the feature (XY coordinates).

The center of the flange (control point) is aligned with the feature.

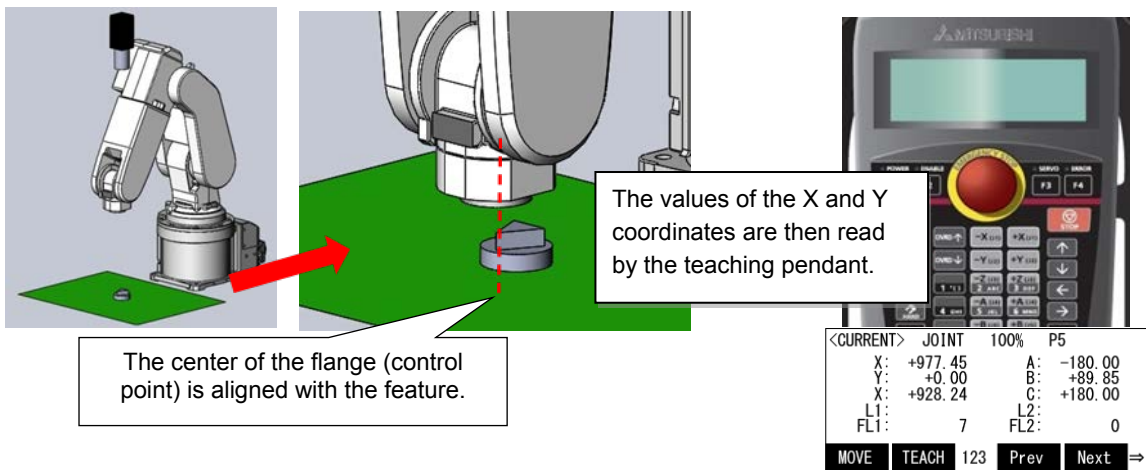


Fig. 3.13: Alignment of the control point with the workpiece feature

It is likely that you will question where the robot control point is during this step. As there is no marking on the center of the flange, it is very difficult to align the center of the flange with the feature on the workpiece while using the robot in JOG mode.

Place the calibration tool (which is pointed) in the robot hand to find out and visualize where the robot control point is. Set the tip of the calibration tool as the control point. (Fig. 3.14: Step 1)

If the control point is set correctly, the coordinates of the tip of the calibration tool will be the same as the position data displayed on the teaching pendant.

Move the tip of the calibration tool to the feature of the workpiece (Fig. 3.14: Step 2), then set the position where the feature and control point meet to the position of the feature in the robot coordinates system (Fig. 3.14: Step 3).

* Do not worry if you cannot align the points perfectly by eye. Align them as best as you possible can.

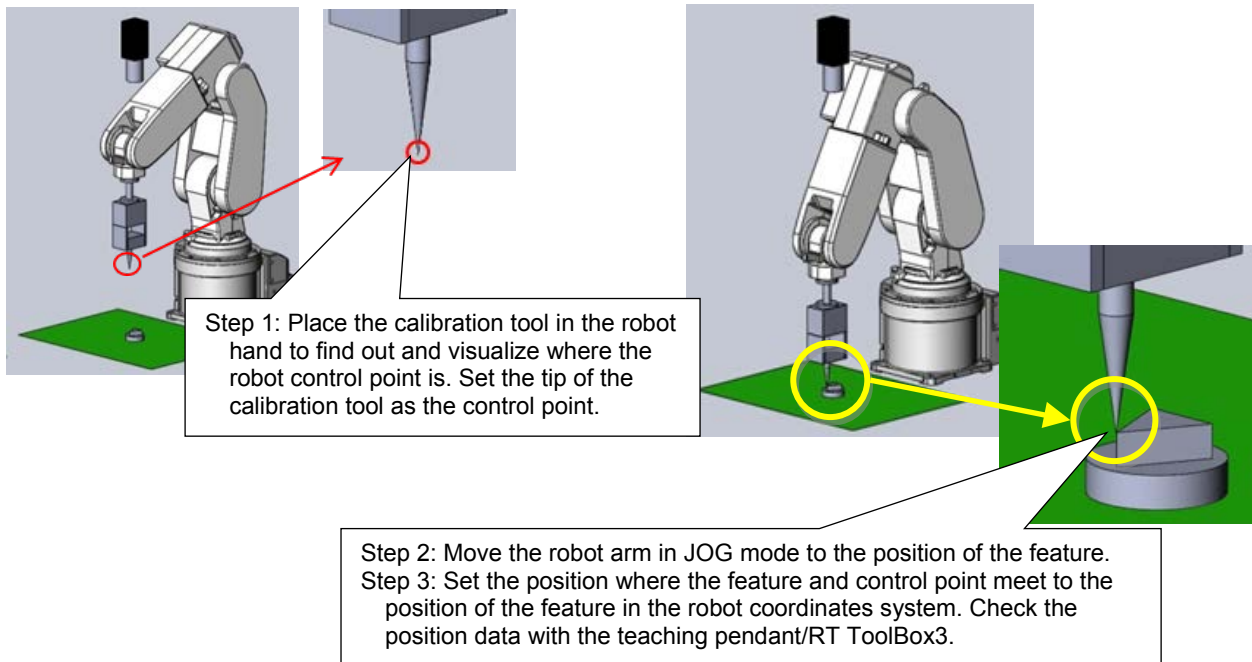


Fig. 3.14: Alignment of the position of the feature with the robot

3.1.5 Fundamentals of the fixed downward-facing camera

Once calibration is complete, the robot can now be told where the workpiece that the vision sensor has identified is.

* At this point in time, the robot knows where the workpiece is, but does not know how to approach it.

Fig. 3.15 shows the relative position of the robot and workpiece at the robot's pick up point.

The position of the workpiece and the workpiece pickup point are only positions with respect to the robot origin.

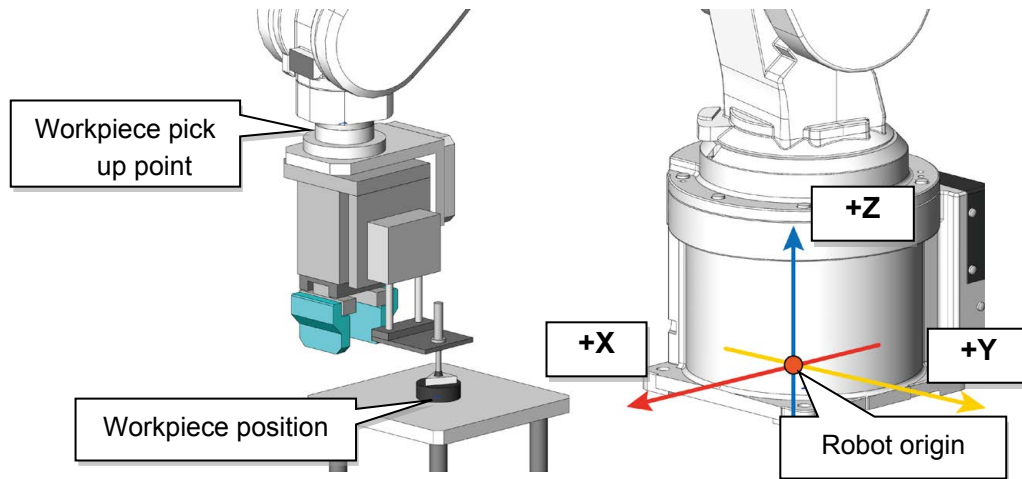


Fig. 3.15: shows the relative position of the robot's pick up point and the position of the workpiece.

Typically, the same picking method is used to pick workpieces of the same type.

The workpiece pick up point, which the robot considers to be the origin of the workpiece, is at a fixed point in space relative to the actual position of the workpiece.

The workpiece pickup point (PTRG) can be thought of as the point that the robot considers to be the position of the workpiece (PVS). That is, at what coordinates in the workpiece coordinate system (A, B, C, X, Y, Z) should the robot pick up the workpiece.

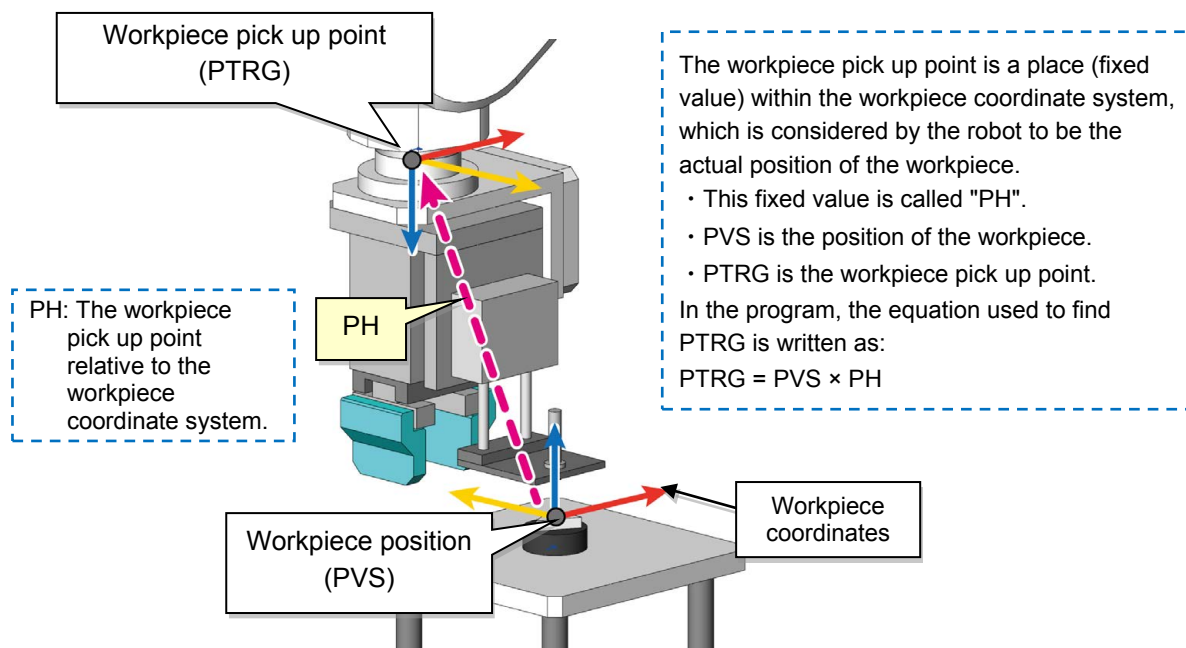


Fig. 3.16: Workpiece pick up point and workpiece position

◇ How to find the distance between the workpiece pick up point and the actual position of the workpiece (PH)

* Miscalibration or improper setting of the control point will result in an incorrect pick up point. Due to this, it is imperative that PH is calculated correctly.

- Workpiece position: The position of the workpiece identified by the vision sensor is registered in the robot coordinate system during calibration.
- Workpiece pick up point: The pick up point taught to the robot. Each point is a position that is measured from the robot origin. (Fig. 3.17)

PH is the distance from position of the workpiece (set as the origin in the workpiece coordinate system) to the workpiece pick up point. PH is expressed as $PH = \text{Inv}(PVS) \times PWK$ in the robot coordinate system.

$$PH = \text{Inv}(PVS) * PWK$$

When working with multiple workpieces of the same type, select one of the workpieces to be the master workpiece and teach the robot PWK and PVS. After PH has been calculated, it is possible for the robot to pick other workpieces in different positions accurately using PH (the calculation of the relationship between the position of the workpiece and the workpiece pick up point). This is only possible if the calculated PH from the master workpiece can be applied to the other pieces.

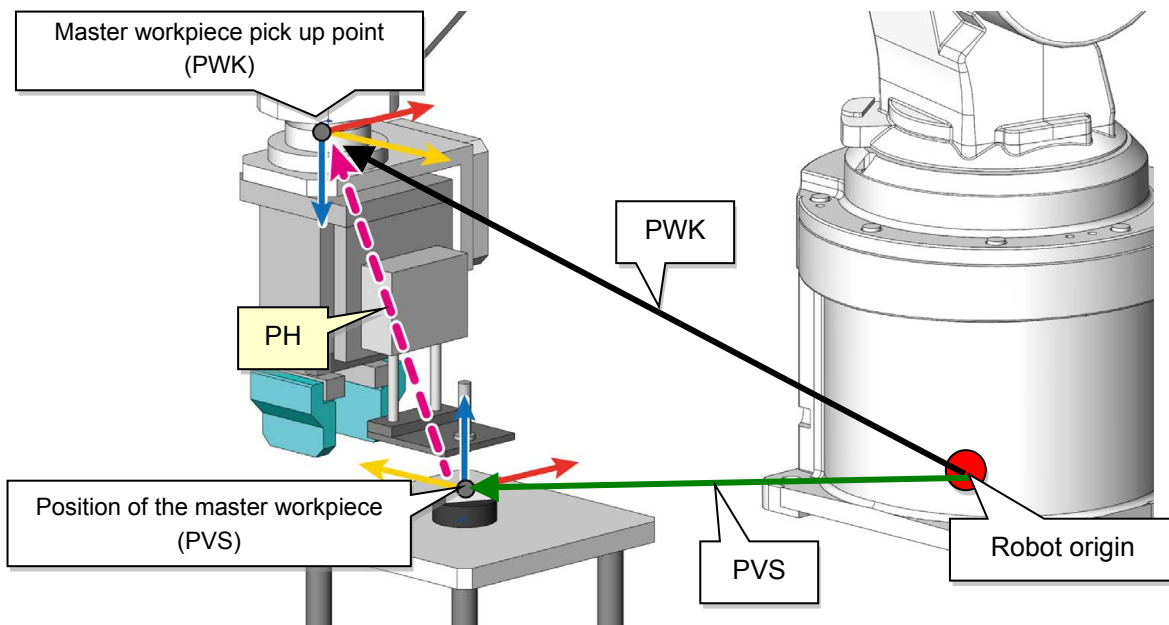


Fig. 3.17: Master workpiece pick up point (PWK) and master workpiece

■ Process of adjusting the fixed downward-facing camera

The following is an overview of the steps required to make automatic operation possible using the fixed downward-facing camera. The Steps 1 to 4 in the following table are the same as those described in Chapter 2, "Communication settings".*1

Therefore, only Steps 5 to 11 will be explained in this chapter.

Step	Description	Remarks
1	Preparing the robot and vision sensor	Chapter 2 - Communication settings and lens adjustment
2	Configuring computer network settings	
3	Connecting the robot and camera (RT ToolBox3) *1	
4	Configuring camera communication settings (In-Sight Explorer)	

*1) The COM port communication settings used in this seminar differ depending on the camera. Refer to "** Cameras used in this seminar and COM port communication settings".



Step	Description	Main tasks
5	Setting the control point used for calibration	Add a user base program. Place the calibration tool in the robot hand. Position the calibration tool. Set the control point used for calibration.
6	Calibration	Position the workpiece used for calibration. Connect a camera and adjust the lens. Create user-defined points. Carry out N point calibration.
7	Create an identification job.	Set the workpiece to be identified and the identification range. Select an identified pattern. Save the job.
8	Setting up the relative position between the robot and the workpiece	Position the master workpiece used for adjustment. Teach the safe position and suction area on the workpiece. Teach the destination position.
9	Checking the robot movement using step operation and automatic operation	-

3.2 Setting a control point used for calibration (Fixed downward-facing camera)

We will set a control point that will be used for robot calibration using the robot and a robot program.

3.2.1 Program for setting a control point used for calibration

The programs used in this chapter are UBP.prg and TLUP.prg.

- UBP.prg: Used for defining user external variables. This program is used for each camera.
- TLUP.prg: Used for setting a control point which is used for calibration. Align the tip of the calibration tool held in the robot's hand with the tip of the pointed object on the platform.

◇ Program UBP.prg

```
1 '-----
2 ' User external variable definition program UBP.prg
3 '-----
4 Def Pos P_TLUP ' Fixed downward-facing camera: Control point data used for calibration
5 Def Pos P_TLLW ' Fixed upward-facing camera: Control point data used for calibration
6 Def Pos P_CMTL ' Hand camera: Camera center position data for calibration
7 '
8 Def Pos P_PH01 ' Fixed downward-facing camera: Coefficient for calculating the handling position
from the position of the identified workpiece
9 Def Pos P_PH02 ' Fixed upward-facing camera: Coefficient for calculating the handling position from
the position of the identified workpiece
10 Def Pos P_PH03 ' Hand camera: Coefficient for calculating the handling position from the position of
the identified workpiece
11 '
12 Def Pos P_WRK01 ' Fixed downward-facing camera: Master workpiece grasp position
13 Def Pos P_WRK02 ' Fixed upward-facing camera: Master workpiece grasp position
14 Def Pos P_WRK03 ' Hand camera: Master workpiece grasp position
15 '
16 Def Pos P_PVS01 ' Fixed downward-facing camera: Master workpiece identification point
17 Def Pos P_PVS02 ' Fixed upward-facing camera: Master workpiece identification point
18 Def Pos P_PVS03 ' Hand camera: Master workpiece identification point
19 '
20 Def Pos P_PHome ' Safe position
21 Def Pos P_CLPos ' Hand camera: Reference point to start calibration
22 Def Pos P_CMH ' Hand camera: Offset from the surface of the identified workpiece to the imaging
point
23 Def Pos P_HVSP1 ' Hand camera: Default imaging point
24 Def Pos P_LVSP1 ' Fixed upward-facing camera: Default imaging point
25 Def Pos P_MPUT1 ' Fixed upward-facing camera: Master workpiece destination position
26 '
27 Def Char C_C01 ' Fixed downward-facing camera: COM name
28 Def Char C_C02 ' Fixed upward-facing camera: COM name
29 Def Char C_C03 ' Hand camera: COM name
30 '
31 Def Char C_J01 ' Fixed downward-facing camera: Job name
32 Def Char C_J02 ' Fixed upward-facing camera: Job name
```

(continued on the next page)

```

33 Def Char C_J03 ' Hand camera: Job name
34 '
35 '----- Dummy -----
36 ' Completion: 100%
37 '
38 '
39 ##### User base program #####
40 '
41 ' NTOOL: Starts the user base program during automatic operation.
42 '
43 '
44 ' Position variable
45 *****prg1010***** Return to origin
46 Def Pos PNow
47 '
48 *****prg2001*****Automatic operation (release workpiece from grasp)
49 Def Pos P_Home
50 Def Pos P_Whaji_vis_b
51 Def Pos P_Whaji_b
52 Def Pos P_ura_b
53 Def Pos P_Wkaiho1_b
54 Def Pos P_Wkaiho2_b
55 Def Pos P_Wkaiho3_b
56 Def Pos P_Wkaiho4_b
57 Def Pos P_Whaji_vis_e
58 Def Pos P_Whaji_e
59 Def Pos P_ura_e
60 Def Pos P_Wkaiho1_e
61 Def Pos P_Wkaiho2_e
62 Def Pos P_Wkaiho3_e
63 Def Pos P_Wkaiho4_e
64 Def Pos P_Wkaiho_vis_b
65 Def Pos P_Wkaiho_vis_e
66 Def Pos P_vision1syutoku
67 '
68 'Numeric variable
69 Def Inte M_Sim
70 Def Inte M_suuti
71 Def Inte M_suuti2
72 '
73 End
P_TLUP=(+0.62,-0.61,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(0,0)
P_TLLW=(+0.41,-91.10,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(0,0)
P_CMTL=(-97.85,-5.06,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(0,0)
P_PH01=(-10.22,+90.64,+253.74,-180.00,+0.00,+178.70,+0.00,+0.00)(7,0)
P_PH02=(-90.33,+3.45,+368.35,-180.00,+0.00,-89.68,+0.00,+0.00)(7,0)
P_PH03=(-0.84,+90.90,+140.90,+0.00,+0.00,-0.01,+0.00,+0.00)(7,0)

```

(continued on the next page)

P_WRK01=(+253.62,+299.22,+253.74,-180.00,+0.00,+180.00,+0.00,+0.00)(7,0)
 P_WRK02=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_WRK03=(+251.47,+297.44,+254.63,+179.96,-0.04,-178.78,+0.00,+0.00)(7,0)
 P_PVS01=(+265.90,+208.84,+0.00,+0.00,+0.00,+1.30,+0.00,+0.00)(7,0)
 P_PVS02=(+325.48,-4.01,+0.00,+0.00,+0.00,-358.49,+0.00,+0.00)(7,0)
 P_PVS03=(+10.33,+4.63,+0.00,+0.00,+0.00,+0.01,+0.00,+0.00)(0,0)

P_PHome=(+187.48,-13.81,+320.67,-180.00,+0.00,+180.00,+0.00,+0.00)(7,0)
 P_CLPos=(+184.47,-178.00,+376.40,-180.00,+0.00,-178.77,+0.00,+0.00)(7,0)
 P_CMH=(+88.38,-90.49,-140.90,+0.00,+0.00,+0.00,+0.00,+0.00)(7,0)
 P_HVSP1=(+165.14,+205.01,+395.53,+179.96,-0.04,-178.78,+0.00,+0.00)(7,0)
 P_LVSP1=(+235.10,-2.95,+368.35,-180.00,+0.00,-88.17,+0.00,+0.00)(7,0)
 P_MPUT1=(+282.76,-87.11,+245.01,-180.00,+0.00,-180.00)(7,0)

PNow=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_Home=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_Whaji_vis_b=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_Whaji_b=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_ura_b=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_Wkaiho1_b=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_Wkaiho2_b=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_Wkaiho3_b=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_Wkaiho4_b=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_Whaji_vis_e=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_Whaji_e=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_ura_e=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_Wkaiho1_e=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_Wkaiho2_e=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_Wkaiho3_e=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_Wkaiho4_e=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_Wkaiho_vis_b=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_Wkaiho_vis_e=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

P_vision1syutoku=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

(End of UBP.prg)

◇ Program TLUP.prg

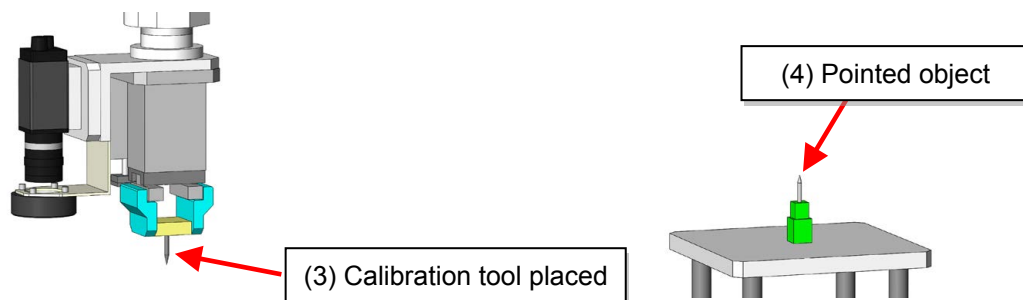
```
1 '-----
2 ' MELFA Sample Program, "XY-tool setting program" TLUP.prg
3 '-----
4 Def Pos P_TLUP ' Fixed downward-facing camera: Control point data used for calibration
5 Loadset 1,1
6 OAdl On
7 Servo On
8 Wait M_Svo=1
9 '---- HAND INIT ----
10 '---- Wait M_Svo=1
11 '---- If M_EHOrg=0 Then                                ' If hand has not returned to origin:
12 '---- EHOrg 1                                         ' returns Hand 1 to origin.
13 '---- Wait M_EHOrg=1                                  ' waits for Hand 1 to return to origin.
14 '---- EndIf
15 '---- EHOpen 1,100,100                                ' Opens Hand 1 (speed = 100%, Force = 20%)
16 '---- Wait M_EHBusy=0                                 ' Checks if operation is complete
17 '-----
18 PTool=P_Tool
19 '---- Align hand. Align the tip of the calibration tool held in the hand with the tip of the object on the
platform.
20 '
21 P0=P_Fbc
22 P91=P0*(+0.00,+0.00,+0.00,+0.00,+0.00,+90.00)
23 Mvs P91
24 ' Moving the hand in only the X and Y axes, use the teaching pendant's XYZ jog mode to align the tip
of the calibration tool held in the robot hand with the tip of the object on the platform.
25 P90=P_Fbc
26 PTL=P_Zero
27 PT=Inv(P90)*P0
28 PTL.X=(PT.X+PT.Y)/2
29 PTL.Y=(-PT.X+PT.Y)/2
30 P_TLUP=PTool*PTL
31 Tool P_TLUP
32 Hit
33 End
P_TLUP=(+0.62,-0.61,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(0,0)
PTool=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(0,0)
P0=(+248.75,+204.28,+293.04,+180.00,+0.00,-137.58,+0.00,+0.00)(7,0)
P91=(+248.75,+204.28,+293.04,+180.00,+0.00,+132.42,+0.00,+0.00)(7,0)
P90=(+249.58,+203.36,+293.04,+180.00,+0.00,+132.42,+0.00,+0.00)(7,0)
PTL=(+0.62,-0.61,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(0,0)
PT=(+1.23,+0.01,+0.00,+0.00,+0.00,-90.00,+0.00,+0.00)(7,0)
PTL2=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(.)
```

(End of TLUP.prg)

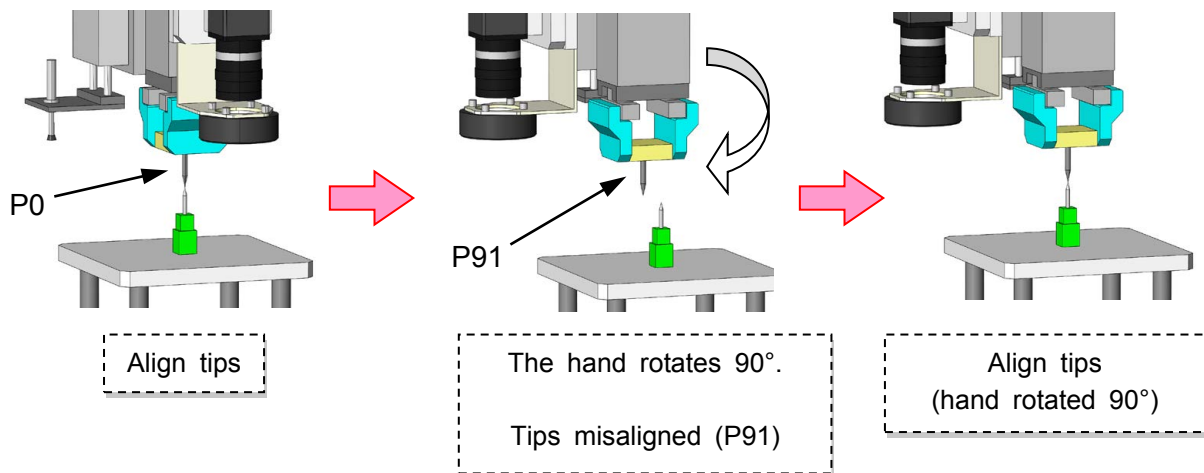
3.2.2 Setting the control point used for calibration

■ Process of setting the control point used for calibration

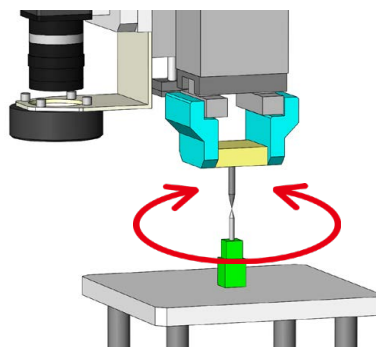
Step	Description
(1)	Add a user base program.
(2)	Execute the program "TLUP".
(3)	Place the calibration tool in the robot hand.
(4)	Position a pointed object on the platform.
(5)	Set a control point which is used for calibration.
(6)	Check the control point used for calibration.



(5) Setting of control point used for calibration



(6) Check the control point used for calibration.

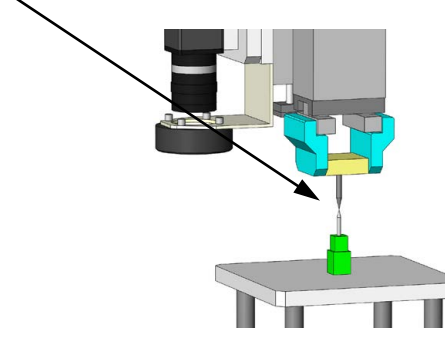
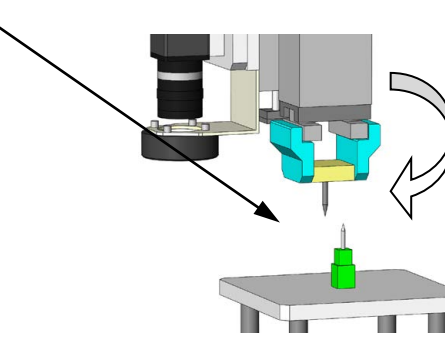
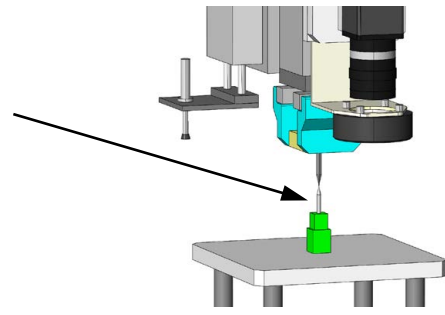


About the program that is used for setting a control point which is used for calibration

The following part of the program is used to calculate a control point which is used for calibration. After setting an object with a sharp point on the platform, this program is run using step operation to find a control point which is used for calibration.

◇ Program TLUP.prg

```
.....  
18 PTool=P_Tool  
19 ' Align hand. Align the tip of the needle in the hand with the  
    tip of the needle on the platform.  
20 '  
21 P0=P_Fbc  
22 P91=P0*(+0.00,+0.00,+0.00,+0.00,+0.00,+90.00)  
23 Mov P91 ' Rotates the hand 90° (tips misaligned).  
24 ' Move the robot using the teaching pendant's XYZ jog  
    mode in the X and Y axes only to realign the tip of the  
    calibration tool held in the robot hand with the tip of the  
    object on the platform.  
25 P90=P_Fbc  
26 PTL=P_Zero  
27 PT=Inv(P90)*P0  
28 PTL.X=(PT.X+PT.Y)/2  
29 PTL.Y=(-PT.X+PT.Y)/2  
30 P_TLUP=PTool*PTL  
31 Tool P_TLUP
```

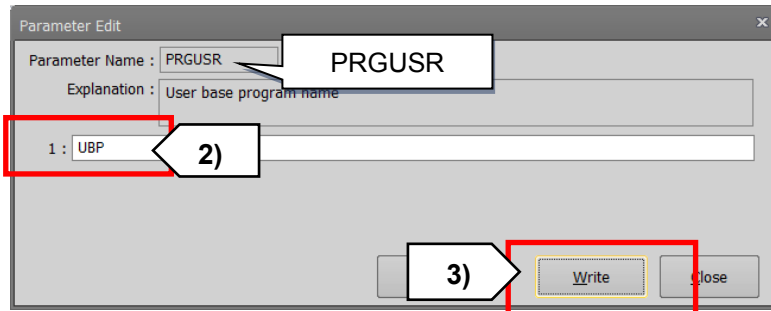


(1) Adding a user base program

(1.1) Add a user base program (UBP).

Add a user base program in RT ToolBox3. This will set up variables which can be used in other programs.

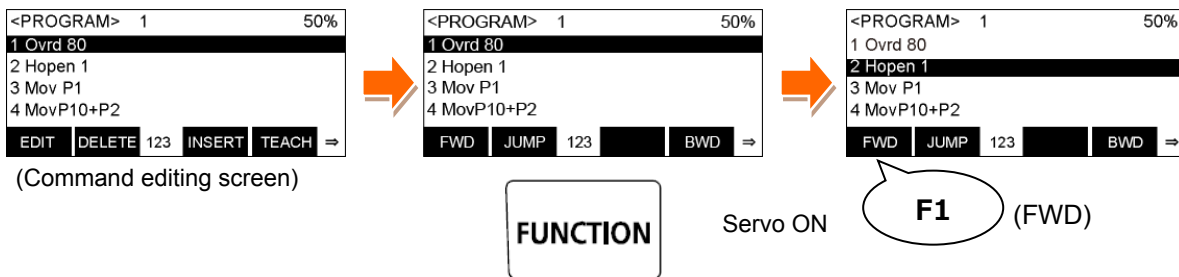
- 1) Go to Online → Parameter → Parameter List in the project tree.
- 2) Read (search for) parameter "PRGUSR". Then double click PRGUSR and enter UBP in the field shown below.
- 3) Click Write to write the parameter to the robot controller.
- 4) Restart the robot controller.



(2) Executing the program "TLUP".

(2.1) Run TLUP.prg.

- 1) Open TLUP.prg.
- 2) Run TLUP.prg up to line 19 using step operation.
 - ◇ When executing line 12, press and hold the F1 key until the hand operation completes. (Initialization of the electric hand)
 - ◇ When executing line 18 "PTool = P_Tool", the current robot control point is assigned to PTool.



(3) Placing the calibration tool in the robot hand

Place the pointed calibration tool in the hand of the robot.

(3.1) Align the hand.

- 1) Press the HAND button on the teaching pendant.
- 2) Press the FUNCTION button on the teaching pendant.
- 3) Press F3 (which corresponds to NORMAL on the screen) to change from the Electric Hand Operation screen to the Hand screen.

Teaching pendant

Electric Hand Operation screen

```

<Electric Hand Operation>      HAND1
SPEED (100)%      ORG ■ SRVO■
FORCE (100)%      RDY ■ BUSY□
POINT No.(0)      INPS □ HOLD■
CURRENT POS( 0.00)mm
HAND2 | HAND3 | 123 | NORMAL | CLOSE =>
    
```

Hand screen

```

<HAND> ±X:HAND6      ±A:HAND3
        ±Y:HAND5      ±B:HAND2
        ±Z:HAND4      ±C:HAND1
        76543210      76543210
OUT-900 □□□□□□□□ IN-900 □□□□□□□□
SAFE | ALIGN | HND | | CLOSE =>
    
```

4) Press the ALIGN button to align the hand

On the screen, select ALIGN by long pressing F2. The hand will align itself while the button is being pressed.

Check that all the ABC components in the Tool jog and XYZ screens are all multiples of 90 then release F2.

```

<HAND> ±X:HAND6      ±A:HAND3
        ±Y:HAND5      ±B:HAND2
        ±Z:HAND4      ±C:HAND1
        76543210      76543210
OUT-900 □□□□□□□□ IN-900 □□□□□□□□
SAFE | ALIGN | HND | | CLOSE =>
    
```

(3.2) Place the calibration tool in the robot hand.

- 1) Press the HAND button on the teaching pendant to select the Electric Hand Operation screen.
- 2) Place the calibration tool in the center of the robot hand. Use F1 and F2 (which correspond to HOPEN and HCLOSE on the screen) to open and close the hand.
 - * Move the robot hand in the Z axis so that the tip of the calibration tool is parallel with the tip of the pointed object on the platform. (Refer to "B" in the images of Step 4.)

Teaching pendant

Electric Hand Operation screen

```

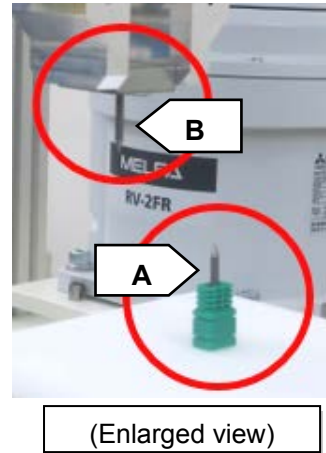
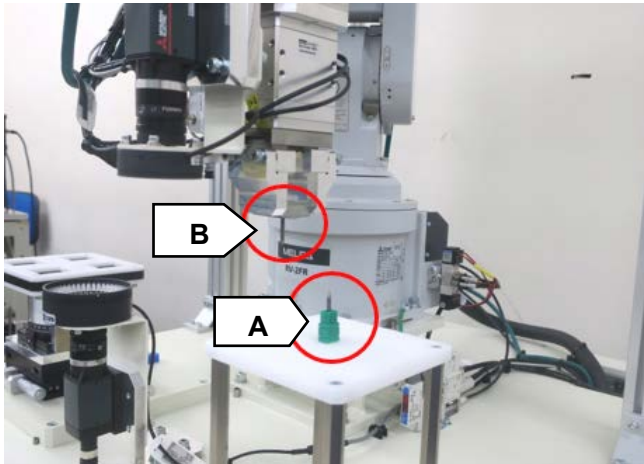
<Electric Hand Operation>      HAND1
SPEED (100)%      ORG ■ SRVO■
FORCE (100)%      RDY ■ BUSY□
POINT No.(1)      INPS ■ HOLD□
CURRENT POS( 0.00)mm
HOPEN | HCLOSE | 123 | HMOV | HORG =>
    
```

(4) Positioning the pointed object on the platform

A pointed object is used to visualize the current position of the robot.

(4.1) Place the object on the platform.

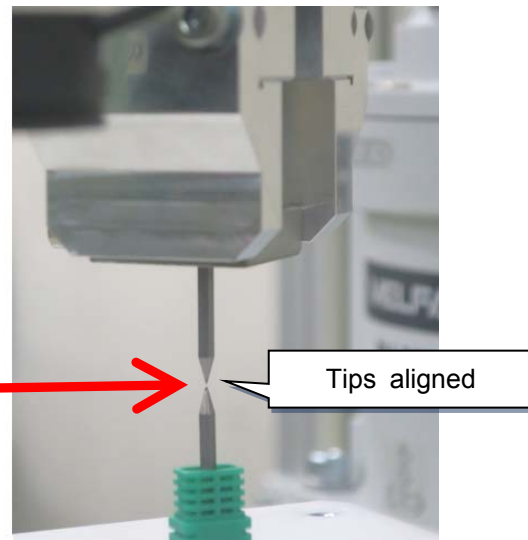
1) Place the pointed object on the platform parallel to the surface the robot is on.



(5) Setting a control point used for calibration

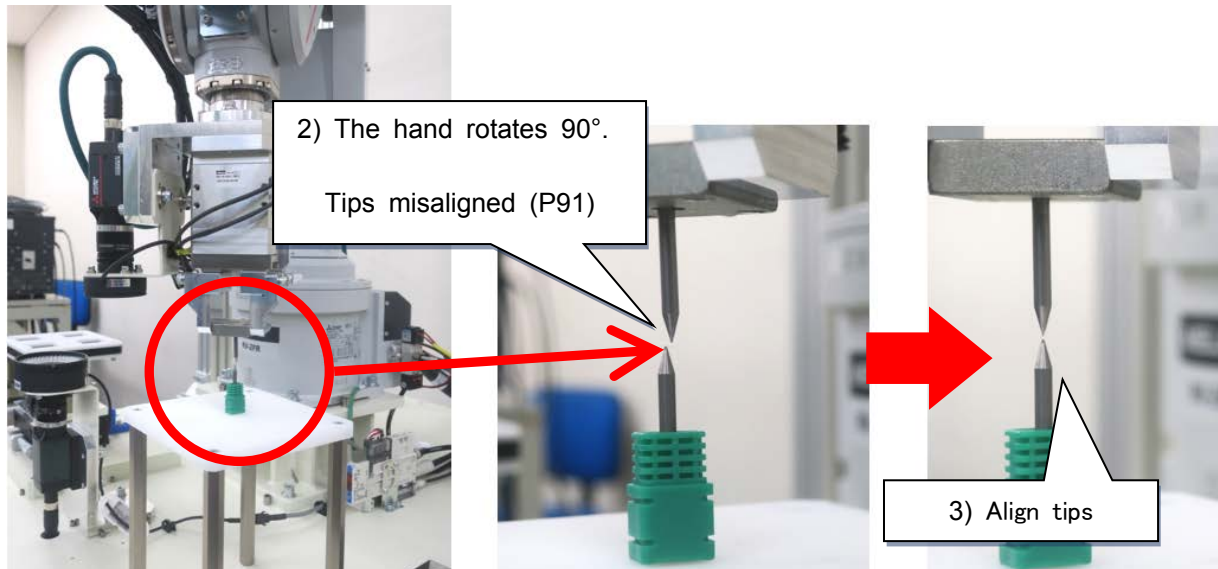
(5.1) Align the calibration tool in the robot hand with the sharp object on the platform.

1) With the teaching pendant in either XYZ jog mode or Tool jog mode, align the calibration tool in the robot hand with the sharp object placed on the platform in Step 4.1.



(5.2) Run TLUP.prg.

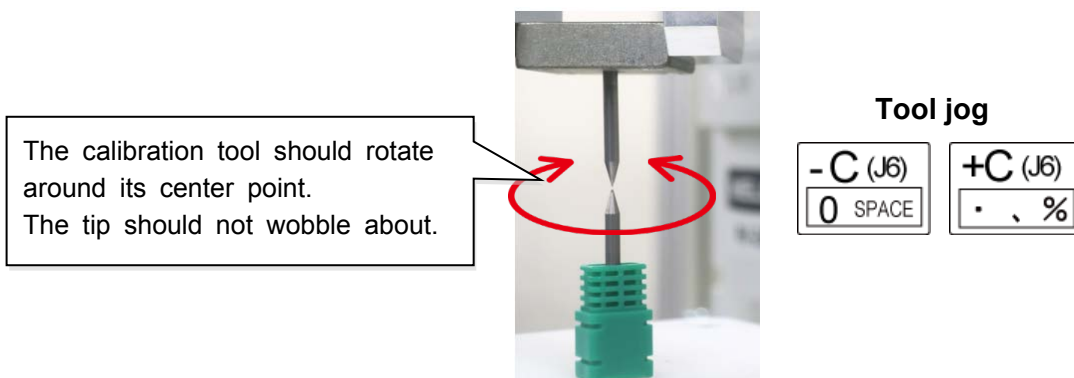
- 1) Run TLUP.prg up to line 23 using step operation.
- 2) When executing "Mvs P91" in line 23, the hand will rotate 90° and the tips of both the calibration tool and the pointed object on the platform will no longer be aligned.
- 3) With the teaching pendant in JOG mode move the robot in the XY axes to realign the tips of the calibration tool and the pointed object on the platform.



(6) Checking the control point used for calibration

(6.1) Check the robot movement.

- 1) Run TLUP.prg up to line 32 (Hlt) using step operation.
 - ◇ When executing lines 25 to 30, the control point data in P_TLUP which is used for calibration will be set.
 - 2) After those lines have been executed, rotate the C axis in Tool jog mode to make sure that the robot hand is revolving around the center of the calibration tool.
- * If the tip of the calibration tool is not rotating perfectly around its center point, repeat the steps starting from "(5) Setting a control point used for calibration".



⚠ CAUTION The calibration tool is required for calibration so do not remove it from the robot hand. Make sure not to accidentally knock, dislodge, or interfere with the calibration tool.

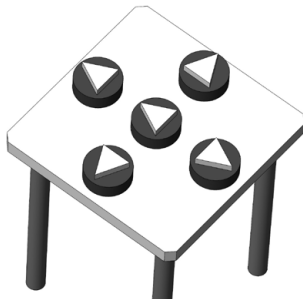
3.3 Calibration (Fixed downward-facing camera)

3.3.1 Calibration method

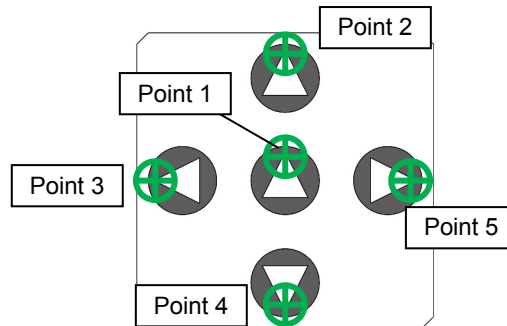
■ Calibration process

Step	Description
(1)	Create a new job.
(2)	Position calibration workpieces and connect the camera.
(3)	Adjust the lens.
(4)	Set user-defined points.
(5)	Create N points.
(6)	Carry out N point calibration.

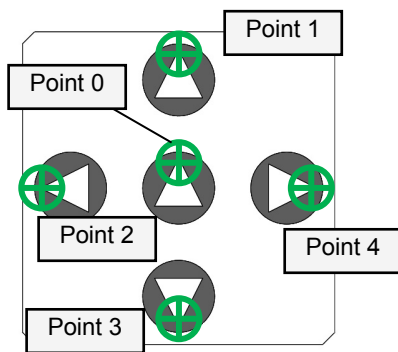
(2) Position calibration workpieces



(4) Set user-defined points.



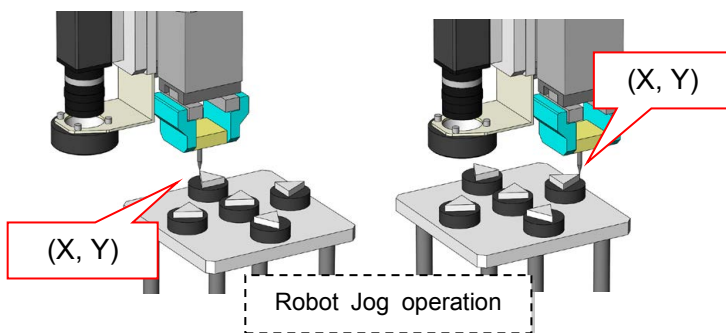
(5) Create N points.



Camera coordinates
 Point 0 (x, y)
 Point 1 (x, y)
 Point 2 (x, y)
 Point 3 (x, y)
 Point 4 (x, y)

Points are named in the order they are selected.
 Point 0 (first selected point),
 point 1 (second selected point).....

(6) Carry out N point calibration.



Enter the teaching pendant value (robot coordinates) into x and y.
 (World X, world Y)

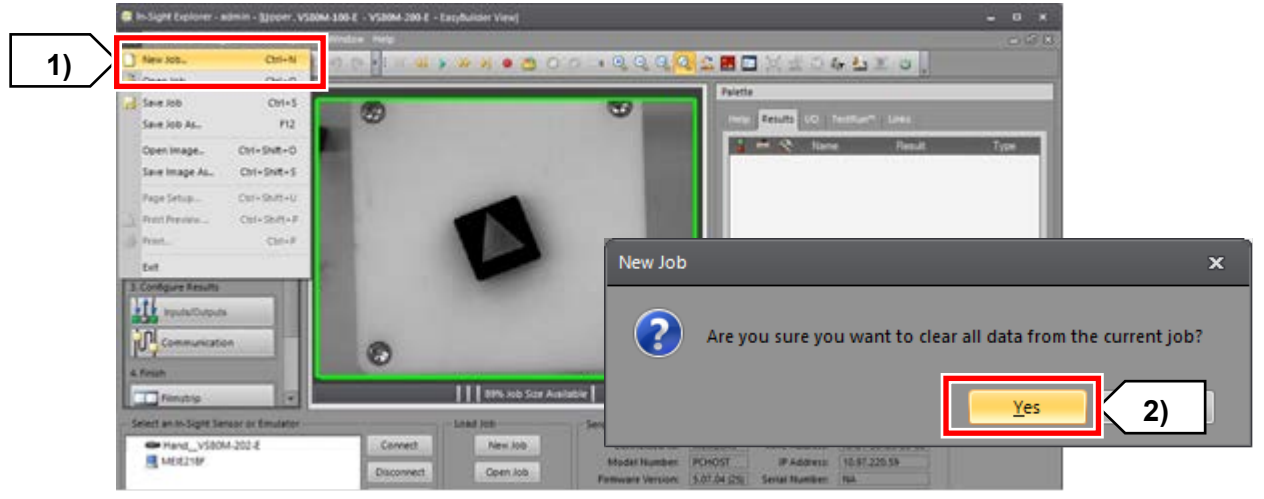
Camera coordinates →
 Robot coordinates
 Point 0 (x, y) → Point 0 (X, Y)
 Point 1 (x, y) → Point 1 (X, Y)
 Point 2 (x, y) → Point 2 (X, Y)
 Point 3 (x, y) → Point 3 (X, Y)
 Point 4 (x, y) → Point 4 (X, Y)

(1) Creating a new job

(1.1) Create a new job.

1) Go to File then New Job.

2) The dialog "Are you sure you want to clear all data from the current job?" will appear. Click Yes.

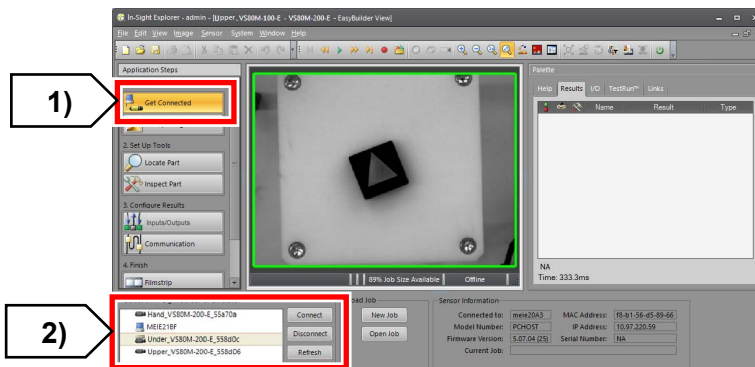


(2) Positioning calibration workpieces and connecting the camera

(2.1) Position the calibration workpieces and connect the camera.

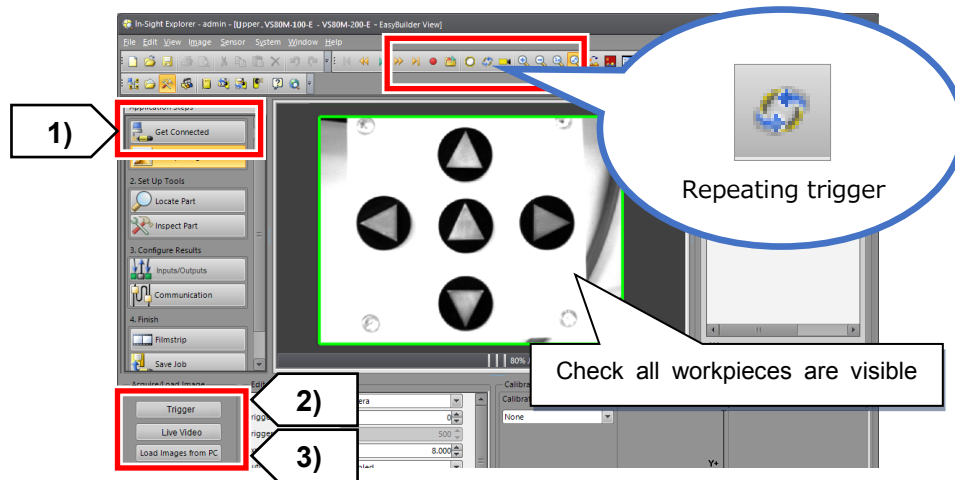
1) Position the calibration workpieces.

2) Select Get Connected from the Application Steps window. Select the fixed downward-facing camera and click Connect.



(2.2) Acquire an image.

- 1) Select Set Up Image from the Application Steps window.
- 2) Select Live Video from Acquire/Load Image. Or, click the Repeating trigger button on the tool bar.
- 3) Ensure that all the calibration workpieces are visible in EasyBuilder View.



(3) Adjusting the lens

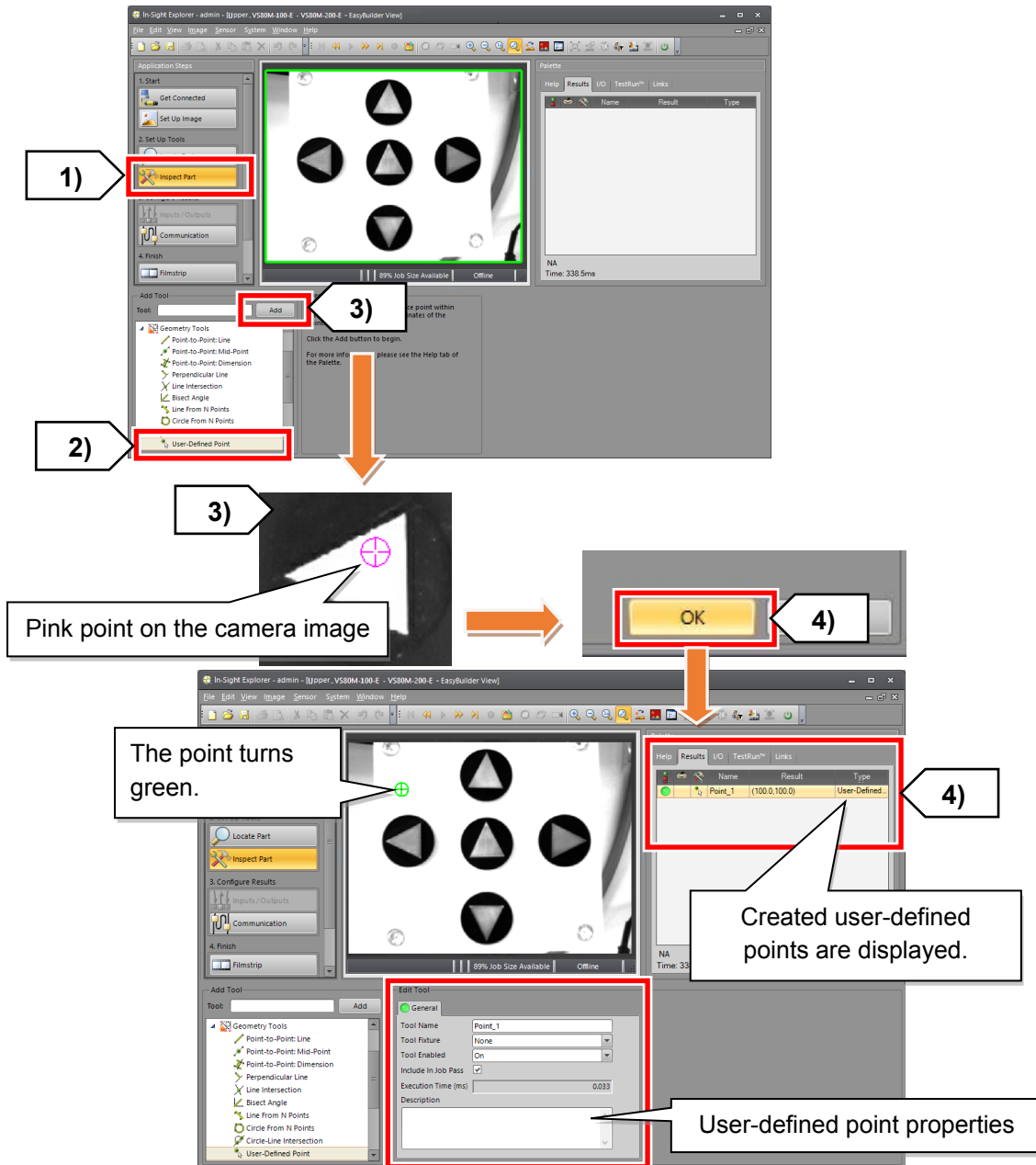
(3.1) Adjust the focus and aperture of the camera.

- 1) Select Live Video from Acquire/Load Image, or click the Repeating trigger button on the tool bar.
- 2) Adjust the focus and aperture to get a clear image of the workpiece suction point.
Refer to 2.5 Adjusting the lens (focus and aperture) for information on how to adjust the lens.

(4) Setting user-defined points

(4.1) Create user-defined points.

- 1) Select Inspect Part from the Application Steps window.
- 2) Select User-Defined Point from Geometry Tools in the bottom left Add Tool window.
- 3) Clicking Add will create a pink point on the camera image.
- 4) Clicking OK will add the user-defined point (Point_1) to the Palette on the right. The pink point will turn green. (Relocate the point later.)

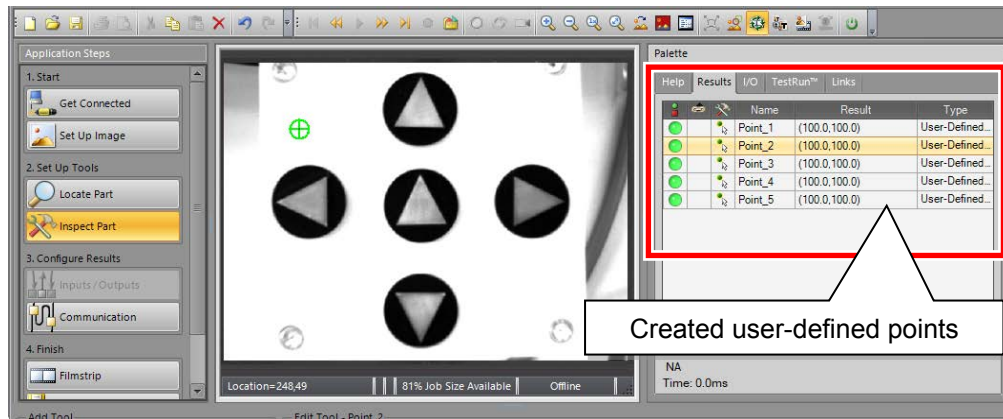


(4.2) Create an equal number of user-defined points and calibration marks.

Create an equal number of user-defined points and calibration marks. Five points will be used in this seminar.

1) Create the five points using steps 3 and 4 in section 4.1.

The created user-defined points will highlight in green on the camera image and added to the Palette. (In the Palette, the user-defined points will be listed in the order they were created.)



(4.3) Align the user-defined points with the workpieces.

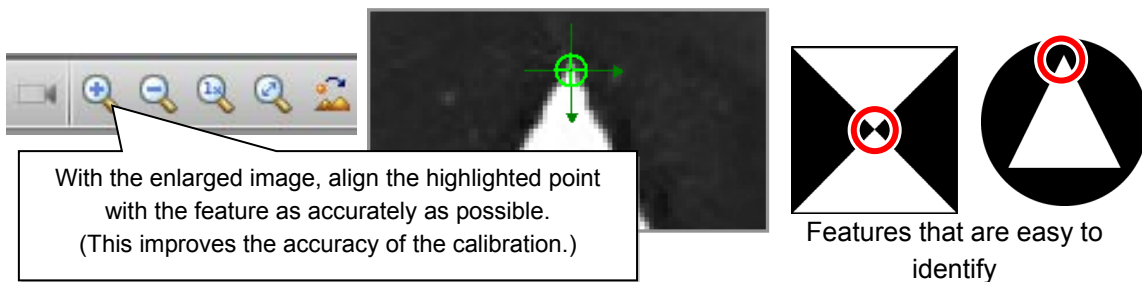
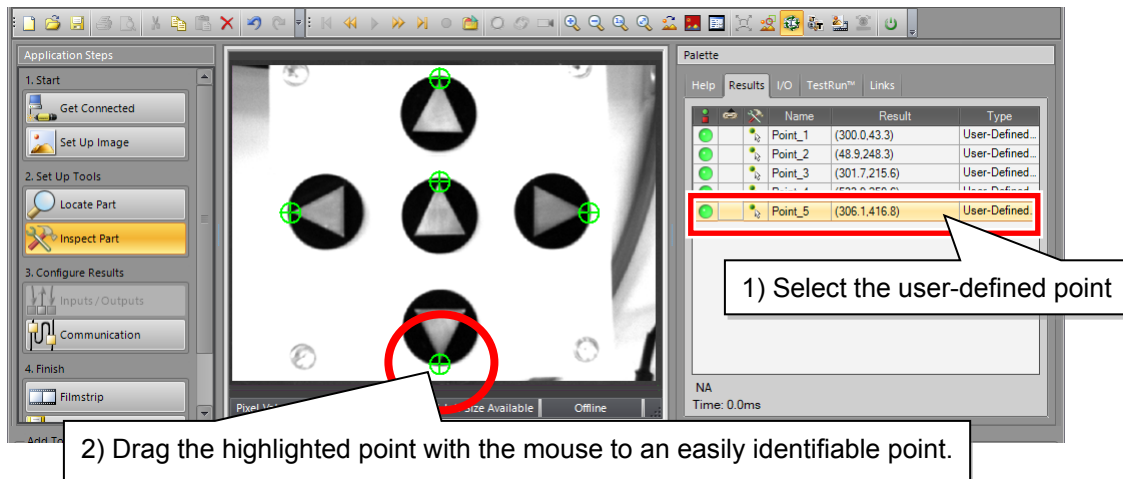
Place the five points in easily recognizable positions on the workpieces. (at the apex of each triangle)

* It is helpful to remember where each point (1 to 5) has been placed.

1) Select Point_1 from the Palette. The corresponding point will highlight in green on the camera image.

2) Drag the highlighted point with the mouse to an easily identifiable point on a workpiece (the apex of the triangle or a point where two lines intersect). * With the image enlarged, align the highlighted point with the feature as accurately as possible.

3) Repeat the same process for points 2 to 4.



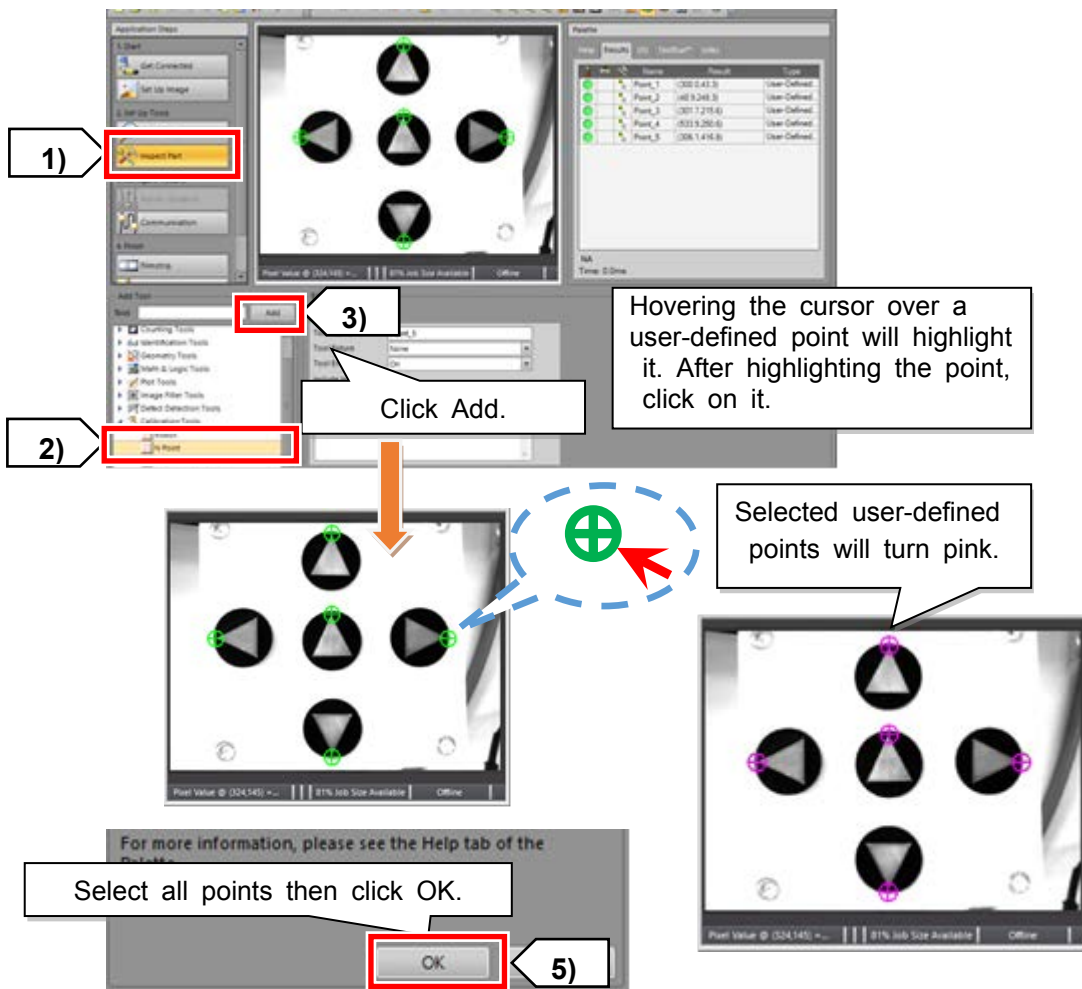
(5) Create N points



In the process of creating N points, we will move the robot arm to each user-defined point.
However, calibration will not complete if the user-defined points and the robot

(5.1) Create N points.

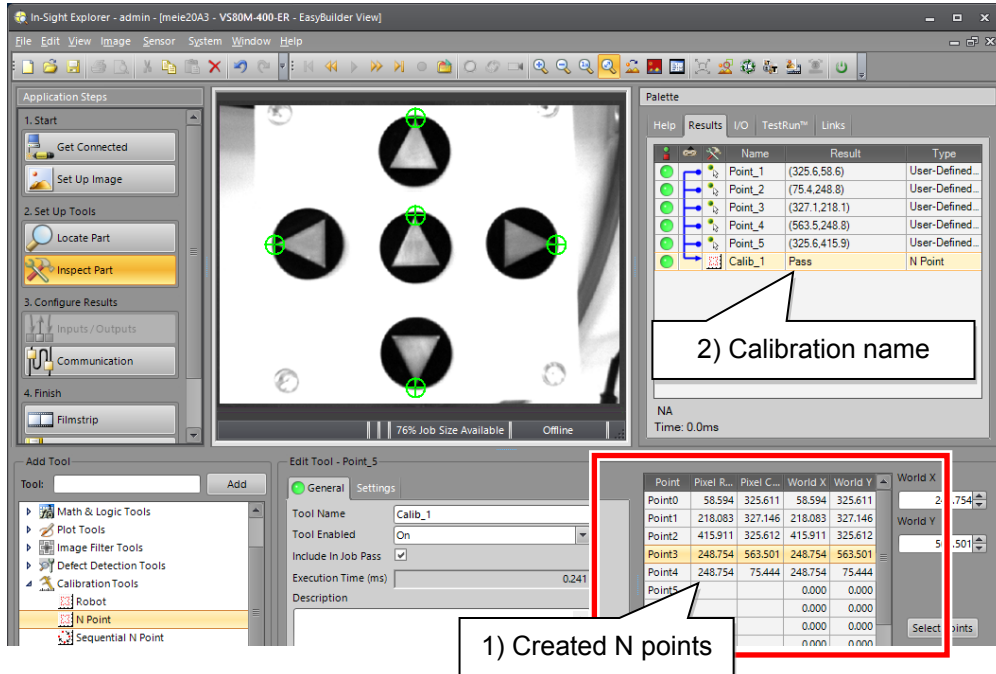
- 1) Select Inspect Part from the Application Steps window.
- 2) Select N Point from Calibration Tools in the bottom left Add Tool window.
- 3) Click Add. All the user-defined points created in Step 4 will highlight in green.
- 4) Select all of the highlighted user-defined points. (Refer to the following image for how to select user-defined points.)
Selected user-defined points will turn pink.
* It is helpful to remember what order the user-defined points have been selected.
- 5) After all the user-defined points have been selected, click OK.



Note: Clicking the user-defined points will create N points.
If anything other than a user-defined points is clicked, or a user-defined point does not turn pink when selected, click Cancel and repeat the process from Step 3 in section 5.1.

(5.2) N point calibration data will be created.

- 1) The selected N points will be listed in the bottom right of the window.
- 2) The calibration name will be displayed in the Palette to the right of the image. (Example: Calib_1)
N points are named in the order in which the points were selected in Step 5.1. For example, the point selected first is "Point 0", the point selected second is "Point 1", and so on.



Note:

If the number of listed N points differs from the number of N points you want to create...

Something else other than the user-defined points may have been selected in Step 5.1. (Click user-defined points to create N points.)

In this case, recreate N points in the following steps.

- 3) Select the calibration name from Palette, then right-click and select Delete to delete the data.
- 4) Create N points by repeating the process from Step 3 in section 5.1 onwards.

The number of listed N points is 6, but the number of N points you want to create is 5.
↓
In this case, delete the calibration data and repeat the steps in section 5.1.

Point	Pixel R...	Pixel C...	World X	World Y	World X
Point0	11.525	25.254	11.525	25.254	11.525
Point1	58.594	325.611	58.594	325.611	World Y
Point2	218.083	327.146	218.083	327.146	25.254
Point3	415.911	325.612	415.911	325.612	
Point4	248.754	563.501	248.754	563.501	
Point5	248.754	75.444	248.754	75.444	
Point6			0.000	0.000	
Point7			0.000	0.000	

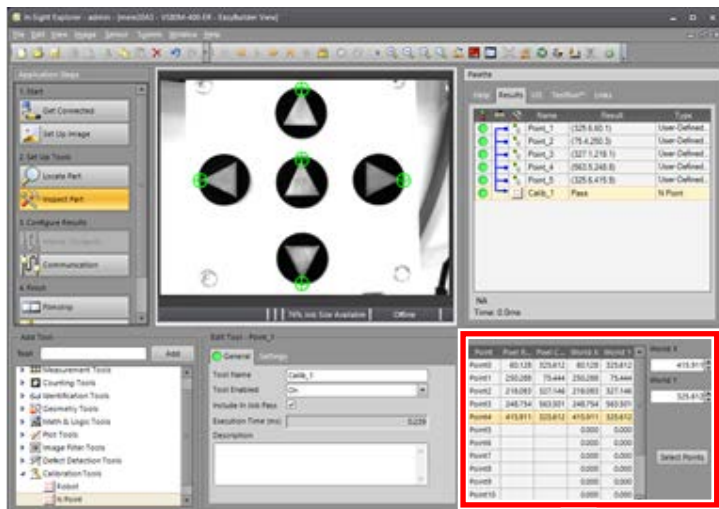
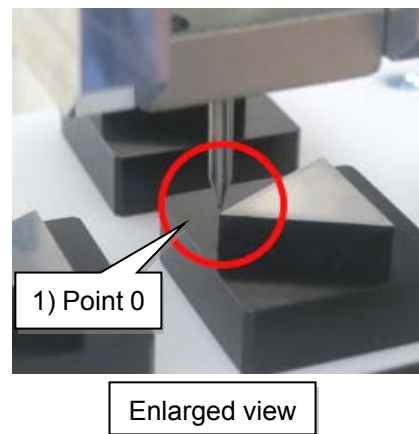
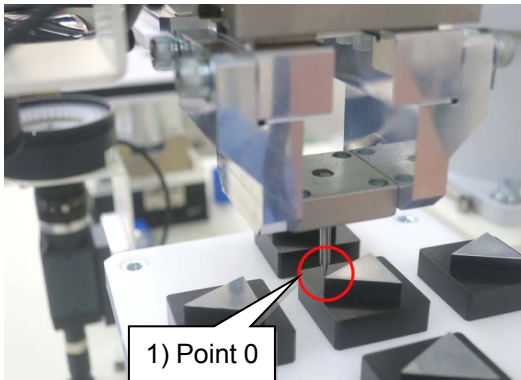
(6) Carry out N point calibration.

In this step we will calibrate the 5 points.

The vision sensor feature points (image coordinates) will be associated with the corresponding robot feature points (robot coordinates). (Refer to 3.1.4 Calibration of the fixed downward-facing camera.)

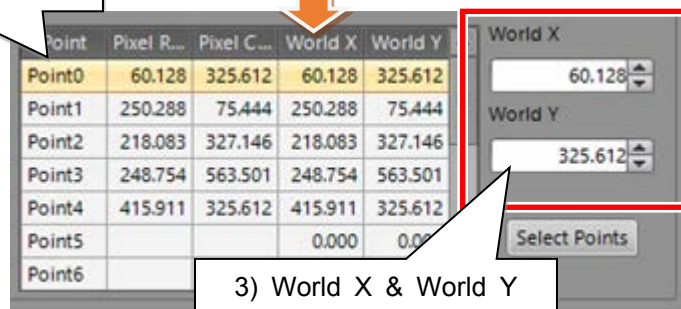
(6.1) Use the teaching pendant to acquire position data of the calibration points.

- 1) Move the calibration tool in the robot hand to the position that corresponds with the on-screen position of Point 0.
- 2) Once the robot has moved to the position of Point 0, move the selector in the bottom right of In-Sight Explorer to Point 0.
- 3) Enter the X and Y robot coordinates displayed on the teaching pendant in the World X and World Y fields.
- 4) Repeat the process for points 1 to 4. (Five points in total)



2) Select a point number.

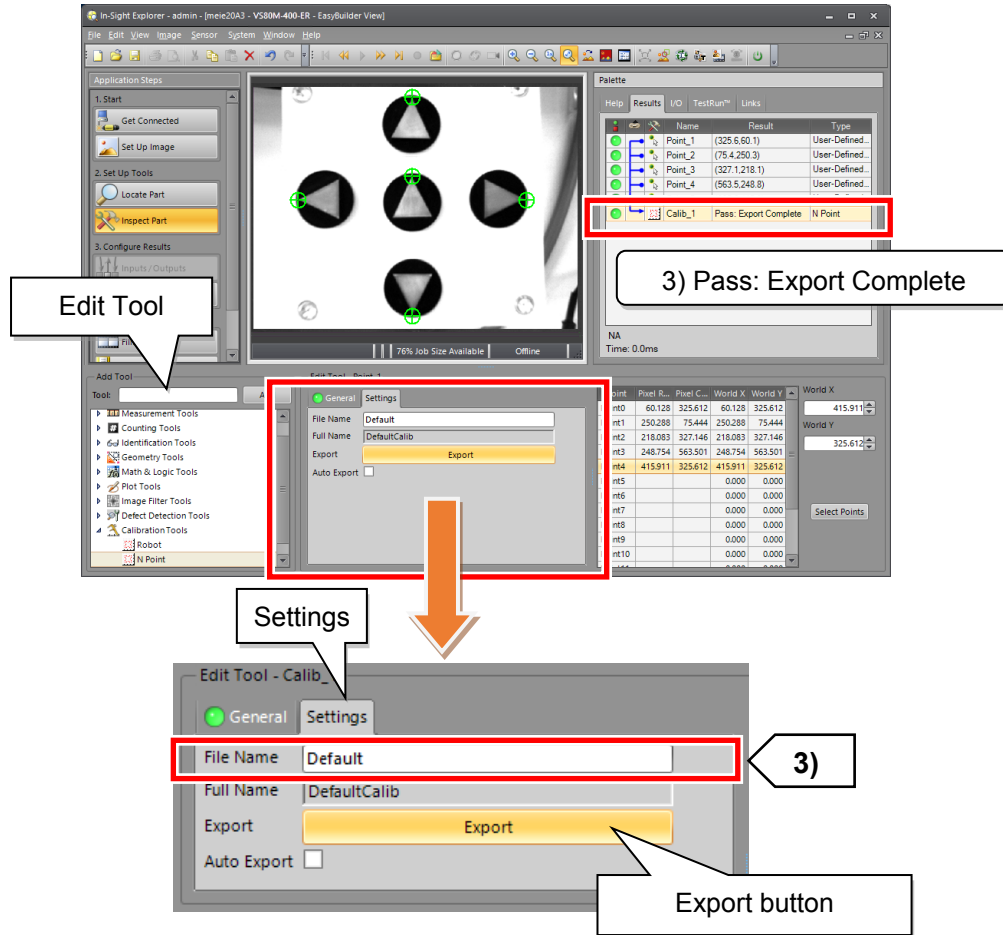
Points 0 to 4
Five points in total



Clicking a point number will display the corresponding values in the World X and World Y fields. You can overwrite the values.

(6.2) Export calibration data.

- 1) Select the Settings tab under Edit Tool in the bottom center of In-Sight Explorer.
- 2) If the Repeating trigger is enabled, disable it.
- 3) Enter a file name and press Export.
- 4) Pass: Export Complete will be shown in the Results tab of the Palette if the data has been exported successfully.



Additional

Reason for exporting data _____

This vision sensor operates using vision programs known as "jobs".

Calibration usually has to be performed with each Job. However, exporting the data saves the formula for converting the camera pixel data into robot coordinates.

This means that calibration does not have to be performed again if this conversion formula is imported when creating a different job.

Additional information

If the camera cannot read/write data

If calibration data cannot be exported, the job cannot be read, or a network error occurs when data is accessed from the camera, change the settings of Windows Firewall.

- (1) Go to Control Panel → System and Security → Allow a Program through Windows Firewall and click Change settings.
- (2) From the Allowed programs and features list, select the name of the camera.
- (3) Select Domain, Home/Work (private), and Public.
- (4) Select RT ToolBox3, then select the same rules mentioned above and click OK.

(Procedure for Windows 7®)

1) **System and Security** (Control Panel)

Windows Firewall: **Allow a program through Windows Firewall** (System and Security)

2) **Change settings** (Allowed programs)

3) **Change the settings for the camera and RT ToolBox3.**

Name	Domain	Home/Work (Pri...)	Public	Group Policy
<input checked="" type="checkbox"/> RoboSim800Q	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No
<input checked="" type="checkbox"/> RoboSim800R	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No
<input type="checkbox"/> Routing and Remote Access	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No
<input checked="" type="checkbox"/> RT ToolBox3	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	No
<input checked="" type="checkbox"/> RTCPUmelFtpsvr	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No
<input type="checkbox"/> Secure Socket Tunneling Protocol	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No
<input type="checkbox"/> Secure World Wide Web Services (HTT...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No
<input checked="" type="checkbox"/> Sony	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	No
<input type="checkbox"/> SHARP Tray	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No
<input checked="" type="checkbox"/> TODD MILLISEC 3D	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	No
<input checked="" type="checkbox"/> TODD MILLISEC 3D	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	No

4) **Change the settings for the camera and RT ToolBox3.**

Select Domain, Home/Work (private), and Public, then click OK.

3.4 Creating an identification job

■ Identification job creation process

Step	Description
(1)	Create a new job.
(2)	Configure calibration file settings.
(3)	Set the workpiece to be identified and the identification range.
(4)	Set threshold and rotation tolerance values.
(5)	Configure communication settings.
(6)	Select an identified pattern.
(7)	Save the job.
(8)	Check calibration.

A job contains the program data of the 2D vision project.

Jobs are needed to manage data and communications transmitted between the robot and 2D vision sensor(s).

MELFA BASIC robot programs can control job access, imaging triggers and data acquisition of jobs created in In-Sight Explorer.

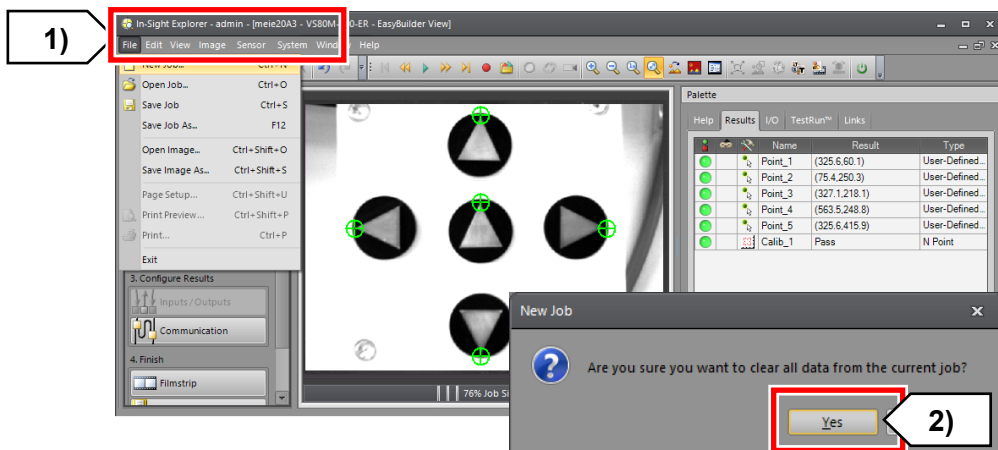
(1) Creating a new job

(1.1) Create a new job.

1) Go to File then New Job.

2) The dialog "Are you sure you want to clear all data from the current job?" will appear. Click Yes.

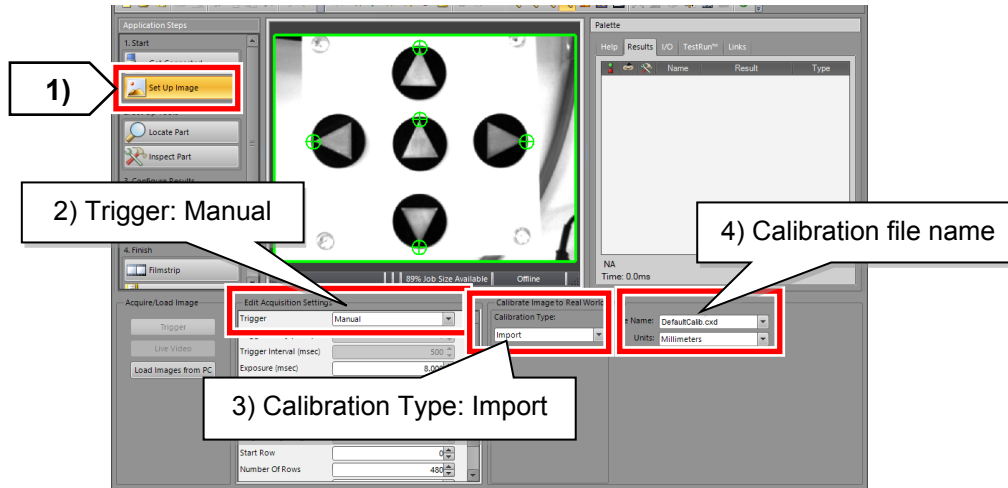
* Even if Yes is clicked, the calibration data will still be displayed.



(2) Configuring calibration file settings

(2-1) Select how to acquire an image and set a calibration file.

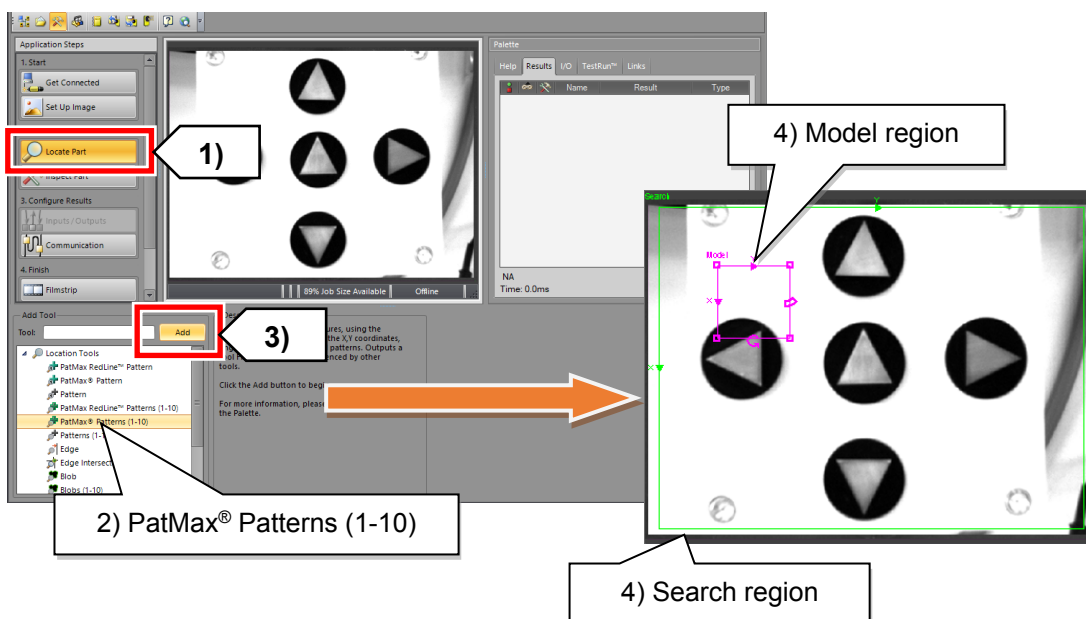
- 1) Select Set Up Image from the Application Steps window.
 - 2) Select Manual or External from the Trigger drop-down box under Edit Acquisition Settings.
 - * Select Manual in this seminar. (In actual operation, select an appropriate mode.)
 - 3) Select Import from the Calibration Type drop-down box under Calibrate Image to Real World Units.
 - 4) In the File Name field, select the calibration file exported in 3.3.2 Calibration method (Step 4.2).
- * Image data will be converted into robot coordinates from now on.



(3) Setting the workpiece to be identified and the identification range

(3.1) Select a location tool.

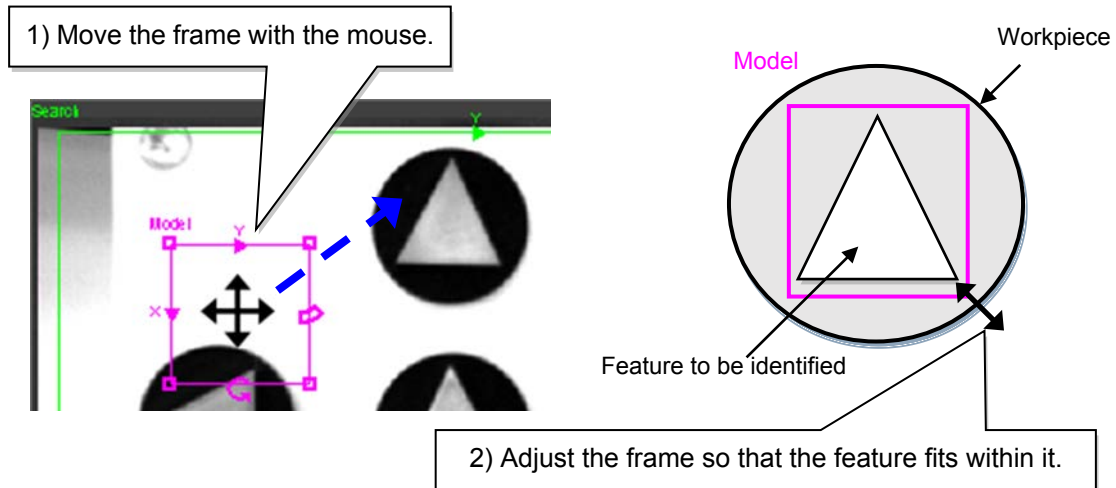
- 1) Select Locate Part from the Application Steps window.
- 2) Select PatMax® Patterns (1-10) from Location Tools in the bottom left.
- 3) Click Add.
- 4) Two boxes, the Model region box and the Search region, box will appear on the image.



(3.2) Set the feature of the workpiece to be identified.

Use the Model region box to identify a feature of the workpiece.

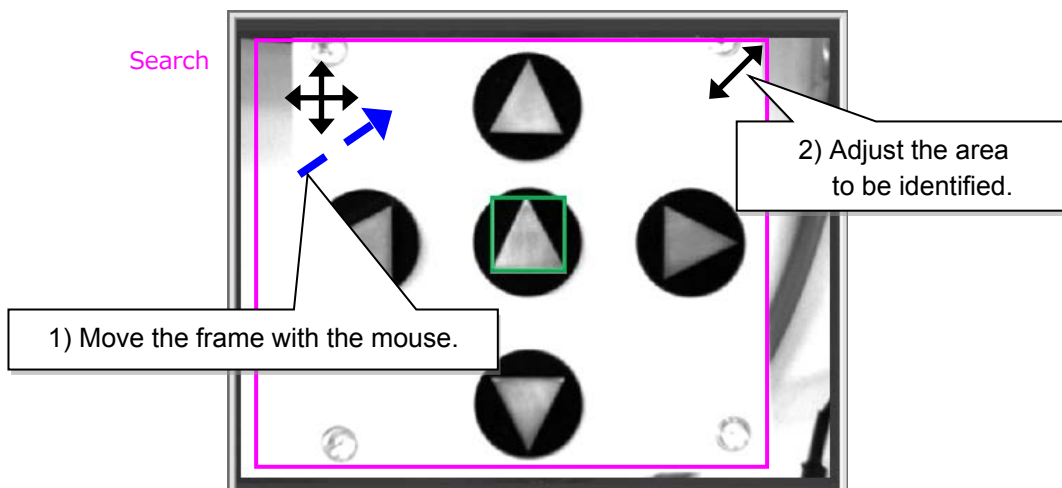
- 1) Click inside the Model region box. The frame of the Model region box will turn pink.
(The frame of the Search region box will turn green.)
- 2) Place the cursor inside the frame of the Model region box, press and hold down the left mouse button, then drag the Model region box over the workpiece.
- 3) Adjust the frame so that the workpiece fits within the Model region box.



(3.3) Set the region to be identified.

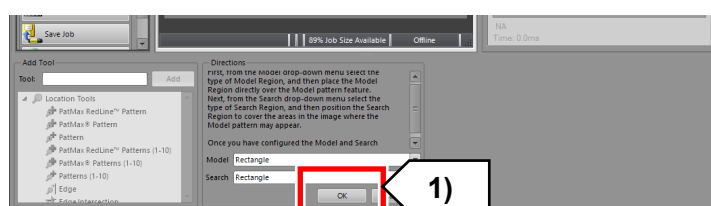
Set the region to be identified using the Search region box.

- 1) To select the Search region box, click inside its frame. The frame of the Search region box will turn pink. (The frame of the Model region box will turn green.)
- 2) Move and adjust the frame in the same manner as Step 3.2 to set the area to be identified.

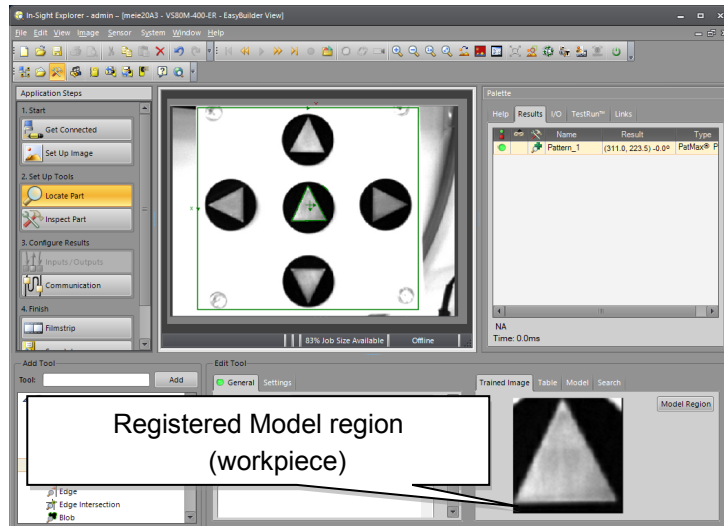


(3.4) Set the Model region and Search region.

- 1) After setting the Model region and Search region, click OK.



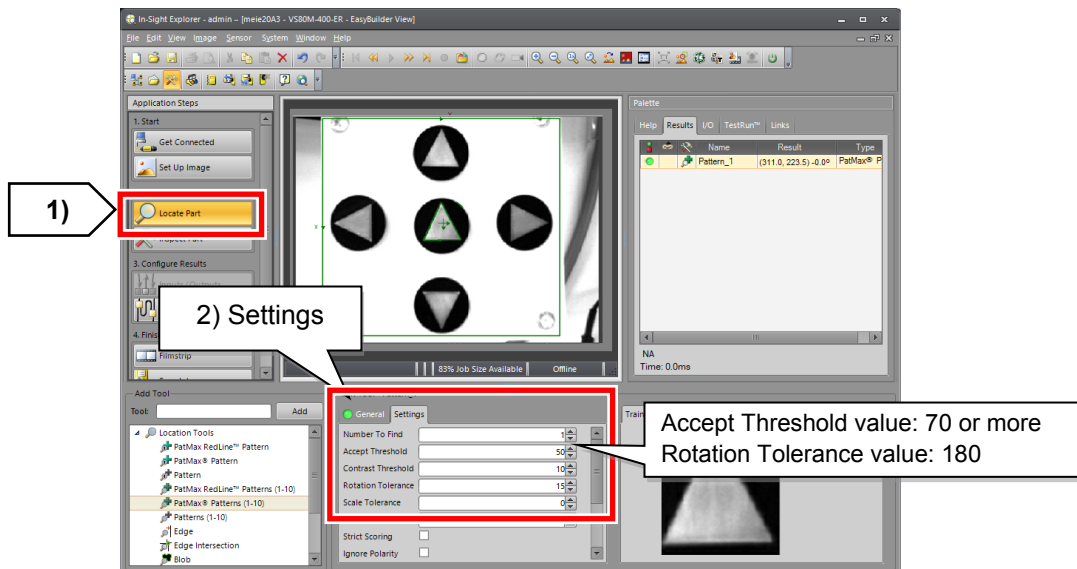
2) The registered image of the workpiece to be identified (Model region) will be displayed in the bottom right.



(4) Setting threshold and rotation tolerance values

(4.1) Set threshold and rotation tolerance values.

- 1) Select Locate Part from the Application Steps window.
- 2) Click the Settings tab and set the Accept Threshold and Rotation Tolerance values.
 - * Set the Accept Threshold value to 70 or more and the Rotation Tolerance value to 180.
 - * Changing horizontal and vertical offsets will enable the tool center point (TCP) to be changed freely.
- 3) To perform the next calibration check, move the TCP to the apex of the center triangle.



Accept Threshold value: The amount (%) in which the identified workpiece matches the registered image.

- A rotation tolerance value is an angle at which the actual workpiece can be identified even if the workpiece is rotated (\pm deg).

(5) Configuring communication settings

(5) Configure the communication settings.

- 1) Select Communication from the Application Steps window.
 - 2) Click Add Device.
 - 3) The Device Setup window will appear on the right. Configure the settings in the following order: (1) Device, (2) Manufacturer, (3) Protocol. (Selecting an item will display the corresponding drop-down list below.)
- * Select Robot for Device, Mitsubishi for Manufacturer, and Ethernet Native String for Protocol.
- 4) Click OK.

Field	Setting value
Device	Robot
Manufacturer	Mitsubishi
Protocol	Ethernet Native String

After a field has been set, the next field will appear below.

- 5) Ethernet Native String will be added to Communications in the bottom left. The Format Output String window, Edit Device button, and Remove Device button will also appear. To revise device settings (Step 3), click Edit Device.

(6) Selecting an identified pattern

Select the output data of an identified pattern in the order that the data is output to the robot.

* By setting the order of the output string of the identified pattern to X, Y, Angle, position variables can be input as they are when using the robot command "EBREAD".

Select the data in the following order. (The order can be changed later.)

Number of identified patterns	Pattern_1.Number_Found
First X of the matching result	Pattern_1.Fixture.X
First Y of the matching result	Pattern_1.Fixture.Y
First rotational angle C of the matching result	Pattern_1.Fixture.Angle

(6.1) Select the output data of an identified pattern.

- 1) Select Communication from the Application Steps window.
- 2) Select Ethernet Native String under Communications in the bottom left to display the Format Output String window.
- 3) Click Add under Format Output String to display the Select Output Data window.
- 4) Click the blue arrow next to Pattern_1 in the Select Output Data window to display the Pattern_1 list.
- 5) Select data in the following order while holding down the Ctrl key. (1) Pattern_1.Number_Found, (2) Pattern_1.Fixture.X, (3) Pattern_1.Fixture.Y, (4) Pattern_1.Fixture.Angle. (The order can be changed later.)

Caution: Be careful when selecting output data because the names are similar to each other.

- 6) Click OK. The data will be displayed in the Format Output String window in the order selected.

1) **Inspect Part**

2) **Ethernet Native String**

3) **Add...**

4) Click the blue arrow of **Pattern_1**.

Select Output Data window

5) Select each piece of data in order while holding down the Ctrl key.
 Pattern_1.Number_Found
 Pattern_1.Fixture.X
 Pattern_1.Fixture.Y
 Pattern_1.Fixture.Angle

Format Output String

Name	Data Type	Value
Pattern_1.Number_Found	Floating Point	0.000
Pattern_1.Fixture1.X	Floating Point	
Pattern_1.Fixture1.Y	Floating Point	
Pattern_1.Fixture1.Angle	Floating Point	

6) **OK**

Be careful when selecting data because the names are similar to each other.

Each piece of data is displayed in the order it is selected in

The order can be changed.

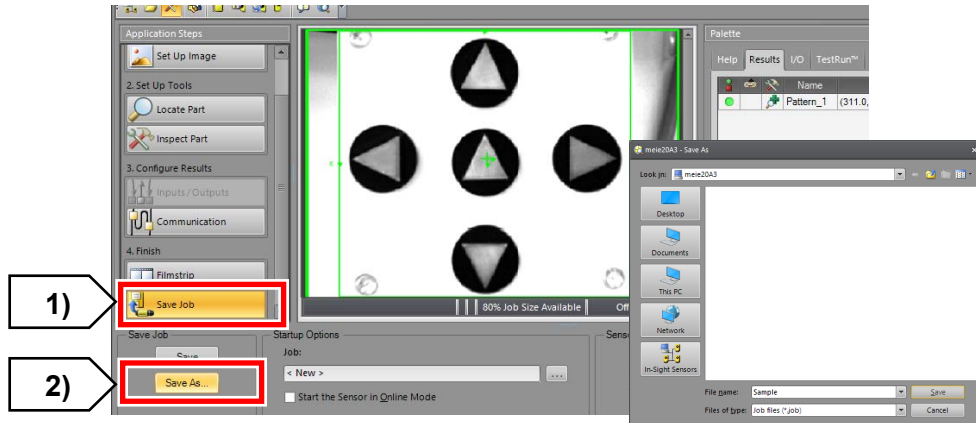
(7) Saving the job

(7.1) Save the Job.

- 1) Select Save Job from the Application Steps window.
- 2) Click Save As and enter "Sample" in the File name field.

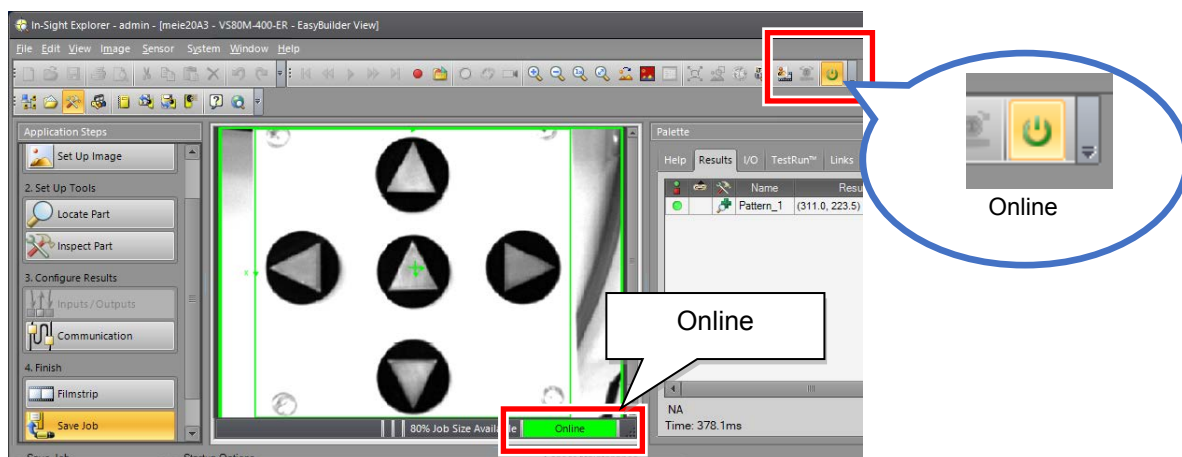
* This file name will be used in UVS1.prg.

The job file can be named freely, but the file name must be used in UVS1.prg.



(7.2) Change the operation mode to Online.

- 1) Click the Online button on the tool bar to change the operation mode to Online.



Notes

(8) Checking calibration

Once calibration is complete, the XY coordinates output by the vision sensor will become the robot coordinates after the calibration file is imported.

The method below explains how to check if calibration has been performed correctly or not.

(8.1) Checking calibration without moving the workpiece (master workpiece check)

Complete the following steps without moving the workpiece that is used for calibration.

Create a new job and identify the workpiece. Move the robot arm in Jog mode to the coordinates displayed on the teaching pendant, then move robot arm up and down in the Z axis.

Calibration has been successful if the point of the calibration tool (control point) lines up with the TCP on the image in In-Sight Explorer.

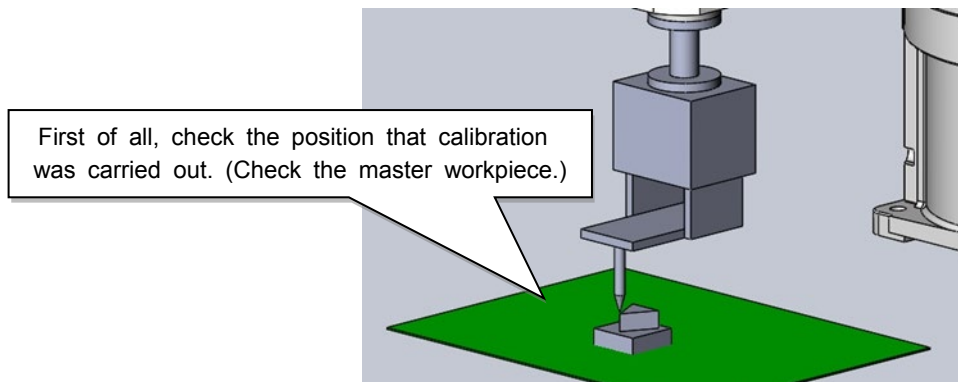


Fig. 3.5: Calibration check using the TCP

(8.2) Checking calibration by moving the workpiece adjacent to itself

After checking that the control point lines up with the TCP in In-Sight Explorer, position the workpiece adjacent to itself. Then check to see if the control point still lines up with the TCP and that the workpiece can be identified.

If the control point does not line up with the TCP on the image in In-Sight Explorer, the XY coordinates output by the vision sensor do not match the control point. The reason for this happening could be due to the following points:

- (1) The workpiece has not been identified properly. (Check that the TCP is in the same place on the image as the position of the control point.)
- (2) There is too much glare from the sides of the workpiece. (Register a workpiece which does not have as much glare coming from it.)
- (3) Calibration was not carried out at the same height of the workpiece. (Carry out calibration at the same height of the workpiece.)
- (4) Points were selected incorrectly (Check the positions of the robot and vision sensor during calibration.)

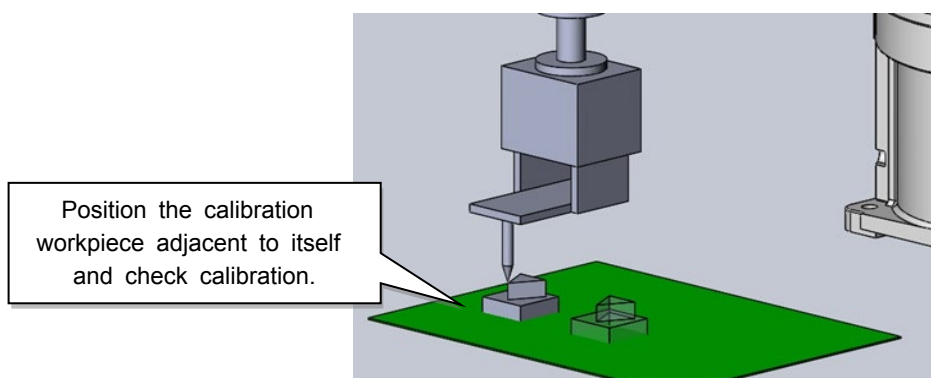


Fig. 3.6: Calibration check moving

(8.3) Checking calibration by rotating the workpiece

After checking calibration by moving the workpiece adjacent to itself, check that the TCP and control point match up when the workpiece is rotated.

If calibration has been successful, the TCP and control point will match when the workpiece has been rotated like the workpiece in Fig. 3.7.

The following reasons may account for why the TCP and control points line up when the workpiece is moved adjacent to itself but not when it is rotated.

- (1) The control point (tool point) is not set correctly.
- (2) Calibration was carried out without aligning the hand.

Redo the calibration steps from TLUP.prg.

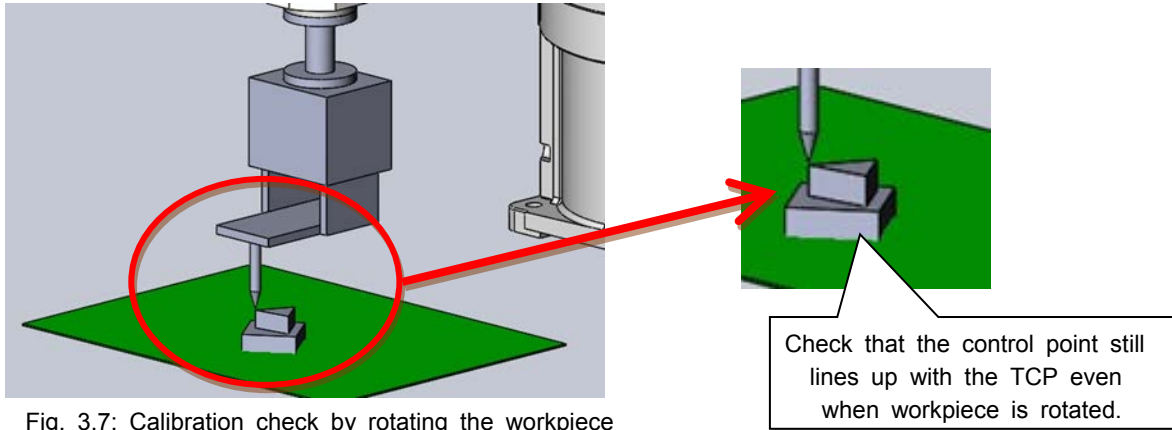


Fig. 3.7: Calibration check by rotating the workpiece

3.5 Setting up the relative position between the workpiece and the robot

3.5.1 Program for setting up the relative position between the workpiece and the robot

The programs used in this chapter are UVS1.prg and UVS2.prg.

UVS1.prg: Fixed downward-facing camera adjustment program. Corrects the workpiece grasp position.

UVS2.prg: Fixed downward-facing camera operation execution program. Calculates the workpiece picking position based on the value output from the vision sensor, allowing the robot to pick and place workpieces automatically.

◇ Program UVS1.prg

```
1 '-----
2 ' UVS1.prg: Fixed downward-facing camera adjustment program
3 '-----
4 Def Pos P_WRK01 ' Fixed downward-facing camera: Master workpiece grasp position
5 Def Pos P_PH01 ' Fixed downward-facing camera: Coefficient for calculating the handling position
  from the position of the identified workpiece
6 Def Pos P_TLUP ' Fixed downward-facing camera: Control point data used for calibration
7 Def Pos P_PVS01 ' Fixed downward-facing camera: Master workpiece identification point
8 Def Char C_J01 ' Fixed downward-facing camera: Job name
9 Def Char C_C01 ' Fixed downward-facing camera: COM name
10 '
11 Loadset 1,1
12 OAdl On
13 OvrD M_NOvrD
14 Spd M_NSpd
15 Tool P_TLUP
16 Servo On
17 Wait M_Svo=1
18 CCOM$="COM3:"      ' COM port
19 CPRG$="sample.job" ' Vision sensor job name
20 '---- HAND INIT ----
21 Wait M_Svo=1
22 If M_EHOrg=0 Then          ' If hand has not returned to origin:
23   EHOrg 1                 ' returns Hand 1 to origin.
24   Wait M_EHOrg=1         ' waits for Hand 1 to return to origin.
25 EndIf
26 EHOpen 1,100,100        ' Opens Hand 1 (speed = 100%, Force = 20%)
27 Wait M_EHBusy=0         ' Checks if operation is complete
28 '-----
29 ' 1. Teach safe position "PHOME".
30 ' 2. Set "1" in the robot parameter NVTRGTMG and restart the robot.
31 ' 3. Close Hand 1 with the teaching pendant and lower the suction pad.
32 ' 4. Teach the robot the place on the workpiece it should pick it up by in PWK and then send the
  robot arm to the safe position.
33 ' 5. With the teaching pendant, change OVRD to 3% and start automatic operation.
34 Hlt
35 ' ---- Automatic operation ----
36 Mov PHOME
37 Dly 0.5
38 If M_NvOpen(1)<>1 Then
```

(continued on the next page)


```

39   NVOpen CCOM$ As #1
40   EndIf
41   NVRun #1,CPRG$
42   EBRead #1,,MNUM,PVS
43   If MNUM=0 Then *ELOOP
44   PVS.FL1=PWK.FL1
45   PVS.FL2=PWK.FL2
46   PH=Inv(PVS)*PWK      '---- Calculation for correcting the workpiece grasp position ----
47   '-----
48   P_PH01=PH
49   P_PHome=PHOME
50   P_WRK01=PWK
51   P_PVS01=PVS
52   C_C01$=CCOM$
53   C_J01$=CPRG$
54   '-----
55   Hlt
56   End
57   '***** Error occurrence *****
58   *ELOOP
59   Error 9000
60   Hlt
61   GoTo *ELOOP
62   End
P_WRK01=(+253.62,+299.22,+253.74,-180.00,+0.00,+180.00,+0.00,+0.00)(7,0)
P_PH01=(-10.22,+90.64,+253.74,-180.00,+0.00,+178.70,+0.00,+0.00)(7,0)
P_TLUP=(+0.62,-0.61,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(0,0)
P_PVS01=(+265.90,+208.84,+0.00,+0.00,+0.00,+1.30,+0.00,+0.00)(7,0)
PHOME=(+187.48,-13.81,+320.67,-180.00,+0.00,+180.00,+0.00,+0.00)(7,0)
PVS=(+265.90,+208.84,+0.00,+0.00,+0.00,+1.30,+0.00,+0.00)(7,0)
PWK=(+253.62,+299.22,+254.74,-180.00,+0.00,+180.00,+0.00,+0.00)(7,0)
PH=(-10.22,+90.64,+254.74,-180.00,+0.00,+178.70,+0.00,+0.00)(7,0)
P_PHome=(+187.48,-13.81,+320.67,-180.00,+0.00,+180.00,+0.00,+0.00)(7,0)

```

(End of UVS1.prg)

◇ Program UVS2.prg

```
1 '-----
2 ' UVS2.prg: Fixed downward-facing camera operation execution program
3 '-----
4 '
5 GoSub *INIT
6 '
7 Mov PHOME
8 Dly 0.5
9 If M_NvOpen(1) <> 1 Then
10   NvOpen CCOM$ As #1
11 EndIf
12 *VSCHK
13 NVRun #1,CPRG$
14 EBRead #1,,MNUM,PVS
15 If MNUM>=1 Then *VSOK
16 GoTo *VSCHK
17 *VSOK
18 PVS.FL1=PWK.FL1
19 PVS.FL2=PWK.FL2
20 PTRG=PVS*PH      '--- Calculation of the workpiece grasp position ----
21 Mov PTRG,-50 ' (RH series: +100) *1
22 Dly 0.2
23 HClose 1
24 Fine 0.1,P
25 Mvs PTRG
26 Cnt 0
27 Dly 0.2
28 M_Out(10129)=1
29 Dly 0.2
30 Mvs PTRG,-50 ' (RH series: +100) *1
31 HIt
32 '
33 ' Teach the destination position "PPUT" here.
34 '
35 Mov PPUT,-50 ' (RH series: +100) *1
36 Fine 0.1,P
37 Mvs PPUT
38 Fine 0
39 Dly 0.2
40 M_Out(10129)=0
41 M_Out(10128)=1 Dly 0.1
42 Dly 0.2
43 HOpen 1
44 Dly 0.2
45 Mvs PPUT,-50 ' (RH series: +100) *1
46 ' ***** Return to safe position *****
47 Mov PHOME
48 HIt
49 End
```

(continued on the next page)

```

50 '-----
51 *INIT
52 Def Pos P_WRK01 ' Fixed downward-facing camera: Master workpiece grasp position
53 Def Pos P_PH01 ' Fixed downward-facing camera: Coefficient for calculating the handling position
from the position of the identified workpiece
54 Def Pos P_TLUP ' Fixed downward-facing camera: Control point data used for calibration
55 Def Pos P_PVS01 ' Fixed downward-facing camera: Master workpiece identification point
56 Def Char C_J01 ' Fixed downward-facing camera: Job name
57 Def Char C_C01 ' Fixed downward-facing camera: COM name
58 Loadset 1,1
59 OAdl On
60 OvrD M_NOvrD
61 Spd M_NSpd
62 Tool P_TLUP
63 Servo On
64 Wait M_Svo=1
65 Cnt 0
66 M_Out(10129)=0
67 M_Out(10128)=1 Dly 0.1
68 PH=P_PH01
69 PHOME=P_PHome
70 PWK=P_WRK01
71 CCOM$=C_C01$
72 CPRG$=C_J01$
73 HOpen 1
74 M_Out(10128)=0
75 Dly 0.2
76 If P_Fbc.Z < PHOME.Z Then
77   PNow=P_Fbc
78   PNow.Z=PHOME.Z
79   Mvs PNow
80   Dly 0.2
81 EndIf
82 '---- HAND INIT ----
83 Wait M_Svo=1
84 If M_EHOrg=0 Then                                ' If hand has not returned to origin:
85   EHOrg 1                                        ' returns Hand 1 to origin.
86   Wait M_EHOrg=1                                ' waits for Hand 1 to return to origin.
87 EndIf
88 EHOpen 1,100,100                                ' Opens Hand 1 (speed = 100%, Force = 20%)
89 Wait M_EHBusy=0                                  ' Checks if operation is complete
90 '-----
91 Dly 0.2
92 Return
PHOME=(+187.48,-13.81,+320.67,-180.00,+0.00,+180.00,+0.00,+0.00)(7,0)
PVS=(+255.79,+223.38,+0.00,+0.00,+0.00,-323.73,+0.00,+0.00)(7,0)
PWK=(+253.62,+299.22,+253.74,-180.00,+0.00,+180.00,+0.00,+0.00)(7,0)
PTRG=(+193.93,+290.41,+253.74,+180.00,+0.00,-145.03,+0.00,+0.00)(7,0)
PH=(-10.22,+90.64,+253.74,-180.00,+0.00,+178.70,+0.00,+0.00)(7,0)
PPUT=(+280.35,-87.22,+248.68,+180.00,+0.00,-180.00,+0.00,+0.00)(7,0)

```

(continued on the next page)

```
P_WRK01=(+253.62,+299.22,+253.74,-180.00,+0.00,+180.00,+0.00,+0.00)(7,0)
P_PH01=(-10.22,+90.64,+253.74,-180.00,+0.00,+178.70,+0.00,+0.00)(7,0)
P_TLUP=(+0.62,-0.61,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(0,0)
P_PVS01=(+265.90,+208.84,+0.00,+0.00,+0.00,+1.30,+0.00,+0.00)(7,0)
P_PHome=(+187.48,-13.81,+320.67,-180.00,+0.00,+180.00,+0.00,+0.00)(7,0)
PNow=(+280.35,-87.22,+320.67,+180.00,+0.00,-180.00)(7,0)
```

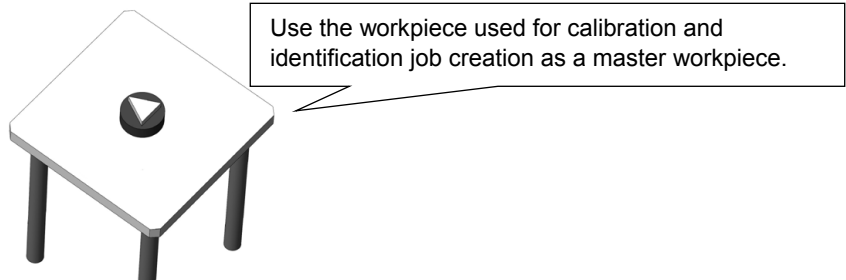
(End of UVS2.prg)

3.5.2 Setting up the relative position between the workpiece and the robot

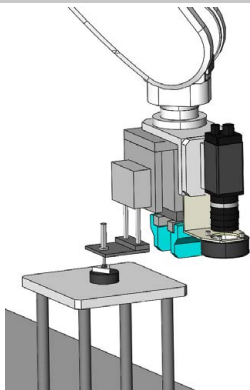
■ Procedure

Step	Description
(1)	Positioning the master workpiece used for adjustment
(2)	Teaching the safe position (PHOME) and suction area on the master workpiece (PWK)
(3)	Running the robot program "UVS1" automatically
(4)	Teaching the destination position "PPUT"
(5)	Checking the robot movement using step operation and automatic operation (robot program "UVS2")

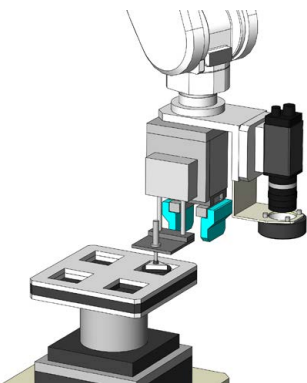
(1) Position the master workpiece used for adjustment.



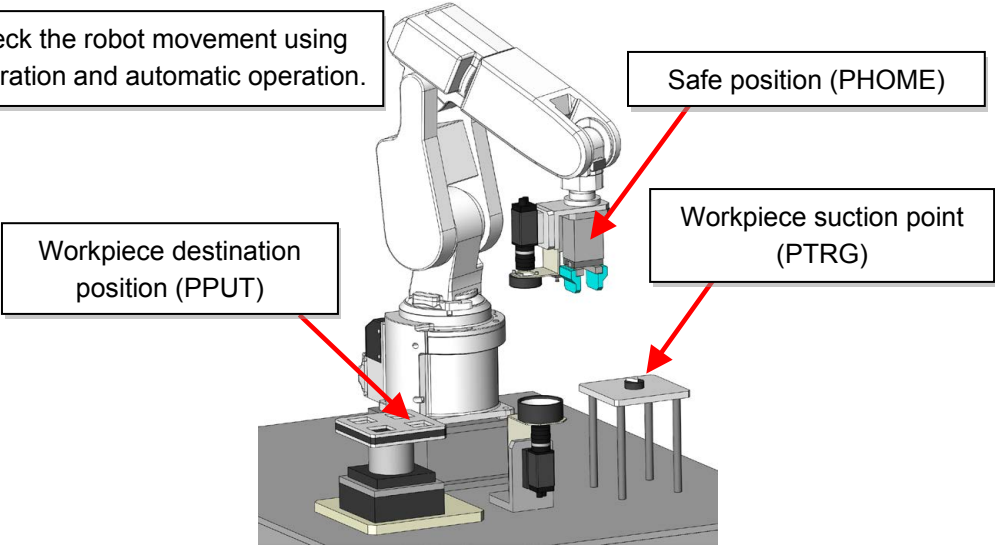
(2) Teach the suction area on the master workpiece (PWK).



(4) Teach the destination position (PPUT).



(5) Check the robot movement using step operation and automatic operation.



(1) Positioning the master workpiece used for adjustment

- (1) Position the master workpiece used for adjustment and check the parameter.
- 1) Remove the calibration workpiece and pointed object, then place a workpiece on the platform. This workpiece will now be used as the master workpiece for adjustment.
 - 2) Check that the robot parameter "NVTRGTMG" is set to "1". If not, set "1" and restart the robot controller.

(2) Teaching the safe position (PHOME) and suction area on the master workpiece (PWK)

- (2.1) Run the robot program "UVS1".
- 1) Run the program up to line 28 using step operation.
 - ◇ Specify a job file in line 19 (CPRG\$="sample.job"). (The file is the one saved in Step 7 of Section 3.4.)
 - ◇ In line 23, press and hold the F1 key until the hand operation completes. (Initialization of the electric hand)
- ◇ Program UVS1.prg

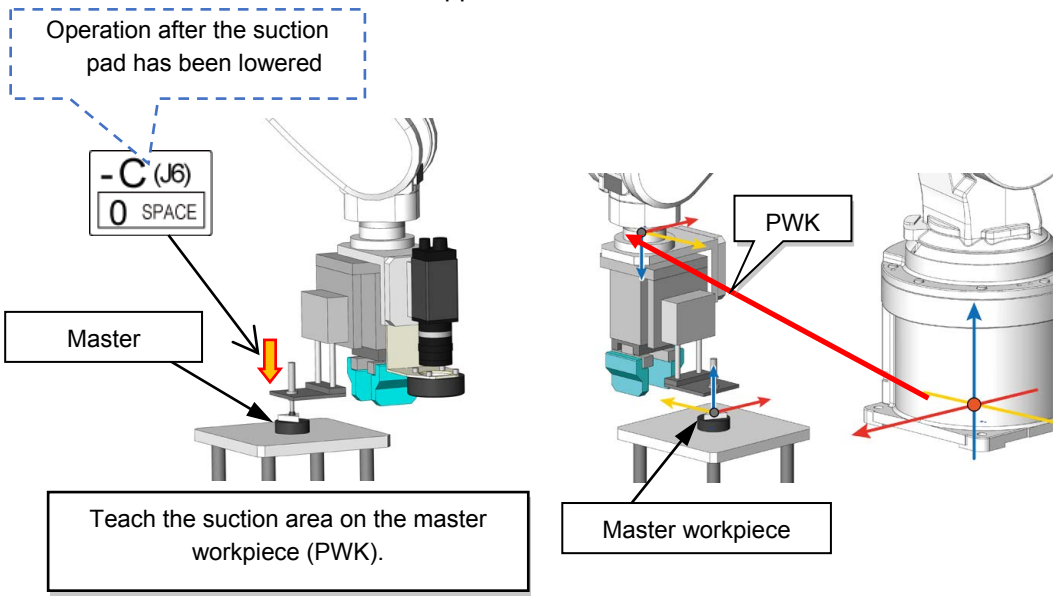
.....

```
18 CCOM$="COM3:" 'COM port
19 CPRG$="sample.job" ' Vision sensor job name
```
- Specify an identification job file. →

(2.2) Teach the safe position (PHOME) and suction area on the master workpiece (PWK).

- 1) Teach the safe position (PHOME). PHOME: Safe position or position during imaging.
- 2) Close Hand 1 with the teaching pendant and lower the suction pad. In the pneumatic hand (standard hand) window, press -C.
- 3) Move the robot to a place where it can apply suction to the master workpiece used for adjustment (set in Step 1), and teach the position as PWK. PWK: The suction area on the master workpiece. Used for finding PH (correction value for workpiece suction).
- 4) Move the robot arm to a place where it will not interfere with anything.

⚠ CAUTION Make sure the workpiece does not move after teaching PWK.
 *If the workpiece moves at this point, its position will deviate when suction is applied to it.



(3) Running the robot program "UVS1" automatically

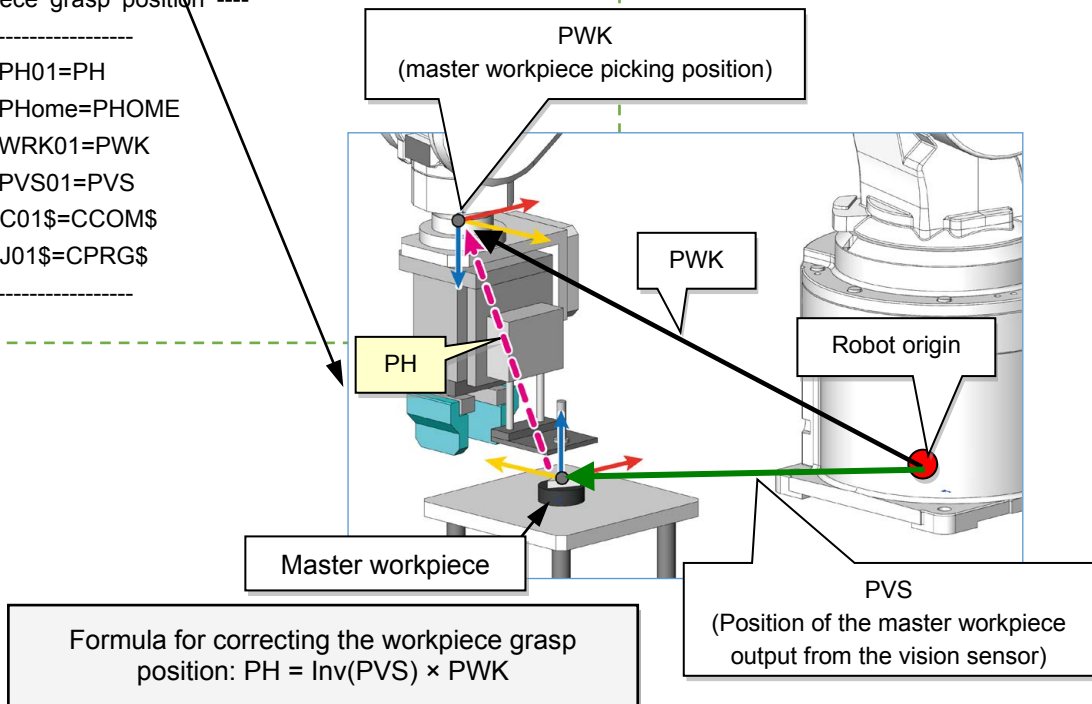
(3.1) Run the program automatically.

- 1) Move the robot to the safe position "PHOME" using Jog operation.
- 2) Set OvrD to 3%.
- 3) Check that OvrD is set to 3% and run UVS1.prg automatically.
- 4) The program will stop at line 34. Run the program again until it stops at line 55.
- 5) Check the variable values of PVS in the Variables monitor of RT ToolBox3. Check that the X, Y, and C components of PVS are the same as the X, Y, and C components of the vision sensor. The component values of the vision sensor can be checked in In-sight Explorer in the following area: the window on the right of the Format Output String window in Communications settings.

◇ Program UVS1.prg

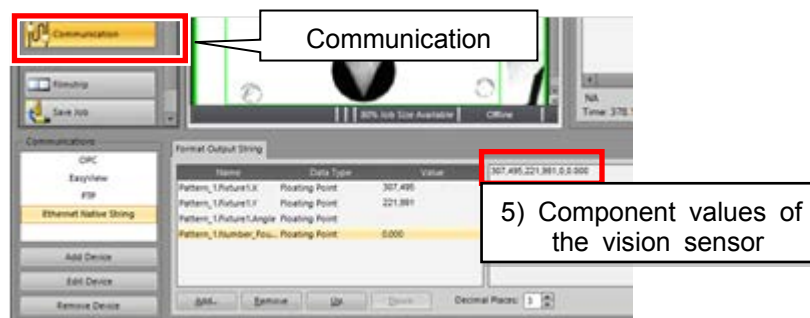
```

.....
44 PVS.FL1=PWK.FL1
45 PVS.FL2=PWK.FL2
46 PH=Inv(PVS)*PWK '---- Calculation for correcting the
workpiece grasp position ----
47 '-----
48 P_PH01=PH
49 P_PHome=PHOME
50 P_WRK01=PWK
51 P_PVS01=PVS
52 C_C01$=CCOM$
53 C_J01$=CPRG$
54 '-----
55 Hit
  
```



PVS: Position of the master workpiece output from the vision sensor

PWK: Corrected position of the master workpiece



(4) Teaching the destination position (PPUT)

(4.1) Run the robot program "UVS2" automatically.

- 1) Run the program automatically.
- 2) The robot will stop operation at line 31 "Hit". Put the robot in manual mode and teach the destination position (PPUT).
- 3) After teaching PPUT, raise the robot arm to keep it away from the destination position.
- 4) Rerun UVS2.prg automatically.

◇ Program UVS2.prg

```

5 GoSub *INIT
6 '
7 Mov PHOME
8 Dly 0.5
9 If M_NvOpen(1) <> 1 Then
10 NVOpen CCOM$ As #1
11 EndIf
12 *VSCHK
13 NVRun #1,CPRG$
14 EBRead #1,,MNUM,PVS
15 If MNUM>=1 Then *VSOK
16 GoTo *VSCHK.....
17 *VSOK
18 PVS.FL1=PWK.FL1
19 PVS.FL2=PWK.FL2
20 PTRG=PVS*PH '--- Calculation of the workpiece
   grasp position ----
21 Mov PTRG,-50 ' (RH series: +100)
22 Dly 0.2
23 HClose 1
24 Fine 0.1,P
25 Mvs PTRG
.....
30 Mvs PTRG,-50 ' (RH series: +100) *1
31 Hit
32 '
33 ' Teach the destination position
   "PPUT" here.
.....

```

◇ Program UVS2.prg

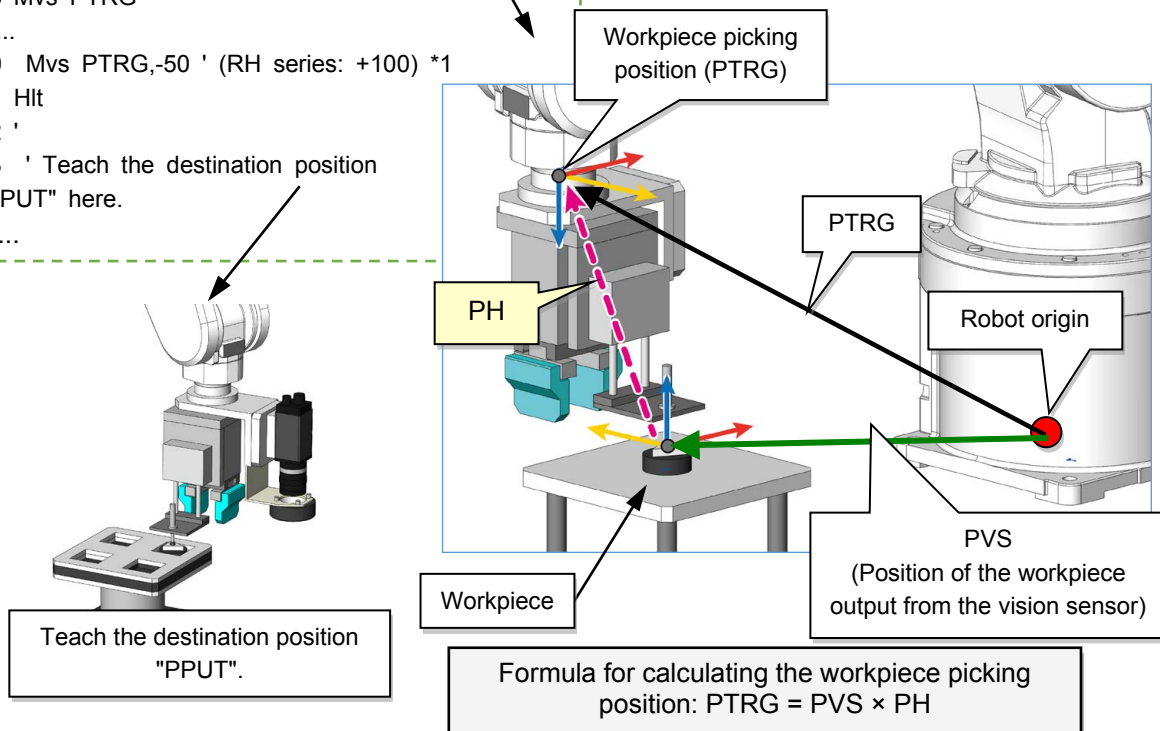
```

.....
51 *INIT
.....
62 Tool P_TLUP
.....
68 PH=P_PH01
69 PHOME=P_PHome
70 PWK=P_WRK01
71 CCOM$=C_C01$
72 CPRG$=C_J01$
.....
76 If P_Fbc.Z < PHOME.Z Then
77 PNow=P_Fbc
78 PNow.Z=PHOME.Z
79 Mvs PNow
.....

```

Vision sensor
operating

PH and other data
in the UVS1.prg
are set.

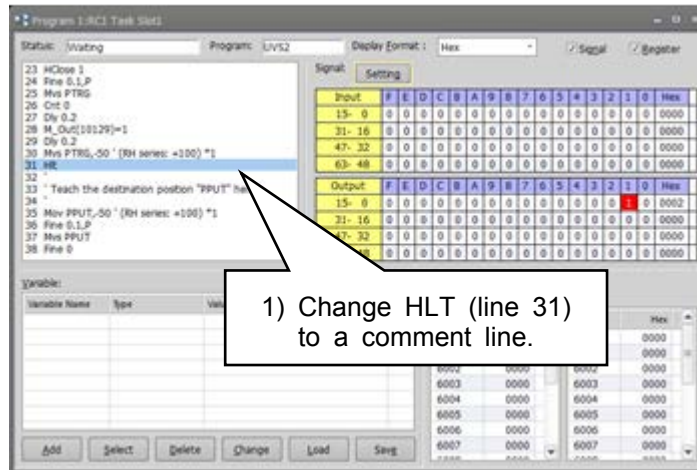


PVS: Position of the workpiece output from the vision sensor
PH: Correction value for workpiece grasp position calculated in UVS1.prg

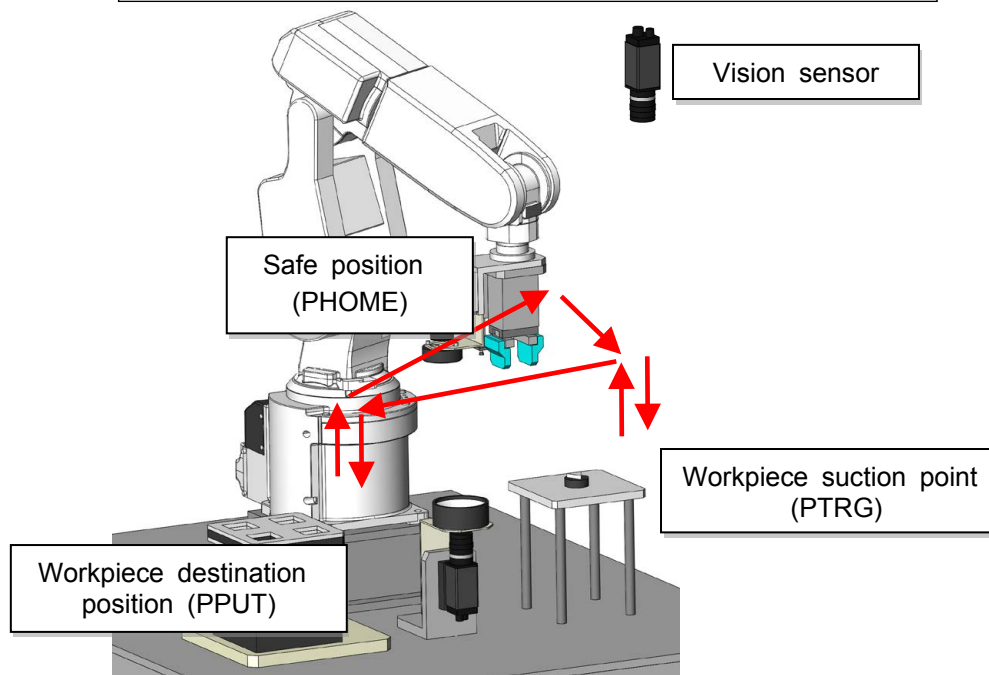
3.6 Checking the robot movement using step operation and automatic operation

- 1) In the program edit window, change HLT (line 31) of UVS2.prg to a comment line and save the program.
- 2) Check that OvrD is set to 3%, and run UVS2.prg automatically.

Robot program "UVS2"



- 2) Run the program automatically with OvrD set to 3%.



Chapter 4 - Setting up the hand eye

This chapter explains the process of setting up the hand eye.

4.1 Overview of the hand eye

4.1.1 Setting a control point used for calibration

Fig. 4.1 shows a standard application of the hand eye.

The robot recognizes how far the workpiece is away from the robot's origin and determines how to move toward the workpiece in the same manner as when a fixed downward-facing camera is used.

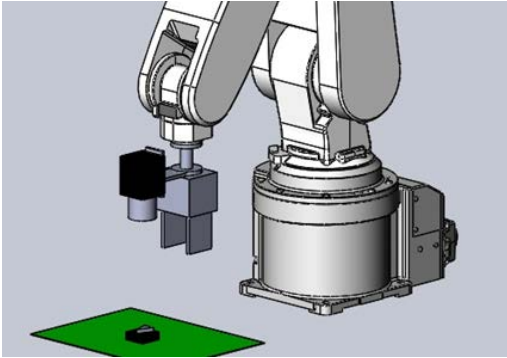


Fig. 4.1: Standard application

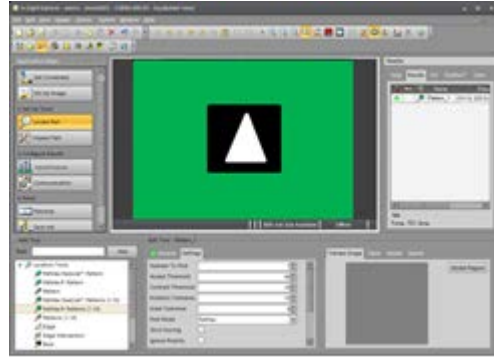


Fig. 4.1a: Workpiece seen from the hand eye shown in Fig. 4.1

However, calibration cannot be accomplished in the same manner as when a fixed downward-facing camera is used.

This is because the vision sensor is attached to the flange of the robot. If the robot moves, the vision sensor's FOV changes accordingly. Therefore, it is not easy to get coordinates of the robot's origin from the vision sensor's FOV.

The images taken by the vision sensor shown in Fig. 4.1, Fig. 4.2, and Fig. 4.3 are those shown in Fig. 4.1a, Fig. 4.2a, and Fig. 4.3a respectively. This makes it difficult to find where the workpiece is.

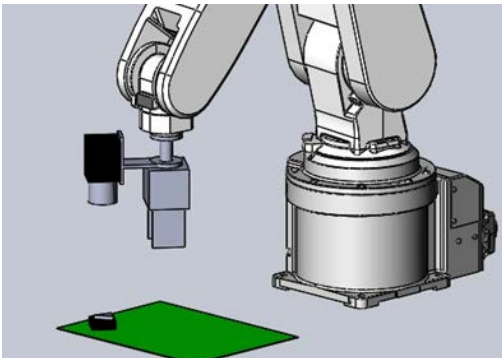


Fig. 4.2: Imaging position 1

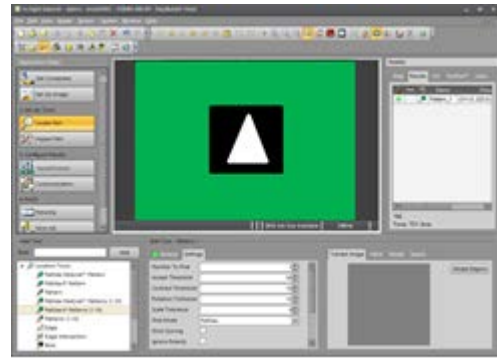


Fig. 4.2a: Workpiece seen from the hand eye shown in Fig. 4.2

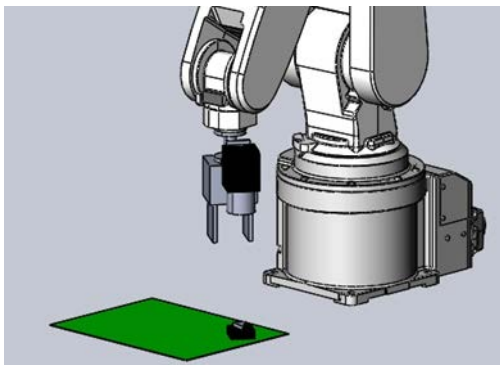


Fig. 4.3: Imaging position 2

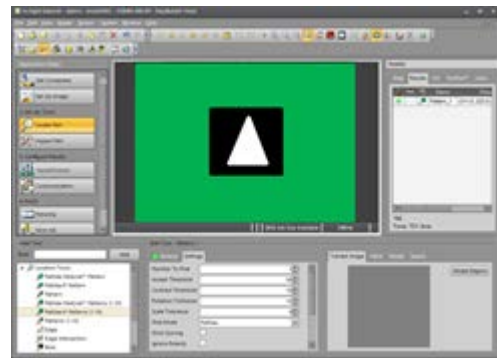


Fig. 4.3a: Workpiece seen from the hand eye shown in Fig. 4.3

Therefore, when the hand eye is used, the position of the workpiece can be calculated in the following way.

Workpiece position 4 = imaging position 1 × vision sensor center 2 seen from flange center × output position 3 seen from vision sensor center

- The robot always knows where the center of the flange is. (1 in Fig. 4.4)
- The vision sensor is attached to the flange. Therefore, even if the robot moves, the distance between the flange center and the vision sensor center is always the same. In the tool coordinates, the coordinate values of the vision sensor center change according to those of the flange center. (2 in Fig. 4.4)
- The robot can find workpieces only when they are within the vision sensor's FOV. The position of the workpieces can be calculated with reference to the center of the vision sensor. (3 in Fig. 4.4)

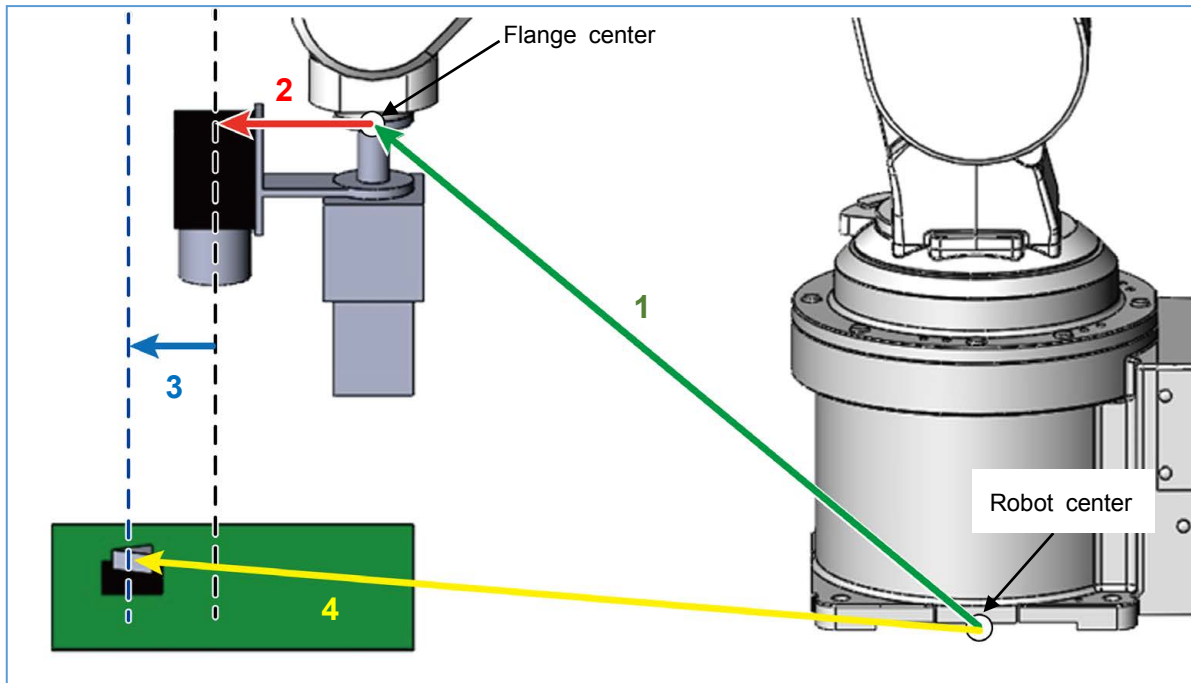


Fig. 4.4: Principle of how the position of the workpiece can be calculated using the hand eye

4.1.2 Control point for calibrating the hand eye

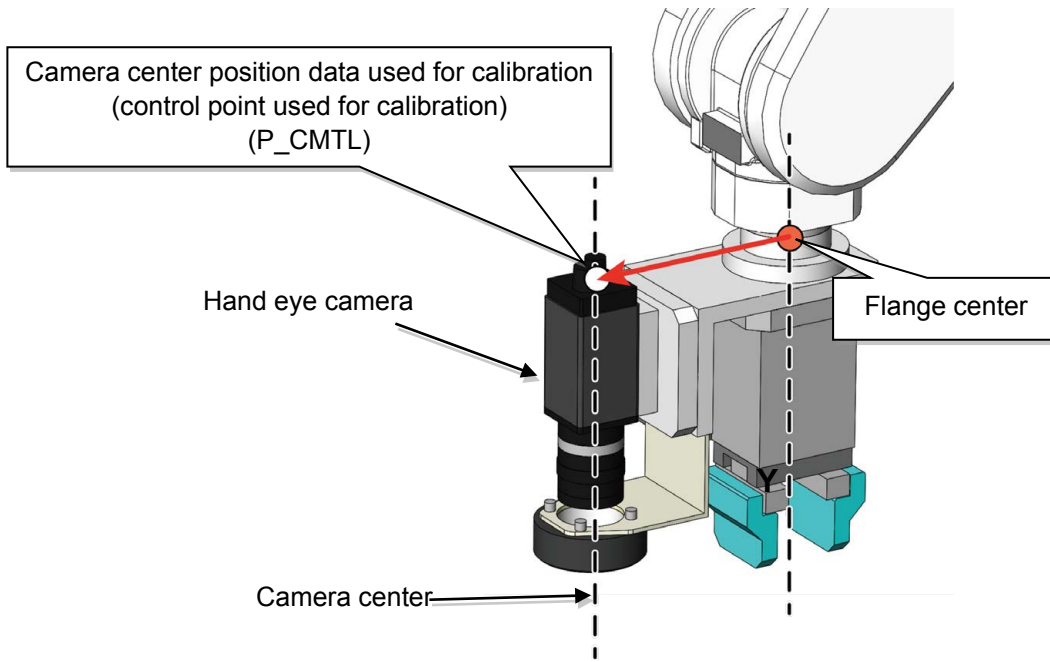


Fig. 4.5: Control point for calibrating the hand eye

4.1.3 Calibrating the hand eye

Hand eye calibration is carried out in tool coordinates instead of XYZ coordinates.

Draw two red intersecting lines in the center of the screen as shown in Fig. 4.7 to identify the center of the vision sensor. Use the center for the control point.

Move the robot so that the TCP(yellow lines) and crosshairs overlap each other. For the first calibration, the coordinates where they cross are (0, 0).

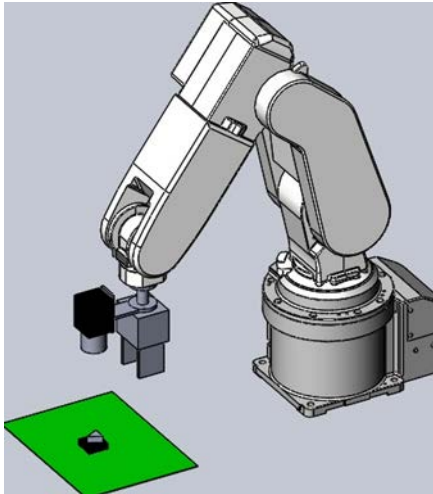


Fig. 4.6: Calibration 1

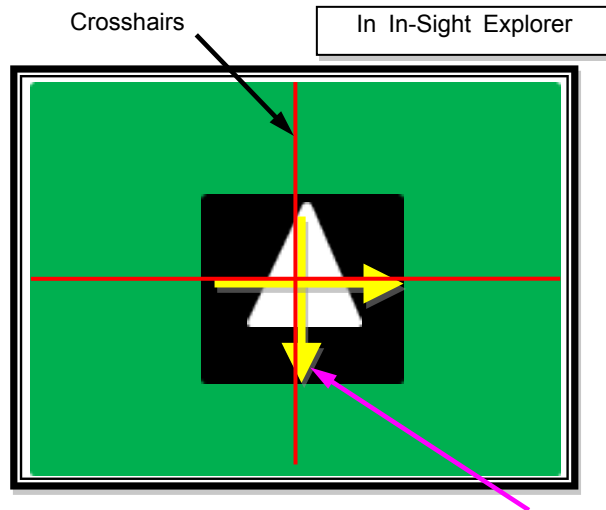


Fig. 4.7: Calibration screen 1 Tool center point

Then, move the robot in the direction of the X axis in the tool coordinates from (0, 0) as shown in Fig. 4.8, and take an image again. (The robot is moved only in direction of the X axis, and therefore the Y component is always 0.)

After the robot has moved, the TCP is located as shown in Fig. 4.9.

A second calibration is made using the reciprocal of the point in the tool coordinates and the tool center point in the pixel coordinates of the vision sensor.

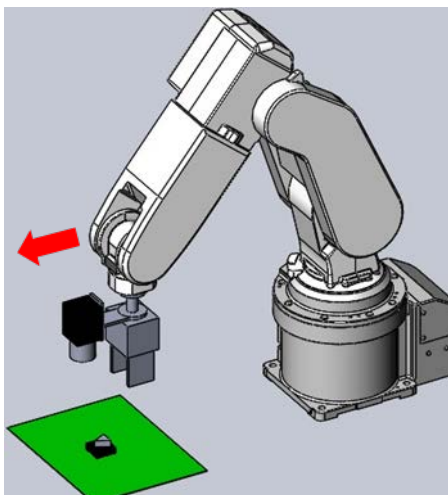


Fig. 4.8: Calibration 2

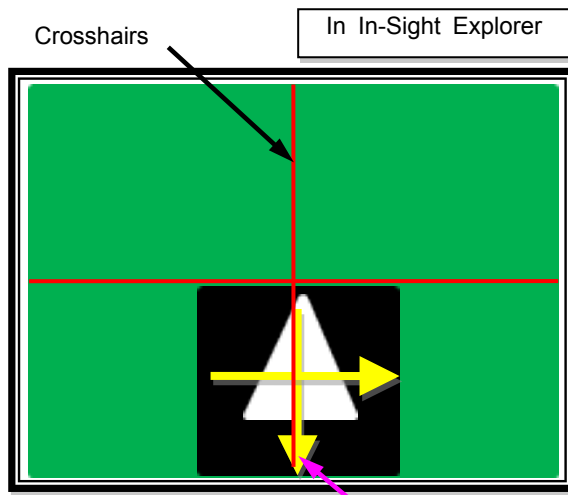


Fig. 4.9: Calibration screen 2 Tool center point

When the robot is moved in the +X, -X, +Y, and -Y directions, the pixel coordinates of the vision sensor will be converted into the tool coordinates.

4.1.4 Principal of how the robot identifies the workpiece using the hand eye

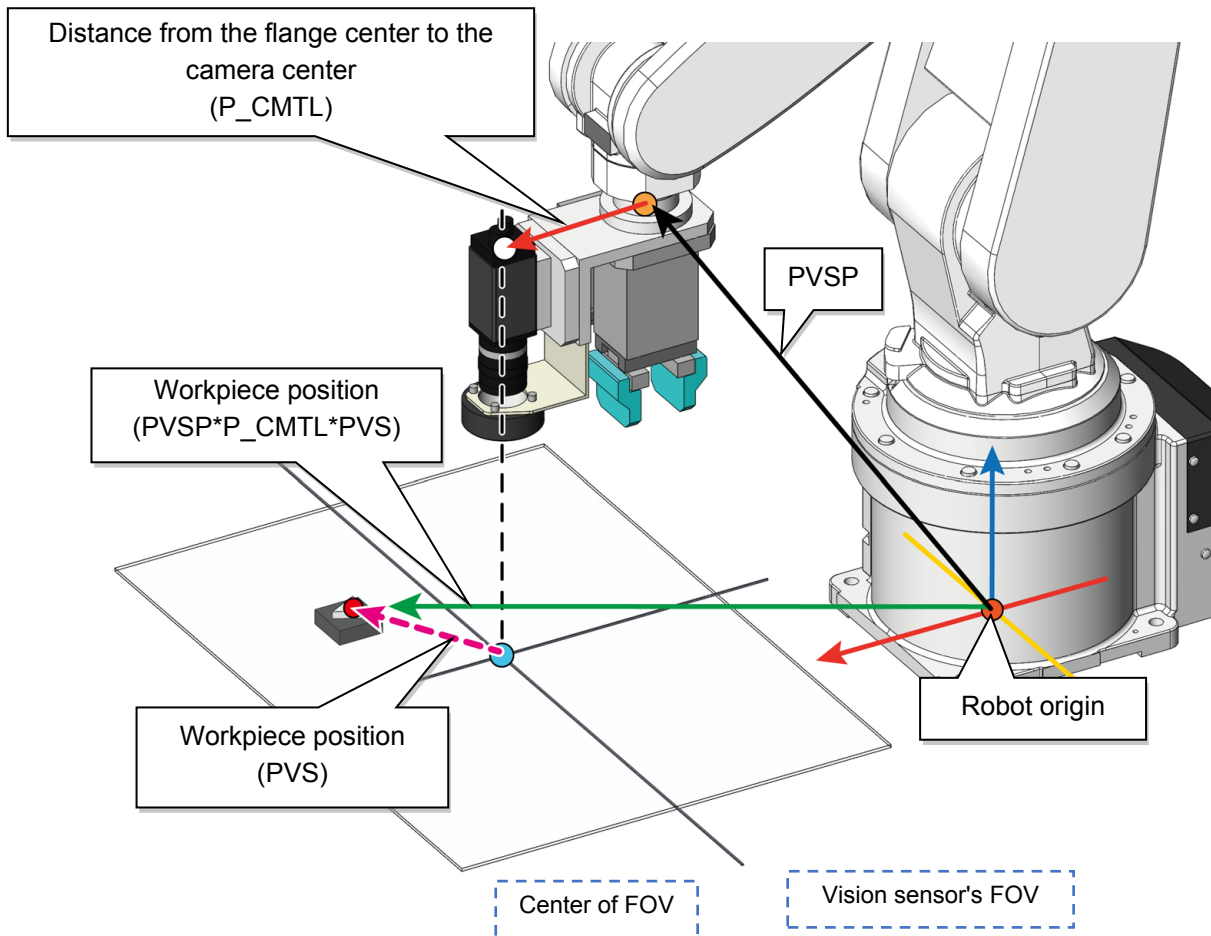


Fig. 4.10: Workpiece identification using the hand eye

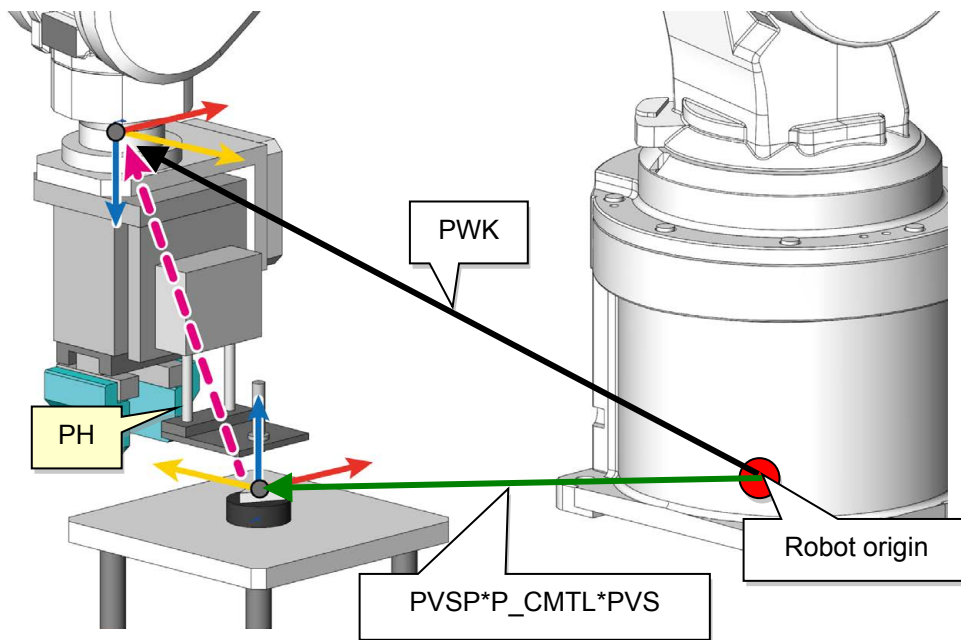


Fig. 4.11: Suction area on the master workpiece (PWK) and master workpiece position (PVS)

■ Process of setting up the hand eye

Below is the procedure of how to setup the hand eye. It covers everything from device connection to the automatic operation of a robot using 2D vision sensors. The Steps 1 to 4 in the following table are the same as those described in Chapter 2, "Communication settings". (*1)

Therefore, only Steps 5 to 11 will be explained in this chapter.

Step	Description	Remarks
1	Preparing the robot and vision sensor	Chapter 2 - Communication settings and lens adjustment
2	Configuring computer network settings	
3	Connecting the robot and camera (RT ToolBox3) *1	
4	Configuring camera communication settings (In-Sight Explorer)	

*1) The COM port communication settings used in this seminar differ depending on the camera. Refer to "** Cameras used in this seminar and COM port communication settings".



Step	Description	Main tasks
5	Setting the control point used for calibration	<ul style="list-style-type: none"> Adding a user base program Checking the camera image Positioning the workpiece used for calibration Adjusting the lens Creating the workpiece feature point Setting the control point used for calibration
6	Calibration	<ul style="list-style-type: none"> Measuring the camera's FOV Creating user-defined points Entering the size of the camera's FOV Fine-tuning user-defined points Carrying out N point calibration Setting the height between the workpiece identification point and the camera
7	Creating an identification job	<ul style="list-style-type: none"> Positioning the master workpiece Setting the workpiece to be identified and the identification range Selecting an identified pattern Saving the job
8	Setting up the relative position between the robot and the workpiece	-
9	Checking the robot movement using step operation and automatic operation	-

4.2 Setting the control point used for calibration using the hand eye

Set the center of the camera as the control point for calibration using the camera and robot programs.

4.2.1 Program for setting the control point used for calibration

The programs used in this chapter are UBP.prg and HND1.prg.

- UBP.prg: Used for defining user external variables.
- HND1.prg: Used for aligning the crosshairs in the center of the screen with the feature of the workpiece to set the control point for robot calibration.
Calculate the distance between the center of the camera and the center of the robot.

◇ Program UBP.prg

For details on the program, refer to 3.2.1 Program for setting a control point used for calibration.

◇ Program HND1.prg

```
1 '-----
2 ' Step 1. Program to adjust the hand camera
3 ' Camera center calculation program
4 '-----
5 Def Pos P_CMTL ' Hand camera: Camera center position data used for calibration
6 Def Pos P_CLPos ' Hand camera: Reference point to start calibration
7 Loadset 1,1
8 OAdl On
9 Servo On
10 Wait M_Svo=1
11 '---- HAND INIT ----
12 Wait M_Svo=1
13 If M_EHOrg=0 Then ' If hand has not returned to origin:
14   EHOrg 1 ' returns Hand 1 to origin.
15   Wait M_EHOrg=1 ' waits for Hand 1 to return to origin.
16 EndIf
17 EHOpen 1,100,100 ' Opens Hand 1 (speed = 100%, force = 20%).
18 Wait M_EHBusy=0 ' Checks if operation is complete.
19 '-----
20 PTool=P_Tool
21 ' Align the hand. Place the crosshairs drawn in the center of the screen over
22 ' the feature while checking the screen.
23 P0=P_Fbc
24 P91=P0*(+0.00,+0.00,+0.00,+0.00,+0.00,+90.00)
25 Mov P91
26 ' Align the crosshairs using the XYZ jog mode in the X and Y axes only.
27 P90=P_Fbc
28 PTL=P_Zero
29 PT=Inv(P90)*P0
30 PTL.X=(PT.X+PT.Y)/2
31 PTL.Y=(-PT.X+PT.Y)/2
32 P_CMTL=PTool*PTL
33 Tool P_CMTL
```

(continued on the next page→)


```
34 Hit
35 Tool PTool
36 Mov P0
37 Dly 0.5
38 Tool P_NTool
39 P_CLPos=P_Fbc
40 End
P_CMTL=(-97.85,-5.06,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(0,0)
P_CLPos=(+184.47,-178.00,+376.40,-180.00,+0.00,-178.77,+0.00,+0.00)(7,0)
PTool=(+0.62,-0.61,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(0,0)
P0=(+183.87,-178.62,+376.40,-180.00,+0.00,-178.77,+0.00,+0.00)(7,0)
P91=(+183.87,-178.62,+376.40,+180.00,+0.00,+91.23,+0.00,+0.00)(7,0)
P90=(+284.75,-82.43,+376.40,+180.00,+0.00,+91.23,+0.00,+0.00)(7,0)
PTL=(-98.47,-4.45,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(0,0)
PT=(-94.02,-102.91,+0.00,+0.00,+0.00,-90.00,+0.00,+0.00)(7,0)
```

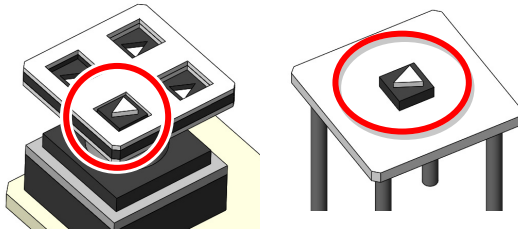
(End of HND1.prg)

4.2.2 Setting the control point used for calibration

■ Process of setting the control point used for calibration

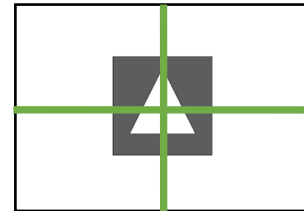
Step	Description
(1)	Adding a user base program (Not required if the program has already been added.)
(2)	Connecting a camera
(3)	Positioning the workpiece used for calibration
(4)	Adjusting the lens
(5)	Drawing two intersecting lines in In-Sight Explorer
(6)	Setting a control point used for calibration
(7)	Checking the control point used for calibration

(3) Position the workpiece used for calibration.



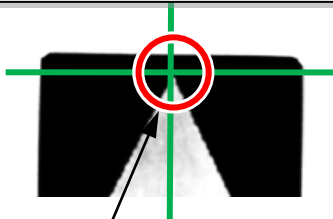
On the workpiece destination platform or workpiece suction platform

(5) Draw two intersecting lines.

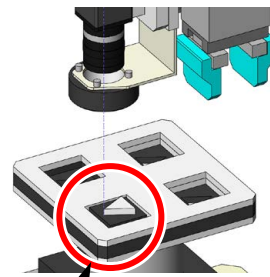


In-Sight Explorer

(6) Set a control point used for calibration.

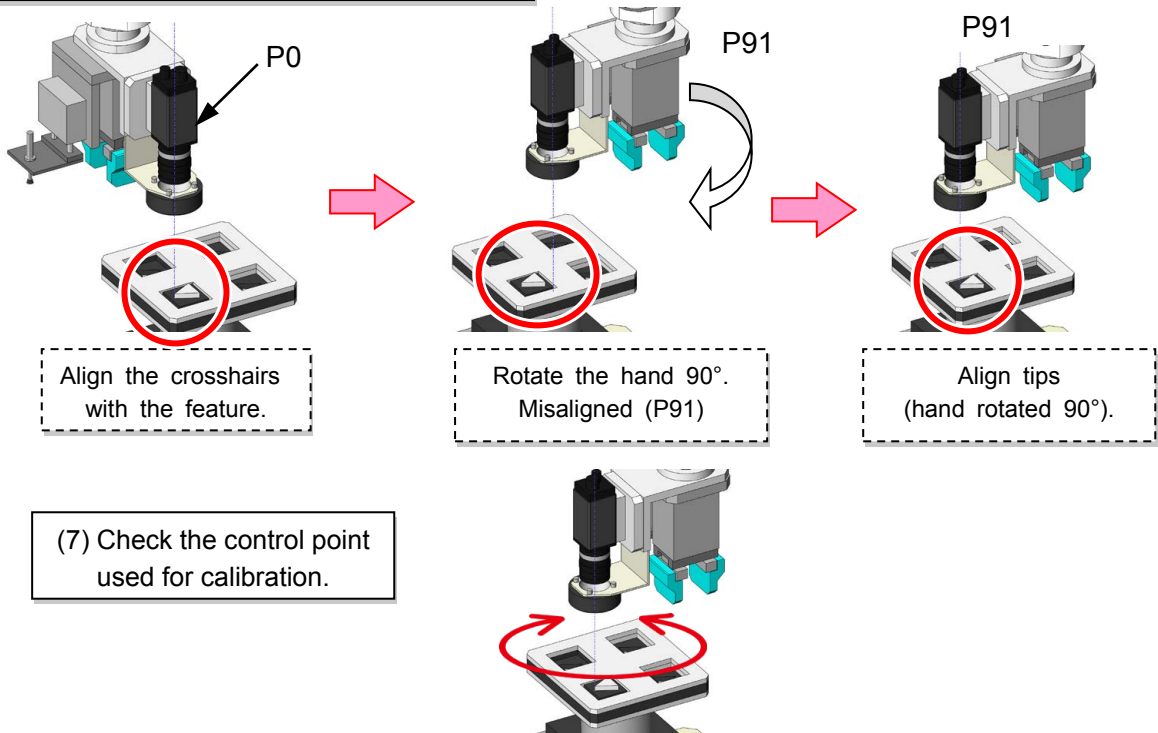


Move the robot to align the crosshairs with the feature.



If the crosshairs align with the feature in EasyBuilder View, this means that the center of the camera is aligned with the feature.

(6) Set the control point used for calibration.



(7) Check the control point used for calibration.

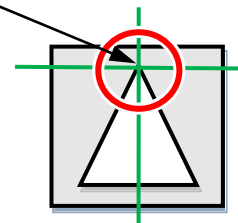
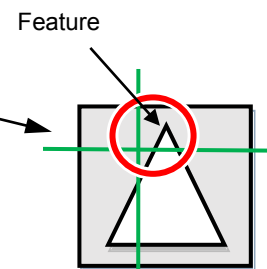
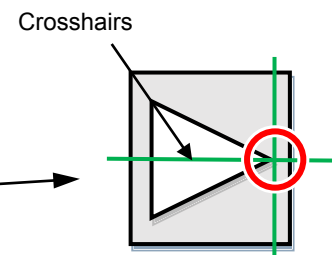
Program for setting the control point used for calibration

To calculate the control point used for calibration, draw intersecting lines in the center of EasyBuilder View, and then run the following lines of HND1.prg using step operation.

◇ Program HND1.prg

```

.....
20 PTool=P_Tool
21 ' Align the hand. Place the crosshairs drawn in the center
    of EasyBuilder View over
22 ' the feature while checking EasyBuilder View.
23 P0=P_Fbc
24 P91=P0*(+0.00,+0.00,+0.00,+0.00,+0.00,+90.00)
25 Mov P91 ' Rotate the hand 90° (misaligned).
26 ' Align the crosshairs using the XYZ jog mode in the
    X and Y axes only.
27 P90=P_Fbc
28 PTL=P_Zero
29 PT=Inv(P90)*P0
30 PTL.X=(PT.X+PT.Y)/2
31 PTL.Y=(-PT.X+PT.Y)/2
32 P_CMTL=PTool*PTL
33 Tool P_CMTL
34 Hit
  
```



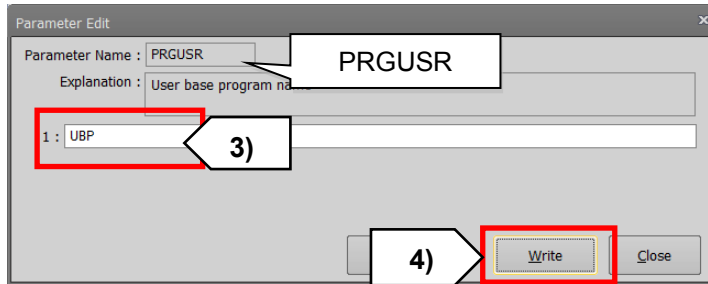
Align the feature with the crosshairs when the hand is positioned at 0° and 90°.

(1) Adding a user base program

(1.1) Add a user base program (UBP). (Not required if the program has already been added.)

Add a user base program in RT ToolBox3. This will set up variables which can be used in other programs.

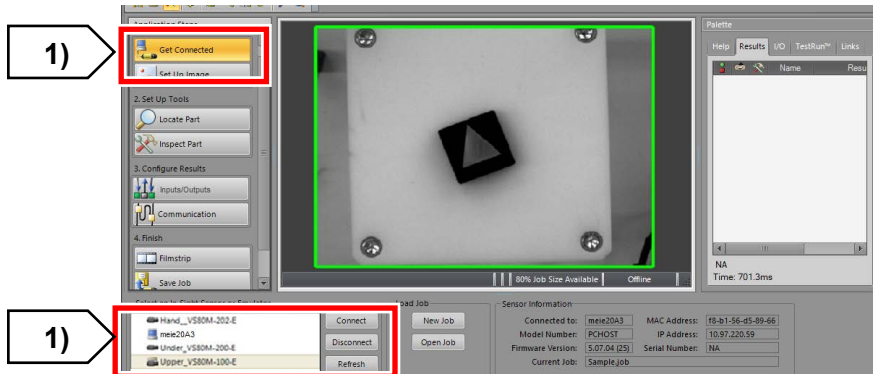
- 1) Go to Online → Parameter → Parameter List in the project tree.
- 2) Read (search for) parameter "PRGUSR". Then double click PRGUSR and enter UBP in the field shown below.
- 3) Click Write to write the parameter to the robot controller.
- 4) Restart the robot controller.



(2) Connecting a camera

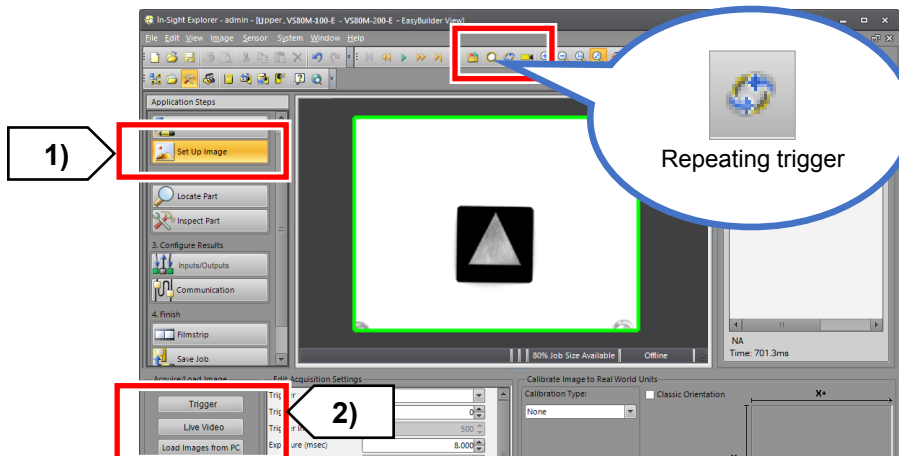
(2.1) Connect a camera.

- 1) Select Get Connected from the Application Steps window. Select the hand eye camera and click Connect.



(2.2) Acquire an image.

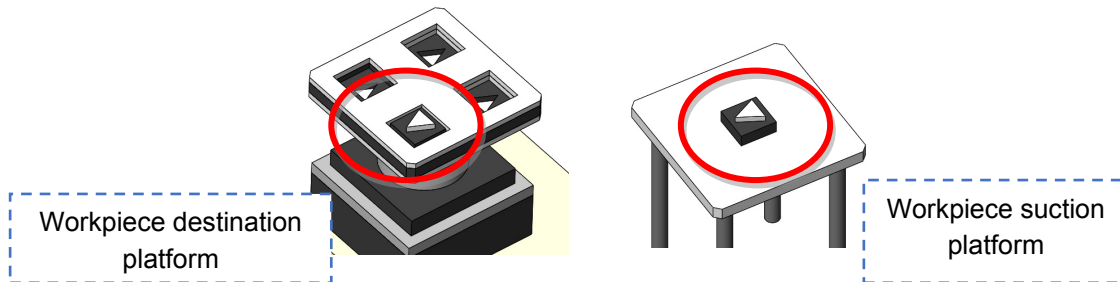
- 1) Select Set Up Image from the Application Steps window.
- 2) Select Live Video from Acquire/Load Image. Or, click the Repeating trigger button on the tool bar. Check that the image can be seen in EasyBuilder View.



(3) Positioning the workpiece used for calibration

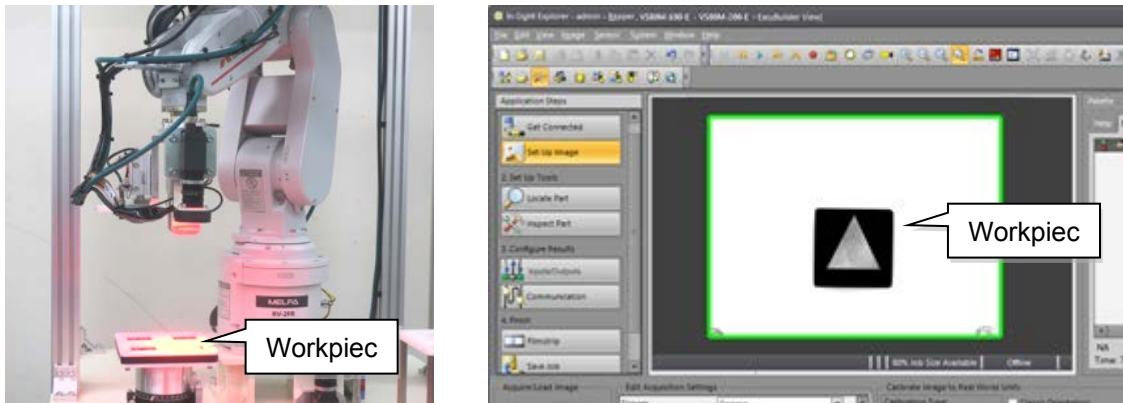
(3.1) Position the workpiece used for calibration.

- 1) Place a workpiece on a surface parallel to the surface the robot is on.



(3.2) Locate the hand camera above where the workpiece is picked.

- 1) Align the hand.
- 2) Move the robot using Jog operation so that the workpiece shown in Step 3.1 can be located near the center of the camera's FOV.
The height of the camera should be appropriate for your purpose.



(4) Adjusting the lens

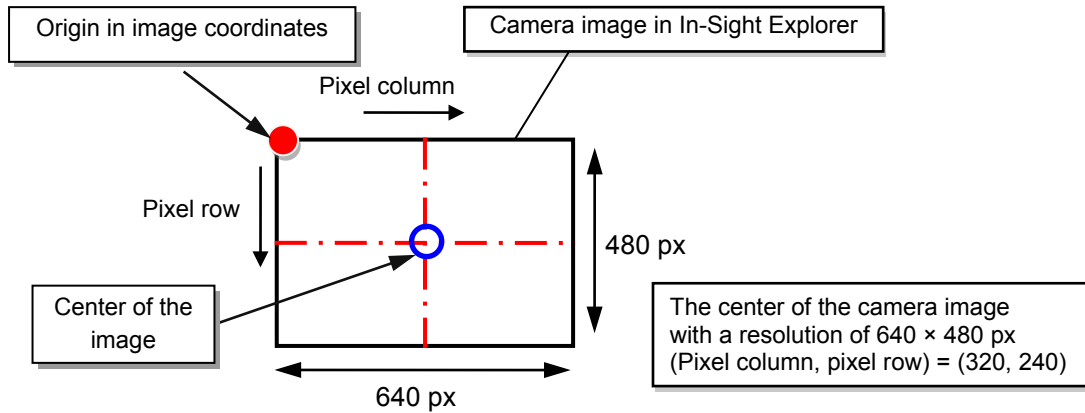
(4.1) Adjust the focus and aperture of the camera.

- 1) Select Live Video from Acquire/Load Image, or click the Repeating trigger button on the tool bar.
- 2) Adjust the focus and aperture to get a clear image of the workpiece suction point.
Refer to 2.5 Adjusting the lens (focus and aperture) for information on how to adjust the lens.

(5) Drawing two intersecting lines in In-Sight Explorer

To configure tool settings and carry out calibration, draw two intersecting lines in the center of the screen in In-Sight Explorer.

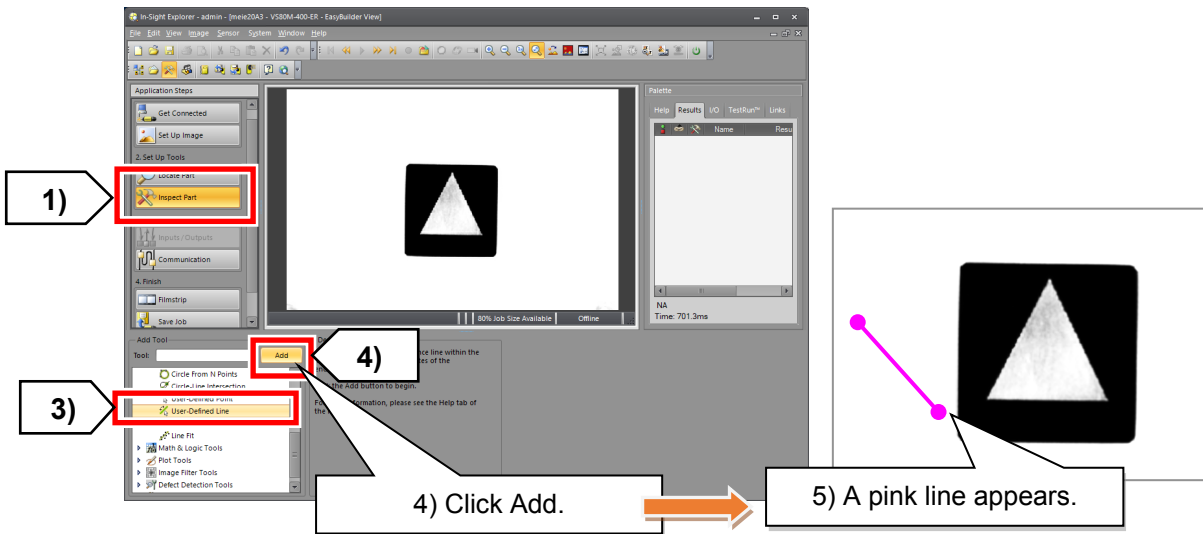
* The center of the image shown below is for the camera with a resolution of 640x480 pixels.



(5.1) Draw two intersecting lines in the center of EasyBuilder View.

* Select Manual from the Trigger drop-down box under Edit Acquisition Settings.

- 1) Select Inspect Part from the Application Steps window.
- 2) If the Repeating trigger is enabled, disable it.
- 3) Select User-Defined Line from Geometry Tools in the bottom left Add Tool window.
- 4) Click Add.
- 5) A pink line will appear on the image.



Additional

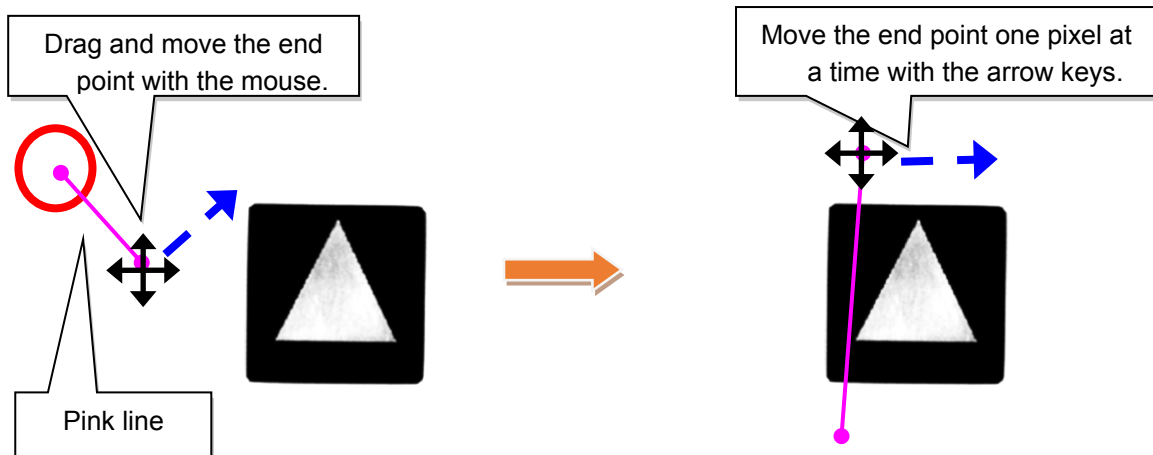
Drawing lines

Selecting Line from Plot Tools also allows you to draw lines on EasyBuilder View.

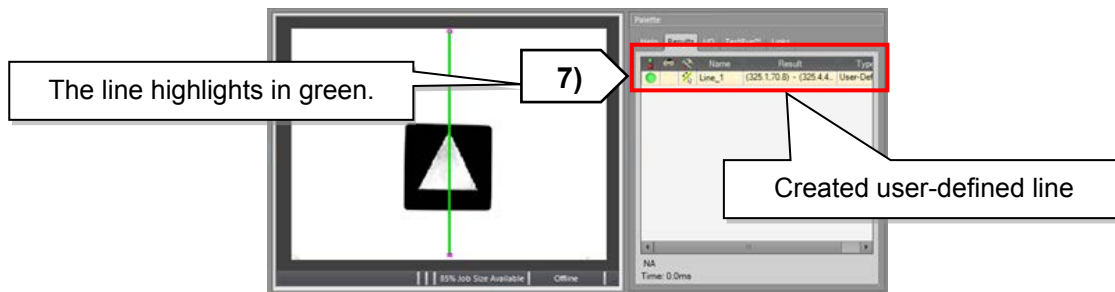
- 1) Select Inspect Part from the Application Steps window.
- 2) Select Line from Plot Tools in the bottom left Add Tool window.

6) Move both end points of the pink line with the mouse and draw a vertical (or horizontal) line in the center of the screen.

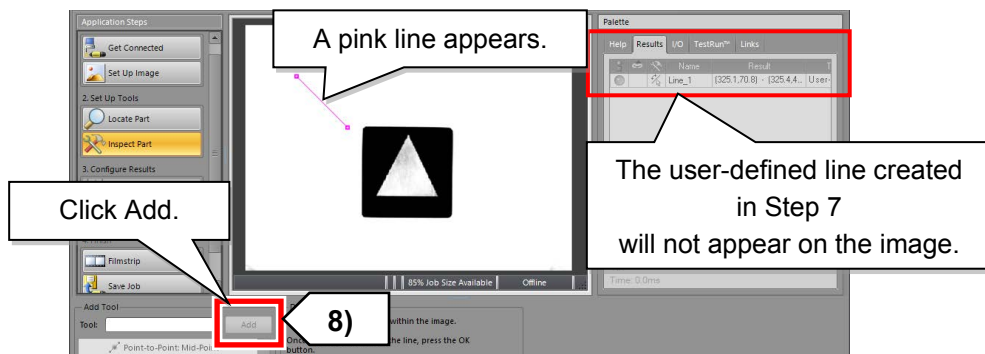
* Clicking an end point of the user-defined line and pressing an arrow key on the key board will move the end point one pixel at a time.



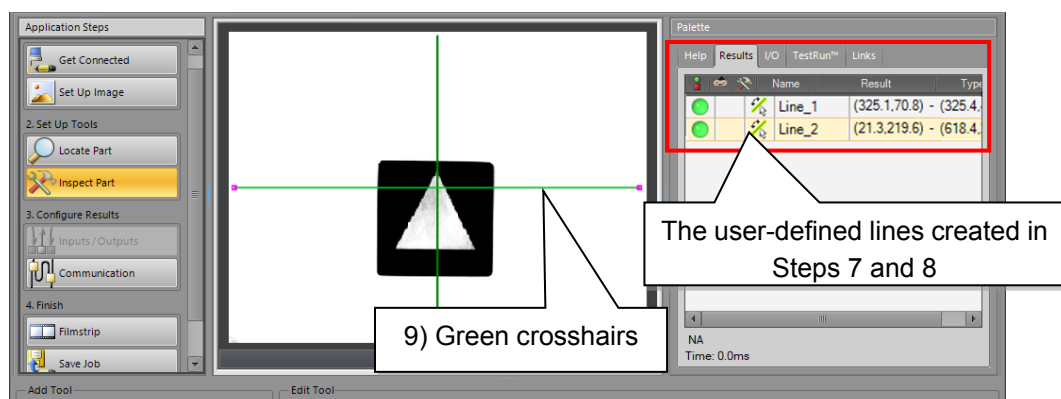
7) Click OK. The line will turn green, and be added to the Palette shown on the right.



8) To make crosshairs, draw a horizontal (or vertical) line by using Steps 3 to 6.



9) Click OK to create user-defined lines (green crosshairs).



(6) Setting the control point used for calibration

(6.1) Move the robot to align the crosshairs created in Step 2 with the feature of the workpiece.

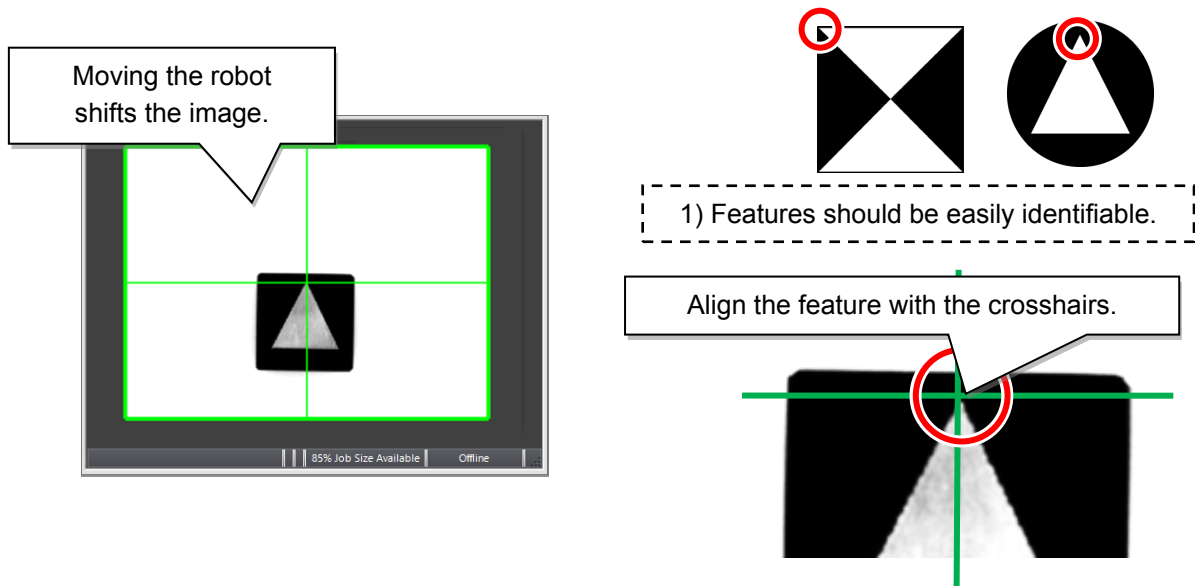
* Select Continuous from the Trigger drop-down box under Edit Acquisition Settings.

- 1) Determine which part the feature of the workpiece is.
- 2) Open HND1.prg.
- 3) Run the program up to line 20 using step operation.

- ◇ In line 14, press and hold the **F1** key until the hand operation completes. (Initialization of the electric hand)
- ◇ In line 20 "PTool = P_Tool", the current robot tool data is assigned to PTool.

4) Move the robot in the X, Y, and C axes in XYZ jog mode while checking EasyBuilder View to align the crosshairs with the feature.

* With the enlarged image, move the robot so that the crosshairs can be aligned with the feature as accurately as possible.

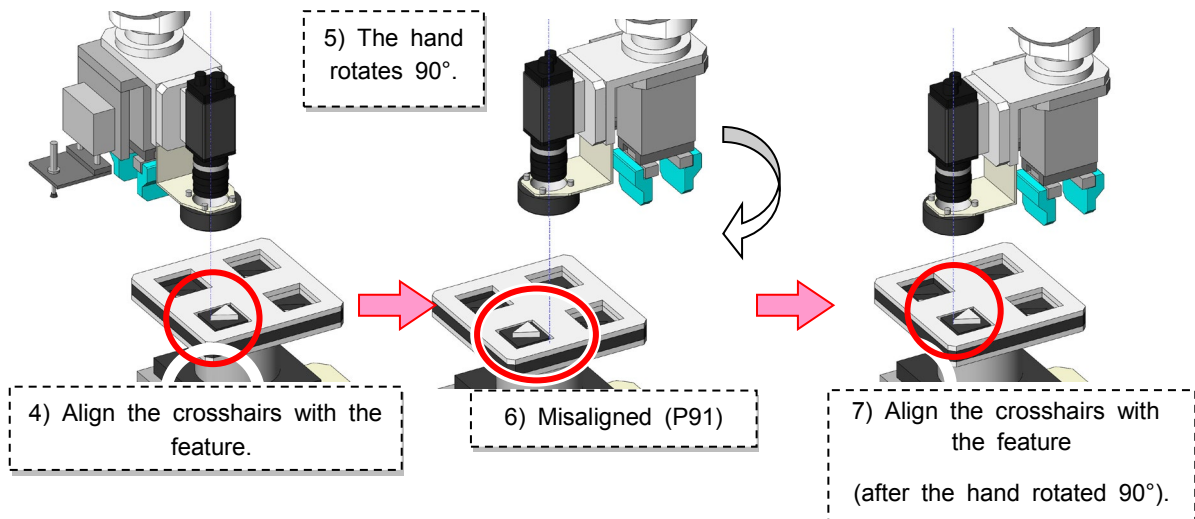


5) After the alignment, run HND1.prg up to line 26 using step operation.

- ◇ In line 25 "Mov P91", the hand rotates 90°.

6) In EasyBuilder View, after the workpiece has rotated 90°, the crosshairs will be misaligned with the feature.

7) Move the robot using the XYZ jog mode in the X and Y axes only to align the crosshairs with the feature again as shown in Step 3.



(7) Checking the control point used for calibration

(7.1) Check the robot movement.

1) Run HND1.prg up to line 34 using step operation.

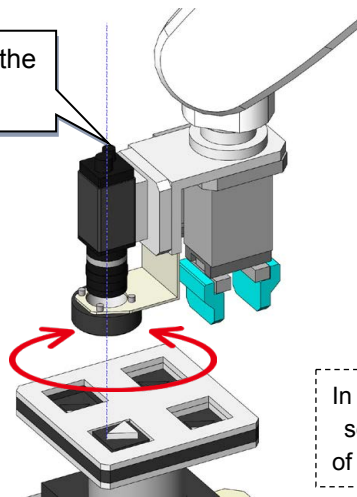
- ◇ In lines 29 to 32, the control point used for calibration is set at the center of the camera.
- ◇ In line 33 "Tool = P_CMTL", the robot tool is set at the calibration control point.

2) Rotate the C axis in Tool jog mode to check that the crosshairs are aligned with the feature at all times in EasyBuilder View.

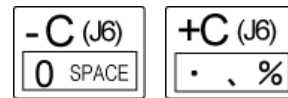
* The control point used for calibration is set at the center of the camera as long as the crosshairs align with the feature while the C axis rotates.

* If a misalignment occurs, perform Step 6 again.

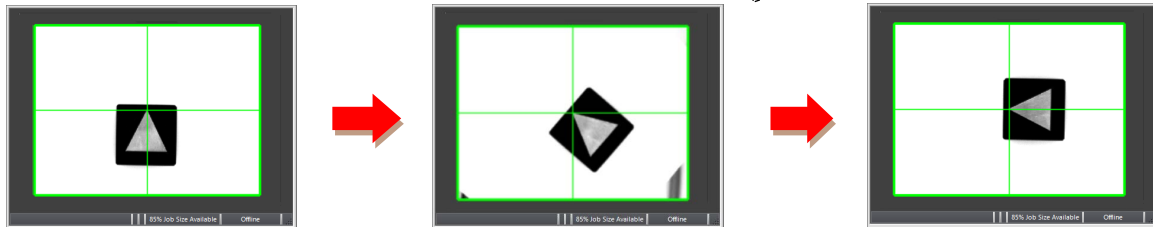
2) Revolve the hand around the center of the camera.



Tool jog



In EasyBuilder View, the workpiece seems to spin around the center of the crosshairs.



The crosshairs should align with the feature at all times in EasyBuilder View while the C axis rotates.

3) After checking that the alignment is correct, run the program up to line 40 using step operation.

- ◇ In line 35 "Tool PTool", the robot tool is reset.
- ◇ In line 36 "Mov P0", the robot returns to the original position.

4.3 Calibration using the hand eye

4.3.1 Calibration program

Use the camera and robot program to carry out calibration.

The program used in this chapter is HND2.prg.

HND2.prg : HND2.prg: Used for calibration by entering the size of the camera's FOV and aligning user-defined points with crosshairs on the workpiece after the robot has moved.

◇ Program HND2.prg

```
1 '-----
2 ' Step 2. Program to adjust the hand camera
3 ' Calibration assistance program HND2.prg
4 '-----
5 Def Pos P_CMTL ' Hand camera: Camera center position data used for calibration
6 Def Pos P_CLPos ' Hand camera: Reference point to start calibration
7 Def Pos P_CMH ' Hand camera: Offset from the workpiece suction point to the imaging point
8 Loadset 1,1
9 OAdl On
10 Servo On
11 Wait M_Svo=1
12 '---- HAND INIT ----
13 Wait M_Svo=1
14 If M_EHOrg=0 Then ' If hand has not returned to origin:
15   EHOrg 1 ' returns Hand 1 to origin.
16   Wait M_EHOrg=1 ' waits for Hand 1 to return to origin.
17 EndIf
18 EHOpen 1,100,100 ' Opens Hand 1 (speed = 100%, Force = 20%).
19 Wait M_EHBusy=0 ' Checks if operation is complete.
20 '-----
21 PTool=P_CMTL
22 Tool P_NTool
23 P0=P_CLPos
24 Mov P0
25 ' Place the crosshair in the center of the camera over the crosshair on the jig within the camera's
FOV.
26 ' Align the user-defined points with the crosshairs.
27 ' Put the camera in "Live mode". Then after moving the robot in Tool jog mode,
28 ' find the points in the +X, -X, +Y, and -Y axes where the tool can move to without the crosshair on
the jig leaving the camera's FOV.
29 ' Set these points in the fields for travel amount below. Set the first point as the center of the image
and set the XY world coordinates to 0.
30 ' Enter the reciprocal of the points as world coordinates into the camera calibration tool.
31 ' (*Convert the camera coordinates into tool coordinates)
32 Mov P0*(-30.00,+0.00,+0.00,+0.00,+0.00,+0.00)
```

(continued on the next page→)

```

33 ' Align the feature of the workpiece with the center of the user-defined point to which the workpiece
moved.
34 Mov P0
35 Mov P0*(+30.00,+0.00,+0.00,+0.00,+0.00,+0.00)
36 ' Align the feature of the workpiece with the center of the user-defined point to which the workpiece
moved.
37 Mov P0
38 Mov P0*(+0.00,-40.00,+0.00,+0.00,+0.00,+0.00)
39 ' Align the feature of the workpiece with the center of the user-defined point to which the workpiece
moved.
40 Mov P0
41 Mov P0*(+0.00,+40.00,+0.00,+0.00,+0.00,+0.00)
42 ' Align the feature of the workpiece with the center of the user-defined point to which the workpiece
moved.
43 ' Carry out calibration with the calibration tool.
44 Mov P0
45 Hlt
46 ' Touch the hand onto the surface of the workpiece.
47 PSur=P_Fbc
48 P_CMH=Inv(PSur)*P0
49 '
50 End
P_CMTL=(-97.85,-5.06,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(0,0)
P_CLPos=(+184.47,-178.00,+376.40,-180.00,+0.00,-178.77,+0.00,+0.00)(7,0)
P_CMH=(+88.38,-90.49,-140.90,+0.00,+0.00,+0.00,+0.00,+0.00)(7,0)
PTool=(-97.85,-5.06,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(0,0)
P0=(+184.47,-178.00,+376.40,-180.00,+0.00,-178.77,+0.00,+0.00)(7,0)
PSur=(+270.89,-85.64,+235.50,-180.00,+0.00,-178.77)(7,0)
P101=(+186.55,-174.60,+376.40,-180.00,+0.00,-178.77)(7,0)

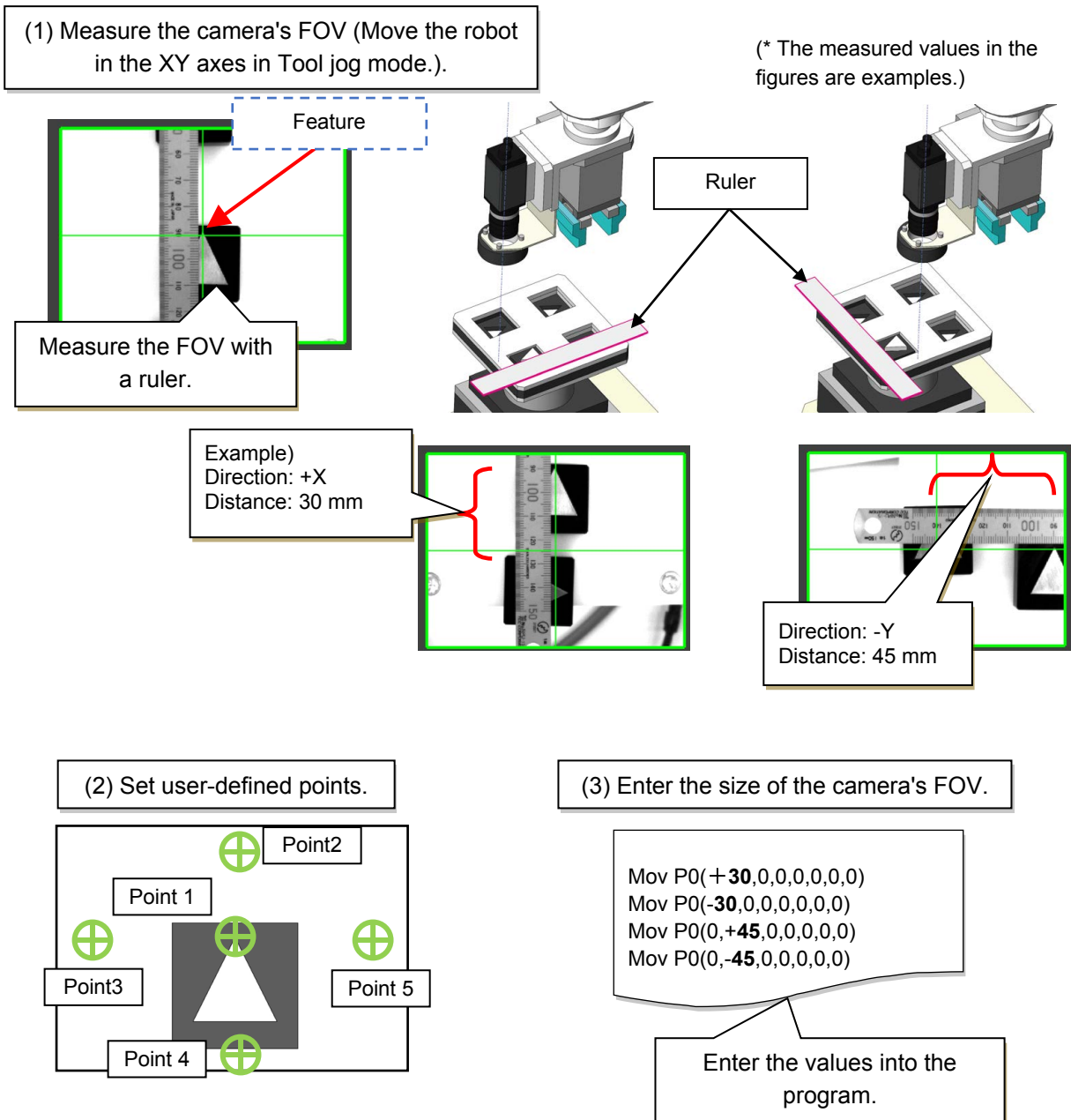
```

(End of HND2.prg)

4.3.2 Calibration method

■ Calibration process

Step	Description
(1)	Measuring the camera's FOV
(2)	Setting user-defined points
(3)	Entering the size of the camera's FOV
(4)	Fine-tuning user-defined points
(5)	Creating N points
(6)	Carrying out N point calibration
(7)	Setting the height between the surface of the workpiece to be identified and the camera center

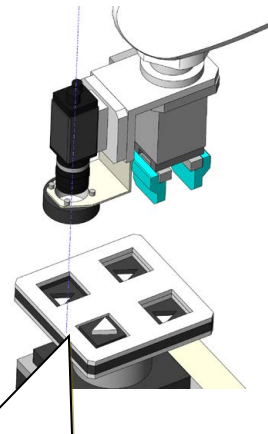
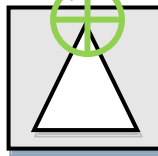


(4) Fine-tune user-defined points.

Mov P0(+30,0,0,0,0,0,0)
 Mov P0(-30,0,0,0,0,0,0)
 Mov P0(0,+45,0,0,0,0,0)
 Mov P0(0,-45,0,0,0,0,0)

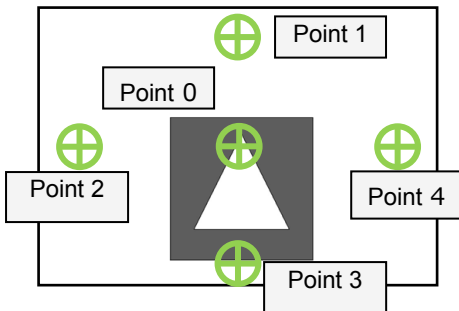
Run the program using step operation one point at a time.

Align the feature with the user-defined point.



Mov P0(+30,0,0,0,0,0,0)

(5) Create N



N points matched up with the camera coordinates

Camera coordinates
 Point 0 (x, y)
 Point 1 (x, y)
 Point 2 (x, y)
 Point 3 (x, y)
 Point 4 (x, y)

Points are named in the order they are selected.
 Point 0 (first selected point),
 point 1 (second selected point).....

(6) Carry out N point calibration.

Camera coordinates → robot coordinates
 Point 0 (x, y) → Point 0 (X, Y)
 Point 1 (x, y) → Point 1 (X, Y)
 Point 2 (x, y) → Point 2 (X, Y)
 Point 3 (x, y) → Point 3 (X, Y)
 Point 4 (x, y) → Point 4 (X, Y)

Enter the size of the camera's FOV into x and y (World X, world Y)

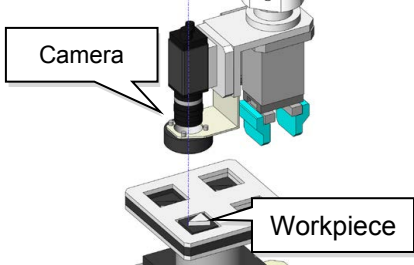
(1) Measuring the camera's FOV

1) Measure the distances in the +X, -X, +Y, and -Y axes in which the arm can move without the crosshairs on the jig leaving the camera's FOV.

* Move the robot in Tool jog mode.

(1.1) Locate the hand camera above the workpiece pick up point.

1) Locate the hand camera above the workpiece pick up point.



(1.2) Measure the distances in the X or Y axis in which the arm can move without the feature leaving the camera's FOV.

* Select Continuous from the Trigger drop-down box under Edit Acquisition Settings.

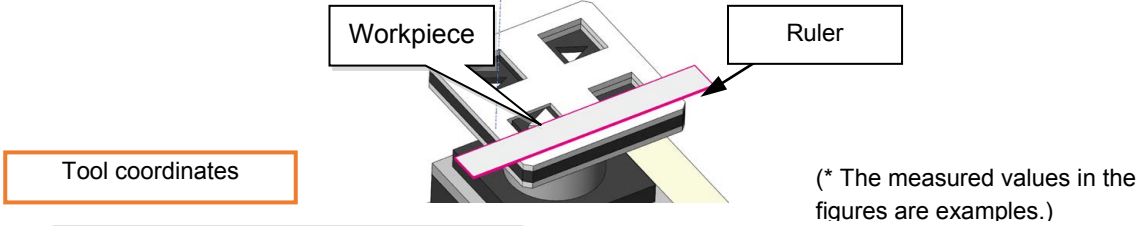
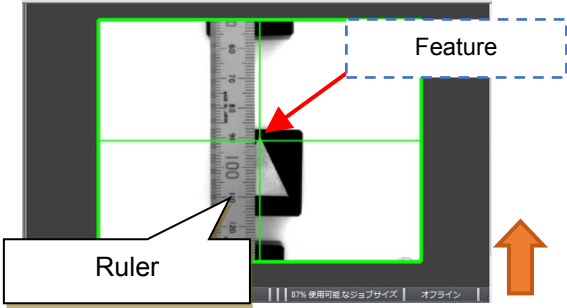
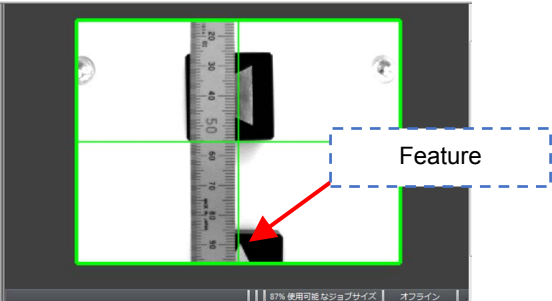
1) Put the ruler in the direction of the X or Y axis of the robot tool coordinates beside the feature set in Step 4.2.2 (6).

2) Move the robot in Tool jog mode along the ruler in the +X or +Y direction.
The hand and camera move together, and thereby the camera image in In-Sight Explorer changes according to the movement.

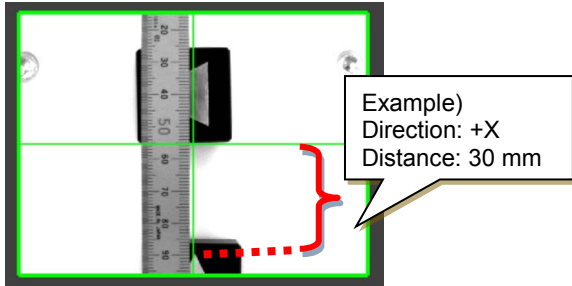
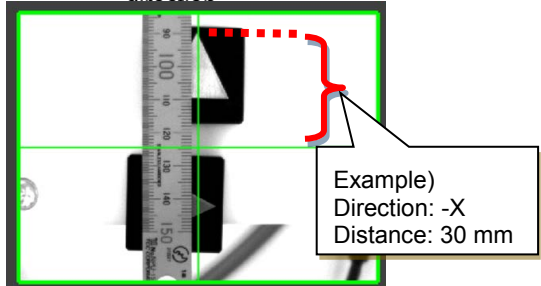
3) Stop the robot at a point where the feature placed in the center of the crosshairs does not leave the camera's FOV by checking EasyBuilder View.

4) Write down the distance and in which direction the feature has moved in EasyBuilder View.

5) Move the robot in Tool jog mode in the opposite direction and write down the distance in which the feature does not leave the camera's FOV.

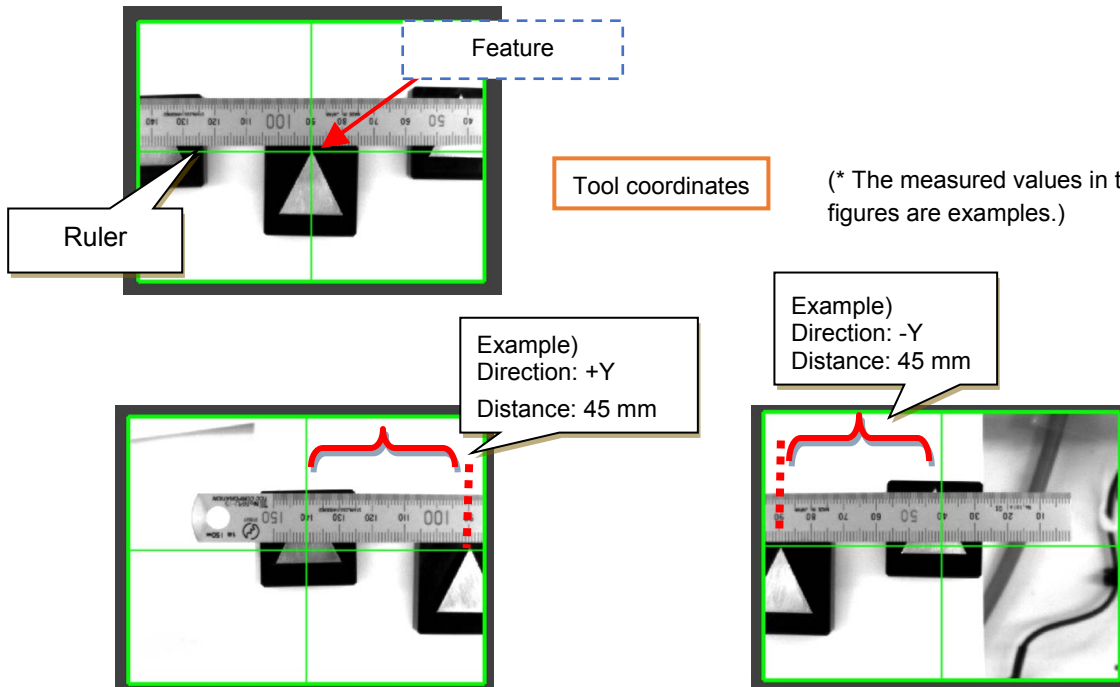




2) When the robot is moved in the arrow direction

(1.3) Measure the distance in the other direction (Y or X direction) in which the arm can move without the feature leaving the camera's FOV.

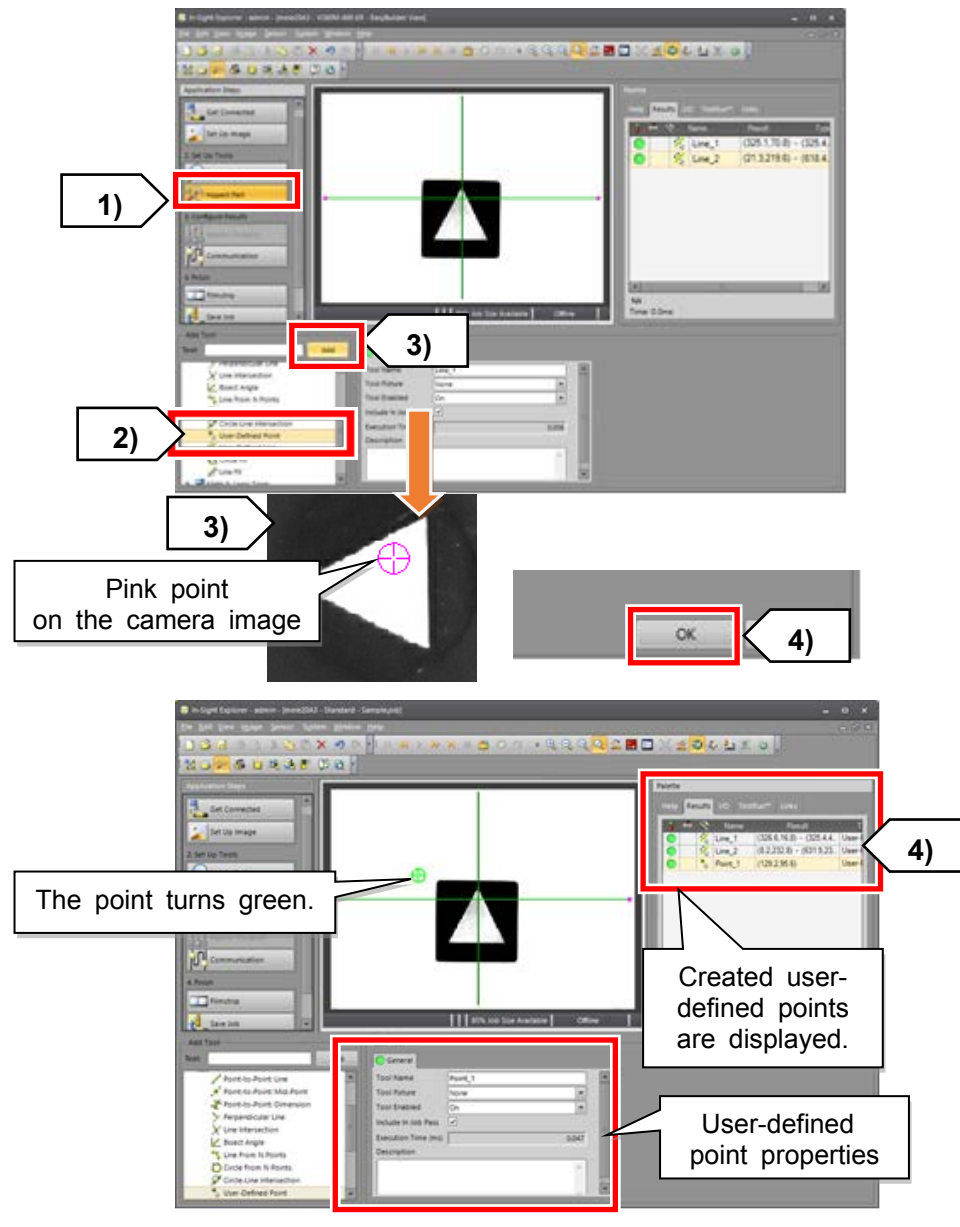
1) Put the ruler in the opposite direction. Repeat Step 1.2 and write down the distance in the Y direction (X direction) in which the arm can move without the feature leaving the camera's FOV. Steps 1.2 and 1.3 allow you to measure four distances (+X, -X, +Y, -Y).



(2) Setting user-defined points

(2.1) Create user-defined points.

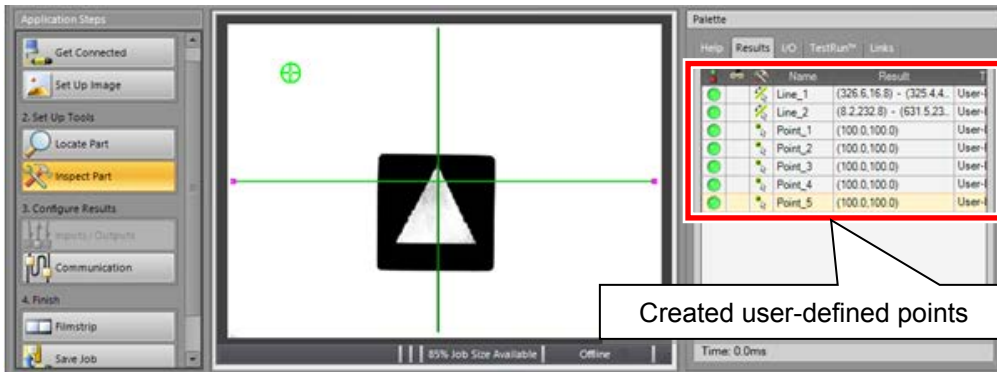
- 1) Select Inspect Part from the Application Steps window.
- 2) Select User-Defined Point from Geometry Tools in the bottom left Add Tool window.
- 3) Clicking Add will create a pink point on the camera image.
- 4) Clicking OK will add the user-defined point (Point_1) to the Palette on the right. The pink point will turn green. (Relocate the point later.)



(2.2) Create five user-defined points.

1) Create the five points using Steps 3 and 4 in section 2.1.

The created user-defined points will highlight in green on the camera image and added to the Palette. (In the Palette, the user-defined points will be listed in the order they were created.)



Created user-defined points

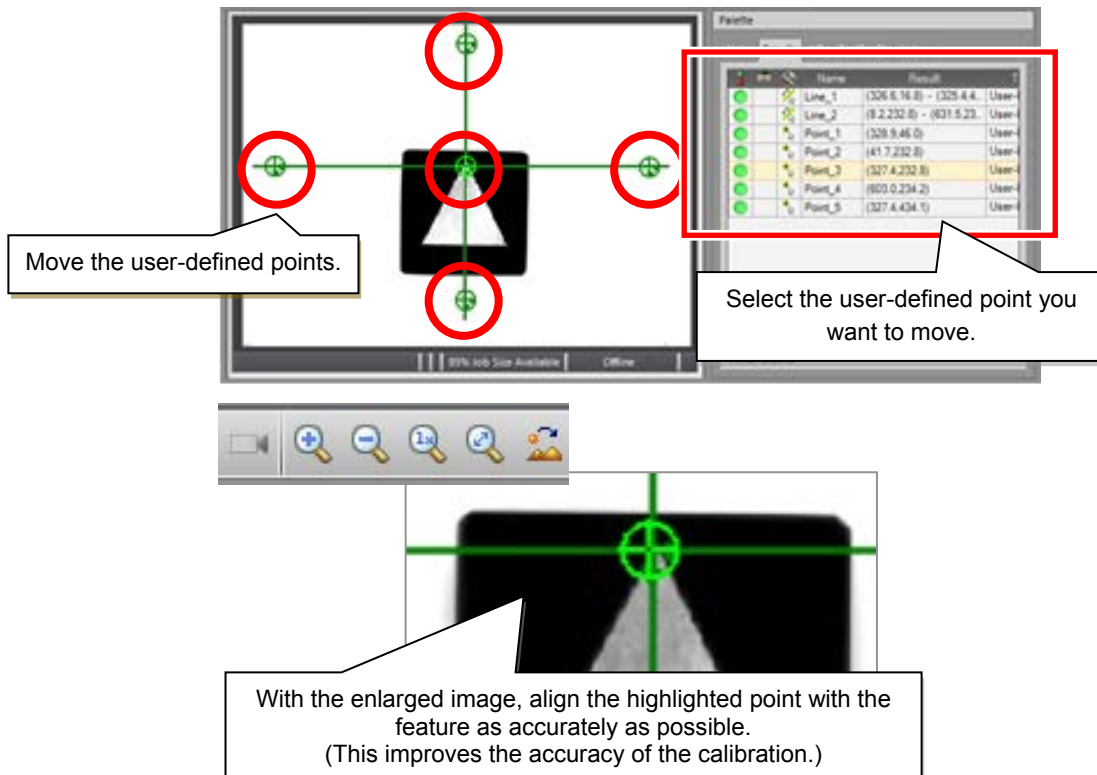
(2.3) Relocate user-defined points.

Move the created five user-defined points to the following positions on the crosshairs: center crosshair, +X axis, -X axis, +Y axis, and -Y axis.

* To fine-tune the positions of the user-defined points, perform step 4 "Fine-tune user-defined points".

- 1) Select Point_1 from the Palette. The corresponding point will highlight in green on the camera image.
- 2) Drag the highlighted point with the mouse to the center of the screen.
- 3) Move the other user-defined points "Point_2" to "Point_5" to the following positions on the crosshairs: +X axis, -X axis, +Y axis, and -Y axis.

* It is helpful to remember where Point 1 to Point 5 have been located.



Move the user-defined points.

Select the user-defined point you want to move.

With the enlarged image, align the highlighted point with the feature as accurately as possible. (This improves the accuracy of the calibration.)

(3) Entering the size of the camera's FOV

(3.1) Enter the values into the program to carry out calibration.

- 1) Open HND2.prg.
- 2) Enter the -X value measured in "(1) Find the camera's FOV" in the X component of line 32.
- 3) Do the same for +X of line 35, -Y of line 38, and +Y of line 41.
- 4) Save HND2.prg in the robot.

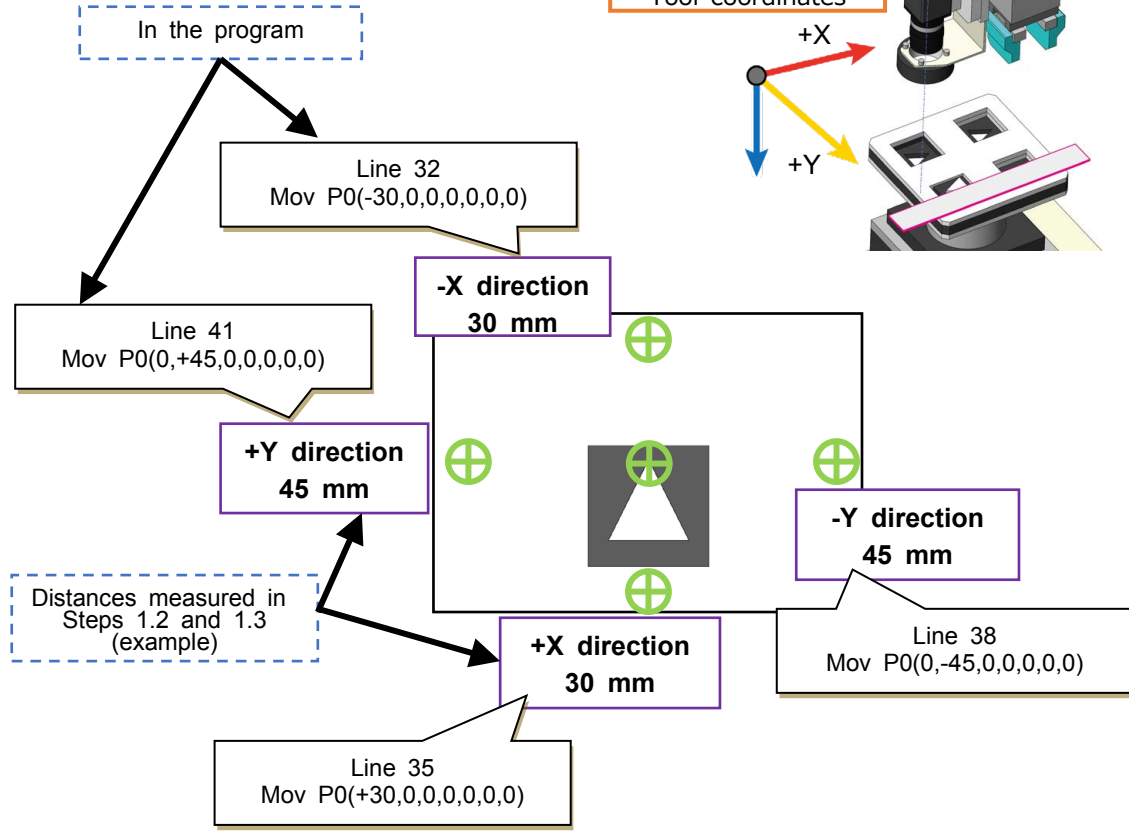
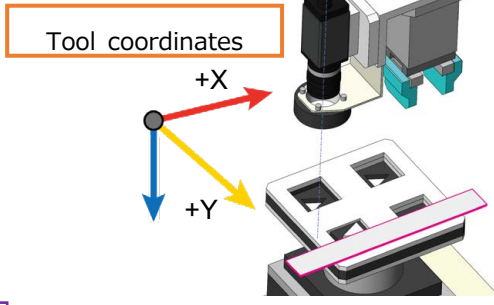
```

1 ' Step 2. Program to adjust the hand camera
2 ' Calibration assistance program HND2.prg
3
4 Def Pos P_CMTL ' Hand camera: Camera center position
5 Def Pos P_CLFPos ' Hand camera: Reference point to
6 Def Pos P_CMH ' Hand camera: Offset from the work
7 Loadset P_1
8 QAdt On
9 Servo On
10
11 Wait M_Svo=1
12 '----- HAND INIT
13 Wait M_Svo=1
14 If M_EHOrg=0 Then
15   EHOrg :
16   Wait M_EHOrg=1
17 Endif
18 EHOpen 1,100,100
19 Wait M_EHBusy=0
20 '-----
21 PTool=P_CMTL
22 Tool_P_NTool
23 P0=P_CLFPos
24 Mov P0
25 ' Place the crosshair
26 ' Align the user-de
27 ' Put the camera in
28 ' find the points in
29 ' Set these points in the fields for travel amount below. Set
30 ' Enter the reciprocal of the points as world coordinates into
31 ' (*Convert the camera coordinates into tool coordinates)
32 Mov P0* (-30.00,+0.00,+0.00,+0.00,+0.00,+0.00)
33 ' Align the feature
34 Mov P0
35 Mov P0* (+30.00,+0.00,+0.00,+0.00,+0.00,+0.00)
36 ' Align the feature
37 Mov P0
38 Mov P0* (0,-45.00,+0.00,+0.00,+0.00,+0.00)
39 ' Align the feature of the workpiece with the center of the use
40 Mov P0
41 Mov P0* (0,+45.00,+0.00,+0.00,+0.00,+0.00)
42 ' Align the feature of the workpiece with the center of the use
43
44 '-----
45 '
46 '
47 '
48 '
49 '
50 '
51 '
52 '
53 '
54 '
55 '
56 '

```

Line 32 (when the -X value is 30)
Mov P0(-30,0,0,0,0,0)

Line 35 (when the +X value is 30)
Mov P0(+30,0,0,0,0,0)



(4) Fine-tuning user-defined points

(4.1) Run HND2.prg.

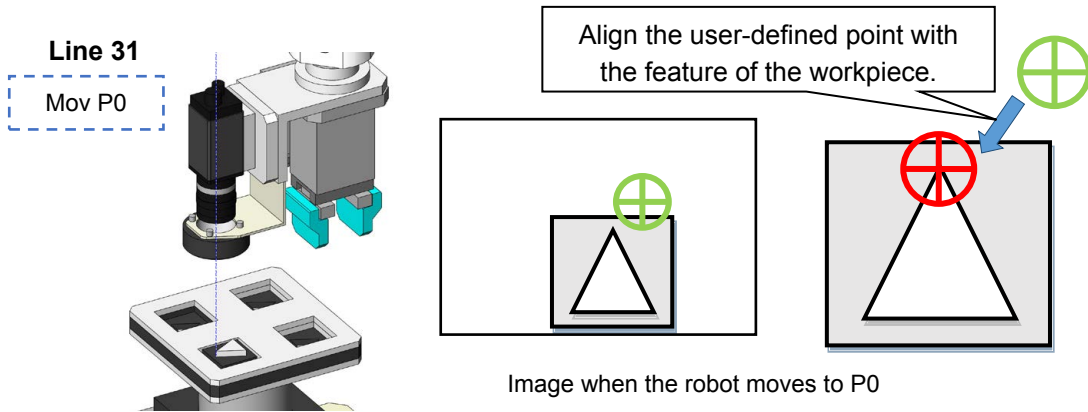
1) Run the program up to line 31 using step operation.

- ◇ In line 15, press and hold the F1 key until the hand operation completes. (Initialization of the electric hand)
- ◇ In line 24, move the robot to the reference point to start calibration (set in the program HND1.prg).

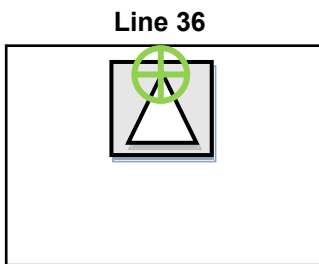
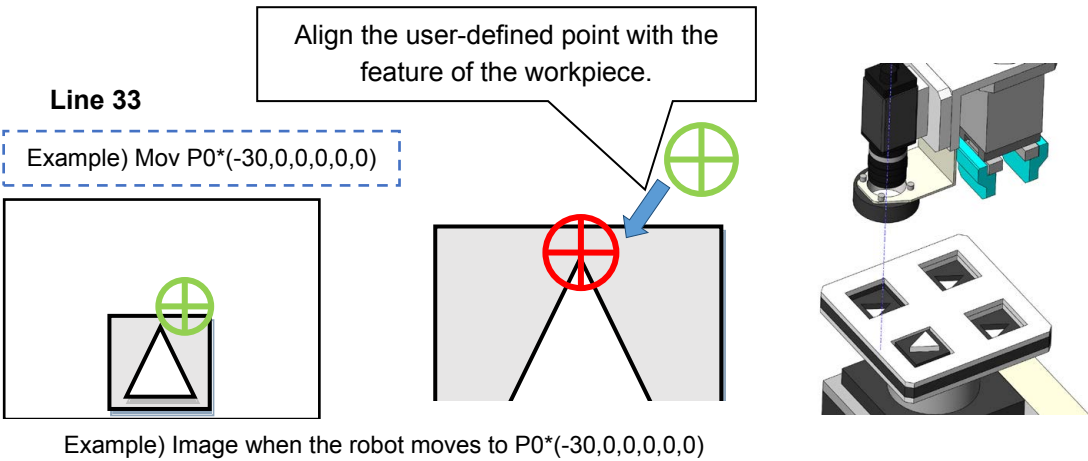
(4.2) Take an image of the jig and fine-tune the positions of the user-defined points.

1) In In-Sight Explorer, press the F5 key (trigger) on the keyboard to take an image. The camera image will be updated.

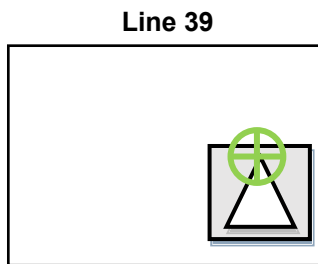
2) Align the user-defined point near the crosshairs with the feature of the workpiece. (Fine-tune the position of the points with the mouse or arrow keys.)



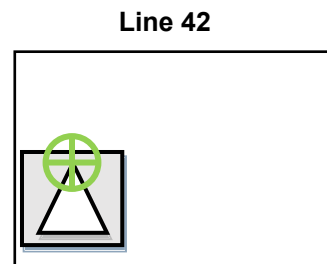
3) Run the program up to lines 33, 36, 39, and 42 using step operation, and perform Steps 1 and 2 in each line. Align the user-defined points with the feature (five points in total).



Example) Mov P0*(+30,0,0,0,0,0)



Example) Mov P0*(0,-45,0,0,0,0)



Example) Mov P0*(0,+45.0,0,0,0,0)

(5) Creating N points



In the process of creating N points, we will move the robot arm to each user-defined point.
However, calibration will not complete if the user-defined points and the robot

(5.1) Create N points.

* Select Manual from the Trigger drop-down box under Edit Acquisition Settings.

1) Select Inspect Part from the Application Steps window.

2) Select N Point from Calibration Tools in the bottom left Add Tool window.

3) Click Add. All the user-defined points on the camera image will highlight in green.

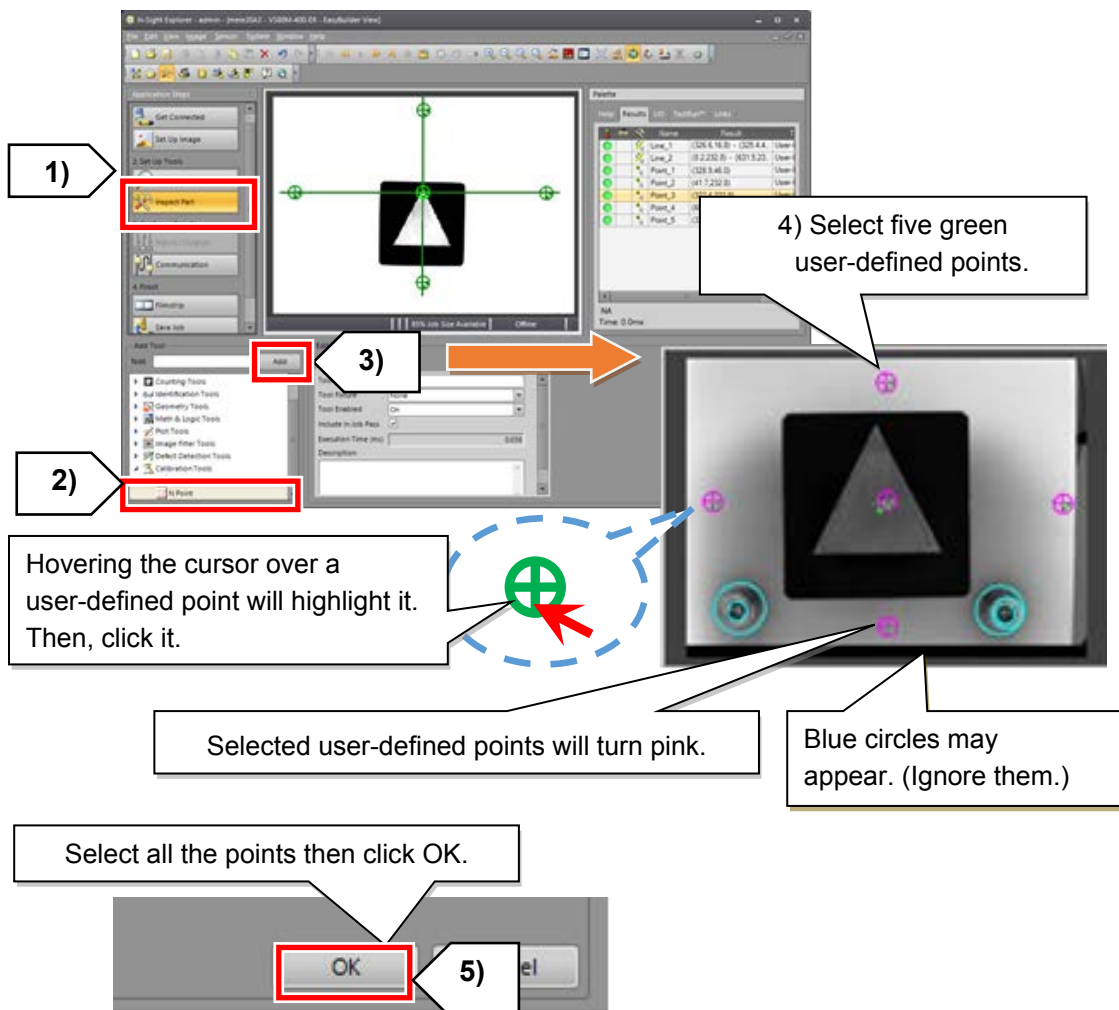
* Clicking Add may highlight some parts in blue. Ignore them. (Blue circles may appear depending on the image.)

4) Left-click the five user-defined points.

Selected user-defined points will turn pink. (Refer to the following image for how to select user-defined points.)

* It is helpful to remember in what order the user-defined points have been selected.

5) After all the user-defined points have been selected, click OK. N points will be listed in the bottom right window.



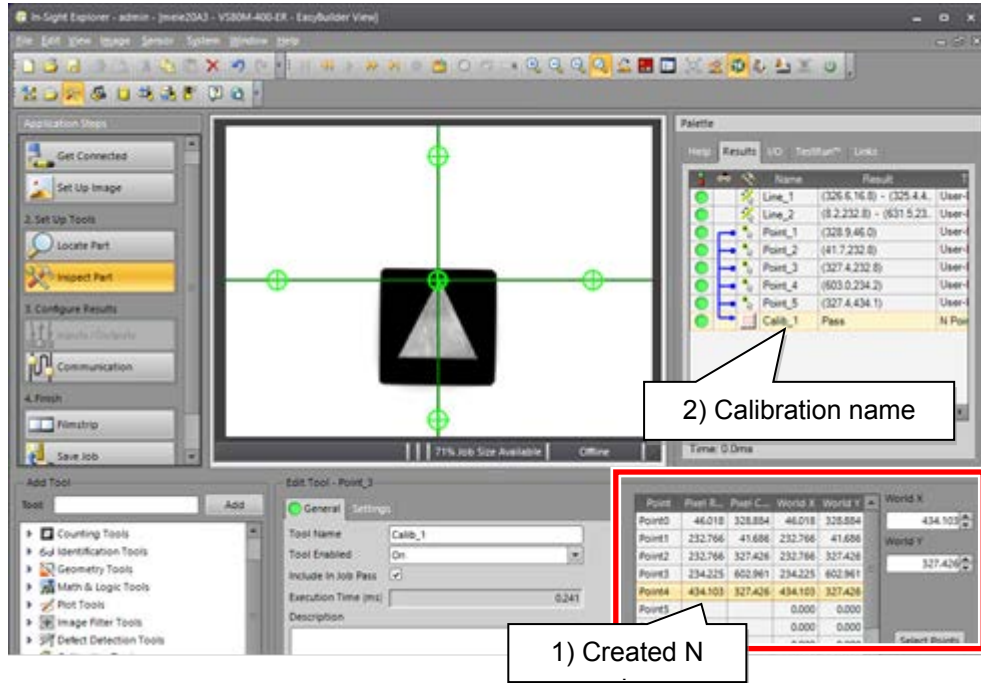
Note Click user-defined points to create N points.

If something else other than the user-defined points is clicked, or a user-defined point does not turn pink when selected, click Cancel and repeat from Step 3 onwards.

(5.2) N point calibration data will be created.

- 1) The selected N points will be listed in the bottom right of the window.
- 2) The calibration name will be displayed in the Palette to the right of the image. (Example: Calib_1)

N points are named in the order in which the points were selected in Step 5.1. For example, the point selected first is "Point 0", the point selected second is "Point 1", and so on.



Note

If the number of listed N points differs from the number of N points you want to create...

Something else other than the user-defined points may have been clicked in Step 5.1.

(Click user-defined points to create N points.)

In this case, recreate N points in the following steps.

- 3) Select the calibration name from Palette, then right-click and select Delete to delete the data.
- 4) Create N points by repeating the process from Step 3 in Section 5.1 onwards.

The number of listed N points is 6, but the number of N points you want to create is 5.

↓

In this case, delete the calibration data and repeat the steps in Section 5.1.

Point	Pixel R...	Pixel C...	World X	World Y
Point0	29.970	37.312	29.970	37.312
Point1	46.018	328.884	46.018	328.884
Point2	232.766	43.144	232.766	43.144
Point3	232.766	327.426	232.766	327.426
Point4	234.225	602.961	234.225	602.961
Point5	434.103	327.426	434.103	327.426
Point6			0.000	0.000

(6) Carrying out N point calibration

In this step we will calibrate the 5 points.

(6.1) Set N points.

- 1) Enter the world X and world Y values for the user-defined points in the list displayed in the bottom right. Use the amount the robot was moved during calibration in inverted values for the world X and world Y values.

The following is an example using the position the robot moved (point 1) to when line 32 (Mov P0*(-30,0,0,0,0,0)) of HND2.prg was executed.

For point 1, enter "+30" in the World X field and "0" in the World Y field. Repeat this for the remaining four points.

Callout: Selecting "Calib_" will display the following window.

Callout: User-defined points

Point	Pixel R...	Pixel C...	World X	World Y
Point0	46.018	328.884	46.018	328.884
Point1	232.766	43.144	232.766	43.144
Point2	232.766	327.426	232.766	327.426
Point3	234.225	602.961	234.225	602.961
Point4	434.103	327.426	434.103	327.426
Point5			0.000	0.000

- 2) Select a point number.

Points 0 to 4
Five points in total

Callout: 3) World X & World Y

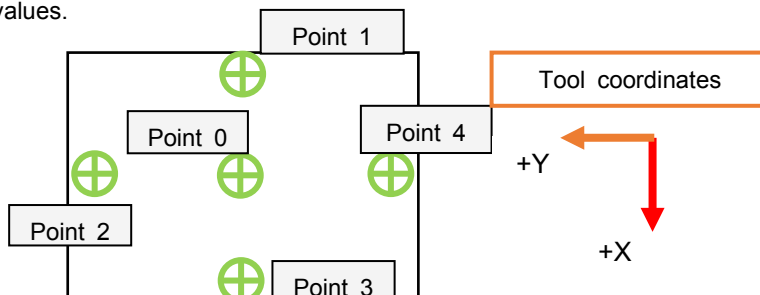
Clicking a point number will display the corresponding values in the World X and World Y fields.

You can overwrite the values.

Input example

The right figure shows the N points created in Step 5.1 as an example.

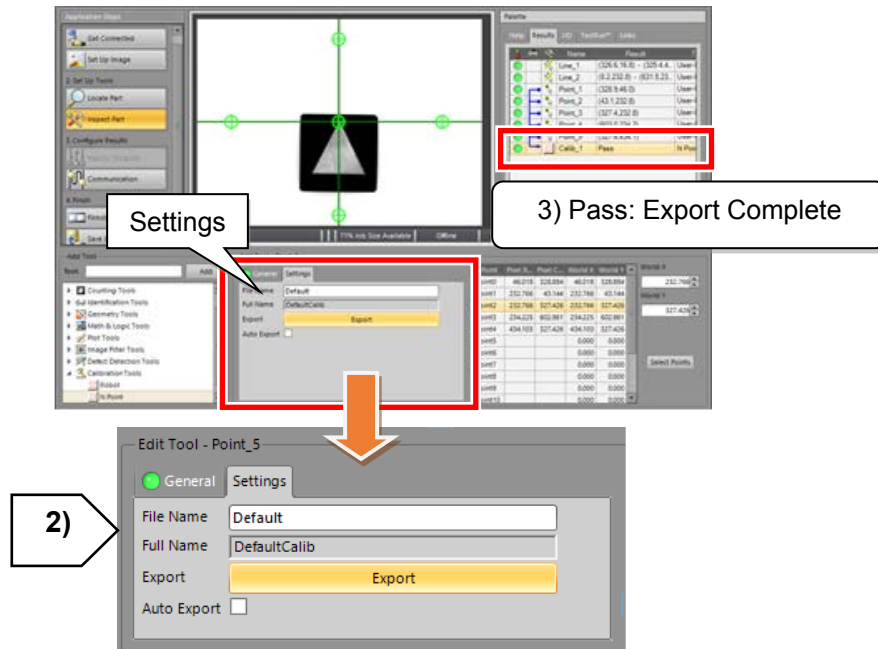
The following table shows the values entered into the program in Step 3.1.



N point	Direction in tool coordinates	Measured value	Program	World X	World Y
Point 0	-	0	P0*(0,0,0,0,0,0)	0	0
Point 1	-X	30	P0*(-30,0,0,0,0,0)	30	0
Point 2	+Y	45	P0*(+45,0,0,0,0,0)	0	-45
Point 3	+X	30	P0*(+30,0,0,0,0,0)	-30	0
Point 4	-Y	45	P0*(0,-45,0,0,0,0)	0	+45

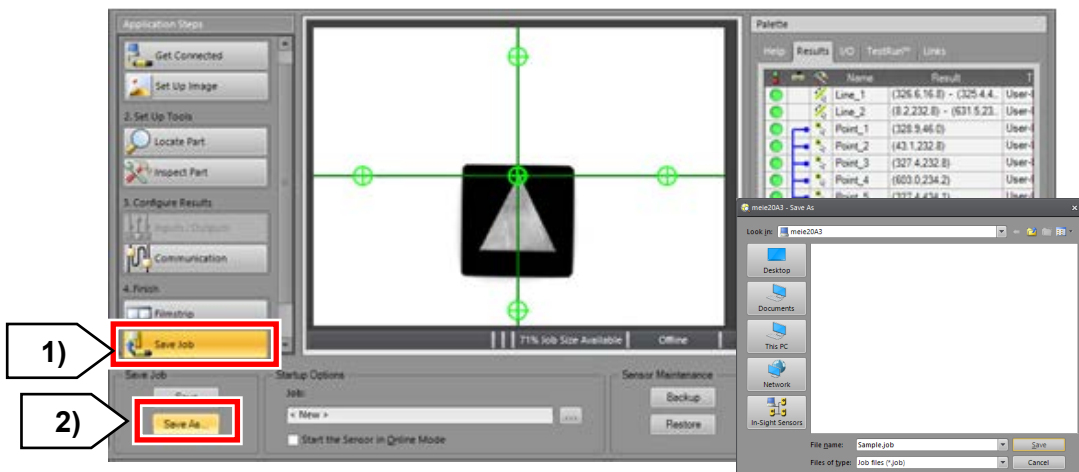
(6.2) Export calibration data.

- 1) Select the Settings tab from the Edit Tool in the bottom center.
- 2) Enter a file name and press Export.
- 3) Pass: Export Complete will be shown in the Results tab of the Palette if the data has been exported successfully.



(6.3) Save the Job.

- 1) Select Save Job from the Application Steps window.
- 2) Click Save As and enter a file name to save the file.



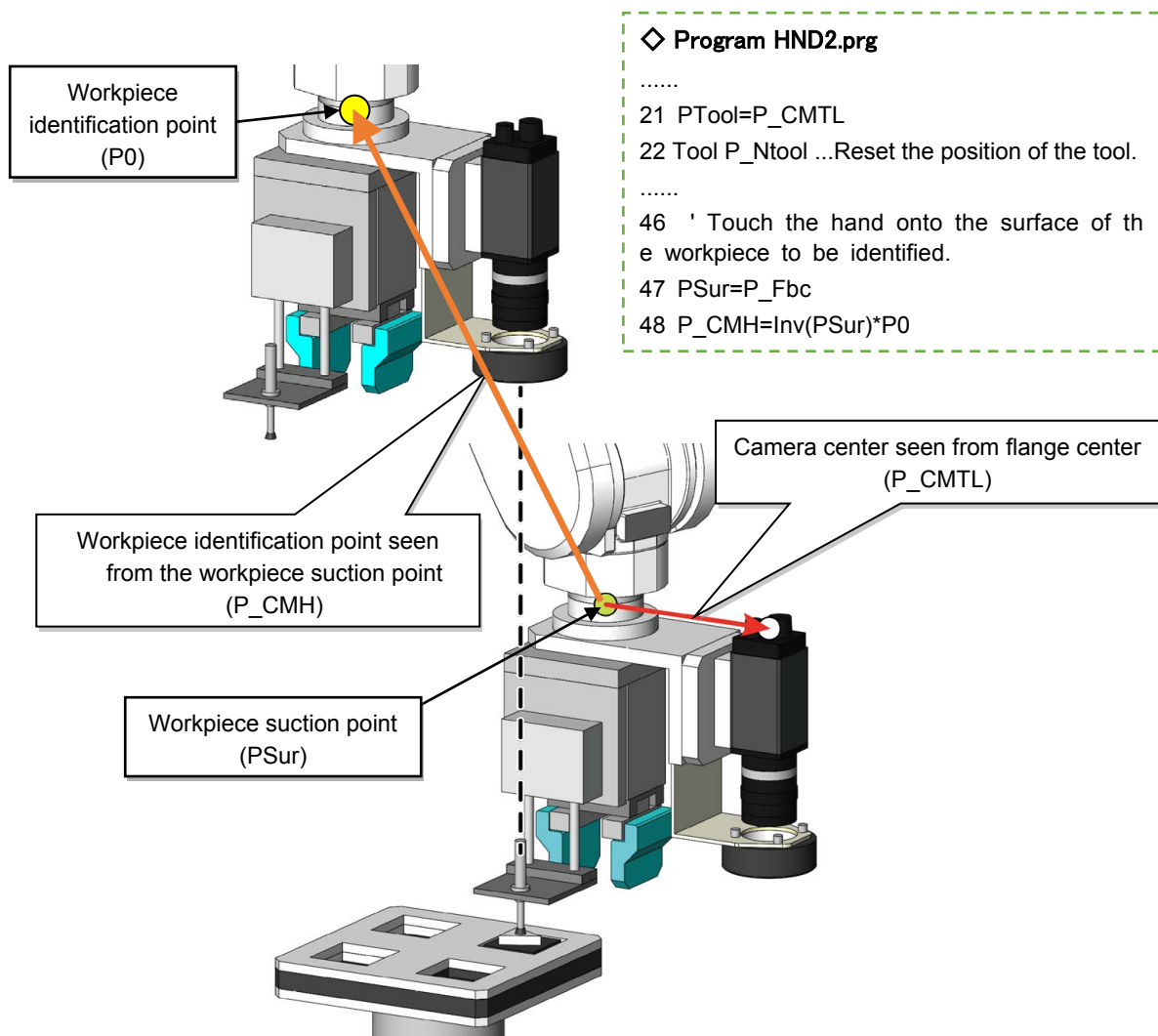
(7) Setting an offset from the workpiece suction point to the imaging point

(8.1) Run HND2.prg.

- 1) Run the program up to line 45 using step operation.
- 2) Close Hand 1 with the teaching pendant and lower the suction pad.

(8.2) Move the robot and make the hand touch the surface of the identified workpiece.

- 1) Move the robot using jog operation and make the suction pad touch the surface of the identified workpiece.
- 2) Once the robot has touched the surface, run the program to END using step operation.
 - ◇ In lines 47 and 48, an offset from the workpiece suction point to the imaging point is set to P_CMH.
- 3) After the program goes to END, move up the robot to the area where no interference occurs.



Notes

4.4 Creating an identification job

4.4.1 Program for creating an identification job

The program used in this chapter is HND3.prg up to line 35.

HND3.prg: Used for setting the reference workpiece grasp position, identification point, and relative position.

◇ Program HND3.prg

```
1 '-----
2 ' Step 3. Program to adjust the hand camera
3 ' Calibration assistance program HND3.prg
4 '-----
5 Def Pos P_CMTL ' Hand camera: Camera center position data
6 Def Pos P_CLPos ' Hand camera: Reference point to start calibration
7 Def Pos P_CMH ' Hand camera: Offset from the workpiece suction surface to the imaging point
8 Def Pos P_HVSP1 ' Hand camera: Default imaging point
9 Def Pos P_WRK03 ' Hand camera: Master workpiece grasp position
10 Def Pos P_PVS03 ' Hand camera: Master workpiece identification point
11 Def Pos P_PH03 ' Hand camera: Calculated coefficient of the handling position from the position of
the identified workpiece
12 Def Char C_C03 ' Hand camera: COM name
13 Def Char C_J03 ' Hand camera: Job name
14 Loadset 1,1
15 OAdl On
16 Servo On
17 Wait M_Svo=1
18 '---- HAND INIT ----
19 Wait M_Svo=1
20 If M_EHOrg=0 Then ' If hand has not returned to origin:
21   EHOrg 1 ' Returns Hand 1 to origin.
22   Wait M_EHOrg=1 ' Waits for Hand 1 to return to origin.
23 EndIf
24 EHOpen 1,100,100 ' Opens Hand 1 (speed = 100%, Force = 20%)
25 Wait M_EHBusy=0 ' Checks if operation is complete
26 '-----
27 Tool P_NTool
28 ' Move the robot to the grasp/suction position.
29 ' Open and close the hand several times and check to see if the workpiece deviates from its position.
30 If M_Mode=1 Then P_WRK03=P_Fbc ' The grasp position of the workpiece to be registered.
31 ' Touches the hand on the surface of the workpiece.
32 If M_Mode=1 Then PWork=P_Fbc ' Job position data
33 If M_Mode=1 Then P_HVSP1=PWork*P_CMH ' The position in which the touched workpiece will be
identified. (Fine adjustments can be made in the XY axes using the Jog mode.)
34 Mov P_HVSP1
35 ' Create an identification job.
```

(continued on the next page→)

```

36 ' Configure communication settings and turn the operation mode to Online.
37 ' Execute automatic operation.
38 If M_NvOpen(3) <> 1 Then
39   NvOpen "COM4:" As #3
40   Wait M_NvOpen(3)=1
41 EndIf
42 Dly 0.5
43 NvRun #3,"sample.job"
44 EBRead #3,,MNUM,PVS
45 Dly 0.1
46 If MNUM=0 Then *ELOOP
47 PTRG0=P_HVSP1*P_CMTL*PVS
48 P_PH03=Inv(PTRG0)*P_WRK03
49 P_PVS03=PVS
50 C_C03$="COM4:"
51 C_J03$="sample.job"
52 Hlt
53 End
54 *ELOOP
55 Error 9000
56 GoTo *ELOOP
57 Hlt
58 End
P_CMTL=(-97.85,-5.06,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(0,0)
P_CLPos=(+184.47,-178.00,+376.40,-180.00,+0.00,-178.77,+0.00,+0.00)(7,0)
P_CMH=(+88.38,-90.49,-140.90,+0.00,+0.00,+0.00,+0.00,+0.00)(7,0)
P_HVSP1=(+165.14,+205.01,+395.53,+179.96,-0.04,-178.78,+0.00,+0.00)(7,0)
P_WRK03=(+251.47,+297.44,+254.63,+179.96,-0.04,-178.78,+0.00,+0.00)(7,0)
P_PVS03=(+10.33,+4.63,+0.00,+0.00,+0.00,+0.01,+0.00,+0.00)(0,0)
P_PH03=(-0.84,+90.90,+140.90,+0.00,+0.00,-0.01,+0.00,+0.00)(7,0)
PWork=(+251.47,+297.44,+254.63,+179.96,-0.04,-178.78)(7,0)
PVS=(+10.33,+4.63,+0.00,+0.00,+0.00,+0.01,+0.00,+0.00)(0,0)
PTRG0=(+252.65,+206.45,+395.47,+179.96,-0.04,-178.79,+0.00,+0.00)(7,0)
PH=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)
PHEIGHT=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)
PVSP1=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)
PWK=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)

```

(End of HND3.prg)

4.4.2 Method of creating an identification job

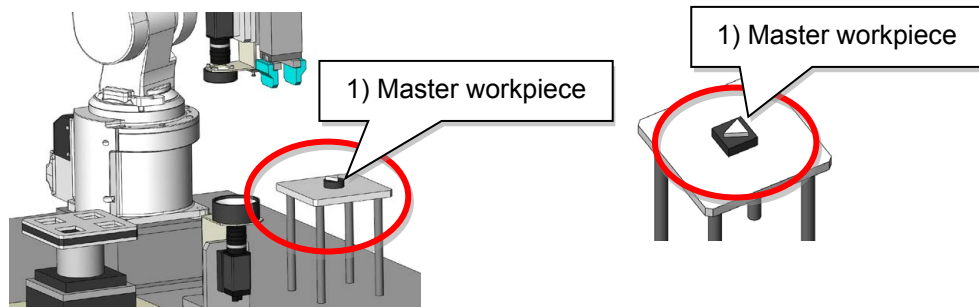
■ Process of creating the identification job

Step	Description
(1)	Adjusting the suction and identification points of the master workpiece
(2)	Creating a job
(3)	Configuring calibration file settings
(4)	Setting the workpiece to be identified and the identification range
(5)	Setting threshold and rotation tolerance values
(6)	Configuring communication settings
(7)	Selecting an identified pattern
(8)	Saving the job

(1) Adjusting the suction and identification points of the master workpiece

(1.1) Place the workpiece to be sucked.

1) Place the workpiece on the platform. (This workpiece is handled as the master workpiece.)



(1.2) Run HND3.prg.

1) Run the program up to line 28 using step operation.

- ◇ In line 21, press and hold the F1 key until the hand operation completes. (Initialization of the electric hand)
- ◇ In line 27 "PTool = P_NTool", the current robot tool data is assigned to PTool.

◇ Program HND3.prg

```
.....
27 Tool P_NTool
28 ' Move the robot to the grasp/suction position.
```

Tool P_NTool
Tool (0,0,0,0,0,0,0,0) (0,0)
The tool position is set back to the flange position.

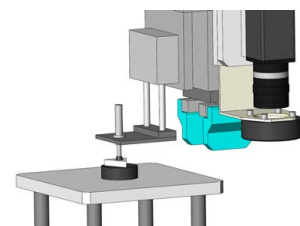
(1.3) Move the robot to a place where it sucks the master workpiece.

1) Move the robot using jog operation to a place where it sucks the master workpiece.

- * If grasping the workpiece, adjust the opening and closing mechanism of the tool to ensure the workpiece does not deviate from its position.

- * If sucking the workpiece, close Hand 1 with the teaching pendant and lower the suction pad.

In the pneumatic hand (standard hand) window, press -C.



(1.4) Run HND3.prg.

1) Run the program up to line 31 using step operation.

◇ In line 31 "P_WRK03=P_Fbc", assign the current position data to P_WRK03.

This is the reference point where the workpiece is sucked.

⚠ CAUTION Executing this command will set the suction point. Therefore, do not move the master workpiece until the job is created.

2) Move the robot using jog operation and make it touch the surface of the workpiece.

(If the hand is already on the workpiece, go to Step 3.)

3) Run HND3.prg up to line 33 using step operation.

◇ In lines 32 and 33, the workpiece detection point (P_HVSP1) is set.

4) Move up the robot hand slightly to prevent the workpiece from shifting.

5) Run HND3.prg up to line 35 using step operation.

The robot will move to the workpiece detection point.

◇ **Program HND3.prg**

.....

```
30 If M_Mode=1 Then P_WRK03=P_Fbc ' The grasp position of the workpiece to be registered.
```

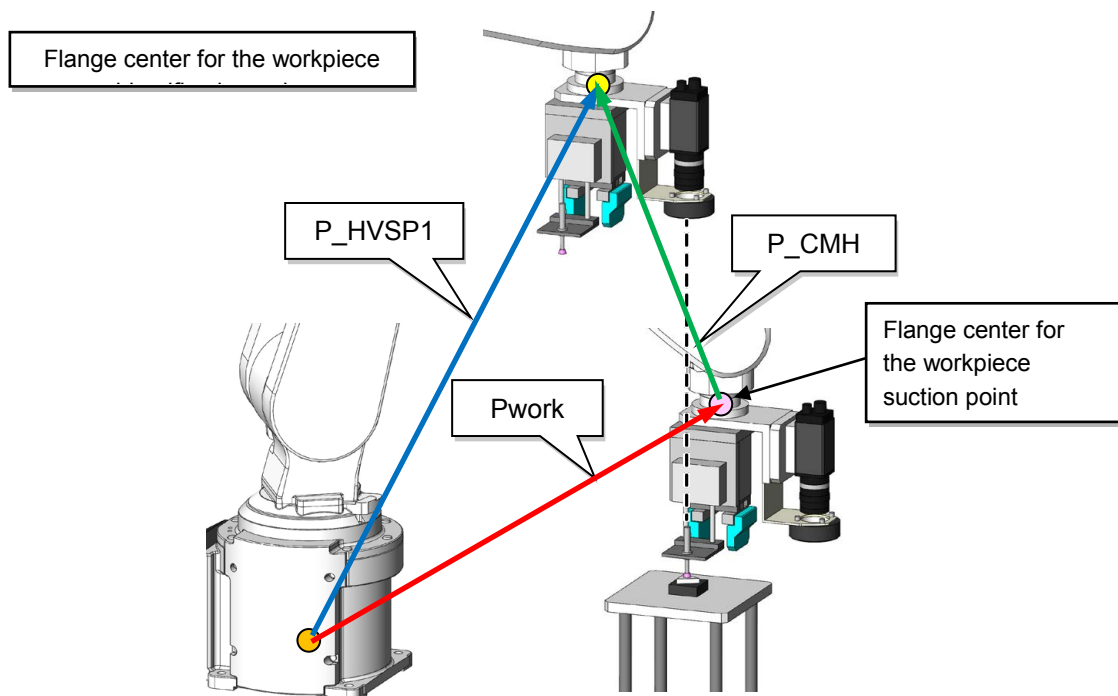
```
31 ' Touches the hand on the surface of the workpiece.
```

```
32 If M_Mode=1 Then PWork=P_Fbc ' Job position data
```

```
33 If M_Mode=1 Then P_HVSP1=PWork*P_CMH ' The position in which the touched workpiece will be identified. (Fine adjustments can be made in the XY axes using the Jog mode.)
```

```
34 Mov P_HVSP1
```

```
35 ' Create an identification job.
```

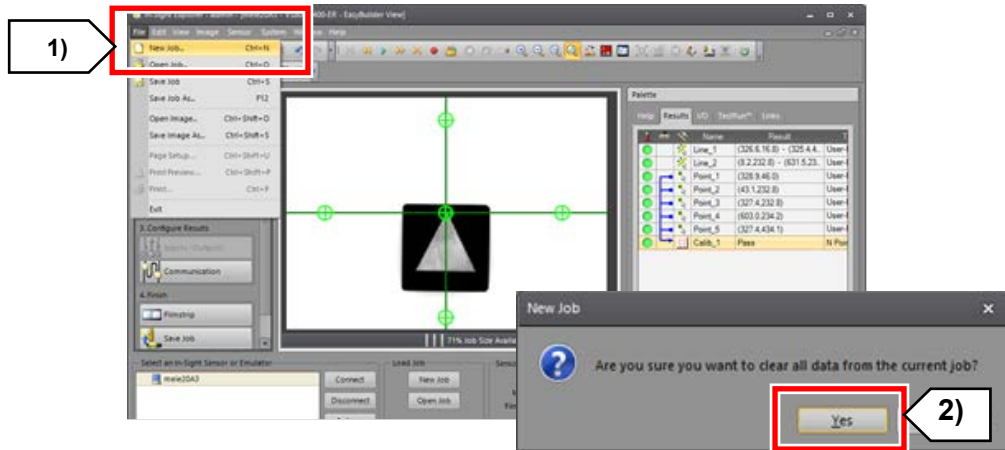


The formula for calculating the position in which the touched workpiece will be identified:
 $P_HVSP1 = PWork * P_CMH$

(2) Creating a job

(2.1) Create a new job.

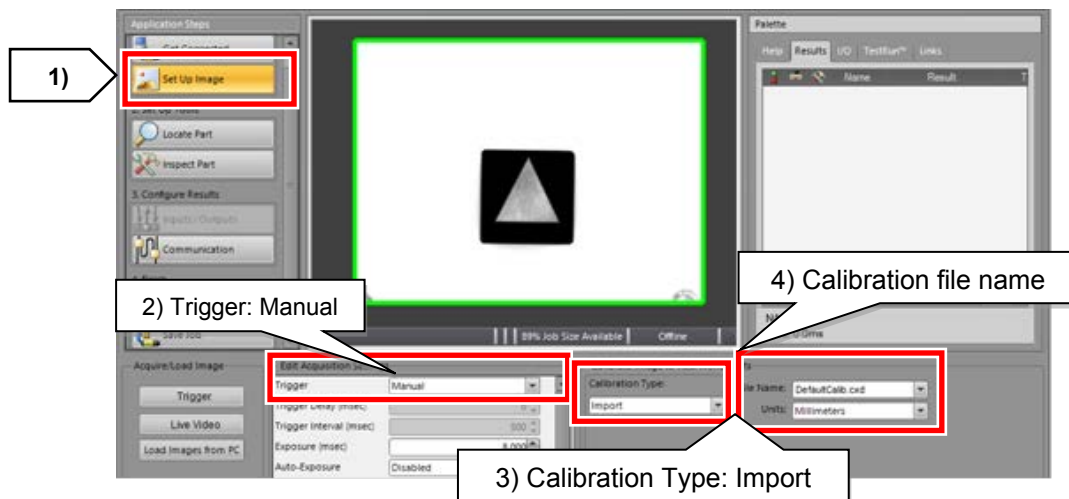
- 1) Go to File then New Job.
 - 2) The message "Are you sure you want to clear all data from the current job?" will appear. Click Yes.
- * Even if Yes is clicked, the calibration data will still be displayed.
- 3) Press the F5 key to acquire an image.



(3) Configuring calibration file settings

(3.1) Select how to acquire an image and set a calibration file.

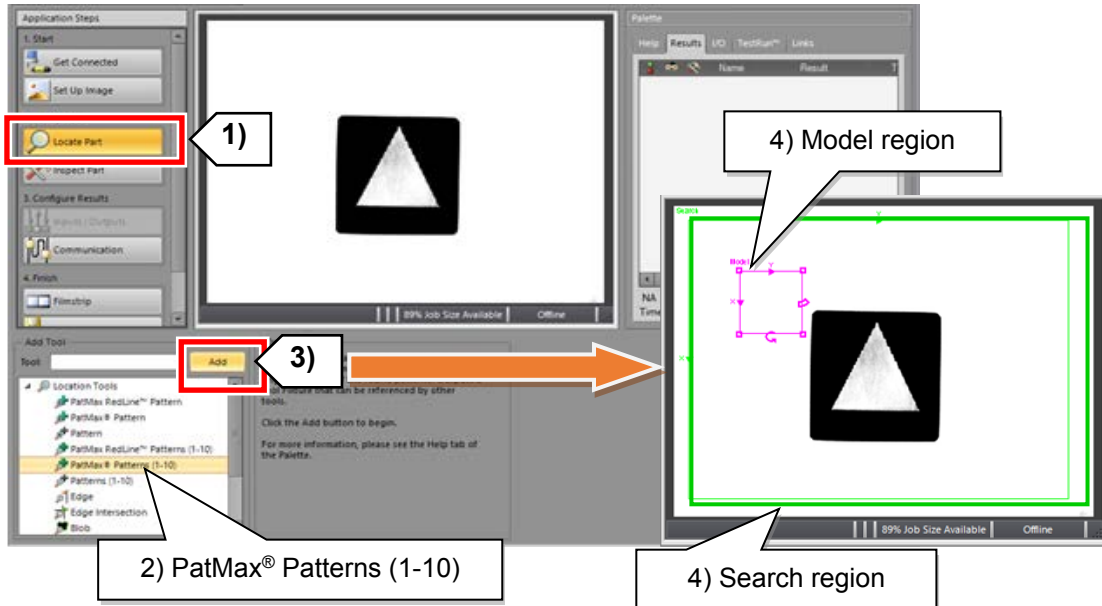
- 1) Select Set Up Image from the Application Steps window.
 - 2) Select Manual or External from the Trigger drop-down box under Edit Acquisition Settings.
* Select Manual in this seminar. (In actual operation, select an appropriate mode.)
 - 3) Select Import from the Calibration Type drop-down box under Calibrate Image to Real World Units.
 - 4) In the File Name field, select the calibration file exported in 4.3.2 Calibration method (Step 6.2).
- * Image data will be converted into robot coordinates from now on.



(4) Setting the workpiece to be identified and the identification range

(4.1) Select a location tool.

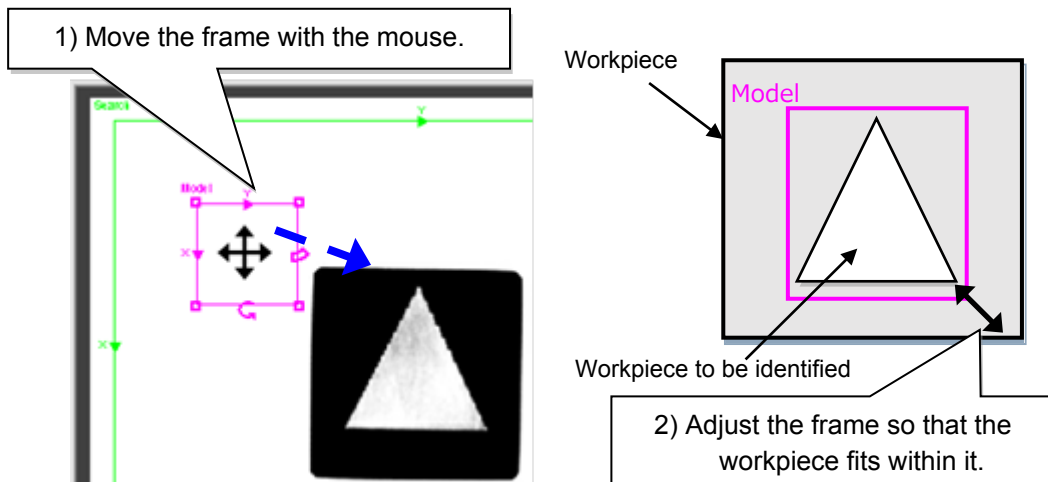
- 1) Select Locate Part from the Application Steps window.
- 2) Select PatMax® Patterns (1-10) from Location Tools in the bottom left.
- 3) Click Add.
- 4) Two boxes, the Model region box and the Search region, box will appear on the image.



(4.2) Set the feature of the workpiece to be identified.

Use the Model region box to identify a feature of the workpiece.

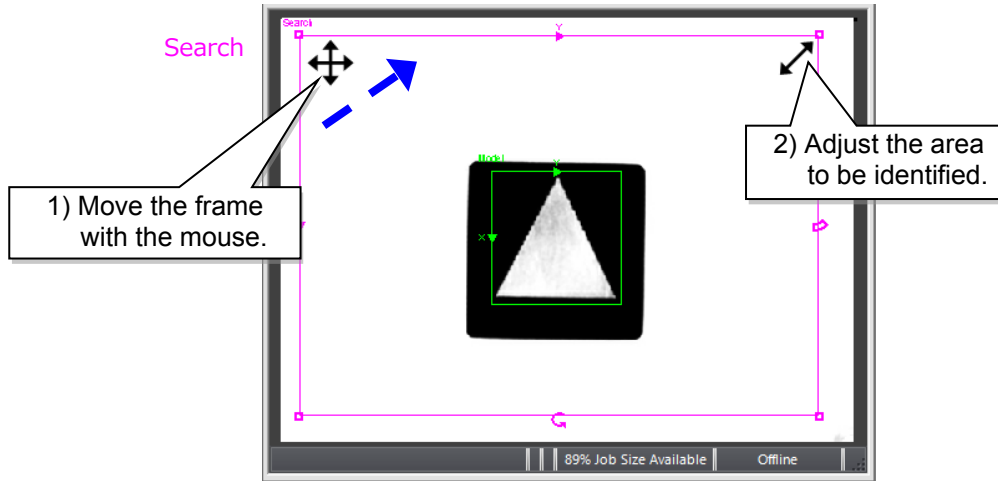
- 1) Left-click inside the Model region box. The frame of the Model region box will turn pink. (The frame of the Model region box will turn pink, and the frame of the Search region box will turn green.)
- 2) Place the cursor inside the frame of the Model region box, press and hold down the left mouse button, then drag the Model region box over the workpiece.
- 3) Adjust the frame so that the workpiece fits within the Model region box.



(4.3) Set the region to be identified.

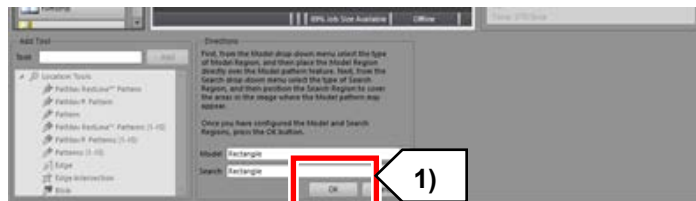
Set the region to be identified using the Search region box.

- 1) To select the search region, left-click in the Search region outside the Model region. (The frame of the Search region will turn pink, and the frame of the Model region will turn green.)
- 2) Move and adjust the frame in the same manner as Step 4.2 to set the area to be identified.

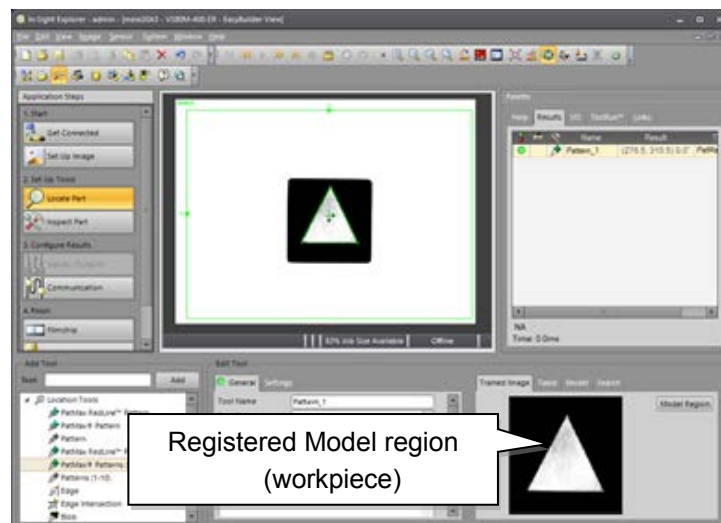


(4.4) Set the Model region and Search region.

- 1) After setting the Model region and Search region, click OK.



- 2) The registered image of the workpiece to be identified (Model region) will be displayed in the bottom right.



(5) Setting threshold and rotation tolerance values

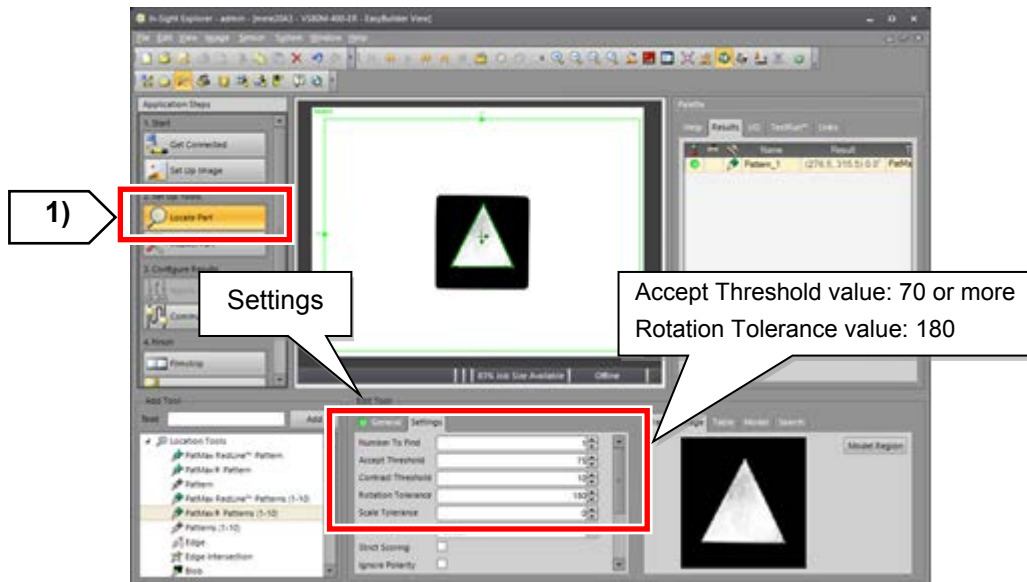
(5.1) Set threshold and rotation tolerance values.

1) Select Locate Part from the Application Steps window.

2) Click the Settings tab and set the Accept Threshold and Rotation Tolerance values.

* Set the Accept Threshold value to 70 or more and the Rotation Tolerance value to 180.

* Changing horizontal and vertical offsets will enable the tool center point to be changed freely.



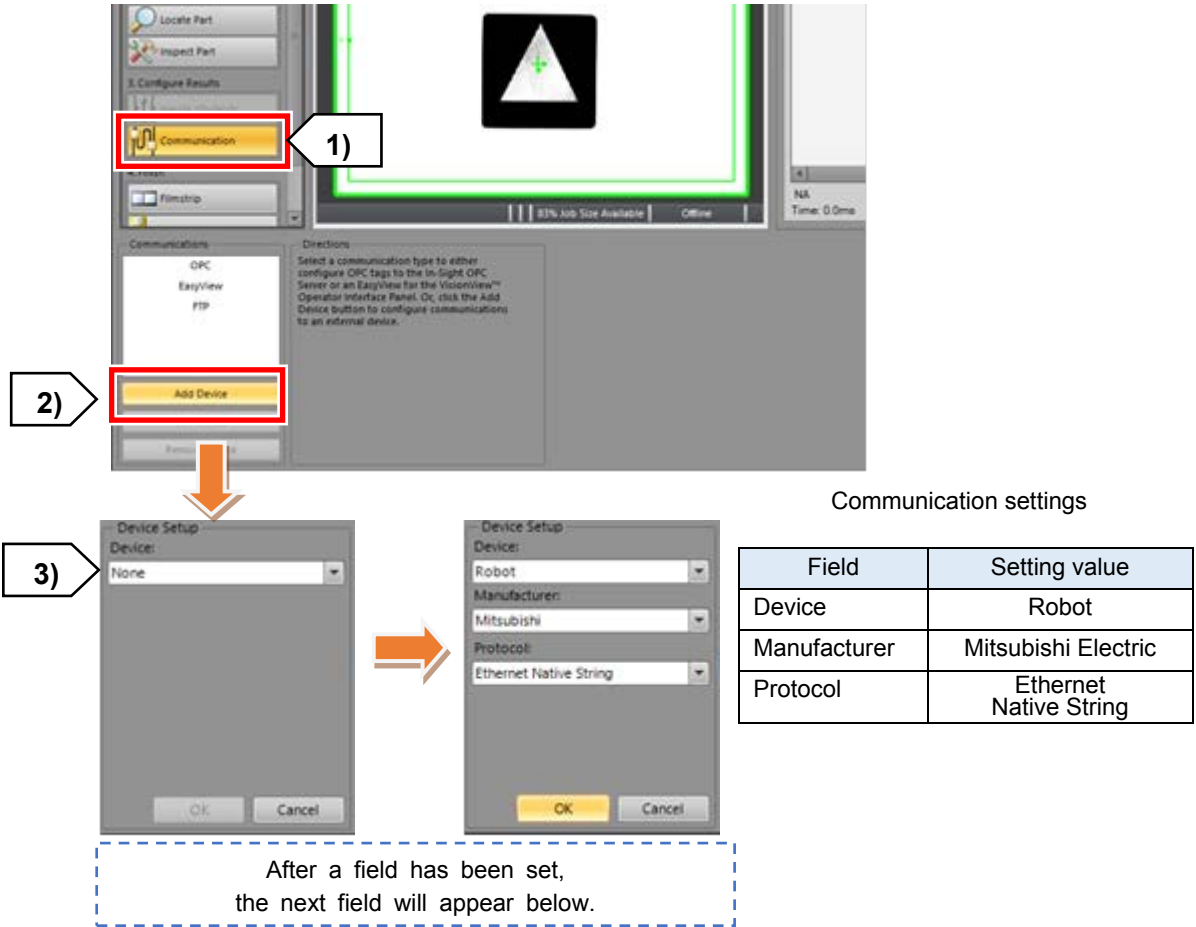
Accept Threshold value: The amount (%) in which the identified workpiece matches the registered image.

- A rotation tolerance value is an angle at which the actual workpiece can be identified even if the workpiece is rotated (\pm deg).

(6) Configuring communication settings

(6.1) Configure the communication settings.

- 1) Select Communication from the Application Steps window.
 - 2) Click Add Device.
 - 3) The Device Setup window will appear on the right. Configure the settings in the following order: (1) Device, (2) Manufacturer, (3) Protocol. (Selecting an item will display the corresponding drop-down list below.)
- * Select Robot for Device, Mitsubishi for Manufacturer, and Ethernet Native String for Protocol.
- 4) Click OK.



1)

2)

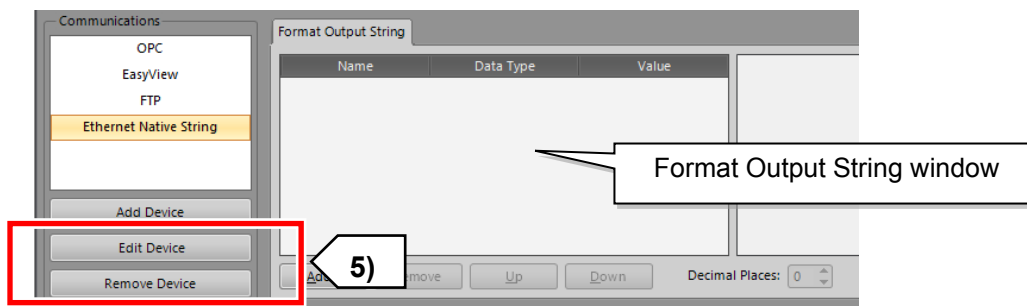
3)

Communication settings

Field	Setting value
Device	Robot
Manufacturer	Mitsubishi Electric
Protocol	Ethernet Native String

After a field has been set, the next field will appear below.

- 5) Ethernet Native String will be added to Communications in the bottom left. The Format Output String window, Edit Device button, and Remove Device button will also appear. To revise device settings (Step 3), click Edit Device.



5)

Format Output String window

(7) Selecting an identified pattern

Select the output data of an identified pattern in the order that the data is output to the robot.

* By setting the order of the output string of the identified pattern to X, Y, Angle, position variables can be input as they are when using the robot command "EBREAD".

Select the data in the following order.

Number of identified patterns	Pattern_1.Number_Found
First X of the matching result	Pattern_1.Fixture.X
First Y of the matching result	Pattern_1.Fixture.Y
First rotational angle C of the matching result	Pattern_1.Fixture.Angle

(7.1) Select the output data of an identified pattern.

- 1) Select Communication from the Application Steps window.
 - 2) Select Ethernet Native String under Communications in the bottom left to display the Format Output String window.
 - 3) Click Add under Format Output String to display the Select Output Data window.
 - 4) Click the blue arrow next to Pattern_1 in the Select Output Data window to display the Pattern_1 list.
 - 5) Select data in the following order while holding down the Ctrl key. (1) Pattern_1.Number_Found, (2) Pattern_1.Fixture.X, (3) Pattern_1.Fixture.Y, (4) Pattern_1.Fixture.Angle. (The order can be changed later.)
- Caution: Be careful when selecting output data because the names are similar to each other.
- 6) Click OK. The data will be displayed in the Format Output String window in the order selected.

1) Communication

2) Ethernet Native String

3) Add

4) Click the blue arrow of Pattern_1.

Select Output Data window

5) Select each piece of data in order while holding down the Ctrl key.
 Pattern_1.Number_Found
 Pattern_1.Fixture.X
 Pattern_1.Fixture.Y
 Pattern_1.Fixture.Angle

Format Output String

6) OK

Be careful when selecting data because the names are similar to each other.

Each piece of data is displayed in the order it is selected in

The order can be changed.

Up Down Decima

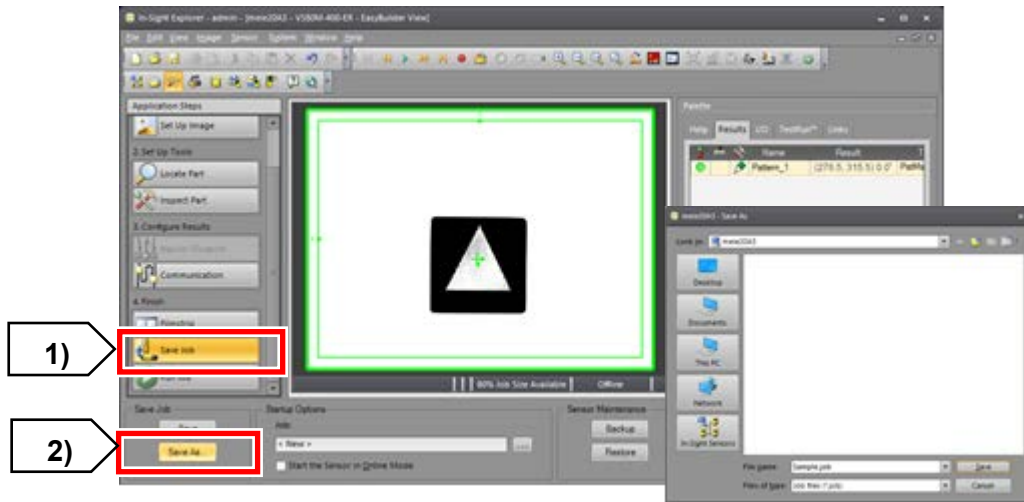
(8) Saving the job

(8.1) Save the job.

- 1) Select Save Job from the Application Steps window.
- 2) Click Save As and enter "Sample" in the File name field.

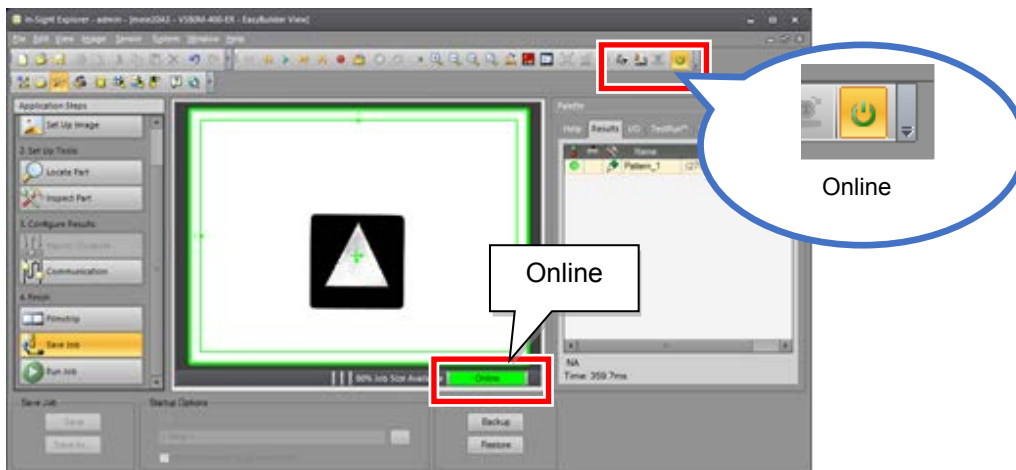
* The file name is used in HND3.prg.

The job file can be named freely, but the file name must be used in HND3.prg.



(8.2) Change the operation mode to Online.

- 1) Click the Online button on the tool bar to change the operation mode to Online.



4.5 Setting up the relative position between the workpiece and the robot

4.5.1 Program for setting up the relative position between the workpiece and the robot

The program used in this chapter is HND3.prg.

HND3.prg: Used for setting up the relative position between the workpiece and the picking point.

◇ Program HND3.prg

For details on the program, refer to 4.4.1 Program for creating an identification job.

4.5.2 Setting up the relative position between the workpiece and the robot

(1) Running HND3.prg automatically

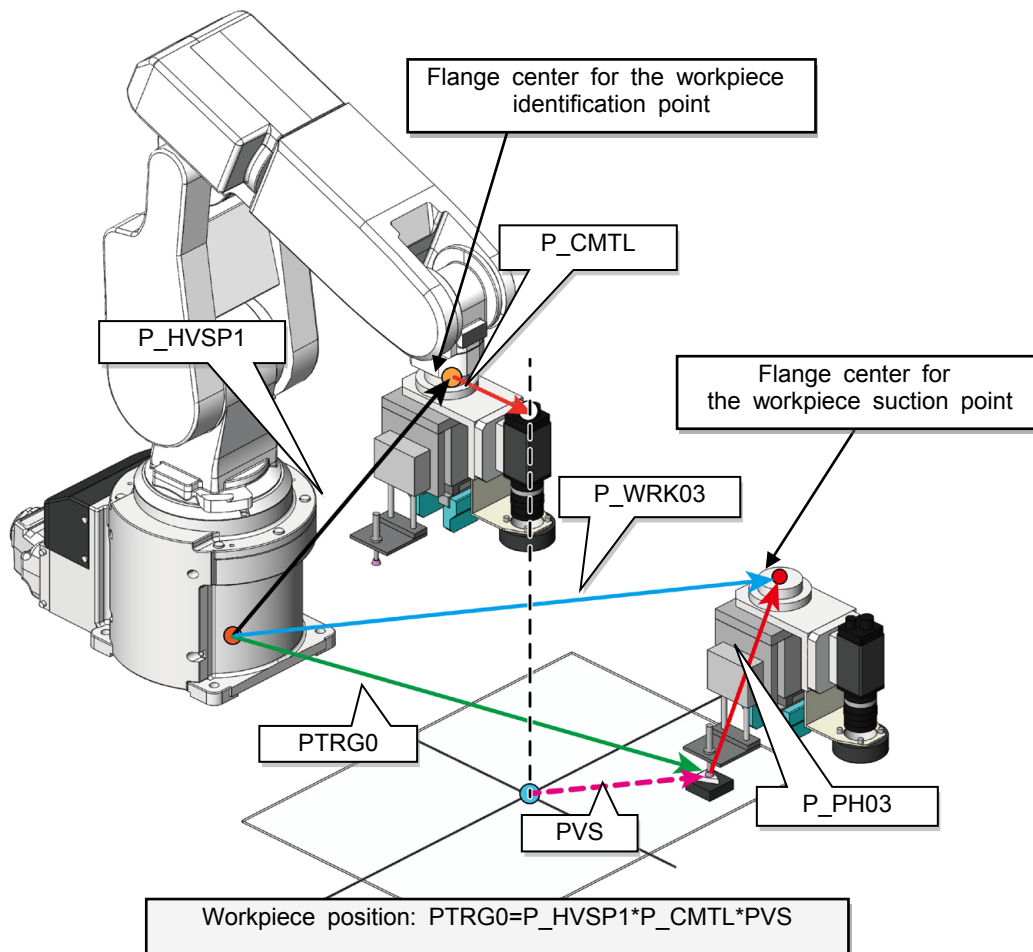
- 1) Change the robot's speed to 3%. (* Set the robot's speed to 3% during automatic operation.)
- 2) Switch the operation mode from "Manual" to "Automatic" with the key switch.
Caution If the operation mode is set to "Manual", automatic operation will not run.
- 2) Check that OvrD is set to 3% and run HND3.prg automatically.
- 3) Wait for the program to stop at line 52.

◇ Program HND3.prg

.....

```
38 If M_NvOpen(3) <> 1 Then
39 NvOpen "COM4:" As #3
40 Wait M_NvOpen(3)=1
41 EndIf
42 Dly 0.5
43 NvRun #3,"sample.job"
44 ERead #3,,MNUM,PVS
```

```
45 Dly 0.1
46 If MNUM=0 Then *ELOOP
47 PTRG0=P_HVSP1*P_CMTL*PVS '-- Workpiece position
48 P_PH03=Inv(PTRG0)*P_WRK03
49 P_PVS03=PVS
50 C_C03$="COM4:"
51 C_J03$="sample.job"
52 Hlt
53 End
```



4.6 Checking the robot movement using step operation and automatic operation

4.6.1 Program for checking robot movement

The program used in this chapter is HND4.prg.

HND4.prg: Used for calculating the workpiece picking position based on the workpiece output position information from the vision sensor to make the robot pick and place the workpiece.

◇Program HND4.prg

```
1 '-----
2 ' Step 3. Program to adjust the hand camera
3 ' Calibration assistance program HND4.prg
4 '-----
5 Def Pos P_CMTL ' Hand camera: Camera center position data
6 Def Pos P_CLPos ' Hand camera: Reference point to start calibration
7 Def Pos P_CMH ' Hand camera: Offset from the surface of the identified workpiece to the imaging
point
8 Def Pos P_HVSP1 ' Hand camera: Default imaging point
9 Def Pos P_WRK03 ' Hand camera: Master workpiece grasp position
10 Def Pos P_PVS03 ' Hand camera: Master workpiece identification point
11 Def Pos P_PH03 ' Hand camera: Coefficient for calculating the handling position from the position of
the identified workpiece
12 Def Char C_C03 ' Hand camera: COM name
13 Def Char C_J03 ' Hand camera: Job name
14 Def Pos P_PHome ' Safe position
15 Loadset 1,1
16 OAdl On
17 Servo On
18 Wait M_Svo=1
19 '---- HAND INIT ----
20 Wait M_Svo=1
21 If M_EHOrg=0 Then ' If hand has not returned to origin:
22   EHOrg 1 ' returns Hand 1 to origin.
23   Wait M_EHOrg=1 ' waits for Hand 1 to return to origin.
24 EndIf
25 EHOpen 1,100,100 ' Opens Hand 1 (speed = 100%, Force = 20%)
26 Wait M_EHBusy=0 ' Checks if operation is complete.
27 '-----
28 Tool P_NTool
29 MCNT=1
30 Mov P_PHome
31 HOpen 1
32 M_Out(10129)=0
33 M_Out(10128)=1 Dly 0.1
34 Dly 0.5
```

(continued on the next page→)

```

35 If M_NvOpen(3)<>1 Then
36   NVOpen C_C03$ As #3
37   Wait M_NvOpen(3)=1
38 EndIf
39 Mov P_HVSP1
40 Dly 1
41 *RETRY
42 NVRun #3,C_J03$
43 EBRead #3,,MNUM,PVS
44 Dly 0.1
45 If MNUM>=1 Then *OK
46 MCNT=MCNT+1
47 If MCNT>3 Then *ELOOP
48 Dly 0.2
49 GoTo *RETRY
50 *OK
51 MCNT=1
52 PTRG=P_HVSP1*P_CMTL*PVS*P_PH03
53 MJUDGH=PosCq(PTRG)
54 If MJUDGH<>1 Then
55   *CNMV
56   Error 9001 'Can Not Move'
57   Hlt
58   GoTo *CNMV
59 EndIf
60 Mov PTRG,-100
61 HClose 1
62 Dly 0.1
63 Mvs PTRG
64 Dly 0.2
65 M_Out(10129)=1
66 Dly 0.3
67 Mvs PTRG,-100
68 Dly 0.1
69 Hlt ' Operation 1: Teach PPUT, and then remove the comment "Hlt" in line 69 of the program.
70 Mov PPUT,-100
71 Mvs PPUT
72 Dly 0.2
73 M_Out(10129)=0
74 M_Out(10128)=1 Dly 0.1
75 Dly 0.2
76 HOpen 1
77 Dly 0.2

```

(continued on the next page→)

```

78 Mvs PPUT,-100
79 Mov P_PHome
80 Hlt
81 End
82 *ELOOP
83 Error 9000
84 GoTo *ELOOP
PHOME=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)
PVSP=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)
PSEARCH=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)
PVSPoS=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(,)
PVS=(-9.11,-0.84,+0.00,+0.00,+0.00,-40.08,+0.00,+0.00)(0,0)
PTRG=(+212.75,+270.33,+254.64,+179.94,+0.00,-138.69,+0.00,+0.00)(7,0)
PPUT=(+243.18,-121.93,+253.56,+179.96,-0.04,-178.82,+0.00,+0.00)(7,0)
P_CMTL=(-97.85,-5.06,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)(0,0)
P_CLPos=(+184.47,-178.00,+376.40,-180.00,+0.00,-178.77,+0.00,+0.00)(7,0)
P_CMH=(+88.38,-90.49,-140.90,+0.00,+0.00,+0.00,+0.00,+0.00)(7,0)
P_HVSP1=(+165.14,+205.01,+395.53,+179.96,-0.04,-178.78,+0.00,+0.00)(7,0)
P_WRK03=(+251.47,+297.44,+254.63,+179.96,-0.04,-178.78,+0.00,+0.00)(7,0)
P_PVS03=(+10.33,+4.63,+0.00,+0.00,+0.00,+0.01,+0.00,+0.00)(0,0)
P_PH03=(-0.84,+90.90,+140.90,+0.00,+0.00,-0.01,+0.00,+0.00)(7,0)
P_PHOME=(+187.48,-13.81,+320.67,-180.00,+0.00,+180.00,+0.00,+0.00)(7,0)

```

(End of HND4.prg)

4.6.2 Checking the movement of the robot

(1) Teaching the release position

(1.1) Run HND4.prg automatically

1) Set the robot's speed to 3%. (* Set the robot's speed to 3% during automatic operation.)

2) Run HND4.prg automatically.

3) "HLT" in line 69) will stop the robot.

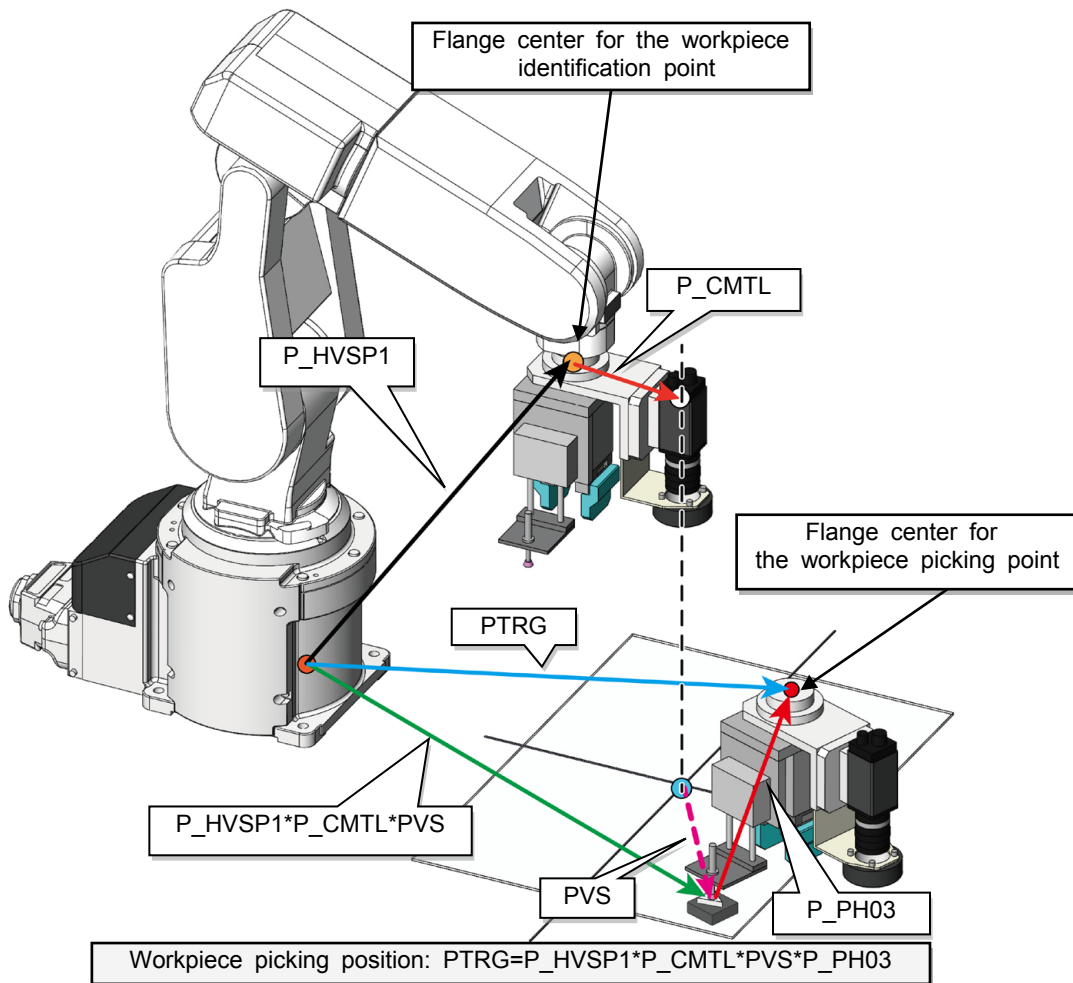
◇ The camera will find the workpiece, and the robot will suck it. Then, the robot will move up its arm and stop.

◇ Program HND4.prg

```

.....
42 NVRun #3,C_J03$
43 EBRead #3,,MNUM,PVS    '--- Workpiece output position information from the vision sensor
.....
52 PTRG=P_HVSP1*P_CMTL*PVS*P_PH03    '--- Calculate the workpiece picking point.
.....
60 Mov PTRG,-100
61 HClose 1

```



P_HVSP1: Default imaging point

P_CMTL: Camera center position data (control point)

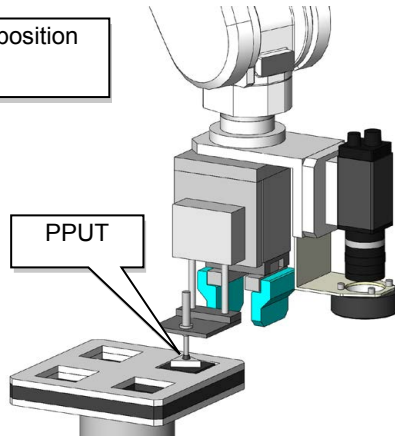
PVS: Workpiece output position information from the vision sensor

P_PH03: Coefficient for calculating the handling position from the position of the identified workpiece

(1.2) Teach the release position.

- 1) Move the robot using jog operation to a place where the hand releases the workpiece.
- 2) Open HND4.prg and teach the release position PPUT.
- 3) After the teaching process, move the robot with the workpiece to a place where no interference occurs.

Teach the release position
"PPUT".



(2) Checking the operation

- (2.1) Run the program automatically and check the operation.

- 1) Remove the comment "Hlt" from line 69 of HND4.prg and save the program.
- 2) Run the program automatically and check that the suction and release operations function properly.

Notes

Appendix 1. Vectors and operations on vectors

Appendix 1.1 Robot position data and vectors

Within the robot program, the position variable P1 is written as $P1=(X, Y, Z, A, B, C)$ (FL1, FL2).

The values of each unit are the change in position from the robots origin to the taught position. These position variables can be treated as vectors.

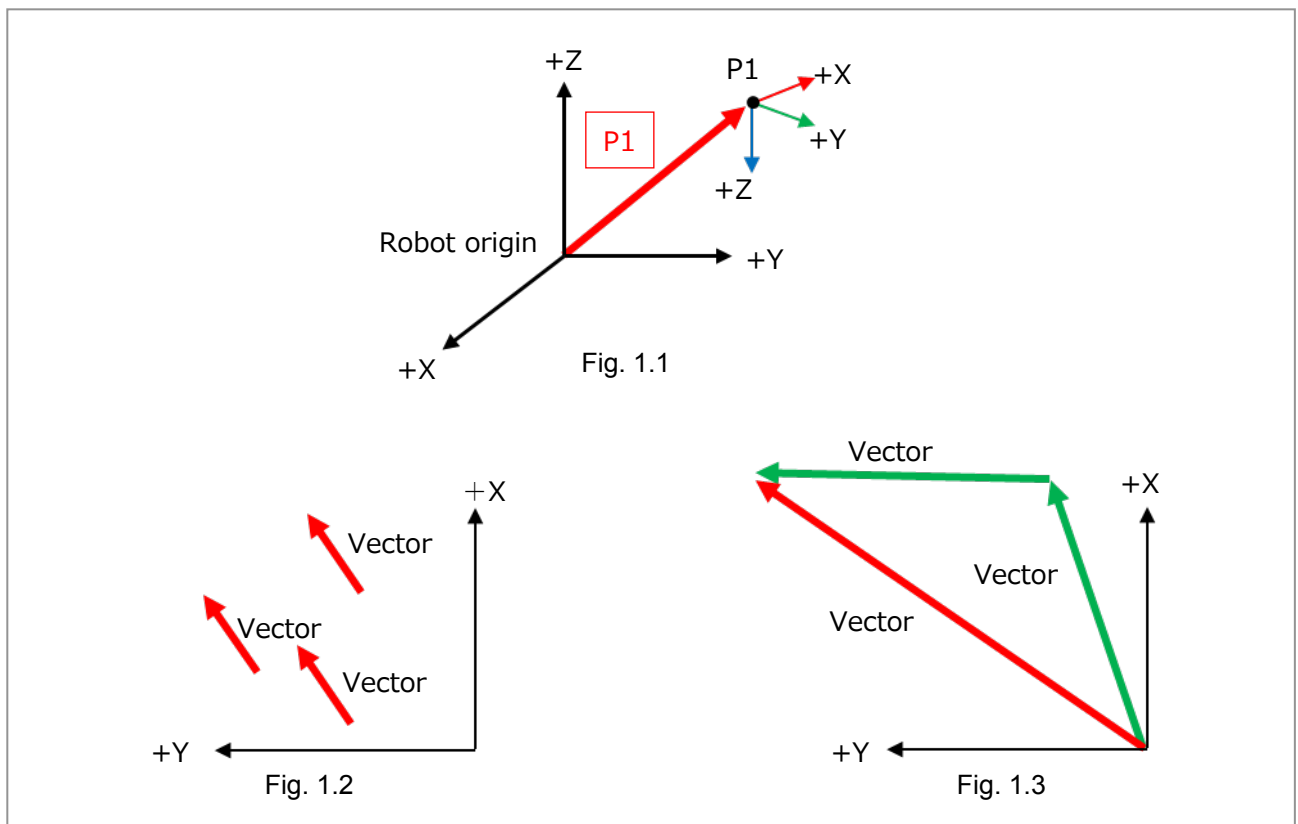
That is to say that, the position variable P1 can be used as a vector that starts at the origin and ends at the taught position P1. (Fig.1.1)

Vectors have two important characteristics.

Characteristics of vectors

1) Even if they originate from different points, vectors are the same if their size and direction are identical to each other.

2) Even if they have multiple paths, vectors are the same if their start points and end points are identical to each other.



*Fig. 1.2: Vector A = Vector B = Vector C

*Fig. 1.3: Vector A = Vector B × Vector C ≠ Vector C × Vector B

Appendix 1.2 Method of finding PH

Referring to Figure 3 on the previous page, in the program "UVS1" (fixed downward-facing camera) vectors A, B, and C are represented as follows: A = PWK, B = PVS, C = PH. (Fig. 1.4)

As was previously mentioned, there is PVS (position of the workpiece) output from the vision sensor and the taught position "PWK" (grasp position of the workpiece).

Using the characteristics of vectors, from these two variables we can find PH (the vector of the position of the identified workpiece to the grasp position).

As was mentioned in the 2nd characteristic of vectors on the previous page, the origin and end point of PH can be split up within the same vector.

Therefore, if PVS is inverted, PH will become $\text{Inv PVS} \times \text{PWK}$.

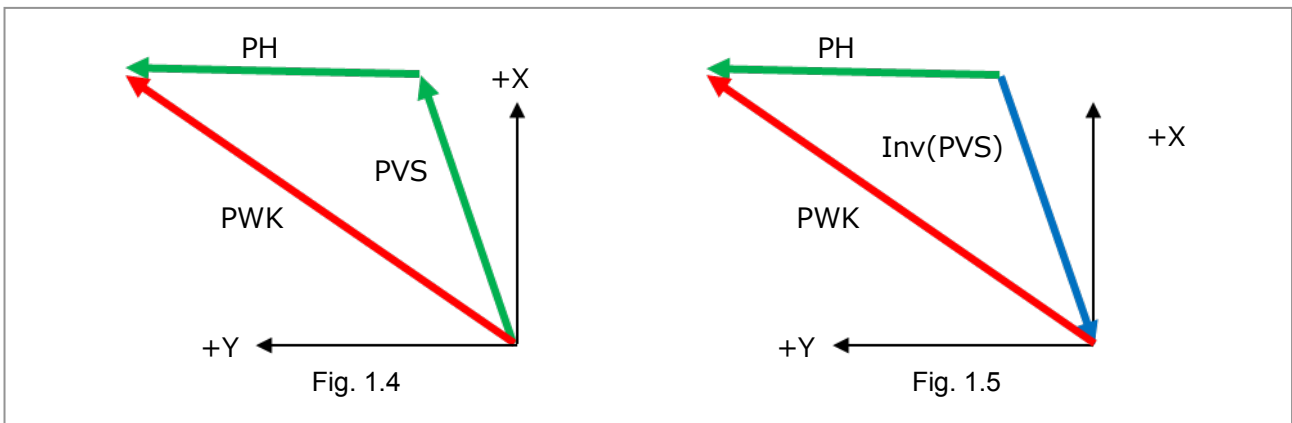
PVS: From the robot's origin to the position where the vision sensor identified the workpiece.

Inv PVS: From the position where the vision sensor identified the workpiece to the robot's origin.

Inverse functions are vectors that have the start and end points of a user-defined vector swapped around.

By using inverse functions, the following points (1 to 3) become the basic principles behind integrating vision sensors and robots.

- 1) Teach the robot the grasp position (PWK) of the master workpiece to make the robot identify it (PVS).
- 2) Calculate the vector "PH" from PWK and PVS. (The vector for where the workpiece is identified up to where it is grasped by the robot.)
- 3) In actual operation, when the output position (PVS) is multiplied by PH, the grasp position becomes PTRG.



Formula for your reference

Regarding Fig. 1.4, the relationship between vectors is expressed in the following formula.

$$\text{PWK} = \text{PVS} * \text{PH}$$

To find PH, multiply both sides by $(\text{PVS})^{-1}$ as follows:

$$(\text{PVS})^{-1} * \text{PWK} = (\text{PVS})^{-1} * \text{PVS} * \text{PH}$$

Then, $(\text{PVS})^{-1} * \text{PVS}$ on the right side becomes 1, and the equation becomes

$$(\text{PVS})^{-1} * \text{PWK} = \text{PH}$$

When the equation is reversed, it becomes

$$\text{PH} = (\text{PVS})^{-1} * \text{PWK}$$

In the robot language, $(\text{PVS})^{-1}$ is $\text{Inv}(\text{PVS})$. Therefore, PH becomes as follows:

$$\text{PH} = \text{Inv}(\text{PVS}) * \text{PWK}$$

Appendix 2. Vision sensor commands and status variables

The robot controller has dedicated commands and status variables which control the vision sensor. Dedicated commands and status variables are described below.

Definitions of terms

Function: A description of the command's function.

Syntax: An example of an argument in the command line

< > indicate an argument.

[] indicate if something can be omitted.

Term: A description of the argument and its setting range.

Example: An example of the command in a program.

Description: Includes detailed information and cautions.

Error: A description of possible errors related to the command.

Appendix 2.1 Vision sensor commands

Command name	Function
NVOpen	Connects and logs on to the vision sensor
NVClose	Disconnects the vision sensor
NVLoad	Changes the state of a specified program so that it can be run
NVRun	Runs the specified program
NVTrg	Requests an image from the vision sensor, and acquires encoder values after a specified period of time
EBRead	Specifies a vision sensor tag and reads the tag's data.
EBWrite	Specifies the tag name of the vision sensor to write data.

NVOpen (Open network vision sensor port)

Function

Connects and logs on to a specified vision sensor.

Syntax

```
NVOpen "<COM No.>" AS # <Vision sensor No.>
```

Term

<COM No.> (cannot be omitted)

Specify the communication port No. in the same way as the Open command.

"COM1" cannot be specified because it is occupied by O/P front RS-232C.

Setting range: COM2: to COM8:

<Vision sensor No.> (cannot be omitted)

Specify a constant from 1 to 8 (Vision sensor No.). The vision sensor connected to the COM port specified with <COM No.> is expressed as a number.

Caution is advised when setting the vision sensor number as it is shared with the Open command <File No.>.

Setting range: 1 to 8

Examples

```
1 If M_NvOpen(1)<>1 Then ' If vision sensor number 1 is not logged on
2 NVOpen "COM2:" As #1 ' connect to the vision sensor connected to COM2, and set the nu
mber to 1.
3 End If
4Wait M_NvOpen(1)=1          ' Connect to vision sensor No. 1 and wait until it has logge
d on.
```

Comments

- 1) Connects and logs on to the vision sensor connected to the port specified by <Com No.>.
- 2) Up to seven vision sensors can be connected simultaneously. <Vision sensor No.> is used to distinguish which vision sensor to communicate with.
- 3) When NVOpen is used together with Open command, <COM No.> and <File No.> of Open Command and the <COM No.> and <Vision sensor No.> of NVOpen are shared, so use numbers other than those specified with the Open command <COM No.> and <File No.>.

Correct	Incorrect
10 Open "COM1:" As #1	10 Open "COM2:" As #1
20 NVOpen "COM2:" As #2	20 NVOpen "COM2:" As #2 ⇒ <COM No.> is used.
30 NVOpen "COM3:" As #3	30 NVOpen "COM3:" As #1 ⇒ <Vision sensor No.> is used.

More than two ports cannot be opened with a setup consisting of one robot controller and one vision sensor. When configuring the parameter "NETHSTIP", the error "Ethernet parameter NETHSTIP setting error" will occur if the IP address of NETHSTIP and NVOpen are the same.

- 4) A username and password are required to logon to the vision sensor. The same username and password set for the vision sensor need to be set in robot controller's parameters "NVUSER" and "NVPSWD".

The username and password must be within 15 characters. It can contain any uppercase letters from A to Z, any numbers from 0 to 9, a hyphen (-) or an underscore (_). (Do not use lowercase characters when creating a new vision sensor user as the teaching pendant does not support them.)

The vision sensor's default administrator username is "admin". The password is "" (two quotation marks).

"NVUSER" and "NVPSWD" share the same administrator username and password.

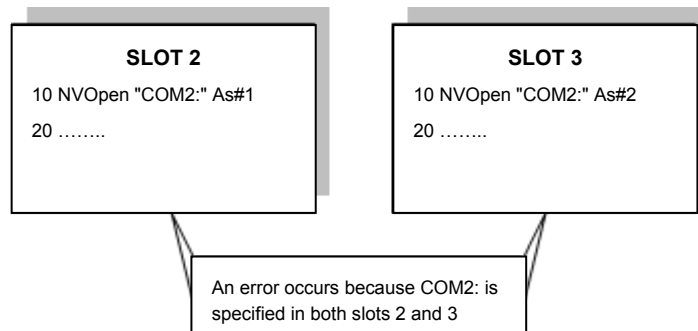
The administrator password can be changed in MELFA-Vision. When changing the administrator password or adding a new user, be sure to change the administrator usernames and passwords of "NVUSER" and "NVPSWD" too. * * * * will appear when changing the username and password of "NVPSWD".

Once the vision sensor password has been changed, open parameter "NVPSWD" and change the password. Restart the robot controller after the password has been changed.

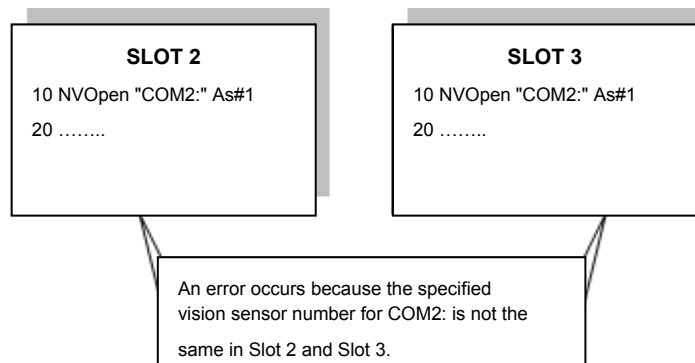
Caution

When connecting multiple vision sensors to a robot controller, ensure that all the usernames and passwords of the vision sensors are the same.

- 5) The communication status of vision sensors can be checked with M_NvOpen once NVOpen has been executed. For further details, refer to M_NvOpen.
- 6) Communication stops immediately if the program is aborted while executing this command. To logon to the vision sensor, reset the robot program and restart MELFA-Vision.
- 7) The following limitations apply when using this command for multi-tasking.
Different tasks cannot have the same <COM No.> and <Vision sensor No.> when multi-tasking.
 - (1) The error "COM file already open" will occur if the same COM number is used for a different task.



- (2) The error "Attempt was made to open an already open communication file" will occur if the same vision sensor number is used for a different task.



- 8) Not supported if the program's startup settings are set to "ALWAYS" or the continuity function (CTN) is enabled.
- 9) Up to three robots can control the same vision sensor simultaneously. If four robots logon to the same vision sensor, one of the four robots will be disconnected. Take this into account during the system design process.
- 10) The program's End command called by the CallP command does not close the port. However, the main program's End command does close the port. The port also closes when the program is reset.
- 11) If interrupt conditions are satisfied while this command is executing, interrupt processing is performed immediately even if the processing of the command is in progress.

Errors

- 1) If the data types of arguments are different, the error "Syntax error in input command" will occur.
- 2) If there is a discrepancy with the number of arguments (too many or too few), the error "Incorrect argument count" will occur.
- 3) Specifying a COM number outside the range of COM2 to COM8 for <COM No.> will result in the error "Argument out of range".
- 4) Specifying a value outside the range of 1 to 8 for <Vision sensor No.> will result in the error "Argument out of range".

- 5) If a COM port <COM No.> which is already connected is specified, the error "COM file already open" will occur. (This also applies to files <File No.> already opened by the Open command.)
- 6) If a COM port is opened before a vision sensor is connected, the error "Vision sensor not connected" will occur. (The same manufacturer parameter "COMTIMER" as in the Ethernet specifications is used. Currently set at "1s".)
- 7) If the same COM number or vision sensor number is used for a different task, the error "COM file already open" will occur.
- 8) If the username and password specified by parameters "NVUSER" (username) and "NVPSWD" (password) are different, the error "Incorrect password" will occur.
- 9) If communication is disconnected while this command is being executed, the error "The communication is abnormal" will occur and the robot controller's port will close.
- 10) If the program's start conditions are set to "ALWAYS", the error "The command cannot be used when the start conditions are ERR or ALW." will occur.

NVClose (Disconnect network vision sensor)

Function

Disconnects from the specified vision sensor.

Syntax

NVClose [[#]<Vision sensor No.> [, [[#]<Vision sensor No.>...]
--

Term

<Vision sensor No.> (can be omitted)

Specify a constant from 1 to 8 (Vision sensor No.).

The vision sensor connected to the COM port specified with <COM No.> is expressed as a number. When omitted, all connections (vision sensor connections) established by the NVOpen command are closed. Up to eight vision sensor numbers can be specified and each one should be separated with a comma.

Setting range: 1 to 8

Examples

```
1 If M_NvOpen(1)<>1 Then ' If vision sensor number 1 is not logged on
```

```
2 NVOpen "COM2:" As #1 ' connect to the vision sensor connected to COM2, and set the number to 1.
```

```
3 EndIf
```

```
4 Wait M_NvOpen(1)=1          ' Connect to vision sensor No. 1 and wait until it has logged on.
```

```
10 ...
```

```
20 NVClose #1                ' Disconnect from the vision sensor connected to COM2.
```

Comments

- 1) Disconnects from the vision sensor to which a connection was established with the NVOpen command.
- 2) If the <Vision sensor No.> is omitted, all connections are closed.
- 3) If a connection has been already closed, the process proceeds to the next step.
- 4) It is possible to connect to up to seven vision sensors simultaneously. Therefore, the <Vision sensor No.> is used to identify which vision sensor is to be disconnected.
- 5) If the program is aborted while executing this command, execution is continued until processing of this command has completed.
- 6) If this command is used for multi-tasking, execute the command "NVOpen" for the relevant task and close only the connection that is open. Use the number specified with the NVOpen command for the vision sensor number that is to be used.
- 7) Not supported if the program's startup settings are set to "ALWAYS" or the continuity function (CTN) is enabled.
- 8) If the End command is used, all connections established by the NVOpen or Open command are closed. However, connections are not closed with End command inside programs called with the CallP command.
Furthermore, connections are also closed when resetting the program. Therefore, if the End command is specified or the program is reset, there is no need to close connections using this command.
- 9) The continuity function is not supported.
- 10) If interrupt conditions are satisfied while this command is executing, interrupt processing is performed after processing of this command has completed.

Errors

- 1) If the value specified for the <Vision sensor No.> is anything other than 1 to 8, the error "Argument out of range" will occur.
- 2) If there are more than eight arguments in the command, the error "Argument out of range" will occur.

NVLoad (Load network vision sensor)

Function

Loads a specified vision program to a vision sensor.

Syntax

```
NVLoad #<Vision sensor No.>,<Vision program (job) name>
```

Term

<Vision sensor No.> (cannot be omitted)

Specify the number of the vision sensor to control. Setting range: 1 to 8

<Vision program (job) name> (can be omitted)

Specify the name of the vision program to be started.

The filename extension (.job) can be omitted.

Acceptable characters include: 0 to 9, A to Z (upper and lower case), hyphens (-), and underscores (_).

Examples

```
1 If M_NvOpen(1)<>1 Then ' If vision sensor number 1 is not logged on
2 NVOpen "COM2:" As #1 ' connect to the vision sensor connected to COM2, and set the numb
er to 1.
3 EndIf
4 Wait M_NvOpen(1)=1
5 NVLoad #1,"TEST" ' Load program "TEST".
6 NVRun #1,"TEST" ' Run program "TEST".
7 EBRead #1,,MNUM,PVS1,PVS2 ' Read the tag data of "Job.Robot.FormatString" and save
it in the variables MNUM, PVS1, and PVS2.
8 ...
30 NVClose #1 ' Disconnect from the vision sensor connected to COM2.
```

Comments

- 1) Loads the specified program for the specified vision sensor.
- 2) The program moves to the next step once NVLoad has loaded the vision sensor program to the vision sensor.
- 3) This command will suspend immediately if the program is aborted while it is being executed.
- 4) If the specified vision program name is already loaded, processing of this command is ended.
- 5) If this command is used for multi-tasking, it is necessary to use the NVOpen command for each task. Use the vision sensor number specified with the NVOpen command.
- 6) Not supported if the program's startup settings are set to "ALWAYS" or the continuity function (CTN) is enabled.
- 7) If interrupt conditions are satisfied while this command is executing, interruption processing will be executed immediately.

Errors

- 1) If the data types of arguments are different, the error "Syntax error in input command" will occur.
- 2) If there is a discrepancy with the number of arguments (too many or too few), the error "Incorrect argument count" will occur.
- 3) If the value specified for <Vision sensor No.> is anything other than 1 to 8, the error "Argument out of range" will occur.
- 4) If the NVOpen command is not executed with the number set in <Vision sensor No.>, the error "Illegal vision sensor number" will occur.
- 5) If <Vision program name> exceeds 15 characters, the error "Vision program name is abnormal" will occur.
- 6) If characters other than letters from A to Z, numbers from 0 to 9, a hyphen (-) or an underscore (_) are used in <Vision program name>, the error "Vision program name is abnormal" will occur.
- 7) If the program specified in <Vision program name> has not been loaded to the vision sensor, the

error "Specified vision program not loaded" will occur.

- 8) If the vision sensor is offline, the error "Change status to Online" will occur. Change the vision sensor's status to Online.
- 9) If communication is disconnected while this command is being executed, the error "The communication is abnormal" will occur and the robot controller's port will close.

NVRun (Run network vision sensor program)

Function

Runs the specified program

Syntax

NVRun #<Vision sensor No.>,<Vision program (job) name>
--

Term

<Vision sensor No.> (cannot be omitted)

Specify the number of the vision sensor you want to control. Setting range: 1 to 8

<Vision program (job) name> (can be omitted)

Specify the name of the vision program you want to start. The filename extension (.job) can be omitted.

Acceptable characters include: 0 to 9, A to Z (upper and lower case), hyphens (-), and underscores (_).

Examples

```
1 If M_NvOpen(1)<>1 Then ' If vision sensor number 1 is not logged on
2 NVOpen "COM2:" As #1 ' connect to the vision sensor connected to COM2, and set the number to 1.
3 Endif
4 Wait M_NvOpen(1)=1      ' Connect to vision sensor No. 1 and wait until it has logged on.
5 NVLoad #1,"TEST" ' Load program "TEST".
6 NVRun #1,"TEST" ' Run program "TEST".
7 EBRead #1,,MNUM,PVS1,PVS2 ' Read the tag data of "Job.Robot.FormatString" and save it in the variables MNUM, PVS1, and PVS2.
8 ...
30 NVClose #1          ' Disconnect from the vision sensor connected to COM2.
```

Comments

- 1) Runs the specified program for the specified vision sensor.
- 2) The timing of when this command finishes processing differs depending on the setting of the parameter NVTRGTMG. If the parameter NVTRGTMG is a factory setting, the next command is executed after communication with the image processing command (image request) has finished.
- 3) This command will suspend immediately if the program is aborted while it is being executed.
- 4) If the specified vision program name is already loaded, only image capture and image processing are executed. (The vision program is not loaded.)
- 5) Use the EBRead command to get data from the vision sensor.
- 6) If this command is used for multi-tasking, it is necessary to use the NVOpen command for each task. Use the vision sensor number specified with the NVOpen command.
- 7) Not supported if the program's startup settings are set to "ALWAYS" or the continuity function (CTN) is enabled.
- 8) Set the trigger of EasyBuilder's image capture settings to "External trigger", "Manual trigger" or "Network" ("Camera" can be used when the setting value of NVTRGTMG is "0" or "2".)
- 9) Up to three robots can control the same vision sensor at the same time, but this command cannot be used by more than one robot at the same time. Use this command for only one of the robots.
- 10) If interrupt conditions are satisfied while this command is executing, interruption processing will be executed immediately.

Errors

- 1) If the data types of arguments are different, the error "Syntax error in input command" will occur.
- 2) If there is a discrepancy with the number of arguments (too many or too few), the error "Incorrect argument count" will occur.
- 3) If the value specified for <Vision sensor No.> is anything other than 1 to 8, the error "Argument out of range" will occur.
- 4) If the NVOpen command is not executed with the number set in <Vision sensor No.>, the error "Illegal vision sensor number" will occur.
- 5) If <Vision program name> exceeds 15 characters, the error "Vision program name is abnormal" will occur.
- 6) If characters other than letters from A to Z, numbers from 0 to 9, a hyphen (-) or an underscore (_) are used in <Vision program name>, the error "Vision program name is abnormal" will occur.
- 7) If the program specified in <Vision program name> has not been loaded to the vision sensor, the error "Specified vision program not loaded" will occur.
- 8) If the trigger of EasyBuilder's image capture setting is set to anything other than "External trigger", "Manual trigger" or "Network", the error "Abnormal image capture specification" will occur. The same error occurs if the image capture setting is set to "Camera" and NVTRGTMG is set to "1".
- 9) If the vision sensor is offline, the error "Change status to Online" will occur. Change the vision sensor's status to Online.
- 10) If communication is disconnected while this command is being executed, the error "The communication is abnormal" will occur and the robot controller's port will close.

NVTrg (Network vision sensor trigger)

Function

Requests the specified vision program to capture an image

Syntax

```
NVTrg #<Vision sensor No.>,<Delay time>,<Encoder 1 value read-out variable>[, [<Encoder 2 value read-out variable>]
[, [<Encoder 3 value read-out variable>]], [<Encoder 4 value read-out variable>]
[, [<Encoder 5 value read-out variable>]], [<Encoder 6 value read-out variable>]
[, [<Encoder 7 value read-out variable>]], [<Encoder 8 value read-out variable>]
```

Term

<Vision sensor No.> (cannot be omitted)

Specify the number of the vision sensor you want to control. Setting range: 1 to 8

<Delay Time> (cannot be omitted)

Specify the delay time (ms) from when the image capture request is output to the vision sensor until the encoder value is obtained.

Setting range: 0 to 150 ms

<Encoder n value read-out variable> (Can be omitted from the second one on)

Specify the double precision numeric variable into which the read external encoder n value is set . Note: n is 1 to 8

Examples

```
1 If M_NvOpen(1)<>1 Then ' If vision sensor number 1 is not logged on
2 NVOpen "COM2:" As #1 ' connect to the vision sensor connected to COM2, and set the nu
  mber to 1.
3 EndIf
4 Wait M_NvOpen(1)=1      ' Connect to vision sensor No. 1 and wait until it has logged on.
5 NVLoad #1,"TEST"       ' Load program "TEST".
6 NVTrg #1,15,M1#,M2#    ' Request the vision sensor to capture an image and acquire enc
  oders 1 and 2 after 15 ms.
7 EBRead #1,,MNUM,PVS1,PVS2 ' Read the tag data of "Job.Robot.FormatString" and save
  it in the variables MNUM, PVS1, and PVS2.
8 ...
30 NVClose #1           ' Disconnect from the vision sensor connected to COM2.
```

Comments

- 1) Outputs the image capture request to the specified vision sensor and acquires the encoder value after the specified period of time. The acquired encoder value is stored in the specified numeric variable.
- 2) The timing of when this command finishes processing differs depending on the setting of the parameter NVTRGTMG. If the parameter NVTRGTMG is a factory setting, the next command is executed after communication with the image processing command (image request) has finished.
- 3) This command will suspend immediately if the program is aborted while it is being executed.
- 4) Use the EBRead command to get data from the vision sensor.
- 5) If this command is used for multi-tasking, it is necessary to use the NVOpen command for each task. Use the vision sensor number specified with the NVOpen command.
- 6) Not supported if the program's startup settings are set to "ALWAYS" or the continuity function (CTN) is enabled.
- 7) Set the trigger of EasyBuilder's image capture settings to "External trigger", "Manual trigger" or "Network" ("Camera" can be used when the setting value of NVTRGTMG is "0".)
- 8) Up to three robots can control the same vision sensor at the same time, but this command cannot be used by more than one robot at the same time. Use this command for only one of the robots.
- 9) If interrupt conditions are satisfied while this command is executing, interruption processing will be executed immediately.

Errors

- 1) If the data types of arguments are different, the error "Syntax error in input command" will occur.
- 2) If there is a discrepancy with the number of arguments (too many or too few), the error "Incorrect argument count" will occur.
- 3) If the value specified for <Vision sensor No.> is anything other than 1 to 8, the error "Argument out of range" will occur.
- 4) If the NVOpen command is not executed with the number set in <Vision sensor No.>, the error "Illegal vision sensor number" will occur.
- 5) If the vision program's image capture setting is set to anything other than "Camera" (All trigger commands) "External trigger", "Manual trigger" or "Network", the error "abnormal image capture specification" will occur.
- 6) If the trigger of EasyBuilder's image capture setting is set to anything other than "External trigger", "Manual trigger" or "Network", the error "abnormal image capture specification" will occur. The same error occurs if the image capture setting is set to "Camera" and NVTRGTMG is set to "1" or "2".
- 7) If the vision sensor is offline, the error "Change status to Online" will occur. Change the vision sensor's status to Online.
- 8) If communication is disconnected while this command is being executed, the error "The communication is abnormal" will occur and the robot controller's port will close.

EBRead (EasyBuilder read)

Function

Specifies a vision sensor tag and reads the tag's data.
Stores the data read by the vision sensor in a specified variable.
Specify the tag name and read the data using this command if the vision program (job) has been created with the vision tool "EasyBuilder" by Cognex Corporation.

Syntax

EBRead #<Vision sensor No.>, [<Tag name>] , <Variable name 1> [,<Variable name 2>]... [,<Time out>]
--

Term

- <Vision sensor No.> (Cannot be omitted)
Specify the number of the vision sensor you want to control. Setting range: 1 to 8
- <Tag name> (Can be omitted)
Specify the name of the symbolic tag where data read by the vision sensor is to be stored.
When omitting the tag name, the value of parameter EBRDTAG (initial value is the custom format tag name "Job.Robot.FormatString") is specified.
- <Variable name> (Cannot be omitted):
Specify the variable read from the vision sensor to send to the robot. Multiple variables can be delimited by a comma.
Numeric value variables, position variables, and string variables can be specified.
When the position variable is specified, the components that have vision data set in them are X, Y, and C. The values of components that do not have any data set in them are set to "0".
- <Time out> (If omitted, 10)
Specify the time-out time in seconds.
Setting range: 1 to 32767 (integers)

Examples

```
1 If M_NvOpen(1)<>1 Then ' If vision sensor number 1 is not logged on
2 NvOpen "COM2:" As #1 ' connect to the vision sensor connected to COM2, and set the number to 1.
3 EndIf
4 Wait M_NvOpen(1)=1      ' Connect to vision sensor No. 1 and wait until it has logged on.
5 NVLoad #1,"TEST" ' Load program "TEST".
6 NVRun #1,"TEST" ' Run program "TEST".
7 EBRead #1,,MNUM,PVS1,PVS2      ' Read the tag data of "Job.Robot.FormatString"
                                and save it in the variables MNUM, PVS1, and PVS2.
20 ...
21 NVClose #1              ' Disconnect from the vision sensor connected to COM2.
```

Comments

- 1) Reads data by specifying a tag name from an active vision program of a specified vision sensor.
- 2) Stores the data read by the vision sensor in a specified variable.
- 3) In cases where vision data consists of multiple values delimited by commas, the data is stored in the order that the specified variable names were enumerated and delimited in. In such cases, the data and variables should be of the same type.
- 4) When the position variable is specified, the components that have vision data set in them are X, Y, and C. The values of components that do not have any data set in them are set to "0". The value converted into the radian is set to the C component.
- 5) Values are only set in specified variables when the number of specified variables is less than the amount of data received.
- 6) Variables exceeding the amount of data are not updated when the number of specified variables exceed the amount of data received.
- 7) If the tag name is omitted, the setting value of parameter EBRDTAG is set instead. (The factory setting is "Job.Robot.FormatString".)

- 8) The time-out period can be specified with numeric values. Within the time-out period, the program will not move to the next step until it has received data from the vision sensor. However, this command will be suspended if the program is aborted. The program will resume once it has been restarted.
- 9) When this command is used with multi-tasking, it is necessary to execute the NVOpen command and NVRun command depending on the task. Use the number specified with the NVOpen command for the vision sensor number that is to be used.
- 10) Not supported if the program's startup settings are set to "ALWAYS" or the continuity function (CTN) is enabled.
- 11) If interruption conditions have been satisfied during this command, interruption processing will be executed immediately. Processing will be executed after interruption processing has complete.
- 12) In order to reduce tact time, other work can be done after executing the NVRun command and EBRead can be executed when required.
- 13) Set "1" in parameter NVTRGTMG if the EBRead command is on the line immediately after the NVRun command in the program.
If parameter NVTRGTMG is set to the factory setting (NVTRGTMG = "2"), the NVRun command starts processing the next command without waiting for completion of vision identification processing. Therefore, the results of previous identification data may be read if the EBRead command is still being executed.
- 14) Note that if the program stops between NVRun and EBRead, the results when NVRun is executed and the results when EBRead is executed may be different.

< Values set in variables >

Values substituted by variables when the EBRead command is executed are as follows.

(1) Content of specified tag (Pattern_1.Number_Found) is 10

- The value when "EBRead #1,"Pattern_1.Number_Found",MNUM" is executed is:
→MNUM=10
- The value when "EBRead #1,"Pattern_1.Number_Found",CNUM" is executed is:
→CMNUM="10"

(2) Content of specified tag (Job.Robot.FormatString) is: 2, 125.75, 130.5, -117.2, 55.1, 0, 16.2

- The value when "EBRead #1,,MNUM,PVS1,PVS2" is executed is:
→MNUM=2
PVS1.X=125.75 PVS1.Y=130.5 PVS1.C=-117.2
PVS2.X=55.1 PVS2.Y=0, PVS2.C=16.2
* The value of the component not set by vision data (excluding X, Y, and C components) is "0".
- The value when "EBRead #1,,MNUM,MX1,MY1,MC1,MX2,MY2,MC2" is executed is:
→MNUM=2
MX1=125.75 MY1=130.5 MC1=-117.2
MX2=55.1 MY2=0 MC2=16.2
- The value when "EBRead #1,,CNUM,CX1,CY1,CC1,CX2,CY2,CC2" is executed is:
→CNUM="2"
CX1="125.75" CY1="130.5" CC1="-117.2"
CX2="55.1" CY2="0" CC2="16.2"

(3) Content of specified tag (Job.Robot.FormatString) is: 2, 125.75, 130.5

- The value when "EBRead #1,,MNUM,PVS1,PVS2" is executed is:
→MNUM=2
PVS1.X=125.75 PVS1.Y=130.5 PVS1.C=-117.2
PVS2.X=55.1 PVS2.Y=0, PVS2.C=16.2
* The value of the component not set by vision data (excluding X, Y, and C components) is "0".

Errors

- 1) If the data types of arguments are different, the error "Syntax error in input command" will occur.
- 2) If there is a discrepancy with the number of arguments (too many or too few), the error "Incorrect argument count" will occur.
- 3) If the value specified for <Vision sensor No.> is anything other than 1 to 8, the error "Argument out of range" will occur.
- 4) If the NVOpen command is not executed with the number set in <Vision sensor No.>, the error "The NVOPEN command not is not executed" will occur.
- 5) If the string data format received from the vision sensor and format of the variable which substituted it are different, the error "Illegal Receive data (EBREAD)" will occur.
- 6) If the value specified for <Time out> is anything other than 1 to 32767, the error "Argument out of range" will occur.
- 7) If the vision sensor does not respond within the specified time, or within the first 10 seconds if the <Time out> parameter has been omitted, the error "Vision sensor response timeout" will occur.
- 8) If communication is disconnected while this command is being executed, the error "The communication is abnormal" will occur and the robot controller's port will close.
- 9) If the specified tag name does not exist in the active vision program, the error "Vision tag name is abnormal" will occur.
- 10) Specify no more than 31 variables. (No. of identifications + coordinate values XYZ × 10) If 32 variables or more are specified, the error "Syntax error in input command" will occur.
- 11) If <Vision program name> exceeds 15 characters, the error "Vision program name is abnormal" will occur.
- 12) If characters other than letters from A to Z, numbers from 0 to 9, a hyphen (-) or an underscore (_) are used in <Vision program name>, the error "Vision program name is abnormal" will occur.
- 13) If the program specified in <Vision program name> has not been loaded to the vision sensor, the error "Vision program does not exist" will occur.
- 14) If the program specified in <Vision program name> has not been started by the NVRun command, the error "Vision program name is abnormal" will occur.
- 15) If <Identification count cell>, <Start cell>, or <End cell> contains a number other than 0 to 399, or a letter other than A-Z, the error "Argument out of range" will occur.
- 16) If there is no value in the cell specified in <Identification count cell>, the error "Invalid value in identification count cell" will occur.
- 17) If <Start cell> and <End cell> are reversed, the error "Argument out of range" will occur.
- 18) If the amount of data included in the cell specified by <Start cell> and <End cell> exceeds 90 lines, the error "Specified cell value out of range" will occur.
- 19) If the range specified by <Start cell> and <End cell> exceeds line 30 and element 10, the error "Specified cell value out of range" will occur.
- 20) If the value of <Type> is not a number from 0 to 7, the error "Argument out of range" will occur.

EBWrite (EasyBuilder write)

Function

Specifies the tag name of the vision sensor to write data.

When the vision program (job) is made with the vision tool EasyBuilder made by Cognex Corporation, you can write data to the cell specified with the tag name by using this command.

Syntax

EBWrite #<Vision sensor No.>, [<Tag name>], <Writing data> [, <Time out>]

Term

<Vision sensor No.> (Cannot be omitted)

Specify the number of the vision sensor you want to control with a numeric constant. Setting range: 1 to 8

<Tag name> (Can be omitted)

Specify the name of the symbolic tag for the cell to which data is written.
If the name is not specified, the value of parameter EBWRTAG is set.

<Writing data> (Cannot be omitted)

Specify the data to be written to the vision sensor.

Numeric constants, numeric variables, string constants, string variables, position component data, joint component data, numerical expressions, and character string expressions can be used.

<Time out> (Can be omitted)

Specify the time-out time (in seconds) with a numeric constant. If the time is not specified, a timeout of 10 seconds is set.

Setting range: 1 to 32767 (integers)

Examples

```
1 If M_NvOpen(1)<>1 Then ' If vision sensor No. 1 is not logged on
2 NVOpen "COM2:" As #1 ' connect to the vision sensor connected to COM2, and set the number to 1.
3 Wait M_NvOpen(1) = 1 ' Wait until vision sensor No. 1 has logged on.
4 End If
5 NVOpen #1, "TEST" ' Load program (job) "TEST".
6 EBWrite #1, "Sample.Float", 5 ' Rewrite "Sample.Float" tag data as 5.
7 EBWrite #1, "Sample.String", "Test" ' Rewrite "Sample.String" tag data as "Test".
8 NVRun #1, "TEST" ' Run program (job) "TEST".
:
:
20 End
```

Comments

- 1) Writes data to the cell specified with the tag name in the active vision program (job) of the specified vision sensor.
- 2) The error (L.3141) occurs if no NVOpen command is executed for the vision sensor specified with <Vision sensor No.>.
- 3) If <Tag name> is not specified, the value of parameter EBWRTAG is set. (The factory setting is ""(NULL).)
- 4) The error (L.8637) occurs if the active vision program does not have the specified <Tag name>.

- 5) The type of the data written to the cell of the vision sensor varies depending on the type of <Writing data>.
 (When a double-precision real number is specified, the single-precision real number converted from the double-precision real number is used.)

<Writing data>	Type of data written to cells
Numeric value type (integer)	Integer (Int)
Numeric value type (real number)	Single-precision real number (Float)
Character string type	Character string (String)

- 6) Processes are performed according to the combinations of the types of <Writing data> and the cell value types of the vision program specified with <Tag name> as shown below.

<Writing data>	Cell value type	Process
Numeric value type (integer)	Boolean value editing control Integer editing control Floating-point number editing control Text editing control	Cell value update (integer) Cell value update (integer) Cell value update (single-precision real number) Execution error (L.8637)
Numeric value type (real number)	Boolean value editing control Integer editing control Floating-point number editing control Text editing control	Cell value update (integer, rounding down decimals) Cell value update (integer, rounding down decimals) Cell value update (single-precision real number) Execution error (L.8637)
Character string type	Boolean value editing control Integer editing control Floating-point number editing control Text editing control	Execution error (L.8637) Execution error (L.8637) Execution error (L.8637) Cell value update (character string)

- 7) You can specify the time-out time with a numeric constant. During the time-out time, the next step is not performed until data containing writing results is received from the vision sensor.
- 8) When the execution of a robot program is stopped, the processing of this command is interrupted. The interrupted processing restarts by executing the program again.
- 9) When this command is used with multi-tasking, it is necessary to execute the NVOpen command in the task slot you use. Use the number specified with NVOpen command for <Vision sensor number>.
- 10) This command cannot be used if ALWAYS is specified in the start conditions of the task slot.
- 11) If interruption conditions have been satisfied during this command, interruption processing will be executed immediately.

Appendix 2.2 Vision sensor status variables

The following is information on the status variable of the 2D vision sensor.
Please be aware that the data of the status variable is not backed up by RT ToolBox3's backup function.

Variable name	Array elements	Function	Attribute(*)	Data type
M_NvOpen	8	Port connection status	R	Integer

(*) "R" indicates that a status variable is read-only.

M_NvOpen

Function

Indicates the connection status of the port for the vision sensor

Array meaning

Array elements 1 to 8: Vision sensor numbers

Explanation of values returned

0: Connecting to port (logon not complete) 1: Logon complete -1: Not connected

Usage

After the NVOpen command is executed, M_NvOpen connects the vision sensor to the port and checks if the vision sensor has been logged on to.

Examples

```
1 If M_NvOpen(1)<>1 Then ' If vision sensor number 1 is not logged on
2 NVOpen "COM2:" As #1 ' connect to the vision sensor connected to COM2, and set the
  number to 1.
3 EndIf
4 Wait M_NvOpen(1)=1 ' Connect to vision sensor No. 1 and wait until it has logged on.
5 ...
10 NVClose #1          ' Disconnect from the vision sensor connected to COM2.
```

Comments

- 1) Indicates the connection status of the port connected to the vision sensor when it was opened with the NVOpen command.
- 2) The initial value is "-1". At the point in time when the NVOpen command is executed and the port is connected, the value changes to "0" (connecting to port). At the point in time when the vision sensor has successfully been logged on to, the value changes to "1" (logon complete).
- 3) This variable strongly resembles the status of status variable M_Open, but whereas M_Open changes to "1" when the connection is verified, M_NVOpen becomes "1" when the vision sensor is successfully logged on to.

Errors

- 1) If the type of data specified as an array element is incorrect, the error "Syntax error in input command" will occur.
- 2) If there is a discrepancy with the number of array elements (too many or too few), the error "Incorrect array element" will occur.
- 3) If an array other than 1 to 8 is specified, the error "Array element mistake" will occur.

Appendix 3. 2D vision sensor parameters

Parameter settings for use with vision sensors are listed below.

Parameter	Parameter name	No. of arrays	Details	Factory setting												
Username	NVUSER	Character string 1	Set a username to log on to the vision sensor. (15 characters or less)	"admin"												
Password	NVPSWD	Character string 1	Set a password to log on to the vision sensor. (15 characters or less)	""												
Trigger timing	NVTRGTMG	Integer 1	<p>Defines how the NVRun and NVTrg commands are processed. The processing details of each set value are as follows.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>NVRun</th> <th>NVTrg</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Trigger</td> <td>Trigger</td> </tr> <tr> <td>1</td> <td>Trigger + Image processing</td> <td>Trigger + Image processing</td> </tr> <tr> <td>2</td> <td>Trigger</td> <td>Trigger + Image processing</td> </tr> </tbody> </table> <ul style="list-style-type: none"> • Trigger: The next command is executed upon completion of communication between the vision sensor and image processing instruction (demand to take picture). Set this setting to reduce tact time if the robot performs other tasks while the vision sensor is processing image data. • Trigger + Image processing: A request to process the image data is sent to the vision sensor. Once image processing is complete, the next command is executed. Set this setting for the robot to request the vision sensor to process the image and for the robot to acquire the result of that processing in the next step (i.e. by executing the EBRead command). 	Value	NVRun	NVTrg	0	Trigger	Trigger	1	Trigger + Image processing	Trigger + Image processing	2	Trigger	Trigger + Image processing	2
Value	NVRun	NVTrg														
0	Trigger	Trigger														
1	Trigger + Image processing	Trigger + Image processing														
2	Trigger	Trigger + Image processing														
Initial value of tag name specified by the EBRead command	EBRDTAG	Character string 1	<p>Set the initial value of the symbolic tag name specified by EBRead command. (128 characters or less)</p> <p>If the tag name in EBRead command is omitted, the value of this parameter is specified.</p>	"Job.Robot.FormatString"												

Notes

Appendix 4. Troubleshooting

This chapter lists errors that may occur when using network vision sensors. It also explains the causes and solutions to these errors.

(Errors not described in this chapter may occur depending on the conditions and timing of error detection.)

For errors not described in the list, refer to the following manuals.

CR800 Series Controller INSTRUCTION MANUAL Troubleshooting (BFP-A3480)

CR750/CR751/CR760 Series Controller INSTRUCTION MANUAL Troubleshooting (BFP-A8871)

Appendix 4.1 Error numbers

When an error occurs, the ERROR LED on the front panel of the robot controller will turn on or flash.

ERROR LED status	Description
ON	Low level error or caution
Flashing	High level error
OFF	Operating normally

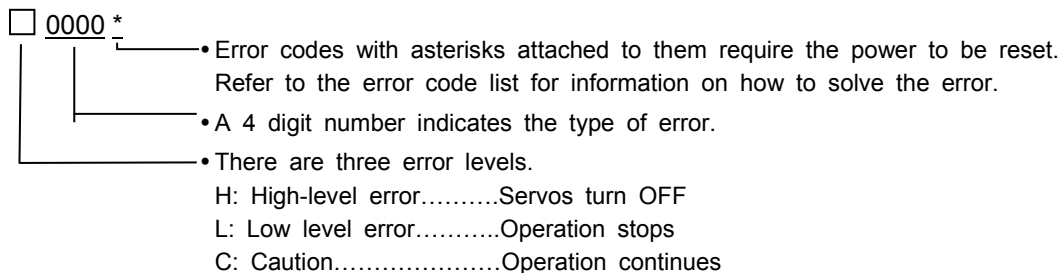
A 4 digit error number also appears on the teaching pendant's LCD screen. (The letter at the beginning of the error number is not displayed.) For example, when the error C0010 occurs, "0010" and an error message are displayed.

An alarm will also sound in 0.5 second intervals. When power is restored or when the time between power OFF and ON is too short, an alarm will sound in 0.1 second intervals (constantly).

Error codes, messages, causes and solutions can be found in Section 4.2.2 "2D vision sensor error code list".

Detailed information on the error number is displayed in the Error history screen of the teaching box. Check the Error history screen after any errors have been reset.

⚠ CAUTION Error codes are explained in the following figure.



"n" after the error code indicates the axis number.

Example: H0931J1 Axis motor overcurrent

Appendix 4.2 2D vision sensor error code list

The meanings of letters at the beginning of error codes are as follows: H: High level error, L: Low level error, C: Caution.

"n" at the end of last digit of the error number in this list indicates the axis number (1 to 8).

Error code	Causes and solutions	
L3110	Error message	Argument out of range
	Cause	One of the argument values specified in a command is out of range.
	Solution	Check argument range and re-enter the values if necessary.
L3120	Error message	Incorrect argument count
	Cause	The number of arguments in the executed command is incorrect.
	Solution	Check the number of arguments and re-enter them if necessary.
L3130	Error message	Attempt was made to open an already open communication file.
	Cause	The communication port is already open.
	Solution	Check the COM number and vision sensor number. Re-enter them if necessary. Or check communication parameters.
L3141	Error message	The NVOpen command is not executed.
	Cause	The NVOpen command was not executed before execution of a command communicating with the vision sensor.
	Solution	Revise the robot program to execute the NVOpen command.
L3142	Error message	The communication line cannot be opened.
	Cause	The port for communication with the vision sensor cannot be opened.
	Solution	Check the communication cable or communication parameters.
L3287	Error message	This command cannot be used if the start condition is ERR or ALW.
	Cause	This command cannot be used if the start condition is ERR or ALW.
	Solution	Revise the program
L3501	Error message	Illegal Receive data(EBRead)
	Cause	The type of data received by EBRead command is different from the type of variable specified.
	Solution	Revise the program.

Error code	Causes and solutions	
L3810	Error message	Incorrect argument type
	Cause	Either the arithmetic operator, monadic operator, or comparison operator is incorrect, or the arguments of each function are incorrect.
	Solution	Check the data of the specified vision sensor tag.
L4220	Error message	Syntax error in input command
	Cause	There is an error in the syntax of the input command.
	Solution	Check the program and correct the syntax if required.
L4370	Error message	Array element mistake
	Cause	1. The array element is outside of the defined range. 2. A variable that cannot be arrayed was specified.
	Solution	1. Edit the array elements so that they are within the array element limit. 2. Do not specify that variable.
L7810	Error message	Vision sensor not connected
	Cause	There is no vision sensor connected to the specified COM number.
	Solution	Check parameters such as NETHSTIP, NETPORT, and NETMODE.
L8600	Error message	Controller data corrupted
	Cause	The internal data of the controller has been corrupted.
	Solution	Check the vision sensor program number and settings of parameters such as COMDEV.
L8601	Error message	Logon not possible
	Cause	The communication port was opened, but there is no response from the vision sensor.
	Solution	Reset the program and start it again.
L8602	Error message	Wrong password
	Cause	The password for the user set in the parameter "NVUSER" is not set in the parameter "NVPSWD".
	Solution	Set the correct password
L8603	Error message	Parameter abnormality
	Cause	There is a problem with the username parameter or the password parameter.
	Solution	Check the parameters "NVUSER" and "NVPSWD".

Error code	Causes and solutions	
L8610	Error message	Abnormal communications
	Cause	Communication with the vision sensor was cut off before or during command execution.
	Solution	Check the communication cable connecting the robot and vision sensor.
L8620	Error message	Abnormal vision sensor number specification
	Cause	The specified vision sensor number is not defined with an NVOpen command.
	Solution	Check whether the vision sensor number is correct. Check that the number is also defined by an NVOpen command.
L8621	Error message	Abnormal vision program name
	Cause	The vision program name exceeds 15 characters.
	Solution	Specify a vision program name that does not exceed 15 characters.
L8622	Error message	Vision program not present
	Cause	The vision sensor cannot find the program.
	Solution	Check whether the program has been loaded to the vision sensor. Check whether the program name is correct.
L8632	Error message	Vision sensor response timeout
	Cause	There was no response from the vision sensor within the specified time limit.
	Solution	Check whether the specified time is correct. Or check that the vision sensor settings are correct.
L8633	Error message	NVTRG response timeout
	Cause	No response to image capture request.
	Solution	Check the communication cable.
L8636	Error message	Vision tag name is abnormal
	Cause	The symbolic tag name does not exist in the active vision program.
	Solution	Check that the symbolic tag name of Easy Builder is the same as the tag name specified by the robot program. If it is not the same, correct it.

Error code	Causes and solutions	
L8640	Error message	Abnormal image capture specification
	Cause	The image capture settings are set to something other than "Camera", "External", or "Manual".
	Solution	Change the image capture settings to "Camera", "External", or "Manual".
L8650	Error message	Put online
	Cause	The vision sensor is offline.
	Solution	Turn the vision sensor online and enable external control.
L8660	Error message	Not permitted to control vision sensor
	Cause	The NVUSER and NVPSWD parameters set for logging on to the vision sensor do not have full access rights to logon to the vision sensor.
	Solution	Check the list of registered users for the vision sensor and specify which users are allowed full access in parameters NVUSER and NVPSWD.
L8670	Error message	Restart not possible after stop
	Cause	The program was started without being reset after it stopped.
	Solution	Reset the robot program then start it.

Mitsubishi Electric Industrial Robot

