FANUC Robot series

R-30*i*B Plus/R-30*i*B Mate Plus/R-30*i*B Compact Plus/
R-30*i*B Mini Plus CONTROLLER

*i*RPickTool

OPERATOR'S MANUAL

B-83924EN/03

- **Original Instructions**

Thank you very much for purchasing FANUC Robot.
Before using the Robot, be sure to read the "FANUC Robot series SAFETY HANDBOOK (B-80687EN)" and understand the content.

The products in this manual are controlled based on Japan's "Foreign Exchange and Foreign Trade Law". The export from Japan may be subject to an export license by the government of Japan.
Further, re-export to another country may be subject to the license of the government of the country from where the product is re-exported. Furthermore, the product may also be controlled by re-export regulations of the United States government.
Should you wish to export or re-export these products, please contact FANUC for advice.

In this manual, we endeavor to include all pertinent matters. There are, however, a very large number of operations that must not or cannot be performed, and if the manual contained them all, it would be enormous in volume. It is, therefore, requested to assume that any operations that are not explicitly described as being possible are "not possible".

SAFETY PRECAUTIONS

This chapter describes the precautions which must be followed to enable the safe use of the robot. Before using the robot, be sure to read this chapter thoroughly.

For detailed functions of the robot operation, read the relevant operator's manual to understand fully its specification.

For the safety of the operator and the system, follow all safety precautions when operating a robot and its peripheral equipment installed in a work cell.
For safe use of FANUC robots, you must read and follow the instructions in the “FANUC Robot series SAFETY HANDBOOK (**B-80687EN)**”.

1 PERSONNEL

Personnel can be classified as follows.

Operator:
- Turns the robot controller power ON/OFF
- Starts the robot program from operator panel

Programmer or Teaching operator:
- Operates the robot
- Teaches the robot inside the safeguarded space

Maintenance technician:
- Operates the robot
- Teaches the robot inside the safeguarded space
- Performs maintenance (repair, adjustment, replacement)

- The operator is not allowed to work in the safeguarded space.
- The programmer or teaching operator and maintenance technician are allowed to work in the safeguarded space. Work carried out in the safeguarded space include transportation, installation, teaching, adjustment, and maintenance.
- To work inside the safeguarded space, the person must be trained on proper robot operation.

Table 1 (a) lists the work outside the safeguarded space. In this table, the symbol "O" means the work allowed to be carried out by the specified personnel.

Table 1 (a) List of work outside the Safeguarded Space

	Operator	Programmer or Teaching operator	Maintenance technician
Turn power ON/OFF to Robot controller	O	O	O
Select operating mode (AUTO/T1/T2)		O	O
Select remote/local mode		O	O
Select robot program with teach pendant		O	O
Select robot program with external device		O	O
Start robot program with operator's panel	O	O	O
Start robot program with teach pendant		O	O
Reset alarm with operator's panel		O	O
Reset alarm with teach pendant		O	O
Set data on teach pendant		O	O
Teaching with teach pendant		O	O
Emergency stop with operator's panel	O	O	O
Emergency stop with teach pendant	O	O	O
Operator's panel maintenance			O
Teach pendant maintenance			O

During robot operation, programming and maintenance, the operator, programmer, teaching operator and maintenance technician take care of their safety using at least the following safety protectors:

- Use clothes, uniform, overall adequate for the work
- Safety shoes
- Helmet

2 DEFINITION OF SAFETY NOTATIONS

To ensure the safety of users and prevent damage to the machine, this manual indicates each precaution on safety with "**WARNING**" or "**CAUTION**" according to its severity. Supplementary information is indicated by "**NOTE**". Read the contents of each **"WARNING"**, "**CAUTION**" and "**NOTE**" before using the robot.

Symbol	Definitions
⚠**WARNING**	Used if hazard resulting in the death or serious injury of the user will be expected to occur if he or she fails to follow the approved procedure.
⚠**CAUTION**	Used if a hazard resulting in the minor or moderate injury of the user, or equipment damage may be expected to occur if he or she fails to follow the approved procedure.
NOTE	Used if a supplementary explanation not related to any of WARNING and CAUTION is to be indicated.

TABLE OF CONTENTS

1 PREFACE

This chapter describes an overview of this manual which should be noted before operating the *i*RPickTool function.

1.1 OVERVIEW OF THE MANUAL

This manual describes how to operate *i*RPickTool on the R-30*i*B Plus controller, R-30*i*B Mate Plus controller, R-30*i*B Compact Plus controller, and R-30*i*B Mini Plus controller. This manual describes each function which is provided by *i*RPickTool. When you would like to start up the robot system which uses *i*RVision, please refer to manuals which are introduced in 1.3 “RELATED MANUALS”.

> ⚠ **CAUTION**
> This manual is based on the R-30*i*B Plus system software version 7DF5/14. Note that the functions and settings not described in this manual may be available, and some notation differences are present, depending on the software version.

1.2 HOW TO USE THIS MANUAL

This manual explains the procedures, etc. to program tracking systems that require vision or queue management.
For the '*i*RPickTool Basic' startup procedures, please read Chapter 3.
For the '*i*RPickTool' startup procedures, please read Chapter 4.
Please refer to section 2.1, “CONFIGURATION OF SOFTWARE OPTIONS” for the difference between the “iRPickTool basic” and the “iRPickTool”.
Chapter 5, 6 and 7 explain the details of the procedures and functions that are not explained in the startup procedures in Chapters 3 and 4.
Furthermore, Chapter 8 explains specific setting examples that suit each application, and Chapter 9 explains other useful functions of *i*RPickTool. After you have completed *i*RPickTool startup, refer to this manual in accordance with your intended use.

Contents of this manual

Chapter	Title	Details
Chapter 1	PREFACE	Gives an explanation of this manual and related manuals. In particular, please read the precautions.
Chapter 2	OVERVIEW	Gives an overview of *i*RPickTool and some configuration examples using *i*RPickTool.
Chapter 3	*i*RPICKTOOL BASIC (BASIC FUNCTIONS) STARTUP PROCEDURES	Explains from preparation to setup for starting up *i*RPickTool Basic. Refer to this chapter if you are using *i*RPickTool Basic for the first time.
Chapter 4	*i*RPICKTOOL (BASIC FUNCTIONS + PLUG & PLAY) STARTUP PROCEDURES	Explains from preparation to setup for starting up *i*RPickTool. Refer to this chapter if you are using *i*RPickTool that includes Plug & Play for the first time.
Chapter 5	OTHER PREPARATIONS BEFORE SETUP	Explains preparations other than the preparations in Chapter 3 and 4.
Chapter 6	BASIC FUNCTIONS REFERENCE	Explains each function of the basic functions and details on screen items.
Chapter 7	PLUG & PLAY REFERENCE	Explains each function of Plug & Play and gives details on tree-view items.
Chapter 8	EXAMPLES OF SETUP IN ACCORDANCE WITH USE	Explains customizations for advanced robot functionality iRPickTool, and gives specific examples.
Chapter 9	OTHER USEFUL FUNCTIONS	Explains useful functions of *i*RPickTool other than the functions explained up to Chapter 8.
Chapter 10	TROUBLESHOOTING	Explains common problems that may occur in tracking systems created using *i*RPickTool, and measures for resolving them.
Appendix A to I	The appendices explain the following items. System variable files / Setting up a teaching PC / Setting up HDI / Pulsecoders and their connection / Correspondence table for KAREL used in the previous software and in *i*RPickTool / Sensor task customization / External Machine Vision Interface Add-on	

Indications in this Manual

The symbol below is used in this manual. Please refer to it when looking for information.

Symbol	Description
Memo	Gives information that will provide hints for performing screen operations, and information that will provide a reference for function explanations and setting details.

1.3 RELATED MANUALS

This section introduces related manual.

Manual	Spec. No.	Description
OPERATOR'S MANUAL (Basic Operation)	B-83284EN	This is the main manual of the controller. This manual describes the following items for manipulating parts with the robot: • Setting the system for manipulating parts • Operating the robot • Creating and changing a program • Executing a program • Status indications • Backup and restore robot programs. This manual is used on an applicable design, robot installation, robot teaching.

Manual	Spec. No.	Description
MAINTENANCE MANUAL	B-83195EN	This manual describes the maintenance and connection of R-30*i*B/R-30*i*B Plus Controller.
	B-83525EN	This manual describes the maintenance and connection of R-30*i*B Mate/R-30*i*B Mate Plus Controller.
	B-83555EN	This manual describes the maintenance and connection of R-30*i*B Mate/R-30*i*B Mate Plus Controller (Open Air).
	B-84035EN	This manual describes the maintenance and connection of the R-30*i*B Compact Plus Controller.
	B-84175EN	This manual describes the maintenance and connection of the R-30*i*B Mini Plus Controller.
OPERATOR'S MANUAL (Alarm Code List)	B-83284EN-1	This manual describes the error code listings, causes, and remedies.
Optional Function OPERATOR'S MANUAL	B-83284EN-2	This manual describes the software optional functions.
Sensor Mechanical Unit / Control Unit OPERATOR'S MANUAL	B-83984EN	This manual describes the connection between sensors which is a camera or 3D Laser Sensor and a controller, and maintenance of sensors.
*i*RVision OPERATOR'S MANUAL (Reference)	B-83914EN	This manual is the reference manual for *i*RVision. This manual describes each function which is provided by *i*RVision. This manual describes the meanings (e.g. the items on *i*RVision setup screen, the arguments of the instruction, and so on.
*i*RVision 2D Camera OPERATOR'S MANUAL	B-83914EN-2	This manual is desired to first refer to when you start up systems of *i*RVision 2D Compensation and 2.5D Compensation. This manual describes startup procedures of *i*RVision 2D Compensation and 2.5D Compensation system, creating programs, caution, technical know-how, response to several cases, and so on.
*i*RVision Inspection Application OPERATOR'S MANUAL	B-83914EN-5	This manual is desired to first refer to when you start up systems of inspection which uses *i*RVision. This manual describes startup procedures of inspection system which uses *i*RVision, creating programs, caution, technical know-how, response to several cases, and so on.
Ethernet Function OPERATOR'S MANUAL	B-82974EN	This manual describes the robot networking options such as FTP, RIPE, *i*RConnect, PC Share, and so on.
Line Tracking OPERATOR'S MANUAL	B-83474EN	This manual describes the line tracking function which the robot system with *i*RPickTool is based on.
*i*RPickTool (Auto Visual Track Frame Setup QUICK SETTING GUIDE)	B-83924EN-1	This manual describes how to start up tracking system with *i*RPickTool by using Auto Visual Tracking Frame Setup function.

1.4 REGARDING MICROSOFT® PRODUCTS

The abbreviations below are used in this manual
Microsoft® Windows® 7 Professional: Windows 7
Windows® operating system: Windows

Regarding trademarks

Windows, Windows XP, Windows 7 Professional and Internet Explorer are registered trademarks or trademarks of Microsoft Corporation in the United States and in all other countries.

2 OVERVIEW

*i*RPickTool is robot software that has been developed for applications where the robot picks / places parts on a conveyor while tracking the motion of the conveyor.
It enables the robot to pick / place parts on a moving conveyor intelligently.
This is achieved by combining motion synchronization with parts on a moving conveyor (linear or circular), detecting those parts with a camera or photoelectric sensor, and, if multiple robots are utilized, the load balancing of the incoming or outgoing part flow.
It also supports the following kinds of applications.

- 'Conveyor to conveyor' applications, where each robot picks parts from one conveyor and places them on another conveyor.
- Applications such as 'conveyor to fixed station' and 'fixed station to conveyor', where the infeed or outfeed location is not a conveyor but a fixed zone (fixed station).

Basically, the following software options are available as *i*RPickTool configurations:

1 *i*RPickTool Basic
2 *i*RPickTool
3 *i*RPickTool Basic + *i*RVision options (Visual tracking)
4 *i*RPickTool + *i*RVision options (Visual tracking)

"4D Graphics" option and "*i*RPickTool / Option Package" option can be added for each combination.

Visual tracking is an application which detects parts, in particular, using a camera. An application which detects parts using a sensor such as a photo eye instead of using a camera is called queue managed tracking in this manual. The functions required for enabling this application are contained in "*i*RPickTool basic" and "*i*RPickTool" as standard. The difference between "*i*RPickTool basic" and "*i*RPickTool" is described in Section 2.1, "CONFIGURATION OF SOFTWARE OPTIONS" later. The main difference is that *i*RPickTool Plug-&-Play provides the Teach pendant (TP) programs that utilize the data setup in the menus with the goal of providing a "plug and play" philosophy for users.

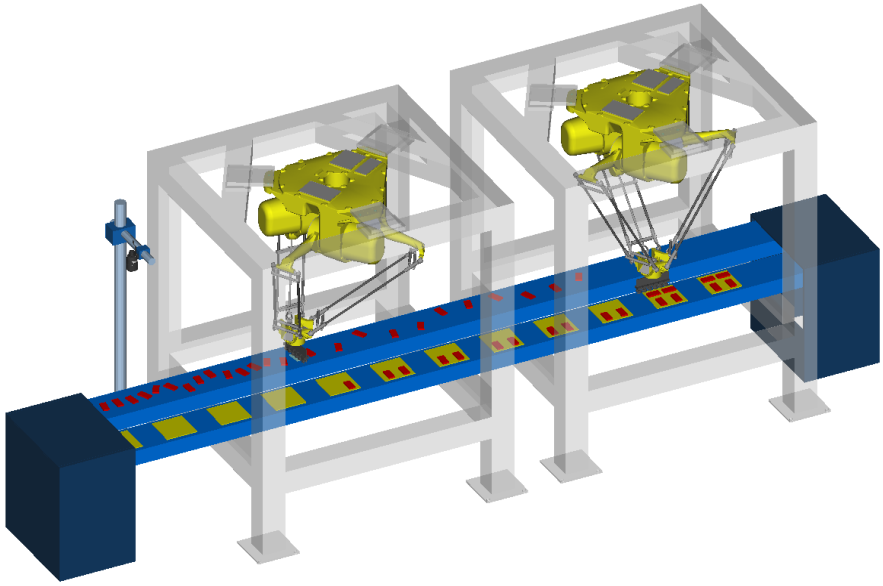


Fig. 2(a) Example of use of visual tracking
(Camera on the left side is used)

2.1 CONFIGURATION OF SOFTWARE OPTIONS

*i*RPickTool has the following software configuration.

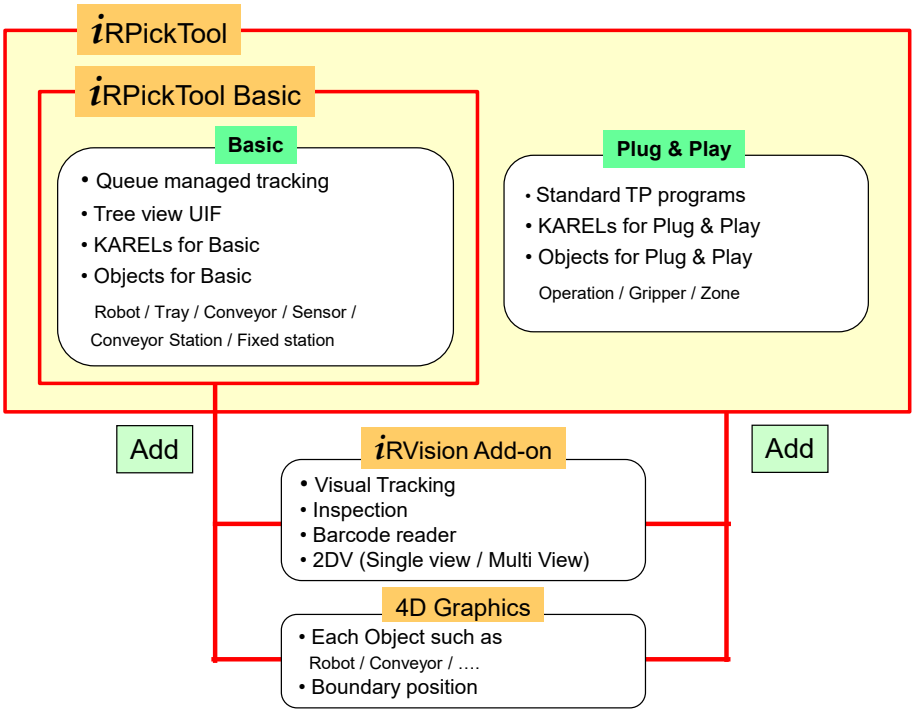


Fig. 2.1(a) *i*RPickTool software configuration diagram

*i*RPickTool basic contains basic functions for building a queue managed tracking system. You can build a visual tracking system using a camera by adding *i*RVision options to *i*RPickTool basic.
*i*RPickTool contains standard TP programs and others in addition to the basic functions contained in *i*RPickTool basic to make setup easier. The functions for making setup easier are called Plug & Play functions. The difference between *i*RPickTool and *i*RPickTool basic is whether the Plug & Play functions are contained.

> ⚠ **CAUTION**
> When *i*RPickTool (J945) is installed, *i*RPickTool basic (J944) cannot be used. For example, it is not possible to load and use a backup of *i*RPickTool basic (J944) on a robot controller in which *i*RPickTool (J945) is installed.

2.2 CONFIGURATIONS

This section describes device configurations of a queue managed tracking system and a visual tracking system using *i*RVision.

2.2.1 Queue Managed Tracking

A queue managed tracking system consists of the following devices.

Devices required for system running

- Robot controller
- Robot
- Conveyor
- Photo eye
- Pulsecoder
- Line tracking interface board (installed in the robot controller)
- Pulsecoder cable
- Pulse multiplexer (required when multiple robots are used and the Ethernet Encoder is not used. Pulsecoder to multiplexer cables and multiplexer cables are also required instead of Pulsecoder cables.)
- Ethernet communication cable
- Ethernet hub

Devices required only for system start-up

- Teach Pendant with a touch panel (or a personal computer for setup and Ethernet communication cable)
- Pointer

Sample configurations

When the Ethernet Encoder is used

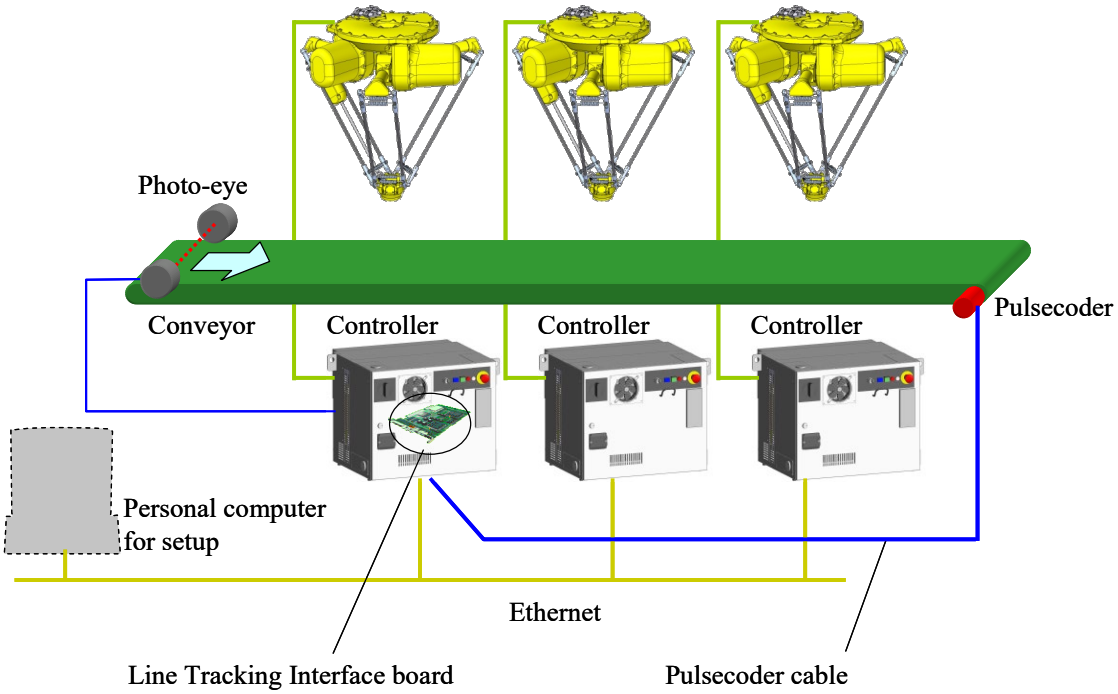


Fig. 2.2.1 (a)

When a pulse multiplexer is used

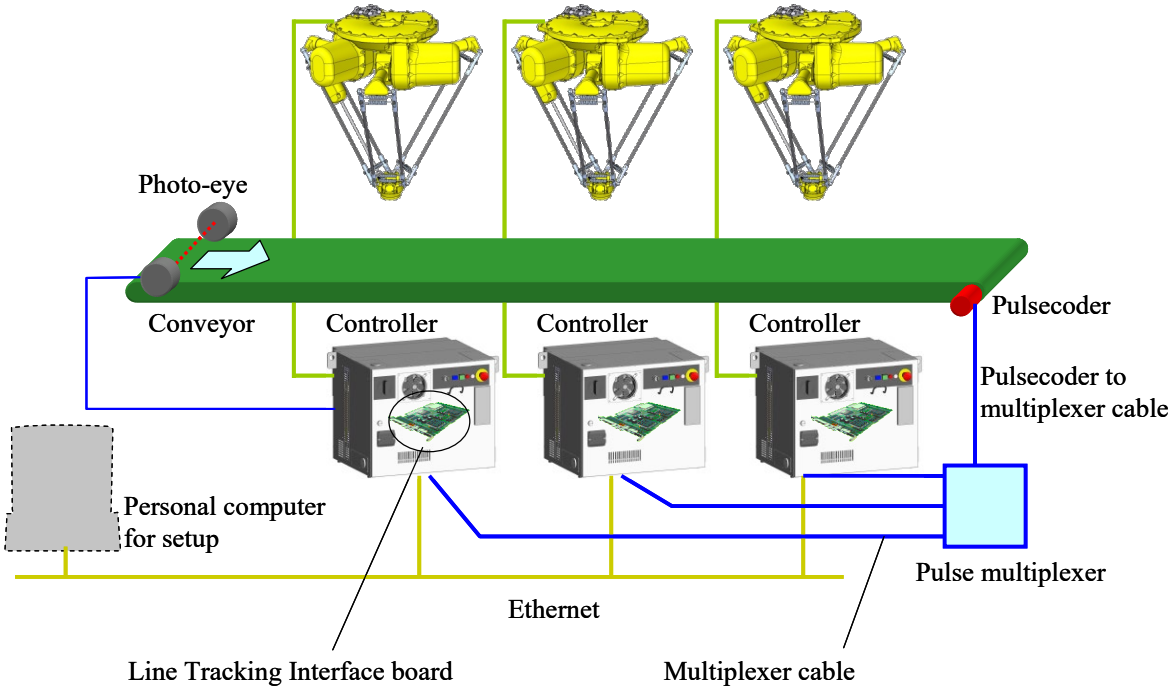


Fig. 2.2.1 (b)

2.2.2 Visual Tracking

A visual tracking system consists of the following devices.

Devices required for system running

- Robot controller
- Robot
- Conveyor
- Photo eye (required to be used as a trigger sensor for snapping an image using a camera)
- Camera
- Lens
- Camera cable
- Lighting
- Pulsecoder
- Line tracking interface board (installed in the robot controller)
- Pulsecoder cable
- Pulse multiplexer (required when multiple robots are used and the Ethernet Encoder is not used. Pulsecoder to multiplexer cables and multiplexer cables are also required instead of Pulsecoder cables.)
- Ethernet communication cable
- Ethernet hub

Devices required for only for system start-up

- Teach Pendant with a touch panel (or a personal computer for setup and Ethernet communication cable)
- Grid pattern (camera calibration plate)
- Pointer

Sample configurations

When the Ethernet Encoder is used

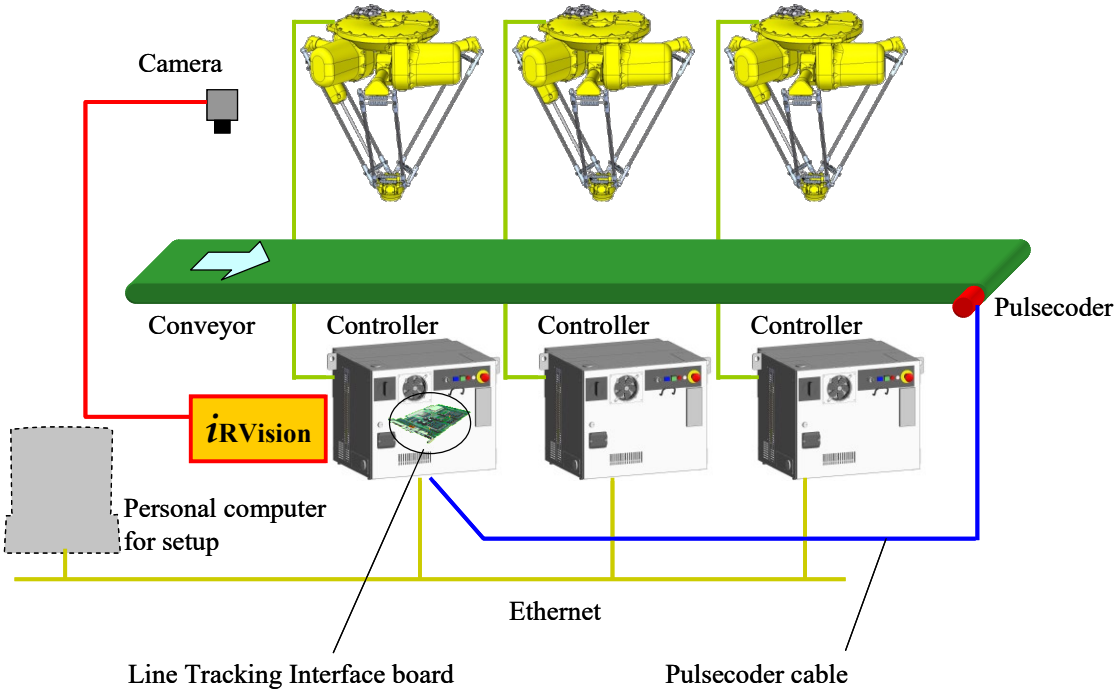


Fig. 2.2.2 (a)

When a pulse multiplexer is used

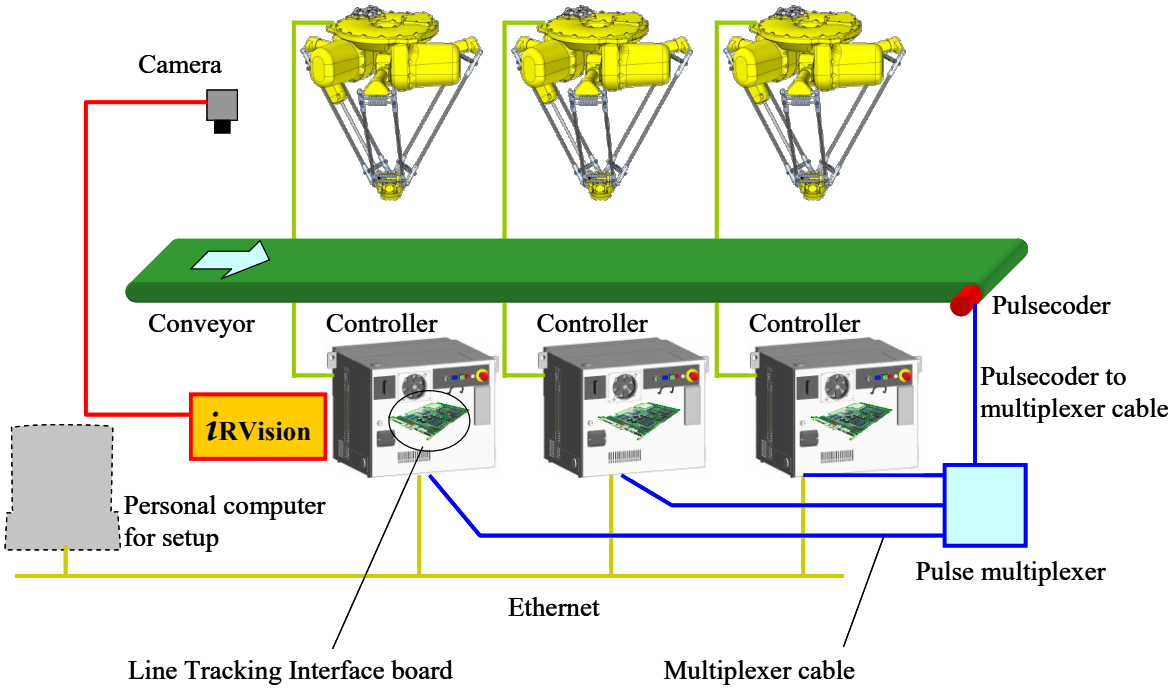


Fig. 2.2.2 (b)

2.3 KEY CONCEPTS

This section describes terms used in this manual.
The following concepts are used for building a tracking system using *i*RPickTool:

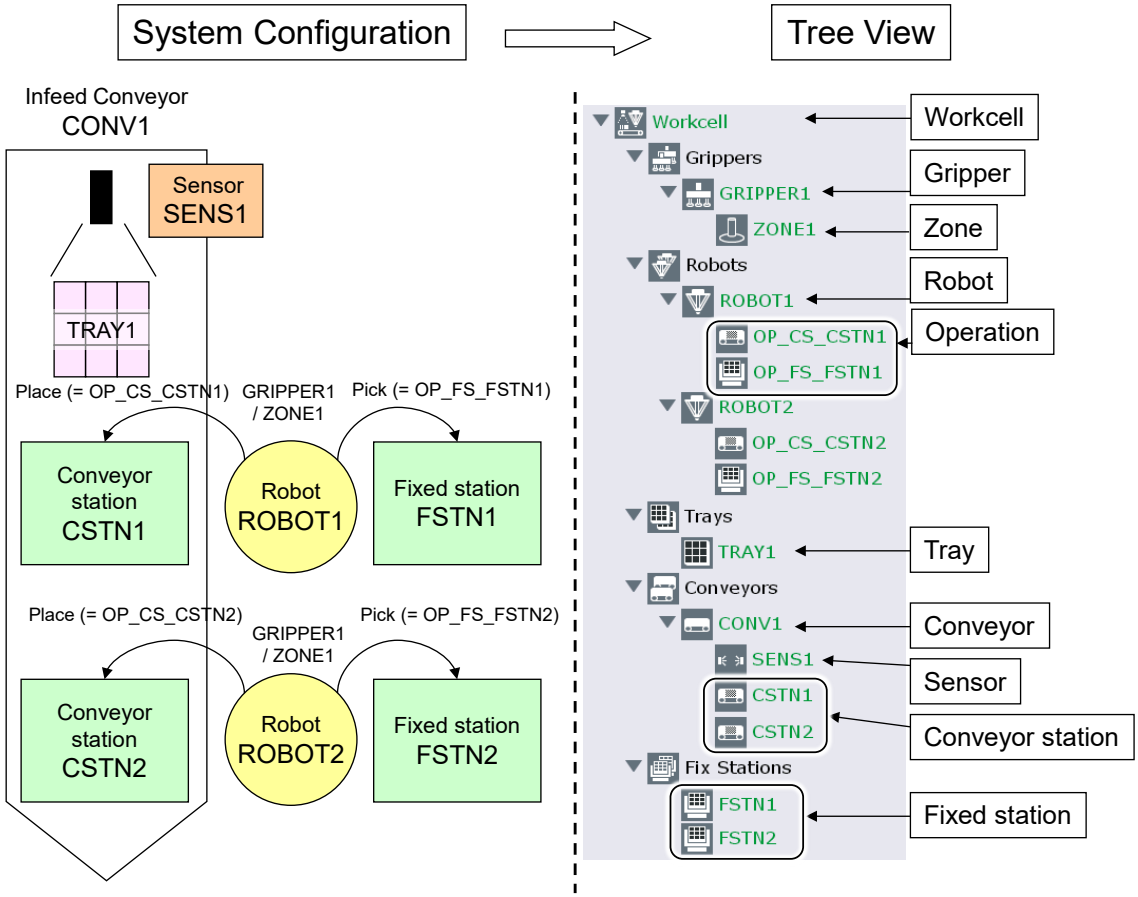


Fig. 2.3 (a) System configuration and *i*RPickTool tree view display

With *i*RPickTool, set items using the tree view shown in Fig. 2.3 (a). Each component used in this tree view is called an "object" Sequentially set objects from the top in the tree view.

[Work cell]

A work cell is an object indicating the entire tracking system configuration. Only one work cell can be set for each robot controller. If you want to change the system configuration, for example, for changing the setup, replace the entire work cell using a recipe described later.

[Gripper]

A gripper is an object indicating a gripper installed on the faceplate of a robot. It is contained only in Plug & Play and is referenced from a robot described later. When robots set in the same work cell have different grippers, it is necessary to set these grippers individually. On the other hand, when the robots have the same gripper, the setting of the same gripper can be shared among the robots. A Gripper contains one or more Zones.

[Zone]

A zone is an object belonging to a gripper and indicating a holding section of the gripper. It is contained only in Plug & Play. For example, when a gripper has a two-system holding section (picks up two parts), set two zones. The gripper shown in the figure below has six suction cups and picks up a part three times

with the three air systems. In this case, three zones are required. Each zone of the gripper contains a UTOOL value, a set of activation I/O values (for gripper open and close) and a set of part presence I/O values.

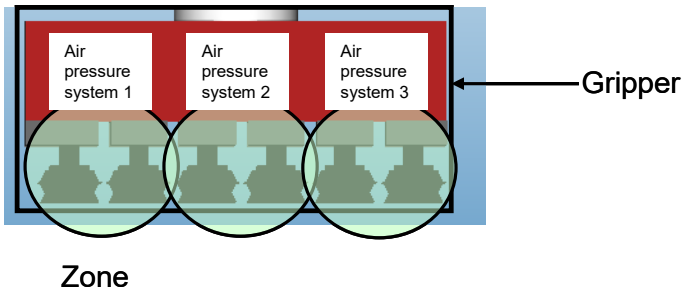


Fig. 2.3 (b) Grippers and zones

[Robot]

A robot is an object indicating a robot that performs tasks. It is necessary to create as many robot objects as the number of robots. The robot is used in a station (conveyor or fixed).

[Operation]

An operation is an object that belongs to the robot. It is included only in Plug & Play. It is an object that represents the pick / place operations of the robot at a conveyor station and fixed station, which are described later.
There is a one-to-one correspondence between a conveyor station and an operation or between a fixed station and an operation. When a conveyor or fixed station is generated, an operation is also generated automatically.

[Tray]

In some systems, parts picked up from an infeed conveyor may be placed in a "box" flowing on an outfeed conveyor. Generally, multiple parts are placed in a "box" A tray is the tree-view object that represents this "box".

[Cell]

Each place where a part is placed (or picked up) is called a cell. For example, a box in which four parts can be laid out is defined as a tray pattern having four cells.

[Tray frame]

A tray frame is a frame used to define a cell layout in a tray. The position and orientation of each cell are set as coordinates in the tray frame. A tray frame can be defined in any position in the tray.

Defining a tray pattern and teaching the robots the position of only one cell in the tray pattern can make the robots place parts in or pick up them from other cells. The cell in which a robot places (or from which the robot picks up) a part is managed, so multiple robots can be made to share the work related to the tray.

An example of a tray having four cells is given below, which will help you understand a tray more specifically.

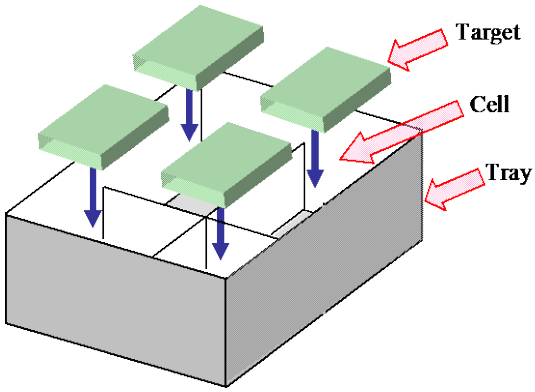


Fig. 2.3 (c) Example: A tray that has four cells

In the example in Figure 2.3 (c), four parts can be placed in the box (or can be picked from the box). The box itself is called a "tray" and each empty location at which to place (or from which to pick up) a part in the box is called a "cell" A non-empty cell containing a part is called "target". That is, this tray has four cells and is represented as the figure below. Figure 2.3 (d) is Figure 2.3 (c) seen from above. In this example, a tray frame is defined at the lower-left corner of the box.

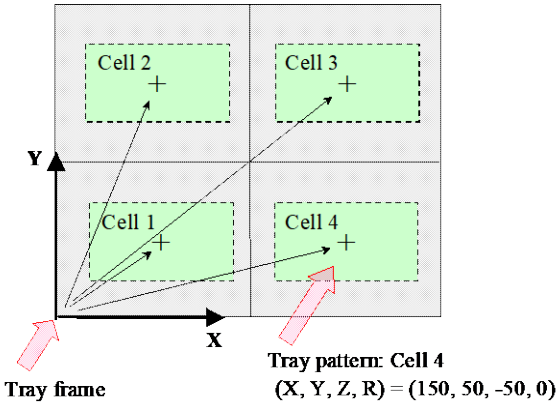


Fig. 2.3 (d) An example of overlaying a frame on a tray that has four cells

The + mark indicates the center of each cell. The position of each cell is set as the relative position viewed from the tray frame.

The tray frame can be determined in any position in the tray without restrictions. To teach a robot the position of the tray frame, however, determine the position where the touch-up operation can be performed easily, such as the corner of a box, because the origin of the tray frame and a point along the X-axis must be touched up to set the TCP of each robot.

[Conveyor]

A conveyor is an ordered collection of the sensor and conveyor station objects (described later). A conveyor object indicates which sensor detects a part flowing on the conveyor and how the part sequentially flows from a conveyor station to another conveyor station. A conveyor object virtually represents the actual conveyor status.

[Sensor]

A sensor is an object which detects parts flowing on a conveyor and inputs the detection result to the most upstream conveyor station defined by the relevant conveyor object. It is available both for detecting parts using a vision system and for detecting them using a sensor such as a photo eye without using a vision system.

NOTE
In an actual system, a KAREL program is automatically executed to perform processing according to the "sensor" setting. You can customize this KAREL program if needed, but the source code of the program is not provided as standard. If you need to customize the program, refer to "APPENDIX H SENSOR CUSTOMIZATION" or contact your local FANUC representative.

[Conveyor station]

A conveyor station is an object indicating an area on a conveyor in which a robot performs an operation on a moving part. The robot picks up a part for a conveyor station and places it in another conveyor station or fixed station. A conveyor station has tracking frame and tracking boundaries that limit the picking area.

An example is given below, which will help you understand conveyors, sensors, and conveyor stations more specifically.

Figure 2.3 (e) shows a visual tracking application consisting of two robots and two conveyors. Conveyor 1 is used as an infeed conveyor and conveyor 2 is used as an outfeed conveyor. Both robots 1 and 2 pick up parts from conveyor 1 and place them on conveyor 2. One camera is installed at the upstream end of conveyor 1. This camera is used to detect each flowing part. One photo eye is installed at the upstream end of conveyor 2. This photo eye is used to detect each flowing box. This system picks up parts from conveyor 1 and places them in boxes flowing on conveyor 2.

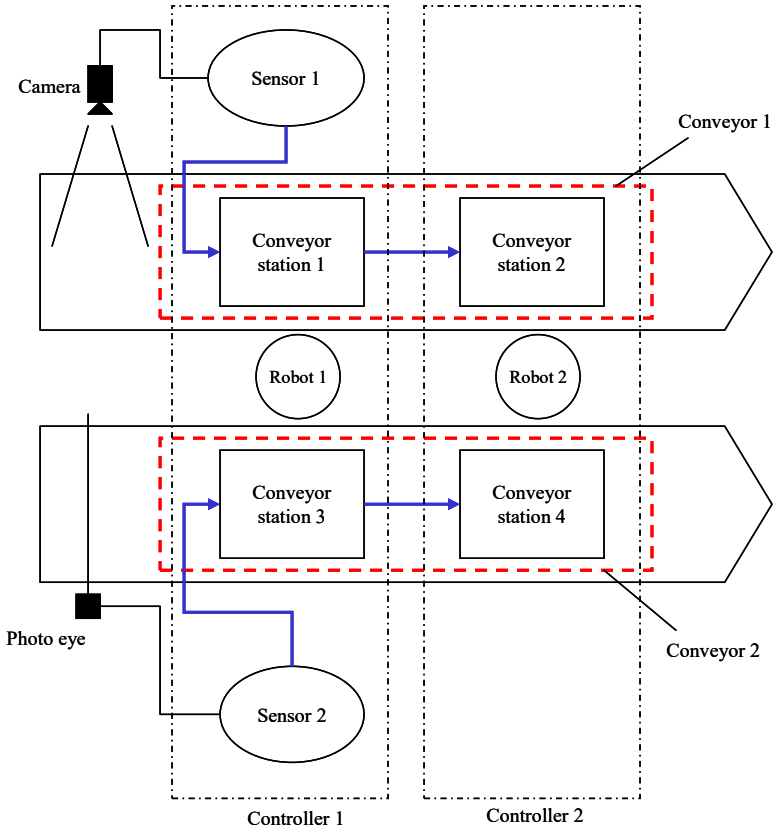


Fig. 2.3 (e) Conveyor station system example 1

This system has two conveyors, conveyors 1 and 2, and four conveyor stations, conveyor stations 1 to 4. Conveyor stations 1 and 2 are defined on conveyor 1 and conveyor stations 3 and 4 are defined on conveyor 2. Robot 1 picks up parts from conveyor station 1 on conveyor 1 and places them in conveyor station 3 on conveyor 2. Robot 2 picks up parts from conveyor station 2 on conveyor 1 and places them in conveyor station 4 on conveyor 2.

Sensor 1 detects parts on conveyor 1 using a camera. The detected part information is input to conveyor 1.
Sensor 2 detects parts on conveyor 2 using a photoelectric tube sensor. The detected part information is input to conveyor 3.
The information of the part that sensor 1 inputs to conveyor station 1 flows from conveyor station 1 to conveyor station 2 as the part flows. Similarly, the information of the part that sensor 2 inputs to conveyor station 3 flows from conveyor station 3 to conveyor station 4.

Each robot receives the information of the flowing parts from the conveyor station, and picks up a part or places it into a box. If a part does not flow to a conveyor station, its information is not passed to the robot. If a part passes through a conveyor station, its information is sent to the next conveyor station. The information of the flowing parts is given to each robot based on information including the position of each part and predetermined allocation ratio set for each conveyor station.

Generally, one conveyor object is defined for one physical conveyor. If two sensors are arranged on the same conveyor in series as shown in the figure below, it could still be modeled as single conveyor. However, you can also model it as two different conveyor objects for one physical conveyor.

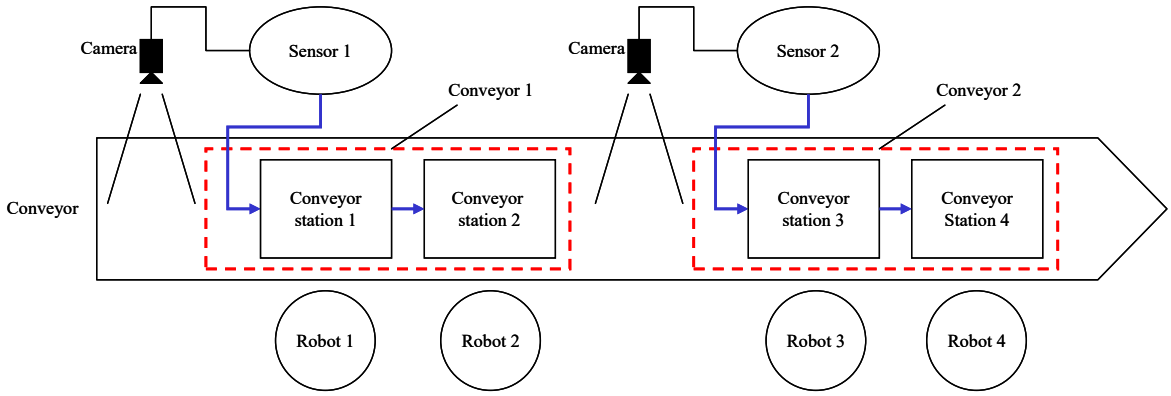


Fig. 2.3 (f) Conveyor station system example 2

Line track schedule

This is a data such as Track Frame for an operation in a conveyor station. The maximum number of the line track schedule is eight. When the line track schedule is set in a TP program as the attribute, a robot can track a part based on the data.

[Fixed station]

While a conveyor station indicates a robot work area defined on a "moving" conveyor, a fixed station is an object indicating a work area defined in an "unmoving" place. In the same way as for a conveyor station, a robot can pick up parts from a fixed station and place them in a fixed station or a conveyor station.

In the following figure, the outfeed conveyor in the configuration in Fig. 2.3 (e) is replaced with fixed stations. Both robots 1 and 2 pick up parts from conveyor 1 and place them on fixed stations 1 and 2. In this system, a box used for placing picked up parts is set in the same place.

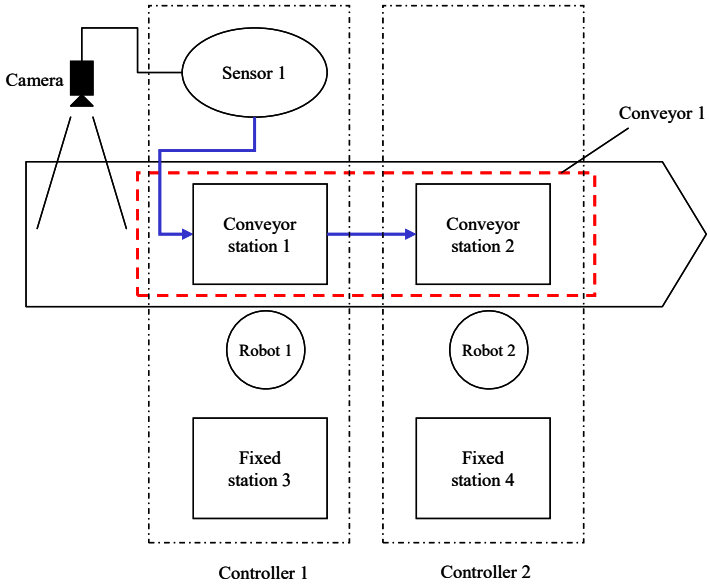


Fig. 2.3 (g) Fixed station system example

2.4 SAMPLE SYSTEM CONFIGURATIONS

This section introduces some sample system configurations to help you understand *i*RPickTool system configurations more specifically.

Sample configuration 1

This is the most basic configuration. Each robot picks up parts flowing on the infeed conveyor and places them on the outfeed conveyor. This configuration is typically called “parallel flow”.

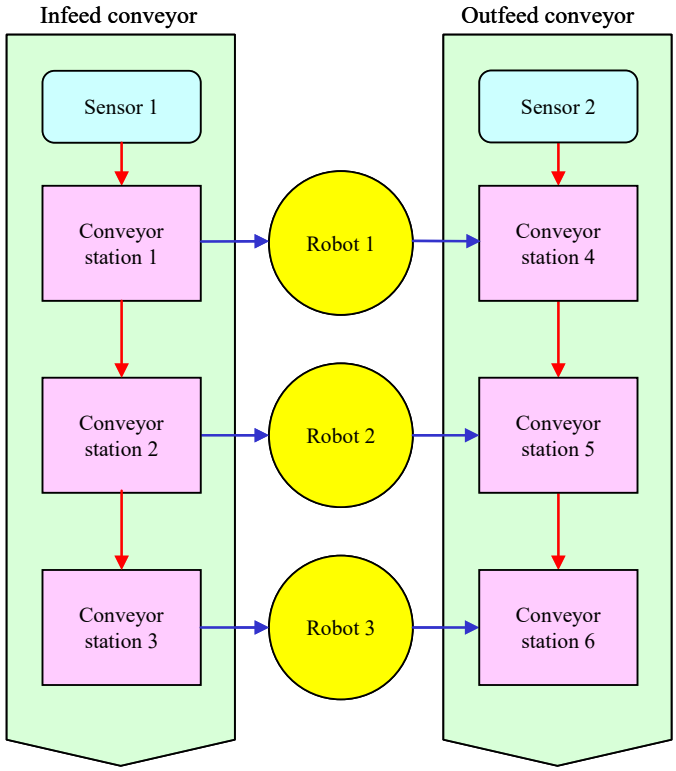


Fig. 2.4 (a)

Sample configuration 2

In this sample, the outfeed conveyor in sample configuration 1 is replaced with fixed stations. A configuration reverse of this sample may also be possible, in which fixed stations are used for pick up and a conveyor is used for placement.

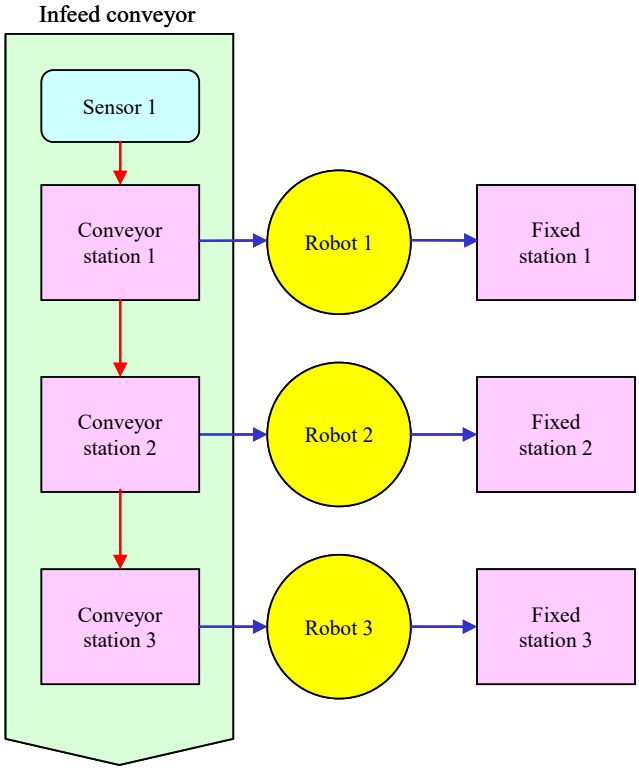


Fig. 2.4 (b)

Sample configuration 3

In this sample, fixed stations are added as buffers to sample configuration 1.

If parts are supplied from the infeed conveyor irregularly, then when the supply becomes too much, the parts are temporarily placed and stored at the fixed station. These fixed stations function as “buffers”.
If the supply of parts runs short, the robot picks up parts from the fixed station and places them in the conveyor station of the outfeed conveyor.

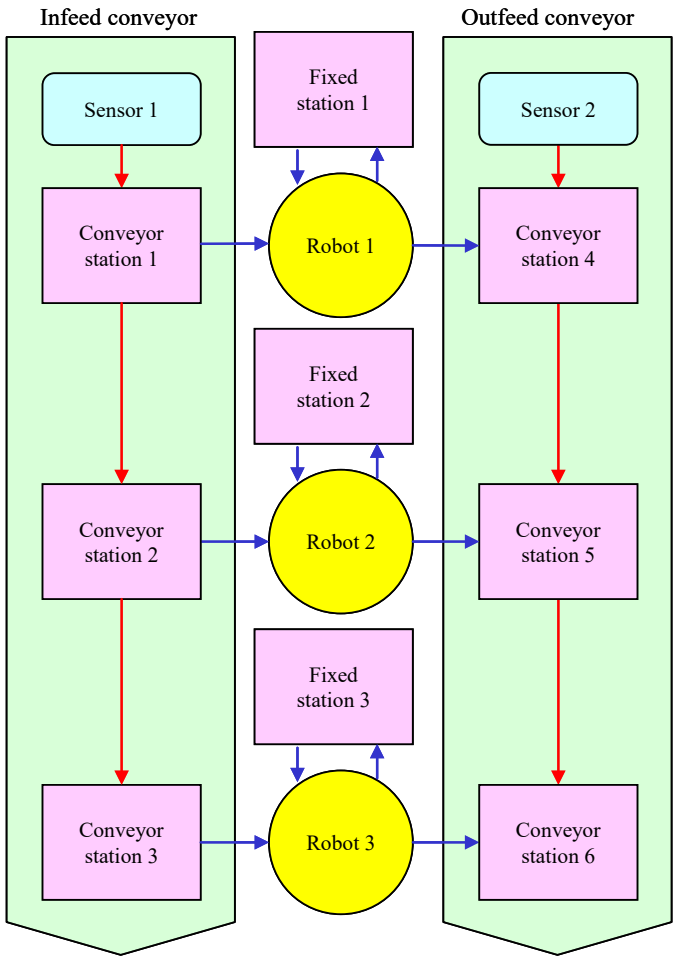


Fig. 2.4 (c)

Sample configuration 4

This sample is basically the same as sample configuration 1. The flow direction is different between the infeed and outfeed conveyors. This configuration is typically called “counterflow”.

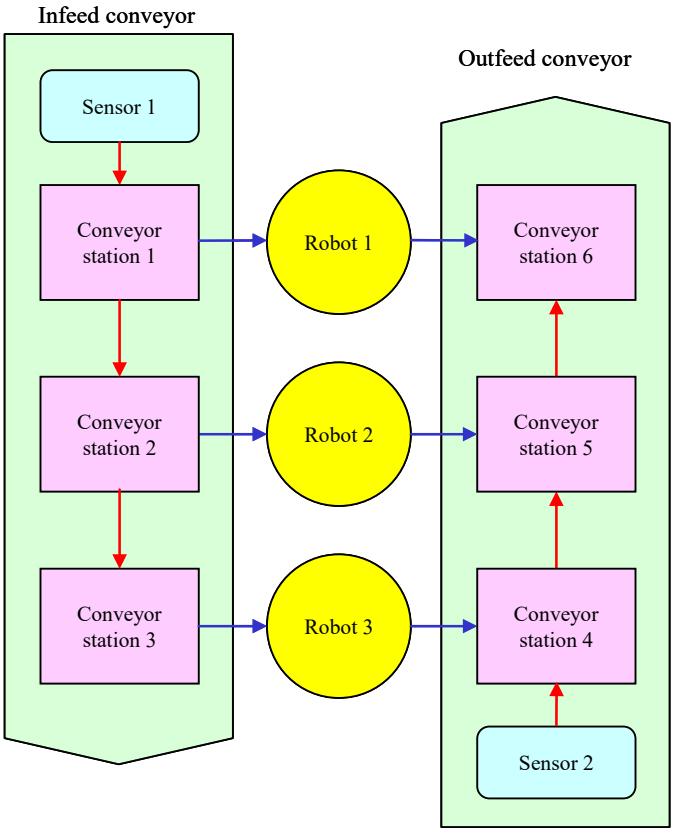


Fig. 2.4 (d)

Sample configuration 5

This sample is also basically the same as sample configuration 1. In this sample, conveyor 1 is wide and cannot be covered with one camera. In this case, install two cameras on the infeed conveyor, one covers the right half and the other covers the left half, and configure the system so that two sensors input the information of the flowing parts to the same conveyor station.

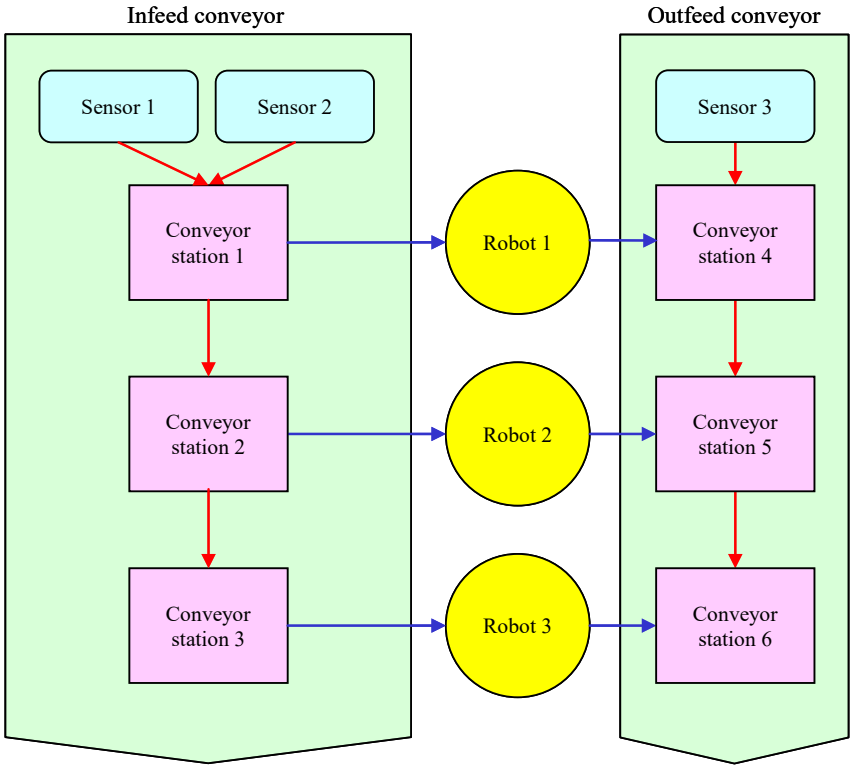


Fig. 2.4 (e)

Sample configuration 6

This system has a little more complicated configuration, in which two infeed and outfeed conveyors are installed and arranged crossways.

This system is seemingly complicated, but when you focus on operation of each robot, you can see that robot operation itself is almost the same as that in sample configuration 1. For example, robot 1 picks up parts from conveyor station 1 and places them in conveyor station 5.

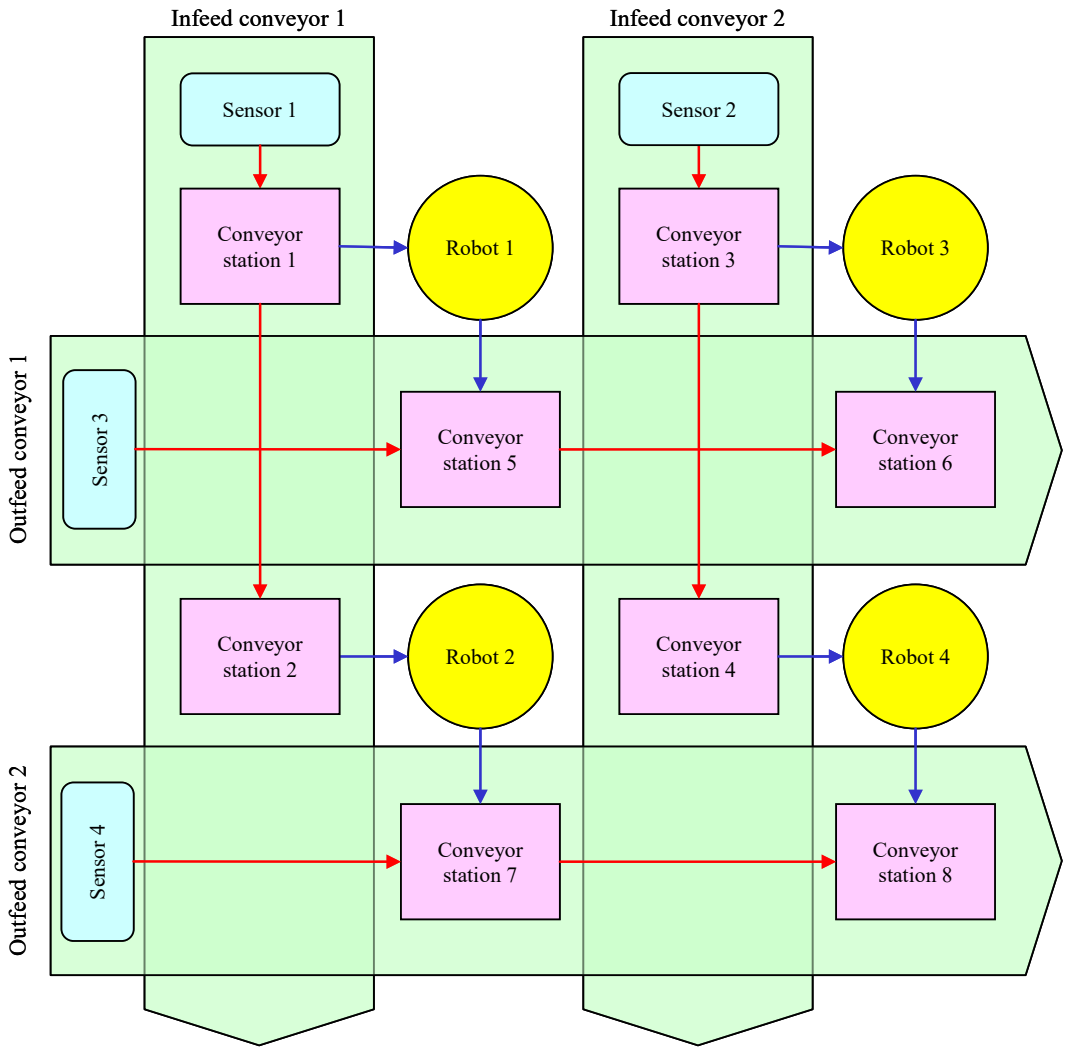


Fig. 2.4 (f)

Sample configuration 7

In this system, the infeed and outfeed conveyors are physically the same conveyor.
This configuration applies to the case where parts flowing disorderly on a conveyor are rearranged on the same conveyor. Making sensor 1 detect parts on the conveyor using a camera and sensor 2 detect markings made at equal spaces on the conveyor using a sensor such as a proximity sensor can arrange disorderly flowing parts on the conveyor at equal spaces.
Although only one conveyor is used, it is necessary to create two conveyor objects on the infeed and outfeed sides.

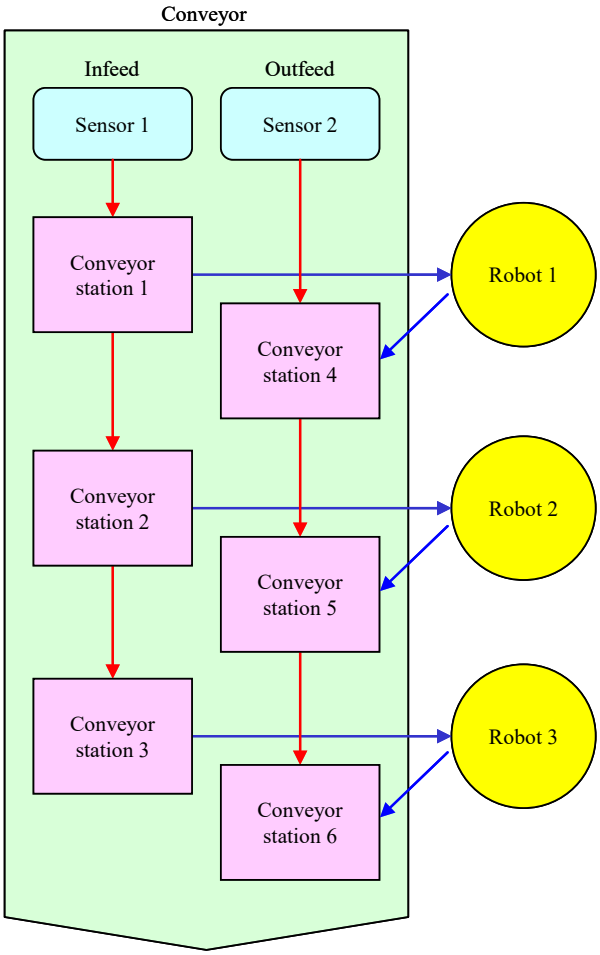


Fig. 2.4 (g)

Sample configuration 8

Two infeed conveyors are installed. Each robot picks up a part from either infeed conveyor and places it on the outfeed conveyor. This configuration applies to the case where there are two types of parts, parts of type A flow on infeed conveyor 1, parts of type B flow on infeed conveyor 2, and parts of types A and B are mixed in a box flowing on the outfeed conveyor.

To configure this system, the robot program is complex. An example of the actual program is introduced in Chapter 8, "EXAMPLES OF SETUP IN ACCORDANCE WITH USE".

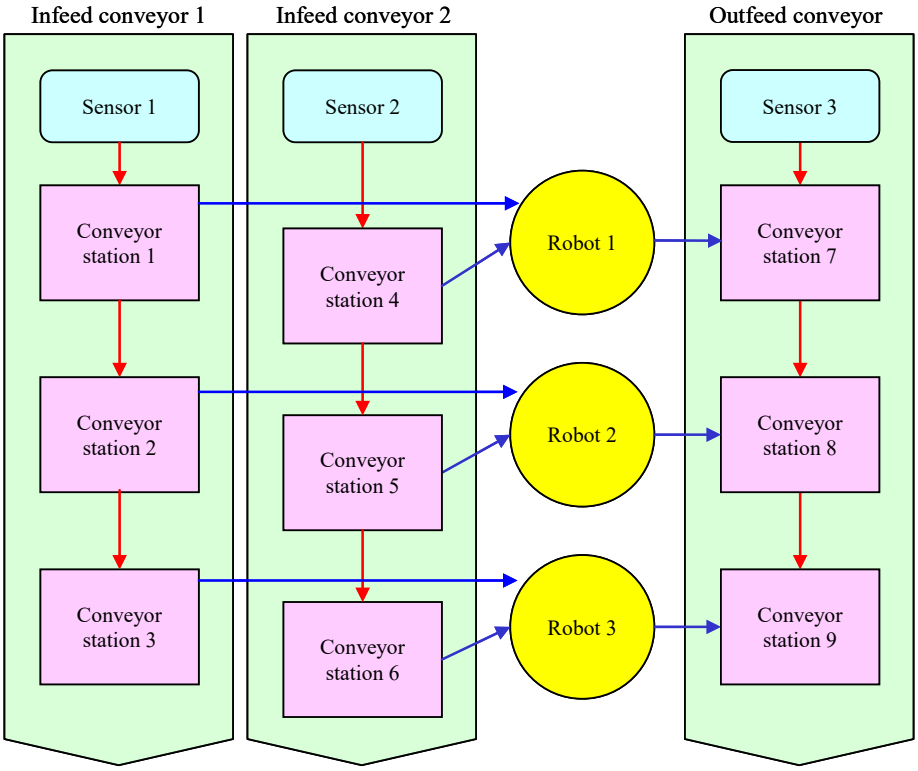


Fig. 2.4 (h)

2.5 STUDY FOR APPLICATION

This section provides information to design your "picking" application such as the layout, the robot and camera selection, the method for sharing the pulse count between robots, and so on.

System layout

Design a layout of robots, conveyors, fixed stations, trays, and other devices to minimize the distance the robots move. As the distance from the part pickup location (conveyor) to the part placement location (another conveyor or fixed station) decreases, the time required for one pick-and-place motion becomes shorter and you will get higher rates.

Robot

Determine the type of robot, the number of robots, the type of gripper, and so forth by considering the required transport quantity, part weight, and so forth.

The maximum robot speed depends on the type of robot. The distance moved by a robot and the gripper open/close time need to be considered. A higher-speed robot can make more pick-and-place motions per minute over a specified distance, handling a larger transport quantity. In general, however, such a robot can bear less weight.

When a gripper that can hold multiple parts is used, the number of pick-and-place motions for parts is reduced, so that one robot can handle a larger transport quantity. On the other hand, the gripper becomes heavier. So, be careful to ensure that the weight of the gripper plus parts does not exceed the transportable weight of the robot. Generally, the gripper becomes larger in this case, so that interference with peripheral equipment and robots can occur.

> ⚠ **CAUTION**
> To use a 4-, 5-axis robot, install the robot and conveyor such that the conveyor surface and XY plane of the world frame are parallel to each other. For a 4-, 5-axis robot, the tracking frame will be parallel to the XY plane of the world frame automatically when the tracking frame is set according to the standard visual tracking setting method described in this manual. If the conveyor surface and XY plane of the robot world frame are not parallel to each other, the tracking accuracy will be reduced.

Part detection method

Study the sensor to be used to detect parts flowing on each conveyor. There are two main methods.

Detection using a photo eye

A photo eye is installed in the upstream area of the conveyor to detect a part by checking whether it passes in front of the photo eye. With a photo eye, only the position in the conveyor moving direction can be recognized. The orientation and longitudinal conveyor position of flowing parts need to be constant.

Detection using a camera

A camera is installed in the upstream area of the conveyor, an image on the conveyor is snapped, and parts in the image are detected by *i*RVision. A part placed at any position in any orientation on the conveyor can be picked up correctly because the position and orientation are recognized with the vision system.

When parts are detected using a camera, the following two methods are available: With one method, as the conveyor moves over a certain distance, the system snaps an image using the camera and continues monitoring the conveyor. With the other method, the system snaps an image using the camera to detect a part only when the part passes in front of a photo eye installed with the camera.

> ⚠ **CAUTION**
> To use multiple cameras with multiple robots, do not connect two or more cameras to one robot controller. This is to distribute the image processing load to the robot controllers.

Distribution of the pulse count

To use multiple robots, study a method for distributing the pulse count of the Pulsecoder to each robot. The following two methods are available.

Distribution using a pulse multiplexer

The pulse count is input directly to each robot controller via a pulse multiplexer. The pulse count recognized by each robot controller becomes equal, and accurate tracking is possible.
However, more devices are required for system running. A line tracking interface board is necessary in all robot controllers. Also, the cable for connecting each robot controller and pulse multiplexer and the cable for connecting the pulse multiplexer and Pulsecoder are necessary.

Distribution using the Ethernet Encoder

The pulse count is input to one robot controller that is connected to the Pulsecoder. The input pulse

count is distributed to the other robot controllers using the Ethernet robot ring. Compared to distribution using a pulse multiplexer, you can reduce the cost of the hardware since this method does not require the pulse multiplexer, Pulsecoder to multiplexer cable, cables for connecting the pulse multiplexer and robot controllers, and line tracking interface board on each slave robot controller. However, when each robot controller recognizes the pulse count, some variation may occur due to the effect of communication. Compared to distribution using a pulse multiplexer, this variation may be between −2.0 ms and +3.0 ms in terms of time. For example, when the conveyor speed is 400 mm/s, the variation of the handling position will be between −0.8 mm and +1.2 mm.

⚠ CAUTION

1 When using the Ethernet Encoder, connect the Pulsecoder to the master controller of the robot ring to reduce the communication traffic.
2 When using the Ethernet Encoder, connect a camera or sensor to the robot controller connected to the Pulsecoder to obtain an accurate pulse count at the moment that parts are detected. Combined with CAUTION 1, connect the Pulsecoder and camera or sensor to the master controller of the robot ring.
3 For a system in which two or more cameras are used, the use of a pulse multiplexer is recommended. As described in the previous caution, if one camera is connected to each robot controller, you cannot avoid connecting a camera to each slave controller when the Ethernet Encoder is used. As a result, the variation of the handling position may be greater than mentioned above because the slave controller cannot obtain the accurate pulse count at the moment that parts are detected.

Conveyor speed

A transport quantity is determined by the conveyor speed and the density of parts on the conveyor. When the conveyor speed is 200 mm/s and parts are located at average intervals of 100 mm, the transport quantity is 120 parts/min. When the conveyor speed is 100 mm/s and parts are located at intervals of 50 mm, the transport quantity is unchanged, namely, 120 parts/min. In this case, as the conveyor moves slower, the robot tracking precision and vision detection precision are improved. This means that a slower conveyor offers better overall system performance.

Camera's field of view

To use a camera to detect a part, consider its field of view. The field of view is determined by the types of lens and camera, and the height of the camera from the part. The expression is shown below.

$Y = y \times ((L - f) \div f)$

Y: Field of view
y: CCD size
L: Height of the camera from the part
f: Focal length

The following figure shows the positional relationship.

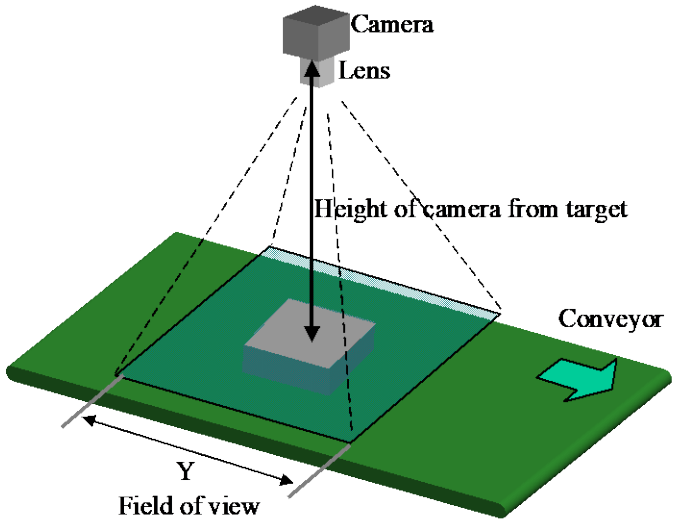


Fig. 2.5(a) Positional relationship that determines a camera's field of view

The CCD size is the size of the CCD (charge-coupled device) of the camera. It can be determined using the following expression:

CCD size = cell size x image size

The following table lists the CCD size of each camera FANUC can provide as standard.

Table 2.5 Types of camera and resolution

Camera	Cell size	Image size	CCD size
Digital camera (grayscale) SC130EF2	5.300 μm	1280 x 1024	6.784 mm x 5.427 mm
Digital camera (color) SC130EF2C	10.600 μm	640 x 512	6.784 mm x 5.427 mm

For example, when the focal length is 8 mm, and the height of the camera from the part is about 1000 mm, the field of view is determined as follows:

Horizontal direction: 6.784 (mm) x ((1000 (mm) – 8 (mm)) ÷ 8 (mm)) = 841.2 (mm)
Vertical direction: 5.427 (mm) x ((1000 (mm) – 8 (mm)) ÷ 8 (mm)) = 672.9 (mm)

NOTE
It is desirable that the field of view is as small as possible as long as parts can be detected. The smaller the field of view, the greater the position detection resolution of the vision system, because the real-space area per pixel in which the image is projected is smaller.

Exposure time

To use a camera to detect a part, study the appropriate camera exposure time. If the part moves during exposure, the snapped image is blurred and the detection accuracy is reduced. Determine the exposure time to reduce the image displacement as much as possible. The distance a part moves depends on the conveyor speed, so the exposure time depends on the conveyor speed. As the conveyor speed is higher, the exposure time has to be shorter. On the other hand, as the conveyor speed is lower, the exposure time can be longer. However, as the exposure time is longer, the time required for detection by the vision system also becomes longer. So, even when the conveyor moves slowly, a relatively short exposure time should be used.

Use the following expression to calculate the exposure time suitable for the conveyor speed:

$E \leq \Delta \div V$

E: Exposure time (ms)
Δ: Image displacement (mm)
V: Conveyor speed (mm/s)

For example, when the conveyor speed is 100 mm/s, the expected exposure time to make the image displacement when a part is detected by the vision system 0.1 mm or less is computed by the following expression:
0.1 (mm) ÷ 100 (mm/s) = 1 (ms)
Accordingly, set an exposure time of 1 ms or less.

As the exposure time is shorter, stronger lighting is needed. Prepare lighting to enable a sufficiently bright image to be snapped with the exposure time calculated above.

Vision system trigger interval

To detect parts on a conveyor using a camera, pay attention to the intervals at which to execute the vision process. The vision system needs to perform image processing at least one time as the conveyor moves over a certain distance so that parts on the conveyor are not missed. Accordingly, the vision system is highly affected by the conveyor speed. That is, as the conveyor moves faster, the image processing needs to be performed more frequently. On the contrary, as the conveyor speed is lower, the image processing needs to be performed less frequently.

The vision system trigger interval can roughly be calculated using the following expression:

Trigger interval (s) = 0.5 x camera's field of view (mm) ÷ conveyor speed (mm/s)

As the camera's field of view is enlarged, the vision system trigger interval is increased, but the detection precision is decreased.

Handling precision

Visual tracking handling precision is affected by the tracking frame setting error, TCP setting error, robot position teaching error, robot tracking delay, and other factors in addition to the error caused due to the vision detection precision. Furthermore, the handling precision may be reduced by pulsation occurring in the movement of the conveyor or by a large part such as a press formed product swinging on the conveyor.

By taking the following measures, handling precision can be improved:

- Set the TCP for touch-up accurately (UTOOL should be precise).
- Decrease the conveyor speed to detect many parts at a time.
- Prepare sufficient bright lighting and shorten the exposure time.
- Teach the track frames and reference positions accurately.

Load balancing

When multiple robots share the work of handling parts on one conveyor, you can specify the quantity and type of parts each robot is to handle. This specification of the "type of part each robot is to handle, and the quantity of parts the robot is to handle" is called load balancing.

The load balancing function can be used to specify the ratio of the quantity of parts each robot is to handle. Operation for changing the robot to which to pass the information of parts according to such specification is called "allocation" A typical allocation method is "equal distribution", which makes each robot handle the same quantity of parts. For equal distribution, for example, when two robots are used, specify 1:1 for load balancing. If the robots have different processing capabilities, you may also specify 2:1.

When the load balancing function is disabled, each robot attempts to handle as many parts as it can. As a result, the maximum transport capability of the system is exerted. If a small number of parts flow, however, the robot in the upstream area handles more parts and only few parts flow to the robot in the downstream area.

When parts are detected using a camera, you can also specify which robot to handle which type of part in addition to the quantity of parts each robot to handle. For example, specification can be made so that robot 1 handles round parts and robot 2 handles triangle parts.

For allocation according to the type, use model IDs of *i*RVision. Model IDs are typically used to determine types of parts. Using the *i*RVision conditional execution tool together with model IDs allows various types of allocation in addition to the determination of types of parts.

For example,

- Allocation based on the part detection position (For example, robot 1 handles the parts on the right side of the conveyor and robot 2 handles the parts on the left side of the conveyor.)
- Allocation based on the part orientation (For example, robot 1 handles the parts of which orientation is between 0 and 180 degrees and robot 2 handles the parts of which orientation is between −180 and 0 degrees.)

You can also combine these allocation methods. For specific allocation methods, see Chapter 8, "EXAMPLES OF SETUP IN ACCORDANCE WITH USE"

Robot controller which executes a "sensor"

In some systems introduced in Section 2.4, "SAMPLE SYSTEM CONFIGURATIONS", multiple sensors are used. In the most general system, one sensor for the infeed conveyor and one sensor for the outfeed conveyor, a total of two sensors, are generally used.

As means for detecting parts on a conveyor with a sensor, the following two main methods are introduced: Detecting parts using the vision system with a camera and detecting them using a photo eye. No special consideration is required for a "sensor" using a photo eye because the amount of processing for the sensor is small. A "sensor" using the vision system with a camera performs image processing, so a considerable processing load rests on the robot controller.
For this reason, when two or more "sensors" use the vision system, configure the system so that the same robot controller does not process multiple "sensors". More specifically, configure the system so that two or more cameras are not connected to one robot controller.

HDI Sensors

Use the HDI option when you require detection from photo-eye sensors in 1 ms or less.

2.6 RESTRICTIONS

As of 7DF5/14, the following restrictions are imposed on queue-managed tracking and visual tracking.

- Does not support rail tracking.
- Does not support hot start mode. Disable the hot start.
- Does not support analog cameras.
- Does not support robot-mounted camera.
- Cannot use “Distance before” function. (Can use “Time before” function)
- There are restrictions concerning the sensors that can be operated simultaneously. For details, refer to Subsection 6.5.7, “Restrictions on Sensors that can be Operated Simultaneously”.
- When using a recipe, the software versions of all the controllers connected via the ROS Interface Packets Over Ethernet (RIPE) need to match.
- Line tracking function cannot be used with the following functions.
  - Palletizing
  - Coordinated Motion
  - Space Check
  - Remote TCP
  - Singularity Avoidance (Unavailable for tracking motion. Only available for normal motion)
  - Finishing Function Package
  - Servo Gun Change function
  - Robot Link Function
  - Basic/Intelligent Interference Check
  - Arc Sensor (TAST)
  - AVC
  - RPM
  - Touch Sensor
  - Restart Position check function

2.7 RECOMMENDATIONS

The following settings are recommended in *i*RPickTool:

- Disable restart position check (disabled by default).
- Disable brake control (disabled by default).
- Disable automatic backup.

3 *i*RPICKTOOL BASIC (BASIC FUNCTIONS) STARTUP PROCEDURES

This chapter explains the startup procedures for *i*RPickTool Basic (Basic Functions).
By performing the procedures as explained in this chapter, you will complete the setup of systems that use four-axis Genkotsu robots as shown in the figure below.

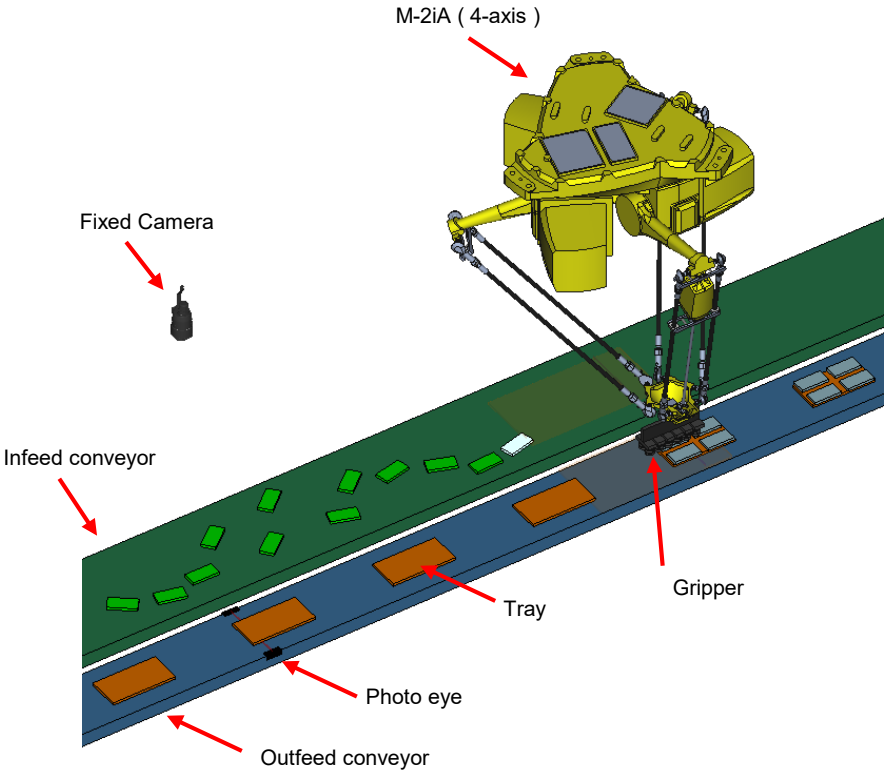


Fig. 3(a) System configuration

The procedure to start up the conveyor varies depending on the combination of the [Trigger Condition] for detection and the [Trigger Action] that is executed when the condition for the trigger is met. Please select the combination in accordance with the use.

This chapter explains the following combinations as system configuration examples.

Infeed: [Distance] + [Find part by vision]
Outfeed: [DI] / [RI] / [HDI] + [Put part in queue] (Use tray)

For information on the startup procedures when conveyors and fixed stations in other combinations are used, and details on each tree item, refer to Chapter 6, “BASIC FUNCTIONS REFERENCE”.

3.1 PREPARATION BEFORE SETTING UP THE APPLICATION

This section explains the preparation for the startup of tracking systems that use *i*RPickTool, such as the connection of hardware and creation of programs.

3.1.1 Connection and Setup of Pulsecoders

Install Pulsecoders and connect them to the robot controller.
After connecting, set parameters on the teach pendant of the robot controller.

NOTE
In this manual, a device that detects the travel distance of a conveyor is called a 'Pulsecoder.'
In the setup screens of the teach pendant and *i*RPickTool, the term 'encoder' is used as a synonym for 'Pulsecoder.'

αA1000S Pulsecoders and incremental Pulsecoders can be used.

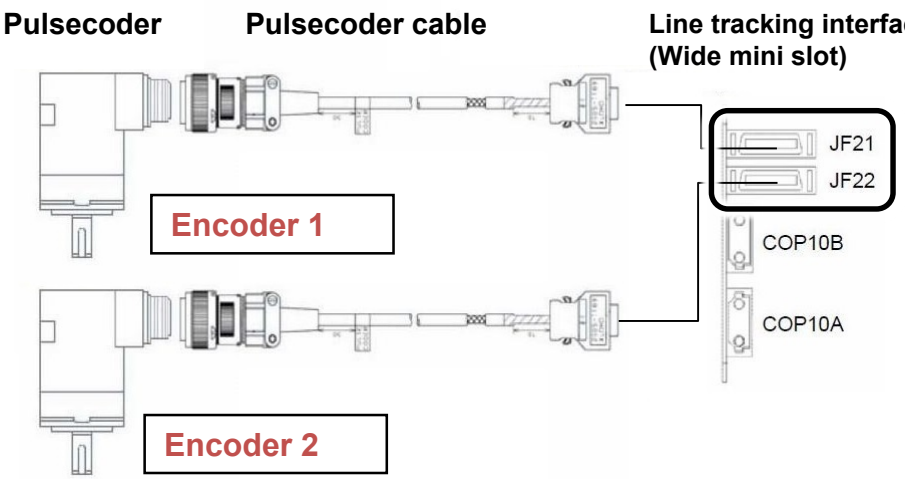
NOTE
If you use one Pulsecoder or three Pulsecoders, you can use them by connecting an αA1000S Pulsecoder to the main board. Refer to Appendix E, "INCREMENTAL PULSECODER" for information on connection, and Subsection 5.2.3, "Setting Up Pulsecoders That Are Connected to the Main Board" for information on setup.

3.1.1.1 Connecting Pulsecoders

Connect Pulsecoders to the robot controller.

Connecting to interface board for line tracking

Use the following figure as a reference, and connect Pulsecoders so that the Pulsecoder that is installed at the infeed conveyor will be encoder No. 1 and the Pulsecoder that is installed at the outfeed conveyor will be encoder No. 2.



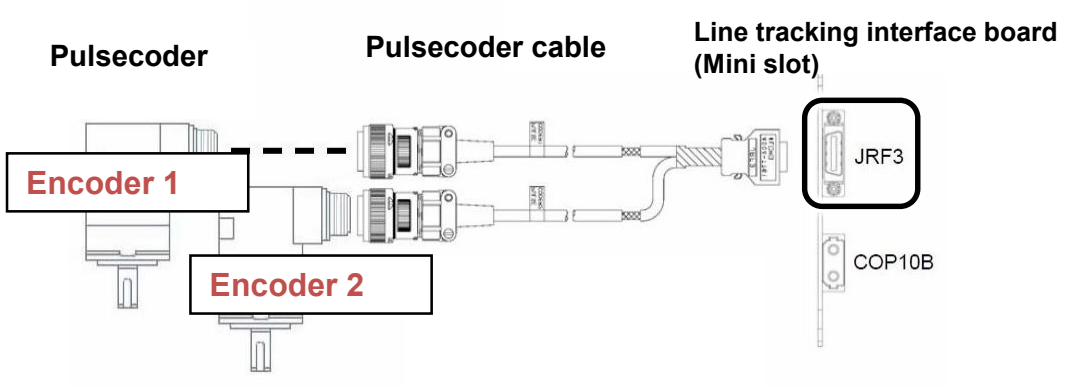


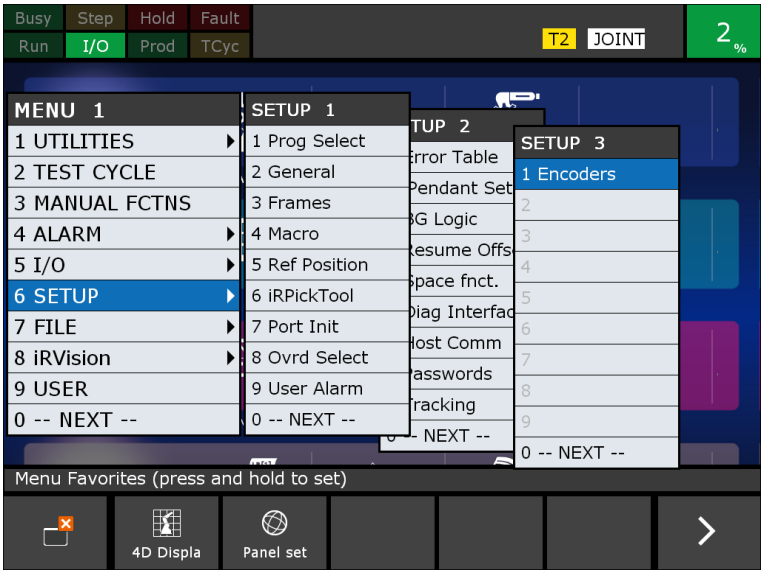
Fig. 3.1.1.1(a) Connection of interface board for line tracking and Pulsecoders

3.1.1.2 Setting up Pulsecoders

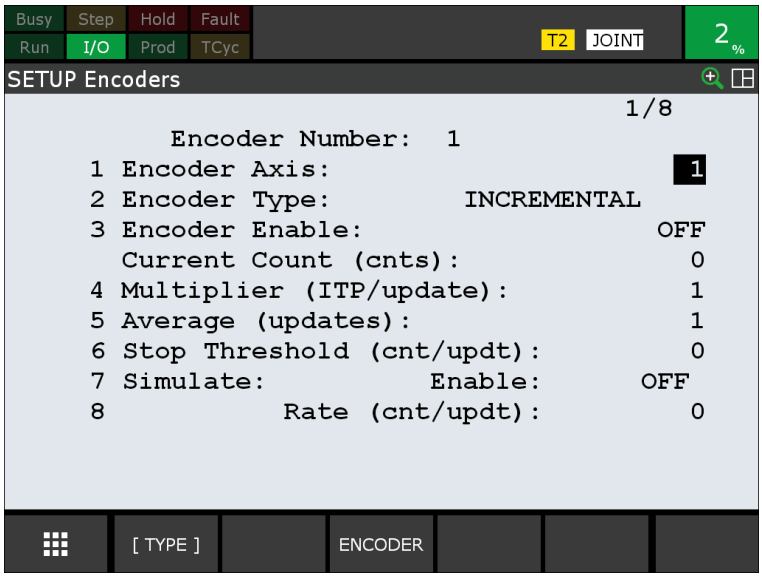
Set the parameters for the Pulsecoders. Set the following items.

- [Encoder Axis]
- [Encoder Type]
- [Average (updates)]

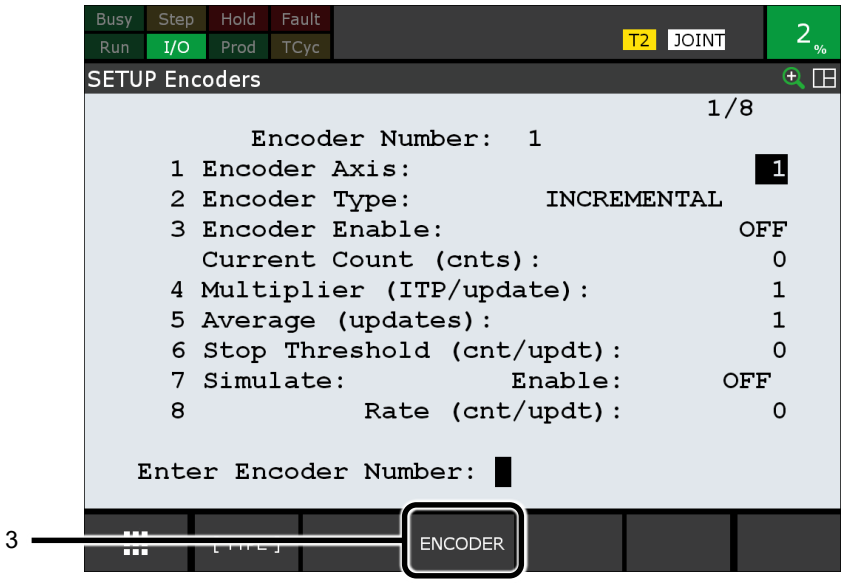
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SETUP] → [Encoders] from the menu.



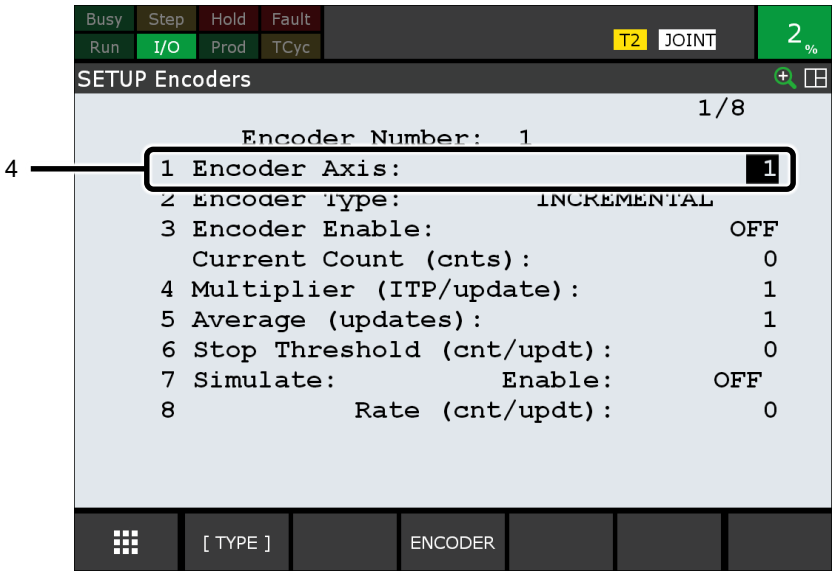
A screen like the following one will appear.
Number [9] and after in the setting items are the menu for option functions. They will not be set here.



3 First, set the parameters for the Pulsecoder that is installed at the infeed conveyor. Check that [Encoder Number] is 1. If it is a value other than 1, press F3 [ENCODER] and input 1.

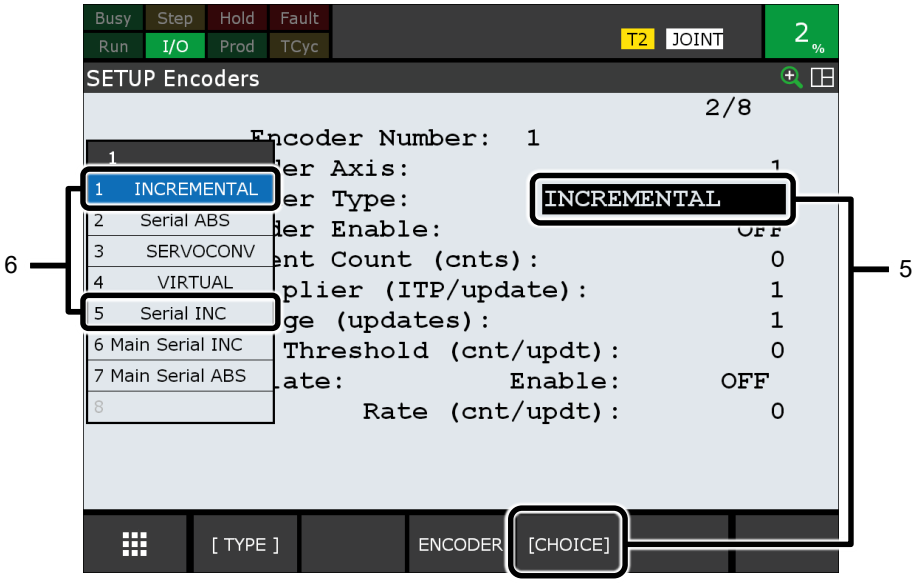


4. Move the cursor to [Encoder Axis] and input 1. However, if connecting to the main board, or for R-30*i*B Compact Plus / R-30*i*B Mini Plus, leave it 0 as the default value.



5 Move the cursor to [Encoder Type] and press F4 [CHOICE].
A menu will appear.

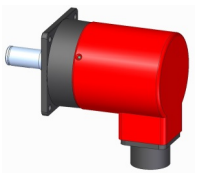
6 Select the encoder type that corresponds to the Pulsecoder that is to be connected.



Encoder type

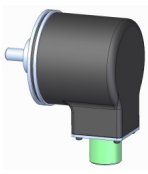
There are the following types for encoders.

- αA1000S Pulsecoder (red Pulsecoder): Serial INC
- Incremental Pulsecoder (black Pulsecoder): INCREMENTAL



αA 1000S (A860-0372-T001)
Encoder type: Serial INC

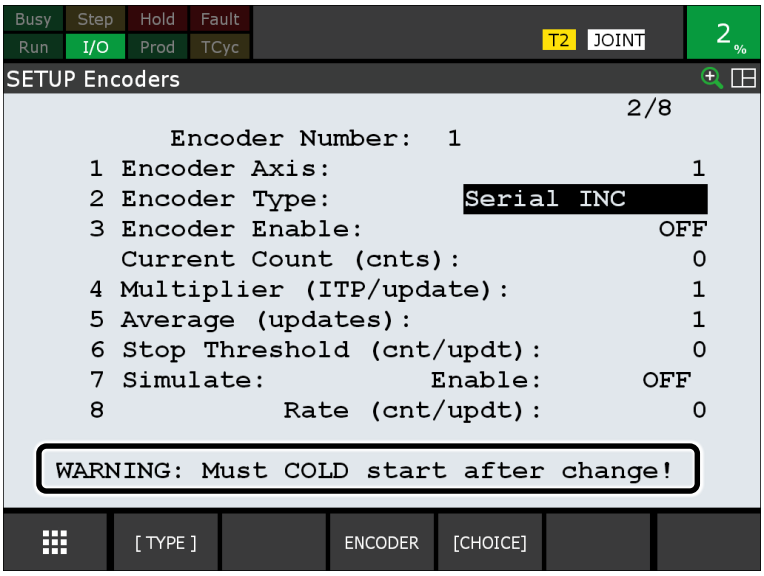
* If connecting to the main board, or for R-30*i*B Compact Plus / R-30*i*B Mini Plus,
Main Serial INC



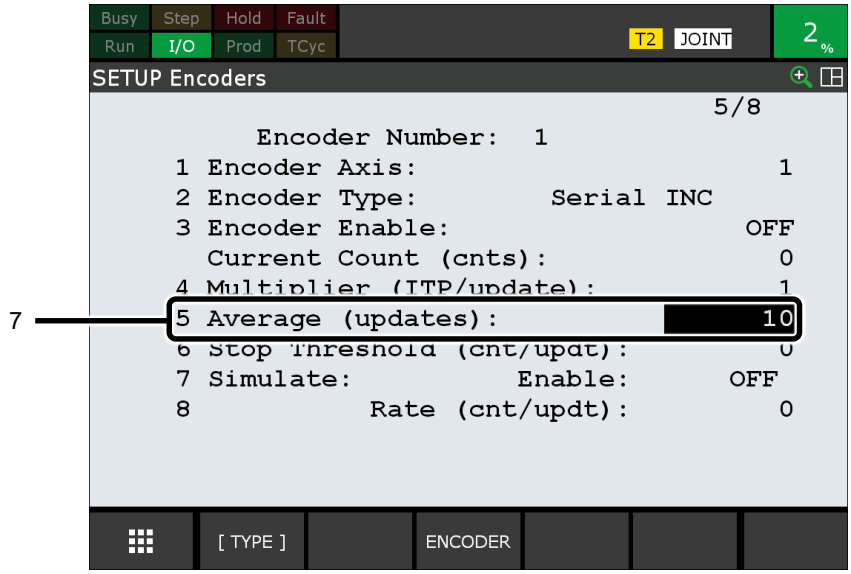
A860-0301-T001 to T004
Encoder type: INCREMENTAL

* Cannot be connected to the main board

If encoder axes or encoder types are changed, the following warning prompting a cold start will appear.

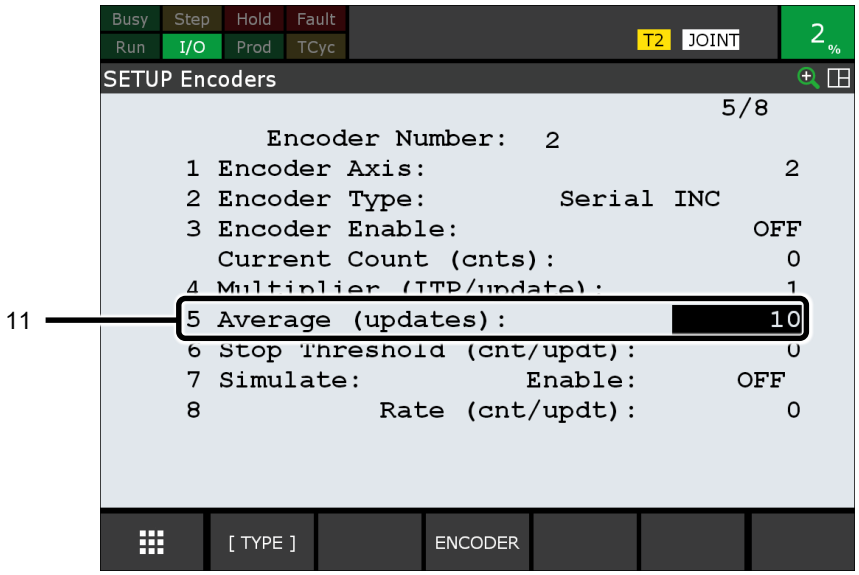


7 Move the cursor to [Average (updates)] and input the accumulated number of instantaneous speeds (width of moving average).
By setting the accumulated number of instantaneous speeds, the robot will stop smoothly and not suddenly, even if the conveyor suddenly stops while the robot is tracking.
Usually, enter '10.'



8 Next, set the parameters for the Pulsecoder that is installed at the outfeed conveyor. Press F3 [ENCODER], then input 2.
9 Move the cursor to [Encoder Axis] and input 2.
10 Move the cursor to [Encoder Type], press F4 [CHOICE], and select the encoder type that corresponds to the Pulsecoder that is to be connected.

11 Move the cursor to [Average (updates)] and input a value. This is the same as step 7. Usually, enter '10.'

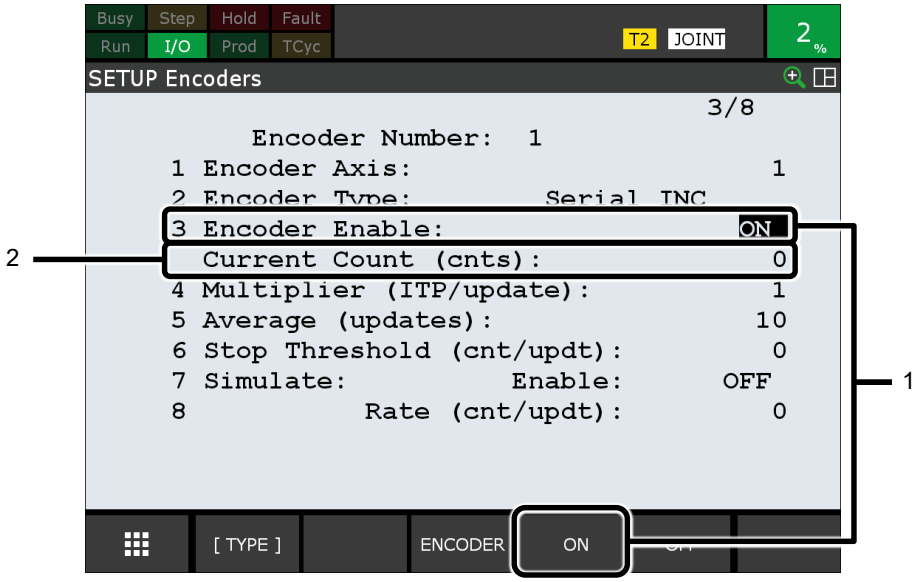


* When an HDI signal is used as an option, [9 HDI Port Num.] is displayed, so input the port number that corresponds to the HDI signal that is to be used.

12 Finally, cold start the robot controller.

3.1.1.3 Checking the Pulsecoder settings

1 Open the [Encoders] screen and move the cursor to [Encoder Enable], then press F4 [ON].
2 Move the conveyor and confirm that the pulse count is displayed at [Current Count (cnts)].



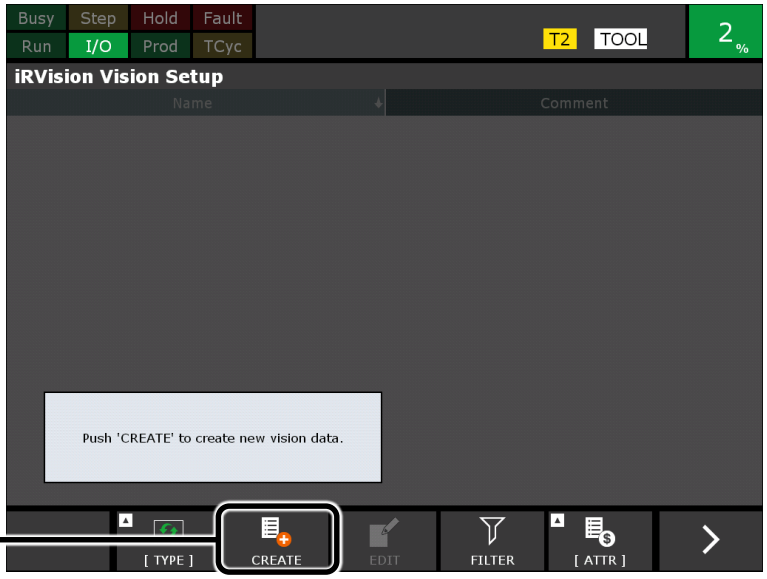
3 In the same way, for the Pulsecoder that is connected to the outfeed conveyor, select Encoder Number: 2, then confirm.

3.1.2 Checking the Connection of the Camera

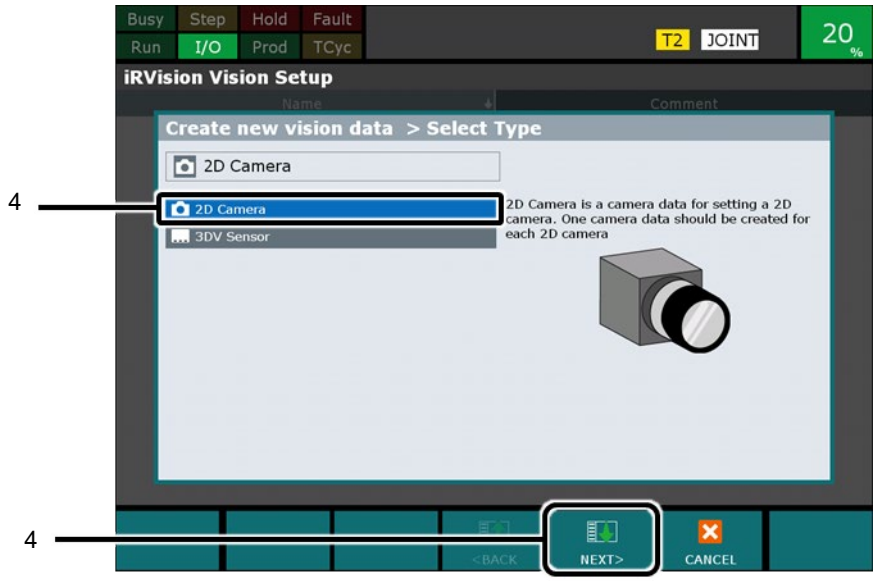
Check the connection of the camera that is installed above the infeed conveyor.

1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [*i*RVision] → [Vision Setup] from the menu.

The [*i*RVision Vision Setup] screen will appear.
3 Press F2 [CREATE].



4 Select [2D Camera] and press F4 [NEXT>].



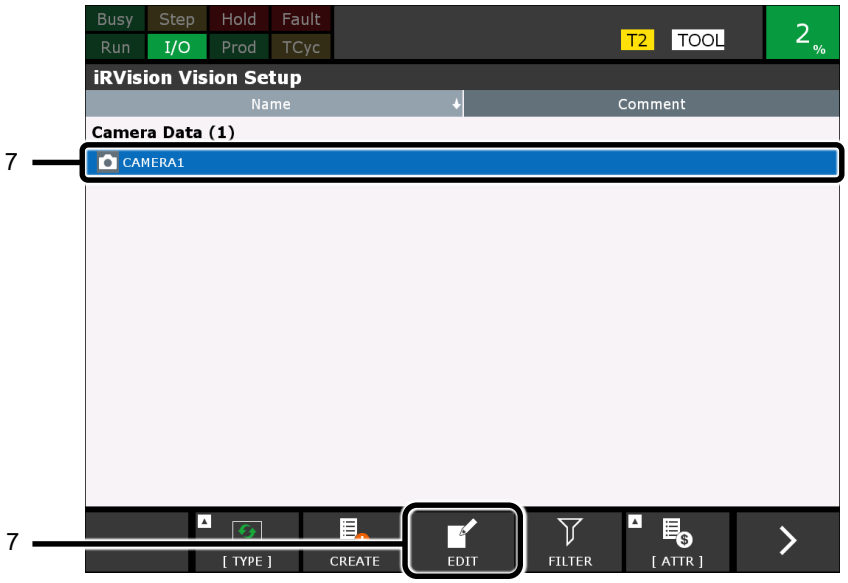
A screen like the following one will appear.

5 Input the name of the camera in [Name].
Input using half-width alphanumeric characters and half-width katakana. The name must contain no spaces, not contain the symbol other than underscore, and start with a letter.
Example: CAMERA1

6 Press F4 [OK].

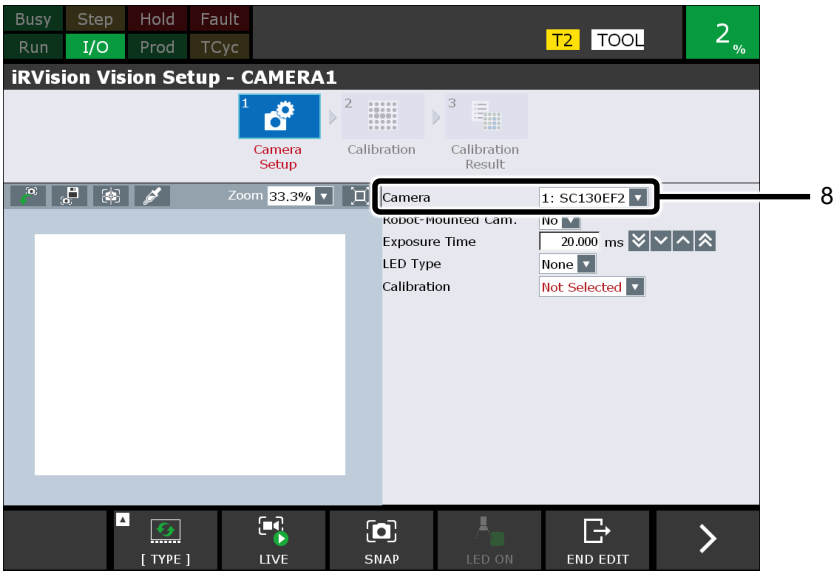
The created camera data will appear on the list.

7 Select the created camera data and press F3 [EDIT].



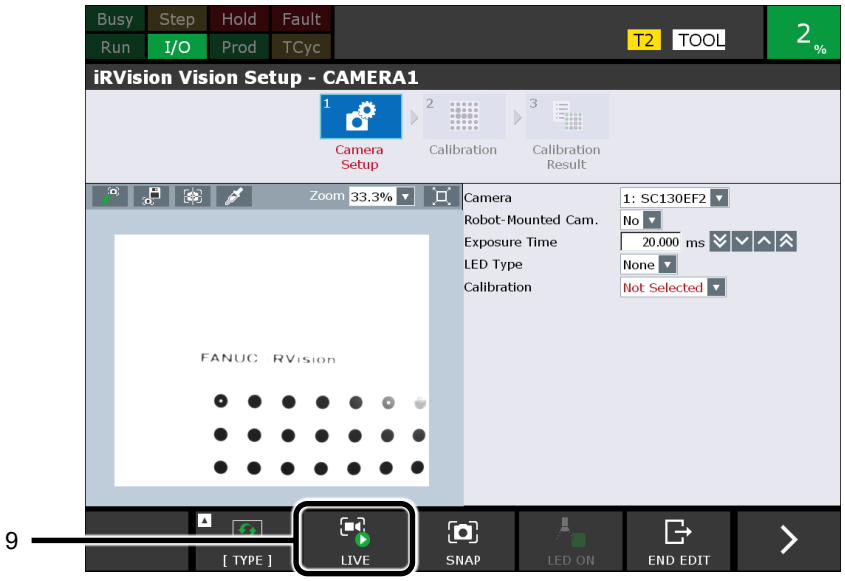
A screen like the following one will appear.

8 Select [Camera] from the menu.



9 To adjust the focus of the lens, place something that has characters displayed on it within the field of view of the camera, then press F2 [LIVE].
Confirm that the item that was placed has been captured within the frame on the left side of the screen.
* If the screen is dark, adjust the following.
- Open the diaphragm of the lens.
- Adjust the lighting.

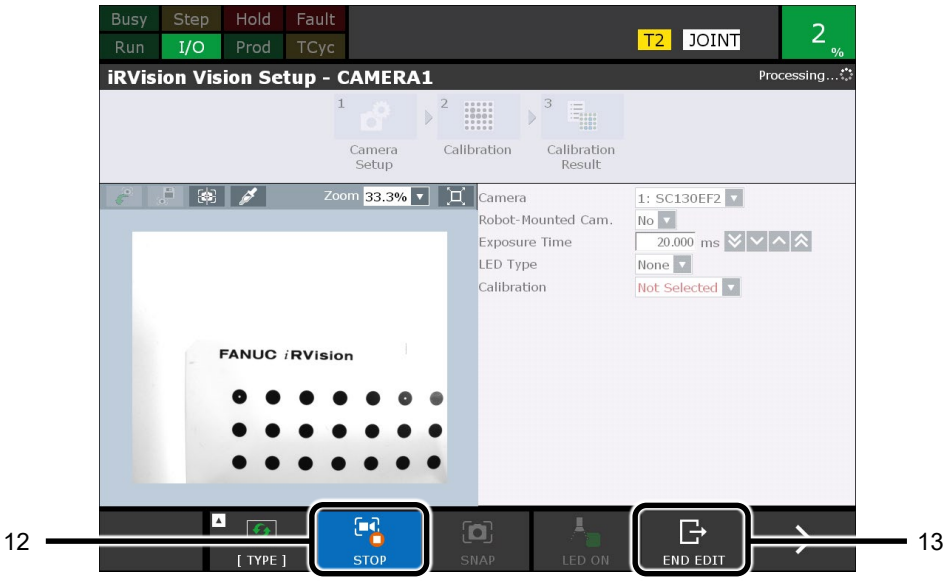
10 Check the field of view of the camera.
If the field of view is too wide or too narrow, adjust the position of the camera.



11 When the camera position has been set, adjust the focus of the lens.

Fig. 3.1.2(a) Adjusting the focus

12 Press F2 [STOP].
Live will stop.
13 Press F5 [END EDIT].
The edit screen will close.



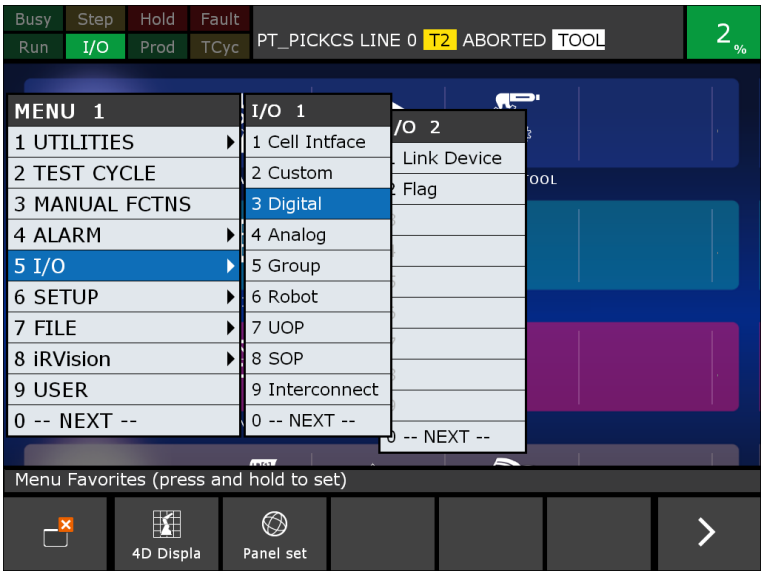
3.1.3 Checking the Input of the Trigger Sensor

Check the input of the trigger sensor.

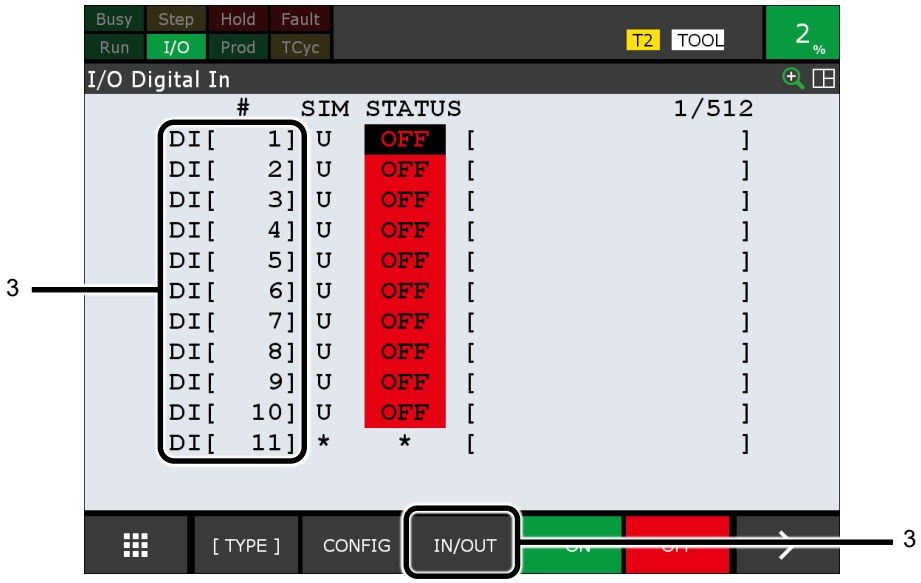
3.1.3.1 Using [DI] / [RI]

Check the input of the trigger sensor when DI / RI are used.

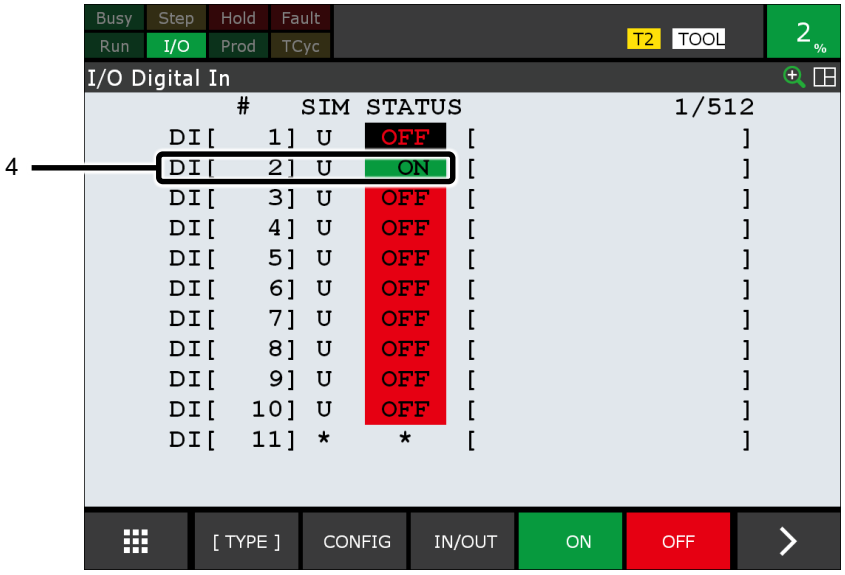
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 If using DI, select [I/O] → [Digital] from the menu. If using RI, select [I/O] → [Robot] from the menu.
The screen is the situation when DI is used.



3 Press F3 [IN/OUT] and confirm that DI (RI) appears.



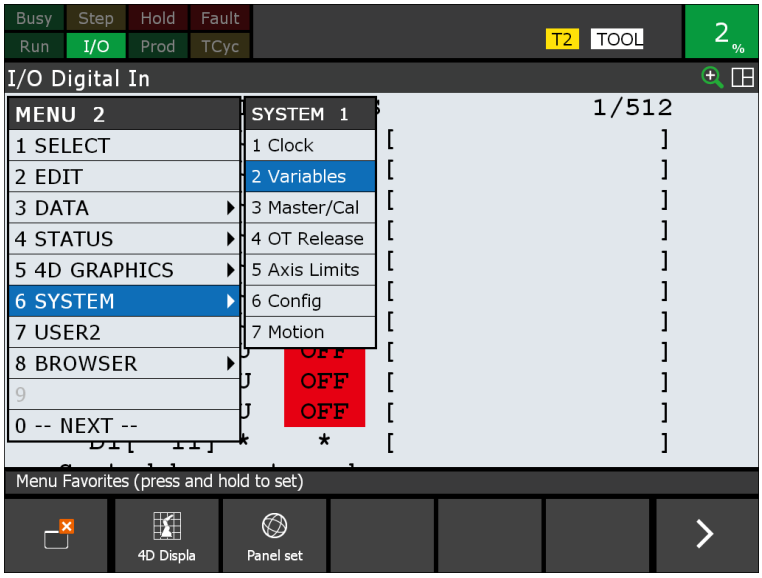
4 Confirm that DI (RI) changes to ON with input from the sensor.



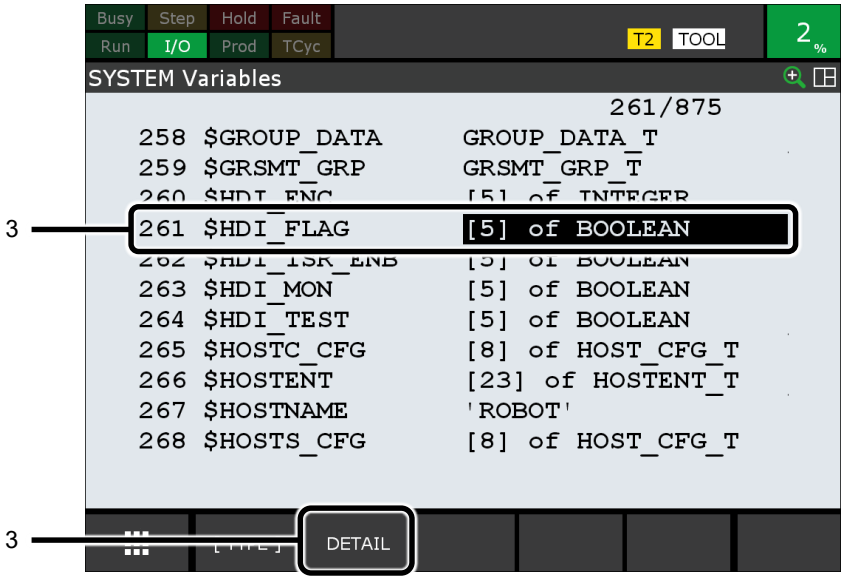
3.1.3.2 Using [HDI]

Check the input of the trigger sensor when HDI is used.

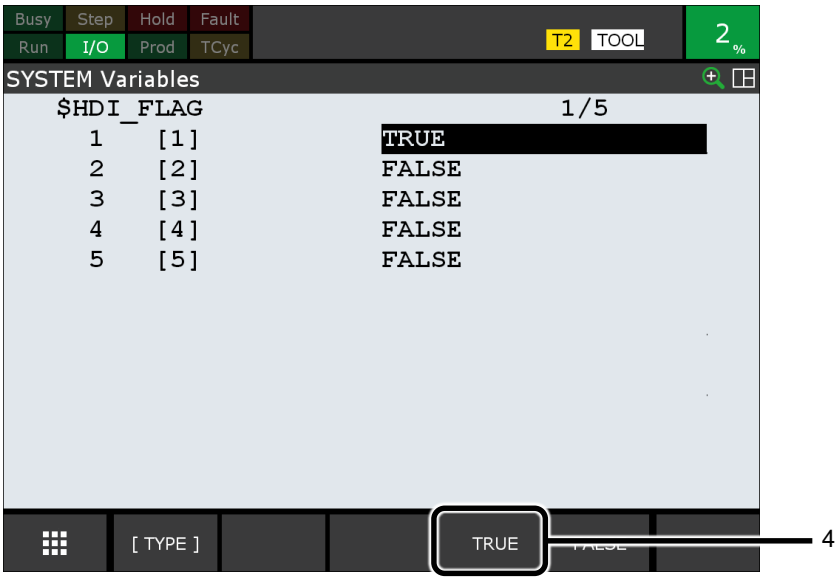
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [NEXT] → [SYSTEM] → [Variables].



3 Place the cursor over '$HDI_FLAG' and press F2 [DETAIL].

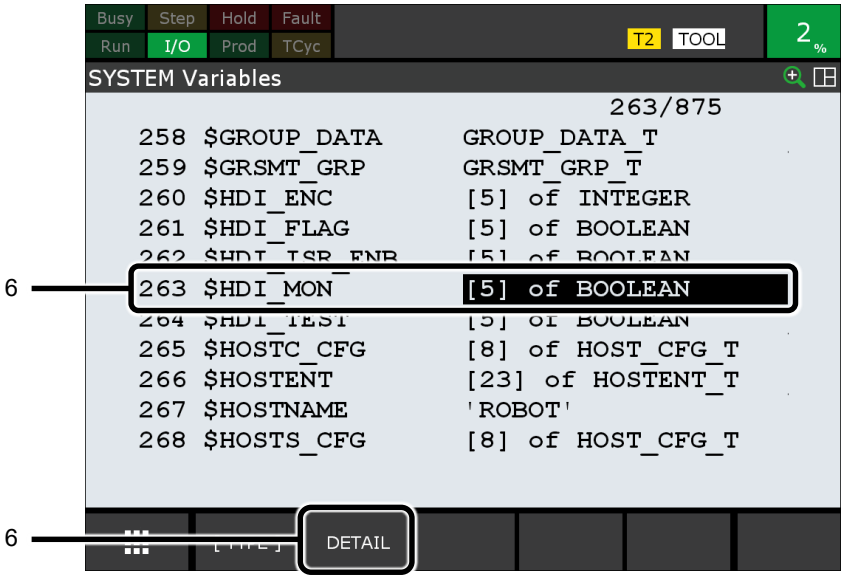


4. Move the cursor to the HDI port number to check and press F4 [TRUE].

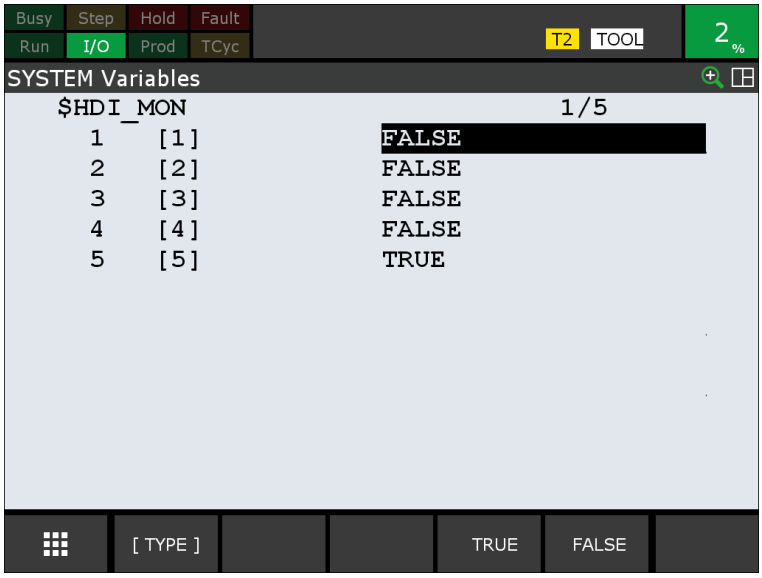


5 Press the [PREV] key on the teach pendant of the robot controller.
The [SYSTEM Variables] screen will appear.

6 Move the cursor to '$HDI_MON' and press F2 [DETAIL].

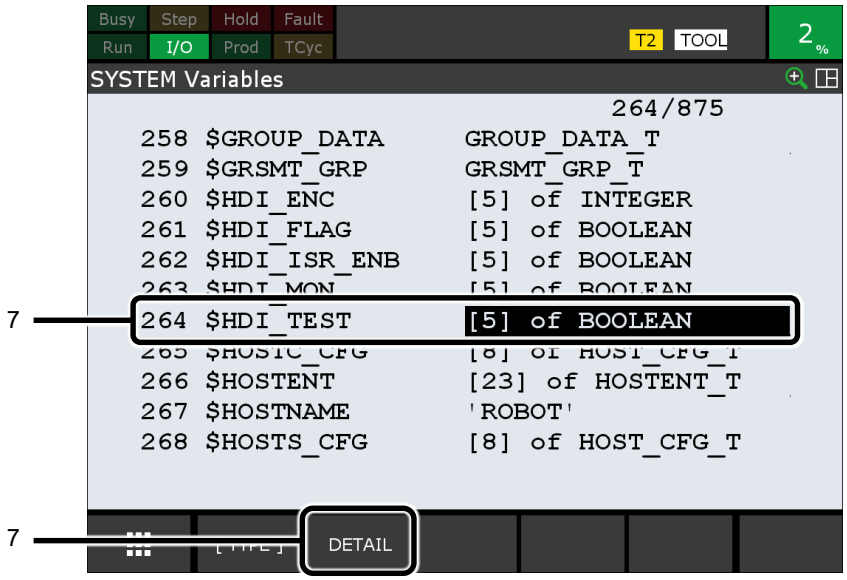


Check the variable that corresponds to the port number that is connected to the sensor.
Input ON : TRUE
Input OFF : FALSE

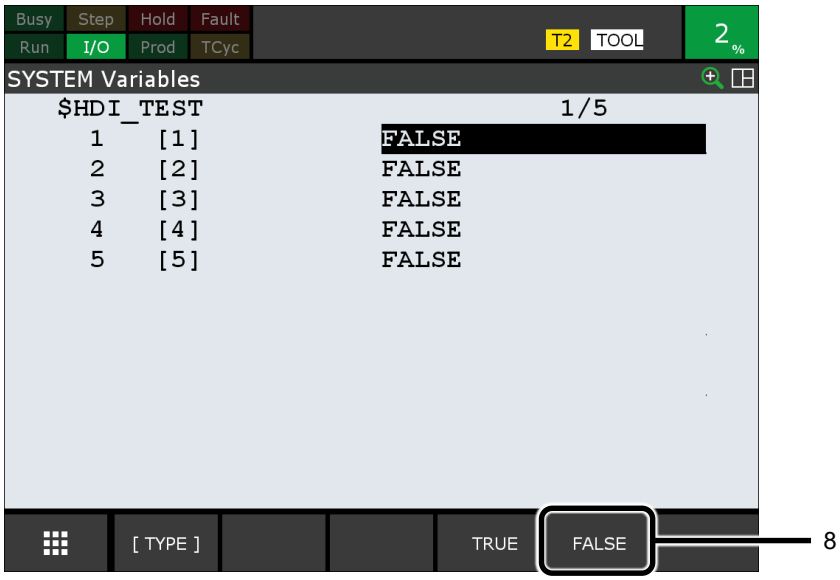


If the input time is too short and cannot be checked by means of step 7 check it using the following method.

7 Move the cursor to '$HDI_TEST' and press F2 [DETAIL].



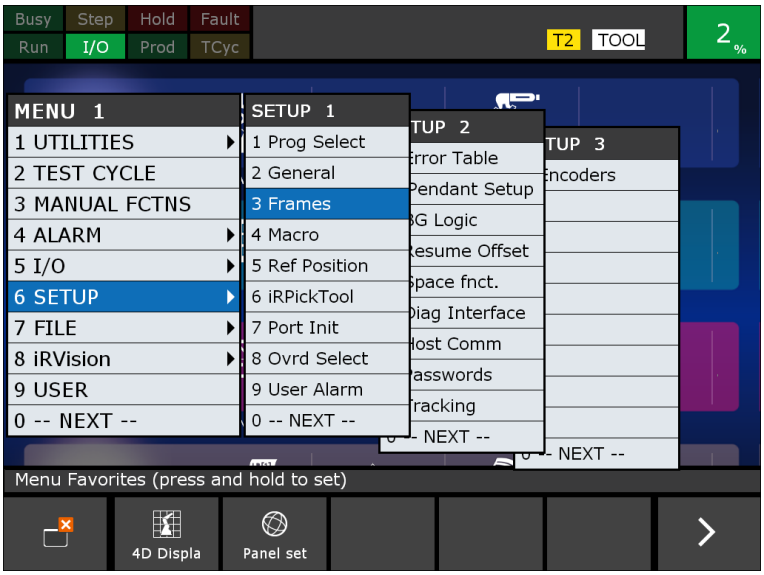
8 Move the cursor to the HDI port number to check and press F5 [FALSE].
Once the input turns ON, it changes to TRUE and retains the TRUE state even if the input turns OFF.



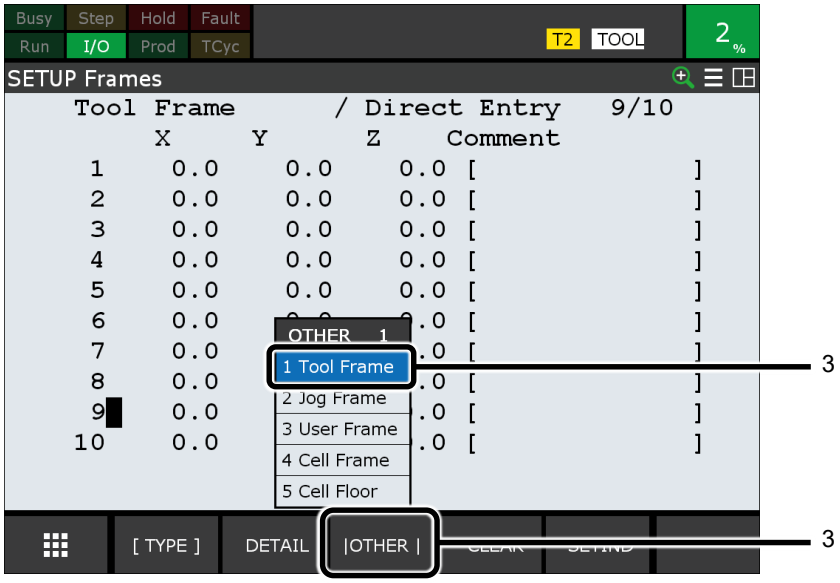
3.1.4 Setting up the Tool Frame of the Pointer Tool

Set up the tool frame of the pointer tool.

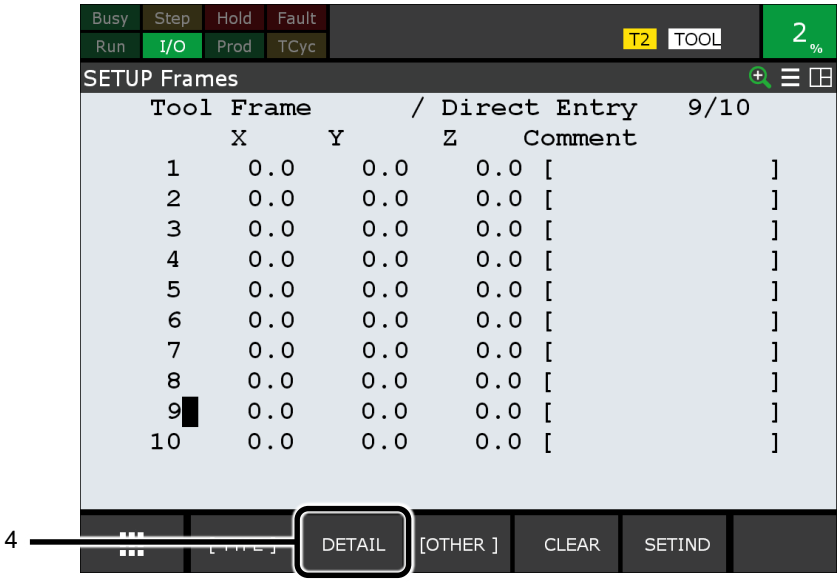
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SETUP] → [Frames] from the menu.



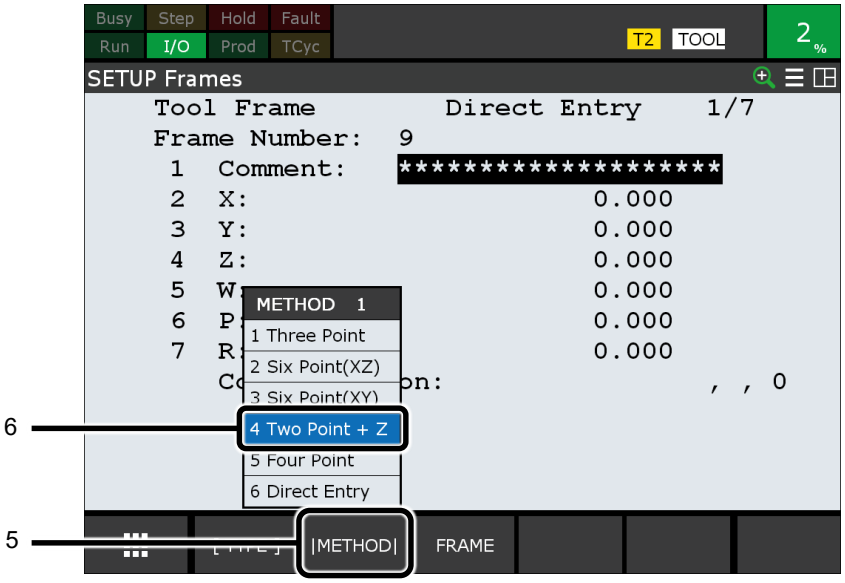
3 If frames other than the tool frames are displayed, press F3 [OTHER] and select [Tool Frame] from the menu.
A menu will appear.



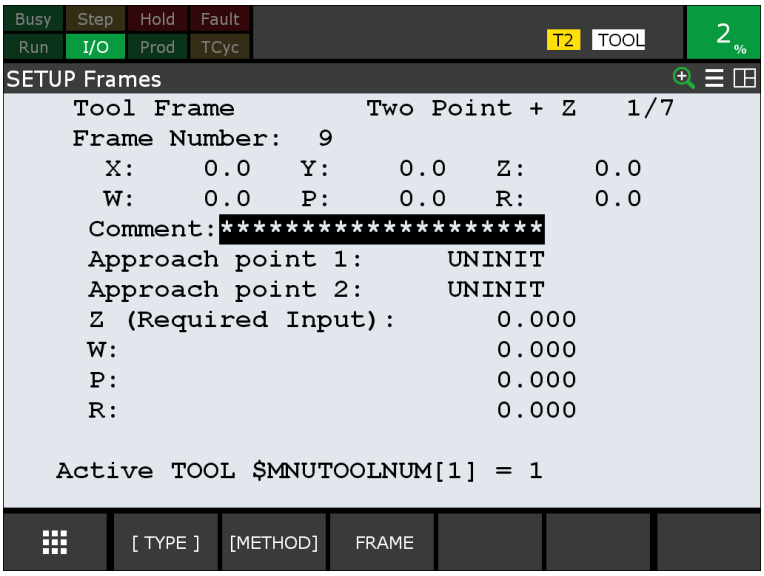
4 Move the cursor to the line of the tool frame number to set and press F2 [DETAIL].
The setup screen for the tool frame for the selected frame number will appear.
The screen is an example of when performing setup using Tool 9.



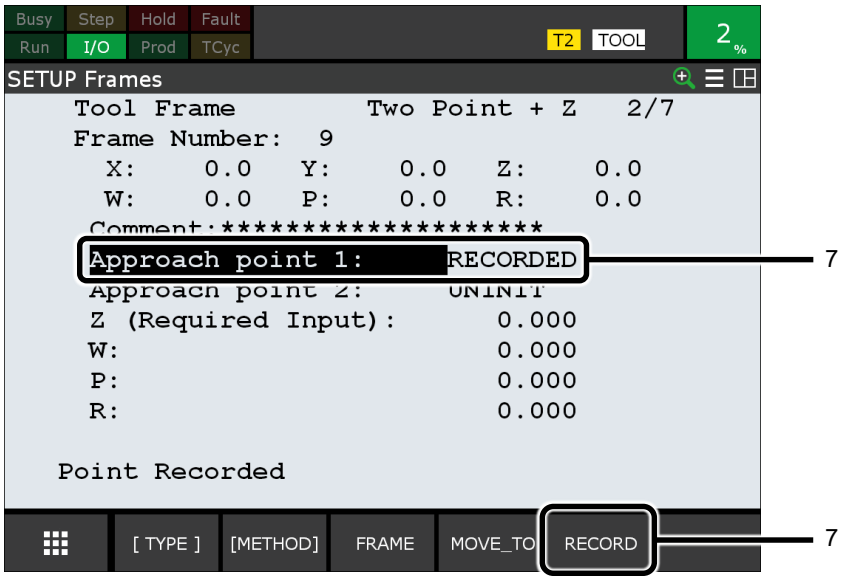
5 Press F2 [METHOD].
A menu will appear.
6 Select [Two Point + Z] from the menu.



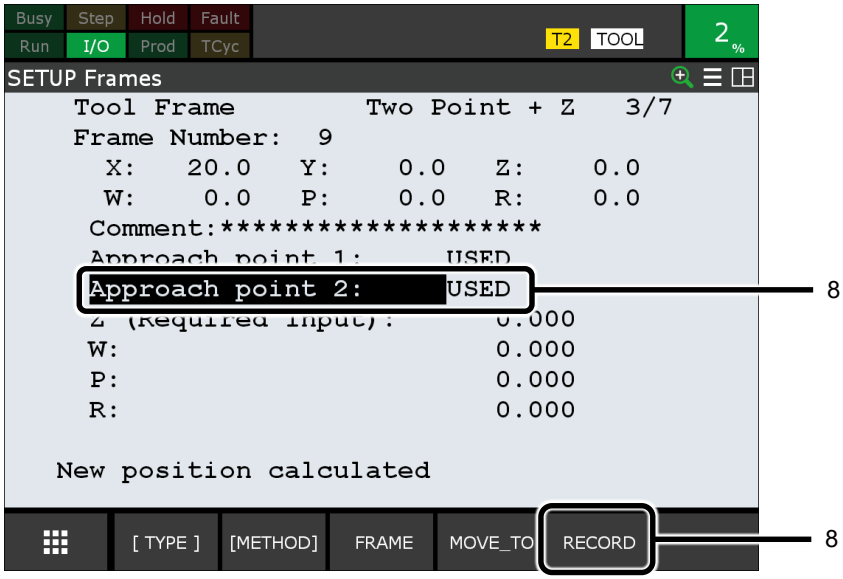
A screen like the following one will appear.



7 Move the cursor to [Approach point 1] and input approach point 1.
Move the robot by jog operation to the point that must be recorded.
While holding down the [SHIFT] key on the teach pendant, press F5 [RECORD] and input the current value as the approach point.
'RECORDED' will be displayed.



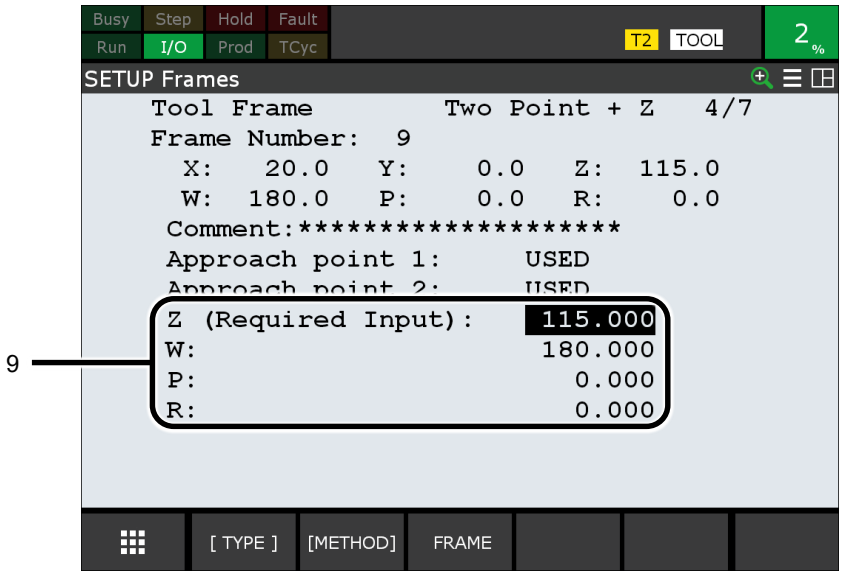
8 Move the cursor to [Approach point 2] and input approach point 2.
Move the robot by jog operation to the point that must be recorded.
While holding down the [SHIFT] key on the teach pendant, press F5 [RECORD] and input the current value as the approach point.
When all the approach points are input, 'USED' will be displayed and the X and Y of the tool frame will be set.



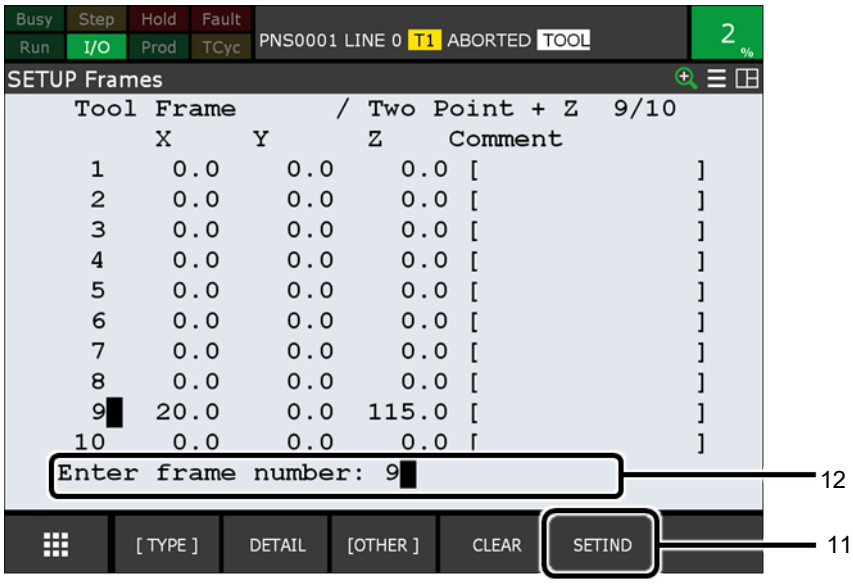
⚠ CAUTION

1 When inputting approach points, have the flange surface facing down.
2 Input approach points 1 and 2 at different positions.
3 If the above conditions are not met, a message saying 'Invalid set of input points' will appear.

9 Move the cursor to [Z], [W], [P] and [R] and input a value for each.
For [Z], input a numerical value measured with a ruler, etc.
Input [W], [P] and [R] directly.
* (W, P, R) = (180, 0, 0) is recommended.

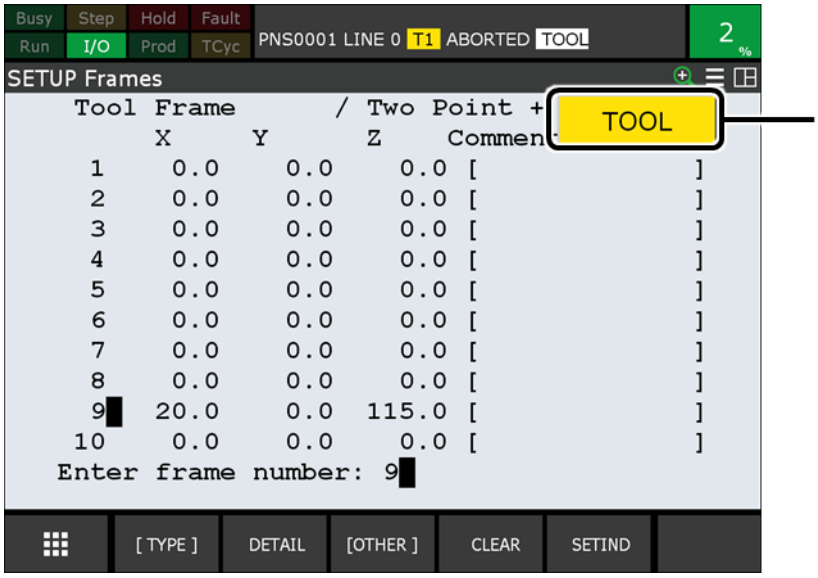


10 Press the [PREV] key on the teach pendant of the robot controller.
The list screen for tool frames will appear.
The setting values of all the tool frames can be checked.
11 Press F5 [SETIND].
12 Input the frame number of the tool frame that has been set.



The tool frame will be enabled.
* If you do not to carry out steps 11 and 12, the frame that has been set up will not be enabled.

13 Set the manual continuous feed of the robot in 'TOOL.'
Press and hold the [COORD] key on the teach pendant of the robot controller until the manual continuous feed operation screen switches to 'TOOL'.



14 Press the -Z (J6) key or +Z (J6) key on the teach pendant of the robot controller.
Apply a jog operation to the rotation of the Z axis.
Check that the tool frame (TCP) has been set accurately.

3.1.5 Creating a Robot Program

Create a robot program.
Create the following programs for robot programs.
- Main program
- Tracking program
- Vacuum ON/OFF program

Creating a main program

Create a main program.

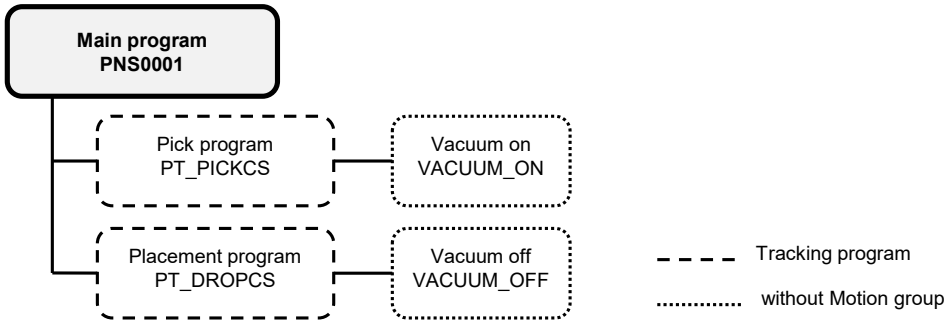
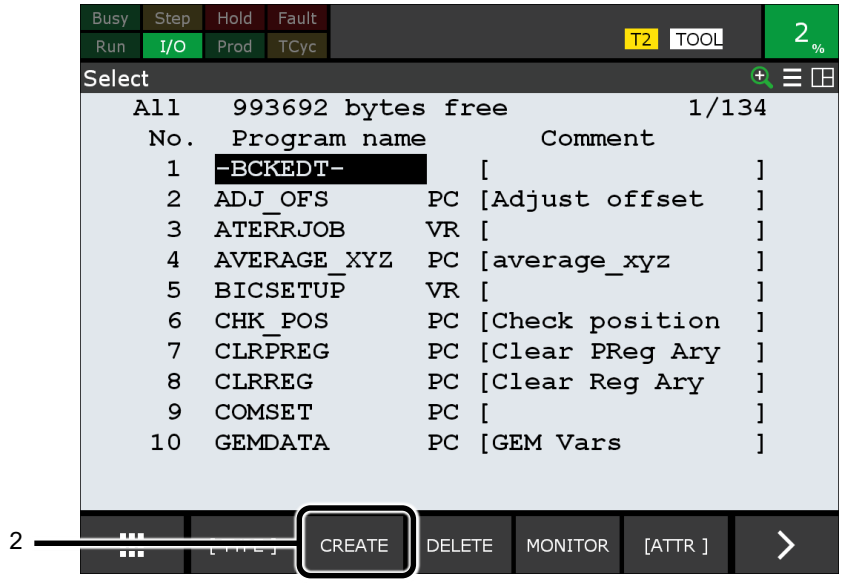


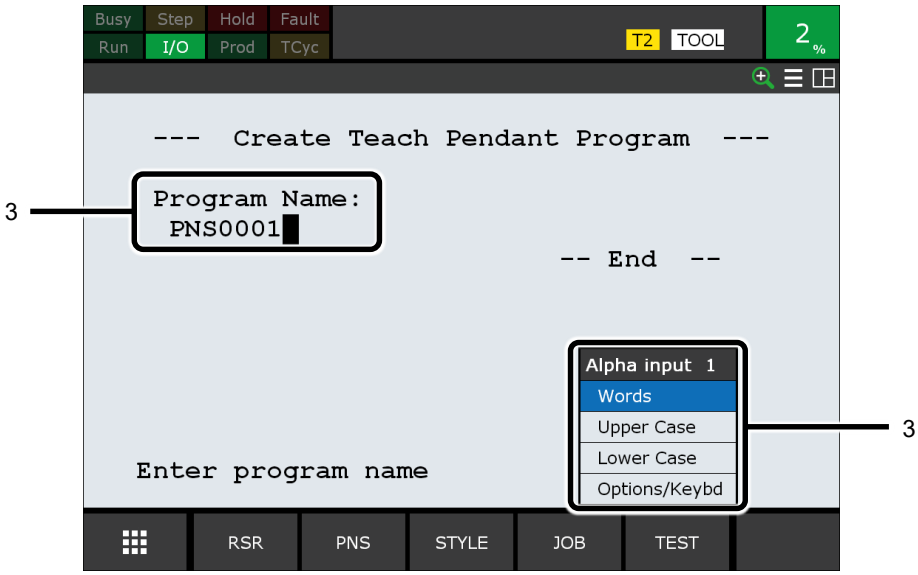
Fig. 3.1.5 (a) Main program positional relationship

1 Press the [SELECT] key on the teach pendant of the robot controller.
A program list screen will appear.
2 Press F2 [CREATE].
A screen to create programs will appear.
If [CREATE] is not displayed, press [> (NEXT)] to switch screens.

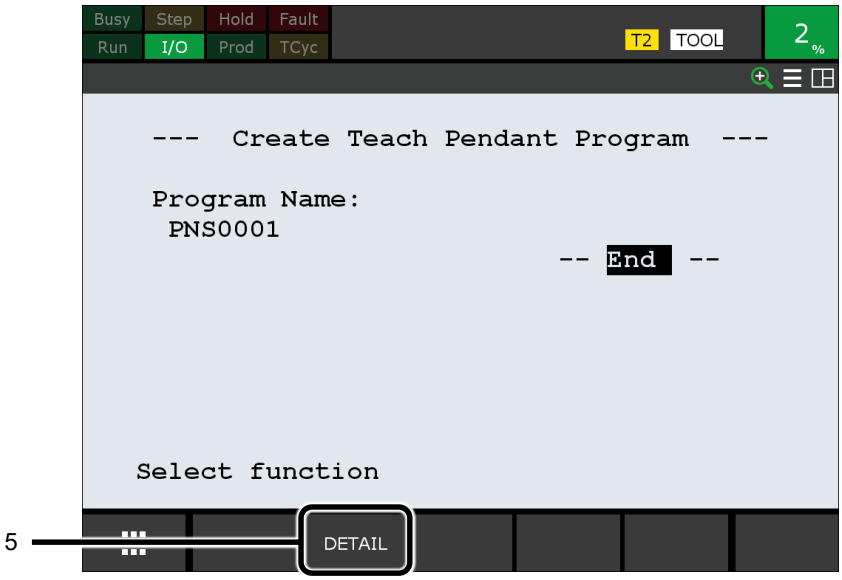


3 Move the cursor to [Program Name] and input a program name.
The input method for program names can be selected from [Words], [Upper Case], [Lower Case] and [Options/Keybd], which are displayed in a pop-up.
Example : Here, the name is set to PNS0001, on the assumption that PNS will start up.

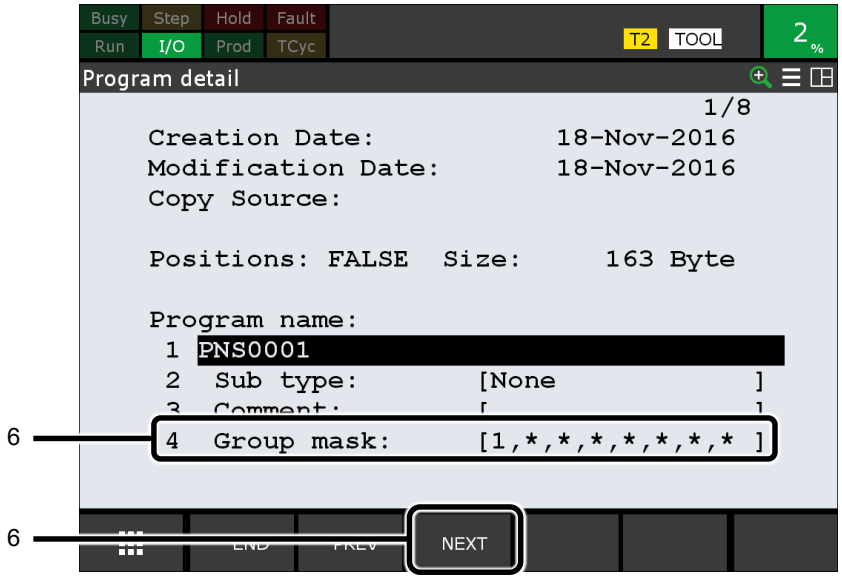
4 Press the [ENTER] key on the teach pendant of the robot controller.



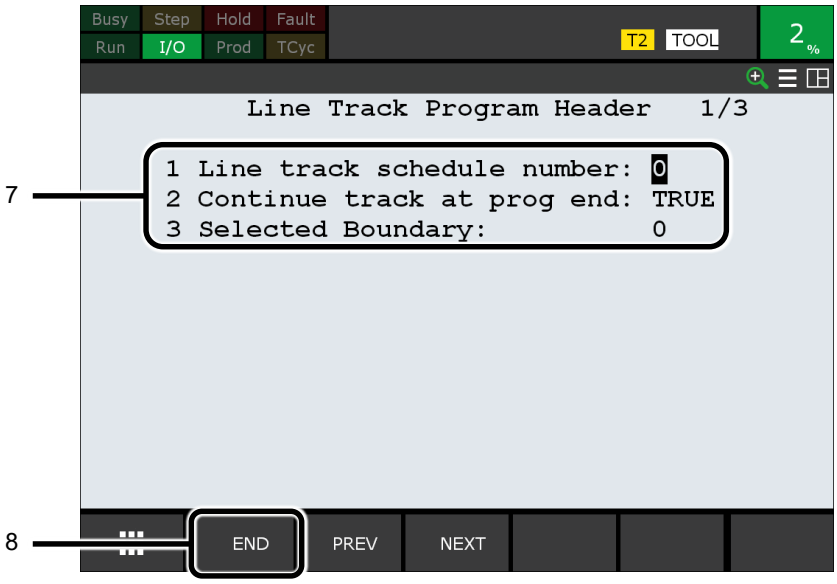
5 Press F2 [DETAIL].
A program details screen will appear.



6 Confirm that '1' has been input for the first group of [Group mask] and press F3 [NEXT].
A tracking program details screen will appear.



7 Confirm that '0' has been set in [Line track schedule number].
8 Press [END].
A main program will be created.
* [Group mask] : [1,*,*,*,*,*,*,*]
[Line track schedule number] : 0
* For programs, refer to the program samples given below.



Creating a tracking program

Create a tracking program. For the tracking program, create a 'pick program' and 'placement program'.

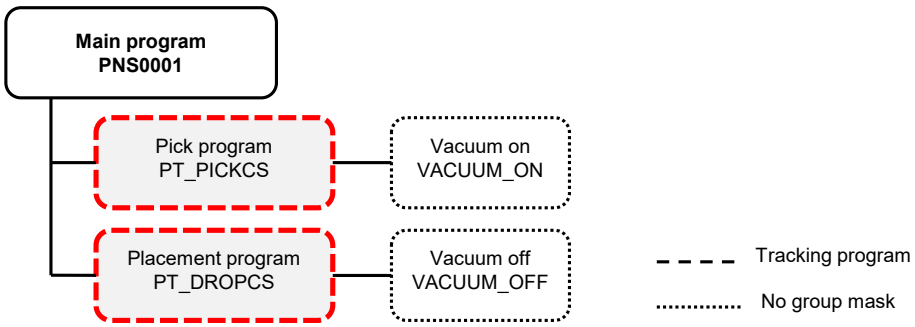
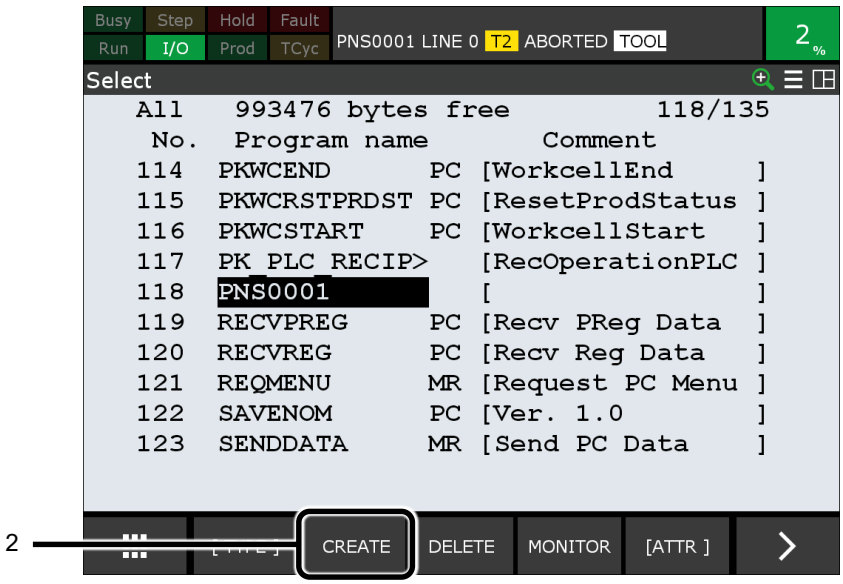


Fig. 3.1.5 (b) Tracking program positional relationship

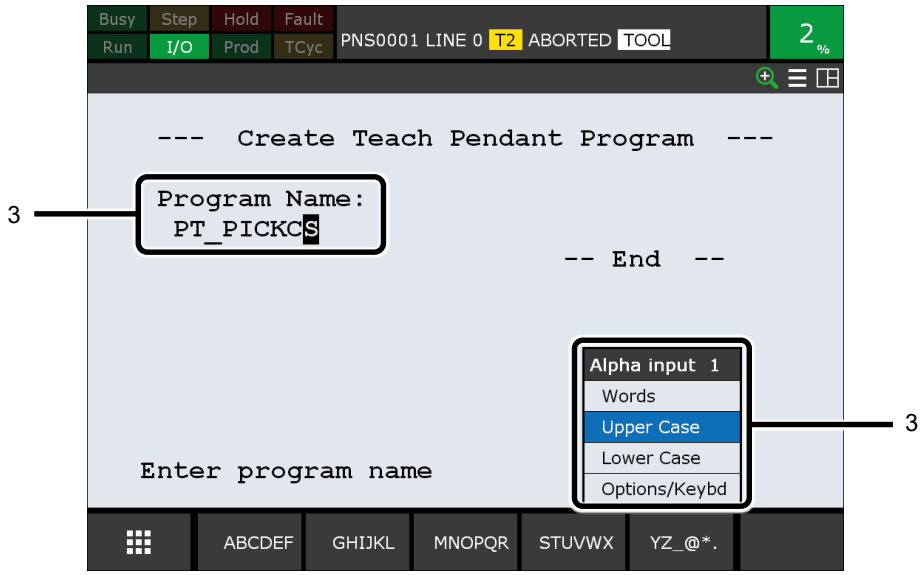
1 Press the [SELECT] key on the teach pendant of the robot controller.
A program list screen will appear.

2 Press F2 [CREATE].
A screen to create programs will appear.
If F2 [CREATE] is not displayed, press [> (NEXT)] to switch screens.

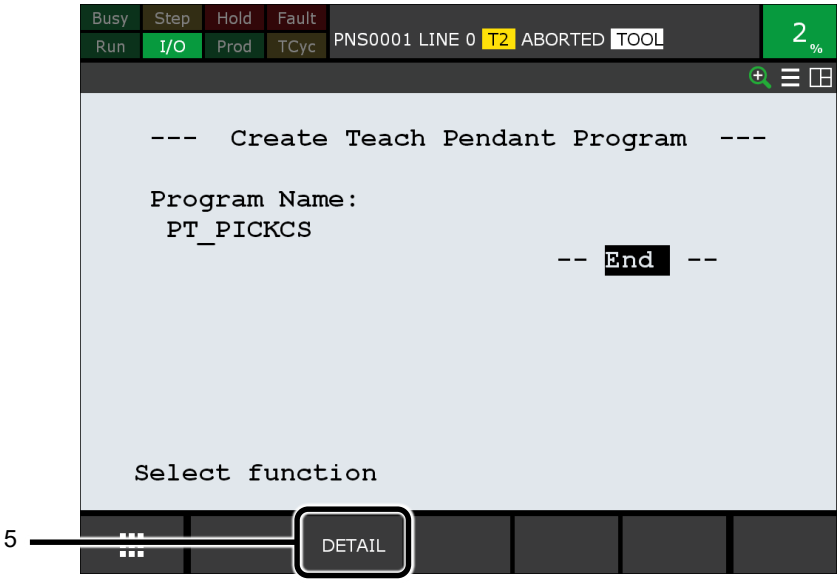


3 Move the cursor to [Program Name] and input a program name.
The input method for program names can be selected from [Words], [Upper Case], [Lower Case] and [Options/Keybd], which are displayed in a pop-up.
Example : PT_PICKCS, PT_DROPCS

4 Press the [ENTER] key on the teach pendant of the robot controller.

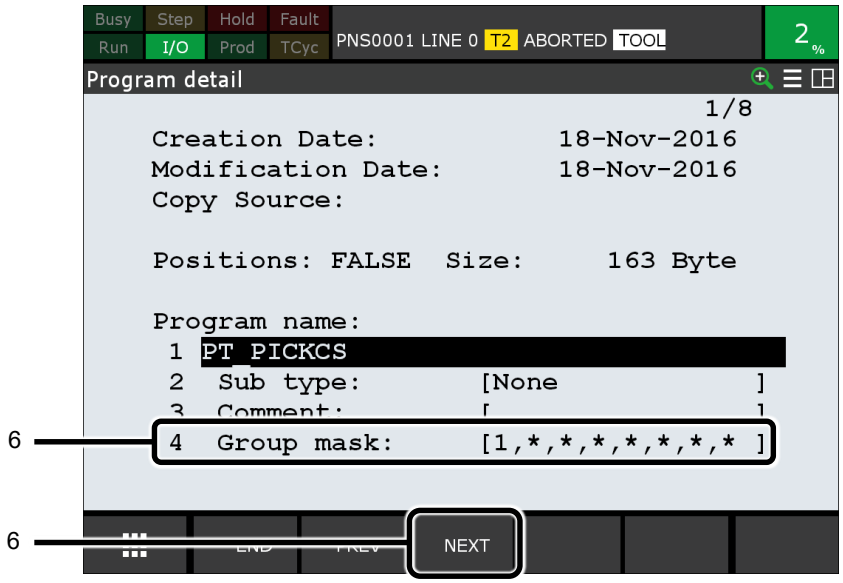


5 Press F2 [DETAIL].



A program details screen will appear.

6 Confirm that '1' has been input for the first group of [Group mask] and press F3 [NEXT].

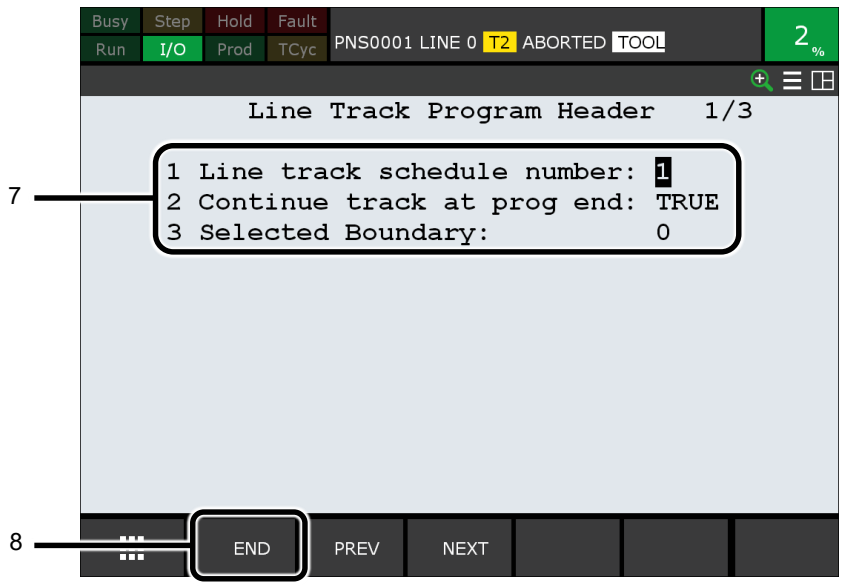


The following tracking program details screen will appear.

7 Set [1] or [2] in [Line track schedule number].
In the case of picking (PICK) : 1
In the case of placing (DROP) : 2

8 Press F1 [END].
A tracking program will be created.
* [Group mask] : [1,*,*,*,*,*,*,*]
[Line track schedule number] : Picking (PICK) : 1
Placing (DROP) : 2

* For programs, refer to the program samples given below.



Creating a vacuum ON/OFF program

Create a vacuum ON/OFF program. For the vacuum ON/OFF program, set vacuum ON for 'Pick program' and vacuum OFF for 'Placement program.'

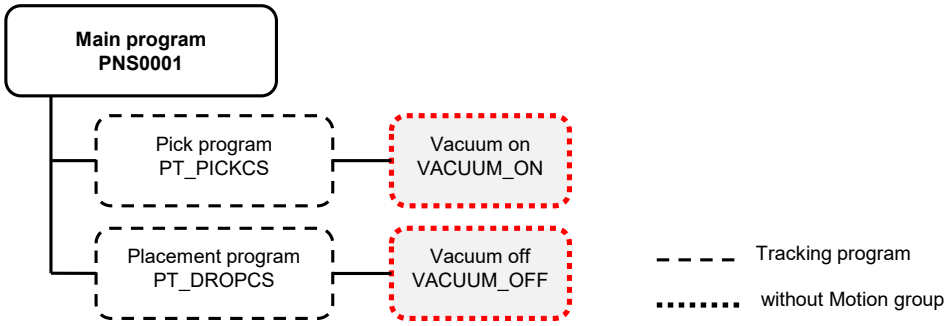


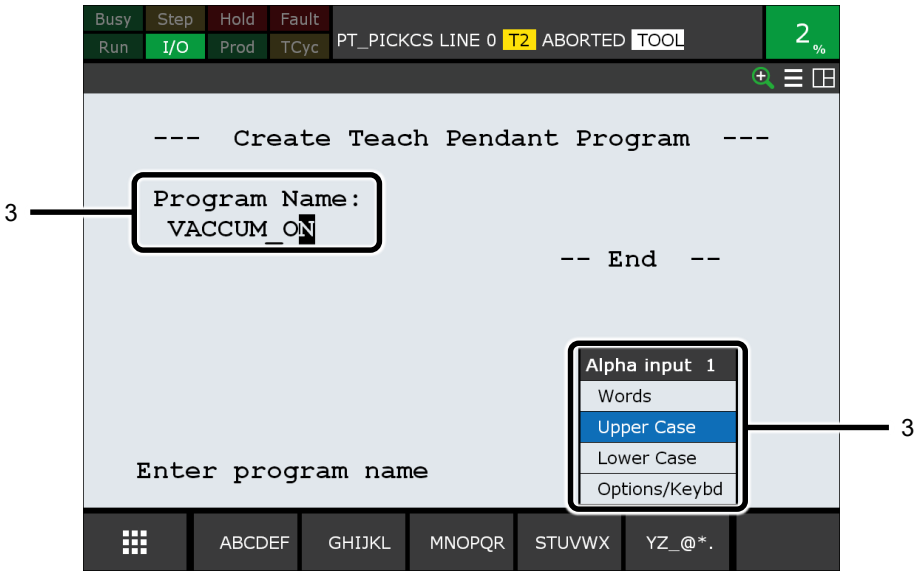
Fig. 3.1.5 (c) Vacuum ON/OFF program positional relationship

1 Press the [SELECT] key on the teach pendant of the robot controller.
A program list screen will appear.

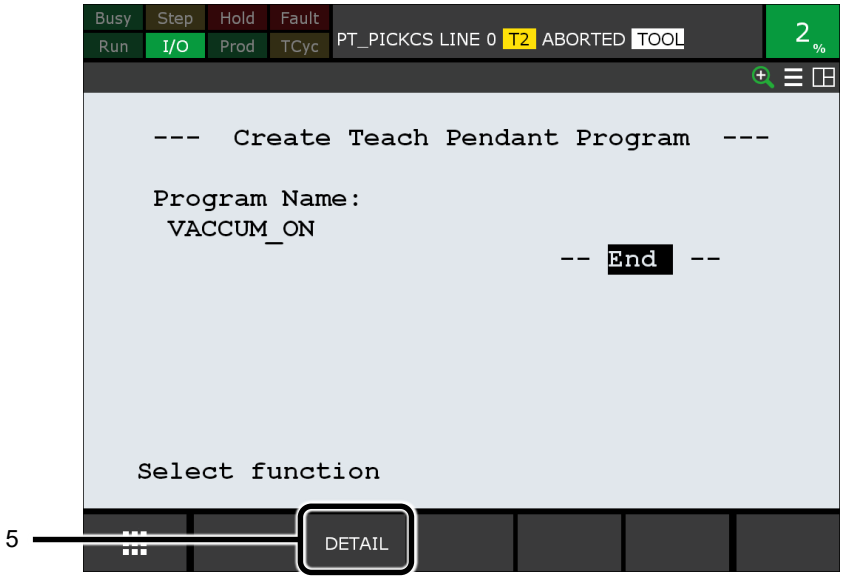
2 Press F2 [CREATE].
A screen to create programs will appear.
If F2 [CREATE] is not displayed, press [> (NEXT)] to switch screens.

3 Move the cursor to [Program Name] and input a program name.
The input method for program names can be selected from [Words], [Upper Case], [Lower Case] and [Options/Keybd], which are displayed in a pop-up.
Example : VACUUM_ON, VACUUM_OFF

4 Press the [ENTER] key on the teach pendant of the robot controller.



5 Press F2 [DETAIL].
A program details screen will appear.



6 Move the cursor to [Group mask] and change all of the group mask to '*.'
You can change it by pressing F5 [*] after moving the cursor to the location you want to change.

7 Press F1 [END].
A vacuum ON/OFF program will be created.
* [Group mask] : [*,*,*,*,*,*,*,*]
[Line track schedule number] : 0
* For programs, refer to the program samples given below.

Sample programs

Here are examples for each of the programs described in this section.
For information on KAREL programs, refer to “Using KAREL programs” below.
An example of a main program (PNS0001) is shown below.

PNS0001

```
   1:  !PKCSGETID
   2:  ! (Get conveyor station ID)
   3:  !*1:Conveyor name
   4:  !*2:Register number
   5:  !     for conveyor station ID
   6:  !*3:Group number (Can be omitted)
   7:  CALL PKCSGETID('CONV1',CStn ID Reg=11)
   8:  CALL PKCSGETID('CONV2',CStn ID Reg=12)
   9:
  10:  UFRAME_NUM=0
  11:  UTOOL_NUM=1
  12:  PAYLOAD[1]
  13:L PR[1:HOME POS] 500mm/sec FINE Wjnt
  14:
  15:  !--- System startup ---
  16:  !PKWCSTART (System startup,
  17:  ! including check and clear)
  18:  CALL PKWCSTART
  19:
  20:  LBL[101]
  21:  !--- Pick program ---
  22:  CALL PT_PICKCS
  23:  IF DI[1:CYCLE_STOP]=ON,JMP LBL[901]
  24:  !--- Placement program ---
  25:  CALL PT_DROPCS
  26:  IF DI[1:CYCLE_STOP]=ON,JMP LBL[901]
  27:  JMP LBL[101]
  28:
  29:  LBL[901]
  30:  UFRAME_NUM=0
  31:  UTOOL_NUM=1
  32:L PR[1:HOME POS] 500mm/sec FINE Wjnt
  33:
  34:  !PKWCEND (System end)
  35:  CALL PKWCEND
[End]
```

KAREL program

KAREL program

KAREL program

KAREL program

An example of a tracking program (pick program) (PT_PICKS) is shown below.

PT_PICKCS

```
   1:  UFRAME_NUM=0
   2:  UTOOL_NUM=1
   3:
   4:  LBL[101:QUE check]
   5:  STOP_TRACKING
   6:  !PKCSGETQUE(Get part information)
   7:  !*1:Register number
   8:  !      for conveyor station ID
   9:  !*2:Usually 1.
  10:  !      2 or more in the case of
  11:  !      consecutive acquisition)
  12:  !*3:Waiting time msec
  13:  !*4:Vision register number
  14:  !      for part information
  15:  !*5:Process status
  16:  !      register number
  17:  !*6:Model ID (Can be omitted)
  18:  CALL PKCSGETQUE(CStn ID=R[11:CONV1 ID],
       Consec Flag=1,Timeout (ms)=100,
       Offset VR=1,Stat Reg=2)
  19:  IF R[2:GETQ Status Pick]=0,JMP LBL[111]
  20:  IF DI[1:CYCLE_STOP]=ON,JMP LBL[901]
  21:  JMP LBL[101]
  22:
  23:  LBL[111:Pick operation]
  24:  !ADJ_OFS (Adjust offset)
  25:  !*1:Usually 1 (vision register)
  26:  !*2:Vision register number
  27:  !      before adjustment
  28:  !*3:Position register number
  29:  !      for adjustment amount
  30:  !*4:Vision register number
  31:  !      after adjustment
  32:  CALL ADJ_OFS(1,1,21,1)
  33:L PR[7:Pick Pos] R[26:Pick Speed1]mm/sec
       CNT100 VOFFSET,VR[1] Offset,PR[11:Pick Apr Z]
  34:L PR[7:Pick Pos] R[27:Pick Speed2]mm/sec
       CNT R[24:Pick CNT] VOFFSET,VR[1] AP_LDR[21] TB R[25]sec,
       CALL VACUUM_ON
  35:  WAIT R[23]
  36:  PAYLOAD[2]
  37:L PR[7:Pick Pos] R[28:Pick Speed3]mm/sec
       CNT100 VOFFSET,VR[1] Offset,PR[11:Pick Apr Z] RT_LDR[22]
  38:
  39:  !PKCSACKQUE (Part information
  40:  ! process notification)
  41:  !*1:Register number
  42:  !      for conveyor station ID
  43:  !*2:Processing results.
  44:  !      Normally 1
  45:  CALL PKCSACKQUE(CStn ID=R[11:CONV1 ID],Success=1)
  46:
  47:  LBL[901:Pick finished]
[End]
```

KAREL program

KAREL program

Pick approach position

Pick position

Pick retract position

KAREL program

An example of a tracking program (placement program) (PT_DROPCS) is shown below.

PT_DROPCS

```
   1:  UFRAME_NUM=0
   2:  UTOOL_NUM=1
   3:
   4:  LBL[101:QUE check]
   5:  STOP_TRACKING
   6:  !PKCSGETQUE(Get part information)
   7:  !*1:Register number
   8:  !      for conveyor station ID
   9:  !*2:Usually 1.
  10:  !      2 or more in the case of
  11:  !      consecutive acquisition)
  12:  !*3:Waiting time msec
  13:  !*4:Vision register number
  14:  !      for part information
  15:  !*5:Process status
  16:  !      register number
  17:  !*6:Model ID (Can be omitted)
  18:  CALL PKCSGETQUE(CStn ID=R[12:CONV2 ID],
       Consec Flag=1,Timeout (ms)=100,
       Offset VR=2,Stat Reg=3)
  19:  IF R[3:GETQ Status Drop]=0,JMP LBL[111]
  20:  IF DI[1:CYCLE_STOP]=ON,JMP LBL[901]
  21:  JMP LBL[101]
  22:
  23:  LBL[111:Place operation]
  24:L PR[8:Drop Pos] R[36:Drop Speed1]mm/sec
       CNT100 VOFFSET,VR[2] Offset,PR[12:Drop Apr Z] Wjnt
  25:L PR[8:Drop Pos] R[37:Drop Speed2]mm/sec
       CNT R[34:Drop CNT] VOFFSET,VR[2] AP_LDR[31] TB R[35]sec,
       CALL VACUUM_OFF
  26:  WAIT R[33]
  27:  PAYLOAD[1]
  28:L PR[8:Drop Pos] R[38:Drop Speed3]mm/sec
       CNT100 VOFFSET,VR[2] Offset,PR[12:Drop Apr Z] RT_LDR[32]
  29:
  30:  !PKCSACKQUE (Part information
  31:  ! process notification)
  32:  !*1:Register number
  33:  !      for conveyor station ID
  34:  !*2:Processing results.
  35:  !      Normally 1
  36:  CALL PKCSACKQUE(CStn ID=R[12:CONV2 ID],Success=1)
  37:
  38:  LBL[901:Place finished]
[End]
```

Drop position

Place approach position

KAREL program

KAREL program

Place retract position

An example of a vacuum ON program (VACUUM_ON) is shown below.

VACUUM_ON

```
   1:  DO[2:Vacuum burst command]=OFF
   2:  DO[1:Vacuum command]=ON
[End]
```

An example of a vacuum OFF program (VACUUM_OFF) is shown below.

VACUUM_OFF

```
   1:  DO[1:Vacuum command]=OFF
   2:  DO[2:Vacuum burst command]=PULSE,0.3sec
[End]
```


Register list

Display a register list screen using the following procedure.

1 Press the [DATA] key on the teach pendant of the robot controller.
2 Press F1 [TYPE].
  A menu will appear.
3 Select [Register] from the menu.
  A register list screen will be displayed.

Table 3.1.5 (a) Register list

R[2:GETQ Status Pick]=0	Pick part processing status
R[3:GETQ Status Drop]=0	Place part processing status
R[11:CONV1 ID]=1	CONV1 (pick) ID
R[12:CONV2 ID]=2	CONV 2 (place) ID
R[21:Pick AP_LD]=30	Pick approach distance specification [mm]
R[22:Pick RT_LD]=30	Pick retract distance specification [mm]
R[23:Pick Wait]=.05	Index delay at pick position [sec]
R[24:Pick CNT]=0	Pick position CNT specification
R[25:Pick TB]=.1	TIME BEFORE time (VACUUM_ON) [sec]
R[26:Pick Speed1]=4000	Pick program speed 1 [mm/sec]
R[27:Pick Speed2]=4000	Pick program speed 2 [mm/sec]
R[28:Pick Speed3]=2000	Pick program speed 3 [mm/sec]
R[31:Drop AP_LD]=30	Place approach distance specification
R[32:Drop RT_LD]=30	Place retract distance specification
R[33:Drop Wait]=.05	Index delay at drop position [sec]
R[34:Drop CNT]=0	Drop position CNT specification
R[35:Drop TB]=.1	TIME BEFORE time (VACUUM_OFF) [sec]
R[36:Drop Speed1]=2000	Placement program speed 1 [mm/sec]
R[37:Drop Speed2]=2000	Placement program speed 2 [mm/sec]
R[38:Drop Speed3]=4000	Placement program speed 3 [mm/sec]

Display a position register list screen using the following procedure.

1 Press the [DATA] key on the teach pendant of the robot controller.
2 Press [TYPE].
  A menu will appear.
3 Select [Position Reg.] from the menu.
  The position register list screen will appear.

Table 3.1.5 (b) Position register list

PR[1:HOME POS]=R	Home position
PR[7:Pick Pos]=R	Pick position
PR[8:Drop Pos]=R	Drop position
PR[11:Pick Apr Z]=R	Offset from pick position
PR[12:Drop Apr Z]=R	Offset from drop position
PR[21:ADJ_OFS]=R	For pick position offset adjustment

Set all the initial values for the amount of adjustment for the pick position offset to '0.'

PR[21]	UF:F	UT:F		CONF: 0	
X	0.000	mm	W	0.000	deg
Y	0.000	mm	P	0.000	deg
Z	0.000	mm	R	0.000	deg

Robot operation

The pick / place operation of the robot will be like in the following figure.

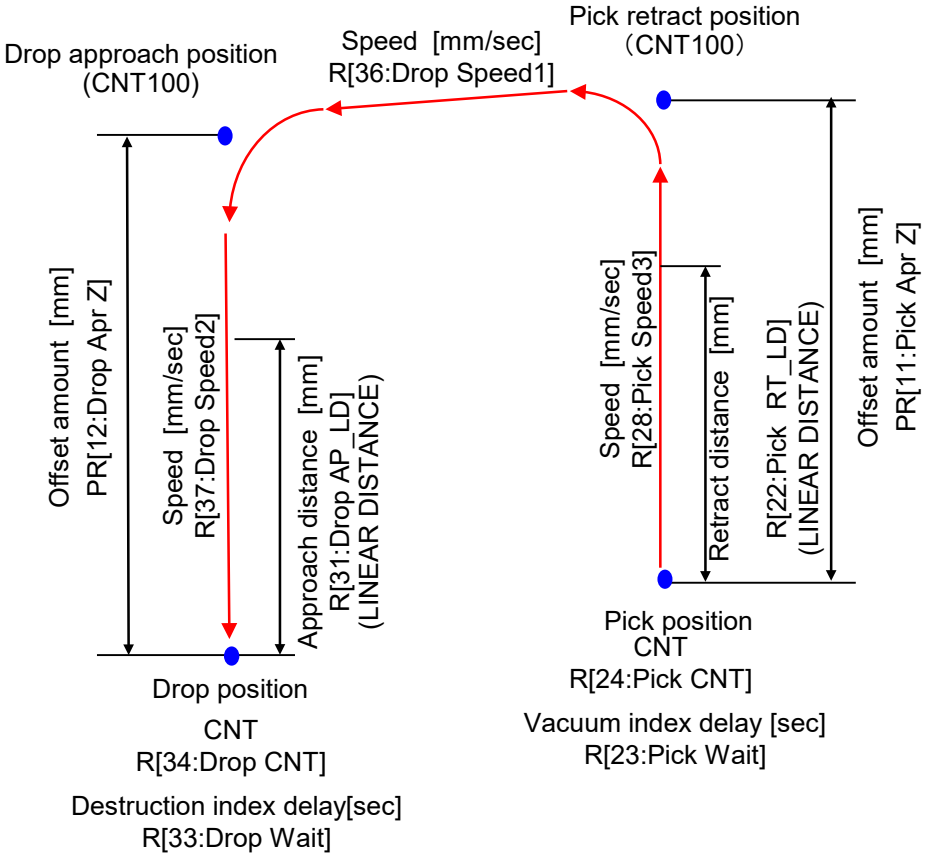


Fig. 3.1.5 (d) Robot operation (Pick → Place)

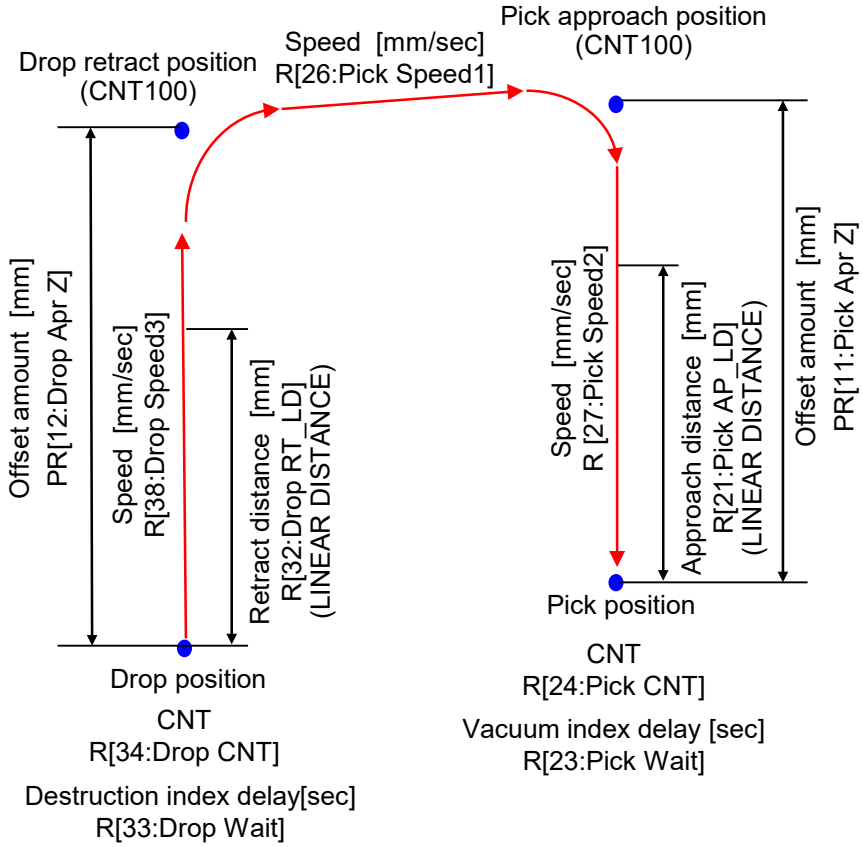


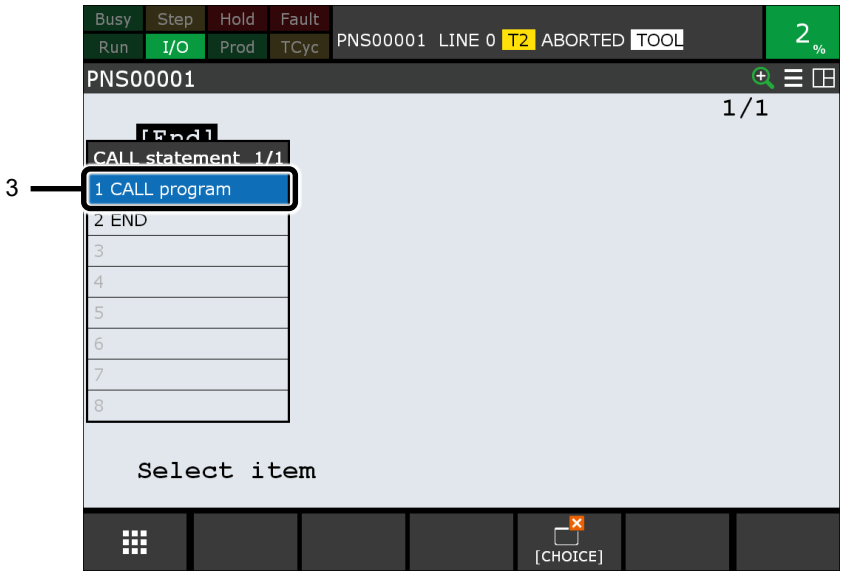
Fig. 3.1.5 (e) Robot operation (Place → Pick)

Using KAREL programs

To use a KAREL program, add it using the following procedure.

1 Press F1 [INST] on the program edit screen.
2 Select [CALL] from the menu.

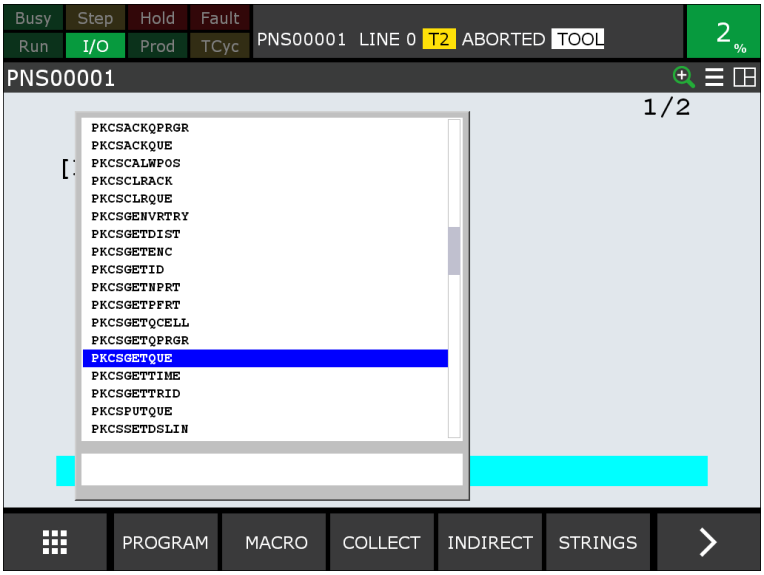
3 Select [CALL program] from the menu.



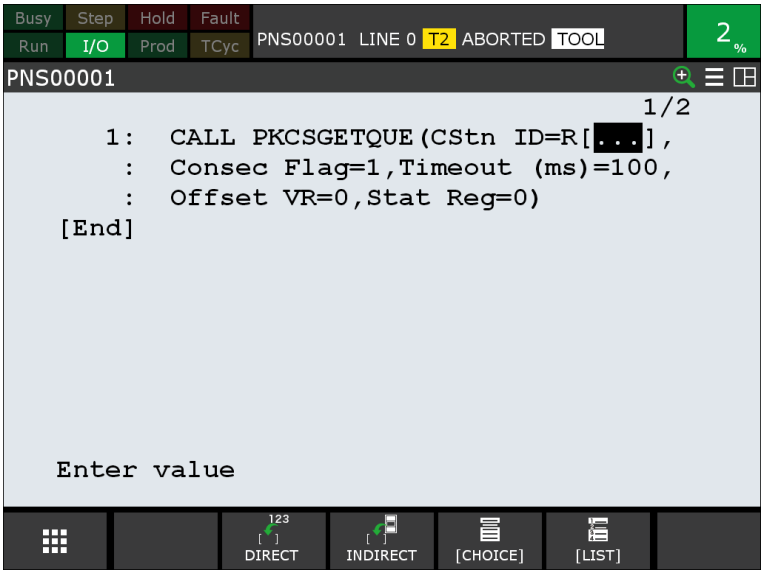
4 Press F3 [COLLECT].
'KAREL Progs' will be displayed in a list.

5 Select 'KAREL Progs' and press the [ENTER] key on the teach pendant of the robot controller.
A KAREL program list screen will appear.

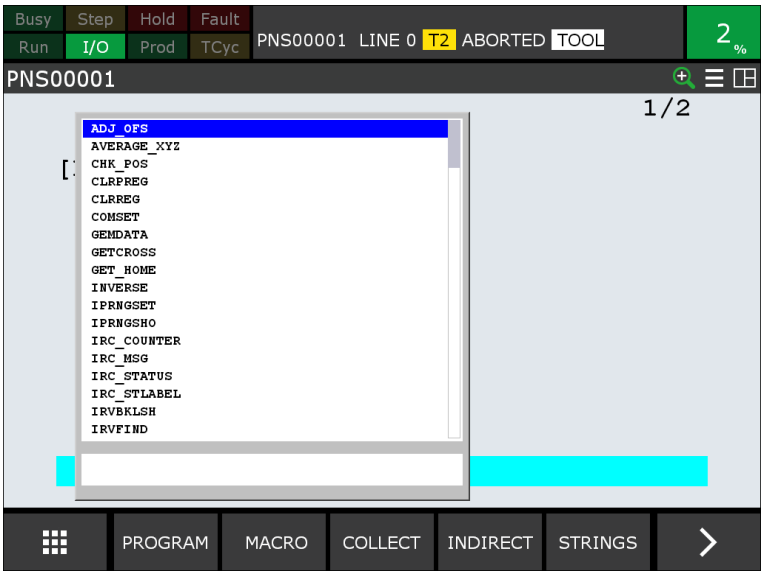
6 Select the KAREL program to call.
On the program editing screen, the KAREL program will be added.
The following screen is an example of calling a KAREL program that starts with 'PK.'



For KAREL programs that start with 'PK,' arguments will be displayed automatically.
If you want to change an argument, move the cursor to the argument and input the required value.
The operation is now complete.



The following screen is an example of calling a KAREL program other than one that starts with 'PK.'



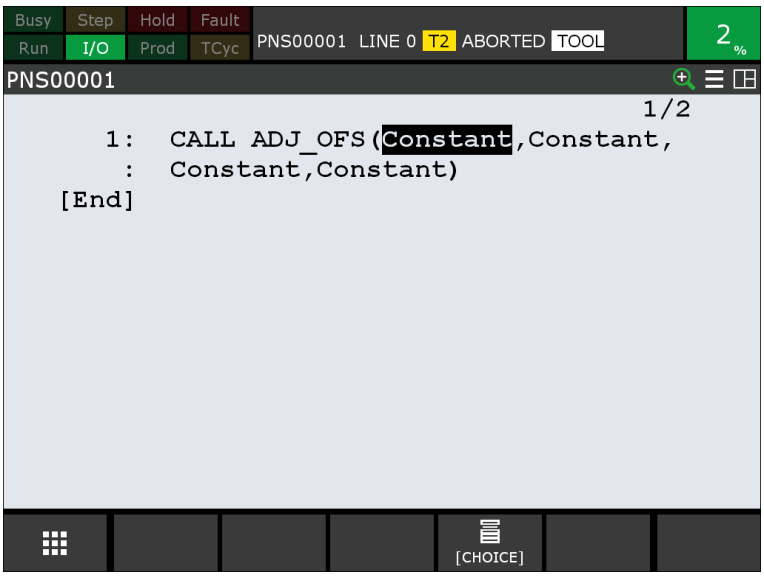
The following procedure is required only when you call a KAREL program other than one that starts with 'PK.'
This procedure is explained using a case of calling 'ADJ_OFS' as an example.

7 Move the cursor to the right side of 'ADJ_OFS,' which has been added to the program editing screen, and press F4 [CHOICE].

If you want to add an argument, move the cursor to the ')' on the right and press F4 [CHOICE].

8 Select [Constant] from the menu.

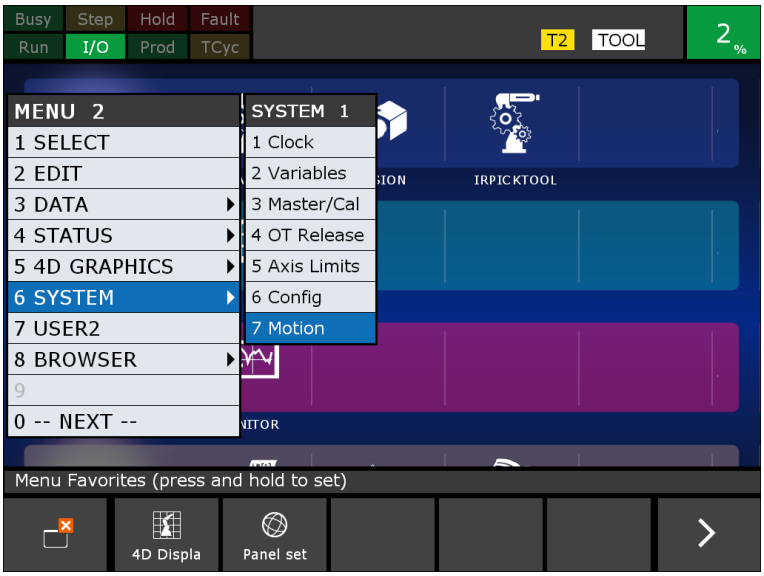
9 After adding the required number of arguments, input a numerical value for each constant.



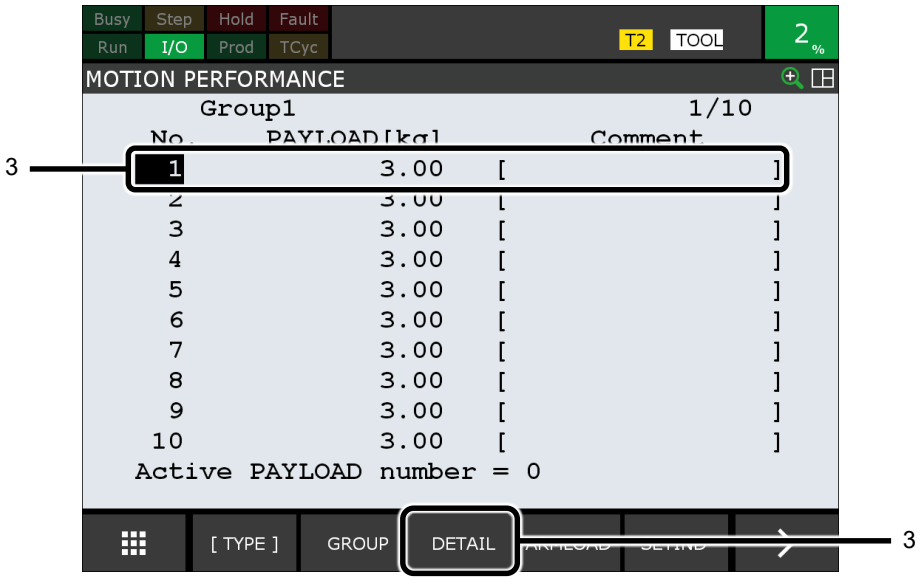
3.1.6 Setting Payloads (Motion Performance)

Set the payloads for the robot on the [MOTION PERFORMANCE] screen.

1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SYSTEM] → [Motion].

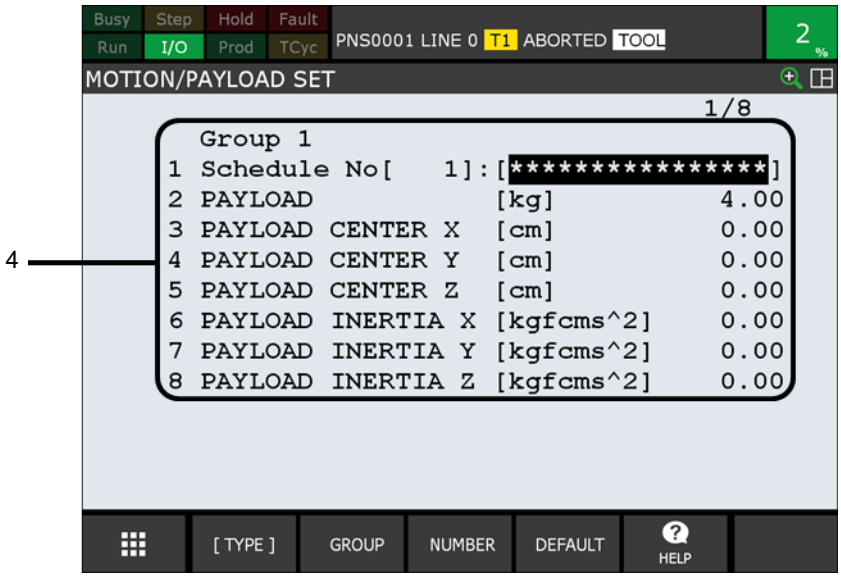


3 Move the cursor to [1] and press F3 [DETAIL].

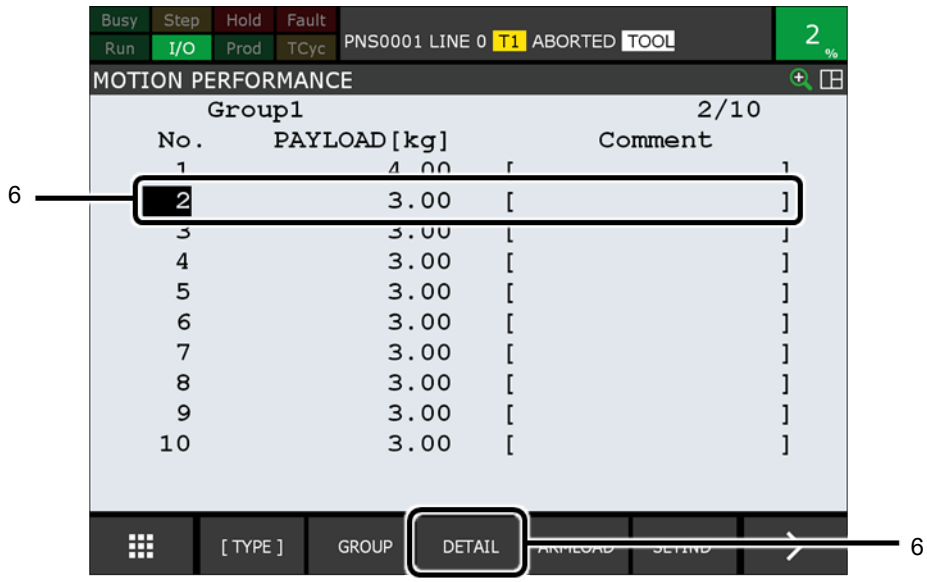


The setup screen for payload condition No.1 will appear.

4 At the least, input the [PAYLOAD]. Not all of the items necessarily need to be input.
* Like the sample programs (PT_PICKCS、PT_DROPCS) given above as programs for pick / place, if Set Payload [1] is used when the robot is not holding a part, set a payload for 'Hand only' in this step.

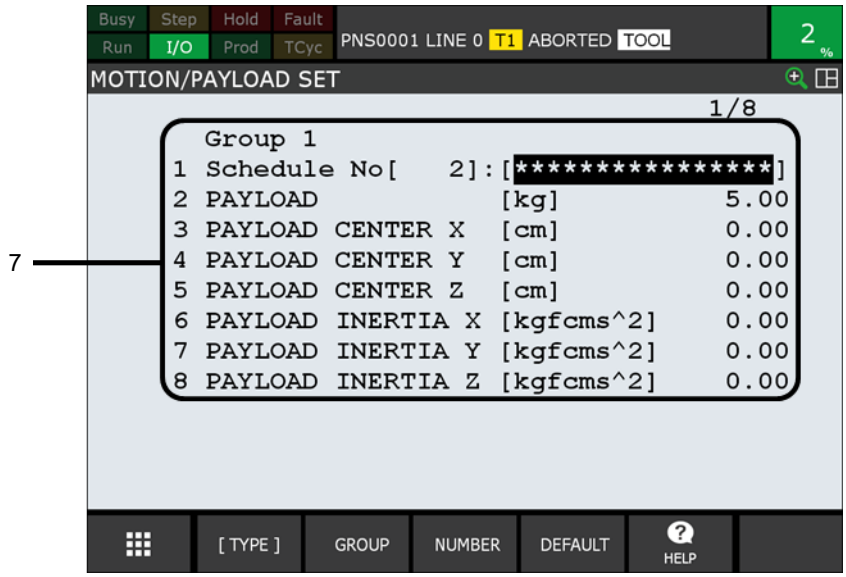


5 Press the [PREV] key on the teach pendant of the robot controller.
Return to the list on the motion performance screen.
6 Move the cursor to [2] and press F3 [DETAIL].



The setup screen for payload condition No.2 will appear.

7 At the least, input the [PAYLOAD]. Not all of the items necessarily need to be input.
* Like the sample programs (PT_PICKCS、PT_DROPCS) given above as programs for pick / place, if Set Payload [2] is used when the robot is holding a part, set a payload for 'Hand + part' in this step.



With a robot program, use the Set Payload [*] command to change the payload.

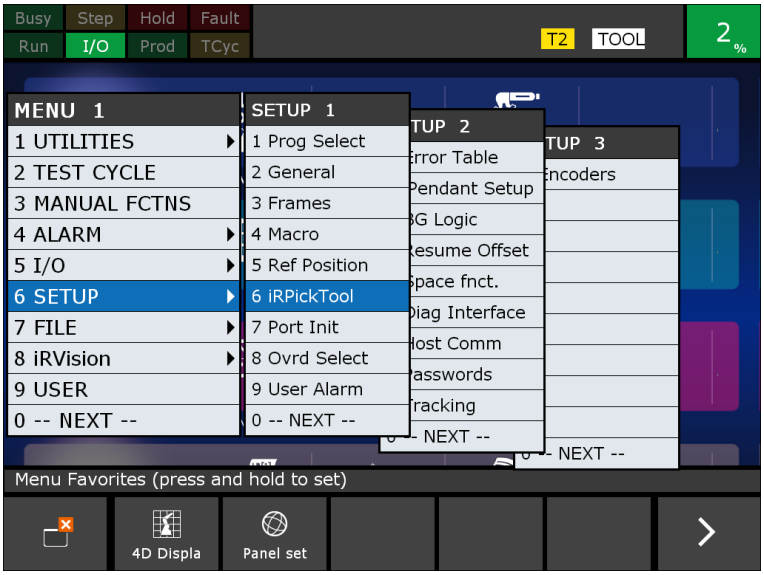
3.2 SETTING UP APPLICATIONS

This section explains the setup of *i*RPickTool applications, such as setting up workcells, sensors, etc.

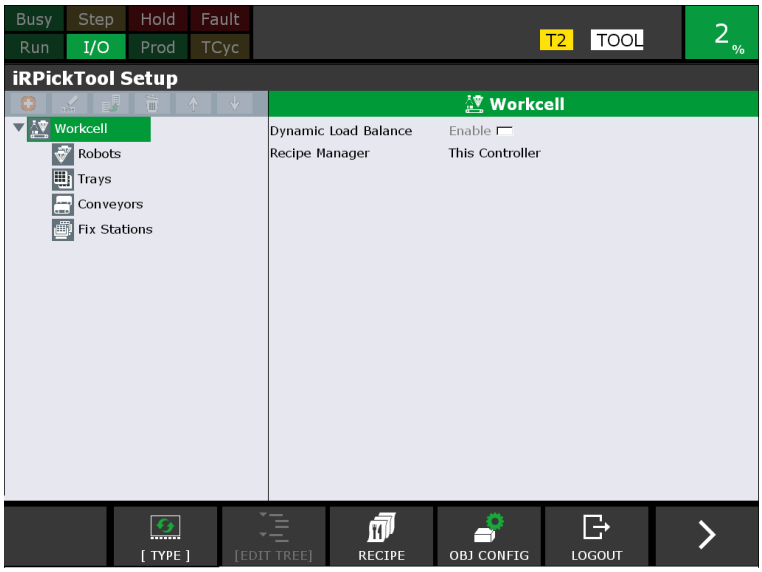
3.2.1 Workcell Setup

Set up a workcell.

1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SETUP] → [*i*RPickTool] from the menu.



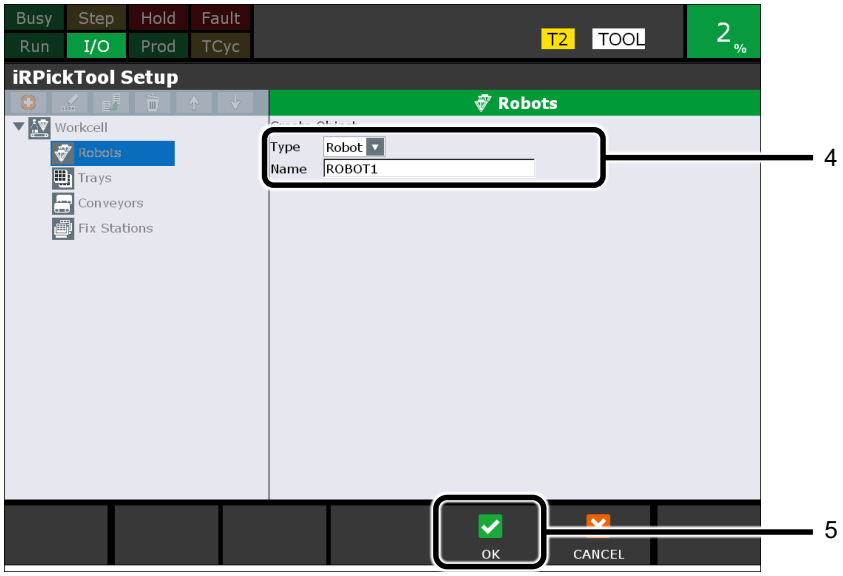
The following *i*RPickTool screen will appear.



3 Select [Robots], then press the [CREATE] button above the tree.

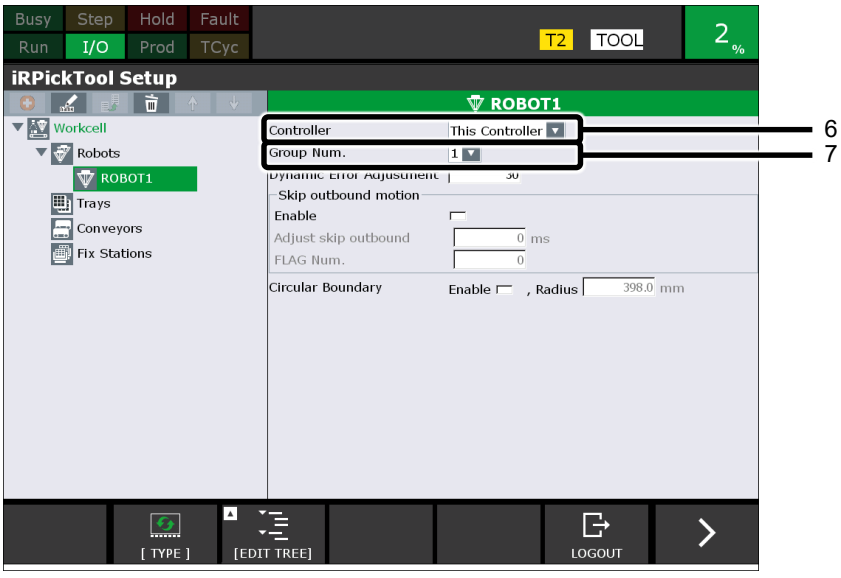
4 Input the name of the robot in [Name].
We recommend using the name that is displayed as the initial value just as it is.
Input using half-width alphanumeric characters and half-width katakana. The name must contain no spaces, not contain the symbol other than underscore, and start with a letter.

5 Press F4 [OK].
The object is created.

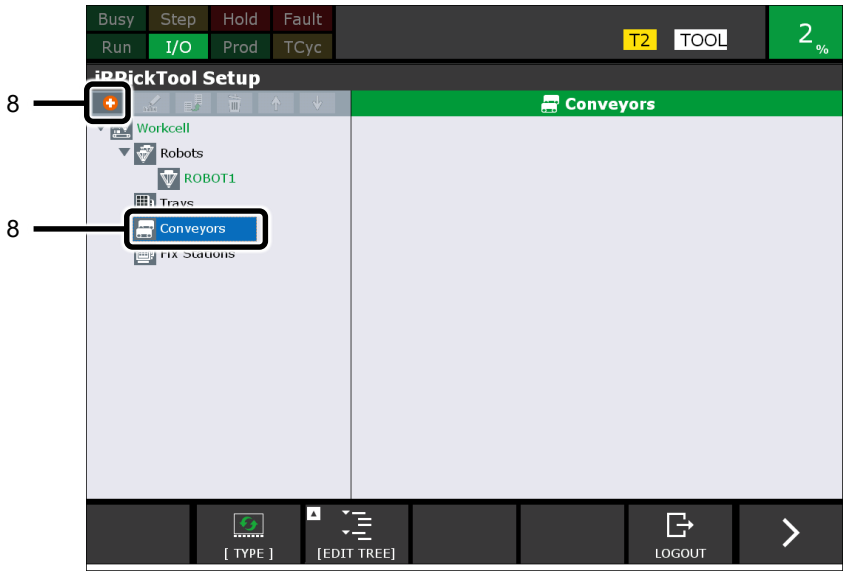


6 For [Controller], select [This Controller] from the menu (if there is one robot).

7 For [Group Num.], select [1] from the menu.

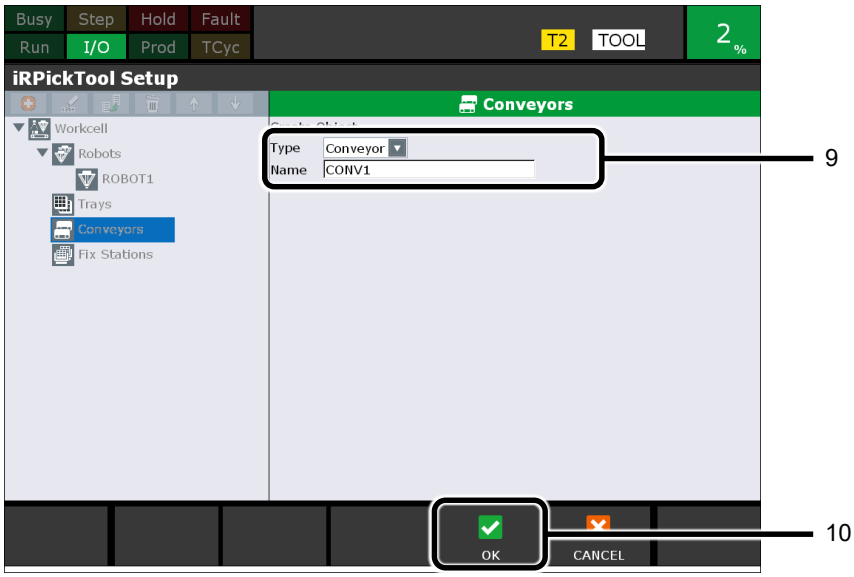


8 First, create an object for the infeed conveyor. Select [Conveyors], then press the [CREATE] button above the tree.

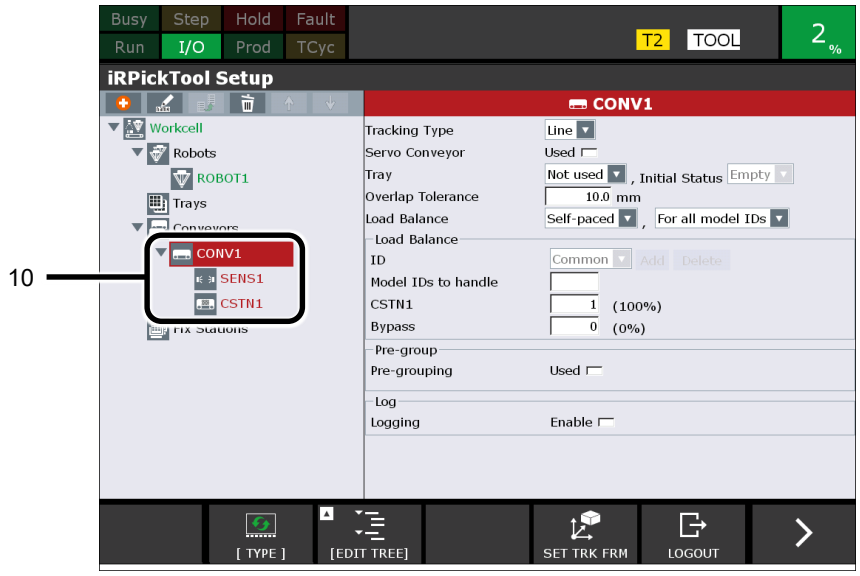


9 Input the name of the infeed conveyor in [Name].
We recommend using the name that is displayed as the initial value just as it is.
Input using half-width alphanumeric characters and half-width katakana. The name must contain no spaces, not contain the symbol other than underscore, and start with a letter.

10 Press F4 [OK].
The object is created.

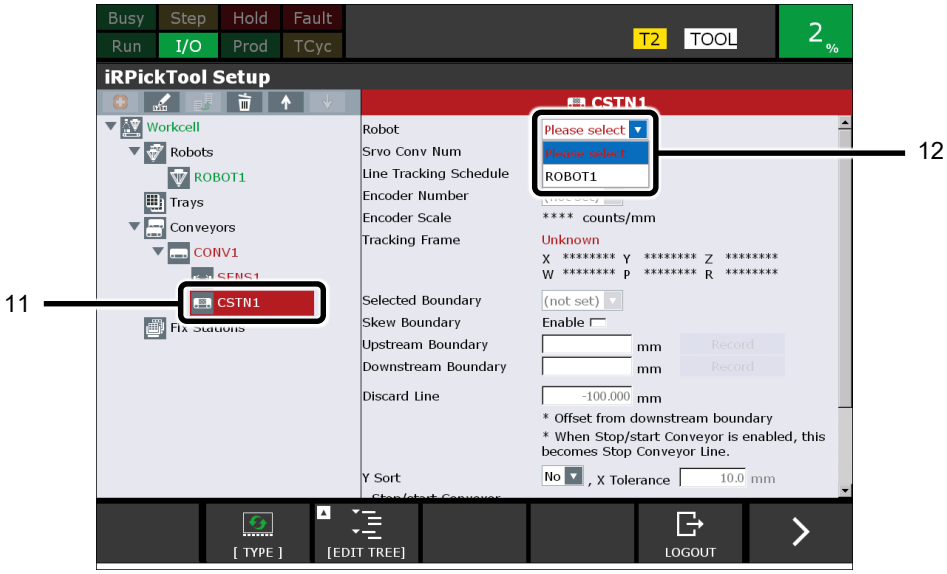


When a new conveyor is created, a sensor and conveyor station will be automatically created.



11 Select the conveyor station [CSTN1] that was automatically created in step 10.

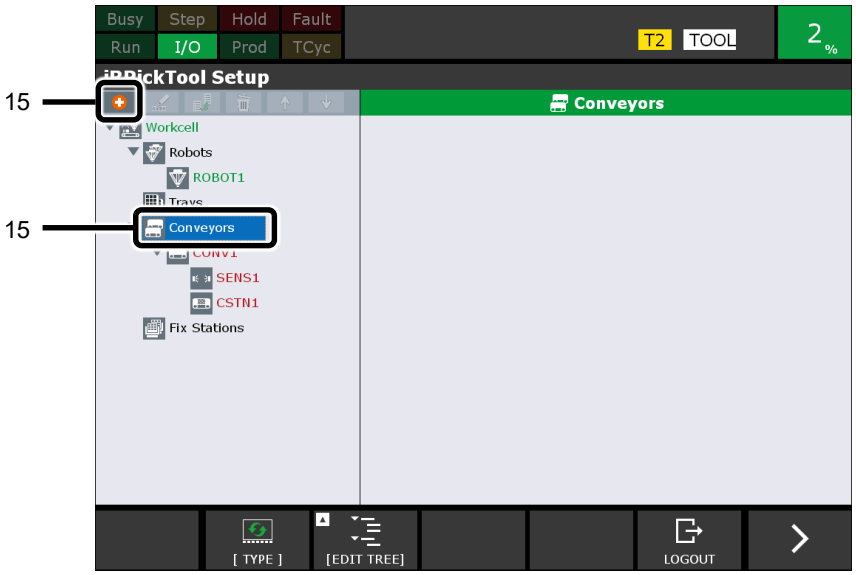
12 For [Robot], select a robot from the menu.
Select the robot name that was input in step 4.



13 Select [Line Tracking Schedule] from the menu. Select '1.'

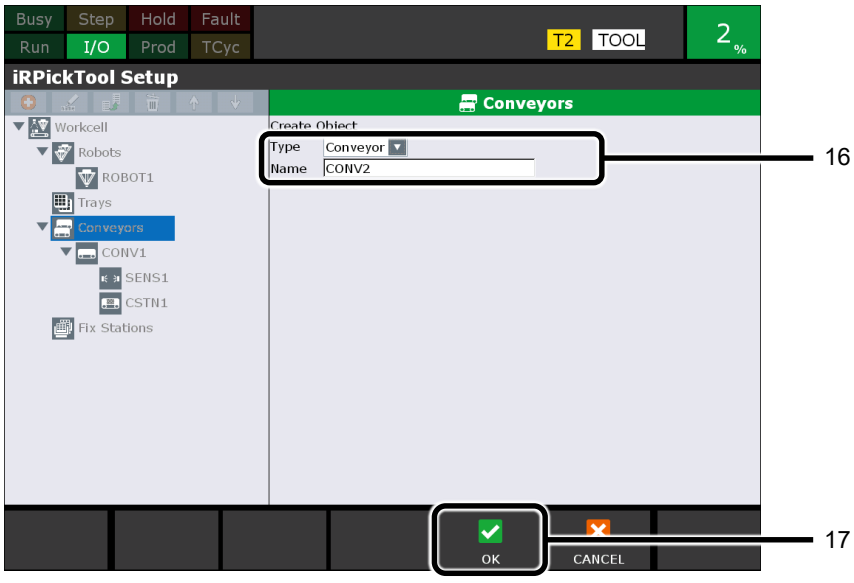
14 Select [Encoder Number] from the menu. Select '1.'

15 Next, create an object for the outfeed conveyor. Select [Conveyors], then press the [CREATE] button above the tree.

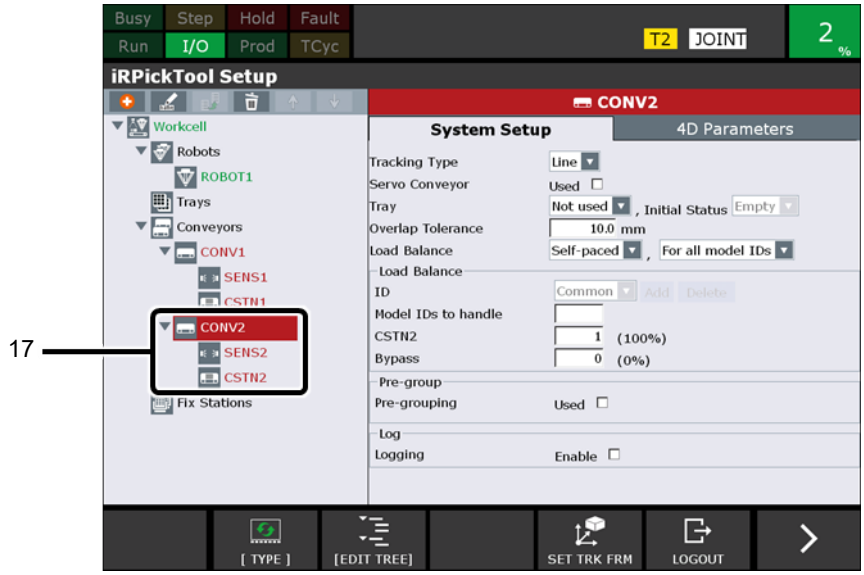


16 Input the name of the outfeed conveyor in [Name].
We recommend using the name that is displayed as the initial value just as it is.
Input using half-width alphanumeric characters and half-width katakana. The name must contain no spaces, not contain the symbol other than underscore, and start with a letter.

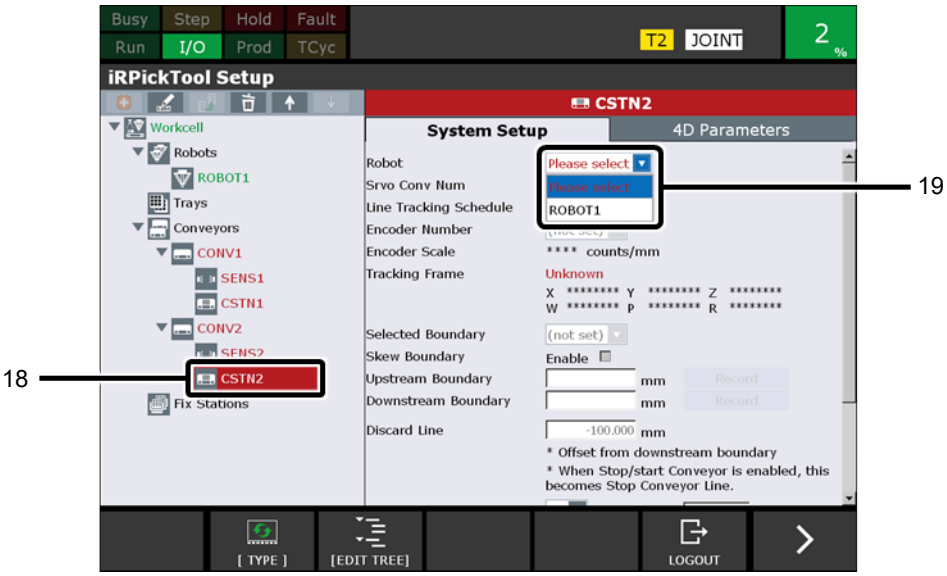
17 Press F4 [OK].
The object is created.



When a new conveyor is created, a sensor and conveyor station will be automatically created.

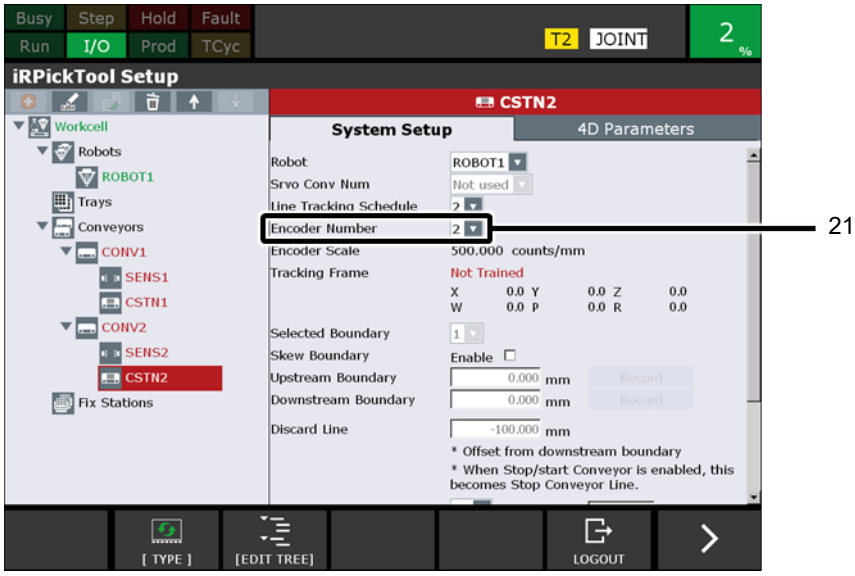


18 Select the conveyor station [CSTN2] that was automatically created in step 17.
19 For [Robot], select a robot from the menu.
Select the robot name that was input in step 4.



20 Select [Line Tracking Schedule] from the menu. Select '2.'

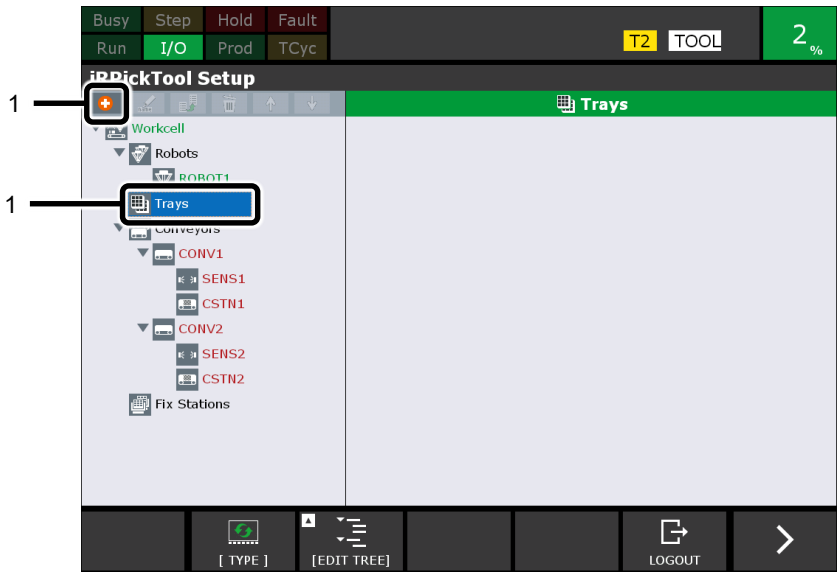
21 Select [Encoder Number] from the menu. Select '2.'



3.2.2 Setting Up Trays

Set up the trays that are used on the outfeed conveyor.

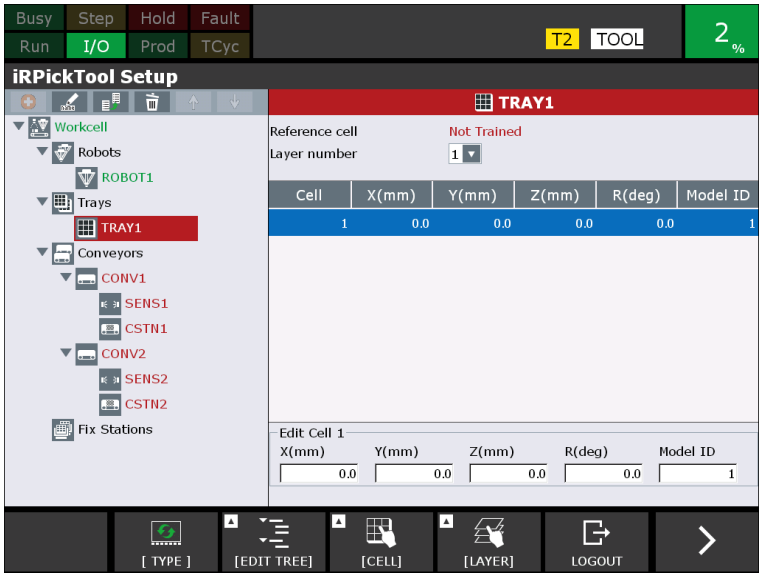
1 Select [Trays], then press the [CREATE] button above the tree.



2 Enter the name of the tray in [Name].
We recommend using the name that is displayed as the initial value just as it is.
Input using half-width alphanumeric characters and half-width katakana. The name must contain no spaces, not contain the symbol other than underscore, and start with a letter.

3 Press F4 [OK].
The object is created.

A screen like the following one will appear.



4 Decide the tray frame.
The origin will be an arbitrary position.
Aligning the flow direction of the conveyor with the X direction of the tray frame will make things easier to understand. (Other directions are also OK.)

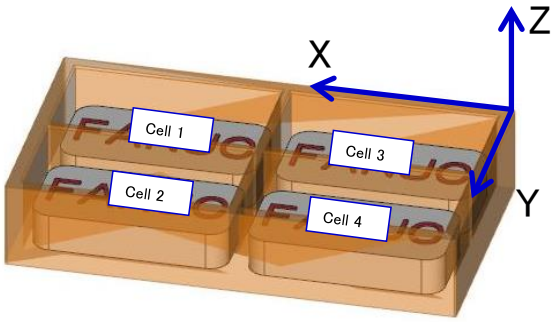
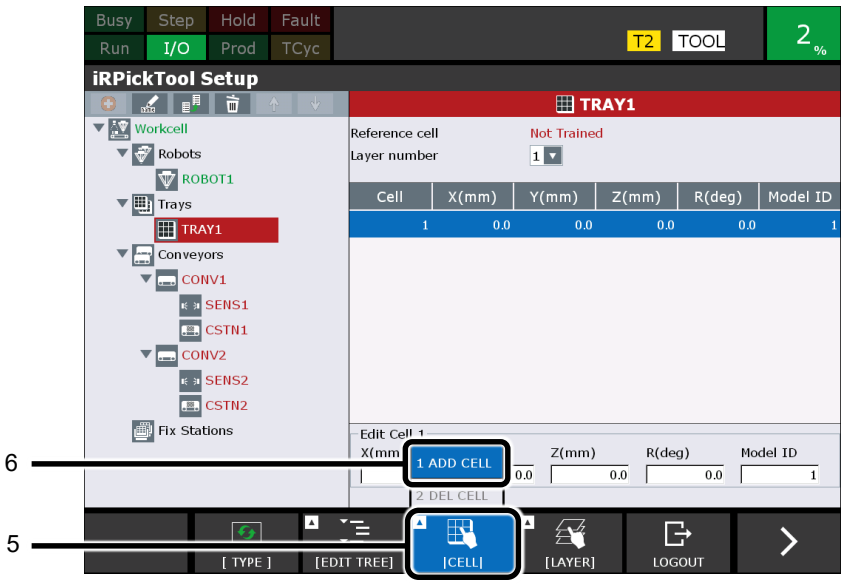


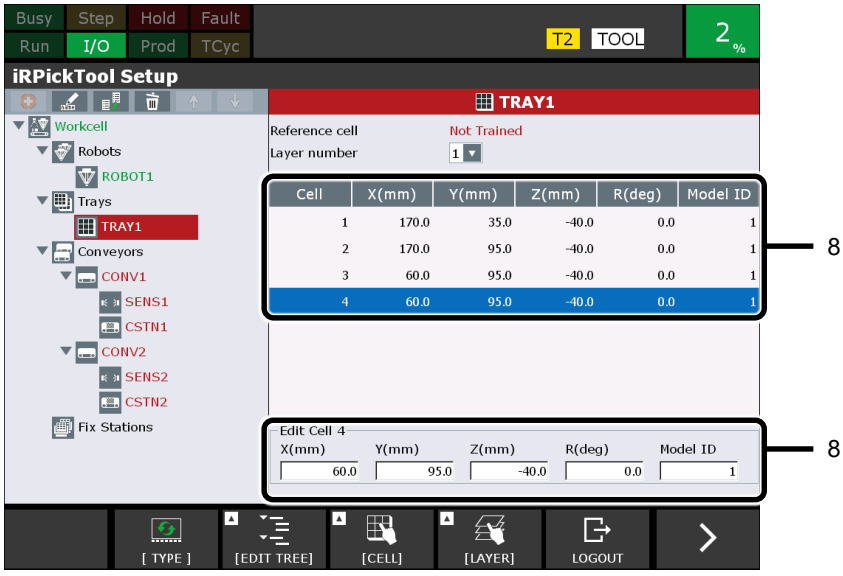
Fig. 3.2.2(a) Deciding the origin of a tray

5 Press F3 [CELL].
6 Select [ADD CELL] from the menu.



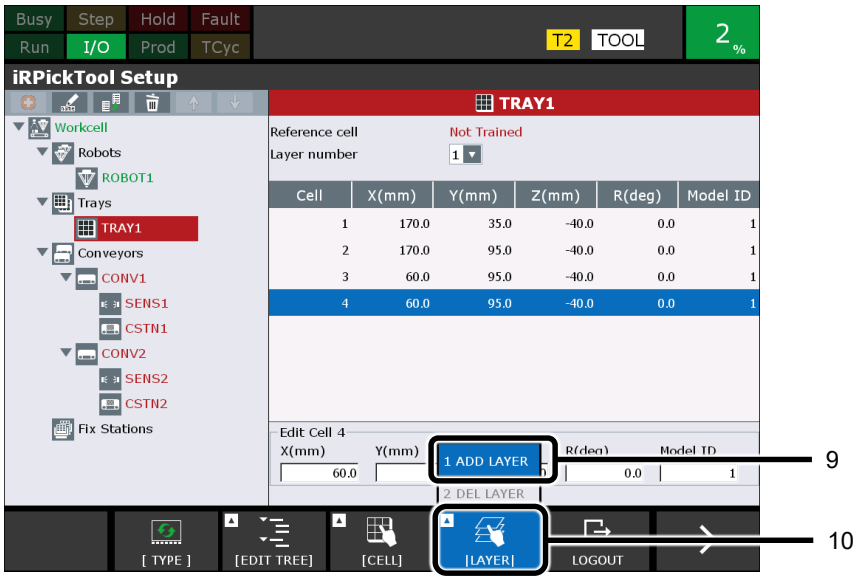
7 Add the cells for the first stage.

8 Enter the coordinates of each cell in the cell edit screen.
Select the row of the cell that you want to edit, and enter the coordinates in the frame at the bottom of the screen.
Enter the coordinates of a cell in the tray frame coordinate system. For details, refer to Section 6.3, “SETUP OF A TRAY”.

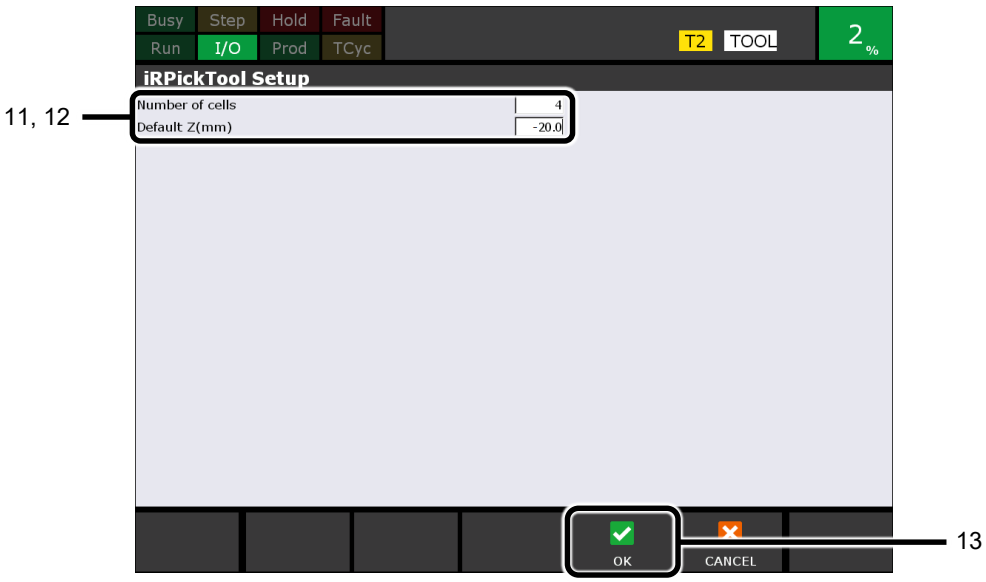


If trays are double-stacked, perform the following operation.

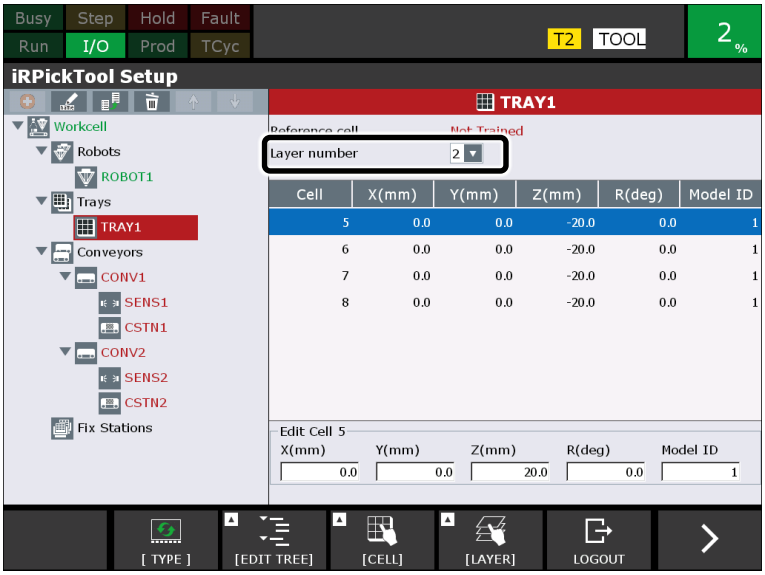
9 Press F4 [LAYER].
10 Select [ADD LAYER] from the pull-down menu.



11 Enter the number of cells in the second stage in [Number of cells].
12 Enter a Z value (mm) in [Default Z(mm)].
13 Press F4 [OK].

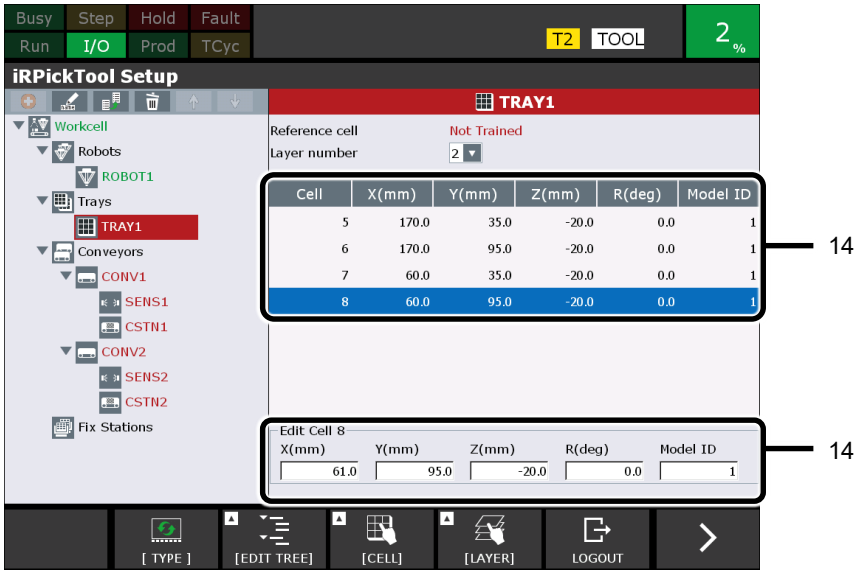


A screen like the following one will appear.
Layer '2' is created in [Layer number].



14 Enter the coordinates of each cell in Layer 2 in the cell edit screen.
Select the row of the cell that you want to edit, and enter the coordinates in the frame at the bottom of the screen.

* The reference cell can remain untrained at this point. It is automatically taught in Subsection 3.2.10, "Setting Up a Reference Position and Teaching a Robot Position".



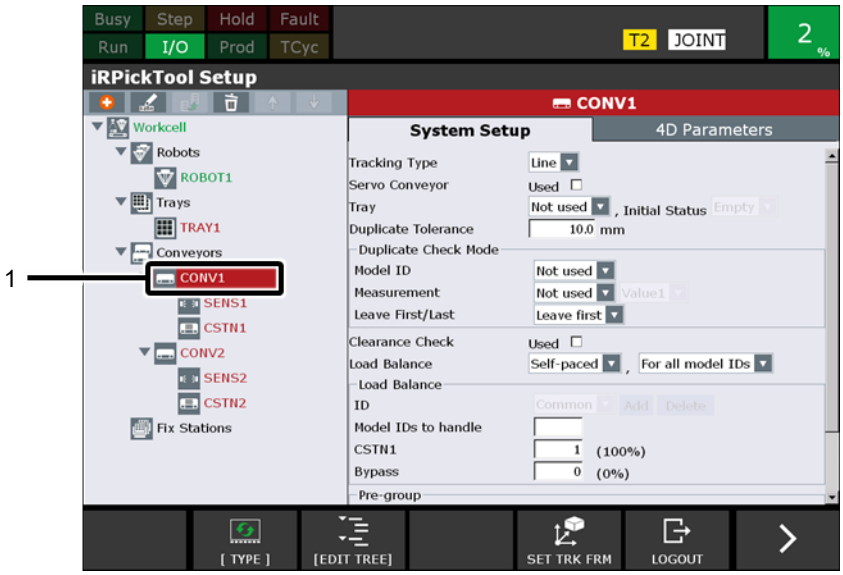
3.2.3 Setting Up a Conveyor

Set up a conveyor. Depending on whether trays are used or not, the procedure will be different.

3.2.3.1 Setting up an infeed conveyor

Set up an infeed conveyor. This is the setup when trays are not used.

1 Select the conveyor [CONV1].



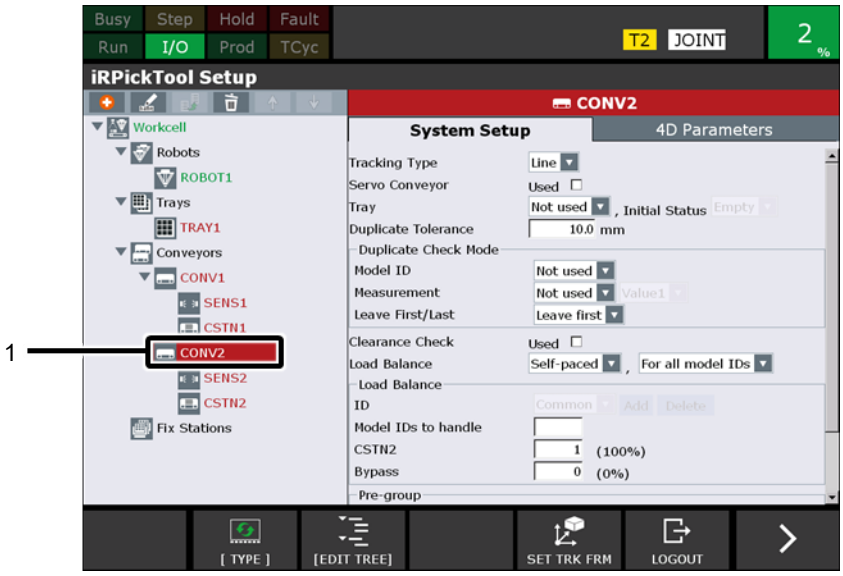
The conveyor setup screen will appear.
For conveyor setup, the following data that appears as the initial values can be used as it is.

[Tracking Type] : Line
[Servo Conveyor] : Do not check [Used]
[Tray] : Not used
[Duplicate Tolerance] : 10.0 mm
[Load Balance] : Self-paced, For all model IDs.

3.2.3.2 Setting up an outfeed conveyor

Set up an outfeed conveyor. This is the setup when trays are used.

1 Select the conveyor [CONV2].



The conveyor setup screen will appear.

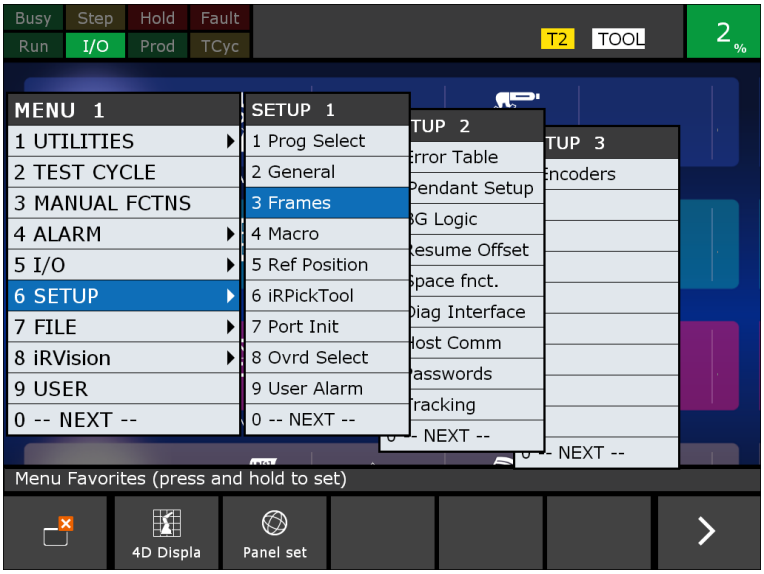
2 Select [TRAY1] in [Tray] and [Empty] in [Initial Status].
For other settings for conveyors, the following data that appears as the initial values can be used as it is.

[Tracking Type] : Line
[Servo Conveyor] : Do not check [Used]
[Duplicate Tolerance] : 10.0 mm (default)
[Load Balance] : Self-paced, For all model IDs

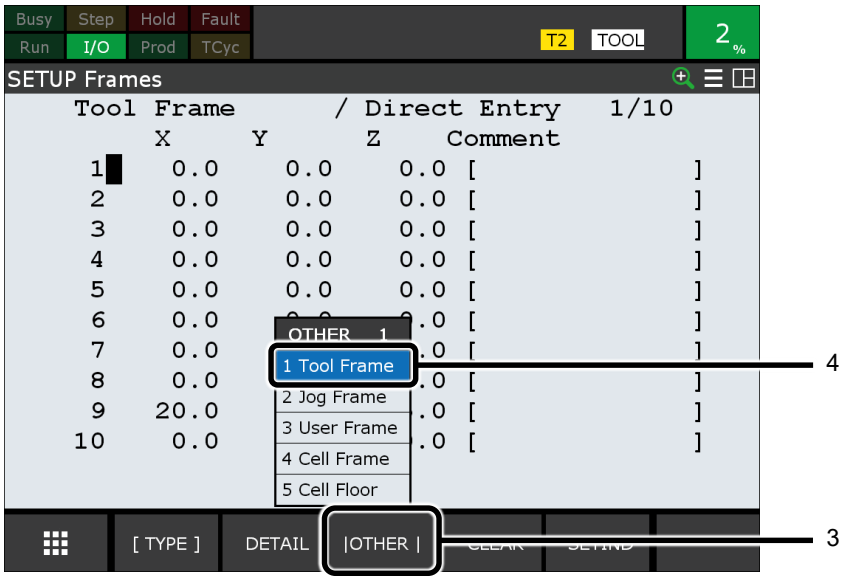
3.2.4 Setting Up a Tracking Frame

Set up a tracking frame.

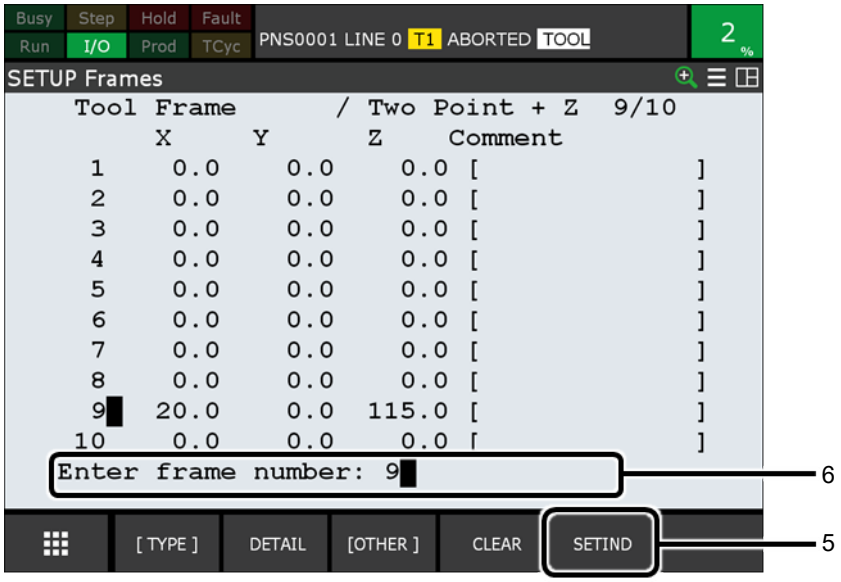
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SETUP] → [Frames] from the menu.



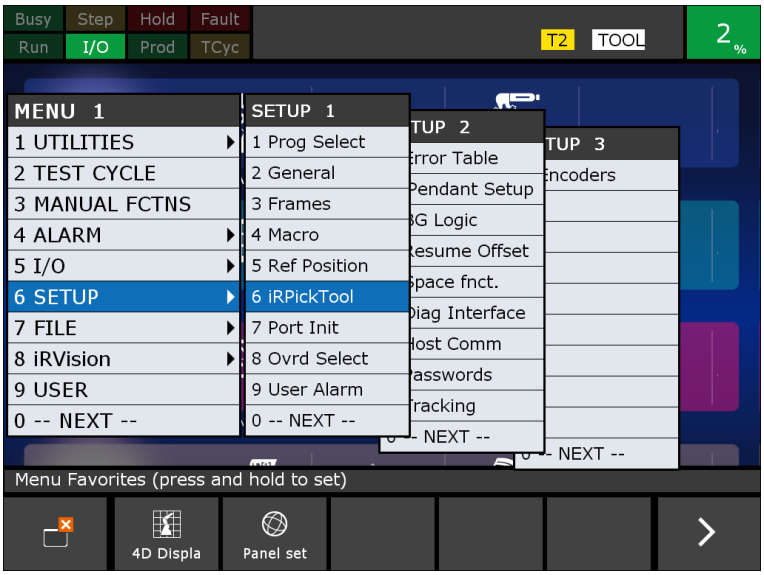
3 Press F3 [OTHER].
4 Select [Tool Frame] from the menu.
The list screen for tool frames will appear.



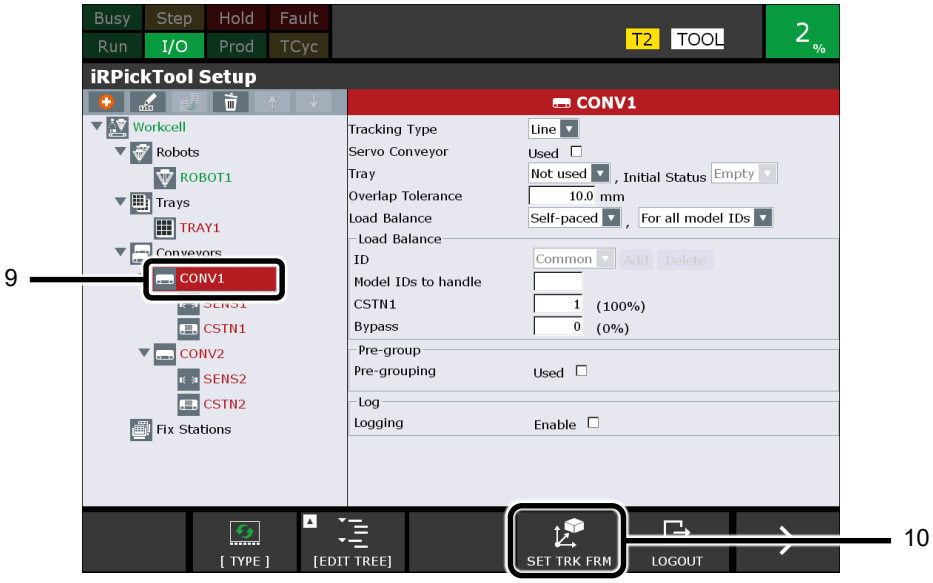
5 Press F5 [SETIND].
6 Enter the frame number that has been set for the pointer tool.
This procedure is explained using a case where Tool 9 is used as an example.



7 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
8 Select [SETUP] → [*i*RPickTool] from the menu.



9 First, set up the tracking frame of the infeed conveyor.
Select the conveyor [CONV1].
10 Press F4 [SET TRK FRM].



A screen like the following one will appear.

11 Set a calibration grid to the upstream side of the conveyor, in accordance with the procedure shown on the screen.
Fix it in place with tape, etc. to avoid misalignment.

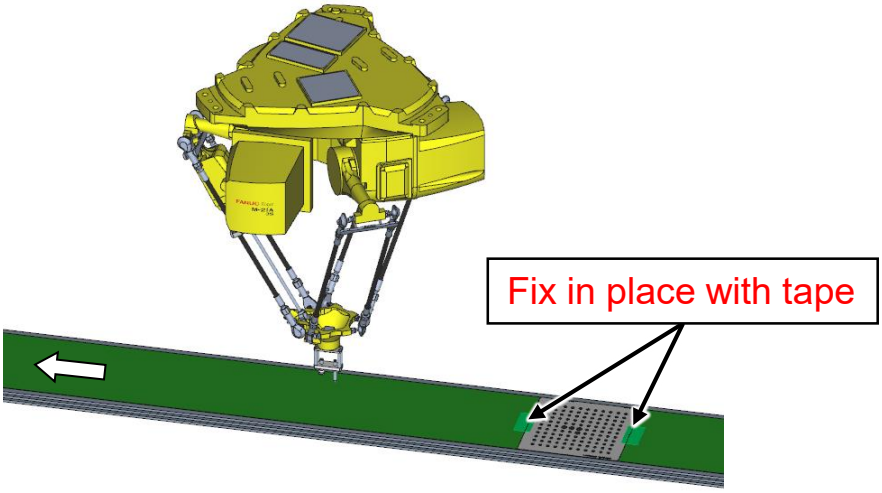


Fig. 3.2.4 (a) Setting a calibration grid to the upstream side

12 Set the flow direction of the conveyor and the X direction of the calibration grid to be in the same direction.

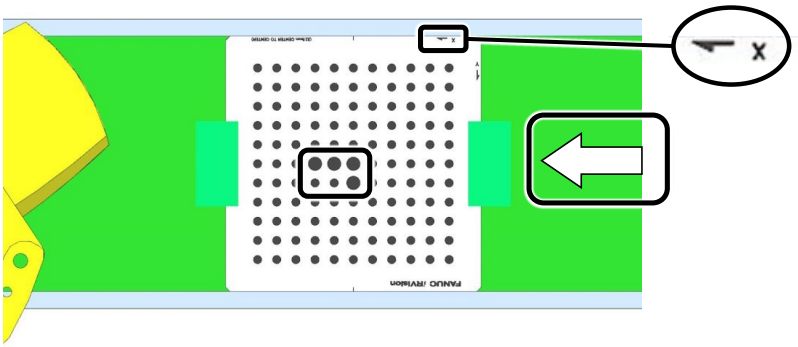
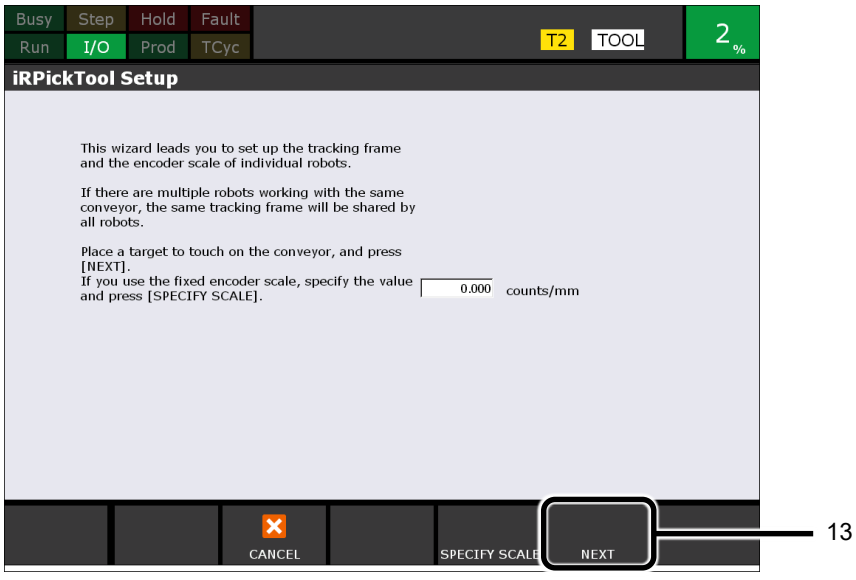
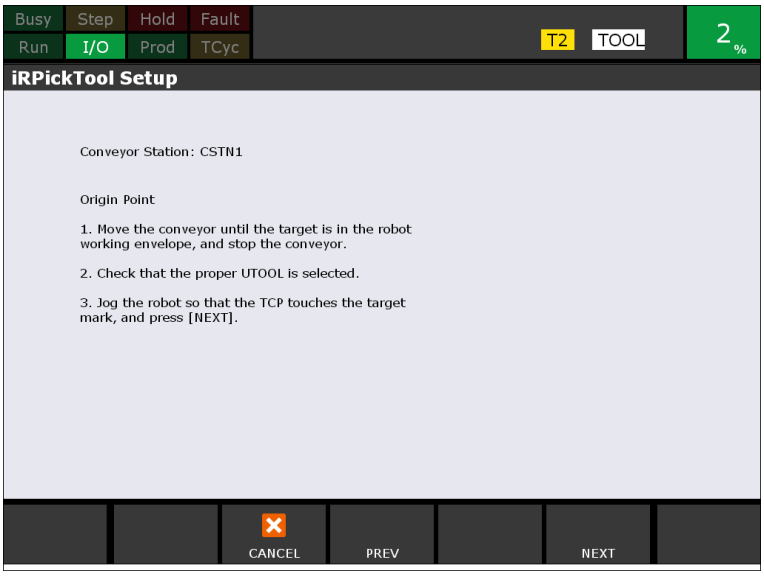


Fig. 3.2.4 (b) Align the setting direction

13 Press F5 [NEXT].



A screen like the following one will appear.



14 Move the conveyor. When the calibration grid comes to the upstream within the operating range of the robot, stop.

15 Touch up the origin of the calibration grid.

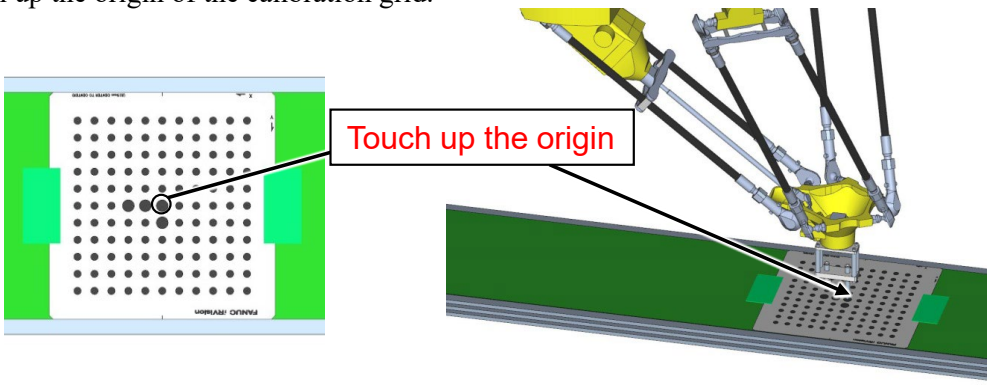
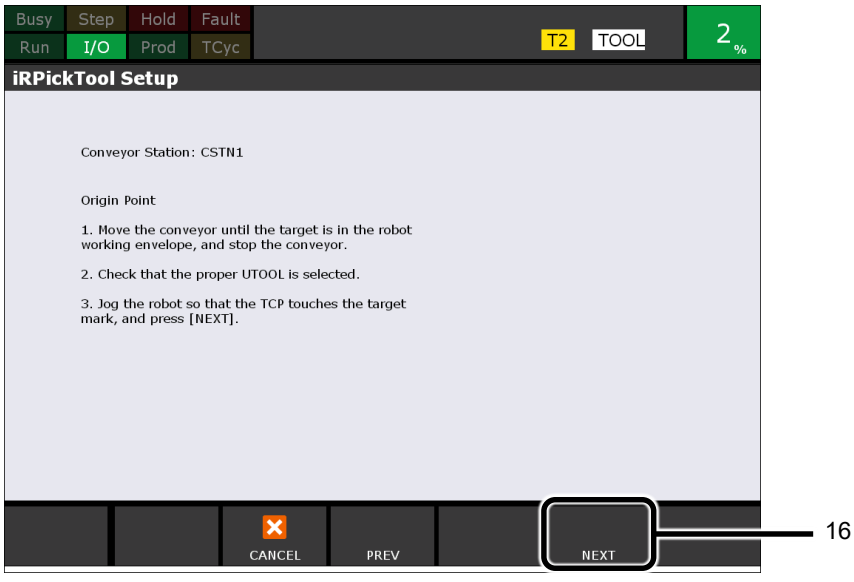


Fig. 3.2.4 (c) Touching up the origin of the calibration grid.

16 Press F5 [NEXT].



A screen like the following one will appear.

17 Move the robot skyward.

18 Move the conveyor. When the calibration grid comes to the downstream within the operating range of the robot, stop.

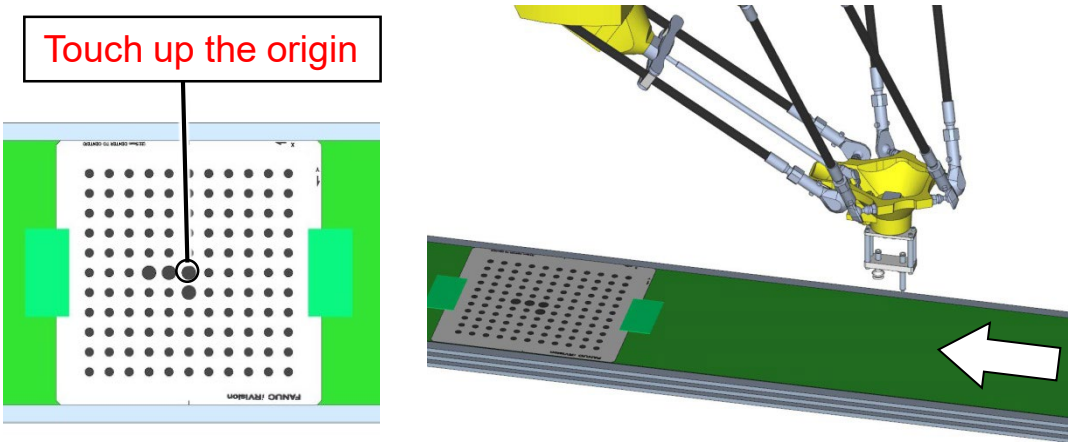


Fig. 3.2.4 (d) Stopping the conveyor

19 Touch up the origin of the calibration grid.

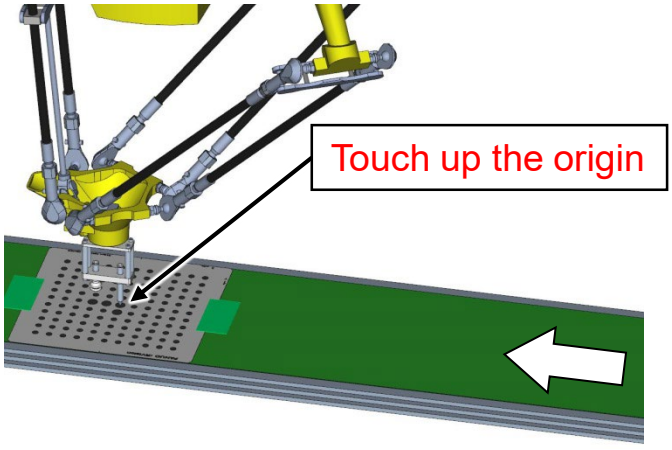
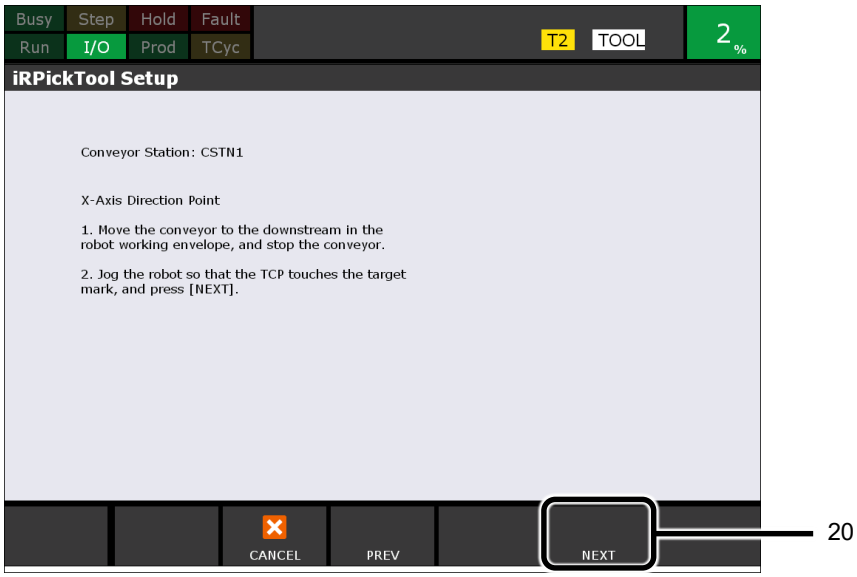
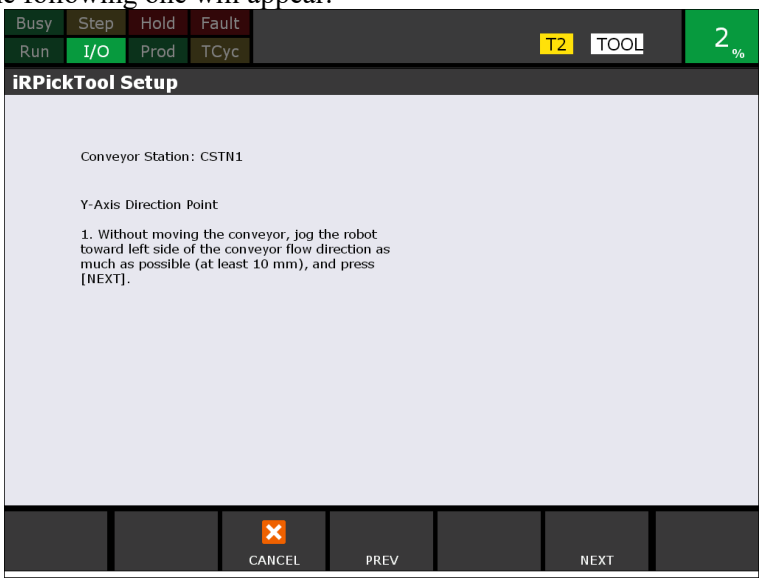


Fig. 3.2.4 (e) Touching up the origin of the calibration grid.

20 Press F5 [NEXT].



A screen like the following one will appear.



21 Touch up the Y direction of the calibration grid.
Do not move the conveyor at this time.
It is not the Y direction of the robot. It is on the left side relative to the direction of movement of the conveyor.

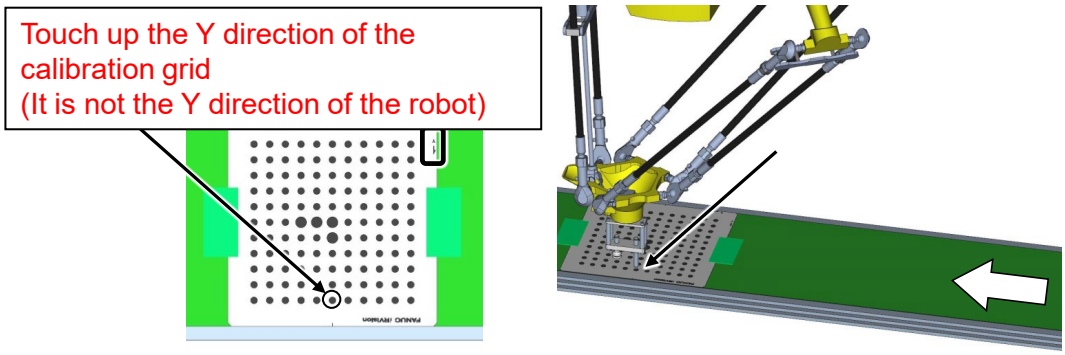
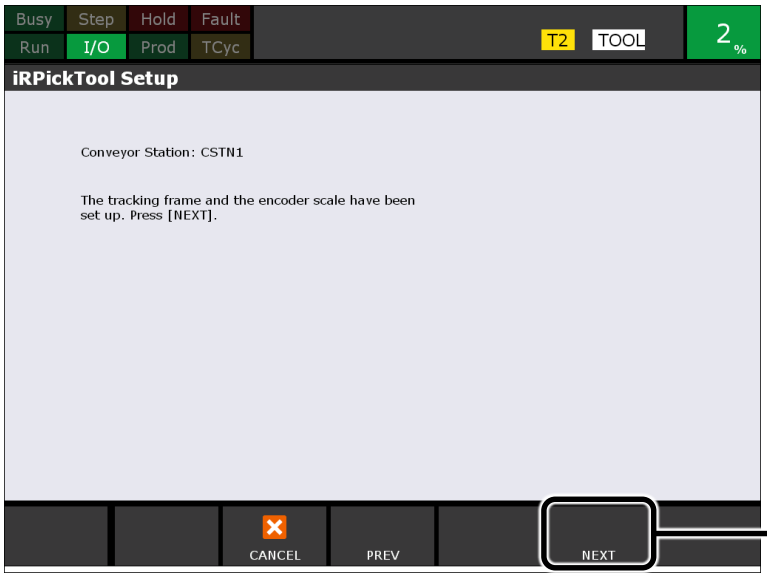


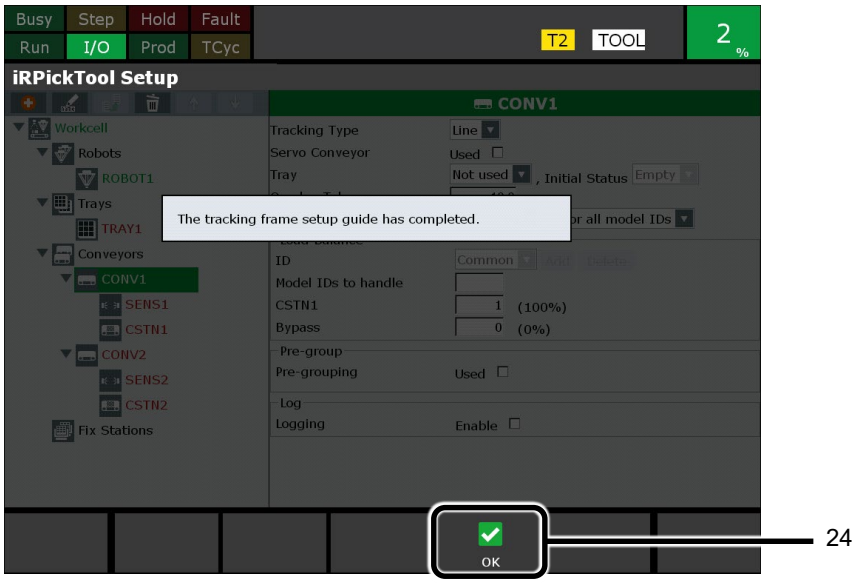
Fig. 3.2.4 (f) Touching up the Y direction of the calibration grid.

22 Press F5 [NEXT].

23 Press F5 [NEXT].



24 Press F4 [OK].



The tracking frame of the infeed conveyor has been set up.

25 Next, set up the tracking frame of the outfeed conveyor.
Select the conveyor [CONV2].
26 Repeat the steps 10 to 24 to set up the tracking frame of the outfeed conveyor.

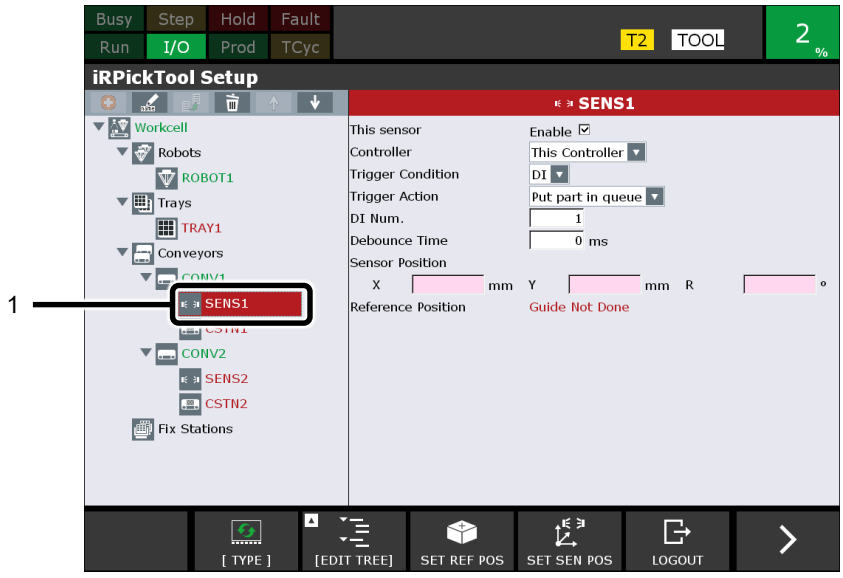
3.2.5 Setting Up a Sensor

Set up a sensor.

3.2.5.1 Setting up an infeed conveyor sensor

Set up an infeed conveyor sensor. The setup is for the case of [Distance] + [Find part by vision].

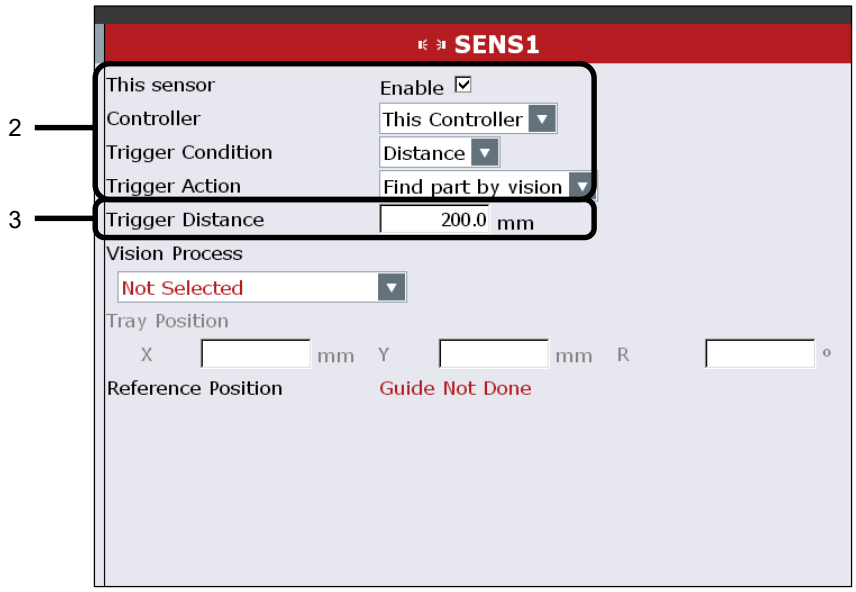
1 Select the sensor [SENS1].



2 Set the following items.
[This sensor] : Check [Enable]
[Controller] : [This Controller]
[Trigger Condition] : [Distance]
[Trigger Action] : [Find part by vision]

When [Trigger Condition] and [Trigger Action] are changed, the setup items change.

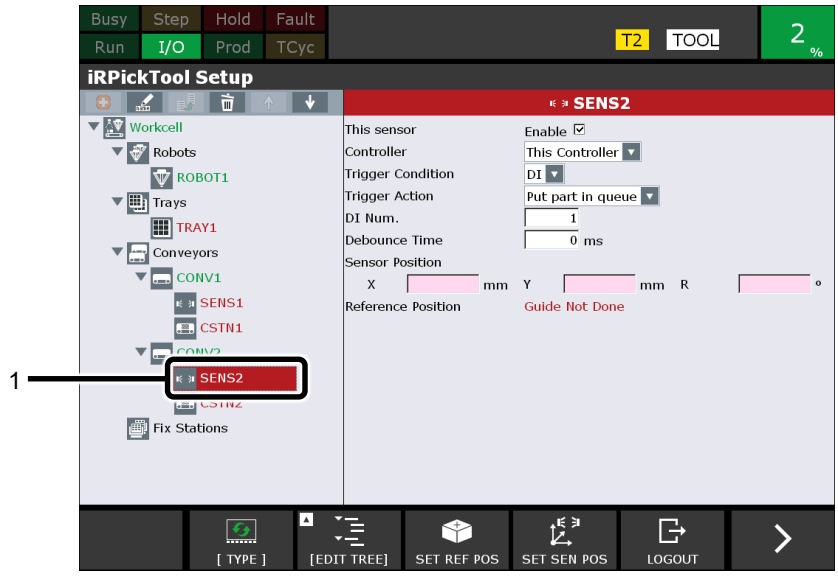
3 Set the following items.
[Trigger Distance] : Set how many mm the conveyor advances each time for vision detection to be performed.
*It is recommended that the trigger distance be set such that a workpiece can be detected twice in the camera FOV.
[Vision process] : You do not have to set this up at this stage.
*Set this up immediately before setting up reference positions.
For setup, refer to Subsection 3.2.10.1, “Setting up a reference position for the infeed conveyor, and teaching a robot position”.



3.2.5.2 Setting up an outfeed conveyor sensor (In the case of [DI] / [RI] + [Put part in queue])

Set up an outfeed conveyor sensor. The setup is for a sensor in the case of [DI] / [RI] + [Put part in queue]. In the case of [HDI] + [Put part in queue], skip this section and go to the next section, Subsection 3.2.5.3, “Setting up an outfeed conveyor sensor (In the case of [HDI] + [Put part in queue])”.

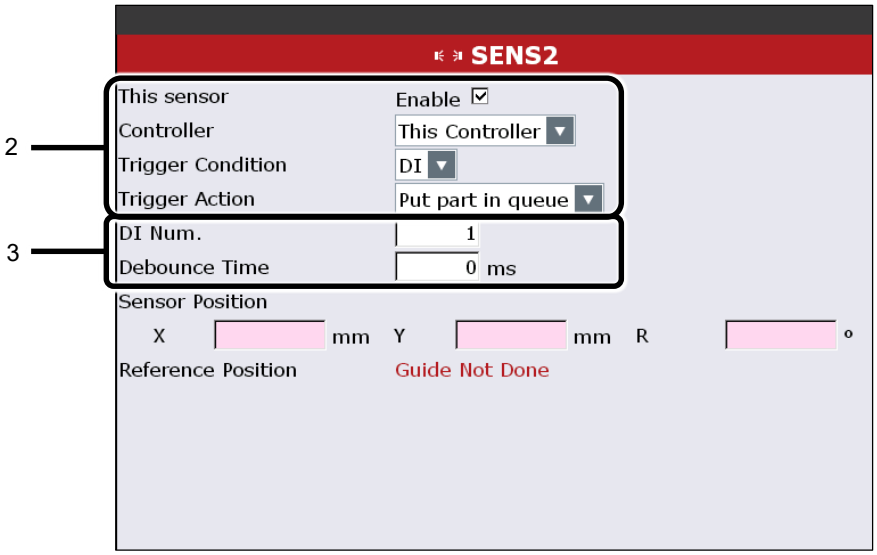
1 Select the sensor [SENS2].



2 Set the following items.
[This sensor] : Check [Enable]
[Controller] : [This Controller]
[Trigger Condition] : [DI]
[Trigger Action] : [Put part in queue]

When [Trigger Condition] and [Trigger Action] are changed, the setup items change.

3 Set the following items.
[DI Num.] : Enter the DI number to which sensor input has been assigned.
[Debounce Time] : Enter the time you want to disable input for (to prevent chattering)
[Sensor Position] : You do not have to set this at this point.
*This is set in the setup of the sensor position. Refer to Subsection 3.2.6, “Setting Up a Sensor Position” for the setup.

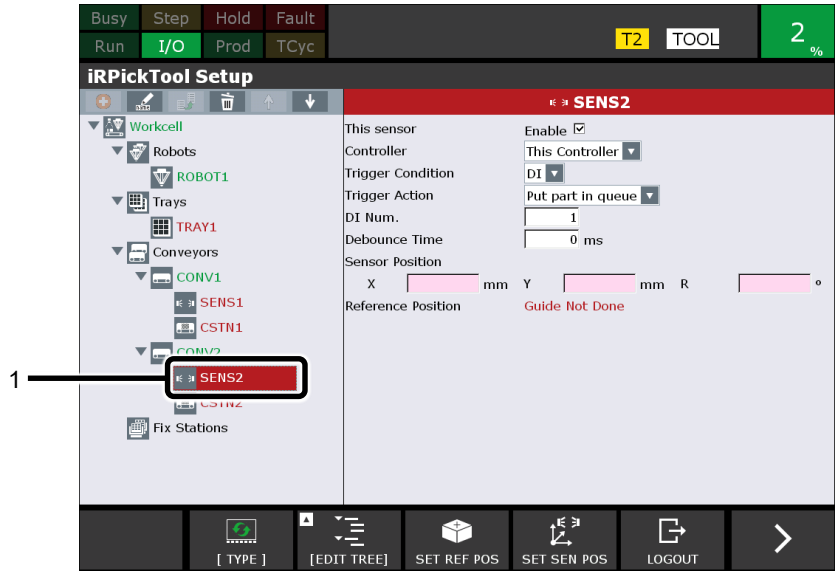


Skip the next section 3.2.5.3, “Setting up an outfeed conveyor sensor (In the case of [HDI] + [Put part in queue])” , and go to the section 3.2.6, “Setting Up a Sensor Position”.

3.2.5.3 Setting up an outfeed conveyor sensor (In the case of [HDI] + [Put part in queue])

Set up an outfeed conveyor sensor. The setup is for a sensor in the case of [HDI] + [Put part in queue]. In the case of [DI] / [RI] + [Put part in queue], do not refer to this section, but refer to the previous section 3.2.5.2, “Setting up an outfeed conveyor sensor (In the case of [DI] / [RI] + [Put part in queue])” .

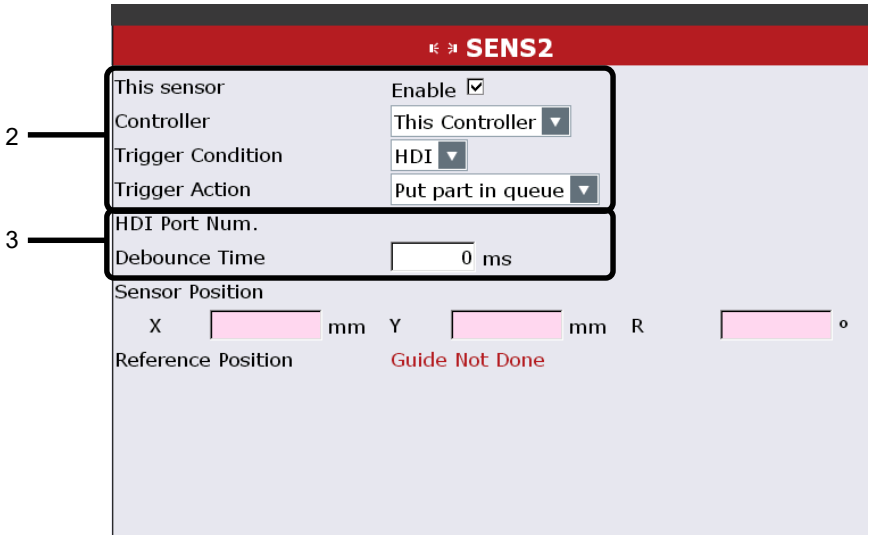
1 Select the sensor [SENS2].



2 Set the following items.
[This sensor] : Check [Enable]
[Controller] : [This Controller]
[Trigger Condition] : [HDI]
[Trigger Action] : [Put part in queue]

When [Trigger Condition] and [Trigger Action] are changed, the setup items change.

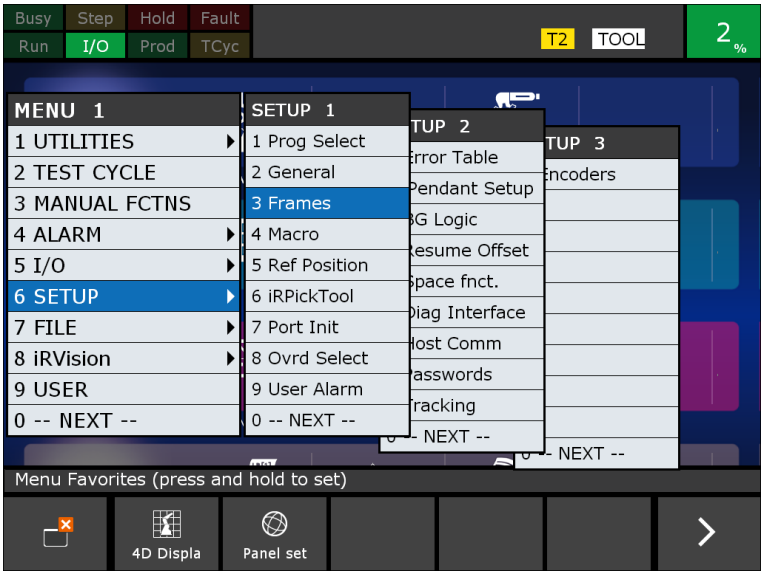
3 Set the following items.
[HDI Port Num.] : The number that was specified in the encoder setup will appear.
[Debounce Time] : Enter the time you want to disable input for (to prevent chattering)
[Sensor Position] : You do not have to set this in this procedure.
*This is set in the setup of the sensor position. Refer to Subsection 3.2.6, “Setting Up a Sensor Position” for the setup.



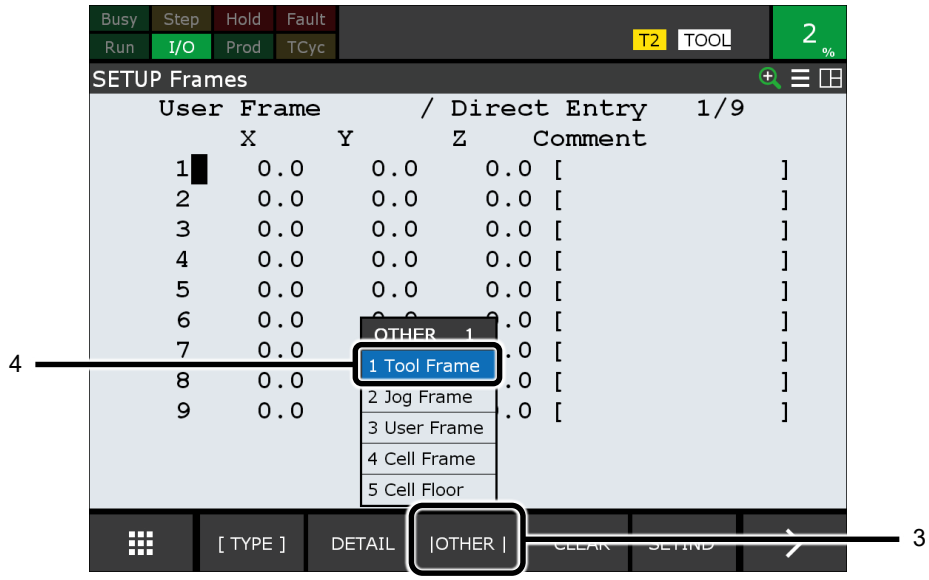
3.2.6 Setting Up a Sensor Position

Set up the sensor position of the outfeed conveyor. This is the setup when trays are used.

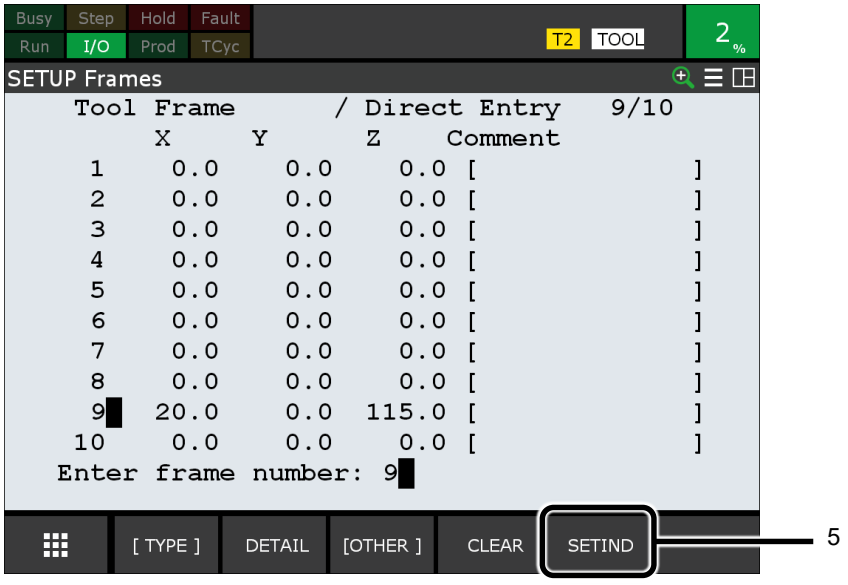
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SETUP] → [Frames] from the menu.



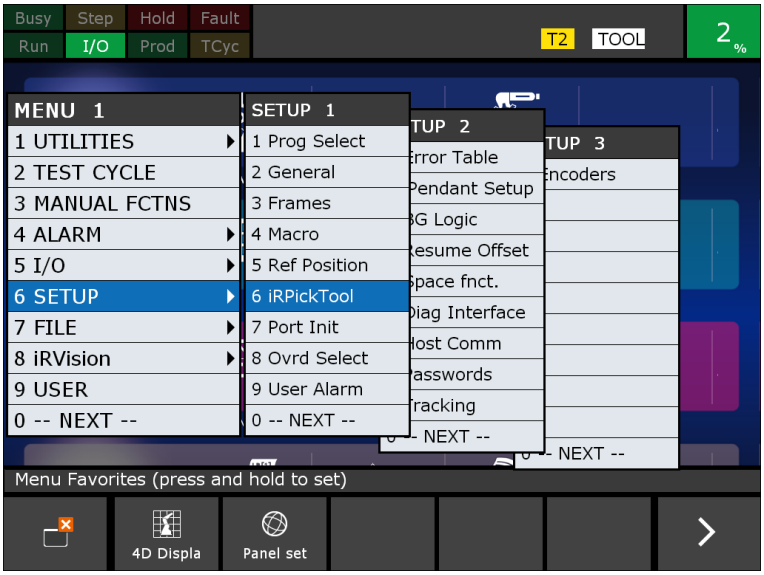
3 Press F3 [OTHER].
4 Select [Tool Frame] from the menu.



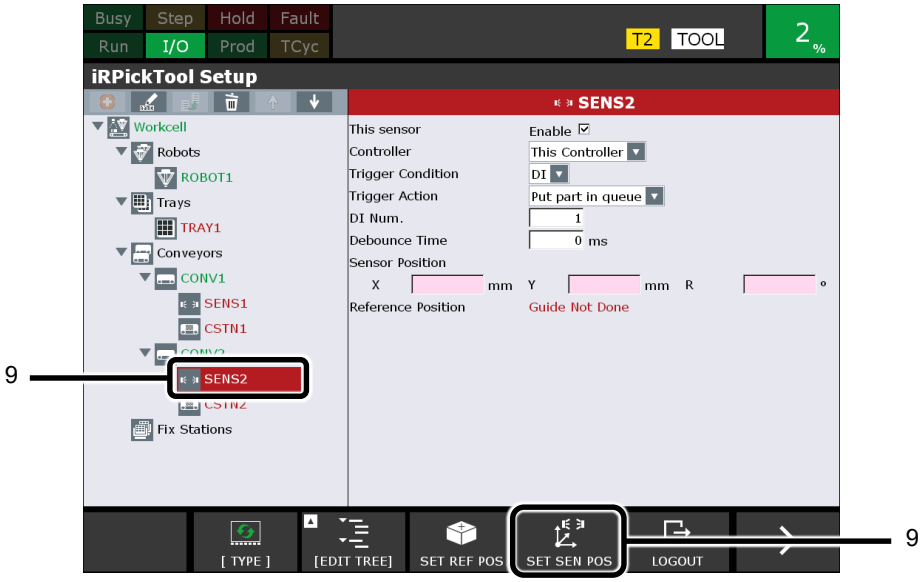
The list screen for tool frames will appear.
5 Press F5 [SETIND].
6 Enter the frame number that has been set for the pointer tool.
This procedure is explained using a case where Tool 9 is used as an example.



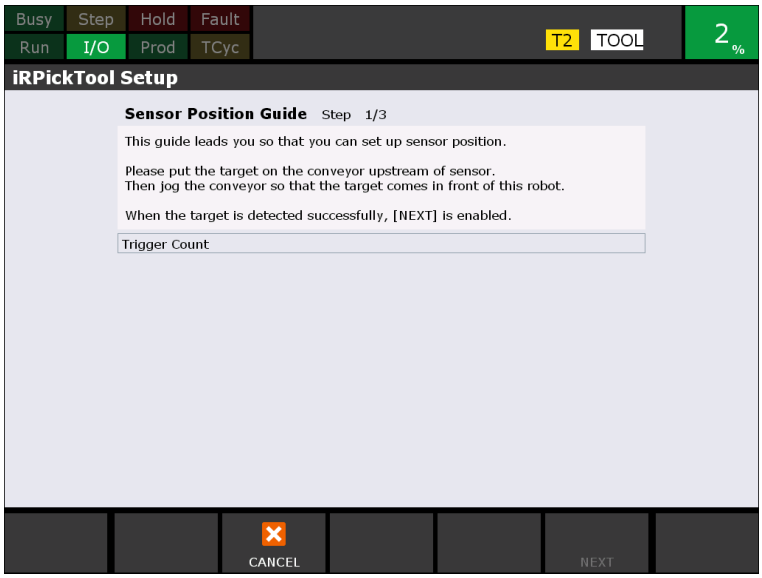
7 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
8 Select [SETUP] → [*i*RPickTool] from the menu.



9 Select [SENS2], then press F4 [SET SEN POS].



A screen like the following one will appear.



10 Set a tray further upstream than the sensor on the conveyor.

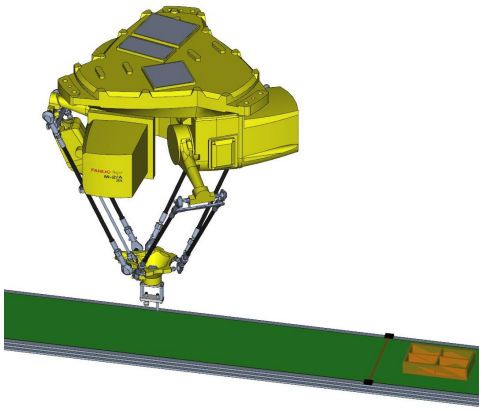


Fig. 3.2.6 (a) Setting a tray further upstream than the sensor

11 Move the conveyor. When the tray comes within the operating range of the robot, stop. The trigger count at the instant the sensor finds a part will appear.

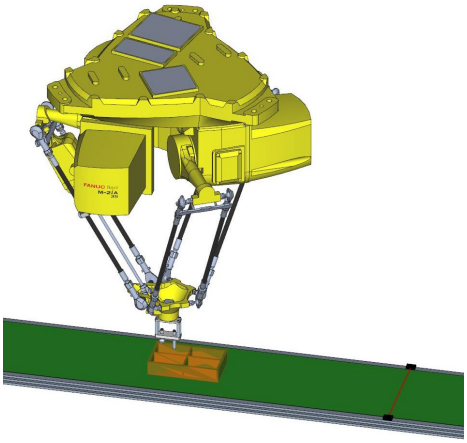
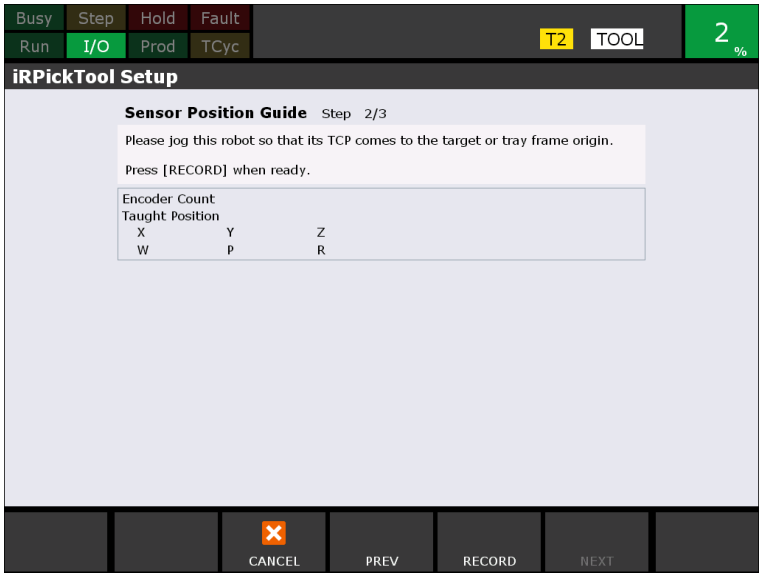


Fig. 3.2.6 (b) Stopping when the tray comes within the operating range of the robot

12 Press F5 [NEXT].

A screen like the following one will appear.



13 Touch up the origin of the tray.

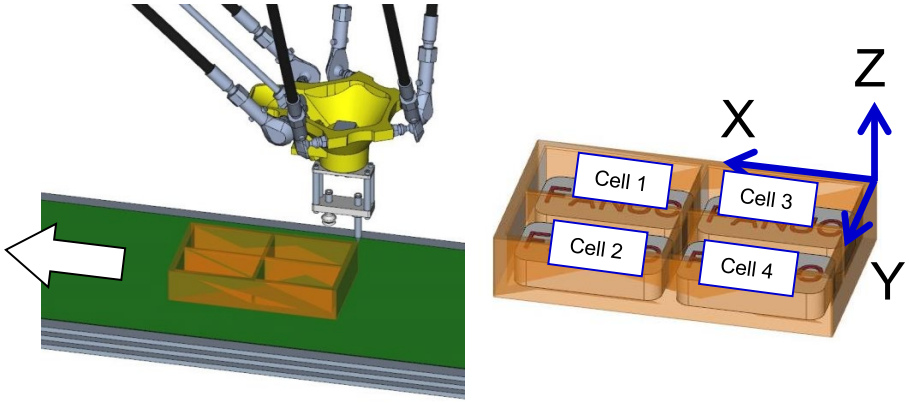
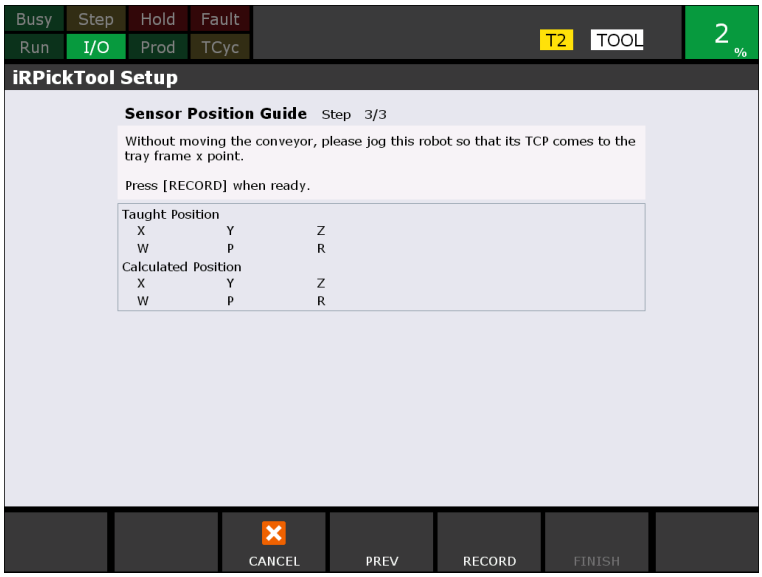


Fig. 3.2.6 (c) Touching up the origin of the tray

14 Press F4 [RECORD].
The current encoder count and taught position will appear.
15 Press F5 [NEXT].

A screen like the following one will appear.



16 Touch up the X-axis direction of the tray.

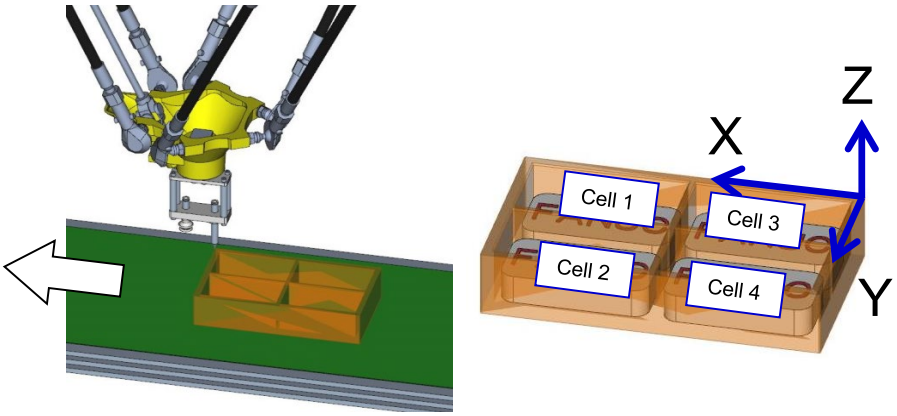


Fig. 3.2.6 (d) Touching up the X-axis direction of the tray

17 Press F4 [RECORD].
The current encoder count and taught position will appear.
18 Press F5 [FINISH].

Return to the original setup screen. The values will go in the X, Y and R of Sensor Position.

3.2.7 Vision Setup

Set up the infeed conveyor vision.

3.2.7.1 Camera calibration

Perform camera calibration.

1 Prepare a calibration grid that is appropriate for the field of view of the camera.
Check the grid spacing on the lower right.

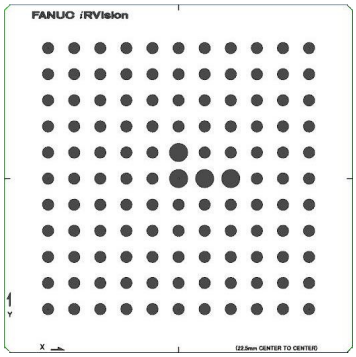


Fig. 3.2.7.1 (a) Calibration grid

> **⚠ CAUTION**
> The whole of the dot pattern does not have to appear in the image.
> It is OK if some dots to lie outside the image, so make it so that the dot pattern is distributed over the whole area in which the parts in the image are going to be found. If the dot pattern appears in only part of the area in which the parts in the image are going to be found, precise calibration cannot be performed.

> **NOTE**
> There are also metal plate types for the jig for the fixing calibration grid in place. Please prepare something that is appropriate for your environment. We offer some standard products.

2 Set the calibration grid so it is directly under the camera.
Fix it in place with tape, etc. to avoid misalignment.

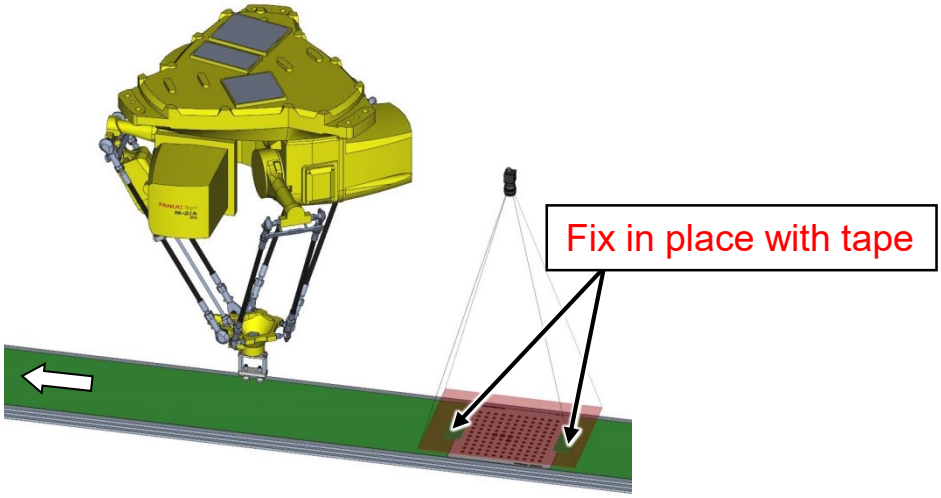


Fig. 3.2.7.1 (b) Setting a calibration grid

3 The direction of the calibration grid is arbitrary.
For example, in the case of line tracking, set the conveyor flow direction and the X direction of the calibration grid to be in the same direction.

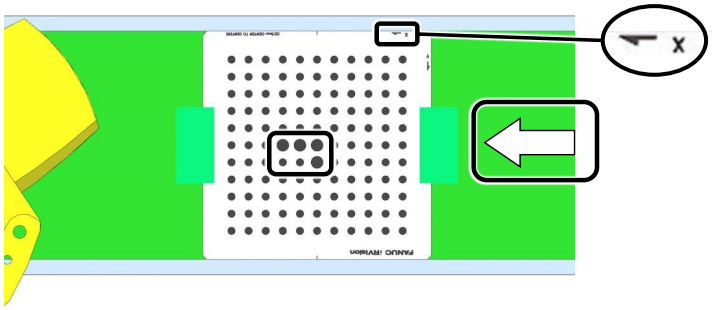
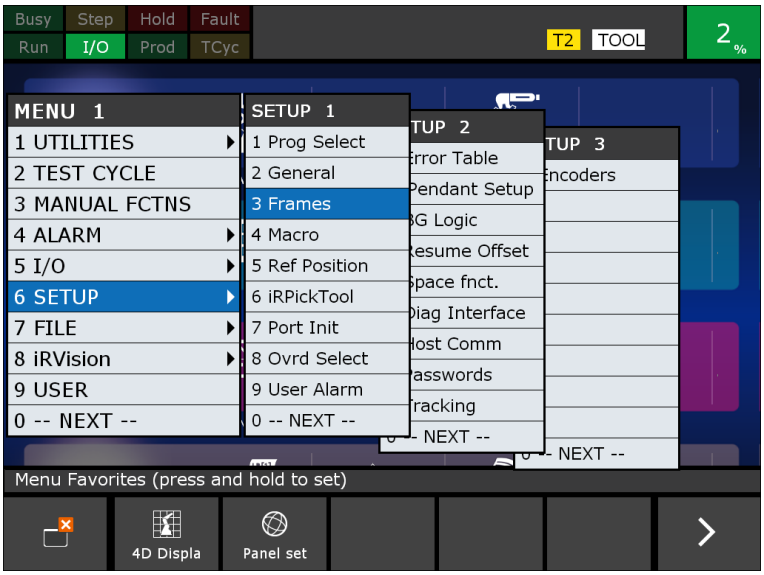
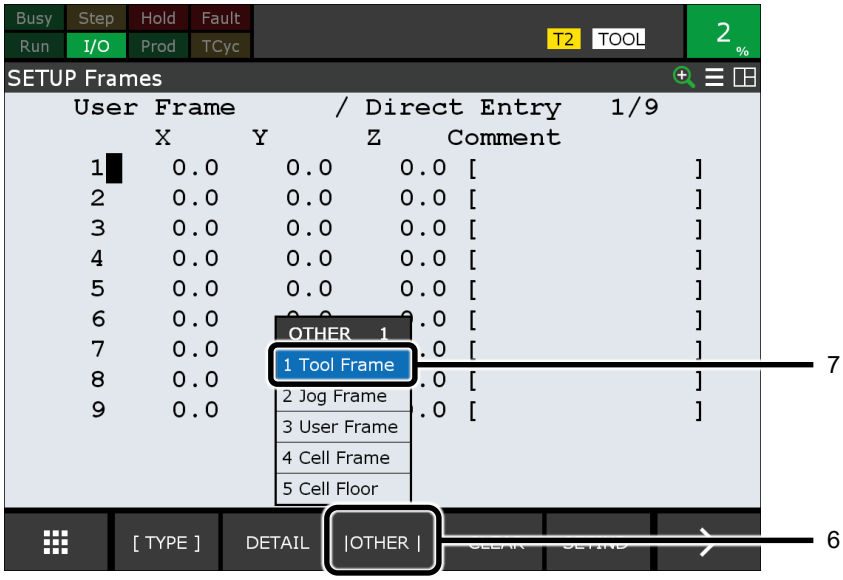


Fig. 3.2.7.1 (c) Align the setting direction

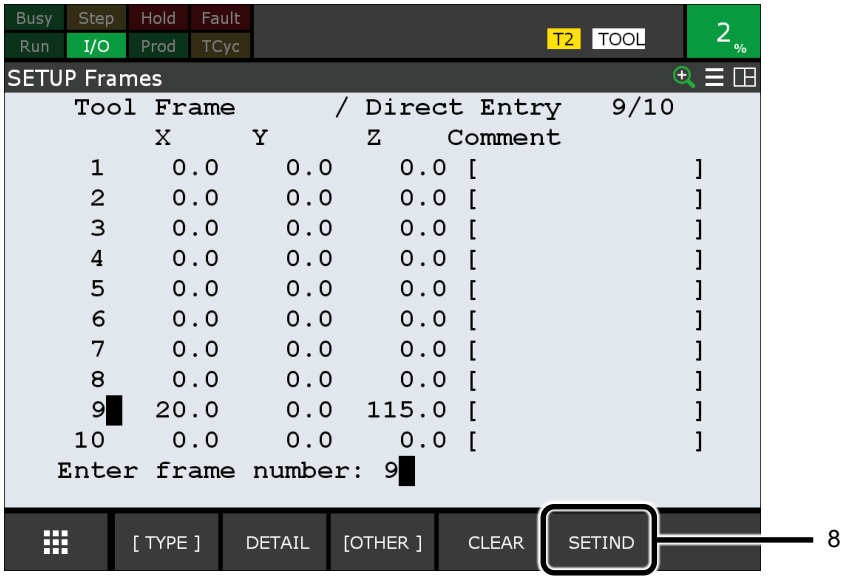
4 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
5 Select [SETUP] → [Frames] from the menu.



6 Press F3 [OTHER].
7 Select [Tool Frame] from the menu.
The list screen for tool frames will appear.

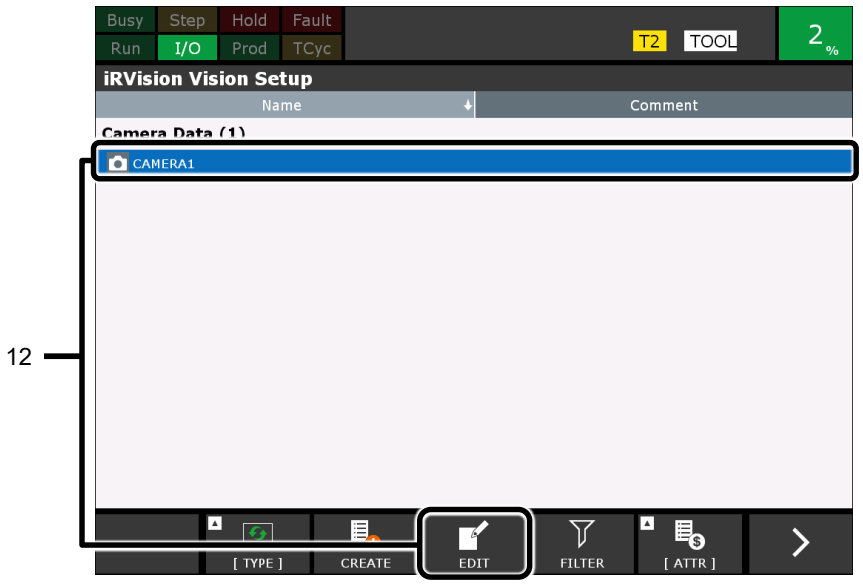


8 Press F5 [SETIND].
9 Enter the frame number that has been set for the pointer tool.
This procedure is explained using a case where Tool 9 is used as an example.



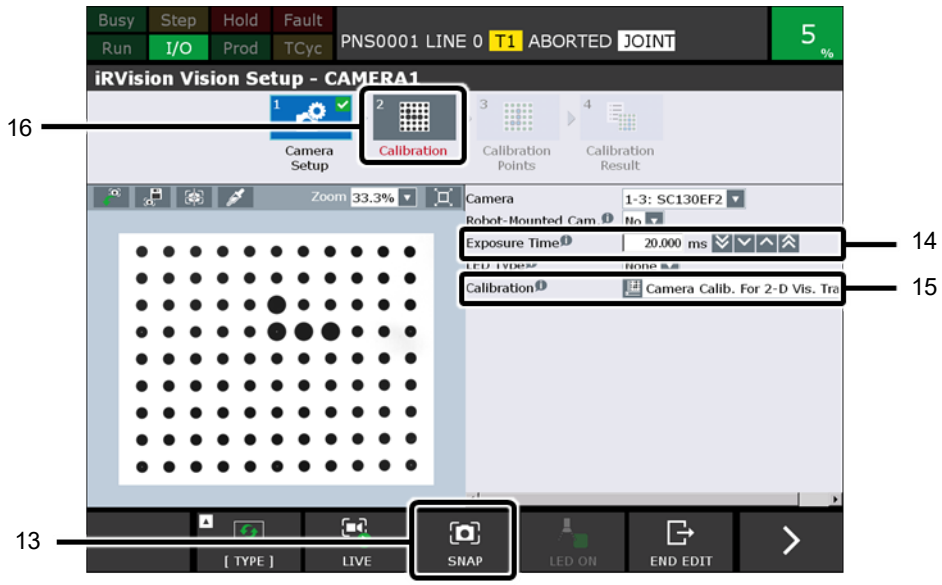
10 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
11 Select [*i*RVision] → [Vision Setup] from the menu. The [*i*RVision Vision Setup] screen will appear.

12 Select the camera data created in Subsection 3.1.2, “Checking the Connection of the Camera” and press F3 [EDIT].



The camera setup screen will appear.

13 Press F3 [SNAP] and check that a picture of the calibration grid has been taken within the frame on the left side of the screen.
14 Enter an [Exposure Time].
15 For [Calibration], select [Camera Calib. For Vis. Track] from the menu.
16 Press [Calibration] at the top of the screen.



The calibration setup screen will appear.

17 Select [Conveyor Name].
18 Enter the [Grid Spacing] of the dot pattern.
19 Select [Perspective] in [Projection].
[Perspective] has been selected as the initial value.

> ⚠ **CAUTION**
> [Orthogonal] can be selected only when the heights of the parts that are to be found are constant and the heights of the calibration grid surface and the upper surface of the parts that are to be found are the same.

20 In [Camera Distance], enter the focal distance of the lens that is being used.

> ⚠ **CAUTION**
> With regard to the calculation method for [Camera Distance], when [Auto] is selected, it is in principal possible that the focal distance might not be calculated accurately, because the calibration grid on the conveyor and the camera's imaging surface are nearly parallel.

NOTE

By placing the calibration grid at an inclination, an accurate [Camera Distance] can be calculated using [Auto].

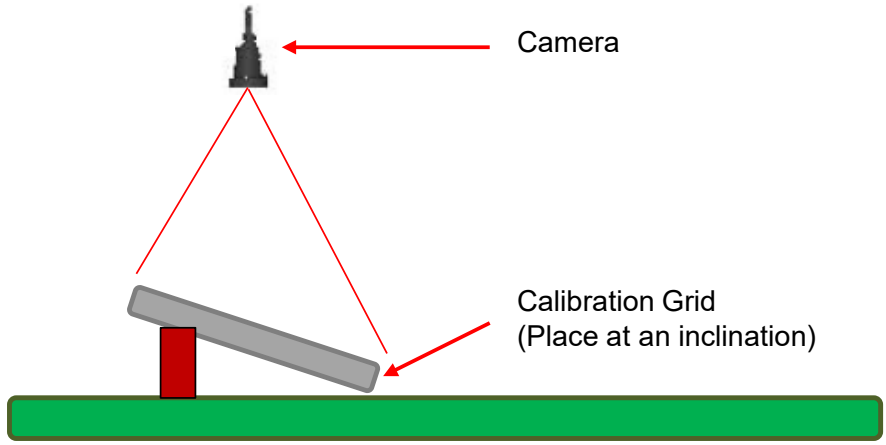
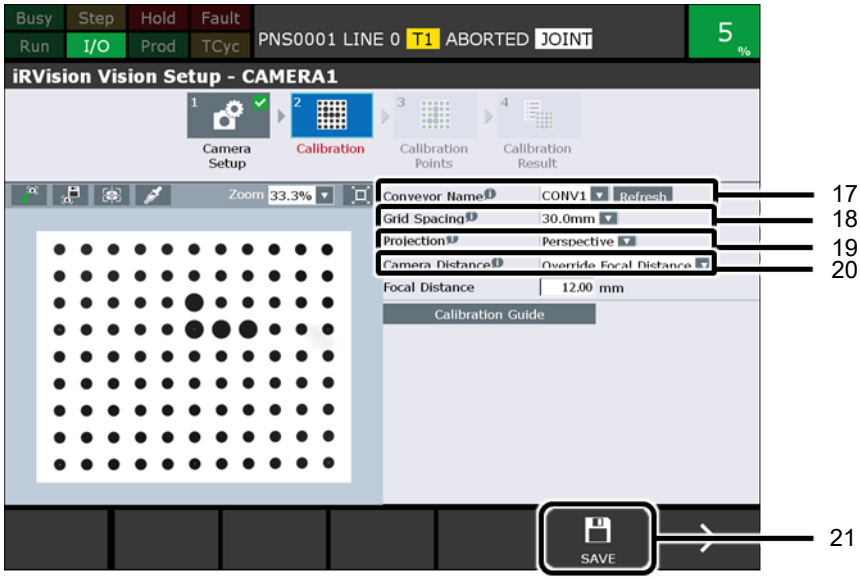
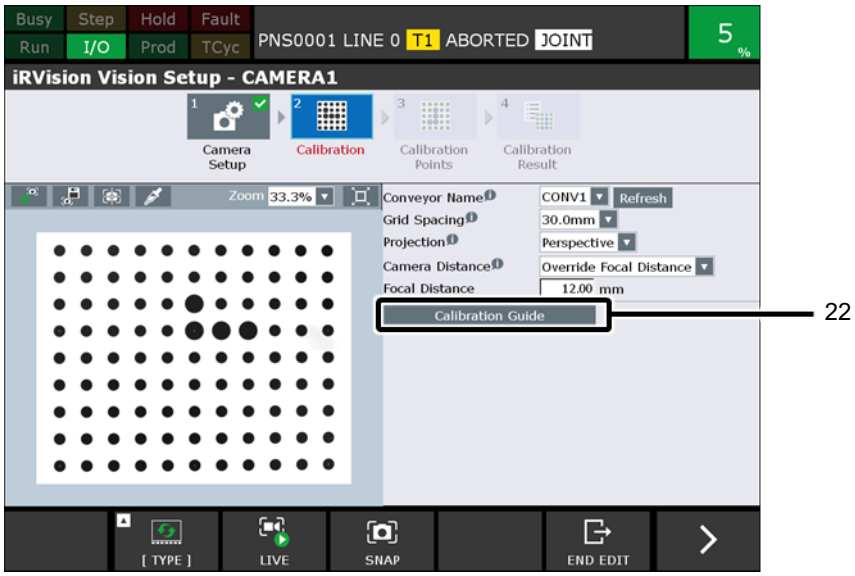


Fig. 3.2.7.1 (d)

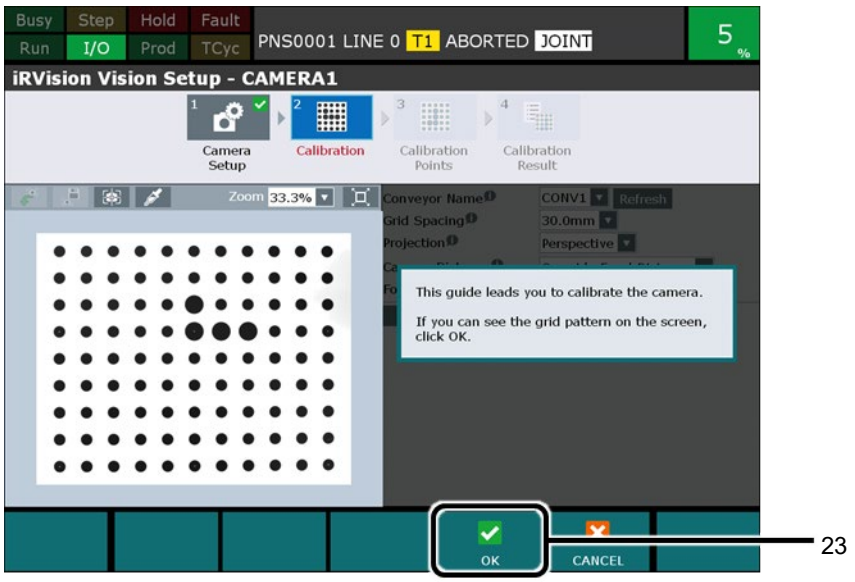
21 Press [> (Next page)] and press F5 [SAVE].



22 Press [Calibration Guide].

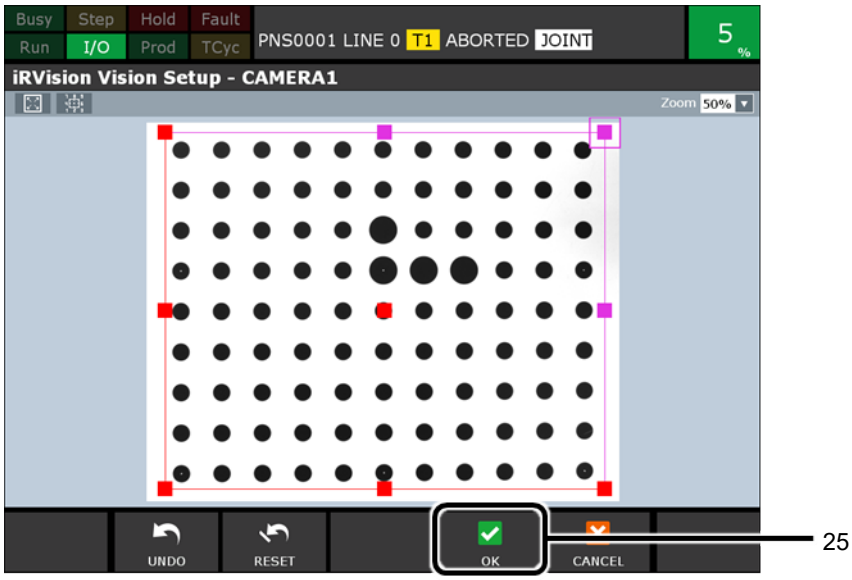


23 Check that the calibration grid appears on the screen, and press F4 [OK].

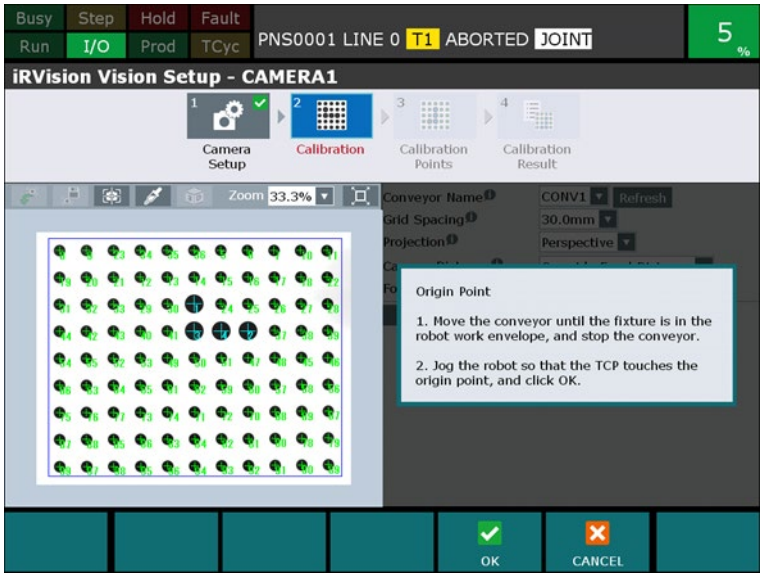


24 Enclose the dot pattern that appears on the screen with the reddish-purple rectangular window.

25 Press F4 [OK].



26 Move the conveyor. When the calibration grid gets to around the center of the operating range of the robot, stop.



27 Touch up the origin of the calibration grid.

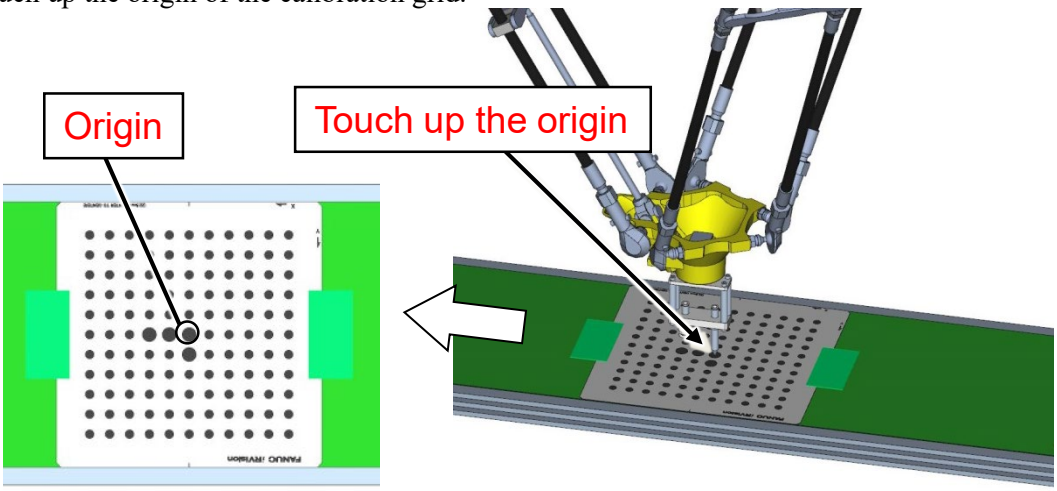
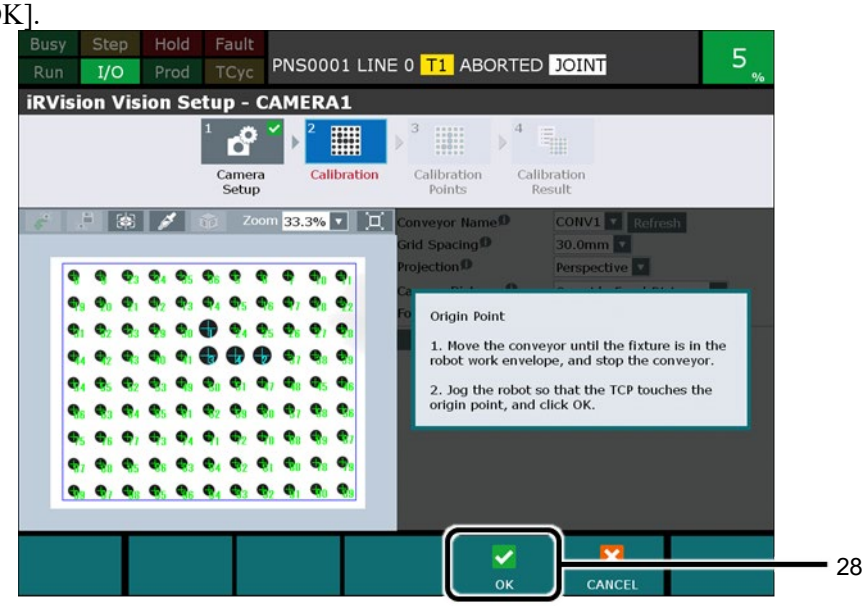


Fig. 3.2.7.1 (e) Touching up the origin of the calibration grid

28 Press F4 [OK].



29 Keeping the conveyor stopped, touch up the X-axis direction of the calibration grid.

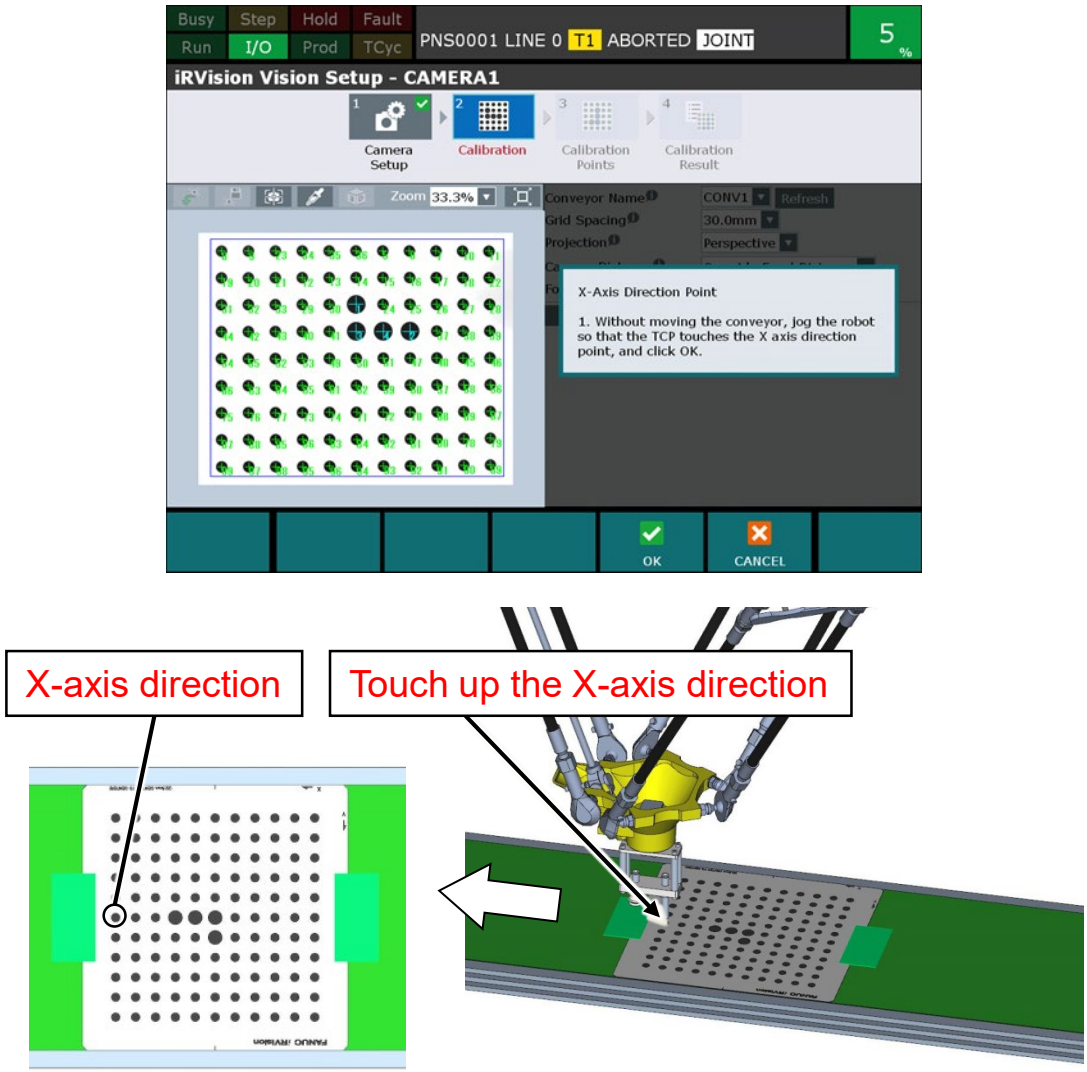


Fig. 3.2.7.1 (f) Touching up the X-axis direction of the calibration grid

30 Press F4 [OK].

31 Keeping the conveyor stopped, touch up the Y-axis direction of the calibration grid.

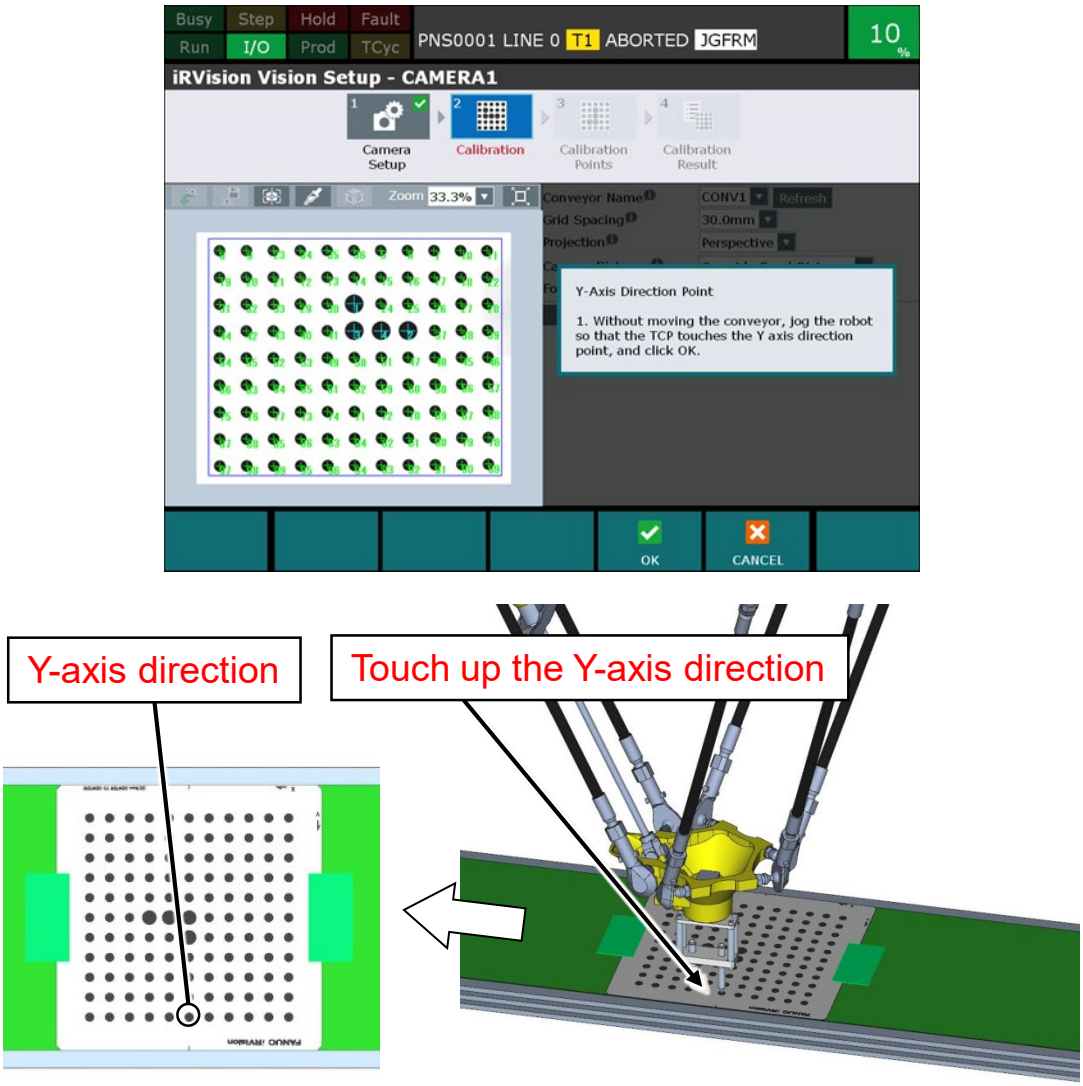
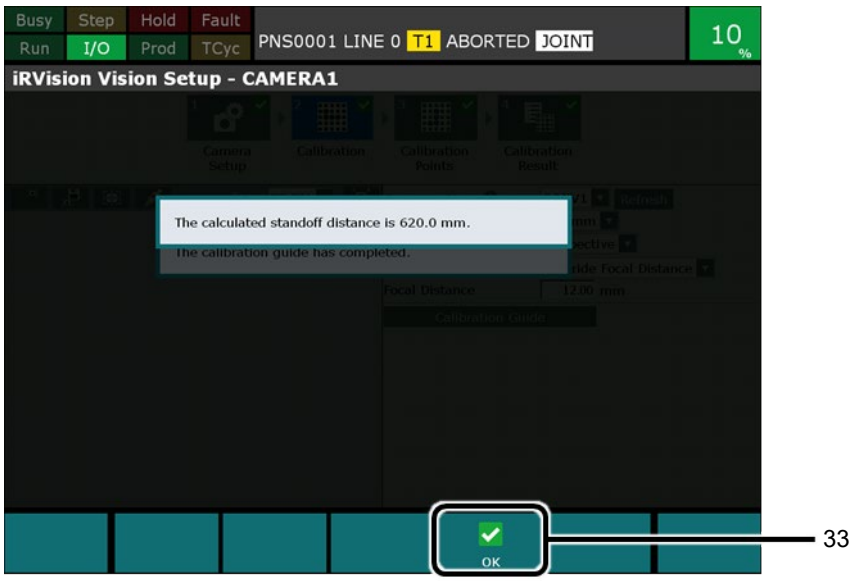


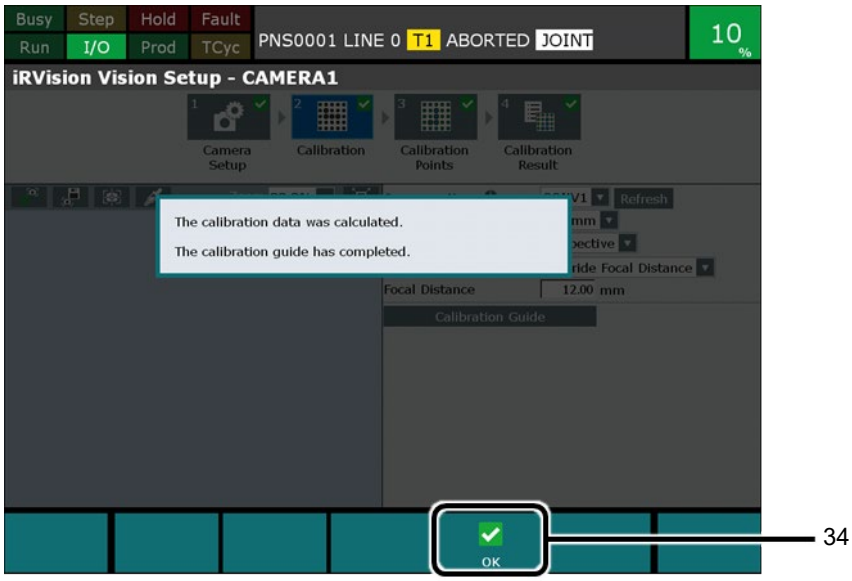
Fig. 3.2.7.1 (g) Touching up the Y-axis direction of the calibration grid

32 Press F4 [OK].

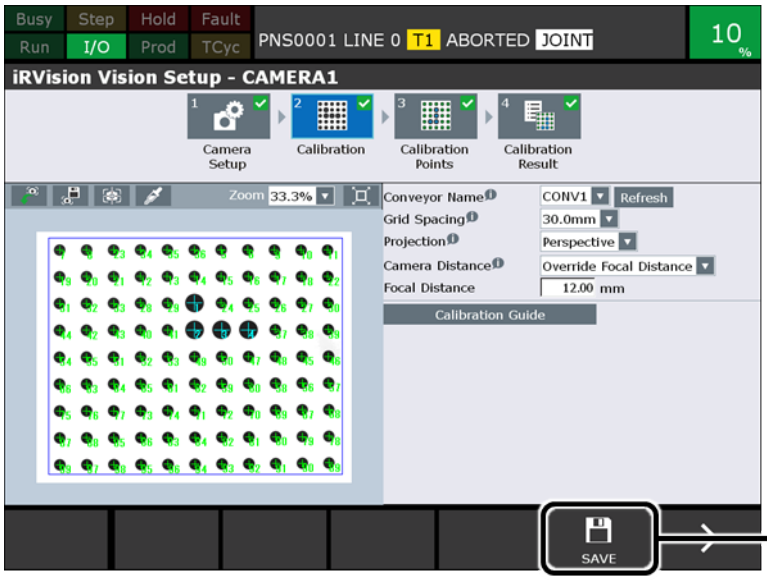
33 The calculated standoff distance is displayed. Confirm that the value is suitable for the actual standoff distance, and press F4 [OK]. If the value is unsuitable, check whether the setting procedure and set parameters are correct.



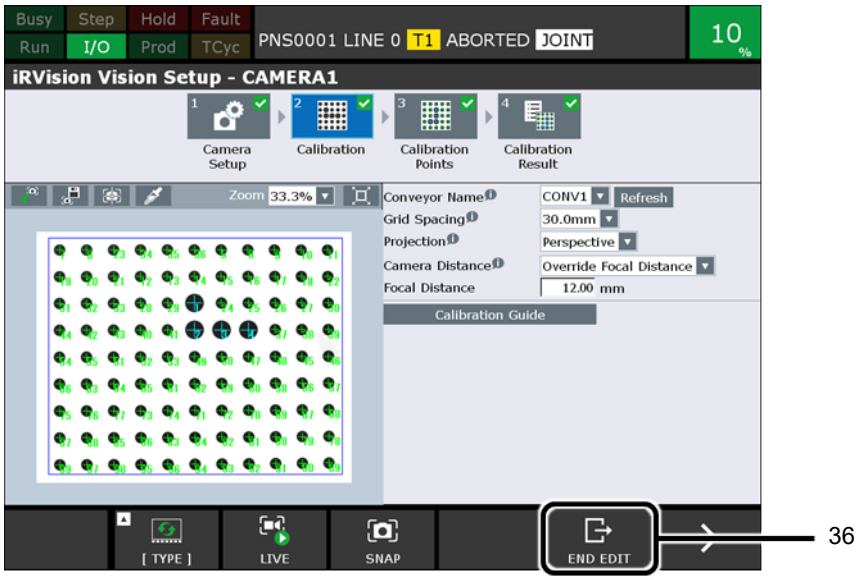
34 Press F4 [OK].



35 Press [> (Next page)] and press F5 [SAVE].



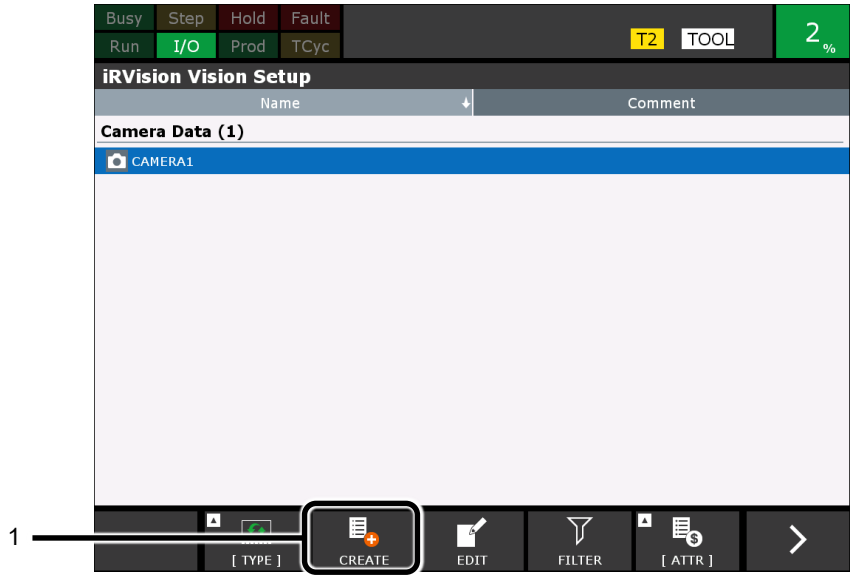
36 Press [> (Next page)] again and press F5 [END EDIT].



3.2.7.2 Vision process

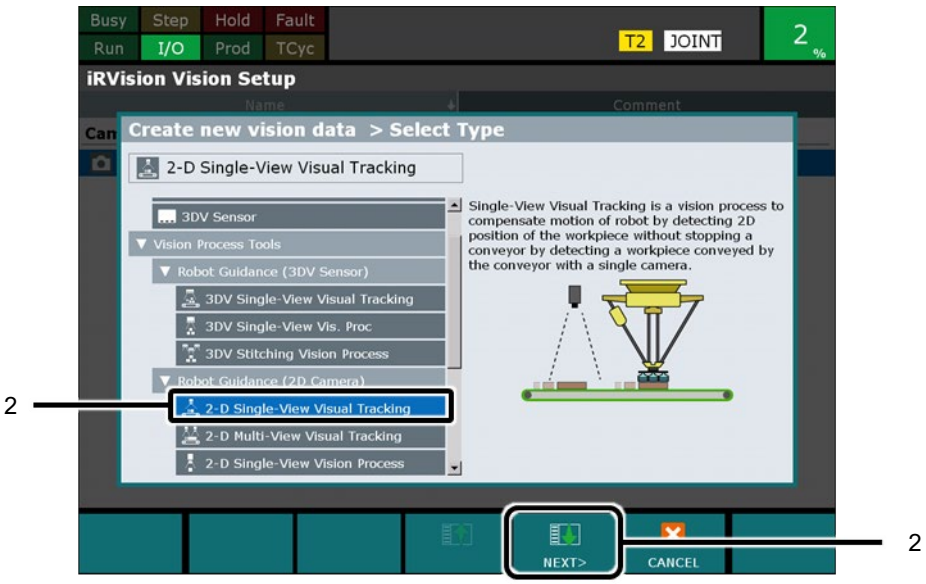
Set up a vision process.

1 Press F2 [CREATE].



The [Create new vision data] screen will appear.

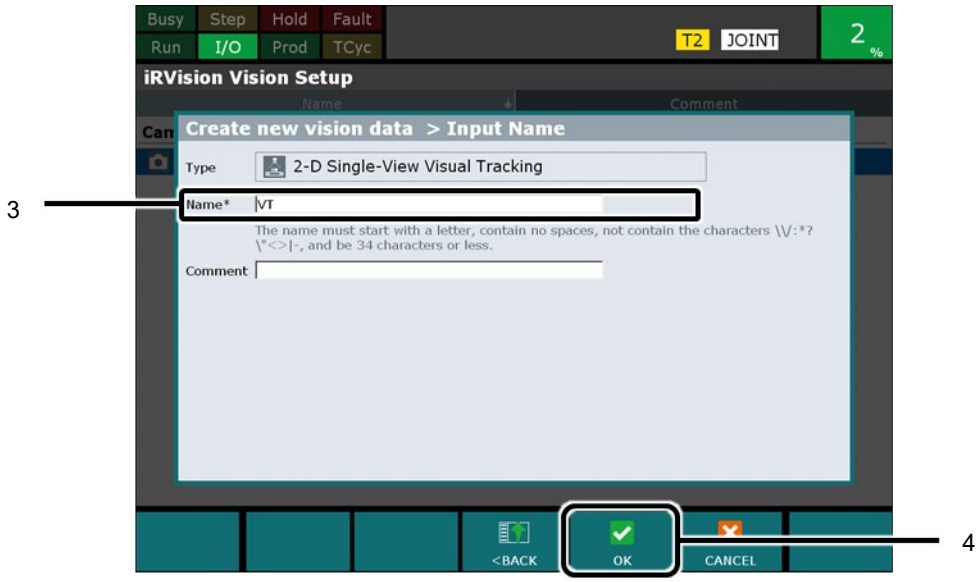
2 Select [2-D Single-View Visual Tracking] and press F4 [NEXT].



A screen like the following one will appear.

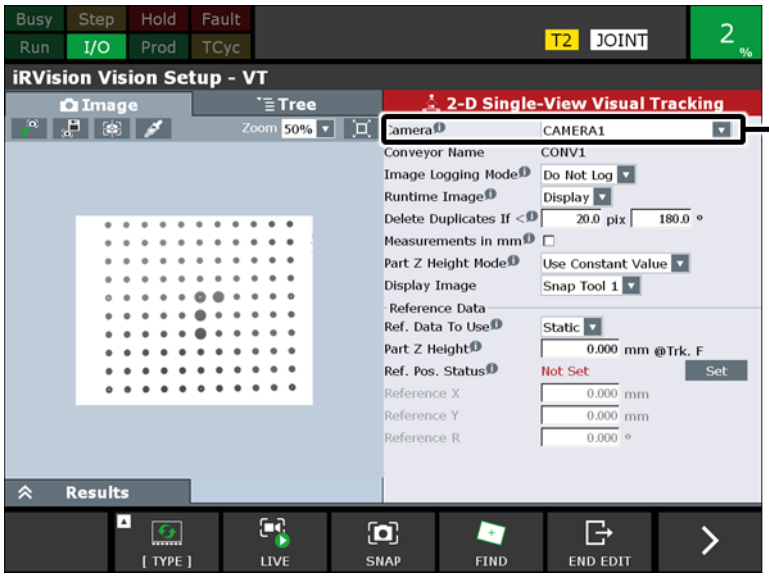
3 Enter the name of the vision process in [Name].
Input using half-width alphanumeric characters and half-width katakana. The name must contain no spaces, not contain the symbol other than underscore, and start with a letter.
Example : VT

4 Press F4 [OK].



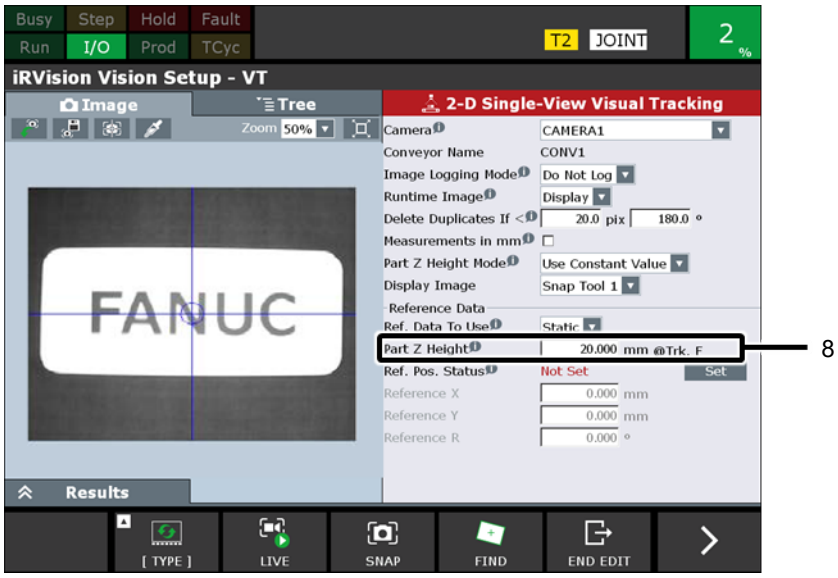
5 Select the created vision process and press F3 [EDIT].

6 Select [Camera] from the menu.



7 Measure the thickness of a part.
Measure the distance from the plane at which the tracking frame was taught (the calibration grid surface) to the plane where the features that are to be found by vision will be.

8 Enter the thickness of a part in [Part Z Height].
Enter the distance from the plane at which the tracking frame was taught (the calibration grid surface) to the plane where the features that are to be found by vision will be.



NOTE

[Part Z Height] is the distance from the plane at which the tracking frame was taught (the calibration grid surface) to the plane where the features that are to be found by vision will be. Measure it using a ruler, etc.
The relationship between the system and 'Part Z Height' is as shown in the following figure.

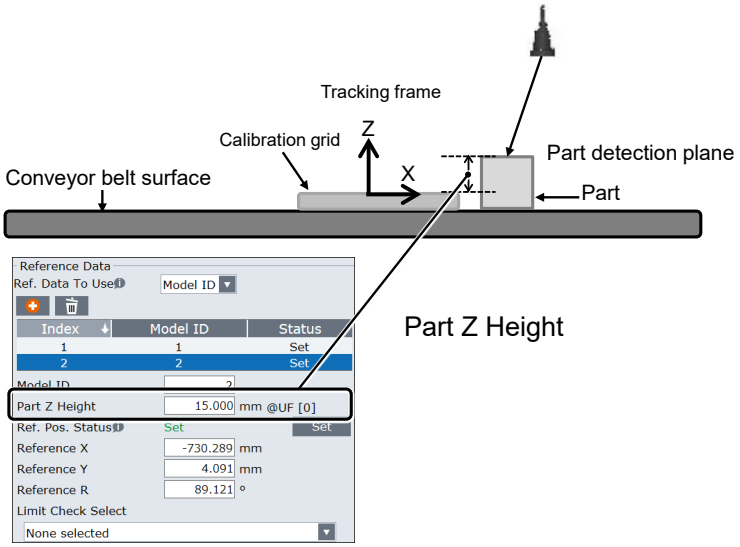
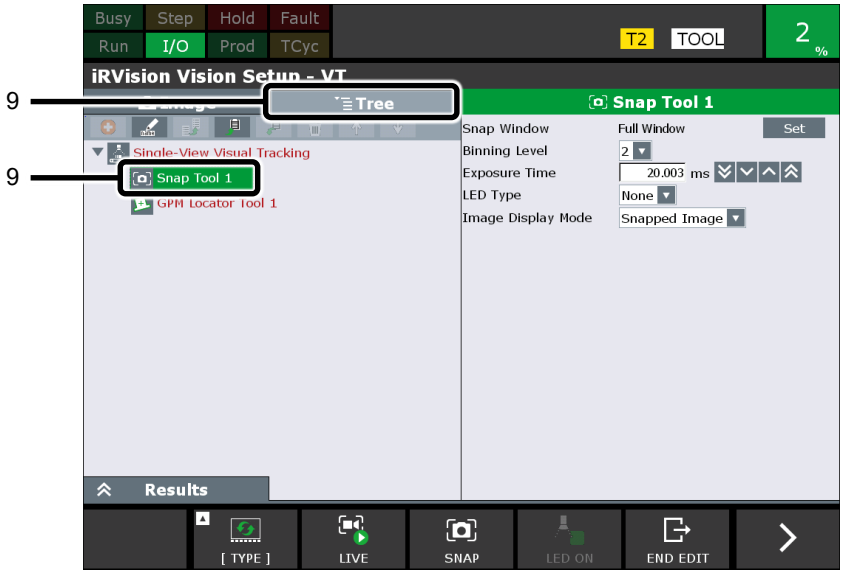


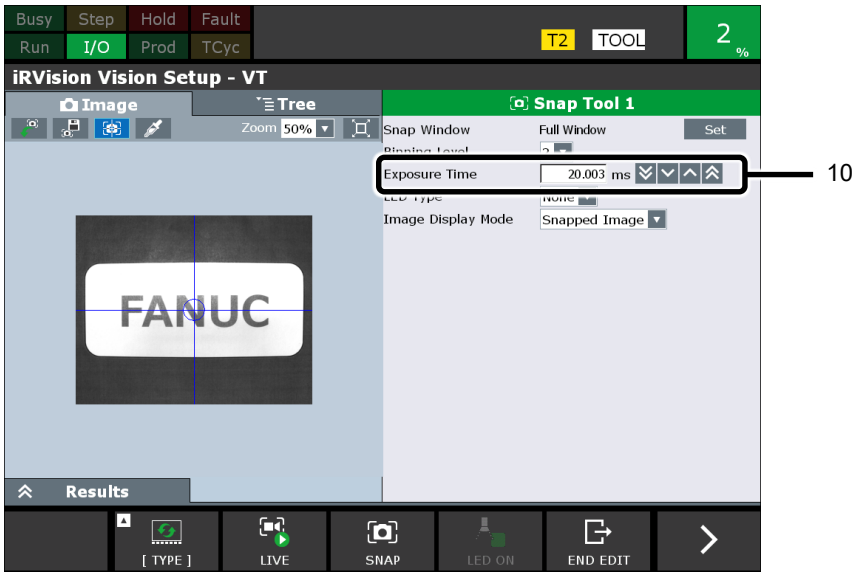
Fig. 3.2.7.2 (a) Relationship between the system and 'Part Z Height'

9 Press the Tree tab and select [Snap Tool 1].



10 Enter an [Exposure Time].
Set the exposure time so that an image of a part that moves during exposure will have as little blurring as possible.
For example, taking into account image blurring, in order to suppress misalignment during finding by vision to less than 0.1 mm when the conveyor speed is 100 mm/s, set a value that is below 1 ms as the exposure time.
0.1 (mm) ÷ 100 (mm/s) = 1 (ms)

For information on calculation of an appropriate exposure time to suit the conveyor speed, refer to "Exposure time" in Section 2.5, "STUDY FOR APPLICATION".



11 Set a part in the field of view of the camera.
If the feed direction has been decided, align the direction when setting.
If the feed posture is ±180 degrees, teaching may be easier if the posture is aligned with the posture for the placing side.

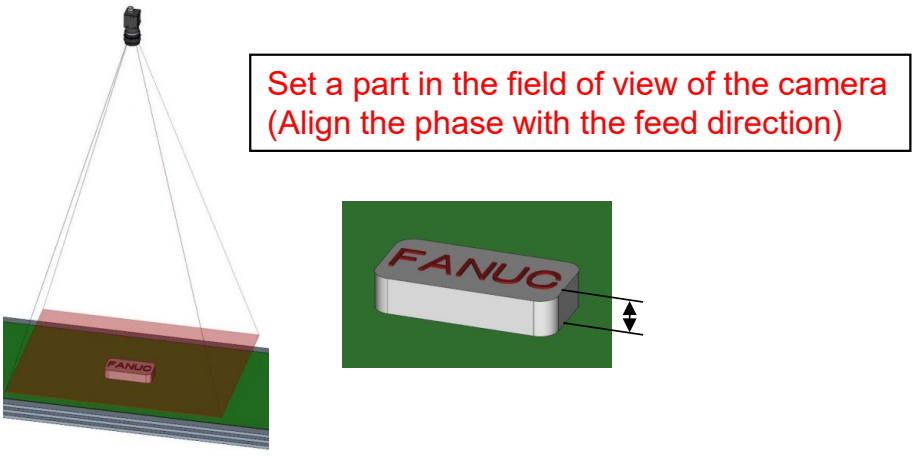
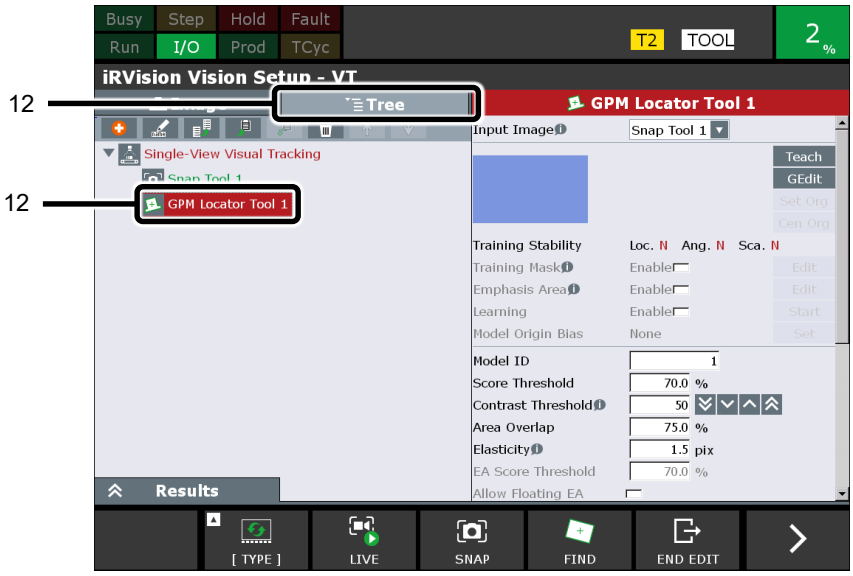


Fig. 3.2.7.2 (b) Setting a part

12 Press the Tree tab and select [GPM Locator Tool 1].

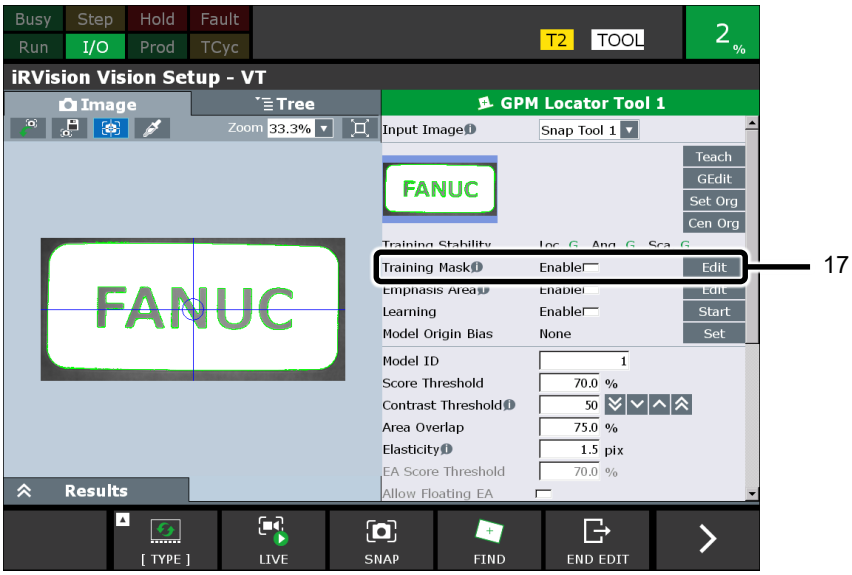


13 Press F3 [SNAP].
Check that the part appears on the screen.

14 Press [Teach].

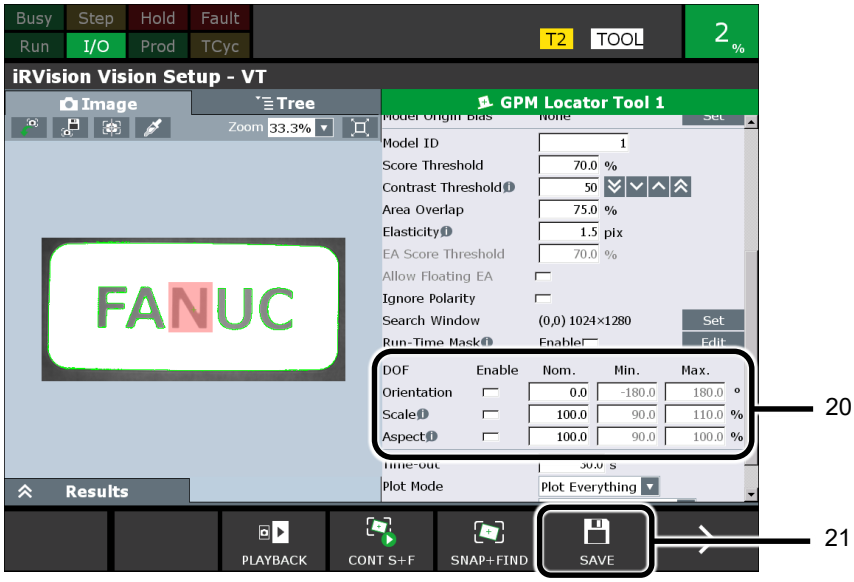
15 Enclose the part in the reddish-purple rectangular window.
16 Press F4 [OK].

17 Press [Edit] for [Training Mask].

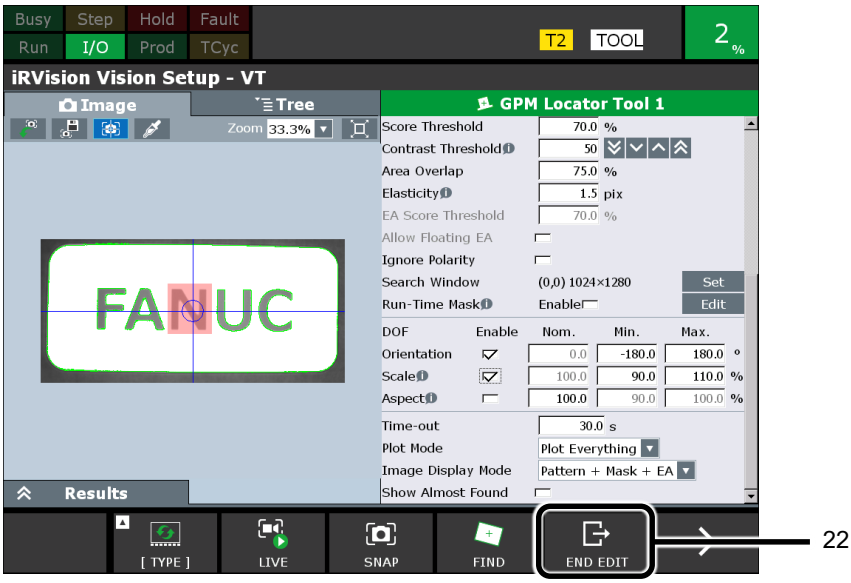


18 Select a tool at the top of the screen, and paint the section you do not want to make a feature for matching in red.
19 Press F4 [OK].

20 Set up the DOF as required.
For details on the setup, refer to the section, “GPM LOCATOR TOOL” in the chapter “COMMAND TOOLS” in the “*i*RVision OPERATOR'S MANUAL (Reference)”.
21 Press [> (Next page)] and press F5 [SAVE].



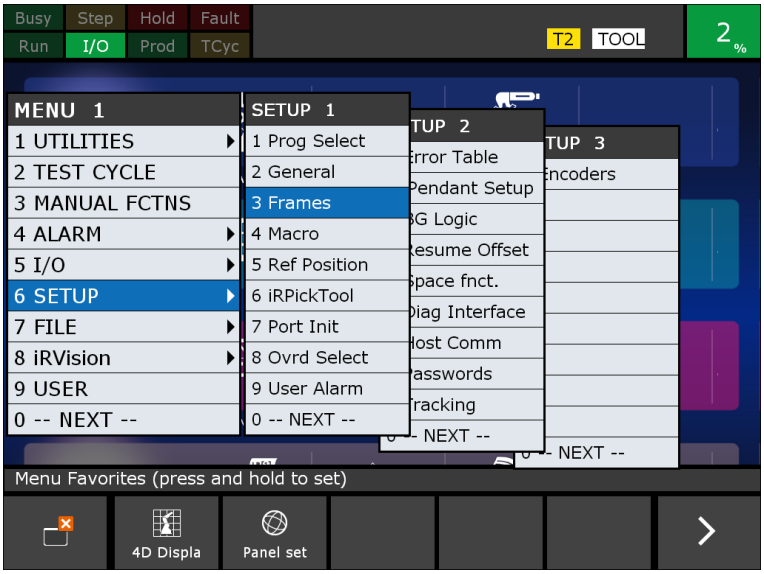
22 Press [> (Next page)] again and press F5 [END EDIT].



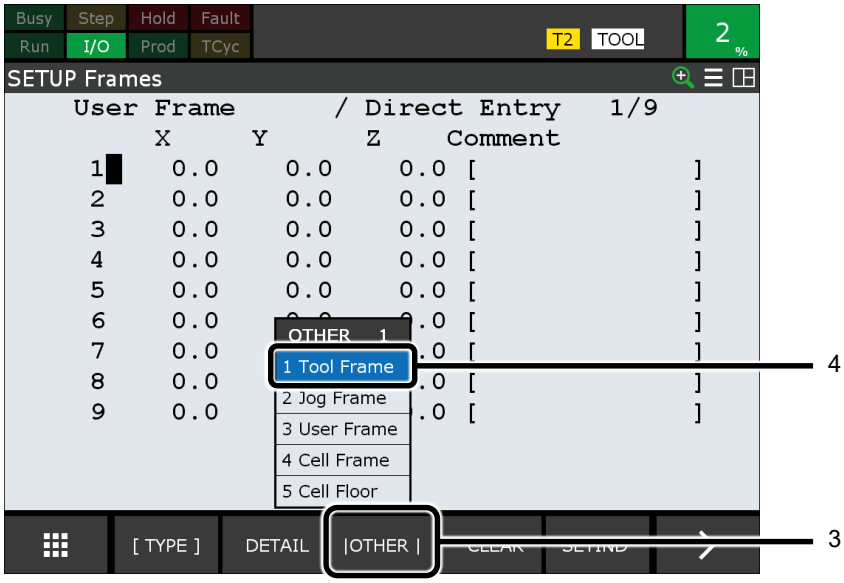
3.2.8 Setting Up a Conveyor Station

Set up a conveyor station.

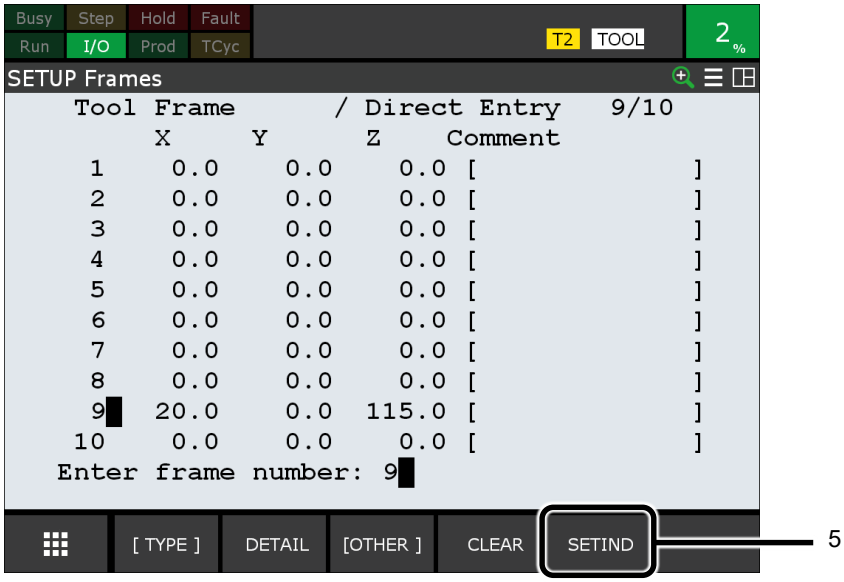
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SETUP] → [Frames] from the menu.



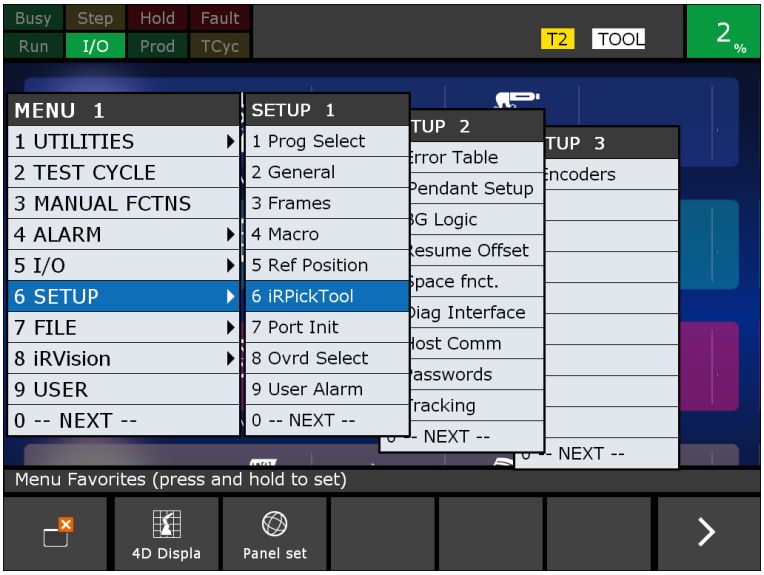
3 Press F3 [OTHER].
4 Select [Tool Frame] from the menu.
The list screen for tool frames will appear.



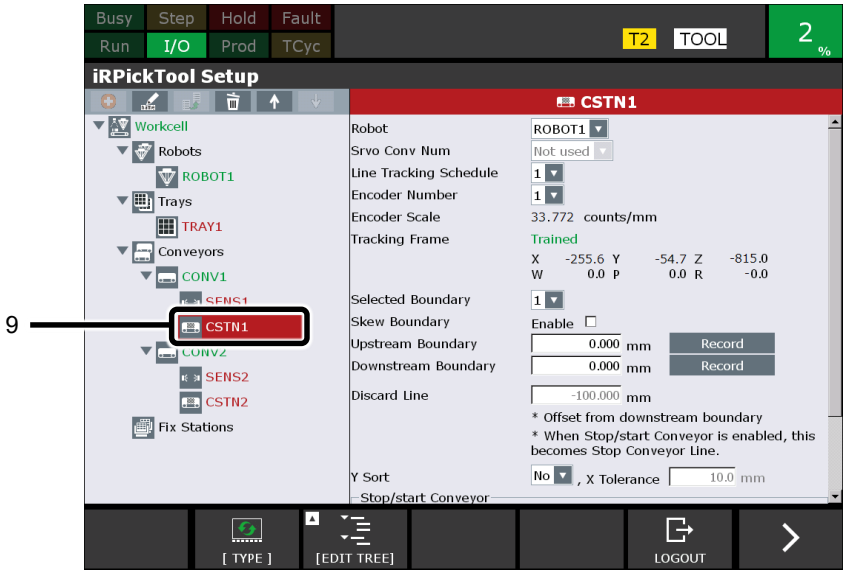
5 Press F5 [SETIND].
6 Enter the frame number that has been set for the pointer tool.
This procedure is explained using a case where Tool 9 is used as an example.



7 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
8 Select [SETUP] → [*i*RPickTool] from the menu.



9 First, set up the conveyor station of the infeed conveyor. Select the conveyor station [CSTN1].



10 Jog the robot to the upstream of the work area.

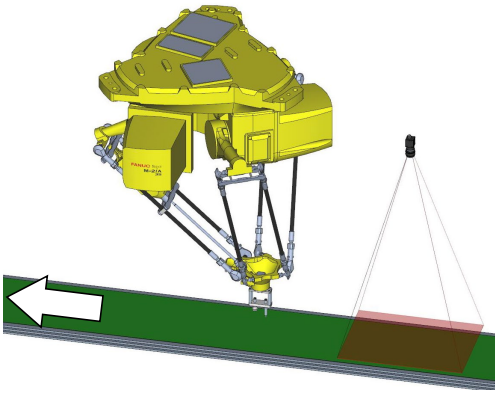
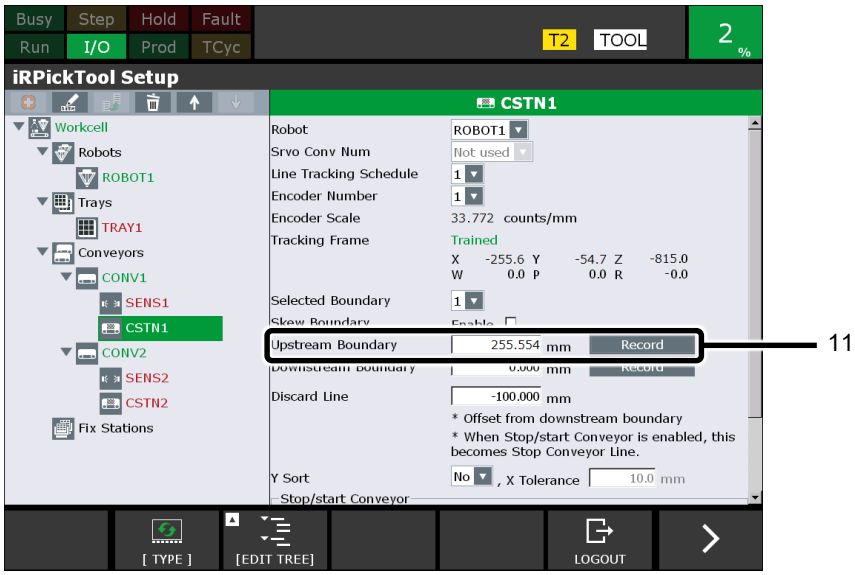


Fig. 3.2.8 (a) Jogging the robot

11 Press [Record] for [Upstream Boundary].
The current position in the tracking frame will be saved.



12 Jog the robot to the downstream of the work area.

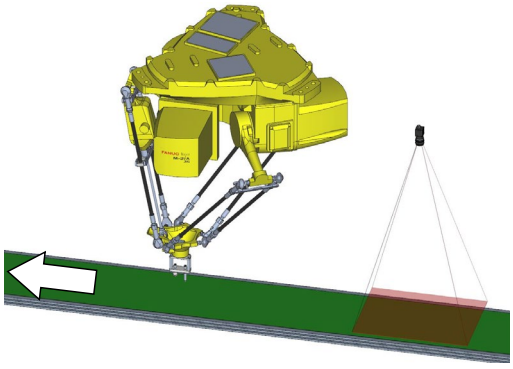


Fig. 3.2.8 (b) Jogging the robot

13 Press [Record] for [Downstream Boundary].
The current position in the tracking frame will be saved.

14 Enter [Discard Line].
The discard line value must be larger than the value calculated as below.

Time required for tracking* (sec) × Conveyor speed (mm/sec)

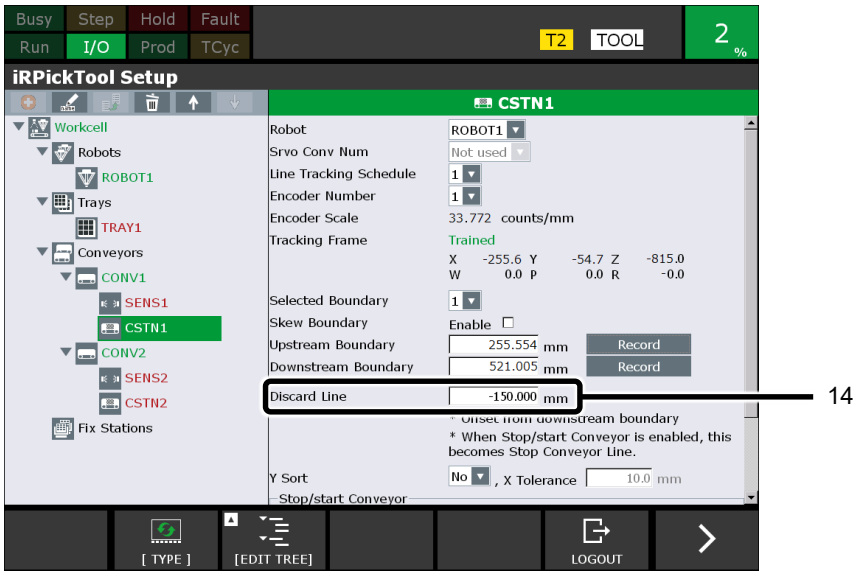
* The time required for tracking means the time from when the robot starts the tracking motion for a particular part to when the work on the part (picking, etc.) is complete.

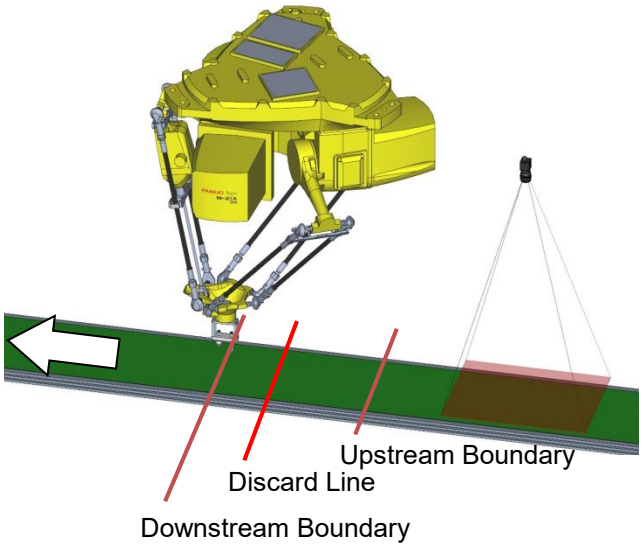
For example, if the robot's operation time is about 0.8 sec and the conveyor speed is 150 mm/sec, the free-running distance of the conveyor is

0.8 (sec) × 150 (mm/sec) = 120 (mm)

Therefore, allowing for a certain degree of error, set the discard line to around -150 mm.

For details on discard lines, refer to Section 6.6, “SETUP OF A CONVEYOR STATION”.

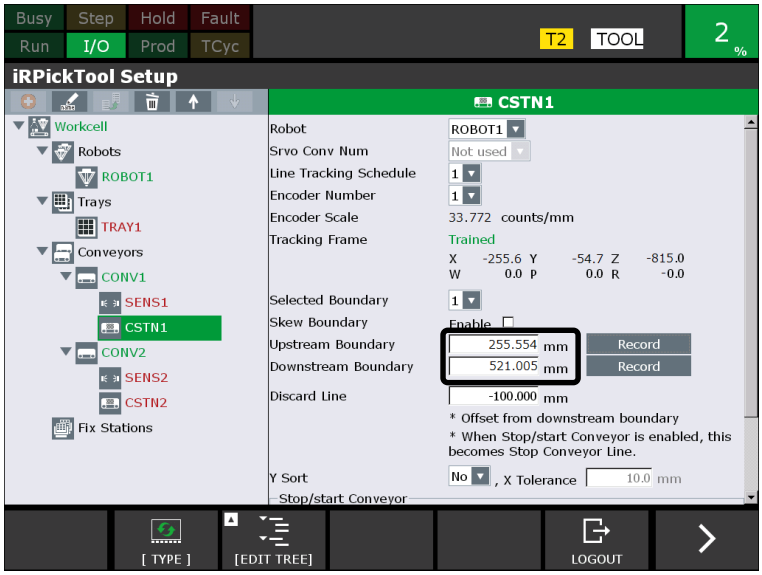




Discard Line (mm) >
Time required for tracking (sec) × Conveyor speed (mm/sec)

Fig. 3.2.8 (c) Setting up a discard line

[Upstream Boundary] and [Downstream Boundary] can also be corrected by direct input.



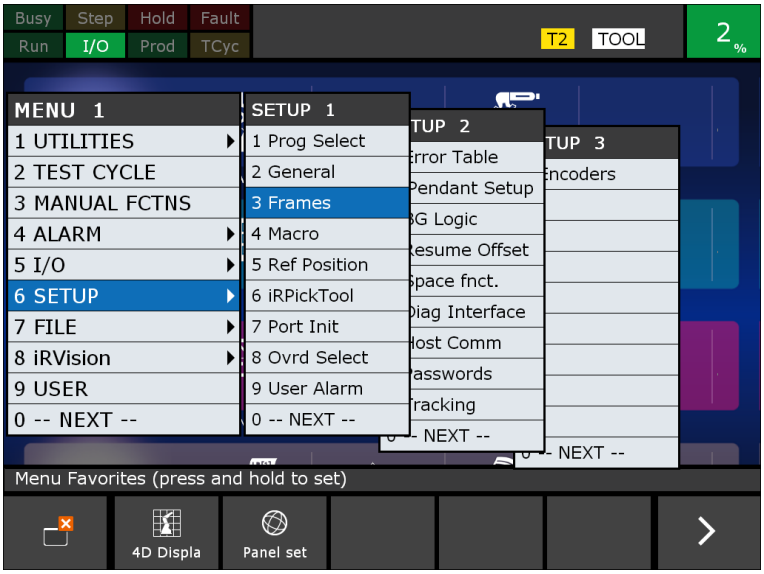
15 Next, set up the conveyor station of the outfeed conveyor. Select the conveyor station [CSTN2].
16 Repeat the steps 10 to 14 to set [Upstream Boundary], [Downstream Boundary], and [Discard lLine] for the conveyor station of the outfeed conveyor.

3.2.9 Setting Up a Tool Frame for the Hand

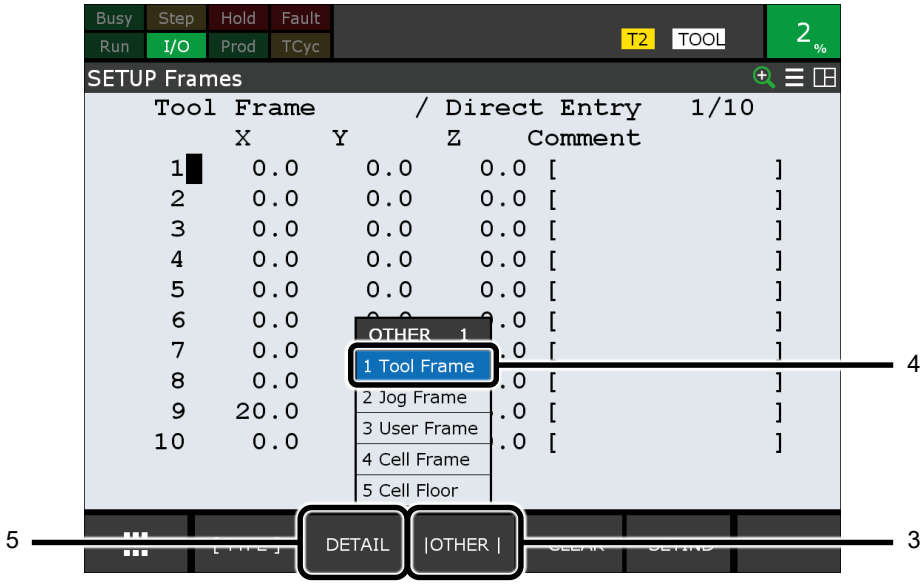
Set up a tool frame for the hand.

In this section, a four-axis Genkotsu robot is used as an example.
With a four-axis Genkotsu robot, we recommend that X and Y of the tool frame be set to 0, even if (X, Y) of the tool center point (TCP) of the hand in the mechanical interface frame is not (0, 0). This is to prevent the cycle time from being delayed or the robot's failing to stay in its operating range, as a result of large movement by the basic axes (J1 to J3 axes) when a part is picked from the conveyor.
To make the tool's Z-axis face upward, directly input the tool's posture. Directly input the values of the rotation angles W, P, and R of the tool frame around the mechanical interface frame's X, Y, and Z axes.

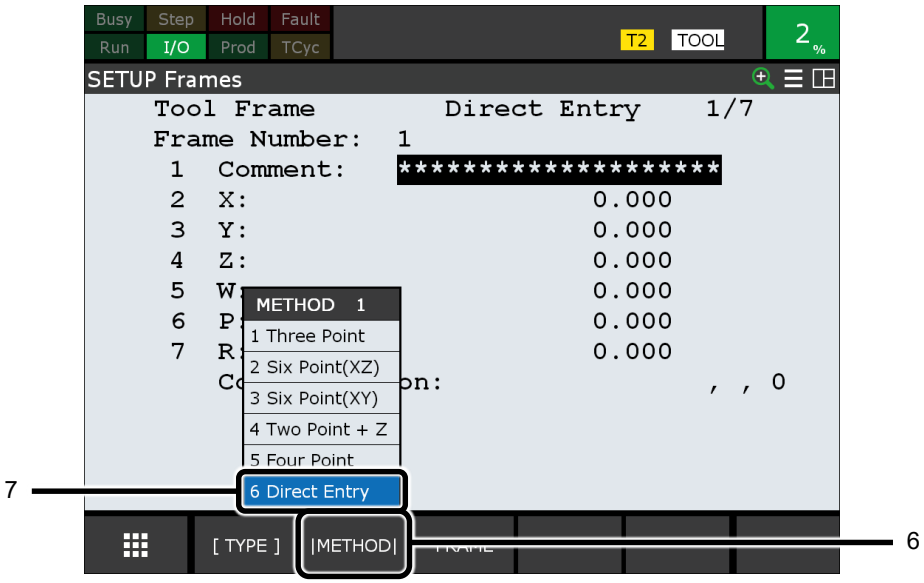
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SETUP] → [Frames] from the menu.



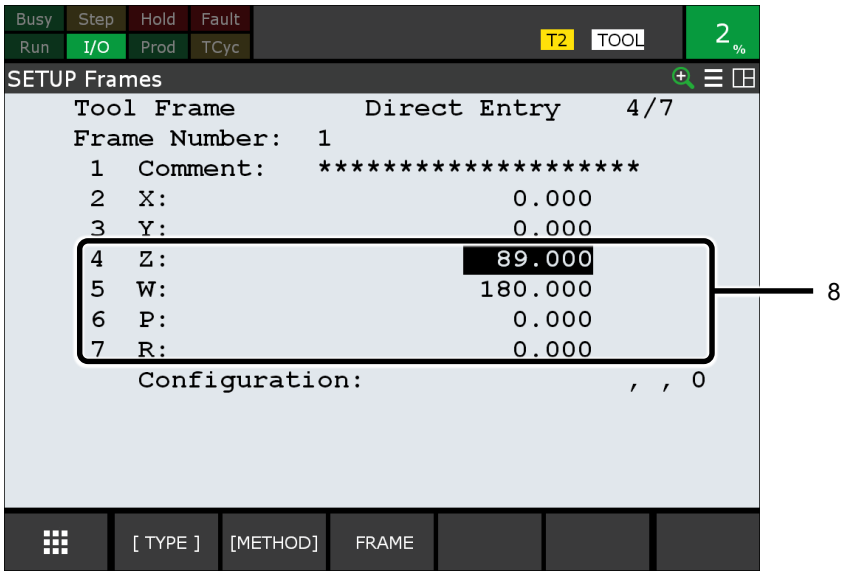
3 Press F3 [OTHER].
4 Select [Tool Frame] from the menu.
The list screen for tool frames will appear.
5 Move the cursor to the line of the tool frame number to set and press F2 [DETAIL].
The screen is an example of when performing setup using Tool 1.
The setup screen for the tool frame for the selected frame number will appear.



6 Press F2 [METHOD].
7 Select [Direct Entry] from the menu.

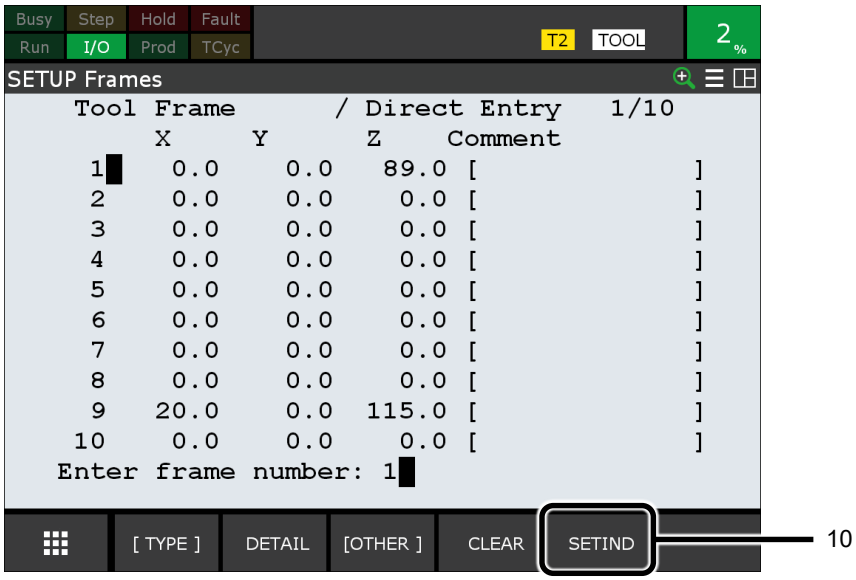


8 Move the cursor to [Z], [W], [P] and [R] and enter a value for each.
For [Z], enter a numerical value measured with a ruler, etc., or a hand design value.
Input [W], [P] and [R] directly.
* (W, P, R) = (180, 0, 0) is recommended.

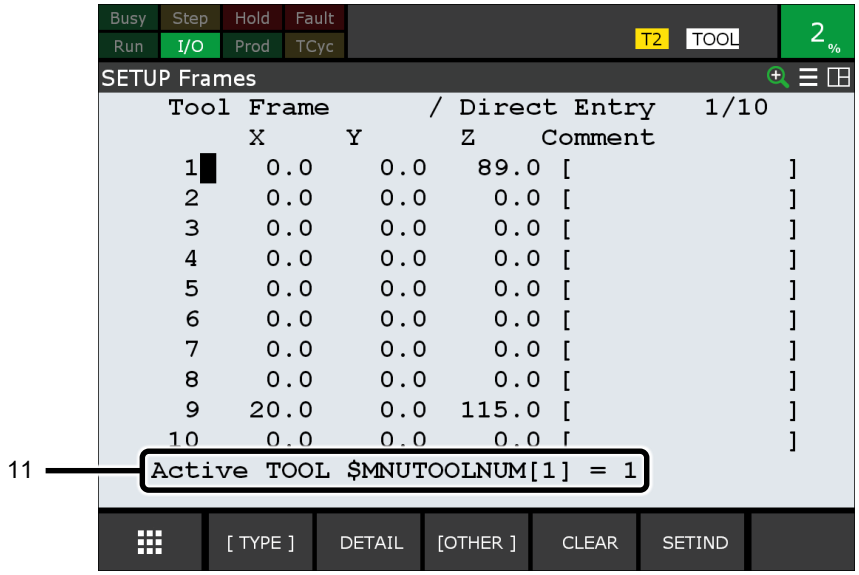


9 Press the [PREV] key on the teach pendant of the robot controller.
The list screen for tool frames will appear.
The setting values of all the tool frames can be checked.

10 Press F5 [SETIND].



11 Enter a frame number.
The tool frame will be enabled.
* If you do not carry out steps 10 and 11, the frame that has been set up will not be enabled.



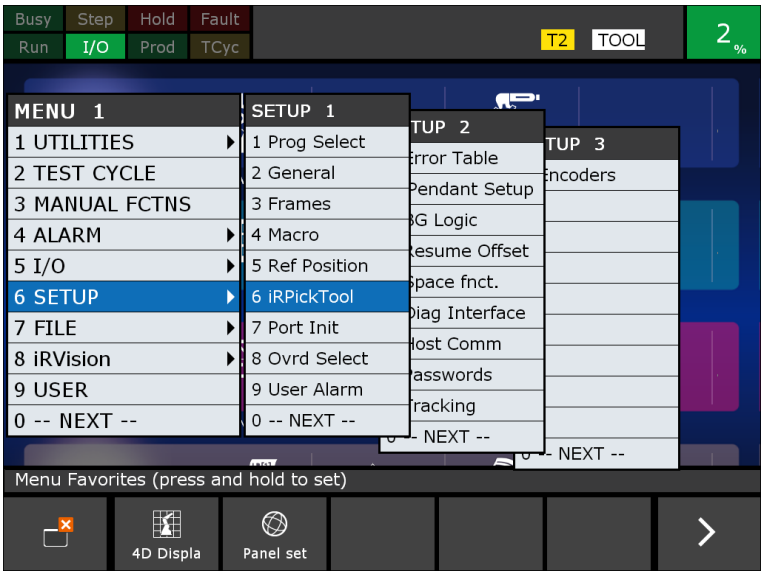
3.2.10 Setting Up a Reference Position and Teaching a Robot Position

Set up a reference position, and teach a robot position.

3.2.10.1 Setting up a reference position for the infeed conveyor, and teaching a robot position

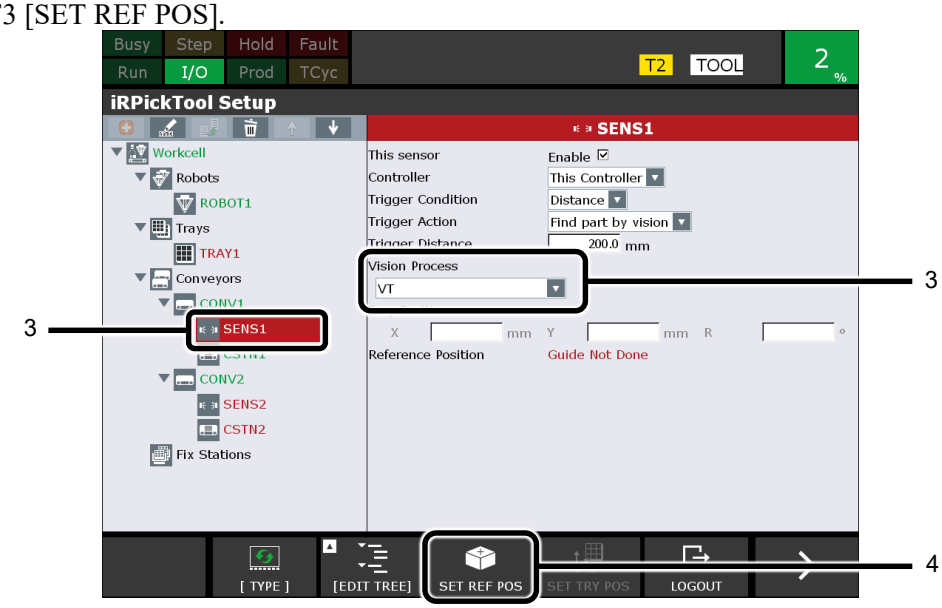
Set up a reference position for the infeed conveyor, and teach a robot position. The procedure is for the case of [Distance] + [Find part by vision]

1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SETUP] → [*i*RPickTool] from the menu.

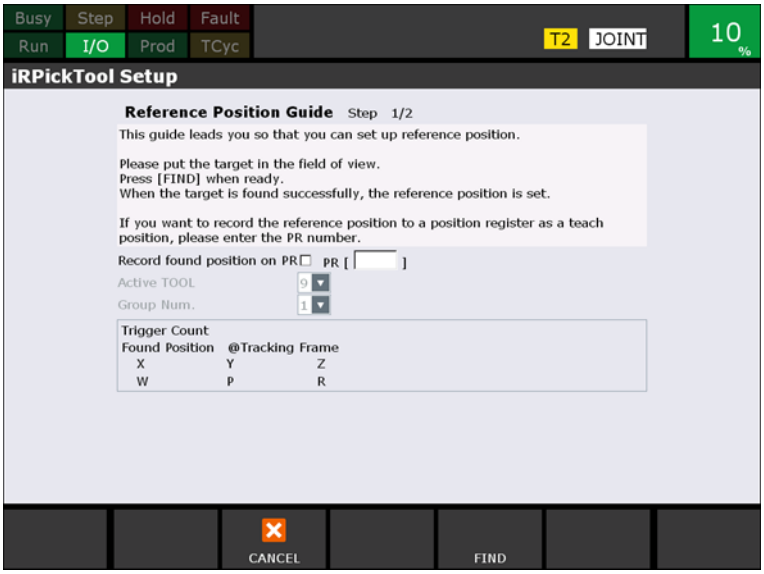


3 Select the sensor [SENS 1], and for [Vision Process], select the name of the vision process (VT) that you created in Subsection 3.2.7.2, “Vision process” .

4 Press F3 [SET REF POS].



[Reference Position Guide Step: 1/2] will appear as below.



NOTE

When [Record found position on PR] is available, the robot position teaching, which will be described later, can be simplified by using this function. For details, refer to “Using a position register number for [Record found position on PR]” below.

5 Place a part around the center of the field of view of the camera.

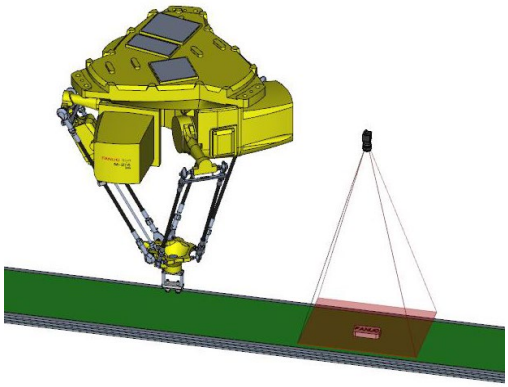
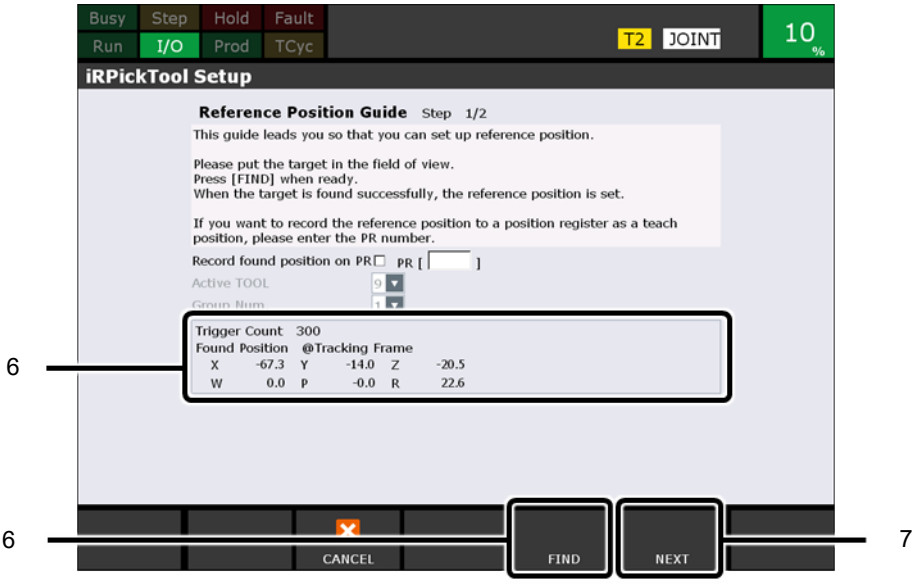


Fig. 3.2.10.1 (a) Placing a part inside the field of view of the camera

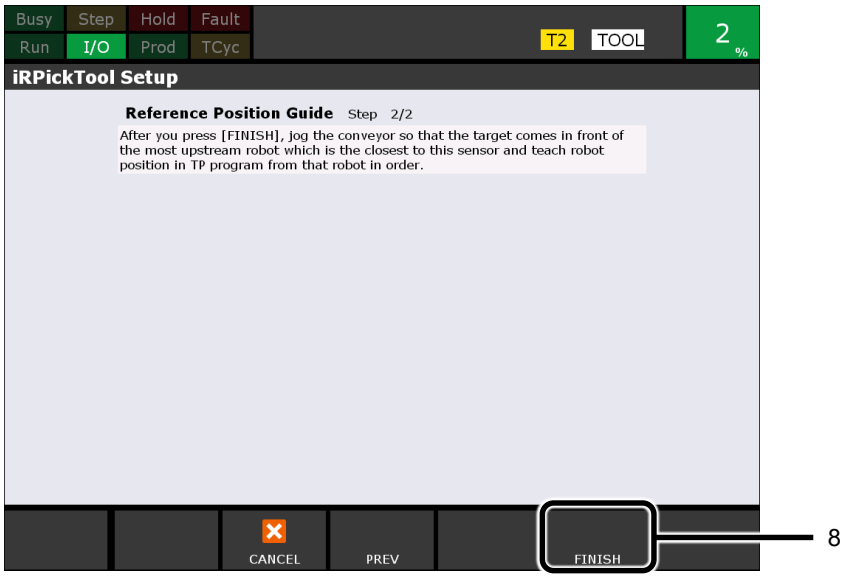
6 Press F4 [FIND].
If detection is successful, [Found Position] will appear.
* If detection is not successful
  - Open the vision process and perform a detection test.
  - When finding using a detection test, check to see if the selection of the vision process is correct on the sensor screen (step 3).

If detection is successful, F5 [NEXT] will become active.

7 Press F5 [NEXT].



8 Press F5 [FINISH].



A screen like the following one will appear.
Check that [Guide Done] is displayed for [Reference Position].

9 Move the conveyor. When the part reaches within the operating range of the robot, stop.
* Be careful that the part does not become misaligned.

10 Jog the robot so that it moves to the part pick position.

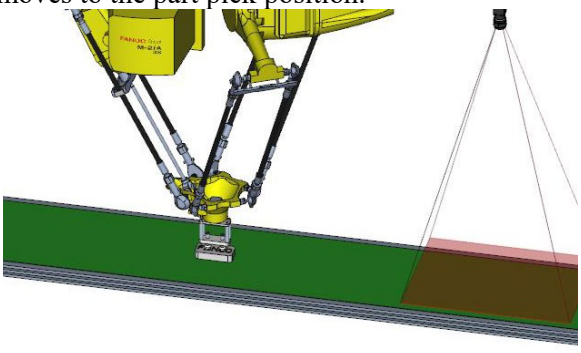
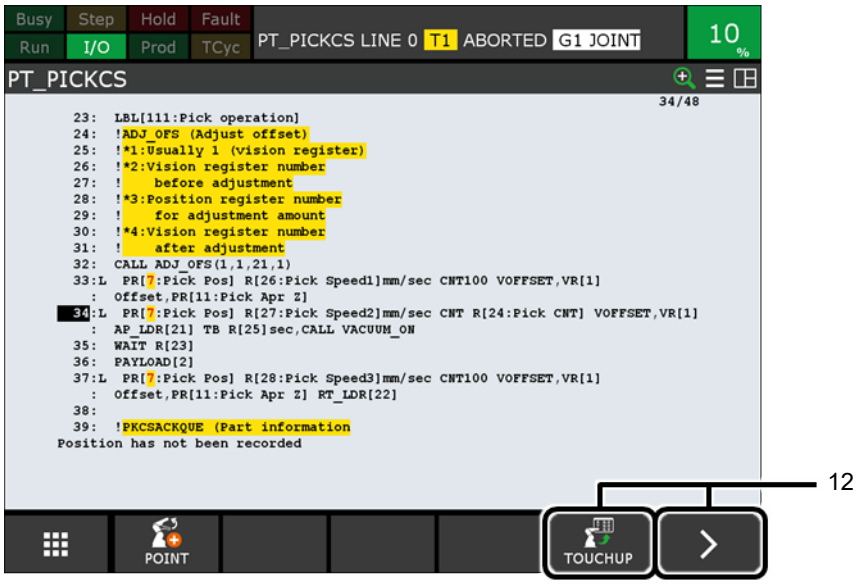


Fig. 3.2.10.1 (b) Moving the robot to the part pick position.

11 Display the pick program PT_PICKCS, and move the cursor to the line where the pick position is taught.
12 While holding down the [SHIFT] key, press F5 [TOUCHUP].
If F5 [TOUCHUP] is not displayed, press [> (Next page)] to switch the display.



13 A message saying 'VR is UNINIT, continue?' is displayed, but press F4 [YES] and teach the position.

NOTE

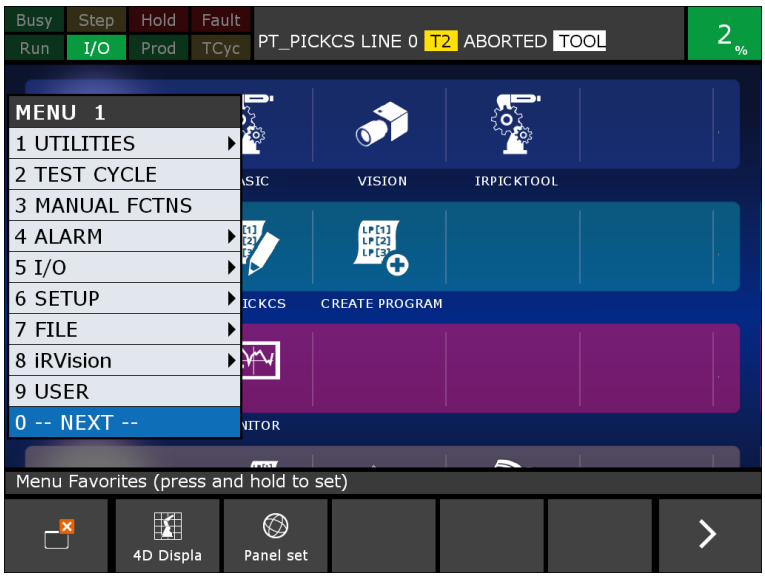
1 If there is a value in the vision register when you are setting it again, 'Subtract VR from current pos?' will be displayed. Press F5 [NO].

NOTE

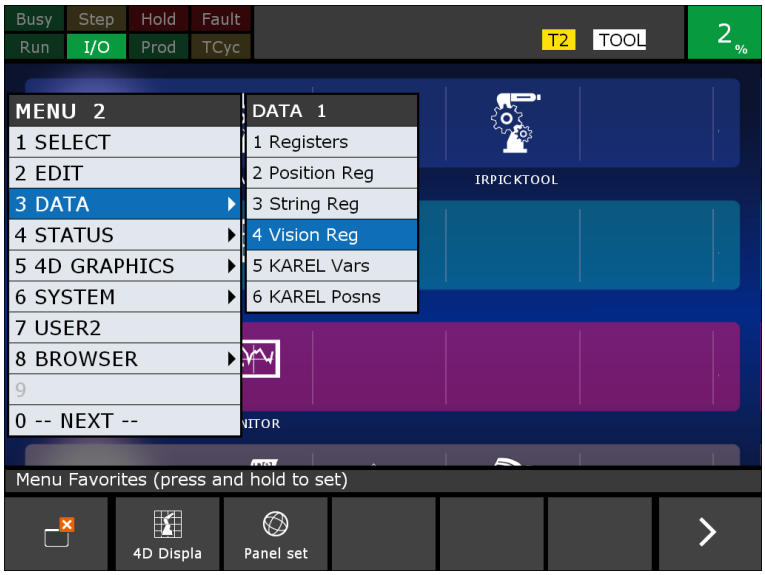
2 If there is a value in the vision register when you are setting it again, then in order to avoid making a mistake, clear the vision register value by following the procedure below, and then teach the position. If you do this, the same position will be displayed whether you select F4 [YES] or F5 [NO] for 'Subtract VR from current pos?'

1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.

2 Select [NEXT] from the menu.

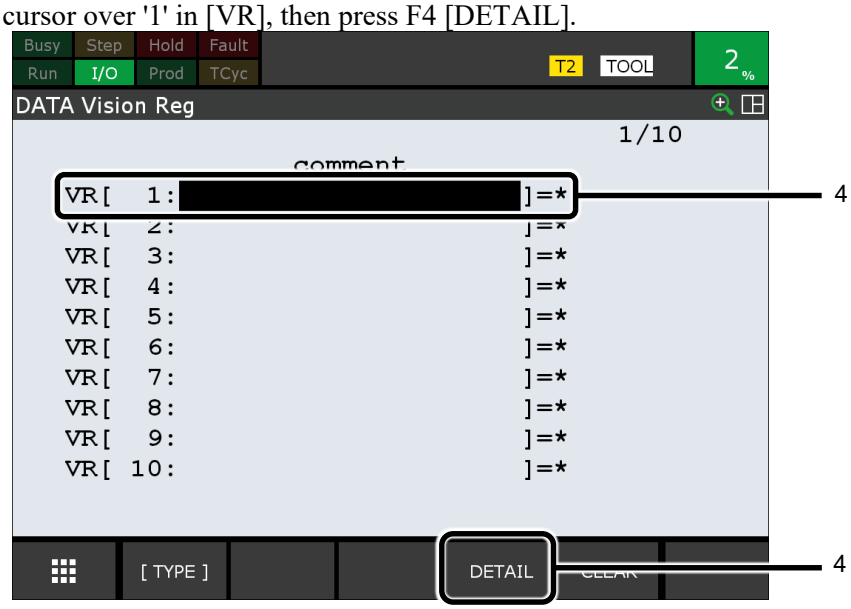


3 Select [DATA] → [Vision Reg].

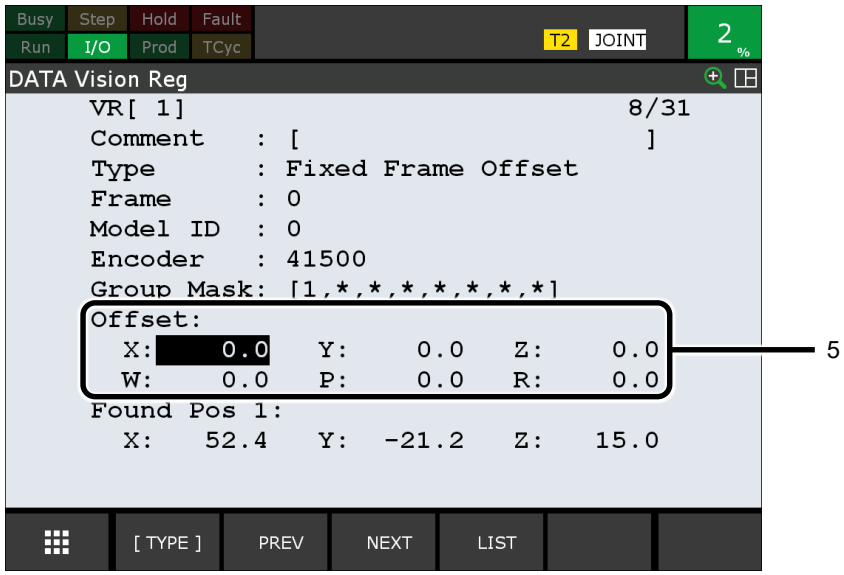


NOTE

4 Place the cursor over '1' in [VR], then press F4 [DETAIL].



5 Enter '0' for all of the values [X], [Y], [Z], [W], [P] and [R] in [Offset].



Using a position register number for [Record found position on PR]

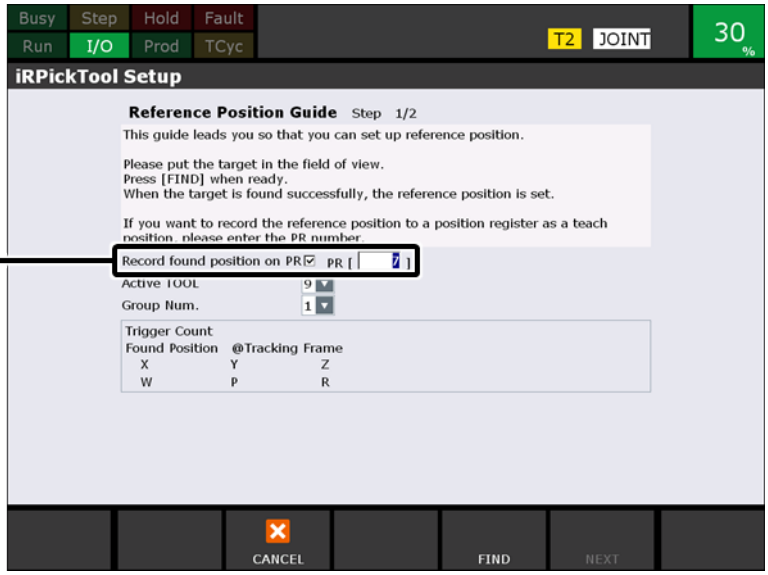
In the [Reference Position Guide Step 1/2] displayed in step 4, if a position register number is entered for [Record found position on PR], the found position will be automatically set as the reference position in the position register, and the robot position teaching can be simplified.

This function can be used under the following conditions.

- The vision model origin is correct.
- The part Z height is correct.
- The tool frame setup for the hand is correct.

The procedure for setting up a reference position using a position register is as follows.

1 Check the [Record found position on PR] box, and enter the position register number that is to be used as a pick position in [PR] in [Reference Position Guide Step: 1/2]. Enter 7 if you are going to use position register [7] as in the sample program (PT_PICKCS) given above as a pick program.



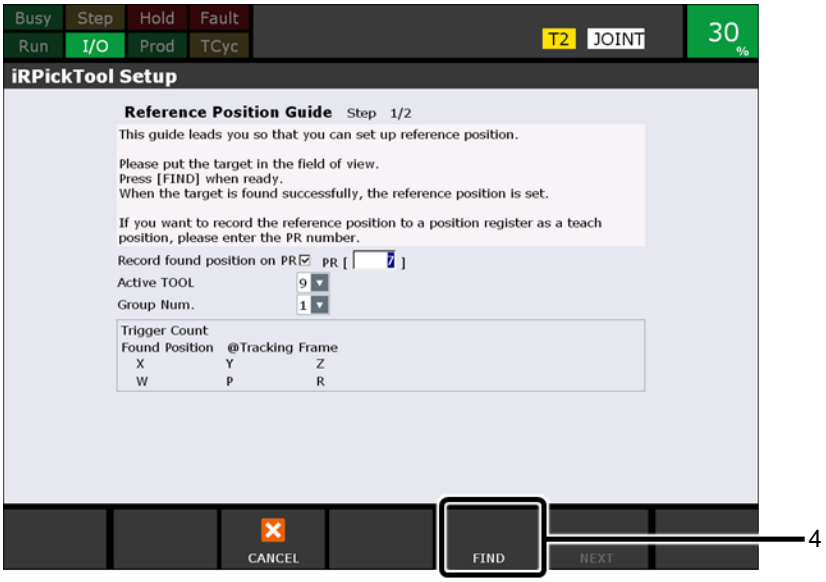
2 Enter the tool frame number in [Active TOOL].

> **⚠ CAUTION**
> The posture and configuration of the robot that will be stored in the position registers are determined from the current position of the selected tool. If a tool frame number that differs from the one that was set in Subsection 3.2.9, “Setting Up a Tool Frame for the Hand” is specified in [Active TOOL], the robot may not be in the posture or configuration that was intended, which may be dangerous. Carry out the operations with regard to the setup of the reference position with the utmost caution.

3 Moving the robot by jog operation, adjust the tool position so that the robot will be in the posture for when picking up a part.

⚠ CAUTION
In the case of circular tracking, the workpiece phase changes between when the part is detected and when the part moves to the tracking area and is picked up by the robot. Adjust the pickup position to the phase at the time of part detection, not to the phase at the time when the part moves to the tracking area and is picked up by the robot.

4 Press F4 [FIND].



If detection is successful, a value will appear in [Found Position].

5 Check to see if detection has gone well using Runtime Image.

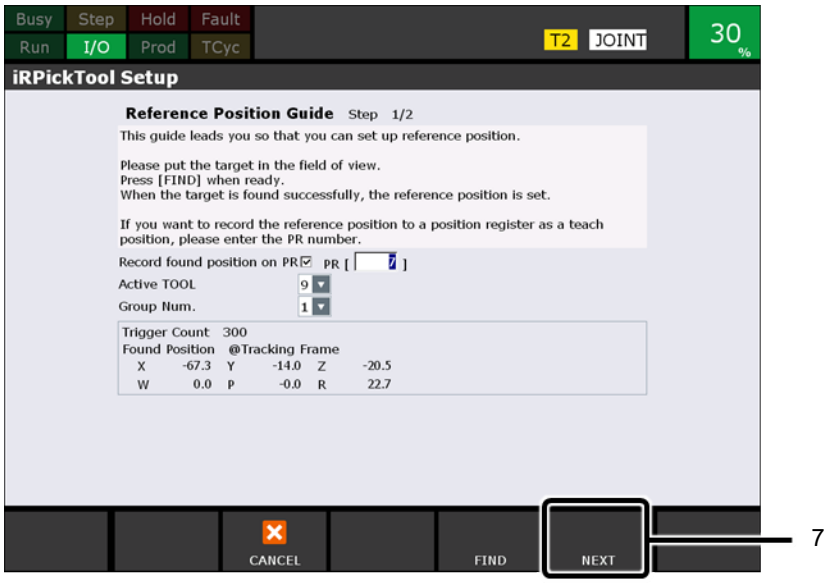
6 If Runtime Image is already being displayed, press the [DISP] key a few times while holding down the [i] key on the teach pendant, and a screen for the detection result will be displayed on the '*i*RVision' Runtime Image. If Runtime Image does not appear, display the [*i*RVision] Runtime Image.

After checking the detection result, press the [DISP] key a few times while holding down the [i] key again to return to the Reference Position Guide screen.

NOTE
If detection does not succeed, display the vision process on the *i*RVision Vision Setup screen and perform a detection test. If you cannot check the detection result in Runtime Image even though the detection test succeeded on the *i*RVision Vision Setup screen, check to see if the vision process is correct.

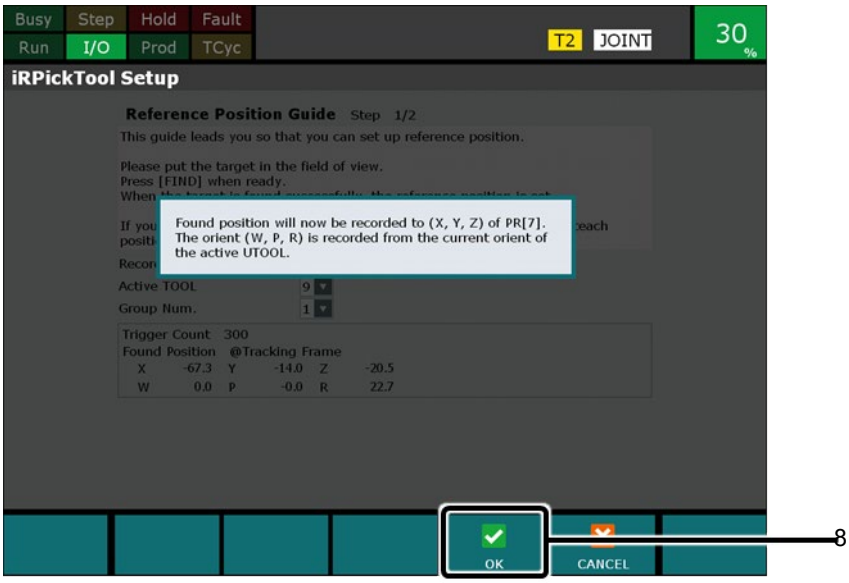
If detection is successful, [Found Position] will appear.

7 Press F5 [NEXT].



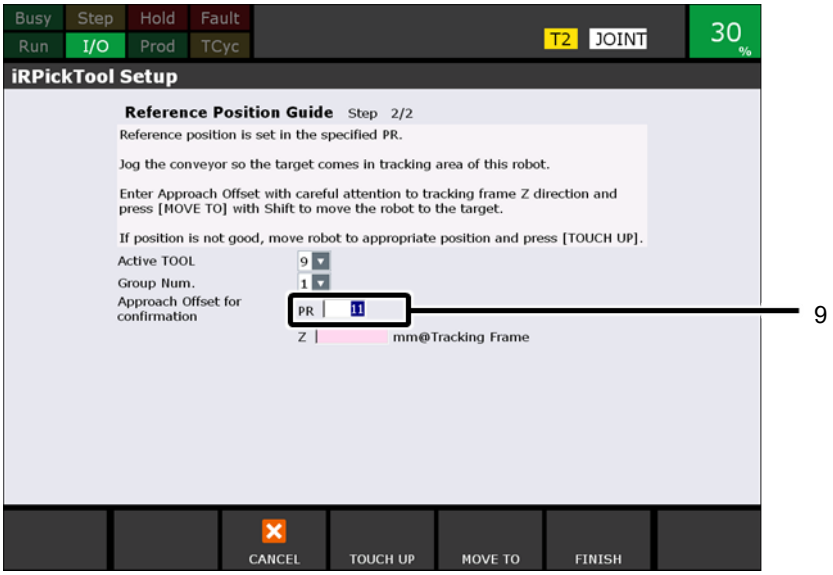
A message to confirm the position register that was set in step 2 will appear.

8 Press F4 [OK].



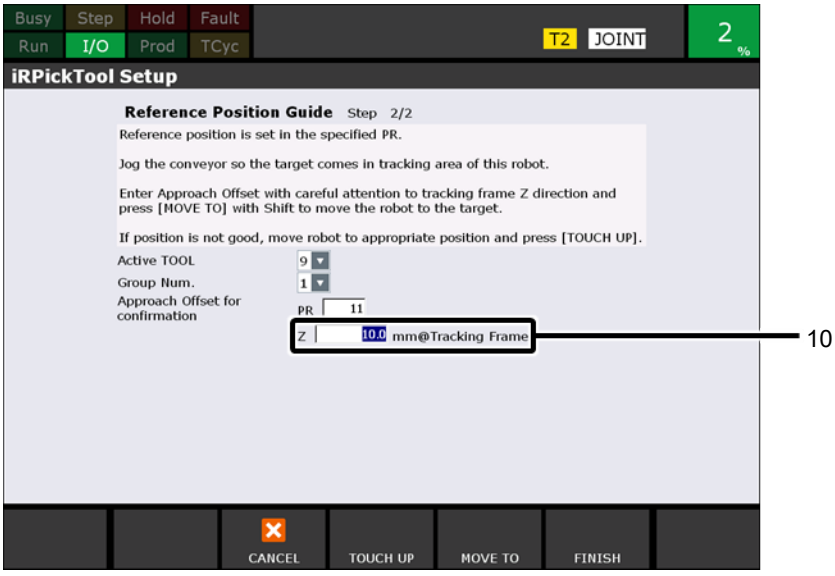
The found position is recorded to the position register, and [Reference Position Guide Step: 2/2] will appear.

9 Enter the position register number that is to be used as the offset from the pick position in [PR]. Enter 11 if you are going to use position register [11] as in the sample program (PT_PICKCS) given above as a pick program.



> **⚠ CAUTION**
> It is assumed that [PR Num. to Set] will be used as a fixed frame offset for a tracking frame.

10 In [Z], enter an approach offset for Z as viewed from the tracking frame. It is usually a positive value in the case of a line conveyor.



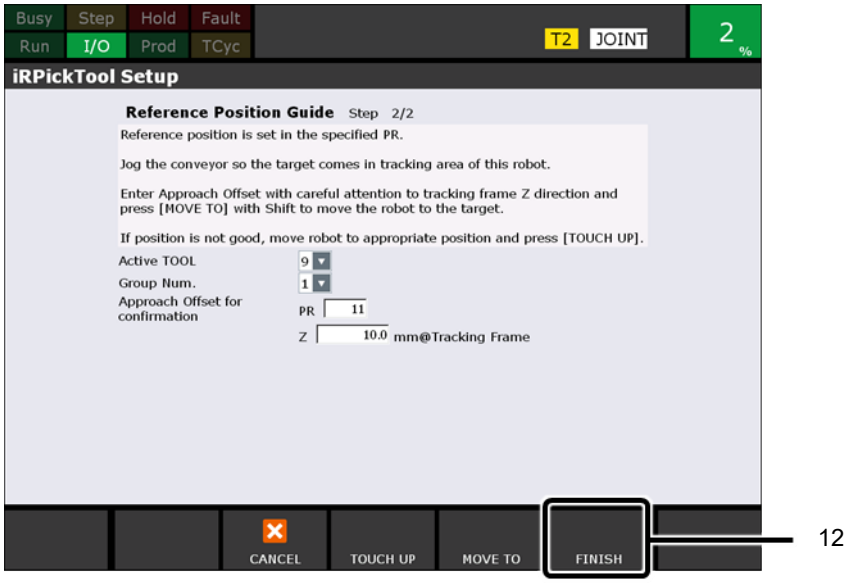
Move the conveyor until the part is directly in front of the robot.
Press F4 [MOVE TO] while holding down the [SHIFT] key.

> **⚠ CAUTION**
> Keep the speed override in the upper right of the screen at 10% or below, and work carefully.

11 The tip point of the suction cup will touch up the found position via the approach points.

NOTE
In order for the tip point to touch up an accurate position, the model origin position for the vision process and the tool frame must be set up correctly.
If the touch-up position is not correct, move the robot to an appropriate position by jog operation and press F3 [TOUCH UP].

12 Press F5 [FINISH].



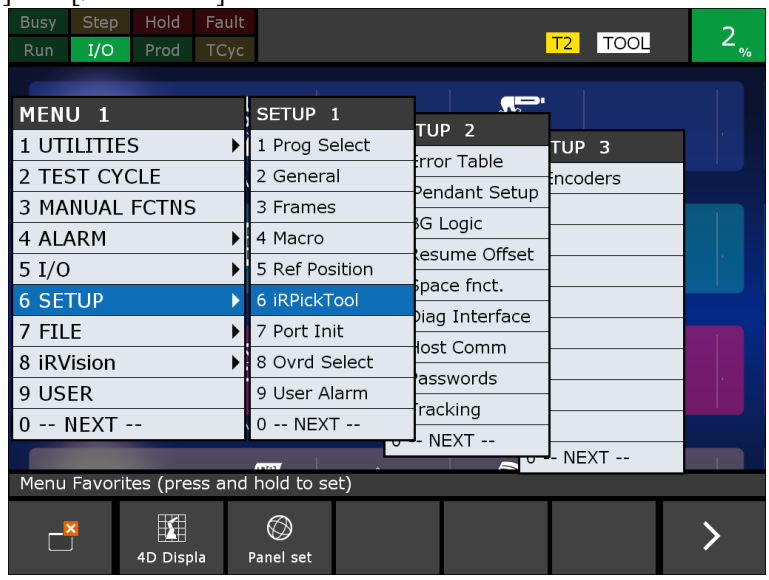
Setup of a reference position that uses a position register will finish.
[Guide Done] will be displayed on the [*i*RPickTool Setup] screen.

3.2.10.2 Setting up a reference position for the outfeed conveyor, and teaching a robot position

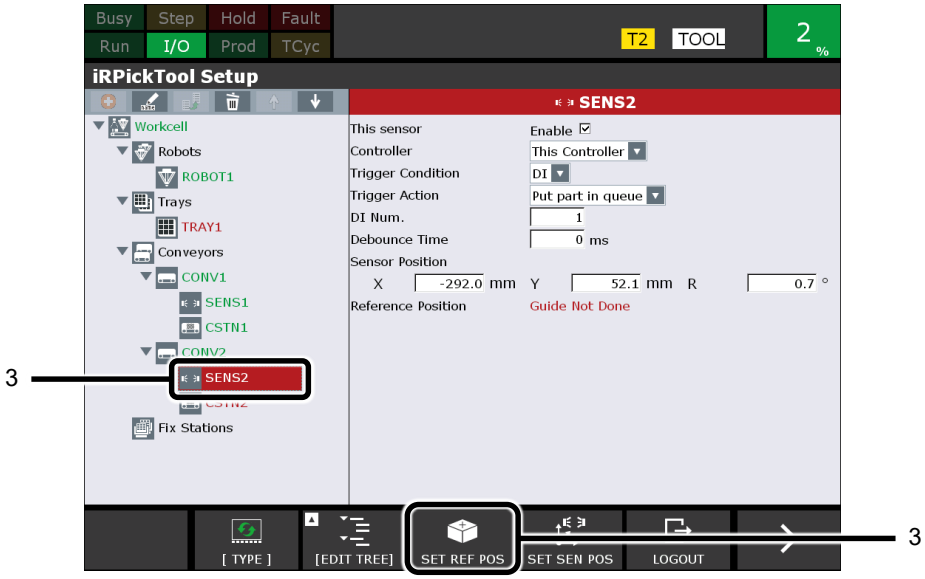
Set up a reference position for the outfeed conveyor, and teach a robot position. The procedure is for the case of [DI] / [RI] / [HDI] + [Put part in queue].

1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.

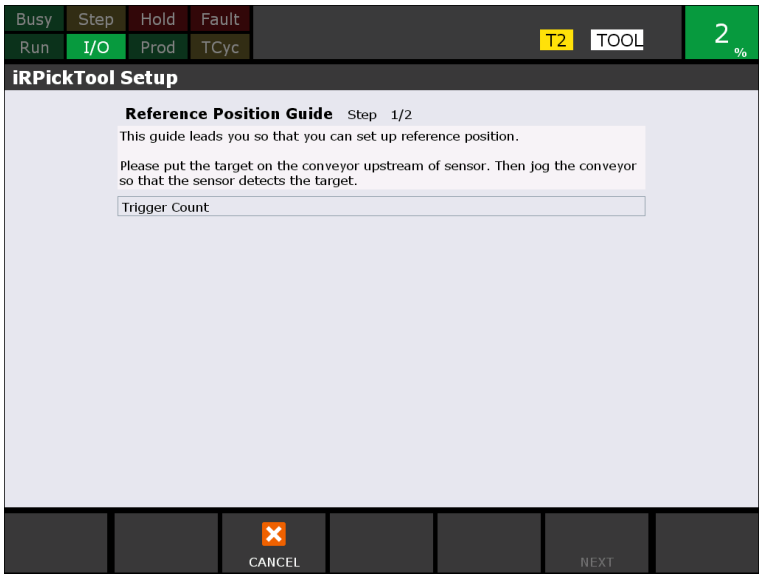
2 Select [SETUP] → [*i*RPickTool] from the menu.



3 Select the sensor [SENS2], then press F3 [SET REF POS].



A screen like the following one will appear.



4 Place a tray upstream of the trigger sensor.

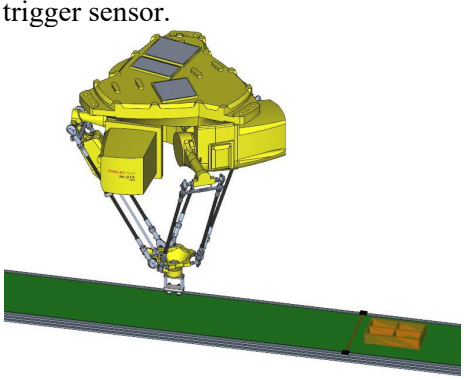
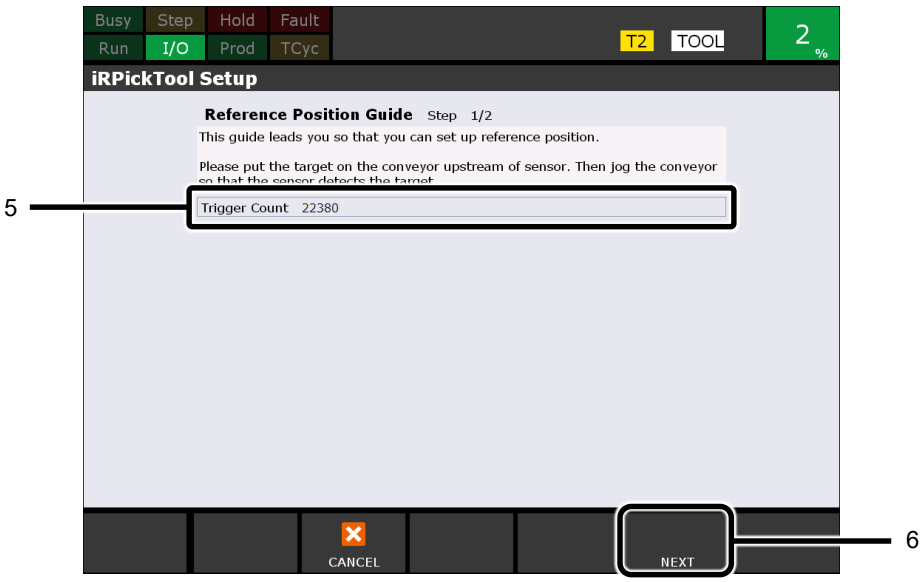


Fig. 3.2.10.2 (a) Placing a tray upstream of the trigger sensor.

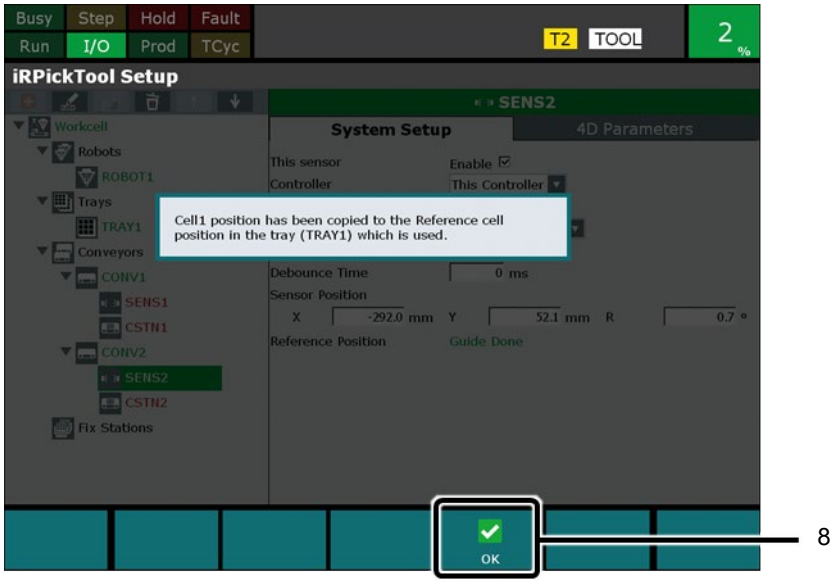
5 Move the conveyor. When the tray comes within the operating range of the robot, stop.
If detection is successful, [Trigger Count] will appear.

6 Press F5 [NEXT].
If detection is successful, F5 [NEXT] will become active.

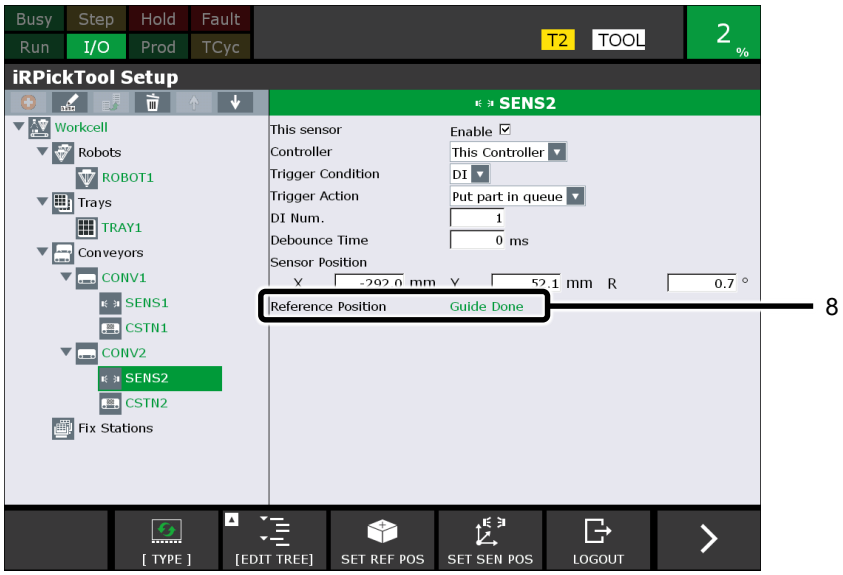


7 Press F5 [FINISH].

8 The following message will appear. Press F5 [OK].



A screen like the following one will appear.
Check that [Guide Done] is displayed for [Reference Position].



9 Move the conveyor. When the tray comes within the operating range of the robot, stop.
* Be careful that the tray does not become misaligned.

10 Jog the robot so that it moves to the drop position of Cell 1 on the tray.

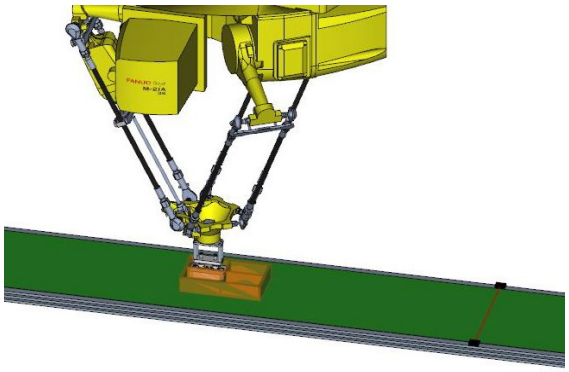


Fig. 3.2.10.2 (b) Moving the robot to the drop position of Cell 1 on the tray

11 Display the placement program (PT_DROPCS) and move the cursor to the line where the drop position is taught.

12 While holding down the [SHIFT] key, press F5 [TOUCHUP].
If F5 [TOUCHUP] is not displayed, press [> (Next page)] to switch the display.

13 A message saying 'VR is UNINIT, continue?' is displayed, but press F4 [YES] and teach the position.

NOTE

1 If there is a value in the vision register when you are setting it again, 'Subtract VR from current pos?' will be displayed. Press F5 [NO].

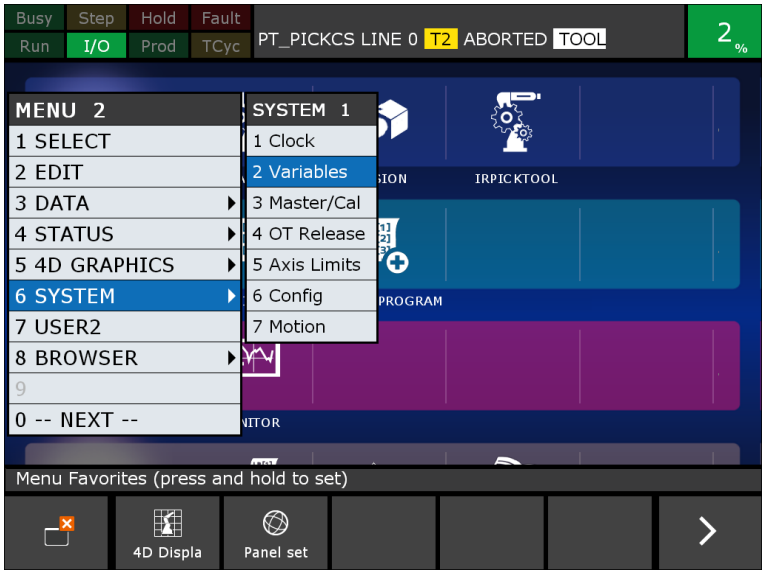
NOTE

2 If there is a value in the vision register when you are setting it again, then in order to avoid making a mistake, clear the vision register value by following the procedure below, and then teach the position. If you do this, the same position will be displayed whether you select F4 [YES] or F5 [NO] for 'Subtract VR from current pos?'

1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.

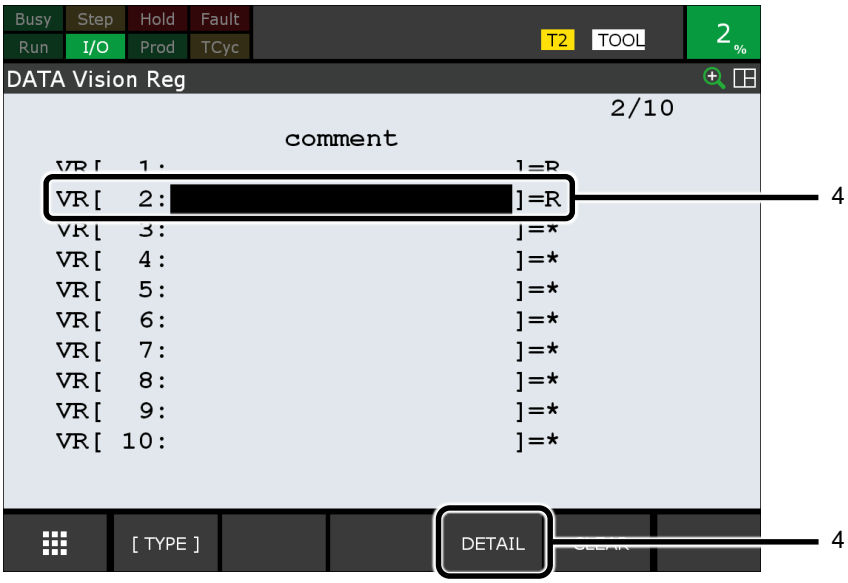
2 Select [NEXT] from the menu.

3 Select [DATA] → [Vision Reg].



NOTE

4 Place the cursor over '2' in [VR], then press F4 [DETAIL].



5 Enter '0' for all of the values [X], [Y], [Z], [W], [P] and [R] in [Offset].

3.2.11 Fine Adjustment of the Tracking Motion

In systems that require precision, fine adjustments are made to the tracking motion. Refer to Section 6.12, “FINE ADJUSTMENT OF THE TRACKING MOTION”.

4 *i*RPickTool (BASIC FUNCTIONS + PLUG & PLAY) STARTUP PROCEDURES

This chapter explains the startup procedures for *i*RPickTool (Basic Functions + Plug & Play).
By performing the procedures as explained in this chapter, you will complete the setup of systems that use four-axis Genkotsu robots as shown in the figure below.

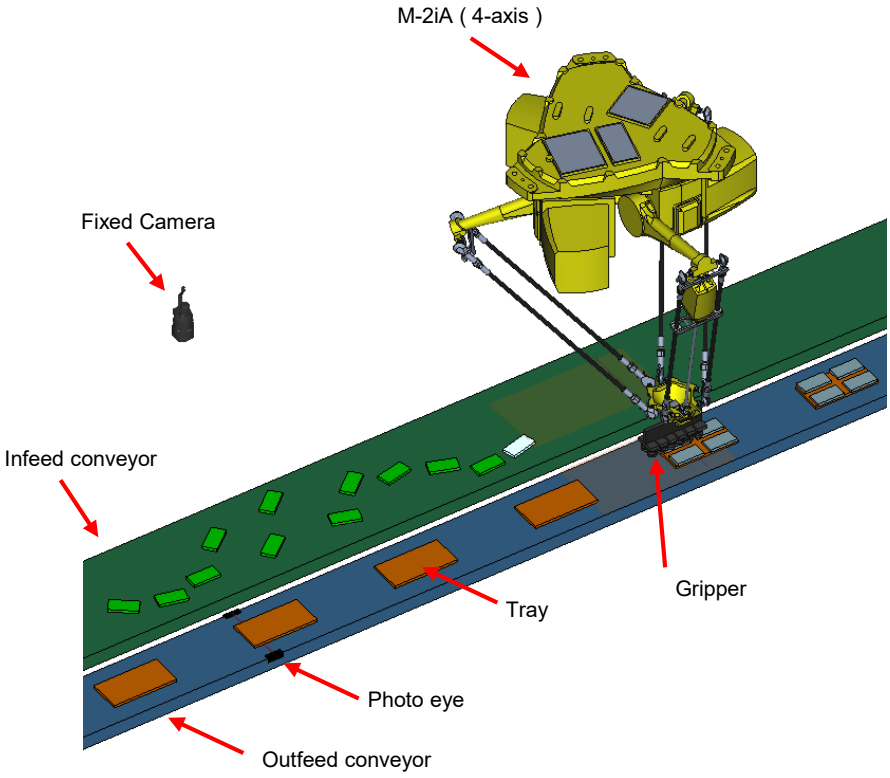


Fig. 4(a) System configuration

The procedure to start up the conveyor varies depending on the combination of the [Trigger Condition] for detection and the [Trigger Action] that is executed when the condition for the trigger is met. Please select the combination in accordance with your system.

This chapter explains the following combinations as system configuration examples.

Infeed : [Distance] + [Find part by vision]
Outfeed : [DI] / [RI] / [HDI] + [Put part in queue] (Use tray)

For information on the startup procedures when conveyors and fixed stations in other combinations are used and details on each tree item, refer to Chapter 6, “BASIC FUNCTIONS REFERENCE”.For the standard programs and objects for Plug & Play (gripper/zone/object), refer to Chapter7, “PLUG & PLAY REFERENCE”.

As standard programs are provided in Plug & Play, program creation is not required in the startup procedures for *i*RPickTool (Basic Functions + Plug & Play).

4.1 PREPARATION BEFORE SETTING UP THE APPLICATION

This section explains the preparation for the startup of tracking systems that use *i*RPickTool including the connection of hardware.

4.1.1 Connection and Setup of Pulsecoders

Install Pulsecoders and connect them to the robot controller.
After connecting, set parameters on the teach pendant of the robot controller.

NOTE
In this manual, a 'Pulsecoder' is a device that detects the travel distance of a conveyor.
In the setup screens of the teach pendant and *i*RPickTool, the term 'encoder' is used as a synonym for 'Pulsecoder.'

αA1000S Pulsecoders and incremental Pulsecoders can be used.

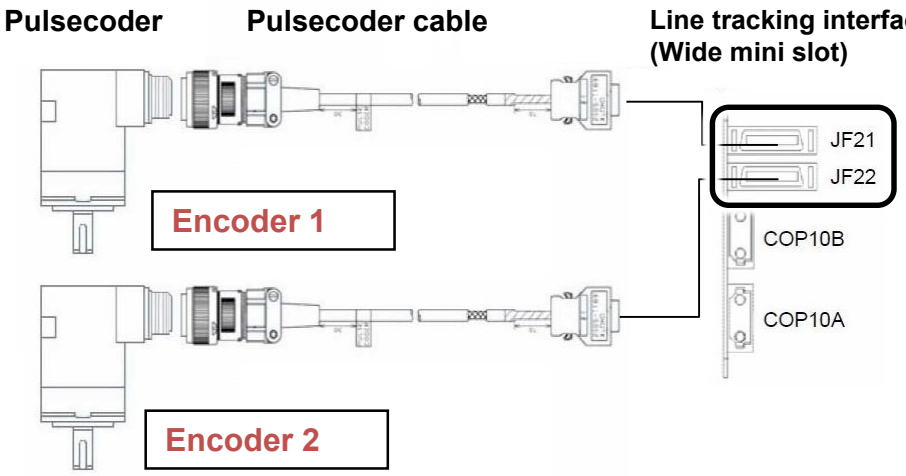
NOTE
You can use one Pulsecoder or three Pulsecoders by connecting an αA1000S Pulsecoder to the main board. Refer to Appendix E, “INCREMENTAL PULSECODER” for information about incremental Pulsecoder connection, and Subsection 5.2.3, “Setting Up Pulsecoders That Are Connected to the Main Board” for information about setup.

4.1.1.1 Connecting Pulsecoders

Connect Pulsecoders to the robot controller.

Connecting to interface board for line tracking

Use the following figure as a reference, and connect Pulsecoders so that the Pulsecoder that is installed at the infeed conveyor will be encoder No. 1 and the Pulsecoder that is installed at the outfeed conveyor will be encoder No. 2.



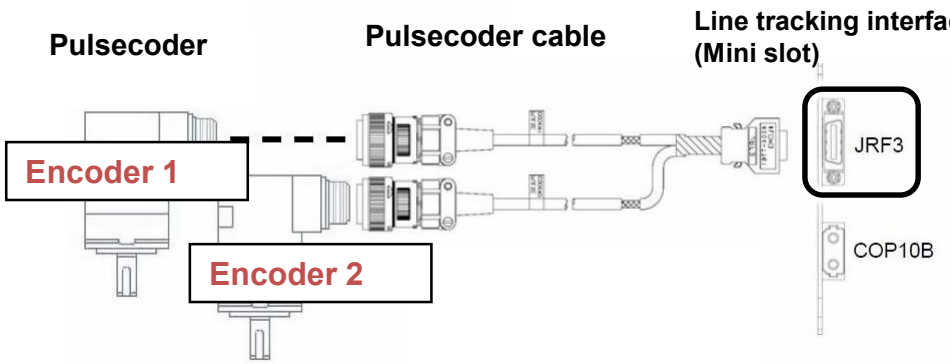


Fig. 4.1.1.1(a) Connection of interface board for line tracking and Pulsecoders

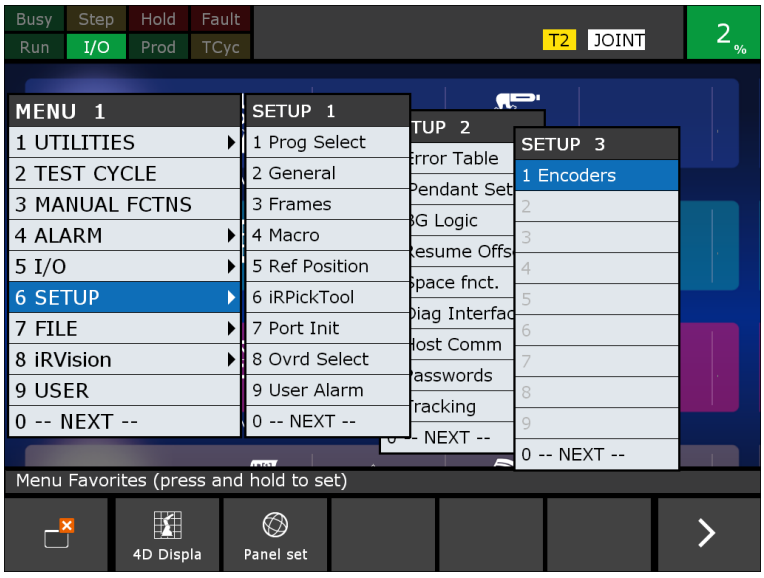
4.1.1.2 Setting up Pulsecoders

Set the following parameters for each Pulsecoder.

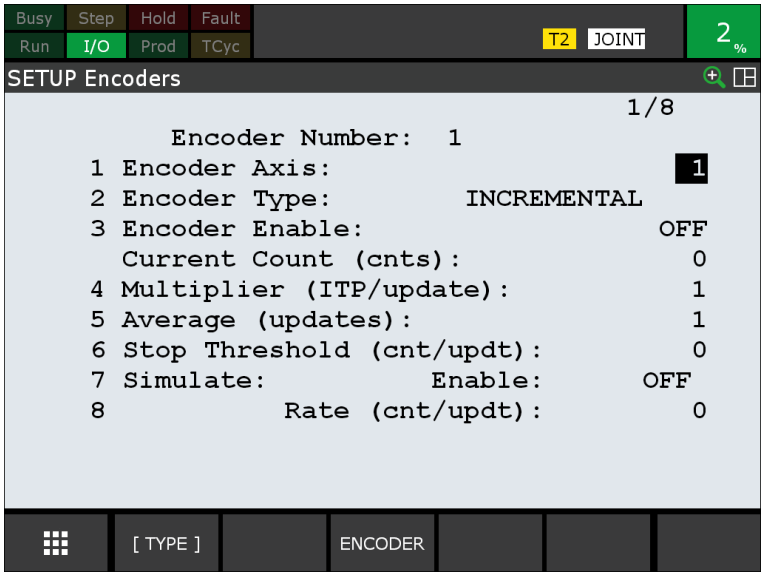
- [Encoder Axis]
- [Encoder Type]
- [Average (updates)]

1 Press the [MENU] key on the teach pendant of the robot controller.

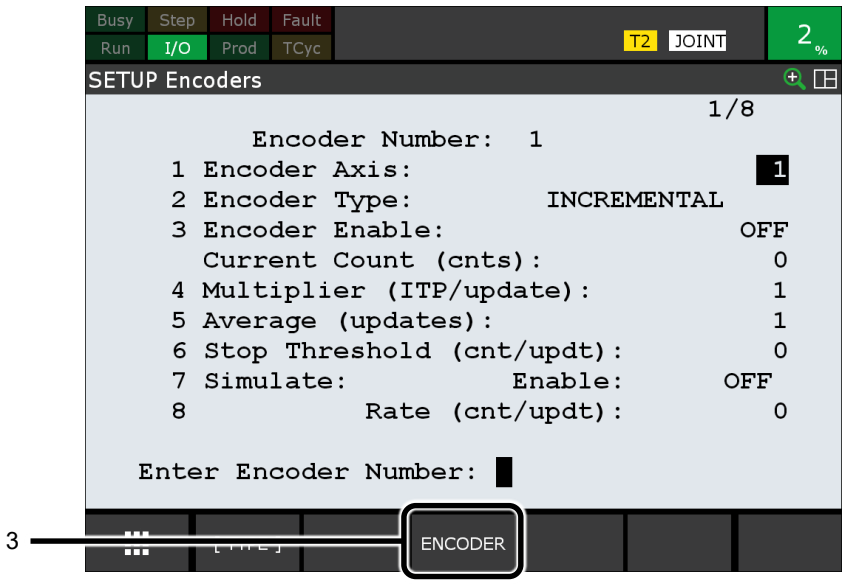
2 Select [SETUP] → [Encoders] from the menu.



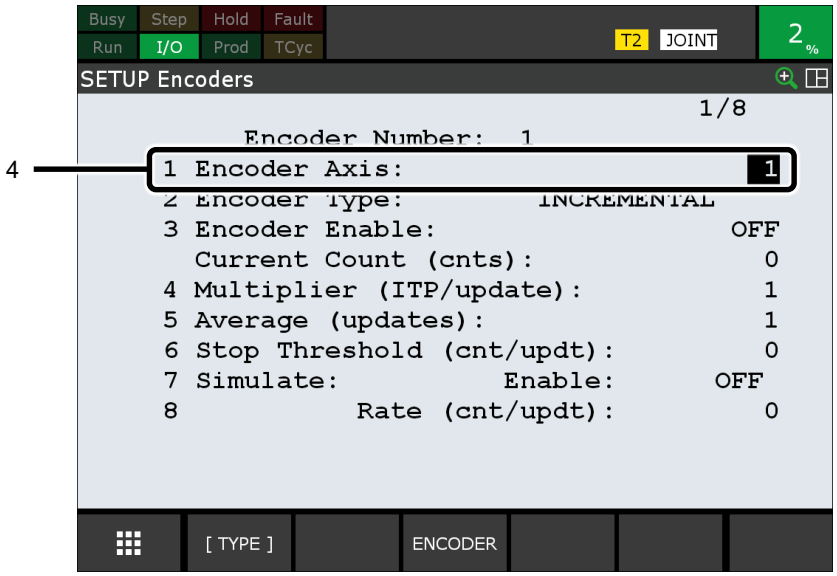
A screen similar to the one below will appear.
Number [9] and after in the setting items are the menu for optional functions. These will not be set here.



3 First, set the parameters for the Pulsecoder that is installed at the infeed conveyor. Check and confirm that [Encoder Number] is 1. If it is set to a value other than 1, press F3 [ENCODER] and input 1.

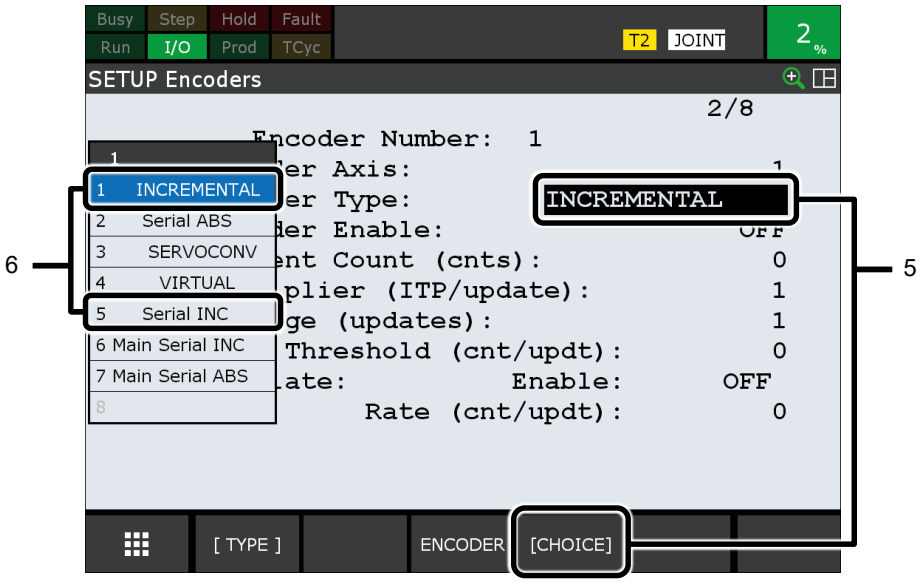


4 Move the cursor to [Encoder Axis] and input 1. However, if connecting to the main board, or for R-30*i*B Compact Plus / R-30*i*B Mini Plus , leave it 0 as the default value.



5 Move the cursor to [Encoder Type] and press F4 [CHOICE].
A menu will appear.

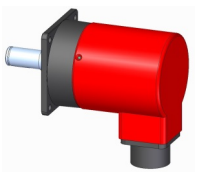
6 Select the encoder type that corresponds to the Pulsecoder that is to be connected.



Encoder type

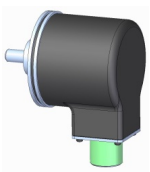
There are the following types for encoders.

- αA1000S Pulsecoder (red Pulsecoder): Serial INC
- Incremental Pulsecoder (black Pulsecoder): INCREMENTAL



αA 1000S (A860-0372-T001)
Encoder type: Serial INC

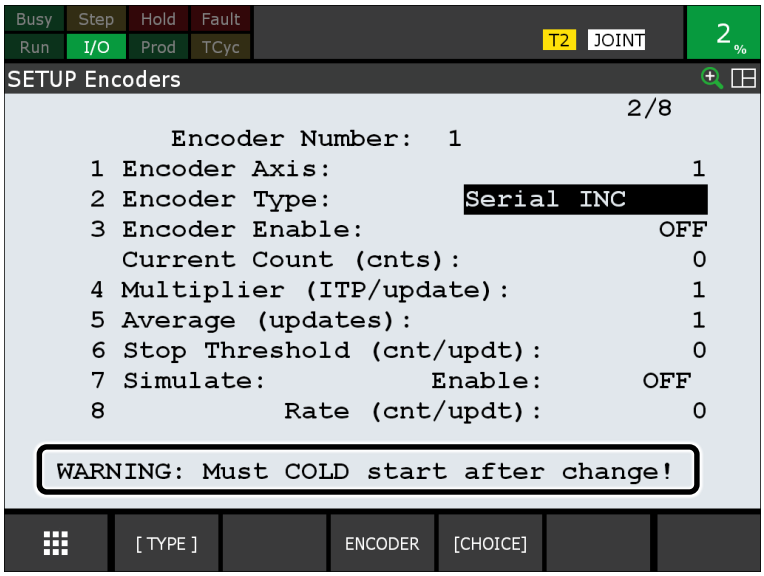
* If connecting to the main board, or for R-30*i*B Compact Plus / R-30*i*B Mini Plus,
Main Serial INC



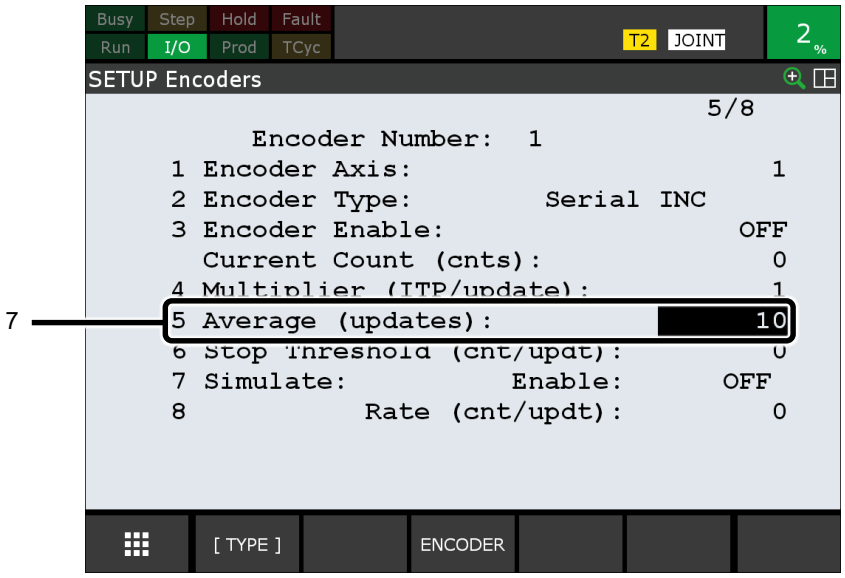
A860-0301-T001 to T004
Encoder type: INCREMENTAL

* Cannot be connected to the main board

If encoder axes or encoder types are changed, a warning prompting to cold start the robot will appear.

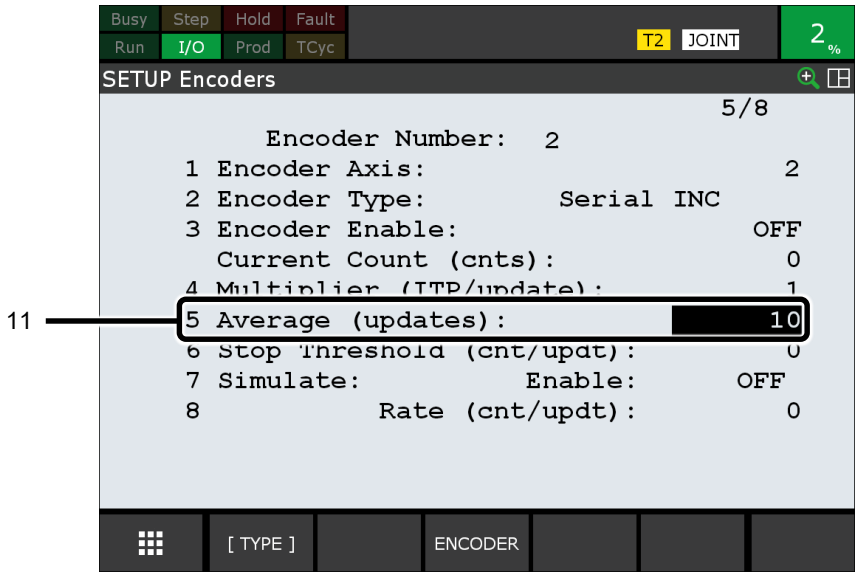


7 Move the cursor to [Average (updates)] and input the accumulated number of instantaneous speeds (width of moving average window).
By setting the accumulated number of instantaneous speeds, the robot will stop smoothly rather than suddenly, even if the conveyor suddenly stops while the robot is tracking.
Usually, enter '10.'



8 Next, set the parameters for the Pulsecoder that is installed at the outfeed conveyor. Press F3 [ENCODER], then input 2.
9 Move the cursor to [Encoder Axis] and input 2.
10 Move the cursor to [Encoder Type], press F4 [CHOICE], and select the encoder type that corresponds to the Pulsecoder that is to be connected.

11 Move the cursor to [Average (updates)] and input a value. This is the same as step 7. Usually, enter '10.'

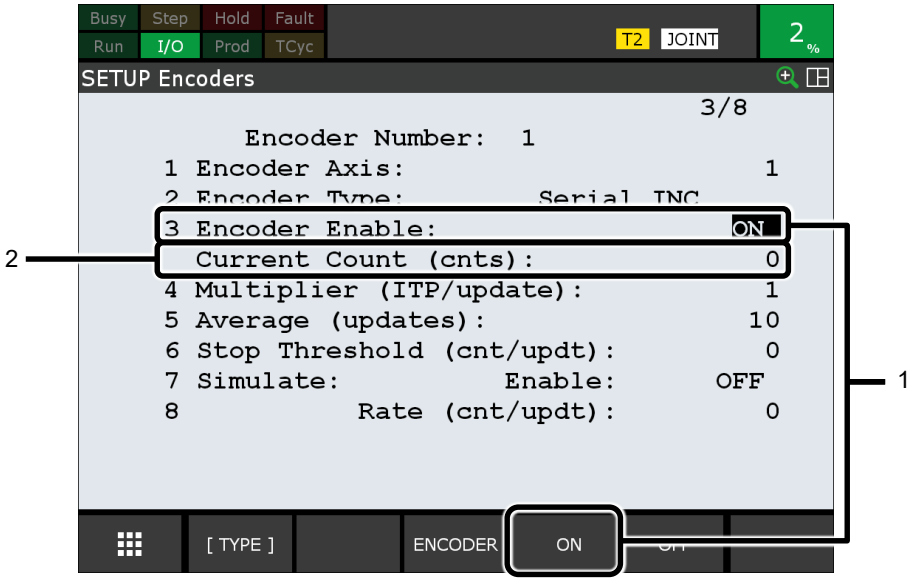


* When an HDI signal is used as an option, [9 HDI Port Num.] is displayed, so that the input port number that corresponds to the HDI signal can be assigned.

12 Finally, cold start the robot controller.

4.1.1.3 Checking the Pulsecoder settings

1 Open the [Encoders] screen and move the cursor to [Encoder Enable], then press F4 [ON].
2 Move the conveyor and confirm that the pulse count is displayed at [Current Count (cnts)].



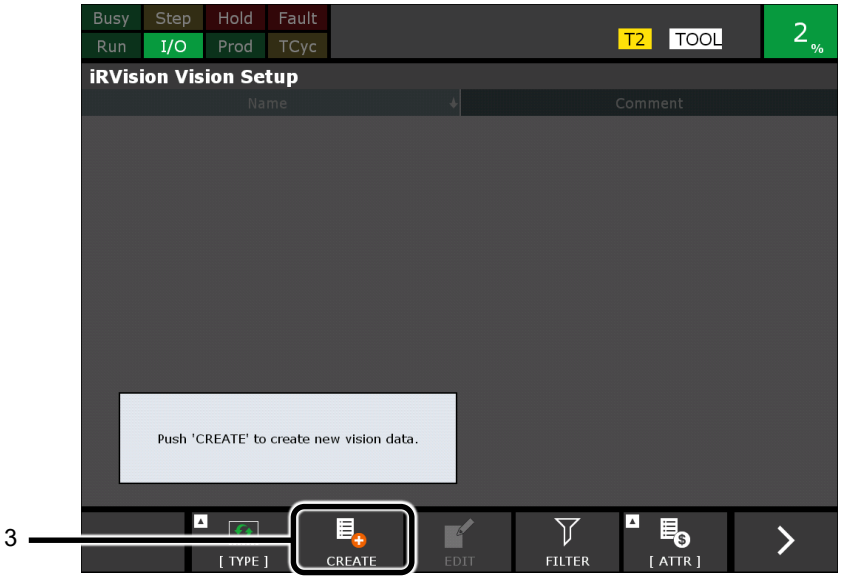
3 In the same manner, select Encoder Number: 2 for the Pulsecoder that is connected to the outfeed conveyor, then confirm.

4.1.2 Checking the Connection of the Camera

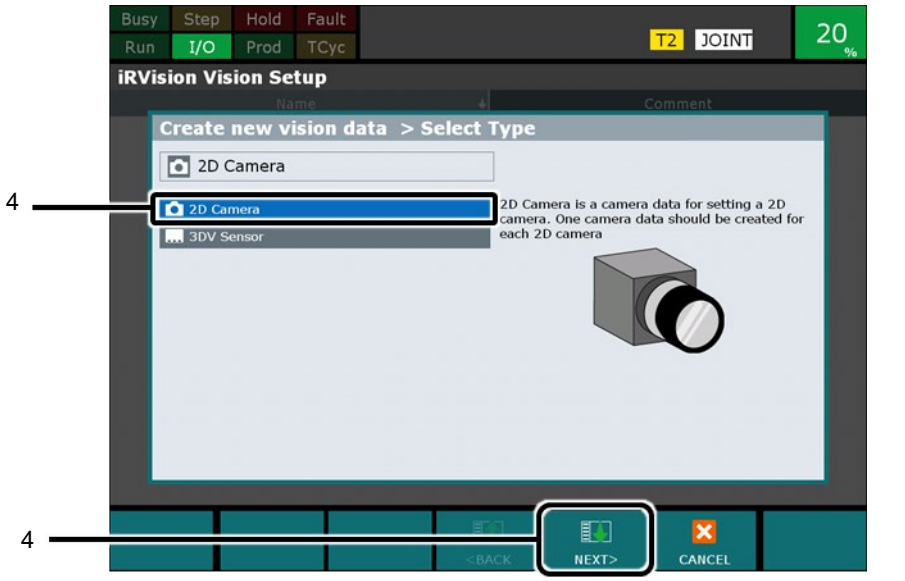
Check the connection of the camera that is installed above the infeed conveyor.

1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 From the menu, select [*i*RVision] → [Vision Setup].

The [*i*RVision Vision Setup] screen will appear.
3 Press F2 [CREATE].



4 Select [2D Camera] and press F4 [NEXT>].



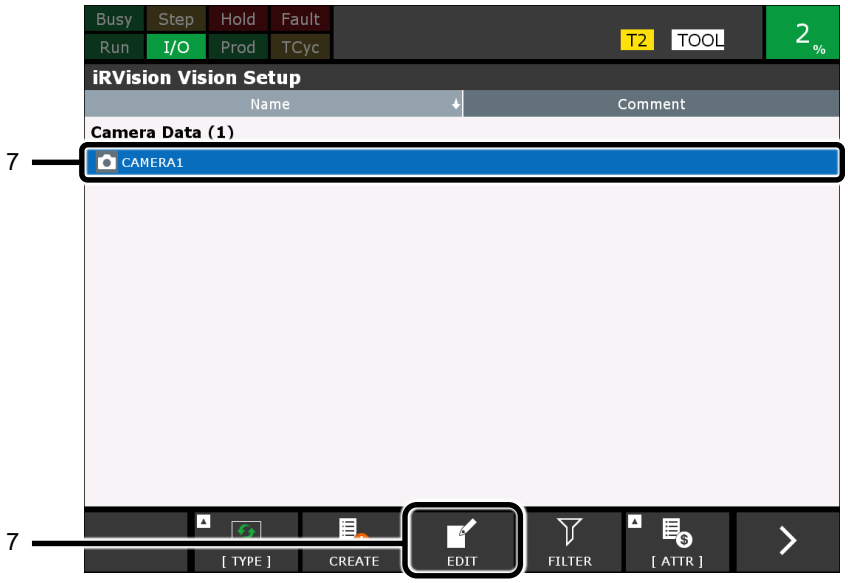
A screen similar to the one below will appear.

5 Input the name of the camera in [Name].
Input using half-width alphanumeric characters and half-width katakana. The name must contain no spaces, not contain the symbol other than underscore, and start with a letter.
Example: CAMERA1

6 Press F4 [OK].

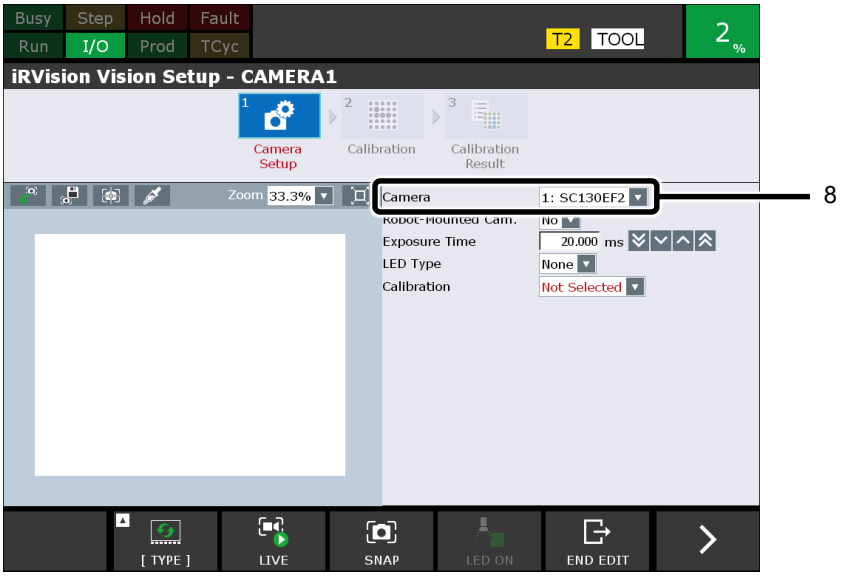
The created camera data will appear on the list.

7 Select the created camera data and press F3 [EDIT].



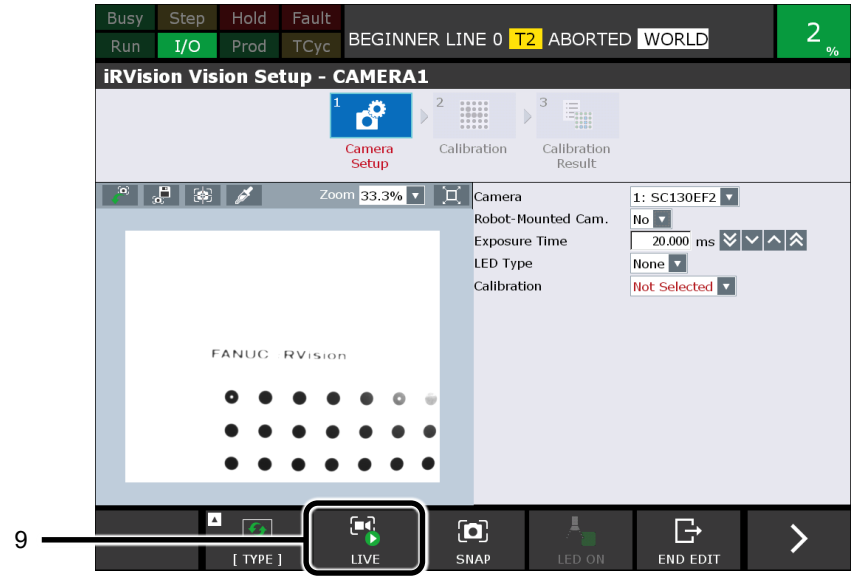
A screen similar to the one below will appear.

8 Select [Camera] from the menu.



9 To adjust the focus of the lens, place something that displays high contrast characters within the field of view of the camera on the conveyor surface, then press F2 [LIVE].
Confirm that the item that was placed has been captured within the frame on the left side of the screen.
* If the screen is dark, adjust the following.
- Open the diaphragm of the lens.
- Adjust the lighting.
- Adjust the exposure time.

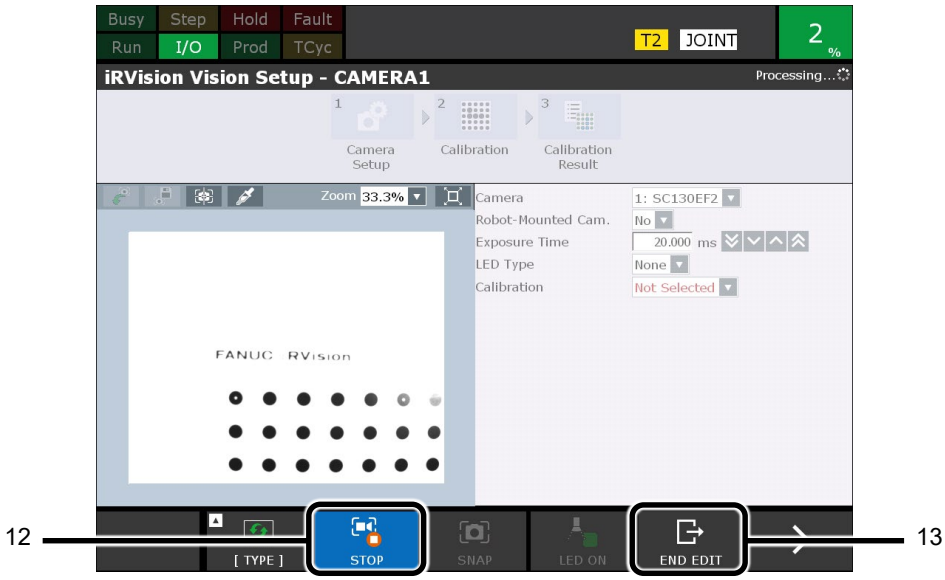
10 Check the field of view of the camera.
If the field of view is too wide or too narrow, adjust the position of the camera.



11 When the camera position has been set, adjust the focus of the lens.

Fig. 4.1.2(a) Adjusting the focus

12 Press F2 [STOP].
Live will stop.
13 Press F5 [END EDIT].
The edit screen will close.



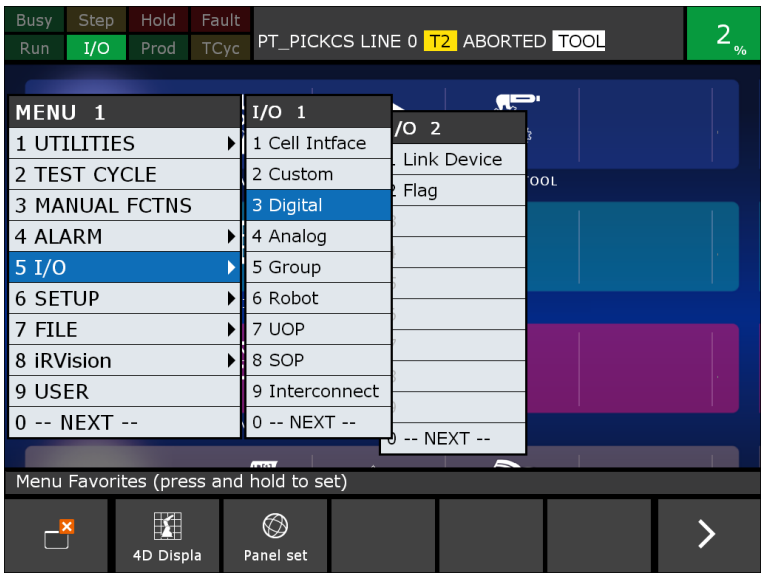
4.1.3 Checking the Input of the Trigger Sensor

Check the input of the trigger sensor.

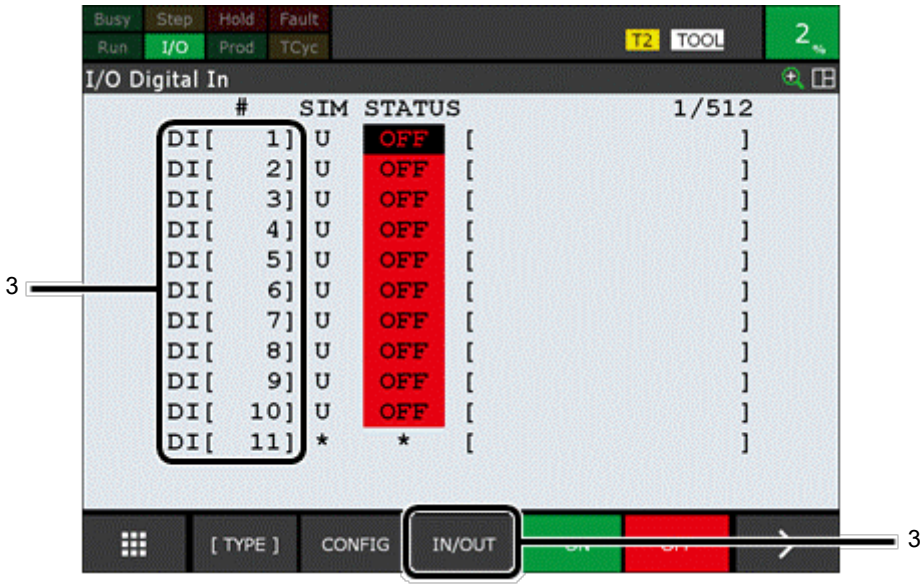
4.1.3.1 Using [DI] / [RI]

Check the input of the trigger sensors when DI / RI are used.

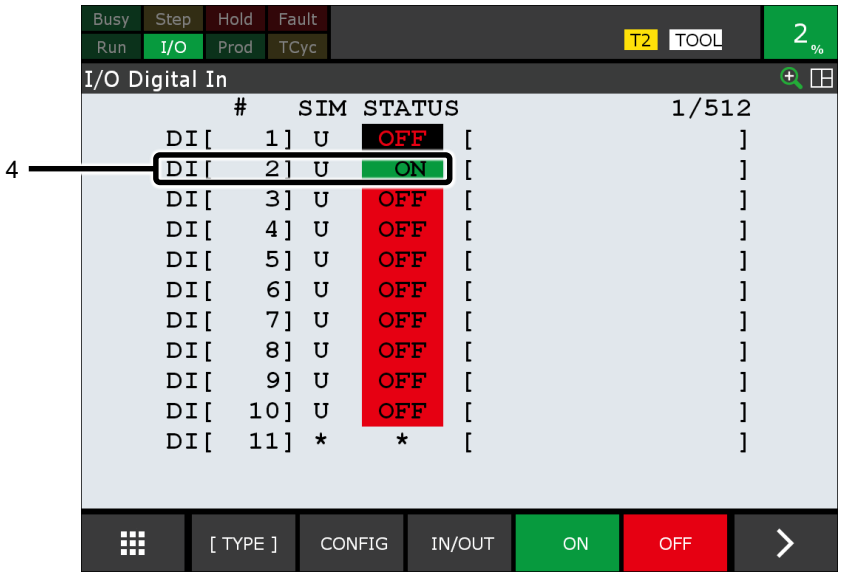
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 If you are using DI, select [I/O] → [Digital] from the menu. If you are using RI, select [I/O] → [Robot] from the menu.
For example, the screen is the situation when DI is used will look like this.



3 Press F3 [IN/OUT] and confirm that DI (RI) appears.



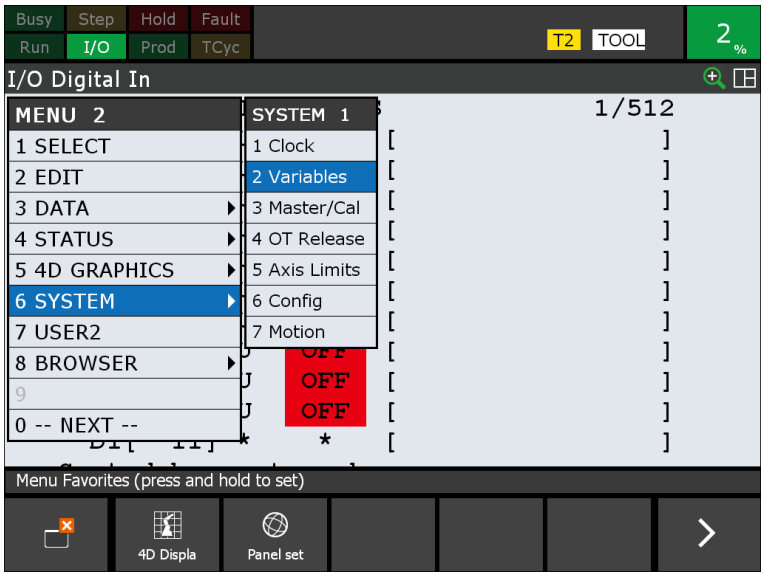
4 Confirm that the DI (or RI) changes to ON with input from the sensor, such as when a part passes through a photo-eye.



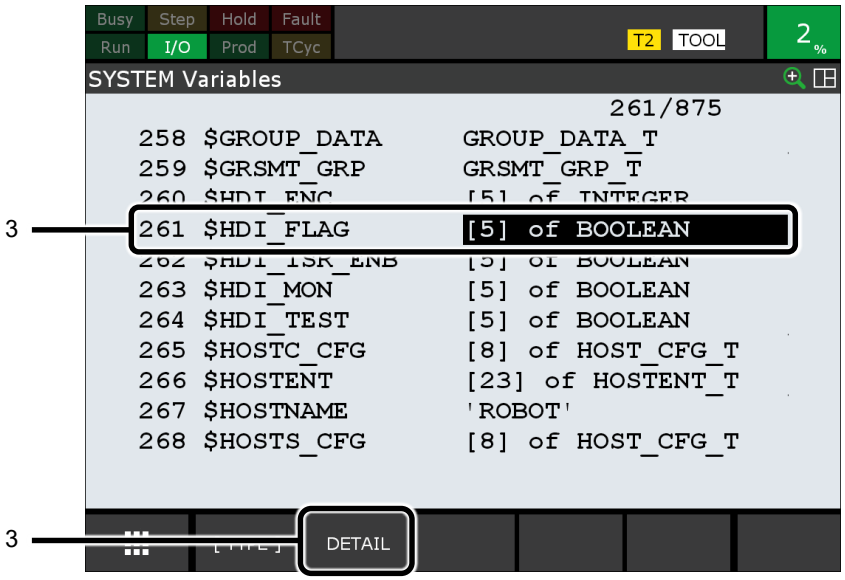
4.1.3.2 Using [HDI]

Check the input of the trigger sensors when HDI is used.

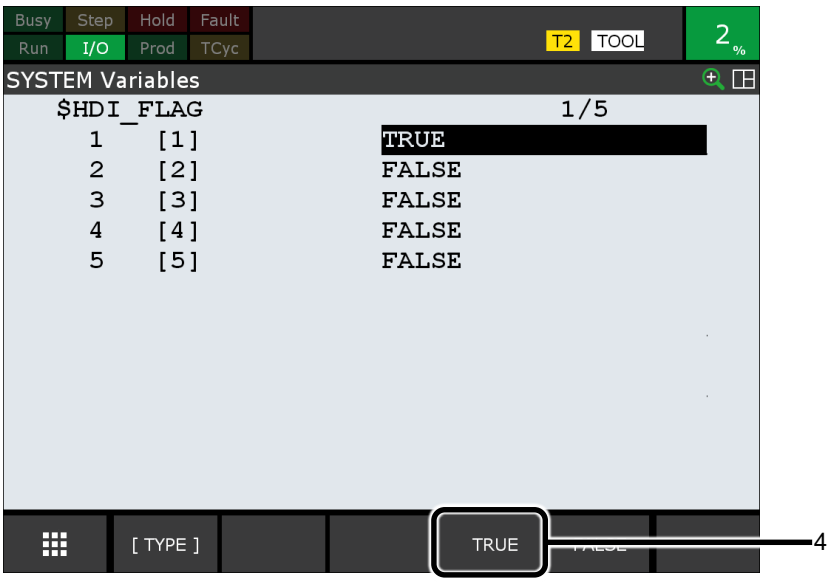
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [NEXT] → [SYSTEM] → [Variables].



3 Place the cursor over '$HDI_FLAG' and press F2 [DETAIL].

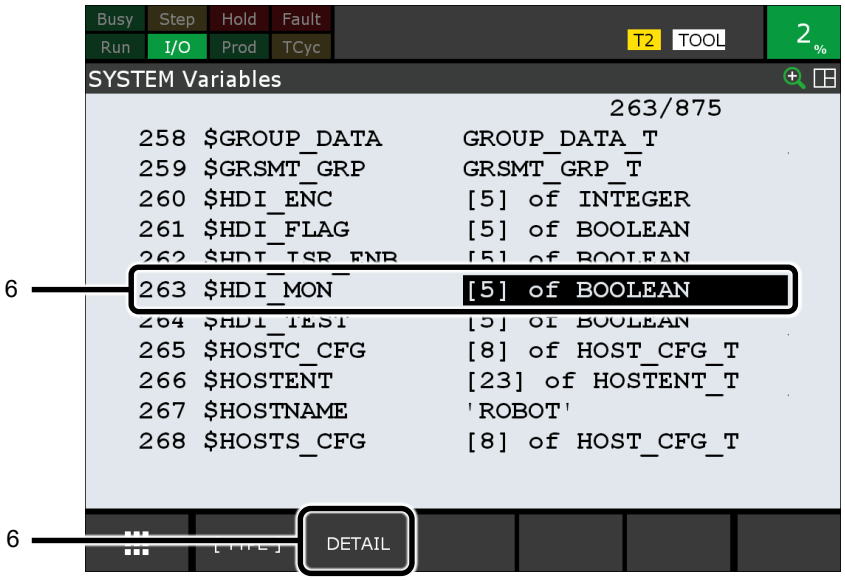


4 Move the cursor to the HDI port number to check and press F4 [TRUE].

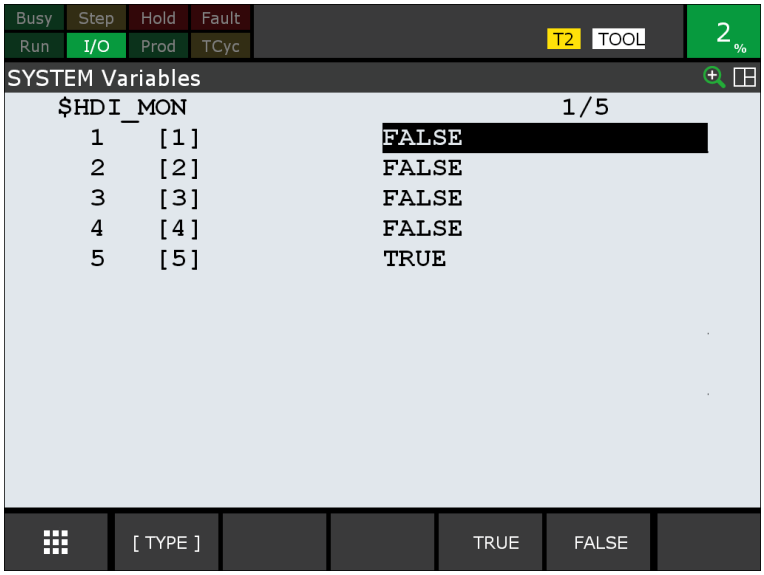


5 Press the [PREV] key on the teach pendant of the robot controller.
The [SYSTEM Variables] screen will appear.

6 Move the cursor to '$HDI_MON' and press F2 [DETAIL].

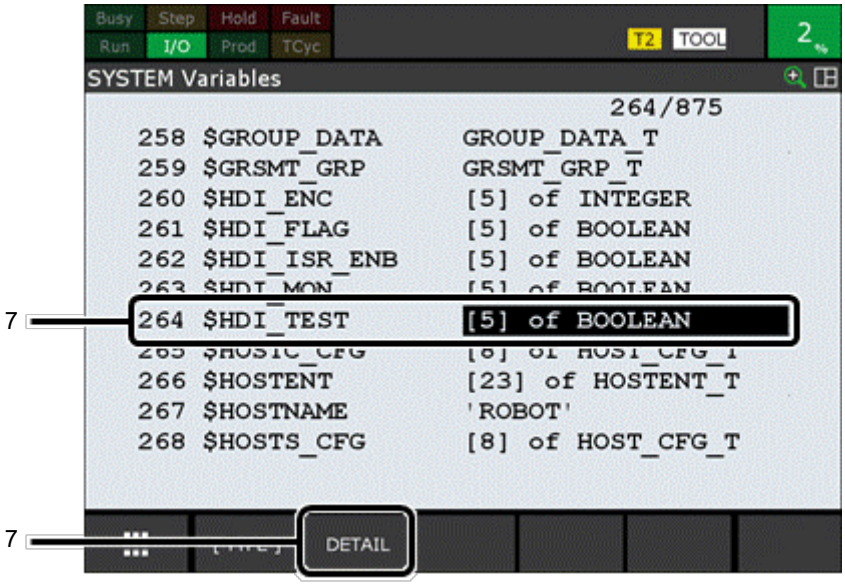


Check the variable that corresponds to the port number that is connected to the sensor.
Input ON : TRUE
Input OFF : FALSE

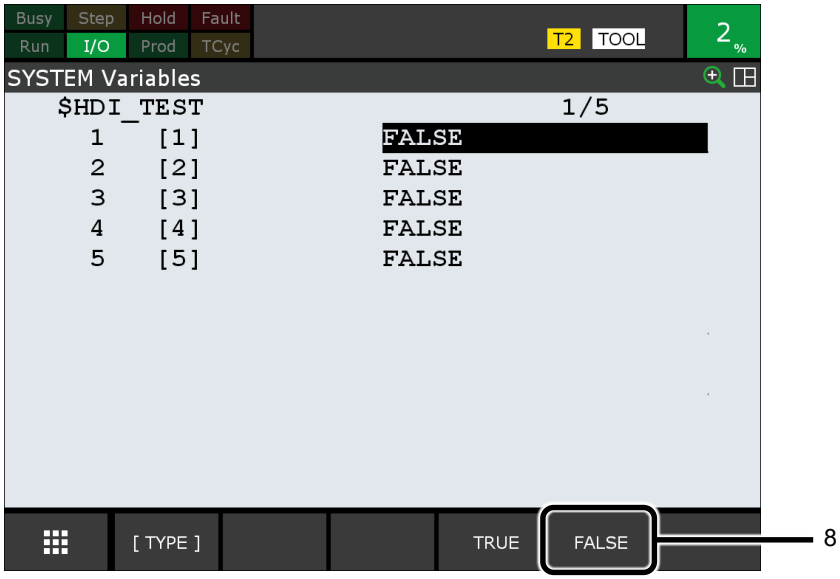


Due to a short input duration, if you are unable to confirm the functionality of the HDI using the step 7, check it using the following method.

7 Move the cursor to '$HDI_TEST' and press F2 [DETAIL].



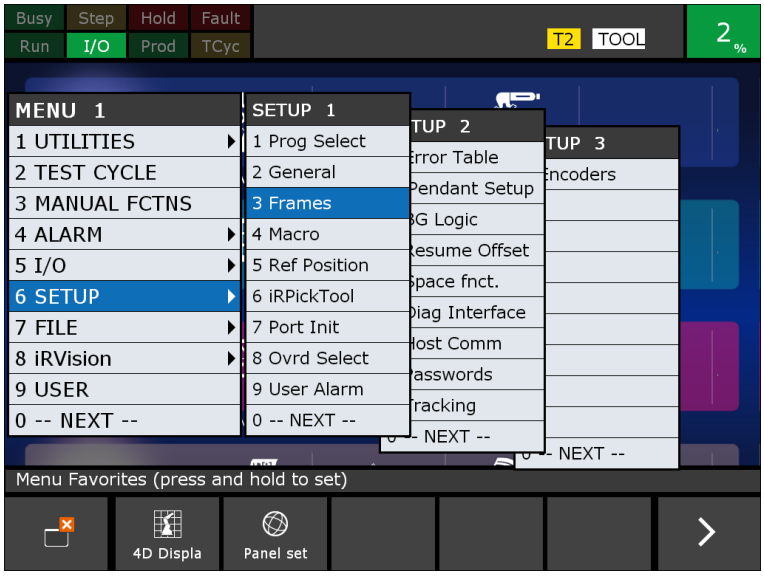
8 Move the cursor over the HDI port number to check and press F5 [FALSE].
Once the input turns ON, it changes to TRUE and retains the TRUE state even if the input turns OFF.



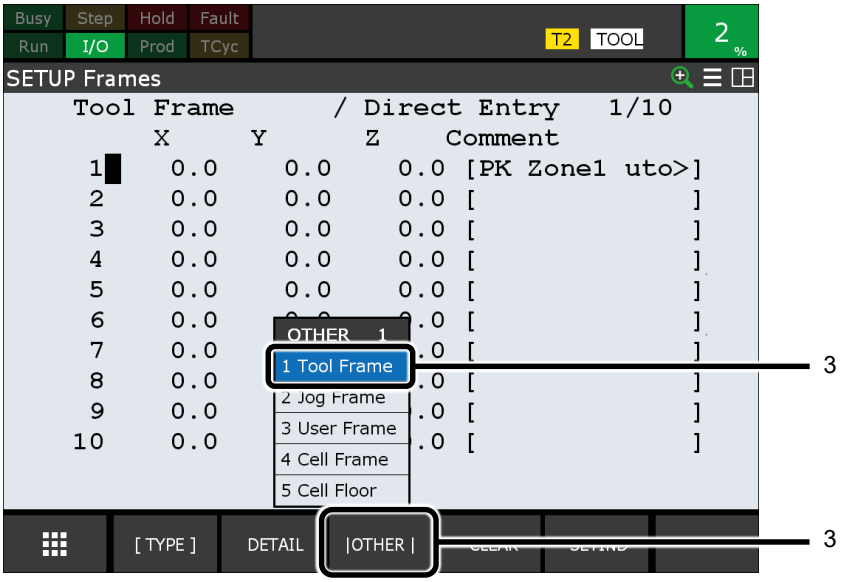
4.1.4 Setting Up the Tool Frame of the Pointer Tool

Set up the tool frame of the pointer tool.

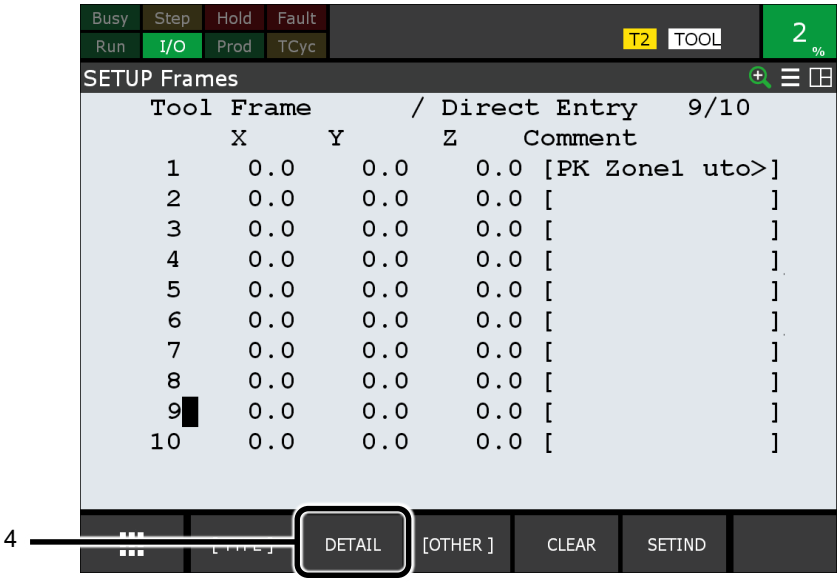
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SETUP] → [Frames] from the menu.



3 If frames other than the tool frames are displayed, press F3 [OTHER] and select [Tool Frame] from the menu.



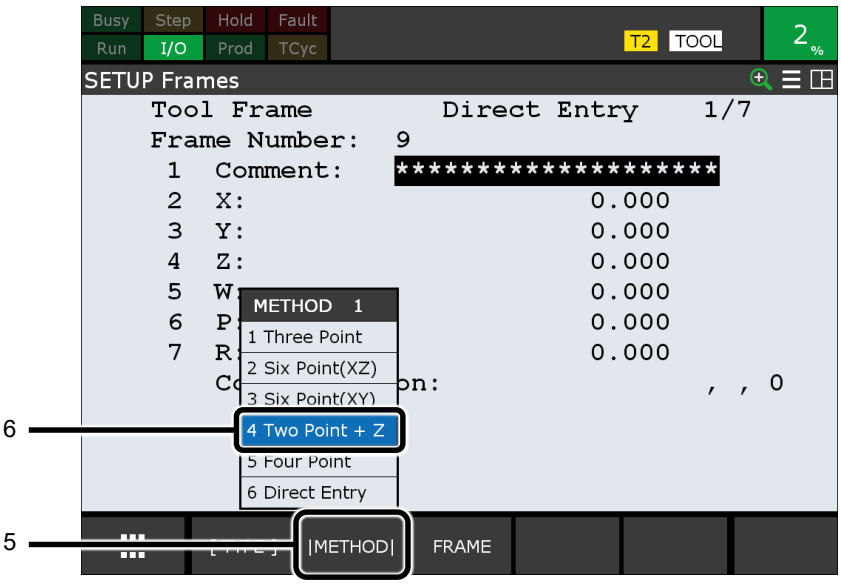
4 Move the cursor to the line of the tool frame number to set and press F2 [DETAIL].
The setup screen for the tool frame for the selected frame number will appear.
The screen is an example of when performing setup using Tool 9.



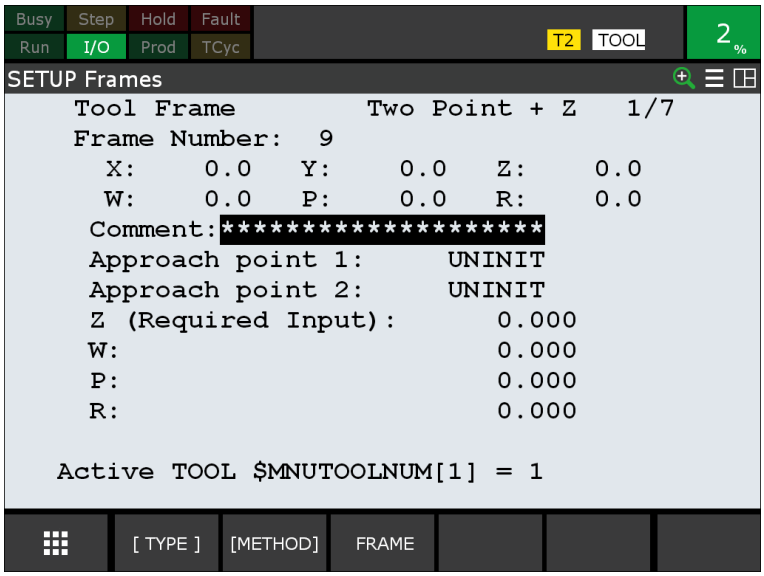
⚠ CAUTION
With the Plug & Pay standard programs, if you are using a multiple-item picking hand, use as many tool frames as the number of parts that can be gripped in order starting from frame number 1. To prevent tool frames from being overwritten, set the tool frame for the pointer tool to be an unused tool frame number.

5 Press F2 [METHOD].
A menu will appear.

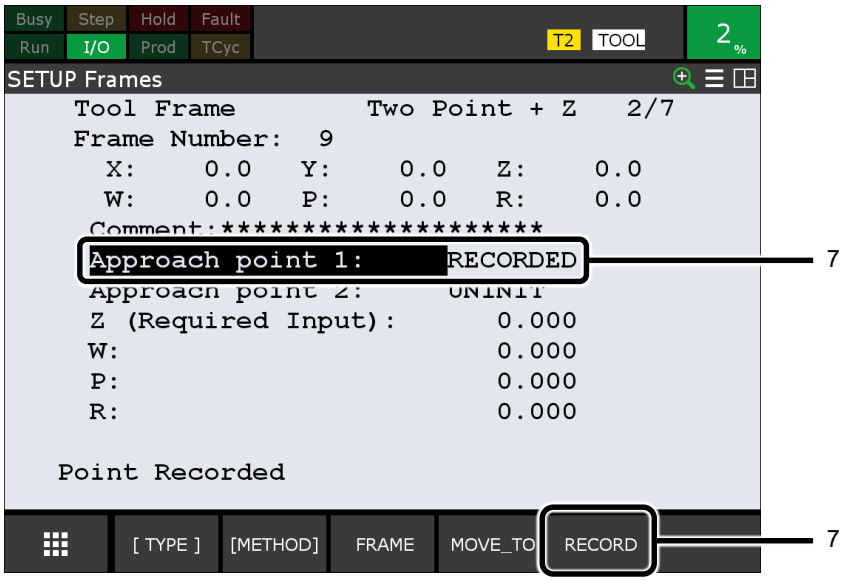
6 Select [Two Point + Z] from the menu.



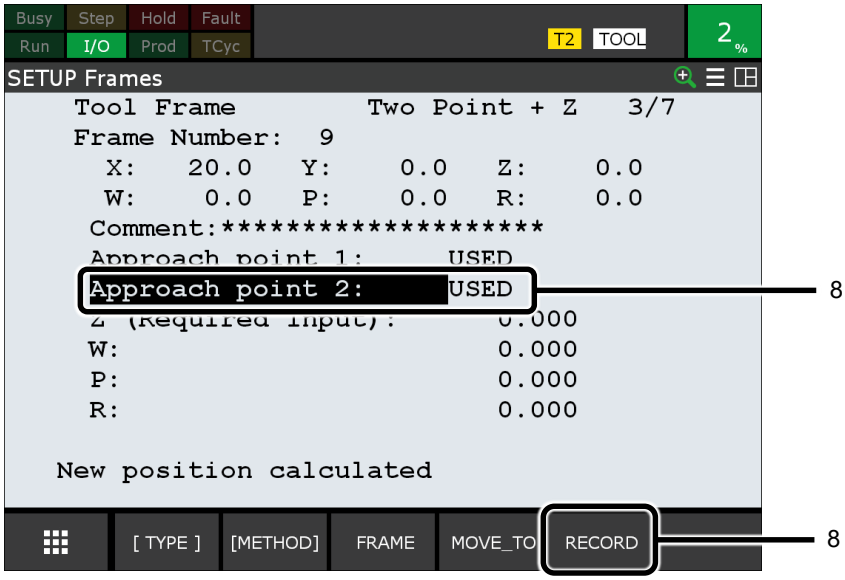
A screen similar to the one below will appear.



7 Move the cursor to [Approach point 1].
Jog the robot to the approach point to be recorded.
While holding down the [SHIFT] key on the teach pendant, press F5 [RECORD] to record the current position as the approach point.
'RECORDED' will be displayed.



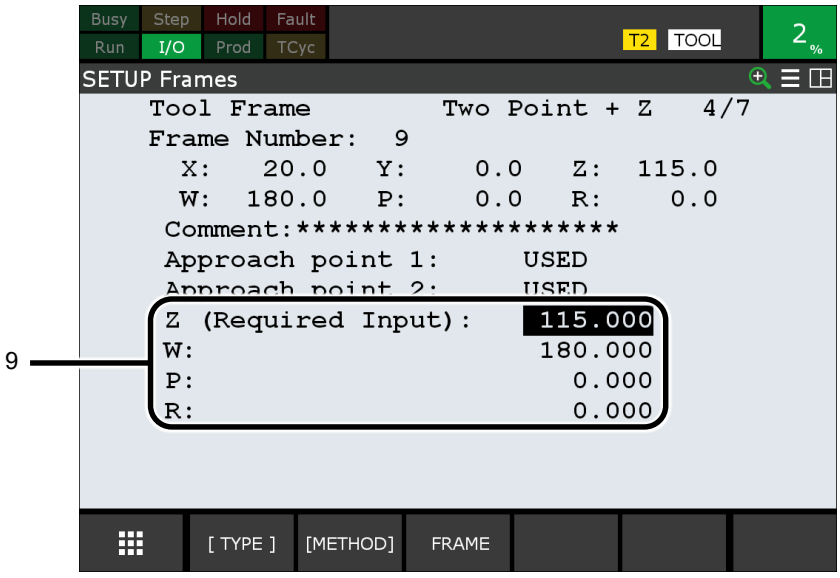
8 Move the cursor to [Approach point 2]
Jog the robot to the approach point to be recorded.
While holding down the [SHIFT] key on the teach pendant, press F5 [RECORD] to record the current position as the approach point.
When all the approach points are input, 'USED' will be displayed and the X and Y of the tool frame will be set.



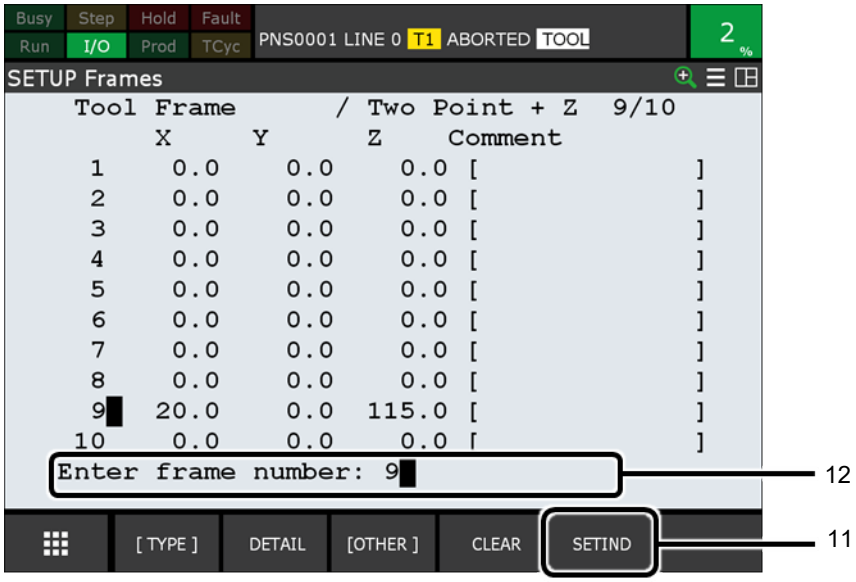
⚠ CAUTION

1 When inputting approach points, have the flange surface facing down.
2 Input approach points 1 and 2 at different positions.
3 If the above conditions are not met, a message saying 'Invalid set of input points' will appear.

9 Move the cursor to [Z], [W], [P] and [R] and enter a value for each.
For [Z], input a numerical value measured with a ruler, etc.
Input [W], [P] and [R] directly.
* (W, P, R) = (180, 0, 0) is recommended.



10 Press the [PREV] key on the teach pendant of the robot controller.
The list screen for tool frames will appear.
The values of all the tool frames can be verified on this screen.
11 Press F5 [SETIND] to change the active tool frame.
12 Input the frame number of the pointer tool frame.

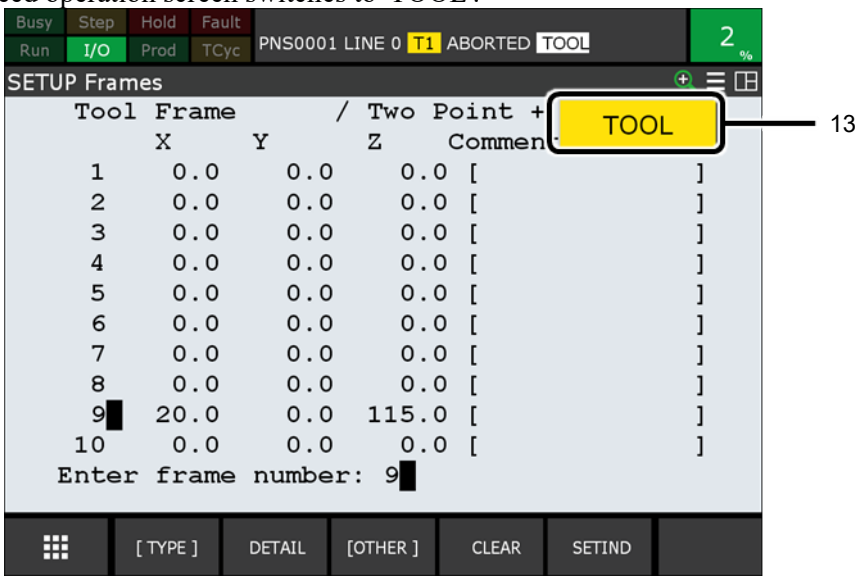


The tool frame will be enabled.

* If you do not to carry out steps 11 and 12, the frame that has been set up will not be active.

13 Set the manual continuous feed of the robot in 'TOOL.'

Press and hold the [COORD] key on the teach pendant of the robot controller until the manual continuous feed operation screen switches to 'TOOL'.



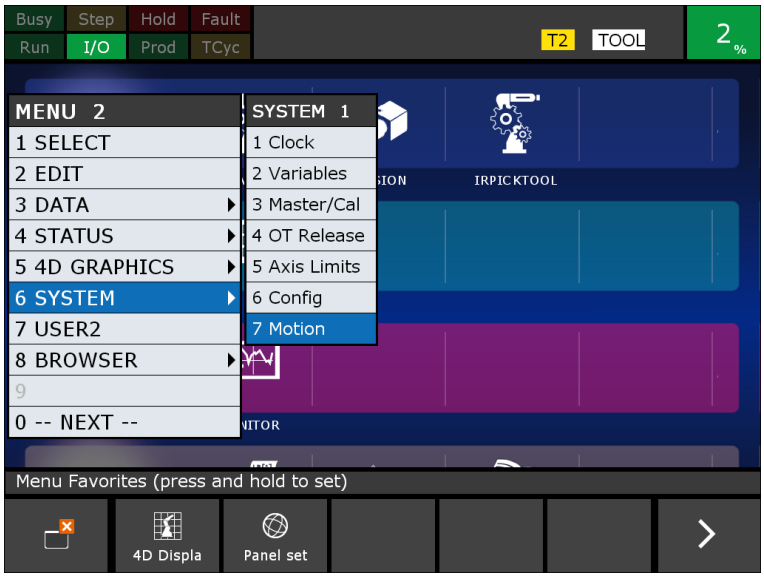
14 Press the -Z (J6) key or +Z (J6) key on the teach pendant of the robot controller to jog the robot around the Z axis(vertical).

Verify that the tool frame (TCP) has been set accurately. The pointer should not drift as the robot rotates around the Z axis.

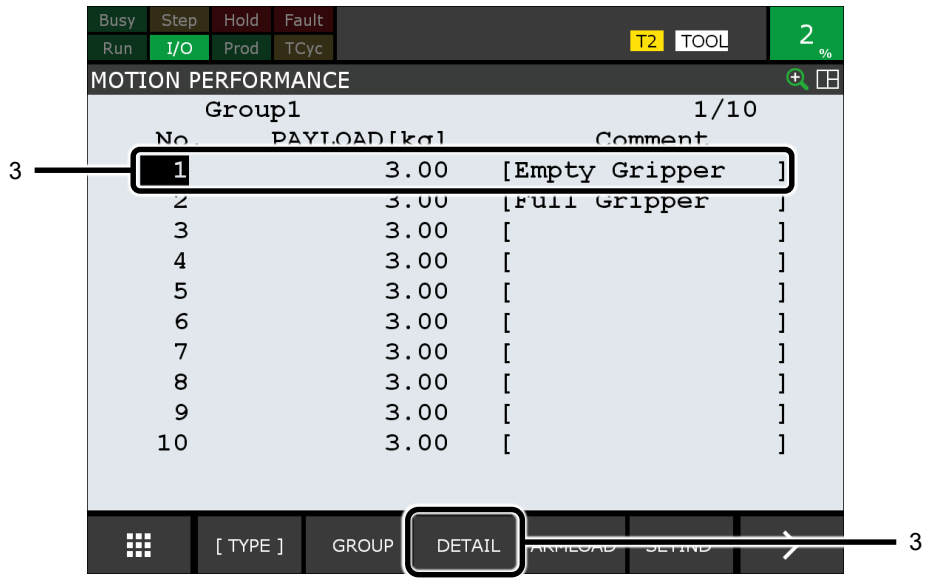
4.1.5 Setting Payloads (Motion performance)

Set the payloads for the robot on the [MOTION PERFORMANCE] screen.

1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SYSTEM] → [Motion].

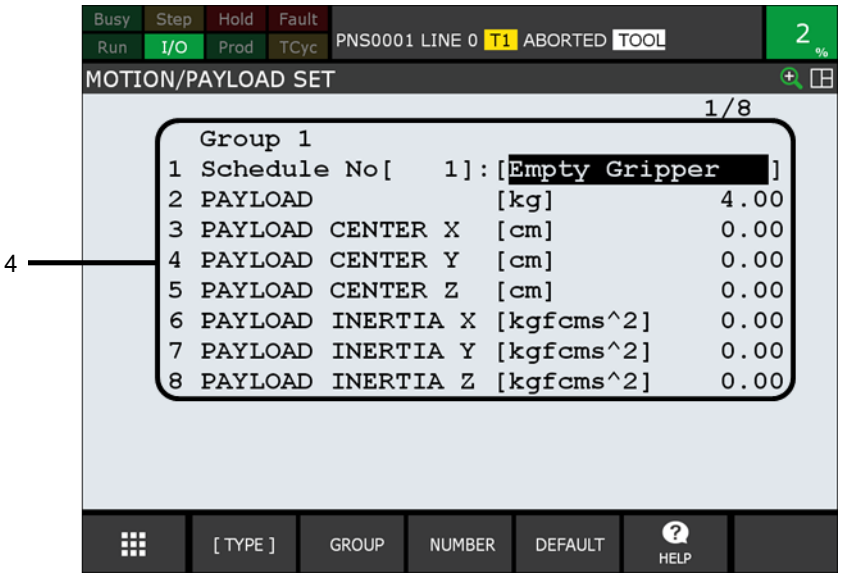


3 Move the cursor to [1] and press F3 [DETAIL].

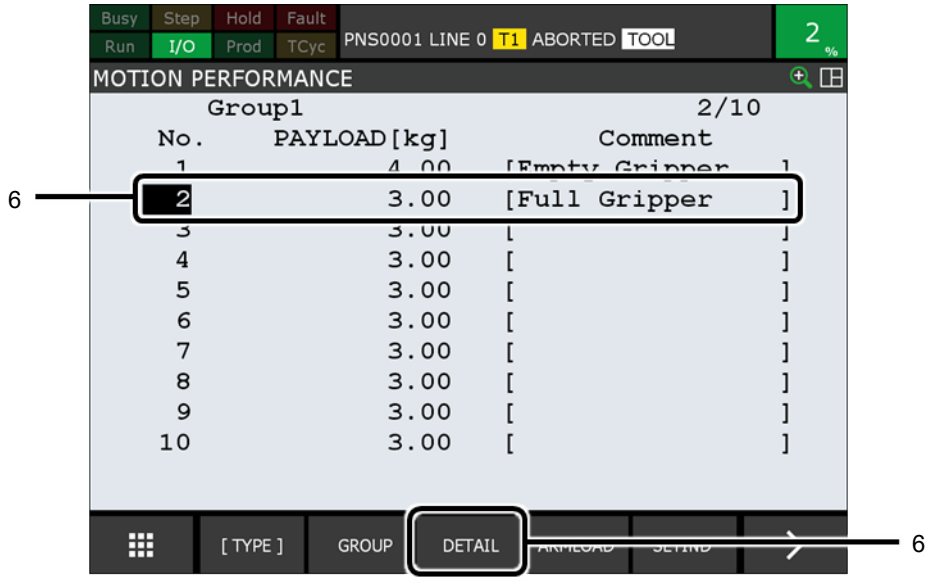


The setup screen for payload condition No.1 will appear.

4 Input the [PAYLOAD]. Not all of the items necessarily need to be input.
* Set a 'Hand only' payload for schedule no. 1

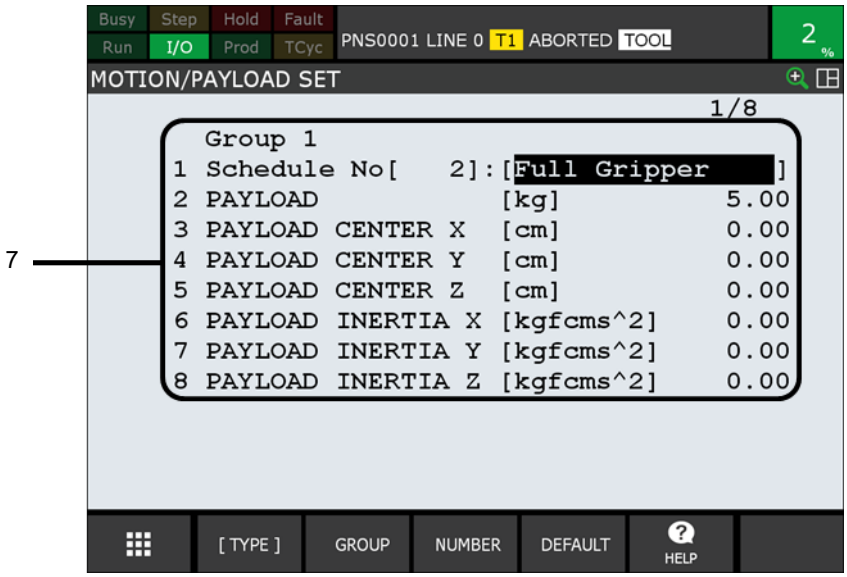


5 Press the [PREV] key on the teach pendant of the robot controller to return to the list on the motion performance screen.
6 Move the cursor to [2] and press F3 [DETAIL].



The setup screen for payload condition No.2 will appear.

7 Input the [PAYLOAD]. Not all of the items necessarily need to be input.
* Set a 'Hand + part' payload



With a standard TP program, use the Set Payload [*] command to change the payload number

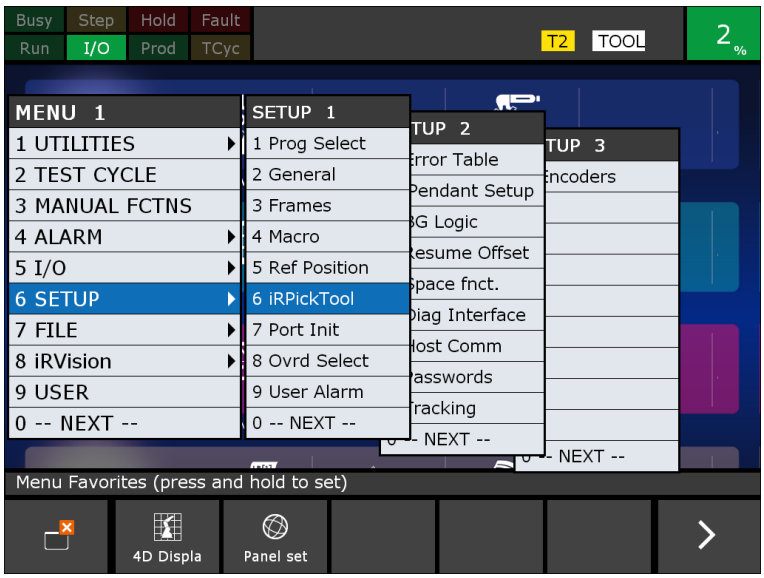
4.2 SETTING UP APPLICATIONS

This section explains the setup of *i*RPickTool applications, such as setting up workcells, sensors, etc.

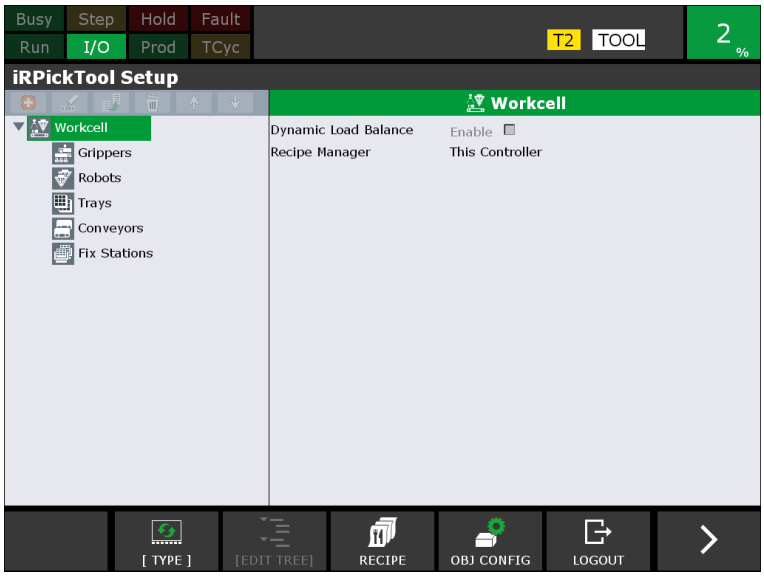
4.2.1 Workcell Setup

Set up a workcell.

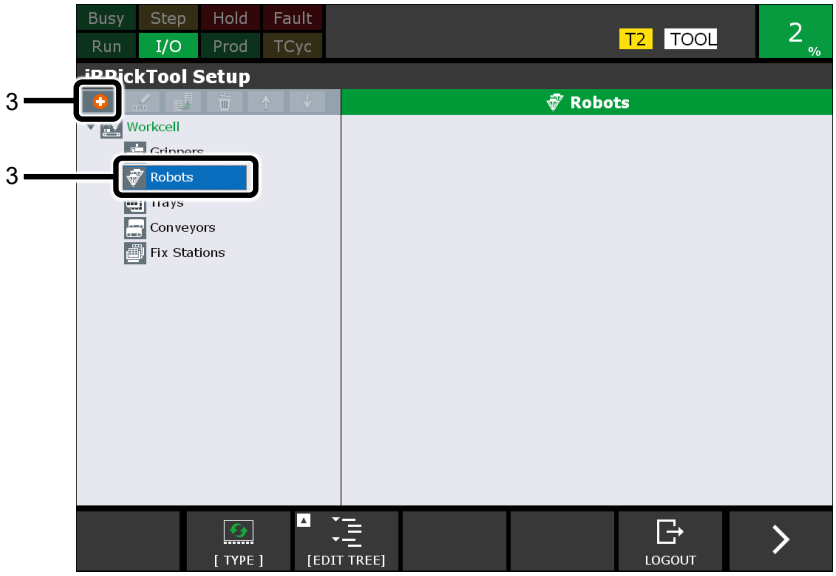
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SETUP] → [*i*RPickTool] from the menu.



The following *i*RPickTool setup screen will appear.

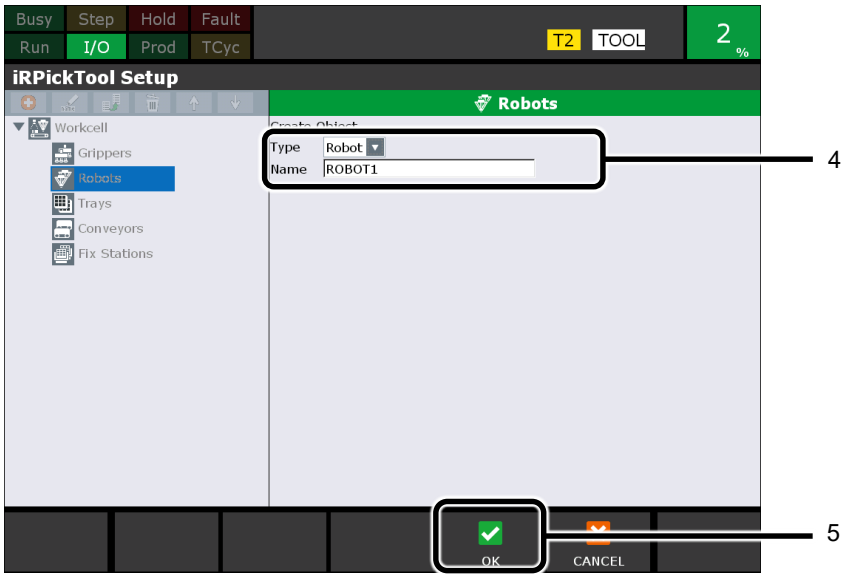


3 Select [Robots], then press the [CREATE] button above the tree.



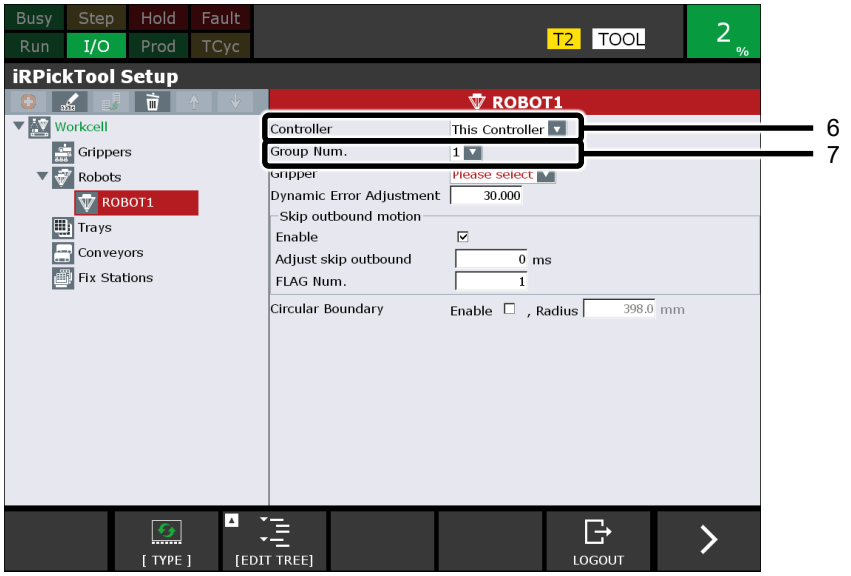
4 Input the name of the robot in [Name].
We recommend using the name that is displayed as the initial value just as it is.
Input using half-width alphanumeric characters and half-width katakana. The name must contain no spaces, not contain the symbol other than underscore, and start with a letter.

5 Press F4 [OK].
The robot object is created.

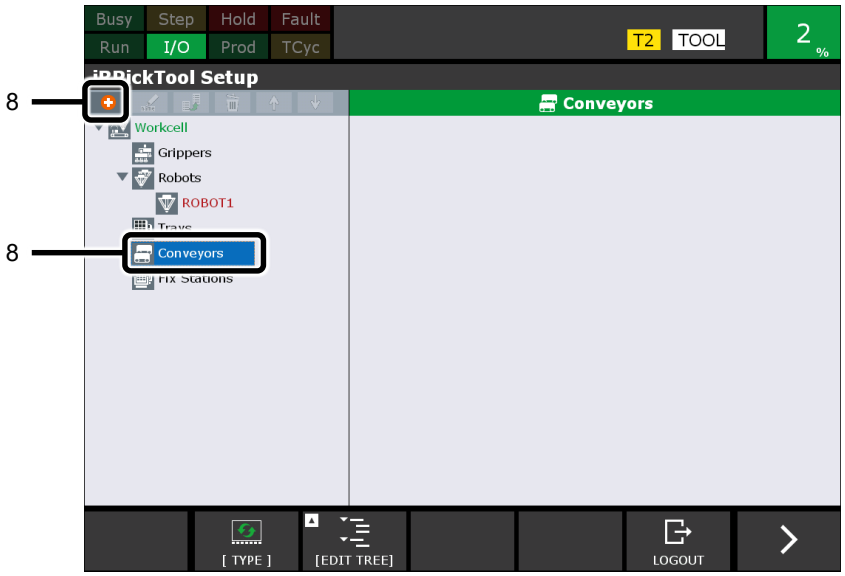


6 For [Controller], select [This Controller] from the menu (if there is one robot).

7 For [Group Num.], select [1] from the menu.

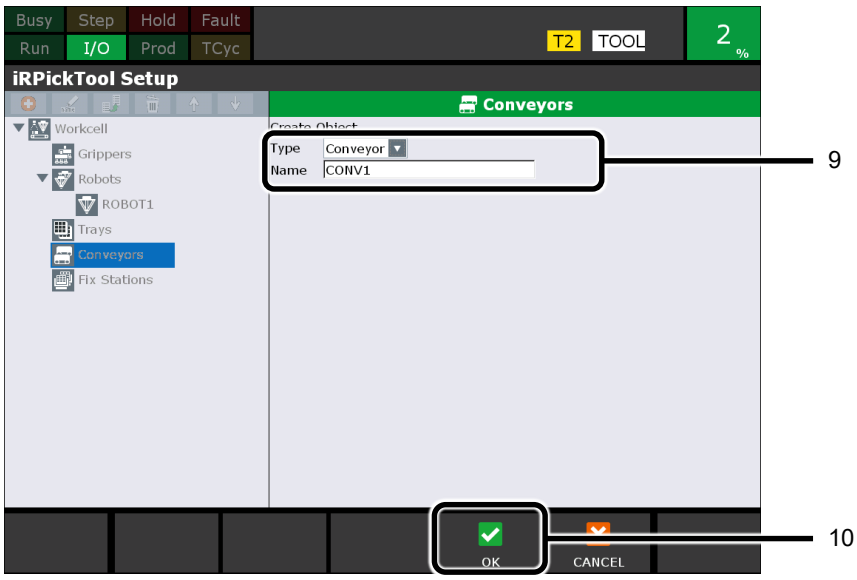


8 Next, create an object for the infeed conveyor. Select [Conveyors], then press the [CREATE] button above the tree.

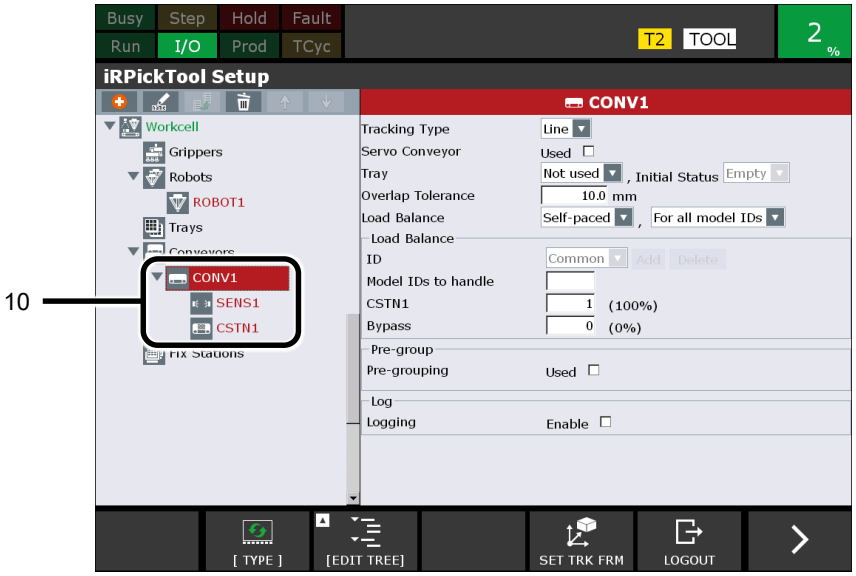


9 Input the name of the infeed conveyor in [Name].
We recommend using the name that is displayed as the initial value just as it is.
Input using half-width alphanumeric characters and half-width katakana. The name must contain no spaces, not contain the symbol other than underscore, and start with a letter.

10 Press F4 [OK].
The object is created.

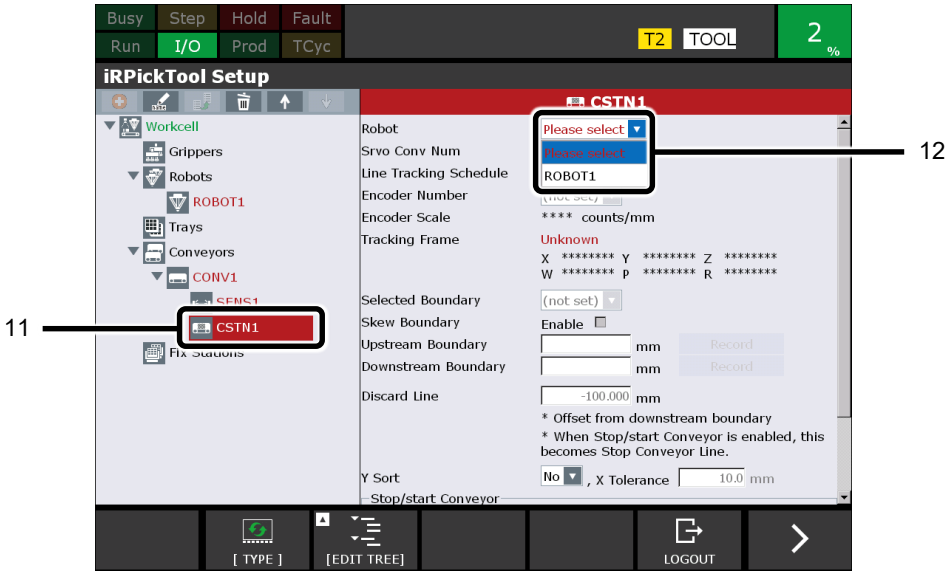


When a new conveyor is created, a sensor and conveyor station will be automatically created.

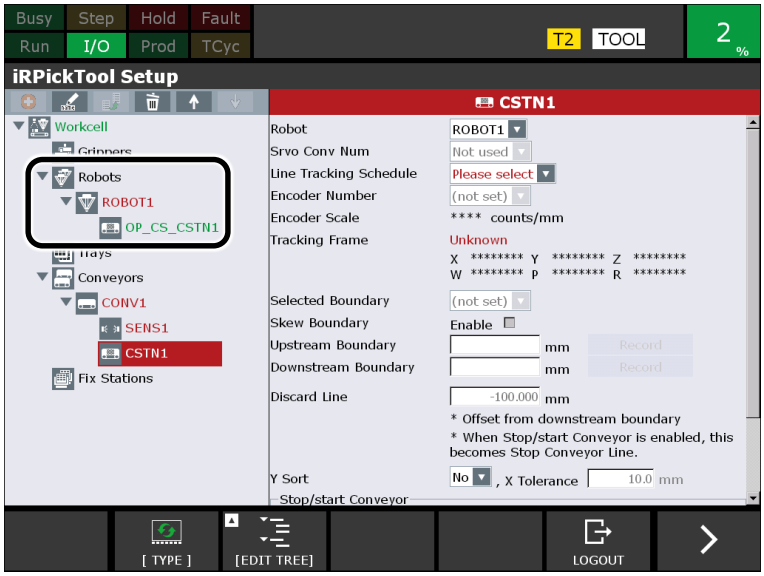


11 Select the conveyor station [CSTN1] that was automatically created in step 10.

12 For [Robot], select a robot from the menu.
Select the robot name that was input in step 4.

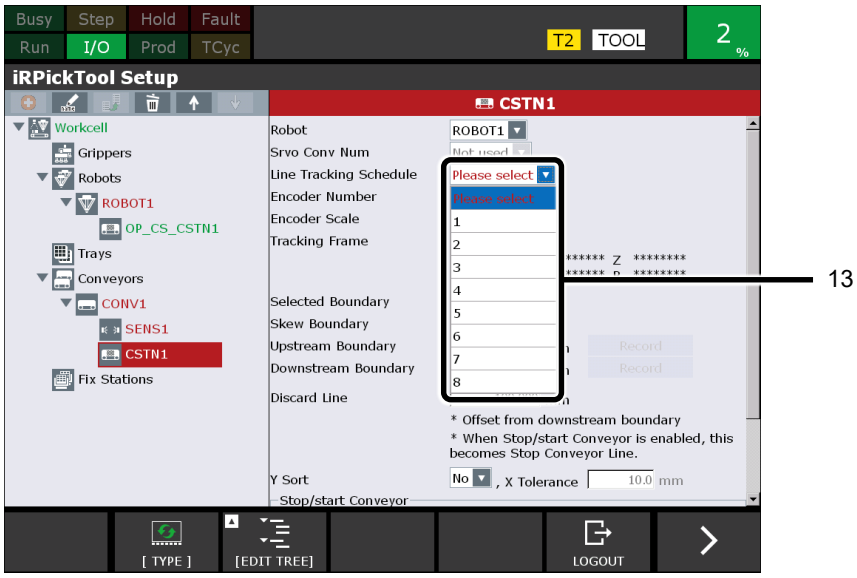


When the name of the robot is selected, the conveyor station operation [OP_CS_CSTN1] will be automatically added underneath the selected robot, as shown below.



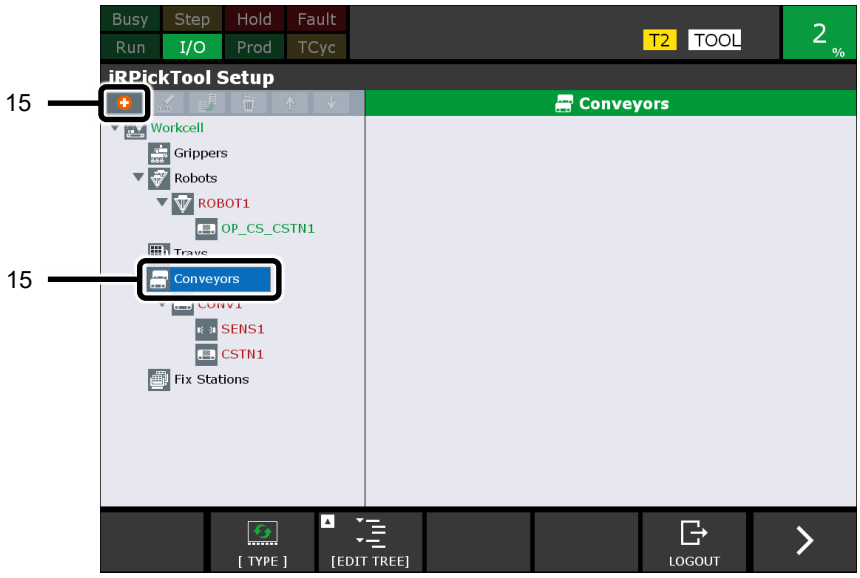
Refer to Subsection 4.2.10, “Setting Up Operations” for information on the setup of conveyor station operations.

13 Select [Line Tracking Schedule] from the menu. Select '1.'



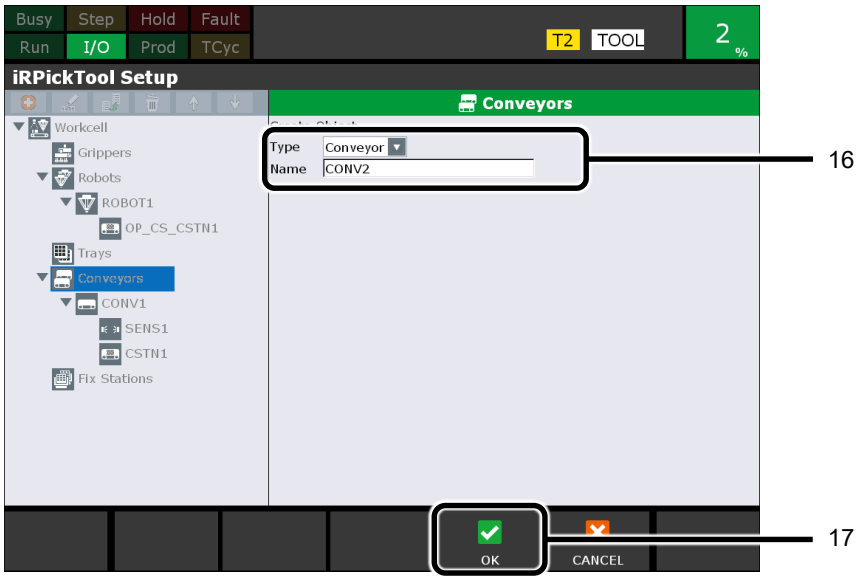
14 Select [Encoder Number] from the menu. Select '1.'

15 Create an object for the outfeed conveyor. Select [Conveyors], then press the [CREATE] button above the tree.

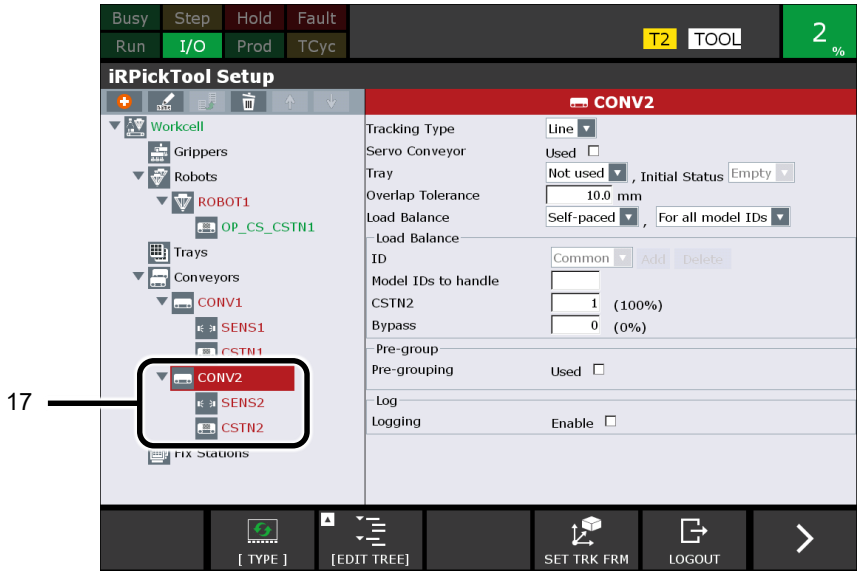


16 Input the name of the outfeed conveyor in [Name].
We recommend using the name that is displayed as the initial value just as it is.
Input using half-width alphanumeric characters and half-width katakana. The name must contain no spaces, not contain the symbol other than underscore, and start with a letter.

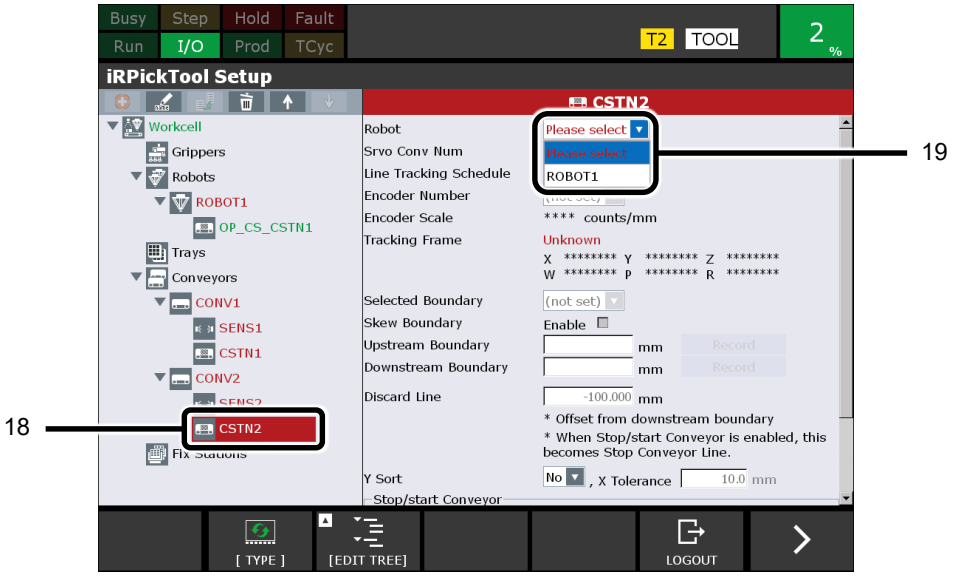
17 Press F4 [OK].
The object is created.



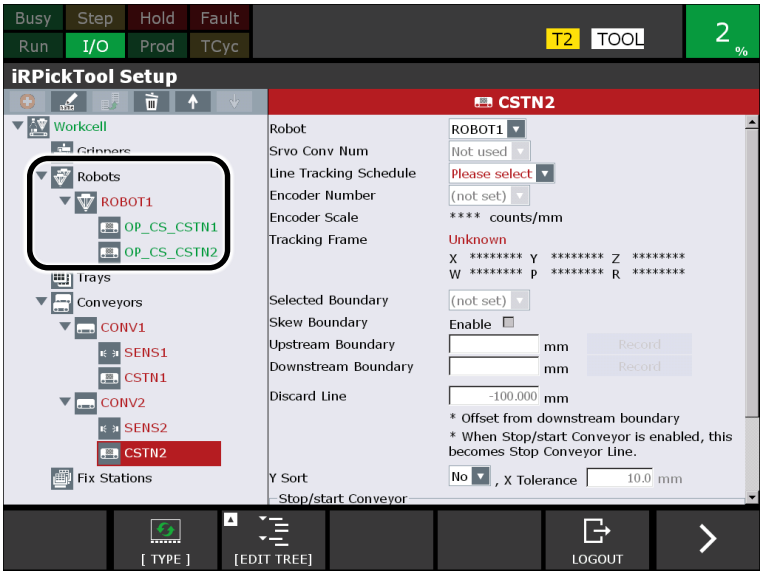
When a new conveyor is created, a sensor and conveyor station will be automatically created.



18 Select the conveyor station [CSTN2] that was automatically created in step 17.
19 For [Robot], select a robot from the menu.
Select the robot name that was input in step 4.



When the name of the robot is selected, the conveyor station operation [OP_CS_CSTN2] will be automatically added underneath the selected robot, as below.



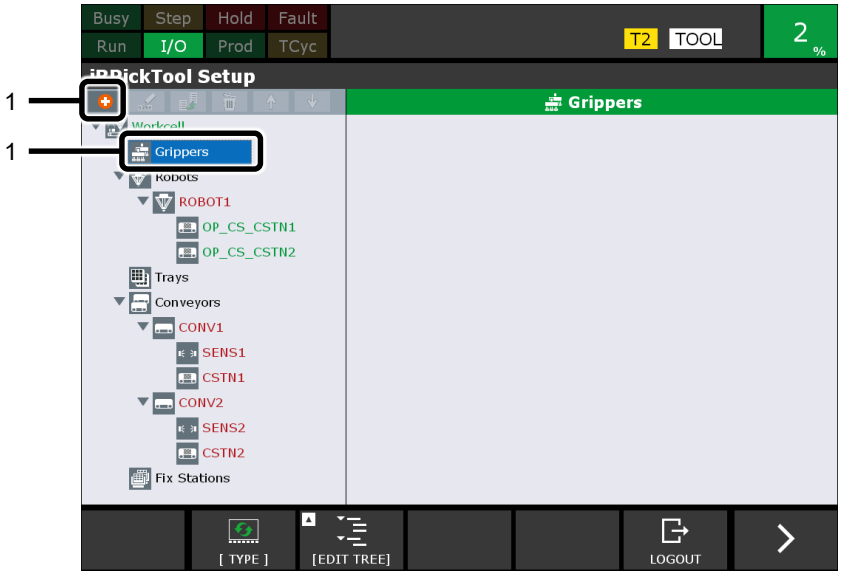
20 Select [Line Tracking Schedule] from the menu. Select '2.'

21 Select [Encoder Number] from the menu.
Select '2.'

4.2.2 Setting Up a Gripper

Set up a gripper and zones.
For the gripper, enter index delays, etc. that are suitable for the picking / placing motion. For the setup of each zone, which is a child object of a gripper, enter items such as tool frame, zone activation I/O, zone part presence I/O, etc. Single Pick Grippers will have a single zone, multi pick grippers will have multiple zones.

1 In the tree view, select [Grippers], then press the [CREATE] button.



2 Input the name of the gripper in [Name].
We recommend using the name that is displayed as the initial value just as it is.
Input using half-width alphanumeric characters and half-width katakana. The name must contain no spaces, not contain the symbol other than underscore, and start with a letter.
3 Press F4 [OK].

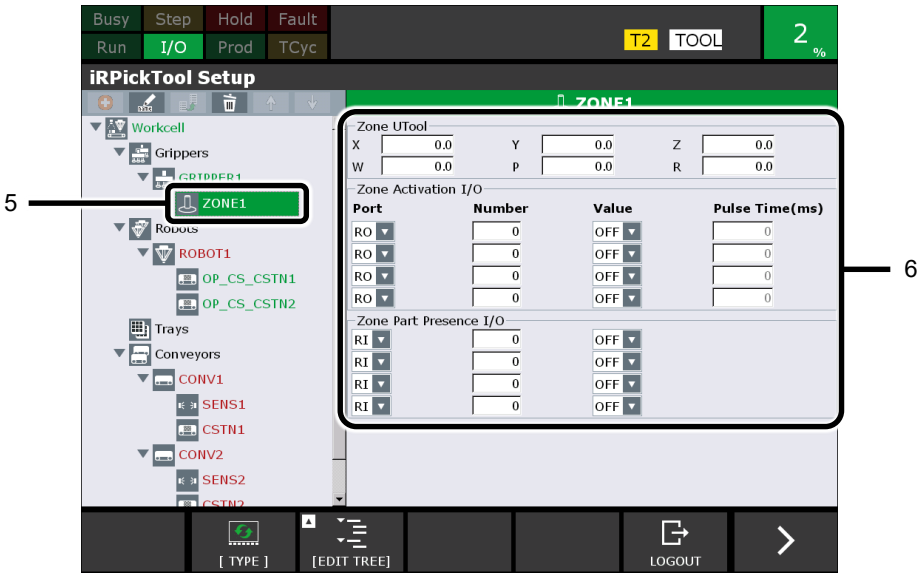
A gripper is added. Under [Grippers] in the tree view, a zone will be automatically added.
4 Set [Pick Delay], [Place Delay], and [Part Presence Delay].
- Pick Delay: Set the time the robot will wait for after it reaches the pick position

- Place Delay: Set the time the robot will wait for after it reaches the drop position.
- Part Presence Delay: Set the time from when the pick output signal is sent until pick confirmation is executed.

5 Select [ZONE].
The zone setting screen will appear.

6 Set [Zone UTool], [Zone Activation I/O], and [Zone Part Presence I/O].

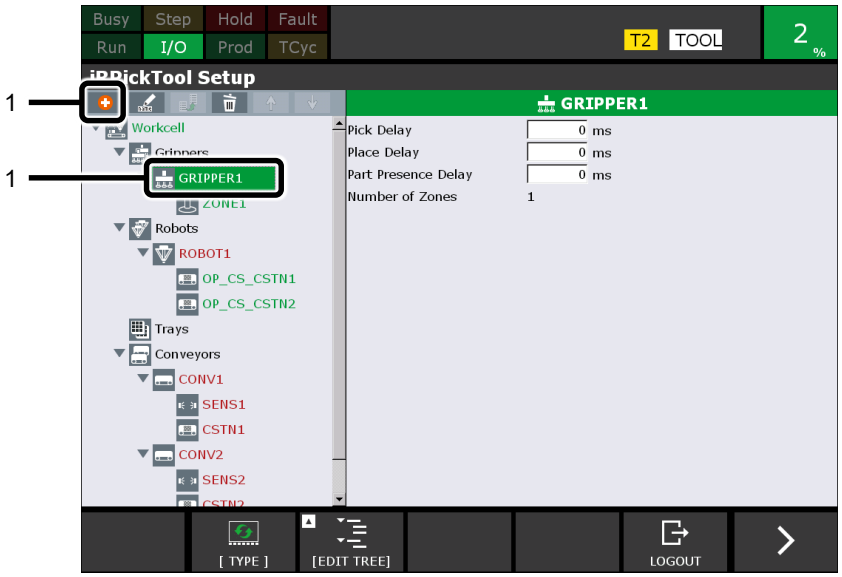
- Zone UTool: Set the tool frames that correspond to each zone of the gripper.
  * It is OK to leave Zone UTool unset at this stage. Set it up in Subsection 4.2.11, “Setting Up a Tool Frame for the Hand” .
- Zone Activation I/O: The output value is either [ON], [OFF], or [Pulse]. When [Pulse] is selected, set the pulse output time in seconds.
- Zone Part Presence I/O: The value of the input signal is either [ON], or [OFF].



Adding zones

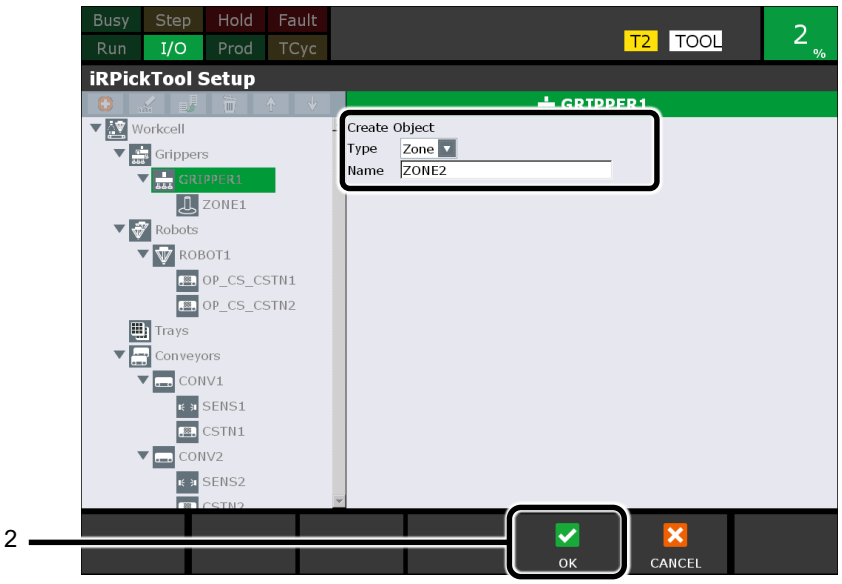
When using a multiple-item picking hand, add zones to the gripper using the following procedure.

1 In the tree view, select the [GRIPPER] you want to add to, then press the [CREATE] button.

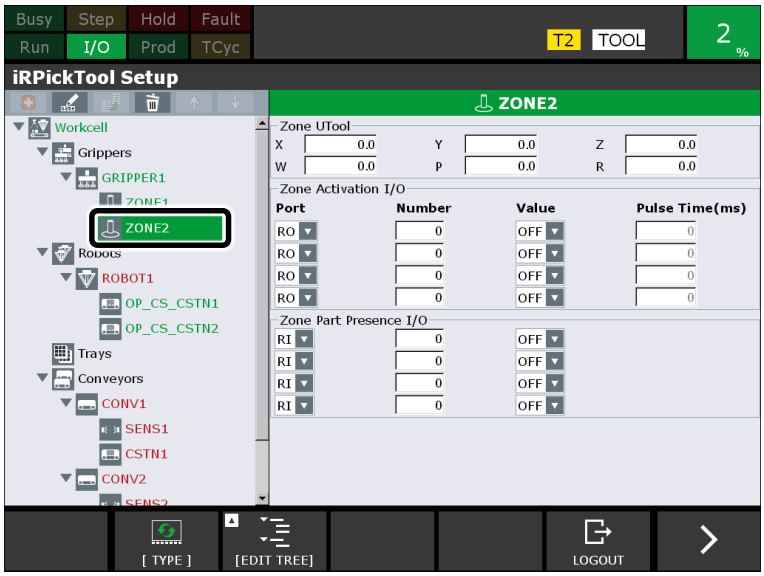


[ZONE 2] will be displayed in the create screen.
A standard name (e.g. Zone 2) will be input automatically in [Name]. Change it as required.

2 Press F4 [OK].



[Zone 2] will be added to the tree view.

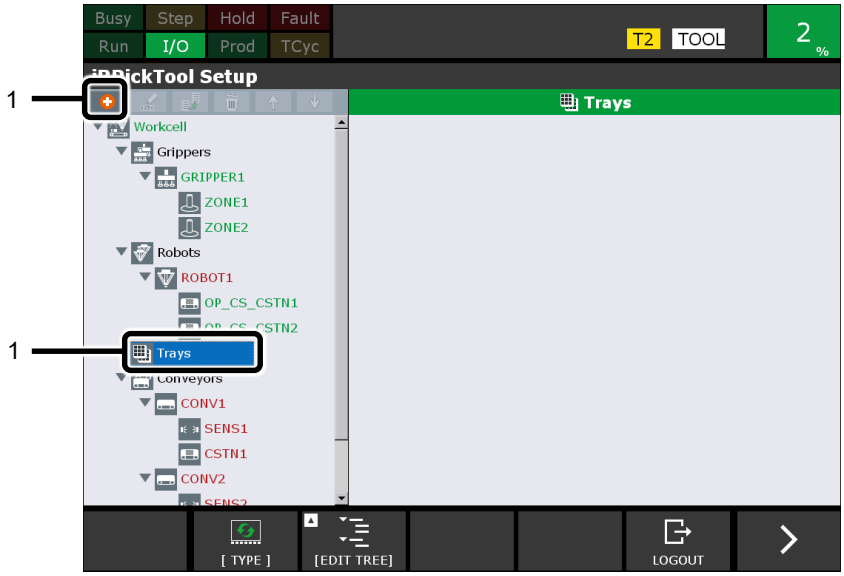


3 If you are going to create further zones, repeat steps 1 and 2 for each zone.

4.2.3 Setting up Trays

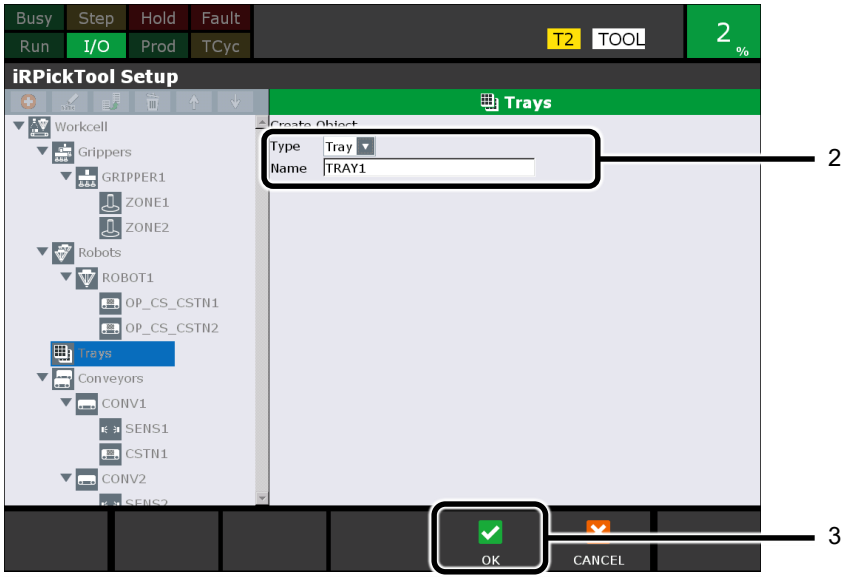
Set up the trays that are used on the outfeed conveyor.

1 Select [Trays], then press the [CREATE] button above the tree.



2 Enter the name of the tray in [Name].
We recommend using the name that is displayed as the initial value just as it is.
Input using half-width alphanumeric characters and half-width katakana. The name must contain no spaces, not contain the symbol other than underscore, and start with a letter.

3 Press F4 [OK].
The object is created.



A screen similar to the one below will appear.

4 Decide the tray frame.
The origin may will be an arbitrary position, such as one corner of the box.
Aligning the flow direction of the conveyor with the X direction of the tray frame will make things easier to understand. (Other directions are also OK.)

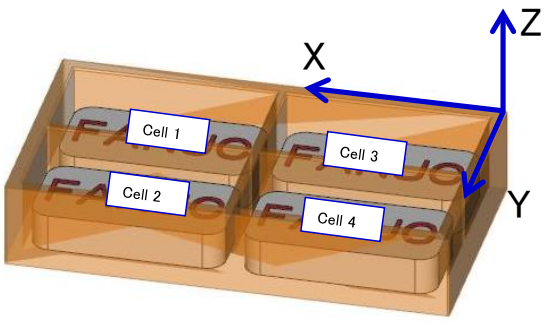
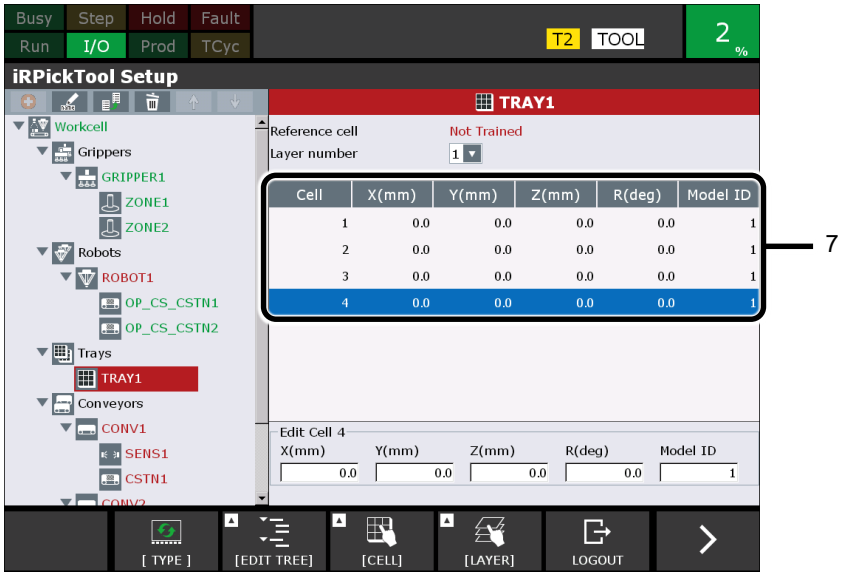


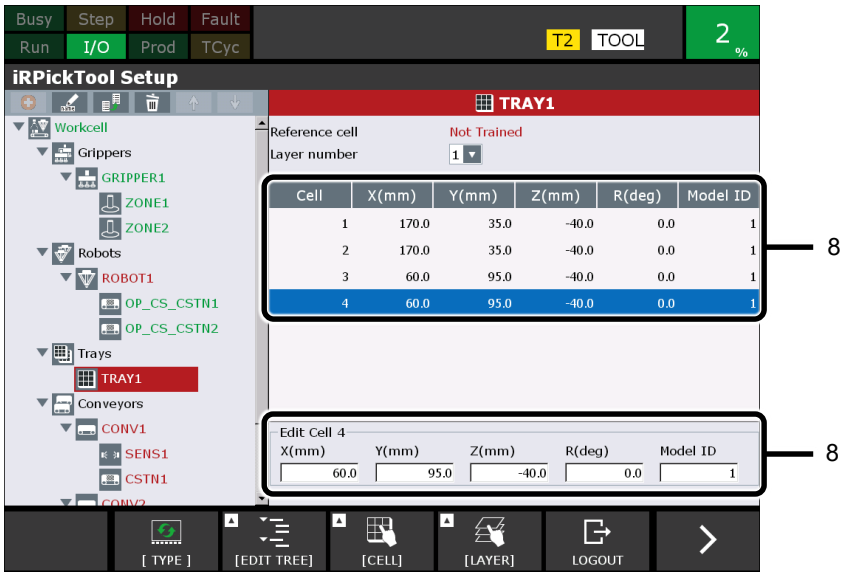
Fig. 4.2.3(a) Deciding the origin of a tray

5 Press F3 [CELL].
6 Select [ADD CELL] from the menu.

7 Add the cells for the first stage.



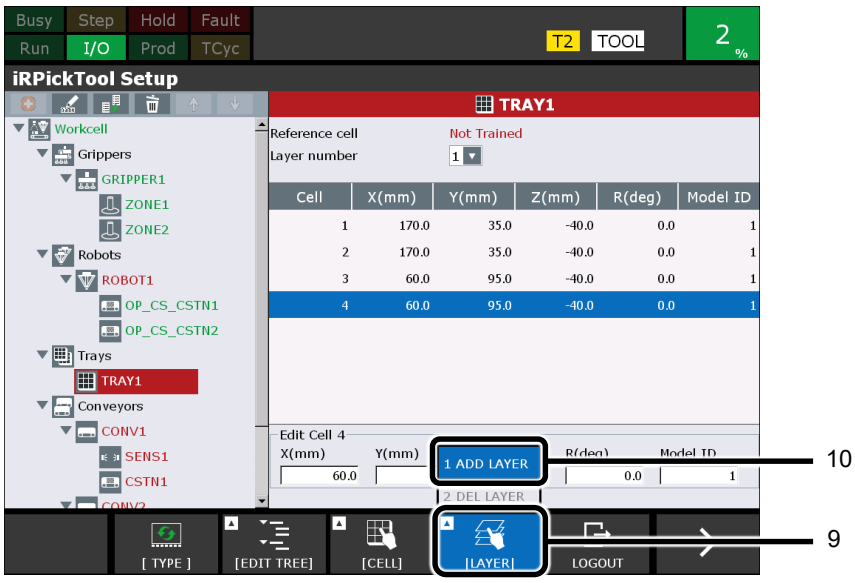
8 Enter the coordinates of each cell in the cell edit screen.
Select the row of the cell that you want to edit, and enter the coordinates in the frame at the bottom of the screen.
Enter the coordinates of a cell in the tray frame coordinate system. For details, refer to Section 6.3, “SETUP OF A TRAY”.



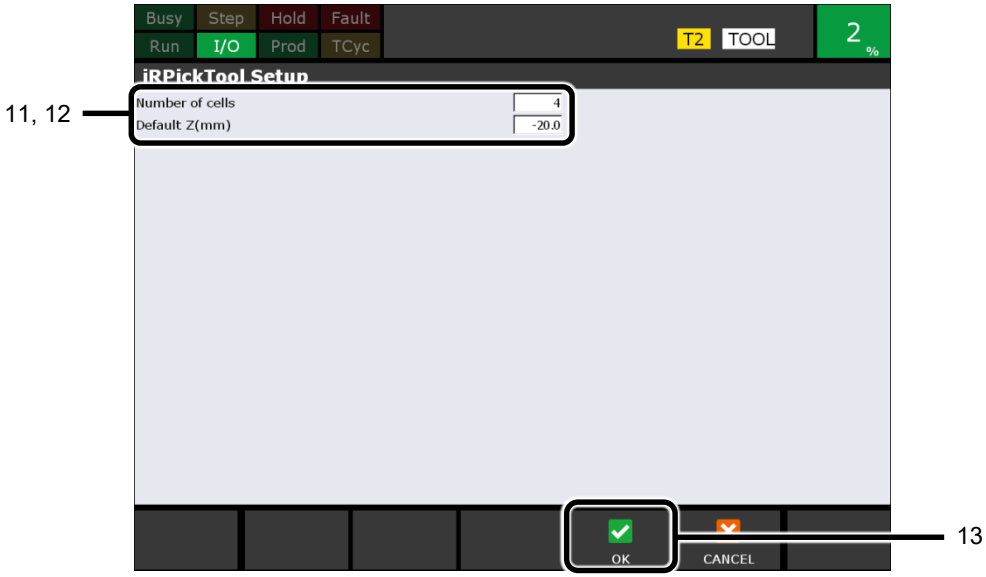
If trays are double-stacked, perform the following operation.

9 Press F4 [LAYER].

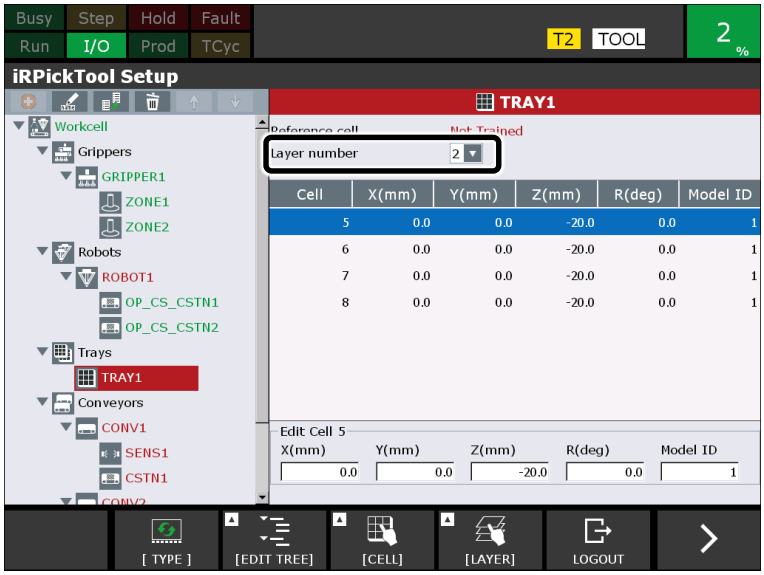
10 Select [ADD LAYER] from the menu.



11 Enter the number of cells in the second stage in [Number of cells].
12 Enter a Z value (mm) in [Default Z(mm)].
13 Press F4 [OK].

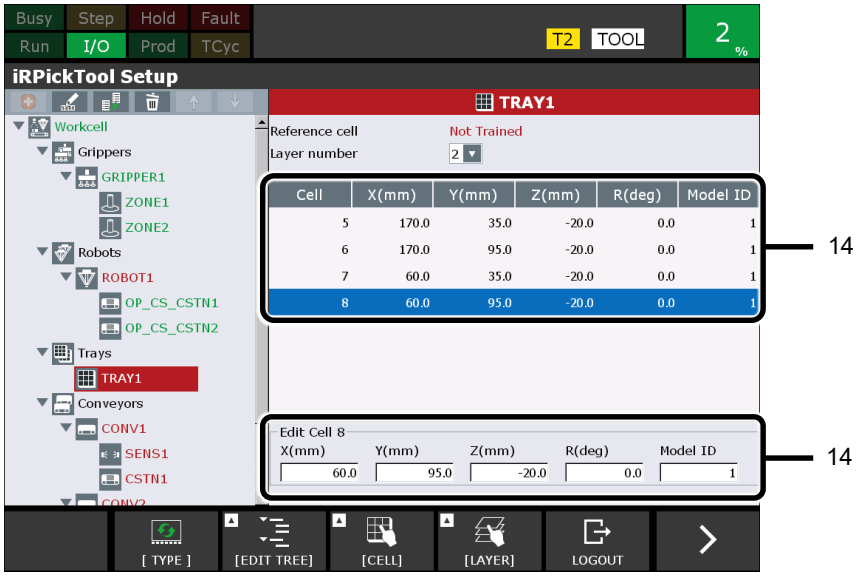


A screen similar to the one below will appear.
Layer '2' is created in [Layer number].



14 Enter the coordinates of each cell in Layer 2 in the cell edit screen.
Select the row of the cell that you want to edit, and enter the coordinates in the frame at the bottom of the screen.

* The reference cell can remain untrained at this point. It is automatically taught in Subsection 4.2.12, “Setting Up a Reference Position and Teaching a Robot Position”.



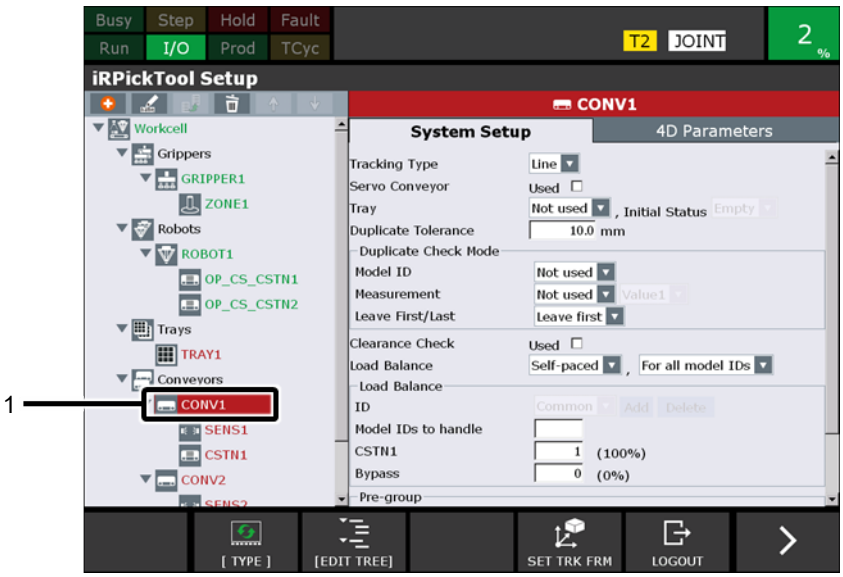
4.2.4 Setting up a Conveyor

Set up a conveyor. Depending on whether trays are used or not, the procedure will be different.

4.2.4.1 Setting up an infeed conveyor

Set up an infeed conveyor. This setup will not use trays.

1 Select the conveyor [CONV1].



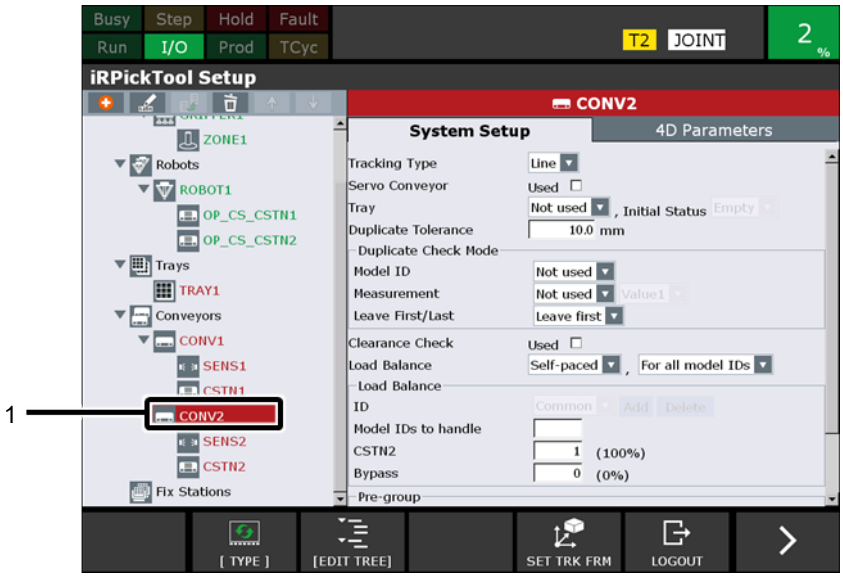
The conveyor setup screen will appear.
For conveyor setup, the following data that appears as the initial values can be used as it is.

[Tracking Type] : Line
[Servo Conveyor] : Do not check [Used]
[Tray] : Not used
[Duplicate Tolerance] : 10.0 mm
[Load Balance] : Self-paced, For all model IDs

4.2.4.2 Setting up an outfeed conveyor

Set up an outfeed conveyor. This setup will use trays.

1 Select the conveyor [CONV2].



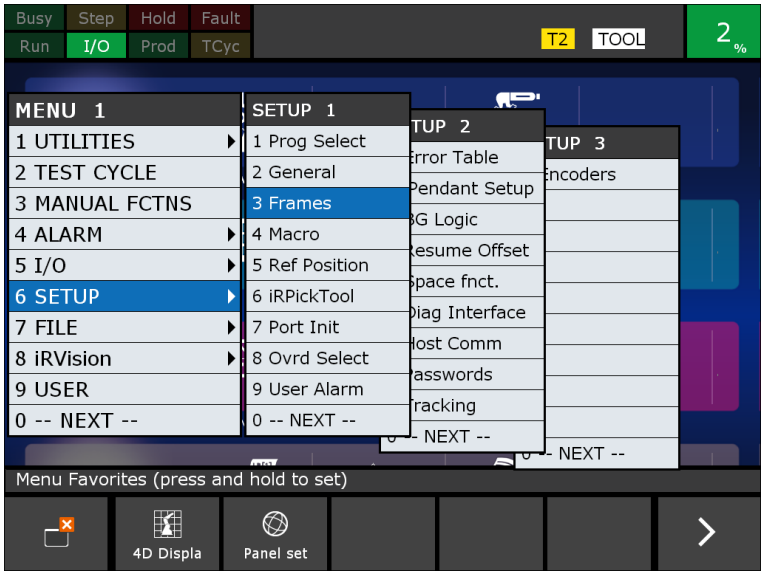
The conveyor setup screen will appear.

2 Select [TRAY1] in [Tray] and [Empty] in [Initial Status].
For other settings for conveyors, the following data that appears as the initial values can be used as it is.
[Tracking Type] : Line
[Servo Conveyor] : Do not check [Used]
[Duplicate Tolerance] : 10.0 mm
[Load Balance] : Self-paced, For all model IDs

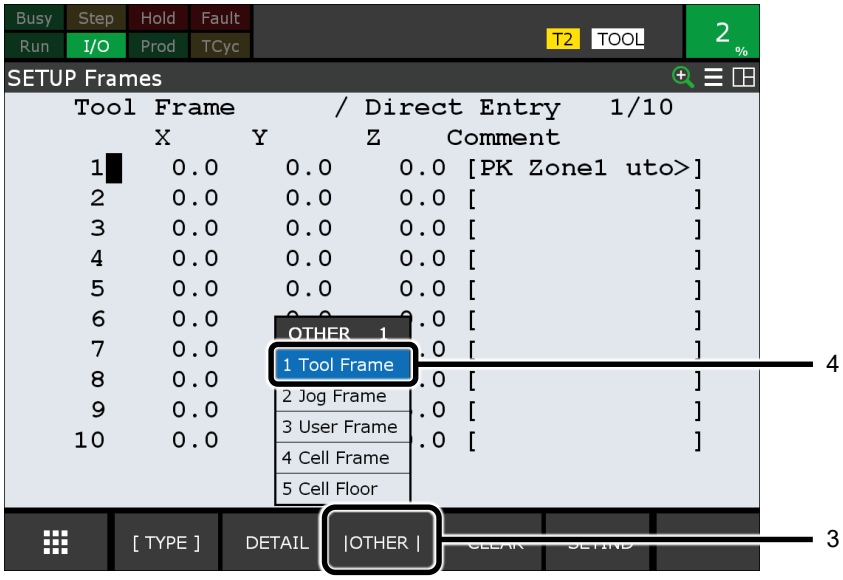
4.2.5 Setting up a Tracking Frame

Set up a tracking frame.

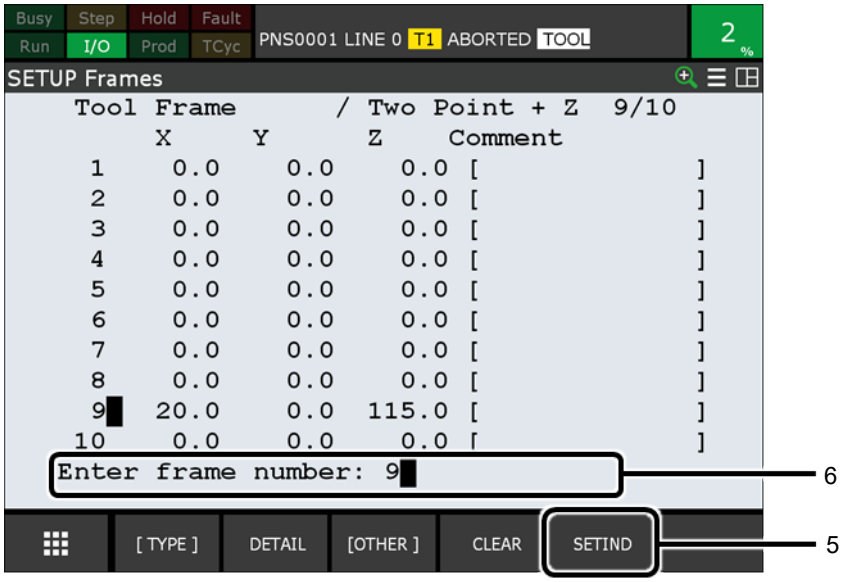
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SETUP] → [Frames] from the menu.



3 Press F3 [OTHER].
4 Select [Tool Frame] from the menu.
The list screen for tool frames will appear.

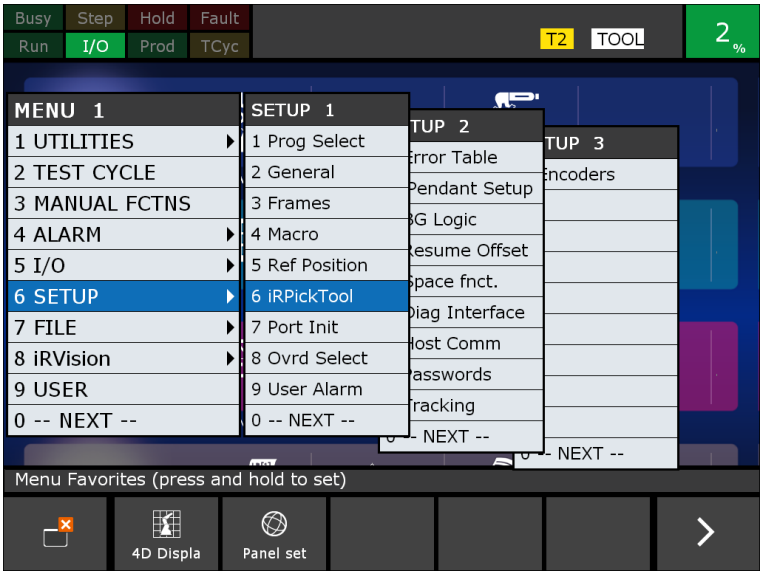


5 Press F5 [SETIND].
6 Enter the frame number that has been used for the pointer tool.
This procedure is explained where Tool 9 is used as the pointer as an example.

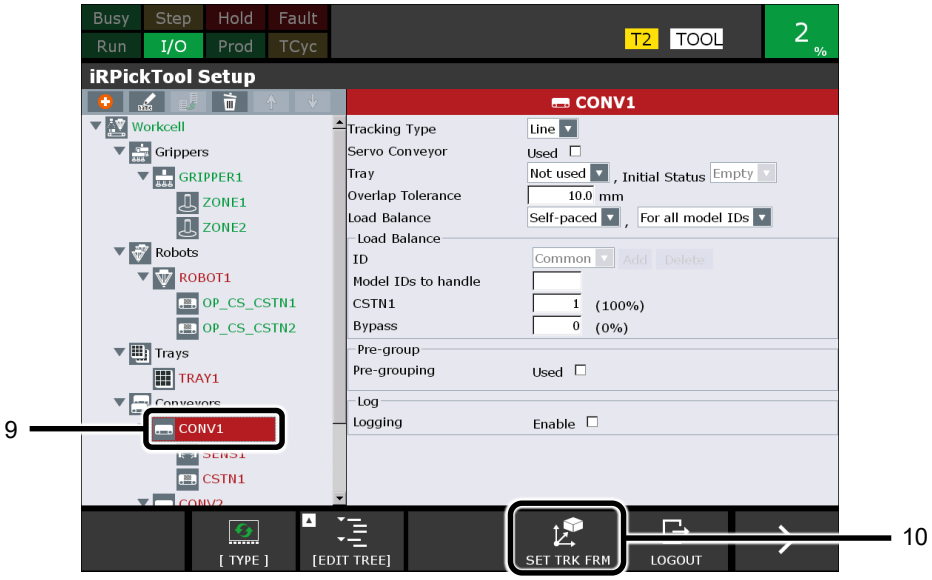


7 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.

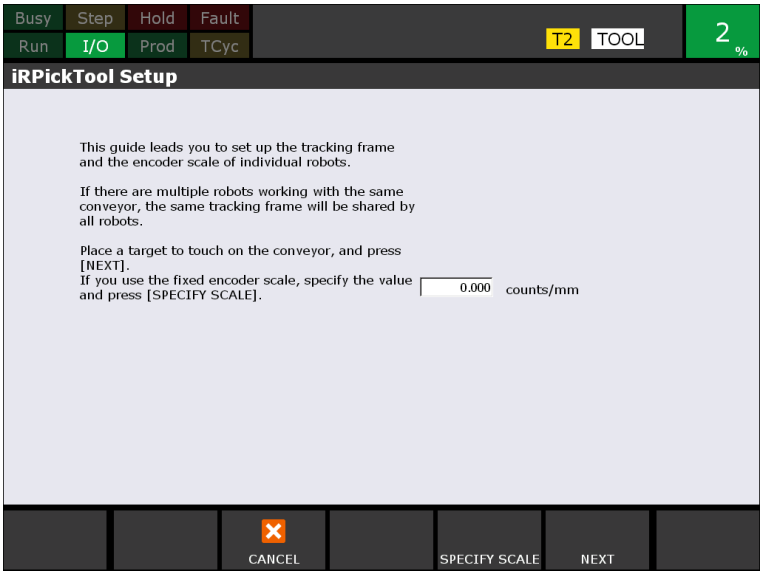
8 Select [SETUP] → [*i*RPickTool] from the menu.



9 First, set up the tracking frame of the infeed conveyor.
Select the conveyor [CONV1].
10 Press F4 [SET TRK FRM].



A screen similar to the one below will appear.



11 Set a calibration grid to the upstream side of the conveyor, in accordance with the procedure shown on the screen.
Fix it in place with tape, etc. to avoid misalignment.

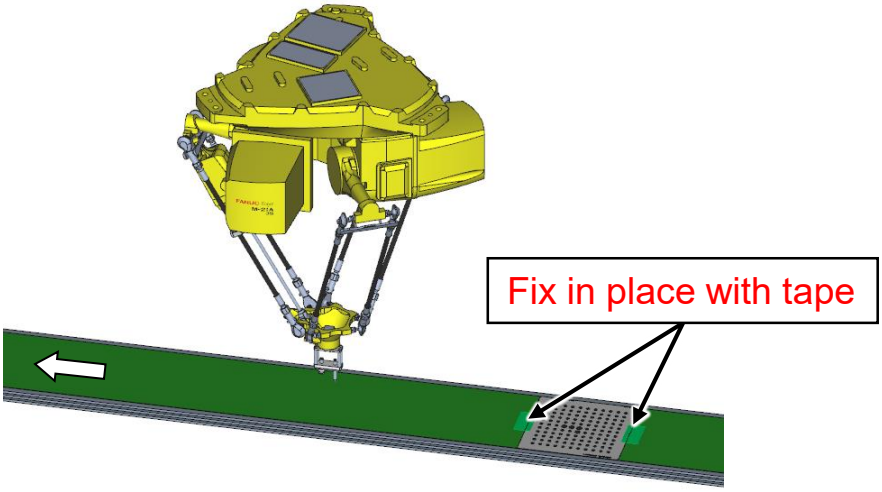


Fig. 4.2.5 (a) Setting a calibration grid to the upstream side

12 Set the flow direction of the conveyor and the X direction of the calibration grid to be in the same direction.

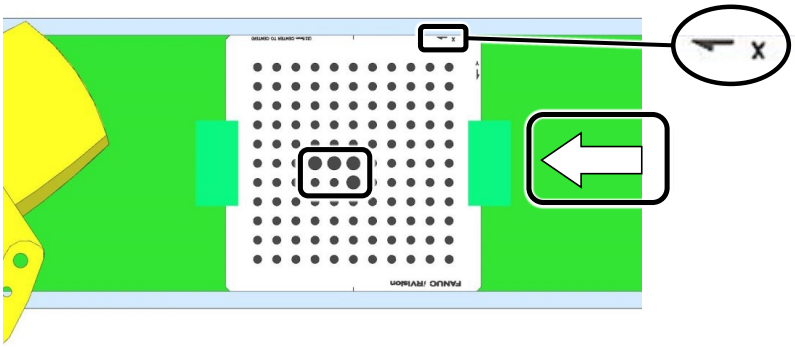
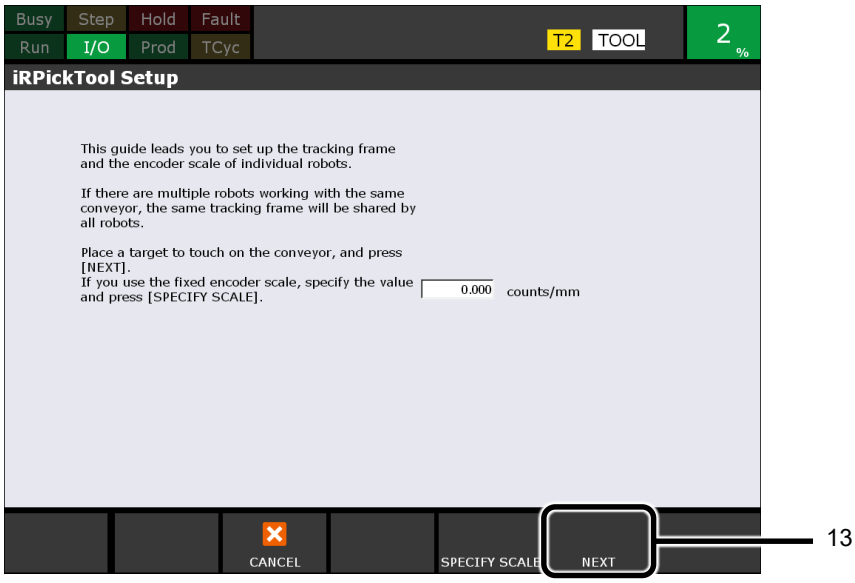
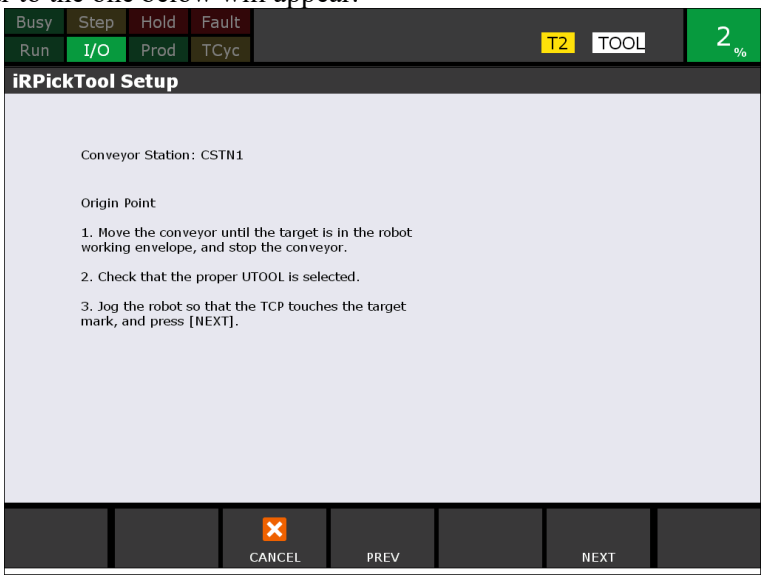


Fig. 4.2.5 (b) Align the setting direction

13 Press F5 [NEXT].



A screen similar to the one below will appear.



14 Move the conveyor. When the calibration grid comes to the upstream within the operating range of the robot, stop the conveyor.

15 Touch up the origin of the calibration grid by jogging the robot pointer to line up with it.

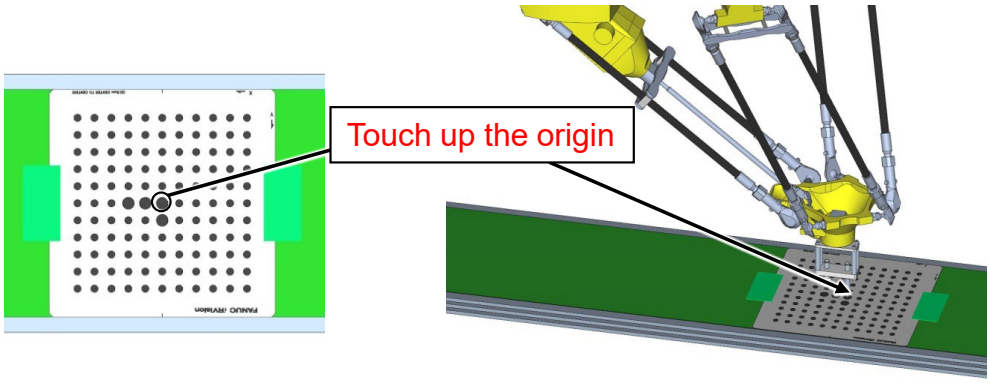
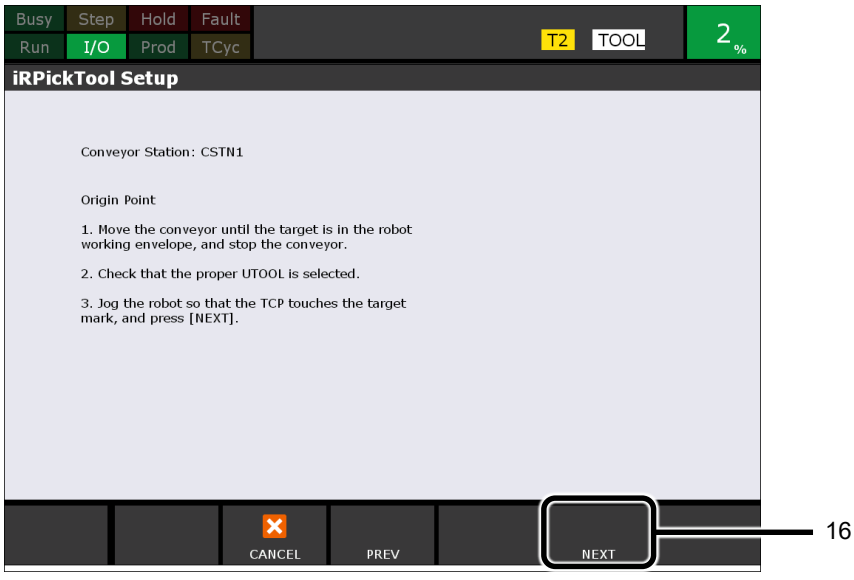


Fig. 4.2.5 (c) Touching up the origin of the calibration grid.

16 Press F5 [NEXT].



A screen similar to the one below will appear.

17 Move the robot up, above the surface of the conveyor .

18 Move the conveyor. When the calibration grid enters to the downstream envelope, still within the operating range of the robot, stop the conveyor.

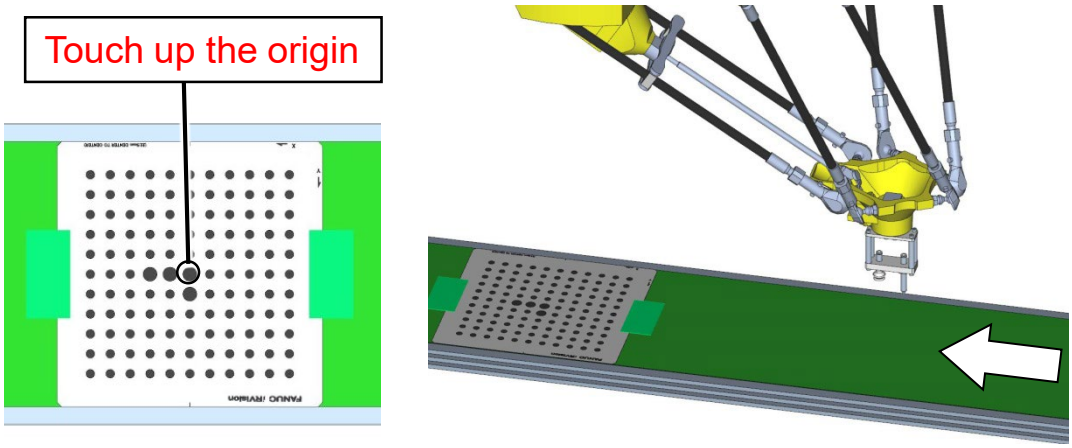


Fig. 4.2.5 (d) Stopping the Conveyor

19 Touch up the origin of the calibration grid.

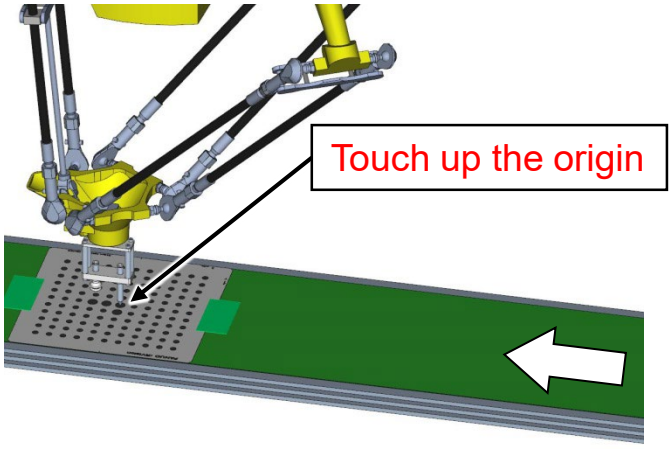
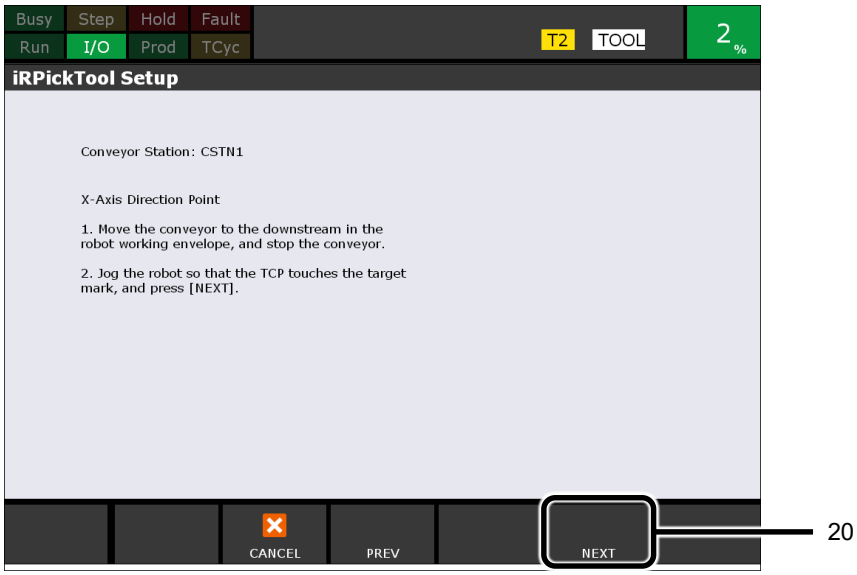


Fig. 4.2.5 (e) Touching up the origin of the calibration grid.

20 Press F5 [NEXT].



A screen similar to the one below will appear

21 Touch up the Y direction of the calibration grid.
Do not move the conveyor at this time.
This is not the Y direction of the robot, it is on the left side relative to the direction of movement of the conveyor. If you are facing downstream in line with the direction of conveyor travel, this point will be on your left, and on the surface of the conveyor.

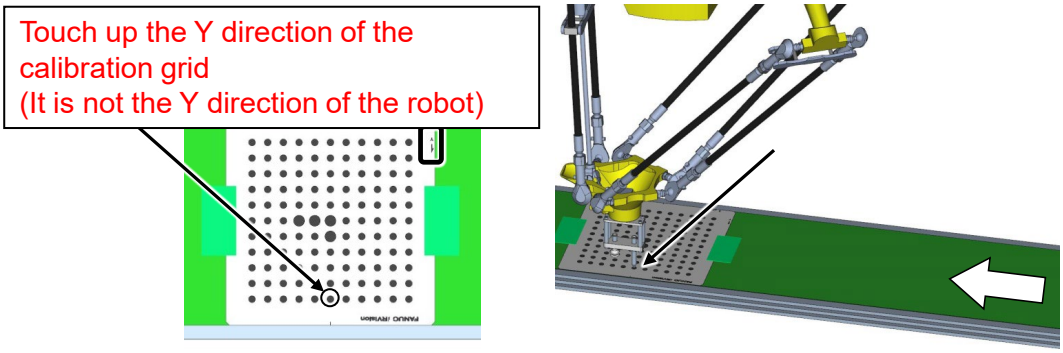
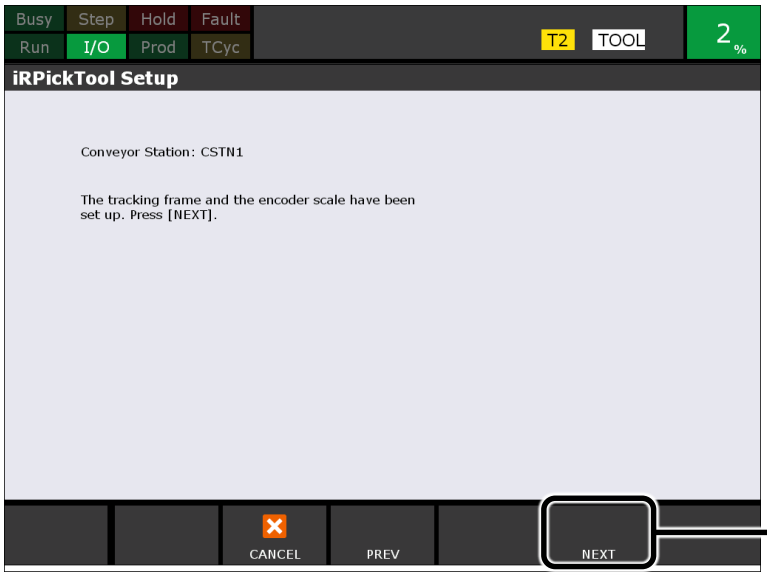


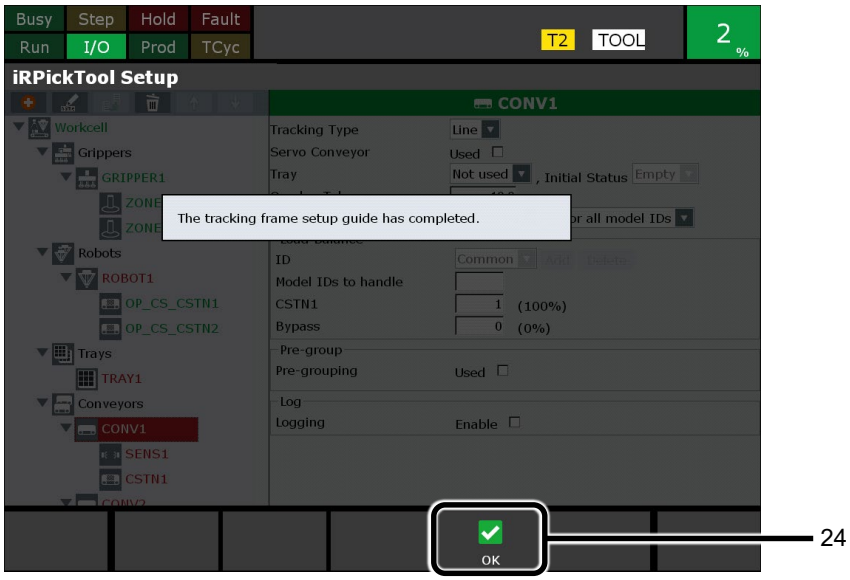
Fig. 4.2.5 (f) Touching up the Y direction of the calibration grid.

22 Press F5 [NEXT].

23 Press F5 [NEXT].



24 Press F4 [OK].



The tracking frame of the infeed conveyor has been set up.

25 Next, set up the tracking frame of the outfeed conveyor.
Select the conveyor [CONV2].
26 Repeat the steps 10 to 24 to set up the tracking frame of the outfeed conveyor.

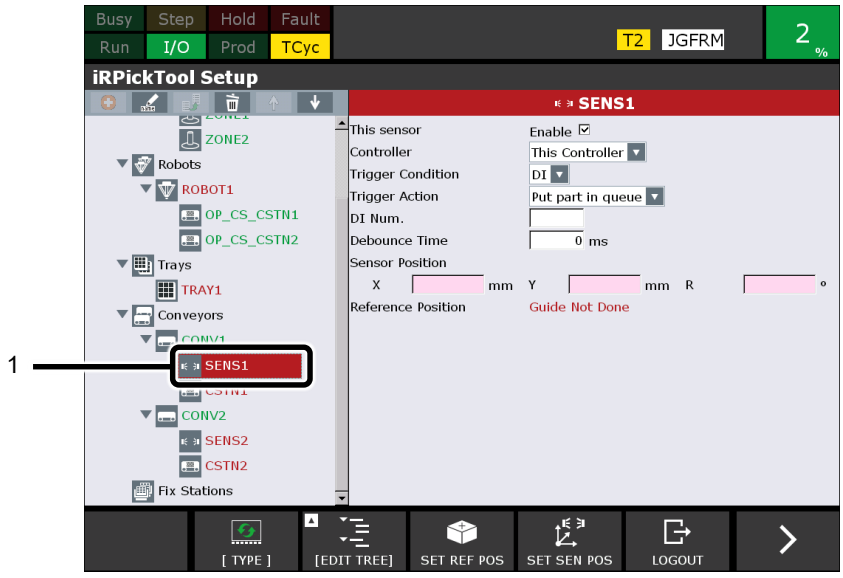
4.2.6 Setting up a Sensor

Set up a sensor.

4.2.6.1 Setting up an infeed conveyor sensor

Set up an infeed conveyor sensor. The setup is for the case of [Distance] + [Find part by vision].

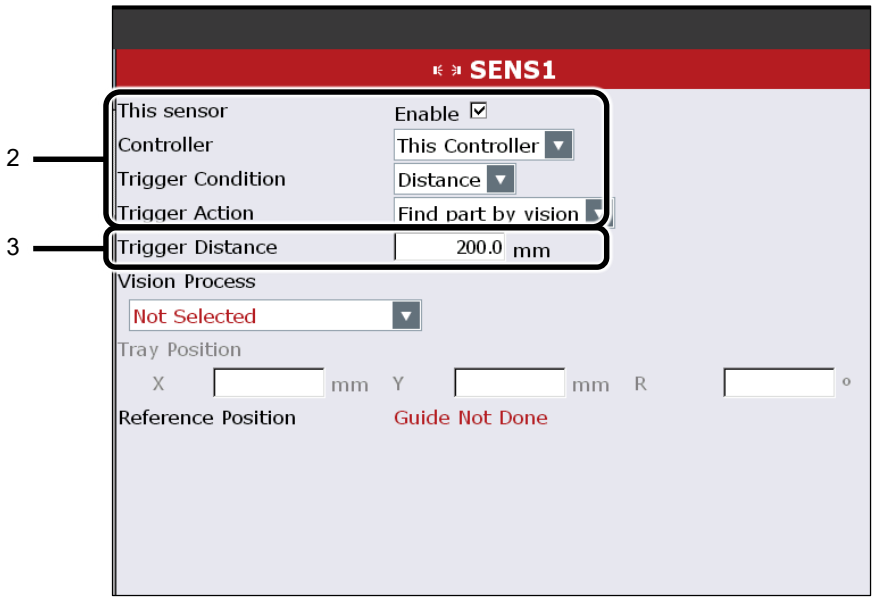
1 Select the sensor [SENS1].



2 Set the following items.
[This sensor] : Check [Enable]
[Controller] : [This Controller]
[Trigger Condition] : [Distance]
[Trigger Action] : [Find part by vision]

When [Trigger Condition] and [Trigger Action] are changed, the setup items change.

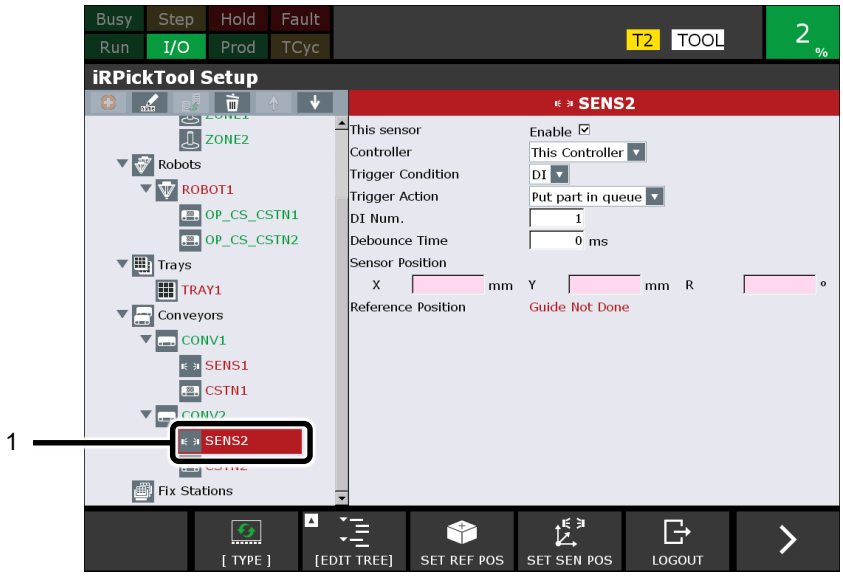
3 Set the following items.
[Trigger Distance] : Set how many mm the conveyor advances each time for vision detection to be performed.
*It is recommended that the trigger distance be set such that a workpiece can be detected twice in the camera FOV.
[Vision process] : You do not have to set this up at this stage.
* Set this up immediately before setting up reference positions. For setup, refer to Subsection 4.2.12.1, “Setting up a reference position for the infeed conveyor, and teaching a robot position”.



4.2.6.2 Setting up an outfeed conveyor sensor (In the case of [DI] / [RI] + [Put part in queue])

Set up an outfeed conveyor sensor. The setup is for a sensor in the case of [DI] / [RI] + [Put part in queue]. In the case of [HDI] + [Put part in queue], skip this section and go to the next section, Subsection 4.2.6.3, “Setting up an outfeed conveyor sensor (In the case of [HDI] + [Put part in queue])”.

1 Select the sensor [SENS2].



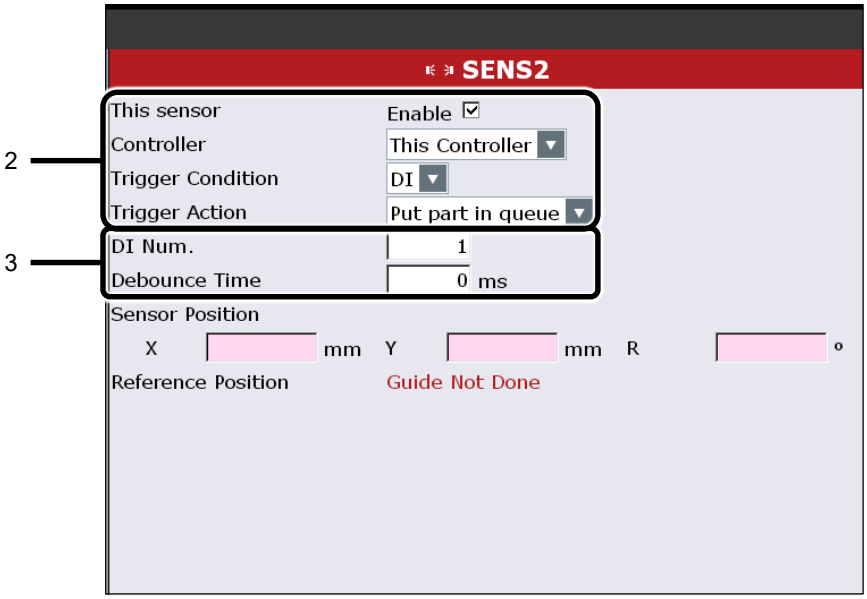
2 Set the following items.

[This sensor] : Check [Enable]
[Controller] : [This Controller]
[Trigger Condition] : [DI]
[Trigger Action] : [Put part in queue]

When [Trigger Condition] and [Trigger Action] are changed, the setup items change.

3 Set the following items.

[DI Num.] : Enter the DI number the sensor input has been assigned.
[Debounce Time] : Enter the time you want to disable input for (to prevent chattering)
[Sensor Position] : You do not have to set this at this stage.
*This is set in the setup of the sensor position. Refer to Subsection 4.2.7, “Setting Up a Sensor Position” for the setup.

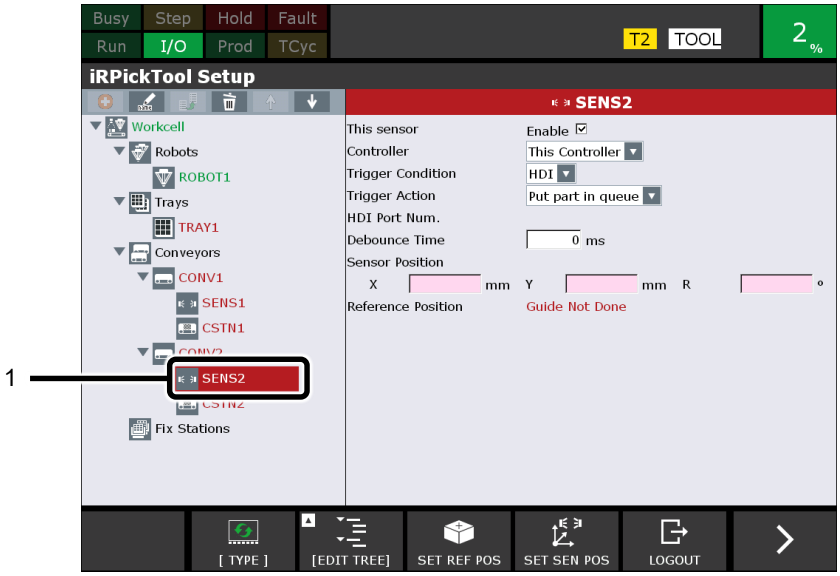


Skip the next section '4.2.6.3 Setting up an Outfeed Conveyor Sensor ([HDI] + [Put part in queue]),' and go to the section '4.2.7 Setting up a Sensor Position.'

4.2.6.3 Setting up an outfeed conveyor sensor (In the case of [HDI] + [Put part in queue])

Set up an outfeed conveyor sensor. The setup is for a sensor in the case of [HDI] + [Put part in queue]. In the case of [DI] / [RI] + [Put part in queue], do not refer to this section, but refer to the previous section 4.2.6.2, “Setting up an outfeed conveyor sensor (In the case of [DI] / [RI] + [Put part in queue])” .

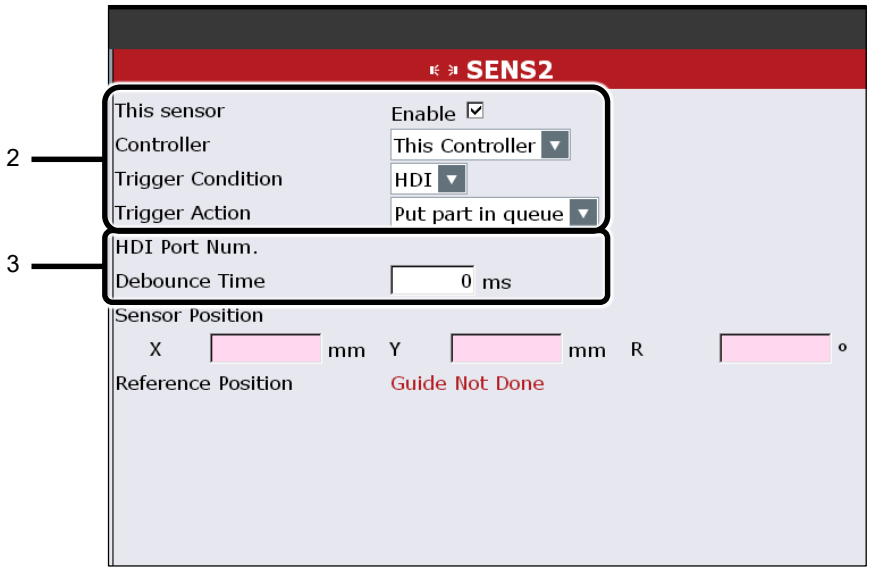
1 Select the sensor [SENS2].



2 Set the following items.
[This Sensor] : Check [Enable]
[Controller] : [This Controller]
[Trigger Condition] : [HDI]
[Trigger Action] : [Put part in queue]

When [Trigger Condition] and [Trigger Action] are changed, the setup items change.

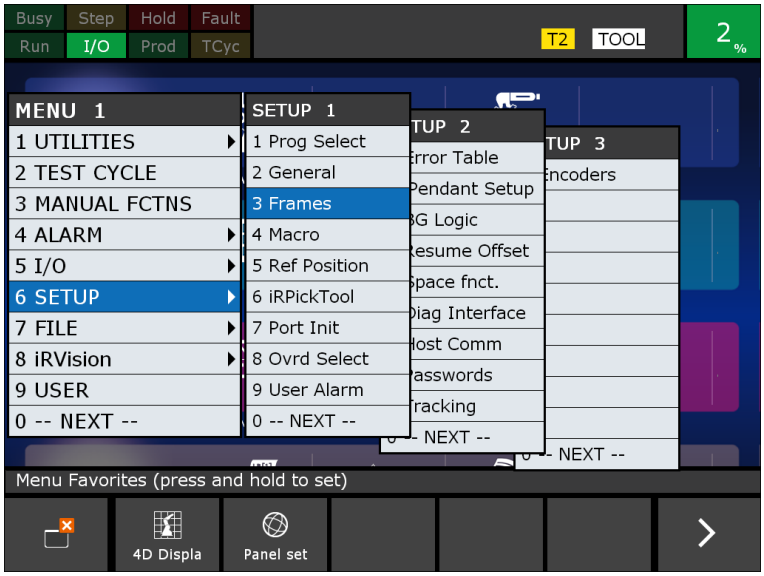
3 Set the following items.
[HDI Port Num.] : The number that was specified in the encoder setup will appear.
[Debounce Time] : Enter the time you want to disable input for (to prevent chattering)
[Sensor Position] : You do not have to set this in this procedure.
*This is set in the setup of the sensor position. Refer to Subsection 4.2.7, “Setting Up a Sensor Position” for the setup.



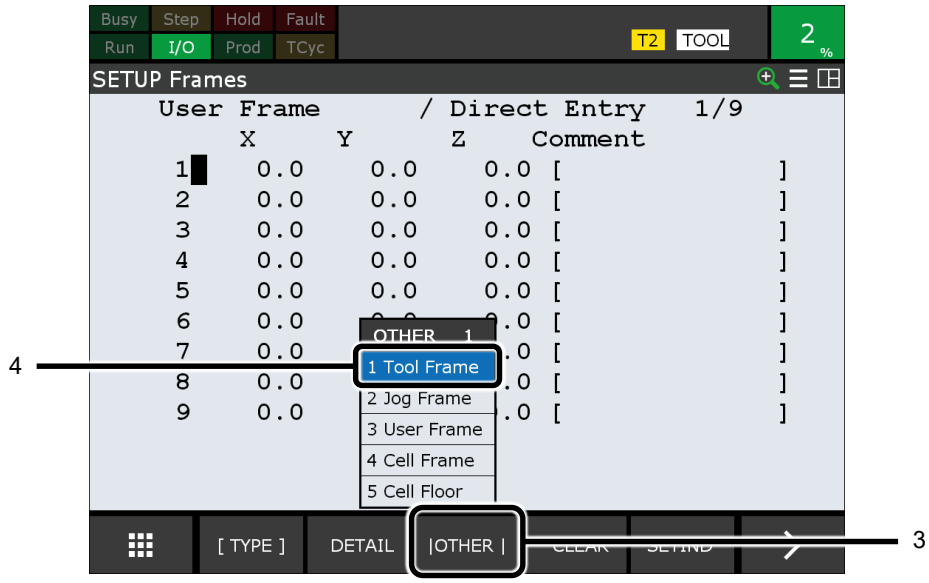
4.2.7 Setting up a Sensor Position

Set up the sensor position of the outfeed conveyor. This is the setup when trays are used.

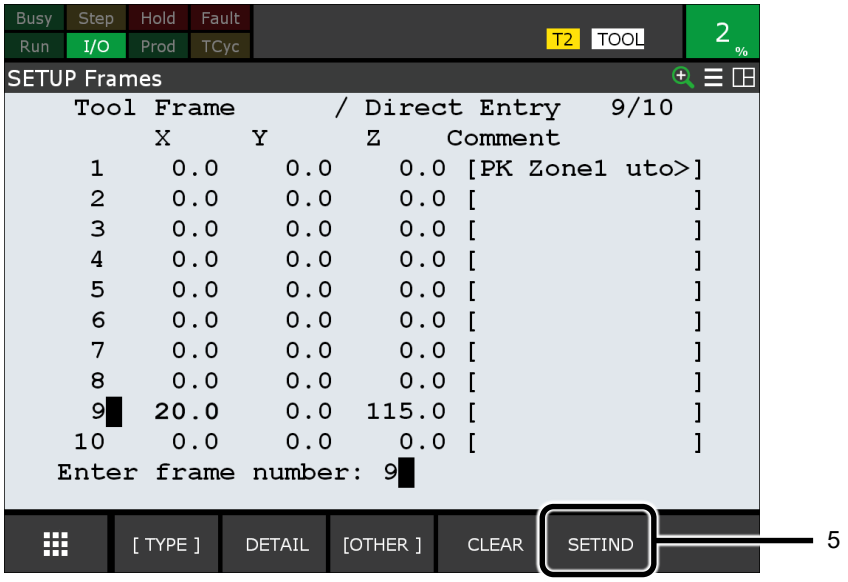
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SETUP] → [Frames] from the menu.



3 Press F3 [OTHER].
4 Select [Tool Frame] from the menu.

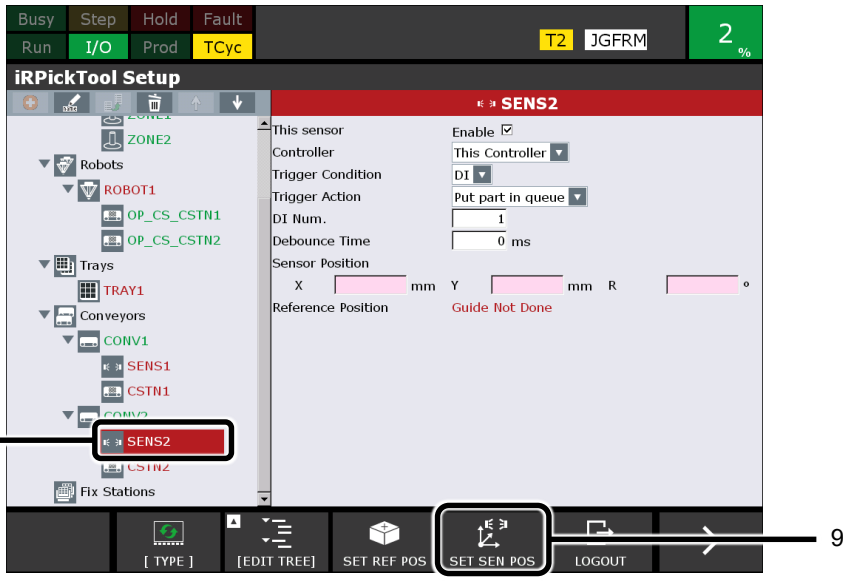


The list screen for tool frames will appear.
5 Press F5 [SETIND].
6 Enter the frame number that has been used for the pointer tool.
This procedure is explained where Tool 9 is used as the pointer an example.

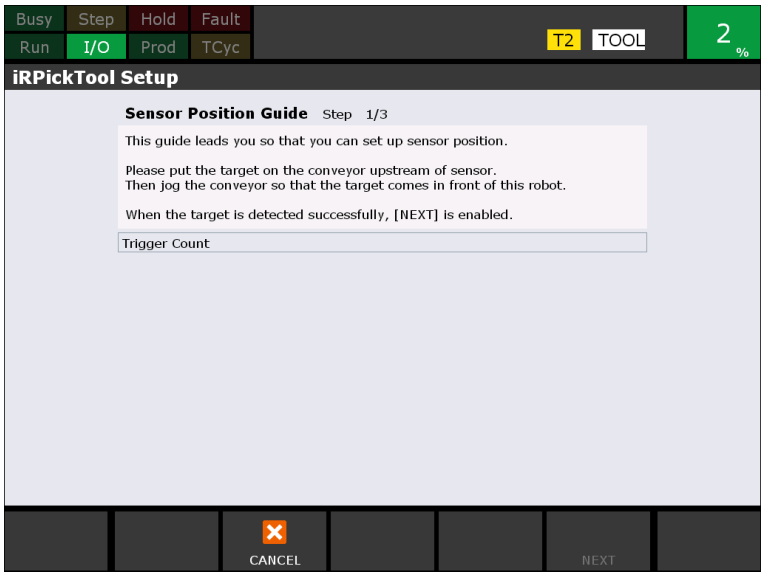


7 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
8 Select [SETUP] → [*i*RPickTool].

9 Select [SENS2], then press F4 [SET SEN POS].



A screen similar to the one below will appear.



10 Set a tray on the conveyor upstream from the sensor.

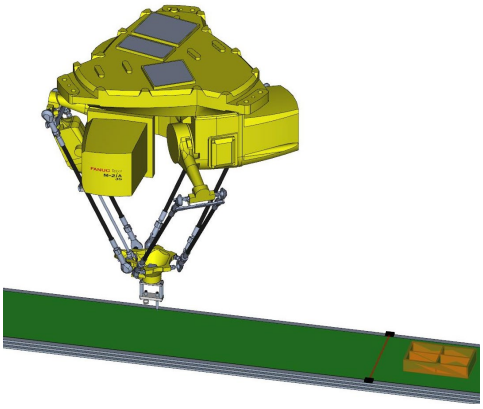


Fig. 4.2.7 (a) Setting a tray upstream from the sensor

11 Move the conveyor. When the tray comes within the operating range of the robot, stop the conveyor. The trigger count at the instant the sensor found a part will appear

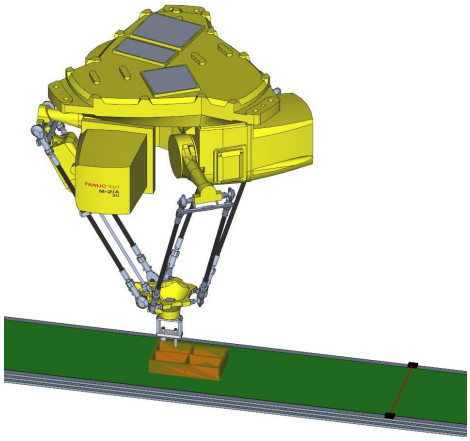
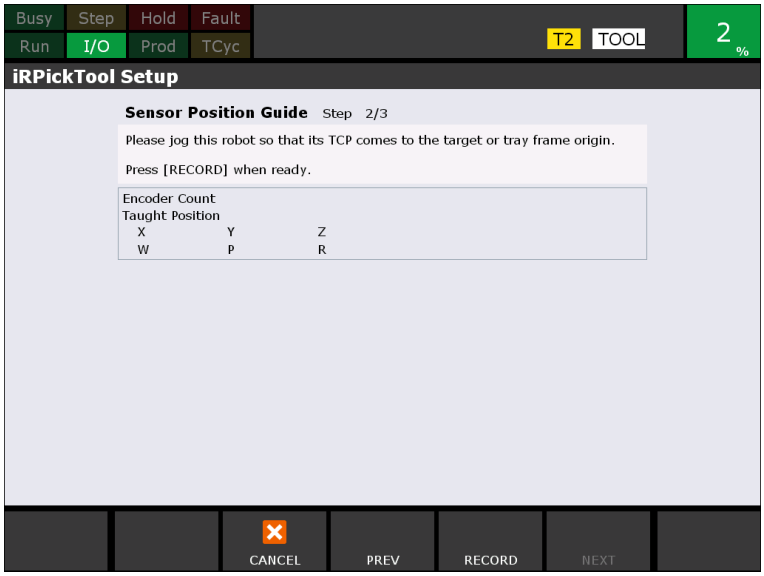


Fig. 4.2.7 (b) Stopping when the tray comes within the operating range of the robot

12 Press F5 [NEXT].

A screen similar to the one below will appear.



13 Touch up the origin of the tray.

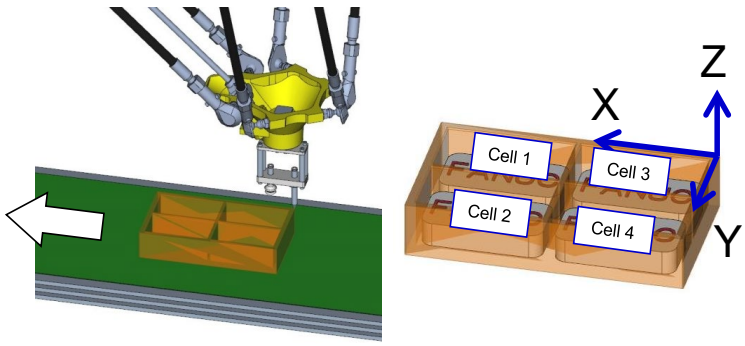
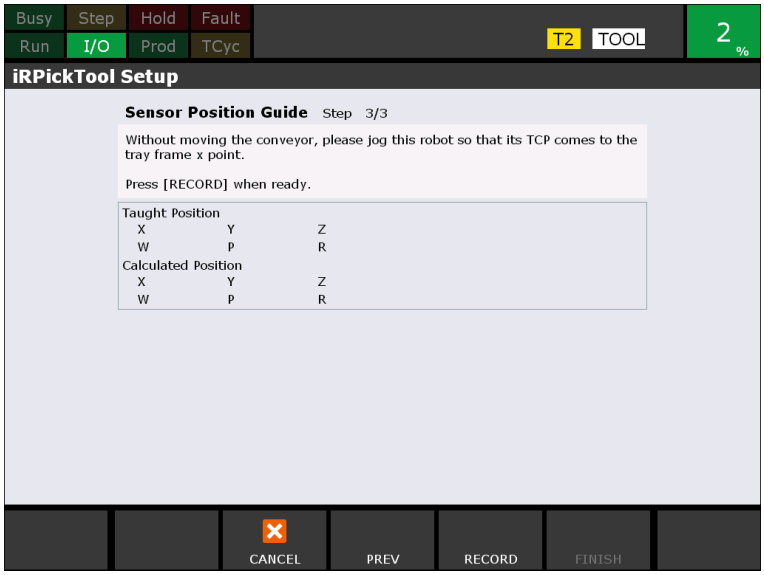


Fig. 4.2.7 (c) Touching up the origin of the tray

14 Press F4 [RECORD].
The current encoder count and taught position will appear.
15 Press F5 [NEXT].

A screen similar to the one below will appear.



16 Touch up the X-axis direction of the tray.

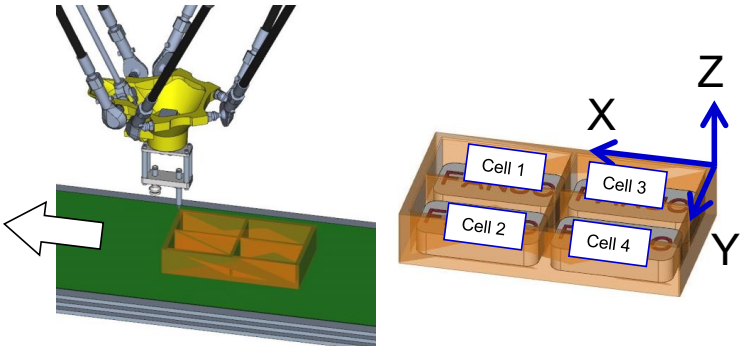


Fig. 4.2.7 (d) Touching up the X axis direction of the tray

17 Press F4 [RECORD].
The current encoder count and taught position will appear.
18 Press F5 [FINISH].

Return to the original setup screen. The values will populate the X, Y and R of Sensor Position.

4.2.8 Vision Setup

Set up the infeed conveyor vision.

4.2.8.1 Camera calibration

Perform camera calibration.

1 Prepare a calibration grid that is appropriate for the field of view of the camera.
Check the grid spacing on the lower right corner of the grid.

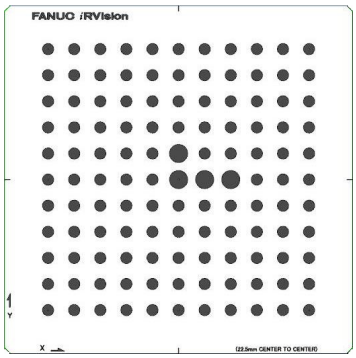


Fig. 4.2.8.1 (a) Calibration grid

⚠ CAUTION
The whole of the dot pattern does not have to appear in the image.
It is OK if some dots to lie outside the image. Adjust the grid position so that the dot pattern is distributed over the whole area in which the parts in the image are going to be found. If the dot pattern covers only part of the area in which the parts in the image are going to be found, precise calibration cannot be performed.

NOTE
There are also metal plate types for the jig for the fixing calibration grid in place. Please prepare something that is appropriate for your environment. We offer some standard products

2 Set the calibration grid so it is directly under the camera.
Fix it in place with tape, etc. to avoid misalignment.

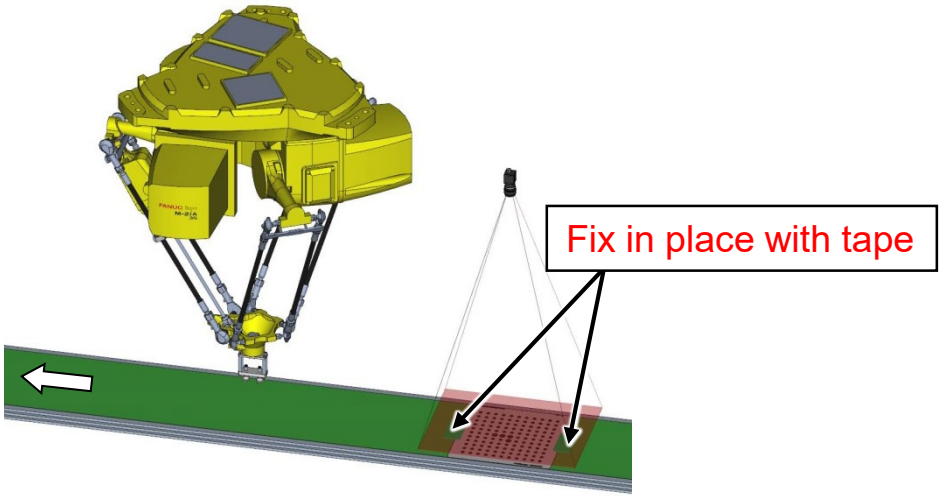


Fig. 4.2.8.1 (b) Setting a calibration grid

3 The direction of the calibration grid is arbitrary.
For example, in the case of line tracking, set the conveyor flow direction and the X direction of the calibration grid to be in the same direction. This does not need to be exact, but the grid must remain in position on the conveyor through the calibration procedure.

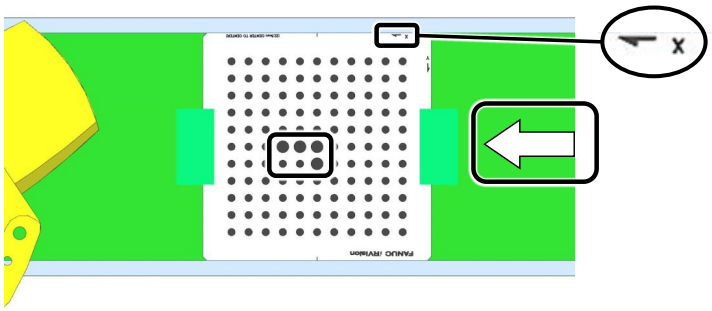
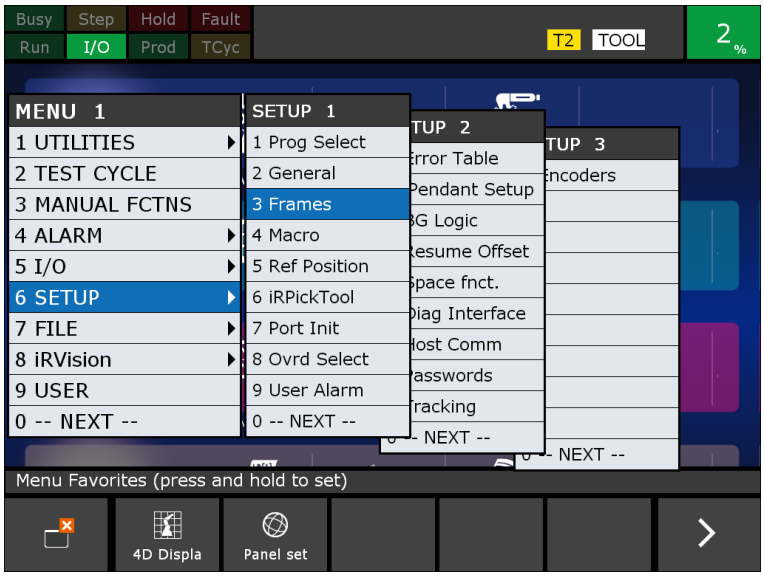


Fig. 4.2.8.1 (c) Align the setting direction

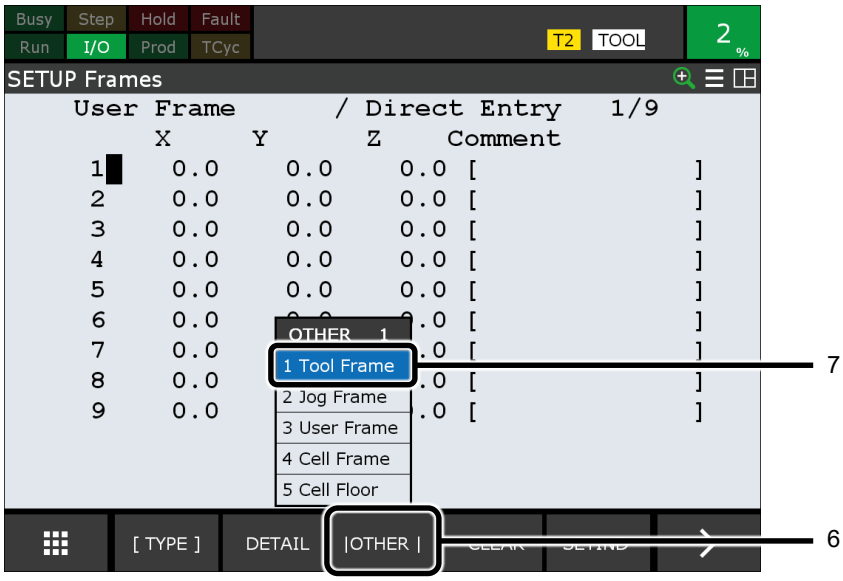
4 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.

5 Select [SETUP] → [Frames] from the menu.



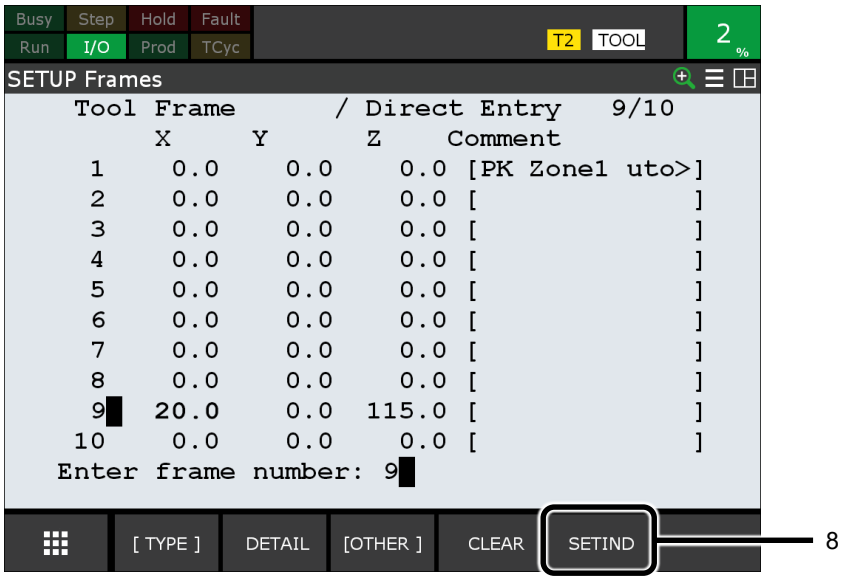
6 Press F3 [OTHER].

7 Select [Tool Frame] from the menu.
The list screen for tool frames will appear.



8 Press F5 [SETIND].

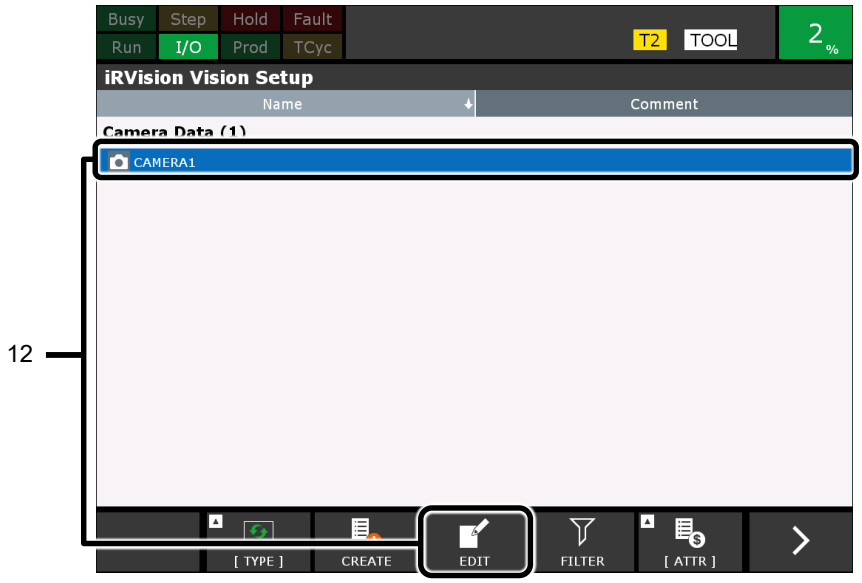
9 Enter the frame number that has been set for the pointer tool.
This procedure is explained using Tool 9 as the pointer as an example.



10 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.

11 From the menu, select [*i*RVision] → [Setup]. The [*i*RVision Vision Setup] screen will appear.

12 Select the camera data created in Section 4.1.2, “Checking the connection of the camera” and press F3 [EDIT].



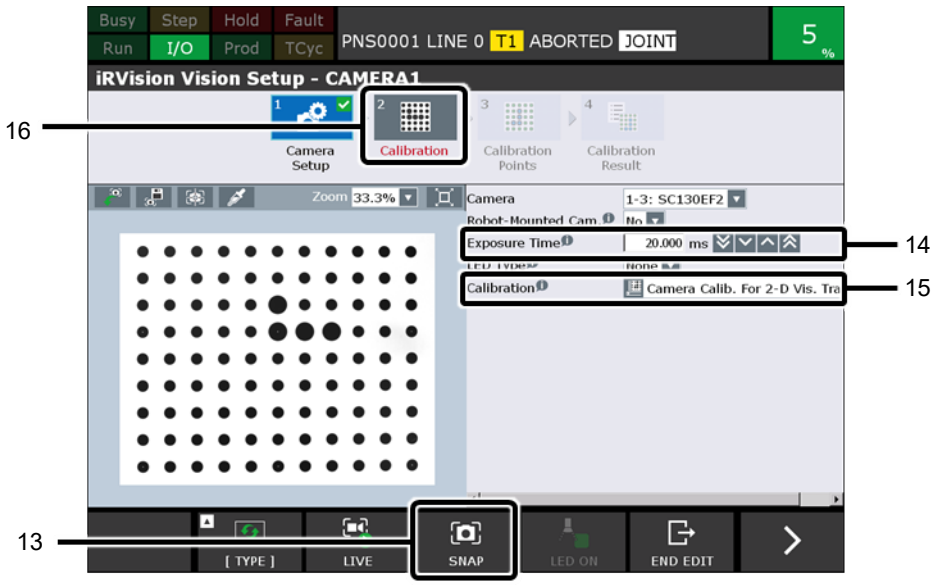
The camera setup screen will appear.

13 Press F3 [SNAP] and check that a picture of the calibration grid has been taken within the frame on the left side of the screen.

14 Enter an [Exposure Time].

15 For [Calibration], select [Camera Calib. For Vis. Track] from the menu.

16 Press [Calibration] at the top of the screen.



The calibration setup screen will appear.

17 Select [Conveyor Name].

18 Enter the [Grid Spacing] of the dot pattern.

19 Select [Perspective] in [Projection].
[Perspective] has been selected as the initial value.

⚠ CAUTION
[Orthogonal] can be selected only when the heights of the parts that are to be found are constant and the heights of the calibration grid surface and the upper surface of the parts that are to be found are the same.

20 In [Camera Distance], enter the focal distance of the lens that is being used.

⚠ CAUTION
With regard to the calculation method for [Camera Distance], when [Auto] is selected, it is in principal possible that the focal distance might not be calculated accurately, because the calibration grid on the conveyor and the camera's imaging surface are nearly parallel.

NOTE
By placing the calibration grid at an inclination, an accurate [Camera Distance] can be calculated using [Auto].

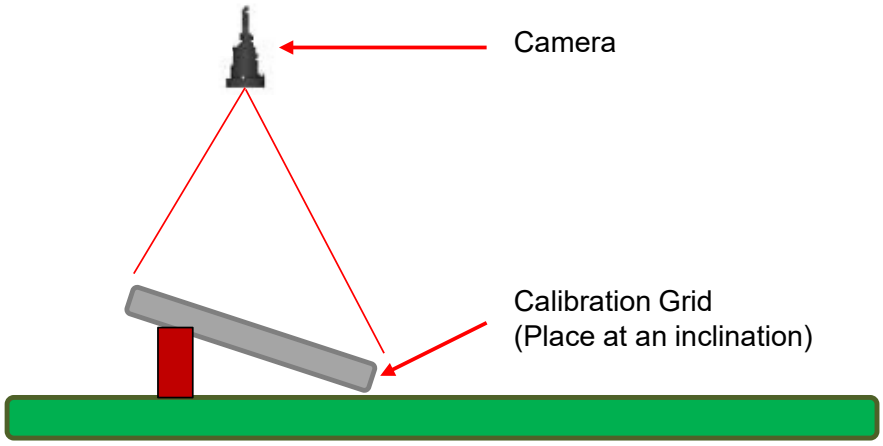
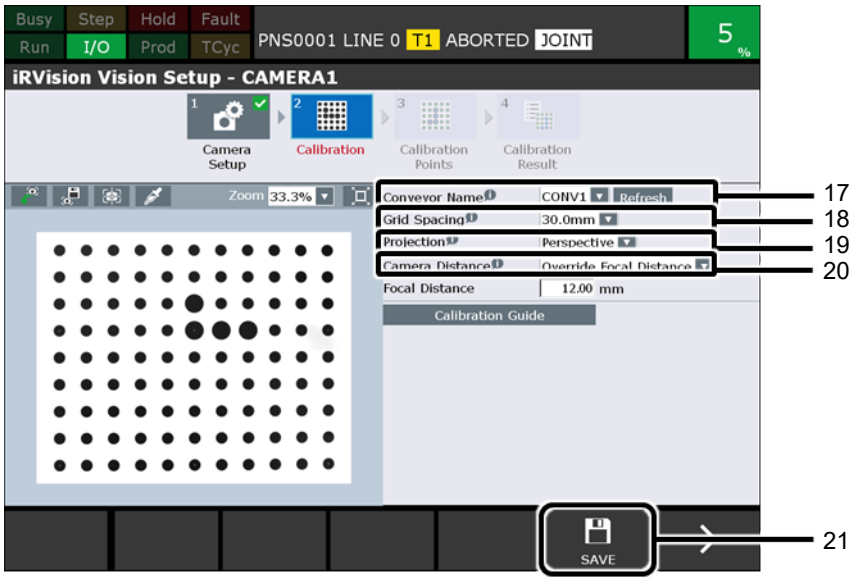
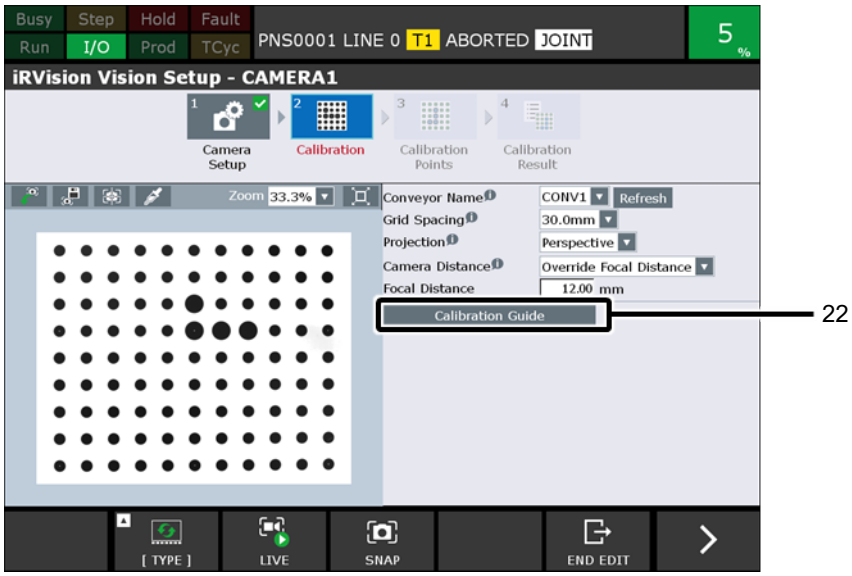


Fig. 4.2.8.1 (d)

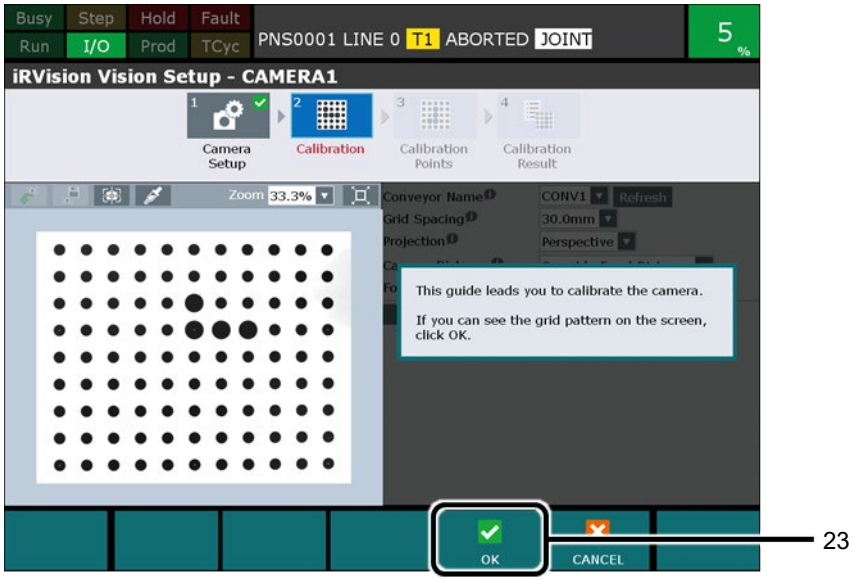
21 Press [> (Next page)] and press F5 [SAVE].



22 Press [Calibration Guide].

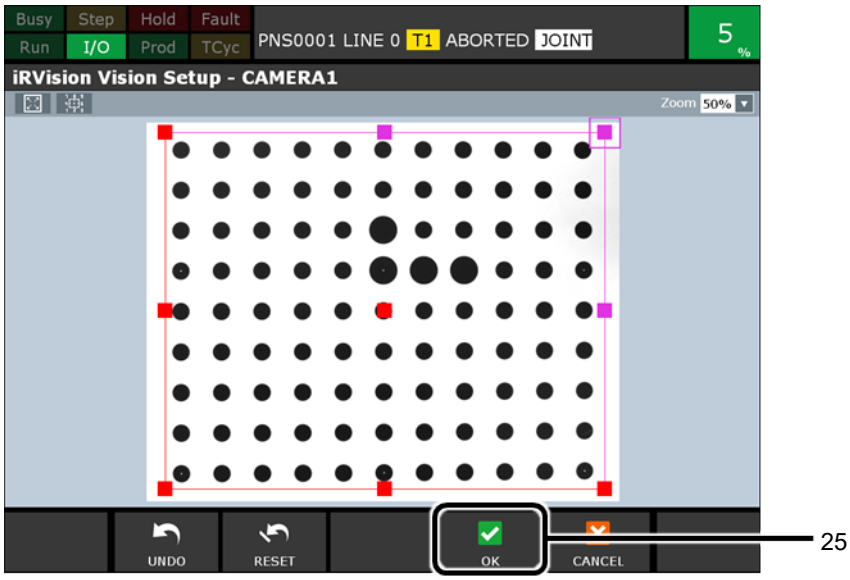


23 Check that the calibration grid appears on the screen, and press F4 [OK].

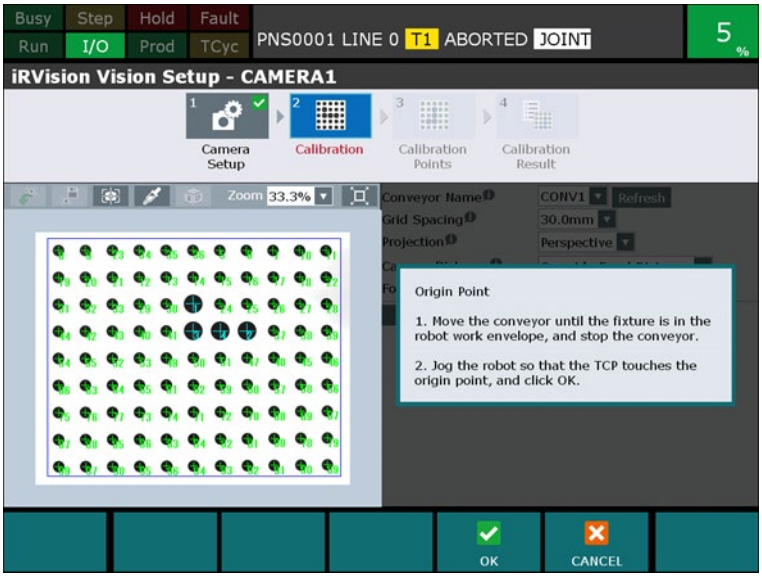


24 Enclose the dot pattern that appears on the screen with the reddish-purple rectangular window.

25 Press F4 [OK].



26 Move the conveyor. When the calibration grid is near the center of the operating range of the robot, stop the conveyor.



27 Touch up the origin of the calibration grid.

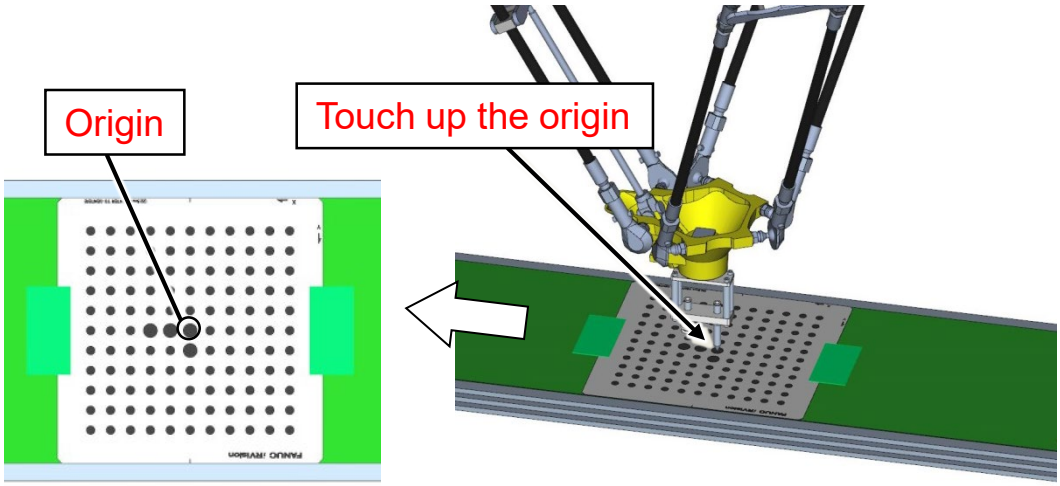


Fig. 4.2.8.1 (e) Touching up the origin of the calibration grid

28 Press F4 [OK].

29 Without moving the conveyor, touch up the X-axis direction of the calibration grid.

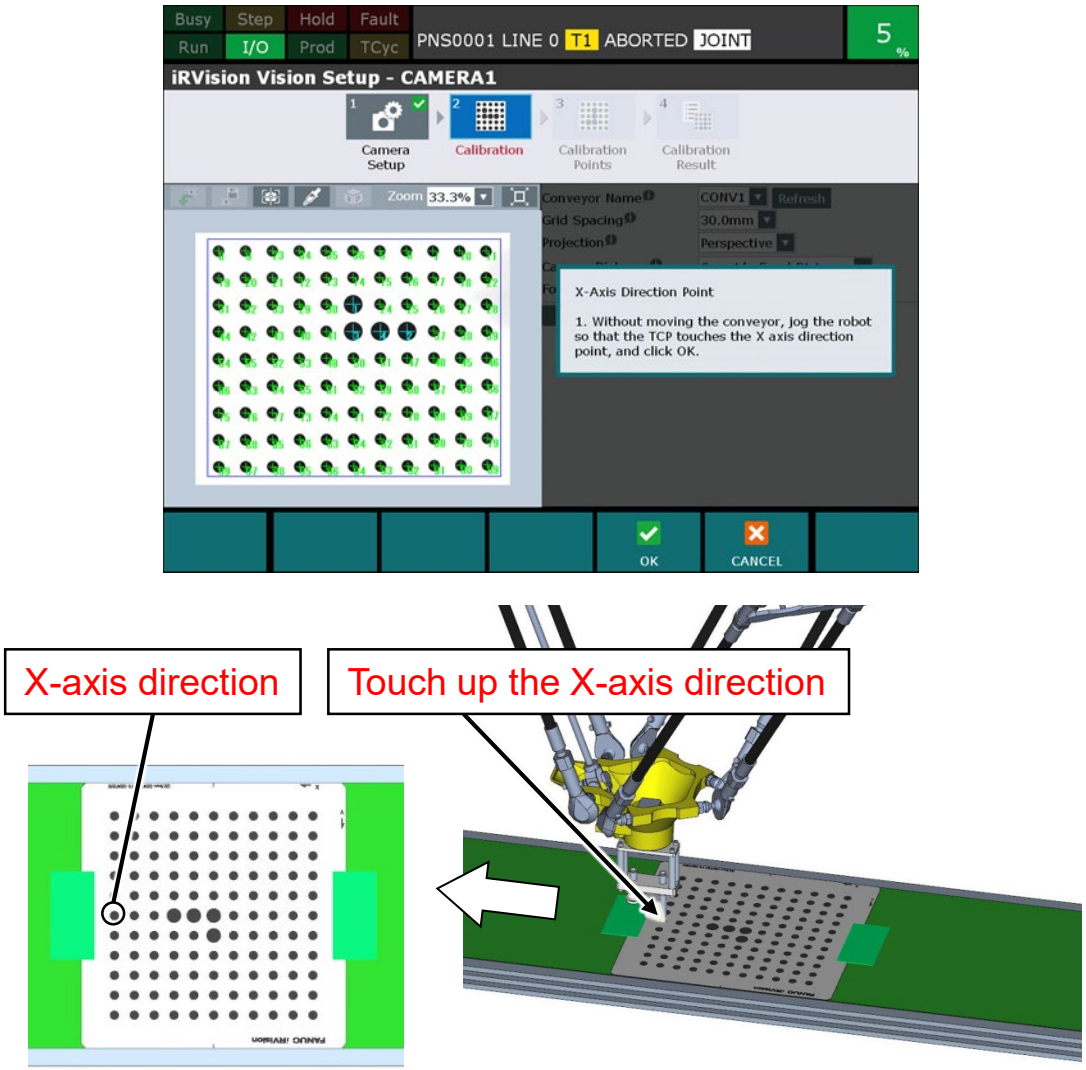


Fig. 4.2.8.1 (f) Touching up the X-axis direction of the calibration grid

30 Press F4 [OK].

31 Without moving the conveyor, touch up the Y-axis direction of the calibration grid.

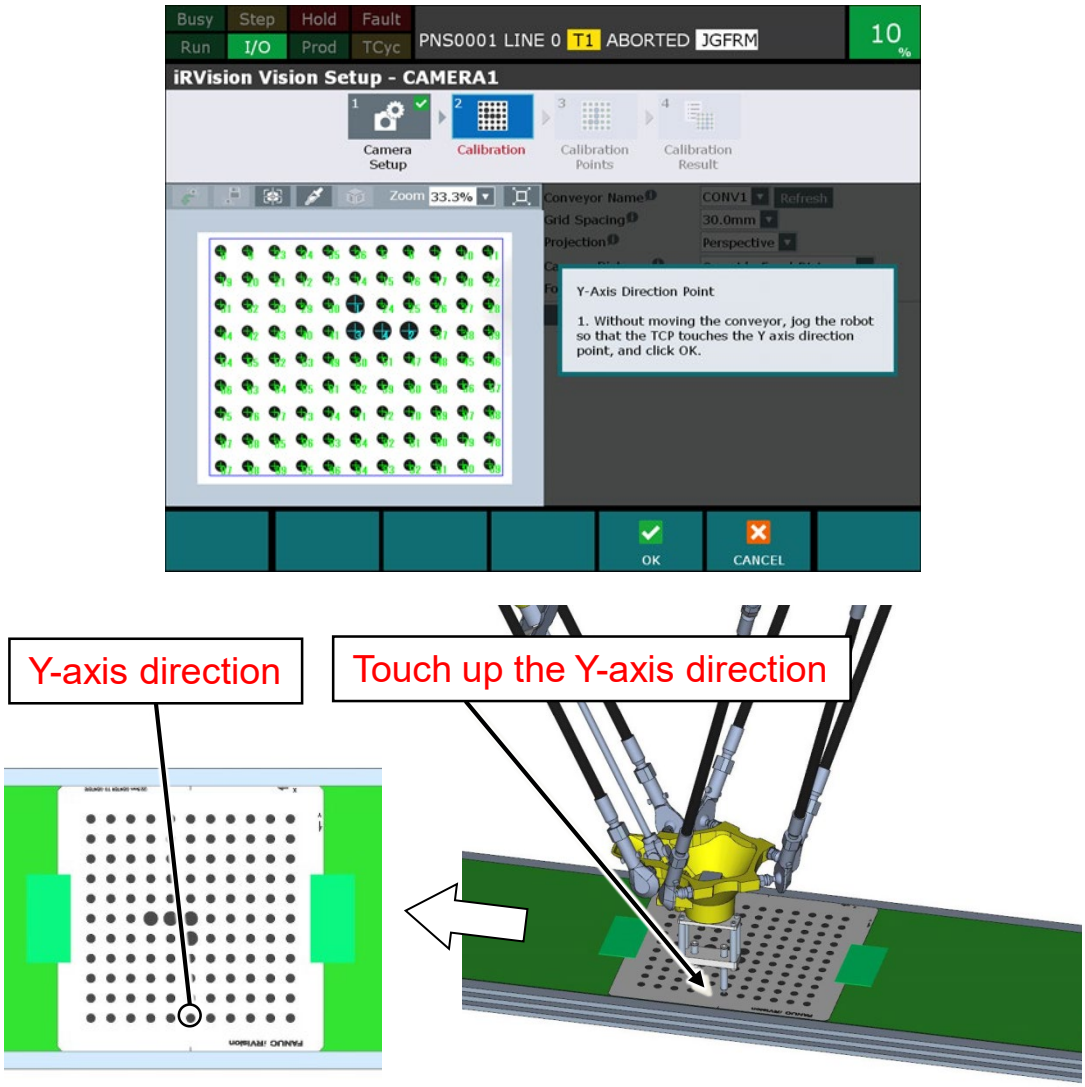
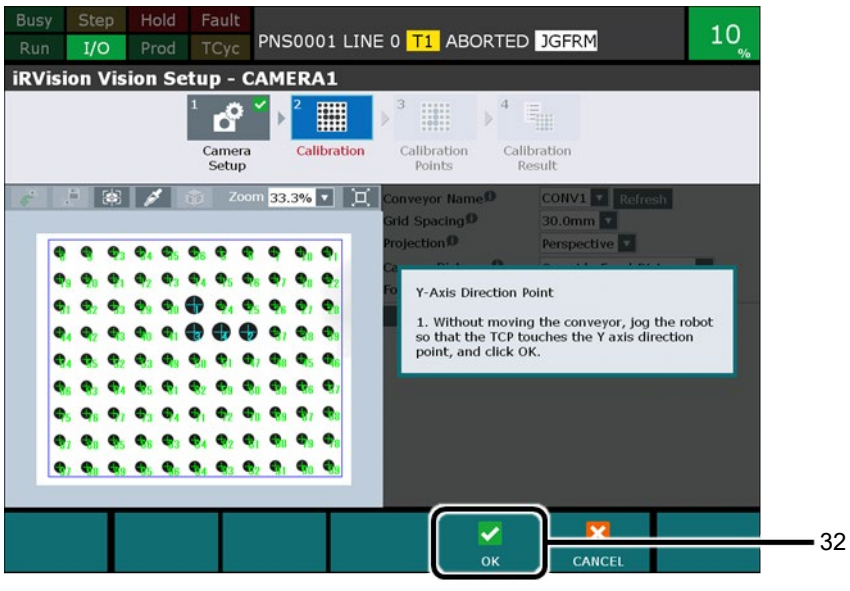
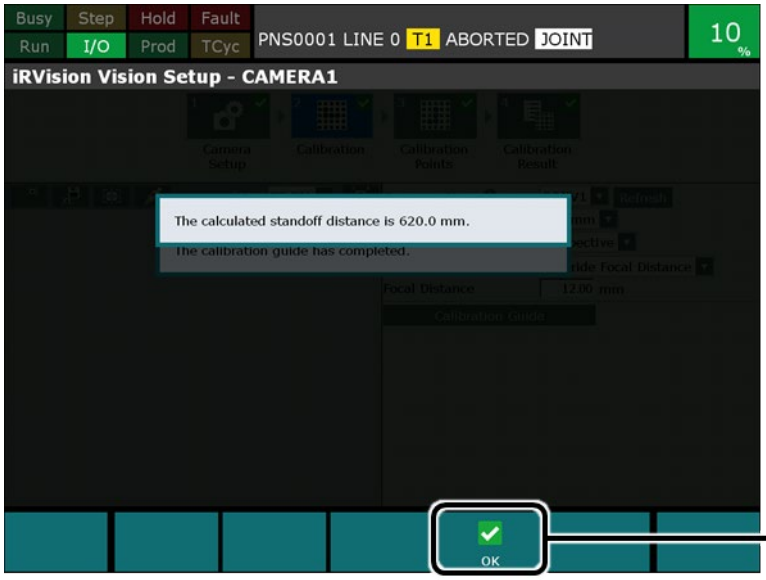


Fig. 4.2.8.1 (g) Touching up the Y-axis direction of the calibration grid

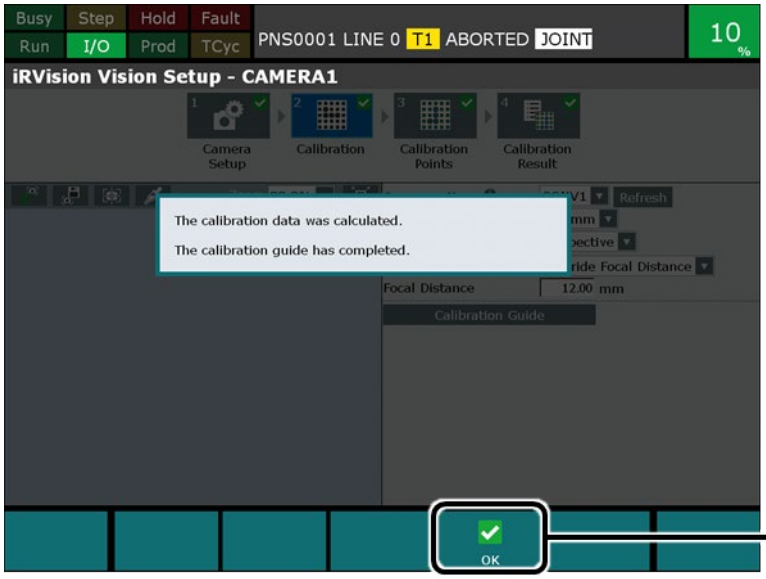
32 Press F4 [OK].



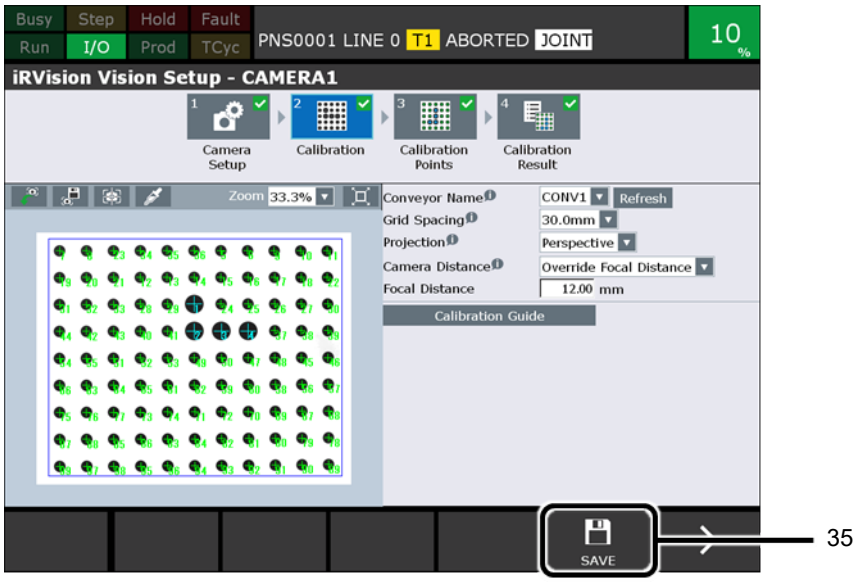
33 The calculated standoff distance is displayed. Confirm that the value is suitable for the actual standoff distance, and press F4 [OK]. If the value is unsuitable, check whether the setting procedure and set parameters are correct.



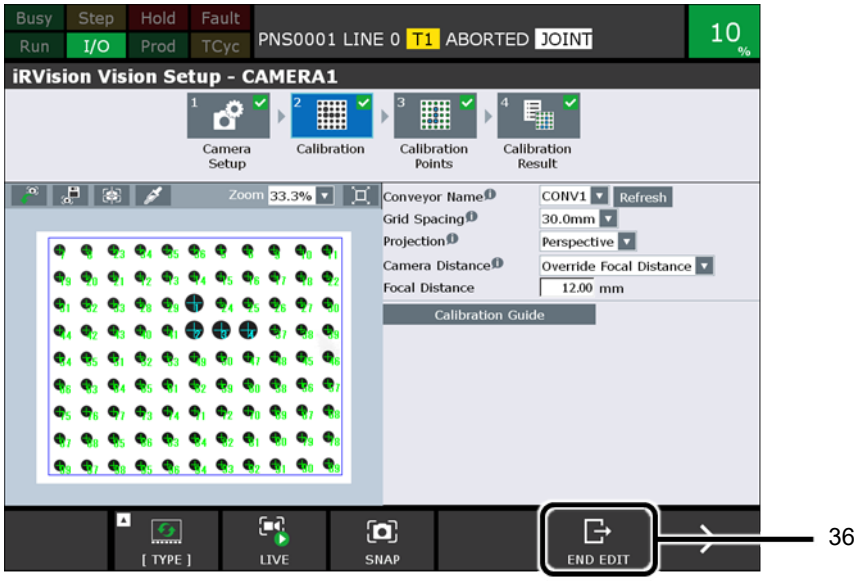
34 Press F4 [OK].



35 Press [> (Next page)] and press F5 [SAVE].



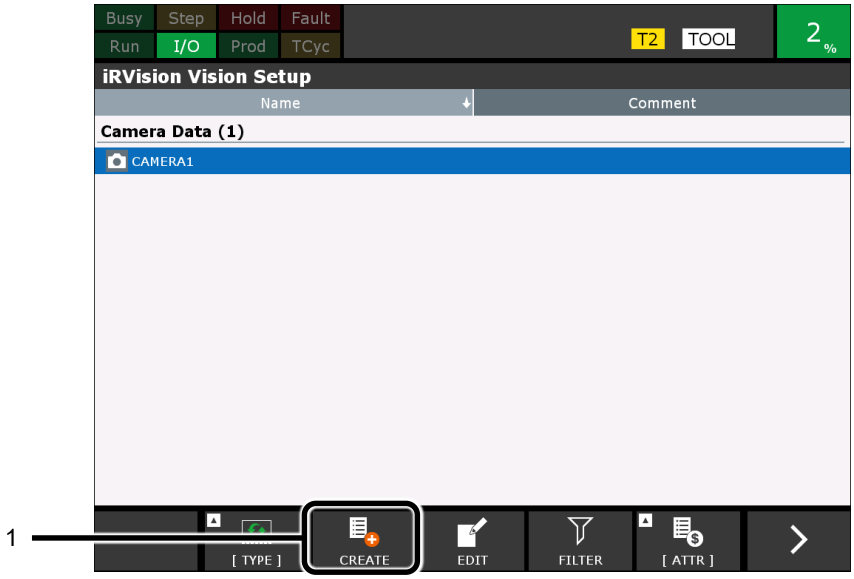
36 Press [> (Next page)] again and press F5 [END EDIT].



4.2.8.2 Vision process

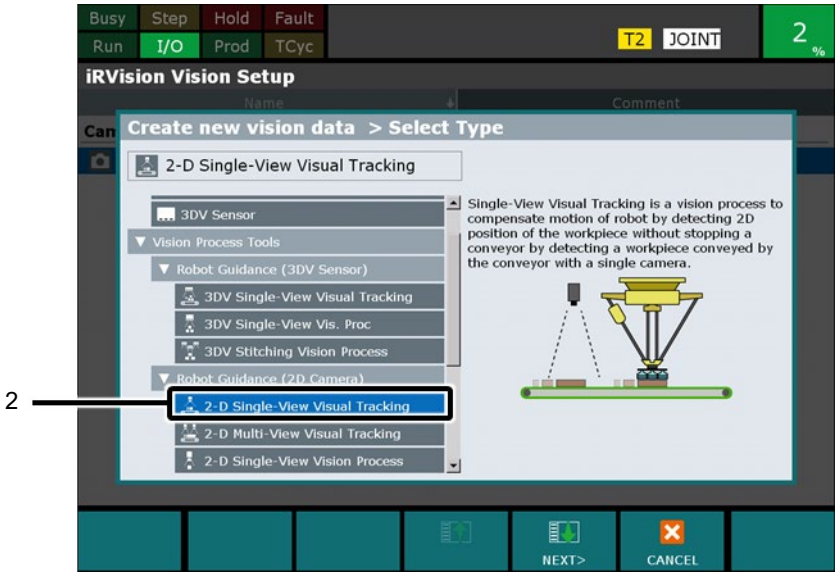
Set up a vision process.

1 Press F2 [CREATE].



The [Create new vision data] screen will appear.

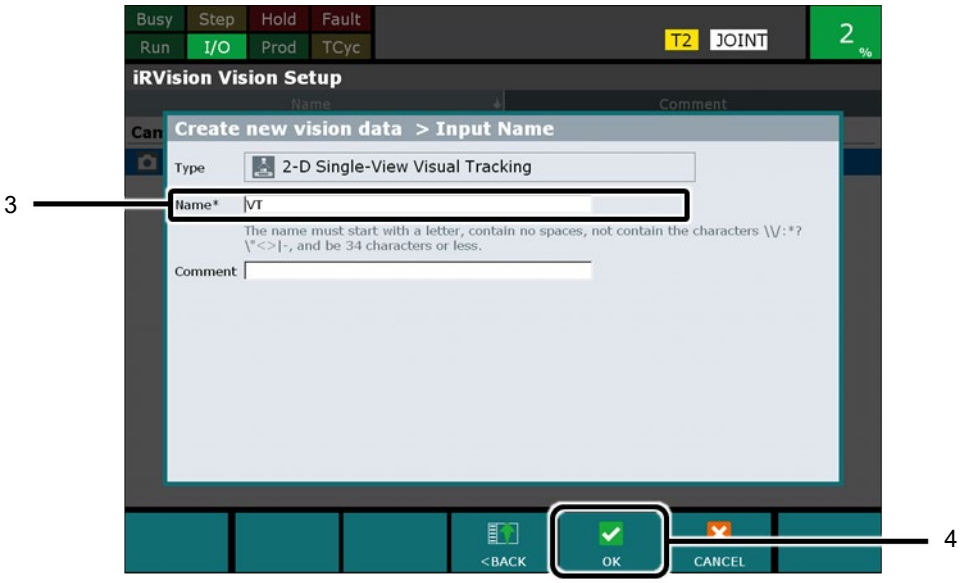
2 Select [2-D Single-View Visual Tracking].



A screen similar to the one above will appear.

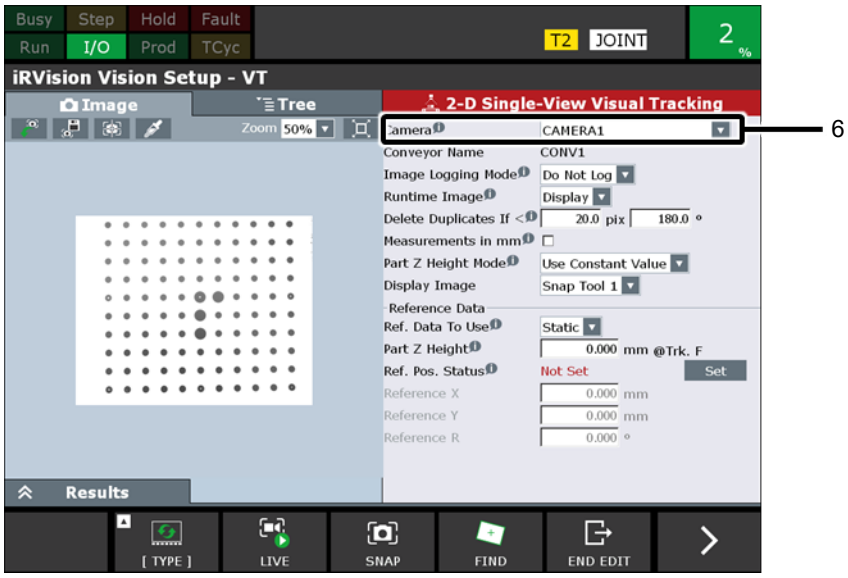
3 Enter the name of the vision process in [Name].
Input using half-width alphanumeric characters and half-width katakana. The name must contain no spaces, not contain the symbol other than underscore, and start with a letter.
Example : VT

4 Press F4 [OK].



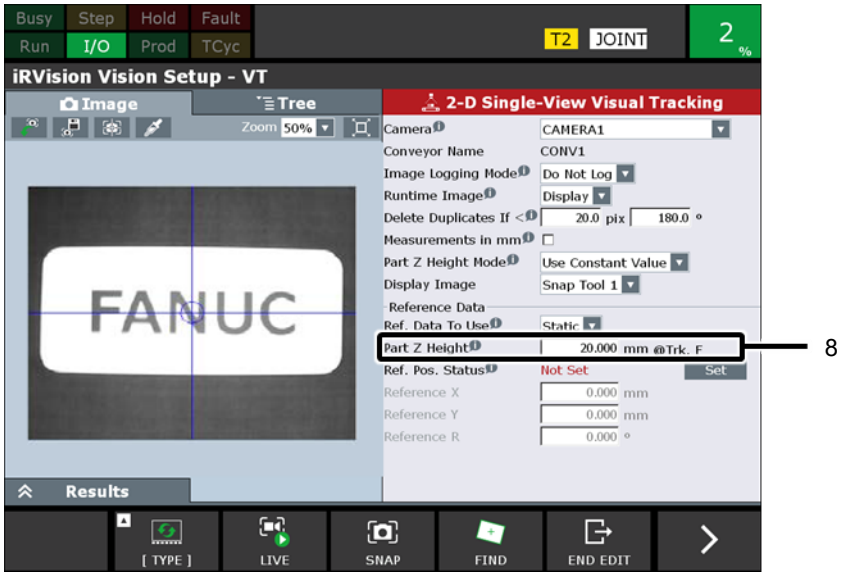
5 Select the created vision process and press F3 [EDIT].

6 Select [Camera] from the menu.



7 Measure the thickness of a part.
Measure the distance from the plane at which the tracking frame was taught (the calibration grid surface) to the plane where the GPM features that are to be found by vision will be.

8 Enter the thickness of the part measured in step 7 into [Part Z Height].
Enter the distance from the plane at which the tracking frame was taught (the calibration grid surface) to the plane where the features that are to be found by vision will be.



NOTE

[Part Z Height] is the distance from the plane at which the tracking frame was taught (the calibration grid surface) to the plane where the features that are to be found by vision will be. Measure it using a ruler, etc.
The relationship between the system and 'Part Z Height' is as shown in the following figure.

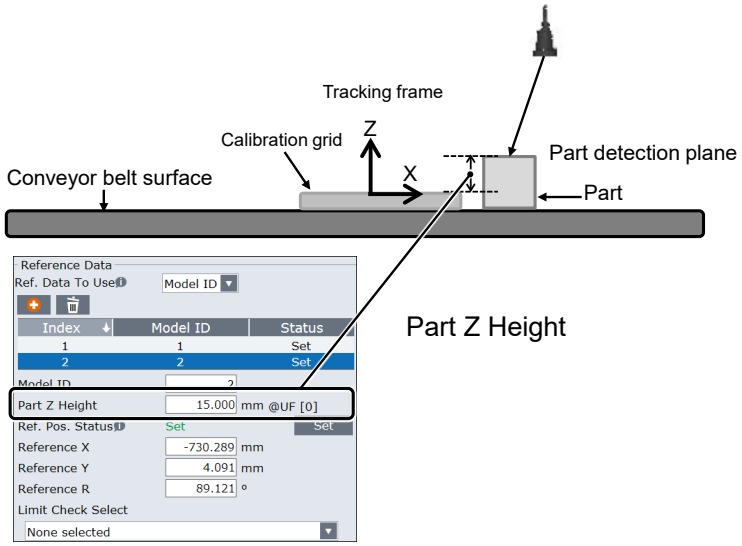
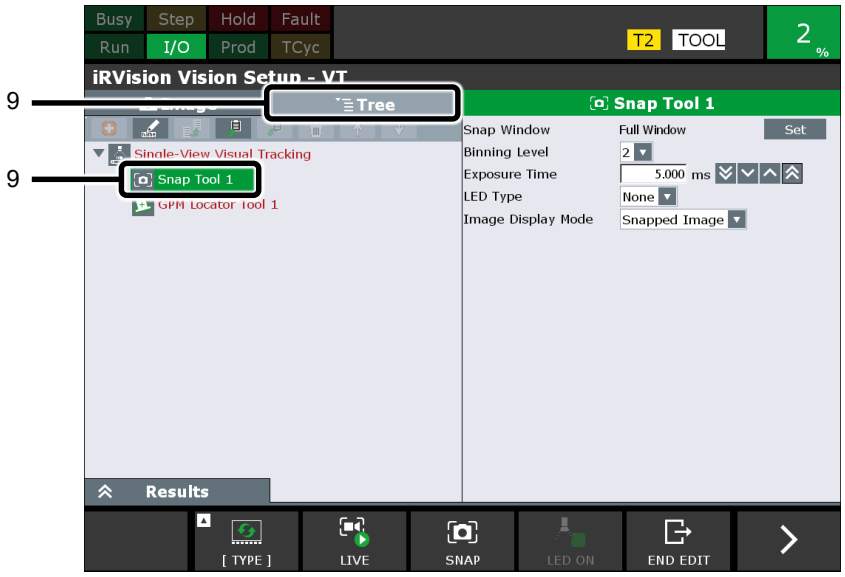


Fig. 4.2.8.2 (a) Relationship between the system and 'Part Z Height'

9 Press the tree tab and select [Snap Tool 1].



10 Enter an [Exposure Time].
Set the exposure time so that an image of a part that moves during exposure will have as little blurring as possible.
For example, taking into account image blurring, in order to suppress misalignment during finding by vision to less than 0.1 mm when the conveyor speed is 100 mm/s, set a value that is below 1 ms as the exposure time.
0.1 (mm) ÷ 100 (mm/s) = 1 (ms)
For information on calculation of an appropriate exposure time to suit the conveyor speed, refer to "Exposure time" in Section 2.5, "STUDY FOR APPLICATION".

11 Set a part in the field of view of the camera.
If the feed orientation is fairly consistent, align the.part in this orientation
If the feed part variation is ±180 degrees, teaching may be easier if the orientation is aligned with the placing operation.

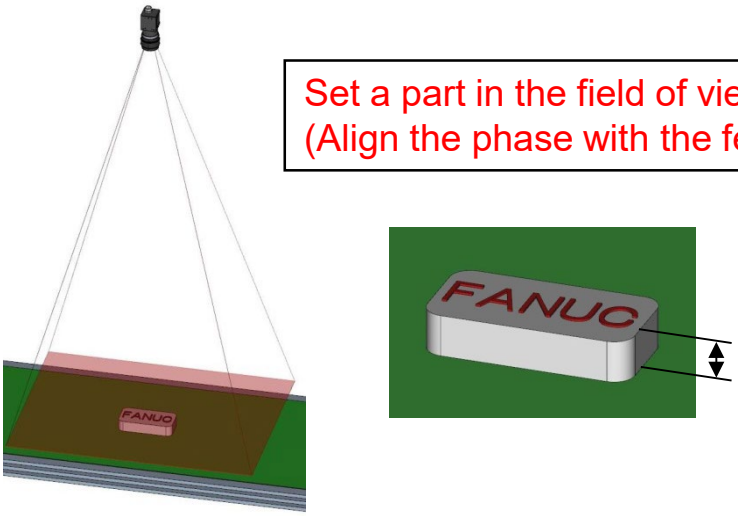
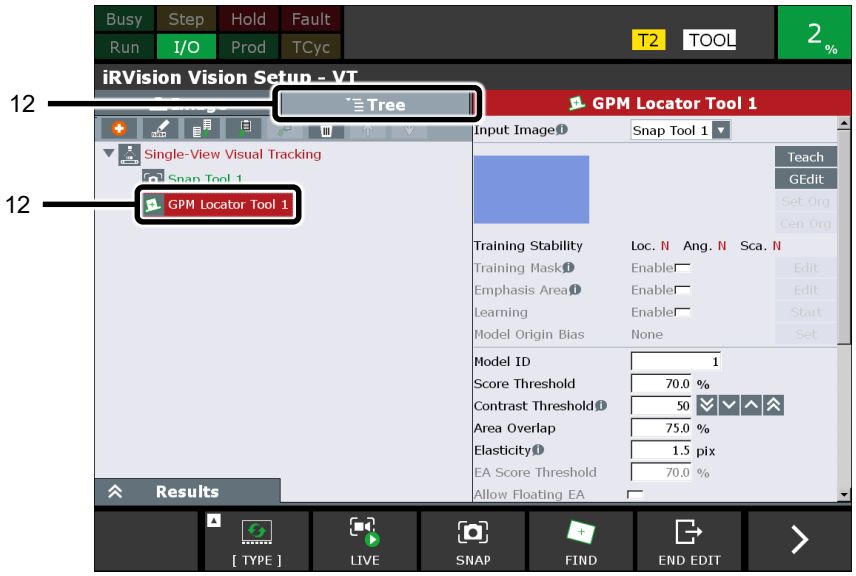


Fig. 4.2.8.2 (b) Setting a part

12 Press the tree tab and select [GPM Locator Tool 1].



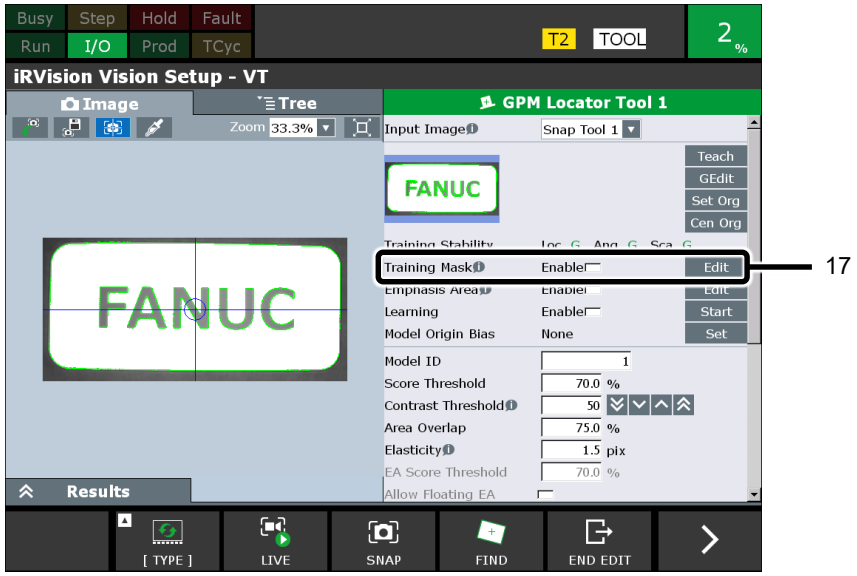
13 Press F3 [SNAP].
Verify that the part appears on the screen.

14 Press [Teach].

15 Enclose the part in the reddish-purple rectangular window.

16 Press F4 [OK].

17 Press [Edit] for [Training Mask].
If this option is not visible, scroll the screen downwards.

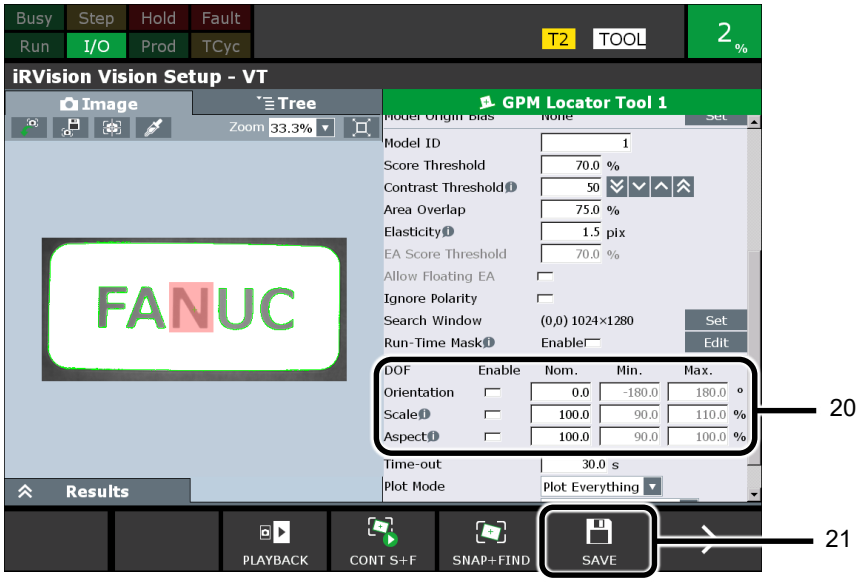


18 Select a tool at the top of the screen, and paint the section you do not want to make a feature for matching in red.

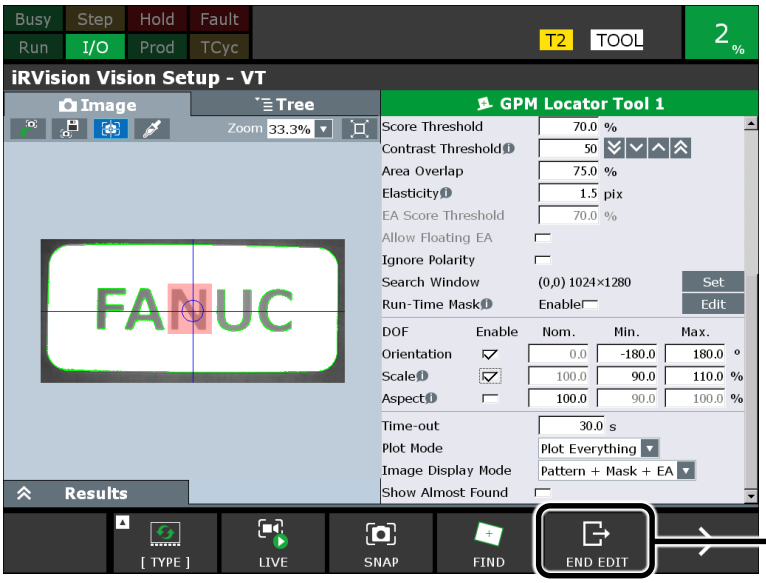
19 Press F4 [OK].

20 Set up the DOF as required.
For details on the setup, refer to the section, “GPM LOCATOR TOOL” in the chapter “COMMAND TOOLS” in the “*i*RVision OPERATOR'S MANUAL (Reference)”.

21 Press [> (Next page)] and press F5 [SAVE].



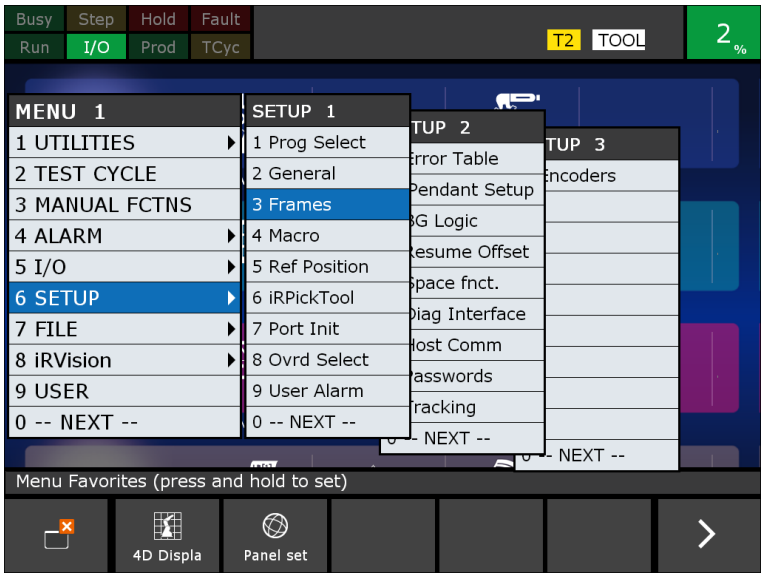
22 Press [> (Next page)] again and press F5 [END EDIT].



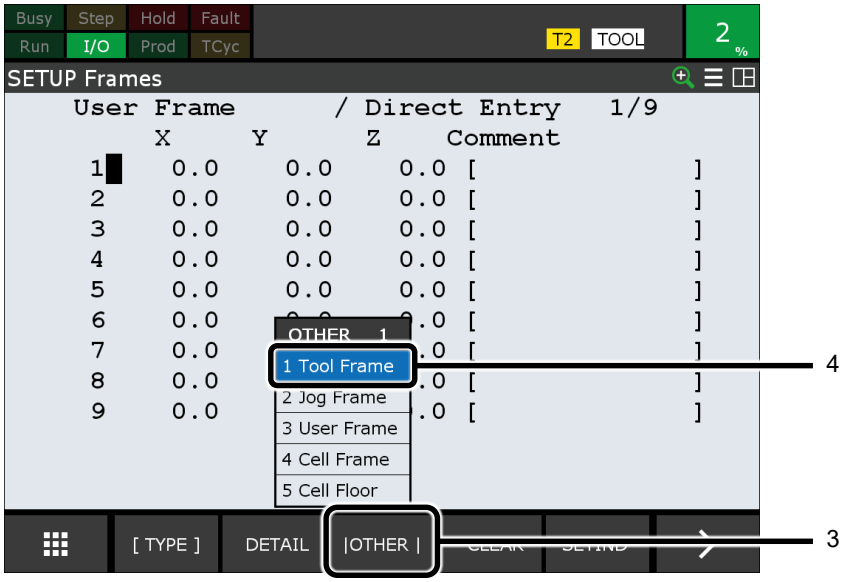
4.2.9 Setting up a Conveyor Station

Set up a conveyor station.

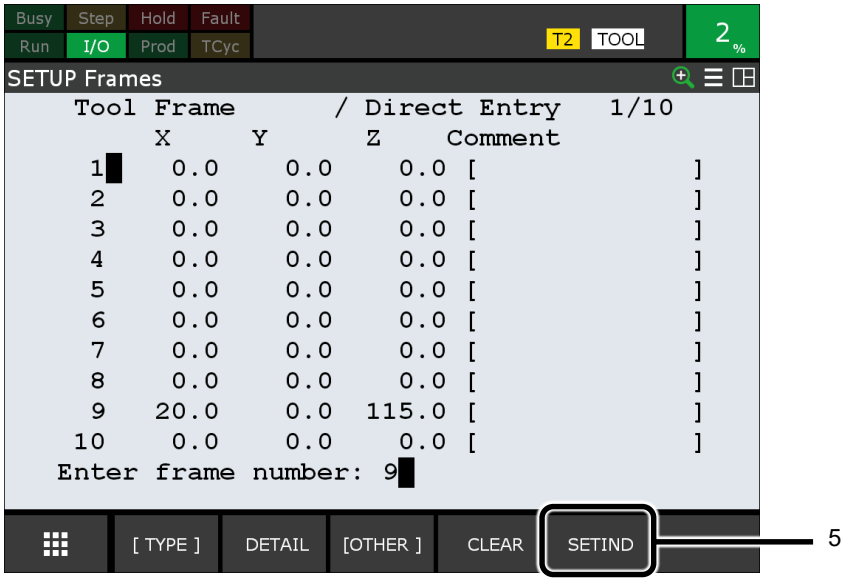
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SETUP] → [Frames] from the menu.



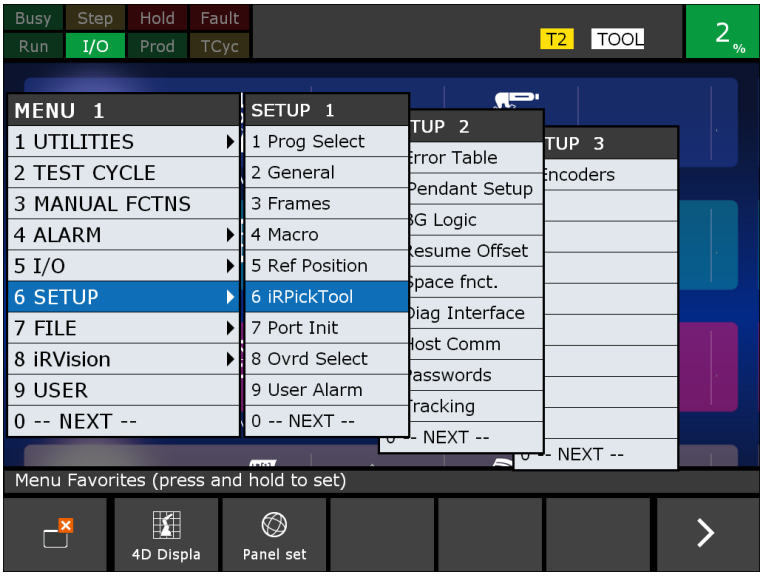
3 Press F3 [OTHER].
4 Select [Tool Frame] from the menu.
The list screen for tool frames will appear.



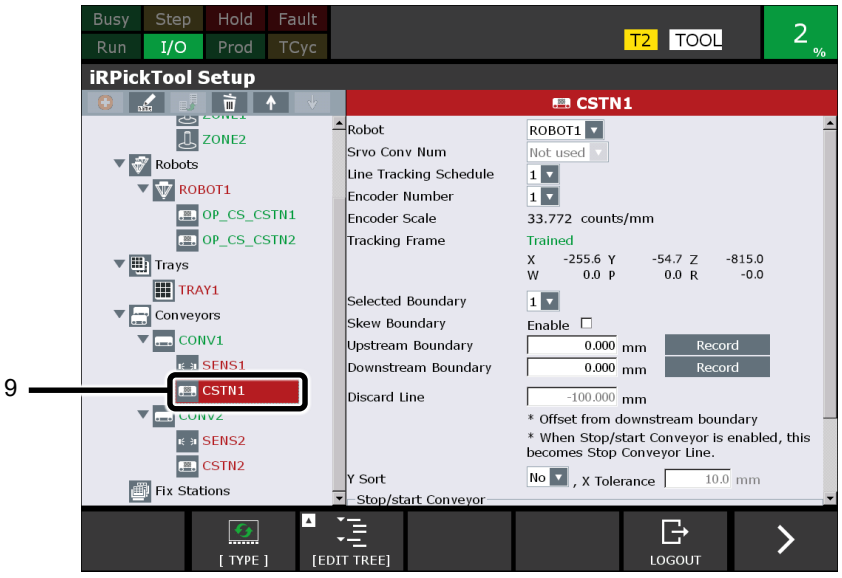
5 Press F5 [SETIND].
6 Enter the frame number that has been set for the pointer tool.
This procedure is explained using a case where Tool 9 is used as an example.



7 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
8 Select [SETUP] → [*i*RPickTool] from the menu.



9 First, set up the conveyor station of the infeed conveyor. Select the conveyor station [CSTN1].



10 Jog the robot to the upstream of the work area.

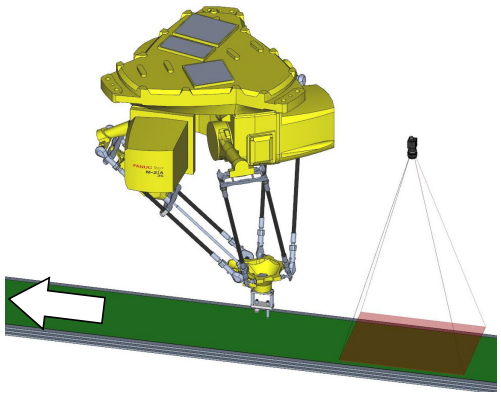


Fig. 4.2.9 (a) Jogging the robot

11 Click [Record] for [Upstream Boundary].
The current position of the robot in the tracking frame will be saved in [Upstream Boundary].

12 Jog the robot to the downstream of the work area.

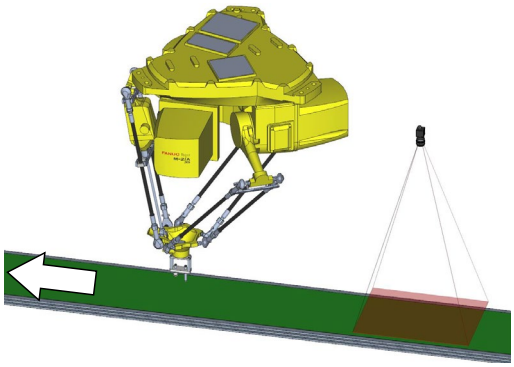


Fig. 4.2.9 (b) Jogging the robot

13 Click [Record] for [Downstream Boundary].
The current position of the robot in the tracking frame will be saved in [Downstream Boundary].

14 Enter [Discard Line].
The discard line value must be larger than the value calculated as below.

Time required for tracking* (sec) × Conveyor speed (mm/sec)

* The time required for tracking means the time from when the robot starts the tracking motion for a particular part to when the work on the part (picking, etc.) is complete.

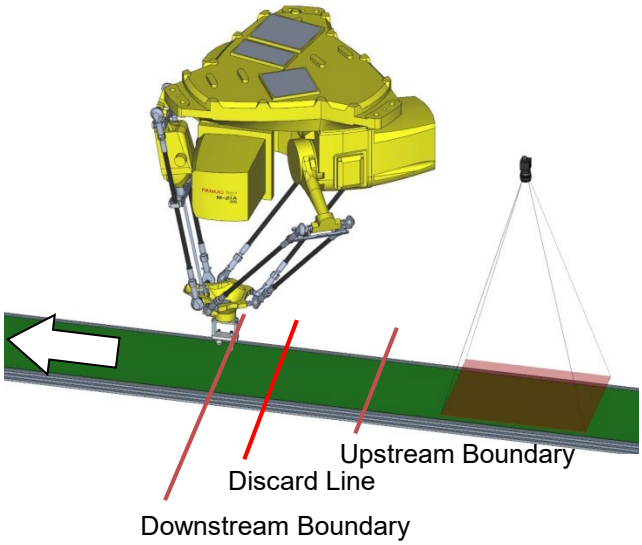
For example, if the robot's operation time is about 0.8 sec and the conveyor speed is 150 mm/sec, the free-running distance of the conveyor is

0.8 (sec) × 150 (mm/sec) = 120 (mm)

.

Therefore, allowing for a certain degree of error, set the discard line to around -150 mm.

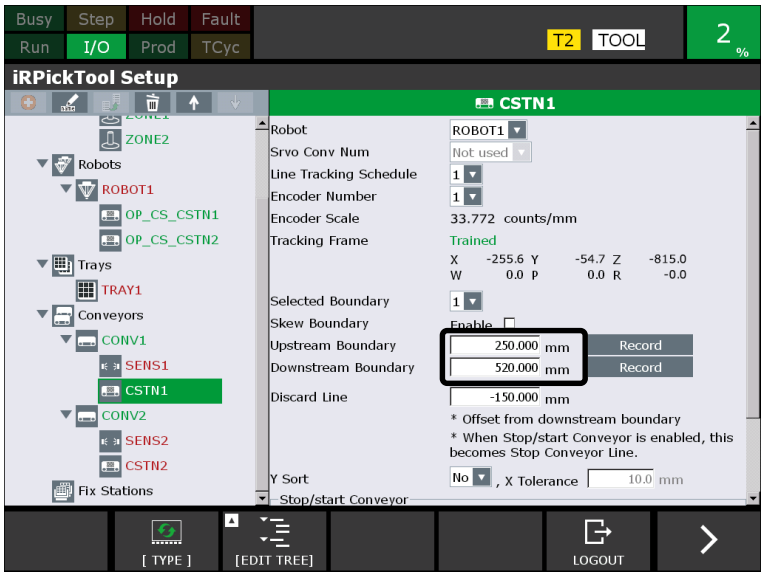
For details on discard lines, refer to Section 6.6, “SETUP OF A CONVEYOR STATION”.



Discard Line (mm) >
Time required for tracking (sec) × Conveyor speed (mm/sec)

Fig. 4.2.9 (c) Setting up a discard line

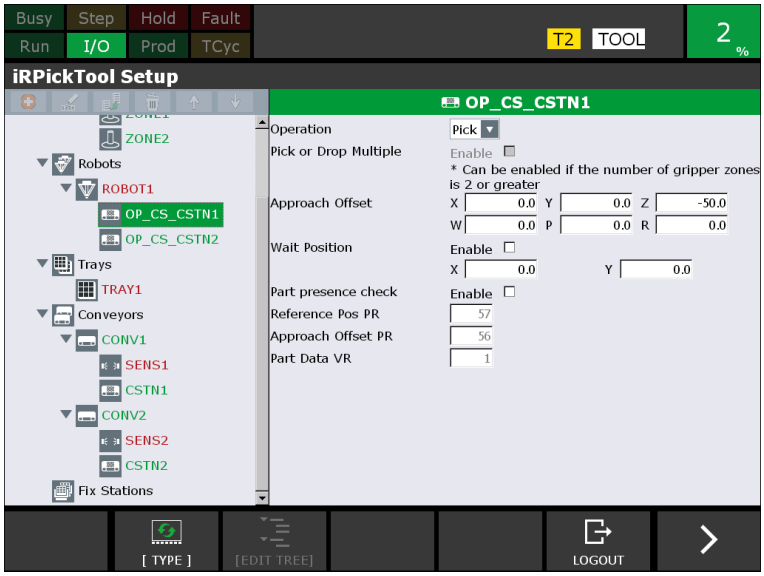
[Upstream Boundary] and [Downstream Boundary] can also be modified by direct input.



15 Next, set up the conveyor station of the outfeed conveyor. Select the conveyor station [CSTN2].
16 Repeat the steps 10 to 14 to set [Upstream Boundary], [Downstream Boundary], and [Discard lLine] for the conveyor station of the outfeed conveyor.

4.2.10 Setting up Operations

Set up operations.



For a conveyor station operation, set the following items.

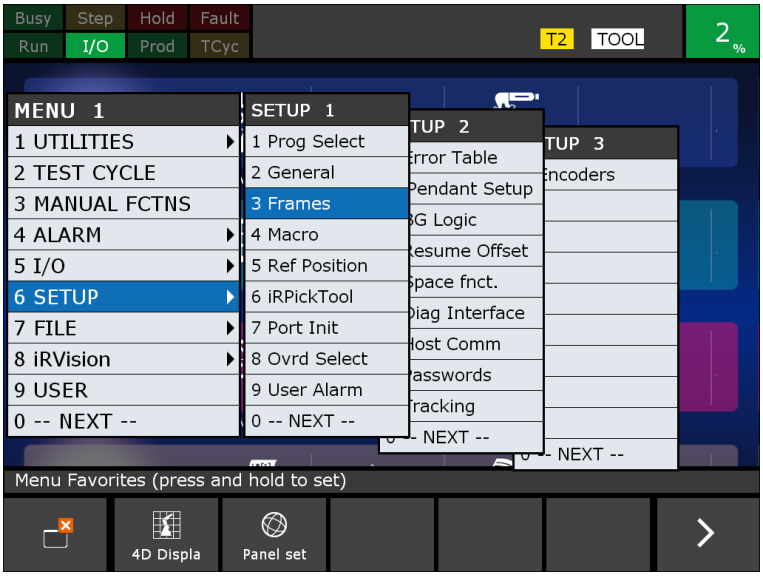
Item		Setting details
Operation		Select whether the task the robot executes at the conveyor station will be [Pick] or [Place].
Pick or Drop Multiple		If you enable [Pick or Drop Multiple], the robot will simultaneously pick/place the same number of parts as the number of zones that was added to the gripper. If it is not enabled, the robot will pick each part individually(sequentially).
Approach Offset		Input an offset for the approach position relative to the pick / place position, using a value in the tool frame. In most applications, only Z offsets are set.
Wait Position		If you have enabled a wait position, then if there is no part to pick / place, the robot moves to the wait position calculated in accordance with the settings Wait Position X, Y, which are explained below, and waits for a part. A standard wait position is at a height which takes into account the pick/place position and the Z value of the approach position offset, and is on the upstream boundary of the tracking zone of the conveyor station. If you move the upstream boundary, the wait position will update automatically with it. If a wait position is not used, the robot will wait at the release position of the previous cycle when the pick or place operation was completed.
	X	Set the shift amount for the wait position in millimeters. This is the shift amount in the X direction of the tracking frame of the corresponding conveyor station. When you input a positive value, the wait position shifts in the downstream direction along the conveyor flow. When you input negative value, the wait position moves upstream.
	Y	Set the value for the shift amount of the wait position in millimeters. This is the shift amount in the Y direction of the tracking frame of the corresponding conveyor station.
Part presence check		Set enable/disable for the part presence check for when a robot picks / places a part at the corresponding conveyor station.
Reference Pos PR		This value cannot be changed.
Approach Offset PR		This value cannot be changed.
Part Data VR		This value cannot be changed.

4.2.11 Setting up a Tool Frame for the Hand

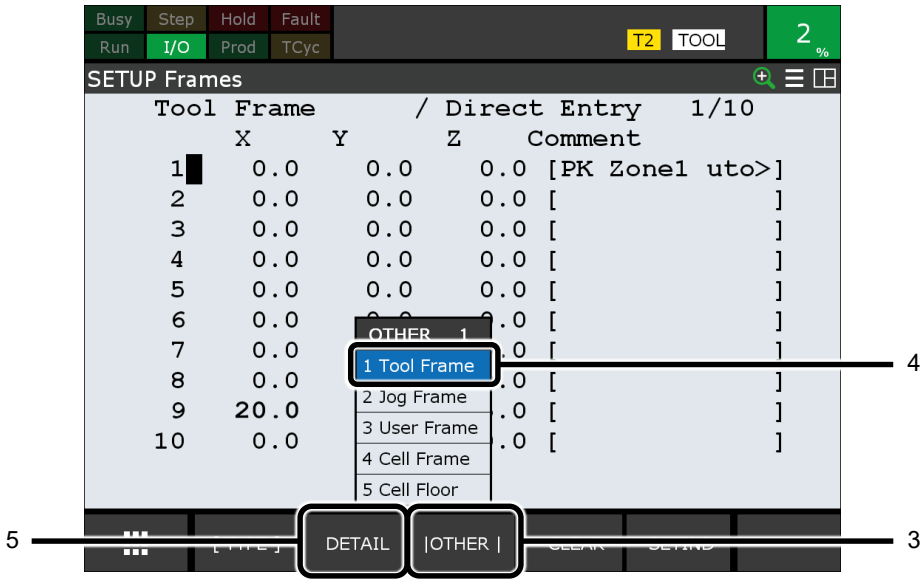
Set up a tool frame for the hand.

In this section, a four-axis Genkotsu robot is used as an example.
With a four-axis Genkotsu robot, we recommend that X and Y of the tool frame be set to 0, even if (X, Y) of the tool center point (TCP) of the hand in the mechanical interface frame is not (0, 0). This is to prevent the cycle time from being delayed or the robot's failing to stay in its operating range, as a result of large movement by the major axes (J1 to J3 axes) when a part is picked from the conveyor.
To make the tool's Z-axis face upward, directly input the tool's orientation. Directly input the values of the rotation angles W, P, and R of the tool frame around the mechanical interface frame's X, Y, and Z axes.

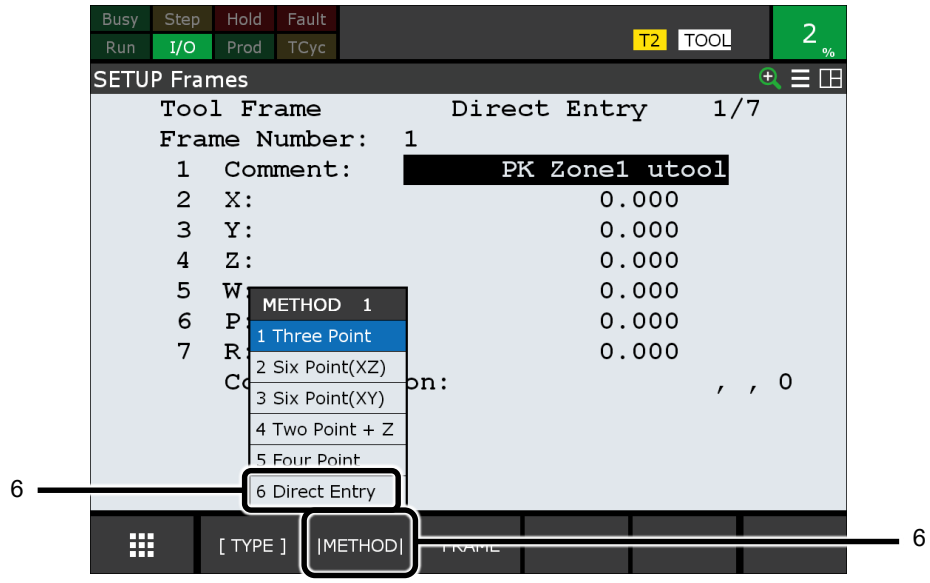
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SETUP] → [Frames] from the menu.



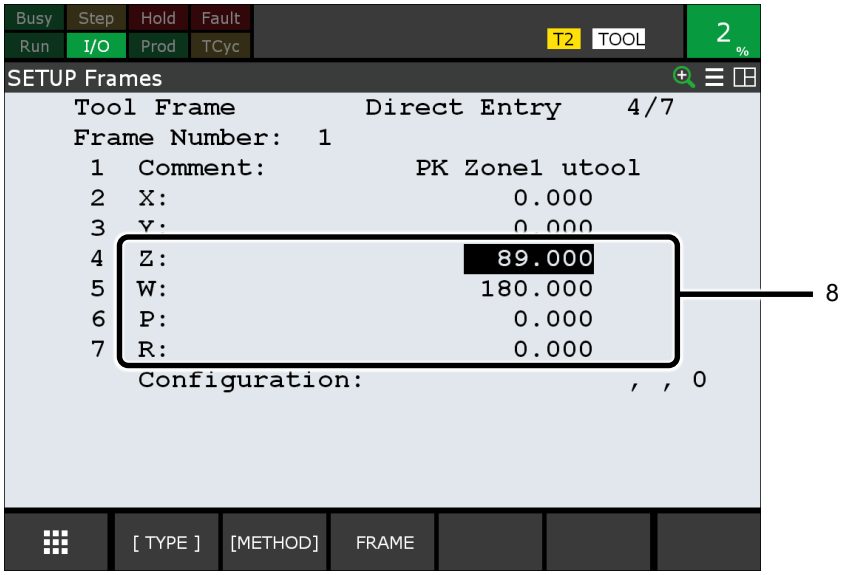
3 Press F3 [OTHER].
4 Select [Tool Frame] from the menu.
The list screen for tool frames will appear.
5 Place the cursor over Tool 1 and press F2 [DETAIL].
The tool frame setup screen for tool frame number 1 will appear.



6 Press F2 [METHOD].
7 Select [Direct Entry] from the menu.



8 Move the cursor to [Z], [W], [P] and [R] and enter a value for each.
For [Z], input a numerical value measured with a ruler, etc.
Input [W], [P] and [R] directly.
* (W, P, R) = (180, 0, 0) is recommended.

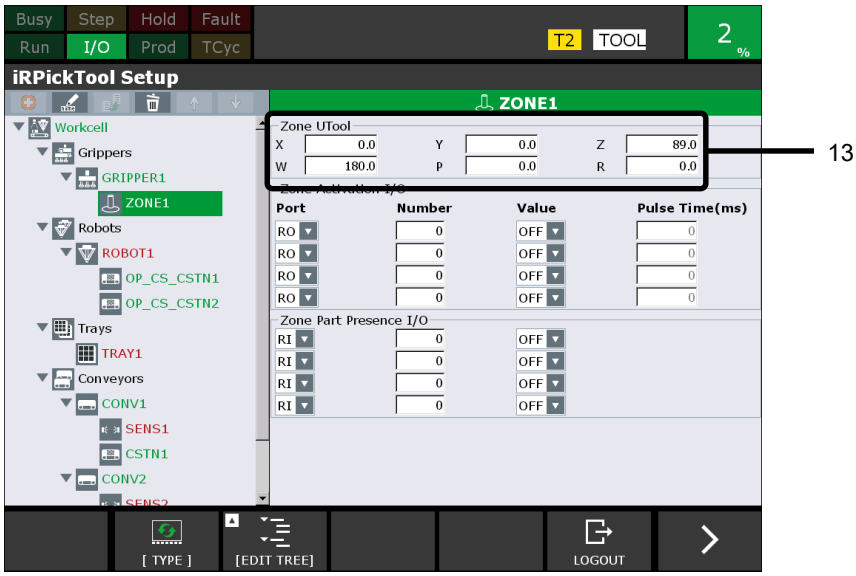


9 Make a note of the [Z], [W], [P] and [R] entered in step 8.
10 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.

11 Select [SETUP] → [*i*RPickTool].

12 Select the zone [ZONE1].

13 For [Z], [W], [P] and [R] of [Zone UTool], enter the same values that you entered in step 8 (the values you made a note of step 9).

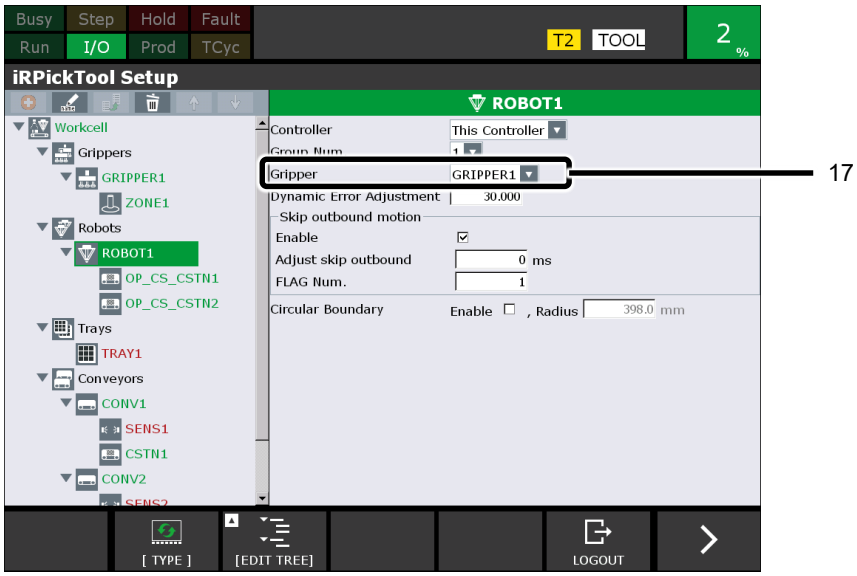


14 When using a multiple-item picking hand, set tool frame 2 and [ZONE2] by following the same procedure as steps 1 to 14.

15 In the same manner, repeat the procedure for tool frame 3 and [ZONE3], tool frame 4 and [ZONE4] ... up to the number of zones to be used.

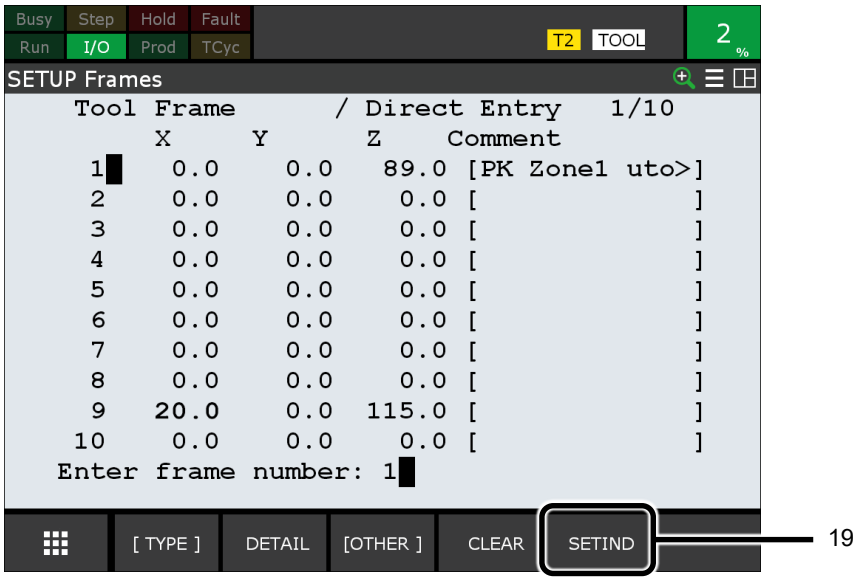
16 Select the robot [ROBOT1].

17 Select [GRIPPER1] for [Gripper].

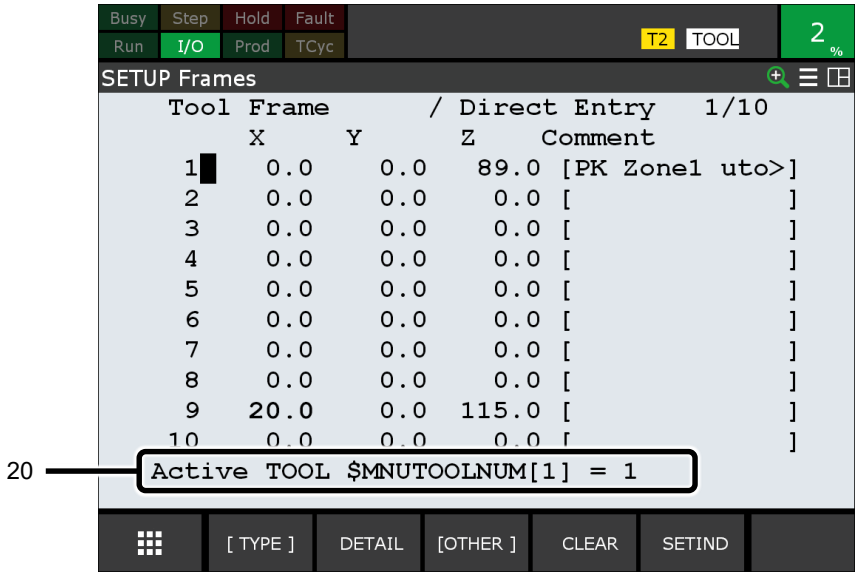


18 In the same way as steps 1 to 4, display the list screen for the tool frames.
The setting values of all the tool frames can be checked.

19 Press F5 [SETIND].



20 Enter tool frame number 1.
Tool frame 1 will be enabled.
* If you do not to carry out steps 19 and 20, tool frame 1 will not be enabled.



4.2.12 Setting up a Reference Position and Teaching a Robot Position

Set up a reference position, and teach a robot position.

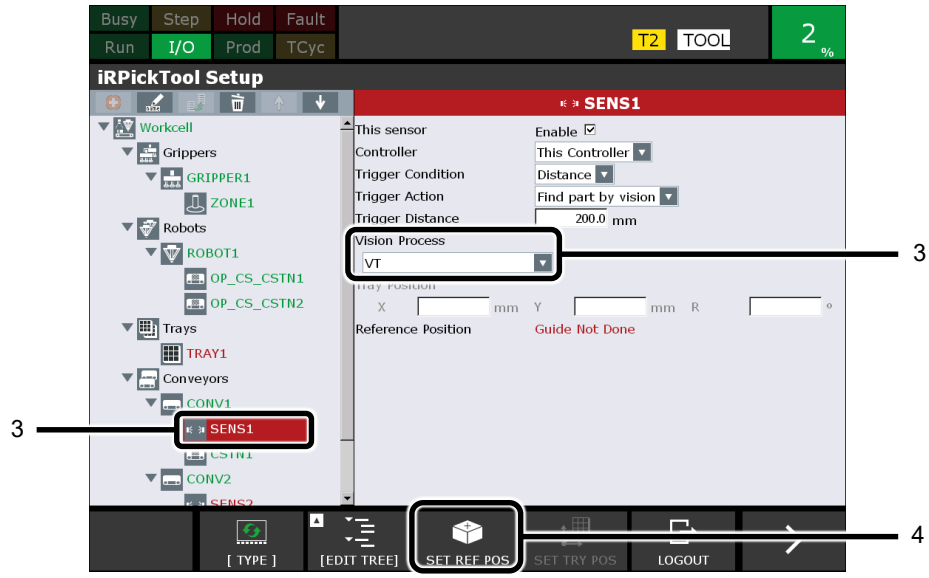
4.2.12.1 Setting up a reference position for the infeed conveyor, and teaching a robot position

Set up a reference position for the infeed conveyor, and teach a robot position. The procedure is for the case of [Distance] + [Find part by vision]

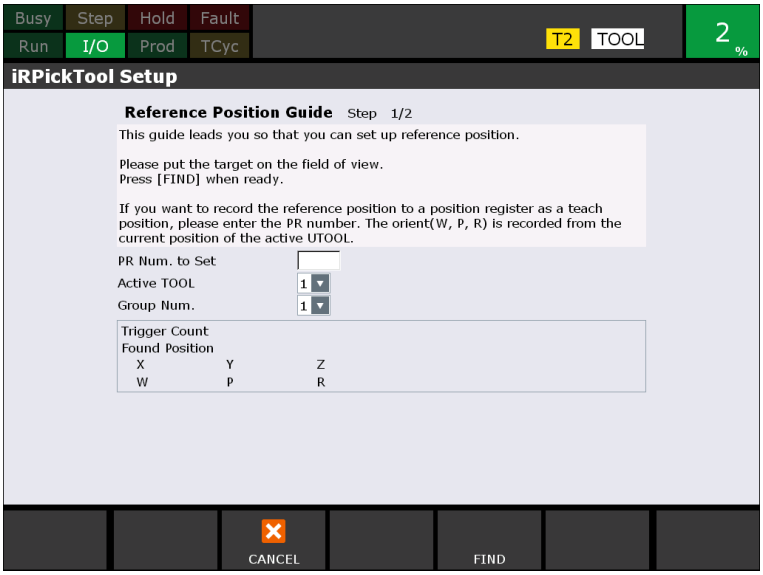
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SETUP] → [*i*RPickTool].

3 Select the sensor [SENS1], and for [Vision Process], select the name of the vision process (VT) that you created in Subsection 4.2.8.2, “Vision process” .

4 Press F3 [SET REF POS].



[Reference Position Guide Step: 1/2] will appear as below.



> **⚠ CAUTION**
> [Record found position on PR] cannot be used with Plug & Play. This is because with [Record found position on PR], it is assumed that an approach offset will be used as a fixed frame offset, but in the standard program, an approach offset is used as a tool offset.

5 Place a part around the center of the field of view of the camera.

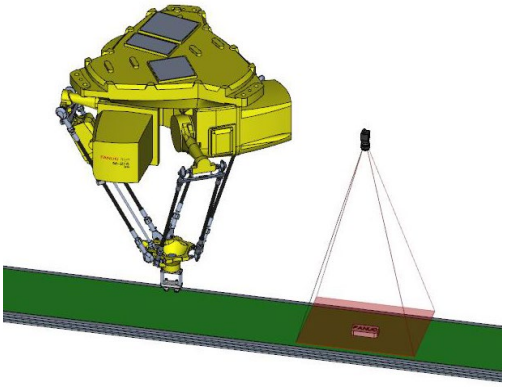
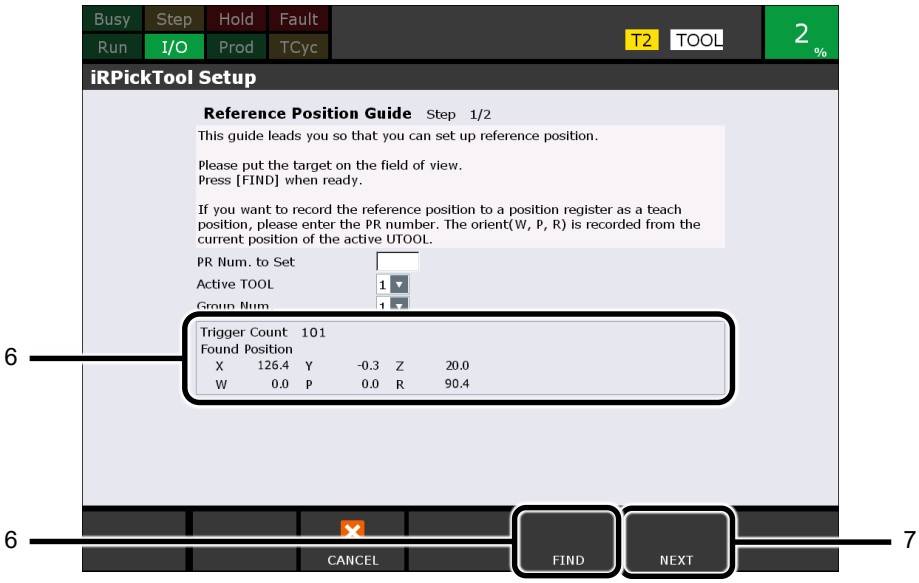


Fig. 4.2.12.1 (a) Placing a part inside the field of view of the camera

6 Press F4 [FIND].
If detection is successful, [Found Position] will appear.
* If detection is not successful
  - Open the vision process and perform a detection test.
  - When finding using a detection test, check to see if the selection of the vision process is correct on the sensor screen (step 3).

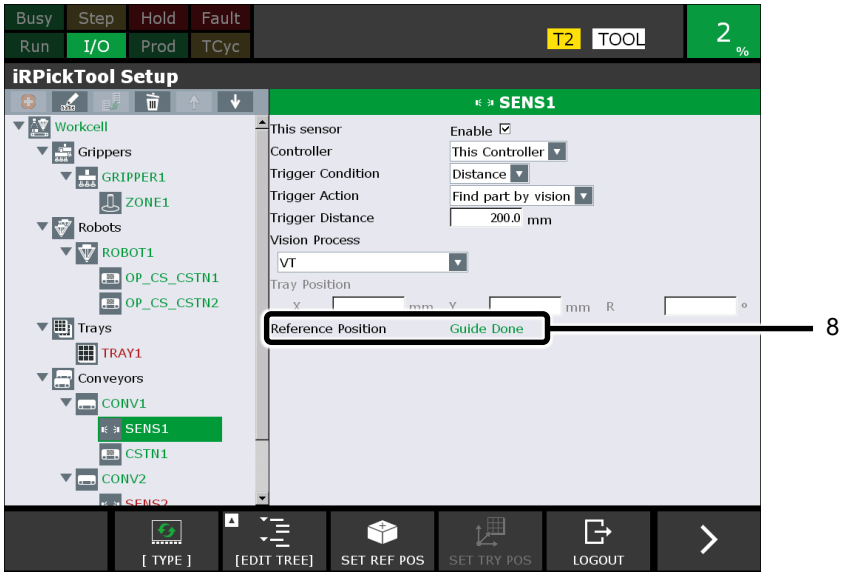
If detection is successful, F5 [NEXT] will become active.

7 Press F5 [NEXT].



8 Press F5 [FINISH].

A screen similar to the one below will appear.
Check that [Guide Done] is displayed for [Reference Position].



9 Move the conveyor. When the part reaches within the operating range of the robot, stop.
* Be careful that the part does not become misaligned.

10 Jog the robot so that it moves to the part pick position.

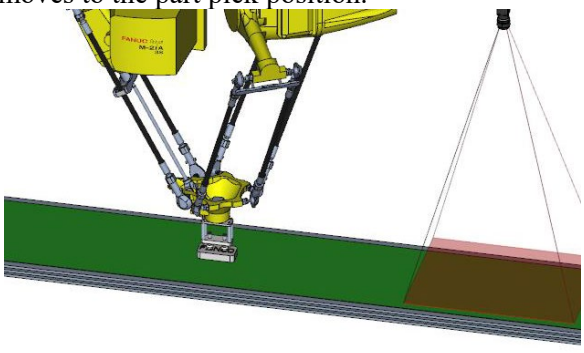


Fig. 4.2.12.1 (b) Moving the robot to the part pick position

11 Display the pick program PK_CV_PICK11, and move the cursor to the following line, where the pick position is taught.

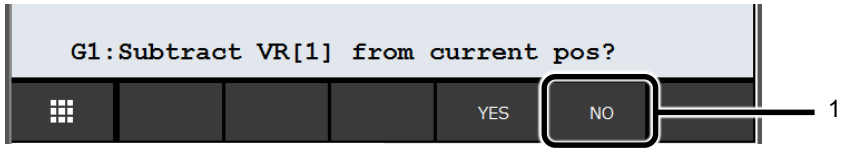
70: L PR[57: Pk1 Cv Ref Pos] max_speed CNT0 VOFFSET, VR[1] TB R[125]sec, CALL PKGRCLOSE ("Gripper ID"=R[104:Gripper Id G1],"Start Zone"=R[107:Counter G1],"End Zone"=R[106:Gr EndZone G1])

12 While holding down the [SHIFT] key, press F5 [TOUCHUP].
If F5 [TOUCHUP] is not displayed, press [> (NEXT)] to switch screens.

13 A message saying 'VR is UNINIT, continue?' is displayed, but press F4 [YES] and teach the position.

NOTE

1 If there is a value in the vision register when you are setting it again, 'Subtract VR from current pos?' will be displayed. Press F5 [NO].

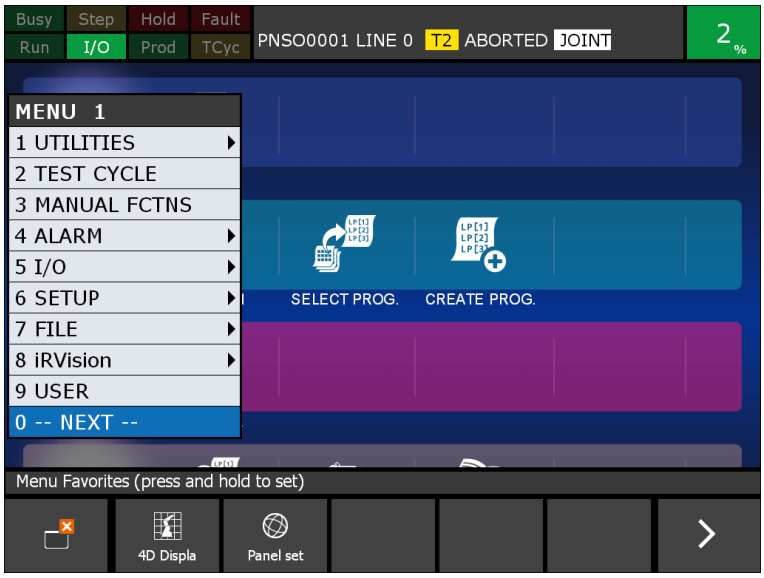


NOTE

2 If there is a value in the vision register when you are setting it again, then in order to avoid making a mistake, clear the vision register value by following the procedure below, and then teach the position. If you do this, the same position will be displayed whether you select F4 [YES] or F5 [NO] for 'Subtract VR from current pos?'

1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.

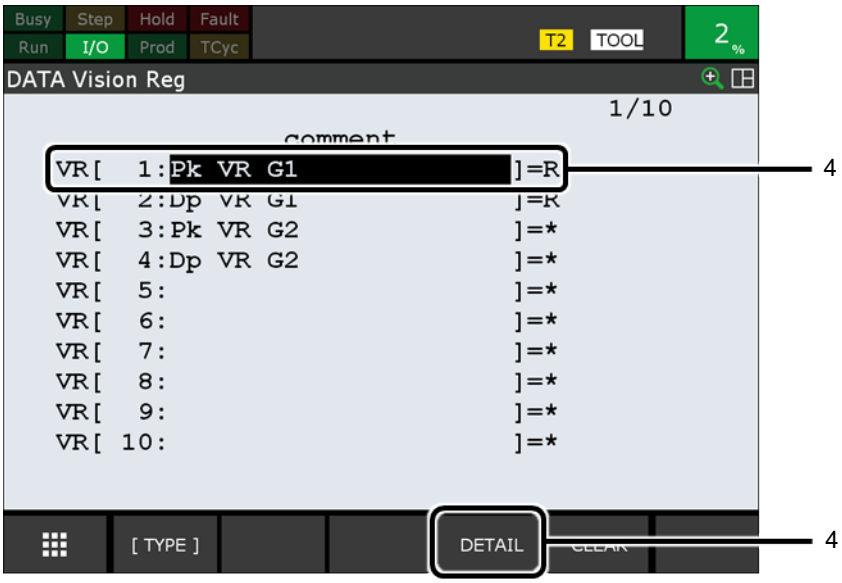
2 Select [NEXT] from the menu.



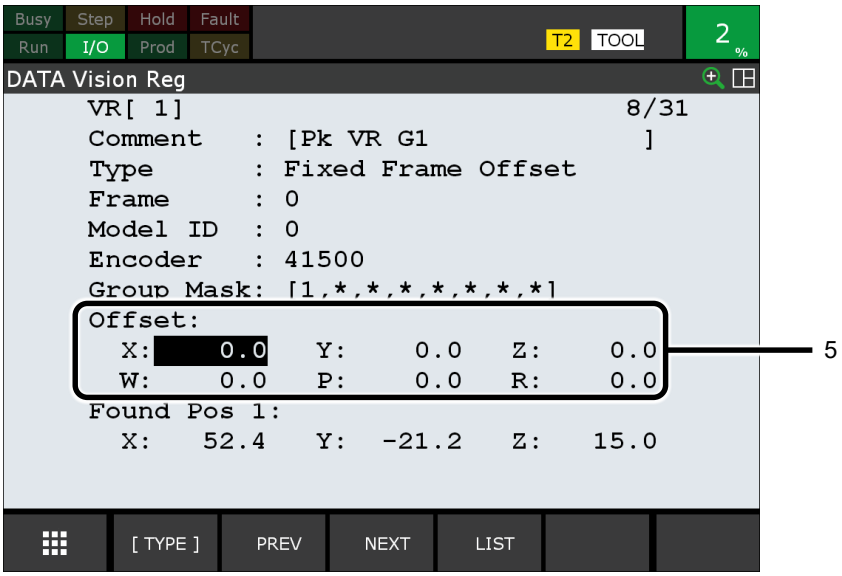
3 Select [DATA] → [Vision Reg].

NOTE

4 Place the cursor over '1' in [VR], then press F4 [DETAIL].



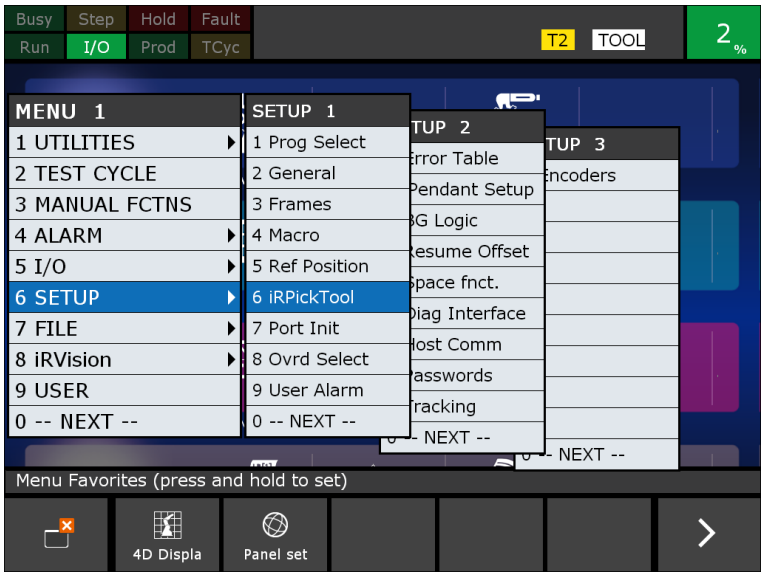
5 Enter '0' for all of the values [X], [Y], [Z], [W], [P] and [R] in [Offset].



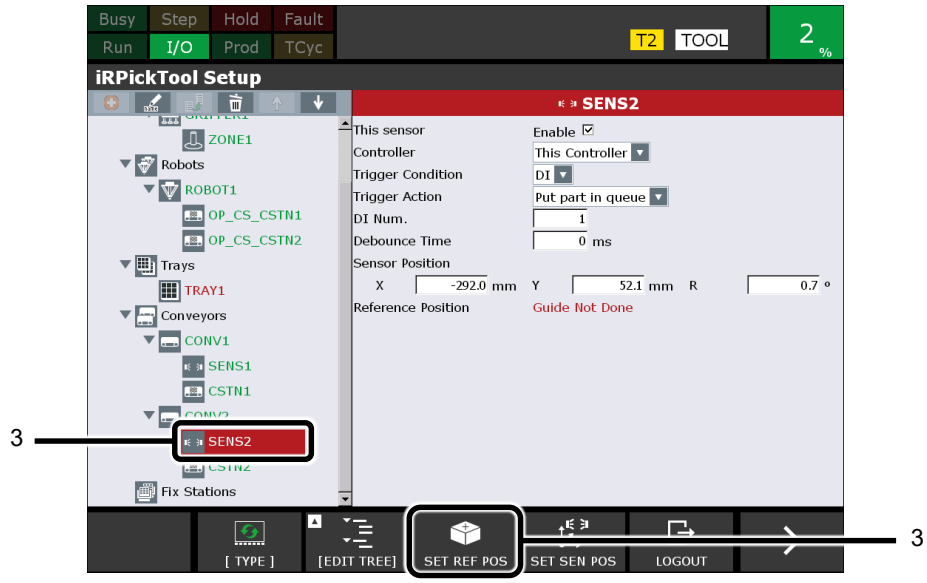
4.2.12.2 Setting up a reference position for the outfeed conveyor, and teaching a robot position

Set up a reference position for the outfeed conveyor, and teach a robot position. The procedure is for the case of [DI] / [RI] / [HDI] + [Put part in queue].

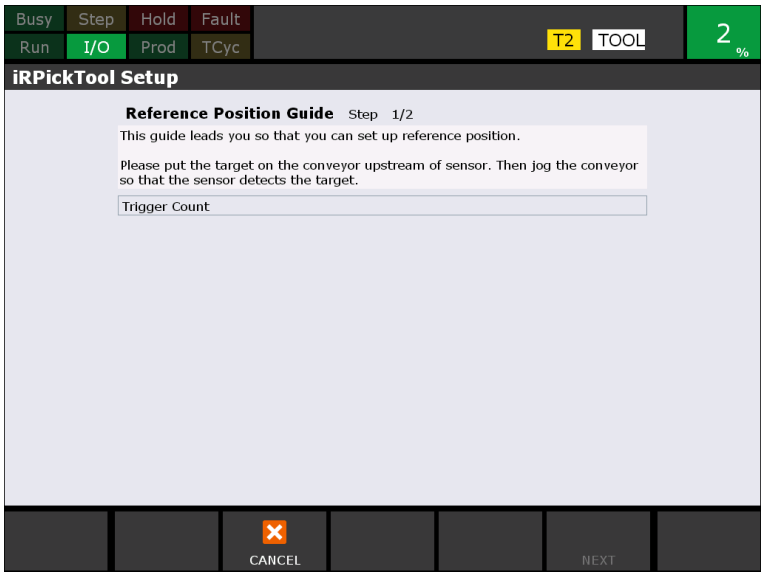
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SETUP] → [*i*RPickTool] from the menu.



3 Select the sensor [SENS2], then press F3 [SET REF POS].



A screen similar to the one below will appear.



4 Place a tray upstream of the trigger sensor.

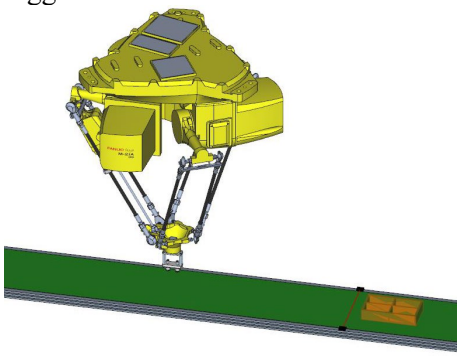
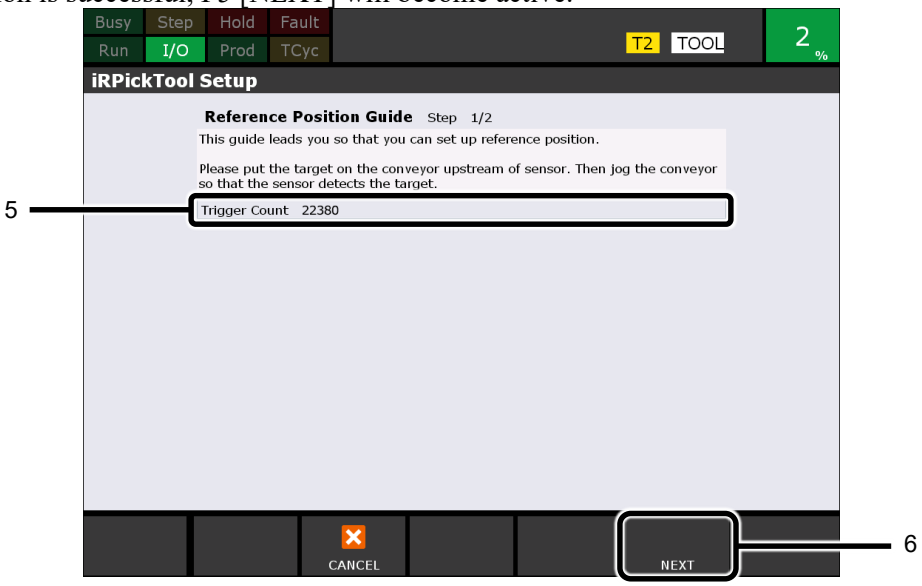


Fig. 4.2.12.2 (a) Placing a tray upstream of the trigger sensor.

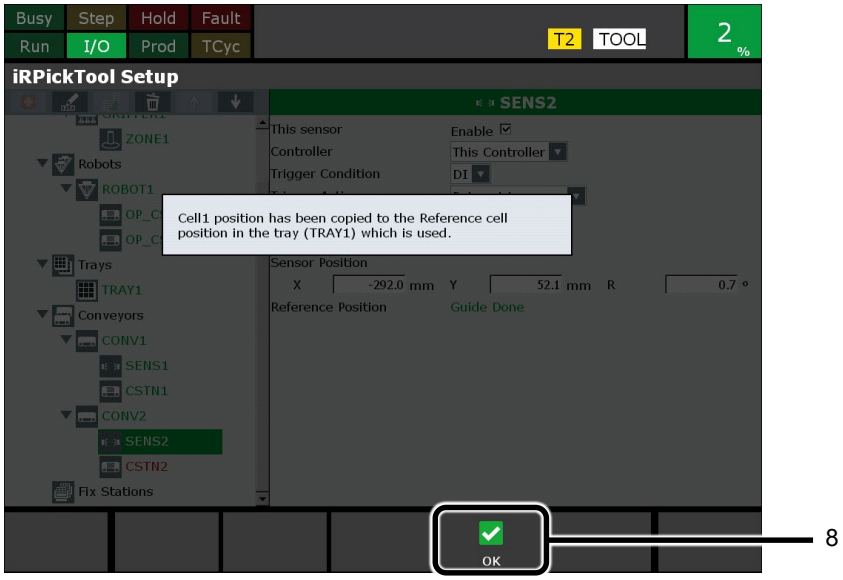
5 Move the conveyor. When the tray comes within the operating range of the robot, stop.
If detection is successful, [Trigger Count] will appear.

6 Press F5 [NEXT].
If detection is successful, F5 [NEXT] will become active.

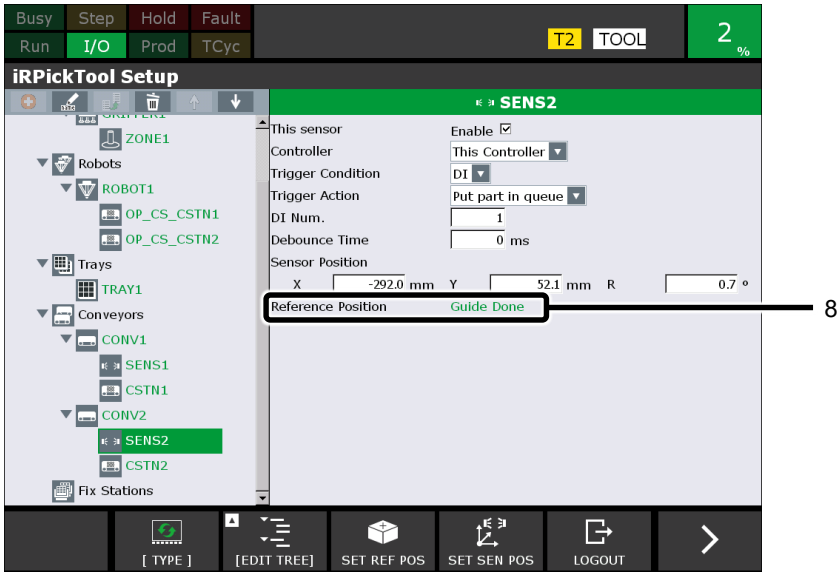


7 Press F5 [FINISH].

8 The following message will appear. Press F5 [OK].



A screen similar to the one below will appear.
Check that [Guide Done] is displayed for [Reference Position].



9 Move the conveyor. When the tray comes within the operating range of the robot, stop.
* Be careful that the tray does not become misaligned.

10 Jog the robot so that it moves to the drop position of Cell 1 on the tray.

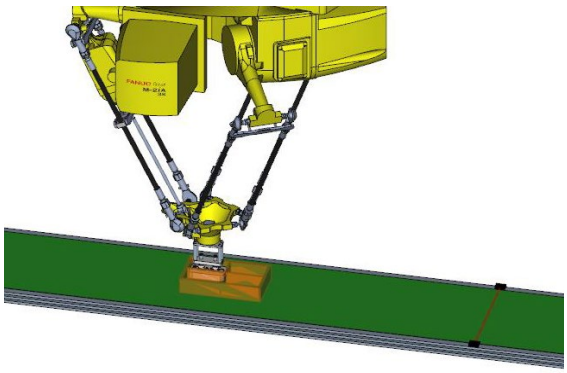
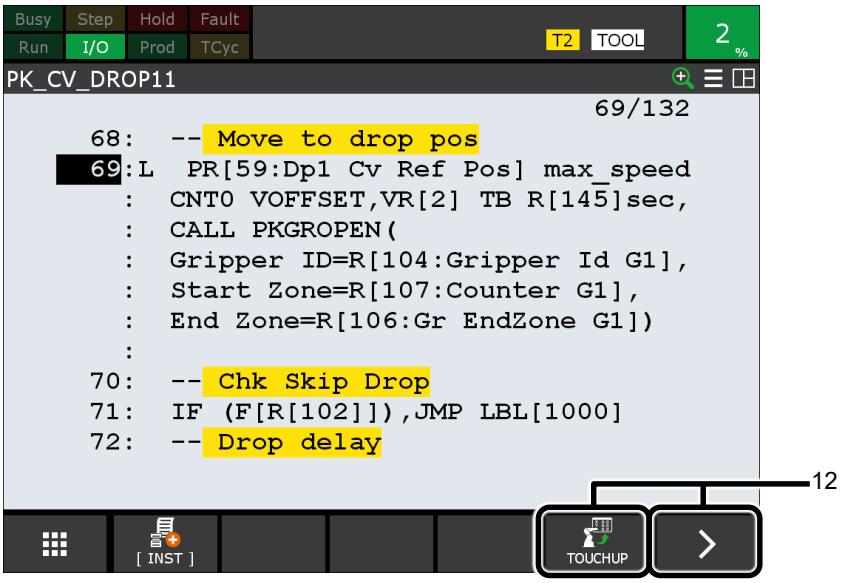


Fig. 4.2.12.2 (b) Moving the robot to the drop position of Cell 1 on the tray

11 Display the placement program, and move the cursor to the following line, where the drop position is taught.

69: L PR[59: Dp1 Cv Ref Pos] max_speed CNT0 VOFFSET, VR[2] TB R[145]sec, CALL PKGROPEN (“Gripper ID”=R[104:Gripper Id G1],“Start Zone”=R[107:Counter G1],“End Zone”=R[106:Gr EndZone G1]))

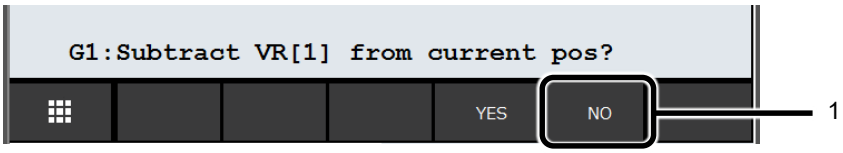
12 While holding down the [SHIFT] key, press F5 [TOUCHUP].
If F5 [TOUCHUP] is not displayed, press [> (NEXT)] to switch screens.



13 A message saying 'VR is UNINIT, continue?' is displayed, but press F4 [YES] and teach the position.

NOTE

1 If there is a value in the vision register when you are setting it again, 'Subtract VR from current pos?' will be displayed. Press F5 [NO].



NOTE

2 If there is a value in the vision register when you are setting it again, then in order to avoid making a mistake, clear the vision register value by following the procedure below, and then teach the position. If you do this, the same position will be displayed whether you select F4 [YES] or F5 [NO] for 'Subtract VR from current pos?'

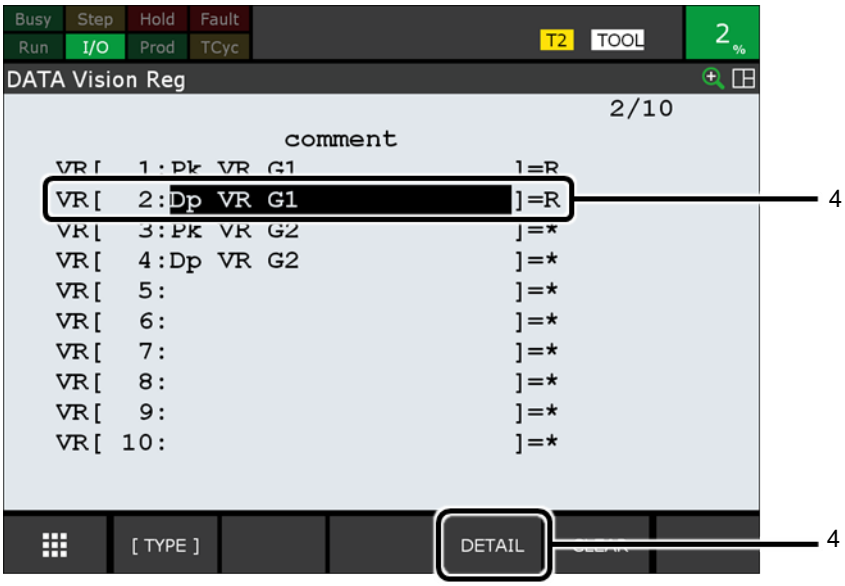
1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.

2 Select [NEXT] from the menu.

3 Select [DATA] → [Vision Reg].

NOTE

4 Place the cursor over '2' in [VR], then press F4 [DETAIL].

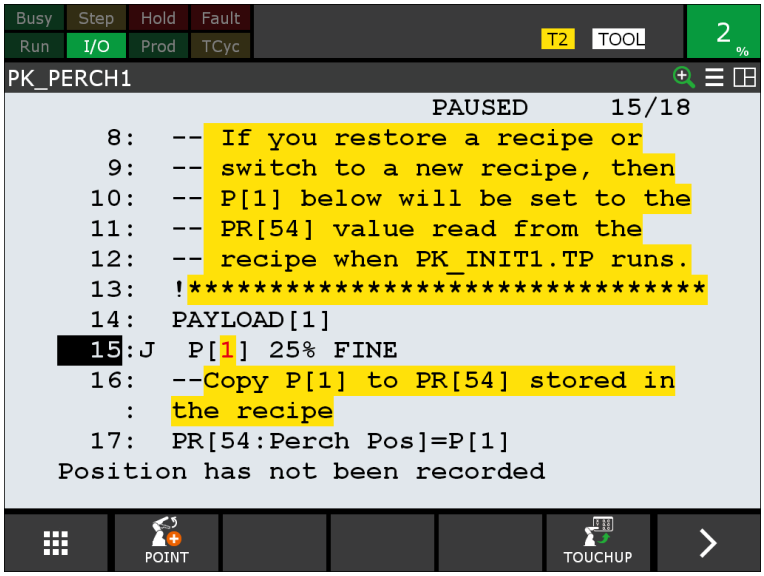


5 Enter '0' for all of the values [X], [Y], [Z], [W], [P] and [R] in [Offset].

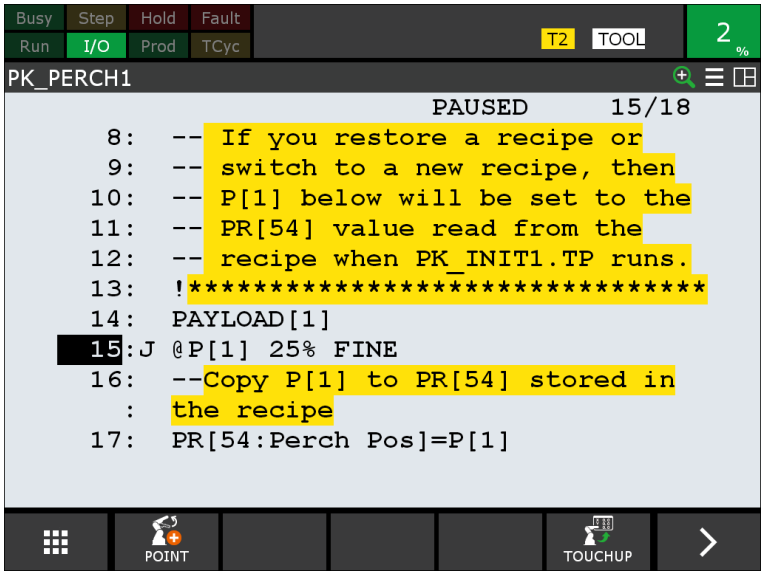
4.2.12.3 Teach the robot's home position

Teach the robot's home position.

1 Display the main program PK_MAIN1.
2 Set the override to something less than 5%.
3 Run PK_MAIN1 from the beginning using teaching mode (T1, T2).
It will stop with an alarm, because P[1] in line 15 of PK_PERCH1 is untaught.



4 Without changing the selected user frame or tool frame, jog the robot to the home position.
5 While holding down the [SHIFT] key, press F5 [TOUCHUP].
The current position will be taught to P[1].



4.2.13 Fine Adjustment of the Tracking Motion

In systems that require precision, fine adjustments are made to the tracking motion. Refer to Section 6.12, “FINE ADJUSTMENT OF THE TRACKING MOTION”.

5 OTHER PREPARATIONS BEFORE SETUP

This chapter explains other preparations not explained in the setup procedures in Chapters 3 and 4, such as settings and precautions for creating a robot ring and using *i*RPickTool.
It also explains the preparation for when creating a robot ring and using *i*RPickTool when there are multiple robots.

5.1 NETWORK CREATION

When the system contains multiple robot controllers, connect them via a network.

If you are setting up *i*RPickTool on a personal computer, even when there is only one robot controller in the system, network setup is necessary.

5.1.1 Connecting a Network Cable

In a tracking system using *i*RPickTool, a robot controller exchanges a large amount of information with other robot controllers. Any communication delay affects the performance of the system. For this reason, create a local network for making robot controllers communicate with one another, in which only the robot controllers are connected, so that other communication traffic does not affect communication among the robot controllers.

The R-30*i*B Plus / R-30iB Compact Plus / R-30iB Mini Plus controller has two Ethernet ports. Accordingly, one robot controller can belong to two different networks. If it is necessary to connect a robot controller to an external network such as a factory network, use port 1 to connect the robot controller to the factory network and port 2 to make the robot controller communicate with other controllers as shown in the figure below. In this case, connect the local network created using port 2 only to the robot controllers involved with visual tracking and a personal computer for setting up *i*RPickTool.

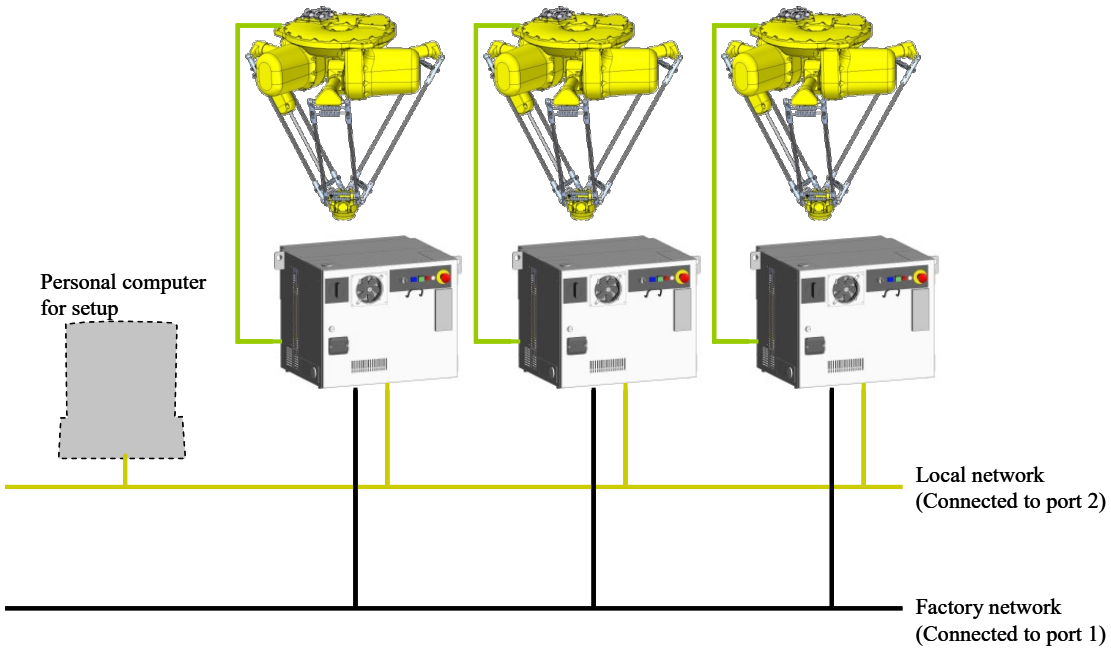


Fig. 5.1.1(a) Connection examples for two different networks

When port 1 is used, connect the Ethernet cable to the CD38A port on the MAIN board. When port 2 is used, connect the cable to the CD38B port.

NOTE
1 When a robot controller is not connected to any external network, you can use port 1 or 2, whichever you prefer.
2 Connect a personal computer for setting up *i*RPickTool to the local network for communication among robot controllers.

5.1.2 Caution on Setting up a Network

There are the following precautions on setting up a network. Observe these restrictions when performing each step below.

- As described in Subsection 5.1.1, "Connecting a Network Cable", one robot controller has two ports. Set the IP address and subnet mask of each port used according to the instructions described in Subsection 5.1.3, "Setting IP Addresses".
- If you want to use both ports 1 and 2, you should set different network addresses for ports 1 and 2. For details, see Subsection 5.1.3, "Setting IP Addresses".

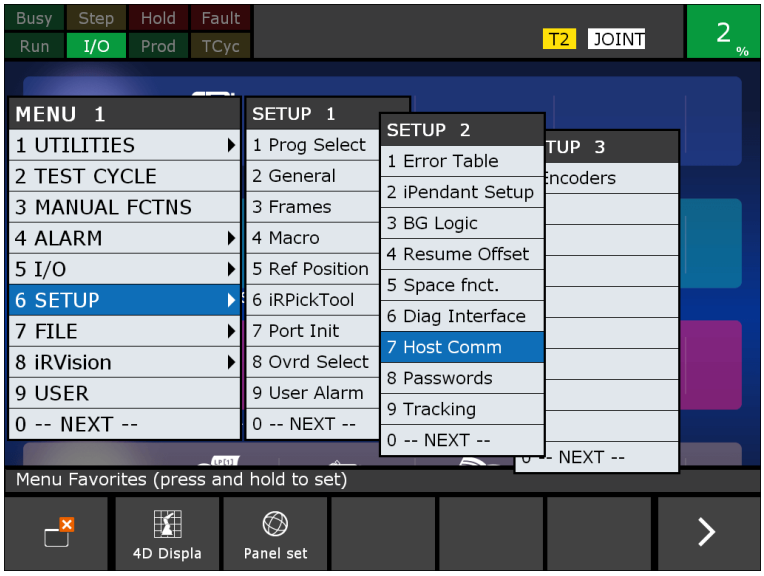
Observe the following precautions when the system contains multiple robot controllers. When the system contains only one robot, skip them.

- When setting IP addresses, assign a unique robot name to each robot controller.
- When setting IP addresses, use consecutive numbers for the IP addresses of the port used for communication among robot controllers. For example, when the system contains four robot controllers and port 2 is used, the IP address of port 2 of each robot controller is set sequentially to 172.16.0.1, 172.16.0.2, 172.16.0.3, or 172.16.0.4.
- When the Ethernet Encoder is used, the IP address of the robot controller connected to a Pulsecoder must be the starting number. In the above example, set the IP address for port 2 of the robot controller connected to a Pulsecoder to 172.16.0.1.

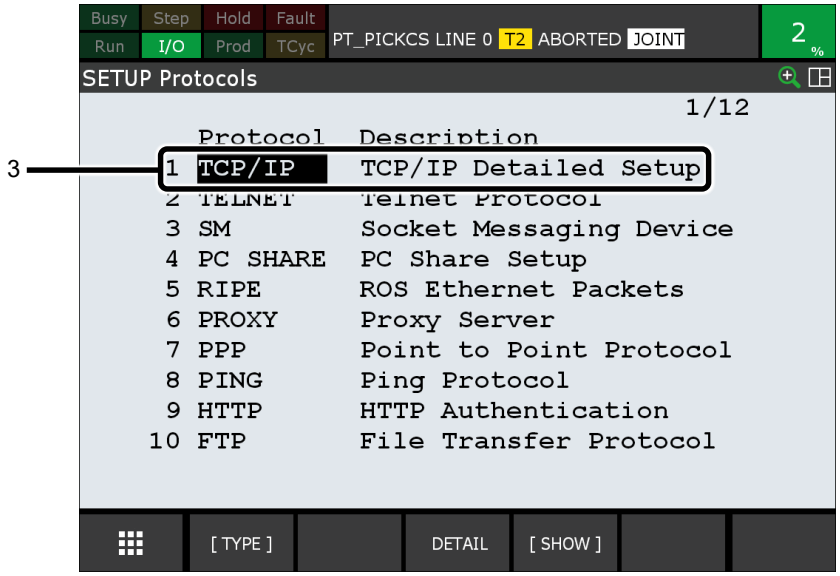
5.1.3 Setting IP Addresses

Follow the procedure below to set IP addresses for each robot controller:

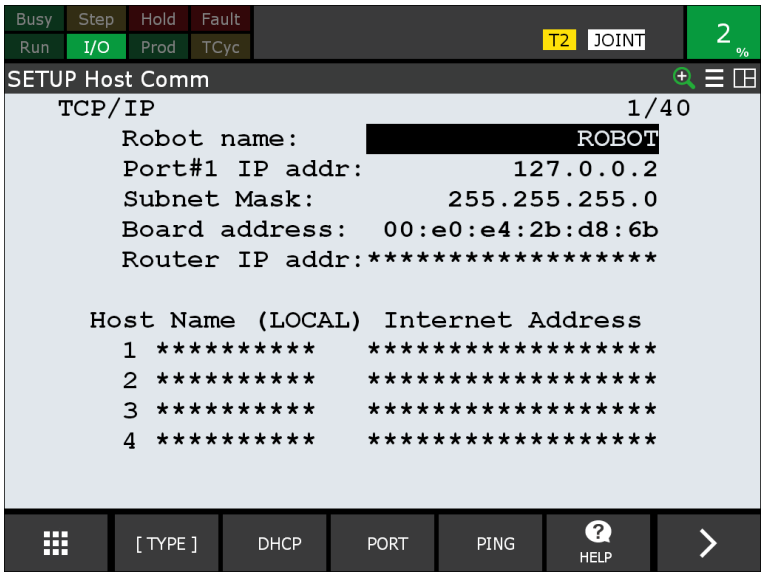
1 Press [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Choose [SETUP] → [Host Comm] from the menu.



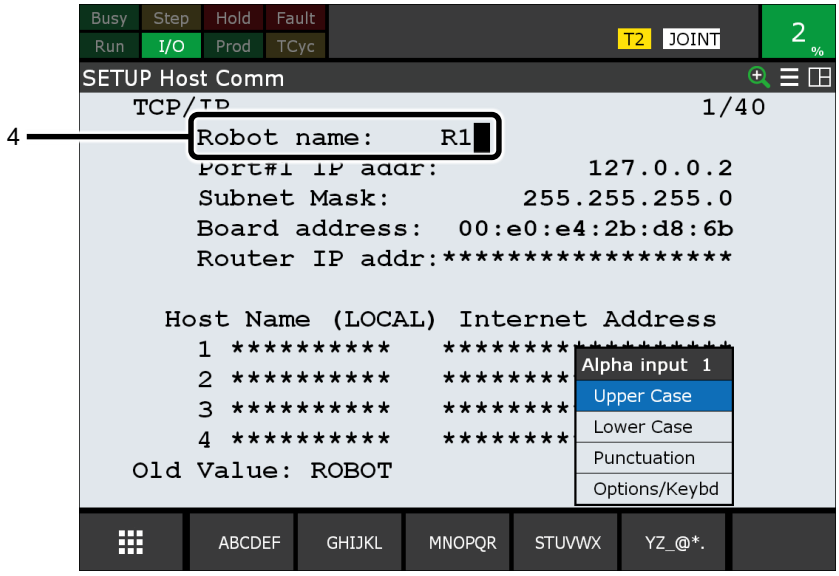
The [SETUP Protocols] screen will appear.
3 Move the cursor to [TCP/IP] and press the [ENTER] key.



The [SETUP Host Comm] screen will appear.



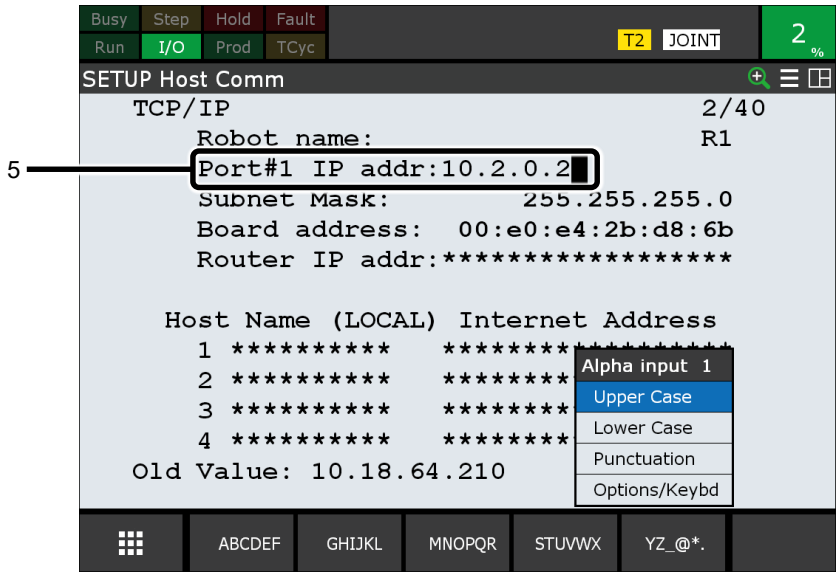
4 Enter the name of this robot controller in [Robot name]



⚠ CAUTION

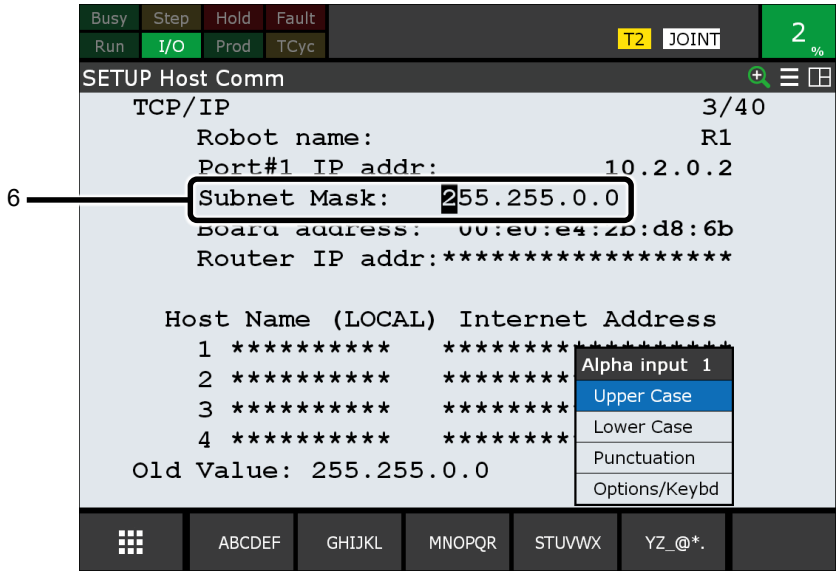
A robot name can contain only alphanumeric characters and minus sign. The first character must be an alphabetic character. The last character must not be a minus sign. The robot name cannot contain any spaces.

5 Enter the IP address of the robot controller in [Port#1 IP addr].

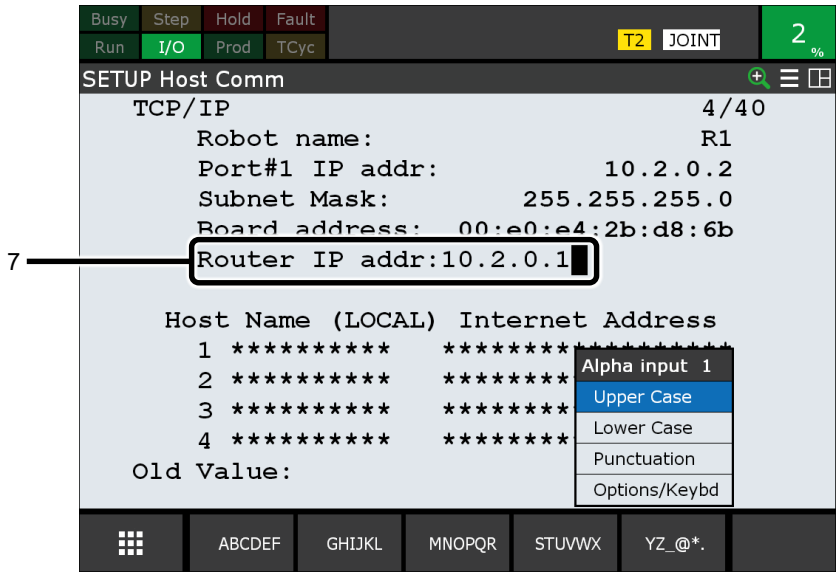


> **⚠ CAUTION**
> The IP address must not contain any space or unnecessary “0”. If there is any space or unnecessary “0” in the IP address, correct communication is disabled.

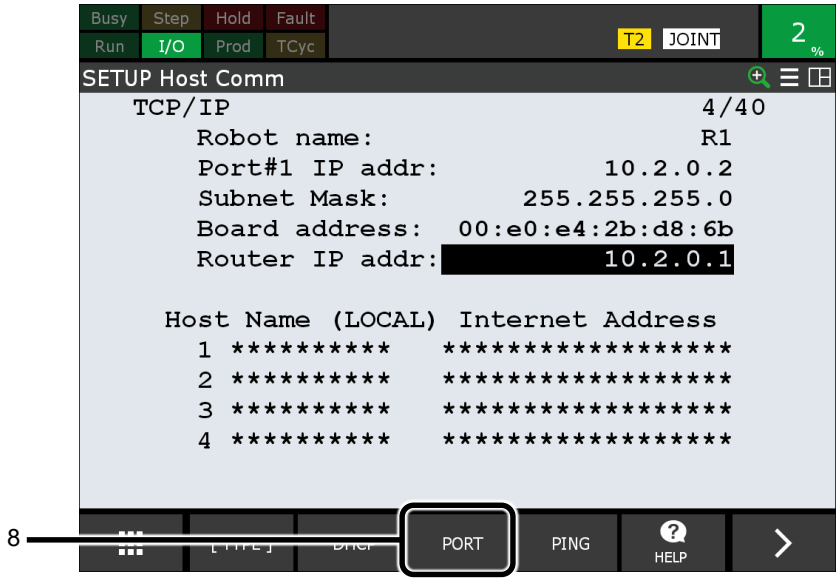
6 Enter a subnet mask in [Subnet Mask].



7 Enter the IP address of the default gateway in [Router IP addr].

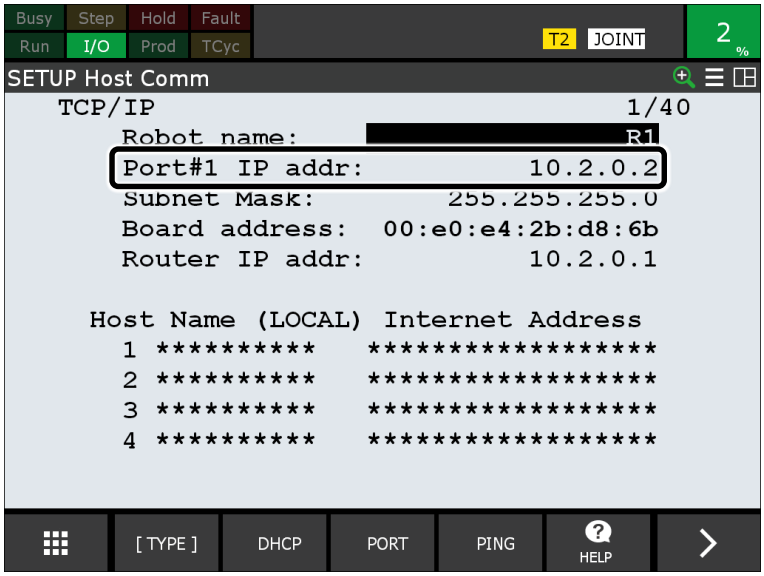


8 Press F3 [PORT] key and select a port.

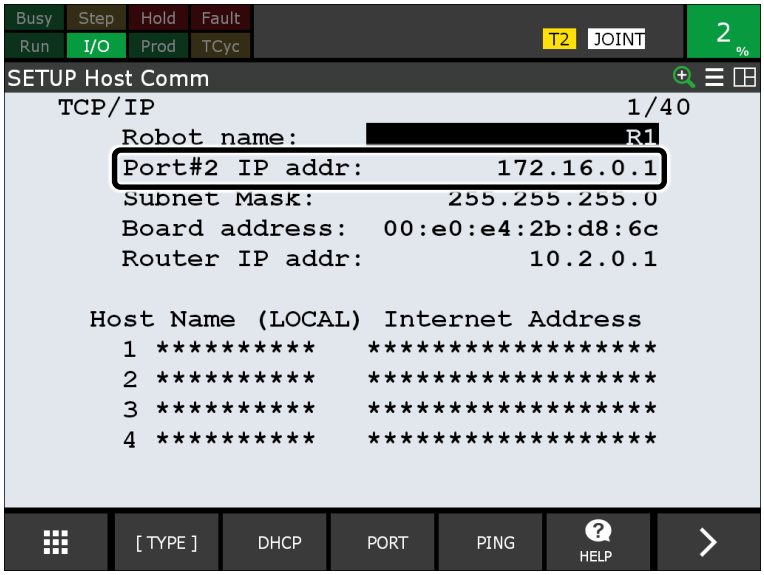


9 Set the IP address and subnet mask for port 2 in the same way as in steps 6 and 7.
10 Turn the power to the robot controller off, then on again.

> ⚠ **CAUTION**
> If you do not turn the power to the robot controller off, then on again, the following setup is not performed correctly. Be sure to turn the power to the robot controller off, then on again.



Example of setting port 1



Example of setting port 2

> **⚠ CAUTION**
> To use both ports 1 and 2, set different network addresses for ports 1 and 2. For example, when the subnet mask is 255.255.0.0 and the IP address is 172.16.0.1, the network address is 172.16 and the host address is 0.1. If IP addresses containing the same network address are set for ports 1 and 2, the alarm “HOST-179 IP Address mis-configuration” is issued at power-on, and only port 1 is enabled and port 2 does not function.

5.1.4 Setting up the Robot Ring

When the system contains multiple robot controllers, set the robot ring. When the system contains only one robot, it is unnecessary to set the robot ring. Skip this subsection.

For communication among robot controllers, a communication function called "ROS interface packets over Ethernet (RIPE)" is used internally. For details of RIPE, refer to the chapter, "ROS INTERFACE PACKETS OVER Ethernet (RIPE)" in the "Ethernet Function Operator's Manual B-82974EN".

Master and Slave

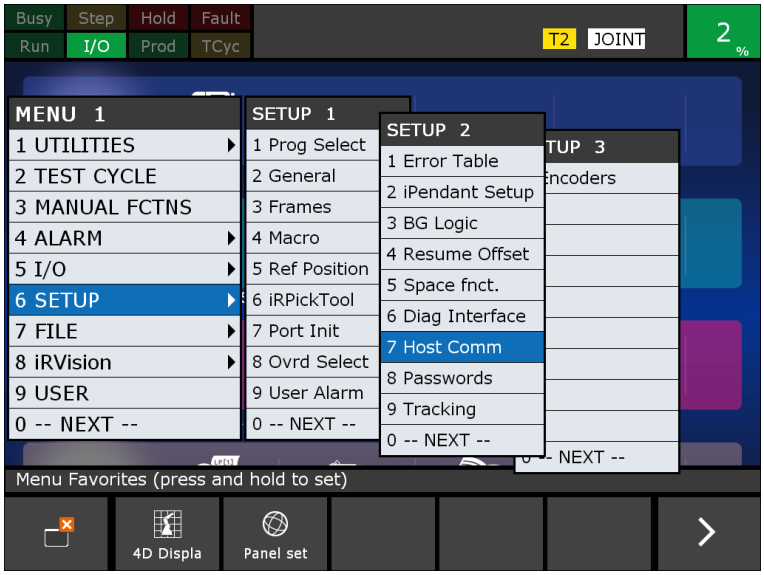
A robot ring consists of one master controller and other slave controllers. With *i*RPickTool, any robot controller can be the RIPE master. When the Ethernet Encoder is used, however, it is recommended that the robot controller connected to a Pulsecoder be used as the RIPE master controller.

Set up a robot ring, first on the slave controllers. After setting up the robot ring on all slave controllers, set it up on the master controller. When the robot ring has been set up on the master controller, all controllers automatically restart and the setup is completed. The following describes the procedure in detail.

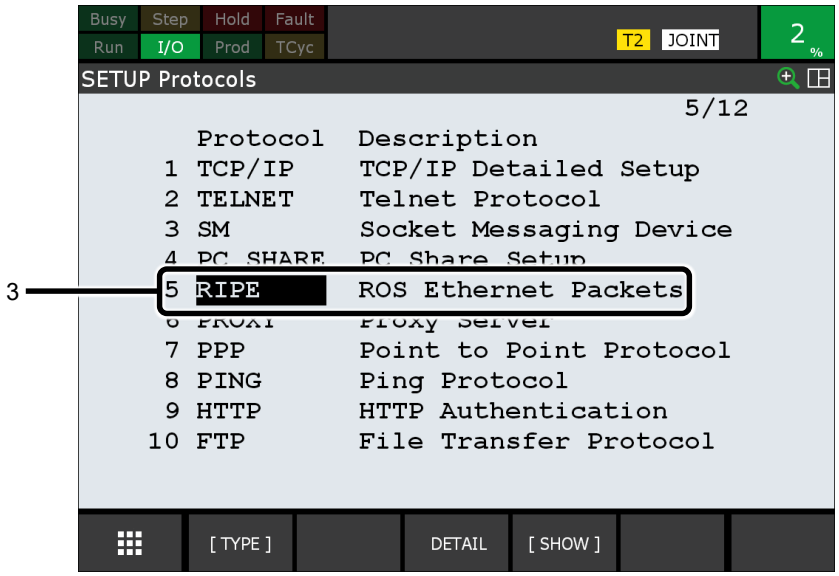
Setup on a Slave

Follow the procedure below to set up a robot ring on a slave controller:

1 Press [MENU] key on the teach pendant of the robot controller.
2 Choose [SETUP] → [Host Comm] from the menu.

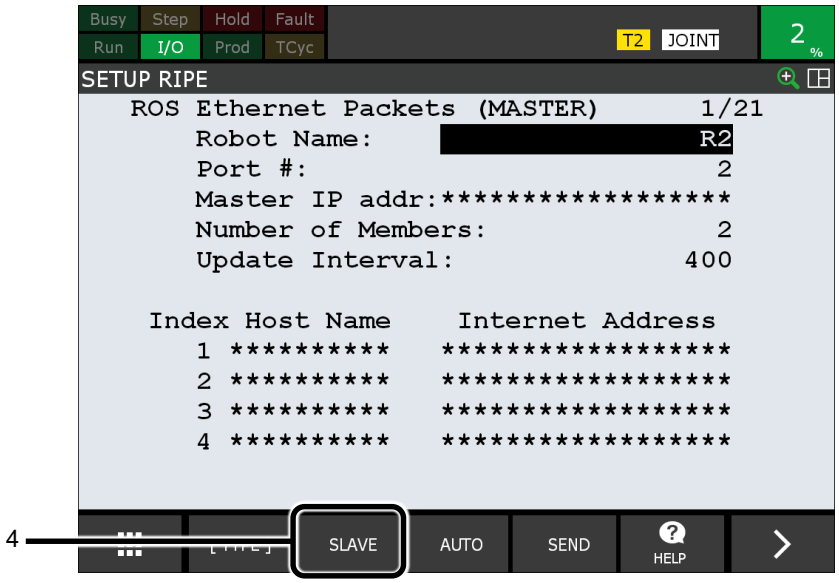


3 Position the cursor to [RIPE] and press [ENTER] key.

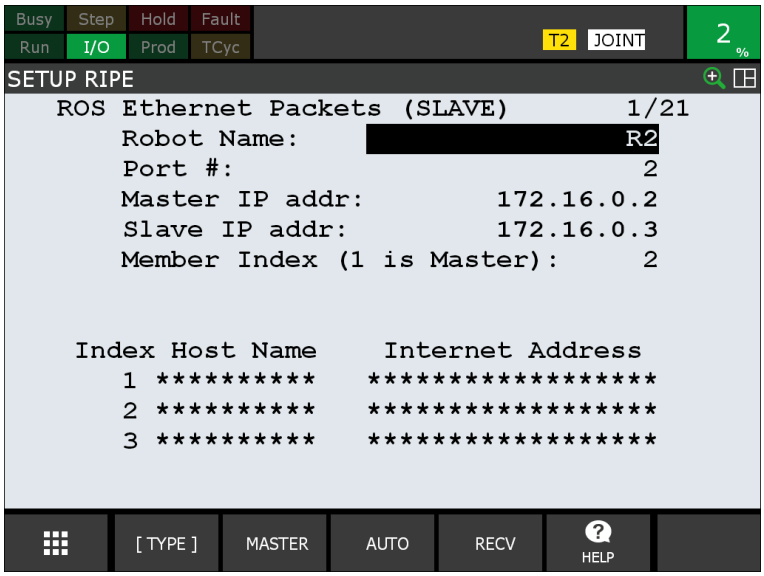


The following screen appears.

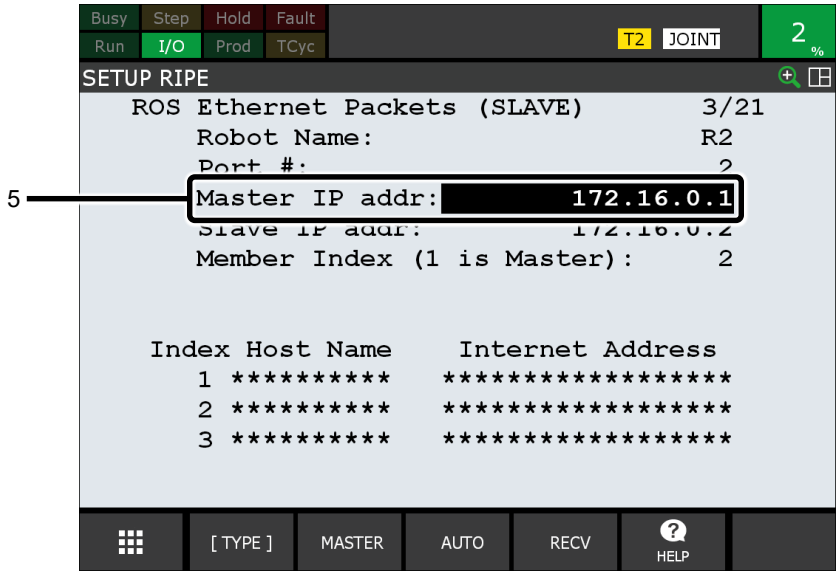
4 Press F2 [SLAVE].
This step is unnecessary when the screen for slave screens is already open and F2 indicates [MASTER].



The screen switches to a screen like the following ROS Ethernet Packets (SLAVE) screen.

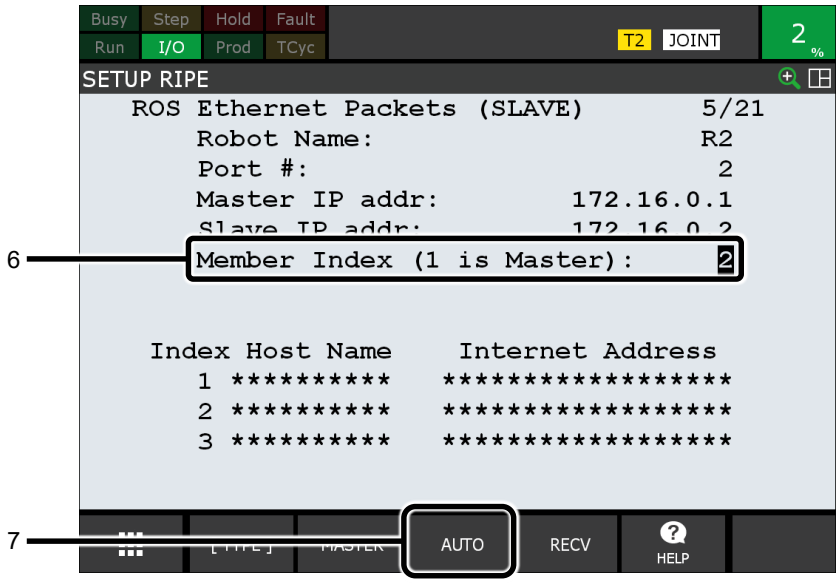


5 In [Master IP addr], set the IP address of the master controller for the port used for the robot ring.



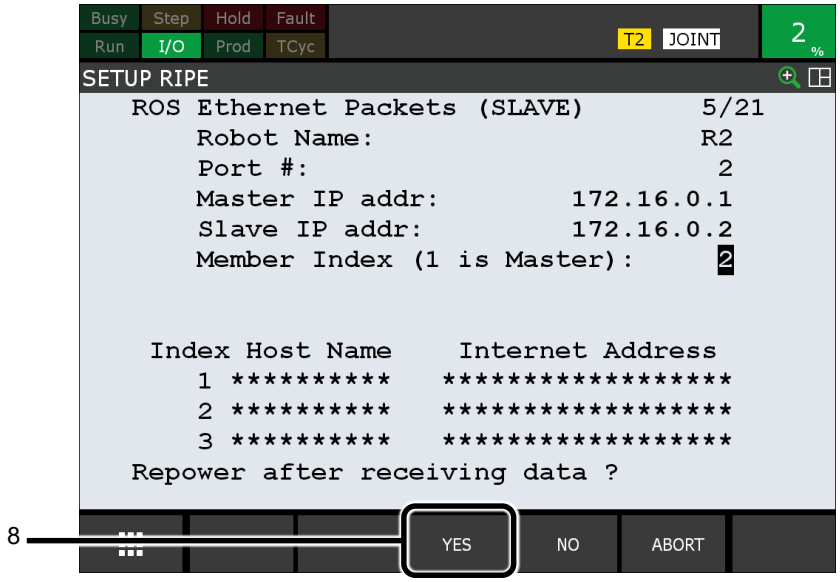
6 Set [Member Index]. You have already set consecutive numbers for the IP address for the port used for the robot ring for each robot controller as described in Subsection 5.1.3, “Setting IP Addresses”. The starting IP address should be set for the master controller. Enter the order of the IP address for the robot controller from the top. According to this order setting, [Slave IP addr] is automatically set. Check that the displayed IP address is the same as that set for this controller as described in Subsection 5.1.3, “Setting IP Addresses”.

7 After setting the above items, press F3 [AUTO] key.

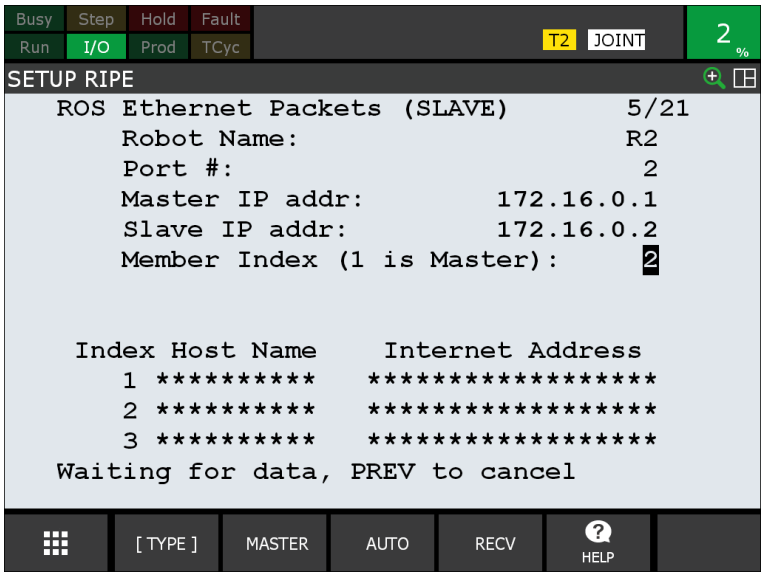


The following screen appears.

8 A message appears, which asks you whether to restart the controller after data reception. Press F3 [YES] key.



9 Perform the above steps on all slave controllers. Place all slave controllers in the status in which they are waiting for data from the master controller as shown in the screen below.

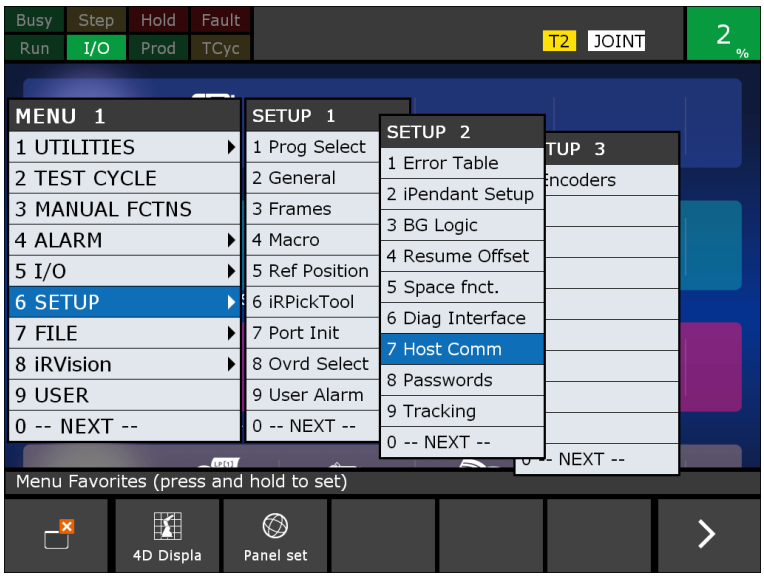


Setup on the Master

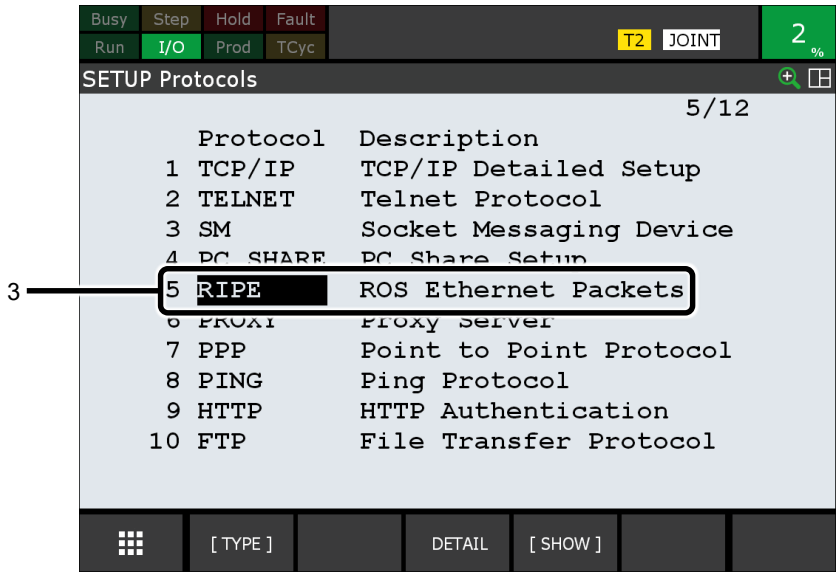
After you finish setup on all slave controllers, follow the procedure below to set up the robot ring on the master controller:

1 Press [MENU] key on the teach pendant of the robot controller.

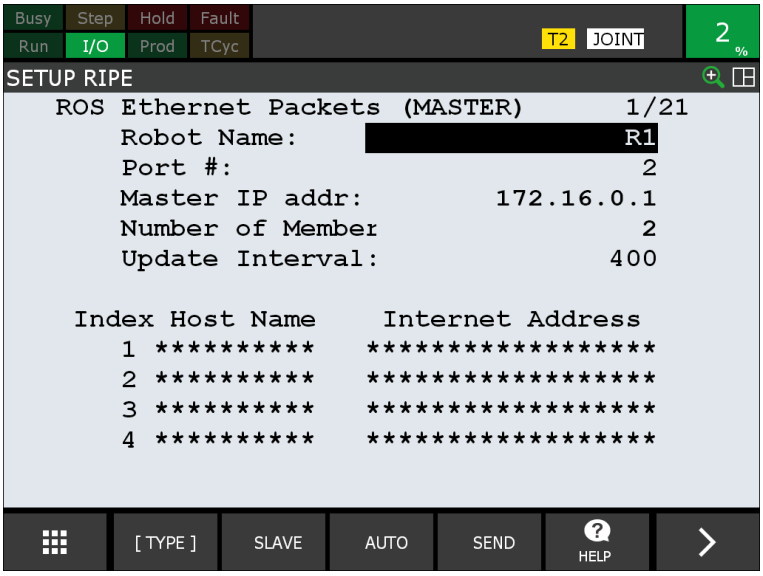
2 Choose [SETUP] → [Host Comm] from the menu.



3 Position the cursor to [RIPE] and press [ENTER] key.

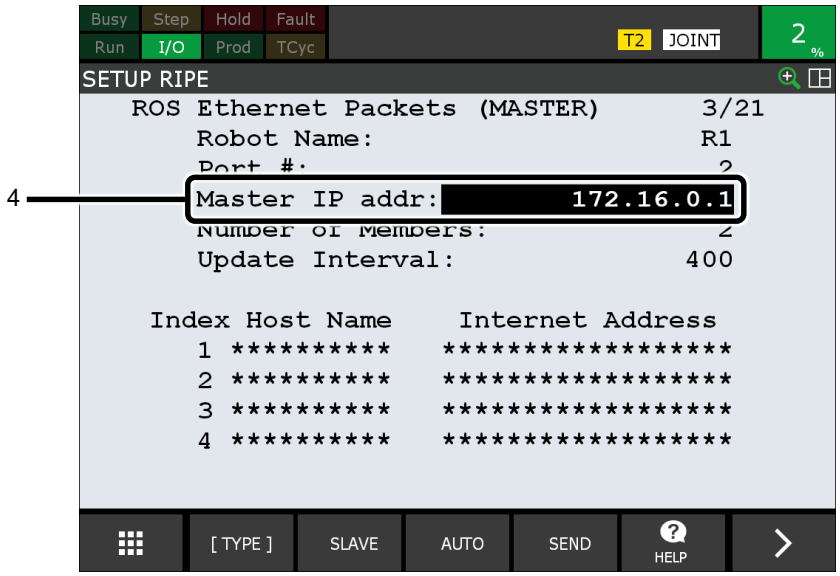


A screen like the following ROS Ethernet Packets (MASTER) screen will appear.

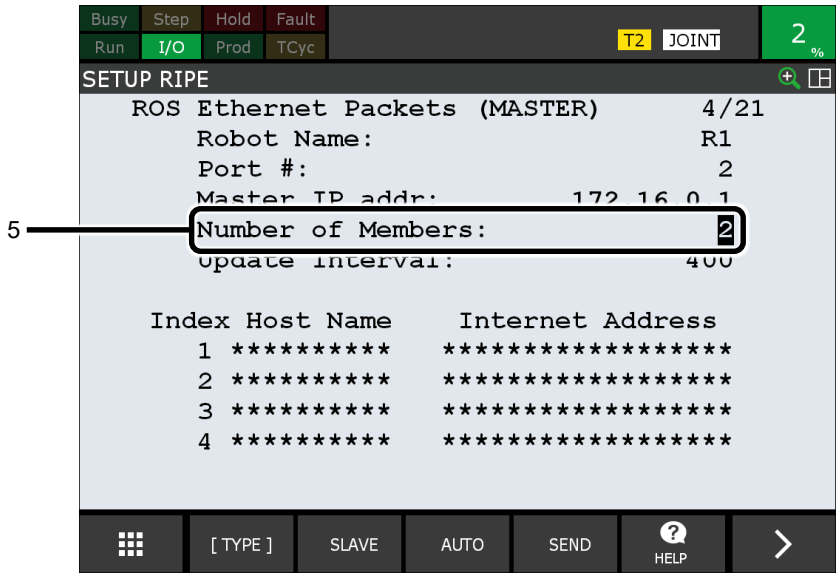


NOTE
If the ROS Ethernet Packets (SLAVE) screen has opened, press F2 [MASTER] to switch to the ROS Ethernet Packets (MASTER) screen.

4 In [Master IP addr], set the IP address of the master controller for the port used for the robot ring. The IP address is that of this robot controller.

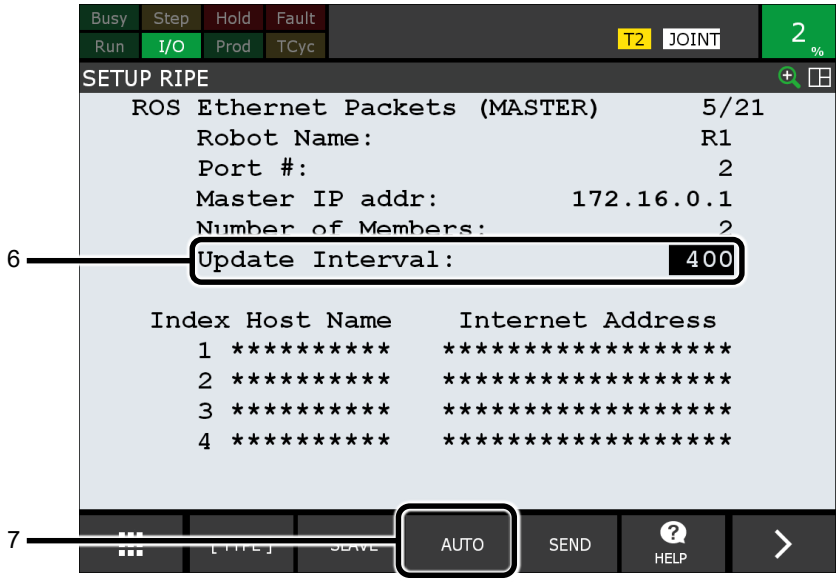


5 In [Number of Members], set the number of controllers which join the robot ring.

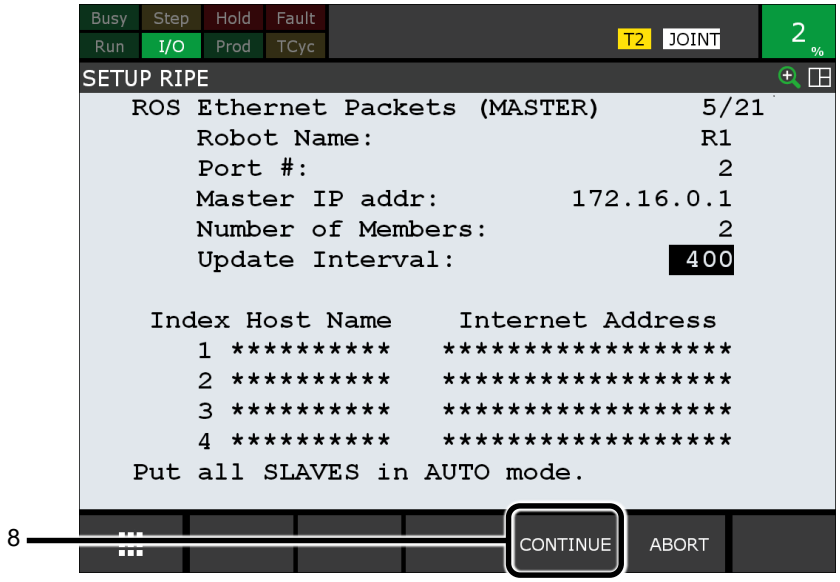


6 Set [Update Interval] if necessary. At intervals of this setting, each robot is checked to see whether it is online.

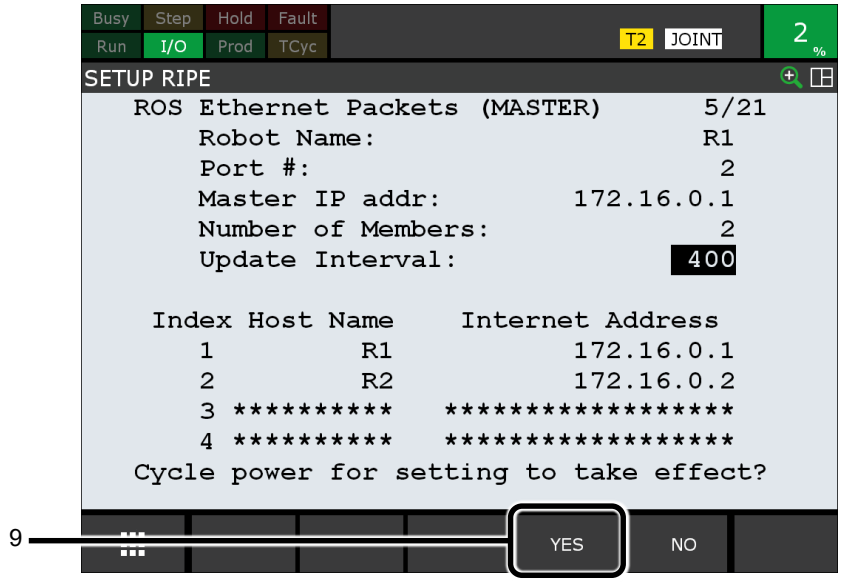
7 After setting the above items, press F3 [AUTO] key. The following screen appears.



8 The message “Put all SLAVES in AUTO mode.” appears. All slave controllers are waiting for data from the master controller as set, described above, in the procedure for setting up the robot ring on slave controllers. Press F4 [CONTINUE] key. The following screen appears.



9 Under [Host Name], the robot names and IP addresses of all controllers that join the robot ring are listed. A message appears, which asks you to whether to restart the controllers to reflect the setting. Press F4 [YES] key.



After that, all controllers restart and the setup is completed.

> **⚠ CAUTION**
> Do not change any robot or host name after you finish the setup of the robot ring. The robot and host names must be kept unchanged. If you want to change any robot or host name, perform the steps for setting up a robot ring from the beginning again.

5.2 PULSECODER INSTALLATION, CONNECTION AND SETUP

Install a Pulsecoder and connect it to a robot controller.
This section explains the setup for when installing a Pulsecoder and connecting to main boards, and when using with multiple robots.
For connection and setup of Pulsecoders and robot controllers, refer to Chapter 3, “*i*RPICKTOOL BASIC (BASIC FUNCTIONS) STARTUP PROCEDURES”, and Chapter 4, “*i*RPICKTOOL (BASIC FUNCTIONS + PLUG & PLAY) STARTUP PROCEDURES”.

> **NOTE**
> In this manual, a device for detecting the conveyor travel distance is called a “Pulsecoder”. On the teach pendant and *i*RPickTool setup screen, the term “encoder” is used as a synonym of “Pulsecoder”.

With *i*RPickTool, the αA1000S Pulsecoder and incremental Pulsecoders are available.
For the configuration required for using each Pulsecoder with the R-30*i*B Plus / R-30iB Mate Plus / R-30*i*B Compact Plus / R-30*i*B Mini Plus controllers, see the table below.

Table 5.2(a) Configuration required for each robot controller

Pulsecoder	R-30*i*B Plus Controller	R-30*i*B Mate Plus Controller	R-30*i*B Compact Plus / R-30*i*B Mini Plus Controller
αA1000S Pulsecoder (A860-0372-T001)	Table D.1 (a) in Appendix D, "αA1000S PULSECODER"	Table D.1 (b) in Appendix D, "αA1000S PULSECODER"	Table D.1 (c) in Appendix D, "αA1000S PULSECODER"
Incremental Pulsecoder (A860-0301-T001 to T004)	Table E.1 (a) in Appendix E, "INCREMENTAL PULSECODER"	Table E.1 (b) in Appendix E, "INCREMENTAL PULSECODER"	Not Available

5.2.1 Installing a Pulsecoder

Install a Pulsecoder on a conveyor.

Pulsecoder

Install the Pulsecoder at a location outside the robot operation area so that the Pulsecoder does not interfere with the robot during operation. When a camera is used to detect parts, install the Pulsecoder outside the camera's field of view.

Attaching a rotary disk to the tip of the Pulsecoder so that the rotary disk rotates by directly touching the conveyor belt enables the conveyor travel distance to be measured more accurately than attaching the Pulsecoder directly to the drive or follower axis of the conveyor.

Ensure that no slippage occurs between the rotary disk and conveyor belt. If a slippage occurs, the conveyor travel distance cannot be measured accurately, resulting in degraded robot handling precision. Pressing the Pulsecoder against the conveyor using a spring is effective in securing the Pulsecoder.

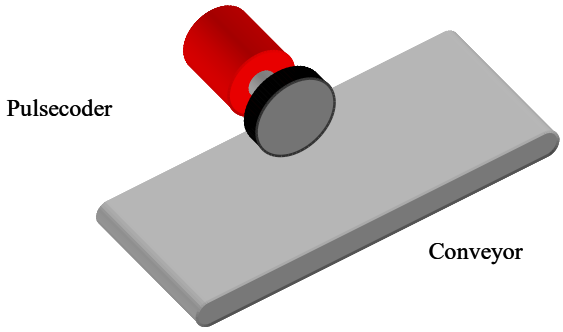


Fig. 5.2.1(a) Installation location for Pulsecoder

5.2.2 Connecting the Pulsecoder

Connect the Pulsecoder to a robot controller.

> ⚠ **CAUTION**
> If you install the Pulsecoder cable near the power supply cable, a robot may be not able to read the Pulsecoder signal correctly due to the noise of the power supply cable. Please keep the Pulsecoder cable away from the power suply cable.

For the connection method for Pulsecoders, refer to "Appendix D αA1000S PULSECODER" or "Appendix E INCREMENTAL PULSECODER".

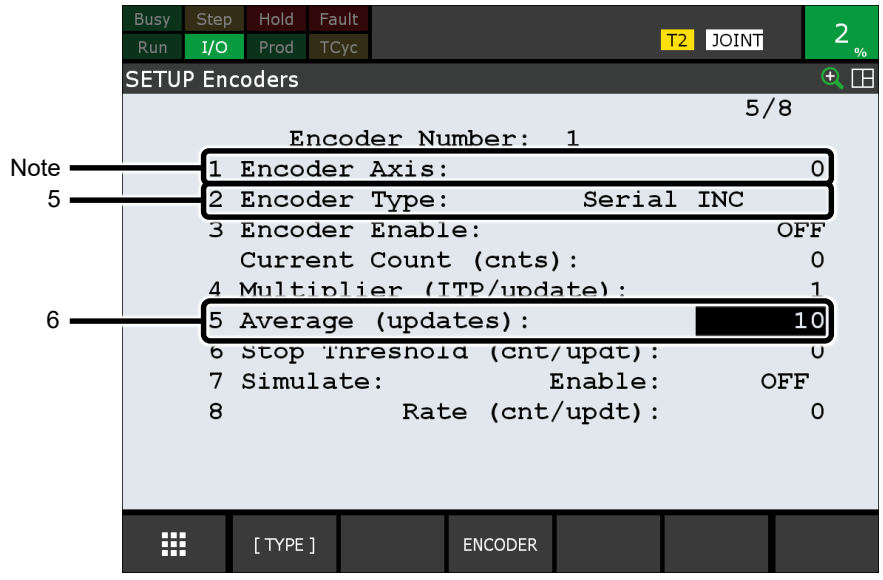
5.2.3 Setting Up Pulsecoders That Are Connected to the Main Board

This section explains the setup for when using an αA1000S Pulsecoder by connecting it to the main board.

Set the following items.
- [Encoder Type]
- [Average (updates)]

1 Press the [MENU] key on the teach pendant of the robot controller.
A menu will appear.
2 Select [SETUP] → [Encoders] from the menu.
3 Press F3 [ENCODER] and the [Encoder Number]. If no other Pulsecoders are connected to the interface board, select Encoder Number 1, and if other Pulsecoders are connected to the interface board, select Encoder Number 3. When connecting two αA1000S Pulsecoders to Main CPU board (R-30iB Compact Plus / R-30iB Mini Plus), select Encoder Number 1 if the Pulsecoder is connected to **PULSECODER1**, and select Encoder Number 2 if the Pulsecoder is connected to **PULSECODER2**.
4 Move the cursor to [Encoder Type] and press F4 [CHOICE].
5 Select [Main Serial INC].
6 Move the cursor to [Average (updates)] and input the cumulative accumulated number of instantaneous speeds (width of moving average).
By setting the accumulated number of instantaneous speeds, the robot will stop smoothly and not suddenly, even if the conveyor suddenly stops while the robot is tracking.
Usually, enter '10.'

NOTE
If connecting to the main board, leave [Encoder Axis] as 0, the initial value.



7 Cold start the robot controller.

5.2.4 Set Up Pulsecoders for Multiple Robots

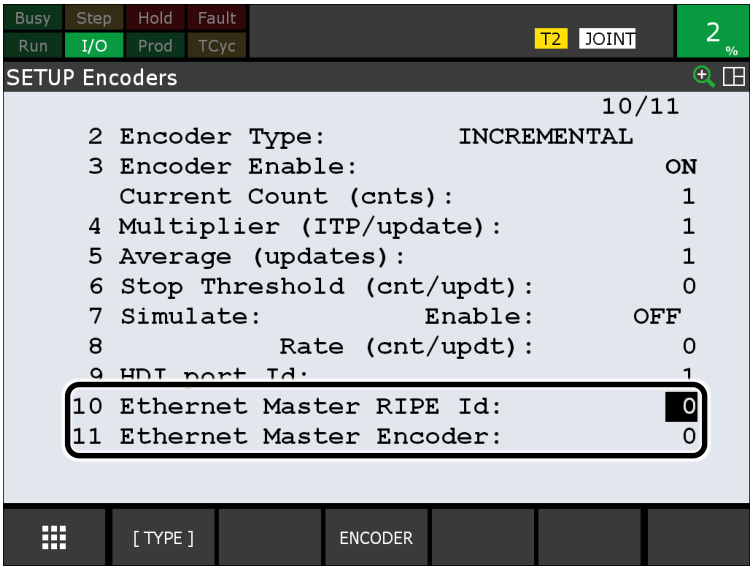
This section explains the setup for when using ethernet encoders with multiple robots.
If there is one robot or if you use a pulse multiplexer with multiple robots, refer to Subsection 3.1.1.2,

"Setting up Pulsecoders".
The setup for when using ethernet encoders with multiple robots is as follows.

Setup on the master

Follow the procedure below to set up the Pulsecoder on the robot controller connected to the Pulsecoder.

1 Set up in the same way as steps 1 to 7 in Subsection 3.1.1.2, "Setting up Pulsecoders" or Subsection 4.1.1.2, " Setting up Pulsecoders".
The displayed Encoder setup screen is as shown below, however. When a robot controller has the Ethernet Encoder option, items [Ethernet Master RIPE Id] and [Ethernet Master Encoder] are added.



2 Position the cursor to [Ethernet Master RIPE Id] and enter the RIPE ID of the controller. The RIPE ID for the master controller of the robot ring is 1. For a slave controller, it is the [Member Index] value. For how to check [Member Index], refer to "Setup on a Slave" in Subsection 5.1.4, "Setting up the Robot Ring".
3 Position the cursor to [Ethernet Master Encoder] and enter the number of the selected Encoder. The number to enter is the encoder number that is set. When using on a line tracking interface board of the wide mini slot type, the number is 1 if the Pulsecoder is connected to the JF21 connector on the line tracking interface board or the number is 2 if the Pulsecoder is connected to the JF22 connector. When using a line tracking interface board of the mini slot type, the number is 1 if one Pulsecoder is used or the relevant number. When using 2 Pulsecoders, the numbers are the numbers that correspond to each cable tag.
4 Turn the power to the robot controller off, then on again.

> ⚠ **CAUTION**
> When using the Ethernet encoder by an Pulsecoder of a main board connection, you must set the encoder number of the master encoder to a smaller number than the encoder number of slave encoder.

Setup on a slave

1 Open the Encoder setup screen. You can select any [Encoder Number].
2 Enter the same values set for the master controller in [Ethernet Master RIPE Id] and [Ethernet Master Encoder].
3 You do not need to change other items.
4 Turn the power to the robot controller off, then on again.

> ⚠ **CAUTION**
> To use the Etherner encoder, all controllers connected with RIPE must have the same software version.

5.3 CAMERA INSTALLATION AND CONNECTION

Install a camera and connect it to a robot controller. For a queue managed tracking system, skip this section.

Selecting a camera to be used

For a visual tracking system, a digital camera(monochrome, color) and an analog camera are available.

Connecting the camera

Connect the camera to a robot controller. For details, refer to the chapter "2. CONFIGURATION" in the "SENSOR MECHANICAL UNIT/CONTROL UNIT OPERATOR'S MANUAL B-83984EN".

> ⚠ **CAUTION**
> If you want to use the Ethernet Encoder, connect the camera to the master controller of the robot ring to obtain an accurate Encoder count at the moment parts are detected.

Installing the camera

Attach the lens to the camera and install the camera over the conveyor. Orient the optical axis of the camera so that it is perpendicular to the conveyor plane.

Also install the camera so that the direction in which the conveyor moves is the size larger direction of the camera image. Generally, select the transverse direction. This ensures a wider field of view in the direction in which the conveyor moves.

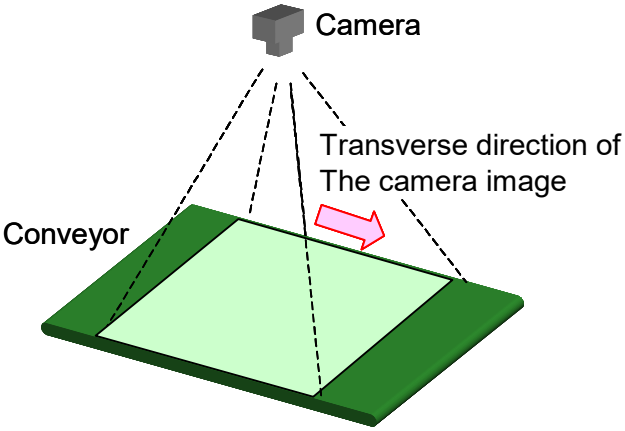


Fig. 5.3 (a) Example of mounting a camera

Installing the camera and photo eye

When the system is designed to snap an image with a camera after the signal from a photo eye is input, the sensor monitors the DI (RI) signal. After the DI (RI) signal is input, the sensor waits for the conveyor to move a specified distance, then snaps an image with the camera. The time from the input of the DI (RI) signal to the snap of an image by the vision system is about 0.1 seconds depending on the configuration of hardware used. For this reason, the expected timing of snap may be delayed. It is recommended that you install the camera in the downstream direction from the photo eye by the distance of the conveyor speed x 0.1 sec. You can install the camera farther in the downstream direction. In this case, you can snap an image

in the desired position of a part by adjusting sensor setup parameter [Trigger Offset]. For details of [Trigger Offset], see Section 6.5, “SETUP OF A SENSOR”.

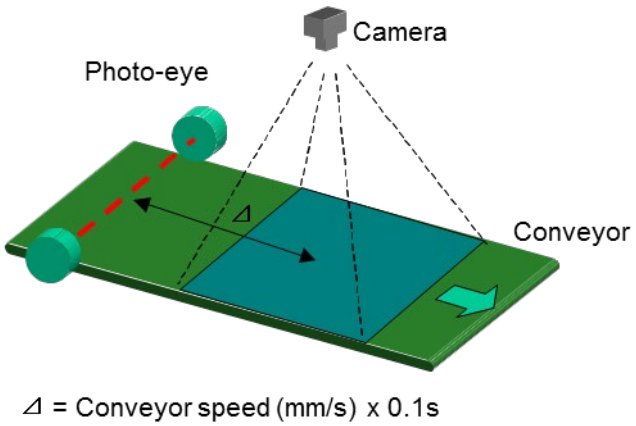


Fig. 5.3 (b) Example of mounting a camera and a photoelectric tube sensor

5.4 SETTING UP THE TOOL FRAME

Setup of a tool frame sets the tracking frame and defines the position of the tool center point (TCP) and tool posture with a pointer tool that is used in camera calibration.
When multiple robots are used, it is necessary to make this setting for all robots.

Generally, you will need two TCP settings or UTOOLs. One is for the pointer which you will use for accurate teaching of all frames and positions. The second one is for the actual gripper you will use. In this case, your gripper may have one or more picking zones and the TCP should be accurately determined for each zone of the gripper.
Fix a pointer to the wrist of a robot, then set the user tool frame at the tip of the pointer. Assign a tool frame data item to the pointer and make a setting so that the origin of the user tool frame matches the tip of the pointer. The specific setting methods include a three point method, six point method, direct list method, and two point + Z method (4-axis robot). For details of operation, refer to the section, “SETTING COORDINATE SYSTEMS” in the chapter, “SETTING UP THE ROBOT SYSTEM” in the “OPERATOR'S MANUAL (Basic Operation) B-83284EN”.

The pointer is used for tracking frame setting, camera calibration, sensor/tray position setting, and other purposes. After the completion of required work, the pointer can be detached. After the pointer is detached, however, the set tool center point is invalidated and it is necessary to set a tool center point again unless the pointer is precisely restored mechanically. For this reason, it is recommended that the mechanical design for the pointer be made to be restored to the robot in preparation for operation using the pointer after that. It is also recommended that the pointer be left mounted immediately before robot program position teaching described in Section 6.11, “TEACHING THE POSITION TO ROBOTS” or Section 7.7, “TEACHING THE POSITION TO ROBOTS.”

5.5 CALIBRATION GRID

For visual tracking camera calibration, a grid pattern is used. Prepare a grid pattern in advance. Notes and others on creating a grid pattern are described about calibration grid in the “*i*RVision OPERATOR'S MANUAL (Reference)”. FANUC provides a standard set of calibration grids. The use of them is recommended.

6 BASIC FUNCTIONS REFERENCE

This chapter explains the menu items and setup procedures when using *i*RPickTool Basic (Basic Functions) or *i*RPickTool (Basic Functions + Plug & Play).
Furthermore, examples of robot TP programs and KAREL programs are also given.

For the procedure to start up *i*RPickTool Basic (Basic Functions), refer to Chapter 3, "*i*RPICKTOOL BASIC (BASIC FUNCTIONS) STARTUP PROCEDURES" For the procedure to start up *i*RPickTool (Basic Functions + Plug & Play), refer to Chapter 4, "*i*RPICKTOOL (BASIC FUNCTIONS + PLUG & PLAY) STARTUP PROCEDURES".

The *i*RPickTool Basic menu and related items in this manual shown in the figure below.

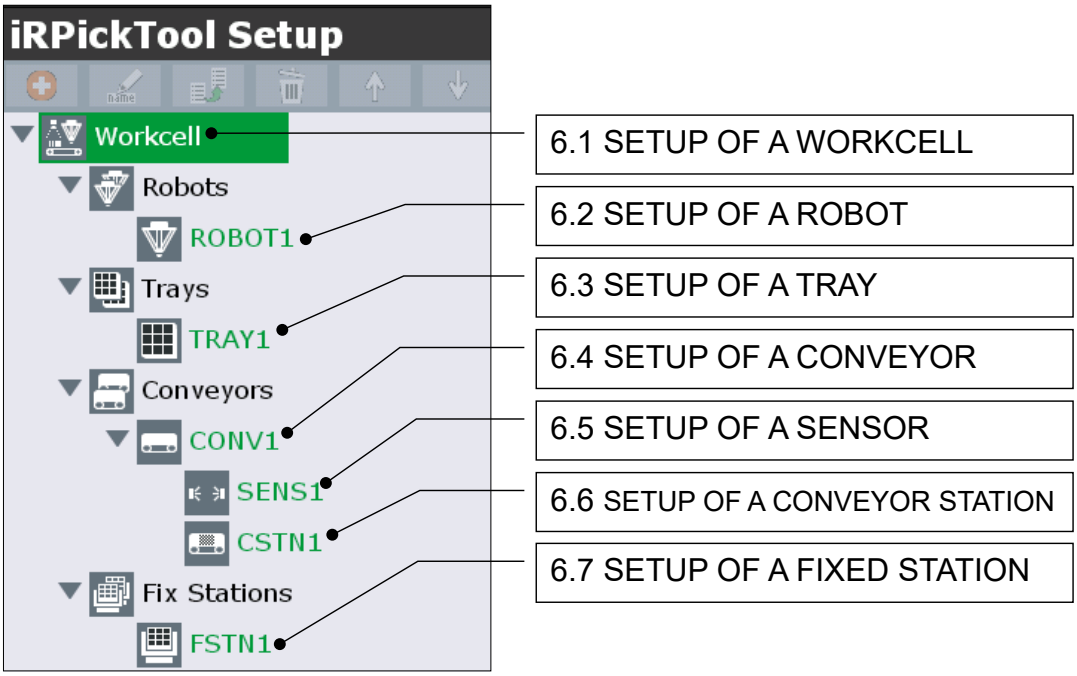


Fig. 6(a) *i*RPickTool Basic menu and related items

Besides the *i*RPickTool Basic menu items in Fig. 6(a), as a reference for the basic functions, Section 6.8, "KAREL PROGRAMS" and Section 6.9, "ROBOT PROGRAMS" explain programs, and Section 6.10, "REFERENCE POSITION SETTING" and Section 6.11, "TEACHING THE POSITION TO ROBOTS" explain robot position teaching using specific examples.

Furthermore, Section 6.12 "FINE ADJUSTMENT OF THE TRACKING MOTION" explains the method for optimizing the robot's tracking motions after finishing basic motion teaching.

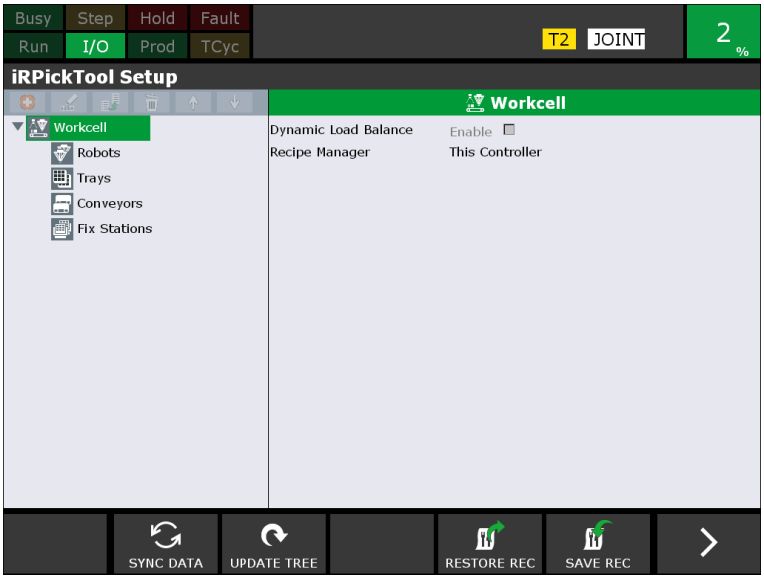
Section 6.13, "PRECAUTIONS FOR WHEN STARTING UP MULTIPLE ROBOTS" gives precautions for when starting production using multiple robots.
Section 6.14, "SETUP OF PRE-GROUPING" contains the specific setup procedure of the Pre-grouping function.

Basic operations

With *i*RPickTool, the system components such as robots and conveyors are set individually. These components are called "objects". Here, the methods are described for the operations that need to be performed in preparation for the setting of these objects, such as the creation of an object.
*i*RPickTool may be set up from any controller in a work cell.
However, you cannot setup *i*RPickTool simultaneously for two or more robot controllers with regard to robots connected by the robot ring.

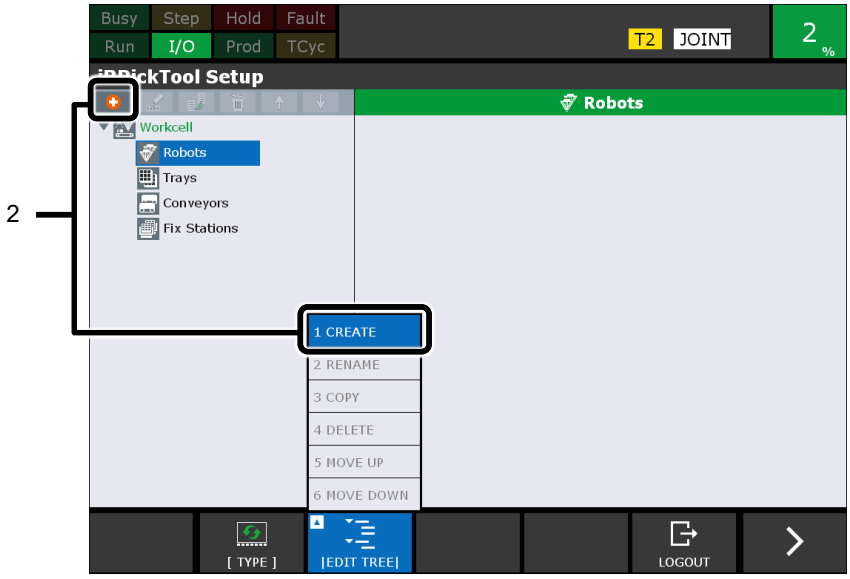
Creating a new object

On the teach pendant of the robot controller, press the [MENU] key and from the menu, select [SETUP] - [*i*RPickTool]. The following *i*RPickTool setup screen will appear.



On the *i*RPickTool setup screen, select the collection of objects that you want to create, from the tree view on the left.
For example, if you want to create a robot, follow the procedure below.

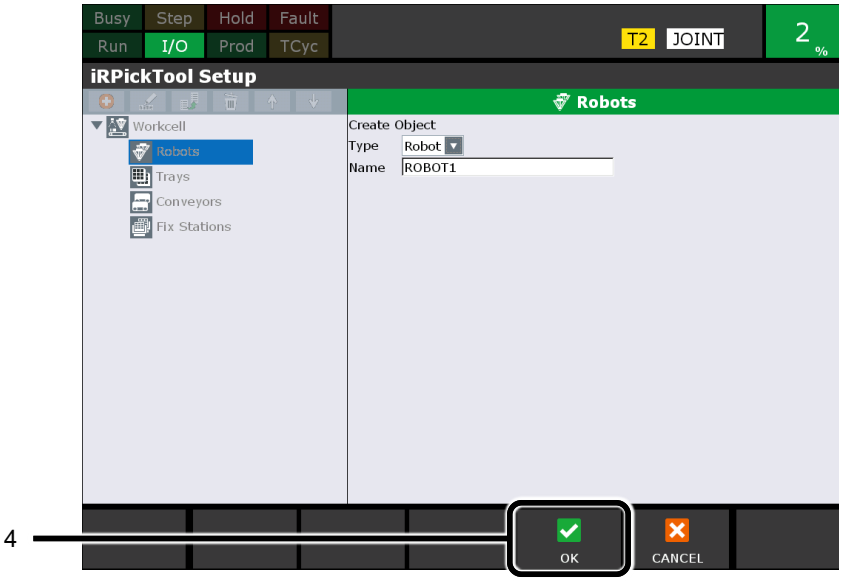
1 Select [Robots].
2 Press the create button above the tree view, or select [CREATE] from F2 [EDIT TREE].



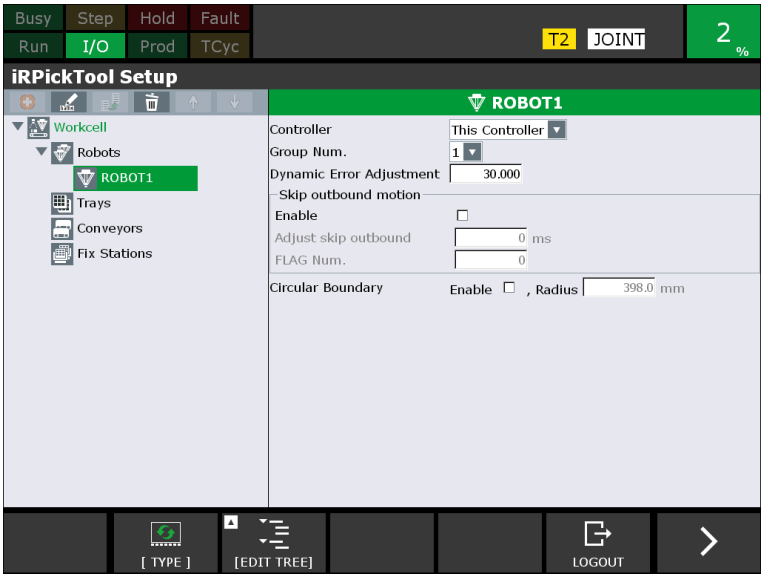
A screen to create a new object will appear.

3 On the Create Object screen, enter a name that is different from the other objects.
The name displayed initially can also be used as it is.
Input using half-width alphanumeric characters and half-width katakana. The name must contain no spaces, not contain the symbol other than underscore, and start with a letter.

4 Press F4 [OK] and create an object.
Press F5 [CANCEL] to cancel creation.



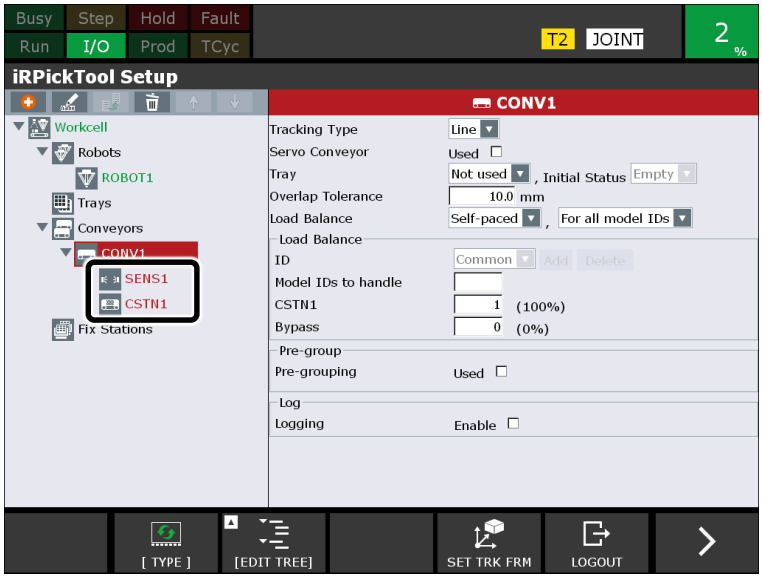
If you create a new object, the created object will be added to the tree view and its setup screen will appear. If you have created a robot, the following setup screen will be displayed.



With regard to the color of the objects that are displayed in the tree view, red indicates “not trained” status, and green indicates “trained” status.

Creating a new sensor and a new conveyor station

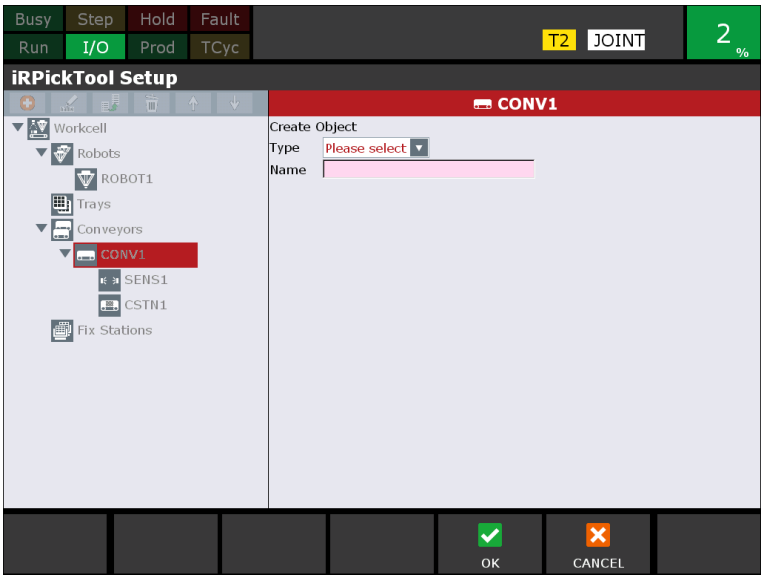
If you create a new conveyor, a sensor and a conveyor station are automatically created, as shown below.



If multiple robots perform a task on a conveyor, add a conveyor station. (Displayed as [CSTN1] in the tree view.)
Furthermore, if there are multiple sensors on a conveyor, add a sensor. (Displayed as [SENS1] in the tree view.)

If you want to create an object, follow the procedure below.

1 With a conveyor selected, press the create button, or select [CREATE] from [EDIT TREE].
A screen like the following one will appear.



2 In [Type], specify whether the object you are adding is a conveyor station or sensor.
The default name appears in [Name], as when you create a new object of another type.
Create a new object in the same way as with an object of another type.

Setting up an object individually

In the tree view on the left side of the *i*RPickTool setup screen, select an object you want to set up. Then, set each item in the setup screen on the right side.

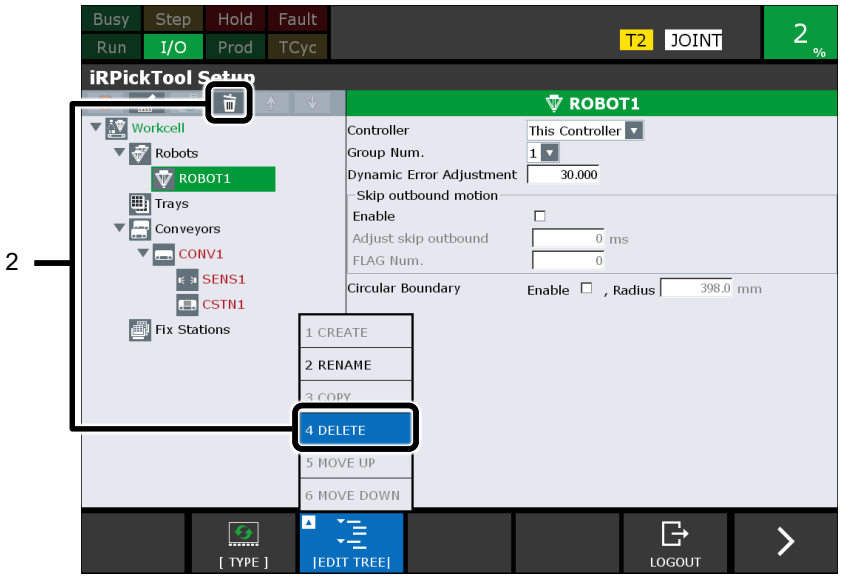
With regard to the color of the objects that are displayed in the tree, red indicates "not trained" status, and green indicates "trained" status.

For details on each object, refer to Section 6.1, "SETUP OF A WORKCELL" to Section 6.7, "SETUP OF A FIXED STATION". For Plug & Play, see also Section 7.1, "SETUP OF A WORKCELL" to Section 7.4, "SETUP OF A FIXED STATION."

Deleting an object

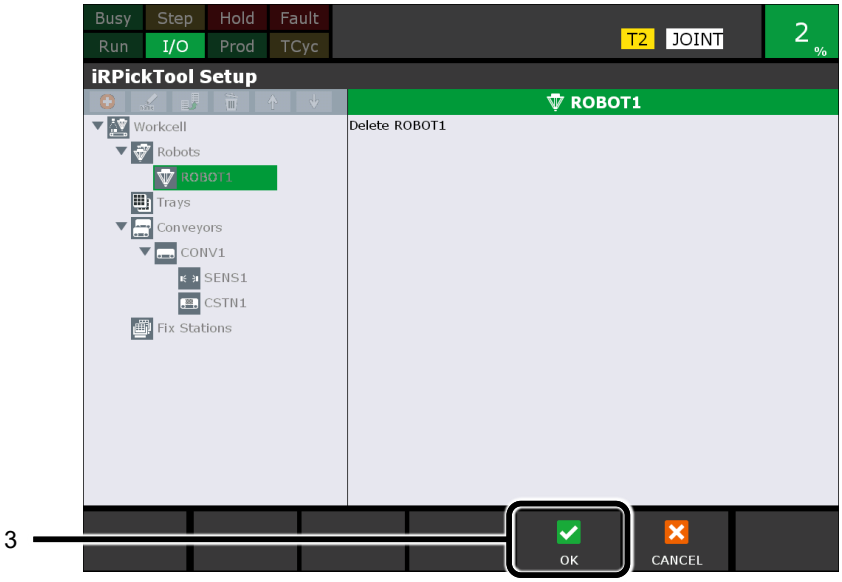
If you want to delete an object, follow the procedure below.

1 In the tree view at the left of the *i*RPickTool setup screen, select the object that you want to delete.
2 Press the delete button above the tree view, or select [DELETE] from [EDIT TREE].



A screen as shown below is displayed.

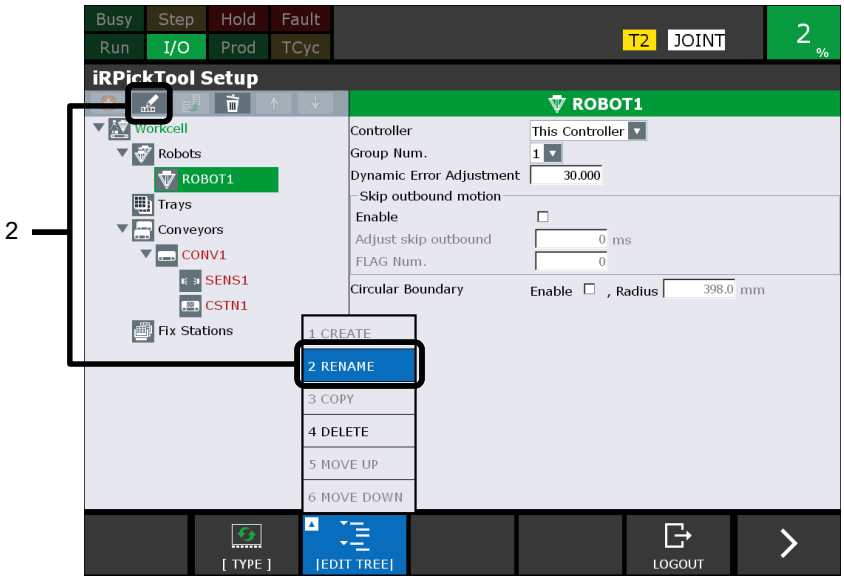
3 Press F4 [OK] key to delete the object. To quit deleting the object, press F5 [CANCEL] key.



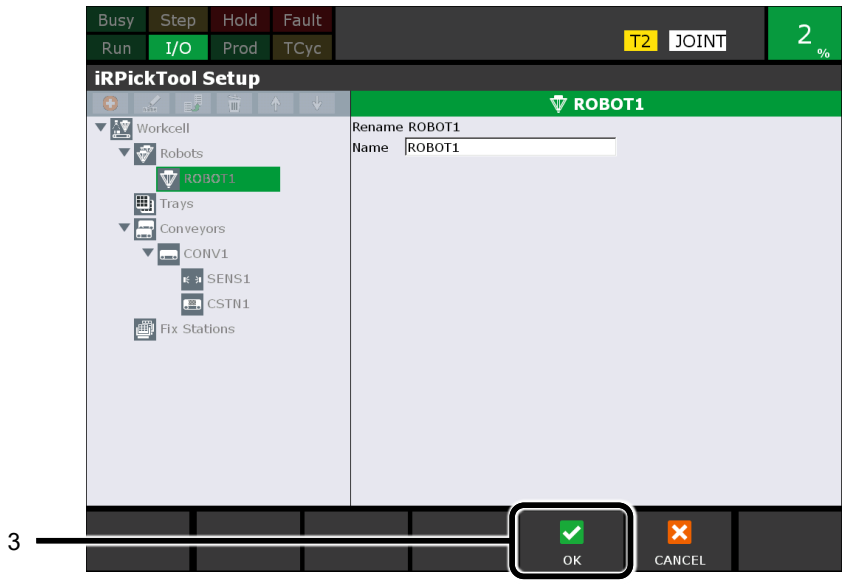
Renaming an object

If you want rename an object, follow the procedure below.

1 In the tree view at the left of the *i*RPickTool setup screen, select the object that you want to rename.
2 Press the rename button above the tree view, or select [RENAME] from [EDIT TREE].



A screen as shown below is displayed.
3 Enter the new name of the object, and press F4 [OK] key.
To quit renaming the object, press F5 [CANCEL] key.

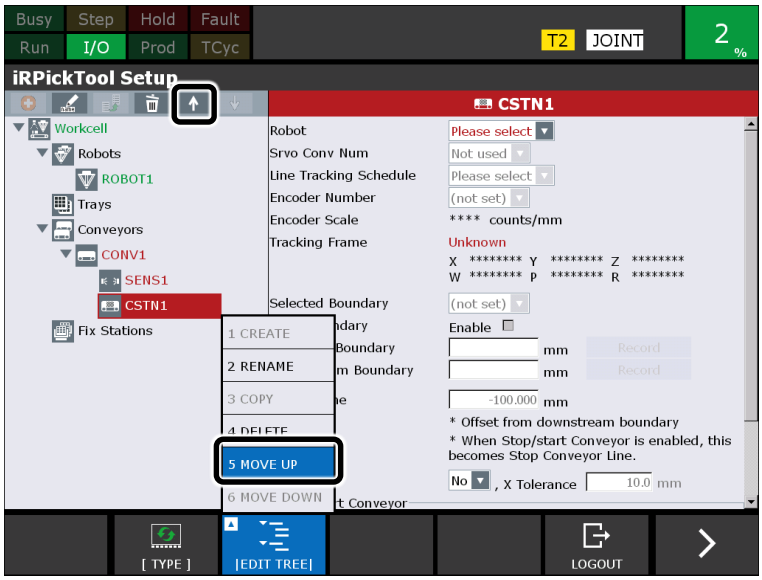


Moving an object in the order

To replace conveyor stations, change the order of the conveyor stations (objects).

> ⚠ **CAUTION**
> Conveyor stations are placed from the upstream of the conveyor to the downstream in the same order they are displayed in the tree view on the screen.

In the tree view on the left side of the *i*RPickTool setup screen, select an object you want to move in the order. To move it up in the order, click [5. MOVE UP] of F2 [EDIT TREE] key or the button above the tree. To move it down, click [6. MOVE DOWN] of F2 [EDIT TREE] key or the button above the tree.

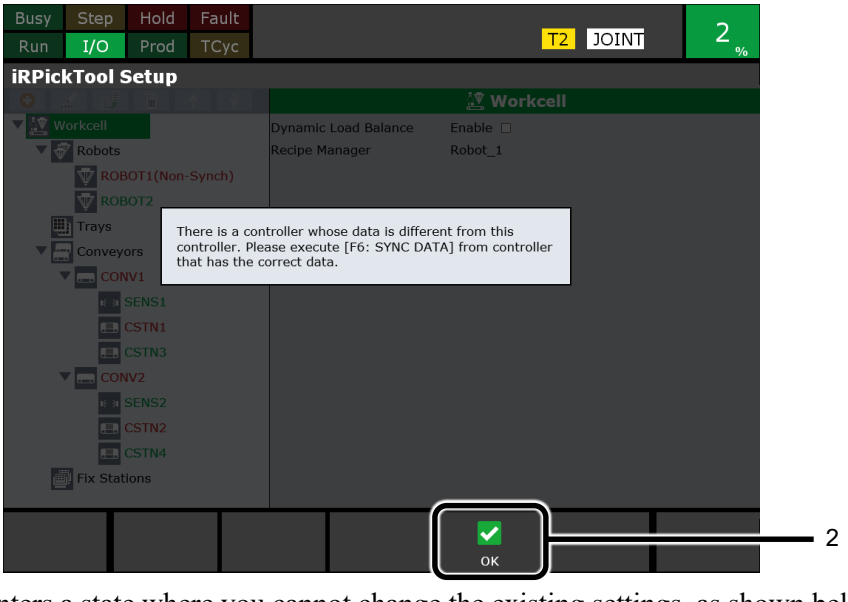


Synchronizing data

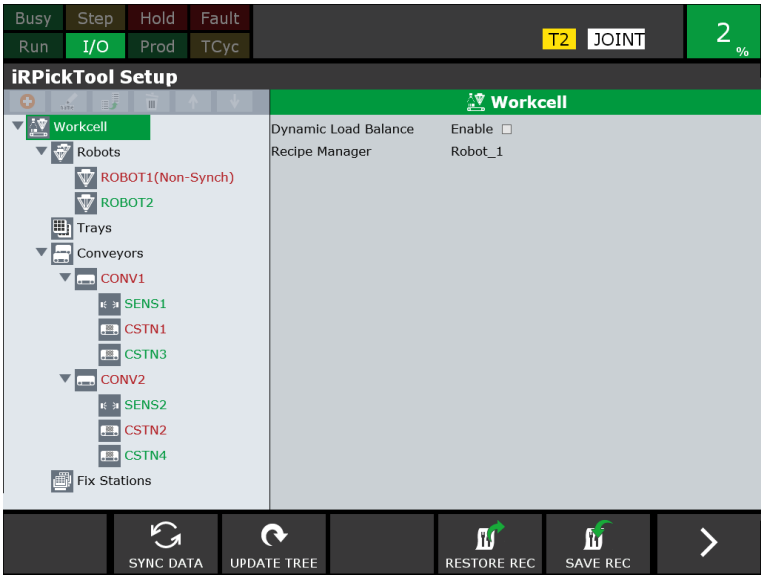
The settings of *i*RPickTool are data stored in the system variables of the robot controller. The same data is set in all the robot controllers in the system. If the robot ring is set, the settings you make using the screens described in this chapter are automatically transferred to and saved in each robot.

You can set up *i*RPickTool even if the system has any robot controller whose power is off. In that case, however, the offline robot controller is left with inconsistent data. If this happens, perform the procedure described below, with all the robot controllers in the system powered on, to synchronize the data so that the *i*RPickTool settings are the same for all the robot controllers.

1 If you open the *i*RPickTool setup screen when there is any data inconsistency, a screen as shown below is displayed.
2 Press F4 [OK] key.



The screen enters a state where you cannot change the existing settings, as shown below.



3 Check the settings of each individual object to see whether it is appropriate to transfer those settings to the other robot controllers. To perform synchronization based on the settings of another robot controller, press F5 [LOGOUT] key to close the *i*RPickTool setup screen. Then, open the *i*RPickTool setup screen for that other robot controller and check its settings as instructed above.

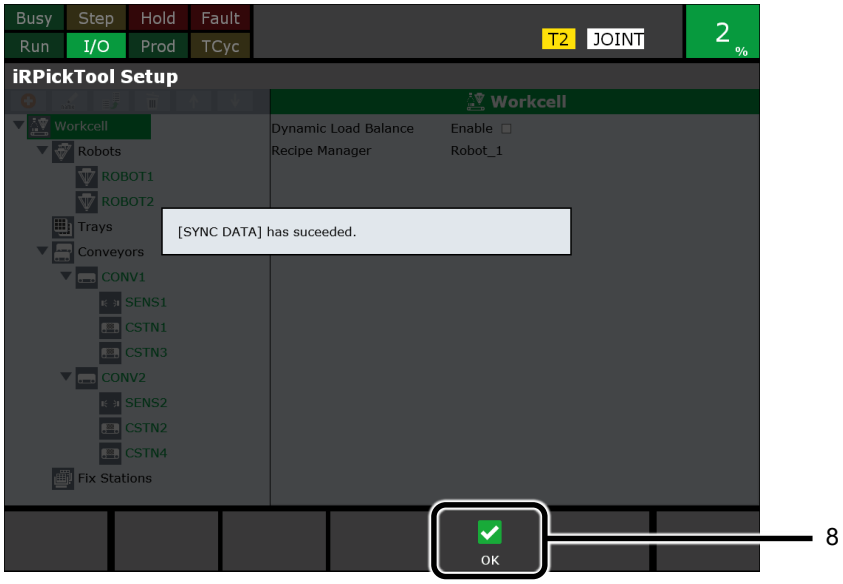
> **⚠ CAUTION**
> Once you perform synchronization as instructed below based on unintended settings, you may be unable to restore the data to the original settings. Check carefully to make sure that it is appropriate to transfer the settings of the selected robot controller to the other robot controllers.

4 After making sure that it is appropriate to transfer the settings to the other robot controllers, press the [NEXT] key on a desired *i*RPickTool setup screen to display the function key shown below.

5 Press F1 [SYNC DATA] key.
When a screen as shown below is displayed.

6 Press F4 [OK] key.

7 When data synchronization is complete, a screen as shown below is displayed.
8 Press F4 [OK] key.



Refreshing the tree

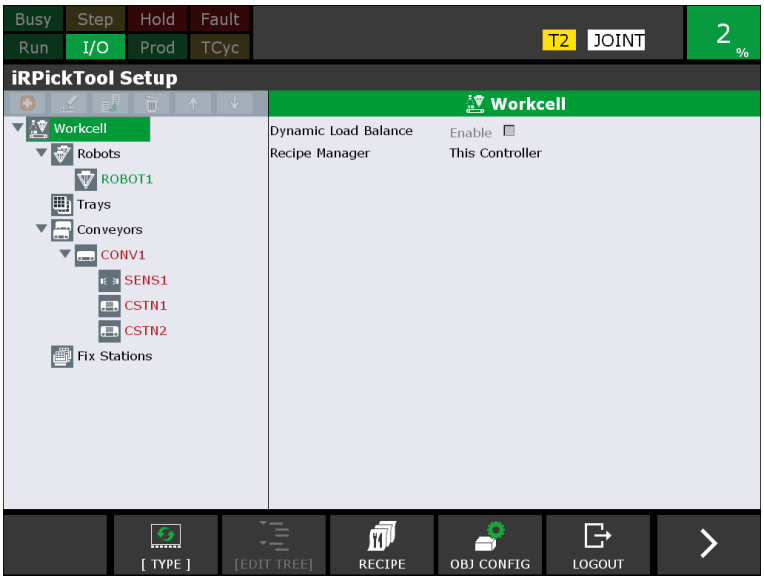
Since the *i*RPickTool setup screen does not automatically reflects the latest status, you need to execute [UPDATE TREE] to explicitly make the setup screen reflect the latest status. Specifically, use this function in the following cases.

- If an offline controller goes online when the *i*RPickTool setup screen is open, this change in the status is not automatically reflected in the setup screen.

In these cases, press the [> (NEXT)] key on a desired *i*RPickTool setup screen to display the function key shown below and then press F2 [UPDATE TREE] key.

6.1 SETUP OF A WORKCELL

The setup screen for a workcell is as below.



The details of each item are as follows.

[Dynamic Load Balance]

If a robot goes offline on a conveyor for which load balancing is enabled, the parts to be handled by that robot are handled by other robots. If the robot goes online, it resumes the handling of the parts it is supposed to handle.

[Recipe Manager]

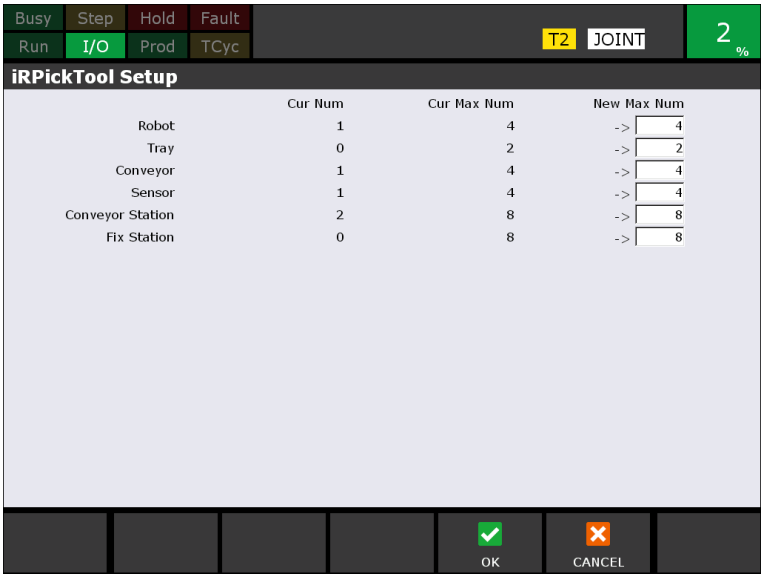
Recipe Manager is the controller in which the *i*RPickTool Recipes are stored. For more information on Recipes, see the section 9.1, “EXCHANGE WORKCELL WITH USING RECIPE”. The Recipe function allows you to store and restore data for each product that the robot uses. All recipes will exist only in the “Recipe Manager” controller.

NOTE
Recipe Manager cannot be changed by default.

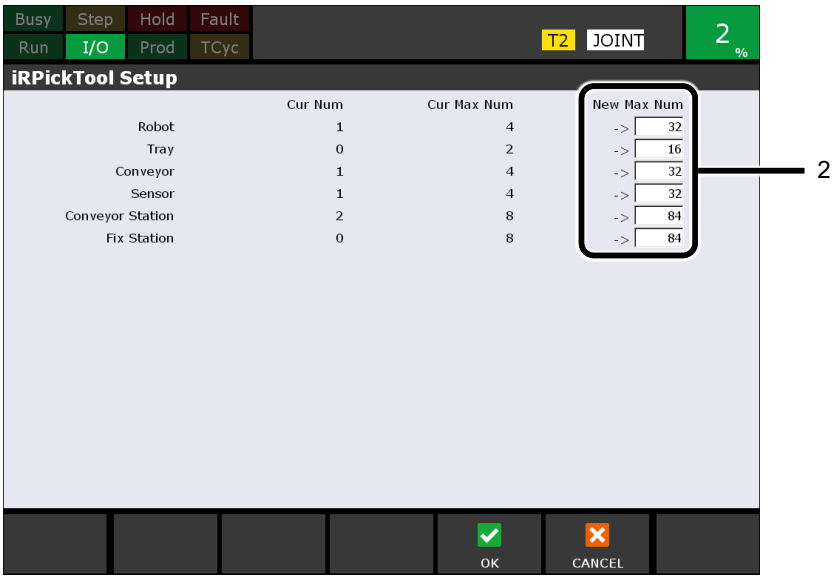
6.1.1 Setting the Number of Objects

On the workcell setup screen, you can modify the maximum number that can be set for each object. The procedure for changing the maximum number is as follows.

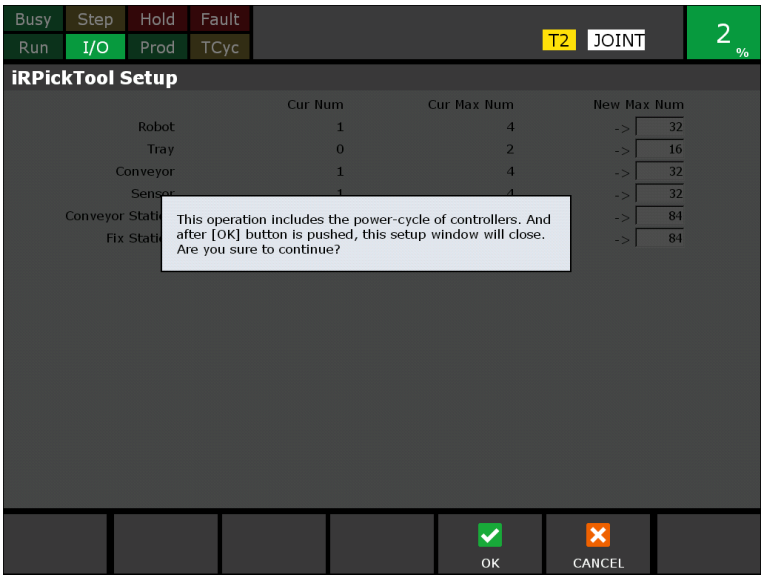
1 Press F4 [OBJ CONFIG] key. A screen as shown below is displayed.



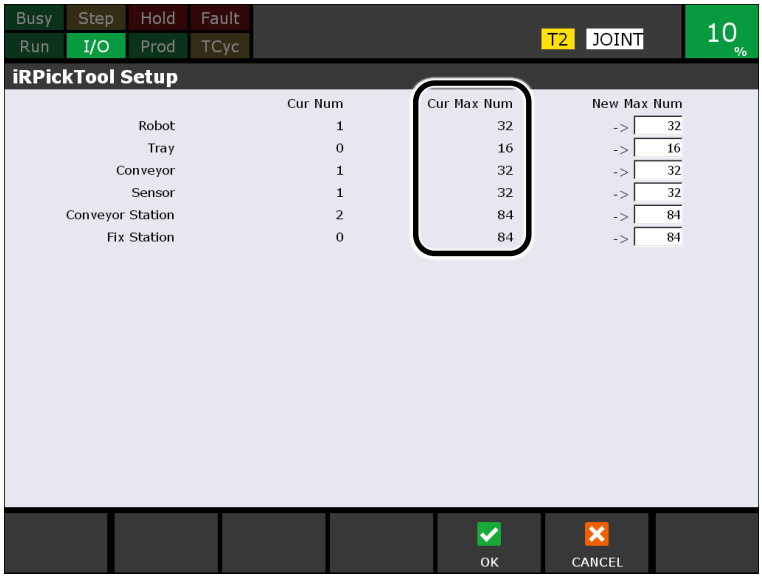
2 In [New Max Num] of the object that you want to increase, enter the new maximum number of objects after the increase.



3 Press F4 [OK] key. Then, a message as shown below is displayed.



4 Press F4 [OK] key. Then, all the controllers for which RIPE is set are automatically restarted. If you press [OBJ CONFIG] key again after the restart, the current maximum numbers of objects have changed to the values you set, as shown below.



NOTE
1 Regardless of the maximum number of sensors set here, the maximum number of sensors that can be simultaneously operated by one robot is four.
2 The maximum number that can be modified for each object shown Table 6.1.1(a).

Table 6.1.1(a) Maximum number that can be modified for each object

Object	Maximum number
Robot	32
Tray	16
Sensor	32
Conveyor	32
Conveyor Station	84
Fix Station	84

6.1.2 Recipe

A recipe is a summary of an entire workcell of *i*RPickTool.
By using different recipes, you can replace an entire workcell. For example, this is useful in cases such as:

- When changing from one product to another
- When running production for the same product using different grippers, trays, robots, sensors, conveyor stations and fixed stations.

> ⚠ **CAUTION**
> When using a recipe, the software versions of all the controllers connected via the ROS Interface Packets Over Ethernet (RIPE) need to match.

When a recipe is used, one recipe (called the 'active recipe') is selected, and the workcell corresponding to its content will be executed.

For details, refer to Section 9.1, "EXCHANGE WORKCELL WITH USING RECIPE".

6.2 SETUP OF A ROBOT

The setup screen for a robot is as below.

The details of each item are as follows.

[Controller]

Select a controller that corresponds to the robot. If RIPE is not set, only "This Controller" is displayed.

[Group Num.]

Select a group number that corresponds to the robot. If multiple groups are not ordered by the controller selected in [Controller], only 1 is displayed.

[Dynamic Error Adjustment]

Set this item after completing the teaching of the basic motions. You do not need to set it before completing the teaching of the basic motions. For details, refer to Section 6.12, "FINE ADJUSTMENT OF THE TRACKING MOTION".
This means the error that is caused by tracking delay of the robot to the conveyor. Set the appropriate value with following 6.12.3 "Adjustment of the Dynamic Error of the Robot". But this set value is enabled for a conveyor except the servo conveyor.
When you use a servo conveyor, you set the appropriate value to the following system variable for each.

$SLTK_GRP[number of servo conveyor group].$SRVO_DELAY

[Skip outbound motion]

Set these items after completing the teaching of the basic motions. You do not need to set them before completing the teaching of the basic motions. For details, refer to Section 9.4, "SKIPPING THE TRACKING MOTION OUTSIDE THE AREA".

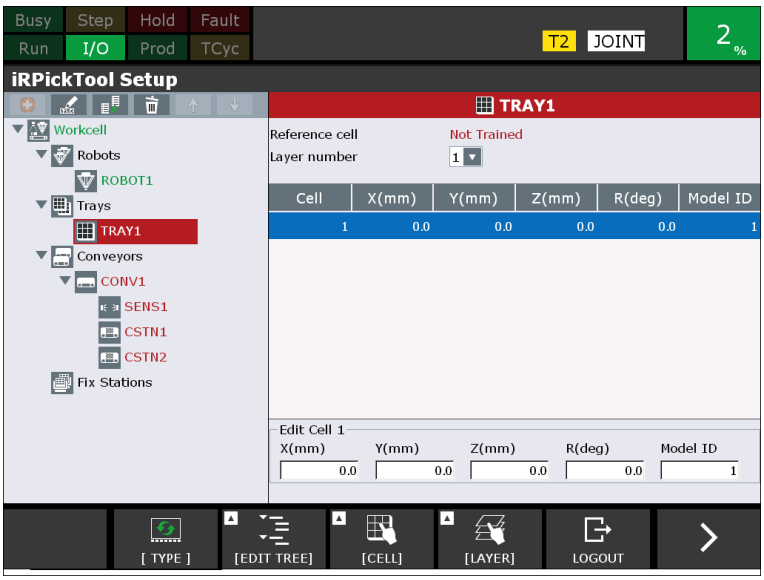
[Circular Boundary]

If you want to set a circular boundary, enable this and set a radius of 0 to 5000.
The [Circular Boundary] item is displayed on the conveyor station setup screen only when circular boundary is enabled on the selected robot.

When circular boundary is enabled, you cannot use the pre-grouping function and Y sort function of the tracking frame.

6.3 SETUP OF A TRAY

Set up a tray.
The setup screen for a tray is as below.



If a fixed station is not used and the conveyor does not use a tray, you do not need to set a tray. In that case, skip this section.

The details of each item are as follows.

[Reference cell]

The offset for each cell is calculated from X, Y, Z and R for each cell and X, Y, Z and R for the reference cell.

If you press F3 [SET REF CELL], the values for Cell 1 will be set for the reference cell.
In the case of a conveyor that uses a tray, if you set the reference position on the sensor setup screen, the values for Cell 1 will be set for the reference cell.

In the case of a fixed station, press F3 [SET REF CELL] and set the reference cell, then teach the robot position for Cell 1.
In the case of a conveyor that uses a tray, set the reference position on the sensor setup screen, then teach the robot position for Cell 1. In the case of a conveyor that uses a tray, you can omit the step to set the reference cell on the tray setup screen.

If you want to make fine adjustments to the robot position for a specific cell after teaching the robot position for Cell 1, change the coordinates for each cell.

After changing the reference cell, you need to set the reference position of the sensor and teach the robot position again.

[Layer number]

When multiple layers are set, the layer number is used to switch the layer whose cell list is displayed.

NOTE
1 When trays are handled by the conveyor, the order in which you add cells is not relevant to the order in which parts are put on (or taken from) trays during execution. Parts are handled sequentially from the cell on the downstream side as appropriate for the situation. If you want to control the order intentionally, use a 'layer', which is described in Subsection 6.3.2, “Layer Operation.”
2 Up to 160 cells can be added per tray.

[Edit Cell 1] to [Edit Cell 160]

Edit cells. In accordance with the cell that is selected, the item name changes from [Edit Cell 1] to [Edit Cell 160].

[X(mm)], [Y(mm)], [Z(mm)] and [R(deg)]

Enter the coordinates of the highlighted cell in the tray frame.

The procedure to set coordinates is explained below, using the case where a tray is used to handle four parts.

First, decide on the position of the tray frame. For example, suppose that you have decided to place it at a corner of the top side of a box, as shown in the figure (Fig. 6.3(a)) below.

NOTE

When trays are used on the conveyor, the posture of the tray frame is determined as follows.
The Z-axis direction of the tray frame is the same as that of the tracking frame.
Also, the XY plane of the tray frame is the same as that of the tracking frame.
Therefore, if you decide on the X-axis direction of the tray frame, the directions of all the other axes are automatically determined.

While you can define the tray frame freely on the tray, select a position that is easy to touch up. When the tray frame on the conveyor is taught to the robot, the origin of the tray frame and a point in the X-axis direction need to be touched up with the TCP of the robot. Also, set the tray frame on the fixed station as the user frame.

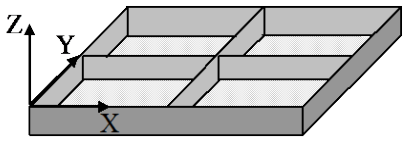


Fig. 6.3 (a)

Add cells to a tray, and enter the coordinates of each cell. For example, if the tray in Fig. 6.3 (a) is viewed from above, it will be like the figure below.

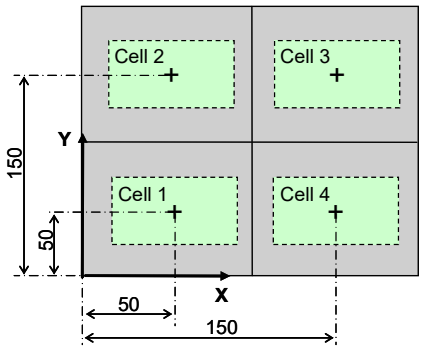
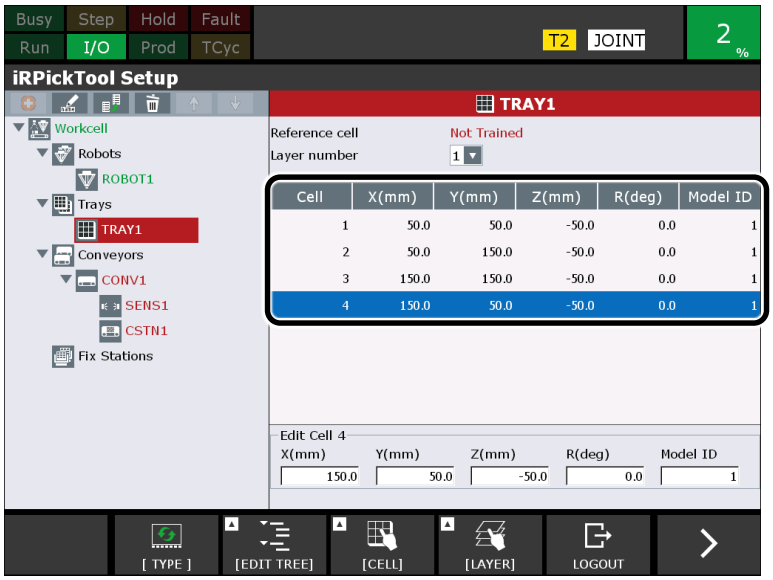


Fig. 6.3 (b)

The cross mark (+) represents the center position of each individual cell. The coordinates of these cells are as follows.



Setting of a tray when parts are different in height

When parts to be handled are different in height, use the Z value of the cell.

For example, consider a case where there are two cells and the part to be placed in each of them has a different height, as shown in the figure below.

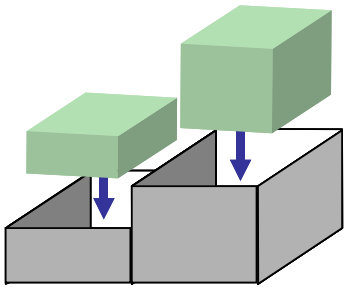
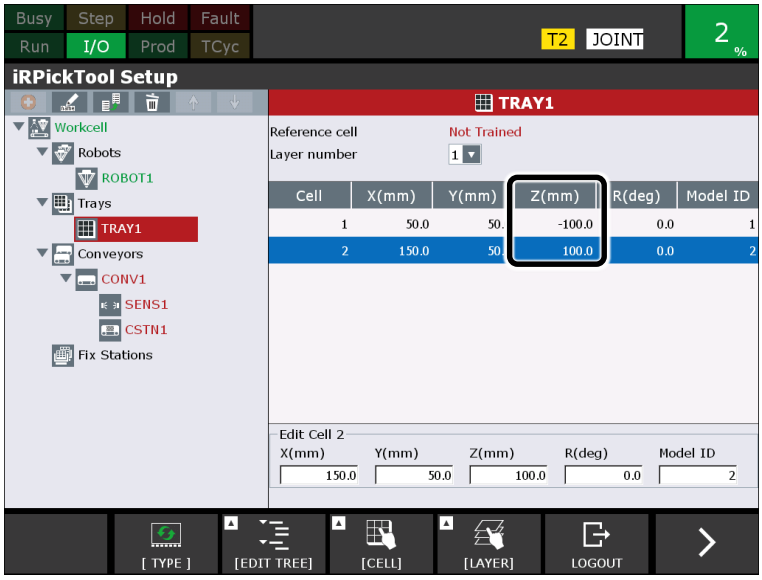


Fig. 6.3 (c) Example of when parts have different heights

In this case, change the Z value of each cell according to the height of the part. For example, the tray may be set as shown below. Here, since the model of each part is different, different model IDs are set.



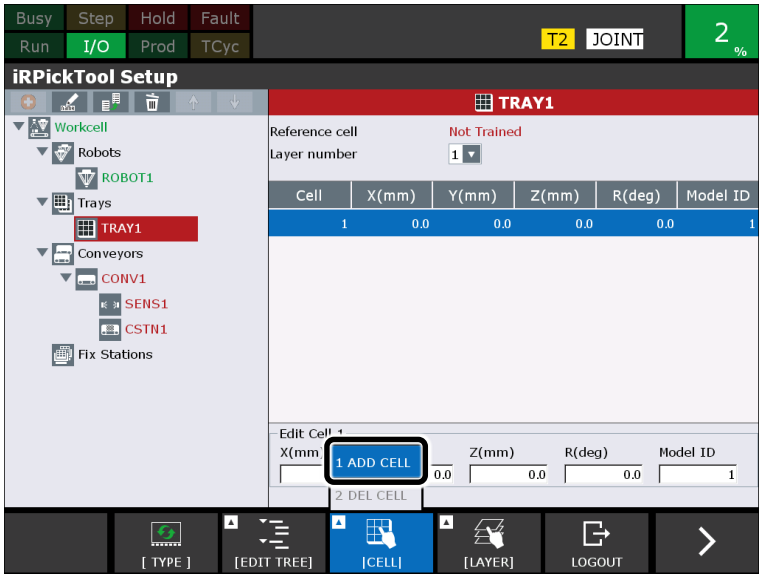
[Model ID]

A model ID can be set for each cell. Model IDs are used in such cases as when you want the robot TP program to perform a different type of processing according to which cell is allocated.
There is no need to change the default value of 1 unless any other number is particularly required.

Examples of using model IDs are given in Chapter 8, “EXAMPLES OF SETUP IN ACCORDANCE WITH USE”.

6.3.1 Cell Operation

To add a cell to a tray, select [1. ADD CELL]. To delete a highlighted cell, select [2. DEL CELL].



NOTE

1 When trays are handled by the conveyor, the order in which you add cells is not relevant to the order in which parts are put on (or taken from) trays during execution. Parts are handled sequentially from the cell on the downstream side as appropriate for the situation. To control the order intentionally, use "Layer" described later.

2 Up to 160 cells can be added per tray.

Setting of a tray when a robot hand that can hold multiple parts is used

When a robot hand that can hold multiple parts is used, the robot can handle more than one part in a single motion.

For example, consider a tray that can contain 12 parts with a robot hand capable of holding three parts, as shown below. Each dotted-line circle represents a part.

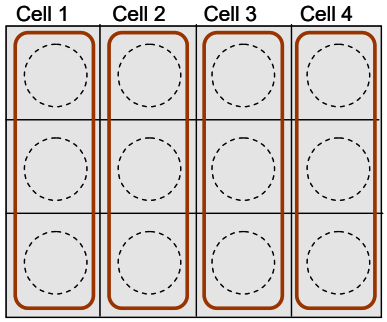
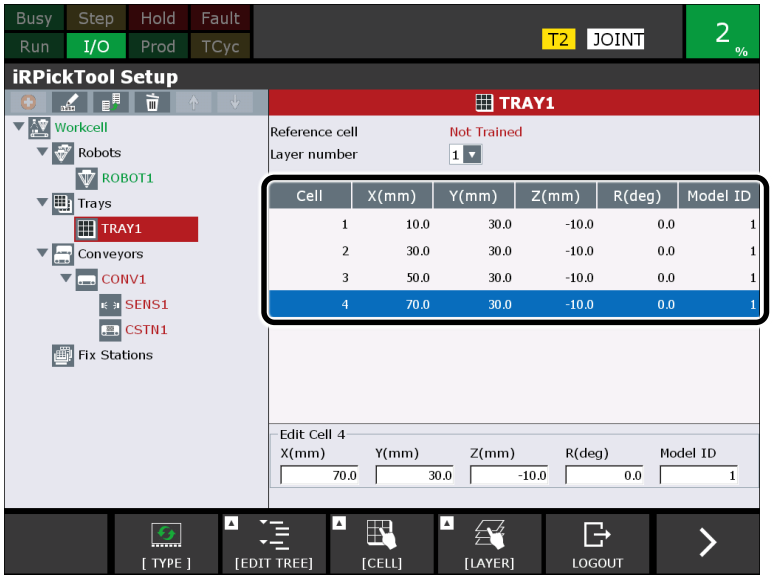


Fig. 6.3.1(a) Example of a tray for a hand that holds three parts

The robot handles three parts at a time. In such a case, three parts that can be handled at a time are considered to be a cell. Therefore, the tray has four cells. For example, the tray may be set as follows.



6.3.2 Layer Operation

When parts are laid in layers in the tray, define each layer individually. To add a layer, select [1. ADD LAYER]. To delete the layer selected in [Layer number], select [2. DEL LAYER].
Normally, create all cells in a single layer. If there are many cells of the same layer in the tray, the order in which those cells are handled is determined as appropriate for the situation. For example, a robot may handle the cells within the tracking area sequentially, beginning with the one on the downstream side.

Multiple layers are used when control needs to be exerted over the order in which parts are placed in the tray - e.g., when the tray has multiple layers and a part cannot be placed in an upper layer unless the placement of parts in the lower layer is completed. When placing parts in a tray, set [Initial Status] to [Empty] for the conveyor and fixed station, as described later. This prevents a part from being placed in an upper-layer cell having a larger layer number when any lower-layer cell having a smaller layer number is empty.

The figure below shows an example of using layers. Parts are piled in three layers in a box. In this case, the tray has three cells. Priorities are given as follows. A part cannot be placed in the second or third layer unless a part is placed in the first layer. Likewise, a part cannot be placed in the third layer unless a part is placed in the second layer.

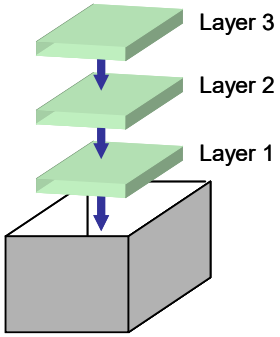
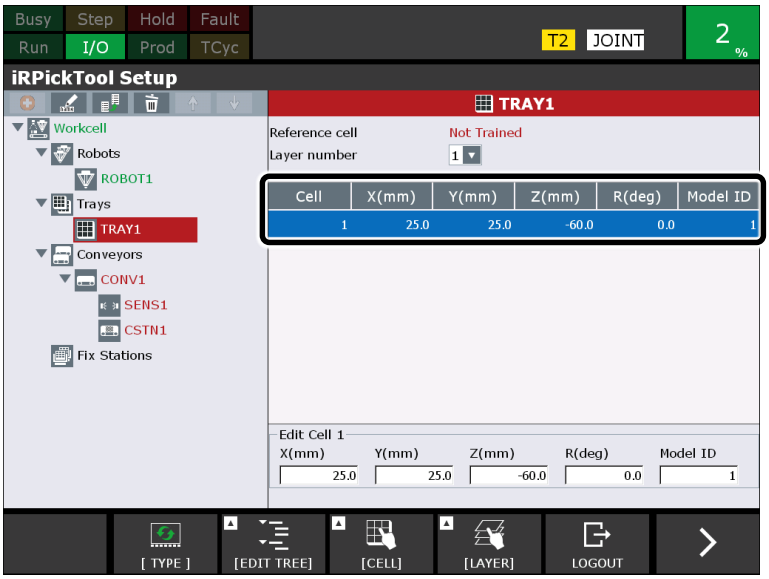


Fig. 6.3.2(a) Example of when parts are stacked in multiple layers in a tray

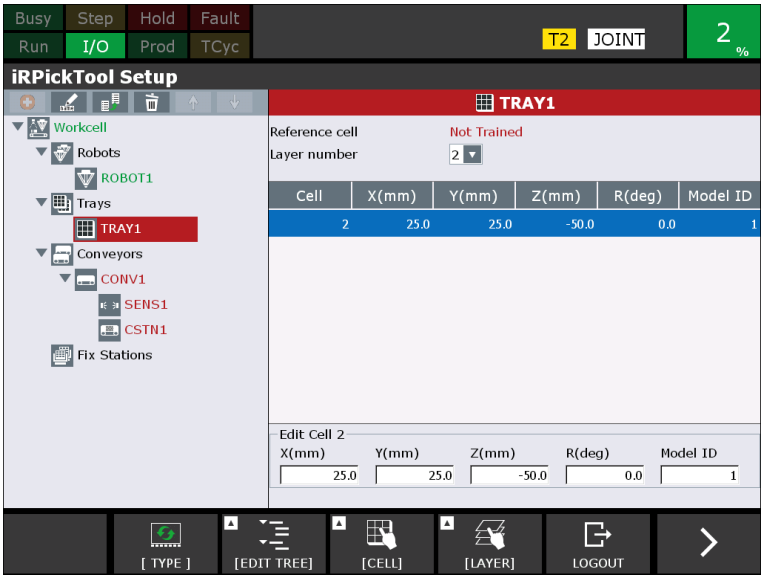
In this case, set the tray as instructed below.

1 First, set Cell 1 in the first layer as follows.

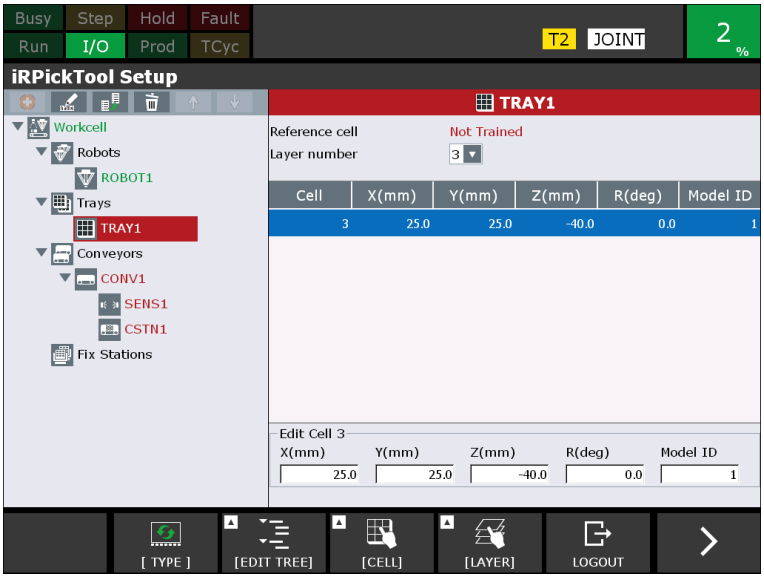


2 Next, to add the second layer, select [1. ADD LAYER] from F4 [LAYER] key.
When a screen as shown below is displayed, specify 1 as the number of cells for the second layer and –50 mm as the value of Z. Then, press F4 [OK] key.

3 Layer 2 you have added is selected. Enter 25.0 mm as the values of X and Y for Cell2, as with Cell 1.



4 Likewise, add Layer 3 with the value of Z specified, and then set the values of X and Y for Cell 3.



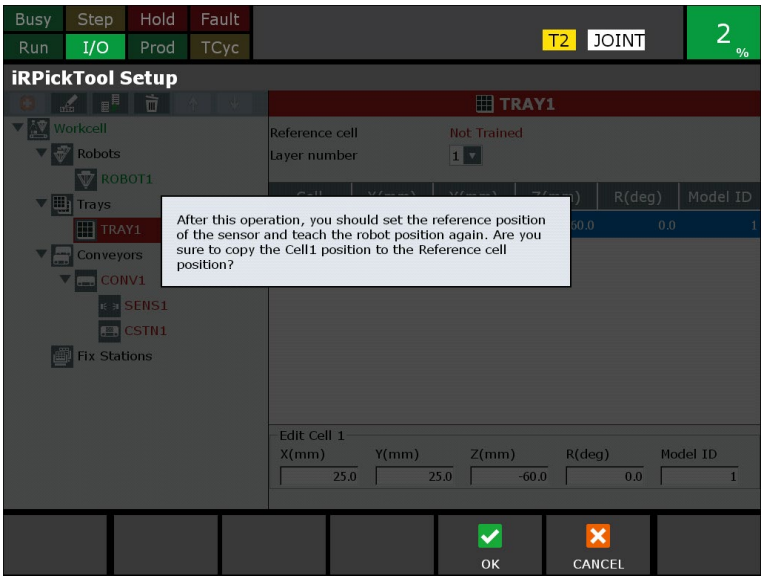
As described above, set Cell 1 as Layer 1 (bottom layer), Cell 2 as Layer 2 (middle layer), and Cell 3 as Layer 3 (top layer). The values of X and Y of each cell are the same, and a cell in a higher layer has a greater value of Z.

In addition, set [Initial Status] to [Empty] for the conveyor and fixed station that use this tray, as described later. If this item is set as mentioned here, the data of Cell 2 or 3 is not allocated to the robot unless a part is placed in Cell 1 having the smallest layer number, thus preventing a part from being placed in these locations.

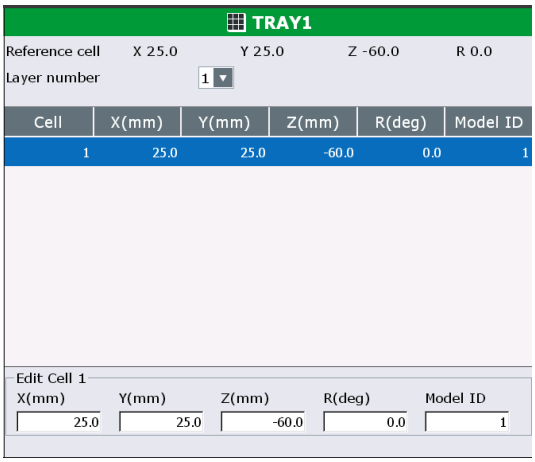
6.3.3 Setting the Reference Cell

Set a reference cell. Carry out setting of a reference cell immediately before Section 6.11, “TEACHING THE POSITION TO ROBOTS”.

If you press F3 [SET REF CELL], a message that prompts you to set the sensor reference position and teach the robot position again will appear, as shown below.



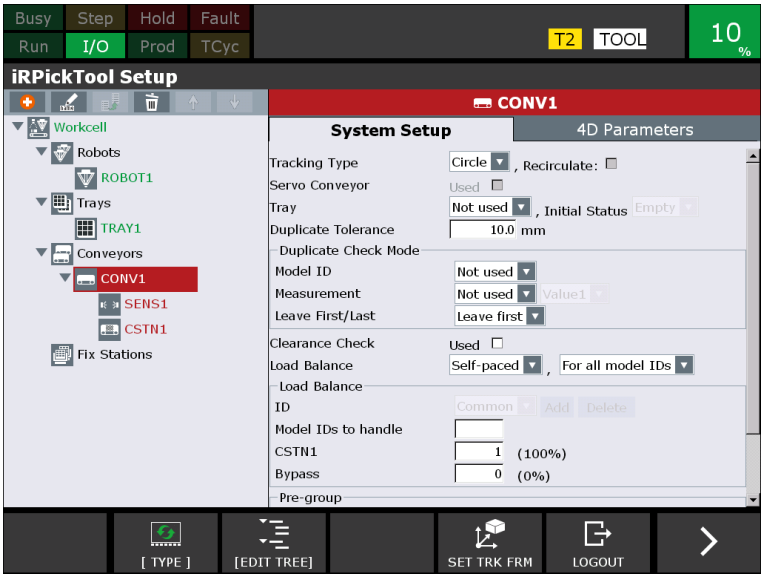
If you press F4 [OK], 'Not Trained (red letters)' disappears from [Reference cell] and the X, Y, Z and R values that you have set in [Edit Cell 1] will be displayed.



6.4 SETUP OF A CONVEYOR

Set up a conveyor. The detailed settings for a conveyor's child objects, a sensor and a conveyor station, will be described in Section 6.5, “SETUP OF A SENSOR” and Section 6.6, “SETUP OF A CONVEYOR STATION”.

The setup screen for a conveyor is as below.



[Tracking Type]

Select the line tracking or the circular tracking. If you want to use the ling tracking, select “Line”. Otherwise, select “Circle”. If you change it, child objects like the sensor and the conveyor station will be initialized.

[Recirculate]

If you put a check for [Recirculate], which is displayed only when [Circle] is selected for [Tracking Type], a tray for which the process is not yet complete will be taken over from the most downstream station by the most upstream station.
For example, in cases where a tray on a circular conveyor is not completely filled, you can recirculate it and fill the tray.

The conditions in which you can set [Recirculate] are as follows:

- [Circle] has been selected for [Tracking Type]
- [Tray] is being used
- Two or more conveyor stations have been added to the child object

A robot circulates through the conveyor top until the tray will be completed.

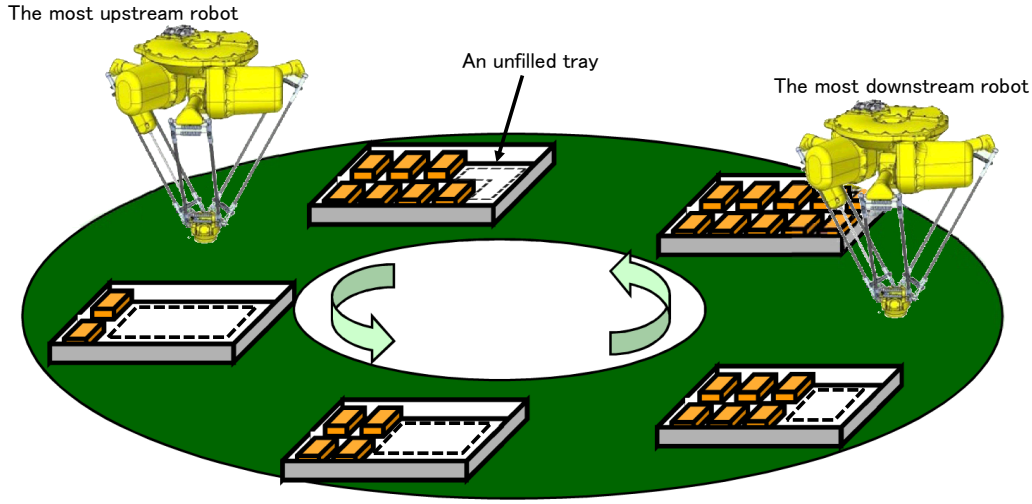


Fig.6.4 (a) Image of recirculation of an unfilled tray

[Servo Conveyor]

Only when the [Tracking Type] is set to “Line”, you can set this item. When you use the servo driven conveyor, check the [Used] box. If you enable it, child objects like the sensor and the conveyor station will be initialized. To use the servo conveyor, you set up the servo conveyor setting in advance. For details, see Section 9.7, “SERVO CONVEYOR LINE TRACKING FUNCTION”.

[Tray]

When using trays on the conveyor, select the tray to be used. When not using trays, select [Not used].

[Initial Status]

When placing parts into empty trays carried on the conveyor, select [Empty]. When picking up parts from filled trays carried on the conveyor, select [Full].

[Duplicate Tolerance]

If the vision system detects more than one identical part, this tolerance is used to determine whether the detected parts are the same. Normally, when the same part is detected more than once, no current part position completely matches others due to detection error or some other factor. Set a tolerance so that the vision system retains only one of the multiple detected parts.

If the detected parts are identified as being the same, the vision system retains only one of the parts, in accordance with the setting in [Duplicate Check Mode]. Normally, set this value to 10 mm or so.

[Duplicate Check Mode] box

Specify which result is to be retained when the duplicate check identifies detected parts as being the same.

[Model ID]

Select from [Not used], [Leave max.], [Leave min.], [Overwrite only model ID by max. value], and [Overwrite only model ID by min. value].

[Measurement]

When [Model ID] is set to [Not used], select from among [Not used], [Leave max.], and [Leave min.]. When [Model ID] is [Leave max.] or [Leave min.], select from among [Not used], [Leave max. in identical model ID], and [Leave min. in identical model ID].

No selection can be made when [Model ID] is [Overwrite only model ID by max. value] or [Overwrite only model ID by min. value]. The item is automatically set to [Not used].

When a setting other than [Not used] is selected, select which measurement value to use from [Value 1] to [Value 10].

[Leave First/Last]

When [Model ID] is [Not used] and [Measurement] is [Not used], select from among [Leave first] and [Leave last].
When [Model ID] is [Not used] and [Measurement] is [Leave max.] or [Leave min.], select from among [Leave first in identical meas.] and [Leave last in identical meas.].
When [Model ID] is [Leave max.] or [Leave min.] and [Measurement] is [Not used], select from among [Leave first in identical model ID] and [Leave last in identical model ID].
When [Model ID] is [Leave max.] or [Leave min.] and [Measurement] is [Leave max. in identical model ID] or [Leave min. in identical model ID], select from among [Leave first in identical model ID and meas.] and [Leave last in identical model ID and meas.].
When the [Model ID] is [Overwrite only model ID by max. value] or [Overwrite only model ID by min. value], no selection can be made. The item is automatically set to [Leave first (other than model ID)].

The following list describes the behavior of each setting:

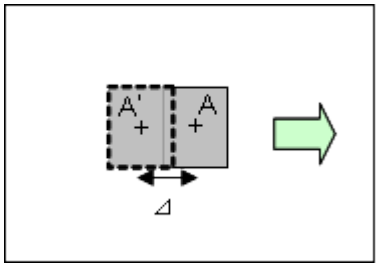
- When [Model ID] is [Not used], [Measurement] is [Not used], and [Leave First/Last] is [Leave first], the first result is retained.
- When [Model ID] is [Not used], [Measurement] is [Not used], and [Leave First/Last] is [Leave last], the last result is retained.
- When [Model ID] is [Not used], [Measurement] is [Leave max.], and [Leave First/Last] is [Leave first in identical meas.], the result for which the specified measurement value is largest is retained. If all the measurement values are the same, the first result is retained.
- When [Model ID] is [Not used], [Measurement] is [Leave max.], and [Leave First/Last] is [Leave last in identical meas.], the result for which the specified measurement is the largest is retained. If all the measurement values are the same, the last result is retained.
- When [Model ID] is [Not used], [Measurement] is [Leave min.], and [Leave First/Last] is [Leave first in identical meas.], the result for which the specified measurement value is the smallest is retained. If all the measurement values are the same, the first result is retained.
- When [Model ID] is [Not used], [Measurement] is [Leave min.], and [Leave First/Last] is [Leave last in identical meas.], the result for which the specified measurement value is the smallest is retained. If all the measurement values are the same, the last result is retained.
- When [Model ID] is [Leave max.], [Measurement] is [Not used], and [Leave First/Last] is [Leave first when model IDs are the same], the result with the largest model ID is retained. If all the model IDs are the same, the first result is retained.
- When [Model ID] is [Leave max.], [Measurement] is [Not used], and [Leave First/Last] is [Leave last when model IDs are the same], the result with the largest model ID is retained. If all the model IDs are the same, the last result is retained.
- When [Model ID] is [Leave max.], [Measurement] is [Leave max. in identical model ID], and [Leave First/Last] is [Leave first in identical model ID and meas.], the result with the largest model ID is retained. If all the model IDs are the same, the result for which the specified measurement value is the largest is retained. If all the model IDs and measurement values are the same, the first result is retained.
- When [Model ID] is [Leave max.], [Measurement] is [Leave max. in identical model ID], and [Leave First/Last] is [Leave last in identical model ID and meas.], the result with the largest model ID is retained. If all the model IDs are the same, the result for which the specified measurement value is the largest is retained. If all the model IDs and measurement values are the same, the last result is retained.
- When [Model ID] is [Leave max.], [Measurement] is [Leave min. in identical model ID], and [Leave First/Last] is [Leave first in identical model ID and meas.], the result with the largest model ID is retained. If all the model IDs are the same, the result for which the specified measurement value is

the smallest is retained. If all the model IDs and measurement values are the same, the first result is retained.

- When [Model ID] is [Leave max.], [Measurement] is [Leave min. in identical model ID], and [Leave First/Last] is [Leave last in identical model ID and meas.], the result with the largest model ID is retained. If all the model IDs are the same, the result for which the specified measurement value is the smallest is retained. If all the model IDs and measurement values are the same, the last result is retained.
- When [Model ID] is [Leave min.], [Measurement] is [Not used], and [Leave First/Last] is [Leave first when model IDs are the same], the result with the smallest model ID is retained. If all the model IDs are the same, the first result is retained.
- When [Model ID] is [Leave min.], [Measurement] is [Not used], and [Leave First/Last] is [Leave last when model IDs are the same], the result with the smallest model ID is retained. If all the model IDs are the same, the last result is retained.
- When [Model ID] is [Leave min.], [Measurement] is [Leave max. in identical model ID], and [Leave First/Last] is [Leave first in identical model ID and meas.], the result with the smallest model ID is retained. If all the model IDs are the same, the result for which the specified measurement value is the largest is retained. If all the model IDs and measurement values are the same, the first result is retained.
- When [Model ID] is [Leave min.], [Measurement] is [Leave max. in identical model ID], and [Leave First/Last] is [Leave last in identical model ID and meas.], the result with the smallest model ID is retained. If all the model IDs are the same, the result for which the specified measurement value is largest is retained. If all the model IDs and measurement values are the same, the last result is retained.
- When [Model ID] is [Leave min.], [Measurement] is [Leave min. in identical model ID], and [Leave First/Last] is [Leave first in identical model ID and meas.], the result with the smallest model ID is retained. If all the model IDs are the same, the result for which the specified measurement value is the smallest is retained. If all the model IDs and measurement values are the same, the first result is retained.
- When [Model ID] is [Leave min.], [Measurement] is [Leave min. in identical model ID], and [Leave First/Last] is [Leave last in identical model ID and meas.], the result with the smallest model ID is retained. If all the model IDs are the same, the result for which the specified measurement value is the smallest is retained. If all the model IDs and measurement values are the same, the last result is retained.
- When [Model ID] is [Overwrite only model ID by max. value], [Measurement] is [Not used], and [Leave First/Last] is [Leave first (other than model ID)], the first result is retained. However, the model IDs are overwritten with the largest result for the same part.
- When [Model ID] is [Overwrite only model ID by min. value], [Measurement] is [Not used], and [Leave First/Last] is [Leave first (other than model ID)], the first result is retained. However, the model IDs are overwritten with the smallest result for the same part.

When the Duplicate Tolerance is not exceeded

When the value for ⊿ is less than the Duplicate Tolerance value, the vision system retains only A', the n-1th part detected, with the default setting in which [Not used] is selected for [Model ID], [Not used] for [Measurement], and [Leave first] for [Leave First/Last].



A‘: Current position of the n-1th workpeice detected

A : Current position of the nth workpeice detected

⊿ : Distance between A and A'

Fig. 6.4 (b) Example of a case when the Duplicate Tolerance is not exceeded

> **⚠ CAUTION**
> The duplicate check may result in the vision system leaving out parts other than the same part detected multiple times. If the distance between the current position of the part detected earlier and that of the part detected subsequently is less than the tolerance, the vision system judges the part detected subsequently as the same part and leaves it out, even when it has a different model ID.

> **NOTE**
> In 7DF1/14 or older software, [Duplicate Tolerance] is displayed as [Overlap Tolerance]. In addition, [Duplicate Check Mode] is not supported. In the case of 7DF1/14 or older software, the behavior of the vision system is the same as that of the default setting in which [Not used] is selected for [Model ID], [Not used] is selected for [Measurement], and [Leave first] is selected for [Leave First/Last].

[Load Balance]

Specify how parts are to be allocated on the conveyor.

[No]

Each robot handles as many parts as possible.

[Self-paced]

Parts are allocated so that each robot handles them at a specified ratio. Note that, if too many parts are supplied resulting in some of them being bypassed, there is no compensation for such bypassing. As a result, the number of parts handled by each robot may not be equal.

[Keep rate]

Parts are allocated so that each robot handles them at a specified ratio. Ultimately, the number of parts handled by each robot becomes equal.

Also, select a method of specifying a balance (the ratio of parts to be handled by each robot) unless load balancing is disabled.

[For all model IDs]

Only one set of balance data can be specified. Parts are allocated according to the specified balance, regardless of their model IDs.

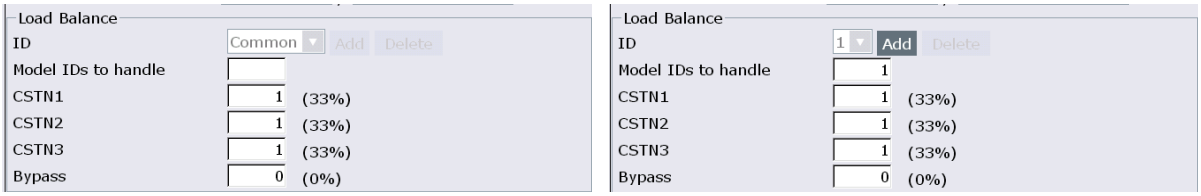
[For each model ID]

Up to eight sets of balance data can be specified. Which balance to use depends on the model ID of the part handled.

> **NOTE**
> Some practical examples of specifying load balancing based on model IDs are given in Chapter 8, "EXAMPLES OF SETUP IN ACCORDANCE WITH USE".

[Load Balance] box

The [Load Balance] box refers to the following parts in [Workcell Setup].



When load balancing is not disabled, specify the ratio of parts to be handled by the robot of each conveyor station while changing the model ID.

If you specify [For all model IDs], the screen looks as shown on the item to the left. For example, suppose that the conveyor has three conveyor stations and that you want parts to be allocated evenly. In this case, specify 1 for each conveyor station, and parts will be allocated at a ratio of 1:1:1. Note that a ratio of 1:1:1 is exactly the same as a ratio of 2:2:2.

If you specify [For each model ID], the screen looks as shown on the item to the right. Set a ratio for each model ID while selecting a model ID individually. In [Model IDs to handle], specify a model ID. To specify the same balance for two or more model IDs, you can add model IDs by clicking the [Add] button. Up to eight model IDs can be specified. An unnecessary model ID can be deleted by clicking the [Delete] button.

If you set a value other than 0 in [Bypass], as many parts as the set value are bypassed intentionally on the conveyor. Set a value other than 0 if more than one conveyor object is defined on the same physical conveyor and a certain number of parts need to be supplied to another conveyor object.

[Pre-group] box

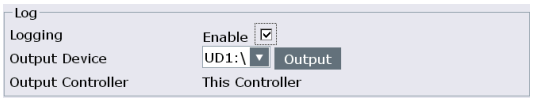
If you enable the pre-grouping function which arrange parts on the same conveyor, you set items in this box. For detail, see Section 6.14, "SETUP OF PRE-GROUPING".

[Pre-grouping]

If you enable the pre-grouping function, check the [Used] checkbox and you can see the specific items to be set. For detail, see Section 6.14, "SETUP OF PRE-GROUPING".

[Log] box

If you enable the logging, the part information put to the queue is recorded in the temporary memory. And you can output it to the specified device on runtime. This log can be utilized in order to play back the part flow in ROBOGUIDE.This play back function can be used with ROBOGUIDE in V8 Rev.L or later.



[Logging]

If you enable the logging, please check the [Enable] checkbox. From the moment of check, the logging is started. The maximum number of the logged part is 1000. If it is over 1000, the data is deleted from the oldest one.

[Output Device]

In this device, the recorded logs are output. You can select “UD1” or “MC”. When you push the [Output] button, the current all logs recorded are output. You can push this button during robot’s running.

[Output Controller]

The controller where the logs are output is displayed. They are output in the most upstream on-line robot in this conveyor.

6.4.1 Setting a Tracking Frame

Set a tracking frame and encoder scale.

You can set a tracking frame and encoder scale by using the tracking schedule setting screen of the teach pendant of the robot. However, the procedure described below is recommended particularly if multiple robots are used for tracking, because a tracking frame common to all the robots needs to be set.

> ⚠ **CAUTION**
> If you set a tracking frame for a 3-, 4- or 5-axis robot by using the tracking schedule setting screen of the teach pendant, a value other than 0 may be set as the value of W or P of the tracking frame. By contrast, if you set a tracking frame by using the procedure described below, 0 is always set as the values of W and P of the tracking frame for a 3-, 4- or 5-axis robot. Since a 3-, 4- or 5-axis robot can only move with its flange face facing downward (or upward) during tracking, set a tracking frame according to the procedure described below.
> On the other hand, this procedure does not support tracking where the value of Z of the world frame changes for a 3-, 4- or 5-axis robot. If such tracking motion is involved, you need to use the tracking schedule setting screen of the teach pendant to set a tracking frame. Note that, even in this case, the flange face needs to keep facing downward (or upward) during tracking and it is therefore necessary to fine-adjust the frame by directly entering the frame components.

When setting a tracking frame, have a fixture carried on the conveyor. Any fixture may be used if it has a part that the robot can touch up accurately with the TCP. If you have purchased a calibration grid from FANUC, a dot on the calibration grid can be used as well.
The following description assumes the use of a calibration grid.

⚠ CAUTION

1 If you change the tracking frame, you need to perform conveyor station setup, camera calibration, reference position setting, and robot position teaching again.
2 When there is more than one robot, you cannot set a tracking frame unless the encoder count values of all the controllers on the conveyor are identical. If they are not identical, turn the power of all the robot controllers off then on again.
3 When there is more than one robot, make sure that the fixture does not move on the conveyor belt. If the fixture moves because it comes into contact with the robot or other object while it is carried on the conveyor, you need to redo the tracking frame setting from the beginning.
4 The tracking frame affects the ultimate handling accuracy. Touch up the fixture using an accurately set TCP.
5 You cannot set a tracking frame correctly if the robot wrist posture (W, P and R) changes during the setting process. Jog the robot only in the X, Y and Z directions so that the robot wrist posture does not change.

The setting procedure is different depending on tracking type.
For line conveyor setup, refer to Chapter 3, "*i*RPICKTOOL BASIC (BASIC FUNCTIONS) STARTUP PROCEDURES." If there are multiple robots on one conveyor, setup of the second robot's tracking frame will start when the setup for the first robot's tracking frame is finished.

If you want to redo the tracking frame setup for only a specific conveyor station for a conveyor for which the tracking frame has been set up, refer to Subsection 6.6.1, "Adding a Tracking Frame."

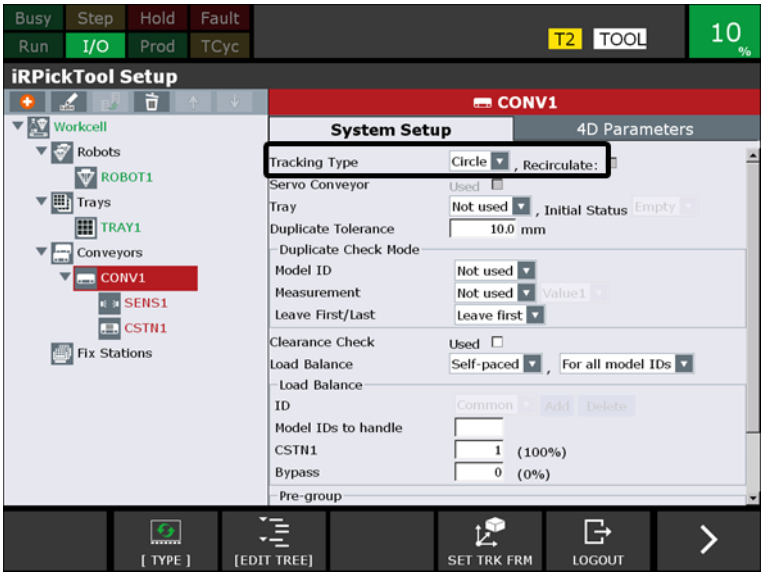
6.4.1.1 Circular conveyor

The procedure for when using a circular conveyor is as follows.

Setup of circular tracking

The procedure to set up *i*RPickTool for when using circular tracking is explained below. Be careful of the following points as remarks for when using circular tracking.

Be sure to select [Circle] for [Tracking Type] before proceeding to "Setup of a circular conveyor" given later.



Z-axis direction of the tracking frame

In circular tracking, the Z-axis direction of the tracking frame can become downward.

- When the conveyor flow direction is counterclockwise, the Z-axis direction of the tracking frame is upward.

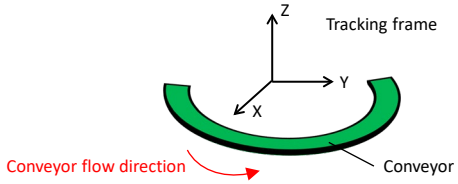


Fig. 6.4.1.1 (a) Conveyor: Counterclockwise, Z-axis: Upward

- When the conveyor flow direction is clockwise, the Z-axis direction of the tracking frame is downward.

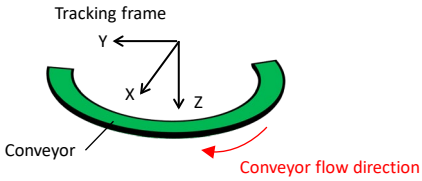


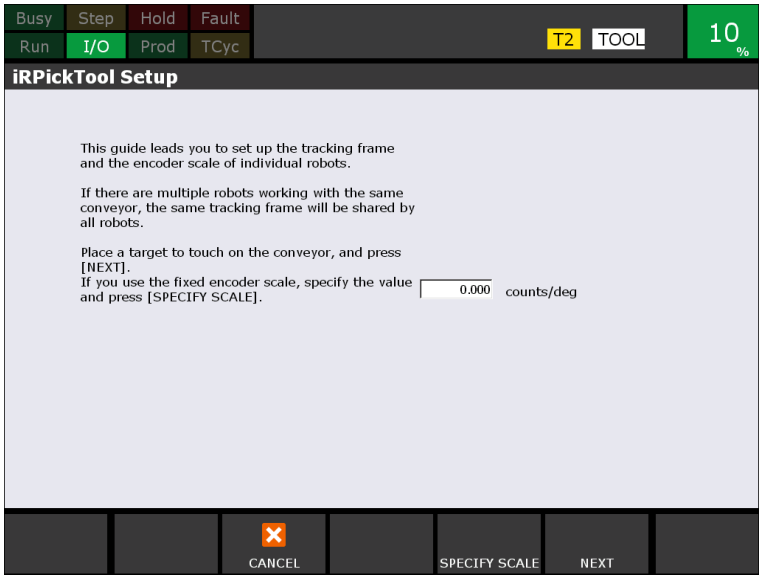
Fig. 6.4.1.1 (b) Conveyor: Clockwise, Z-axis: Downward

The Z-axis direction of the tray frame is the same as that of the tracking frame. Check the Z-axis direction of the tracking frame before setting X, Y, Z and R of tray. Refer to Section 6.3 “SETUP OF A TRAY”. In addition, check the Z-axis direction of the tracking frame before setting [Part Z Height] in [Reference Data]. Refer to Subsection 3.2.7.2 “Vision process”.

> ⚠ **CAUTION**
> When setting a tracking frame, be sure to be as low as possible the speed of the conveyor.

Setup of a circular conveyor

1 Add as many conveyor stations as the number of robots that will operate on the conveyor. In the setup screen for each conveyor station, select a robot, tracking schedule number, and encoder number. For details, see Section 6.6, “SETUP OF A CONVEYOR STATION”.
2 In the conveyor setup screen, press F4 [SET TRK FRM] key. A screen as shown below is displayed.



3 Place a fixture that the robot can touch up with the TCP in the most upstream area of the conveyor, and press F5 [NEXT] key. If you intend to use the value calculated from the actual measurement, instead of having the encoder scale calculated automatically, enter the scale value in the text box and press F4 [SPECIFY SCALE] key. In either case, a screen as shown below is displayed.

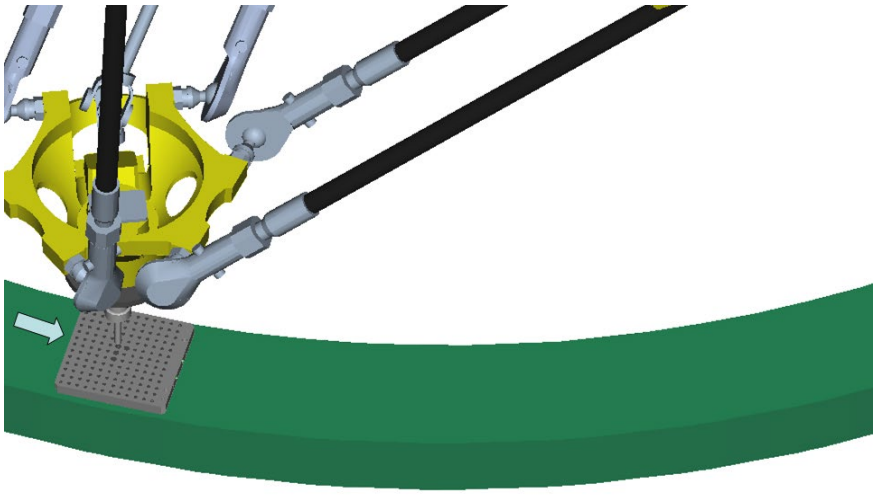
If you use the calibration grid of *i*RVision, you can set the direction of the big circle of the calibration grid in any direction because you touch up the same point three times later in this setup procedure.

⚠ CAUTION
Be sure to touch up as much as possible the outer circumference. This is necessary to improve the tracking accuracy.

4 As the fixture feed frame, select the user tool frame that you set in Section 5.4, "Setting Up the Tool Frame".

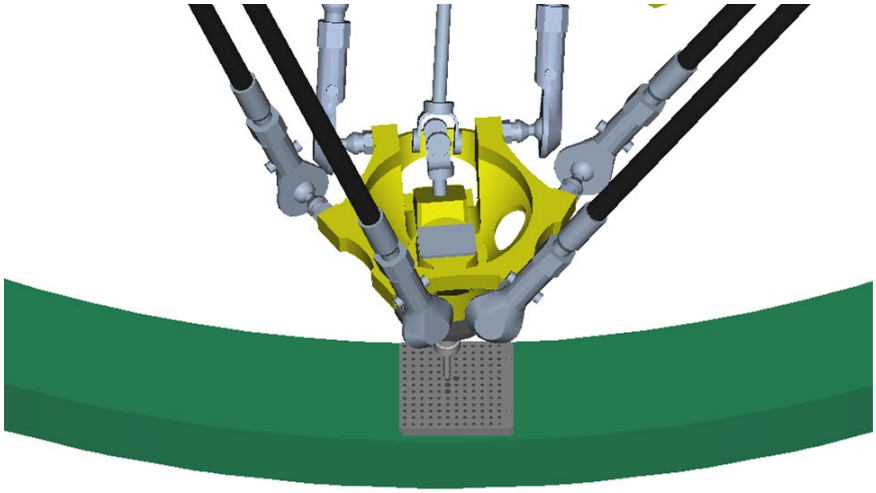
⚠ CAUTION
Select the user tool frame with the robot that touches up the fixture. It is not necessarily the robot controller for which the *i*RPickTool setup screen is open.

5 Move the conveyor until the fixture reaches the upstream area as much as possible within the robot operation range, and the stop it.
6 Move the robot by jog operation, and touch up the fixture with the TCP.
Here, the dot at the intersection of the large dots in an L shape on the calibration plate is touched up.



7 With the fixture touched up, press F5 [NEXT] key. A screen as shown below is displayed.

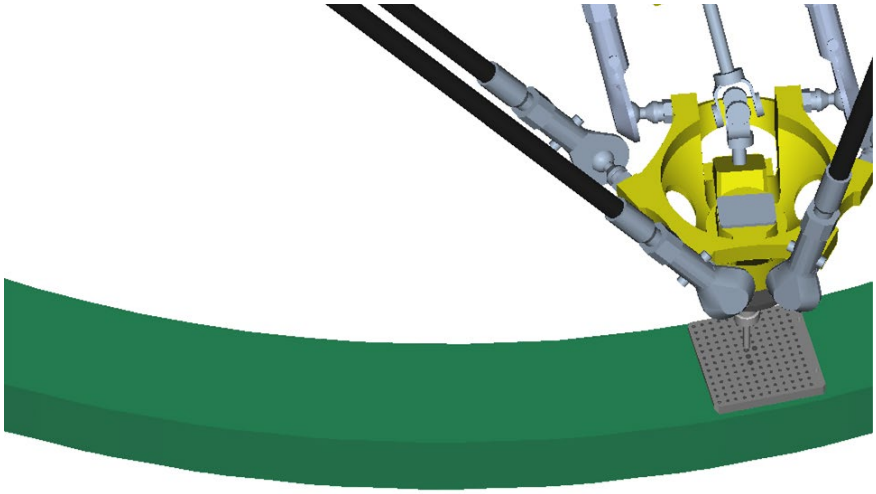
8 After moving the robot to a position where the TCP does not interfere when the conveyor is moved, move the conveyor so that the fixture comes to the middle area within the robot operation range and then stop it.
9 Then, jog the robot again to touch up the same part of the fixture that you touched up with the TCP earlier.



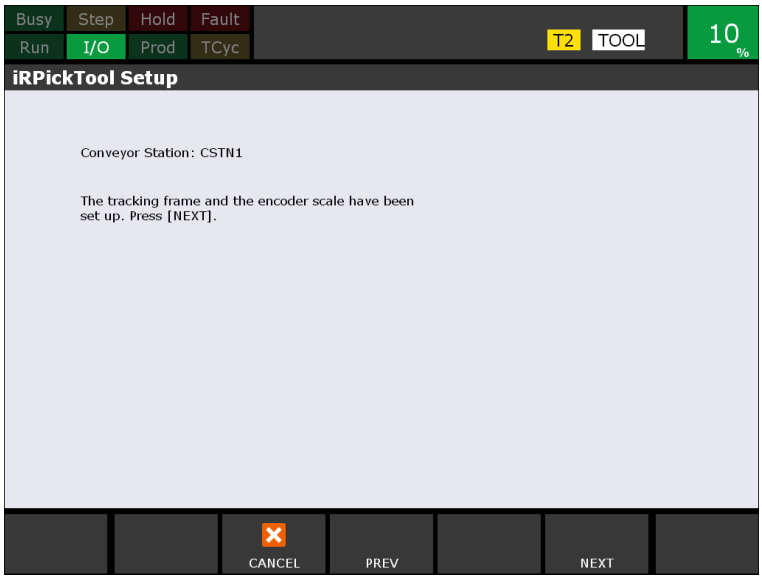
10 With the fixture touched up, press F5 [NEXT] key. A screen as shown below is displayed.

11 After moving the robot to a position where the TCP does not interfere when the conveyor is moved, move the conveyor so that the fixture comes to the downstream area as much as possible within the robot operation range and then stop it.

12 Then, jog the robot again to touch up the same part of the fixture that you touched up with the TCP earlier.

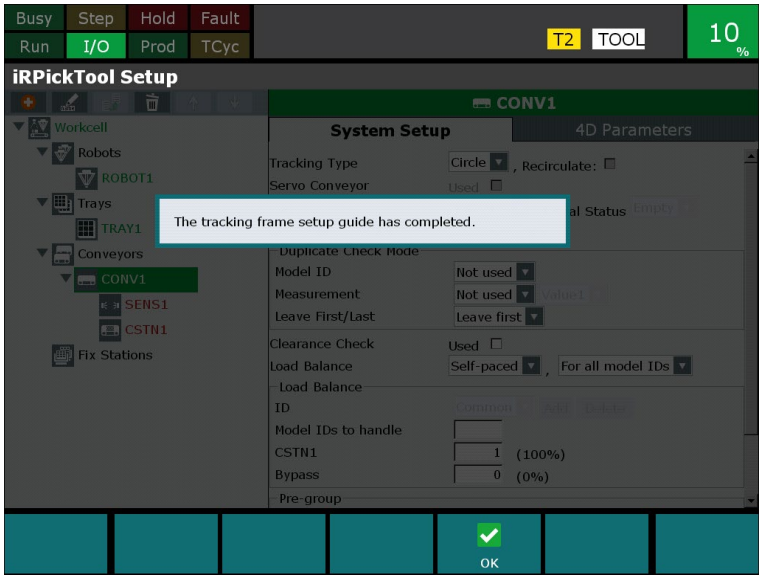


13 With the fixture touched up, press F5 [NEXT] key. A screen as shown below is displayed.



14 This completes the tracking frame setting procedure for the first robot, and a tracking frame and scale have now been set. If you have more than one robot for the same conveyor, you can start the tracking frame setting procedure for the second robot by pressing F5 [NEXT] key.

15 Repeat steps 4 through 12 for each robot. When you have completed the procedure for all robots, a screen as shown below is displayed.



16 Press F4 [OK] key to finish the tracking frame setting procedure.

NOTE
If you want to redo the touch-up operation for the robot, you can return to the previous setup page by pressing F3 [PREV] key.

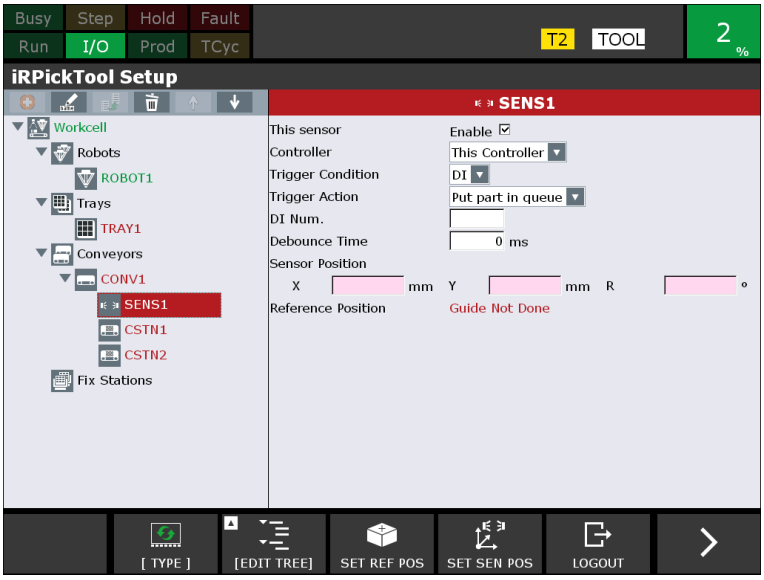
6.5 SETUP OF A SENSOR

Set up a sensor.

⚠ CAUTION
1 You cannot set up a sensor unless you have completed the tracking frame setting procedure in the conveyor setup process.
2 The setting for sensor still remain though you finish setup here. The sensor object is trained after setting reference position (“6.10 REFERENCE POSITION SETTING”).

6.5.1 Setting a Sensor in Detail

If you select [SENS1] in the tree view, a screen like the one below will appear.



[This sensor]

Specify whether to use the sensor selected on the screen. To use this sensor, check the [Enable] box.

[Controller]

If the robot ring is set, select the name of the controller to which the sensor is connected. If the robot ring is not set (i.e., there is only one controller), only [This Controller] is available.

> **NOTE**
> The robot for the controller selected in [Controller] must be selected at any one of the conveyor stations on the same conveyor.

[Trigger Condition]

Select the trigger condition from the following:

[Distance]

The action is triggered each time the conveyor travels a specified distance.

[DI]

The action is triggered when the rising edge of the DI signal (state change from off to on) is detected.

[HDI]

The action is triggered when the input of the HDI signal is detected. This is a more precise trigger condition than DI.

[RI]

The action is triggered when the input of the RI signal is detected.

[FLAG]

The action is triggered when the flag turns ON. This is automatically selected when the servo conveyor is used.

[Trigger Action]

Select the action to be taken when the trigger condition is met from the following:

[Put part in queue]

A part is put into the queue.
The following restrictions are applied:

- The model ID is fixed as 0.
- [For each model ID] cannot be selected in [Load Balance] for a conveyor.

[Find part by vision]

The vision system searches for a part and puts a detected part into the queue. This action is not available when the trigger condition is [HDI] or [FLAG].

[Run program]

The specified TP program is executed.
The following restrictions are applied:

- When the Trigger Condition is HDI or FLAG, this action cannot be selected.
- This action can be used only on a linear conveyor.
- This action cannot be used on a conveyor using a tray.
- This action cannot be used for an infeed conveyor in pre-grouping.
- This action can be used only on an external machine vision interface add-on. For details, refer to "I. EXTERNAL MACHINE VISION INTERFACE ADD-ON".

NOTE
In 7DF1/16 or older software, [Run program] cannot be used.
In addition, [Run program] cannot be used in ROBOGUIDE.

The possible combinations of a trigger condition and an action are shown below.

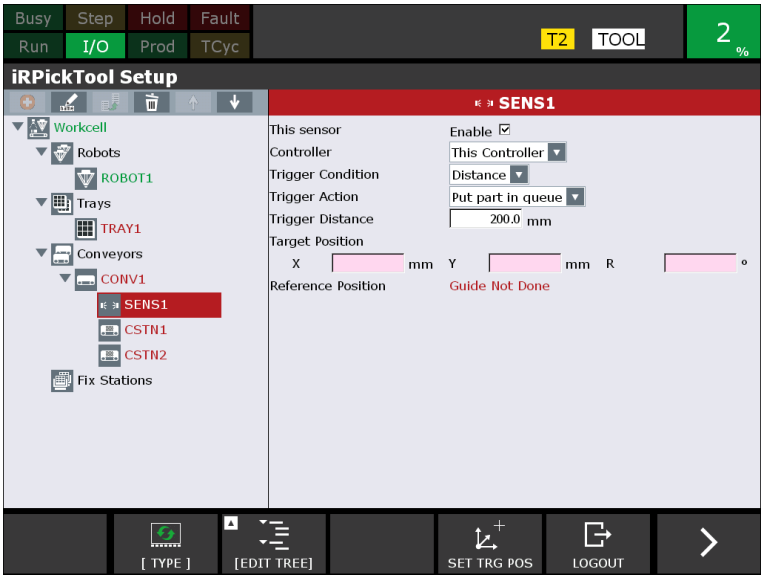
Table 6.5.1 Possible combinations of trigger condition and action

	Distance	DI	HDI	RI	FLAG
Put part in queue	○	○	○	○	○
Find part by vision	○	○	×	○	×
Run program	○	○	×	○	×

For the setup screen that will be displayed depending on the combination of the trigger condition and action, refer to Subsection 6.5.1.1, "When [Distance] and [Put part in queue] are selected" to Subsection 6.5.1.11, "When [RI] and [Run program] are selected."

6.5.1.1 When [Distance] and [Put part in queue] are selected

A part is put into the queue each time the conveyor travels a specified distance.
In order to place parts at regular intervals on the conveyor, it has formerly been necessary to attach strips of tape at regular intervals and detect those strips of tape. The use of this combination enables parts to be placed at regular intervals without the need to attach strips of tape.
Since the sensor task can put a part into the queue without having to actually detect the part, this combination is also useful when you want to simulate the pickup motion.



[Trigger Distance]

Specify the distance in mm that the conveyor is to travel before a part is put. When the tracking type is "Circle", you specify the angle in degree.

[Target Position]

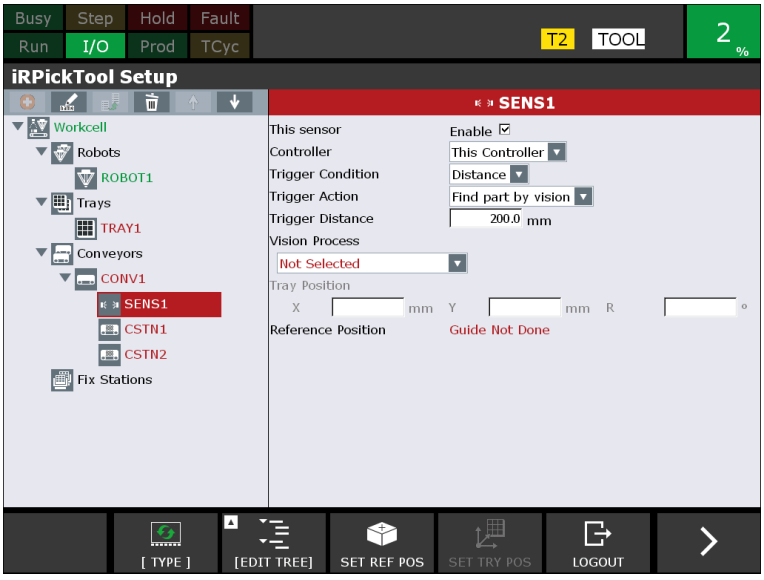
This is the part position that is generated each time the conveyor travels the trigger distance. Set these values using the target position setting guide described later.

[Reference Position]

When you finish the guide to set the target postion, "Guide Done" is displayed in green.

6.5.1.2 When [Distance] and [Find part by vision] are selected

A vision process is executed each time the conveyor travels a specified distance.



[Trigger Distance]

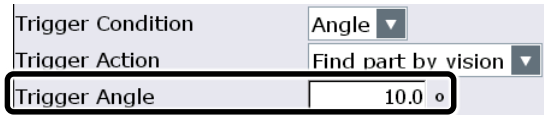
Specify the distance in mm that the conveyor is to travel before a vision process is executed.

It is recommended that the trigger distance be set such that a workpiece can be detected twice in the camera FOV.

> **⚠ CAUTION**
> If you set too short [Trigger Distance] and the vision detection is started so often before the previous vision detection completes, the alarm of "INTP-302 Stack overflow" may occur. In this case, you should set longer [Trigger Distance] so that the vision detection can complete before the next starts. With more concretely, refer the [Action Time] in the subsection 9.2.2 "Checking the Status of a Sensor".

For circular conveyors

For circular conveyors, specify the angle (units: degrees) for the item [Trigger Distance] in the case of [Distance] + [Find part by vision].



[Vision Process]

Enter the name of the vision process to be executed.

[Tray Position]

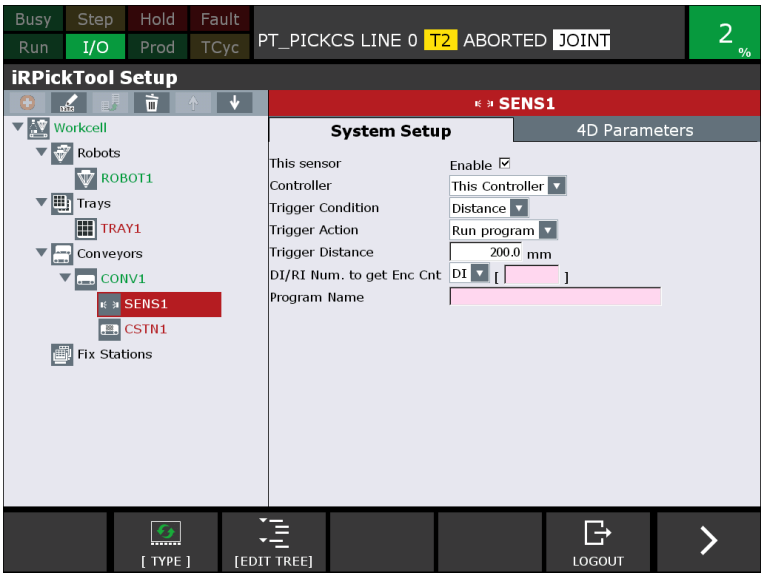
This is the relative position of the tray detection position and the origin of the tray. Set these values using the tray position setting guide described later.

[Reference Position]

When you finish the guide to set the reference position, "Guide Done" is displayed in green.

6.5.1.3 When [Distance] and [Run program] are selected

The specified TP program is executed each time the conveyor travels the specified distance.



[Trigger Distance]

Specify the distance in mm that the conveyor is to travel before executing the TP program.

[DI/RI Num. to get Enc Cnt]

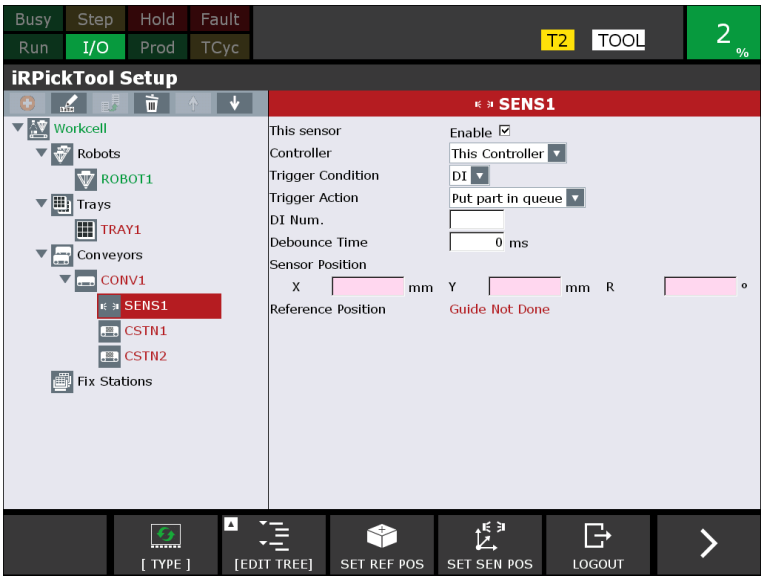
This is the number of the input signal used to inform the controller of the image-capturing timing for the external machine vision. The controller obtains the encoder count when this signal is input. You cannot specify DI/RI numbers to get encoder counts that have already been specified for other sensors.

[Program Name]

Specify the name of the TP program to be executed. Programs using motion groups cannot be used. You cannot specify program names that have already been specified for other sensors.

6.5.1.4 When [DI] and [Put part in queue] are selected

A part is put into the queue when the rising edge of the DI signal (state change from off to on) is detected.



[DI Num]

Specify the number of the digital input signal to be detected.

> **⚠ CAUTION**
> This number is different from [Trigger INPUT number] displayed on the tracking schedule setting screen of the robot's teach pendant.

[Debounce Time]

When the DI is detected with shorter interval than this [Debounce Time] after the previous DI was detected, this detection is ignored as a wrong input or detection. By specifying proper value, you can prevent the duplicated detection of a same part caused by the debounce input of an external sensor.

[Sensor Position]

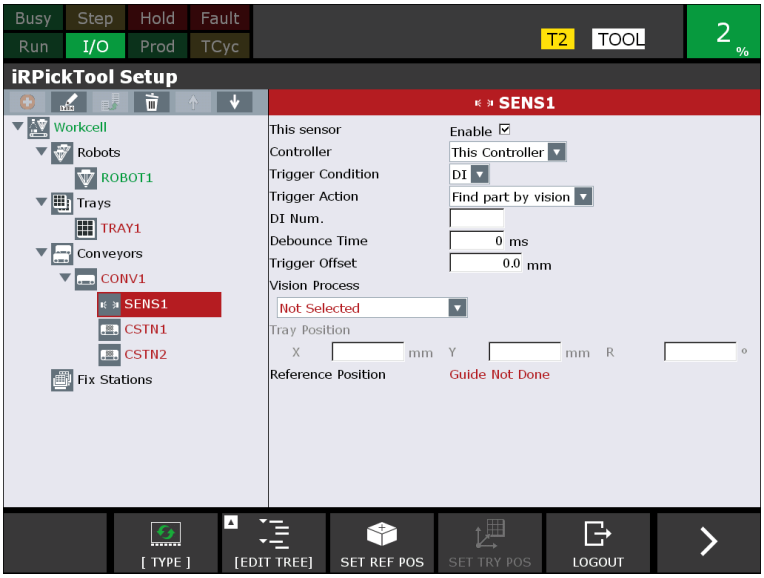
When a part is detected, the position specified here is used as the detection position as the part is put into the queue. Set these values using the sensor position setting guide described later.

[Reference Position]

When you finish the guide to set the reference position, "Guide Done" is displayed in green.

6.5.1.5 When [DI] and [Find part by vision] are selected

When the rising edge of the DI signal (state change from off to on) is detected, a vision process is executed and the detected part is put into the queue.



[DI Num.]

Specify the number of the digital input signal to be detected.

> **⚠ CAUTION**
> This number is different from [Trigger INPUT number] displayed on the tracking schedule setting screen of the robot's teach pendant.

[Debounce Time]

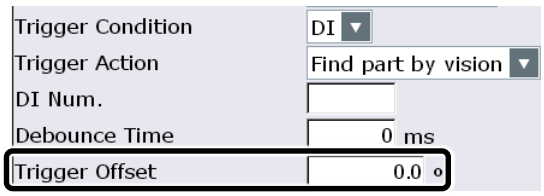
When the DI is detected with shorter interval than this [Debounce Time] after the previous DI was detected, this detection is ignored as a wrong input or detection. By specifying proper value, you can prevent load of CPU from increasing by multiple detections caused by the debounce input of an external sensor.

[Trigger Offset]

A vision process is executed if the conveyor travels the distance specified here after the rising edge of the DI is detected. Use this offset in such cases as when the camera is installed apart from the photo eye.

For circular conveyors

For circular conveyors, specify the angle (units: degrees) for the item [Trigger Offset] in the case of [DI]/[RI] + [Find part by vision].



[Vision Process]

Specify the name of the vision process to be executed.

[Tray Position]

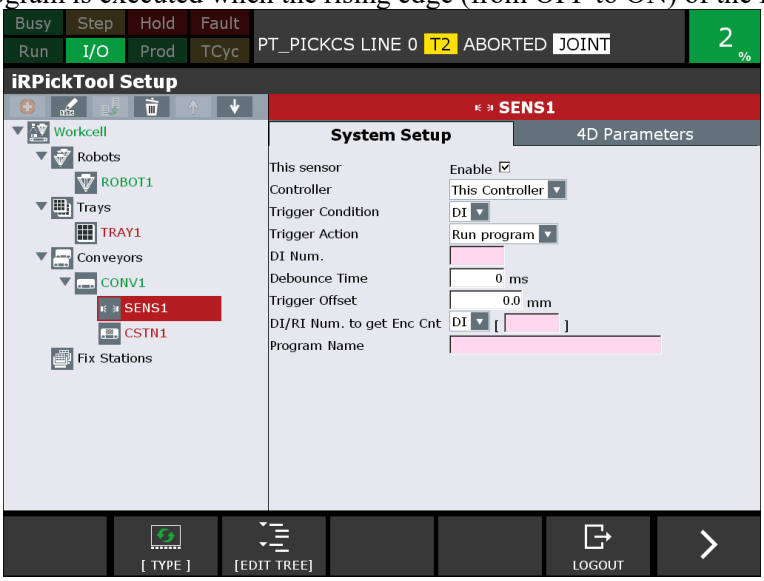
This is the relative position of the tray detection position and the origin of the tray. Set these values using the tray position setting guide described later.

[Reference Position]

When you finish the guide to set the reference position, “Guide Done” is displayed in green.

6.5.1.6 When [DI] and [Run program] are selected

The specified TP program is executed when the rising edge (from OFF to ON) of the DI signal is detected.



[DI Num.]

Specify the number of the digital input signal to be detected.

> **⚠ CAUTION**
> This number is different from [Trigger INPUT number] displayed on the tracking schedule setting screen of the robot’s teach pendant.

[Debounce Time]

When the DI is detected with shorter interval than this [Debounce Time] after the previous DI was detected, this detection is ignored as a wrong input or detection. By specifying proper value, you can prevent load of CPU from increasing by multiple detections caused by the debounce input of an external sensor.

[Trigger Offset]

The TP program is executed when the conveyor travels the distance specified here after the rising edge of the DI signal is detected.

[DI/RI Num. to get Enc Cnt]

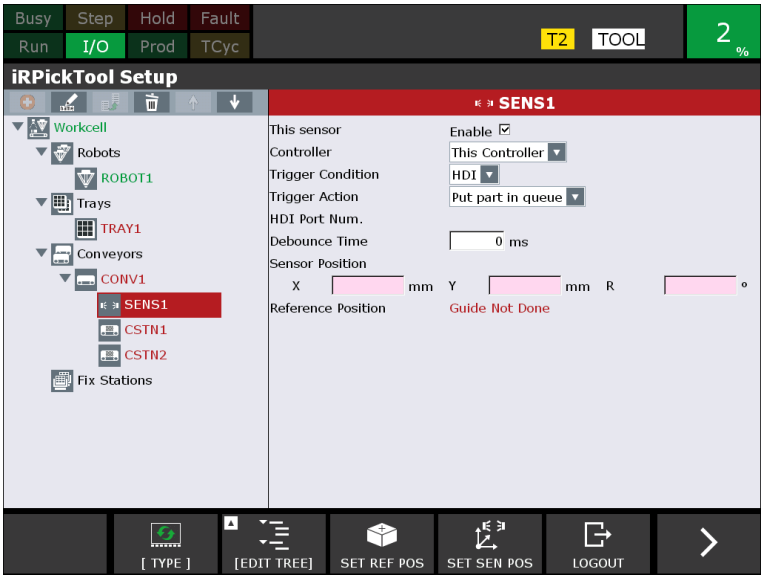
This is the number of the input signal used to inform the controller of the image-capturing timing for the external machine vision. The controller obtains the encoder count when this signal is input. You cannot specify the same input signal as that specified for DI Num. Also, you cannot specify DI/RI numbers to get encoder counts that have already been specified for other sensors.

[Program Name]

Specify the name off the TP program to be executed. Program using motion groups cannot be used. You cannot specify program names that have already been specified for other sensors.

6.5.1.7 When [HDI] and [Put part in queue] are selected

When the input of the HDI signal is detected, a part is put into the queue



[HDI Port Num.]

The port number of the HDI signal to be detected is displayed. The displayed port number is the one that is set for the schedule of the encoder number specified for the conveyor station.

[Debounce Time]

When the HDI is detected with shorter interval than this [Debounce Time] after the previous HDI was detected, this detection is ignored as a wrong input or detection. By specifying proper value, you can prevent load of CPU from increasing by multiple detections caused by the debounce input of an external sensor.

[Sensor Position]

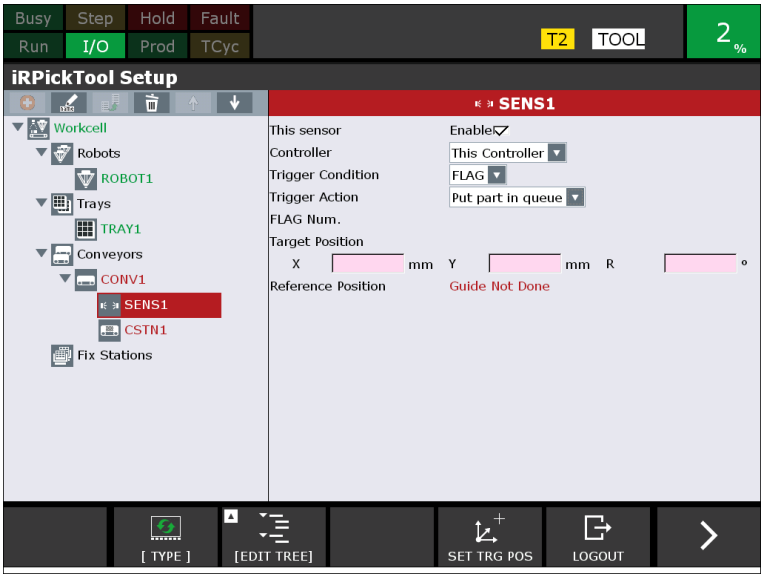
When a part is detected, the position specified here is used as the detection position as the part is put into the queue. Set these values using the sensor position setting guide described later.

[Reference Position]

When you finish the guide to set the reference position, “Guide Done” is displayed in green.

6.5.1.8 When [FLAG] and [Put part in queue] are selected

This is selected automatically only if a servo conveyor is used.
When the flag turns ON, a part is put into the queue. The flag is input automatically every time the servo conveyor moves by one pitch.



[FLAG Num.]

*i*RPickTool uses this dedicated FLAG number that is specified in the servo conveyor setup menu. You cannot change this number in this menu.

[Target Position]

This is the part position that is generated when the flag turns ON. Set these values using the target position setting guide described later.

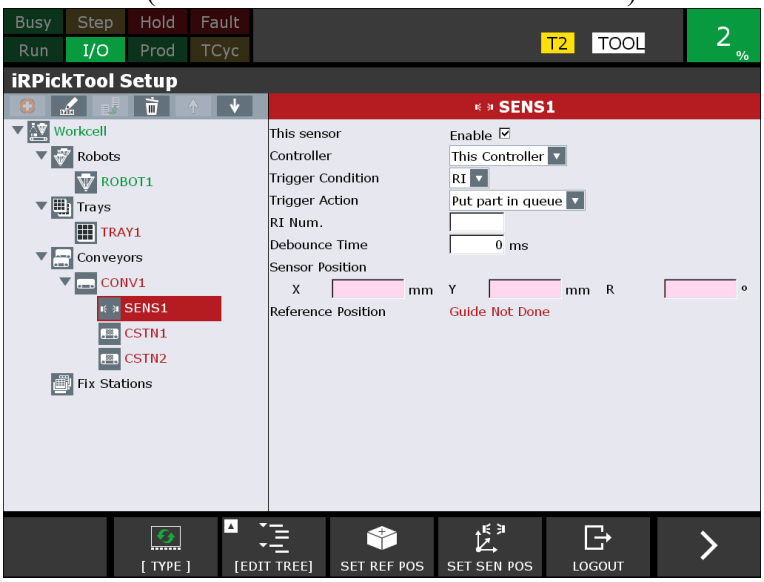
[Reference Position]

When you finish the guide to set the target position, “Guide Done” is displayed in green.

6.5.1.9 When [RI] and [Put part in queue] are selected

Detects the input of the RI signal and puts one part in the queue.

The procedure to start up the combination [RI] + [Put part in queue] is the same as that for [DI] + [Put part in queue].
For details, refer to Chapter 3 “*i*RPICKTOOL BASIC (BASIC FUNCTIONS) STARTUP PROCEDURES” or Chapter 4, “ *i*RPICKTOOL (BASIC FUNCTIONS + PLUG & PLAY) STARTUP PROCEDURES.”



[RI Num.]

Specify the number of the digital input signal to be detected.

> **⚠ CAUTION**
> This number is different from [Trigger INPUT number] displayed on the tracking schedule setting screen of the robot’s teach pendant.

[Debounce Time]

When the RI is detected with shorter interval than this [Debounce Time] after the previous RI was detected, this detection is ignored as a wrong input or detection. By specifying proper value, you can prevent the duplicated detection of a same part caused by the debounce input of an external sensor.

[Sensor Position]

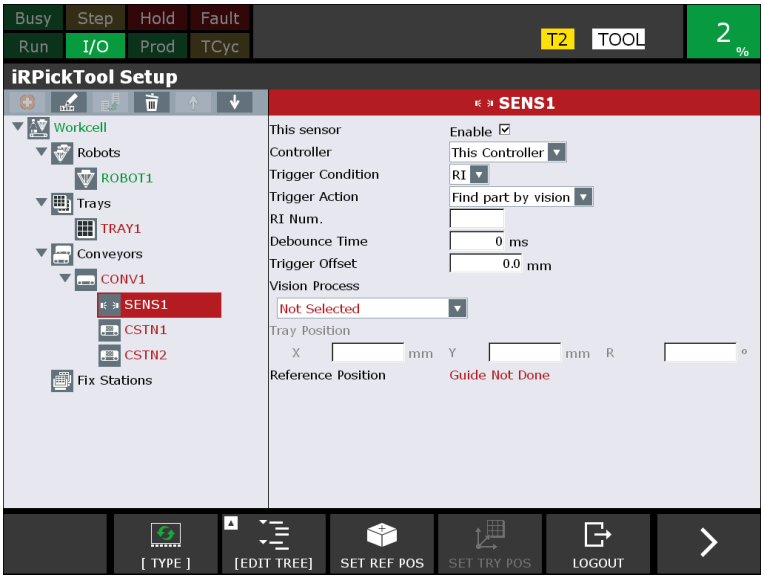
When a part is detected, the position specified here is used as the detection position as the part is put into the queue. Set these values using the sensor position setting guide described later.

[Reference Position]

When you finish the guide to set the reference position, “Guide Done” is displayed in green.

6.5.1.10 When [RI] and [Find part by vision] are selected

When the rising edge of the RI signal (state change from off to on) is detected, a vision process is executed and the detected part is put into the queue.



[RI Num.]

Specify the number of the digital input signal to be detected.

> **⚠ CAUTION**
> This number is different from [Trigger INPUT number] displayed on the tracking schedule setting screen of the robot's teach pendant.

[Debounce Time]

When the RI is detected with shorter interval than this [Debounce Time] after the previous RI was detected, this detection is ignored as a wrong input or detection. By specifying proper value, you can prevent load of CPU from increasing by multiple detections caused by the debounce input of an external sensor.

[Trigger Offset]

A vision process is executed if the conveyor travels the distance specified here after the rising edge of the RI is detected. Use this offset in such cases as when the camera is installed apart from the photo eye.

For circular conveyors

For circular conveyors, specify the angle (units: degrees) for the item [Trigger Offset] in the case of [DI]/[RI] + [Find part by vision].

[Vision Process]

Specify the name of the vision process to be executed.

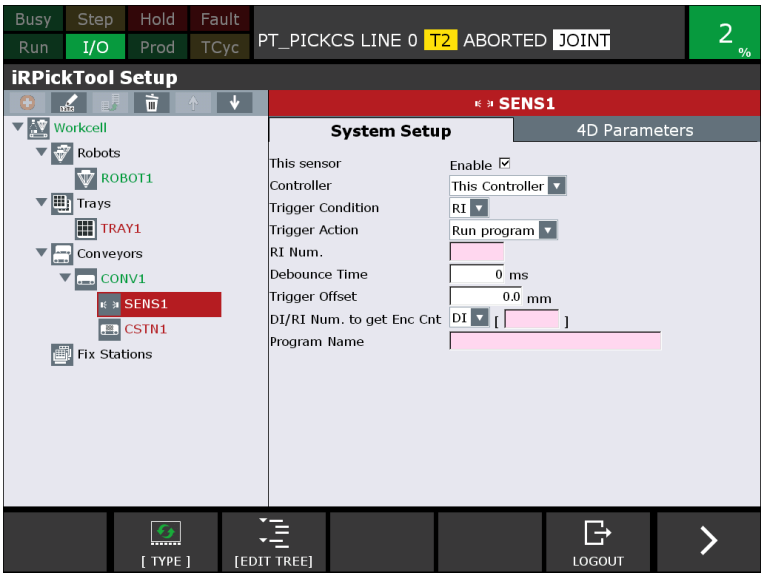
[Tray Position]

This is the relative position of the tray detection position and the origin of the tray. Set these values using the tray position setting guide described later.

[Reference Position]
When you finish the guide to set the reference position, “Guide Done” is displayed in green.

6.5.1.11 When [RI] and [Run program] are selected

The specified TP program is executed when an input of the RI signal is detected.



[RI Num.]
Specify the number of the digital input signal to be detected.

> **⚠ CAUTION**
> This number is different from [Trigger INPUT number] displayed on the tracking schedule setting screen of the robot’s teach pendant.

[Debounce Time]
When the RI is detected with shorter interval than this [Debounce Time] after the previous RI was detected, this detection is ignored as a wrong input or detection. By specifying proper value, you can prevent load of CPU from increasing by multiple detections caused by the debounce input of an external sensor.

[Trigger Offset]
The TP program is executed when the conveyor travels the distance specified here after the rising edge of the DI signal is detected.

[DI/RI Num. to get Enc Cnt]
This is the number of the input signal used to inform the controller of the image-capturing timing for the external machine vision. The controller obtains the encoder count when this signal is input. You cannot specify the same input signal as that specified for RI Num. Also, you cannot specify DI/RI numbers to get encoder counts that have been already specified for other sensors.

[Program Name]
Specify the name of the TP program to be executed. Program using motion groups cannot be used. You cannot specify program names that have already been specified for other sensors.

6.5.2 Settings Required for Each Combination of a Trigger Condition and a Trigger Action

The setup required will vary depending on the combination of trigger condition and action.

Table 6.5.2 Setup required for each combination of trigger condition and action

[Trigger Condition]	[Trigger Action]	Tray used / not used	6.5.3 Setting the Sensor Position	6.5.4 Setting Up the Vision System	6.5.5 Setting the Tray Position	6.5.6 Setting the Target Position
[Distance]	[Put part in queue]	Used	×	×	×	○
		Not used	×	×	×	○
	[Find part by vision]	Used	×	○	○	×
		Not used*1	×	○	×	×
	[Run program]	Not used*2	×	×	×	×
[DI]	[Put part in queue]	Used*1	○	×	×	×
		Not used	○	×	×	×
	[Find part by vision]	Used	×	○	○	×
		Not used	×	○	×	×
	[Run program]	Not used*2	×	×	×	×
[HDI]	[Put part in queue]	Used*1	○	×	×	×
		Not used	○	×	×	×
[FLAG]	[Put part in queue]	Used	×	×	×	○
		Not used	×	×	×	○
[RI]	[Put part in queue]	Used*1	○	×	×	×
		Not used	○	×	×	×
	[Find part by vision]	Used	×	○	○	×
		Not used	×	○	×	×
	[Run program]	Not used*2	×	×	×	×

*1 For the setup and startup procedure, refer to Chapter 3, "*i*RPICKTOOL BASIC (BASIC FUNCTIONS) STARTUP PROCEDURES" or Chapter 4, "*i*RPICKTOOL (BASIC FUNCTIONS + PLUG & PLAY) STARTUP PROCEDURES."

*2 When [Run program] is specified as the action, these settings are configured in the TP program, not on the *i*RPickTool Setup screen. For details, refer to "I. EXTERNAL MACHINE VISION INTERFACE ADD-ON".

6.5.3 Setting the Sensor Position

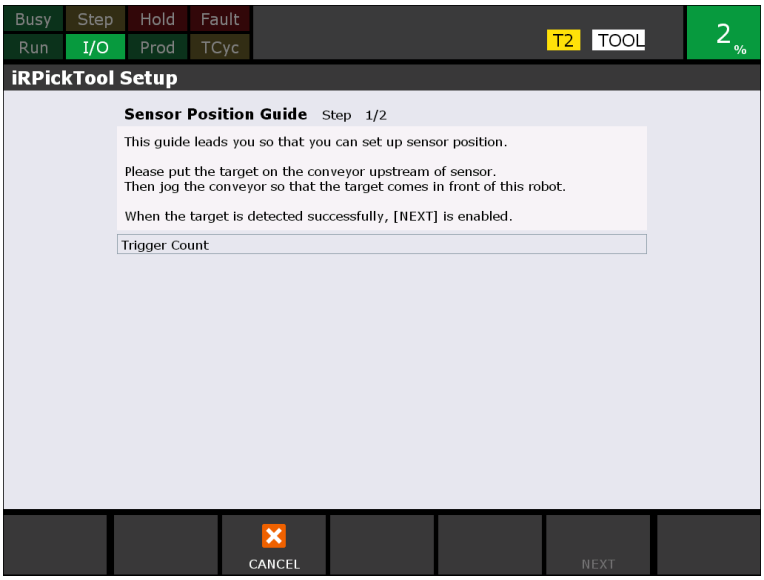
Teach the position where the photo eye is installed.
This setup is necessary when any of [DI]/[HDI]/[RI] is selected for the trigger condition and the action is "Put part in queue".
When a part crosses the photoelectric tube sensor, this position is loaded to the queue. The procedures differ slightly between the cases when a tray is used and not used on a conveyor.

NOTE
To set the sensor position, you need to open the iRPickTool Setup screen with the controller selected for this sensor.

For the procedure to set the sensor position for when using a tray, refer to Chapter 3, "*i*RPICKTOOL BASIC (BASIC FUNCTIONS) STARTUP PROCEDURES" or Chapter 4, "*i*RPICKTOOL (BASIC FUNCTIONS + PLUG & PLAY) STARTUP PROCEDURES."

6.5.3.1 When a tray is not used

1 Stop the conveyor.
2 Press F4 [SET SEN POS] key. A screen as shown below is displayed.



3 Place a part upstream of the phototube sensor, and move the conveyor until the part is in front of the robot.

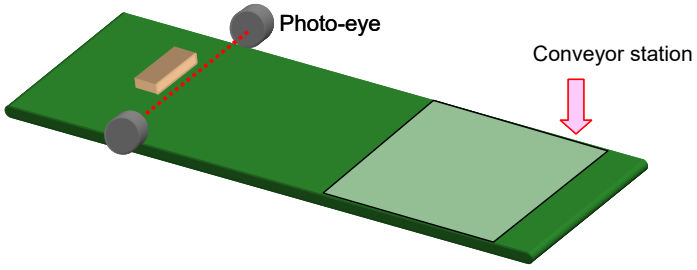
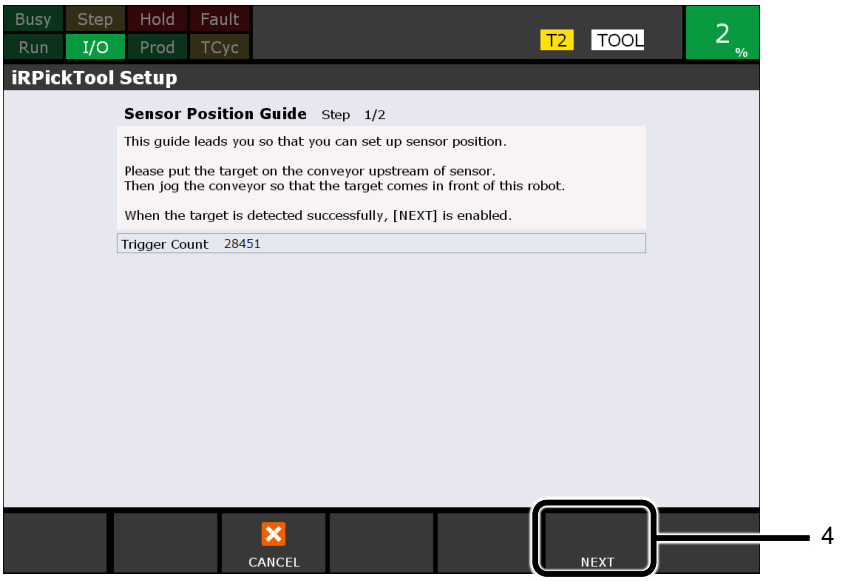


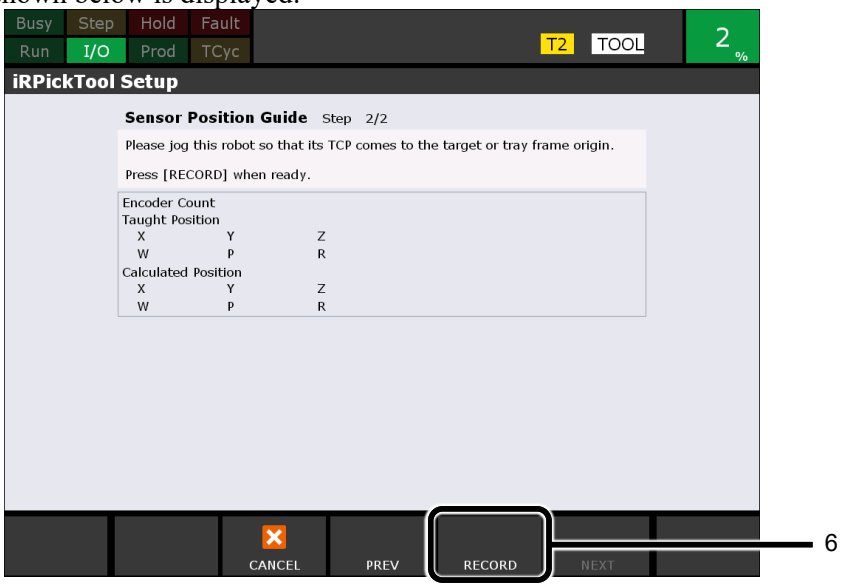
Fig.6.5.3.1 (a)

When the sensor detects the part, the encoder value obtained at the time of detection is displayed and F5 [NEXT] key is enabled.

4 Press F5 [NEXT] key.



A screen as shown below is displayed.



5 Move the conveyor until the part comes in front of the robot.
Jog the robot, and touch up the downstream-side end of the part (the part detected by the photo eye) with the TCP.

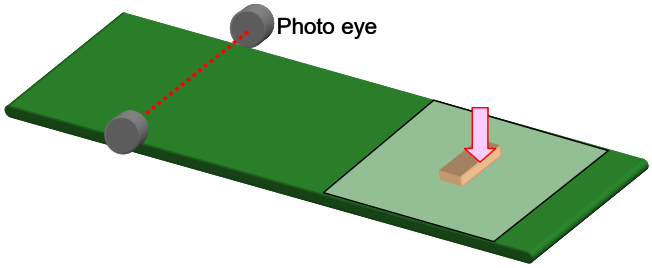
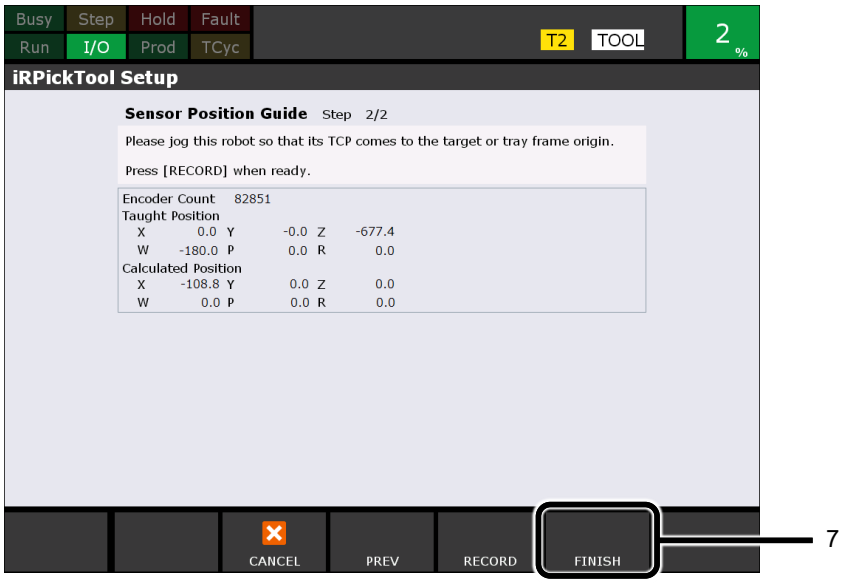


Fig.6.5.3.1 (b)

6 Press F4 [RECORD] key.
The encoder value obtained at the time of touch-up, the touch-up position, and the calculated position are displayed, and F5 [FINISH] key is enabled.
7 Press F5 [FINISH] key.



The initial setup screen is displayed again. The calculated value is input for X of the sensor position (the values of Y and R remain unchanged from 0).

6.5.4 Setting up the Vision System

Set up and calibrate the camera to be used for part detection, and teach the vision process.
This setting is required when [Find part by vision] is selected for [Trigger Action].

> ⚠ **CAUTION**
> To set up the vision system, you need to use a controller to which a camera is connected.

For the procedure to set up vision using one 2D camera, refer to Chapter 3, "*i*RPICKTOOL BASIC (BASIC FUNCTIONS) STARTUP PROCEDURES" or Chapter 4, "*i*RPICKTOOL (BASIC FUNCTIONS + PLUG & PLAY) STARTUP PROCEDURES".

NOTE

Check the Z-axis direction of the tracking frame before setting [Part Z Height] in [Reference Data].

- When the conveyor flow direction is counterclockwise, the Z-axis direction of the tracking frame is upward.

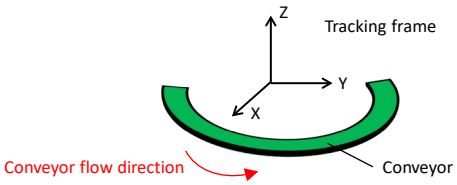


Fig. 6.5.4 (a) Z-axis when the conveyor moves counterclockwise

- When the conveyor flow direction is clockwise, the Z-axis direction of the tracking frame is downward.

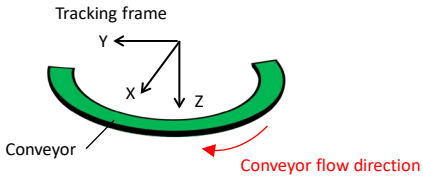


Fig. 6.5.4 (b) Z-axis when the conveyor moves clockwise

For the procedure to set up vision using one 3DV sensor, refer to Section 9.9, “3D VISION SENSOR”.
For the procedure to set up vision when using multiple 2D cameras (multiview), refer to Section 9.8, “MULTIVIEW”.

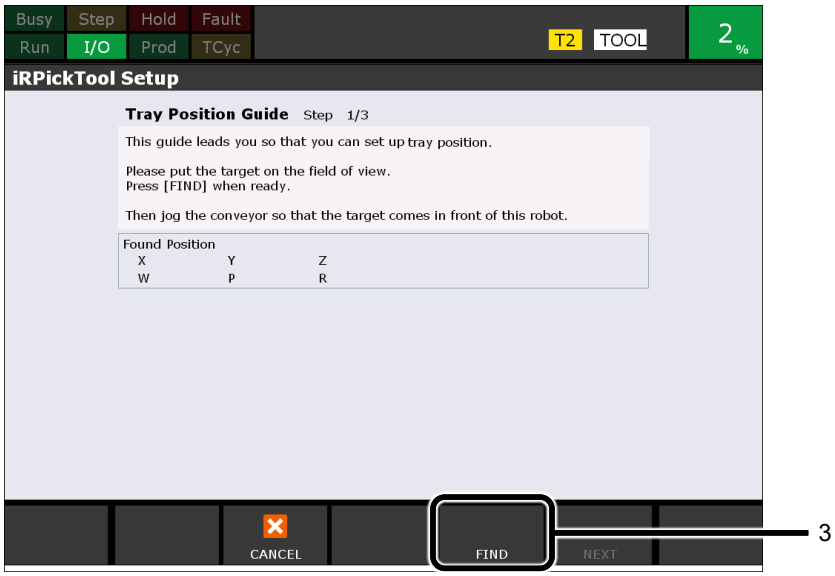
6.5.5 Setting the Tray Position

If [Find part by vision] is selected for [Trigger Action] and a tray is used, set the relative position of the origin and detection position of the tray. To do this requires that camera calibration and vision process teaching be completed in advance.

> **NOTE**
> To set the tray position, you need to open the *i*RPickTool Setup screen with the controller selected for this sensor.

6.5.5.1 When [Distance] and [Find part by vision] are selected

1 Stop the conveyor.
2 Press F4 [SET TRY POS] key. A screen as shown below is displayed.



3 Place a tray in the camera's field of view, and press F4 [FIND] key.

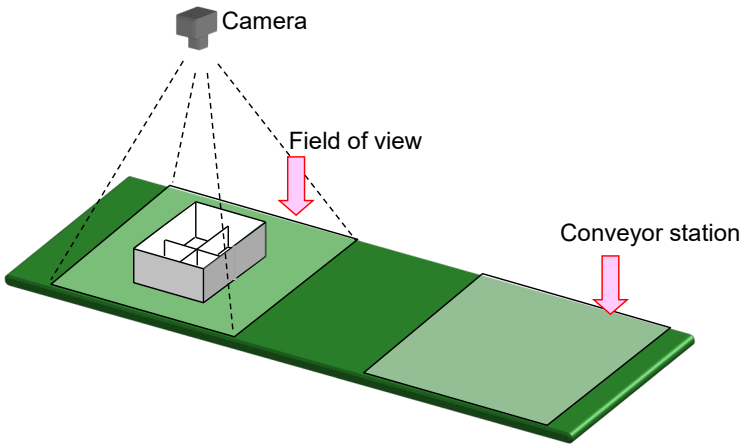
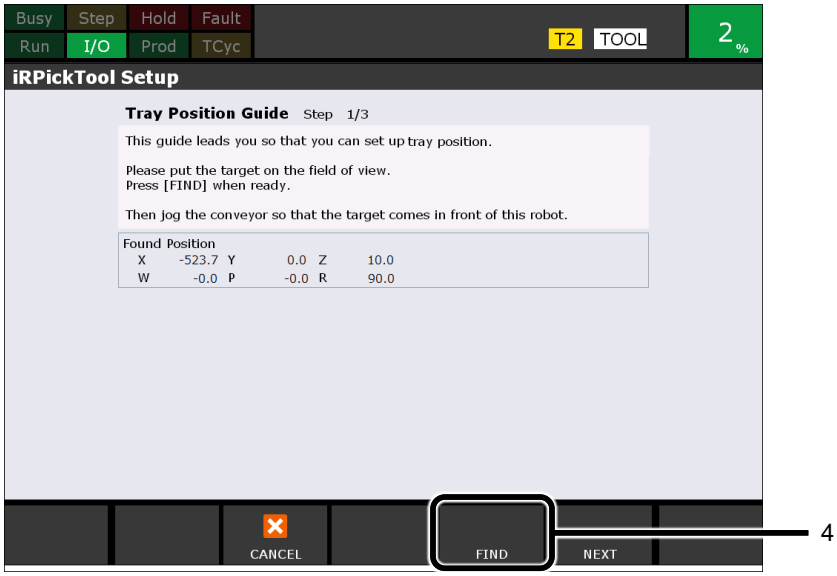
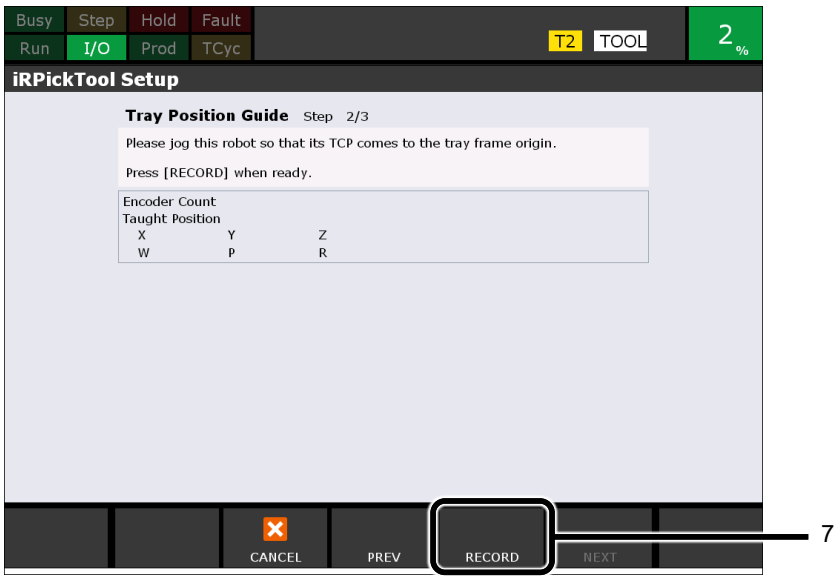


Fig.6.5.5.1 (a)

When the tray is successfully detected, the detection position is displayed and F5 [NEXT] key is enabled.
4 Press F5 [NEXT] key



A screen as shown below is displayed.



5 Move the conveyor until the tray comes in front of the robot.
6 Jog the robot, and touch up the origin of the tray frame.

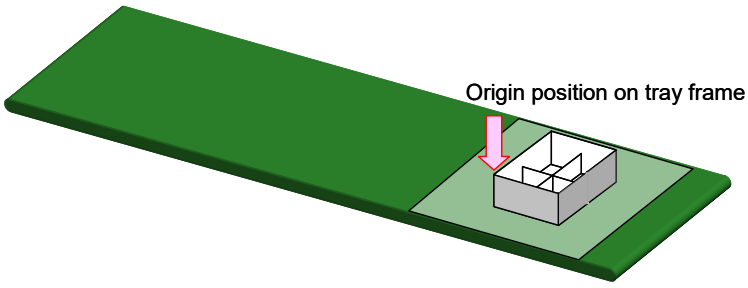


Fig.6.5.5.1 (b)

7 Press F4 [RECORD] key.
The encoder value obtained at the time of touch-up and the touch-up position are displayed, and F5 [NEXT] key is enabled.

8 If you press F5 [NEXT] key.

A screen as shown below is displayed.

9 Jog the robot, and touch up the X direction point of the tray frame.

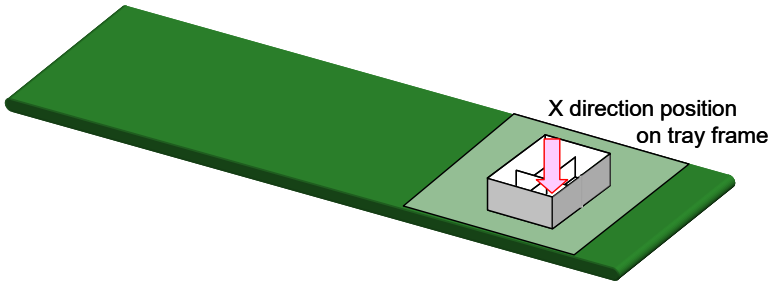
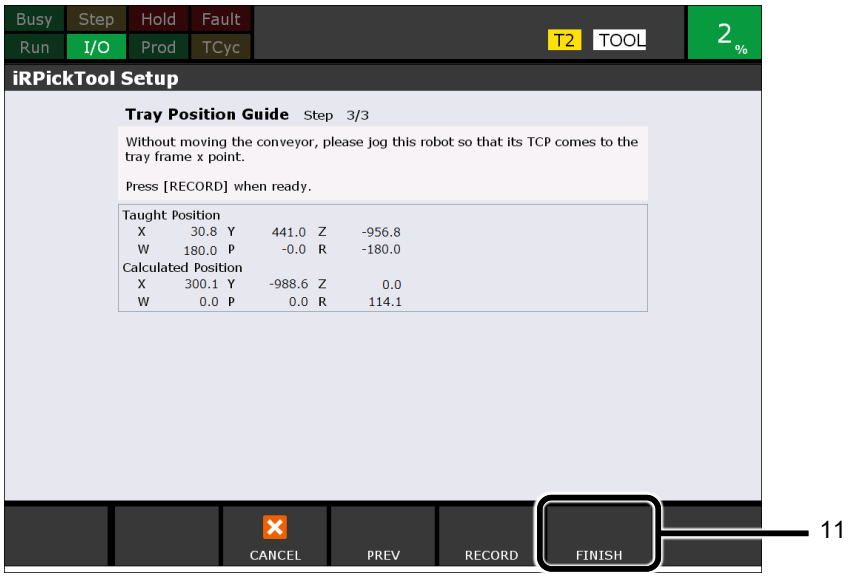


Fig.6.5.5.1 (c)

10 Press F4 [RECORD] key.
The touch-up position and the calculated position are displayed, and F5 [FINISH] key is enabled.

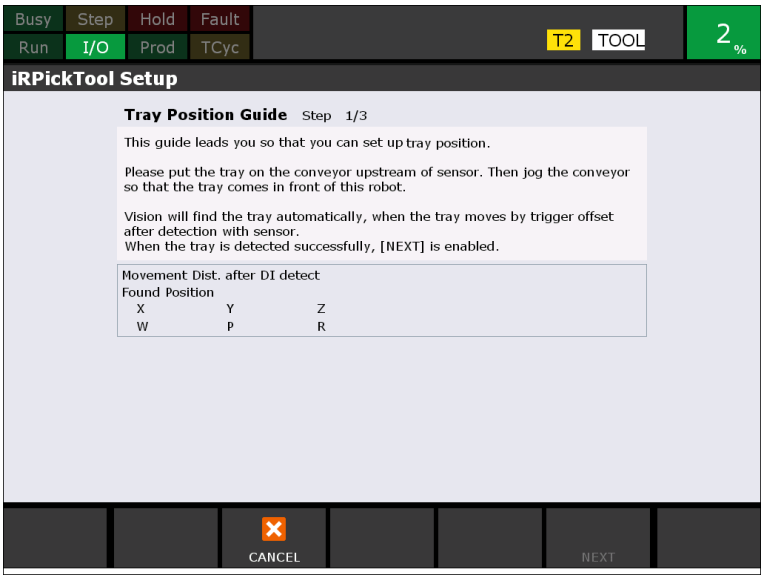
11 Press F5 [FINISH] key.



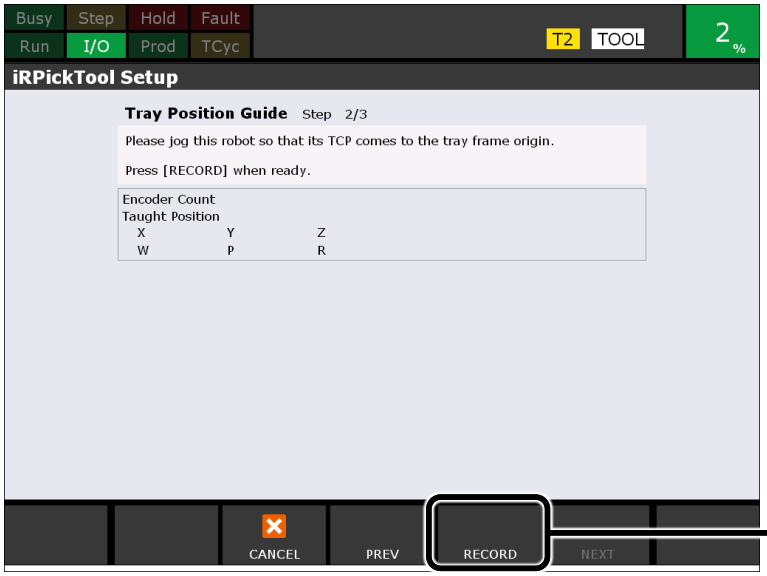
The initial setup screen is displayed again.
Values are input for X, Y, and R of the tray position.

6.5.5.2 When [DI] and [Find part by vision] are selected

1 Stop the conveyor.
2 Press F4 [SET TRY POS] key. A screen as shown below is displayed.



3 Place a tray on the most upstream section of the conveyor, and operate the conveyor until the tray is in front of the robot. When the conveyor has moved by the trigger offset after the photoelectric tube sensor has detected the tray, *i*RVision will detect the part automatically. You do not have to stop the conveyor when *i*RVision detects the tray. When *i*RVision successfully detects the part, the detection position are displayed and F5 [NEXT] key is enabled.
4 Press F5 [NEXT] key.
A screen as shown below is displayed.



5 Jog the robot, and touch up the origin of the tray frame.

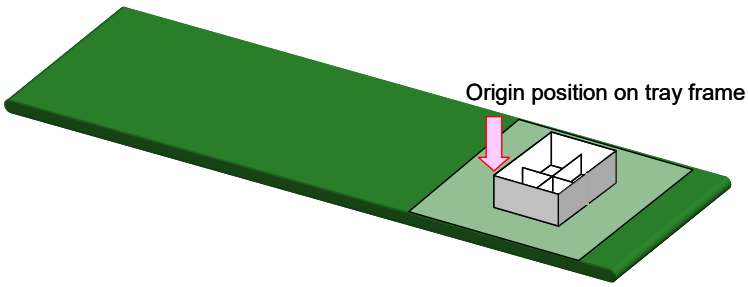


Fig.6.5.5.2 (a)

6 Press F4 [RECORD] key. The encoder value obtained at the time of touch-up and the touch-up position are displayed, and F5 [NEXT] key is enabled.

7 Press F5 [NEXT] key.

A screen as shown below is displayed.

8 Jog the robot, and touch up the X direction point of the tray frame.

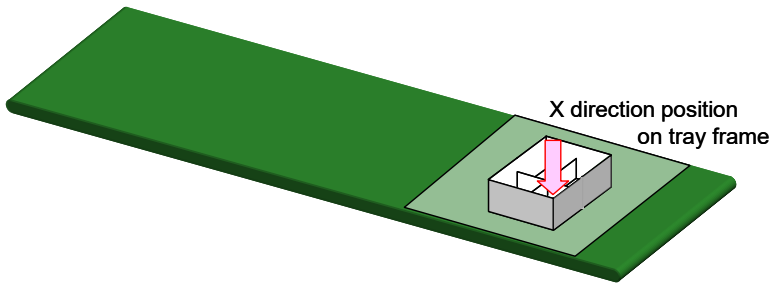


Fig.6.5.5.2 (b)

9 Press F4 [RECORD] key. The touch-up position and the calculated position are displayed, and F5 [FINISH] key is enabled.

10 If you press F5 [FINISH] key.

The initial setup screen is displayed again. Values are input for X, Y, and R of the tray position.

6.5.6 Setting the Target Position

This setting is required only when [Distance] and [Put part in queue] or [Flag] and [Put part in queue] are the combination of a trigger condition and an action. Teach the position where a part is to be generated. This position is put into the queue each time the conveyor moves a certain distance or the [Index Advance Trigger DI] of the servo conveyor is input. Since a part is not actually detected when this combination is selected, perform robot position teaching at the same time you set the target position. The procedure slightly differs depending on whether a tray is used on the conveyor or not.

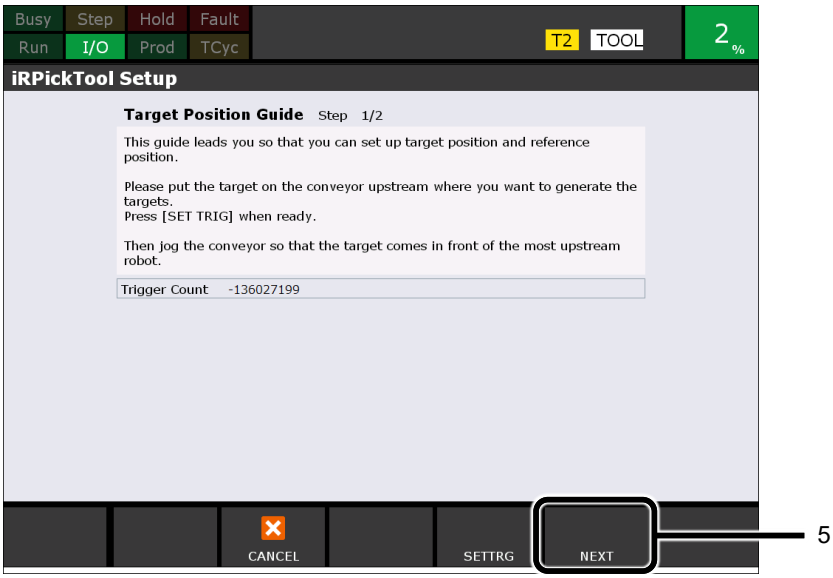
NOTE
To set the target position, you need to open the *i*RPickTool Setup screen with the controller selected for this sensor.

6.5.6.1 When a tray is not used

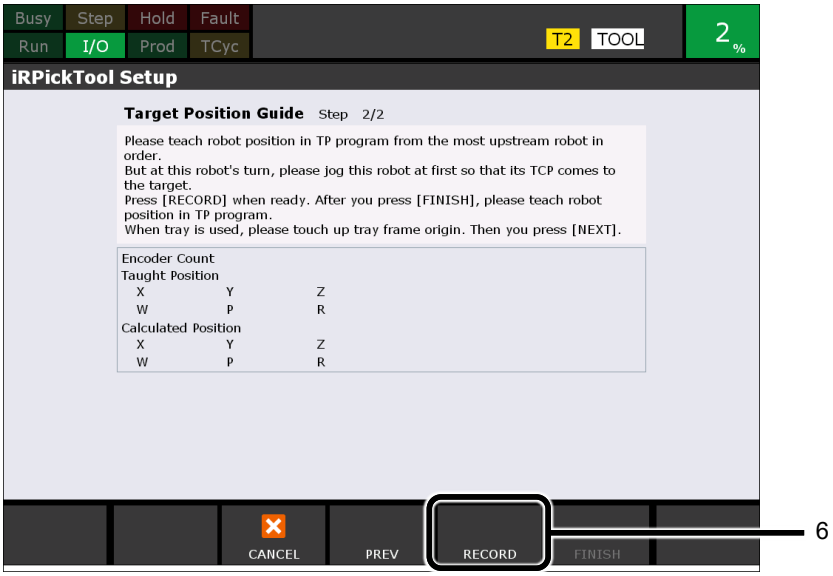
1 Stop the conveyor.
2 Press F4 [SET TRG POS] key. A screen as shown below is displayed.

3 Place a part at the position where you want to generate it. In order to perform robot position teaching for all robots, the part needs to be placed upstream of the most upstream robot operation range.
4 Press F4 [SETTRG] key. The current encoder value is displayed, and F5 [NEXT] key is enabled.

5 If you press F5 [NEXT] key.



A screen as shown below is displayed.



6 Move the conveyor, and perform robot position teaching sequentially beginning with the most upstream robot. (See Section 4.11, "TEACHING THE POSITION TO ROBOTS".)
Note that, when the part comes in front of this robot, keep the robot at the taught position and then press F4 [RECORD] key.

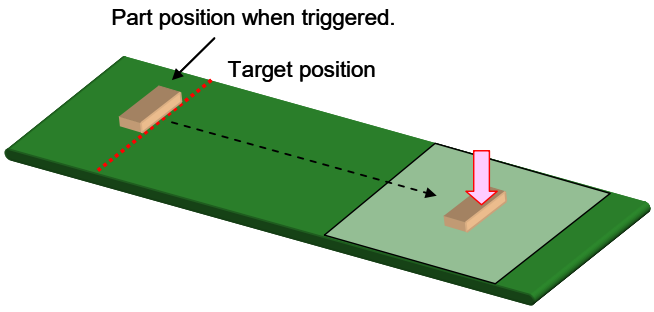


Fig.6.5.6.1(a)

The encoder value obtained at the time of touch-up, the touch-up position, and the calculated position are displayed, and F5 [FINISH] key is enabled.

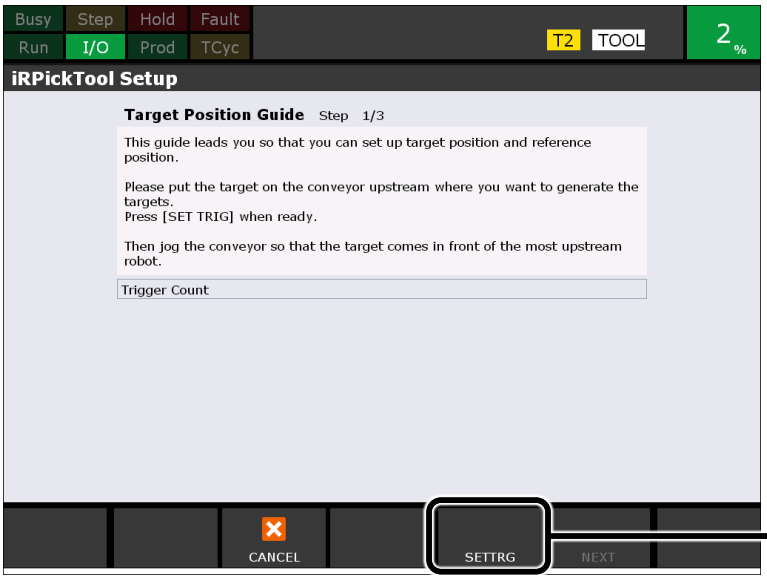
7 If you press F5 [FINISH] key.

The initial setup screen is displayed again.
The calculated value is input for X of the target position (the values of Y and R remain unchanged from 0).

8 Teach the position to this robot.
9 en, move the conveyor further to teach the position to the downstream robots.

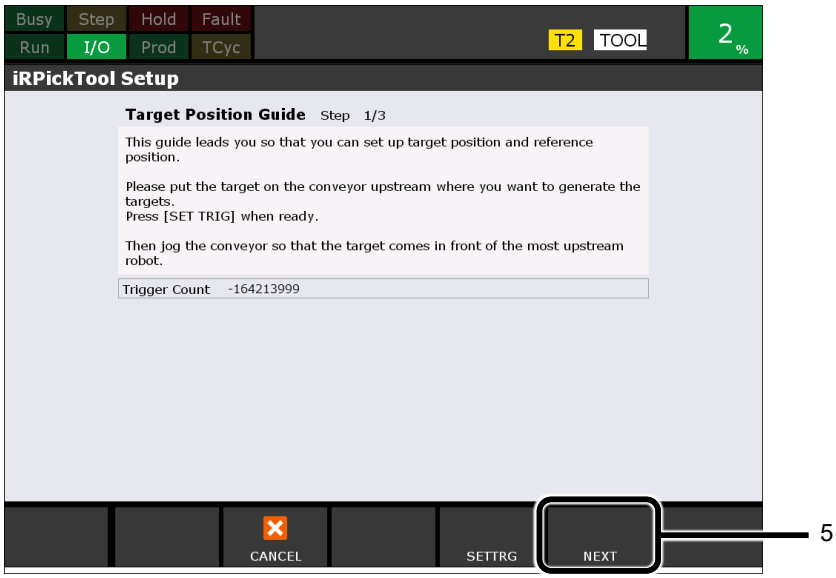
6.5.6.2 When a tray is used

1 Stop the conveyor.
2 Press F4 [SET TRG POS] key. A screen as shown below is displayed.

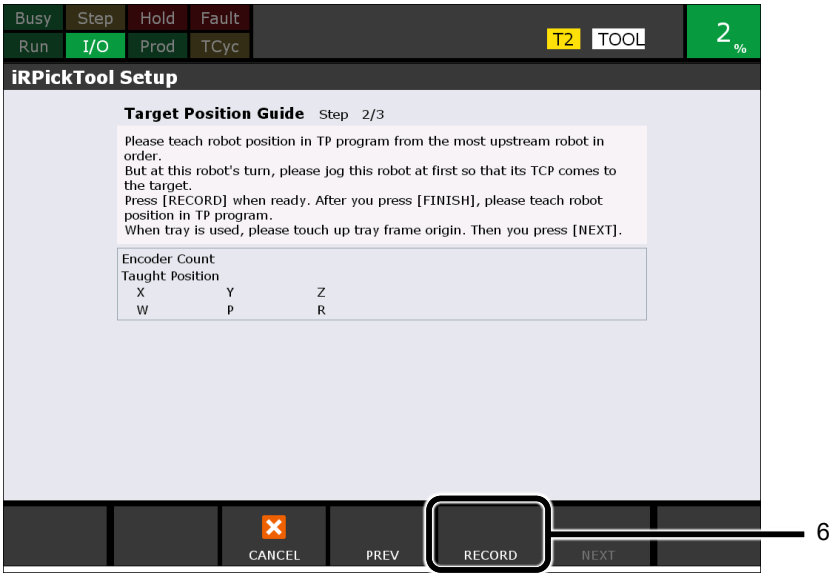


3 Place a part at the position where you want to generate it. In order to perform robot position teaching for all robots, the part needs to be placed upstream of the most upstream robot operation range.

4 Press F4 [SETTRG] key. The current encoder value is displayed, and F5 [NEXT] key is enabled.
5 If you press F5 [NEXT] key.



A screen as shown below is displayed.



6 Move the conveyor, and perform robot position teaching sequentially beginning with the most upstream robot. (See Section 6.11, "TEACHING THE POSITION TO ROBOTS".)
Note that, when the part comes in front of this robot, first jog the robot and touch up the origin of the tray frame. Then, press F4 [RECORD] key.

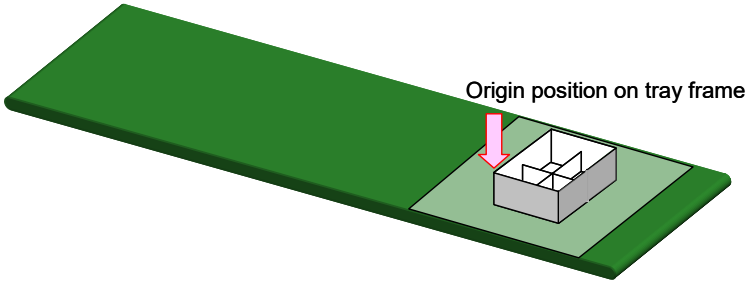
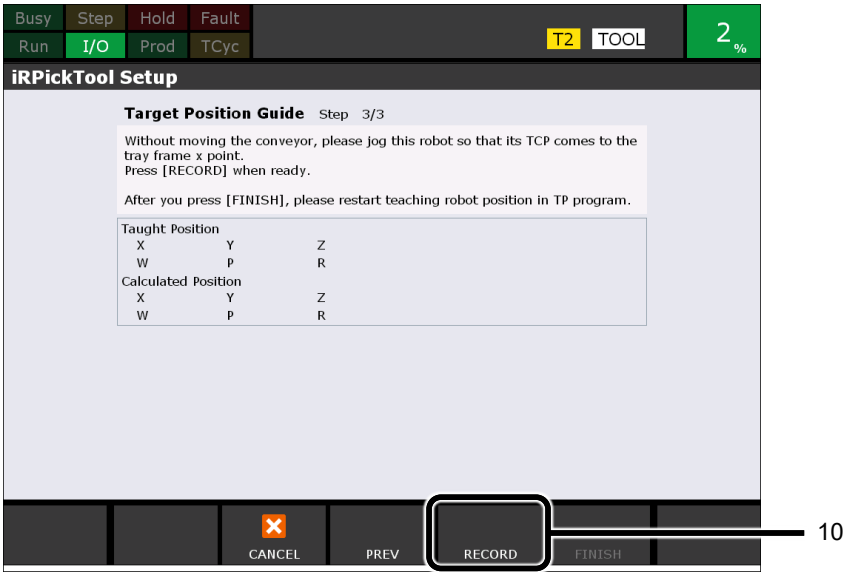


Fig.6.5.6.2 (a)

7 The encoder value obtained at the time of touch-up and the touch-up position are displayed, and F5 [NEXT] key is enabled.
8 If you press F5 [NEXT] key.

A screen as shown below is displayed.



9 Jog the robot, and touch up the X direction point of the tray frame.
10 Press F4 [RECORD] key. The touch-up position and the calculated position are displayed, and F5 [FINISH] key is enabled.

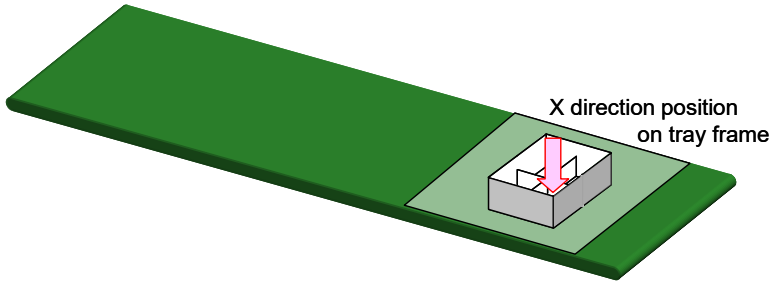


Fig.6.5.6.2 (b)

11 If you press F5 [FINISH] key.

The initial setup screen is displayed again.
Values are input for X, Y, and R of the target position.

12 Teach the position to this robot.
13 Then, move the conveyor further to teach the position to the downstream robots.

6.5.7 RESTRICTION CONCERNING SENSORS THAT CAN BE OPERATED SIMULTANEOUSLY

Up to four sensors can be operated simultaneously by one robot.
If there is a sensor combination that satisfies all the following conditions, the sensors cannot be started. In such a case, take measures such as changing the controller to which the sensors are connected.

- All sensors are connected to the same controller.
- All sensors use ACCUTRIG. *1
- The encoder used in the conveyor station corresponding to each sensor is the same. *2

*1 For the combinations of the trigger conditions and actions for using ACCUTRIG, refer to Table 6.5.7(a).

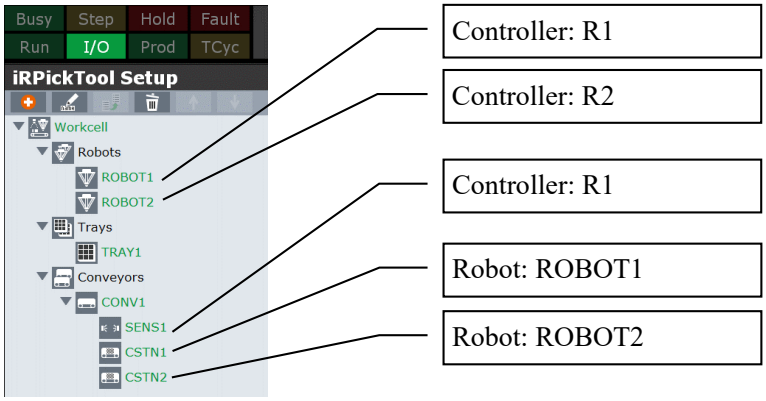
Table 6.5.7(a) Combinations of trigger conditions and actions for using ACCUTRIG

Trigger condition	Action	Use of ACCUTRIG
[Distance]	Put part in queue	×
	Find part by vision	×
	Run program	○
[DI]	Put part in queue	○
	Find part by vision	×
	Run program	○
[HDI]	Put part in queue	×
[FLAG]	Put part in queue	×
[RI]	Put part in queue	○
	Find part by vision	×
	Run program	○

*2 The conveyor station corresponding to the sensor refers to the one that satisfies all of the following conditions:

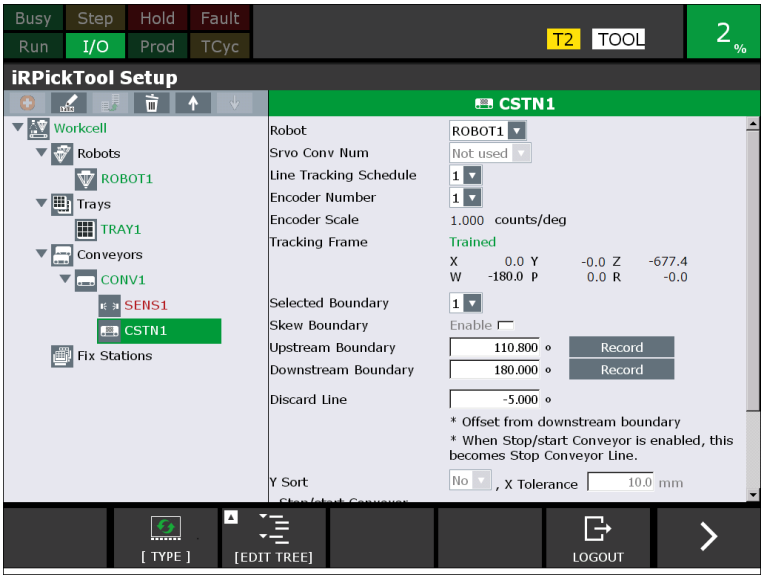
- The conveyor station belongs to the same conveyor to which the sensor belongs.
- The controller corresponding to the robot at the conveyor station and the controller to which the sensor is connected are the same.

In the configuration shown in the figure below, CSTN1 is the conveyor station corresponding to SENS1.



6.6 SETUP OF A CONVEYOR STATION

Set up each conveyor station.
The setup screen for a conveyor station is as below.



> **⚠ CAUTION**
> Most of the items on the conveyor station setup screen cannot be set unless the setup of the tracking frame is complete. Set up the tracking frame before setting up individual conveyor stations.

[Robot]

Select the robot that operates with this conveyor station.

[Srvo Conv Num]

Select the number of the servo conveyor settings to be used for this conveyor station.

You set up the servo conveyor in the dedicated menu, not in *i*RPickTool menu. For details, see Section 9.7, "SERVO CONVEYOR LINE TRACKING FUNCTION".

[Line Tracking Schedule]

Select the number of the tracking schedule to be used for this conveyor station. You specify the different number for each conveyor normally.

[Encoder Number]

Select the number of the encoder to be used for this conveyor station.

[Encoder Scale]

The currently set encoder scale is displayed.

[Tracking Frame]

The status and coordinates of the currently set tracking frame are displayed.

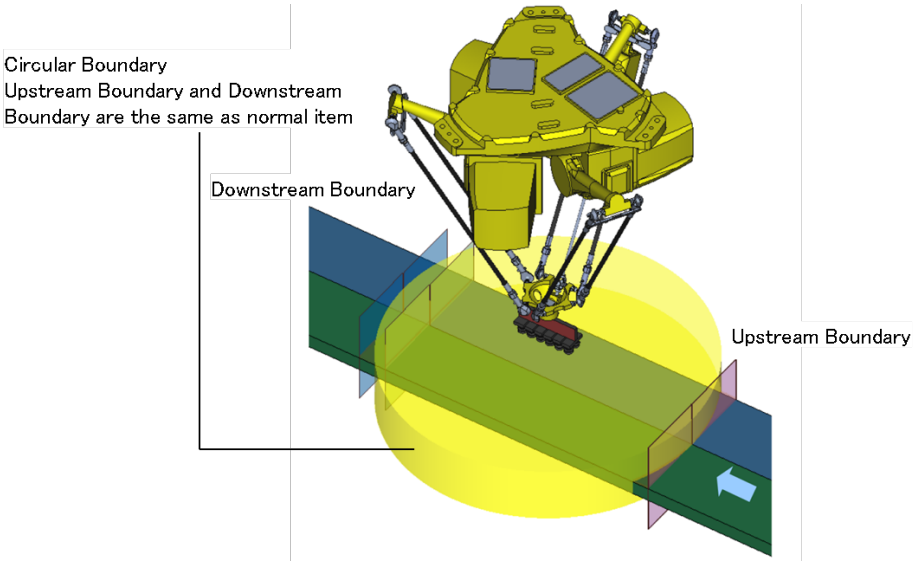
[Selected Boundary]

Select the number of the tracking area to be used. Normally, select 1.

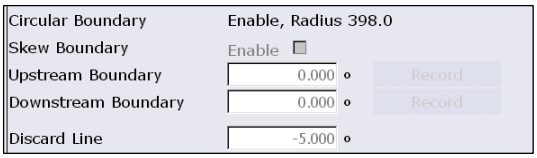
NOTE

To change the tracking area to be used by a TP program, use the SELBOUND tracking instruction. For details, refer to the description of Tracking Instructions in "Line Tracking OPERATOR'S MANUAL B-83474EN".
Changes made by the SELBOUND tracking instruction are not automatically reflected on the *i*RPickTool Setup screen. Close the Setup screen and reopen it, or refresh the tree.

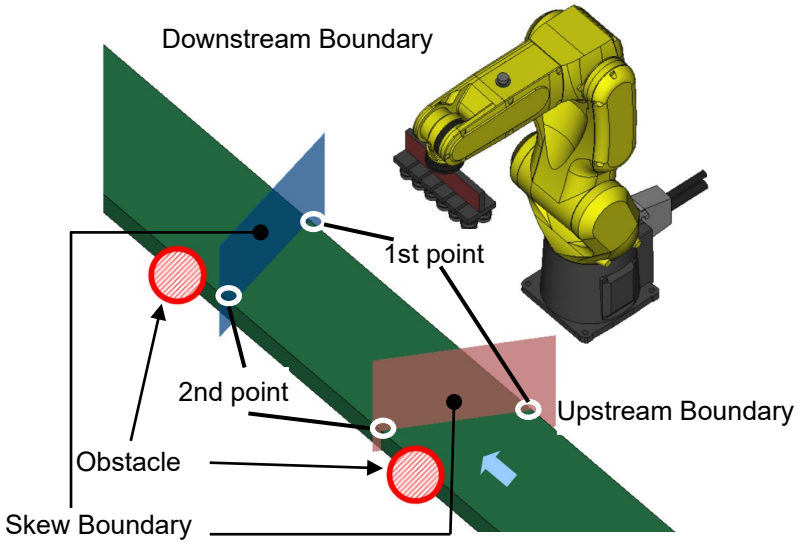
[Circular Boundary]



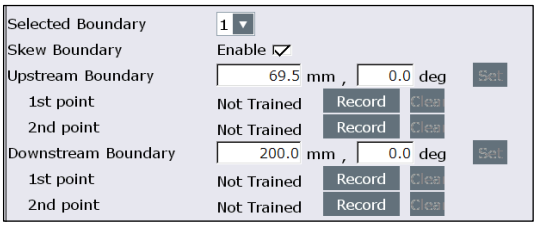
[Enable] and [Radius] will be displayed as shown below only when [Circular Boundary] for the selected robot is enabled.



[Skew Boundary]



If you want to set a skew boundary, enable this.
If you enable skew boundary, skew positions on the upstream and downstream sides can be recorded, as shown below.



If you have selected circular tracking, this cannot be selected.

[1st point]

Touch up the 1st point on the upstream boundary and the 1st point on the downstream boundary, and set record.

[2nd point]

Touch up the 2nd point on the upstream boundary and the 2nd point on the downstream boundary, and set record.

[RECORD] button

Record the set positions for the 1st and 2nd points. If you press the [RECORD] button, 'Not Trained' will change to 'Trained'.

[Clear] button

Clear the set positions for the 1st and 2nd points.

[Upstream Boundary]

Specify the position of the upstream boundary of the tracking area. [Upstream Boundary] represents the most upstream position that the robot can reach.
When setting [Upstream Boundary] and [Downstream Boundary], make sure that the user tool frame is selected that is used by the program that causes the robot to follow the motion of the conveyor. Normally, the user tool frame is used that is set for the mechanical interface of the robot wrist. See Subsection 6.9.2.2, "Pick program".
Move the TCP by jogging to the upstream boundary of the tracking area, and click the [Record] button. The X coordinate value in the tracking frame is set. When the tracking type is "Circle", the angle in degree is set. The value can be changed manually as well.

If you enable skew boundary, you will be able to set the 1st point and 2nd point for skew boundary.

[Downstream Boundary]

Specify the position of the downstream boundary of the tracking area. [Downstream Boundary] represents the most downstream position that the robot can reach.
Move the TCP by jogging to the downstream boundary of the tracking area, and click the [Record] button. The X coordinate value in the tracking frame is set. When the tracking type is "Circle", the angle in degree is set. The value can be changed manually as well.

If you enable skew boundary, you will be able to set the 1st point and 2nd point for skew boundary.

[Set] button

Set the mm value and deg value for the upstream boundary of the tracking area and the downstream boundary of the tracking area.

[Discard Line]

Specify the position where the robot decides to discard a part carried on the conveyor. This is an offset from the downstream boundary of the tracking area ([Downstream Boundary]). Normally, set a slightly small negative value. This allows the robot to decide that a part cannot be picked up shortly before it leaves the tracking area. A positive value cannot be set.

Determine the value of the discard line based on the average time from the moment when the robot starts tracking a part until it completes the operation (e.g., pickup) for that part and the conveyor speed, as follows.

When Line conveyor is used:

If the robot operation time is approximately 0.8 seconds and the conveyor speed is 100 mm/s, the free-running distance of the conveyor is:
0.8 (s) x 100 (mm/s) = 80 (mm)
In this case, set the value of the discard line to −100 mm or so, allowing for some margin.

When Circular conveyor is used:

If the robot operation time is approximately 0.8 seconds and the conveyor speed is 5 deg/s, the free-running distance of the conveyor is:
0.8 (s) x 5 (deg/s) = 4 (deg)
In this case, set the value of the discard line to −5 deg or so, allowing for some margin.

> ⚠ **CAUTION**
> When the stop/start conveyor function described below is enabled, even a part passes the discard line, a robot moves to pick it up. Because the conveyor starts stopping after the part passes the discard line as described below, to pick it up after the conveyor stops, the robot should move in the downstream of the discard line.

NOTE

If the space between the upstream boundary of the tracking area and the discard line is narrow, the situation is prone to occur where there is no part that can be allocated to the conveyor station. If there is no part that can be allocated to the conveyor station, the robot needs to wait until the next part passes the upstream boundary of the tracking area. If there is always at least one part between the upstream boundary of the tracking area and the discard line, the robot can pick up parts continuously. For example, if parts are supplied at almost even intervals, widening the space between the upstream boundary of the tracking area and the discard line makes it easier for the robot to pick up parts continuously, rather than supplying parts at even intervals.

Positional relationship of the items

The positional relationship of the "upstream boundary of the tracking area", "downstream boundary of the tracking area", and "discard line" is outlined below.

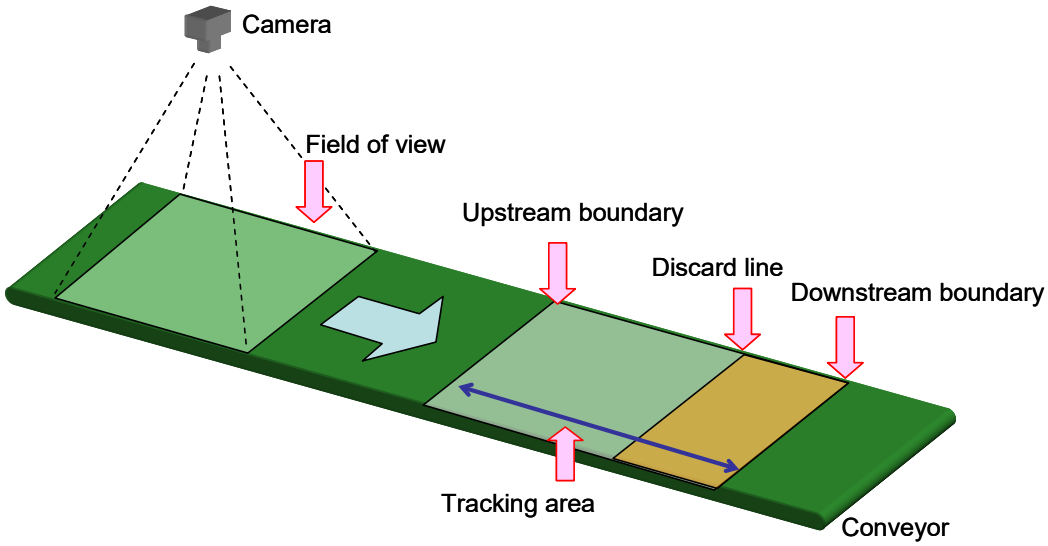


Fig.6.6 (a) Positional relationship between tracking area and discard line

[Y Sort]

This function rearranges the order of the information about the parts on the conveyor in the Y direction of the tracking frame - that is, the direction perpendicular to the conveyor moving direction - so that the parts are handled sequentially beginning with the one at an edge of the conveyor.

In [X Tolerance], specify a range of the conveyor moving direction in which parts are to be regarded as being in the same row. For example, if you specify 100 mm in [X Tolerance], the parts that are within 100 mm of the most downstream part are searched to find the part located at the position whose Y value is the smallest (or largest) and the information about that part is allocated to the robot first.

⚠ CAUTION

Y sort cannot work in the following cases.

- A tray is set for the conveyor
- Skew Boundary is enabled
- Circular Boundary is enabled for the robot

[Stop/start Conveyor]

This function automatically stops the conveyor when a part passes the discard line. When the robot completes the processing of that part, the conveyor is restarted.

When using this function, specify the number of the DO signal used to stop and restart the conveyor, as well as the polarity indicating whether to turn the signal on or off to stop the conveyor.

Enabling this function changes the setting criterion for the discard line. Specify the distance that the conveyor coasts during the time from the issuance of the conveyor stop signal until the conveyor actually stops. For example, if the conveyor coasts 150 mm, set the value of the discard line to −150 mm or so.

[Restart Conveyor Line]

After a part passes the discard line and the conveyor stops, even if the part is picked up, the conveyor does not restart until there exists no other part in the downstream area of this line.
Therefore, if there are many parts near the discard line, the conveyor will stop right away even if it is restarted. To avoid such a situation, set a restart line for the conveyor.

See the figure below. The conveyor stopped because the part A passes the stop onveyor line (= discard line). Even if the part A is picked up, because the part B exists in the downstream area of the restart conveyor line, the conveyor does not restart.

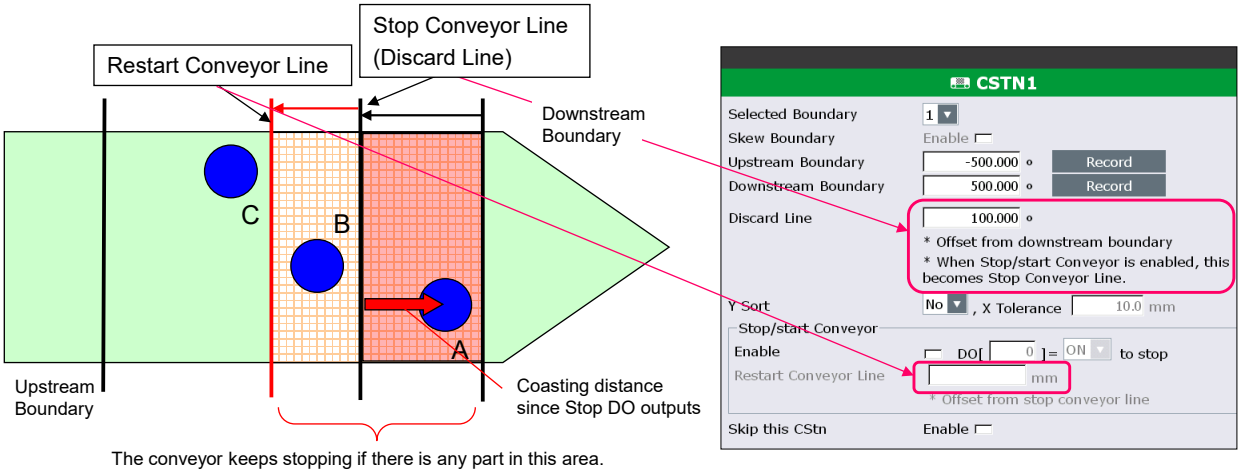


Fig.6.6 (b) Setup items for restart conveyor line and discard line

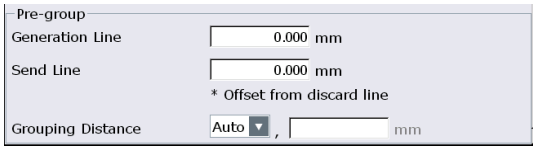
> **⚠ CAUTION**
> 1 In cases where multiple conveyor stations are defined with the same robot controller, if the same DO signal number is specified for the stop/start conveyor function and the polarity of the DO signal is reversed (i.e., the conveyor stops for some conveyor stations when DO is ON, while it stops for the other conveyor stations when DO is OFF), the stop/start conveyor function does not function properly.
> 2 To enable the stop/start conveyor function, set the average value (number of updates) to 10 or higher when setting up the encoder. If the default value of 1 is used, “Move error excess” may occur because the robot attempts to follow the motion of the conveyor acutely as the conveyor stops or restarts. .

[Skip this CStn]

If you enable this function, no part information is input to the conveyor station. Enable it when you intend to have a specific robot rest.

[Pre-Grouping]

Depending on the setup of other items, the setting items of the pre-grouping function may be shown as below.



For details, see Section 6.14, “SETUP OF PRE-GROUPING” .

6.6.1 Adding a Tracking Frame

After you complete the setting of a tracking frame for a conveyor on the conveyor setup page, you can redo the tracking frame setting for a specific conveyor station. Use this procedure in cases such as when you have selected the wrong user tool frame for a robot during the tracking frame setting or when you have added a new robot. Note that you cannot redo the setting for the most upstream conveyor station. The tracking frame remains unchanged for conveyor stations other than the one for which you redo the setting.

The setting procedure is different with depending on the tracking type.

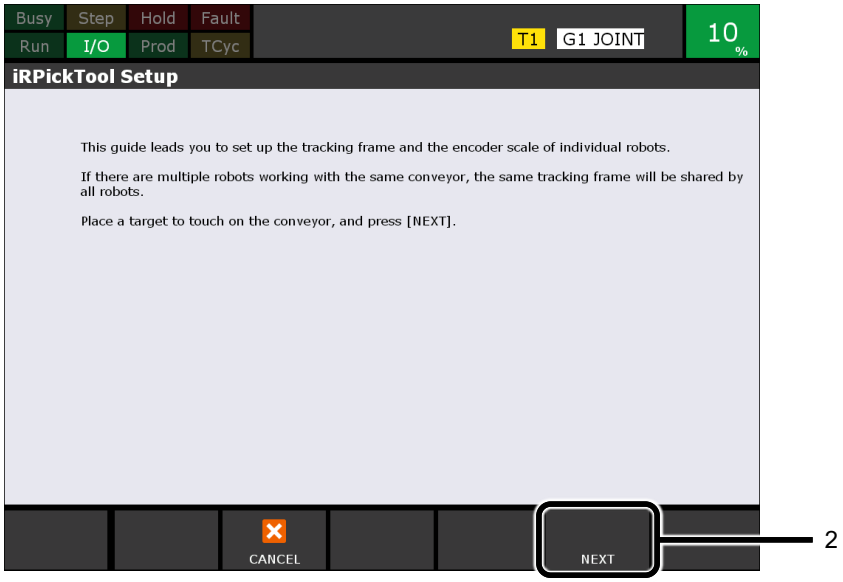
6.6.1.1 Line conveyor

When the tracking type is “Line”, see this procedure. When the tracking type is “Circle”, see the 6.6.1.2, “Circular Conveyor”.

1 In the conveyor station setup screen, press F4 [ADD TRK FRM] key. A screen as shown below is displayed.

A screen as shown below is displayed.

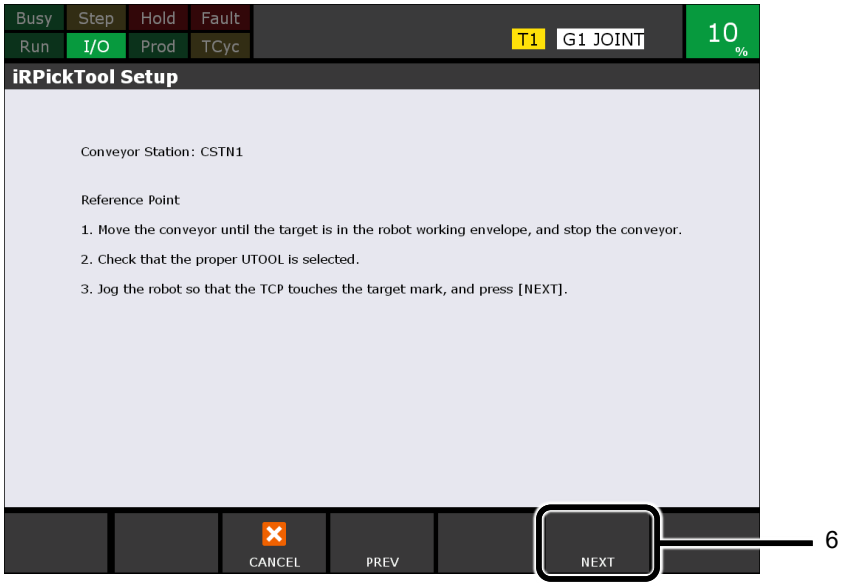
2 Place a fixture that can be touched up with the TCP of the robot at the most upstream area of the conveyor, and press F5 [NEXT] key.



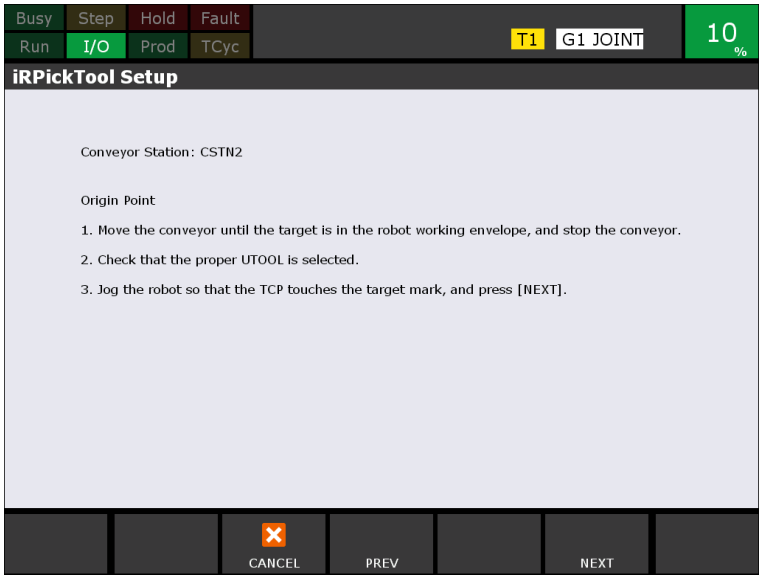
NOTE
1 F4 [ADD TRK FRM] key is not displayed on the setup page for the most upstream conveyor station.
2 F4 [ADD TRK FRM] key is disabled if a tracking frame is not set for the most upstream conveyor station. In that case, set up a tracking frame for all conveyor stations on the conveyor setup page.

A screen as shown below is displayed.

3 For the most upstream robot, select the user tool frame set in Section 3.4, “SETTING OF THE TCP” as the jog feed frame.
4 Move the conveyor until the fixture comes into the most upstream robot operation range. and stop it.
5 Jog the robot to touch up the fixture with the TCP.
6 With the fixture touched up, press F5 [NEXT] key.



A screen as shown below is displayed. The name of the conveyor station for which you are redoing the tracking frame setting is displayed on the first line.

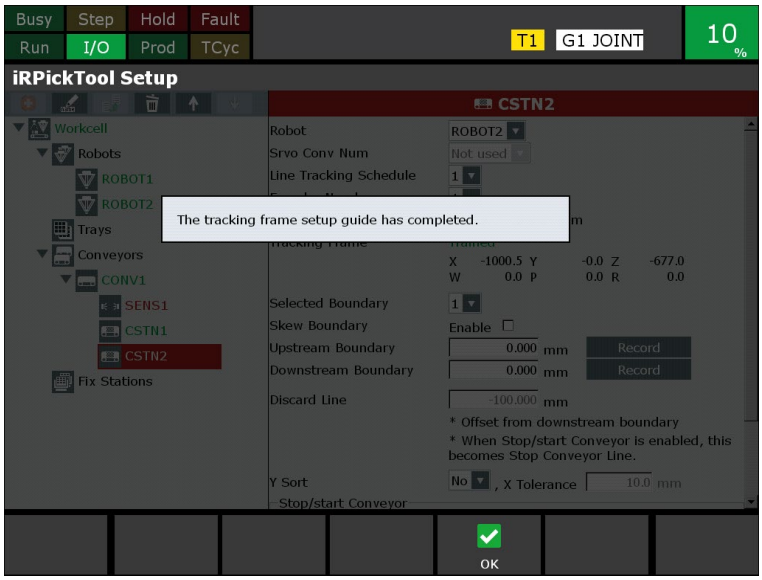


7 For the robot that you are redoing the tracking frame setup for, select the tool frame that you set up in Section 5.4, “SETUP OF A TOOL FRAME” as the manual feed frame.

8 For the conveyor station for which you are redoing the tracking frame setting, touch up the fixture as instructed in steps 14 through 22 in Subsection 3.2.4, “Setting a Tracking Frame”. When you have proceeded to the step of setting the Y-axis direction point, a screen as shown below is displayed. This completes the tracking frame addition procedure, and the same encoder scale as that of the most upstream robot and the newly calculated tracking frame are set.

9 Press F5 [NEXT] key.

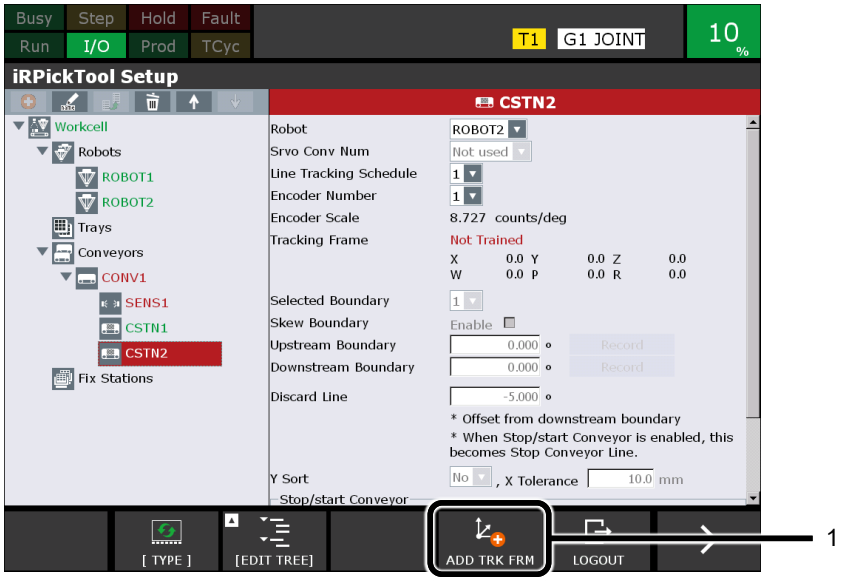
A screen as shown below is displayed.



6.6.1.2 Circular conveyor

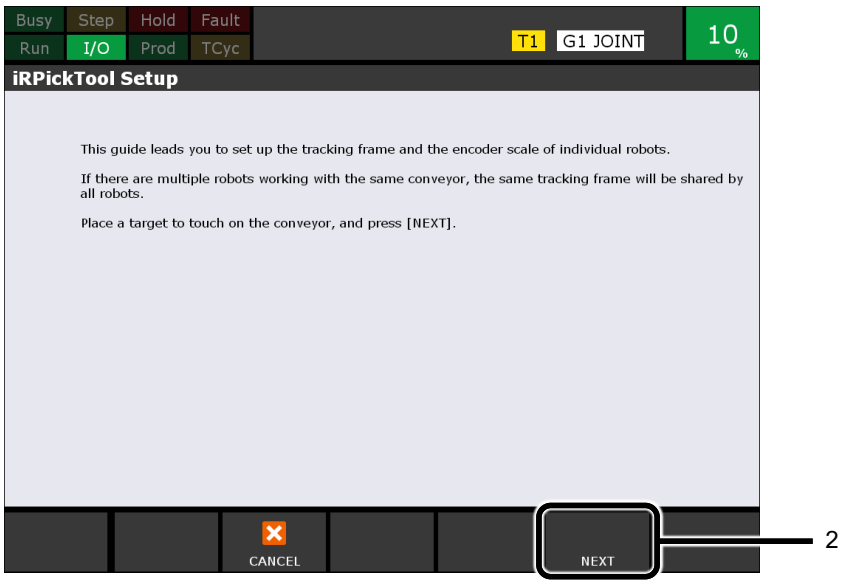
When the tracking type is “Circle”, see this procedure.

1 In the conveyor station setup screen, press F4 [ADD TRK FRM] key.



A screen as shown below is displayed.

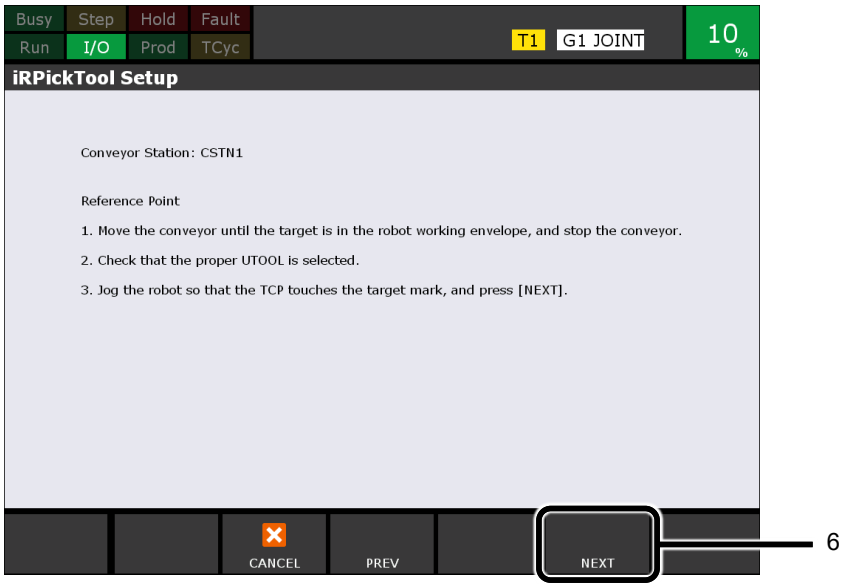
2 Place a fixture that can be touched up with the TCP of the robot at the most upstream area of the conveyor, and press F5 [NEXT] key.



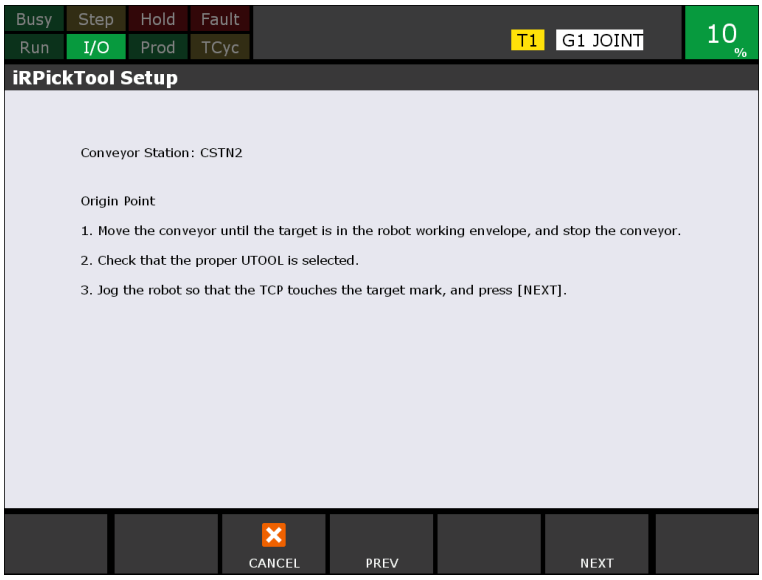
NOTE
1 F4 [ADD TRK FRM] key is not displayed on the setup page for the most upstream conveyor station.
2 F4 [ADD TRK FRM] key is disabled if a tracking frame is not set for the most upstream conveyor station. In that case, set up a tracking frame for all conveyor stations on the conveyor setup page.

A screen as shown below is displayed.

3 For the most upstream robot, select the user tool frame set in Section 5.4, “SETTING OF THE TCP” as the jog feed frame.
4 Move the conveyor until the fixture comes into the most upstream robot operation range, and stop it.
5 Jog the robot to touch up the fixture with the TCP.
6 With the fixture touched up, press F5 [NEXT] key.

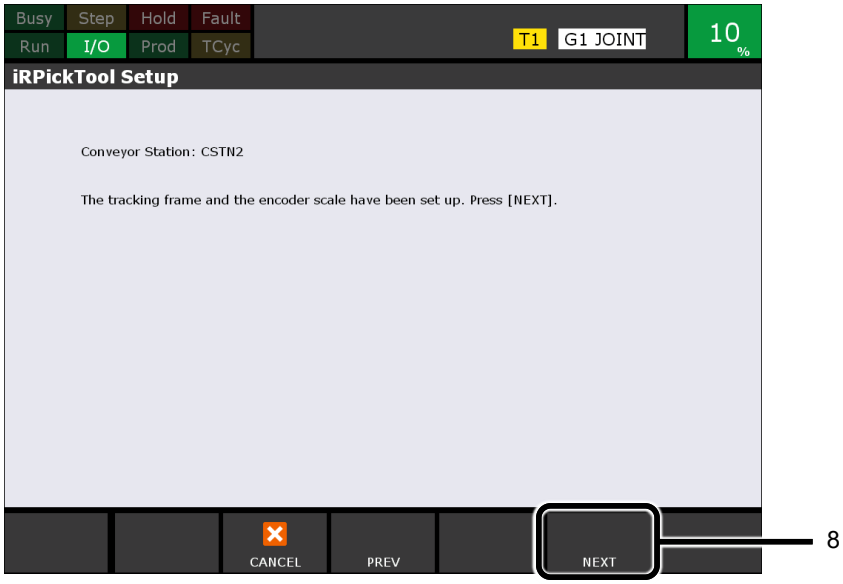


A screen as shown below is displayed. The name of the conveyor station for which you are redoing the tracking frame setting is displayed on the first line.

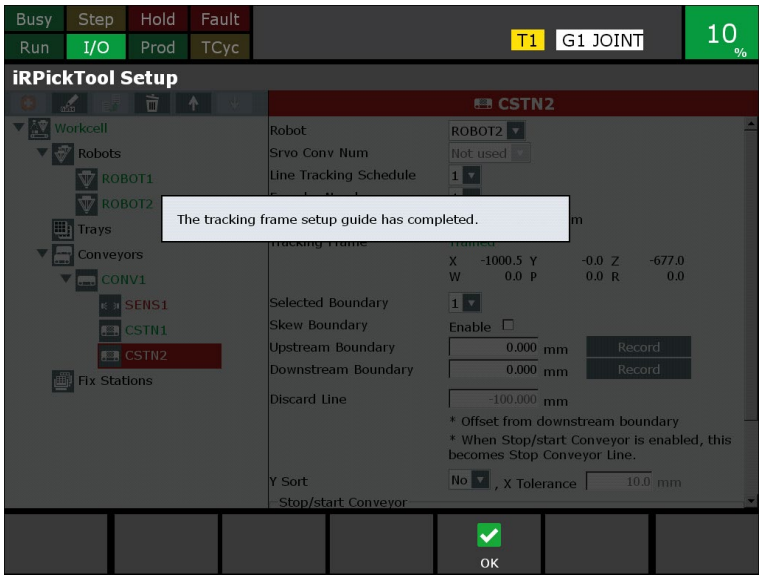


7 This completes the tracking frame addition procedure, and the same encoder scale as the most upstream robot and the newly calculated tracking frame are set. For the conveyor station for which you are redoing the tracking frame setting, touch up the fixture as instructed in steps 4 through 13 in Subsection 6.4.1, "Setting a Tracking Frame". When you have proceeded to the step of setting the assistant point, a screen as shown below is displayed.

8 Press F5 [NEXT] key.

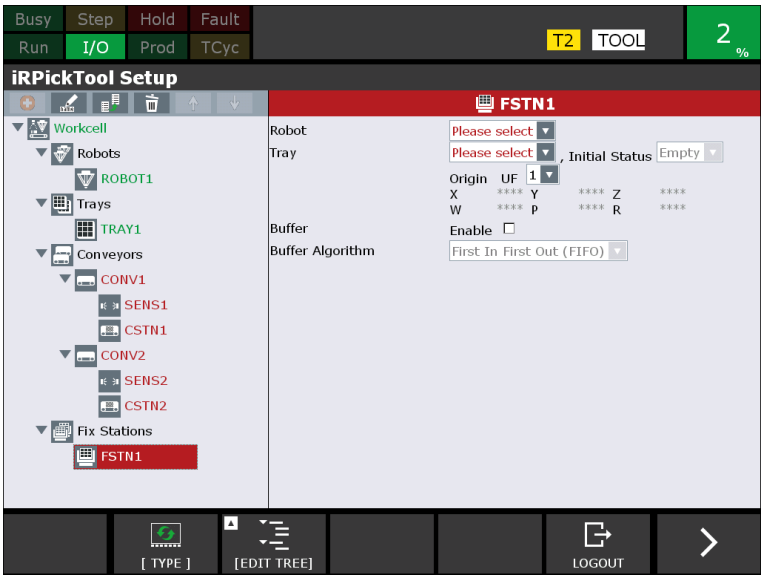


A screen as shown below is displayed.



6.7 SETUP OF A FIXED STATION

Set up a fixed station.



[Robots]

Select the robot that operates with this fixed station.

[Trays]

Select the tray to be placed on this fixed station. As the initial status, select one of the options described below. The initial status is meaningful when layers are specified for the tray.

> **⚠ CAUTION**
> You need to specify a tray for a fixed station even if there is only one position where the tray is to be placed.

[Empty]

An empty tray is put into the queue. In this case, the robot fills the tray with parts beginning with the bottom layer.

[Full]

A tray filled with parts is put into the queue. In this case, the robot picks parts beginning with the top layer.

Also, specify the user frame number indicating the tray frame. The tray needs to be defined for this user frame.
To set the user frame, use the frame setup screen, which is displayed when you select [SETUP] and then [Frames] from the [MENU] key, instead of the *i*RPickTool setup screen. For details, refer to the subsection, "Setting a User Coordinate System" in the section, "SETTING COORDINATE SYSTEMS" in the chapter, "SETTING UP THE ROBOT SYSTEM" in the "OPERATOR'S MANUAL (Basic Operation) B-83284EN".

> ⚠ **CAUTION**
> User frame number can be selected at a fixed station is from 1 to 9.

[Buffer]

To use the tray as the buffer, check the [Enable] box. If you enable the buffer, you can place a part in a tray temporarily, and you can pick it again from there. Even the feed rate fluctuates, you can keep the throughput constant with this buffer function.

[Buffer Algorithm]

Specify the order to pick up parts from the buffer. When you enable the buffer, you can select this.

[First In First Out (FIFO)]

Pick up the oldest part in the buffer.

[Last In First Out (LIFO)]

Pick up the newest part in the buffer.

6.8 KAREL PROGRAMS

The *i*RPickTool basic functions provide the KAREL programs described below. Various functions are implemented by calling these KAREL programs with the CALL instruction. This section describes the specifications of the KAREL programs that are provided.

Every KAREL program used for *i*RPickTool follows the rules mentioned below.

- The name begins with two letters, [PK].
- The following two letters represent the type of object related to the program.
- The fifth and subsequent letters indicate the function of the program.

6.8.1 Workcell-Related Programs

The two letters following [PK] are [WC]; namely, the names of these programs begin with these four letters, [PKWC] though there are some exceptions.

6.8.1.1 PKWCSTART.PC

This program is called at system startup and performs the tasks described below. The first two are the same tasks that are performed by PKWCCHECK described later.

- Checks whether there is any inconsistency in the settings of the controllers within the work cell.
- Checks whether there is any invalid setting.
- Clears the part information from all the conveyor stations corresponding to the calling controller. The program executes the same processing as PKCSCLRQUE described later for all the conveyor stations corresponding to the calling controller. For details, see Subsection 6.8.5.2, "PKCSCLRQUE.PC".
- Starts the monitoring of the trigger condition that is set for each sensor of the calling controller. The program executes the same processing as PKSNSTART described later. For details, see Subsection 6.8.4.5, "PKSNSTART.PC".

There is no argument.

Example:

```
CALL PKWCSTART
```

6.8.1.2 PKWCEND.PC

This program is called at system shutdown and performs the task described below.

- Stops the monitoring of the trigger condition that is set for each sensor of the calling controller. The program executes the same processing as PKSNSTOP described later

There is no argument.

Example:

```
CALL PKWCEND
```

6.8.1.3 PKWCCHECK.PC

This program checks whether there is any inconsistency in the settings of the controllers within the work cell and whether there is any invalid setting. It is called instead of PKWCSTART when PKCSCLRQUE and PKSNSTART are called separately.

There is no argument.

Example:

```
CALL PKWCCHECK
```

6.8.1.4 PKWCRSTPRDST.PC

This program resets Production Status of *i*RPickTool. For Production Status, see the section 9.2 "CHECKING THE PRODUCTION STATUS" in detail.

There is no argument.

Example:

```
CALL PKWCRSTPRDST
```

6.8.1.5 PKGETVAR.PC

This program is used to get the value of a KAREL or system variable into a Numeric Register. You can get the value of variable that is up to 3 dimensions deep. Examples of variable names are:
var1_name
var1_name[var1_idx]
var1_name[var1_idx].var2_name[var2_idx].var3_name[var3_idx].

Note that var_name can be system variable. For example, the system variable $pkcs[1].$n2pick[2] are set as follows, in accordance with the above example:

var1_name = ”$ pkcs”
var1_idx = 1
var2_name = ”$ n2pick”
var2_idx = 2
var3_name = ”...”
var3_idx = 0

You can use this to access the variable data of your own custom KAREL programs also.
Sometimes, you may need it to overcome the limitation of the TP instruction “R[x] = $...” because of the limitation of the size of the string you can specify in that instruction.

Argument 1:
Specify the program name as a string. If using system variable, program name = ‘*SYSTEM*’.

Argument 2:
Specify the data type of the variable being retrieved. Use the following constants:
1 for INTEGER
2 for REAL
3 for BOOLEAN.
You can only retrieve the value of the above 3 data types.

Argument 3:
Specify the var1_name as a string (see variable examples above). If using system variable, make sure to begin the var1_name with $.

Argument 4:
Specify the var1_index as integer. If the variable does not have an index, specifiy zero.

Argument 5:
Specify the var2_name as a string (see variable examples above). If using system variable, make sure to begin the var2_name with $. If there is no var2_name, leave it as blank ‘…’.

Argument 6:
Specify the var2_index as integer. If the variable does not have an index, specify zero.

Argument 7:
Specify the var3_name as a string (see variable examples above). If using system variable, make sure to begin the var2_name with $. If there is no var2_name, leave it as blank ‘…’.

Argument 8:
Specify the var3_index as integer. If the variable does not have an index, specify zero.

Argument 9:
Specify the Numeric Register number into which the value of the whole variable should be retrieved.

Example 1:

To get the value of variable whose program is “PROGA” and whose name is “cycle[1].infeed[2].box_orient” and whose value is integer into Numeric Register 95.

```
CALL PKGETVAR (Program Name='PROGA',Integer,Var1 Name='CYCLE',Var1 Index=1,
Var2 Name='INFEED',Var2 Index=2,Var3 Name='BOX_ORIENT',Var3 Index=0,
Value Reg=95)
```

Example 2:

To get the value of variable whose program is “PROGA” and whose name is ‘box_rot” and whose value is real into Numeric Register 96.

```
CALL PKGETVAR (Program Name= 'PROGA',Real,Var1 Name='BOX_ROT',
Var1 Index=0,Var2 Name= '…',Var2 Index=0,Var3 Name= '…',Var3 Index=0,
Value Reg=96)
```


Example 3:

To get the value of a system variable whose name is “$pkcs[1].$n2pick[1]” and whose value is an integer into Numeric Register 95.

```
CALL PKGETVAR (Program Name='*SYSTEM*',Integer,Var1 Name=''$PKCS',
Var1 Index=1,Var2 Name='$N2PICK',Var2 Index=2,Var3 Name='…',Var3 Index=0,
Value Reg=95)
```

6.8.1.6 PKSETVAR.PC

This program is used to set the value in a Numeric Register into a KAREL or system variable. You can use this program to set the value of a variable in another controller over the RIPE network also.

Argument 1:

Specify the RIPE name of the controller in which you wish to set the data as a STRING. If you wish to set the data in the controller where the macro executes, then specify blank name or ‘…’.

Argument 2:

Specify the program name as a string. If using system variable, program name = ‘*SYSTEM*’.

Argument 3:

Specify the complete variable name. If using system variable, make sure to begin the name with $. Example: cycle[1].box_rot, $pkcs[1].$n2pick[2], $pkclrq

Argument 4:

Specify the value of the variable. If the value is a Boolean, then use 0 for off or false and 1 for on or true.

Argument 5:

Specify if the variable value is a Boolean. For integer and real values, specify 0. For Boolean values specify 1.

Example 1:

To set the value of a real variable in the same controller where the macro runs and whose program is “PROGA” and whose name is “cycle[1]. box_rot” from Numeric Register 95.

```
CALL PKSETVAR (RIPE Name='…',Program Name='PROGA',
Var Name='CYCLE[1].BOX_ROT',Var Value=R[95],Boolean=0)
```

Example 2:

To set the value of an integer system variable whose name is "$pkcs[1].$n2pick[1]" in a different controller over RIPE network named "ROBOT2" from Numeric Register 95.

```
CALL PKSETVAR (RIPE Name='ROBOT2',Program Name='*SYSTEM*',
Var Name='$PKCS[1].$N2PICK[1]',Var Value=R[95],Boolean=0)
```

6.8.1.7 PKPRGRGETEN.PC

This program returns whether pre-grouping can be used or not in this controller. This program can be used in 7DC3/23(V8.30P/23) or later.

Argument 1:

Specify the number of the register to store the value which shows whether pre-grouping can be used or not. 1 is returned if pre-grouping can be used. Otherwise, 0 is returned.

Example:

The following instruction returns whether pre-grouping can be used or not to R[7].

```
CALL PKPRGRGETEN(PreGrp Enb Reg=7)
```

6.8.1.8 PKRSTWC.PC

This program is used to reset the *i*RPickTool setup data to factory defaults. You would call this program only if you wanted to erase all the setup data of conveyors, conveyor stations, fixed stations, sensors, etc. that you defined in the *i*RPickTool treeview.

- The program only clears the data in the controller in which it is run. Therefore, if you had multiple robot cells and you wanted to clear the data in all the controllers, then you need to run the program in each controller.
- This program only clears only the *i*RPickTool data. It does not clear or reset the track frames or the Position Registers that you have taught.
- This program does not delete the Recipes.

There is no argument.

Example:

```
CALL PKRSTWC
```

6.8.1.9 PKABO.PC

This program is used to abort all the *i*RPickTool programs (tasks) in every controller of your workcell in one shot. You may need to use it in case you want to stop all the robots in case your application detects a fatal error. You could also used it during testing and debugging of your application.

- You need to call PKABO only in one controller to abort all the *i*RPickTool tasks in all the controllers referenced in the treeview.
- This program uses Ethernet to communicate with the other controllers. Therefore, if a controller is off-line, it will not be able to abort all the tasks in that controller.

There is no argument.

Example:

```
CALL PKABO
```

6.8.1.10 PKLABO.PC

This program aborts all the running tasks in the controller in which it is executed.

There is no argument.

Example:

```
CALL PKLABO
```

6.8.1.11 PKPOSTERR.PC

This program is used to post an error to the error line and to the error log with the specified severity (warn, pause or abort).

Argument 1:
Specify the error number.
Argument 2:
Specify the severity (WARN, PAUSE, ABORT).

NOTE
When the argument guidance is supported, you select one of the following options as the severity.
1 WARN
2 PAUSE
3 ABORT
However, the severity is internally set as WARN=0, PAUSE=1 and ABORT=2. So if the argument guidance is not supported in your software version, specify 0 as WARN, 1 as PAUSE and 2 as ABORT.

Example:
To post an error “IRPK-232 Invalid Position Register” as an error that should pause the robot, specify

```
CALL PKPOSTERR(Error Code=127232, PAUSE)
```

6.8.1.12 PKCOPYPR2POS.PC

This program copies the value of a Position Register to the position variable of a TP program.

Argument 1:
Specify the number of the copy source Position Register.

Argument 2:
Specify the TP program name.

Argument 3:
Specify the number of the copy destination position variable.

Argument 4:
Specify the group number.

Example:
The value of the PR [11] is copied to P[2] of MAIN1.TP regarding group 1.

```
CALL PKCOPYPR2POS(PR Num=11,Program Name='MAIN1',Position Num=2,Group Num=1)
```

6.8.2 Robot-Related Programs

The two letters following [PK] are [RB] namely, the names of these programs begin with these four letters, [PKRB].

6.8.2.1 PKRBGETID.PC

This program gets the ID of the robot from a group number.

Argument 1:
Specify the motion group number.

Argument 2:
Specify the Numeric Register in which the ID of the robot corresponding to the group number specified in argument 1 should be returned.

Example:
To get the ID of robot that uses motion group 1 into Numeric Register 7 use the following instruction.

```
CALL PKRBGETID(Group Num=1,Robot ID Reg=7)
```

6.8.2.2 PKRBGETSFLG.PC

This program gets the FLAG number in Skip outbound motion of the robot from the robot ID.

Argument 1:
Specify an ID of a Robot.

Argument 2:
Specify the Numeric Register in which the FLAG number is to be returned.

Example:
To get the FLAG number in Skip outbound motion of the robot whose ID is 1, use the following instruction. The FLAG number is returned in Numeric Register 7.

```
CALL PKRBGETSFLG(Robot ID=1,SkipFlag Reg=7)
```

6.8.3 Conveyor-Related Programs

The two letters following [PK] are [CV]; namely, the names of these programs begin with these four letters, [PKCV].

6.8.3.1 PKCVSETLBD.PC

This program changes the load balance of each conveyor station on a conveyor.
The change is reflected immediately.
When there are multiple robots, if you execute this in one robot, the setting is reflected to all robots.

Argument 1:
Specify the name of the conveyor using a character string.

Argument 2:

Specify the number of the register storing the load balance data. The value stored in R[Register Number] is used as the load balance data for the most upstream conveyor station. The value stored in R[Register Number + 1] is used as the load balance data for the second most upstream conveyor station. The range of values that can be set in the registers is 0 to 255.

Argument 3:

Specify the number of conveyor stations for which to set the load balance. Normally, specify the value that is equal to the number of conveyor stations that are set on the conveyor. For example, if there are three conveyor stations on the conveyor and if you specify 3 in this argument, you can set the load balance for all those conveyor stations.

Argument 4:

Set the value of [Bypass].

Argument 5 (optional):

Specify the model ID for which to set the load balance. If [For all model IDs] is selected for the load balance mode, do not specify this argument.

Example:

The following instruction sets the load balance for conveyor data CONV1 to 1:2:3 beginning with the most upstream conveyor station. The value of [Bypass] is set to 0. The registers to be used are R[11], R[12], and R[13]. Set 1 in R[11], 2 in R[12], and 3 in R[13] in advance.

```
CALL PKCVSETLBD('CONV1',First Reg=11,Num Cstn=3,Bypass=0)
```

⚠ CAUTION

1 When you execute this program during opening the setup page, the parameter to be changed might not be updated soon. But they are updated automatically after a few seconds.

2 When you use multiple controllers and call KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC), be careful about the following points.

- Call the listed KAREL programs for setting parameters before PKWCSTART or PKWCCHECK
- Insert WAIT instruction before PKWCSTART or PKWCCHECK in all controllers in order to wait for the completion of the parameter changes in all controllers. Set the wait time to 5 seconds or so. If parameter changes have not completed when PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
- If you call the multiple listed KAREL programs for setting parameters, call them in one controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other controllers also. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCCHECK is called.

6.8.3.2 PKCVSETTRNAM.PC

This program changes a tray in a conveyor. When there are multiple robots, if you execute this in one robot, the setting is reflected to all robots.

Argument 1:
Specify the name of the conveyor using a character string.

Argument 2:
Specify the name of the tray using a character string.

Example:
The following instruction changes a tray name which is selected in “CONV1” to “TRAY2”.

```
CALL PKCVSETTRNAM ('CONV1','TRAY2')
```

⚠ CAUTION

1 Please call this program in top of the main program, that is before PKWCSTART or PKSNSTART, so that the changed parameters are reflected correctly.
2 When you execute this program during opening the setup page, the parameter to be changed might not be updated soon. But they are updated automatically after a few seconds.
3 When you use multiple controllers and call KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC), be careful about the following points.
- Call the listed KAREL programs for setting parameters before PKWCSTART or PKWCCHECK
- Insert WAIT instruction before PKWCSTART or PKWCCHECK in all controllers in order to wait for the completion of the parameter changes in all controllers. Set the wait time to 5 seconds or so. If parameter changes have not completed when PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
- If you call the multiple listed KAREL programs for setting parameters, call them in one controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other controllers also. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCCHECK is called.

6.8.3.3 PKCVENBLB.PC

This program enables or disables the load balancing function. The change is reflected immediately. When there are multiple robots, if you execute this in one robot, the setting is reflected to all robots.

Argument 1:
Specify the name of the conveyor using a character string.

Argument 2:
Specify “Enable” to enable the load balancing function or “Disable” to disable it.

NOTE
When the argument guidance is supported, you select one of the following options.
1 Enable
2 Disable
However, the load balancing function is internally set as Enable=1 and Disable=0. So if the argument guidance is not supported in your software version, specify 1 as Enable or 0 as Disable.

Example:
The following instruction enables the load balancing function for conveyor data CONV1.

```
CALL PKCVENBLB ('CONV1',Enable)
```


⚠ CAUTION
1 When enabling the load balancing function by specifying 1 in argument 2, be careful about when this KAREL program is called. Care needs to be exercised when parts flow at a fast pace and some of them pass the discard line. If a part passes the discard line immediately after the load balancing function is enabled, the robot judges that it has missed the first part and picks two parts in a row, when it is supposed to pick every other part. To prevent this, call this KAREL program while the parts yet to be processed remain in the upstream side; for example, immediately after PKCSGETQUE is called.
2 When you execute this program during opening the setup page, the parameter to be changed might not be updated soon. But they are updated automatically after a few seconds.
3 When you use multiple controllers and call KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC), be careful about the following points.
- Call the listed KAREL programs for setting parameters before PKWCSTART or PKWCCHECK
- Insert WAIT instruction before PKWCSTART or PKWCCHECK in all controllers in order to wait for the completion of the parameter changes in all controllers. Set the wait time to 5 seconds or so. If parameter changes have not completed when PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
- If you call the multiple listed KAREL programs for setting parameters, call them in one controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other controllers also. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCCHECK is called.

6.8.3.4 PKCVENBSC.PC

This program enables or disables the stop/start conveyor function for a specified conveyor station. The change is reflected immediately. When there are multiple robots, if you execute this in one robot, the setting is reflected to all robots.

Argument 1:

Specify the name of the conveyor using a character string.

Argument 2:

Specify the number of the register storing the value indicating whether to enable the stop/start conveyor function (1) or disable it (0). The value stored in R[Register Number] is used to enable or disable the function for the most upstream conveyor station. The value stored in R[Register Number + 1] is used to enable or disable the function for the second most upstream conveyor station.

Argument 3:

Specify the number of conveyor stations for which to enable or disable the stop/start conveyor function. Normally, specify the value that is equal to the number of conveyor stations that are set on the conveyor. For example, if there are three conveyor stations on the conveyor and if you specify 3 in this argument, you can enable or disable the stop/start conveyor function for all those conveyor stations.

Example:

The following instruction enables and disables the stop/start conveyor function for the conveyor stations of conveyor data CONV1 as follows. The instruction enables the function for the most upstream conveyor station, disables it for the second most upstream conveyor station, and enables it for the third most upstream conveyor station. The registers to be used are R[11], R[12], and R[13]. Set 1 in R[11], 0 in R[12], and 1 in R[13] in advance.

```
CALL PKCVENBSC('CONV1',First Reg=11,Num CStn=3)
```

⚠ CAUTION

1 When you execute this program during opening the setup page, the parameter to be changed might not be updated soon. But they are updated automatically after a few seconds.

2 When you use multiple controllers and call KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC), be careful about the following points.

- Call the listed KAREL programs for setting parameters before PKWCSTART or PKWCCHECK
- Insert WAIT instruction before PKWCSTART or PKWCCHECK in all controllers in order to wait for the completion of the parameter changes in all controllers. Set the wait time to 5 seconds or so. If parameter changes have not completed when PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
- If you call the multiple listed KAREL programs for setting parameters, call them in one controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other controllers also. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCCHECK is called.

6.8.3.5 PKCVGETNAME.PC

This program is used to get the name of a conveyor you defined in the treeview from the ID of the conveyor. The name is returned in a String Register.

Argument 1:
Specify the ID of the conveyor.

Argument 2:
Specify a String Register number where the conveyor name is stored.

Example:
To return the name of the conveyor whose ID is 1 in String Register 10, use the following instruction:

```
CALL PKCVGETNAME(Conv ID=1,Conv Name SR=10)
```

6.8.4 Sensor-Related Programs

The two letters following [PK] are [SN]; namely, the names of these programs begin with these four letters, [PKSN].

6.8.4.1 PKSNENBSENS.PC

This program enable or disable a sensor. When there are multiple robots, if you execute this in one robot, the setting is reflected to all robots.

Argument 1:
Specify the name of the sensor using a character string.

Argument 2:
Specify “Enable” to enable the sensor or “Disable” to disable it.

NOTE
When the argument guidance is supported, you select one of the following options.
1 Enable
2 Disable
However, the sensor is internally set as Enable=1 and Disable=0. So if the argument guidance is not supported in your software version, specify 1 as Enable or 0 as Disable.

Example:
The following instructions disable ‘SENS1’ and enable ‘SENS2’ and start the sensors.

```
CALL PKSNENBSENS ('SENS1',Disable)
CALL PKSNENBSENS ('SENS2',Enable)

CALL PKWCSTART
```

⚠ CAUTION

1 When you switch enable or disable after PKWCSTART or PKSNSTART, all sensors need to be enabled when the sensors start with PKWCSTART or PKSNSTART.
2 When you execute this program during opening the setup page, the parameter to be changed might not be updated soon. But they are updated automatically after a few seconds.
3 When you use multiple controllers and call KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC), be careful about the following points.
   - Call the listed KAREL programs for setting parameters before PKWCSTART or PKWCCHECK
   - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all controllers in order to wait for the completion of the parameter changes in all controllers. Set the wait time to 5 seconds or so. If parameter changes have not completed when PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
   - If you call the multiple listed KAREL programs for setting parameters, call them in one controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other controllers also. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCCHECK is called.

6.8.4.2 PKSNSETVPNAM.PC

This program changes a vision process in a sensor. The vision program set for the sensor by [Find by Multiview] can also be changed. When there are multiple robots, if you execute this in one robot, the setting is reflected to all robots. This program can be used in 7DC3/02(V8.30P/02) or later.

Argument 1:

Specify the name of the sensor using a character string.

Argument 2:

Specify the name of the vision process using a character string.

Example:

The following instruction changes a vision process name which is selected in “SENS1” to “VT2”.

```
CALL PKSNSETVPNAM ('SENS1','VP Name=VT2')
```

⚠ CAUTION

1 Please call this program in top of the main program, that is before PKWCSTART or PKSNSTART, so that the changed parameters are reflected correctly.
2 When you execute this program during opening the setup page, the parameter to be changed might not be updated soon. But they are updated automatically after a few seconds.
3 When you use multiple controllers and call KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC), be careful about the following points.
   - Call the listed KAREL programs for setting parameters before PKWCSTART or PKWCCHECK
   - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all controllers in order to wait for the completion of the parameter changes in all controllers. Set the wait time to 5 seconds or so. If parameter changes have not completed when PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
   - If you call the multiple listed KAREL programs for setting parameters, call them in one controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other controllers also. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCCHECK is called.

NOTE

To specify a position register for Argument 2, follow the procedure below:
1 Enter an appropriate character string in the argument.
2 Bring the cursor to the entered character string and press F4 [SELECT].
3 Select [String register].

6.8.4.3 PKSNSETVTOFS.PC

When you use [DI] + [Find part by vision], this program changes a trigger offset in a sensor. When there are multiple robots, if you execute this in one robot, the setting is reflected to all robots.

Argument 1:
Specify the name of the sensor using a character string.

Argument 2:
Specify a value as the trigger offset.

Example:
The following instruction changes a trigger offset which is selected in “SENS1” to 100 mm.

```
CALL PKSNSETVTOFS('SENS1',Trigger Offset=100)
```

⚠ CAUTION

1 Please call this program in top of the main program, that is before PKWCSTART or PKSNSTART, so that the changed parameters are reflected correctly.
2 When you execute this program during opening the setup page, the parameter to be changed might not be updated soon. But they are updated automatically after a few seconds.
3 When you use multiple controllers and call KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC), be careful about the following points.
   - Call the listed KAREL programs for setting parameters before PKWCSTART or PKWCCHECK
   - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all controllers in order to wait for the completion of the parameter changes in all controllers. Set the wait time to 5 seconds or so. If parameter changes have not completed when PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
   - If you call the multiple listed KAREL programs for setting parameters, call them in one controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other controllers also. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCCHECK is called.

6.8.4.4 PKSNSETMTOFS.PC

When you use [DI] + [Find part by multiview], this program changes a trigger offset of each view in a sensor. When there are multiple robots, if you execute this in one robot, the setting is reflected to all robots.

Argument 1:

Specify the name of the sensor using a character string.

Argument 2:

Specify a value as the trigger offset for view 1.

Argument 3:

Specify a value as the trigger offset for view 2.

Example:

The following instruction changes trigger offsets of multiview which are selected in “SENS1” to 100 and 200 mm.

```
CALL PKSNSETMTOFS('SENS1',Trigger Offset 1=100,Trigger Offset 2=200)
```

⚠ CAUTION

1 Please call this program in top of the main program, that is before PKWCSTART or PKSNSTART, so that the changed parameters are reflected correctly.
2 When you execute this program during opening the setup page, the parameter to be changed might not be updated soon. But they are updated automatically after a few seconds.
3 When you use multiple controllers and call KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC), be careful about the following points.
   - Call the listed KAREL programs for setting parameters before PKWCSTART or PKWCCHECK
   - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all controllers in order to wait for the completion of the parameter changes in all controllers. Set the wait time to 5 seconds or so. If parameter changes have not completed when PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
   - If you call the multiple listed KAREL programs for setting parameters, call them in one controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other controllers also. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCCHECK is called.

6.8.4.5 PKSNSTART.PC

This program starts the monitoring of the trigger condition that is set for each sensor of the controller that calls the program. If the trigger condition is met, the specified action is automatically performed.
Executing this KAREL program for a controller that has no sensor generates warning IRPK-031. This is a mere informational message and does not affect the operation.
There is no need to use this program if you call PKWCSTART.

There is no argument.

Example:

```
CALL PKSNSTART
```

6.8.4.6 PKSNSTOP.PC

This program stops the monitoring of the trigger condition that is set for each sensor of the controller that calls the program.
There is no need to use this program if you call PKWCEND.

There is no argument.

Example:

```
CALL PKSNSTOP
```

6.8.5 Conveyor Station-Related Programs

The two letters following [PK] are [CS]; namely, the names of these programs begin with these four letters, [PKCS].

6.8.5.1 PKCSGETID.PC

This program gets the ID of the conveyor station to which the robot that calls the program belongs in a specified conveyor.

Argument 1:
Specify a conveyor name.

Argument 2:
Specify the number of the register storing the ID of the conveyor station.

Argument 3 (optional):
Specify a group number. If you omit this argument, the program gets the ID of the conveyor station whose group number is 1.

> ⚠ **CAUTION**
> In argument 1, specify a conveyor name, not a conveyor station name.

If any of these arguments is invalid, one of the messages shown below may be displayed in the status window during the program execution, potentially ending the program. The causes of these messages and the actions to be taken are as follows.

Bad Conveyor Name:
The controller that called this program does not exist in the child objects of the specified conveyor and, if a group is specified, the conveyor station for which the robot corresponding to that group number is selected does not exist.

Internal error 17030:
A nonexistent register number is specified. If a register is given in the argument or indirectly specified, check that an existing register number is set in the corresponding register.

Example:
Conveyor station 1 associated with the robot that executes this KAREL program belongs to conveyor CONV1. The following instruction stores the ID of conveyor station 1 in R[11]. Since the robot belongs to group 1, the third argument is omitted.

```
CALL PKCSGETID ('CONV1',CStn ID Reg=11)
```

6.8.5.2 PKCSCLRQUE.PC

This program clears the part information from a specified conveyor station. All the part information that is present in the conveyor station when the program is executed is cleared. Normally, call this program only once at system startup. There is no need to use the program if you call PKWCSTART.

Argument 1:
Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.

Example:
The following instruction clears the part information from conveyor station 1. The ID is stored in R[11].

```
CALL PKCSCLRQUE (CStn ID=R[11:CSTN1 ID])
```

6.8.5.3 PKCSGETQUE.PC

This program gets the information of a part from a conveyor station. The acquired part information is stored in a Vision Register. If the part whose information is acquired is within the tracking area, the value of the encoder is set as the trigger for the tracking motion. The robot waits for a specified time before the part whose information is acquired reaches the tracking area.

Argument 1:
Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.

Argument 2:
Normally, specify 1. If you want the information of successive parts to be allocated regardless of the load balance for conveyor stations that is set during the conveyor setup, specify 2 or a greater value. For details, see Subsection 6.9.2.2, "Pick program".

Argument 3:
Specify the time in milliseconds that the robot waits before the part whose information is acquired reaches the tracking area. If you specify a negative value, the robot waits infinitely. If you specify 0, the robot performs the next statement without waiting.

Argument 4:
Specify the number of the Vision Register in which to store the acquired part information. The information is stored in the Vision Register even if the specified time expires before the part reaches the tracking area.

Argument 5:
Specify the number of the register to store the processing status. The following statuses may be stored.
0: The information of the part within the tracking area is allocated, and the trigger for the tracking motion is set.
1: The part whose information is acquired fails to reach the tracking area within the specified time.
2: No part information is acquired.

Argument 6 (optional):
If you want to acquire the information of parts with a specific model ID, specify that model ID. This argument is optional and normally does not need to be specified. An example of using this argument is given in Chapter 8, "EXAMPLES OF SETUP IN ACCORDANCE WITH USE".

If any of these arguments is invalid, "P_CSGETQUE error (error number)" may be displayed in the status window during the program execution, potentially ending the program. The causes of these messages and the actions to be taken are as follows.

P_CSGETQUE 17030:
A nonexistent register number is specified. If a register is given in the argument or indirectly specified, check that an existing register number is set in the corresponding register.

Example:
When the ID of conveyor station 1 is stored in R[11], the following instruction stores the information

about the unprocessed part from conveyor station 1 in VR[1] and the processing status in R[2]. If the unprocessed part has not reached the tracking area, the robot waits for 100 milliseconds.

```
CALL PKCSGETQUE (CStn ID=R[11:CSTN1 ID],Consec Flag=1,Timeout (ms)=100,
Offset VR=1,Stat Reg=2)
```

⚠ CAUTION

1 If the acquired part information is a cell of a tray, it is calculated based on the relative position between this cell and the cell 1. So when you teach the robot position to a tray, you must make the taught position conform to the position of the cell 1.
2 If the processing status is 1, the part information has not actually been allocated. Even if you immediately call PKCSGETQUE described next, you may get the information of a different part. Also, if the processing status is 1, PKCSACKQUE described next cannot notify the conveyor station of the processing result.

6.8.5.4 PKCSGETQCELL.PC

This program gets the information of a cell from a conveyor station. The cell which has the same cell ID as the specified one is acquired. The specified cell can be acquired even though there are other unfilled cells which should be filled before the specified cell. The acquired cell information is stored in a Vision Register. If the cell whose information is acquired is within the tracking area, the value of the encoder is set as the trigger for the tracking motion. The robot waits for a specified time before the cell whose information is acquired reaches the tracking area.

Argument 1:

Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.

Argument 2:

Specify the ID of a cell.

Argument 3:

Specify the time in milliseconds that the robot waits before the cell whose information is acquired reaches the tracking area. If you specify a negative value, the robot waits infinitely. If you specify 0, the robot performs the next statement without waiting.

Argument 4:

Specify the number of the Vision Register in which to store the acquired cell information. The information is stored in the Vision Register even if the specified time expires before the cell reaches the tracking area.

Argument 5:

Specify the number of the register to store the processing status. The following statuses may be stored.

0: The information of the cell within the tracking area is allocated, and the trigger for the tracking motion is set.
1: The cell whose information is acquired fails to reach the tracking area within the specified time.
2: No cell information is acquired.

Argument 6

Specify the number of the register to store the number of unfilled cells which should be filled before the specified cell.

If any of these arguments is invalid, "pkcsgetqcell error (error number)" may be displayed in the status window during the program execution, potentially ending the program. The causes of these messages and the actions to be taken are as follows.

pkcsgetqcell 17030:

A nonexistent register number is specified. If a register is given in the argument or indirectly specified, check that an existing register number is set in the corresponding register.

Example:

When the ID of conveyor station 1 is stored in R[11], the following instruction stores the information about Cell5 from conveyor station 1 in VR[1], the processing status in R[2] and the number of unfilled cells which should be filled before Cell5 in R[6]. If the Cell5 has not reached the tracking area, the robot wait for 100 milliseconds.

```
CALL PKCSGETQCELL (CStn ID=R[11:CSTN1 ID],Cell num=5,Timeout (ms)=100,
Offset VR=1,Stat Reg=2,NumUnfilCel Reg=6)
```

6

⚠ CAUTION

1. It is calculated based on the relative position between the specified cell and the cell 1. So when you teach the robot position to a tray, you must make the taught position conform to the position of the cell 1.
2. If the processing status is 1, the part information has not actually been allocated. Even if you immediately call PKCSGETQCELL described next, you may get the information of a cell which belongs to another tray. Also, if the processing status is 1, PKCSACKQUE described next cannot notify the conveyor station of the processing result.

6.8.5.5 PKCSACKQUE.PC

After the processing status of PKCSGETQUE or PKCSGETQCELL becomes 0 and the part information is allocated, this program notifies the conveyor station of the processing that has been performed for the part.

Argument 1:

Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.

Argument 2:

Specify one of the following values to indicate the processing result.

1: "Success"
Specify this value when the part has been handled normally (e.g., picked or placed in a cell of the tray).

2: "Return"
Specify this value when the part is returned to the conveyor station before being handled.

3: "Skip"
Specify this value when the part has not been handled as scheduled. This applies, for example, when the handling of the part is canceled intentionally after the position or model ID of the allocated part is evaluated by the TP program of the robot. In this case, the information of this part is passed to the next conveyor station.

4: "Remove"
Specify this value the handling of the part fails. This applies, for example, when the pickup check is unacceptable for some reason such as because the part drops after being picked. In this case, since the robot has moved the part, a downstream robot cannot handle it properly even if the information of that part is returned to the conveyor station. Therefore, the part information is deleted.

For cells in the tray, all other pending cells in the same tray are also be deleted.

If the stop/start conveyor function is enabled during the conveyor station setup and the conveyor is stopped because the part fails to be handled in time, the conveyor starts to move after PKCSACKQUE is executed with the processing result set to other than 2.

Example:

When the ID of conveyor station 1 is stored in R[11], the following instruction notifies that the processing result is "Success".

```
CALL PKCSACKQUE (CStn ID=R[11:CSTN1 ID],Success)
```

6.8.5.6 PKCSPUTQUE.PC

This program inputs part information to a conveyor station. Normally, it is not used. Use the program when you want to input part information to a conveyor station by using a unique program without the use of the sensor function provided by *i*RPickTool.

Argument 1:

Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.

Argument 2:

Specify the number of the Vision Register storing the part information.

Argument 3:

Specify the register number for storing the result of the duplicate check. If the information is input with no duplicate parts, 0 is stored in the specified register. If there are any duplicate parts and the information fails to be input, 1 is stored.

If any of these arguments is invalid, "P_CSPUTQUE error (error number)" may be displayed in the status window during the program execution, potentially ending the program. The causes of these messages and the actions to be taken are as follows.

P_CSPUTQUE error 17030:

A nonexistent register number is specified. If a register is given in the argument or indirectly specified, check that an existing register number is set in the corresponding register.

Example:

When the ID of conveyor station 1 is stored in R[11], the following instruction put a part information that is stored in VR[3] in the conveyor station 1 and the processing status in R[21].

```
CALL PKCSPUTQUE (CStn ID=R[11:CSTN1 ID],Offset VR=3,OverlapStat Reg=21)
```

6.8.5.7 PKCSGETTIME.PC

This program returns the estimated time required before a part to be picked up according to a specified order arrives at the upstream boundary of the tracking area. The estimated time is expressed in milliseconds. To estimate the time, the conveyor speed at the time when this function is called is used. If the part has not yet passed the upstream boundary of the tracking area, a positive value is stored. If it has passed the boundary, a negative value is stored. When the conveyor is stopped, the program returns 2147483646 if the part is upstream of the upstream boundary of the tracking area. The program returns -2147483647 if the part is downstream of the upstream boundary of the tracking area. In addition, the program also obtains information on the part to be picked up according to the specified order.

This KAREL program cannot be used from the time PKCSGETQUE returns 0 as the processing status until PKCSACKQUE is called. Calling the program during this period causes the error "No ACKQUE after the last GETQUE". This KAREL program does not support circular conveyors.

NOTE
PKCSGETTIME.PC cannot be used in the following cases.
- "Circle" is set in [Tracking Type] for the conveyor
- [Clearance check] is used for the conveyor

Argument 1:
Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.

Argument 2:
Specify the order. For example, if you specify 3, the program returns the estimated time for the part whose information is to be acquired by the third subsequent PKCSGETQUE program. Since it is based on the queue information available when this program is executed, the order may change if the robot misses a part or detects a new one.

Argument 3:
Normally, specify 1. When a robot gripper capable of holding multiple parts is used and you want the robot to consecutively pick as many parts as the number of parts that its gripper can hold, specify which part to pick next. For example, if the gripper can hold four parts and it is now holding two parts, specify 3.

Argument 4:
Normally, specify 1. When a robot gripper capable of holding multiple parts is used and you want the robot to consecutively pick as many parts as the number of parts that its gripper can hold, specify the number of parts that the gripper can hold. For example, if the gripper can hold four parts, specify 4.

Argument 5:
Specify the number of the register to store the estimated time.

Argument 6:
Specify the number of the Vision Register to store the part information.

Argument 7:
Specify the number of the register to store the processing status. The following statuses may be stored.
0: The part arrival time is estimated while the conveyor moves. The part information is stored in the specified Vision Register.
1: The conveyor is stopped and the part is downstream of the upstream boundary of the tracking area. The program returns -2147483647 as the estimated part arrival time, and the part information is stored in the specified Vision Register.
2: The conveyor is stopped and the part is upstream of the upstream boundary of the tracking area. The program returns 2147483646 as the estimated part arrival time, and the part information is stored in the specified Vision Register.
3: No part that matches the specified condition exists in the queue. No value is written to the register to store the estimated time or the Vision Register to store the part information.

Argument 8 (optional):
If you want to know the estimated time for a part of a specific model ID, specify that model ID. This argument is optional and normally does not need to be specified.

If any of these arguments is invalid, "pkcsgettime error (error number)" may be displayed in the status window during the program execution, potentially ending the program. The causes of these messages and the actions to be taken are as follows.

pkcsgettime error 17030:

A nonexistent register number is specified. If a register is given in the argument or indirectly specified, check that an existing register number is set in the corresponding register.

Example:

When the ID of conveyor station 1 is stored in R[11] and a robot gripper that can hold four parts is holding only one part (i.e., the robot is about to pick the second part next), the following instruction returns the estimated arrival time for the third part to R[13], the part information to Vision Register [5], and the processing status to R[12].

```
CALL PKCSGETTIME (CStn ID=R[11:CSTN1 ID],Order=3,Concec Flag=2,Quantity=4,
Time Reg=13,Offset VR=5,Stat Reg=12)
```

6.8.5.8 PKCSGETDIST.PC

This program returns the distance from the part to be picked up according to a specified order to the upstream boundary of the tracking area. The distance is expressed in millimeters. In addition, the program also obtains information on the part to be picked up according to the specified order.
This KAREL program cannot be used from the time PKCSGETQUE returns 0 as the processing status until PKCSACKQUE is called. Calling the program during this period causes the error "No ACKQUE after the last GETQUE". This KAREL program does not support circular conveyors.

NOTE

PKCSGETDIST.PC cannot be used in the following cases.

- "Circle" is set in [Tracking Type] for the conveyor
- [Clearance check] is used for the conveyor

Argument 1:

Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.

Argument 2:

Specify the order. For example, if you specify 3, the program returns the distance for the part whose information is to be acquired by the third subsequent PKCSGETQUE program. Since it is based on the queue information available when this program is executed, the order may change if the robot misses a part or detects a new one.

Argument 3:

Normally, specify 1. When a robot gripper capable of holding multiple parts is used and you want the robot to consecutively pick as many parts as the number of parts that its gripper can hold, specify which part to pick next. For example, if the gripper can hold four parts and it is now holding two parts, specify 3.

Argument 4:

Normally, specify 1. When a robot gripper capable of holding multiple parts is used and you want the robot to consecutively pick as many parts as the number of parts that its gripper can hold, specify the number of parts that the gripper can hold. For example, if the gripper can hold four parts, specify 4.

Argument 5:

Specify the number of the register to store the distance to the upstream boundary of the tracking area.

In the case of upstream than upstream boundary of the tracking area, the sign is positive. In the case of downstream than upstream boundary of the tracking area, the sign is negative.

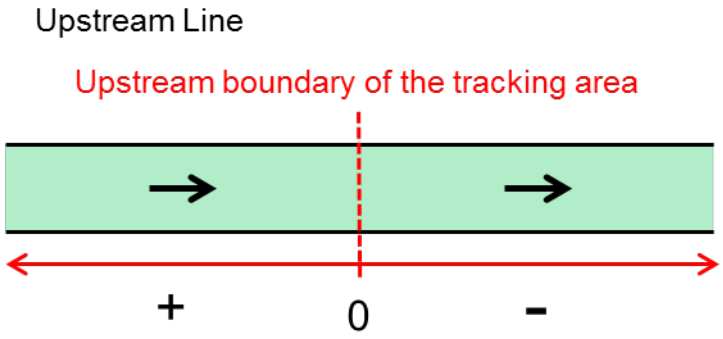


Fig. 6.8.5.8(a)

Argument 6:

Specify the number of the Vision Register to store the part information.

Argument 7:

Specify the number of the register to store the processing status. The following statuses may be stored.

0: A part that matches the specified condition exists in the queue. The distance to the upstream boundary of the tracking area is stored in the specified register, and the part information is stored in the specified Vision Register.

1: No part that matches the specified condition exists in the queue. No value is written to the register to store the distance or the Vision Register to store the part information

Argument 8 (optional):

If you want to know the distance for a part of a specific model ID, specify that model ID. This argument is optional and normally does not need to be specified.

If any of these arguments is invalid, "'pkcsgetdist error (error number)" may be displayed in the status window during the program execution, potentially ending the program. The causes of these messages and the actions to be taken are as follows.

pkcsgetdist error 17030:

A nonexistent register number is specified. If a register is given in the argument or indirectly specified, check that an existing register number is set in the corresponding register.

Example:

When the ID of conveyor station 1 is stored in R[11] and a robot gripper that can hold four parts is holding only one part (i.e., the robot is about to pick the second part next), the following instruction returns the distance from the third part to the upstream boundary of the tracking area to R[15], the part information to Vision Register [6], and the processing status to R[14].

```
CALL PKCSGETDIST (CStn ID=R[11:CSTN1 ID],Order=3,Concec Flag=2,Quantity=4,
Time Reg=15,Offset VR=6,Stat Reg=14)
```

6.8.5.9 PKCSGETPFRT.PC

This program returns the number of parts to be processed per minute immediately after PKCSGETPFRT is called for a specified conveyor station (expected part flow rate).
The expected part flow rate is calculated as follows.
Expected part flow rate (parts/min) = Expected number of parts (number) x conveyor speed (mm/s) x 60 (s/min) ÷ distance to the conveyor station (mm)
Here, the "expected number of parts" and "distance to the conveyor station" are calculated as follows.

For each conveyor station, the range to be used for the calculation is defined. The expected number of parts is calculated by using only those parts located within this range. The downstream boundary of the range is the discard line of the corresponding conveyor station. The upstream boundary of the range is one of the following three positions, depending on the conveyor station.

1 In the case of the most upstream conveyor station, the upstream boundary is the most upstream detection position of any of the parts held by the conveyor station when PKCSGETPFRT is called for the first time with the conveyor station holding one or more parts.
2 If the conveyor station is not the most upstream one and [Y Sort] is disabled, the upstream boundary is the discard line of the preceding conveyor station.
3 If the conveyor station is not the most upstream one and [Y Sort] is enabled, the upstream boundary is the position offset downstream by [X Tolerance] set in [Y Sort] from the discard line of the preceding conveyor station.

Expected number of parts

Of those parts located within the range used for the calculation, the expected number of parts refers to the number of parts that are expected to be processed by the conveyor station based on the load balance setting of the conveyor, the setting of [Y Sort] of the conveyor station, the layer and initial status settings of the tray, and the current status. The value is calculated not based on the conveyance capacity of the robot, but from the proportion that should be processed by the relevant robot based on the load balance setting and actual supply amount. In reality, however, the robot may not be able to process the full number of workpieces calculated as the expected number. For example, when a load balance of 50% is set for the most upstream conveyor station, four of seven preceding parts have been picked, and three parts are located within the range used for the calculation, then the expected number of parts is 1 ((7 + 3) ÷ 2 – 4).

Distance to the conveyor station

The distance between the upstream and downstream boundaries of the range used for the calculation of the expected number of parts is defined as the distance to the conveyor station.

NOTE
PKCSGETPFRT.PC cannot be used in the following cases.
- "Circle" is set in [Tracking Type] for the conveyor
- [Clearance check] is used for the conveyor

Argument 1:

Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.

Argument 2:

Specify the number of the register to store the expected part flow rate.

If any of these arguments is invalid, "pkcsgetpfrt error (error number)" may be displayed in the status window during the program execution, potentially ending the program. The causes of these messages and the actions to be taken are as follows.

pkcsgetpfrt error 17030:

A nonexistent register number is specified. If a register is given in the argument or indirectly specified, check that an existing register number is set in the corresponding register.

Example:

When the ID of conveyor station 1 is stored in R[11], the following instruction returns the expected part flow rate to R[12].

```
CALL PKCSGETPFRT (CStn ID=R[11:CSTN1 ID],PrtFlowRate Reg=12)
```

6.8.5.10 PKCSGETNPRT.PC

This program returns the number of parts that are held by a specified conveyor station.

Argument 1:

Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.

Argument 2:

Specify the number of the register to store the number of parts held by the conveyor station.

Example:

The following instruction returns the number of parts held by conveyor station 1 to R[13].

```
CALL PKCSGETNPRT (CStn ID=R[11:CSTN1 ID],NumParts Reg=13)
```

6.8.5.11 PKCSSETDSLIN.PC

This program changes a discard line of the specified conveyor station. This setting is reflected soon. When there are multiple robots, if you execute this in one robot, the setting is reflected to all robots.

Argument 1:

Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.

Argument 2:

Specify the discard line value.

Example:

The following instruction changes a discard line of the conveyor station 1 to -200.

```
CALL PKCSSETDSLIN (CStn ID=R[11:CSTN1 ID],Discard Line=-200)
```

⚠ CAUTION

1 Please call this program in top of the main program, that is before PKWCSTART or PKSNSTART, so that the changed parameters are reflected correctly.
2 When you execute this program during opening the setup page, the parameter to be changed might not be updated soon. But they are updated automatically after a few seconds.
3 When you use multiple controllers and call KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC), be careful about the following points.
   - Call the listed KAREL programs for setting parameters before PKWCSTART or PKWCCHECK
   - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all controllers in order to wait for the completion of the parameter changes in all controllers. Set the wait time to 5 seconds or so. If parameter changes have not completed when PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
   - If you call the multiple listed KAREL programs for setting parameters, call them in one controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other controllers also. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCCHECK is called.

6.8.5.12 PKCSGETENC.PC

This program returns the encoder ID which is used in the specified conveyor station.

Argument 1:
Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.

Argument 2:
Specify the number of the register to store the encoder ID.

Example:
The following instruction returns the encoder ID used in the conveyor station 1 to R[7].

```
CALL PKCSGETENC(CStn ID=1, Enc Num Reg=7)
```

6.8.5.13 PKCSGETTRID.PC

This program returns the tray ID which is used in the specified conveyor station.

Argument 1:
Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.

Argument 2:
Specify the number of the register to store the tray ID.

Example:
The following instruction returns the tray ID used in the conveyor station 1 to R[7].

```
CALL PKCSGETTRID(CStn ID=1,Tray ID Reg=7)
```

6.8.5.14 PKCSCLRACK.PC

This program clears any pending ACK for a conveyor station. Sometimes when you abort your robot program, a PKCSGETQUE call may have been made but not the PKCSACKQUE call. In this case, if you restart your main program, then if a PKCSGETQUE call is made prior to handling the pending PKCSACKQUE call, then you will get a "No Getque before Ackque" error. However, if you use the PKCSCLRACK macro prior to the PKCSGETQUE call, then there will be no error. However, you should note that the part that was gotten by the PKCSGETQUE call prior to the Abort will be permanently lost by the system.

Argument 1:
Specify an ID of a conveyor station.

Example:
To clear any pending acks for conveyor station whose ID is 1, use the following instruction:

```
CALL PKCSCLRACK(CStn ID=1)
```

6.8.6 Fixed Station-Related Programs

The two letters following [PK] are [FS]; namely, the names of these programs begin with these four letters, [PKFS].

6.8.6.1 PKFSGETID.PC

This program gets the ID of a specified fixed station.

Argument 1:
Specify a fixed station name.

Argument 2:
Specify the number of the register to store the ID of the fixed station.

If any of these arguments is invalid, one of the messages shown below may be displayed in the status window during the program execution, potentially ending the program. The causes of these messages and the actions to be taken are as follows.

Cannot Find Fixed Station:
No fixed station having the specified name exists.

Internal error 17030:
A nonexistent register number is specified. If a register is given in the argument or indirectly specified, check that an existing register number is set in the corresponding register.

Example:
The following instruction stores the ID of fixed station 1 in R[12].

```
CALL PKFSGETID ('FSTN1',FStn ID Reg=12)
```

6.8.6.2 PKFSPUTQUE.PC

This program places a new tray on a specified fixed station.
To specify the initial status of the placed tray (“Empty” or “Full”), use the fixed station setup screen.
To put another tray after all the cells in the placed tray are processed, call this KAREL program again.

Argument 1:
Specify the ID of the fixed station. The ID can be acquired by PKFSGETID described earlier.

Example:
When the ID of fixed station 1 is stored in R[12], the following instruction put a tray in the fixed station 1.

```
CALL PKFSPUTQUE (FStn ID=R[12:FSTN1 ID])
```

6.8.6.3 PKFSCLRQUE.PC

This program clears the queue information of the tray placed on a specified fixed station.

Argument 1:
Specify the ID of the fixed station. The ID can be acquired by PKFSGETID described earlier.

Example:
The following instruction clears the information from fixed station 1. The ID is stored in R[12].

```
CALL PKFSCLRQUE (FStn ID=R[12:FSTN1 ID])
```

6.8.6.4 PKFSGETQUE.PC

This program gets the information about the unprocessed cells in the tray placed on specified fixed station.
If the initial status of the tray is “Empty”, the robot places a part in each cell whose information is acquired.
If it is “Full”, the robot picks the part from each cell.
The program gets the information about the unprocessed cells according to the following order of priority:

- Cells in a tray placed by PKFSPUTQUE
- Cell having the smallest layer number - namely, the cell in the bottom layer - if the initial status of the tray is “Empty”
  Cell having the largest layer number of the cells containing parts - namely, the cell in the top layer - if the initial status of the tray is “Full”
- Cell having a specified model ID if any is specified
- Cell having the smallest cell number

Argument 1:
Specify the ID of the fixed station. The ID can be acquired by PKFSGETID described earlier.

Argument 2:
Specify the number of the Vision Register to store the vision offset.

Argument 3:
Specify the number of the register to store the processing status.
0: Success
1: Failure (The queue contains no cell that can be processed.)

Argument 4 (optional):
Specify a model ID.

If any of these arguments is invalid, "P_FSGETQUE error (error number)" may be displayed in the status window during the program execution, potentially ending the program. The causes of these messages and the actions to be taken are as follows.

P_FSGETQUE error 17030:
A nonexistent register number is specified. If a register is given in the argument or indirectly specified, check that an existing register number is set in the corresponding register.

Example:
The following instruction stores the information about the unprocessed cell from fixed station 1 in VR[1] and the processing status in R[2]. The ID is stored in R[12].

```
CALL PKFSGETQUE (FStn ID=R[12:FSTN1 ID],Offset VR=1,Stat Reg=2)
```

6

⚠ CAUTION
1 The vision offset of each cell of a tray is calculated based on the relative position between this cell and the cell 1. So when you teach the robot position to a tray, you must make the taught position conform to the position of the cell 1.

6.8.6.5 PKFSACKQUE.PC

After the processing status of PKFSGETQUE becomes 0 and the unprocessed cell information is allocated, this program notifies the fixed station of the processing that has been performed for the cell.

Argument 1:
Specify the ID of the fixed station. The ID can be acquired by PKFSGETID described earlier.

Argument 2:
Specify one of the following values to indicate the processing result.
1: "Success"
Specify this value when the cell is processed normally. After the notification, this cell is regarded as processed and never acquired again by PKFSGETQUE.
2: "Return"
Specify this value when the cell fails to be processed normally. After the notification, this cell is returned to the queue and can be acquired again by PKFSGETQUE.

Example:
The following instruction notifies that the processing result is "Success". The ID is stored in R[12].

```
CALL PKFSACKQUE (FStn ID=R[12:FSTN1 ID],Success)
```

6.8.6.6 PKFSPUTBUF.PC

This KAREL program is used only when a tray is used as a buffer.
It places a new buffer on a specified fixed station.
To specify the initial status of the placed buffer ("Empty" or "Full"), use the fixed station setup screen.
When a tray is used as a buffer, the robot can pick a part once placed in a cell again.

Argument 1:
Specify the ID of the fixed station. The ID can be acquired by PKFSGETID described earlier.

Argument 2:

Specify an identification number to distinguish each buffer. As an identification number, 1 or a larger number can be specified. If you call this KAREL program again with the same identification number specified in it, it clears the previous buffer and places a new buffer on the fixed station.
To place a new buffer in addition to the previous one, specify an identification number that is different from that of the previous buffer.

Example:

When the ID of fixed station 1 is stored in R[12], the following instruction put a buffer in the fixed station 1. The identification number of this buffer is 1.

```
CALL PKFSPUTBUF (FStn ID=R[12:FSTN1 ID],Index=1)
```

6.8.6.7 PKFSCLRBUF.PC

This KAREL program is used only when a tray is used as a buffer.
It clears the queue information of the buffer placed on a specified fixed station.

Argument 1:

Specify the ID of the fixed station. The ID can be acquired by PKFSGETID described earlier.

Argument 2:

Specify the identification number of the buffer. This must be the identification number specified by PKFSPUTBUF.

Example:

The following instruction clears the information of the buffer whose identification number is 1 from fixed station 1. The ID is stored in R[12].

```
CALL PKFSCLRBUF (FStn ID=R[12:FSTN1 ID],Index=1)
```

6.8.6.8 PKFSGETBUF.PC

This KAREL program is used only when a tray is used as a buffer.
It gets the target (empty cell information or information of the part placed in a cell) from the buffer placed on a specified fixed station. When a tray is used as a buffer, you need to specify the target you want to get, because the robot can either place a part in an empty cell or pick a part from a cell.
The program gets the information about the target according to the following order of priority:

- Cell in the buffer having the specified identification number
- Cell having the smallest layer number of the empty cells - namely, the cell in the bottom layer - if the target is a cell
  Part having the largest layer number of the cells containing parts - namely, the part in the top layer - if the target is a part
- Cell having a specified model ID if any is specified. A cell is selected based on the model ID even if the target is a part.
- Cell having the smallest cell number if the target is a cell
  When FIFO is used, First part that is placed in the cell, if the target is a part
  When LIFO is used, Last part that is placed in the cell, if the target is a part

Argument 1:

Specify the ID of the fixed station. The ID can be acquired by PKFSGETID described earlier.

Argument 2:

Specify the identification number of the buffer. You need to specify 1 or a greater value. This must be the identification number specified by PKFSPUTBUF.

Argument 3:

Specify the target.
1: Cell
2: Part (placed in a cell)

Argument 4:

Specify the number of the Vision Register to store the vision offset. If the target is a part and the detection information is stored in a buffer, the part detection information is stored as the measurement value, model ID, detection position, and encoder value of the Vision Register.

Argument 5:

Specify the number of the register to store the processing status.
0: Success
1: Failure (The queue contains no target that can be processed.)

Argument 6 (optional):

Specify a model ID.

If any of these arguments is invalid, "P_FSGETBUF error (error number)" may be displayed in the status window during the program execution, potentially ending the program. The causes of these messages and the actions to be taken are as follows.

P_FSGETBUF error 17030:

A nonexistent register number is specified. If a register is given in the argument or indirectly specified, check that an existing register number is set in the corresponding register.

Example:

The following instruction stores the information about the unprocessed cell of the buffer whose identification number is 1 from fixed station 1 in VR[1] and the processing status in R[2]. The ID is stored in R[12].

```
CALL PKFSGETBUF (FStn ID=R[12:FSTN1 ID],Index=1,Cell,Offset VR=1,Stat Reg=2)
```

⚠ CAUTION

The vision offset of each cell of a tray is calculated based on the relative position between this cell and the cell 1. So when you teach the robot position to a tray (buffer), you must make the taught position conform to the position of the cell 1.

6.8.6.9 PKFSACKBUF.PC

This KAREL program is used only when a tray is used as a buffer.
After the processing status of PKFSGETBUF becomes 0 and the target information is allocated, this program notifies the fixed station of the processing that has been performed for the target.

Argument 1:

Specify the ID of the fixed station. The ID can be acquired by PKFSGETID described earlier.

Argument 2:

Specify one of the following values to indicate the processing result.

1: "Success"
Specify this value when the cell is processed normally. After the notification, the status of this target changes as follows.
If the target is a cell, the cell changes to the status in which it has a part placed in it. In other words, when a part is specified as the target in PKFSGETBUF after the notification, the information of the cell containing this part can be acquired.
If the target is a part, the cell containing a part changes to the empty status. In other words, when a cell is specified as the target in PKFSGETBUF after the notification, the information of this empty cell can be acquired.
2: "Return"
Specify this value when the cell fails to be processed normally. After the notification, this target is returned to the queue and can be acquired again by PKFSGETBUF.

Argument 3: (optional)
When the part detection information is stored in the buffer, specify the number of the Vision Register storing the detection information. The information that is stored in the buffer here can be acquired again by calling PKFSGETBUF later with a part specified as the target. The information stored in the buffer is the measurement value, model ID, detection position, and encoder value.

Example:
The following instruction notifies that the processing result is "Success". The ID is stored in R[12].

```
CALL PKFSACKBUF (FStn ID=R[12:FSTN1 ID],Success)
```

6.8.6.10 PKFSGETTRID.PC

This program gets the ID of the tray that you assigned to a Fixed Station in the *i*RPickTool treeview.

Argument 1:
Specify an ID of a fixed station.

Argument 2:
Specify the Numeric Register in which to return the "Tray ID".

Example:
To get the "Tray ID" of a fixed station whose ID is 1 in Numeric Register 7, use the following instruction:

```
CALL PKFSGETTRID(FStn ID=1,Tray ID Reg=7)
```

6.8.6.11 PKFSGETUF.PC

This program gets the UFrame (UF) number that you assigned to a Fixed Station Origin in the *i*RPickTool treeview.

Argument 1:
Specify an ID of a fixed station.

Argument 2:
Specify the Numeric Register in which to return the "Uframe number" of the Fixed Station Origin.

Example:

To get the "Uframe num" of a fixed station whose ID is 1 in Numeric Register 7, use the following instruction:

```
CALL PKFSGETTUF(FStn ID=1,UFrameNum Reg=7)
```

6.8.6.12 PKFSCLRACK.PC

This program clears any pending ACK for a fixed conveyor station. Sometimes when you abort your robot program, a PKFSGETQUE (or PKFSGETBUF) call may have been made but not the PKFSACKQUE (or PKFSACKBUF) call. In this case, if you restart your main program, then if a PKFSGETQUE (or PKFSGETBUF) call is made prior to handling the pending PKFSACKQUE (or PKFSACKBUF) call, then you will get the alarm IRPK-051: "No ACKQUE after the last GETQUE." However, if you execute PKFSCLRACK prior to the PKFSGETQUE (or PKFSGETBUF) call, then there will be no error. However, note that the system will permanently lose the part that was gotten by the PKFSGETQUE (or PKFSGETBUF) call prior to the Abort.

Argument 1:
Specify an ID of a fixed conveyor station.

Example:
To clear any pending ACKs for a fixed conveyor station whose ID is 1, use the following instruction:

```
CALL PKFSCLRACK(FStn ID=1)
```

6.8.7 Tray-Related Programs

The two letters following [PK] are [TR]; namely, the names of these programs begin with these four letters, [PKTR].

6.8.7.1 PKTRSETCLPOS.PC

This program changes a pattern in a tray. When there are multiple robots, if you execute this in one robot, the setting is reflected to all robots. This program can be used in 7DC3/02(V8.30P/02) or later.

Argument 1:
Specify the name of the tray using a character string.

Argument 2:
Specify the cell number from 1 to 160.

Argument 3:
Specify the target parameter number of the below.
1 : X
2 : Y
3 : Z
4 : R
5 : Model ID
6 : Layer Number

> ⚠ **CAUTION**
> If the layer number is 1 or more, the cell is handled. So if you want to add new cell, you specify 1 or more to the layer number. But you have to add a cell from the last cell number + 1.
> On the other hand, if you want to delete a cell, you specify 0 to the layer number. But you have to delete a cell from the last cell. If you want to delete all cells, you can delete from the first cell.

Argument 4:

Specify the value of the target parameter.

Example:

The following instruction changes “X” of Cell 1 in “TRAY1” to 33.3.

```
CALL PKTRSETCLPOS ('TRAY1',Cell num=1,X,Value=33.3)
```

The following instruction adds cell 2 to layer 1.

```
CALL PKTRSETCLPOS('TRAY1',Cell num=2,Layer num,Value=1)
```

The following instruction deletes cell 2 from layer 1.

```
CALL PKTRSETCLPOS('TRAY1',Cell num=2,Layer num,Value=0)
```

> ⚠ **CAUTION**
> 1 Please call this program in top of the main program, that is before PKWCSTART or PKSNSTART, so that the changed parameters are reflected correctly.
> 2 When you execute this program during opening the setup page, the parameter to be changed might not be updated soon. But they are updated automatically after a few seconds.
> 3 When you use multiple controllers and call KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC), be careful about the following points.
> - Call the listed KAREL programs for setting parameters before PKWCSTART or PKWCCHECK
> - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all controllers in order to wait for the completion of the parameter changes in all controllers. Set the wait time to 5 seconds or so. If parameter changes have not completed when PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
> - If you call the multiple listed KAREL programs for setting parameters, call them in one controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other controllers also. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCCHECK is called.

6.8.7.2 PKTRGETNUMCL.PC

This program returns the number of cells in the specified layer of the specified tray.

Argument 1:
Specify the ID of a tray. The ID of the tray which is used in a conveyor can be acquired by PKCSGETTRID described earlier.

Argument 2:
Specify the layer number. If you specify 0, this program returns the total number of cells in all layers of the tray.

Argument 3:
Specify the number of the register to store the number of cells.

Example:
The following instruction returns the total number of cells in the tray to R[7].

```
CALL PKTRGETNUMCL(Tray ID=1,Layer num=0,NumCells Reg=7)
```

6.9 ROBOT PROGRAMS

Robot programs create programs that match the system. This section contains examples of the basic programs and describes the concepts and other details of these programs. Tailor the sample programs provided herein to suit your system.

Subsection 6.9.1, "Tracking Program" explains the details of tracking programs. Programs that perform pickup and placement motions on a conveyor need to be created as tracking programs. Tracking programs require different settings from those of ordinary programs. Subsections 6.9.2 and later explain about sample programs for each configuration.

Note that this section mainly discusses only the creation of the program logic. For information about the actual task of robot position teaching, see Section 6.11, "TEACHING THE POSITION TO ROBOTS" later in this chapter.

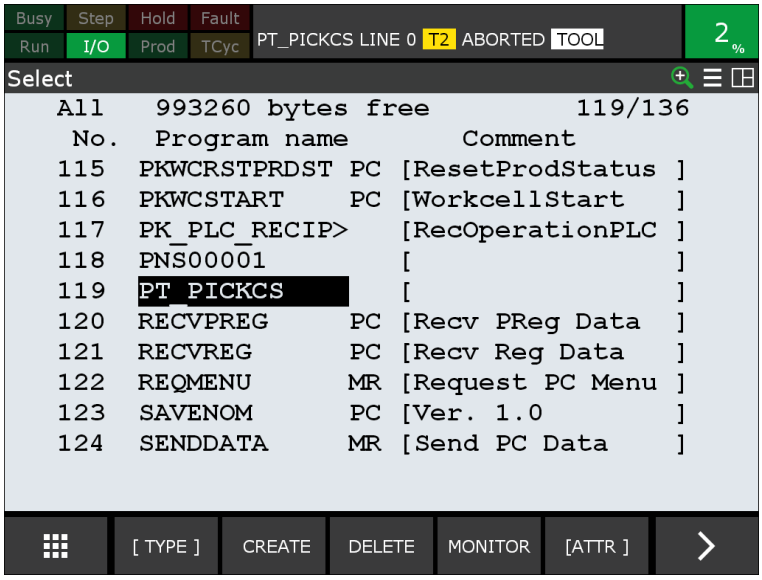
> ⚠ **CAUTION**
> By default, *i*RPickTool has hot start disabled.

6.9.1 Tracking Program

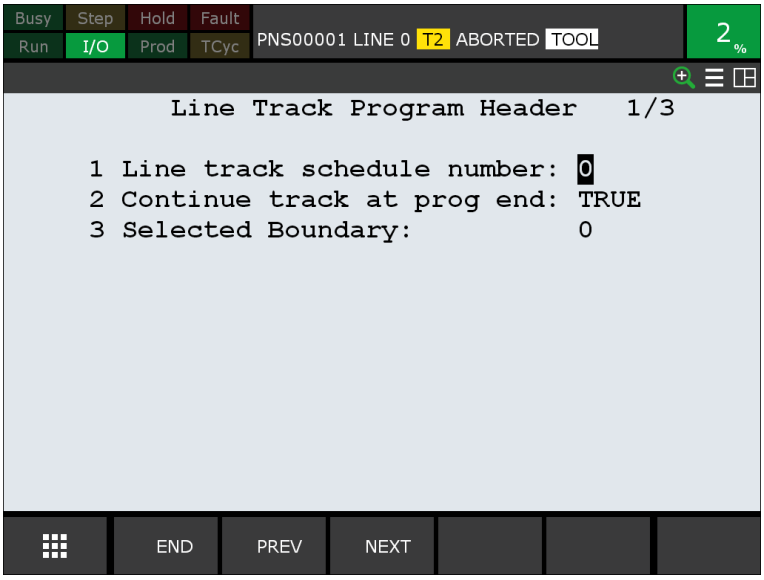
A program that causes the robot to follow the motion of the conveyor is called a tracking program. Unlike ordinary programs, a tracking program requires that a tracking schedule number and other data be set in advance using the program detail screen.

The setting procedure is as follows.

1 Press the [Select] key on the teach pendant.
A program list screen will appear.



2 From the list, select a tracking program.
3 Press [> (next page)], and then press F2 [DETAIL].
4 Press F3 [NEXT].
A screen as shown below is displayed.



5 In [Line track schedule number], set the tracking schedule number specified for the conveyor station.
6 In [Continue track at prog end], set [TRUE].

NOTE
If you set [FALSE] in [Continue track at prog end], the robot stops the tracking motion when the tracking program ends. As a result, the robot cannot move smoothly, even if it resumes tracking immediately when the next tracking program starts. A visual tracking system requires the robot to continue to move fast. Therefore, set [TRUE] in [Continue track at prog end] to ensure smooth robot motion.

7 Leave the value in [Selected Boundary] unchanged from 0. If 0 is set, the tracking area specified for the conveyor station is used.

6.9.2 Sample Programs for the Conveyor-to-Conveyor Configuration

This subsection contains a set of sample programs for a robot that picks parts from an infeed conveyor and places them onto an outfeed conveyor. Tailor the sample programs presented herein to suit your system.

Generally, a tracking system uses the following three types of robot programs:

Main program
Program that performs pickup motions on the infeed conveyor (pick program)
Program that performs placement motions on the outfeed conveyor (placement program)

The pick program and placement program are called from the main program. When the robot picks parts from a conveyor and places them on a table, only the pick program is used as a tracking program. When the robot picks parts from one conveyor and places them on another conveyor, both the pick program and placement program are used as tracking programs.

The sample programs presented herein use the registers and Position Registers listed below. Change the register numbers as necessary.

Table 6.9.2(a) Registers

R[1]	Cycle stop flag. Setting 1 in this register causes the system to stop the cycle.
R[2]	PKCSGETQUE status
R[3]	Robot gripper zone* counter
R[4]	Number of gripper zones of the robot. When using a hand that can hold one part, set 1 in advance; when using a hand that can hold N parts, set N in advance.
R[11]	Stores the conveyor station ID of infeed conveyor CONV1.
R[12]	Stores the conveyor station ID of outfeed conveyor CONV2.

* A gripper zone refers to that part of the robot that picks parts. For example, when the robot hand has four grippers and each of the grippers picks one part at a time, then the number of gripper zones is 4.

Table 6.9.2(b) Position registers

PR[1 to 9]	Pickup position for the Nth gripper zone
PR[10]	Tool offset to create an approach point for the pickup motion
PR[11]	Placement position on the outfeed conveyor
PR[12]	Tool offset to create an approach point for the placement motion

6.9.2.1 Main program

This is the main program of the system.

Setup example for a main program

MAIN1.TP

```
  1:    R[1:CYCLE_STOP]=0
  2:    CALL PKCSGETID('CONV1',CStn ID Reg=11)
  3:    CALL PKCSGETID('CONV2',CStn ID Reg=12)
  4:
  5:    UTOOL_NUM=1
  6:    UFRAME_NUM=0
  7: J P[1] 50% FINE
  8:
  9:    CALL PKWCSTART
 10:
```

```
 11:    LBL[100]
 12:    CALL PICK1
 13:    IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 14:    CALL DROP1
 15:    IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 16:    JMP LBL[100]
 17:
 18:    LBL[900]
 19:    CALL PKWCEND
[End]
```

PICK1.TP is called on line 12 to pick a part, and DROP1.TP is called on line 14 to place the part. The main program mainly repeats the PICK1.TP and DROP1.TP calls alternately. When a hand that can hold multiple parts is used, PICK1.TP first picks all parts and then DROP1.TP places them at once.

R[1] serves as a cycle stop flag. The cycle stop flag is checked between a pickup motion and a placement motion and, if the flag is set to 1, the cycle is stopped.

PKCSGETID is called on lines 2 and 3 to store the IDs of the conveyor station in R[11] and R[12]. These are used in PICK1.TP and DROP1.TP, respectively.

PKWCSTART is called on line 9 to initialize the conveyor station and start a sensor task. If the cycle is stopped, PKWCEND is called on line 19 to stop the sensor task.
The initialization of the conveyor station and the startup of a sensor task can be performed separately by using PKCSCLRQUE and PKSNSTART. To do so, replace line 9 with those shown below.

```
        CALL PKWCCHECK
        CALL PKCSCLRQUE(CStn ID=R[11:CSTN1])
        CALL PKCSCLRQUE(CStn ID=R[12:CSTN2])
        CALL PKSNSTART
```

First, PKWCCHECK is called to check for any invalid *i*RPickTool setting. Next, PKCSCLRQUE is called to initialize the conveyor station. The registers storing the conveyor station IDs acquired by PKCSGETID are used. The conveyor station is initialized before PKSNSTART starts the sensor.

In this sample program, if the cycle stop flag is set to 1 when line 13 is reached, the cycle is stopped immediately, regardless of whether the robot is holding a part or not. If you want the cycle to be stopped after the robot places the part it is holding, modify line 13 as follows.

```
 13:    IF R[1:CYCLE_STOP]=1 AND R[3:ZONE]=1,JMP LBL[900]
```

The pick program described later, PICK1.TP, increments the value of R[3] each time the robot picks a part. Therefore, when the robot is holding a part, 2 or a greater value is set in R[3]. This is used to determine whether the robot is holding a part or not.

When multiple robots are used

When multiple robots are used, MAIN1.TP, PICK1.TP, and DROP1.TP can be used for all the robots as they are. Note, however, that the tracking schedule number in each tracking program needs to be changed. If PKWCSTART or PKSNSTART is executed for a robot without a sensor (not connected to a camera or photo eye), warning IRPK-031 is generated. This is a mere informational message and does not affect the operation.

6.9.2.2 Pick program

This program performs a pickup motion on an infeed conveyor.

Setup example for a pick program

PICK1.TP

```
  1:    UTOOL_NUM=1
  2:    UFRAME_NUM=0
  3:    R[3:ZONE]=1
  4:
  5:    LBL[100]
  6:    STOP_TRACKING
  7:    CALL PKCSGETQUE(CStn ID=R[11:CSTN1],Consec Flag=R[3:ZONE],
        Timeout (ms)=100, Offset VR=1,Stat Reg=2)
  8:    IF R[2:GETQ_STATUS]=0,JMP LBL[200]
  9:    IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 10:    JMP LBL[100]
 11:
 12:    LBL[200]
 13: L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
 14: L PR[R[3:ZONE]] 4000mm/sec CNT0 VOFFSET,VR[1]
 15:    CALL VACUUM_ON
 16: L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
 17:    CALL PKCSACKQUE(CStn ID=R[11:CSTN1],Success)
 18:
 19:    R[3:ZONE]=R[3:ZONE]+1
 20:    IF R[3:ZONE]<=R[4:NUM_ZONE],JMP LBL[100]
 21:    LBL[900]
[End]
```

In [Line track schedule number] of the program detail screen, set the tracking schedule number specified for the conveyor station for which this robot is selected on conveyor CONV1.
In line 1, the tool frame number is selected.
The tool frame number specifies the number of the tool frame that is set for the gripper.
As an example, suppose that the gripper tool frame has been set to number 1.
In this example, the gripper tool frame is common, and the position register is changed for each gripper zone. As an alternative way, you can do the same thing if you set a common position register and change the tool frame that is set for each gripper zone.

In line 2, the user frame number is selected.
For the user frame number, always select 0.
In a tracking program, the robot may make an unexpected motion if a user frame is specified.
(In a tracking program, when the user frame number is 0, it refers to the robot's tracking frame and not the world frame.)

PKCSGETQUE is called on line 7 to make the robot wait for up to 100 milliseconds before the part to be picked next enters the tracking area. When the part enters the tracking area, the vision offset is stored in VR[1] and 0 is stored in R[2]. After the part enters the tracking area, the program jumps to line 12 and performs a pickup motion. Then, PKCSACKQUE is called on line 17 to notify the conveyor station that the pickup motion is complete.

> ⚠ **CAUTION**
> When a hand that can hold multiple parts is used, parts are handled one by one sequentially as with PICK1.TP. Each time the status of PKCSGETQUE becomes 0, the trigger for the tracking motion is set. If you call PKCSGETQUE and PKCSACKQUE in advance as many times as the number of parts to be picked by the hand to get the information of those parts at once, the tracking motion trigger set in the last PKCSGETQUE program is used, potentially making the robot unable to pick the earlier acquired parts at the correct position.

Since the wait time is set to 100 milliseconds in the third argument of PKCSGETQUE, a timeout occurs in 100 milliseconds if there is no part to pick in the tracking area when PKCSGETQUE is called. In that case, R[2] contains a value other than 0, and PICK1.TP checks the cycle stop flag, calls PKCSGETQUE again, and waits for a part to be assigned.

> ⚠ **CAUTION**
> The robot controller that executes the *i*RPickTool programs handles the TP program and the sensor task in parallel. Therefore, if the processing load of the TP program becomes heavy, the sensor task takes longer to complete the detection process. As a result, even if a part on the conveyor has entered the tracking area, the sensor may be unable to complete the detection process for that part, preventing it from being allocated to the robot. In that case, reduce the processing load by inserting a wait instruction in the repetition processing of the TP program as appropriate.
> For example, setting the timeout time to 0 milliseconds in the third argument of PKCSGETQUE causes PKCSGETQUE to be called in rapid succession, possibly resulting in the TP program becoming busy and the sensor task taking longer to complete the detection process.

In R[4], set the number of parts to be handled by the hand in advance. When using a hand that can hold one part, set 1. When using a hand that can hold multiple parts, set the number of gripper zones of the hand.

R[3] contains the number of the gripper zone to be used for the next pickup motion. This register is initialized on line 3 and incremented by 1 on line 19 each time a part is picked. Therefore, in the case of a hand that can hold multiple parts, the value in R[3] is incremented, with 1 being set when the first part is picked, 2 being set when the second part is picked, and so on. If the value in R[3] exceeds the number specified in R[4], it indicates that as many parts as the number of gripper zones have been picked.

In the second argument of PKCSGETQUE on line 7, R[3] storing the gripper zone number is specified. While PKCSGETQUE allocates parts according to the specified load balance when 1 is set in the second argument, it allocates parts successively regardless of the load balance when 2 or a greater value is set. Therefore, in the case of a hand that can hold N parts, the robot picks N successive parts and then discards M successive parts (where M depends on the system configuration and settings).

NOTE
The number of parts to be discarded, M, is determined by the load balance. For example, in the case of a hand that can hold four parts, when parts are allocated evenly to two robots, the first robot picks four successive parts and then discards four successive parts. When parts are allocated evenly to three robots, the first robot picks four successive parts and then discards eight successive parts. Note that, even in the case of a two-robot system, the first robot picks four successive parts and then discards eight successive parts, if the load balance is set to 1:2.

⚠ CAUTION
If a part passes the "discard line" during the time from the moment the value of R[3] is incremented by 1 until PKCSGETQUE is called, its information is passed to a downstream robot even when the robot handles successive parts.

To prevent the robot from picking N successive parts even with a hand that can hold multiple parts, pass 1 as a constant to the second argument of PKCSGETQUE. How often the robot picks a part depends on the load balance. In this case, the system also exits the program when all the gripper zones are filled.

The pickup motion is performed on lines 13 to 16. The position to be taught to the robot is set in a Position Register, not in the position data of the program. This makes it easier to fine-adjust the taught position during the program execution. The Position Register is specified indirectly in R[3] storing the gripper zone number. Therefore, the robot moves to the taught position corresponding to the selected gripper zone.

"CNT0" is specified for the pickup position. This is because the robot may be unable to perform the pickup motion fast enough if "FINE" is specified for high-speed tracking. For details, see Subsection 6.12.4, "Continuous Termination Type Specification for a Taught Position". You can optimize the tracking motion further by using the linear distance specification instruction and time before instruction. For details, see Subsection 6.12.5, "Linear Distance Specification" and Subsection 6.12.8, "Time Before Instruction".

NOTE
With a tracking program, it is recommended to use only one TCP set at the mechanical interface of the wrist, regardless of whether the hand can hold one part or multiple parts. In a tracking motion, whether the target position is within the tracking area is checked based on the TCP position of the robot. If the TCP is set at a position offset from the mechanical interface of the wrist, there may be cases where the wrist is outside the tracking area even though the TCP at the target position is within the tracking area, potentially making the robot unable to reach the destination position.

The approach point and retract point for the pickup motion are determined relative to the pickup position using the tool offset that is set in a Position Register in advance, instead of teaching the positions directly. This makes it easier to fine-adjust the pickup position, approach point, and retract point.
The tracking stop instruction is called on line 6. This instruction stops the tracking motion. As mentioned in Subsection 6.9.1, "Tracking Program", the sample programs presented herein have [Continue track at prog end] enabled. This setting is intended to allow the robot to move as fast as possible when performing successive pickup motions and placement motions. If a part to pick is not conveyed for a certain period of time, however, the robot may go outside the conveyor station as it continues the previous tracking motion while waiting for a part. To prevent the robot from going outside the operating space, the

instruction stops the tracking motion. Even in this case, the robot starts the next tracking motion smoothly if a tracking trigger is set.

NOTE
When creating a pick program exclusively for a hand that can hold one part, you can make the program a little simpler than PICK1.TP. For information, read the following description of the placement program.

6.9.2.3 Placement program

This program performs a placement motion on an outfeed conveyor.

Setup example of a placement program

DROP1.TP

```
  1:    UTOOL_NUM=1
  2:    UFRAME_NUM=0
  3:
  4:    LBL[100]
  5:    STOP_TRACKING
  6:    CALL PKCSGETQUE (CStn ID=R[12:CSTN2],Consec Flag=1,Timeout (ms)=100,
        Offset VR=2,Stat Reg=2)
  7:    IF R[2:GETQ_STATUS]=0,JMP LBL[200]
  8:    IF R[1:CYCLE_STOP]=1,JMP LBL[900]
  9:    JMP LBL[100]
 10:
 11:    LBL[200]
 12: L PR[11] 4000mm/sec CNT100 VOFFSET,VR[2] Tool_Offset,PR[12]
 13: L PR[11] 4000mm/sec CNT0 VOFFSET,VR[2]
 14:    CALL VACUUM_OFF
 15: L PR[11] 4000mm/sec CNT100 VOFFSET,VR[2] Tool_Offset,PR[12]
 16:    CALL PKCSACKQUE(CStn ID=R[12:CSTN2],Success)
 17:
 18:    LBL[900]
[End]
```

This program is basically the same as the pick program.

In [Line track schedule number] of the program detail screen, set the tracking schedule number specified for the conveyor station for which this robot is selected on conveyor CONV2.

The main difference is that, because all picked parts are placed at once even when the hand can hold multiple parts, grippers are not managed using R[3] and R[4], regardless of whether the hand can hold one part or multiple parts. For the same reason, 1 is set as a constant in the second argument of PKCSGETQUE on line 6.

6.9.3 Sample Programs for the Conveyor-to-Fixed Station Configuration

This subsection contains a set of sample programs for a robot that picks parts from a conveyor and places them onto a fixed station.
The contents of these programs are similar to those used to convey parts from conveyor to conveyor. See also Subsection 6.9.2, “Sample Programs for the Conveyor-to-Conveyor Configuration”.

The sample programs presented herein use the registers and Position Registers listed below. Change the register numbers as necessary.

Table 6.9.3(a) Registers

R[1]	Cycle stop flag. Setting 1 in this register causes the system to stop the cycle.
R[2]	PKCSGETQUE status
R[5]	PKFSGETQUE status
R[6]	Stores the conveyor station ID of conveyor CONV1.
R[7]	Stores the fixed station ID of fixed station FSTN1.

Table 6.9.3(b) Position registers

PR[2]	Pickup position on the conveyor
PR[3]	Placement position on the fixed station
PR[11]	Tool offset to create an approach point for the pickup motion
PR[12]	Tool offset to create an approach point for the placement motion

6.9.3.1 Main program

This is the main program of the system.

Setup example for a main program

MAIN1.TP

```
  1:  R[1:CYCLE_STOP]=0
  2:  CALL PKCSGETID('CONV1',CStn ID Reg=6)
  3:  CALL PKFSGETID('FSTN1',FStn ID Reg=7)
  4:
  5:  !Put a tray on a FSTN
  6:  CALL PKFSPUTQUE(FStn ID=R[7:FSTN1_ID])
  7:
  8:  UTOOL_NUM=1
  9:  UFRAME_NUM=0
 10: J PR[1:Perch Pos] 50% FINE
 11:
 12:  CALL PKWCSTART
 13:
 14:  LBL[100]
 15:  CALL PICKCS
 16:  IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 17:
 18:  CALL DROPFS
 19:  IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 20:  JMP LBL[100]
 21:
 22:  LBL[900]
 23:  CALL PKWCEND
[End]
```

PICKCS.TP is called on line 13 to pick a part, and DROPFS.TP is called on line 16 to place the part. The main program mainly repeats the PICKCS.TP and DROPFS.TP calls alternately.

R[1] serves as a cycle stop flag. The cycle stop flag is checked between a pickup motion and a placement motion and, if the flag is set to 1, the cycle is stopped.

PKCSGETID is called on lines 2 and 3 to store the ID of the conveyor station in R[6] and the ID of the fixed station in R[7]. These are used in PICKCS.TP and DROPFS.TP, respectively.

PKFSPUTQUE is called on line 6 to place a tray on the fixed station.

PKWCSTART is called on line 10 to initialize the conveyor station and start a sensor task. If the cycle is stopped, PKWCEND is called on line 21 to stop the sensor task.

When multiple robots are used

In principle, the same programs may be copied for use on other robots.
Be careful about the following points.
In MAIN1.TP, change the fixed station name to that of the fixed station where the relevant robot operates.
PICKCS.TP and DROPFS.TP can be copied to other robots and used as they are. However, the tracking schedule number of the tracking program (PICKCS.TP) needs to be changed according to the conveyor station where the robot operates.
If PKWCSTART is executed for a robot without a sensor (not connected to a camera or photo eye), warning IRPK-031 is generated. This is a mere informational message and does not affect the operation.

6.9.3.2 Pick program

This program performs a pickup motion on a conveyor.

Setup example for a pick program

PICKCS.TP

```
  1:  UTOOL_NUM=1
  2:  UFRAME_NUM=0
  3:
  4:  LBL[100]
  5:  ! Get a part from a CSTN
  6:  CALL PKCSGETQUE (CStn ID=R[6:CSTN ID],Consec Flag=1,Timeout (ms)=100,
      Offset VR=1,Stat Reg=2)
  7:  IF R[1:CYCLE_STOP]=1,JMP LBL[900]
  8:  IF R[2:GETQ_STATUS]>0,JMP LBL[100]
  9:
 10:L PR[2] 10000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[11]
 11:L PR[2] 10000mm/sec CNT0 VOFFSET,VR[1]
 12:  CALL VACUUM_ON
 13:L PR[2] 10000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[11]
 14:
 15:  !Ack the part has been picked from the CSTN
 16:  CALL PKCSACKQUE(CStn ID=R[6:CSTN_ID],Success)
 17:
 18:  LBL[900]
[End]
```

In [Line track schedule number] of the program detail screen, set the tracking schedule number specified for the conveyor station for which this robot is selected on conveyor CONV1.

The contents of the program is the almost same as those of the sample program for the conveyor-to-conveyor configuration. For details, see Subsection 6.9.2.2, “Pick program”. Note that gripper zones are not used here.

6.9.3.3 Placement program

This program performs a placement motion on a fixed station.

Setup example for a placement program

DROPFS.TP

```
  1:  UTOOL_NUM=1
  2:  UFRAME_NUM=0
  3:
  4:  LBL[100]
  5:  !Check if a tray is ready
  6:  WAIT DI[21]=ON
  7:
  8:  !Get a cell from the tray
  9:  CALL PKFSGETQUE(FStn ID=R[7:FSTN1_ID],Offset VR=2,Stat Reg=5)
 10:  IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 11:
 12:  !Drop a part to the cell
 13:L PR[3] 10000mm/sec CNT100 VOFFSET,VR[2] Offset,PR[12]
 14:L PR[3] 10000mm/sec CNT0 VOFFSET,VR[2]
 15:  CALL VACUUM_OFF
 16:L PR[3] 10000mm/sec CNT100 VOFFSET,VR[2] Offset,PR[12]
 17:
 18:  !Ack the part has been dropped to the tray
 19:  CALL PKFSACKQUE(FStn ID=R[7:FSTN1_ID],Success)
 20:
 21:  !Check if the tray is full
 22:  CALL PKFSGETQUE(FStn ID=R[7:FSTN1_ID],Offset VR=2,Stat Reg=5)
 23:  IF R[5:GETQ_STATUS]>0,JMP LBL[316]
 24:
 25:  !Ack to return the cell to the tray
 26:  CALL PKFSACKQUE(FStn ID=R[7:FSTN1_ID],Return)
 27:
 28:  JMP LBL[900]
 29:
 30:  LBL[316]
 31:  !Output DO to inform that the tray is full
 32:  DO[21]=PULSE,1.0sec
 33:
 34:  !Prepare new tray
 35:  CALL PKFSPUTQUE(FStn ID=R[7:FSTN1_ID])
 36:
 37:  LBL[900]
[End]
```

A user frame number is specified on line 2. While the tracking program requires setting 0 for the user frame, a desired user frame number can be set for a fixed station.

On line 6, a check is made using the DI signal to see whether a tray has been prepared.

On line 9, a cell is picked from the tray. On line 21 and later, a check is made to see whether the tray is full and, if full, a new tray is prepared. Since the program waits for a tray to be prepared on line 6, the robot must be able to pick a part here. Therefore, the PKFSGETQUE status is not checked at this point.

On lines 13 to 16, a placement motion is performed. You can optimize the motion further by using the linear distance specification instruction, maximum linear speed function, and time before instruction. For

details, see Subsections 6.12.5 “Linear Distance Specification”, 6.12.7 “Maximum Linear Speed Function”, and 6.12.8 “Time Before Instruction”.

On lines 22 and 23, a check is made using PKFSGETQUE to see whether the tray has any empty cell. If the tray is full, the signal to prepare a new tray (DO[21] in this example) is output on lines 31 and 32 and a new tray is prepared on the fixed station using PKFSPUTQUE on line 35. If the tray has any empty cell, PKFSACKQUE returns the ACK signal on line 26 to return the acquired cell to the queue in order to put the tray in its previous status.

6.9.4 Sample Programs for the Conveyor-to-Buffer-to-Conveyor Configuration

This subsection contains a set of sample programs for a robot that uses a tray as a buffer.
The contents of these programs are similar to those used to convey parts from conveyor to conveyor. See also Subsection 6.9.2, “Sample Programs for the Conveyor-to-Conveyor Configuration”.

The sample programs presented herein use the registers and Position Registers listed below. Change the register numbers as necessary.

Table 6.9.4(a) Registers

R[1]	Cycle stop flag. Setting 1 in this register causes the system to stop the cycle.
R[2]	PKCSGETQUE status
R[3]	PKCSGETTIME status
R[5]	PKFSGETQUE status
R[6]	Stores the conveyor station ID of conveyor CONV1.
R[7]	Stores the conveyor station ID of conveyor CONV2.
R[8]	Stores the fixed station ID of fixed station FSTN1.
R[10]	Pickup status (1: Picked; 2: Placed)
R[13]	Part arrival time

Table 6.9.4(b) Position registers

PR[2]	Pickup position on the conveyor
PR[3]	Placement position on the conveyor
PR[5]	Pickup position on the fixed station
PR[6]	Placement position on the fixed station
PR[12]	Tool offset to create an approach point for the pickup motion on the conveyor
PR[13]	Tool offset to create an approach point for the placement motion on the conveyor
PR[15]	Tool offset to create an approach point for the pickup motion on the fixed station
PR[16]	Tool offset to create an approach point for the placement motion on the fixed station

6.9.4.1 Main program

This is the main program of the system.

Setup example for a main program

MAIN1.TP

```
  1:   R[1:CYCLE_STOP]=0
  2:
  3:   !Get station id
  4:   CALL PKCSGETID('CONV1',CStn ID Reg=6)
  5:   CALL PKCSGETID('CONV2',CStn ID Reg=7)
  6:   CALL PKFSGETID('FSTN1',FStn ID Reg=8)
  7:
  8:   !Put a buffer to a FSTN
  9:   CALL PKFSPUTBUF(FStn ID=R[8:BufferID],Index=1)
 10:
 11:   UTOOL_NUM=1
 12:   UFRAME_NUM=0
 13: J PR[1:Perch Pos] 100% FINE
 14:
 15:   CALL PKWCSTART
 16:
 17:   R[10:Pick]=0
 18:   LBL[100]
 19:   !Pick a part from a CSTN
 20:   CALL PICKCS
 21:   IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 22:   IF R[10:Pick]=1,JMP LBL[200]
 23:
 24:   !Pick a part from a buffer
 25:   CALL PICKBF
 26:   IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 27:   IF R[10:Pick]=0,JMP LBL[100]
 28:
 29:   LBL[200]
 30:   !Drop a part to a CSTN
 31:   CALL DROPCS
 32:   IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 33:   IF R[10:Pick]=0,JMP LBL[100]
 34:
 35:   !Drop a part to a buffer
 36:   CALL DROPBF
 37:   IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 38:   IF R[10:Pick]=1,JMP LBL[200]
 39:   JMP LBL[100]
 40:
 41:   LBL[900]
 42:   CALL PKWCEND
[End]
```

If there is no part when the robot attempts to pick one, it tries to pick a part from the buffer.
If there is no placement position when the robot attempts to place a part, it places that part in the buffer.
The robot checks the value of R[10:Pick] to determine whether to use the buffer.
The value of R[10:Pick] is set to 1 when the robot hand holds a part or to 0 when the robot hand holds no part.

In other words, if 0 is set in R[10:Pick] when the robot attempts to pick a part from the conveyor, the robot picks a part from the buffer.
If 0 is set in R[10:Pick] when the robot attempts to place a part on the conveyor, the robot places the part in the buffer.

When multiple robots are used

In principle, the same programs may be copied for use on other robots.
Be careful about the following points.
In MAIN1.TP, change the fixed station name to that of the fixed station where the relevant robot operates.
PICKCS.TP and DROPFS.TP can be copied to other robots and used as they are. However, the tracking schedule number of the tracking program (PICKCS.TP) needs to be changed according to the conveyor station where the robot operates.
If PKWCSTART is executed for a robot without a sensor (not connected to a camera or photo eye), warning IRPK-031 is generated. This is a mere informational message and does not affect the operation.

6.9.4.2 Pick program

This program performs a pickup motion on a conveyor.

Setup example for a pick program

PICKCS.TP

```
 1:  UTOOL_NUM=1
 2:  UFRAME_NUM=0
 3:
 4:  STOP_TRACKING
 5:  !Get a part from a CSTN
 6:  CALL PKCSGETQUE(CStn ID=R[6:CstnIDIn],Consec Flag=1,Timeout (ms)=0,
     Offset VR=1,Stat Reg=2)
 7:  IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 8:  IF R[2:CsGetQStat]=0,JMP LBL[300]
 9:  IF R[2:CsGetQStat]=1,JMP LBL[100]
10:  IF R[2:CsGetQStat]=2,JMP LBL[900]
11:  JMP LBL[900]
12:
13:  LBL[100]
14:  !Check if the part is too upstream
15:  CALL PKCSGETTIME(CStn ID=R[6:CstnIDIn],Order=1,Concec Flag=1,
     Quantity=1,Time Reg=13,Offset VR=2,Stat Reg=3)
16:  IF R[13:Time]>=1000,JMP LBL[900]
17:
18:  LBL[200]
19:  ! Wait for the part from the CSTN
20:  CALL PKCSGETQUE(CStn ID=R[6:CstnIDIn],Consec Flag=1,Timeout (ms)=100,
     Offset VR=1,Stat Reg=2)
21:  IF R[1:CYCLE_STOP]=1,JMP LBL[900]
22:  IF R[2:CsGetQStat]=0,JMP LBL[300]
23:  IF R[2:CsGetQStat]=1,JMP LBL[200]
24:  IF R[2:CsGetQStat]=2,JMP LBL[900]
25:  JMP LBL[900]
26:
27:  LBL[300]
28:  !Pick the part from the CSTN
29:L PR[2] 10000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[12]
30:L PR[2] 10000mm/sec CNT0 VOFFSET,VR[1]
31:  CALL VACUUM_ON
```

```
 32:L PR[2] 10000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[12]
 33:
 34:   !Ack the part has been picked from the CSTN
 35:   CALL PKCSACKQUE(CStn ID=R[6:CstnIDIn],Success)
 36:   R[10:Pick]=1
 37:
 38:   LBL[900]
[End]
```

In [Line track schedule number] of the program detail screen, set the tracking schedule number specified for the conveyor station for which this robot is selected on conveyor CONV1.

The difference from the same program used to convey parts from conveyor station to conveyor station is that the robot judges whether to pick a part from the buffer by estimating the part arrival time if the part has not arrived at the tracking area.
On line 6, 0 is set as the wait time in PKCSGETQUE. This is intended to make the robot access the buffer immediately if there is no part to pick in the tracking area.
If a timeout occurs with PKCSGETQUE on line 6, the part arrival time is checked on lines 15 and 16. If it is found that the part to be picked next will not arrive soon, the robot ends this program, with 0 set in R[10:Pick], and accesses the buffer.
If the arrival time check reveals that the part will arrive soon, the robot repeatedly calls PKCSGETQUE on line 20 and waits for the part. Here, the wait time is set to 100 ms in order to reduce the CPU load.

6.9.4.3 Placement program

This program performs a placement motion on a conveyor.

Setup example for a placement program

DROPCS.TP

```
  1:  UTOOL_NUM=1
  2:  UFRAME_NUM=0
  3:
  4:  STOP_TRACKING
  5:  !Get a cell from a tray on a CSTN
  6:  CALL PKCSGETQUE(CStn ID=R[7:CstnIDIn],Consec Flag=1,Timeout (ms)=0,
      Offset VR=4,Stat Reg=2)
  7:  IF R[1:CYCLE_STOP]=1,JMP LBL[900]
  8:  IF R[2:CsGetQStat]=0,JMP LBL[300]
  9:  IF R[2:CsGetQStat]=1,JMP LBL[100]
 10:  IF R[2:CsGetQStat]=2,JMP LBL[900]
 11:  JMP LBL[900]
 12:
 13:  LBL[100]
 14:  !Check if the cell is too upstream
 15:  CALL PKCSGETTIME(CStn ID=R[7:CstnIDIn],Order=1,Concec Flag=1,
      Quantity=1,Time Reg=13,Offset VR=4,Stat Reg=3)
 16:  IF R[13:Time]>=1000,JMP LBL[900]
 17:
 18:  LBL[200]
 19:  !Wait for the cell from the CSTN
 20:  CALL PKCSGETQUE(CStn ID=R[7:CstnIDIn],Consec Flag=1,Timeout (ms)=100,
      Offset VR=3,Stat Reg=2)
 21:  IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 22:  IF R[2:CsGetQStat]=0,JMP LBL[300]
 23:  IF R[2:CsGetQStat]=1,JMP LBL[200]
```

```
  24:   IF R[2:CsGetQStat]=2,JMP LBL[900]
  25:   JMP LBL[900]
  26:
  27:   LBL[300]
  28:   !Drop the part to the cell
  29:L PR[3] 10000mm/sec CNT100 VOFFSET,VR[4] Tool_Offset,PR[13]
  30:L PR[3] 10000mm/sec CNT0 VOFFSET,VR[4]
  31:   CALL VACUUM_OFF
  32:L PR[3] 10000mm/sec CNT100 VOFFSET,VR[4] Tool_Offset,PR[13]
  33:
  34:   !Ack the cell has been occupied
  35:   CALL PKCSACKQUE(CStn ID=R[7:CstnIDIn],Success)
  36:   R[10:Pick]=0
  37:
  38:   LBL[900]
[End]
```

This program is basically the same as the pick program for picking parts from a conveyor.

In [Line track schedule number] of the program detail screen, set the tracking schedule number specified for the conveyor station for which this robot is selected on conveyor CONV2.

6.9.4.4 Pick program (Buffer)

This program performs a pickup motion on a fixed station for buffer.

Setup example of a pick program (Buffer)

PICKBUF.TP

```
   1:   UTOOL_NUM=1
   2:   UFRAME_NUM=0
   3:
   4:   !Get a part from a buffer
   5:   CALL PKFSGETBUF(FStn ID=R[8:BufferID],Index=1,Part,Offset VR=3,Stat Reg=5)
   6:   IF R[5:FsGetQStat]<=0,JMP LBL[100]
   7:   IF R[5:FsGetQStat]>0,JMP LBL[900]
   8:
   9:   LBL[100]
  10:   !Pick the part from the buffer
  11:L PR[5] 10000mm/sec CNT100 VOFFSET,VR[3] Offset,PR[15]
  12:L PR[5] 10000mm/sec CNT0 VOFFSET,VR[3]
  13:   CALL VACUUM_ON
  14:L PR[5] 10000mm/sec CNT100 VOFFSET,VR[3] Offset,PR[15]
  15:
  16:   !Ack the part has been picked from the buffer
  17:   CALL PKFSACKBUF(FStn ID=R[8:BufferID],Success)
  18:   R[10:Pick]=1
  19:
  20:   LBL[900]
[End]
```

A user frame number is specified on line 2. While the tracking program requires setting 0 for the user frame, a desired user frame number can be set for a fixed station.
On line 5, a part is picked from the buffer.

On lines 11 to 14, a pickup motion is performed. You can optimize the motion further by using the linear distance specification instruction, maximum linear speed function, and time before instruction. For details, see Subsections 6.12.5 “Linear Distance Specification”, 6.12.7 “Maximum Linear Speed Function”, and 6.12.8 “Time Before Instruction”.

6.9.4.5 Placement program (Buffer)

This program performs a placement motion on a fixed station for buffer.

Setup example of a placement program (Buffer)

DROPBUF.TP

```
  1:  UTOOL_NUM=1
  2:  UFRAME_NUM=0
  3:
  4:  !Get a cell from a buffer
  5:  CALL PKFSGETBUF(FStn ID=R[8:BufferID],Index=1,Cell,Offset VR=2,Stat Reg=5);
  6:  IF R[5:FsGetQStat]<=0,JMP LBL[100]
  7:  IF R[5:FsGetQStat]>0,JMP LBL[900]
  8:
  9:  LBL[100]
 10:  !Drop a part on the buffer
 11:L PR[6] 10000mm/sec CNT100 VOFFSET,VR[2] Offset,PR[16]
 12:L PR[6] 10000mm/sec CNT0 VOFFSET,VR[2]
 13:  CALL VACUUM_OFF
 14:L PR[6] 10000mm/sec CNT100 VOFFSET,VR[2] Offset,PR[16]
 15:
 16:  !Ack the cell has been occupied
 17:  CALL PKFSACKBUF(FStn ID=R[8:BufferID],Success,2)
 18:  R[10:Pick]=0
 19:
 20:  LBL[900]
[End]
```

This program is basically the same as the program for picking parts from a buffer fixed station.

On line 5, an empty cell is picked from the buffer. On lines 11 to 14, a placement motion is performed.

6.10 REFERENCE POSITION SETTING

The reference position is the part detection position to be used when the pickup position or placement position is taught to the robot program.
Teach the pickup position or placement position to all the robots that operate on the same conveyor in relation to the part whose reference position is set. To set the reference position, use the reference position setting guide in the sensor setup screen. Teach the position to each robot while making sure that the position of the part whose reference position is set on the conveyor remains unchanged.
In the case of a fixed station, this procedure is not necessary. Teach the position to the robot directly.

⚠ CAUTION
1 After setting the reference position, the conveyor may move but you need to take care to prevent the part position relative to the conveyor from changing, until robot position teaching is complete. If the conveyor has multiple conveyor stations, prevent the part from moving, until robot position teaching is complete for all the conveyor stations. If the part is moved inadvertently, redo the reference position setting procedure from the beginning.
2 If there are multiple robots, check the encoder screen of each robot controller, with the conveyor stopped, to see whether the value set in [Current Count (cnts)] is the same. If the values are different, turn the power of all the robot controllers off and then on.

NOTE
1 To track both the pickup position and placement position, first set the reference position and perform robot position teaching in the pick program. Next, start the conveyor, let the robot pick a part with a tracking motion. Then, set the reference position and perform robot position teaching in the placement program.
2 If [Ref. Data To Use] is set to [Model ID] in the vision process, repeat the procedure described below for each model ID. In such a case, set each model ID individually using a locator tool such as the GPM locator tool, and create the reference data for the number of model IDs by adding the reference data before setting the reference position.
3 You need to open the *i*RPickTool Setup screen with the controller selected for this sensor and configure the reference position.

For the operation procedures for the following combinations, refer to Subsection 3.2.10, "Setting Up a Reference Position and Teaching a Robot Position".

- In the case of [Distance] + [Find part by vision]
  The procedure for an infeed conveyor is given, but it is the same in other cases, too.
- In the case of [DI]/[RI]/[HDI] + [Put part in queue]
  The procedure for when using a tray on an outfeed conveyor is given, but it is the same in other cases, too.

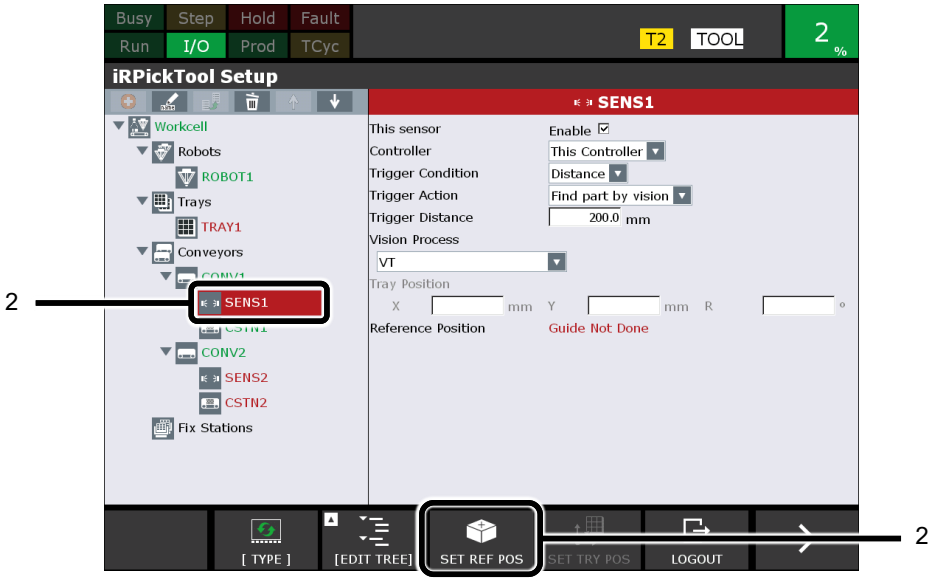
In the case of the combination below, the part is not actually detected, so set the reference position at the same time as you set the target position. For details, refer to Subsection 6.5.6, "Setting the Target Position."

- In the case of [Distance]/[Flag] + [Put part in queue]

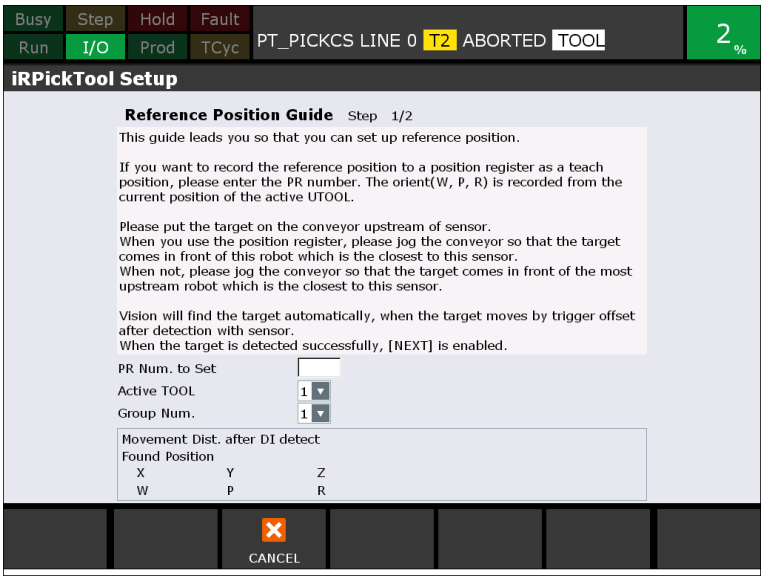
6.10.1 [DI] / [RI] and [Find part by vision]

This section explains the procedure to set the reference position for the case where you have set the trigger condition to [DI] or [RI] and the action to [Find part by vision] on the sensor setup screen.
For sensor setup, refer to Section 6.5, "SETUP OF A SENSOR."

1 Stop the conveyor.
2 Select [SENS*] for the sensor, and press F3 [SET REF POS].



The following [Reference Position Guide Step: 1/2] will appear.



NOTE
When [PR Num. to Set] is available, the robot position teaching described later can be simplified by using this function. For details, refer to "Using a position register number for [PR Num. to Set]" in Subsection 3.2.10.1, "Setting up a reference position for the infeed conveyor, and teaching a robot position."

3 Place a part upstream of the trigger sensor.
Place it around the center of the conveyor's width direction.

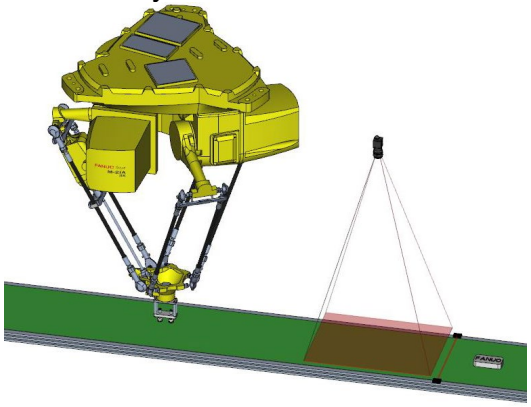
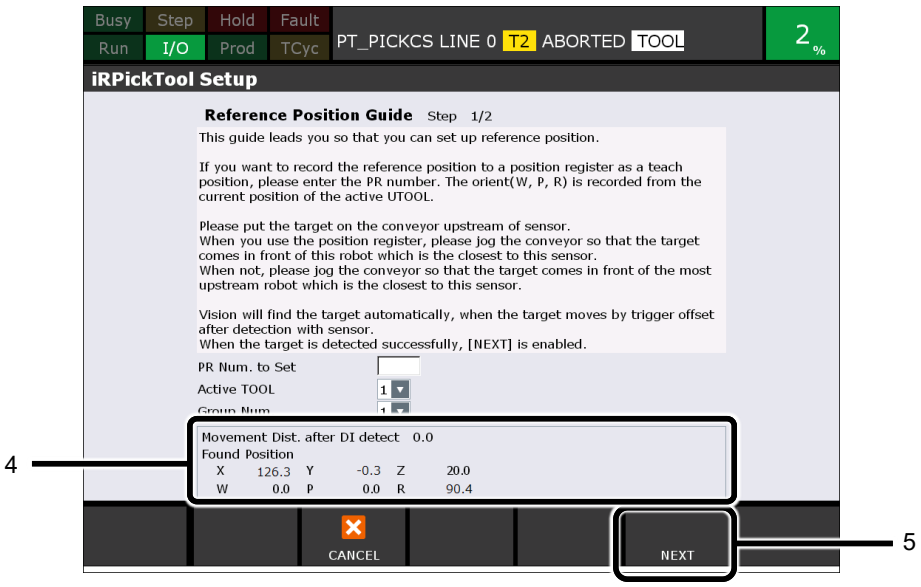


Fig. 6.10.1(a) Placing a part upstream of the trigger sensor

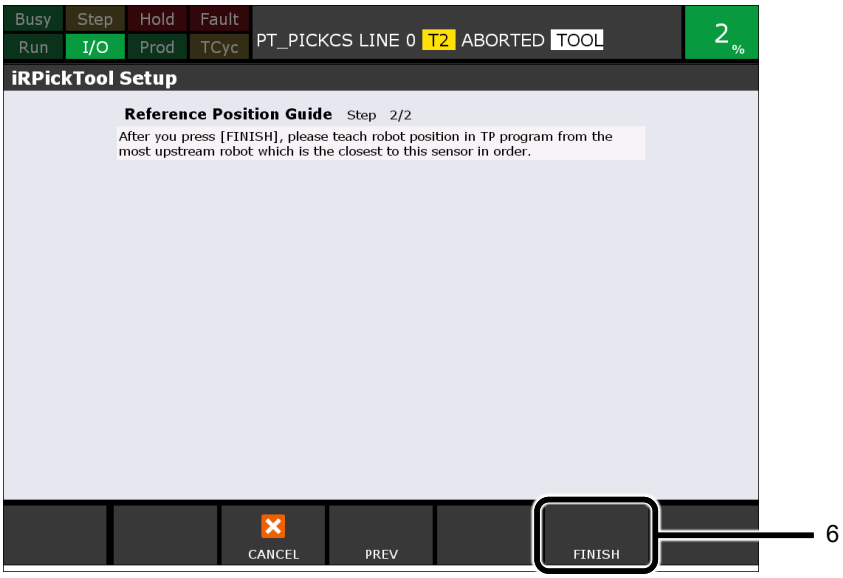
4 Move the conveyor, and stop it when the part comes within the robot's operating space.
If detection succeeds, [Found Position] will be displayed.
* If detection does not succeed
- Open the vision program and select Runtime Image '100% Display'.
- Open the Runtime Image. Check the image.
- If the detection timing is off, change the value of [Trigger Offset].

If detection succeeds, F5 [NEXT] will become activated.

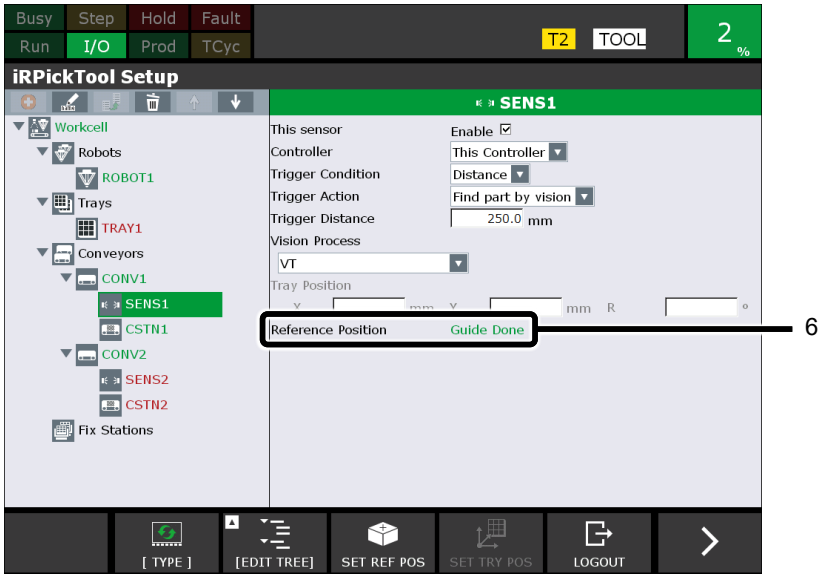
5 Press F5 [NEXT].



6 Press F5 [FINISH].



A screen like the following one will appear.
Check that [Guide Done] appears in [Reference position].



7 Next, teach the robot position.
Refer to Section 6.11, “TEACHING THE POSITION TO ROBOTS.”

6.11 TEACHING THE POSITION TO ROBOTS

After setting the reference position, teach the position to robots.
For the operation procedures for the following combinations, refer to Subsection 3.2.10, "Setting Up a Reference Position and Teaching a Robot Position."

- In the case of [Distance] + [Find part by vision]
  The procedure for an infeed conveyor is given, but it is the same in other cases, too.
- In the case of [DI]/[RI]/[HDI] + [Put part in queue]
  The procedure for when using a tray on an outfeed conveyor is given, but it is the same in other cases, too.

6.11.1 Conveyor Station

1 When the part enters the tracking area after the reference position is set in the sensor guide screen, stop the conveyor.

> ⚠ **CAUTION**
> Be careful that the part does not become misaligned on the conveyor when the conveyor has been stopped.

2 From the program list, open the tracking program.
3 Check that the user frame and user tool frame used in the tracking program are selected.
4 Jog the robot to teach the pickup position (or placement position).

> ⚠ **CAUTION**
> 1 Be sure to teach the pickup (or placement) position in the tracking program. This is necessary to teach the coordinates in the tracking frame. Even if a Position Register is used, do not teach the position on the Position Register screen. If you place the cursor on the line number of the motion instruction in the tracking program that moves the robot to the pickup position (or placement position), with the robot located at the pickup position (or placement position), and press F5 [TOUCHUP] key while holding down the shift key, the position is taught. When you are asked "Subtract VR[*] from current pos ?", select [NO]. Or when you are asked "VR[*] is UNINIT, continue ?", select [YES].
> 2 When using trays, teach the placement (or pickup) position for cell 1. The offset for each cell is calculated automatically from the coordinates of the cell and the coordinates of the reference cell. By using a vision correction instruction, the robot moves to each cell by referring to the offset stored in a vision register.
> 3 In the case of a 3-, 4- or 5-axis robot, you must teach the robot position as each W and P of it are 0 or 180 or -180.
> 4 In the case of a 5-axis robot, there are postures that the robot cannot take due to its structural limitations. Therefore, teach the pickup position (or placement position) so that the flange face faces downward (or upward). For details, refer to "Motion of 5-axis robot" in the subsection, "Motion Format" in the section, "MOTION INSTRUCTIONS" in the chapter, "PROGRAM STRUCTURE" in the "OPERATOR'S MANUAL (Basic Operation) B-83284EN".

NOTE
The positions of the approach point and retract point can be created relative to the taught pickup position (or placement position), using the tool offset instruction (or position offset instruction). In that case, you do not need to teach the positions of the approach point and retract point.

5 If the conveyor has multiple conveyor stations, repeat the above steps for each conveyor station.

NOTE
If the hand grippers of all robots have the same shape, you can omit robot position teaching for the second and subsequent robots. The position taught to the first robot can be copied and used as is.

6.11.2 Fixed Station

On the tray setup screen, press F3 [SET REF CELL] and set the values of Cell 1 for the reference cell, then teach the placement/picking position for Cell 1 only. For details, refer to Subsection 6.3.3, "Setting the Reference Cell."

The position for each cell is created in relation to the reference cell, using a vision correction instruction.

NOTE
The placement/picking position for Cell 1 can be taught in relation to any arbitrary user frame other than the user frame that indicates the tray frame.

6.12 FINE ADJUSTMENT OF THE TRACKING MOTION

This section describes how to further optimize the tracking motion of the robot. It is assumed that the teaching of the basic motions has been completed.

Adjust the position while performing the actual tracking motion.

If there is any error in handling precision, it is important to make a careful observation of the condition, direction, and amount of displacement to identify the cause and take an appropriate action for that cause. You cannot improve precision unless you take an appropriate action for the cause.

The following subsections describe how to adjust the tracking motion for different types of error.

6.12.1 Initial Fine Adjustment

This is the first adjustment you need to make after robot position teaching. Fine-adjust the taught position while conveying a part on the conveyor several times. Make this adjustment for each robot individually. When you use the vision system, you will take similar steps in the procedure described in the next subsection, "6.12.2 Adjustment of the Tracking Frame Error". In that case, skip this subsection and start with the next subsection.

1 Insert a wait instruction immediately after the pickup position of the tracking program.
The wait instruction is inserted to check the error visually. Set the wait time to 30 seconds or so.
2 Move the conveyor at the operating speed.
3 Start the main program as usual, and convey a part on the conveyor.

> ⚠ **CAUTION**
> Select T2 mode or AUTO mode to run the tracking program. If you select T1 mode, you will see the alarm, "LNTK-041 Encoder is moved in T1 mode".

4 When the robot reaches immediately above the position where the part is to be picked, stop the conveyor. Leave the program running.
5 When the conveyor stops completely, stop the program temporarily.
6 In this condition, check the amount of displacement between the robot position and the part position.
7 If there is any displacement, jog the robot to the correct position. To adjust the position, place the cursor on the line of the motion instruction that moves the robot to the pickup position. When asked about whether to subtract the offset value, select [YES].
8 Delete the inserted wait instruction from the tracking program.

6.12.2 Adjustment of the Tracking Frame Error

When a camera is used to detect parts, the tracking motion is subject to error if the tracking frame recognized by the camera does not match that recognized by each individual robot. While this error is corrected properly if the part is not rotating, displacement is generated if the part is rotating.

This error occurs because the tracking frame is not set up correctly. To fix the error, it is necessary in principle to set up the tracking frame again beginning with the setting of the tool frame. You can improve the situation, however, by adding an amount of adjustment to the vision offset calculated by *i*RVision as instructed below. Make this adjustment for each robot individually.
In the sample program shown below, Position Register 20 is used to store the amount of adjustment. Change it as necessary.

1 Add the line indicated below after PKCSGETQUE is succeeded in the tracking program.

Example: PICK1.TP in Subsection 6.9.2.2, "Pick program"

```
 5:  LBL[100]
 6:  STOP_TRACKING
 7:  CALL PKCSGETQUE (CStn ID=R[11:CSTN1],Consec Flag=R[3:ZONE],
      Timeout (ms)=100, Offset VR=1,Stat Reg=2)
 8:  IF R[2:GETQ_STATUS]=0,JMP LBL[200]
 9:  IF R[1:CYCLE_STOP]=1,JMP LBL[900]
10:  JMP LBL[100]
11:
12:  LBL[200]
13:  CALL ADJ_OFS(1, 1, 20, 3)
```

2 Modify the tracking program according to the number of the Vision Register that stores the adjusted vision offset.

```
12:  LBL[200]
13:  CALL ADJ_OFS(1, 1, 20, 3)
14:L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[3] Tool_Offset,PR[10]
15:L PR[R[3:ZONE]] 4000mm/sec CNT0 VOFFSET,VR[3]
16:  CALL VACUUM_ON
17: L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[3] Tool_Offset,PR[10]
```

3 Insert a motion instruction and a wait instruction immediately after the approach point so as to make the robot stop immediately above the pickup position set in the tracking program.

```
12:   LBL[200]
13:   CALL ADJ_OFS(1, 1, 20, 3)
14:L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[3] Tool_Offset,PR[10]
15:L PR[R[3:ZONE]] 4000mm/sec FINE VOFFSET,VR[3] Tool_Offset,PR[15]
16:   WAIT 10.00(sec)
17:L PR[R[3:ZONE]] 4000mm/sec CNT0 VOFFSET,VR[3]
18:   CALL VACUUM_ON
19: L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[3] Tool_Offset,PR[10]
```

The wait instruction is inserted to check the displacement visually. Set the wait time to 10 seconds or so. Also, set a small value in Z of PR[15] (tool offset to create an approach point for the pickup motion; set 0 for the values other than Z). This is necessary to make the robot stop immediately above the pickup position.

4 In PR[20], set X, Y, Z, W, P and R to 0, 0, 0, 0, 0, and 0.
5 Move the conveyor at the operating speed.
6 Start the main program as usual, and convey a part on the conveyor. Make sure that the part faces the same direction as when the reference position was set.

⚠ CAUTION
Select T2 mode or AUTO mode to run the tracking program. If you select T1 mode, you will see the alarm, “LNTK-041 Encoder is moved in T1 mode”.

7 When the robot reaches the approach point, stop the conveyor. Leave the program running.
8 When the conveyor stops completely, stop the program temporarily.
9 Check the amount of displacement between the part position and the robot position.
10 If there is any displacement, jog the robot to the correct position. When asked about whether to subtract the offset value, select [YES].
11 Start the main program again, and convey a part on the conveyor.
This time around, place the part so that it faces 180 degrees opposite the direction used when the reference position was set.
12 When the robot reaches the approach point, stop the conveyor. Leave the program running.
13 When the conveyor stops completely, stop the program temporarily.
14 Check the amount of displacement between the robot position and the part position.
15 If there is any displacement, measure the displacement in the X direction (parallel to the conveyor moving direction) and in the Y direction (orthogonal to the conveyor moving direction) individually. Set the amount of adjustment for the X direction in X of PR[20] and the amount of adjustment for the Y direction in Y of PR[20]. As the amount of adjustment, set a value equal to half of each amount of displacement. Execute the program from the line of ADJ_OFS, and check whether the robot moves to the correct position. If there is any displacement, change the amount of adjustment and execute the program again from the line of ADJ_OFS to see whether the robot moves to the correct position.
16 After confirming that the robot moves to the correct position, delete the inserted motion instruction and wait instruction from the tracking program.

About ADJ_OFS

ADJ_OFS is a KAREL program that adds the specified amount of adjustment to the vision offset stored in a Vision Register. The meanings of its arguments are described below. For details, refer to the section, "VISION SUPPORT TOOLS", in the chapter, "OTHER OPTIONS" in the "*i*RVision OPERATOR'S MANUAL (Reference)".

Argument 1:

Type of the register storing the vision offset. If the vision offset is stored in a Vision Register, specify 1.

Argument 2:

If the vision offset is stored in a Vision Register, specify the number of that register.

Argument 3:

Number of the Position Register storing the amount of adjustment

Argument 4:

Number of the Vision Register storing the adjusted vision offset

NOTE

With *i*RPickTool, ADJ_OFS is available as a standard feature even if the vision support tool option is not ordered.

6.12.3 Adjustment of the Dynamic Error of the Robot

While the robot correctly reaches above the pickup position of a part when the conveyor is stopped, a certain amount of displacement occurs when the conveyor is moving. This error is due to a tracking delay in the robot motion.

NOTE

1. Before adjusting the dynamic error, make sure that no error occurs when the conveyor is stopped by making the error adjustments described in Subsections 6.12.1 and 6.12.2.
2. If the robot works on the plural conveyors, use the fastest conveyor to adjust the dynamic error.

This error is adjusted using the following procedure.

1 Insert a motion instruction and a wait instruction immediately after the approach point so as to make the robot stop immediately above the pickup position set in the tracking program.

Example: PICK1.TP in Subsection 6.9.2.2, "Pick program"

```
12:   LBL[200]
13:L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
14:L PR[R[3:ZONE]] 4000mm/sec FINE VOFFSET,VR[1] Tool_Offset,PR[15]
15:   WAIT 10.00(sec)
16:L PR[R[3:ZONE]] 4000mm/sec CNT0 VOFFSET,VR[1]
17:   CALL VACUUM_ON
```

The wait instruction is inserted to check the displacement visually. Set the wait time to 10 seconds or so. Also, set a small value in Z of PR[15] (tool offset to create an approach point for the pickup motion; set 0 for the values other than Z). This is necessary to make the robot stop immediately above the pickup position.

2 Move the conveyor at the operating speed.
3 Start the main program as usual, and convey a part on the conveyor. Make sure that the part faces the same direction as when the reference position was set.

> ⚠ **CAUTION**
> Select T2 mode or AUTO mode to run the tracking program. If you select T1 mode, you will see the alarm, "LNTK-041 Encoder is moved in T1 mode".

4 Check that the robot position correctly follows the pickup position of the part when the conveyor is moving and the program is running.
5 If there is any displacement, enter the value calculated by the following equation in [Dynamic Error Adjustment] mentioned in Section 6.2, "SETUP OF A ROBOT" : Displacement amount (mm) ÷ conveyor speed (mm/s) x 1000 (unit: ms). The conveyor speed is that observed when the robot follows the part. The displacement amount has a positive sign when the robot is displaced from the part in the upstream direction of the conveyor and a negative sign when displaced in the downstream direction. If a value is already set in [Dynamic Error Adjustment], add the value calculated above to the existing value.
6 Delete the inserted motion instruction and wait instruction from the tracking program.

Calculation example

When 30 is set in [Dynamic Error Adjustment], the conveyor speed is 500 mm/s, and the pickup position of the part is displaced by 3 mm in the upstream direction of the conveyor:
[Dynamic Error Adjustment] = +30 (ms) + (3 (mm) ÷ 500 (mm/s)) x 1000 = +36 (ms)

6.12.4 Continuous Termination Type Specification for a Taught Position

Normally, three motions are taught in the pick program and placement program; namely, movement to the approach point, movement to the pickup position (or placement position), and movement to the retract point. The Fine (fine position) or CNT (continuous) setting for these motions can not only change the robot operation path but affect the cycle time as well.

Approach point and retract point

Since the robot does not need to pass exactly over the approach point and retract point, you can shorten the cycle time by specifying the largest possible value in CNT. The specifiable range of CNT values is 0 to 100, with CNT100 representing the highest speed. However, a larger CNT value causes the robot operation path to curve more inwardly. If the robot moves inwardly, it may catch another part on the conveyor. Adjust the CNT value taking into consideration the operation path and cycle time.

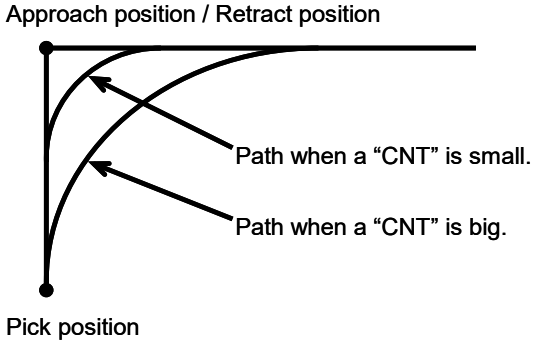


Fig.6.12.4 (a)

Pickup position and placement position

Since the robot needs to reach the exact pickup position (or placement position), normally specify FINE, instead of CNT. However, when FINE is specified, the robot moves slowly compared to when CNT is specified. To shorten the cycle time without significantly affecting the operation path, specify a small value in CNT. The specifiable range of CNT values is 0 to 100. Adjust the value in the range of 0 to 2, taking into consideration the operation path and cycle time.

Note, however, that if CNT is specified, the robot moves slightly inward even when the value is small. This causes the robot to start moving toward the retract point before it arrives at the taught pickup position (or placement position). In the case of taught position A in the figure below, for example, the robot moves down to a specified level when FINE is specified. When CNT is specified, by contrast, the robot starts moving up toward the retract point before moving down to a specified level.
Therefore, as with taught position B, set the destination position slightly lower than the position to be actually reached. Lowering the position sharply is dangerous; adjust the position in small increments while checking the robot motion.

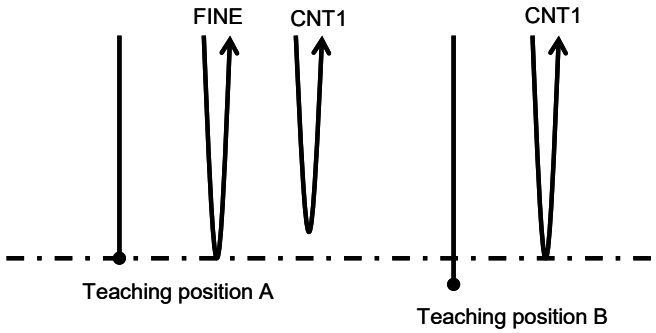


Fig.6.12.4 (b)

6.12.5 Linear Distance Specification

If a large CNT value is specified for the approach point or retract point and CNT is specified for the pickup position (or placement position) instead of FINE, the robot may fail to reach the destination position not only in height but also in the X and Y directions. The remaining distance appears to be an extra handling error.

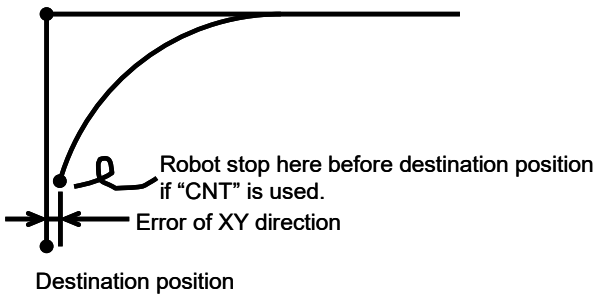


Fig.6.12.5 (a)

In such a case, add a linear distance specification instruction to the motion program for the pickup position (or placement position).
If this instruction is added, the robot moves straight over a specified distance leading up to the destination position, even when CNT is specified for the preceding position.
However, having the robot move straight over a longer distance leads to a longer cycle time. Therefore, make an adjustment taking into consideration the robot motion and cycle time.

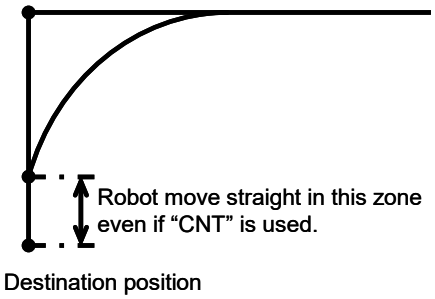


Fig.6.12.5 (b)

An example of the modified program is shown below. This is part of the program described in Subsection 6.9.2.2, “Pick program”.

```
13:L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
14:L PR[R[3:ZONE]] 4000mm/sec CNT0 VOFFSET,VR[1] AP_LD10
15:   CALL VACUUM_ON
16:L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10] RT_LD10
```

A linear distance specification instruction is added on line 14. This causes the robot to move straight over a distance of 10 mm leading up to the destination position.

6.12.6 Linear Face Plate Instruction

A linear face plate instruction is added to a motion instruction in a TP program. Add this instruction when the TCP is offset from the center of the face plate.

In a normal linear motion, control is exerted so that the TCP moves straight, as shown in the figure below. As a result, when the TCP is offset from the center of the face plate, particularly as in the case of a robot hand that can hold multiple parts, the center of the face plate zigzags and the robot has to move a longer distance, potentially resulting in the cycle time being not optimal.

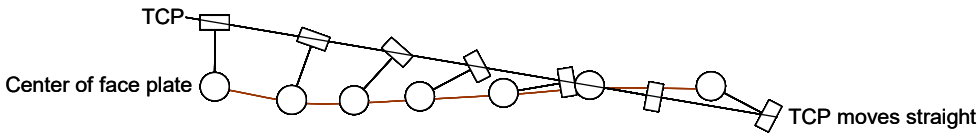


Fig.6.12.6 (a) Normal linear motion

If a linear face plate instruction is used, the center of the face plate becomes straight as shown in the figure below, although the linearity of the TCP is not maintained. This minimizes the robot movement distance and leads to a shorter cycle time.

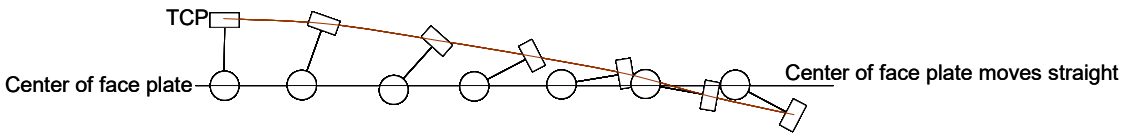


Fig.6.12.6 (b) When a linear face plate instruction is used

The instruction is expected to be effective particularly with a “Genkotsu” robot with a fast-moving wrist that repeats a linear motion above the conveyor during the pickup operation by using a hand where the TCP is offset from the center of the face plate.

An example of the modified program is shown below. This is part of the program described in Subsection 6.9.2.2, “Pick program”.

```
13:L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10] FPLIN
14:L PR[R[3:ZONE]] 4000mm/sec CNT0 VOFFSET,VR[1] AP_LD10
15:   CALL VACUUM_ON
16:L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10] RT_LD10
```

A linear face plate instruction is added on line 13. This is used to move the robot to the approach position.

⚠ CAUTION
1 If a linear face plate instruction is used, the TCP does not move straight. Use this instruction for a motion in an area where no interference with a surrounding object is likely to occur, such as above the conveyor.
2 The effectiveness of a linear face plate instruction depends on the robot in use and the motion involved.

6.12.7 Maximum Linear Speed Function

The maximum speed specified in a linear motion instruction (e.g., 10000 mm/sec) is based on a limited motor performance. The robot can actually move faster than the speed described in the motion instruction. In *i*RPickTool applications, the robot is required to move as fast as possible. This function enables the robot to perform a linear motion at its maximum speed.
Note, however, that this function is automatically ignored by a tracking program. It can thus be used only for operations on a fixed station.

⚠ CAUTION
1 If this function is used, the robot moves at a high speed. Before executing a motion instruction, check carefully that there is no dangerous object around the robot.
2 This function is automatically disabled for a tracking motion.
3 The execution time may vary with the time before Instruction.
4 The distance specified with the Linear Distance Specification function may not be maintained.

An example of the modified program is shown below. This is part of the program described in Subsection 6.9.3.2, “Pick program”.

```
10:L PR[2] max_speed CNT100 VOFFSET,VR[1] Tool_Offset,PR[11]
11:L PR[2] 10000mm/sec CNT0 VOFFSET,VR[1]
12:   CALL VACUUM_ON
13:L PR[2] 10000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[11]
```

In the motion instruction on line 10, [max_speed] is selected as the speed. This enables the maximum linear speed function when the robot moves to the approach position. This function is mainly used for movement to the approach position.

6.12.8 Time before Instruction

When the system operates at a high speed, the performance may be affected by the time for waiting for the exchange of signals to turn vacuum on and off. In such a case, add a time before instruction to the motion program for the pickup position (or placement position).
If this instruction is added, signals can be output while the robot is in motion. This eliminates the wait time for the exchange of signals, leading to a shorter cycle time.

An example of the modified program is shown below. This is part of the program described in Subsection 6.9.2.2, "Pick program".

```
13:L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
14:L PR[R[3:ZONE]] 4000mm/sec CNT0 VOFFSET,VR[1] TB .5sec,CALL VACUUM_ON
15:
16:L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
```

A time before instruction is added on line 14. Here, subprogram "VACUUM_ON" is executed 0.5 seconds before the destination position is reached. By adjusting the number of seconds here, vacuum can be turned on at the right timing before the robot arrives at the destination position.

6.13 PRECAUTIONS FOR WHEN STARTING UP MULTIPLE ROBOTS

When visual tracking is performed with multiple robots, the robot controller program that executes the sensor task should be started last. This is necessary to ensure that the sensor task starts detecting parts after every robot controller clears part information from all conveyor stations.
If there is more than one robot controller that executes the sensor task, insert a wait instruction before the start of the sensor task so that the sensor task is started after part information is cleared from all conveyor stations, as described in Subsection 6.9.2.1, "Main program".

6.14 SETUP OF PRE-GROUPING

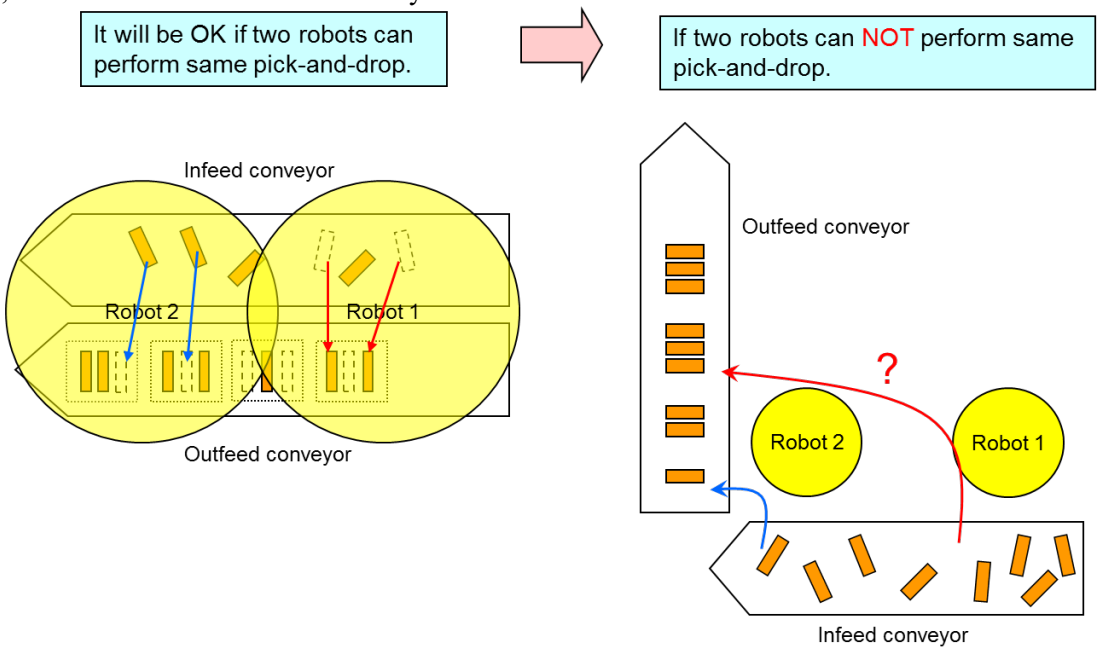
This section describes the setup procedure of the pre-grouping function which arranges parts on the same conveyor.

For settings specific to North America, in order to use the pre-grouping function, you will need the option for North America R856: *i*RPickTool / Pre-grouping.

Basically, you have pre-grouping robots which arrange random parts on the infeed conveyor into an arranged group. The arranged group is also sometimes called as a virtual tray. The non pre-grouping robots which are usually downstream of the pre-grouping robots pick the arranged group in one shot and place it on the outfeed conveyor. Pre-grouping is desired in the following cases:

Example 1

When two robots cannot perform same pick-and-drop, due to the customer's requirement such as a system layout, this feature can be used effectively.



Example 2

Even two robots can perform same pick-and-drop, due to too long path, cycle time for each robot is sometimes too slow. This feature may overcome the situation.

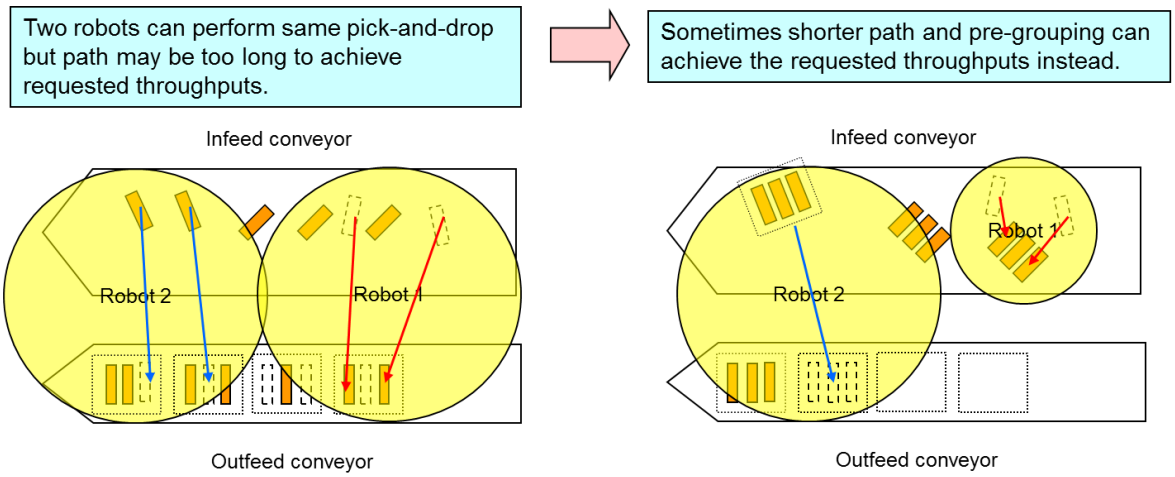


Fig.6.14 (a) Example of a configuration in which the pre-grouping function can be used effectively

Suppose there are the three robots system which consists of two upstream robots which arranges parts on the same conveyor into virtual trays and the 3rd robot which exists in the most downstream and conveys arranged parts to another conveyor at one time.

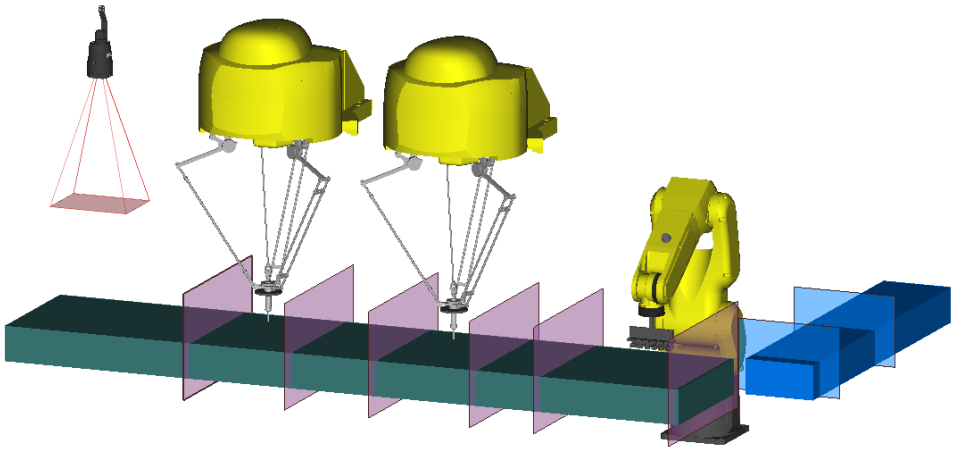


Fig.6.14 (b)

The system configuration and tree view composition in this case are shown in Fig. 6.14 (c).

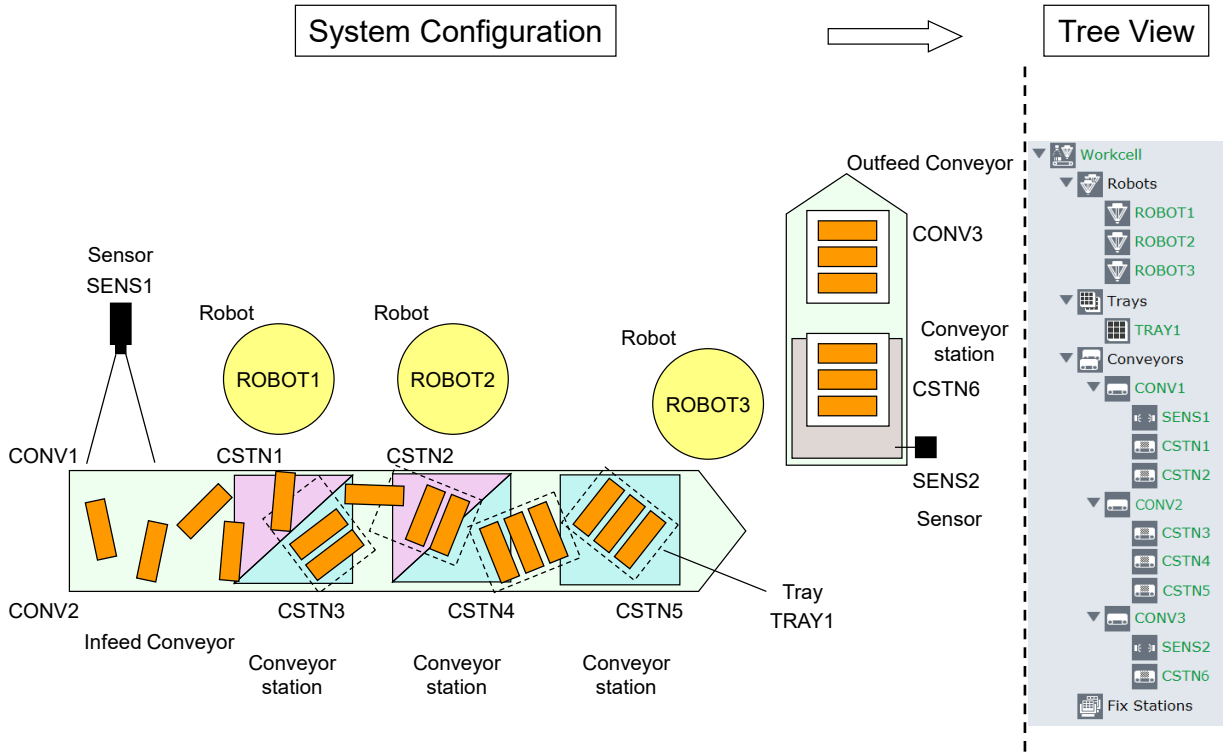


Fig. 6.14 (c)

As you see, two conveyor station objects are created to the same area for arrangement on the same conveyor in the tree view. One is for picking operation and the other is for placement operation. So when you wish to setup pre-grouping, you must
Create an infeed conveyor as usual.
Create a pre-grouping outfeed conveyor. This represents the same physical conveyor as the infeed conveyor.

You will note in steps described later that there are additional setup items on the pre-grouping outfeed conveyor and its conveyor stations.
You should also note that the picking conveyor station CSTN5 in Fig.6.14 (c) of the non pre-grouping robot ROBOT3 which picks the arranged parts is actually defined in the pre-grouping conveyor CONV2 itself and not the Infeed conveyor CONV1.

6.14.1 Key Concepts

This section describes terms related to a pre-grouping function.

Infeed Conveyor

This is a conveyor object for picking operation.
In Fig. 6.14 (c), CONV1 is an infeed conveyor.

Pre-grouping Conveyor

This is a conveyor object for arranged placement or pre-grouping operation. In Fig. 6.14 (c), CONV2 is a pre-grouping conveyor.

Virtual Tray

If parts are arranged on the same conveyor, the tray is not actually flown, but parts seems to be arranged in line on the virtual tray. This means such a virtual tray mentioned above.

The arranged pattern to be created by the pre-grouping robot is always defined using a tray object and assigned to the pre-grouping conveyor. During run-time, even if only random parts flow and not the defined trays as in the traditional or non pre-grouping conveyor, the parts are grouped into the defined tray pattern. This pre-grouped tray pattern is also called a virtual tray since there is no physical tray. The orientations of the virtual trays can vary widely depending upon the randomness of the part flow and part density.

Base Part

The way a virtual tray is built, first one part on the conveyor which has sufficient space around it to build the tray without any interference with other parts is chosen. This is called the base part. The base part part is not moved when arranging.

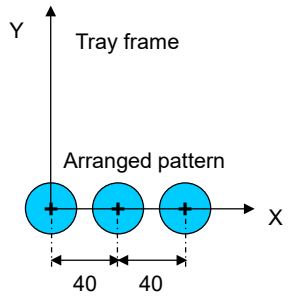
6.14.2 Setup of a Workcell and a Robot

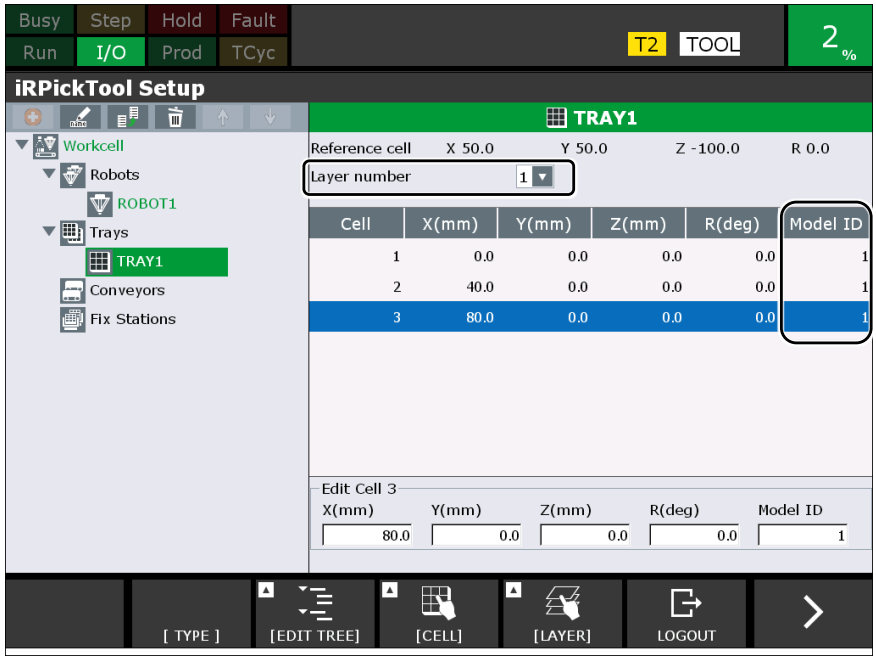
The setup of the Workcell and the Robot are done as usual. There are no special steps for pre-grouping. Refer to 6.1, “SETUP OF A WORKCELL” and 6.2, “SETUP OF A ROBOT”.

6.14.3 Setup of a Tray

This section shows how to set up a virtual tray.

- The origin point of a tray frame can be set as arbitrary positions, when treating the part of a [Circle (without Phase)]. For example, it is set up so that X of the cell 1 and Y may be set to 0.
- When treating the part of a [Circle (without Phase)], the direction of X of a tray frame is automatically calculated so that it may become the same as the direction of X of a tracking frame. However, according to a condition, the direction of X of a tray frame is selected from plural candidates of directions when the setting item [Orientation] mentioned later is set to be enabled.
- The arranged pattern is set as the layer 1. With the pre-grouping function, the robot is allowed to create only a 1 layer tray. Presently, pre-grouping does not support multiple layer trays.
- A model ID cannot be used by a pre-grouping function. The model ID of all cells in the layer 1 must be set to 1.





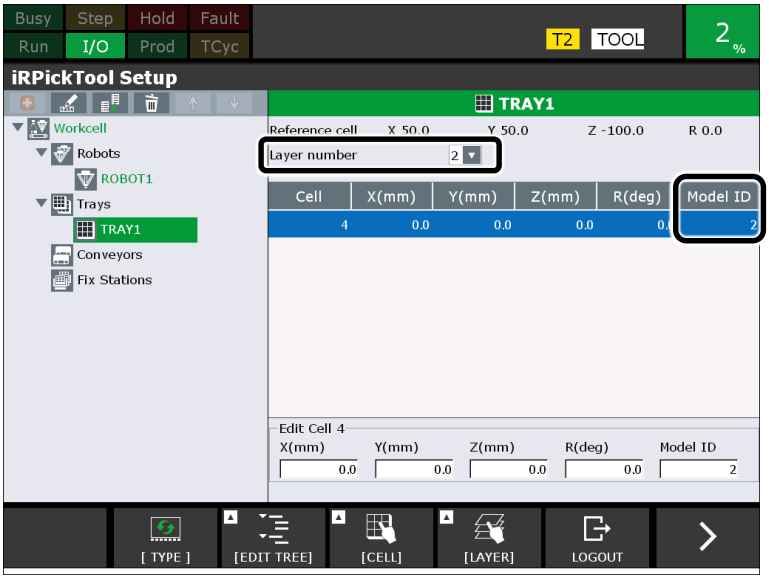
⚠ CAUTION

The center position of a part is set to the coordinates of each cell in the layer 1. If the center position of a part is not set to the coordinates of a cell correctly, the part to be placed for arrangement may collide with the part on a conveyor, because it may be calculated correctly of the check of interference between the part on a conveyor and the part to be placed.

NOTE

If cells are too close to each other, it may be judged that the base part and arranged parts will collide with each other when PKCSGENVRTRY is called, and the error IRPK-088: “There is an invalid pre-grouping setting.” may occur.

- Even though pre-grouping allows creating only single layer virtual trays, you can define one cell in layer 2. This has a special purpose. This cell is used to pick all the arranged parts at one shot by the downstream non pre-grouping robot.
- In order to pick only completely arranged parts, the cell of the layer 2 should be set up with a model ID different from the cell of the layer 1. For example, a model ID is set as 2. When the most downstream robot takes out all arranged parts, this model ID must be specified by PKCSGETQUE.

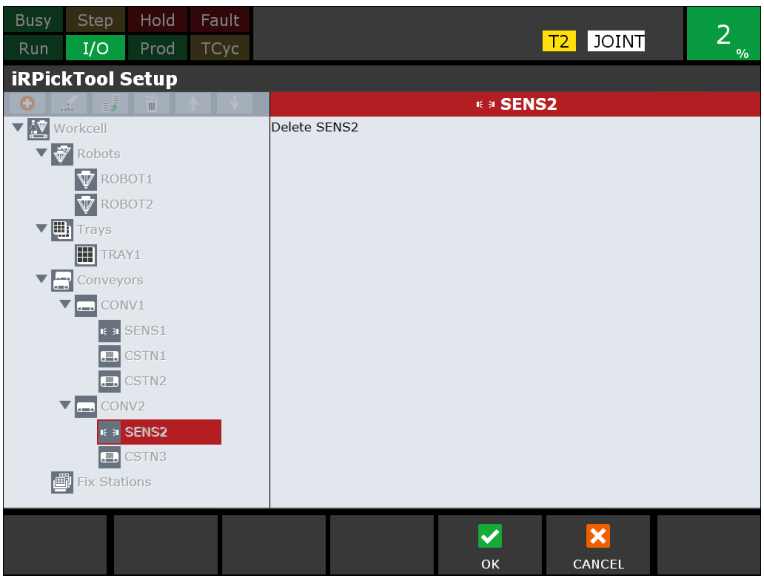


6.14.4 Setup of a Conveyor

In pre-grouping you always setup two conveyors: the infeed conveyor and the pre-grouping conveyor. In the real-world, both the conveyors are the same physical conveyors. These two conveyors use the same tracking frame. However, here is no need to teach the tracking frame for each conveyor. It is sufficient to teach it for the pre-grouping conveyor.

1 The object of an infeed conveyor is set up. Refer to 6.4 “SETUP OF A CONVEYOR”. As mentioned above, you need not set up a tracking frame for the infeed conveyor. There are no other special procedures.

2 The object of a pre-grouping conveyor is set up. Under the object of a pre-grouping conveyor, a sensor object cannot be created. A setup of a pre-grouping cannot be performed in a conveyor object with a sensor object. When a conveyor object is created to use a pre-grouping function, the sensor object generated automatically must be deleted manually.



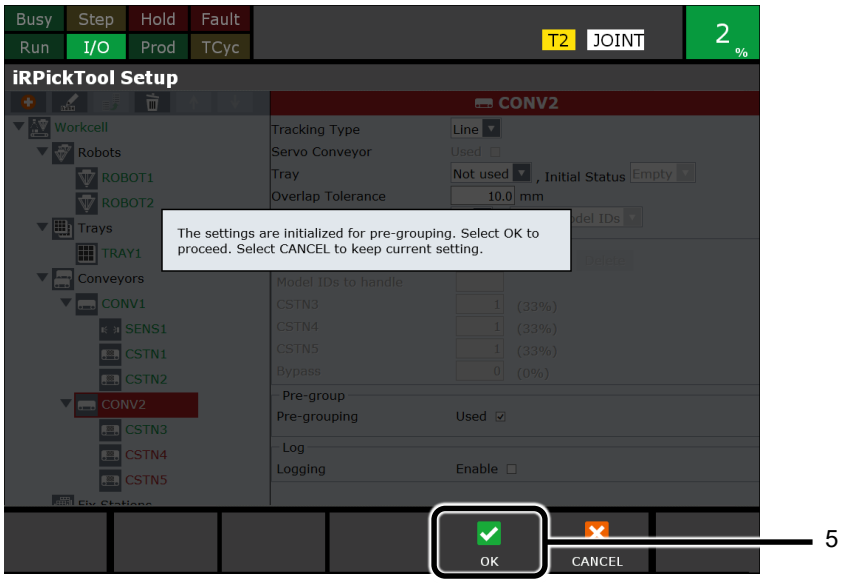
3 Then, add a required number of conveyor station objects.

4 Check the [Used] checkbox of the [pre-grouping] in the [Pre-group] item.

NOTE

You cannot do pre-grouping with circular conveyors or with Servo conveyors. Therefore, when [Circle] is selected as the [Tracking Type] or the [Used] checkbox of the [Servo Conveyor] is checked, the setting item of a [Pre-group] is not displayed.

If it confirms [Used] by a [pre-grouping], the following confirmation screens will be displayed.



5 Press F4 [OK] key to display the setting item of a pre-grouping.

NOTE

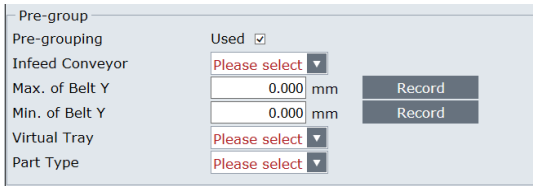
1 Setting items initialized for pre-groupings are the [Load Balance] and the [Y Sort] of child conveyor station objects. The settings of [Load Balance] are initialized to [No] and [For all model IDs]. The [Y Sort] of child conveyor station objects are initialized to be disabled.

2 In the pre-grouping conveyor, settings of the [Tracking Type], the [Servo Conveyor], the [Tray] and the [Load Balance] cannot be changed.

3 When plural robots perform arrangement on the same conveyor with using a load balance function, the [Load Balance] must be set in the setting menu of an infeed conveyor.

6 Set up other items than a [Pre-grouping] according to 6.4 “SETUP OF A CONVEYOR”.

[Pre-group]



[Infeed Conveyor]

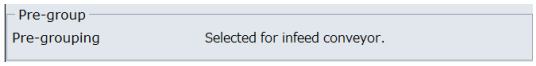
Select the infeed conveyor object.

NOTE

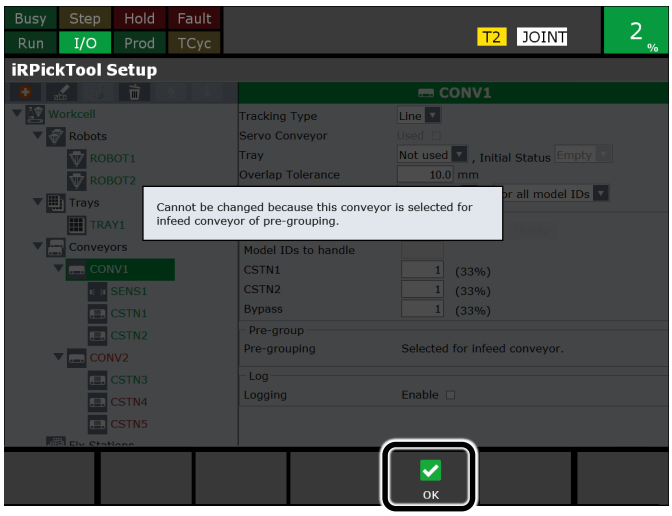
Only the conveyor objects, which meet the following conditions, are displayed as options of the [Infeed Conveyor].

- The [Line] is selected as the [Tracking Type].
- The [Used] checkbox of the [Servo Conveyor] is not checked.
- The [Not used] is selected as the [Tray].
- The [Used] checkbox of the [Pre-grouping] is not checked.

The setting of a [Pre-grouping] is shown in the setting menu of the conveyor object selected by [Infeed Conveyor], as the next figure.



Moreover, in the setting menu of the conveyor object selected by [Infeed Conveyor], settings of the [Tracking Type], the [Servo Conveyor] and the [Tray] cannot be changed. The following messages will be displayed if you try to change.



Press F4 [OK] key to return.

When the [Infeed Conveyor] is selected and there are some child conveyor stations of the pre-grouping conveyor which select the same robot as the child conveyor stations of the infeed conveyor, a tracking schedule can be set only on the setting menu of the child conveyor stations of the infeed conveyor.

> ⚠ **CAUTION**
> If the [Infeed Conveyor] is selected, the [Line Tracking Schedule] of each child conveyor station of the pre-grouping conveyor might be modified. In the child conveyor station objects of the pre-grouping conveyor, if the same robot is selected as one of child conveyor station objects of the infeed conveyor, the tracking schedule number of the child conveyor station objects of the pre-grouping conveyor will be modified to the same value of the child conveyor station objects of the infeed conveyor.

When the same robot is selected in the child conveyor station objects in both the pre-grouping conveyor and the infeed conveyor, a tracking schedule number and an encoder number must be selected in the setting menu of the child conveyor station object in the infeed conveyor. When the robot is selected in the child conveyor station of only one side conveyor, a tracking schedule number and an encoder number should be set in the setting menu of the conveyor station where that robot is selected. Then, in the setting menu of the pre-grouping conveyor, press F4 [SET TRK FRM] key to start the setup of a tracking frame.Refer to 3.2.4 “Setting Up a Tracking Frame”
Setup of the second robot's tracking frame will start when the setup for the first robot's tracking frame is finished.

[Max. of Belt Y]

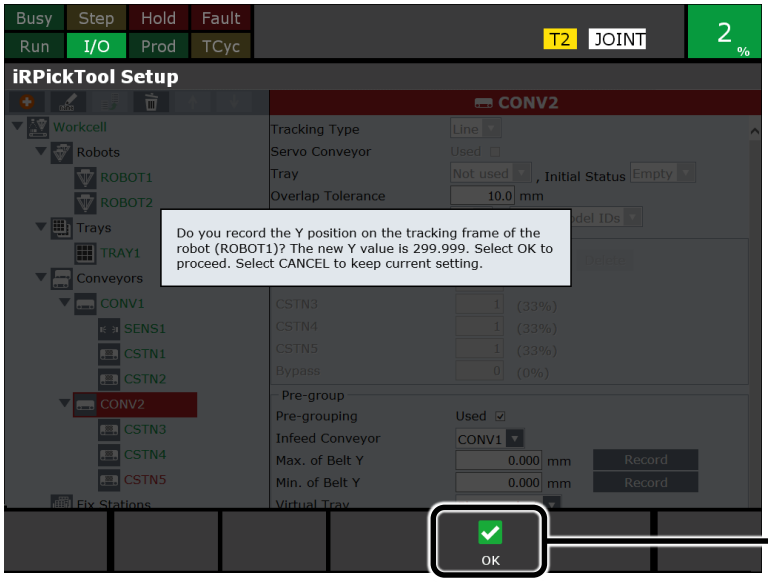
This is the upper limit of the direction of Y which can generate a virtual tray. This value is based on a tracking frame.

NOTE
A virtual tray is generated so that not the model origin point of the vision but whole parts may not overrun this upper limit.

Select the tool frame set up by “5.4 SETTING UP THE TOOL FRAME” as a jog frame and move the TCP by jog and tap the [Record] button, then the Y value in a tracking frame will be set. For example, touch up the end of a left-hand side of the conveyor belt to the direction of movement of a conveyor, in order to set this Y value,.

⚠ CAUTION
1 You must complete the setup of the tracking frame before this setup.
2 You must touch up with the robot connected with the controller whose teach pendant has been opened with the *i*RPickTool treeview.

1 After you tap of the [Record] button, the Y value of a current robot position in the tracking frame will be displayed.



2 Press F4 [OK] key to set the new value.
Press F5 [CANCEL] key to keep the previous value.

NOTE
If you press the F4 [Record] button when there is no conveyor station containing the robot connected to the controller whose teach pendant is being used to teach, the warning of IRPK-065 “Missing my conveyor station” will occur.

[Min. of Belt Y]

This is a lower limit of the direction of Y which can generate a virtual tray. This value is based on a tracking frame.

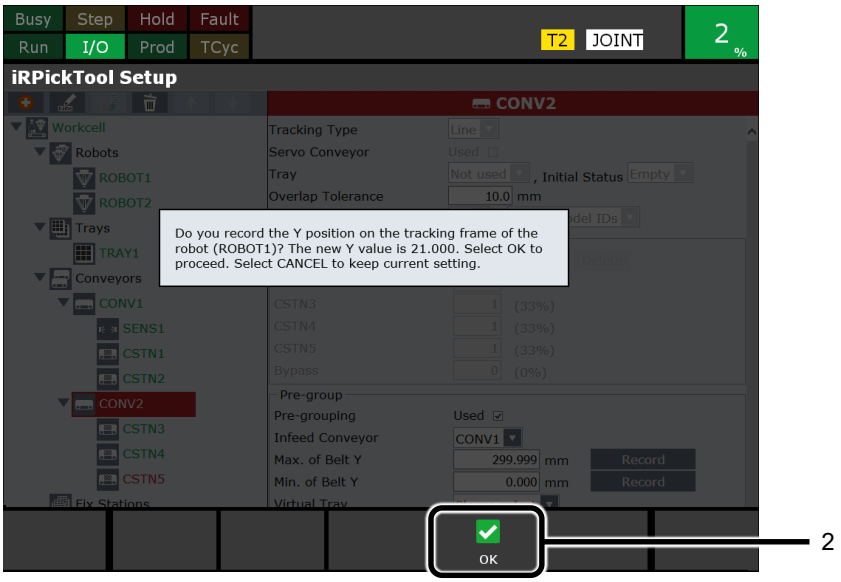
NOTE
A virtual tray is generated so that not the model origin point of the vision but whole parts may not overrun this lower limit.

Select the tool frame set up by 5.4 “SETTING UP THE TOOL FRAME” as a jog frame and move the TCP by jog and tap the [Record] button, then the Y value in a tracking frame will be set. For example, touch

up the end of a right-hand side of the conveyor belt to the direction of movement of a conveyor, in order to set this Y value,.

> **⚠ CAUTION**
> 1 You must complete the setup of the tracking frame before this setup.
> 2 You must touch up with the robot connected with the controller whose teach pendant has been opened with the *i*RPickTool treeview.

1 After you tap of the F4[Record] button, the Y value of a current robot position in the tracking frame will be displayed.



2 Press F4 [OK] key to set the new value.
Press F5 [CANCEL] key to keep the previous value.

> **NOTE**
> If you press the F4 [Record] button when there is no conveyor station containing the robot connected to the controller whose teach pendant is being used to teach, the warning of IRPK-065 “Missing my conveyor station” will occur.

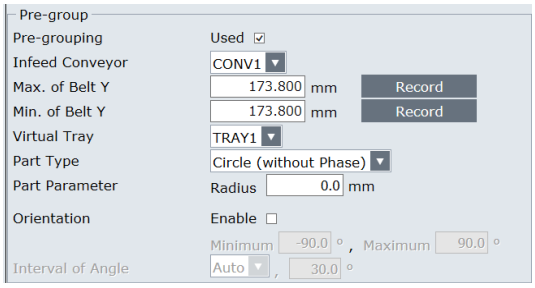
[Virtual Tray]

Select the tray which defines the arranged pattern.

[Part Type]

Select the form of the arranged part.
The [Part Type] which can be selected are [Circle (without Phase)] and [Rectangle]. According to the selected [Part Type], a setting menu changes as follows.

In the case of [Circle (without Phase)],



[Part Parameter]

Set the radius of a circle to the [Radius].

> **⚠ CAUTION**
> If the radius value is smaller than the actual one, the part which the robot picks and places for arrangement may collide with the part on a conveyor due to the incorrect calculation of the interference check between both kinds of the parts. When the form of the actual part is not the circle, you must set the radius of the circle which includes whole of the part.

> **NOTE**
> If the radius is too large for the cell interval of the tray for which the arrangement pattern is set, it may be judged that the base part and arranged parts will collide with each other when PKCSGENVRTRY is called, and the error IRPK-088: "There is an invalid pre-grouping setting." may occur.

[Orientation]

In the case that the [Circle (without Phase)] is selected, virtual trays may be generated in arbitrary directions. When the [Enable] is not checked, a pattern is generated only in one way so that the X direction of a tray frame may become the same as the X direction of a tracking frame as shown below.

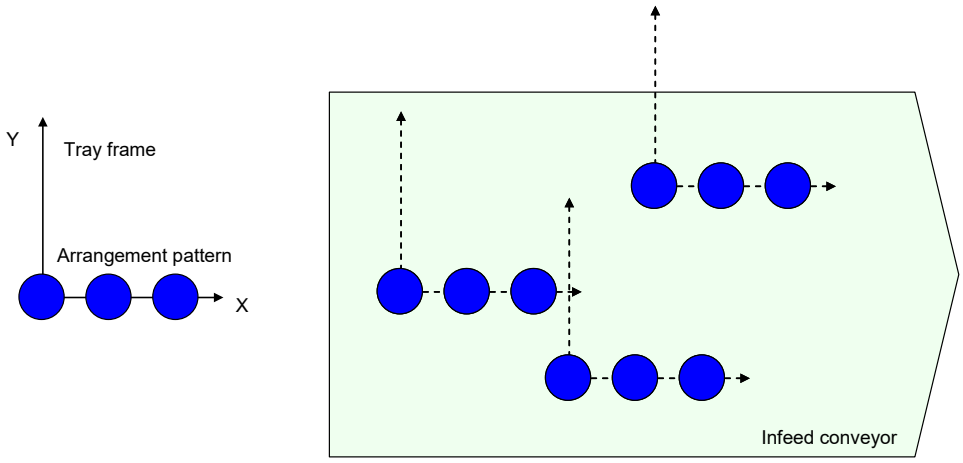


Fig.6.14.4 (a) When orientation is not [Enabled]

When the [Enable] is checked, the X direction of a tray frame will be searched out of plural directions. The X direction of a tracking frame is made into a standard (0 degree), and a direction will be searched in the range which does not exceed the angle set as the [Minimum] and the [Maximum] with the interval of the [Interval of Angle].

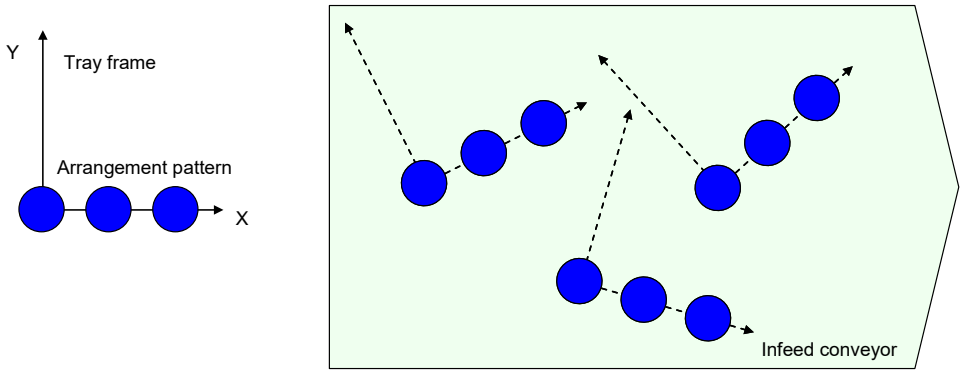
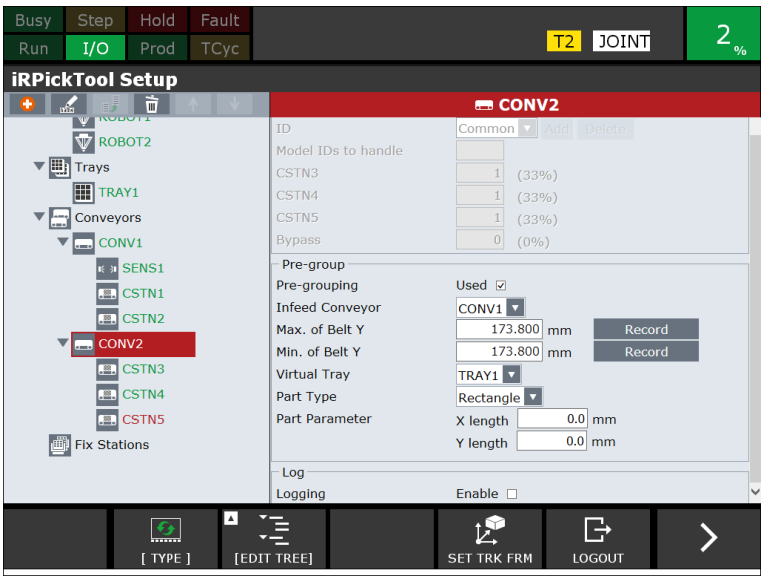


Fig.6.14.4 (b) When orientation is [Enabled]

[Interval of Angle]

When the [Enable] of the [Orientation] is checked, [Interval of Angle] can be set.You can select the [Auto] or the [Specify]. When the [Auto] is selected, the interval is calculated automatically and the virtual tray is searched with the calculated interval. The displayed interval cannot be changed. When the [Specify] is selected, the virtual tray is searched with the interval input into the text box. The range of the value which can be set is from 1 degree to 360 degree.

In the case of [Rectangle],



[Part Parameter]

Set the length of X side on the tray frame to the [X length]. Set the length of Y side on the tray frame to the [Y length].

> **⚠ CAUTION**
> If the X length or the Y length value is smaller than the actual one, the part which the robot picks and places for arrangement may collide with the part on a conveyor due to incorrect calculation of the interference check between both kinds of the parts. When the form of the actual part is not rectangle, you must set the length of both sides of the rectangle which includes whole of the part.

NOTE
If the X length or Y length is too large for the cell interval of the tray for which the arrangement pattern is set, it may be judged that the base part and arranged parts will collide with each other when PKCSGENVRTRY is called, and the error IRPK-088: “There is an invalid pre-grouping setting.” may occur.

This completes the setup of the pre-grouping conveyor.

Finally, set up the conveyor object of the most downstream robot which picks the arranged parts at one time if it places the picked parts on a conveyor. Otherwise, if the picked parts are placed in a Fixed Station, setup the Fixed Station.

Since all the parts in a tray are placed in one operation even when filling a part in the tray which flows through a conveyor, select the [Not used] for the [Tray]. If you fill the plural sets of arranged parts to one tray, then select the tray to which has the number of cells was added, that is equal to the number of sets to be filled in one tray.

Refer to 6.4 “SETUP OF A CONVEYOR” for the other items.

6.14.5 Setup of a Sensor

You need to set up the sensor object which is a child object of the infeed conveyor in the usual manner. Basically, refer to 6.5 “SETUP OF A SENSOR”.
However, for pre-grouping, there are some special notes.

First, if some parts are not detected, the robot may put picked parts on the place where un-detected parts exist for arrangement.

⚠ CAUTION
If some parts are not detected by a vision, the part which robot picks and places for arrangement may collide with the part which is not detected.

NOTE
In the child sensor object of the infeed conveyor used for the arrangement on the same conveyor, when you select the [Distance] as the [Trigger Condition and select the [Find part by vision] as the [Trigger Action], set the [Trigger Distance] as short as possible in order to prevent un-detection with following the CAUTION of the [Trigger distance] in 6.5.1 “Setting a Sensor in Detail”

Next, if the position detected by the vision or the photoeye does not match the center position of the actual part, the part which the robot picks and places for arrangement may collide with the part on a conveyor due to incorrect calculation of the interference check between both kinds of parts. When you refer to 6.5.3.1 “When a tray is not used” under 6.5.3 “Setting the Sensor Position” and 3.2.7.2 “Teaching a GPM locator tool”, be careful of teaching the detection point.

⚠ CAUTION

1 When you select the [Find part by vision] as the [Trigger Action] in the child sensor object of the infeed conveyor used for the arrangement on the same conveyor, the model origin point is set to the center position of a part. If the model origin point does not match the center position of a part, the part which the robot picks and places for arrangement may collide with the part on a conveyor due to incorrect calculation of the interference check between both kinds of the parts.

2 When you select the [Put part in queue] as the [Trigger Action] in the child sensor object of the infeed conveyor used for the arrangement on the same conveyor, the touch-up position in the setup of the sensor position must be center position of a part, not the end of a part on the downstream side (which is equal to the position detected with a photoeye). If the touch-up position does not match to the center position of a part, the part which the robot picks and places for arrangement may collide with the part on a conveyor due to incorrect calculation of the interference check between both kinds of the parts.

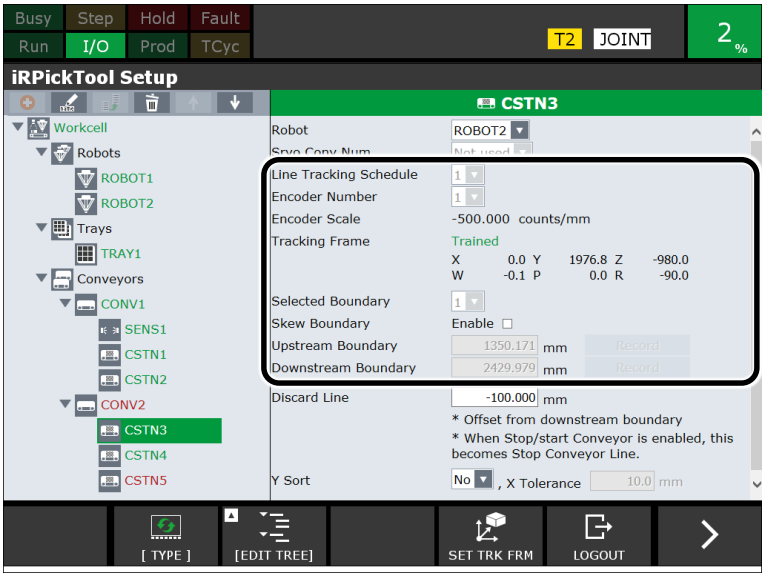
Additionally, if the position is not detected by the vision and the [Rectangle] is selected as the [Part Type] at the pre-grouping conveyor, you should give attention to the order of settings. Setting of the sensor position or the target position must be performed after setting of the [Virtual Tray Position] mentioned later.

⚠ CAUTION

When you do not select the [Find part by vision] as the [Trigger Action] in the child sensor object of the infeed conveyor used for the arrangement on the same conveyor and select [Rectangle] as the [Part type] in the pre-grouping conveyor, you must set the sensor position or the target position after setting the [Virtual Tray Position] mentioned later. Otherwise, the robot will be unable to pick the parts at the correct position.

6.14.6 Setup of a Conveyor Station

When there are child conveyor stations of the infeed conveyor where the same robot is selected as the one which is selected in the child conveyor station of the pre-grouping, a tracking schedule cannot be set in the child conveyor station of the pre-grouping conveyor. It must be set on the setting menu of the child conveyor station of the infeed conveyor. If it is set at the child conveyor station of the infeed conveyor, it will be automatically reflected to the child conveyor station of the pre-grouping conveyor.



[Discard Line]

At the child conveyor station of the pre-grouping conveyor, this is the position which judges whether the robot should give up the placement position on the generated a virtual tray which flows on a conveyor In the case of a pre-grouping, since the part information from the child conveyor station of the infeed conveyor and the placement position for arrangement from the child station of the pre-grouping conveyor are obtained simultaneously, the discard line should be determined from the conveyor speed and the average time from the start of the tracking motion for a part before arrangement to the finish of arrangement of the part. For details, refer to the item of the [Discard line] of 6.6 “SETUP OF A CONVEYOR STATION”.

[Y Sort]

At the child conveyor station of the pre-grouping conveyor, Y-sort function cannot be used. When the [Used] checkbox of the [Pre-grouping] is checked at the parent conveyor, it is automatically disabled and cannot be changed.

[Stop/start Conveyor]

At the child conveyor station of the pre-grouping conveyor, Stop/start Conveyor function cannot be used.

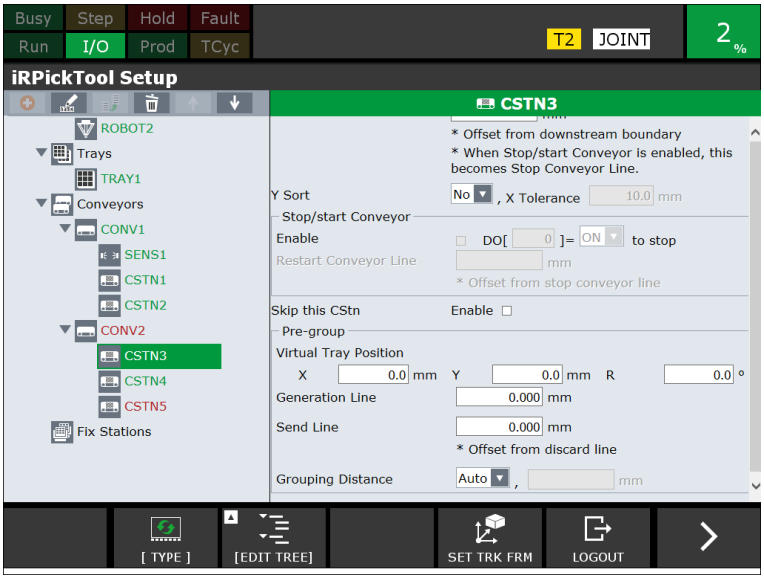
[Skip Station]

If you enable this function at the child conveyor station of the pre-grouping robot in order to give the robot a rest intentionally, you must enable it also in the child conveyor station at the infeed conveyor.

[Pre-group]

When the [Infeed Conveyor] is selected at the pre-grouping conveyor, the setting items of the pre-grouping is additionally displayed on the child conveyor station where the same robot as the child conveyor station just behind the sensor object of the infeed conveyor. There is no sensor in a pre-grouping conveyor. This conveyor station plays the role of a sensor. Using the KAREL program mentioned later, it generates the data of a placement position for arrangement.

In the case that the [Rectangle] is selected as the [Part Type] at the parent conveyor, setting items are as follows.



[Virtual Tray Position]

This is the relative position of the origin of the virtual tray and the detection position of the part.
In the case that the [Circle (without Phase)] is selected as the [Part Type] at the parent conveyor, this item is not displayed because it is computed automatically.
In the case that the [Rectangle] is selected as the [Part Type] at the parent conveyor, perform the procedure described below.

> **⚠ CAUTION**
> If you set the [Put part in queue] as the [Trigger Action] of the sensor in an infeed conveyor, you must perform this procedure before setting the sensor position or the target position. If you have already set either one of the two, you must set it again after performing this procedure.

1 In the pre-grouping conveyor setup page, uncheck the “Used” of [Pre-grouping]. (This is necessary for the next step.)
2 In the infeed conveyor setup page, select the tray which is set as the arrangement pattern of the pre-grouping.
3 In the child sensor setup page on the infeed conveyor, set the tray position, the sensor position or the target position.
- If the [Find part by vision] is set as the [Trigger Action] of the sensor in the infeed conveyor, set the tray position. Refer to 6.5.5 “Setting the Tray Position “.
- If the [Put part in queue] is set as the [Trigger Action] of the sensor in the infeed conveyor, set the sensor position when the [DI] or the [HDI] is set as the [Trigger Condition]. Refer to 3.2.6 “Setting Up a Sensor Position” , 6.5.3 “Setting the Sensor Position” or 6.5.6 “Setting the Target Position”.

In here, touch up the origin or the X direction point of the tray frame as if the detected part is on the cell 1 of the tray.

> **⚠ CAUTION**
> When touching up these points, you must consider that the detected part is on the cell1 of the tray. Otherwise, the part to be placed for arrangement may collide with the part on a conveyor.

4 After the guide completion, make a note of X, Y and R of the tray position, the sensor position or the target position.
5 In the infeed conveyor setup page, select “Not used” in Tray again.

6 In the pre-grouping conveyor setup page, check the “Used” of [Pre-grouping] again. Then, select F4 [OK] key when the message “The settings are initialized for pre-grouping. Select OK to proceed. Select CANCEL to keep current setting.” is displayed. After that, “Infeed Conveyor” becomes initialized. Select “Infeed Conveyor” again.

7 In the child conveyor station setup page on the pre-grouping conveyor, enter X, Y and R of the virtual tray position with referring to the note mentioned in 4.

[Genaration Line]

Specify the position which starts generation of a virtual tray with the X value on a tracking frame (not an offset from upstream boundary).
If you set the [Find part by vision] as the [Trigger Action] of the sensor in an infeed conveyor, set this line on the downstream side further rather than the end on the downstream side of the camera view.
If you set the [Put part in queue] as the [Trigger Action] of the sensor in an infeed conveyor, set this line on the position of the downstream side of the sensor by the following minimum value or more.

- In the case that the [Circle (without Phase)] is selected as the [Part Type] at the parent conveyor, the value is the radius of a circle set at [Part Parameter].
- In the case that the [Rectangle] is selected as the [Part Type] at the parent conveyor, the value is the half length of the diagonal line on the rectangle. This value is computed from the value(x) set at [X lengh] of [Part Parameter] and the value(y) set at [Y length] of [Part Parameter] as shown in Fig. 6.14.6 (a).

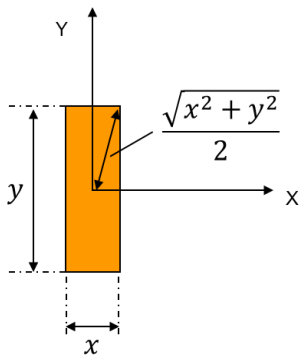
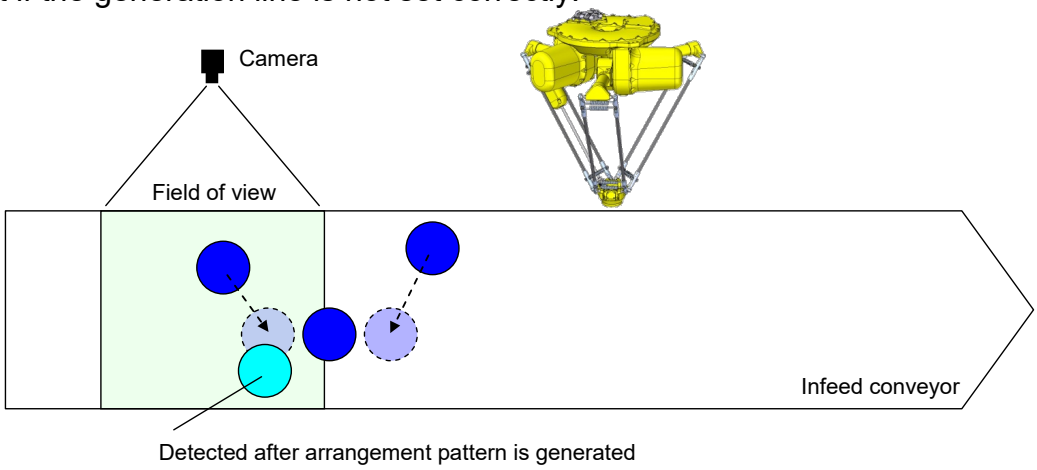


Fig.6.14.6 (a) Calculation of the generation line

⚠ CAUTION

1 When using a vision at an infeed conveyor, if a pattern is generated in a camera view, a part may newly be detected at the position which interferes the generated pattern. In this case, the part that the robot picks and places for arrangement may collide with this newly detected part.

2 Even when not using a vision at an infeed conveyor, a part may be newly detected at the position which interferes the generated pattern, and the part that the robot picks and places for arrangement may collide with this newly detected part if the generation line is not set correctly.



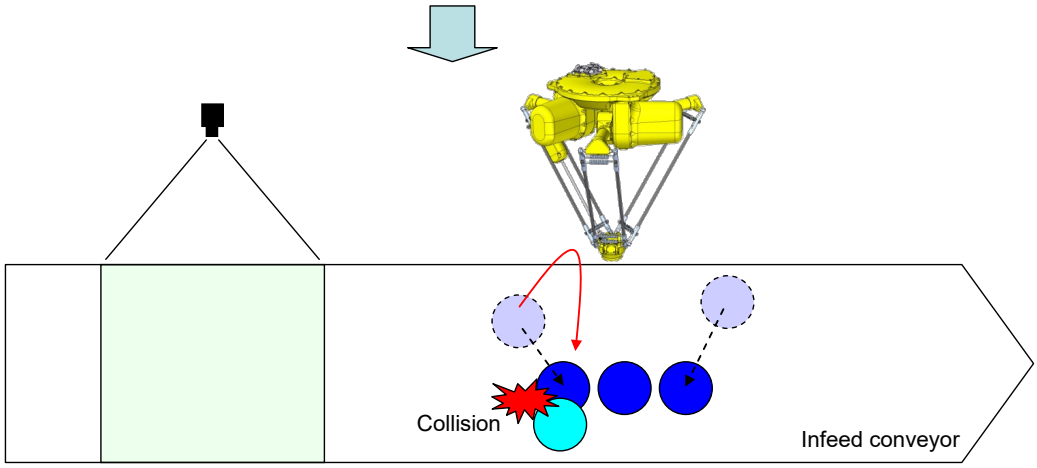


Fig.6.14.6 (b)

[Send Line]

Specify the position at this conveyor station and the corresponding child conveyor station of the infeed conveyor, where the part information and the data of the placement position for arrangement are sent to the next conveyor station. This is an offset from the discard line. Usually, if the [Find part by vision] is selected as the [Trigger Action] and the model origin point passes the discard line, the part information will be sent to the next conveyor station. Moreover, if the [Put part in queue] is selected as the [Trigger Action] and the touch-up position on a part, which is set at the sensor position setup, passes the discard line, the part information will be sent to the next conveyor station. Although the part information is sent to the next conveyor station and deleted from the current conveyor station, a part of the actual part body might not pass the discard line yet. Because a newly generated pattern can interfere with a part of this part body, the part that the robot picks and places for arrangement may collide with it as below.

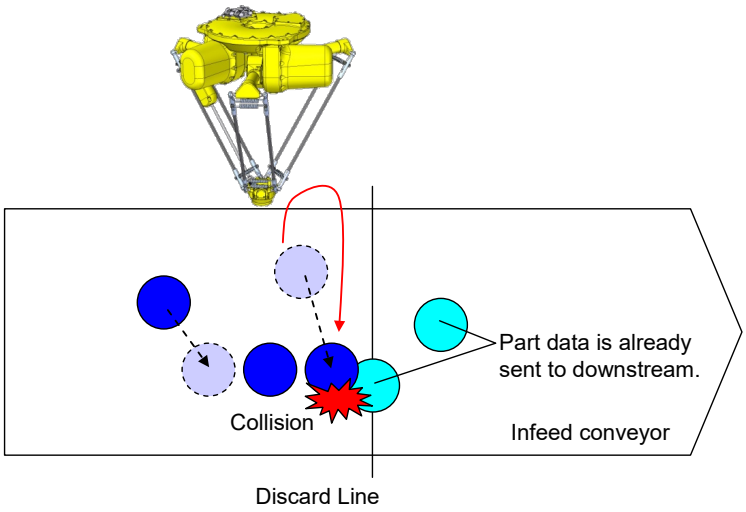


Fig.6.14.6 (c)

Part information is held until passing the Send line even after passing the discard line so that the check of interference may be correctly made at the virtual tray generation.

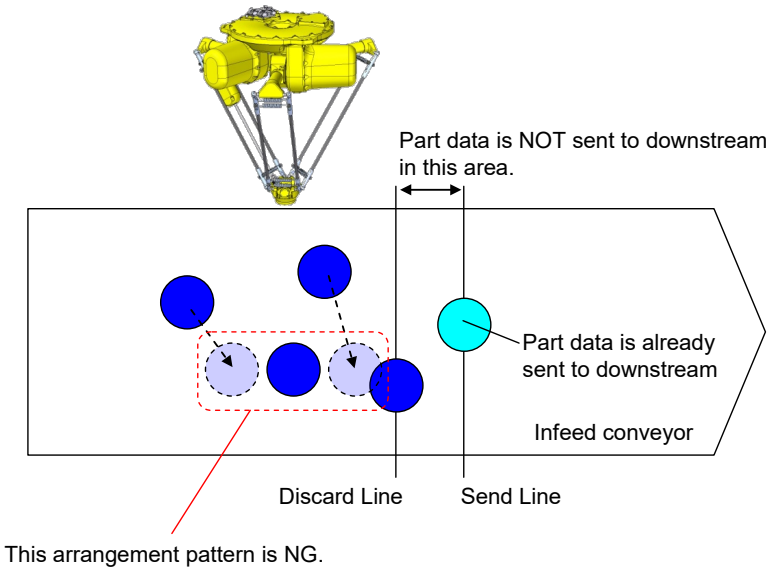


Fig.6.14.6 (d)

A virtual tray is generated at a position in which the whole virtual tray does not pass the discard line on the conveyor in the time of generating a virtual tray. In the case that the [Circle (without Phase)] is set as the [Part Type], specify the radius of a part or larger value than it. In the case that the [Rectangle] is set as the [Part Type], specify the half length of the diagonal line on the rectangle or larger value than it. Refer to Fig.6.14.6 (a)

> ⚠ **CAUTION**
> If the send line is not set correctly, the part that the robot picks and places for arrangement may collide with another parts on a conveyor.

[Grouping Distance]

About the direction of movement of a conveyor, a virtual tray is generated with the parts within a certain distance.
If parts which constitutes a virtual tray separates too much each other about the direction of movement of a conveyor, even if the robot picks up a part for arrangement, the corresponding placement position can already pass a tracking area and the robot cannot place the part.

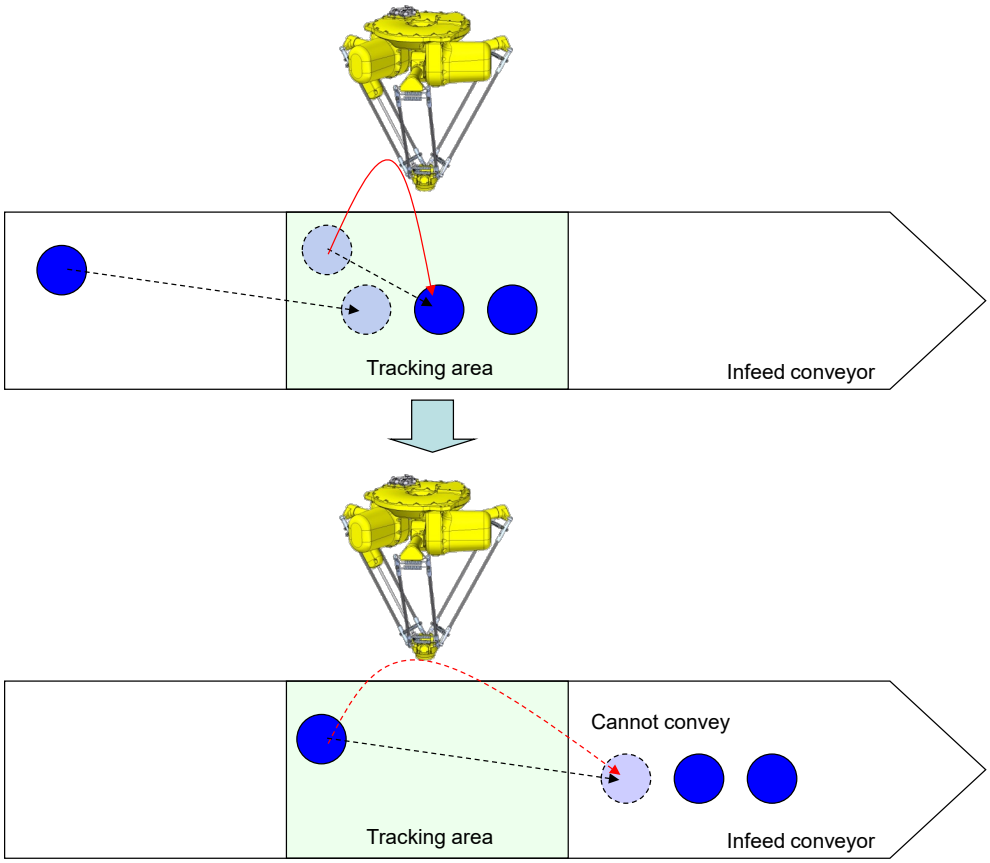


Fig.6.14.6 (e)

When the [Auto] is selected, the distance between the discard line and the upstream boundary of the child conveyor station of the infeed conveyor is used. When the [Specify] is selected, the distance specified in the text box is used.

[Spatial relationship of each item]

The spatial relationship of the “Generation Line”, the “Send Line”, the “Upstream boundary”, the “Downstream boundary” and the “Discard Line” is summarized as the following figure. However, about the “Send Line” and the “Downstream boundary”, as a result of the setup according to this manual, the “Send Line” may exist more downstream than the “Downstream boundary”.

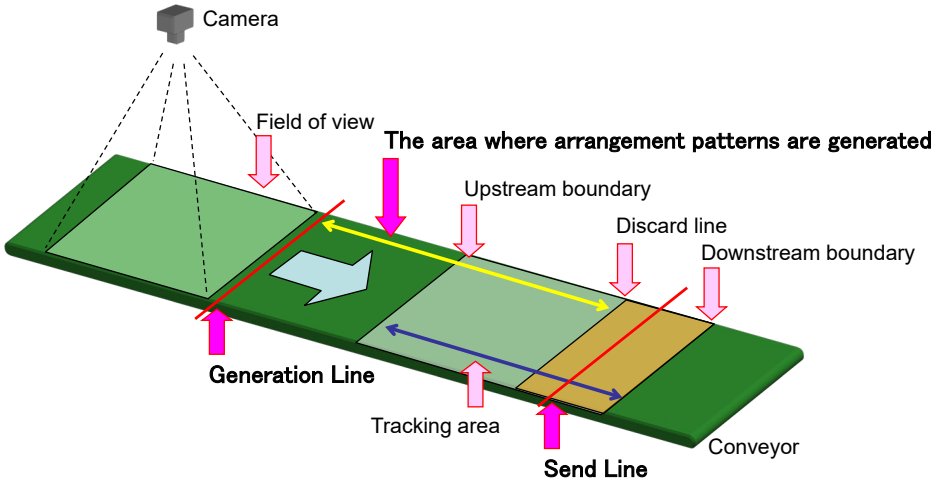


Fig.6.14.6 (f)

6.14.7 Setup of a Fixed Station

When you use a fixed station, refer to 6.7 “SETUP OF A FIXED STATION.”

6.14.8 KAREL Programs

This subsection explains the KAREL programs for a pre-grouping function. All of them are related to a conveyor station.

6.14.8.1 PKCSGENVRTRY.PC

This program generates one virtual tray at the specified conveyor station.

Argument 1:

Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described in 4.8.4.1 “PKCSGETID.PC”. You can specify only the child conveyor station of the pre-grouping conveyor, of which the corresponding conveyor station of the infeed conveyor is placed just behind the sensor object. When you specify the conveyor station which does not belongs to the pre-grouping conveyor, it will cause the error of IRPK-087 “This is dedicated instruction for pre-grouping.” Moreover, when you specify the conveyor station of which the corresponding conveyor station of the infeed conveyor is not placed just behind the sensor object, it will cause the error of IRPK-035 “Invalid conveyor station specified.”

Argument 2:

Specify the number of register storing the processing status. The following statuses may be stored.

0: One virtual tray is generated.
1: A virtual tray cannot be generated because of interference.
2: A virtual tray cannot be generated because the distance between parts is larger than a specified value or the automatically calculated value (which is the distance between the discard line of the conveyor station of the infeed conveyor and the upstream boundary).
3: A virtual tray cannot be generated because less than the number of parts passes the generation line than necessary to generate a virtual tray.

Argument 3 (optional):

Specify the model ID to generate a virtual tray with a specific model ID. This argument is optional and normally does not need to be specified.

Example:

When the ID of the conveyor station of the pre-grouping conveyor is stored in R[12], the following instruction generates a virtual tray at the conveyor station of a pre-grouping conveyor and stores the processing status in R[2].

```
CALL PKCSGENVRTRY(CStn ID=R[12:CStnIDOut],Stat Reg=4)
```

6.14.8.2 PKCSGETQPRGR.PC

This program gets the part information from the conveyor station of the infeed conveyor and the cell information of the corresponding placement position from the conveyor station of the pre-grouping conveyor. The acquired information of the part and the cell are stored in Vision Registers. When the load balance is enabled in the infeed conveyor, each information is obtained according to the settings of the load balance.

Argument 1:

Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described in 4.8.4.1 “PKCSGETID.PC”. You can specify only the conveyor station of the pre-grouping conveyor. When you specify the conveyor station which does not belongs to a pre-grouping, it will cause the error of IRPK-087 “This is dedicated instruction for pre-grouping.”

Argument 2:

Specify the number of the Vision Register in which to store the information of the acquired part information.

Argument 3:

Specify the number of the Vision Register in which to store the information of the acquired cell information of the corresponding placement position. When you specify the same number as the argument 2, it will cause the error of IRPK-057 “Specified value in argument is invalid.”

Argument 4:

Specify the number of register storing the processing status. The following statuses may be stored. Only either the part information or the cell information is not acquired

0: Both the part information and the cell information of the corresponding placement position are acquired.

1: Neither the part information nor the cell information of the corresponding placement position are acquired.

Example:

When the ID of the conveyor station of the pre-grouping conveyor is stored in R[12], the following instruction stores the unprocessed part information from the conveyor station of the infeed conveyor in VR[1], the cell information of the placement position from the conveyor station of the pre-grouping conveyor in VR[2] and the processing status in R[3].

```
CALL PKCSGETQPRGR(CStn ID=R[12:CStnIDOut],Pick Offset VR=1,Drop Offset VR=2,
Stat Reg=3)
```

6.14.8.3 PKCSTRGPRGR.PC

This program sets the value of the encoder as the trigger for the tracking motion if the part information or the cell information to be placed which is acquired by PKCSGETQPRGR is within the tracking area. The robot waits for a specified time before the part information or the cell information to be placed which is acquired reaches the tracking area.

Argument 1:

Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described in 4.8.4.1 “PKCSGETID.PC”. You can specify only the conveyor station of the pre-grouping conveyor or the infeed conveyor. When you specify the other conveyor station, it will cause the error of IRPK-087 “This is dedicated instruction for pre-grouping.”

Argument 2:

Specify the time in milliseconds that the robot waits before the part information or the cell information to be placed which is acquired reaches the tracking area. If you specify a negative value, the robot waits infinitely. If you specify 0, the robot performs the next statement without waiting.

Argument 3:

Specify the number of register storing the processing status. The following statuses may be stored.

0: The trigger for the tracking motion is set.

1: The part information or the cell information to be placed which is acquired fails to reach the tracking area within the specified time.

Example:

When the ID of the conveyor station of the infeed conveyor is stored in R[11], the following instruction sets the trigger for the tracking motion on the conveyor station of the infeed conveyor and stores the processing status in R[3]. If the unprocessed part has not reached the tracking area, the robot waits for 100 milliseconds.

```
CALL PKCSTRGPRGR(CStn ID=R[11:CStnIDIn],Timeout (ms)=100,Stat Reg=3)
```

6.14.8.4 PKCSACKQPRGR.PC

After the processing status of PKCSGETQPRGR becomes 0 and, the part information and the cell information of the corresponding placement position is allocated, this program notifies the conveyor station of the processing result that has been performed for the part and the cell to be placed.

Argument 1:

Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described in 4.8.4.1 “PKCSGETID.PC”. You can specify only the conveyor station of the pre-grouping conveyor. When you specify the conveyor station which does not belongs to a pre-grouping, it will cause the error of IRPK-087 “This is dedicated instruction for pre-grouping.”

Argument 2:

Specify one of the following values to indicate the processing result.

1: “Success”
Specify this value when the part has been placed to the corresponding placement position normally.

2: “Return”
Specify this value when the part is returned to the conveyor station before being handled.

3: “Skip”
Specify this value when the part has not been handled as scheduled. In this case, the information on this part is moved to the next conveyor station. This applies, for example, when the handling of the part is canceled intentionally after the position or model ID of the allocated part is evaluated by the TP program of the robot. In this case, the information of this part is passed to the next conveyor station.

4: “Remove”
Specify this value when the handling of the part fails. In this case, this part information is deleted. The part, which belongs to the same virtual tray and had not been handled yet, is reused. when PKCSGENVRTRY is called again. Moreover, the downstream robot which takes out the

arranged parts at one time is able to take out the incompletely arranged parts with using PKCSGETQCELL.

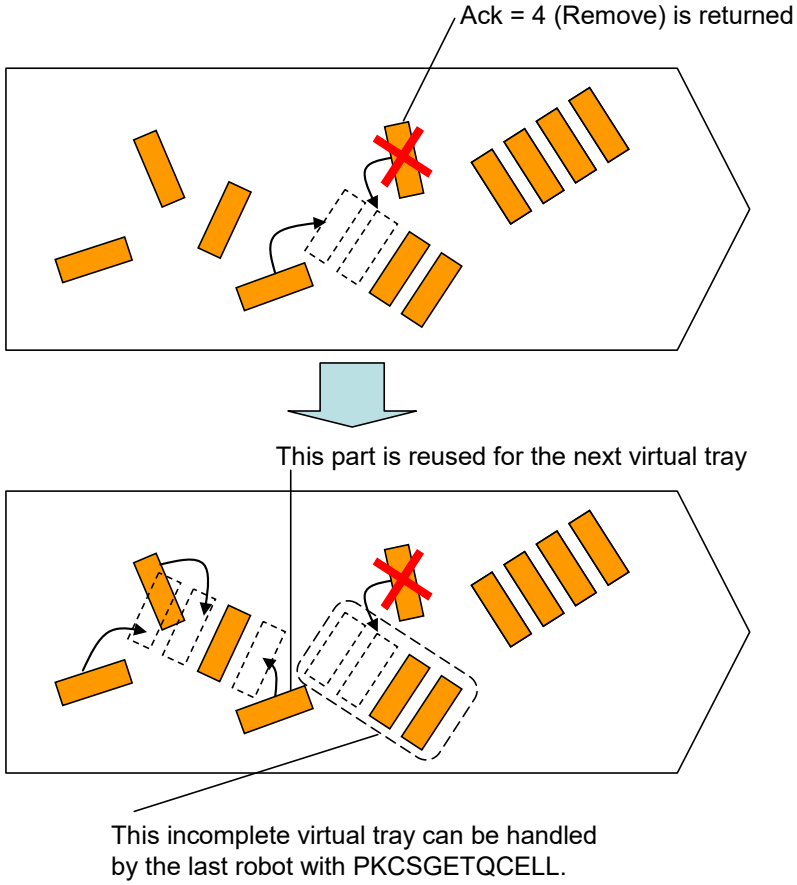


Fig.6.14.8.4(a)

Example:

When the ID of the conveyor station is stored in R[12], the following instruction notifies the part and the cell to be placed have been processed successfully.

```
CALL PKCSACKQPRGR(CStn ID=R[12:CStnIDOut],Success)
```

6.14.9 Robot Programs

This subsection contains examples of the programs which perform arrangement on the same conveyor and describes the concepts and other details of these programs. Tailor the sample programs provided herein to suit your system.

6.14.9.1 Sample programs for the conveyor-to-the same conveyor configuration (the first robot)

This subsection contains a set of sample programs for the first robot whose position is immediately after a sensor on a conveyor.

Unlike in the case of the general tracking system, pre-grouping tracking system uses the following two types of robot programs.

- Main Program
- Program that performs arrangement motion on the same conveyor (pre-grouping program)

The program that performs arrangement motion is a tracking program called from a main program.
The sample programs presented herein use the registers and Position Registers listed below. Change the register numbers as necessary.

Table 6.14.9.1(a) Registers

R[1]	Cycle stop flag. Setting 1 in this register causes the system to stop the cycle.
R[2]	PKCSGETQPRGR status
R[3]	PKCSTRGPRGR status
R[4]	PKCSGENVRTRY status
R[11]	Stores the conveyor station ID of infeed conveyor CONV1.
R[12]	Stores the conveyor station ID of pre-grouping conveyor CONV2.

Table 6.14.9.1(b) Position registers

PR[1]	It is on a conveyor and is a position (and placing position). Pickup position on a conveyor (and placement position on the conveyor)
PR[10]	Tool offset to create an approach point for the pickup motion
PR[12]	Tool offset to create an approach point for the placement motion

Main Program

This is the main program of the system.

MAIN1.TP

```
 1:  R[1:CYCLE_STOP]=0
 2:  CALL PKCSGETID('CONV1', 11)
 3:  CALL PKCSGETID('CONV2', 12)
 4:
 5:  UTOOL_NUM=1
 6:  UFRAME_NUM=0
 7:  J P[1:Perch Pos] 50% FINE
 8:
 9:  CALL PKWCSTART
10:
11:  LBL[100]
12:  CALL PKCSGENVRTRY(CStn ID=R[12:CStnIDOut],Stat Reg=4)
13:  IF R[4:GENVRTRY STATUS]=0,JMP LBL[200]
14:  IF R[4:GENVRTRY STATUS]=1,JMP LBL[300]
15:  IF R[4:GENVRTRY STATUS]=2,JMP LBL[400]
16:  WAIT      .02(sec)
17:  IF R[1:CYCLE_STOP]=1,JMP LBL[900]
18:  JMP LBL[100]
19:
20:  LBL[200]
21:  CALL PREGROUPING1
22:  IF R[1:CYCLE_STOP]=1,JMP LBL[900]
23:  JMP LBL[100]
24:
25:  LBL[300]
26:  CALL PKCSGETQUE(CStn ID=R[11:CStnIDIn],Consec Flag=1,Timeout (ms)=0,
       Offset VR=1,Stat Reg=2)
27:  IF R[2:GETQ STATUS]<>0,JMP LBL[100]
28:  CALL PKCSACKQUE(CStn ID=R[11: CStnIDIn],Skip)
29:  JMP LBL[100]
30:
31:  LBL[400]
32:  CALL PICKCS
```

```
 33:    IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 34:    CALL DROPFS
 35:    IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 36:    JMP LBL[100]
 37:
 38:    LBL[900]
 39:    CALL PKWCEND
[End]
```

A virtual tray is generated on line 12. If a virtual tray is generated, the program jumps to line 20 and calls PREGROUP1.TP which performs arrangement motion on a conveyor. PREGROUP1.TP ends only after performing arrangement operation as much as possible. If PREGROUP1.TP ends, the next virtual tray is generated after the cycle stop flag is checked.
When the processing status is 1, that is, a new virtual tray is not generated because of interference, the program jumps to line 25 and discard one most downstream part intentionally. In this case, since the density of parts on the conveyor is high, there is a high possibility that the robot cannot handles all the parts, the program discard a part intentionally.
When the processing status is 2, that is, the distance between parts is large, the program jumps to line 31, one part is picked from the conveyor, and it is placed on a fixed station. In this case, since the density of parts on the conveyor is low, there is a low possibility that the robot discard some parts even if the robot performs long motion path between the conveyor and the fixed station. Refer to 6.9.3 “Sample Programs for the Conveyor-to-Fixed Station Configuration” for the details about PICKCS.TP and DROPFS.TP.
When the processing status is 3, that is, the number of parts which have passed the generation line is less than that in the arranged pattern, the program repeats the instruction to generate a new virtual tray and the wait instruction until enough parts have passed the generation line.

Pre-grouping Program

This program performs arrangement motion on the same conveyor.

PREGROUPING1.TP

```
  1:    UTOOL_NUM=1
  2:    UFRAME_NUM=0
  3:
  4:    LBL[100]
  5:    STOP_TRACKING
  6:    CALL PKCSGETQPRGR(CStn ID=R[12:CStnIDOut],Pick Offset VR=1,
        Drop Offset VR=2,Stat Reg=2)
  7:    IF R[2:GETQ STATUS]=0,JMP LBL[200]
  8:    JMP LBL[800]
  9:
 10:    LBL[200]
 11:    CALL PKCSTRGPRGR(CStn ID=R[11:CStnIDIn],Timeout (ms)=100,Stat Reg=3)
 12:    IF R[3:TRGPRGR STATUS]=0,JMP LBL[300]
 13:    IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 14:    JMP LBL[200]
 15:
 16:    LBL[300]
 17: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
 18: L PR[1] 4000mm/sec CNT0 VOFFSET,VR[1]
 19:    CALL VACUUM_ON
 20: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
 21:
 22:    LBL[400]
 23:    STOP_TRACKING
 24:    CALL PKCSTRGPRGR(CStn ID=R[12:CStnIDOut],Timeout (ms)=100,Stat Reg=3)
```

```
  25:    IF R[3:TRGPRGR STATUS]=0,JMP LBL[500]
  26:    JMP LBL[400]
  27:
  28:    LBL[500]
  29: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[2] Tool_Offset,PR[12]
  30: L PR[1] 4000mm/sec CNT0 VOFFSET,VR[2]
  31:    CALL VACUUM_OFF
  32: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[2] Tool_Offset,PR[12]
  33:    CALL PKCSACKQPRGR(CStn ID=R[12:CStnIDOut],Success)
  34:    JMP LBL[100]
  35:
  36:    LBL[800]
  37:    END
  38:
  39:    LBL[900]
  40:    CALL PKCSACKQPRGR(CStn ID=R[12:CStnIDOut],Return)
[End]
```

On line 6, the part data and the corresponding cell data are obtained. If data are obtained successfully, the program jumps to line 10. If data are not obtained, the program jumps to line 37 and ends.

On line 11, the tracking trigger for pickup motion is set. Since the wait time is set to 100 milliseconds in the 2nd argument of PKCSTRGPRGR, a timeout occurs in 100 milliseconds if the part has not come into a tracking area when PKCSTRGPRGR is called. In that case, R[3] contains a value 1, and the program checks the cycle stop flag, calls PKCSTRGPRGR again, and waits for the part to come into a tracking area. When the cycle stop flag is 1, the program jumps to line 39, and ends after calling PKCSACKQPRGR for return the part data and corresponding cell data.

The pickup motion is performed on lines 17 to 20.

On line 24, the tracking trigger for placement motion is set. Since the wait time is set to 100 milliseconds in the 2nd argument of PKCSTRGPRGR, a timeout occurs in 100 milliseconds if the place where the picked part is placed has not come into a tracking area when PKCSTRGPRGR is called. In that case, R[3] contains a value 1, and the program checks the cycle stop flag, calls PKCSTRGPRGR again, and waits for the place where the picked part is placed to come into a tracking area.

The placement motion is performed on lines 29 to 32. The same Position Register is used as pickup motion.

PKCSACKQPRGR is called on line 33 to notify the conveyor station that the arrangement motion is complete. And then the program jumps to line 4 and obtains the next part data and corresponding cell data.

6.14.9.2 Sample programs for the conveyor-to-the same conveyor configuration (downstream robots)

This sub-subsection contains a set of sample programs for the downstream robot whose position is not immediately after a sensor on a conveyor. The difference from the first robot whose position is immediately after a sensor on a conveyor is the point that this robot does not generate a virtual tray. This sub-subsection only contains the main program of the system because only main program is different from the sample programs showed in "6.14.9.1 Sample programs for the conveyor-to-the same conveyor configuration (the first robot)."

Main Program

This is the main program of the system.

MAIN1.TP

```
  1:   R[1:CYCLE_STOP]=0
  2:   CALL PKCSGETID('CONV1',CStn ID Reg=11)
  3:   CALL PKCSGETID('CONV2',CStn ID Reg=12)
  4:
  5:   UTOOL_NUM=1
  6:   UFRAME_NUM=0
  7:   J P[1:Perch Pos] 50% FINE
  8:
  9:   CALL PKWCSTART
 10:
 11:   LBL[100]
 12:   CALL PREGROUPING1
 13:   IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 14:
 15:   CALL PICKCS
 16:   IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 17:   CALL DROPFS
 18:   IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 19:   JMP LBL[100]
 20:
 21:   LBL[900]
 22:   CALL PKWCEND
[End]
```

PREGROUP1.TP is called on line 12 to arrange parts on the same conveyor. PREGROUP1.TP ends only after performing arrangement operation as much as possible. PREGROUP1.TP is the same as "Pre-grouping Program" shown in "6.14.9.1 Sample programs for the conveyor-to-the same conveyor configuration (the first robot)."

When there is no acquirable data and arrangement operation is ended, PICKCS.TP is called on line 15 to pickup a part discarded by the first robot. DROP.FS is called on line 17 to place the picked part on the fixed station. Refer to 6.9.3 "Sample Programs for the Conveyor-to-Fixed Station Configuration" for the details about PICKCS.TP and DROPFS.TP.
After handling one part discarded by the first robot, the program jumps to line 11 and calls PREGROUP1.TP again.

6.14.9.3 Sample programs for the conveyor-to-conveyor configuration

This sub-subsection contains a set of sample programs for the last robot that picks arranged parts at the same time and places them out to another conveyor. This sub-subsection only contains the program which performs a pickup motion on an infeed conveyor because only pick program is defferent from the sample programs showed in 6.9.2 "Sample Programs for the Conveyor-to-Conveyor Configuration". Refer to 6.9.2.1 "Main program" and 6.9.2.3 "Placement program" for the main program of the system and the program which performs placement motion on an outfeed conveyor, respectively.

Pick Program

This section explains about a program that picks up only parts for which the alignment pattern is complete, and a program that also picks up parts for which the alignment pattern is not complete.

Example of a program that picks up only parts for which the alignment pattern is complete

PICK1.TP

```
  1:   UTOOL_NUM=1
```

```
   2:    UFRAME_NUM=0
   3:
   4:    LBL[100]
   5:    STOP_TRACKING
   6:    CALL PKCSGETQUE( “CStn ID” =R[11:CStnIDIn], “Consec Flag”=1,
         “Timeout (ms)”=100, “Offset VR”=1, “Stat Reg”=2,2) ;
   7:    IF R[2:GETQ STATUS]=0,JMP LBL[200]
   8:    IF R[1:CYCLE_STOP]=1,JMP LBL[900]
   9:    JMP LBL[100]
  10:
  11:    LBL[200]
  12: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
  13: L PR[1] 4000mm/sec CNT0 VOFFSET,VR[1]
  14:    CALL VACUUM_ON
  15: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
  16:    CALL PKCSACKQUE(CStn ID=R[11:CStnIDIn],Success)
  17:
  18:    LBL[900]
[End]
```

The difference from the usual pick program in conveyor-to-conveyor configuration is that the model ID is specified by PKCSGETQUE on line 6. In here, model ID 1 is set to all cells in the layer 1 of the virtual tray, and model ID 2 is set to the cell in the layer 2. By specifying the model ID 2 for the argument 6 of PKCSGETQUE, only when all the cells of the layer 1 are filled, information is acquired.

Next, this is a program which performs a pickup motion in the case of picking incompletely arranged parts too.

Example of a program that also picks up parts for which the alignment pattern is not complete

PICK2.TP

```
   1:    UTOOL_NUM=1
   2:    UFRAME_NUM=0
   3:
   4:    LBL[100]
   5:    STOP_TRACKING
   6:    CALL PKCSGETQCELL(CStn ID=R[11:CStnIDIn],Cell num=4,Timeout (ms)=100,
         Offset VR=1,NumUnfilCel Reg=21)
   7:    IF R[2:GETQ STATUS]=0,JMP LBL[200]
   8:    IF R[1:CYCLE_STOP]=1,JMP LBL[900]
   9:    JMP LBL[100]
  10:
  11:    LBL[200]
  12: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
  13: L PR[1] 4000mm/sec CNT0 VOFFSET,VR[1]
  14:    CALL VACUUM_ON
  15: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
  16:    CALL PKCSACKQUE(CStn ID=R[11:CStnIDIn],Success)
  17:
  18:    LBL[900]
[End]
```

The difference from PICK1.TP is the point of using PKCSGETQCELL instead of PKCSGETQUE on line 6. When the arranged pattern contains three parts, specify the cell 4 that is added to the layer 2 for picked

up at the same time by this robot. The number of the unfilled cells in layer1 of an virtual tray is returned to R[21].

6.14.10 Reference Position Setting

Set the reference position on the infeed conveyor and the outfeed conveyor for placing all parts in the virtual tray by the last robot. Refer to 6.10 “REFERENCE POSITION SETTING”.

6.14.11 Teaching the Position to Robots

After setting the reference position, teach the position to robots. In the program which performs arrangement motion, teaches only the pickup position. Since the same position data is used, it is unnecessary to teach the placement position. Refer to 6.11 “TEACHING THE POSITION TO ROBOTS”.

6.14.12 Fine Adjustment of The Tracking Motion

Refer to 6.12 “FINE ADJUSTMENT OF THE TRACKING MOTION”. As special affairs in the case of a pre-grouping, care needs to be exercised when adjusting the tracking frame error.

6.14.12.1 Adjustment of the tracking frame error

When using ADJ_OFS, modify the program which performs arrangement motion shown in 6.14.9.1 “Sample programs for the conveyor-to-the same conveyor configuration (the first robot)” as follows. In the sample program shown below, Position Register 20 is used to store the amount of adjustment. Change it as necessary.

PREGROUPING1.TP

```
 1:   UTOOL_NUM=1
 2:   UFRAME_NUM=0
 3:
 4:   LBL[100]
 5:   STOP_TRACKING
 6:   CALL PKCSGETQPRGR(CStn ID=R[12:CStnIDOut],Pick Offset VR=1,
       Drop Offset VR=2,Stat Reg=2)
 7:   IF R[2:GETQ STATUS]=0,JMP LBL[200]
 8:   JMP LBL[800]
 9:
10:   LBL[200]
11:   CALL PKCSTRGPRGR(CStn ID=R[11:CStnIDIn],Timeout (ms)=100,Stat Reg=3)
12:   IF R[3:TRGPRGR STATUS]=0,JMP LBL[300]
13:   IF R[1:CYCLE_STOP]=1,JMP LBL[900]
14:   JMP LBL[200]
15:
16:   LBL[300]
17:   CALL ADJ_OFS(1, 1, 20, 3)
18: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[3] Tool_Offset,PR[10]
19: L PR[1] 4000mm/sec CNT0 VOFFSET,VR[3]
20:   CALL VACUUM_ON
21: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[3] Tool_Offset,PR[10]
22:
23:   LBL[400]
24:   STOP_TRACKING
25:   CALL PKCSTRGPRGR(CStn ID=R[12:CStnIDOut],Timeout (ms)=100,Stat Reg=3)
26:   IF R[3:TRGPRGR STATUS]=0,JMP LBL[500]
```

```
  27:    JMP LBL[400]
  28:
  29:    LBL[500]
  30:    CALL ADJ_OFS(1, 2, 20, 4)
  31: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[4] Tool_Offset,PR[12]
  32: L PR[1] 4000mm/sec CNT0 VOFFSET,VR[4]
  33:    CALL VACUUM_OFF
  34: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[4] Tool_Offset,PR[12]
  35:    CALL PKCSACKQPRGR(CStn ID=R[12:CStnIDOut],Success)
  36:    JMP LBL[100]
  37:
  38:    LBL[800]
  39:    END
  40:
  41:    LBL[900]
  42:    CALL PKCSACKQPRGR(CStn ID=R[12:CStnIDOut],Return)
[End]
```

- Just before the pickup motion and the placement motion, ADJ_OFS is called for adjusting the vision offset.
- The same amount of adjustment stored in the same Position Register is used for both the pickup motion and the placement position.
- When setting up the amount of adjustment, insert a motion instruction and a wait instruction immediately after the approach point as follows so as to make the robot stop immediately above the pickup position,.

```
  17:    CALL ADJ_OFS(1, 1, 20, 3)
  18: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[3] Tool_Offset,PR[10]
  19: L PR[1] 4000mm/sec FINE VOFFSET,VR[3] Tool_Offset,PR[15]
  20:    WAIT   10.00(sec)
  21: L PR[1] 4000mm/sec CNT0 VOFFSET,VR[3]
  22:    CALL VACUUM_ON
```

- Refer to 6.12.2 “Adjustment of the Tracking Frame Error” for details.

7 PLUG & PLAY REFERENCE

This chapter explains the details and setup procedures for Plug & Play menu items, for reference when using *i*RPickTool (Basic Functions + Plug & Play).
Furthermore, explanations of specific programs such as standard TP programs and KAREL programs are also given.

For *i*RPickTool (Basic Functions + Plug & Play) startup procedure, refer to Chapter 4, "*i*RPICKTOOL (BASIC FUNCTIONS + PLUG & PLAY) STARTUP PROCEDURES".

The *i*RPickTool screen menu and related items in this manual are as shown in the figure below.
Among the items below, Section 7.1, "SETUP OF A WORKCELL" details the maximum number of Plug & Play objects. Section 7.4, "SETUP OF A FIXED STATION" explains the [Skip] settings that can only be set with Plug & Play.
Section 7.2, "SETUP OF A GRIPPER AND A ZONE" and Section 7.3, "SETUP OF A ROBOT OPERATION" are items that can only be set with Plug & Play. For other items, refer to the corresponding items in Chapter 6, "BASIC FUNCTIONS REFERENCE".

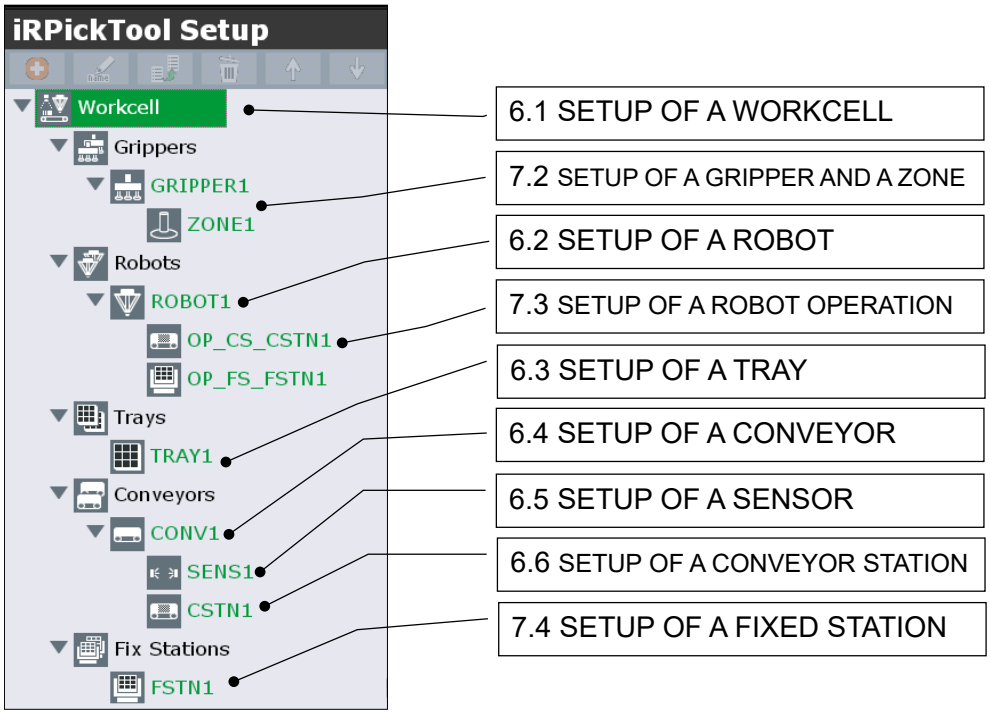


Fig. 7(a) *i*RPickTool menu and related items

Besides the menu items in Fig. 7(a), as a reference for Plug & Play, Section 7.5 "ROBOT PROGRAMS", Section 7.9, "KAREL PROGRAMS", and Section 7.10 "STANDARD TP PROGRAMS" explain the their corresponding programs. Section 7.6 "REFERENCE POSITION SETTING" and Section 7.7 "TEACHING THE POSITION TO ROBOTS" explain robot position teaching.

Furthermore, Section 7.8 "FINE ADJUSTMENT OF THE TRACKING MOTION" explains the method to adjust errors, in particular with Plug & Play standard TP programs.

Besides these, Section 7.11 "SETUP OF PRE-GROUPING" explains content that has been specialized for the pre-grouping function.
Section 7.12, "BUFFER FUNCTION" explains the buffer function used with Plug & Play.

> ⚠ **CAUTION**
> As a software option for all robot controllers that are connected in the robot ring setup, order *i*RPickTool and not *i*RPickTool Basic.

Basic operations

With *i*RPickTool, do the setup separately for each configuration object, such as each robot and each conveyor.
For the basic operation of *i*RPickTool, refer to "Basic operations" under Chapter 6 "BASIC FUNCTIONS REFERENCE".

7.1 SETUP OF A WORKCELL

For the workcell setup screen, refer to Section 6.1, "SETUP OF A WORKCELL".
Furthermore, for the procedure to set up a Plug & Play workcell, refer to Chapter 4, "*i*RPICKTOOL (BASIC FUNCTIONS + PLUG & PLAY) STARTUP PROCEDURES".

Setting the number of objects

In a workcell, you can modify the maximum number of each object.
The maximum numbers that can be modified in Plug & Play are shown in Table 7.1(a).

For the procedure to modify the maximum number of each object, refer to "Setting the number of objects" under Section 6.1, "SETUP OF A WORKCELL".

Table 7.1(a) : Maximum number that can be modified for each object

Object	Maximum number
Gripper	32
Zone	100
Robot	32
Tray	16
Sensor	32
Conveyor	32
Conveyor Station	84
Fix Station	84

7.2 SETUP OF A GRIPPER AND A ZONE

The Gripper Setup is the first step to perform when you have J945: *i*RPickTool. The Gripper Node is the topmost node in the tree-view as shown below.

For the gripper, enter an index delay, etc. that is suitable for the pick / place operation. For the setup of a zone, which is a child object of a gripper, enter items such as tool frame, zone activation I/O, zone part presence I/O, etc.
If you enter these values correctly, you will not need to edit a standard TP program of Plug & Play.

A Gripper is used by a Robot to pick or place parts. A Gripper represents whole hand to pick parts and its child object Zone represents the section to pick a part actually. Following figure represents its concept.

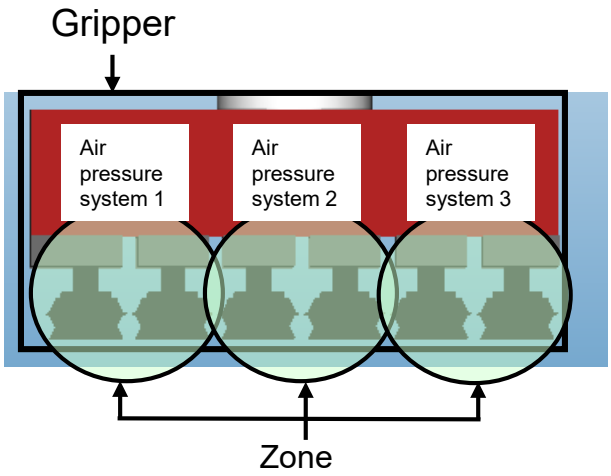
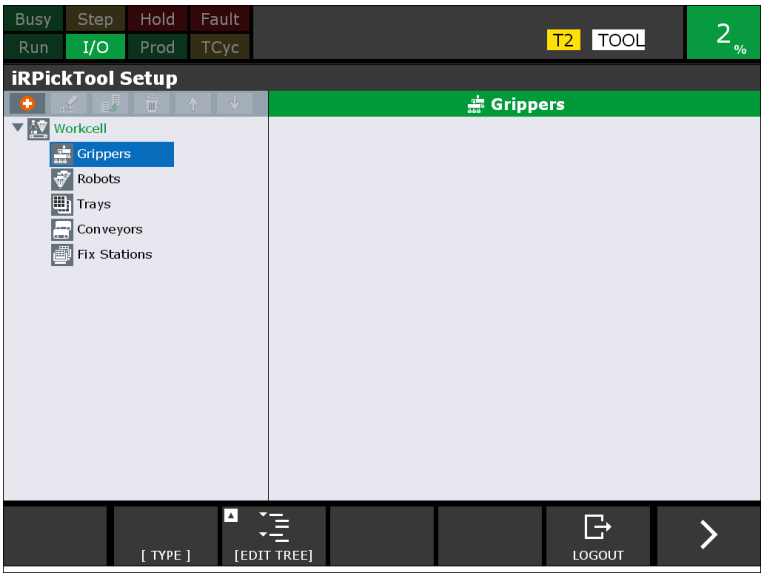


Fig. 7.2(a) Image of gripper and zones

A Gripper contains up to 20 Zones. Each Zone contains a UTOOL, up to 4 part presence inputs, and up to 4 gripper activation outputs. A Zone can be used to pickup a part. A Gripper may pick or place multiple parts using multiple Zones.
Gripper data can be saved to recipes. This allows you to use different Zone UTools, Pick / Place Delays, etc. for each recipe you are running without any custom programming on your part.
For details on recipes, refer to Section 9.1, “EXCHANGE WORKCELL WITH USING RECIPE”.



7.2.1 Gripper and Zone

If you add a gripper, a zone will be added automatically.

Adding a Gripper Zone

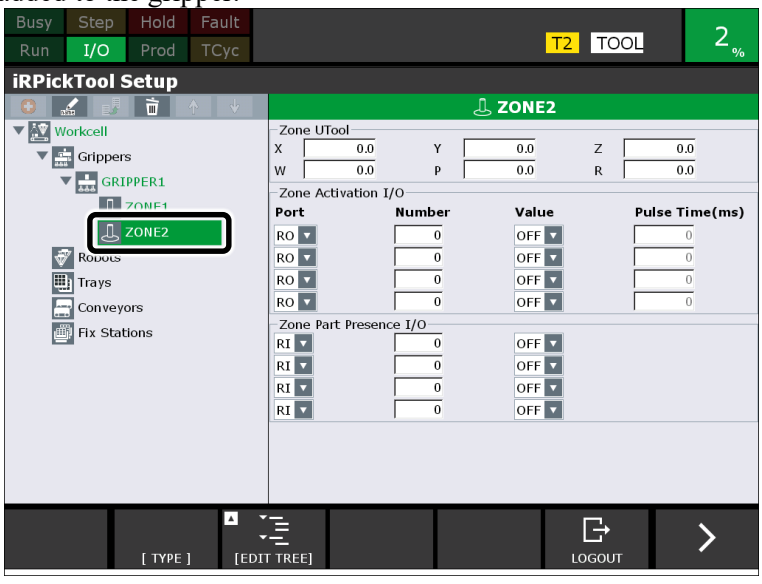
Add a zone to the gripper using the following procedure.

1 Select the [Gripper] that you want to add to.

2 Press the create button above the tree view, or select [CREATE] from [EDIT TREE].
A screen as shown below is displayed.

In [Name], a default name (for ex: ZONE1) is automatically provided. You can enter a different name if you wish to.
Input using half-width alphanumeric characters and half-width katakana. The name must contain no spaces, not contain the symbol other than underscore, and start with a letter.

3 Press F4 [OK] key.
A zone will be added to the gripper.

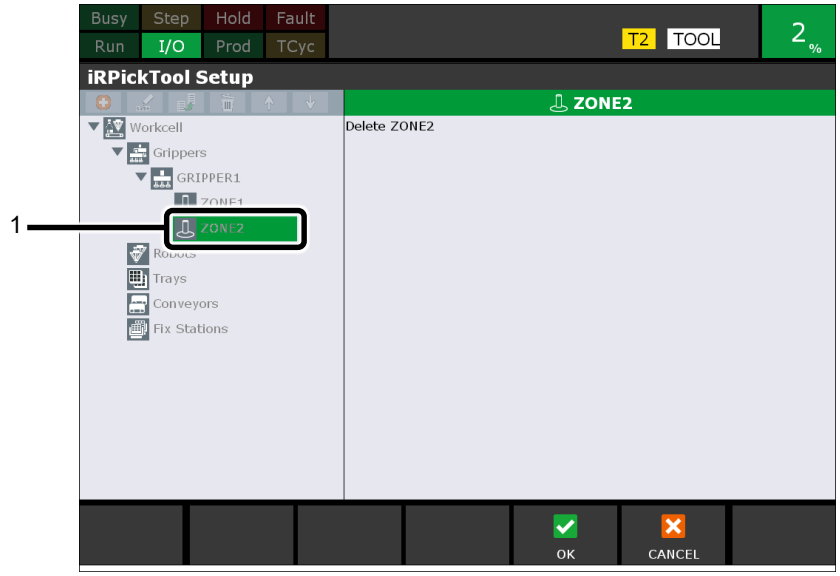


4 When there is more than one zone for this gripper, repeat the above procedure for each zone.

Deleting a gripper Zone

A zone can be deleted from the Gripper. However, you can only delete the last zone of a gripper. You cannot delete the first zone since a gripper must always contain at least one zone. If you wish to delete the first zone, then delete the Gripper itself.

1 In the tree view, select the [ZONE] (the last zone of a griller) to be deleted.
2 Press the button above the tree view, or select [DELETE] from [EDIT TREE].
3 A confirmation message like the one shown below will appear, so press F4 [OK].



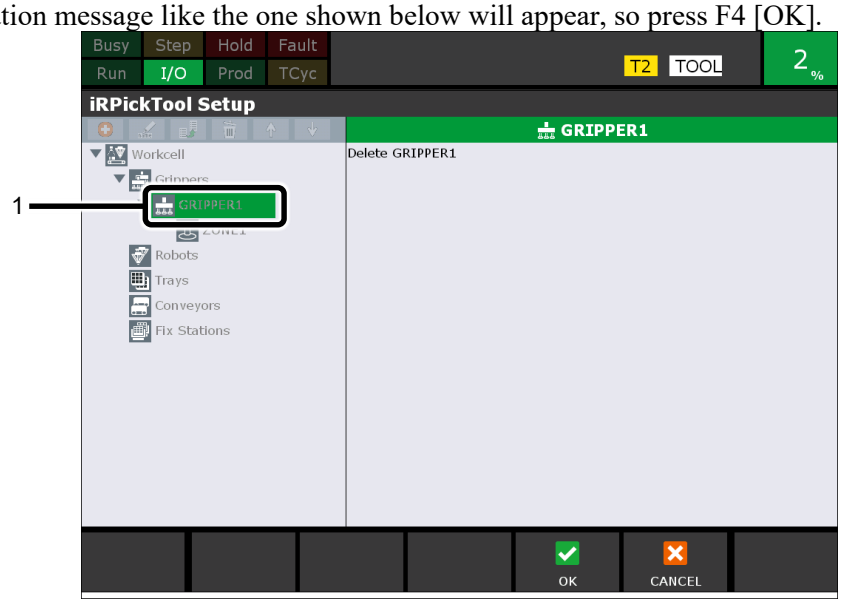
The zone will be deleted from the Gripper.

> **⚠ CAUTION**
> If you try to delete a zone that is not the last one in the list, you will get a popup message saying “Only the last zone of the multiple zones can be deleted.”

Deleting a Gripper

A gripper can be deleted from the Grippers collection.

1 In the tree view, select the [Gripper] that you want to delete.
2 Press the button above the tree view, or select [DELETE] from [EDIT TREE].
3 A confirmation message like the one shown below will appear, so press F4 [OK].



The gripper and all its zones will be deleted.

7.2.1.1 Setting up the gripper in detail

Set a Gripper in detail.
In the tree view on the left side, select the gripper.

[Pick Delay]

Pick delay is the time the robot should wait after a pick operation. Typically you use this delay to ensure that the gripper activation is complete and the part is firmly held by the gripper before the robot begins its next move. Enter the gripper delay in milliseconds.

[Place Delay]

Place delay is the time the robot should wait after a place or drop operation. Typically you use this delay to ensure that the gripper activation is complete and the part is properly released by the gripper before the robot begins its next move. Enter the gripper delay in milliseconds.

During production run, the *i*RPickTool TP programs call the PKGRGETDPDLY.pc macro to get the place delay into a Numeric Register.

[Part Presence Check Delay]

Part presence check delay is the time after the gripper activation I/O has been fired that the part presence check should be made. Typically, you use this delay to deal with the latency of the part presence sensor (for example, the vacuum made signal). If a part presence check is made prior to this delay, then it may fail.

During production run, the *i*RPickTool TP programs call the PKGRCHKPP.PC macro to do the part presence check. The PKGRCHKPP.PC macro uses the delay to wait before initiating the part presence check.

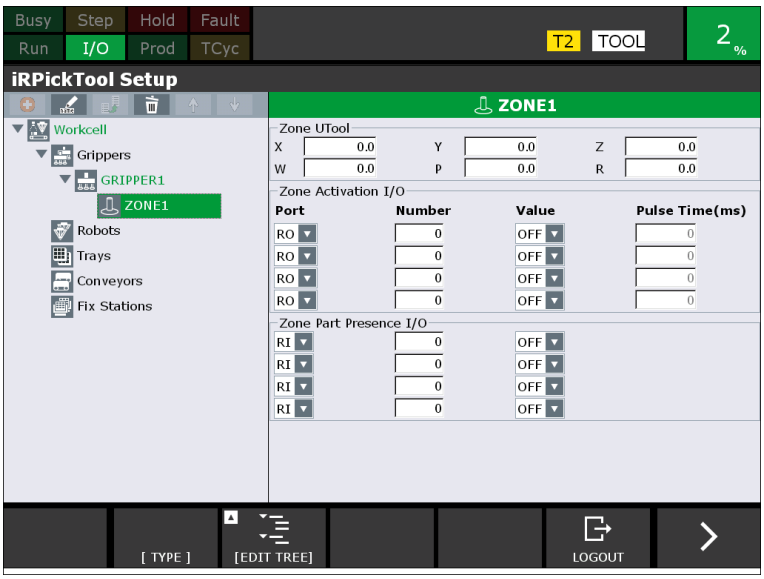
> **⚠ CAUTION**
> When using any of the gripper delay properties, you should be aware that it will reduce the cycle time of the application. You should also be aware that when these delays are used in tracking TP programs then the robot will continue to track the part downstream till the end of the delay. However, in majority of the applications some small delay is usually required to pick and place the parts accurately.

[Number of Zones]

The number of zones in the gripper is displayed here as a read-only property. You cannot change it. However, when you add or delete zones in a gripper, this number is updated automatically.

7.2.1.2 Setting up the zone in detail

Set a Gripper zone in detail.
In the tree view on the left side, select the zone under the Gripper.



[Zone UTool]

Each zone of the gripper has a UTOOL value.
You might not know the UTOOL of each zone. In order to determine the proper value of the Zone UTOOL, you need to teach the UTOOL with the robot from the MENU -> Setup -> Frames menu. After you complete teaching the UTOOL, come back to this menu and enter the values of the UTOOL for the zone.

*i*RPickTool provides three macros for utilizing the zone UTOOL data.

PKGRGETZNCNT gets the total zone count for a specified gripper in a Numeric Register.

PKGRGETZNUT gets the specified gripper's specified zone's UTOOL into a Position Register.

PK_GR_GET_UTOOLS1.TP which uses both PKGRGETZNCNT.PC and PKGRGETZNUT.PC gets the UTOOL specified for each zone into a Position Register and sets the UTOOL[n] = PR[x] instruction where n is the zone id and x is the Position Register number to which PKGRGETZNUT returned the UTOOL you specified in the User Interface above.

[Zone Activation I/O]

In each row, enter the I/O output for a zone. Up to 4 rows can be used to enter the I/O output for a zone, if more than one output is used for a zone. The value of the output can be ON, OFF or PULSE. If PULSE, is specified as the value of an output, then specify the duration of the pulse output in seconds.

The condition you specify in each row for the Zone Activation I/O is for the "Gripper Close" or "Pick part" operation. For example, if you specify RO[1] = ON, then the Robot Output RO[1] will be turned on when the PKGRCLOSE macro is executed during production run. RO[1] will be turned off when the PKGROPEN macro is executed during production run. If you specify RO[1] = OFF, then the RO[1] will be turned off during gripper pick operation and turned on during gripper place operation.

You can specify more than one output of either RO or DO type for a zone. You might want to do this when the gripper zone is meant for handling a larger part.

[Zone Part Presence I/O]

In each row, enter the input I/O for part presence for a zone. Up to 4 rows can be used to enter the I/O input for a zone, if more than one input is used for a zone. The value of the I/O input can be ON or OFF.

The condition you specify in each row for the Zone Part Presence I/O is for the "Gripper Close" or "Pick part" operation. For example, if you specify RI[1] = ON, then the Robot Input RI[1] will be checked for the ON condition when the PKGRCHKPP macro is executed during production run. If you specify RI[1] = OFF, then the RI[1] will be checked for the OFF condition during gripper pick operation and the ON condition during gripper place operation.

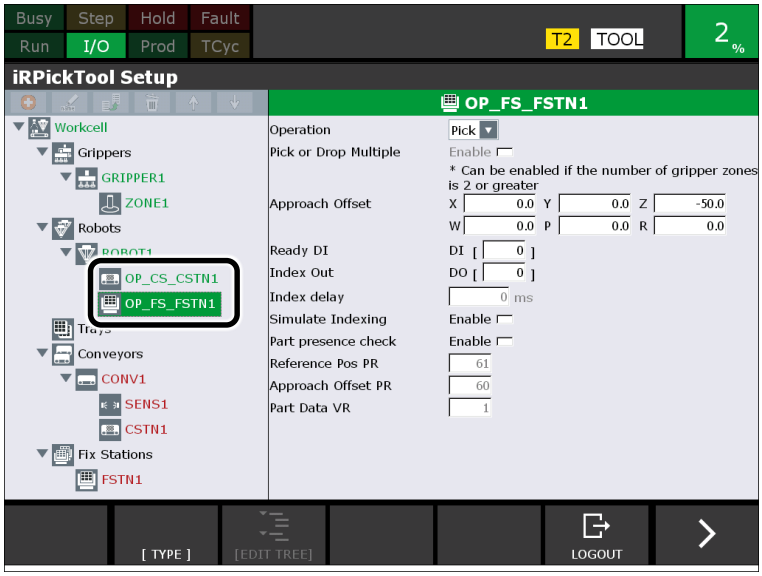
You can specify more than one input of either RI or DI type for a zone. You might want to do this when the gripper zone is meant for handling a larger part.

7.3 SETUP OF A ROBOT OPERATION

An operation means the picking/placing motions that the robot executes at each station.
For stations, there are [Conveyor Stations] and [Fixed Stations], and for each station, there are the corresponding operations under the robot in the tree view.
For operations, there are setting items such as the type of task performed at the station (picking/placing), whether to pick up parts one by one or pick them up together in cases where there are multiple zones, and whether to execute a part present check after the operation.
If you set these correctly, you will be able to complete the task you are aiming for without editing a Plug & Play standard TP program.

The Robot Operations Setup is done after you create a conveyor station under the Conveyors node or a Fixed Station under the Fixed Stations node. When you create a conveyor station or a fixed station, the robot operation node under the appropriate Robot in the tree is automatically created with default properties You cannot create it using the  icon in the treeview toolbar or from the [EDIT TREE] CREATE function key. Similarly, you cannot delete a robot operation node either. However, when you delete a Conveyor Station or a Fixed Station, the corresponding robot operation node is automatically deleted.

The robot operation nodes related to a conveyor station has a name in the following format: “OP_CS_xxxx” where OP_CS_ is the prefix and xxxx is the name of the conveyor station.
The robot operation nodes related to a Fixed station has a name in the following format: “OP_FS_xxxx” where OP_FS_ is the prefix and xxxx is the name of the fixed station.
You cannot change the name of a robot operation node. However, if you rename the Conveyor Station or the Fixed Station, the name of the Robot operation node is also renamed automatically for the xxxx part.



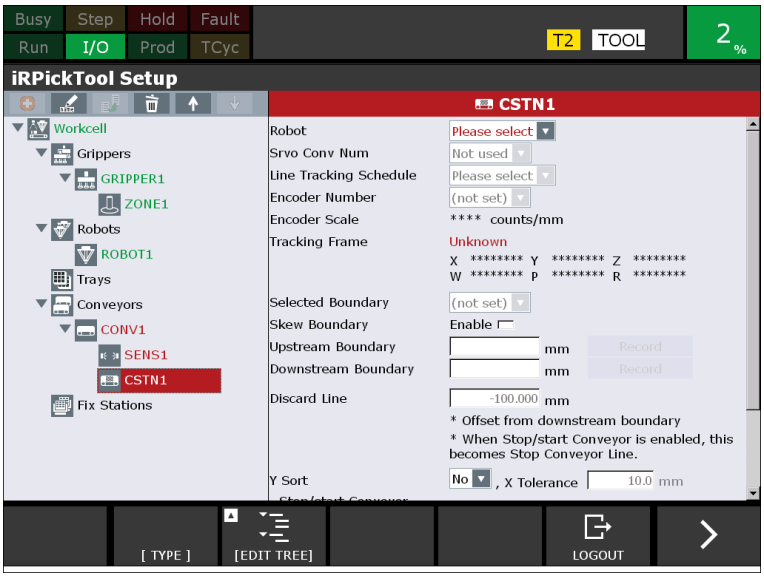
7.3.1 Conveyor Station Robot Operation

Set a Conveyor Station Robot Operation node.

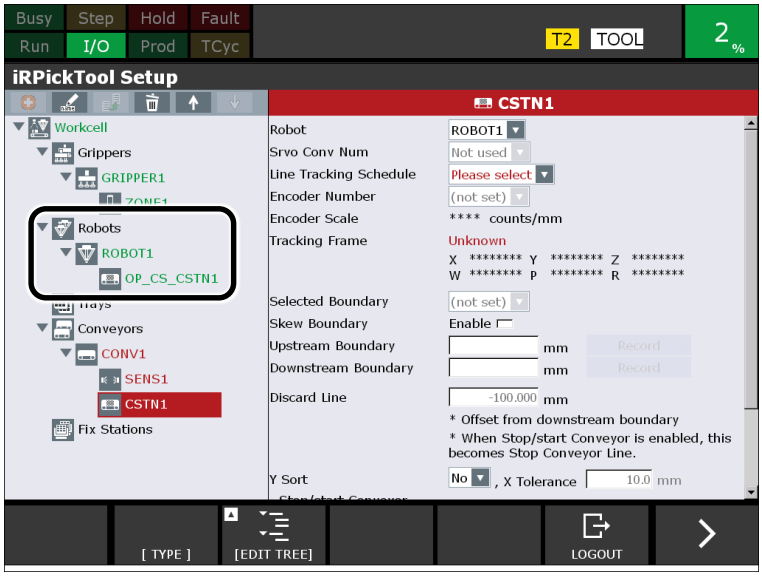
Creating a new Conveyor Station Robot Operation Node

Create a new Conveyor Station Robot Operation Node.

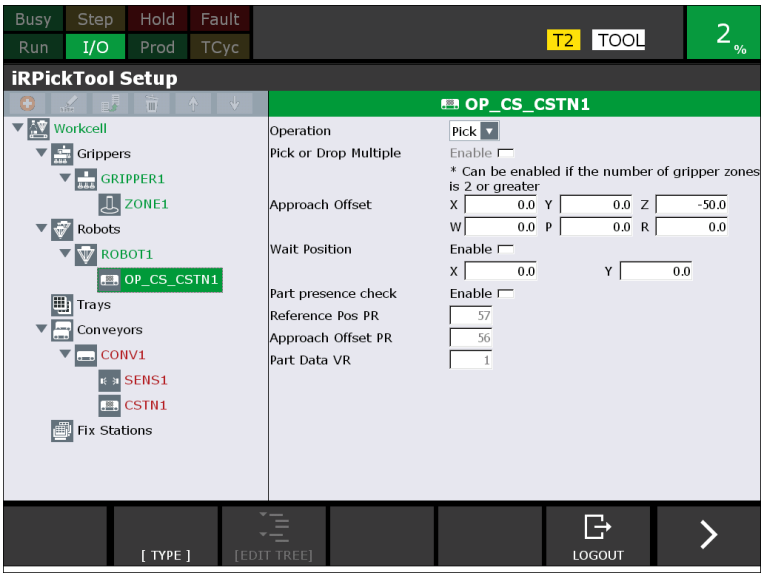
The screen below is the display screen for when creating a conveyor station.
Notice that there is no operation node under the Robot yet.



1 Select [Robots] from among the conveyor station settings.
Notice that as soon as you did that, the [OP_CS_CSTN1] node was immediately added under the Robot automatically.



2 Choose the operation [OP_CS_CSTN1] under the robot.
On the right hand side you will see all the properties as shown in the figure below.



7.3.1.1 Setting up conveyor station robot operation in detail

Set a conveyor station robot operation in detail.
In the tree view on the left side, select the conveyor station robot operation node under the Robot.

[Operation]

Select the “Pick” or the “Place” operation that the robot should perform at the conveyor station from the drop-down box.

[Pick or Drop Multiple]

This item can be set only when there are two or more [Zones] for the [Gripper].
If you enable [Pick or Drop Multiple], the robot will simultaneously pick/place the same number of parts as the number of [Zones].

> ⚠ **WARNING**
> If you set a tray incorrectly, serious effects such that the collision between a robot gripper and the tray can occur when a robot places multiple parts into the tray or picks up multiple parts from the tray simultaneously by using the hand that can hold multiple parts. Refer to “Setting of a tray when a robot hand that can hold multiple parts is used” in 6.3.1 “Cell Operation” for more information.

It will pick the first part from the zone 1 of the gripper and then go to the retreat position (or the Wait position if it is enabled and there is no part to pick). Then when the next part becomes available, it will go back to the pick position, pick the part and then return to the retreat position (or the Wait position if it is enabled and there is no part to pick). This continues until the last part has been picked.

If this item is not selected, then the robot will pick each part individually.

The KAREL macro PKCSGETPMUL.PC can be used to retrieve the value of the quantity into a Numeric Register.

[Approach Offset]

Enter the x, y, z, w, p, r offset of the approach position relative to the pick / place position. In most applications, you will only use the Z offset because the approach position is in line with the pick or place position.

The KAREL macro PKCSGETAPPR.PC can be used to assign the value of the approach offset into the dedicated Position Register "Approach offset PR".

> ⚠ **CAUTION**
> In the *i*RPickTool TP programs, the approach offset is actually used as a Tool_offset and not as an Offset instruction. A negative Z value indicates that the Tool coordinate Z direction is pointed out from the faceplate towards the Pick or the Place Position. If you have setup the Tool Coordinate Z direction to be in the opposite direction then you will need to use a positive value for approach offset Z.

[Wait Position]

You can enable or disable the wait position move using this checkbox. If the wait position is enabled, then the robot will move to the Wait Position of the conveyor station when there is no part to pick / place. The wait position is a position at the upstream boundary of the conveyor station at the height of approach offset.z value from the pick / place position. This position is calculated automatically by *i*RPickTool. If you move the upstream boundary, then the wait position also automatically moves with it.

You use the Wait position to improve the cycle time in your application. If you don't use the wait position, then the robot will wait at the retreat position after completing its operation. For example, if your pick and place stations are both conveyor stations, then after completing a pick, if the place target is not yet available, the robot will wait at the retreat position of the pick station. If wait position is enabled for the place conveyor station, then the robot will wait at the upstream boundary of the place station. If the distance between pick and place stations is large, then you will see significant cycle time improvement and fewer missed parts by using this feature.

[Wait Position X]

Enter a positive value in millimeters to shift the Wait Position downstream along conveyor flow. A negative value will move the Wait position upstream of conveyor flow. Internally in *i*RPickTool, the shift is along the track frame X of that Conveyor station.

[Wait Position Y]

Enter a value in millimeters to shift the Wait Position perpendicular to conveyor flow direction. A positive value will shift the wait position along the conveyor station's track frame Y direction. A negative value will shift it in the opposite direction.

[Part Presence Check]

Enable or disable the part presence check when the robot picks or places a part at this conveyor station. The part presence check is done by the KAREL macro PKGRCHKPP.PC.

> ⚠ **CAUTION**
> If you enable Part Presence Check but you have not setup the part presence I/O in the robot's gripper menu, then the part presence check will not be done.

[Reference Pos PR]

*i*RPickTool uses this dedicated Position Register as the Reference Position when you go through the REF_POS wizard in the Sensor menu. (For information about the Reference Position, see Section 4.2.12,

"Setting up a reference position for the outfeed conveyor, and teaching a robot position".) This is the pick / place position in the tracking frame. You cannot change this Position Register.

[Approach Offset PR]

*i*RPickTool uses this dedicated Position Register to store the approach offset relative to the Ref Pos in Tool coordinates. You cannot change this Position Register.

The KAREL macro PKCSGETAPPR.PC can be used to assign the value of the approach offset into this dedicated Position Register.

[Part Data VR]

*i*RPickTool uses this dedicated Vision Register to get the part from the part queue using the PKCSGETQUE macro. You cannot change this Vision Register.

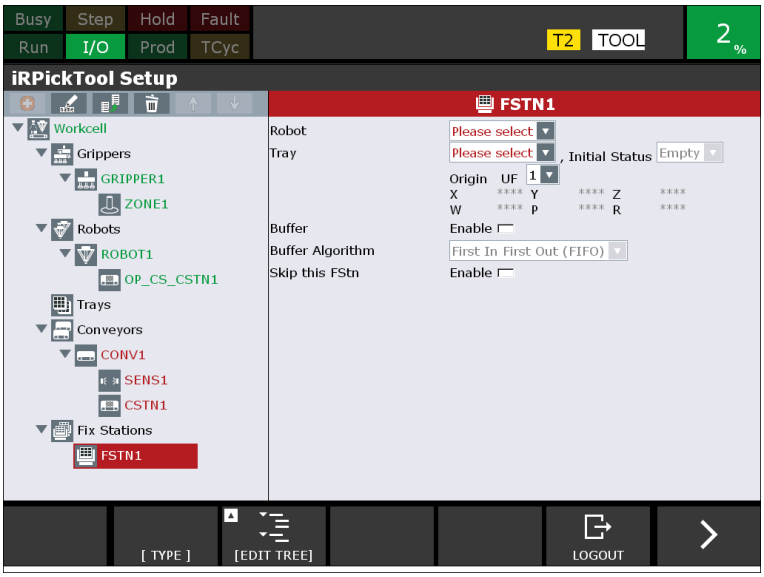
7.3.2 Fixed Station Robot Operation

Set a Fixed Station Robot Operation node.

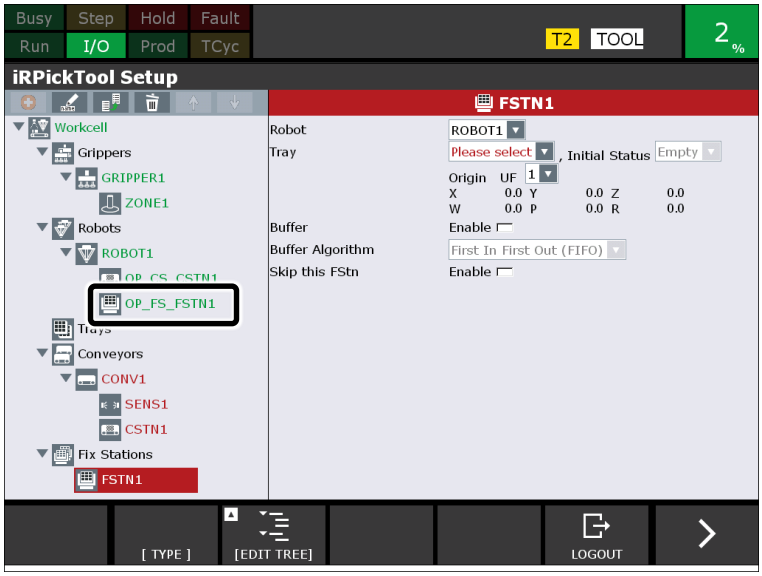
Creating a new Fixed Station Robot Operation Node

Create a new Fixed Station Robot Operation Node.

The screen below is the display screen for when creating a fixed station.
Notice that there is no operation node under the Robot yet.



1 Select [Robots] from the among the fixed station settings.
Notice that as soon as you did that, the [OP_FS_FSTN1] node was immediately added under the Robot automatically.



2 Choose the operation [OP_CS_FSTN1] under the robot.
On the right hand side you will see all the properties as shown in the figure below.
The buffer frame will be displayed only if the R857: *i*RPickTool / Option Package Add-on is installed (for North America, R827: *i*RPickTool / Production Control Add-on). The properties in the buffer frame and the operation of the buffer are described in the section on 7.12 “BUFFER FUNCTION”.

7.3.2.1 Setting up fixed station robot operation in detail

Set a fixed station robot operation in detail.
In the tree view on the left side, select the fixed station robot operation node under the Robot.

[Operation]

Select the “Pick” or the “Place” operation that the robot should perform at the fixed station from the drop-down box.

[Pick or Drop Multiple]

This item can be set only when there are two or more [Zones] for the [Gripper].
If you enable [Pick or Drop Multiple], the robot will simultaneously pick/place the same number of parts as the number of [Zones].

The first part is picked from the zone 1 of the gripper and each next part is picked from the next consecutive zone in one shot.

If this item is not selected, then the robot will pick each part individually. It will pick the first part from the zone 1 of the gripper and then go to the retreat position. Then when the next part becomes available, it will go back to the pick position, pick the part and then return to the retreat position. This continues until the last part has been picked.

If you select this item, then the robot will pick all the parts simultaneously.

The KAREL macro PKFSGETPMUL.PC can be used to retrieve the value of the pick or drop multiple property into a Numeric Register.

[Approach Offset]

Enter the x, y, z, w, p, r offset of the approach position relative to the pick / place position. In most applications, you will only use the Z offset because the approach position is in line with the pick or place position.

The KAREL macro PKFSGETAPPR.PC can be used to assign the value of the approach offset into the dedicated Position Register “Approach offset PR”.

> ⚠ **CAUTION**
> In the *i*RPickTool TP programs, the approach offset is actually used as a Tool_offset and not as an Offset instruction. A negative Z value indicates that the Tool coordinate Z direction is pointed out from the faceplate towards the Pick or the Place Position. If you have setup the Tool Coordinate Z direction to be in the opposite direction then you will need to use a positive value for approach offset Z.

[Ready DI]

An *i*RPickTool Fixed Station has a Digital Input (DI) to indicate the readiness for the pick or the place operation.

The KAREL macro PKFSGETRDYDI.PC can be used to retrieve the value of the Ready DI port number into a Numeric Register.

[Index Out DO]

Enter the Digital Output port number to turn on to index the station. During production run, the TP program PK_FS_INDEX handles the Indexing of the Fixed Station using this output when all the parts in its tray have been picked from or placed into.

Once this output goes high, the station should index (*i*RPickTool should see the Ready DI go low) and the new empty station should come in (*i*RPickTool should see the Ready DI go high) before the robot will start picking or placing parts at this Fixed Station.

The KAREL macro PKFSGETIXODO.PC can be used to retrieve the value of the Index Out DO into a Numeric Register.

[Simulate Indexing]

If you enable Indexing simulation, then after entering valid I/O values for the Station Ready and Index Out Signals, *i*RPickTool will automatically simulate the indexing using an Index Delay specified in the property "Index Delay".

With this feature, *i*RPickTool will turn on the Index Out Signal, simulate the Ready DI port to go low and then come back on after an Indexing Delay. The robot is then able to continue servicing this fixed station.

This feature is useful for continuous running of your workcell in a test-run mode.

The KAREL macro PKFSGETIXSIM.PC can be used to simulate the Station Ready DI's initial state.

[Index Delay]

Specify the Index delay for simulation of indexing in milliseconds.
The value is used only if simulation of indexing is enabled.

[Part Presence Check]

Enable or disable the part presence check when the robot picks or places a part at this fixed station.

> ⚠ **CAUTION**
> If you enable Part Presence Check but you have not setup the part presence I/O in the robot's gripper menu, then the part presence check will not be done.

[Reference Pos PR]

*i*RPickTool uses this dedicated Position Register as the Reference Position for the pick or place position of this fixed station. You cannot change this Position Register.

[Approach Offset PR]

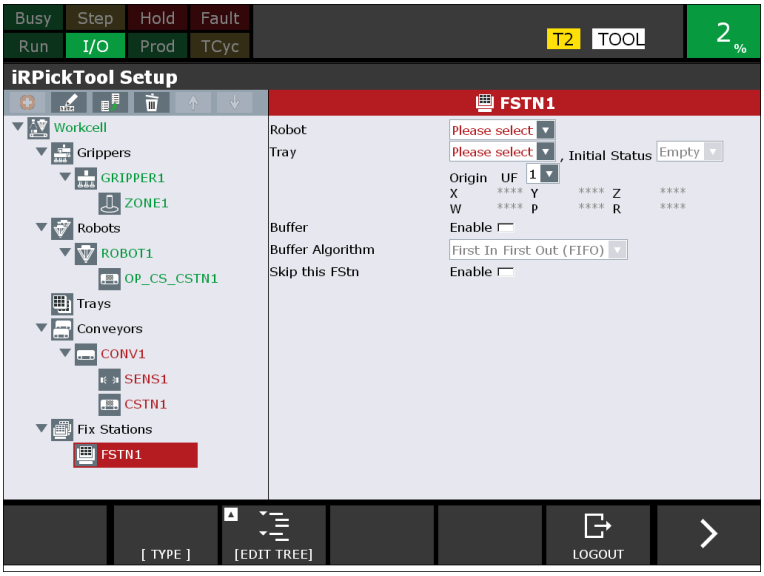
*i*RPickTool uses this dedicated Position Register to store the approach offset relative to the Ref Pos in Tool coordinates. You cannot change this Position Register.
The KAREL macro PKFSGETAPPR.PC can be used to assign the value of the approach offset into this dedicated Position Register.

[Part Data VR]

*i*RPickTool uses this dedicated Vision Register to get the part from the Fixed Station Tray queue using the PKFSGETQUE macro. You cannot change this Vision Register.

7.4 SETUP OF A FIXED STATION

Set up a fixed station.
This section explains the settings for [Skip], which can be set only with Plug & Play. For other settings, refer to Chapter 6, “BASIC FUNCTIONS REFERENCE”.



[Skip this FStn]

If you enable this function, the robot does not work in this fixed station. Enable it when you set the same robot to several fixed station and intend to disable this station.

7.5 ROBOT PROGRAMS

A set of standard TP programs is provided for Plug & Play. If you execute the main program, PK_MAIN1.TP, the necessary robot operations are performed based on the data you set in Section 7.1, "SETUP OF A WORKCELL" to Section 7.4, "SETUP OF A FIXED STATION".

Also, with regard to the task of actually teaching the robot position, carry it out by referring to Subsection 4.2.12, "Setting Up a Reference Position and Teaching a Robot Position" for conveyor stations, and to Section 6.10, "REFERENCE POSITION SETTING" and Section 6.11, "TEACHING THE POSITION TO ROBOTS" for fixed stations.

> ⚠ **CAUTION**
> *i*RPickTool does not support power failure handling. Power failure handling is disabled by default. After rebooting, be sure to execute the main program PK_MAIN1.TP from the top.

Your system requirements may make it necessary to modify the robot programs. Subsection 7.5.1, "Robot Behavior Under the Standard TP Programs" describes how the robot behaves with the standard TP programs.

The standard TP programs for Plug & Play exclusively use the registers, Position Registers, character registers, Vision Registers, load settings, user alarms, user frames, and user tool frames described in Subsection 7.5.2, "Numeric Registers" and later. Do not use these registers and settings.

> ⚠ **WARNING**
> If any of the registers, Position Registers, character registers, Vision Registers, load settings, user alarms, user frames, and user tool frames exclusively intended for Plug & Play is used for any other purpose, the robot may show unexpected behavior or some other serious effect may be caused on the system.

7.5.1 Robot Behavior Under the Standard TP Programs

The standard TP programs assume that the robot behaves according to certain rules. The assumed rules are described below.

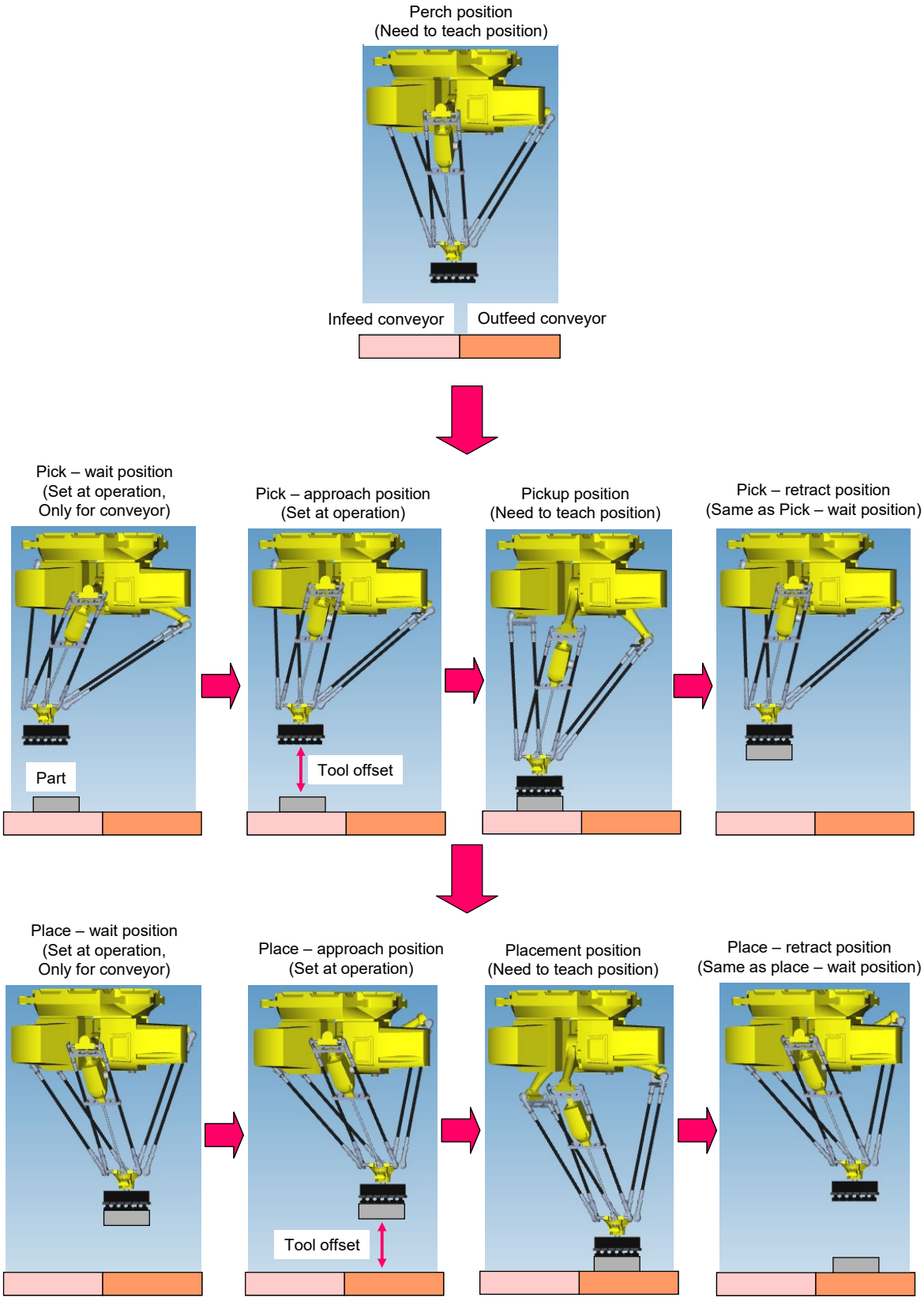


Fig. 7.5.1(a) Robot moving in accordance with a fixed rule

Once the placement motion is complete, the robot performs the pickup motion again, thus repeating the pickup and placement motions. The robot moves to the leftmost waiting position during the pickup or placement motion only when it picks a part from a conveyor or places a part onto a conveyor. In the case of a fixed station, the robot does not move this way.
Regarding the behavior of the robot, the following three rules are particularly crucial:

1 The approach position is at the same location as the retract position.
2 The value to be set to teach the approach position is the tool offset value that is set with respect to the pickup or placement position.
3 The points that need to be taught during the entire flow are the origin, pickup position, and placement position only.

If the robot behaves in a way that violates any of these three assumed rules, modify the KAREL programs and standard TP programs as necessary, after carefully reading Section 6.8, "KAREL PROGRAMS", 7.9, "KAREL PROGRAMS", and Section 7.10, "STANDARD TP PROGRAMS".

7.5.2 Numeric Registers

*i*RPickTool uses the following Numeric Registers in the TP programs.
Presently, Registers 95-230 are used.
Do not use these registers for other purposes. If you need more registers, please increase the registers using the Program Setup function at Controlled Start.

Table 7.5.2(a) Registers

R[95]~[98], R[100]	Temporary use registers. They are re-used in the various TP programs.
R[99]	Register for storing Active Recipe.
R[97]~[100]	Registers used commonly in all groups. These are used in programs for Group 1 and Group 2
R[101]~[108]	Registers global to group 1 only.
R[109]~[125]	Registers for the first pick operation (regardless of conveyor or fixed station picks)
R[126]~[127]	Registers for Group 1 Buffer operations
R[129]~[145]	Registers for the first drop operation (regardless of conveyor or fixed station drops)
R[147]	Register for Group 1 Buffer operations
R[148]~[150]	Temporary use registers. They are re-used in the various TP programs.
R[151]~[158]	Registers global to group 2 only.
R[159]~[175]	Registers for the second pick operation (regardless of conveyor or fixed station picks)
R[176]~[177]	Registers for Group 2 Buffer operations
R[179]~[195]	Registers for the first drop operation (regardless of conveyor or fixed station drops)
R[197]	Register for Group 2 Buffer operations
R[199]~[205]	Registers for Group 1 Buffer operations
R[209]~[215]	Registers for Group 2 Buffer operations
R[219]~[229]	Reserved
R[230]	Register for pre-grouping (This is used even when pre-grouping is not used. It is used for judging whether pre-grouping is used of not.)

7.5.3 Position Registers

*i*RPickTool uses the following Position Registers in the TP programs.
Presently, Position registers 51-90 are used.
Do not use these position registers for other purposes. If you need more Position registers, please increase the Position Registers using the Program Setup function at Controlled Start.

Table 7.5.3(a) Position registers

PR[51]	Temporary Use Position Register.
PR[52]	Buffer approach offset for all groups
PR[53]	Buffer Reference Position for all groups
PR[54]	Perch Position for all groups
PR[55]	Wait Position for Group 1 robot
PR[56]	First pick Conveyor Station Approach Offset for all groups
PR[57]	First pick Conveyor Station Reference Position for all groups
PR[58]	First drop Conveyor Station Approach Offset for all groups
PR[59]	First drop Conveyor Station Reference Position for all groups
PR[60]	First pick Fixed Station Approach Offset for all groups
PR[61]	First pick Fixed Station Reference Position for all groups
PR[62]	First drop Fixed Station Approach Offset for all groups
PR[63]	First drop Fixed Station Reference Position for all groups
PR[64]	Reserved
PR[65]	Second pick Conveyor Station Approach Offset for all groups
PR[66]	Second pick Conveyor Station Reference Position for all groups
PR[67]	Second drop Conveyor Station Approach Offset for all groups
PR[68]	Second drop Conveyor Station Reference Position for all groups
PR[69]	Second pick Fixed Station Approach Offset for all groups
PR[70]	Second pick Fixed Station Reference Position for all groups
PR[71]	Second drop Fixed Station Approach Offset for all groups
PR[72]	Second drop Fixed Station Reference Position for all groups
PR[73]	Reserved
PR[74]	Third pick Conveyor Station Approach Offset for all groups
PR[75]	Third pick Conveyor Station Reference Position for all groups
PR[76]	Third drop Conveyor Station Approach Offset for all groups
PR[77]	Third drop Conveyor Station Reference Position for all groups
PR[78]	Third pick Fixed Station Approach Offset for all groups
PR[79]	Third pick Fixed Station Reference Position for all groups
PR[80]	Third drop Fixed Station Approach Offset for all groups
PR[81]	Third drop Fixed Station Reference Position for all groups
PR[82]	Reserved
PR[83]	Fourth pick Conveyor Station Approach Offset for all groups
PR[84]	Fourth pick Conveyor Station Reference Position for all groups
PR[85]	Fourth drop Conveyor Station Approach Offset for all groups
PR[86]	Fourth drop Conveyor Station Reference Position for all groups
PR[87]	Fourth pick Fixed Station Approach Offset for all groups
PR[88]	Fourth pick Fixed Station Reference Position for all groups
PR[89]	Fourth drop Fixed Station Approach Offset for all groups
PR[90]	Fourth drop Fixed Station Reference Position for all groups

7.5.4 String Registers

*i*RPickTool uses the following Position Registers in the TP programs. Do not use these string registers for other purposes.

Table 7.5.4(a) String registers

SR[1]	First pick station name (conveyor or fixed station)
SR[2]	First drop station name (conveyor or fixed station)
SR[3]	Temporary use SR (reserved)
SR[4]	First pick conveyor name
SR[5]	First pick conveyor name

7.5.5 Vision Registers

*i*RPickTool uses the following Vision Registers in the TP programs. Do not use these vision registers for other purposes.

Table 7.5.5(a) Vision registers

VR[1]	Group 1 pick part data (conveyor or fixed station)
VR[2]	Group 1 drop part data (conveyor or fixed station)
VR[3]	Group 2 pick part data (conveyor or fixed station)
VR[4]	Group 2 drop part data (conveyor or fixed station)

Above Vision Registers are used regardless whether the station is conveyor station or fixed station.

7.5.6 Payload IDs

*i*RPickTool uses the following payload IDs in the TP programs. Set the empty gripper payload and full gripper payload. Do not use these payload IDs for other purposes.

Table 7.5.6(a) Payload IDs

PAYLOAD[1]	Empty Gripper
PAYLOAD[2]	Full Gripper

7.5.7 User Alarms

*i*RPickTool uses the following user alarms in the TP programs Do not use these user alarms for other purposes.

Alarm No. User Message
[1]: [Invalid Robot Id]
[2]: [Invalid Gripper Id]
[3]: [No pick station]
[4]: [No drop station]
[5]: [Pick or Drop Qty > Zones]
[6]: [Cannot use PRGR and Buffers]

7.5.8 User Frames

The TP programs use the user frames that you reference in your Fixed Station property page in the *i*RPickTool treeview. It does not use any dedicated user frames for internal use.

7.5.9 Tool Frames

The number of Tool Frames used by *i*RPickTool depends on the number of zones in the gripper used by the robot (belonging to a motion group).

For example, if you have only one zone in your gripper, only Tool Frame #1 will be used.
If you had 10 zones in the gripper, then *i*RPickTool TP programs will use Tool Frames 1 through 10.

If you have 20 zones (maximum number of zones) in the gripper, then *i*RPickTool TP programs will use Tool Frames 1-20. However, you will have to additional setup to increase the number of tool frames at Control Start when the number of zones exceed 10 because by default you only get 10 Tool Frames in the standard product.

7.6 REFERENCE POSITION SETTING

The reference position is the part detection position to be used when the pickup position or placement position is taught to the robot program.
Teach pick-up/placement positions to all the robots that work on the same conveyor, with regard to a part for which a reference position has been specified.

With regard to the setting of the reference position for Plug & Play, refer to Subsection 4.2.12, “Setting Up a Reference Position and Teaching a Robot Position” for conveyor stations, and to Section 6.10, “REFERENCE POSITION SETTING” for fixed stations.

7.7 TEACHING THE POSITION TO ROBOTS

After setting the reference position, teach the position to robots.
With regard to robot position teaching, refer to Section 4.2.12, “Setting Up a Reference Position and Teaching a Robot Position” for conveyor stations, and to Section 6.11 “TEACHING THE POSITION TO ROBOTS” for fixed stations.

7.8 FINE ADJUSTMENT OF THE TRACKING MOTION

The procedure to make fine adjustments to the robot's tracking motions with Plug & Play is the same as the procedure for the basic functions.
Refer to Section 6.12, “FINE ADJUSTMENT OF THE TRACKING MOTION”.
This section describes in particular the procedure to make error adjustments in a Plug & Play standard program.

7.8.1 Adjustment of the Tracking Frame Error

When a camera is used to detect parts, the tracking motion is subject to error if the tracking frame recognized by the camera does not match that recognized by each individual robot. While this error is corrected properly if the part is not rotating, displacement is generated if the part is rotating.

This error occurs because the tracking frame is not set up correctly. To fix the error, it is necessary in principle to set up the tracking frame again beginning with the setting of the TCP. You can improve the situation, however, by adding an amount of adjustment to the vision offset calculated by *i*RVision as instructed below. Make this adjustment for each robot individually.

In the sample program shown below, Position Register 20 is used to store the amount of adjustment. Change it as necessary.

1 Add the line indicated below just before moving to approach position.

Example: Standard TP program in Subsection 7.10.2.12 "PK_CV_PICK11.TP"

```
56:  CALL ADJ_OFS(1,1,20,1)
57:  -- Move to approach
58:L PR[57:Pk1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[1]
       Tool_Offset,PR[56:Pk1 Cv Ap Ofst]
```

2 Insert a motion instruction and a wait instruction immediately after the approach point so as to make the robot stop immediately above the pickup position set in the tracking program.

```
69:  LBL[88]
70:  -- Move to pick pos
71:L PR[57:Pk1 Cv Ref Pos] max_speed CNT0 VOFFSET,VR[1] Tool_Offset,PR[15]
72:  WAIT 10.00(sec)
73:L PR[57:Pk1 Cv Ref Pos] max_speed CNT0 VOFFSET,VR[1] TB R[125]sec,
       CALL PKGRCLOSE(Gripper ID=R[104:Gripper Id G1],
       Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone G1])
74:  -- Chk Skip Pick
```

The wait instruction is inserted to check the displacement visually. Set the wait time to 10 seconds or so. Also, set a small value in Z of PR[15] (tool offset to create an approach point for the pickup motion set 0 for the values other than Z). This is necessary to make the robot stop immediately above the pickup position.

3 In PR[20], set X, Y, Z, W, P and R to 0, 0, 0, 0, 0, and 0.
4 Move the conveyor at the operating speed.
5 Start the main program (PK_MAIN1) as usual, and convey a part on the conveyor. Make sure that the part faces the same direction as when the reference position was set.

> ⚠ **CAUTION**
> Select T2 mode or Auto mode to run the tracking program. If you select T1 mode, you will see the alarm, "LNTK-041 Encoder is moved in T1 mode".

6 When the robot reaches the approach point, stop the conveyor. Leave the program running.
7 When the conveyor stops completely, stop the program temporarily.
8 Check the amount of displacement between the part position and the robot position.
9 If there is any displacement, jog the robot to the correct position. When asked about whether to subtract the offset value, select [YES].
10 Start the main program again, and convey a part on the conveyor.
This time around, place the part so that it faces 180 degrees opposite the direction used when the reference position was set.
11 When the robot reaches the approach point, stop the conveyor. Leave the program running.
12 When the conveyor stops completely, stop the program temporarily.
13 Check the amount of displacement between the robot position and the part position.
14 If there is any displacement, measure the displacement in the X direction (parallel to the conveyor moving direction) and in the Y direction (orthogonal to the conveyor moving direction) individually. Set the amount of adjustment for the X direction in X of PR[20] and the amount of adjustment for the Y direction in Y of PR[20]. As the amount of adjustment, set a value equal to half of each amount of displacement. Execute the program from the line of ADJ_OFS, and check whether the robot moves to the correct position. If there is any displacement, change the amount of adjustment and

execute this procedure again from step 4 "Move the conveyor at the operating speed". to see whether the robot moves to the correct position.

15 After confirming that the robot moves to the correct position, delete the inserted motion instruction and wait instruction from the tracking program.

About ADJ_OFS

ADJ_OFS is a KAREL program that adds the specified amount of adjustment to the vision offset stored in a Vision Register. The meanings of its arguments are described below. For details, refer to the section, "VISION SUPPORT TOOLS", in the chapter, "OTHER OPTIONAL FEATURES" in the "*i*RVision OPERATOR'S MANUAL (Reference)".

Argument 1: Type of the register storing the vision offset. If the vision offset is stored in a Vision Register, specify 1.
Argument 2: If the vision offset is stored in a Vision Register, specify the number of that register.
Argument 3: Number of the Position Register storing the amount of adjustment
Argument 4: Number of the Vision Register storing the adjusted vision offset

NOTE
With *i*RPickTool, ADJ_OFS is available as a standard feature even if the vision support tool option is not ordered.

7.8.2 Adjustment of the Dynamic Error of the Robot

While the robot correctly reaches above the pickup position of a part when the conveyor is stopped, a certain amount of displacement occurs when the conveyor is moving. This error is due to a tracking delay in the robot motion.

NOTE
1 Before adjusting the dynamic error, make sure that no error occurs when the conveyor is stopped by making the error adjustments described in Subsections 7.8.1.
2 If the robot works on the plural conveyors, use the fastest conveyor to adjust the dynamic error.

To adjust this displacement, perform the procedure described below.

1 Insert a motion instruction and a wait instruction immediately after the approach point so as to make the robot stop immediately above the pickup position set in the tracking program.

Example: Standard TP program in Subsection 7.10.2.12 "PK_CV_PICK11.TP"

```
70:   LBL[88]
71:   -- Move to pick pos
72:L PR[57:Pk1 Cv Ref Pos] max_speed CNT0 VOFFSET,VR[1] Tool_Offset,PR[15]
73:   WAIT 10.00(sec)
74:L PR[57:Pk1 Cv Ref Pos] max_speed CNT0 VOFFSET,VR[1] TB R[125]sec,
       CALL PKGRCLOSE(Gripper ID=R[104:Gripper Id G1],
       Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone G1])
75:   -- Chk Skip Pick
```

The wait instruction is inserted to check the displacement visually. Set the wait time to 10 seconds or so. Also, set a small value in Z of PR[15] (tool offset to create an approach point for the pickup motion set 0 for the values other than Z). This is necessary to make the robot stop immediately above the pickup position.

2 Move the conveyor at the operating speed.
3 Start the main program (PK_MAIN1) as usual, and convey a part on the conveyor. Make sure that the part faces the same direction as when the reference position was set.

> ⚠ **CAUTION**
> Select T2 mode or Auto mode to run the tracking program. If you select T1 mode, you will see the alarm, "LNTK-041 Encoder is moved in T1 mode".

4 Check that the robot position correctly follows the pickup position of the part when the conveyor is moving and the program is running.
5 If there is any displacement, enter the value calculated by the following equation in [Dynamic Error Adjustment] mentioned in Subsection 6.12.3, "Adjustment of the Dynamic Error of the Robot": Displacement amount (mm) ÷ conveyor speed (mm/s) x 1000 (unit: ms). The conveyor speed is that observed when the robot follows the part. The displacement amount has a positive sign when the robot is displaced from the part in the upstream direction of the conveyor and a negative sign when displaced in the downstream direction. If a value is already set in [Dynamic Error Adjustment], add the value calculated above to the existing value.
6 Delete the inserted motion instruction and wait instruction from the tracking program.

Calculation example

When 30 is set in [Dynamic Error Adjustment], the conveyor speed is 500 mm/s, and the pickup position of the part is displaced by 3 mm in the upstream direction of the conveyor:
[Dynamic Error Adjustment] = +30 (ms) + (3 (mm) ÷ 500 (mm/s)) x 1000 = +36 (ms)

7.9 KAREL PROGRAMS

Since Plug & Play uses the standard programs in principle, you do not normally need to care about the KAREL programs used in these standard programs. Depending on circumstances, however, it may become necessary to modify the standard programs. When making any such modification, make sure you understand the specifications of the KAREL programs described below.

Every KAREL program used for *i*RPickTool follows the rules mentioned below.

- The name begins with two letters, [PK].
- The following two letters represent the type of object related to the program.
- The fifth and subsequent letters indicate the function of the program.

7.9.1 Workcell-Related Programs

The two letters following [PK] are [WC] namely, the names of these programs basically begin with these four letters, [PKWC] though there are some exceptions.

Furthermore, with regard to workcell-related programs, KAREL programs' basic functions are the same. Refer to Section 6.8, "KAREL PROGRAMS".

7.9.2 Gripper-Related Programs

The two letters following [PK] are [GR] namely, the names of these programs begin with these four letters, [PKGR].

Once you setup the Gripper and zone data, when you run production, the plug and play TP programs will use the gripper macros to handle the picking and placing of the parts without any coding on your part. A brief description of each gripper macro is provided below.

7.9.2.1 PKGRCHKPP.PC

This program checks if the required parts are present in the gripper after a “gripper close” or pick operation, or if the required parts are absent after a “gripper open” or place operation.
It uses the data you setup in the Gripper and Zone menu in the treeview.

The part presence check is performed for the specified zones of the gripper.
The part presence check is initiated immediately if the “Part presence check delay” is 0 milliseconds in the Gripper menu in the treeview. If the value of part presence check delay is greater than zero, then the check is initiated after that delay. This is required for some grippers such as vacuum grippers where there may be some delay after activating the I/O before “vacuum is made”.
If you use the part presence check delay when the robot is running this check in a tracking TP program, then you may see the robot tracking the part downstream during the part presence delay period.

Argument 1:
Specify an ID of a gripper.

Argument 2:
Specify the operation type, PICK or PLACE.

NOTE
When the argument guidance is supported, you select one of the following options as the operation type.
1 Pick
2 Place
However, the operation type is internally set as Pick=0 and Place=1. So if the argument guidance is not supported in your software version, specify 0 as Pick, or 1 as Place.

Argument 3:
Specify the Numeric Register in which to return the status. If the status is non-zero then the part presence check failed. That is, if the operation is pick, then required parts are not present. If the operation is place, then the required parts are not absent. You should always check the status when you use this macro.

Argument 4:
Specify the first zone of the gripper to start the part presence check.

Argument 5:
Specify the last zone of the gripper for the part presence check.

Example 1:
To check if the parts are present after a pick operation in the first two zones of a gripper whose ID is 1, use the following instruction. The status is returned in Numeric register 7. If the status is non-zero, then not all required parts were picked.

```
CALL PKGRCHKPP(Gripper ID=1,Pick,Stat Reg=7,Start Zone=1,End Zone=2)
```

7.9.2.2 PKGRCLOSE.PC

This program performs the “gripper close” or pick operation for the specified zones of a gripper. It uses the data setup in the Gripper and zone menus in the treeview to turn on the activation I/O.

Argument 1:
Specify an ID of a gripper.

Argument 2:
Specify the first zone of the gripper to close.

Argument 3:
Specify the last zone of the gripper to close.

Example:
To close zones 1-2 of a gripper whose ID is 1, use the following instruction.

```
CALL PKGRCLOSE(Gripper ID=1,Start Zone=1,End Zone=2)
```

7.9.2.3 PKGRGETDPDLY.PC

This program is used to get the drop or the place delay in seconds for a gripper that you specified as “Place Delay” in the Gripper setup menu in the treeview.

Argument 1:
Specify an ID of a gripper.

Argument 2:
Specify the Numeric Register in which to return the delay in seconds.

Example:
To get the place delay in seconds for a gripper whose ID is 1 in Numeric Register 7, use the following instruction.

```
CALL PKGRGETDPDLY(Gripper ID=1,DropDelay Reg=7)
```

7.9.2.4 PKGRGETID.PC

This program gets the ID of the gripper used by a specified robot ID.

Argument 1:
Specify the ID of the robot whose gripper ID is needed.

Argument 2:
Specify the Numeric Register in which the ID of the gripper should be returned.

Example:
To get the ID of a gripper used by a robot whose ID is 1 into Numeric Register 7, use the following instruction.

```
CALL PKGRGETID(Robot ID=1,Gripper ID Reg=7)
```

7.9.2.5 PKGRGETPKDLY.PC

This program is used to get the pick delay in seconds for a gripper that you specified as “Pick Delay” in the Gripper setup menu in the treeview.

Argument 1:
Specify an ID of a gripper.

Argument 2:
Specify the Numeric Register in which to return the delay in seconds.

Example:
To get the pick delay in seconds for a gripper whose ID is 1 in Numeric Register 7, use the following instruction.

```
CALL PKGRGETPKDLY(Gripper ID=1,PickDelay Reg=7)
```


7.9.2.6 PKGRGETZNCNT.PC

This program is used to get the number of zones in a gripper.

Argument 1:
Specify an ID of a gripper.

Argument 2:
Specify the Numeric Register in which to return the zone count.

Example:
To get the total number of zones for a gripper whose ID is 1 in Numeric Register 7, use the following instruction.

```
CALL PKGRGETZNCNT(Gripper ID=1,ZoneCount Reg=7)
```

7.9.2.7 PKGRGETZNUT.PC

This program is used to get the UTOOL of a Zone of a Gripper specified in the *i*RPickTool treeview into a Position Register.

Argument 1:
Specify the ID of a robot whose gripper’s zone UTOOL you want to get.

Argument 2:
Specify the ID of a gripper whose zone’s UTOOL you want to get.

Argument 3:
Specify the Zone ID of the gripper whose UTOOL you want to get.

Argument 4:
Specify the Position register number in which to return the zone UTOOL in XYZWPR format.

Example:

To get the UTOOL of the Zone 3 of a gripper whose ID is 2 and is used by a robot whose ID is 1 into Position Register 51, use the following instruction.

```
CALL PKGRGETZNUT(Robot ID=1,Gripper ID=2,Zone Index=3,ZoneUtool PR=51)
```

7.9.2.8 PKGRINIT.PC

This program resets the gripper and zone data for all the grippers to factory defaults. You use this only in case you want to wipe out all the gripper data and start from scratch.

There is no argument.

Example:

```
CALL PKGRINIT
```

7.9.2.9 PKGROPEN.PC

This program performs the "gripper open" or place operation for the specified zones of a gripper. It uses the data setup in the Gripper and zone menus in the treeview to turn off the activation I/O.

Argument 1:

Specify an ID of a gripper.

Argument 2:

Specify the first zone of the gripper to open.

Argument 3:

Specify the last zone of the gripper to open.

Example:

To open zones 1-2 of a gripper whose ID is 1, use the following instruction.

```
CALL PKGROPEN(Gripper ID=1,Start Zone=1,End Zone=2)
```

7.9.3 Robot-Related Programs

The two letters following [PK] are [RB] namely, the names of these programs begin with these four letters, [PKRB].

KAREL programs' basic functions are the same as the the programs' listed below.
Furthermore, with regard to the programs below, KAREL programs' basic functions are the same. Refer to Section 6.8, "KAREL PROGRAMS".

- PKRBGETID.PC
- PKRBGETSFLG.PC

7.9.3.1 PKRBGETCVCNT.PC

This program gets the conveyor count (which will be the same as the conveyor station count) for the specified robot for a specified operation (pick or place) based on analyzing the *i*RPickTool treeview data.

Argument 1:
Specify an ID of a Robot.

Argument 2:
Specify the operation, PICK or PLACE.

NOTE
When the argument guidance is supported, you select one of the following options as the operation type.
1 Pick
2 Place
However, the operation type is internally set as Pick=0 and Place=1. So if the argument guidance is not supported in your software version, specify 0 as Pick, or 1 as Place.

Argument 3:
Specify the Numeric Register in which the conveyor count is to be returned.

Example:
To get the number of conveyors used by a robot whose ID is 1 for a pick operation, use the following instruction. The conveyor count is returned in Numeric Register 7.

```
CALL PKRBGETCVCNT(Robot ID=1,Pick,ConvCount=7)
```

7.9.3.2 PKRBGETFSCNT.PC

This program gets the fixed station count for a specified operation (pick or place) by the specified robot based on analyzing the *i*RPickTool tree view data.

Argument 1:
Specify an ID of a Robot.

Argument 2:
Specify the operation, PICK or PLACE.

NOTE
When the argument guidance is supported, you select one of the following options as the operation type.
1 Pick
2 Place
However, the operation type is internally set as Pick=0 and Place=1. So if the argument guidance is not supported in your software version, specify 0 as Pick, or 1 as Place.

Argument 3:
Specify the Numeric Register in which the Fixed Station count is to be returned.

Example:
To get the number of Fixed Stations used by a robot whose ID is 1 for a place operation, use the following instruction. The Fixed Station count is returned in Numeric Register 7.

```
CALL PKRBGETFSCNT(Robot ID=1,Place,FStnCount Reg=7)
```

7.9.3.3 PKRBGETPRGRT.PC

This program gets the type of pre-grouping. The types of pre-grouping are defined as follows. For details of pre-grouping, see Section 7.11 "SETUP OF PRE-GROUPING".

0: Pre-grouping is not used.
1: Pre-grouping is used and this is the first robot that performs arrangement on the same conveyor. Virtual trays are generated by this robot.
2: Pre-grouping is used and this is the second or any successive robot that performs arrangement on the same conveyor.
3: Pre-grouping is used and this is the robot that picks all the arranged parts at one shot.

Argument 1:
Specify an ID of an infeed conveyor.

Argument 2:
Specify an ID of an conveyor station on the infeed conveyor.

Argument 3:
Specify an ID of an outfeed conveyor.

Argument 4:
Specify an ID of an conveyor station on the outfeed conveyor.

Argument 5:
Specify the Numeric Register in which the type of pre-grouping is to be returned.

Example:
The following instruction gets the type of pre-grouping for the robot that picks parts from the conveyor station whose ID is 2 on the infeed conveyor whose ID is 1 and places them onto the conveyor station whose ID is 4 on the outfeed conveyor whose ID is 2. The type of pre-grouping is returned in Numeric Register 7.

```
CALL PKRBGETPRGRT(Pk Conv ID=1,Pk CStn ID=2,Dp Conv ID=2,Dp CStn ID=4,PreGrp
Type Reg=7)
```

7.9.4 Conveyor-Related Programs

The two letters following [PK] are [CV] namely, the names of these programs begin with these four letters, [PKCV].

Furthermore, with regard to sensor-related programs, KAREL programs' basic functions are the same. Refer to Section 6.8, "KAREL PROGRAMS".

7.9.5 Sensor-Related Programs

The two letters following [PK] are [SN] namely, the names of these programs begin with these four letters, [PKSN].

Furthermore, with regard to sensor-related programs, KAREL programs' basic functions are the same. Refer to Section 6.8, "KAREL PROGRAMS".

7.9.6 Conveyor Station-Related Programs

The two letters following [PK] are [CS] namely, the names of these programs begin with these four letters, [PKCS].

Furthermore, with regard to the programs below, KAREL programs' basic functions are the same. Refer to Section 6.8, "KAREL PROGRAMS".

- PKCSGETID.PC
- PKCSCLRQUE.PC
- PKCSGETQUE.PC
- PKCSGETQCELL.PC
- PKCSACKQUE.PC
- PKCSPUTQUE.PC
- PKCSGETTIME.PC
- PKCSGETDIST.PC
- PKCSGETPFRT.PC
- PKCSGETNPRT.PC
- PKCSSETDSLIN.PC
- PKCSGETENC.PC
- PKCSGETTRID.PC
-
- PKCSCLRACK.PC

7.9.6.1 PKCSCALWPOS.PC

This program calculates a wait position for the robot at the upstream boundary of a conveyor station. This program has two outputs. Output #1 tells you if there is a need to do the wait position move or not. For example, if the part is already in the track window, there is no need to do the wait position move. Or if the robot is already at the wait position, there is no need to make the wait position move. Output #2 is the calculated wait position at the upstream boundary of the conveyor station.

Argument 1:

Specify the ID of the conveyor station where you want the wait position to be calculated. If you do not know the ID of the conveyor station, use PKCSGETID to get the ID of the conveyor station from the name of the conveyor station. For details, refer to the subsection 6.8.5.1, "PKCSGETID.PC".

Argument 2:

Specify the Position Register number where the Reference Position for the conveyor station was taught.

Argument 3:

Specify the Vision Register (VR) number that is used to get the part data for this conveyor station using the PKCSGETQUE call.

Argument 4:

Specify the Numeric Register as R[x] that you used to get the status of the PKCSGETQUE call. Note: do not enter the Numeric Register index but actually specify the R[x] which is the register containing the value of the PKCSGETQUE call.

Argument 5:

Specify the Numeric Register number that is used to store the output of this call which indicates if the move to wait position is necessary or not.

Argument 6:

Specify to the Position Register number that is used to store the calculated wait position.

Argument 7:

Specify a distance value in millimeters. This is a distance upstream of the tracking upstream boundary of the conveyor station starting from the upstream boundary. If the part is in this area, then you do not want the robot to do the wait position move. For example, specify 150 mm. The reason you may need to use this argument is that if the part is very close to the upstream boundary, then even without the wait position move, the robot will be able to pick the part without waiting because by the time the robot gets to the Conveyor station from its current position, the part is already in the tracking window of the conveyor station. You may need to adjust this number depending upon the conveyor speed.

Example:

To calculate the wait position at conveyor station 1 which uses Ref Pos PR[56], which uses VR[1] for the PKCSGETQUE call to get the part data, which uses R[10] for getting the status of the PKCSGETQUE call, and you want to know whether the wait position move is needed or not from R[11], and you want the calculated position to go to PR[40], and the no-wait-move distance is 150 mm, then use the following instruction:

```
   CALL PKCSCALWPOS(CStn ID=1,Ref Pos PR=56,Offset VR=1,GetQ Stat=R[10],
Do WaitMove Reg=11,Wait Pos PR=40,NoMove Distance=150)
```

7.9.6.2 PKCSGETAPPR.PC

This program gets and sets the approach offset of a conveyor station's reference position you specified in the OP_CS_cstnName menu in the *i*RPickTool treeview. It also returns the position register number used for the approach position.

Argument 1:

Specify an ID of a conveyor station.

Argument 2:

Specify a Numeric Register in which to return the Position Register number used to store the approach offset. This Position Register number is specified in the conveyor station operation menu of the robot (OP_CS_cstnName) in the *i*RPickTool treeview.

Example:

To get and set the approach offset of a conveyor station whose ID is 1 and return the Position Register used for the approach offset in Numeric Register 95, use the following instruction:

```
   CALL PKCSGETAPPR(CStn ID=1,ApproachPR Reg=95)
```

7.9.6.3 PKCSGETOPPR.PC

This program gets the "Reference Pos PR" of a conveyor station you specified in the OP_CS_cstnName menu in the *i*RPickTool treeview.

Argument 1:

Specify an ID of a conveyor station.

Argument 2:

Specify the Numeric Register in which to return the Position Register number of the Reference Position.

Example:
To get the Reference Position PR of a conveyor station whose ID is 1 in Numeric Register 2, use the following instruction:

```
CALL PKCSGETOPPR(CStn ID=1,RefPosPR Reg=2)
```

7.9.6.4 PKCSGETPMUL.PC

This program gets the "Pick or Drop Multiple" setting of a conveyor station (TRUE = 1, FALSE = 0) you specified in the OP_CS_cstnName menu in the *i*RPickTool treeview.

Argument 1:
Specify an ID of a conveyor station.

Argument 2:
Specify the Numeric Register in which to return the "Pick or Drop Multiple" enable or disable property.

Example:
To get the Pick or Drop Multiple property of a conveyor station whose ID is 1 in Numeric Register 4, use the following instruction:


```
CALL PKCSGETPMUL(CStn ID=1,PikMultiple Reg=4)
```

7.9.6.5 PKCSGETPPCHK.PC

This program gets the "Part Presence Check" setting of a conveyor station (TRUE = 1, FALSE = 0) you specified in the OP_CS_cstnName menu in the *i*RPickTool treeview.

Argument 1:
Specify an ID of a conveyor station.

Argument 2:
Specify the Numeric Register in which to return the "Part Presence Check" enable or disable property.

Example:
To get the Part Presence Check property of a conveyor station whose ID is 1 in Numeric Register 5, use the following instruction:

```
CALL PKCSGETPPCHK(CStn ID=1,PartPresChk Reg=5)
```

7.9.6.6 PKCSGETVR.PC

This program gets the "Part Data VR" of a conveyor station you specified in the OP_CS_cstnName menu in the *i*RPickTool treeview.

Argument 1:
Specify an ID of a conveyor station.

Argument 2:
Specify the Numeric Register in which to return the Vision Register (VR) number used for Part Data storage.

Example:

To get the Vision Register VR of a conveyor station whose ID is 1 in Numeric Register 7, use the following instruction:

```
CALL PKCSGETVR(CStn ID=1,VR Reg=7)
```

7.9.6.7 PKCSGETWPOS.PC

This program gets the "Wait Position" setting of a conveyor station (TRUE = 1, FALSE = 0) you specified in the OP_CS_cstnName menu in the *i*RPickTool treeview.

Argument 1:

Specify an ID of a conveyor station.

Argument 2:

Specify the Numeric Register in which to return the "Wait Position" enable or disable property.

Example:

To get the Wait Position property of a conveyor station whose ID is 1 Numeric Register 8, use the following instruction:

```
CALL PKCSGETWPOS(CStn ID=1,WaitPos Reg=8)
```

7.9.7 Fixed Station-Related Programs

The two letters following [PK] are [FS] namely, the names of these programs begin with these four letters, [PKFS].

Furthermore, with regard to the programs below, KAREL programs' basic functions are the same. Refer to Section 6.8, "KAREL PROGRAMS".

- PKFSGETID.PC
- PKFSPUTQUE.PC
- PKFSCLRQUE.PC
- PKFSGETQUE.PC
- PKFSACKQUE.PC
- PKFSPUTBUF.PC
- PKFSCLRBUF.PC
- PKFSGETBUF.PC
- PKFSACKBUF.PC
- PKFSGETTRID.PC
- PKFSGETUF.PC

7.9.7.1 PKFSGETAPPR.PC

This program gets and sets the approach offset of a fixed station's reference position you specified in the OP_FS_fstnName menu in the *i*RPickTool treeview. It also returns the Position Register number used for the approach position.

Argument 1:

Specify an ID of a fixed station.

Argument 2:

Specify the a Numeric Register in which to return the Position Register number used to store the

approach offset. This Position Register number is specified in the fixed station operation menu of the robot (OP_FS_fstnName) in the *i*RPickTool treeview.

Example:

To get and set the approach offset of a fixed station whose ID is 1 and return the Position Register used for the approach offset in Numeric Register 95,, use the following instruction:

```
CALL PKFSGETAPPR(FStn ID=1,ApproachPR Reg=95)
```

7.9.7.2 PKFSGETIXDLY.PC

This program gets the "Index delay" you specified in the OP_FS_fstnName menu in the *i*RPickTool treeview. The index delay is used only for simulating the indexing of a Fixed Station. Its unit is milliseconds.

Argument 1:

Specify an ID of a fixed station.

Argument 2:

Specify the Numeric Register in which to return the "Index delay".

Example:

To get the index delay of a fixed station whose ID is 1 in Numeric Register 6, use the following instruction:

```
CALL PKFSGETQTY(FStn ID=1,IndexDelay Reg=6)
```

7.9.7.3 PKFSGETIXODO.PC

This program gets the "Index Out DO" or the Digital Output port number used for indexing out a fixed station you specified in the OP_FS_fstnName menu in the *i*RPickTool treeview.

Argument 1:

Specify an ID of a fixed station.

Argument 2:

Specify the Numeric Register in which to return the "Index Out DO".

Example:

To get the "Index Out DO" port number of a fixed station whose ID is 1 in Numeric Register 7, use the following instruction:

```
CALL PKFSGETIXODO(FStn ID=1,IndexOutDO Reg=7)
```

7.9.7.4 PKFSGETIXSIM.PC

This program gets the "Simulate Indexing" setting of a conveyor station (TRUE = 1, FALSE = 0) you specified in the OP_FS_fstnName menu in the *i*RPickTool treeview.

Argument 1:

Specify an ID of a fixed station.

Argument 2:

Specify the Numeric Register in which to return the "Simulate Indexing" enable or disable property.

Example:

To get the “Simulate Indexing” property of a conveyor station whose ID is 1 Numeric Register 8, use the following instruction:

```
CALL PKFSGETIXSIM(FStn ID=1,IndexSim Reg=8)
```

7.9.7.5 PKFSGETOPPR.PC

This program gets the “Reference Pos PR” of a fixed station you specified in the OP_FS_fstnName menu in the *i*RPickTool treeview.

Argument 1:

Specify an ID of a fixed station.

Argument 2:

Specify the Numeric Register in which to return the Position Register number of the Reference Position.

Example:

To get the Reference Position PR of a fixed station whose ID is 1 in Numeric Register 2, use the following instruction:

```
CALL PKFSGETOPPR(FStn ID=1,RefPosPR Reg=2)
```

7.9.7.6 PKFSGETPMUL.PC

This program gets the “Pick or Drop Multiple” setting of a fixed station (TRUE = 1, FALSE = 0) you specified in the OP_FS_fstnName menu in the *i*RPickTool treeview.

Argument 1:

Specify an ID of a fixed station.

Argument 2:

Specify the Numeric Register in which to return the “Pick or Drop Multiple” enable or disable property.

Example:

To get the Pick or Drop Multiple property of a fixed station whose ID is 1 in Numeric Register 4, use the following instruction:

```
CALL PKFSGETPMUL(FStn ID=1,PikMultiple Reg=4)
```

7.9.7.7 PKFSGETPPCHK.PC

This program gets the “Part Presence Check” setting of a fixed station (TRUE = 1, FALSE = 0) you specified in the OP_FS_fstnName menu in the *i*RPickTool treeview.

Argument 1:

Specify an ID of a fixed station.

Argument 2:

Specify the Numeric Register in which to return the “Part Presence Check” enable or disable property.

Example:

To get the Part Presence Check property of a fixed station whose ID is 1 in Numeric Register 5, use the following instruction:

```
CALL PKFSGETPPCHK(FStn ID=1,PartPresChk Reg=5)
```

7.9.7.8 PKFSGETRDY.PC

This program gets the internal status of "fixed station ready" (TRUE = 0, FALSE = 1). When you run the robot with a fixed station, *i*RPickTool internally tracks the "ready" status. If the ready status is FALSE, then the robot is not allowed to pick or place at the Fixed Station. The conditions for ready to be true are normally that the Station Ready DI is high and the Index DO is low. However, during indexing of the Fixed Station, *i*RPickTool first sets the internal status of ready to be FALSE. It then monitors for the Station Ready DI to go low and then come back high (Conceptually, the previous fixed station leaves and a new fixed station comes in). When those conditions are satisfied and the Index DO is low, the internal status is set to TRUE again.

Argument 1:

Specify an ID of a fixed station.

Argument 2:

Specify the Numeric Register in which to return the internal "Ready" status of a Fixed Station.
1: Ready
0: Not Ready

Example:

To get the internal "ready" status of a fixed station whose ID is 1 Numeric Register 8, use the following instruction:

```
CALL PKFSGETRDY(FStn ID=1,FStnReady Reg=8)
```

7.9.7.9 PKFSGETRDYDI.PC

This program gets the "Ready DI" or the Digital Input port number used for checking if the fixed station is present (or ready to pick or place) that you specified in the OP_FS_fstnName menu in the *i*RPickTool treeview.

Argument 1:

Specify an ID of a fixed station.

Argument 2:

Specify the Numeric Register in which to return the "Ready DI" port number.

Example:

To get the "Ready DI" port number of a fixed station whose ID is 1 in Numeric Register 7, use the following instruction:

```
CALL PKFSGETRDYDI(FStn ID=1,ReadyDI Reg=7)
```

7.9.7.10 PKFSGETVR.PC

This program gets the “Part Data VR” of a fixed station you specified in the OP_CS_fstnName menu in the *i*RPickTool treeview.

Argument 1:
Specify an ID of a fixed station.

Argument 2:
Specify the Numeric Register in which to return the Vision Register (VR) number used for Part Data storage.

Example:
To get the Vision Register VR of a fixed station whose ID is 1 in Numeric Register 7, use the following instruction:

```
CALL PKFSGETVR(FStn ID=1,VR Reg=7)
```

7.9.7.11 PKFSGETMONIDX.PC

This program is to be used immediately after indexing a Fixed Station. It sets the internal status of station ready to FALSE. It then starts monitors to watch the Station Ready DI going low (station is indexed) and then to come high again (new Fixed station in place). When that happens it sets the internal station ready status to TRUE. (A robot will not be allowed inside a Fixed Station unless this status is TRUE).

Argument 1:
Specify an ID of a fixed station.

Argument 2:
Specify the Numeric Register in which to return the status.

Example:
To start the monitoring of the Fixed station Ready DI to go low and then high, use the following instruction. The status is returned in Numeric register 7. If the status is non-zero, then an error occurred while attempting to start the monitors.

```
CALL PKFSGETVR(FStn ID=1,VR Reg=7)
```

7.9.7.12 PKFSSETRDY.PC

This program sets the internal status of “fixed station ready” (“Ready” or “NotReady”). Refer to the KAREL macro 7.9.7.8 “PKFSGETRDY.PC” to know the details of how *i*RPickTool handles the station ready status.

Argument 1:
Specify an ID of a fixed station.

Argument 2:
Specify “Ready” or “NotReady” to set the internal status for “fixed station ready”.

NOTE
When the argument guidance is supported, you select one of the following options.
1 Ready
2 NotReady
However, the status of the fixed station is internally set as Ready=1 and NotReady=0. So if the argument guidance is not supported in your software version, specify 1 as Ready, or 0 as NotReady.

Example:
To set the internal "ready" status of a fixed station whose ID is 1 to "Not Ready", use the following instruction:

```
CALL PKFSSETRDY(FStn ID=1,NotReady)
```

7.9.8 Tray-Related Programs

The two letters following [PK] are [TR] namely, the names of these programs begin with these four letters, [PKTR].

Furthermore, with regard to tray-related programs, KAREL programs' basic functions are the same. Refer to Section 6.8, "KAREL PROGRAMS".

7.10 STANDARD TP PROGRAMS

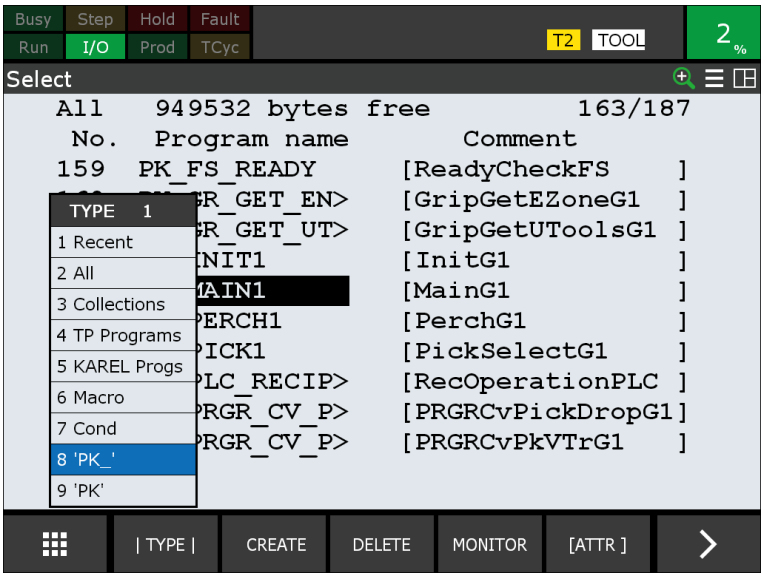
This section describes the standard TP programs provided with the J945: *i*RPickTool option. These programs allow you to run any picking or visual tracking application based on the data you setup in the *i*RPickTool treeview user interface. You are allowed to and encouraged to customize these programs to meet the detailed requirements of your application. But these programs will allow you to be more productive because you have to write less code and also help you to get picking application working quickly. If you are a System Integrator, then once you become familiar with these programs which have been completely tested and validated by FANUC, you will find it easier and quicker to develop solutions for your various customers to meet a wide variety of their requirements.

The TP programs support
All the FANUC robots. The robots can be Delta robots like the M-3*i*A/6S, M-2*i*A/3S, the LR Mate series like the LR Mate 200*i*D, or any 3, 4, 5 or 6 axis robots.
Applications where the robots pick from a conveyor and places on a conveyor
Applications where the robots pick from a conveyor and place to a Fixed Station or vice-versa
Applications where the robots pick from a fixed station and places on a fixed station.
The picking or placing can be individual parts or tray cells
The picking or placing can be single or multiple parts (upto 20).
Many other functions like automatic move to wait position when there are no parts, part presence checks for the gripper, etc.

7.10.1 TP Program Naming Conventions

All the *i*RPickTool TP programs begin with “PK_”.

In the teach pendant’s SELECT menu, when you press the TYPE pull-up, you can quickly group and view only the *i*RPickTool programs by selecting the PK_ filter. See the figure below.
You may use the “PK” filter to view all the *i*RPickTool KAREL macros.



The TP Programs are named based on the following two conventions which will help you to find the program you are looking for quickly.

Name = PK_ + Name
Name = PK_ + Name + group
Name = PK_ + Object_name + Action_verb + group + sequence

The object names are two letter abbreviations.
CV: Conveyor
CS: Conveyor Station
FS: Fixed Station
GR: Gripper

The action verb examples are:
GET_PICK_DATA11
PICK11
DROP11

Note that the action verbs end with some digits:
When there are two digits, the first digit refers to the motion group number and the second number represents the “Sequence”.

For example PICK11 means Pick with Group 1 and Sequence 1.

There are some programs which use a simpler convention such as
Name = PK_ + Name + group
For example:
PK_MAIN1: Main program for the group 1 robot
PK_INIT1: Initialization (of data) program for the group 1 robot.

PK_PERCH1: Move to perch program for the group 1 robot.

When you have additional groups, you can use the same convention to name the programs. This will help you to manage your TP programs effectively.

7.10.2 TP Program Details

7.10.2.1 List of all the TP Programs

The following is the list of all the Group 1 TP Programs.

No.	Program name	Comment
2	PK_CV_DROP11	[DropG1Cv1]
3	PK_CV_GET_DROP_DATA11	[GetDpDataG1Cv1]
4	PK_CV_GET_PICK_DATA11	[GetPkDataG1Cv1]
5	PK_CV_PICK11	[PickG1Cv1]
6	PK_CV_WAITPOS1	[WaitG1Cv]
7	PK_CYCSTOP1	[CycleStopG1]
8	PK_DROP1	[DropSelectG1]
9	PK_FS_DROP11	[DropG1FS1]
10	PK_FS_GET_DROP_DATA11	[GetPkDataG1FS1]
11	PK_FS_GET_PICK_DATA11	[GetPkDataG1FS1]
12	PK_FS_INDEX	[IndexFS]
13	PK_FS_PICK11	[PickG1FS1]
14	PK_FS_READY	[ReadyCheckFS]
15	PK_GR_GET_ENDZONE1	[GripGetEZoneG1]
16	PK_GR_GET_UTOOLS1	[GripGetUToolsG1]
17	PK_INIT1	[InitG1]
18	PK_MAIN1	[MainG1]
19	PK_PERCH1	[PerchG1]
20	PK_PICK1	[PickSelectG1]

7.10.2.2 PK_MAIN1.TP: Main Program for Group 1

PK_MAIN1.TP is the main program. By default, this program is run automatically when a UOP cycle start (for example from a PLC) is received.
This program has the overall sequence of steps which are:

1 Initialize Data: Get the data set in the *i*RPickTool tree-view and assign them to registers. Set UTOOLs and UFRAMEs. During initialization, determine the pick and place programs to run – primarily if the programs to run are tracking programs or non-tracking programs.
2 Start the sensor task to be ready to detect parts from the camera, DI or HDI sensors and put them into the part queue.
3 Move the robot to the perch position
4 When pre-grouping is used, the main program for pre-grouping PK_PRGR_MAIN1.TP is called.
5 When pre-grouping is not used, do the pick and place indefinite loop. Always check if cycle stop is invoked at the end of the pick operation and the end of the place operation.
6 If cycle stop is invoked, process the cycle stop function.
7 Upon exit from the pick and place loop (for example, via cycle stop) or exit from the main program for pre-grouping, stop the sensor task.

PK_MAIN1.TP

```
    1:  ! Main Prog for Grp:1
    2:
    3:  LBL[10]
    4:  -- Initializations
    5:  CALL PK_INIT1
    6:  -- Run sensor task
    7:  CALL PKSNSTART
    8:  -- Move to perch
    9:  CALL PK_PERCH1
   10:  ! Pre-grouping is done by
   11:  ! PK_PRGR_MAIN1.TP
   12:  !R[230] is set in PK_PRGR_INIT1.
   13:  !TP called in PK_INIT1.TP
   14:  IF R[230:PRGR Robot Type]<=0 OR R[230:PRGR Robot Type]>2,JMP LBL[100]
   15:  CALL PK_PRGR_MAIN1
   16:  JMP LBL[10000]
   17:
   18:  LBL[100:pk n pl]
   19:  -- Pick the part
   20:  CALL PK_PICK1
   21:  IF R[101:Cycle Stop G1]>0,JMP LBL[10000]
   22:  -- Drop the part
   23:  CALL PK_DROP1
   24:  LBL[200]
   25:  IF R[101:Cycle Stop G1]>0,JMP LBL[10000]
   26:  JMP LBL[100]
   27:
   28:  LBL[10000: End]
   29:  -- Stop the sensor task
   30:  CALL PKSNSTOP
[End]
```

7.10.2.3 PK_INIT1.TP

PK_INIT1.TP is the initialization program for the group 1 main task which is PK_MAIN1.TP.
The initialization consists of:

1 Checking if *i*RPickTool data is synchronized across all controllers.
2 Getting the active recipe id.
3 Determining the Robot ID for the group 1 robot.
4 Determining the Robot's gripper ID.
5 Getting the zone data of the gripper and assigning to UTOOLs.
6 Setting the P[1] position of the PK_PERCH1.TP program whenever a recipe is restored or switched to using PKCOPYPR2POS KAREL macro.
7 Determining if the pick station is a conveyor station or a fixed station.
8 Determining if the place station is a conveyor station or a fixed station.
9 Get all the data from the treeview for the robot and the required conveyor and fixed stations.
10 The buffer id for the group 1 robot is obtained.
If buffer id > 0, then PK_BF_GET_DATA1.TP is called which gets the buffer system variable data into registers used by the pick and drop buffer programs. The PK_BF_GET_DATA1.TP program is available only when the R857: *i*RPickTool / Option Package Add-on (for North America, R827: iRPickTool / Production Control Add-on) option is installed.
11 Getting type for pre-grouping. When pre-grouping is not used, the type is 0.
12 Initialize or reset certain registers like the part presence check registers.

PK_INIT1.TP

```
 1:  ! PKINIT1
 2:  ! Check data is synchronized and
 3:  ! load balance data is valid
 4:  CALL PKWCCHECK
 5:
 6:  ! Group 1 inits
 7:  R[99:Active Recipe]=$PKWC.$ACT_RC_ID
 8:
 9:  ! Get robot from group
10:  CALL PKRBGETID(Group Num=1,Robot ID Reg=103)
11:
12:  ! Get gripper for robot
13:  ! PKGRGETID(rob_id, reg_gripId)
14:  CALL PKGRGETID(Robot ID=R[103:Robot Id G1],Gripper ID Reg=104)
15:  ! Set $MNUTOOLs from Gripper Zone
16:  CALL PK_GR_GET_UTOOLS1(Robot ID=R[103:Robot Id G1],
      Gripper ID=R[104:Gripper Id G1])
17:
18:
19:  -- If recipe was loaded, it set perch position to PR[54]
20:  -- Copy PR[54] to P[1] of PK_PERCH1.TP.
21:  IF R[105:Recipe loaded G1]<>1,JMP LBL[25]
22:  -- Set the UTOOL_NUM that should be in P[1]
23:  UTOOL_NUM=0
24:  CALL PKCOPYPR2POS(PR Num=54,Program Name='PK_PERCH1',
      Position Num=1,Group Num=1)
25:  R[105:Recipe loaded G1]=0
26:  LBL[25]
27:
28:  LBL[50: Pick Inits]
29:  ! Get the count of pick conveyors
30:  !   for this robot
31:  CALL PKRBGETCVCNT(Robot ID=R[103:Robot Id G1],Pick=0,ConvCount Reg=95)
32:  IF R[95:TemporaryUse1]=0,JMP LBL[100]
33:  ! Get 1st pick conveyor's data
34:  CALL PK_CV_GET_PICK_DATA11
35:  JMP LBL[400]
36:
37:  LBL[100: GetPkFStn]
38:  ! Get the count of pick FStns
39:  !    for this robot
40:  CALL PKRBGETFSCNT(Robot ID=R[103:Robot Id G1],Pick=0,FStnCount Reg=95)
41:  IF R[95:TemporaryUse1]=0,JMP LBL[300]
42:  ! Get pick FStn's data
43:  CALL PK_FS_GET_PICK_DATA11
44:  JMP LBL[400]
45:
46:  LBL[300: No Pick Stns]
47:  UALM[3]
48:  JMP LBL[10000]
49:
50:
51:  LBL[400: Drop Inits]
52:  ! Get the count of drop conveyors
```

```
   53:   !   for this robot
   54:   CALL PKRBGETCVCNT(Robot ID=R[103:Robot Id G1],Place=1,ConvCount Reg=95)
   55:   IF R[95:TemporaryUse1]=0,JMP LBL[500]
   56:   ! Get drop conveyor's data
   57:   CALL PK_CV_GET_DROP_DATA11
   58:   JMP LBL[1000]
   59:
   60:   LBL[500: GetDpFStn]
   61:   ! Get the count of drop FStns
   62:   !   for this robot
   63:   CALL PKRBGETFSCNT(Robot ID=R[103:Robot Id G1],Place=1,FStnCount Reg=95)
   64:   IF R[95:TemporaryUse1]=0,JMP LBL[600]
   65:   ! Get drop FStn's data
   66:   CALL PK_FS_GET_DROP_DATA11
   67:   JMP LBL[1000]
   68:
   69:   LBL[600: No Drop Stns]
   70:   UALM[4]
   71:   JMP LBL[10000]
   72:
   73:
   74:   LBL[1000: Buf Inits]
   75:   ! Buffer initializations
   76:   CALL PKRBGETBUFID(Robot ID=R[103:Robot Id G1],BufferID Reg=199)
   77:   IF R[199:Buf Id G1]<=0,JMP LBL[2000]
   78:   CALL PK_BF_GET_DATA1
   79:
   80:   LBL[2000:Pre-group Inits]
   81:   ! Check for Pre-group option
   82:   !PKPRGRGETEN(Enb_Reg)
   83:   CALL PKPRGRGETEN(PreGrp Enb Reg=230)
   84:   IF R[230:PRGR Robot Type]=0,JMP LBL[3000]
   85:   !Check if pre-group is used
   86:   CALL PK_PRGR_INIT1(PreGrp Type Reg=230)
   87:   IF R[230:PRGR Robot Type]=0,JMP LBL[3000]
   88:   IF R[199:Buf Id G1]<>0,JMP LBL[2025]
   89:   JMP LBL[3000]
   90:   LBL[2025]
   91:   ! Cannot use both PRGR and Buffer
   92:   UALM[6]
   93:
   94:   LBL[3000:User Init]
   95:   ! User initialization
   96:   R[117:Pk1x PPStat]=0
   97:   R[137:Dp1x PPStat]=0
   98:
   99:   LBL[10000:End]
[End]
```

7.10.2.4 PK_PERCH1.TP

This program moves the group 1 robot to the perch position using UTOOL 0 and UFRAME 0 in the robot world coordinates. Note that P[1] is the perch position. In Line 17: the perch position P[1] is assigned to PR[54: Perch Pos]. This line is needed because when you save a recipe, it saves the perch position from PR[54] only.

PK_PERCH1.TP

```
    1:  -- Move to Perch G1
    2:
    3:  UTOOL_NUM=0
    4:  UFRAME_NUM=0
    5:  !*********************************
    6:  -- NOTE: The recipe stores the
    7:  -- perch position from PR[54].
    8:  -- If you restore a recipe or
    9:  -- switch to a new recipe, then
   10:  -- P[1] below will be set to the
   11:  -- PR[54] value read from the
   12:  -- recipe when PK_INIT1.TP runs.
   13:  !*********************************
   14:  PAYLOAD[1]
   15:J P[1] 25% FINE
   16:  -- Copy P[1] to PR[54] stored in the recipe
   17:  PR[54:Perch Pos]=P[1]
[End]
```

7.10.2.5 PK_PICK1.TP

This program decides if a tracking or non-tracking program is used for the pick move by the group 1 robot.

This program normally uses PK_CV_PICK11.TP for picking from an infeed conveyor. However, if a buffer is used, then it executes PK_BUF_PICK1.TP which contains the buffer algorithm to decide whether the part should be picked from the infeed conveyor or from the buffer. The buffer logic should only executes only when the R857: *i*RPickTool / Option Package Add-on (for North America, R827: iRPickTool / Production Control Add-on) is installed. For additional details on the buffer logic in this program, please refer to the section 7.12 "BUFFER FUNCTION".

When pre-grouping is used, this program uses PK_PRGR_CV_PICKVTRAY1.TP for picking all the arranged parts at one shot. For details, see Section 7.11 "SETUP OF PRE-GROUPING".

PK_PICK1.TP

```
    1:  !Pick program G1
    2:  !Choose trk or non-trk pick prog
    3:
    4:  R[107:Counter G1]=1
    5:  IF R[109:Pk1x CvId]=0,JMP LBL[1000]
    6:  IF (R[199:Buf Id G1]<>0),JMP LBL[50]
    7:  IF R[230:PRGR Robot Type]=3,JMP LBL[60]
    8:  ! Pick from conveyor
    9:  CALL PK_CV_PICK11
   10:  JMP LBL[10000]
   11:  ! Buffer handling
   12:  LBL[50: Buf loop]
   13:  CALL PK_BUF_PICK1
```

```
   14:  JMP LBL[10000]
   15:  LBL[60:Virtual Tray]
   16:  !Pick virtual tray
   17:  CALL PK_PRGR_CV_PICKVTRAY1
   18:  JMP LBL[10000]
   19:
   20:  LBL[1000: FStn]
   21:  ! Non-tracking pick from FStn
   22:  CALL PK_FS_PICK11
   23:  LBL[10000: End]
[End]
```

7.10.2.6 PK_DROP1.TP

This program decides if a tracking or non-tracking program is used for the place (or drop) move by the group 1 robot.

This program normally uses PK_CV_DROP11.TP for placing to an outfeed conveyor. However, if a buffer is used, then it executes PK_BUF_DROP1.TP which contains the buffer algorithm to decide whether the part should be placed in the outfeed conveyor or in the buffer. The buffer logic executes only when the R857: *i*RPickTool / Option Package Add-on is installed. For additional details on the buffer logic (for North America, R827: iRPickTool / Production Control Add-on), please refer to the section 7.12 "BUFFER FUNCTION".

PK_DROP1.TP

```
    1:  !Drop program G1
    2:  !Choose trk or non-trk drop prog
    3:
    4:  R[107:Counter G1]=1
    5:  IF R[129:Dp1x CvId]=0,JMP LBL[1000]
    6:  IF (R[199:Buf Id G1]<>0),JMP LBL[50]
    7:  !Place on conveyor
    8:  CALL PK_CV_DROP11
    9:  JMP LBL[10000]
   10:  ! Buffer handling
   11:  LBL[50: Buffer drop]
   12:  CALL PK_BUF_DROP1
   13:  JMP LBL[10000]
   14:
   15:  LBL[1000: FixedPos]
   16:  ! Non-tracking drop at FStn
   17:  CALL PK_FS_DROP11
   18:  LBL[10000: End]
[End]
```

7.10.2.7 PK_GR_GET_UTOOLS1.TP

This program sets the UTOOL for each zone of the gripper used by the group 1 robot. This program

1 Uses the KAREL macro PKGRGETZNCNT to get the zone count of the specified gripper
2 Uses the KAREL macro PKGRGETZNUT to get the UTOOL value specified in the Zone menu of the Gripper in the *i*RPickTool treeview.
3 Sets the retrieved UTOOL of the zone to the system's UTOOL.

This program is called by two arguments.
Argument 1: Robot ID
Argument 2: Gripper ID.

PK_GR_GET_UTOOLS1.TP

```
   1:  -- PK_GR_GET_UTOOLS1
   2:  -- Assign Gripper zone
   3:  -- Utools for grp 1 robot
   4:  -- Also set pick and place qty
   5:  -- Arg 1: Robot id
   6:  -- Arg 2: Gripper id
   7:
   8:  -- Get num zones in gripper
   9:  CALL PKGRGETZNCNT( "Gripper ID" =AR[2], "ZoneCount Reg" =96)
  10:  FOR R[95:TemporaryUse1]=1 TO R[96:TemporaryUse2]
  11:  --eg: Get each zone UT to PR[1]
  12:  CALL PKGRGETZNUT( "Robot ID" =AR[1],
       "Gripper ID" =AR[2], "Zone Index" =R[95:TemporaryUse1], "ZoneUTool PR" =51)
  13:  --eg: Assign zone UT to UTOOL
  14:  UTOOL[R[95]]=PR[51:TemporaryUse1]
  15:  ENDFOR
  16:  ! PickQty = PlaceQty = ZoneCnt
  17:  R[111:Pk1x Qty]=R[96:TemporaryUse2]
  18:  R[131:Dp1x Qty]=R[96:TemporaryUse2]
[End]
```

7.10.2.8 PK_CV_GET_PICK_DATA11.TP

This program is used to get the data setup in the *i*RPickTool treeview for a pick operation at a conveyor station of a conveyor. This program works as follows.

1 Gets the conveyor ID of the first pick from $PKMAIN[1].$PK_CV[1] (which is initialized when the KAREL macro PKRBGETCVCNT.PC is executed).
2 Get the name of the first conveyor to String register 4 (SR[4])
3 Gets the ID of the conveyor station on the conveyor using the KAREL macro PKCVGETNAME.PC.
4 Gets the "Pick or drop multiple" property for the pick operation using the KAREL macro PKCSGETPMUL.PC.
5 Gets the approach offset of the conveyor station from the treeview data and sets it to the approach offset Position Register using the KAREL macro PKCSGETAPPR.PC.
6 Gets the FLG used for checking the "Skip Outbound" condition in the tracking pick program using the KAREL macro PKRBGETSFLG.PC.
7 Gets the ID of the tray used on the conveyor using the KAREL macro PKCSGETTRID.
8 Clears any pending ACK to avoid the "GetQueue before Ackque" error using the KAREL macro PKCSCLRACK.PC.
9 Clears the queue of the conveyor station based on the setting of the system variable $PKCLRQ using the KAREL macro PKCSCLRQUE.PC.

⚠ CAUTION

The register R[95:TemporaryUse1] is used multiple times in this program. Sometimes the value of R[95] is not used at all as in line 12: CALL PKCSGETAPPR(CStn ID=R[110:Pk1x StnId], ApproachPR Reg=95). For your customization, if you need the value for later use, you should assign an unused register to it. But such calls cannot be removed because they may perform additional function. For example the above call also sets the approach offset. Refer to the description of the KAREL macro for details.

PK_CV_GET_PICK_DATA11.TP

```
   1:  -- PK_CV_GET_PICK_DATA11
   2:  -- Get Pick Data for G1 pick     cnv1
   3:  --       & assign to registers used
   4:  --       in PK_CV_PICK11.TP
   5:
   6:  -- Get the actual 1st pick conveyor's id
   7:  R[109:Pk1x CvId]=$PKMAIN[1].$PK_CV[1]
   8:  -- Get the conveyor name
   9:  CALL PKCVGETNAME( “Conv ID” =R[109:Pk1x CvId],“ConvName SR” =4)
  10:  CALL PKCSGETID(“CONV1” =SR[4],“CStn ID Reg” =110)
  11:  CALL PKCSGETPMUL(“CStn ID” =R[110:Pk1x StnId],“PikMultiple Reg” =112)
  12:  CALL PKCSGETAPPR(“CStn ID” =R[110:Pk1x StnId],“ApproachPR Reg” =95)
  13:  CALL PKRBGETSFLG(“Robot ID” =R[103:Robot Id G1],“SkipFlag Reg” =102)
  14:  !Get CStn's tray id
  15:  CALL PKCSGETTRID(“CStn ID” =R[110:Pk1x StnId],“Tray ID Reg” =121)
  16:  -- Clear any pending acks
  17:  CALL PKCSCLRACK(“CStn ID” =R[110:Pk1x StnId])
  18:  ! Clear the queues based on
  19:  ! $PKCLRQ for G1 CStns
  20:  R[95:TemporaryUse1]=$PKCLRQ
  21:  IF R[95:TemporaryUse1]=0,JMP LBL[5000]
  22:  CALL PKCSCLRQUE(“CStn ID” =R[110:Pk1x StnId])
  23:
  24:  LBL[5000: Eff Tools]
  25:  !Get position check flag
  26:  !PKSCSGETPOSCK(cs_id,
  27:  ! operation, reg_PosCheck)
  28:  CALL PKCSGETPOSCK(“CStn ID” =R[110:Pk1x StnId],
       “Pick” =0,“PosCheck Reg” =115)
  29:  LBL[10000:End]
[End]
```

7.10.2.9 PK_CV_GET_DROP_DATA11.TP

This program is used to get the data setup in the *i*RPickTool treeview for a drop or place operation at a conveyor station of a conveyor. This program works as follows.

1 Gets the conveyor ID of the first drop from $PKMAIN[1].$DP_CV[1] (which is initialized when the KAREL macro PKRBGETCVCNT.PC is executed).
2 Get the name of the first conveyor to String register 5 (SR[5]
3 Gets the ID of the conveyor station on the conveyor using the KAREL macro PKCVGETNAME.PC.
4 Gets the "Pick or drop multiple" property for the pick operation using the KAREL macro PKCSGETPMUL.PC.
5 Gets the approach offset of the conveyor station from the treeview data and sets it to the approach offset Position Register using the KAREL macro PKCSGETAPPR.PC.
6 Gets the FLG used for checking the "Skip Outbound" condition in the tracking pick program using the KAREL macro PKRBGETSFLG.PC.
7 Gets the ID of the tray at the Conveyor using the KAREL macro PKCSGETTRID.PC
8 Clears any pending ACK to avoid the "GetQueue before Ackque" error using the KAREL macro PKCSCLRACK.PC.
9 Clears the queue of the conveyor station based on the setting of the system variable $PKCLRQ using the KAREL macro PKCSCLRQUE.PC.

7

> ⚠ **CAUTION**
> The register R[95:TemporaryUse1] is used multiple times in this program. Sometimes the value of R[95] is not used at all as in line 12: CALL PKCSGETAPPR(CStn ID=R[130:Dp1x StnId], ApproachPR Reg=95). For your customization, if you need the value for later use, you should assign an unused register to it. But such calls cannot be removed because they may perform additional function. For example the above call also sets the approach offset. Refer to the description of the KAREL macro for details.

PK_CV_GET_DROP_DATA11.TP

```
   1:  -- PK_CV_GET_DROP_DATA11
   2:  -- Get Drop Data for G1 Drop cnv1
   3:  -- & assign to registers used
   4:  -- in PK_CN_DROP11.TP
   5:
   6:  --   Get the actual 1st drop conveyor's name
   7:  R[129:Dp1x CvId]=$PKMAIN[1].$DP_CV[1]
   8:  --   Get the conveyor name
   9:  CALL PKCVGETNAME( "Conv ID" =R[129:Dp1x CvId], "ConvName SR" =5)
  10:  CALL PKCSGETID( "CONV1" =SR[5], "CStn ID Reg" =130)
  11:  CALL PKCSGETPMUL( "CStn ID" =R[130:Dp1x StnId], "PikMultiple Reg" =132)
  12:  CALL PKCSGETAPPR( "CStn ID" =R[130:Dp1x StnId], "ApproachPR Reg" =95)
  13:  CALL PKRBGETSFLG( "Robot ID" =R[103:Robot Id G1], "SkipFlag Reg" =102)
  14:  !Get CStn's tray id
  15:  CALL PKCSGETTRID( "CStn ID" =R[130:Dp1x StnId], "Tray ID Reg" =141)
  16:  -- Clear any pending acks
  17:  CALL PKCSCLRACK( "CStn ID" =R[130:Dp1x StnId])
  18:  ! Clear the queues based on
[End]
```

7.10.2.10 PK_FS_GET_PICK_DATA11.TP

This program is used to get the data setup in the *i*RPickTool treeview for a pick operation at a Fixed station. This program

1 Gets the Fixed Station ID of the first pick from $PKMAIN[1].$PK_FS[1] (which is initialized when the KAREL macro PKRBGETFSCNT.PC is executed).
2 Gets the "Pick or drop multiple" property for the pick operation using the KAREL macro PKFSGETPMUL.PC.
3 Gets the approach offset of the Fixed station from the treeview data and sets it to the approach offset Position Register using the KAREL macro PKFSGETAPPR.PC.
4 Gets the Station Ready DI of the Fixed Station (for later station_ready? Check) using the KAREL macro PKFSGETRDYDI.PC.
5 Gets the Index Out DO digital output of the Fixed Station (for later indexing the fixed station) using the KAREL macro PKFSGETIXODO.PC.
6 Sets the initial value of the internal station_ready flag to FALSE (later if station is indeed foud to be ready during the station_ready? check, it will be set to TRUE) using the KAREL macro PKFSSETRDY.PC.
7 Checks if "simulate indexing" is enabled, and if so simulates the Station Ready DI port to go high using the KAREL macro PKFSGETIXSIM.PC
8 Gets the ID of the tray at the Fixed Station using the KAREL macro PKFSGETTRID.PC
9 Clears any pending ACK to avoid the "GetQueue before Ackque" error using the KAREL macro PKFSCLRACK.PC.
10 Initializes the tray using the KAREL macros PKFSCLRAUE.PC and PKFSPUTQUE.PC based on the $PKCLRQ system variable. You can modify this initialization based on DI signals or other means to suit your convenience.

> ⚠ **CAUTION**
> The register R[95:TemporaryUse1] is used multiple times in this program. Sometimes the value of R[95] is not used at all as in line 9: CALL PKFSGETAPPR(FStn ID=R[110:Pk1x StnId], ApproachPR Reg=95). For your customization, if you need the value for later use, you should assign an unused register to it. But such calls cannot be removed because they may perform additional function. For example the above call also sets the approach offset. Refer to the description of the KAREL macro for details.

PK_FS_GET_PICK_DATA11.TP

```
 1:  -- PK_FS_GET_PICK_DATA11
 2:  -- Grp 1 Seq 1
 3:  -- Set Num Reg for FStn
 4:
 5:  -- Get the actual 1st pick
  :  Fstn's name
 6:  R[109:Pk1x CvId]=0
 7:  R[110:Pk1x StnId]=$pkmain[1].$pk_fs[1]
 8:  CALL PKFSGETPMUL( "FStn ID" =R[110:Pk1x StnId], "PikMultiple Reg" =112)
 9:  CALL PKFSGETAPPR( "FStn ID" =R[110:Pk1x StnId], "ApproachPR Reg" =95)
10:  CALL PKFSGETRDYDI( "FStn ID"=R[110:Pk1x StnId], "ReadyDI Reg"=118)
11:  CALL PKFSGETIXODO( "FStn ID"=R[110:Pk1x StnId], "IndexOutDO Reg"=119)
12:  CALL PKFSSETRDY( "FStn ID"=R[110:Pk1x StnId], "Ready"=1)
13:  CALL PKFSGETIXSIM( "FStn ID"=R[110:Pk1x StnId], "IndexSim Reg"=95)
14:  CALL PKFSGETTRID( "FStn ID"=R[110:Pk1x StnId], "Tray ID Reg"=121)
15:  CALL PKFSCLRACK( "FStn ID"=R[110:Pk1x StnId])
```

```
   16:   IF R[121:Pk1x TrayId]=0,JMP LBL[10000]
   17:   R[95:TemporaryUse1]=$PKCLRQ
   18:   IF R[95:TemporaryUse1]=0,JMP LBL[1000]
   19:   CALL PKFSCLRQUE("FStn ID"=R[110:Pk1x StnId])
   20:   CALL PKFSPUTQUE("FStn ID"=R[110:Pk1x StnId])
   21:   --   Reset tray cell counter
   22:   R[122:Pk1x TrayCtr]=1
   23:   LBL[1000]
   24:
   25:   LBL[10000: End]
[End]
```

7.10.2.11 PK_FS_GET_DROP_DATA11.TP

This program is used to get the data setup in the *i*RPickTool treeview for a drop or place operation at a Fixed station. This program works as follows.

1 Gets the Fixed Station ID of the first place from $PKMAIN[1].$DP_FS[1] (which is initialized when the KAREL macro PKRBGETFSCNT.PC is executed).
2 Gets the "Pick or drop multiple" property for the pick operation using the KAREL macro PKFSGETPMUL.PC.
3 Gets the approach offset of the Fixed station from the treeview data and sets it to the approach offset Position Register using PKFSGETAPPR.PC.
4 Gets the Station Ready DI of the Fixed Station (for later station_ready? Check) using the KAREL macro PKFSGETRDYDI.PC.
5 Gets the Index Out DO digital output of the Fixed Station (for later indexing the fixed station) using the KAREL macro PKFSGETIXODO.PC.
6 Sets the initial value of the internal station_ready flag to FALSE (later if station is indeed foud to be ready during the station_ready? check, it will be set to TRUE) using the KAREL macro PKFSSETRDY.PC.
7 Checks if "simulate indexing" is enabled, and if so simulates the Station Ready DI port to go high using the KAREL macro PKFSGETIXSIM.PC
8 Gets the ID of the tray at the Fixed Station using the KAREL macro PKFSGETTRID.PC
9 Clears any pending ACK to avoid the "GetQueue before Ackque" error using the KAREL macro PKFSCLRACK.PC.
10 Initializes the tray using the KAREL macros PKFSCLRAUE.PC and PKFSPUTQUE.PC based on the $PKCLRQ system variable.

> ⚠ **CAUTION**
> The register R[95:TemporaryUse1] is used multiple times in this program. Sometimes the value of R[95] is not used at all as in line 9: CALL PKFSGETAPPR(FStn ID=R[130:Dp1x StnId], ApproachPR Reg=95). For your customization, if you need the value for later use, you should assign an unused register to it. But such calls cannot be removed because they may perform additional function. For example the above call also sets the approach offset. Refer to the description of the KAREL macro for details.

PK_FS_GET_DROP_DATA11.TP

```
    1:  -- PK_FS_GET_DROP_DATA11
    2:  --   Grp 1 Seq 1
    3:  --   Set Num Reg for FStn
    4:
    5:  --   Get the actual 1st pick
     :  Fstn's name
    6:  R[129:Dp1x CvId]=0
    7:  R[130:Dp1x StnId]=$pkmain[1].$dp_fs[1]
    8:  CALL PKFSGETPMUL("FStn ID"=R[130:Dp1x StnId],"PikMultiple Reg"=132)
    9:  CALL PKFSGETAPPR("FStn ID"=R[130:Dp1x StnId],"ApproachPR Reg"=95)
   10:  CALL PKFSGETRDYDI("FStn ID"=R[130:Dp1x StnId],"ReadyDI Reg"=138)
   11:  CALL PKFSGETIXODO("FStn ID"=R[130:Dp1x StnId],"IndexOutDO Reg"=139)
   12:  CALL PKFSSETRDY("FStn ID"=R[130:Dp1x StnId],"Ready"=1)
   13:  CALL PKFSGETIXSIM("FStn ID"=R[130:Dp1x StnId],"IndexSim Reg"=95)
   14:  CALL PKFSGETTRID("FStn ID"=R[130:Dp1x StnId],"Tray ID Reg"=141)
   15:  CALL PKFSCLRACK("FStn ID"=R[130:Dp1x StnId])
   16:  IF R[141:Dp1x TrayId]=0,JMP LBL[10000]
   17:  R[95:TemporaryUse1]=$PKCLRQ
   18:  IF R[95:TemporaryUse1]=0,JMP LBL[1000]
   19:  CALL PKFSCLRQUE("FStn ID"=R[130:Dp1x StnId])
   20:  CALL PKFSPUTQUE("FStn ID"=R[130:Dp1x StnId])
   21:  --   Reset tray cell counter
   22:  R[142:Dp1x TrayCtr]=1
   23:  LBL[1000]
   24:
   25:  LBL[10000: End]
[End]
```

7.10.2.12 PK_CV_PICK11.TP

This program is used by the group 1 robot to perform the first pick operation at a conveyor station of a conveyor. This program works as follows.

1 Sets the skip outbound FLG to OFF before beginning the pick moves.
2 Gets a part from the part queue using PKCSGETQUE.
3 Regardless of getting or not getting a part from the queue, if the property "Wait Position" is enabled in the *i*RPickTool treeview for this conveyor station, checks if the robot needs to wait at the Wait Position of the Conveyor station which is the Upstream boundary and calculates the wait position using PKCSCALWPOS.PC.
4 If the robot needs to wait because there is no part or part is upstream of the upstream boundary, moves to the wait position using PK_CV_WAITPOS1.TP.
5 If PKCSGETQUE did not return a part, it makes the request to get the part again.
6 If PKCSGETQUE returned a part,
7 Sets the UTOOL to the gripper's first zone's UTOOL.
8 Moves the robot to the approach position.
9 If the Skip Outbound FLG came on (implying that the robot cannot pick the part which just crossed the downstream boundary), then the part is returned to the queue with ACK=Skip (3) using the PKCSACKQUE macro and it goes back to the top of the program to request a new part to pick.
10 If the pick position is reachable, then it moves the robot to the pick position while executing the "Gripper Close" operation using Time Before (TB) condition. The gripper close operation is executed via the PKGRCLOSE.PC macro.
11 If the Skip Outbound FLG came on (implying that the robot cannot pick the part which just crossed the downstream boundary), then the part is returned to the queue with ACK=Skip (3) using the PKCSACKQUE macro and it goes back to the top of the program to request a new part to pick.

12 If the move to the pick position succeeded, then it gets the pick delay using the PKGRGETPKDLY.PC macro.
13 Checks if the required parts are picked using the PKGRCHKPP.PC macro if the property “Part presence Check” is enabled for this conveyor station operation in the treeview.
14 If the part presence check fails, it returns the part to the queue with ACK=Remove (4) and goes back to the top of the program to request a part to pick the missing part again.
15 If the part presence check succeeds, it sets the payload of the robot to “Full gripper” or Payload[2].
16 It moves the robot to the retreat position.
17 If the robot is to pick multiple parts individually, goes to the top of the program to request the next part to pick, otherwise the program ends.

PK_CV_PICK11.TP

```
 1:  -- Pick - Conveyor Seq1 G1
 2:  LBL[5]
 3:  F[R[102]]=(OFF)
 4:  UFRAME_NUM=0
 5:  UTOOL_NUM=1
 6:
 7:  LBL[10: Pick Loop]
 8:  IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
 9:  STOP_TRACKING
10:  -- Get part from que
11:  IF R[115:Pk1x UsePosChk]=0,JMP LBL[15]
12:  -- PKCSGETQINRG allocates only
13:  -- reachable parts (requires R846)
14:  CALL PKCSGETQINRG(CStn ID=R[110:Pk1x StnId],
      Consec Flag=R[107:Counter G1],Timeout (ms)=100,
      Do Pos Check=R[115:Pk1x UsePos],Ref Pos PR=57,Appr Ofst PR=56,
      Tool Offset=1,Offset VR=1,Stat Reg=123)
15:  JMP LBL[20]
16:
17:  LBL[15]
18:  -- Standard PKCSGETQUE
19:  CALL PKCSGETQUE(CStn ID=R[110:Pk1x StnId],Consec Flag=R[107:Counter G1],
      Timeout (ms)=100,Offset VR=1,Stat Reg=123)
20:
21:  LBL[20]
22:  //-- Is part pickable? PIR
23:
24:
25:  LBL[30]
26:  -- Caclulate WAIT pos
27:  CALL PKCSCALWPOS(CStn ID=R[110:Pk1x StnId],Ref Pos PR=57,Offset VR=1,
      GetQ stat=R[123:Pk1x GetQ stat], Do WaitMove Reg=95,Wait Pos PR=55,
      NoMove Distance=150)
28:  -- Move to wait position
29:  IF (R[95:TemporaryUse1]=0),
      CALL PK_CV_WAITPOS1(UTOOL Num=R[107:Counter G1],Appr Ofst PR=56,
      WaitPos PR=55,PkReachable Reg=115)
30:
31:  LBL[40:ChkGetQStat]
32:  IF ((R[199:Buf Id G1]>0) AND (R[123:Pk1x GetQ stat]>0)),JMP LBL[10000]
33:  IF R[123:Pk1x GetQ stat]<>0,JMP LBL[10]
34:
35:
```

```
   LBL[50]

   LBL[60]
   //-- Smart Pick


   LBL[70]
   //-- Check if a reject part


   LBL[80]

   -- Set Utool
   UTOOL_NUM=R[107:Counter G1]
   IF R[115:Pk1x UsePosChk]=0,JMP LBL[85]
   -- Check if tracking position is reachable
   CALL PKCSCHKPOS(CStn ID=R[110:Pk1x StnId],Offset VR=1,Ref Pos PR=57,
    Appr Ofst PR=56,Tool Offset=1,Stat Reg=108)
   IF R[108:PosNotReach G1]<>0,JMP LBL[1000]

   LBL[85]
   -- Move to approach
L PR[57:Pk1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[1]
    Tool_Offset,PR[56:Pk1 Cv Ap Ofst]
   -- Flag on when Skip outbound true
   IF (F[R[102]]),JMP LBL[1000]

   -- Set Pick/drop registers
   CALL PK_GR_GET_ENDZONE1(PickDropCounter=R[107:Counter ],
    Multiple=R[112:Pk1x PkMult],Quantity=R[111:Pk1x Qty],End Zone Reg=106)
   IF R[115:Pk1x UsePosChk]=0,JMP LBL[88]
   -- Check if tracking position is reachable
   CALL PKCSCHKPOS(CStn ID=R[110:Pk1x StnId],Offset VR=1,Ref Pos PR=57,
    Appr Ofst PR=0,Tool Offset=1,Stat Reg=108)
   IF R[108:PosNotReach G1]<>0,JMP LBL[1000]

   LBL[88]
   -- Move to pick pos
L PR[57:Pk1 Cv Ref Pos] max_speed CNT0 VOFFSET,VR[1] TB R[125]sec,
    CALL PKGRCLOSE(Gripper ID=R[104:Gripper Id G1],
    Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone G1])
   -- Chk Skip Pick
   IF (F[R[102]]),JMP LBL[1000]
   -- Pick delay
   CALL PKGRGETPKDLY(Gripper ID=R[104:Gripper Id G1],PickDelay Reg=113)
   WAIT R[113]
   -- Check part pres
   CALL PKCSGETPPCHK(CStn ID=R[110:Pk1x StnId],PartPresChk Reg=116)
   IF R[116:Pk1x UsePPChk]=0,JMP LBL[90]
   -- PKGRCHKPP grip_id, operation,
   -- stat_reg, StartZone,
   -- EndZone (opt)
   CALL PKGRCHKPP(Gripper ID=R[104:Gripper Id G1],Pick=0,Stat Reg=117,
    Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone G1])
   LBL[90]
```

```
   84:   PAYLOAD[2]
   85:   -- Move to retreat
   86:L PR[57:Pk1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[1]
         Tool_Offset,PR[56:Pk1 Cv Ap Ofst]
   87:   IF R[117:Pk1x PPStat]<>0,JMP LBL[1500]
   88:   JMP LBL[2000]
   89:
   90:
   91:   LBL[1000: Missed part]
   92:   F[R[102]]=(OFF)
   93:   -- Return part in VR1 to Queue
   94:   CALL PKCSACKQUE(CStn ID=R[110:Pk1x StnId],Skip=3)
   95:   IF R[199:Buf Id G1]>0,JMP LBL[10000]
   96:   JMP LBL[10]
   97:
   98:
   99:   LBL[1200: Return part]
  100:   F[R[102]]=(OFF)
  101:   -- Return part in VR1 to Queue
  102:   CALL PKCSACKQUE(CStn ID=R[110:Pk1x StnId],Return=2)
  103:   IF (R[199:Buf Id G1]>0),JMP LBL[10000]
  104:   JMP LBL[10]
  105:
  106:
  107:   LBL[1500: Dropped part]
  108:   F[R[102]]=(OFF)
  109:   -- Return part in VR1 to Queue
  110:   CALL PKCSACKQUE(CStn ID=R[110:Pk1x StnId],Remove=4)
  111:   R[117:Pk1x PPStat]=0
  112:   IF R[199:Buf Id G1]>0,JMP LBL[10000]
  113:   JMP LBL[10]
  114:
  115:
  116:   LBL[2000: Pick OK]
  117:   -- PKACKQ grp, vr, ack
  118:   CALL PKCSACKQUE(CStn ID=R[110:Pk1x StnId],Success=1)
  119:   -- Update the pick counter
  120:   IF R[112:Pk1x PkMult]=0,JMP LBL[2050]
  121:   R[107:Counter G1]=R[107:Counter G1]+R[111:Pk1x Qty]
  122:   JMP LBL[2100]
  123:
  124:   LBL[2050]
  125:   R[107:Counter G1]=R[107:Counter G1]+1
  126:   IF R[107:Counter G1]<=R[111:Pk1x Qty],JMP LBL[10]
  127:
  128:   LBL[2100]
  129:   R[205:Picked G1]=1
  130:   LBL[10000: End]
  131:   STOP_TRACKING
  132:   F[R[102]]=(OFF)
[End]
```

7.10.2.13 PK_CV_DROP11.TP

This program is used by the group 1 robot to perform the first drop or place operation at a conveyor station of a conveyor. This program works as follows.

1 Sets the skip outbound FLG to OFF before beginning the place moves.
2 Gets a part from the part queue using PKCSGETQUE.
3 Regardless of getting or not getting a part from the queue, if the property "Wait Position" is enabled in the *i*RPickTool treeview for this conveyor station, checks if the robot needs to wait at the Wait Position of the Conveyor station which is the Upstream boundary and calculates the wait position using PKCSCALWPOS.PC.
4 If the robot needs to wait because there is no part or the part is upstream of the upstream boundary, moves to the wait position using PK_CV_WAITPOS1.TP.
5 If PKCSGETQUE did not return a part, it makes the request to get the part again.
6 If PKCSGETQUE returned a part,
7 Sets the UTOOL to the gripper's first zone's UTOOL.
8 Moves the robot to the approach position.
9 If the Skip Outbound FLG came on (implying that the robot cannot place to the part which just crossed the downstream boundary), then the part is returned to the queue with ACK=Return (2) using the PKCSACKQUE macro and it goes back to the top of the program to request a new part to place into.
10 If the move to approach succeeded, then it moves the robot to the place position while executing the "Gripper Open" operation using Time Before (TB) condition. The gripper open operation is executed via the PKGROPEN.PC macro.
11 If the Skip Outbound FLG came on (implying that the robot cannot place into the part which just crossed the downstream boundary), then the part is returned to the queue with ACK=Return(2) using the PKCSACKQUE macro and it goes back to the top of the program to request a new part to pick.
12 If the move to the place position succeeded, then it gets the place delay using the PKGRGETDPDLY.PC macro.
13 Checks if the required parts are placed using the PKGRCHKPP.PC macro if the property "Part presence Check" is enabled for this conveyor station operation in the treeview.
14 If the part presence check fails, it returns the part to the queue with ACK=Remove(4) and goes back to the top of the program to request a part to place into again.
15 If the part presence check succeeds, it sets the payload of the robot to "Empty gripper" or Payload[1].
16 It moves the robot to the retreat position.
17 If the robot is to place multiple parts individually, goes to the top of the program to request the next part to place, otherwise the program ends.

PK_CV_DROP11.TP

```
 1:  -- Drop - Conv Seq1 G1
 2:  LBL[5]
 3:  F[R[102]]=(OFF)
 4:  UFRAME_NUM=0
 5:
 6:  LBL[10: Pick Loop]
 7:  IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
 8:  STOP_TRACKING
 9:  -- Get part from que
10:  IF R[135:Dp1x UsePosChk]=0,JMP LBL[15]
11:  -- PKCSGETQINRG allocates only
12:  -- reachable parts (requires R846)
13:  CALL PKCSGETQINRG(CStn ID=R[130:Dp1x StnId],
       Consec Flag=R[107:Counter G1],Timeout (ms)=100,
       Do Pos Check=R[135:Dp1x UsePos],Ref Pos PR=59,Appr Ofst PR=58,
       Tool Offset=1,Offset VR=2,Stat Reg=143)
14:  JMP LBL[20]
```

```
 15:
 16:  LBL[15]
 17:  -- Standard PKCSGETQUE
 18:  CALL PKCSGETQUE(CStn ID=R[130:Dp1x StnId],Consec Flag=R[107:Counter G1],
      Timeout (ms)=100,Offset VR=2,Stat Reg=143)
 19:
 20:  LBL[20]
 21:  //-- Is part droppable?   PIR
 22:
 23:
 24:  LBL[30]
 25:  -- Caclulate WAIT pos
 26:  CALL PKCSCALWPOS(CStn ID=R[130:Dp1x StnId],Ref Pos PR=59,
      Offset VR=2,GetQ stat=R[143:Dp1x GetQ stat],Do WaitMove Reg=95,
      Wait Pos PR=55,NoMove Distance=150)
 27:  -- Move to wait position
 28:  IF (R[95:TemporaryUse1]=0),
      CALL PK_CV_WAITPOS1(UTOOL Num=R[107:Counter G1],Appr Ofst PR=58,
      WaitPos PR=55,PkReachable Reg=135)
 29:
 30:  LBL[40:ChkGetQStat]
 31:  IF ((R[199:Buf Id G1]>0) AND (R[143:Dp1x GetQ stat]>0)),JMP LBL[10000]
 32:  IF R[143:Dp1x GetQ stat]<>0,JMP LBL[10]
 33:
 34:
 35:  LBL[50]
 36:
 37:  LBL[60]
 38:  -- Smart Drop
 39:
 40:
 41:  LBL[70]
 42:  -- Reject part
 43:
 44:
 45:  LBL[80]
 46:
 47:  -- Set Utool
 48:  UTOOL_NUM=R[107:Counter G1]
 49:  IF R[135:Dp1x UsePosChk]=0,JMP LBL[85]
 50:  -- Check if tracking position is reachable
 51:  CALL PKCSCHKPOS(CStn ID=R[130:Dp1x StnId],Offset VR=2,Ref Pos PR=59,
      Appr Ofst PR=58,Tool Offset=1,Stat Reg=108)
 52:  IF R[108:PosNotReach G1]<>0,JMP LBL[1000]
 53:
 54:  LBL[85]
 55:  -- Move to approach
 56:L PR[59:Dp1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[2]
      Tool_Offset,PR[58:Dp1 Cv Ap Ofst]
 57:  -- Flag on when Skip outbound true
 58:  IF (F[R[102]]),JMP LBL[1000]
 59:
 60:  -- Set Pick/drop registers
 61:  CALL PK_GR_GET_ENDZONE1(PickDropCounter=R[107:Counter ],
      Multiple=R[132:Dp1x DpMult],Quantity=R[131:Dp1x Qty],End Zone Reg=106)
```

```
  62:   IF R[135:Dp1x UsePosChk]=0,JMP LBL[88]
  63:   -- Check if tracking position is reachable
  64:   CALL PKCSCHKPOS(CStn ID=R[130:Dp1x StnId],Offset VR=2,Ref Pos PR=59,
        Appr Ofst PR=0,Tool Offset=1,Stat Reg=108)
  65:   IF R[108:PosNotReach G1]<>0,JMP LBL[1000]
  66:
  67:   LBL[88]
  68:   -- Move to drop pos
  69:L PR[59:Dp1 Cv Ref Pos] max_speed CNT0 VOFFSET,VR[2] TB R[145]sec,
        CALL PKGROPEN(Gripper ID=R[104:Gripper Id G1],
        Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone G1])
  70:   -- Chk Skip Drop
  71:   IF (F[R[102]]),JMP LBL[1000]
  72:   -- Drop delay
  73:   CALL PKGRGETDPDLY(Gripper ID=R[104:Gripper Id G1],DropDelay Reg=133)
  74:   WAIT R[133]
  75:   -- Check part pres
  76:   CALL PKCSGETPPCHK(CStn ID=R[130:Dp1x StnId],PartPresChk Reg=136)
  77:   IF R[136:Dp1x UsePPChk]=0,JMP LBL[90]
  78:   -- PKGRCHKPP grip_id, operation,
  79:   -- stat_reg, StartZone,
  80:   -- EndZone (opt)
  81:   CALL PKGRCHKPP(Gripper ID=R[104:Gripper Id G1],Place=1,Stat Reg=137,
        Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone G1])
  82:   LBL[90]
  83:   PAYLOAD[1]
  84:   -- Move to retreat
  85:L PR[59:Dp1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[2]
        Tool_Offset,PR[58:Dp1 Cv Ap Ofst]
  86:   IF R[137:Dp1x PPStat]<>0,JMP LBL[1500]
  87:   JMP LBL[2000]
  88:
  89:
  90:   LBL[1000: Missed part]
  91:   F[R[102]]=(OFF)
  92:   -- Return part in VR1 to Queue
  93:   CALL PKCSACKQUE(CStn ID=R[130:Dp1x StnId],Skip=3)
  94:   IF R[199:Buf Id G1]>0,JMP LBL[10000]
  95:   JMP LBL[10]
  96:
  97:   LBL[1200: Return part]
  98:   F[R[102]]=(OFF)
  99:   -- Return part in VR to Queue
 100:   CALL PKCSACKQUE(CStn ID=R[130:Dp1x StnId],Return=2)
 101:   IF R[199:Buf Id G1]>0,JMP LBL[10000]
 102:   JMP LBL[10]
 103:
 104:
 105:   LBL[1500: Dropped part]
 106:   F[R[102]]=(OFF)
 107:   -- Return part in VR1 to Queue
 108:   CALL PKCSACKQUE(CStn ID=R[130:Dp1x StnId],Remove=4)
 109:   R[137:Dp1x PPStat]=0
 110:   IF R[199:Buf Id G1]>0,JMP LBL[10000]
 111:   JMP LBL[10]
```

```
  112:
  113:
  114:   LBL[2000: Drop OK]
  115:
  116:   -- PKACKQ grp, vr, ack
  117:   CALL PKCSACKQUE(CStn ID=R[130:Dp1x StnId],Success=1)
  118:   -- Update the pick counter
  119:   IF R[132:Dp1x DpMult]=0,JMP LBL[2050]
  120:   R[107:Counter G1]=R[107:Counter G1]+R[131:Dp1x Qty]
  121:   JMP LBL[2100]
  122:
  123:   LBL[2050]
  124:   R[107:Counter G1]=R[107:Counter G1]+1
  125:   IF R[107:Counter G1]<=R[131:Dp1x Qty],JMP LBL[10]
  126:
  127:   LBL[2100]
  128:   R[205:Picked G1]=0
  129:   LBL[10000: End]
  130:   STOP_TRACKING
  131:   F[R[102]]=(OFF)
[End]
```


7.10.2.14 PK_CV_WAITPOS1.TP

This program is used by the group 1 robot to move to a wait position at the Upstream boundary of a conveyor station of a conveyor when there are no parts to pick.

This program is called by four arguments.
Argument 1: UTOOL_NUM which is usually the same as the pick or place counter.
Argument 2: Position Register number of the approach offset of the conveyor station
Argument 3: Position Register number used for the move to wait position.
Argument 4: Numeric Register number that contains 0.

This program is used to move the robot to the wait position at the “pick” conveyor station or the “place” conveyor station by passing in different arguments.

PK_CV_WAITPOS1.TP

```
    1:   -- PK_CV_WAITPOS1
    2:   -- Wait for parts at this position
    3:   -- at the conveyor
    4:   -- AR[1] is the Utool_num to use
    5:   -- AR[2] is position register for appr offset
    6:   -- AR[3] is position register for move
    7:   -- AR[4] is register for pick-only-reachable-parts
    8:
    9:   UTOOL_NUM=AR[1]
   10:   IF R[AR[4]]=0,JMP LBL[85]
   11:   PR[51:TemporaryUse1]=PR[AR[2]]
   12:   PR[51,3:TemporaryUse1]=PR[51,3:TemporaryUse1]*(-1)
   13:   -- Check if fixed position is reachable
   14:   CALL PKOPPOSCHK(Position PR=AR[3],Group Num=1,Uframe Num=0,
         Tool Frame Num=AR[1],Stat Reg=108,Offset PR=51)
   15:   IF R[108:PosNotReach G1]<>0,JMP LBL[100]
   16:   LBL[85]
   17:   -- Dynamic Pk Wait pos
```

```
   18:L PR[AR[3]] max_speed CNT100 Tool_Offset,PR[AR[2]]
   19:   LBL[100]
[End]
```

7.10.2.15 PK_FS_PICK11.TP

This program is used by the group 1 robot to perform the first pick operation at a Fixed station. This program works as follows.

1 Gets the UFRAME number of the Fixed Station that you set in the Fixed Station menu of the *i*RPickTool treeview using the PKFSGETUF.PC macro and sets the system's UFRAME_NUM to it.
2 Checks if the Fixed Station is Ready for pick using the TP macro PK_FS_READY.MR macro.
3 If the Fixed Station is not ready, then repeatedly re-executes the PK_FS_READY macro until the Station is Ready. Station is ready means that the Fixed Station Ready DI is high, the Index Out DO is low and the internal flag for station_ready is TRUE. These are interlocks needed for the robot to safely operate in the fixed station without collisions.
4 Gets a part from the part queue using PKFSGETQUE.
5 If PKFSGETQUE did not return a part, and its status is non-zero, it means that all parts in the tray have already been allocated. Therefore, it executes the Indexing of the Fixed Station via PK_FS_INDEX macro. After indexing the Fixed station, it re-initializes the tray on the Fixed Station using the PKFSPUTQUE macro.
6 If PKFSGETQUE returned a part,
7 Sets the UTOOL to the gripper's first zone's UTOOL.
8 Moves the robot to the approach position.
9 If the move to approach succeeded, then it moves the robot to the pick position while executing the "Gripper Close" operation using Time Before (TB) condition. The gripper close operation is executed via the PKGRCLOSE.PC macro.
10 If the move to the place position succeeded, then it gets the place delay using the PKGRGETDPDLY.PC macro.
11 Checks if the required parts have been picked using the PKGRCHKPP.PC macro if the property "Part presence Check" is enabled for this conveyor station in the treeview.
12 If the part presence check fails, it returns the part to the queue with ACK=Return (2) and goes back to the top of the program to request a part to place into again. (It would get the same part again).
13 If the part presence check succeeds, it sets the payload to "full gripper" payload which is Payload[2].
14 It moves the robot to the retreat position.
15 If the robot is to pick multiple parts individually, goes to the top of the program to request the next part to pick, otherwise the program ends.

PK_FS_PICK11.TP

```
   1:  -- Pick - FStn1 G1 Seq: 1
   2:
   3:  R[107:Counter G1]=1
   4:
   5:  LBL[5]
   6:  IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
   7:
   8:  -- Get FStn Tray Uframe
   9:  -- PKFSGETUF(fs_id, reg_uf)
  10:  CALL PKFSGETUF(FStn ID=R[110:Pk1x StnId],UFrameNum Reg=95)
  11:  UFRAME_NUM=R[95:TemporaryUse1]
  12:
  13:  LBL[10]
  14:  -- Check if FS is ready
  15:  -- PK_FS_READY(fs_id, stat)
  16:  CALL PK_FS_READY(FStn ID=R[110:Pk1x StnId],Stat Reg=95)
```

```
 17:   IF R[95:TemporaryUse1]=0,JMP LBL[20]
 18:   IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
 19:   WAIT       .02(sec)
 20:   JMP LBL[10]
 21:
 22:   LBL[20: Pick Loop]
 23:   IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
 24:
 25:   -- Get a cell from the tray
 26:   -- PKFSGETQUE(fs_id, cell_data_vr, stat_reg, model_id (opt))
 27:   CALL PKFSGETQUE(FStn ID=R[110:Pk1x StnId],Offset VR=1,Stat Reg=123)
 28:   IF R[123:Pk1x GetQ stat]=0,JMP LBL[200]
 29:   !Index tray
 30:   CALL PK_FS_INDEX(FStn ID=R[110:Pk1x StnId],CellCounter Reg=122,Stat Reg=95)
 31:   JMP LBL[5]
 32:
 33:   LBL[200]
 34:
 35:
 36:   LBL[300]
 37:   UTOOL_NUM=R[107:Counter G1]
 38:   -- Set Pick/drop registers
 39:   CALL PK_GR_GET_ENDZONE1(PickDropCounter=R[107:Counter ],
       Multiple=R[112:Pk1x PkMult],Quantity=R[111:Pk1x Qty],End Zone Reg=106)
 40:   -- Move to approach
 41:L PR[61:Pk1 FS Ref Pos] max_speed CNT100 VOFFSET,
       VR[1] Tool_Offset,PR[60:Pk1 FS Ap Ofst]
 42:   -- Move to pick pos
 43:L PR[61:Pk1 FS Ref Pos] max_speed CNT0 VOFFSET,VR[1] TB R[125]sec,
       CALL PKGRCLOSE(Gripper ID=R[104:Gripper Id G1],
       Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone G1])
 44:   -- Pick delay
 45:   CALL PKGRGETPKDLY(Gripper ID=R[104:Gripper Id G1],PickDelay Reg=113)
 46:   WAIT R[113]
 47:   -- Check part pres
 48:   CALL PKFSGETPPCHK(FStn ID=R[110:Pk1x StnId],PartPresChk Reg=116)
 49:   IF R[116:Pk1x UsePPChk]=0,JMP LBL[400]
 50:   -- PKGRCHKPP grip_id, operation,
 51:   -- stat_reg, StartZone,
 52:   -- EndZone (opt)
 53:   CALL PKGRCHKPP(Gripper ID=R[104:Gripper Id G1],Pick=0,Stat Reg=117,
       Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone G1])
 54:   LBL[400]
 55:   -- Full gripper payload
 56:   PAYLOAD[2]
 57:   -- Move to retreat
 58:L PR[61:Pk1 FS Ref Pos] max_speed CNT100 VOFFSET,
       VR[1] Tool_Offset,PR[60:Pk1 FS Ap Ofst]
 59:   IF R[117:Pk1x PPStat]<>0,JMP LBL[1500]
 60:   JMP LBL[2000]
 61:
 62:
 63:   LBL[1500: PartGotdropped]
 64:   -- Return part in VR1 to Queue
 65:   CALL PKFSACKQUE(FStn ID=R[110:Pk1x StnId],Return=2)
```

```
   66:   R[117:Pk1x PPStat]=0
   67:   JMP LBL[20]
   68:
   69:
   70:   LBL[2000: Pick ok]
   71:   -- Ack the part has been picked
   72:   CALL PKFSACKQUE(FStn ID=R[110:Pk1x StnId],Success=1)
   73:   -- Update tray cell count
   74:   R[122:Pk1x TrayCtr]=R[122:Pk1x TrayCtr]+1
   75:   -- Update pick counter
   76:   IF R[112:Pk1x PkMult]=0,JMP LBL[2050]
   77:   R[107:Counter G1]=R[107:Counter G1]+R[111:Pk1x Qty]
   78:   JMP LBL[3000]
   79:
   80:   LBL[2050]
   81:   R[107:Counter G1]=R[107:Counter G1]+1
   82:   IF (R[107:Counter G1]<=R[111:Pk1x Qty]),JMP LBL[20]
   83:
   84:   LBL[3000: CheckIndex]
   85:   !Check if tray ready to be indexd
   86:   CALL PKFSGETQUE(FStn ID=R[110:Pk1x StnId],Offset VR=1,Stat Reg=123)
   87:   IF R[123:Pk1x GetQ stat]=0,JMP LBL[3500]
   88:   !Index tray
   89:   CALL PK_FS_INDEX(FStn ID=R[110:Pk1x StnId],CellCounter Reg=122,Stat Reg=95)
   90:   JMP LBL[10000]
   91:   LBL[3500]
   92:   -- Return part for getting back in next cycle
   93:   CALL PKFSACKQUE(FStn ID=R[110:Pk1x StnId],Return=2)
   94:
   95:
   96:   LBL[10000: End]
[End]
```

7.10.2.16 PK_FS_DROP11.TP

This program is used by the group 1 robot to perform the first drop or place operation at a Fixed station. This program

1 Gets the UFRAME number of the Fixed Station that you set in the Fixed Station menu of the *i*RPickTool treeview using the PKFSGETUF.PC macro and sets the system's UFRAME_NUM to it.
2 Checks if the Fixed Station is Ready for pick using the TP macro PK_FS_READY.MR macro.
3 If the Fixed Station is not ready, then repeatedly re-executes the PK_FS_READY macro until the Station is Ready. Station is ready means that the Fixed Station Ready DI is high, the Index Out DO is low and the internal flag for station_ready is TRUE. These are interlocks needed for the robot to safely operate in the fixed station without collisions.
4 Gets a part from the part queue using PKFSGETQUE.
5 If PKFSGETQUE did not return a part, and its status is non-zero, it means that all parts in the tray have already been allocated. Therefore, it executes the Indexing of the Fixed Station via PK_FS_INDEX macro. After indexing the Fixed station, it re-initializes the tray on the Fixed Station using the PKFSPUTQUE macro.
6 If PKFSGETQUE returned a part,
7 Sets the UTOOL to the gripper's first zone's UTOOL.
8 Moves the robot to the approach position.
9 If the move to approach succeeded, then it moves the robot to the pick position while executing the "Gripper Open" operation using Time Before (TB) condition. The gripper open operation is executed via the PKGROPEN.PC macro.

10 If the move to the place position succeeded, then it gets the place delay using the PKGRGETDPDLY.PC macro.
11 Checks if the required parts have been placed using the PKGRCHKPP.PC macro if the property “Part presence Check” is enabled for this conveyor station in the treeview.
12 If the part presence check fails, it returns the part to the queue with ACK=Return (2) and goes back to the top of the program to request a part to place into again. (It would get the same part again).
13 If the part presence check succeeds, it sets the payload to “empty gripper” payload which is Payload[1].
14 It moves the robot to the retreat position.
15 If the robot is to place multiple parts individually, goes to the top of the program to request the next part to place, otherwise the program ends.

PK_FS_DROP11.TP

```
 1:  -- Drop - FStn1 G1 Seq: 1
 2:
 3:  R[107:Counter G1]=1
 4:
 5:  LBL[5]
 6:  IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
 7:
 8:  -- Get FStn Tray Uframe
 9:  -- PKFSGETUF(fs_id, reg_uf)
10:  CALL PKFSGETUF(FStn ID=R[130:Dp1x StnId],UFrameNum Reg=95)
11:  UFRAME_NUM=R[95:TemporaryUse1]
12:
13:  LBL[10]
14:  -- Check if FS is ready
15:  -- PK_FS_READY(fs_id, stat)
16:  CALL PK_FS_READY(FStn ID=R[130:Dp1x StnId],Stat Reg=95)
17:  IF R[95:TemporaryUse1]=0,JMP LBL[20]
18:  IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
19:  WAIT      .05(sec)
20:  JMP LBL[10]
21:
22:  LBL[20: Drop Loop]
23:  IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
24:
25:  -- Get a cell from the tray
26:  -- PKFSGETQUE(fs_id, cell_data_vr, stat_reg, model_id (opt))
27:  CALL PKFSGETQUE(FStn ID=R[130:Dp1x StnId],Offset VR=2,Stat Reg=143)
28:  IF R[143:Dp1x GetQ stat]=0,JMP LBL[200]
29:  !Index tray
30:  CALL PK_FS_INDEX(FStn ID=R[130:Dp1x StnId],CellCounter Reg=142,
       Stat Reg=95)
31:  JMP LBL[5]
32:
33:  LBL[200]
34:
35:
36:  LBL[300]
37:  UTOOL_NUM=R[107:Counter G1]
38:  -- Set Pick/drop registers
39:  CALL PK_GR_GET_ENDZONE1(PickDropCounter=R[107:Counter ],
       Multiple=R[132:Dp1x DpMult],Quantity=R[131:Dp1x Qty],End Zone Reg=106)
40:  -- Move to approach
41:L PR[63:Dp1 FS Ref Pos] max_speed CNT100 VOFFSET,
```

```
      VR[2] Tool_Offset,PR[62:Dp1 FS Ap Ofst]
42:   -- Move to drop pos
43:L PR[63:Dp1 FS Ref Pos] max_speed CNT0 VOFFSET,VR[2] TB R[145]sec,
      CALL PKGROPEN(Gripper ID=R[104:Gripper Id G1],
      Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone G1])
44:   -- Drop delay
45:   CALL PKGRGETDPDLY(Gripper ID=R[104:Gripper Id G1],DropDelay Reg=133)
46:   WAIT R[133]
47:   -- Check part pres
48:   CALL PKFSGETPPCHK(FStn ID=R[130:Dp1x StnId],PartPresChk Reg=136)
49:   IF R[136:Dp1x UsePPChk]=0,JMP LBL[400]
50:   -- PKGRCHKPP grip_id, operation,
51:   -- stat_reg, StartZone,
52:   -- EndZone (opt)
53:   CALL PKGRCHKPP(Gripper ID=R[104:Gripper Id G1],Place=1,Stat Reg=137,
      Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone G1])
54:   LBL[400]
55:   -- Empty gripper payload
56:   PAYLOAD[1]
57:   -- Move to retreat
58:L PR[63:Dp1 FS Ref Pos] max_speed CNT100 VOFFSET,
      VR[2] Tool_Offset,PR[62:Dp1 FS Ap Ofst]
59:   IF R[137:Dp1x PPStat]<>0,JMP LBL[1500]
60:   JMP LBL[2000]
61:
62:
63:   LBL[1500: PartGotdropped]
64:   -- Return part in VR1 to Queue
65:   CALL PKFSACKQUE(FStn ID=R[130:Dp1x StnId],Return=2)
66:   R[137:Dp1x PPStat]=0
67:   JMP LBL[20]
68:
69:
70:   LBL[2000: Pick ok]
71:   -- Ack the part has been picked
72:   CALL PKFSACKQUE(FStn ID=R[130:Dp1x StnId],Success=1)
73:   -- Update tray cell count
74:   R[142:Dp1x TrayCtr]=R[142:Dp1x TrayCtr]+1
75:   -- Update pick counter
76:   IF R[132:Dp1x DpMult]=0,JMP LBL[2050]
77:   R[107:Counter G1]=R[107:Counter G1]+R[131:Dp1x Qty]
78:   JMP LBL[3000]
79:
80:   LBL[2050]
81:   R[107:Counter G1]=R[107:Counter G1]+1
82:   IF (R[107:Counter G1]<=R[131:Dp1x Qty]),JMP LBL[20]
83:
84:   LBL[3000: CheckIndex]
85:   !Check if tray ready to be indexd
86:   CALL PKFSGETQUE(FStn ID=R[130:Dp1x StnId],Offset VR=2,Stat Reg=143)
87:   IF R[143:Dp1x GetQ stat]=0,JMP LBL[3500]
88:   !Index tray
89:   CALL PK_FS_INDEX(FStn ID=R[130:Dp1x StnId],CellCounter Reg=142,
      Stat Reg=95)
90:   JMP LBL[10000]
```

```
   91:   LBL[3500]
   92:   -- Return part for getting back in next cycle
   93:   CALL PKFSACKQUE(FStn ID=R[130:Dp1x StnId],Return=2)
   94:
   95:
   96:   LBL[10000: End]
[End]
```

7.10.2.17 PK_FS_INDEX.TP

This program is used to index a Fixed Station.

This program is called by three arguments.
Argument 1: Fixed Station ID.
Argument 2: Tray cell counter register.
Argument 3: Status register. If the value of status in this register is non-zero, then an error occurred during the indexing.

This program
1 Determines the Index Out DO for the Fixed Station based on the operation parameter.
2 Turns on the Index Out DO.
3 Starts the monitor to watch for the Station Ready DI to go low and then come back high via the PKFSMONIDX macro.

PK_FS_INDEX.TP

```
    1:   -- PK_FS_INDEX
    2:   -- Index the FStn
    3:   -- Arg 1: fs_id
    4:   -- Arg 2: FS Tray Counter Reg
    5:   -- Arg 3: stat_reg
    6:
    7:   R[AR[3]]=0
    8:   R[96:TemporaryUse2]=0
    9:
   10:   -- Get Index DO of FStn
   11:   CALL PKFSGETIXODO(FStn ID=AR[1],IndexOutDO Reg=96)
   12:
   13:   -- If valid Idx DO
   14:   IF R[96:TemporaryUse2]=0,JMP LBL[1500]
   15:
   16:
   17:   LBL[1000: Do_Idx]
   18:   -- TurnIdx Output high
   19:   DO[R[96]]=ON
   20:
   21:
   22:   -- Now monitor stnRdy DI going
   23:   -- low and then coming back high
   24:   -- (until then stn is not clear )
   25:   -- PKFSMONIDX(fs_id, stat_reg)
   26:   CALL PKFSMONIDX(FStn ID=AR[1],Stat Reg=AR[3])
   27:   LBL[1500]
   28:   -- Prepare new tray
   29:   CALL PKFSPUTQUE(FStn ID=AR[1])
   30:   ! Reset tray counter
```

```
   31:   R[AR[2]]=1
   32:   JMP LBL[10000]
   33:
   34:   LBL[2000: Not ok]
   35:   R[AR[3]]=1
   36:
   37:   LBL[10000: End]
[End]
```

7.10.2.18 PK_FS_READY.TP

This program is used to check if a Fixed Station is ready for pick or place by a robot.

This program is called by two arguments.
Argument 1: Fixed Station ID.
Argument 2: "Ready" status register. If the value of status in this register is non-zero, then the station is not ready. A station is not ready if its Index Out DO is high, its internal ready flag is FALSE or if the Station Ready DI is low.

This program
1 Determines the Index Out DO for the Fixed Station based on the operation parameter.
2 If the Index Out DO is high, then return "not ready" in the status register argument and return.
3 Then determine if the internal station ready flag is FALSE. If FALSE, then return "not ready" in the status register and return. The internal flag will be FALSE if after an Index, the station Ready DI did not go low and come back high again.
4 Then determine if the station ready DI is high. If not high, then return "Not Ready" in the status register argument and return.
5 If all the station ready conditions are satisfied, then return 0 for success in the status register.

PK_FS_READY.TP

```
   1:  -- PK_FS_READY
   2:  -- checks if the FStn is ready
   3:  --   for picking or dropping
   4:  --   Arg 1: fs_id
   5:  --   Arg 2: ready_reg
   6:
   7:  R[96:TemporaryUse2]=0
   8:  R[AR[2]]=0
   9:
  10:  -- Get Index DO of FStn
  11:  CALL PKFSGETIXODO(FStn ID=AR[1],IndexOutDO Reg=96)
  12:
  13:  -- If Idx Output is high, not ready
  14:  IF R[96:TemporaryUse2]<=0,JMP LBL[100]
  15:  IF DO[R[96]]=ON,JMP LBL[1000]
  16:
  17:  LBL[100]
  18:  -- If stn_clr_reg<>0, not ready
  19:  -- Get FS ready status
  20:  -- PKFSGETRDY(fs_id, fs_rdy_reg)
  21:  CALL PKFSGETRDY(FStn ID=AR[1],FStnReady Reg=97)
  22:  IF R[97:TemporaryUse3]<>1,JMP LBL[1000]
  23:
  24:  -- Get Ready DI
  25:  CALL PKFSGETRDYDI(FStn ID=AR[1],ReadyDI Reg=96)
```

```
   26:
   27:   -- If rdyDI is high, ready
   28:   IF R[96:TemporaryUse2]<=0,JMP LBL[2000]
   29:   IF DI[R[96]]=ON,JMP LBL[2000]
   30:   JMP LBL[1000]
   31:
   32:   LBL[1000: NotReady]
   33:   R[AR[2]]=1
   34:   JMP LBL[10000]
   35:
   36:   LBL[2000: Ready]
   37:   R[AR[2]]=0
   38:   JMP LBL[10000]
   39:
   40:   LBL[10000: End]
[End]
```


7.10.2.19 PK_GR_GET_ENDZONE1.TP

This program calculates the last (or ending) zone of a gripper used by the group 1 robot based on the start zone, pick_or_drop_multiple flag and the quantity for pick or place. It is useful when you have multiple picks or drops.

This program is called by four arguments.
Argument 1: Pick or drop counter
Argument 2: “Pick or drop multiple” property for the Conveyor Station or the Fixed Station in which the robot is operating.
Argument 3: The number of zones that have been added to the gripper
Argument 4: The end_zone register in which the value of the end zone is returned.

PK_GR_GET_ENDZONE1.TP

```
    1:   -- PK_GR_GET_ENDZONE1 G1
    2:   -- Calculate Gripper end
    3:   --     zone based on single
    4:   -- or multi-pick
    5:   -- AR[1]: pick or drop counter
    6:   -- AR[2]: pick or drop multiple
    7:   -- AR[3]: pick or drop qty
    8:   -- AR[4]: end zone register
    9:
   10:   IF (AR[2]=0),JMP LBL[100]
   11:   R[AR[4]]=AR[1]+AR[3]-1
   12:   JMP LBL[1000]
   13:
   14:   LBL[100]
   15:   R[AR[4]]=AR[1]
   16:   LBL[1000]
   17:
[End]
```

7.10.2.20 PK_CYCSTOP1.TP

This program contains the logic for processing the cycle stop. You should insert instructions here based on the needs of the application. In the shipped version, it only PAUSES the application task.

PK_CYCSTOP1.TP

```
    1:  -- PK Cycle Stop G1
    2:  -- Users should replace contents
    3:  PAUSE
[End]
```

7.11 SETUP OF PRE-GROUPING

This section describes the procedure for setting up functions that arranges parts on the same conveyor (pre-grouping function) in connection with the Plug & Play gripper and operation.
The setup of the pre-grouping function in relation to other objects is the same as the setup for standard functions. Refer to Section 6.14, “SETUP OF PRE-GROUPING”.”

For settings specific to North America, in order to use the pre-grouping function, you will need the option R856: iRPickTool / Pre-grouping for North America.

7.11.1 Setup of a Gripper and a Zone

Regarding the robot that performs arrangement on the same conveyor, there are no special steps for pre-grouping. Refer to 7.2 “SETUP OF A GRIPPER AND A ZONE”.

Regarding the robot that picks all the arranged parts at one shot, you should give attention to zone settings. You should add a zone per the arranged parts picked at one shot instead of adding a zone per a part. For example, in Fig. 6.14 (c), add a zone per three parts.

7.11.2 Setup of a Robot Operation

Regarding the robot that performs arrangement on the same conveyor, there are no special steps for pre-grouping. Refer to 7.3 “SETUP OF A ROBOT OPERATION”.

Regarding the robot that picks all the arranged parts at one shot, you should give attention to conveyor station operation settings and fixed station operation settings.

In conveyor station operations for picking operations, parts that have been put in lines are picked together in one batch, but [Pick or Drop Multiple] is disabled.

In conveyor station operations or fixed station operations for placing operations, [Pick or Drop Multiple] is set up as follows.
In order to place all the parts in one go, enable [Pick or Drop Multiple].
If you will perform a placing operation after performing a picking operation several times, disable [Pick or Drop Multiple] if you will also perform a placing operation in units of parts that have been picked together in one batch.

7.11.3 Fine Adjustment of The Tracking Motion

Refer to 6.12 “FINE ADJUSTMENT OF THE TRACKING MOTION”. As special affairs in the case of a pre-grouping, care needs to be exercised when adjusting the tracking frame error.

7.11.3.1 Adjustment of the tracking frame error (When you use ADJ_OFS)

When using ADJ_OFS, modify the program which performs arrangement motion shown in “Sample programs for the conveyor-to-the same conveyor configuration (the first robot)” as follows. In the sample program shown below, Position Register 20 is used to store the amount of adjustment. Change it as necessary.

```
  40:  LBL[80: PickPart]
  41:  CALL ADJ_OFS(1,1,20,1)
  42:  ! Move to approach
  43:L PR[57:Pk1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[1] Tool_Offset,PR[56:Pk1
       Cv Ap Ofst]
```

```
  64:  LBL[200: PlacePart]
  65:  CALL ADJ_OFS(1,2,20,2)
  66:  ! Move to approach
  67:L PR[57:Pk1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[2] Tool_Offset,PR[58:Dp1
       Cv Ap Ofst]
```

- Just before the pickup motion and the placement motion, ADJ_OFS is called for adjusting the vision offset.
- The same amount of adjustment stored in the same Position Register is used for both the pickup motion and the placement position.
- When setting up the amount of adjustment, insert a motion instruction and a wait instruction immediately after the approach point as follows so as to make the robot stop immediately above the pickup position,.

```
  45:  ! Move to pick pos
  46:L PR[57:Pk1 Cv Ref Pos] max_speed CNT0 VOFFSET,VR[1] Tool_Offset,PR[15]
  47:  WAIT 10.00(sec)
  48:L PR[57:Pk1 Cv Ref Pos] max_speed CNT0 VOFFSET,VR[1] TB R[125]sec,CALL
PKGRCLOSE(Gripper ID=R[104:Gripper Id G1],Start Zone=1,End Zone=1)
```

- Refer to 7.8.1 “Adjustment of the Tracking Frame Error” for details.

7.11.4 Standard TP programs

This section describes the standard TP programs of pre-grouping.

They works depending on the type of pre-grouping defined as follows.

1: Pre-grouping is used and this is the first robot that performs arrangement on the same conveyor. Virtual trays are generated by this robot (Robot 1).
2: Pre-grouping is used and this is the second or any successive robot that performs arrangement on the same conveyor (Robot 2).
3: Pre-grouping is used and this is the robot that picks all the arranged parts at one shot (Robot 3).

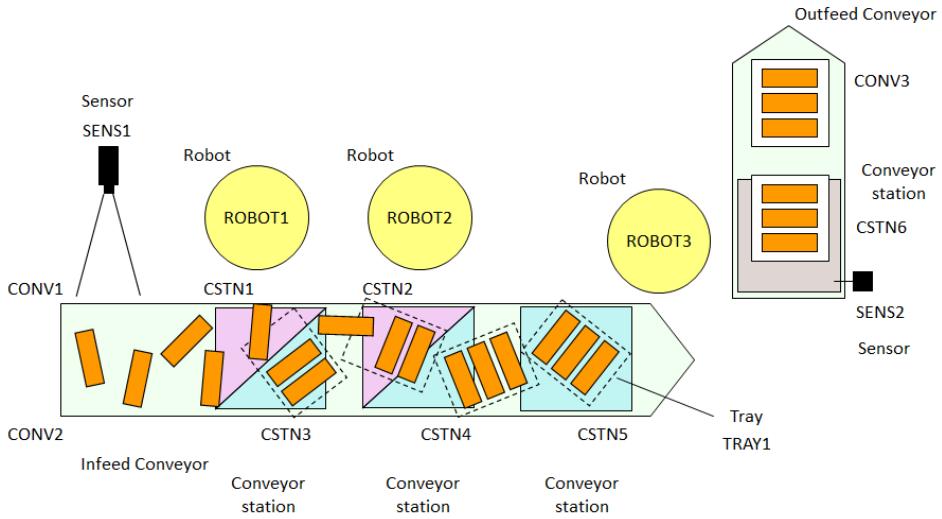


Fig. 7.11.4 (a) System configuration

7.11.4.1 PK_PRGR_MAIN1.TP

This program is the main program of pre-grouping for the group 1 robot.

When the type of pre-grouping is 1, this program generates a virtual tray and calls PK_PRGR_CV_PICKDROP1.TP indefinitely. Before generating a virtual tray, always check if cycle stop is invoked. If a virtual tray cannot be generated because of interference, after PKCSGETQUE returns a part, the part is returned to the queue using ACK=Skip (3) and then this program tries to generate another virtual tray. If a virtual tray cannot be generated because of other reasons, this program tries to generate a virtual tray again after waiting 0.02s.

When the type of pre-grouping is 2, this program calls PK_PRGR_CV_PICKDROP1.TP indefinitely. Before calling PK_PRGR_CV_PICKDROP1.TP, always check if cycle stop is invoked.

PK_PRGR_MAIN1.TP

```
1:  ! Main Prog for Grp:1
2:  ! for Pre-grouping
3:
4:  IF R[230:PRGR Robot Type]<=0 OR R[230:PRGR Robot Type]>2,JMP LBL[10000]
5:
6:  LBL[100:pk n pl]
7:  IF R[101:Cycle Stop G1]>0,JMP LBL[10000]
8:  ! Generate vtray only in the
9:  ! sensor controller.
```

```
 10:  IF R[230:PRGR Robot Type]<>1,JMP LBL[200]
 11:  ! Generate Virtual tray
 12:  ! PKCSGENVRTRY(cstn_id, stat_reg)
 13:  CALL PKCSGENVRTRY(CStn ID=R[130:Dp1x StnId],Stat Reg=95)
 14:  IF R[95:TemporaryUse1]=0,JMP LBL[200]
 15:  IF R[95:TemporaryUse1]=1,JMP LBL[300]
 16:  IF R[95:TemporaryUse1]=2,JMP LBL[400]
 17:  WAIT      .02(sec)
 18:  JMP LBL[100]
 19:
 20:  LBL[200: Pre-group]
 21:  ! Do pre-grouping
 22:  CALL PK_PRGR_CV_PICKDROP1
 23:  IF R[230:PRGR Robot Type]<>1,JMP LBL[200]
 24:  JMP LBL[100]
 25:
 26:  LBL[300: Stat=1]
 27:  ! No virtual tray is generated.
 28:  ! Interference cannot be avoided
 29:
 30:  ! Strategy 1 (Default: Discard
 31:  ! the most downstream part and
 32:  ! re-generate tray again
 33:  CALL PKCSGETQUE(CStn ID=R[110:Pk1x StnId],Consec Flag=1,Timeout
      (ms)=0,Offset VR=1,Stat Reg=123)
 34:  IF R[123:Pk1x GetQ stat]<>0,JMP LBL[375]
 35:  CALL PKCSACKQUE(CStn ID=R[110:Pk1x StnId],Skip=3)
 36:  JMP LBL[100]
 37:
 38:  ! Strategy 2: If it is feasible
 39:  !   to build the tray on a clear
 40:  !   area of the conveyor,
 41:  !   contact FANUC.
 42:  !CALL PK_PRGR_CV_SPECIAL_PICKDROP
 43:
 44:  JMP LBL[375]
 45:
 46:  ! Strategy 3: If a separate
 47:  ! Fixed station or indexing
 48:  ! conveyor is available, then
 49:  ! insert calls for the
 50:  ! pick and place programs
 51:  JMP LBL[375]
 52:
 53:  LBL[375]
 54:  WAIT      .02(sec)
 55:  JMP LBL[100]
 56:
 57:  LBL[400:Stat=2]
 58:  ! No virtual tray is generated.
 59:  ! Grouping cannot be achieved in
 60:  ! distance from the leading part
 61:  ! Insert program to handle this.
 62:  ! ex: Pick a part and place it
 63:  ! on a Fixed Stn or
```

```
   64:   ! Indexing conveyor
   65:   WAIT      .02(sec)
   66:   JMP LBL[100]
   67:
   68:   LBL[10000: End]
 [End]
```

7.11.4.2 PK_PRGR_INIT1.TP

This program is the initialization program of pre-grouping for the group 1 robot.

This program gets the type of pre-grouping.

PK_PRGR_INIT1.TP

```
    1:   ! PK_PRGR_INIT1.TP
    2:   !Initializations for pre-grouping
    3:
    4:   !AR[1]: Register num for PRGR tye
    5:
    6:   !Get PRGR type
    7:   CALL PKRBGETPRGRT(Pk Conv ID=R[109:Pk1x CvId],Pk CStn ID=R[110:Pk1x
         StnId],Dp Conv ID=R[129:Dp1x CvId],Dp CStn ID=R[130:Dp1x StnId],PreGrp Type
         Reg=AR[1])
    8:   LBL[10000:End]
[End]
```

7.11.4.3 PK_PRGR_CV_PICKDROP1.TP

This program is used by the group 1 robot to perform arrangement on the same conveyor. This program works as follows.

1 Gets a part from the part queue using PKCSGETQPRGR
2 Regardless of getting or not getting a part from the queue, if the property “Wait Position” is enabled in the *i*RPickTool treeview for this conveyor station, checks if the robot needs to wait at the Wait Position of the Conveyor station which is the Upstream boundary and calculates the wait position using PKCSCALWPOS.PC.
3 If the robot needs to wait because there is no part or part is upstream of the upstream boundary, moves to the wait position using PK_CV_WAITPOS1.TP.
4 If PKCSGETQPRGR did not return a part, the program ends. Here, if the type of pre-grouping is not 1, it waits 0.02 s before ending.
5 If PKCSGETQPRGR returned a part,
6 Sets a tracking trigger for the pickup motions using PKCSTRGPRGR
7 If PKCSTRGPRGR did not set a tracking trigger, it makes the request to set the tracking trigger again.
8 If PKCSTRGPRGR set a tracking trigger,
9 Moves the robot to the approach position.
10 Moves the robot to the pick position while executing the “Gripper Close” operation using Time Before (TB) condition. The gripper close operation is executed via the PKGRCLOSE.PC macro.
11 If the move to the pick position succeeded, then it gets the pick delay using the PKGRGETPKDLY.PC macro.
12 Sets the payload of the robot to “Full gripper” or Payload[2].
13 Moves the robot to the retreat position.
14 Sets a tracking trigger for the plcement motions using PKCSTRGPRGR
15 If PKCSTRGPRGR did not set a tracking trigger, it makes the request to set the tracking trigger again.
16 If PKCSTRGPRGR set a tracking trigger,
17 Moves the robot to the approach position.

18 Moves the robot to the place position while executing the “Gripper Open” operation using Time Before (TB) condition. The gripper open operation is executed via the PKGROPEN.PC macro.
19 If the move to the place position succeeded, then it gets the place delay using the PKGRGETDPDLY.PC macro.
20 Sets the payload of the robot to “Empty gripper” or Payload[1].
21 Moves the robot to the retreat position.
22 Goes to the top of the program to request the next part to arrange.

PK_PRGR_CV_PICKDROP1.TP

```
 1:  ! Pre-grouping Pick & Place G1
 2:  ! Note: This program does not
 3:  ! handle skip outbound flag
 4:  ! and gripper part presence check
 5:  ! If they are needed, modify this
 6:  ! program. For reference, use
 7:  ! PK_CV_PICK11.TP.
 8:  ! This program does not support
 9:  ! multiple-pick & wait position.
10:
11:  LBL[5]
12:  UFRAME_NUM=0
13:  UTOOL_NUM=1
14:
15:  !-------------------------------
16:  LBL[10: Pick&Place Loop]
17:  IF R[101:Cycle Stop G1]>0,JMP LBL[10000]
18:  STOP_TRACKING
19:  ! Get part from que
20:  CALL PKCSGETQPRGR(CStn ID=R[130:Dp1x StnId],Pick Offset VR=1,Drop Offset
       VR=2,Stat Reg=123)
21:  !Convert GETQ stat
22:  IF (R[123:Pk1x GetQ stat]=(1)),R[96:TemporaryUse2]=(2)
23:  IF (R[123:Pk1x GetQ stat]=(0)),R[96:TemporaryUse2]=(1)
24:  !Caclulate WAIT pos
25:  CALL PKCSCALWPOS(CStn ID=R[110:Pk1x StnId],Ref Pos PR=57,Offset
       VR=1,GetQ stat=R[96:TemporaryUse2],Do WaitMove Reg=95,Wait Pos
       PR=55,NoMove Distance=0)
26:  ! Move to wait position
27:  IF (R[95:TemporaryUse1]=0),CALL PK_CV_WAITPOS1(UTOOL Num=1,Appr Ofst
       PR=56,WaitPos PR=55,PkReachable Reg=115)
28:  IF R[123:Pk1x GetQ stat]=0,JMP LBL[20]
29:  IF R[230:PRGR Robot Type]<>1,JMP LBL[9000]
30:  JMP LBL[10000]
31:
32:  LBL[20:SetPickTrigger]
33:  CALL PKCSTRGPRGR(CStn ID=R[110:Pk1x StnId],Timeout (ms)=100,Stat Reg=95)
34:  IF R[95:TemporaryUse1]=0,JMP LBL[80]
35:  IF R[101:Cycle Stop G1]>0,CALL PKCSACKQPRGR(CStn ID=R[130:Dp1x
       StnId],Return=2)
36:  IF R[101:Cycle Stop G1]>0,JMP LBL[10000]
37:  JMP LBL[20]
38:
39:
40:  LBL[80: PickPart]
41:  ! Move to approach
```

```
  42:L PR[57:Pk1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[1] Tool_Offset,PR[56:Pk1
       Cv Ap Ofst]
  43:
  44:   ! Move to pick pos
  45:L PR[57:Pk1 Cv Ref Pos] max_speed CNT0 VOFFSET,VR[1] TB R[125]sec,CALL
       PKGRCLOSE(Gripper ID=R[104:Gripper Id G1],Start Zone=1,End Zone=1)
  46:   ! Pick delay
  47:   CALL PKGRGETPKDLY(Gripper ID=R[104:Gripper Id G1],PickDelay Reg=113)
  48:   WAIT R[113]
  49:   PAYLOAD[2]
  50:   ! Move to retreat
  51:L PR[57:Pk1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[1] Tool_Offset,PR[56:Pk1
       Cv Ap Ofst]
  52:
  53:   !--------------------------------
  54:   ! Place the part in virtual tray
  55:   LBL[125: SetPlaceTrigger]
  56:   STOP_TRACKING
  57:   CALL PKCSTRGPRGR(CStn ID=R[130:Dp1x StnId],Timeout (ms)=100,Stat Reg=95)
  58:   IF R[95:TemporaryUse1]=0,JMP LBL[200]
  59:   IF R[101:Cycle Stop G1]>0,CALL PKCSACKQPRGR(CStn ID=R[130:Dp1x
       StnId],Remove=4)
  60:   IF R[101:Cycle Stop G1]>0,JMP LBL[10000]
  61:   JMP LBL[125]
  62:
  63:   LBL[200: PlacePart]
  64:   ! Move to approach
  65:L PR[57:Pk1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[2] Tool_Offset,PR[58:Dp1
       Cv Ap Ofst]
  66:
  67:   ! Move to place pos
  68:L PR[57:Pk1 Cv Ref Pos] max_speed CNT0 VOFFSET,VR[2] TB R[125]sec,CALL
       PKGROPEN(Gripper ID=R[104:Gripper Id G1],Start Zone=1,End Zone=1)
  69:   ! Place delay
  70:   CALL PKGRGETDPDLY(Gripper ID=R[104:Gripper Id G1],DropDelay Reg=133)
  71:   WAIT R[133]
  72:   PAYLOAD[1]
  73:   ! Move to retreat
  74:L PR[57:Pk1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[2] Tool_Offset,PR[58:Dp1
       Cv Ap Ofst]
  75:   !--------------------------------
  76:
  77:   LBL[2000: Pick/Place OK]
  78:   CALL PKCSACKQPRGR(CStn ID=R[130:Dp1x StnId],Success=1)
  79:   JMP LBL[10]
  80:
  81:   LBL[9000]
  82:   WAIT      .02(sec)
  83:
  84:   LBL[10000: End]
  85:   STOP_TRACKING
[End]
```

7.11.4.4 PK_PRGR_CV_PICKVTRAY1.TP

This program is used by the group 1 robot to perform the pick operation at a conveyor station of a conveyor. The robot picks all the arranged parts at one shot. This program works as follows.

1 Sets the skip outbound FLG to OFF before beginning the pick moves.
2 Gets the number of cells in a tray using PKTRGETNUMCL.
3 Gets a part from the part queue using PKCSGETQCELL. The number of cells is used for the ID of the cell in the layer 2.
4 Regardless of getting or not getting a part from the queue, if the property "Wait Position" is enabled in the *i*RPickTool treeview for this conveyor station, checks if the robot needs to wait at the Wait Position of the Conveyor station which is the Upstream boundary and calculates the wait position using PKCSCALWPOS.PC.
5 If the robot needs to wait because there is no part or part is upstream of the upstream boundary, moves to the wait position using PK_CV_WAITPOS1.TP.
6 If PKCSGETQCELL did not return a part, it makes the request to get the part again.
7 If PKCSGETQCELL returned a part,
Checks whether all cells in layer 1 are filled or not. If some cells in layer 1 are not filled, then the part is returned to the queue using ACK=Skip (3) in the LBL[1000] block and then the steps above are repeated to get a new part.
8 Sets the UTOOL to the gripper's first zone's UTOOL.
9 Moves the robot to the approach position.
10 If the Skip Outbound FLG came on (implying that the robot cannot pick the part which just crossed the downstream boundary), then the part is returned to the queue with ACK=Skip (3) using the PKCSACKQUE macro and it goes back to the top of the program to request a new part to pick.
11 Moves the robot to the pick position while executing the "Gripper Close" operation using Time Before (TB) condition. The gripper close operation is executed via the PKGRCLOSE.PC macro.
12 If the Skip Outbound FLG came on (implying that the robot cannot pick the part which just crossed the downstream boundary), then the part is returned to the queue with ACK=Skip (3) using the PKCSACKQUE macro and it goes back to the top of the program to request a new part to pick.
13 If the move to the pick position succeeded, then it gets the pick delay using the PKGRGETPKDLY.PC macro.
14 Checks if the required parts are picked using the PKGRCHKPP.PC macro if the property "Part presence Check" is enabled for this conveyor station operation in the treeview.
15 If the part presence check fails, it returns the part to the queue with ACK=Remove (4) and goes back to the top of the program to request a part to pick the missing part again.
16 If the part presence check succeeds, it sets the payload of the robot to "Full gripper" or Payload[2].
17 It moves the robot to the retreat position.
18 If the robot is to pick multiple virtual trays, goes to the top of the program to request the next virtual tray to pick, otherwise the program ends.

PK_PRGR_CV_PICKVTRAY1.TP

```
  1:  ! Pre-grouping "Pick Virtual Tray
  2:  ! for grp1 robot.
  3:  ! This program picks the entire
  4:  ! tray built by an upstream
  5:  ! pre-grouping robot in one shot.
  6:
  7:  ! It does not pick up unfilled
  8:  ! trays. Modification is needed
  9:  ! if it is necessary to handle
 10:  ! trays that are not full.
 11:
 12:  ! NOTE: It is necessary to define
 13:  ! one cell in the second layer of
 14:  ! the tray that has the same
```

```
15:   ! coordinates as the cell 1 of
16:   ! layer 1 but a different
17:   ! model_id that is not used in
18:   ! layer 1.
19:
20:   LBL[5]
21:   F[R[102]]=(OFF)
22:   UFRAME_NUM=0
23:
24:   LBL[10: Pick Loop]
25:   IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
26:   STOP_TRACKING
27:   ! Get total cells in tray
28:   CALL PKTRGETNUMCL(Tray ID=R[121:Pk1x TrayId],Layer num=0,NumCells
      Reg=122)
29:   ! If R[122]=0, invalid tray
30:   IF R[122:Pk1x TrayCtr]=0,CALL PKPOSTERR(Error Code=127038,ABORT=2)
31:   ! Get tray cell from queue
32:   CALL PKCSGETQCELL(CStn ID=R[110:Pk1x StnId],Cell num=R[122:Pk1x
      TrayCtr],Timeout (ms)=100,Offset VR=1,Stat Reg=123,NumUnfilCel Reg=96)
33:   CALL PKCSCALWPOS(CStn ID=R[110:Pk1x StnId],Ref Pos PR=57,Offset
      VR=1,GetQ stat=R[123:Pk1x GetQ stat],Do WaitMove Reg=95,Wait Pos
      PR=55,NoMove Distance=0) ;
34:   IF (R[95:TemporaryUse1]=0),CALL PK_CV_WAITPOS1(UTOOL Num=R[107:Counter
      G1],Appr Ofst PR=56,WaitPos PR=55,PkReachable Reg=115)
35:   IF R[123:Pk1x GetQ stat]<>0,JMP LBL[10]
36:   ! If unfilled cells exist,
37:   ! skip them and don't pick.
38:   IF R[96:TemporaryUse2]<>0,JMP LBL[1000]
39:
40:   LBL[50: Multipick]
41:
42:   LBL[80]
43:
44:   ! Set Utool
45:   UTOOL_NUM=R[107:Counter G1]
46:   ! Move to approach
47:L PR[57:Pk1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[1] Tool_Offset,PR[56:Pk1
      Cv Ap Ofst]
48:   ! Flag on when Skip outbound true
49:   IF (F[R[102]]),JMP LBL[1000]
50:
51:   ! Set Pick/drop registers
52:   CALL
      PK_GR_GET_ENDZONE1(PickDropCounter=R[107:Counter ],Multiple=R[112:Pk1x
      PkMult],Quantity=R[111:Pk1x Qty],End Zone Reg=106)
53:   ! Move to pick pos
54:L PR[57:Pk1 Cv Ref Pos] max_speed CNT0 VOFFSET,VR[1] TB R[125]sec,CALL
      PKGRCLOSE(Gripper ID=R[104:Gripper Id G1],Start Zone=R[107:Counter G1],End
      Zone=R[106:Gr EndZone G1])
55:   ! Chk Skip Pick
56:   IF (F[R[102]]),JMP LBL[1000]
57:   ! Pick delay
58:   CALL PKGRGETPKDLY(Gripper ID=R[104:Gripper Id G1],PickDelay Reg=113)
59:   WAIT R[113]
```

```
  60:   ! Check part pres
  61:   CALL PKCSGETPPCHK(CStn ID=R[110:Pk1x StnId],PartPresChk Reg=116)
  62:   IF R[116:Pk1x UsePPChk]=0,JMP LBL[90]
  63:   ! PKGRCHKPP grip_id, operation,
  64:   ! stat_reg, StartZone,
  65:   !EndZone (opt)
  66:   CALL PKGRCHKPP(Gripper ID=R[104:Gripper Id G1],Pick=0,Stat Reg=117,Start
        Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone G1])
  67:   LBL[90]
  68:   PAYLOAD[2]
  69:   ! Move to retreat
  70:L PR[57:Pk1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[1] Tool_Offset,PR[56:Pk1
        Cv Ap Ofst]
  71:   IF R[117:Pk1x PPStat]<>0,JMP LBL[1500]
  72:   JMP LBL[2000]
  73:
  74:   LBL[1000: Missed part]
  75:   F[R[102]]=(OFF)
  76:   ! Return part in VR1 to Queue
  77:   CALL PKCSACKQUE(CStn ID=R[110:Pk1x StnId],Skip=3)
  78:   JMP LBL[10]
  79:
  80:   LBL[1500: Dropped part]
  81:   F[R[102]]=(OFF)
  82:   ! Return part in VR1 to Queue
  83:   CALL PKCSACKQUE(CStn ID=R[110:Pk1x StnId],Remove=4)
  84:   R[117:Pk1x PPStat]=0
  85:   ! Insert code for rejecting parts
  86:   ! a reject location.
  87:   JMP LBL[10]
  88:
  89:
  90:   LBL[2000: Pick OK]
  91:   ! PKACKQ grp, vr, ack
  92:   CALL PKCSACKQUE(CStn ID=R[110:Pk1x StnId],Success=1)
  93:   -- Update the pick counter
  94:   IF R[112:Pk1x PkMult]=0,JMP LBL[2050]
  95:   R[107:Counter G1]=R[107:Counter G1]+R[111:Pk1x Qty]
  96:   JMP LBL[2100]
  97:
  98:   LBL[2050]
  99:   R[107:Counter G1]=R[107:Counter G1]+1
 100:   IF R[107:Counter G1]<=R[111:Pk1x Qty],JMP LBL[10]
 101:
 102:   LBL[2100]
 103:   LBL[10000: End]
 104:   STOP_TRACKING
 105:   F[R[102]]=(OFF)
[End]
```

7.12 BUFFER FUNCTION

This section describes the setup and use of the Plug & Play buffer functions. This option is included in the R857: *i*RPickTool / Option Package Add-on (for North America, R827: iRPickTool / Production Control Add-on) option, and it contains the User Interface for the buffer operations in the OP_FS_xxx menu, TP programs for buffer handling and some KAREL macros to get the buffer data that is setup in the user interface. You can then use buffers without extra programming using the Plug & Play concept when the robot picks and places single parts.

NOTE
This option is not available for J944: *i*RPickTool Basic.
However, *i*RPickTool Basic contains the fundamental KAREL macros for buffers (such as PKFSCLRBUF, PKFSPUTBUF, PKFSGETBUF) that customers can use to build their own buffer application.

A buffer is a Fixed Station with a tray.
The buffer is called a Static buffer. This buffer does not index. As long as there are unfilled cells in the buffer, a robot can place parts in buffer. As long as there are filled cells, a robot can pick parts from buffer. A static buffer with 4 tray cells is shown below:

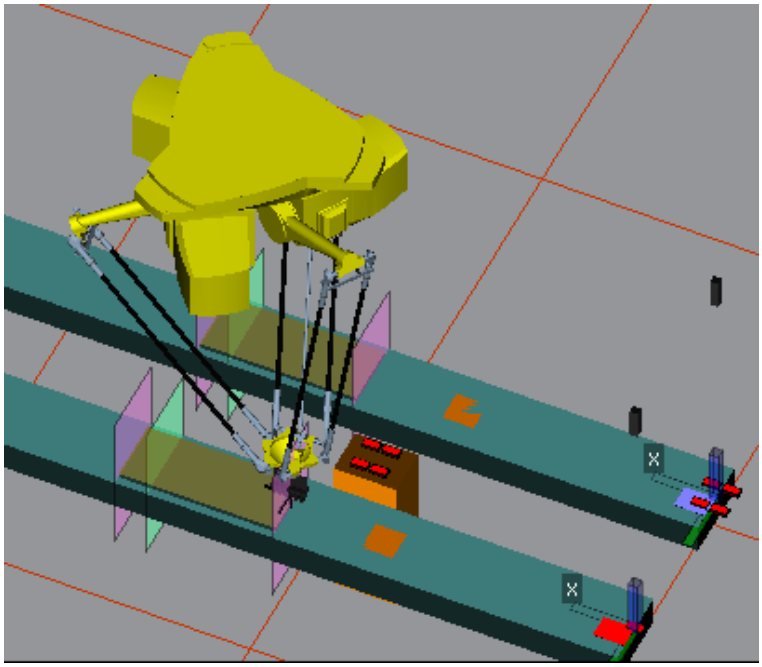


Fig. 7.12 (a) A buffer that consists of 4 cells

*i*RPickTool buffers are supported only when the pick and drop are at Conveyor Stations. The following cases are NOT supported:

- Conveyor Station pick and Fixed Station place.
- Fixed Station pick and Conveyor Station place.
- Fixed Station pick and Fixed Station place.

There can only be one buffer per *i*RPickTool robot (group).

The *i*RPickTool Buffer operations will work well for only “single pick” and “single drop” applications.

7.12.1 The Buffer Algorithm

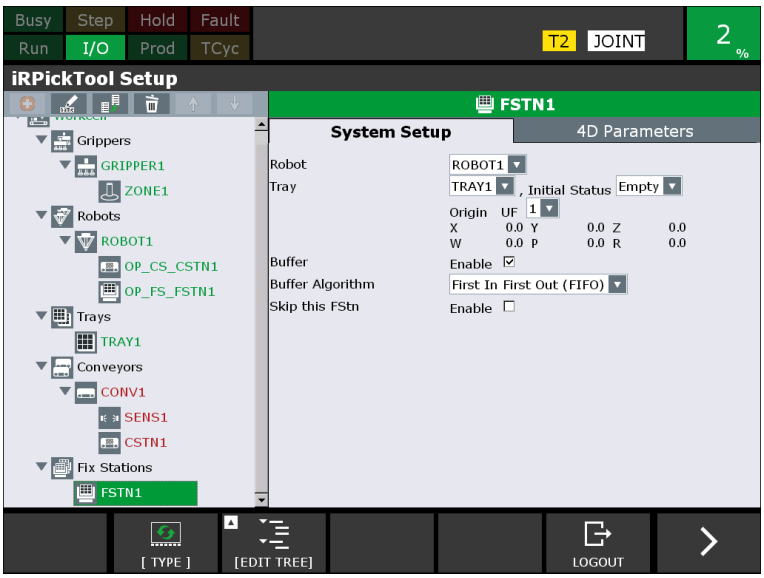
Two types of buffer algorithms will be available: “First In First Out (FIFO)” and “Last In First Out (LIFO)”. The default algorithm shall be FIFO.

The buffer algorithm works as follows:

- When a pick target is available, but drop target is not available, and the buffer has empty cells, the robot places the part in the buffer.
- When a pick target is known to exist but it is upstream of the tracking upstream boundary, then a place target is available, and, the robot may still pick from a buffer if parts exist in it, based on “part time for buffer pick”. Part time for buffer pick is the minimum time for the part to arrive at the upstream boundary of the infeed conveyor based on the current conveyor speed.
- When a place target is available but not a pick target, and the buffer has parts, the robot picks the part from the buffer.
- When a place target is known to exist but it is upstream of the tracking upstream boundary, then the robot may still place to a buffer if empty cells exist in it, based on “part time for buffer drop”. The part time for buffer drop is the minimum time for the drop target to arrive at the upstream boundary of the outfeed conveyor based on the current conveyor speed.
- When a pick target is not available, and the buffer is empty, then the robot waits at the “wait position” of the input conveyor if the “Wait Position” property of the pick conveyor station is enabled.
- When a drop target is not available, and the buffer is full, and the robot has already picked the parts from the infeed conveyor, then the robot waits at the “wait position” of the output conveyor.
- When there are parts in the buffer to pick from, but there is no pick target, the robot will pick from the buffer only if a drop target is available at that time.

7.12.2 Setting up a Buffer

To create a buffer, create a Fixed Station in the *i*RPickTool tree-view. Section 6.7, “SETUP OF A FIXED STATION” has details on how to create a Fixed Station.



Once the Fixed Station is created,

- Assign a tray with [Initial Status] = “Empty”.
- Enable the buffer using the checkbox for [Buffer].
- Select the [Buffer Algorithm], “First in First Out(FIFO)” or “Last in First Out(LIFO)”

[Buffer]

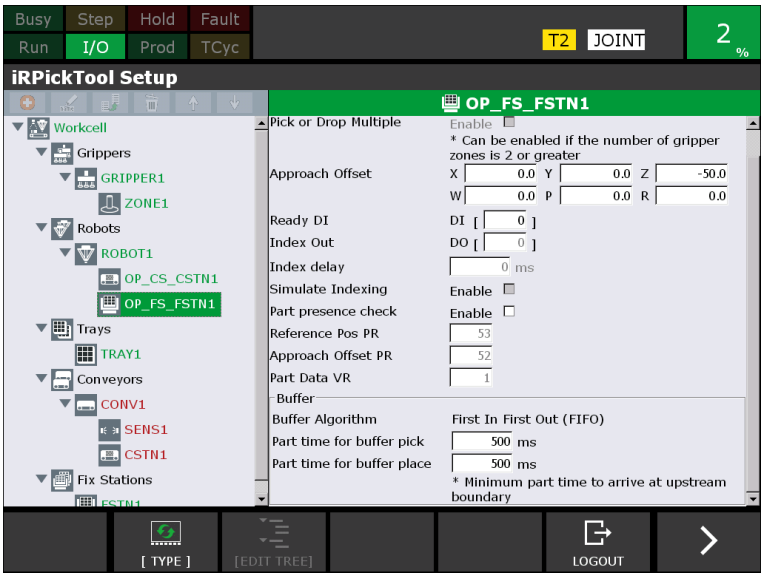
If “Enable” is checked, this Fixed Station is used for a buffer.

[Buffer Algorithm]

Specify the order in which parts are picked out of the buffer. This item can be specified only when the buffer is enabled.

First in First Out (FIFO) : The first part placed in a buffer will be the first one to be picked.
Last in First Out (LIFO) : The last part placed in the buffer will be the first one to be picked.

An OP_FS_xxxx node is automatically created under the Robot node when you create a Fixed Station – for ex: OP_FS_FSTN1 as shown in the figure below.



In the OP_FS_ node, set the properties for the buffer correctly. Subsection 7.3.2, “Fixed Station Robot Operation” has details.

[Part time for buffer pick]

When a pick target is known to exist on the infeed conveyor but it is upstream of the tracking upstream boundary by this amount, then the robot will pick from a buffer if parts exist in it. Set the Part time for buffer pick with the knowledge of the conveyor speed that will be used at the infeed.

[Part time for buffer place]

When a place target is known to exist on the outfeed conveyor but it is upstream of the tracking upstream boundary by this amount, then the robot will place the part to a buffer if empty cells exist in it. Set the Part time for buffer place with the knowledge of the conveyor speed that will be used at the outfeed.

NOTE

You can enter even a negative value in milliseconds. When the value of time is positive, the part is upstream of the track upstream boundary. When the value is negative, it is downstream of the track upstream boundary.

When you run production, adjust the values of the “part time to pick from buffer” and “part time to place to buffer” until you see the robot picking and placing correctly from/to the conveyors and the buffers. You will be able to change these time values from the OP_FS_ menu without any confirmation prompts.

7.12.3 Buffer KAREL Macros

The buffer related KAREL macros are defined in this section.

7.12.3.1 PKRBGETBUFID.PC

This program is used to get the Fixed Station ID of the buffer station based on the robot ID.
If an invalid robot ID is passed as input, buffer ID will be zero.
If a buffer is not defined, then buffer ID will be zero.

Argument 1:
Specify an ID of a robot.

Argument 2:
Specify the Numeric Register in which to return the ID of the buffer that is used by the robot.

Example:
To get the ID of the buffer used by a Robot whose ID is 1 into R[88], use the following instruction.

```
CALL PKRBGETBUFID(Robot ID=1,BufferID Reg=88)
```

7.12.3.2 PKFSGETBUFPT.PC

This program is used to get the Part Time for Buffer pick in milliseconds for a specified buffer that you set in the *i*RPickTool treeview in the OP_FS_ menu.

Argument 1:
Specify the ID of the Fixed Station which is used as a buffer.

Argument 2:
Specify the Numeric Register in which to return the Part Time for buffer pick in milliseconds.

Example:
To get the Part Time for Buffer Pick of a buffer whose ID is 1 in R[95], use the following instruction.

```
CALL PKFSGETBUFPT(Buffer Stn ID=1,BufPickTime Reg=95)
```

7.12.3.3 PKFSGETBUFDT.PC

This program is used to get the Part Time for Buffer place in milliseconds for a specified buffer that you set in the *i*RPickTool treeview in the OP_FS_ menu.

Argument 1:
Specify the ID of the Fixed Station which is used as a buffer.

Argument 2:
Specify the Numeric Register in which to return the Part Time for buffer place in milliseconds.

Example:
To get the Part Time for Buffer Place of a buffer whose ID is 1 in R[95], use the following instruction.

```
CALL PKFSGETBUFDT(Buffer Stn ID=1,BufDropTime Reg=95)
```

7.12.4 Buffer TP Programs

This section describes the buffer TP programs. In addition to these programs, you should also become familiar with the following TP programs which have some branches for buffer logic.

- PK_CV_PICK11.TP
- PK_CV_DROP11.TP
- PK_INIT1.TP
- PK_PICK1.TP
- PK_DROP1.TP

Refer to the section 7.10, "STANDARD TP PROGRAMS" in detail.

7.12.4.1 PK_BF_GET_DATA1.TP

This program is called from PK_INIT1.TP.
This program contains the instructions for retrieving the buffer and Fixed Station related data you set up in the *i*RPickTool treeview into Numeric and Position Registers.

- It clears any pending acks for the buffer using PKFSCLRACK.
- It clears the buffer using PKFSCLRBUF.
- It initializes the buffer to empty status using PKFSPUTBUF.

PK_BF_GET_DATA1.TP

```
    1:  ! PK_BF_GET_DATA1
    2:  ! Set Num Reg for G1 Buffer
    3:
    4:  CALL PKFSGETIXSIM(FStn ID=R[199:Buf Id G1],IndexSim Reg=95)
    5:  CALL PKFSGETAPPR(FStn ID=R[199:Buf Id G1],ApproachPR Reg=95)
    6:  CALL PKFSSETRDY(FStn ID=R[199:Buf Id G1],Ready)
    7:  CALL PKFSCLRACK(FStn ID=R[199:Buf Id G1])
    8:  CALL PKFSCLRBUF(FStn ID=R[199:Buf Id G1],Index=1)
    9:  CALL PKFSPUTBUF(FStn ID=R[199:Buf Id G1],Index=1)
   10:  --
   11:  R[204:Buf Ctr G1]=0
   12:  R[205:Picked G1]=0
   13:  R[126:Buf PP Stat G1]=0
   14:  LBL[1000]
   15:
   16:  LBL[10000: End]
   17:
[End]
```

7.12.4.2 PK_BF_PICK1.TP

This program does the “pick from buffer” operation.

- Note that it uses the same counters and pick quantity as the tracking pick program PK_CV_PICK11.TP.
- Refer to the description of PK_FS_PICK11.TP in the subsection 7.10.2.15 “PK_FS_PICK11.TP” for understanding the logic of this program.

PK_BF_PICK1.TP

```
 1:  -- Pick - Buffer G1
 2:
 3:  LBL[5]
 4:  IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
 5:
 6:  --
 7:  --
 8:  CALL PKFSGETUF(FStn ID=R[199:Buf Id G1],UFrameNum Reg=95)
 9:  UFRAME_NUM=R[95:TemporaryUse1]
10:
11:  LBL[10]
12:  --
13:  --
14:  CALL PK_FS_READY(FStn ID=R[199:Buf Id G1],Stat Reg=95)
15:  IF R[95:TemporaryUse1]=0,JMP LBL[20]
16:  IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
17:  WAIT      .02(sec)
18:  JMP LBL[10]
19:
20:  LBL[20: Pick Loop]
21:  IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
22:
23:  --
24:  --
25:  CALL PKFSGETBUF(FStn ID=R[199:Buf Id G1],Index=1,Part, Offset VR=1,
       Stat Reg=123)
26:  IF R[123:Pk1x GetQ stat]=0,JMP LBL[200]
27:  JMP LBL[10000]
28:
29:  LBL[200: Multipick]
30:  CALL PK_GR_GET_ENDZONE1( “PickDropCounter”=R[107:Counter ],
       “Multiple”=0, “Quantity”=1, “End Zone Reg”=106)
31:
32:
33:  LBL[300]
34:  UTOOL_NUM=R[107:Counter G1]
35:  --Move to approach
36:  L PR[53:Bf Ref Pos] max_speed CNT100 VOFFSET,VR[1] Tool_Offset,PR[52:Bf Ap
       Ofst]
37:  Move to pick pos ;
38:  L PR[53:Bf Ref Pos] max_speed CNT0 VOFFSET,VR[1] TB R[127]sec,
         CALL PKGRCLOSE(“Gripper ID”=R[104:Gripper Id G1],“Start
       Zone”=R[107:Counter G1],
         “End Zone”=R[106:Gr EndZone G1])
39:  Pick delay ;
```

```
  40:   CALL PKGRGETPKDLY(“Gripper ID”  =R[104:Gripper Id G1],“PickDelay Reg”=95)
  41:   WAIT R[95] ;
  42:   -Check part pres ;
  43:   CALL PKFSGETPPCHK(“FStn ID”=R[199:Buf Id G1],“PartPresChk Reg”=95)
  44:   IF R[95:TemporaryUse1]=0,JMP LBL[400]
  45:   PKGRCHKPP grip_id, operation,
  46:   stat_reg, StartZone,
  47:   EndZone (opt)
  48:   CALL PKGRCHKPP(“Gripper ID”=R[104:Gripper Id G1],“Pick”=0,“Stat Reg”=126,
           “Start Zone”=R[107:Counter G1],“End Zone”=R[106:Gr EndZone G1])
  49:   LBL[400]
  50:   Full gripper payload
  51:   PAYLOAD[2]
  52:   Move to retreat
  53:L PR[53:Bf Ref Pos] max_speed CNT100 VOFFSET,VR[1] Tool_Offset,PR[52:Bf Ap
       Ofst]
  54:   IF R[126:Buf PP Stat G1]<>0,JMP LBL[1500]
  55:   JMP LBL[2000]
  56:
  57:
  58:   LBL[1500: PartGotdropped]
  59:   -Return part in VR1 to Queue
  60:   CALL PKFSACKBUF(“FStn ID”=R[199:Buf Id G1],“Return”=2)
  61:   R[126:Buf PP Stat G1]=0
  62:   JMP LBL[10000]
  63:
  64:
  65:   LBL[2000: Pick ok]
  66:   Ack the part has been picked
  67:   ! PKFSACKBUF(buf_id, ack, vr (opt
  68:   CALL PKFSACKBUF(“FStn ID”=R[199:Buf Id G1],“Success”=1)
  69:   Update tray cell count
  70:   R[204:Buf Ctr G1]=R[204:Buf Ctr G1]-1
  71:   Update pick counter
  72:   R[107:Counter G1]=R[107:Counter G1]+1
  73:   R[205:Picked G1]=1
  74:
  75:   LBL[3000: CheckIndex]
  76:   ! For indexing buffers only
  77:
  78:
  79:   LBL[10000: End]
[End]
```

7.12.4.3 PK_BF_DROP1.TP

This program does the “place to buffer” operation.

- Note that it uses the same counters and place quantity as the tracking drop program PK_CV_DROP11.TP.
- Refer to the description of PK_FS_DROP11.TP in the subsection 7.10.2.16 “PK_FS_DROP11.TP” for understanding the logic of this program.

PK_BF_DROP1.TP

```
  1:  -- Drop - Buffer G1
  2:
  3:  LBL[5]
  4:  IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
  5:
  6:  -- Get FStn Tray Uframe
  7:  -- PKFSGETUF(fs_id, reg_uf)
  8:  CALL PKFSGETUF( “FStn ID”=R[199:Buf Id G1], “UFrameNum Reg”=95)
  9:  UFRAME_NUM=R[95:TemporaryUse1] ;
 10:
 11:  LBL[10]
 12:  -- Check if FS is ready
 13:  -- PK_FS_READY(fs_id, stat)
 14:  CALL PK_FS_READY( “FStn ID”=R[199:Buf Id G1], “Stat Reg”=95)
 15:  IF R[95:TemporaryUse1]=0,JMP LBL[20]
 16:  IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
 17:  WAIT      .02(sec)
 18:  JMP LBL[10]
 19:
 20:  LBL[20]
 21:  IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
 22:
 23:  -- Get an empty cell from the tray
 24:  -- PKFSGETBUF(buf_id, index, part_or_cell, vr, stat_reg, model_id (opt))
 25:  CALL PKFSGETBUF( “FStn ID”=R[199:Buf Id G1],
        “Index”=1, “Cell”=1, “Offset VR”=1, “Stat Reg”=143)
 26:  IF R[143:Dp1x GetQ stat]=0,JMP LBL[200]
 27:  JMP LBL[10000]
 28:
 29:  LBL[200]
 30:  CALL PK_GR_GET_ENDZONE1( “PickDropCounter”=R[107:Counter ],
      “Multiple”=0, “Quantity”=1, “End Zone Reg”=106)
 31:
 32:
 33:  LBL[300]
 34:  UTOOL_NUM=R[107:Counter G1]
 35:  -- Move to approach
 36:L PR[53:Bf Ref Pos] max_speed CNT100 VOFFSET,VR[1] Tool_Offset,
    PR[52:Bf Ap Ofst]
 37:  -- Move to drop pos
 38:L PR[53:Bf Ref Pos] max_speed CNT0 VOFFSET,VR[1] TB R[147]sec,
      CALL PKGROPEN( “Gripper ID”=R[104:Gripper Id G1],
       “Start Zone”=R[107:Counter G1], “End Zone”=R[106:Gr EndZone G1])
 39:  -- Drop delay
 40:  CALL PKGRGETDPDLY( “Gripper ID”=R[104:Gripper Id G1], “DropDelay Reg”=95)
 41:  WAIT R[95]
 42:  -- Check part pres
 43:  CALL PKFSGETPPCHK( “FStn ID”=R[199:Buf Id G1], “PartPresChk Reg”=95)
 44:  IF R[95:TemporaryUse1]=0,JMP LBL[400]
 45:  -- PKGRCHKPP grip_id, operation,
 46:  -- stat_reg, StartZone,
 47:  -- EndZone (opt)
 48:  CALL PKGRCHKPP( “Gripper ID”=R[104:Gripper Id G1],
```

```
        “Place”=1,“Stat Reg”=126,“Start Zone”=R[107:Counter G1],
         “End Zone”=R[106:Gr EndZone G1])
   49:  LBL[400]
   50:  -- Empty gripper payload
   51:  PAYLOAD[1]
   52:  -- Move to retreat
   53:L PR[53:Bf Ref Pos] max_speed CNT100 VOFFSET,VR[1] Tool_Offset,
     PR[52:Bf Ap Ofst]
   54:  IF R[126:Buf PP Stat G1]<>0,JMP LBL[1500]
   55:  JMP LBL[2000]
   56:
   57:
   58:  LBL[1500: PartGotdropped]
   59:  -- Return part in VR1 to Queue
   60:  CALL PKFSACKBUF( “FStn ID”=R[199:Buf Id G1], “Return”=2)
   61:  R[126:Buf PP Stat G1]=0
   62:  JMP LBL[10000]
   63:
   64:
   65:  LBL[2000: Pick ok]
   66:  -- Ack the part has been picked
   67:  ! PKFSACKBUF(buf_id, ack, vr (opt
   68:  CALL PKFSACKBUF( “FStn ID”=R[199:Buf Id G1], “Success”=1)
   69:  -- Update tray cell count
   70:  R[204:Buf Ctr G1]=R[204:Buf Ctr G1]+1
   71:  -- Update pick counter ;
   72:  R[107:Counter G1]=R[107:Counter G1]+1
   73:  R[205:Picked G1]=0
   74:
   75:  LBL[3000: CheckIndex]
   76:  ! For indexing buffers only
   77:
   78:
   79:  LBL[10000: End]
[End]
```

7.12.4.4 PK_BF_OK2PICK1.TP

This program is called from PK_BUF_PICK1.TP.
This program checks if a pick operation could be performed from a buffer station. A pick operation can be performed only if there is a part in the buffer. The check is performed by trying to get a part from the buffer using PKFSGETBUF, checking its status and then returning the part back to the buffer.

It is an argument-based program which is called by two arguments.
Argument 1: Buffer ID
Argument 2: The register number in which the status is returned. If a part exists, 1 is returned, else 0 is returned.

PK_BF_OK2PICK1.TP

```
   1:  !PK_BF_OK2PICK1
   2:  ! Arguments
   3:  ! Arg 1: Buf Id
   4:  ! Arg 2: ok2Pick reg (0 or 1)
   5:
```

```
    6:  ! Get a part from buf
    7:  ! PKFSGETBUF(buf_id, index,
    8:  ! part_or_cell, vr, stat_reg,
    9:  ! model_id (opt))
   10:  CALL PKFSGETBUF(FStn ID=AR[1],Index=2,Part,Offset VR=1,Stat Reg=123)
   11:  IF R[123:Pk1x GetQ stat]>0,JMP LBL[1000]
   12:  ! Ok
   13:  ! Return the part or cell
   14:  CALL PKFSACKBUF(FStn ID=AR[1],Return)
   15:  R[AR[2]]=1
   16:  JMP LBL[10000]
   17:
   18:  LBL[1000: Not ok]
   19:  R[AR[2]]=0
   20:
   21:  LBL[10000: End]
[End]
```


7.12.4.5 PK_BF_OK2DROP1.TP

This program is called from PK_BUF_DROP1.TP and PK BUF_PICK1.TP.
This program checks if a place operation could be performed at a buffer station. A place operation can be performed only if there is an empty cell in the buffer. The check is performed by trying to get a cell from the buffer using PKFSGETBUF, checking its status and then returning the part back to the buffer.

It is an argument-based program which is called by two arguments.
Argument 1: Buffer ID
Argument 2: The register number in which the status is returned. If an empty cell exists, 1 is returned, else 0 is returned.

PK_BF_OK2DROP1.TP

```
    1:  !PK_BF_OK2DROP1
    2:  ! Arguments
    3:  ! Arg 1: Buf Id
    4:  ! Arg 2: ok2Pick reg (0 or 1)
    5:
    6:  ! Get an empty cell from buf
    7:  ! PKFSGETBUF(buf_id, index,
    8:  ! part_or_cell, vr, stat_reg,
    9:  ! model_id (opt))
   10:  CALL PKFSGETBUF(FStn ID=AR[1],Index=1,Cell,Offset VR=1,Stat Reg=143)
   11:  IF R[143:Dp1x GetQ stat]>0,JMP LBL[1000]
   12:  ! Ok
   13:  ! Return the part or cell
   14:  CALL PKFSACKBUF(FStn ID=AR[1],Return)
   15:  R[AR[2]]=1
   16:  JMP LBL[10000]
   17:
   18:  LBL[1000: Not ok]
   19:  R[AR[2]]=0
   20:
   21:  LBL[10000: End]
[End]
```

7.12.4.6 PK_CV_OK2PICK1.TP

This program is called from PK_BUF_PICK1.TP.
This program checks if a pick operation could be performed at a conveyor station. A pick operation could be performed only if the part on the infeed is upstream by less than or equal to "part time to pick from buffer". Otherwise, it is necessary to check if a pick can be performed at the buffer. The check is performed by using PKCSGETTIME to get the location of the part on the conveyor relative to the upstream boundary in time units.

It is an argument-based program which is called by six arguments.
Argument 1: Conveyor ID
Argument 2: Buffer ID
Argument 3: The order (refer to Argument 2 in Subsection 6.8.5.7, "PKCSGETTIME.PC.")
Argument 4: Consec Flag (refer to Argument 3 in Subsection 6.8.5.7, "PKCSGETTIME.PC.")
Argument 5: The register number where "Part time for buffer pick" is returned
Argument 6: The register number in which the status is returned.

PK_CV_OK2PICK1.TP

```
 1:  !PK_CV_OK2PICK1
 2:  ! Input
 3:  ! Arg 1: Cv Id
 4:  ! Arg 2: Buf id
 5:  ! Arg 3: Order
 6:  ! Arg 4: Pick Counter
 7:  ! Arg 5: Buf partTime Reg
 8:  ! Arg 6: ok2Pick reg (0 or 1)
 9:
10:  ! Get the conveyor name
11:  CALL PKCVGETNAME(Conv ID=AR[1],ConvName SR=4)
12:  ! Get conv id from name
13:  CALL PKCSGETID(SR[4],110)
14:  ! Get pick time for buffer
15:  ! PKFSGETBUFPT(buf_id, time_reg)
16:  CALL PKFSGETBUFPT(Buffer Stn ID=AR[2],BufPickTime Reg=AR[5])
17:
18:  ! Check if the part is
19:  ! sufficiently upstream
20:  ! Get time from Up bound
21:  ! PKCSGETTIME(cs_id, order,
22:  !   pick_ctr, pick_qty,
23:  ! time_reg, vr, stat_reg)
24:  CALL PKCSGETTIME(CStn ID=R[110:Pk1x StnId],Order=AR[3],
      Consec Flag=AR[4],Quantity=R[111:Pk1x Qty],Time Reg=95,Offset VR=1,
      Stat Reg=96)
25:  IF R[96:TemporaryUse2]=3,JMP LBL[1000]
26:  IF (R[95:TemporaryUse1]>R[AR[5]]),JMP LBL[1000]
27:
28:  LBL[500: Ok]
29:  R[AR[6]]=1
30:  JMP LBL[10000]
31:
32:  LBL[1000: Not ok]
33:  R[AR[6]]=0
34:
35:  LBL[10000: End]
```

[End]

7.12.4.7 PK_CV_OK2DROP1.TP

This program is called from PK_BUF_PICK1.TP and PK BUF_DROP1.TP.
This program checks if a place operation could be performed at a conveyor station. A place operation could be performed only if the part on the outfeed is upstream by less than or equal to "part time for buffer place". Otherwise, it is necessary to check if a place can be performed at the buffer. The check is performed by using PKCSGETTIME to get the location of the part on the conveyor relative to the upstream boundary in time units.

It is an argument-based program which is called by six arguments.
Argument 1: Conveyor ID
Argument 2: Buffer ID
Argument 3: The order (refer to Argument 2 in Subsection 6.8.5.7, "PKCSGETTIME.PC.")
Argument 4: Consec Flag (refer to Argument 3 in Subsection 6.8.5.7, "PKCSGETTIME.PC.")
Argument 5: The register number where "Part time for buffer place" is returned
Argument 6: The register number in which the status is returned.

PK_CV_OK2DROP1.TP

```
 1:  !PK_CV_OK2DROP1
 2:  ! Input
 3:  ! Arg 1: Cv Id
 4:  ! Arg 2: Buf id
 5:  ! Arg 3: Order
 6:  ! Arg 4: Drop Counter
 7:  ! Arg 5: Buf partTime Reg
 8:  ! Arg 6: ok2Pick reg (0 or 1)
 9:
10:  ! Get the conveyor name
11:  CALL PKCVGETNAME( "Conv ID"=AR[1],ConvName SR=5)
12:  ! Get conv id from name
13:  CALL PKCSGETID(CONV1=SR[5],CStn ID Reg=130)
14:  ! Get drop time for buffer
15:  ! PKFSGETBUFDT(buf_id, time_reg)
16:  CALL PKFSGETBUFDT(Buffer Stn ID=AR[2],BufDropTime Reg=AR[5])
17:
18:  ! Check if the part is
19:  ! sufficiently upstream
20:  ! Get time from Up bound
21:  ! PKCSGETTIME(cs_id, order,
22:  !   pick_ctr, pick_qty,
23:  ! time_reg, vr, stat_reg)
24:  CALL PKCSGETTIME(CStn ID=R[130:Dp1x StnId],Order=AR[3],
       Consec Flag=AR[4],Quantity=R[131:Dp1x Qty],Time Reg=95,
       Offset VR=2,Stat Reg=96)
25:  IF R[96:TemporaryUse2]=3,JMP LBL[1000]
26:  IF (R[95:TemporaryUse1]>R[AR[5]]),JMP LBL[1000]
27:
28:  LBL[500: Ok]
29:  R[AR[6]]=1
30:  JMP LBL[10000]
31:
32:  LBL[1000: Not ok]
33:  R[AR[6]]=0
```

```
   34:
   35:   LBL[10000: End]
[End]
```

7.12.4.8 PK_BUF_PICK1.TP

This program is called from PK_PICK1.TP.

This program decides if the part should be picked from a buffer (PK_BF_PICK1.TP) or from the infeed conveyor (PK_CV_PICK11.TP).

PK_BUF_PICK1.TP

```
    1:   ! PK_BUF_PICK1
    2:   ! Buffer Pick Grp1
    3:
    4:   LBL[50: Buf loop]
    5:   IF R[205:Picked G1]>0,JMP LBL[10000]
    6:   R[202:Buf Available G1]=0
    7:   R[203:CS Available G1]=0
    8:   R[97:TemporaryUse3]=0
    9:   LBL[100]
   10:   IF R[129:Dp1x CvId]>0,CALL PK_CV_OK2DROP1( “Conv ID”=R[129:Dp1x CvId],
“Buffer ID”=R[199:Buf Id G1],“Order”=1, “Drop Counter”=1, “BufDropTime Reg”=201,
“Ok2Drop Reg”=97)
   11:   CALL PK_BF_OK2PICK1(“Buffer ID”=R[199:Buf Id G1], “Ok2Pick Reg”=202)
   12:   CALL PK_CV_OK2PICK1(“Conv ID”=R[109:Pk1x CvId], “Buffer ID”=R[199:Buf Id G1],
“Order”=1,“Pick Counter”=R[107:Counter G1],“BufPickTime Reg”=200,“Ok2Pick Reg”=203)
   13:   IF R[203:CS Available G1]>0,CALL PK_CV_PICK11
   14:   IF ((R[203:CS Available G1]<=0) AND (R[202:Buf Available G1]>0) AND
(R[97:TemporaryUse3]>0)),CALL PK_BF_PICK1
   15:   IF R[205:Picked G1]<1,CALL PK_CV_PICK11
   16:   IF R[205:Picked G1]<1,JMP LBL[50]
   17:   LBL[10000]
 [End]
```

7.12.4.9 PK_BUF_DROP1.TP

This program is called from PK_DROP1.TP.

This program decides if the part should be placed to a buffer (PK_BF_DROP1.TP) or to the outfeed conveyor (PK_CV_DROP11.TP).

PK_BUF_DROP1.TP

```
    1:  ! PK_BUF_DROP1
    2:  ! Buffer Drop Grp1
    3:
    4:  LBL[50: Buf loop]
    5:  IF R[205:Picked G1]<1,JMP LBL[10000]
    6:  R[202:Buf Available G1]=0
    7:  R[203:CS Available G1]=0
    8:  LBL[100]
    9:  CALL PK_BF_OK2DROP1( "Buffer ID"=R[199:Buf Id G1],"Ok2Drop Reg"=202)
   10:  CALL PK_CV_OK2DROP1("Conv ID"=R[129:Dp1x CvId],"Buffer ID"=R[199:Buf Id
G1],"Order"=1,"Drop Counter"=R[107:Counter G1],"BufDropTime Reg"=201,"Ok2Drop
Reg"=203)
   11:  IF R[203:CS Available G1]>0,CALL PK_CV_DROP11
   12:  IF ((R[203:CS Available G1]<=0) AND (R[202:Buf Available G1]>0) AND
(R[205:Picked G1]>0)),CALL PK_BF_DROP1
   13:  IF R[205:Picked G1]>0,CALL PK_CV_DROP11
   14:  IF R[205:Picked G1]>0,JMP LBL[50]
   15:  LBL[10000]
 [End]
```

7.13 *i*RPickTool Safe Retreat with Skip Outbound Motion

Currently if the robot encounters "LNTK-042 Skip Outbound move" error during or after picking a part (or also during / after placing a part) on a conveyor, it will skip the retreat motion that should follow.
As shown in figure 1, if the robot is placing parts into a deep tray or box, skipping the retreat motion can cause the robot to collide with the side-wall of the current tray as it moves to the next taught position.

NOTE
LNTK-042 Skip outbound error occurs when the part is very close to the downstream boundary and the internal tracking software has determined that the robot does not have sufficient time to pick up the part before it exits the downstream boundary. Therefore, by default, the motions that would cause the robot to go beyond the downstream boundary are skipped, and the robot moves to pick the next part on the conveyor.

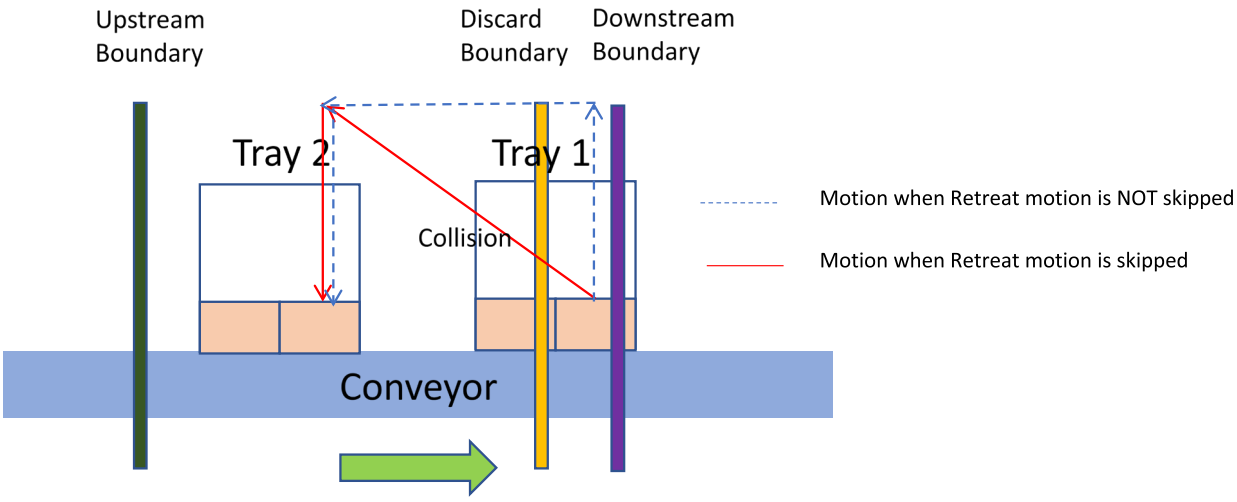


Figure 7.13: Illustration of LNTK-042 Skip outbound error

The TP program PK_SAFE_RETREAT.TP has been provided with J945: *i*RPickTool to move the robot up perpendicular to the conveyor if the retreat motion is skipped. The retreat motion is shown with dotted line in figure 1. PK_SAFE_RETREAT.TP can be called in the tracking pick (PK_CV_PICK11.TP) or tracking place (PK_CV_DROP11.TP) program to avoid this issue.
PK_SAFE_RETREAT.TP program is shown below.

PK_SAFE_RETREAT.TP takes two arguments

Argument 1:
Utool number.

Argument 2:
Tool_offset that is used to safely retreat the robot.

When using PK_SAFE_RETREAT.TP, modify the program PK_CV_PICK11.TP if the Skip Outbound error occurs on the infeed conveyor, or PK_CV_DROP11.TP if the Skip Outbound error occurs while placing the part.
PK_SAFE_RETREAT.TP is available only with J945: *i*RPickTool.

The example below shows the usage of the PK_SAFE_RETREAT.TP program in the tracking pick program PK_CV_PICK11.TP. Note that if the LNTK-042 Skip Outbound error occurs then the Flag F[R[102]] comes on. You can call PK_SAFE_RETREAT.TP when the Flag turns on.

```
 71:  -- Move to pick pos
 72:L  PR[57:Pkl Cv Ref Pos] max_speed
   :  CNT0 VOFFSET,VR[1] TB R[125]sec,
   :  CALL PKGRCLOSE(
   :  Gripper ID=R[104:Gripper Id Gl],
   :  Start Zone=R[107:Counter Gl],
   :  End Zone=R[106:Gr EndZone Gl])
   :
 73:  -- Chk Skip Pick
 74:  IF (F[R[102]]),
   :  CALL PK_SAFE_RETREAT(
   :  UT num=R[107:Counter Gl],
   :  TOffst PR=56)
```

```
 87:  -- Move to retreat
 88:L  PR[57:Pkl Cv Ref Pos] max_speed
   :  CNT100 VOFFSET,VR[1]
   :  Tool_Offset,PR[56:Pkl Cv Ap Ofst]
   :
 89:  IF (F[2]),CALL PK_SAFE_RETREAT(
   :  UT num=R[107:Counter Gl],
   :  TOffst PR=56)
 90:  IF R[117:Pklx PPStat]<>0,
   :  JMP LBL[1500]
 91:  JMP LBL[2000]
```

7.14 Fixed approach height for trays with multiple layers of cells

When placing a part to a tray or box with multiple layers, the approach height will change as the robot moves to higher layers as shown in Figure 7.14(a). The additional higher motion can become inefficient for tall trays. Also, robot motion limit errors can also occur which would pause the robot.

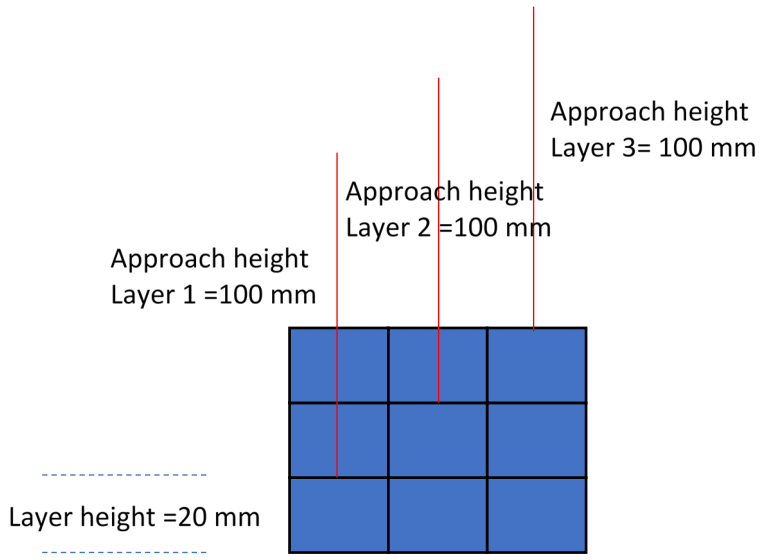


Figure 7.14(a): The approach height the robot has to go to increases with each higher layer

The users may use the following KAREL programs to move the robot to a fixed approach height regardless of the number of layers in the tray.

- PKCSFIXAPRZ.PC
  For multi-layer trays on a conveyor.
- PKFSFIXAPRZ.PC
  For multi-layer trays on a fixed station.

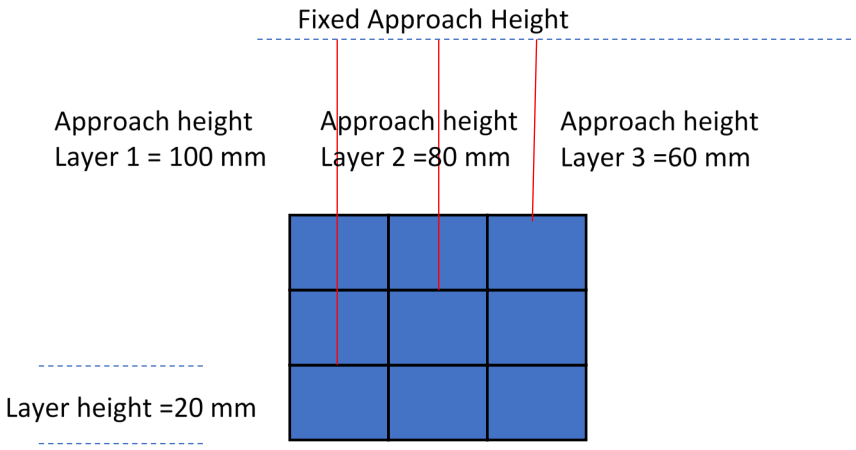


Figure 7.14(b): Fixed Approach Height

This function is available only with J945: *i*RPickTool.

7.14.1 PKCSFIXAPRZ.PC

The program PKCSFIXAPRZ.PC can be called from the conveyor picking program PK_CV_PICK11.TP or in the drop program PK_CV_DROP11.TP.

Argument 1:

INTEGER. The ID of the conveyor station.

Argument 2:

Offset type. "Tool Offset" is set as a default and usually it does not need modification. Offset is applied relative to the UTOOL with the motion to the approach position in PK_CV_PICK11.TP or PK_CV_DROP11.TP.

NOTE
To apply the offset to the tracking frame with the motion to the approach position in PK_CV_PICK11.TP or PK_CV_DROP11.TP, modify the Argument 2 to "Offset". In addition, check and modify the TP program so that the offset can be applied to the tracking frame properly.

Argument 3:

INTEGER. The processing status of the PKCSGETQUE.

Argument 4:

INTEGER. If non-zero, then this function did not work correctly.

Example:

```
CALL PKCSFIXAPRZ(CStn ID=R[110:Pk1x StndId], Tool Offset, GetQ Stat = R[123: Pk1x
GetQ stat], Stat Reg=1)
```

Example of using in the PK_CV_PICK11.TP program. PKCSFIXAPRZ needs to be called after the PKCSGETQUE call. You also should call it before the move to the approach position.

```
 46:  LBL[80]
 47:  CALL PKCSFIXAPRZ(CStn ID=R[110:Pk1x StnId],Tool Offset,
   :  GetQ Stat=R[123:Pk1x GetQ stat],Stat Reg=1)
 48:  -- Set Utool
 49:  UTOOL_NUM=R[107:Counter G1]
 50:  IF R[115:Pk1x UsePosChk]=0,JMP LBL[85]
 51:  --Check if tracking position is reachable
 52:  CALL PKCSCHKPOS(CStn ID=R[110:Pk1x StnId],Offset VR=1,Ref Pos PR=57,
   :  Appr Ofst PR=56,Tool Offset,Stat Reg=108)
 53:  IF R[108:PosNotReach G1]<>0,JMP LBL[1000]
 54:
 55:  LBL[85]
 56:  -- Move to approach
```

NOTE
When using PKCSFIXAPRZ.PC, set [X], [Y], [W], [P], and [R] of [Approach Offset] to zero in the conveyor station operation menu.

7.14.2 PKFSFIXAPRZ.PC

The program PKFSFIXAPRZ.PC can be called from the fixed station picking program PK_FS_PICK11.TP or in the drop program PK_FS_DROP11.TP

Argument 1:
INTEGER. The ID of the fixed station.

Argument 2:
Offset type. “Tool Offset” is set as a default and usually it does not need modification. Offset is applied relative to the UTOOL with the motion to the approach position in PK_CV_PICK11.TP or PK_CV_DROP11.TP.

NOTE
To apply the offset to the UFrame of the fixed station with the motion to the approach position in PK_CV_PICK11.TP or PK_CV_DROP11.TP, modify the Argument 2 to “Offset”. In addition, check and modify the TP program so that the offset can be applied to the UFrame of fixed station properly.

Argument 3:
INTEGER. If non-zero, then this function did not work correctly.

Example:

```
CALL PKFSFIXAPRZ(FStn ID=R[110:Pk1x StndId], Tool Offset, Stat Reg=1)
```

Example of using in the PK_FS_PICK11.TP program. PKFSFIXAPRZ needs to be called before the move to the approach position.

```
36:  LBL[300]
37:  UTOOL_NUM=R[107:Counter G1]
38:  -- Set Pick/drop registers
39:  CALL PK_GR_GET_ENDZONE1(PickDropCounter=R[107:Counter G1],
   :  Multiple=R[132:Dp1x DpMult],Quantity=R[131:Dp1x Qty],End Zone Reg=106)
40:  CALL PKFSFIXAPRZ(FStn ID=R[130:Dp1x StnId],Tool Offset,Stat Reg=1)
41:  -- Move to approach
42:L  PR[63:Dp1 FS Ref Pos] max_speed CNT100 VOFFSET,VR[2]
   :  Tool_Offset,PR[62:Dp1 FS Ap Ofst]
43:  -- Move to drop pos
```

NOTE
When using PKFSFIXAPRZ.PC, set [X], [Y], [W], [P], and [R] of [Approach Offset] to zero in the fixed station operation menu.

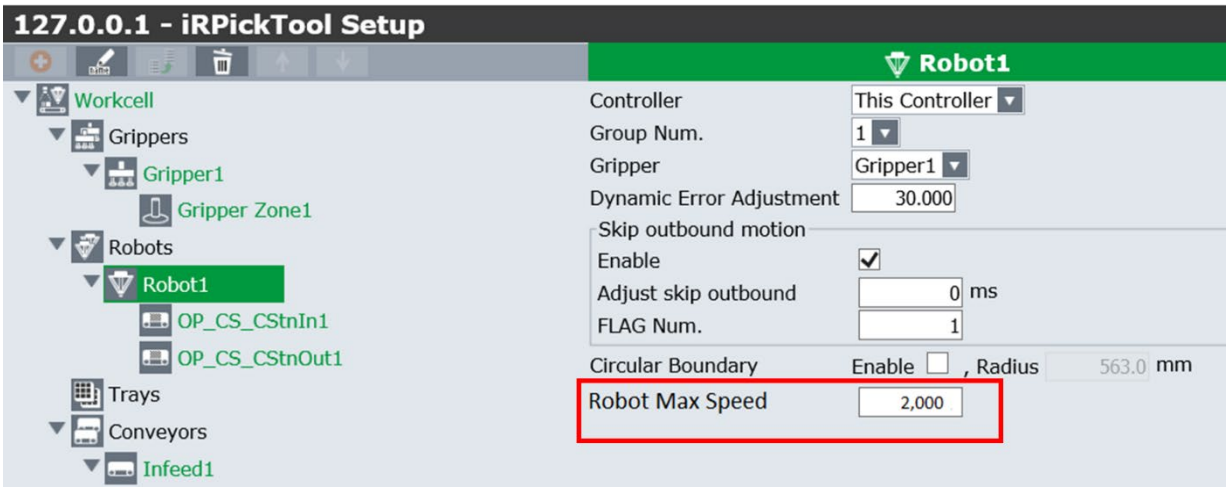
7.15 *i*RPickTool robot maximum speed specification to prevent duty failures

The J945: *i*RPickTool TP programs use the max_speed instruction for achieving the highest speed motions. An example of using max_speed is shown below.

```
86:L PR[57:Pk1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[1]
     Tool_Offset,PR[56:Pk1 Cv Ap Ofst]      ;
```

In *i*RPickTool high-speed applications users may run into duty cycle issues such as overheat alarms (which will stop the robot and require a cycle power) while running the robot at default max_speed for very long periods. To prevent the duty errors, you can set the proper value of the "Robot Max Speed" in the *i*RPickTool "Robot" object as shown in the figure below. Once it is set, this value will be applicable to all the TP programs that use the max_speed instruction.
For Delta robots, the default value of "Robot Max Speed" is 10,000 mm/s.
For SCARA, LRMate and other serial link robots the default "Robot Max Speed" is 2000 mm/s.

The value of the "Robot Max Speed" is also saved to the Recipes.



8 EXAMPLES OF SETUP IN ACCORDANCE WITH USE

This chapter presents specific examples of setup in accordance with the use of *i*RPickTool Basic.

8.1 USING DIFFERENT ROBOTS TO PICK PARTS DEPENDING ON THEIR POSITIONS ON THE CONVEYOR

This section describes the case in which different robots are used to pick parts depending on their positions on the conveyor. In the example presented here, there are two robots, and the upstream robot picks parts on the right half of the conveyor while the downstream robot picks parts on the left half. A camera is used to detect parts on the conveyor.

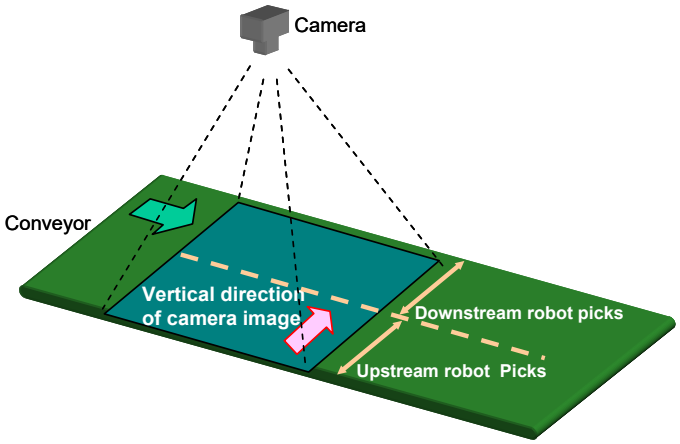


Fig.8.1 (a) Example of setup where the robot that picks a part up is changed depending on the position of the part on the conveyor

In addition, the following assumptions are made.

- The conveyor configuration is defined as shown below.
  This is the same configuration as Configuration 1 shown in Section 2.4, "SAMPLE SYSTEM CONFIGURATIONS".

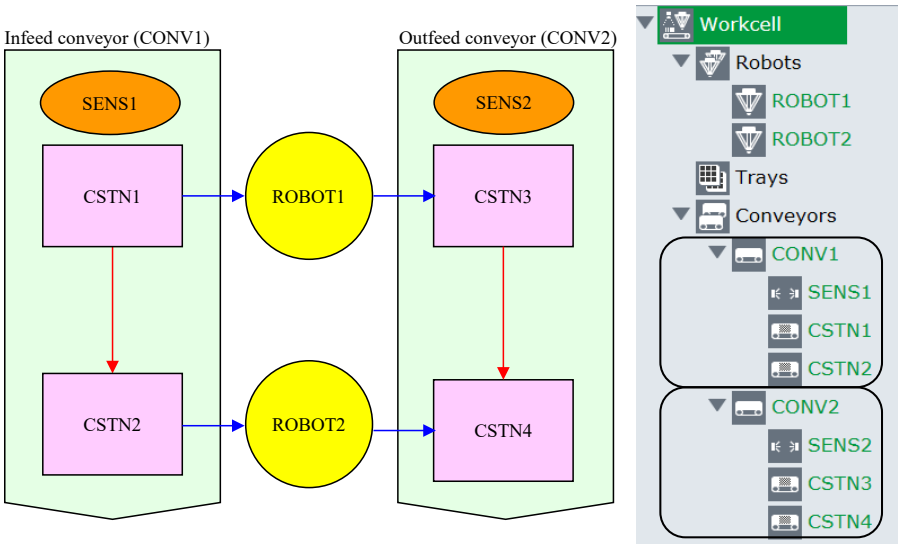
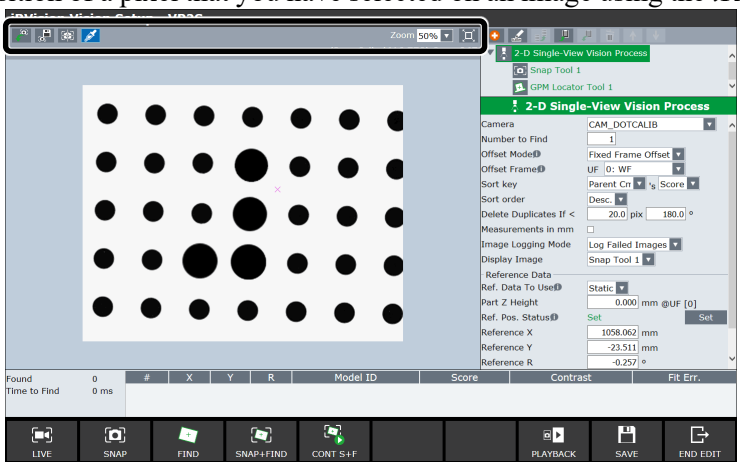


Fig. 8.1 (b) Conveyor configuration example

- The camera is installed so that the horizontal direction of the camera image is parallel to the moving direction of the conveyor. That is, the width direction of the conveyor is aligned with the vertical direction of the camera image. In *i*RVision, Vt represents the coordinate in the vertical direction of the camera image.
- The model ID of the part detected by the GPM locator tool is 1. If the Vt coordinate on the detected image is larger than the Vt coordinate of the conveyor center, select 2 for the model ID using a conditional execution tool. (For the Vt coordinate of the conveyor center, check using 'Sampling Tool'.)

Checking the Vt coordinates of pixels with *i*RVision

You can check the position of a pixel that you have selected on an image using the *i*RVision sampling tool.

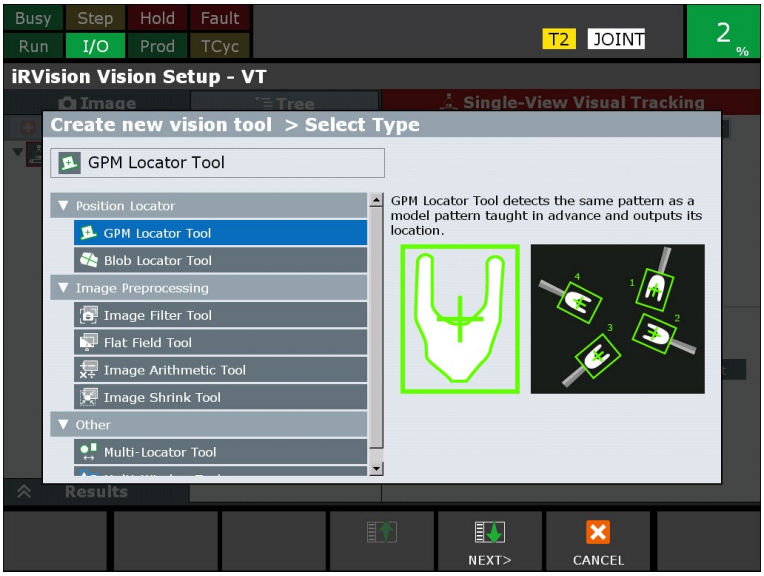


1 Click the button
The sampling tool will be enabled, and ✕ will be displayed at the upper left.
2 Click any pixel on the image.
✕The mark moves and the position of the pixel (Vt, Hz) will be displayed at the upper right.

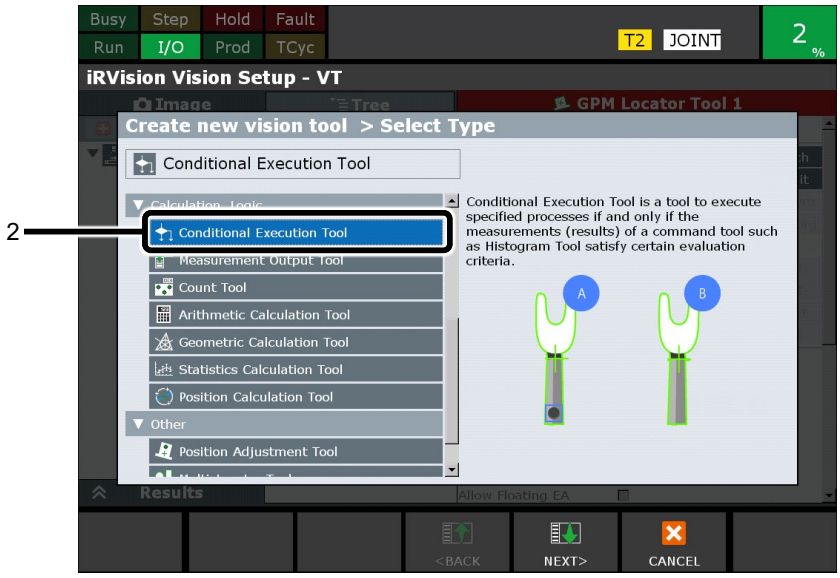
Inserting the conditional execution tool

Insert the conditional execution tool into the vision process.

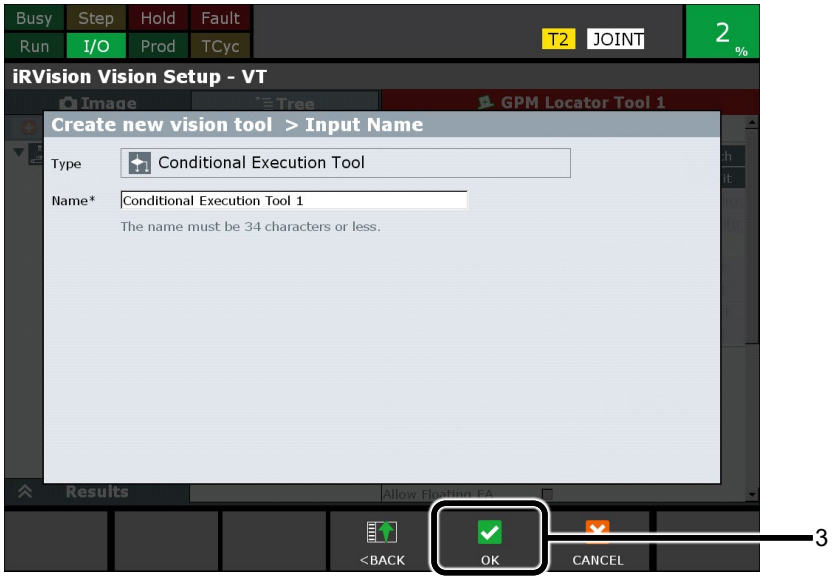
1 In the tree view of the edit screen, select [GPM Locator Tool 1] and click the button. A screen as shown below is displayed.



2 Select [Conditional Execution Tool] and press F4 [NEXT].



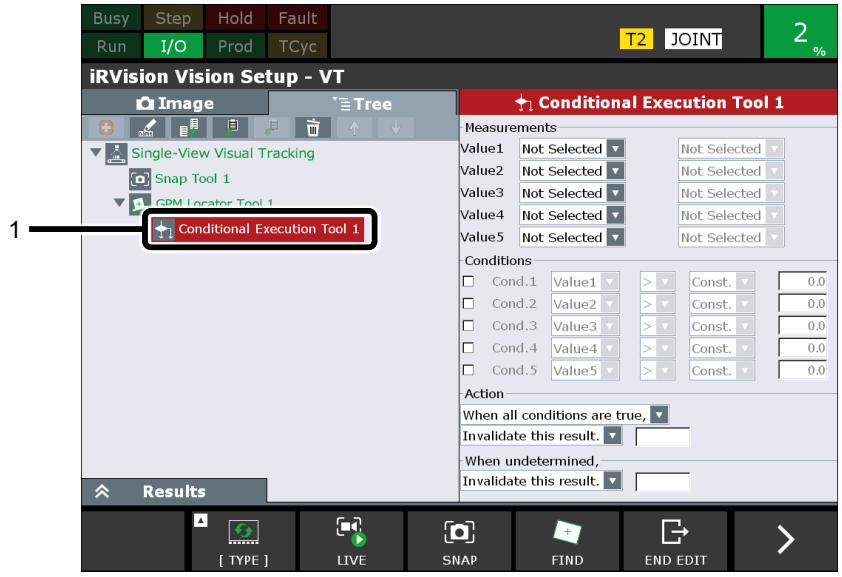
3 In [Name], enter a tool name (e.g., “Conditional Execution Tool 1”), and press F4 [OK] key.



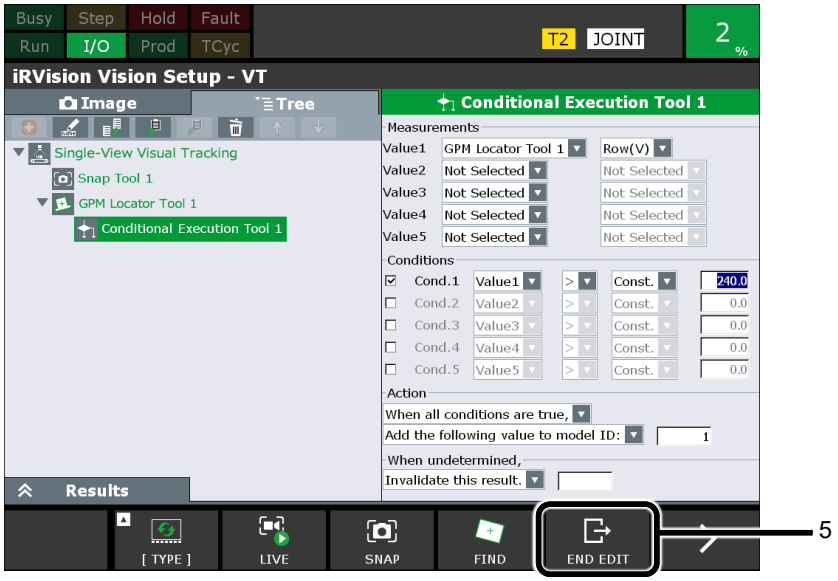
Setting the conditional execution tool

Set it so that if the Vt coordinate is larger than the Vt coordinate of the conveyor center (here, the explanation supposes it to be 240), 1 is added to the model ID.

1 In the tree view, select [Conditional Execution Tool1]. A screen as shown below is displayed.



2 Under [Measurements], select [GPM Locator Tool 1] and [Row(V)] next to [Value1].
3 Under[Conditions], check [Cond.1] and select [Value1], [>], [Const.], and [240].
4 Under [Action], select [Add the following value to model ID], and specify 1.
5 Press F5 [END EDIT] .



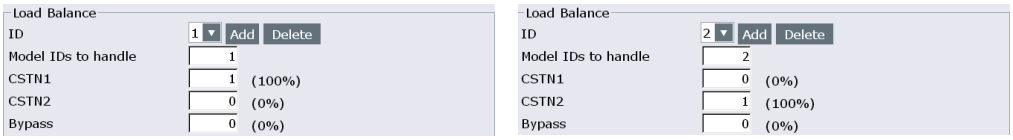
6 Press [> (NEXT)] to display the next function menu, and press F5 [SAVE].

Setting the load balance

Set the load balance of the conveyor so that the upstream robot handles all parts whose model ID is 1 and the downstream robot handles all parts whose model ID is 2.
Open the conveyor setup screen, and set the load balance. Here, only the infeed conveyor is shown.

1 Open the conveyor setup screen. A screen as shown below is displayed.
2 Set [Self-paced] or [Keep rate] in the left selection box next to [Load Balance] and [For each model ID] in the right selection box.

3 In [Model IDs to handle], specify 1 and 2.
4 Set the load balance so that the upstream robot handles all parts whose model ID is 1 and the downstream robot handles all parts whose model ID is 2.



8.2 USING DIFFERENT ROBOTS TO PICK PARTS DEPENDING ON THEIR CHARACTERISTICS

This section describes the case in which the models of individual parts are identified from differences in their characteristics and different robots are used to pick parts depending on the model. In the example presented here, parts of two models - one with a hole and the other without a hole - are conveyed and allocated to different robots based on whether a hole is present or not. The following figure assumes that the target parts are nuts of two models.

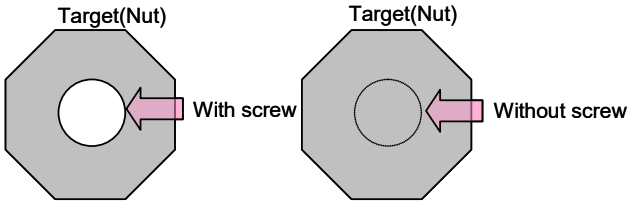


Fig. 8.2 (a) Example of a partial characteristic of a part (a part with and without a hole in one section)

In addition, the following assumptions are made.

- Whether a hole is present is determined by measuring the standard deviation of brightness with a histogram tool. If a nut has an even surface, the standard deviation of brightness becomes large when a hole is present and becomes small when a hole is not present.

	With a hole	Without a hole
Standard deviation	Large	Small

⚠ CAUTION
The tendency described above may not apply depending on differences in the background and other factors. The information provided here is for reference purposes only.

NOTE
Whether a hole is present can also be determined using the blob tool or GPM locator tool. Here, the method that uses a histogram tool is described as an example.

- The part model ID taught to the GPM locator tool is 1.
- If a hole is present, the model ID is set to 2 using the conditional execution tool.
- The conveyor configuration is defined as shown below.
  This is the same configuration as Configuration 1 shown in Section 2.4, "SAMPLE SYSTEM CONFIGURATIONS".

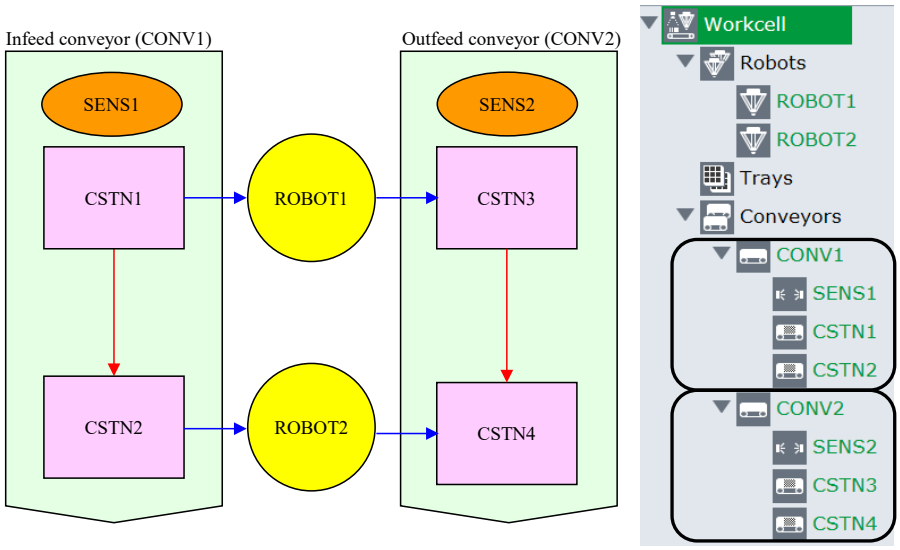
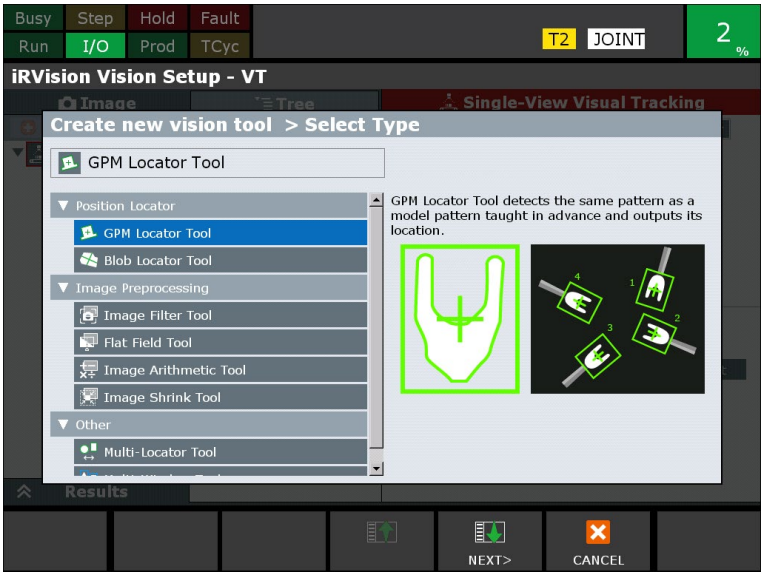


Fig. 8.2 (b) Conveyor configuration example

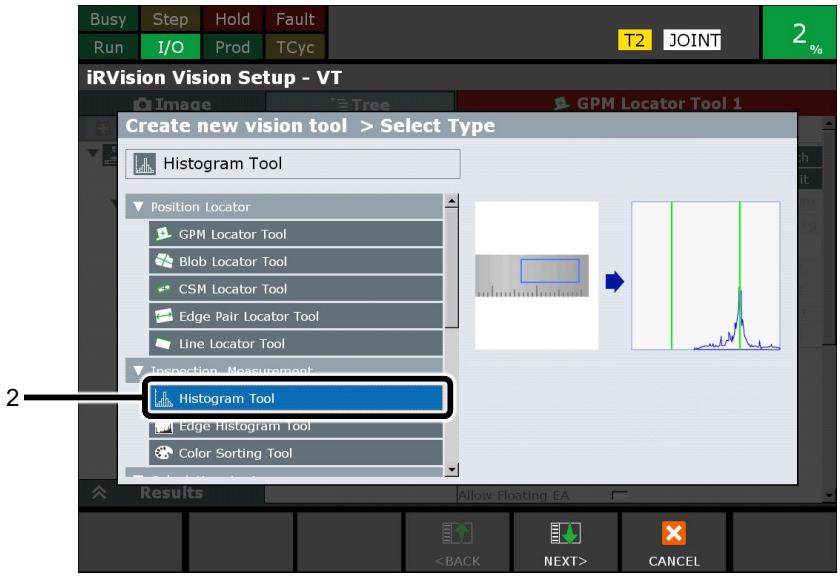
Inserting the histogram tool

Insert the histogram tool to the vision process.

1 In the tree view of the edit screen, select [GPM Locator Tool 1] and click the button. A screen as shown below is displayed.



2 Select [Histogram Tool] and press F4 [NEXT].



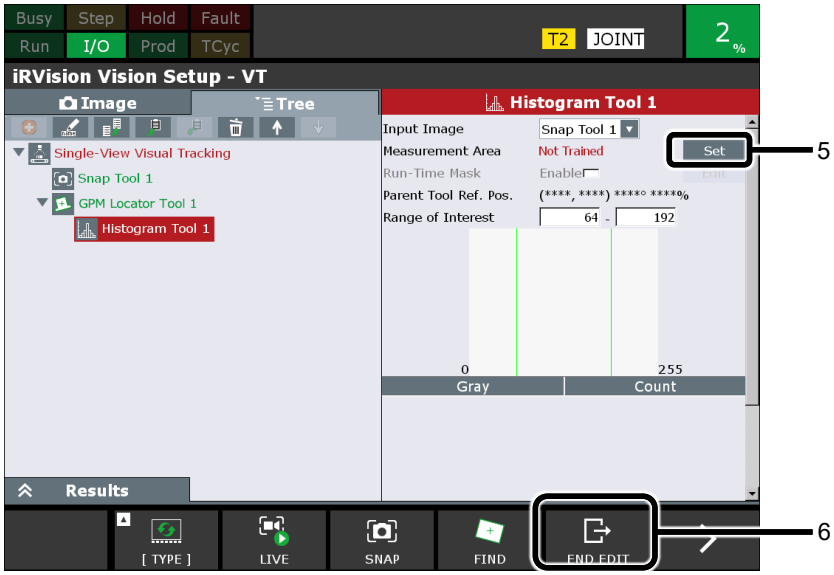
3 In [Name], enter a tool name (e.g., “Histogram Tool 1”), and press F4 [OK] key.

[Histogram Tool 1] will be displayed in the tree view.

4 In the tree view, select [Histogram Tool 1].

5 Click the [SET] button to teach [Measurement Area].

6 Press F5 [END EDIT] key.



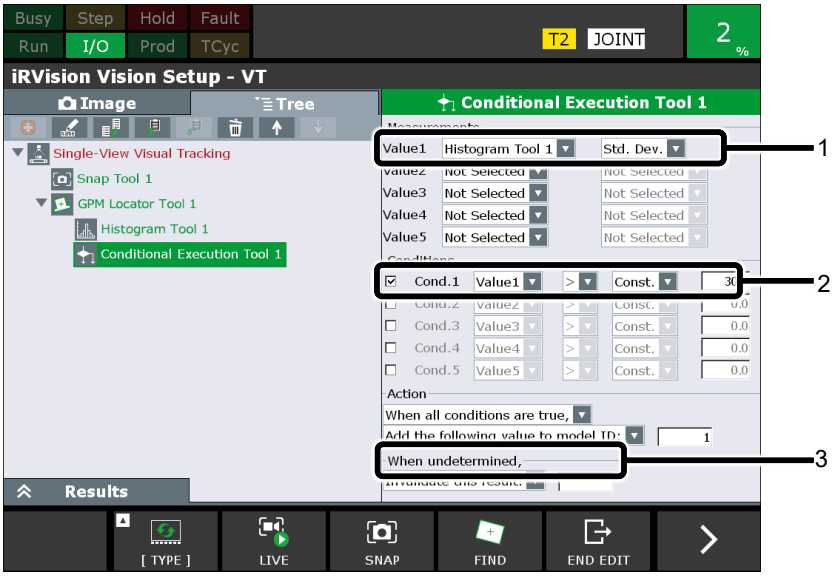
7 Press [> (NEXT)] to display the next function menu, and press F5 [SAVE].

Setting the conditional execution tool

Set the conditional execution tool so that it sets the model ID to 2 if the part has a hole. For information about how to insert the conditional execution tool, see Section 8.1, “USING DIFFERENT ROBOTS TO PICK PARTS DEPENDING ON THEIR POSITIONS ON THE CONVEYOR”.

If the standard deviation is 50 or so when a hole is present and 10 or so when a hole is not present, an appropriate threshold is considered to be a value between them, e.g., 30. Set the conditional execution tool so that it adds 1 to the model ID if the standard deviation of the histogram is 30 or more.

1 Under [Measurements], select [Histogram Tool 1] and [Std. Dev.] for [Value1].
2 Under [Conditions], check [Cond.1] and then select [Value1], [>], [Const.], and [30.0].
3 Under [Action], select [Add the following value to model ID] and specify 1.



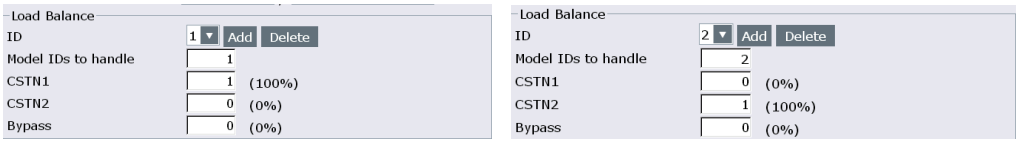
Setting the load balance

Set the load balance of the conveyor so that the upstream robot handles all parts whose model ID is 1 and the downstream robot handles all parts whose model ID is 2.

Open the conveyor setup screen, and set the load balance. Here, only the infeed conveyor is shown.

1 Open the conveyor setup screen. A screen as shown below is displayed.
2 Set [Self-paced] or [Keep rate] in the left selection box next to [Load Balance] and [For each model ID] in the right selection box.

3 In [Model IDs to handle], specify 1 and 2.
4 Set the load balance so that the upstream robot handles all parts whose model ID is 1 and the downstream robot handles all parts whose model ID is 2.



8.3 STACKING PARTS IN MULTIPLE LAYERS IN A BOX

This refers to cases in which parts are stacked in multiple layers in a box.
This section describes the case in which parts are placed in two layers in a box where each layer contains four parts. While the example presented here uses a tray on the conveyor, the same procedure also applies when a fixed station is used.

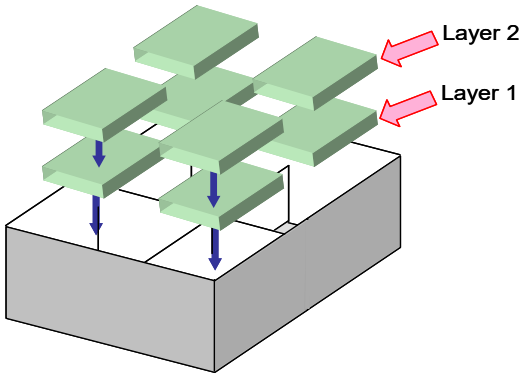


Fig. 8.3 (a) Example of stacking parts in two layers, in a box that can hold four parts in one layer

In this case, no part can be placed in the upper layer until the lower layer is filled with parts. To observe this rule, it is necessary to use the "layer" function of the tray to prevent a part from being placed in the upper layer until the lower layer is filled with parts.

It is assumed that the conveyor configuration is defined as shown below. This is the same configuration as Configuration 1 shown in Section 2.4, "SAMPLE SYSTEM CONFIGURATIONS".

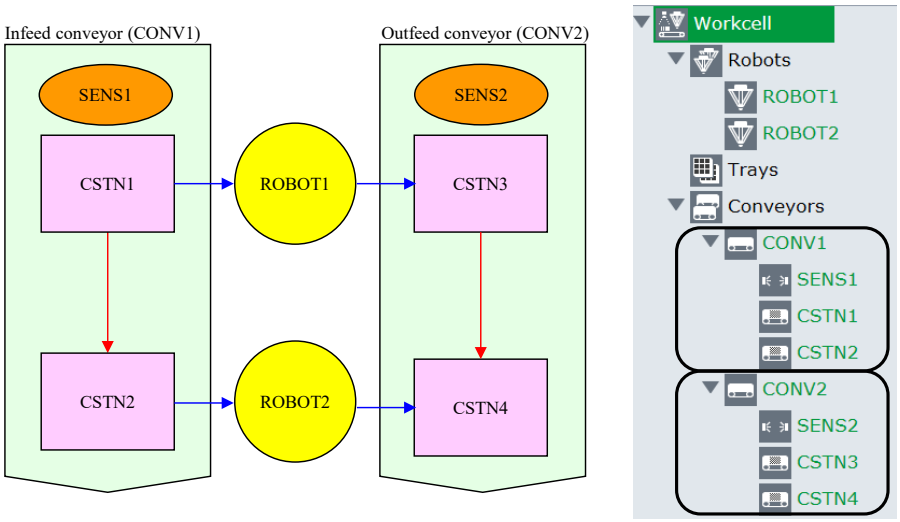
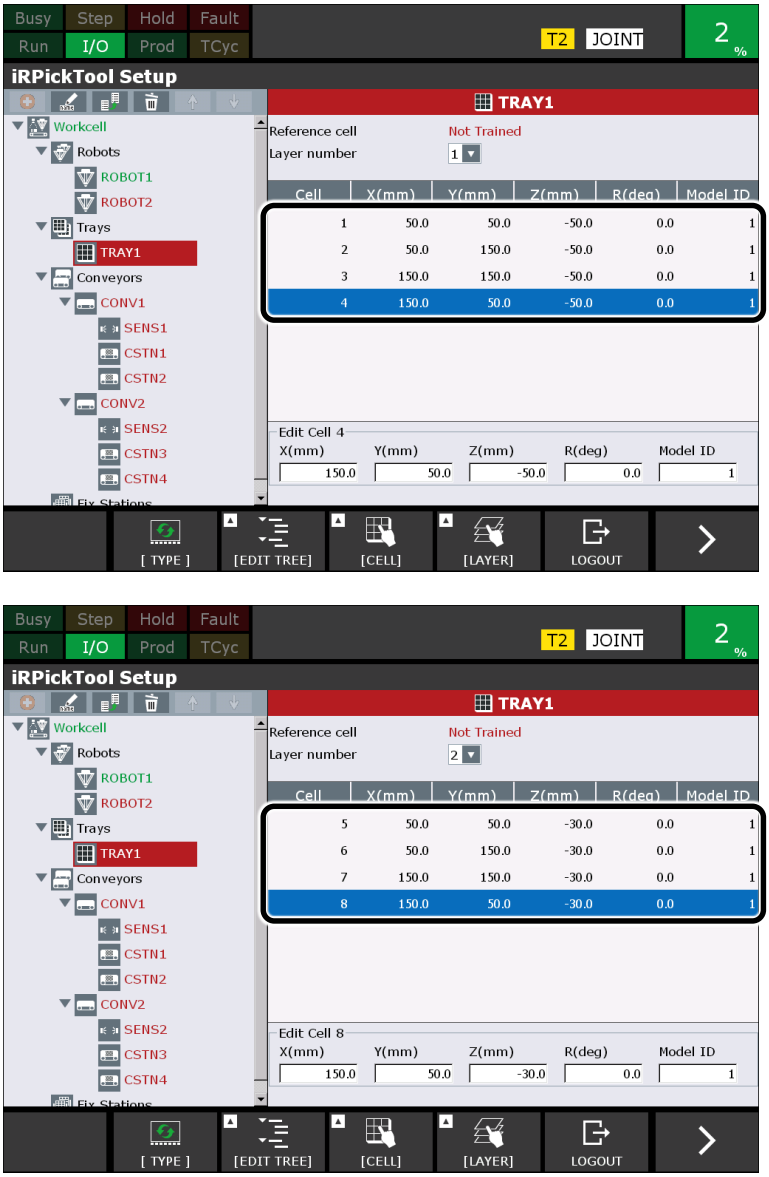


Fig. 8.3 (b) Conveyor configuration example

Setting the tray

Set the tray as shown below.

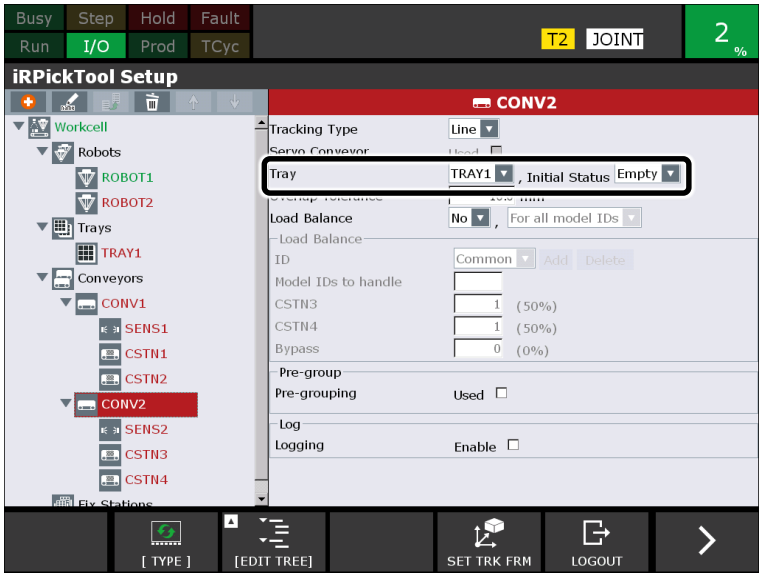


Cells 1 to 4 with layer number 1 belong to layer 1, and cells 5 to 8 with layer number 2 belong to layer 2. If you set the tray as shown above, the robot does not place a part in cells 5 to 8 of layer 2 until all the cells of layer 1 are filled.

Assign all the cells of the same layer to the same layer number. This way, parts are placed in the cells of the same layer, beginning with the downstream cells.

Setting the conveyor

Set the conveyor as shown below. Here, only the outfeed conveyor is shown. The settings for the infeed conveyor do not need to be changed.



In [Tray], select the tray you have set. To fill the tray with parts, select [Empty] for [Initial Status]. If you disable the load balance, the robot places as many parts as possible in the cells while respecting the layer settings.

NOTE

While the load balance is disabled in this example, you may have it enabled. In that case, the cell information is allocated so as to maintain the load balance while at the same time respecting the layer settings.

8.4 PLACING PARTS IN SEQUENCE FROM THE END OF THE BOX

This section describes the case in which parts are placed in sequence from the end of an unpartitioned box. While the example presented here uses a tray on the conveyor, the same procedure also applies when a fixed station is used.

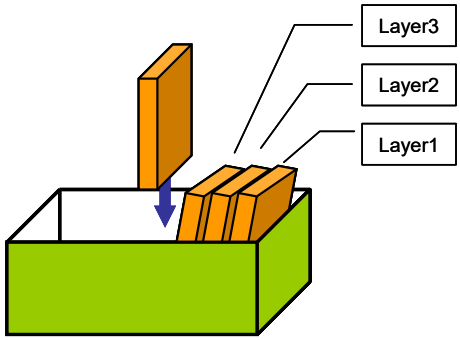


Fig. 8.4(a) Example of placing parts in a box, in sequence from the end

In cases like this, set the tray using its "layer" function to prevent a part from being placed in the adjacent cell until the cell at the end is filled with a part.

Example of setting the tray

In the case mentioned above, each layer of the tray may be set as shown below.

TRAY1

Reference cell Not Trained
Layer number 1

Cell	X(mm)	Y(mm)	Z(mm)	R(deg)	Model ID
1	80.0	20.0	-5.0	0.0	1

Layer number 2

Cell	X(mm)	Y(mm)	Z(mm)	R(deg)	Model ID
2	60.0	20.0	-5.0	0.0	1

Layer number 3

Cell	X(mm)	Y(mm)	Z(mm)	R(deg)	Model ID
3	40.0	20.0	-5.0	0.0	1

Cells 1 to 3 are added to different layers, respectively. Cell 1 is at the end. If you set the tray as shown above, parts are placed sequentially in cells 1, 2, and 3 of the tray when [Empty] is selected for [Initial Status] for the conveyor.

Example of setting the conveyor

Set the conveyor in the same way as described in Section 8.3, "STACKING PARTS IN MULTIPLE LAYERS IN A BOX".
For details, refer to Section 8.3 "STACKING PARTS IN MULTIPLE LAYERS IN A BOX."

8.5 PLACING PARTS IN DIFFERENT DIRECTIONS IN A BOX

This section describes the case in which parts are placed in different directions in difference cells in a box. In the example shown below, the direction of cells 2 and 4 is 90 degrees different from the other two cells.

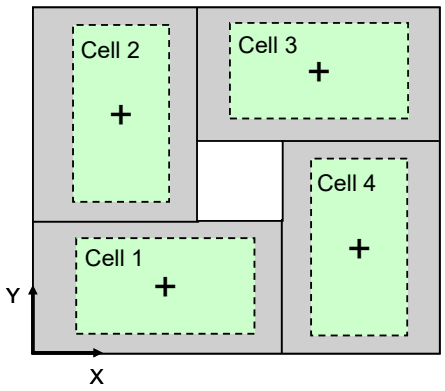
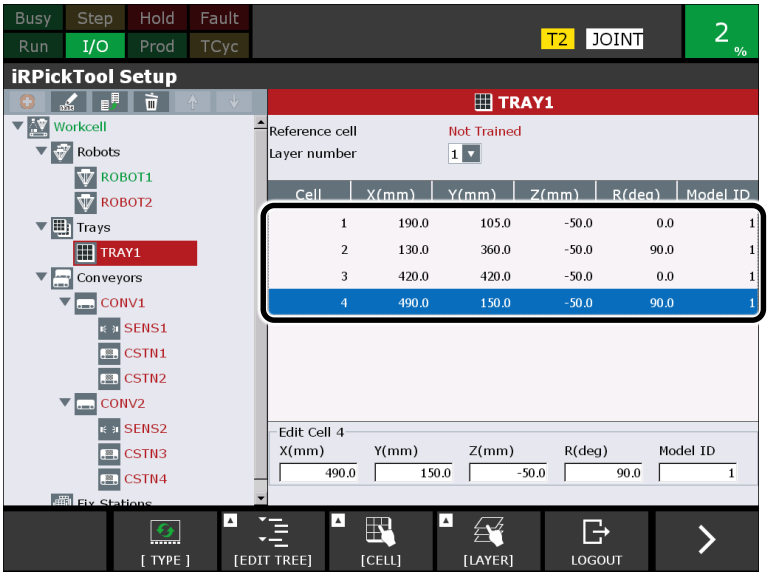


Fig. 8.5(a) Example of when the directions of the parts placed in a box are different depending on the cell

In this case, you need to change only the tray settings.

Setting the tray

An example of setting the tray is given below.



The angle of cell 1 is 0 degrees, and the angle of each other cell is relative to that of cell 1. The angle of cells 2 and 4 is 90 degrees.

Note that robot position teaching is performed only with cell 1, as in the case of a normal tray. Since the tray settings indicate that the direction of cells 2 and 4 is 90 degrees different, the robot automatically changes the direction by 90 degrees when placing a part in cell 2 or 4.

8.6 CHANGING THE PLACEMENT POSITION DEPENDING ON THE PART TYPE

This section describes the case of a system that places picked parts in a box in which parts of two models are placed in different predetermined places in the box depending on their model IDs. While the example presented here uses a tray on the conveyor, the same procedure also applies when a fixed station is used.

Consider that two part models (one is large and the other is small) are used with two types of boxes (one is the cell for storing the big part and the other is the cell for storing the small part), as in the example shown below. In this case, the cell in which a picked part is placed is determined by the model of that part.

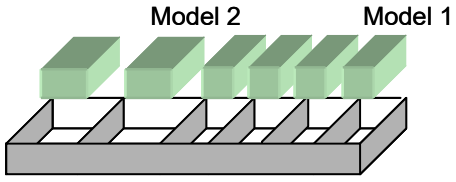


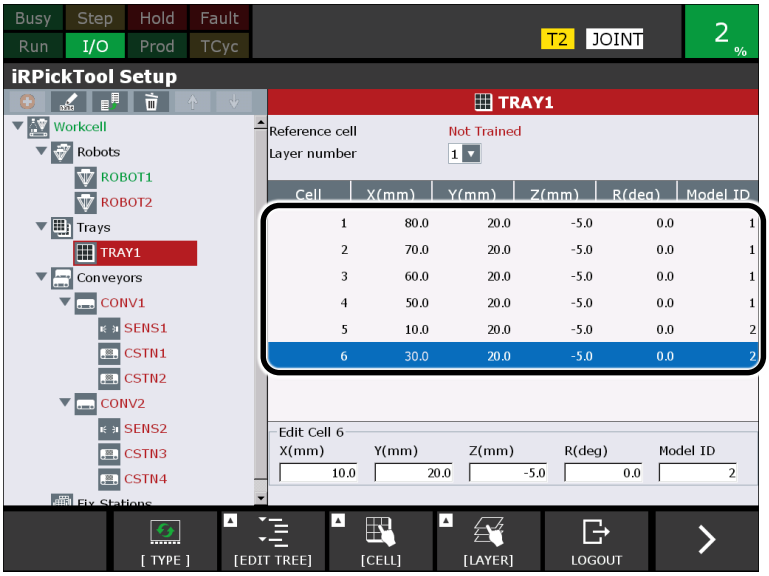
Fig. 8.6(a) Example of changing the placement position depending on the part type (large/small)

In cases like this, set the model IDs when setting the tray. Also, specify the model IDs of the cells you want to be allocated when PKCSGETQUE is called.

Here, it is assumed that you set the model ID of the small part to 1 and the model ID of the large one to 2 for the infeed conveyor.

Setting the tray

An example of setting the tray is given below.



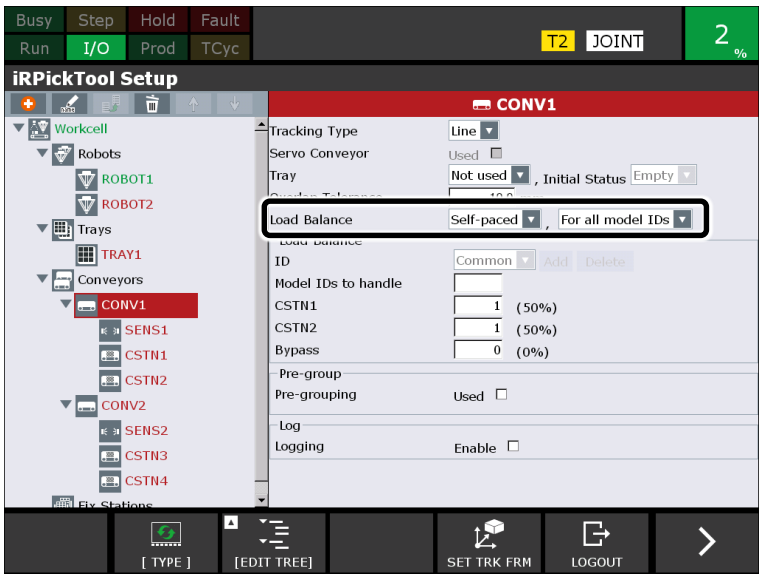
Cells 1 to 4 are for small parts, and cells 5 and 6 are for big parts.
The model IDs of the cells are set to 1 and 2, as with the part model IDs for the infeed conveyor. Parts with model ID 1 are placed in the cells with model ID 1, and parts with model ID 2 are placed in the cells with model ID 2. The model IDs of cells do not necessarily need to match those of parts. However, this makes the model IDs of parts identical to those of the corresponding cells, thus making it easier to identify them.

Add all cells to the same layer. If you add any cell to a different layer, a cell with model ID 1 may not be able to be allocated due to the layer setting constraint when a cell with that model ID is required.

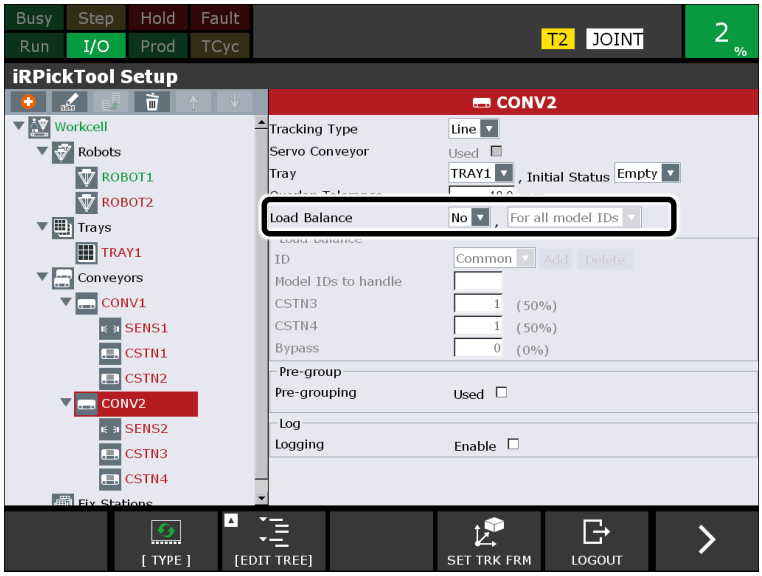
Setting the conveyor

An example of setting the conveyor is given below.

Infeed conveyor



Outfeed conveyor



Since the load balance for the infeed conveyor is set to 1:1 by selecting [For all model IDs] in the right selection box next to [Load Balance], parts of two models mix with one another when they are conveyed, but each robot handles an equal number of parts of either model.

The load balance is disabled for the outfeed conveyor. Therefore, each robot waits for the cell corresponding to the model ID of a picked part as they attempt to place parts in as many places as possible.

Example of modifying the program

In this case, it is necessary to customize the program for the placement motion of the robot. The main program and the pick program do not need to be modified.

Modify the program shown in Subsection 6.9.2.3, “Placement program” as follows. Major changes are underlined.

DROP1.TP

```
  1:    UTOOL_NUM=0
  2:    UFRAME_NUM=0
  3:
  4:    LBL[100]
  5:    STOP_TRACKING
  6:    R[5:MODELID]=VR[1].MODELID
  7:    CALL PKCSGETQUE(R[12:CSTN2], 1, 100, 2, 2, R[5:MODELID])
  8:    IF R[2:GETQ_STATUS]=0,JMP LBL[200]
  9:    IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 10:    JMP LBL[100]
 11:
 12:    LBL[200]
 13: L PR[11] 4000mm/sec CNT100 VOFFSET,VR[2] Tool_Offset,PR[12]
 14: L PR[11] 4000mm/sec CNT3 VOFFSET,VR[2]
 15:    CALL VACUUM_OFF
 16: L PR[11] 4000mm/sec CNT100 VOFFSET,VR[2] Tool_Offset,PR[12]
 17:    CALL PKCSACKQUE(CStn ID=R[12:CSTN2],Success)
 18:
 19:    LBL[900]
[End]
```

On line 6, the model ID of the part picked from the infeed conveyor is stored in R[5], and an argument is added to PKCSGETQUE on line 7 to specify R[5] as a model ID. This way, cells having the model ID specified in R[5] are always allocated. Note that because the argument 6 (=model ID) is specified in PKCSGETQUE, the argument guidance is not displayed on line 7.

> **⚠ CAUTION**
> If there is no cell corresponding to a picked part, the robot cannot place the part that it holds. In order to make this type of system operate normally, the ratio of parts 1 and 2 conveyed on the infeed conveyor needs to match the ratio of parts 1 and 2 conveyed on the outfeed conveyor.

8.7 PLACING PARTS ON DIFFERENT OUTFEED CONVEYORS DEPENDING ON THE MODEL ID

This section describes the case in which there is more than one outfeed conveyor and which outfeed conveyor to use is determined by the model of the part picked from the infeed conveyor.

Consider the case in which the system has one infeed conveyor and two outfeed conveyors, with each conveyor having one conveyor station, as shown in the figure below.

Parts of two models (model IDs 1 and 2) are conveyed from the infeed conveyor. Parts of model ID 1 are placed on outfeed conveyor 1, and those of model ID 2 are placed on outfeed conveyor 2.

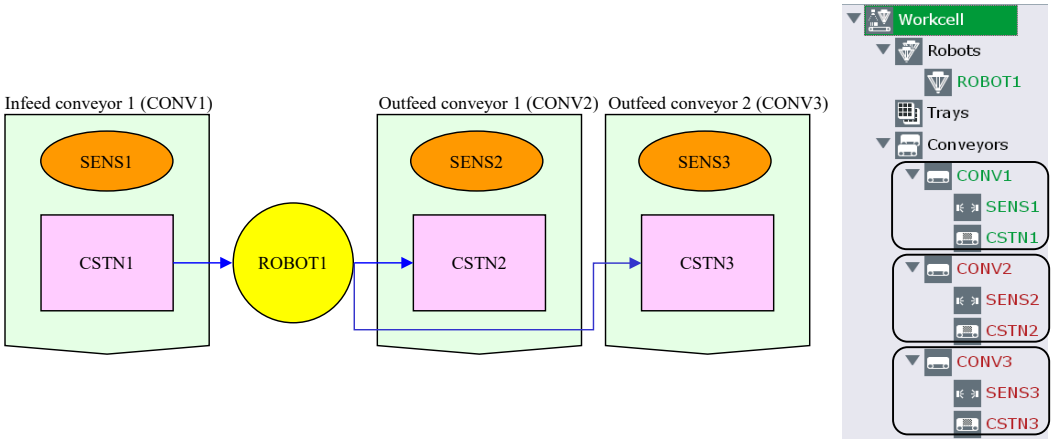


Fig. 8.7(a) Example of placing parts on different outfeed conveyors depending on the part type (large/small)

Setting the conveyor

Set the infeed conveyor so that all parts are to be picked from it regardless of the model ID. To do so, disable the load balance. Disable the load balance for the two outfeed conveyors as well.

Examples of modifying the programs

In this case, it is necessary to customize the main program of the robot. Also, a new placement program for the second outfeed conveyor, DROP2.TP, is required. The pick program does not need to be modified.

Modify the main program, MAIN1.TP, shown in Subsection 6.9.2.1, "Main program" as follows. Major changes are underlined.

MAIN1.TP

```
  1:    R[1:CYCLE_STOP]=0
  2:    CALL PKCSGETID('CONV1',CStn ID Reg=11)
  3:    CALL PKCSGETID('CONV2',CStn ID Reg=12)
  4:    CALL PKCSGETID('CONV3',CStn ID Reg=13)
  5: L P[1] 1000mm/sec FINE
  6:    CALL PKWCSTART
  7:
  8:    LBL[100]
  9:    CALL PICK1
 10:    IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 11:
 12:    R[5:MODELID]=VR[1].MODELID
 13:    IF R[5:MODELID]=2,JMP LBL[200]
 14:    CALL DROP1
 15:    JMP LBL[300]
 16:    LBL[200]
 17:    CALL DROP2
 18:    LBL[300]
 19:    IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 20:    JMP LBL[100]
 21:
 22:    LBL[900]
 23:    CALL PKWCEND
[End]
```

On line 12, the model ID of the part picked from the infeed conveyor is stored in R[5]. If the value of R[5] is 1, the placement program for outfeed conveyor 1, DROP1.TP, is called. If the value is 2, the placement program for outfeed conveyor 2, DROP2.TP, is called.

An example of the placement program for outfeed conveyor 2, DROP2.TP, is shown below.

DROP2.TP

```
  1:    UTOOL_NUM=0
  2:    UFRAME_NUM=0
  3:
  4:    LBL[100]
  5:    STOP_TRACKING
  6:    CALL PKCSGETQUE(CStn ID=R[13:CSTN3],Consec Flag=1,Timeout (ms)=100,
        Offset VR=3,Stat Reg=2)
  7:    IF R[2:GETQ_STATUS]=0,JMP LBL[200]
  8:    IF R[1:CYCLE_STOP]=1,JMP LBL[900]
  9:    JMP LBL[100]
 10:
 11:    LBL[200]
 12: L PR[20] 4000mm/sec CNT100 VOFFSET,VR[3] Tool_Offset,PR[12]
 13: L PR[20] 4000mm/sec CNT3 VOFFSET,VR[3]
 14:    CALL VACUUM_OFF
 15: L PR[20] 4000mm/sec CNT100 VOFFSET,VR[3] Tool_Offset,PR[12]
 16:    CALL PKCSACKQUE(CStn ID=R[13:CSTN3],Success)
 17:
 18:    LBL[900]
[End]
```

Basically, this program is the same as the DROP1.TP program shown in Subsection 6.9.2.3, "Placement program" except for the number of the Position Register storing the robot teaching position. A Position Register of a different number needs to be used for each conveyor. Here, PR[20] is used as the placement position for outfeed conveyor 2.

8.8 PICKING PARTS FROM TWO OR MORE INFEED CONVEYORS

This section describes the case in which there are multiple infeed conveyors and parts are picked from each of them and placed on an outfeed conveyor. This is the same configuration as Configuration 8 shown in Section 2.4, "SAMPLE SYSTEM CONFIGURATIONS".

Consider the case in which the system has two infeed conveyors and one outfeed conveyor, with each conveyor having one conveyor station, as shown in the figure below.

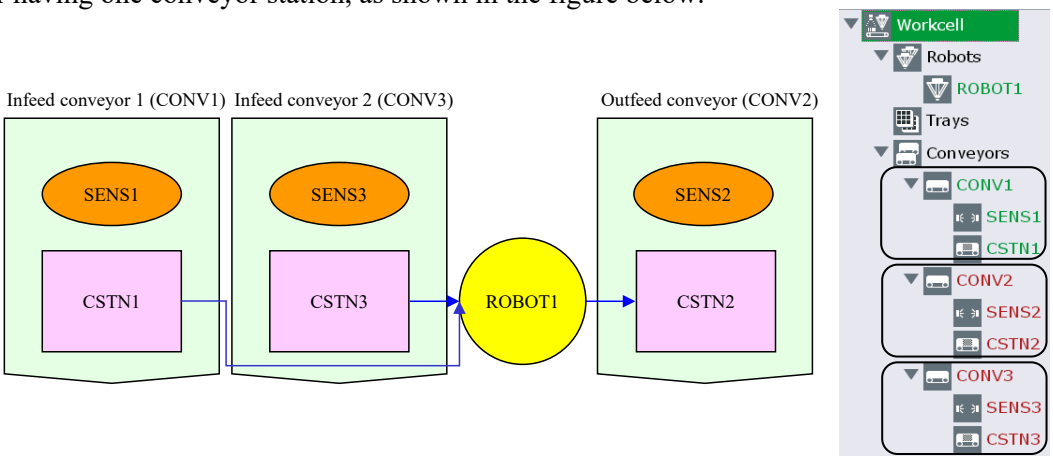


Fig. 8.8(a) Example of picking up parts from multiple infeed conveyors

It is assumed that the robot picks parts alternately from either conveyor. If the robot fails to pick a part from one conveyor, it attempts to pick one from the other.

Example of modifying the main program

In this case, it is necessary to customize the main program and pick program of the robot. Also, a new pick program for the second infeed conveyor is required.

Examples of modifying the main program, MAIN1.TP, shown in Subsection 6.9.2.1, "Main program" and the program, PICK1.TP, shown in Subsection 6.9.2.2, "Pick program" are shown below. Major changes are underlined.

MAIN1.TP

```
  1:    R[1:CYCLE_STOP]=0
  2:    R[3:ZONE]=1
  3:    R[5:SELECT_CNV]=1
  4:    CALL PKCSGETID('CONV1',CStn ID Reg=11)
  5:    CALL PKCSGETID('CONV2',CStn ID Reg=12)
  6:    CALL PKCSGETID('CONV3',CStn ID Reg=13)
  7: L P[1] 1000mm/sec FINE
  8:    CALL PKWCSTART
  9:
 10:    LBL[100]
 11:    STOP_TRACKING
 12:    IF R[5:SELECT_CNV]=2,JMP LBL[200]
```

```
 13:    CALL PICK1
 14:    JMP LBL[300]
 15:    LBL[200]
 16:    CALL PICK2
 17:    LBL[300]
 18:    IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 19:
 20:    R[5: SELECT_CNV]=3-R[5: SELECT_CNV]
 21:
 22:    IF R[3:ZONE]<=R[4:NUM_ZONE] OR R[2:GETQ_STATUS]<>0,JMP LBL[100]
 23:    R[3:ZONE]=1
 24:
 25:    CALL DROP1
 26:    IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 27:    JMP LBL[100]
 28:
 29:    LBL[900]
 30:    CALL PKWCEND
[End]
```

R[5] is used to switch conveyors. If the value of R[5] is 1, a part is picked from infeed conveyor 1. If the value of R[5] is 2, a part is picked from infeed conveyor 2.

NOTE

If both conveyors have a part to pick, the application program determines the conveyor from which to pick a part; it cannot be determined generally. In the above case, the program gives priority to the conveyor different from the one from which a part was picked last.

PICK1.TP is a tracking program for the pickup motion for infeed conveyor 1, and PICK2.TP is a tracking program for the pickup motion for infeed conveyor 2.

PICK1.TP

```
  1:    UTOOL_NUM=1
  2:    UFRAME_NUM=0
  3:
  4:    CALL PKCSGETQUE(CStn ID=R[11:CSTN1],Consec Flag=R[3:ZONE],
        Timeout (ms)=100, Offset VR=1,Stat Reg=2)
  5:    IF R[2:GETQ_STATUS]<>0,JMP LBL[900]
  6:
  7: L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
  8: L PR[R[3:ZONE]] 4000mm/sec CNT3 VOFFSET,VR[1]
  9:    CALL VACUUM_ON
 10: L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
 11:    CALL PKCSACKQUE(CStn ID=R[11:CSTN1],Success)
 12:    R[3:ZONE]=R[3:ZONE]+1
 13:
 14:    LBL[900]
[End]
```

No line has been added to the original PICK1.TP program, while several lines have been deleted. The difference from the original PICK1.TP program is that PICK1.TP ends immediately once PKCSGETQUE on line 4 exits due to a timeout, that is, when there is no part to pick in the conveyor station. This causes PICK2.TP to be called next, resulting in a part being picked from conveyor 2.

PICK2.TP

```
  1:    UTOOL_NUM=1
  2:    UFRAME_NUM=0
  3:
  4:    CALL PKCSGETQUE(CStn ID=R[13:CSTN1],Consec Flag=R[3:ZONE],
        Timeout (ms)=100, Offset VR=3,Stat Reg=2)
  5:    IF R[2:GETQ_STATUS]<>0,JMP LBL[900]
  6:
  7:    R[6:PR_NUM]=R[3:GRIPPER]+20
  8: L PR[R[6:PR_NUM]] 4000mm/sec CNT100 VOFFSET,VR[2] Tool_Offset,PR[10]
  9: L PR[R[6:PR_NUM]] 4000mm/sec CNT3 VOFFSET,VR[2]
 10:    VACUUM_ON
 11: L PR[R[6:PR_NUM]] 4000mm/sec CNT100 VOFFSET,VR[2] Tool_Offset,PR[10]
 12:    CALL PKCSGETQUE(CStn ID=R[13:CSTN3],Success)
 13:    R[3:ZONE]=R[3:ZONE]+1
 14:
 15:    LBL[900]
[End]
```

Basically, this program is the same as PICK1.TP modified above. The difference from PICK1.TP is that, because the robot teaching position differs between infeed conveyors 1 and 2, R[6] is used for indirect Position Register specification, instead of R[3]. In R[6], a value obtained by adding 20 to the value of R[3] is set. This assumes that PR[21] to PR[29] are used as the teaching positions for infeed conveyor 2.

8.9 REPORTING FAILURE TO PICK A PART

This section describes the case in which a check made after a robot attempts to pick a part reveals a failure to pick the part.
In such a case, the position of the part is likely to differ from that detected by the sensor. Therefore, specify “Remove” (4) in PKCSACKQUE to prevent any subsequent robot from picking this part.
The only program that needs to be modified is PICK1.TP shown in Subsection 6.9.2.2, “Pick program”. An example of modifying this program is shown below. Major changes are underlined.

PICK1.TP

```
  1:    UTOOL_NUM =0
  2:    UFRAME_NUM =0
  3:    R[3:ZONE]=1
  4:
  5:    LBL[100]
  6:    STOP_TRACKING
  7:    CALL PKCSGETQUE(CStn ID=R[11:CSTN1],Consec Flag=R[3:ZONE],
        Timeout (ms)=100, Offset VR=1,Stat Reg=2)
  8:    IF R[2:GETQ_STATUS]=0,JMP LBL[200]
  9:    IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 10:    JMP LBL[100]
 11:
 12:    LBL[200]
 13: L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
 14: L PR[R[3:ZONE]] 4000mm/sec CNT3 VOFFSET,VR[1]
 15:    CALL VACUUM_ON
 16: L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
 17:    IF RI[R[3:ZONE]]=ON,JMP LBL[300]
 18:    CALL PKCSACKQUE(CStn ID=R[11:CSTN1],Remove)
```

```
 19:    JMP LBL[100]
 20:
 21:    LBL[300]
 22:    CALL PKCSACKQUE(CStn ID=R[11:CSTN1],Success)
 23:    R[3:ZONE]=R[3:ZONE]+1
 24:    IF R[3:ZONE]<=R[4:NUM_ZONE],JMP LBL[100]
 25:    LBL[900]
[End]
```

On line 17, whether the part has been picked is checked. The RI corresponding to the gripper zone number is checked. If the RI is on, the pickup is successful. If it is off, the pickup is unsuccessful. If the pickup is successful, the program jumps to label [300] and ends normally. If the pickup is unsuccessful, “Remove”(4) (=pickup failure) is reported by PKCSACKQUE on line 18 and the program jumps to label [100] to pick the next part.

8.10 PICKING A TRAY ONLY WHEN ALL CELLS ARE FILLED

This section describes the method to check whether all cells are filled with parts when the most downstream robot on a conveyor picks a tray.

Consider the case in which a tray has four cells and the most downstream robot picks the tray only when all its cells are filled with parts. It is assumed that all the most downstream robot does is to pick a tray. Also, the most downstream robot is assumed to have a different hand from those of other robots.

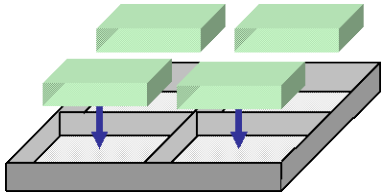


Fig. 8.10(a) Example of picking up a tray only when all the cells are full

In such a case, it is necessary to set the tray and load balance differently.
In this particular case, define one extra cell for the tray. Add the last cell to the layer above that of the other cells so that the last cell is not allocated until all the other cells are filled. Also, assign the last cell a different model ID from that of the other cells so that it is allocated only to the most downstream robot.

Setting the tray

Set the tray as shown below.

Reference cell X 20.0 Y 10.0 Z -5.0 R 0.0
Layer number 1

Cell	X(mm)	Y(mm)	Z(mm)	R(deg)	Model ID
1	20.0	10.0	-5.0	0.0	1
2	20.0	30.0	-5.0	0.0	1
3	50.0	30.0	-5.0	0.0	1
4	50.0	10.0	-5.0	0.0	1

Layer number 2

Cell	X(mm)	Y(mm)	Z(mm)	R(deg)	Model ID
5	20.0	10.0	-5.0	0.0	1

The tray actually has only four cells, but define five cells. Set cell 5 as follows.

- Add it to the second layer.

- Assign a different model ID from that of the other cells.
- Make the position (X,Y,Z,R) the same as reference cell.

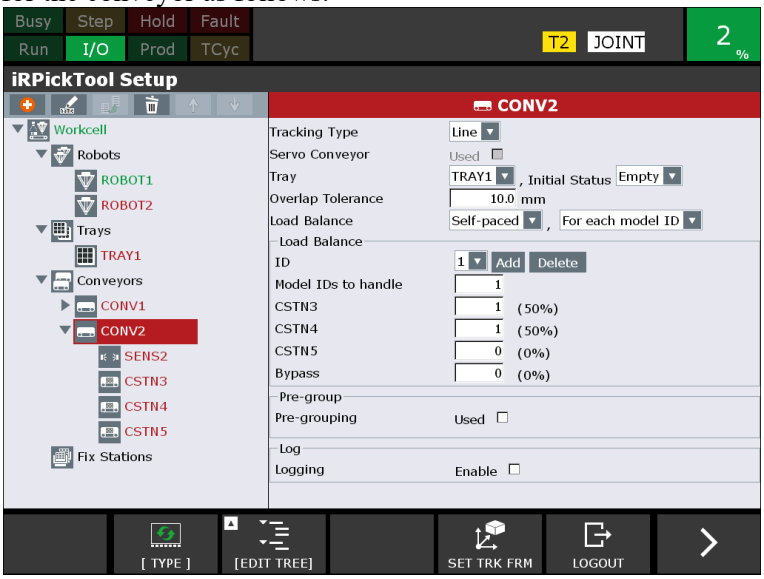
The purpose of making the position (X,Y,Z,R) the same as reference cell is to allow the robot motion for the last cell to be taught independently from the other cells.

NOTE

Normally, teach the robot position only for reference cell. The robot position for each cell is automatically calculated from the tray settings based on reference cell. However, since the operation for cell 5 (picking a tray) is different from the operation for the other cells (placing a part in a cell), the robot position needs to be able to be taught independently. In such a case, make the X, Y, Z, and R values of cell 5 the same as reference cell. This makes the position of cell 5 the same as reference cell, and the offset calculated based on cell 1 becomes 0. As a result, the robot position for cell 5 can be taught freely, as with reference cell.

Setting the conveyor

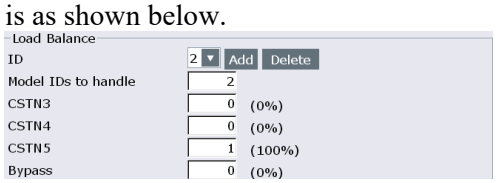
Set the load balance for the conveyor as follows.



This conveyor has three conveyor stations. It is assumed that the first two robots place parts in the cells in a tray, with the third robot assigned to pick the tray.

Since only those cells with model ID 1 need to be allocated to the first two conveyor stations, the load balance for model ID 1 is as shown above (50% set for the first and second conveyor stations, and 0% set for the third conveyor station).

On the other hand, since only those cells with model ID 2 need to be allocated to the last conveyor station, the load balance for model ID 2 is as shown below.



The load balance for model ID 2 in the last conveyor station is 100%. However, since [Empty] is selected for [Initial Status] for the tray, the cell in the second layer (cell 5) is not allocated until any of the cells in

the first layer (cells 1 to 4) is empty. Therefore, the last robot can receive only the information of a tray where cells 1 to 4 are all filled.

Reference position setting and robot position teaching

When performing robot position teaching for the individual robots after setting the reference position in the sensor task screen, teach the tray pickup position to the most downstream robot, instead of the position of cell 1.

> ⚠ **CAUTION**
> Here, it is assumed that all the most downstream robot does is to pick a tray and that the X, Y, Z and R values are the same for both reference cell and cell 5 in the tray settings.

Changing the program

No program needs to be modified.
The most downstream robot gets the information of cell 5 using PKCSGETQUE, as the other robots do, and then picks a tray.

8.11 HANDLING PARTS DETECTED BEFORE STOP WHEN RESTARTING THE SYSTEM

This section describes the method to handle parts on the conveyor that have been detected before the system was stopped, but have not been processed when restarting the main program from the beginning after the system was stopped for an emergency or other reasons.

To avoid deleting information on parts that have been detected, call PKWCCHECK and PKSNSTART instead of calling PKWCSTART. Do not call PKCSCLRQUE. However, if the main program is restarted from the beginning when PKCSGETQUE has been executed before the system was stopped and the execution of PKCSACKQUE has not been completed, PKCSGETQUE is executed before PKCSACKQUE, which has not been executed, resulting in the alarm IRPK-051: "No ACKQUE after the last GETQUE". To avoid this alarm, execute PKCSCLRACK. If PKCSCLRACK is executed, the part information obtained by PKCSGETQUE before the system was stopped will be lost. If part information has been obtained by PKCSGETQUE, the system behavior will be the same as that when PKCSACKQUE is called by specifying "4: Remove".

Example of changing the main program

An example of changing the program MAIN1.TP described in Subsection 6.9.2.1, "Main Program" is shown below. The changed parts are underlined.

MAIN1.TP

```
  1:   R[1:CYCLE_STOP]=0
  2:   CALL PKCSGETID('CONV1', CStn ID Reg=11)
  3:   CALL PKCSGETID('CONV2', CStn ID Reg=12)
  4:
  5:   UTOOL_NUM =1
  6:   UFRAME_NUM =0
  7:   J P[1:Perch Pos] 50% FINE
  8:
  9:   CALL PKWCCHECK
 10:   CALL PKCSCLRACK(CStn ID=R[11:CSTN1])
 11:   CALL PKCSCLRACK(CStn ID=R[12:CSTN2])
 12:   CALL PKSNSTART
 13:
 14:   LBL[100]
 15:   CALL PICK1
 16:   IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 17:   CALL DROP1
 18:   IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 19:   JMP LBL[100]
 20:
 21:   LBL[900]
 22:   CALL PKWCEND
[End]
```

9 OTHER USEFUL FUNCTIONS

This chapter describes the functions that are useful in the *i*RPickTool tracking system. These functions can be used to construct a more stable and flexible system.

9.1 EXCHANGE WORKCELL WITH USING RECIPE

This section describes the setup of the Recipes.

> ⚠ **CAUTION**
> When using a recipe, the software versions of all the controllers connected via the ROS Interface Packets Over Ethernet (RIPE) need to match.

A Recipe is a collection of *i*RPickTool workcell data that is relevant for a unique production run.
For example:
Recipe A is a collection of {Part A or VisionProcess1, 4 x 3 tray matrix, Gripper 1, Ref_Pos1}
Recipe B is a collection of {Part B or VisionProcess2, 8 x 6 tray matrix, Gripper 2, Ref_Pos2}

*i*RPickTool always runs one recipe at a time which is called the Active Recipe.

Users may initiate Recipe changeover procedures only when production is not running. *i*RPickTool will then prepare the workcell to run the new recipe as the active recipe.

1. Recipes may be changed by a PLC.
2. Recipes may be exported to a memory card or a USB device.
3. Recipes may be imported from a memory card or a USB device.

All the recipes are always stored in only one controller in the entire workcell to minimize recipe management and avoid confusion. The controller where all the recipes are stored is called the “Recipe Manager”.

Saving recipes can be done from the teach pendant.
On the [File] screen of the teach pendant, if you select [All] and save, the settings that are common to the recipes will be backed up in the IRPICKTOOL_CFG.ZIP file, and each recipe will be backed up in an IRPK_RECIPEx.ZIP (x is the recipe number) file.

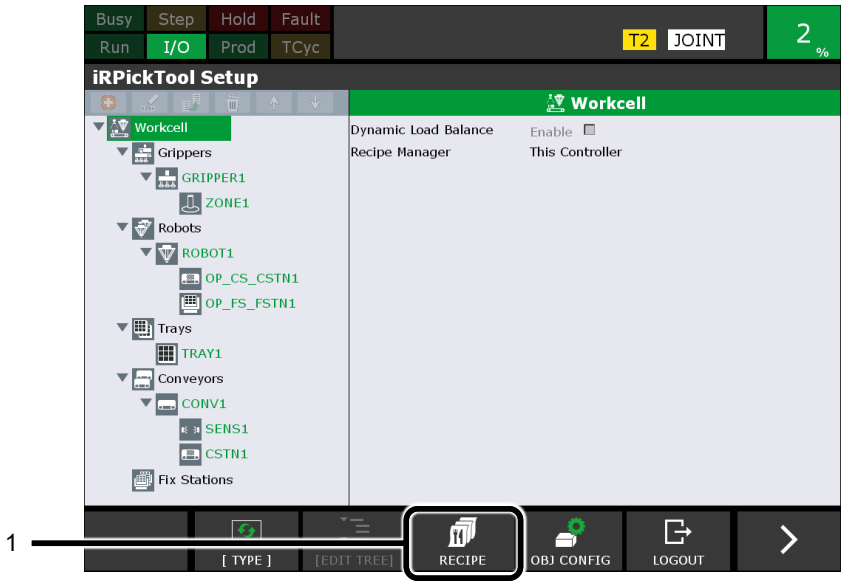
The IRPICKTOOL_CFG.ZIP file and IRPK_RECIPEx.ZIP files can also be loaded to the robot controller. Its extracted contents will go the FR:¥IRPICKTOOL¥ directory.

9.1.1 Setting a Recipe

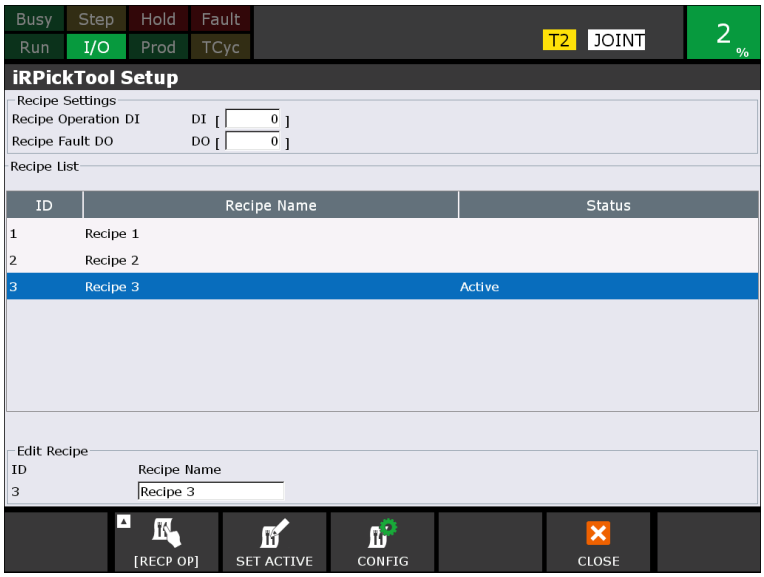
Recipe setup is performed in the Workcell property page.
For details on setup of a workcell, refer to Section 6.1, “SETUP OF A WORKCELL” or Section 7.1, “SETUP OF A WORKCELL”.

The Workcell Page contains a function key called RECIPE. It also contains a property called the “Recipe Manager”. The Recipe Manager refers to the controller in which the Recipes are stored. When you have multiple controllers in the workcell, by default the Recipes are stored in the first controller in the RIPE ring. For details on robot rings, refer to Subsection 5.1.4, “Setting Up the Robot Ring”.

1 On the [Workcell] settings screen, press F3 [RECIPE].



2 The recipe settings screen will appear.
If there is already a recipe, the recipe name will be displayed under [Recipe List].



[Recipe Operation DI]

When this DI is turned ON by external PLC, Recipe will be operated. See Subsection 9.1.3, “Recipe Operation by Means of a PLC” in detail.

[Recipe Fault DO]

If the recipe operate fails due to some reason, this DO turns ON.

[Recipe Name]

You can change the name of recipe which is shown here. See Subsection 9.1.2, “Detail Setting of a Recipe” in detail.

9.1.2 Detail Setting of a Recipe

Set a Recipe.

Creating a new Recipe

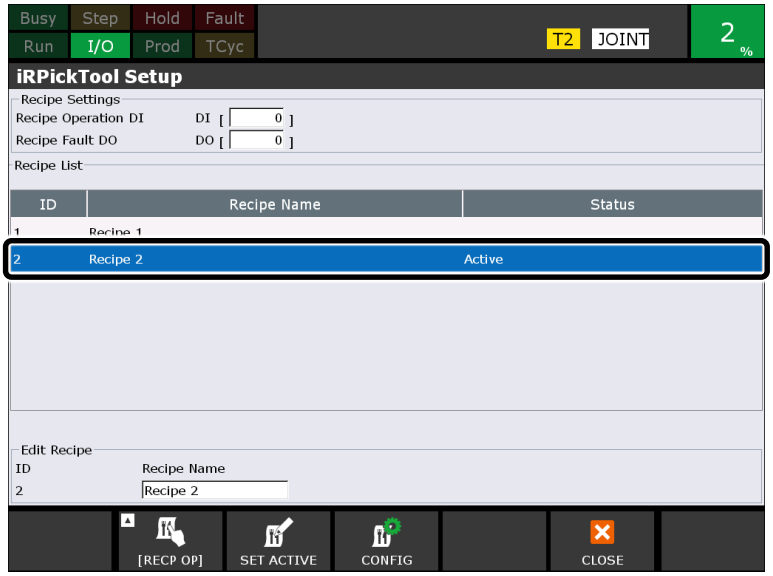
Create a new Recipe.

When you create a Recipe you save the currently setup *i*RPickTool data for all the objects: Workcell, Grippers, Trays, Robots, Conveyors, Sensors, Conveyor Stations and Fixed Stations as XML files into the controller's on-board flash file system in the directory FR:¥IRPICKTOOL¥RECIPEx¥ directory where x is the Recipe ID you see in the Recipe List Window. The XML files also contain the track frame, fixed station frames and the various positions you are required to teach by *i*RPickTool such as the Perch Position, Reference Position, Approach position, etc.

Press F1 [RECP OP] on the recipe settings screen and select [CREATE].

A new recipe with a default name of RecipeX where x is the recipe id is immediately created.
While a recipe is being created, "Now Creating Recipe" will be displayed on the screen. When the recipe is successfully created, you will see a confirmation prompt (not shown). The newly created recipe contains all the current *i*RPickTool data you have setup using the treeview.

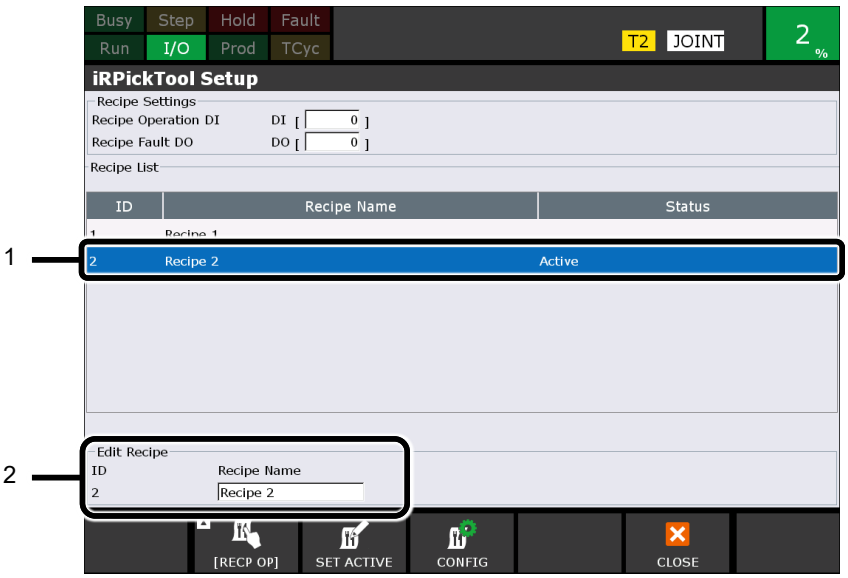
The created recipe will be added to [Recipe List] as shown on the screen below.



Renaming a Recipe

Rename a Recipe.

1 From [Recipe List], select the recipe that you want to rename.
2 The current name of the recipe will be displayed under [Edit Recipe] at the bottom, so edit it directly.



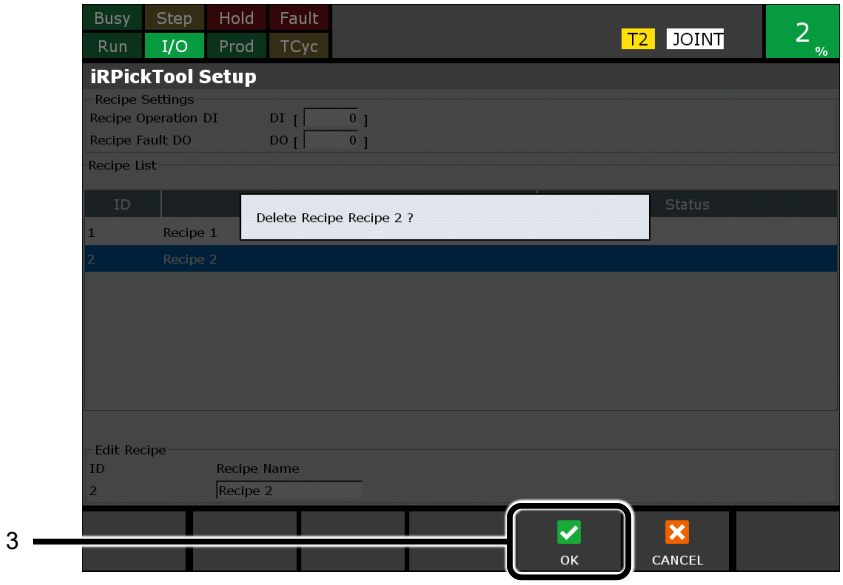
3 Select the text box that contains the current name, and change the name.
4 If you press the 'ENTER' key on the teach pendant, the recipe name will be changed and the new name for the recipe will be displayed in [Recipe List].
Once you change the name, you should save the Recipe. See “Saving a Recipe” section below.
A recipe name can consist of up to 20 one-byte characters. You can include space in the Recipe name. For example, “Panini 2 slice.” The name must not contain a “&”, “ ' ”, “ " ”, “/”, or “¥.”

Deleting a Recipe

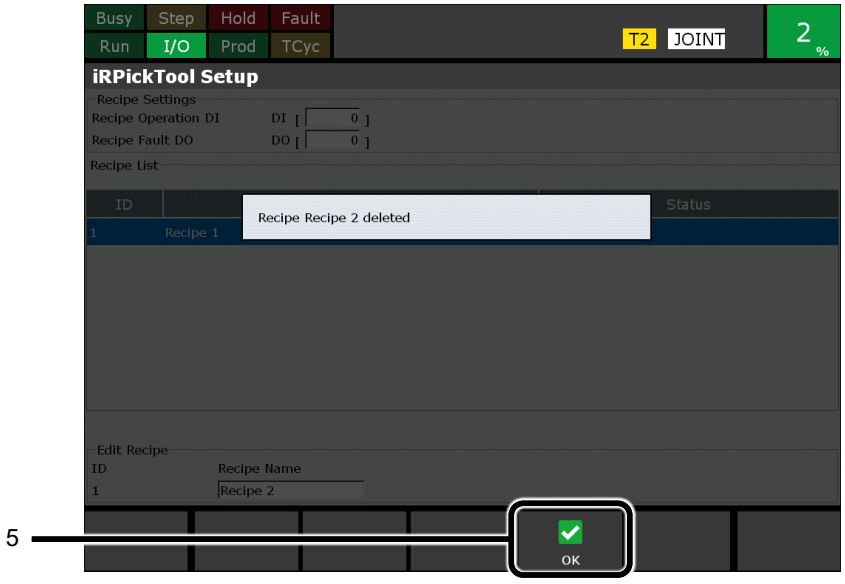
Delete a Recipe.

When you delete the recipe, you actually delete all the data associated with the Recipe in XML files from the FR:¥IRPICKTOOL¥IRPK_RECIPEx directory where x is the Recipe ID that you see in the Recipe List window. You also clear that Recipe name from the Recipe List Window.

1 From [Recipe List], select the recipe that you want to delete.
The selected recipe will be highlighted.
2 Press F1 [RECP OP] and select [DELETE].
A confirmation message like the one shown below will appear.
3 Press F4 [OK] key.



4 You may see the screen display “Now Deleting Recipe” if it takes some time to delete the Recipe. When the recipe has been successfully deleted, you will see the prompt as shown in the figure below.
5 Press F4 [OK] key.



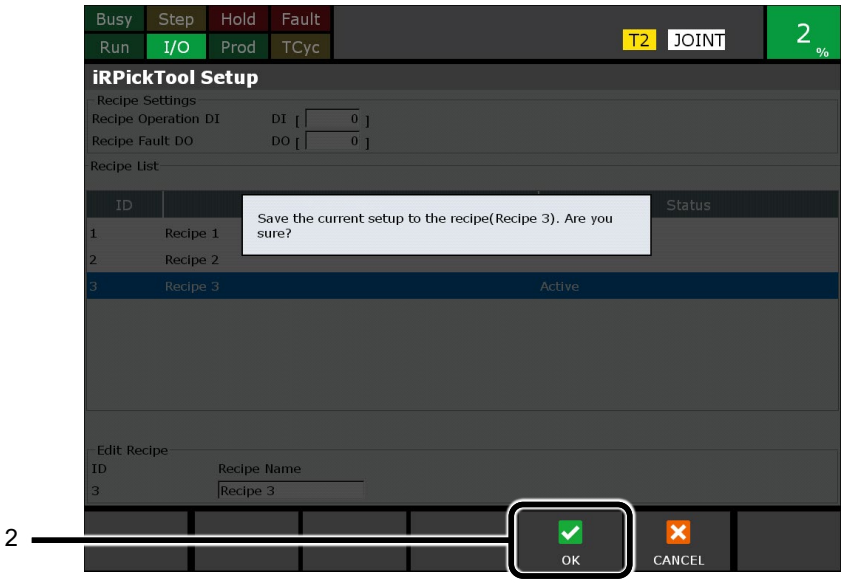
Saving a Recipe

Save a Recipe.

When you save a recipe, you save all the current *i*RPickTool data to the Active Recipe. The Active Recipe is always shown in the Recipe List window. You do the Save operation when you have made some changes to the *i*RPickTool data since you created the recipe or since you last saved the recipe.

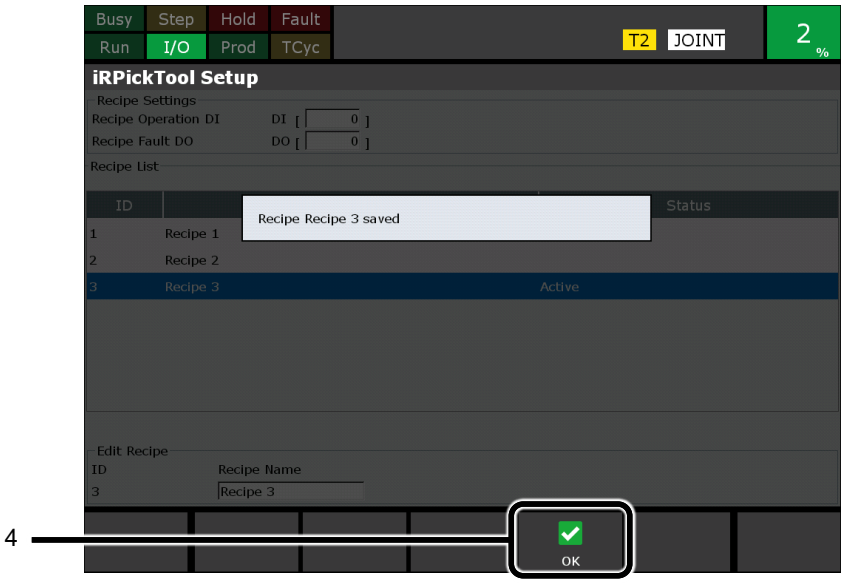
> **⚠ CAUTION**
> A recipe other than the Active Recipe may be highlighted in the Recipe List window. However, the Save operation always saves the data to the Active Recipe. Also, the entire set of *i*RPickTool data is completely saved.

1 Press on F1 [RECP OP] key. You will see a pull-up containing the SAVE choice. Select [SAVE]. You will be asked to confirm as shown in the figure below.
2 Press F4 [OK] key.



3 You may see the screen display “Now Saving Recipe” if it takes some time to save the Recipe. When the recipe has been successfully saved, you will see the prompt as shown in the figure below.

4 Press F4 [OK] key.



Restoring a Recipe

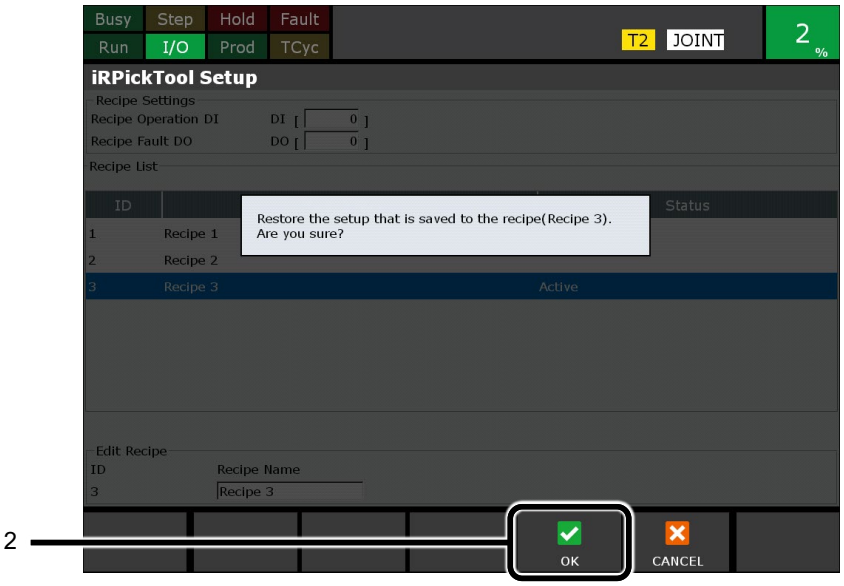
Restore a Recipe.

When you restore a recipe, you load the last saved Active Recipe data to the memory to make it current. You do the Restore operation when you have made some changes to the *i*RPickTool data since you since you last saved the recipe but wish to get rid of those changes and go back to the last saved version. It is a kind of undo operation.

> **⚠ CAUTION**
> A recipe other than the Active Recipe may be highlighted in the Recipe List window. However, the Restore operation always restores the data to the Active Recipe. Also, the entire set of *i*RPickTool data is completely restored.

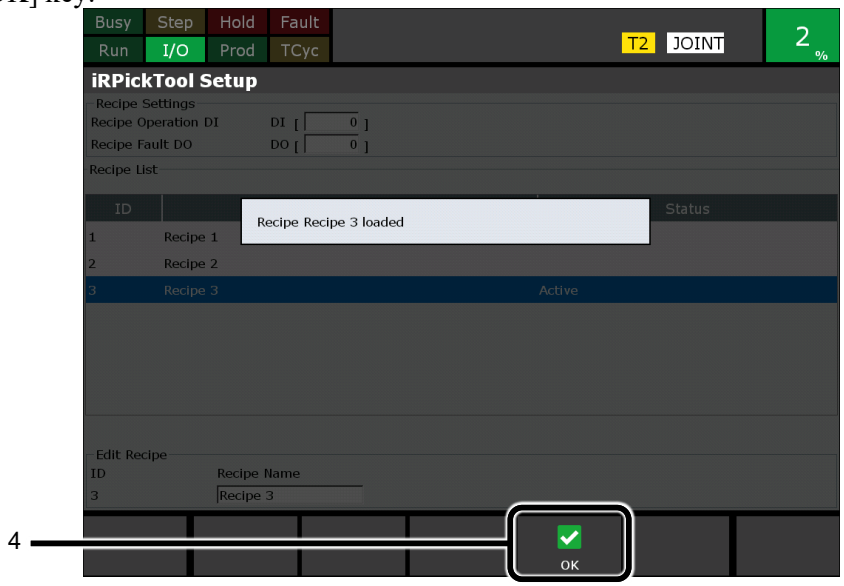
1 Press on F1 [RECP OP] key. You will see a pull-up containing the RESTORE choice.
Select [RESTORE].
You will be asked to confirm as shown in the figure below.

2 Press F4 [OK] key.



3 You may see the screen display “Now Restoring Recipe” if it takes some time to restore the Recipe. When the recipe has been successfully restored, you will see the prompt as shown in the figure below.

4 Press F4 [OK] key.

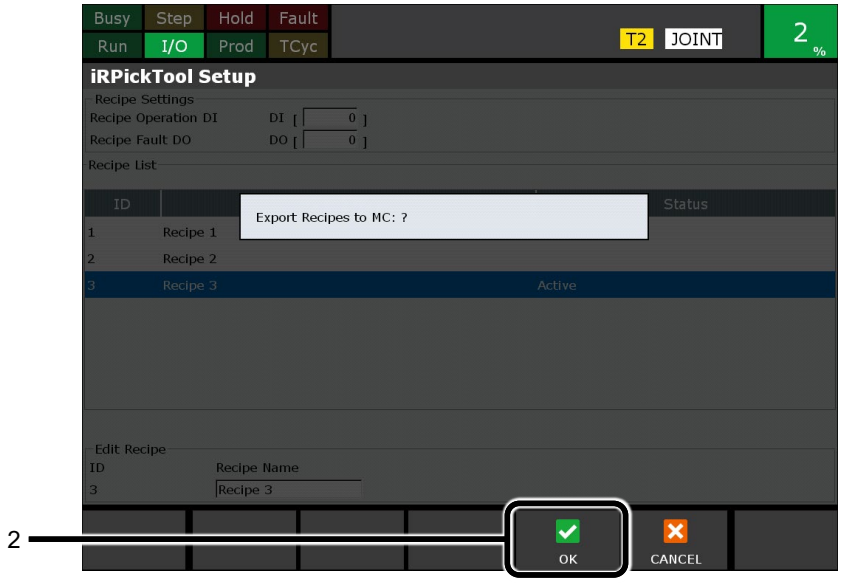


Exporting Recipes

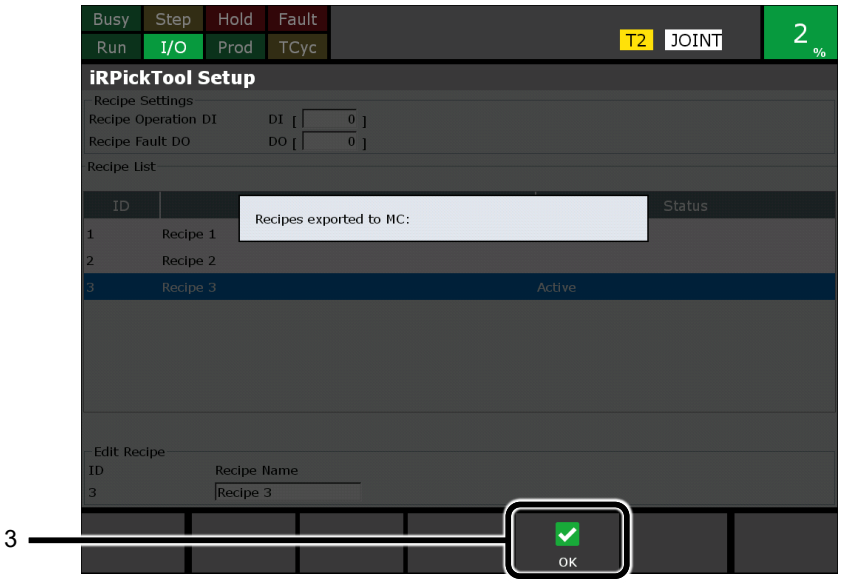
Export Recipes.

When you Export Recipes you copy the recipes stored in FR:¥IRPICKTOOL¥ directory in the controller to an external storage device such as the memory card or the USB stick. The device you are exporting to is the same as $DEVICE. You can change the default device from the File menu – find the [UTIL] function key and press it to display the pull-up. Select “Set Device” and then choose your default device such as “MC” or “UD1” or “UT1:”.

1 Press on F1 [RECP OP] key. in the Recipe page. Select [EXPORT] from the pull-up menu. You will be asked to confirm as shown in the figure below.
2 Press F4 [OK] key.



You will see the screen display “Now Exporting Recipe” if it takes some time to Export the Recipe. When the recipe has been successfully exported, you will see the prompt as shown in the figure below.
3 Press F4 [OK] key.

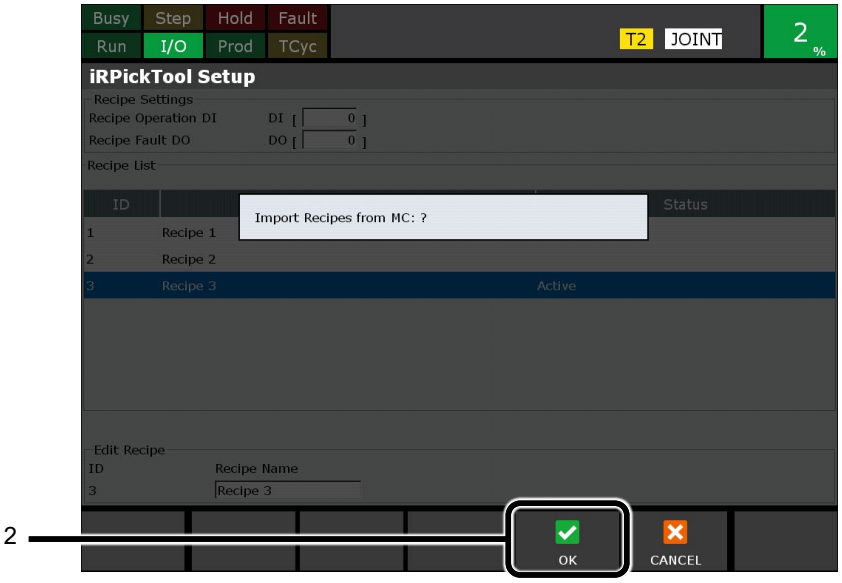


Importing Recipes

Import Recipes.

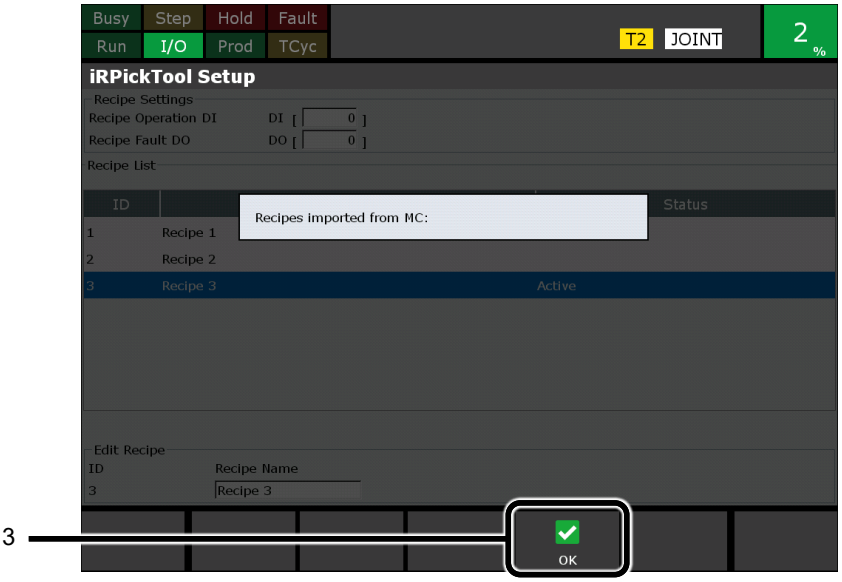
When you Import Recipes you copy the recipes stored in an external storage device such as the memory card or the USB stick into the controller's FR:¥IRPICKTOOL¥ directory. The device you are importing from is the same as $DEVICE. You can change the default device from the File menu – find the [UTIL] function key and press it to display the pull-up. Select "Set Device" and then choose your default device such as "MC" or "UD1" or "UT1:".

1 Press on F1 [RECP OP] key in the Recipe page. Select [5 IMPORT] from the pull-up menu. You will be asked to confirm as shown in the figure below.
2 Press F4 [OK] key.



You will see the screen display "Now Importing Recipe" if it takes some time to Import the Recipe. When the recipe has been successfully imported, you will see the prompt as shown in the figure below.

3 Press F4 [OK] key.

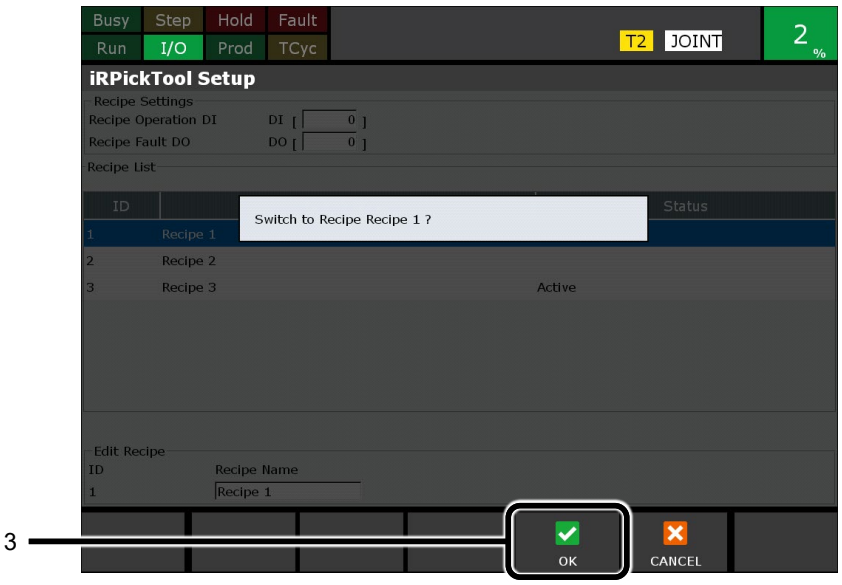


Switching Recipes

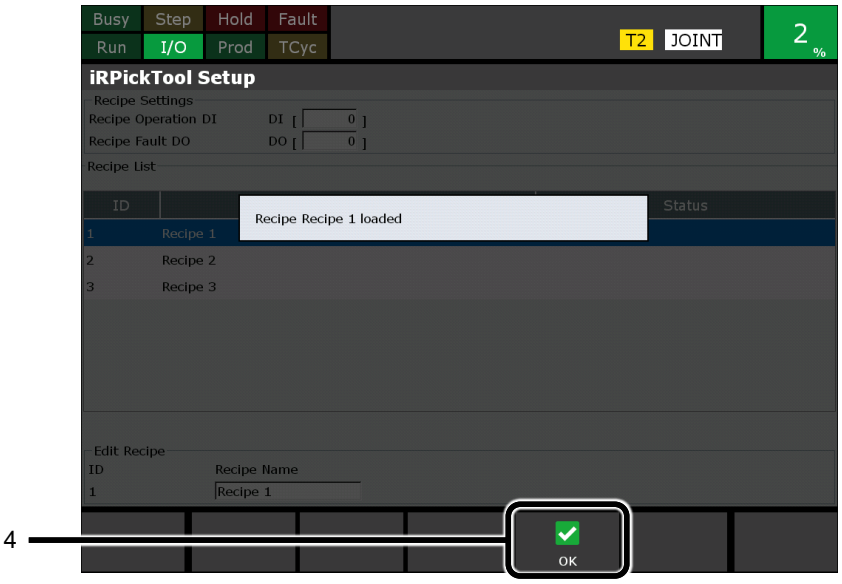
Switch to a different Recipe from the current recipe.

You switch recipes because you wish to run a different product which uses different data that you have saved before. For example, you want to switch from 6 inch pizza to a 18 inch pizza.

1 Press on the Recipe you wish to switch to in the Recipe list window. The Recipe row will be highlighted.
2 Press F2 [SET ACTIVE] key.
The screen will become dark. You will get a confirmation prompt as shown below.
3 Press F4 [OK] key.



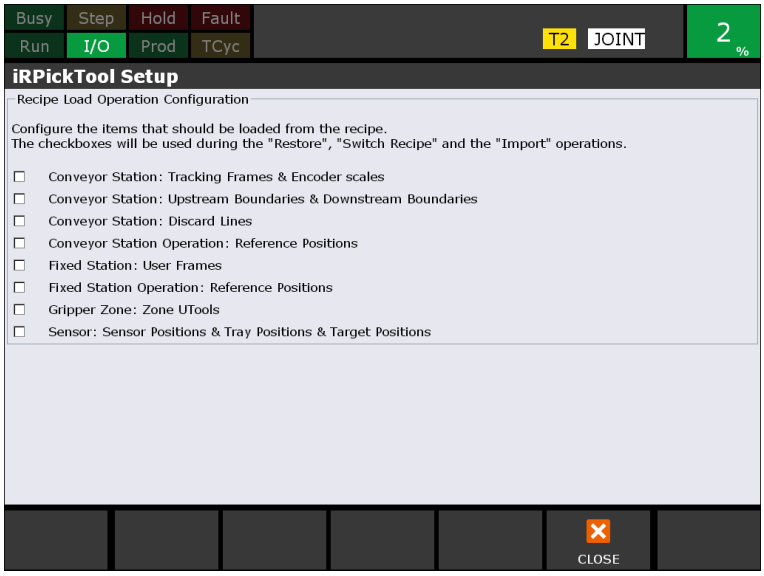
You may see the screen display “Now Switching Recipe” if it takes some time to switch to the new Recipe.
When the recipe has been successfully switched to, you will see the prompt as shown in the figure below.
4 Press F4 [OK] key.



Configuring Recipe Operations

Configure Recipe Operations to follow some rules.

You can configure a recipe operation using this function which is tied to the F3[CONFIG] key on the main Recipe menu. Presently, you can configure only the “Load” function of the recipe which applies to the Restore, Switch Recipe and Import operations.



Typically, when you Switch to a new recipe, or do a Restore of the active recipe, all data saved in the Recipe is loaded. However, there are some cases where you do not want all data to be loaded. For example, you re-taught the track frames for a conveyor or you re-taught the Reference positions to be more accurate in the active recipe. Now when you switch to a different recipe, you do not want the newly taught track frames or the Reference Positions to be overwritten by the old data (in the recipe you are trying to switch to) but you want all other information to be loaded. Then when the new recipe is loaded as you expect, you do a Save Recipe to save the current track frames (new) or Reference positions to the changed recipe. If you have such a need, then you would want to use the Configure Recipe Operation function.
The menu displays the data categories that will be loaded when you Switch Recipe, Restore Recipe or Import Recipe. If you do not want track frames to be loaded, then just uncheck the item “Conveyor Station: Tracking Frames & Encoder scales”. If there are other items in the list that you do not wish to load, then uncheck those items. Press the F5[CLOSE] key to save the current settings and return to the main Recipe page.

In the main Recipe page, you can perform the Switch Recipe operation to switch to a different recipe. Once you switched to the recipe successfully without the data you specified in the “Recipe Operation Load Configuration” menu, do a “Save Recipe” operation. This saves all the latest data to the recipe.
Repeat the steps for other recipes that you wish to configure. When you are done configuring all the recipes, go back to the Configure Recipe menu. Now you can once again “check” all the items if you want all recipes to be loaded completely in the future.

NOTE
1 You can only control the loading of the items in the recipe that are listed in the Recipe Load Operation Configuration menu.
2 The “Save Recipe” operation always saves all the latest data in the controllers of your workcell without any exception.
3 If you always wish to have the listed categories of data to be strictly the same in all the recipes, it may be an advantage to leave all the checkboxes “unchecked”.
4 As default, all the checkboxes are “unchecked” and the listed categories of the recipe data are not loaded. In the case of North America Setting, the checkboxes for all the items displayed on this screen will have checks as default.

9.1.3 Recipe Operation by Means of a PLC

You can operate recipes from a PLC. This operation is performed using the following procedure. Recipe operation by means of a PLC can be performed only with a recipe manager controller.

1 On the [Recipe] settings screen for a robot controller that is a recipe manager, set the DI number for [Recipe Operation DI].
2 From the PLC, turn ON the DI that you have set in step 1.
3 When the DI is turned ON, it will be detected by the robot controllers and the program to perform a handshake with the PLC will be executed. As a standard program for this purpose, PK_PLC_RECIPE_OPERATION.TP has been prepared.
4 If you use a KAREL program, you can operate recipes. The arguments for KAREL programs are provided by group input signals in standard programs.
5 In the event of an error during recipe operation, the DO that has been set for [Recipe Fault] will turn ON.

If recipe operation is already being executed, recipe operation by means of a PLC is not allowed. For example, if [Recipe Operation DI] is turned ON while executing a recipe operation on the [Recipe] settings screen, recipe operation will not be performed, so [Operation Fault DO] will turn ON.

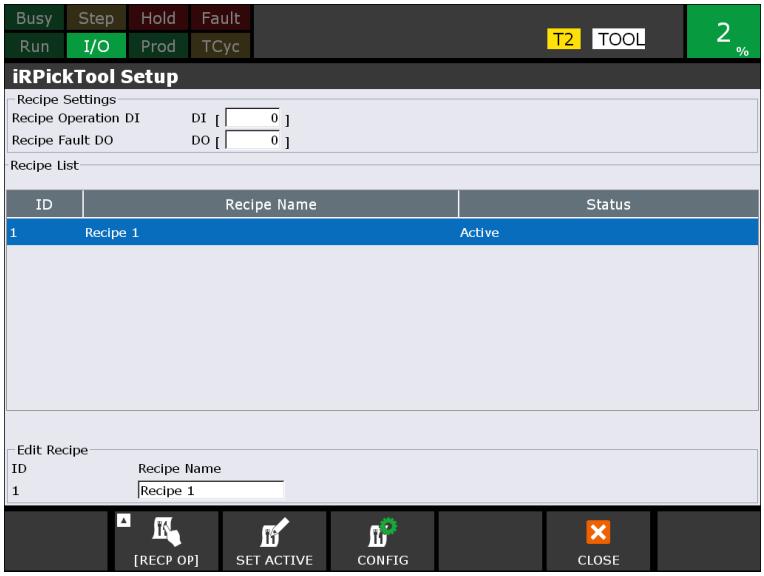
NOTE
It sometime takes 5 and more seconds to complete this recipe operation by means of a PLC.

9.1.3.1 Settings required for recipe operation by means of a PLC

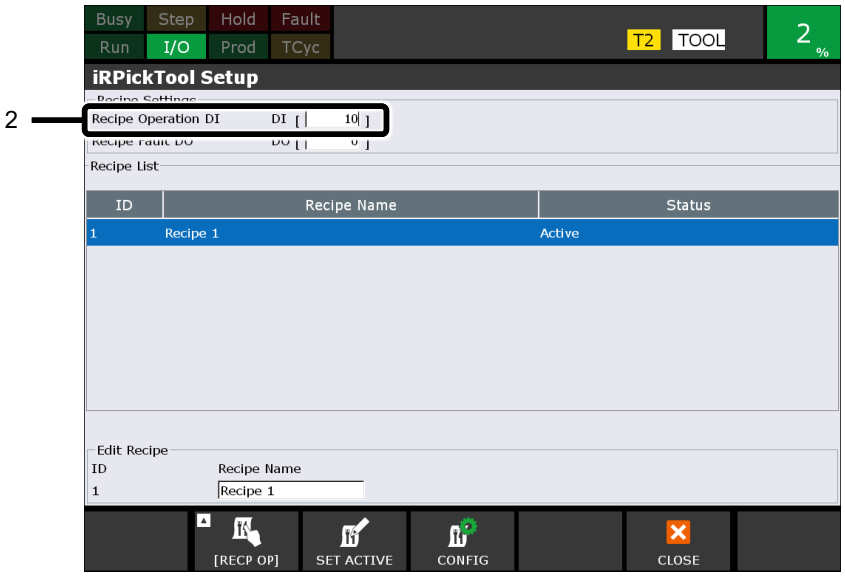
Set up settings that are required for recipe operation by PLC.

Settings required for recipe operation by means of a PLC

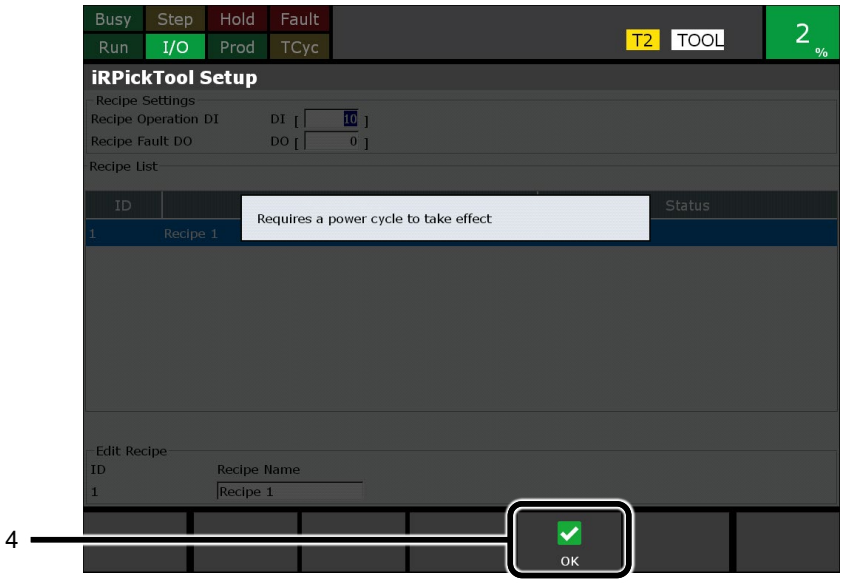
1 Display the Recipe menu by clicking on F3 [RECIPE] key from the Workcell menu. The Recipe menu will be displayed as shown below. Notice the Recipe Settings Window at the top.



2 In the Recipe Settings window, enter a valid Digital Input for the property “Recipe Change DI” as shown below:



3 When you press Enter you will get the message below.
4 Press F4 [OK] key.



5 If it is necessary to check whether an error has occurred in the PLC, set the DO for [Recipe Fault DO].
6 Reboot the robot controller.

Modify the PLC handshake program PK_PLC_RECIPE_OPERATION.TP

The standard PK_PLC_RECIPE_OPERATION.TP program is shown below. You can modify the program to suit your need. The basic steps are:

1 For the PLC, specify operations and parameters, depending on group input (GI), etc.
2 Call a KAREL program, depending on the specified operation.
3 Check the KAREL program status.

This program obtains operations and parameters by communicating with a PLC, and then performs recipe operation. It is executed automatically when the [Recipe Operation DI] that was set on the [Recipe] settings screen under the tree view is turned ON.

This program can only be run when you are not running production – i.e., all the robots are aborted.

This program obtains operations and parameters by means of group I/O after performing a handshake with the PLC, but the means of communication can be changed to suit the application.

In this program:
1 DO[32] is used by the robot to ack the PLC that is ready to receive data.
2 DI[32] is used to receive the ACK from the PLC that is ready to send the data.
3 GI[1] is used in order for the PLC to send operations and parameters to the robot.
4 GO[1] is used in order for the robot to re-send the operations and parameters that have been received to the PLC to confirm receipt.
5 A KAREL program will be selected in accordance with the operation sent from the PLC. Depending on the operation, the parameters sent from PLC are specified as the arguments for a KAREL program, and the KAREL program is called.
6 The KAREL program status indicates whether the recipe operation has succeeded or not. On the [Recipe] settings screen, the DO number will be set to 'Recipe Fault DO', and if the state is anything other than 0, the 'Recipe Fault' DO will be turned ON.

PK_PLC_RECIPE_OPERATION.TP

```
 1:  !-------------------------------
 2:  ! Communication program between a
 3:  ! PLC and iRPickTool to perform
 4:  ! Recipe operations.
 5:  ! This program runs if the Recipe
 6:  ! Change DI setup in the Recipe
 7:  ! menu is turned on.
 8:  ! Users can modify the PLC
 9:  ! handshake - you can change
10:  ! the DI/DO, GI/GO and registers
11:
12:  !-------------------------------
13:
14:  !Recipe Operation Function Codes
15:  ! CREATE       = 1
16:  ! DELETE       = 3
17:  ! SAVE_ACTIV_RECIPE = 4
18:  ! RESTORE_ACTIVE_RECIPE = 5
19:  ! EXPORT       = 6
20:  ! IMPORT       = 7
21:  ! SWITCH       = 8
22:  !
23:  ! Numeric registers R[95]-R[98]
24:  ! are used in this program.
25:  ! Users can re-assign the
26:  ! registers to other numbers
27:  ! to suit their conveninence.
28:  !
29:  !R[95] is used for storing status
30:
31:
32:  ! Turn off DO[33] (Success)
33:  DO[33:RecipeSuccess]=OFF
34:  ! Turn off DO[34] (Error)
35:  DO[34:RecipeFault]=OFF
36:
37:  !Tell PLC robot ready to receive
38:  DO[32:RecipeRobotReady]=ON
39:  ! Wait for PLC ready to send
40:  WAIT DI[32:RecipePLCReady]=ON TIMEOUT,LBL[4000]
41:  !Read function code
42:  R[97:TemporaryUse3]=GI[1:RecipeReadData]
43:  ! Echo Recipe Operation to PLC
44:  GO[1:RecipeEchoData]=R[97:TemporaryUse3]
45:  ! Signal PLC that data received.
46:  DO[32:RecipeRobotReady]=OFF
47:  ! Wait for PLC acknowledgement.
48:  WAIT DI[32:RecipePLCReady]=OFF TIMEOUT,LBL[4000]
49:
50:  SELECT R[97:TemporaryUse3]=1,JMP LBL[100]
51:              =3,JMP LBL[300]
52:              =4,JMP LBL[400]
53:              =5,JMP LBL[500]
54:              =6,JMP LBL[600]
```

```
   55:            =7,JMP LBL[700]
   56:            =8,JMP LBL[800]
   57:            ELSE,JMP LBL[4000]
   58:
   59:  LBL[100:Create]
   60:  !Tell PLC robot ready to receive
   61:  DO[32:RecipeRobotReady]=ON
   62:  ! Wait for PLC ready to send
   63:  WAIT DI[32:RecipePLCReady]=ON TIMEOUT,LBL[4000]
   64:  ! Read recipe_id register
   65:  R[98:TemporaryUse4]=GI[1:RecipeReadData]
   66:  ! Echo Recipe Id reg to PLC
   67:  GO[1:RecipeEchoData]=R[98:TemporaryUse4]
   68:  ! Signal PLC that data received.
   69:  DO[32:RecipeRobotReady]=OFF
   70:  ! Wait for PLC acknowledgement.
   71:  WAIT DI[32:RecipePLCReady]=OFF TIMEOUT,LBL[4000]
   72:  !Tell PLC robot ready to receive
   73:  DO[32:RecipeRobotReady]=ON
   74:  ! Wait for PLC ready to send
   75:  WAIT DI[32:RecipePLCReady]=ON TIMEOUT,LBL[4000]
   76:  ! Read string reg id for storing
   77:  ! recipe name
   78:  R[96:TemporaryUse2]=GI[1:RecipeReadData]
   79:  ! Echo string reg Id to PLC
   80:  GO[1:RecipeEchoData]=R[96:TemporaryUse2]
   81:  ! Signal PLC that data received.
   82:  DO[32:RecipeRobotReady]=OFF
   83:  ! Wait for PLC acknowledgement.
   84:  WAIT DI[32:RecipePLCReady]=OFF TIMEOUT,LBL[4000]
   85:  CALL PKRCCREATE(“RecipeID Reg”=R[98:TemporaryUse],“RecipeNm
StrReg”=R[96:Temporary],“Stat Reg”=95)
   86:  JMP LBL[5000]
   87:
   88:  LBL[300:Delete]
   89:  !Tell PLC robot ready to receive
   90:  DO[32:RecipeRobotReady]=ON
   91:  ! Wait for PLC ready to send
   92:  WAIT DI[32:RecipePLCReady]=ON TIMEOUT,LBL[4000]
   93:  ! Read recipe id to delete
   94:  R[98:TemporaryUse4]=GI[1:RecipeReadData]
   95:  ! Echo Recipe Id to PLC
   96:  GO[1:RecipeEchoData]=R[98:TemporaryUse4]
   97:  ! Signal PLC that data received.
   98:  DO[32:RecipeRobotReady]=OFF
   99:  ! Wait for PLC acknowledgement.
  100:  WAIT DI[32:RecipePLCReady]=OFF TIMEOUT,LBL[4000]
  101:  CALL PKRCDELETE(“Recipe ID”=R[98:TemporaryUse4],“Stat Reg”=95)
  102:  JMP LBL[5000]
  103:
  104:  LBL[400:Save]
  105:  CALL PKRCSAVE(“Stat Reg”=95)
  106:  JMP LBL[5000]
  107:
  108:  LBL[500:Restore]
```

```
  109:   CALL PKRCRESTORE(“Stat Reg”=95)
  110:   JMP LBL[5000]
  111:
  112:   LBL[600:Export]
  113:   CALL PKRCEXPORT(“Stat Reg”=95)
  114:   JMP LBL[5000]
  115:
  116:   LBL[700:Import]
  117:   CALL PKRCIMPORT(“Stat Reg”=95)
  118:   JMP LBL[5000]
  119:
  120:   LBL[800:Switch]
  121:   !Tell PLC robot ready to receive
  122:   DO[32:RecipeRobotReady]=ON
  123:   ! Wait for PLC ready to send
  124:   WAIT DI[32:RecipePLCReady]=ON TIMEOUT,LBL[4000]
  125:   ! Read recipe id to switch to
  126:   R[98:TemporaryUse4]=GI[1:RecipeReadData]
  127:   ! Echo Recipe Id to PLC
  128:   GO[1:RecipeEchoData]=R[98:TemporaryUse4]
  129:   ! Signal PLC that data received.
  130:   DO[32:RecipeRobotReady]=OFF
  131:   ! Wait for PLC acknowledgement.
  132:   WAIT DI[32:RecipePLCReady]=OFF TIMEOUT,LBL[4000]
  133:   CALL PKRCACTIVE(“Recipe ID”=R[98:TemporaryUse4],“Stat Reg”=95)
  134:   JMP LBL[5000]
  135:
  136:   LBL[4000:PLC Recipe Error]
  137:   ! 127294 = PLC Rec Op failed
  138:   R[95:TemporaryUse1]=127294
  139:   JMP LBL[5000]
  140:
  141:   LBL[5000:Check status]
  142:   IF (R[95:TemporaryUse1]<>0) THEN
  143:   ! Turn Error DO on
  144:   DO[34:RecipeFault]=ON
  145:   ! Post error and abort
  146:   CALL PKPOSTERR(“Error Code”=R[95:TemporaryUse1],“ABORT”=2)
  147:   ELSE
  148:   ! Turn success DO on
  149:   DO[33:RecipeSuccess]=ON
  150:   ENDIF
  151:   JMP LBL[10000]
  152:
  153:   LBL[10000:End]
/END
```

9.1.4 KAREL Program

> ⚠ **CAUTION**
> When performing a recipe operation using the KAREL programs introduced here, call PKWCEND first to stop the sensors. In addition, if the recipe operation is forced to stop in the middle, the error IRPK-083 “Another recipe process is running.” may occur when a recipe operation is performed next time. To prevent the recipe from being forced to stop during operation, we recommend setting the TP programs that use the KAREL programs introduced here to ignore pause in the programs' detail settings.

9.1.4.1 PKRCCREATE.PC

This KAREL program is used only when creating a recipe. You can run the program only when all the robot controller programs that have been set in the tree view have finished.

When you run this program, it does the following:
- Turns [Recipe Fault DO] OFF if it is ON.
- Creates a recipe.
- Sets the ID of the created recipe.
- Sets the name of the created recipe.
- Sets the status.
- Turns [Recipe Fault DO] ON if the status is not 0.

Argument 1:
Specify the register where the recipe ID is stored.

Argument 2:
Specify the string register where the recipe name is stored.

Argument 3:
Specify the register to return the status to.

Usage example:
Create a recipe. Return the recipe ID to R[2], the recipe name to SR[3], and the status to R[1].

```
CALL PKRCCREATE （“RecipeID Reg”=2,“RecipeNm StrReg”=3,“Stat Reg”=1)
```

9.1.4.2 PKRCDELETE.PC

This KAREL program is used only when deleting a recipe. You can run the program only when all the robot controller programs that have been set in the tree view have finished.

When you run this program, it does the following:
- Turns [Recipe Fault DO] OFF if it is ON.
- Checks whether the specified recipe is valid.
- If it is valid, it deletes the specified recipe.
- Sets the status.
- Turns [Recipe Fault DO] ON if the status is not 0.

Argument 1:
Specify the ID of the recipe that you want to delete.

Argument 2:
Specify the register to return the status to.

Usage example:
Delete the recipe for which the ID is 3. Return the status to R[1].

```
PKRCDELETE("Recipe ID" =3, "Stat Reg"=1)
```

9.1.4.3 PKRCSAVE.PC

This KAREL program is used only when saving a recipe. You can run the program only when all the robot controller programs that have been set in the tree view have finished.

When you run this program, it does the following:
- Turns [Recipe Fault DO] OFF if it is ON.
- Saves a recipe
- Sets the status.
- Turns [Recipe Fault DO] ON if the status is not 0.

Argument 1:
Specify the register to return the status to.

Usage example:
Save a recipe. Return the status to R[1].

```
PKRCSAVE("Stat Reg"=1)
```

9.1.4.4 PKRCRESTORE.PC

This KAREL program is used only when loading a recipe. You can run the program only when all the robot controller programs that have been set in the tree view have finished.

When you run this program, it does the following:
- Turns [Recipe Fault DO] OFF if it is ON.
- Loads a recipe.
- Sets the status.
- Turns [Recipe Fault DO] ON if the status is not 0.

Argument 1:
Specify the register to return the status to.

Usage example:
Load a recipe. Return the status to R[1].

```
PKRCRESTORE("Stat Reg"=1)
```

9.1.4.5 PKRCEXPORT.PC

This KAREL program is used only when exporting a recipe. You can run the program only when all the robot controller programs that have been set in the tree view have finished.

When you run this program, it does the following:
- Turns [Recipe Fault DO] OFF if it is ON.
- Exports a recipe.
- Sets the status.
- Turns [Recipe Fault DO] ON if the status is not 0.

Argument 1:
Specify the register to return the status to.

Usage example:
Export recipe. Return the status to R[1].

```
PKRCEXPORT("Stat Reg"=1)
```

9.1.4.6 PKRCIMPORT.PC

This KAREL program is used only when importing a recipe. You can run the program only when all the robot controller programs that have been set in the tree view have finished.

When you run this program, it does the following:
- Turns [Recipe Fault DO] OFF if it is ON.
- Imports a recipe.
- Sets the status.
- Turns [Recipe Fault DO] ON if the status is not 0.

Argument 1:
Specify the register to return the status to.

Usage example:
Import recipe. Return the status to R[1].

```
PKRCIMPORT("Stat Reg"=1)
```

9.1.4.7 PKRCACTIVE.PC

This KAREL program is used only when switching recipes. You can run the program only when all the robot controller programs that have been set in the tree view have finished.

When you run this program, it does the following:
- Turns [Recipe Fault DO] OFF if it is ON.
- It then validates if the recipe is a valid recipe.
- Loads the new recipe.
- Sets the status.
- Turns [Recipe Fault DO] ON if the status is not 0.

Argument 1:
Specify an ID of the new recipe you wish to switch to.

Argument 2:
Specify the Numeric Register in which to return the status.

Usage:
CALL PKRCSWITCH(recipe_id, status)

```
CALL PKRCACTIVE("Recipe ID"=3,"Stat Reg"=1)
```

9.2 CHECKING THE PRODUCTION STATUS

The production status of each *i*RPickTool object can be checked.

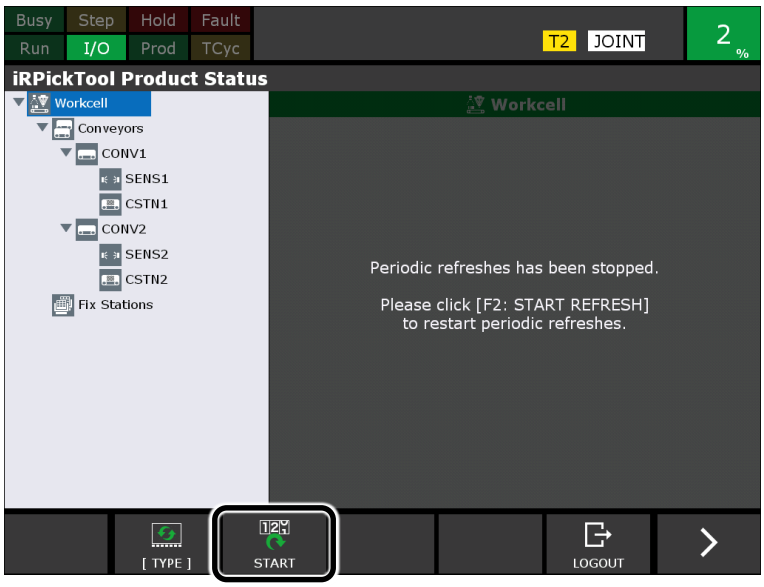
1 Press the [MENU] key on the teach pendant.
2 Select [NEXT].
3 Select [STATUS].
4 Press the F1 [TYPE] key, and select [*i*RPickTool] from the options.

Basic operations

The operations common to all objects are described below.

Stopping and resuming start refresh

If you want to stop periodic refresh intentionally, press F2 [STOP REFRESH] during periodic refresh.



Press F2 [START] to resume start refresh.

Resetting the production status

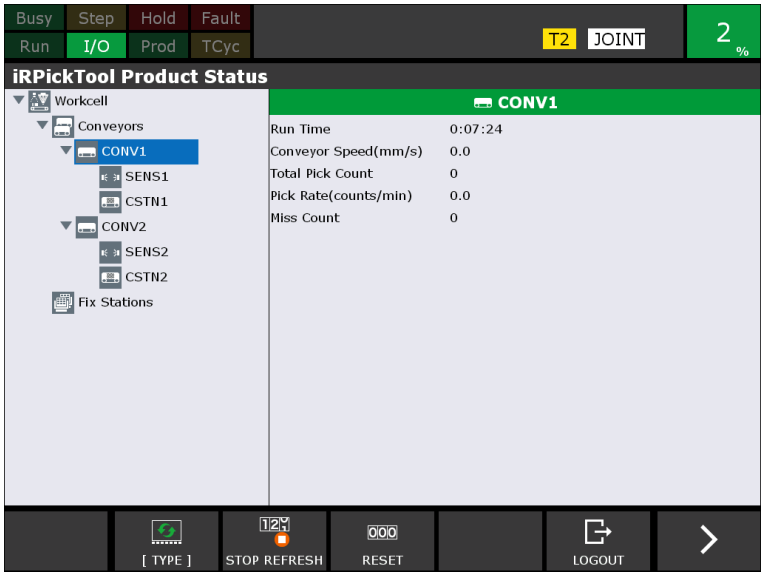
To reset the production time and statistic values and resume the collection of statistics from that point of time, press F3 [RESET] key.

Refreshing the tree

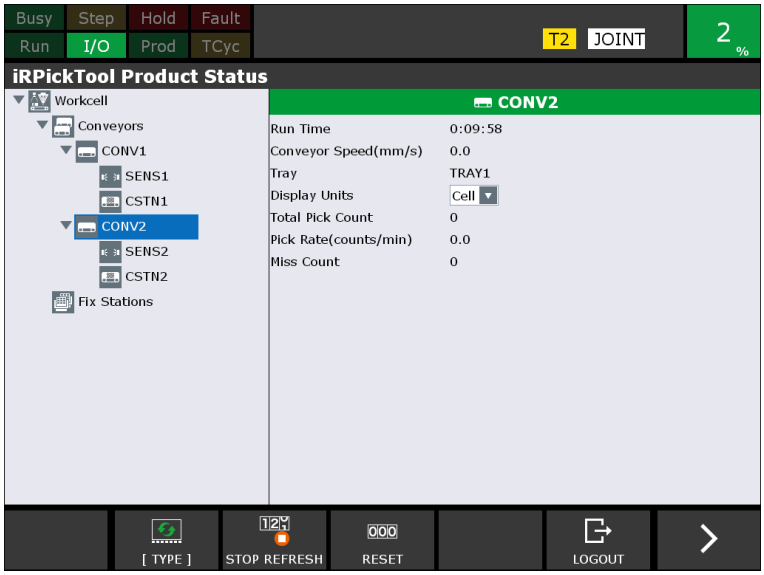
If you perform an operation related to the tree, such as adding a conveyor station in the *i*RPickTool setup screen, while logged in to the production status screen, you cannot have the change reflected in the production status screen without doing anything. To have the change reflected, press the [> (NEXT)] key and then F1 [UPDATE TREE] key. Doing so displays the change (the newly added conveyor station in this case) on the tree.
You can also refresh the tree by logging out from the production status screen and then logging in again.

9.2.1 Checking the Status of a Conveyor

The part processing status of an entire conveyor can be checked.
From the tree, select a desired conveyor. When trays are not used, a screen as shown below is displayed.



When trays are used, a screen as shown below is displayed.



[Run Time]
Time elapsed since production started

[Conveyor Speed(mm/s)]
Conveyor speed

[Tray]
Name of the tray used on this conveyor. This is displayed only when trays are used.

[Display Units]
Select whether to display the number of conveyed parts on a per-cell or per-tray basis. This item is displayed only when trays are used.

[Total Pick Count]

Total number of parts processed on the selected conveyor. When trays are used, the total number of cells or trays processed on the selected conveyor is displayed, depending on the selected display unit.

[Pick Rate(counts/min)]

Average number of conveyed parts obtained by dividing the total number of parts processed on the selected conveyor by the production time. When trays are used, the average number of conveyed cells or trays obtained by dividing the total number of cells or trays processed on the selected conveyor by the production time, depending on the selected display unit.

[Miss Count]

Total number of parts not processed on any conveyor station. It does not include parts which has been failed to be detected by vision.

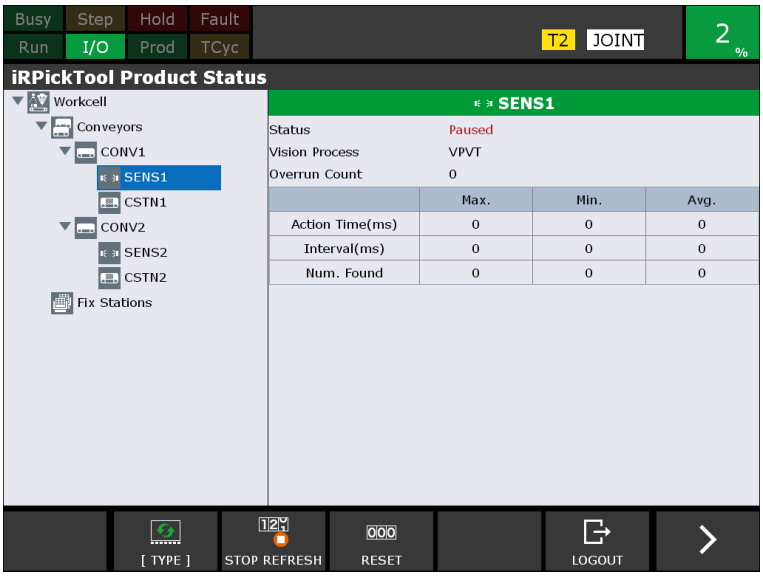
[RESET]

Pressing F3 [RESET] key resets all the data. After the reset, recording starts again.

9.2.2 Checking the Status of a Sensor

The status of a sensor can be checked.

From the tree, select a desired sensor.



[Status]

Status of the selected sensor. When the sensor is active, [Running] is displayed in green. When the sensor is inactive, [Paused] is displayed in red.

[Vision Process]

Name of the vision process used by the selected sensor. This item is displayed only when the vision system is used.

[Overrun Count]

Number of times the next trigger has occurred before the end of the preceding image processing when the vision system is used. A value other than 0 indicates that the system is not capable of completing the image processing fast enough, in which case you need to reduce the frequency with which a trigger occurs. To be more concrete, set the value longer than the following [Action Time] in [Trigger Distance], which is set in the Subsection 6.5.1.2, "When [Distance] and [Find part by vision] are selected" of Subsection 6.5.1, "Setting a Sensor in Detail" , Subsection, 3.2.5 "Setting Up a Sensor" or Subsection 4.2.6, "Setting Up a Sensor".

[Action Time(ms)]

Time it takes before the selected sensor detects a part and that detected part is put to the most upstream conveyor station. If this value is larger than the trigger interval, the value in [Overrun Count] becomes larger.

[Interval(ms)]

Interval at which a trigger occurs

[Num. Found]

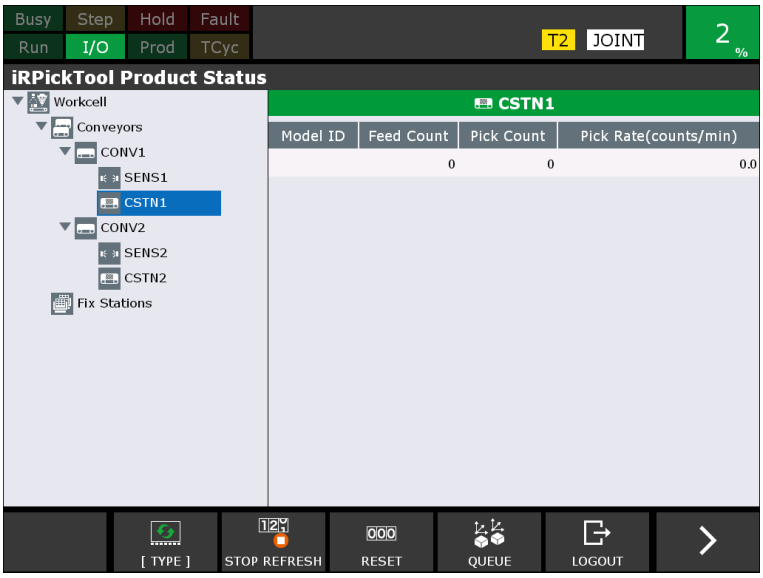
Number of parts detected each time the image processing is performed when the vision system is used (number of parts in the camera's field of view)

[RESET]

Pressing F3 [RESET] key resets all the data. After the reset, recording starts again.

9.2.3 Checking the Status of a Conveyor Station

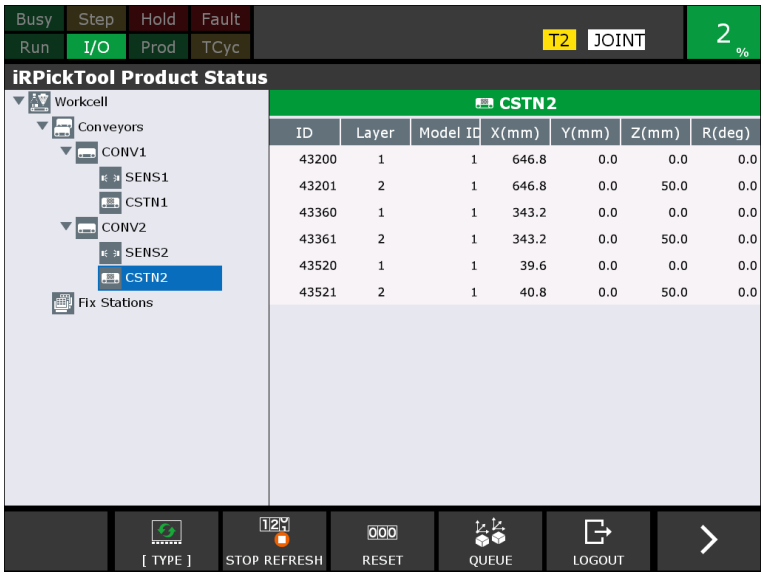
The part processing status of a conveyor station can be checked.
From the tree, select a desired conveyor station. When trays are not used, the production status screen is displayed as shown below.



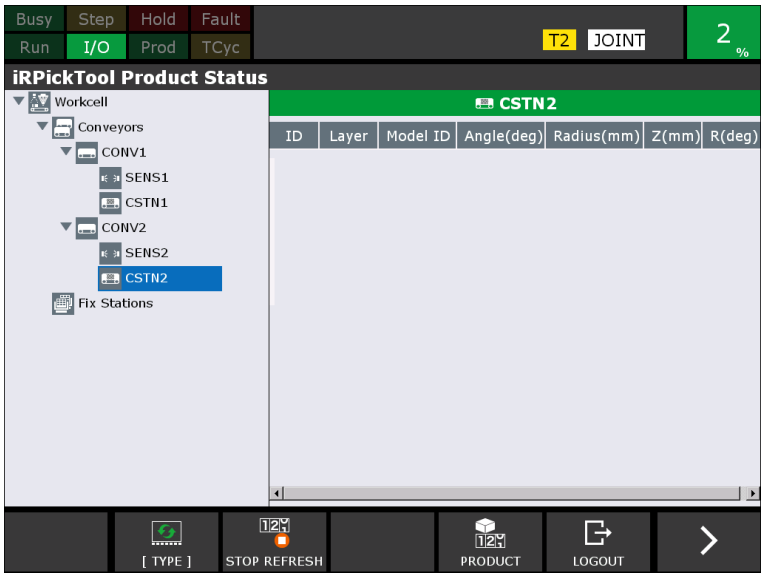
When trays are used, the production status screen is displayed as shown below.

If you press F4 [QUEUE] key, the status, you can view the queue of parts or parts that is yet to be processed by the conveyor station. You can view the leading 10 entries in the queue. Each entry shows the distance of the part on the conveyor along the flow direction (x), and along y and r directions from the nominal tracking frame. This screen is the one displayed when trays are not used.

When trays are used, the status of each cell in the conveyor station's queue is displayed as shown below.



When circular conveyor are used, current position of the parts is displayed at an angle and radius instead of X and Y as shown below.



If you press F4 [PRODUCT] key, the processing status screen is displayed again.

Product

The information about the part processing status for each individual model ID is listed.

[Tray]

Name of the tray used on the conveyor. This item is displayed only when trays are used.

[Display Units]

Select whether to display the number of conveyed parts on a per-cell or per-tray basis. This item is displayed only when trays are used.

[Model ID]

Model ID of the parts handled by the selected conveyor station

[Feed Count]
Total number of parts conveyed on the selected conveyor station

[Pick Count]
Total number of parts processed on the selected conveyor station. When trays are used, the total number of cells or trays processed on the selected conveyor station is displayed, depending on the selected display unit.

[Pick Rate(counts/min)]
Average number of conveyed parts obtained by dividing the total number of parts processed on the selected conveyor station by the production time. When trays are used, the average number of conveyed cells or trays obtained by dividing the total number of cells or trays processed on the selected conveyor station by the production time, depending on the selected display unit.

Queue

The Queue displays the leading 10 entries sequentially in the queue of the conveyor station.

[ID]
Number to identify the part or cell

[Layer]
Layer number of the cell; this item is displayed only when trays are used.

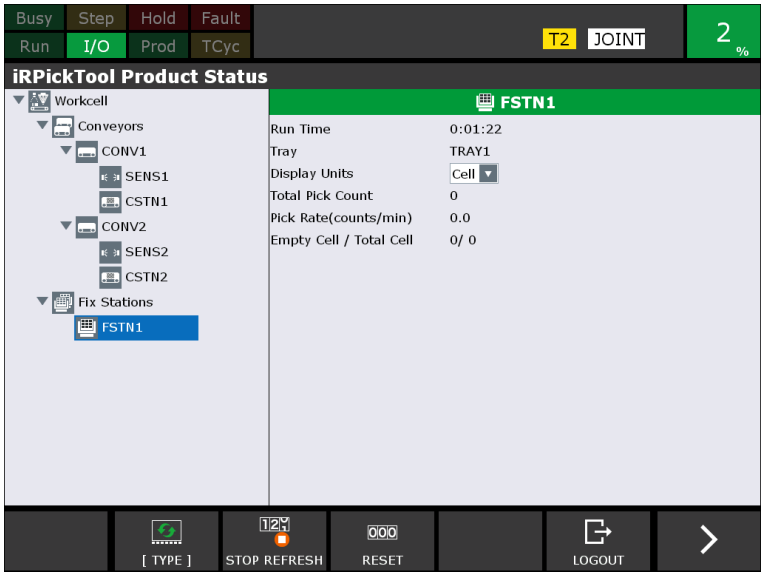
[Model ID]
Model ID of the part or cell. When trays are used, the model ID of the cell set for the tray is displayed. When the vision system is used, instead of trays, the model ID detected by the vision system is displayed.

[X], [Y], [Z], [R]
Current position of the part or cell. This indicates the position as seen from the tracking frame. Z is displayed only when trays are used.

9.2.4 Checking the Status of a Fixed Station

The cell or tray processing status of a fixed station can be checked.

From the tree, select a desired fixed station.



[Run Time]
Production time

[Tray]
Name of the tray placed on the selected fixed station

[Display Units]
Select whether to display the number of conveyed parts on a per-cell or per-tray basis.

[Pick Count]
Total number of cells or trays processed by the selected fixed station

[Pick Rate(counts/min)]
Average number of conveyed cells or trays obtained by dividing the total number of cells or trays processed on this fixed station by the production time.

[Empty Cell / Total Cell]
Number of empty cells among all the cells in the currently processed tray. This represents the unfilled quantity of the tray.

[RESET]
Pressing F3 [RESET] key resets all the data. After the reset, recording starts again.

9.3 LIMITING THE MOVABLE RANGE OF THE FINAL AXIS

When performing a linear motion involving a posture change, the robot moves so as to minimize the rotation angle of each axis. Such a movement is called a minimum rotation. In a visual tracking system where the rotation angle of the final axis changes every time on the pickup and placement sides, the angle of the final axis may move out of the axis movable range as a minimum rotation is repeated, resulting in a stroke limit alarm.

This function prevents the stroke limit of the final axis from being violated by going a long way intentionally in cases when a minimum rotation causes a stroke limit alarm.

The function takes effect when the following conditions are met:
- The tracking type is [Line].
- The line tracking is in progress.
- A linear motion is performed.
- There is a wrist posture.

By default, this function is enabled. Set the following system variables:
- $LNCFG_GRP[group-number].$LMT_CHK_ENB
  This is the flag indicating whether this function is enabled or disabled. To disable the function, set this variable to FALSE.
- $LNCFG_GRP[group-number].$LMT_CHK_UL
  This is the offset (units: degrees) from the upper limit angle of the final axis. Set this variable as necessary.
- $LNCFG_GRP[group-number].$LMT_CHK_LL
  This is the offset (units: degrees) from the lower limit angle of the final axis. Set this variable as necessary.

A change to any of these system variables takes effect immediately.

For example, consider a 6-axis robot where the stroke limit of the J6 axis is –270 to 270 degrees. If you set 90 degrees in both $LMT_CHK_UL and $LMT_CHK_LL, the movable range of the J6 axis is limited to –180 to 180 degrees.

9.4 SKIPPING THE TRACKING MOTION OUTSIDE THE AREA

If [Discard Line] is properly set in the conveyor station settings, the robot judges that it cannot catch up with a part when that part passes the “discard line”. The information about that part is passed to the next conveyor station. However, the robot may be unable to catch up with a part even when the part is upstream of the discard line if the speed override of the robot is low, the conveyor travels at a higher speed than expected, the robot is away from the part, etc. This function causes the robot to stop tracking a part in such cases.

Generally, a tracking program consists of motion instructions for traveling to the three positions shown below.

P1: Approach position
P2: Pickup position
P3: Retraction position

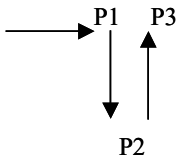


Fig. 9.4(a) Positions for a tracking program

This function determines whether the destination position is within the tracking area during the execution of a motion instruction for traveling to any of the above positions. If it determines that the destination position is outside the tracking area, the function skips the instruction for traveling to that position and proceeds to the next instruction.

To use this function, set the items under [Skip outbound motion] in the robot setup screen.

[Enable]
Select whether to enable or disable the function.

[Adjust skip outbound]
Set the value to be considered when determining whether the destination position is within the tracking area at the start of a motion. The unit is the millisecond.

Suppose that the robot travels to pick a part in the above-mentioned motion pattern. If when it reaches to P1, it is still within the tracking area, but it goes out of the tracking area while it moves from P1 to P2 or it moves from P2 to P3, it is more efficient to go to pick the following part immediately, with giving up this part before going to P1. In such a case, if you set the travel time from P1 to P3 via P2, you can save the unnecessary travel time to move to P1.

For example, measure the maximum travel time from P1 to P3 via P2 with the TIMER instruction. If the measured time is 100 ms, set 100 to [Adjust skip outbound].

Once the robot starts to travel to P1, whether the destination position is within the tracking area is determined at the position reached after only 100 milliseconds of travel. If the destination position is determined to be outside the tracking area, all the motion instructions for P1, P2, and P3 are skipped.

[FLAG Num.]

If a skip occurs, the flag corresponding to the number specified here is turned on. Note that the system software only turns the flag on. If the flag is turned on, it needs to be turned off by a robot program.

NOTE

This function is not effective if the destination position moves out of the tracking area when the robot continues the tracking motion while it waits for the next part after picking a part with a hand capable of holding multiple parts. This is because the continuation of the tracking motion while waiting for the next workpiece is not taken into consideration.
In this case, stop the tracking motion by executing the “STOP_TRACKING” instruction before calling PKCSGETQUE, as described in Subsection 6.9.2.2, “Pick program”.

Sample program

In order to use this function, the tracking program for the pickup and placement motions needs to be modified. An example of the modified program is shown below. Major changes are underlined. The flag number is 1, and the time required to arrive from P1 to P3 via P2 is 100 ms.

[FLAG Num.] = 1
[Adjust skip outbound] = 100 (ms)

PICK1.TP

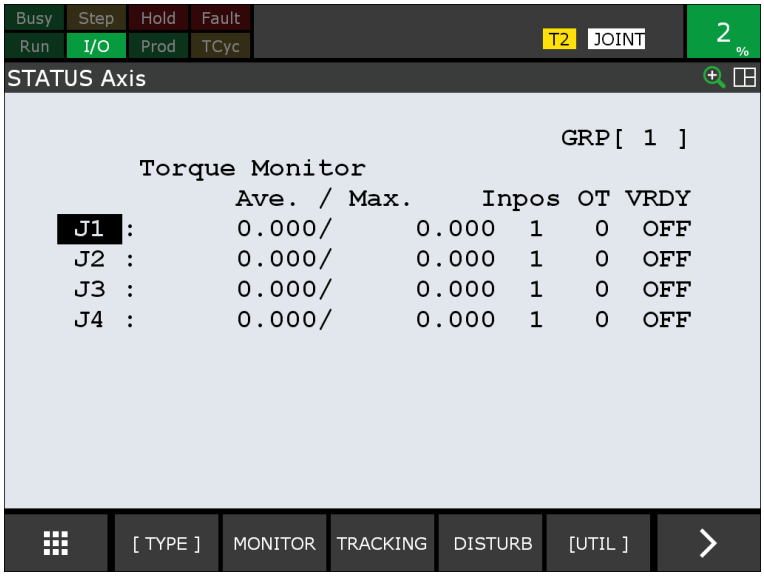
```
   1:   UTOOL_NUM=1
   2:   UFRAME_NUM=0
   3:   R[3:ZONE]=1
   4:
   5:   LBL[100]
   6:   STOP_TRACKING
   7:   F[1]=(OFF)
   8:   CALL PKCSGETQUE(CStn ID=R[11:CSTN1],Consec Flag=R[3:ZONE],
        Timeout (ms)=100,Offset VR=1,Stat Reg=2)
   9:   IF R[2:GETQ_STATUS]=0,JMP LBL[200]
  10:   IF R[1:CYCLE_STOP]=1,JMP LBL[900]
  11:   JMP LBL[100]
  12:
  13:   LBL[200]
  14: L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
  15:   IF (F[1]=ON),JMP LBL[300]
  16: L PR[R[3:ZONE]] 4000mm/sec CNT3 VOFFSET,VR[1]
  17:   IF (F[1]=ON),JMP LBL[300]
  18:   CALL VACUUM_ON
  19: L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
  20:
  21:   CALL PKCSACKQUE(CStn ID=R[11:CSTN1],Success)
  22:   R[3:ZONE]=R[3:ZONE]+1
  23:   IF R[3:ZONE]<=R[4:NUM_ZONE],JMP LBL[100]
  24:   JMP LBL[900]
  25:
  26:   LBL[300]
  27:   CALL PKCSACKQUE(CStn ID=R[11:CSTN1],Skip)
  28:   JMP LBL[100]
  29:
  30:   LBL[900]
[End]
```

If the destination position is determined to be outside the tracking area, specify “Skip”(3) in PKCSACKQUE to report the failure to pick the part and have a downstream robot pick that part.
Turn the flag off explicitly in the program.

9.5 CHECKING THE ROBOT TORQUE

In the event of a motor overheat, the robot stops as a result of an alarm. Check the torque applied to each axis during a tracking motion to see whether the motor is not overheating.
You can display the torque monitor screen as follows.

1 Press the [MENU] key on the teach pendant.
2 Select [NEXT].
3 Select [STATUS].
4 Press the F1 [TYPE] key, and select [Axis] from the options.
5 Press the [NEXT] key and then F2 [MONITOR] key. A screen as shown below is displayed.



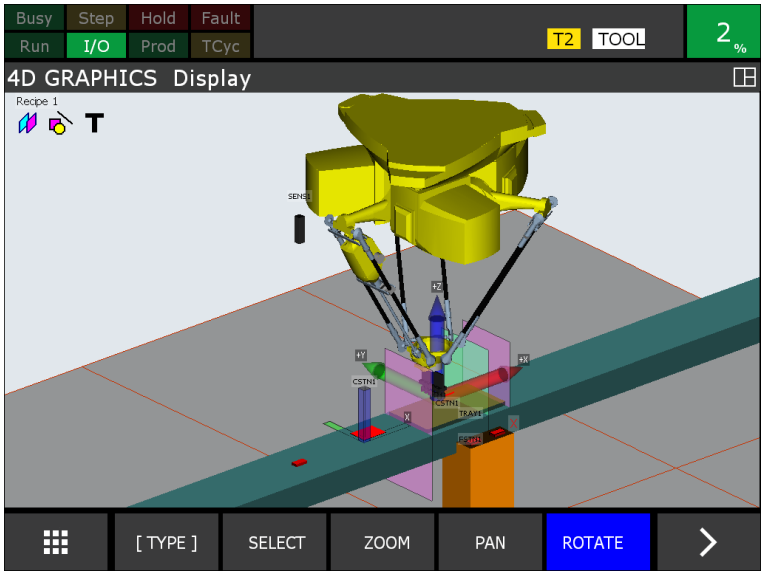
If the [Ave.] value for each axis displayed here increases gradually while the system is continuing constant operation, there is a risk of a motor overheat. If so, decrease the speed of the tracking motion.

9.6 *i*RPickTool 4D GRAPHICS DISPLAY

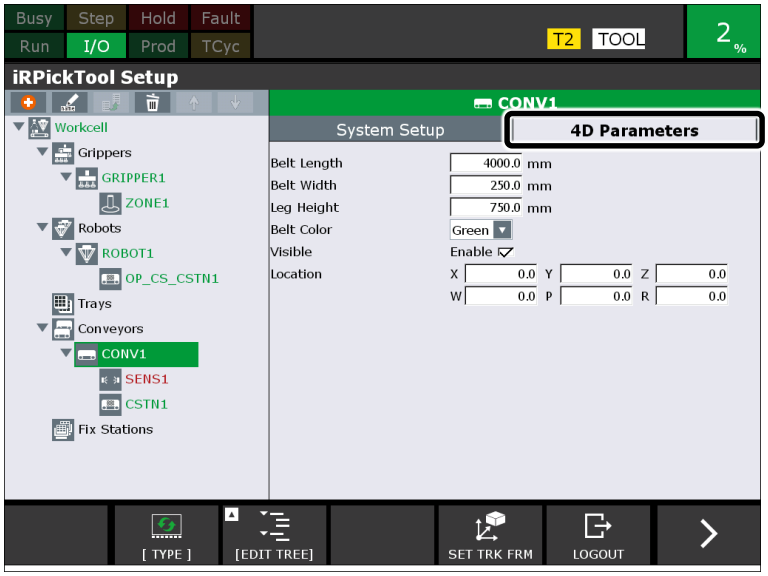
This chapter describes the setup and use of the R764: 4D Graphics option.

When the R764: 4D Graphics option is installed, you can visualize all your workcell objects in 3-dimensions on the *i*Pendant. It provides a visual confirmation of whether the setup was done accurately or not. The option allows you to visualize the robots, the grippers, the Line Tracking frames, the Line Tracking upstream and downstream boundaries, the discard boundaries, the tray with its cells and the locations of your Fixed Stations.

The figure below illustrates what you see in the 4D Display menu on the Teach Pendant when you have setup *i*RPickTool completely for a 2-robot workcell.



The 4D Setup is done in the *i*RPickTool treeview. When the 4D Graphics (order number R764) is installed, the right pane of the treeview will contain a “4D Parameters” tab for some objects such as the Workcell, Gripper, Robot, Tray, Conveyor, Conveyor Station, Sensor and the Fixed Station. The 4D tab contains the properties that you can change for the selected object in the User Interface tree (UI tree). The typical 4D properties you modify are: dimensions of objects, color, transparency, visibility and location. The figure below shows the 4D Parameters tab for a Conveyor object.



9.6.1 Displaying the 4D Workcell

Once you change the 4D properties of an object, you will then need to go to the [MENU] key – 4D Graphics – 4D Display menu on the *i*Pendant. In the 4D Display menu, your workcell will be displayed in 3D and you can see the effect of the change you made.

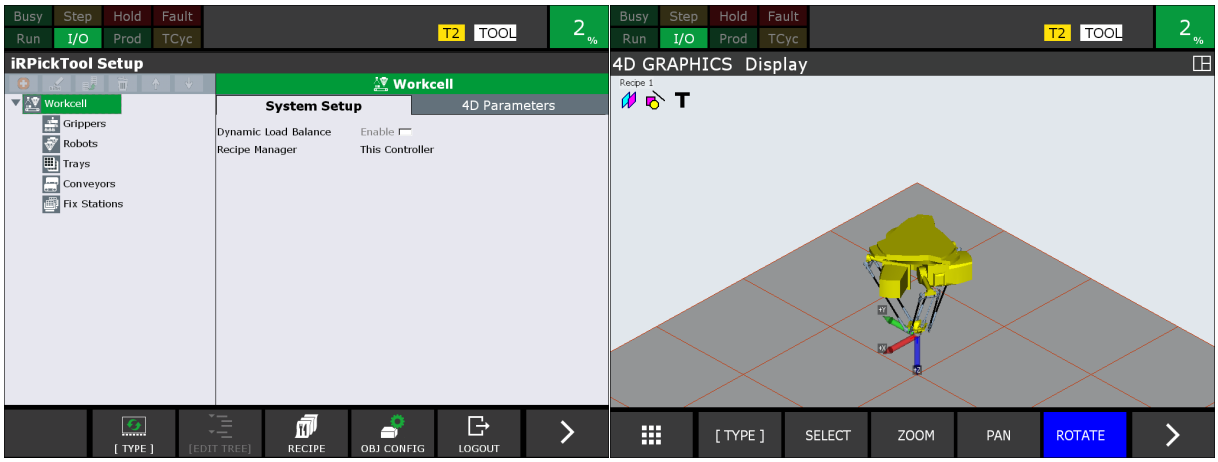
9.6.2 Creating the 4D Workcell

Empty Workcell

When you have the R764: 4D Graphics option installed, and you have done no setup of *i*RPickTool in the treeview, then when you go to the 4D Display menu, you will see the empty workcell.

The empty workcell contains at least one robot – the robot that the teach pendant is connected to. If your workcell has multiple robots and you have done RIPE Setup, then you will see all the robots in the RIPE ring. However, the robots may not be spaced correctly. They may all show up as being on top of each other at the same workcell origin. The robots will be spaced correctly only after you complete the track frame setup using the track guide in the Conveyor node of the treeview.

Any empty workcell contains no other *i*RPickTool objects such as the gripper, the conveyor or the fixed stations. See the empty workcell in the figure below.

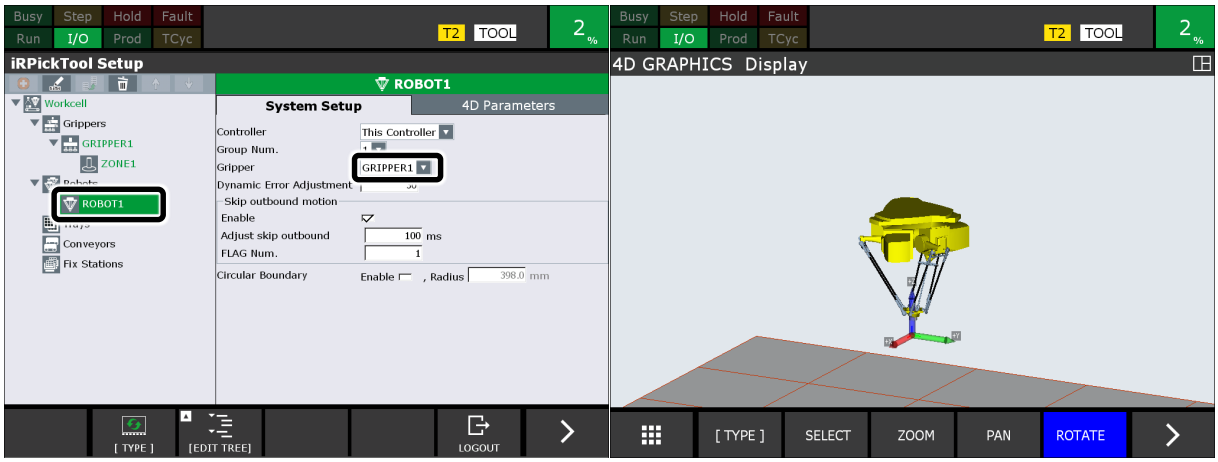


Creating 4D Workcell Objects

You create 4D Workcell Objects in the 4D Display menu automatically when you create that object in the treeview as you go through the setup steps.

Creating a new 4D Gripper

When you create a gripper in the treeview and assign it to a robot as shown in the figure below, the 3D gripper is created automatically. You can see the gripper on the end-of-arm and also a text display above the robot showing the name of the robot.



Modifying the 4D Gripper

In the tree view on the left side, press the gripper – for example, GRIPPER1 - to setup in 4D. All the 4D Gripper properties that you can modify are shown in the right pane. The properties allow you to control the size, color, visibility and location of the gripper CAD file relative to the faceplate of the robot. You cannot change the CAD file of the gripper from the 4D Gripper menu. See Subsection 9.6.7, “Loading Custom CAD Files”.

[Length]

The length of the gripper in millimeters.

[Width]

The width of the gripper in millimeters.

[Height]
The height of the gripper in millimeters.

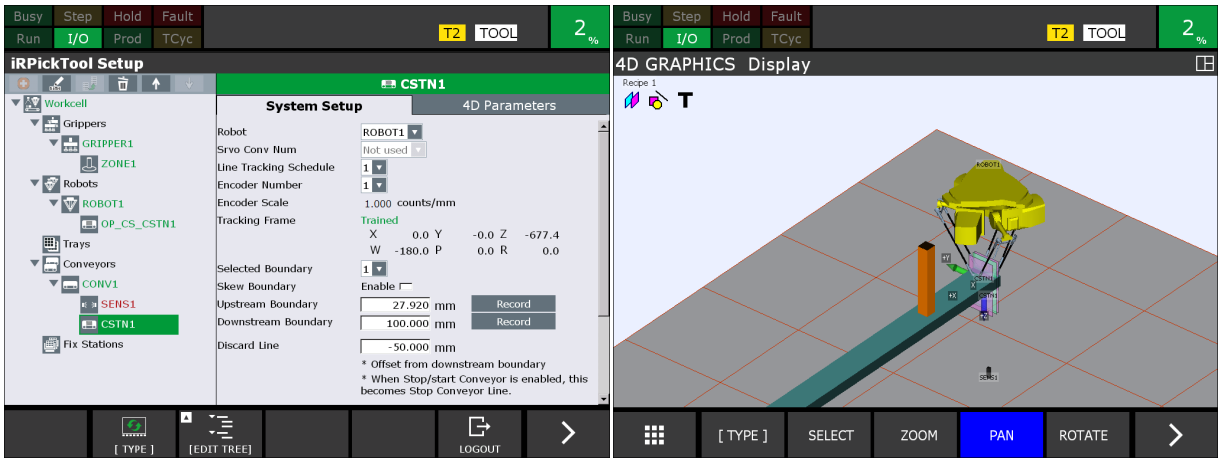
[Color]
The color of the gripper. Select a color from the drop-down box.

[Visible]
Enable or disable the visibility of the gripper using the checkbox.

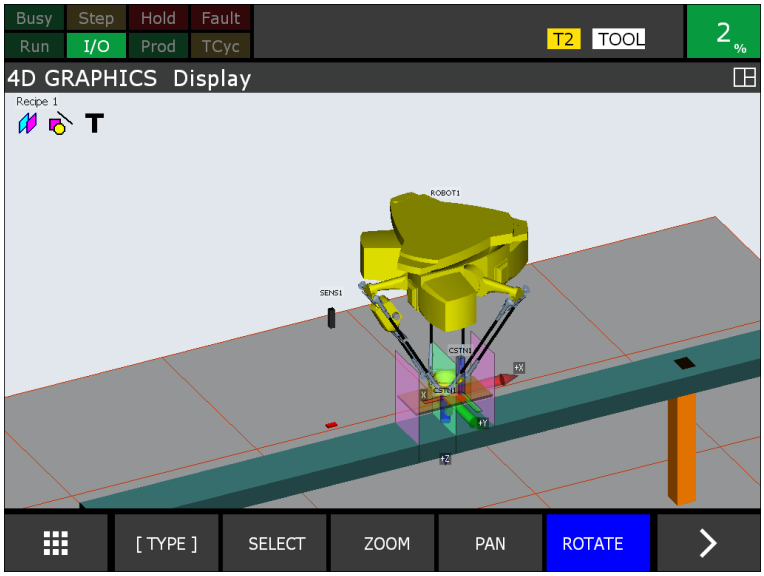
[Location]
The x, y, z, w, p, r offset from the faceplate of the gripper where the origin of the CAD file is located. For the default gripper CAD file provided, leave all the values at zero.

Creating a new 4D Conveyor
When you create a conveyor in the treeview, a 3D conveyor is automatically created with a default size of 400 mm x 250 mm x 100 mm in the 4D Display menu as shown in the figure below. The origin of the 3D conveyor is at the top center of the belt. By default, *i*RPickTool aligns the CAD origin of the 3D conveyor to the track schedule of the first conveyor station of the conveyor. If the track frame has not been taught, its value is nilpos which is the same as the robot's world zero position. Therefore, you will see the conveyor running through the robot zero position when you first create it. Later when you use the track guide to teach the track frame, the conveyor will move to the proper location.
When you create a new conveyor in the treeview, it also creates a sensor and a conveyor station by default. The 4D display menu will also attempt to display the sensor and the conveyor station. However, the conveyor station itself will not be displayed in the 4D Display menu until you assign a valid Line Tracking Schedule in the conveyor station's tree view menu. When you assign a track schedule, a conveyor station is created on the conveyor with a brown rectangular base and three boundaries – the track upstream boundary, the track downstream boundary and the discard boundary. The upstream nor the downstream boundary are displayed in violet color. The discard boundary in spring green color is offset from the downstream boundary by a distance of -100 mm by default.

The sensor is displayed as a black rectangle box of size 25 mm x 25 mm x 100 mm at a default height of 900 mm from the track frame of the first conveyor station of the conveyor.

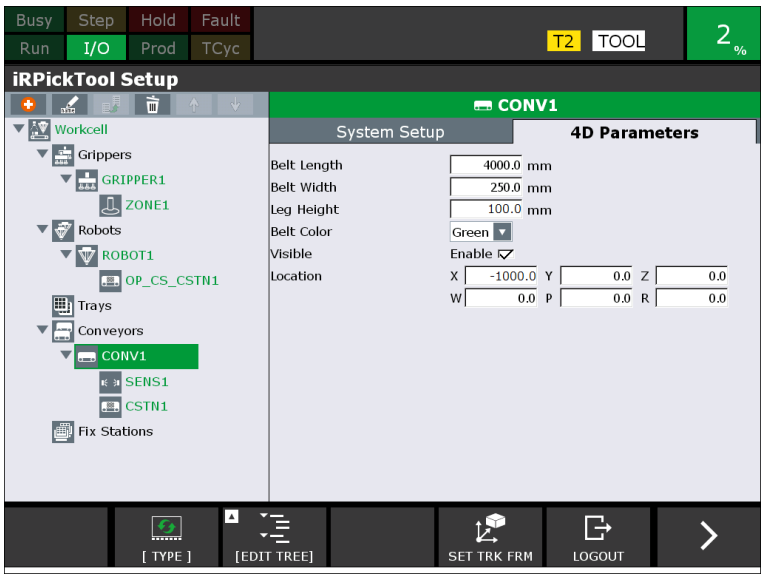


If you later go through the track guide in the treeview and teach the track frame properly, and you go through the Sensor Guide in the Sensor menu, then the conveyor, the sensor and the conveyor station will all be located correctly in the workcell as shown in the figure below.



Modifying a 4D Conveyor

In the tree view on the left side, press the conveyor– for example, CONV1 – to setup in 4D. All the 4D Conveyor properties that you can modify are shown in the right pane. The properties allow you to control the size, color, visibility and location of the Conveyor CAD file. You cannot change the CAD file of the conveyor from the 4D Conveyor menu. See Subsection 9.6.7, “Loading Custom CAD Files”.



[Belt Length]

The length of the conveyor belt in millimeters.

[Belt Width]

The width of the conveyor belt in millimeters.

[Leg Height]

The height of the conveyor leg in millimeters.

[Belt Color]

The color of the conveyor belt. Select a color from the drop-down box.

[Visible]

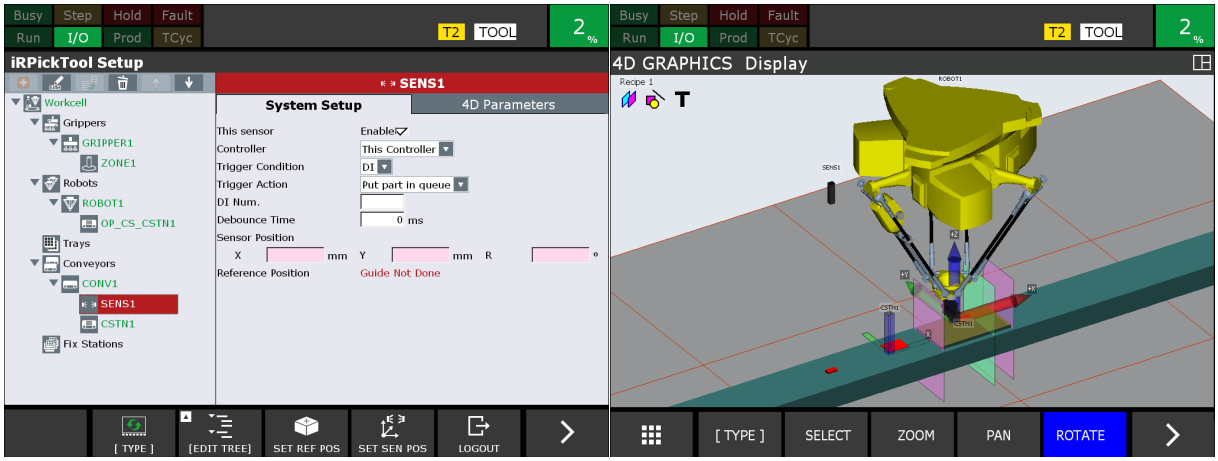
Enable or disable the visibility of the conveyor using the check box.

[Location]

The x, y, z, w, p, r offset of the origin of the 3D conveyor from the track frame of the first conveyor station of the conveyor. The origin of the CAD data is at the top center of the belt.

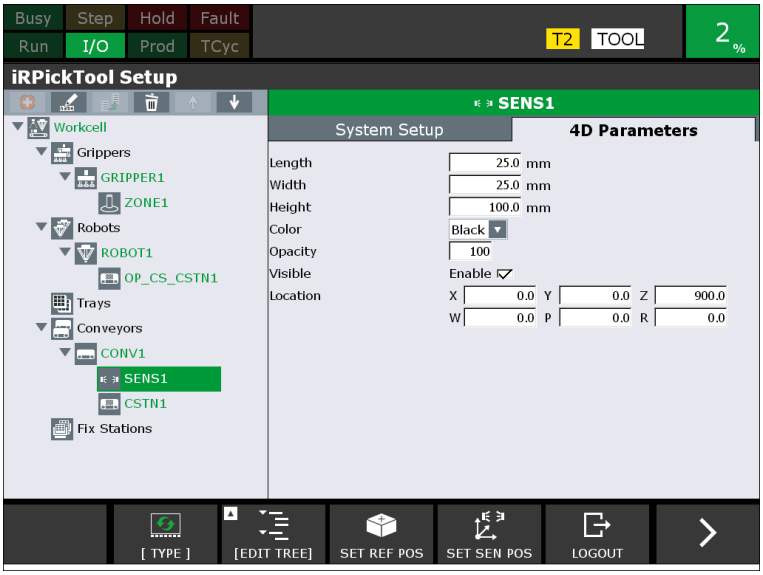
Creating a new 4D Sensor

When you create a Sensor in the treeview, a 3D Sensor is automatically created as a rectangular box with a default size of 25 mm x 25 mm x 100 mm in the 4D Display menu as shown in the figure below. The origin of the Sensor is at the center of the rectangular box. By default, *i*RPickTool offsets the CAD origin of the 3D Sensor from the track frame of the first conveyor distance so that it is right above it at a height of 900 mm. When you complete the sensor guide step in the tree view, the sensor will be located at the proper x, y offset from the track frame. The height at which it is displayed will continue to be 900 mm unless you change it in the 4D Sensor menu.



Modifying the 4D Sensor

In the tree view on the left side, press the Sensor – for example, SENS1 – to setup in 4D. All the 4D Sensor properties that you can modify are shown in the right pane. The properties allow you to control the size, color, visibility and location of the Conveyor CAD file. You cannot change the CAD file of the Sensor from the 4D Fixed Station menu. See Subsection 9.6.7, “Loading Custom CAD Files”.



All the properties that you can modify are shown in the right pane.

[Length]
The length of the Sensor box in millimeters.

[Width]
The width of the Sensor box in millimeters.

[Height]
The height of the Sensor box in millimeters.

[Color]
The color of the Sensor. Select a color from the drop-down box.

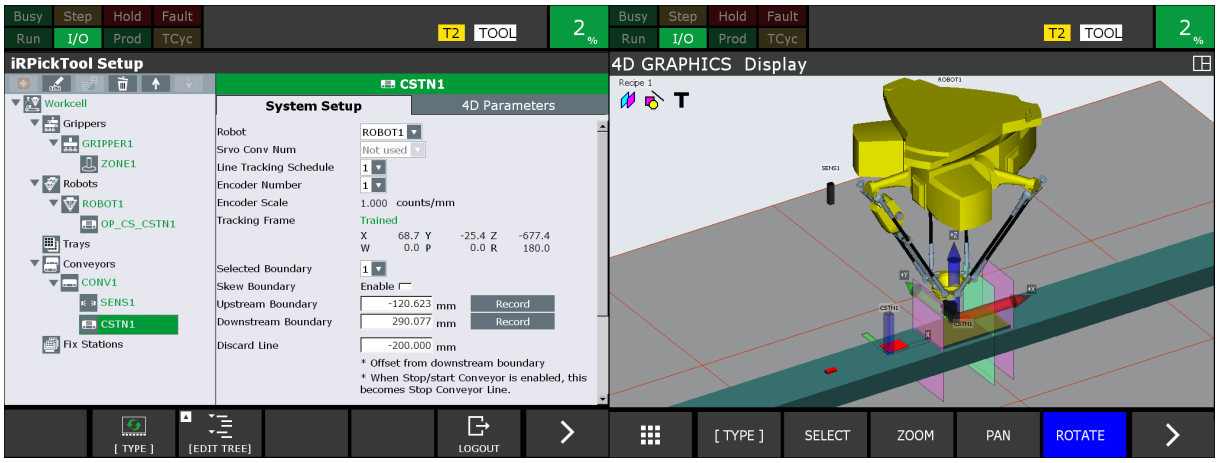
[Opacity]
The opacity of the Sensor box. Enter a value in the range from 0-100 where 0 is full transparent and 100 is fully opaque.

[Visible]
Enable or disable the visibility of the Sensor box using the checkbox.

[Location]
The x, y, z, w, p, r offset from the track frame of the first conveyor Station that the Sensor should be drawn.

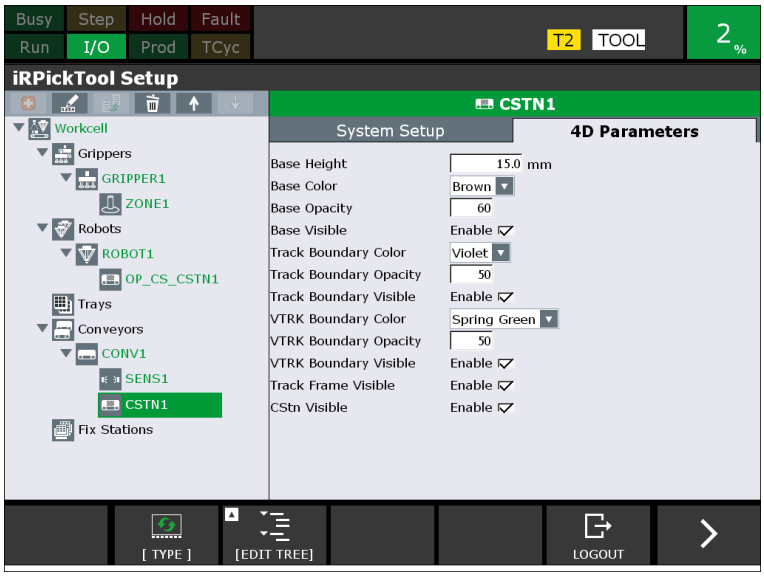
Creating a new 4D Conveyor Station

When you create a Conveyor Station in the treeview, a 3D Conveyor Station is automatically created as follows:
The tracking frame of the conveyor is displayed on the conveyor as a thin rectangular box with a triad on it.
The upstream boundary and the downstream boundary are displayed on the conveyor in Violet color.
The discard boundary is displayed in Spring Green color.
The tracking area between the upstream and downstream boundaries is displayed in brown color on the conveyor.



Modifying the 4D Conveyor Station

In the tree view on the left side, press the Conveyor Station – for example, CSTN1 – to setup in 4D. All the 4D Conveyor Station properties that you can modify are shown in the right pane. The properties allow you to control the size, color, opacity and visibility of the tracking area, the track frame and the tracking boundaries. You can make the entire conveyor station visible or invisible.



All the properties that you can modify are shown in the right pane.

[Base Height]

The tracking or picking area is represented by a rectangular box. The area is referred to as “Base” in 4D terminology. The base height represents the height of the box in millimeters.

[Base Color]

The color of the box representing the tracking area or base. Select a color from the drop-down box.

[Base Opacity]

The opacity of the box representing the tracking area or base. Enter a value in the range from 0-100 where 0 is full transparent and 100 is fully opaque.

[Base Visible]

Enable or disable the visibility of the tracking area box or base.

[Track Boundary Color]

The color of the tracking boundaries. Both the upstream and the downstream boundaries are represented by the same color. Select a color from the drop-down box.

[Track Boundary Opacity]

The opacity of the walls representing the tracking boundaries. Enter a value in the range from 0-100 where 0 is full transparent and 100 is fully opaque.

[Track Boundary Visible]

Enable or disable the visibility of the tracking upstream and downstream boundaries (together).

[VTRK Boundary Color]

The color of the discard boundary. Select a color from the drop-down box.

[VTRK Boundary Opacity]

The opacity of the wall representing the discard boundary. Enter a value in the range from 0-100 where 0 is full transparent and 100 is fully opaque.

[VTRK Boundary Visible]

Enable or disable the visibility of the discard boundary.

[Track Frame Visible]

Enable or disable the visibility of the track frame of this conveyor station.

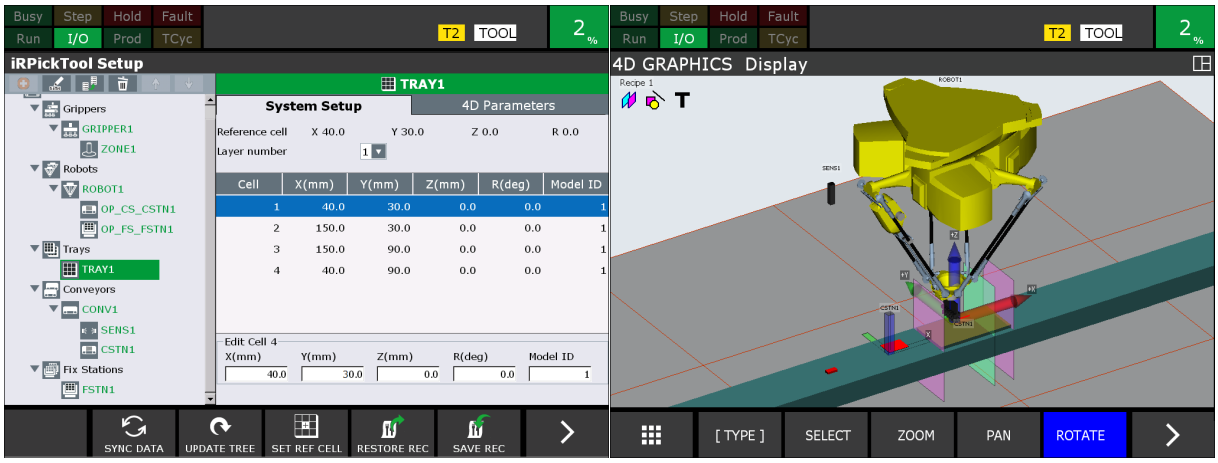
[CStn Visible]

Enable or disable the visibility of the entire conveyor station. The entire conveyor station includes the tracking frame, the tracking upstream and downstream boundaries, the discard boundary and the base of the Conveyor station.

Creating a new 4D Tray

When you create a tray in the treeview and assign it to a Fixed Station as shown in the figure below, the 3D tray cells are created automatically at the Fixed Station. You can also assign trays to conveyors in the conveyor menu of the treeview. The tray cells are always drawn in the tray frame. For a Fixed Station tray, the tray frame is the same as the Uframe of the Fixed station. For a conveyor station tray, the tray frame is calculated by the sensor guide in the treeview as an offset from the track frame of the first conveyor station of the conveyor.

A tray will be visible in the 4D scene only if it is assigned to either a Fixed Station or a Conveyor Station.



Modifying the 4D Tray

In the tree view on the left side, press the tray – for example, TRAY1 – to setup in 4D. All the 4D Tray properties that you can modify are shown in the right pane. The properties allow you to assign a part id to the tray and control the visibility of the tray frame.

All the properties that you can modify are shown in the right pane.

Part

A tray contains one or more cells. The part that is used at each cell location can be selected in the drop-down box. In *i*RPickTool, there is no User interface to setup the part properties. You can setup the size and color of the part from the system variable $pkpt4d.

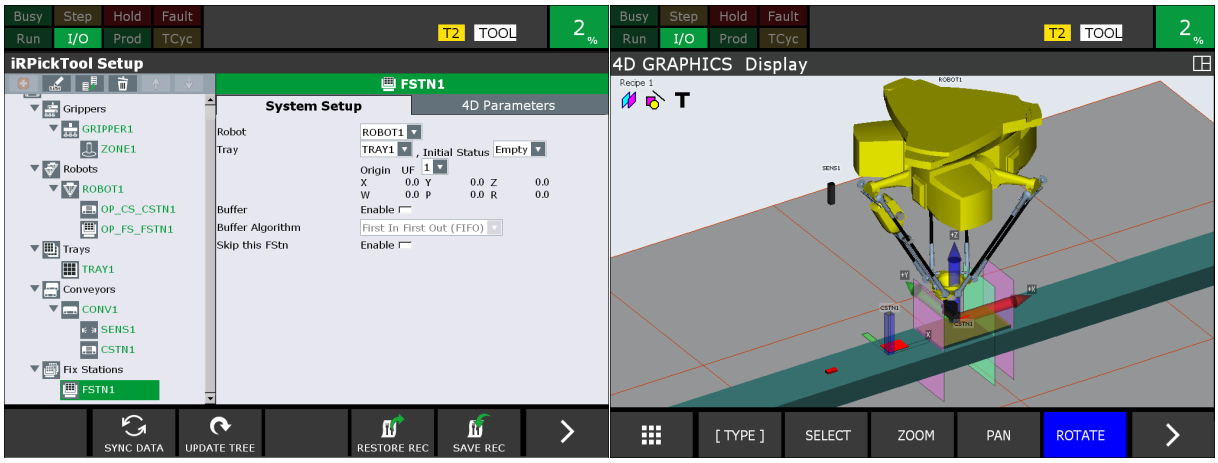
Frame Visible

Enable or disable the visibility of the Tray Frame using the checkbox. You will be able to view the names of the trays only when the tray frame is made visible.

Creating a new 4D Fixed Station

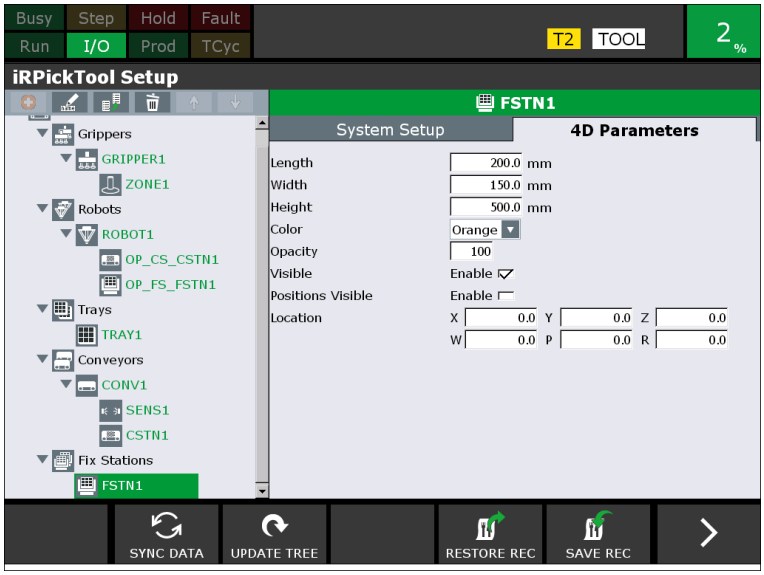
When you create a Fixed Station in the treeview, a 3D Fixed Station is automatically created as a rectangular box with a default size of 200 mm x 150 mm x 500 mm in the 4D Display menu as shown in the figure below. The origin of the 3D Fixed Station is at the center of the rectangular box. By default, *i*RPickTool aligns the CAD origin of the 3D Fixed Station to the Uframe number you specified for the Fixed Station in the treeview menu. If the Uframe has not been taught, its value is nilpos which is the same as the robot's world zero position. Therefore, you will see the Fixed Station running through the robot zero position when

you first create it. Later when you teach the Uframe correctly, the Fixed Station will be located correctly in the 3D workcell.



Modifying the 4D Fixed Station

In the tree view on the left side, press the Fixed Station– for example, FSTN1 – to setup in 4D. All the 4D Fixed Station properties that you can modify are shown in the right pane. The properties allow you to control the size, color, visibility and location of the Conveyor CAD file. You cannot change the CAD file of the Fixed Station from the 4D Fixed Station menu. See Subsection 9.6.7, “Loading Custom CAD Files”.



All the properties that you can modify are shown in the right pane.

[Length]

The length of the Fixed Station box in millimeters.

[Width]

The width of the Fixed Station box in millimeters.

[Height]

The height of the Fixed Station box in millimeters.

[Color]

The color of the Fixed Station. Select a color from the drop-down box.

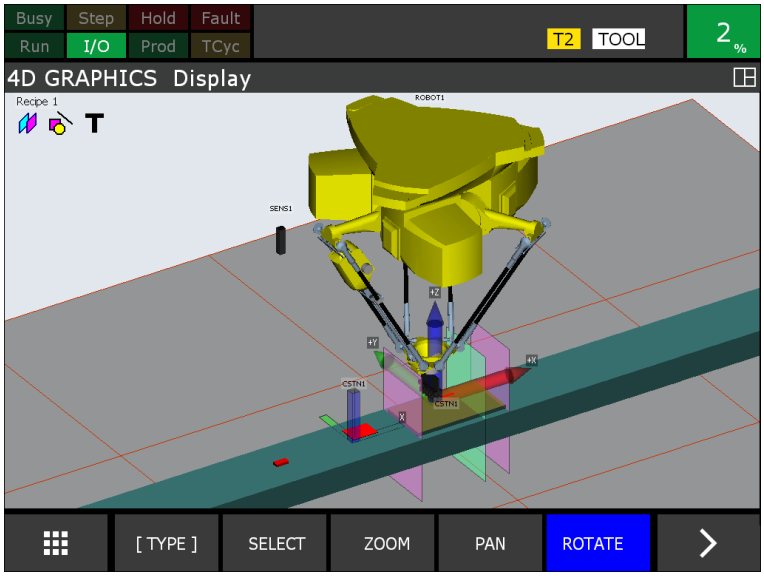
[Opacity]
The opacity of the Fixed Station. Enter a value in the range from 0-100 where 0 is full transparent and 100 is fully opaque.

[Visible]
Enable or disable the visibility of the Fixed Station box using the checkbox.

[Positions Visible]
Enable or disable the checkbox to view the approach and pick / place position of the Fixed Station in the Tray frame.

[Location]
The x, y, z, w, p, r offset from the UFrame of the Fixed Station that the Fixed Station should be drawn.

When you teach the Uframe correctly, your 4D display will look similar to the following:



9.6.3 Using the Touch Screen Icons

There are 3 icons in the top left hand corner of the 4D Display menu. The icons are shown in the figure below.

Clicking on the icon when the *i*Pendant has a touch screen will allow you to perform some operations related to the 4D scene quickly. In order to make the TP screen enabled for touch, you need to first click on F2[SELECT] key. Each icon acts like a toggle. Each icon has an enabled and disabled state.

Boundary Show/Hide Icon

The Boundary icon is meant for showing and hiding all the boundaries on all the conveyors. The boundaries are Track Upstream boundary, Track Downstream boundary and Discard boundary.

*i*RPickTool Show/Hide Icon

The *i*RPickTool icon is meant for showing and hiding all the *i*RPickTool visualizations. You may hide the *i*RPickTool visualizations when you want to get a clearer view of some other general purpose 4D Graphics function like “Visual Jog”, “4D Edit Node Map”, or “4D Position Reg” or “4D *i*RVision”.

Text Show/Hide Icon

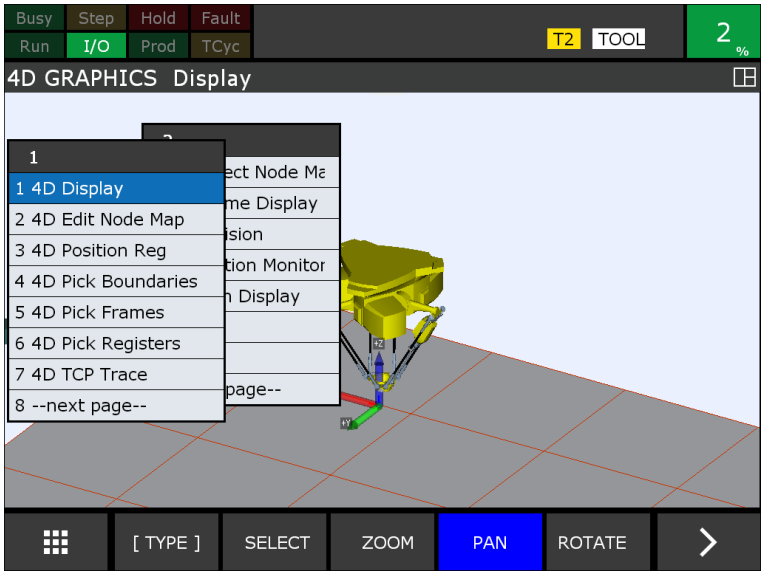
The Text icon is meant for showing and hiding Text entries in the 4D scene. *i*RPickTool allows you to display the names of the Robots, conveyor stations, fixed stations, sensors, tray frames and track frames.

9

9.6.4 Using the 4D [TYPE] Scenes

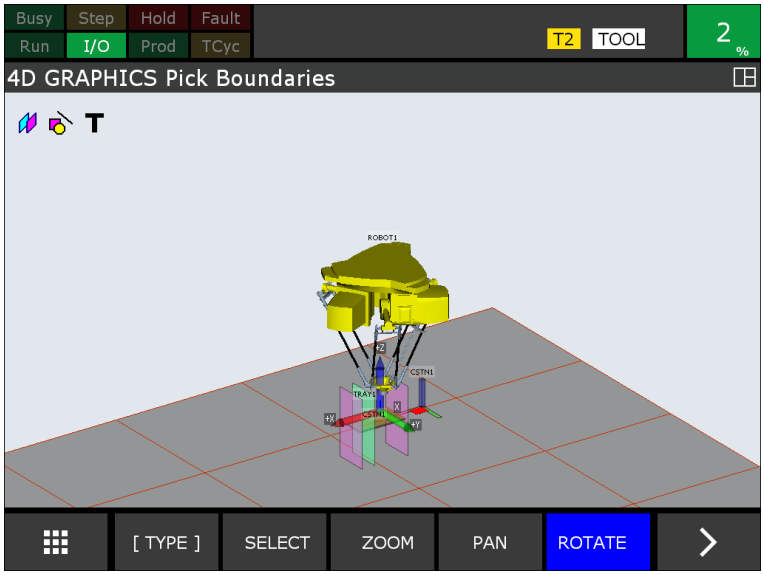
In the 4D Display menu, the F1 function key is assigned to [TYPE]. When you click on F1 [TYPE] key, a subwindow will open up containing additional scenes that you can view as shown below. You can explore all the scenes, but the following three scenes may be useful to you check your setup:

- 4D Pick Boundaries
- 4D Pick Frames
- 4D Pick Registers (Not implemented).



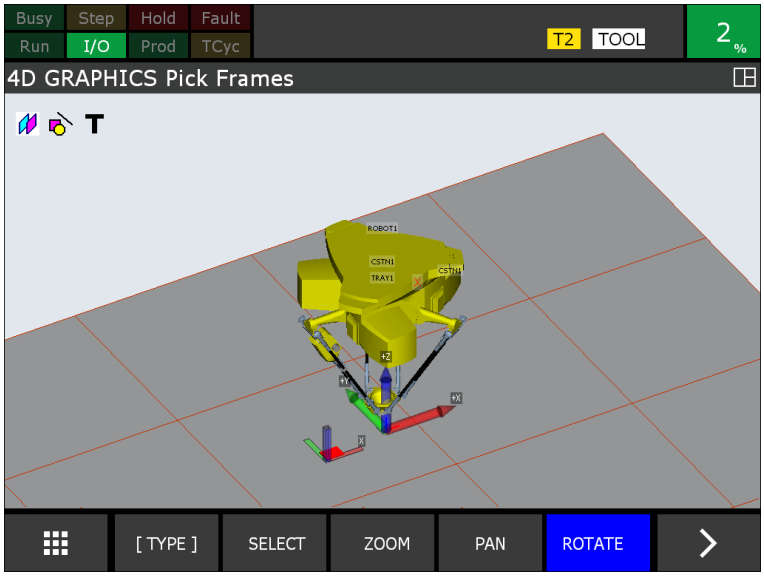
9.6.4.1 4D pick boundaries

When you press the 4D Pick Boundaries scene, you will see a 3D scene containing only the boundaries and track frames on all the conveyors. This helps you to confirm the boundaries relative to the robots and also the location of your track frames.



9.6.4.2 4D pick frames

When you click on the 4D Pick Frames scene, you will see a 3D scene containing only the track frames on all the conveyors and the Fixed Station Frames. This helps you to confirm if you have taught all the frames and if they appear to be in the proper locations.



9.6.5 Deleting the 4D Workcell

By deleting all objects from the *i*RPickTool tree view, you can delete the whole 4D workcell.
You can delete individual objects such as sensors and conveyors from the tree view, but you cannot delete the robot from the 4D graphic display.

If you wish to temporarily turn off the visualizations of the *i*RPickTool objects you can use the *i*RPickTool Show/Hide icon  .

9.6.6 Changing the Workcell Scene Using Recipes

When you load a recipe, the workcell scene will display the objects based on the setup data you saved for the recipe. All the objects and object locations will be based on the data in the recipe.

The 4D display menu always displays the name of the Active Recipe being displayed on the top left hand corner of the screen above the *i*RPickTool icons. See figure below. The text “PANINI ROUND 2 SLICES” is the name of the active recipe.

9.6.7 Loading Custom CAD Files

By default, *i*RPickTool creates primitive shapes or utilizes its own CAD files for the workcell objects. It does not allow you to change them.

However, if you wish to use your own CAD files to create realistic 4D scenes, then you can do so by using a FANUC product called “ROBOGUIDE 4D Editor”. ROBOGUIDE 4D Editor allows you to create your workcell and then download it to the controller.

If you use the ROBOGUIDE 4D Editor, then you want to make sure that *i*RPickTool will not try to create the physical objects in the workcell such as the grippers, sensor boxes, conveyors and Fixed Station boxes. To prevent *i*RPickTool from creating the physical objects, go to the Workcell menu in the treeview and then click on the 4D Parameters tab in the right side of the screen. You will need to disable the “Display physical objects” property. Once you do this, *i*RPickTool will only create logical objects such as conveyor stations with boundaries or tray cells. See figure below.

9.6.8 Limitations

Do not use the 4D Graphics for *i*RPickTool if you have any of the cases below. The objects will not be displayed correctly.

- If you have more than four robots in your workcell.
- If you have more than one infeed or more than one outfeed conveyor.
- If you have the circular tracking.

9.7 SERVO CONVEYOR LINE TRACKING FUNCTION

9.7.1 OVERVIEW

Servo Conveyor Line Tracking function is the function for using an extended axis as a conveyor. Therefore, the robot can track the conveyor with high accuracy.
To use the conveyor as an index conveyor, order the optional index-servo conveyor-tracking function. To use the conveyor as a continuous conveyor, order the optional continuous-servo conveyor-tracking function. This function requires Line Tracking option and Servo Conveyor Line Tracking option. In addition, Independent Auxiliary Axis option is necessary to control extended axis.
Servo Conveyor Line Tracking option includes Multi Motion Group option and Continuous Turn option. These are used for keeping on moving an extended axis as a conveyor.

9.7.2 SETUP

Please set up Servo Conveyor Line Tracking system by the following step.

9.7.2.1 Independent Extended Axis Setup
9.7.2.2 Servo Conveyor Setup
9.7.2.3 How to Create TP Program for Servo Conveyor
9.7.2.4 Tracking Schedule Setup
9.7.2.5 Example of Main Program

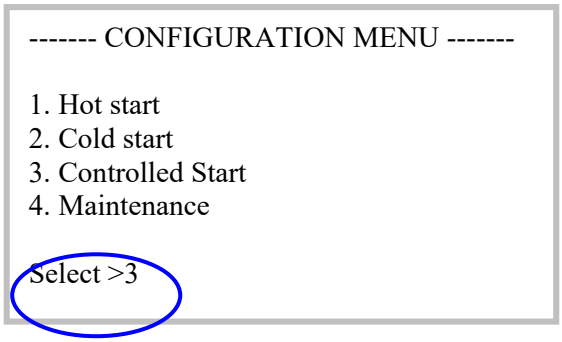
9.7.2.1 Independent Extended Axis Setup

In Servo Conveyor Line Tracking function, you can use Independent Extended Axis as a conveyor. Please setup Independent Extended Axis on ROBOT MAINTENANCE MENU at Control Start.

Procedure: Controlled Start

Step

1 While holding the [PREV] key and the [NEXT] key, turn on the power. After a while, you will see a screen as following.

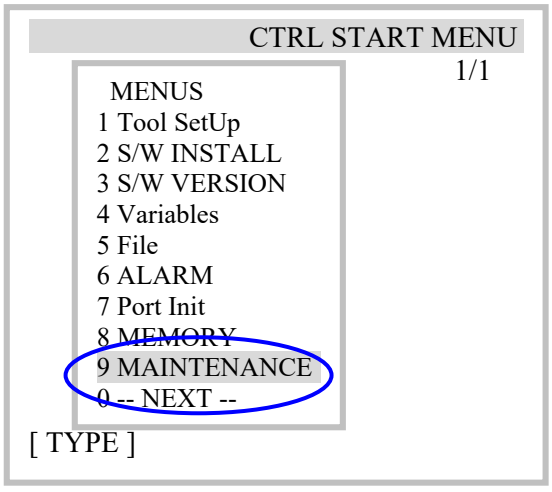


2 Select “3. Controlled Start” and press the [ENTER] key. After a while a Control Start Menu appears.

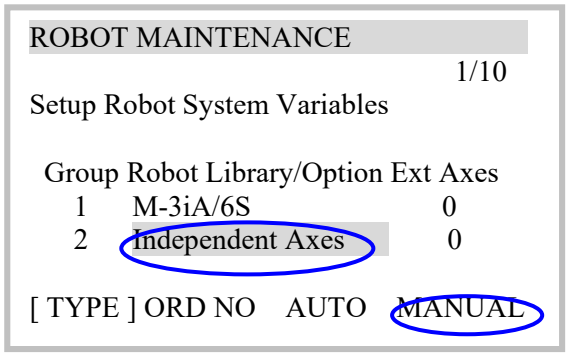
Procedure: Robot Maintenance

Step

1 Press the [MENU] key to display the screen menu. Select [MAINTENANCE] and press the [ENTER] key. ROBOT MAINTENANCE screen appears.



2 Select [Independent Axes] and press F4 [MANUAL] key.



3 "Independent Axes Setup Menu" appears. If you add axis, please select [Add Axis]. Press "2" key and the [ENTER] key and then "MOTOR SELECTION" appears. When axis setup is finished, you return to "Independent Axes Setup Menu". If you finish the axis setup, please select [4 EXIT].

```
*** Group 2 Total Axes Installed =    0
  1. Display/Modify Nobot Axis 1~4
  2. Add Axis
  3. Delete Axis
  4. EXIT
Select Item? 2
```

NOTE
Please refer to the mechanical specification for the following procedure.

4 Select a servo motor which is used as Independent Axis. Select [Standard Method] and then select MOTOR SIZE and MOTOR TYPE.

```
-- MOTOR SELECTION

1: Standard Method
2: Enhanced Method
3: Direct Entry Method
Select ==> 1
```

```
MOTOR SIZE (Beta standard, Beta is)

80. biS0.2      84. biS1      88. biS12
81. biS0.3      85. biS2      89. biS22
82. biS0.4      86. biS3
83. biS0.5      87. biS6
   0. Next page
Select ==> 85
```

```
          MOTOR TYPE

1. /2000          11. /4000
2. /3000          12. /5000
       13. /6000
Select ==> 11
```

For example, if you would like to select β*i*S2/4000*i*, please select 85 and 11.

5 Select a current limit for amplifier. If current limit for amplifier is 20A, please select 10.

```
    CURRENT LIMIT FOR AMPLIFIER

2.     4A          10.    20A
5.    40A          12. 160A
80A

Select==> 10
```

6 Select [2: Rotary Axis] as Independent axis type.

```
-- INDEPENDENT AXES TYPE –
1. Linear Axis
2. Rotary Axis

Select? 2
```

7 Enter Gear Ratio.

```
-- GEAR RATIO --
Enter Gear Ratio?
```

(a) In the case of setting a servo conveyor as [Index] on 7DC3 series and later
Enter the number of revolution of the motor which corresponds to one pitch of the conveyor. By this setting, a conveyor moves one pitch when Independent Axis moves 360 degrees.
Pitch: The distance of conveyor when move conveyor to one bucket of conveyor
Bucket: A part of a conveyor is divided by a constant distance.

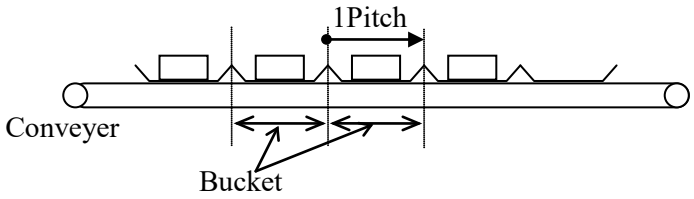


Fig. 9.7.2.1 (a) Bucket and Pitch

If a conveyor rotates “N” times and the motor of it rotates “M” times, the gear ratio can be calculated by the following. Please enter the calculated value.

$$\text{GearRatio} = \frac{\text{M}}{\text{N x Number of Buckets}}$$

- It is also possible to calculate the gear ratio from the set value (Motor Gear teeth and so on) of Servo Conveyor Setup. In this case, the gear ratio can be calculated by the following. See Subsection 9.7.2.2.

$$\text{GearRatio} = \frac{\text{Rotor Input Gear Teeth x Conveyer Belt Teeth}}{\text{Motor Gear Teeth x Rotor Output Gear Teeth x Number of Flight}}$$

(b) In the case of setting a servo conveyor as “Continuous”.
Enter the number of revolution of the motor which corresponds to a belt length of section of the conveyor. By this setting, a conveyor moves the belt length of section when Independent Axis moves 360 degrees.
If a conveyor rotates “N” times and the motor of it rotates “M” times, the gear ratio can be calculated by the following. Please enter the calculated value.

$$\text{GearRatio} = \frac{\text{M x Belt Section Teeth}}{\text{N x Belt Teeth of the conveyor}}$$

- It is also possible to calculate the gear ratio from the set value (Motor Gear teeth and so on) of Servo Conveyor Setup. In this case, the gear ratio can be calculated by the following. See Subsection 9.7.2.2.

$$\text{GearRatio} = \frac{\text{Rotor Input Gear Teeth x Belt Section Teeth}}{\text{Motor Gear Teeth x Rotor Output Gear Teeth}}$$

8 Select [2:NO Change] for setting a suggested speed as a max joint speed.

```
--MAX JOINT SPEED SETTING --
  Suggested Speed = 800.000(deg/s)
  (Calculated with Max Motor Speed)

Enter (1:Change, 2:No Change)? 2
```

9 Select Motor Direction.
Choose [1:TRUE] if the joint coordinate position of the conveyor increases when the motor rotates in the plus direction.
Choose [2:FALSE] if the joint coordinate position of the conveyor decreases when motor rotates in the plus direction.

```
-- MOTOR DIRECTION
INDEPENDENT AXES 1 Motion Sign = TRUE
Enter (1:TRUE, 2:FALSE)?
```

If you look at a motor from the front of the flange, a counter clockwise rotation is plus direction of a motor.

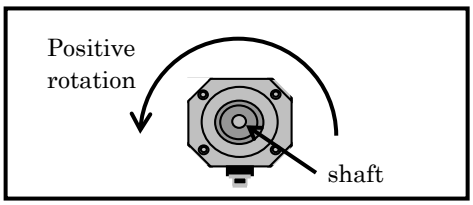


Fig. 9.7.2.1 (b) Positive rotation of a motor

10 Enter the limit of the axis. Please enter 180[deg] as an upper limit and –180[deg] as a lower limit.

```
-- UPPER LIMIT --
Enter Upper Limit ( deg)? 180
```

```
--LOWER LIMIIT--
Enter Lower Limit ( deg)? -180
```

11 Enter the mastering position of the axis. Please enter the position where it is possible to carry out mastering within the motion range. Normally, the position is “0”.

```
--MASTER POSITION –
Enter Master Position (deg)?
```

12 Enter the acceleration/deceleration time constants (ACC/DEC time).
First, enter the 1st ACC/DEC time (acc_time1). “Default value of acc_time1” is the default value.

```
-- ACC/DEC TIME--
Default Value of acc_time1 = 384(ms)
Enter (1:Change, 2:No Change)?
```

If you would like to change the value, choose [1:Change] and enter the new value. If you don’t change the value, choose [2:No Change].

```
Enter Accel Time 1 (ms)?
```

Next, enter the 2nd ACC/DEC time (acc_time2). Default value of acc_time2 is the default value. Please set half the value of acc_time1.

```
Default value of acc_time2 = 192 (ms)
Enter (1:Change, 2:No Change)?
```

If you would like to change the value, choose [1:Change] and enter the new value. If you don’t need to change the value, choose [2:No Change].

```
Enter Accel Time 2 (ms)?
```

13 Enter Minimum Accel Time. When doing motion, if the calculated acceleration/deceleration time is smaller than the specified time, the acceleration/deceleration time will be clamped to the specified time.

```
-- MIN_ACCEL TIME --
Default Value of min_acctime = 384(ms)
Enter (1:Change, 2:No Change)?
```

min_acctime should be the sum of acc_time1 and acc_time2. Please choose [1:Change] and enter it.

```
Enter Minimum Accel Time (ms)?
```

14 Enter Load Ratio. This value is the ratio of all load inertia to the rotor inertia and should be calculated by the specification of the servo conveyor in advance. The valid range of Load Ratio is from 1.0 to 5.0. If you don’t set this value, enter “0”.

```
-- LOAD RATIO --
Load Ratio = (Load Inertia + Motor Inertia) / Motor Inertia

Enter Load Ratio -> (0:None 1->5:Valid)
```

15 Enter Amplifier Number.

```
--SELECT AMP NUMBER--
Enter amplifier number (1-56)?
```

16 Select Amplifier Type

```
-- SELECT AMP TYPE --
 1.   A06B-6107 series 6 axes amplifier
 2.   A06B-6117 series Alpha i amp. or
      A06B-6130 series Beta i amp.

Select?
```

17 Enter Brake Number.

```
--BRAKE SETTING --
Enter Brake Number (0~16)?
```

18 Select the type of brake control (Servo Timeout). The brake control function put on a brake automatically when an axis does not move for a given length of time.

```
--SERVO TIMEOUT --
Servo off is Disable
Enter (1:Enable, 2:Disable)?
```

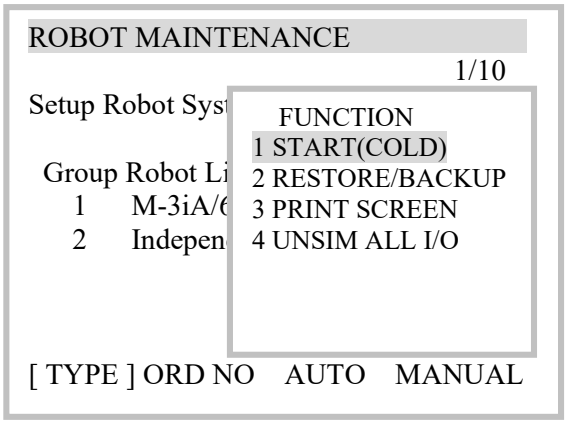
If you choose “1:Enable”, then enter the delay time of brake control (Servo Off Time). The valid range of Servo Off Time is from 0 to 30(sec).

```
-- SERVO TIMEOUT VALUE --
Enter Servo Off Time? (0.0~30.0)
```

19 After the above setting, “Independent Axes Setup Menu” appears. Please select “EXIT” and finish the axis setup.

```
*** Group 2 Total Axes Installed =    0
  1. Display/Modify Nobot Axis 1~4
  2. Add Axis
  3. Delete Axis
  4. EXIT
Select Item? 4
```

20 After axis setup is finished, press the FCTN key to display the function menu. Select “1 START (COLD)” and press the ENTER key and then Cold Start is executed.



9.7.2.2 Servo Conveyor Setup

Please set up Servo Conveyor by the following step. The setting of encoder and the setting of Continuous Turn are updated automatically by setting Servo Conveyor.

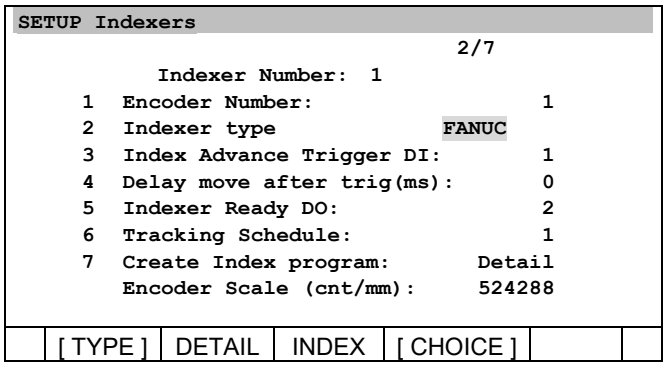
NOTE
It is possible to specify a flag to use on Servo Conveyor Setup display.

9

Procedure: Servo Conveyor Setup

Step

1 Press the [MENU] key.
2 Select [SETUP].
3 Press F1 [TYPE] key.
4 Select “Indexers”. You will see a screen as following.



5 Move the cursor to [1 Encoder Number] and enter the number of the encoder that is used as servo conveyor.
6 In [2 Indexer type], one of the following settings is configured according to the optional servo conveyor tracking function that has been ordered. This item cannot be changed.

Index: This conveyor repeatedly stops after a certain distance and then moves again.
This indexer type is selected when the optional index servo conveyor tracking function is ordered.
Continuous: This conveyor moves continuously at a constant speed.
This indexer type is selected when the optional continuous servo conveyor tracking function is ordered.

7 Move the cursor to [6 Tracking Schedule] and enter the value of the tracking schedule that is used for servo conveyor.

8 Move the cursor to [2 Indexer type] and press F2 [DETAIL] key or the [Enter] key. You will see a screen for setting up axis. The screen for setting up axis is different to indexer type.

In the case of that Indexer type is “Indexer”

9 You will see a screen as following. Because there is no [TYPE] in this menu, you can not select any other setup menu. To return back to previous menu with [TYPE], press the [PREV] key.

```
SETUP Indexer axis:
                                    1/12
         Indexer  1: FANUC motor DONE
        Encoder Number:    1
    1   Robot Group:                    2
             Axis:                      1
    2   Motor Gear teeth                1
    3   Rotor input Gear teeth          1
    4   Rotor output Gear teeth         1
    5   Conveyor belt teeth             1
    6   Number of Flight              100
    7   Index Distance (mm)         10.000
    8   Index Advance Trigger DI:       1
    9   Delay move after trig(ms):     16
   10   Indexer Ready DO:               1
   11   Flag for internal use:        131
   12   Flag 2 for internal use:      132
 |      |      |      | EXEC |      |      |
```

10 Move the cursor to [1 Robot Group] and enter the group number of the extended axis for the servo conveyor and then move the cursor to “Axis” and enter the axis number.

11 Move the cursor to [2 Motor Gear teeth], [3 Rotor input Gear teeth], [4 Rotor output Gear teeth] and [5 Conveyor belt teeth] and enter each value in the positive integer. Move the cursor to [6 Number of Flight] and enter the number of bucket on the servo conveyor. The following figure shows the relationship between these values. The gear ratio of the servo conveyor is calculated from these value.

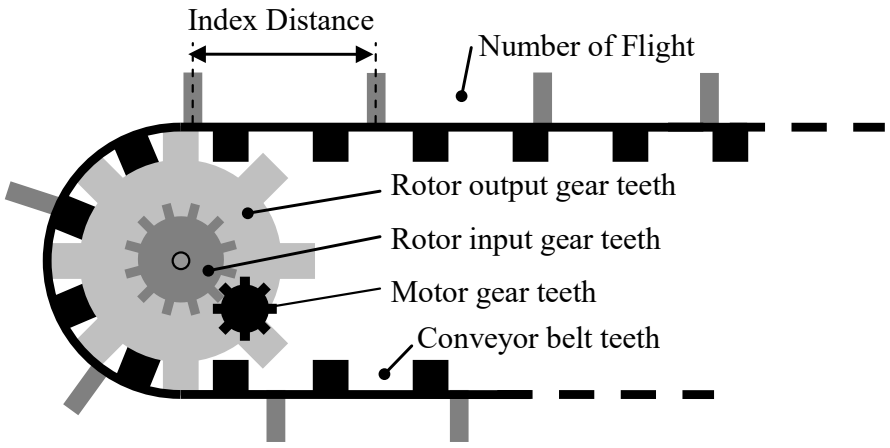


Fig. 9.7.2.2 (a) Relationship between motor, rotor and conveyor

- It is also possible to set up from the relation between the number of revolution of motor and the number of revolution of the conveyor. If a conveyor rotates “N” times and a motor of it rotates “M” times, please setup as following. “M” and “N” must be an integer.
  a. Enter the value of “N” to [2 Motor Gear teeth]
  b. Enter the value of “M” to [3 Rotor input Gear teeth]
  c. Enter “1” to [4 Rotor output Gear teeth] and [5 Conveyor belt teeth]

12 Move the cursor to [7 Index Distance (mm)] and the distance a one pitch on the servo conveyor.

13 Move the cursor to [8 Index Advance Trigger DI] and enter the index of DI is used for moving the servo conveyor.

14 Move the cursor to [9 Delay move after trig (ms)] and enter the value of delay time until the servo conveyor starts after trigger.

15 Move the cursor to [10 Indexer Ready DO]. If you would like to output DO at start of TP program for the servo conveyor, enter the index of DO. Please refer to TP program for Servo Conveyor (INDXG*.TP).

16 Move the cursor to [11 Flag for internal use] and enter the index of flag is used for the servo conveyor. Please confirm that the specified Flag is not used for another use.

17 Move the cursor to [12 Flag2 for internal use] and enter the index of flag is used for the servo conveyor. Please confirm that the specified Flag is not used for another use.

18 After the above setup, press F2 [EXEC] key and be sure to repower.

NOTE
By the above setting, Continuous Rotation setup of the conveyor and Encoder setup of the specified encoder are also done.

In the case of that Indexer type is “Continuous”

9 You will see a screen as following. Because there is no [TYPE] in this menu, you can not select any other setup menu. To return back to previous menu with [TYPE], press the [PREV] key.

```
SETUP Conveyor axis
                                1/8
          Conveyor  1: motor    DONE
         Encoder Number:   1
     1   Robot Group:                2
           Axis:                     1
     2   Motor Gear teeth            1
     3   Rotor input Gear teeth      1
     4   Rotor output Gear teeth     1
     5   Belt Section teeth          1
     6   Belt Section length (mm)  10.000
     7   Conveyor ON DI:             1
     8   Flag for internal use:    133

          |      |      | EXEC |      |      |
```

10 Move the cursor to [1 Robot Group] and enter the group number of the extended axis for the servo conveyor and then move the cursor to “Axis” and enter the axis number.

11 Move the cursor to [2 Motor Gear teeth], [3 Rotor input Gear teeth], [4 Rotor output Gear teeth] and [5 Belt Section teeth] and enter each value in the positive integer. Move the cursor to [6 Belt Section length (mm)] and enter the belt length corresponding to [5 Belt Section teeth].
The following figure shows the relationship between these values. The gear ratio of the servo conveyor is calculated from these values.

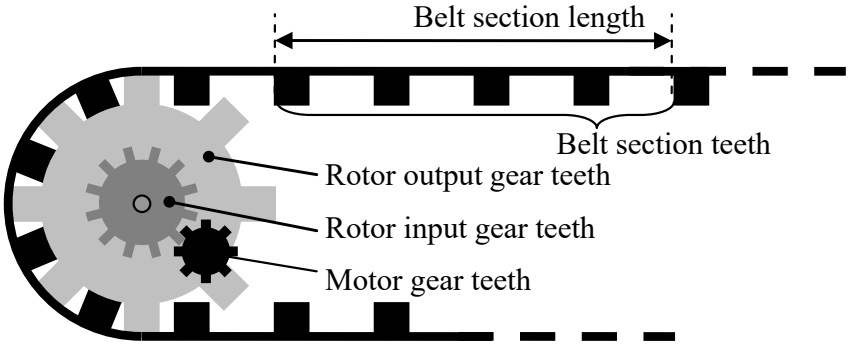


Fig. 9.7.2.2 (b) Relationship between motor, rotor and conveyor

- It is also possible to set up from the relation between the number of revolution of motor and the number of revolution of the conveyor. If a conveyor rotates “N” times and a motor of it rotates “M” times, please setup as following. “M” and “N” must be an integer.
  a. Enter the value of “N” to [2 Motor Gear teeth].
  b. Enter the value of “M” to [3 Rotor input Gear teeth].
  c. Enter the number of teeth of the conveyor belt to [4 Rotor output Gear teeth].
  d. Enter the number of teeth corresponding to [6 Belt Section length] and [5 Belt Section teeth].

12 Move the cursor to [7 Conveyor ON DI] and enter the index of DI is used for moving the servo conveyor.

13 Move the cursor to [8 Flag for internal use] and enter the index of flag is used for the servo conveyor. Please confirm that the specified flag is not used for another use.
14 After the above setup, press F2 [EXEC] key and be sure to repower.

NOTE
By the above setting, Continuous Rotation setup of the conveyor and Encoder setup of the specified encoder are also done.

9.7.2.3 How to Create TP Program for Servo Conveyor

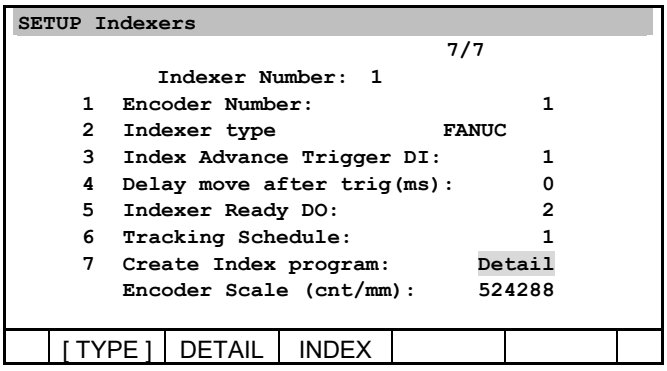
It is necessary to prepare TP program for moving the servo conveyor on tracking because the servo conveyor is setup as extended axis. By following step, the standard TP program for moving the servo conveyor.

Procedure: How to create TP program for Servo Conveyor

Step

NOTE
Before generating indexer program, please set max payload which is used in an indexer tracking motion on MOTION/PAYLOAD SET menu.

1 Move the cursor to [7 Create Index program] in "SETUP Indexers" and press F2 [DETAIL] key or the [ENTER] key.



You will see a screen as following. Because there is no [TYPE] in this menu, you can not select any other setup menu. To return back to previous menu with [TYPE], press the [PREV] key.

2 Move the cursor to [1 Index Speed (part/min)] and enter the value of "Index Speed" of the servo conveyor. Please set the number of pitches per minute.

3 Move the cursor to [2 Index Dwell (ms)] and enter the value of the time to stop the servo conveyor. The speed pattern of the servo conveyor is different according to the value. Please refer to following figures.
If you would like to use a conveyor as Index Conveyor, it is necessary to "0" and over as "Index Dwell". If you would like to move a conveyor at constant speed, it is necessary to set "-1" as "Index Dwell".
Movement Time of Conveyor per Pitch is calculated from "Index Speed".

- Index Dwell is "0" and over:
  Movement Time of Conveyor per Pitch [ms] = (60*1000/Index Speed) – Index Dwell

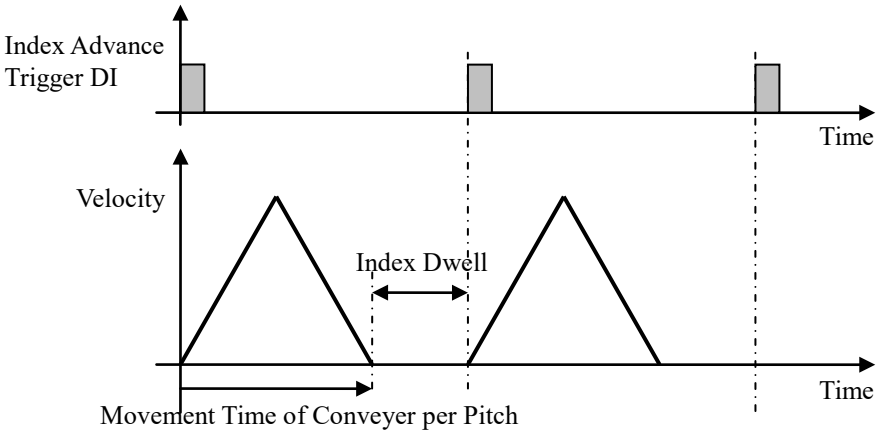


Fig. 9.7.2.3 (a) Speed pattern if Index Dwell is "0" and over

- Index Dwell is "-1":
  Movement Time of Conveyor per Pitch [ms] = (2*60*1000/Index Speed)

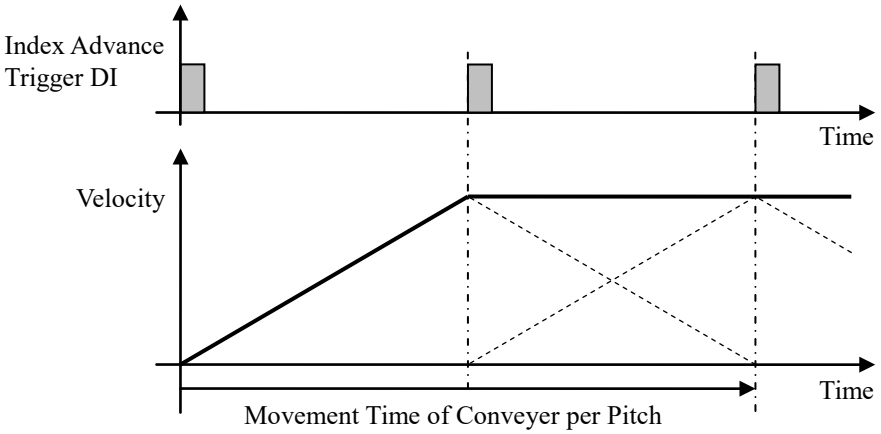


Fig. 9.7.2.3 (b) Speed pattern if Index dwell is "-1"

4 Move the cursor to [3 Indexer Register start] and enter the start index of Register is used in TP program for the servo conveyor. Two registers are used in TP program.
For example, R[60] and R[61] are used when [3 Indexer Register start] is "60". R[60] is used for setting the number of DO. The specified DO by R[60] is used for checking whether the conveyor finishes moving to initial position. R[61] is used for counting the number of pitches of the conveyor. Please refer to example of INDXG*.TP.

5 After the above setting move the cursor to [4 Generate Index program] and press F2 [CREATE] key. If there is no problem in the setting, "Done" is displayed and INDXG*.TP (* is group number of the servo conveyor) is created.

NOTE

- Do not change INDXG*.TP directly because it is default program in the system. Please rename INDXG*.TP and use it.
- "Part Rate exceeds allowable value" might be displayed by the setting. This message shows that the calculated conveyor acceleration from the setting for TP exceeds allowable acceleration of the servo conveyor or robot. In this case, INDX*.TP is not created. Please adjust the setting (Index Speed, Index Dwell) so that the acceleration decreases.

- If there is INDXG*.TP and you press F2 [CREATE] key, “Index Program already exist” is displayed. In this case, INDXG*.TP is not created. If you would like to create TP program, please press the [SHIFT] key and F3 [REPLACE] key. If there is no problem in the setting, existing INDXG*.TP is overwritten by the created TP program from current setting.
  - If you press only F3 [REPLACE] key, “Hold Shift & Replace to replace program” is displayed. In this case, INDXG*.TP is not overwritten.
  - If INDXG*.TP is currently open in editor, the system will not be able to update the program. In this case the error message will displayed and “done” will not be displayed.
  - “Part Rate exceed allowable value” might be displayed by the setting when you press the [SHIFT] key and F3 [REPLACE] key. Please adjust the setting so that the acceleration decreases.

Sample Program

The following is an example of INDXG*.TP.

Example of INDXG*.TP

```
1:   J P[1] 50% CNT0      ;                          Movement to P[1] as an initial position.
2:     IF R[60:G2 Ready DO]=0,JMP LBL[3] ;
3:     DO[R[60]]=ON ;                                Specified DO by R[60] become on
4:     LBL[3] ;
5:     R[61:G2 cur slot ID]=0      ;                 Reset R[61]
6:     LBL[1] ;
7:     $INDEXER[1].$INDEX_MV=1 ;                     Setting for waiting for DI trigger
8:   J P[2] 180msec CNT100 INC ACC66      ;         One pitch movement
9:     $INDEXER[1].$INDEX_MV=0 ;                     Setting for waiting for DI trigger
10:    R[61:G2 cur slot ID]=R[61:G2 cur slot ID]+1;  Count the number of pitches
11:    IF R[61:G2 cur slot ID]=32,JMP LBL[2] ;       If conveyor is turned once, Jump to LBL[2].
12:    JMP LBL[1] ;
13:    LBL[2] ;
14:  J P[1] 100% CNT100      ;                       Movement to P[1] to correct iteration error
15:    R[61:G2 cur slot ID]=0      ;                 Reset R[61]
16:    JMP LBL[1] ;
```

```
/POS
P[1:""]{
   GP2:
            UF : 0, UT : 1,
            J1=        0.000 deg
};
P[2:""]{
   GP2:
            UF : 0, UT : 1,
            J1=     360.000 deg
};
```

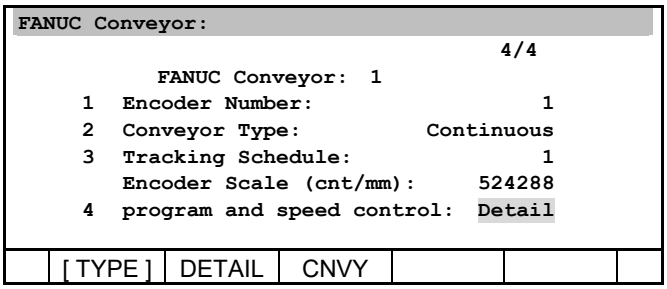
- At line 1, the conveyor moves to the initial position P[1].
- DO is output to check whether the conveyor finishes moving to the initial position by instructions from line 2 to line 4.
- At line 8, the conveyor moves 1 pitch. In other words, the extended axis that is used as the conveyor moves 360 degrees.
- ”$INDEXER[*].$INDEX_MV=1” at line 7 and “$INDEXER[*].$INDEX_MV=0” at line 9 are setting for watching input of Index Advance Trigger DI (* is number of Servo Conveyor). Motion instruction at line 8 is not executed until the DI changes from OFF to ON.
- IF instruction at line11 checks whether the conveyor makes one revolution. In this example case, the conveyor makes one revolution per 32 pitches.
- At line 14, the conveyor moves to P[1] to correct iteration error. Every time the conveyor makes one revolution this line is executed.

- R[60] and R[61] are set by specifying “Indexer Register start”. R[60] is used for setting the number of DO. The specified DO by R[60] is used for checking whether the conveyor finishes moving to initial position. R[61] is used for counting the number of pitches of the conveyor.

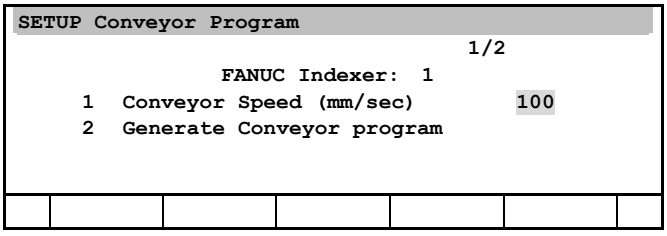
Procedure: How to create TP program for “Continuous” Conveyor

Step

1 Move the cursor to [4 Program and speed control] in “SETUP Indexers” and press F2 [DETAIL] key or the [ENTER] key.



You will see a screen as following. Because there is no [TYPE] in this menu, you can not select any other setup menu. To return back to previous menu with [TYPE], press the [PREV] key.



2 Move the cursor to [1 Conveyor Speed (mm/sec)] and enter a value of the servo conveyor speed in the positive integer.

NOTE
The value of [1 Conveyor Speed (mm/sec)] is used for moving a servo conveyor with TP program. If you would like to change the conveyor speed, please change this value instead of editing TP program.

3 After the above setting move the cursor to “Generate Conveyor program” and press F2 [CREATE] key. If there is no problem in the setting, “Done” is displayed and CNVYG*.TP (* is group number of the servo conveyor) is created.

NOTE

- Do not change CNVYG*.TP directly because it is default program in the system. Please rename CNVYG*.TP and use it.
- If there is CNVYG*.TP, F3 [REPLACE] key is displayed instead of F2 [CREATE] key. If you would like to create TP program, please press the [SHIFT] key and F3 [REPLACE] key. If there is no problem in the setting, existing CNVYG*.TP is overwritten by the created TP program from current setting.
  - If you press only F3 [REPLACE] key, “Hold Shift & Replace to replace program” is displayed. In this case, CNVYG*.TP is not overwritten.
  - If INDXG*.TP is currently open in editor, the system will not be able to update the program.
- If you would like to change the servo conveyor speed, please change the value of [1 Conveyor Speed (mm/sec)]. It is not necessary to edit TP program for changing a conveyor speed. The specified value is used for moving a conveyor. The conveyor speed is updated even if TP program is running.

If the specified value is bigger than the conveyor capability, “Speed exceed conveyor capability” is displayed and the value is not updated.

Sample Program

The following is an example of CNVYG*.TP.

Example of CNVYG*.TP

```
1:    LBL[1] ;
2:    $INDEXER[1].$INDEX_MV=1 ;              Setting for waiting for DI trigger
3:  J P[2] 48msec CNT100 INC ACC100    ;     One pitch movement
4:    $INDEXER[1].$INDEX_MV=0 ;              Setting for waiting for DI trigger
5:    JMP LBL[1] ;
```

- ”$INDEXER[*].$INDEX_MV=1” at line 2 and “$INDEXER[*].$INDEX_MV=0” at line 4 are setting for watching input of Index Advance Trigger DI (* is number of Servo Conveyor). Motion instruction at line 3 is not executed until the DI changes from OFF to ON.

9.7.2.4 Tracking Schedule Setup

Tracking Schedule Setup for servo conveyor is the same as ordinary Line Tracking function. Please set items of the specified tracking schedule by the servo conveyor setup display. Please refer to Subsection 3.2.8, “Setting Up a Conveyor Station” or Subsection 4.2.9, “Setting Up a Conveyor Station”.

9.7.2.5 Example of Main Program

Basically, how to teach TP program is the same as ordinary Line tracking function. However, it is necessary to run TP program for moving a servo conveyor by multitask.
CNVEYOR.TP for moving a servo conveyor is run at line 3 in the following Sample Program. It is possible to create TP program for moving a servo conveyor (INDXG*.TP, CNVYG*.TP) by the procedure of Subsection 9.7.2.3, “How to Create TP Program for Servo Conveyor”.

Example of Main Program

```
1:   J P[1] 100% FINE ;
2:     LINE[1] ON ;
3:     RUN CONVEYOR;          CONVEYOR.TP for moving Servo Conveyor is executed.
4:     LBL[1] ;
5:     WAIT DI[2]=ON     ;
6:     LINECOUNT[1] R[1] ;
7:     SETTRIG LNSCH[1] R[1] ;
8:     CALL TRACK     ;       TRACK.TP for tracking motion is executed.
9:     CALL NORM      ;       NORM.TP for non-tracking motion is executed.
10:    JMP LBL[1] ;
```

- Motion instruction at line 1 moves robot to the initial position.
- At line 3, “RUN” CONVEYOR.TP for moving the conveyor.
- At line 5, wait until a workpiece on the conveyor is detected. DI[2] is used for detecting a workpiece.
- At line 6, 7, trigger value is got and set before tracking motion.
- At line 8, “CALL” TRACK.TP for tracking motion.
- At line 9, “CALL” NORM.TP for non-tracking motion.

9.7.3 FUNCTION FOR SERVO CONVEYOR LINE TRACKING

9.7.3.1 Dynamic Error Tune Variable

On Servo Conveyor Line Tracking, please adjust a dynamic error of the tracking motion by using the following parameter instead of [Dynamic Error Adjustment], which is described in Section 6.2, "SETUP OF A ROBOT".
Refer to the description of FINE TUNING HIGH SPEED ACCURACY in "Line Tracking OPERATOR'S MANUAL B-83474EN" . It is possible to adjust a dynamic error on every servo conveyor.

$SLTK_GRP[number of servo conveyor group].$SRVO_DELAY

9.7.3.2 Wait Indexer Stop Function

This function is disabled by default. When this function is disabled, a robot starts/ends a tracking regardless of the motion of the servo conveyor. Therefore, for example, if robot starts a tracking motion while a servo conveyor accelerates, the acceleration of the robot is increased in order to catch up the conveyor and the motion of the robot might become aggressive.
When this function is enabled, Robot starts or ends a tracking motion while a servo conveyor stops. By this setting, the acceleration at the start or end of tracking motion is restricted.

NOTE
If you enable this function and there is no time the conveyor stops, it is not possible to start or end a tracking motion.

The flag for switching the setting of this function is bit 1 and bit 2 of $INDX_TRACK[schedule number]. If you would like to enable this function, please set these flag by the following procedure.

- If bit 1 of $INDX_TRACK[schedule number], Robot starts a tracking motion while a servo conveyor stops.
- If bit 2 of $INDX_TRACK[schedule number], Robot ends a tracking motion while a servo conveyor stops.

NOTE
If the setting of this function is changed, please create TP program for the servo conveyor again.

Procedure: Wait Indexer Stop Function Setup

Step

1 Setting bit 1 of $INDX_TRACK[schedule number]
1.1 Please divide the value of $INDX_TRACK[schedule number] by "2" and then check current setting by following step.
- If the integer part of the calculated value is Odd number, bit 1 is TRUE.
- If the integer part of the calculated value is Even number, bit 1 is FALSE.

1.2 Please change the value of $INDX_TRACK[schedule number] according to the current setting.
- If bit 1 is TRUE and you would like to change it to FALSE, please subtract "2" from the value of $INDX_TRACK[schedule number].
- If bit 1 is FALSE and you would like to change it to TRUE, please add "2" to the value of $INDX_TRACK[schedule number].

2 Setting bit 2 of $INDX_TRACK[schedule number]
2.1 Please divide the value of $INDX_TRACK[schedule number] by "4" and then check current setting by following step.
- If the integer part of the calculated value is Odd number, bit 2 is TRUE.

- If the integer part of the calculated value is Even number, bit 2 is FALSE.

2.2 Please change the value of $INDX_TRACK[schedule number] according to the current setting.
- If bit 2 is TRUE and you would like to change it to FALSE, please subtract "4" from the value of $INDX_TRACK[schedule number].
- If bit 2 is FALSE and you would like to change it to TRUE, please add "4" to the value of $INDX_TRACK[schedule number].

3 After the above setting, please Repower.

9.7.3.3 KAREL Program for Servo Conveyor Line Tracking

The following KAREL programs can be used. By using these, it is possible to get a trigger value and save it every time Servo Conveyor moves a fixed distance. It is also possible to use Discard Line function and Stop Conveyor function and Distribution Line function.

NOTE
You can also use *i*RPickTool KAREL programs.

- SLTKINIT
- SLTKREST
- SLTKPSHQ
- SLTKPOPQ
- SLTKDELQ
- SLTKRSTQ
- SLTKGTPP

This subsection explains these KAREL programs and Discard Line, Stop Conveyor and Distribution Line.

NOTE
The functions related to KAREL programs for servo conveyor line tracking and the *i*RPickTool functions cannot be used in combination. To use an *i*RPickTool function, use the KAREL programs of iRPickTool, not the KAREL program for servo conveyor tracking.

SLTKINIT

This program clears information about trigger values. Normally, this program is called once when the system is started.

Argument1: Specify the number of tracking schedule which Servo Conveyor Line Tracking uses.
Argument2: Specify the Position Register number that is saved an initial position of the conveyor.
Argument3: Specify the Position Register number.

The specified Position Register by argument 3 is set to the distance between the current conveyor position and an initial position of the conveyor. The value within 1 pitch (from 0 to 360 degrees) is set. This is available for adjusting the conveyor position at the start.

SLTKREST

Because the distance of one pitch of Servo Conveyor is constant, an encoder value can be got as a trigger value every a constant distance. This program is used for setting a reference value of the trigger. When SLTKREST is called, the encoder value is saved as a reference value of the trigger. It is necessary to call this program once just after the servo conveyor moves to an initial position.

Argument1: Specify the number of tracking schedule which Servo Conveyor Line Tracking uses.

In the following case, PR[2] is set to the distance between the current conveyor position and an initial position of the conveyor by SLTKINIT at line 1. Next, the conveyor moves to an initial position by the motion instruction and Reference position for trigger is saved by SLTKREST.

```
1:    CALL SLTKINIT(1,1,2)     ;    Initialization and Setting of Pos. Register.
2:  J PR[2] 50% CNT0 INC       ;    Movement to an initial position.
3:    CALL SLTKREST(1) ;            Setting of Reference Pos. for Trigger.
```

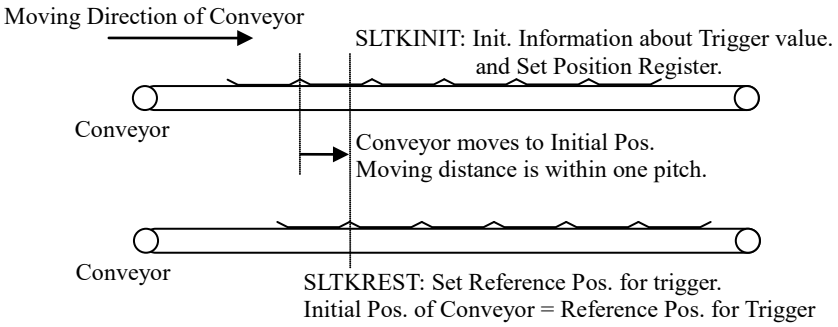


Fig. 9.7.3.3 (a) SLTKINIT, SLTKREST

SLTKPSHQ

This is used for saving trigger value after the conveyor moves the number of pitch.

Argument1: Specify a number of tracking schedule which Servo Conveyor Line Tracking uses.
Argument2: Specify a number of pitch

If you would like to change the save-able number of trigger values, please change the value of the following system variable and repower.

$SLTKSCH[n].$QUE_SIZE: Default value is 20 and max value is 100.
"n" is the number of tracking schedule which Servo Conveyor Line Tracking uses.

NOTE
When the saved number of trigger values become the specified value by $SLTKSCH[n].$QUE_SIZE, the oldest saved trigger value is removed.

It is necessary call this program after the conveyor moves the specified number of pitch. When this program is called, a trigger value is calculated from the reference trigger value and the specified number of pitch and it is saved. At the first, the reference trigger value is saved as a trigger value.
For example, if the conveyor moves 1 pitch and the trigger value is saved once, Argument2 should be set "1". If the conveyor moves 2 pitches and the trigger value is saved once, Argument2 should be set "2". The following shows the case that trigger values are saved during the conveyor moves four pitches.

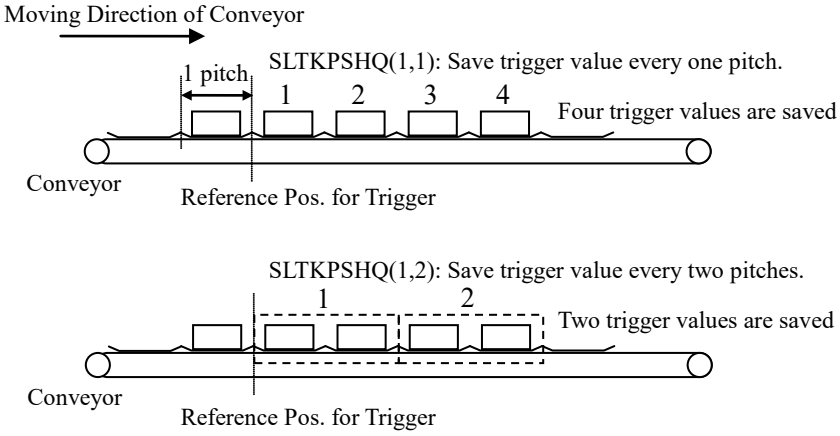


Fig. 9.7.3.3 (b) SLTKPSHQ

SLTKPOPQ

This program is used for getting the saved trigger value. When this program is called, it is possible to get the oldest saved trigger value. If you succeed in getting the trigger value, the trigger value is removed and it is impossible to get it again.

Argument1: Specify a number of tracking schedule which Servo Conveyor Line Tracking uses.
Argument2: Specify a register number for getting a trigger value.
Argument3: Specify a register number for getting a status. If SLTKPOPQ fails to get a trigger value the value of status become nonzero.

The following shows the case that there are four saved trigger values.

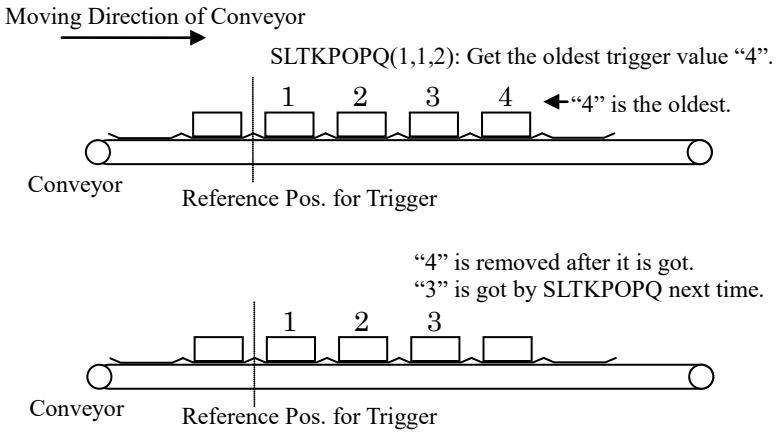


Fig. 9.7.3.3 (c) SLTKPSHQ

SLTKDELQ

This program is used for deleting the specified the saved trigger value. It is not possible to get the deleted trigger value by SLTKPOPQ.

Argument1: Specify a number of tracking schedule which Servo Conveyor Line Tracking uses.
Argument2: Specify a number of trigger to delete.

For example, if argument2 is “2”, SLTKDELQ deletes the second trigger value “2” from the newest trigger value.

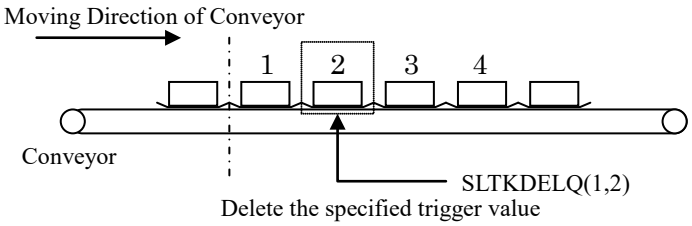


Fig. 9.7.3.3 (d) SLTKDELQ

SLTKRSTQ

This program turns off the specified DO for Stop Conveyor. Please refer to STOP CONVEYOR below.

Argument1: Specify a number of tracking schedule which Servo Conveyor Line Tracking uses.

SLTKGTPP

This program is used for calculating a distance along X axis of tracking frame from the origin of tracking frame to the corresponding position to the oldest available trigger value.

Argument1: Specify a number of tracking schedule which Servo Conveyor Line Tracking uses.
Argument2: Specify a register number for getting a distance from the origin of tracking frame.
Argument3: Specify a register number for getting a status. If SLTKGTPP succeeds, the value of the status becomes "0". If SLTKGTPP fails to calculate a distance, the value of the status becomes nonzero and the value of the register specified by argument 2 is not updated.

The following shows the case that there are four saved trigger values.
For example, if the oldest available trigger value is "4", the corresponding distance to the trigger value "4" is calculated.

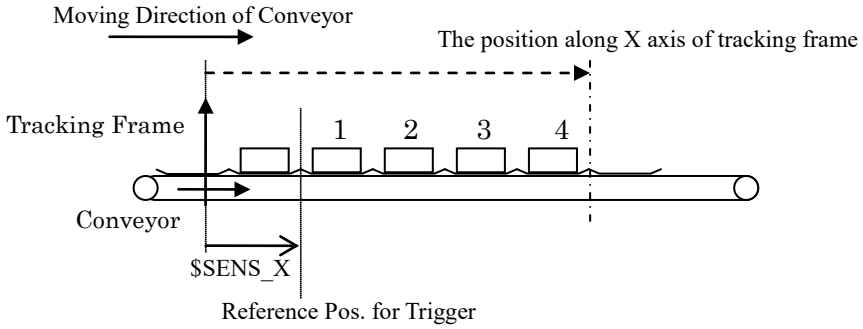


Fig. 9.7.3.3 (e) SLTKGTPP

If "4" is deleted by SLTKDELQ, "3" becomes the oldest available trigger value. Therefore, the corresponding distance to the trigger value "3" is calculated by SLTKGTPP.

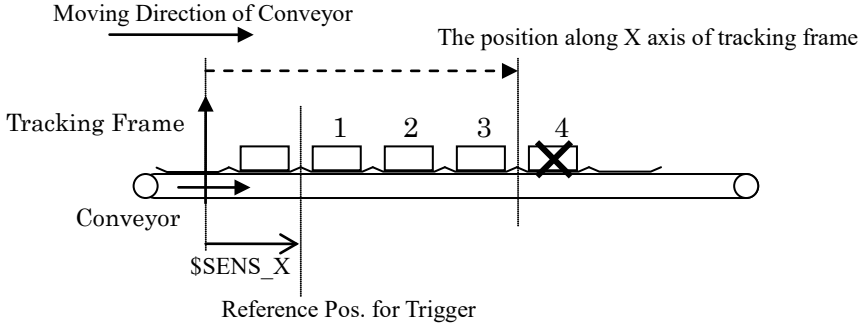


Fig. 9.7.3.3 (f) SLTKGTPP and SLTKDELQ

If Discard Line is set and “4” passes the discard line, it is not possible to get the trigger value “4”. In this case, the corresponding distance to the trigger value “3” is calculated by SLTKGTPP.

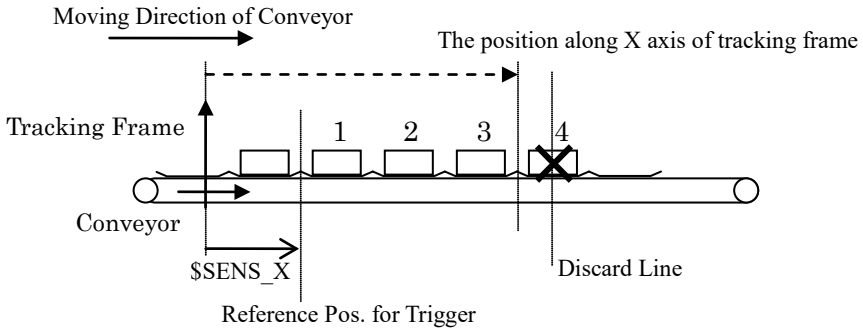


Fig. 9.7.3.3 (g) SLTKGTPP and Discard Line

If Stop Conveyor is set and “4” passes the discard line, DO is out put in order to call upon to stop the conveyor. Even if “4” passes the discard line in Stop Conveyor case, it is possible to get the trigger value “4”. The corresponding distance to the trigger value “4” is calculated by SLTKGTPP.

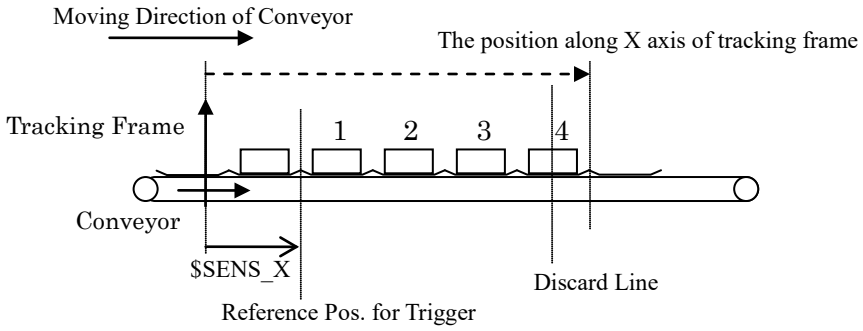


Fig. 9.7.3.3 (h) SLTKDELQ and Stop Conveyor

DISCARD LINE

If a corresponding position to the trigger value passes the discard line, it is possible to discard the trigger value. It is necessary to set Discard Line on the basis of downstream boundary of the tracking area.

It is necessary to set following system variables in order to use this function.

$SLTKSCH[n].$SLQ_ENABLE: TRUE

$SLTKSCH[n].$DISCARD_BND: Specify the distance from the downstream boundary of the tracking area to the Discard Line. The unit is [mm]

$SLTKSCH[n].$SENS_X: Specify the distance from the origin of Tracking frame to Reference Pos. along X axis of Tracking frame. The unit is [mm]

“n” is the number of tracking schedule which Servo Conveyor Line Tracking uses.

If $DISCARD_BND is negative, Discard Line is set up Down-stream boundary. Normally, you should set $DISCARD_BND to the negative value. If a corresponding position to the trigger value passes the discard line, it is not possible to get the trigger value by SLTKPOPQ. The oldest saved trigger value is used for checking whether this function discard the trigger value.

In the following case, because the corresponding position to the trigger value “4” passes the discard line, “4” is discarded. After that, the corresponding position to the trigger value “3” is checked whether this function discard the trigger value.

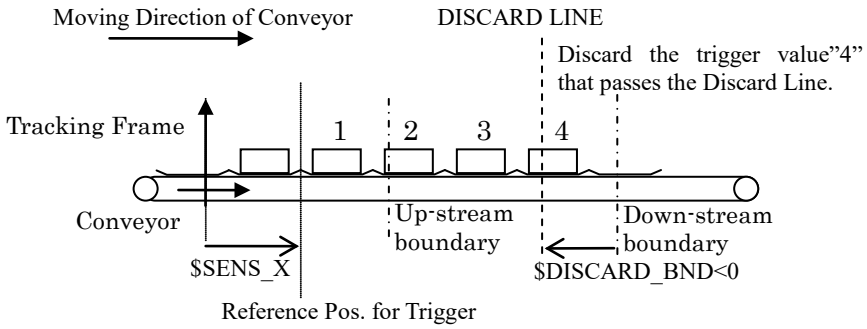


Fig. 9.7.3.3 (i) Discard line

STOP CONVEYOR

If a corresponding position to the trigger value passes Discard Line, the specified DO become ON. While the Specified DO is ON, you should not set [3 Index Advance Trigger DI] to ON in order to stop the conveyor.
It is necessary to set Discard Line so that the desired tracking motion can be done at the conveyor stop position.
It is necessary to set following system variables in order to use this function.

$SLTKSCH[n].$SLQ_ENABLE:	TRUE
$SLTKSCH[n].$DISCARD_BND:	Specify the distance from the downstream boundary of the tracking area to the Discard Line. The unit is [mm]
$SLTKSCH[n].$SENS_X:	Specify the distance from the origin of Tracking frame to Reference Pos. along X axis of Tracking frame. The unit is [mm]
$SLTKSCH[n].$STOPBELT:	TRUE
$SLTKSCH[n].$STOP_DONUM:	Specify DO number for Stop Conveyor

"n" is the number of tracking schedule which Servo Conveyor Line Tracking uses.

When Stop Conveyor function is enabled and the corresponding position to the trigger value passes Discard Line, the trigger value is kept. While the Specified DO is ON, it is possible to get the trigger value by SLTKPOPQ. If you would like to change the specified DO from ON to OFF, it is necessary to use SLTKRSTQ. The next trigger value is used for checking the Stop Conveyor after SLTKRSTQ is executed. In the following case, because the corresponding position to the trigger value "4" passes the discard line, the specified DO becomes ON. According the DO, it is necessary to stop the conveyor. The trigger value "4" can be got by SLTKPOPQ. After a tracking motion is done at the corresponding position to the trigger value "4", the DO becomes OFF by executing SLTKRSTQ. According the DO, it is necessary to resume the conveyor. When SLTKRSTQ is executed, the corresponding position to the trigger value "3" will be checked whether this function discard the trigger value.

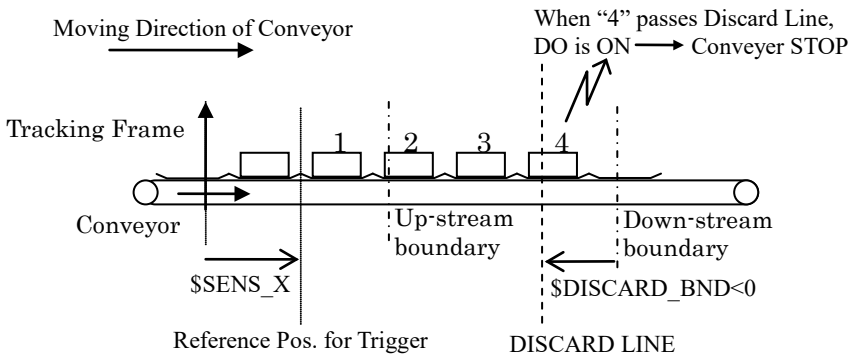


Fig. 9.7.3.3 (j) Stop conveyor

NOTE
It is impossible to use both Discard Line and Stop Conveyor on the same Servo Conveyor.

DISTRIBUTION LINE

Until a corresponding position to a trigger value passes the distribution line, it is not possible to get the trigger value with SLTKPOPQ. It is necessary to set the distribution line on the basis of upstream boundary of the tracking area.

It is necessary to set following system variables in order to use this function.

$SLTKSCH[n].$ALLOC_ENB: TRUE
$SLTKSCH[n].$ALLOC_BND: Specify the distance from the upstream boundary of the tracking area to the Distribution Line. The unit is [mm]
$SLTKSCH[n].$SENS_X: Specify the distance from the origin of Tracking frame to Reference Pos. along X axis of Tracking frame. The unit is [mm]
"n" is the number of tracking schedule which Servo Conveyor Line Tracking uses.

If $ALLOC_BND is negative, the distribution line is set up upstream boundary. Usually, you should set $ALLOC_BND to the negative value. Until a corresponding position to a trigger value passes the distribution line, it is not possible to get the trigger value with SLTKPOPQ.
In the following case, because the corresponding position to the trigger value "4" passes the distribution line, you can get the trigger value "4" with SLTKPOPQ.

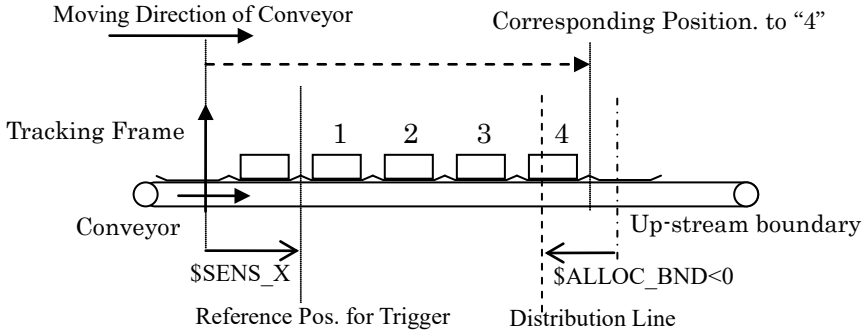


Fig. 9.7.3.3 (k) Distribution Line

Sample Program

The following are sample programs (Main Program and CONVEYOR.TP for moving Servo Conveyor) in case of using Stop Conveyor.

- The used Tracking Schedule number is "1".
- The initial position of the conveyor is saved to PR[1].
- The trigger value is saved every time the conveyor moves 2 pitches

In Main Program, the information about the trigger value is initialized by SLTKINIT at first. After that, TP program (CONVEYOR.TP) for moving Servo Conveyor is executed by the other task.

Example of Main Program

```
1:    J P[1] 100% FINE      ;
2:      LINE[1] ON ;
3:      CALL SLTKINIT(1,1,2) ;      Information about trigger value is initialized and PR[2] is set.
4:      RUN CONVEYOR;               CONVEYOR.TP for moving Servo Conveyor is executed.
5:      LBL[1] ;
6:      WAIT      .10(sec) ;
7:      CALL SLTKPOPQ(1,1,2) ;      The trigger value is got.
8:      IF R[2]<>0,JMP LBL[1] ;     If it is impossible to get the trigger value, Jump to LBL[1]
9:      SETTRIG LNSCH[1] R[1] ;     R[1] is set as a trigger value.
10:   CALL TRACK      ;            TRACK.TP for tracking motion is executed.
11:   CALL SLTKRSTQ(1) ;           Turn off DO for Stop Conveyor.
12:   CALL NORM      ;             NORM.TP for non-tracking motion is executed.
13:   JMP LBL[1] ;
```

- At line 3, the information about the trigger value is initialized by SLTKINIT and PR[2] is set to the distance from current position to the position of PR[1].
- At line 4, RUN CONVEYOR.TP in order to execute the program for moving Servo Conveyor by the other task.
- At line from 5 to 8, a trigger value is got by SLTKPOPQ every 100[ms]. At line 8, whether getting a trigger value succeed is checked by R[2]. If it is impossible to get a trigger value, jump to LBL[1] at line 5.
- At line 11, if DO for Stop Conveyor is ON, turn off the DO by SLTKRSTQ.

In TP program for moving Servo Conveyor, a trigger value is got every 2 pitches by SLTKPSHQ. R[3] is used for counting the number of pitches and SLTKPSHQ is executed according to the value of R[3].

Example of CONVEYOR.TP

```
1:    J PR[2] 50% CNT0 INC      ;   Movement to the initial position.
2:      CALL SLTKREST(1) ;          Setting of Reference Pos. for Trigger.
3:      IF R[60:G2 Ready DO]=0,JMP LBL[3] ;
4:      DO[R[60]]=ON ;              Specified DO by R[60] become ON.
5:      LBL[3] ;
6:      R[61:G2 cur slot ID]=0      ;   Reset R[61].
7:      R[3]=0      ;               R[3] is the counter for SLTKPSHQ.
8:      LBL[1] ;
9:      $INDEXER[1].$INDEX_MV=1 ;   Setting for waiting for DI trigger.
10:   J P[2] 180msec CNT100 INC ACC66      ;   One pitch movement.
11:     $INDEXER[1].$INDEX_MV=0 ;   Setting for waiting for DI trigger.
12:     R[61:G2 cur slot ID]=R[61:G2 cur slot ID]+1;   Count the number of pitches.
13:     R[3]=R[3]+1      ;          Count the number of pitches.
14:     IF R[61:G2 cur slot ID]=32,JMP LBL[2] ;
15:     JMP LBL[4] ;                Jump to LBL[4] at line 19.
16:     LBL[2] ;
17:   J PR[1] 100% CNT100      ;    Movement to PR[1] to correct iteration error.
18:     R[61:G2 cur slot ID]=0      ;   Reset R[61]
19:     LBL[4] ;
```

```
20:     IF R[3]<2,JMP LBL[1] ;          Check whether conveyor moves 2 pitches.
21:     CALL SLTKPSHQ(1,2) ;            When conveyor moves 2 pitches, the trigger value
                                         is saved by SLTKPSHQ.
22:           R[3]=0      ;             Reset R[3].
23:           JMP LBL[1] ;
/POS
P[2:""]{
   GP2:
         UF : 0, UT : 1,
         J1=    360.000 deg
};
```

- Because PR[2] is set to the distance from current position to the position of PR[1], at line 1, the conveyor moves from the current position to the position of PR[1].
- At line 2, Reference position for Trigger is set by SLTKREST.
- R[3] is the counter for SLTKPSHQ. At line 7, 13, 22, R[3] is initialized, counted and reset.
- At line 17, the conveyor moves to PR[1] to correct iteration error. Every time the conveyor makes one revolution this line is executed.
- At line 20, the number of pitches is checked. When the conveyor moves 2 pitches, SLTKPSHQ is executed to save to a trigger value.
- If you would like to change the number of pitches per trigger. Please change “2” at line 20, 21 to the desired number.

9.7.4 LIMITATIONS ON SERVO CONVEYOR LINE TRACKING

- Servo Conveyor Line Tracking function requires Line tracking function.
- It is necessary to separate a robot and an extended axis as a conveyor into different groups.
- It is possible to add up to four servo conveyor to a controller. (G1:Robot, G2-G5:Servo Conveyor)
- Servo Conveyor Line Tracking function can be used together with a traditional encoder conveyor. However, if you would like to continue a tracking motion after a tracking motion of a different schedule, it is necessary to execute a normal motion once before next tracking motion starts after a tracking motion of a different schedule.
- Servo Conveyor Line Tracking function supports HDI and ACCUTRIG instruction.
- It is not possible to use Visual Tracking on Servo conveyor. (It is possible to use Visual Tracking on traditional encoder conveyor.)
- The tracking robot can not use Continuous Turn function.
- Original Path Resume feature is disabled.
- It is not possible to execute TP program of which a motion mask has the tracking robot group and Servo conveyor group.
- In a tracking program, Please add the wrist joint motion instruction (Wjnt) to all motion instruction or don’t add Wjnt to motion instruction. If there are a motion instruction with Wjnt and a motion instruction without Wjnt, TCP might be deviated from a destination at tracking motion.
- It is possible to specify flags on setup display.

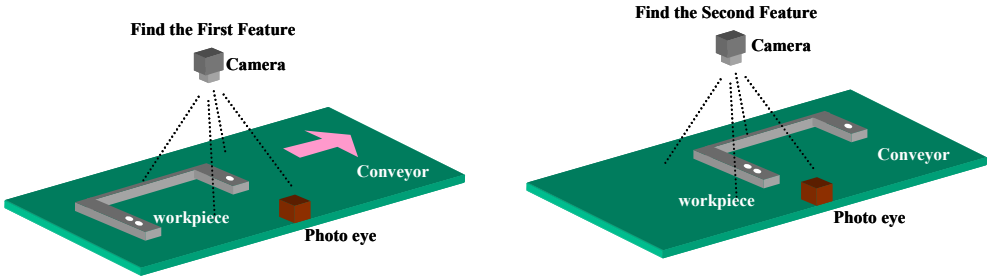
9.8 MULTI-VIEW

This chapter explains the setup of multi-view using one or two cameras in *i*RPickTool.

The multi-view function is useful in a case that a whole part cannot be detected by one measurement for iRPickTool application. A large size part such as an automotive floor panel moving on a conveyer belt is given as an example.

9.8.1 Part Detection Procedure

When you use the multi-view function, a part is detected as follows.
A sensor such as a photo eye detects the part. After the part has moved by specified distance, the first feature on the part is snapped and found. Then the second feature on the part is snapped and found when the part has moved by another specified distance. The sensor task combines the found results and calculates the offset data.



9.8.2 Limitations

- You can use multi-view function only in line conveyor. You cannot use this function in circular conveyor or servo conveyor.
- When you use two cameras, you need to connect them to the same robot controller.
- You cannot use the multi-view together with the tray function.
- You cannot use a single view vision process and a multi-view vision process simultaneously. However you can execute them by switching them exclusively using 6.8.4.1, “PKSNENBSENS.PC”. For example, when you want to switch “SENS1” to be disabled and switch “SENS2” to enabled in TP program, you call PKSNENBSENS.PC as below.

```
CALL PKSNENBSENS ('SENS1',Disable)
CALL PKSNENBSENS ('SENS2',Enable)
CALL PKSNSTART
```

  After calling PKSNENBSENS.PC, you have to call PKWCSTART.PC (or PKSNSTART.PC).
- Multi-view function cannot find two or more parts at the same time.
- Before finishing the both measurements of view 1 and view 2 for a workpiece, you cannot begin detecting another workpiece.

9.8.3 Preparation for Camera

Number of Cameras

You can use one or two cameras.
When you use one camera, the vision detects two different features on a part with the same camera. Therefore, these features go through a field of view for one camera to detect these features.
When you use two cameras, it detects the first feature on the part by one camera and the second one by the other camera. In this case, you can arrange the cameras layout.

Connection

When you use two cameras, you have to connect them to the same robot controller.

Camera Layout

You can arrange the camera layout.
But the time until camera snaps image after a DI signal is input is about 0.1 second depending on a hardware configuration as mentioned in Section 5.3, "CAMERA INSTALLATION AND CONNECTION". Therefore, it is recommended that you install a camera in the downstream side from the photo eye by about (conveyor speed x 0.1).

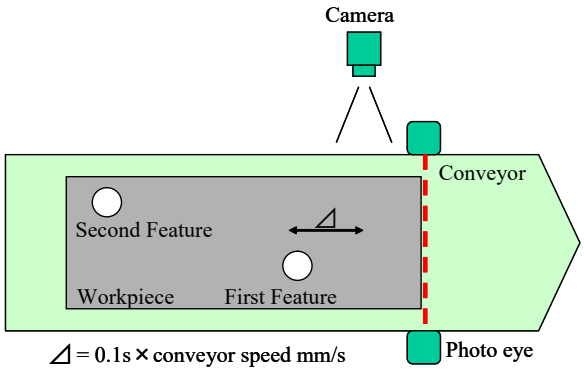


Fig. 9.8.3(a) Example of camera layout

9.8.4 Setup

The setup procedures of Conveyor, Camera data, creation of robot program, and execution of robot program are the same as those for a single view configuration. The procedures for other data are described below.

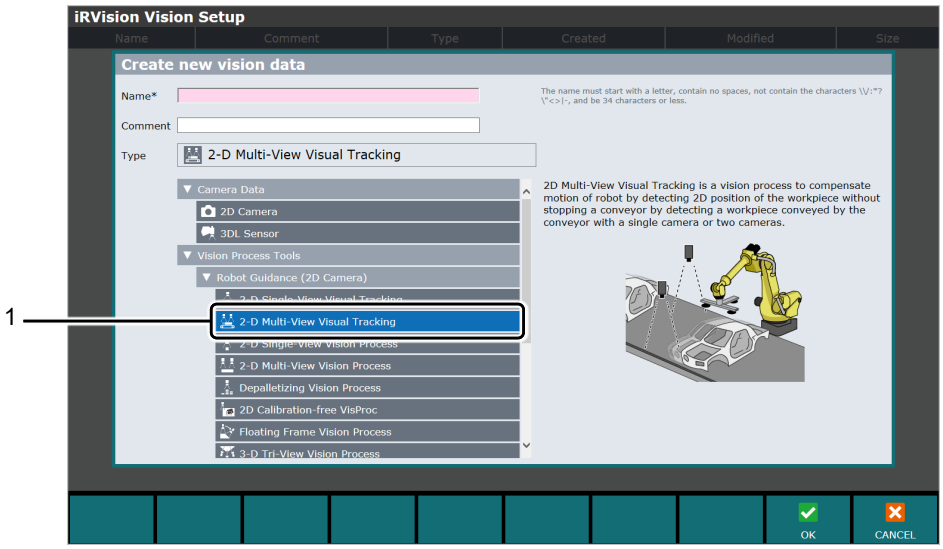
9.8.4.1 Vision process

In mulit-view, you use 2-D Multi-View Visual Tracking vision process. Camera view 1 is for the first feature on a part, and the camera view 2 is for the second feature on the same part.
Train vision process after making each feature be placed right under the corresponding camera view.

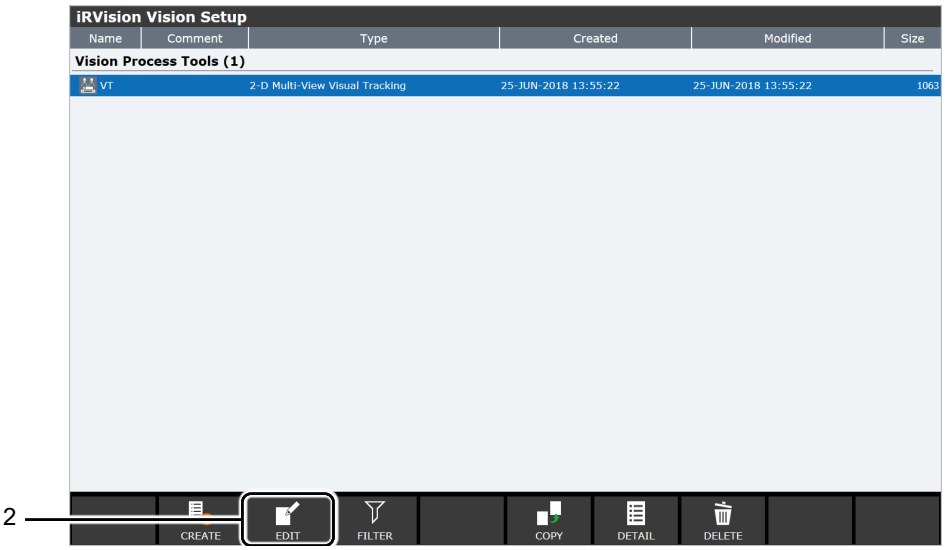
> ⚠ **CAUTION**
> It is necessary to set the camera view 1 for the detection first (the first feature) and the camera view 2 for the detection later (the second feature).

The setup procedures for the setting each camera view is shown below.

1 Create new vision data of [2-D Multi-View Visual Tracking] on the iRVision config screen.

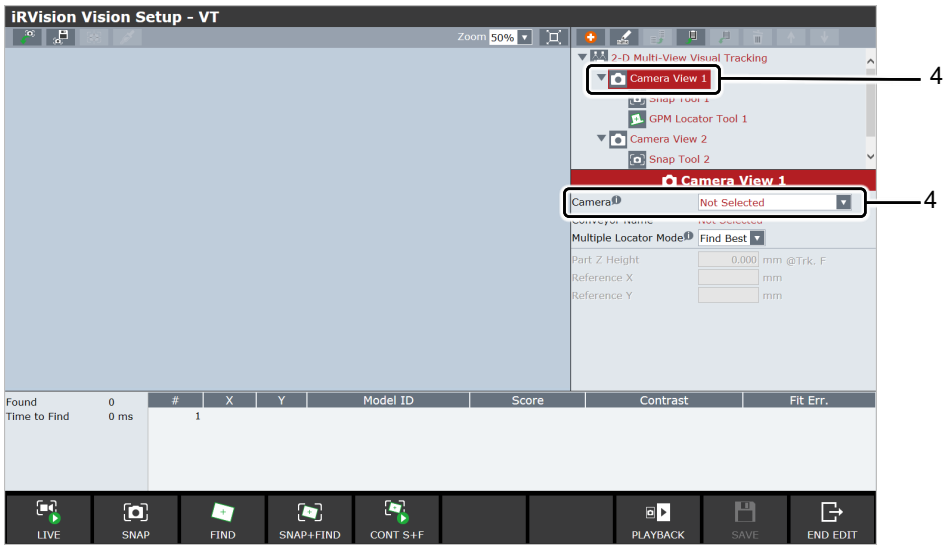


2 On the vision data list screen, select a created vision process, and click [Edit].



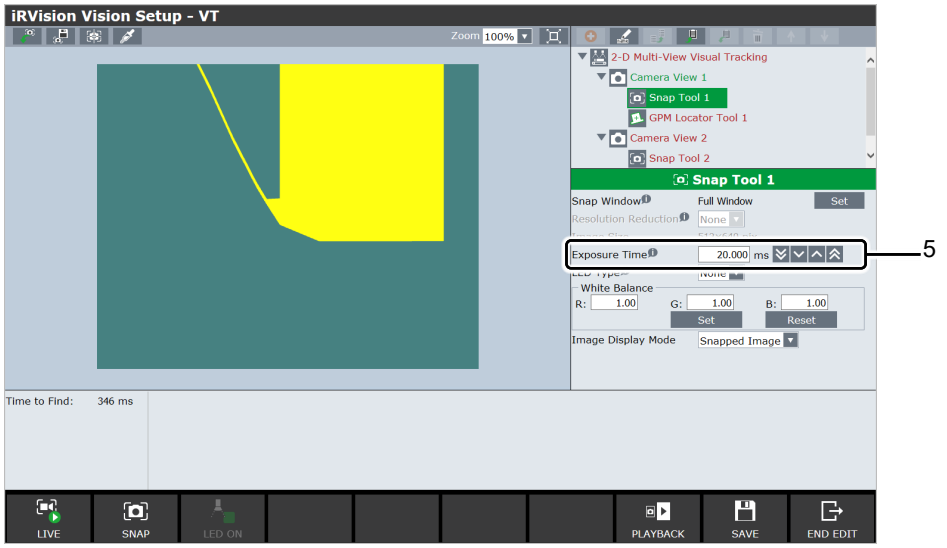
3 Set a part so that the first feature is inside the camera view 1.
If the supply direction is fixed, set it by aligning it.

4 Select [Camera View 1] from the tree view. Select the camera data to be used from the [Camera] drop-down box.



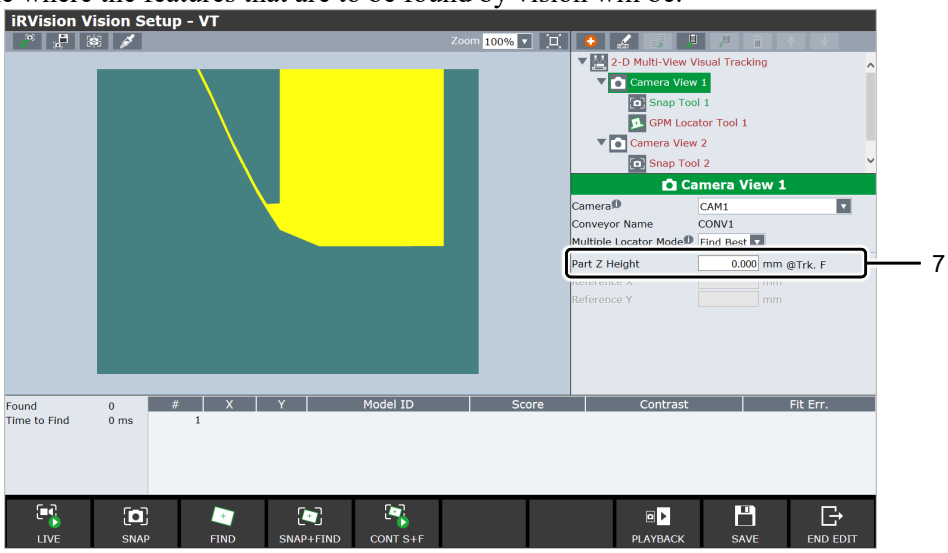
5 Select [Snap Tool 1] under the [Camera View 1], and enter an [Exposure Time].
Set the exposure time so that an image of a part that moves during exposure will have as little blurring as possible.
For, example, taking into account image blurring, in order to suppres misalignment during finding by vision to less than 0.1 mm when the conveyor speed is 100 mm/s, set avalue that is below 1ms as the exposure time.
0.1(mm) ÷ 100(mm/s) = 1(ms)

For information on calculation of appropriate exposure time to suit the conveyor speed, refer to “Exposure time” in Section2.5,”STYUDY FOR APPLICATION”.



6 Measure the thickness of a part.
Measure the distance from the plane at which the tracking frame was taught (the calibration grid surface) to the plane where the features that are to be found by vision will be.

7 Select [Camera View 1] from the tree, and enter the thick ness of a part in [Part Z Height]. Measure the distance from the plane at which the tracking frame was taught (the calibration surface) to the plane where the features that are to be found by vision will be.



NOTE

[Part Z Height] is the distance from the plane at which the tracking frame was taught (the calibration grid surface) to the plane where the features that are to be found by vision will be. Measure it using a ruler, etc.
The relationship between the system and'Part Z Height' is as shown in the following figure.

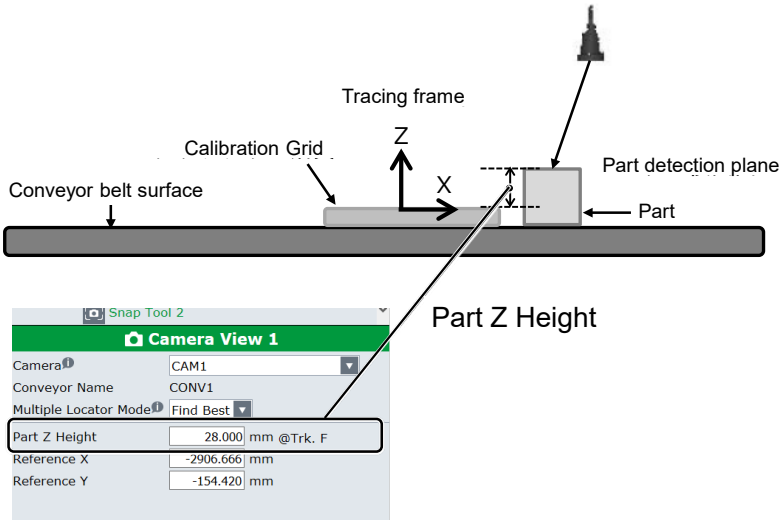


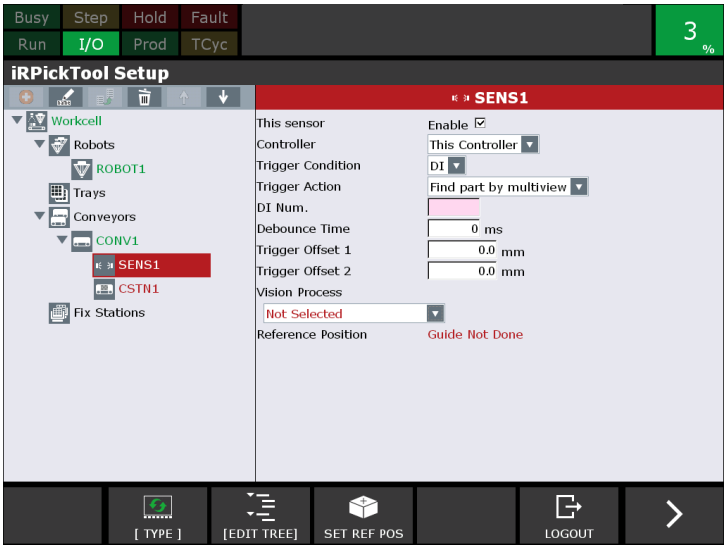
Fig. 9.8.4.1(a) Relationship between the system and 'Part Z Height'

8 Setup the [GPM Locator Tool1] under the [Camera View 1]. For information on setup GPM Locator Tool, refer to “”

9 Set a part so that the second feature is inside the camera view 2. Make the same setting as in steps 4 to 8 for camera view 2.

9.8.4.2 Settings of sensor task

Press [MENU] key on the teach pendant, select [6 SETUP] -> [iRPickTool].
In the displayed screen, when you select sensor object in a tree and set the cursor at [Trigger Condition] and select [DI] and set the cursor at [Trigger Action] and select [Find part by multiview], the following screen is displayed.



Trigger Offset 1, Trigger Offset 2

[Trigger Offset 1] is a distance in mm by which the conveyer moves after the part passes by the sensor such as a photo eye until the first feature is snapped and found by the camera view 1.
[Trigger Offset 2] is another distance in mm by which the conveyer moves after the part passes by the sensor such as a photo eye until the second feature is snapped and found by the camera view 2.

Case of one camera

When you use one-camera configuration shown in the figure below, specify [Trigger Offset 1] and [Trigger Offset 2] as follows.

Trigger Offset 1 = D1 (mm) - L1 (mm) - 0.1 (s) x Conveyor speed (mm/s)
Trigger Offset 2 = D2 (mm) - L1 (mm) - 0.1 (s) x Conveyor speed (mm/s)

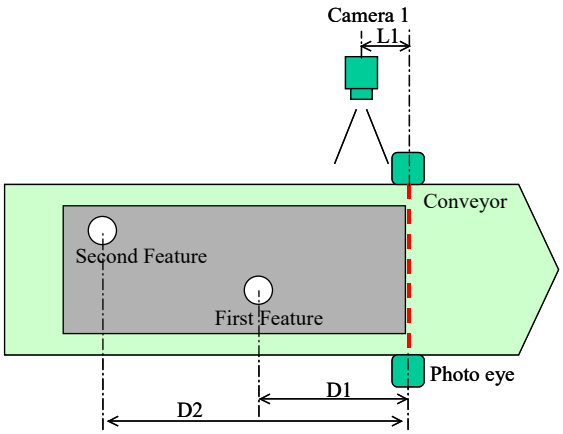


Fig. 9.8.4.2(a) Example of one-camera configuration

Case of two cameras

When you use two-camera configuration shown in the figure below to make Camera 1 find the first feature and to make Camera2 find the second feature, specify [Trigger Offset 1] and [Trigger Offset 2] as follows.

Trigger Offset 1 = D1 (mm) - L1 (mm) - 0.1 (s) x Conveyor speed (mm/s)
Trigger Offset 2 = D2 (mm) + L2 (mm) - 0.1 (s) x Conveyor speed (mm/s)

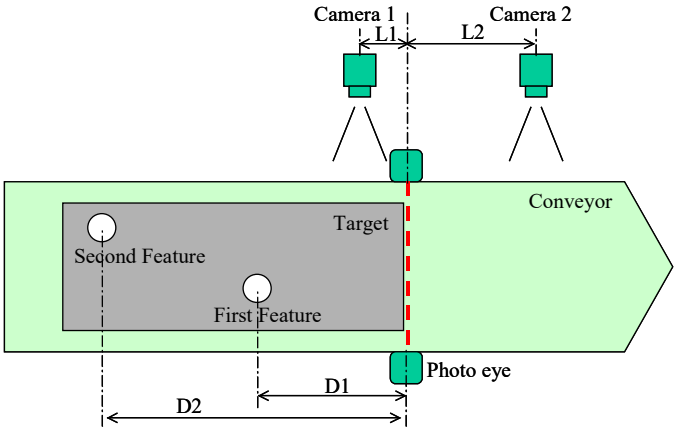


Fig. 9.8.4.2(b) Example of two-camera configuration

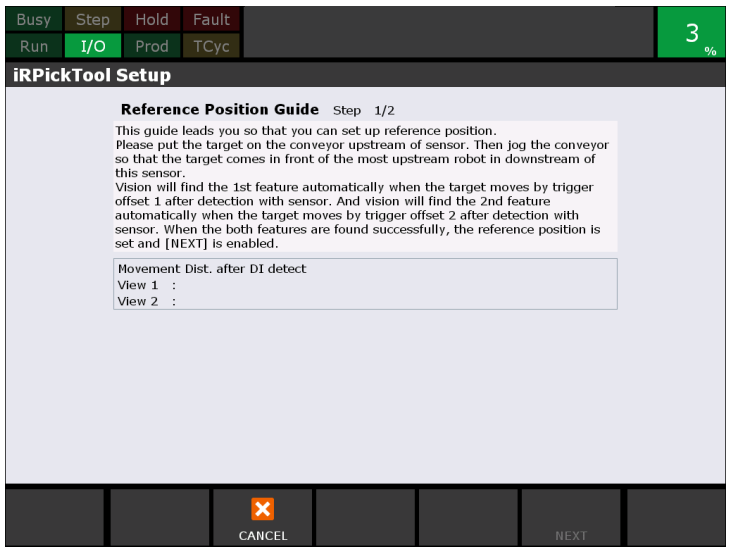
Vision Process Name

Specify the vision process name of 2-D Multi-View Visual Tracking vision process

9.8.4.3 Reference position setting and robot position teaching

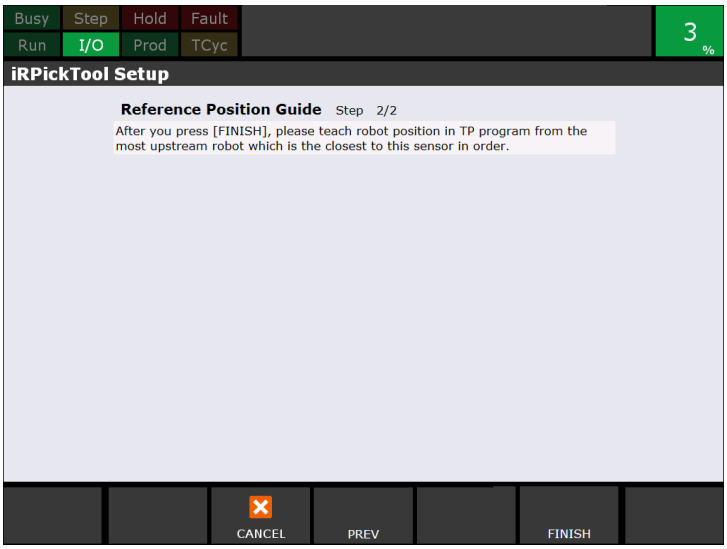
You set reference positions according to the Guide instructions.

1 Put a part at upstream side of the sensor such as a photo eye position on the conveyor.
2 Press F3 [SET REF POS] key to start the reference position setting Guide. The following screen is displayed.



3 Move the conveyer to make the photo eye detect the part. Then move the conveyor furthermore. Stop the conveyor when the part reaches in front of each robot. Here, when the part moves by [Trigger offset 1] distance after the photo eye detection, the first feature will be snapped automatically by Camera View 1. And when the part moves by [Trigger offset 2] distance after the photo eye detection, the second feature will be snapped automatically by Camera View 2.
When the part is detected by the photo eye successfully, the movement distance is displayed and updated. And if the part is successfully found in each view, each found position and the integrated found position of two found positions are also displayed and then F5 [NEXT] key is enabled.

When you press F5 [NEXT] key, the following screen is displayed.



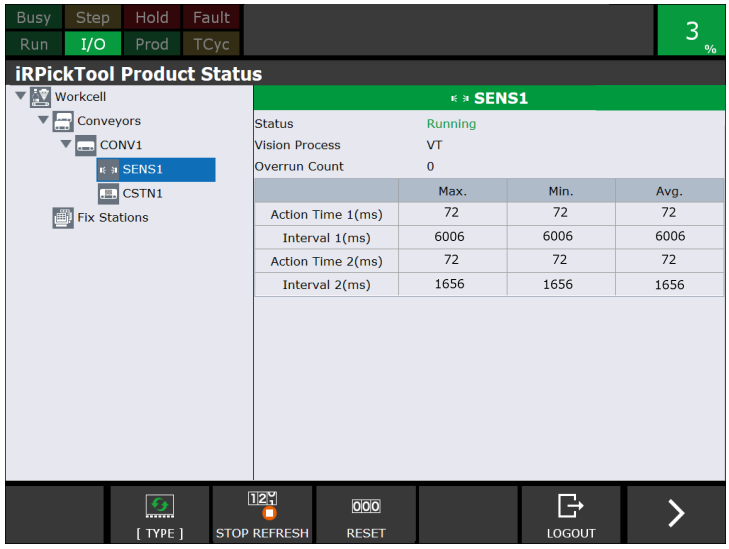
Memo
The trigger is set with the encoder value when it is detected by camera view 1

4 Teach the robot positions. This teaching procedure is the same as that for a single view configuration.
5 Press F5 [Finish] key to finish the Guide.

Memo
The position of the part get by PKCSGETQUE is the position between the feature1 and feature2. It is only after the teaching position of the robot exceeds upstream boundary to start chasing the part.

9.8.5 Production Status

The following screen is displayed when you use multi-view in iRPickTool production status.



Vision Process

Vision process name which you specify in [Vision Process] of a sensor.

Action Time 1

Total process time since the Camera View 1 is executed till its snap is taken.

Action Time 2

Total process time since the Camera View 2 is executed till the vision process result is input to the queue.

Interval 1

Interval between the start time of the Camera View 1 and the start time of the Camera View 2. If this value is smaller than Action Time 1, the overrun occurs.

Interval 2

Interval between the start time of the Camera View 2 and the start time of the Camera View 1. If this value is smaller than Action Time 2, the overrun occurs.

Memo

After a DI is input, the next DI input is ignored until a conveyer travels the distance as much as (Trigger Offset 2 - Trigger Offset 1). This is because it prevents that only the Camera View 1 is executed continuously. In this case, the Overrun Count is not updated.

9.9 3D VISION SENSOR

This section describes the method for setting up *i*RPickTool when using the 3D Vision Sensor.

9.9.1 Overview

You can pick the workpiece on the upper side of overlapped workpieces or tilted workpiece using the 3D Vision Sensor. By using depth images obtained by the 3D Vison Sensor, you can also perform 2D visual tracking for workpieces that are difficult to detect using normal 2D images.

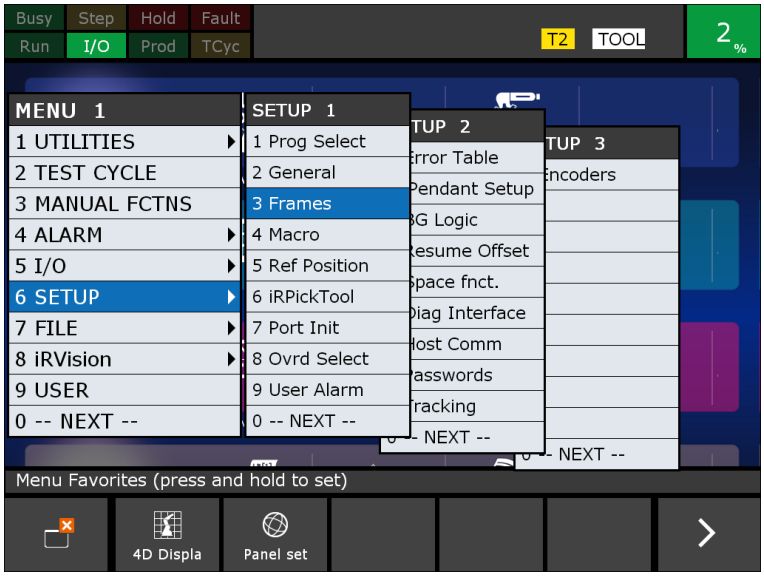
9.9.2 Restrictions

- The tray function cannot be used for conveyors using “3DV Single-View Visual Tracking” .
- The clearance check function cannot be used in conveyors using “3DV Single-View Visual Tracking”.
- Conveyors using “3DV Single-View Visual Tracking” cannot be selected as the infeed conveyor of the pre-grouping function.

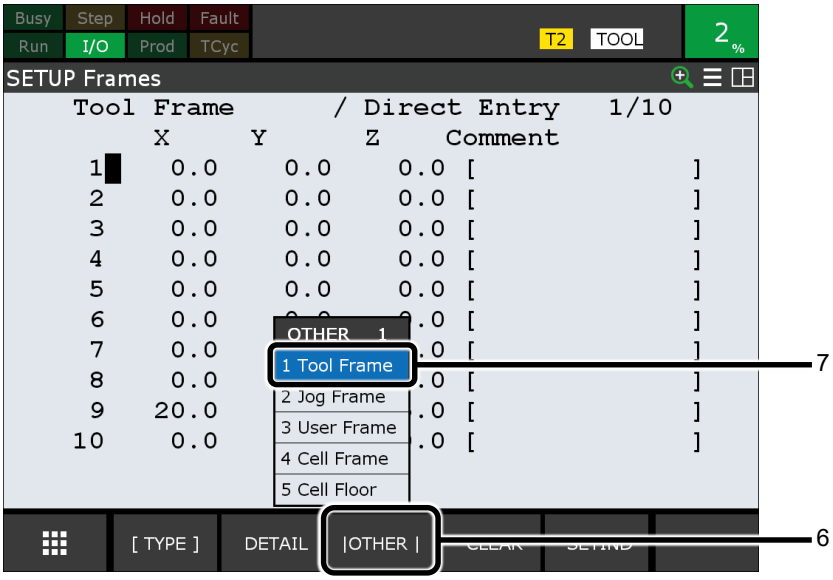
9.9.3 Mounting Position Setup

The following describes how to set up the mounting position.

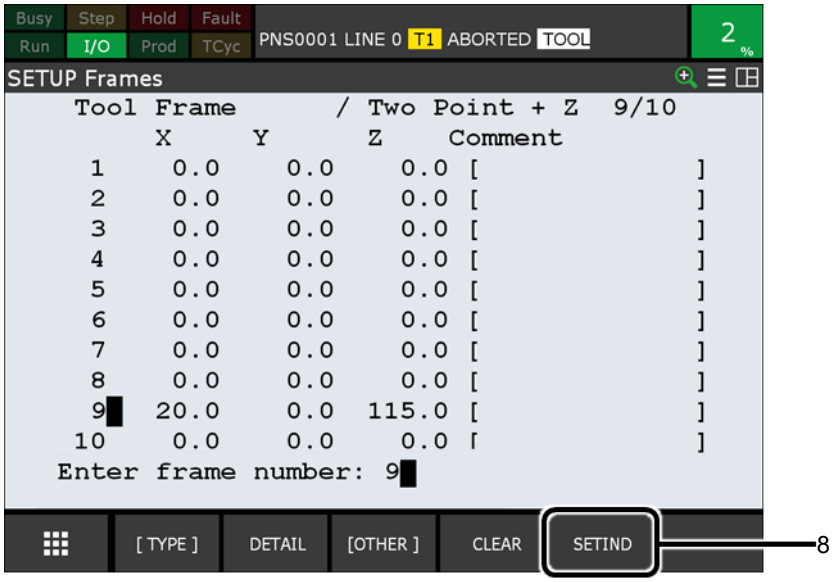
1 Prepare a grid pattern suitable for the 3D Vision Sensor's field of view.
2 Set the grid pattern directly blow the camera.
Fix it with tape to prevent it from moving.
3 The grid pattern can be set at any orientation. For example, for line tracking, match the flow direction of the conveyor and the X direction of the grid pattern.
4 Press the [MENU] key on the teach pendant of the robot controller.
The menu appears.
5 From the menu, select [SETUP] → [Frames].



6 Press F3 [OTHER].
7 Select [Tool Frame] from the menu.
The Tool Frame list screen appears.



8 Press F5 [SETIND].
9 Enter the frame number that has been used for the pointer tool.
This procedure is explained using a case where Tool 9 is used as an example.

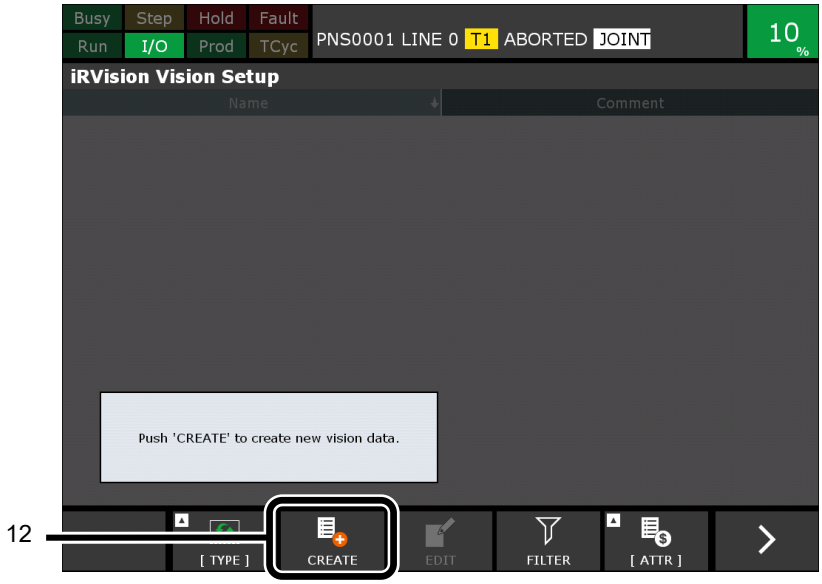


10 Press the [MENU] key on the robot controller's teach pendant.
The menu appears.

11 Select [*i*RVision] → [Vision Setup].

The [*i*RVision Vision Setup] screen appears.

12 Press F2 [CREATE].



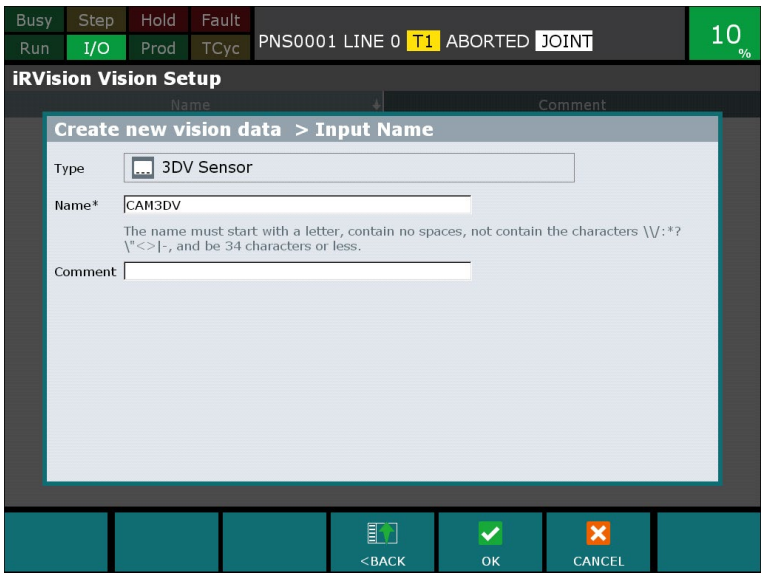
9

13 Select [3DV Sensor] and press F4 [NEXT>].

A screen such as that shown below appears.

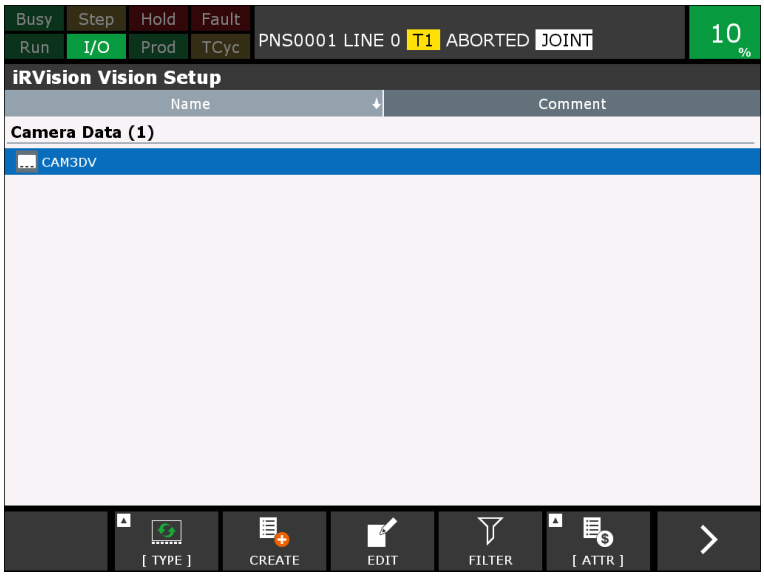
14 Enter the name of the 3DV Sensor in [Name].
Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.
Example: CAM3DV

15 Press F4 [OK].



The created camera data is displayed in the list.

16 Select the created camera data and press F3 [EDIT].

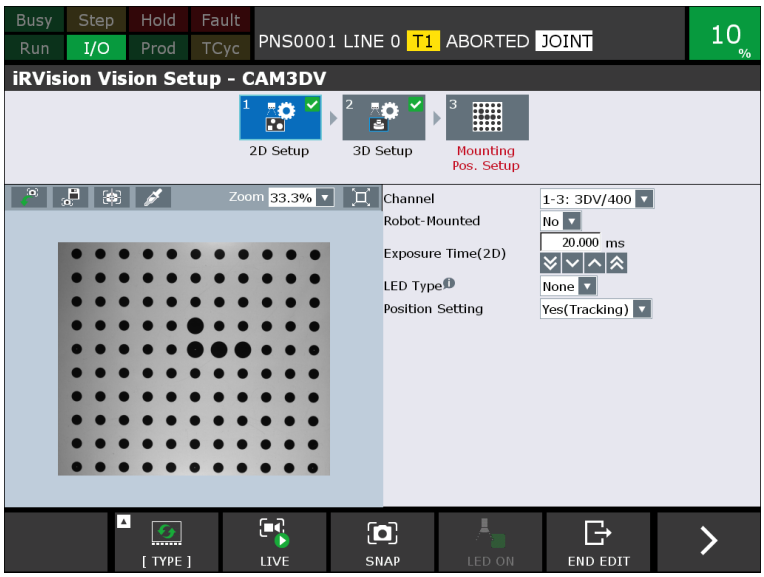


A screen such as that shown below appears.

17 With [2D Setup] selected in the navigation area, select [Channel] from the menu.

18 Enter [Exposure Time (2D)].

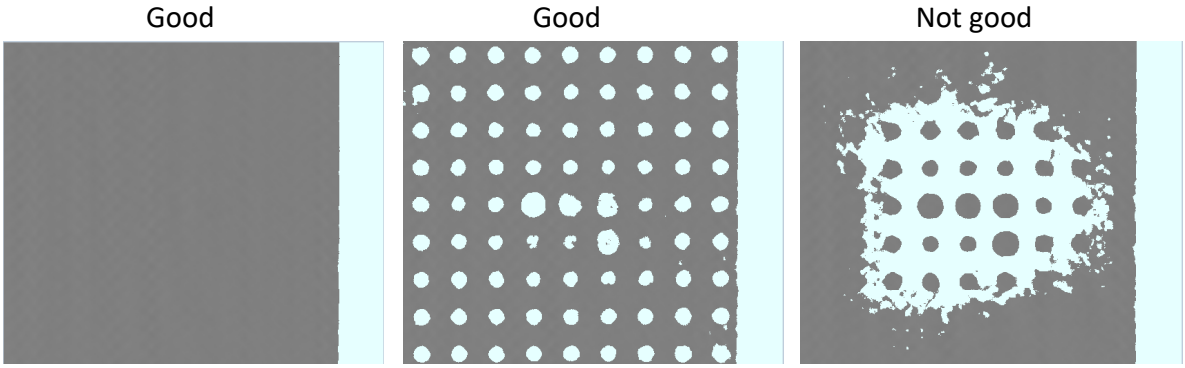
19 In [Position Setting], select [Yes (Tracking)] from the menu.



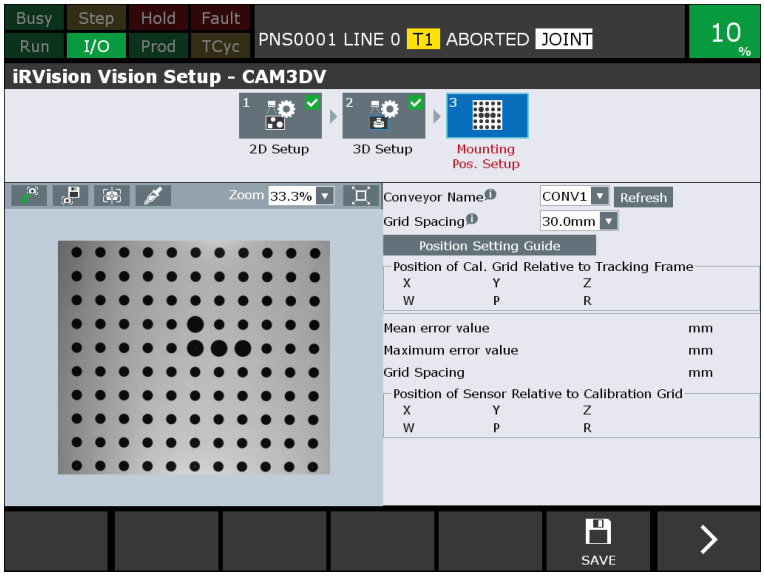
20 Select [3D Setup] in the navigation area.

21 Adjust exposure time, projector intensity and noise removal level so that depth data on surface of the grid pattern can be measured.

The depth data on the grid points are not necessary. Adjust so that depth data around grid points can be measured.

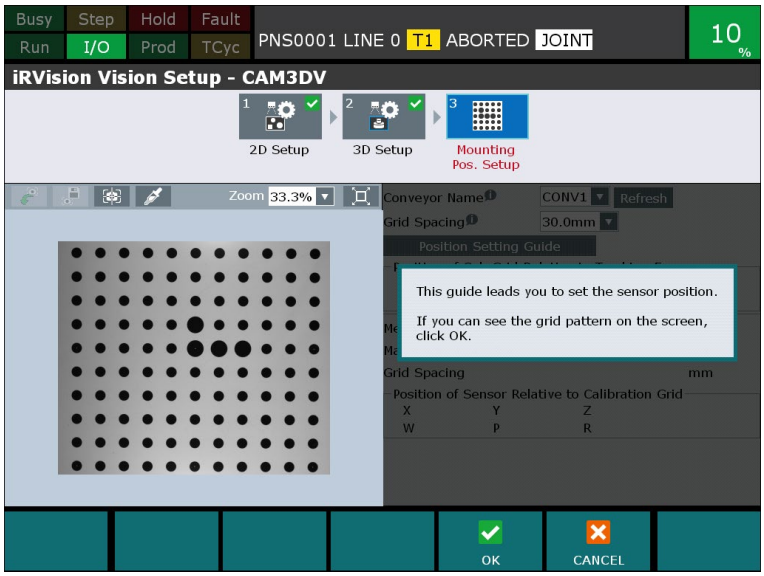


22 Select [Mounting Pos. Setup] in the navigation area.
23 Select [Conveyor Name].
24 Enter [Grid Spacing] of the grid pattern.
25 Press [> (Next page)] and press F5 [SAVE].

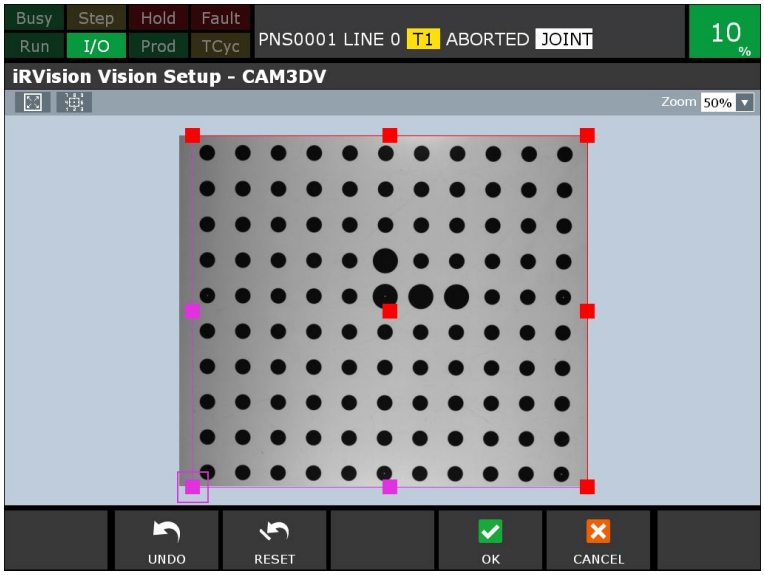


26 Press [Position Setting Guide].

27 Confirm that the grid pattern is displayed on the screen, and press F4 [OK].

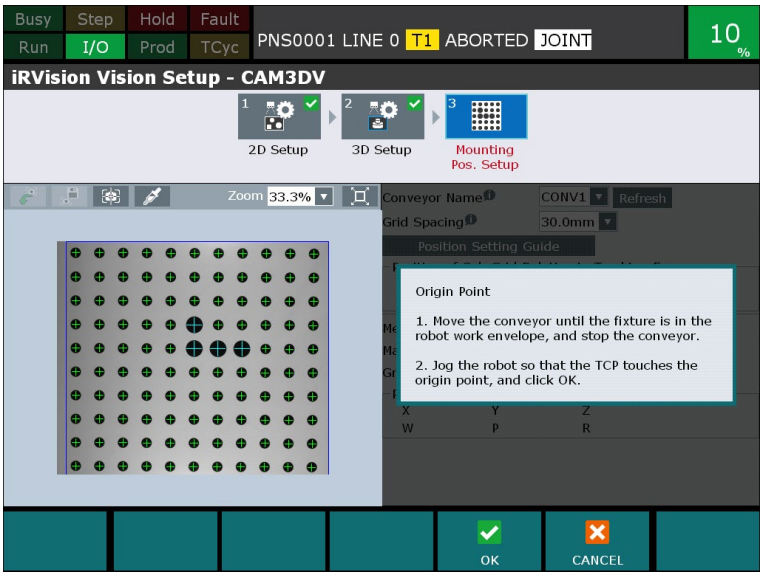


28 Enclose the dot pattern that appears on the screen with a reddish-purple rectangular window.
29 Press F4 [OK].



9

30 Move the conveyor, and stop when the grid pattern gets to around the center of the robot's operating range.



31 Touch up the origin of the grid pattern.

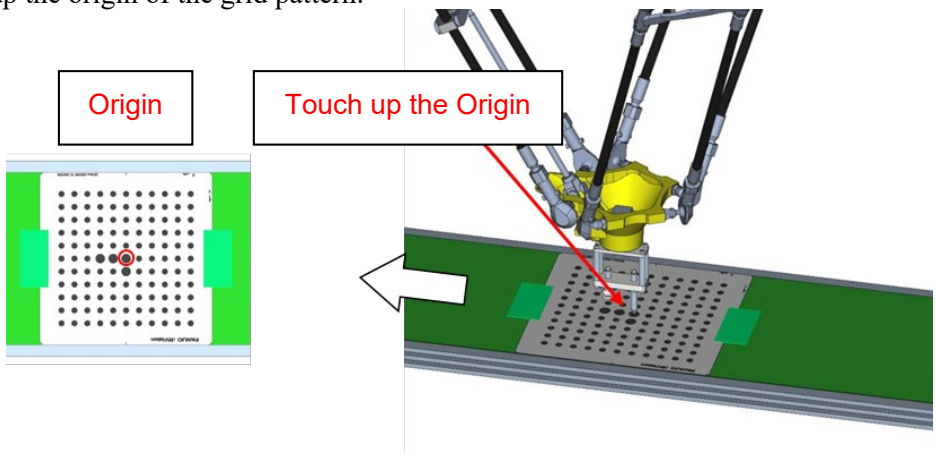


Fig. 9.9.3(a) Touching up the origin of the grid pattern

32 Press F4 [OK].

33 Without moving the conveyor, touch up the X-axis direction of the grid pattern.

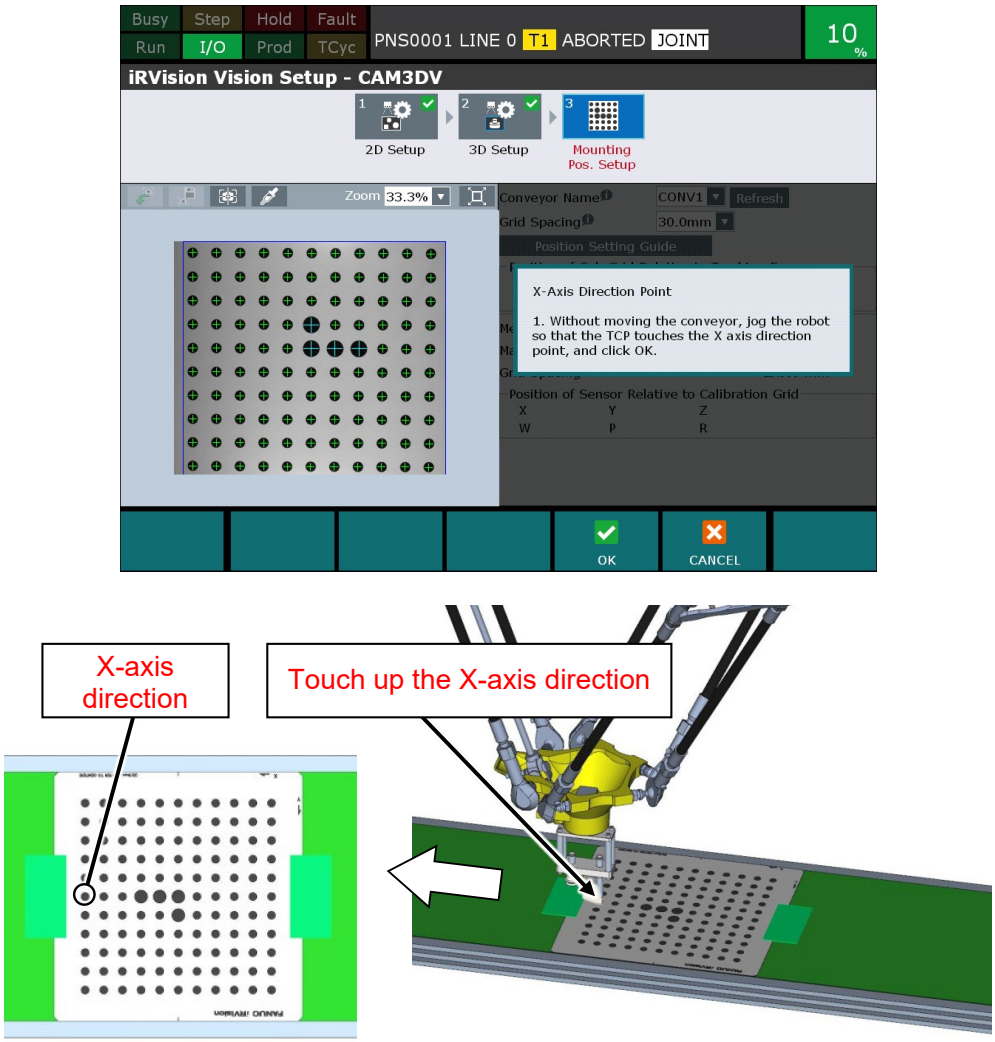


Fig. 9.9.3(b) Touching up the X-axis direction of the grid pattern

34 Press F4 [OK].
35 Without moving the conveyor, touch up the Y-axis direction of the grid pattern.

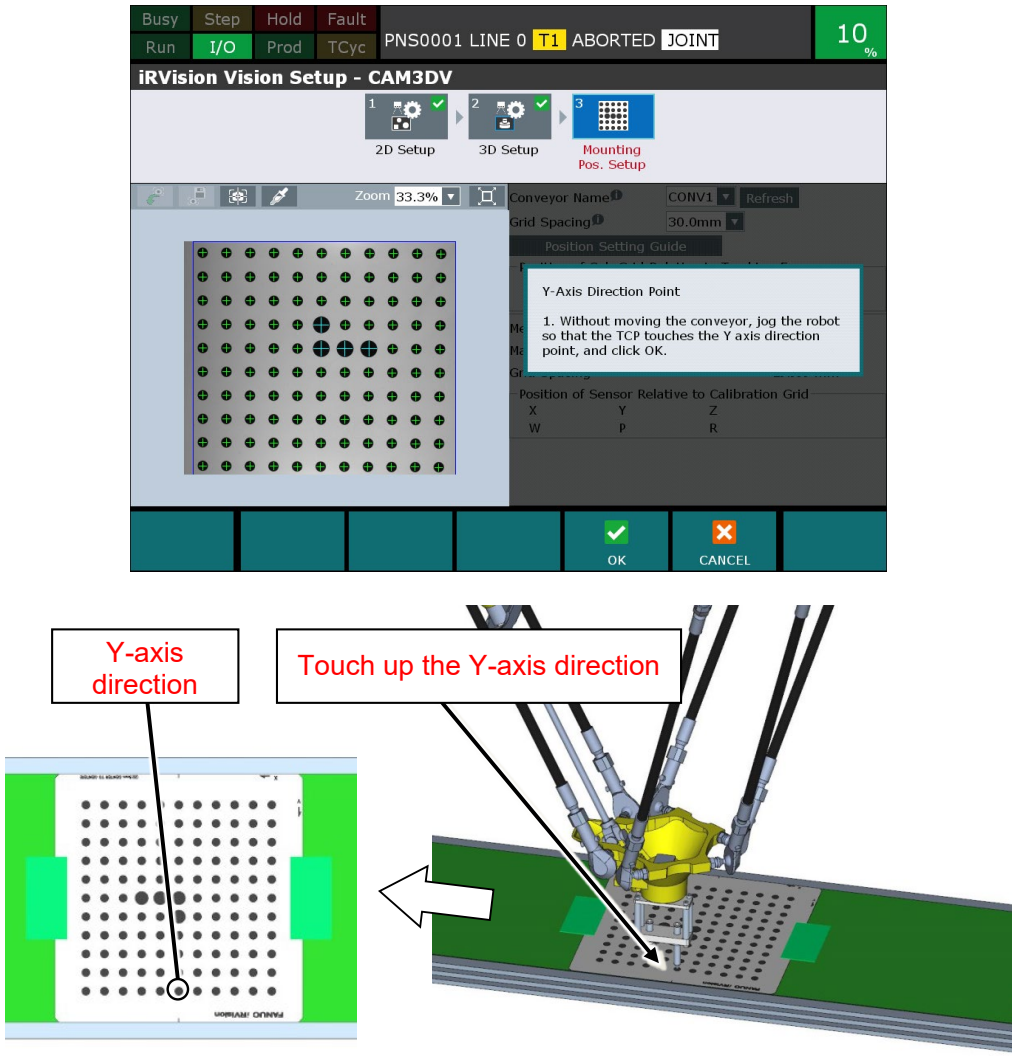
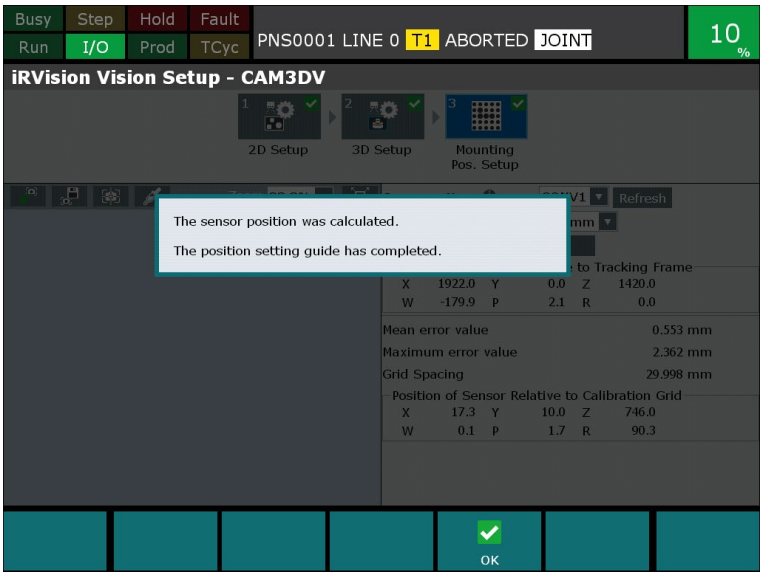
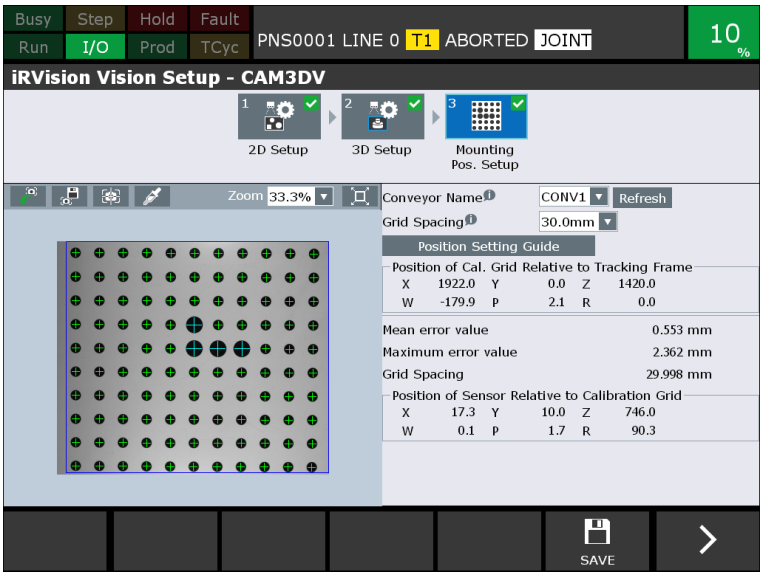


Fig. 9.9.3(c) Touching up the Y-axis direction of the grid pattern

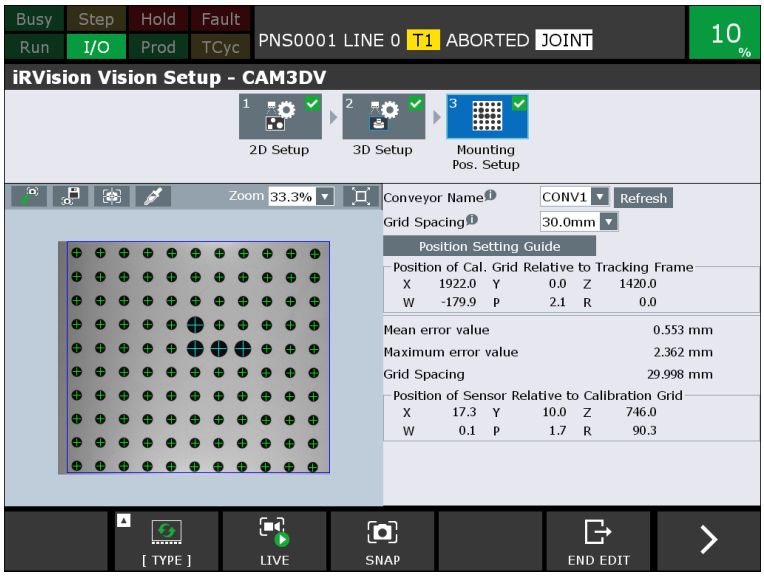
36 Press F4 [OK].
The mounting position is calculated.



37 Press F4 [OK].
38 Press [> (Next page)] and press F5 [SAVE].



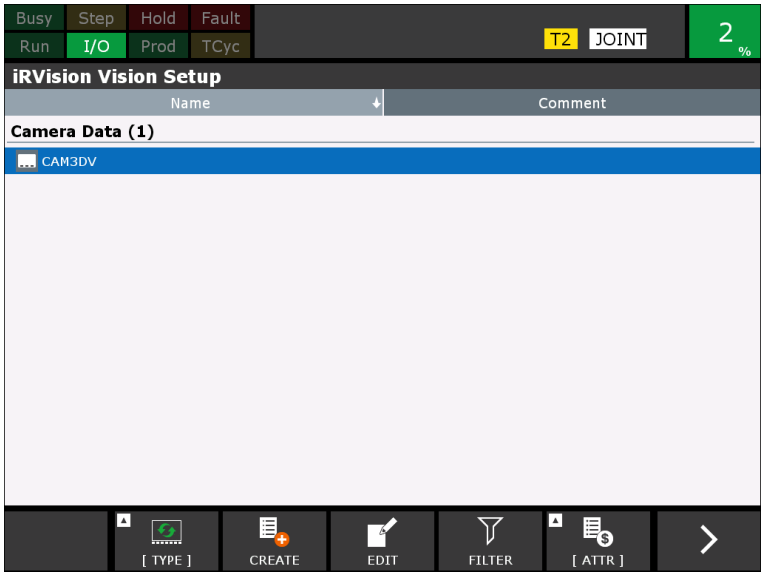
39 Press [> (Next page)] again and press F5 [END EDIT].



9.9.4 Vision process (3D)

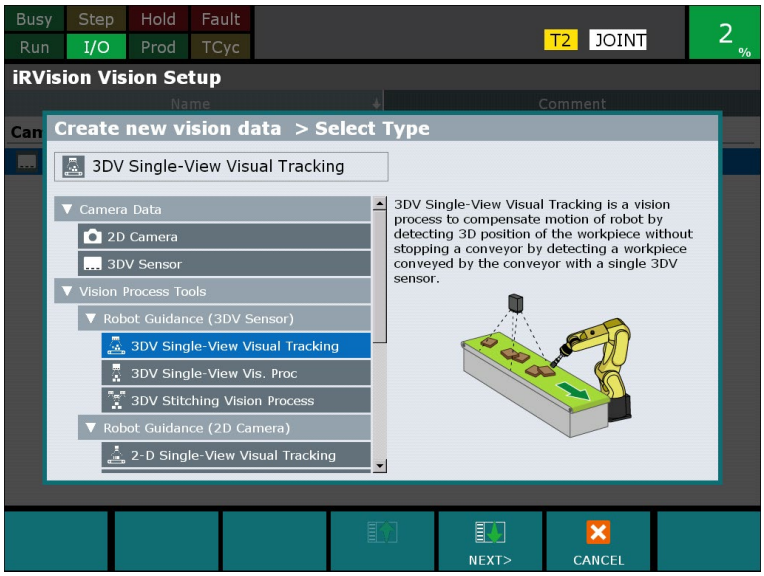
The following describes how to set up the vision process when picking up a part three-dimensionally, such as when picking up the part on the top of overlapped parts or a tilted part. For how to perform 2D visual tracking using depth images provided by the 3D Vison Sensor, refer to the following subsection, “Vision process (2D)” .

1 Press F2 [CREATE].



The [Create new vision data] screen appears.

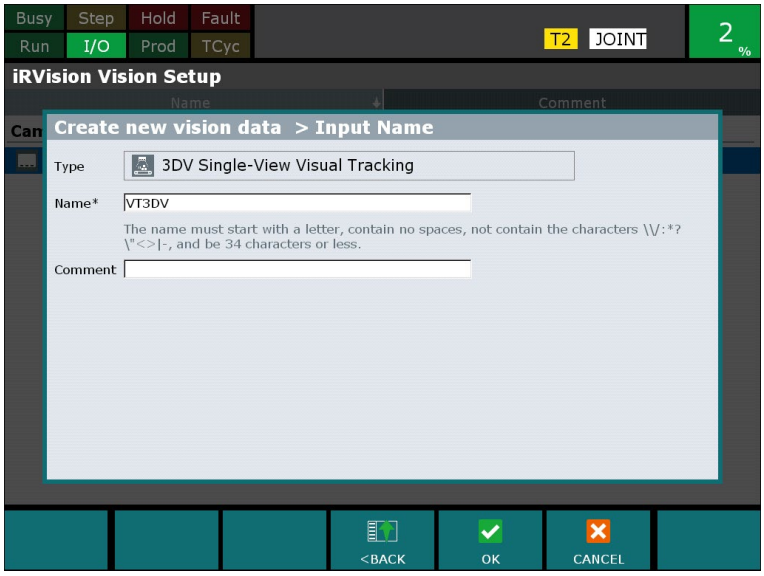
2 Select [3DV Single-View Visual Tracking] and press F4 [NEXT].



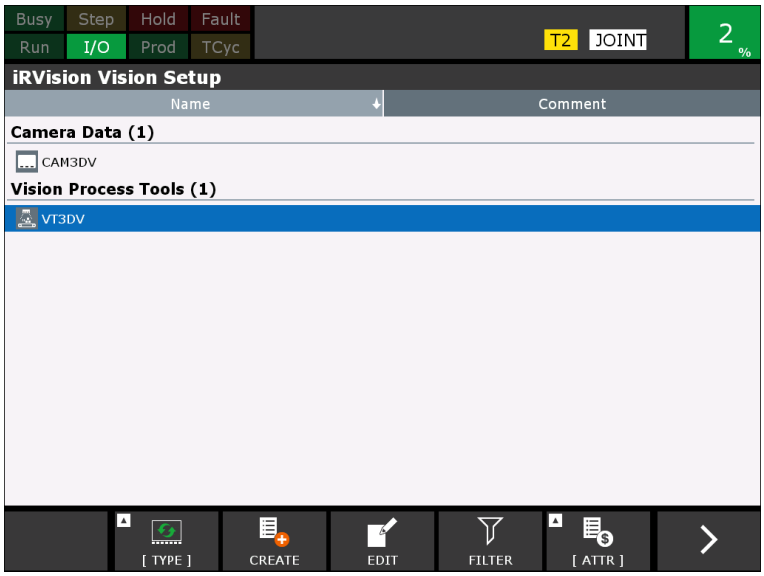
A screen such as that shown below appears.

3 Enter the vision process name in [Name].
Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.
Example: VT3DV

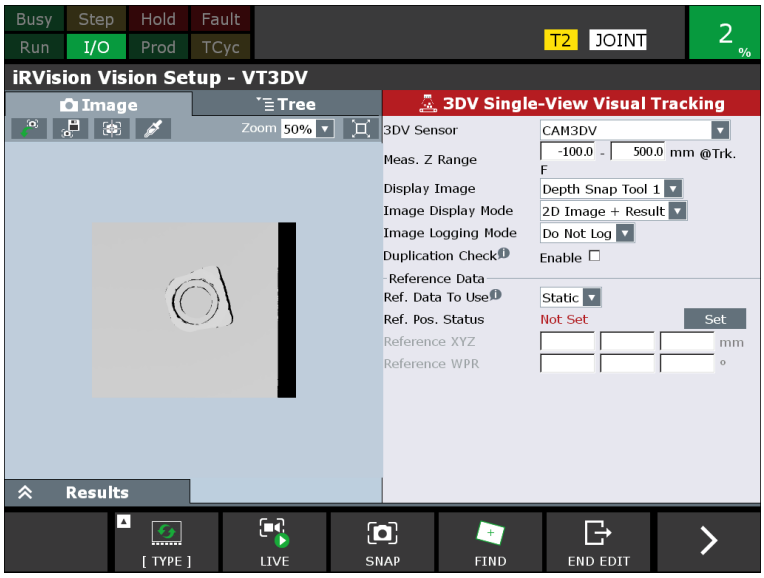
4 Press F4 [OK].



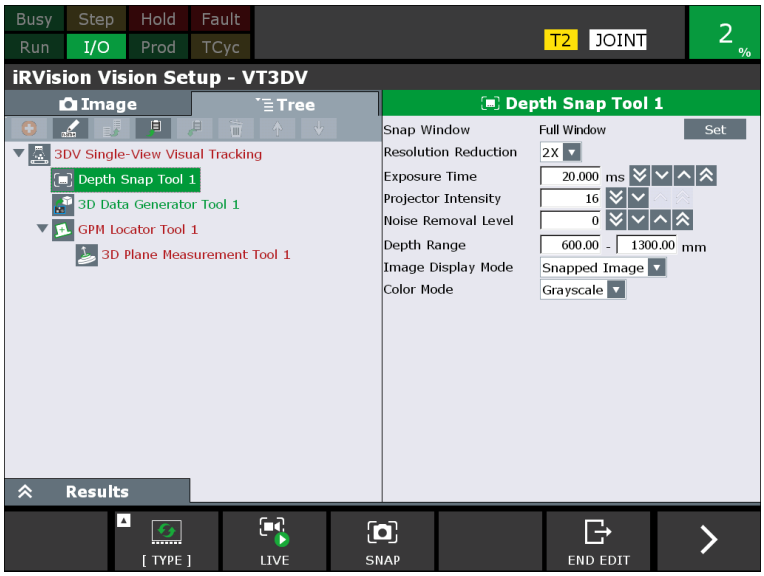
5 Select the created vision process and press F3 [EDIT].



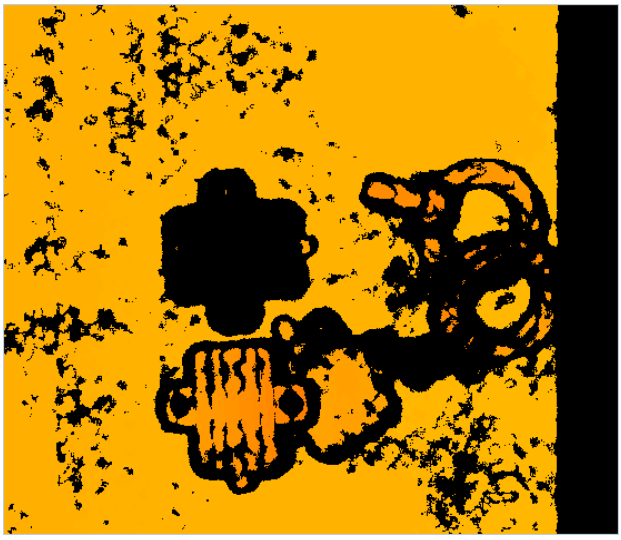
6 Select [3DV Sensor] from the menu.



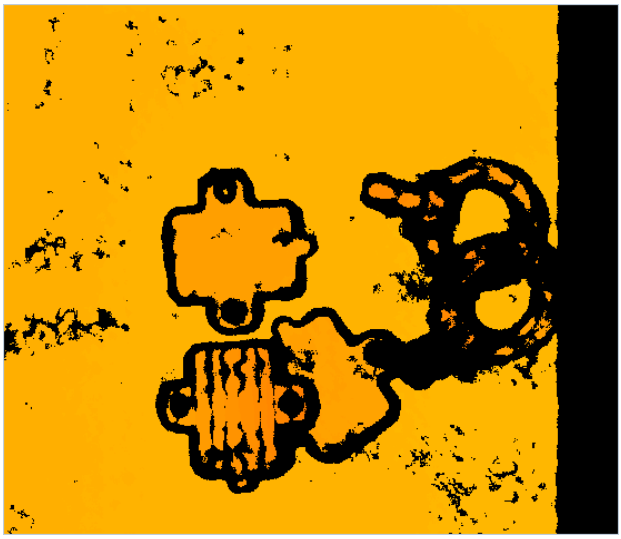
7 Press the Tree tab, and select [Depth Snap Tool 1].



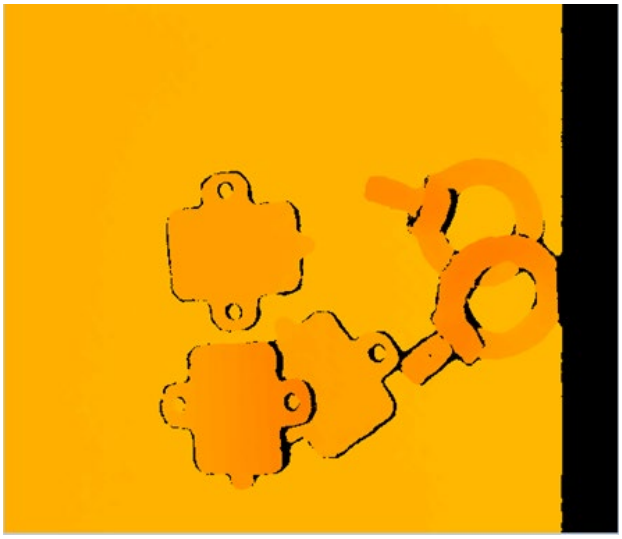
8 When you wish to capture a partial image, rather than a full depth image, set [Snap Window].
9 You can reduce the image size at the time of snapping the image to reduce the resolution. Select the degree of reduction in the drop-down box for [Resolution Reduction].
10 Set [Depth Range]. This is the depth range to be converted into a grayscale image. The three-dimensional points in the specified depth range will be converted into a grayscale image and output to a 2D command tool.
11 Adjust [Exposure Time], [Projector Intensity], and [Noise Removal Level] by following the procedure below.
12 Set [Color Mode] to Color. The pixels for which depth could not be measured are displayed in black.
13 Set [Projector Intensity] to 16 and [Noise Removal Level] to 10. These are tentative settings for correctly adjusting [Exposure Time]. Black pixels may temporarily increase.



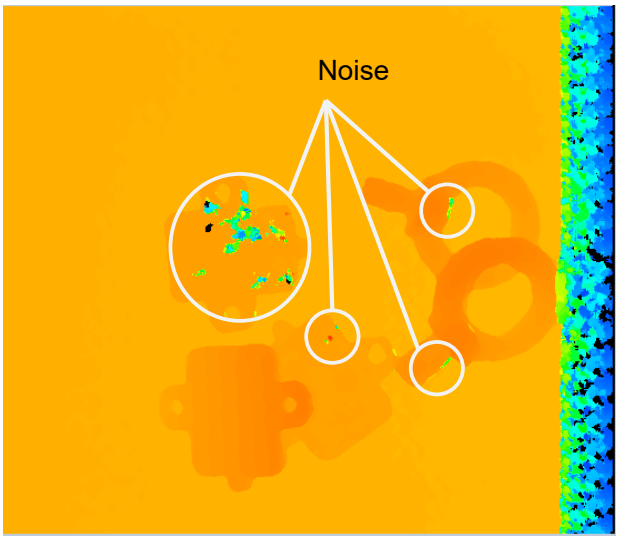
14 Adjust [Exposure Time] so as to reduce black pixels to the extent possible.
15 Reduce [Projector Intensity] so as to reduce black pixels to the extent possible. When there is no noticeable change in the image even after changing [Projector Intensity], set a large number for [Projector Intensity]. The larger the value, the more resistant the image is to ambient light.



16 Reduce [Noise Removal Level] so as to reduce black pixels to the extent possible within a range in which noise does not increase.



When a small value is set for [Noise Removal Level], snapped depth images will be noisy. The following is an example of an image with noise.



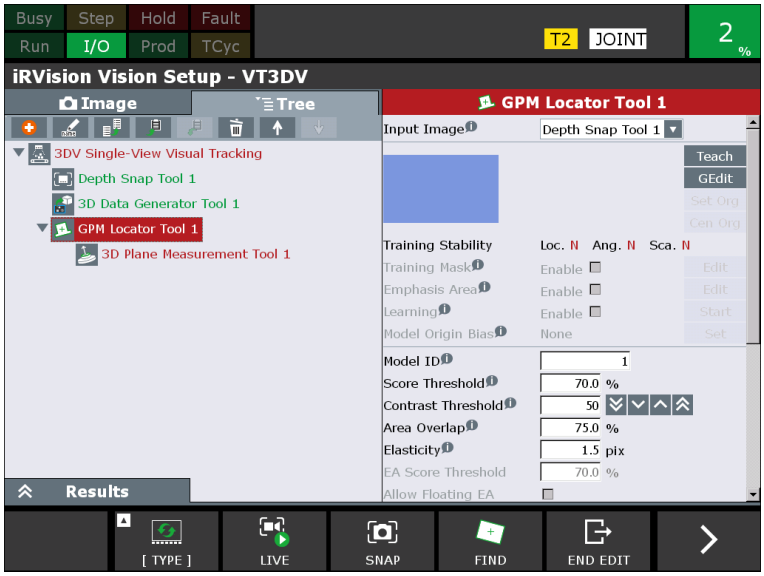
17 Press the Tree tab and select [3DV Single-View Visual Tracking].
18 Enter [Meas. Z Range]. Enter the range of Z in the tracking frame.

NOTE
When the Z-axis direction of the tracking frame is downward, enter the range of Z in the “detection frame” in [Meas. Z Range]. The “detection frame” refers to the frame in which the tracking frame is rotated 180 degrees around the X-axis, with the Z-axis facing upward.
To be specific, when the grid pattern position viewed from the tracking frame at the time of mounting position setup is downward, the direction of the Z-axis of the tracking frame is downward. For example, the direction of the Z-Axis of the tracking frame is downward in the case of a circular conveyor that moves clockwise.

Each color represents the following.
Blue – red: Three-dimensional points used for detection within the measured Z range.
Black: Three-dimensional points that could not be measured in the image snapped by the 3D Vision Sensor. If the target part is displayed in black, re-perform the procedure from setup of the depth snap tool.
While: Three-dimensional points that are located higher than the upper limit of the measured Z range. These points are not used for detection.
Gray: Three-dimensional points that are located lower than the lower limit of the measured Z zone. These points are not used for detection.

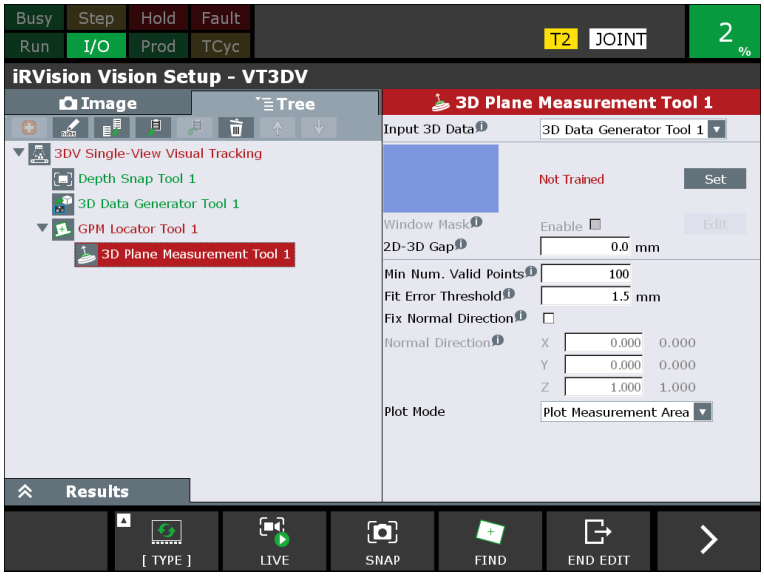
Perform adjustment so that the conveyor surface become bluish and the top surface of the part becomes red.

19 Press the Tree tab and select [GPM Locator Tool 1].



20 Place the part so that it is inside the measurement area.
21 Press F3 [SNAP] to capture an image.
22 Press [Teach].
23 Enclose the part with a reddish-purple rectangular window.
24 Press F4 [OK].
25 Press [EDIT] of [Training Mask].
26 Select a tool at the top of the screen, and paint in red the section you do not want to make a feature for matching.

27 Press F4 [OK].
28 Set up the DOF as necessary.
For details of the setup, refer to the section “GPM LOCATOR TOOL” in the chapter “COMMAND TOOLS” in the “*i*RVision OPERATOR'S MANUAL (Reference)”.
29 Press the Tree tab, and select [3D Plane Measurement Tool 1].



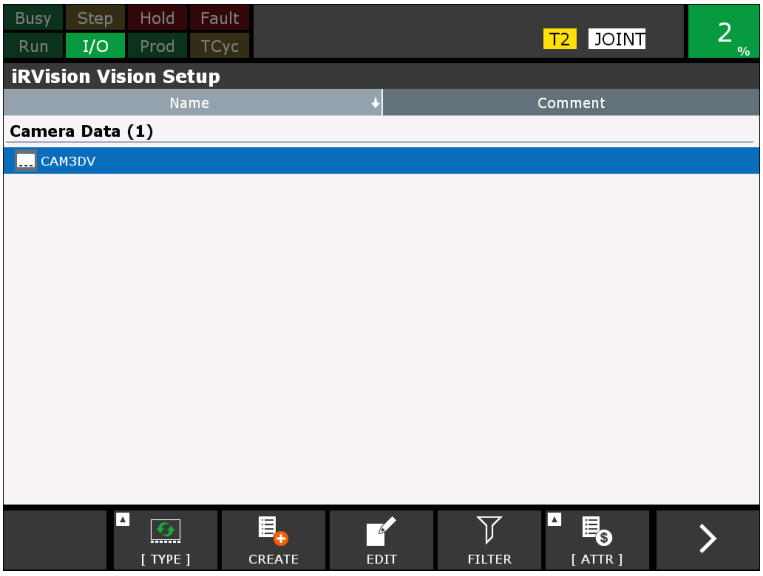
30 Place the part so that it is inside the measurement area.
31 Press [Teach].
32 Enclose the area for which the plane is measured with a reddish-purple rectangular window.
33 Press F4 [OK].
34 Enter [2D-3D Gap].
[2D-3D Gap] is the height difference between the measured plane and 2D measurement surface. When the 2D measurement surface is closer to the camera than the measured plane, this is a positive value.
35 Press [Edit] of [Window Mask].
36 When there are sections that you wish to exclude from the measurement area, such as a stepped plane inside the measurement area, click the [Edit] button of [Window Mask] and teach the Window Mask. Even when the Window Mask is taught, it will be ignored if you uncheck the [Enable] checkbox.

37 Press [> (Next page)] and press F5 [SAVE].
38 Press [> (Next page)] again and press F5 [END EDIT].

9.9.5 Vision Process (2D)

The following describes the method for performing 2D visual tracking using depth images obtained by the 3D Vision Sensor.

1 Press F2 [CREATE].



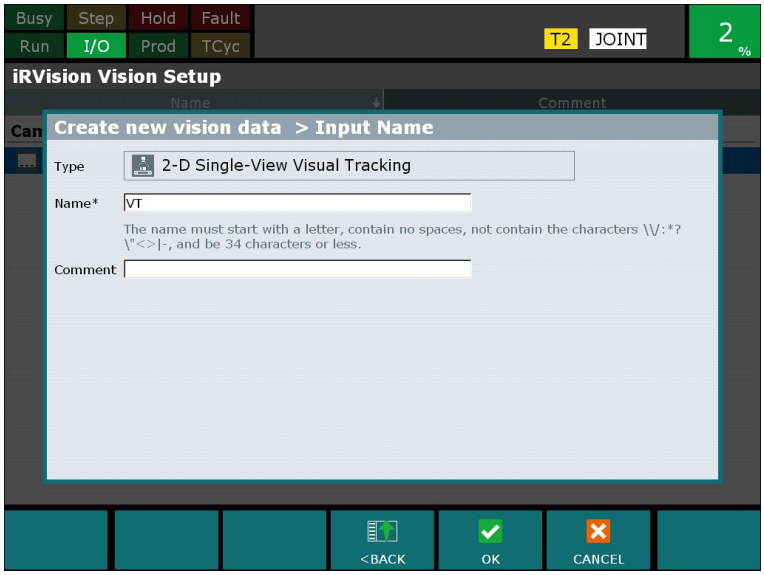
The [Create new vision data] screen appears.

2 Press [2-D Single-View Visual Tracking], and select F4 [NEXT].

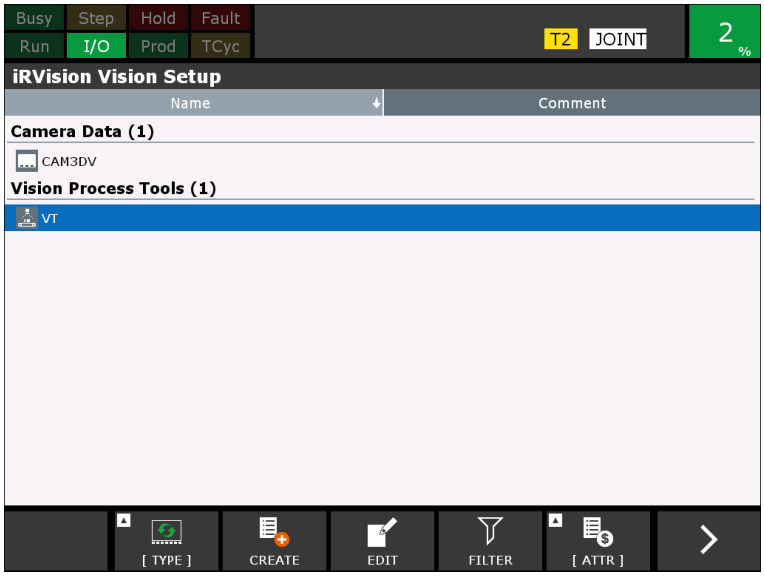
A screen such as that shown below appears.

3 Enter the name of the vision process in [Name].
Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.
Example: VT

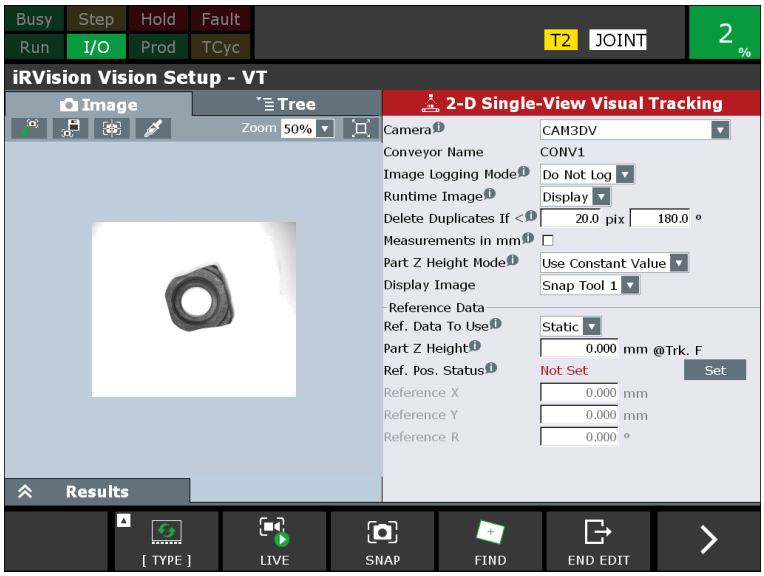
4 Press F4 [OK].



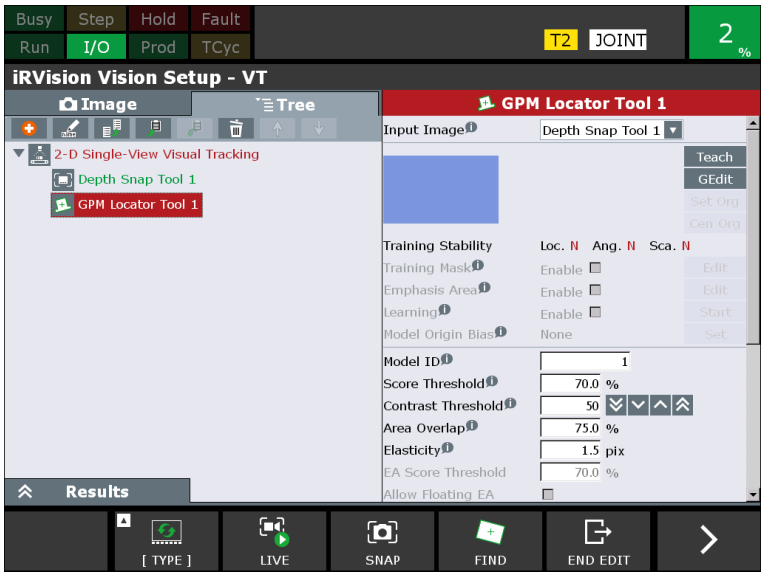
5 Select the created vision process and press F3 [EDIT].



6 Select [Camera] from the menu.



7 Select [Snap Tool] in the CREATE screen of the command tool to create a new Depth Snap Tool.
8 Press the Tree tab and select [Snap Tool 1], and delete it by pressing the button.
9 Press the Tree tab and select [GPM Locator 1], and select [Depth Snap Tool 1] for [Input Image].



10 Set [Depth Snap Tool 1] and [GPM Locator Tool 1] by following the same procedures as those described in the previous subsection, “Vision Process (3D)” .
11 Press [> (Next page)] and press F5 [SAVE].
12 Press [> (Next page)] and press F5 [END EDIT].

9.10 CLEARANCE CHECK FUNCTION

9.10.1 Overview

During visual tracking, you may accidentally move the parts around the target part when you are trying to pick a part from densely packed parts. In such a case, this function enables you to set an arbitrary range (clearance) around each part, check if there are other parts in the clearance of the part to be picked, and pick the part only when there is no other parts in the clearance.
For example, in a case such as described in the figure below, parts A and E can be picked because there is no other part in the clearance. However, parts B, C, and D cannot be picked because there is another part in the clearance. When part E is picked, however, there is no longer another part in the clearance of part D, and therefore part D can be picked.

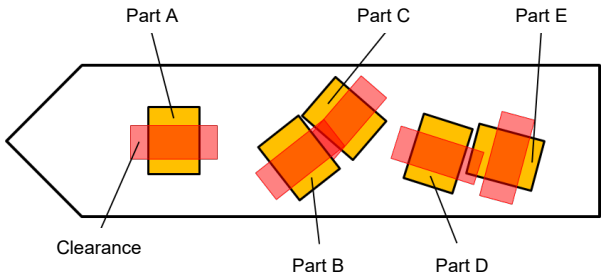


Fig. 9.10.1(a) Example of clearance check

9.10.2 Restrictions

Clearance check cannot be used if any of the following conditions apply.

- Two or more sensors are used for one conveyor station.
- Any of the following settings are enabled in the conveyor setup screen:
  - [Tray] is selected.
  - [Pre-grouping] is used.
  - [Servo Conveyor] is used.
  - [Y Sort] is used.
- Any of the following are enabled in the sensor setup screen:
  - When ROBOGUIDE is not used.
    [Trigger Condition] and [Action] are not [Distance] and [Find part by vison].
  - When ROBOGUIDE is used.
    [Trigger Condition] and [Action] are not [Distance] and [Find part by vision] or [DI] and [RG put part in queue].
  - A vision process of [3DV Single-View Visual Tracking] is selected.

9.10.3 Restrictions

A clearance check may not be performed correctly in the following cases:

- Two or more models of parts are simultaneously supplied to the conveyor.
  A clearance check can be set only for a single model of part. For this reason, when two types of parts are simultaneously supplied to the conveyor, as shown in the figure below, part B does not become the target of the clearance check, even though it is in the clearance of part A for which the clearance check is set, because the clearance check is not set for part B. Therefore, it is judged that part A can be picked.

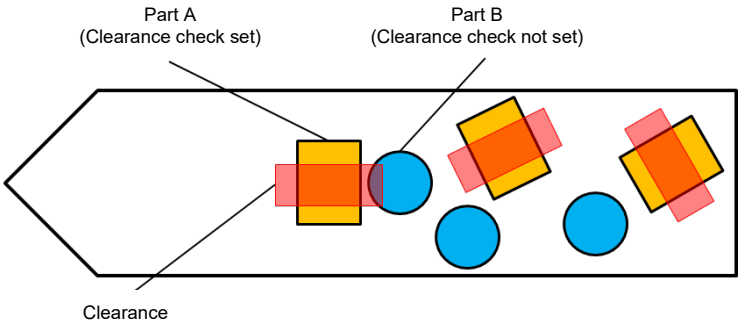


Fig. 9.10.3(a) Example in which two or more models of parts are supplied to a conveyor

NOTE
If the model and model origin are in the same position, a clearance check can be used even when multiple model IDs are used.

- When the field of view of the sensor is close to the upstream boundary of the tracking area of the most upstream robot.
  In the case of the figure below, part B is in the clearance of part A, but is not included in the target of the clearance check because it not detected. Therefore, it is judged that part A can be picked.

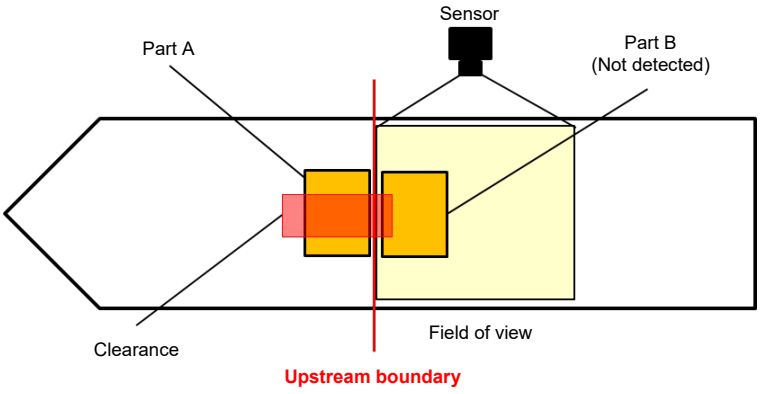


Fig. 9.10.3 (b) Example in which the field of view of the sensor is close to the upstream boundary of the tracking area of the most upstream robot

- The send position (send line) of the upstream robot is close to the upstream boundary of the tracking area of the downstream robot.
  In the case of the figure below, part B is in the clearance of part A. However, part B has not crossed over the send position (send line) of robot 1 and data of part B has not been sent to robot 2. Therefore, part B is excluded from the clearance check target. As a result, it is judged that part A can be picked.

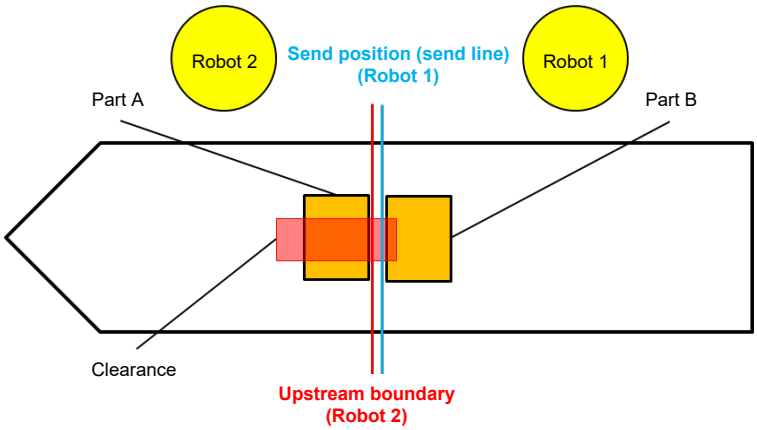


Fig. 9.10.3 (c) Example in which the send position of the upstream robot is close to the upstream boundary of the tracking area of the downstream robot

9.10.4 Setup

When using the clearance check function, first complete the procedures described in Chapter 5, “OTHER PREPARATIONS BEFORE SETUP”.

9.10.4.1 Setup of a workcell and robot

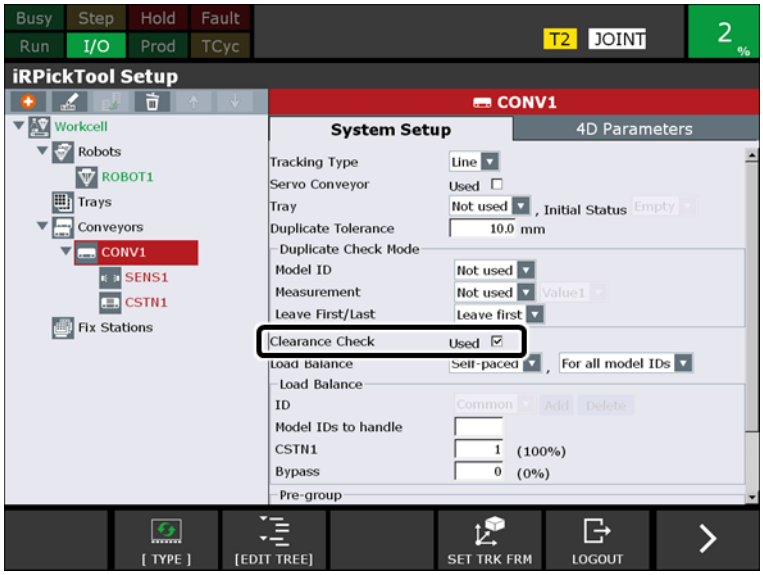
There are no special notes concerning the setup of a workcell and robot for clearance check. Refer to Sections 6.1, “SETUP OF A WORKCELL” and 6.2, “SETUP OF A ROBOT”.

9.10.4.2 Setup of a tray

There are no special notes concerning the setup of a tray for a clearance check.
Trays cannot be used in a conveyor in which a clearance check is used. When using trays in a conveyor not using a clearance check, or when using a fixed station, refer to Section 6.3, “SETUP OF A TRAY”.

9.10.4.3 Setup of a conveyor

In the setup screen of the conveyor in which a clearance check is to be used, check the checkbox of [Used] for [Clearance Check].
There are no other special notes concerning the setup of a conveyor for a clearance check. Refer to Section 6.4, “SETUP OF A CONVEYOR”.



9.10.4.4 Setup of a sensor

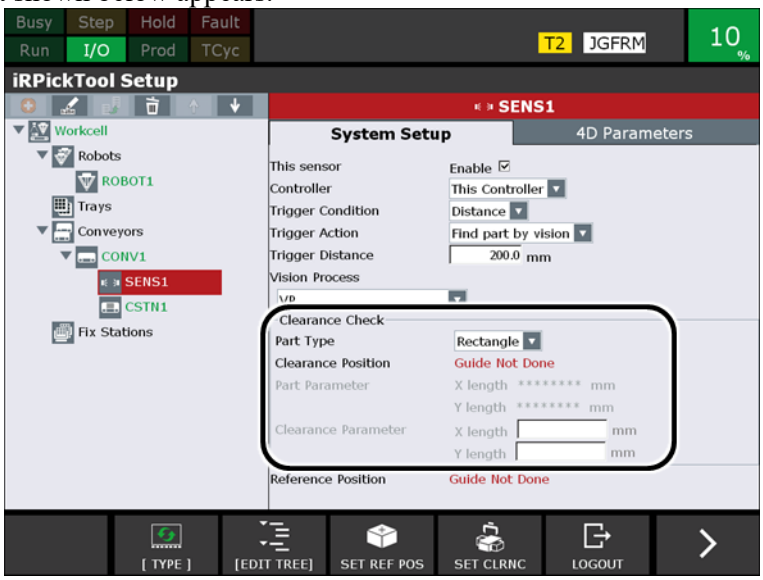
When clearance check is used in the parent conveyor, the clearance check setting item appears on the sensor setup screen.

[Part Type]

Select the shape of the part for which a clearance check is set. [Circle (without Phase)] or [Rectangle] can be selected. The setup screen switches as follows according to Part Type.

When [Part Type] is [Rectangle]

A screen such as that shown below appears.



9

[Clearance Position]
When the procedure provided in Subsection 9.10.4.5, “Clearance guide” is complete, [Guide Complete] appears in green.

[Part Parameter]
Set the part length in the X direction in [X length]. Set the part length in the Y direction in [Y length]. This setting is configured in Subsection 9.10.4.5, “Clearance guide”.

[Clearance Parameter]
Set the clearance length of the part in the X direction in [X length]. Set the clearance length of the part in the Y direction in [Y length]. This setting is configured in Subsection 9.10.4.5, “Clearance guide”.

For example, the figure below illustrates a case in which [X length] and [Y length] of [Part parameter] are 30 mm and 50 mm, respectively, and [X length] and [Y length] of [Clearance Parameter] are 15 mm and −10 mm, respectively.

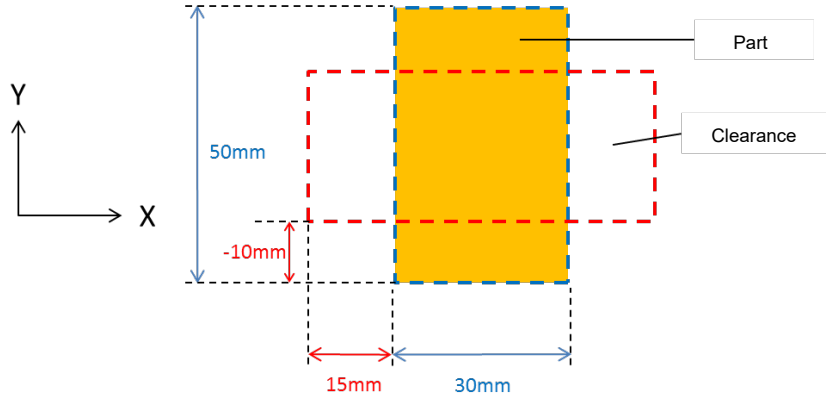


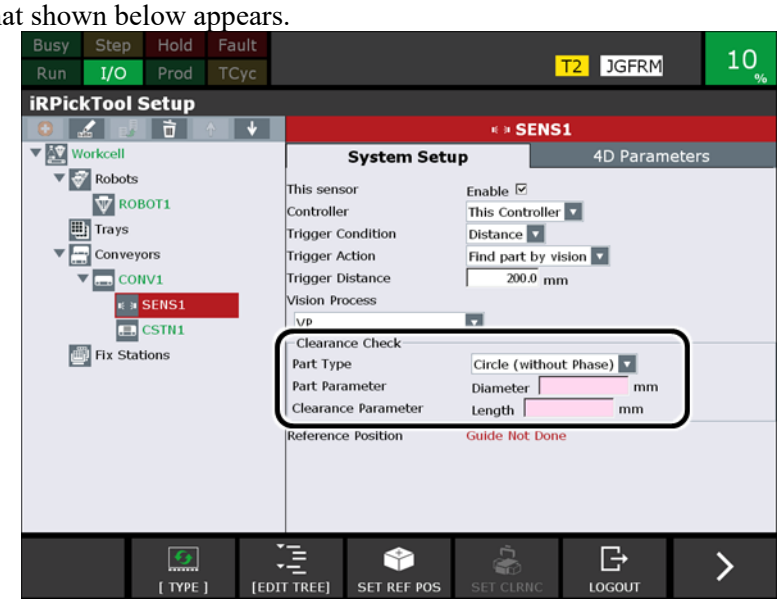
Fig. 9.10.4.4 (a) Parameter setting example (when [Part Type] is [Rectangle])

[Set Clearance]
This item is located on function key F4. Pressing this key starts the procedure in Subsection 9.10.4.5, “Clearance guide”.

> ⚠ **CAUTION**
> The procedure in Subsection 9.10.4.5, “Clearance guide” is performed with the touchup pin attached. Perform this procedure before reference position setup, which is performed with the hand attached.

When [Part Type] is [Circle (without Phase)]

A screen such as that shown below appears.



[Part Parameter]

Set the diameter of the circle in [Diameter].

[Clearance Parameter]

Set the clearance length in [Length].

The figure below illustrates a case in which [Part Parameter] is 30 mm and [Clearance Parameter] is 10 mm, for example.

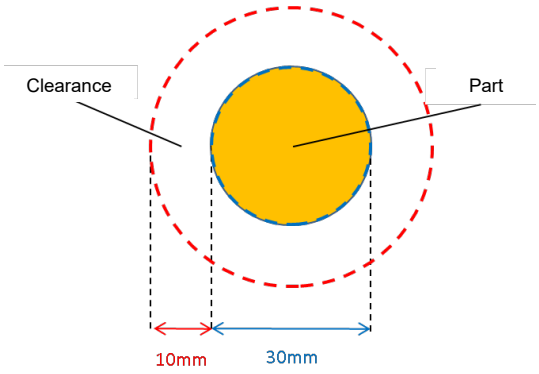
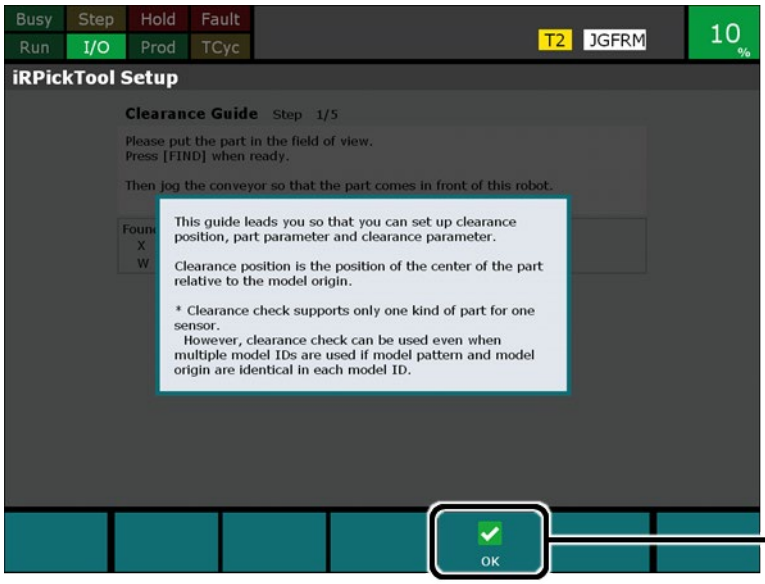


Fig. 9.10.4.4(b) Parameter setting example (When [Part Parameter] is [Circle (without Phase)])

9.10.4.5 Clearance guide

Configure this setting only when [Rectangle] is selected for [Part Type] in Subsection 9.10.4.4, “Setup of a sensor” .

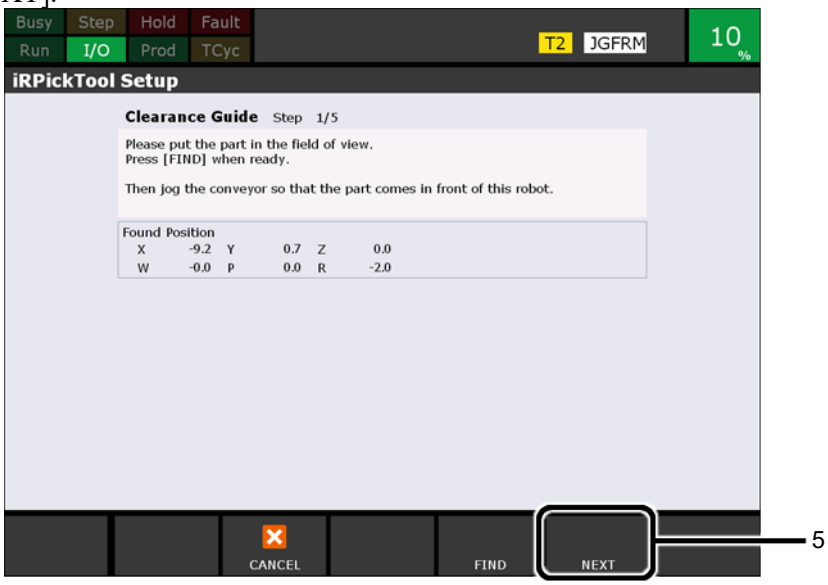
1 Stop the conveyor.
2 Press F4 [Set CLRNC].
The following screen appears.



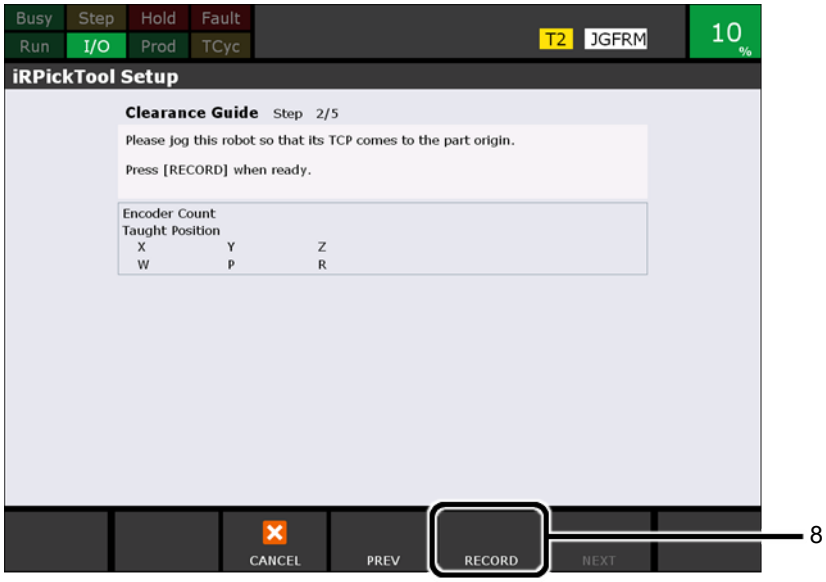
3 Confirm the content of the message displayed, and press F4 [OK].
The following screen appears.

4 Place the part inside the camera's field of view, and press F4 [FIND].
When the part is detected successfully, the detection position is displayed and F5 [NEXT] is enabled.

5 Press F5 [NEXT].



The following screen appears.



6 Move the conveyor until the part is located in front of the robot.
7 Move the robot by jog operation, and touch up the origin of the part.

NOTE

Touch up the part at three points: the origin point, X direction point, and Y direction point.
The points to be touched up differ depending on the Z-axis direction of the tracking frame, as described in the following.

- When the +Z axis of the tracking frame is in the upward direction to the conveyor surface,
  The Y direction point should be in the direction in which the X direction point is rotated counter-clockwise by 90°.

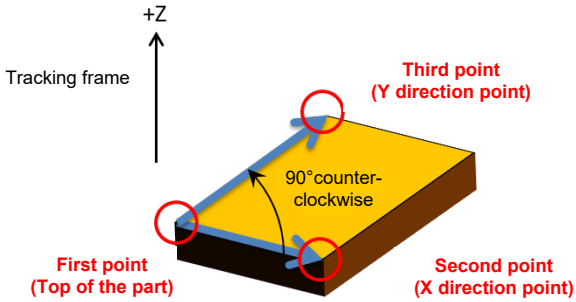


Fig. 9.10.4.5 (a) When the +Z axis is in an upward direction

- When the +Z axis of the tracking frame is in the downward direction the conveyor surface
  The Y direction point should be in the direction in which the X direction point is rotated clockwise by 90°.

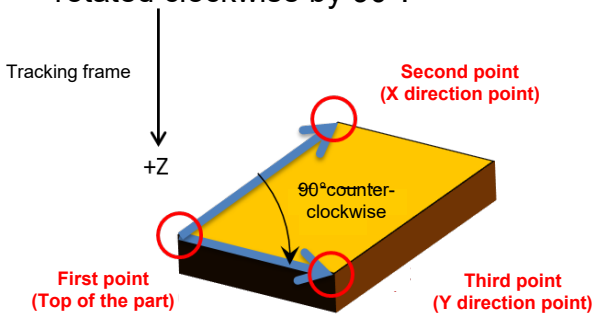
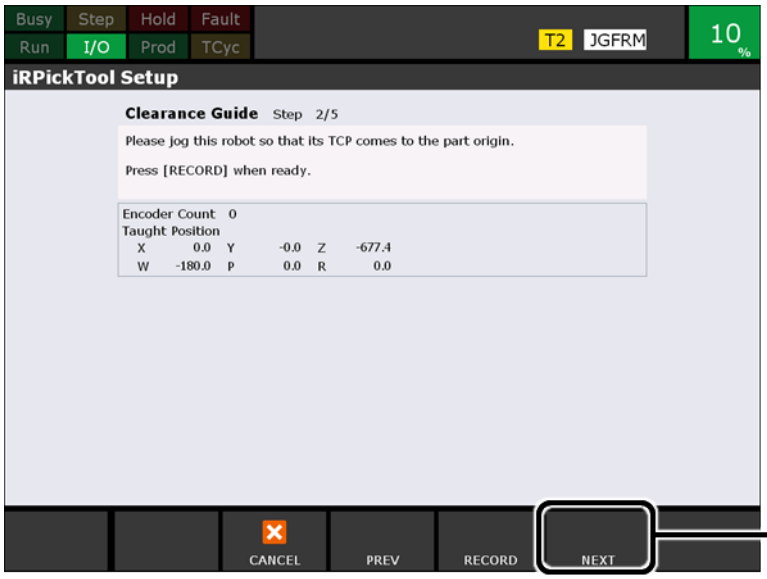


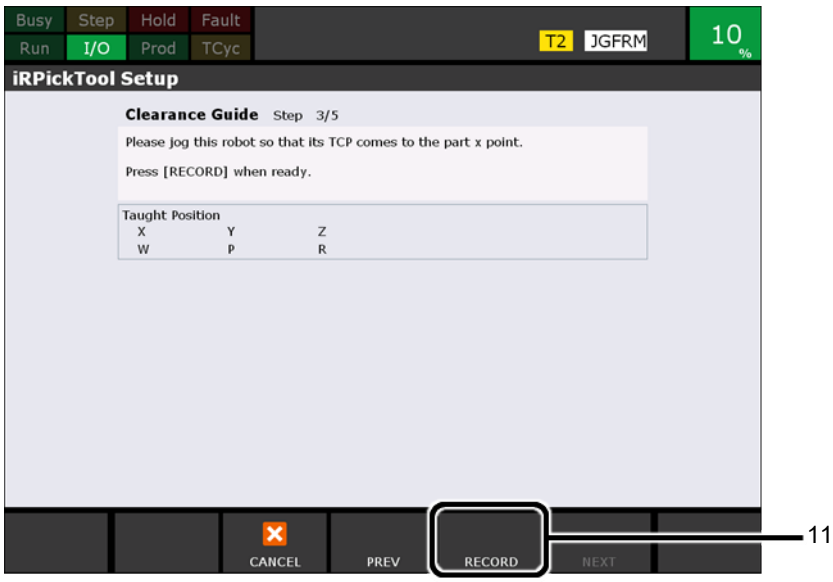
Fig. 9.10.4.5 (b) When the +Z axis of the tracking frame is in the downward direction

8 Press F4 [RECORD].
The encoder value obtained at the time of touch-up and the touch-up position are displayed, and F5 [NEXT] is enabled.

9 Press F5 [NEXT].



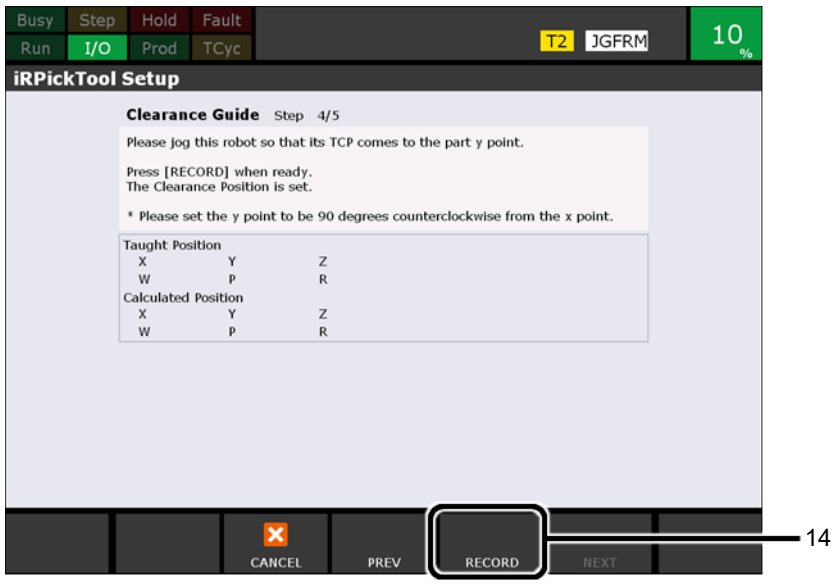
The following screen appears.



10 Move the robot by jog operation, and touch up the X direction of the part.
11 Press F4 [RECORD].
The touch-up position is displayed, and F5 [NEXT] is enabled.

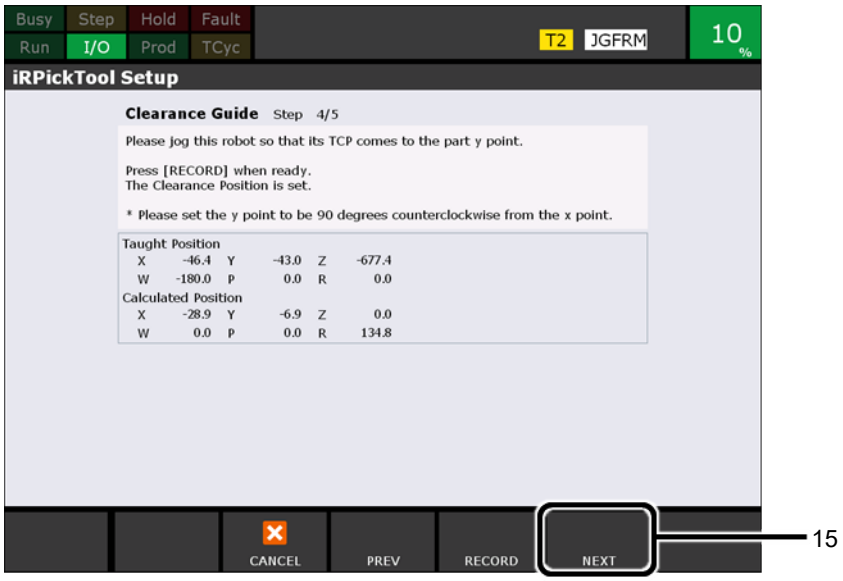
12 Press F5 [NEXT].

The following screen appears.



13 Move the robot by jog operation, and touch up the Y direction point of the part.
14 Press F4 [RECORD].
The touch-up position and calculated position are displayed, and F5 [NEXT] is enabled.

15 Press F5 [NEXT].

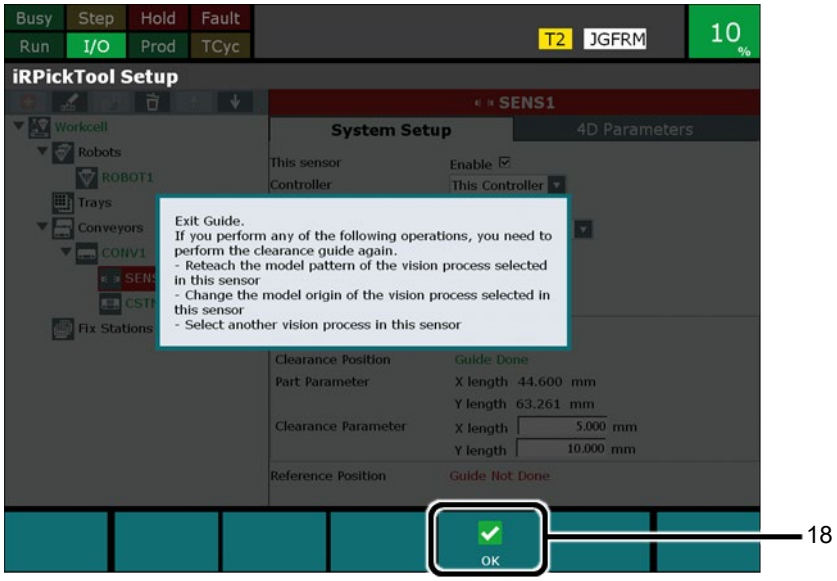


The following screen appears.

16 Enter [X length] and [Y length] of [Clearance Parameter].
F5 [FINISH] is enabled.

17 Press F5 [FINISH].

The following screen appears.



18 Confirm the content of the displayed message, and press F4 [OK].
The screen returns to the original setup screen.
[Guide Complete] is displayed in green next to [Clearance Position], and the values set in the guide are entered in [Part Parameter] and [X length] and [Y length] of [Clearance Parameter].

⚠ CAUTION
When any of the following procedure is performed in a vision process used in the sensor for which a clearance check is set, it is necessary to perform clearance guiding again.
- Re-teaching the model
- Changing the model origin

9.10.4.6 Setup of a conveyor station

In the case of a line conveyor

[Send Position] is displayed as shown in the figure below.

The send position refers to the position on this conveyor station where the part data is sent to the downstream station, and is expressed as a position on the tracking frame.

If setup of the clearance check in Subsection 9.10.4.4, "Setup of a sensor", setup of the tracking area (upstream, downstream), and setup of the discard line have been completed, values are automatically set in [Send Position] for all the conveyor stations corresponding to the sensors for which clearance check is set.

There are no other special notes concerning the setup of a conveyor station for a clearance check. Refer to Section 6.6, "SETUP OF A CONVEYOR STATION".

In the case of a circular conveyor

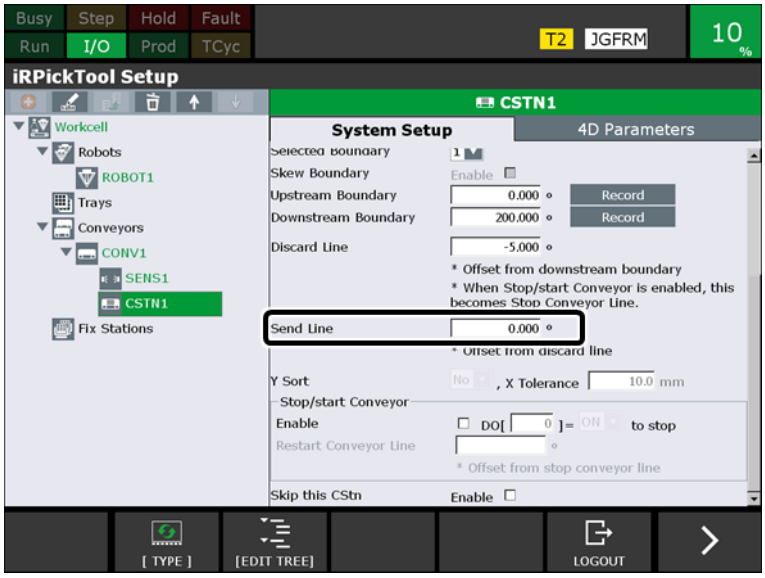
[Send Line] is displayed as shown in the figure below.

The send line refers to the position on this conveyor station, where the part data is sent to the downstream station, and is expressed as an offset from the discard line.

The send line is entered by the user.

There are no special notes concerning the setup of a circular conveyor for a clearance check. Refer to Section 6.6, "SETUP OF A CONVEYOR STATION".

> ⚠ **CAUTION**
> The send line position changes depending on whether parts flow on the inside or outside of the circular conveyor.
> For example, when parts flow near the center of the conveyor, it is necessary to set a large value for the send line.



9.10.4.7 Setup of a fixed station

There are no special notes concerning the setup of a fixed station for a clearance check. Refer to Section 6.7, “SETUP OF A FIXED STATION”.

10 TROUBLESHOOTING

This chapter describes common issues in a tracking system using *i*RPickTool and their countermeasures.

10.1 WHEN THE ROBOT DOES NOT PICK UP PARTS

Cause1: The conveyer station is disabled.
Remedy1: If the conveyer station has “Skip this CStn” enabled,part queue information will not be sent to this conveyor station’s robot. Disable “Skip this CStn”.

Cause2: The load balance is set improperly.
Remedy2: Confirm that the pickup ratio or load balance quota of the robot is not set to zero in the Conveyor menu.
If the load balance quota for putting down parts is incorrect, pickup motion may not operate properly. Confirm that the load balance quota for picking up and putting down is set properly.

Cause3: The model ID is incorrect.
Remedy3: Confirm that the model ID of the part detected by the vision process matches the model ID specified by the load balance in the conveyer setting.
If using a tray, confirm that the model ID of a cell matches the model ID of the load balance.
When a model ID is specified as an argument in PKCSGETQ, PKFSGETQ, or PKFSGETBUF call, confirm that the model ID specified in the call matches the model ID set in the vision process or cell.

Cause4: Parts are not detected by the vision process.
Remedy4: Confirm that parts are detected correctly using the *i*RVision runtime image.

Cause5: The sensor task is not started.
Remedy5: Confirm that PKWCSTART.PC (or PKSNSTART.PC) has been executed. You may also confirm the execution status by checking the sensor or conveyor station statuses on the production status screen. For details, refer to Section 9.2 “CHECKING THE PRODUCTION STATUS.”

10.2 WHEN PARTS ARE MISSED FREQUENTLY

Cause1: Processing by the robot is too slow.
Remedy1: Disable the hot start when it is enabled.

> ⚠ **CAUTION**
> The *i*RPickTool tracking does not support the hot start.

Disable Automatic Backup.In the FILE menu press F1[TYPE] key and select “Auto Backup”. Press F5 key “DISABLE” in the “Automatic Backup” menu.

Cause2: The motion of the robot is too slow.
Remedy2: If the motion of the robot is too slow, the robot will not be able to pick parts on a moving conveyor . Adjust the speed in the motion program. When the motion speed is set to “max_speed” in the motion program, the maximum speed that can be set for the robot is automatically specified. Confirm that an appropriate override is also selected.

Cause3: No part is detected by the vision process.
Remedy3: If parameters of the vision process are set inappropriately, no parts may be detected. Adjust the parameters of the GPM locator tool or Blod tool so that parts are detected correctly.

Cause4: Vision detection is too slow.
Remedy4: If vision processing is too slow relative to the conveyor speed, the next vision process cannot be executed at the requested timing and parts may pass by the field of view without being detected. Confirm that vision execution is not slow (the overrun count is less than 1) by using the sensor menu of the *i*RPickTool production status screen and, if it is too slow, decrease the speed of the conveyor or enlarge the field of view.
When multiple vision systems are connected to one robot controller, distribute the sensors among multiple robot controllers.

Cause5: Vision detection time is too long.
Remedy5: Because the priority of vision detection is lower than that of a TP program, if the execution of a TP program is busy, the vision detection cannot run as often. As a result, the detection time becomes longer. Particularly, if there are no pickable parts on the conveyer and a program including the zero distance motion to the waiting position is executed repeatedly, the TP program takes a large portion of CPU time. In such a case, temporarily stop the TP program so that vision detection can be made by setting the third argument of PKCSGETQUE to a positive value as described in Subsection 6.9.2.2, “Pick program”. When the third argument of PKCSGETQUE is set to a positive value, PKCSGETQUE works as follows, as described in Subsection “6.8.5.3 PKCSGETQUE.PC”.

- If there are pickable parts, it does not wait.
- If there is no pickable parts, it waits for specified time. However if any pickable part comes in the conveyer station before the specified time passes, it gets the information of the part and finish waiting.

In the case of using a WAIT instruction, even if a pickable part comes in the conveyer station, the program waits for the specified time. For the program to wait for the necessary time, set the third argument of PKCSGETQ to the waiting time. Usually, 100 is enough.
If the third argument of PKCSGETQ is set to 0, when there are no pickable parts, the loop to call PKCSGETQ (line 5 to line 10 of Subsection 6.9.2.2 “Pick program”) is executed repeatedly, thereby extending the time before completion of vision detection.

Cause6: The setting of the discard line is inappropriate.
Remedy6: If the position of the discard line is too upstream, a pickable part is determined to be not pickable and is missed. Adjust the appropriate position of the discard line by actually operating the system.

Cause7: The tracking area is set inappropriately.
Remedy7: Check that the set tracking area is appropriate.

Cause8: The model ID is incorrect.
Remedy8: Confirm that the model ID specified by the vision process matches [Model IDs to handle] in the load balance box of the conveyer setting. If the model IDs are different, the parts with the model ID are not picked up by any robots.

10.3 WHEN THE SAME PART IS PICKED UP TWICE

Cause1: Camera calibration is inappropriate.
Remedy1: Confirm that the pointer TCP used for camera calibration is correct. Confirm that the grid spacing and focal distance of calibration data are correct.

Cause2: The [Duplicate Tolerance] setting is inappropriate.
Remedy2: Confirm that this setting is not too small.

Cause3: A phantom part is falsely detected.
Remedy3: Confirm that a wrong part is not detected in its vicinity using Vision run-time window.

10.4 INACCURACY WHEN A PART ROTATES

Cause: The tracking frame is not set correctly.
Remedy: See Subsection 6.12.2, “Adjustment of the Tracking Frame Error”.
Or, the camera calibration is not correct.

10.5 INACCURACY WHEN A PART IS NEAR THE CONVEYOR EDGES

Cause1: The calibration grid used for camera calibration may be too small to calculate the lens distortion correctly.
Remedy1: If calibration is performed with the calibration grid captured only by part of camera's field of view, the part position detected in the section where the grid pattern is not captured will not be calculated correctly. Make the calibration grid larger so that it is fully covers the field of view, and calibrate it again.

Cause2: If the [Part Z Height] setting specified by the vision process is incorrect, this symptom may appear.
Remedy2: Set [Part Z Height] to the height of the feature to be detected in the tracking frame properly.

10.6 WHEN ROBOT MOTION IS NOT SMOOTH

Cause: The value of the average speed (updates) is small.
Remedy: If the conveyor in use does not move smoothly, the robot also may not move smoothly because it follows the motion of the conveyor. In such a case, increase the [5. Average (updates)] on the encoder schedule menu of the teach pendant. This value is the accumulated number of instantaneous speeds used to calculate the average speed of the encoder. The default value is 1. Set the value to 10 initially and check the operation of the robot.

It could also be due to high CPU load – for example, you are running two vision processes and motion in one controller.

10.7 THE STOP CONVEYOR DOES NOT WORK

Cause: The setup of DO values may be inconsistent.
Remedy: Confirm that, in multiple conveyer stations defined in the same robot controller, the same DO number is not specified for the stopped conveyor, and polarities of DO are not set opposite to each other (in some conveyer stations the conveyor stops when DO is ON, but in others the conveyor stops when DO is OFF).

Also, make sure that the discard boundary is not too small for the part to stop before it crosses the downstream boundary.

10.8 SLOW VISION PROCESSING DUE TO DELAYED IMAGE SNAP TIMING

Cause: Vision processing may be too slow because the history of images is getting stored in the memory device.
Remedy: Disable image loggin by unchecking [Enable Logging] in the [iRVision Configuration] screen of *i*RVision
Also, ensure “Do Not Log” is selected as Image Logging Mode in the vision process setup .

10.9 THE OVERRUN COUNT CONTINUES TO INCREASE

Cause: If the overrun count continues to increase in the sensor menu of the production status sensor screen, vision processing is too slow because the vision processing time is longer as compared with the interval at which a vision start trigger occurs. However, since vision performs the next processing after the previous detection is completed, some parts may pass by the field of view of the camera. In such a case, the setting needs to be changed.

Remedy: When detection is made each time a certain distance of movement is performed, open the sensor setting screen and set [Trigger Distance (mm)] to a value as large as possible and check if this symptom persists.
When detection is made each time the DI signal is input, adjust the interval at which part are fed to extend the interval at which the DI signal is input as much as possible.
The vision processing time may be reduced by changing vision settings etc. Possible measures are described below.

- If the vision log is enabled, disable it.
- If the vision runtime is displayed, uncheck [Enable Logging] on the [iRVision Configuration] screen of *i*RVision and select [Don't Display] in [Runtime Image] on the teach screen of the vision process so that images are not displayed.
- Adjust the location parameters of GPM Locator tool. In particular, if [Aspect] is checked, uncheck it. In visual tracking in which the detected surface of a part is not tilted, [Aspect] is not used. If the model pattern printed on the surface of the pouched part looks different for each part, adjust [Elasticity] instead of [Aspect].

Reducing processes other vision could also reduce the vision processing time. For example, if multitasking is used, try to stop using it.

10.10 WHEN SETTINGS CANNOT BE CHANGED DURING PRODUCTION

Cause: There may be an item whose settings cannot be changed during production.

Remedy: Due to safety concerns, *i*RPickTool does not allow some *i*RPickTool setup items to be changed during production. There are three types of setup items.

1. Setup items that can be changed. For example, you can change the load-balance quota even during production.
2. Setup items that can be changed but requires confirmation. For example, if you want to change the “upstream boundary” of a conveyor station, a confirmation prompt will display if you want to make this change.
3. Setup items that cannot be changed during production. For example, you will not be allowed to change the “Trigger action” of a sensor during a production run. If you try to change the trigger action of a sensor, a warning: “IRPK-075 This operation is allowed only when the program is aborted” will appear.

Regarding item 3, “3. Setup items that cannot be changed” , these items cannot be changed during production. To change these items, you must stop production and abort all robot programs, and then change the settings.

Note: All robots in the workcell should be in an aborted state. If any robot is paused or running, workcell will assume the system is “in production” and you cannot make the changes.

APPENDIX

A SYSTEM VARIABLE FILES

System variable file for *i*RPickTool is shown below.

SYSPKCFG.SV
In these files, all *i*RPickTool data are saved

LNTKBCK.SV
In this file, all line tracking data are saved.

B SETTING A TEACHING PC

In *i*RPickTool, all settings can be made with the touch-panel-enabled *i*Pendant basically, but a personal computer can also be connected to make settings. This chapter describes preparation for making settings with a personal computer.

NOTE
When making settings with the touch-panel-enabled *i*Pendant, it is not necessary to set a teaching PC.

A teaching PC is required only when settings of *i*RPickTool are made and the PC can be removed when the system is operating.

NOTE
A teaching PC is connected to the local network used for the robot ring.

B.1 OPERATING SYSTEM AND BROWSER

Refer to the following table for the tested PC and browser.

OS	Windows 7 Professional (32bit) Windows 7 Professional (64bit)
Browser	Internet Explorer 9 (32bit) Internet Explorer 10 Internet Explorer 11

⚠ CAUTION
1 The tested languages of Windows are Japanese and US English.
2 All Windows versions assume that the latest Service Pack is installed.
3 When you log in to your PC as a user without the Administrator password, the PC might not normally communicate with the robot. Log in to your PC as a user with the Administrator password.

B.2 COMMUNICATION CABLE

The communication cable is used to connect the robot controller to a teaching PC for *i*RPickTool. Select a 10BASE-T or 100BASE-T cable that meets the following specifications.

Cable	Twisted part
Shield	Shielded

B.3 CONNECTION OF THE COMMUNICATION CABLE

Connect the robot controller to a teach PC via the Ethernet cable. One end of the cable is connected to the Ethernet connector on the front face of the main board of the robot controller. The other end is connected to the connector of the PC marked with 品.

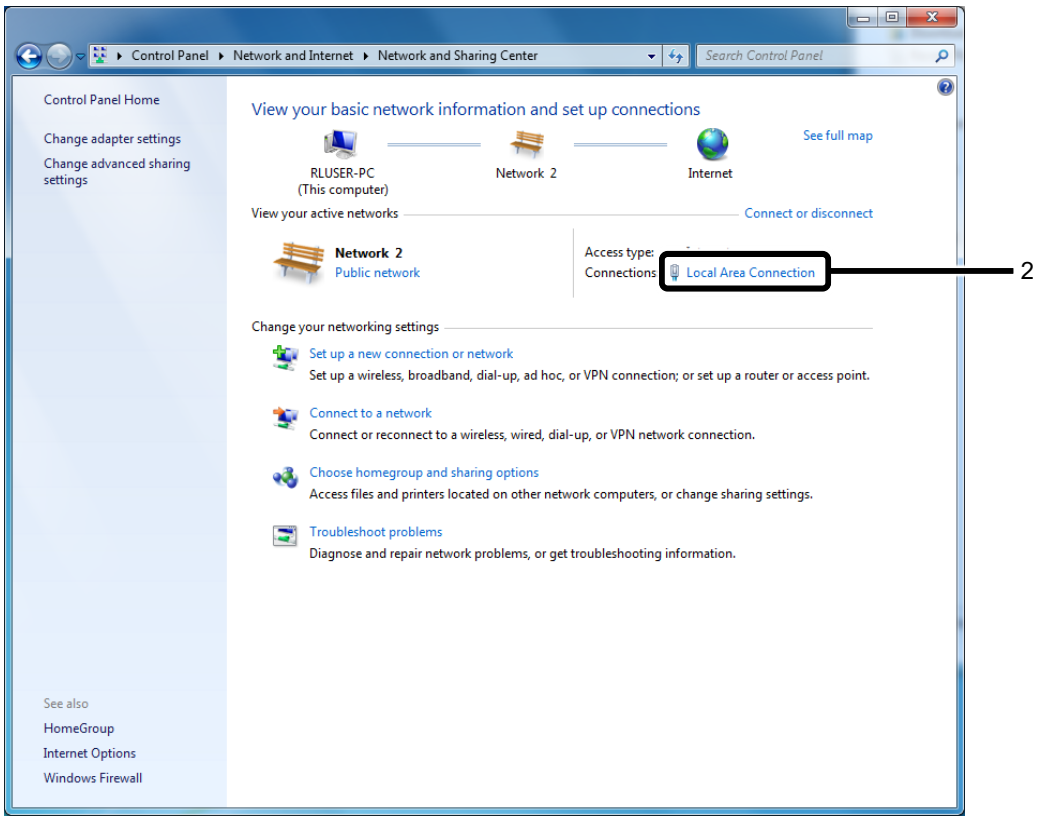
B.4 SETTING AN IP ADDRESS FOR THE PC

Set an IP address for the teach PC. When three robot controllers and the teach PC are connected and not connected to another network, the IP addresses as shown below can be set, for example.

Robot controller 1	172.16.0.1
Robot controller 2	172.16.0.2
Robot controller 3	172.16.0.3
PC	172.16.0.4
Gateway	172.16.0.5
Subnet mask	255.255.0.0

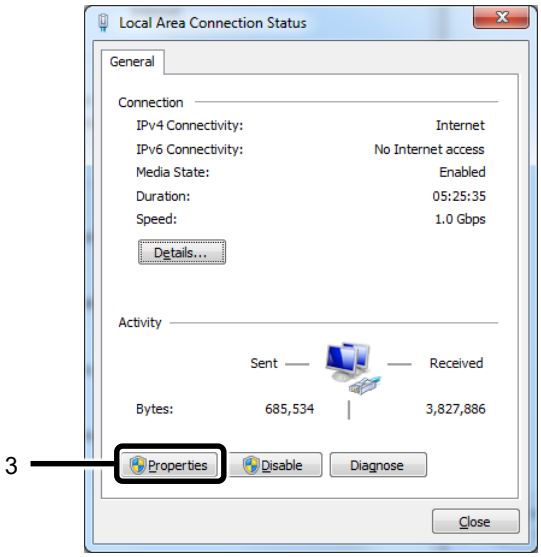
Set up the IP address using the following procedure.

1 From [Start] → [Control Panel] of a PC, open [Network and Sharing Center].
2 Click [Local Area Connections] under [View your active networks].

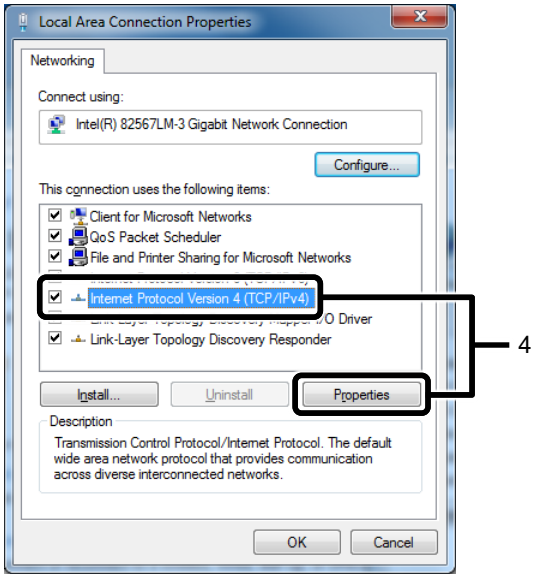


The [Local Area Connection Status] screen will appear.

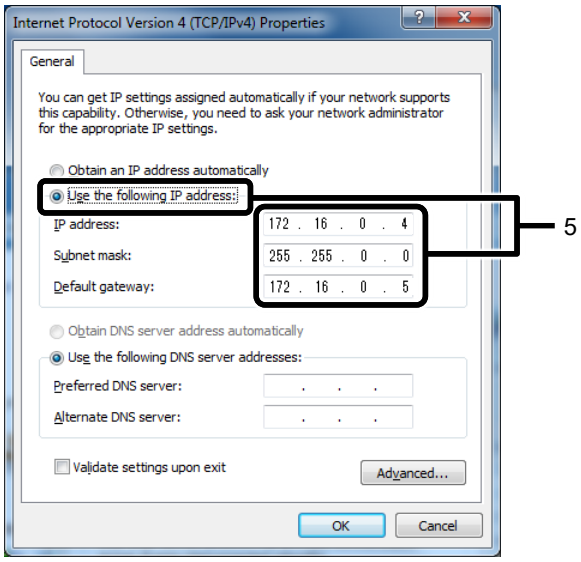
3 Click the [Properties] button.



4 Select [Internet Protocol Version 4 (TCP/IPv4)] and click the [Properties] button.



5 Check [Use the following IP address], enter values in [IP address], [Subnet mask], and [Default gateway], and click the [OK] button to close the window.

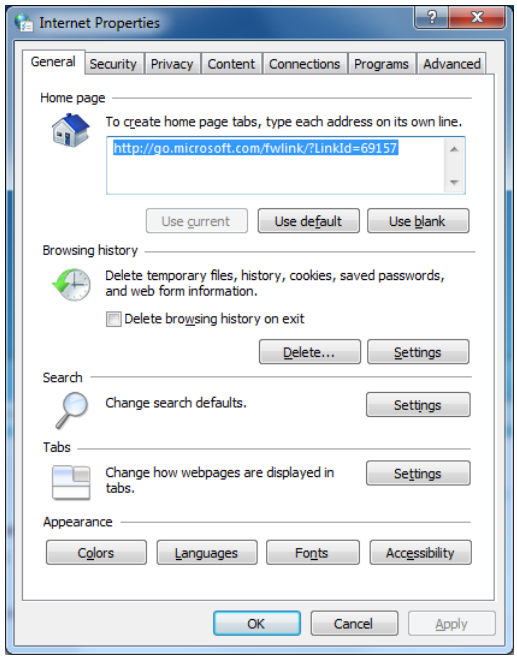


6 Click the [OK] button and close the screen.

B.5 CHANGING SETTINGS OF INTERNET EXPLORER

The following procedure changes the settings of Internet Explorer to prevent Internet Explorer from blocking the communication with the robot controller. The procedure is described with reference to an example of a screen of Windows 7.

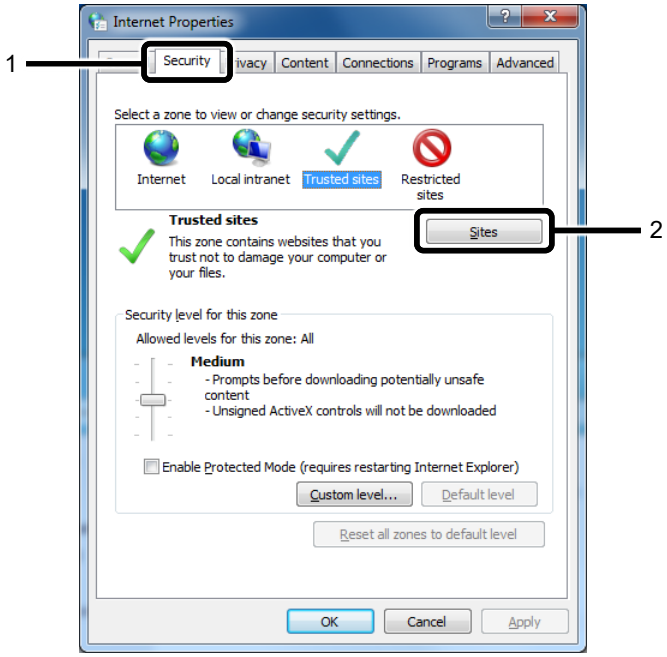
1 From [Start] → [Control Panel] of a PC, open [Internet Options].
The [Internet Properties] screen will appear.



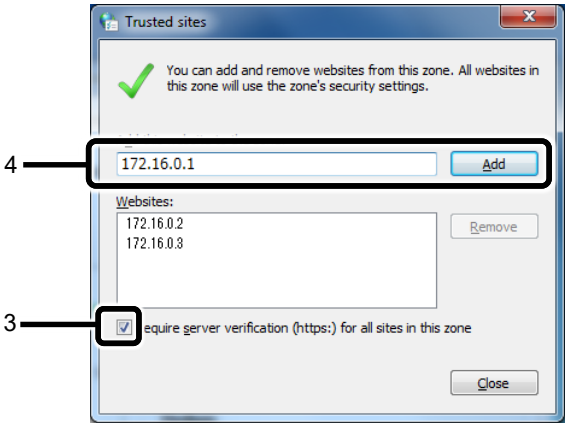
Trusted Site

The following procedure registers the robot controller as a trusted site.

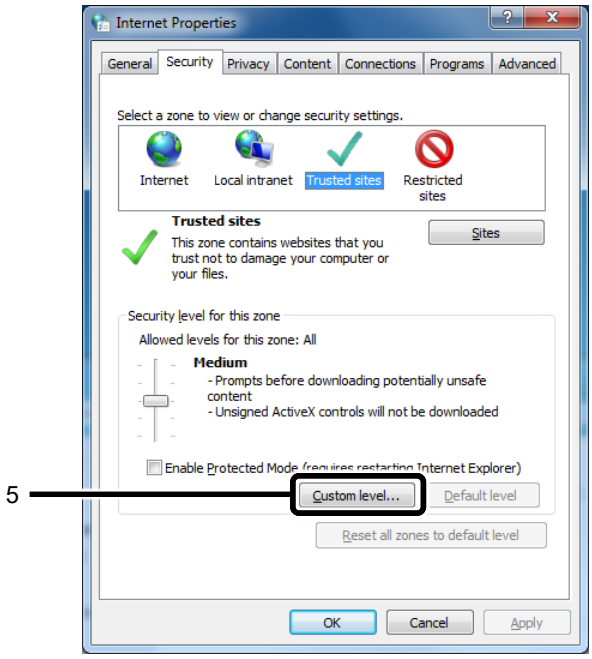
1 Select the [Security] tab.
2 Select [Trusted Site] as the zone and click the [Sites] button.



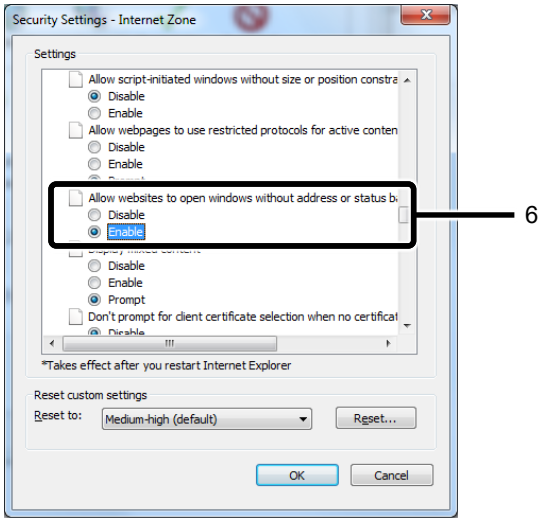
The [Trusted Sites] screen will appear.
3 Uncheck [Require server verification (https:) for all the sites in this zone].
4 Enter the IP address of the robot controller in the [Add this Web site to the zone] text box and click the [Add] button.



5 Move to the [Security] tab. Click the [Custom level] button. Enable “Allow websites to open windows without address or status bars” in “Miscellaneous”.



6 Enable [Allow websites to open windows without address or status bars] in [Miscellaneous].

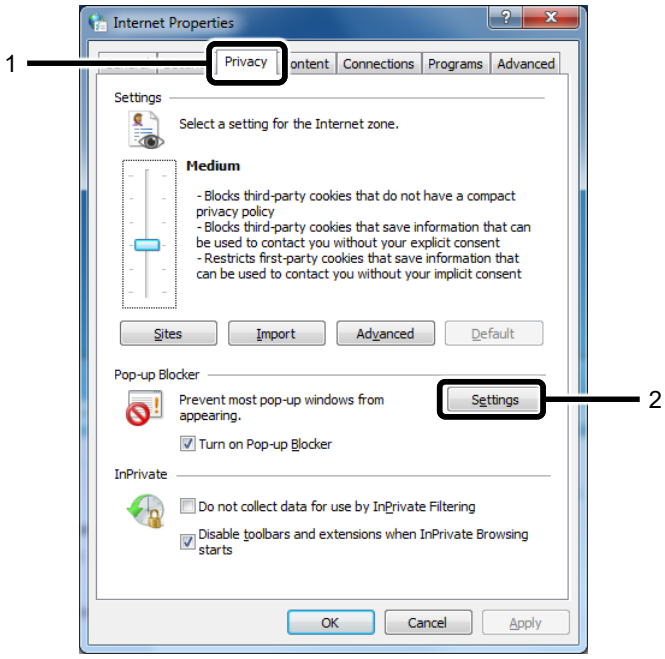


7 Press the [Close] button to close the dialog box.
8 Re–open the Internet Explorer.

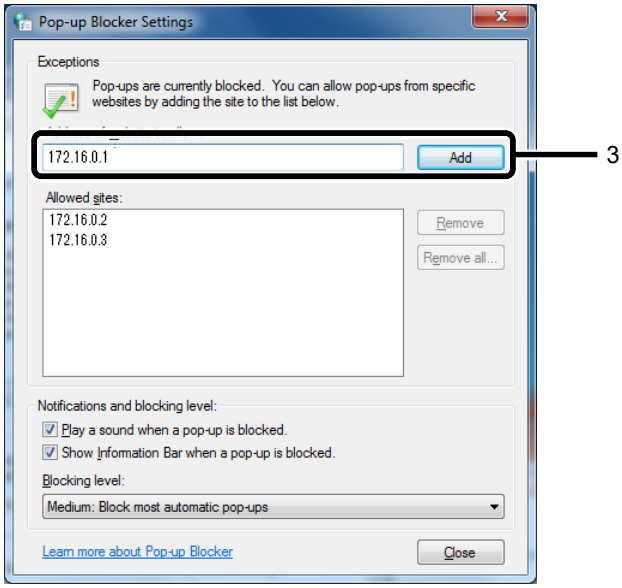
Preventing Popups from being Blocked

The following procedure prevents popups from being blocked during access to the homepage in the robot controller.

1 On the [Internet Properties] screen, select the [Privacy] tab.
2 Click the [Settings] button under [Pop-up Blocker].



3 Enter the IP address of the robot controller in the [Address of Web site to allow] text box and click the [Add] button.

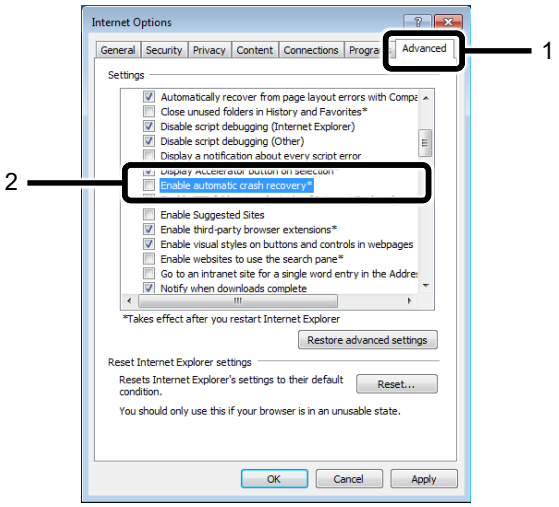


4 Press the [Close] button to close the dialog box.

Disable Automatic Crash Recovery

Automatic Crash Recovery should be disabled in Internet Explorer. This setting is necessary to install Vision Controls properly.

1 Select the [Advanced] tab, and scroll to the Browsing section.
2 If the [Enable automatic crash recovery] box is checked, uncheck it.

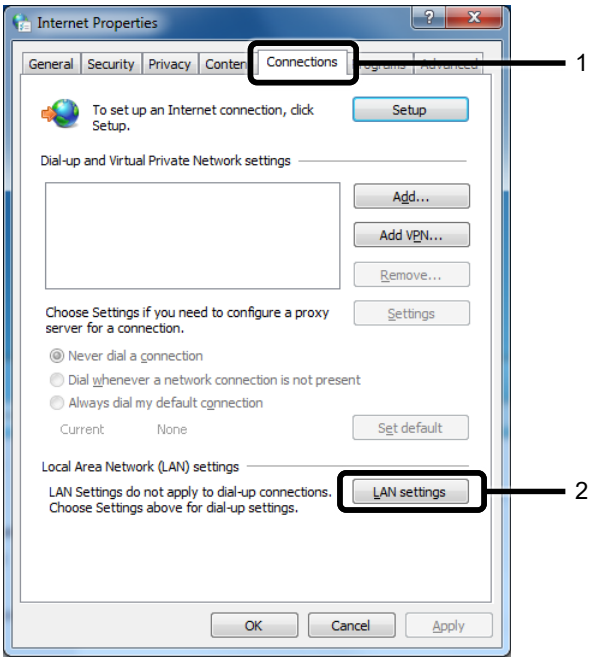


3 Click [OK]. When you changed the item, restart your PC.

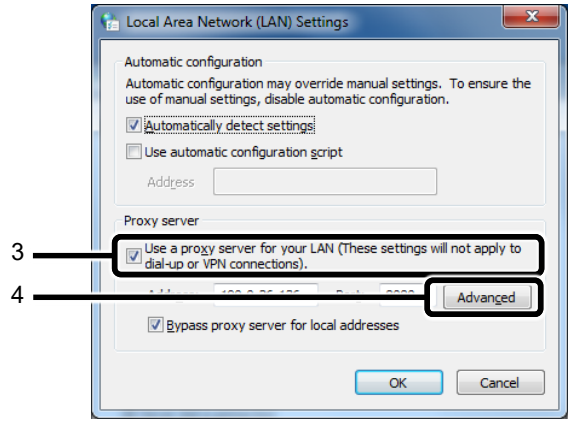
Proxy Server

The following procedure registers the IP address of the robot controller as a local address to prevent the proxy server from being used for communication with the robot controller.

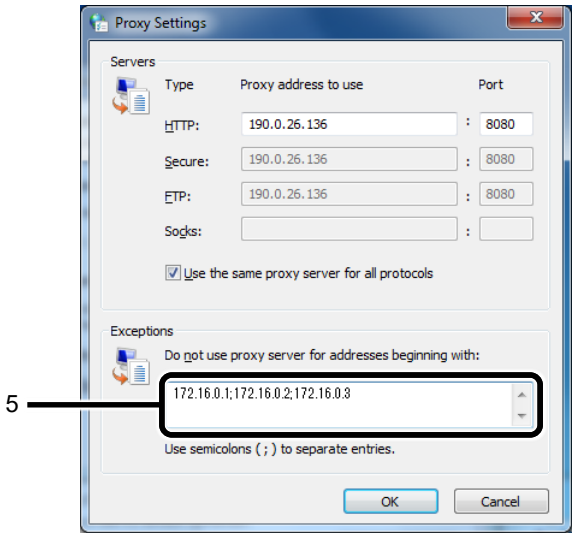
1 Select the [Connections] tab.
2 Click the [LAN Settings] button.



3 When [Use a proxy server for your LAN] is not checked, go to step 7.
When [Use a proxy server for your LAN] is checked, execute steps 4 to 6.
4 Click the [Advanced…] button under [Proxy server].



5 Enter the IP address of the robot controller in the [Exceptions] text box.

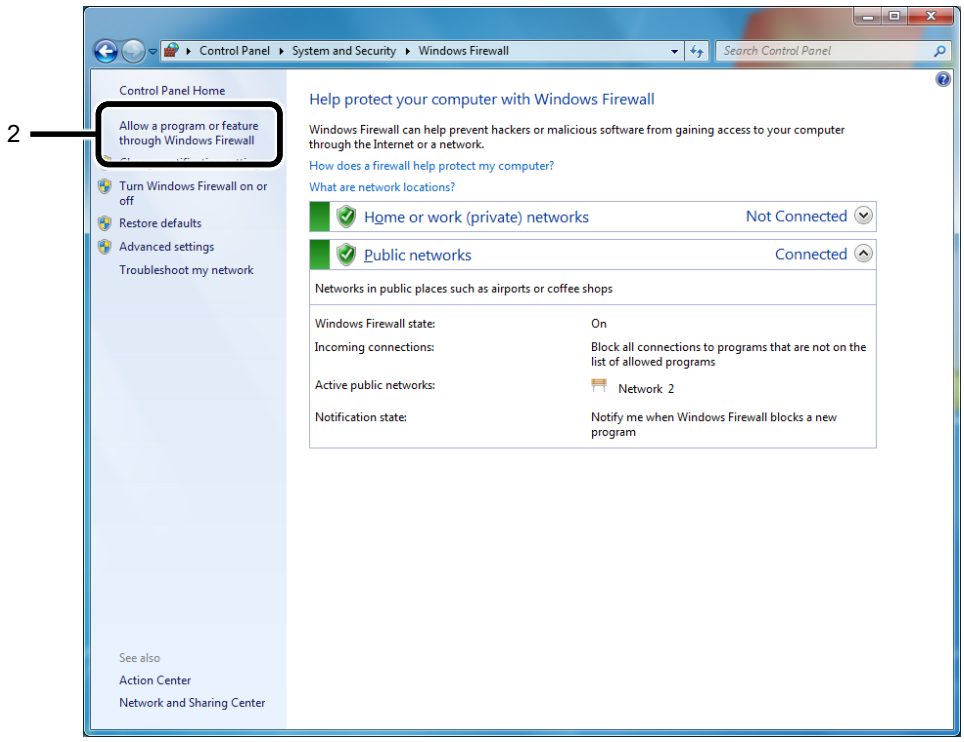


6 Click the [OK] button to close the screen.

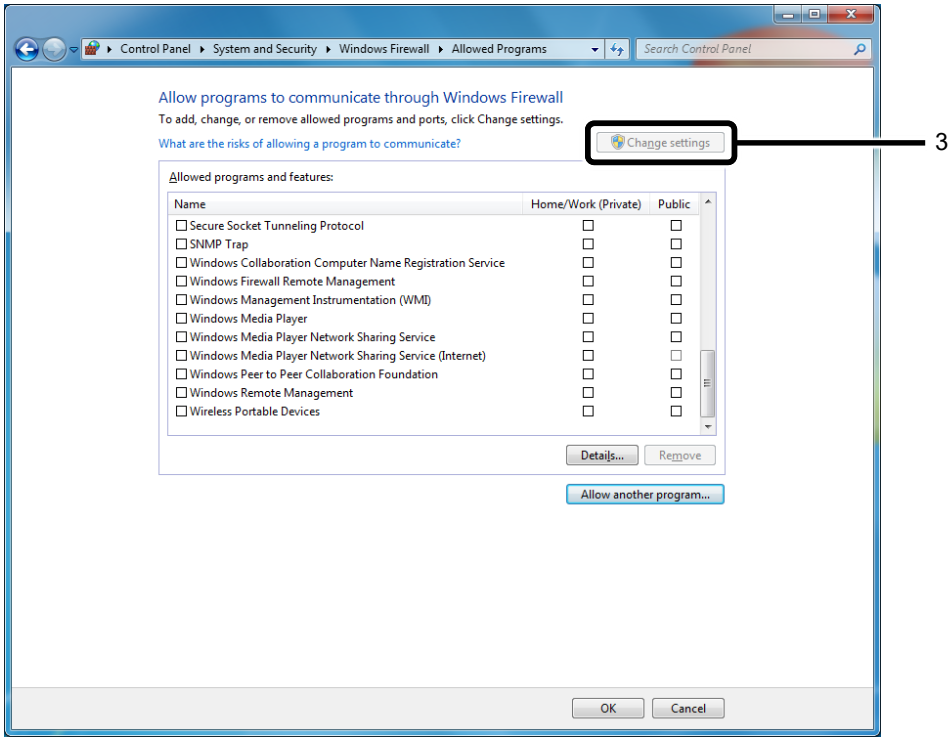
B.6 CHANGING SETTINGS OF WINDOWS FIREWALL

The following procedure sets Windows Firewall to prevent Windows Firewall from blocking communication with the robot controller.

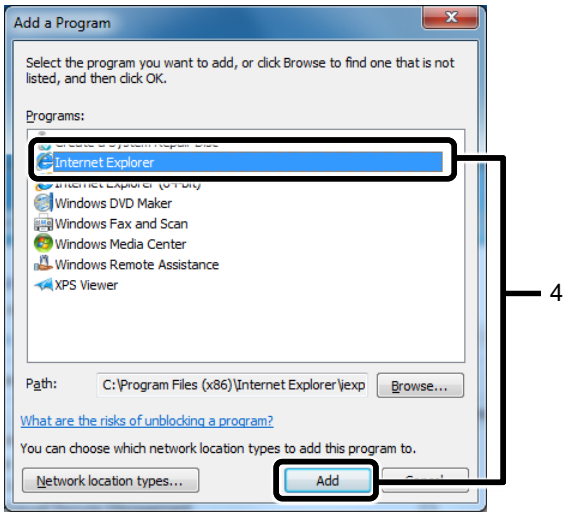
1 In the Control Panel, open [Windows Firewall].
2 Click [Allow a program or feature through Windows Firewall].



3 Click the [Change settings] button.



4 Select [Internet Explorer] from the list and click the [Add] button.



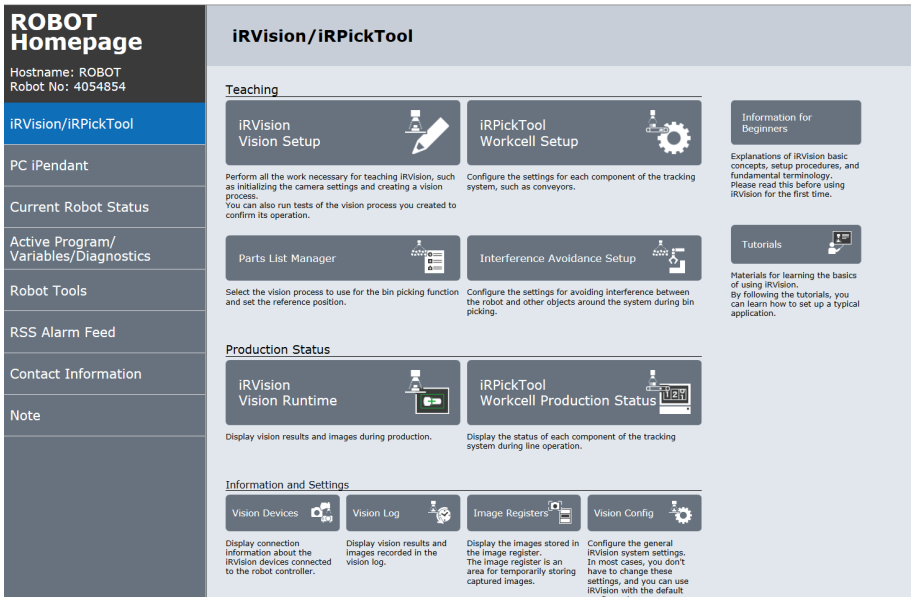
5 Click the [OK] button to close the screen.

NOTE
Communication with the robot controller might be prevented due to causes other than the above, such as Microsoft® Internet Explorer add-ons or security software installed in your PC. If a PC-related problem occurs during setting of *i*RPickTool, refer to Section B.9, “TROUBLESHOOTING” first.

B.7 OPENING *i*RPickTool SCREENS

Before opening the *i*RPickTool screen, follow the procedure below to display the robot homepage.

1 From the [Start] button of Windows, start Internet Explorer.
2 Enter the IP address of the robot controller in the address bar.
The homepage of the robot will be displayed.



All robot controllers have their robot homepages. When the robot controller has *i*RPickTool options, the following links related to *i*RPickTool appear on the robot homepage.

Workcell Setup

Opens the edit screen for setting *i*RPickTool. For details, refer to Chapter 6, "BASIC FUNCTIONS REFERENCE" or Chapter 7, "PLUG&PLAY REFERENCE".

Workcell Product Status

Opens the edit screen for displaying the system production status set with *i*RPickTool. For details, see Section 9.2, "CHECKING THE PRODUCTION STATUS".

B.8 OPENING *i*RVision SCREENS

In visual tracking, camera setting, camera calibration setting, and vision program setting need to be performed on the *i*RVision setup screen. In *i*RPickTool queue managed tracking, these settings are not necessary.

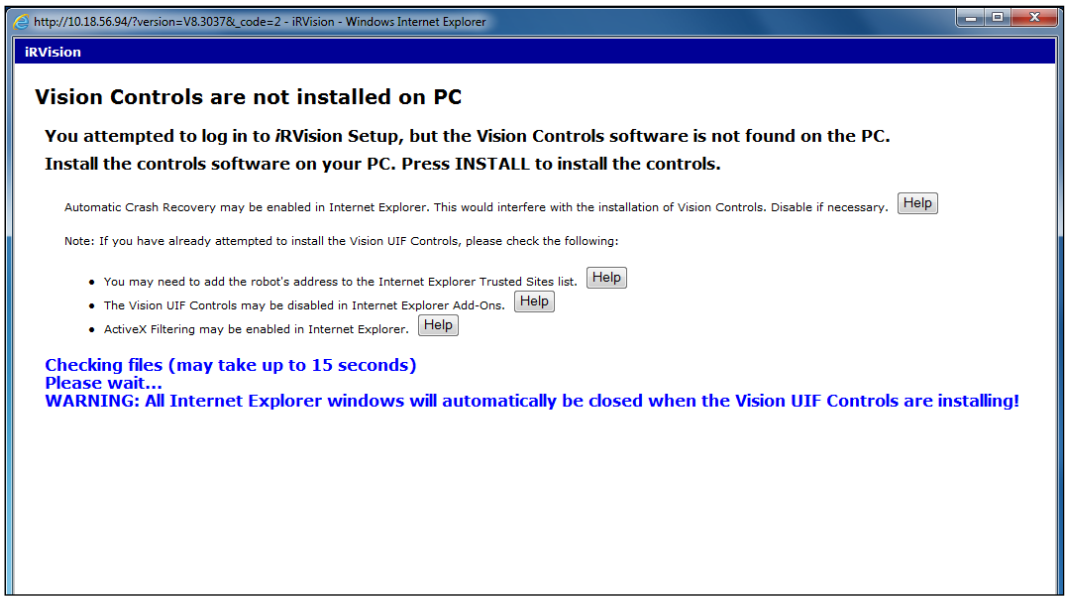
Installing Vision UIF Controls

Vision UIF Controls need to be installed in your PC to open the *i*RVision setup screen. The Vision UIF Controls are installed from the robot controller when the above link is clicked for the first time. The procedure is described below.

1 Click [Vision Setup] of *i*RVision on the screen. If Vision UIF Controls are already installed in the PC used, the Vision Setup Page opens. In this case, the processing ends.
If Vision UIF Controls are not installed in the PC, the following screen appears.

2 Click the [INSTALL] button.

A message like the one shown below to confirm the file will appear.



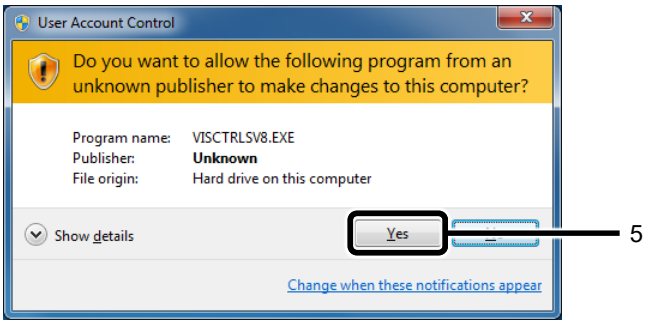
3 Click the [Run] button.

After a while, the following screen appears.

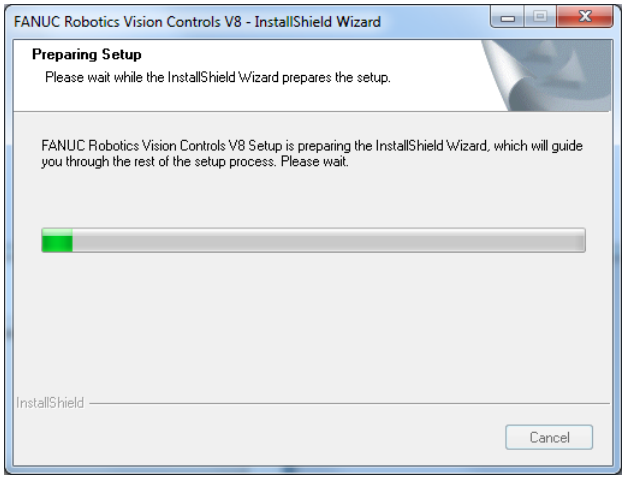
4 Click the [Run] button.

After a while, the following screen appears.

5 Click the [Yes] button.



6 Installation of Vision UIF Controls starts.



7 When the installation is completed, all windows of Internet Explorer are closed.
8 Start Internet Explorer again and open the homepage of the robot.

B.9 TROUBLESHOOTING

If you encounter any trouble while you are setting up *i*RPickTool on a PC, refer to the troubleshooting below.

The robot homepage cannot be opened.

Cause: If Internet Explorer of your PC is configured to use the proxy server, the PC and controller may not be able to communicate with each other correctly.
Remedy: Set it as described in B.5, “CHANGING SETTINGS OF INTERNET EXPLORER”.

When you click [Workcell Setup] or [Workcell Product Status] of *i*RPickTool, [Failed to login Vision Setup] appears.

Cause: Windows firewall may be set incorrectly.
Remedy: Set it as described in B.6, “CHANGING SETTINGS OF WINDOWS FIREWALL”.

When you click [Workcell Setup] or [Workcell Product Status] of *i*RPickTool, [Enables popup on Internet Explorer] appears.

Cause: Internet Explorer may be set incorrectly.
Remedy: Set it as described in B.5, “CHANGING SETTINGS OF INTERNET EXPLORER”.

When you attempt to newly create objects for work cells, a runtime error occurs.

Cause: Internet Explorer may be set incorrectly.
Remedy: Set it as described in B.5, “CHANGING SETTINGS OF INTERNET EXPLORER”.

When you click [Workcell Setup] or [Workcell Product Status] of *i*RPickTool, [70: Cannot write] appears.

Cause: Internet Explorer may be set incorrectly.
Remedy: Set it as described in B.5, “CHANGING SETTINGS OF INTERNET EXPLORER”.

Even when you click [Workcell Setup] or [Workcell Product Status] of *i*RPickTool, no window opens.

Verify the following points.

Setting Windows Firewall

Cause1: Windows firewall may be set incorrectly.
Remedy1: Set it as described in B.6, “CHANGING SETTINGS OF WINDOWS FIREWALL”.

Security Software

Cause2: If security software is installed in your PC, communication may be blocked by the security software.
Remedy2: Disable the security software.

Internet Explorer

Cause3: An issue that Internet Explorer fails to open a new window on Windows 7 is reported.
Remedy3: It can cause the case. Verify whether it is the case with the following procedures:

1 Open the robot homepage on Internet Explorer.
2 Open [Tools] – [Developer Tools]. (It can be opened by pressing the F12 key.)
3 Select [Script] tab on the Developer Tools and select [Console] on the right pane.
4 Click [VISION SETUP] on the robot homepage.
5 If the message “script16386 no such interface supported” is shown on the console of the Developer Tools, it can be the case.

Some countermeasures are suggested on the Internet. If you contact your FANUC representative, they will provide you a countermeasure which has been effective.

Installing Vision UIF Controls is required even after it has been installed.

Verify the following points.

Trusted Sites

Cause1: If the setting of Trusted Sites in Internet Explorer is incorrect, you may not be able to install the Vision UIF Controls.
Remedy1: Refer to Subsection B.5, “CHANGING SETTINGS OF INTERNET EXPLORER”, and make sure that of the robot IP address is in Trusted Sites.

Add-ons

Cause2: If the Vision UIF Controls is disabled in [Manage add-ons] of Internet Explorer, the problem can occur.
Remedy2: You can verify whether it is disabled or not with the following procedures:

1 Open the robot homepage on Internet Explorer.
2 Open [Tools] - [Manage add-ons].
3 Select [Toolbars and Extensions] and then select [FRImageDisplay Control] in the list.

4 If its status is “Disabled”, click the [Enable] button at the bottom right of the window.
5 Click [Close]. Close all Internet Explorer windows, then restart IE.

ActiveX Filtering

Cause3: If the AcitveX Filtering is enabled in Internet Explorer, the problem can occur.
Remedy3: Open [Tool] – [Safety], and make sure that the [ActiveX Filtering] box is unchecked.

The alarm [PMON-001 Failed to notify PC Monitor] is displayed on the teach pendant during setting of a work cell.

Cause: Windows firewall may be set incorrectly.
Remedy: Set it as described in B.6, “CHANGING SETTINGS OF WINDOWS FIREWALL”.
If security software is installed in your PC, communication may be blocked by the security software. Disable the security software.

When you click [Workcell Setup] or [Workcell Product Status] of *i*RPickTool, Internet Explorer is closed with the message [A problem occurred].

Cause: Communication with the robot controller may not be performed normally due to the influence of the add-on software of Internet Explorer.
Remedy: Disable all add-ons except for those created by FANUC Robotics North America or FRNA, by choosing “Manage Add-on's” from the “Tools” menu of Internet Explorer. In this state, check whether the *i*RPickTool setup operation can be performed normally. If no problem arises, enable the disabled add-ons one at a time while checking that the *i*RPickTool setup operation is not affected.

No image is displayed on the *i*RVision teach screen.

Verify the following points.

Restarting PC

Cause1: Cases restarting PC resolves the problem have been reported.
Remedy1: Restart your PC and re-open the *i*RVision setup screen.

Administrator password

Cause2: When you log in to your PC as a user without the Administrator password, the PC might not normally communicate with the robot.
Remedy2: Log in to your PC as a user with the Administrator password.

Internet Information Server

Cause3: When Microsoft® Internet Information Server is installed in your PC and Worldwide Web Publishing Service is enabled, the PC might not communicate normally with the robot controller.
Remedy3: Disable the Worldwide Web Publishing Service.

Temporary Internet Files

Cause4: If you use a single PC to open Vision Setup on multiple robot controllers, the same IP address are used for the controllers, and different versions of robot software are installed to them, various problems, for example Vision Setup cannot be displayed or operated properly, may occur. This is because a setup page file from a controller is cashed by IE, and the caches page file is used for Vision Setup of another controller.
Remedy4: In this case, delete temporary internet files by opening [Delete Browsing History] dialog box from [Internet Options] – [General] – [Delete] and clicking [Delete] button. The problem can occur also when the controller software is updated.

NOTE
The [Delete Browsing History] dialog box has the check box [Preserve favorite website data] in recent versions of Internet Explorer. If it is checked, the [Delete] button does not delete the browsing history of the favorite website. It means the cached page files from a controller are not deleted if you added the robot homepage to the favorite website list. Un-checking it before clicking the [Delete] button.

When you try to load an image file, [Runtime error '0'] occurs.

Cause: When Internet Information Services (IIS) is enabled, communication with the robot controller may not be performed correctly.

Remedy: Choose Control Panel then open "Add or Remove Program". Next, uncheck "Internet Information Services (IIS)" on the list of "Windows Components".

Buttons to edit mask are hidden, and not operatable.

Verify the following points.

Trusted Sites

Cause1: If the setting of trusted sites in Internet Explorer is incorrect, the mask setting buttons are hidden and you may not be able to operate them.

Remedy1: Refer to Subsection B.5, "CHANGING SETTINGS OF INTERNET EXPLORER", and make sure that of the robot IP address is in Trusted Sites.

Address Bar, Status Bar

Cause2: If the address bar or the status bar is shown on the Vision Setup window, the mask setting buttons are hidden and you may not be able to operate them.

Remedy2: Refer to Subsection B.5, "CHANGING SETTINGS OF INTERNET EXPLORER" and make sure that the [Allow websites to open windows without address or status bars] box is checked.

The address bar is displayed on the Vision Setup page.

Cause: If the setting of trusted sites in Internet Explorer is incorrectly, the address bar is displayed on the Vision Setup screen and you may not be able to operate it.

Remedy: Refer to Subsection B.5, "CHANGING SETTINGS OF INTERNET EXPLORER", and make sure that of the robot IP address is in Trusted Sites.

When you try to finish editing masks, the CVIS-005 alarm is issued.

Cause: When Internet Information Services (IIS) is enabled, communication with the robot controller may not be performed correctly.

Remedy: Choose Control Panel then open "Add or Remove Program". Next, uncheck "Internet Information Services (IIS)" on the list of "Windows Components".

Vision UIF Controls cannot be installed

Remedy: Confirm that the "*i*RVision UIF Controls" option has been ordered in the controller used. The Vision UIF Controls can be installed only in the controller in which this option has been ordered. If this option has not been ordered in all robot controllers that participate in visual tracking, contact your FANUC representative.

C HIGH SPEED DI (HDI)

This chapter contains the hardware connection and the software setup of HDI (High speed DI for application).

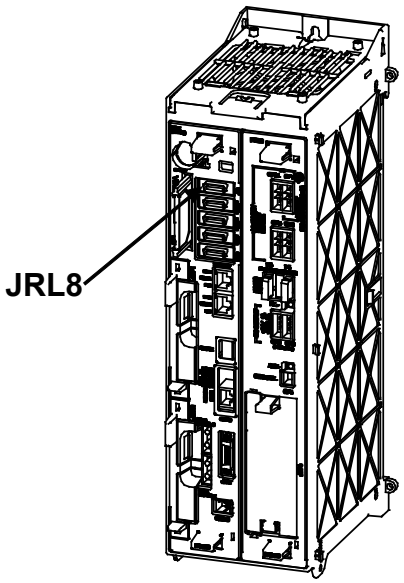
NOTE
HDI for Line Tracking option (J831) is required.

⚠ CAUTION
The HDI signals cannot be used as general-purpose DIs.

C.1 SCHEMATICS

This section contains the schematic drawings of cables used for the HDI interface and line tracking encoders.

C.1.1 For R-30*i*B Plus



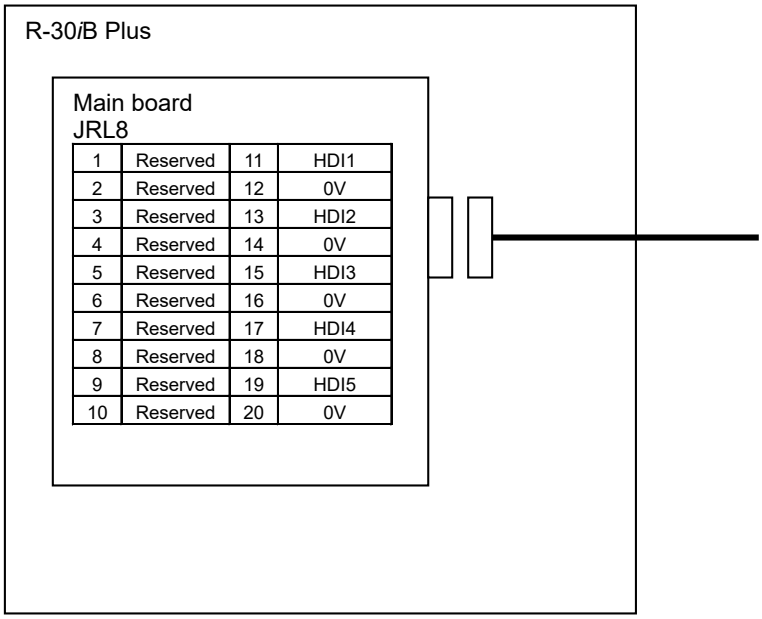


Fig. C.1.1 (a) R-30*i*B Plus HDI interface

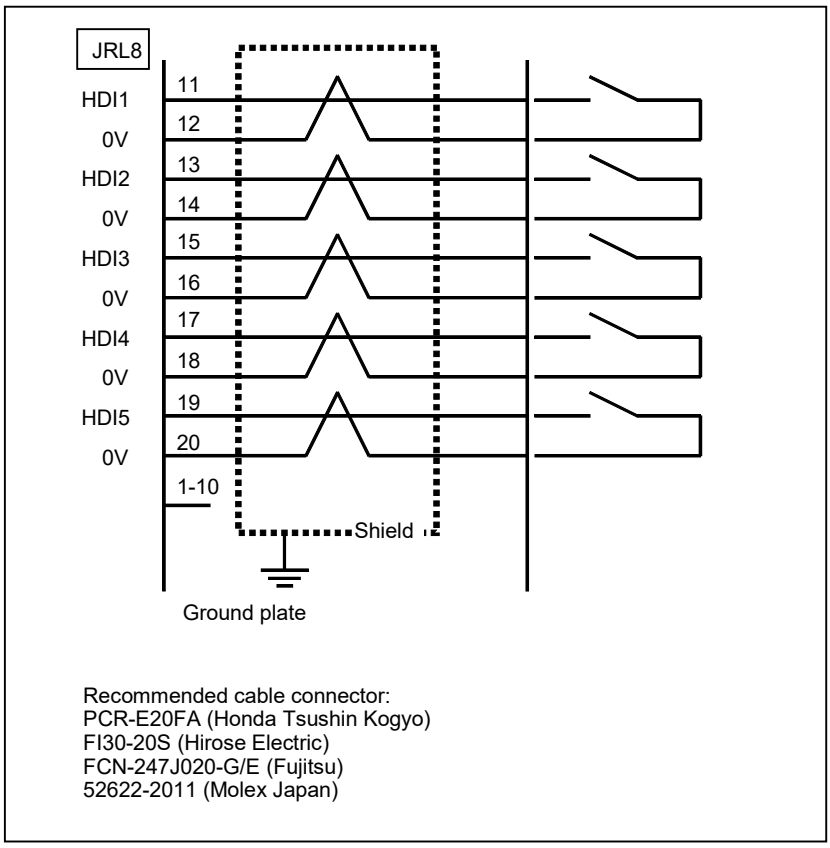
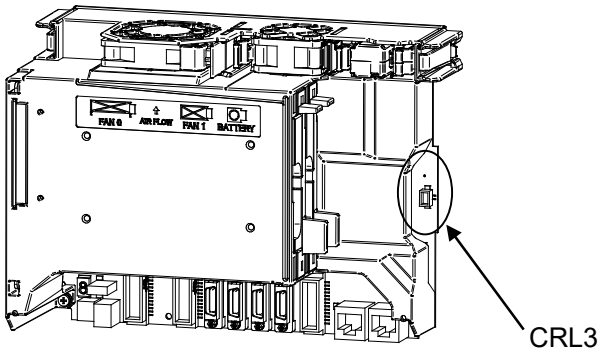


Fig. C.1.1 (b) R-30*i*B Plus HDI cable connections

C.1.2 For R-30*i*B Mate Plus



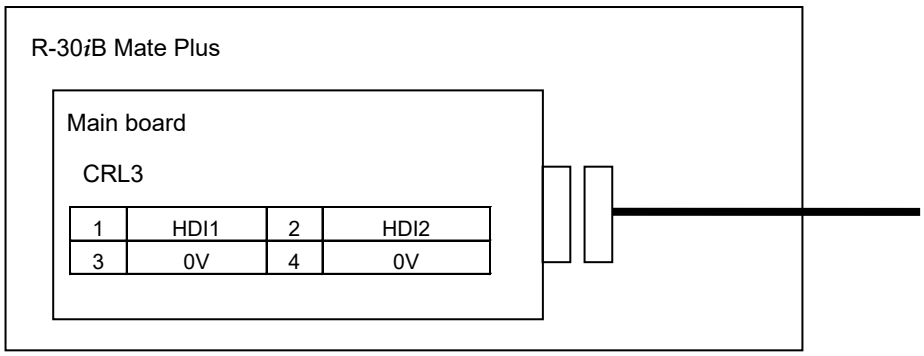


Fig. C.1.2 (a) R-30*i*B Mate Plus HDI interface

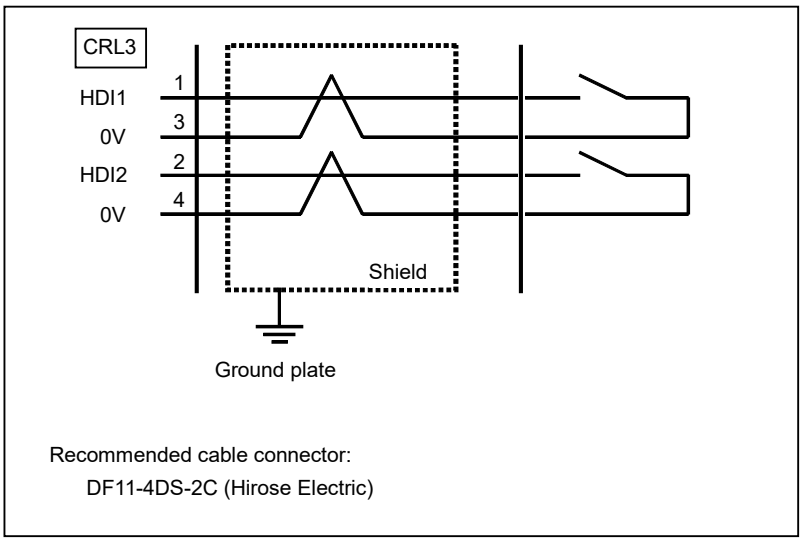
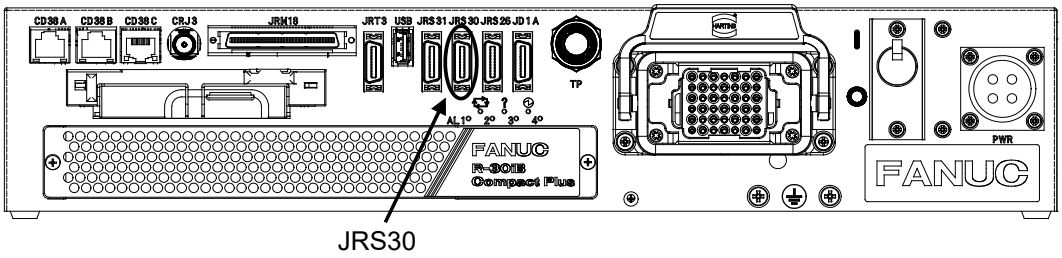
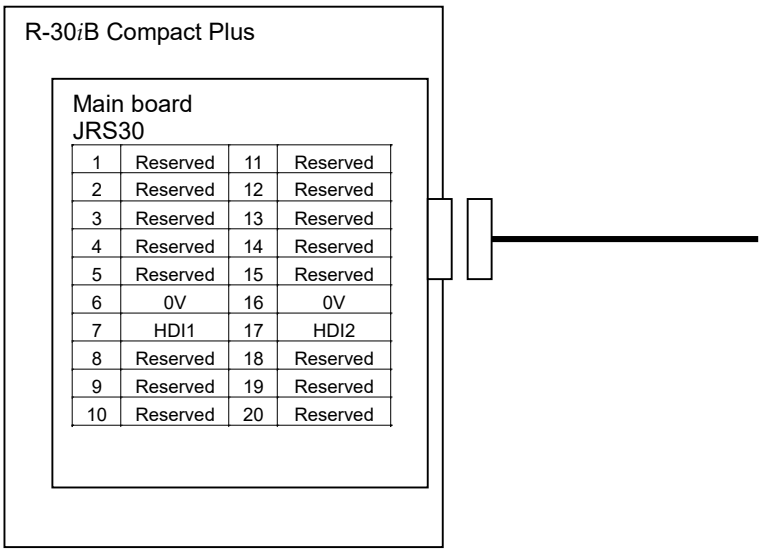


Fig. C.1.2 (b) R-30*i*B Mate Plus HDI cable connections

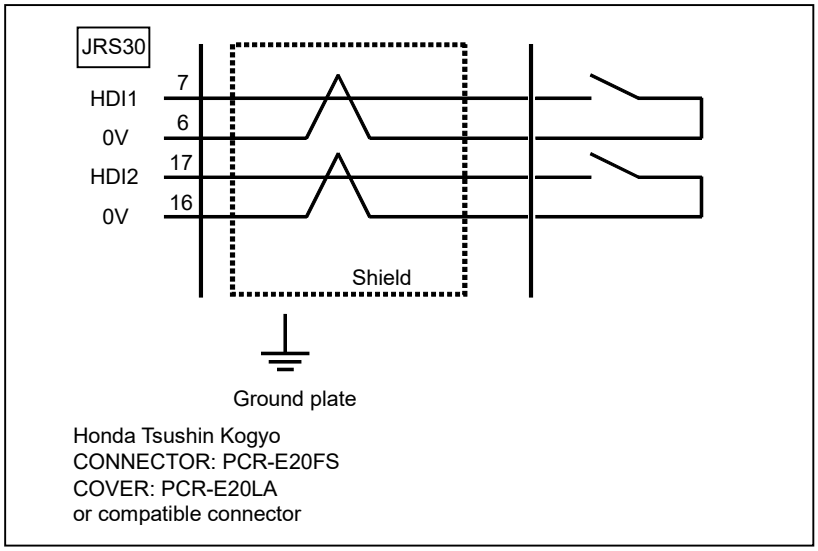
C.1.3 For R-30*i*B Compact Plus





1	Reserved	11	Reserved
2	Reserved	12	Reserved
3	Reserved	13	Reserved
4	Reserved	14	Reserved
5	Reserved	15	Reserved
6	0V	16	0V
7	HDI1	17	HDI2
8	Reserved	18	Reserved
9	Reserved	19	Reserved
10	Reserved	20	Reserved

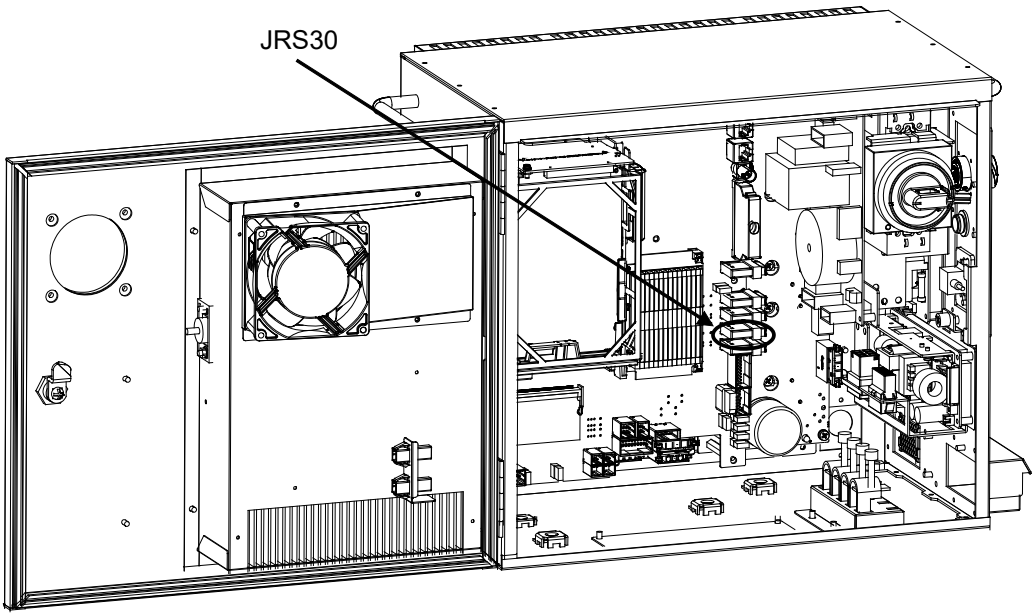
Fig. C.1.3 (a) R-30*i*B Compact Plus HDI interface



Honda Tsushin Kogyo
CONNECTOR: PCR-E20FS
COVER: PCR-E20LA
or compatible connector

Fig. C.1.3 (b) R-30*i*B Compact Plus HDI cable connections

C.1.4 For R-30*i*B Mini Plus



R-30*i*B Mini Plus

Main board
JRS30

1	Reserved	11	Reserved
2	Reserved	12	Reserved
3	Reserved	13	Reserved
4	Reserved	14	Reserved
5	Reserved	15	Reserved
6	0V	16	0V
7	HDI1	17	HDI2
8	Reserved	18	Reserved
9	Reserved	19	Reserved
10	Reserved	20	Reserved

Fig. C.1.4 (a) R-30*i*B Mini Plus HDI interface

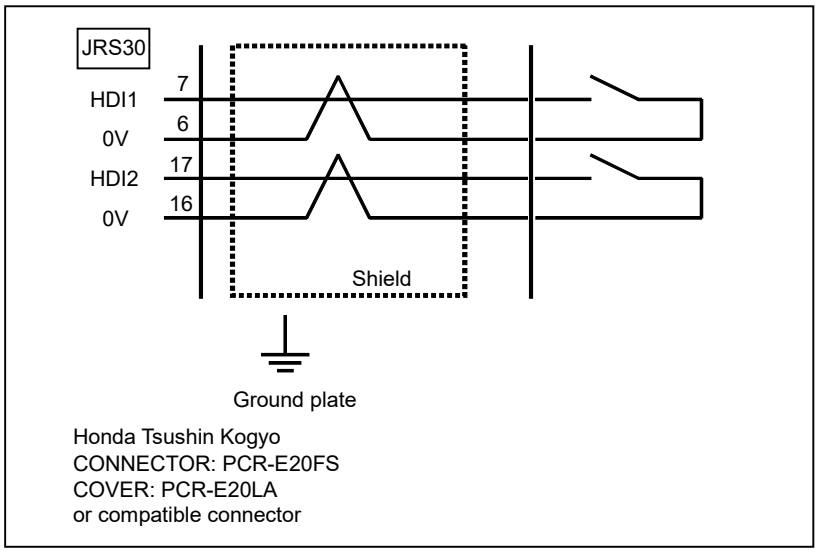


Fig. C.1.4 (b) R-30*i*B Mini Plus HDI cable connections

C.2 INPUT SIGNAL

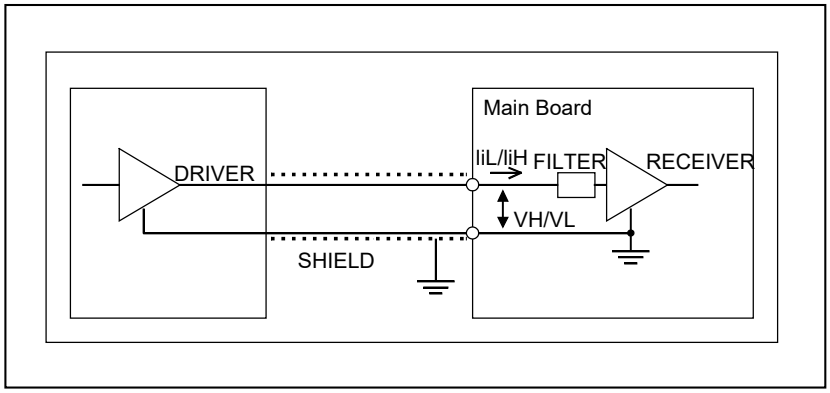


Fig. C.2 (a) HDI input signal

Absolute maximum rating
Input voltage range Vin: -3.6 to +10 V
Input characteristics

Unit	Symbol	Specification	Unit	Remark
High level input voltage	VH	3.6 to 11.6	V	
Low level input voltage	VL	0 to 1.0	V	
High level input current	IiH	2 max	mA	Vin=5 V
		11 max	mA	Vin = 10 V
Low level input current	IiL	-8.0 max	mA	Vin = 0 V
Input signal pulse duration		20 min	μs	
Input signal delay or variations		0.02(max)	ms	

NOTE
1 The plus (+) sign of IiH/IiL represents the direction of flow into the receiver. The minus (-) sign of IiH/IiL represents the direction of flow out of the receiver.
2 The high-speed skip signal is assumed to be 1 when the input voltage is at the low level and 0 when it is at the high level.

C.3 SETUP OF HDI

C.3.1 Overview

The HDI for Line Tracking feature (J831) ensures an accurate part detection process when the conveyor operates at very fast speeds. This function uses HDI (High speed DI for application) in place of the standard digital input normally used for part detection.
To use HDI for Line Tracking, you must

- Set HDI port id in Encoder Setup Menu

C.3.2 Enabling High Speed Scanning

Use Procedure C-1 to enable the High Speed Scanning feature.

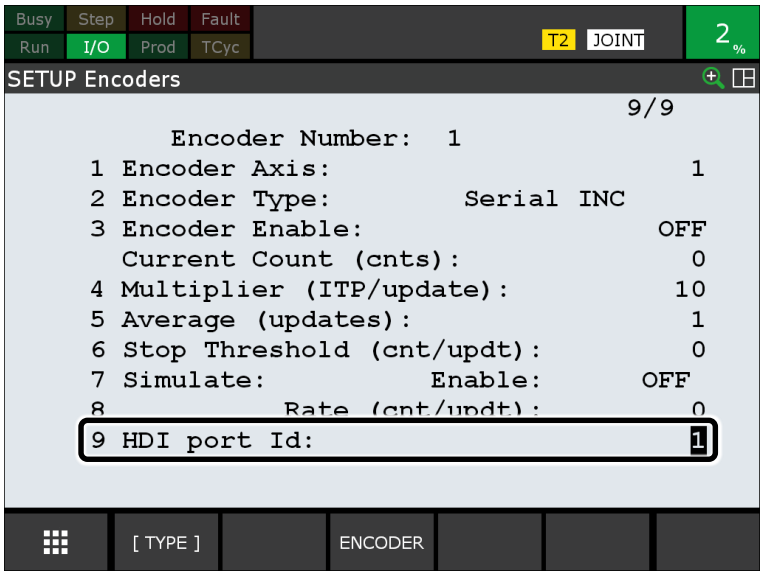
Procedure C-1 Enabling High Speed Scanning

Preparations

- Order the High Speed Scanning option.
- Wire the part detect hardware to HDI#1. (Refer to Appendix C.1 to C.2)
- R-30*i*B Plus supports up to 5 High Speed Digital Input (HDI) #1 to #5, located on the JRL8 connector of the controller.

Steps

1. Press the [MENU] key on the teach pendant of the robot controller.
2. Select [NEXT]-[SYSTEM]-[Variables].
3. Move the cursor to the following variables and check they are set to the following values.
4. When High Speed Scanning option has been loaded, they are set to the following values by default.
   - $HSLTENBL = TRUE
   - $LNCFG.$HSDI_ENABLE = TRUE
5. After changing above variables, turn off the controller, and then turn it on again to accept the new setting.
6. Use the Encoder Setup Menu to input used HDI port id to HDI port id.
7. Turn off the controller, and then turn it on again to accept the new setting.



NOTE

When HDI for Line Tracking (J831) is ordered and tracking sub program is taught by using normal DI input, set $LNCFG.$HSDI_ENABLE to FALSE. Otherwise, it is possible that tracking program can be not edited.

D αA1000S PULSECODER

This chapter contains the usage of αA1000S Pulsecoder (A860-0372-T001).

D.1 REQUIREMENTS

Table D.1 (a) Requirements (R-30*i*B Plus: αA1000S Pulsecoder: A860-0372-T001)

Required Components	R-30*i*B Plus Controller A-Cabinet	R-30*i*B Plus Controller B-Cabinet	Comments
Hardware			
Line Tracking Interface Board	A20B-8101-0421 (wide-mini slot) or A20B-8101-0601 (mini slot)	A20B-8101-0421 (wide-mini slot) or A20B-8101-0601 (mini slot)	• The Line tracking interface board in the left column is included in the following order specification. A05B-2600-J035, A05B-2660-J035 (A/B-Cabinet, wide-mini slot) A05B-2600-J036, A05B-2660-J036 (A-Cabinet, mini slot) A05B-2600-J037 (B-Cabinet, mini slot) • Separate Detector Unit (SDU) - A02B-0323-C205 can be used in place of Line tracking Interface board. NOTE: SDU requires retrofit work to mount in the container. (See APPENDIX F “SEPARATE DETECTOR UNIT (SDU)”.) • αA1000S Pulsecoder can be also connected to encoder terminal (JD17) on Main CPU board. In this case, Line tracking interface board is made redundant, but Learning Vibration Control function (A05B-2600-J573) (option) cannot be used.
Fiber Optic (FSSB) Cable	A66L-6001-0023	A66L-6001-0023	• The Fiber Optic cable in the left column is included in the following order specification. A05B-2600-J035, A05B-2660-J035 (A/B-Cabinet, wide-mini slot) A05B-2600-J036, A05B-2660-J036 (A-Cabinet, mini slot) A05B-2600-J037 (B-Cabinet, mini slot)

Required Components	R-30*i*B Plus Controller A-Cabinet	R-30*i*B Plus Controller B-Cabinet	Comments
Hardware			
Pulsecoder Cable (In case of an αA1000S Pulsecoder as an incremental Pulsecoder)	for A20B-8101-0421 (wide-mini slot) : A05B-2601-J220 (7M) A05B-2601-J221 (14M) A05B-2601-J222 (20M) A05B-2601-J223 (30M)	for A20B-8101-0421 (wide-mini slot) : A05B-2603-J220 (7M) A05B-2603-J221 (14M) A05B-2603-J222 (20M) A05B-2603-J223 (30M)	• In case of a αA1000S Pulsecoder as an absolute Pulsecoder, cables in this table cannot be used. • When the pulse multiplexer is used, αA1000S Pulsecoder cannot be used. If you want to use multiple robots with αA1000S Pulsecoder, use Ethernet Encoder function (A05B-2600-R762) (option).
	for A20B-8101-0601 (mini slot) (one Pulsecoder) : A05B-2601-J210 (7M) A05B-2601-J211 (14M) A05B-2601-J212 (20M) A05B-2601-J213 (30M)	for A20B-8101-0601 (mini slot) (one Pulsecoder) : A05B-2603-J210 (7M) A05B-2603-J211 (14M) A05B-2603-J212 (20M) A05B-2603-J213 (30M)	
	for A20B-8101-0601 (mini slot) (two Pulsecoders) : A05B-2601-J260 (7M) A05B-2601-J261 (14M) A05B-2601-J262 (20M) A05B-2601-J263 (30M)	for A20B-8101-0601 (mini slot) (two Pulsecoders) : A05B-2603-J260 (7M) A05B-2603-J261 (14M) A05B-2603-J262 (20M) A05B-2603-J263 (30M)	
	for Main board : A05B-2601-J270 (7M) A05B-2601-J271 (14M) A05B-2601-J272 (20M) A05B-2601-J273 (30M)	for Main board : A05B-2603-J270 (7M) A05B-2603-J271 (14M) A05B-2603-J272 (20M) A05B-2603-J273 (30M)	

Table D.1 (b) Requirements (R-30*i*B Mate Plus: αA1000S Pulsecoder: A860-0372-T001)

Required Components	R-30*i*B Mate Plus Controller	R-30*i*B Mate Plus Controller (Open Air)	Comments
Hardware			
Line Tracking Interface Board	A20B-8101-0601 (mini slot)	A20B-8101-0601 (mini slot)	• The Line tracking interface board in the left column is included in the following order specification. A05B-2650-J035, A05B-2661-J035 (R-30*i*B Mate Plus, mini slot) A05B-2655-J035, A05B-2662-J035 (R-30*i*B Mate Plus (Open Air), mini slot) • αA1000S Pulsecoder can be also connected to encoder terminal (CRS41) on Main CPU board. In this case, Line tracking interface board is made redundant, but Learning Vibration Control function (A05B-2600-J573) (option) cannot be used.

Required Components	R-30*i*B Mate Plus Controller	R-30*i*B Mate Plus Controller (Open Air)	Comments
	Hardware		
Fiber Optic (FSSB) Cable	A66L-6001-0026	A66L-6001-0023	• The Fiber Optic cable in the left column is included in the following order specification. A05B-2650-J035, A05B-2661-J035 (R-30*i*B Mate Plus, mini slot) A05B-2655-J035, A05B-2662-J035 (R-30*i*B Mate Plus (Open Air), mini slot)
Pulsecoder Cable (In case of an αA1000S Pulsecoder as an incremental Pulsecoder)	for A20B-8101-0601 (mini slot) (one Pulsecoder) : A05B-2650-J205 (7M) A05B-2650-J206 (14M) A05B-2650-J207 (20M) A05B-2661-J205 (7M) A05B-2661-J206 (14M) A05B-2661-J207 (20M)	for A20B-8101-0601 (mini slot) (one Pulsecoder) : A05B-2655-J205 (7M) A05B-2655-J206 (14M) A05B-2655-J207 (20M) A05B-2662-J205 (7M) A05B-2662-J206 (14M) A05B-2662-J207 (20M)	• In case of a αA1000S Pulsecoder as an absolute Pulsecoder, cables in this table cannot be used. • When the pulse multiplexer is used, αA1000S Pulsecoder cannot be used. If you want to use multiple robots with αA1000S Pulsecoder, use Ethernet Encoder function (A05B-2600-R762) (option). • For R-30*i*B Mate Plus Main CPU board (A20B-8201-0750), interfaces connecting αA1000S Pulsecoder to Main CPU board cannot be used. See Table D.1 (d) for Main CPU boards that αA1000S Pulsecoder can be connected.
	for A20B-8101-0601 (mini slot) (two Pulsecoders) : A05B-2650-J215 (7M) A05B-2650-J216 (14M) A05B-2650-J217 (20M) A05B-2661-J215 (7M) A05B-2661-J216 (14M) A05B-2661-J217 (20M)	for A20B-8101-0601 (mini slot) (two Pulsecoders) : A05B-2655-J215 (7M) A05B-2655-J216 (14M) A05B-2655-J217 (20M) A05B-2662-J215 (7M) A05B-2662-J216 (14M) A05B-2662-J217 (20M)	
	for Main CPU board : A05B-2650-J220 (7M) A05B-2650-J221 (14M) A05B-2650-J222 (20M) A05B-2661-J220 (7M) A05B-2661-J221 (14M) A05B-2661-J222 (20M)	for Main CPU board : A05B-2655-J220 (7M) A05B-2655-J221 (14M) A05B-2655-J222 (20M) A05B-2662-J220 (7M) A05B-2662-J221 (14M) A05B-2662-J222 (20M)	

Table D.1 (c) Requirements (R-30*i*B Compact Plus / R-30*i*B Mini Plus: αA1000S Pulsecoder: A860-0372-T001)

Required Components	R-30*i*B Compact Plus Controller	R-30*i*B Mini Plus Controller	Comments
Hardware			
Pulsecoder Cable (In case of an αA1000S Pulsecoder as an incremental Pulsecoder)	for Main CPU board (one Pulsecoder) : A05B-2690-J300 (7M) A05B-2690-J301 (14M) A05B-2690-J302 (20M)	for Main CPU board (one Pulsecoder) : A05B-2696-J300 (7M) A05B-2696-J301 (14M) A05B-2696-J302 (20M)	• In case of a αA1000S Pulsecoder as an absolute Pulsecoder, cables in this table cannot be used. • If you want to use multiple robots with αA1000S Pulsecoder, use Ethernet Encoder function (A05B-2600-R762) (option). • For R-30*i*B Compact Plus Main CPU board (A20B-8100-0800), interfaces connecting αA1000S Pulsecoder to Main CPU board cannot be used. See Table D.1 (e) for Main CPU board that αA1000S Pulsecoder can be connected.
	for Main CPU board (two Pulsecoders) : A05B-2690-J310 (7M) A05B-2690-J311 (14M) A05B-2690-J312 (20M)	for Main CPU board (two Pulsecoders) : A05B-2696-J310 (7M) A05B-2696-J311 (14M) A05B-2696-J312 (20M)	

Refer to Fig. D.2 (a) to Fig. D.2 (b) for Pulsecoder signal information, specifications, and images containing the dimensions of the Pulsecoders.
Refer to the following for information on connections and installation of the cables.

- For R-30*i*B Plus : Fig. D.3.1 (a) to Fig. D.3.1 (e)
- For R-30*i*B Mate Plus : Fig. D.3.2 (a) to Fig. D.3.2 (d)
- For R-30*i*B Compact Plus : Fig. D.3.3 (a) to Fig. D.3.3 (c)
- For R-30*i*B Mini Plus : Fig. D.3.4 (a) to Fig. D.3.4 (c)

αA1000S Pulsecoder cannot be connected to Main CPU board, depending on the Board Specification of the controller. Details are shown in the following table. Be careful in consideration of configuration

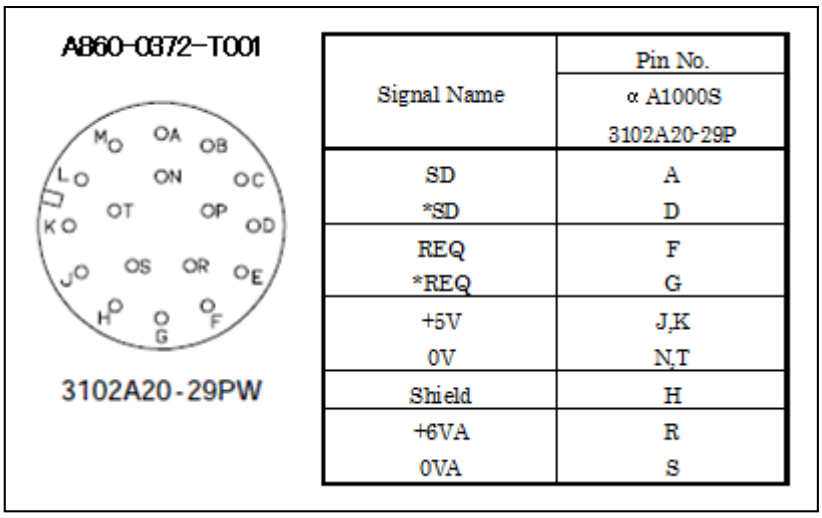
Table D.1 (d) Requirements (R-30*i*B Mate Plus Main CPU board: αA1000S Pulsecoder)

Main CPU Board	Board Specification	Connection of αA1000S Pulsecoder	Comments
Main CPU Board A	A20B-8201-0750 (R-30*i*B Mate Plus)	Not Available	• The Main CPU board in the left column is included in the following order specification. A05B-2680-H001 (R-30*i*B Mate Plus) A05B-2685-H001 (R-30*i*B Mate Plus (Open Air))
Main CPU Board B	A20B-8201-0751 (R-30*i*B Mate Plus)	Available	• The Main CPU board in the left column is included in the following order specification. A05B-2680-H002 (R-30*i*B Mate Plus) A05B-2685-H002 (R-30*i*B Mate Plus (Open Air))
Main CPU Board C	A20B-8201-0752 (R-30*i*B Mate Plus)	Available	• The Main CPU board in the left column is included in the following order specification. A05B-2680-H001 (R-30*i*B Mate Plus) A05B-2685-H003 (R-30*i*B Mate Plus (Open Air))

Table D.1 (e) Requirements (R-30*i*B Compact Plus Main CPU board: αA1000S Pulsecoder)

Main CPU Board	Board Specification	Connection of αA1000S Pulsecoder	Comments
Main CPU Board A	A17B-8100-0800	Not Available	• The Main CPU board in the left column is included in the following order specification. A05B-2690-H001
Main CPU Board B	A17B-8100-0801	Available	• The Main CPU board in the left column is included in the following order specification. A05B-2690-H002

D.2 FIGURES



Signal Name	Pin No. α A1000S 3102A20-29P
SD *SD	A D
REQ *REQ	F G
+5V 0V	J,K N,T
Shield	H
+6VA 0VA	R S

Fig. D.2 (a) αA1000S Pulsecoder (A860-0372-T001) connection signal information

Table D.2 A1000S Pulsecoder (A860-0372-T001) specifications

Item		Specification
Power voltage		5 [V]±5%
Current consumption		Up to 0.3 [A]
Working temperature range		0 to +60 [°C]
Resolution		1 000 000 [/rev.] (NOTE: The resolution that is used in Line tracking function is 65 536 [/rev.].)
Maximum speed of revolution		4000 [min^{-1}]
Input shaft inertia		Up to 1 x 10^{-4} [kg m^2]
Input shaft startup torque		Up to 0.1 [N m]
Ratio loads	Radial	98 [N]
	Axial	49 [N]
Shaft diameter runout		0.02 x 10^{-3} [m]
Configuration		Dust proof and drip-proof (equivalent to IP55: when using waterproof connector)
Vibration proof acceleration		5 [G] (50 to 2,000[Hz])
Weight		Approx. 0.75 [kg]

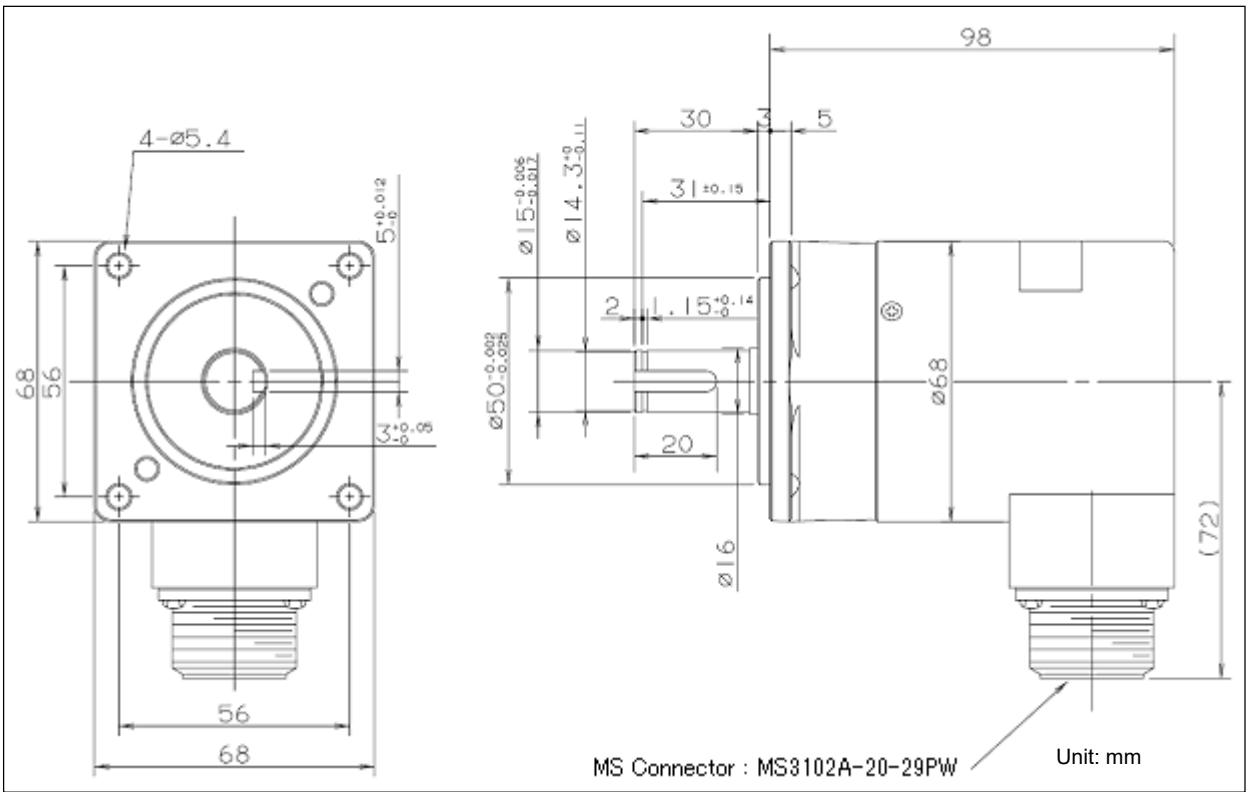


Fig. D.2 (b) αA1000S Pulsecoder dimensions (A860-0372-T001)

D.3 HOW TO CONNECT

For line tracking using R-30*i*B Plus / R-30*i*B Mate Plus / R-30*i*B Compact Plus / R-30*i*B Mini Plus, the encoder configurations shown in Table D.3 are available.

NOTE
The only encoder that can be connected to the encoder terminal of Main CPU board is αA1000S Pulsecoder (A860-0372-T001).

Table D.3 Examples of encoder configuration (with αA1000S Pulsecoder)

Number of Encoders	Example of encoder configuration	Comments
1	• **Encoder #1 :** Connect to encoder terminal of Main CPU board (R-30*i*B Plus: JD17、R-30*i*B Mate Plus: CRS41(*NOTE 1) / R-30*i*B Compact Plus: JRS31(*NOTE 2) / R-30*i*B Mini Plus: JRS31)	
2	• **Encoder #1, #2 :** For R-30*i*B Plus / R-30*i*B Mate Plus : Connect to encoder terminal of line tracking interface board (JF21 to JF22) For R-30*i*B Compact Plus(*NOTE 2) / R-30*i*B Mini Plus : Connect to encoder terminal of Main CPU board(JRS31)	
3	• **Encoder #1, #2 :** Connect to encoder terminal of line tracking interface board (JF21 to JF22) • **Encoder #3:** Connect to encoder terminal of Main CPU board (R-30*i*B Plus: JD17、R-30*i*B Mate Plus: CRS41(*NOTE 1))	• If you configure with combination of a Line tracking interface board and an interface of Main CPU board, make sure to set the last encoder number to the encoder connected to Main CPU board. In case of the example in the left column, set Encoder #1 to #2 for the Pulsecoders connected to the Line tracking interface board, and Encoder #3 for the Pulsecoder connected to the Main CPU board.
4	• **Encoder #1 to #4 :** Connect to encoder terminal of Separate Detector Unit (SDU) basic unit (JF101 to JF104)	• When you use SDU, you need retrofit work to mount SDU in the container, Refer to “F. SEPARATE DETECTOR UNIT (SDU)” for details.

Number of Encoders	Example of encoder configuration	Comments
5	• **Encoder #1 to #4 :** Connect to encoder terminal of Separate Detector Unit (SDU) basic unit (JF101 to JF104) • **Encoder #5 :** Connect to encoder terminal of Main CPU board (R-30*i*B Plus: JD17、R-30*i*B Mate Plus: CRS41(*NOTE 1))	• If you configure with combination of SDU and an interface of Main CPU board, make sure to set the last encoder number to the encoder connected to Main CPU board. In case of the example in the left column, set Encoder #1 to #4 for the Pulsecoders connected to the SDU, and Encoder #5 for the Pulsecoder connected to the Main CPU board. • When you use SDU, you need retrofit work to mount SDU in the container, Refer to “F. SEPARATE DETECTOR UNIT (SDU)” for details.
6	• **Encoder #1 to #4 :** Connect to encoder terminal of Separate Detector Unit (SDU) basic unit (JF101 to JF104) • **Encoder #5, #6 :** Connect to encoder terminal of SDU expansion unit (JF105 to JF106)	• When you use SDU, you need retrofit work to mount SDU in the container, Refer to “F. SEPARATE DETECTOR UNIT (SDU)” for details.
7	• **Encoder #1 to #4 :** Connect to encoder terminal of Separate Detector Unit (SDU) basic unit (JF101 to JF104) • **Encoder #5 to #7 :** Connect to encoder terminal of SDU expansion unit (JF105 to JF107)	• When you use SDU, you need retrofit work to mount SDU in the container, Refer to “F. SEPARATE DETECTOR UNIT (SDU)” for details.
8	• **Encoder #1 to #4 :** Connect to encoder terminal of Separate Detector Unit (SDU) basic unit (JF101 to JF104) • **Encoder #5 to #8 :** Connect to encoder terminal of SDU expansion unit (JF105 to JF108)	• When you use SDU, you need retrofit work to mount SDU in the container, Refer to “F. SEPARATE DETECTOR UNIT (SDU)” for details.

NOTE

1 For R-30*i*B Mate Plus Main CPU Board A (A20B-8201-0750), interfaces that connect αA1000S Pulsecoder to Main CPU board cannot be used. Use Line tracking interface boards in such case. Refer to Table D.1 (d) for Main CPU boards that αA1000S Pulsecoder can be connected.

2 For R-30*i*B Compact Plus Main CPU Board A (A17B-8100-0800), interfaces that connect αA1000S Pulsecoder to Main CPU board cannot be used. Refer to Table D.1 (e) for Main CPU board that αA1000S Pulsecoder can be connected.

3 Use Separate Detector Unit (SDU) described in APPENDIX F, in the case that Line tracking interface boards and Main CPU boards cannot be used or are not available.

D.3.1 For R-30*i*B Plus

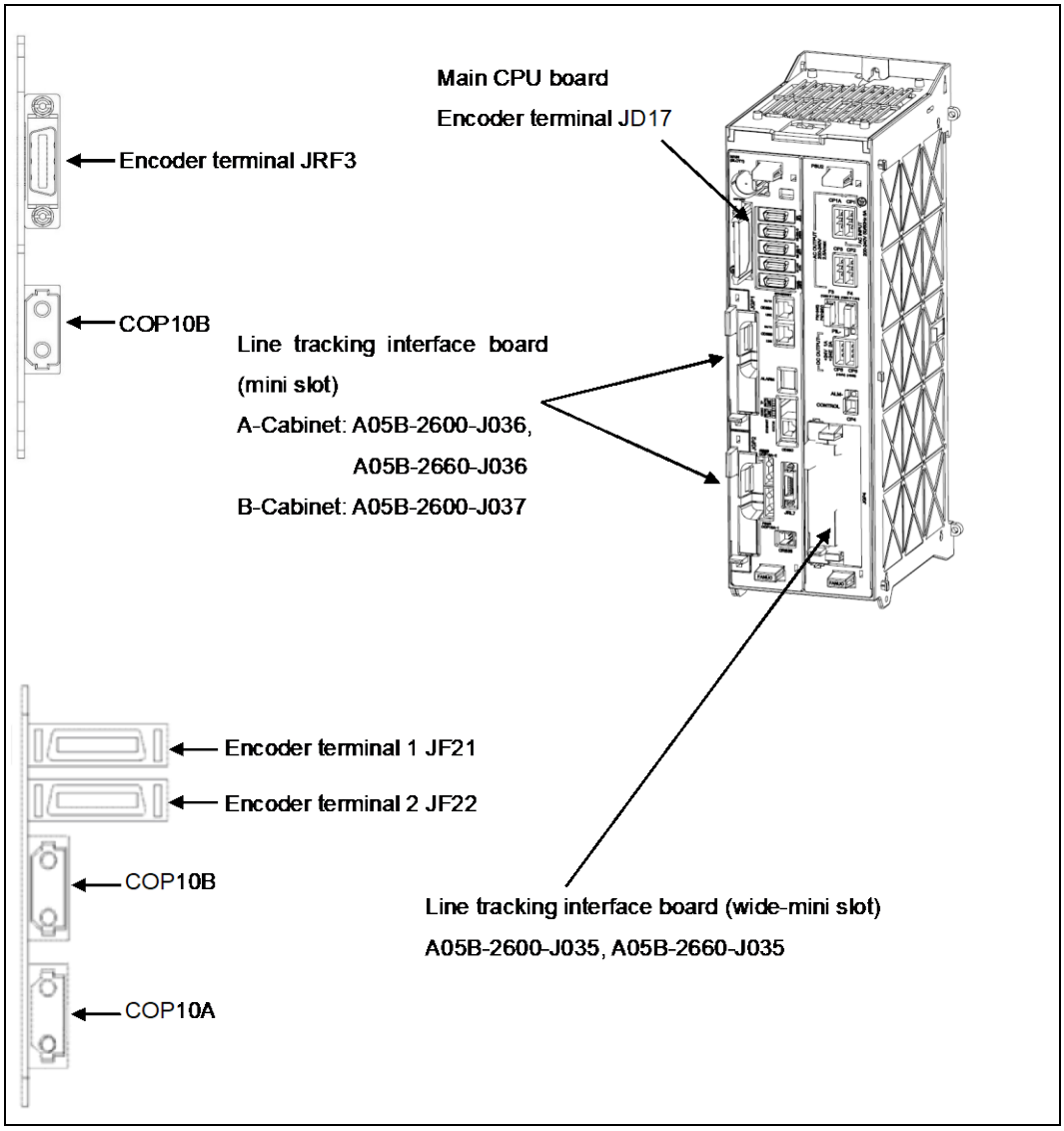


Fig. D.3.1 (a) R-30*i*B Plus Controller connecting encoders (Pulsecoder)

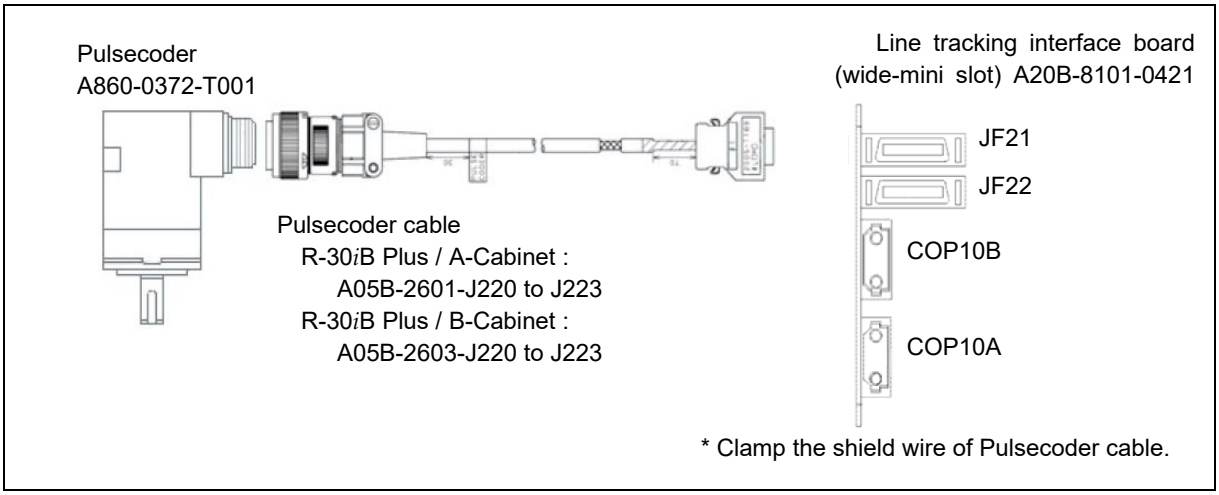


Fig. D.3.1 (b) Connecting cables with Line tracking interface board A05B-2600-J035 or A05B-2660-J035 (one αA1000S Pulsecoder)

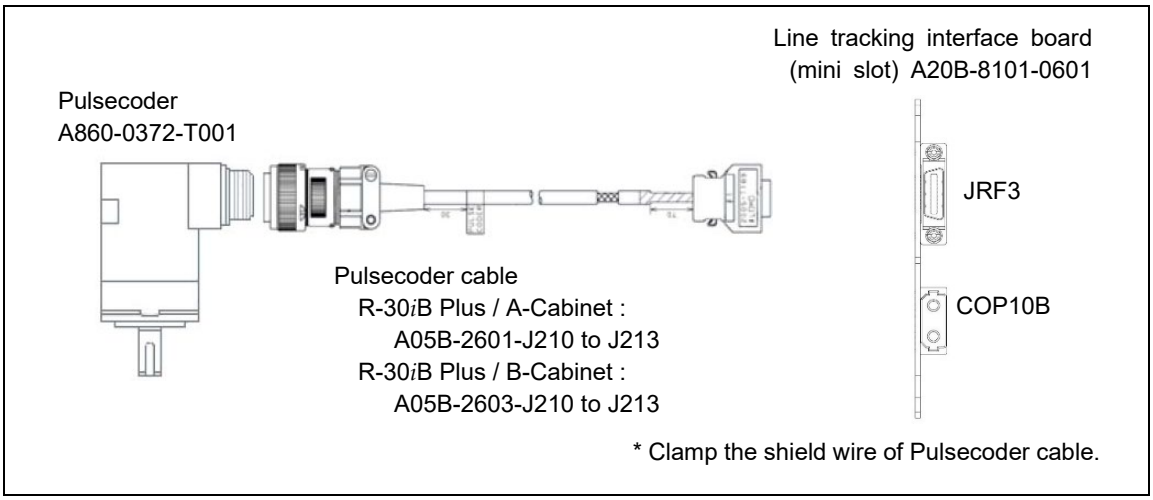


Fig. D.3.1 (c) Connecting cables with Line tracking interface board A05B-2600-J036, A05B-2660-J036 or A05B-2600-J037 (one αA1000S Pulsecoder)

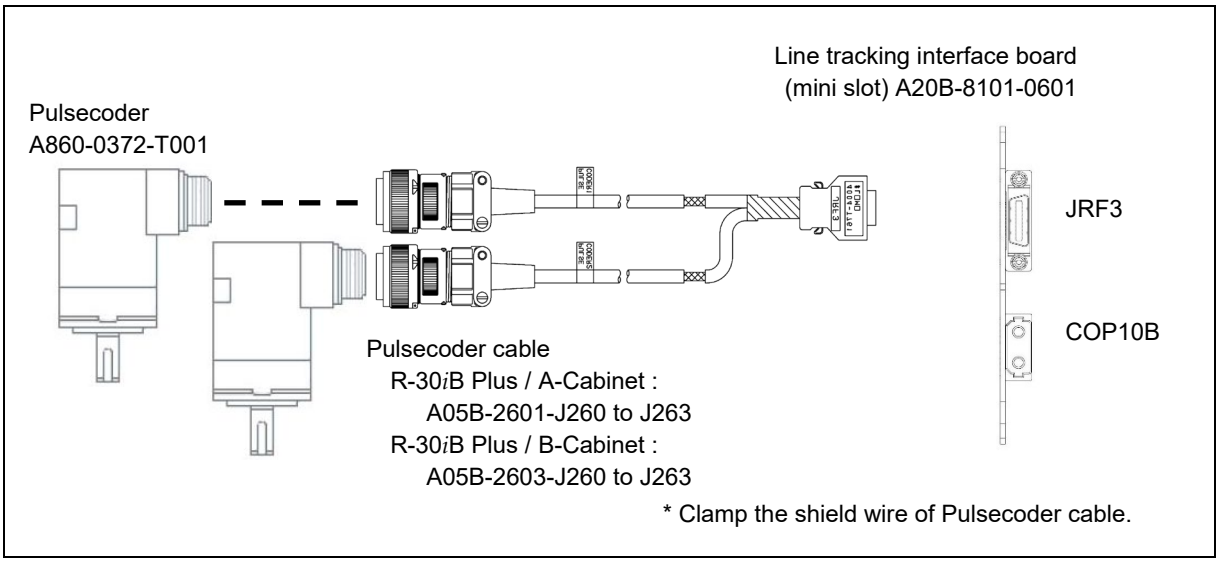


Fig. D.3 (d) Connecting cables with Line tracking interface board A05B-2600-J036, A05B-2660-J036 or A05B-2600-J037 (two αA1000S Pulsecoders)

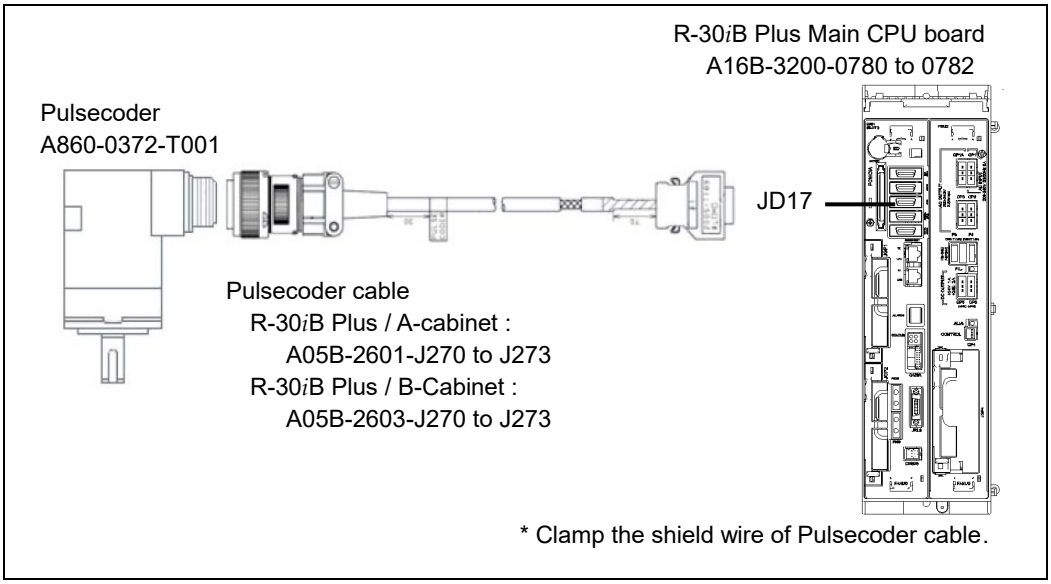


Fig. D.3.1 (e) Connecting cables with R-30iB Plus Main CPU board (one αA1000S Pulsecoder)

D.3.2 For R-30*i*B Mate Plus

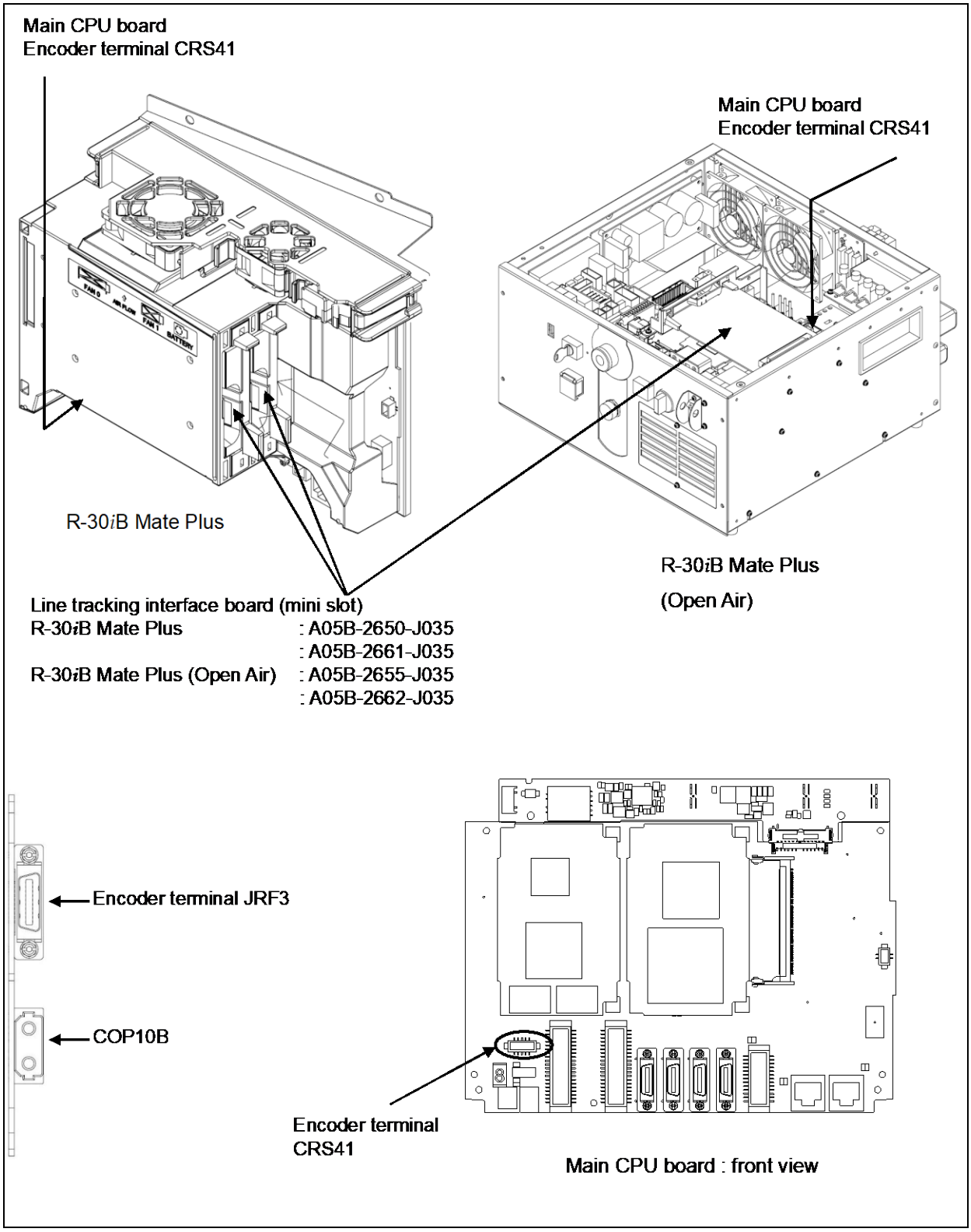


Fig. D.3.2 (a) R-30*i*B Mate Plus Controller connecting encoders (Pulsecoder)

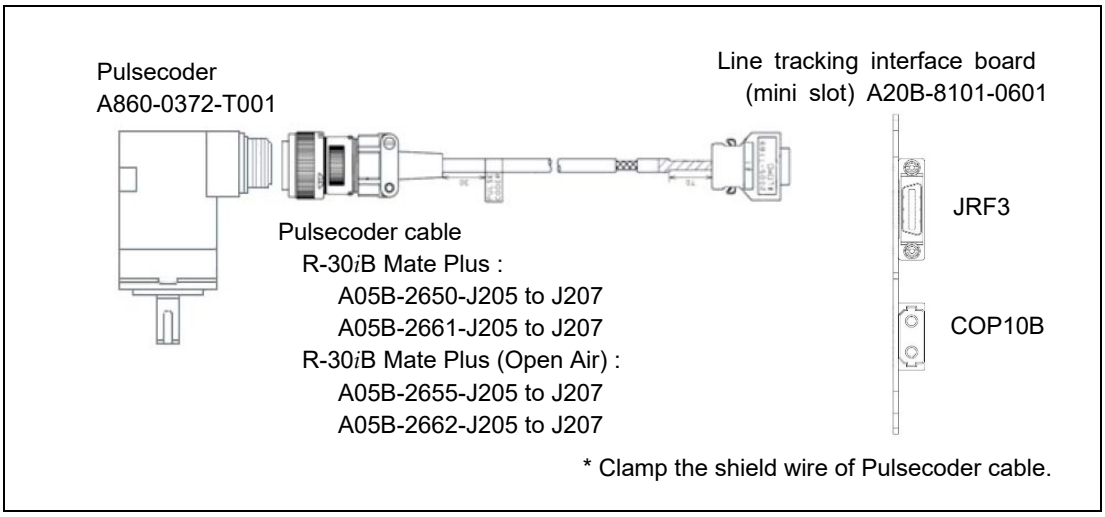


Fig. D.3.2 (b) Connecting cables with Line tracking interface board A05B-2650-J035, A05B-2655-J035, A05B-2661-J035, A05B-2662-J035 (one αA1000S Pulsecoder)

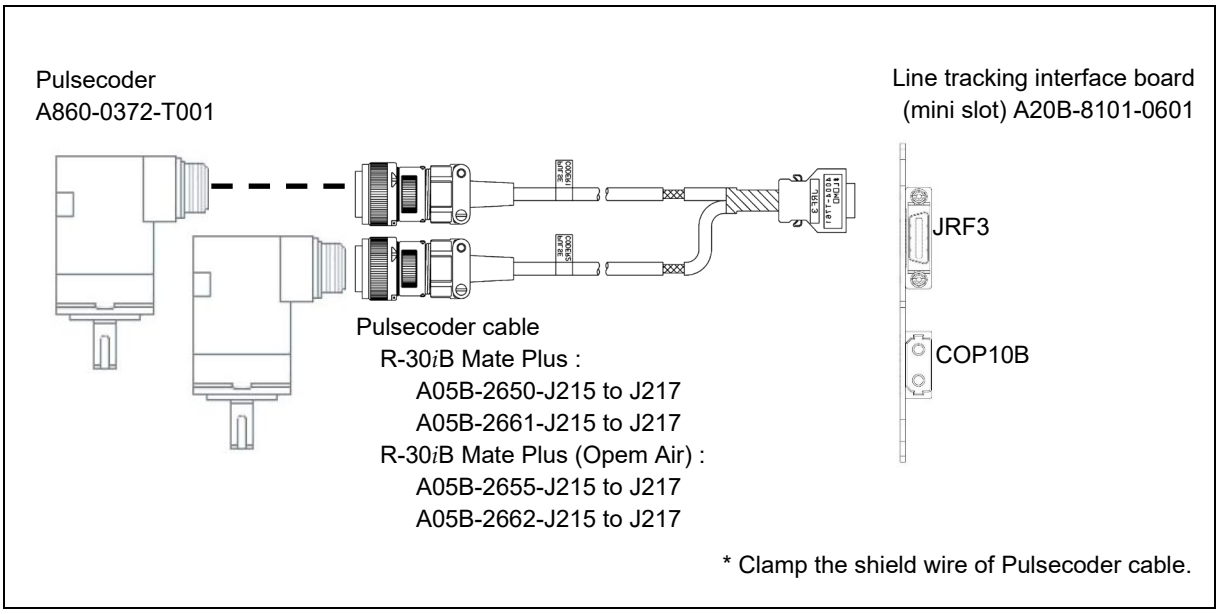


Fig. D.3.2 (c) Connecting cables with Line tracking interface board A05B-2650-J035, A05B-2655-J035, A05B-2661-J035, A05B-2662-J035 (two αA1000S Pulsecoders)

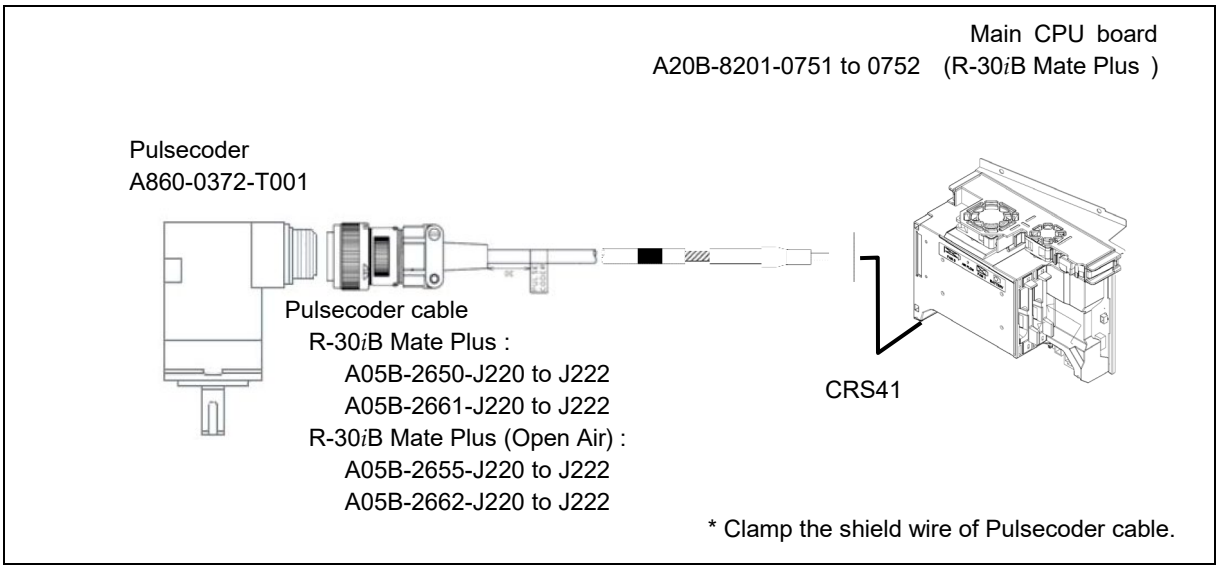


Fig. D.3.2 (d) Connecting cables with R-30*i*B Mate Plus Main CPU board (one αA1000S Pulsecoder)

D.3.3 For R-30*i*B Compact Plus

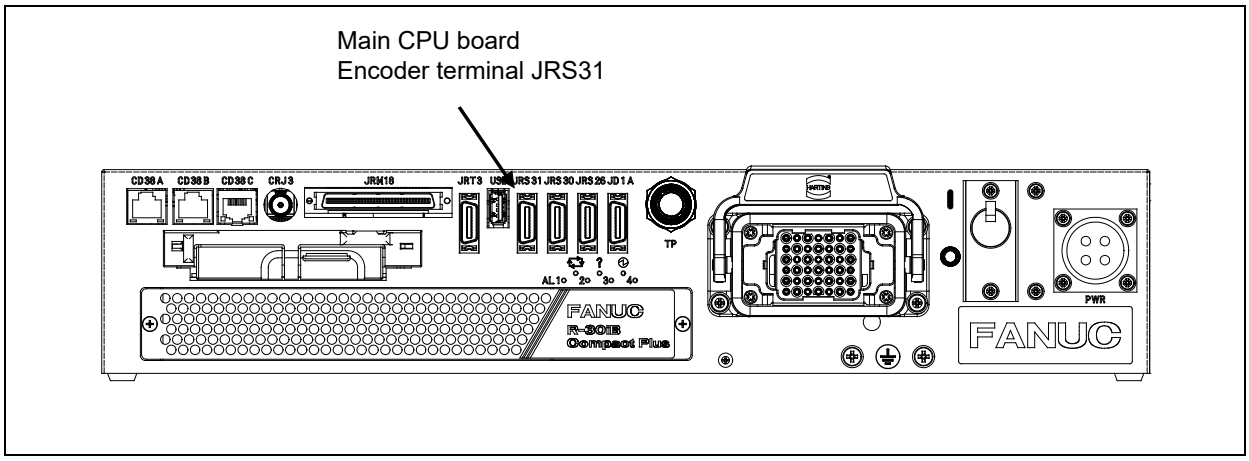


Fig. D.3.3 (a) R-30*i*B Compact Plus Controller connecting encoders (Pulsecoder)

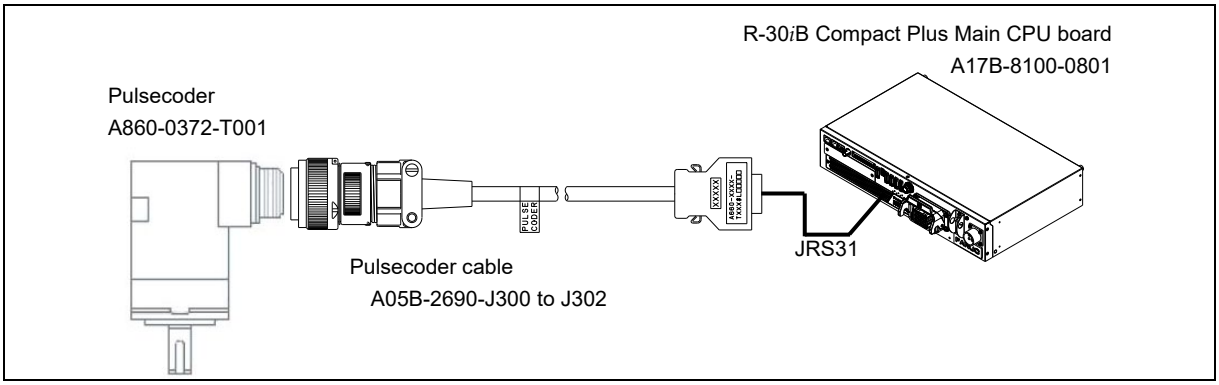


Fig. D.3.3 (b) Connecting cables with R-30*i*B Compact Plus Main CPU board (one αA1000S Pulsecoder)

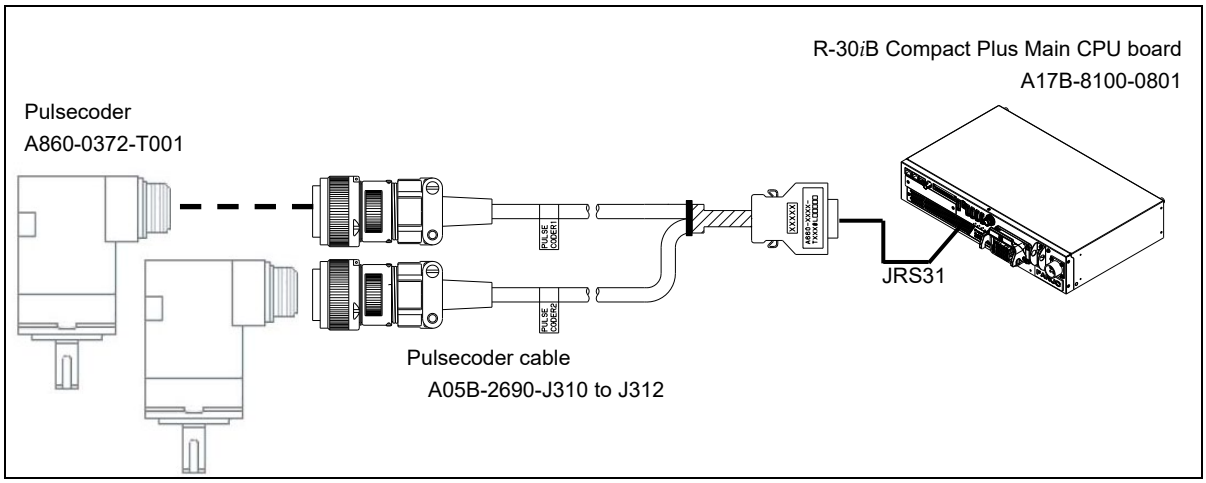


Fig. D.3.3 (c) Connecting cables with R-30*i*B Compact Plus Main CPU board (two αA1000S Pulsecoders)

D.3.4 For R-30*i*B Mini Plus

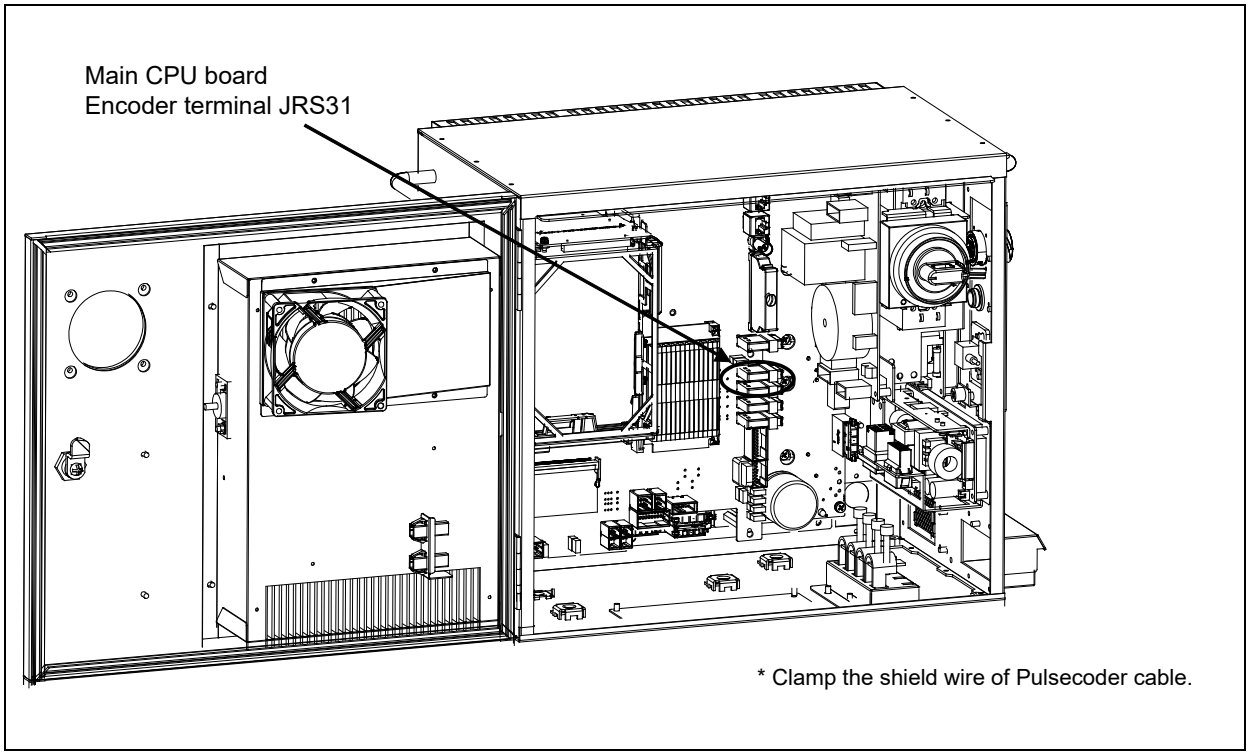


Fig. D.3.4 (a) R-30*i*B Mini Plus Controller connecting encoders (Pulsecoder)

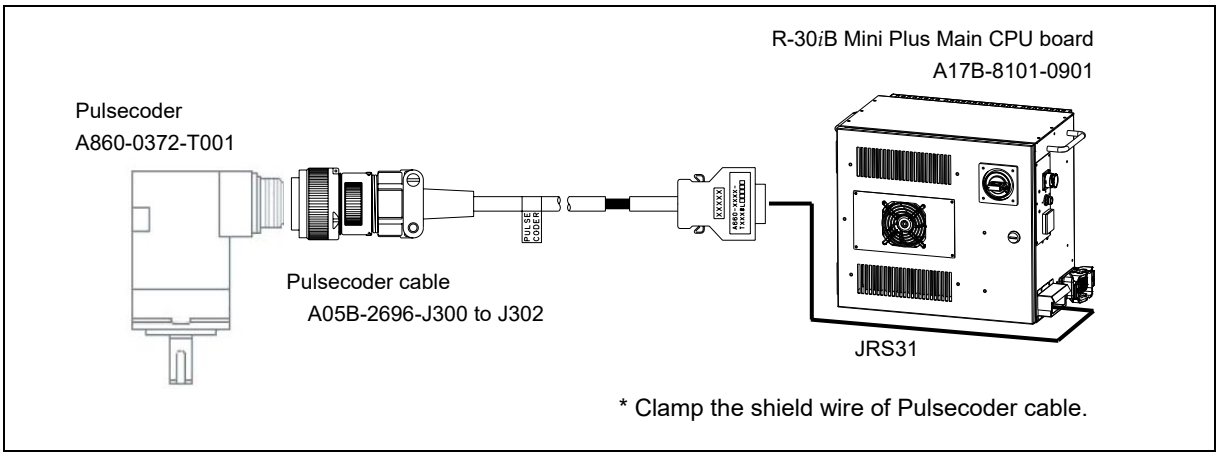


図 D.3.4 (b) Connecting cables with R-30*i*B Mini Plus (one αA1000S Pulsecoder)

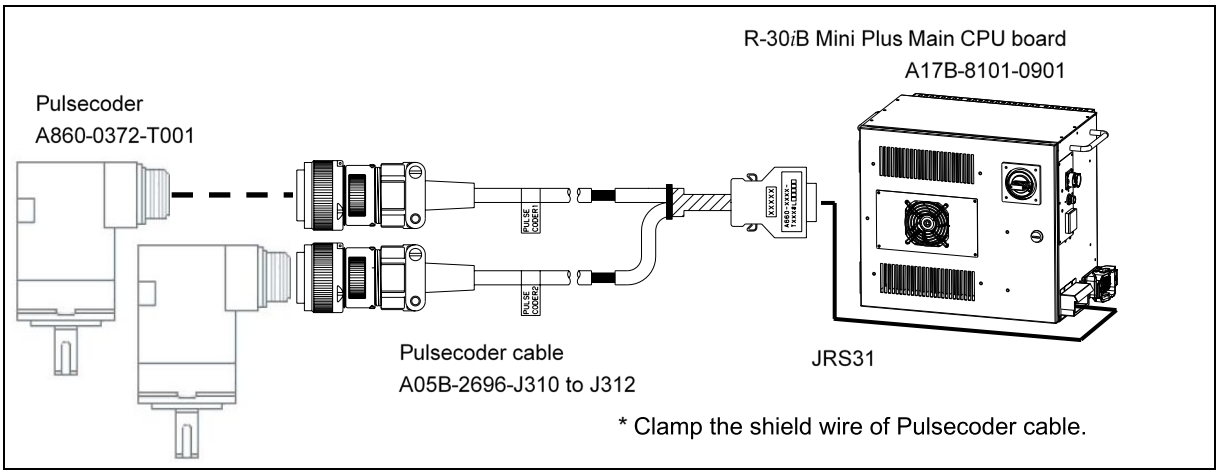


Fig. D.3.4 (c) Connecting cables with R-30*i*B Mini Plus (two αA1000S Pulsecoders)

E INCREMENTAL PULSECODER

This chapter contains the usage of Incremental Pulsecoder (A860-0301-T001 to T004).

> ⚠ **CAUTION**
> Incremental Pulsecoders cannot be connected to R-30*i*B Compact Plus or R-30*i*B Mini Plus.

E.1 REQUIREMENTS

Table E.1 (a) Requirements (R-30*i*B Plus: Incremental Pulsecoder A860-0301-T001 to T004)

Required Components	R-30*i*B Plus Controller A-Cabinet	R-30*i*B Plus Controller B-Cabinet	Comments
Hardware			
Line Tracking Interface Board	A20B-8101-0421 (wide-mini slot) or A20B-8101-0601 (mini slot)	A20B-8101-0421 (wide-mini slot) or A20B-8101-0601 (mini slot)	• The Line tracking interface board in the left column is included in the following order specification. A05B-2600-J035, A05B-2660-J035 (A/B-Cabinet, wide-mini slot) A05B-2600-J036, A05B-2660-J036 (A-Cabinet, mini slot) A05B-2600-J037 (B-Cabinet, mini slot) • Separate Detector Unit (SDU) - A02B-0323-C205 can be used in place of Line tracking Interface board. NOTE: SDU requires retrofit work to mount in the container. (See APPENDIX F “SEPARATE DETECTOR UNIT (SDU)”.)
Fiber Optic (FSSB) Cable	A66L-6001-0023	A66L-6001-0023	• The Fiber Optic cable in the left column is included in the following order specification. A05B-2600-J035, A05B-2660-J035 (A/B-Cabinet, wide-mini slot) A05B-2600-J036, A05B-2660-J036 (A-Cabinet, mini slot) A05B-2600-J037 (B-Cabinet, mini slot)

Required Components	R-30*i*B Plus Controller A-Cabinet	R-30*i*B Plus Controller B-Cabinet	Comments
Hardware			
Pulsecoder Cable	for A20B-8101-0421 (wide-mini slot) : A05B-2601-J380 (7M) A05B-2601-J381 (14M) A05B-2601-J382 (20M) A05B-2601-J383 (30M) A05B-2660-J380 (7M) A05B-2660-J381 (14M) A05B-2660-J382 (20M) A05B-2660-J383 (30M)	for A20B-8101-0421 (wide-mini slot) : A05B-2603-J380 (7M) A05B-2603-J381 (14M) A05B-2603-J382 (20M) A05B-2603-J383 (30M)	• Incremental Pulsecoder A860-0301-T001 to T004 cannot be connected to encoder terminal on Main CPU board.
	for A20B-8101-0601 (mini slot) (one Pulsecoder) : A05B-2601-J370 (7M) A05B-2601-J371 (14M) A05B-2601-J372 (20M) A05B-2601-J373 (30M) A05B-2660-J370 (7M) A05B-2660-J371 (14M) A05B-2660-J372 (20M) A05B-2660-J373 (30M)	for A20B-8101-0601 (mini slot) (one Pulsecoder) : A05B-2603-J370 (7M) A05B-2603-J371 (14M) A05B-2603-J372 (20M) A05B-2603-J373 (30M)	
Multiplexer cable	for A20B-8101-0421 (wide-mini slot) : A05B-2601-J385 (7M) A05B-2601-J386 (14M) A05B-2601-J387 (20M) A05B-2601-J388 (30M) A05B-2660-J385 (7M) A05B-2660-J386 (14M) A05B-2660-J387 (20M) A05B-2660-J388 (30M)	for A20B-8101-0421 (wide-mini slot) : A05B-2603-J385 (7M) A05B-2603-J386 (14M) A05B-2603-J387 (20M) A05B-2603-J388 (30M)	• When using multiple robots to track parts on the conveyor, use a Multiplexer to Controller cable, or use Ethernet encoder function (A05B-2600-R762) (option).
	for A20B-8101-0601 (mini slot) : A05B-2601-J375 (7M) A05B-2601-J376 (14M) A05B-2601-J377 (20M) A05B-2601-J378 (30M) A05B-2660-J375 (7M) A05B-2660-J376 (14M) A05B-2660-J377 (20M) A05B-2660-J378 (30M)	for A20B-8101-0601 (mini slot) : A05B-2603-J375 (7M) A05B-2603-J376 (14M) A05B-2603-J377 (20M) A05B-2603-J378 (30M)	

Table E.1 (b) Requirements (R-30*i*B Mate Plus: Incremental Pulsecoder A860-0301-T001 to T004)

Required Components	R-30*i*B Mate Plus Controller	R-30*i*B Mate Plus Controller (Open Air)	Comments
Hardware			
Line Tracking Interface Board	A20B-8101-0601 (mini slot)	A20B-8101-0601 (mini slot)	• The Line tracking interface board in the left column is included in the following order specification. A05B-2650-J035, A05B-2661-J035 (R-30*i*B Mate Plus, mini slot) A05B-2655-J035, A05B-2662-J035 (R-30*i*B Mate Plus (Open Air), mini slot)
Fiber Optic (FSSB) Cable	A66L-6001-0026	A66L-6001-0023	• The Fiber Optic cable in the left column is included in the following order specification. A05B-2650-J035, A05B-2661-J035 (R-30*i*B Mate Plus, mini slot) A05B-2655-J035, A05B-2662-J035 (R-30*i*B Mate Plus (Open Air), mini slot)
Pulsecoder Cable	for A20B-8101-0601 (mini slot) (one Pulsecoder) : A05B-2650-J200 (7M) A05B-2650-J201 (14M) A05B-2650-J202 (20M) A05B-2661-J200 (7M) A05B-2661-J201 (14M) A05B-2661-J202 (20M)	for A20B-8101-0601 (mini slot) (one Pulsecoder) : A05B-2655-J200 (7M) A05B-2655-J201 (14M) A05B-2655-J202 (20M) A05B-2662-J200 (7M) A05B-2662-J201 (14M) A05B-2662-J202 (20M)	• Incremental Pulsecoder A860-0301-T001 to T004 cannot be connected to encoder terminal on Main CPU board.
	for A20B-8101-0601 (mini slot) (two Pulsecoders) : A05B-2650-J210 (7M) A05B-2650-J211 (14M) A05B-2650-J212 (20M) A05B-2661-J210 (7M) A05B-2661-J211 (14M) A05B-2661-J212 (20M)	A20B-8101-0601 (mini slot) (two Pulsecoders) : A05B-2655-J210 (7M) A05B-2655-J211 (14M) A05B-2655-J212 (20M) A05B-2662-J210 (7M) A05B-2662-J211 (14M) A05B-2662-J212 (20M)	
Multiplexer Cable	for A20B-8101-0601 (mini slot) (one Pulsecoder) : A05B-2650-J225 (7M) A05B-2650-J226 (14M) A05B-2650-J227 (20M) A05B-2650-J228 (30M)	-	• When using multiple robots to track parts on the conveyor, use a Multiplexer to Controller cable, or use Ethernet encoder function (A05B-2600-R762) (option).
	for A20B-8101-0601 (mini slot) (two Pulsecoders) : A05B-2650-J230 (7M) A05B-2650-J231 (14M) A05B-2650-J232 (20M) A05B-2650-J233 (30M)	-	

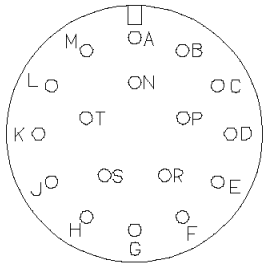
Table E.1 (c) Requirements (R-30*i*B Plus / R-30*i*B Mate Plus: Multiplexer A16B-1212-0290)

Required Components	R-30*i*B Plus / R-30*i*B Mate Plus Controller	Comments
Hardware		
Multiplexer	A16B-1212-0290	• When using multiple robots to track parts on the conveyor, use a Multiplexer to Controller cable, or use Ethernet encoder function (A05B-2600-R762) (option).
Pulsecoder to Multiplexer Cable	A05B-2451-K102(7M) A05B-2451-K103(14M)	
Multiplexer to Multiplexer Cable	A05B-2600-K155 (7M) A05B-2600-K156(14M) A05B-2660-K155 (7M) A05B-2660-K156(14M)	• An investigation is necessary if three or more Multiplexers are connected in series.
5V Multiplexer Connector	A02B-0061-K203	• One housing and six contacts (soldering type) are included.
5V DC power supply (rated power 10W or more) is necessary when a Multiplexer is used.		

Refer to Table E.2 (a) and Fig. E.2 (a) to Fig. E.2 (c) for Pulsecoder signal information, and images containing the dimensions of the Pulsecoders.
Refer to Table E.2 (b) and Fig. E.2 (c) for Multiplexer specifications, and images containing the dimensions of the multiplexers.
Refer to Fig. E.3.1 (a) to Fig. E.3.1 (d) and Fig. E.3.2 (a) to Fig.3.2 (c) for information on connections and installation of the cables.

E.2 FIGURES

A860-0301-T001



3102A20-29P

Signal Name	Pin No. 310A20-29P
A	A
*A	D
B	B
*B	E
Z	F
*Z	G
C1	-
C2	-
C4	-
C8	-
+5V	C,J,K
0V	N,P,T
Shield	H
OH1	
OH2	
REQ	
+6VA	
0VA	

Fig. E.2 (a) Incremental Pulsecoder (A860-0301-T001) connection signal information

Table E.2 (a) Incremental Pulsecoder (A860-0301-T001) specifications

Item		Specification
Power voltage		5 (V) 5%
Current consumption		Up to 0.35 (A)
Working temperature range		0 to 60 (℃)
Maximum response frequency		100 x 10^3(Hz)
Input shaft inertia		Up to 5 x 10^{-3}(kg • m^2)
Input shaft startup torque		Up to 0.8 (Nm)
Rated loads	Radial	20 (N)
	Axial	10 (N)
Shaft diameter runout		0.02 x 10^{-3} (m)
Weight		Approx. 2.0 (kg)

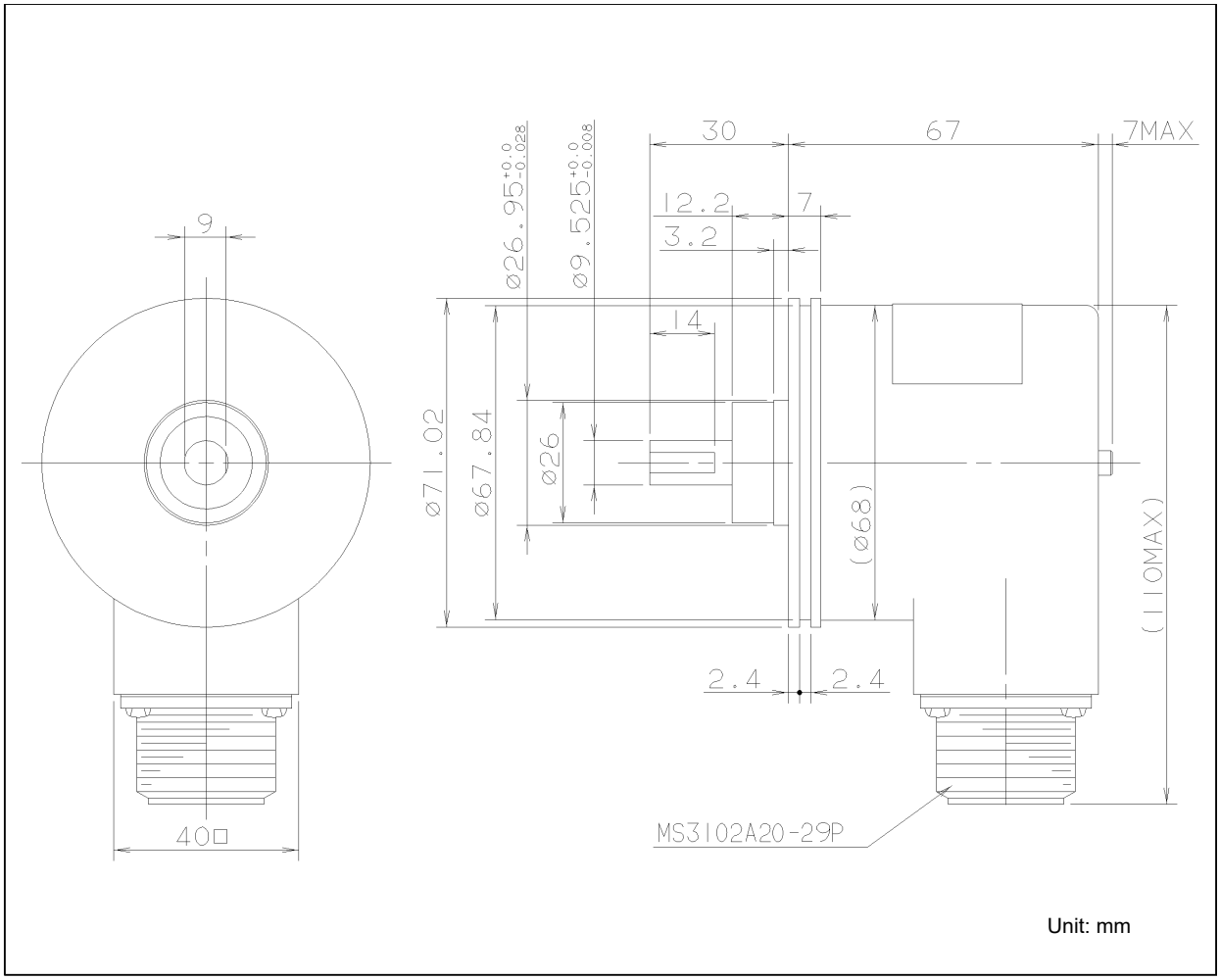


Fig. E.2 (b) Incremental Pulsecoder dimensions (A860-0301-T001)

Table E.2 (b) Installation conditions of multiplexer (A16B-1212-0290)

Item	Condition
Permissible ambient temperature	Operating 0°C to 55°C Storage, Transport -20°C to +60°C
Permissible ambient humidity	95%RH or less, no condensation
Vibration acceleration	4.9m/s^2 (0.5G) or less.
Atmosphere	Multiplexers must be installed in a cabinet to keep from conductive dust, cutting fluid and organic solvent.

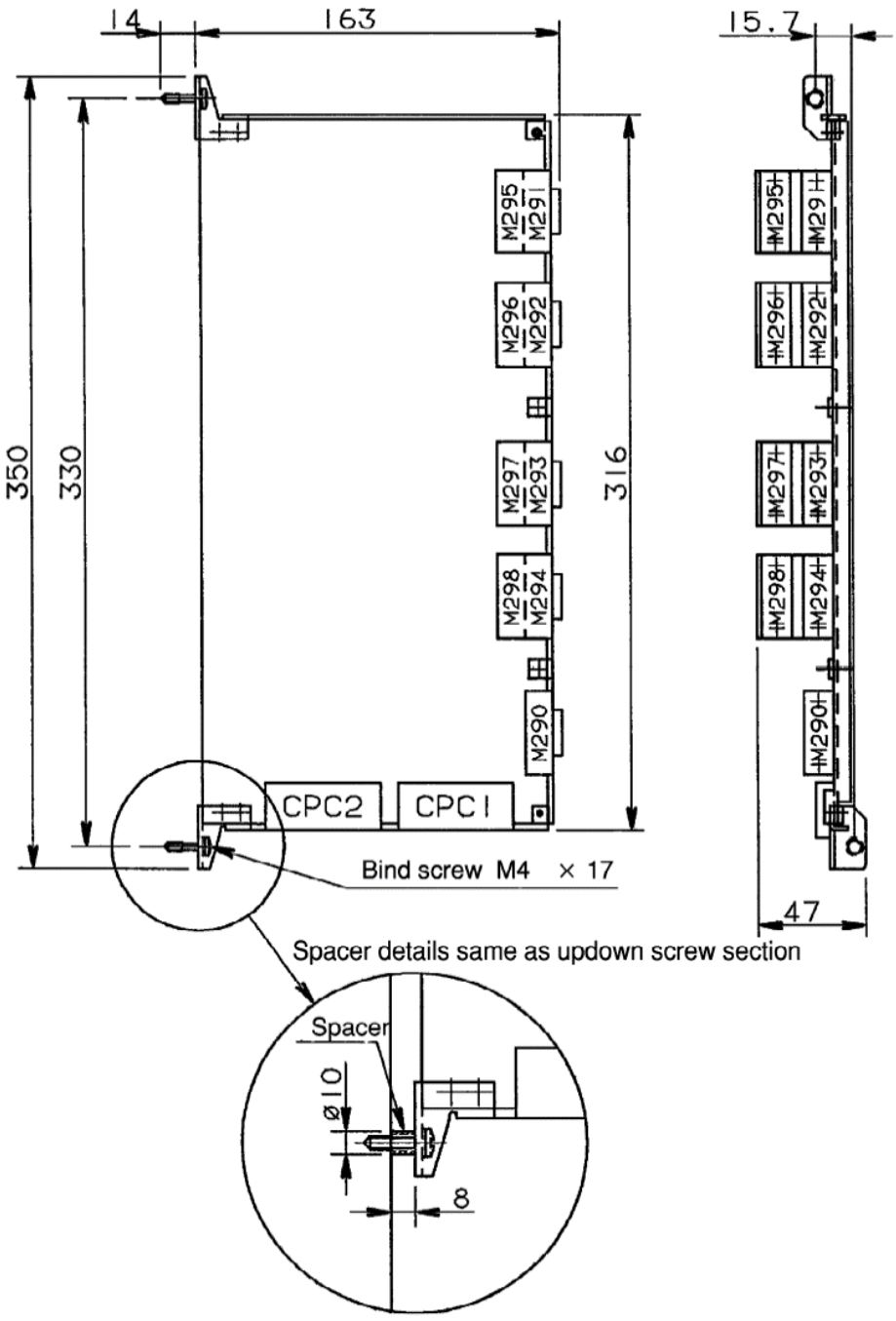


Fig. E.2 (c) Multiplexer dimensions (A16B-1212-0290)

E.3 HOW TO CONNECT

E.3.1 Connecting to Robot Controller Directly

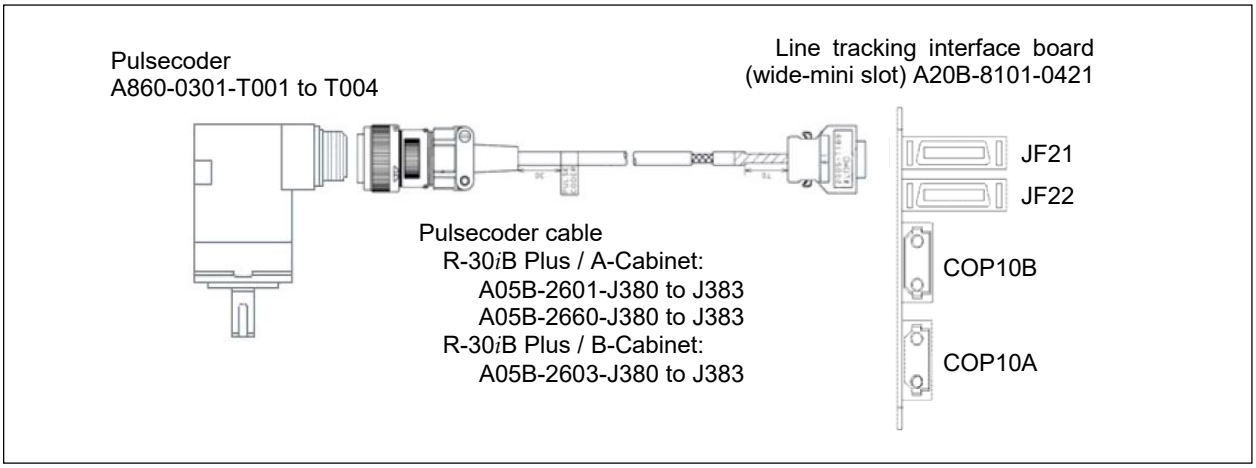


Fig. E.3.1 (a) Connecting cables with Line tracking interface board A05B-2600-J035, A05B-2660-J035 (one Pulsecoder)

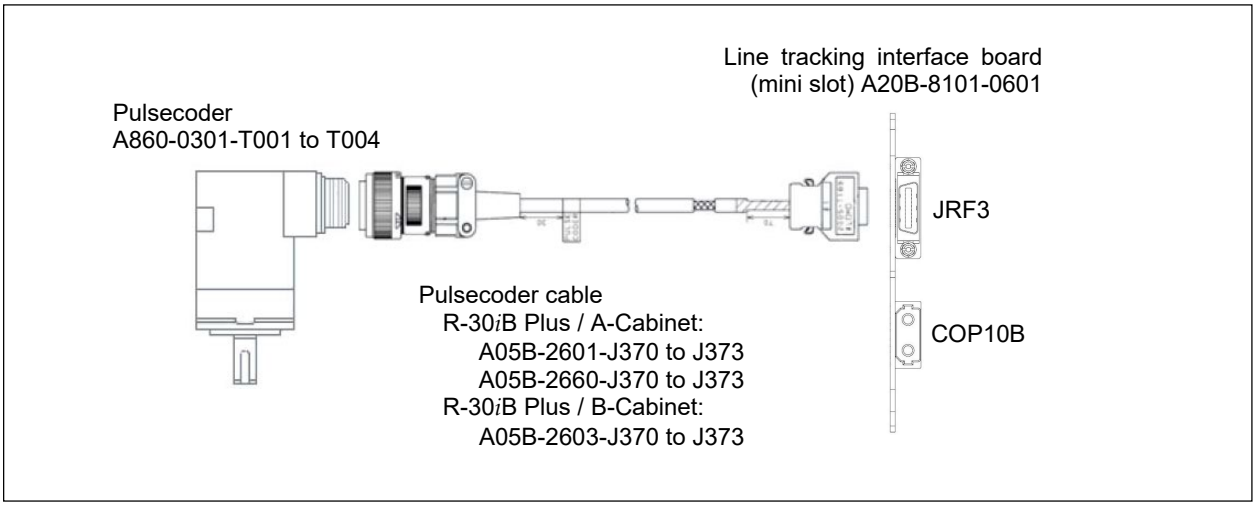


Fig. E.3.1 (b) Connecting cables with Line tracking interface board A05B-2600-J036, A05B-2660-J036 or A05B-2600-J037 (one Pulsecoder)

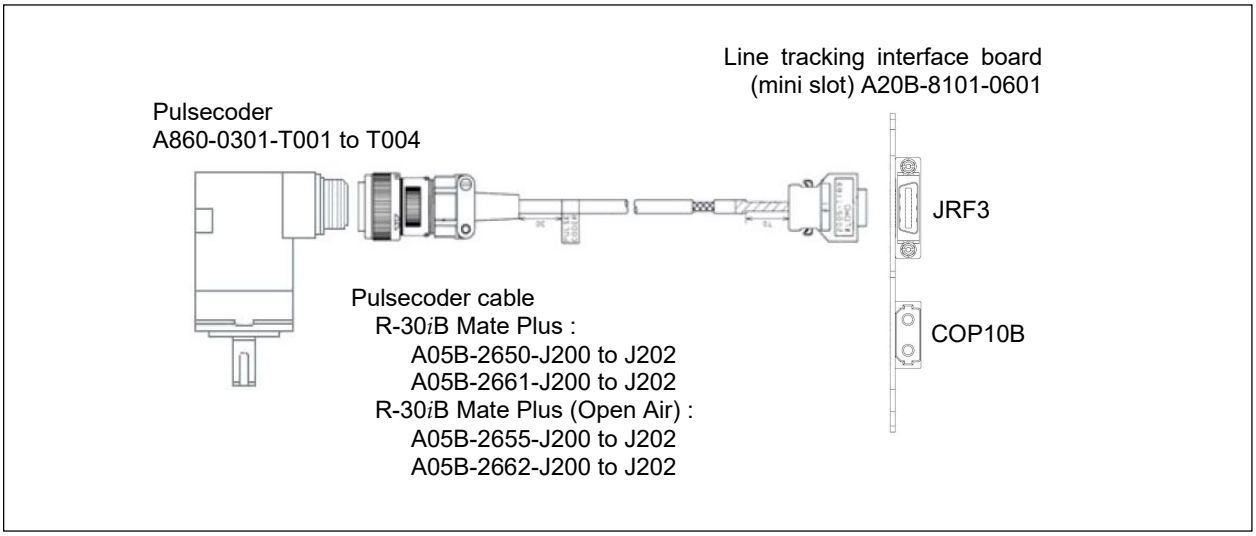


Fig. E.3.1 (c) Connecting cables with Line tracking interface board A05B-2650-J035, A05B-2655-J035, A05B-2661-J035 or A05B-2662-J035 (one Pulsecoder)

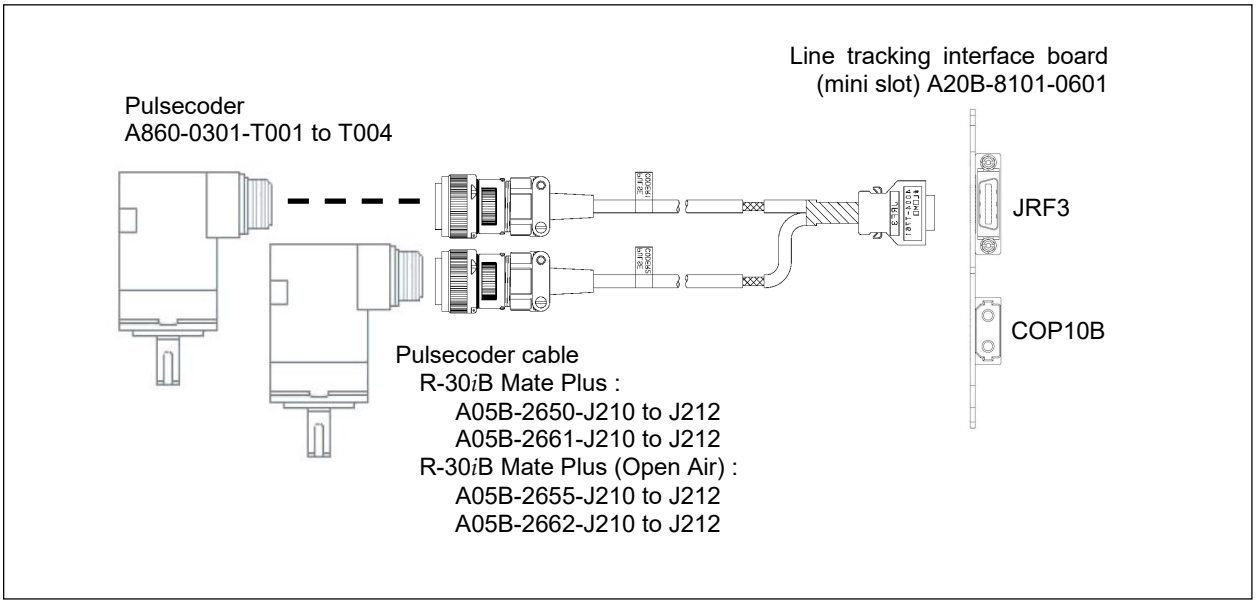


Fig. E.3.1 (d) Connecting cables with Line tracking interface board A05B-2650-J035, A05B-2655-J035, A05B-2661-J035 or A05B-2662-J035 (two Pulsecoders)

E.3.2 Connecting to Pulse Multiplexer

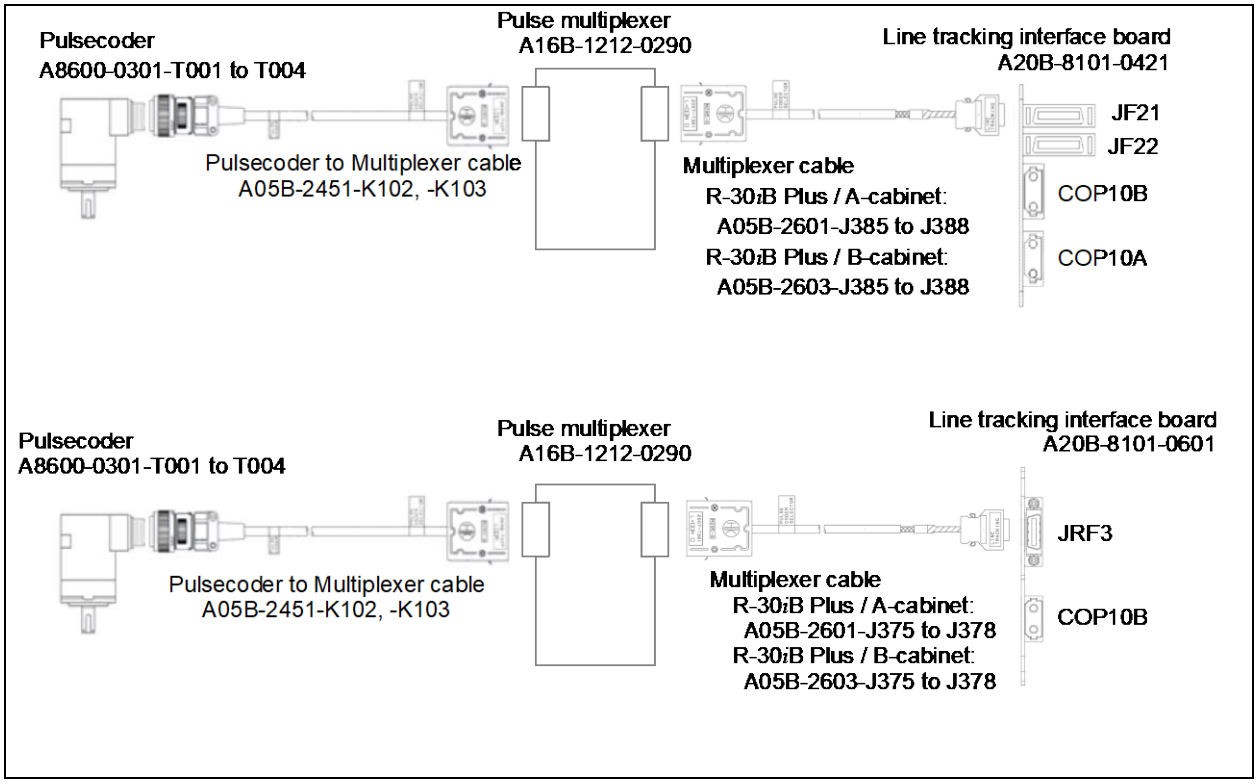


Fig. E.3.2 (a) Pulse multiplexer and connecting cables 1 (Pulsecoder, A860-0301-T001 to T004)

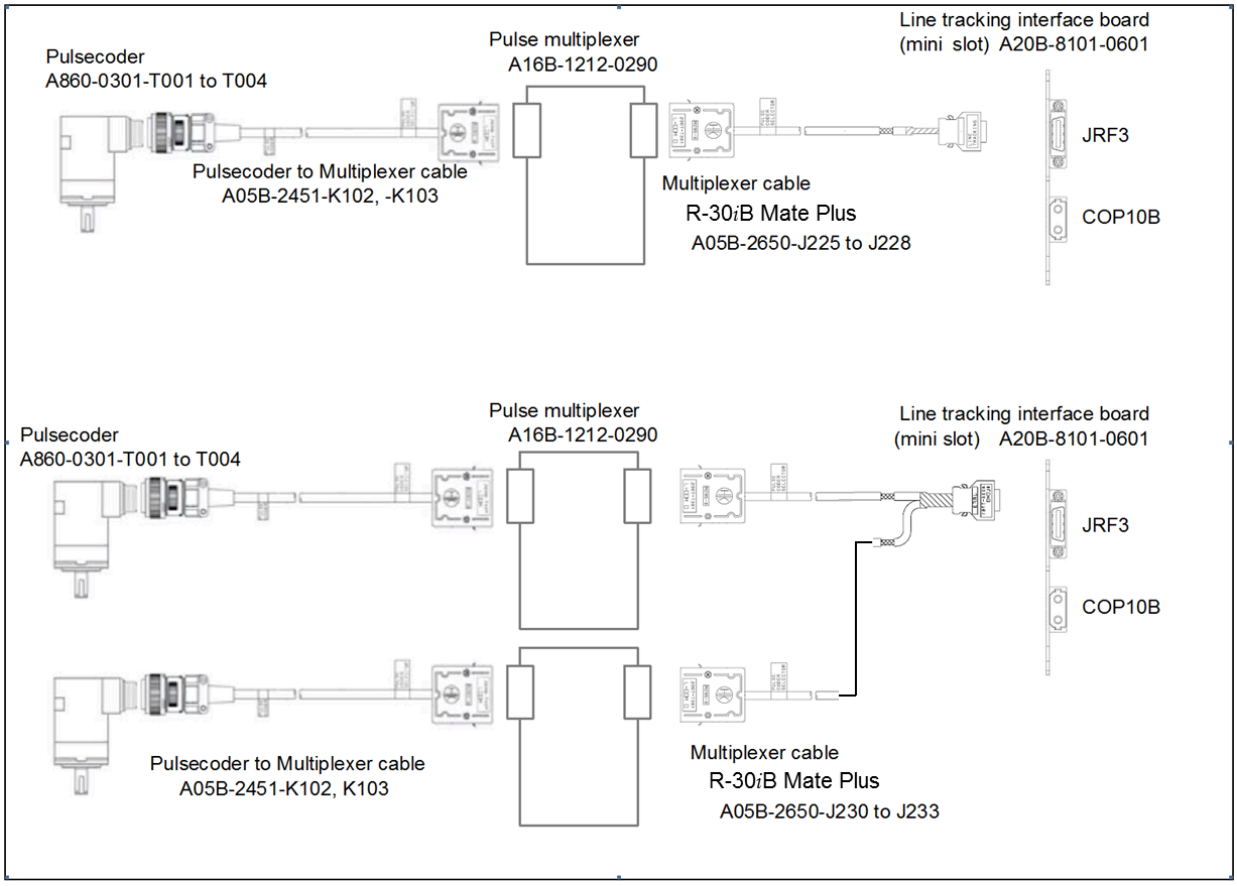


Fig. E.3.2 (b) Pulse multiplexer and connecting cables 2 (Pulsecoder, A860-0301-T001 to T004)

NOTE
If the line tracking interface board cannot be used or is not available, you can use the SDU shown in Fig. F (a) to Fig. F (i).

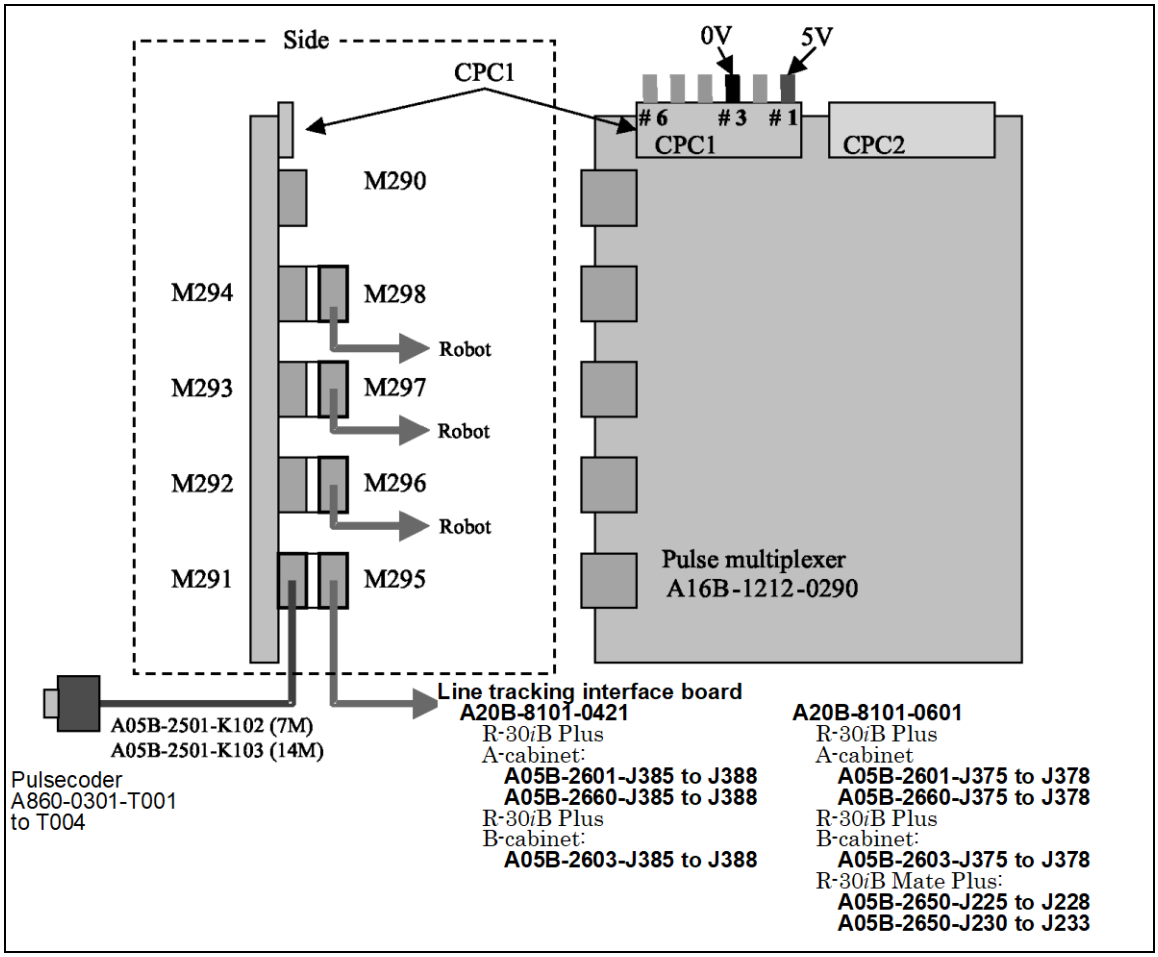


Fig. E.3.2 (c) Connecting cables to the multiplexer 3 (Pulsecoder, A860-0301-T001 to T004)

NOTE
You can connect up to four cables (pulse multiplexer to / from controller) to one pulse multiplexer.

F SEPARATE DETECTOR UNIT (SDU)

If the line tracking interface board and main board cannot be used or are not available, such as when more than 4 Pulsecoders are connected to only one controller, you can use the SDU. Because SDU requires retrofit work to mount in the container, please refer Fig. F(a) to Fig. F(i).

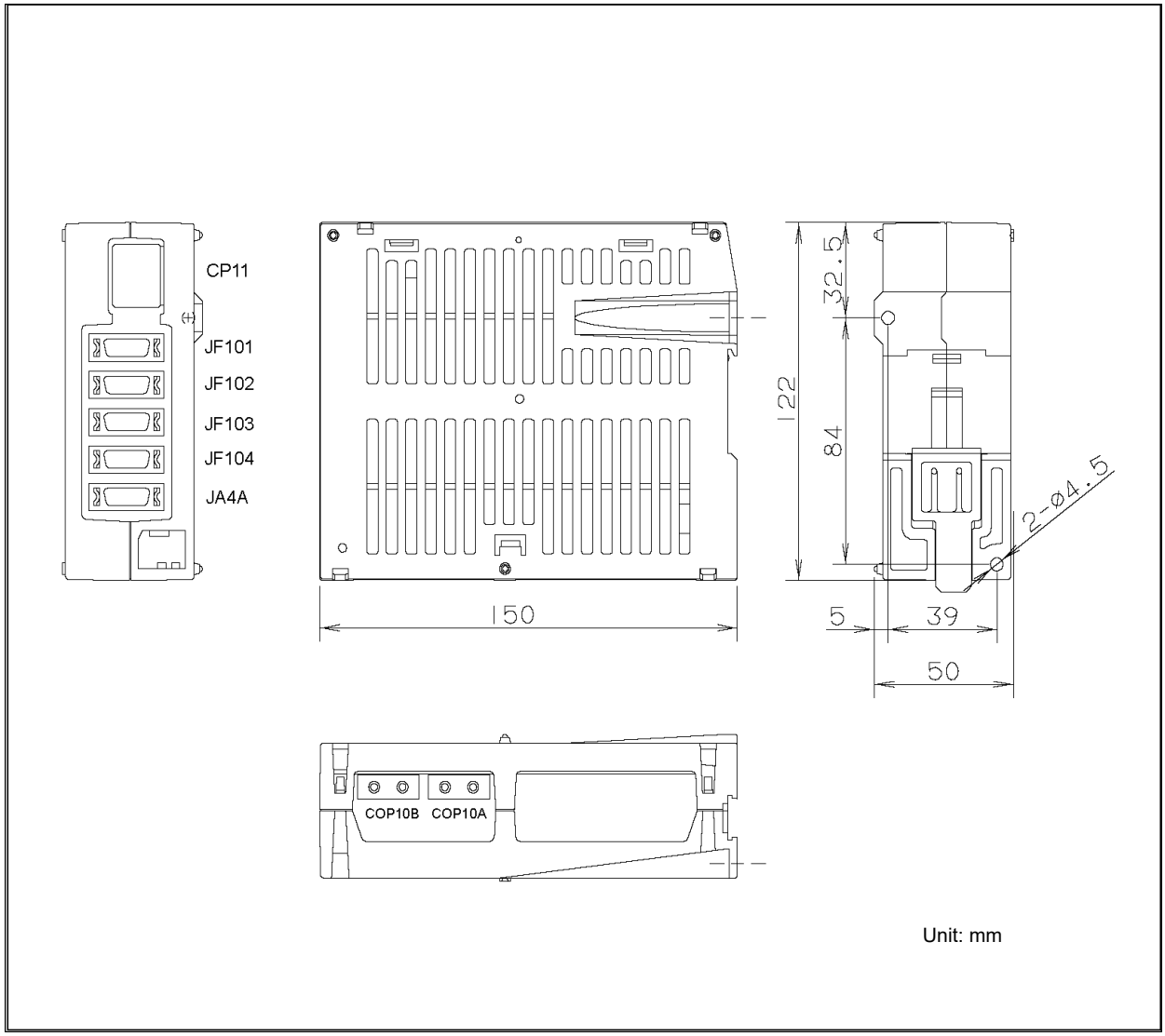


Fig. F (a) External dimensions of separate detector unit (SDU)

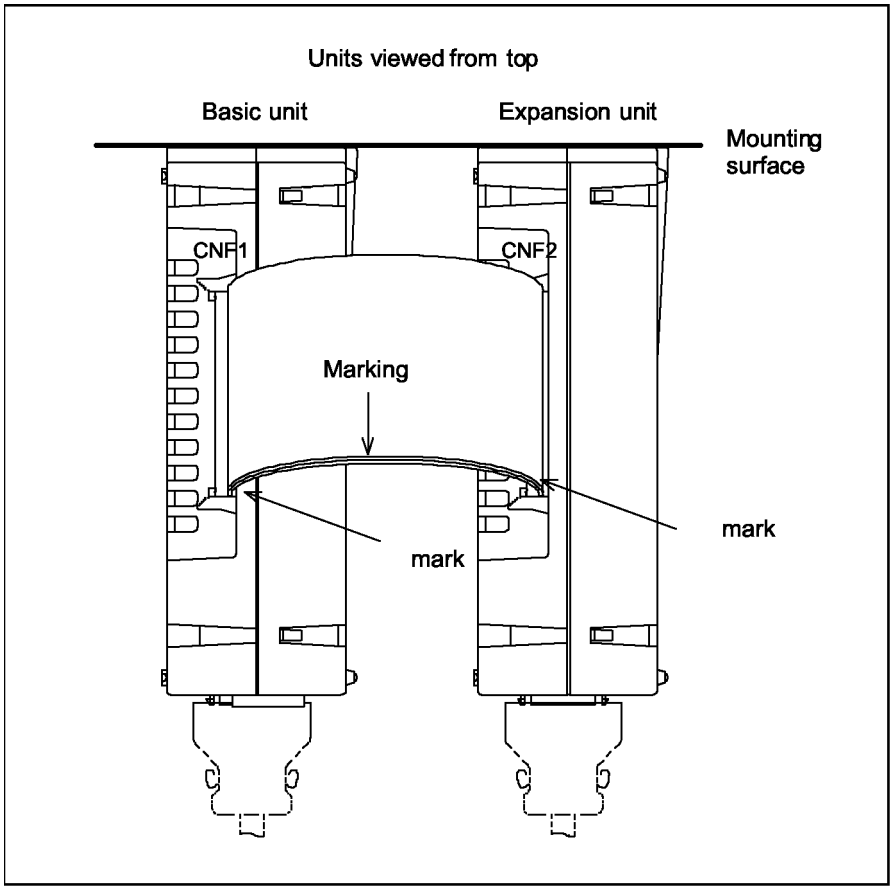


Fig. F (b) Cable connection between basic unit and expansion unit

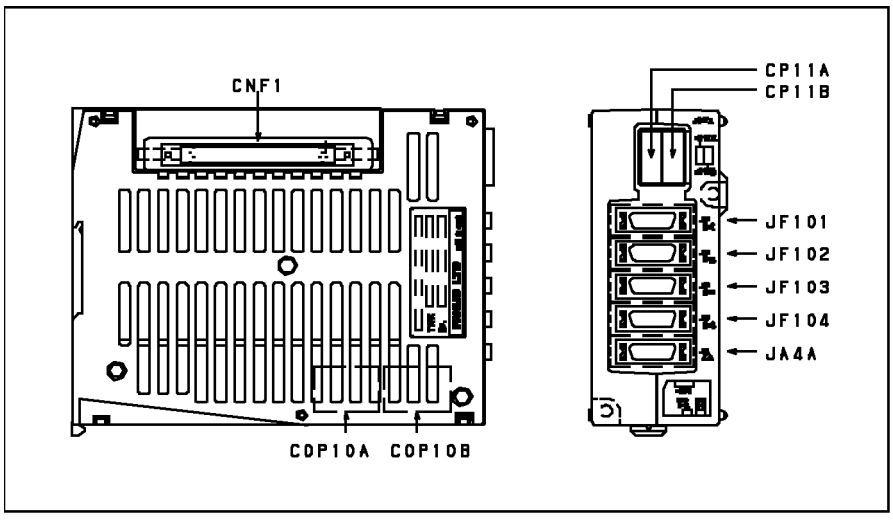


Fig. F (c) Connector locations on the basic unit

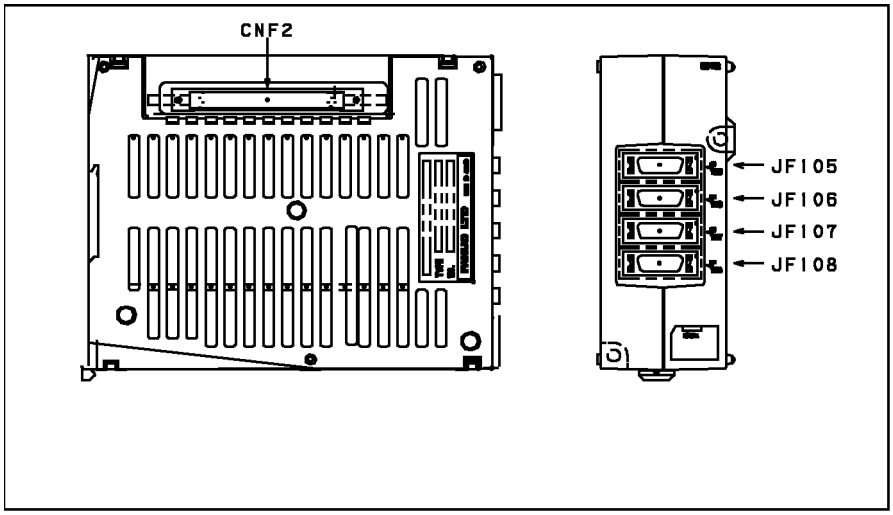


Fig. F (d) Connector locations on the expansion unit

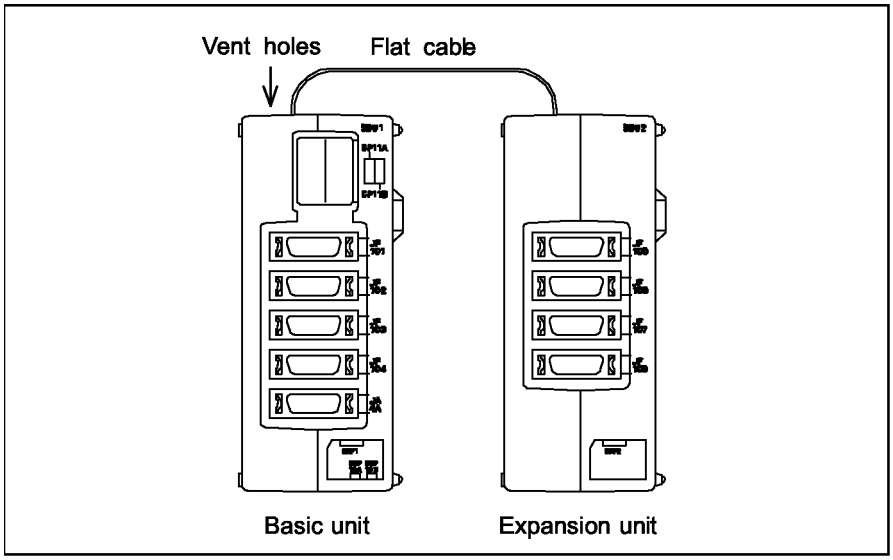


Fig. F (e) Flat cable placement during installation

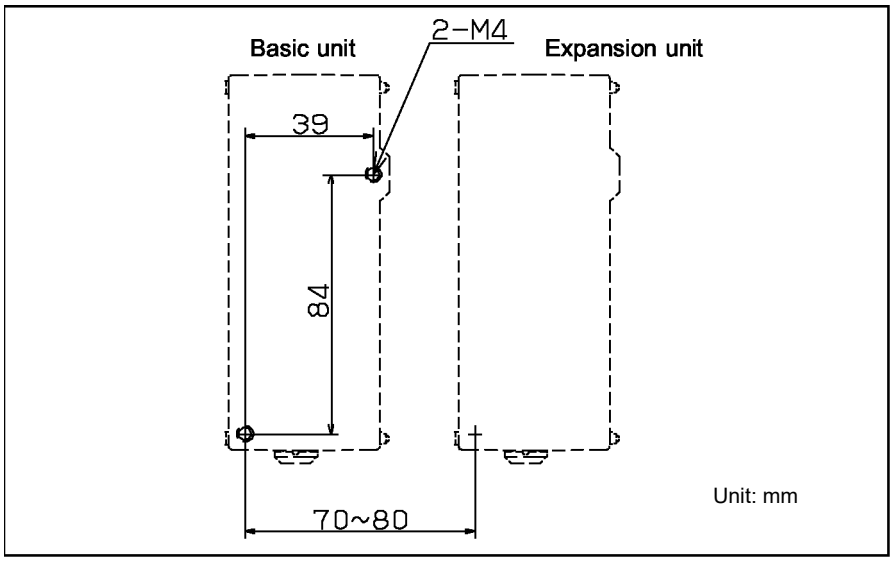
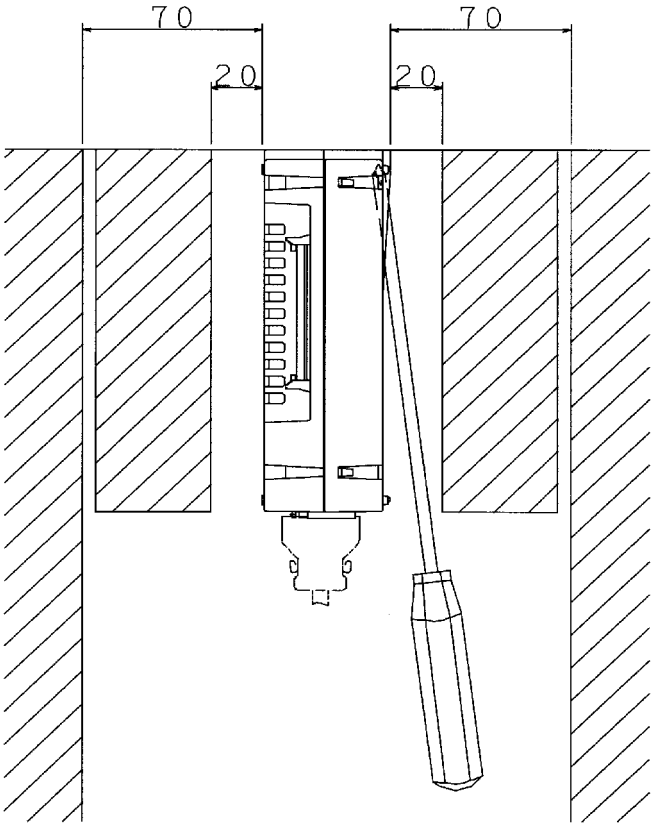


Fig. F (f) Horizontal separation of mounting holes during installation

⚠CAUTION
To install or remove the unit, you must insert a screwdriver obliquely. Therefore, you must have sufficient access clearance on both sides of the units. As a general guideline, if the front of an adjacent unit appears flush with the unit or slightly set back, allow a clearance of about 20 mm between the two units. If the front of an adjacent unit protrudes beyond the front of the unit, allow a clearance of about 70 mm between the two units. Also, when you are installing the unit near the side of a cabinet, you must allow a clearance of about 70 mm between the unit and the side of the cabinet.



Access clearance near a separate detector interface unit

Fig. F (g) Accessing the unit

⚠ CAUTION
When you are removing the unit, be careful not to damage the lock by applying excessive force. When you are installing and removing the unit, hold the upper and lower ends of the unit so that stress is not applied to the side of the unit (the surface with slits).

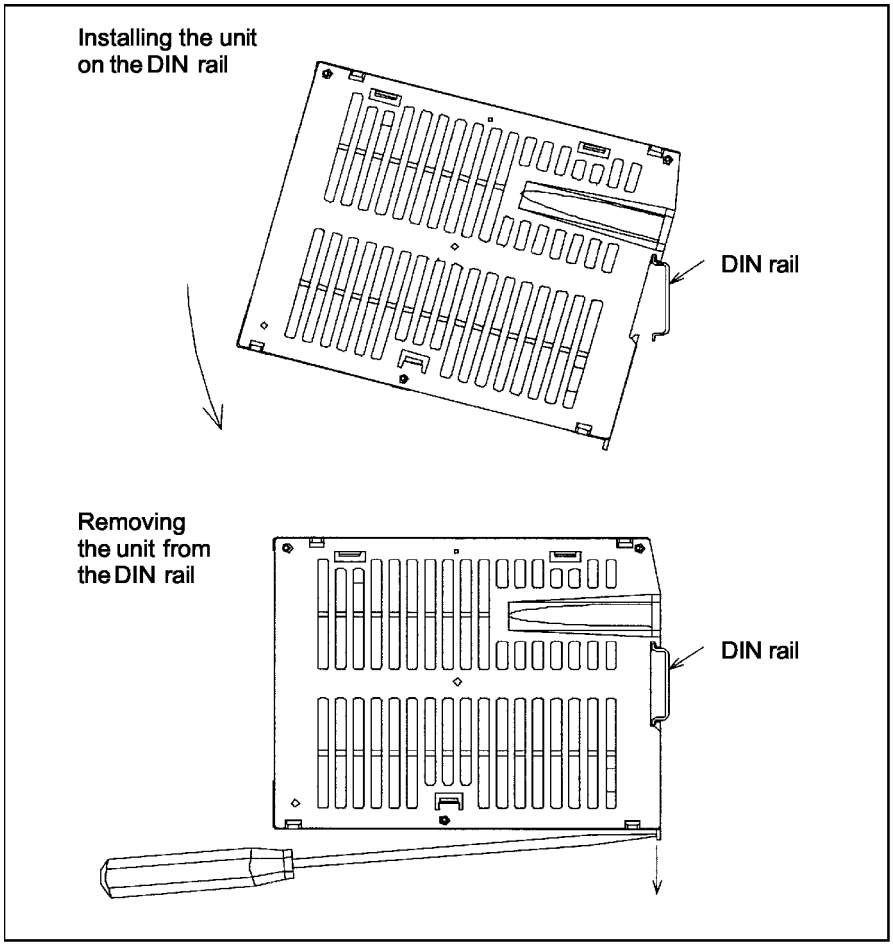


Fig. F (h) Installing and removing the unit

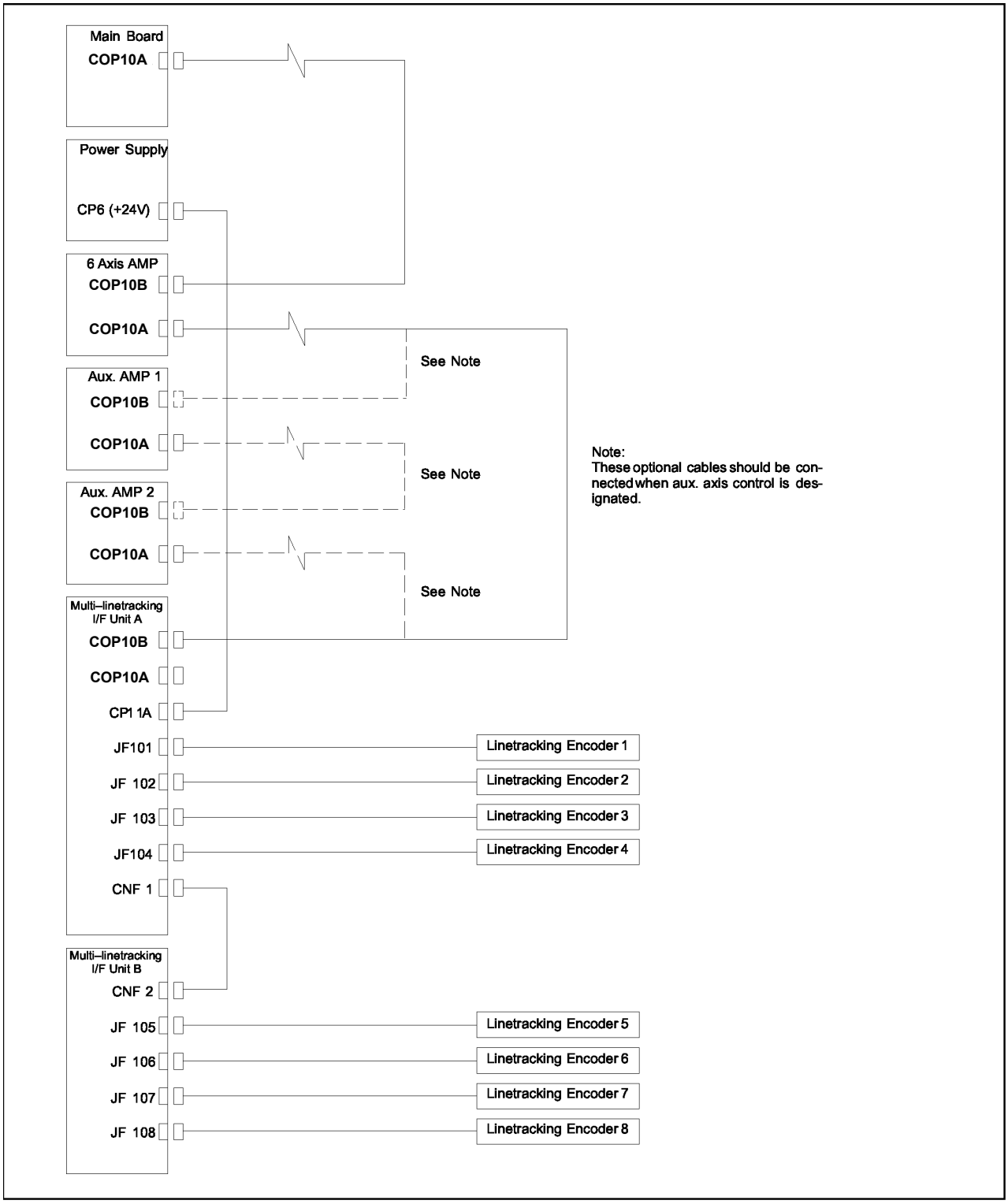


Fig. F (i) Connection diagram

G KAREL PROGRAM LIST

This is a correspondence table of KAREL programs of the old visual tracking software on and before 7DC1 series and *i*RPickTool.

No.	Old KAREL	Description	Argument	New KAREL	Argument	Comment
1	VSTKCLRQ.PC	Clear a queue.	1: Workarea name	PKCSCLRQUE.PC	1: Conveyor Station ID	ID is gotten by PKCSGETID.
2	VSTKGETQ.PC	Get a part from a queue.	1: Workarea name 2: Vision register number to output part 3: Time to wait for part 4: Register number for status 5: Flag to handle part consecutively 6(Optional): Model ID	PKCSGETQUE.PC	1: Conveyor Station ID 2: Flag to handle part consecutively 3: Time to wait for part 4: Vision register number to output part 5: Register number for status 6(Optional): Model ID	ID is gotten by PKCSGETID. The order of argument is changed.
3	VSTKACKQ.PC	Return acknowledge for the part gotten from the queue.	1: Workarea name 2: Vision register number specified in GETQ 3: Processing result	PKCSACKQUE.PC	1: Conveyor Station ID 2: Processing result	You need not to specify vision register number as argument.
4	VSTKSTRT.PC	Start sensor.	None	PKSNSTART.PC	None	
5	VSTKSTOP.PC	Stop sensor.	None	PKSNSTOP.PC	None	
6	VSTKSTLB.PC	Change load balance.	1: Line name 2: Register number to specify load balance 3: Number of workarea 4: Bypass 5(Optional): Model ID	PKCVSETLBD.PC	1: Conveyor name 2: Register number to specify load balance 3: Number of conveyor station 4: Bypass 5(Optional): Model ID	
7	VSTKENLB.PC	Switch load balance enabled/disabled.	1: Line name 2: Enable/Disable	PKCVENBLB.PC	1: Conveyor name 2: Enable/Disable	
8	VSTKENSC.PC	Switch stop conveyor enabled/disabled.	1: Line name 2: Register number to specify enable/disable 3: Number of workarea	PKCVENBSC.PC	1: Conveyor name 2: Register number to specify enable/disable 3: Number of conveyor station	
9	VSTKSTVN.PC	Change vision program name.	1: Sensor number 2: Vision process name	PKSNSETVPNAM.PC	1: Sensor name 2: Vision process name	
10	VSTKSTTN.PC	Change tray name.	1: Sensor number 2: Tray name	PKCVSETTRNAM.PC	1: Conveyor name 2: Tray name	

No.	Old KAREL	Description	Argument	New KAREL	Argument	Comment
11	VSTKGETT.PC	Output estimated time for part to arrive the upstream boundary.	1: Workarea name 2: Register number to output estimated time 3: Flag to handle part consecutively 4(Optional): Model ID	PKCSGETTIME.PC	1: Conveyor Station ID 2: Order 3: Flag to handle part consecutively. 4: Quantity of pick 5: Register number to output estimated time 6: Vision register number to output part 7: Register number for status 8: (Optional): Model ID	ID is gotten by PKCSGETID. Though some arguments are added, what we can do increase by them.
12	VSTKPFRT.PC	Return the prediction number of part to be processed per minute immediately after PKCSGETPFRT.	1: Workarea name 2: Register number to output prediction pick rate	PKCSGETPFRT.PC	1: Conveyor Station ID 2: Register number to output prediction pick rate	ID is gotten by PKCSGETID.
13	VSTKNPRT.PC	Return the number of part in a queue.	1: Workarea name 2: Register number to output number of part in queue	PKCSGETNPRT.PC	1: Conveyor Station ID 2: Register number to output number of part in queue	ID is gotten by PKCSGETID.

H SENSOR CUSTOMIZATION

H.1 OVERVIEW

Sensor takes the following role in the work cell of *i*RPickTool.

- A sensor monitors a conveyor movement and an external signal like DI, RI or HDI. And when if a specified condition is met, the sensor executes an action like execution of vision program and puts information of found parts to a queue.

Application engineers can customize this action.

For example, they can add their original processes peculiar to their application in between finding parts and putting information of them to queues by customizing sensor.

Here, the procedure of customizing a sensor and the essential points of the customization for the action are explained.

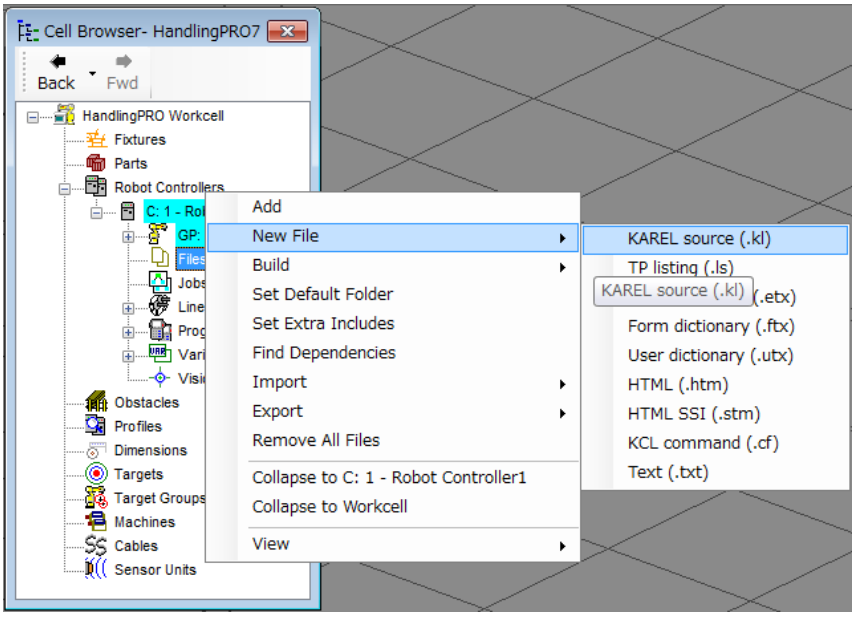
> ⚠ **CAUTION**
> The customer is responsible for the use of any customized programs. Be sure to perform sufficient operation checks on the actual machine before using customized programs.

H.2 PROCEDURE TO CUSTOMIZE A SENSOR TASK

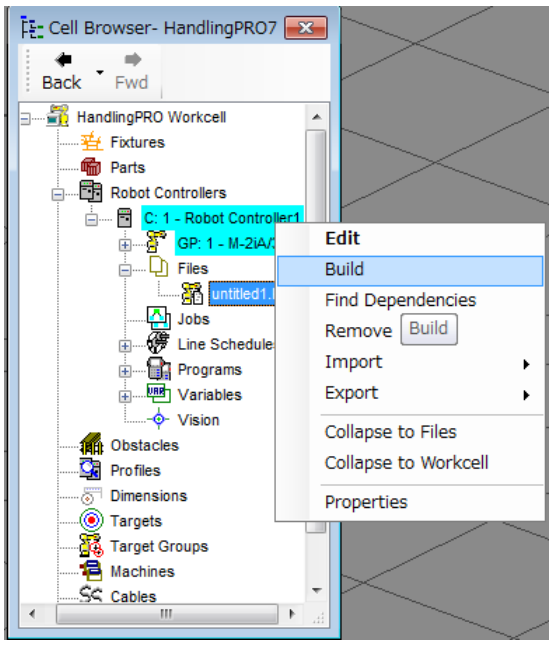
Sensor should be customized using ROBOGUIDE. The following steps are performed in ROBOGUIDE.

1 Make a work cell in ROBOGUIDE. You have to order the following option at least when you create a virtual robot.
- *i*RPickTool Basic (J944)

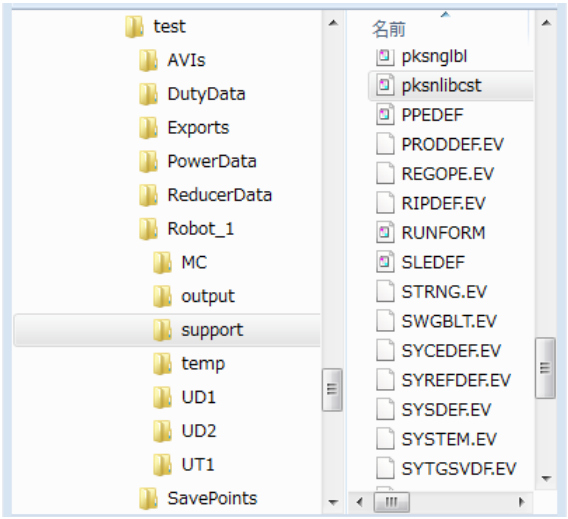
2 Right click [Files] under Cell Browser and select [New File] → [KAREL source (.kl)] from the displayed menu.



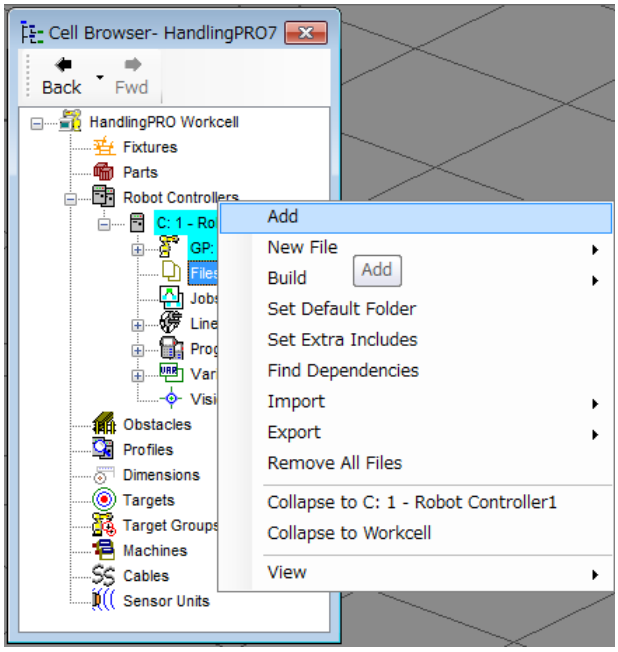
3 Build the created KAREL source without editing anything.
"File cannot be opened or created. …" might be displayed on another window, but the message can be ignored.



The [support] folder which includes the sample KAREL program (pksnlibcst.kl) is created in Robot_1 folder of the work cell as blow.



4 Add pksnlibcst.kl to Cell Brower as below.



5 Customize pksnlibcst.kl.

> ⚠ **CAUTION**
> You cannot change the program name and the function name defined in this KAREL program.

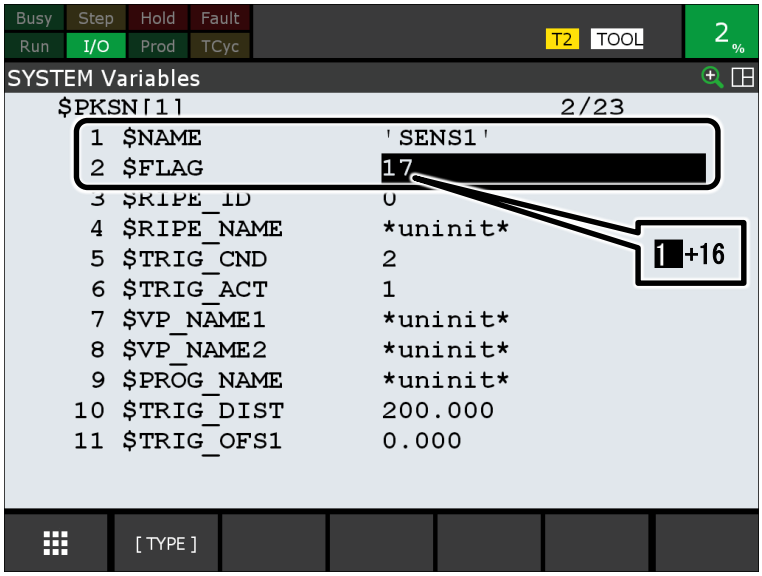
6 Build the customized pksnlibcst.kl.

The above-mentioned steps are performed in ROBOGUIDE.
Here, all of following steps are performed in a real robot controller.

7 Load the built pksnlibcst(.PC) to the real controller from the FILE menu of TP during the cold start mode.

NOTE
The built KAREL sources loaded from the FILE menu during the cold start mode can be saved by “All of above” backup.

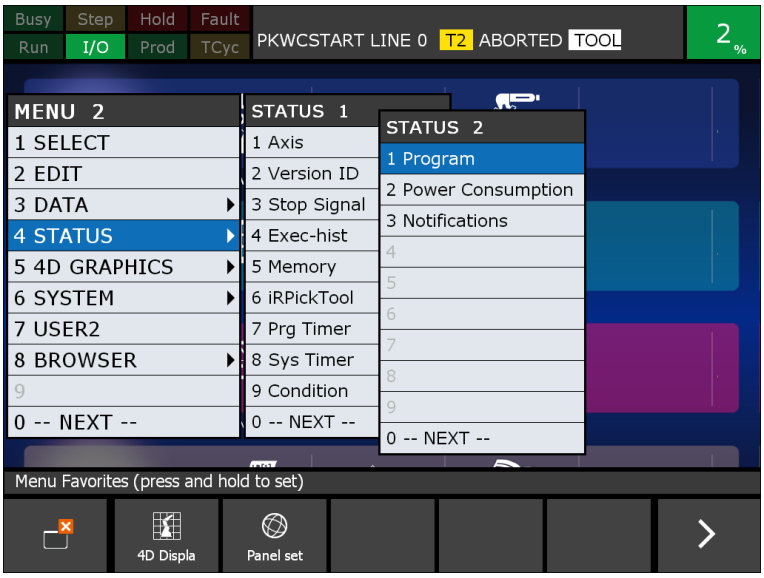
8 Perform the following settings in order to execute the customized program as an action.
If you log in the *i*RPickTool setup page, please log out it before the following procedure.
If you do not create sensor object in *i*RPickTool setup page, please create it before the following procedure. You set BIT4(=16) in $PKSN[n].$FLAG(system variable). Probably you have only to add 16 to $PKSN[n].$FLAG because BIT4(=16) is not set in $PKSN[n].$FLAG as default. Regarding the value of “n”, you can decide it as below. You search the sensor name like [SENS1], which you want to customize, from the element of $PKSN[n] array. The element is $PKSN[n].$NAME.



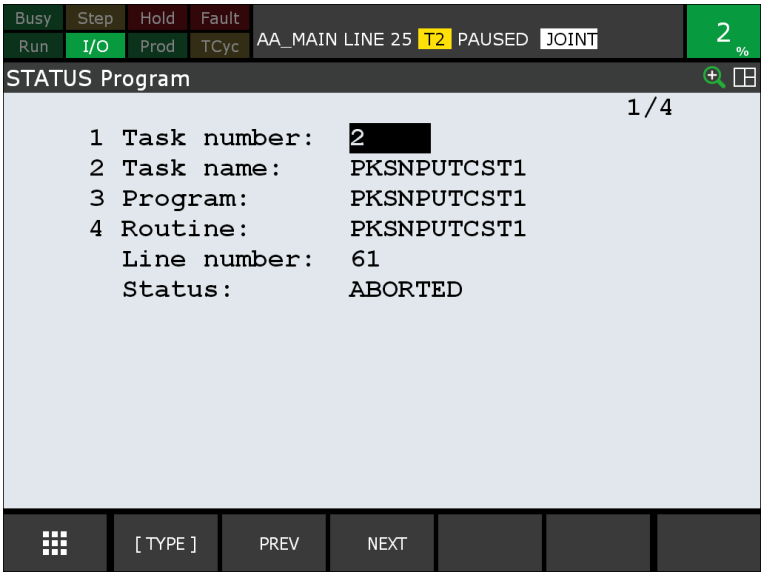
⚠ CAUTION
If you set the above bit with keeping logging in *i*RPickTool setup page, the bit can be reset when you change the sensor parameter in the setup page. If the bit is reset, please set the bit again after you log out the setup page.

9 Start/Stop sensor by PKWCSTART/PKWCEND or PKSNSTART/PKSNSTOP in TP program as usual. The customized action is executed in the sensor where the bit is set. You can confirm whether the customized action is executed or not as below.

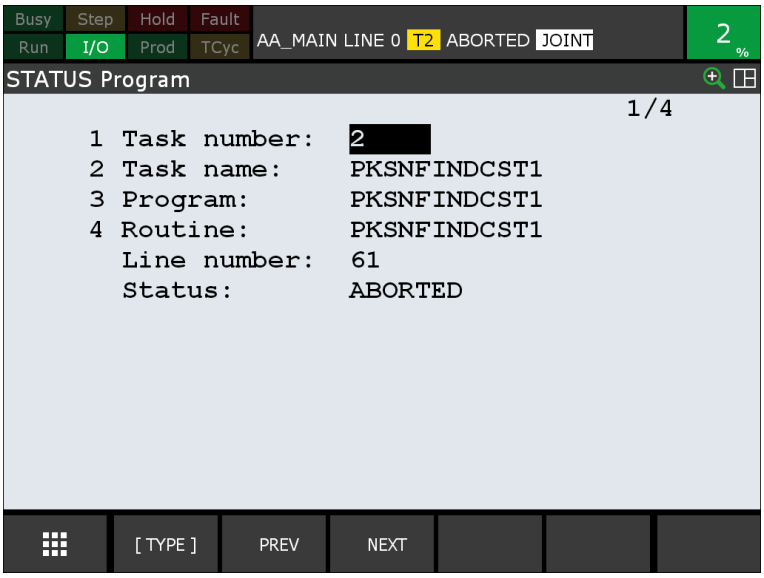
10 Execute the action one or more times. Select [Program] in [STATUS] menu as below.



When the action is [Put part in queue], you can know the customized action is executed if there is PKSNPUTCSTn (n = 1, 2, 3, 4) in [Program]. Normally, PKSNPUTn (n = 1, 2, 3, 4) is executed.



When the action is [Find part by vision], you can know the customized action is executed if there is PKSNFINDCSTn (n = 1, 2, 3, 4) in [Program]. Normally, PKSNFINDn (n = 1, 2, 3, 4) is executed.



The "n" of PKSNPUTCSTn (n = 1, 2, 3, 4) or PKSNFINDCSTn (n = 1, 2, 3, 4) is ID for system software to identify the enabled sensor.
For example, when "SENS1" is enabled and "SENS2" is disenabled and "SENS3" is enabled and their settings are stored in the system variable as below
$PKSN[1].$NAME = "SENS1" (Enabled)
$PKSN[2].$NAME = "SENS2" (Disenabled)
$PKSN[3].$NAME = "SENS3" (Enabled)
$PKSN[4].$NAME = "" ,
system software assigns a number for the enabled sensor from above.
In the above case, "SENS1" is n = 1 and "SENS3" is n = 2.
When the action of each sensor is executed, KAREL program corresponding to n assigned here is called.
When the action of "SENS1" is, [Put part in queue] the KAREL program is PKSNPUTCST1.
When the action of "SENS3" is [Find part by vision], the KAREL program is PKSNFINDCST2.
Then each KAREL program calls routine corresponding to each the action defined in pksnlibcst. The routine is described later.

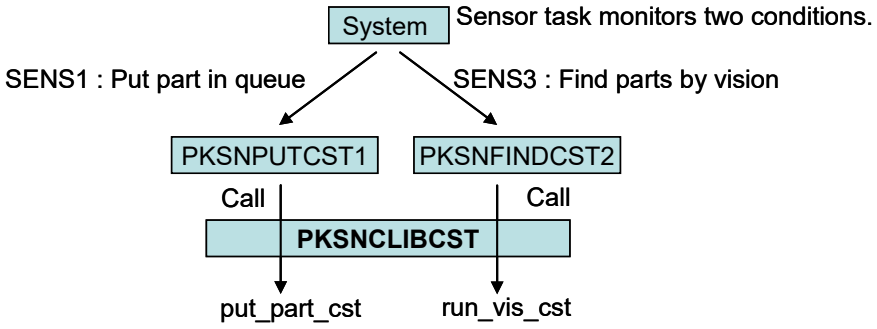


Fig. H.2(a) The flow of calling when 'SENS1' and 'SENS3' are enabled and their conditions are met.

H.3 ESSENTIAL POINTS OF CUSTOMIZATION

In pksnlibcst.kl, the sample program of [Put part in queue] and [Find part by vision] action is described.
In the sensor where the previous described bit is set, the routine described here is called from the system software when a condition is met.
Please refer to H.4 "ROUTINES" or H.5 "BUILT-INS" described later for the detail of each function.
Here the essential points of customization are explained.

Definition Part :

You should NOT change here.

```
-------------------------------------------------------------------------
PROGRAM pksnlibcst
-- You cannot change this KAREL program name
-------------------------------------------------------------------------
%COMMENT='CustomSenLib'
%SYSTEM
%NOBUSYLAMP
%NOLOCKGROUP
%NOPAUSESHFT
%NOPAUSE=ERROR+COMMAND+TPENABLE
%NOABORT=ERROR+COMMAND+TPENABLE

%ENVIRONMENT cvis                -- Builtins for vision
%ENVIRONMENT ltrk                -- Builtins for line tracking
%ENVIRONMENT pksndef             -- Variable for sensor

%INCLUDE pksnglbl
%INCLUDE vierrdef                -- Vision error definition. VEXE_E_NOMOR

-------------------------------------------------------------------------
ROUTINE get_ecnt_idx(indexer_id : INTEGER) : INTEGER FROM pksnlib
-------------------------------------------------------------------------
-------------------------------------------------------------------------
ROUTINE update_nfnd(cond_id : INTEGER; num_found : INTEGER) FROM pksnlib
-------------------------------------------------------------------------
-------------------------------------------------------------------------
ROUTINE start_srv(cond_id : INTEGER) FROM pksnlib
-------------------------------------------------------------------------
-------------------------------------------------------------------------
ROUTINE incre_wid(work_id : INTEGER) FROM pksnlib
-------------------------------------------------------------------------
-------------------------------------------------------------------------
ROUTINE put_queue(cnv_id : INTEGER; stn_id : INTEGER; work_id : INTEGER; enc_count :
INTEGER; model_id : INTEGER; offset : XYZWPR; found_pos : ARRAY OF XYZWPR;
meas : ARRAY OF REAL; duplicated : INTEGER; stat : INTEGER) FROM pksnlib
    -------------------------------------------------------------------------
```

The process of [Put part in queue] action :

```
--------------------------------------------------------------------------
ROUTINE put_part_cst(cond_id : INTEGER; stat : INTEGER)
-- Action: “Put part in queue”
-- You cannot change this function name
--------------------------------------------------------------------------
VAR
   model_id      : INTEGER
   enc_count     : INTEGER
   offset        : XYZWPR
   i             : INTEGER
   dmy_ofs       : XYZWPR
   dmy_meas      : ARRAY[MAX_MEAS] OF REAL
   duplicated    : INTEGER
   nfound        : INTEGER

BEGIN
   -- Initialize number of found part
   nfound = 0

   -- Read the triggered encoder count
   IF sens_cfg[cond_id].trig_cnd = PKSN_CND_FLG THEN
      -- For servo conveyor (indexer)
      enc_count = get_ecnt_idx(sens_cfg[cond_id].indexer_id)
   ELSE
      IF sens_cfg[cond_id].trig_cnd = PKSN_CND_DST THEN
         -- For belt conveyor where part is put every the specified distance
         enc_count = $pksnact[cond_id].$enc_count
      ELSE
         -- For belt conveyor where part is put when DI or HDI is input
         enc_count = LINE_COUNT(sens_cfg[cond_id].enc_id)
      ENDIF
   ENDIF

   -- Start the service to read accurate encoder for the next action
   start_srv(cond_id)

   -- Set dummy results. You can customize them before they are put
   dmy_ofs.x = 0.0
   dmy_ofs.y = 0.0
   dmy_ofs.z = 0.0
   dmy_ofs.w = 0.0
   dmy_ofs.p = 0.0
   dmy_ofs.r = 0.0
   FOR i = 1 TO MAX_MEAS DO
      dmy_meas[i] = 0.0
   ENDFOR
   model_id = 0

   -- Put part information to queue
   put_queue(sens_cfg[cond_id].cnv_id,
             sens_cfg[cond_id].stn_id,
             sens_cfg[cond_id].work_id,
             enc_count,
             model_id,
```

Here you can store the original value of your application.

```
                dmy_ofs,
                sens_cfg[cond_id].sens_pos,
                dmy_meas,
                duplicated,
                stat)
   IF (stat = ER_SUCCESS) THEN
      nfound = 1
      -- Update work id. Work id is updated as below. Work id = 1,2,3...
      incre_wid(sens_cfg[cond_id].work_id)
   ENDIF

   -- Notify of the number of found part. You must call this function in the end of this routine.
   update_nfnd(cond_id, nfound)
END put_part_cst
```

The process of [Find part by vision] action :

```
--------------------------------------------------------------------------
ROUTINE run_vis_cst(cond_id : INTEGER; stat : INTEGER)
-- Action: “Find part by vision”
-- You cannot change this function name
--------------------------------------------------------------------------
VAR
   model_id       : INTEGER
   enc_count      : INTEGER
   offset         : XYZWPR
   tray_pos       : XYZWPR
   found_pos      : ARRAY[MAX_NUM_VIEW] OF XYZWPR
   meas           : ARRAY[MAX_MEAS] OF REAL
   nfound         : INTEGER
   duplicated     : INTEGER
```

```
BEGIN
   -- Initialize number of found parts
   nfound = 0

   -- Find parts
   V_RUN_FIND(sens_cfg[cond_id].vp_name1, -1, stat)
   WHILE (stat = ER_SUCCESS) DO

      -- Get information of found parts
      VT_GET_FOUND(sens_cfg[cond_id].vp_name1, model_id, enc_count, offset,
found_pos, meas, stat)
      IF (stat = ER_SUCCESS) THEN

         -- If tray, compute tray position on track frame
         IF (sens_cfg[cond_id].tray_id <> 0) THEN
            found_pos[1] = found_pos[1] : sens_cfg[cond_id].tray_pos
         ENDIF

         -- Put part information to queue
         put_queue(sens_cfg[cond_id].cnv_id,
                   sens_cfg[cond_id].stn_id,
                   sens_cfg[cond_id].work_id,
                   enc_count,
                   model_id,
                   offset,
                   found_pos,
                   meas,
                   duplicated,
                   stat)
         IF (stat = ER_SUCCESS) THEN
            nfound = nfound + 1
            -- Update work id. Work id is updated as below. Work id = 1,2,3...
            incre_wid(sens_cfg[cond_id].work_id)
         ENDIF
      ENDIF
   ENDWHILE
   IF stat = VEXE_E_NOMOR THEN
      stat = ER_SUCCESS
   ENDIF

   -- Notify of the number of found part. You must call this function in the end of this routine.
   update_nfnd(cond_id, nfound)
END run_vis_cst
```

Here you can store the original value of your application.

When you want to put a part to another conveyor according to model ID, you branch a process before put_queue. Regarding getting coveyor ID and conveyor station ID, you can use P_CVGETID and P_CSGETID BUILT-In described later.

H

H.4 ROUTINES

H.4.1 put_part_cst

Purpose:

When [Put part in queue] action is executed, this routine is called from system software.

Syntax:

put_part_cst(cond_id, status)
Function Return Type: None
[in] cond_id : INTEGER
[out] status : INTEGER

Details:

- cond_id is defined by system software to identify the enabled sensor. It is different from sensor id. If you want to know sensor id, you can know it from sens_cfg[cond_id].sens_id. That is, you can know sensor name of cond_id from $PKSN[sens_cfg[cond_id].sens_id].$NAME.
- The status of the process is returned to status.

H.4.2 run_vis_cst

Purpose:

When [Find part by vision] action is executed, this routine is called from system software.

Syntax:

run_vis_cst(cond_id, status)
Function Return Type: None
[in] cond_id : INTEGER
[out] status : INTEGER

Details:

- cond_id is defined by system software to identify the enabled sensor. It is different from sensor id. If you want to know sensor id, you can know it from sens_cfg[cond_id].sens_id. That is, you can know sensor name of cond_id from $PKSN[sens_cfg[cond_id].sens_id].$NAME.
- The status of the process is returned to status.

H.4.3 get_ecnt_idx

Purpose:

When you use servo conveyor, you can get encoder count by this.

Syntax:

get_ecnt_idx(indexer_id)
Function Return Type: INTEGER
[in] indexer_id : INTEGER

Details:

- Specify servo conveyor id as indexer_id. The servo conveyor id used in this sensor is stored in sens_cfg[cond_id].indexer_id.
- Return encorder count.

H.4.4 update_nfnd

Purpose:

This notifies system software of the number of found parts. You must call this routine in the action.

Syntax:

update_nfnd(cond_id, num_found)
Function Return Type: None
[in] cond_id : INTEGER
[in] num_found : INTEGER

Details:

- cond_id is defined by system software to identify the enabled sensor. It is different from sensor id. If you want to know sensor id, you can know it from sens_cfg[cond_id].sens_id. That is, you can know sensor name of cond_id from $PKSN[sens_cfg[cond_id].sens_id].$NAME.
- num_found is the number of found parts.

H.4.5 start_srv

Purpose:

This starts a service to realize the accurate trigger. You must call this routine in the action.

Syntax:

start_srv(cond_id)
Function Return Type: None
[in] cond_id : INTEGER

Details:

- cond_id is defined by system software to identify the enabled sensor. It is different from sensor id. If you want to know sensor id, you can know it from sens_cfg[cond_id].sens_id. That is, you can know sensor name of cond_id from $PKSN[sens_cfg[cond_id].sens_id].$NAME.

H.4.6 incre_wid

Purpose:

This increments work ID. When the work ID runs over 13421771, the work ID increments from 0 again. When tray is used, after the tray information is put into a tracking queue, the work ID for each cell is assigned as follow:
Cell work ID = Tray work ID * 160 + Cell index – 1
The maximum work ID is 13421771 because above cell work ID should not run over the maximum of INTEGER.

Syntax:

incre_wid(work_id)
Function Return Type: None
[in/out] work_id : INTEGER

Details:

- Specify previous work id as work_id.
- Incremented work id is returned to work_id.

H.4.7 put_queue

Purpose:

This puts part information into a tracking queue. It simultaneously performs a duplicate check, and outputs a status (= status in the "syntax" below) when a part is identified as a duplicate. When the robot of the target conveyor station is off-line, it puts a part to the next conveyor station automatically. All robots are off-line, a message to indicate it is displayed in TP and SUCCESS (= 0) is returned as status in order to continue the process.

Syntax:

put_queue(cnv_id, stn_id, work_id, enc_count, model_id, offset, found_pos, meas, duplicated, stat)
Function Return Type: None
[in] cnv_id : INTEGER
[in] stn_id : INTEGER
[in] work_id : INTEGER
[in] enc_count : INTEGER
[in] model_id : INTEGER
[in] offset : XYZWPR
[in] found_pos : ARRAY[4] OF XYZWPR
[in] meas : ARRAY[10] OF REAL
[out] duplicated : INTEGER
[out] stat : INTEGER

Details:

- Specify a conveyor id as cnv_id where parts flows. You can get the conveyor id from sens_cfg[cond_id].cnv_id.
- Specify a conveyor station id as cstn_id. The part information is put into the queue of the specified conveyor station. You can get the conveyor station id from sens_cfg[cond_id].stn_id.
- Specify the unique identifier of the part as work_id when needed. Specify 0 when you do not use the IDs.
- Specify an encoder value as enc_count. It is the encoder value got at the moment when the image is acquired in order to find the part or when the sensor such as photo eye detects the part.
- Specify the model id for the part as model_id.
- Specify the offset data of the part as offset.
- Specify the found positions as found_pos[1-4].
- Specify the measurement values as meas_val[1-10].
- Usually, just specify the values acquired by VT_GET_FOUND built-in of *i*RVision from enc_count to meas_val.
- If a new put part already exists in a tracking queue, "duplicated" becomes 1 and the part is not put into the queue. Otherwise, if the new put part does not exist in the queue, "duplicated" becomes 0 and the part is put into the queue.
- The status of the process is returned to status. If the process is successful, then status is set to 0. If the process is not successful, then an alarm code that is not equal to 0 is returned.

H.5 BUILT-INS

H.5.1 LINE_COUNT

Purpose:

This acquires an encoder value.

Syntax:

LINE_COUNT(encoder_id)
Function Return Type: INTEGER
[in] encoder_id : INTEGER
%ENVIRONMENT GROUP: LTRK

Details:

- Specify an encoder number as encoder_id.
- This built-in returns an encoder value. The encoder value acquired at execution of this built-in is usually returned.
- If start_srv is called in advance, the accurate encoder value acquired at the DI signal is turned to ON from OFF is returned.
- If HDI is used, an accurate encoder value acquired at the HDI signal is turned to ON from OFF is returned.

H.5.2 V_RUN_FIND

Purpose:

This executes a vision program.

Syntax:

V_RUN_FIND(vp_name, camera_view,status)
[in] vp_name: STRING
[in] camera_view: INTEGER
[out] status: INTEGER
%ENVIRONMENT GROUP: CVIS

Details:

- Specify a vision program name as vp_name.
- Specify a camera view number as camera_view. If 0 is specified, all the camera views of the vision program are executed.
- The status of the process is returned to status. If the process is successful, then status is set to 0. If the process is not successful, then an alarm code that is not equal to 0 is returned.
- This built-in finishes just after the vision program finishes acquiring an image. The vision program continues the image processing in the background. Therefore, the image processing and the KAREL program can be executed in parallel. When VT_GET_FOUND is executed, it waits for the completion of the image processing.

H.5.3 VT_GET_FOUND

Purpose:

This outputs the information of a part that a vision program finds.

Syntax:

VT_GET_FOUND(vp_name, model_id, enc_count, offset, found_pos, meas_val, status)
[in] vp_name : STRING
[out] model_id : INTEGER
[out] enc_count : INTEGER
[out] offset : XYZWPR
[out] found_pos : ARRAY[4] OF XYZWPR
[out] meas_val : ARRAY[10] OF REAL
[out] status : INTEGER
%ENVIRONMENT GROUP: CVIS

Details:

- Specify a vision program name as vp_name.
- The model ID of the found part is returned to model_id.
- The encoder value got at the moment that the image is acquired in order to find the part is returned to enc_count.
- The offset data of the found part is returned to offset.
- The found positions of camera view 1-4 are returned to found_pos[1-4].
- The measurement values of the found part are returned to meas_val[1-10]. The measurement values are specified in the measurement output tool of a vision program.
- The status of the process is returned to status. If the process is successful, then status is set to 0. If the process is not successful, then an alarm code that is not equal to 0 is returned.
- This built-in is used after V_RUN_FIND built-in.
- If vision program finds more than one part, call this built-in repeatedly until status of this built-in is not equal to 0. When there is no more part to get, the status becomes 117151.

H.5.4 P_CSGETID

Purpose:

This returns the conveyor station id got from a specified conveyor station name.

Syntax:

P_CSGETID(conv_name, group_num)
Function Return Type: INTEGER
[in] conv_name: STRING
[in] group_num: INTEGER
%ENVIRONMENT Group: PKQU

Details:

- Specify a conveyor name as conv_name.
- Specify a robot group number as group_num.
- This built-in returns a conveyor station ID.

H.5.5 P_CVGETID

Purpose:

It returns the conveyor id got from a specified conveyor name.

Syntax:

P_CVGETID(conv_name)
Function Return Type: INTEGER
[in] conv_name: STRING
%ENVIRONMENT Group: PKQU

Details:

- Specify a conveyor name as conv_name.
- This built-in returns the conveyor number.

I EXTERNAL MACHINE VISION INTERFACE ADD-ON

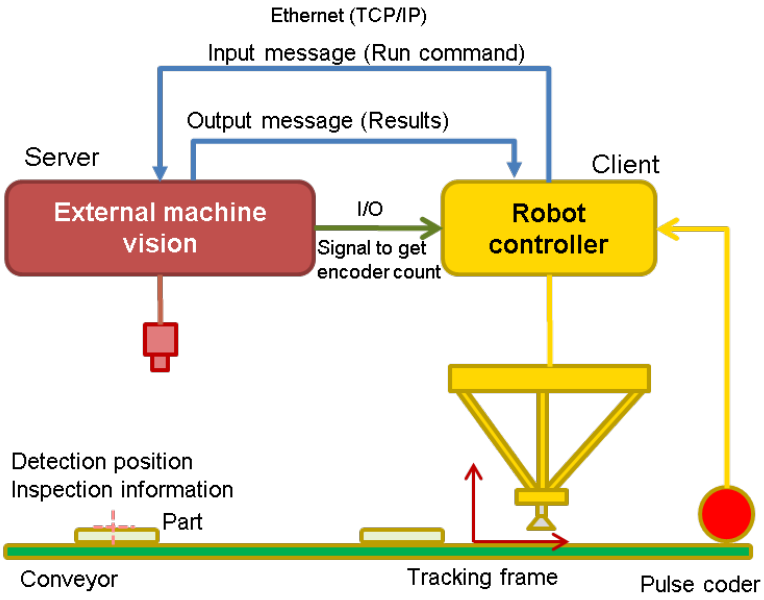
This chapter explains *i*RPickTool External Machine Vision Interface Add-on.

I.1 OVERVIEW

The *i*RPickTool External Machine Vision Interface Add-on provides a function to perform 2D visual tracking based on the results of the part obtained from the External machine vision via TCP/IP communications. As an action of the *i*RPickTool's sensor, this add-on runs a TP program that communicates with the External machine vision. This TP program does the following.

- Sends a run command to the External machine vision and receives the results.
- Receives the signal that the External machine vision outputs when it snaps images (signal to get encoder count) and acquires the encoder count.
- Performs frame conversion for the found position to a location in the tracking frame.
- Inputs the found position to a conveyor station.

The KAREL program is used for message transmission with the External machine vision. You should define the format of the messages to suit to your External machine vision and system.



To use the *i*RPickTool External Machine Vision Interface Add-on, you need the following software options.

- J944 (*i*RPickTool Basic) or J945 (*i*RPickTool)
- R648 (User Socket Messaging)
- S521 (*i*RPickTool/External Machine Vision Interface Add-on)

NOTE
This function is available for order for 7DF1 system software with version 17 or later.

I.2 TERMS

This section describes terms specific to the *i*RPickTool External Machine Vision Interface Add-on.

Data

Elements of a message separated by delimiters.

Input message

A message that the robot controller sends to the External machine vision. A newline character is placed at the end of the message. An input message contains the following.

Run command

A command to make the External machine vision snap images and find and send the detection results. For details, refer to the operation manual of the relevant vision system manufacturers.

Output message

A message that the robot controller receives from the External machine vision. A newline character is placed at the end of the message. An output message contains the following.

Detection results

Information on the part output from the External machine vision to the robot controller. You should choose necessary information for your system and determine the message format.

Response

A status that the External machine vision outputs when it receives a command. For details, refer to the operation manual of the vision system manufacturer.

Action program

A program that communicates with the External machine vision and puts the found position into a queue.

Latch program

A program that acquires the encoder count at the moment when the External machine vision snaps an image.

Signal to get the encoder count

A signal used for latching. The robot controller acquires the encoder count when it receives this signal. For the signal, use a signal such as a strobe signal that synchronizes with image snapping.

I.3 RESTRICTIONS

This section describes the restrictions on robot systems and the External machine visions for which the *i*RPickTool External Machine Vision Interface Add-on can be used.

I.3.1 Restrictions on Robot Systems

- The External machine vision is applicable to 2D visual tracking only.
- The External machine vision is applicable to line conveyor only.
- When using multiple External machine visions on the same robot controller, each of the encoders used for the conveyor stations corresponding to the sensors that use the External machine visions must be different from the others.
- A tray cannot be used for a conveyor to which the External machine vision is applied.
- A dual-arm robot cannot be connected to the External machine vision.
- The External machine vision cannot be used for an infeed conveyor for pre-grouping.

I.3.2 Restrictions on External Machine Visions

- It must be able to communicate based on the TCP/IP protocol using Ethernet.
- It must be able to communicate via TCP/IP using the ASCII format.
- The External machine vision must be able to serve as a TCP/IP server. The robot controller cannot be used as a server.
- Restrictions regarding input messages:
  - The External machine vision must be able to snap images and find positions in response to an input message sent by the robot controller.
  - The string length of each data in an input message must be 34 characters or less.
  - At least one of the newline characters (LF, CR, CR+LF, or None) must be supported.
- Restrictions regarding output messages:
  - When the External machine vision has completed finding or received an input message to acquire the results, it must be able to send an output message (the results) to the robot controller.
  - The External machine vision must be able to send at least the found position (X, Y, R) of the part as the results.
  - The string length of each data in an output message must be 254 characters or less.
  - It must be possible to separate each data in the results with a delimiter, either ',' (comma) or ' ' (space).
  - It must be possible to place at least one of the newline characters (LF, CR, CR+LF) at the end of the results.
- At the moment when an image is snapped, the External machine vision must be able to output a signal to the robot controller.

I.4 SCOPE OF SUPPORT

This section describes the scope of support.

Software:
- Our support covers robot controller functions only.
- For details about support for image snapping by the camera, object detection in images, inspection sections, and others, please contact the respective vision system manufacturers.
- Our robot controller operates on the assumption that the External machine vision sends correct results and signals are sent at the correct time.

Hardware:
- Our support covers the robot and the robot controller only.
- For vision system components such as cameras, lenses, and cables, please contact the respective vision system manufacturers.

For details on the scope of support, see Figure I.4.

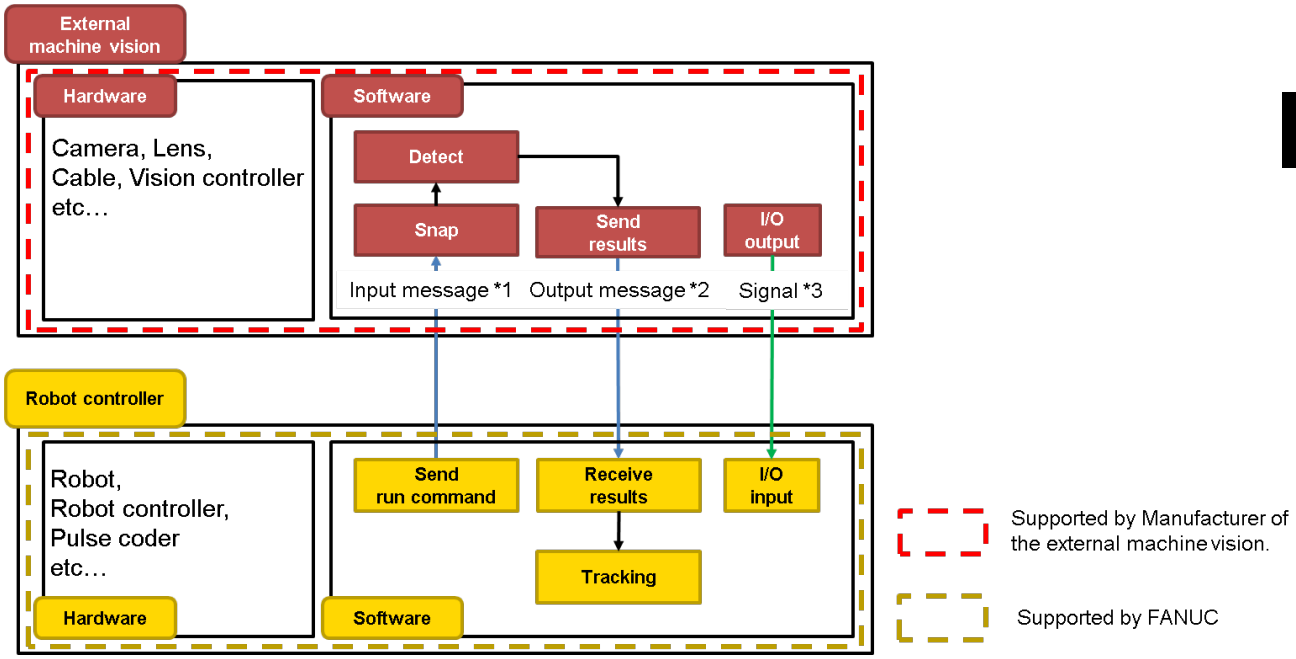


Fig. I.4 Scope of support

*1
Our support covers up to the process where defined input messages are sent to the External machine vision via TCP/IP.
Processes such as position detection that are run based on input messages received by the External machine vision are supported by the respective External machine vision manufacturers.

*2
You, as an operator of the robot, should determine the output message format to suit the system.
The External machine vision manufacturers should support the creation of output messages according to the format and their transmissions by the External machine vision via TCP/IP.
Our support covers the processes up to extraction of results from the output messages received by the robot controller and their tracking.

*3
The transmission and transmission timing of signals used to acquire the conveyor location when snapping images (signals such as a strobe signal that synchronize with image snapping) are supported by the relevant External machine vision manufacturers.
We support signal reception and acquisition of conveyor locations at the time when signals are received.

I.5 STARTUP PROCEDURES

This section describes the startup procedures for the *i*RPickTool External Machine Vision Interface Add-on.
This section explains an example system that has the same configuration as the example shown in Chapter 3, "*i*RPickTool Basic (Basic Functions) Startup Procedures." To use the *i*RPickTool External Machine Vision Interface Add-on, change the infeed conveyor combination as follows.

Infeed: [Distance] + [Run Program]

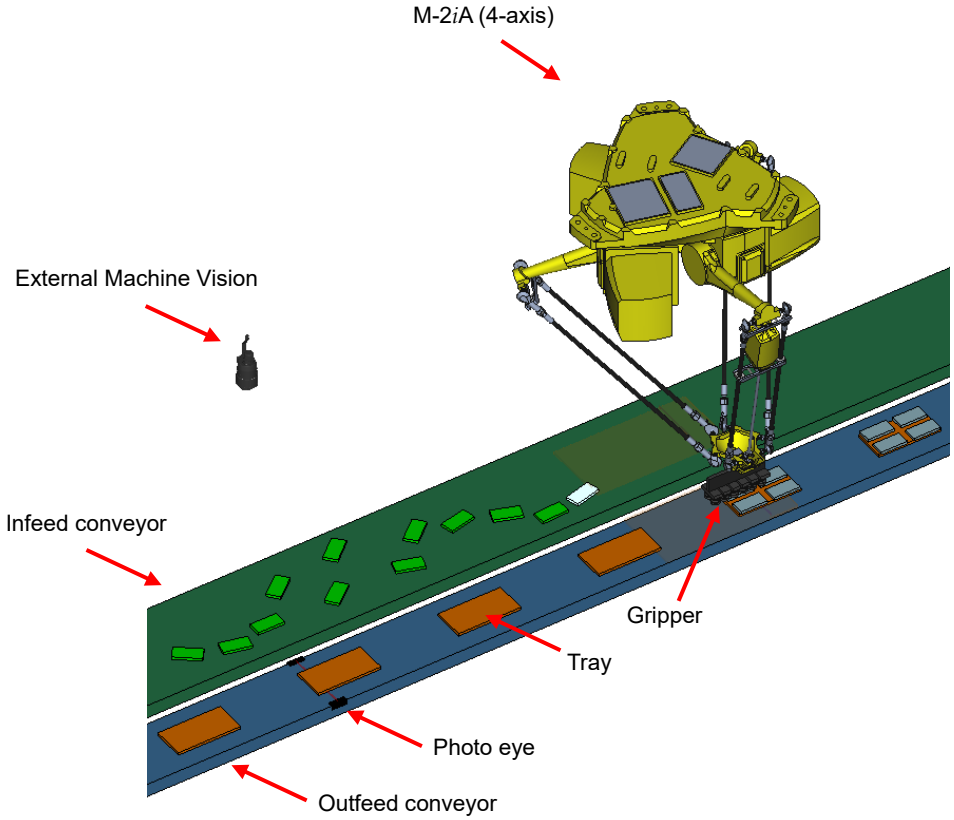


Fig. I.5(a) System configuration

As an example, this section explains a case where the External machine vision communicates with the robot controller using the message format in Table I.5. Change the message format depending on the External machine vision that you use in practice.

Table I.5(a) Message Format

Input message (Controller → External machine vision)	RUNFIND (string) <CR>
Response to an input message (External machine vision → Controller)	When normal : OK (string) <CR> When abnormal : ER (string) <CR>
Output message (External machine vision → Controller)	Number of detections (Integer), Model ID (Integer), X (Real), Y (Real), R (Real), Measurement 1 (Real) <CR>

I.5.1 Connecting and Setting up the External Machine Vision

Connect and set up the External machine vision and the robot controller so that they can communicate with each other. For how to set them up, refer to the operation manuals of the relevant vision system manufacturers.

I.5.1.1 Connecting the External Machine Vision

- Connect the External machine vision to the robot controller with an Ethernet cable.
- Connect a signal such as a strobe signal that synchronizes with image snapping to DI or RI of the robot controller as the signal to get the encoder count.

NOTE
For the Ethernet port to connect with the External machine vision, use an independent port that is not used for other communication.

I.5.1.2 Setting up the External Machine Vision

- Set up the External machine vision so that it can connect with the robot controller via Ethernet based on the TCP/IP protocol.
- Teach the External machine vision so that a part can be found.
- Set up the External machine vision so that results can be sent to the robot controller according to the output message format.
- Set up the signal to get encoder count.

Record the following items to be used for setting up the robot controller:
- IP address and port number used for TCP/IP
- The run command to make the External machine vision snap images and find and send results
- The DI or RI number of the robot controller to which the signal to get the encoder count is connected

I.5.2 Setting up the Communication

Set up the communication with the External machine vision.

I.5.2.1 Setting up the IP Address

1 Press the [MENU] key on the robot controller's teach pendant.
2 Select [SETUP] → [Host Comm] on the menu.
3 Select TCP/IP and press F3 [DETAIL].
4 Set up the IP address and subnet mask.

I.5.2.2 Setting up the Client Tag

Setting up the host communication screen

1 Press the [MENU] key on the robot controller's teach pendant.
2 Select [SETUP] → [Host Comm] on the menu.
3 Select F4 [SHOW] → [2 Clients].
4 Move the cursor to the client tag you are using, and press the [ENTER] key.
5 Move the cursor to [Protocol] and press the [ENTER] key.
6 Select [SM].
7 Move the cursor to [Server IP/Host name] and press the [ENTER] key.
8 Enter the IP address of the External machine vision.

9 Move the cursor to [Startup state] and press the [ENTER] key.
10 Select [Define].
11 Press F2 [ACTION] and select [DEFINE].

Setting up system variables

1 Press the [MENU] key on the robot controller's teach pendant.
2 Select [SYSTEM] → [Variables] on the menu.
3 Select $HOSTC_CFG.
4 Select the number of the client tag that you set in "Setting up the host communication screen."
5 Check that the IP address entered in step 8 "Setting up the host communication screen" appears in $STRT_REMOTE.
6 Enter the port number of the External machine vision in $SERVER_PORT.

I.5.3 Setting up *i*RPickTool

Set up the items common to those in Chapter 3, "*i*RPickTool Basic (Basic Functions) Startup Procedures." For the items below, the same setting is applied when using the *i*RPickTool External Machine Vision Interface Add-on. Refer to the items in Chapter 3, "*i*RPickTool Basic (Basic Functions) Startup Procedures" for setup.

- "3.1.1 Connection and Setup of Pulsecoders"
- "3.1.3 Checking the Input of the Trigger Sensor"
- "3.1.4 Setting up the Tool Frame of the Pointer Tool"
- "3.1.6 Setting Payloads (Motion Performance) "
- "3.2.1 Workcell Setup"
- "3.2.2 Setting Up Trays"
- "3.2.3 Setting Up a Conveyor"
- "3.2.4 Setting Up a Tracking Frame"
- "3.2.5.2 Setting up an outfeed conveyor sensor (In the case of [DI] / [RI] + [Put part in queue])" or "3.2.5.3 Setting up an outfeed conveyor sensor (In the case of [HDI] + [Put part in queue]) "
- "3.2.6 Setting Up a Sensor Position"
- "3.2.8 Setting Up a Conveyor Station"
- "3.2.9 Setting Up a Tool Frame for the Hand"
- "3.2.10.2 Setting up a reference position for the outfeed conveyor, and teaching a robot position"

I.5.4 Creating a Robot Program

Create a robot program. For how to create a program, refer to Section 3.1.5, "Creating a Robot Program."

Create the following robot programs. The tracking program and vacuum ON/OFF program are not explained here as they are the same as those created in Section 3.1.5, "Creating a Robot Program." Refer to Section 3.1.5, "Creating a Robot Program" to create those programs.

- Main program
- Tracking program
- Vacuum ON/OFF program

The *i*RPickTool External Machine Vision Interface Add-on requires a program that communicates with the External machine vision and acquires the results. This program is run from the sensor task as *i*RPickTool's action every time the trigger condition is met. See Figure I.5.4 (a) for its relationship with the main program.

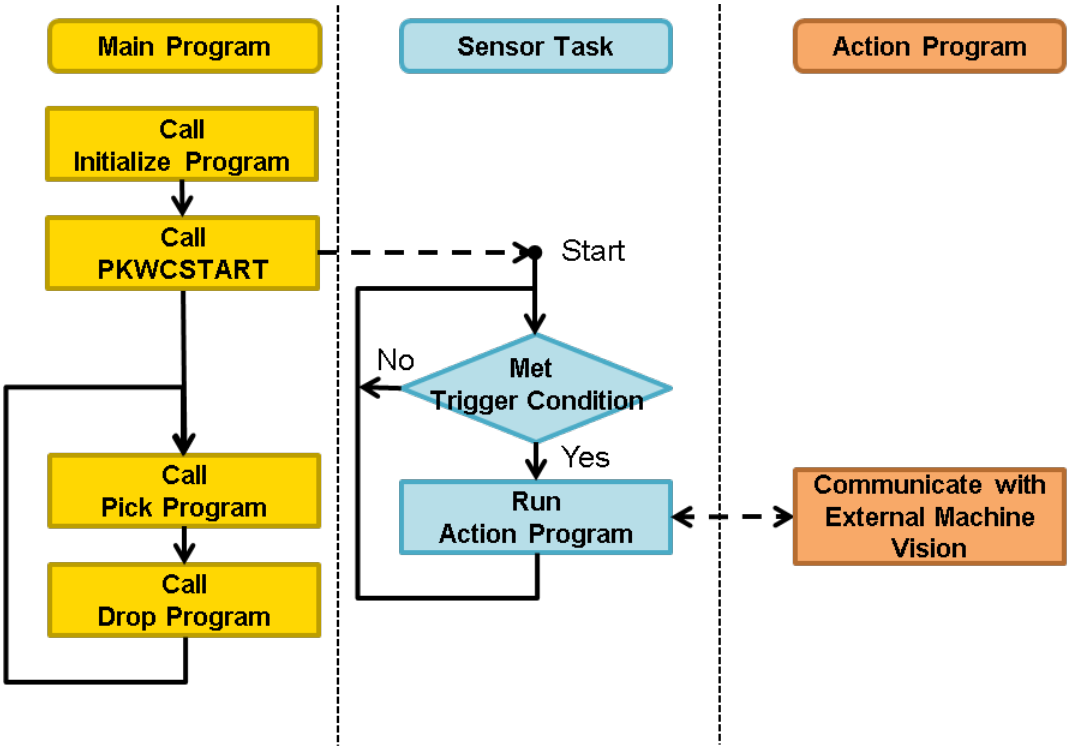


Fig. I.5.4 (a) Executed programs

Create the following programs to communicate with the External machine vision and acquire the results.

- Action program
- Latch program
- Find program

NOTE

Enable "ignore pause" in the program above. Otherwise, the parts information may disappear before it is put into a queue if the program is terminated by an emergency stop etc. during communicating.

These programs do the following. For the flow of the programs, see Figure I.5.4 (b).

1 Send a run command to the External machine vision and receive the results.
2 Receive a signal that the External machine vision outputs when it snaps an image (signal to get encoder count) and acquire the encoder count.
3 Convert the frame for the found position into a location in the tracking frame.
4 Input the found position to a conveyor station.

Also, create the following program to initialize connection with the External machine vision and others.

- Initialization program

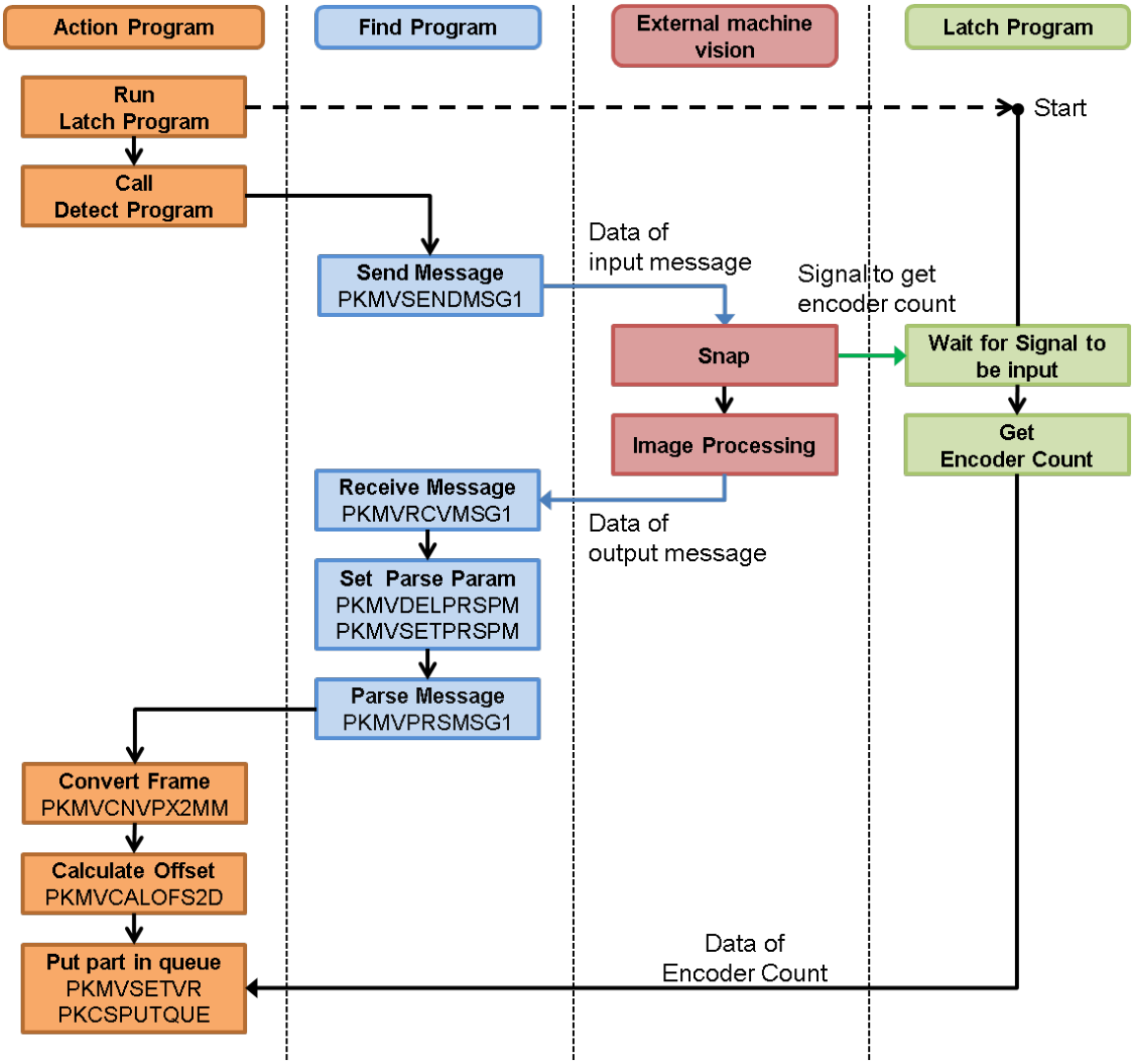


Fig. I.5.4 (b) Flowchart of the action program

For the *i*RPickTool External Machine Vision Interface Add-on, create a program to calculate frame conversion parameters and a program to set a reference position.

- Program to calculate frame conversion parameters
- Program to set reference positions

Definition of registers

These programs use the registers below. Change the register numbers appropriately as needed.

Table I.5.4 (a) Registers

R[2:GETQ Status Pick	]=0	Pick part processing status
R[3:GETQ Status Drop	]=0	Place part processing status
R[11:CONV1 ID	]=1	CONV1 (pick) ID
R[12:CONV2 ID	]=2	CONV2 (place) ID
R[21:Pick AP_LD	]=30	Pick approach distance specification [mm]
R[22:Pick RT_LD	]=30	Pick retract distance specification [mm]
R[23:Pick Wait	]=.05	Index delay at pick position [sec]
R[24:Pick CNT	]=0	Pick position CNT specification
R[25:Pick TB	]=.1	TIME BEFORE time (VACUUM_ON) [sec]
R[26:Pick Speed1	]=4000	Pick program speed 1 [mm/sec]
R[27:Pick Speed2	]=4000	Pick program speed 2 [mm/sec]
R[28:Pick Speed3	]=2000	Pick program speed 3 [mm/sec]
R[31:Drop AP_LD	]=30	Place approach distance specification [mm]
R[32:Drop RT_LD	]=30	Place retract distance specification [mm]
R[33:Drop Wait	]=.05	Index delay at drop position [sec]
R[34:Drop CNT	]=0	Drop position CNT specification
R[35:Drop TB	]=.1	TIME BEFORE time (VACUUM_OFF) [sec]
R[36:Drop Speed1	]=2000	Placement program speed 1 [mm/sec]
R[37:Drop Speed2	]=2000	Placement program speed 2 [mm/sec]
R[38:Drop Speed3	]=4000	Placement program speed 3 [mm/sec]
R[40:Num. to get Enc Cnt	]=2	DI/RI Number to get encoder count (Specify a number to use.)
R[41:Client ID	]=1	Client tag number (Specify a number to use.)
R[43:SENDMSG Status	]=0	Status of PKMVSENDMSG
R[44:RCVMSG Status	]=0	Status of PKMVRCVMSG
R[45:PRSMSG Status	]=0	Status of PKMVPRSMSG
R[46:MVSFIND1 Status	]=0	Status of the results of External machine vision*1
R[47:MVSACTION1 End	]=0	Flag indicating MVS ACTION1 has ended
R[48:LATCH Success	]=0	Flag indicating Latch has succeeded
R[49:Duplicated res	]=0	Duplicated results when results were put into a queue
R[51:x0	]=0	x0 (Found position for calculating frame conversion parameters)
R[52:y0	]=0	y0 (Found position for calculating frame conversion parameters)
R[53:x1	]=0	x1 (Found position for calculating frame conversion parameters)
R[54:y1	]=0	y1 (Found position for calculating frame conversion parameters)
R[55:x2	]=0	x2 (Found position for calculating frame conversion parameters)
R[56:y2	]=0	y2 (Found position for calculating frame conversion parameters)
R[60:Encoder count	]=0	Encoder Count (Results)
R[61:Num of found]	=0	Number of found (Results)
R[62:Model ID	]=0	Model ID (Results)
R[63:x	]=0	x (Results)
R[64:y	]=0	y (Results)
R[65:r	]=0	r (Results)
R[66:Meas 1	]=0	Measurement 1 (Results)

*1 The status of the results of the External machine vision is defined in the TP program as below:

0: PUTQUE successful.
1: An error occurred during communication with the External machine vision.
2: The status of PKMVPRSMSG is not success.
3: Response is abnormal.
4: The number of found positions is 0.
5: No input signal to get encoder count.

Table I.5.4 (b) Position registers

PR[1:HOME POS]=R	Home position
PR[7:Pick Pos]=R	Pick position
PR[8:Drop Pos]=R	Drop position
PR[11:Pick Apr Z]=R	Offset from pick position
PR[12:Drop Apr Z]=R	Offset from drop position
PR[21:ADJ_OFS]=R	For pick position offset adjustment
PR[26:Ref Pos]=R	Reference position
PR[27:Found Pos]=R	Found position
PR[28:Offset]=R	Offset

Table I.5.4 (c) String registers

SR[6 :Response]	Response
SR[7 :OK Response]	Response when it is normal (Specify a response when it is normal.)

Table I.5.4 (d) Vision registers

VR[1]	For pick position offset
VR[2]	For drop position offset
VR[5]	For storing results

I.5.4.1 Main Program

This subsection explains an example of a main program of the system.

PNS0001

```
  1:  !PKCSGETID
  2:  !(Get conveyor station ID)
  3:  !*1:Conveyor name
  4:  !*2:Register number
  5:  !   for conveyor station ID
  6:  !*3:Group number (Can be omitted)
  7:  CALL PKCSGETID('CONV1', "CStn ID Reg" =11)
  8:  CALL PKCSGETID('CONV2', "CStn ID Reg" =12)
  9:
 10:  ! Initialize
 11:  CALL MVSINIT1
 12:
 13:  UFRAME_NUM=0
 14:  UTOOL_NUM=1
 15:  PAYLOAD[1]
 16:L PR[1:HOME POS] 500mm/sec FINE Wjnt
 17:
 18:  !--- System startup ---
 19:  !PKWCSTART(System startup,
 20:  !including check and clear)
 21:  CALL PKWCSTART
 22:
 23:  LBL[101]
 24:  !--- Pick program ---
 25:  CALL PT_PICKCS
 26:  IF DI[1]=ON,JMP LBL[901]
 27:  !--- Placement program ---
 28:  CALL PT_DROPCS
 29:  IF DI[1]=ON,JMP LBL[901]
 30:  JMP LBL[101]
 31:
 32:  LBL[901]
 33:  UFRAME_NUM=0
 34:  UTOOL_NUM=1
 35:L PR[1:HOME POS] 500mm/sec FINE Wjnt
 36:
 37:  !PKWCEND(System end)
 38:  CALL PKWCEND
[End]
```

KAREL program

KAREL program

Initialization

KAREL program

KAREL program

This program initializes *i*RPickTool and the External machine vision, and performs tracking.

Line 11 calls a TP program that initializes the connection with the External machine vision. For details, refer to Section I.5.4.3, "Initialization Program."

Other lines are the same as those of the main program described in Section 3.1.5, "Creating a Robot Program." The programs called on lines 25 and 28 are the same as the pick program and the placement

program in Section 3.1.5, “Creating a Robot Program.” Refer to Section 3.1.5, “Creating a Robot Program” to create a pick program and a placement program.

I.5.4.2 Action Program

This subsection describes an example of a program run as an action of *i*RPickTool.

MVSACTION1

```
  1:  R[47:MVSACTION1 End]=0
  2:
  3:  !Use ACCUTRIG
  4:  ACCUTRIG LNSCH[1]
  5:
  6:  R[48:LATCH Success]=0
  7:  RUN LATCH
  8:
  9:  CALL MVSFIND1
 10:  IF R[46:MVSFIND1 Status]<>0,JMP LBL[990]
 11:
 12:  IF R[48:LATCH Success]=0,JMP LBL[900]
 13:
 14:  !Store found position in PR
 15:  PR[27,1:Found Pos]=R[63:x]
 16:  PR[27,2:Found Pos]=R[64:y]
 17:  PR[27,3:Found Pos]=0
 18:  PR[27,4:Found Pos]=0
 19:  PR[27,5:Found Pos]=0
 20:  PR[27,6:Found Pos]=R[65:r]
 21:
 22:  !Conv fnd pos into tracking frm
 23:  CALL PKMVCNVPX2MM( “Action ID” =1, “Fnd Pos PR” =27, “Cnv Pos PR” =27)
 24:  !Calculate offset
 25:  CALL PKMVCALOFS2D( “Fnd Pos PR” =27, “Ref Pos PR” =26, “Offset PR” =28)
 26:
 27:  !Put result into queue
 28:   CALL PKMVSETVR( “VR” =5, “Fnd Pos PR” =27, “Offset PR” =28, “Model ID Reg” =62, “Enc Count
Reg” =60,66)
 29:  CALL PKCSPUTQUE( “CStn ID” =R[11:CONV1 ID], “Offset VR” =5, “OverlapStat Reg” =49)
 30:  JMP LBL[990]
 31:
 32:  LBL[900:LATCH ERORR]
 33:  R[46:MVSFIND1 Status]=5
 34:
 35:  LBL[990]
 36:  R[47:MVSACTION1 End]=1
 37:  CALL PKMVWAITTASK( “Task Name” =’LATCH’)
[End]
```

Use accuracy trigger

Run latch program

Acquire results

Check latch

Store in PR

Frame conversion
Calculate offset

Put results in queue

Error process

This program is run as an action of *i*RPickTool. It acquires the detection results from the External machine vision and puts it into a queue.

Line 4 sets up the accuracy trigger. If you use a tracking schedule number other than 1, change this argument to a tracking schedule number that you use.
Line 7 runs a latch program that acquires the encoder count when a signal is input from the External machine vision. For details, refer to Section I.5.4.5, “Latch Program.”

Line 9 calls a program that acquires results from the External machine vision. For details, refer to Section I.5.4.4, “Find Program.”

Line 12 checks whether the latch has succeeded. When the output timing of the signal to get encoder count is late or the output width is large, the signal may not be detected before this judgment, resulting in a judgment of latch failure. In such a case, use a signal that is output at the same time as image snapping for the signal to get the encoder count and set a smaller signal output width. Another option is to add a wait instruction before this line.

Lines 15 through 20 store the found position into position registers.
Line 23 converts the frame for the results and line 25 calculates the offset.
Line 28 stores the results in vision registers and line 29 puts the part into a queue.

Lines 32 and 33 define a label to jump to when an error occurs. Use these lines to deal with errors.

To end the latch program, line 36 stores 1 into R [47] to communicate to the latch program that the action program has ended. Line 37 waits for the latch program to end.

> **⚠ CAUTION**
> 1 For the number of the Vision Register of the action program PKMVSETVR, specify the different value from the number of the Vision Register for PKCSGETQUE called in the pick program or the place program. If you specify the same value, the value will be overwritten, and the robot may go to the wrong place to pick up the parts.
> 2 Do not run PKMVSENDMSG and PKMVRSVMSG except that the action program. If multiple programs try to communicate at the same time, the alarm “INTP-326 (PKMVLIB, 207) File var is already used” will appear. However, there is no problem as long as the action program does not use the corresponding communication port.

I.5.4.3 Initialization Program

This subsection describes an example of an initialization program.

MVSINIT1

```
  1:  CALL PKMVCONNECT( “Client ID” =R[41:Client ID])
  2:
  3:  CALL PKMVINIACT( “Action ID” =1,
                       “Client ID” =R[41:Client ID],
                       “In Msg Delimiter” =',',
                       “In Msg Newline char” ='CR',
                       “In Msg Max num data” =10,
                       “Out Msg Delimiter” =',',
                       “Out Msg Newline char” ='CR',
                       “Out Msg Max num data” =20)
  4:
  5:  CALL PKMVSETSNDPM( “Action ID” =1, “Stat Reg” =43,'RUNFIND')
  6:  CALL PKMVSETRCVPM( “Action ID” =1, “Timeout (ms)” =5000, “Stat Reg” =44)

[End]
```

Connect with MVS
Initialize action
Set SNDMSG param
Set RCVMSG param

Line 1 establishes a connection with the External machine vision. Before that, enter the client number you use into R[41].

Line 3 initializes the action. For details on the arguments, refer to Section I.6.2.1, “PKMVINIACT.PC.”

Line 5 sets up the input message. Specify the run command for the External machine vision for the input message.

Line 6 sets up parameters for receiving a message. For the timeout, specify the time longer than the External machine vision's image-snapping and finding time.

NOTE
Depending on the External machine vision, a connection may not be established with only the above program, for reasons such as a need to log in to initiate the connection. In such a case, add necessary processes accordingly.

I.5.4.4 Find Program

This subsection describes an example of a program that communicates with the External machine vision and acquires the results.

MVSFIND1

```
  1:  R[46:MVSFIND1 Status]=1
  2:
  3:  !Send run command
  4:  CALL PKMVSENDMSG1
  5:
  6:  !Receive response
  7:  CALL PKMVRCVMSG1
  8:
  9:  CALL PKMVDELPRSPM( "Action ID" =1)
 10:  CALL PKMVSETPRSPM( "Action ID" =1, "Stat Reg" =45,
                         3,6)
 11:  CALL PKMVPRSMSG1
 12:  IF R[45:PRSMSG Status]<>0,JMP LBL[900]
 13:  IF SR[6]<>SR[7],JMP LBL[910]
 14:
 15:  !Receive results
 16:  CALL PKMVRCVMSG1
 17:
 18:  CALL PKMVDELPRSPM( "Action ID" =1)
 19:  CALL PKMVSETPRSPM( "Action ID" =1, "Stat Reg" =45,
                         1,61,1,62,2,63,2,64,2,65,2,66)
 20:  CALL PKMVPRSMSG1
 21:  IF R[45:PRSMSG Status]<>0,JMP LBL[900]
 22:
 23:  IF R[61:Num of found]<=0,JMP LBL[920]
 24:
 25:  R[46:MVSFIND1 Status]=0
 26:  END
 27:
 28:  LBL[900:PKMVPRSMSG ERROR]
 29:  R[46:MVSFIND1 Status]=2
 30:  JMP LBL[990]
 31:
 32:  LBL[910:RESPONSE ERROR]
 33:  R[46:MVSFIND1 Status]=3
 34:  JMP LBL[990]
 35:
 36:  LBL[920:NUM OF FND IS 0]
 37:  R[46:MVSFIND1 Status]=4
 38:
 39:  LBL[990]
[End]
```

Send run command

Receive response

Set PRSMSG param

Parse response

Receive results

Set PRSMSG param

Parse results

Error process

Line 4 sends a run command to the External machine vision. The run command has been set in Section I.5.4.3, " Initialization Program."

Line 7 receives a response to the run command. PKMVRCVMSG receives a message separated by newline characters every time it is run. Even when the External machine vision sends both responses and results, only responses are received in this process.
Line 10 sets parameters for a parsing response. For details on the parameters, refer to Section I.6.5.1, "PKMVSETPRSPM.PC."

Line 11 parses the response and stores it in SR[6].
Line 13 judges whether the response is normal. Enter the response for the case when the External machine vision has successfully received the run command in SR[7] in advance.

Line 16 receives the results.
Line 19 sets parameters for processes parsing the results.
Line 20 parses the results and stores it into respective registers set for PKMVSETPRSPM.

Lines 28 through 37 define labels to jump to when an error occurs at respective KAREL programs. To process errors, describe the process there.

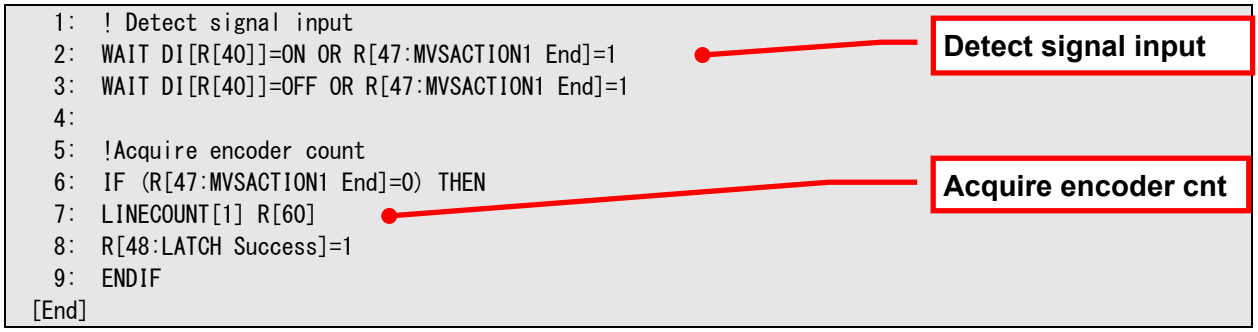
NOTE
Image snapping, finding, and sending of the results may be run by different commands, depending on the External machine vision. In such cases, add processes that send relevant commands.

I.5.4.5 Latch Program

This subsection describes an example of a latch program.

LATCH

```
  1:  ! Detect signal input
  2:  WAIT DI[R[40]]=ON OR R[47:MVSACTION1 End]=1
  3:  WAIT DI[R[40]]=OFF OR R[47:MVSACTION1 End]=1
  4:
  5:  !Acquire encoder count
  6:  IF (R[47:MVSACTION1 End]=0) THEN
  7:  LINECOUNT[1] R[60]
  8:  R[48:LATCH Success]=1
  9:  ENDIF
[End]
```



This program stores the encoder count into a register when a signal is input from the External machine vision.

Lines 2 and 3 wait until the input of the signal to get the encoder count has been detected. Enter the DI/RI number to get the encoder count that you use into R[40] in advance. And when you use RI for the DI/RI number to get the encoder count, change the condition of this wait instruction to RI.
For a case where the action program has ended without any signal input, the wait ends even when R[47], which indicates that the action program has ended, is 1.

Line 7 stores the encoder count into R[60].
Line 8 stores 1 into R[48] to indicate that the latch has succeeded.

I.5.4.6 Program for Calculating Frame Conversion Parameters

This subsection describes an example of a program that calculates frame conversion parameters for the External machine vision.

CALC_EXTPM

```
 1:  CALL PKWCEND
 2:
 3:  CALL PKCSGETID('CONV1', "CStn ID Reg" =11)
 4:
 5:  PAUSE
 6:  !Set x0,y0,x1,y1 and resume program
 7:  CALL PKMVSCLPUTP1( "Action ID" =1, "CStn ID" =R[11:CONV1 ID], "X0" =R[51:x0], "Y0" =R[52:y0])
 8:
 9:  CALL PKMVSCLPUTP2( "Action ID" =1, "CStn ID" =R[11:CONV1 ID], "X1" =R[53:x1], "Y1" =R[54:y1])
10:
11:  PAUSE
12:  !Move conveyor
13:  !Set x2,y2 and resume program
14:  CALL PKMVSCLPUTP3( "Action ID" =1, "CStn ID" =R[11:CONV1 ID], "X2" =R[55:x2], "Y2" =R[56:y2])
15:
16:  PAUSE
17:  !Touchup target1 and resume program
18:  CALL PKMVSCLSETP1( "Action ID" =1, "CStn ID" =R[11:CONV1 ID])
19:
20:  PAUSE
21:  !Touchup target2 and resume program
22:  CALL PKMVSCLSETP2( "Action ID" =1, "CStn ID" =R[11:CONV1 ID])
23:
24:  CALL PKMVSCLCALPM( "Action ID" =1)
[End]
```

Specify found pos

Specify found pos after conv move

Touch up Object 1

Touch up Object 2

Calc parameter

This program calculates frame conversion parameters for the External machine vision. The frame conversion parameters are used to convert the found positions from the External machine vision to locations in the tracking frame.
This program assumes that the found positions of the objects are manually stored into registers.

Line 1 stops the sensor so that no action will be run during teaching.

Lines 7 and 9 give the found position of Object 1, P0 (x0, y0), and that of Object 2, P1 (x1, y1), to the program. The program acquires the found positions before executing this line, and stores them into registers. After that, the conveyor is moved so that the objects are within the External machine vision's field of vision, the found position of Object 2, P2 (x2, y2), is stored into registers, and line 14 is then run.

The program moves the conveyor to a position where objects can be touched up, runs line 18 while touching up Object 1, runs line 22 while touching up Object 2, and teaches the positions of the objects in the tracking frame.

Using the location information specified in line 24, the program calculates the frame conversion parameters for the External machine vision.

I.5.4.7 Program for Setting the Reference Position

This subsection describes an example of a TP program that sets the reference position.

SET_REFPOS

```
  1:  CALL PKWCEND
  2:
  3:  CALL MVSINIT1
  4:
  5:  CALL MVSFIND1
  6:  IF R[46:MVSFIND1 Status]<>0,JMP LBL[900]
  7:  IF R[61:Num of found]<=0,JMP LBL[900]
  8:
  9:  LINECOUNT[1] R[60]
 10:  SETTRIG LNSCH[1] R[60]
 11:
 12:  !Store found position in PR
 13:  PR[27,1:Found Pos]=R[63:x]
 14:  PR[27,2:Found Pos]=R[64:y]
 15:  PR[27,3:Found Pos]=0
 16:  PR[27,4:Found Pos]=0
 17:  PR[27,5:Found Pos]=0
 18:  PR[27,6:Found Pos]=R[65:r]
 19:
 20:  !Conv fnd pos into tracking frm
 21:  CALL PKMVCNVPX2MM( "Action ID" =1, "Fnd Pos PR" =27, "Cnv Pos PR" =26)
 22:
 23:  END
 24:  LBL[900]
 25:  UALM[1]
[End]
```

Initialization

Acquire results

Set trigger

Store in PR

Frame conversion

This program acquires the found position from the External machine vision, performs frame conversion, and stores it into position registers as the reference position.
After executing this program, run the pick program to teach the robot position.

Line 1 stops the sensor so that no action will be run during teaching.

Line 3 calls a TP program that initializes connection with the External machine vision. For details, refer to "I.5.4.3 Initialization Program."

Line 5 calls a TP program that acquires results from the External machine vision. For details, refer to Section I.5.4.4, "Find Program."
Line 7 jumps the program to line 24 if the found number is 0, and generates a user alarm.

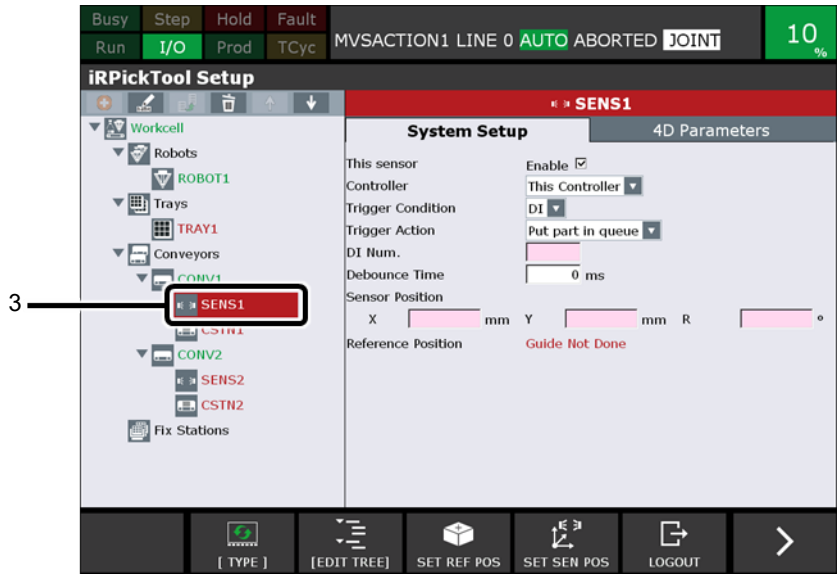
Lines 9 and 10 acquire the encoder count and set a trigger in the tracking schedule. If you use an encoder number other than 1 and a tracking schedule number other than 1, change these arguments to the numbers you use.

Lines 13 through 18 store the acquired results in position registers.
Line 21 performs frame conversion for the found position and stores it into PR[26] as the reference position.

I.5.5 Setting Sensors of Infeed Conveyors

Set the sensors of infeed conveyors.

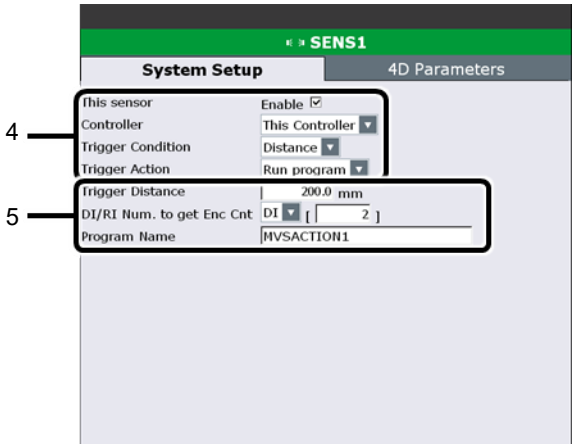
1 Press the [MENU] key on the robot controller's teach pendant.
2 From the menu, select [SETUP] → [*i*RPickTool].
3 Select [SENS1].



4 Set the following items.
[This sensor] : Check [Enable].
[Controller] : [This Controller]
[Trigger Condition] : [Distance]
[Action] : [Run program]

When [Trigger Condition] and [Action] are changed, setting items change accordingly.

5 Set the following items.
[Trigger Distance] : Set how many millimeter the conveyor advances each time when the action program is run.
[DI/RI Num. to get Enc Cnt] : Set the DI or RI number to input the signal to get the encoder count.
[Program Name] : [MVSACTION1]



> ⚠ **CAUTION**
> After these items have been set, the sensor is in a “taught” state, but setting is not yet finished. Do not start the sensor before completing “I.5.7 Calculating Frame Conversion Parameters” and “I.5.8 Setting up a Reference Position for the Infeed Conveyor, and Teaching a Robot Position.”

I.5.6 Setting Position Register Operations

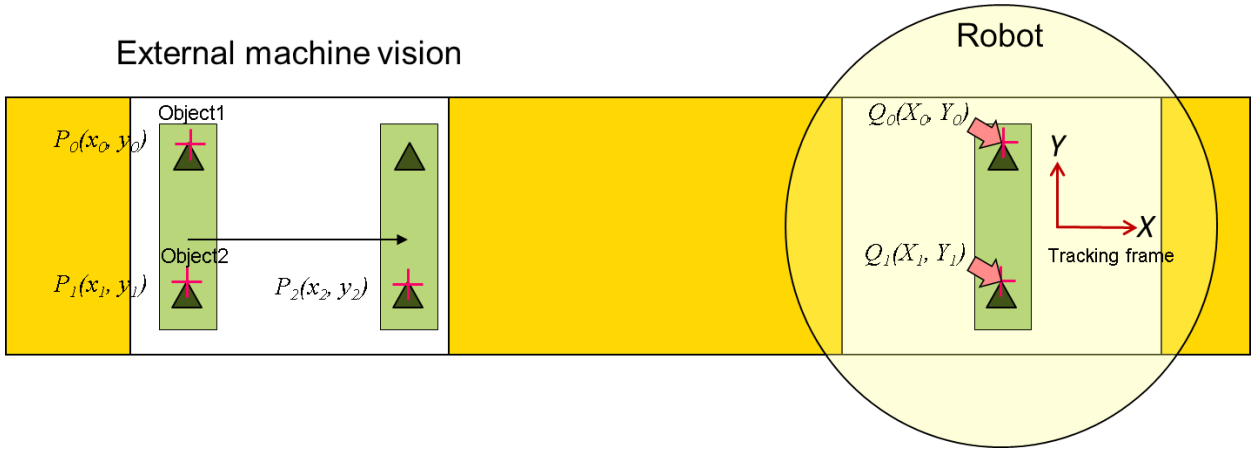
1 Press the [MENU] key on the robot controller's teach pendant.
2 From the menu, select [SYSTEM] → [Config].
3 Enable [No motion PR operate mode].

> ⚠ **CAUTION**
> When [No motion PR operate mode] is TRUE, a TP program in which the group mask has not been set can change the values of position registers. This setting is required because position register operations are performed in an action program. When you change the values of position registers from a TP program in which no group mask has been set other than in an action program, check that the change will not affect the robot's operation.

I.5.7 Calculating Frame Conversion Parameters

This procedure is for calculating the parameters to convert the found position from External machine vision to a location in the tracking frame. Follow the procedure below to run the program for calculating frame conversion parameters. For details on this program, refer to Section I.5.4.6, “Program for Calculating Frame Conversion Parameters.”

1 Switch the coordinates to the tool frame set on the tip of the touch-up pin.
2 Fix two objects within the field of External machine vision so that their locations are perpendicular to the conveyor's direction of movement.
3 Follow the procedure below to run the program to calculate frame conversion parameters.
    1 Record the found position P0 (x0, y0) of Object 1 and P1 (x1, y1) of Object 2 in the External machine vision.
    2 Specify P0 in the argument of PKMVSCLPUTP1 (a KAREL program for manually inputting found positions) and then run the program.
    3 Without moving the conveyor, specify P1 in the argument of PKMVSCLPUTP2, and then run the program.
    4 Move the conveyor a certain distance in the direction of flow, keeping the objects within the field of the External machine vision. After that, record P2 (x2, y2) as the found position of Object 2.
    5 Specify P2 in the argument of PKMVSCLPUTP3, and then run the program.
    6 Move the conveyor to a location where the objects can be touched up, and then touch up the found position Q0 (X0, Y0) of Object 1.
    7 Run PKMVSCLSETP1 (a KAREL program for acquiring the robot's position).
    8 Without moving the conveyor, touch up the found position Q1 (X1, Y1) of Object 2.
    9 Run PKMVSCLSETP2.
    10 Run PKMVSCLCALPM.



NOTE

1 To improve the accuracy, use an object that has a height close to the pick position of the part on which to calculate the frame conversion parameters.

2 If the robot touched up a point apart from the found position in External machine vision, the frame conversion parameters cannot be calculated accurately. Use an object whose found position is easy to judge.

I.5.8 Setting up a Reference Position for the Infeed Conveyor, and Teaching a Robot Position

This procedure is for setting the reference position and teaching positions to the robot. For details on the program for setting the reference position, refer to Section I.5.4.7, "Program for Setting the Reference Position."

1 Set a part within the field of the External machine vision.
2 Run the program for setting the reference position. This sets the following.
- Capture the result from the External machine vision, and then store it in the position register.
- Use PKMVCNVPX2MM to convert the frame of the result.
- Run the trigger settings to set the trigger value in the tracking schedule.

3 Move the conveyor, and then when the part enters the robot's moving range, stop it.
4 Move the robot in jog mode to the part pick position.
5 Display the pick program, and then move the cursor to the line where the pick position is taught.
6 While holding down the [SHIFT] key, press F5 [TOUCHUP].

I.6 KAREL PROGRAMS

This section describes KAREL programs used for the iRPickTool External Machine Vision Interface Add-on.

NOTE

The unique alarms occurred by KAREL programs used for the iRPickTool External Machine Vision Interface Add-on are displayed as "IRPK-201." The cause of the alarm will appear in the alarm message text. Correct the problem by referring to the KAREL program that generated the alarm and the alarm message text.

I.6.1 Communication Programs

I.6.1.1 PKMVCONNECT.PC

This program uses the designated client tag to connect to the External machine vision. Because communication is temporarily disconnected when this program is run, it is not necessary to execute PKMVDISCO in advance even if reconnecting.

Argument 1:
Specify the client tag number for which External machine vision settings have been made.

Example:

```
CALL PKMVCONNECT(“Client ID”=2)
```

I.6.1.2 PKMVDISCO.PC

This program cuts the communication connected to the External machine vision by the designated client tag. Running this KAREL program using a client tag that is yet to be connected sets off an alarm.

Argument 1:
Specify the client tag number for which External machine vision settings have been made.

Example:

```
CALL PKMVDISCO(“Client ID”=2)
```

I.6.2 Initialization Program

I.6.2.1 PKMVINIACT.PC

This program makes initial settings for a specified action. Run this program at the start of the program that acquires the result by communicating with the External machine vision. The subsequent KAREL programs should be executed by specifying the action ID set in this program as an argument.

Augment 1:
Specify the action ID to be set using 1 to 4. Specify different action IDs for different External machine visions to be communicated with.

Argument 2:
Specify the client tag numbers for which External machine vision settings have been made. You cannot specify a client ID that has already been set by another action ID.

Argument 3:
Specify a character that separates the data of the input message.

Argument 4:
Specify a newline character that is to be placed at the end of an input message.
Select the newline character from the following:
“LF”
“CR”
“CRL”
““(None)

Argument 5:

Specify the maximum number of data in an input message. When sending multiple input messages, specify the largest number of data. When the number of data set by PKMVSETSNDPM exceeds this number, the warning “IRPK-201 max_data_in is insufficient” will be generated. The maximum settable number of data is 500.

For example, specify 9 when setting the following input message:

“SUCCESS,FIND,0,0,0,0,0,0,0<LF>”

Augment 6:

Specify a character that separates data in an output message. Do not specify a character that is included in the data.

Argument 7:

Specify a newline character that is to be placed at the end of an output message.

Select the newline character from the following:

“LF”

“CR”

“CRLF”

Argument 8:

Specify the maximum number of data in an output message. When receiving multiple output messages, specify the largest number of data. When the number of data received at the time of running PKMVRCVMSG exceeds this number, the warning “IRPK-201 max_data_out is insufficient” will be generated. The maximum settable number of data is 500.

Example:

```
CALL PKMVINIACT( “Action ID”=1, “Client ID”=2, “In Msg Delimiter”=',', “In Msg Newline
char”='LF', “In Msg Max num data”=1, “Out Msg Delimiter”=',', “Out Msg Newline char”='LF',
 “Out Msg Max num data”=11)
```

I.6.3 Programs for Sending Input Messages

I.6.3.1 PKMVSETSNDPM.PC

This program sets the input message data sent by PKMVSENDMSG. To set more than 28 data-specifying arguments, continue running this program to keep making settings from previously set data. Exceeding the maximum number of data for input messages set by PKMVINIACT generates the warning “IRPK-201 max_data_in is insufficient.” Even running PKMVSENDMSG will not delete the data set by this program. To send only the same input message, executing this program once at the beginning will do. To set a different input message, before running this program, delete the set input message by executing PKMVDELSNDPM.

Argument 1:

Specify the action ID.

Argument 2:

Specify the register number that stores the status of PKMVSENDMSG.

Arguments 3 to 30 (Arguments 4 to 30 are optional.):

Specify the data of the input message. An integer, a real number, or a string can be used. When specifying a real number, a number up to the third decimal place is transmitted. When specifying a string, the maximum settable string length is 34.

Example:

When the following program is executed, “RUNFIND” and “1.234” are set as the data of the input message.

```
CALL PKMVSETSNDPM(“Action ID”=1,“Stat Reg”=10,'RUNFIND',1.2345)
```

I.6.3.2 PKMVDELSNDPM.PC

This program deletes input message sent by PKMVSENDMSG. When setting different input message, execute this program before PKMVSETSNDPM.

Argument 1:
Specify the action ID.

Example:

```
CALL PKMVDELSNDPM(“Action ID”=1)
```

I.6.3.3 PKMVSENDMSG1 to 4.PC

This program sends an input message to the External machine vision. The digit at the end of the program name corresponds to the action ID. This program separates the data set by PKMVSETSNDPM with delimiters, and then sends the message with the newline character at the end. Use PKMVINIACT to set the delimiter and the newline character.

After execution, either one of the following values will be stored in the register as the status.
0: Success
12: Communication error

There is no argument.

Example:

```
CALL PKMVSENDMSG1
```

I.6.4 Programs for Receiving Output Messages

I.6.4.1 PKMVSETRCVPM.PC

This program sets the timeout period for PKMVRCVMSG. Even running PKMVRCVMSG does not delete the data set by this program. If you are not changing the timeout period, running this program once at the beginning will do.

Argument 1:
Specify the action ID.

Argument 2:
Specify the timeout period for receiving output message. The unit is [ms].

Argument 3:
Specify the register number for storing the status of PKMVRCVMSG.

Example:

```
CALL PKMVSETRCVPM(“Action ID”=1,“Timeout (ms)”=5000,“Stat Reg”=11)
```

I.6.4.2 PKMVRCVMSG1 to 4.PC

This program receives output messages sent by the External machine vision. The digit at the end of the program name corresponds to the action ID. When this program is executed, it receives one output message delimited by newline character. PKMVINIACT sets the newline character. Executing this program deletes the previously received message. To store the received output message in the register, run PKMVPRSMSG. Receiving more data than the maximum number of data for output messages set by PKMVINIACT generates the warning “IRPK-201 max_data_out is insufficient.” After executing this program, failing to receive an output message within the timeout period generates the warning “IRPK-201 Receive message timeout.” Receiving an invalid newline character or data comprising 255 or more characters generates the warning “IRPK-201 Receive data is invalid.”

After execution, either one of the following values will be stored as the status.
0: Success
11: Timeout
12: Communication error

There is no argument.

Example:

```
CALL PKMVRCVMSG1
```

I.6.5 Programs for Parsing Output Messages

I.6.5.1 PKMVSETPRSPM.PC

This program sets the parameters for parses by PKMVPRSMSG. Set the data type of the output message to be received and the register number in which to store the data in a parameter. If setting parameters with more than 14 data-specifying arguments, continue running this program to keep making settings starting with the previously set data. The maximum settable number of data is 500 (this maximum settable number differs from the maximum number of data for output messages set by PKMVINIACT). Even running PKMVPRSMSG does not delete the data set by this program. To set a different parameter, before running this program, delete the set parameter by executing PKMVDELPRSPM.

Argument 1:
Specify the action ID.

Argument 2:
Specify the register number that stores the status of PKMVPRSMSG.

Arguments 3, 5, 7, ... , 29 (optional):
Specify a data type. Select one from the data types below. This argument and the register number to be set next form a parameter set for one data.
1: Integer
2: Real number
3: String

Arguments 4, 6, 8, ... , 30 (optional):
Specify the register number that stores data. If the preceding argument specifies the data type as an integer or a real number, the data is stored in the register, and if a string, in the string register.

Example:

The following sets a parameter to store output message “SUCCESS (string), Model ID (integer), X (real number) <LF>“ in SR[5], R[10], and R[11].

```
CALL PKMVSETPRSPM(“Action ID”=1,“Stat Reg”=12,3,5,1,10,2,11)
```

I.6.5.2 PKMVDELPRSPM.PC

This program deletes the parameter set by PKMVSETPRSPM. When setting a parameter for a different message received, run this program before PKMVSETPRSPM.

Argument 1:

Specify the action ID.

Example:

```
CALL PKMVDELPRSPM(“Action ID”=1)
```

I.6.5.3 PKMVPRSMSG1 to 4.PC

This program parses output messages received by PKMVRCVMSG, and then stores the data in the register. The digit at the end of the program name corresponds to the action ID.
Every time this is run, it extracts data from output messages received by PKMVRCVMSG sequentially from the head, converts the data type in the order of parameters set by PKMVSETPRSPM, and then stores each data in the register. If the set data type differs from the parsed type, the program generates the warning “IRPK-201 Data type is invalid.” If empty data is parsed, it generates the warning “IRPK-201 Receive data is invalid.”

> ⚠ **CAUTION**
> If the number of data of the parsed output message is smaller than the number of parameters set by PKMVSETPRSPM, the next time you run PKMVPRSMSG after executing PKMVRCVMSG, the analysis will use the parameters from the previous analysis. For this reason, if an analysis has stopped due to an error, the parameters to be used for the subsequent analysis may deviate from the correct values, causing another error. Therefore, in principle, run PKMVDELPRSPM and PKMVSETPRSPM to reset the parameters before executing PKMVPRSMSG.

After execution, one of the following values will be stored as the status.
0: Success
1: The number of data of the parsed output message is larger than the number of parameters set by PKMVSETPRSPM.
2: The number of data of the prased output message is smaller than the number of parameters set by PKMVSETPRSPM.
3: No data exist in the parsed output message.
4: Failure

There is no argument.

Example:

```
CALL PKMVPRSMSG1
```

I.6.6 Programs for Calculating Frame Conversion Parameters

I.6.6.1 PKMVSCLPUTP1.PC

This program inputs the found position of the first object.

Argument 1:
Specify the action ID.

Argument 2:
Specify the conveyor station ID.

Argument 3:
Specify the value of x0.

Argument 4:
Specify the value of y0.

Example:

```
CALL PKMVSCLPUTP1(“Action ID”=1,“CStn ID”=R[6:CSTN1 ID],
“X0”=R[51:x0],“Y0”=R[52:y0])
```

I.6.6.2 PKMVSCLPUTP2.PC

This program inputs the found position of the second object. After executing PKMVSCLPUTP1, run this program without moving the conveyor.

Argument 1:
Specify the action ID.

Argument 2:
Specify the conveyor station ID.

Argument 3:
Specify the value of x1.

Argument 4:
Specify the value of y1.

Example:

```
CALL PKMVSCLPUTP2(“Action ID”=1,“CStn ID”=R[6:CSTN1 ID],
“X1”=R[53:x1],“Y1”=R[54:y1])
```

I.6.6.3 PKMVSCLPUTP3.PC

This program inputs the found position of the second object after moving the conveyor. After running PKMVSCLPUTP2, to execute this program, move the conveyor a certain distance while keeping the objects within the field of the External machine vision, and then specify the found position of the second object in the argument.

Argument 1:
Specify the action ID.

Argument 2:
Specify the conveyor station ID.

Argument 3:
Specify the value of x2.

Argument 4:
Specify the value of y2.

Example:

```
CALL PKMVSCLPUTP3(“Action ID”=1,“CStn ID”=R[6:CSTN1 ID],
 “X2”=R[55:x2],“Y2”=R[56:y2])
```

I.6.6.4 PKMVSCLSETP1.PC

This program teaches the location of the first object in the tracking frame. Run this program with the state that the robot touches up the found position of the first object.

Argument 1:
Specify the action ID.

Argument 2:
Specify the conveyor station ID.

Example:

```
CALL PKMVSCLSETP1(“Action ID”=1,“CStn ID”=R[6:CSTN1 ID])
```

I.6.6.5 PKMVSCLSETP2.PC

This program teaches the location of the second object in the tracking frame. After executing PKMVSCLSETP1, without moving the conveyor, execute this program with the state that the robot touches up the found position of the second object.

Argument 1:
Specify the action ID.

Argument 2:
Specify the conveyor station ID.

Example:

```
CALL PKMVSCLSETP2(“Action ID”=1,“CStn ID”=R[6:CSTN1 ID])
```

I.6.6.6 PKMVSCLCALPM.PC

This program calculates the frame conversion parameters for the External machine vision. Run this program after executing all the other programs that calculate frame conversion parameters.

Argument 1:
Specify the action ID.

Example:

```
CALL PKMVSCLCALPM(“Action ID”=1)
```

I.6.7 Programs for Converting Found Positions

I.6.7.1 PKMVCNVPX2MM.PC

This program converts the found position on the External machine vision to the position in the tracking frame. This program can be executed after completing calculation of frame conversion parameters.

Argument 1:
Specify the action ID.

Argument 2:
Specify the position register number that stores the found position on the External machine vision.

Argument 3:
Specify the position register number that stores the frame-converted location. If the same value as argument 2 is specified, the specified found position is overwritten with the value after conversion.

Example:

```
CALL PKMVCNVPX2MM(“Action ID”=1,“Fnd Pos PR”=2,“Cnv Pos PR”=1)
```


I.6.7.2 PKMVCALOFS2D.PC

This program calculates the offset from the found position and the reference position. Specify the frame-converted location by using PKMVCNVPX2MM. This program cannot calculate offsets on anything other than the *i*RPickTool External Machine Vision Interface Add-on.

Argument 1:
Specify the position register number that stores the found position.

Argument 2:
Specify the position register number that stores the reference position.

Argument 3:
Specify the position register number that stores the offset.

Example:

```
CALL PKMVCALOFS2D(“Fnd Pos PR”=2,“Ref Pos PR”=1,“Offset PR”=3)
```

I.6.7.3 PKMVSETVR.PC

This program stores a specified value in the vision register. It is used to queue parts using PKCSPUTQUE.

Argument 1:
Specify the vision register number in which to store the value.

Argument 2:
Specify the position register number that stores the found position.

Argument 3:
Specify the position register number that stores the offset value.

Argument 4:
Specify the register number that stores the model ID.

Argument 5:
Specify the register number that stores the encoder count.

Arguments 6 to 15 (optional):
Specify the register number that stores each of measurements 1 to 10. If a measured value is omitted, 0.0 is stored in the corresponding register.

Example:
Specify the following values in vision register 1:
Found position: PR[2]
Offset: PR[3]
Model ID: R[31]
Encoder count: R[28]
Measurement 1: R[38]

```
CALL PKMVSETVR(“VR”=1,“Fnd Pos PR”=2,“Offset PR”=3,“Model ID Reg”=31,
“Enc Count Reg”=28,38)
```

I.6.8 Other Programs

I.6.8.1 PKMVWAITTASK.PC

This program stands by until the specified task completes, and is used for waiting until a program running in the background ends. If the specified task does not exist, this program ends immediately.

Argument 1:
The name of the task whose end it waits for.

Example:

```
CALL PKMVWAITTASK(“Task Name”=' LATCH')
```

I.7 SETTING EXAMPLE APPROPRIATE TO THE USAGE

I.7.1 Detect Multiple Parts in One Snap

This section describes the system startup procedures for External machine vision which sends up to two detection results in one time detection, as shown in the message format below.

Table I.7.1 Message Format

Input message (Controller → External machine vision)	RUNFIND (string) <CR>
Response to an input message (External machine vision → Controller)	When normal : OK (string) <CR> When abnormal : ER (string) <CR>
Output message (External machine vision → Controller)	Number of detections (Integer), Model ID 1 (Integer), X1 (Real), Y1 (Real), R1 (Real), Measurement 1.1 (Real), Model ID 2 (Integer), X2 (Real), Y2 (Real), R2 (Real), Measurement 2.1 (Real)<CR>

With this message format, it is necessary to send the output message without omitting the second detection result data even if there is only one detection result. Therefore, the following process is required when there is only one detection result.

- Stores 1 in "Num of found"
- Stores 0 in the data after "Model ID 2".

The overall startup procedure is the same as in "I.5 STARTUP PROCEDURES", but change the Action Program and Find Program to the following, respectively.

Action Program

MVSACTION1

```
  1:  R[47:MVSACTION1 End]=0      ;
  2:   ;
  3:  !Use ACCUTRIG ;
  4:  ACCUTRIG LNSCH[1] ;
  5:   ;
  6:  R[48:LATCH Success]=0      ;
  7:  RUN LATCH ;
  8:   ;
  9:  CALL MVSFIND1      ;
 10:  IF R[46:MVSFIND1 Status]<>0,JMP LBL[990] ;
 11:   ;
 12:  IF R[48:LATCH Success]=0,JMP LBL[900] ;
 13:   ;
 14:  !Store found position1 in PR ;
 15:  PR[27,1:Found Pos]=R[63:x1]      ;
 16:  PR[27,2:Found Pos]=R[64:y1]      ;
 17:  PR[27,3:Found Pos]=0      ;
 18:  PR[27,4:Found Pos]=0      ;
 19:  PR[27,5:Found Pos]=0      ;
 20:  PR[27,6:Found Pos]=R[65:r1]      ;
 21:   ;
 22:  !Conv fnd pos into tracking frm ;
 23:  CALL PKMVCNVPX2MM("Action ID"=1,"Fnd Pos PR"=27,"Cnv Pos PR"=27) ;
 24:  !Calculate offset ;
 25:  CALL PKMVCALOFS2D("Fnd Pos PR"=27,"Ref Pos PR"=26,"Offset PR"=28) ;
 26:   ;
```

Put result 1 in queue

```
27:  !Put result into queue ;
28:  CALL PKMVSETVR("VR"=5,"Fnd Pos PR"=27,"Offset PR"=28,"Model ID
     Reg"=62,"Enc Count Reg"=60,66) ;
29:  CALL PKCSPUTQUE("CStn ID"=R[11:CONV1 ID],"Offset VR"=5,"OverlapStat
     Reg"=49) ;
30:    ;
31:  IF R[61:Num of found]<2,JMP LBL[990] ;
32:  !Store found position2 in PR ;
33:  PR[27,1:Found Pos]=R[73:x2]       ;
34:  PR[27,2:Found Pos]=R[74:y2]       ;
35:  PR[27,3:Found Pos]=0       ;
36:  PR[27,4:Found Pos]=0       ;
37:  PR[27,5:Found Pos]=0       ;
38:  PR[27,6:Found Pos]=R[75:r2]       ;
39:    ;
40:  !Conv fnd pos into tracking frm ;
41:  CALL PKMVCNVPX2MM("Action ID"=1,"Fnd Pos PR"=27,"Cnv Pos PR"=27) ;
42:  !Calculate offset ;
43:  CALL PKMVCALOFS2D("Fnd Pos PR"=27,"Ref Pos PR"=26,"Offset PR"=28) ;
44:    ;
45:  !Put result into queue ;
46:  CALL PKMVSETVR("VR"=5,"Fnd Pos PR"=27,"Offset PR"=28,"Model ID
     Reg"=62,"Enc Count Reg"=60,66) ;
47:  CALL PKCSPUTQUE("CStn ID"=R[11:CONV1 ID],"Offset VR"=5,"OverlapStat
     Reg"=49) ;
48:  JMP LBL[990] ;
49:    ;
50:  LBL[900:LATCH ERORR] ;
51:  R[46:MVSFIND1 Status]=5       ;
52:    ;
53:  LBL[990] ;
54:  R[47:MVSACTION1 End]=1       ;
55:  CALL PKMVWAITTASK("Task Name"='LATCH') ;
```

Ends when there is 1 result

Put result 2 in queue

Find Program

MVSFIND1

```
 1:  R[46:MVSFIND1 Status]=1       ;
 2:    ;
 3:  !Send run command ;
 4:  CALL PKMVSENDMSG1       ;
 5:    ;
 6:  !Receive response ;
 7:  CALL PKMVRCVMSG1       ;
 8:    ;
 9:  CALL PKMVDELPRSPM("Action ID"=1) ;
10:  CALL PKMVSETPRSPM("Action ID"=1,"Stat Reg"=45,3,6) ;
11:  CALL PKMVPRSMSG1       ;
12:  IF R[45:PRSMSG Status]<>0,JMP LBL[900] ;
13:  IF SR[6]<>SR[7],JMP LBL[910] ;
14:    ;
15:  !Receive results ;
16:  CALL PKMVRCVMSG1       ;
17:    ;
18:  !Parse found position1 ;
19:  CALL PKMVDELPRSPM("Action ID"=1) ;
```

Parse the first result

```
20:  CALL PKMVSETPRSPM("Action ID"=1,"Stat Reg"=45,1,61,1,62,2,63,2,64,2,65,2,66) ;
21:  CALL PKMVPRSMSG1      ;
22:   ;
23:  !Parse found position2 ;
24:  CALL PKMVDELPRSPM("Action ID"=1) ;
25:  CALL PKMVSETPRSPM("Action ID"=1,"Stat Reg"=45,1,72,2,73,2,74,2,75,2,76) ;
26:  CALL PKMVPRSMSG1      ;
27:  IF R[45:PRSMSG Status]<>0,JMP LBL[900] ;
28:   ;
29:  IF R[61:Num of found]<=0,JMP LBL[920] ;
30:   ;
31:  R[46:MVSFIND1 Status]=0      ;
32:  END ;
33:   ;
34:  LBL[900:PKMVPRSMSG ERROR] ;
35:  R[46:MVSFIND1 Status]=2      ;
36:  JMP LBL[990] ;
37:   ;
38:  LBL[910:RESPONSE ERROR] ;
39:  R[46:MVSFIND1 Status]=3      ;
40:  JMP LBL[990] ;
41:   ;
42:  LBL[920:NUM OF FND IS 0] ;
43:  R[46:MVSFIND1 Status]=4      ;
44:   ;
45:  LBL[990] ;
```

Parse the second result

I

I.8 TROUBLESHOOTING

This chapter describes common issues in systems that use the *i*RPickTool External Machine Vision Interface Add-on as well as measures to be taken.

I.8.1 The Robot Can Not be Connected to an External Machine Vision

Cause 1: The client tag is not set properly.
Remedy 1: By referring to Section I.5.2.2, "Setting up the Client Tag," check that the IP address and the port number of the External machine vision are set properly in the client tag. Also check that the [Current State] of the client tag is [DEFINED].

Cause 2: The Ethernet port is not set.
Remedy 2: Check that TCP/IP setting is completed for the Ethernet port connected to the External machine vision.

I.8.2 The External Machine Vision Does Not Execute Snap or Find

Cause 1: The input message (run command) is invalid.
Remedy 1: Check that the input message specified by PKMVSETSENDPM is the same as the run command for the External machine vision. Check that the External machine vision can recognize the delimiter and the newline character of the input message specified by PKMVINIACT.

Cause 2: The External machine vision is not in a state where it can receive commands.
Remedy 2: The External machine vision is not in a state where it can receive commands. Refer to the operation manual of the relevant vision system manufacturers for how to set it.

I.8.3 Timeout Occurs in PKMVRCVMSG

Cause 1: The newline character of the output message is invalid.
Remedy 1: Check that the newline character of the output message is identical to that specified by PKMVINIACT.

Cause 2: The timeout period is too short.
Remedy 2: Set the timeout period that is longer than the find time of the External machine vision.

Cause 3: The input message to send the result is not transmitted.
Remedy 3: Some types of External machine vision may use different commands for image snapping, finding, and sending results. Check that the robot controller transmits an input message that initiates the sending of the result.

I.8.4 PKMVPRSMSG Failure

Cause 1: The output message's analysis parameters are invalid.
Remedy 1: Check that the parameters set by PKMVSETPRSPM are the same as the output message format transmitted by the External machine vision. Check that the output message's delimiter set by PKMVINIACT is valid.

Cause 2: Analysis parameters have deviated.
Remedy 2: If the number of data of parsed output messages is smaller than the number of parameters set by PKMVSETPRSPM, the next time you run PKMVPRSMSG after executing PKMVRCVMSG, the analysis will use the parameters from the previous analysis. For this reason, if an analysis has stopped due to an error, the parameters to be used for the subsequent analysis may deviate from the correct values, causing another error. Therefore, in principle, run PKMVDELPRSPM and PKMVSETPRSPM to reset the parameters before executing PKMVPRSMSG.

Cause 3: The received output message differs from the output message that set the parameters.
Remedy 3: Some types of External machine vision may return a response as an output message when they receive a command. The robot controller may then receive output message that is not the result. Check the output message transmitted by the External machine vision.

I.8.5 Latch Failure

Cause 1: The DI/RI number to get the encoder count is invalid.
Remedy 1: Check that the number of the signal to get encoder count and the number to detect the signal input in the latch program are valid.

Cause 2: The signal to get the encoder count is not output.
Remedy 2: Check that the External machine vision outputs the signal to get the encoder count at the time of image snapping. Check that cables are properly connected between the External machine vision and the robot controller.

Cause 3: The signal to get the encoder count takes long to its falling edge.
Remedy 3: If the output of the signal to get the encoder count is delayed or its output width is large, the analysis of the result finishes before the latch program detects the falling of the signal. This may cause a judgment of a latch failure. The signal to get the encoder count must be output at the same time as image snapping and its width must be short, or a wait instruction must be added before the instruction to judge whether latch of the action program has succeeded.

I.8.6 Invalid Part Position

Cause 1: The calculation of frame conversion parameters is not correct.
Remedy 1: Referring to “I.5.7 Calculating Frame Conversion Parameters,” recalculate the frame conversion parameters. In addition, after doing these calculations, reset the reference position.

Cause 2: The reference position is not set correctly.
Remedy 2: By referring to “I.5.8 Setting up a Reference Position for the Infeed Conveyor, and Teaching a Robot Position,” reset the reference position. At the start of resetting the reference position, run PKWCEND to check that the sensor has finished.

Cause 3: The timing of the signal to get the encoder count varies.
Remedy 3: If the output timing of the signal to get the encoder count varies, you cannot obtain the correct conveyor position when snapping images. Consider using another signal that has little variation in output timing.

Cause 4: The accuracy trigger is not enabled.
Remedy 4: Before sending an input message, execute the accuracy trigger command to enable the accuracy trigger. Check that the DI/RI number to get the encoder count of the sensor object is valid.

Cause 5: Tracking frame has not been set correctly.
Remedy 5: Refer to “6.12.2 Adjustment of the Tracking Frame Error” for details.

INDEX

<Q>

<R>

<S>

<T>

<U>

<V>

<W>

REVISION RECORD

Edition	Date	Contents
03	Oct., 2021	• Support for 7DF5/14 (V9.40P/14) added. • Support for R-30*i*B Mini Plus Controller added.
02	Jan., 2020	• Support for 7DF1/22 (V9.10P/22) added. • Support for R-30*i*B Mate Plus/R-30*i*B Compact Plus Controller added.
01	Sep., 2017	

B-83924EN/03