

FANUC Robot **series**

R-30*i*B/R-30*i*B Mate CONTROLLER

***i*RPickTool**

OPERATOR'S MANUAL

- **Original Instructions**

Thank you very much for purchasing FANUC Robot.

Before using the Robot, be sure to read the "FANUC Robot SAFETY HANDBOOK (B-80687EN)" and understand the content.

- No part of this manual may be reproduced in any form.
- The appearance and specifications of this product are subject to change without notice.

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 your local FANUC representative 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 ensure 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.

In addition, refer to the "FANUC Robot SAFETY HANDBOOK (B-80687EN)".

1 DEFINITION OF USER

The user can be defined as follows.

Operator:

- Turns ON/OFF power to the robot
- Starts the robot program from the operator's panel

Programmer:

- Operates the robot
- Teaches the robot inside the safety fence

Maintenance engineer:

- Operates the robot
- Teaches the robot inside the safety fence
- Performs maintenance (repair, adjustment, replacement)

- Operator is not allowed to work in the safety fence.
- Programmers and maintenance engineers are allowed to work in the safety fence. The work inside the safety fence includes lifting, setting, teaching, adjustment, maintenance, etc.
- To work inside the safety fence, the person must receive a professional training for the robot.

During the operation, programming, and maintenance of your robotic system, the programmer, operator, and maintenance engineer should take additional care of their safety by wearing the following safety items.

- Adequate clothes for the operation
- Safety shoes
- A 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.

- Check this manual thoroughly, and keep it handy for the future reference.

3 SAFETY OF THE USER

User safety is the primary safety consideration. Because it is very dangerous to enter the operating space of the robot during automatic operation, adequate safety precautions must be observed. The following lists the general safety precautions. Careful consideration must be made to ensure user safety.

- (1) Have the robot system users attend the training courses held by FANUC.

FANUC provides various training courses. Contact our sales office for details.

- (2) Even when the robot is stationary, it is possible that the robot is still in a ready to move state, and is waiting for a signal. In this state, the robot is regarded as still in motion. To ensure user safety, provide the system with an alarm to indicate visually or aurally that the robot is in motion.
- (3) Install a safety fence with a gate so that no user can enter the work area without passing through the gate. Install an interlocking device, a safety plug, and so forth in the safety gate so that the robot is stopped as the safety gate is opened.

The controller is designed to receive this interlocking signal of the door switch. When the gate is opened and this signal received, the controller stops the robot (Please refer to "STOP TYPE OF ROBOT" in "SAFETY PRECAUTIONS" for detail of stop type). For connection, see Fig. 3 (b).

- (4) Provide the peripheral equipment with appropriate earth (Class A, Class B, Class C, and Class D).
- (5) Try to install the peripheral equipment outside the robot operating space.
- (6) Draw an outline on the floor, clearly indicating the range of the robot operating space, including the tools such as a hand.
- (7) Install a mat switch or photoelectric switch on the floor with an interlock to a visual or aural alarm that stops the robot when a user enters the work area.
- (8) If necessary, install a safety lock so that no one except the user in charge can turn on the power of the robot.

The circuit breaker installed in the controller is designed to disable anyone from turning it on when it is locked with a padlock.

- (9) When adjusting each peripheral equipment independently, be sure to turn off the power of the robot.
- (10) Operators should be ungloved while manipulating the operator panel or teach pendant. Operation with gloved fingers could cause an operation error.
- (11) Programs, system variables, and other information can be saved on memory card or USB memories. Be sure to save the data periodically in case the data is lost in an accident. (refer to Controller OPERATOR'S MANUAL.)
- (12) The robot should be transported and installed by accurately following the procedures recommended by FANUC. Wrong transportation or installation may cause the robot to fall, resulting in severe injury to workers.
- (13) In the first operation of the robot after installation, the operation should be restricted to low speeds. Then, the speed should be gradually increased to check the operation of the robot.
- (14) Before the robot is started, it should be checked that no one is inside the safety fence. At the same time, a check must be made to ensure that there is no risk of hazardous situations. If detected, such a situation should be eliminated before the operation.
- (15) When the robot is used, the following precautions should be taken. Otherwise, the robot and peripheral equipment can be adversely affected, or workers can be severely injured.
 - Avoid using the robot in a flammable environment.
 - Avoid using the robot in an explosive environment.
 - Avoid using the robot in an environment full of radiation.
 - Avoid using the robot under water or at high humidity.
 - Avoid using the robot to carry a person or animal.
 - Avoid using the robot as a stepladder. (Never climb up on or hang from the robot.)
 - Outdoor
- (16) When connecting the peripheral equipment related to stop (safety fence etc.) and each signal (external emergency, fence etc.) of robot, be sure to confirm the stop movement and do not take the wrong connection.
- (17) When preparing footstep, please consider security for installation and maintenance work in high place according to Fig. 3 (c). Please consider footstep and safety belt mounting position.

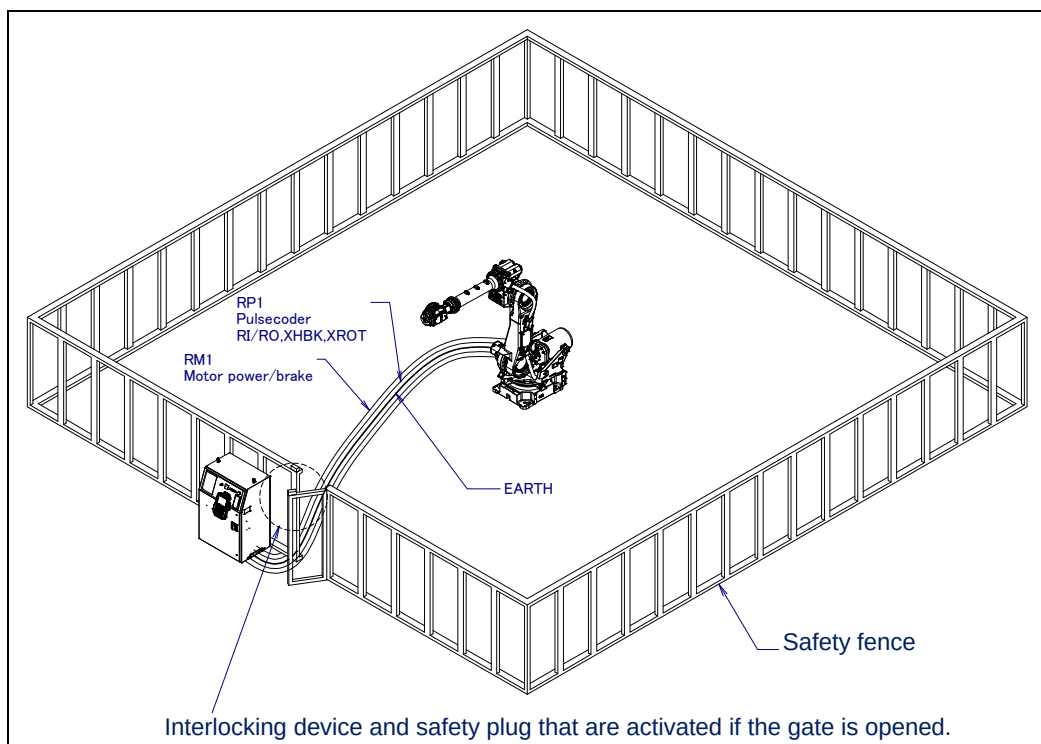


Fig. 3 (a) Safety fence and safety gate

**WARNING**

When you close a fence, please confirm that there is not a person from all directions of the robot.

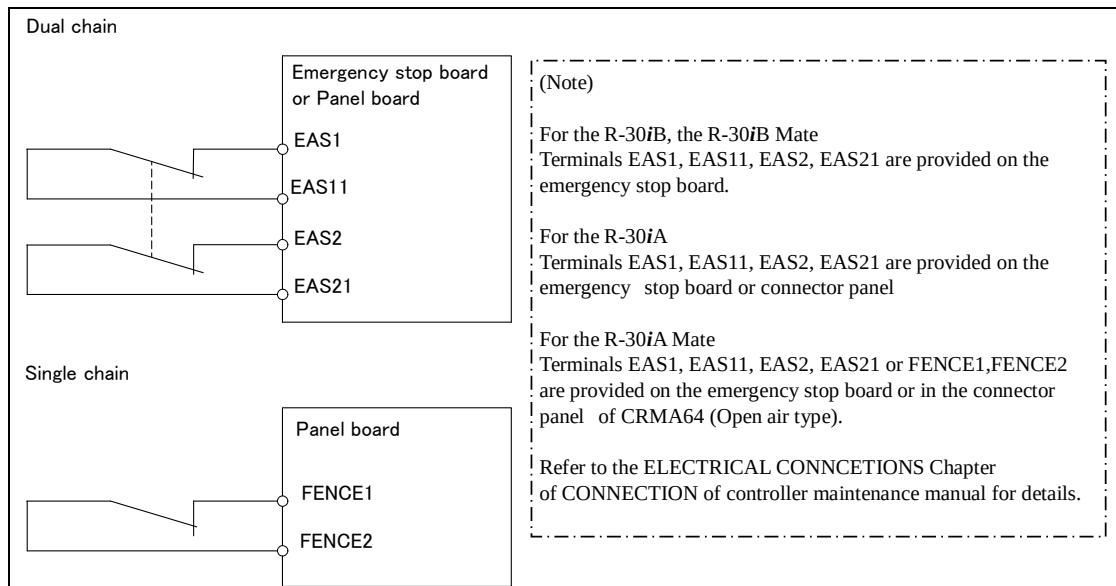


Fig. 3 (b) Connection diagram for the signal of safety fence

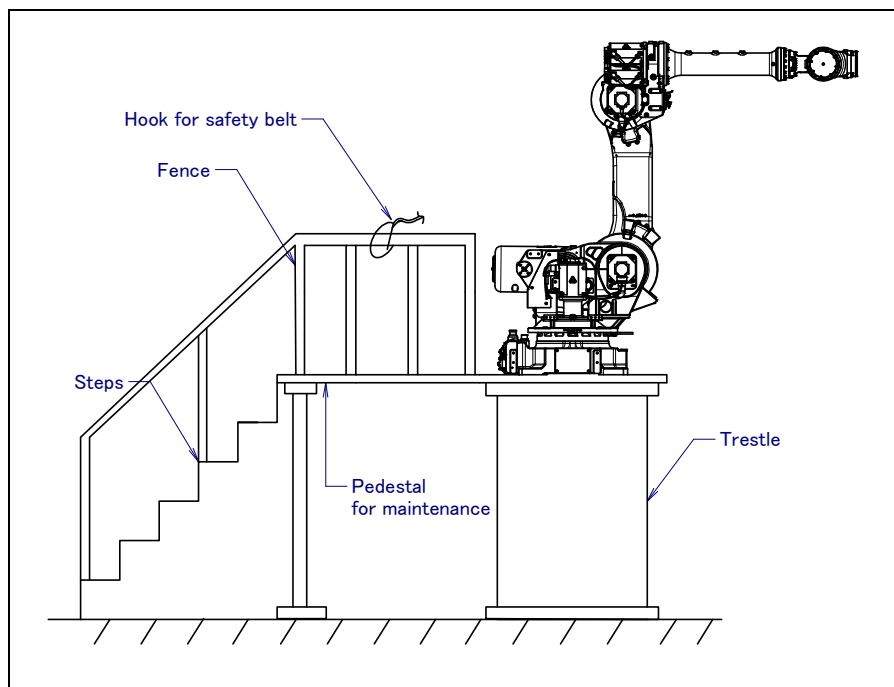


Fig. 3 (c) Pedestal for maintenance

3.1 SAFETY OF THE OPERATOR

An operator refers to a person who turns on and off the robot system and starts a robot program from, for example, the operator panel during daily operation.

Operators cannot work inside of the safety fence.

- (1) If the robot does not need to be operated, turn off the robot controller power or press the EMERGENCY STOP button during working.

- (2) Operate the robot system outside the operating space of the robot.
- (3) Install a safety fence or safety door to avoid the accidental entry of a person other than an operator in charge or keep operator out from the hazardous place.
- (4) Install the EMERGENCY STOP button within the operator's reach.

The robot controller is designed to be connected to an external EMERGENCY STOP button. With this connection, the controller stops the robot operation (Please refer to "STOP TYPE OF ROBOT" in "SAFETY PRECAUTIONS" for detail of stop type) when the external EMERGENCY STOP button is pressed. See the diagram below for connection.

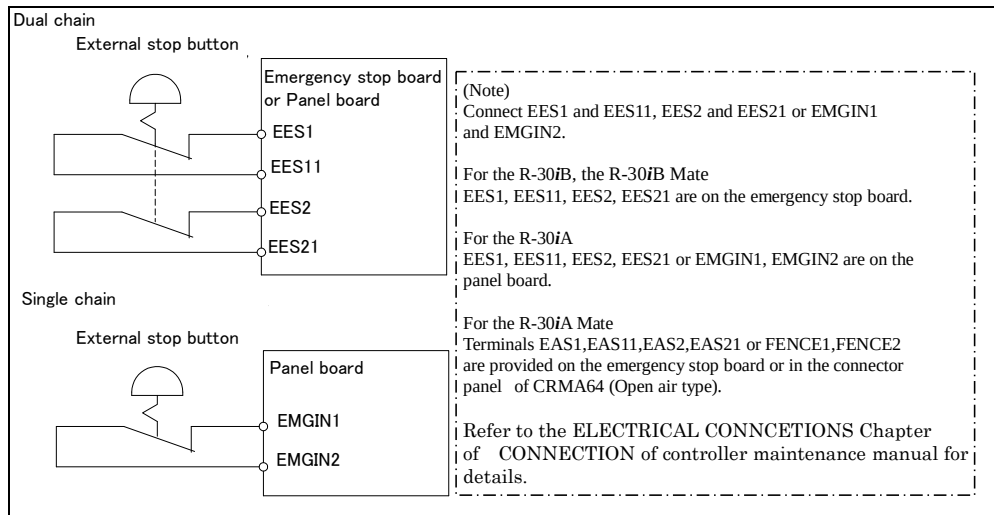


Fig. 3.1 Connection diagram for external emergency stop button

3.2 SAFETY OF THE PROGRAMMER

While teaching the robot, the operator may need to enter the robot operation area. The programmer must ensure the safety especially.

- (1) Unless it is specifically necessary to enter the robot operating space, carry out all tasks outside the operating space.
- (2) Before teaching the robot, check that the robot and its peripheral equipment are all in the normal operating condition.
- (3) If it is inevitable to enter the robot operating space to teach the robot, check the locations, settings, and other conditions of the safety devices (such as the EMERGENCY STOP button, the DEADMAN switch on the teach pendant) before entering the area.
- (4) The programmer must be extremely careful not to let anyone else enter the robot operating space.
- (5) Programming should be done outside the area of the safety fence as far as possible. If programming needs to be done inside the safety fence, the programmer should take the following precautions:
 - Before entering the area of the safety fence, ensure that there is no risk of dangerous situations in the area.
 - Be prepared to press the emergency stop button whenever necessary.
 - Robot motions should be made at low speeds.
 - Before starting programming, check the whole robot system status to ensure that no remote instruction to the peripheral equipment or motion would be dangerous to the user.

Our operator panel is provided with an emergency stop button and a key switch (mode switch) for selecting the automatic operation mode (AUTO) and the teach modes (T1 and T2). Before entering the inside of the safety fence for the purpose of teaching, set the switch to a teach mode, remove the key from the mode switch to prevent other people from changing the operation mode carelessly, then open the safety gate. If the safety gate is opened with the automatic operation mode set, the robot stops (Please refer to "**STOP TYPE OF ROBOT**" in SAFETY PRECAUTIONS for detail of stop type). After the switch is set to a teach mode, the safety gate is disabled. The programmer should understand that the safety gate is disabled and is responsible for keeping other people from entering the inside of the safety fence.

Teach pendant is provided with a switch to enable/disable robot operation from teach pendant and DEADMAN switch as well as emergency stop button. These button and switch function as follows:

- (1) Emergency stop button: Causes the stop of the robot (Please refer to "STOP TYPE OF ROBOT" in SAFETY PRECAUTIONS for detail of stop type) when pressed.
- (2) DEADMAN switch: Functions are different depending on the teach pendant enable/disable switch setting status.

(a) Enable: Servo power is turned off and robot stops when the operator releases the DEADMAN switch or when the operator presses the switch strongly.

(b) Disable: The DEADMAN switch is disabled.

(Note) The DEADMAN switch is provided to stop the robot when the operator releases the teach pendant or presses the pendant strongly in case of emergency. The R-30iB/R-30iB Mate/R-30iA/R-30iA Mate employs a 3-position DEADMAN switch, which allows the robot to operate when the 3-position DEADMAN switch is pressed to its intermediate point. When the operator releases the DEADMAN switch or presses the switch strongly, the robot stops immediately.

The operator's intention of starting teaching is determined by the controller through the dual operation of setting the teach pendant enable/disable switch to the enable position and pressing the DEADMAN switch. The operator should make sure that the robot could operate in such conditions and be responsible in carrying out tasks safely.

Based on the risk assessment by FANUC, number of operation of DEADMAN switch should not exceed about 10000 times per year.

The teach pendant, operator panel, and peripheral device interface send each robot start signal. However the validity of each signal changes as follows depending on the mode switch and the DEADMAN switch of the operator panel, the teach pendant enable switch and the remote condition on the software.

For the R-30iB/R-30iB Mate/R-30iA Controller or CE or RIA specification of the R-30iA Mate Controller

Mode	Teach pendant enable switch	Software remote condition	Teach pendant	Operator panel	Peripheral device
AUTO mode	On	Local	Not allowed	Not allowed	Not allowed
		Remote	Not allowed	Not allowed	Not allowed
	Off	Local	Not allowed	Allowed to start	Not allowed
		Remote	Not allowed	Not allowed	Allowed to start
T1, T2 mode	On	Local	Allowed to start	Not allowed	Not allowed
		Remote	Allowed to start	Not allowed	Not allowed
	Off	Local	Not allowed	Not allowed	Not allowed
		Remote	Not allowed	Not allowed	Not allowed

T1,T2 mode: DEADMAN switch is effective.

For the standard specification of R-30iA Mate Controller

Teach pendant enable switch	Software remote condition	Teach pendant	Peripheral device
On	Ignored	Allowed to start	Not allowed
Off	Local	Not allowed	Not allowed
	Remote	Not allowed	Allowed to start

- (6) (Only when R-30iB/R-30iB Mate /R-30iA Controller or CE or RIA specification of R-30iA Mate controller is selected.) To start the system using the operator panel, make certain that nobody is in the robot operating space and that there are no abnormal conditions in the robot operating space.
- (7) When a program is completed, be sure to carry out the test operation according to the following procedure.
 - (a) Run the program for at least one operation cycle in the single step mode at low speed.
 - (b) Run the program for at least one operation cycle in the continuous operation mode at low speed.
 - (c) Run the program for one operation cycle in the continuous operation mode at the intermediate speed and check that no abnormalities occur due to a delay in timing.
 - (d) Run the program for one operation cycle in the continuous operation mode at the normal operating speed, and check that the system operates automatically without trouble.
 - (e) After checking the completeness of the program through the test operation above, execute it in the automatic operation mode.
- (8) While operating the system in the automatic operation mode, the teach pendant operator must leave the safety fence.

3.3 SAFETY OF THE MAINTENANCE ENGINEER

For the safety of maintenance engineer personnel, pay utmost attention to the following.

- (1) During operation, never enter the robot operating space.
- (2) A hazardous situation may arise when the robot or the system, are kept with their power-on during maintenance operations. Therefore, for any maintenance operation, the robot and the system should be put into the power-off state. If necessary, a lock should be in place in order to prevent any other person from turning on the robot and/or the system. In case maintenance needs to be executed in the power-on state, the emergency stop button must be pressed.
- (3) If it becomes necessary to enter the robot operating space while the power is on, press the emergency stop button on the operator box or operator panel, or the teach pendant before entering the range. The maintenance worker must indicate that maintenance work is in progress and be careful not to allow other people to operate the robot carelessly.
- (4) When entering the area enclosed by the safety fence, the worker must check the whole robot system in order to make sure no dangerous situations exist. In case the worker needs to enter the safety area whilst a dangerous situation exists, extreme care must be taken, and whole robot system status must be carefully monitored.
- (5) Before the maintenance of the pneumatic system is started, the supply pressure should be shut off and the pressure in the piping should be reduced to zero.
- (6) Before the start of maintenance work, check that the robot and its peripheral equipment are all in the normal operating condition.
- (7) Do not operate the robot in the automatic operation while anybody is in the robot operating space.
- (8) When you maintain the robot alongside a wall or instrument, or when multiple users are working nearby, make certain that their escape path is not obstructed.
- (9) When a tool is mounted on the robot, or when any movable device other than the robot is installed, such as belt conveyor, pay careful attention to its motion.
- (10) If necessary, have a user who is familiar with the robot system stand beside the operator panel and observe the work being performed. If any danger arises, the user should be ready to press the EMERGENCY STOP button at any time.
- (11) When replacing a part, please contact your local FANUC representative. If a wrong procedure is followed, an accident may occur, causing damage to the robot and injury to the user.
- (12) When replacing or reinstalling components, take care to prevent foreign material from entering the system.
- (13) When handling each unit or printed circuit board in the controller during inspection, turn off the circuit breaker to protect against electric shock.

If there are two cabinets, turn off the both circuit breaker.

- (14) A part should be replaced with a part recommended by FANUC. If other parts are used, malfunction or damage would occur. Especially, a fuse that is not recommended by FANUC should not be used. Such a fuse may cause a fire.
- (15) When restarting the robot system after completing maintenance work, make sure in advance that there is no person in the operating space and that the robot and the peripheral equipment are not abnormal.
- (16) When a motor or brake is removed, the robot arm should be supported with a crane or other equipment beforehand so that the arm would not fall during the removal.
- (17) Whenever grease is spilled on the floor, it should be removed as quickly as possible to prevent dangerous falls.
- (18) The following parts are heated. If a maintenance user needs to touch such a part in the heated state, the user should wear heat-resistant gloves or use other protective tools.
 - Servo motor
 - Inside the controller
 - Reducer
 - Gearbox
 - Wrist unit
- (19) Maintenance should be done under suitable light. Care must be taken that the light would not cause any danger.
- (20) When a motor, reducer, or other heavy load is handled, a crane or other equipment should be used to protect maintenance workers from excessive load. Otherwise, the maintenance workers would be severely injured.
- (21) The robot should not be stepped on or climbed up during maintenance. If it is attempted, the robot would be adversely affected. In addition, a misstep can cause injury to the worker.
- (22) When performing maintenance work in high place, secure a footstep and wear safety belt.
- (23) After the maintenance is completed, spilled oil or water and metal chips should be removed from the floor around the robot and within the safety fence.
- (24) When a part is replaced, all bolts and other related components should put back into their original places. A careful check must be given to ensure that no components are missing or left not mounted.
- (25) In case robot motion is required during maintenance, the following precautions should be taken :
 - Foresee an escape route. And during the maintenance motion itself, monitor continuously the whole robot system so that your escape route will not become blocked by the robot, or by peripheral equipment.
 - Always pay attention to potentially dangerous situations, and be prepared to press the emergency stop button whenever necessary.
- (26) The robot should be periodically inspected. (Refer to the robot mechanical manual and controller maintenance manual.) A failure to do the periodical inspection can adversely affect the performance or service life of the robot and may cause an accident
- (27) After a part is replaced, a test execution should be given for the robot according to a predetermined method. (See TESTING section of “Controller operator’s manual”.) During the test execution, the maintenance worker should work outside the safety fence.

4 SAFETY OF THE TOOLS AND PERIPHERAL EQUIPMENT

4.1 PRECAUTIONS IN PROGRAMMING

- (1) Use a limit switch or other sensor to detect a dangerous condition and, if necessary, design the program to stop the robot when the sensor signal is received.

- (2) Design the program to stop the robot when an abnormality occurs in any other robots or peripheral equipment, even though the robot itself is normal.
- (3) For a system in which the robot and its peripheral equipment are in synchronous motion, particular care must be taken in programming so that they do not interfere with each other.
- (4) Provide a suitable interface between the robot and its peripheral equipment so that the robot can detect the states of all devices in the system and can be stopped according to the states.

4.2 PRECAUTIONS FOR MECHANISM

- (1) Keep the component cells of the robot system clean, operate the robot where insulated from the influence of oil, water, and dust.
- (2) Don't use unconfirmed liquid for cutting fluid and cleaning fluid.
- (3) Adopt limit switches or mechanical stoppers to limit the robot motion, and avoid the robot from collisions against peripheral equipment or tools.
- (4) Observe the following precautions about the mechanical unit cables. Failure to follow precautions may cause problems.
 - Use mechanical unit cable that have required user interface.
 - Do not add user cable or hose to inside of the mechanical unit.
 - Please do not obstruct the movement of the mechanical unit when cables are added to outside of mechanical unit.
 - In the case of the model that a cable is exposed, please do not perform remodeling (Adding a protective cover and fix an outside cable more) obstructing the behavior of the outcrop of the cable.
 - When installing user peripheral equipment on the robot mechanical unit, please pay attention that the device does not interfere with the robot itself.
- (5) The frequent power-off stop for the robot during operation causes the trouble of the robot. Please avoid the system construction that power-off stop would be operated routinely. (Refer to bad case example.) Please perform power-off stop after reducing the speed of the robot and stopping it by hold stop or cycle stop when it is not urgent. (Please refer to "STOP TYPE OF ROBOT" in "SAFETY PRECAUTIONS" for detail of stop type.)
(Bad case example)
 - Whenever poor product is generated, a line stops by emergency stop and power-off of the robot is incurred.
 - When alteration is necessary, safety switch is operated by opening safety fence and power-off stop is incurred for the robot during operation.
 - An operator pushes the emergency stop button frequently, and a line stops.
 - An area sensor or a mat switch connected to safety signal operates routinely and power-off stop is incurred for the robot.
 - Power-off stop is regularly incurred due to an inappropriate setting for Dual Check Safety (DCS).
- (6) Power-off stop of Robot is executed when collision detection alarm (SRVO-050) etc. occurs. Please try to avoid unnecessary power-off stops. It may cause the trouble of the robot, too. So remove the causes of the alarm.

5 SAFETY OF THE ROBOT MECHANICAL UNIT

5.1 PRECAUTIONS IN OPERATION

- (1) When operating the robot in the jog mode, set it at an appropriate speed so that the operator can

manage the robot in any eventuality.

- (2) Before pressing the jog key, be sure you know in advance what motion the robot will perform in the jog mode.

5.2 PRECAUTIONS IN PROGRAMMING

- (1) When the operating spaces of robots overlap, make certain that the motions of the robots do not interfere with each other.
- (2) Be sure to specify the predetermined work origin in a motion program for the robot and program the motion so that it starts from the origin and terminates at the origin. Make it possible for the operator to easily distinguish at a glance that the robot motion has terminated.

5.3 PRECAUTIONS FOR MECHANISMS

Keep the robot operation area clean, and operate the robot in an environment free of grease, water, and dust.

5.4 PROCEDURE TO MOVE ARM WITHOUT DRIVE POWER IN EMERGENCY OR ABNORMAL SITUATIONS

For emergency or abnormal situations (e.g. persons trapped in or pinched by the robot), brake release unit can be used to move the robot axes without drive power.

Please refer to controller maintenance manual and mechanical unit operator's manual for using method of brake release unit and method of supporting robot.

6 SAFETY OF THE END EFFECTOR

6.1 PRECAUTIONS IN PROGRAMMING

- (1) To control the pneumatic, hydraulic and electric actuators, carefully consider the necessary time delay after issuing each control command up to actual motion and ensure safe control.
- (2) Provide the end effector with a limit switch, and control the robot system by monitoring the state of the end effector.

7 STOP TYPE OF ROBOT

The following three robot stop types exist:

Power-Off stop (Category 0 following IEC 60204-1)

Servo power is turned off and the robot stops immediately. Servo power is turned off when the robot is moving, and the path of the deceleration is uncontrolled.

The following processing is performed at Power-Off stop.

- An alarm is generated and servo power is turned off. The robot operation is stopped immediately.
- Execution of the program is paused.

Frequent Power-Off stop of the robot during operation can cause failures of the robot.

Avoid system designs that require routine or frequent Power-Off stop conditions.

Controlled stop (Category 1 following IEC 60204-1)

The robot is decelerated until it stops, and servo power is turned off.

The following processing is performed at Controlled stop.

- The alarm "SRVO-199 Controlled stop" occurs along with a decelerated stop of the robot. Execution of the program is paused.
- After the decelerated stop, an alarm is generated and servo power is turned off.

Hold (Category 2 following IEC 60204-1)

The robot is decelerated until it stops, and servo power remains on.

The following processing is performed at Hold.

- The robot operation is decelerated until it stops. Execution of the program is paused.



WARNING

The stopping distance and stopping time of Controlled stop are longer than the stopping distance and stopping time of Power-Off stop. A risk assessment for the whole robot system, which takes into consideration the increased stopping distance and stopping time, is necessary when Controlled stop is used.

When the emergency stop button is pressed or the FENCE is open, the stop type of robot is Power-Off stop or Controlled stop. The configuration of stop type for each situation is called *stop pattern*. The stop patterns differ according to the controller type or option configuration.

There are the following 3 stop patterns.

Stop pattern	Mode	Emergency stop button	External Emergency stop	FENCE open	SVOFF input	Servo disconnect
A	AUTO	P-Stop	P-Stop	C-Stop	C-Stop	P-Stop
	T1	P-Stop	P-Stop	-	C-Stop	P-Stop
	T2	P-Stop	P-Stop	-	C-Stop	P-Stop
B	AUTO	P-Stop	P-Stop	P-Stop	P-Stop	P-Stop
	T1	P-Stop	P-Stop	-	P-Stop	P-Stop
	T2	P-Stop	P-Stop	-	P-Stop	P-Stop
C	AUTO	C-Stop	C-Stop	C-Stop	C-Stop	C-Stop
	T1	P-Stop	P-Stop	-	C-Stop	P-Stop
	T2	P-Stop	P-Stop	-	C-Stop	P-Stop

P-Stop: Power-Off stop

C-Stop: Controlled stop

-: Disable

The following table indicates the stop pattern according to the controller type or option configuration.

Option	R-30iB/R-30iB Mate
Standard	A (*)
Controlled stop by E-Stop (A05B-2600-J570)	C (*)

(*) R-30iB/R-30iB Mate does not have servo disconnect. R-30iB Mate does not have SVOFF input.

Option	R-30iA/ R-30iB				R-30iA Mate		
	Standard (Single)	Standard (Dual)	RIA type	CE type	Standard	RIA type	CE type
Standard	B(*)	A	A	A	A(**)	A	A
Stop type set (Stop pattern C) (A05B-2500-J570)	Cannot be selected	Cannot be selected	C	C	Cannot be selected	C	C

(*) R-30iA standard specification (single) does not have servo disconnect.

(**) R-30iA Mate standard specification does not have servo disconnect. The stop type for SVOFF input

is Power-Off stop.

The stop pattern of the controller is displayed in "Stop pattern" line in the software version screen. Refer to "Software version" in operator's manual of the controller for the detail of the software version screen.

"Controlled stop by E-Stop" option

When "Controlled stop by E-Stop" (A05B-2600-J570) option (For the R-30iA/R-30iA Mate, it is Stop type set (Stop pattern C) (A05B-2500-J570)) is specified, the stop type of the following alarms is Controlled stop in AUTO mode. In T1 or T2 mode, the stop type is Power-Off stop.

Alarm	Condition
SRVO-001 Operator panel E-stop	Operator's panel E-stop is pressed.
SRVO-002 Teach pendant E-stop	Teach pendant emergency stop is pressed.
SRVO-007 External emergency stops	External emergency stop input (EES1-EES11, EES2-EES21) is open. (R-30iA/R-30iB/R-30iB Mate controller)
SRVO-194 Servo disconnect	Servo disconnect input (SD4-SD41, SD5-SD51) is open. (R-30iA controller)
SRVO-218 Ext.E-stop/Servo Disconnect	External emergency stop input (EES1-EES11, EES2-EES21) is open. (R-30iA Mate controller)
SRVO-408 DCS SSO Ext Emergency Stop	In DCS Safe I/O connect function, SSO[3] is OFF.
SRVO-409 DCS SSO Servo Disconnect	In DCS Safe I/O connect function, SSO[4] is OFF.

Controlled stop is different from **Power-Off stop** as follows:

- In Controlled stop, the robot is stopped on the program path. This function is effective for a system where the robot can interfere with peripheral equipment or other devices if it deviates from the program path.
- Controlled stop has less physical impact than Power-Off stop. This function is effective for systems where the physical impact on tools is required to be reduced.
- The stopping distance and stopping time of Controlled stop is longer than the stopping distance and stopping time of Power-Off stop. Refer to the operator's manual of a particular robot model for the data of stopping distance and stopping time.

In case of R-30iA/R-30iA Mate, this option is available only in the CE or RIA type controller.

When this option is loaded, this function cannot be disabled.

The stop type of DCS Position and Speed Check functions is not affected by the loading of this option. The stop type set on the DCS screen is used.



WARNING

The stopping distance and stopping time of Controlled stop is longer than the stopping distance and stopping time of Power-Off stop. A risk assessment for the whole robot system, which takes into consideration the stopping distance and stopping time increased in AUTO mode on the above alarms, is necessary when this option is loaded.

160802

TABLE OF CONTENTS

SAFETY PRECAUTIONS	s-1
1 PREFACE.....	1
1.1 OVERVIEW OF THE MANUAL	1
1.2 RELATED MANUALS	1
2 OVERVIEW	3
2.1 CONFIGURATION OF SOFTWARE OPTIONS	4
2.2 CONFIGURATIONS	5
2.2.1 Queue Managed Tracking	5
2.2.2 Visual Tracking	7
2.3 KEY CONCEPTS.....	9
2.4 SAMPLE SYSTEM CONFIGURATIONS	14
2.5 STUDY FOR APPLICATION	21
2.6 BASIC START-UP PROCEDURE	27
2.7 RESTRICTIONS	27
3 PREPARATION.....	28
3.1 NETWORK CREATION	28
3.1.1 Connecting a Network Cable.....	28
3.1.2 Caution on Setting up a Network	29
3.1.3 Setting IP Addresses.....	30
3.1.4 Setting up the Robot Ring	31
3.2 PULSECODER INSTALLATION, CONNECTION AND SETUP	37
3.2.1 Installing a Pulsecoder.....	37
3.2.2 Connecting the Pulsecoder	38
3.2.2.1 Hardware to which to connect a Pulsecoder	38
3.2.2.2 When one robot is used	42
3.2.2.3 When multiple robots are used	44
3.2.3 Setting up the Pulsecoder	45
3.3 CAMERA INSTALLATION AND CONNECTION	48
3.4 SETTING OF THE TCP	49
3.5 GRID PATTERN	50
4 SETUP OF THE BASIC FUNCTIONS.....	51
4.1 SETUP OF A WORKCELL	60
4.2 SETUP OF A ROBOT	63
4.3 SETUP OF A TRAY	64
4.4 SETUP OF A CONVEYOR	70
4.4.1 Setting a Tracking Frame	73
4.4.1.1 Line Conveyor	74
4.4.1.2 Circular Conveyor	78
4.5 SETUP OF A SENSOR	83
4.5.1 Setting a Sensor in Detail	83
4.5.2 Settings Required for Each Combination of a Trigger Condition and a Trigger Action	91
4.5.3 Setting the Sensor Position.....	92
4.5.3.1 When a tray is not used.....	92

	4.5.3.2	When a tray is used.....	94
4.5.4		Setting Up the Vision System	96
	4.5.4.1	Setting up the camera.....	96
	4.5.4.2	Calibrating the camera.....	97
	4.5.4.3	Vision process.....	105
	4.5.4.4	Teaching a vision process.....	105
	4.5.4.5	Teaching a GPM locator tool.....	107
4.5.5		Setting the Tray Position	108
	4.5.5.1	When [Distance] and [Find part by vision] are selected.....	108
	4.5.5.2	When [DI] and [Find part by vision] are selected	110
4.5.6		Setting the Target Position	112
	4.5.6.1	When a tray is not used.....	112
	4.5.6.2	When a tray is used.....	114
4.6		SETUP OF A CONVEYOR STATION	116
4.6.1		Adding a Tracking Frame.....	120
	4.6.1.1	Line Conveyor	120
	4.6.1.2	Circular Conveyor	122
4.7		SETUP OF A FIXED STATION	125
4.8		KAREL PROGRAMS	126
4.8.1		Workcell-Related Programs	126
	4.8.1.1	PKWCSTART.PC	126
	4.8.1.2	PKWCEND.PC.....	127
	4.8.1.3	PKWCCHECK.PC	127
	4.8.1.4	PKWCRSTPRDST.PC	127
	4.8.1.5	PKGETVAR.PC	128
	4.8.1.6	PKSETVAR.PC.....	129
	4.8.1.7	PKPRGRGETEN.PC.....	130
4.8.2		Conveyor-Related Programs	130
	4.8.2.1	PKCVSETLBD.PC.....	130
	4.8.2.2	PKCVSETTRNAM.PC	131
	4.8.2.3	PKCVENBLB.PC.....	132
	4.8.2.4	PKCVENBSC.PC.....	133
4.8.3		Sensor-Related Programs	134
	4.8.3.1	PKSNENBSENS.PC	134
	4.8.3.2	PKSNSETVPNAM.PC.....	135
	4.8.3.3	PKSNSETMVNAM.PC	136
	4.8.3.4	PKSNSETVTOFS.PC	137
	4.8.3.5	PKSNSETMTOFS.PC.....	138
	4.8.3.6	PKSNSTART.PC	139
	4.8.3.7	PKSNSTOP.PC	139
4.8.4		Conveyor Station-Related Programs	140
	4.8.4.1	PKCSGETID.PC	140
	4.8.4.2	PKCSCLRQUE.PC	141
	4.8.4.3	PKCSGETQUE.PC	141
	4.8.4.4	PKCSGETQCELL.PC.....	142
	4.8.4.5	PKCSACKQUE.PC.....	143
	4.8.4.6	PKCSPUTQUE.PC.....	144
	4.8.4.7	PKCSGETTIME.PC	145
	4.8.4.8	PKCSGETDIST.PC.....	146
	4.8.4.9	PKCSGETPFRT.PC	148
	4.8.4.10	PKCSGETNPRT.PC	149
	4.8.4.11	PKCSSETDSLIN.PC.....	149
	4.8.4.12	PKCSGETENC.PC.....	150
	4.8.4.13	PKCSGETTRID.PC	150
4.8.5		Fixed Station-Related Programs.....	151
	4.8.5.1	PKFSGETID.PC	151
	4.8.5.2	PKFSPUTQUE.PC	151

4.8.5.3	PKFSCLRQUE.PC	151
4.8.5.4	PKFSGETQUE.PC	152
4.8.5.5	PKFSACKQUE.PC	153
4.8.5.6	PKFSPUTBUF.PC	153
4.8.5.7	PKFSCLRBUF.PC	154
4.8.5.8	PKFSGETBUF.PC	154
4.8.5.9	PKFSACKBUF.PC	155
4.8.6	Tray-Related Programs	156
4.8.6.1	PKTRSETCLPOS.PC	156
4.8.6.2	PKTRGETNUMCL.PC	157
4.9	ROBOT PROGRAMS	158
4.9.1	Tracking Program	158
4.9.2	Sample Programs for the Conveyor-to-Conveyor Configuration	159
4.9.2.1	Main program	160
4.9.2.2	Pick program	161
4.9.2.3	Placement program	164
4.9.3	Sample Programs for the Conveyor-to-Fixed Station Configuration	164
4.9.3.1	Main program	165
4.9.3.2	Pick program	166
4.9.3.3	Placement program	166
4.9.4	Sample Programs for the Conveyor-to-Buffer-to-Conveyor Configuration	168
4.9.4.1	Main program	168
4.9.4.2	Pick program	170
4.9.4.3	Placement program	171
4.9.4.4	Pick program (Buffer)	172
4.9.4.5	Placement program	172
4.10	REFERENCE POSITION SETTING	173
4.10.1	When [Distance] and [Find part by vision] Are Selected	174
4.10.2	When [DI] and [Find part by vision] Are Selected	175
4.10.3	When [DI] or [HDI] and [Put part in queue] Are Selected	176
4.10.4	When [Distance] or [Flag] and [Put part in queue] Are Selected	177
4.11	TEACHING THE POSITION TO ROBOTS	177
4.11.1	Conveyor Station	177
4.11.2	Fixed Station	178
4.12	FINE ADJUSTMENT OF THE TRACKING MOTION	179
4.12.1	Initial Fine Adjustment	179
4.12.2	Adjustment of the Tracking Frame Error	179
4.12.3	Adjustment of the Dynamic Error of the Robot	181
4.12.4	Continuous Termination Type Specification for a Taught Position	183
4.12.5	Linear Distance Specification	184
4.12.6	Linear Face Plate Instruction	184
4.12.7	Maximum Linear Speed Function	185
4.12.8	Time Before Instruction	186
4.13	SYSTEM STARTUP	186
4.14	SETUP OF PRE-GROUPING	187
4.14.1	Key Concepts	189
4.14.2	Setup of a Workcell and a Robot	189
4.14.3	Setup of a Tray	189
4.14.4	Setup of a Conveyor	191
4.14.5	Setup of a Sensor	198
4.14.6	Setup of a Conveyor Station	199
4.14.7	Setup of a Fixed Station	206
4.14.8	KAREL Programs	206
4.14.8.1	PKCSGENVRTRY.PC	206
4.14.8.2	PKCSGETQPRGR.PC	206

4.14.8.3	PKCSTRGPRGR.PC	207
4.14.8.4	PKCSACKQPRGR.PC.....	208
4.14.9	Robot Programs.....	209
4.14.9.1	Sample programs for the conveyor-to-the same conveyor configuration (the first robot).....	209
4.14.9.2	Sample programs for the conveyor-to-the same conveyor configuration (downstream robots).....	212
4.14.9.3	Sample programs for the conveyor-to-conveyor configuration.....	213
4.14.10	Reference Position Setting	215
4.14.11	Teaching the Position to Robots.....	215
4.14.12	Fine Adjustment of The Tracking Motion.....	215
4.14.12.1	Adjustment of the tracking frame error	215

5 VARIATIONS.....217

5.1	USING DIFFERENT ROBOTS TO PICK PARTS DEPENDING ON THEIR POSITIONS ON THE CONVEYOR	217
5.2	USING DIFFERENT ROBOTS TO PICK PARTS DEPENDING ON THEIR CHARACTERISTICS	220
5.3	PLACING PARTS IN LAYERS	224
5.4	PLACING PARTS IN SEQUENCE FROM THE END OF THE BOX.....	226
5.5	PLACING PARTS IN DIFFERENT DIRECTIONS IN A BOX	227
5.6	PLACING PARTS IN DIFFERENT PLACES OF A BOX DEPENDING ON THE MODEL ID.....	228
5.7	PLACING PARTS ON DIFFERENT OUTFEED CONVEYORS DEPENDING ON THE MODEL ID.....	231
5.8	PICKING PARTS FROM TWO OR MORE INFEED CONVEYORS	233
5.9	REPORTING FAILURE TO PICK A PART	236
5.10	PICKING A TRAY ONLY WHEN ALL CELLS ARE FILLED	237
5.11	MULTI-VIEW	239
5.11.1	Part Detection Procedure.....	239
5.11.2	Limitations.....	239
5.11.3	Preparation for Camera.....	239
5.11.4	Setup.....	240
5.11.4.1	Vision process.....	240
5.11.4.2	Robot program.....	241
5.11.4.3	Settings of sensor task	241
5.11.4.4	Reference position setting and robot position teaching	242
5.11.5	Run Multi-View System.....	243
5.11.6	Production Status.....	244
5.12	CIRCULAR TRACKING	244
5.12.1	Limitations.....	244
5.12.2	Setup.....	245
5.12.2.1	When the vision system is used	246

6 SETUP OF PLUG & PLAY FUNCTIONS.....248

6.1	SETUP OF A WORKCELL	257
6.2	SETUP OF A GRIPPER AND A ZONE	260
6.2.1	Gripper and Zone.....	260
6.2.1.1	Setting up the gripper in detail.....	264
6.2.1.2	Setting up the zone in detail.....	265
6.3	SETUP OF A ROBOT.....	267
6.4	SETUP OF A ROBOT OPERATION.....	268

6.4.1	Conveyor Station Robot Operation	268
6.4.1.1	Setting up conveyor station robot operation in detail	270
6.4.2	Fixed Station Robot Operation.....	272
6.4.2.1	Setting up fixed station robot operation in detail.....	273
6.5	SETUP OF A TRAY	276
6.6	SETUP OF A CONVEYOR.....	282
6.6.1	Setting a Tracking Frame	285
6.6.1.1	Line conveyor.....	286
6.6.1.2	Circular conveyor	290
6.7	SETUP OF A SENSOR	295
6.7.1	Setting a Sensor in Detail	295
6.7.2	Settings Required for Each Combination of a Trigger Condition and a Trigger Action	303
6.7.3	Setting the Sensor Position.....	303
6.7.3.1	When a tray is not used.....	303
6.7.3.2	When a tray is used.....	305
6.7.4	Setting Up the Vision System	307
6.7.4.1	Setting up the camera.....	307
6.7.4.2	Calibrating the camera.....	308
6.7.4.3	Vision process.....	315
6.7.4.4	Teaching a vision process.....	316
6.7.4.5	Teaching a GPM locator tool.....	317
6.7.5	Setting the Tray Position	318
6.7.5.1	When [Distance] and [Find part by vision] are selected.....	318
6.7.5.2	When [DI] and [Find part by vision] are selected	320
6.7.6	Setting the Target Position	322
6.7.6.1	When a tray is not used.....	323
6.7.6.2	When a tray is used.....	324
6.8	SETUP OF A CONVEYOR STATION	326
6.8.1	Adding a Tracking Frame.....	330
6.8.1.1	Line conveyor.....	330
6.8.1.2	Case of circular conveyor	333
6.9	SETUP OF A FIXED STATION	336
6.10	ROBOT PROGRAMS	337
6.10.1	Robot Behavior Under the Standard TP Programs.....	337
6.10.2	Numeric Registers	339
6.10.3	Position Registers	339
6.10.4	String Registers	340
6.10.5	Vision Registers	340
6.10.6	Payload IDs	341
6.10.7	User Alarms.....	341
6.10.8	User Frames	341
6.10.9	Tool Frames.....	341
6.11	REFERENCE POSITION SETTING	341
6.11.1	When [Distance] and [Find part by vision] are Selected.....	342
6.11.2	When [DI] and [Find part by vision] are Selected.....	343
6.11.3	When [DI] or [HDI] and [Put part in queue] are Selected.....	344
6.11.4	When [Distance] or [Flag] and [Put part in queue] are Selected	345
6.12	TEACHING THE POSITION TO ROBOTS	345
6.12.1	Conveyor Station.....	345
6.12.2	Fixed Station	346
6.13	FINE ADJUSTMENT OF THE TRACKING MOTION.....	347
6.13.1	Initial Fine Adjustment.....	347
6.13.2	Adjustment of the Tracking Frame Error	347

6.13.3	Adjustment of the Dynamic Error of the Robot	349
6.14	KAREL PROGRAMS	350
6.14.1	Workcell-Related Programs	350
6.14.1.1	PKWCSTART.PC	350
6.14.1.2	PKWCEND.PC	351
6.14.1.3	PKWCCHECK.PC	351
6.14.1.4	PKWCRSTPRDST.PC	351
6.14.1.5	PKRSTWC.PC	351
6.14.1.6	PKABO.PC	352
6.14.1.7	PKLABO.PC	352
6.14.1.8	PKPOSTERR.PC	352
6.14.1.9	PKOPGETEN.PC	353
6.14.1.10	PKGETVAR.PC	353
6.14.1.11	PKSETVAR.PC	354
6.14.1.12	PKPRGRGETEN.PC	355
6.14.2	Gripper-Related Programs	355
6.14.2.1	PKGRCCHKPP.PC	355
6.14.2.2	PKGRCCHKQTY.PC	356
6.14.2.3	PKGRCLOSE.PC	357
6.14.2.4	PKGRCGETDPDLY.PC	357
6.14.2.5	PKGRCGETID.PC	357
6.14.2.6	PKGRCGETPKDLY.PC	358
6.14.2.7	PKGRCGETZNCNT.PC	358
6.14.2.8	PKGRCGETZNUT.PC	358
6.14.2.9	PKGRCINIT.PC	359
6.14.2.10	PKGRCOPEN.PC	359
6.14.3	Robot-Related Programs	359
6.14.3.1	PKRBGETCVCNT.PC	359
6.14.3.2	PKRBGETFSCNT.PC	360
6.14.3.3	PKRBGETID.PC	360
6.14.3.4	PKRBGETSFLG.PC	361
6.14.3.5	PKRBGETPRGRT.PC	361
6.14.4	Conveyor-Related Programs	362
6.14.4.1	PKCVSETLBD.PC	362
6.14.4.2	PKCVSETTRNAM.PC	363
6.14.4.3	PKCVENBLB.PC	364
6.14.4.4	PKCVENBSC.PC	365
6.14.4.5	PKCVGETNAME.PC	366
6.14.5	Sensor-Related Programs	367
6.14.5.1	PKSNENBSSENS.PC	367
6.14.5.2	PKSNSETVPNAM.PC	368
6.14.5.3	PKSNSETMVNAM.PC	369
6.14.5.4	PKSNSETVTOFS.PC	370
6.14.5.5	PKSNSETMTOFS.PC	371
6.14.5.6	PKSNSTART.PC	372
6.14.5.7	PKSNSTOP.PC	372
6.14.6	Conveyor Station-Related Programs	373
6.14.6.1	PKCSGETID.PC	373
6.14.6.2	PKCSCLRQUE.PC	373
6.14.6.3	PKCSGETQUE.PC	374
6.14.6.4	PKCSGETQCELL.PC	375
6.14.6.5	PKCSACKQUE.PC	376
6.14.6.6	PKCSPUTQUE.PC	377
6.14.6.7	PKCSGETTIME.PC	377
6.14.6.8	PKCSGETDIST.PC	379
6.14.6.9	PKCSGETPFRT.PC	380
6.14.6.10	PKCSGETNPRT.PC	381

6.14.6.11	PKCSSETDSLIN.PC.....	382
6.14.6.12	PKCSCALWPOS.PC	383
6.14.6.13	PKCSCLRACK.PC	384
6.14.6.14	PKCSGETAPPR.PC.....	384
6.14.6.15	PKCSGETOPPR.PC.....	385
6.14.6.16	PKCSGETPMUL.PC.....	385
6.14.6.17	PKCSGETPPCHK.PC.....	385
6.14.6.18	PKCSGETQTY.PC	386
6.14.6.19	PKCSGETVR.PC	386
6.14.6.20	PKCSGETWPOS.PC	386
6.14.6.21	PKCSGETENC.PC.....	387
6.14.6.22	PKCSGETTRID.PC	387
6.14.7	Fixed Station-Related Programs.....	387
6.14.7.1	PKFSGETID.PC.....	387
6.14.7.2	PKFSPUTQUE.PC	388
6.14.7.3	PKFSCLRQUE.PC.....	388
6.14.7.4	PKFSGETQUE.PC.....	388
6.14.7.5	PKFSACKQUE.PC	389
6.14.7.6	PKFSPUTBUF.PC	390
6.14.7.7	PKFSCLRBUF.PC	390
6.14.7.8	PKFSGETBUF.PC	391
6.14.7.9	PKFSACKBUF.PC.....	392
6.14.7.10	PKFSGETAPPR.PC	393
6.14.7.11	PKFSGETIXDLY.PC.....	393
6.14.7.12	PKFSGETIXODO.PC	393
6.14.7.13	PKFSGETIXSIM.PC.....	394
6.14.7.14	PKFSGETOPPR.PC	394
6.14.7.15	PKFSGETPMUL.PC	394
6.14.7.16	PKFSGETPPCHK.PC	395
6.14.7.17	PKFSGETQTY.PC.....	395
6.14.7.18	PKFSGETRDY.PC.....	395
6.14.7.19	PKFSGETRDYDI.PC	396
6.14.7.20	PKFSGETTRID.PC.....	396
6.14.7.21	PKFSGETUF.PC	396
6.14.7.22	PKFSGETVR.PC	397
6.14.7.23	PKFSGETMONIDX.PC.....	397
6.14.7.24	PKFSSETRDY.PC	397
6.14.8	Tray-Related Programs.....	398
6.14.8.1	PKTRSETCLPOS.PC.....	398
6.14.8.2	PKTRGETNUMCL.PC	400
6.15	STANDARD TP PROGRAMS.....	400
6.15.1	TP Program Naming Conventions.....	400
6.15.2	TP Program Details	402
6.15.2.1	List of all the TP Programs	402
6.15.2.2	PK_MAIN1.TP: Main Program for Group 1.....	402
6.15.2.3	PK_INIT1.TP	403
6.15.2.4	PK_PERCH1.TP.....	405
6.15.2.5	PK_PICK1.TP	406
6.15.2.6	PK_DROP1.TP.....	407
6.15.2.7	PK_GR_GET_UTOOLS1.TP	407
6.15.2.8	PK_CV_GET_PICK_DATA11.TP	408
6.15.2.9	PK_CV_GET_DROP_DATA11.TP	409
6.15.2.10	PK_FS_GET_PICK_DATA11.TP	411
6.15.2.11	PK_FS_GET_DROP_DATA11.TP.....	412
6.15.2.12	PK_CV_PICK11.TP.....	413
6.15.2.13	PK_CV_DROP11.TP	417
6.15.2.14	PK_CV_WAITPOS1.TP	421
6.15.2.15	PK_FS_PICK11.TP	422

6.15.2.16	PK_FS_DROP11.TP	424
6.15.2.17	PK_FS_INDEX.TP.....	427
6.15.2.18	PK_FS_READY.TP	428
6.15.2.19	PK_GR_GET_ENDZONE1.TP	429
6.15.2.20	PK_CYCSTOP1.TP	429
6.16	SETUP OF PRE-GROUPING	430
6.16.1	Key Concepts	432
6.16.2	Setup of a Workcell.....	433
6.16.3	Setup of a Gripper and a Zone.....	433
6.16.4	Setup of a Robot.....	433
6.16.5	Setup of a Robot Operation	433
6.16.6	Setup of a Tray	434
6.16.7	Setup of a Conveyor.....	435
6.16.8	Setup of a Sensor.....	443
6.16.9	Setup of a Conveyor Station.....	444
6.16.10	Setup of a Fixed Station	450
6.16.11	Robot Programs	450
6.16.12	Reference Position Setting	450
6.16.13	Teaching the Position to Robots.....	450
6.16.14	Fine Adjustment of The Tracking Motion.....	450
6.16.14.1	Adjustment of the tracking frame error	450
6.16.15	KAREL Programs	451
6.16.15.1	PKCSGENVRTRY.PC.....	451
6.16.15.2	PKCSGETQPRGR.PC	452
6.16.15.3	PKCSTRGPRGR.PC	452
6.16.15.4	PKCSACKQPRGR.PC.....	453
6.16.16	Standard TP programs	454
6.16.16.1	PK_PRGR_MAIN1.TP	454
6.16.16.2	PK_PRGR_INIT1.TP	456
6.16.16.3	PK_PRGR_CV_PICKDROP1.TP.....	456
6.16.16.4	PK_PRGR_CV_PICKVTRAY1.TP.....	459
7	OTHER FUNCTIONS	463
7.1	EXCHANGE WORKCELL WITH USING RECIPE.....	463
7.1.1	Setting a Recipe.....	463
7.1.2	Detail Setting of a Recipe.....	465
7.1.3	PLC Recipe Changeovers.....	476
7.1.3.1	Setting of PLC recipe change data.....	477
7.1.4	KAREL Program.....	479
7.1.4.1	PKRCSWITCH.PC.....	479
7.2	CHECKING THE PRODUCTION STATUS.....	480
7.2.1	Checking the Status of a Conveyor.....	481
7.2.2	Checking the Status of a Sensor	483
7.2.3	Checking the Status of a Conveyor Station.....	484
7.2.4	Checking the Status of a Fixed Station.....	487
7.3	LIMITING THE MOVABLE RANGE OF THE FINAL AXIS.....	488
7.4	SKIPPING THE TRACKING MOTION OUTSIDE THE AREA	488
7.5	CHECKING THE ROBOT TORQUE.....	491
7.6	iRPickTool 4D GRAPHICS DISPLAY	491
7.6.1	Displaying the 4D Workcell.....	493
7.6.2	Creating the 4D Workcell.....	495
7.6.3	Using the Touch Screen Icons.....	505
7.6.4	Using the 4D [TYPE] Scenes	506
7.6.4.1	4D pick boundaries.....	506
7.6.4.2	4D pick frames.....	507

7.6.5	Deleting the 4D Workcell.....	507
7.6.6	Changing the Workcell Scene Using Recipes	507
7.6.7	Loading Custom CAD Files	508
7.6.8	Limitations.....	508
7.7	BUFFER FUNCTION.....	508
7.7.1	The Buffer Algorithm.....	509
7.7.2	Setting up a Buffer	510
7.7.3	Buffer KAREL Macros	511
7.7.3.1	PKRBGETBUFID.PC	512
7.7.3.2	PKFSGETBUFPT.PC.....	512
7.7.3.3	PKFSGETBUFDI.PC.....	512
7.7.4	Buffer TP Programs.....	513
7.7.4.1	PK_BF_GET_DATA1.TP.....	513
7.7.4.2	PK_BF_PICK1.TP	513
7.7.4.3	PK_BF_DROP1.TP.....	515
7.7.4.4	PK_BF_OK2PICK1.TP.....	517
7.7.4.5	PK_BF_OK2DROP1.TP	518
7.7.4.6	PK_CV_OK2PICK1.TP.....	518
7.7.4.7	PK_CV_OK2DROP1.TP	519
7.7.4.8	PK_BUF_PICK1.TP.....	521
7.7.4.9	PK_BUF_DROP1.TP.....	521
7.8	EFFICIENCY TOOLS	522
7.8.1	Pick Only Reachable Parts	522
7.8.2	Setting up Pick Only Reachable Parts.....	523
7.8.3	KAREL Macros.....	524
7.8.3.1	PKCSGETPOSCK.PC.....	524
7.8.3.2	PKCSGETQINRG.PC.....	524
7.8.3.3	PKCSCHKPOS.PC.....	526
7.8.3.4	PKOPPOSCHK.PC	527
7.8.4	Sample Program	528
8	TROUBLESHOOTING	532
8.1	WHEN THE ROBOT DOES NOT PICK UP PARTS	532
8.2	WHEN PARTS ARE MISSED FREQUENTLY.....	532
8.3	WHEN THE SAME PART IS PICKED UP TWICE.....	534
8.4	COMPENSATION DEVIATES WHEN A PART ROTATES	534
8.5	COMPENSATION DEVIATES FOR A PART NEAR THE CONVEYOR EDGES	534
8.6	WHEN ROBOT MOTION IS NOT SMOOTH	534
8.7	THE STOP CONVEYOR DOES NOT WORK.....	535
8.8	IMAGE SNAP TIMING IS DELAYED	535
8.9	THE OVERRUN COUNT CONTINUES TO INCREASE.....	535
8.10	WHEN SETTINGS CANNOT BE CHANGED DURING PRODUCTION	536
 APPENDIX		
A	SYSTEM VARIABLE FILES.....	539
B	SETTING A TEACHING PC.....	540
B.1	OPERATING SYSTEM AND BROWSER	540
B.2	COMMUNICATION CABLE	540
B.3	CONNECTION OF THE COMMUNICATION CABLE.....	540

B.4	SETTING AN IP ADDRESS FOR THE PC	541
B.5	CHANGING SETTINGS OF INTERNET EXPLORER	543
B.6	CHANGING SETTINGS OF WINDOWS FIREWALL.....	548
B.7	OPENING <i>iRPickTool</i> SCREENS.....	550
B.8	OPENING <i>iRVision</i> SCREENS.....	551
B.9	TROUBLESHOOTING.....	555
C	HIGH SPEED DI (HDI)	559
C.1	HARDWARE CONNECTION OF R-30 <i>i</i> B.....	559
C.2	HARDWARE CONNECTION OF R-30 <i>i</i> B Mate.....	561
C.3	INPUT SIGNAL.....	562
C.4	SETUP OF HDI.....	563
C.4.1	Overview	563
C.4.2	Enabling High Speed Scanning.....	563
D	SERVO CONVEYOR LINE TRACKING FUNCTION	565
D.1	OVERVIEW	565
D.2	SETUP.....	565
D.2.1	Independent Extended Axis Setup	565
D.2.2	Servo Conveyor Setup.....	572
D.2.3	How to Create TP Program for Servo Conveyor.....	577
D.2.4	Tracking Schedule Setup.....	585
D.2.5	Example of Main Program	585
D.3	FUNCTION FOR SERVO CONVEYOR LINE TRACKING	585
D.3.1	Dynamic Error Tune Variable	585
D.3.2	Wait Indexer Stop Function	585
D.3.3	KAREL Program for Servo Conveyor Line Tracking.....	586
D.4	LIMITATIONS ON SERVO CONVEYOR LINE TRACKING	595
E	αA1000S PULSECODER.....	596
E.1	REQUIREMENTS.....	596
E.1.1	R-30 <i>i</i> B Controller.....	596
E.1.2	R-30 <i>i</i> B Mate Controller.....	598
E.2	FIGURES.....	599
E.3	HOW TO CONNECT TO R-30 <i>i</i> B	601
E.4	HOW TO CONNECT TO R-30 <i>i</i> B Mate	603
F	INCREMENTAL PULSECODER.....	605
F.1	REQUIREMENTS.....	605
F.1.1	R-30 <i>i</i> B Controller.....	605
F.1.2	R-30 <i>i</i> B Mate Controller.....	607
F.2	FIGURES.....	608
F.3	HOW TO CONNECT	609
F.3.1	Connecting to Robot Controller Directly	609
F.3.2	Connecting to Pulse Multiplexer	611
G	SEPARATE DETECTOR UNIT (SDU)	613
H	KAREL PROGRAM LIST	619
I	SENSOR CUSTOMIZATION.....	620

I.1	OVERVIEW	620
I.2	CUSTOMIZATION STEPS	620
I.3	ESSENTIAL POINTS OF CUSTOMIZATION	625
I.4	ROUTINES	628
I.4.1	put_part_cst	628
I.4.2	run_vis_cst	628
I.4.3	get_ecnt_idx	629
I.4.4	update_nfnd	629
I.4.5	start_srv	629
I.4.6	incre_wid	630
I.4.7	put_queue	630
I.5	BUILT-INS	631
I.5.1	LINE_COUNT	631
I.5.2	V_RUN_FIND	631
I.5.3	VT_GET_FOUND	632
I.5.4	P_CSGETID	632
I.5.5	P_CVGETID	633

1 PREFACE

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

1.1 OVERVIEW OF THE MANUAL

Overview

This manual describes how to operate *iRPickTool* on the R-30*i*B/R-30*i*B Mate controller. This manual describes each function which is provided by *iRPickTool*. When you would like to start up the robot system which uses *iRVision*, please refer to manuals which are introduced in “1.2 RELATED MANUALS”.



CAUTION

This manual is based on the R-30*i*B/R-30*i*B Mate system software version 7DC3/11(V8.30P/11). 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.

Contents of this manual

Chapter 1	Preface Especially, see “CAUTION”.
Chapter 2	Overview
Chapter 3	Preparation
Chapter 4	Setup of the Basic functions
Chapter 5	Variation
Chapter 6	Setup of the Plug & Play functions
Chapter 7	Other functions
Chapter 8	Troubleshooting
Appendix A to I	System variable files, Setting a teaching PC, High speed scanning, Servo conveyor line tracking function, α A1000S Pulsecoder, Incremental Pulsecoder, Separate detector unit (SDU), KAREL program list, Sensor customization

1.2 RELATED MANUALS

This section introduces related manual.

R-30*i*B/R-30*i*B Mate CONTROLLER OPERATOR’S MANUAL (Basic Operation) B-83284EN

This is the main manual of R-30*i*B/R-30*i*B Mate 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.

R-30iB CONTROLLER MAINTENANCE MANUAL B-83195EN

This manual describes the maintenance and connection of R-30iB Controller.

R-30iB Mate CONTROLLER MAINTENANCE MANUAL B-83525EN

This manual describes the maintenance and connection of R-30iB Mate Controller.

R-30iB/R-30iB Mate CONTROLLER OPERATOR'S MANUAL (Alarm Code List) B-83284EN-1

This manual describes the error code listings, causes, and remedies of R-30iB/R-30iB Mate Controller.

R-30iB/R-30iB Mate CONTROLLER Optional Function OPERATOR'S MANUAL B83284EN-2

This manual describes the software optional functions of R-30iB/R-30iB Mate Controller.

R-30iB/R-30iB Mate CONTROLLER Sensor Mechanical Unit / Control Unit OPERATOR'S MANUAL B-83434EN

This manual describes the connection between sensors which is a camera or 3D Laser Sensor and R-30iB/R-30iB Mate Controller, and maintenance of sensors.

R-30iB/R-30iB Mate CONTROLLER iRVision OPERATOR'S MANUAL (Reference) B-83304EN

This manual is the reference manual for iRVision on the R-30iB/R-30iB Mate Controller. This manual describes each function which is provided by iRVision. This manual describes the meanings (e.g. the items on iRVision setup screen, the arguments of the instruction, and so on.

R-30iB/R-30iB Mate CONTROLLER iRVision 2D Vision Application OPERATOR'S MANUAL B-83304EN-1

This manual is desired to first refer to when you start up systems of iRVision 2D Compensation and 2.5D Compensation. This manual describes startup procedures of iRVision 2D Compensation and 2.5D Compensation system, creating programs, caution, technical know-how, response to several cases, and so on.

R-30iB/R-30iB Mate CONTROLLER iRVision Inspection Application OPERATOR'S MANUAL B-83304EN-3

This manual is desired to first refer to when you start up systems of inspection which uses iRVision. This manual describes startup procedures of inspection system which uses iRVision, creating programs, caution, technical know-how, response to several cases, and so on.

R-30iA/R-30iA Mate/R-30iB/R-30iB Mate CONTROLLER Ethernet Function OPERATOR'S MANUAL B-82974EN

This manual describes the robot networking options such as FTP, RIPE, iRConnect, PC Share, and so on.

R-30iB/R-30iB Mate CONTROLLER Line Tracking OPERATOR'S MANUAL B-83474EN

This manual describes the line tracking function which the robot system with iRPickTool is based on.

2 OVERVIEW

iRPickTool is robot software specialized for applications which detect parts flowing on a conveyor using a camera or sensor such as a photo eye and make robots to pick up the parts as they move along the conveyor belt. This software provides functions for making multiple robots share the work of picking up parts flowing on one conveyor. It also supports multiple conveyors, enabling a "conveyor-to-conveyor" application in which each robot picks up parts from one conveyor and places them on the other conveyor, and a "conveyor-to-fixed station" or "fixed station-to-conveyor" application in which not a conveyor, but a fixed station is used for infeed or outfeed. The robots can pick and place single or multiple parts from both these types of stations. The parts may be picked from or placed to trays. The robots can also pick from and place to buffers. *iRPickTool* also provides 4D menus on the teach pendant to visualize the setup of the conveyors, sensors and fixed stations.

iRPickTool supports not only the constant speed conveyors, but also the indexing conveyors. The indexing conveyors are special conveyors that continuously stop and go.

iRPickTool is supported in both the R-30iB and R-30iB Mate platforms.

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

iRPickTool Basic

iRPickTool

iRPickTool Basic + *iRVision* options (Visual tracking)

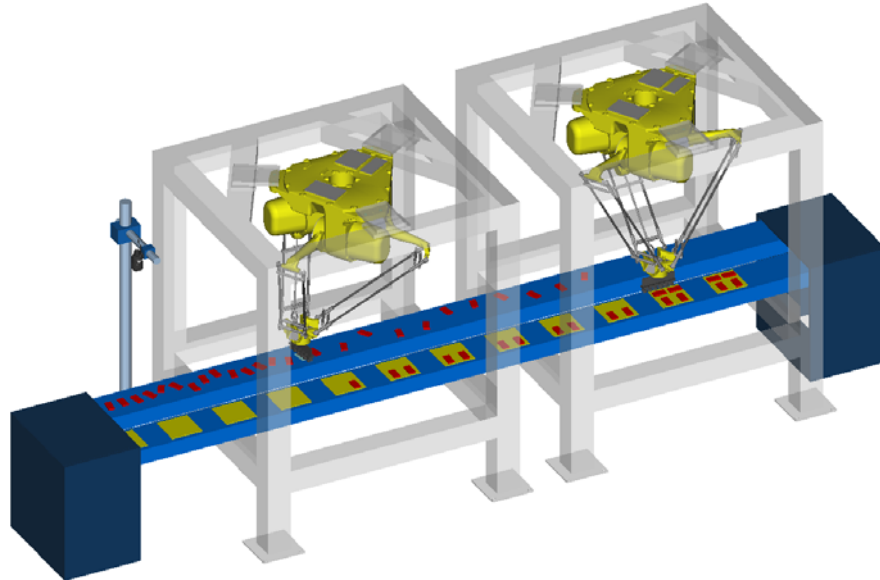
iRPickTool + *iRVision* options (Visual tracking)

"4D Graphics" option and "*iRPickTool* / 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 "*iRPickTool* basic" and "*iRPickTool*" as standard. The difference between "*iRPickTool* basic" and "*iRPickTool*" is described in Section 2.1, "CONFIGURATIONS OF SOFTWARE OPTIONS" later. The main difference is that *iRPickTool* 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.

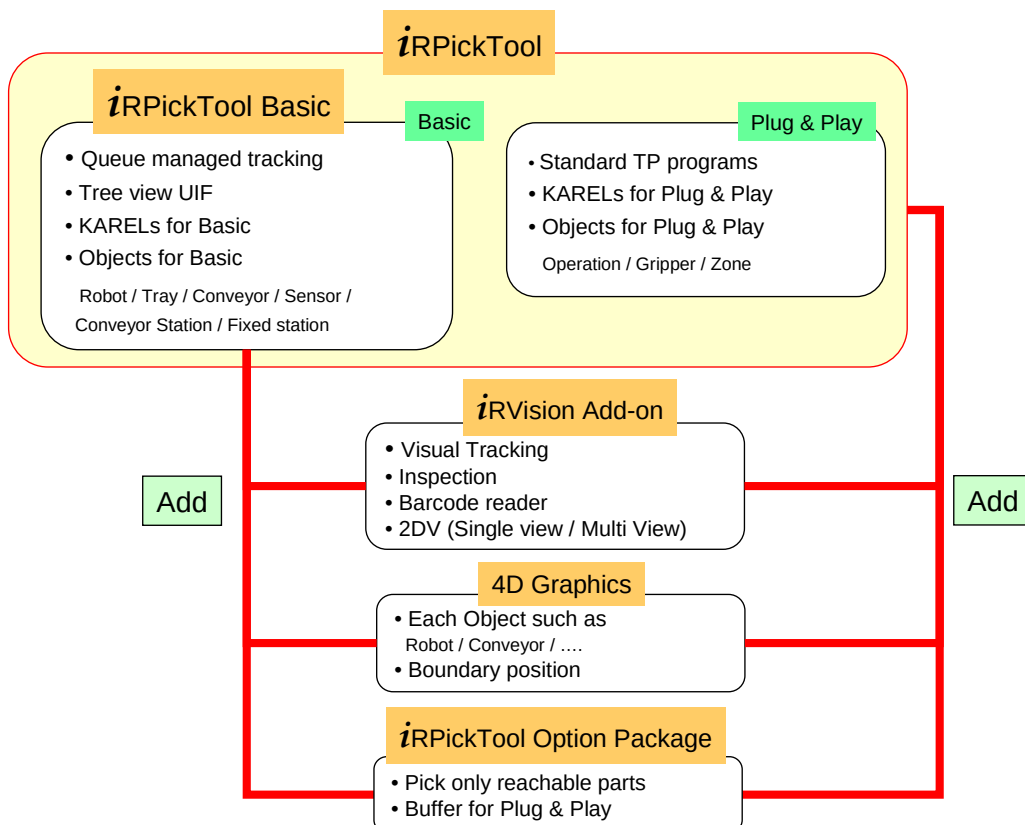
This manual describes the procedure to build a queue managed tracking system (or a visual tracking system). A queue managed tracking system of "*iRPickTool* Basic" can be built by performing the operations described in Chapter 2 through Chapter 4 according to the specified procedure. Chapter 5 introduces many variations and sample applications that are not contained in Chapters 2 to 4. In the other hand, a queue managed tracking system of "*iRPickTool*" can be built by performing the operations described in Chapter 6 instead of Chapter 4.

A visual tracking system can also be built in the same way. Replace the term "queue managed tracking" with "visual tracking".



2.1 CONFIGURATION OF SOFTWARE OPTIONS

iRPickTool has the following software configuration.



iRPickTool basic contains basic functions for building a queue managed tracking system. You can build a visual tracking system using a camera by adding iRVision options to iRPickTool basic.

iRPickTool contains standard TP programs and others in addition to the basic functions contained in iRPickTool basic to make setup easier. The functions for making setup easier are called Plug & Play functions. The difference between iRPickTool and iRPickTool basic is whether the Plug & Play functions are contained.

2.2 CONFIGURATIONS

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

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

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

Sample configurations

When the Ethernet Encoder is used

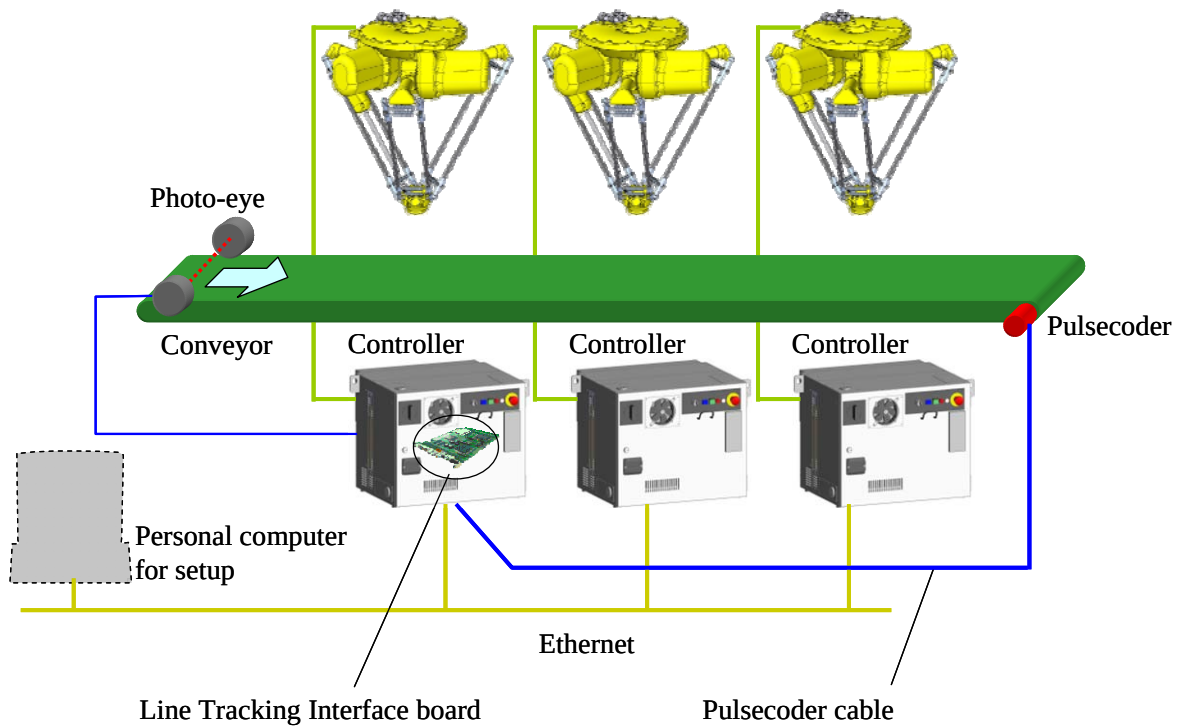


Fig. 2.2.1 (a)

When a pulse multiplexer is used

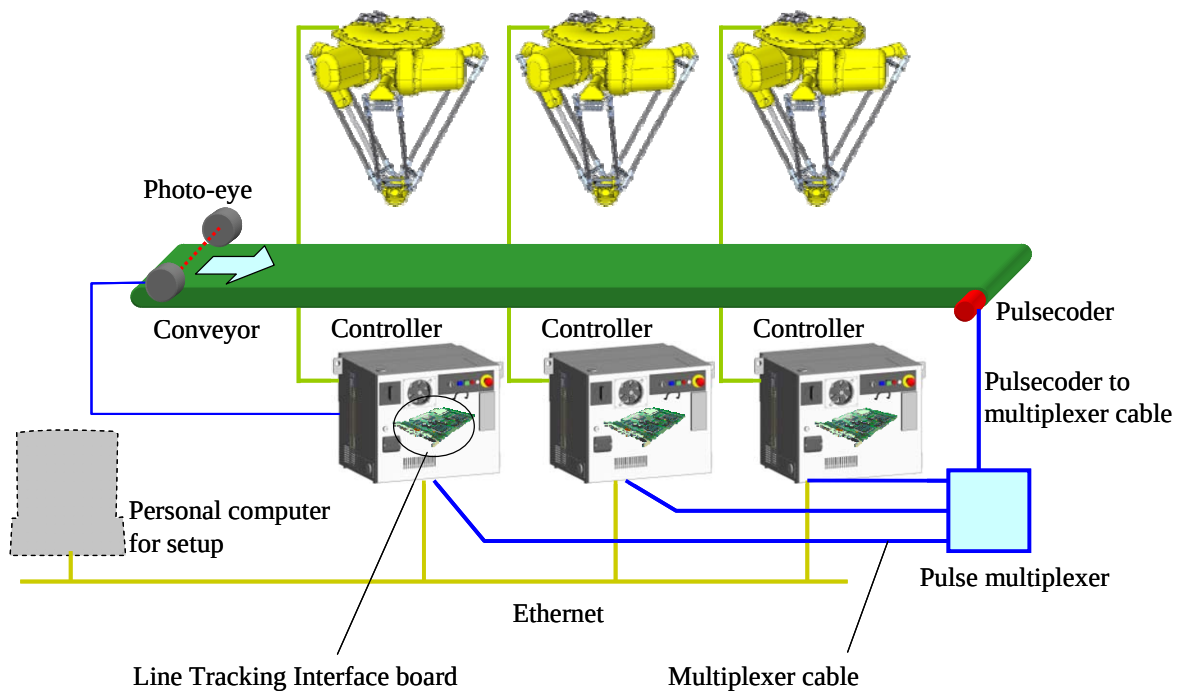


Fig. 2.2.1 (b)

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

- *i*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

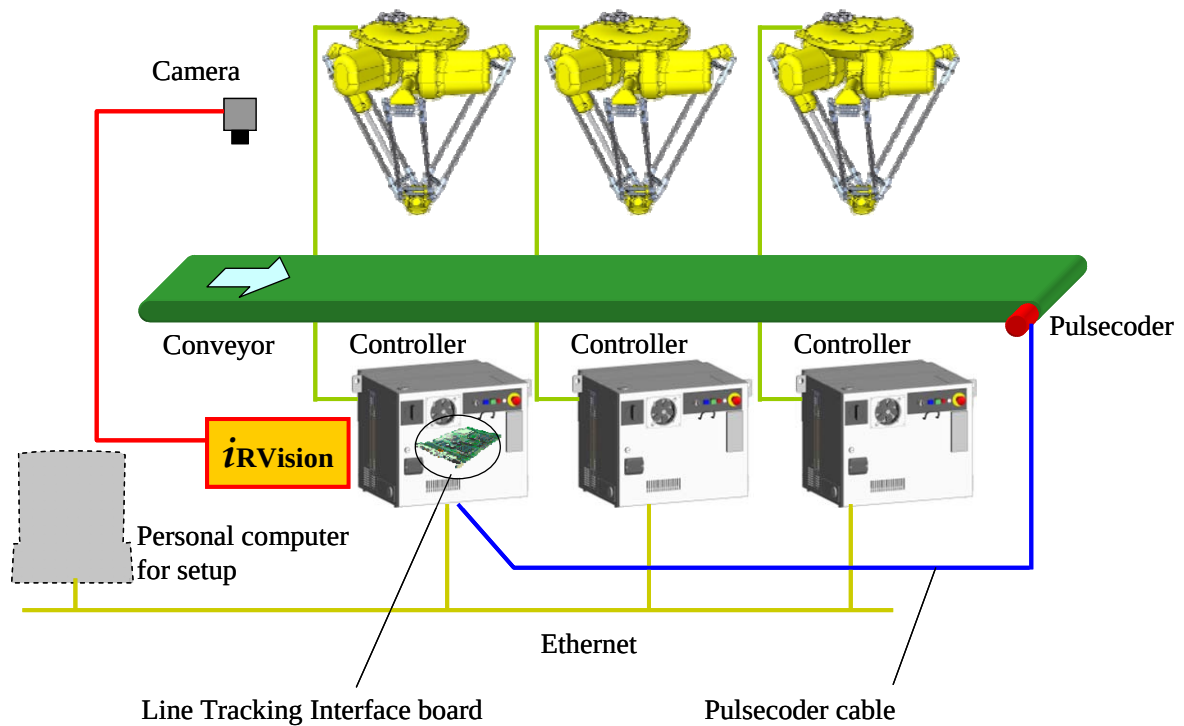


Fig. 2.2.2 (a)

When a pulse multiplexer is used

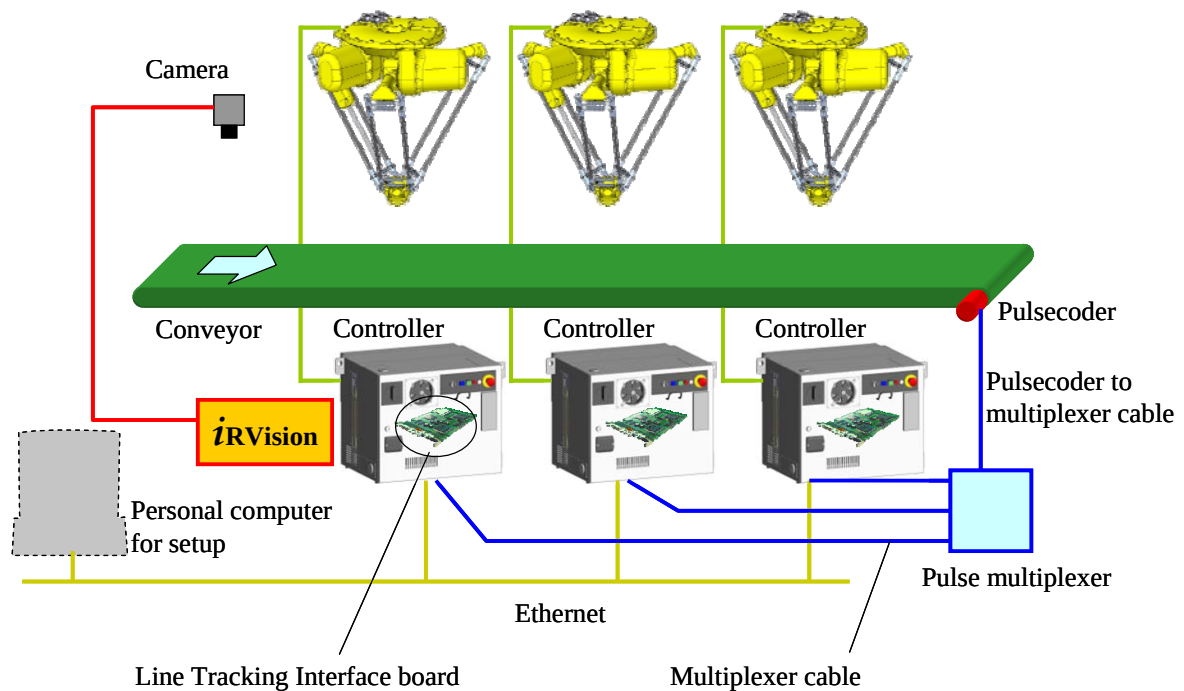


Fig. 2.2.2 (b)

2.3 KEY CONCEPTS

This section describes terms used in this manual.

Throughout the manual the term “plug and play” refers to the J945: *iRPickTool* option.

The following concepts are used for building a tracking system using *iRPickTool*:

Objects using in the tree view

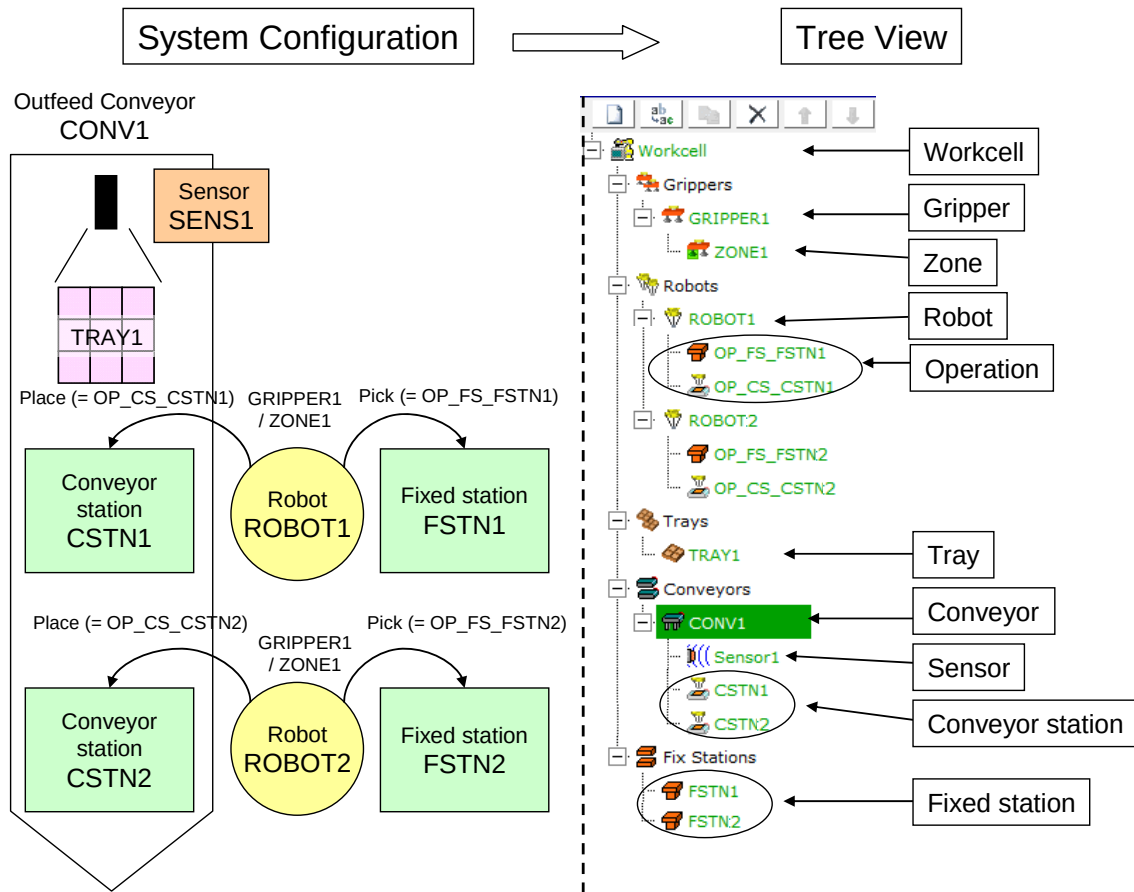


Fig. 2.3 (a)

With *iRPickTool*, 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

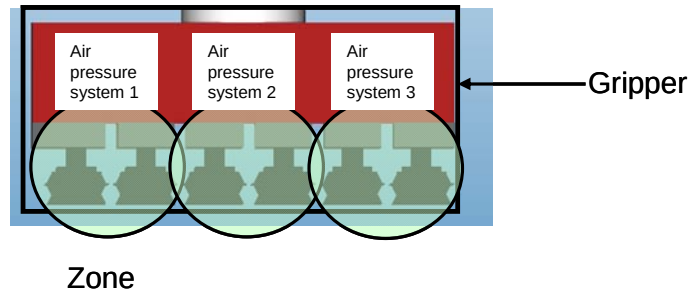
A work cell is an object indicating the entire tracking system configuration. Only one work cell can be set for each robot controller. If you want to change the system configuration, for example, for changing the setup, replace the entire work cell using a recipe described later.

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.



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 performed by a robot at a conveyor or fixed station. There are two types of operations: pick and place. 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 an object indicating this "box".

Cell

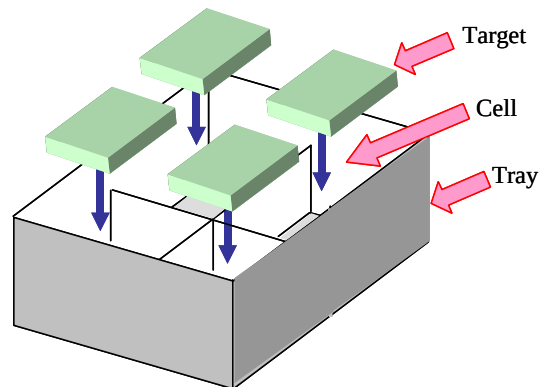
Each place where a part is placed (or picked up) is called a cell. For example, a box in which four parts can be laid out is defined as a tray pattern having four cells.

Tray frame

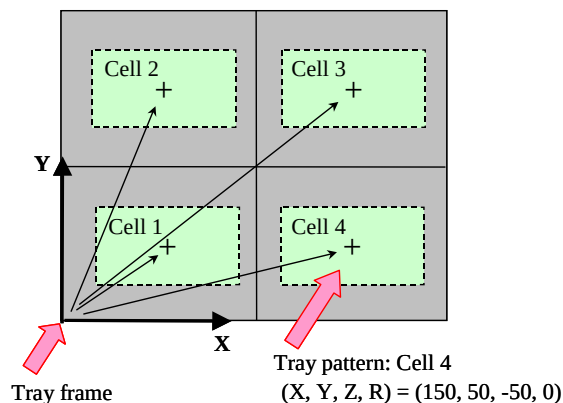
A tray frame is a frame used to define a cell layout in a tray. The position and orientation of each cell are set as coordinates in the tray frame. A tray frame can be defined in any position in the tray.

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.



In the example shown in the above figure, four parts can be placed in (or picked up from) a 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. In this figure, the above tray is viewed from the top. In this example, a tray frame is defined at the lower-left corner of the box.



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 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.

NOTE: There are different procedures to teach a tray frame on a conveyor versus a tray frame on a fixed station. The procedures are described later.

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

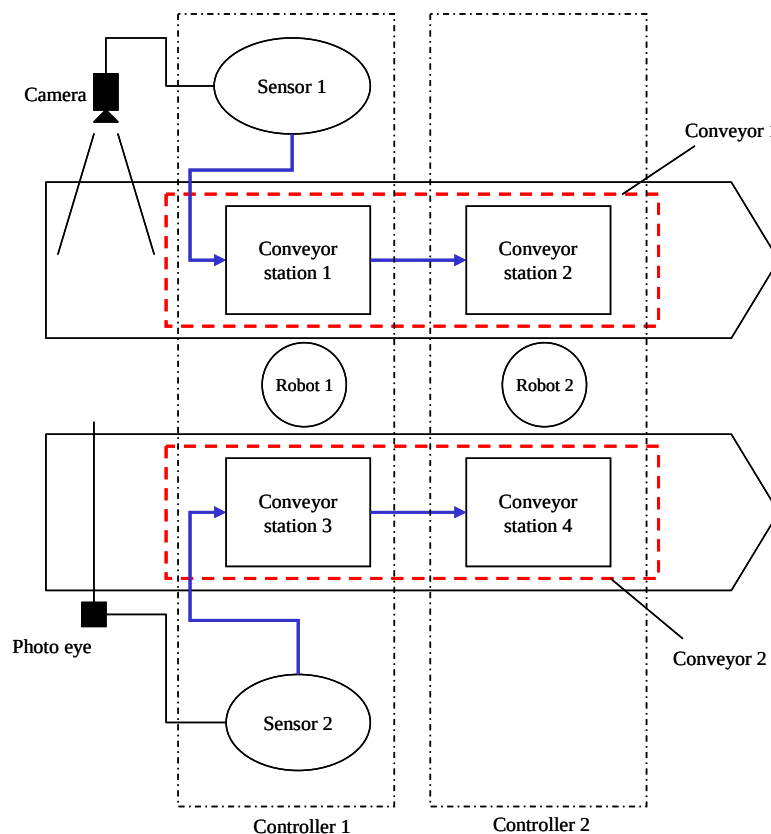
In an actual system, a KAREL program is automatically executed to perform processing according to the "sensor" setting. You can customize this KAREL program if needed, but the source code of the program is not provided as standard. If you need to customize the program, contact your local FANUC representative.

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 from 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.

The following figure 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 (b)**

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.

This system has two sensors, sensors 1 and 2.

Sensor 1 detects a part on conveyor 1 using a camera and inputs the information of the detected part to conveyor station 1. Sensor 2 detects a part on conveyor 2 using a photo eye and inputs the information of the detected part to conveyor station 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.

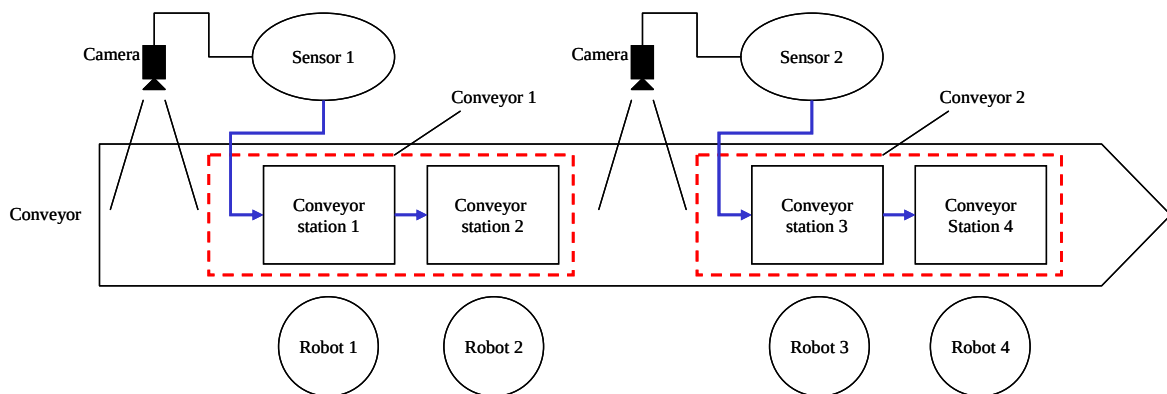


Fig. 2.3 (c)

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 (a) 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.

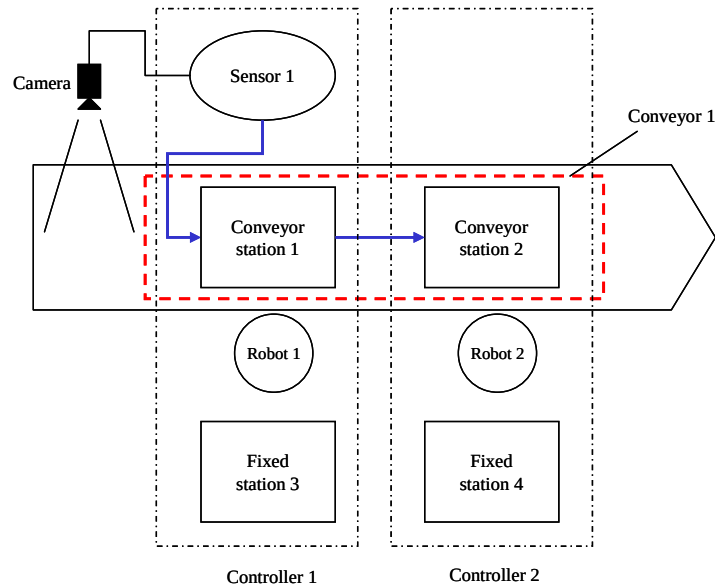


Fig. 2.3 (d)

2.4 SAMPLE SYSTEM CONFIGURATIONS

This section introduces some sample system configurations to help you understand *iRPickTool* 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”.

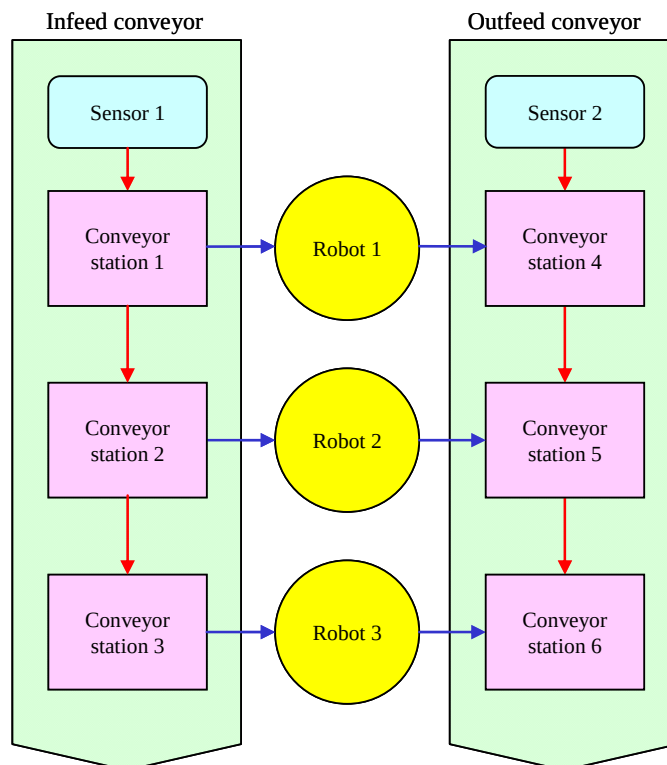


Fig. 2.4 (a)

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.

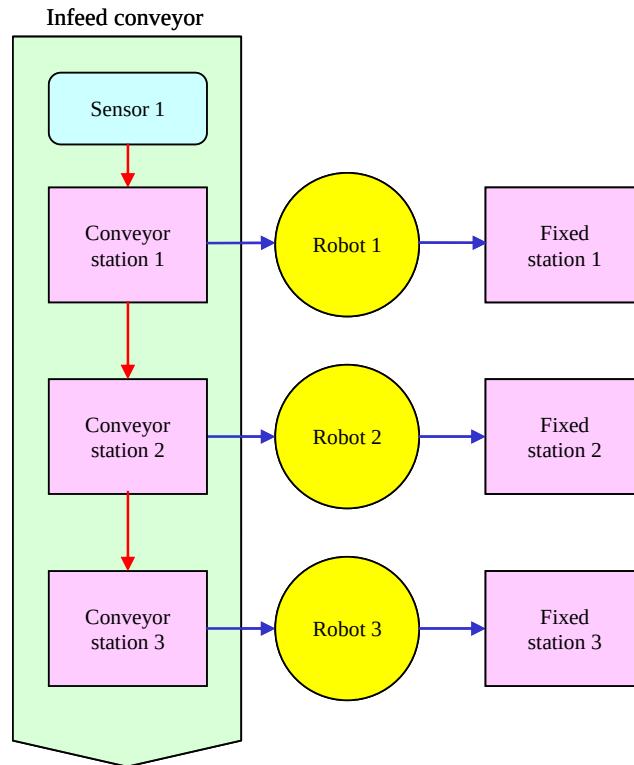


Fig. 2.4 (b)

Sample configuration 3

In this sample, fixed stations are added as buffers to sample configuration 1. Parts may not constantly flow on the infeed conveyor. When too many parts flow on the infeed conveyor, the robots place some of them on the fixed station. When too few parts flow, the robots pick up parts from the fixed station and place them on the conveyor station on the outfeed conveyor.

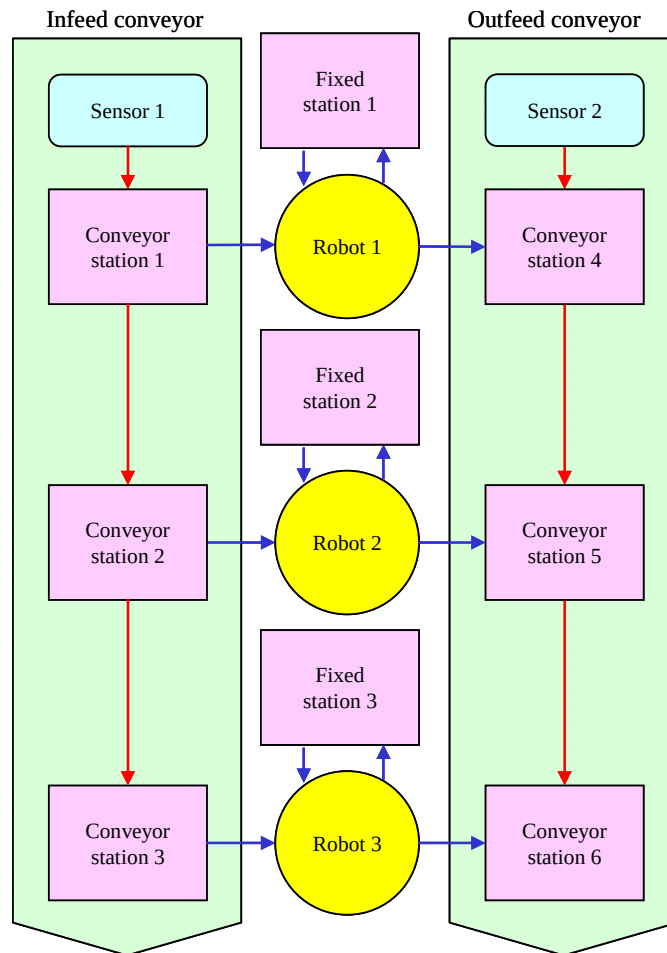


Fig. 2.4 (c)

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”.

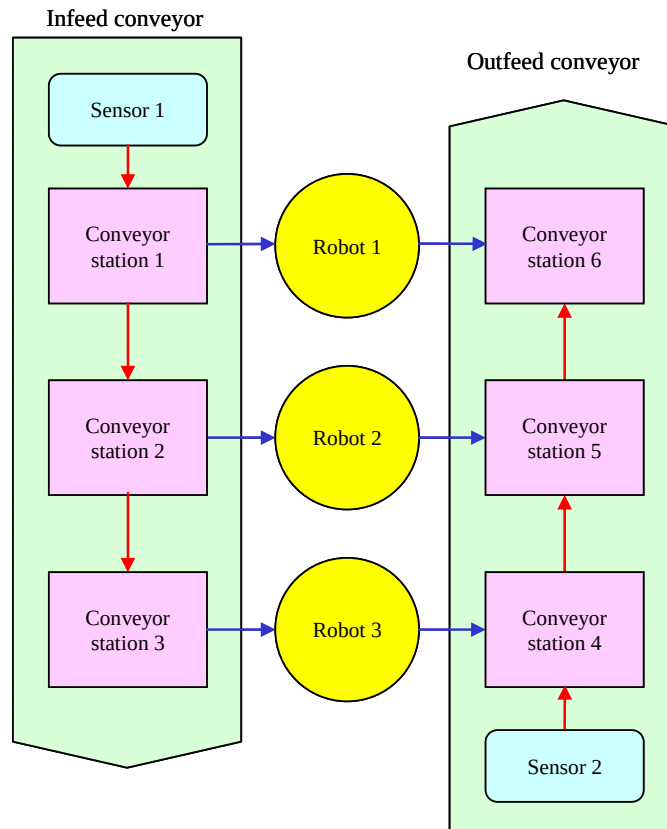
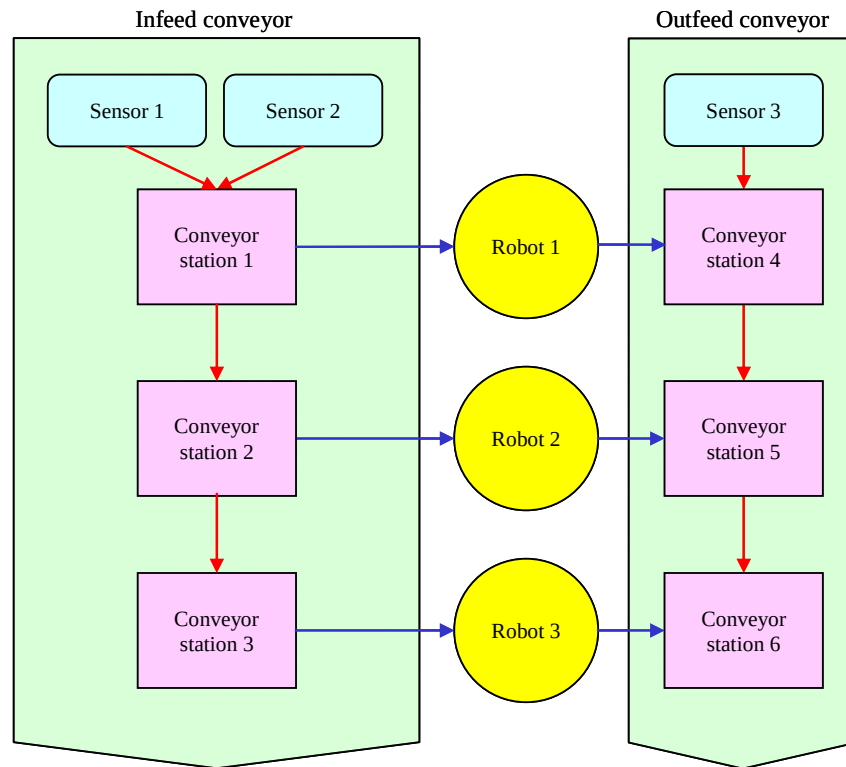


Fig. 2.4 (d)

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.

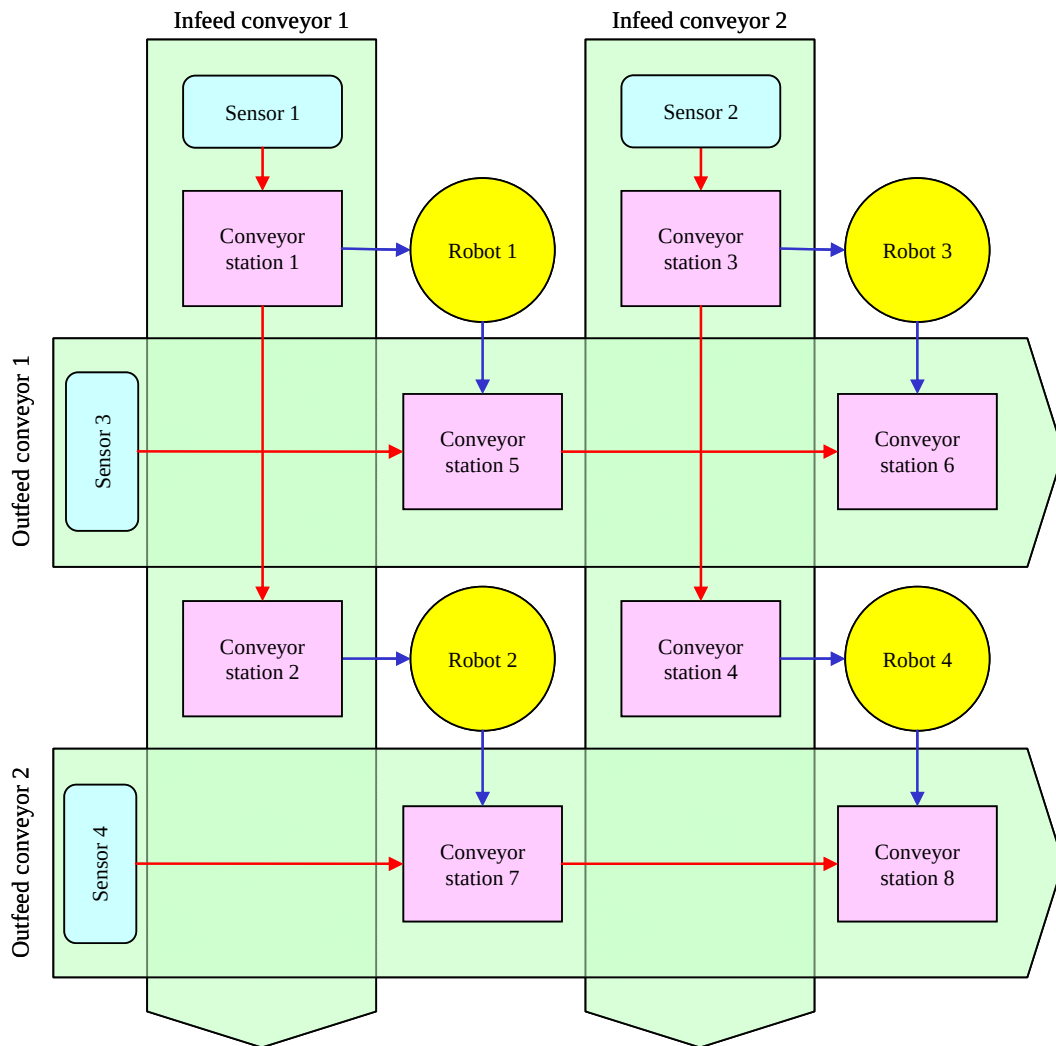


Fig. 2.4 (f)

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.

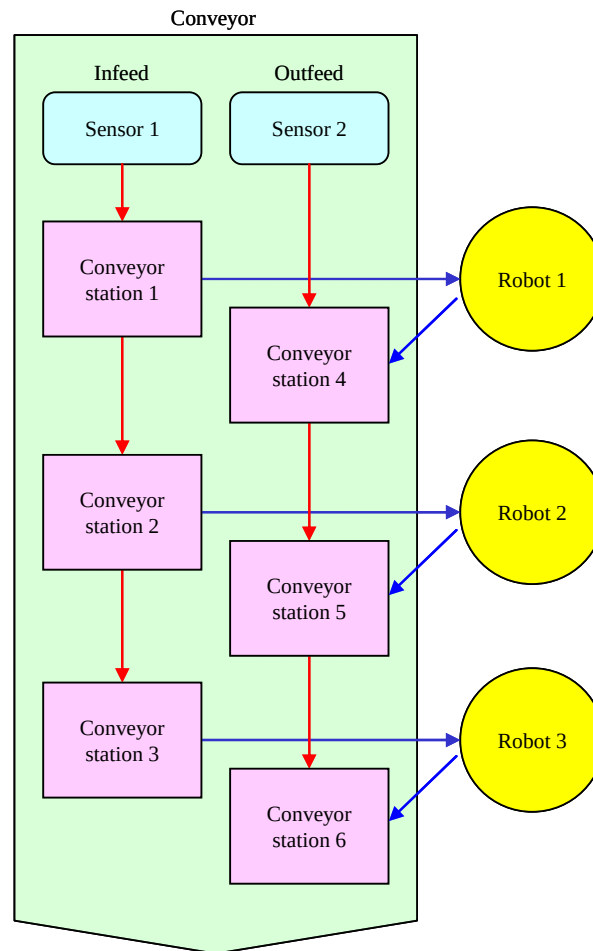


Fig. 2.4 (g)

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 5, "VARIATIONS".

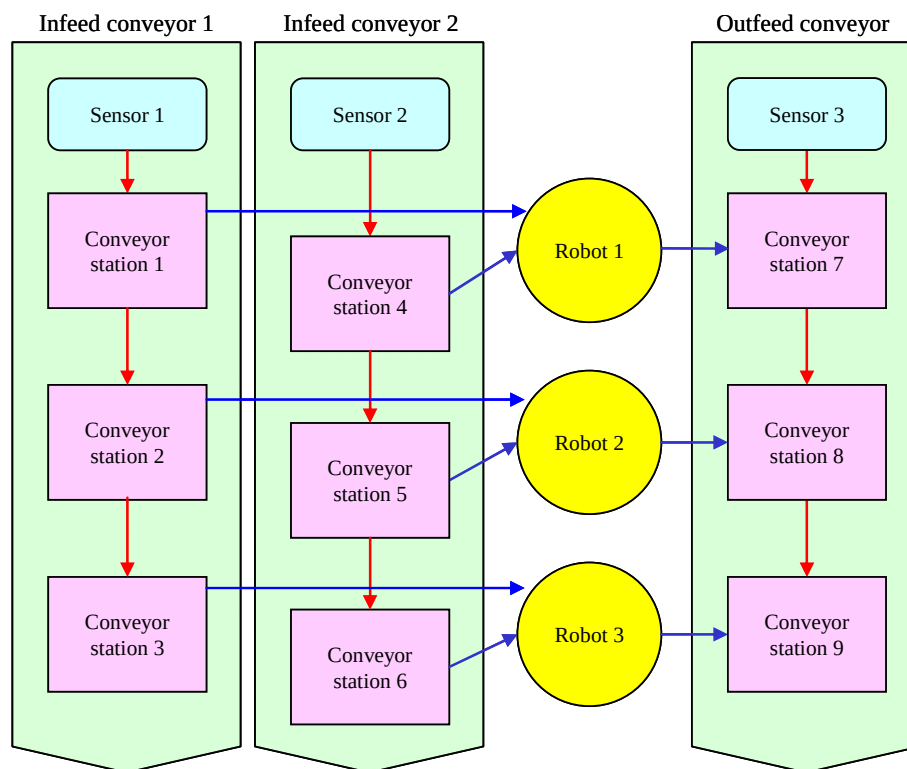


Fig. 2.4 (h)

2.5 STUDY FOR APPLICATION

This section provides information to design your “picking” application such as the layout, the robot and camera selection, use of Pulsecoders or Ethernet encoders 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 iRVision. 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 controllers. Also, the cable for connecting each 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 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.

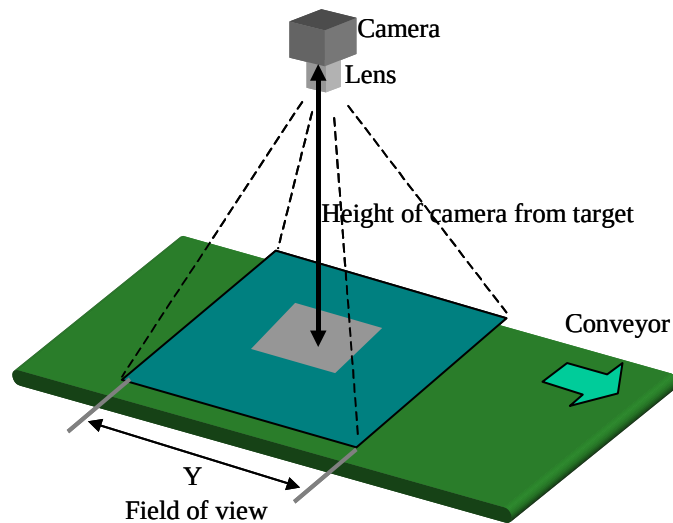


Fig. 2.5 (a)

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

$$\text{CCD size} = \text{cell size} \times \text{image size}$$

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

You can set and change the image size of a digital camera. For details, refer to the section, "DIGITAL CAMERA" in the chapter, "CAMERA" in the "iRVision OPERATOR'S MANUAL (Reference)".

Camera	Cell size	Image size	CCD size
Digital camera (grayscale) SC130E B/W	5.300 μm	1280 x 1024	6.784 mm x 5.427 mm
Digital camera (color) SC130E COLOR	5.300 μm	640 x 480	3.392 mm x 2.544 mm
Analog camera XC-56	7.400 μm	640 x 480	4.736 mm x 3.552 mm

* GigE camera is not supported yet.

For example, when the focal length is 8 mm, the height of the camera from the part is about 1000 mm, the cell size is 5.300 μm , and the image size is 1280 x 1024, the field of view is determined as follows:

Horizontal direction: $6.784 \text{ (mm)} \times ((1000 \text{ (mm)} - 8 \text{ (mm)}) \div 8 \text{ (mm)}) = 841.2 \text{ (mm)}$

Vertical direction: $5.427 \text{ (mm)} \times ((1000 \text{ (mm)} - 8 \text{ (mm)}) \div 8 \text{ (mm)}) = 672.9 \text{ (mm)}$

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:

$$0.1 \text{ (mm)} \div 100 \text{ (mm/s)} = 1 \text{ (ms)}$$

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:

$$\text{Trigger interval (s)} = 0.5 \times \text{camera's field of view (mm)} \div \text{conveyor speed (mm/s)}$$

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 *iRVision*. Model IDs are typically used to determine types of parts. Using the *iRVision* 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 5, "VARIATIONS".

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 BASIC START-UP PROCEDURE

Use the following procedure to start the *iRPickTool* tracking system:

- 0 Setting of a personal computer for setup
Set up a personal computer used to set up *iRPickTool*. This step is unnecessary when *iPendant* with a touch panel is used.
- 1 Setting of the robot ring
To perform tracking with multiple robots, set up the robot ring. This step is unnecessary when one robot is used.
- 2 Pulsecoder installation and setting
Install a Pulsecoder, connect it to a robot controller, and set the parameters for the Pulsecoder.
- 3 Camera installation and connection
Install and connect a camera. This step is necessary only for visual tracking. It is unnecessary for queue managed tracking.
- 4 Setting of the TCP for touch-up
Set the TCP for touch-up used for a purpose such as frame setting. Do this work on each robot.
- 5 Setting of a work cell
Set items related to the system configuration, including conveyors, conveyor stations, and sensors. Also, set the tracking frame, and the tray pattern when using a tray.
- 6 Teaching of a vision process
Perform camera setting, camera calibration, and vision process teaching for visual tracking. This step is unnecessary for photo-eye sensors.
- 7 Teaching of a robot program (required only for *iRPickTool* basic)
Create programs for robots which perform tracking motion.
- 8 Fine adjustment of a tracking motion
Make a fine tracking motion adjustment while performing a tracking motion actually.

2.7 RESTRICTIONS

As of 7DC3/11(V8.30P/11), the following restrictions are imposed on queue managed tracking and visual tracking.

- 1 Does not support rail tracking.
- 2 Does not support multi-view with circular tracking.
- 3 Does not support hot start mode. Disable the hot start.
- 4 Does not support hand camera.
- 5 Can not use "Distance before" function. (Can use "Time before" function)
- 6 This function cannot be used together with Space Check.

3 PREPARATION

This chapter describes hardware connection and other preparations for the start-up of a tracking system using *iRPickTool*.

Subsection 3.1.4, "Setting Up the Robot Ring" describes the settings needed only when the system contains multiple robot controllers. When the system contains only one robot controller, these settings are unnecessary. Section 3.2, "PULSECODER INSTALLATION, CONNECTION, AND SETUP" describes a procedure corresponding to the system configuration. This procedure differs according to whether the system contains a single robot controller or multiple controllers and whether a pulse multiplexer or the Ethernet Encoder is to be used when the system contains multiple robot controllers. Follow the proper procedure corresponding to the system you want to build. Sections 3.3, "CAMERA INSTALLATION AND CONNECTION" and 3.5, "GRID PATTERN" describe procedures needed only for a visual tracking system using *iRVision*. These procedures are unnecessary for a queue managed tracking system using a sensor such as a photo eye.

All other procedures are necessary in any case. Make preparations in the order in which they are described below while determining whether the relevant procedure is needed for your system.

3.1 NETWORK CREATION

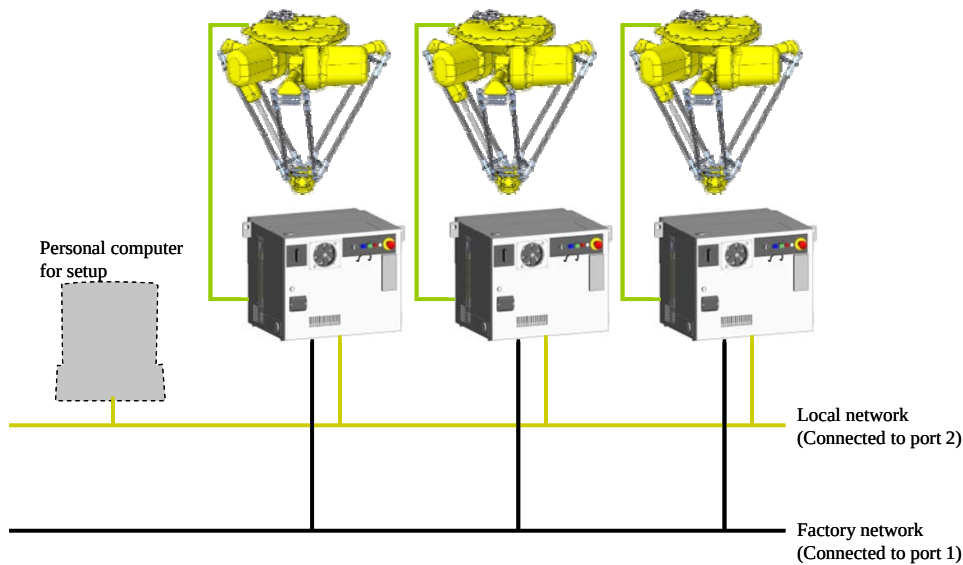
When the system contains multiple robot controllers, connect them via a network.

Even when a single robot controller is to be used for tracking, this setup operation is required since a personal computer is connected to the network to set up *iRPickTool*.

3.1.1 Connecting a Network Cable

In a tracking system using *iRPickTool*, 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*iB* controller has two Ethernet ports (see CAUTION). 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 *iRPickTool*.



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 *iRPickTool* to the local network for communication among robot controllers.



CAUTION

- 1 Some R-30iB Mate controllers have only one Ethernet port according to the specification. If you want to connect a controller to an external network, be sure to select a controller with two ports.

3.1.2 Caution on Setting up a Network

There are the following restrictions on setting up a network. Observe these restrictions when performing each step below.

- As described in Subsection 3.1.1, "Connecting a Network Cable", one controller has two ports. Set the IP address and subnet mask of each port used according to the instructions described in Subsection 3.1.3, "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 3.1.3, "Setting IP Addresses".

Observe the following restrictions 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.

3.1.3 Setting IP Addresses

Follow the procedure below to set IP addresses for each robot controller:

- Press [MENU] key on the teach pendant of the robot controller.
- Select [6 SETUP] from the pull-down menu.
- Press F1 [TYPE] key.
- Select [Host Comm] from the list.
- Position the cursor to [TCP/IP] and press [ENTER] key.
- 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.

- 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.

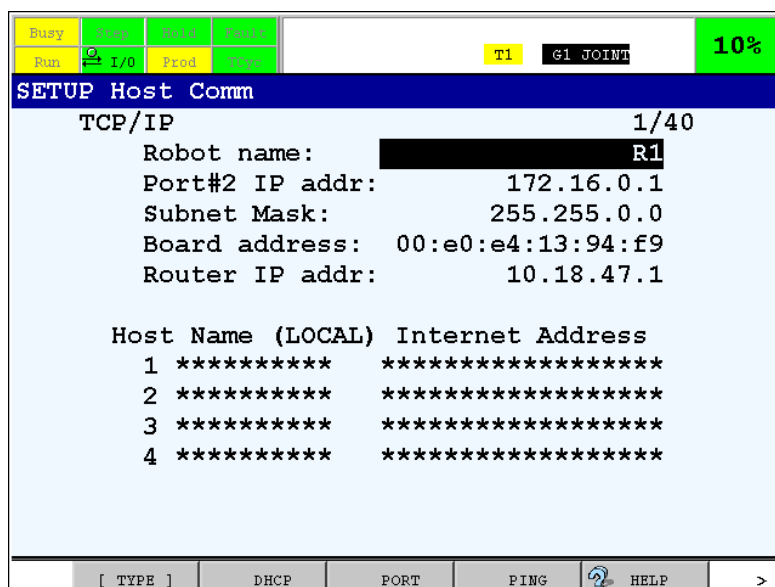
- Enter a subnet mask in [Subnet Mask].
- Enter the IP address of the default gateway in [Router IP addr].
- Press F3 [PORT] key and select a port.
- Set the IP address and subnet mask for port 2 in the same way as in steps 7 and 8.
- Turn the power to the robot controller off, then on again.

⚠ 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.

The screenshot shows the 'SETUP Host Comm' screen on a robot controller. At the top, there are status indicators: 'Busy', 'Run', 'I/O', 'Prod', 'T1', 'G1 JOINT', and '10%'. The main menu is 'SETUP Host Comm'. Under 'TCP/IP', the configuration for '1/40' is shown: 'Robot name: R1', 'Port#1 IP addr: 10.18.50.25', 'Subnet Mask: 255.255.0.0', 'Board address: 00:e0:e4:13:94:f8', and 'Router IP addr: 10.18.47.1'. Below this, there is a section for 'Host Name (LOCAL) Internet Address' with four rows of asterisks. At the bottom, there is a navigation bar with buttons: '[TYPE]', 'DHCP', 'PORT', 'PING', 'HELP', and a right arrow '>'.

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.

3.1.4 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 "R-30iA/R-30iA Mate/R-30iB/R-30iB Mate Controller Ethernet Function Operator's Manual".

Master and Slave

A robot ring consists of one master controller and other slave controllers. With *iRPickTool*, any robot controller can be the RIPE master. When the Ethernet Encoder is used, however, it is recommended that the 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 Select [6 SETUP] from the pull-down menu.
- 3 Press F1 [TYPE] key.
- 4 Select [Host Comm] from the list.

- 5 Position the cursor to [RIPE] and press [ENTER] key. The following screen appears.

Busy	Stop	Hold	Reset	T2		JOINT	4%
Run	I/O	Prod	Stop				

SETUP RIPE

ROS Ethernet Packets (MASTER) 1/21

Robot Name: R2

Port #: 2

Master IP addr:*****

Number of Members: 2

Update Interval: 400

Index	Host Name	Internet Address
1	*****	*****
2	*****	*****
3	*****	*****
4	*****	*****

[TYPE]	SLAVE	AUTO	SEND	?	HELP	>

- 6 Press F2 [SLAVE] key to display the “ROS Ethernet Packet (SLAVE)” screen as shown below. This step is unnecessary when the screen for slave screens is already open and F2 indicates MASTER.

Busy	Stop	Hold	Reset	T2		JOINT	4%
Run	I/O	Prod	Stop				

SETUP RIPE

ROS Ethernet Packets (SLAVE) 1/21

Robot Name: R2

Port #: 2

Master IP addr: 172.16.0.2

Slave IP addr: 172.16.0.3

Member Index (1 is Master): 2

Index	Host Name	Internet Address
1	*****	*****
2	*****	*****
3	*****	*****

[TYPE]	MASTER	AUTO	RCV	?	HELP	

- 7 In [Master IP addr], set the IP address of the master controller for the port used for the robot ring.

Busy	Stop	Hold	Reset			T2	JOINT	4%
Run	I/O	Prod	Prog					

SETUP RIPE

ROS Ethernet Packets (SLAVE) 3/21

Robot Name: R2

Port #: 2

Master IP addr: 172.16.0.1

Slave IP addr: 172.16.0.2

Member Index (1 is Master): 2

Index	Host Name	Internet Address
1	*****	*****
2	*****	*****
3	*****	*****

[TYPE]	MASTER	AUTO	RECV	?	HELP

- 8 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 3.1.3, "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 3.1.3.
- 9 After setting the above items, press F3 [AUTO] key. The following screen appears. A message appears, which asks you whether to restart the controller after data reception. Press F3 [YES] key.

Busy	Stop	Hold	Reset			T2	JOINT	4%
Run	I/O	Prod	Prog					

SETUP RIPE

ROS Ethernet Packets (SLAVE) 5/21

Robot Name: R2

Port #: 2

Master IP addr: 172.16.0.1

Slave IP addr: 172.16.0.2

Member Index (1 is Master): 2

Index	Host Name	Internet Address
1	*****	*****
2	*****	*****
3	*****	*****

Repower after receiving data ?

	YES	NO	ABORT

- 10 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.

Busy	Stop	Hold	Enable			T2	JOINT	4%
Run	I/O	Prod	Prog					

SETUP RIPE

ROS Ethernet Packets (SLAVE) 5/21

Robot Name: R2

Port #: 2

Master IP addr: 172.16.0.1

Slave IP addr: 172.16.0.2

Member Index (1 is Master): 2

Index	Host Name	Internet Address
1	*****	*****
2	*****	*****
3	*****	*****

Waiting for data, PREV to cancel

[TYPE]	MASTER	AUTO	RCV	?	HELP

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 Select [6SETUP] from the pull-down menu.
- 3 Press F1 ([TYPE]) key.
- 4 Select [Host Comm] from the list.
- 5 Position the cursor to [RIPE] and press [ENTER] key.
- 6 When the “ROS Ethernet Packet (SLAVE)” screen appears, press F2 [MASTER] key to display the screen for the master controller shown below. This step is unnecessary when the screen for the master controller is already open and F2 indicates SLAVE.

Busy	Stop	Hold	Enable			T2	JOINT	4%
Run	I/O	Prod	Prog					

SETUP RIPE

ROS Ethernet Packets (MASTER) 1/21

Robot Name: R1

Port #: 2

Master IP addr:*****

Number of Members: 2

Update Interval: 400

Index	Host Name	Internet Address
1	*****	*****
2	*****	*****
3	*****	*****
4	*****	*****

[TYPE]	SLAVE	AUTO	SEND	?	HELP	>

- 7 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 controller.

Busy	Stop	Hold	Enable			T2	JOINT	4%
Run	I/O	Prod	Test					

SETUP RIPE

ROS Ethernet Packets (MASTER) 3/21

Robot Name: R1

Port #: 2

Master IP addr: 172.16.0.1

Number of Members: 2

Update Interval: 400

Index	Host Name	Internet Address
1	*****	*****
2	*****	*****
3	*****	*****
4	*****	*****

[TYPE]	SLAVE	AUTO	SEND	?	HELP	>

- 8 In [Number of Members], set the number of controllers which join the robot ring.
- 9 Set [Update Interval] if necessary. At intervals of this setting, each robot is checked to see whether it is online.
- 10 After setting the above items, press F3 [AUTO] key. The following screen appears.

Busy	Stop	Hold	Enable			T2	JOINT	4%
Run	I/O	Prod	Test					

SETUP RIPE

ROS Ethernet Packets (MASTER) 4/21

Robot Name: R1

Port #: 2

Master IP addr: 172.16.0.1

Number of Members: 2

Update Interval: 400

Index	Host Name	Internet Address
1	*****	*****
2	*****	*****
3	*****	*****
4	*****	*****

Put all SLAVES in AUTO mode.

			CONTINUE	ABORT

- 11 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.

Busy	Stop	Hold	Power		T2	JOINT	4%
Run	I/O	Prod	Stop				

SETUP RIPE

ROS Ethernet Packets (MASTER) 4/21

Robot Name: R1

Port #: 2

Master IP addr: 172.16.0.1

Number of Members: 2

Update Interval: 400

Index	Host Name	Internet Address
1	R1	172.16.0.1
2	R2	172.16.0.2
3	*****	*****
4	*****	*****

Cycle power for setting to take effect?

			YES	NO

- 12 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 [CONTINUE] 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.

3.2 PULSECODER INSTALLATION, CONNECTION AND SETUP

Install a Pulsecoder and connect it to a robot controller.

NOTE

In this manual, a device for detecting the conveyor travel distance is called a "Pulsecoder". On the teach pendant and *iRPickTool* setup screen, the term "encoder" is used as a synonym of "Pulsecoder".

With *iRPickTool*, the α A1000S Pulsecoder and incremental Pulsecoders are available.

For the configuration required for using each Pulsecoder with the R-30iB and R-30iB Mate controllers, see the table below.

Table 3.2 Configuration required for each controller

Pulsecoder	R-30iB controller	R-30iB Mate controller
α A1000S Pulsecoder (A860-0372-T001)	Table E.1.1 (a) in Appendix E, "A1000S PULSECODER"	Table E.1.2 (a) in Appendix E, "A1000S PULSECODER"
Incremental Pulsecoder (A860-0301-T001 to T004)	Table F.1.1 (a) in Appendix F, "R-30iB: Incremental Pulsecoder"	Table F.1.2 (a) in Appendix F, "R-30iB Mate: Incremental Pulsecoder"

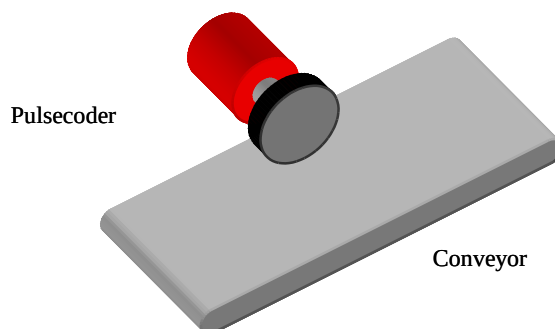
3.2.1 Installing a Pulsecoder

Install a Pulsecoder on a conveyor.

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.



3.2.2 Connecting the Pulsecoder

Connect the Pulsecoder to a robot controller.



CAUTION

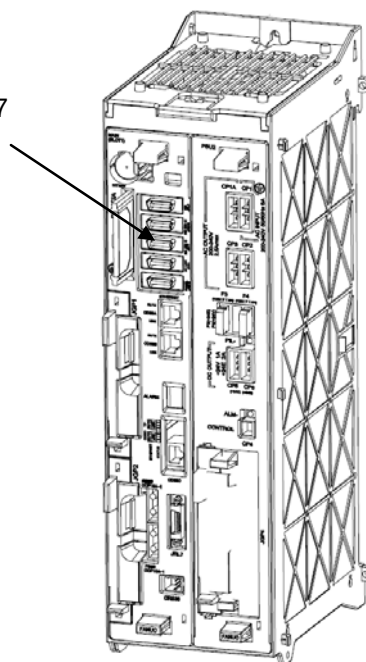
If you install the Pulsecoder cable near the power supply cable, a robot may be not able to read the Pulsecoder signal correctly due to the noise of the power supply cable. Please keep the Pulsecoder cable away from the power supply cable.

3.2.2.1 Hardware to which to connect a Pulsecoder

MAIN board (available only with the α A1000S Pulsecoder)

When you use the α A1000S Pulsecoder from the two types of available Pulsecoders, you can connect the Pulsecoder to the MAIN board. Only one Pulsecoder can be connected to the MAIN board.

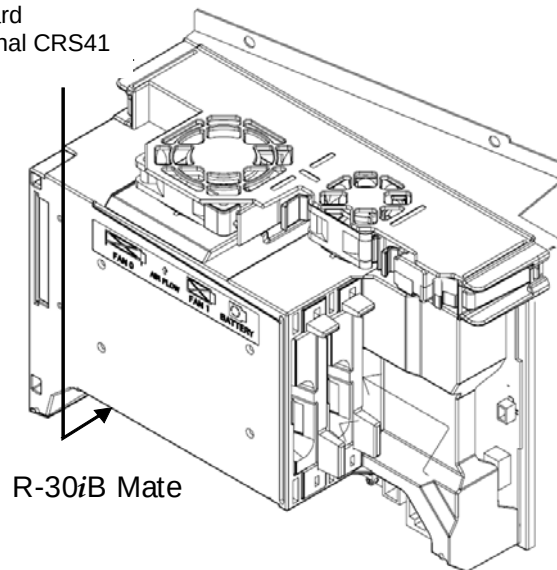
MAIN board
Pulsecoder Terminal JD17



R-30iB

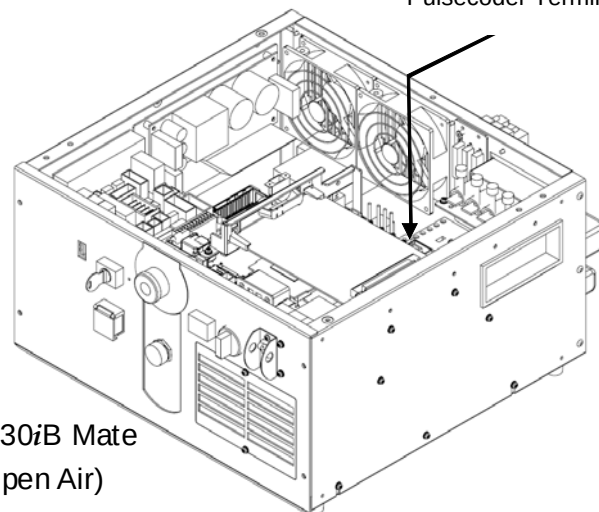
Fig. 3.2.2.1 (a) Connection to the MAIN board on the R-30iB

MAIN board
Pulsecoder Terminal CRS41



R-30iB Mate

MAIN board
Pulsecoder Terminal CRS41



R-30iB Mate
(Open Air)

Fig. 3.2.2.1 (b) Connection to the MAIN board on the R-30iB Mate



CAUTION

You can connect the α A1000S Pulsecoder to the MAIN board when only one Pulsecoder is used. If you want to use an incremental Pulsecoder or two or more Pulsecoders, use a line tracking interface board described later.

Line tracking interface board

You can select a line tracking interface board either of the wide mini slot or mini slot type.

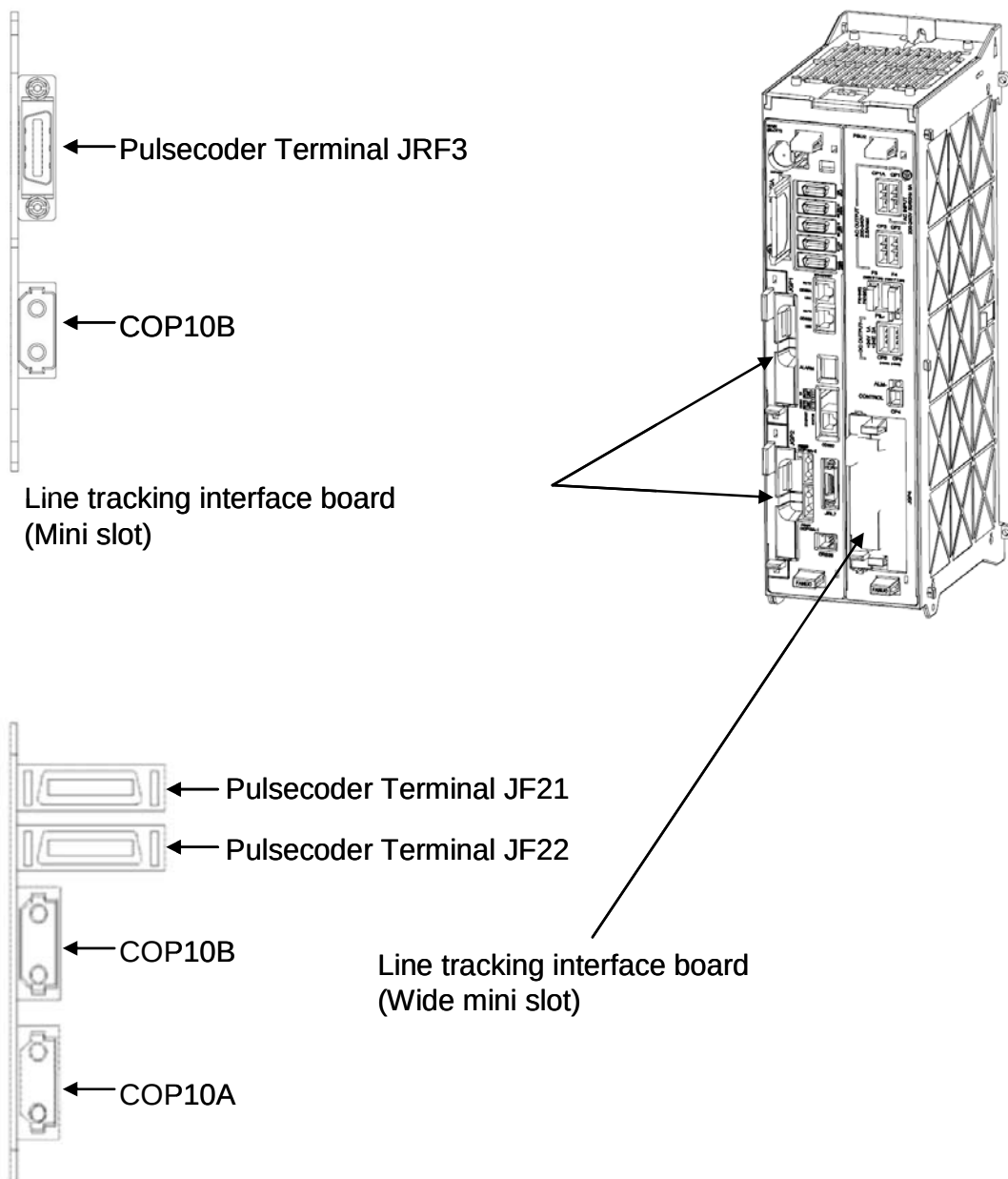


Fig. 3.2.2.1 (c) Connection to a line tracking interface board on the R-30iB

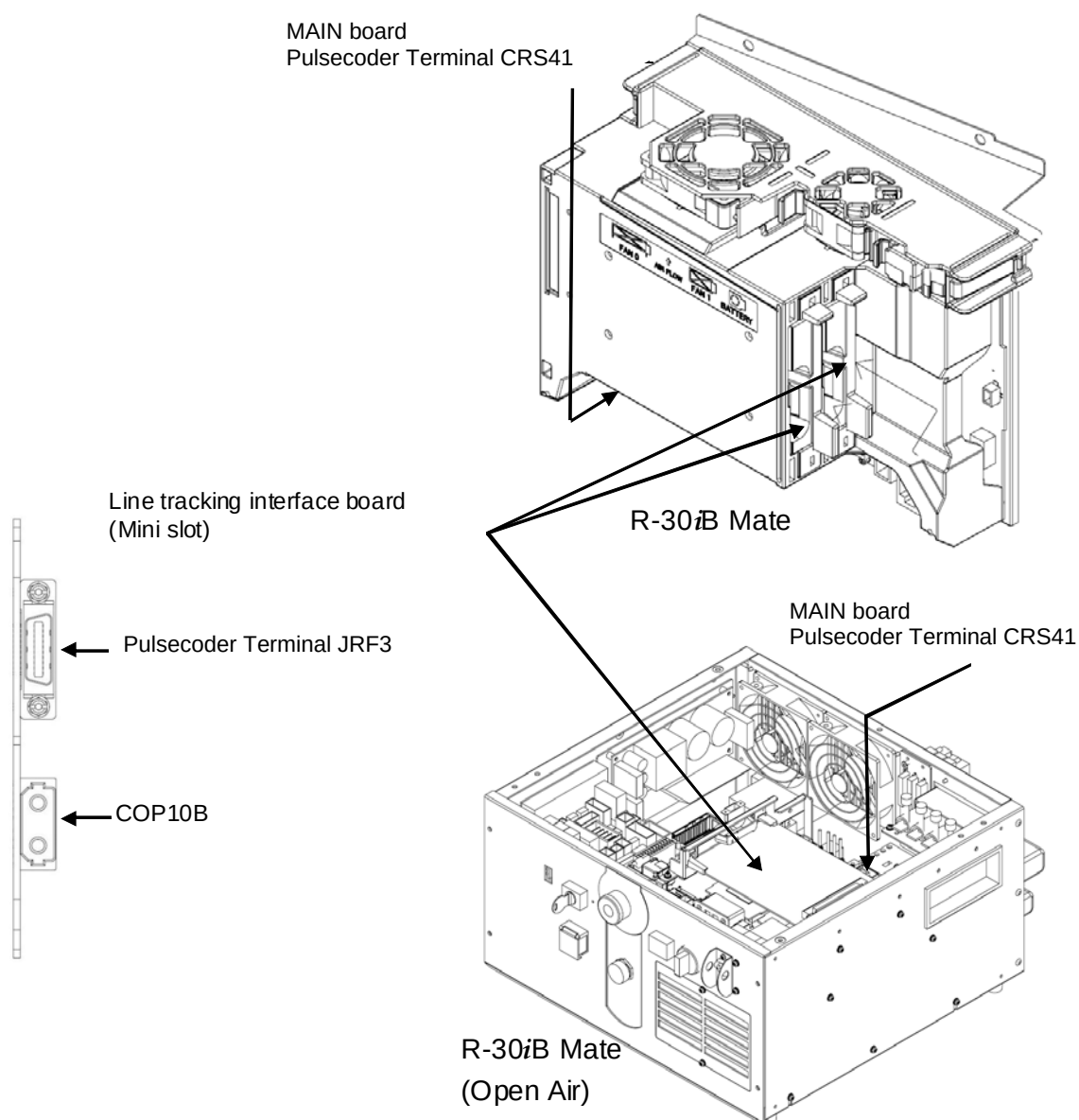


Fig. 3.2.2.1 (d) Connection to a line tracking interface board on the R-30iB Mate

NOTE

With the R-30iB Mate, only a line tracking interface board of the mini slot type can be selected.

3.2.2.2 When one robot is used

Connect the Pulsecoder with the robot controller directly with a cable.

Connecting the Pulsecoder to the MAIN board

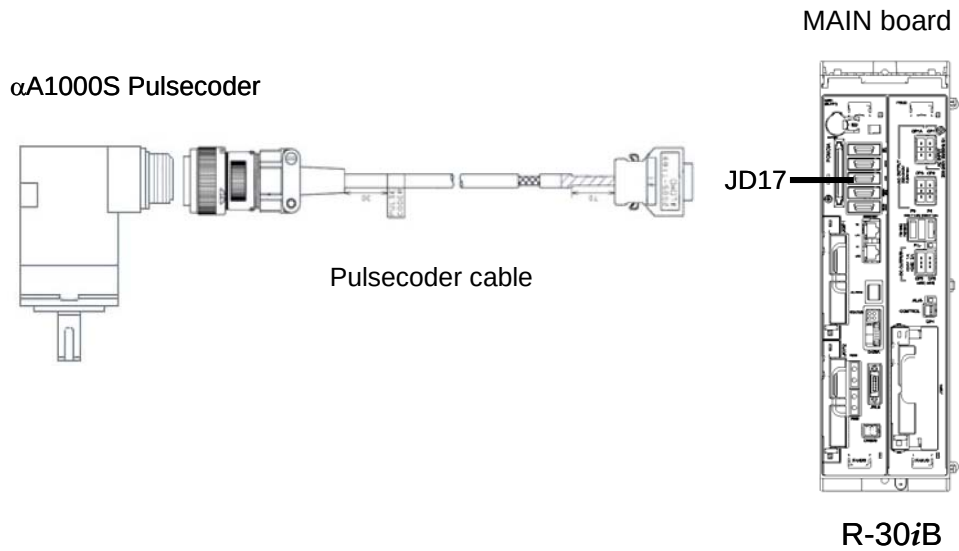


Fig. 3.2.2.2 (a) Connecting the Pulsecoder to the MAIN board on the R-30iB

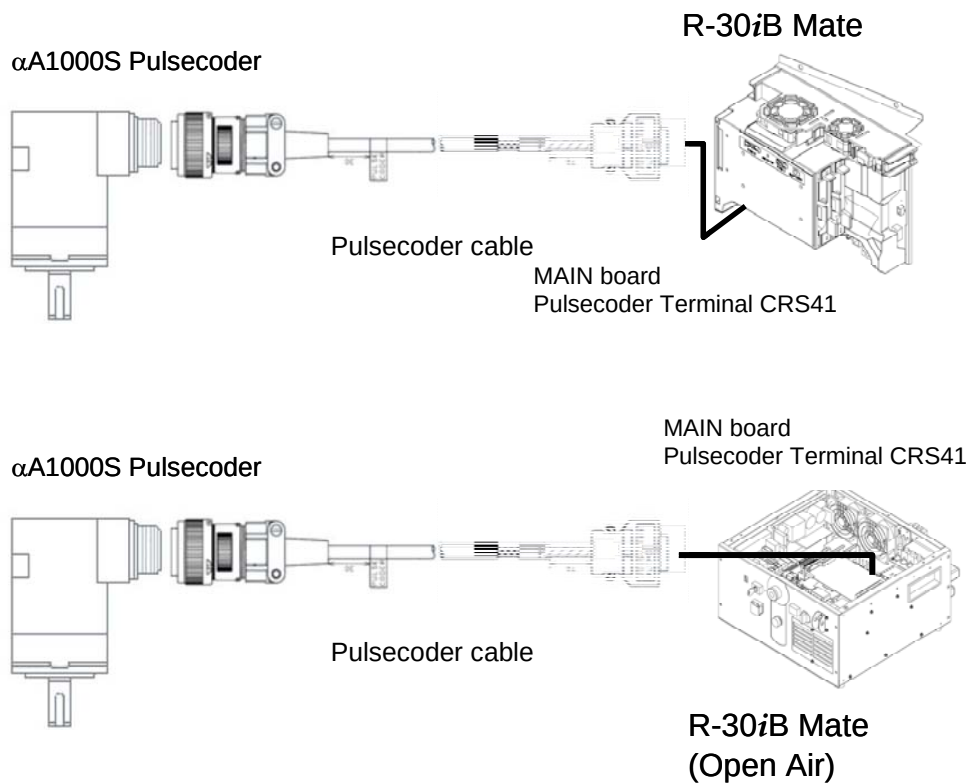


Fig. 3.2.2.2 (b) Connecting the Pulsecoder to the MAIN board on the R-30iB Mate

Connecting the Pulsecoder to a line tracking interface board

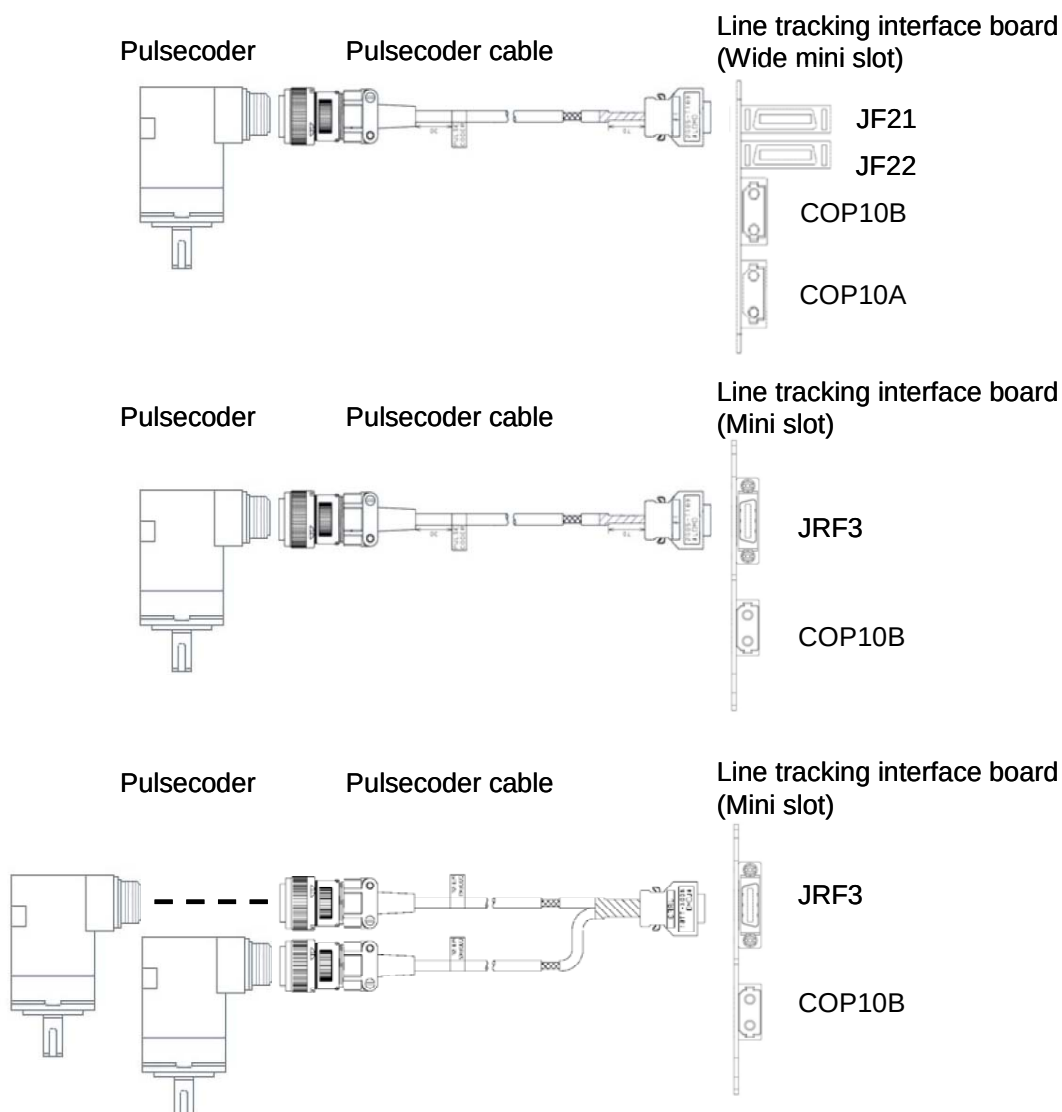


Fig. 3.2.2.2 (c) Connecting the Pulsecoder to a line tracking interface board on the R-30iB

When a line tracking interface board (wide mini slot) is used to connect only one Pulsecoder, connect the Pulsecoder cable to the JF21 connector.

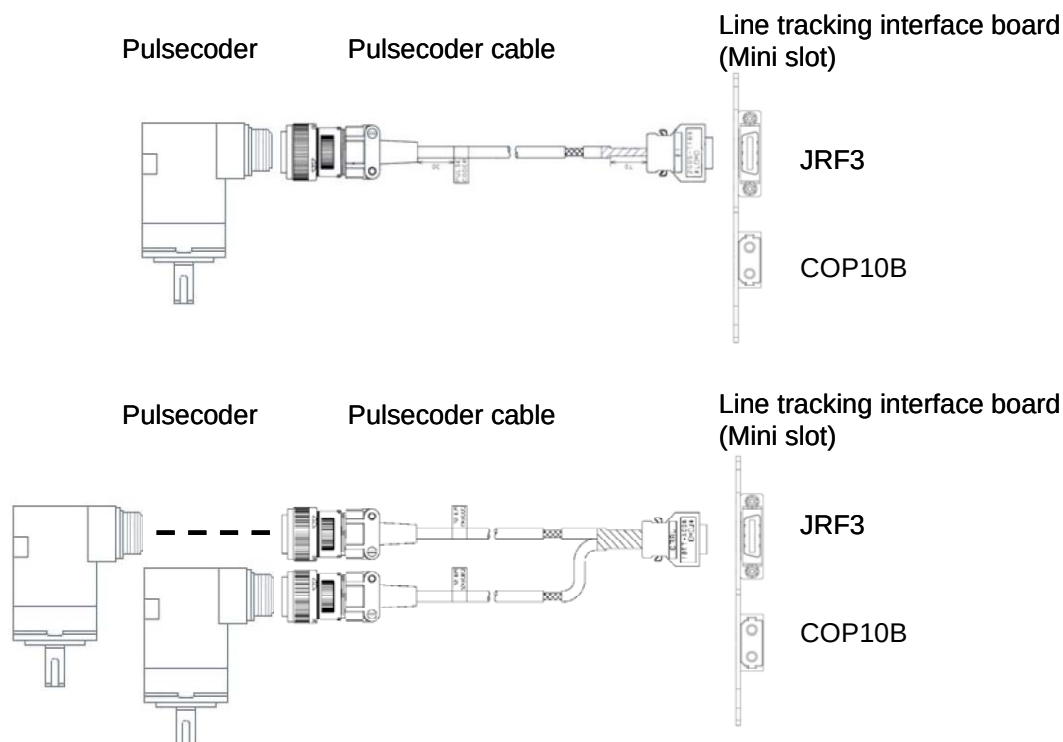


Fig. 3.2.2.2 (d) Connecting the Pulsecoder to a line tracking interface board on the R-30iB Mate

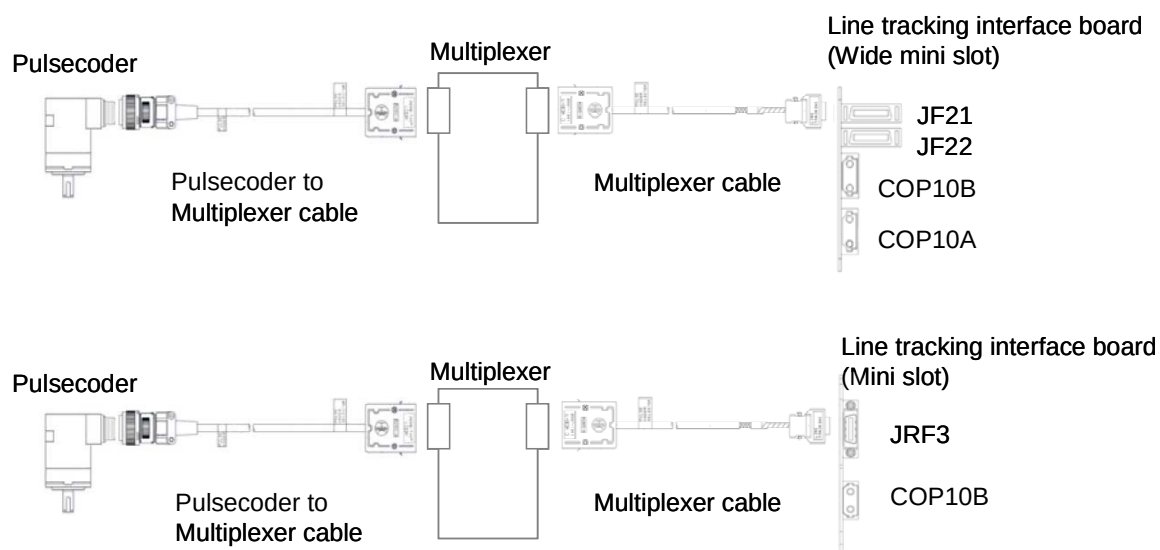
3.2.2.3 When multiple robots are used

When the Ethernet Encoder is used

In the same way as when one robot is used, connect the Pulsecoder to a robot controller directly with a cable. To reduce the communication load, use the robot controller connected to the Pulsecoder as the master controller of the robot ring.

When a pulse multiplexer is used (only for an Incremental Pulsecoder)

Connect the Pulsecoder to the robot controllers via a pulse multiplexer.



When a line tracking interface board (wide mini slot) is used to connect one Pulsecoder, connect the Pulsecoder cable to the JF21 connector on each line tracking interface board (wide mini slot). For details

of how to connect a pulse multiplexer and Pulsecoder cable, see Subsection F.3.2, "Connecting to Pulse Multiplexer" in Appendix.

⚠ CAUTION

- 1 A pulse multiplexer can be used only when the R-30iB controller or R-30iB Mate controller, and incremental Pulsecoders are used. If you want to use another combination, that is, to use the R-30iB Mate (Open Air) controller or α A1000S Pulsecoder, use the Ethernet Encoder described later.
- 2 Please use about 5V, 10W for a power supply of pulse multiplexer.

3.2.3 Setting up the Pulsecoder

Set the parameters for the Pulsecoder.

When one robot is used or multiple robots are used with a pulse multiplexer

- 1 Press [MENU] key on the teach pendant of the robot controller.
- 2 Select [6 SETUP].
- 3 Press F1 [TYPE] key.
- 4 Select [Encoders]. The following screen appears.

Encoder Number:	1
1 Encoder Axis:	0
2 Encoder Type:	INCREMENTAL
3 Encoder Enable:	ON
Current Count (cnts):	0
4 Multiplier (ITP/update):	1
5 Average (updates):	1
6 Stop Threshold (cnt/updt):	0
7 Simulate: Enable:	OFF
8 Rate (cnt/updt):	0
9 HDI port Id:	1

- 5 To reference or set up information of another [Encoder Number], press F3 [ENCODER] key and enter [Encoder Number]. When using a line tracking interface board of the wide mini slot type, select 1 if the Pulsecoder is connected to the JF21 connector on the line tracking interface board or select 2 if the Pulsecoder is connected to the JF22 connector. When using a line tracking interface board of the mini slot type, select 1 if one Pulsecoder is used or select the relevant number if two Pulsecoders are used.

⚠ CAUTION

When using the Pulsecoder connected to the main board together with the Line tracking interface board, you must assign the encoder number 1, 2 to Pulsecoder connected to the Line tracking interface board connected, encoder number 3 to encoder connected to the main board.

For example, if you assign the encoder number 1 to Pulsecoder connected to the main board, it will not be able to use the channel 1 of the Pulsecoder connected to the Line tracking interface board.

- 6 Position the cursor to [1 Encoder Axis] and enter the axis number of the Encoder. Generally, an incremental Encoder is used; valid settings are 1 to 8. For single axis tracking, this field is set to 1. For dual axis tracking, set the Encoder Axis number of Pulsecoder 1 to 1 and set the Encoder Axis number of Pulsecoder 2 to 2.

NOTE: There is no need to setup Encoder Axis number when use α A1000S Pulsecoder which is connected to Main board.

NOTE

If you want to use the HDI signal on a conveyor using an external sensor such as a photo-eye, see Appendix C, "HIGH SPEED SCANNING".

- 7 Move the cursor to the [2 Encoder Type]. Specify the type of tracking Pulsecoder that is to be used in the Table.3.2.3 (a).

Table 3.2.3 (a) Encoder Type

No	Encoder Type	COMMENTS
1	INCREMENTAL	Select "INCREMENTAL" when the Incremental Pulsecoder is used.
2	Serial ABS	Select "Serial ABS" when α A1000S Pulsecoder is used as absolute type encoder.
3	SERVOCONV	"SERVOCONV" is used when Servo Conveyor Line Tracking function. See section Appendix D "SERVO CONVEYER LINE TRACKING FUNCTION" for detail.
4	VIRTUAL	This mode allows you to run the line tracking motion with simulated encoder without actual hardware.
5	Serial INC	Select "Serial INC" when α A1000S Pulsecoder is used as incremental type encoder and connected to Line tracking interface board or Separate Detector Unit (SDU).
6	Main Serial INC	Select "Main Serial INC" when α A1000S Pulsecoder is used as incremental type encoder and connected to Pulsecoder terminal of Main board.
7	Main Serial ABS	Select "Main Serial INC" when α A1000S Pulsecoder is used as absolute type encoder and connected to Pulsecoder terminal of Main board.

NOTE

If "1 INCREMENTAL", "2 Serial ABS", or "5 Serial INC" is selected with the controller has no Line tracking interface board (e.g. when connect α A1000S Pulsecoder to Main board), "SRVO-056 FSSB com error 2 (G: Group number A: Axis number)" could occur. Set correct Encoder Type.

NOTE

If "1 INCREMENTAL" is selected but Pulsecoder is actually not connected (Line tracking interface board is installed), "SRVO-082 DAL alarm (Track encoder: Encoder number)" warning could occur. If the Encoder Axis number is set to an axis number which is used for motor control, the controller turns off servo with SRVO-082 warning. If you do not use Pulsecoder, clear the Encoder Axis number to zero.

- 8 Position the cursor to [5 Average (updates)] and enter the number of sample counts used to calculate the average speed for the Encoder (the width of the moving average). Usually, enter 10.

⚠ CAUTION

If the value of [Average (updates)] is small and the conveyor used does not move smoothly, the robot may not move smoothly because it attempts to follow the motion of the conveyor. In this case, increase the value of [Average (updates)].

- 9 Use the default values for other items.
- 10 After the axis number of the Encoder is changed, it is necessary to start the robot controller in the cold start mode. In this case, turn the power to the robot controller off, then on again.

⚠ CAUTION

The encoder counts of an incremental Encoder are set to 0 when the power to the robot controller is turned on. In a system in which multiple robots work for one conveyor, it is necessary to turn the power to all robot controllers on in the conveyor stopped state to set the encoder counts to the same value. If the power is turned on while the conveyor is moving or the power to only one controller is turned off, then on again after the conveyor starts moving, the encoder counts recognized by multiple robot controllers become different, resulting in incorrect tracking.

When multiple robot controllers are used with the Ethernet Encoder

Setup on the master

Follow the procedure below to set up the Pulsecoder on the robot controller connected to the Pulsecoder.

- 1 Perform steps 1 to 8 described in "When one robot is used or multiple robots are used with a pulse multiplexer". 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.

Busy	Run	Stop	Ready	A_M LINE 0	T2	ABORTED	G1 JGFRM	10%
Run	I/O	Prod	Stop					

SETUP Encoders		10/11
2	Encoder Type:	INCREMENTAL
3	Encoder Enable:	OFF
	Current Count (cnts):	0
4	Multiplier (ITP/update):	1
5	Average (updates):	1
6	Stop Threshold (cnt/updt):	0
7	Simulate: Enable:	OFF
8	Rate (cnt/updt):	0
9	HDI port Id:	1
10	Ethernet Master RIPE Id:	0
11	Ethernet Master Encoder:	0

[TYPE]	ENCODER			

- 2 Position the cursor to [Ethernet Master RIPE Id] and enter the RIPE ID of the controller. The RIPE ID is the value of system variable \$PH_ROSIP.\$MY_INDEX. For the master controller of the robot ring, the RIPE ID is 1.
- 3 Position the cursor to [Ethernet Master Encoder] and enter the number of the selected Encoder (target Encoder for setup). 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 if two Pulsecoders are used.

- 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 Ethernet encoder, all controllers connected with RIPE must have the same software version.

3.3 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. GigE camera is not available now.

Checking the camera setting

Change the setting of the camera unit for iRVision. For details, refer to the chapter, "SETUP" in the "R-30iB/R-30iB Mate CONTROLLER SENSOR MECHANICAL UNIT/CONTROL UNIT OPERATOR'S MANUAL".

Connecting the camera

Connect the camera to a robot controller. For details, refer to the chapter, "SETUP" in the "R-30iB/R-30iB Mate CONTROLLER SENSOR MECHANICAL UNIT/CONTROL UNIT OPERATOR'S MANUAL".

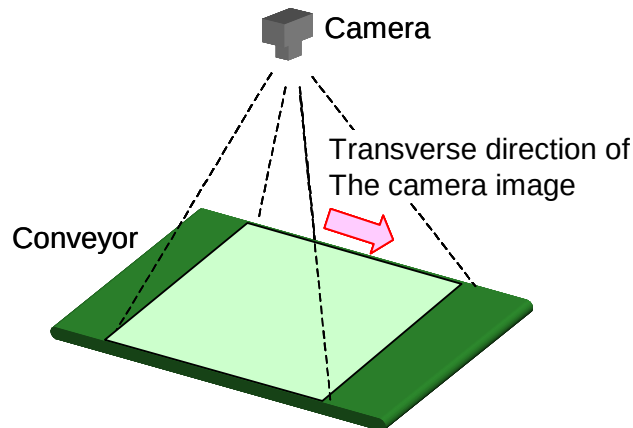
**CAUTION**

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

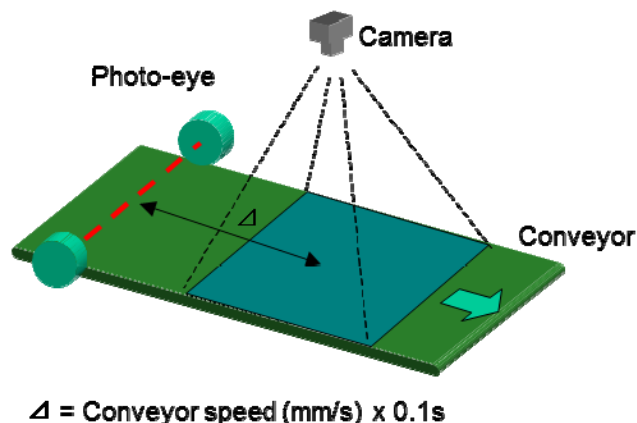
Attach the lens to the camera and install the camera over the conveyor. Orient the optical axis of the camera so that it is perpendicular to the conveyor plane.

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.



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 signal. After the DI 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 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 4.5, "SETUP OF A SENSOR".



3.4 SETTING OF THE TCP

Set the TCP for touch-up used for tracking frame setting and 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 "R-30iB/ R-30iB Mate CONTROLLER OPERATOR'S MANUAL (Basic Operation)".

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 TCP is invalidated and it is necessary to set a TCP 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 4.11, "TEACHING THE POSITION TO ROBOTS" or 6.12, "TEACHING THE POSITION TO ROBOTS".

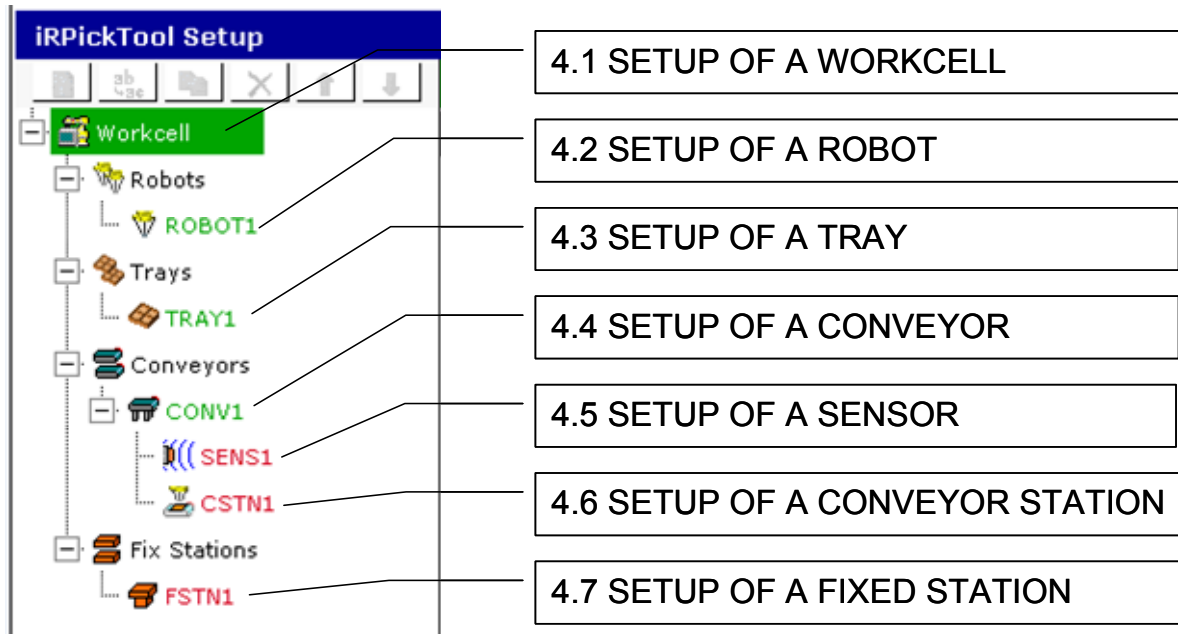
3.5 GRID PATTERN

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 in the chapter, "CALIBRATION GRID" in the "iRVision OPERATOR'S MANUAL (Reference)". FANUC provides a standard set of calibration grids. The use of them is recommended.

4 SETUP OF THE BASIC FUNCTIONS

This chapter describes the *iRPickTool* setup procedure - for its basic functions in particular. For information about the setup for a configuration that includes Plug & Play functions, see Chapter 6, "SETUP OF PLUG & PLAY FUNCTIONS".

The items that are set in Section 4.1, "SETUP OF A WORKCELL" to Section 4.7, "SETUP OF A FIXED STATION" correspond to the *iRPickTool* setup screen as shown in the figure below.



The teaching of the basic motions will be completed by performing the procedures described in Section 4.1, "SETUP OF A WORK CELL" to Section 4.11, "TEACHING THE POSITION TO ROBOTS". Section 4.12, "FINE ADJUSTMENT OF THE TRACKING MOTION" describes how to optimize the tracking motion of the robot after completing the teaching of the basic motions. Section 4.13, "SYSTEM STARTUP" contains the notes on system startup for a multiple robot system. Section 4.14, "SETUP OF PRE-GROUPING FUNCTION" contains the specific setup procedure of the Pre-grouping function.

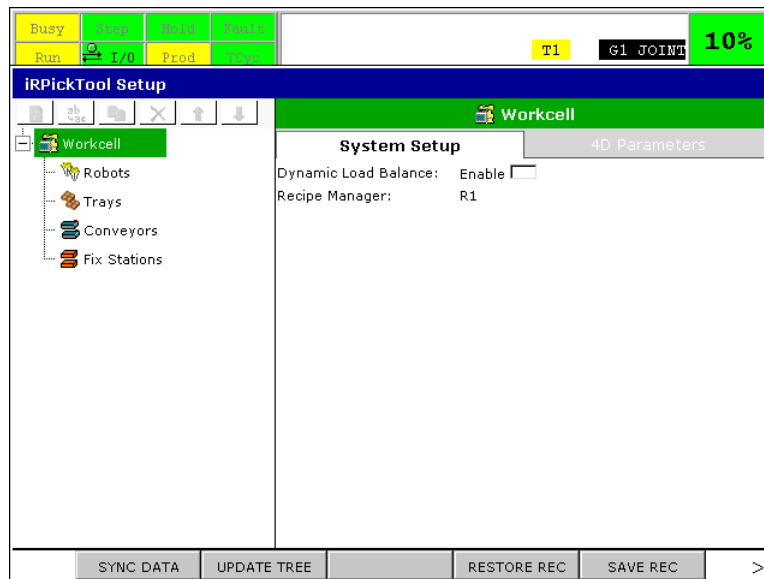
Basic operations

With *iRPickTool*, 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.

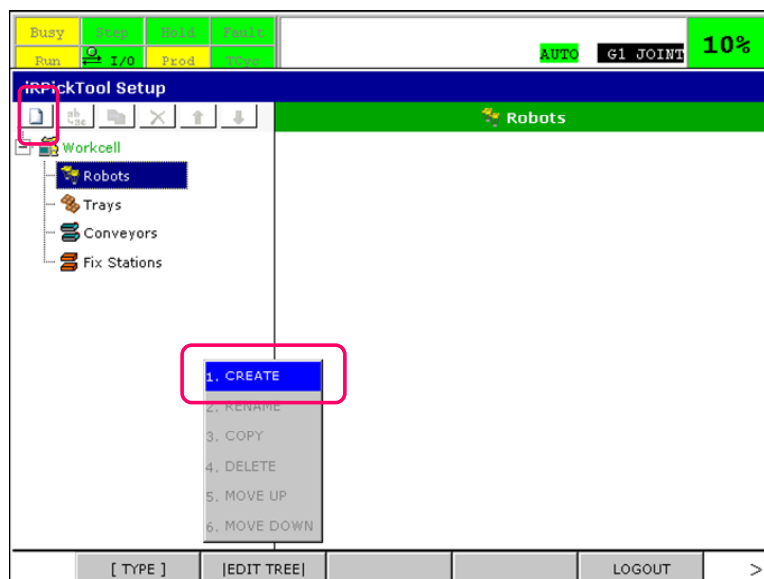
iRPickTool may be set up from any controller in a work cell. Note, however, that two or more controllers cannot be used simultaneously for the setup of *iRPickTool*.

Creating a new object

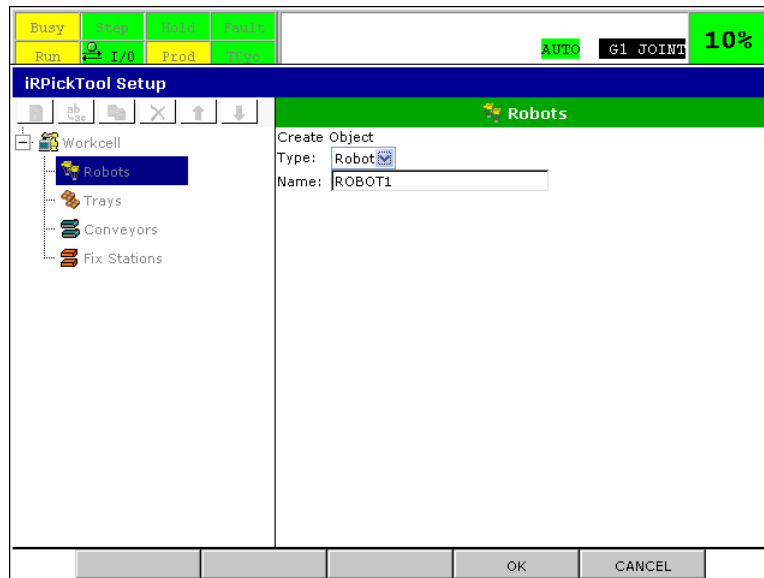
Select [6 SETUP] from [MENU] of the teach pendant and [iRPickTool] from F1 [TYPE] key. The iRPickTool setup screen is displayed, as shown below.



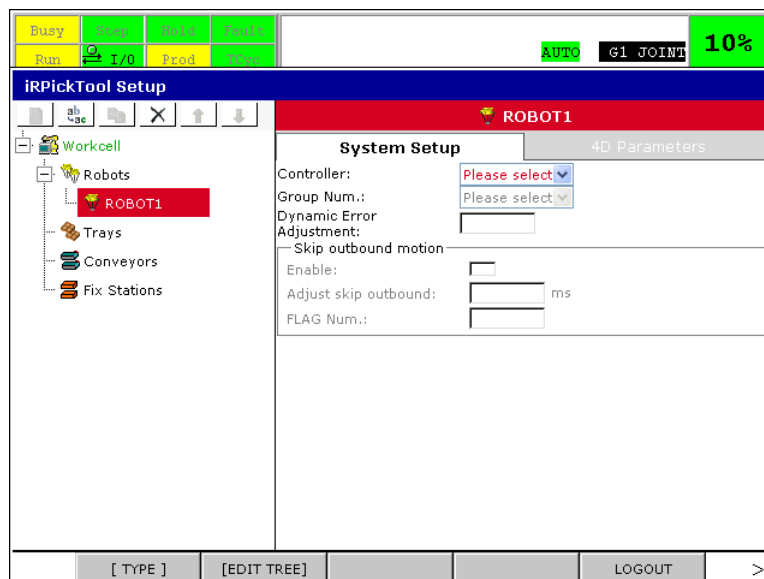
In the tree view on the left side, select the object collection corresponding to the object you want to create. For example, if you want to create a robot, select [Robots]. Then, click [1. CREATE] of F2 [EDIT TREE] key or the  button above the tree.



A screen as shown below is displayed. Enter a name that is not identical to the name of any other object. You can specify a name consisting of up to 20 one-byte characters. The name must not contain a hyphen (-) or begin with a numeric character. Also, the default name can be used as is. Press F4 [OK] key to create an object. To quit creating an object, press F5 [CANCEL] key.

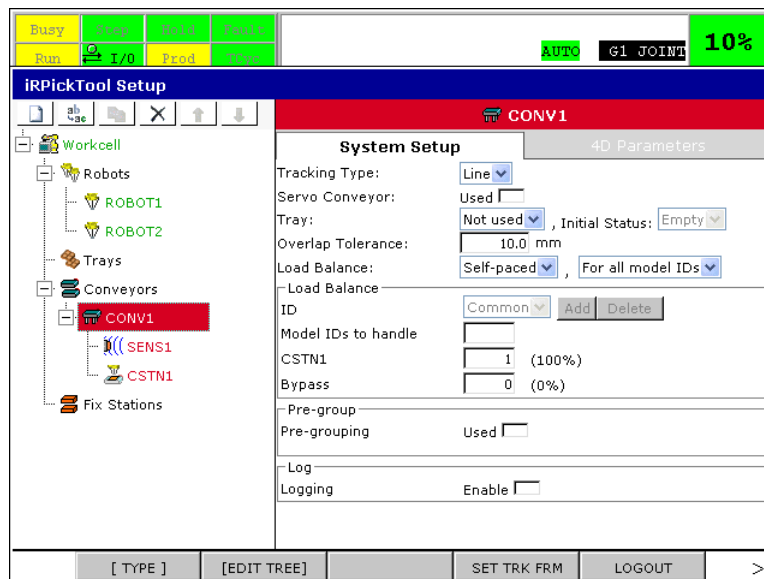


Once you have created a new object, its setup screen is opened, with the created object selected. If you have created a new robot, the screen looks as shown below.

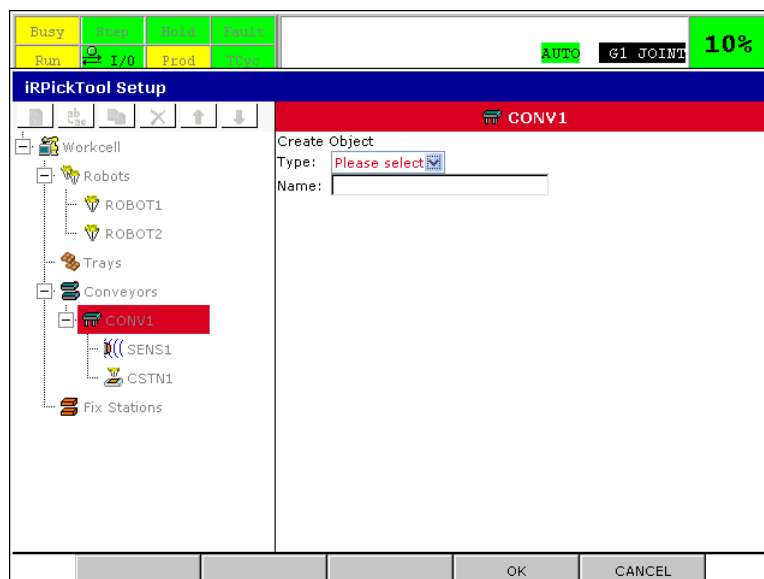


Creating a new sensor and a new conveyor station

If you create a new conveyor, a sensor and a conveyor station are automatically created, as shown below.



If more than one robot operates on the conveyor, add one or more conveyor stations. If there is more than one sensor on the conveyor, add one or more sensors. Click [1. CREATE] of F2 [EDIT TREE] key or the button above the tree, with the conveyor selected. A screen as shown below is displayed.



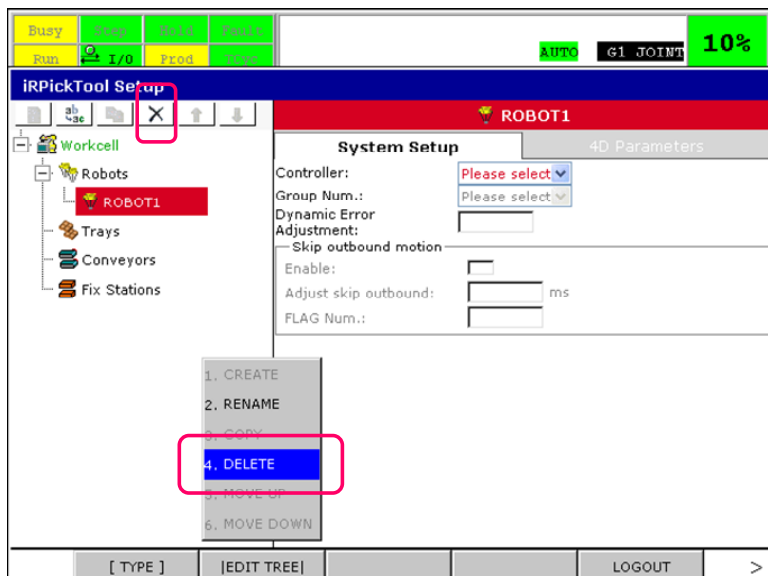
In [Type], specify whether the object you are adding is a conveyor station or sensor. The default name appears in [Name], as when you create a new object of another type. Create a new object in the same way as with an object of another type.

Setting up an object individually

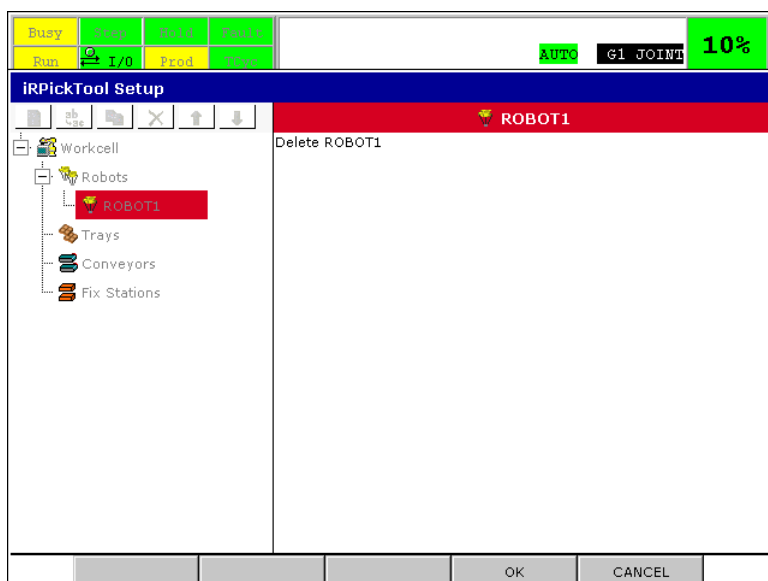
In the tree view on the left side of the iRPickTool setup screen, select an object you want to set up. Then, set each item in the setup screen on the right side. For details about the setup methods of the individual objects, see Section 4.1, "SETUP OF A WORK CELL" to Section 4.7, "SETUP OF A FIXED STATION". Objects that are displayed in red on the tree represent those in an untaught status. Once objects have been taught, they are displayed in green.

Deleting an object

In the tree view on the left side of the iRPickTool setup screen, select an object you want to delete. Then, click [4. DELETE] of F2 [EDIT TREE] key or the  button above the tree.

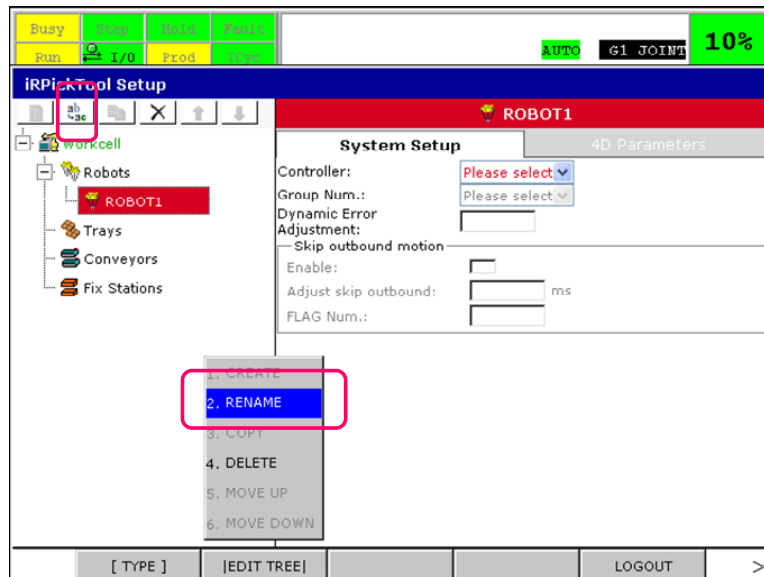


A screen as shown below is displayed. Press F4 [OK] key to delete the object. To quit deleting the object, press F5 [CANCEL] key.

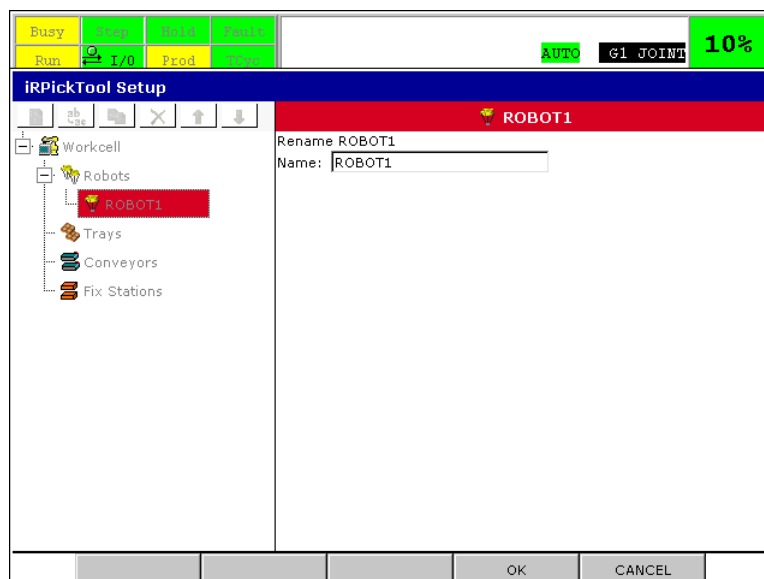


Renaming an object

In the tree view on the left side of the iRPickTool setup screen, select an object you want to rename. Click [2. RENAME] of F2 [EDIT TREE] key or the  button above the tree.

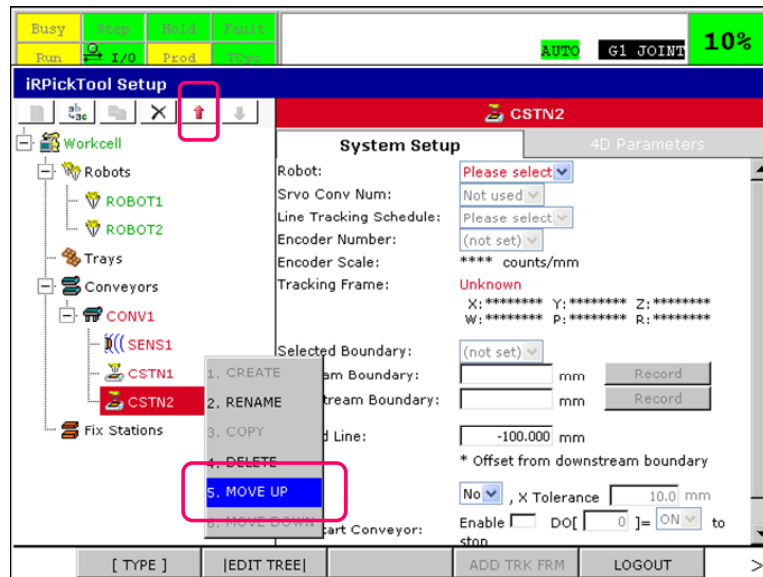


A screen as shown below is displayed. Enter the new name of the object, and press F4 [OK] key. To quit renaming the object, press F5 [CANCEL] key.



Moving an object in the order

Use this function to move a conveyor station in the order. In the tree view on the left side of the iRPickTool 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.

**CAUTION**

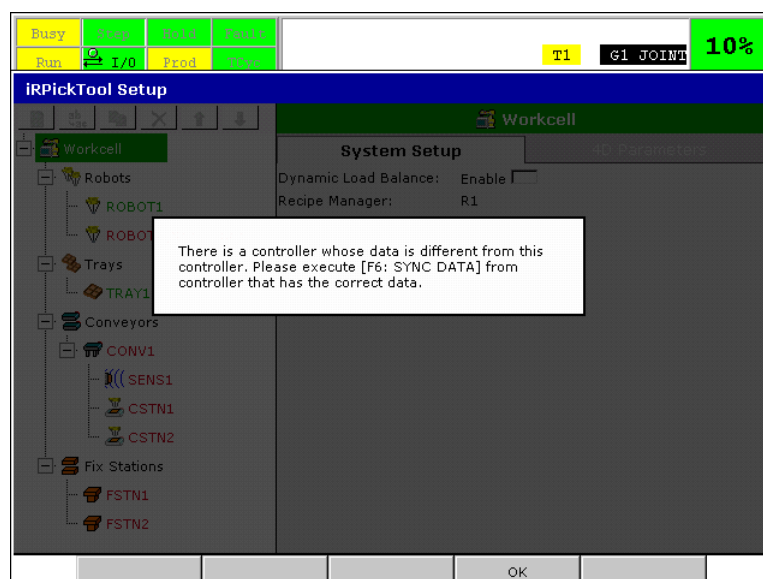
Conveyor stations are placed from the upstream of the conveyor to the downstream in the same order they are displayed in the tree view on the screen.

Synchronizing data

The settings of *iRPickTool* 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 *iRPickTool* 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 *iRPickTool* settings are the same for all the controllers.

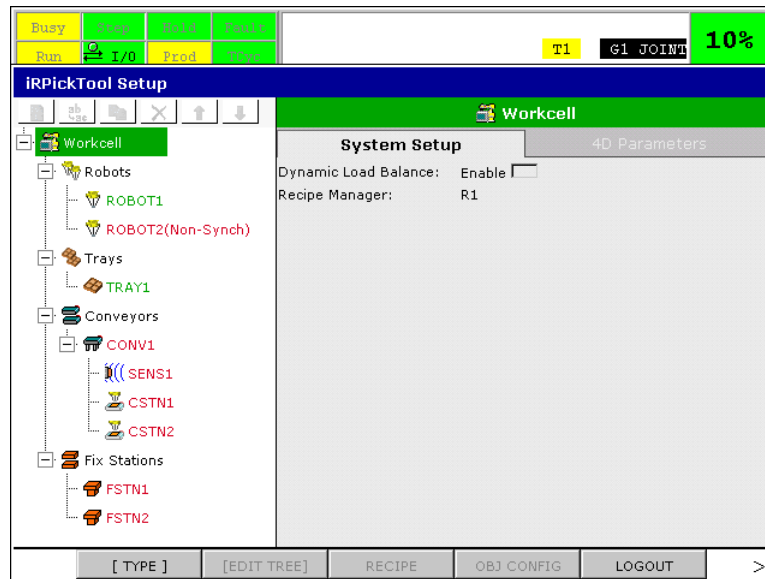
- 1 If you open the *iRPickTool* setup screen when there is any data inconsistency, a screen as shown below is displayed.



4. SETUP OF THE BASIC FUNCTIONS

B-83604EN/04

- Press F4 [OK] key. The screen enters a state where you cannot change the existing settings, as shown below.



- Check the settings of each individual object to see whether it is appropriate to transfer those settings to the other controllers. To perform synchronization based on the settings of another controller, press F5 [LOGOUT] key to close the iRPickTool setup screen. Then, open the iRPickTool setup screen for that other controller and check its settings as instructed above.



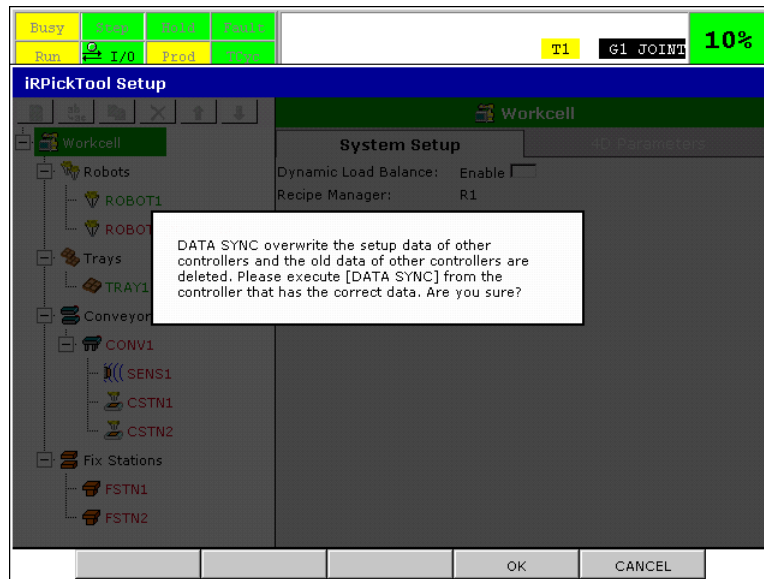
CAUTION

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 controller to the other controllers.

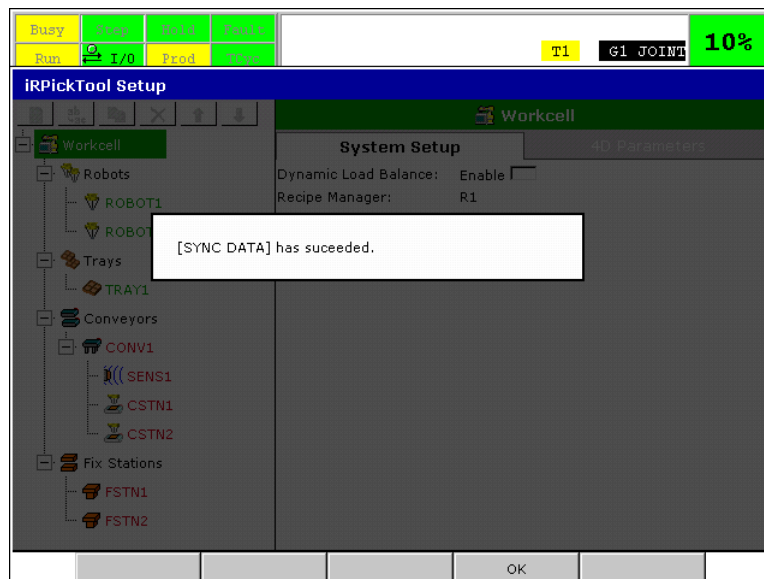
- After making sure that it is appropriate to transfer the settings to the other controllers, press the [NEXT] key on a desired iRPickTool setup screen to display the function key shown below.



- 5 Press F1 [SYNC DATA] key. When a screen as shown below is displayed, press F4 [OK] key.



- 6 When data synchronization is complete, a screen as shown below is displayed. Press F4 [OK] key.



Refreshing the tree

Since the iRPickTool setup screen does not automatically reflect 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.

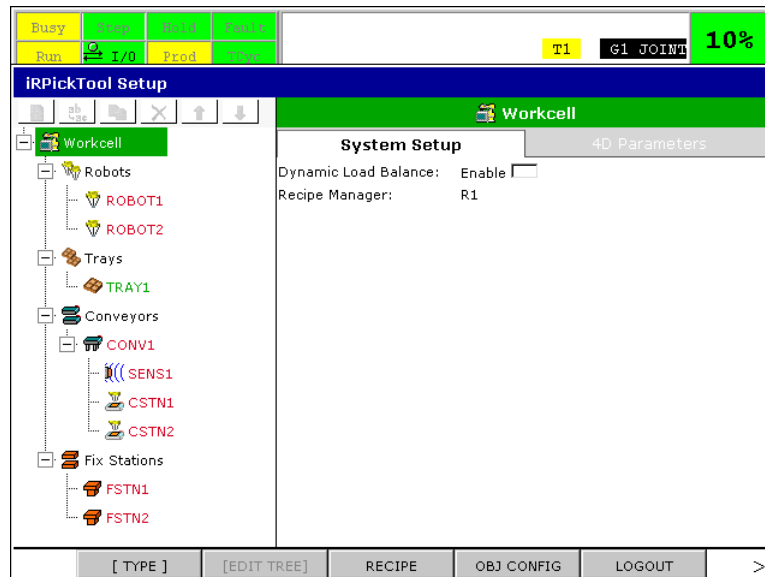
- If an offline controller goes online when the iRPickTool setup screen is open, this change in the status is not automatically reflected in the setup screen.

In these cases, press the [NEXT] key on a desired iRPickTool setup screen to display the function key shown below and then press F2 [UPDATE TREE] key.



4.1 SETUP OF A WORKCELL

Set up a work cell.



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 *iRPickTool* Recipes are stored. For more information on Recipes, see the section 7.1, “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.

Setting the number of objects

Change the maximum number of objects that can be set, on an object-by-object basis.

- 1 Press F4 [OBJ CONFIG] key. A screen as shown below is displayed.

Busy	Stop	Hold	Ready	MAIN LINE 0 T1 ABORTED G1 JOINT		10%
Run	I/O	Prod	Convey			

iPickTool Setup			
	Cur Num	Cur Max Num	New Max Num
Robot:	2	4	-> 4
Tray:	1	2	-> 2
Sensor:	1	4	-> 4
Conveyor:	1	4	-> 4
Conveyor Station:	2	8	-> 8
Fix Station:	2	8	-> 8

		OK	CANCEL

- 2 In the [New Max Num] field of the object whose number you want to increase, enter the new maximum number. Shown below is an example of changing the maximum numbers of trays and conveyors.

Busy	Stop	Hold	Ready	MAIN LINE 0 T1 ABORTED G1 JOINT		10%
Run	I/O	Prod	Convey			

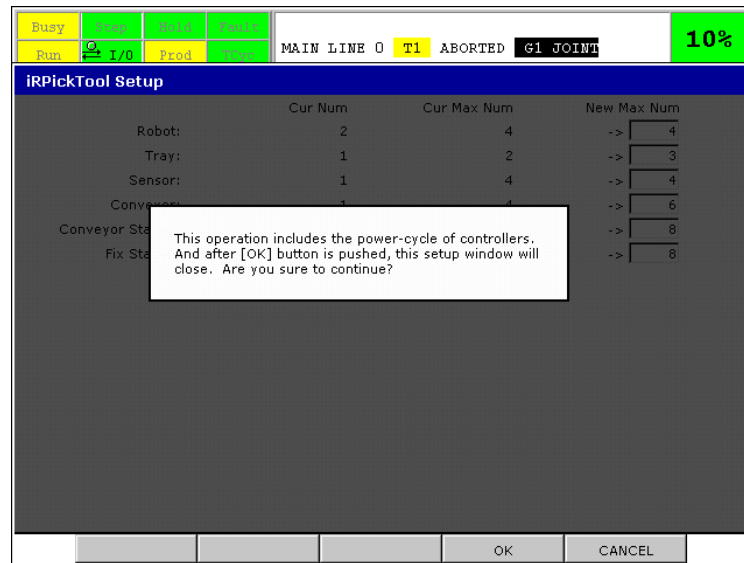
iPickTool Setup			
	Cur Num	Cur Max Num	New Max Num
Robot:	2	4	-> 4
Tray:	1	2	-> 3
Sensor:	1	4	-> 4
Conveyor:	1	4	-> 6
Conveyor Station:	2	8	-> 8
Fix Station:	2	8	-> 8

		OK	CANCEL

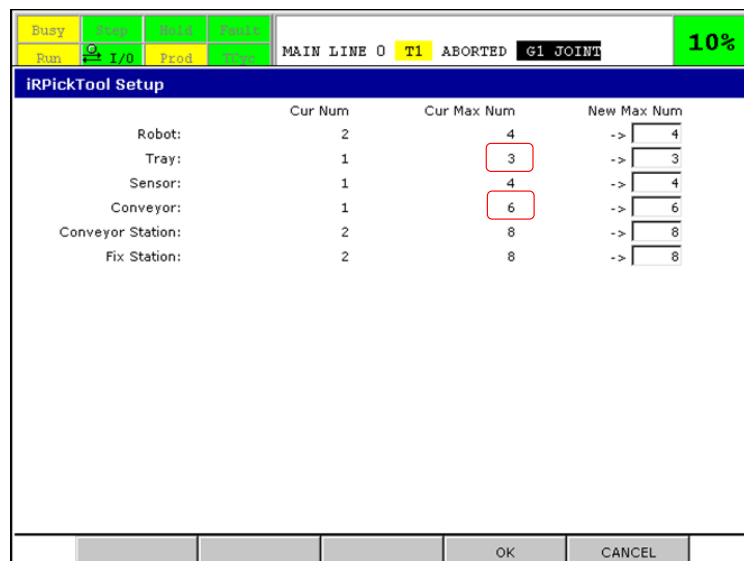
4. SETUP OF THE BASIC FUNCTIONS

B-83604EN/04

- 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

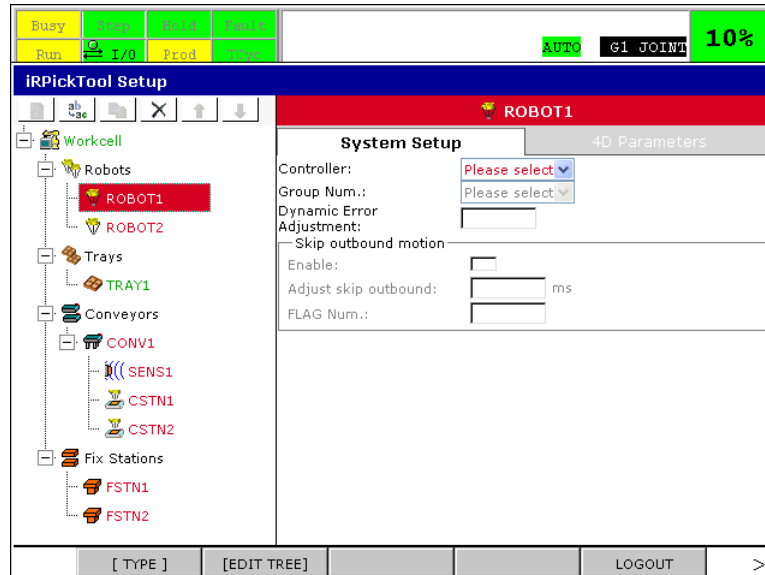
The maximum number that can be modified for each object shown Table 4.1.

Table 4.1 : 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

4.2 SETUP OF A ROBOT

Set up a robot.



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, see Section 4.12, "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 "4.12.3 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.

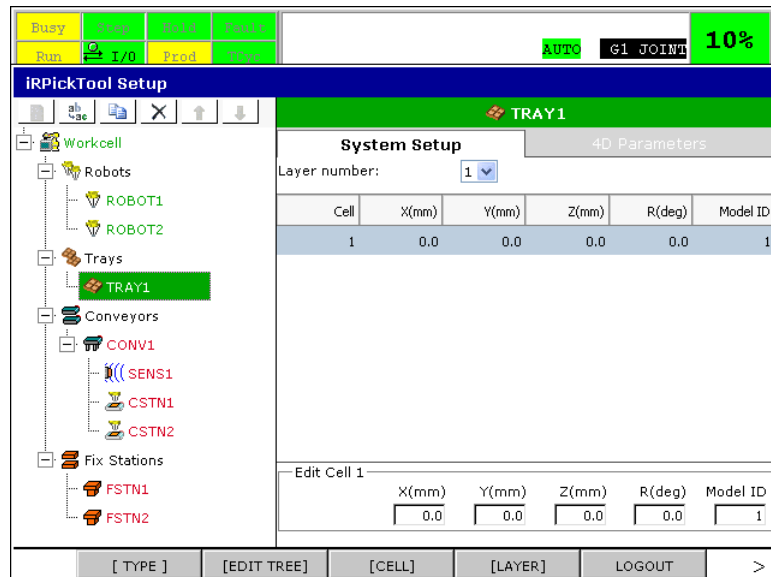
`$SLTK_GRP[number of servo conveyor group].$SRVO_DELAY`

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, see Section 7.4, "SKIPPING THE TRACKING MOTION OUTSIDE THE AREA".

4.3 SETUP OF A TRAY

Set up a tray.



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.

F3 [CELL] key

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.

X, Y, Z and R

Enter the coordinates of the highlighted cell in the tray frame.

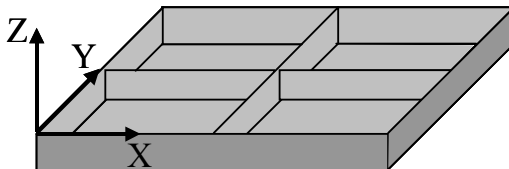
If the tray handles four part, set the coordinates as follows.

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 below.

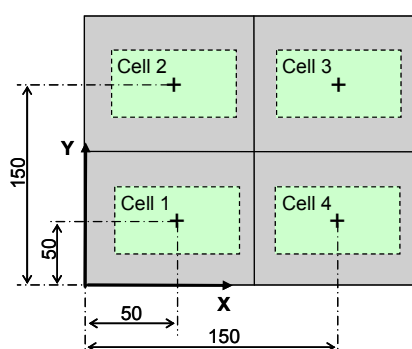
NOTE

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.



Add cells to a tray, and enter the coordinates of each cell. For example, the tray shown in the figure above is as shown in the figure below when viewed from above.



The cross mark (+) represents the center position of each individual cell. The coordinates of these cells are as follows.

Busy Stop Hold Fault
Run I/O Prod Stop

AUTO G1 JOINT 10%

IRPickTool Setup

Workcell

- Robots
 - ROBOT1
 - ROBOT2
- Trays
 - TRAY1
- Conveyors
 - CONV1
- Sensors
 - SENS1
- Fix Stations
 - CSTN1
 - CSTN2
 - FSTN1
 - FSTN2

TRAY1 System Setup

Layer number: 1

Cell	X(mm)	Y(mm)	Z(mm)	R(deg)	Model ID
1	50.0	50.0	-50.0	0.0	1
2	50.0	150.0	-50.0	0.0	1
3	150.0	150.0	-50.0	0.0	1
4	150.0	50.0	-50.0	0.0	1

Edit Cell 4

X(mm)	Y(mm)	Z(mm)	R(deg)	Model ID
150.0	50.0	-50.0	0.0	1

[TYPE] [EDIT TREE] [CELL] [LAYER] LOGOUT >

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 5, "VARIATIONS".

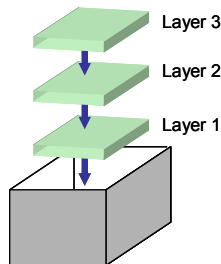
F4 [LAYER] key

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.

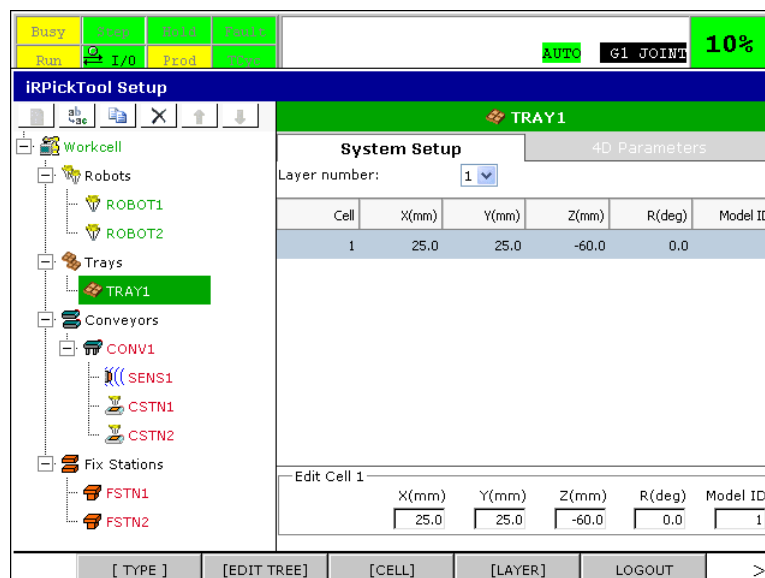
Multiple layers are used when control needs to be exerted over the order in which parts are placed in the tray - e.g., when the tray has multiple layers and a part cannot be placed in an upper layer unless the placement of parts in the lower layer is completed. When placing parts in a tray, set [Initial 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.



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.

iRPickTool Setup

Number of cells: 1

Default Z(mm): -50.0

OK CANCEL

- 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.

iRPickTool Setup

System Setup 4D Parameters

Layer number: 2

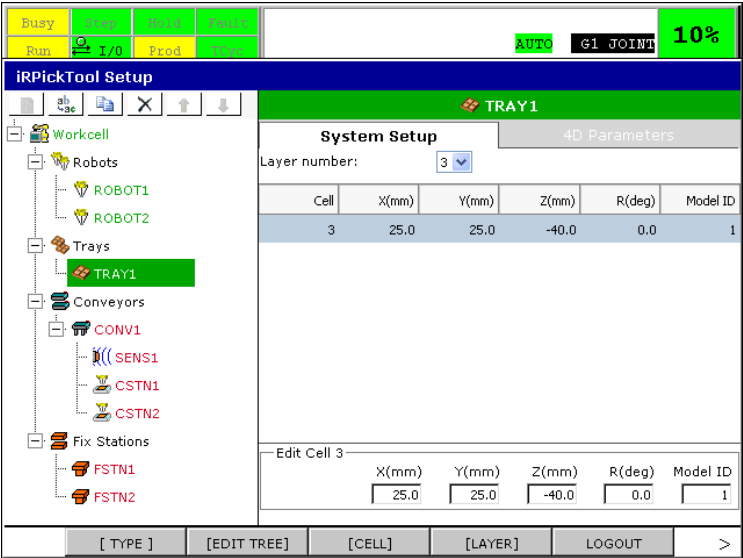
Cell	X(mm)	Y(mm)	Z(mm)	R(deg)	Model ID
2	25.0	25.0	-50.0	0.0	1

Edit Cell 2

X(mm)	Y(mm)	Z(mm)	R(deg)	Model ID
25.0	25.0	-50.0	0.0	1

[TYPE] [EDIT TREE] [CELL] [LAYER] LOGOUT >

- 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.

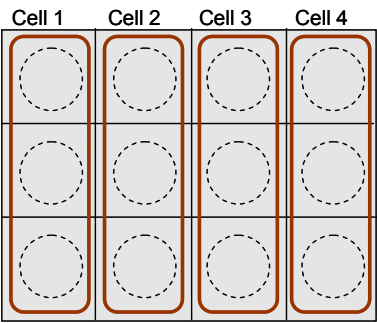
Layer number

When multiple layers are set, the layer number is used to switch the layer whose cell list is displayed.

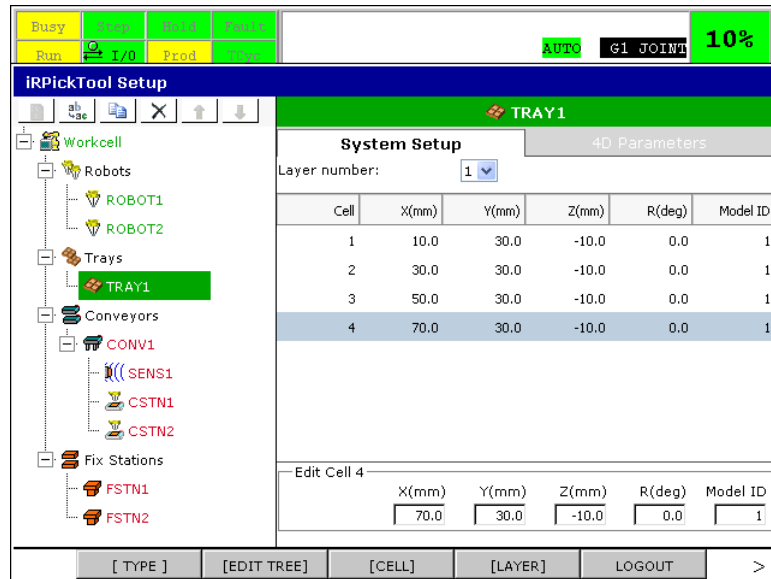
Setting of a tray when a robot hand that can hold multiple parts is used

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.



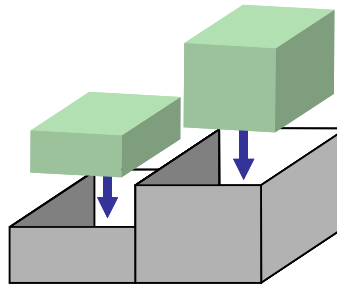
The robot handles three parts at a time. In such a case, three parts that can be handled at a time are considered to be a cell. Therefore, the tray has four cells. For example, the tray may be set as follows.



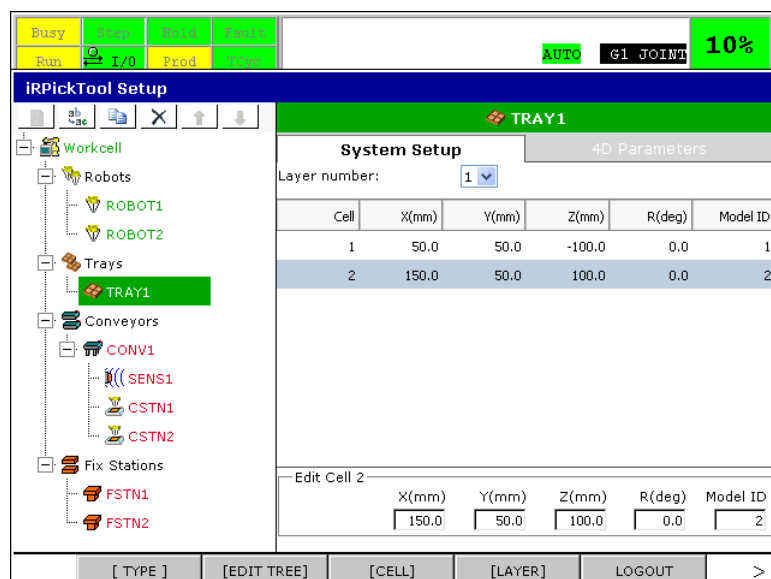
Setting of a tray when parts are different in height

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.

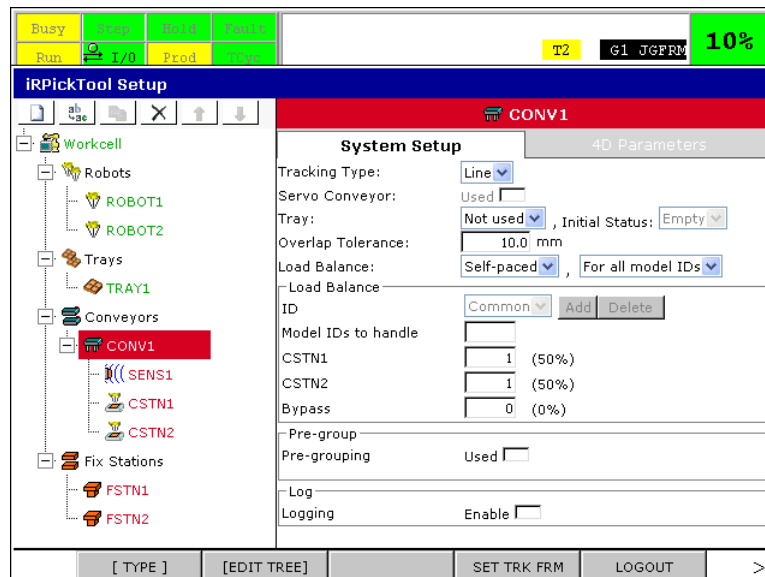


In this case, change the Z value of each cell according to the height of the part. For example, the tray may be set as shown below. Here, since the model of each part is different, different model IDs are set.



4.4 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 4.5, "SETUP OF A SENSOR" and Section 4.6, "SETUP OF A CONVEYOR STATION".



Tracking Type

Select the line tracking or the circular tracking. If you want to use the line tracking, select "Line". Otherwise, select "Circle". If you change it, child objects like the sensor and the conveyor station will be initialized.

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 Appendix D, "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

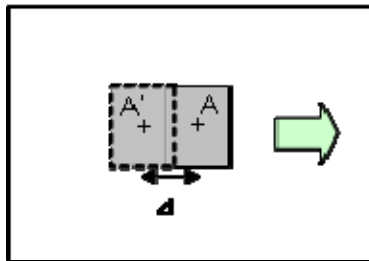
When placing parts into empty trays carried on the conveyor, select [Empty]. When picking up parts from filled trays carried on the conveyor, select [Full].

Overlap 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 any overlap is found, the vision system retains only the part detected earlier. Normally, set this value to 10 mm or so.

In the example shown in the figure below, if Δ is less than the overlap tolerance, the vision system retains only the n-1th part detected, A'.



A': Current position of the n-1th workpiece detected

A : Current position of the nth workpiece detected

Δ : Distance between A and A'



CAUTION

The overlap check may result in the vision system leaving out parts other than the same part detected multiple times as well. If the distance between the current position of the earlier detected part and that of the subsequently detected part is less than the tolerant, the vision system leaves out the subsequently detected part even when it has a different model ID.

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 5, "VARIATIONS".

[Load Balance] box

The [Load Balance] box refers to the portion of the screen that is shown in the figure below.

Load Balance - Common		Load Balance - 1	
ID	Common	ID	1
Model IDs to handle	1 (50%)	Model IDs to handle	1 (50%)
CSTN1	1 (50%)	CSTN1	1 (50%)
CSTN2	1 (50%)	CSTN2	1 (50%)
Bypass	0 (0%)	Bypass	0 (0%)

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 left side of the figure. 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 right side of the figure. 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 4.14, "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 4.14, "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.

Log	
Logging	Enable ☒
Output Device	UD1:\ Output
Output Controller	This Controller

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.

4.4.1 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

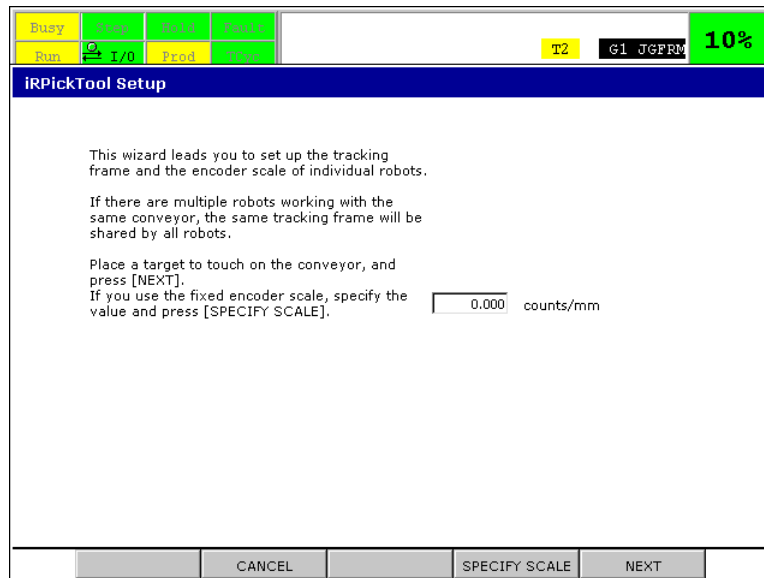
- 1 If you change the tracking frame, you need to perform conveyor station setup, camera calibration, reference position setting, and robot position teaching again.
- 2 When there is more than one robot, you cannot set a tracking frame unless the encoder count values of all the controllers on the conveyor 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 dependind on tracking type.

4.4.1.1 Line Conveyor

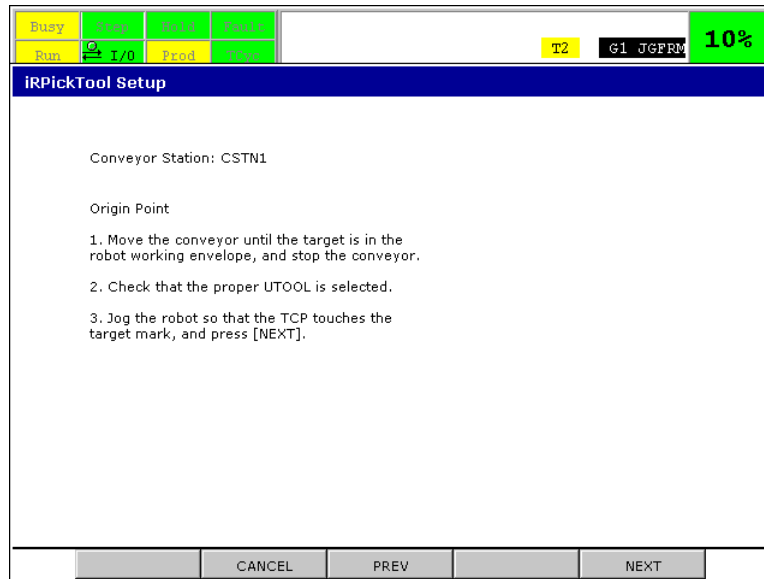
When tracking type is “Line”, please see this procedure. When tracking type is “Circle”, please see the 4.4.1.2, “Circular Conveyor”.

- 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 4.6, "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 *iRVision*, you can touch up the Y-axis direction point easier with turning the direction of the big circle of the calibration grid to the conveyor moving direction or the direction perpendicular to it



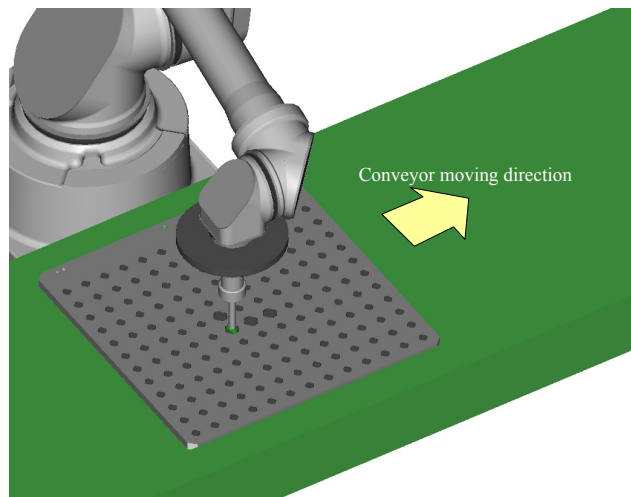
- 4 As the fixture feed frame, select the user tool frame that you set in Section 3.5, "SETTING OF THE TCP".



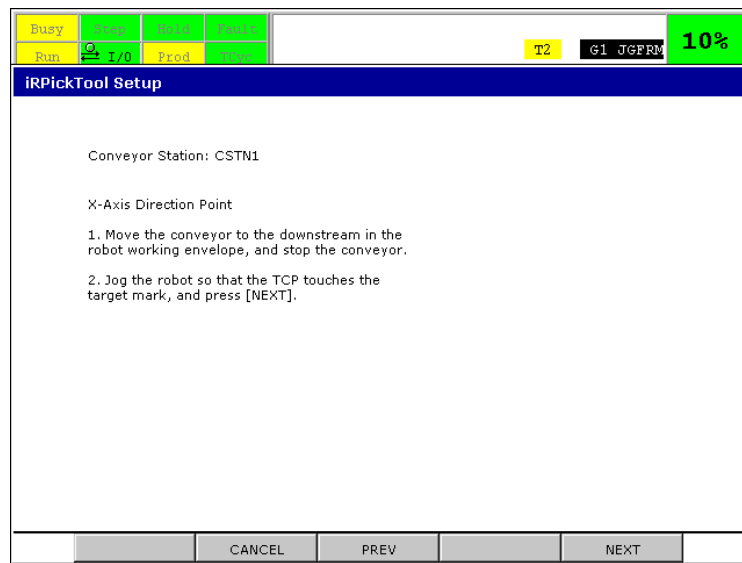
CAUTION

Select the user tool frame with the robot that touches up the fixture. It is not necessarily the robot controller for which the *iRPickTool* setup screen is open.

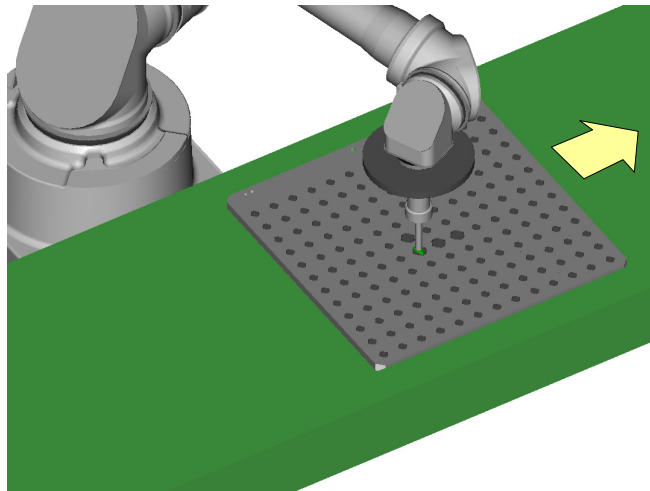
- 5 Move the conveyor until the fixture reaches the upstream area within the robot operation range, and the stop it. Jog the robot to touch up the fixture with the TCP. Here, the robot touches up the dot located at the intersection of large dots arranged in the shape of the letter L on the calibration grid.



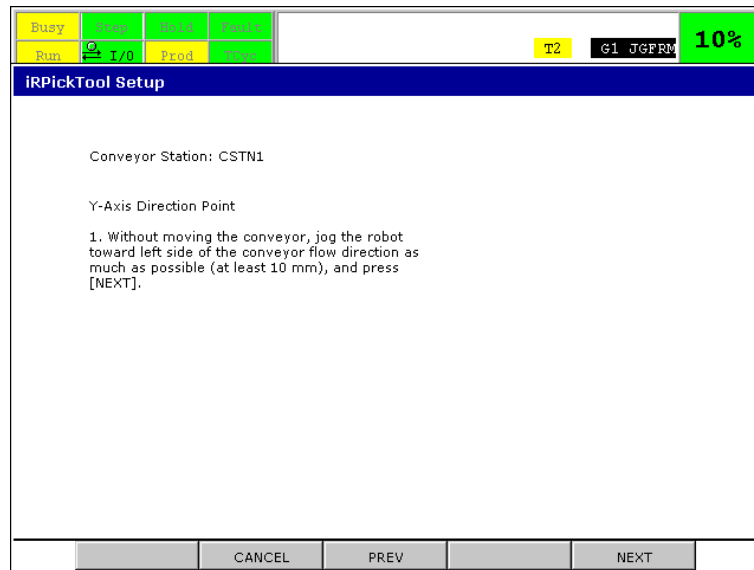
With the fixture touched up, press F5 [NEXT] key. A screen as shown below is displayed.



- 6 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 within the robot operation range and then stop it. Then, jog the robot again to touch up the same part of the fixture that you touched up with the TCP earlier.



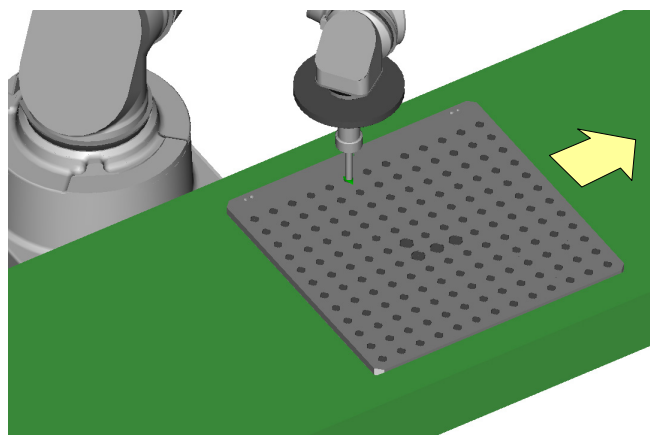
With the fixture touched up, press F5 [NEXT] key. A screen as shown below is displayed.



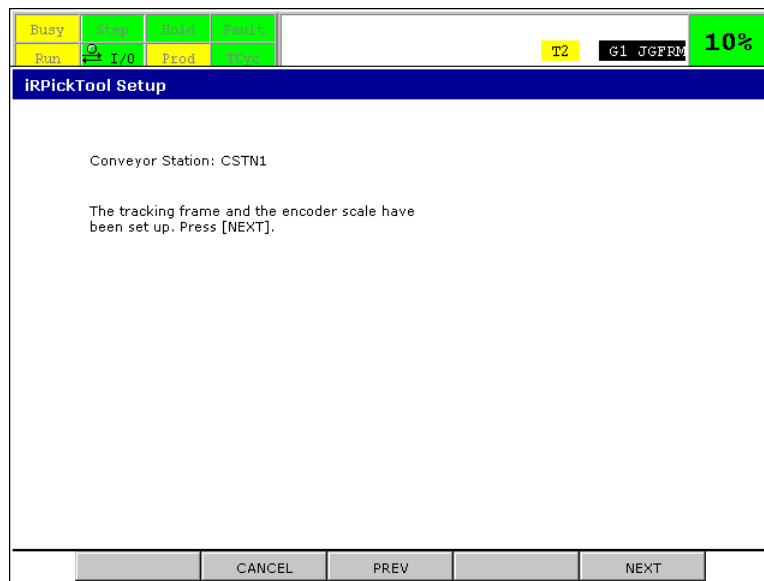
- 7 Jog the robot upward (or away from the conveyor surface) . Next jog the robot to the left as much as possible (at least 10 mm) relative to the conveyor moving direction without moving the conveyor. Then jog the robot downward (or toward the conveyor surface) to touch up a point of the fixture and press F5 [NEXT] key. When doing this, make sure that the TCP is accurately positioned on the plane surface of the fixture.

CAUTION

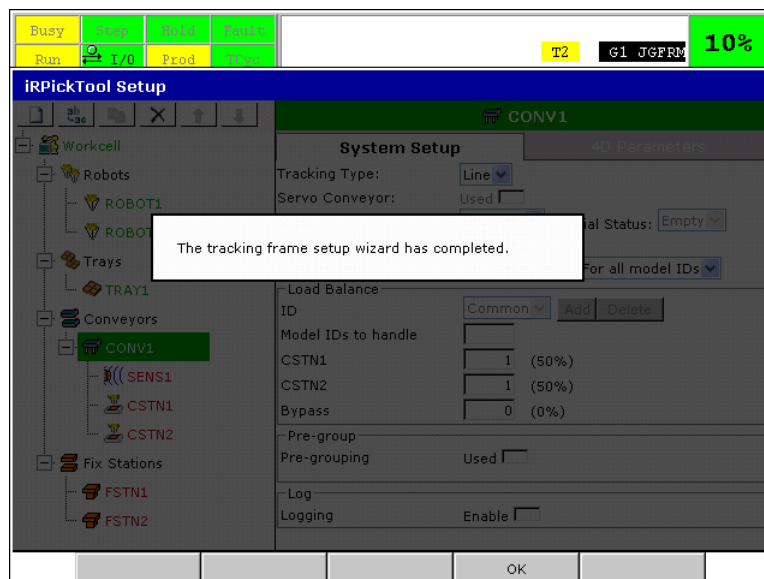
- 1 In the case of a 3-, 4- or 5-axis robot, the Y-axis direction point is used to determine whether the Z-axis direction of the tracking frame is upward or downward. Also, In the case of a 4- or 5-axis robot, since the plane parallel to the XY plane of the world frame that includes the first touched up point becomes the XY plane of the tracking frame, the world frame of the robot and the conveyor surface need to be parallel to each other. In the case of a 3-axis robot, since W, P of the tracking frame are considered, there is no problem even if the conveyor tilts. However, if you touch up a point distant from the fixture as the Y-axis direction point, the tilted frame is set. You must take care so that you touch up a point on the fixture.
- 2 If the conveyor is installed vertically, as in the case where parts are hung on the hooks of the conveyor, set the Y-axis direction point by moving the robot down, not to the left, relative to the moving direction.



A screen as shown below is displayed.



- 8 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. Repeat steps 4 through 7 for each robot. When you have completed the procedure for all robots, a screen as shown below is displayed.



- 9 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.

4.4.1.2 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 4.6, "SETUP OF A CONVEYOR STATION".

- 2 In the conveyor setup screen, press F4 [SET TRK FRM] key. A screen as shown below is displayed.

Busy	Stop	Hold	Enable	A_M LINE 0 T2 ABORTED JGFRM			10%
Run	I/O	Prod	TCyc				

iRPickTool Setup

This wizard leads you to set up the tracking frame and the encoder scale of individual robots.

If there are multiple robots working with the same conveyor, the same tracking frame will be shared by all robots.

Place a target to touch on the conveyor, and press [NEXT].
 If you use the fixed encoder scale, specify the value and press [SPECIFY SCALE].

counts/deg

CANCEL
SPECIFY SCALE
NEXT

- 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 *iR*Vision, 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.

Busy	Stop	Hold	Enable	T1 G1 JOINT			10%
Run	I/O	Prod	TCyc				

iRPickTool Setup

Conveyor Station: CSTN1

X-Axis Direction Point

1. Move the conveyor until the target is in the robot working envelope, and stop the conveyor.
2. Check that the proper UTOOL is selected.
3. Jog the robot so that the TCP touches the target mark, and press [NEXT].

CANCEL
PREV
NEXT



CAUTION

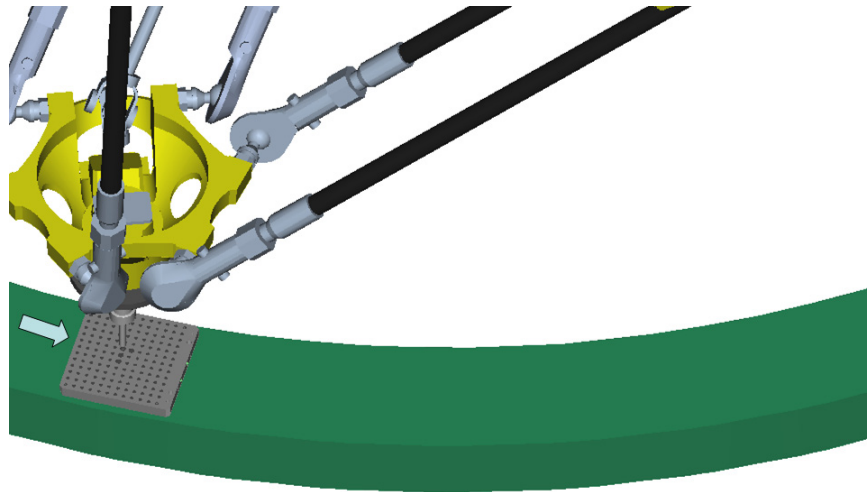
Be sure to touch up as much as possible the outer circumference. This is necessary to improve the tracking accuracy.

- 4 As the fixture feed frame, select the user tool frame that you set in Section 3.5, "SETTING OF THE TCP".

**CAUTION**

Select the user tool frame with the robot that touches up the fixture. It is not necessarily the robot controller for which the *iRPickTool* 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 stop it. Jog the robot to touch up the fixture with the TCP. Here, the robot touches up the dot located at the intersection of large dots arranged in the shape of the letter L on the calibration grid.



With the fixture touched up, press F5 [NEXT] key. A screen as shown below is displayed.

Busy	Stop	Reset	Pause			T1	G1 JOINT	10%
Run	I/O	Prod	Stop					

iRPickTool Setup

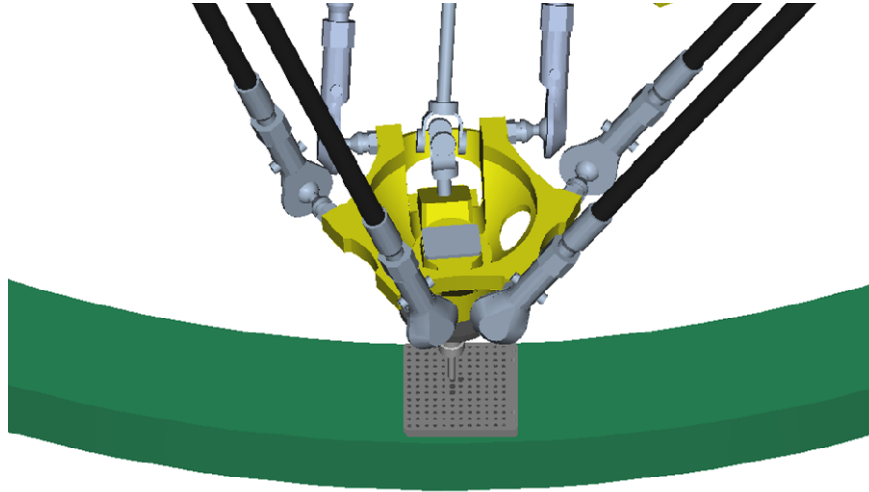
Conveyor Station: CSTN1

Y-Axis Direction Point

1. Move the conveyor to the middle in the robot working envelope, and stop the conveyor.
2. Jog the robot so that the TCP touches the target mark, and press [NEXT].

CANCEL
PREV
NEXT

- 6 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. Then, jog the robot again to touch up the same part of the fixture that you touched up with the TCP earlier.



With the fixture touched up, press F5 [NEXT] key. A screen as shown below is displayed.

Busy	Stop	Hold	Ready			T1	G1 JOIN	10%
Run	I/O	Prod	Down					

iRPickTool Setup

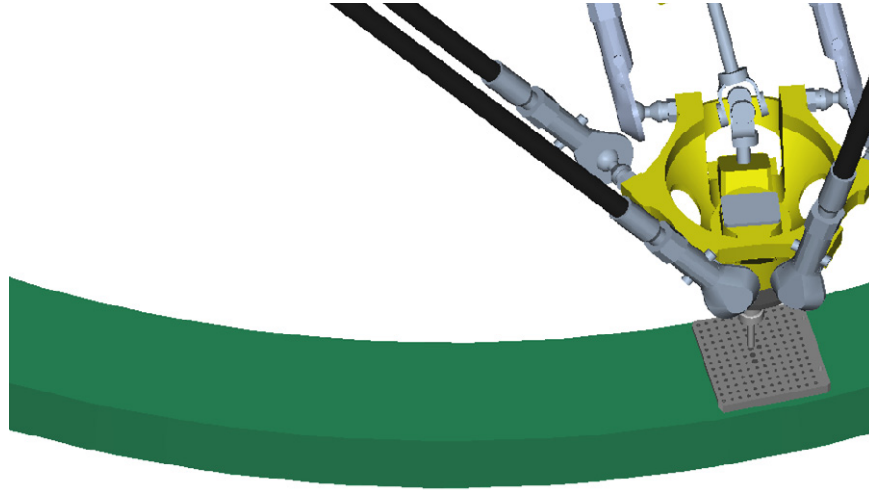
Conveyor Station: CSTN1

Assistant Point

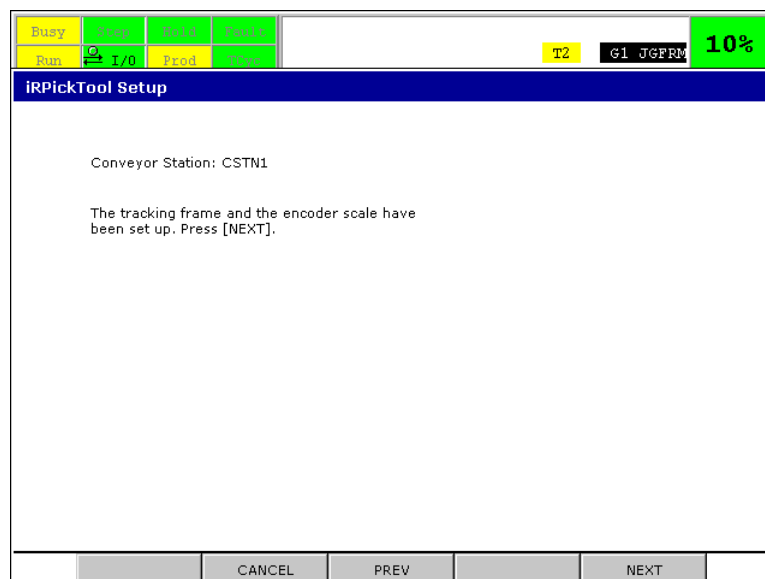
1. Move the conveyor to the downstream in the robot working envelope, and stop the conveyor.
2. Jog the robot so that the TCP touches the target mark, and press [NEXT].

	CANCEL	PREV		NEXT	

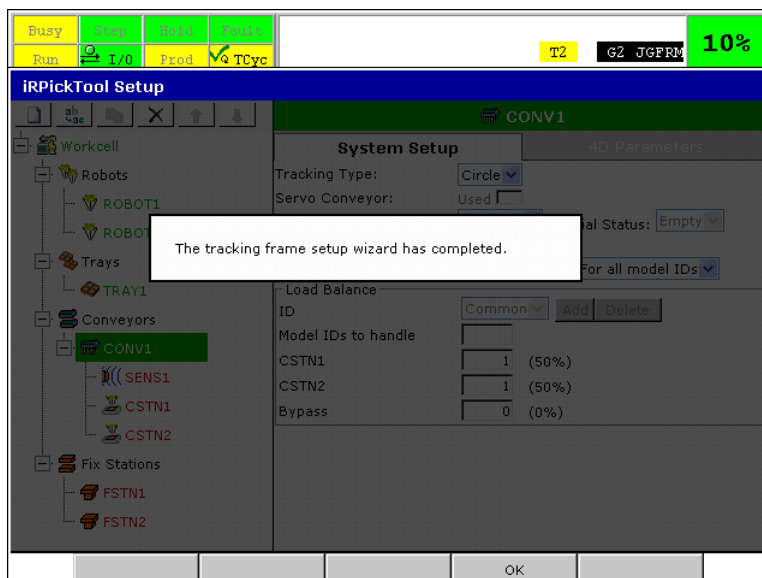
- 7 After moving the robot to a position where the TCP does not interfere when the conveyor is moved, move the conveyor so that the fixture comes to the downstream area as much as possible within the robot operation range and then stop it. Then, jog the robot again to touch up the same part of the fixture that you touched up with the TCP earlier.



With the fixture touched up, press F5 [NEXT] key. A screen as shown below is displayed.



- 8 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. Repeat steps 4 through 7 for each robot. When you have completed the procedure for all robots, a screen as shown below is displayed.



- 9 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.

4.5 SETUP OF A SENSOR

Set up a sensor.

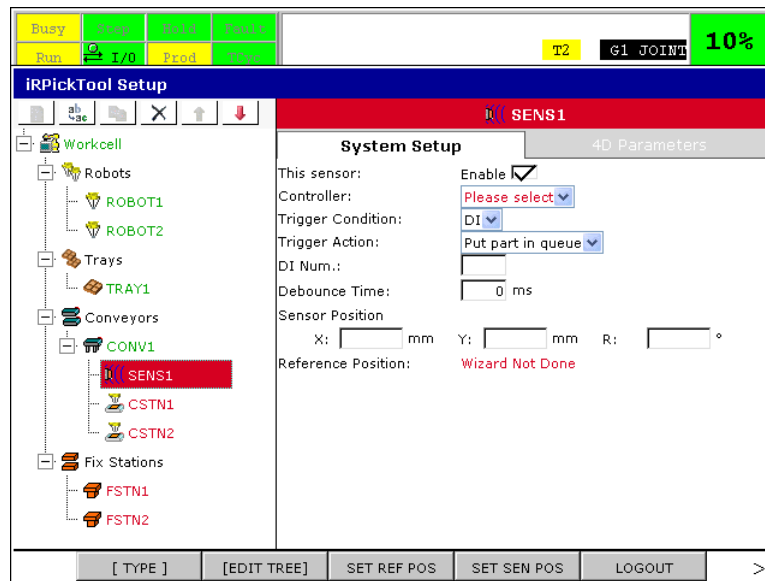


CAUTION

- 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 ("4.10 REFERENCE POSITION SETTING").

4.5.1 Setting a Sensor in Detail

If you select a sensor on the tree, a screen as shown below is displayed.



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.

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.

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.

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].

NOTE

When the tracking type is "Circle", [Find part by vision] can be used in 7DC3/23(V8.30P/23) or later.

Find part by multiview

The vision system searches for a part twice and puts a detected part into the queue. 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.

The possible combinations of a trigger condition and an action are shown below.

	Distance	DI	HDI	FLAG
Put part in queue	O	O	O	O
Find part by vision	O	O	X	X
Find part by multiview	X	O	X	X

Depending on the combination selected, the appearance of the setup screen changes as described below.

NOTE

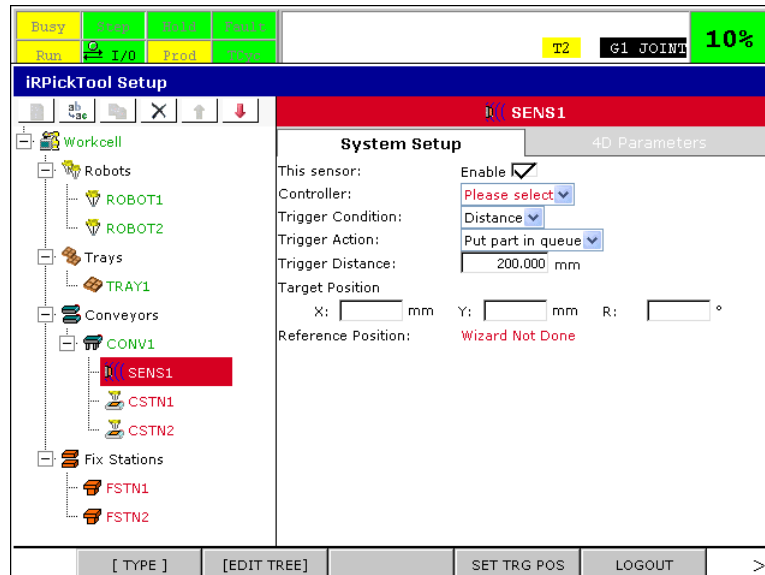
For details of a combination of [DI] + [Find part by multiview], see Section 5.11, "MULTI-VIEW".

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

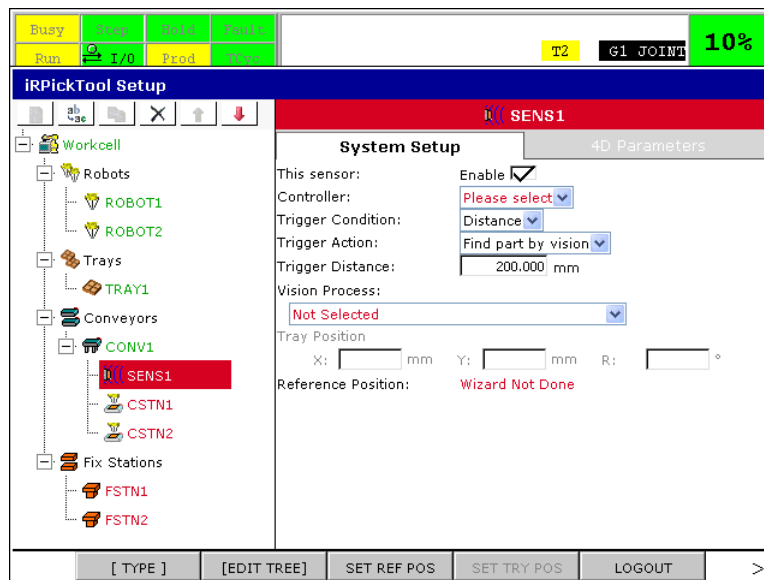
This is the part position that is generated each time the conveyor travels the trigger distance. Set these values using the target position setting wizard described later.

Reference Position

When you finish the wizard to set the target position, “Wizard Done” is displayed in green.

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.



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 7.2.2 “Checking the Status of a Sensor”.

Vision Process

Enter the name of the vision process to be executed.

Tray Position

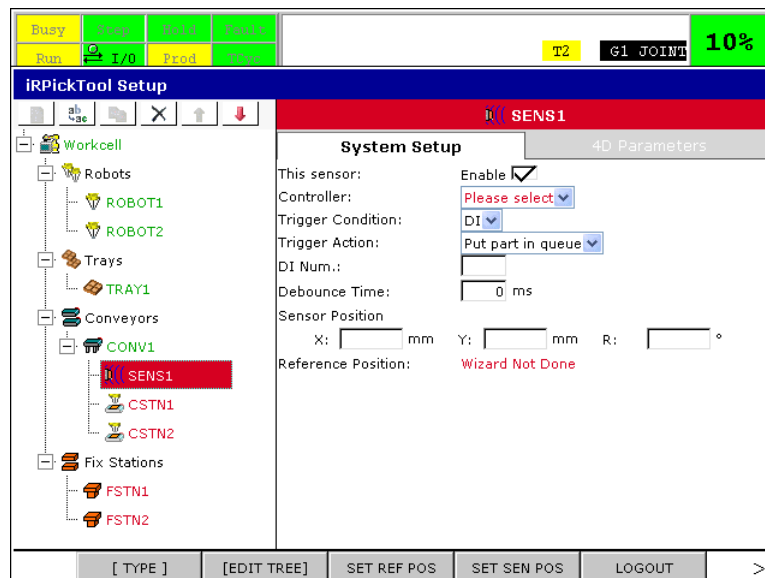
This is the relative position of the tray detection position and the origin of the tray. Set these values using the tray position setting wizard described later.

Reference Position

When you finish the wizard to set the reference position, “Wizard Done” is displayed in green.

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

While you can also set [Trigger INPUT Number] using the tracking schedule setting screen of the teach pendant of the robot, use only the sensor task setup screen to set the DI number.

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

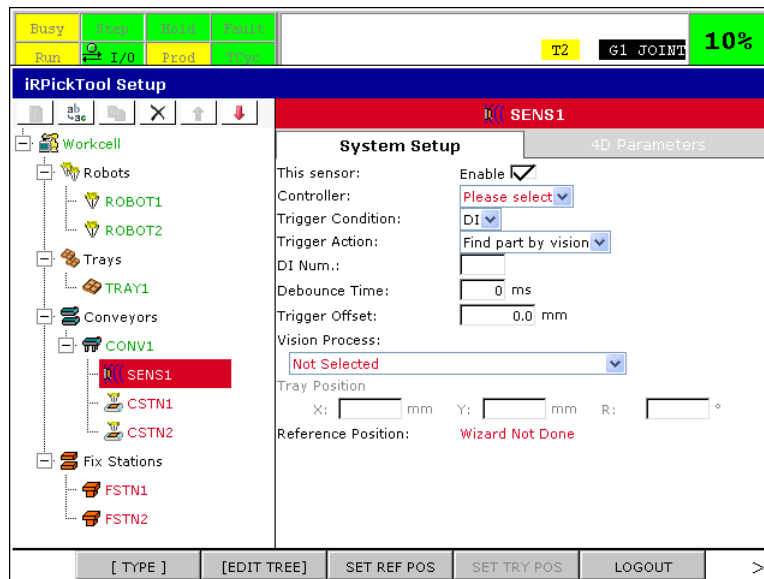
When a part is detected, the position specified here is used as the detection position as the part is put into the queue. Set these values using the sensor position setting wizard described later.

Reference Position

When you finish the wizard to set the reference position, “Wizard Done” is displayed in green.

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

While you can also set [Trigger INPUT Number] using the tracking schedule setting screen of the teach pendant of the robot, use only the sensor task setup screen to set the DI number.

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.

Vision Process

Specify the name of the vision process to be executed.

Tray Position

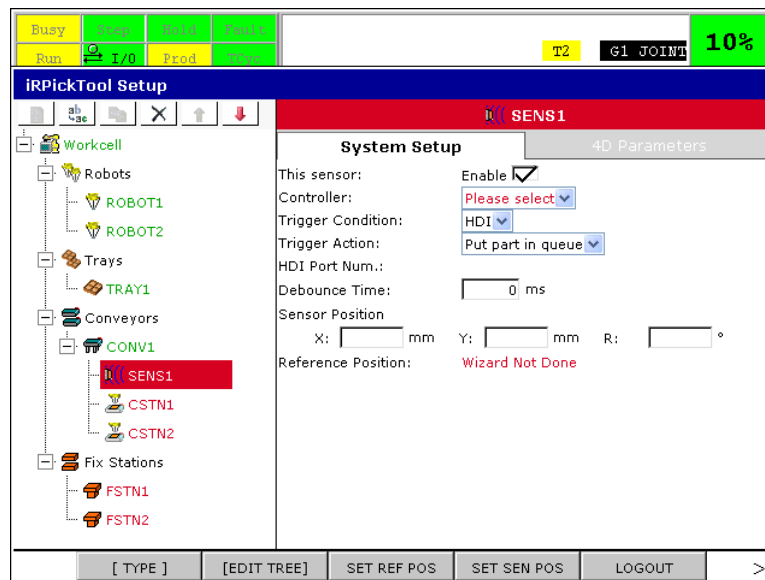
This is the relative position of the tray detection position and the origin of the tray. Set these values using the tray position setting wizard described later.

Reference Position

When you finish the wizard to set the reference position, "Wizard Done" is displayed in green.

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 wizard described later.

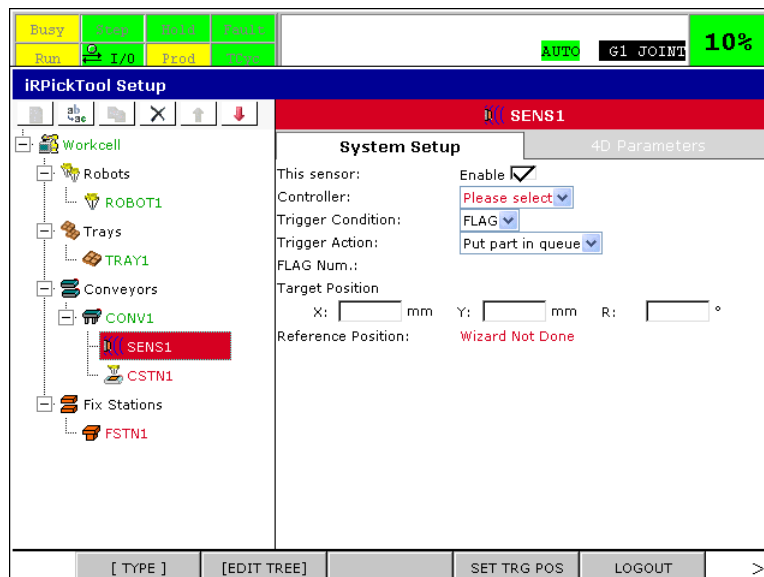
Reference Position

When you finish the wizard to set the reference position, "Wizard Done" is displayed in green.

When [FLAG] and [Put part in queue] are selected

This is used only for the servo conveyor.

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.

iRPickTool 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 wizard described later.

Reference Position

When you finish the wizard to set the target position, "Wizard Done" is displayed in green.

4.5.2 Settings Required for Each Combination of a Trigger Condition and a Trigger Action

The sensor setup procedures described in Subsection 4.5.3, "Setting the Sensor Position" through Subsection 4.5.6, "Setting the Target Position" involve operating robots and the vision system. The required settings differ depending on the combination of a trigger condition and an action. See the table below to find out which settings you need to make.

	4.5.3 Sensor Position	4.5.4 Vision System	4.5.5 Tray Position	4.5.6 Target Position
[Distance]+[Put part in queue]				
Use Tray	x	x	x	○
Not Use Tray	x	x	x	○
[Distance]+[Find part by vision]				
Use Tray	x	○	○	x
Not Use Tray	x	○	x	x
[DI]+[Put part in queue]				
Use Tray	○	x	x	x
Not Use Tray	○	x	x	x
[DI]+[Find part by vision]				
Use Tray	x	○	○	x
Not Use Tray	x	○	x	x
[HDI]+[Put part in queue]				
Use Tray	○	x	x	x
Not Use Tray	○	x	x	x
[FLAG]+[Put part in queue]				
Use Tray	x	x	x	○
Not Use Tray	x	x	x	○

4.5.3 Setting the Sensor Position

Teach the position where the photo eye is installed. This setting is required when [DI] or [HDI] is selected for [Trigger Condition] and [Put part in queue] is selected for [Trigger Action]. When a part passes in front of the photo eye, this position is put into the queue. The procedure slightly differs depending on whether a tray is used on the conveyor or not.

4.5.3.1 When a tray is not used

- 1 Stop the conveyor.
- 2 Press F4 [SET SEN POS] key. A screen as shown below is displayed.

Busy Stop Hold Fault

Run I/O Hold

T2 G1 JGFRM 10%

iRPickTool Setup

Sensor Position Wizard Step : 1/2

This wizard guides you so that you can set up sensor position.

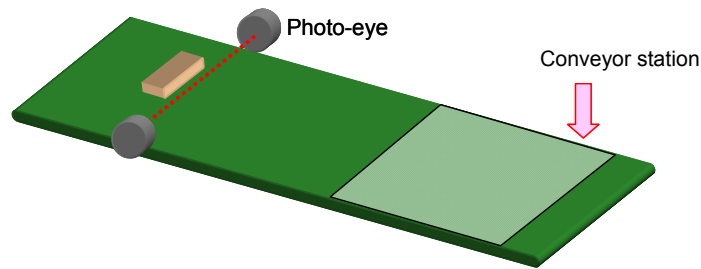
Please put the target on the conveyor upstream of sensor.
Then jog the conveyor so that the target comes in front of this robot.

When the target is detected successfully, [NEXT] is enabled.

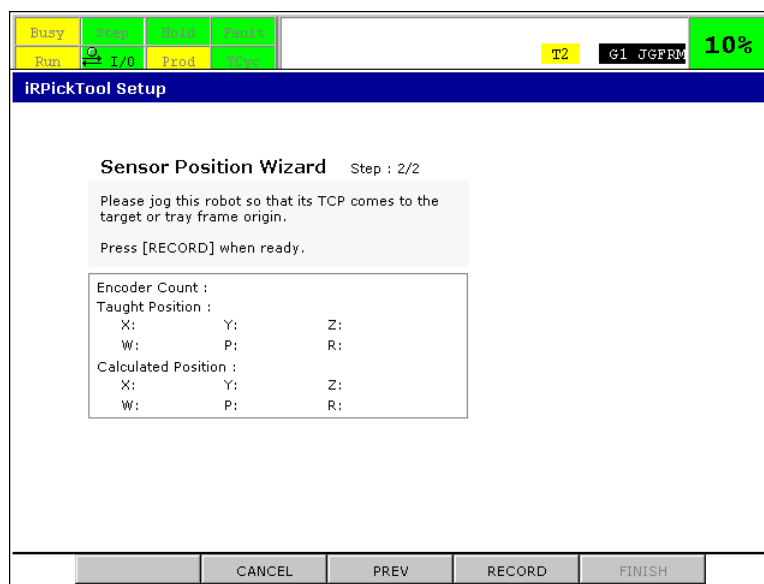
Trigger Count :

CANCEL NEXT

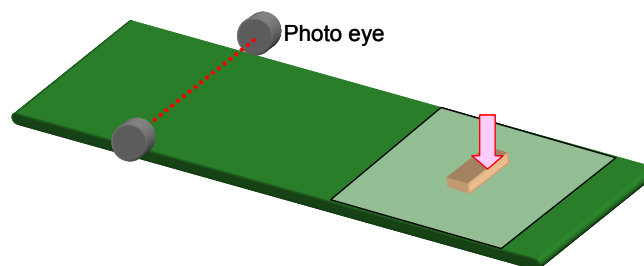
- 3 Place the part upstream of the photo eye, and move the conveyor to let the part pass in front of the DI sensor.



- 4 When the sensor detects the part, the encoder value obtained at the time of detection is displayed and F5 [NEXT] key is enabled.
- 5 If you press F5 [NEXT] key, a screen as shown below is displayed.



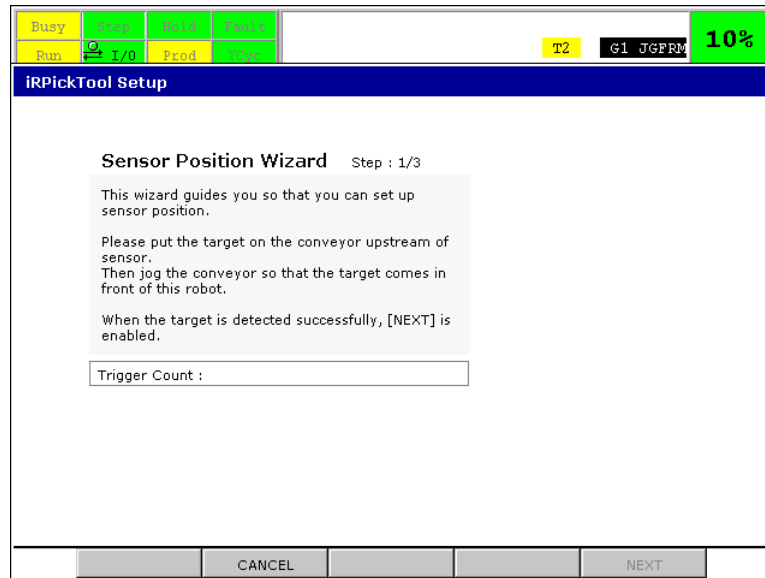
- 6 Move the conveyor until the part comes in front of the robot.
- 7 Jog the robot, and touch up the downstream-side end of the part (the part detected by the photo eye) with the TCP.



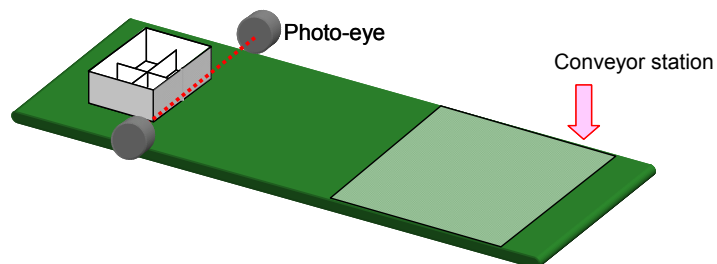
- 8 Press F4 [RECORD] key. The encoder value obtained at the time of touch-up, the touch-up position, and the calculated position are displayed, and F5 [FINISH] key is enabled.
- 9 If you press F5 [FINISH] key, the initial setup screen is displayed again. The calculated value is input for X of the sensor position (the values of Y and R remain unchanged from 0).

4.5.3.2 When a tray is used

- 1 Stop the conveyor.
- 2 Press F4 [SET SEN POS] key. A screen as shown below is displayed.



- 3 Place a tray upstream of the photo eye, and move the conveyor so that the tray passes in front of the photo eye.



- 4 When the sensor detects the tray, the encoder value obtained at the time of detection is displayed and F5 [NEXT] key is enabled.
- 5 If you press F5 [NEXT] key, a screen as shown below is displayed.

Busy	Stop	Hold	Fault			T2	G1 JGPRM	10%
Run	I/O	Prod	Stop					

IRPickTool Setup

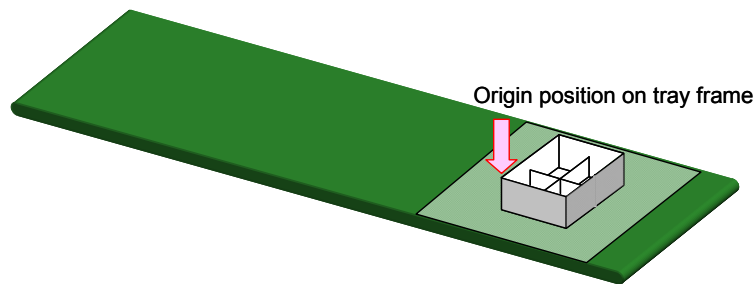
Sensor Position Wizard Step : 2/3

Please jog this robot so that its TCP comes to the target or tray frame origin.
Press [RECORD] when ready.

Encoder Count :		
Taught Position :		
X:	Y:	Z:
W:	P:	R:

	CANCEL	PREV	RECORD	NEXT	

- 6 Move the conveyor until the tray comes in front of the robot.
- 7 Jog the robot, and touch up the origin of the tray frame.



- 8 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.
- 9 If you press F5 [NEXT] key, a screen as shown below is displayed.

Busy	Stop	Hold	Fault			T2	G1 JGPRM	10%
Run	I/O	Prod	Stop					

IRPickTool Setup

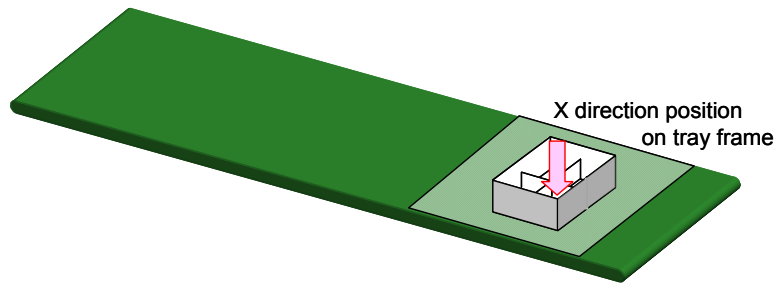
Sensor Position Wizard Step : 3/3

Without moving the conveyor, please jog this robot so that its TCP comes to the tray frame x point.
Press [RECORD] when ready.

Taught Position :		
X:	Y:	Z:
W:	P:	R:
Calculated Position :		
X:	Y:	Z:
W:	P:	R:

	CANCEL	PREV	RECORD	FINISH	

- 10 Jog the robot, and touch up the X direction point of the tray frame. Press F4 [RECORD] key. The touch-up position and the calculated position are displayed, and F5 [FINISH] key is enabled.



- 11 If you press F5 [FINISH] key, the initial setup screen is displayed again. Values are input for X, Y, and R of the sensor position.

4.5.4 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].



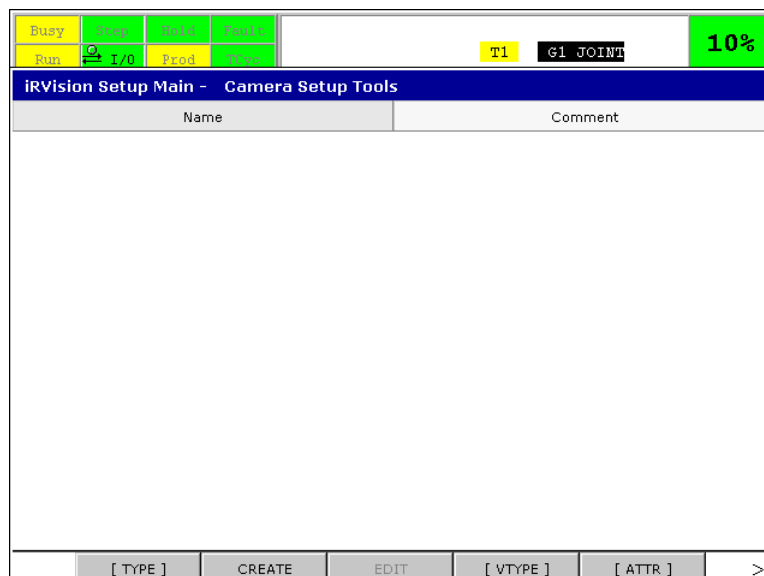
CAUTION

To set up the vision system, you need to use a controller to which a camera is connected.

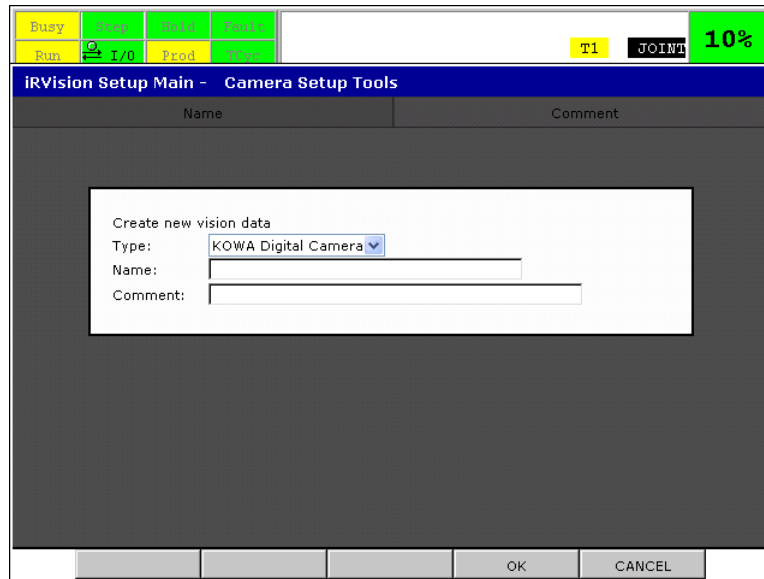
4.5.4.1 Setting up the camera

Set up the camera

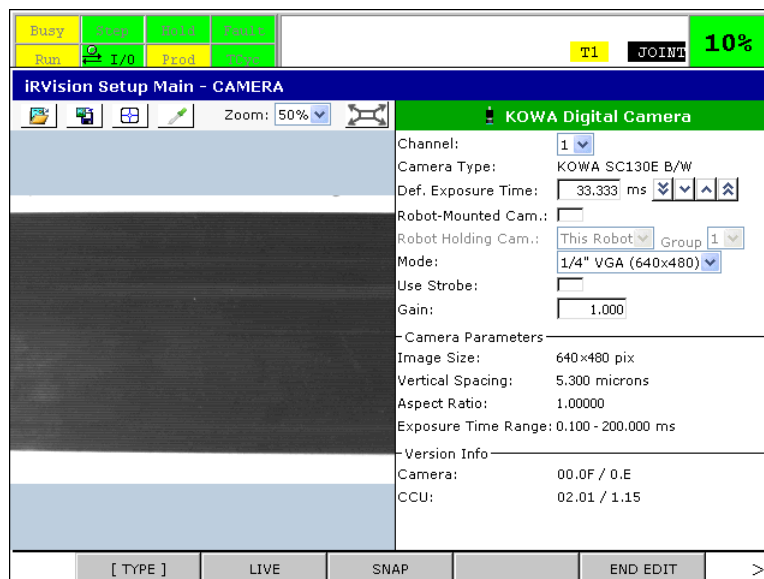
- 1 From the [MENU] key, select [8 iRVision] and then [1 Vision setup] to display the vision data list screen. Press F4 [VTYPE] key, and select [Camera Setup Tools] from the list. A screen as shown below is opened.



- 2 Press F2 [CREATE] key. A screen as shown below is displayed.



- 3 In [Type], select the type of the camera. Regardless of monochrome or color, when using a digital camera, select [KOWA Digital Camera].
- 4 In [Name], enter the name of the camera (e.g., CAMERA).
The camera name may consist of up to 34 alphanumeric characters and must not begin with a numeric character.
- 5 Press F4 [OK] key. The created camera data is displayed in the list.
- 6 Select the created camera data, and press F3 [EDIT] key. A screen as shown below is displayed.



- 7 Press F3 [SNAP] key to check that a camera image can be snapped normally.
- 8 Place a part in the camera's field of view. Press F3 [SNAP] key again, and check the image. Make sure that the part is within the field of view and that the image is one that is viewed from above. If the part is not snapped normally, adjust the position of the camera.
- 9 Press F2 [LIVE] key to adjust the focus and diaphragm of the camera.
- 10 Press F5 [END EDIT] key to close the edit screen.

4.5.4.2 Calibrating the camera

Calibrate the camera.

The iRPickTool only supports vision tracking calibration.

By calibrating the camera, you can convert the position data on the detected image into position data on the tracking frame.

As you calibrate the camera, the camera frame is associated with the tracking frame.

Setting the calibration parameters

- 1 From the [MENU] key, select [8 iRVision] and then [1 Vision setup] to display the vision data list screen. Press F4 [VTYPE] key, and then select [Camera Calibration Tools]] from the list.
- 2 Press F2 [CREATE] key. A screen as shown below is displayed.

The screenshot shows the 'IRVision Setup Main - Camera Calibration Tools' window. At the top, there's a status bar with 'T2', 'G1 JOINT', and '10%'. Below it, the window title is 'IRVision Setup Main - Camera Calibration Tools'. The main area contains a 'Create new vision data' dialog box with the following fields: 'Type' (set to 'Camera Calib. For Vis. Track'), 'Name' (empty), and 'Comment' (empty). At the bottom, there are 'OK' and 'CANCEL' buttons.

- 3 In [Type], select [Camera Calib. For Vis. Track].
- 4 In [Name], enter the name of the calibration data (e.g., VTCALIB).
The name may consist of up to 34 alphanumeric characters and must not begin with a numeric character.
- 5 Press F4 [OK] key. The created calibration data is displayed in the list.
- 6 Select the created calibration data, and press F3 [EDIT] key. A screen as shown below is displayed.

The screenshot shows the 'IRVision Setup Main - VTCALIB' window. The title bar says 'IRVision Setup Main - VTCALIB'. The main area has a 'Setup' tab selected, with sub-tabs 'Data' and 'Points'. The 'Setup' tab contains the following fields: 'Camera' (Not Selected), 'Conveyor Name' (Not Selected), 'Exposure Time' (4.000 ms), 'Grid Spacing' (15.0 mm), 'Projection' (Perspective), and 'Override Focal Dist.' (Yes). There is a 'Calibration Wizard' button at the bottom. The status bar at the bottom shows '[TYPE]', 'LIVE', 'SNAP', 'END EDIT', and '>'. The top status bar shows 'T2', 'G1 JOINT', and '10%'.

- 7 Select [Camera].
- 8 Select [Conveyor Name].

- 9 Adjust the value in [Exposure Time].
- 10 In [Grid spacing], enter a desired value for the grid pattern.
- 11 In [Projection], select [Perspective].

**CAUTION**

[Orthogonal] can be selected only when the part to be detected is uniform in height and when the grid pattern surface is at the same height as the top surface of the part to be detected.

- 12 In [Override Focal Distance], select the calculation method. Select [Yes].

**CAUTION**

If you select [No] in [Override Focal Distance], the focal length may fail to be calculated correctly in principle because the calibration grid on the conveyor is almost parallel to the imaging area of the camera.

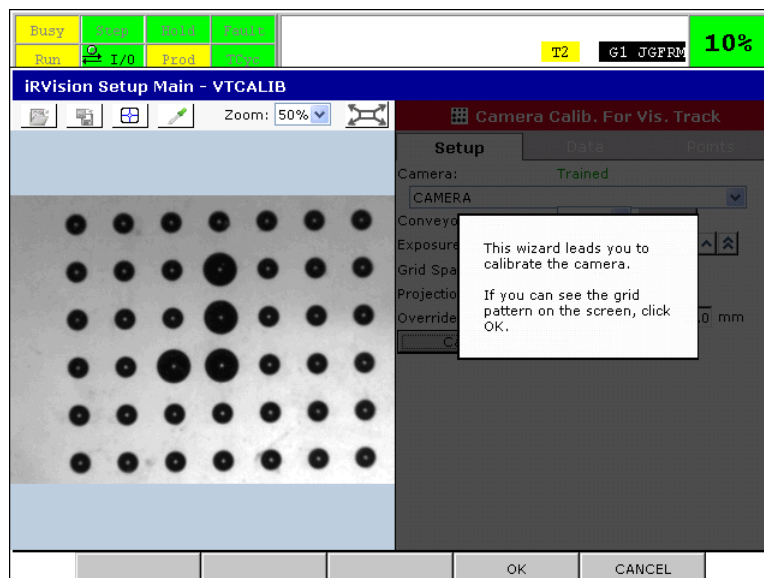
- 13 Set the nominal focal length of the lens.

**CAUTION**

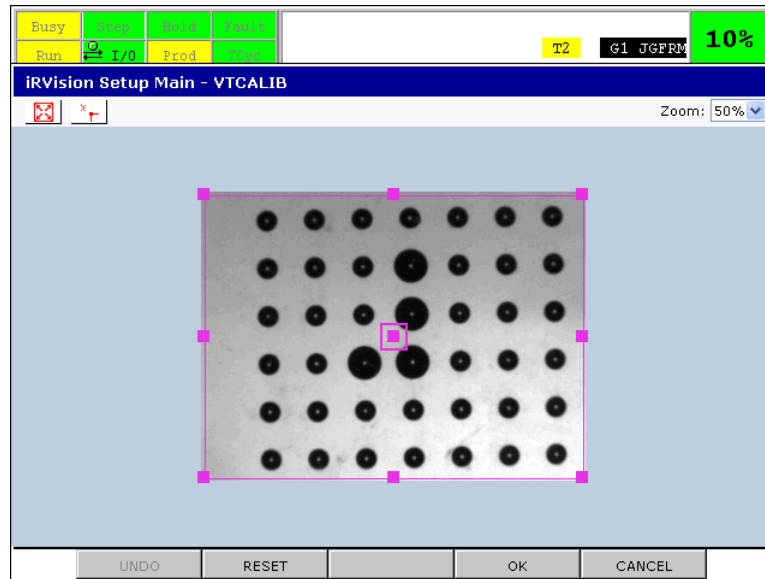
Check the focal length of the camera in use. If you set an incorrect focal length here, you cannot calibrate the camera properly.

Executing calibration

- 1 Stop the conveyor, and place the calibration grid in the field of view of the camera. Then, click the [Calibration Wizard] button. A screen as shown below is displayed.

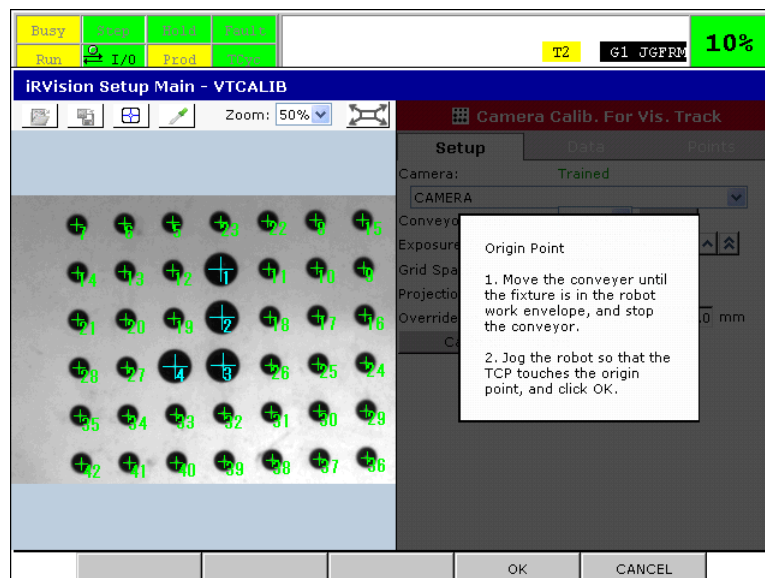


- 2 Check that the grid pattern of the calibration grid is displayed on the screen, and then press F4 [OK] key. A screen as shown below is displayed.

**NOTE**

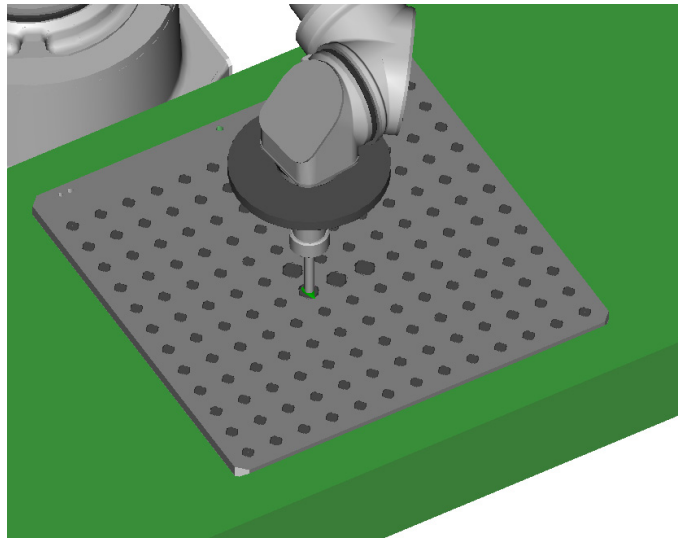
Use a calibration grid that is large enough to fill the entire field of view, and the lens distortion will be calculated more accurately. This leads to improved accuracy when you handle a part that appears in a corner of the camera's field of view.

- 3 Enclose the grid pattern with a magenta rectangular window, and press F4 [OK] key. When the grid pattern is detected, a screen as shown below is displayed.

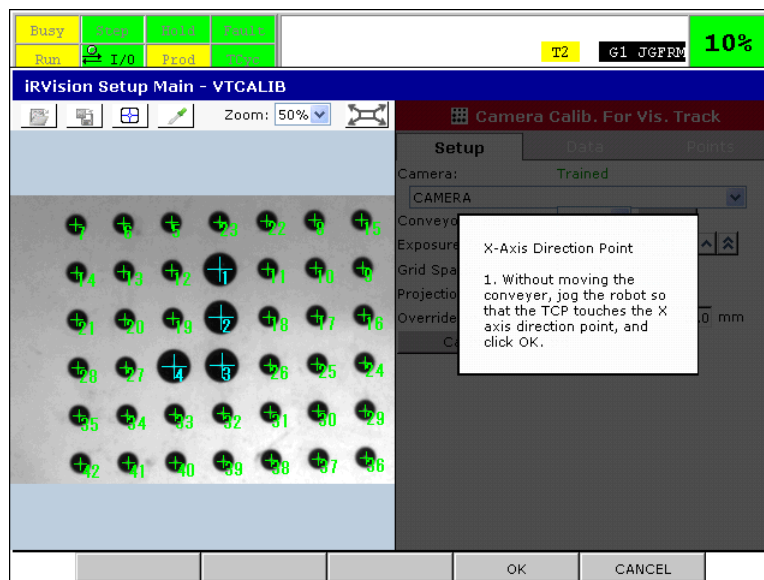


- 4 Check that the number of the user tool frame that you set in Section 3.4, "SETTING OF THE TCP" is selected.
- 5 Move the conveyor so that the calibration grid comes in front of the robot for which you are setting up the vision system.

- 6 Touch up the origin of the calibration grid. The origin is located at the intersection of large dots arranged in the shape of the letter L at the center.



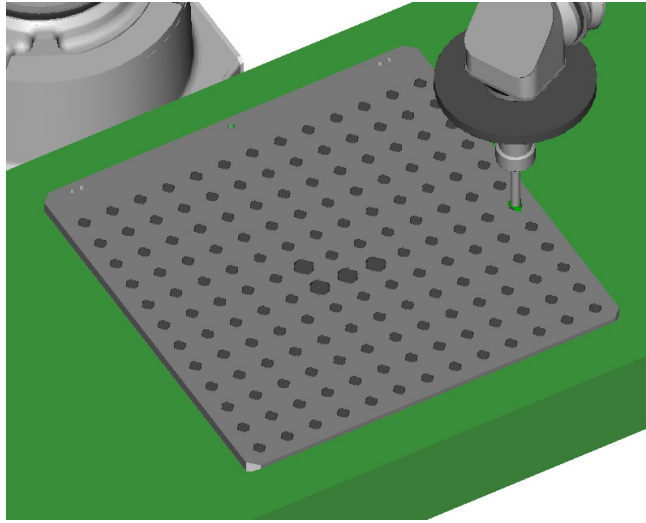
- 7 Press F4 [OK] key with the origin touched up. A screen as shown below is displayed.



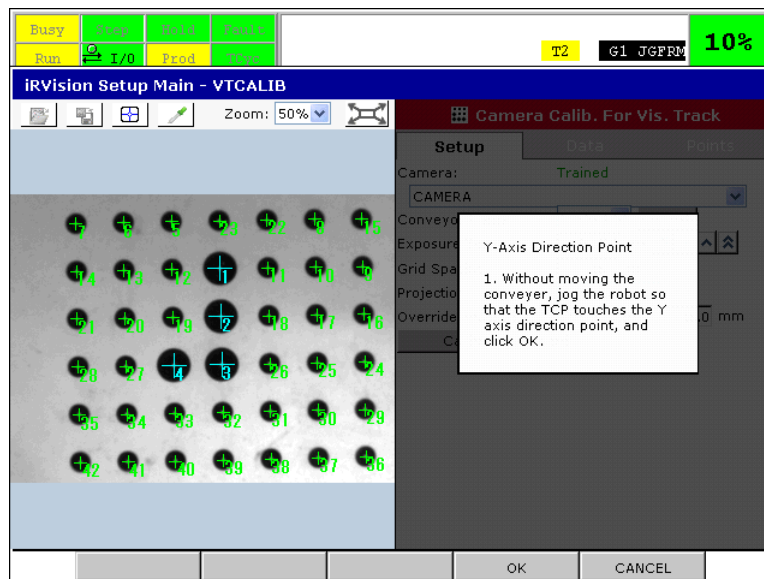
4. SETUP OF THE BASIC FUNCTIONS

B-83604EN/04

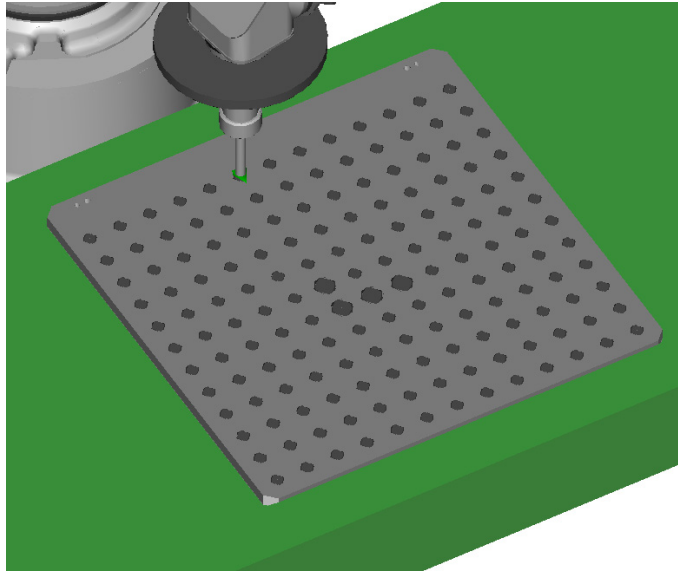
- 8 Jog the robot upward (or away from the conveyor surface) . Next touch up an X-axis direction point with the TCP. The X-axis direction is the direction of one of the lines of large dots arranged in the shape of the letter L that has more dots. Touch up the farthest possible dot from the origin.



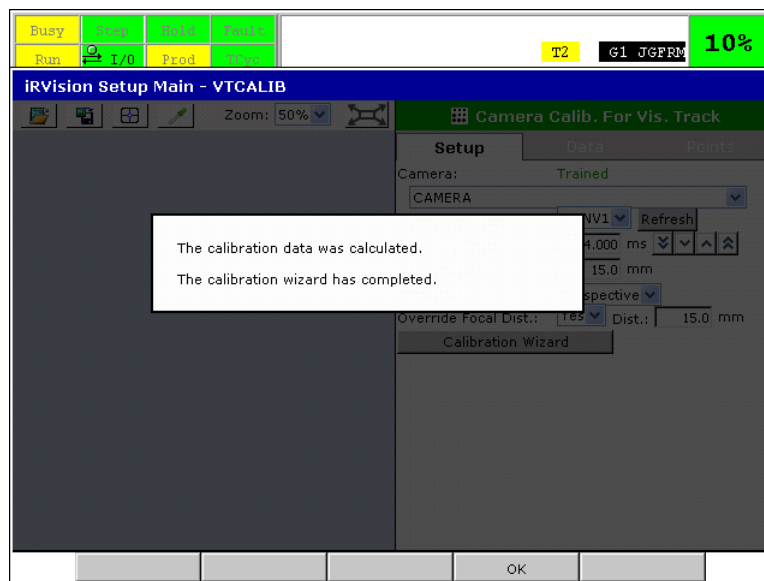
- 9 Press F4 [OK] key with the dot touched up. A screen as shown below is displayed.



- 10 Jog the robot upward (or away from the conveyor surface) . Next touch up a Y-axis direction point with the TCP. The Y-axis direction is the direction of one of the lines of large dots arranged in the shape of the letter L that has fewer dots. Touch up the farthest possible dot from the origin.



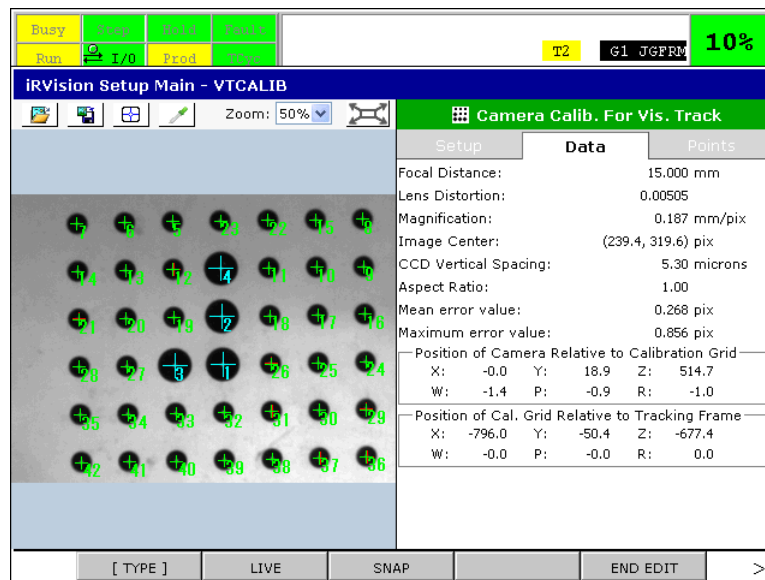
- 11 Press F4 [OK] key with the dot touched up. A screen as shown below is displayed.



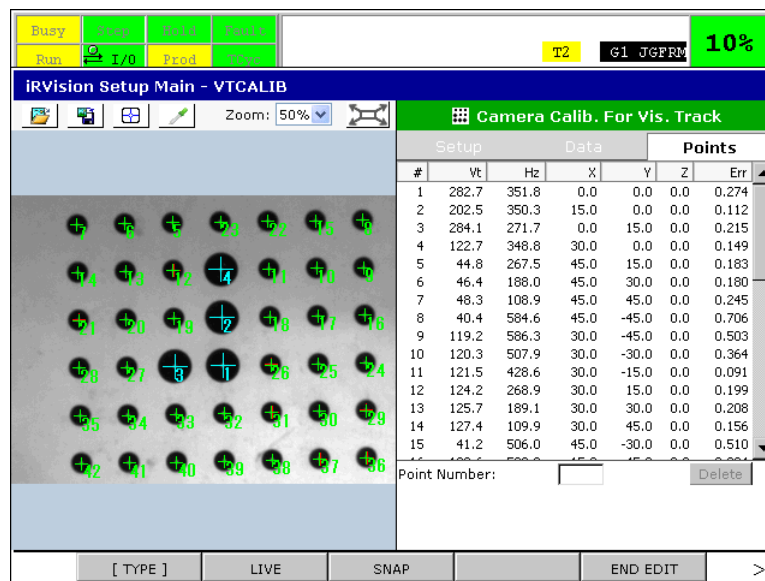
- 12 When this screen is displayed, the calibration is complete. Press F4 [OK] key to finish the procedure.

Checking the calibration result

- 1 Click the [Data] tab. An image as shown below is displayed.



- 2 Check whether the position of each + mark displayed on the image matches that of the corresponding dot of the grid pattern. If there is any incorrectly detected point, delete it as instructed below.
- 3 Select the [Points] tab. An image as shown below is displayed.



- 4 In the box to the right of [Point Number], enter the number of the point you want to delete, and then click [Delete]. The specified dot disappears from the image, and the calibration data is calculated again.
- 5 Press F10 [SAVE] key to save the data.
- 6 Press F5 [END EDIT] key to close the edit screen.

4.5.4.3 Vision process

Teach the vision process, and execute a test.

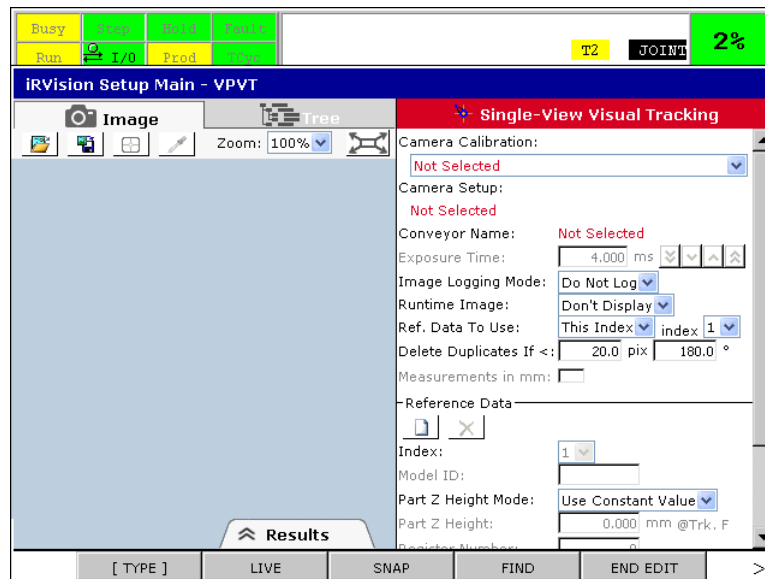
Here, only the GPM locator tool is mentioned as the tool used for detection. Other locator tools such as the blob locator tool may be used as well. For details, refer the chapter, "COMMAND TOOLS" in the "iRVision OPERATOR'S MANUAL (Reference)".

4.5.4.4 Teaching a vision process

- 1 From the [MENU] key, select [8 iRVision] and then [1 Vision setup] to display the vision data list screen. Press F4 [VTYPE] key, and select [Vision Process Tools] from the list.
- 2 Press F2 [CREATE] key. A screen as shown below is displayed.

The screenshot shows the 'iRVision Setup Main - Vision Process Tools' window. At the top, there is a status bar with 'T2', 'G1 JGFRM', and '10%'. Below this is a table with columns 'Name' and 'Comment'. In the center, a dialog box titled 'Create new vision data' is open, containing a dropdown menu for 'Type' (set to 'Single-View Visual Tracking'), and text input fields for 'Name' and 'Comment'. At the bottom of the dialog box are 'OK' and 'CANCEL' buttons.

- 3 In [Type], select [Single-View Visual Tracking].
- 4 In [Name], enter the name of the vision process (e.g., VPVT).
The name may consist of up to 34 alphanumeric characters and must not begin with a numeric character. Do not include a hyphen (-).
- 5 Press F4 [OK] key. The created vision process is displayed in the list.
- 6 Select the created vision process, and press F3 [EDIT] key. A screen as shown below is displayed.



- 7 In [Camera Calibration], select the desired calibration data.
- 8 In [Exposure Time], set the exposure time.
- 9 In [Image Logging Mode], select the image logging mode.

NOTE

Normally, images are saved in order to check undetected images later. Since saving images takes a considerable amount of time, however, it is common to select [Don't Log] for visual tracking so as to reduce the processing time. During the initial adjustment stage or in other cases where an impact on (or delay of) the cycle time can be tolerated, you may save images by selecting [Log Failed Images] or [Log All Images] and use them for such purposes as to adjust the detection parameters. Images are saved to a memory card.

- 10 In [Runtime Image], select the display mode.

NOTE

Normally, select [Display With 100%]. Selecting [Display With 50%] or [Don't Display] slightly reduces the processing time.

- 11 In [Ref. Data To Use], select the data to use.

**CAUTION**

In [Part Z Height], enter the height of the detected part of the part as seen from the tracking frame, not the height from the surface of the conveyor. For example, if a thick fixture is used to set up the tracking frame and the Z-axis direction of the tracking frame is upward relative to the conveyor surface, the value to be set is equal to "the height of the detected part of the part minus the thickness of the fixture". If this value is incorrect, the vision offset cannot be calculated correctly.

NOTE

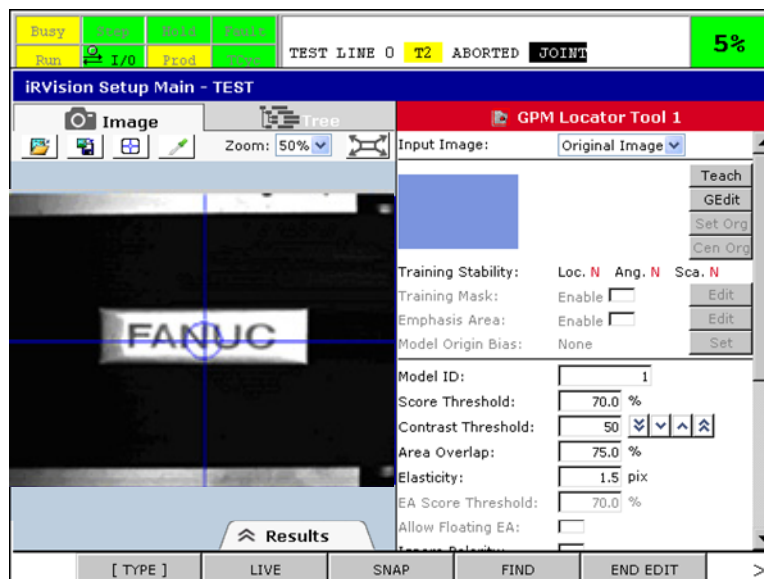
- 1 Normally, select [This Index] and, in [ID], specify the ID number of the reference data to be used at all times. If the reference data needs to be changed depending on the part detected in such cases as when multiple types of parts with different heights are conveyed, select [Model ID] and set different data in [Reference Data] for each model ID. In [Reference Data], enter an appropriate value in [Part Z Height] for each model ID.
- 2 You can switch [Part Z Height] based on a part z height measured by 3rd party sensor. In this case, you select [Use Register Value] in [Part Z Height Mode] and specify a register where the measurement value is stored in [Register Number].

4.5.4.5 Teaching a GPM locator tool

- 1 Select the tree view on the left side of the edit screen. Click [GPM Locator Tool 1], and teach a GPM locator tool. If you want multiple types of parts to be detected, you can create multiple GPM locator tools by clicking the  button with [Single-View Visual Tracking] selected.

NOTE

If parts and trays need to be detected with a single camera, it is necessary to create separate vision processes for parts and trays, respectively.



- 2 Select the image view on the left side of the edit screen, and place the part you want to teach in the center of the camera's field of view. Then, press F3 [SNAP] key to snap an image.
- 3 Click the [Teach] button. Enclose the part with the resulting magenta rectangular window, and press F4 [OK] key.
- 4 Enter appropriate values in the other fields as necessary. For details, refer to the section, "GPM LOCATOR TOOL" in the chapter, "COMMAND TOOLS" in the "iRvision OPERATOR'S MANUAL (Reference)".
- 5 Press F10 [SAVE] key to save the data.
- 6 F5 [END EDIT] key to close the edit screen.

NOTE

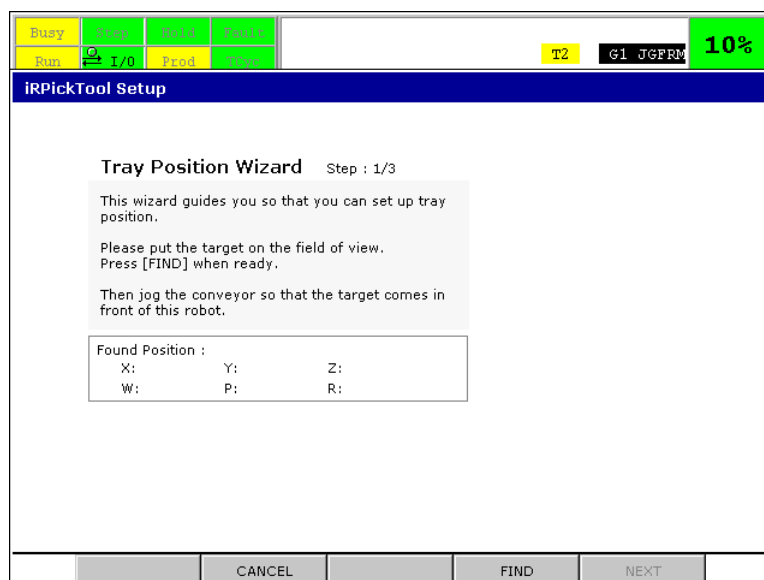
You do not need to set the reference position by clicking the [Set] button in [Reference Data] for the vision process. The reference position will be set in the sensor setup screen described later.

4.5.5 Setting the Tray Position

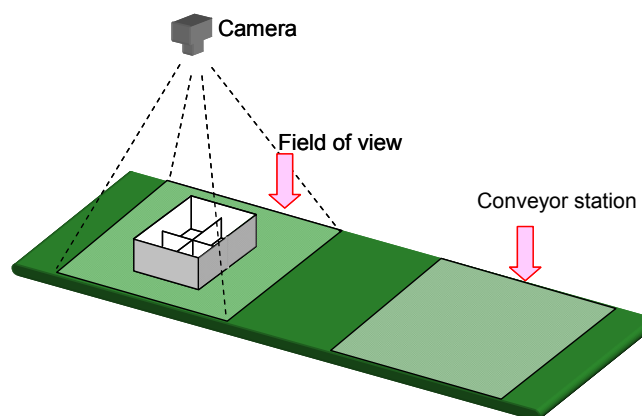
If [Find part by vision] is selected for [Trigger Action] and a tray is used, set the relative position of the origin and detection position of the tray. To do this requires that camera calibration and vision process teaching be completed in advance.

4.5.5.1 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.



- 4 When the tray is successfully detected, the detection position is displayed and F5 [NEXT] key is enabled.

- 5 If you press F5 [NEXT] key, a screen as shown below is displayed.

Busy	Stop	Hold	Fault			T2	G1 JGPRM	10%
Run	I/O	Prod	Stop					

iRPickTool Setup

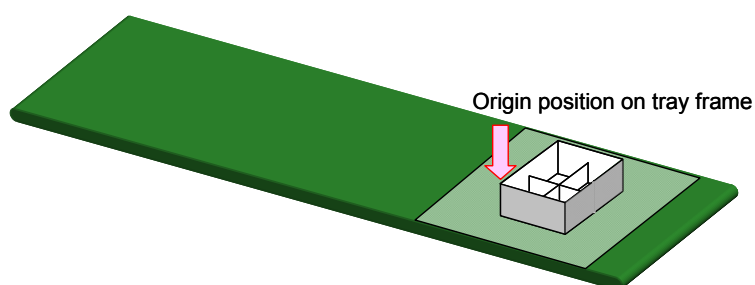
Tray Position Wizard Step : 2/3

Please jog this robot so that its TCP comes to the tray frame origin.
Press [RECORD] when ready.

Encoder Count :
Taught Position :
X: Y: Z:
W: P: R:

CANCEL PREV RECORD NEXT

- 6 Move the conveyor until the tray comes in front of the robot.
7 Jog the robot, and touch up the origin of the tray frame.



- 8 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.
9 If you press F5 [NEXT] key, a screen as shown below is displayed.

Busy	Stop	Hold	Fault			T2	G1 JGPRM	10%
Run	I/O	Prod	Stop					

iRPickTool Setup

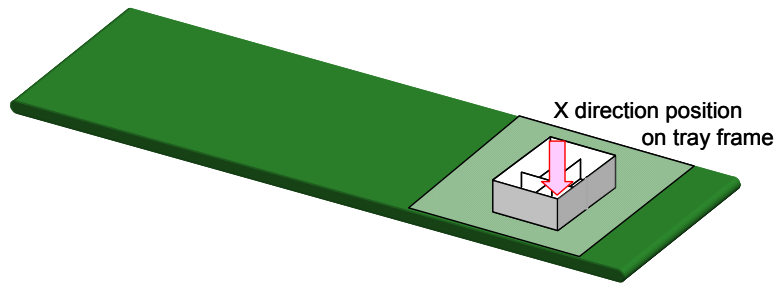
Tray Position Wizard Step : 3/3

Without moving the conveyor, please jog this robot so that its TCP comes to the tray frame x point.
Press [RECORD] when ready.

Taught Position :
X: Y: Z:
W: P: R:
Calculated Position :
X: Y: Z:
W: P: R:

CANCEL PREV RECORD FINISH

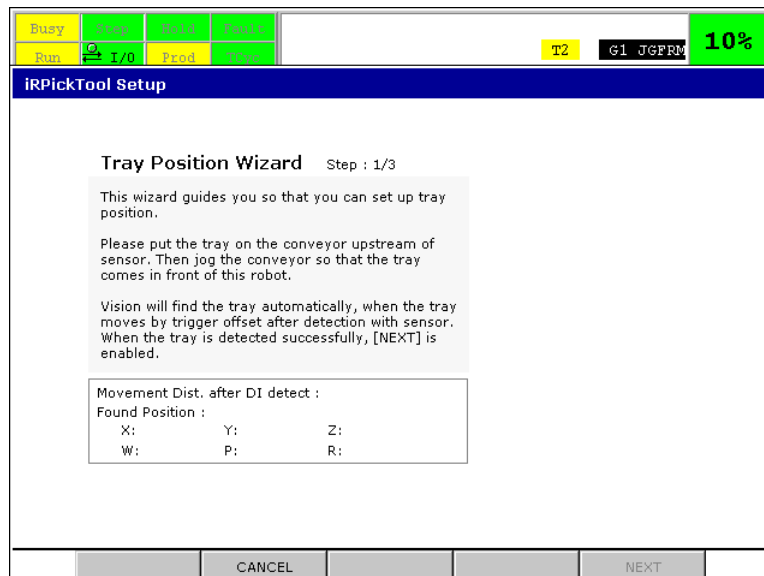
- 10 Jog the robot, and touch up the X direction point of the tray frame.



- 11 Press F4 [RECORD] key. The touch-up position and the calculated position are displayed, and F5 [FINISH] key is enabled.
- 12 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.

4.5.5.2 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 Let the tray travel from the most upstream end of the conveyor. When the conveyor moves by the trigger offset after the photo eye spots the part, iRVision automatically detects the part. You do not need to stop the conveyor when iRVision detects the part. Stop the conveyor after it reaches this robot operation range. When iRVision successfully detects the part, the detection position are displayed and F5 [NEXT] key is enabled.

- 4 If you press F5 [NEXT] key, a screen as shown below is displayed.

Busy	Stop	Hold	Pause			T2	G1 JGPRM	10%
Run	I/O	Prod	Stop					

iRPickTool Setup

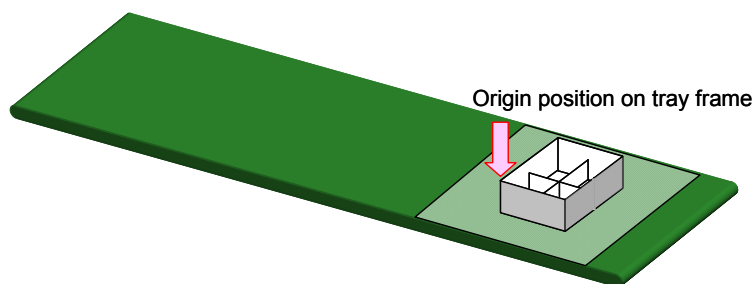
Tray Position Wizard Step : 2/3

Please jog this robot so that its TCP comes to the tray frame origin.
Press [RECORD] when ready.

Encoder Count :
Taught Position :
X: Y: Z:
W: P: R:

CANCEL PREV RECORD NEXT

- 5 Jog the robot, and touch up the origin of the tray frame.



- 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 If you press F5 [NEXT] key, a screen as shown below is displayed.

Busy	Stop	Hold	Pause			T2	G1 JGPRM	10%
Run	I/O	Prod	Stop					

iRPickTool Setup

Tray Position Wizard Step : 3/3

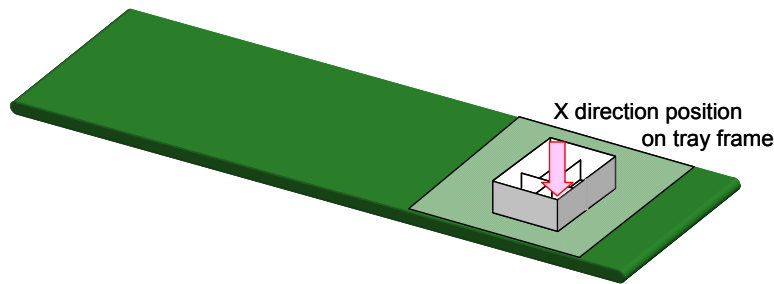
Without moving the conveyor, please jog this robot so that its TCP comes to the tray frame x point.
Press [RECORD] when ready.

Taught Position :
X: Y: Z:
W: P: R:

Calculated Position :
X: Y: Z:
W: P: R:

CANCEL PREV RECORD FINISH

- 8 Jog the robot, and touch up the X direction point of the tray frame.



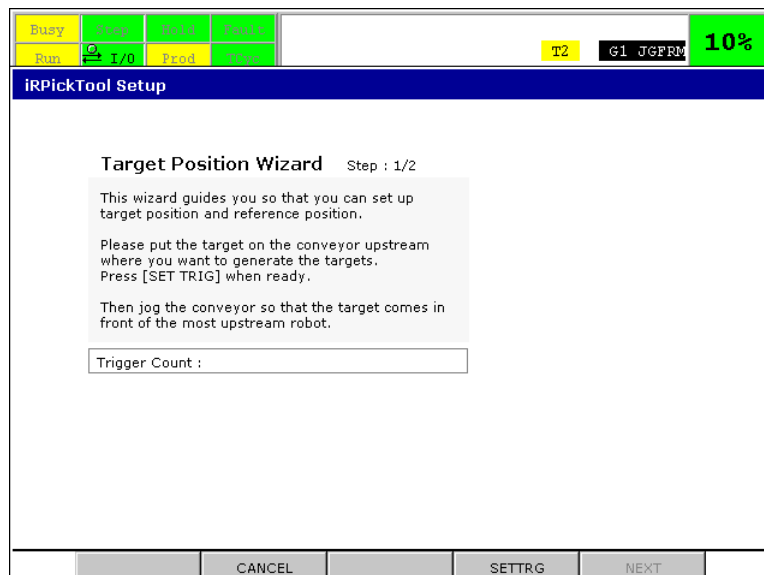
- 9 Press F4 [RECORD] key. The touch-up position and the calculated position are displayed, and F5 [FINISH] key is enabled.
- 10 If you press F5 [FINISH] key, the initial setup screen is displayed again. Values are input for X, Y, and R of the tray position.

4.5.6 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.

4.5.6.1 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.

Busy	Stop	Hold	Teach			
Run	Q I/O	Prod	Reset		T2	G1 JGPRM 10%

irPickTool Setup

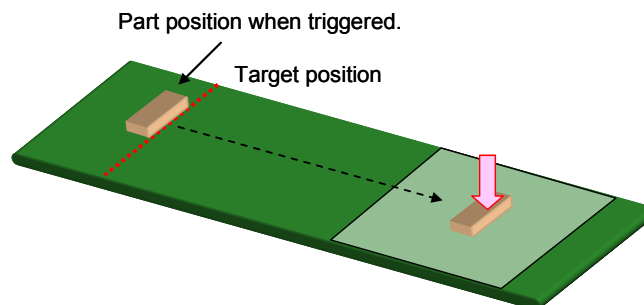
Target Position Wizard Step : 2/2

Please teach robot position in TP program from the most upstream robot in order.
 But at this robot's turn, please jog this robot at first so that its TCP comes to the target.
 Press [RECORD] when ready. After you press [FINISH], please teach robot position in TP program.
 When tray is used, please touch up tray frame origin. Then you press [NEXT].

Encoder Count :		
Taught Position :		
X:	Y:	Z:
W:	P:	R:
Calculated Position :		
X:	Y:	Z:
W:	P:	R:

		CANCEL	PREV	RECORD	FINISH	

- 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, first touch up the downstream-side end of the part with the TCP and then press F4 [RECORD] key.

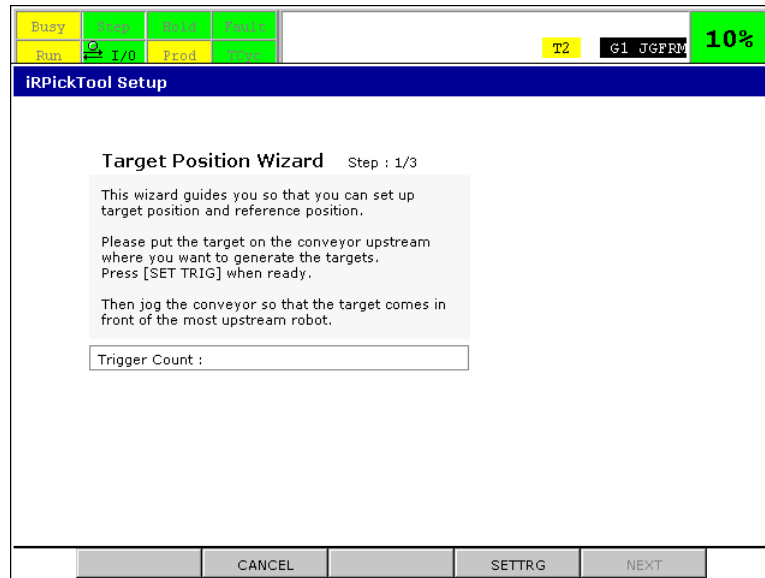


The encoder value obtained at the time of touch-up, the touch-up position, and the calculated position are displayed, and F5 [FINISH] key is enabled.

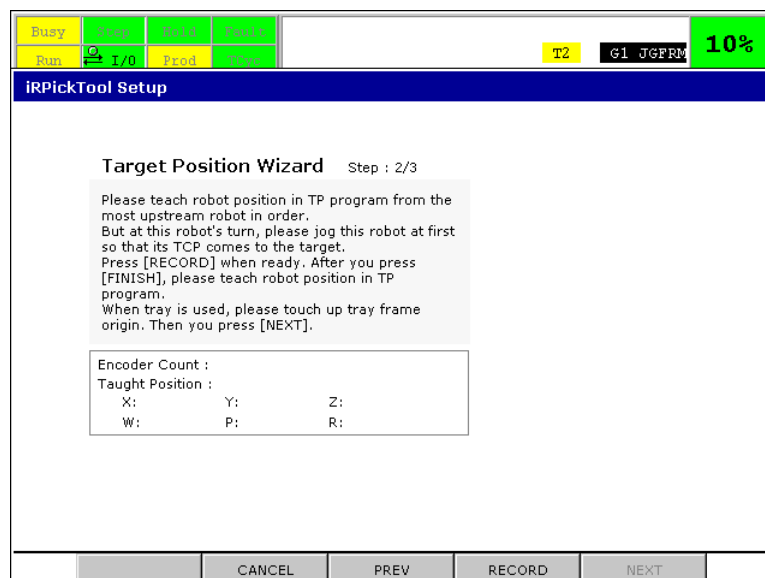
- 7 If you press F5 [FINISH] key, the initial setup screen is displayed again. The calculated value is input for X of the target position (the values of Y and R remain unchanged from 0).
- 8 Teach the position to this robot. Then, move the conveyor further to teach the position to the downstream robots.

4.5.6.2 When a tray is used

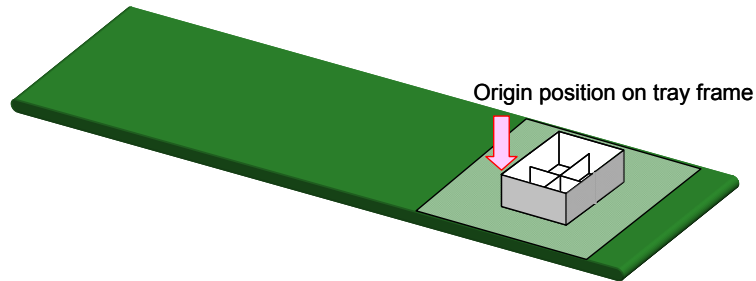
- 1 Stop the conveyor.
- 2 Press F4 [SET TRG POS] key. A screen as shown below is displayed.



- 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 4.11, "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.



- 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.

Busy	Stop	Hold	Teach				
Run	I/O	Prod	Prog	T2	G1 JGFRM	10%	

IRPickTool Setup

Target Position Wizard Step : 3/3

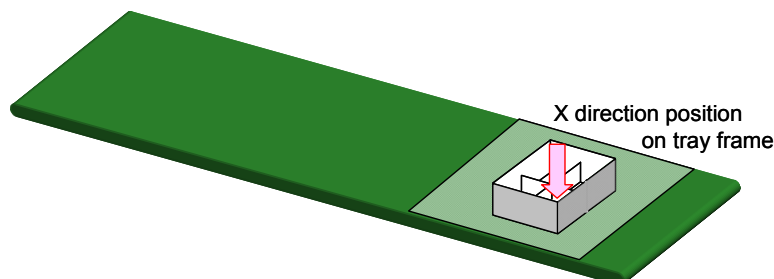
Without moving the conveyor, please jog this robot so that its TCP comes to the tray frame x point. Press [RECORD] when ready.

After you press [FINISH], please restart teaching robot position in TP program.

Taught Position :		
X:	Y:	Z:
W:	P:	R:
Calculated Position :		
X:	Y:	Z:
W:	P:	R:

CANCEL PREV RECORD FINISH

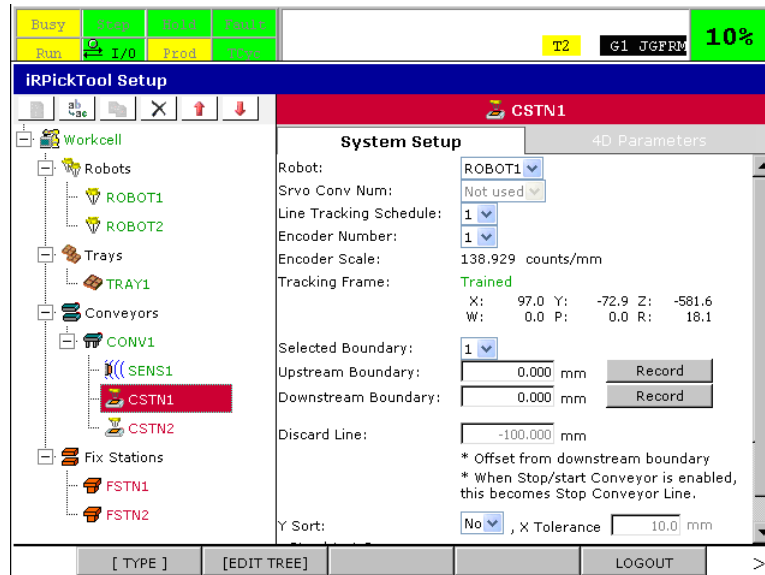
- 9 Jog the robot, and touch up the X direction point of the tray frame. Press F4 [RECORD] key. The touch-up position and the calculated position are displayed, and F5 [FINISH] key is enabled.



- 10 If you press F5 [FINISH] key, the initial setup screen is displayed again. Values are input for X, Y, and R of the target position.
11 Teach the position to this robot. Then, move the conveyor further to teach the position to the downstream robots.

4.6 SETUP OF A CONVEYOR STATION

Set up each conveyor station.



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 iRPickTool menu. For details, see Appendix D, "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.

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 4.9.2.2, "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.

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.

After setting [Upstream Boundary] and [Downstream Boundary], change the tool frame back to the one that is set in Section 3.4, "SETTING OF THE TCP".

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:

$$0.8 \text{ (s)} \times 100 \text{ (mm/s)} = 80 \text{ (mm)}$$

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:

$$0.8 \text{ (s)} \times 5 \text{ (deg/s)} = 4 \text{ (deg)}$$

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.

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**

If you set a tray in a conveyor, Y sort cannot work.

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.

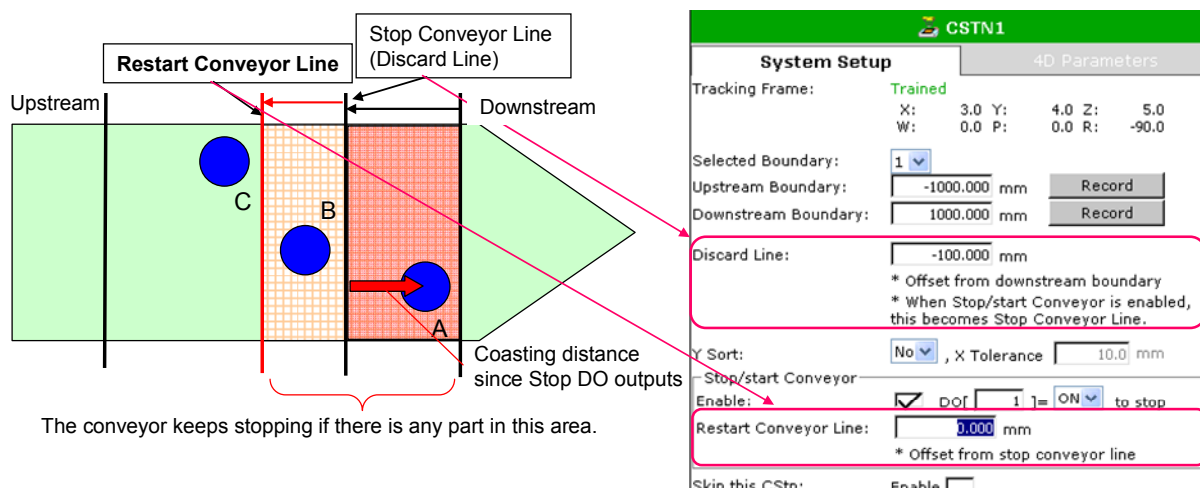
When using this function, specify the number of the DO signal used to stop and restart the conveyor, as well as the polarity indicating whether to turn the signal on or off to stop the conveyor.

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.

See the figure below. The conveyor stopped because the part A passes the stop conveyor 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.



⚠ 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

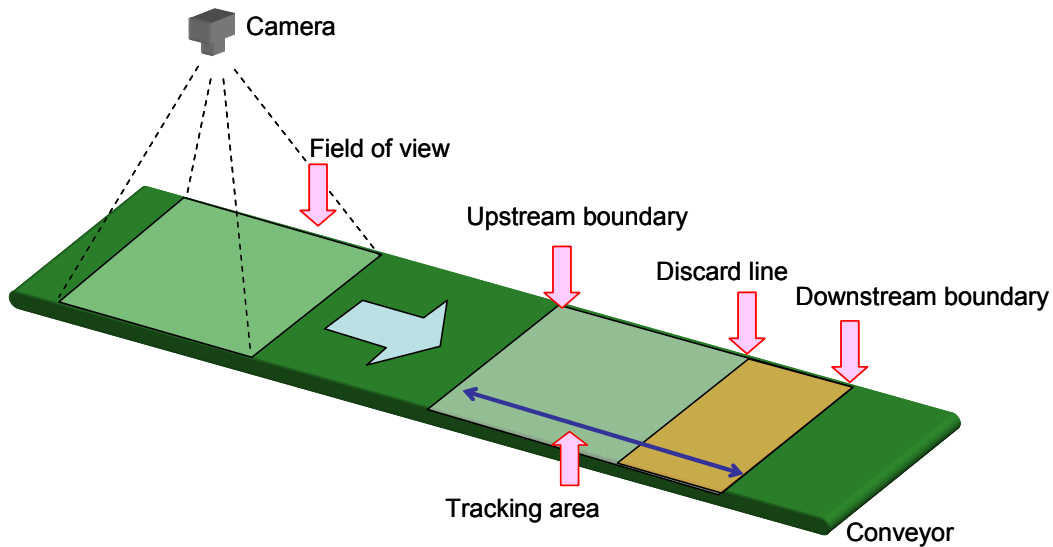
Depending on the setup of other items, the setting items of the pre-grouping function may be shown as below.

Pre-group	
Generation Line	0.000 mm
Send Line	0.000 mm
	* Offset from discard line
Grouping Distance	Auto, mm

For details, see Section 4.14, "SETUP OF PRE-GROUPING".

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.



4.6.1 Adding a Tracking Frame

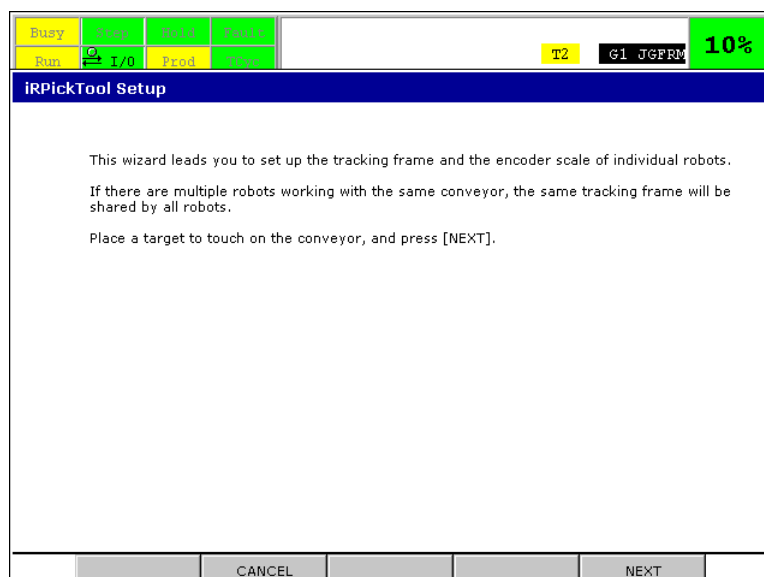
After you complete the setting of a tracking frame for a conveyor on the conveyor setup page, you can redo the tracking frame setting for a specific conveyor station. Use this procedure in cases such as when you have selected the wrong user tool frame for a robot during the tracking frame setting or 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.

4.6.1.1 Line Conveyor

When the tracking type is “Line”, see this procedure. When the tracking type is “Circle”, see the 4.6.1.2, “Circular Conveyor”.

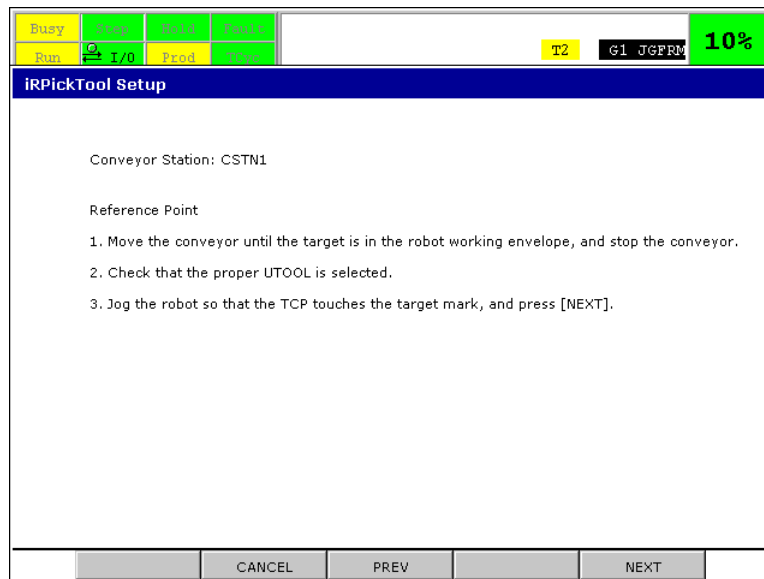
- 1 In the conveyor station setup screen, press F4 [ADD TRK FRM] key. A screen as shown below is displayed.



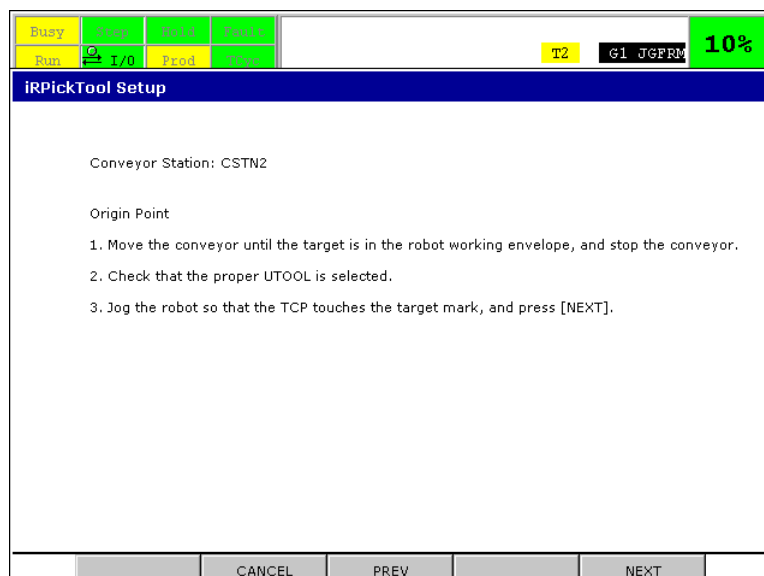
NOTE

- 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.

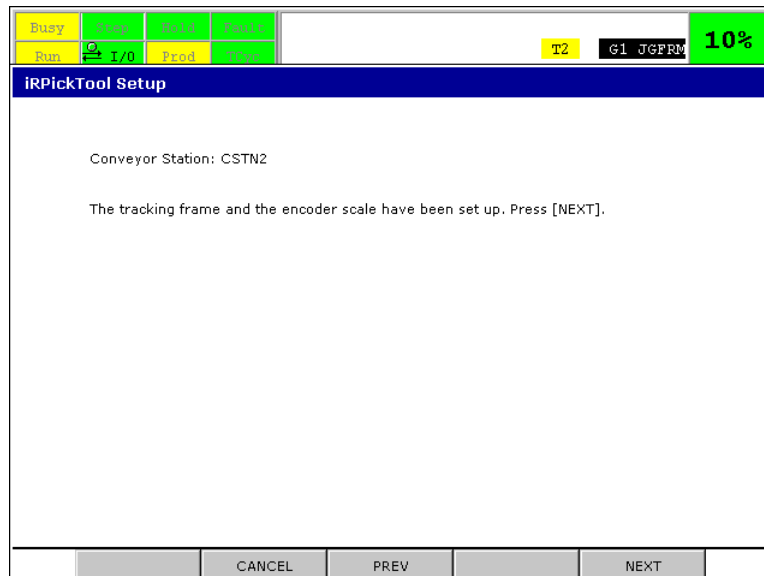
- 2 Place a fixture that can be touched up with the TCP of the robot at the most upstream area of the conveyor, and press F5 [NEXT] key. A screen as shown below is displayed.



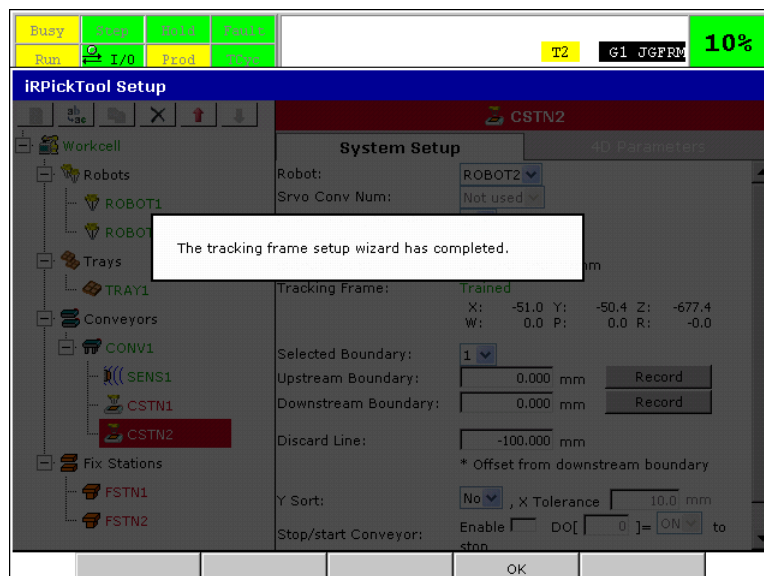
- 3 For the most upstream robot, select the user tool frame set in Section 3.4, "SETTING OF THE TCP" as the jog feed frame.
- 4 Move the conveyor until the fixture comes into the most upstream robot operation range, and stop it. Jog the robot to touch up the fixture with the TCP. With the fixture touched 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.



- 5 For the conveyor station for which you are redoing the tracking frame setting, touch up the fixture as instructed in steps 4 through 7 in Subsection 4.4.1, "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.



- 6 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. If you press F5 [NEXT] key, a screen as shown below is displayed.

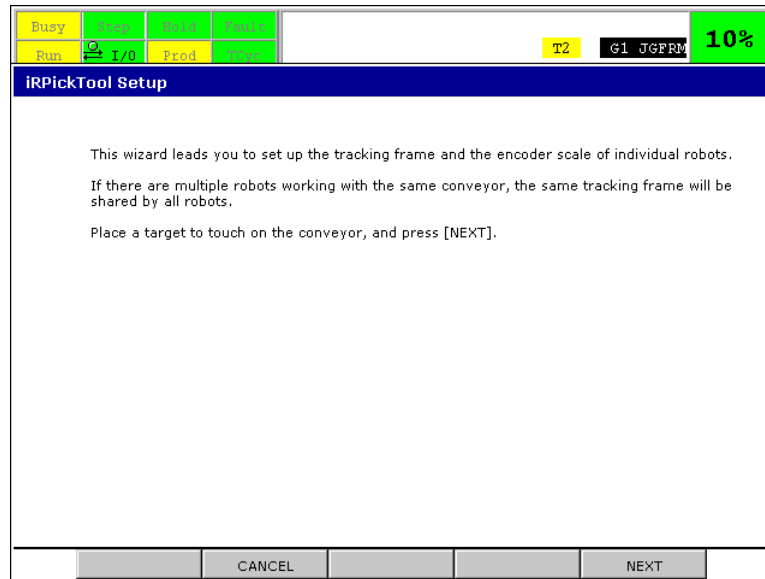


- 7 Press F4 [OK] key to finish the tracking frame addition procedure.

4.6.1.2 Circular Conveyor

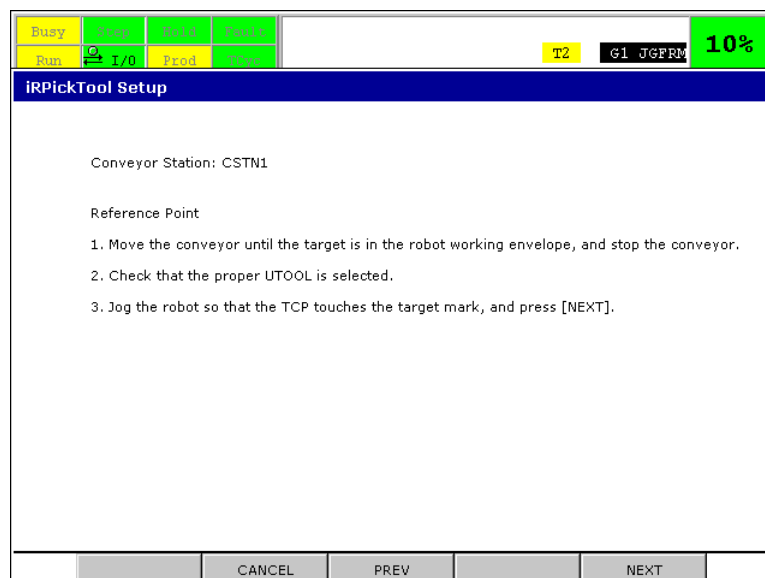
When the tracking type is "Circle", see this procedure.

- 1 In the conveyor station setup screen, press F4 [ADD TRK FRM] key. A screen as shown below is displayed.

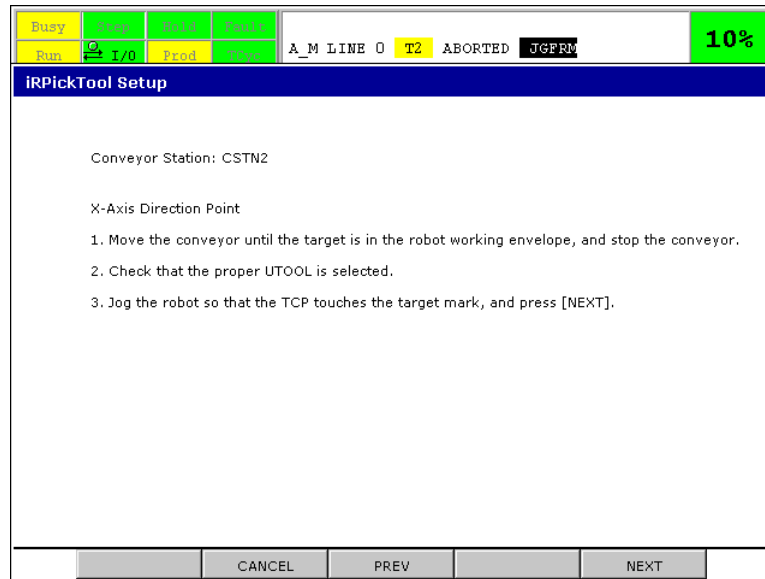
**NOTE**

- 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.

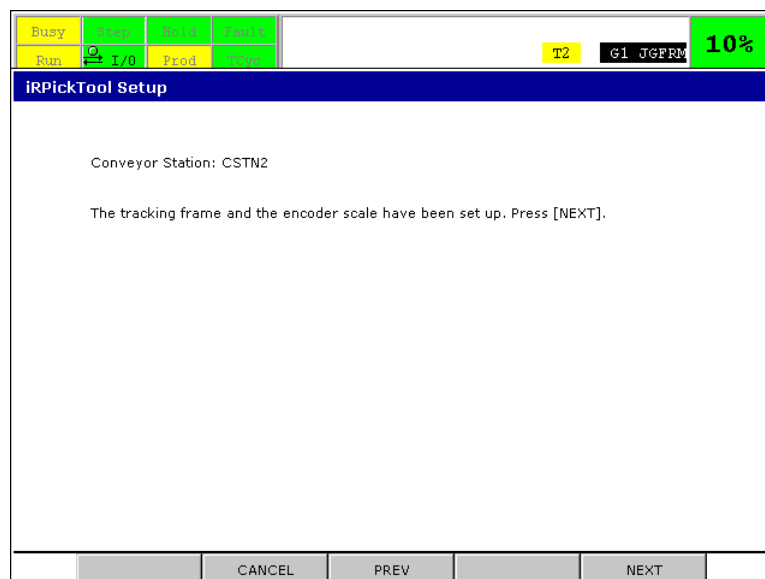
- 2 Place a fixture that can be touched up with the TCP of the robot at the most upstream area of the conveyor, and press F5 [NEXT] key. A screen as shown below is displayed.



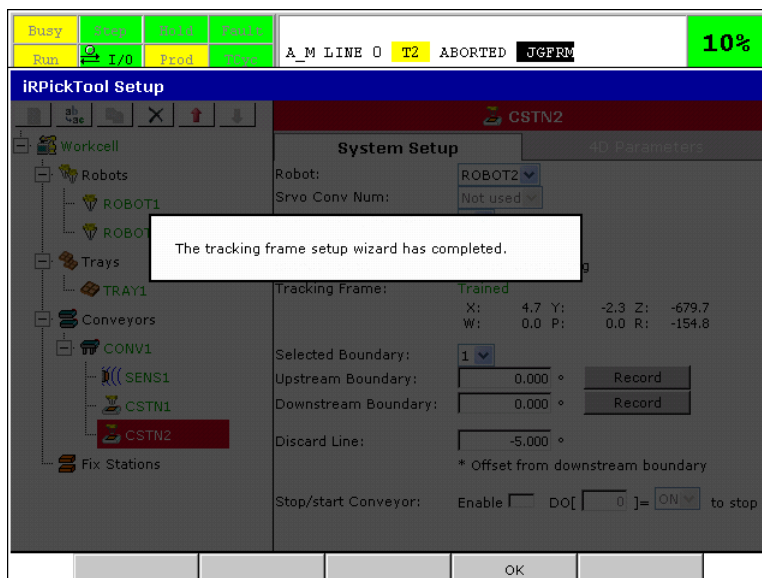
- 3 For the most upstream robot, select the user tool frame set in Section 3.4, "SETTING OF THE TCP" as the jog feed frame.
- 4 Move the conveyor until the fixture comes into the most upstream robot operation range, and stop it. Jog the robot to touch up the fixture with the TCP. With the fixture touched 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.



- 5 For the conveyor station for which you are redoing the tracking frame setting, touch up the fixture as instructed in steps 4 through 7 in Subsection 4.4.1, "Setting a Tracking Frame". When you have proceeded to the step of setting the assistant point, a screen as shown below is displayed.



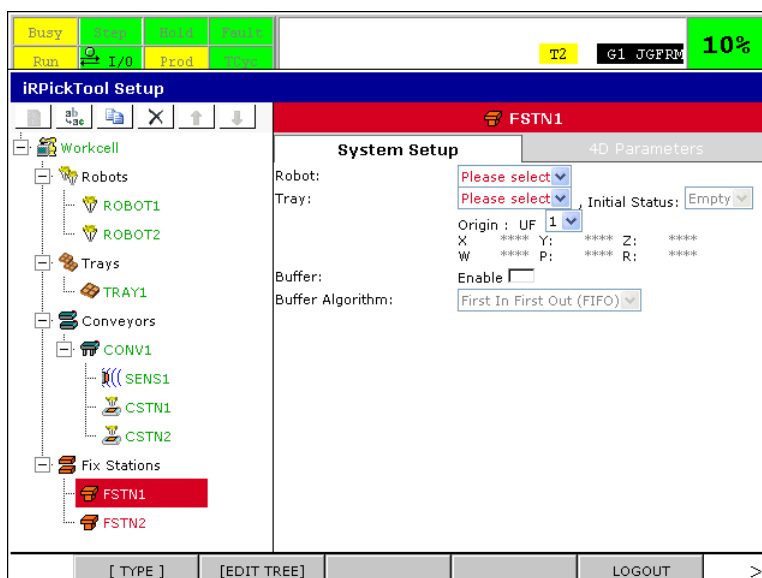
- 6 This completes the tracking frame addition procedure, and the same encoder scale as the most upstream robot and the newly calculated tracking frame are set. If you press F5 [NEXT] key, a screen as shown below is displayed.



- 7 Press F4 [OK] key to finish the tracking frame addition procedure.

4.7 SETUP OF A FIXED STATION

Set up a fixed station.



Robot

Select the robot that operates with this fixed station.

Tray

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 [6 SETUP] and then [Frames] from the [MENU] key, instead of the iRPickTool 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 "R-30iB /R-30iB Mate CONTROLLER OPERATOR'S MANUAL (Basic Operation)".

**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.

4.8 KAREL PROGRAMS

The iRPickTool 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 iRPickTool 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.

4.8.1 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.

4.8.1.1 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 4.8.4.2, "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 4.8.3.6, "PKSNSTART.PC".

There is no argument.

Example:

```
CALL PKWCSTART
```

4.8.1.2 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
```

4.8.1.3 PKWCHECK.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 PKWCHECK
```

4.8.1.4 PKWCRSTPRDST.PC

This program resets Production Status of iRPickTool. For Production Status, see the section 7.2 "CHECKING THE PRODUCTION STATUS" in detail.

There is no argument.

Example:

```
CALL PKWCRSTPRDST
```

4.8.1.5 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, you can get the value of \$pkcs[1].\$n2pick[2]

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, specify 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)
```

4.8.1.6 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)
```

4.8.1.7 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)
```

4.8.2 Conveyor-Related Programs

The two letters following [PK] are [CV]; namely, the names of these programs begin with these four letters, [PKCV].

4.8.2.1 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 If you specify Argument 5 (model ID), the argument guidance is not displayed.
- 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, PKSNSETMVNAM.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 PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 "Need to synchronize data" occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

4.8.2.2 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. This program can be used in 7DC3/02(V8.30P/02) or later.

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, PKSNENBSSENS.PC, PKSNSETVPNAM.PC, PKSNSETMVNAM.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 PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 "Need to synchronize data" occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

4.8.2.3 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, PKSNSETVPPNAM.PC, PKSNSETMVNAM.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 PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 "Need to synchronize data" occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

4.8.2.4 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, PKSNENBSSENS.PC, PKSNSETVPPNAM.PC, PKSNSETMVNAM.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 PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 "Need to synchronize data" occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

4.8.3 Sensor-Related Programs

The two letters following [PK] are [SN]; namely, the names of these programs begin with these four letters, [PKSN].

4.8.3.1 PKSNENBSSENS.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. This program can be used in 7DC3/03(V8.30P/03) or later.

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, PKSNSETMVNAM.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 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCCHECK is called.

4.8.3.2 PKSNSETVPNAM.PC

This program changes a vision process in a sensor. 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, PKSNENBSSENS.PC, PKSNSETVPNAM.PC, PKSNSETMVNAM.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 PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

4.8.3.3 PKSNSETMVNAM.PC

When you use [DI] + [Find part by multiview], this program changes a vision process 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. This program can be used in 7DC3/18(V8.30P/18) or later.

Argument 1:

Specify the name of the sensor using a character string.

Argument 2:

Specify the name of the vision process for view 1 using a character string.

Argument 3:

Specify the name of the vision process for view 2 using a character string.

Example:

The following instruction changes vision process names of multiview which are selected in “SENS1” to “VT1” and “VT2”.

```
CALL PKSNSETMVNAM('SENS1',VP Name 1='VT1',VP Name 2='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, PKSNSETVPPNAM.PC, PKSNSETMVNAM.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 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCCHECK is called.

4.8.3.4 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. This program can be used in 7DC3/18(V8.30P/18) or later.

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, PKSNENBSSENS.PC, PKSNSETVPNAM.PC, PKSNSETMVNAM.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 PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

4.8.3.5 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. This program can be used in 7DC3/18(V8.30P/18) or later.

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, PKSNENBSSENS.PC, PKSNSETVPNAM.PC, PKSNSETMVNAM.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 PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 "Need to synchronize data" occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

4.8.3.6 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
```

4.8.3.7 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
```

4.8.4 Conveyor Station-Related Programs

The two letters following [PK] are [CS]; namely, the names of these programs begin with these four letters, [PKCS].

4.8.4.1 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)
```

NOTE

If you specify Argument 3 (group number), the argument guidance is not displayed.

4.8.4.2 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])

4.8.4.3 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 4.9.2.2, "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 5, "VARIATIONS".

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.
3. If you specify Argument 6 (model ID), the argument guidance is not displayed.

4.8.4.4 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)
```

⚠ 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.

4.8.4.5 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.

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)
```

4.8.4.6 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 *iRPickTool*.

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 number of the register to store the overlap check result. If the information is input with no overlapping part, 0 is stored in the specified register. If there is any overlapping part 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)
```

4.8.4.7 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.

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".

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)
```

NOTE

If you specify Argument 8 (model ID), the argument guidance is not displayed.

4.8.4.8 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.

This KAREL program cannot be used from the time PKCSGETQUEUE returns 0 as the processing status until PKCSACKQUEUE is called. Calling the program during this period causes the error "No ACKQUEUE after the last GETQUEUE".

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 PKCSGETQUEUE 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:

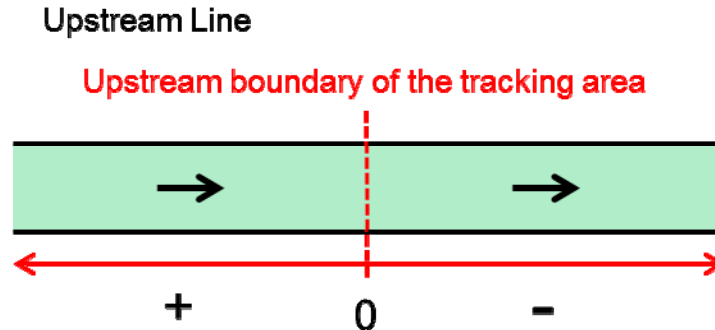
Normally, specify 1. When a robot gripper capable of holding multiple parts is used and you want the robot to consecutively pick as many parts as the number of parts that its gripper can hold, specify which part to pick next. For example, if the gripper can hold four parts and it is now holding two parts, 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.

**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)

NOTE

If you specify Argument 8 (model ID), the argument guidance is not displayed.

4.8.4.9 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 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 is referred to as the expected number of parts. 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 $((7 + 3) \div 2 - 4)$.

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.

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)

4.8.4.10 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)

4.8.4.11 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. This program can be used in 7DC3/03(V8.30P/03) or later.

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 PKSNSSTART, 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, PKSNSETVPPNAM.PC, PKSNSETMVNAM.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 PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 "Need to synchronize data" occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

4.8.4.12 PKCSGETENC.PC

This program returns the encoder ID which is used in the specified conveyor station. This program can be used in 7DC3/23(V8.30P/23) or later.

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)
```

4.8.4.13 PKCSGETTRID.PC

This program returns the tray ID which is used in the specified conveyor station. This program can be used in 7DC3/23(V8.30P/23) or later.

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)
```

4.8.5 Fixed Station-Related Programs

The two letters following [PK] are [FS]; namely, the names of these programs begin with these four letters, [PKFS].

4.8.5.1 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)
```

4.8.5.2 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])
```

4.8.5.3 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])
```

4.8.5.4 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)
```

⚠ 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.
2. If you specify Argument 4 (model ID), the argument guidance is not displayed.

4.8.5.5 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)
```

4.8.5.6 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)
```

4.8.5.7 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)
```

4.8.5.8 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

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 (buffer), you must make the taught position conform to the position of the cell 1.
2. If you specify Argument 6 (model ID), the argument guidance is not displayed.

4.8.5.9 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)
```

NOTE

If you specify Argument 3 (Vision Register), the argument guidance is not displayed.

4.8.6 Tray-Related Programs

The two letters following [PK] are [TR]; namely, the names of these programs begin with these four letters, [PKTR].

4.8.6.1 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, PKSNENBSSENS.PC, PKSNSETVPNAM.PC, PKSNSETMVNAM.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 PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 "Need to synchronize data" occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

4.8.6.2 PKTRGETNUMCL.PC

This program returns the number of cells in the specified layer of the specified tray. This program can be used in 7DC3/23(V8.30P/23) or later.

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)
```

4.9 ROBOT PROGRAMS

Create robot programs as appropriate for 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.

The section first describes 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. A sample program is presented below.

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 4.11, "TEACHING THE POSITION TO ROBOTS" later in this chapter.



CAUTION

By default, *i*RPickTool has hot start disabled.

4.9.1 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.
- 2 From the list, select a tracking program.
- 3 Press F2 [DETAIL] key.
- 4 Press F3 [NEXT] key.
- 5 A screen as shown below is displayed.

Busy	Run	I/O	Prod	Fault	T1	JOINT	10%
Line Track Program Header 1/3							
1 Line track schedule number: 0							
2 Continue track at prog end: TRUE							
3 Selected Boundary: 0							
END	PREV	NEXT					

- 6 In [Line track schedule number], set the tracking schedule number specified for the conveyor station.
- 7 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.

- 8 Leave the value in [Selected Boundary] unchanged from 0. If 0 is set, the tracking area specified for the conveyor station is used.

4.9.2 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 4.9.2(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 4.9.2(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

4.9.2.1 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] 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]

```

Description

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 PKWCHECK
CALL PKCSCLRQUE(CStn ID=R[11:CSTN1])
CALL PKCSCLRQUE(CStn ID=R[12:CSTN2])
CALL PKSNSTART

```

First, PKWCHECK is called to check for any invalid *iRPickTool* 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.

4.9.2.2 Pick program

This program performs a pickup motion on an infeed conveyor.

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]

```

Description

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.

On lines 1 and 2, a user tool frame number and a user frame number are selected. As a user tool frame number, 0 is usually selected. In a tracking program, 0 is always selected as a user frame number. Otherwise, it may cause the unexpected robot motion

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 iRPickTool 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 4.12.4, "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 4.12.5, "Linear Distance Specification" and Subsection 4.12.8, "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 4.9.1, "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 this, 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.

4.9.2.3 Placement program

This program performs a placement motion on an outfeed conveyor.

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]
```

Description

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.

4.9.3 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 4.9.2, "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 4.9.3(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 4.9.3(b) Position registers

PR[1]	Initial standby position
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

4.9.3.1 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=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: J PR[1:Perch Pos] 100% FINE
9:
10: CALL PKWCSTART
11:
12: LBL[100]
13: CALL PICKCS
14: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
15:
16: CALL DROPFS
17: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
18: JMP LBL[100]
19:
20: LBL[900]
21: CALL PKWCEND
[End]
```

Description

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.

4.9.3.2 Pick program

This program performs a pickup motion on a conveyor.

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]
```

Description

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 4.9.2.2, "Pick program". Note that gripper zones are not used here.

4.9.3.3 Placement program

This program performs a placement motion on a fixed station.

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]

```

Description

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 4.12.5 "Linear Distance Specification", 4.12.7 "Maximum Linear Speed Function", and 4.12.8 "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.

4.9.4 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 4.9.2, "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 4.9.4(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 4.9.4(b) Position registers

PR[1]	Initial standby position
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

4.9.4.1 Main program

This is the main program of the system.

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: J PR[1:Perch Pos] 100% FINE

```

```

12: CALL PKWCSTART
13:
14: R[10:Pick]=0
15: LBL[100]
16: !Pick a part from a CSTN
17: CALL PICKCS
18: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
19: IF R[10:Pick]=1,JMP LBL[200]
20:
21: !Pick a part from a buffer
22: CALL PICKBF
23: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
24: IF R[10:Pick]=0,JMP LBL[100]
25:
26: LBL[200]
27: !Drop a part to a CSTN
28: CALL DROPCS
29: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
30: IF R[10:Pick]=0,JMP LBL[100]
31:
32: !Drop a part to a buffer
33: CALL DROPBF
34: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
35: IF R[10:Pick]=1,JMP LBL[200]
36: JMP LBL[100]
37:
38: LBL[900]
39: CALL PKWCEND
[End]

```

Description

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.

4.9.4.2 Pick program

This program performs a pickup motion on a conveyor.

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]

```

Description

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 msec in order to reduce the CPU load.

4.9.4.3 Placement program

This program performs a placement motion on a conveyor.

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]

```

Description

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.

4.9.4.4 Pick program (Buffer)

This program performs a pickup motion on a fixed station for 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:FsgGetQStat]<=0,JMP LBL[100]
7:  IF R[5:FsgGetQStat]>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]
```

Description

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 4.12.5 "Linear Distance Specification", 4.12.7 "Maximum Linear Speed Function", and 4.12.8 "Time Before Instruction".

4.9.4.5 Placement program

This program performs a placement motion on a fixed station for 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:FsgGetQStat]<=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(R[8:BufferID],1,2)
18: R[10:Pick]=0
19:
20: LBL[900]
[End]

```

Description

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.

On line 17, because the argument 3 is specified in PKFSACKBUF, the argument guidance is not displayed. If you do not specify the argument 3, the description is changed below.

```
17: CALL PKFSACKBUF(FStn ID=R[8:BufferID],Success)
```

4.10 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 wizard 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.

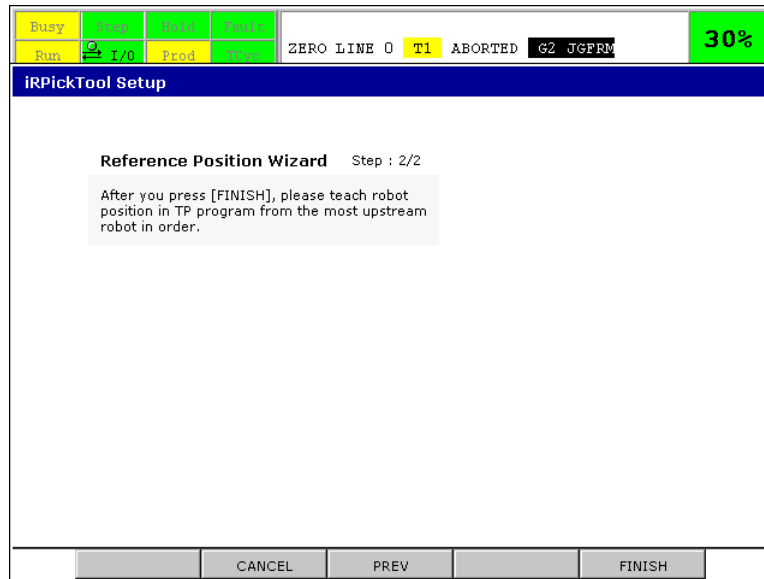
The procedure slightly differs depending on the combination of a trigger condition and an action.

4.10.1 When [Distance] and [Find part by vision] Are Selected

- 1 Stop the conveyor.
- 2 Press F3[SET REF POS] key A screen as shown below is displayed.

The screenshot shows the 'iRPickTool Setup' window. At the top, there is a status bar with buttons: 'Busy' (yellow), 'Stop' (green), 'Reset' (yellow), 'Manual' (green), 'Run' (yellow), 'I/O' (green), 'Prod' (yellow), and 'Abort' (green). To the right of these buttons, it says 'PKRSTWC LINE 0 T1 ABORTED G1 JGPRM' and a green box with '10%'. Below the status bar is a blue header 'iRPickTool Setup'. The main area is titled 'Reference Position Wizard Step : 1/2'. It contains the following text: 'This wizard guides you so that you can set up reference position.', 'Please put the target on the field of view. Press [FIND] when ready.', and 'Then jog the conveyor so that the target comes in front of the most upstream robot.' Below this text is a box for 'Trigger Count : Found Position :' with fields for X, Y, Z, W, P, and R. At the bottom, there are three buttons: 'CANCEL', 'FIND', and 'NEXT'.

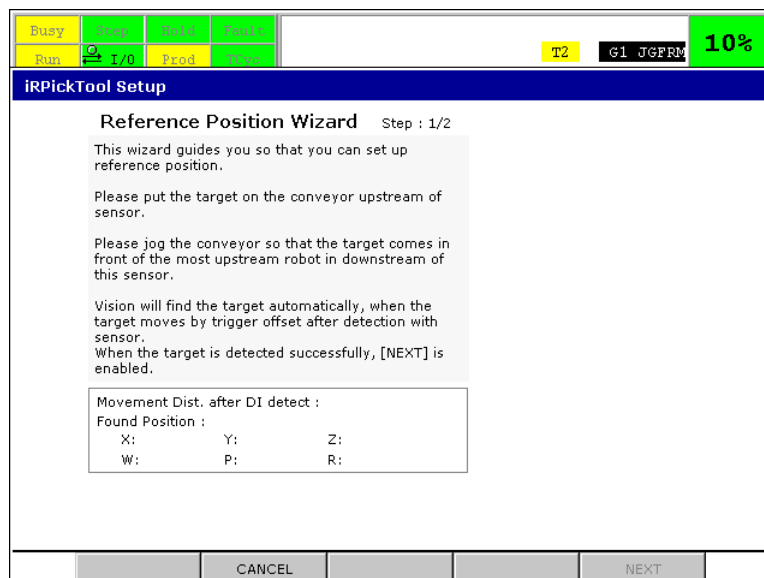
- 3 Place a part near the center of the camera's field of view.
- 4 Press F4 [FIND] key.
- 5 If iRVision detects the part, the encoder value obtained at the time of detection and the detection position are displayed and F5 [NEXT] key is enabled.
- 6 If you press F5 [NEXT] key, a screen as shown below is displayed.



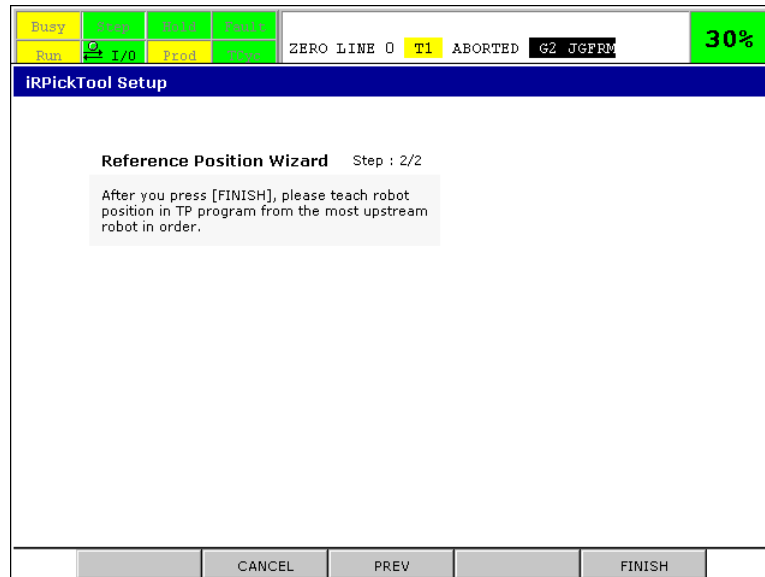
- 7 Press F5 [FINISH] key. This completes the reference position setting procedure.
- 8 Next, perform robot position teaching. (See Section 4.11, "TEACHING THE POSITION TO ROBOTS".)

4.10.2 When [DI] and [Find part by vision] Are Selected

- 1 Stop the conveyor.
- 2 Press F3[SET REF POS] key A screen as shown below is displayed.



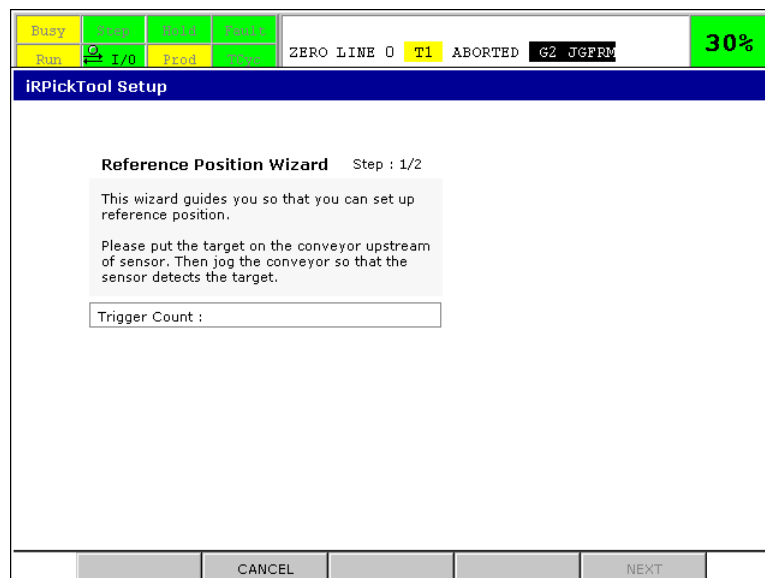
- 3 Let the part travel from the most upstream area of the conveyor. When the conveyor moves by the trigger offset after the photo eye spots the part, iRVision automatically detects the part. The conveyor does not need to be stopped for iRVision to detect the part. Move the conveyor until the most upstream robot operation range is reached and stop it. When iRVision successfully detects the part, the detection position are displayed and F5 [NEXT] key is enabled.
- 4 If you press F5 [NEXT] key, a screen as shown below is displayed.



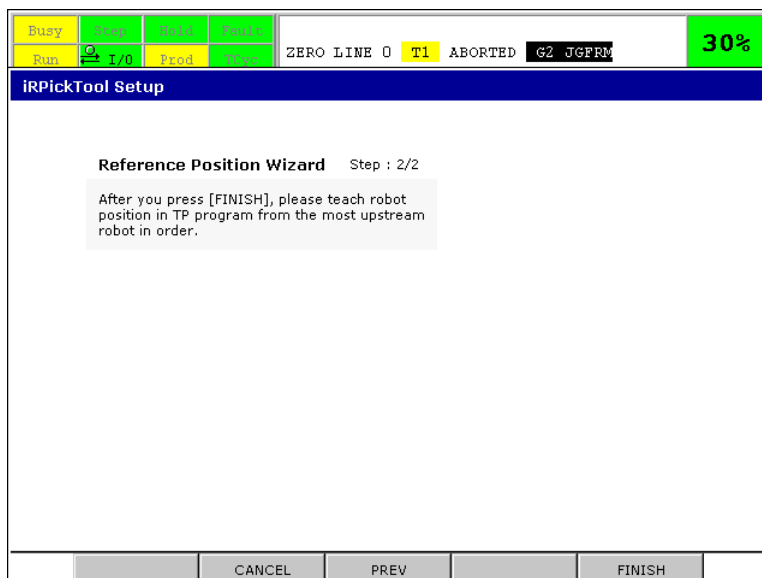
- 5 Press F5 [FINISH] key. This completes the reference position setting procedure.
- 6 Next, perform robot position teaching. (See Section 4.11, "TEACHING THE POSITION TO ROBOTS".)

4.10.3 When [DI] or [HDI] and [Put part in queue] Are Selected

- 1 Stop the conveyor.
- 2 Press F3[SET REF POS] key A screen as shown below is displayed.



- 3 Place a part upstream of the photo eye. Move the conveyor and let the part pass in front of the photo eye. The conveyor does not need to be stopped for the photo eye to detect the part. Move the conveyor until the most upstream robot operation range is reached and stop it. When the photo eye successfully detects the part, the encoder value obtained at the time of detection is displayed and F5 [NEXT] key is enabled.
- 4 If you press F5 [NEXT] key, a screen as shown below is displayed.



- 5 Press F5 [FINISH] key. This completes the reference position setting procedure.
- 6 Next, perform robot position teaching. (See Section 4.11, "TEACHING THE POSITION TO ROBOTS".)

4.10.4 When [Distance] or [Flag] and [Put part in queue] Are Selected

Since a part is not actually detected for this combination, set the reference position at the same time you set the target position. For details, see Subsection 4.5.6, "Setting the Target Position".

4.11 TEACHING THE POSITION TO ROBOTS

After setting the reference position, teach the position to robots.

4.11.1 Conveyor Station

- 1 When the part enters the tracking area after the reference position is set in the sensor wizard screen, stop the conveyor.
- 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 positions for cell 2 and subsequent cells are automatically calculated based on the tray pattern, and the vision offset instruction moves the robot to each cell based on the offset value stored in the 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 "R-30iB/R-30iB Mate CONTROLLER OPERATOR'S MANUAL (Basic Operation)".

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.

4.11.2 Fixed Station

Teach the placement (or pickup) position only for cell 1.

The position of every other cell is created relative to cell 1 by using the vision offset instruction.

NOTE

Cell 1 does not necessarily need to be taught for the user frame representing the tray frame.

4.12 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.

4.12.1 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, "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.

4.12.2 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 TCP. You can improve the situation, however, by adding an amount of adjustment to the vision offset calculated by *iR*Vision 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.

4. SETUP OF THE BASIC FUNCTIONS

B-83604EN/04

- 1 Add the line indicated below after PKCSGETQUE is succeeded in the tracking program.

Example: PICK1.TP in Subsection 4.9.2.2, "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 "iRVision 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 *iR*PickTool, ADJ_OFS is available as a standard feature even if the vision support tool option is not ordered.

4.12.3 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 4.12.1 and 4.12.2.
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: PICK1.TP in Subsection 4.9.2.2, "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 4.2, "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:

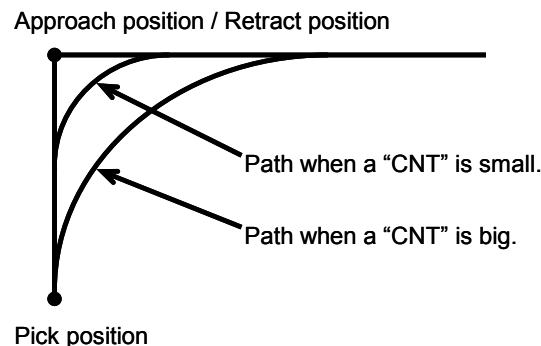
$$[\text{Dynamic Error Adjustment}] = +30 \text{ (ms)} + (3 \text{ (mm)} \div 500 \text{ (mm/s)}) \times 1000 = +36 \text{ (ms)}$$

4.12.4 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.

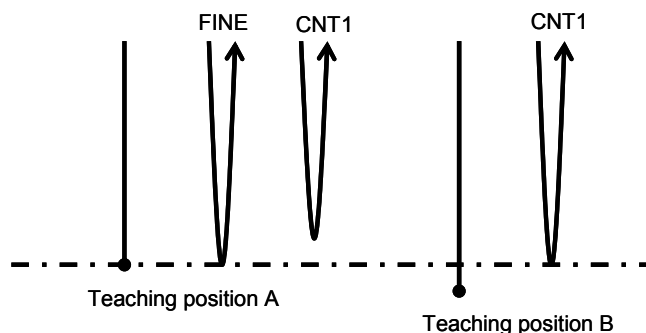


Pickup position and placement position

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.

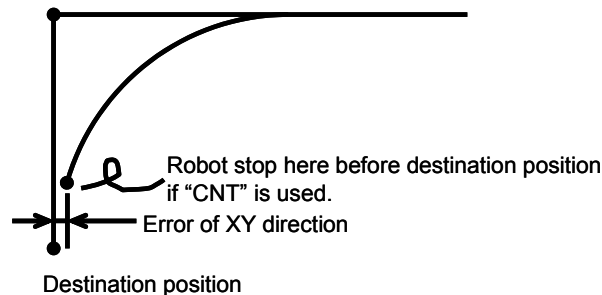
Note, however, that if CNT is specified, the robot moves slightly inward even when the value is small. This causes the robot to start moving toward the retract point before it arrives at the taught pickup position (or placement position). In the case of taught position A in the figure below, for example, 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.



4.12.5 Linear Distance Specification

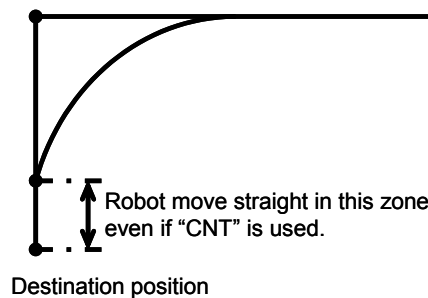
If a large CNT value is specified for the approach point or retract point and CNT is specified for the pickup position (or placement position) instead of FINE, the robot may fail to reach the destination position not only in height but also in the X and Y directions. The remaining distance appears to be an extra handling error.



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.



An example of the modified program is shown below. This is part of the program described in Subsection 4.9.2.2, "Pick program".

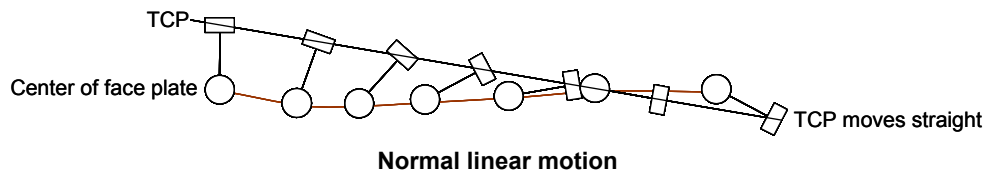
```
13:L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
14:L PR[R[3:ZONE]] 4000mm/sec CNT0 VOFFSET,VR[1] AP_LD10
15: CALL VACUUM_ON
16:L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10] RT_LD10
```

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.

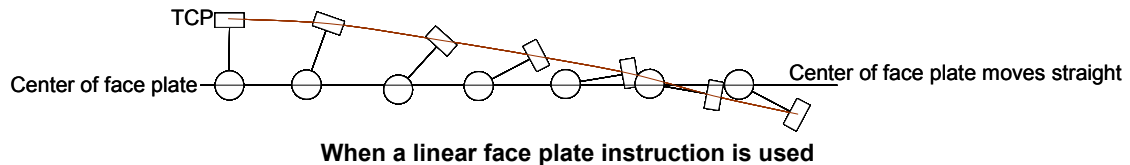
4.12.6 Linear Face Plate Instruction

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.



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.



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 4.9.2.2, "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.

4.12.7 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 iRPickTool 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 4.9.3.2, "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.

4.12.8 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 4.9.2.2, "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.

4.13 SYSTEM STARTUP

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.

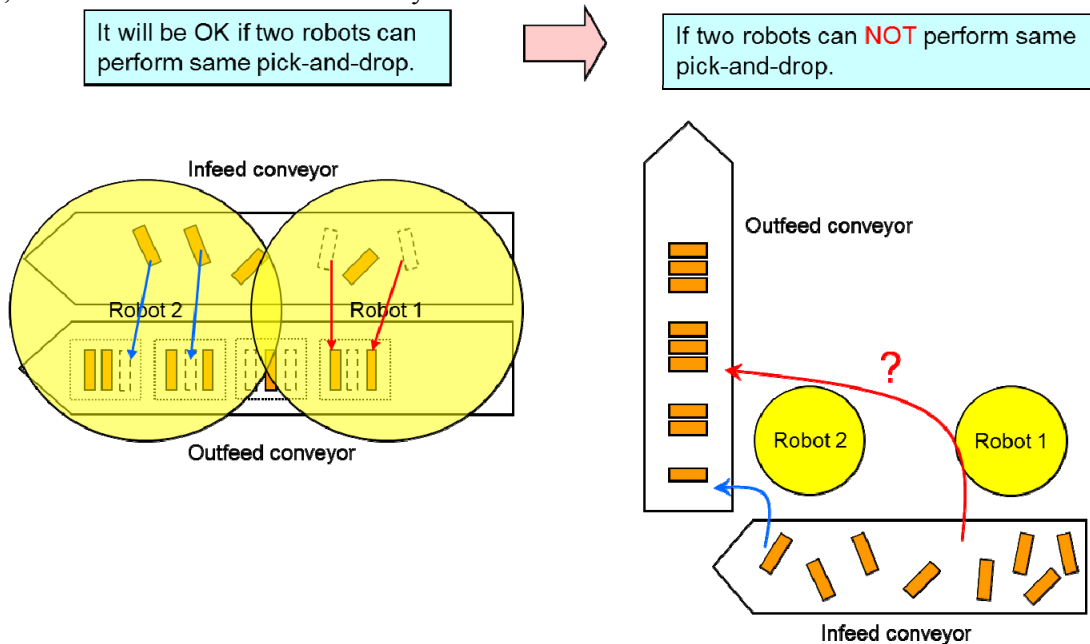
If there is more than one robot controller that executes the sensor task, insert a wait instruction before the start of the sensor task so that the sensor task is started after part information is cleared from all conveyor stations, as described in Subsection 4.9.2.1, "Main program".

4.14 SETUP OF PRE-GROUPING

This section describes the setup procedure of the pre-grouping function which arranges parts on the same conveyor. 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:

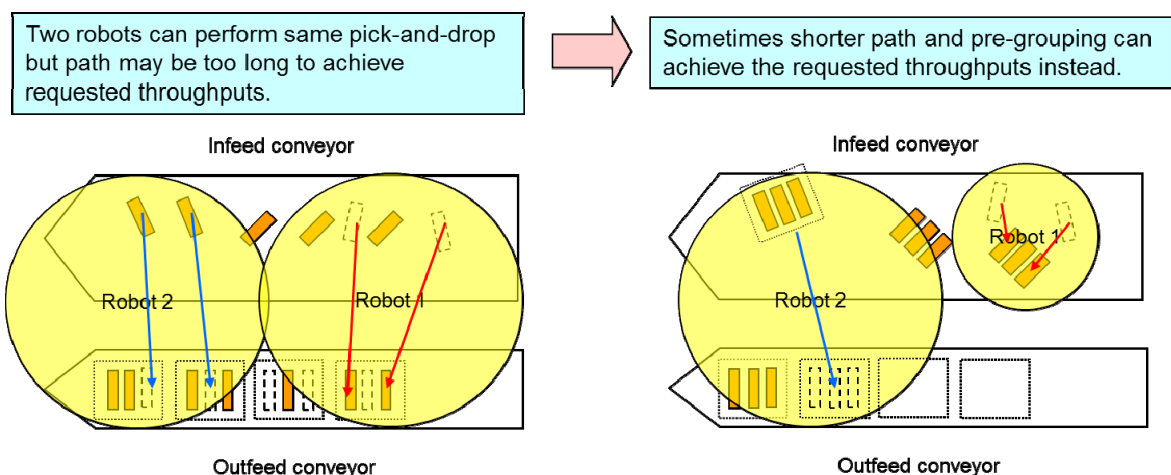
Example 1

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.

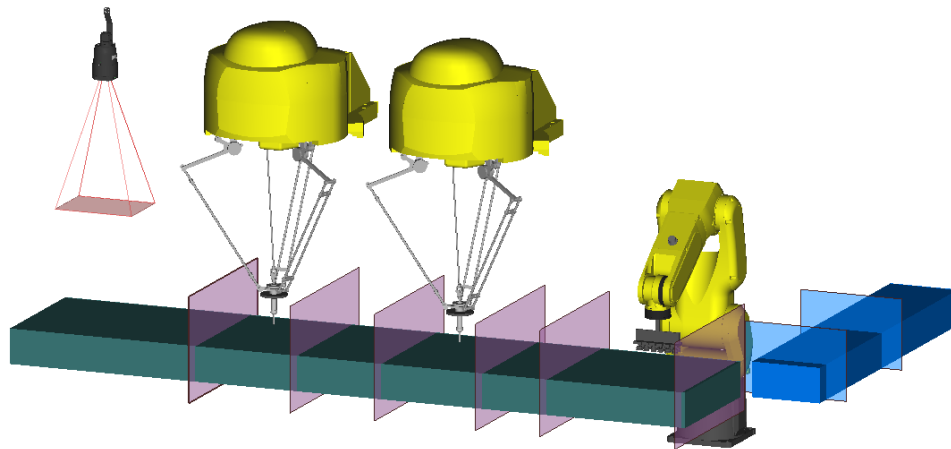


Example 2

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.



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.



The system configuration and tree view composition in this case are shown in Fig. 4.14 (a).

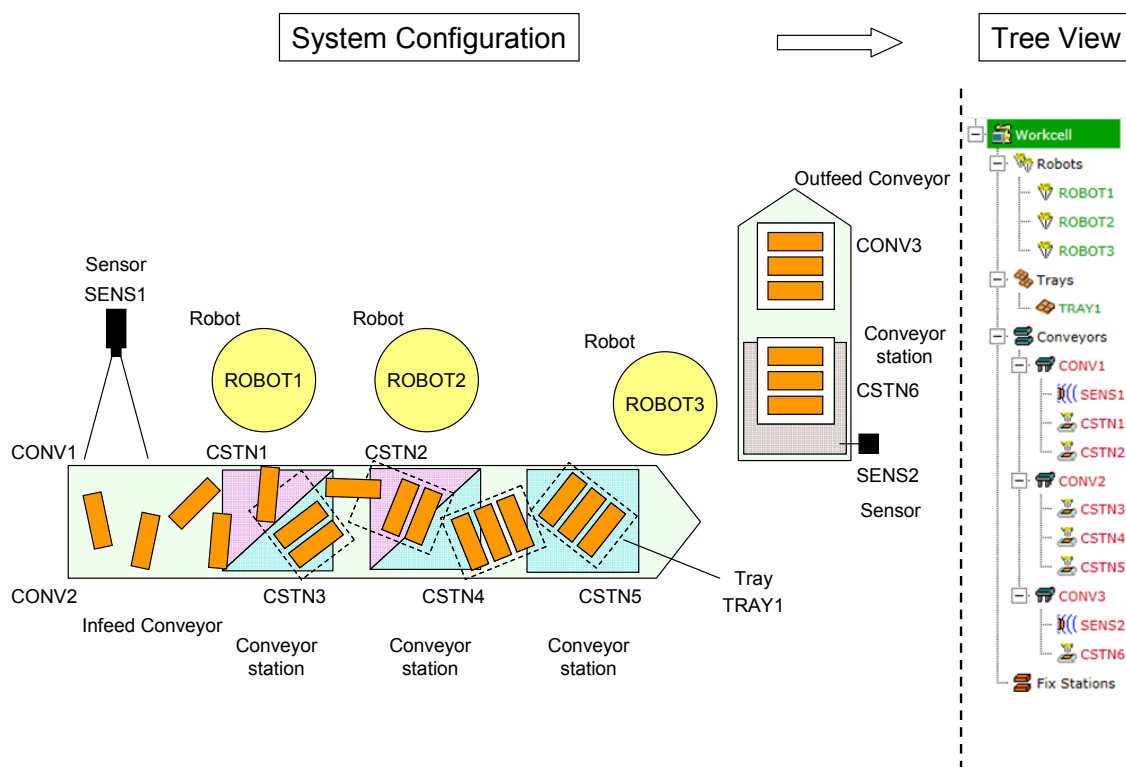


Fig. 4.14 (a)

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.4.14 (a) 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.

4.14.1 Key Concepts

This section describes terms related to a pre-grouping function.

Infeed Conveyor

This is a conveyor object for picking operation.

In Fig. 4.14 (a), CONV1 is an infeed conveyor.

Pre-grouping Conveyor

This is a conveyor object for arranged placement or pre-grouping operation. In Fig. 4.14 (a), 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.

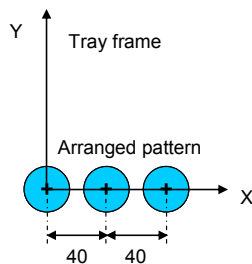
4.14.2 Setup of a Workcell and a Robot

The setup of the Workcell and the Robot are done as usual. There are no special steps for pre-grouping. Refer to 4.1, "SETUP OF A WORKCELL" and 4.2, "SETUP OF A ROBOT".

4.14.3 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.



Busy Stop Hold Fault
Run I/O Prod Stop

T1 JOINT 10%

iRPickTool Setup

Workcell

- Robots
 - ROBOT1
 - ROBOT2
 - ROBOT3
- Trays
 - TRAY1
- Conveyors
- Fix Stations

System Setup

Layer number: 1

Cell	X(mm)	Y(mm)	Z(mm)	R(deg)	Model ID
1	0.0	0.0	0.0	0.0	1
2	40.0	0.0	0.0	0.0	1
3	80.0	0.0	0.0	0.0	1

Edit Cell 3

X(mm)	Y(mm)	Z(mm)	R(deg)	Model ID
80.0	0.0	0.0	0.0	1

[TYPE] [EDIT TREE] [CELL] [LAYER] LOGOUT >

**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.

- 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.

Busy Stop Hold Fault
Run I/O Prod Stop

T1 JOINT 10%

iRPickTool Setup

Workcell

- Robots
 - ROBOT1
 - ROBOT2
 - ROBOT3
- Trays
 - TRAY1
- Conveyors
- Fix Stations

System Setup

Layer number: 2

Cell	X(mm)	Y(mm)	Z(mm)	R(deg)	Model ID
4	0.0	0.0	0.0	0.0	2

Edit Cell 4

X(mm)	Y(mm)	Z(mm)	R(deg)	Model ID
0.0	0.0	0.0	0.0	2

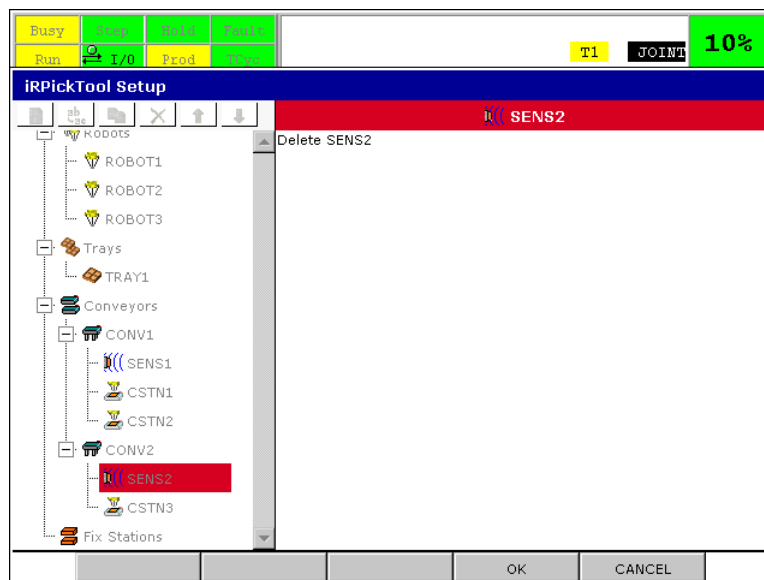
[TYPE] [EDIT TREE] [CELL] [LAYER] LOGOUT >

4.14.4 Setup of a Conveyor

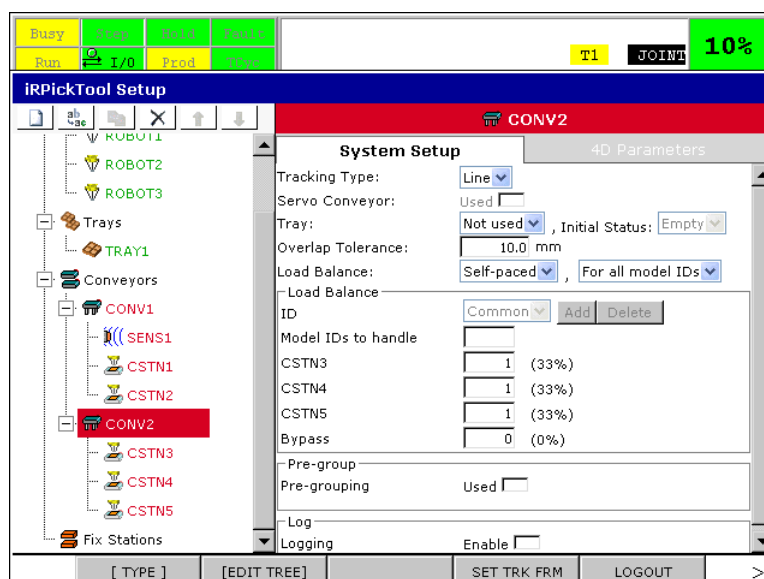
In pre-grouping you always setup two conveyors: the infeed conveyor and the pre-grouping conveyor. In the real-world, both the conveyors are the same physical conveyors. These two conveyors use the same tracking frame. However, here is no need to teach the tracking frame for each conveyor. It is sufficient to teach it for the pre-grouping conveyor.

First, the object of an infeed conveyor is set up. Refer to 4.4 "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.

Next, 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.



Then, add a required number of conveyor station objects.

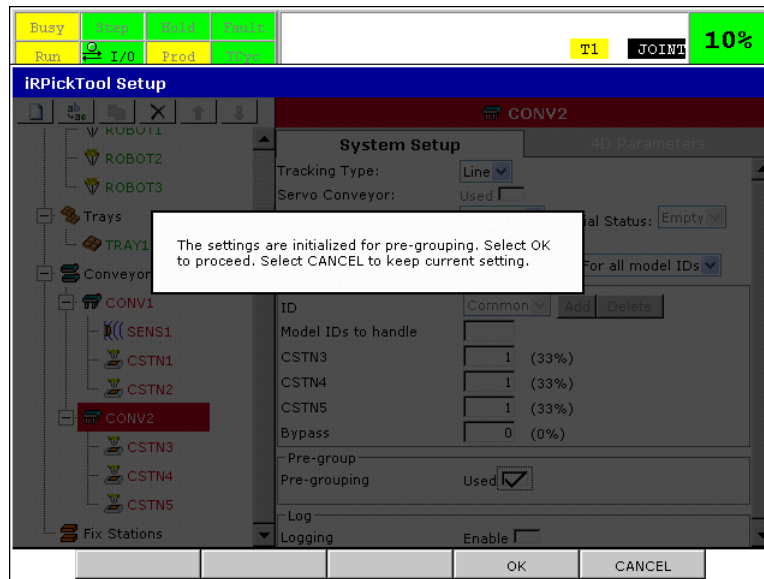


Check the [Used] checkbox of the [pre-grouping] in the [Pre-group] item.

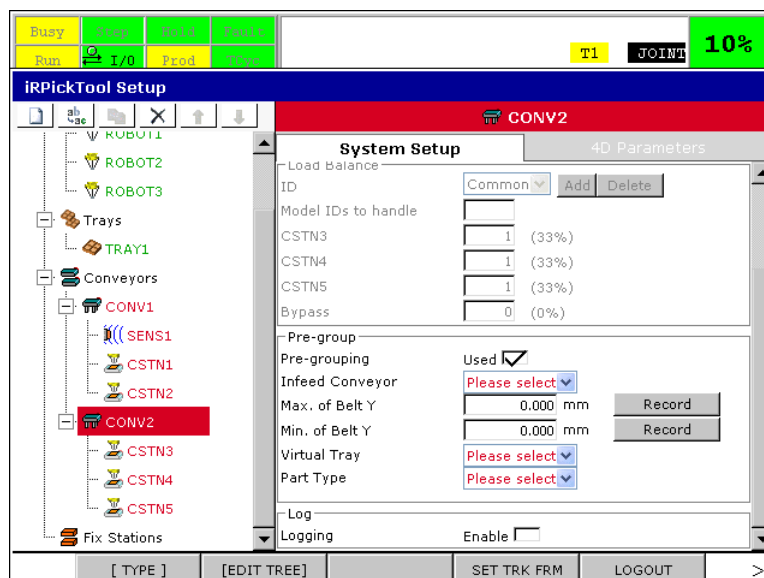
NOTE

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.



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.

Set up other items than a [Pre-grouping] according to 4.4 "SETUP OF A CONVEYOR".

Pre-group

Pre-group		
Pre-grouping	Used ☒	
Infeed Conveyor	Please select ▼	
Max. of Belt Y	0.000 mm	Record
Min. of Belt Y	0.000 mm	Record
Virtual Tray	Please select ▼	
Part Type	Please select ▼	

Infeed Conveyor

Select the infeed conveyor object.

NOTE

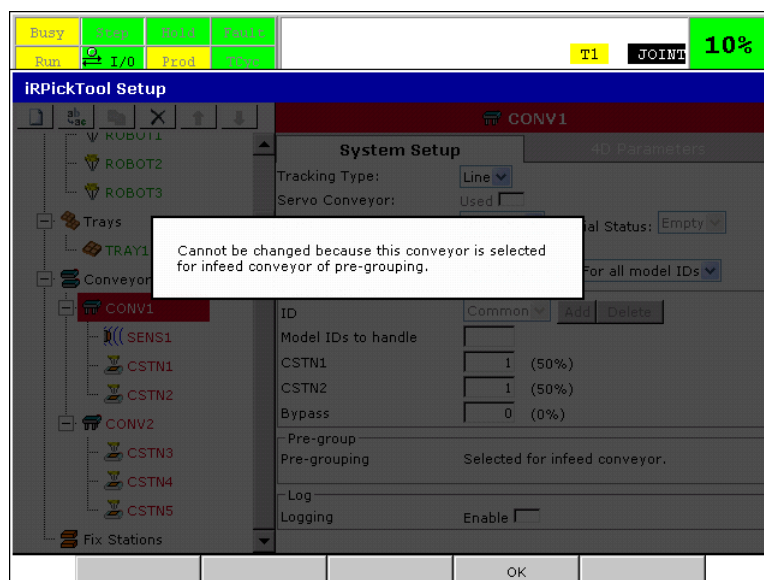
Only the conveyor objects, which meet the following conditions, are displayed as options of the [Infeed Conveyor].

- 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.

Pre-group	
Pre-grouping	Selected for infeed conveyor.

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 4.4.1 "Setup of Tracking Frame" in detail. More concretely, follow the procedure after item 3 mentioned in next 4.4.1.1 "Line Conveyor".

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.

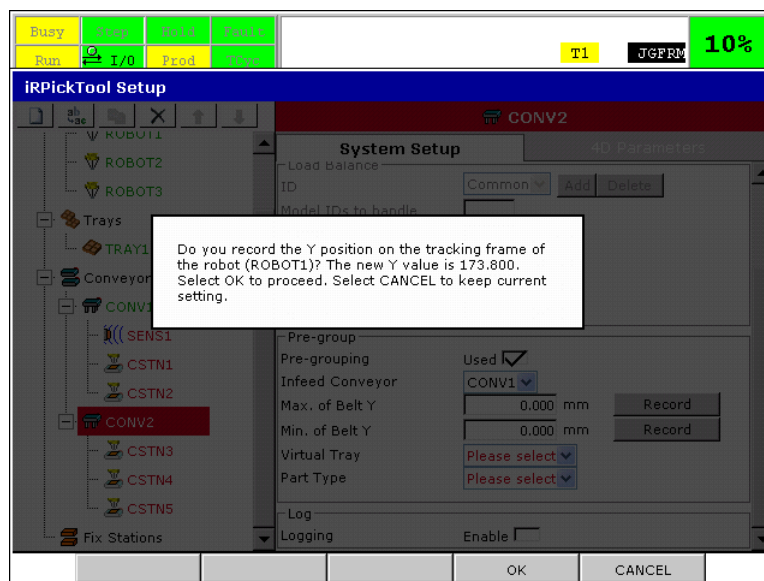
Select the tool frame set up by "3.4 SETTING OF THE TCP" as a jog frame and move the TCP by jog and tap the [Record] button, then the Y value in a tracking frame will be set. For example, touch up the

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 iRPickTool treeview.

After you tap of the [Record] button, the Y value of a current robot position in the tracking frame will be displayed.



Press F4 [OK] key to set the new value.

Press F5 [CANCEL] key to keep the previous value.

NOTE

If you press the [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

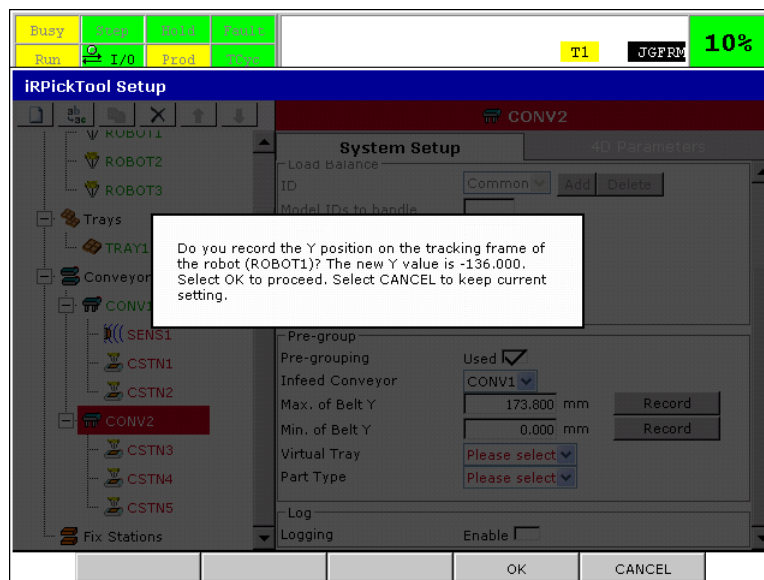
A virtual tray is generated so that not the model origin point of the vision but whole parts may not overrun this lower limit.

Select the tool frame set up by 3.4 "SETTING OF THE TCP" 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 *iRPickTool* treeview.

After you tap of the [Record] button, the Y value of a current robot position in the tracking frame will be displayed.



Press F4 [OK] key to set the new value.

Press F5 [CANCEL] key to keep the previous value.

NOTE

If you press the [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)],

Pre-group		
Pre-grouping	Used ☒	
Infeed Conveyor	CONV1	
Max. of Belt Y	173.800 mm	Record
Min. of Belt Y	-136.000 mm	Record
Virtual Tray	TRAY1	
Part Type	Circle (without Phase)	
Part Parameter	Radius 0.0 mm	
Orientation	Enable ☐	
	Minimum -90.0 °, Maximum 90.0 °	
Interval of Angle	Auto, 30.0 °	

Part Parameter

Set the radius of a circle to the [Radius].

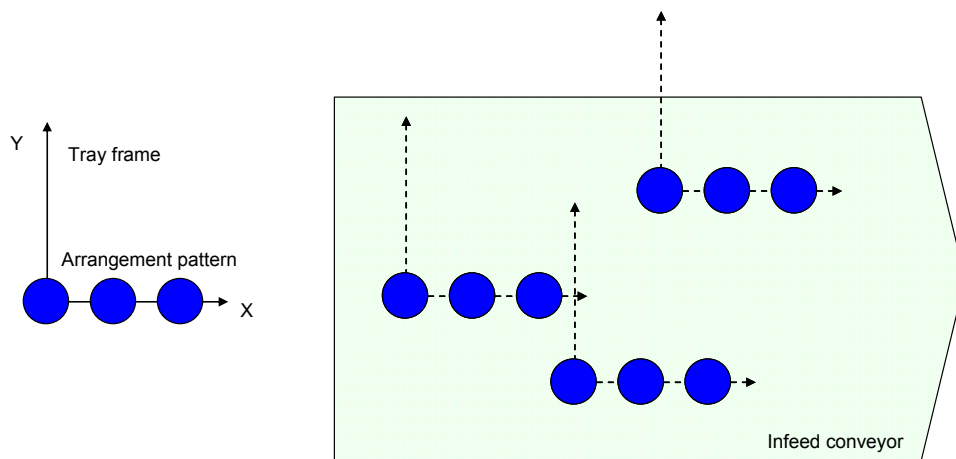


CAUTION

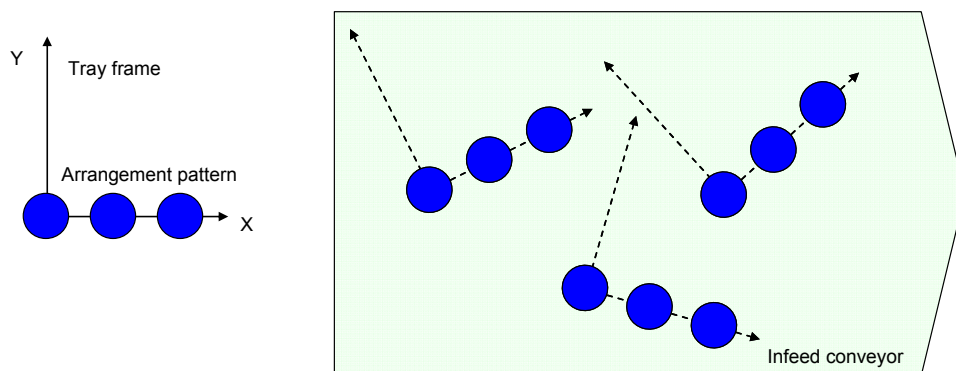
If the radius value is smaller than the actual one, the part which the robot picks and places for arrangement may collide with the part on a conveyor due to the incorrect calculation of the interference check between both kinds of the parts. When the form of the actual part is not the circle, you must set the radius of the circle which includes whole of the part.

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.



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].



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],

Pre-group	Used ☒	
Pre-grouping	CONV1	
Infeed Conveyor		
Max. of Belt Y	173.800 mm	Record
Min. of Belt Y	-136.000 mm	Record
Virtual Tray	TRAY1	
Part Type	Rectangle	
Part Parameter	X length 0.0 mm	
	Y length 0.0 mm	

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 uncorrect 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.

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 4.4 "SETUP OF A CONVEYOR" for the other items.

4.14.5 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 4.5 "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 4.5.1 "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 uncorrect calculation of the interference check between both kinds of parts. When you refer to 4.5.3.1 "When a tray is not used" under 4.5.3 "Setting the Sensor Position" and 4.5.4.5 "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 uncorrect 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 uncorrect calculation of the interference check between both kinds of the parts.

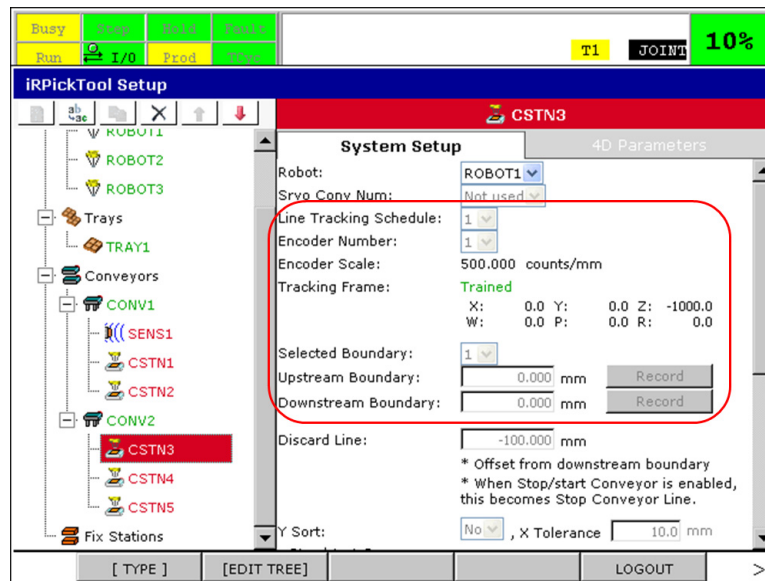
Additionally, if the position is not detected by the vision and the [Rectangle] is selected as the [Part Type] at the pre-grouping conveyor, you should give attention to the order of settings. Setting of the sensor position or the target position must be performed after setting of the [Virtual Tray Position] 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.

4.14.6 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 4.6 "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.

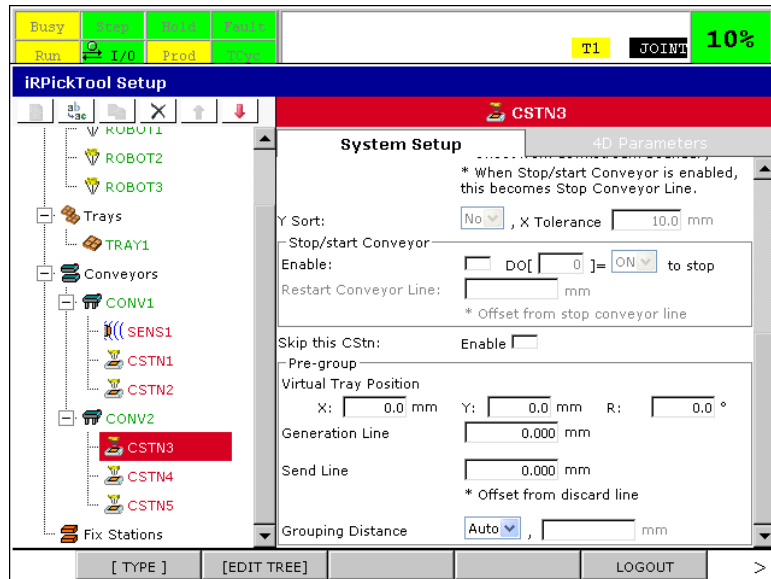
Skip Station

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 4.5.5 "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 4.5.3.2 "When a tray is used" under "4.5.3 Setting the Sensor Position". When the [Distance] is set as the [Trigger Action], set the target position. Refer to 4.5.6.2 "When a tray is used" under "4.5.6 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 wizard 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.

Generation 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 length] of [Part Parameter] and the value(y) set at [Y length] of [Part Parameter] as shown in Fig. 4.14 (b).

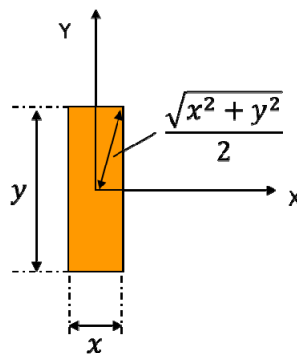
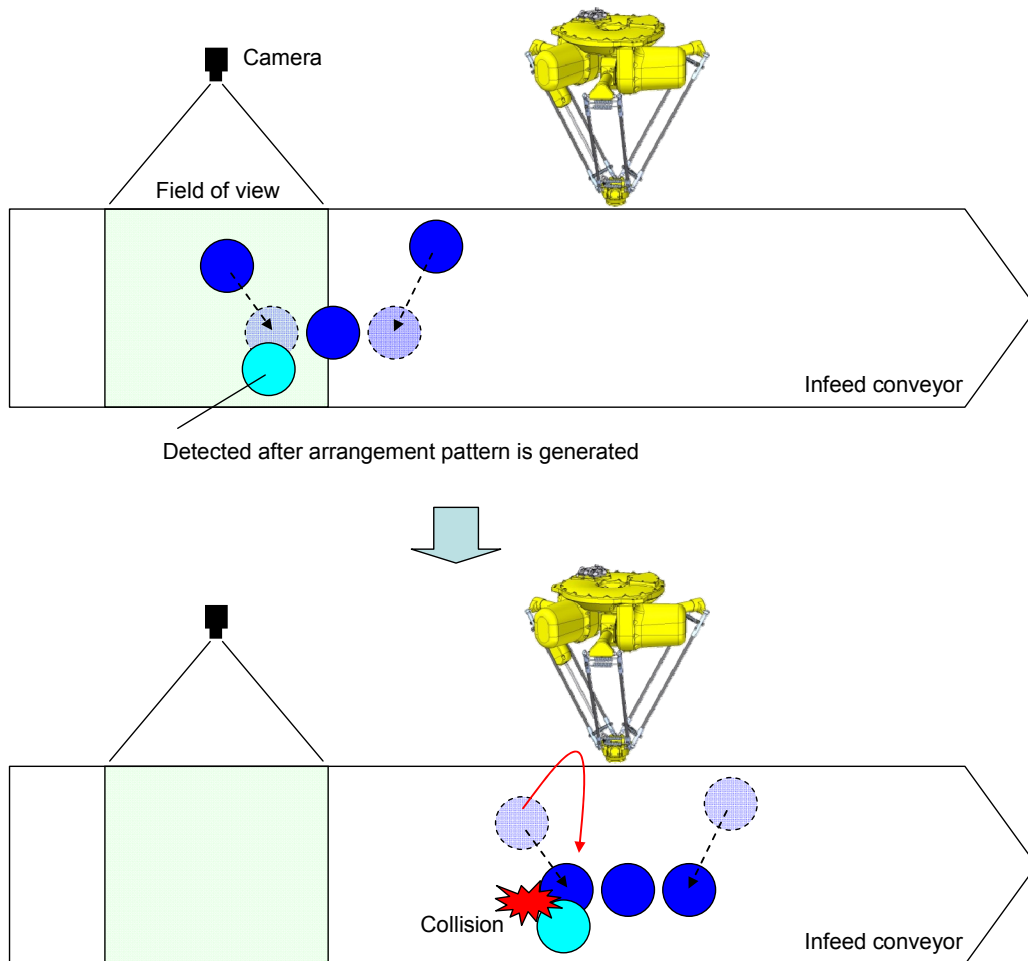


Fig. 4.14 (b)

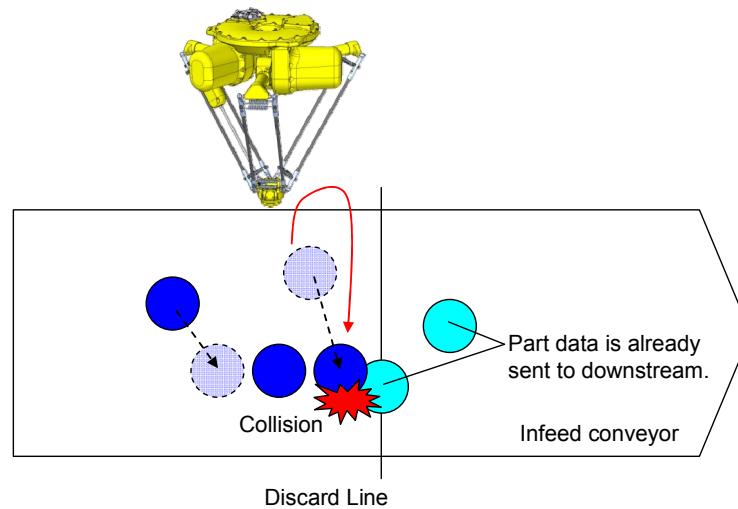
⚠ CAUTION

- 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.

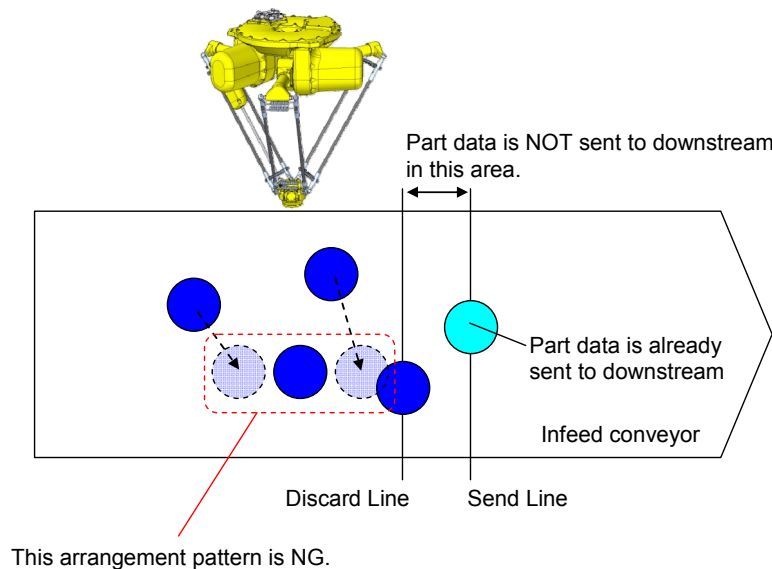


Send Line

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.



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.



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. 4.14 (b).



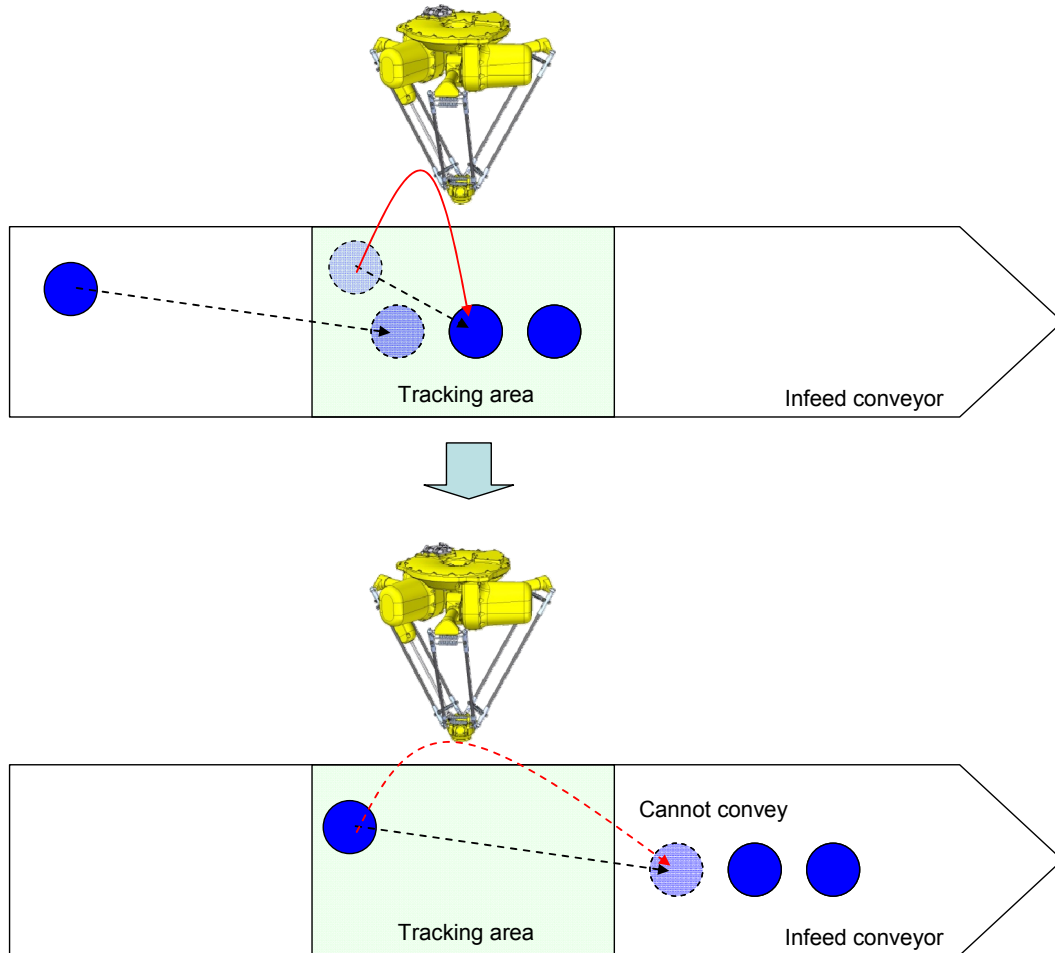
CAUTION

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.

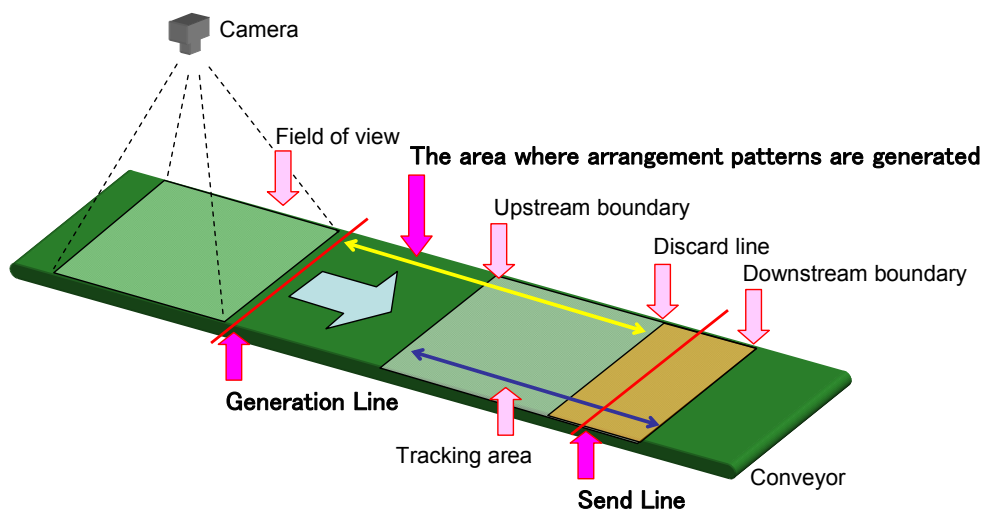
If parts which constitutes a virtual tray separates too much each other about the direction of movement of a conveyor, even if the robot picks up a part for arrangement, the corresponding placement position can already pass a tracking area and the robot cannot place the part.



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"



4.14.7 Setup of a Fixed Station

When you use a fixed station, refer to 4.7 "SETUP OF A FIXED STATION."

4.14.8 KAREL Programs

This subsection explains the KAREL programs for a pre-grouping function. All of them are related to a conveyor station.

4.14.8.1 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 belong 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)
```

NOTE

If you specify Argument 3 (model ID), the argument guidance is not displayed.

4.14.8.2 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 belong 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)
```

4.14.8.3 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)

4.14.8.4 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.

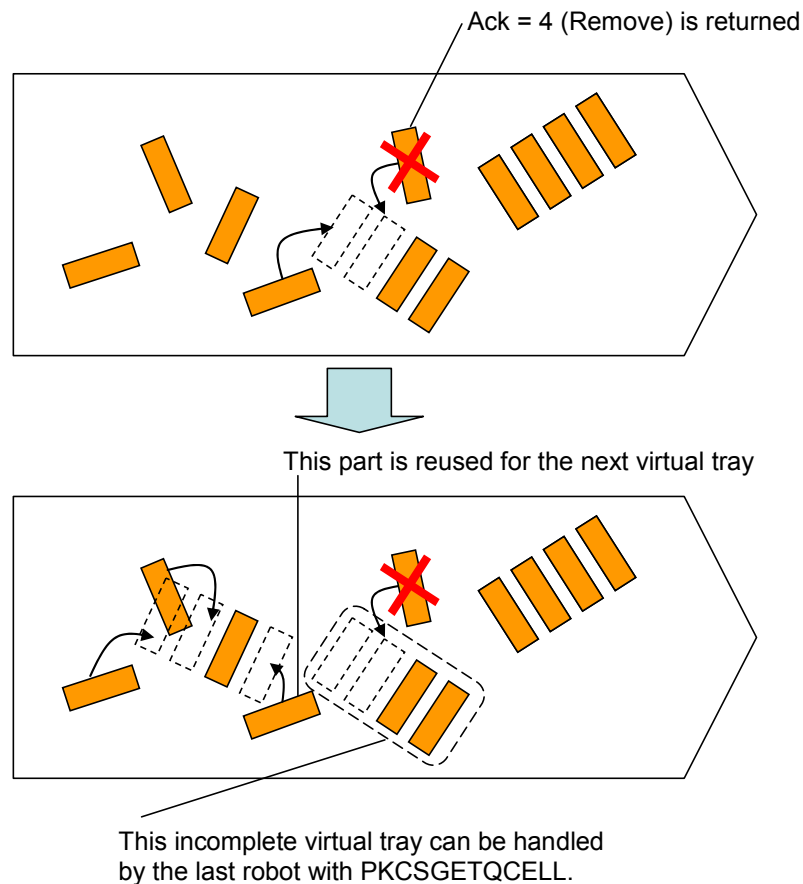
Argument 1:

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 belong 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.

**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)
```

4.14.9 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.

4.14.9.1 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.

4. SETUP OF THE BASIC FUNCTIONS

B-83604EN/04

The sample programs presented herein use the registers and Position Registers listed below. Change the register numbers as necessary.

Table 4.14.9.1(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 4.14.9.1(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]

```

Description

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 4.9.3 "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]

```

Description

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.

4.14.9.2 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 "4.14.9.1 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]
```

Description

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 "4.14.9.1 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. DROPFS is called on line 17 to place the picked part on the fixed station. Refer to 4.9.3 "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.

4.14.9.3 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 different from the sample programs showed in 4.9.2 "Sample Programs for the Conveyor-to-Conveyor Configuration". Refer to 4.9.2.1 "Main program" and 4.9.2.3 "Placement program" for the main program of the system and the program which performs placement motion on an outfeed conveyor, respectively.

Pick Program

First, this is a program which performs a pickup motion in the case of picking only completely arranged parts.

PICK1.TP

```

1:  UTOOL_NUM=1
2:  UFRAME_NUM=0
3:
4:  LBL[100]
5:  STOP_TRACKING
6:  CALL PKCSGETQUE(R[11:CStnIDIn], 1, 100, 1, 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]

```

Description

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. Because the argument 6 (=model ID) is specified in PKCSGETQUE, the argument guidance is not displayed on line 6.

Next, this is a program which performs a pickup motion in the case of picking incompletely arranged parts too.

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 Req=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]

```


Description

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].

4.14.10 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 4.10 "REFERENCE POSITION SETTING".

4.14.11 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 4.11 "TEACHING THE POSITION TO ROBOTS".

4.14.12 Fine Adjustment of The Tracking Motion

Refer to 4.12 "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.

4.14.12.1 Adjustment of the tracking frame error

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.

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

```

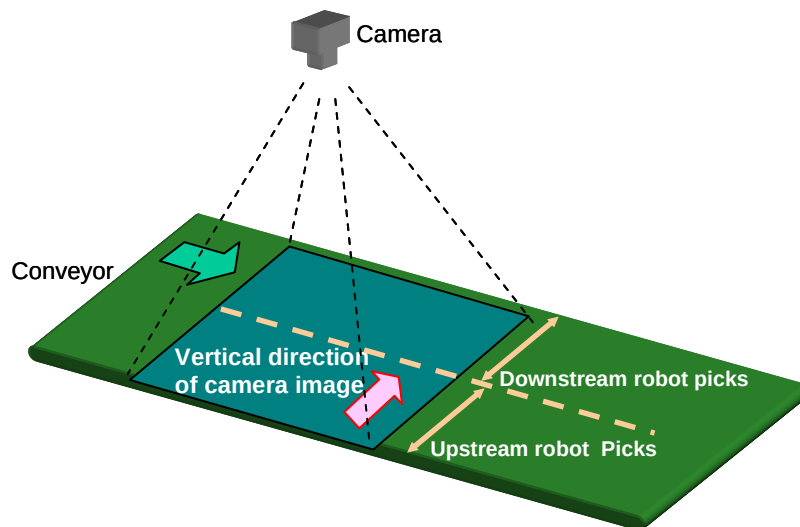
- Refer to 4.12.2 "Adjustment of the Tracking Frame Error" for details.

5 VARIATIONS

This chapter presents some variations of the system that are not discussed in Chapter 4.

5.1 USING DIFFERENT ROBOTS TO PICK PARTS DEPENDING ON THEIR POSITIONS ON THE CONVEYOR

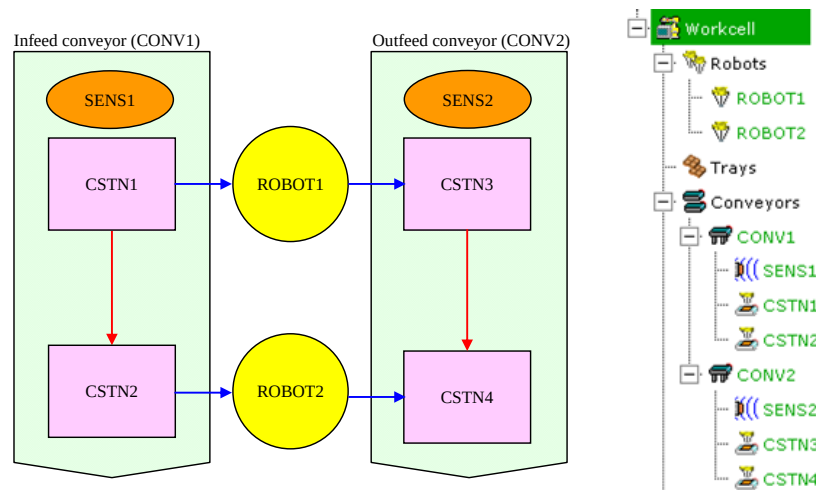
This section describes the case in which different robots are used to pick parts depending on their positions on the conveyor. In the example presented here, there are two robots, and the upstream robot picks parts on the right half of the conveyor while the downstream robot picks parts on the left half. A camera is used to detect parts on the conveyor.



In addition, the following assumptions are made.

- 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 iRVision, 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 value of the detected Vt coordinate on the image is larger than 240, the model ID is set to 2 with the conditional execution tool. (When a digital camera is used, the Vt coordinate value at the center of the conveyor is half the vertical direction size of [Mode] of Kowa Digital Camera setup page. When 1/3" VGA (640x480) is selected, the value is 240. In the case of an analog camera, the value is half the vertical direction size set in the camera parameter [Image Size].)

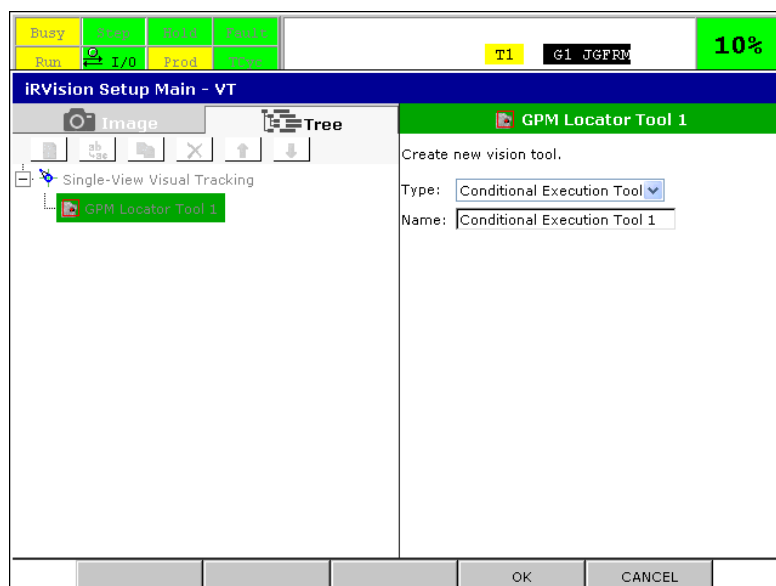
- The conveyor configuration is defined as shown below.
This is the same configuration as Configuration 1 shown in Section 2.4, "SAMPLE SYSTEM CONFIGURATIONS".



Inserting the conditional execution tool

Insert the conditional execution tool into the vision process.

- In the tree view of the edit screen, select [GPM Locator Tool 1] and click the  button. A screen as shown below is displayed.

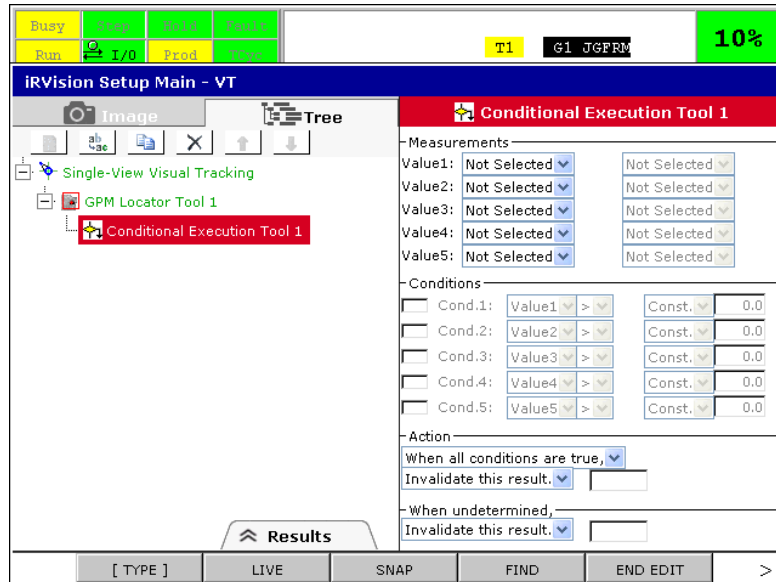


- In [Type], select [Conditional Execution Tool].
- In [Name], enter a tool name (e.g., "Conditional Execution Tool 1"), and press F4 [OK] key.

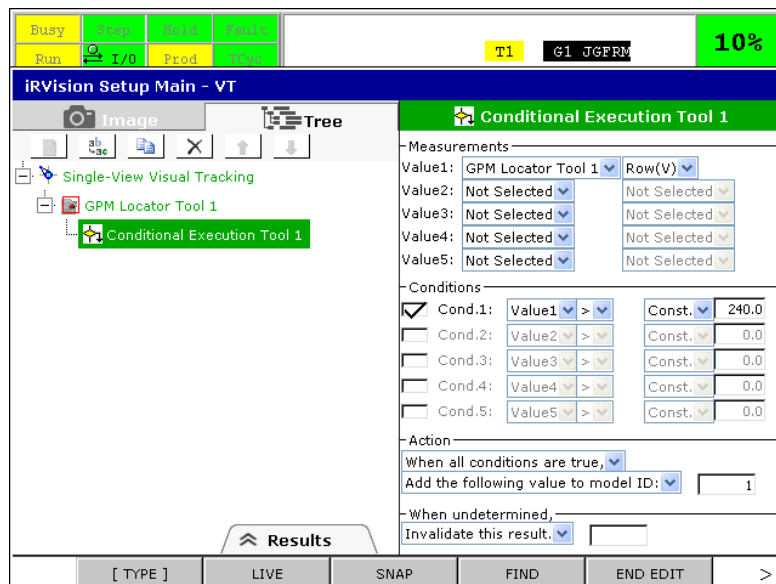
Setting the conditional execution tool

Set the conditional execution tool to add 1 to the model ID if the value of Vt is larger than 240.

- 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.

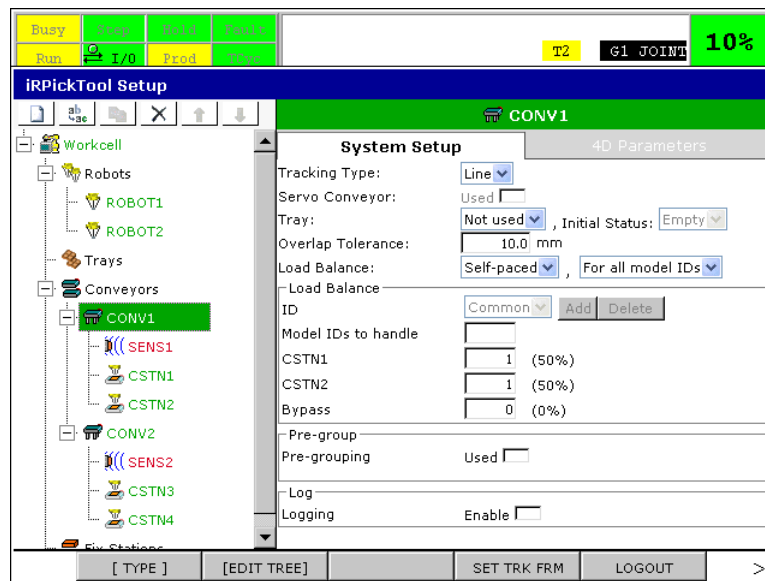


Setting the load balance

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.

Load Balance: Self-paced , For each model ID

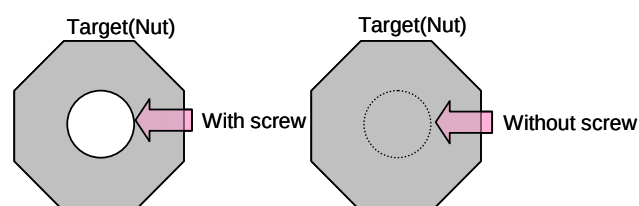
- 3 In [Model IDs to handle], specify 1 and 2.
- 4 Set the load balance so that the upstream robot handles all parts whose model ID is 1 and the downstream robot handles all parts whose model ID is 2.

Load Balance			
ID	1	Add	Delete
Model IDs to handle	1		
CSTN1	1	(100%)	
CSTN2	0	(0%)	
Bypass	0	(0%)	

Load Balance			
ID	2	Add	Delete
Model IDs to handle	2		
CSTN1	0	(0%)	
CSTN2	1	(100%)	
Bypass	0	(0%)	

5.2 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.



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

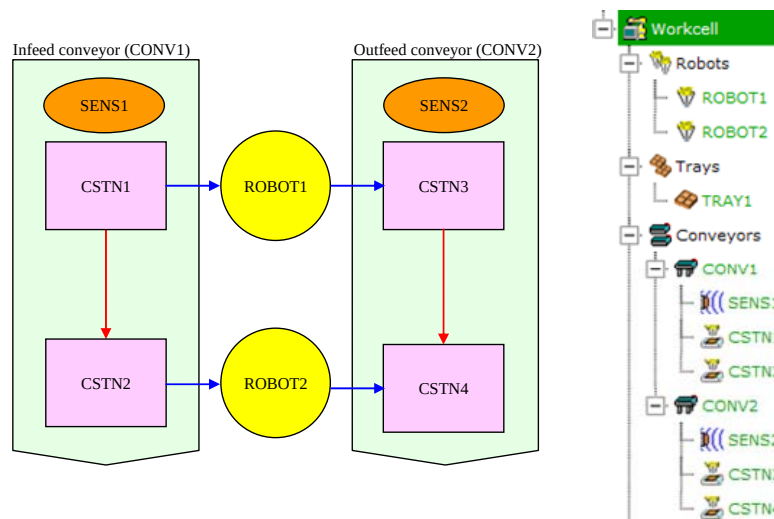
**CAUTION**

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.

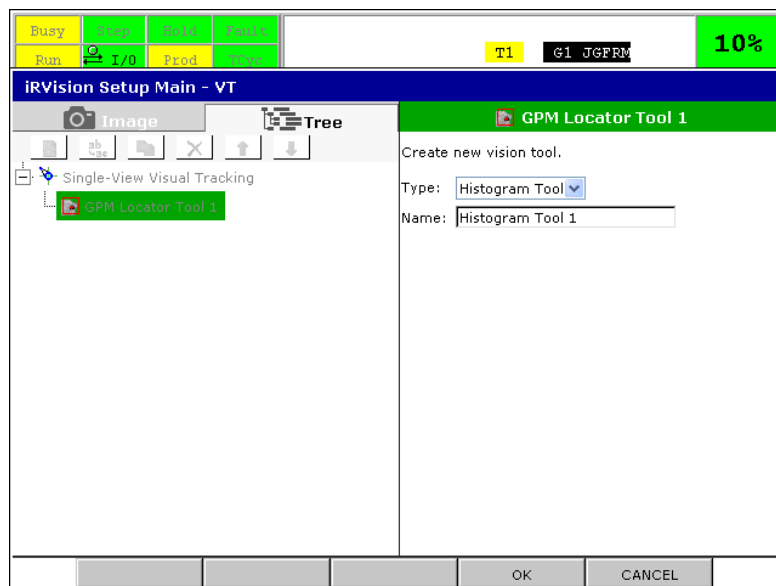
- The part model ID taught to the GPM locator tool is 1.
- If a hole is present, the model ID is set to 2 using the conditional execution tool.
- The conveyor configuration is defined as shown below.
This is the same configuration as Configuration 1 shown in Section 2.4, "SAMPLE SYSTEM CONFIGURATIONS".



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.

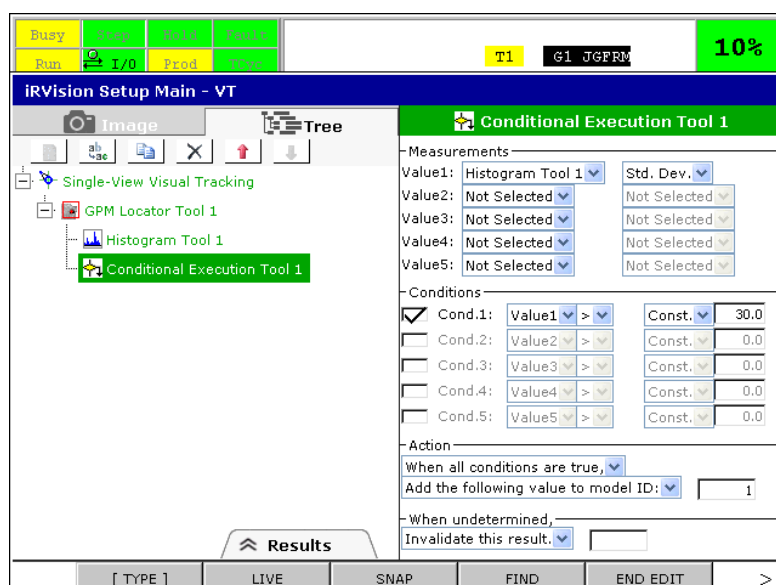


- 2 In [Type], select [Histogram Tool].
- 3 In [Name], enter a tool name (e.g., "Histogram Tool 1"), and press F4 [OK] key.
- 4 In the tree view, select [Histogram Tool 1].
- 5 Click the [SET] button to teach [Measurement Area].
- 6 Press F10 [SAVE] key.

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 5.1, "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.



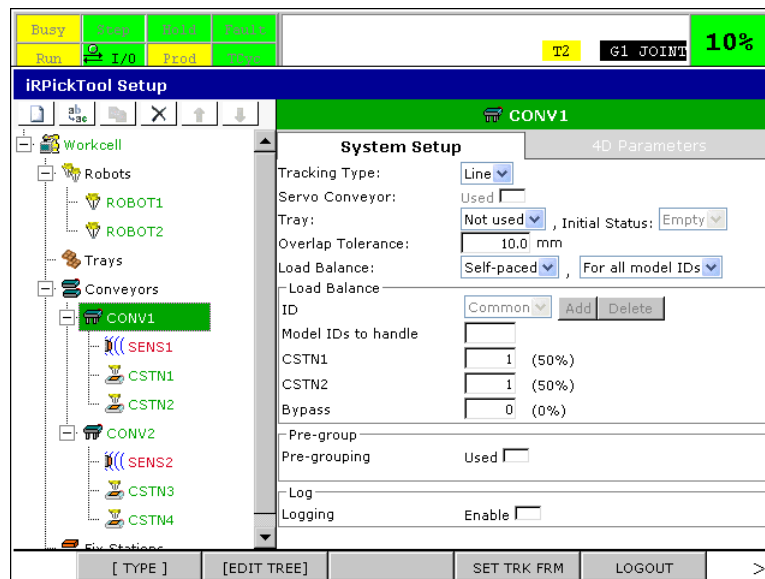
- 1 Under [Measurements], select [Histogram Tool 1] and [Std. Dev.] for [値 1].
- 2 Under [Conditions], check [Cond.1] and then select [Value1], [>], [Const.], and [30.0].
- 3 Under [Action], select [Add the following value to model ID] and specify 1.

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.

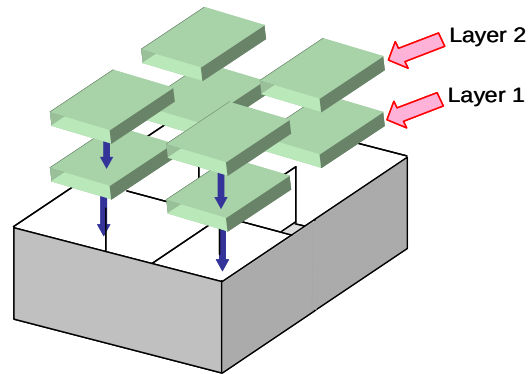
Load Balance: Self-paced , For each model ID

- 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.

Load Balance		Load Balance	
ID	1	ID	2
Model IDs to handle	1	Model IDs to handle	2
CSTN1	1 (100%)	CSTN1	0 (0%)
CSTN2	0 (0%)	CSTN2	1 (100%)
Bypass	0 (0%)	Bypass	0 (0%)

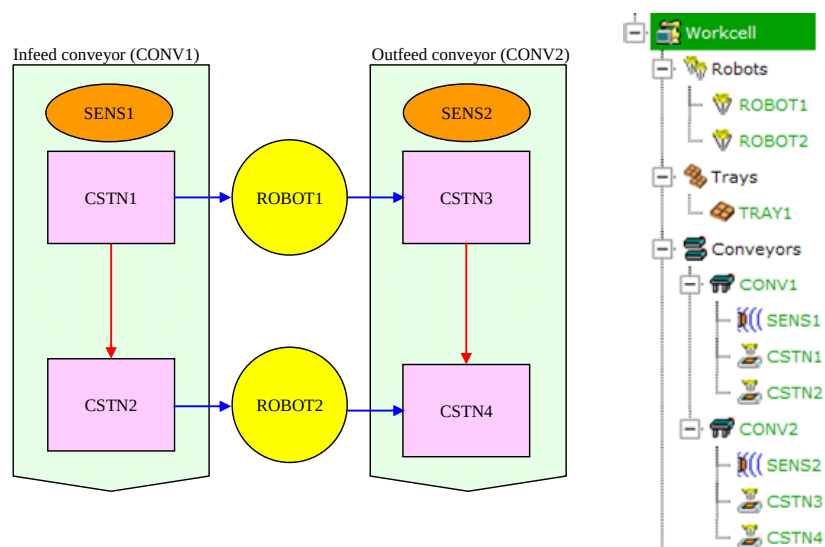
5.3 PLACING PARTS IN LAYERS

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.



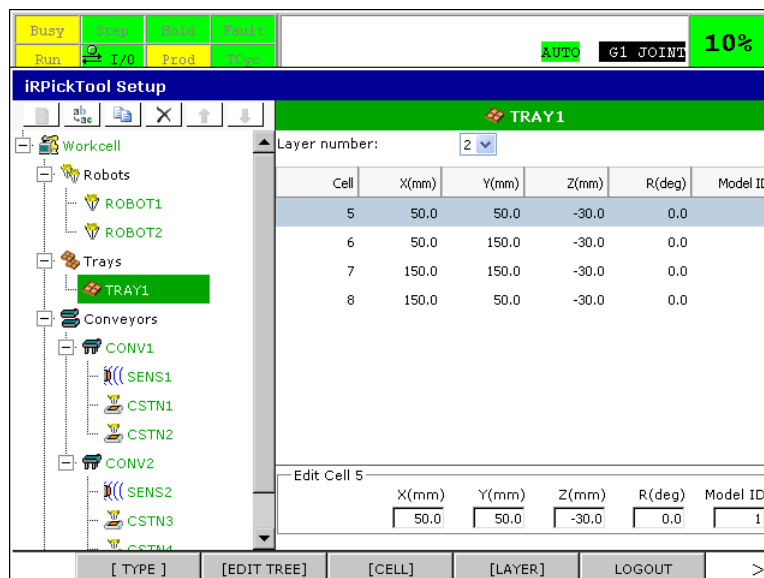
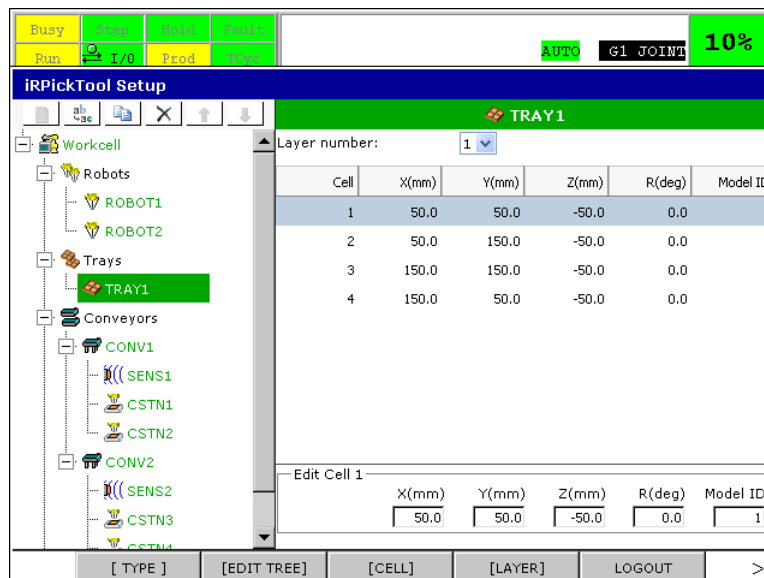
In this case, no part can be placed in the upper layer until the lower layer is filled with parts. To observe this rule, it is necessary to use the "layer" function of the tray to prevent a part from being placed in the upper layer until the lower layer is filled with parts.

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".



Setting the tray

Set the tray as shown below.

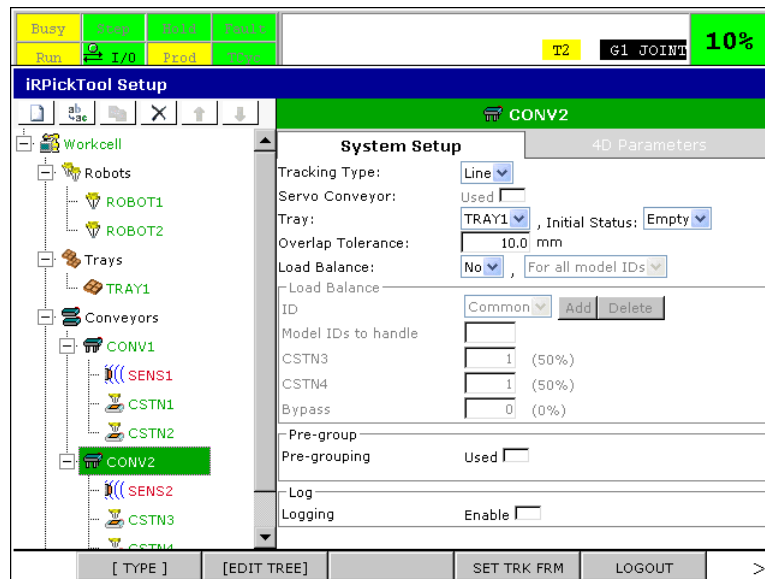


Cells 1 to 4 with layer number 1 belong to layer 1, and cells 5 to 8 with layer number 2 belong to layer 2. If you set the tray as shown above, the robot does not place a part in cells 5 to 8 of layer 2 until all the cells of layer 1 are filled.

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.



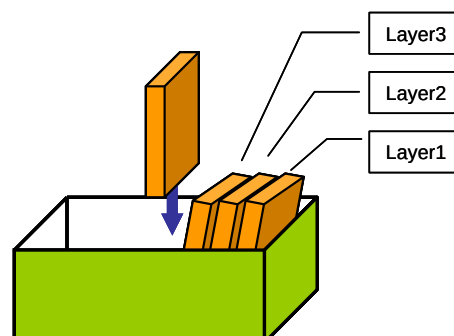
In [Tray], select the tray you have set. To fill the tray with parts, select [Empty] for [Initial Status]. If you disable the load balance, the robot places as many parts as possible in the cells while respecting the layer settings.

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.

5.4 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.



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					
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

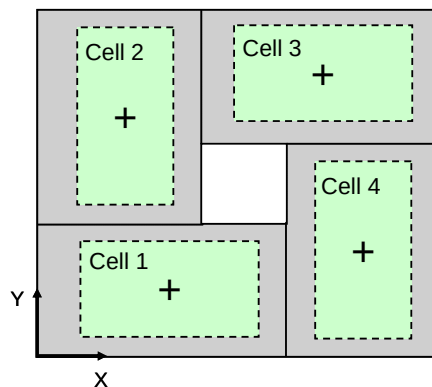
Cells 1 to 3 are added to different layers, respectively. Cell 1 is at the end. If you set the tray as shown above, parts are placed sequentially in cells 1, 2, and 3 of the tray when [Empty] is selected for [Initial Status] for the conveyor.

Example of setting the conveyor

Set the conveyor in the same way as described in Section 5.3, "PLACING PARTS IN LAYERS".

5.5 PLACING PARTS IN DIFFERENT DIRECTIONS IN A BOX

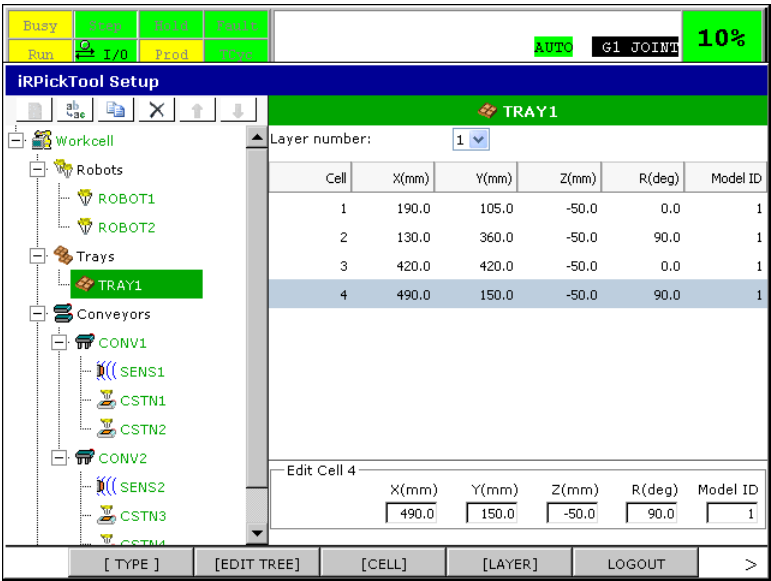
This section describes the case in which parts are placed in different directions in different cells in a box. In the example shown below, the direction of cells 2 and 4 is 90 degrees different from the other two cells.



In this case, you need to change only the tray settings.

Setting the tray

An example of setting the tray is given below.



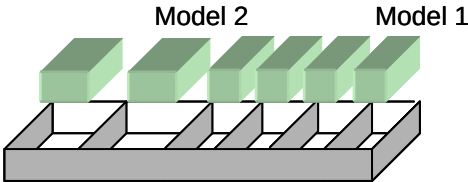
The angle of cell 1 is 0 degrees, and the angle of each other cell is relative to that of cell 1. The angle of cells 2 and 4 is 90 degrees.

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.

5.6 PLACING PARTS IN DIFFERENT PLACES OF A BOX DEPENDING ON THE MODEL ID

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.

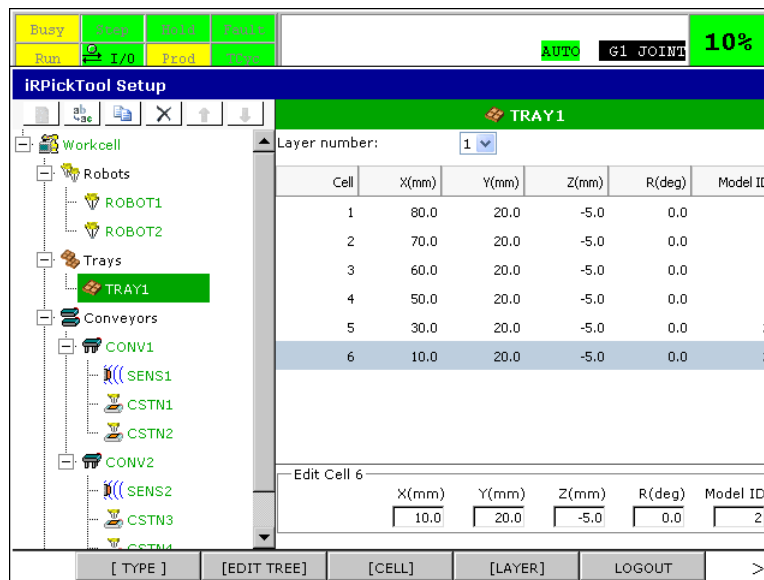


In cases like this, set the model IDs when setting the tray. Also, specify the model IDs of the cells you want to be allocated when PKCSGETQUE is called.

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.

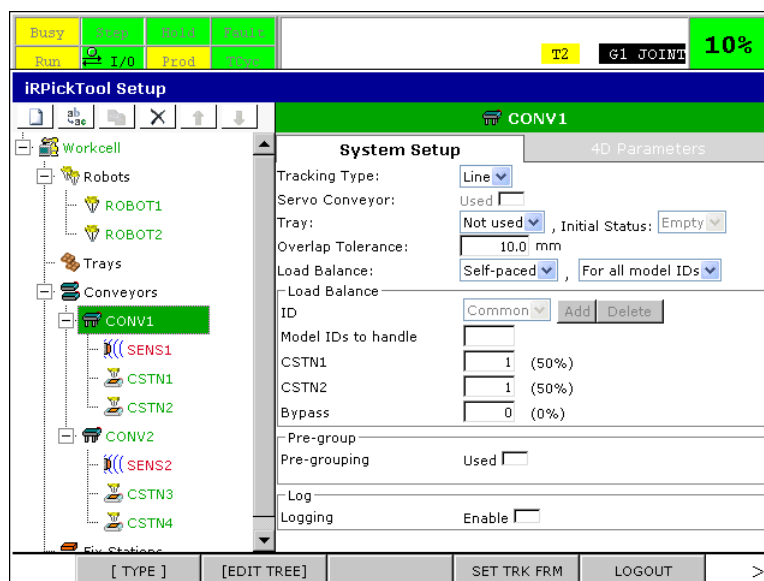
The model IDs of the cells are set to 1 and 2, as with the part model IDs for the infeed conveyor. Parts with model ID 1 are placed in the cells with model ID 1, and parts with model ID 2 are placed in the cells with model ID 2. The model IDs of cells do not necessarily need to match those of parts. 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.

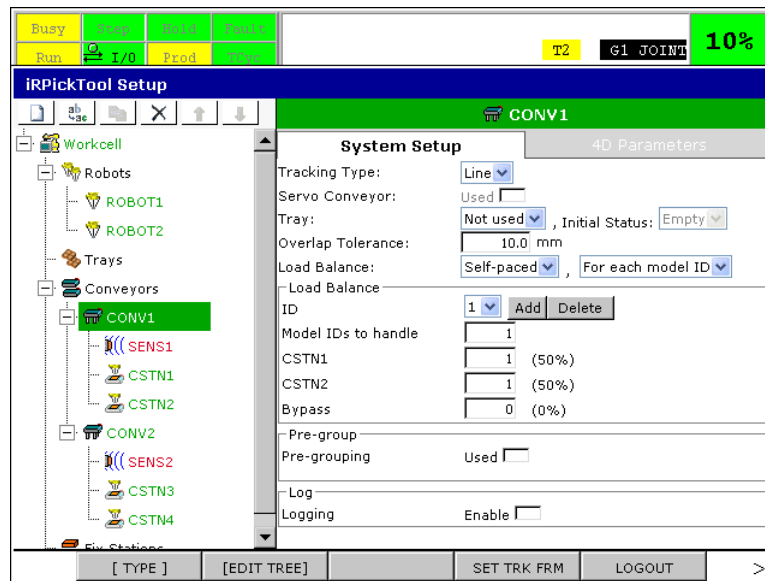
Setting the conveyor

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 4.9.2.3, "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

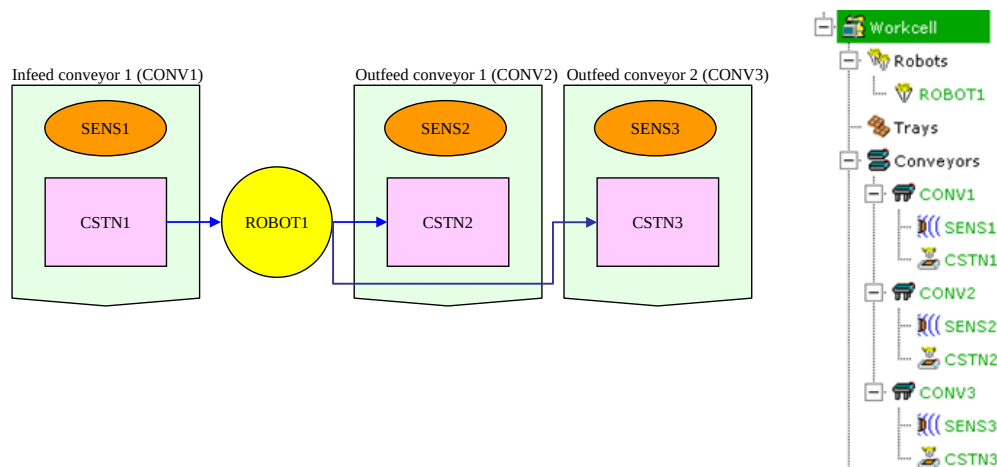
If there is no cell corresponding to a picked part, the robot cannot place the part that it holds. In order to make this type of system operate normally, the ratio of parts 1 and 2 conveyed on the infeed conveyor needs to match the ratio of parts 1 and 2 conveyed on the outfeed conveyor.

5.7 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.



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 4.9.2.1, "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]

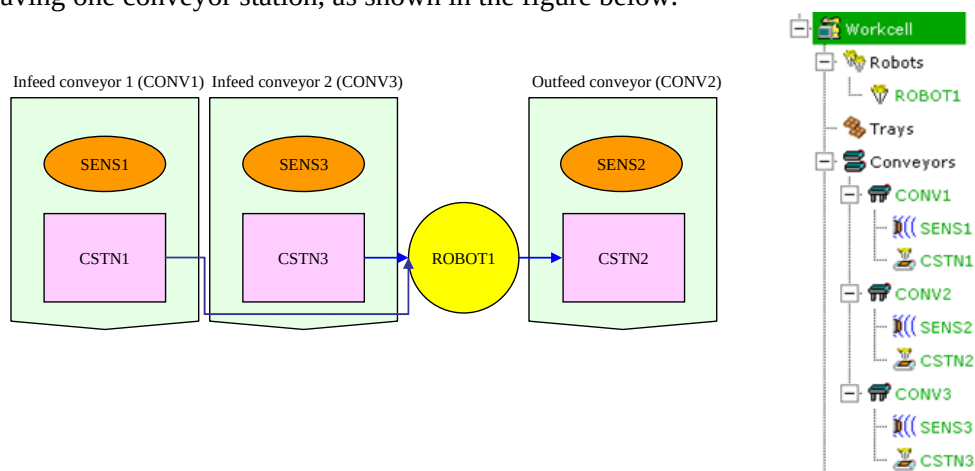
```

Basically, this program is the same as the DROP1.TP program shown in Subsection 4.9.2.3, "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.

5.8 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.



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 4.9.2.1, "Main program" and the program, PICK1.TP, shown in Subsection 4.9.2.2, "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.

5.9 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 4.9.2.2, "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]

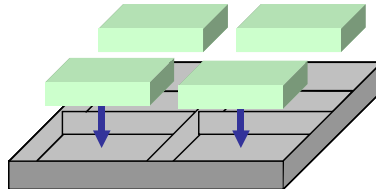
```

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.

5.10 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.



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.

TRAY1						
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	2	

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 cell 1.

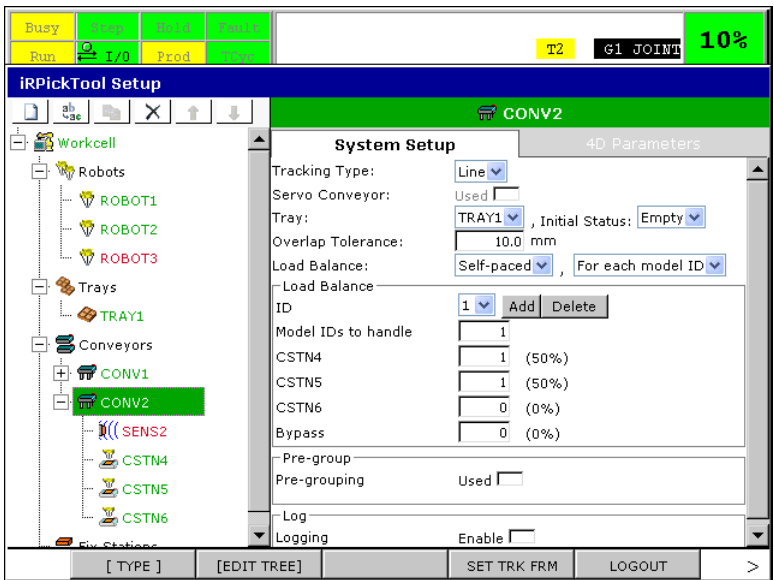
The purpose of making the position (X,Y,Z,R) the same as cell 1 is to allow the robot motion for the last cell to be taught independently from the other cells.

NOTE

Normally, teach the robot position only for cell 1. The robot position for the other cells is automatically calculated from the tray pattern based on cell 1. 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 cell 1. This makes the position of cell 5 the same as cell 1, 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 cell 1.

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.

Load Balance		
ID	2	Add Delete
Model IDs to handle	2	
CSTN4	0	(0%)
CSTN5	0	(0%)
CSTN6	1	(100%)
Bypass	0	(0%)

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 cells 1 and 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.

5.11 MULTI-VIEW

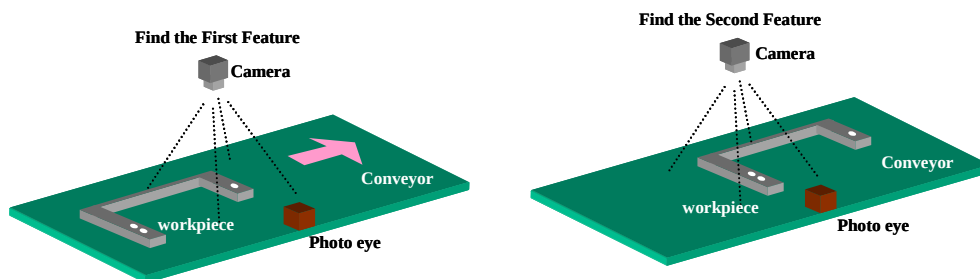
This chapter explains the setup of multi-view using one or two cameras in *iRPickTool*.

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 conveyor belt is given as an example.

5.11.1 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.



5.11.2 Limitations

- 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 4.8.3.1, “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 PKSNSTART (or PKWCSTART).

- Multi-view function cannot find two or more parts at the same time.

5.11.3 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.

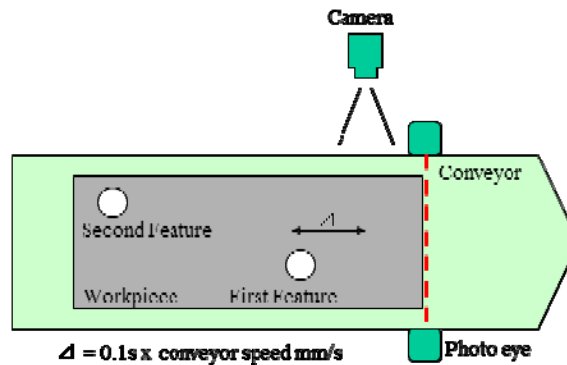
Connection

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 3.3, “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).



A camera layout example

5.11.4 Setup

The setup procedures of Conveyor, Camera and Calibration data are the same as those for a single view configuration. The procedures for other data are described below.

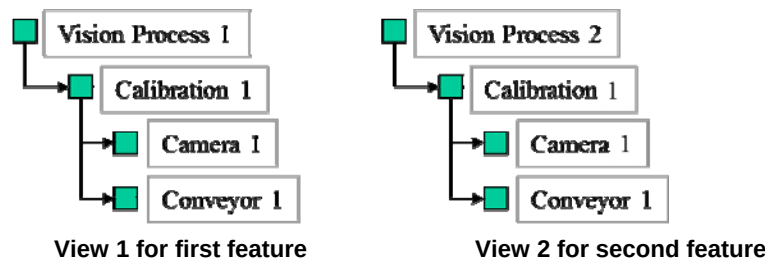
5.11.4.1 Vision process

Case of one camera

Create two single view vision processes. One is for the first feature on a part (view 1), and the other for the second feature on the same part (view 2).

Train each vision process after making each feature be placed right under the corresponding camera. The training procedure itself is the same as that for a single view vision process.

The process diagram is shown below.

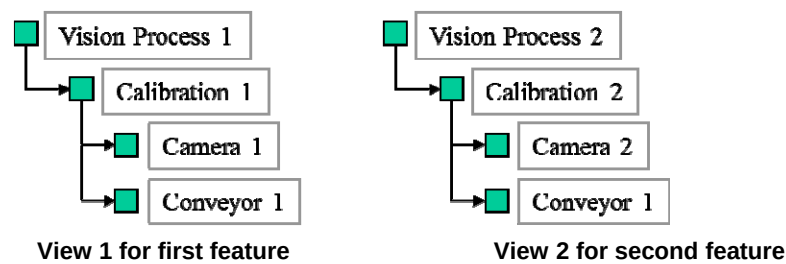


Case of two cameras

Create two camera data and two calibration data for each camera first. Each procedure is the same as that for a single view configuration.

Then create two single view vision processes. One is for the first feature on a part (view 1), and the other for the second feature on the same part (view 2).

Train each vision process after making each feature be placed right under the corresponding camera. The training procedure itself is the same as that for a single view vision process. The process diagram is shown below.



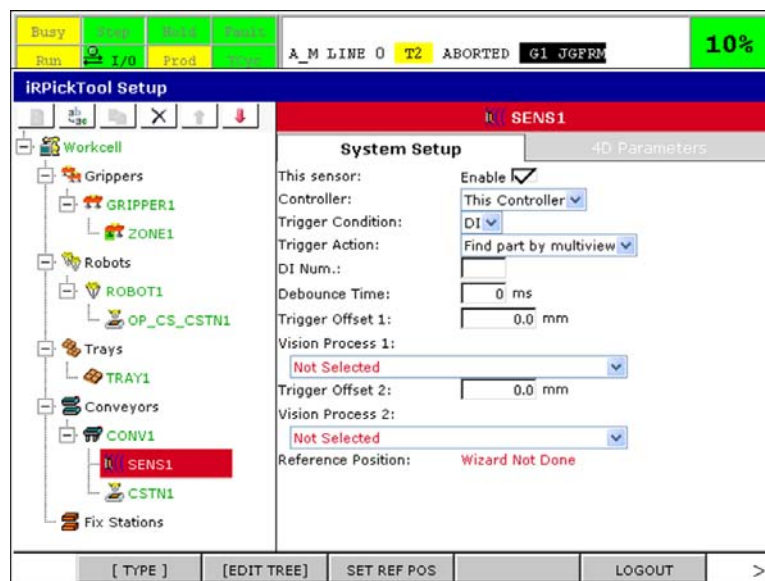
5.11.4.2 Robot program

You create and setup robot programs just as a single-view configuration.

5.11.4.3 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(mm), Trigger Offset 2(mm)

[Trigger Offset 1] is a distance in mm by which the conveyor moves after the part passes by the sensor such as a photo eye until the first feature is snapped and found by the vision.

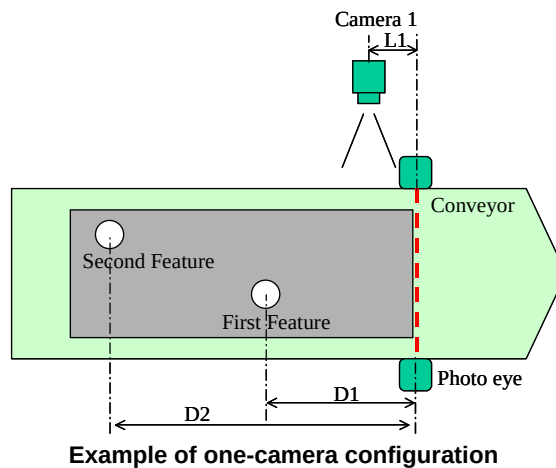
[Trigger Offset 2] is another distance in mm by which the conveyor moves after the part passes by the sensor such as a photo eye until the second feature is snapped and found by the vision.

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)

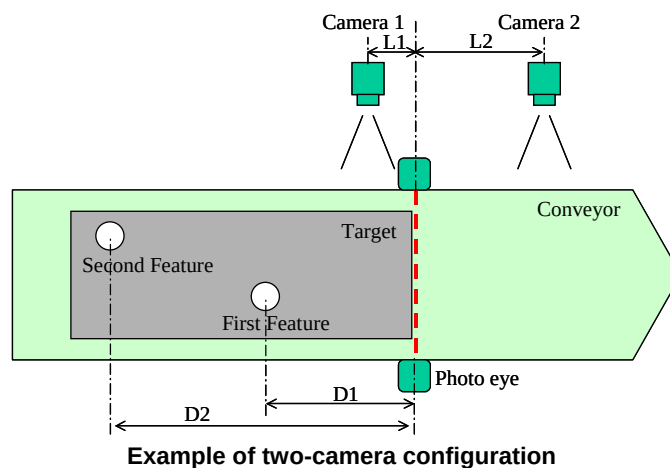


Case of two cameras

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 = $D1 \text{ (mm)} - L1 \text{ (mm)} - 0.1 \text{ (s)} \times \text{Conveyor speed (mm/s)}$

Trigger Offset 2 = $D2 \text{ (mm)} + L2 \text{ (mm)} - 0.1 \text{ (s)} \times \text{Conveyor speed (mm/s)}$



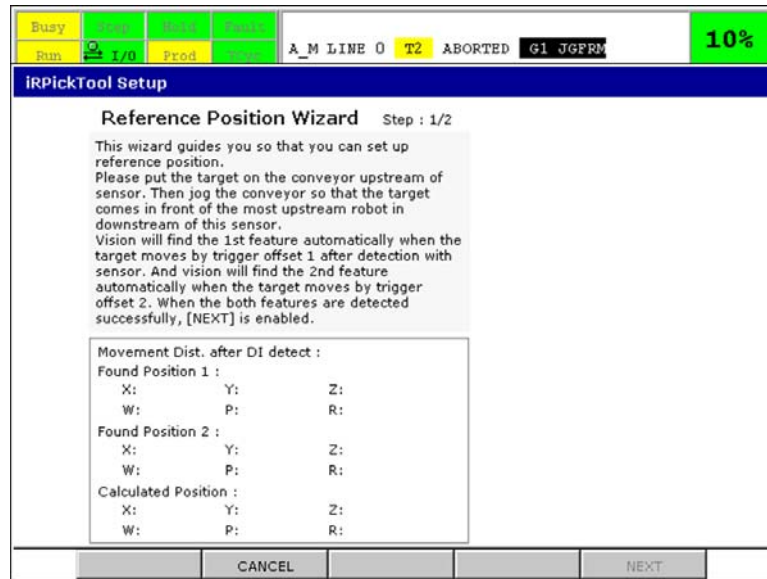
Vision Process Name 1, Vision Process Name 2

Specify the vision process name (view 1) for the first feature to [Vision Process Name 1] and the vision process name (view 2) for the second feature to [Vision Process Name 2].

5.11.4.4 Reference position setting and robot position teaching

You set reference positions according to the Wizard 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 Wizard. 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 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 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.



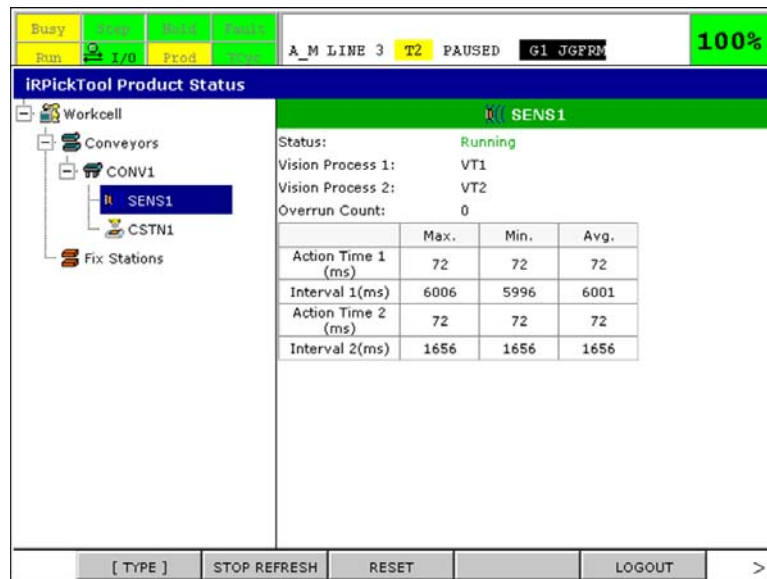
- 4 Teach the robot positions. This teaching procedure is the same as that for a single view configuration.
- 5 Press F5 [Finish] key to finish the Wizard.

5.11.5 Run Multi-View System

A procedure to run the multi-view system is the same as a single view system. Also it is run by PKSNSTART (or PKWCSTART) and stopped by PKSNSTOP (or PKWCEND).

5.11.6 Production Status

The following screen is displayed when you use multi-view in iRPickTool production status.



Vision Process 1

Vision process name which you specify in [Vision Process 1] of a sensor.

Vision Process 2

Vision process name which you specify in [Vision Process 2] of a sensor.

Action Time 1

Total process time regarding the vision process 1.

Action Time 2

Total process time regarding the vision process 2.

Interval 1

Interval between the start time of the vision process 1 and the start time of the vision process 2. If this value is smaller than Action Time 1, the overrun occurs.

Interval 2

Interval between the start time of the vision process 2 and the start time of the vision process 1. If this value is smaller than Action Time 2, the overrun occurs.

5.12 CIRCULAR TRACKING

This section explains the setup of circular tracking.

5.12.1 Limitations

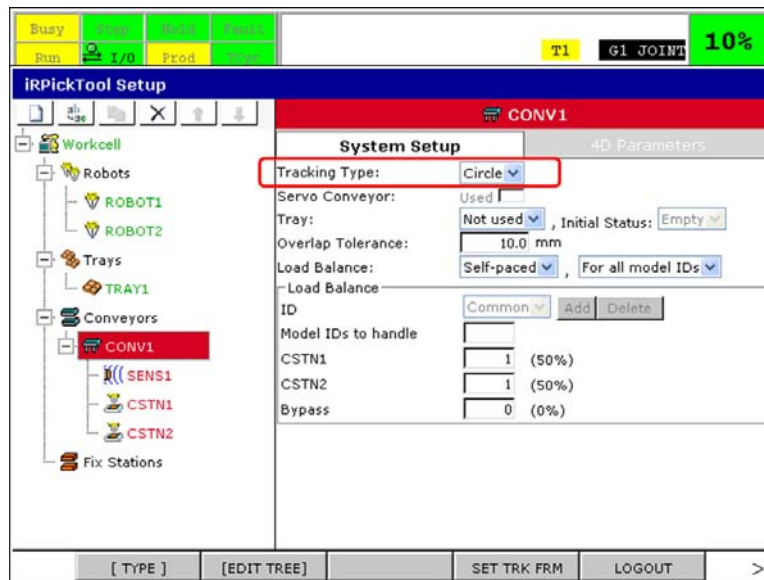
- You cannot use the multi-view together with circular tracking.

5.12.2 Setup

Circular tracking can be set up just as line tracking. Be careful to set up your system according to the following notes.

Setting a tracking frame

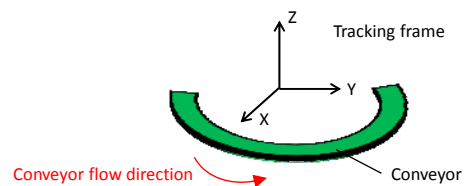
Refer to Subsection 4.4.1.2 “Circular Conveyor” or Subsection 6.6.1.2 “Circular Conveyor”. Before step 1, be sure to select “Circle” in [Tracking Type].



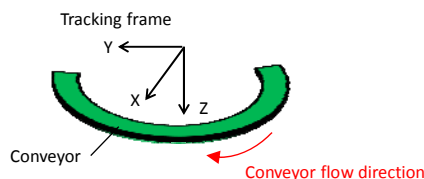
Z-axis direction of the tracking frame

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.



- When the conveyor flow direction is clockwise, the Z-axis direction of the tracking frame is 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 4.3 “SETUP OF A TRAY” or Section 6.5 “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 4.5.4.4 “Teaching a vision process” or Subsection 6.7.4.4 “Teaching a vision process”.



CAUTION

When setting a tracking frame, be sure to be as low as possible the speed of the conveyor.

5.12.2.1 When the vision system is used

If you use the vision system with circular tracking, be careful to set up your system according to the following notes. The vision system can be used with circular tracking in 7DC3/17 (V8.30P/17) or later.

Trigger Distance

In [Trigger Distance], specify an angle expressed in degrees. Refer to “When [Distance] and [Find part by vision] are selected” described in Subsection 4.5.1 “Setting a Sensor in Detail” or Subsection 6.7.1 “Setting a Sensor in Detail”.

Trigger Condition:	Distance
Trigger Action:	Find part by vision
Trigger Distance:	10.000 °

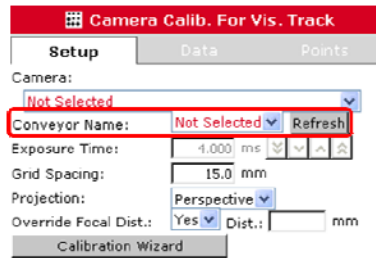
Trigger Offset

In [Trigger Offset], specify an angle expressed in degrees. Refer to “When [DI] and [Find part by vision] are selected” described in Subsection 4.5.1 “Setting a Sensor in Detail” or Subsection 6.7.1 “Setting a Sensor in Detail”.

Trigger Condition:	DI
Trigger Action:	Find part by vision
DI Num.:	
Debounce Time:	0 ms
Trigger Offset:	0.000 °

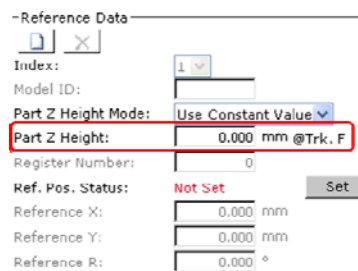
Calibrating the camera

In [Conveyor Name], select the conveyor where “Circle” is selected in [Tracking Type]. Refer to Subsection 4.5.4.2 “Calibrating the camera” or Subsection 6.7.4.2 “Calibrating the camera”.

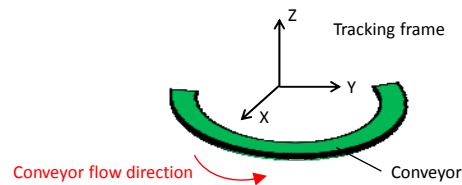


Part Z Height

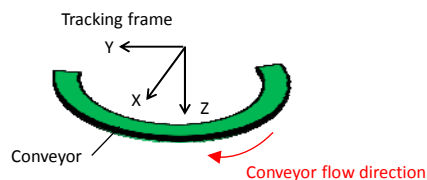
Check the Z-axis direction of the tracking frame before setting [Part Z Height] in [Reference Data]. Refer to Subsection 4.5.4.4 “Teaching a vision process” or Subsection 6.7.4.4 “Teaching a vision process”.



- When the conveyor flow direction is counterclockwise, the Z-axis direction of the tracking frame is upward.



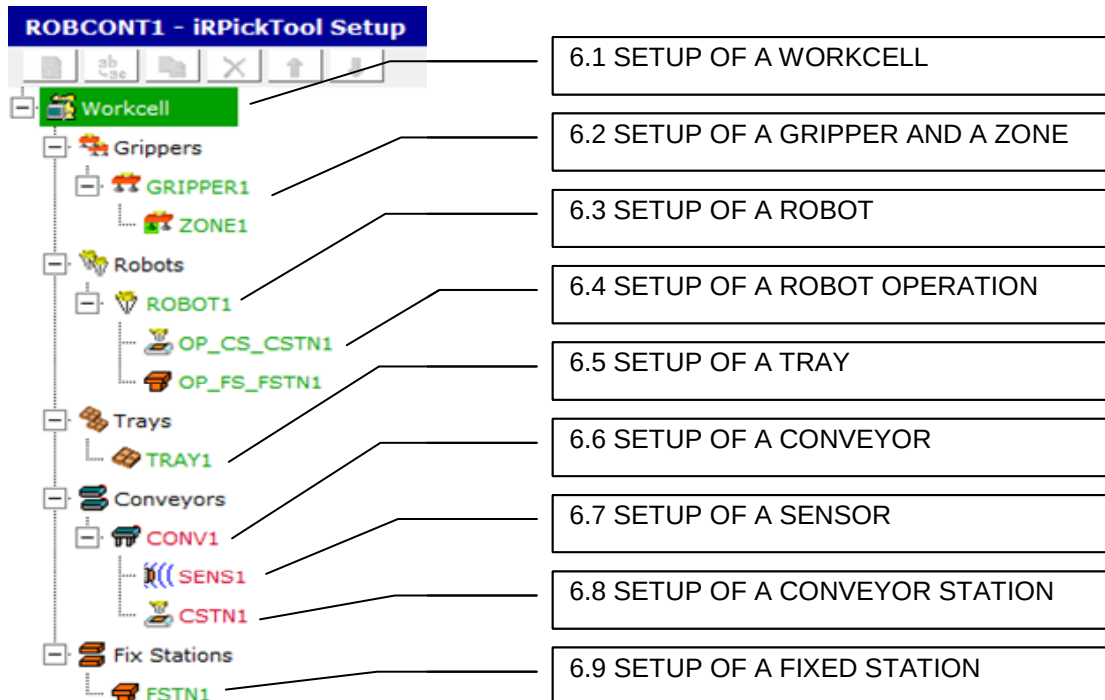
- When the conveyor flow direction is clockwise, the Z-axis direction of the tracking frame is downward.



6 SETUP OF PLUG & PLAY FUNCTIONS

This chapter describes the *iRPickTool* setup procedure - for a configuration that includes Plug & Play functions.

The items that are set in Section 6.1, "SETUP OF A WORKCELL" to Section 6.9, "SETUP OF A FIXED STATION" correspond to the *iRPickTool* setup screen as shown in the figure below.



The teaching of the basic motions will be completed by performing the procedures described in Section 6.1, "SETUP OF A WORKCELL" to Section 6.12, "TEACHING THE POSITION TO ROBOTS". Section 6.13, "FINE ADJUSTMENT OF THE TRACKING MOTION" describes how to optimize the tracking motion of the robot after completing the teaching of the basic motions.

The major difference from Chapter 4, "SETUP OF THE BASIC FUNCTIONS" is that this chapter contains two additional sections: Section 6.2, "SETUP OF A GRIPPER AND ZONES" and Section 6.4, "SETUP OF AN OPERATION". Also, since standard programs are provided, you only need to touch up the relevant positions with the proper UTOOL and UFRAME for the teaching steps. An additional teaching step required is the teaching of the perch position.

The Gripper is the end-effector used for picking the parts or placing the parts. Gripper setup involves providing the UTOOL, the I/O used for the gripper activation, the I/O used for part presence checking and the delays used for proper picking and dropping. Once these are entered correctly, when you run production, the plug and play TP programs will perform the proper gripper operations for pick and place without you having to write any extra TP code.

The Robot Operation refers to the "pick" or "drop" operation performed by the robot at a station. A station contains a robot and a picking area. A station can be a Conveyor Station or a Fixed Station. Each station has an operation menu under the Robot node in the tree-view. In the operation menu, you specify properties such as the operation type (pick or drop), quantity of parts to pick or place at that station,

whether the parts should be picked individually or simultaneously, whether a part presence check must be performed after the operation, etc. Once you enter these properties, when your run production, the plug and play TP programs will perform the robot operations without you having to write any extra TP code.



CAUTION

If *iRPickTool Basic* is ordered, instead of *iRPickTool*, as a software option for any of the controllers connected via the robot ring, Plug & Play cannot be set up for any of those connected controllers. Also, it is preferred that you purchase either *iRPickTool* or *iRPickTool Basic* for all your controllers. It is not recommended to mix and match these two options.

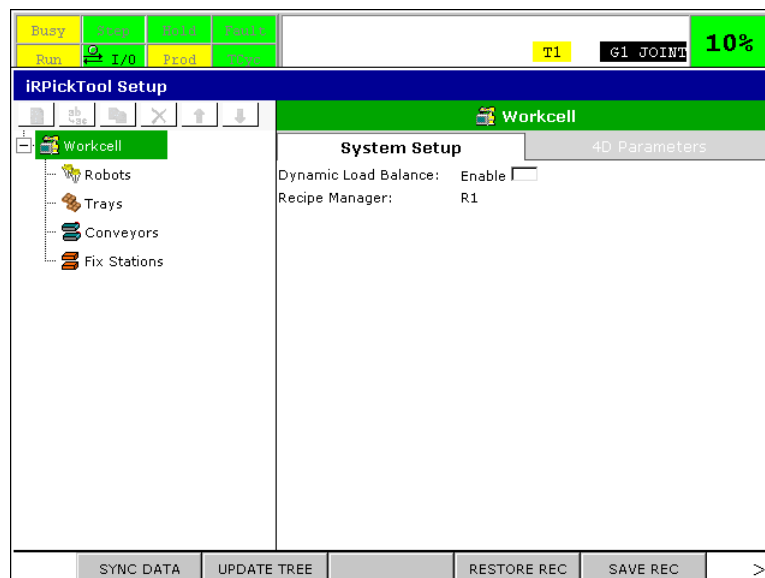
Basic operations

With *iRPickTool*, 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.

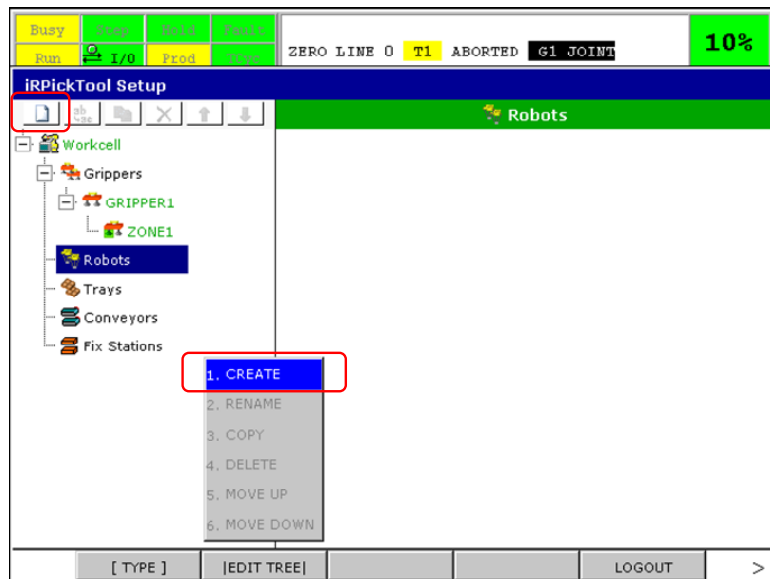
iRPickTool may be set up from any controller in a work cell. Note, however, that two or more controllers cannot be used simultaneously for the setup of *iRPickTool*.

Creating a new object

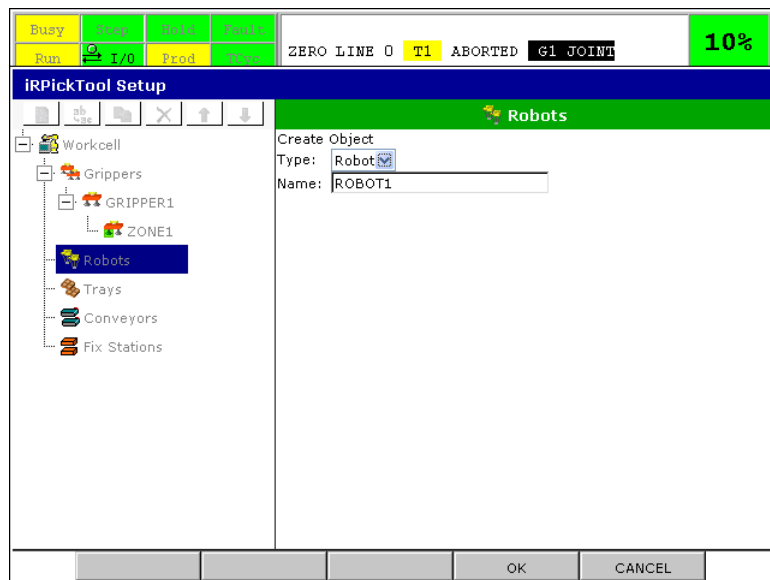
Select [6 SETUP] from [MENU] of the teach pendant and [*iRPickTool*] from F1 [TYPE] key. The *iRPickTool* setup screen is displayed, as shown below.



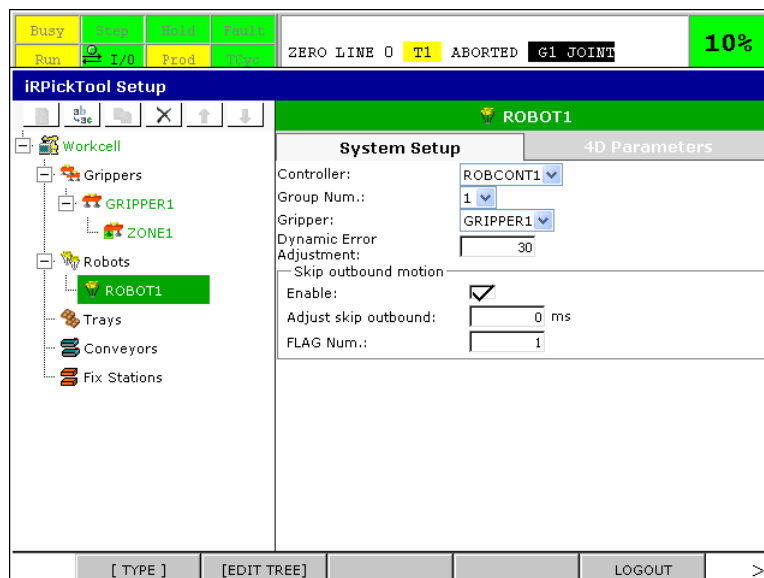
In the tree view on the left side, select the object collection corresponding to the object you want to create. For example, if you want to create a robot, select [Robots]. Then, click [1. CREATE] of F2 [EDIT TREE] key or the  button above the tree.



A screen as shown below is displayed. Enter a name that is not identical to the name of any other object. You can specify a name consisting of up to 20 one-byte characters. The name must not contain a hyphen (-) or begin with a numeric character. Also, the default name can be used as is. Press F4 [OK] key to create an object. To quit creating an object, press F5 [CANCEL] key.

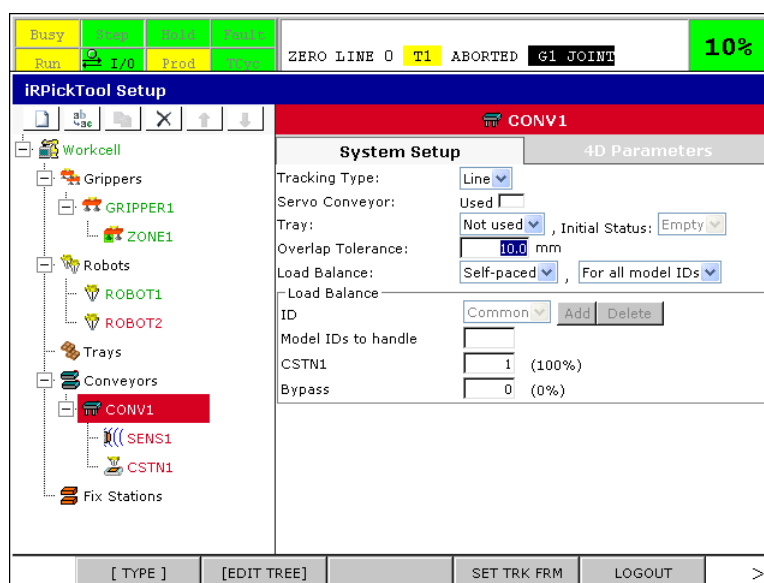


Once you have created a new object, its setup screen is opened, with the created object selected. If you have created a new robot, the screen looks as shown below.

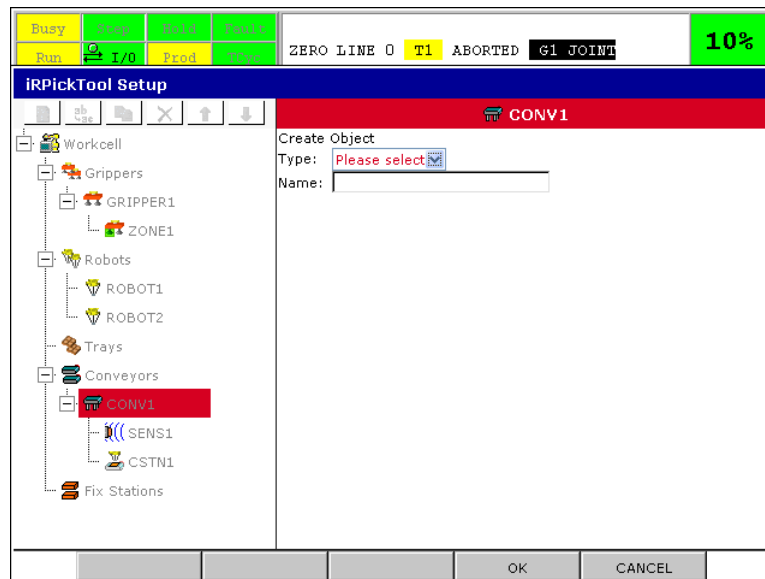


Creating a new sensor and a new conveyor station

If you create a new conveyor, a sensor and a conveyor station are automatically created, as shown below.



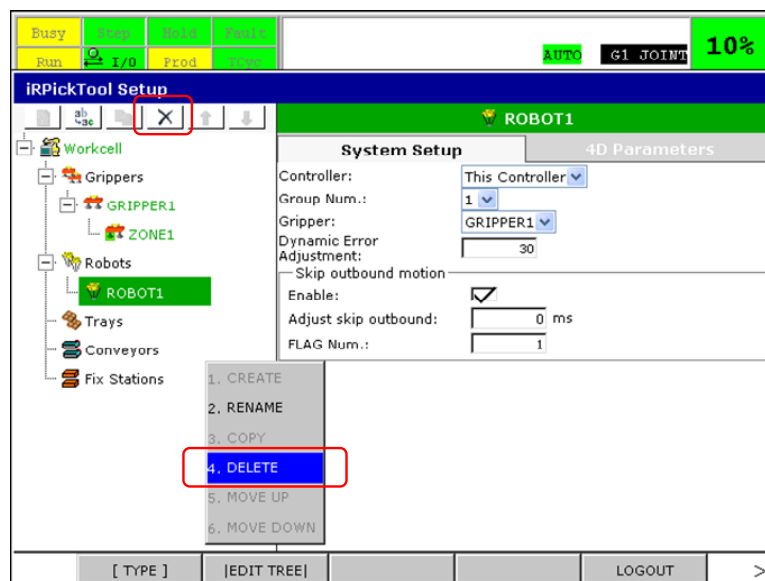
If more than one robot operates on the conveyor, add one or more conveyor stations. If there is more than one sensor on the conveyor, add one or more sensors. Click [1. CREATE] or F2 [EDIT TREE] key or the  button above the tree, with the conveyor selected. A screen as shown below is displayed.



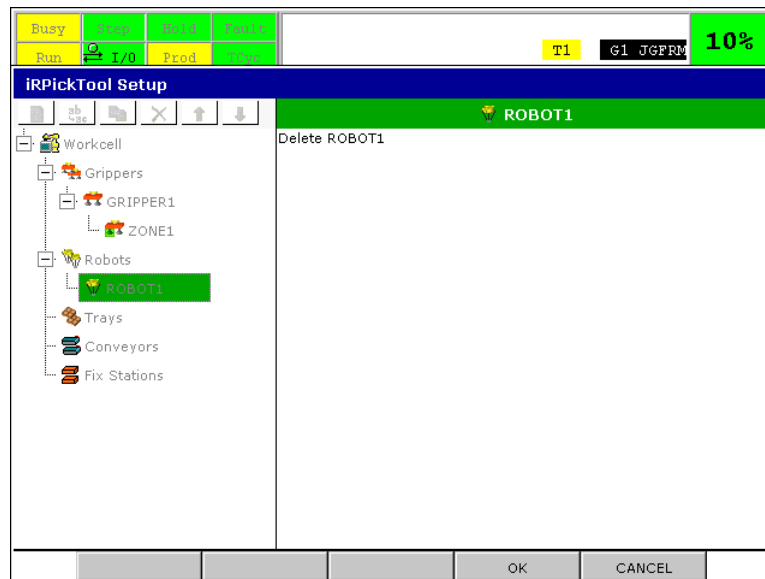
In [Type], specify whether the object you are adding is a conveyor station or sensor. The default name appears in [Name], as when you create a new object of another type. Create a new object in the same way as with an object of another type.

Deleting an object

In the tree view on the left side of the iRPickTool setup screen, select an object you want to delete. Then, click [4. DELETE] of F2 [EDIT TREE] key or the button above the tree.

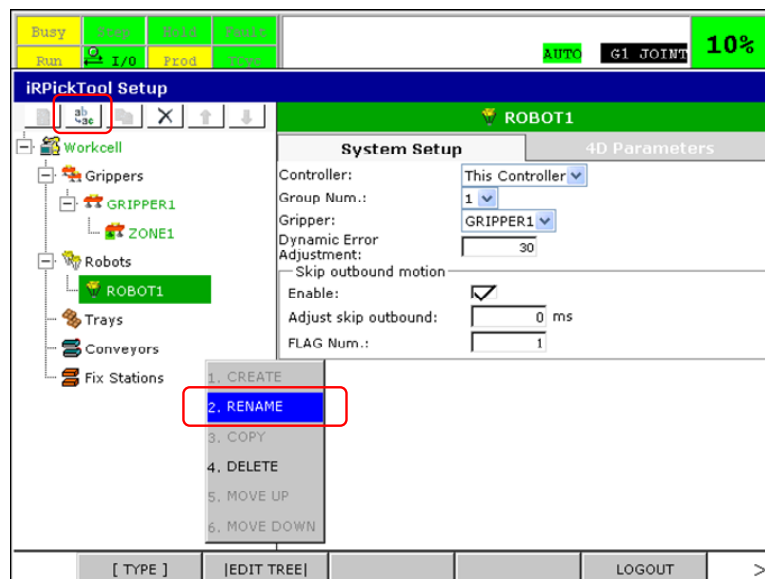


A screen as shown below is displayed. Press F4 [OK] key to delete the object. To quit deleting the object, press F5 [CANCEL] key.

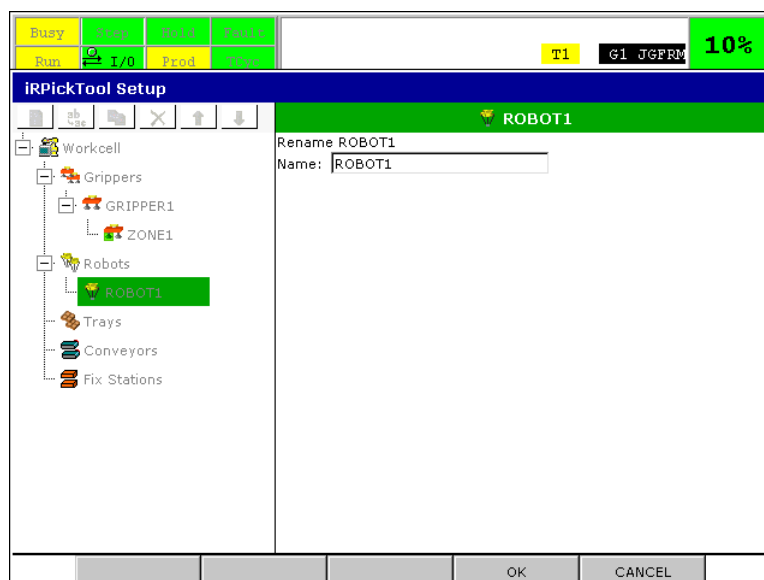


Renaming an object

In the tree view on the left side of the iRPickTool setup screen, select an object you want to rename. Click [2. RENAME] of F2 [EDIT TREE] key or the button above the tree.

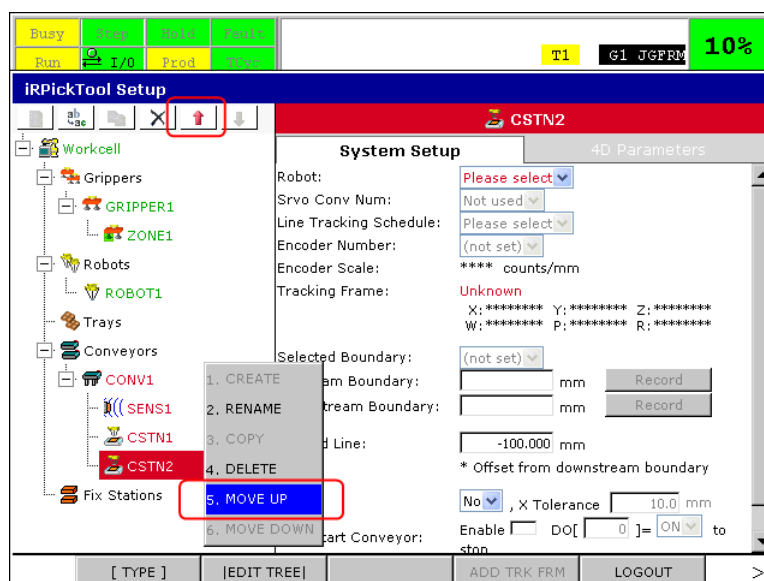


A screen as shown below is displayed. Enter the new name of the object, and press F4 [OK] key. To quit renaming the object, press F5 [CANCEL] key.



Moving an object in the order

Use this function to move a conveyor station in the order. In the tree view on the left side of the iRPickTool 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.



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.

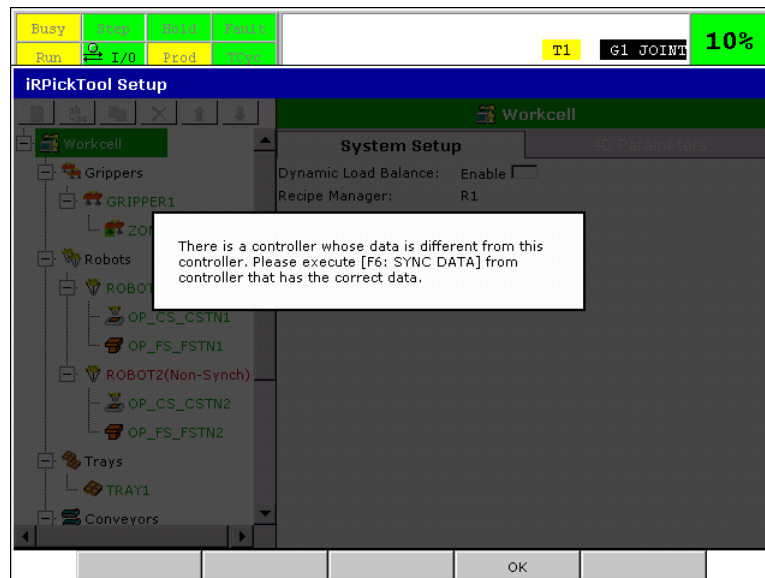
Synchronizing data

The settings of iRPickTool 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.

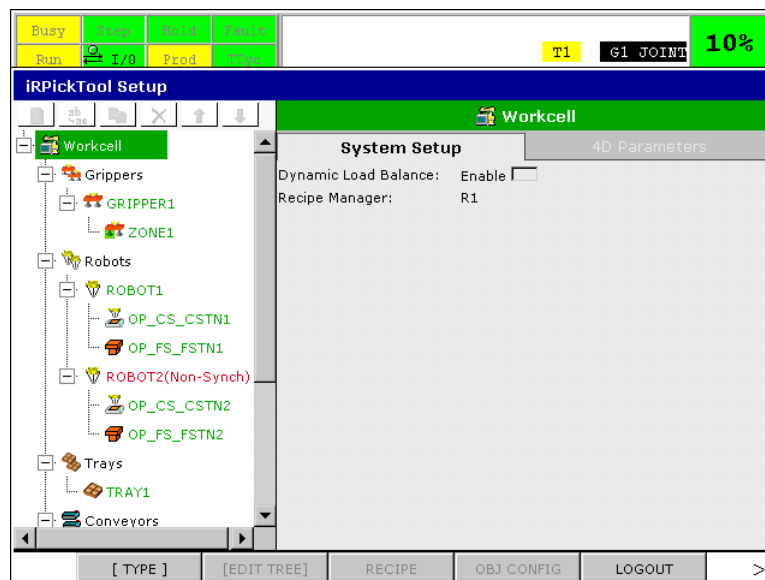
You can set up iRPickTool even if the system has any robot controller whose power is off. In that case, however, the offline robot controller is left with inconsistent data. If this happens, perform the

procedure described below, with all the robot controllers in the system powered on, to synchronize the data so that the *iRPickTool* settings are the same for all the controllers.

- 1 If you open the *iRPickTool* 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 controllers. To perform synchronization based on the settings of another controller, press F5 [LOGOUT] key to close the *iRPickTool* setup screen. Then, open the *iRPickTool* setup screen for that other controller and check its settings as instructed above.

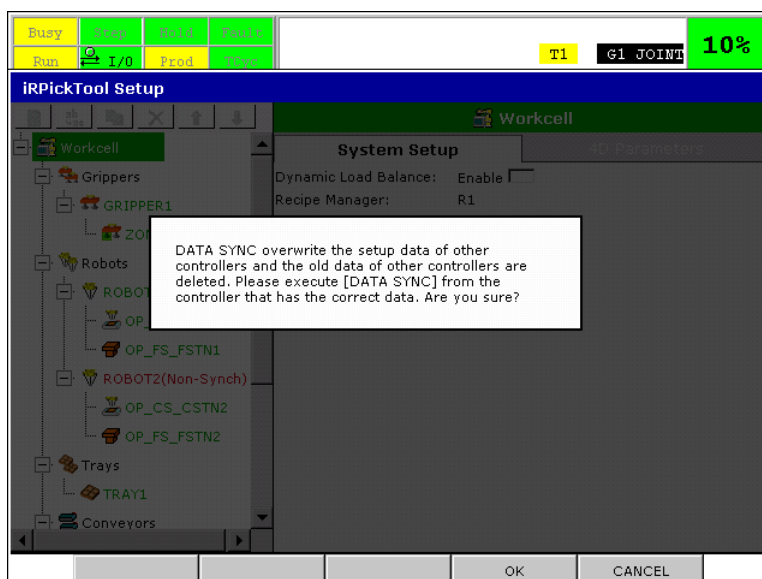
**CAUTION**

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 controller to the other controllers.

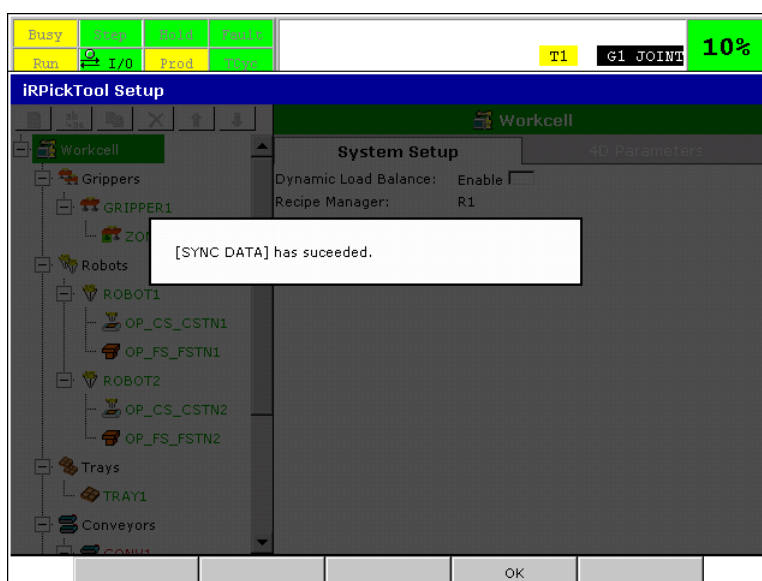
- 4 After making sure that it is appropriate to transfer the settings to the other controllers, press the [NEXT] key on a desired *iRPickTool* setup screen to display the function key shown below.



- 5 Press F1 [SYNC DATA] key. When a screen as shown below is displayed, press F4 [OK] key.



- 6 When data synchronization is complete, a screen as shown below is displayed. Press F4 [OK] key.



Refreshing the tree

Since the *iRPickTool* setup screen does not automatically reflect 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.

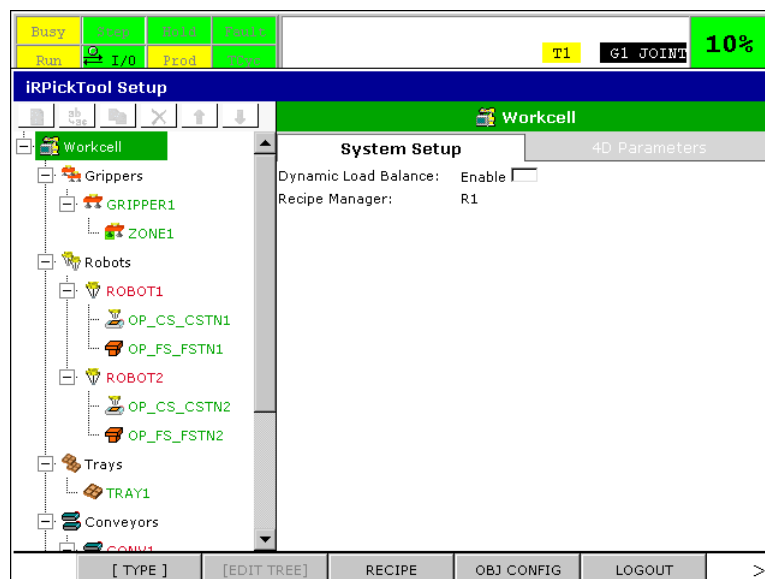
- If an offline controller goes online when the *iRPickTool* setup screen is open, this change in the status is not automatically reflected in the setup screen.

In these cases, press the [NEXT] key on a desired *iRPickTool* setup screen to display the function key shown below and then press F2 [UPDATE TREE] key.



6.1 SETUP OF A WORKCELL

Set up a work cell.



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 *iRPickTool* Recipes are stored. For more information on Recipes, see the section 7.1, “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.

Setting the number of objects

Change the maximum number of objects that can be set, on an object-by-object basis.

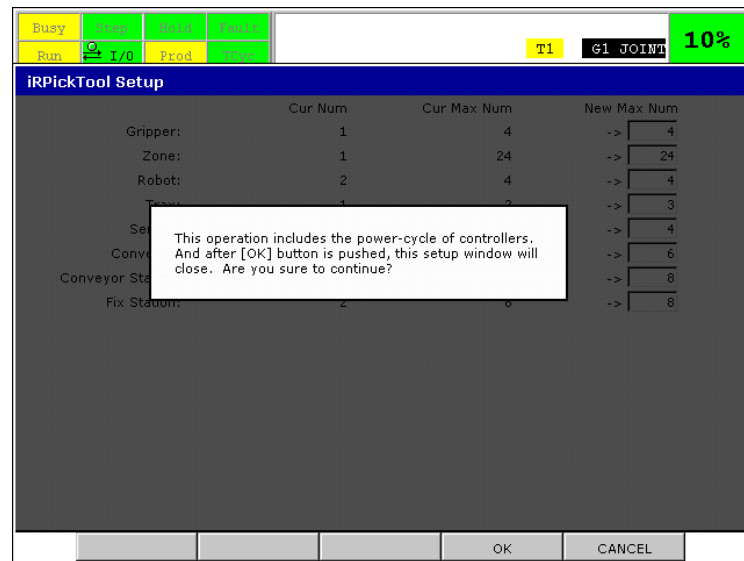
- 1 Press F4 [OBJ CONFIG] key. A screen as shown below is displayed.

Busy	Stop	Hold	Pause			T1	G1 JOINT	10%
Run	I/O	Prod	Stop					
iRPickTool Setup								
				Cur Num	Cur Max Num		New Max Num	
Gripper:				1	4	->	4	
Zone:				1	24	->	24	
Robot:				2	4	->	4	
Tray:				1	2	->	2	
Sensor:				1	4	->	4	
Conveyor:				1	4	->	4	
Conveyor Station:				2	8	->	8	
Fix Station:				2	8	->	8	
						OK	CANCEL	

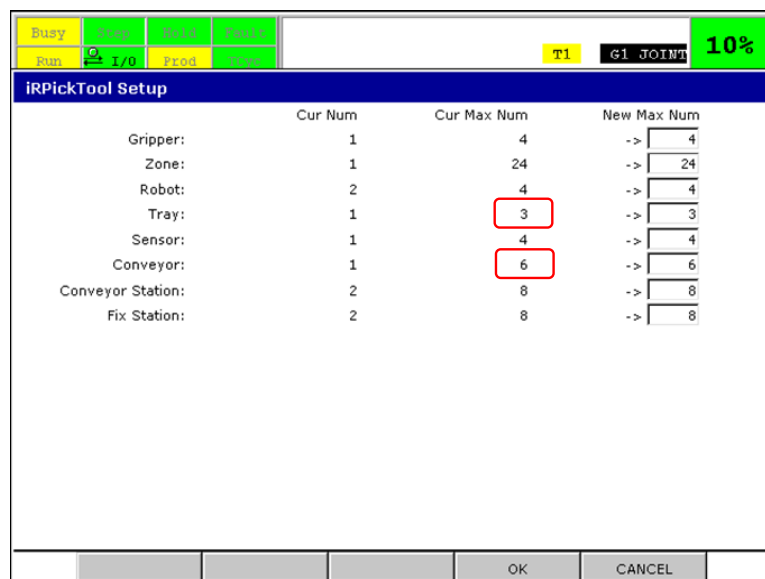
- 2 In the [New Max Num] field of the object whose number you want to increase, enter the new maximum number. Shown below is an example of changing the maximum numbers of trays and conveyors.

Busy	Stop	Hold	Pause			T1	G1 JOINT	10%
Run	I/O	Prod	Stop					
iRPickTool Setup								
				Cur Num	Cur Max Num		New Max Num	
Gripper:				1	4	->	4	
Zone:				1	24	->	24	
Robot:				2	4	->	4	
Tray:				1	2	->	3	
Sensor:				1	4	->	4	
Conveyor:				1	4	->	6	
Conveyor Station:				2	8	->	8	
Fix Station:				2	8	->	8	
						OK	CANCEL	

- 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

The maximum number that can be modified for each object shown Table 6.1.

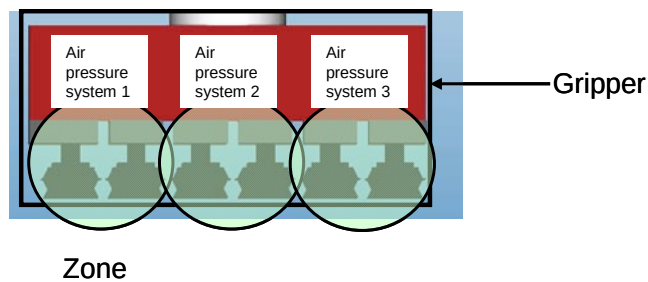
Table 6.1 : Maximum number that can be modified for each object

Object	Maximum number
Gripper	32
Zone	100
Robot	32
Tray	16
Sensor	32
Conveyor	32
Conveyor Station	84
Fix Station	84

6.2 SETUP OF A GRIPPER AND A ZONE

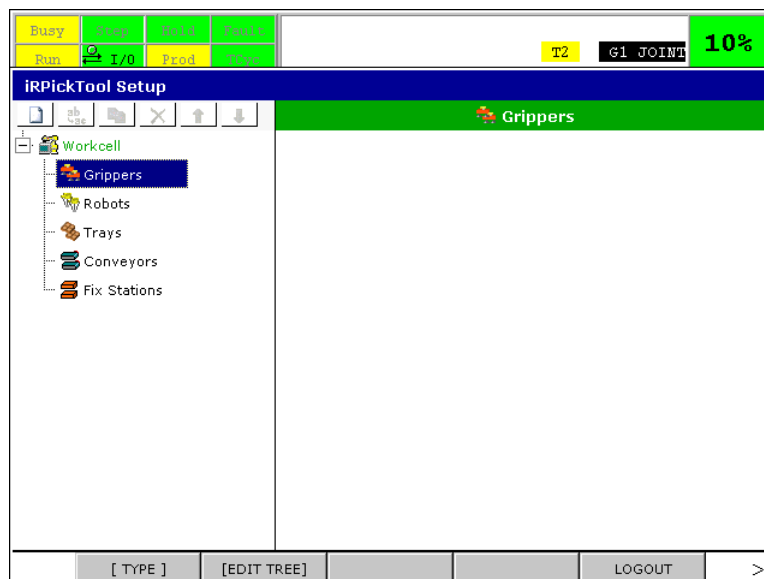
The Gripper Setup is the first step to perform when you have J945: iRPickTool. The Gripper Node is the topmost node in the tree-view as shown below.

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.



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. (For information about recipes, see Section 7.1, "EXCHANGE WORKCELL WITH USING RECIPE".) This allows you to use different zone utools, pick and drop delays, etc. for each recipe you are running without any custom programming on your part.

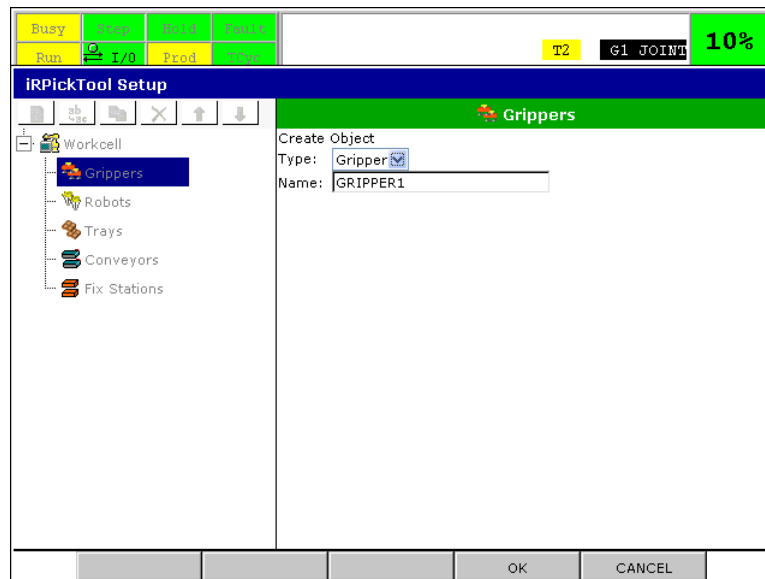


6.2.1 Gripper and Zone

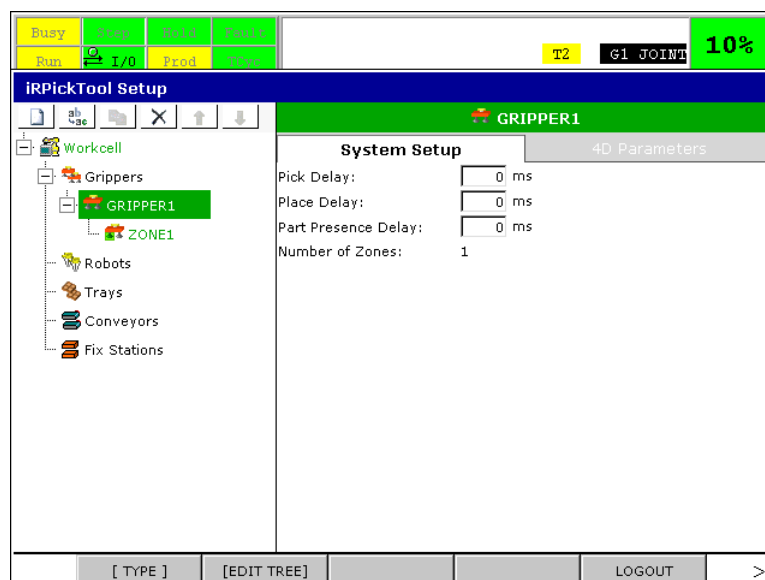
Creating a new Gripper

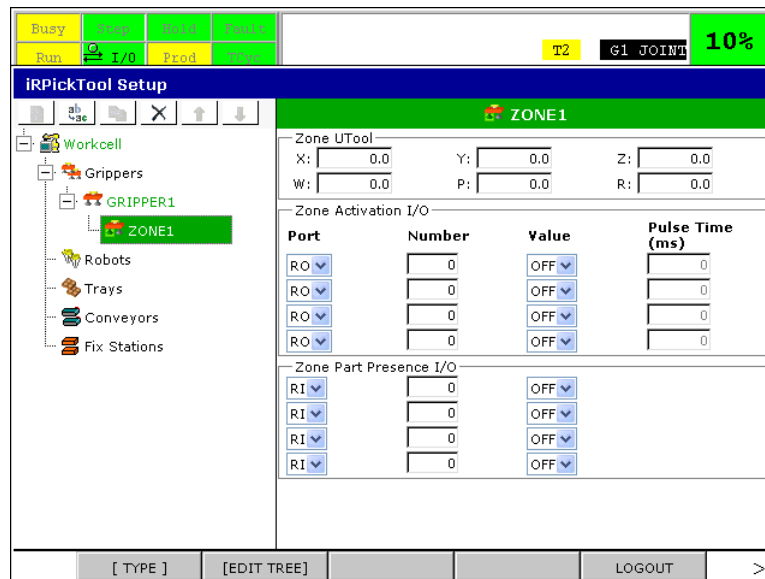
Create a new Gripper.

- 1 Select the "Grippers" node. Then click the  button in the tree-view toolbar or click on the [EDIT TREE] function key and select CREATE. The following screen is displayed:



- 2 In [Name], a default name (for ex: GRIPPER1) is automatically provided. You can enter a different name if you wish to.
A name consisting of up to 20 one-byte characters can be specified. The name may not begin with a numeric character or contain a “-“ (minus sign).
- 3 Click the F4 key, [OK] button.
The created Gripper is added to the Grippers Node with default properties. A gripper Zone is also added under the named gripper.

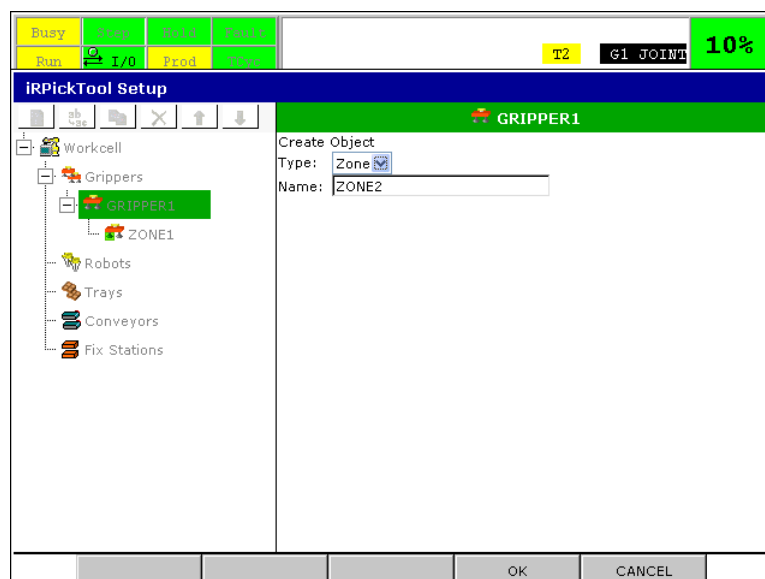




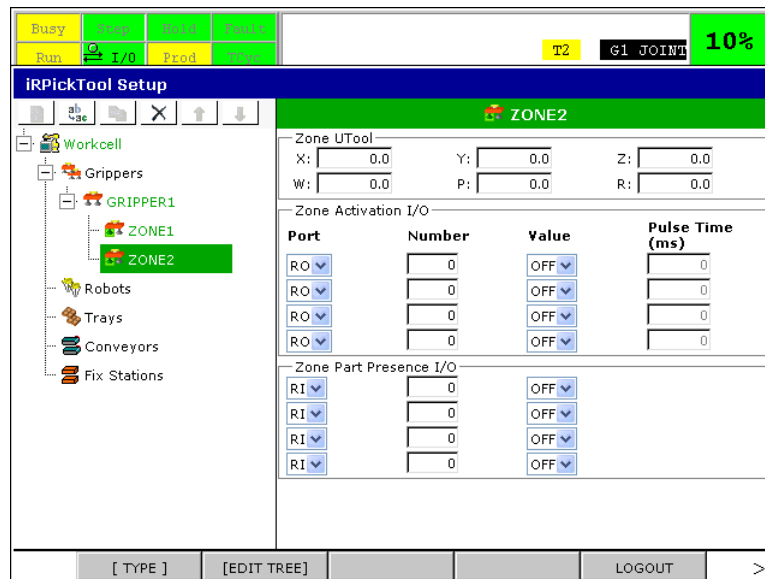
Adding a Gripper Zone

Add a Zone to a Gripper.

- 1 To add a Zone to the Gripper, first select the gripper in the tree and then click the  button or click on the F2 key, [EDIT TREE] function key and select CREATE. A screen as shown below is displayed.



- 2 In [Name], a default name (for ex: ZONE1) is automatically provided. You can enter a different name if you wish to.
A name consisting of up to 20 one-byte characters can be specified. "-"(minus sign) cannot be used. The name cannot begin with a numeric character.
- 3 Click the [OK] button to add the Zone to the Gripper.

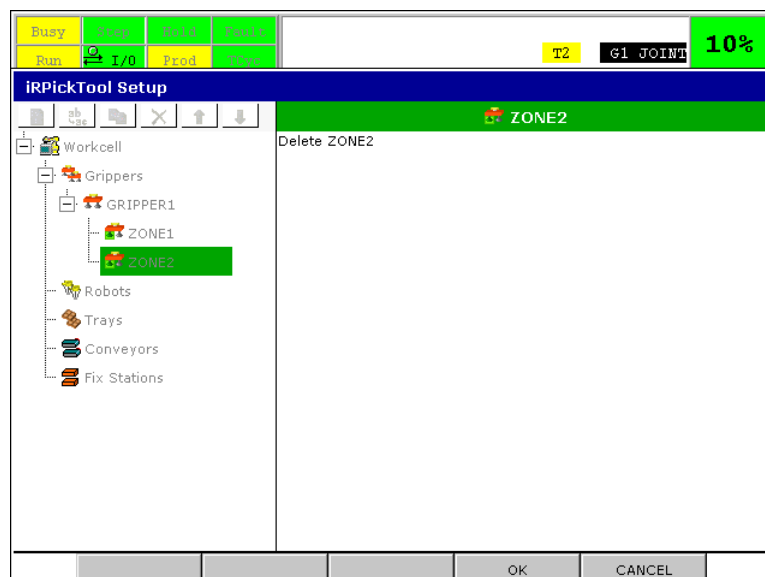


- 4 When there is more than one zone for this gripper, repeat the above procedure for each zone.

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 a zone (the last zone of a gripper) to be deleted.
- 2 Click the button or select DELETE from the [EDIT TREE] function key pull-up.
- 3 Click OK to the delete prompt as shown below:



- 4 The zone will be deleted from the Gripper.

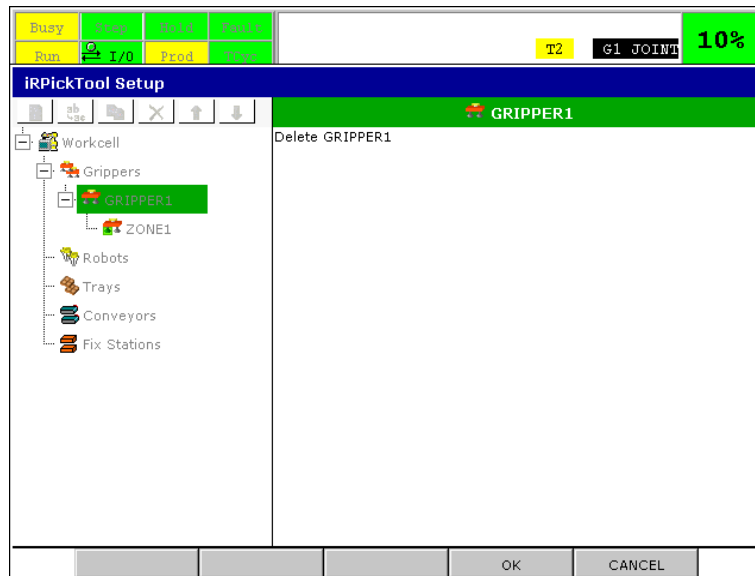
CAUTION

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 a gripper to delete.
- 2 Click the  button or select DELETE from the [EDIT TREE] function key pull-up.
- 3 Click OK to the delete prompt as shown below:

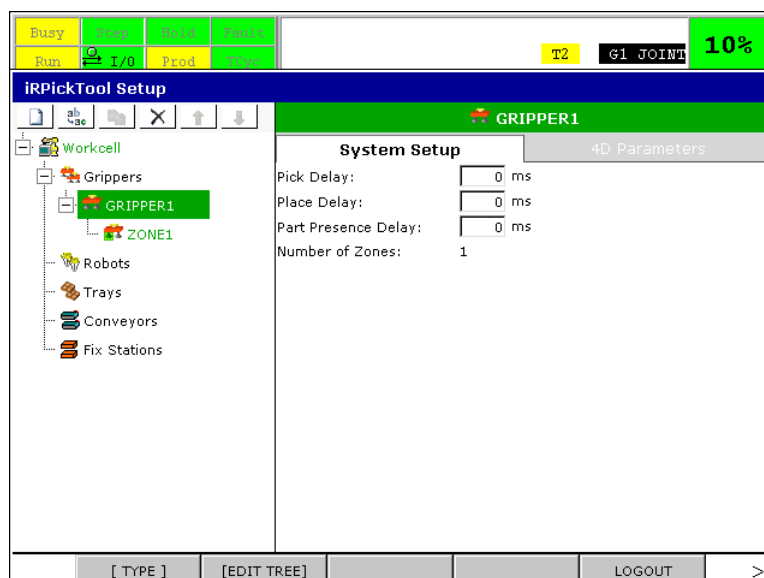


- 4 The gripper and all its zones will be deleted.

6.2.1.1 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.

During production run, the *iRPickTool* TP programs call the PKGRGETPKDLY.pc macro to get the pick delay into a Numeric Register.

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 *iRPickTool* 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 *iRPickTool* 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.

6.2.1.2 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.

The screenshot shows the **iRPickTool Setup** window. On the left is a tree view with the following structure:

- Workcell
 - Grippers
 - GRIPPER1
 - ZONE1** (selected)
 - Robots
 - Trays
 - Conveyors
 - Fix Stations

The main panel displays the configuration for **ZONE1**:

- Zone UTool**: X: 0.0, Y: 0.0, Z: 0.0, W: 0.0, P: 0.0, R: 0.0
- Zone Activation I/O**:

Port	Number	Value	Pulse Time (ms)
RO	0	OFF	0
RO	0	OFF	0
RO	0	OFF	0
RO	0	OFF	0
- Zone Part Presence I/O**:

Port	Number	Value
RI	0	OFF
RI	0	OFF
RI	0	OFF
RI	0	OFF

At the bottom of the window are buttons: [TYPE], [EDIT TREE], LOGOUT, and a right arrow button.

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.

iRPickTool 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

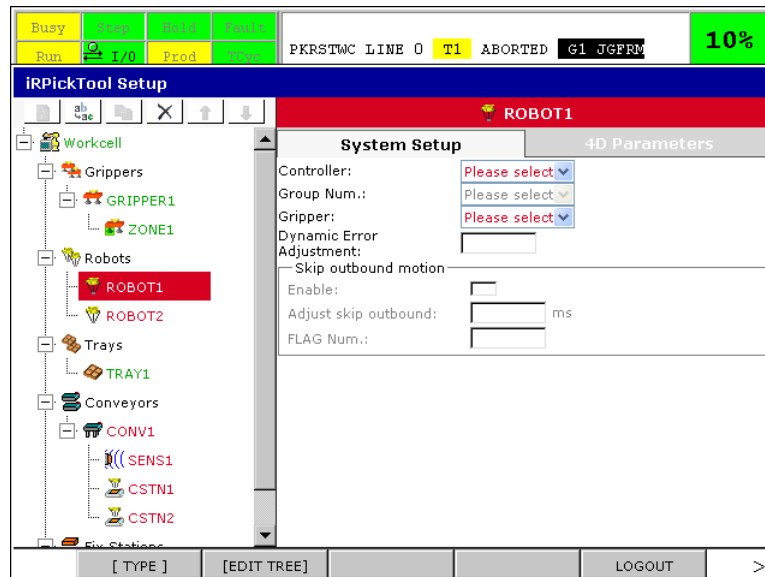
In each row, enter the input I/O for part presence for a zone. Up to 4 rows can be used to enter the I/O input for a zone, if more than one input is used for a zone. The value of the I/O input can be ON or OFF.

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.

6.3 SETUP OF A ROBOT

Set up a robot.



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, see Section 6.13, "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 "6.13.3 Adjustment of the Dynamic Error of the Robot". But this set value is enabled for a conveyor except servo conveyor.

When you use servo conveyor, you set the appropriate value to the following system variable per servo conveyor.

\$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, see Section 7.4, "SKIPPING THE TRACKING MOTION OUTSIDE THE AREA".

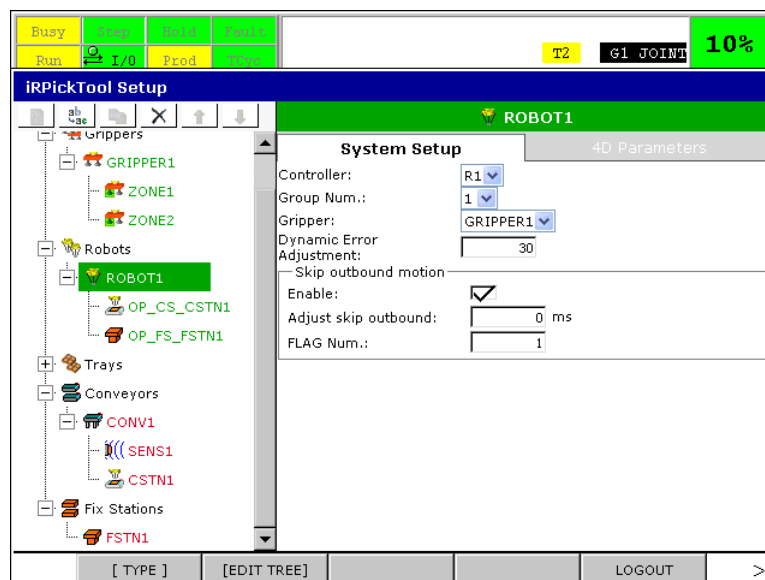
6.4 SETUP OF A ROBOT OPERATION

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 “Create” 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.



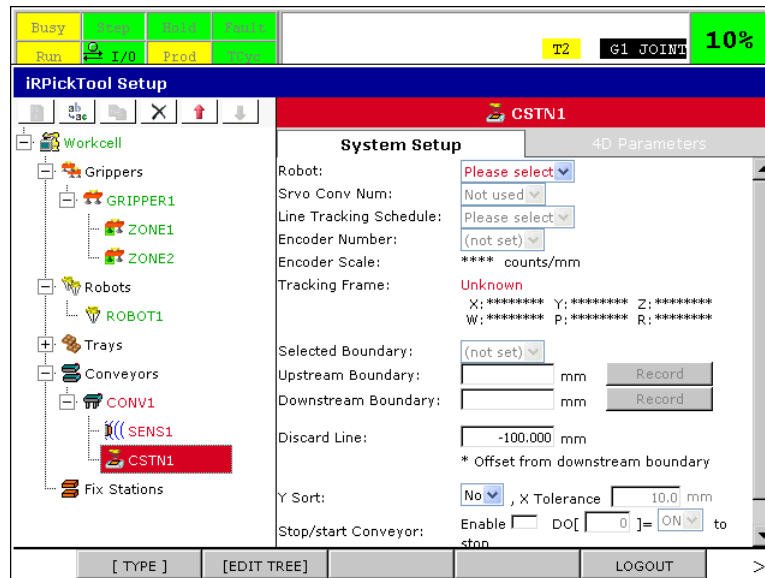
6.4.1 Conveyor Station Robot Operation

Set a Conveyor Station Robot Operation node.

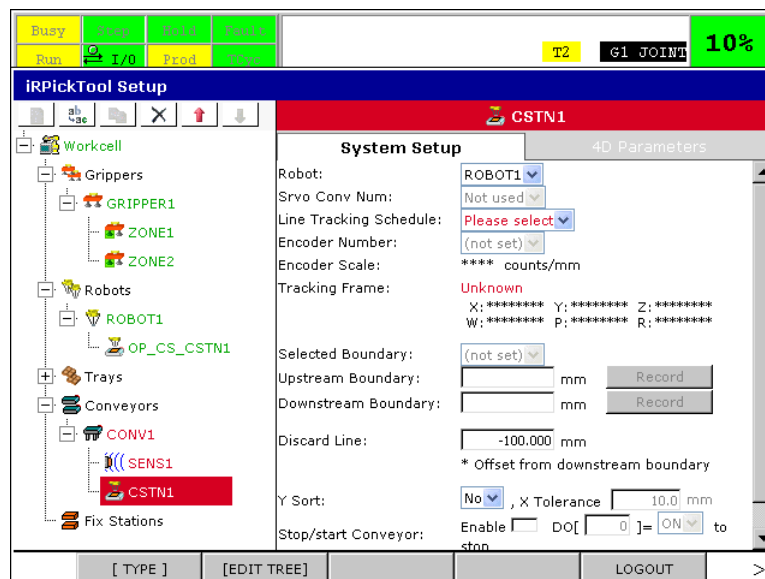
Creating a new Conveyor Station Robot Operation Node

Create a new Conveyor Station Robot Operation Node.

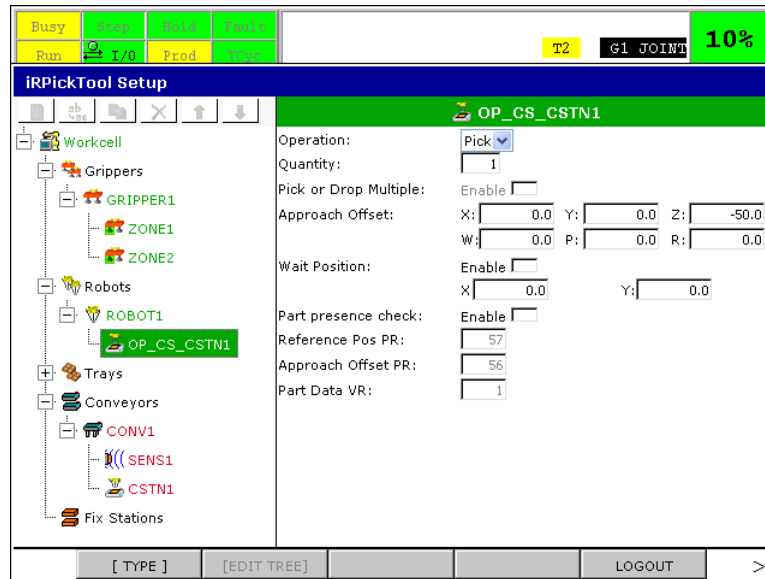
- 1 The Conveyor Station Robot operation node OP_CS_xxxx is created automatically when you assign a Robot to a Conveyor Station. See figure below which shows the display when you create a new conveyor station. Notice that there is no operation node under the Robot yet.



- 2 In the conveyor Station's Robot property select a Robot from the drop-down box as shown in the figure below. Notice that as soon as you did that, the OP_CS_CSTN1 node was immediately added under the Robot automatically.



- 3 Click on the Robot Operation node OP_CS_CSTN1 under the Robot. On the right hand side you will see all the properties as shown in the figure below.



6.4.1.1 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.

Quantity

Enter the number of parts that the robot picks (if the operation = Pick) or the number of parts the robot will place (if operation = Place) in the textbox. The maximum number of parts a robot can handle is 20.

The KAREL macro PKCSGETQY.PC can be used to retrieve the value of the quantity into a Numeric Register.

Pick or Drop Multiple

You can use this property only when the Quantity is greater than one.

If you select this item, then the robot will pick all the parts simultaneously. 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.



CAUTION

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 6.5 “SETUP OF A TRAY” for more information.

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 (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.

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 (or 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 iRPickTool 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 (or 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 (or place) position. This position is calculated automatically by iRPickTool. 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 iRPickTool, 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

iRPickTool 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 6.11, "REFERENCE POSITION SETTING".) This is the pick (or place) position in the tracking frame. You cannot change this Position Register.

Approach Offset PR

iRPickTool 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

iRPickTool uses this dedicated Vision Register to get the part from the part queue using the PKCSGETQUE or PKCSGETQINRG macro. You cannot change this Vision Register.

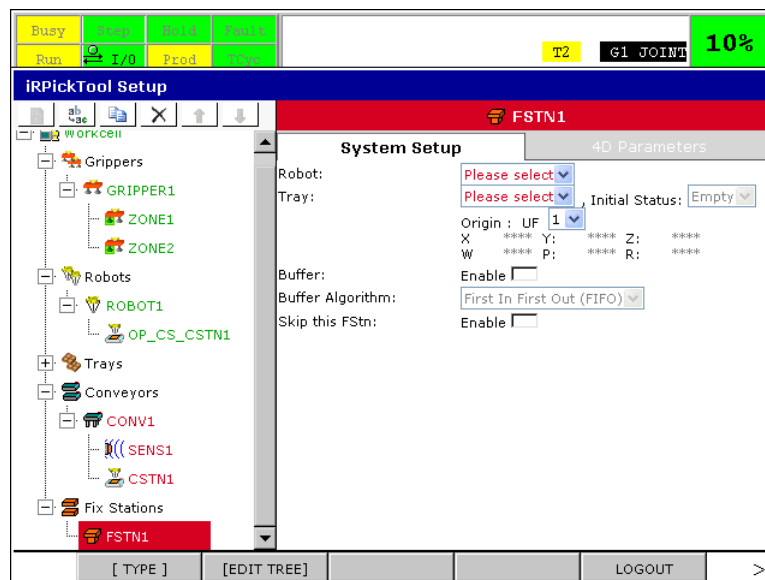
6.4.2 Fixed Station Robot Operation

Set a Fixed Station Robot Operation node.

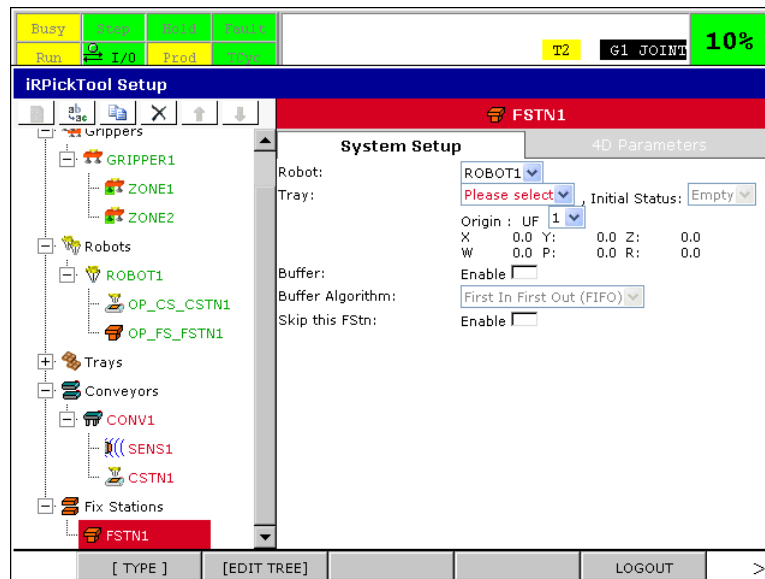
Creating a new Fixed Station Robot Operation Node

Create a new Fixed Station Robot Operation Node.

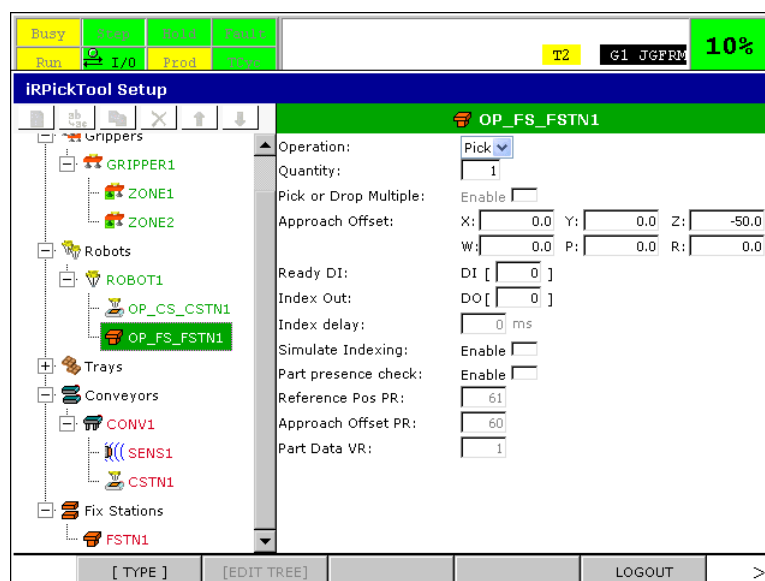
- 1 The Fixed Station Robot operation node OP_FS_XXXX is created automatically when you assign a Robot to a Fixed Station. See figure below which shows the display when you create a new Fixed station. Notice that there is no operation node under the Robot yet.



- 2 In the Fixed Station's Robot property select a Robot from the drop-down box as shown in the figure below. Notice that as soon as you did that, the OP_FS_FSTN1 node was immediately added under the Robot automatically.



- Click on the Robot Operation node OP_FS_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: iRPickTool / Option Package Add-on is installed. The properties in the buffer frame and the operation of the buffer are described in the section on 7.7 “BUFFER FUNCTION”.



6.4.2.1 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.

Quantity

Enter the number of parts that the robot picks (if the operation = Pick) or the number of parts the robot will place (if operation = Place) in the textbox. The maximum number of parts a robot can handle is 20.

The KAREL macro PKFSGETQY.PC can be used to retrieve the value of the quantity into a Numeric Register.

Pick or Drop Multiple

You can use this property only when the Quantity is greater than one.

If you select this item, then the robot will pick all the parts simultaneously. 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.

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 (or 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 *iRPickTool* 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 *iRPickTool* 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 (*iRPickTool* should see the Ready DI go low) and the new empty station should come in (*iRPickTool* 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, *iRPickTool* will automatically simulate the indexing using an Index Delay specified in the property “Index Delay”.

With this feature, *iRPickTool* 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.

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

iRPickTool 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

iRPickTool uses this dedicated Position Register to store the approach offset relative to the Ref Pos in Tool coordinates. You cannot change this Position Register.

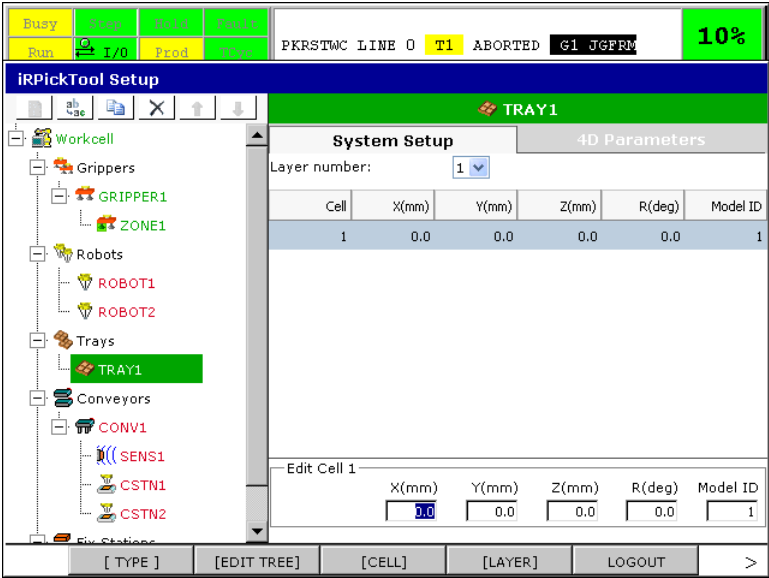
The KAREL macro PKFSGETAPPR.PC can be used to assign the value of the approach offset into this dedicated Position Register.

Part Data VR

iRPickTool 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.

6.5 SETUP OF A TRAY

Set up a tray.



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.

F3 [CELL] key

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.

X, Y, Z and R

Enter the coordinates of the highlighted cell in the tray frame.
If the tray handles four part, set the coordinates as follows.

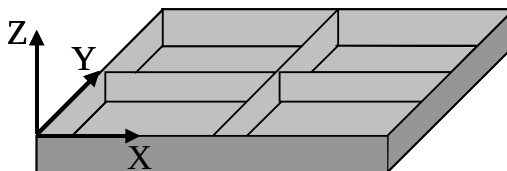
First, decide on the position of the tray frame. For example, suppose that you have decided to place it at a corner of the top side of a box, as shown in the figure below.

NOTE

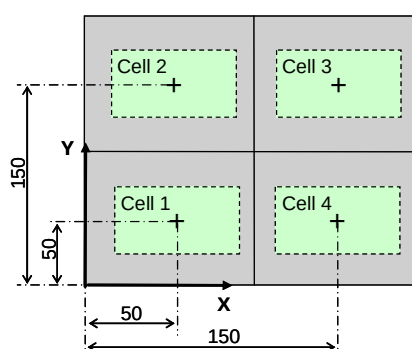
When trays are used on the conveyor, the posture of the tray frame is determined as follows.

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.



Add cells to a tray, and enter the coordinates of each cell. For example, the tray shown in the figure above is as shown in the figure below when viewed from above.



The cross mark (+) represents the center position of each individual cell. The coordinates of these cells are as follows.

Busy Stop Hold Ready Run I/O Prod Stop PKRSTWC LINE 0 T1 ABORTED G1 JGFRM 10%

irPickTool Setup

TRAY1

System Setup 4D Parameters

Layer number: 1

Cell	X(mm)	Y(mm)	Z(mm)	R(deg)	Model ID
1	50.0	50.0	-50.0	0.0	1
2	50.0	150.0	-50.0	0.0	1
3	150.0	150.0	-50.0	0.0	1
4	150.0	50.0	-50.0	0.0	1

Edit Cell 4

X(mm)	Y(mm)	Z(mm)	R(deg)	Model ID
150.0	50.0	-50.0	0.0	1

[TYPE] [EDIT TREE] [CELL] [LAYER] LOGOUT >

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 5, "VARIATIONS".

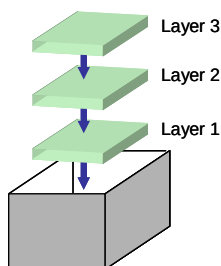
F4 [LAYER] key

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.

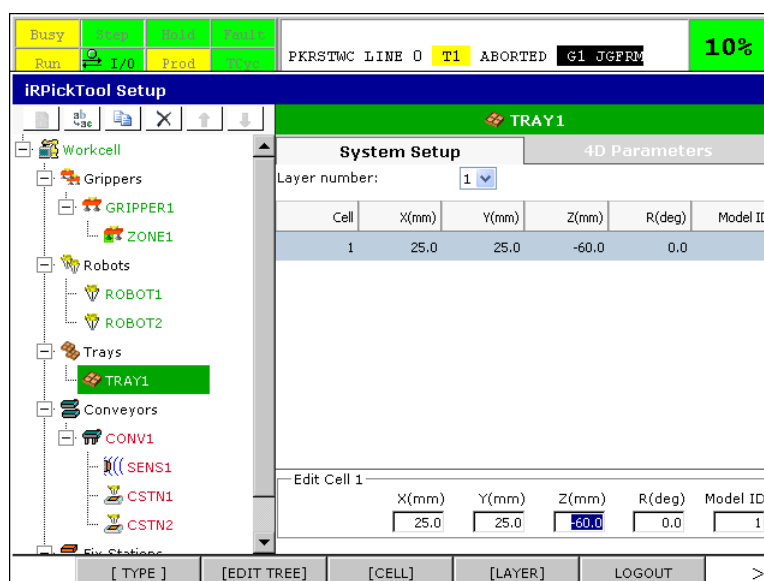
Multiple layers are used when control needs to be exerted over the order in which parts are placed in the tray - e.g., when the tray has multiple layers and a part cannot be placed in an upper layer unless the placement of parts in the lower layer is completed. When placing parts in a tray, set [Initial 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.



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.

iRPickTool Setup

Number of cells:

Default Z(mm):

OK CANCEL

- 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.

iRPickTool Setup

System Setup 4D Parameters

Layer number:

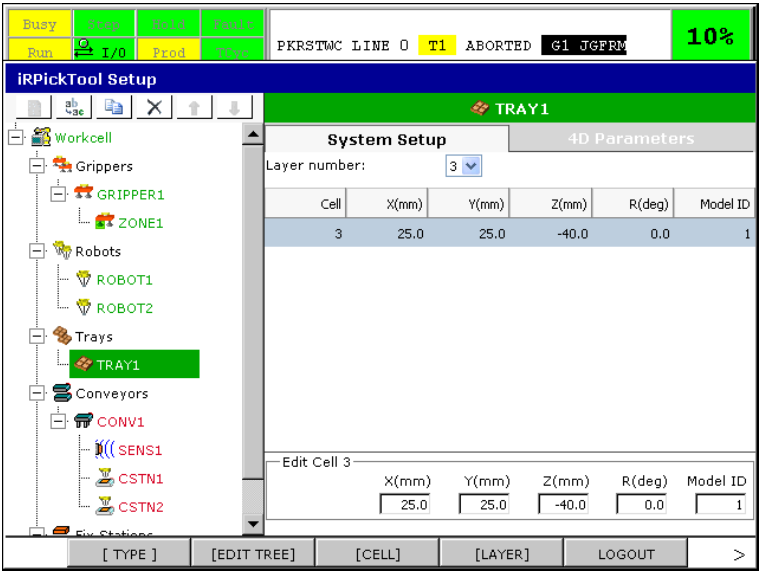
Cell	X(mm)	Y(mm)	Z(mm)	R(deg)	Model ID
2	25.0	25.0	-50.0	0.0	1

Edit Cell 2

X(mm)	Y(mm)	Z(mm)	R(deg)	Model ID
25.0	25.0	-50.0	0.0	1

[TYPE] [EDIT TREE] [CELL] [LAYER] LOGOUT >

- 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.

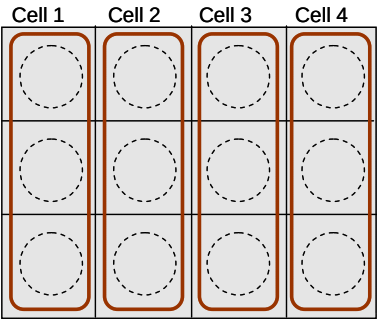
Layer number

When multiple layers are set, the layer number is used to switch the layer whose cell list is displayed.

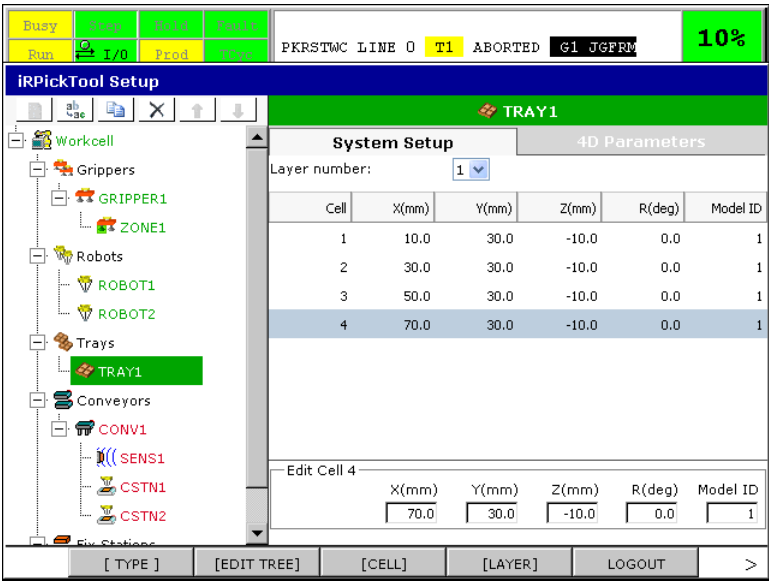
Setting of a tray when a robot hand that can hold multiple parts is used

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.



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.



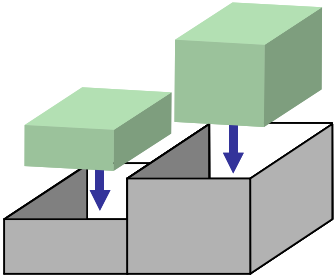
CAUTION

If you set a tray incorrectly, serious effects such that the collision between a robot gripper and the tray can occur when a robot place multiple parts into the tray or picks up multiple parts from the tray simultaneously by using the hand that can hold multiple parts.

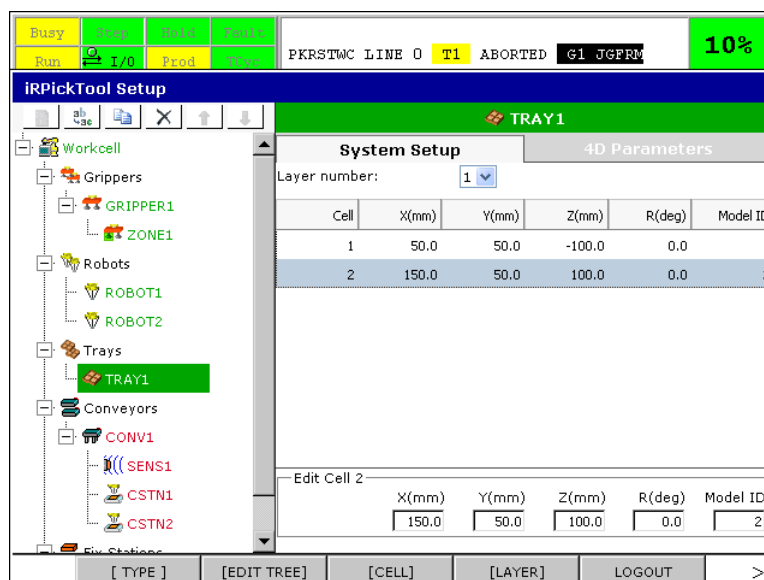
Setting of a tray when parts are different in height

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.

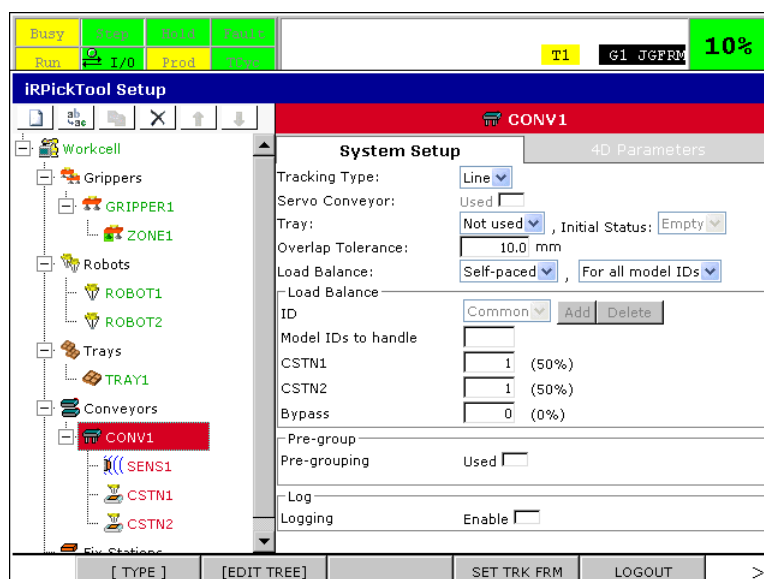


In this case, change the Z value of each cell according to the height of the part. For example, the tray may be set as shown below. Here, since the model of each part is different, different model IDs are set.



6.6 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 6.7, "SETUP OF A SENSOR" and Section 6.8, "SETUP OF A CONVEYOR STATION".



Tracking Type

Select the line tracking or the circular tracking. If you want to use the line tracking, select "Line". Otherwise, select "Circle". If you change it, child objects like the sensor and the conveyor station will be initialized.

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 Appendix D, "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

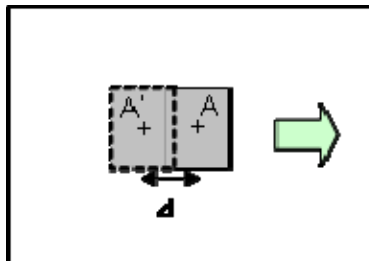
When placing parts into empty trays carried on the conveyor, select [Empty]. When picking up parts from filled trays carried on the conveyor, select [Full].

Overlap 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 any overlap is found, the vision system retains only the part detected earlier. Normally, set this value to 10 mm or so.

In the example shown in the figure below, if Δ is less than the overlap tolerance, the vision system retains only the n-1th part detected, A'.



A': Current position of the n-1th workpiece detected

A : Current position of the nth workpiece detected

Δ : Distance between A and A'



CAUTION

The overlap check may result in the vision system leaving out parts other than the same part detected multiple times as well. If the distance between the current position of the earlier detected part and that of the subsequently detected part is less than the tolerant, the vision system leaves out the subsequently detected part even when it has a different model ID.

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 5, "VARIATIONS".

[Load Balance] box

The [Load Balance] box refers to the portion of the screen that is shown in the figure below.

Load Balance		Load Balance	
ID	Common	ID	1
Model IDs to handle		Model IDs to handle	1
CSTN1	1 (50%)	CSTN1	1 (50%)
CSTN2	1 (50%)	CSTN2	1 (50%)
Bypass	0 (0%)	Bypass	0 (0%)

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 left side of the figure. 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 right side of the figure. 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 6.16, "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 6.16, "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.

Log	
Logging	Enable ☒
Output Device	UD1:\ Output
Output Controller	This Controller

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.

6.6.1 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

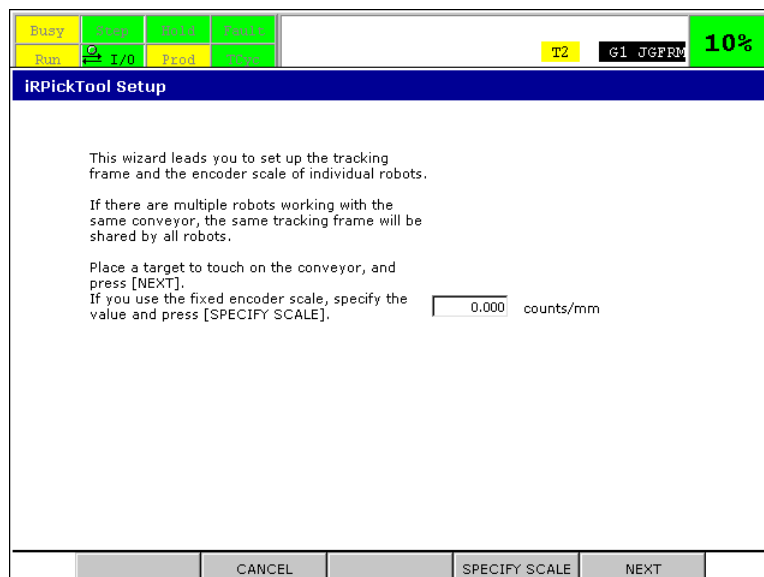
- 1 If you change the tracking frame, you need to perform conveyor station setup, camera calibration, reference position setting, and robot position teaching again.
- 2 When there is more than one robot, you cannot set a tracking frame unless the encoder count values of all the controllers on the conveyor 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 dependind on tracking type.

6.6.1.1 Line conveyor

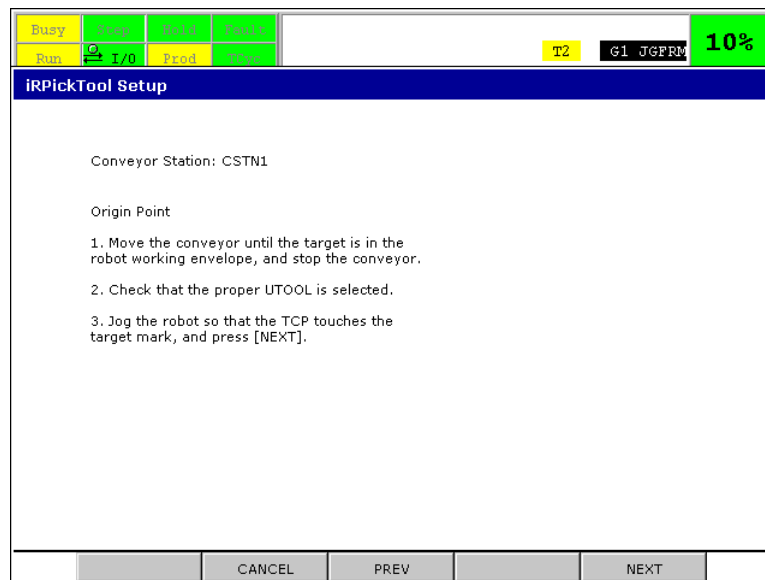
When tracking type is “Line”, please see this procedure. When tracking type is “Circle”, please see the 6.6.1.2, “Circular Conveyor”.

- 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 6.8, "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 *iR*Vision, you can touch up the Y-axis direction point easier with turning the direction of the big circle of the calibration grid to the conveyor moving direction or the direction perpendicular to it



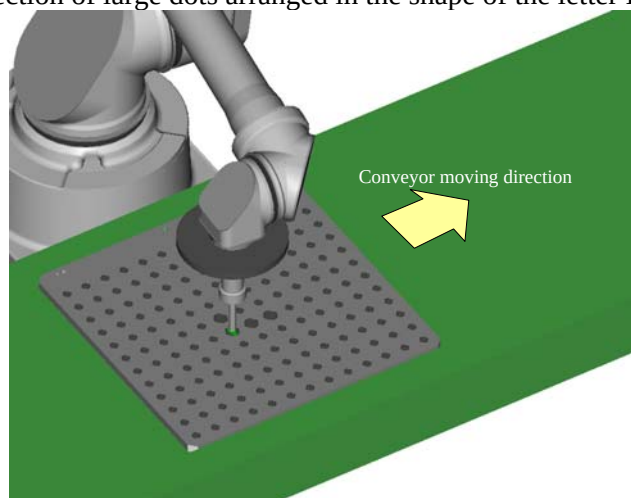
- 4 As the fixture feed frame, select the user tool frame that you set in Section 3.5, "SETTING OF THE TCP".



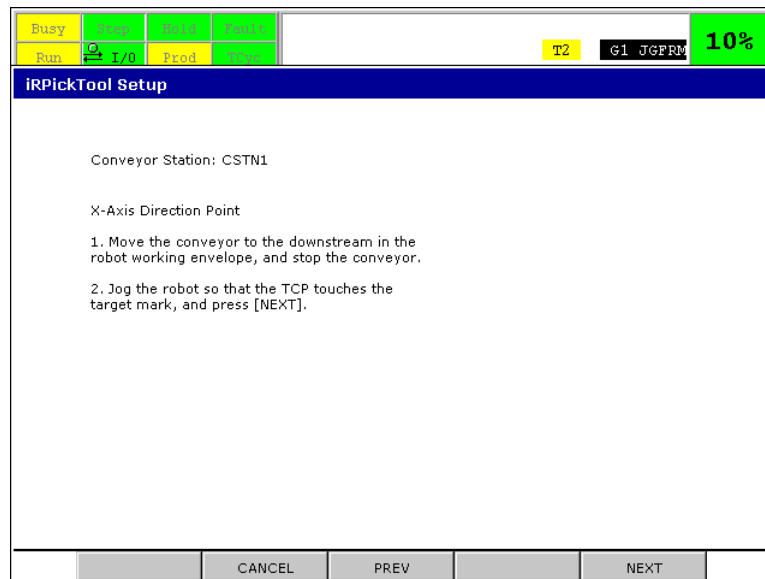
CAUTION

Select the user tool frame with the robot that touches up the fixture. It is not necessarily the robot controller for which the *iRPickTool* setup screen is open.

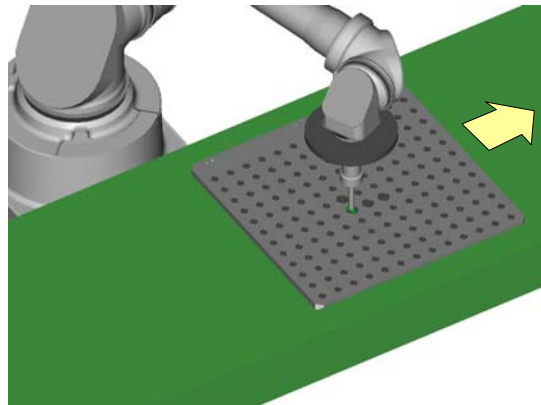
- 5 Move the conveyor until the fixture reaches the upstream area within the robot operation range, and the stop it. Jog the robot to touch up the fixture with the TCP. Here, the robot touches up the dot located at the intersection of large dots arranged in the shape of the letter L on the calibration grid.



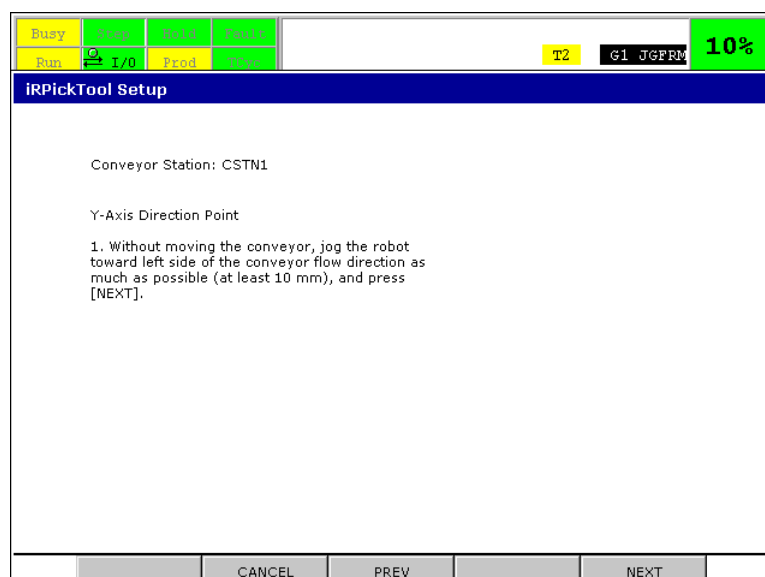
With the fixture touched up, press F5 [NEXT] key. A screen as shown below is displayed.



- 6 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 within the robot operation range and then stop it. Then, jog the robot again to touch up the same part of the fixture that you touched up with the TCP earlier.



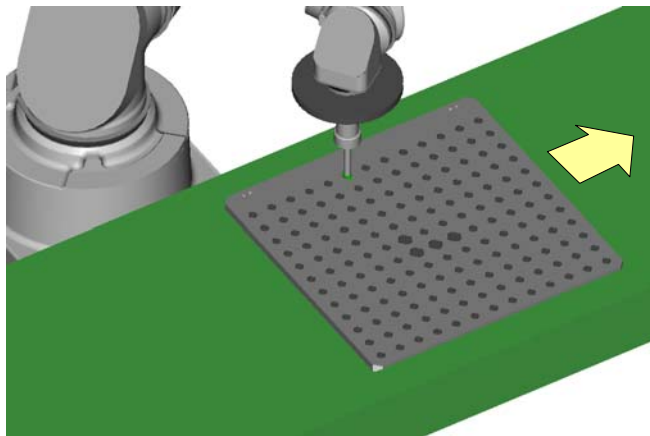
With the fixture touched up, press F5 [NEXT] key. A screen as shown below is displayed.



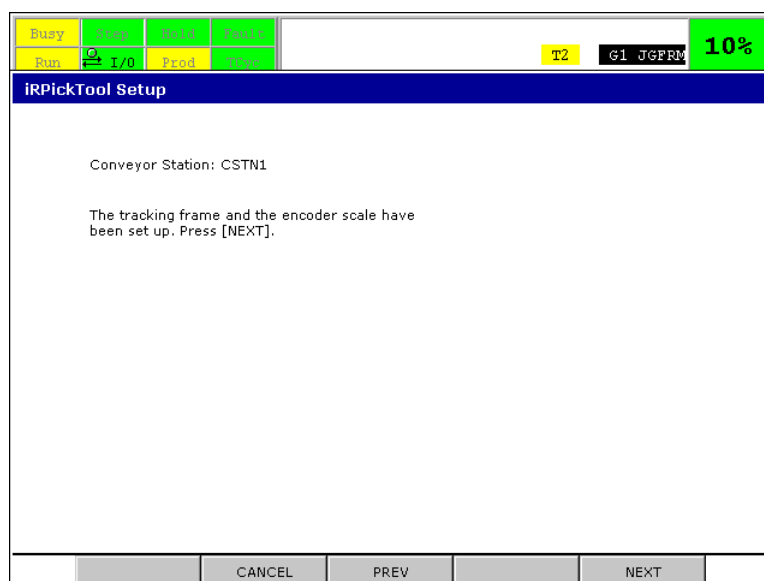
- 7 Jog the robot upward (or away from the conveyor surface) . Next jog the robot to the left as much as possible (at least 10 mm) relative to the conveyor moving direction without moving the conveyor. Then jog the robot downward (or toward the conveyor surface) to touch up a point of the fixture and press F5 [NEXT] key. When doing this, make sure that the TCP is accurately positioned on the plane surface of the fixture.

**CAUTION**

- 1 In the case of a 3-, 4- or 5-axis robot, the Y-axis direction point is used to determine whether the Z-axis direction of the tracking frame is upward or downward. Also, since the plane parallel to the XY plane of the world frame that includes the first touched up point becomes the XY plane of the tracking frame, the world frame of the robot and the conveyor surface need to be parallel to each other.
- 2 If the conveyor is installed vertically, as in the case where parts are hung on the hooks of the conveyor, set the Y-axis direction point by moving the robot down, not to the left, relative to the moving direction.

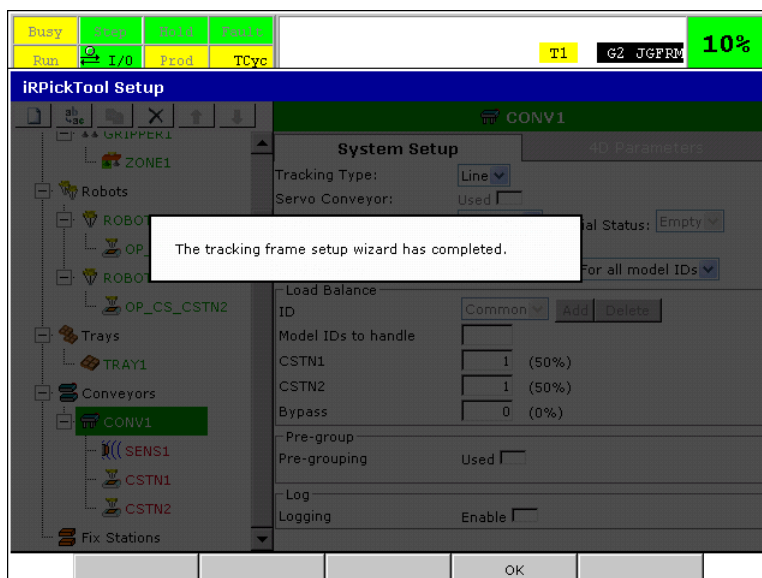


A screen as shown below is displayed.



- 8 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. Repeat steps 4

through 7 for each robot. When you have completed the procedure for all robots, a screen as shown below is displayed.



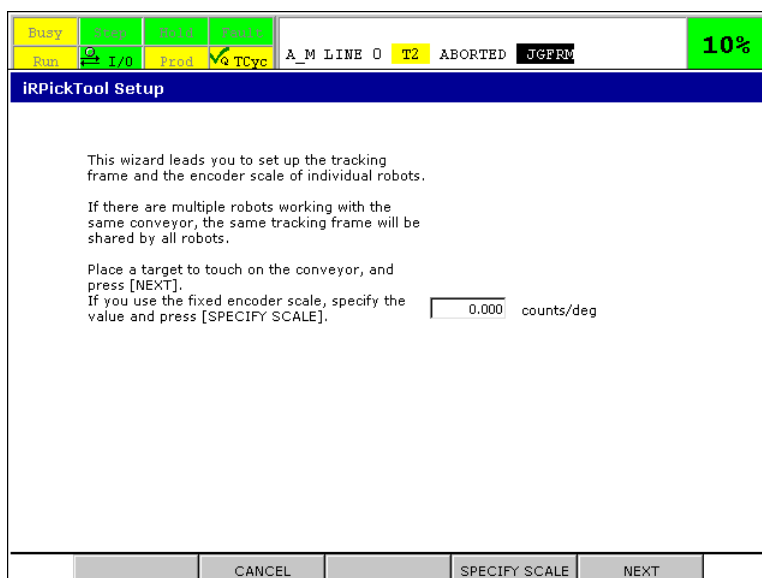
- 9 Press F4 [OK] key to finish the tracking frame setting procedure.

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.

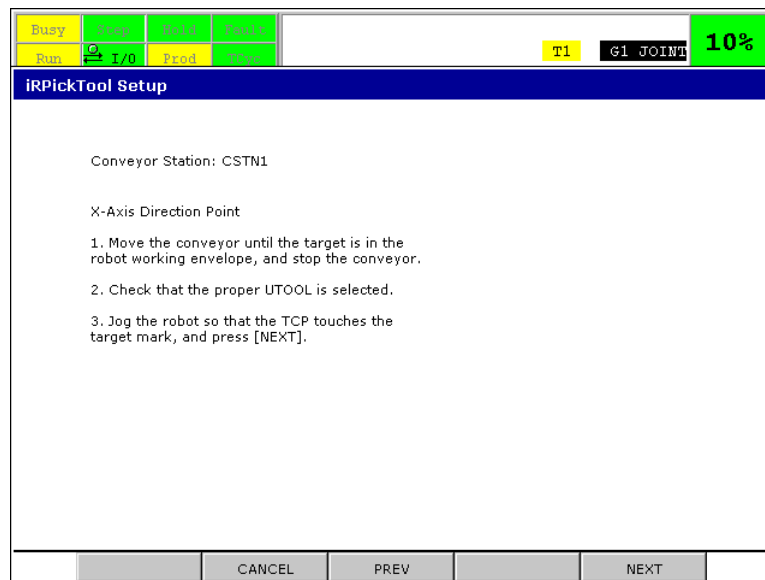
6.6.1.2 Circular conveyor

- 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 6.8, "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 *iRVision*, 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.

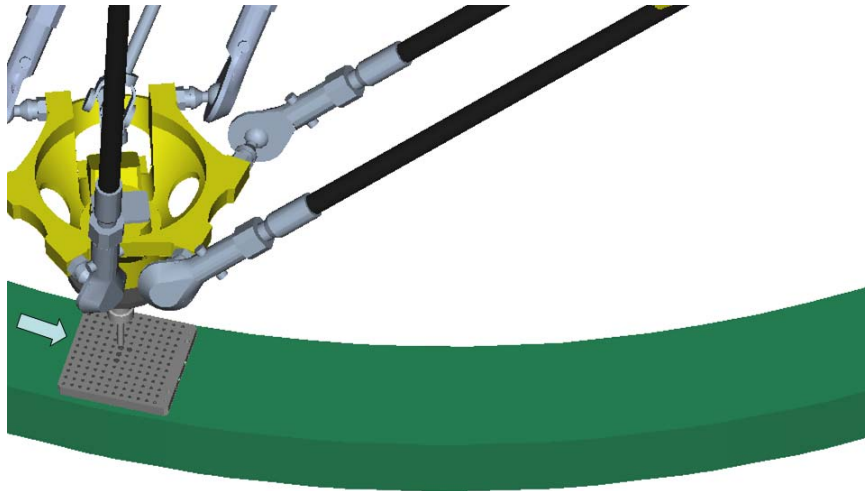
- 4 As the fixture feed frame, select the user tool frame that you set in Section 3.5, "SETTING OF THE TCP".



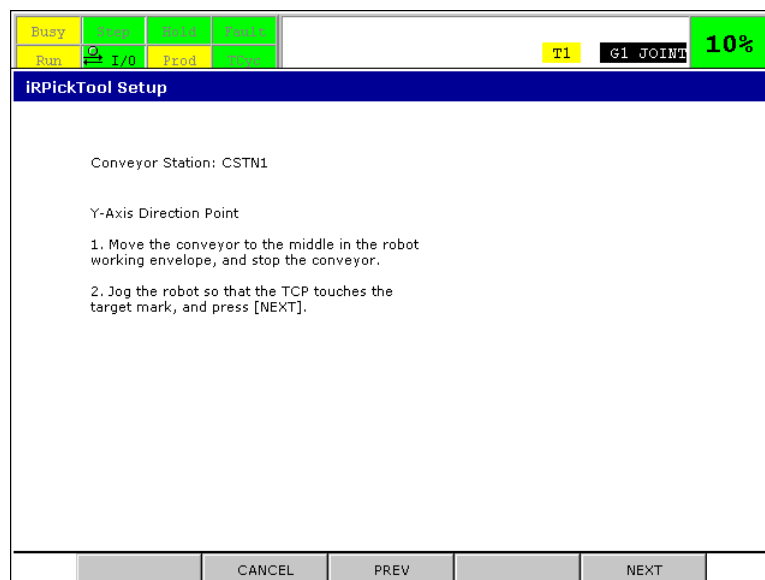
CAUTION

Select the user tool frame with the robot that touches up the fixture. It is not necessarily the robot controller for which the *iRPickTool* setup screen is open.

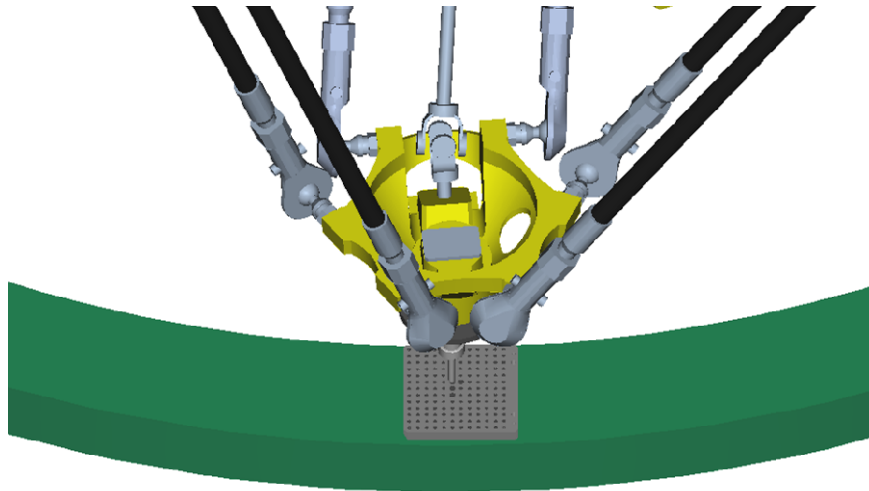
- 5 Move the conveyor until the fixture reaches the upstream area as much as possible within the robot operation range, and the stop it. Jog the robot to touch up the fixture with the TCP. Here, the robot touches up the dot located at the intersection of large dots arranged in the shape of the letter L on the calibration grid.



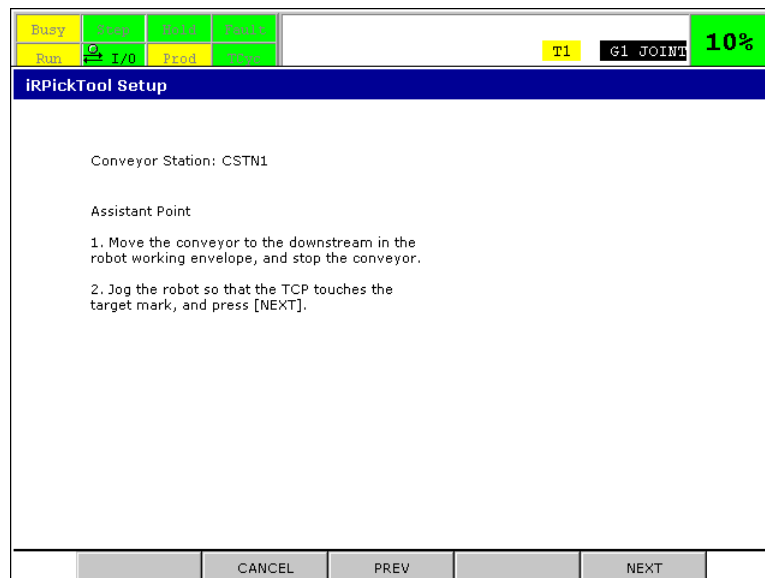
With the fixture touched up, press F5 [NEXT] key. A screen as shown below is displayed.



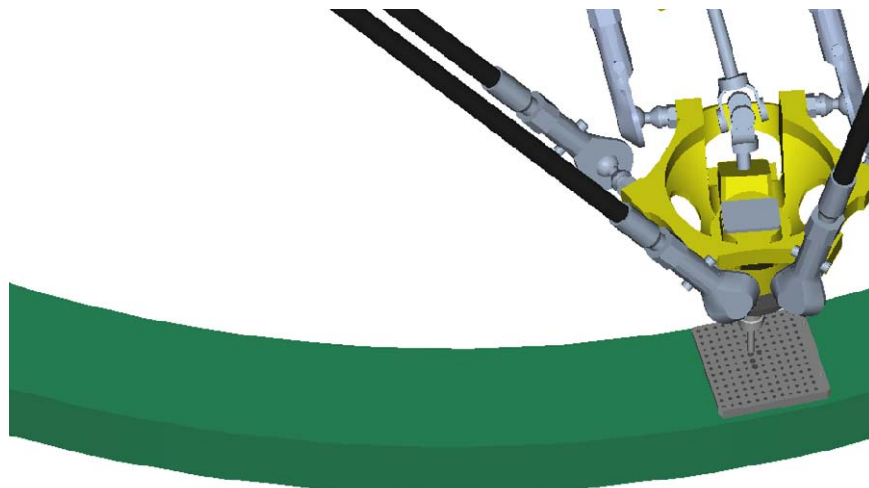
- 6 After moving the robot to a position where the TCP does not interfere when the conveyor is moved, move the conveyor so that the fixture comes to the middle area within the robot operation range and then stop it. Then, jog the robot again to touch up the same part of the fixture that you touched up with the TCP earlier.



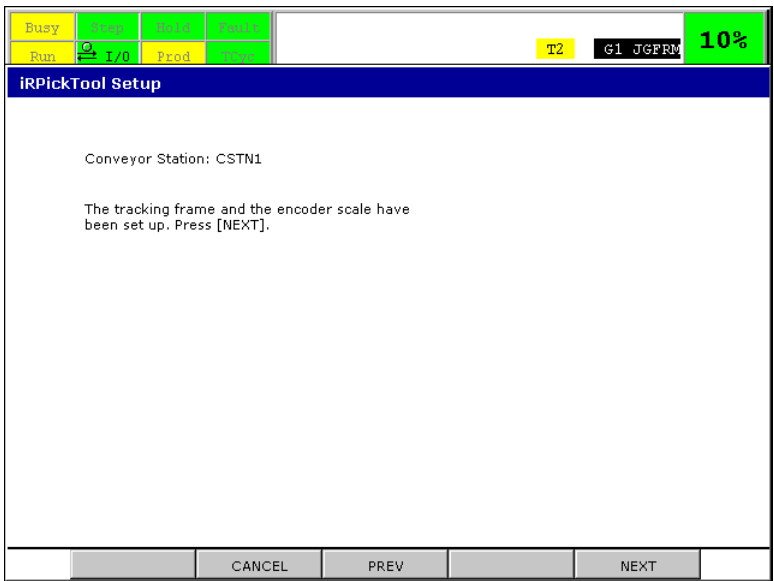
With the fixture touched up, press F5 [NEXT] key. A screen as shown below is displayed.



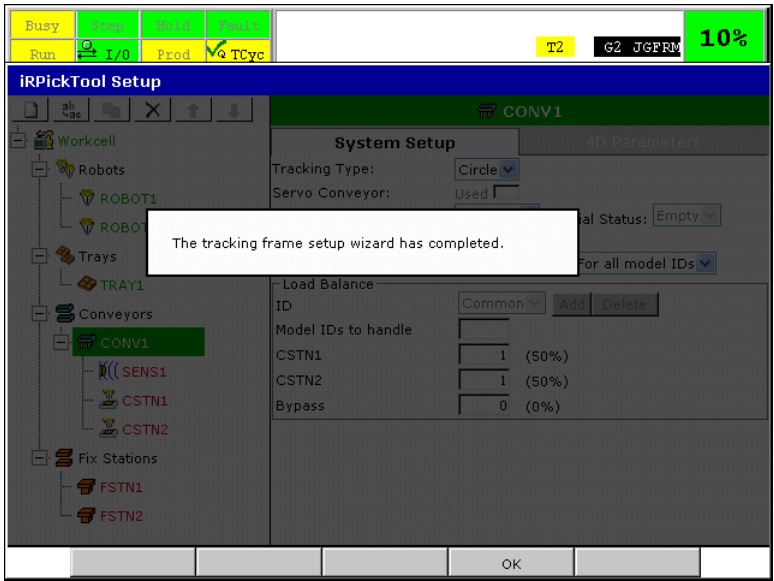
- 7 After moving the robot to a position where the TCP does not interfere when the conveyor is moved, move the conveyor so that the fixture comes to the downstream area as much as possible within the robot operation range and then stop it. Then, jog the robot again to touch up the same part of the fixture that you touched up with the TCP earlier.



With the fixture touched up, press F5 [NEXT] key. A screen as shown below is displayed.



- 8 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. Repeat steps 4 through 7 for each robot. When you have completed the procedure for all robots, a screen as shown below is displayed.



- 9 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.

6.7 SETUP OF A SENSOR

Set up a sensor.

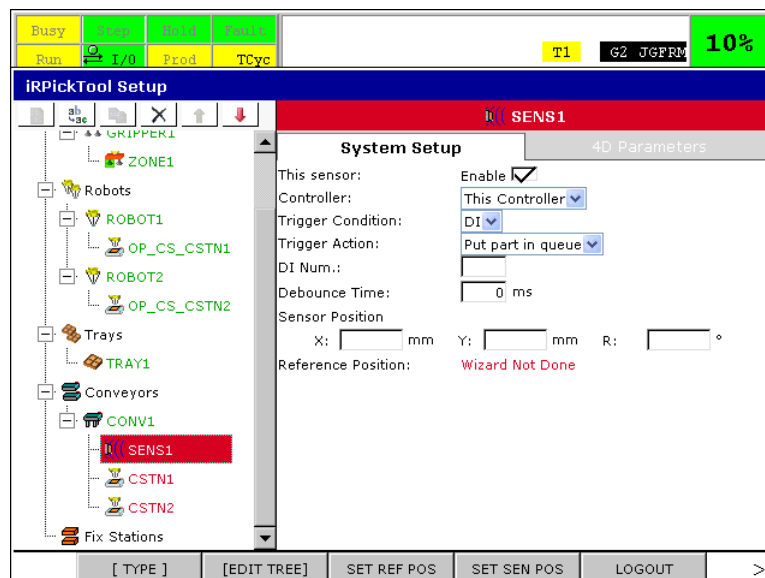


CAUTION

- 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 ("6.11 REFERENCE POSITION SETTING").

6.7.1 Setting a Sensor in Detail

If you select a sensor on the tree, a screen as shown below is displayed.



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.

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.

FLAG

The action is triggered when the flag becomes 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.

Find part by vision

The vision system searches for a part and pushes a detected part into the queue. This action is not available when the trigger condition is [HDI].

Find part by multiview

The vision system searches for a part twice and puts a detected part into the queue. 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 conveyor belt is given as an example.

The possible combinations of a trigger condition and an action are shown below.

	Distance	DI	HDI	FLAG
Put part in queue	O	O	O	O
Find part by vision	O	O	X	X
Find part by multiview	X	O	X	X

Depending on the combination selected, the appearance of the setup screen changes as described below.

NOTE

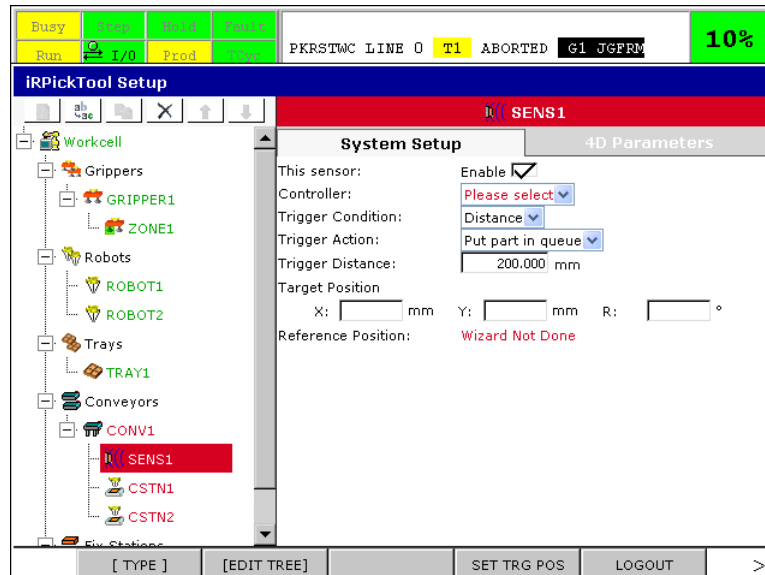
For details of a combination of [DI] + [Find part by multiview], see Section 5.11, "MULTI-VIEW".

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 push 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

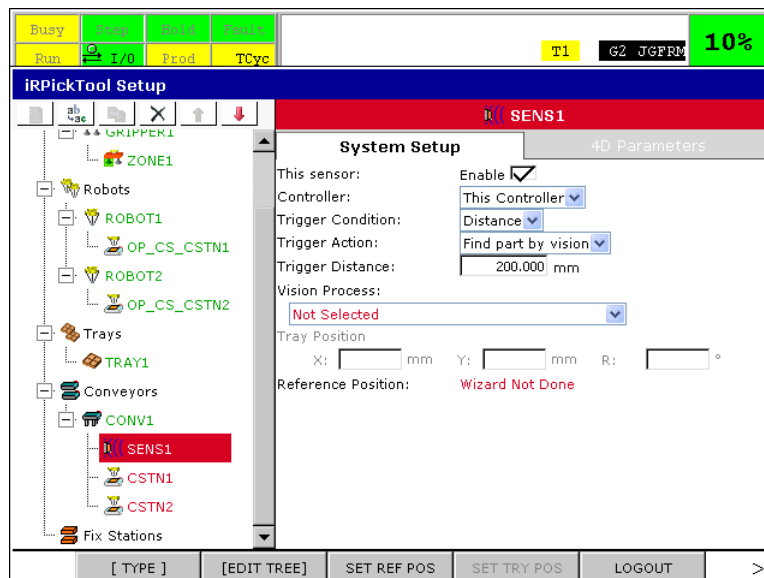
This is the part position that is generated each time the conveyor travels the trigger distance. Set these values using the target position setting wizard described later.

Reference Position

When you finish the wizard to set the target position, “Wizard Done” is displayed in green.

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.



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 7.2.2 “Checking the Status of a Sensor”.

Vision Process

Enter the name of the vision process to be executed.

Tray Position

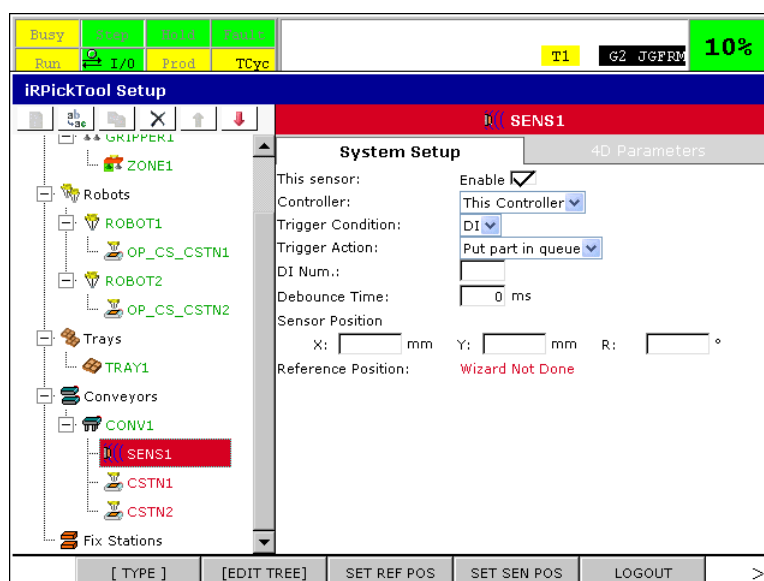
This is the relative position of the tray detection position and the origin of the tray. Set these values using the tray position setting wizard described later.

Reference Position

When you finish the wizard to set the reference position, “Wizard Done” is displayed in green.

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

While you can also set [Trigger INPUT Number] using the tracking schedule setting screen of the teach pendant of the robot, use only the sensor task setup screen to set the DI number.

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.

Sensor Position

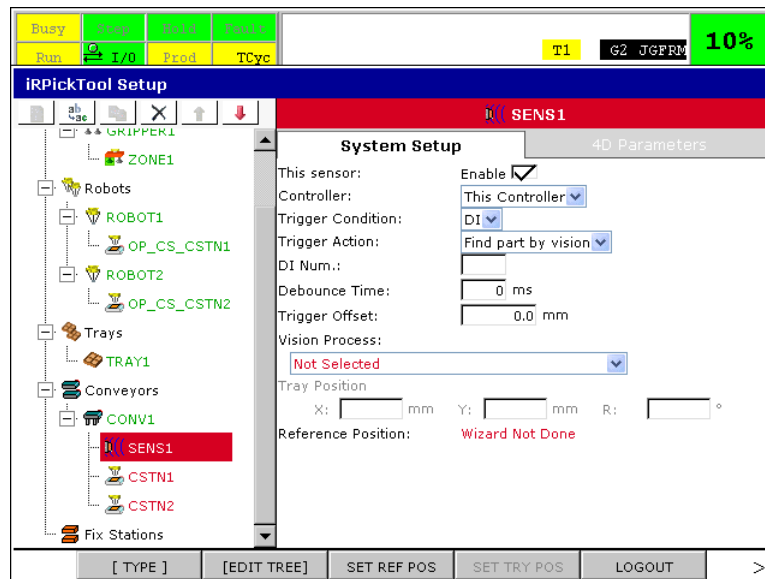
When a part is detected, the position specified here is used as the detection position as the part is put into the queue. Set these values using the sensor position setting wizard described later.

Reference Position

When you finish the wizard to set the reference position, “Wizard Done” is displayed in green.

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

While you can also set [Trigger INPUT Number] using the tracking schedule setting screen of the teach pendant of the robot, use only the sensor task setup screen to set the DI number.

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.

Vision Process

Specify the name of the vision process to be executed.

Tray Position

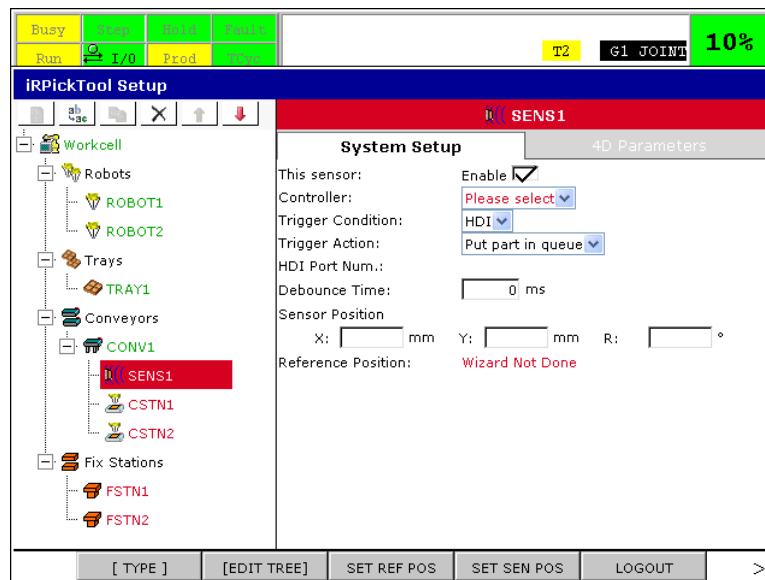
This is the relative position of the tray detection position and the origin of the tray. Set these values using the tray position setting wizard described later.

Reference Position

When you finish the wizard to set the reference position, "Wizard Done" is displayed in green.

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 wizard described later.

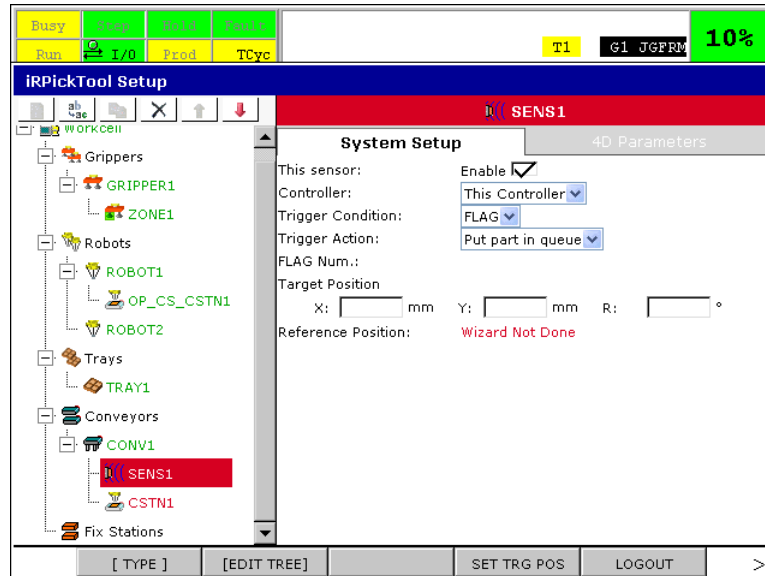
Reference Position

When you finish the wizard to set the reference position, "Wizard Done" is displayed in green.

When [FLAG] and [Put part in queue] are selected

This is used only for the servo conveyor.

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.

iRPickTool 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 becomes ON. Set these values using the target position setting wizard described later.

Reference Position

When you finish the wizard to set the target position, "Wizard Done" is displayed in green.

6.7.2 Settings Required for Each Combination of a Trigger Condition and a Trigger Action

The sensor setup procedures described in Subsection 6.7.3, "Setting the Sensor Position" through Subsection 6.7.6, "Setting the Target Position" involve operating robots and the vision system. The required settings differ depending on the combination of a trigger condition and an action. See the table below to find out which settings you need to make.

	6.7.3 Sensor Position	6.7.4 Vision System	6.7.5 Tray Position	6.7.6 Target Position
[Distance]+[Put part in queue]				
Use Tray	x	x	x	○
Not Use Tray	x	x	x	○
[Distance]+[Find part by vision]				
Use Tray	x	○	○	x
Not Use Tray	x	○	x	x
[DI]+[Put part in queue]				
Use Tray	○	x	x	x
Not Use Tray	○	x	x	x
[DI]+[Find part by vision]				
Use Tray	x	○	○	x
Not Use Tray	x	○	x	x
[HDI]+[Put part in queue]				
Use Tray	○	x	x	x
Not Use Tray	○	x	x	x
[FLAG]+[Put part in queue]				
Use Tray	x	x	x	○
Not Use Tray	x	x	x	○

6.7.3 Setting the Sensor Position

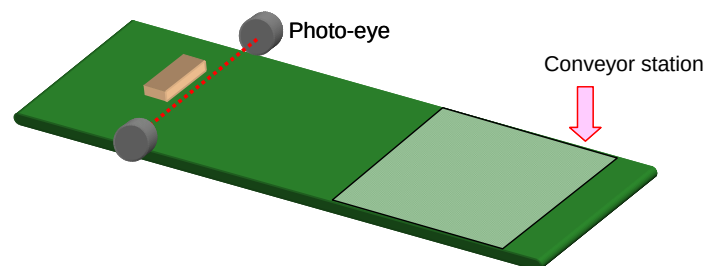
Teach the position where the photo eye is installed. This setting is required when [DI] or [HDI] is selected for [Trigger Condition] and [Put part in queue] is selected for [Trigger Action]. When a part passes in front of the photo eye, this position is put into the queue. The procedure slightly differs depending on whether a tray is used on the conveyor or not.

6.7.3.1 When a tray is not used

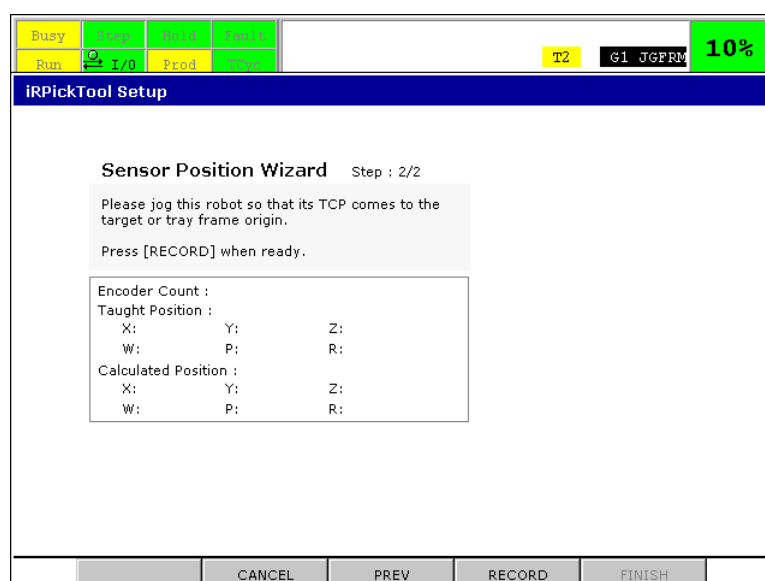
- 1 Stop the conveyor.
- 2 Press F4 [SET SEN POS] key. A screen as shown below is displayed.



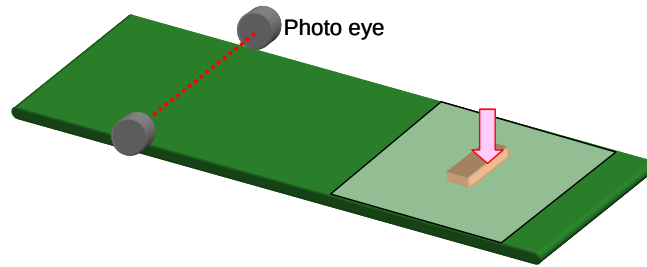
- 3 Place the part upstream of the photo eye, and move the conveyor to let the part pass in front of the DI sensor.



- 4 When the sensor detects the part, the encoder value obtained at the time of detection 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 until the part comes in front of the robot.
- 7 Jog the robot, and touch up the downstream-side end of the part (the part detected by the photo eye) with the TCP.



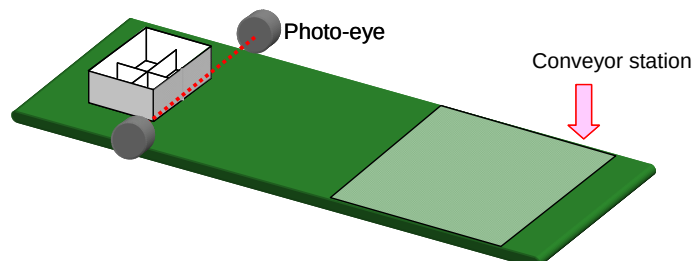
- 8 Press F4 [RECORD] key. The encoder value obtained at the time of touch-up, the touch-up position, and the calculated position are displayed, and F5 [FINISH] key is enabled.
- 9 If you press F5 [FINISH] key, the initial setup screen is displayed again. The calculated value is input for X of the sensor position (the values of Y and R remain unchanged from 0).

6.7.3.2 When a tray is used

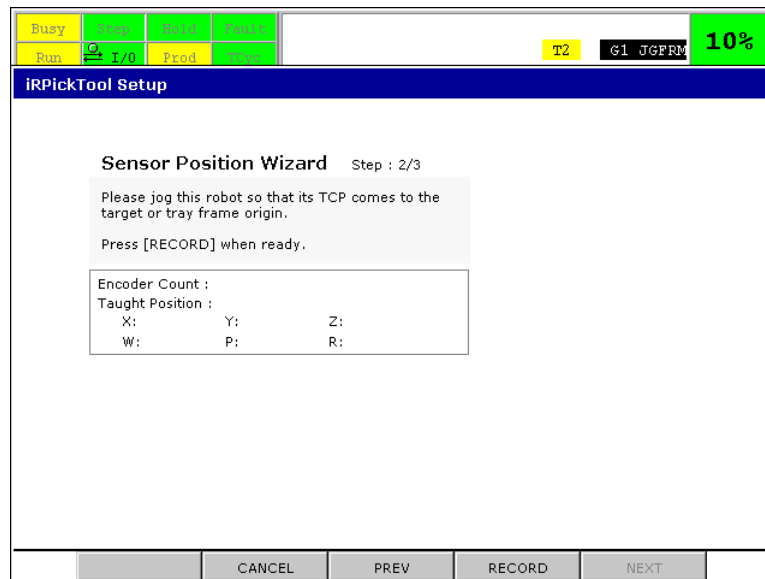
- 1 Stop the conveyor.
- 2 Press F4 [SET SEN POS] key. A screen as shown below is displayed.



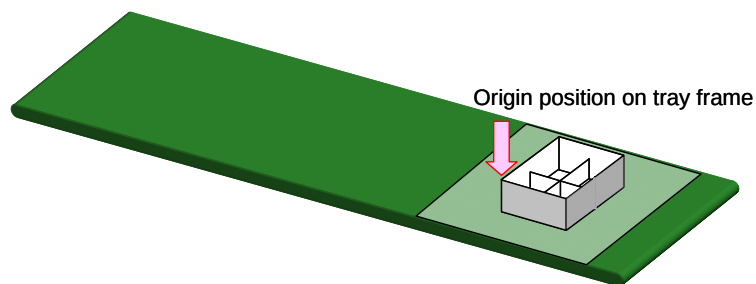
- 3 Place a tray upstream of the photo eye, and move the conveyor so that the tray passes in front of the photo eye.



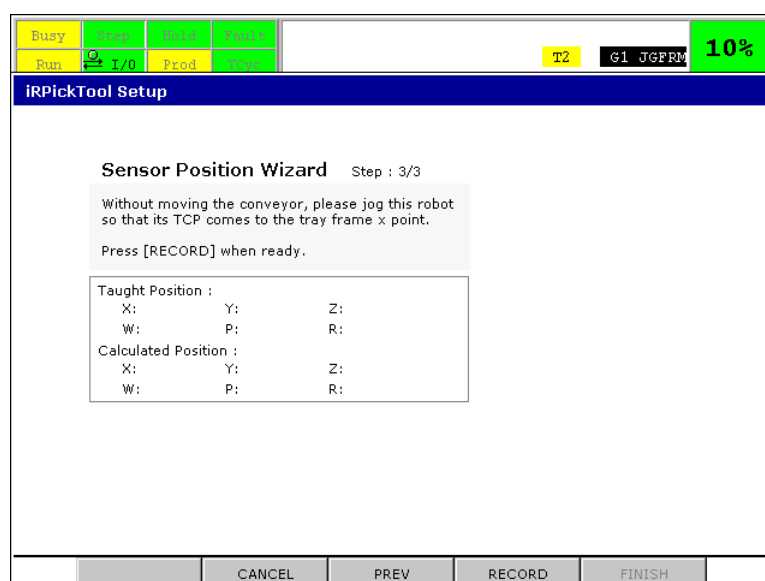
- 4 When the sensor detects the tray, the encoder value obtained at the time of detection is displayed and F5 [NEXT] key is enabled.
- 5 If you press F5 [NEXT] key, a screen as shown below is displayed.



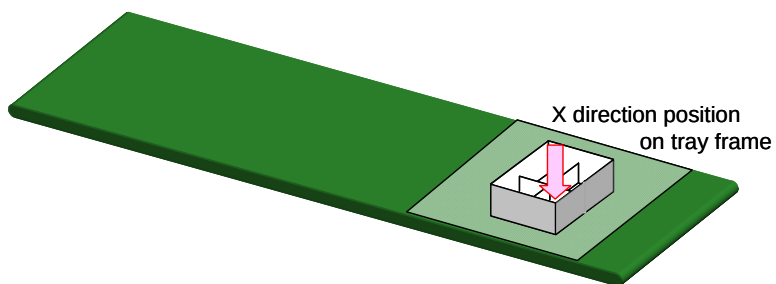
- 6 Move the conveyor until the tray comes in front of the robot.
- 7 Jog the robot, and touch up the origin of the tray frame.



- 8 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.
- 9 If you press F5 [NEXT] key, a screen as shown below is displayed.



- 10 Jog the robot, and touch up the X direction point of the tray frame. Press F4 [RECORD] key. The touch-up position and the calculated position are displayed, and F5 [FINISH] key is enabled.



- 11 If you press F5 [FINISH] key, the initial setup screen is displayed again. Values are input for X, Y, and R of the sensor position.

6.7.4 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].



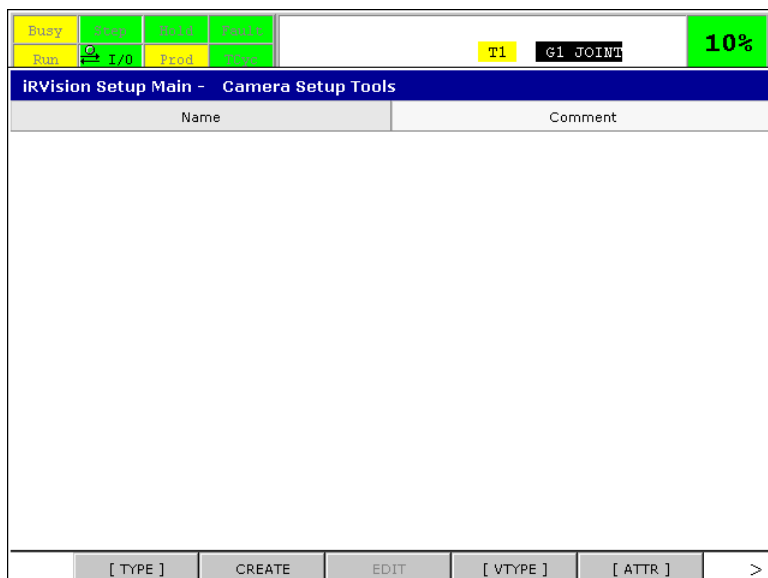
CAUTION

To set up the vision system, you need to use a controller to which a camera is connected.

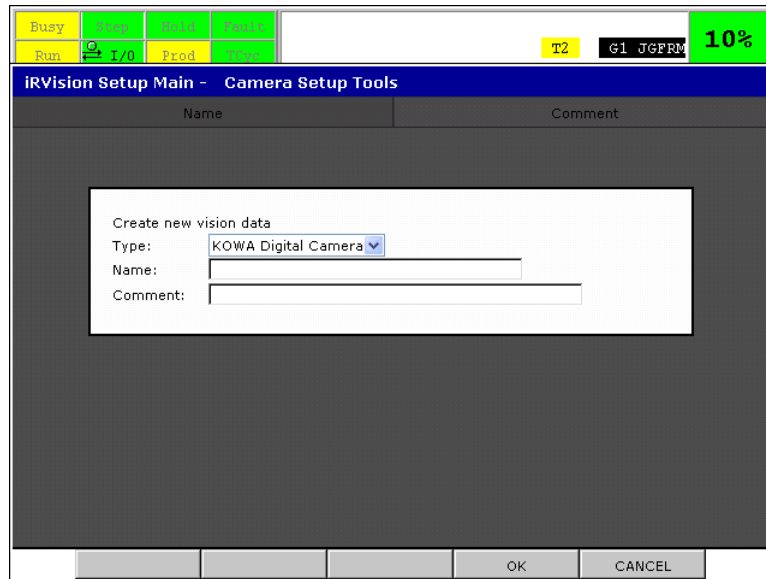
6.7.4.1 Setting up the camera

Set up the camera

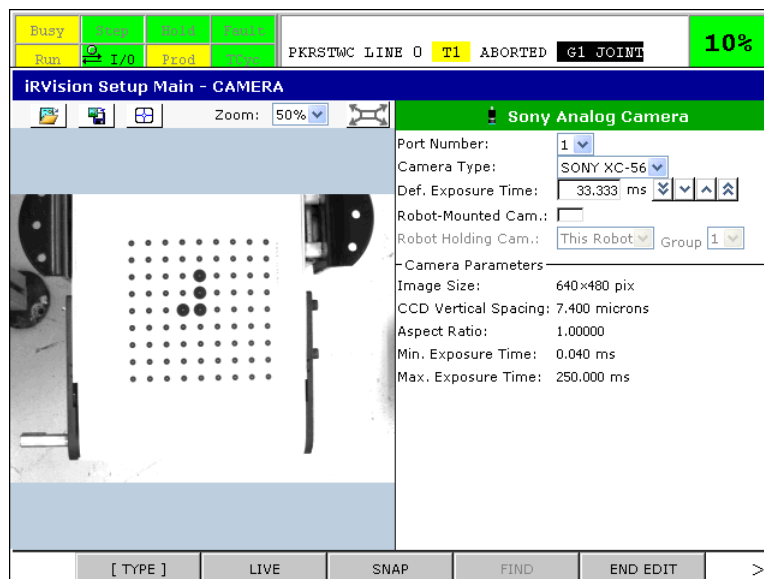
- 1 From the [MENU] key, select [8 iRVision] and then [1 Vision setup] to display the vision data list screen. Press F4 [VTYPE] key, and select [Camera Setup Tools] from the list. A screen as shown below is opened.



- 2 Press F2 [CREATE] key. A screen as shown below is displayed.



- 4 In [Type], select the type of the camera. In the above example, [Sony Analog Camera] is selected. Regardless of monochrome or color, when using a digital camera, select [KOWA Digital Camera].
- 5 In [Name], enter the name of the camera (e.g., CAMERA).
The camera name may consist of up to 34 alphanumeric characters and must not begin with a numeric character.
- 6 Press F4 [OK] key. The created camera data is displayed in the list.
- 7 Select the created camera data, and press F3 [EDIT] key. A screen as shown below is displayed.



- 8 Press F3 [SNAP] key to check that a camera image can be snapped normally.
- 9 Place a part in the camera's field of view. Press F3 [SNAP] key again, and check the image. Make sure that the part is within the field of view and that the image is one that is viewed from above. If the part is not snapped normally, adjust the position of the camera.
- 10 Press F2 [LIVE] key to adjust the focus and diaphragm of the camera.
- 11 Press F5 [END EDIT] key to close the edit screen.

6.7.4.2 Calibrating the camera

Calibrate the camera.

The iRPickTool only supports vision tracking calibration.

By calibrating the camera, you can convert the position data on the detected image into position data on the tracking frame.

As you calibrate the camera, the camera frame is associated with the tracking frame.

Setting the calibration parameters

- 1 From the [MENU] key, select [8 iRVision] and then [1 Vision setup] to display the vision data list screen. Press F4 [VTYPE] key, and then select [Camera Calibration Tools]] from the list.
- 2 Press F2 [CREATE] key. A screen as shown below is displayed.

- 3 In [Type], select [Camera Calib. For Vis. Track].
- 4 In [Name], enter the name of the calibration data (e.g., VTCALIB).
The name may consist of up to 34 alphanumeric characters and must not begin with a numeric character.
- 5 Press F4 [OK] key. The created calibration data is displayed in the list.
- 6 Select the created calibration data, and press F3 [EDIT] key. A screen as shown below is displayed.

- 7 Select [Camera].
- 8 Select [Conveyor Name].

- 9 Adjust the value in [Exposure Time].
- 10 In [Grid spacing], enter a desired value for the grid pattern.
- 11 In [Projection], select [Perspective].

**CAUTION**

[Orthogonal] can be selected only when the part to be detected is uniform in height and when the grid pattern surface is at the same height as the top surface of the part to be detected.

- 12 In [Override Focal Distance], select the calculation method. Select [Yes].

**CAUTION**

If you select [No] in [Override Focal Distance], the focal length may fail to be calculated correctly in principle because the calibration grid on the conveyor is almost parallel to the imaging area of the camera.

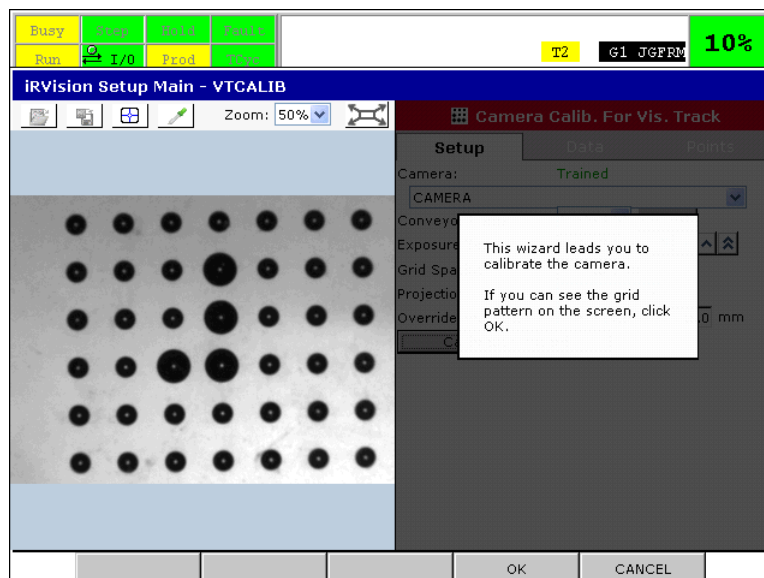
- 13 Set the nominal focal length of the lens.

**CAUTION**

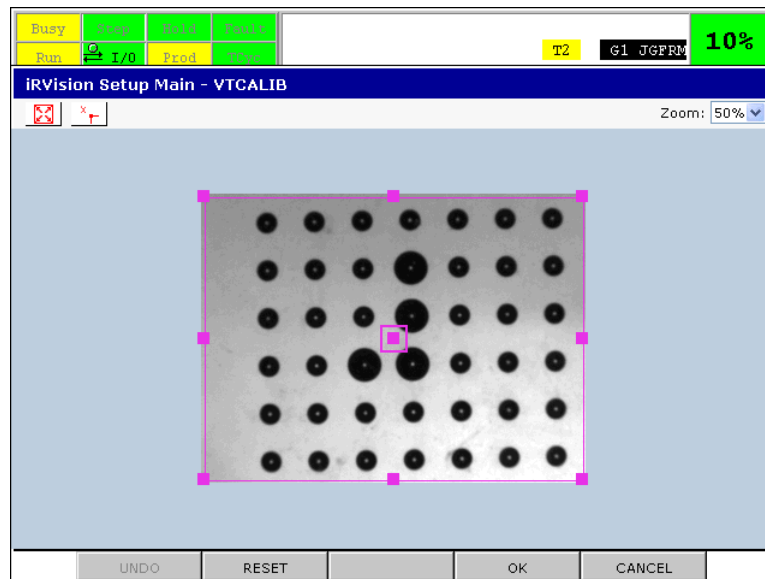
Check the focal length of the camera in use. If you set an incorrect focal length here, you cannot calibrate the camera properly.

Executing calibration

- 1 Stop the conveyor, and place the calibration grid in the field of view of the camera. Then, click the [Calibration Wizard] button. A screen as shown below is displayed.

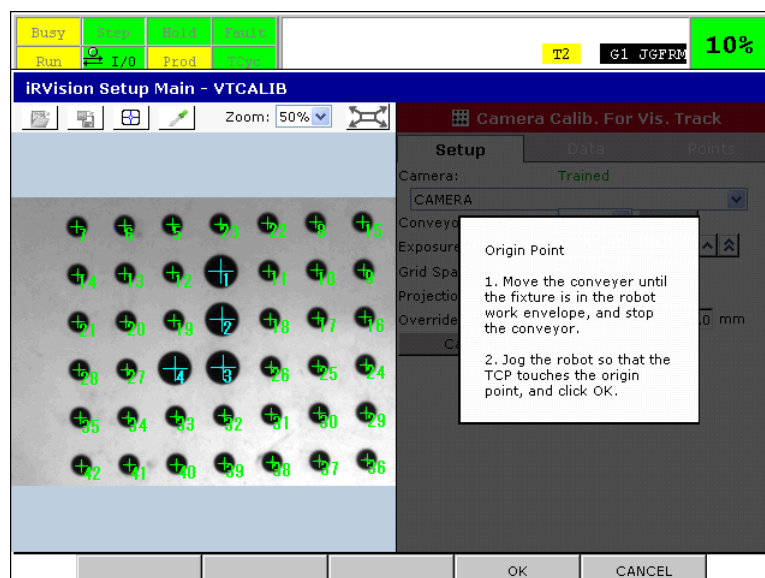


- 2 Check that the grid pattern of the calibration grid is displayed on the screen, and then press F4 [OK] key. A screen as shown below is displayed.

**NOTE**

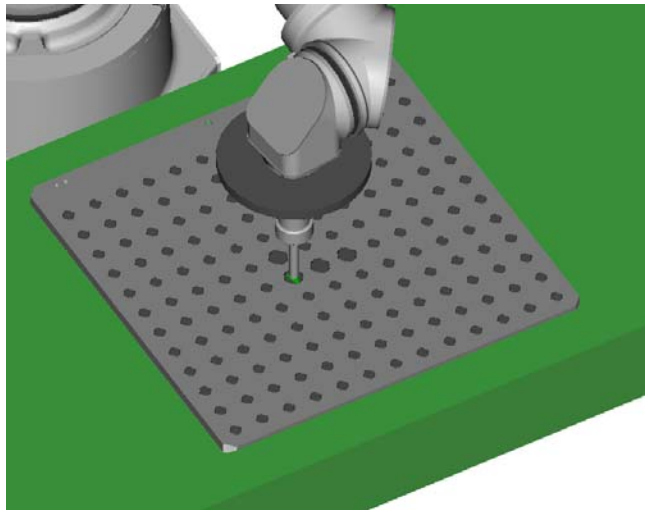
Use a calibration grid that is large enough to fill the entire field of view, and the lens distortion will be calculated more accurately. This leads to improved accuracy when you handle a part that appears in a corner of the camera's field of view.

- 3 Enclose the grid pattern with a magenta rectangular window, and press F4 [OK] key. When the grid pattern is detected, a screen as shown below is displayed.

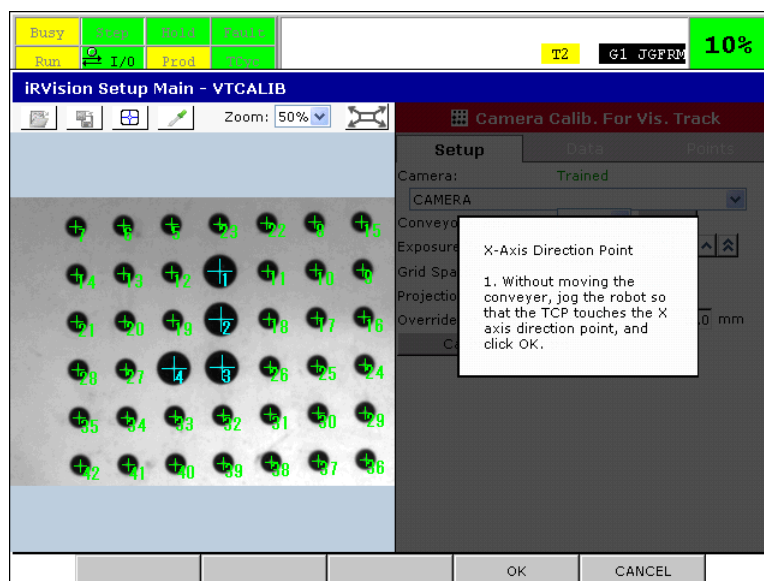


- 4 Check that the number of the user tool frame that you set in Section 3.4, "SETTING OF THE TCP" is selected.
- 5 Move the conveyor so that the calibration grid comes in front of the robot for which you are setting up the vision system.

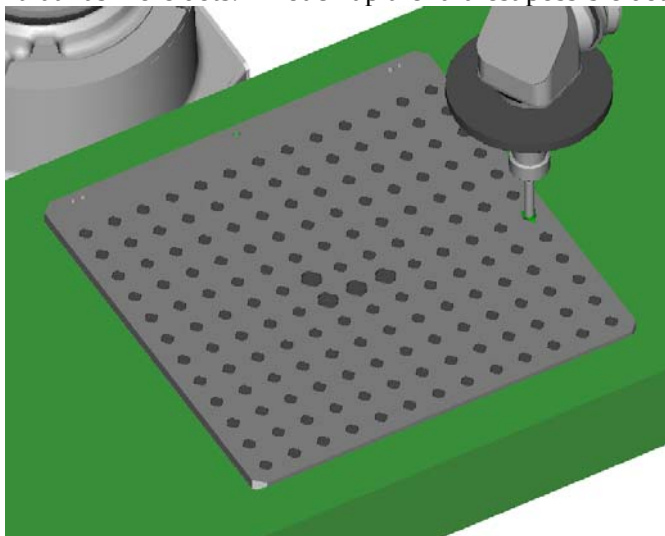
- 6 Touch up the origin of the calibration grid. The origin is located at the intersection of large dots arranged in the shape of the letter L at the center.



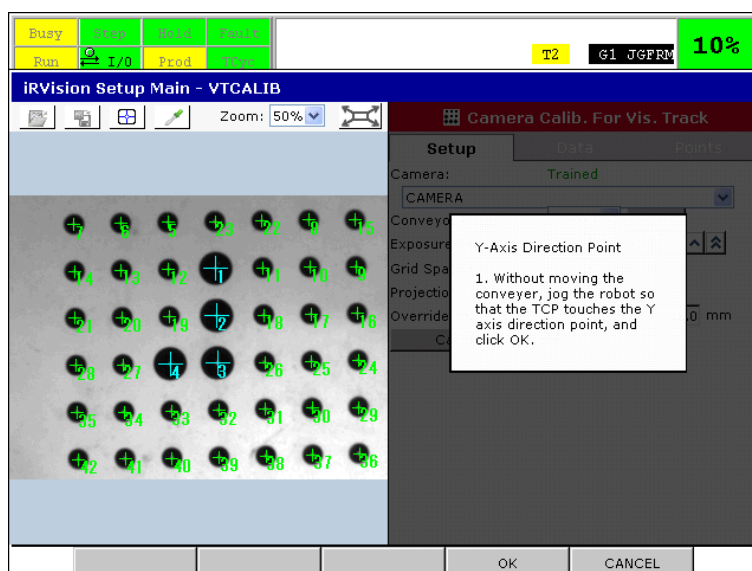
- 7 Press F4 [OK] key with the origin touched up. A screen as shown below is displayed.



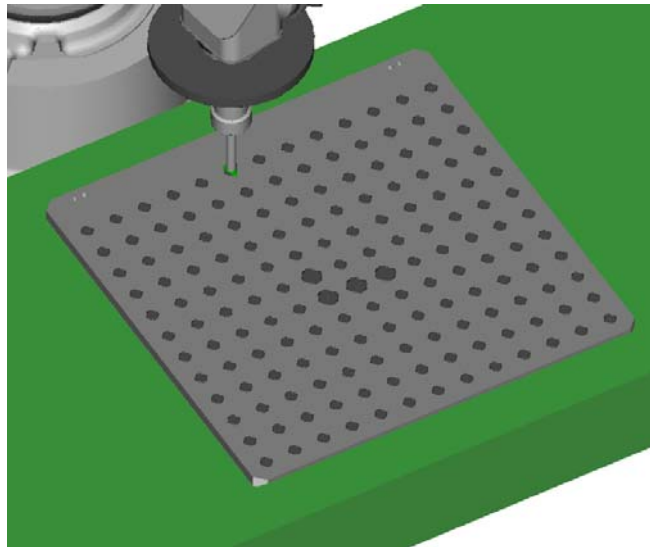
- 8 Jog the robot upward (or away from the conveyor surface) . Next touch up an X-axis direction point with the TCP. The X-axis direction is the direction of one of the lines of large dots arranged in the shape of the letter L that has more dots. Touch up the farthest possible dot from the origin.



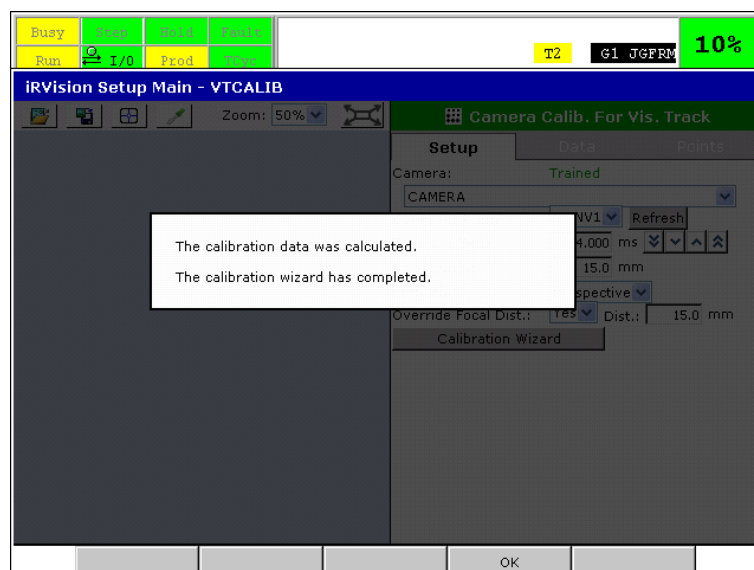
- 9 Press F4 [OK] key with the dot touched up. A screen as shown below is displayed.



- 10 Jog the robot upward (or away from the conveyor surface) . Next touch up a Y-axis direction point with the TCP. The Y-axis direction is the direction of one of the lines of large dots arranged in the shape of the letter L that has fewer dots. Touch up the farthest possible dot from the origin.



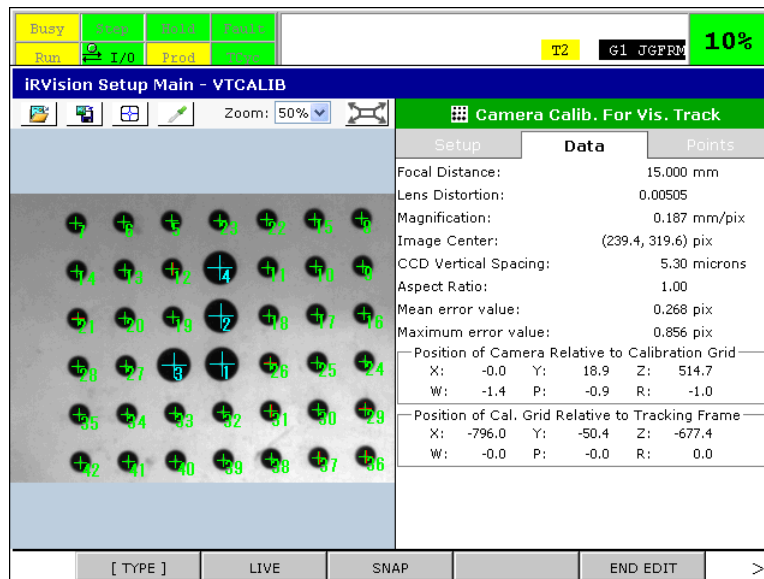
- 11 Press F4 [OK] key with the dot touched up. A screen as shown below is displayed.



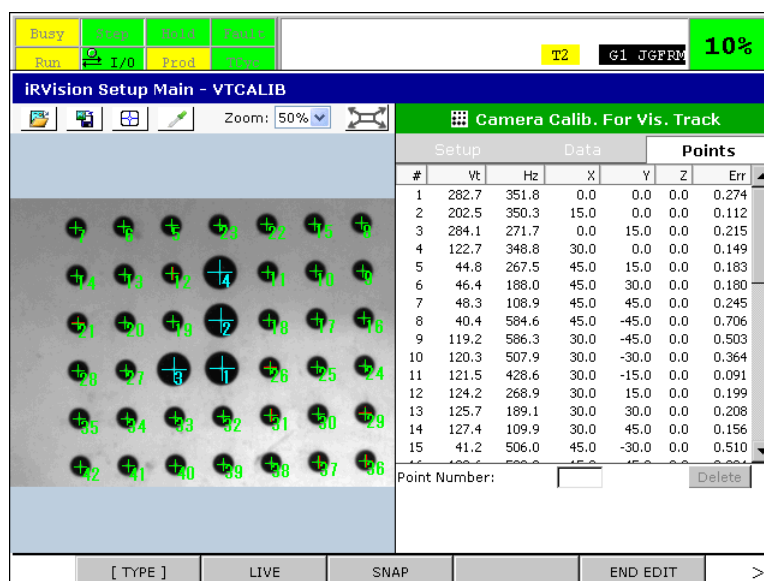
- 12 When this screen is displayed, the calibration is complete. Press F4 [OK] key to finish the procedure.

Checking the calibration result

- 1 Click the [Data] tab. An image as shown below is displayed.



- 2 Check whether the position of each + mark displayed on the image matches that of the corresponding dot of the grid pattern. If there is any incorrectly detected point, delete it as instructed below.
- 3 Select the [Points] tab. An image as shown below is displayed.



- 4 In the box to the right of [Point Number], enter the number of the point you want to delete, and then click [Delete]. The specified dot disappears from the image, and the calibration data is calculated again.
- 5 Press F10 [SAVE] key to save the data.
- 6 Press F5 [END EDIT] key to close the edit screen.

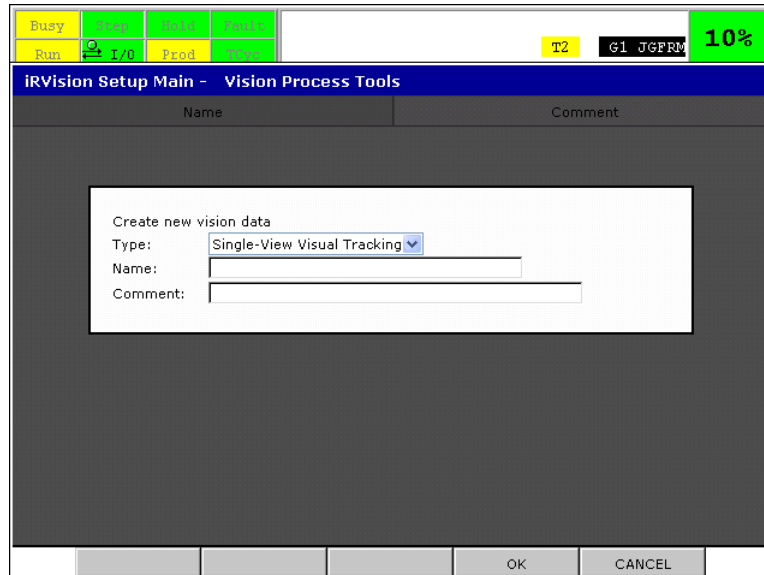
6.7.4.3 Vision process

Teach the vision process, and execute a test.

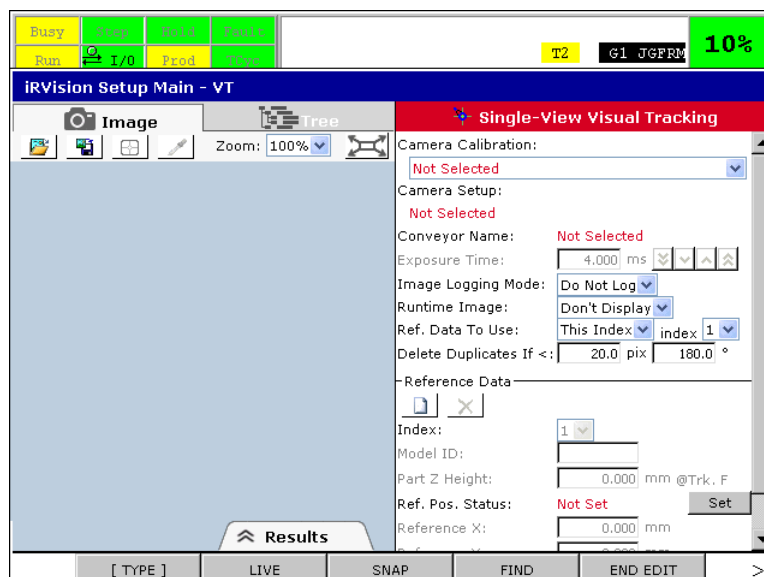
Here, only the GPM locator tool is mentioned as the tool used for detection. Other locator tools such as the blob locator tool may be used as well. For details, refer the chapter, "COMMAND TOOLS" in the "iRvision OPERATOR'S MANUAL (Reference)".

6.7.4.4 Teaching a vision process

- 1 From the [MENU] key, select [8 iRVision] and then [1 Vision setup] to display the vision data list screen. Press F4 [VTYPE] key, and select [Vision Process Tools] from the list.
- 2 Press F2 [CREATE] key. A screen as shown below is displayed.



- 3 In [Type], select [Single-View Visual Tracking].
- 4 In [Name], enter the name of the vision process (e.g., VT).
The name may consist of up to 34 alphanumeric characters and must not begin with a numeric character. Do not include a hyphen (-).
- 5 Press F4 [OK] key. The created vision process is displayed in the list.
- 6 Select the created vision process, and press F3 [EDIT] key. A screen as shown below is displayed.



- 7 In [Camera Calibration], select the desired calibration data.
- 8 In [Exposure Time], set the exposure time.
- 9 In [Image Logging Mode], select the image logging mode.

NOTE

Normally, images are saved in order to check undetected images later. Since saving images takes a considerable amount of time, however, it is common to select [Don't Log] for visual tracking so as to reduce the processing time. During the initial adjustment stage or in other cases where an impact on (or delay of) the cycle time can be tolerated, you may save images by selecting [Log Failed Images] or [Log All Images] and use them for such purposes as to adjust the detection parameters. Images are saved to a memory card.

- 10 In [Runtime Image], select the display mode.

NOTE

Normally, select [Display With 100%]. Selecting [Display With 50%] or [Don't Display] slightly reduces the processing time.

- 11 In [Ref. Data To Use], select the data to use.

**CAUTION**

In [Part Z Height], enter the height of the detected part of the part as seen from the tracking frame, not the height from the surface of the conveyor. For example, if a thick fixture is used to set up the tracking frame and the Z-axis direction of the tracking frame is upward relative to the conveyor surface, the value to be set is equal to "the height of the detected part of the part minus the thickness of the fixture". If this value is incorrect, the vision offset cannot be calculated correctly.

NOTE

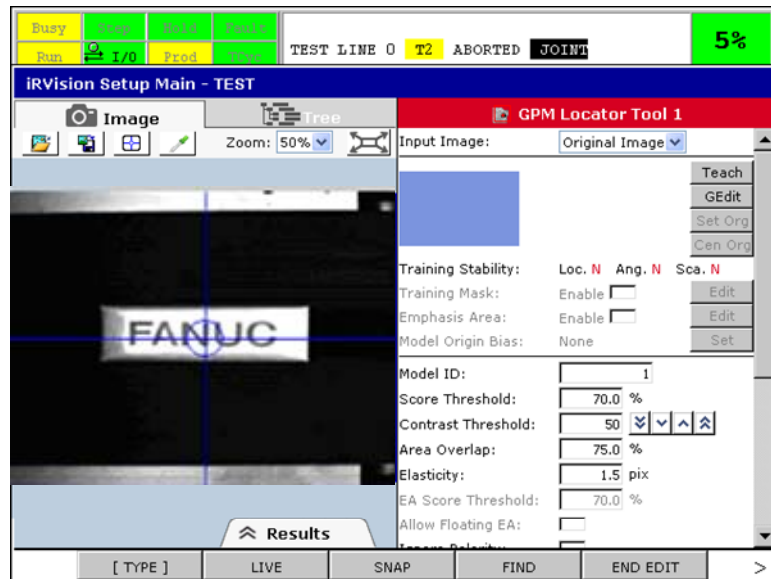
- 1 Normally, select [This Index] and, in [ID], specify the ID number of the reference data to be used at all times. If the reference data needs to be changed depending on the part detected in such cases as when multiple types of parts with different heights are conveyed, select [Model ID] and set different data in [Reference Data] for each model ID. In [Reference Data], enter an appropriate value in [Part Z Height] for each model ID.
- 2 You can switch [Part Z Height] based on a part z height measured by 3rd party sensor. In this case, you select [Use Register Value] in [Part Z Height Mode] and specify a register where the measurement value is stored in [Register Number].

6.7.4.5 Teaching a GPM locator tool

- 1 Select the tree view on the left side of the edit screen. Click [GPM Locator Tool 1], and teach a GPM locator tool. If you want multiple types of parts to be detected, you can create multiple GPM locator tools by clicking the  button with [Single-View Visual Tracking] selected.

NOTE

If parts and trays need to be detected with a single camera, it is necessary to create separate vision processes for parts and trays, respectively.



- 2 Select the image view on the left side of the edit screen, and place the part you want to teach in the center of the camera's field of view. Then, press F3 [SNAP] key to snap an image.
- 3 Click the [Teach] button. Enclose the part with the resulting magenta rectangular window, and press F4 [OK] key.
- 4 Enter appropriate values in the other fields as necessary. For details, refer to the section, "GPM LOCATOR TOOL" in the chapter, "COMMAND TOOLS" in the "iRVision OPERATOR'S MANUAL (Reference)".
- 5 Press F10 [SAVE] key to save the data.
- 6 F5 [END EDIT] key to close the edit screen.

NOTE

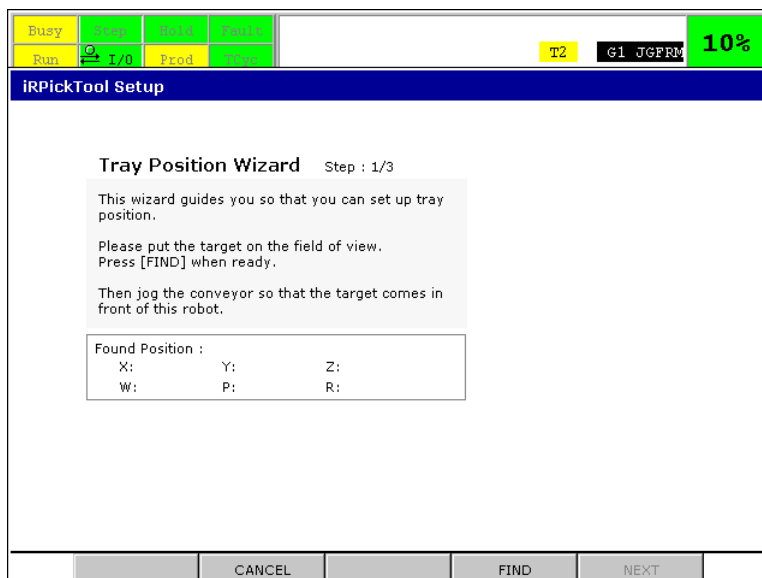
You do not need to set the reference position by clicking the [Set] button in [Reference Data] for the vision process. The reference position will be set in the sensor setup screen described later.

6.7.5 Setting the Tray Position

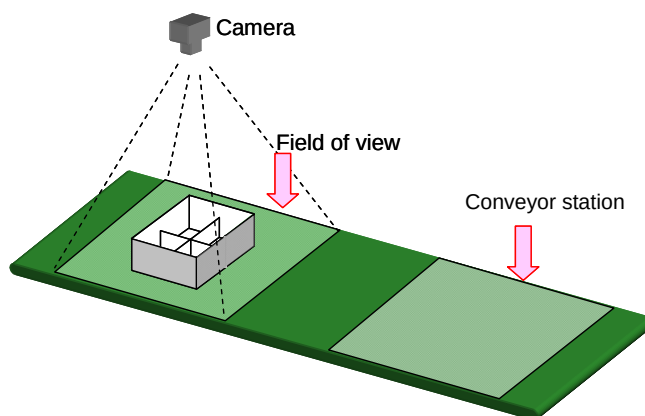
If [Find part by vision] is selected for [Trigger Action] and a tray is used, set the relative position of the origin and detection position of the tray. To do this requires that camera calibration and vision process teaching be completed in advance.

6.7.5.1 When [Distance] and [Find part by vision] are selected

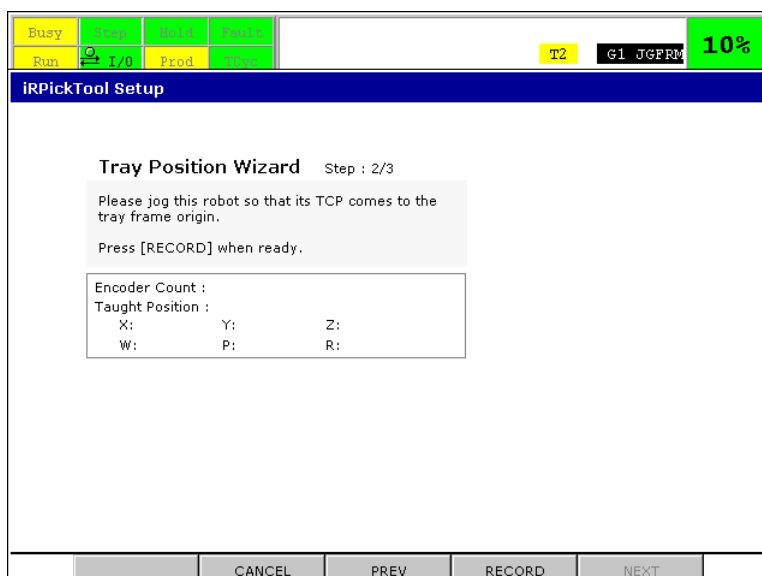
- 1 Stop the conveyor.
- 2 Press F4 [SET TRY POS] key. A screen as shown below is displayed.



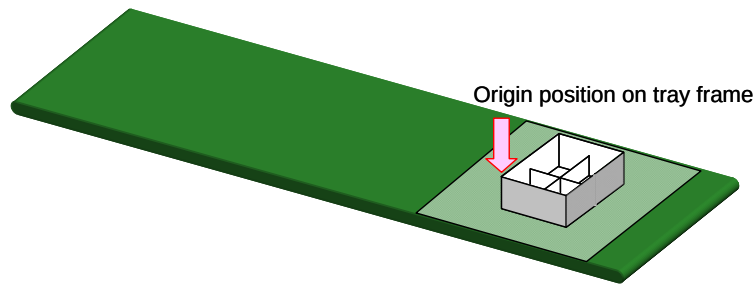
- 3 Place a tray in the camera's field of view, and press F4 [FIND] key.



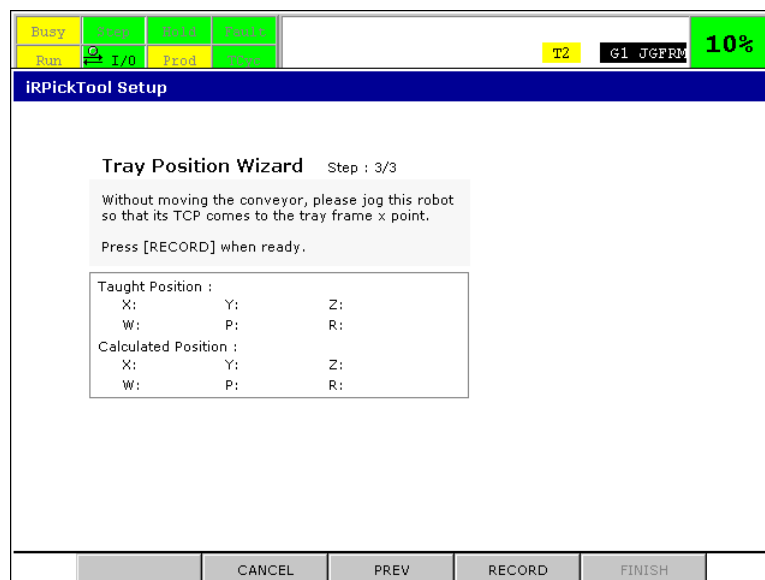
- 4 When the tray is successfully detected, the detection position is displayed and F5 [NEXT] key is enabled.
- 5 If you press F5 [NEXT] key, a screen as shown below is displayed.



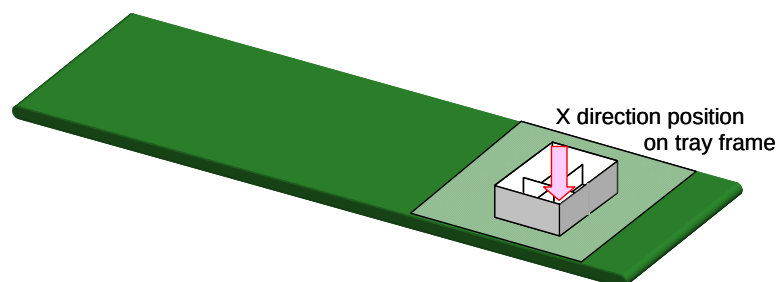
- 6 Move the conveyor until the tray comes in front of the robot.
- 7 Jog the robot, and touch up the origin of the tray frame.



- 8 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.
- 9 If you press F5 [NEXT] key, a screen as shown below is displayed.



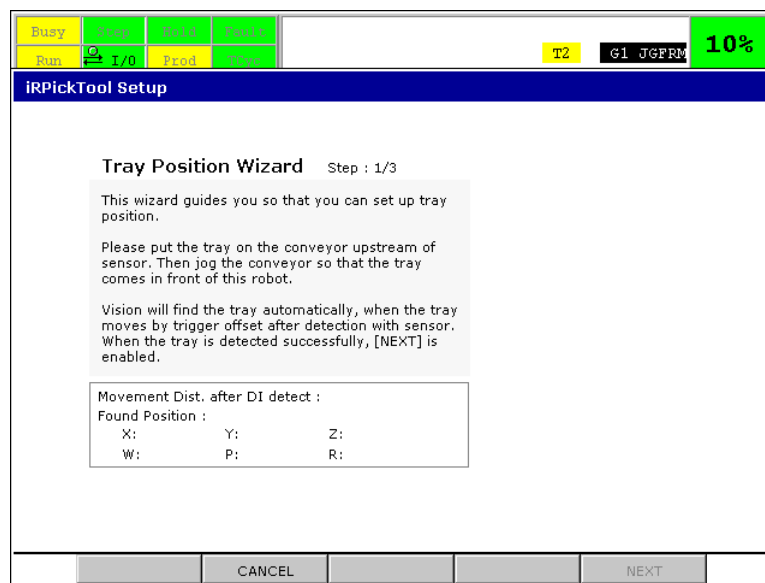
- 10 Jog the robot, and touch up the X direction point of the tray frame.



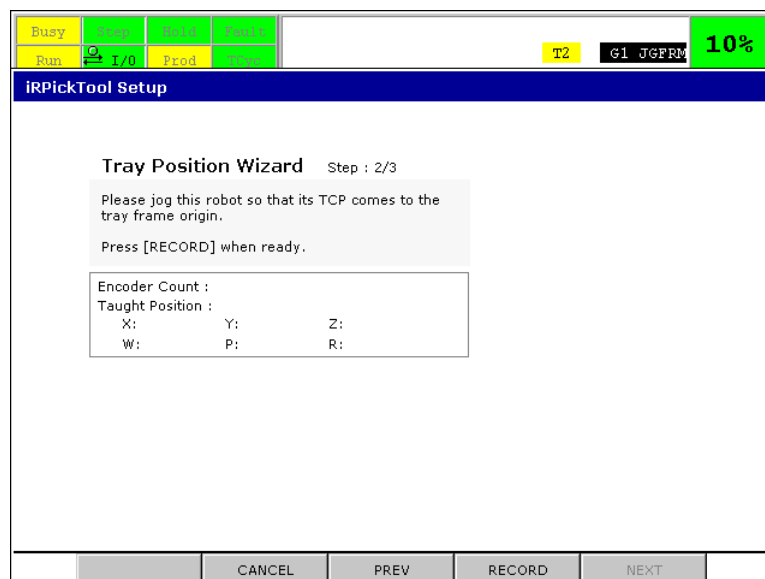
- 11 Press F4 [RECORD] key. The touch-up position and the calculated position are displayed, and F5 [FINISH] key is enabled.
- 12 If you press F5 [FINISH] key, the initial setup screen is displayed again. Values are input for X, Y, and R of the tray position.

6.7.5.2 When [DI] and [Find part by vision] are selected

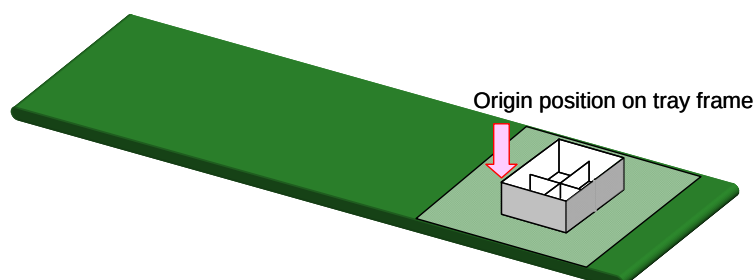
- 1 Stop the conveyor.
- 2 Press F4 [SET TRY POS] key. A screen as shown below is displayed.



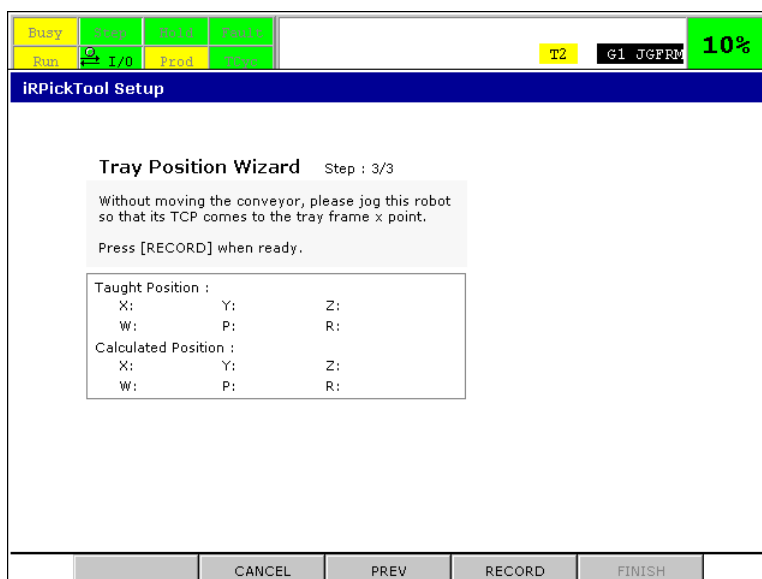
- 3 Let the tray travel from the most upstream end of the conveyor. When the conveyor moves by the trigger offset after the photo eye spots the part, *iR*Vision automatically detects the part. You do not need to stop the conveyor when *iR*Vision detects the part. Stop the conveyor after it reaches this robot operation range. When *iR*Vision successfully detects the part, the detection position are displayed and F5 [NEXT] key is enabled.
- 4 If you press F5 [NEXT] key, a screen as shown below is displayed.



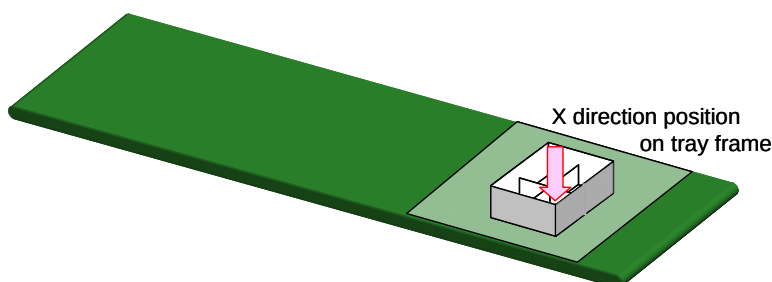
- 5 Jog the robot, and touch up the origin of the tray frame.



- 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 If you 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.



- 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.

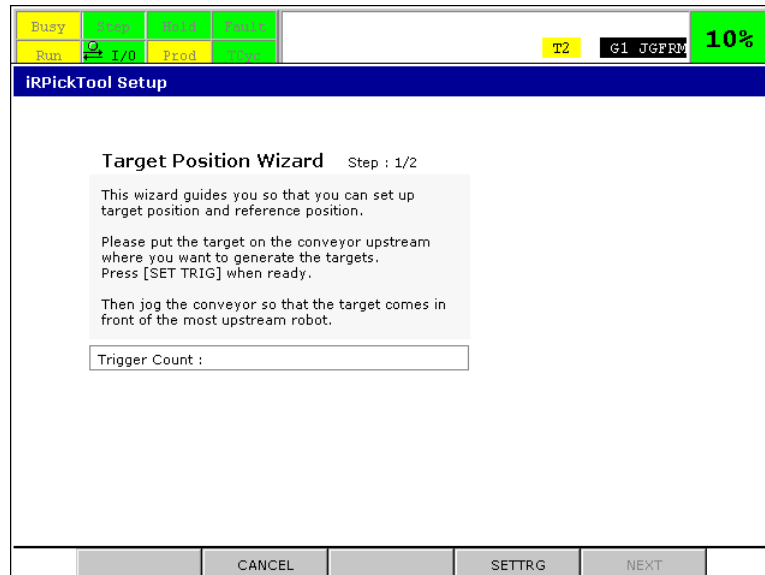
6.7.6 Setting the Target Position

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.

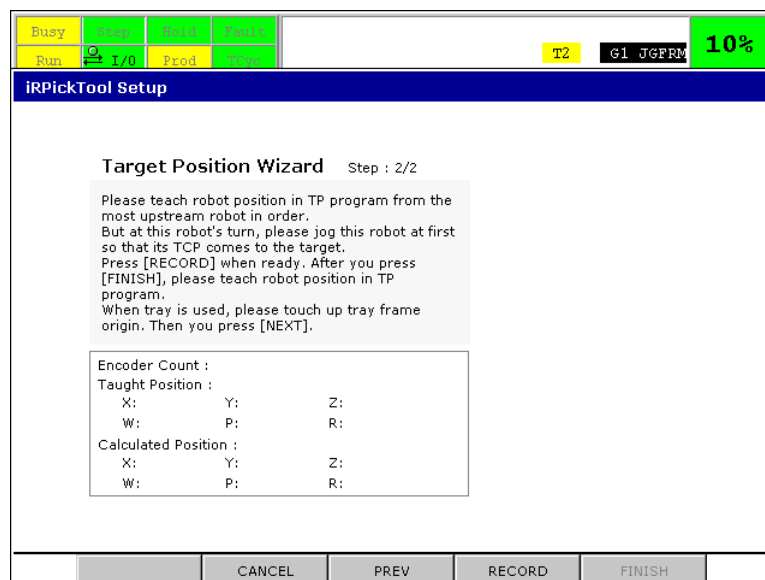
This position is put into the queue each time the conveyor moves a certain distance or the [Index Advance Trigger DI] of the servo conveyor is input. Since a part is not actually detected when this combination is selected, perform robot position teaching at the same time you set the target position. The procedure slightly differs depending on whether a tray is used on the conveyor or not.

6.7.6.1 When a tray is not used

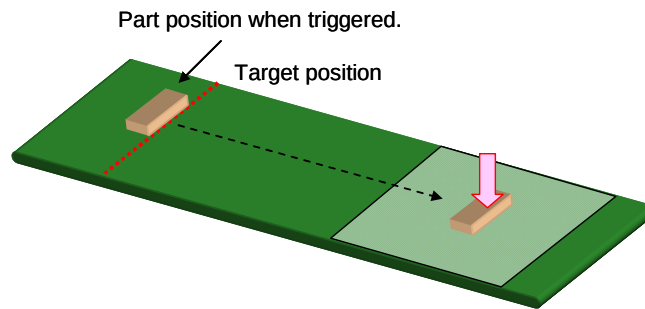
- 1 Stop the conveyor.
- 2 Press F4 [SET TRG POS] key. A screen as shown below is displayed.



- 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 6.12, "TEACHING THE POSITION TO ROBOTS".) Note that, when the part comes in front of this robot, first touch up the downstream-side end of the part with the TCP and then press F4 [RECORD] key.

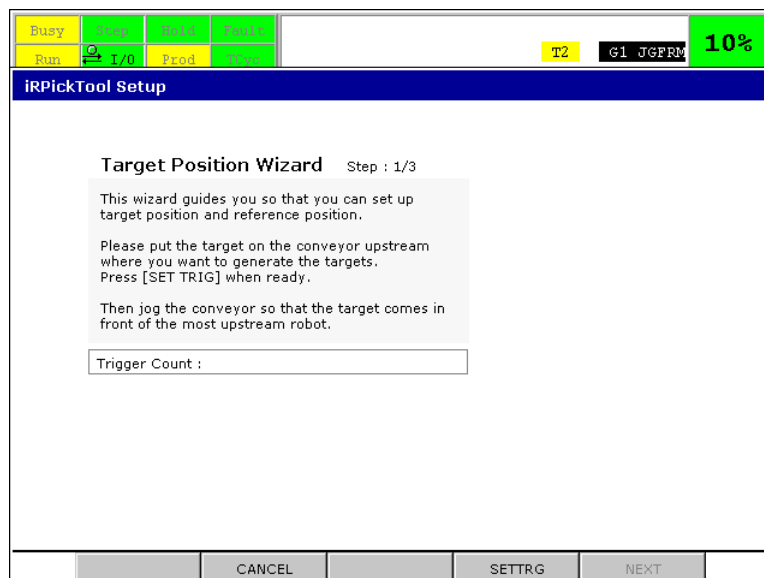


The encoder value obtained at the time of touch-up, the touch-up position, and the calculated position are displayed, and F5 [FINISH] key is enabled.

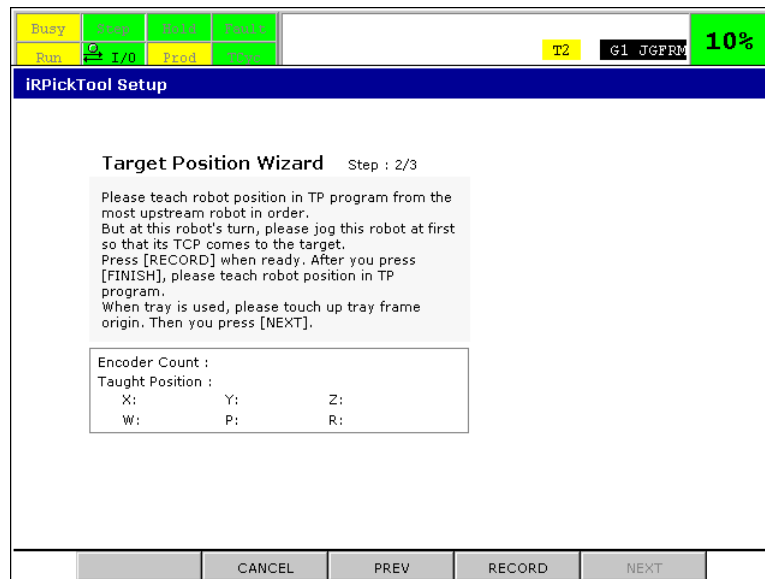
- 7 If you press F5 [FINISH] key, the initial setup screen is displayed again. The calculated value is input for X of the target position (the values of Y and R remain unchanged from 0).
- 8 Teach the position to this robot. Then, move the conveyor further to teach the position to the downstream robots.

6.7.6.2 When a tray is used

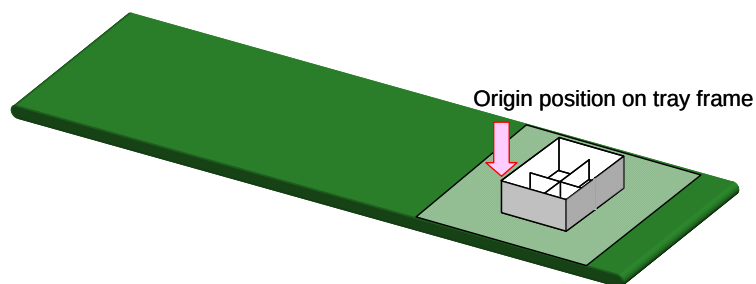
- 1 Stop the conveyor.
- 2 Press F4 [SET TRG POS] key. A screen as shown below is displayed.



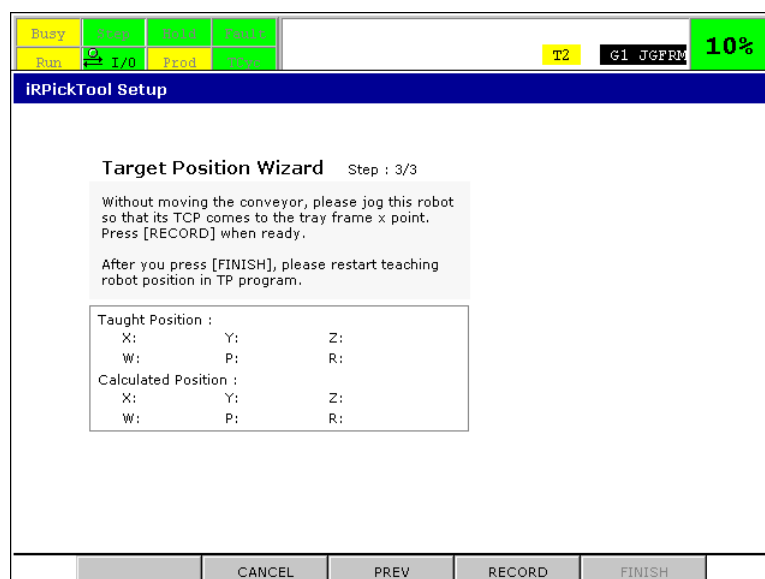
- 3 Place a part at the position where you want to generate it. In order to perform robot position teaching for all robots, the part needs to be placed upstream of the most upstream robot operation range.
- 4 Press F4 [SETTRG] key. The current encoder value is displayed, and F5 [NEXT] key is enabled.
- 5 If you press F5 [NEXT] key, a screen as shown below is displayed.



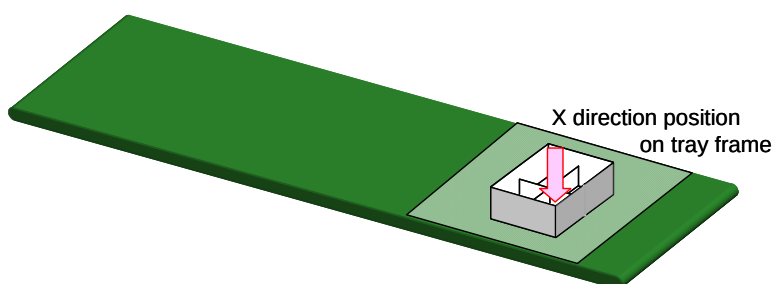
- 6 Move the conveyor, and perform robot position teaching sequentially beginning with the most upstream robot. (See Section 6.12, "TEACHING THE POSITION TO ROBOTS".)
Note that, when the part comes in front of this robot, first jog the robot and touch up the origin of the tray frame. Then, press F4 [RECORD] key.



- 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.



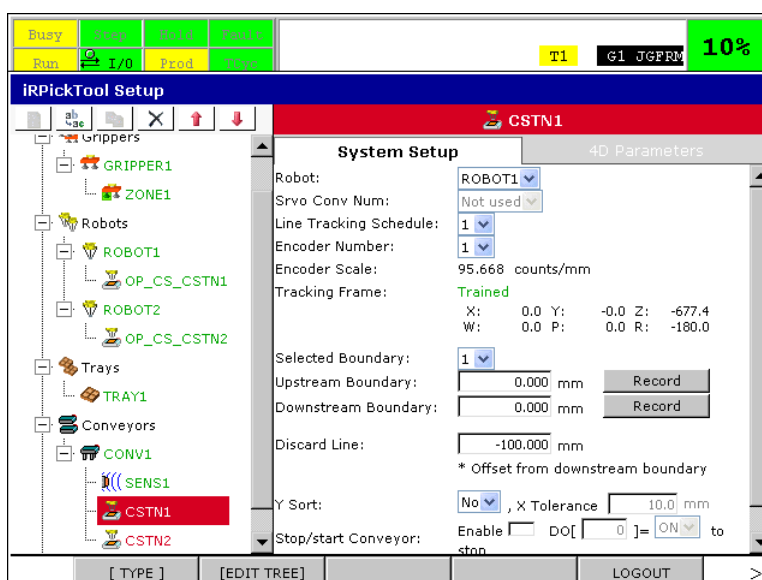
- 9 Jog the robot, and touch up the X direction point of the tray frame. Press F4 [RECORD] key. The touch-up position and the calculated position are displayed, and F5 [FINISH] key is enabled.



- 10 If you press F5 [FINISH] key, the initial setup screen is displayed again. Values are input for X, Y, and R of the target position.
- 11 Teach the position to this robot. Then, move the conveyor further to teach the position to the downstream robots.

6.8 SETUP OF A CONVEYOR STATION

Set up each conveyor station.



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 iRPickTool menu. For details, see Appendix D, "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.

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.

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.

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.

After setting [Upstream Boundary] and [Downstream Boundary], change the tool frame back to the one that is set in Section 3.4, "SETTING OF THE TCP".

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:

$$0.8 \text{ (s)} \times 100 \text{ (mm/s)} = 80 \text{ (mm)}$$

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:

$$0.8 \text{ (s)} \times 5 \text{ (deg/s)} = 4 \text{ (deg)}$$

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.

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**

If you set a tray in a conveyor, Y sort cannot work.

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.

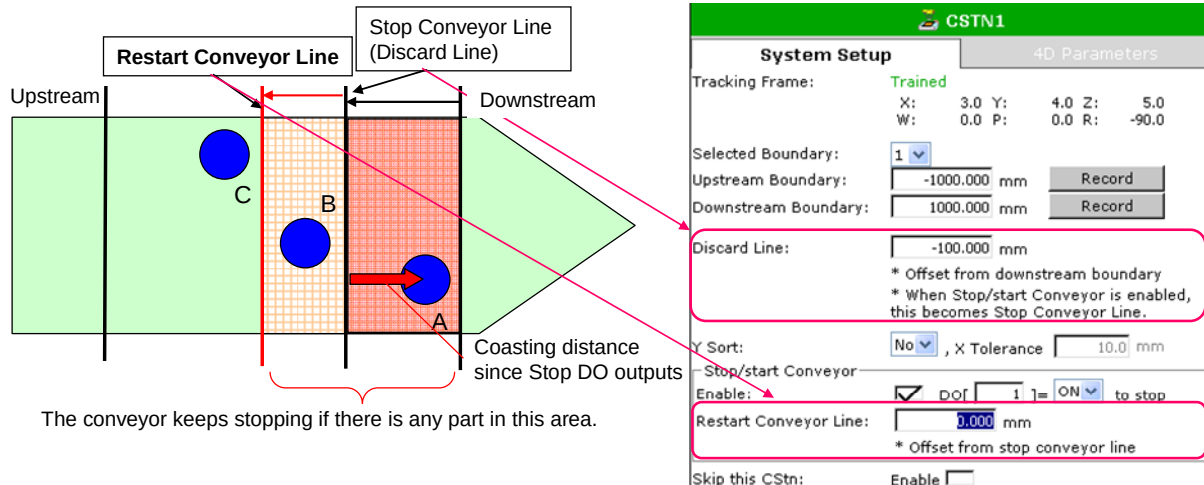
When using this function, specify the number of the DO signal used to stop and restart the conveyor, as well as the polarity indicating whether to turn the signal on or off to stop the conveyor.

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.

See the figure below. The conveyor stopped because the part A passes the stop conveyor 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.



⚠ 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

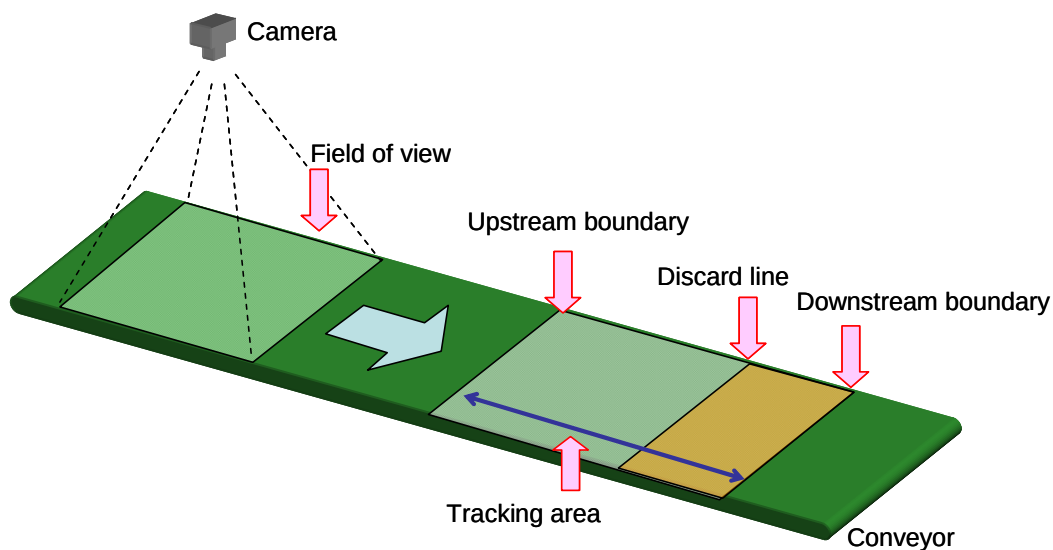
Depending on the setup of other items, the setting items of the pre-grouping function may be shown as below.

Pre-group	
Generation Line	0.000 mm
Send Line	0.000 mm
	* Offset from discard line
Grouping Distance	Auto, mm

For details, see Section 6.16, "SETUP OF PRE-GROUPING".

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.



6.8.1 Adding a Tracking Frame

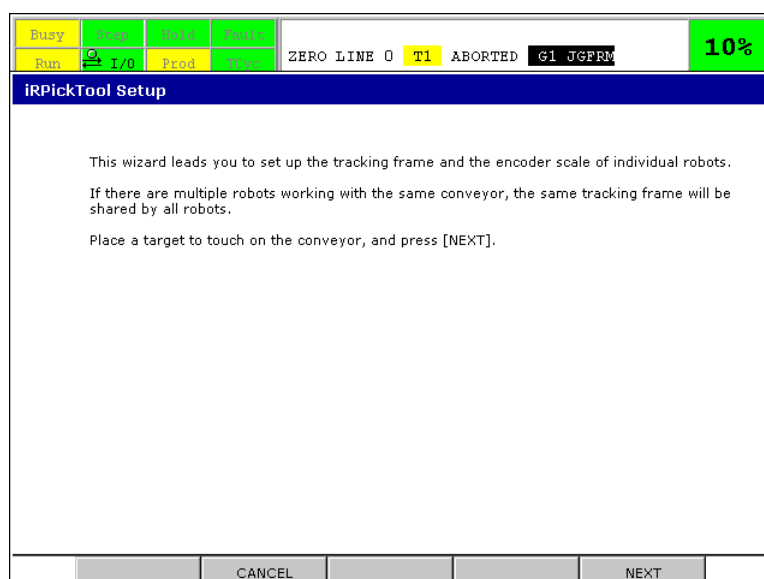
After you complete the setting of a tracking frame for a conveyor on the conveyor setup page, you can redo the tracking frame setting for a specific conveyor station. Use this procedure in cases such as when you have selected the wrong user tool frame for a robot during the tracking frame setting or 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.

6.8.1.1 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”.

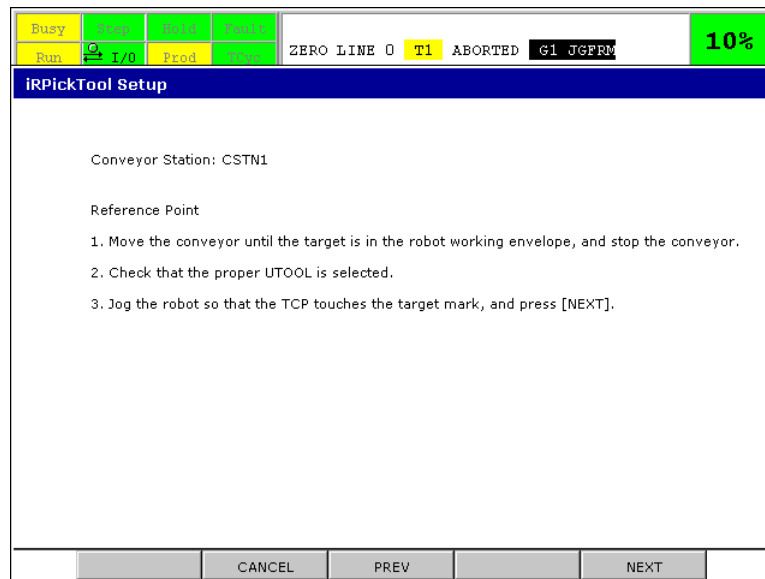
- 1 In the conveyor station setup screen, press F4 [ADD TRK FRM] key. A screen as shown below is displayed.



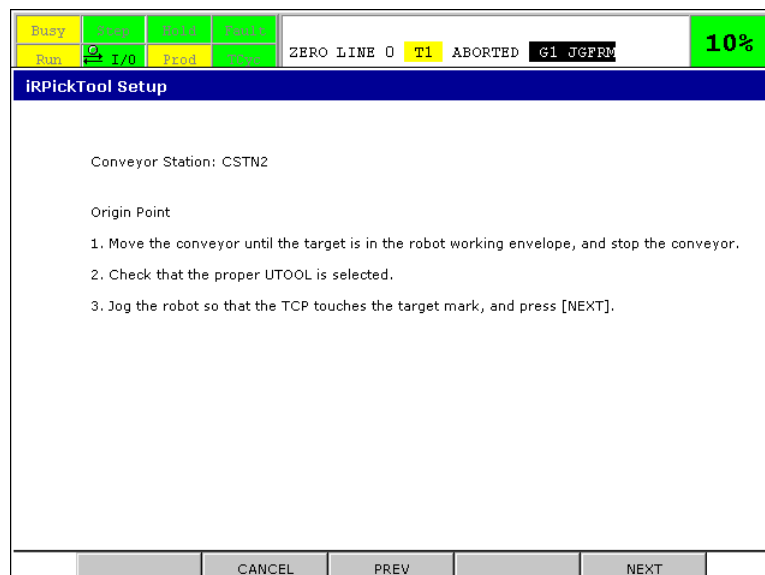
NOTE

- 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.

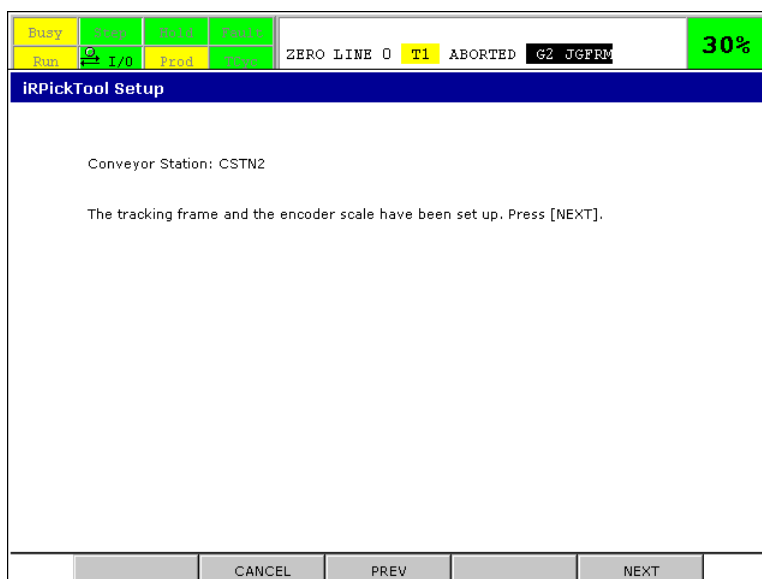
- 2 Place a fixture that can be touched up with the TCP of the robot at the most upstream area of the conveyor, and press F5 [NEXT] key. A screen as shown below is displayed.



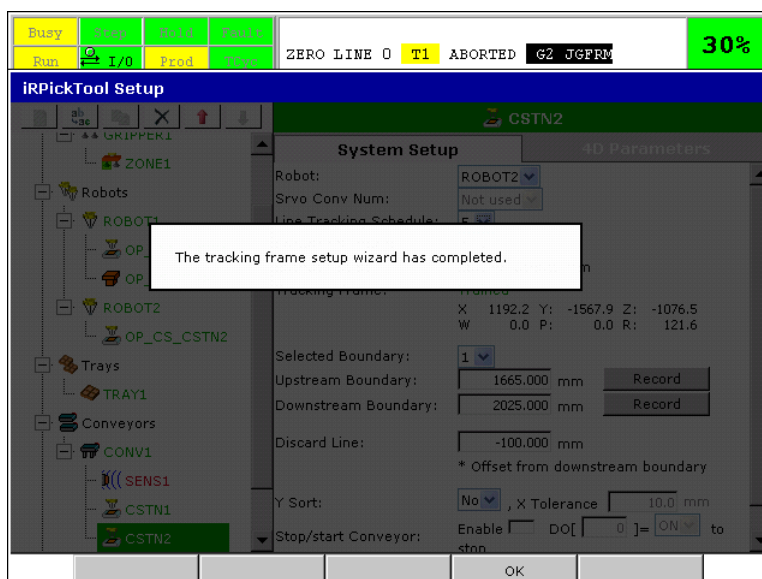
- 3 For the most upstream robot, select the user tool frame set in Section 3.4, "SETTING OF THE TCP" as the jog feed frame.
- 4 Move the conveyor until the fixture comes into the most upstream robot operation range, and stop it. Jog the robot to touch up the fixture with the TCP. With the fixture touched 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.



- 5 For the conveyor station for which you are redoing the tracking frame setting, touch up the fixture as instructed in steps 4 through 7 in Subsection 6.6.1, "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.



- 6 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. If you press F5 [NEXT] key, a screen as shown below is displayed.

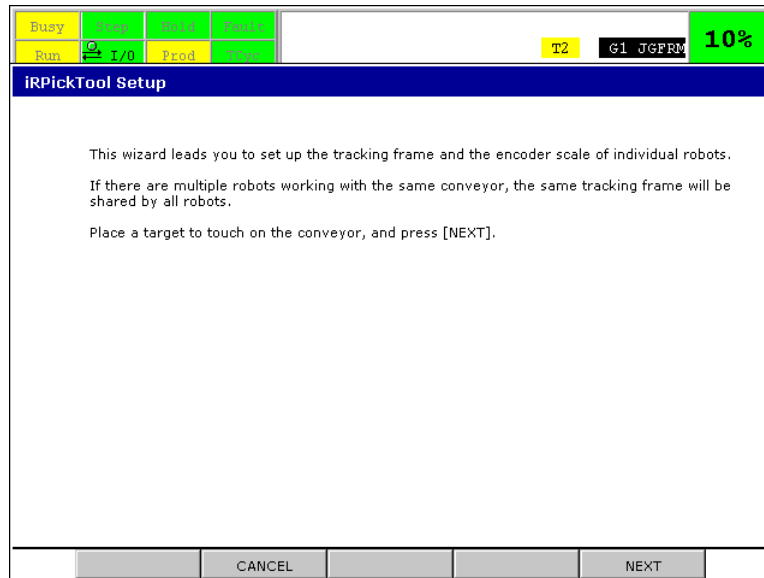


- 7 Press F4 [OK] key to finish the tracking frame addition procedure.

6.8.1.2 Case of circular conveyor

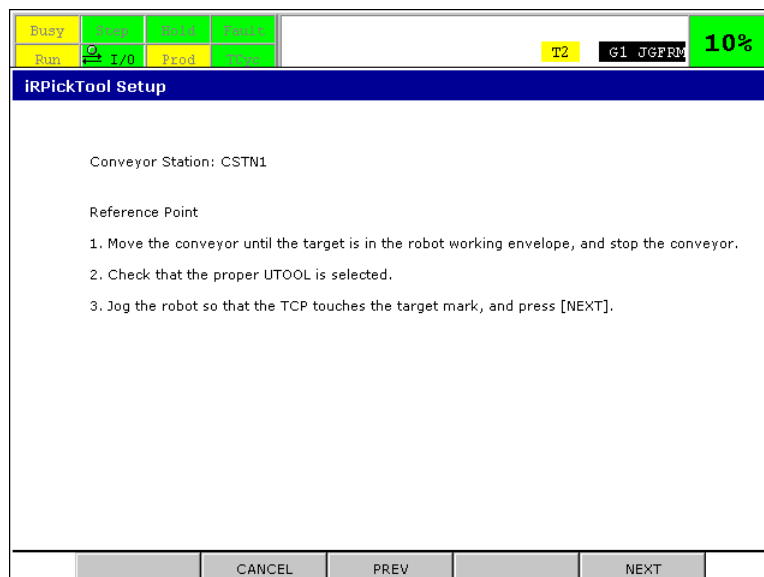
When the tracking type is “Circle”, see this procedure.

- 1 In the conveyor station setup screen, press F4 [ADD TRK FRM] key. A screen as shown below is displayed.

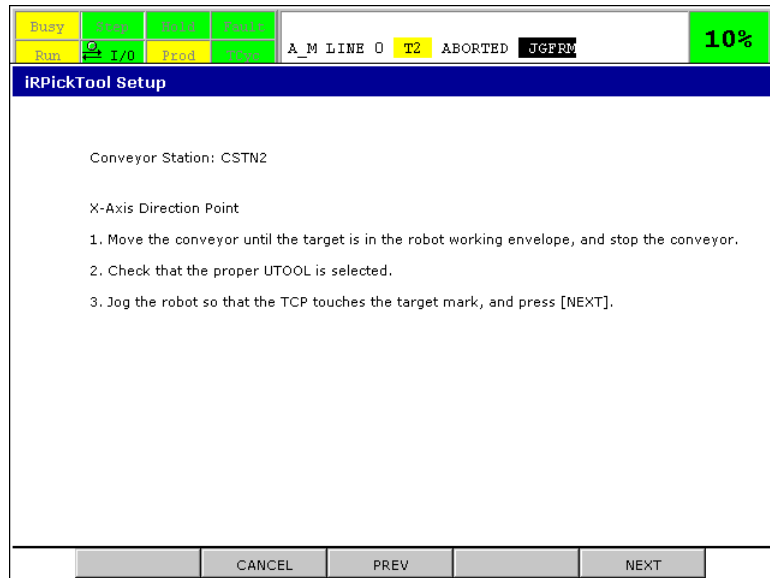


NOTE

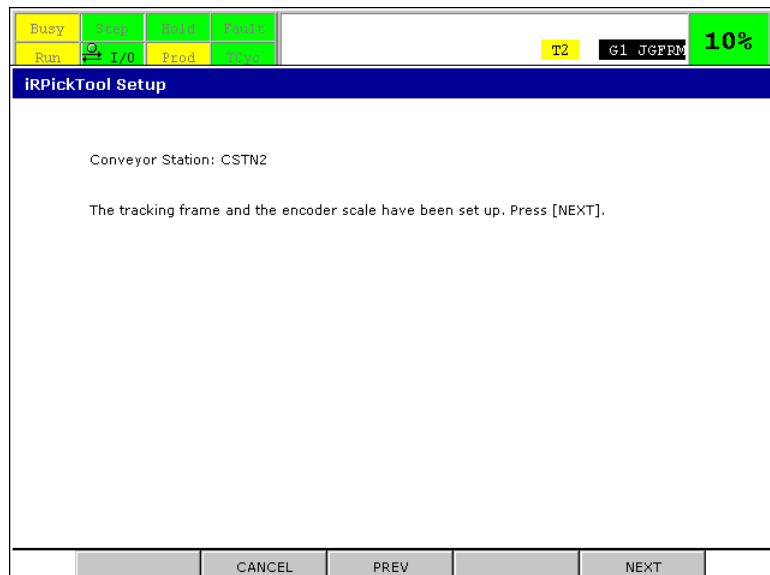
- 1 F4 [ADD TRK FRM] key is not displayed on the setup page for the most upstream conveyor station.
 - 2 F4 [ADD TRK FRM] key is disabled if a tracking frame is not set for the most upstream conveyor station. In that case, set up a tracking frame for all conveyor stations on the conveyor setup page.
- 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. A screen as shown below is displayed.



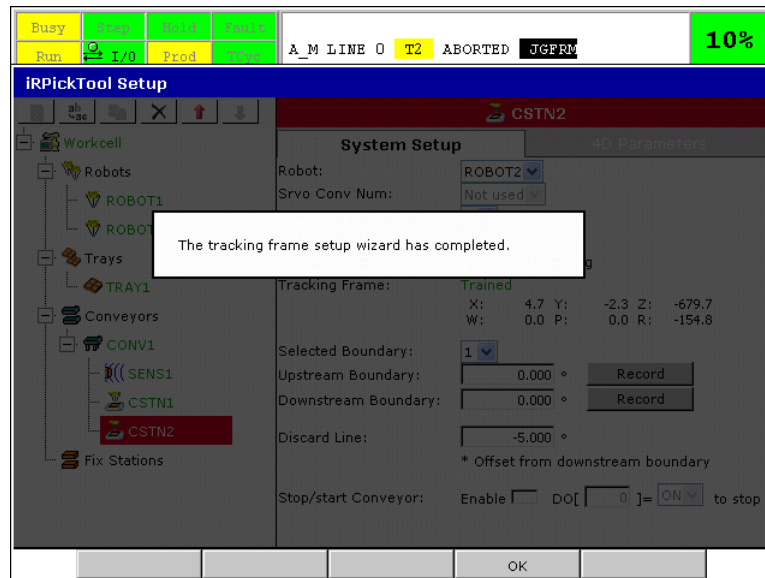
- 3 For the most upstream robot, select the user tool frame set in Section 3.4, "SETTING OF THE TCP" as the jog feed frame.
- 4 Move the conveyor until the fixture comes into the most upstream robot operation range, and stop it. Jog the robot to touch up the fixture with the TCP.
With the fixture touched 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.



- 5 For the conveyor station for which you are redoing the tracking frame setting, touch up the fixture as instructed in steps 4 through 7 in Subsection 6.6.1, "Setting a Tracking Frame". When you have proceeded to the step of setting the assistant point, a screen as shown below is displayed.



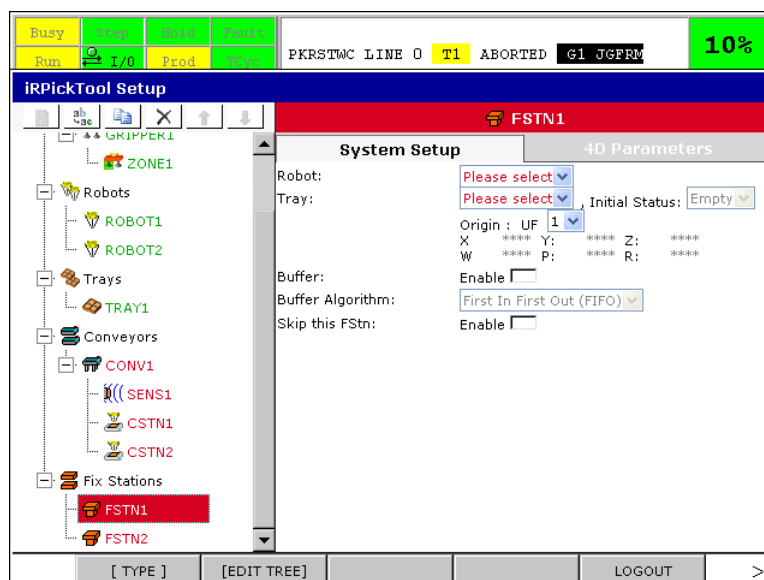
- 6 This completes the tracking frame addition procedure, and the same encoder scale as the most upstream robot and the newly calculated tracking frame are set. If you press F5 [NEXT] key, a screen as shown below is displayed.



- 7 Press F4 [OK] key to finish the tracking frame addition procedure.

6.9 SETUP OF A FIXED STATION

Set up a fixed station.



Robot

Select the robot that operates with this fixed station.

Tray

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 [6 SETUP] and then [Frames] from the [MENU] key, instead of the iRPickTool 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 "R-30iB /R-30iB Mate CONTROLLER OPERATOR'S MANUAL (Basic Operation)".



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.

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.

6.10 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 6.1, "SETUP OF A WORK CELL" to Section 6.9, "SETUP OF A FIXED STATION".

To teach the actual position to the robot, perform the procedures described in Section 6.11, "REFERENCE POSITION SETTING" and Section 6.12, "TEACHING THE POSITION TO ROBOTS".



CAUTION

By default, *i*RPickTool has hot start disabled.

Your system requirements may make it necessary to modify the robot programs. Subsection 6.10.1, "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 6.10.2, "Registers" and later. Do not use these registers and settings.

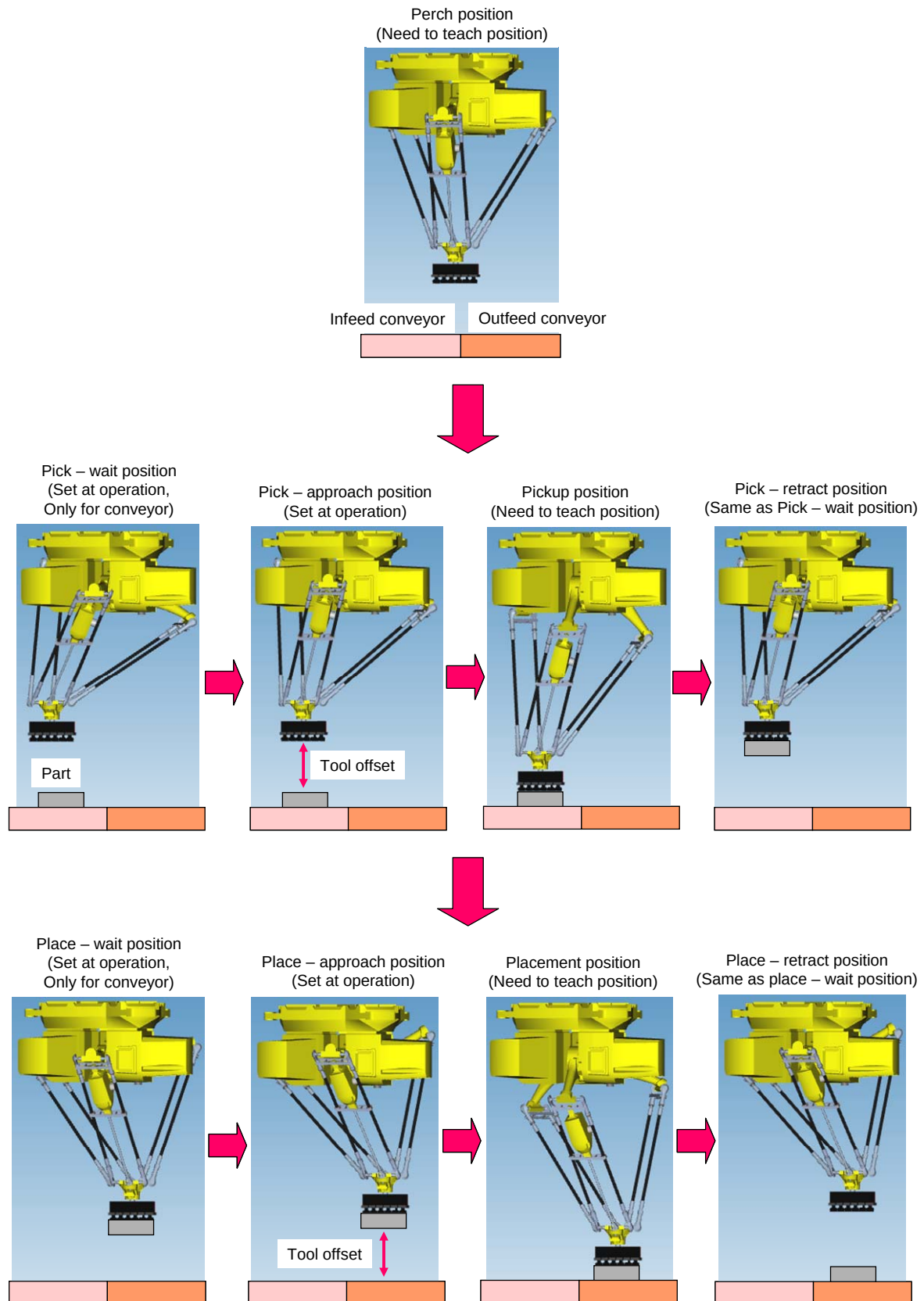


CAUTION

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.

6.10.1 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.



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 6.14, "KAREL PROGRAMS" and Section 6.15, "STANDARD TP PROGRAMS".

6.10.2 Numeric Registers

iRPickTool uses the following Numeric Registers in the TP programs.

Presently, Registers 95-230 are used.

If you need more registers, please increase the registers using the Program Setup function at Controlled Start.

Table 6.10.2 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[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 or not.)

6.10.3 Position Registers

iRPickTool uses the following Position Registers in the TP programs.

Presently, Position registers 51-90 are used.

If you need more Position registers, please increase the Position Registers using the Program Setup function at Controlled Start.

Table 6.10.3 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

6.10.4 String Registers

iRPickTool uses the following Position Registers in the TP programs.

Table 6.10.4 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

6.10.5 Vision Registers

iRPickTool uses the following Vision Registers in the TP programs.

Table 6.10.5 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.

6.10.6 Payload IDs

iRPickTool uses the following payload IDs in the TP programs.

Table 6.10.6 Payload IDs

PAYLOAD[1]	Empty Gripper
PAYLOAD[2]	Full Gripper

6.10.7 User Alarms

iRPickTool uses the following user alarms in the TP programs.

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]

6.10.8 User Frames

The TP programs use the user frames that you reference in your Fixed Station property page in the iRPickTool treeview. It does not use any dedicated user frames for internal use.

6.10.9 Tool Frames

The number of Tool Frames used by iRPickTool 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 iRPickTool TP programs will use Tool Frames 1 through 10.

If you have 20 zones (maximum number of zones) in the gripper, then iRPickTool 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.

6.11 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 wizard 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.

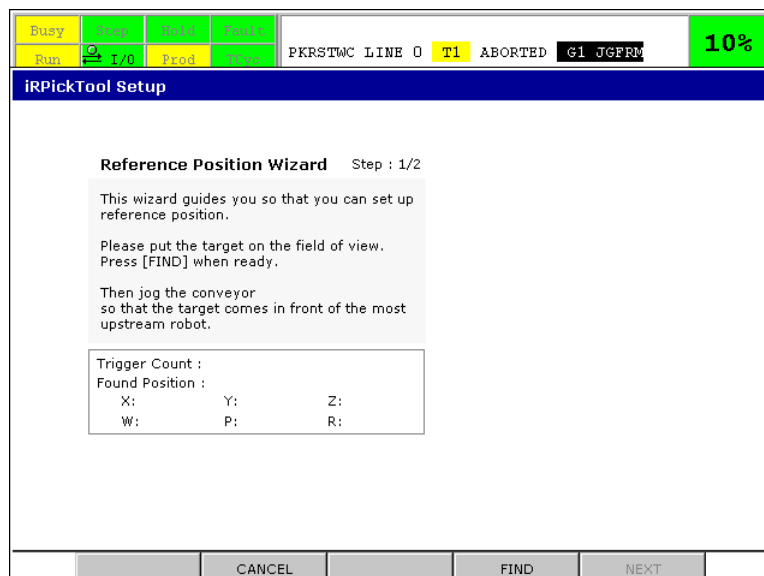
NOTE

- 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.

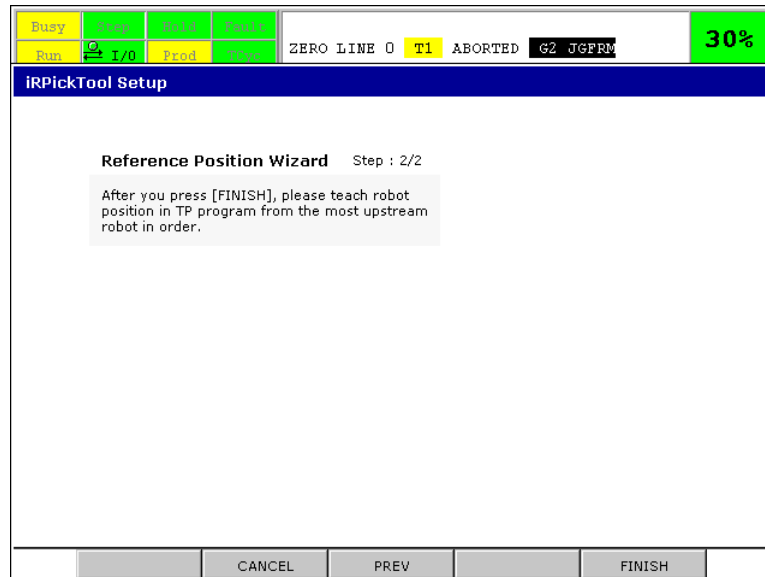
The procedure slightly differs depending on the combination of a trigger condition and an action.

6.11.1 When [Distance] and [Find part by vision] are Selected

- 1 Stop the conveyor.
- 2 Press F3[SET REF POS] key A screen as shown below is displayed.



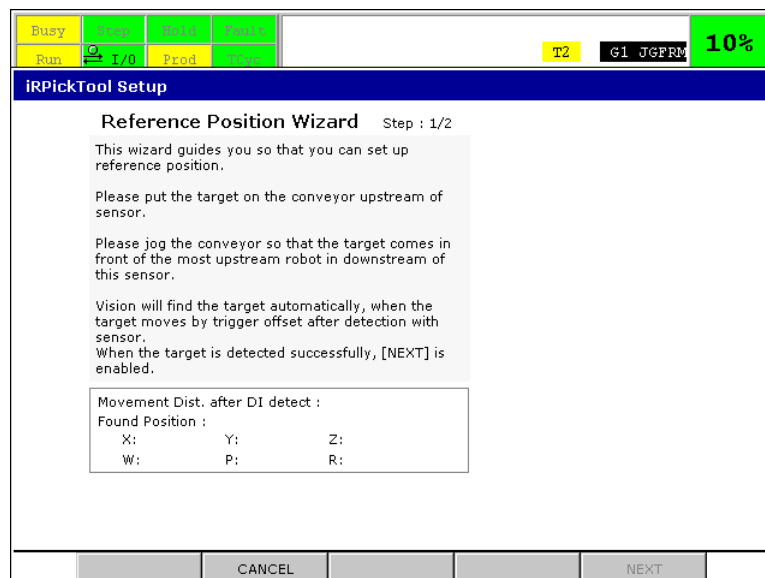
- 3 Place a part near the center of the camera's field of view.
- 4 Press F4 [FIND] key.
- 5 If iRVision detects the part, the encoder value obtained at the time of detection and the detection position are displayed and F5 [NEXT] key is enabled.
- 6 If you press F5 [NEXT] key, a screen as shown below is displayed.



- 7 Press F5 [FINISH] key. This completes the reference position setting procedure.
- 8 Next, perform robot position teaching. (See Section 6.12, "TEACHING THE POSITION TO ROBOTS".)

6.11.2 When [DI] and [Find part by vision] are Selected

- 1 Stop the conveyor.
- 2 Press F3[SET REF POS] key A screen as shown below is displayed.



- 3 Let the part travel from the most upstream area of the conveyor. When the conveyor moves by the trigger offset after the photo eye spots the part, iRVision automatically detects the part. The conveyor does not need to be stopped for iRVision to detect the part. Move the conveyor until the most upstream robot operation range is reached and stop it. When iRVision successfully detects the part, the detection position are displayed and F5 [NEXT] key is enabled.

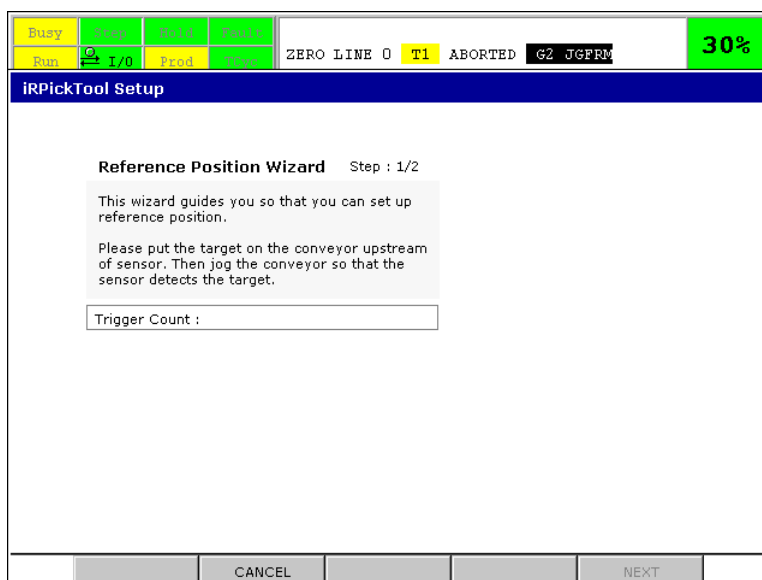
- 4 If you press F5 [NEXT] key, a screen as shown below is displayed.



- 5 Press F5 [FINISH] key. This completes the reference position setting procedure.
 6 Next, perform robot position teaching. (See Section 6.12, "TEACHING THE POSITION TO ROBOTS".)

6.11.3 When [DI] or [HDI] and [Put part in queue] are Selected

- 1 Stop the conveyor.
 2 Press F3[SET REF POS] key A screen as shown below is displayed.



- 3 Place a part upstream of the photo eye. Move the conveyor and let the part pass in front of the photo eye. The conveyor does not need to be stopped for the photo eye to detect the part. Move the conveyor until the most upstream robot operation range is reached and stop it. When the photo eye successfully detects the part, the encoder value obtained at the time of detection is displayed and F5 [NEXT] key is enabled.

- 4 If you press F5 [NEXT] key, a screen as shown below is displayed.



- 5 Press F5 [FINISH] key. This completes the reference position setting procedure.
 6 Next, perform robot position teaching. (See Section 6.12, "TEACHING THE POSITION TO ROBOTS".)

6.11.4 When [Distance] or [Flag] and [Put part in queue] are Selected

Since a part is not actually detected for this combination, set the reference position at the same time you set the target position. For details, see Subsection 6.7.6, "Setting the Target Position".

6.12 TEACHING THE POSITION TO ROBOTS

After setting the reference position, teach the position to robots.

6.12.1 Conveyor Station

- 1 When the part enters the tracking area after the reference position is set in the sensor wizard screen, stop the conveyor.
- 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 positions for cell 2 and subsequent cells are automatically calculated based on the tray pattern, and the vision offset instruction moves the robot to each cell based on the offset value stored in the 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 "R-30iB/R-30iB Mate CONTROLLER OPERATOR'S MANUAL (Basic Operation)".

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.

6.12.2 Fixed Station

Teach the placement (or pickup) position only for cell 1.

The position of every other cell is created relative to cell 1 by using the vision offset instruction.

NOTE

Cell 1 does not necessarily need to be taught for the user frame representing the tray frame.

6.13 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.

6.13.1 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, "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.

6.13.2 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 TCP. You can improve the situation, however, by adding an amount of adjustment to the vision offset calculated by *iRVision* 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 6.15.2.12 “PK_CV_PICK11”

```
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 OPTIONS" in the "iRVision 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 *iRPickTool*, ADJ_OFS is available as a standard feature even if the vision support tool option is not ordered.

6.13.3 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 6.13.1 and 6.13.2.
- 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 6.15.2.12 "PK_CV_PICK11"

```

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 Section 6.3, "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:

$$[\text{Dynamic Error Adjustment}] = +30 \text{ (ms)} + (3 \text{ (mm)} \div 500 \text{ (mm/s)}) \times 1000 = +36 \text{ (ms)}$$

6.14 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 iRPickTool 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.

6.14.1 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.

6.14.1.1 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 6.14.6.2, "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 6.14.5.6, "PKSNSTART.PC".

There is no argument.

Example:

```
CALL PKWCSTART
```

6.14.1.2 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
```

6.14.1.3 PKWCHECK.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 PKCCLRQUE and PKSNSTART are called separately.

There is no argument.

Example:

```
CALL PKWCHECK
```

6.14.1.4 PKWCRSTPRDST.PC

This program resets Production Status of *iRPickTool*. For Production Status, see the section 7.2 "CHECKING THE PRODUCTION STATUS" in detail.

There is no argument.

Example:

```
CALL PKWCRSTPRDST
```

6.14.1.5 PKRSTWC.PC

This program is used to reset the *iRPickTool* 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 *iRPickTool* 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 *iRPickTool* 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
```

6.14.1.6 PKABO.PC

This program is used to abort all the *iRPickTool* 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 use it during testing and debugging of your application.

- You need to call PKABO only in one controller to abort all the *iRPickTool* 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
```

6.14.1.7 PKLABO.PC

This program aborts all the running tasks in the controller in which it is executed.

There is no argument.

Example:

```
CALL PKLABO
```

6.14.1.8 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)
```

6.14.1.9 PKOPGETEN.PC

This program is used to check if the R857: iRPickTool / Option Package Add-on option is enabled. If it returns zero, then the option is not loaded.

Argument 1:

Specify the Numeric Register in which to return enabled (=1) or disabled (=0).

Example:

To check if R857 is enabled, use the following call to return the value in the Numeric Register 95.

```
CALL PKOPGETEN(EffToolEnb Reg=95)
```

6.14.1.10 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, you can get the value of \$pkcs[1].\$n2pick[2]

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, specify 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)
```

6.14.1.11 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)
```

6.14.1.12 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)
```

6.14.2 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.

6.14.2.1 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)
```

6.14.2.2 PKGRCHKQTY.PC

This program is used to check if the gripper has enough zones to handle the required “Quantity” of parts. It is a validation program.

Argument 1:

Specify an ID of a gripper.

Argument 2:

Specify the pick or place quantity. You can use the KAREL macros PKFSGETQTY or PKCSGETQTY to get the quantity of parts to pass to this program.

Argument 3:

Specify the Numeric Register in which to return the status. If the status is non-zero then the gripper does not have enough zones to handle the quantity. You should always check the status when you use this macro.

Example:

To check if a gripper whose ID is 1 has sufficient zones to pick 4 parts, use the following instruction. The status is returned in Numeric register 7. If the status is non-zero, then the gripper does not have sufficient zones.

```
CALL PKGRCHKQTY(Gripper ID=1,Quantiry=4,Stat Reg=7)
```

6.14.2.3 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)
```

6.14.2.4 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)
```

6.14.2.5 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)
```

6.14.2.6 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)
```

6.14.2.7 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)
```

6.14.2.8 PKGRGETZNUT.PC

This program is used to get the UTOOL of a Zone of a Gripper specified in the *iRPickTool* 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)
```

6.14.2.9 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
```

6.14.2.10 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)
```

6.14.3 Robot-Related Programs

The two letters following [PK] are [RB] namely, the names of these programs begin with these four letters, [PKRB].

6.14.3.1 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 iRPickTool 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)
```

6.14.3.2 PKRBGETFSCNT.PC

This program gets the fixed station count for a specified operation (pick or place) by the specified robot based on analyzing the *iRPickTool* 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 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)
```

6.14.3.3 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)
```

6.14.3.4 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)
```

6.14.3.5 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 6.16 "SETUP OF PRE-GROUPING". This program can be used in 7DC3/23(V8.30P/23) or later.

- 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 a conveyor station on the infeed conveyor.

Argument 3:

Specify an ID of an outfeed conveyor.

Argument 4:

Specify an ID of a 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)
```

6.14.4 Conveyor-Related Programs

The two letters following [PK] are [CV] namely, the names of these programs begin with these four letters, [PKCV].

6.14.4.1 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 If you specify Argument 5 (model ID), the argument guidance is not displayed.
- 3 When you use multiple controllers and call KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPPNAM.PC, PKSNSETMVNAM.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 PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 "Need to synchronize data" occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

6.14.4.2 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. This program can be used in 7DC3/02(V8.30P/02) or later.

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, PKSNSETMVNAM.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 PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 "Need to synchronize data" occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

6.14.4.3 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, PKSNSETVPPNAM.PC, PKSNSETMVNAM.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 PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 "Need to synchronize data" occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

6.14.4.4 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, PKSNENBSSENS.PC, PKSNSETVPPNAM.PC, PKSNSETMVNAM.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 PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 "Need to synchronize data" occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

6.14.4.5 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)
```


6.14.5 Sensor-Related Programs

The two letters following [PK] are [SN] namely, the names of these programs begin with these four letters, [PKSN].

6.14.5.1 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. This program can be used in 7DC3/03(V8.30P/03) or later.

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, PKSNNBSENS.PC, PKSNNSETVPNAM.PC, PKSNNSETMVNAM.PC, PKSNNSETVTOFS.PC, PKSNNSETMTOFS.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC), be careful about the following points.
 - Call the listed KAREL programs for setting parameters before PKWCSTART or PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 "Need to synchronize data" occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

6.14.5.2 PKSNNSETVPNAM.PC

This program changes a vision process in a sensor. 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 PKSNNSETVPNAM('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, PKSNENBSSENS.PC, PKSNSETVPNAM.PC, PKSNSETMVNAM.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 PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 "Need to synchronize data" occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

6.14.5.3 PKSNSETMVNAM.PC

When you use [DI] + [Find part by multiview], this program changes a vision process 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. This program can be used in 7DC3/18(V8.30P/18) or later.

Argument 1:

Specify the name of the sensor using a character string.

Argument 2:

Specify the name of the vision process for view 1 using a character string.

Argument 3:

Specify the name of the vision process for view 2 using a character string.

Example:

The following instruction changes vision process names of multiview which are selected in "SENS1" to "VT1" and "VT2".

```
CALL PKSNSETMVNAM('SENS1',VP Name 1='VT1',VP Name 2='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, PKSNENBSSENS.PC, PKSNSETVPNAM.PC, PKSNSETMVNAM.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 PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 "Need to synchronize data" occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

6.14.5.4 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. This program can be used in 7DC3/18(V8.30P/18) or later.

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, PKSNENBSSENS.PC, PKSNSETVPNAM.PC, PKSNSETMVNAM.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 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCCHECK is called.

6.14.5.5 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. This program can be used in 7DC3/18(V8.30P/18) or later.

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, PKSNENBSSENS.PC, PKSNSETVPNAM.PC, PKSNSETMVNAM.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 PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 "Need to synchronize data" occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

6.14.5.6 PKSNSTART.PC

This KAREL 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
```

6.14.5.7 PKSNSTOP.PC

This KAREL 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
```

6.14.6 Conveyor Station-Related Programs

The two letters following [PK] are [CS] namely, the names of these programs begin with these four letters, [PKCS].

6.14.6.1 PKCSGETID.PC

This KAREL 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 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)
```

NOTE

If you specify Argument 3 (group number), the argument guidance is not displayed.

6.14.6.2 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])

6.14.6.3 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.

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 parts 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 5, "VARIATIONS".

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 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 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 wait 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.
- 3 If you specify Argument 6 (model ID), the argument guidance is not displayed.

6.14.6.4 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)
```

⚠ 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.

6.14.6.5 PKCSACKQUE.PC

After the processing status of PKCSGETQUE 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.

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)
```

6.14.6.6 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 iRPickTool.

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 number of the register to store the overlap check result. If the information is input with no overlapping part, 0 is stored in the specified register. If there is any overlapping part 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)
```

6.14.6.7 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.

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".

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)
```

NOTE

If you specify Argument 8 (model ID), the argument guidance is not displayed.

6.14.6.8 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.

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".

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:

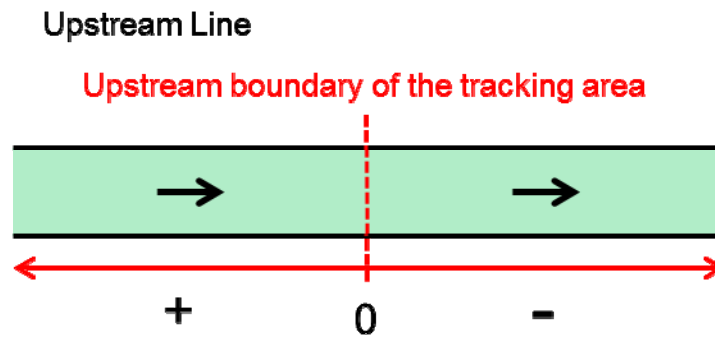
Normally, specify 1. When a robot gripper capable of holding multiple parts is used and you want the robot to consecutively pick as many parts as the number of parts that its gripper can hold, specify which part to pick next. For example, if the gripper can hold four parts and it is now holding two parts, 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.

**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)
```

NOTE

If you specify Argument 8 (model ID), the argument guidance is not displayed.

6.14.6.9 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 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 is referred to as the expected number of parts. 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 $((7 + 3) \div 2 - 4)$.

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.

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, "pkcsgetpfprt 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.

pkcsgetpfprt 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)
```

6.14.6.10 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)
```

6.14.6.11 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. This program can be used in 7DC3/03(V8.30P/03) or later.

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, PKSNENBSSENS.PC, PKSNSETVPNAM.PC, PKSNSETMVNAM.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 PKWCHECK
 - Insert WAIT instruction before PKWCSTART or PKWCHECK 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 PKWCHECK is called, IRPK-079 "Need to synchronize data" occurs. If IRPK-079 occurs 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCHECK is called.

6.14.6.12 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.

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)
```

6.14.6.13 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)
```

6.14.6.14 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 iRPickTool 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 *iRPickTool* 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)
```

6.14.6.15 PKCSGETOPPR.PC

This program gets the “Reference Pos PR” of a conveyor station you specified in the OP_CS_cstnName menu in the *iRPickTool* 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)
```

6.14.6.16 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 *iRPickTool* 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)
```

6.14.6.17 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 *iRPickTool* 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)
```

6.14.6.18 PKCSGETQTY.PC

This program gets the “Quantity” (of parts for the conveyor-station’s robot’s gripper to pick or place) you specified in the OP_CS_cstnName menu in the *iRPickTool* treeview.

Argument 1:

Specify an ID of a conveyor station.

Argument 2:

Specify the Numeric Register in which to return the “Quantity”.

Example:

To get the pick or place quantity of a conveyor station whose ID is 1 in Numeric Register 6, use the following instruction:

```
CALL PKCSGETQTY(CStn ID=1,Quantity Reg=6)
```

6.14.6.19 PKCSGETVR.PC

This program gets the “Part Data VR” of a conveyor station you specified in the OP_CS_cstnName menu in the *iRPickTool* 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)
```

6.14.6.20 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 *iRPickTool* 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)
```

6.14.6.21 PKCSGETENC.PC

This program returns the encoder ID which is used in the specified conveyor station. This program can be used in 7DC3/23(V8.30P/23) or later.

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)
```

6.14.6.22 PKCSGETTRID.PC

This program returns the tray ID which is used in the specified conveyor station. This program can be used in 7DC3/23(V8.30P/23) or later.

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)
```

6.14.7 Fixed Station-Related Programs

The two letters following [PK] are [FS] namely, the names of these programs begin with these four letters, [PKFS].

6.14.7.1 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)
```

6.14.7.2 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])
```

6.14.7.3 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])
```

6.14.7.4 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_FSGETQUEUE 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_FSGETQUEUE 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 PKFSGETQUEUE(FStn ID=R[12:FSTN1 ID],Offset VR=1,Stat Reg=2)
```

**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.
- 2 If you specify Argument 4 (model ID), the argument guidance is not displayed.

6.14.7.5 PKFSACKQUE.PC

After the processing status of PKFSGETQUEUE 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)
```

6.14.7.6 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.

If any of these arguments is invalid, "P_FSPUTBUF 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_FSPUTBUF 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 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)
```

6.14.7.7 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)

6.14.7.8 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**

- 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 (buffer), you must make the taught position conform to the position of the cell 1.
- 2 If you specify Argument 6 (model ID), the argument guidance is not displayed.

6.14.7.9 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)
```

NOTE

If you specify Argument 3 (Vision Register), the argument guidance is not displayed.

6.14.7.10 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 *iRPickTool* 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 *iRPickTool* 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)
```

6.14.7.11 PKFSGETIXDLY.PC

This program gets the "Index delay" you specified in the OP_FS_fstnName menu in the *iRPickTool* 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)
```

6.14.7.12 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 *iRPickTool* 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)
```

6.14.7.13 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 iRPickTool 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)
```

6.14.7.14 PKFSGETOPPR.PC

This program gets the "Reference Pos PR" of a fixed station you specified in the OP_FS_fstnName menu in the iRPickTool 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)
```

6.14.7.15 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 iRPickTool 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)
```

6.14.7.16 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 *iRPickTool* 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)
```

6.14.7.17 PKFSGETQTY.PC

This program gets the “Quantity” (of parts for the fixed-station’s robot’s gripper to pick or place) you specified in the OP_FS_fstnName menu in the *iRPickTool* treeview.

Argument 1:

Specify an ID of a fixed station.

Argument 2:

Specify the Numeric Register in which to return the “Quantity”.

Example:

To get the pick or place quantity of a fixed station whose ID is 1 in Numeric Register 6, use the following instruction:

```
CALL PKFSGETQTY(FStn ID=1,Quantity Reg=6)
```

6.14.7.18 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, *iRPickTool* 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, *iRPickTool* 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)
```

6.14.7.19 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 iRPickTool 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)
```

6.14.7.20 PKFSGETTRID.PC

This program gets the ID of the tray that you assigned to a Fixed Station in the iRPickTool 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)
```

6.14.7.21 PKFSGETUF.PC

This program gets the UFrame (UF) number that you assigned to a Fixed Station Origin in the iRPickTool 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)
```

6.14.7.22 PKFSGETVR.PC

This program gets the “Part Data VR” of a fixed station you specified in the OP_CS_fstnName menu in the iRPickTool 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)
```

6.14.7.23 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)
```

6.14.7.24 PKFSSETRDY.PC

This program sets the internal status of “fixed station ready” (“Ready” or “NotReady”). Refer to the KAREL macro PKFSGETRDY.PC to know the details of how iRPickTool 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)
```

6.14.8 Tray-Related Programs

The two letters following [PK] are [TR] namely, the names of these programs begin with these four letters, [PKTR].

6.14.8.1 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 PKSNSSTART, 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, PKSNSETMVNAM.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 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. If the listed KAREL programs for setting parameters are called in multiple controllers, IRPK-079 can occur when PKWCSTART or PKWCCHECK is called.

6.14.8.2 PKTRGETNUMCL.PC

This program returns the number of cells in the specified layer of the specified tray. This program can be used in 7DC3/23(V8.30P/23) or later.

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)
```

6.15 STANDARD TP PROGRAMS

This section describes the standard TP programs provided with the J945: *iRPickTool* option. These programs allow you to run any picking or visual tracking application based on the data you setup in the *iRPickTool* 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-3iA/6S, M-2iA/3S, the LR Mate series like the LR Mate 200iD, 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).

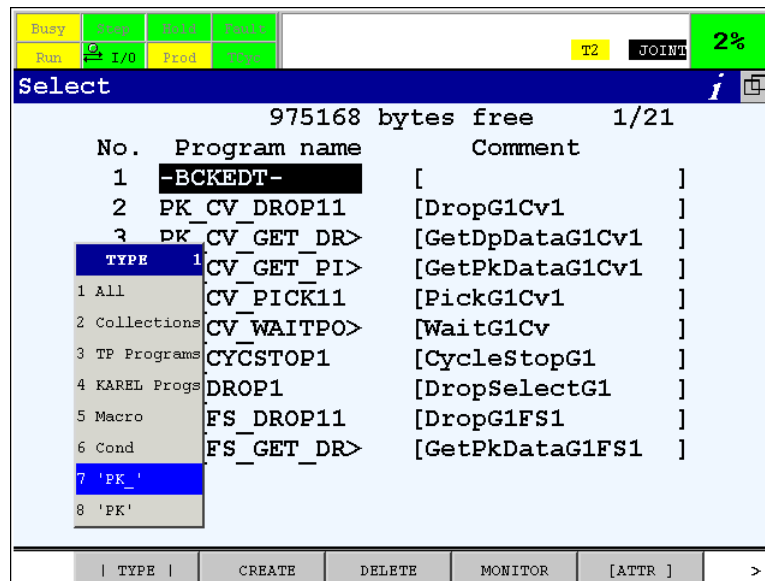
Many other functions like automatic move to wait position when there are no parts, part presence checks for the gripper, etc.

6.15.1 TP Program Naming Conventions

All the *iRPickTool* 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 *iRPickTool* programs by selecting the PK_ filter. See the figure below.

You may use the "PK" filter to view all the *iRPickTool* 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.

6.15.2 TP Program Details

6.15.2.1 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]
21	PK_PLG_REC_CHG	[RecChangeFromPLC]

6.15.2.2 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:

Initialize Data: Get the data set in the iRPickTool 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.

Start the sensor task to be ready to detect parts from the camera, DI or HDI sensors and put them into the part queue.

Move the robot to the perch position

When pre-grouping is used, the main program for pre-grouping PK_PRGR_MAIN1.TP is called.

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.

If cycle stop is invoked, process the cycle stop function.

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]

```

6.15.2.3 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:

Checking if *iRPickTool* data is synchronized across all controllers.

Getting the active recipe id.

Determining the Robot ID for the group 1 robot.

Determining the Robot's gripper ID.

Getting the zone data of the gripper and assigning to UTOOLS.

Setting the P[1] position of the PK_PERCH1.TP program whenever a recipe is restored or switched to using PKCOPYPR2POS KAREL macro.

Determining if the pick station is a conveyor station or a fixed station.

Determining if the place station is a conveyor station or a fixed station.

Get all the data from the treeview for the robot and the required conveyor and fixed stations.

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: *iRPickTool* / Option Package option is installed.

Getting type for pre-grouping. When pre-grouping is not used, the type is 0.

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]

```

6.15.2.4 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]

```

6.15.2.5 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 execute only when the R857: iRPickTool / Option Package Add-on is installed. For additional details on the buffer logic in this program, please refer to the section 7.7 “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 6.16 “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 Cvld]=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]

```


6.15.2.6 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: *iRPickTool* / Option Package Add-on is installed. For additional details on the buffer logic, please refer to the section 7.7 “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]
```

6.15.2.7 PK_GR_GET_UTOOLS1.TP

This program sets the UTOOL for each zone of the gripper used by the group 1 robot. This program Uses the KAREL macro PKGRGETZNCNT to get the zone count of the specified gripper
Uses the KAREL macro PKGRGETZNUT to get the UTOOL value specified in the Zone menu of the Gripper in the *iRPickTool* treeview.
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: -- Arg 1: Robot id
5: -- Arg 2: Gripper id
6:
7: -- Get num zones in gripper
8: CALL PKGRGETZNCNT(Gripper ID=AR[2],ZoneCount Reg=96)
9: FOR R[95:TemporaryUse1]=1 TO R[96:TemporaryUse2]
10: -- Get each zone UT to PR[1]
```

```

11: CALL PKGRGETZNUT(Robot ID=AR[1],Gripper ID=AR[2],
    Zone Index=R[95:TemporaryUse1],ZoneUTool PR=51)
12: -- Assign zone UT to UTOOL
13: UTOOL[R[95]]=PR[51:TemporaryUse1]
14: ENDFOR
15:
[End]

```

6.15.2.8 PK_CV_GET_PICK_DATA11.TP

This program is used to get the data setup in the iRPickTool treeview for a pick operation at a conveyor station of a conveyor. This program works as follows.

Gets the conveyor ID of the first pick from \$PK_MAIN[1].\$PK_CV[1] (which is initialized when the KAREL macro PKRBGETCVCNT.PC is executed).

Get the name of the first conveyor to String register 4 (SR[4])

Gets the ID of the conveyor station on the conveyor using the KAREL macro PKCVGETNAME.PC.

Gets the pick quantity using the KAREL macro PKCSGETQTY.

Checks if there are enough gripper zones to handle the quantity by using the KAREL macro PKGRCHKQTY.PC

Gets the “Pick or drop multiple” property for the pick operation using the KAREL macro PKCSGETPMUL.PC.

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.

Gets the FLG used for checking the “Skip Outbound” condition in the tracking pick program using the KAREL macro PKRBGETSFLG.PC.

Gets the ID of the tray used on the conveyor using the KAREL macro PKCSGETTRID.

Clears any pending ACK to avoid the “GetQueue before Ackque” error using the KAREL macro PKCSCLRACK.PC.

Clears the queue of the conveyor station based on the setting of the system variable \$PKCLRQ using the KAREL macro PKCSCLRQUE.PC.

Gets the value of the “pick only reachable parts” set in the Operation menu (OP_CS_xxxx in the treeview) using the KAREL macro PKCSGETPOSCK. For more information on the “Pick only reachable parts”, refer to the the section 7.8 “EFFICIENCY TOOLS”.



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 19: 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: -- PKCSGETQTY(stn_id, reg_qty)
12: CALL PKCSGETQTY(CStn ID=R[110:Pk1x StnId],Quantity Reg=111)
13: CALL PKGRCHKQTY(Gripper ID=R[104:Gripper Id G1],Quantity=R[111:Pk1xQty],
    Stat Reg=95)
14: IF R[95:TemporaryUse1]=0,JMP LBL[100]
15: UALM[5]
16: LBL[100]
17: -- PKCSGETPMUL(stn_id, reg_pkdpmul)
18: CALL PKCSGETPMUL(CStn ID=R[110:Pk1x StnId],PikMultiple Reg=112)
19: CALL PKCSGETAPPR(CStn ID=R[110:Pk1x StnId],ApproachPR Reg=95)
20: -- Get the Robot's skip outbound flag
21: -- PKRBGETSFLG(robot_id, reg_skipflg)
22: CALL PKRBGETSFLG(Robot ID=R[103:Robot Id G1],SkipFlag Reg=102)
23: !Get CStn's tray id
24: CALL PKCSGETTRID(CStn ID=R[110:Pk1x StnId],Tray ID Reg=121)
25: -- Clear any pending acks
26: CALL PKCSCLRACK(CStn ID=R[110:Pk1x StnId])
27: ! Clear the queues based on
28: ! $PKCLRQ for G1 CStns
29: R[95:TemporaryUse1]=$PKCLRQ
30: IF R[95:TemporaryUse1]=0,JMP LBL[5000]
31: CALL PKCSCLRQUE(CStn ID=R[110:Pk1x StnId])
32:
33: LBL[5000: Eff Tools]
34: !Get position check flag
35: !PKSCSGETPOSCK(cs_id,
36: ! operation, reg_PosCheck)
37: CALL PKCSGETPOSCK(CStn ID=R[110:Pk1x StnId],Pick=0,PosCheck Reg=115)
38: LBL[10000:End]
[End]

```

6.15.2.9 PK_CV_GET_DROP_DATA11.TP

This program is used to get the data setup in the *iRPickTool* treeview for a drop or place operation at a conveyor station of a conveyor. This program works as follows.

Gets the conveyor ID of the first drop from \$PK_MAIN[1].\$DP_CV[1] (which is initialized when the KAREL macro PKRBGETCVCNT.PC is executed).

Get the name of the first conveyor to String register 5 (SR[5])

Gets the ID of the conveyor station on the conveyor using the KAREL macro PKCVGETNAME.PC.

Gets the place quantity using the KAREL macro PKCSGETQTY.

Checks if there are enough gripper zones to handle the quantity by using the KAREL macro PKGRCHKQTY.PC

Gets the “Pick or drop multiple” property for the pick operation using the KAREL macro PKCSGETPMUL.PC.

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.

Gets the FLG used for checking the “Skip Outbound” condition in the tracking pick program using the KAREL macro PKRBGETSFLG.PC.

Clears any pending ACK to avoid the “GetQueue before Ackque” error using the KAREL macro PKCSCLRACK.PC.

Clears the queue of the conveyor station based on the setting of the system variable \$PKCLRQ using the KAREL macro PKCSCLRQUE.PC

Gets the value of the “pick only reachable parts” set in the Operation menu (OP_CS_xxxx in the treeview) using the KAREL macro PKCSGETPOSCK. For more information on the “Pick only reachable parts”, refer to the section 7.8 “EFFICIENCY TOOLS”.

**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 19: 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: -- PKCSGETQTY(stn_id, reg_qty)
12: CALL PKCSGETQTY(CStn ID=R[130:Dp1x StnId],Quantity Reg=131)
13: CALL PKGRCHKQTY(Gripper ID=R[104:Gripper Id G1],Quantity=R[131:Dp1xQty],
  Stat Reg=95)
14: IF R[95:TemporaryUse1]=0,JMP LBL[100]
15: UALM[5]
16: LBL[100]
17: -- PKCSGETPMUL(stn_id, reg_pkdpmul)
18: CALL PKCSGETPMUL(CStn ID=R[130:Dp1x StnId],PikMultiple Reg=132)
19: CALL PKCSGETAPPR(CStn ID=R[130:Dp1x StnId],ApproachPR Reg=95)
20: -- Get the Robot's skip outbound flag
21: -- PKRBGETSFLG(robot_id, reg_skipflg)
22: CALL PKRBGETSFLG(Robot ID=R[103:Robot Id G1],SkipFlag Reg=102)
23: !Get CStn's tray id
24: CALL PKCSGETTRID(CStn ID=R[130:Dp1x StnId],Tray ID Reg=141)
25: -- Clear any pending acks
26: CALL PKCSCLRACK(CStn ID=R[130:Dp1x StnId])
27: ! Clear the queues based on
28: ! $PKCLRQ for G1 CStns
29: R[95:TemporaryUse1]=$PKCLRQ
30: IF R[95:TemporaryUse1]=0,JMP LBL[5000]
31: CALL PKCSCLRQUE(CStn ID=R[130:Dp1x StnId])
32:
33: LBL[5000: Eff Tools]
34: !Get position check flag
35: !PKSCSGETPOSCK(cs_id,
36: ! operation, reg_PosCheck)
37: CALL PKCSGETPOSCK(CStn ID=R[130:Dp1x StnId],Place=1,PosCheck Reg=135)
38: LBL[10000:End]
[End]

```

6.15.2.10 PK_FS_GET_PICK_DATA11.TP

This program is used to get the data setup in the iRPickTool treeview for a pick operation at a Fixed station. This program

Gets the Fixed Station ID of the first pick from \$PK_MAIN[1].\$PK_FS[1] (which is initialized when the KAREL macro PKRBGETFSCNT.PC is executed).

Gets the pick quantity using the KAREL macro PKFSGETQTY.PC.

Checks if there are enough gripper zones to handle the quantity by using the KAREL macro PKGRCHKQTY.PC

Gets the "Pick or drop multiple" property for the pick operation using the KAREL macro PKFSGETPMUL.PC.

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.

Gets the Station Ready DI of the Fixed Station (for later station_ready? Check) using the KAREL macro PKFSGETRDYDI.PC.

Gets the Index Out DO digital output of the Fixed Station (for later indexing the fixed station) using the KAREL macro PKFSGETIXODO.PC.

Sets the initial value of the internal station_ready flag to FALSE (later if station is indeed found to be ready during the station_ready? check, it will be set to TRUE) using the KAREL macro PKFSSETRDY.PC.

Checks if "simulate indexing" is enabled, and if so simulates the Station Ready DI port to go high using the KAREL macro PKFSGETIXSIM.PC

Gets the ID of the tray at the Fixed Station using the KAREL macro PKFSGETTRID.PC

Clears any pending ACK to avoid the "GetQueue before Ackque" error using the KAREL macro PKFSCLRACK.PC.

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 16: 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: -- PKFSGETQTY(stn_id, reg_qty)
9: CALL PKFSGETQTY(FStn ID=R[110:Pk1x StnId],Quantity Reg=111)
10: CALL PKGRCHKQTY(Gripper ID=R[104:Gripper Id G1],Quantity=R[111:Pk1xQty],
    Stat Reg=95)
11: IF R[95:TemporaryUse1]=0,JMP LBL[100]
12: UALM[5]
13: LBL[100]
```

```

14: -- PKFSGETPMUL(stn_id, reg_pkdpmul)
15: CALL PKFSGETPMUL(FStn ID=R[110:Pk1x StnId],PikMultiple Reg=112)
16: CALL PKFSGETAPPR(FStn ID=R[110:Pk1x StnId],ApproachPR Reg=95)
17: CALL PKFSGETRDYDI(FStn ID=R[110:Pk1x StnId],ReadyDI Reg=118)
18: CALL PKFSGETIXODO(FStn ID=R[110:Pk1x StnId],IndexOutDO Reg=119)
19: CALL PKFSSETRDY(FStn ID=R[110:Pk1x StnId],Ready=1)
20: CALL PKFSGETIXSIM(FStn ID=R[110:Pk1x StnId],IndexSim Reg=95)
21: CALL PKFSGETTRID(FStn ID=R[110:Pk1x StnId],Tray ID Reg=121)
22: CALL PKFSCLRACK(FStn ID=R[110:Pk1x StnId])
23: IF R[121:Pk1x TrayId]=0,JMP LBL[10000]
24: R[95:TemporaryUse1]=$PKCLRQ
25: IF R[95:TemporaryUse1]=0,JMP LBL[1000]
26: CALL PKFSCLRQUE(FStn ID=R[110:Pk1x StnId])
27: CALL PKFSPUTQUE(FStn ID=R[110:Pk1x StnId])
28: -- Reset tray cell counter
29: R[122:Pk1x TrayCtr]=1
30:
31: LBL[1000]
32:
33: LBL[10000: End]
[End]

```

6.15.2.11 PK_FS_GET_DROP_DATA11.TP

This program is used to get the data setup in the *iRPickTool* treeview for a drop or place operation at a Fixed station. This program works as follows.

Gets the Fixed Station ID of the first place from \$PK_MAIN[1].\$DP_FS[1] (which is initialized when the KAREL macro PKRBGETFSCNT.PC is executed).

Gets the place quantity using the KAREL macro PKFSGETQTY.

Checks if there are enough gripper zones to handle the quantity by using the KAREL macro PKGRCHKQTY.PC

Gets the “Pick or drop multiple” property for the pick operation using the KAREL macro PKFSGETPMUL.PC.

Gets the approach offset of the Fixed station from the treeview data and sets it to the approach offset Position Register using PKFSGETAPPR.PC.

Gets the Station Ready DI of the Fixed Station (for later station_ready? Check) using the KAREL macro PKFSGETRDYDI.PC.

Gets the Index Out DO digital output of the Fixed Station (for later indexing the fixed station) using the KAREL macro PKFSGETIXODO.PC.

Sets the initial value of the internal station_ready flag to FALSE (later if station is indeed found to be ready during the station_ready? check, it will be set to TRUE) using the KAREL macro PKFSSETRDY.PC.

Checks if “simulate indexing” is enabled, and if so simulates the Station Ready DI port to go high using the KAREL macro PKFSGETIXSIM.PC

Gets the ID of the tray at the Fixed Station using the KAREL macro PKFSGETTRID.PC

Clears any pending ACK to avoid the “GetQueue before Ackque” error using the KAREL macro PKFSCLRACK.PC.

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 16: 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: -- PKFSGETQTY(stn_id, reg_qty)
9: CALL PKFSGETQTY(FStn ID=R[130:Dp1x StnId],Quantity Reg=131)
10: CALL PKGRCHKQTY(Gripper ID=R[104:Gripper Id G1],Quantity=R[131:Dp1xQty],
    Stat Reg=95)
11: IF R[95:TemporaryUse1]=0,JMP LBL[100]
12: UALM[5]
13: LBL[100]
14: -- PKFSGETPMUL(stn_id, reg_pkdpmul)
15: CALL PKFSGETPMUL(FStn ID=R[130:Dp1x StnId],PikMultiple Reg=132)
16: CALL PKFSGETAPPR(FStn ID=R[130:Dp1x StnId],ApproachPR Reg=95)
17: CALL PKFSGETRDYDI(FStn ID=R[130:Dp1x StnId],ReadyDI Reg=138)
18: CALL PKFSGETIXODO(FStn ID=R[130:Dp1x StnId],IndexOutDO Reg=139)
19: CALL PKFSSETRDY(FStn ID=R[130:Dp1x StnId],Ready=1)
20: CALL PKFSGETIXSIM(FStn ID=R[130:Dp1x StnId],IndexSim Reg=95)
21: CALL PKFSGETTRID(FStn ID=R[130:Dp1x StnId],Tray ID Reg=141)
22: CALL PKFSCLRACK(FStn ID=R[130:Dp1x StnId])
23: IF R[141:Dp1x TrayId]=0,JMP LBL[10000]
24: R[95:TemporaryUse1]=$PKCLRQ
25: IF R[95:TemporaryUse1]=0,JMP LBL[1000]
26: CALL PKFSCLRQUE(FStn ID=R[130:Dp1x StnId])
27: CALL PKFSPUTQUE(FStn ID=R[130:Dp1x StnId])
28: -- Reset tray cell counter
29: R[142:Dp1x TrayCtr]=1
30:
31: LBL[1000]
32:
33: LBL[10000: End]
[End]

```

6.15.2.12 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.

Sets the skip outbound FLG to OFF before beginning the pick moves.

Gets a part from the part queue using either PKCSGETQUE or PKCSGETQINRG depending upon whether “pick only reachable” is enabled or not. PKCSGETQINRG allocates only parts that are reachable by the robot.

Regardless of getting or not getting a part from the queue, if the property “Wait Position” is enabled in the *iRPickTool* 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.

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.

If PKCSGETQUE or PKCSGETQINRG did not return a part, it makes the request to get the part again.

If PKCSGETQUE or PKCSGETQINRG returned a part,

Sets the UTOOL to the gripper’s first zone’s UTOOL.

If the property “Pick only reachable parts” is enabled for this conveyor station operation, checks if the approach position is reachable using PKCSCHKPOS. If it is not reachable, 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.

Moves the robot to the approach position.

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.

If the move to approach succeeded, then checks if the pick position is reachable using PKCSCHKPOS. If it is not reachable, 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.

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.

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.

If the move to the pick position succeeded, then it gets the pick delay using the PKGRGETPKDLY.PC macro.

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.

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.

If the part presence check succeeds, it sets the payload of the robot to “Full gripper” or Payload[2].

It moves the robot to the retreat position.

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.

Note that when there is an enabled buffer, this program sets the “picked” flag (R[205]) to 1 when a pick is successful. When a part is not available from the queue, a “skip outbound” occurs or when a part is dropped, it exits the program to PK_PICK1.TP so that the buffer conditions can be checked.

NOTE

To allocate only parts that are reachable by the robot using PKCSGETQINRG and to check if the position is reachable using PKCSCHKPOS are performed only if the R857: *iRPickTool* / Option Package option is installed. For details, refer to the section 7.8 “EFFICIENCY TOOLS”.

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:
36: LBL[50]
37:
38: LBL[60]
39: //-- Smart Pick
40:
41:
42: LBL[70]
43: //-- Check if a reject part
44:
45:
46: LBL[80]
47:
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=1,Stat Reg=108)
53: IF R[108:PosNotReach G1]<>0,JMP LBL[1000]

```

```

54:
55: LBL[85]
56: -- Move to approach
57:L PR[57:Pk1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[1]
   Tool_Offset,PR[56:Pk1 Cv Ap Ofst]
58: -- Flag on when Skip outbound true
59: IF (F[R[102]]),JMP LBL[1000]
60:
61: -- Set Pick/drop registers
62: CALL PK_GR_GET_ENDZONE1(PickDropCounter=R[107:Counter ],
   Multiple=R[112:Pk1x PkMult],Quantity=R[111:Pk1x Qty],End Zone Reg=106)
63: IF R[115:Pk1x UsePosChk]=0,JMP LBL[88]
64: -- Check if tracking position is reachable
65: CALL PKCCHKPOS(CStn ID=R[110:Pk1x StnId],Offset VR=1,Ref Pos PR=57,
   Appr Ofst PR=0,Tool Offset=1,Stat Reg=108)
66: IF R[108:PosNotReach G1]<>0,JMP LBL[1000]
67:
68: LBL[88]
69: -- Move to pick pos
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])
71: -- Chk Skip Pick
72: IF (F[R[102]]),JMP LBL[1000]
73: -- Pick delay
74: CALL PKGRGETPKDLY(Gripper ID=R[104:Gripper Id G1],PickDelay Reg=113)
75: WAIT R[113]
76: -- Check part pres
77: CALL PKCSGETPPCHK(CStn ID=R[110:Pk1x StnId],PartPresChk Reg=116)
78: IF R[116:Pk1x UsePPChk]=0,JMP LBL[90]
79: -- PKGRCHKPP grip_id, operation,
80: -- stat_reg, StartZone,
81: -- EndZone (opt)
82: 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])
83: 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]

```

6.15.2.13 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.

Sets the skip outbound FLG to OFF before beginning the place moves.

Gets a part from the part queue using either PKCSGETQUE or PKCSGETQINRG depending upon whether “pick only reachable” is enabled or not. PKCSGETQINRNG allocates only parts that are reachable by the robot. .

Regardless of getting or not getting a part from the queue, if the property “Wait Position” is enabled in the iRPickTool 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.

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.

If PKCSGETQUE or PKCSGETQINRG did not return a part, it makes the request to get the part again.

If PKCSGETQUE or PKCSGETQINRG returned a part,

Sets the UTOOL to the gripper’s first zone’s UTOOL.

If the property “Pick only reachable parts” is enabled for this conveyor station operation, checks if the approach position is reachable using PKCSCHKPOS. If it is not reachable, 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.

Moves the robot to the approach position.

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.

If the move to approach succeeded, then checks if the place position is reachable using PKCSCHKPOS. If it is not reachable, 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.

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.

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.

If the move to the place position succeeded, then it gets the place delay using the PKGRGETDPDLY.PC macro.

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.

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.

If the part presence check succeeds, it sets the payload of the robot to “Empty gripper” or Payload[1].

It moves the robot to the retreat position.

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.

Note that when there is an enabled buffer, this program sets the “picked” flag (R[205]) to 0 when a placement is successful. When a part is not available from the queue, a “skip outbound” occurs or when a part is dropped, it exits the program to PK_DROP1.TP so that the buffer conditions can be checked.

NOTE

To allocate only parts that are reachable by the robot using PKCSGETQINRNG and to check if the position is reachable using PKCSCHKPOS are performed only if the R857: *iRPickTool* / Option Package option is installed. For details, refer to the section 7.8 “EFFICIENCY TOOLS”.

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: -- PKCSGETQINRNG allocates only
12: -- reachable parts (requires R846)
13: CALL PKCSGETQINRNG(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 PKCCHKPOS(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 PKCCHKPOS(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]

```

6.15.2.14 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 used for the move to wait position.

Argument 4: Numeric Register number used for knowing if pick only reachable parts is used.

This program is used to move the robot to the wait position at both the “pick” conveyor station as well as the “place” conveyor station by passing in different arguments. Based on argument 4, the robot will not move to the wait position if it is unreachable. Otherwise it will do the move. The check for reachability is done using PKOPPOSCHK which will be available only if the R857: iRPickTool / Option Package Add-on is installed. For details, refer to 7.8 “EFFICIENCY TOOLS”.

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]

```

6.15.2.15 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.

Gets the UFRAME number of the Fixed Station that you set in the Fixed Station menu of the iRPickTool treeview using the PKFSGETUF.PC macro and sets the system's UFRAME_NUM to it.

Checks if the Fixed Station is Ready for pick using the TP macro PK_FS_READY.MR macro.

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.

Gets a part from the part queue using PKFSGETQUE.

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.

If PKFSGETQUE returned a part,

Sets the UTOOL to the gripper's first zone's UTOOL.

Moves the robot to the approach position.

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.

If the move to the place position succeeded, then it gets the place delay using the PKGRGETDPDLY.PC macro.

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.

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).

If the part presence check succeeds, it sets the payload to "full gripper" payload which is Payload[2].

It moves the robot to the retreat position.

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: -- PKFSGETQUEUE(fs_id, cell_data_vr, stat_reg, model_id (opt))
27: CALL PKFSGETQUEUE(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]

```

6.15.2.16 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

Gets the UFRAME number of the Fixed Station that you set in the Fixed Station menu of the iRPickTool treeview using the PKFSGETUF.PC macro and sets the system's UFRAME_NUM to it.

Checks if the Fixed Station is Ready for pick using the TP macro PK_FS_READY.MR macro.

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.

Gets a part from the part queue using PKFSGETQUE.

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.

If PKFSGETQUE returned a part,

Sets the UTOOL to the gripper's first zone's UTOOL.

Moves the robot to the approach position.

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.

If the move to the place position succeeded, then it gets the place delay using the PKGRGETDPDLY.PC macro.

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.

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).

If the part presence check succeeds, it sets the payload to “empty gripper” payload which is Payload[1].

It moves the robot to the retreat position.

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]

```

6.15.2.17 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

Determines the Index Out DO for the Fixed Station based on the operation parameter.

Turns on the Index Out DO.

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]

```

6.15.2.18 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

Determines the Index Out DO for the Fixed Station based on the operation parameter.

If the Index Out DO is high, then return “not ready” in the status register argument and return.

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.

Then determine if the station ready DI is high. If not high, then return “Not Ready” in the status register argument and return.

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]

```

6.15.2.19 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: Pick or drop quantity used at the Conveyor Station or the Fixed station in which the robot is operating.

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]

```

6.15.2.20 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]

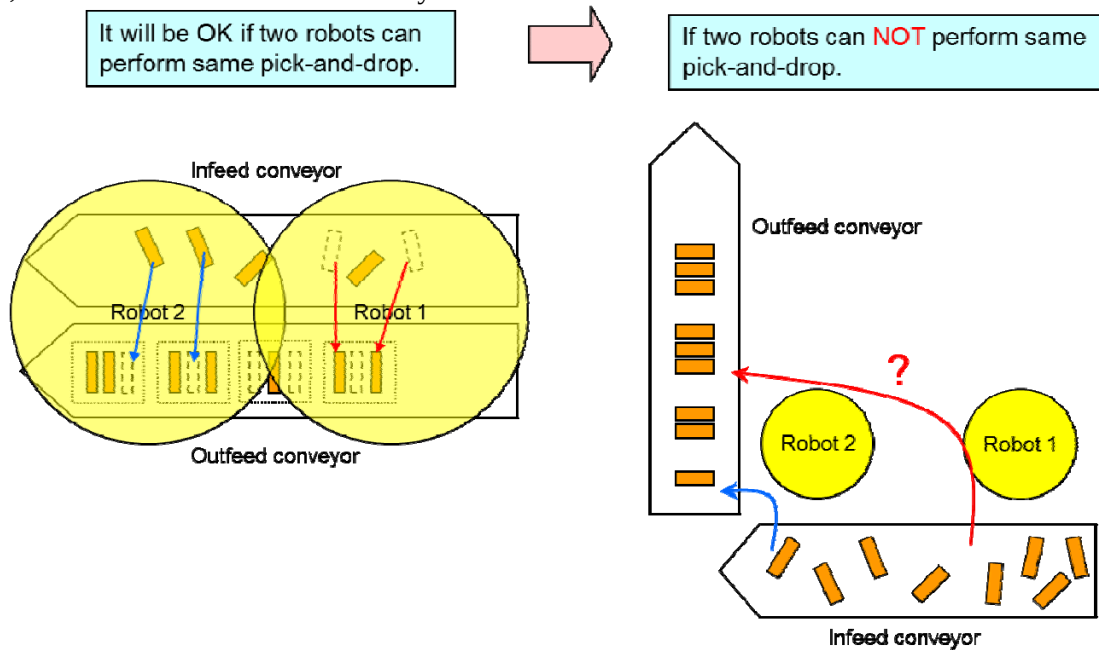
```

6.16 SETUP OF PRE-GROUPING

This section describes the setup procedure of the pre-grouping function which arranges parts on the same conveyor. 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.



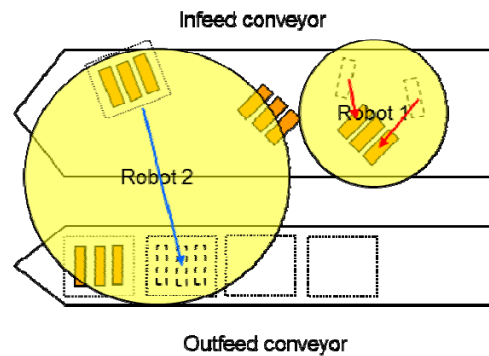
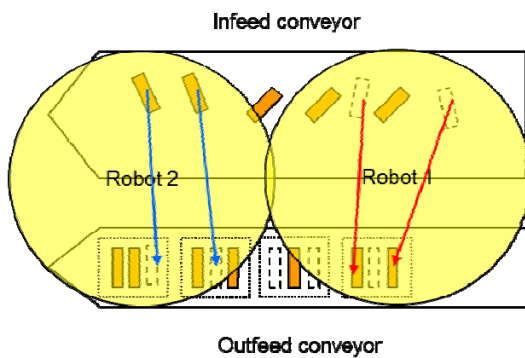
Example 2

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.

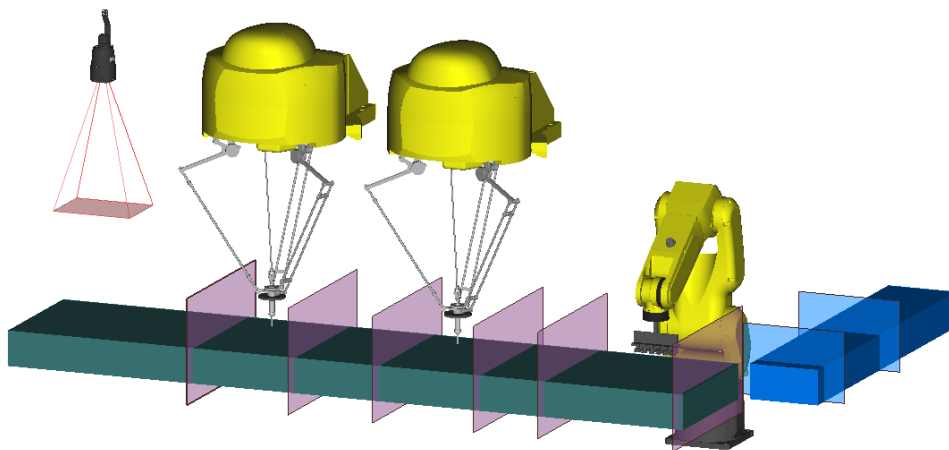
Two robots can perform same pick-and-drop but path may be too long to achieve requested throughputs.



Sometimes shorter path and pre-grouping can achieve the requested throughputs instead.



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.



The system configuration and tree view composition in this case are shown in Fig. 6.16 (a).

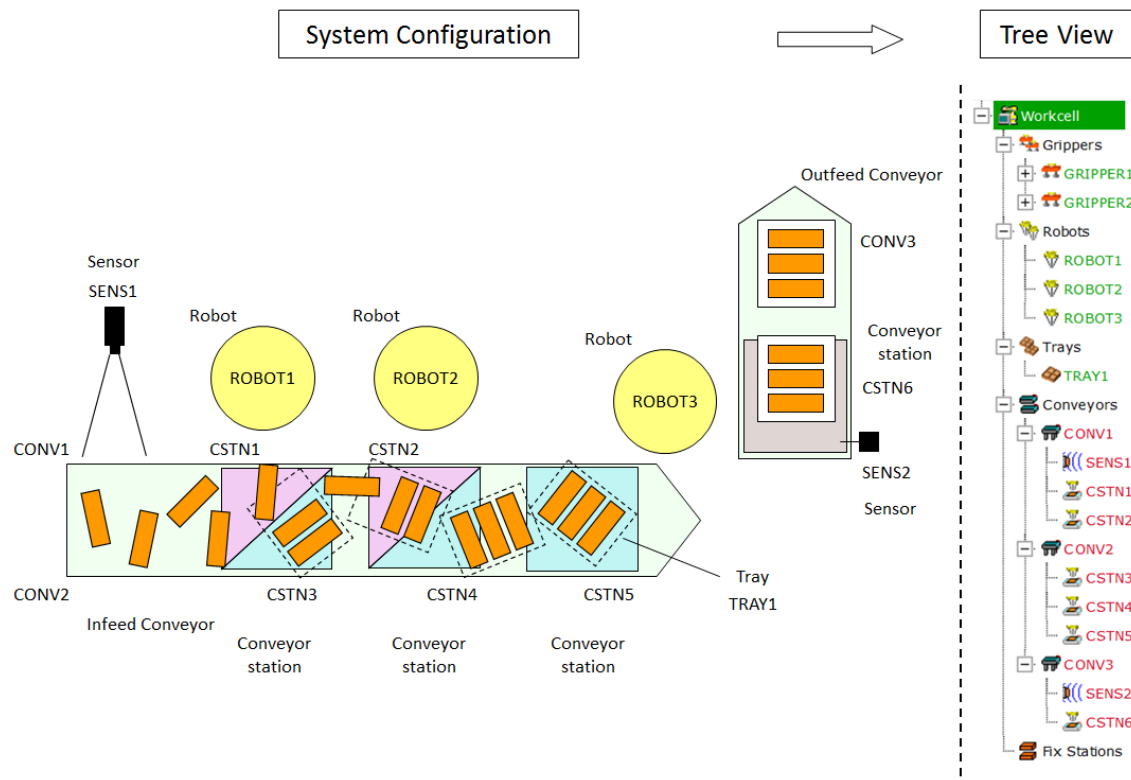


Fig. 6.16 (a)

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.6.16 (a) 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.

6.16.1 Key Concepts

This section describes terms related to a pre-grouping function.

Infeed Conveyor

This is a conveyor object for picking operation.

In Fig. 6.16 (a), CONV1 is an infeed conveyor.

Pre-grouping Conveyor

This is a conveyor object for arranged placement or pre-grouping operation. In Fig. 6.16 (a), 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 is not moved when arranging.

6.16.2 Setup of a Workcell

The setup of the Workcell is done as usual. There are no special steps for pre-grouping. Refer to 6.1 "SETUP OF A WORKCELL".

6.16.3 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 6.2 "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. 6.16 (a), add a zone per three parts.

6.16.4 Setup of a Robot

The setup of the the Robot is done as usual. There are no special steps for pre-grouping. Refer to 6.3 "SETUP OF A ROBOT".

6.16.5 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 6.4 "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.

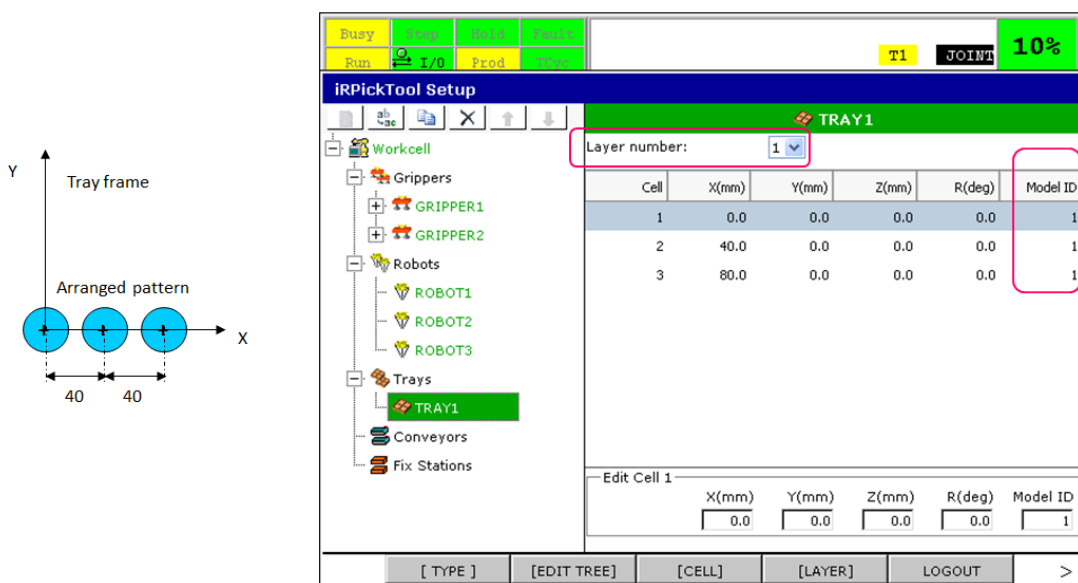
For "Pick" conveyor station operation, you should specify the number of pickup motions before performing placement motions as [Quantity] instead of specifying the number of parts as it. For example, when using a hand that holds six parts is used and the parts are arranged by three parts, specify 2 as [Quantity] instead of 6. In addition, even though the arranged parts are picked at one shot, you should disable [Pick or Drop Multiple].

For "Place" conveyor station operation or fixed station operation, you should specify the same number as that of "Pick" conveyor station operation as [Quantity]. If [Quantity] is 2 or greater and a robot places all the parts held by the gripper at one shot, enable [Pick or Drop Multiple]. Even though [Quantity] is 2 or greater, if a robot places parts at multiple shots per arranged parts on the infeed conveyor, disable [Pick or Drop Multiple].

6.16.6 Setup of a Tray

This section shows how to set up a virtual tray.

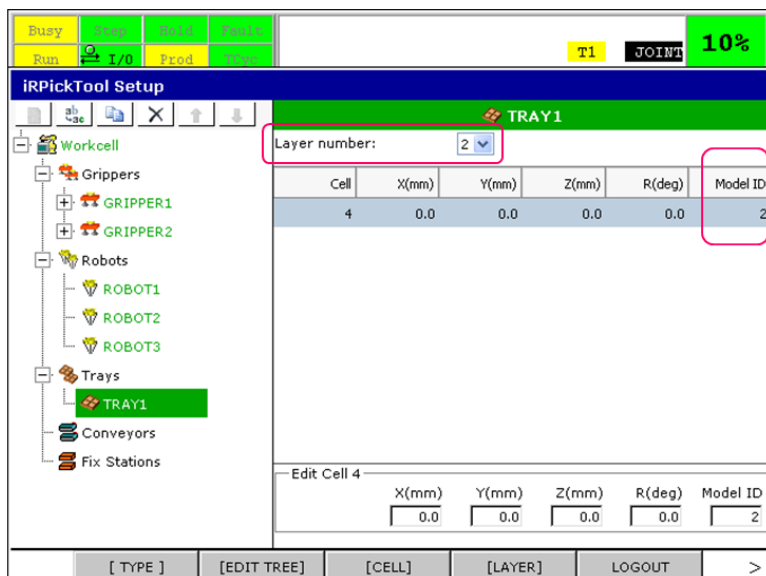
- The origin point of a tray frame can be set as arbitrary positions, when treating the part of a [Circle (without Phase)]. For example, it is set up so that X of the cell 1 and Y may be set to 0.
- When treating the part of a [Circle (without Phase)], the direction of X of a tray frame is automatically 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.

- 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.

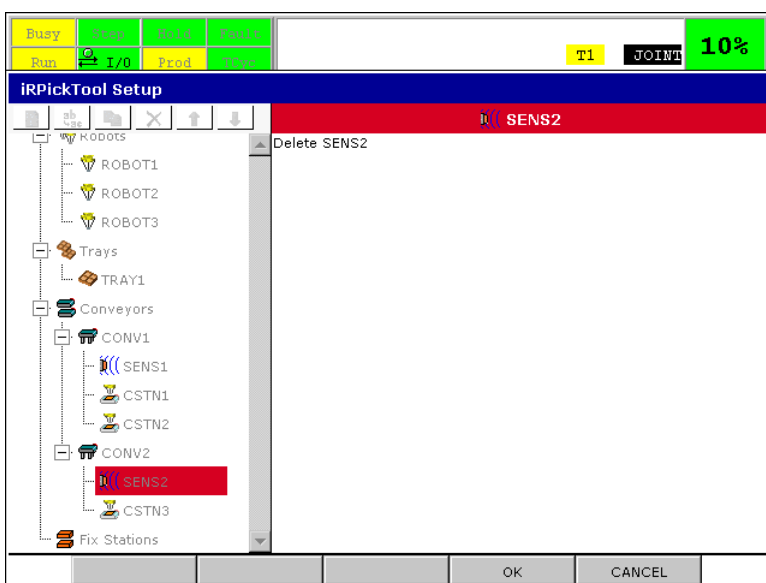


6.16.7 Setup of a Conveyor

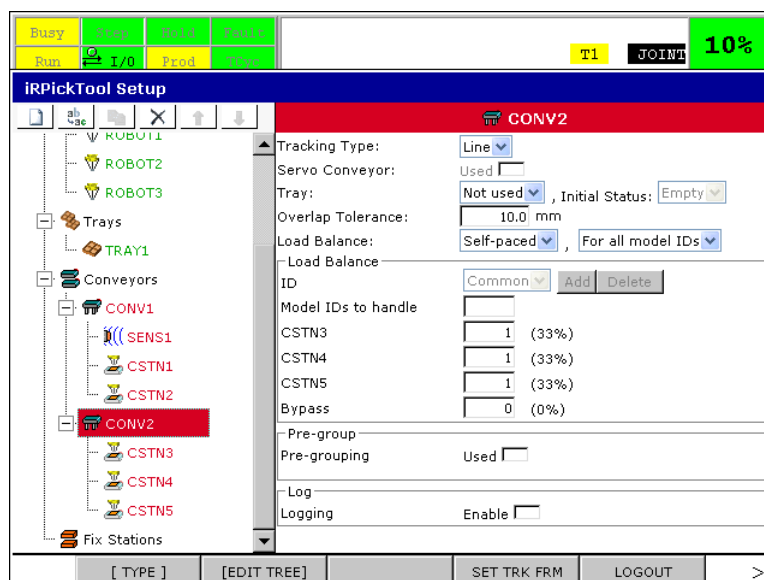
In pre-grouping you always setup two conveyors: the infeed conveyor and the pre-grouping conveyor. In the real-world, both the conveyors are the same physical conveyors. These two conveyors use the same tracking frame. However, here is no need to teach the tracking frame for each conveyor. It is sufficient to teach it for the pre-grouping conveyor.

First, the object of an infeed conveyor is set up. Refer to 6.6 "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.

Next, 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.



Then, add a required number of conveyor station objects.

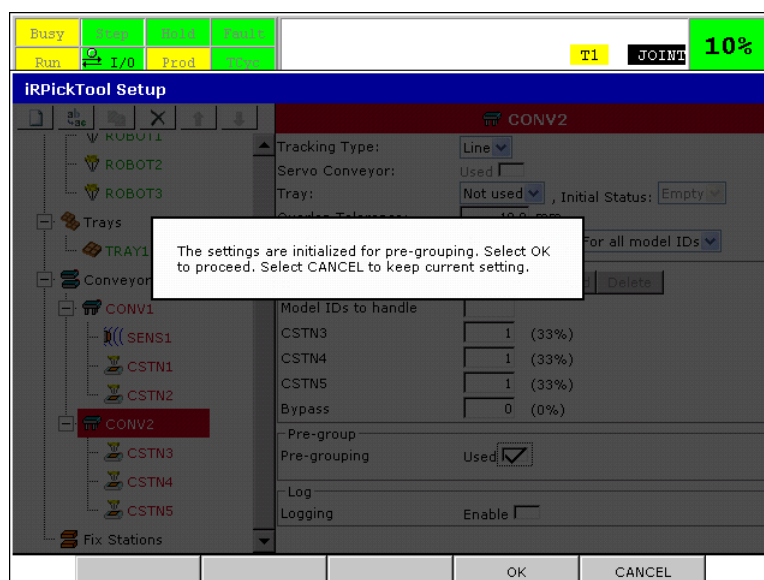


Check the [Used] checkbox of the [pre-grouping] in the [Pre-group] item.

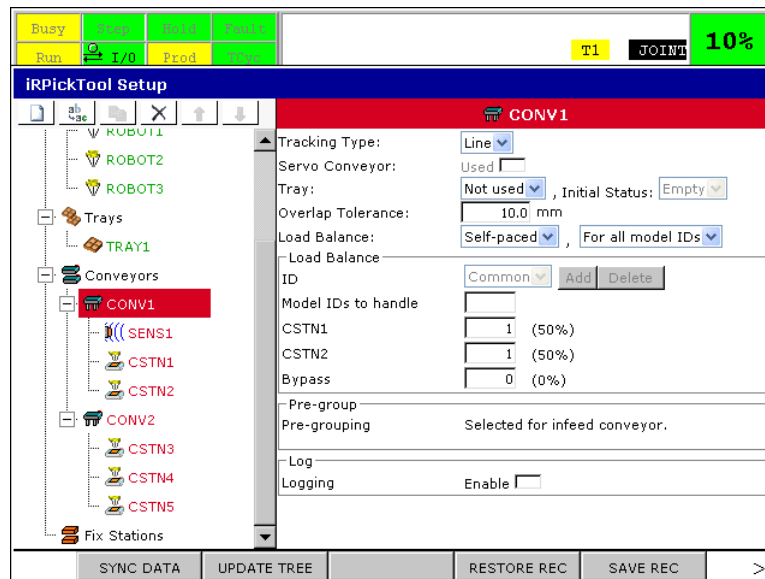
NOTE

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.



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.

Set up other items than a [Pre-grouping] according to 6.6 "SETUP OF A CONVEYOR".

Pre-group

Pre-group	
Pre-grouping	Used ☒
Infeed Conveyor	Please select
Max. of Belt Y	0.000 mm
Min. of Belt Y	0.000 mm
Virtual Tray	Please select
Part Type	Please select

Infeed Conveyor

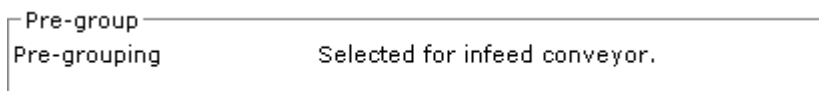
Select the infeed conveyor object.

NOTE

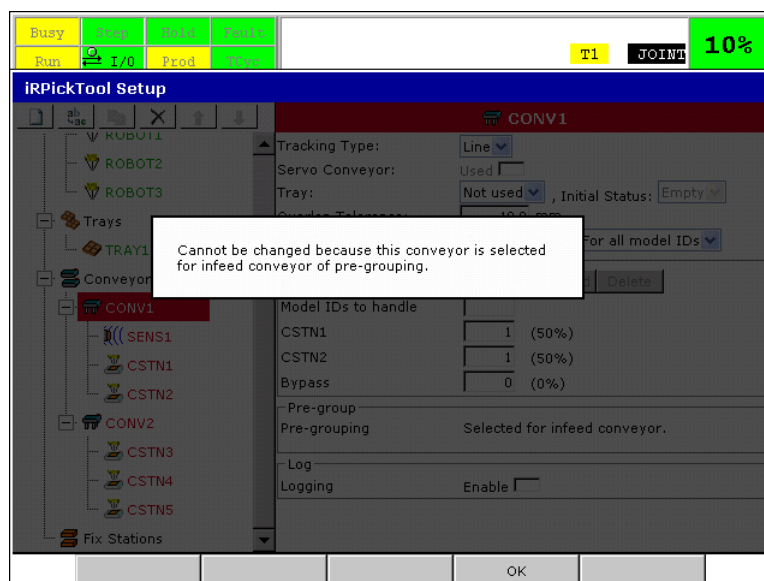
Only the conveyor objects, which meet the following conditions, are displayed as options of the [Infeed Conveyor].

- 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 6.6.1 "Setup of Tracking Frame" in detail. More concretely, follow the procedure after item 3 mentioned in next 6.6.1.1 "Line Conveyor".

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.

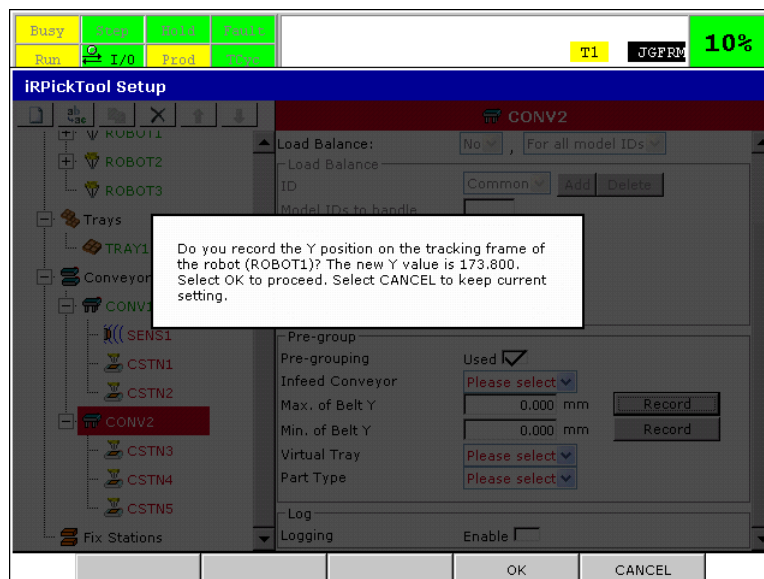
Select the tool frame set up by "3.4 SETTING OF THE TCP" as a jog frame and move the TCP by jog and tap the [Record] button, then the Y value in a tracking frame will be set. For example, touch up the end of a left-hand side of the conveyor belt to the direction of movement of a conveyor, in order to 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 iRPickTool treeview.

After you tap of the [Record] button, the Y value of a current robot position in the tracking frame will be displayed.



Press F4 [OK] key to set the new value.

Press F5 [CANCEL] key to keep the previous value.

NOTE

If you press the [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

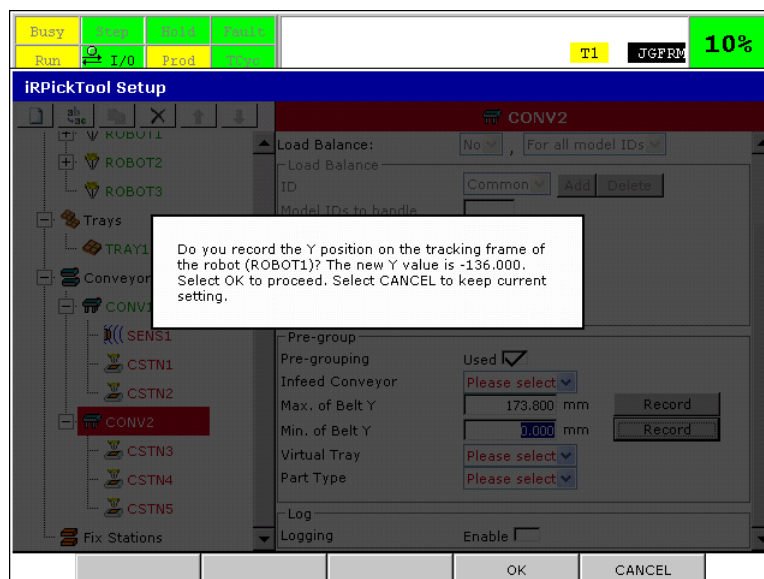
A virtual tray is generated so that not the model origin point of the vision but whole parts may not overrun this lower limit.

Select the tool frame set up by 3.4 "SETTING OF THE TCP" 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 iRPickTool treeview.

After you tap of the [Record] button, the Y value of a current robot position in the tracking frame will be displayed.



Press F4 [OK] key to set the new value.

Press F5 [CANCEL] key to keep the previous value.

NOTE

If you press the [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)],

Pre-group		
Pre-grouping	Used ☒	
Infeed Conveyor	CONV1	
Max. of Belt Y	173.800 mm	Record
Min. of Belt Y	-136.000 mm	Record
Virtual Tray	TRAY1	
Part Type	Circle (without Phase)	
Part Parameter	Radius 0.0 mm	
Orientation	Enable ☐	
	Minimum -90.0 °, Maximum 90.0 °	
Interval of Angle	Auto, 30.0 °	

Part Parameter

Set the radius of a circle to the [Radius].

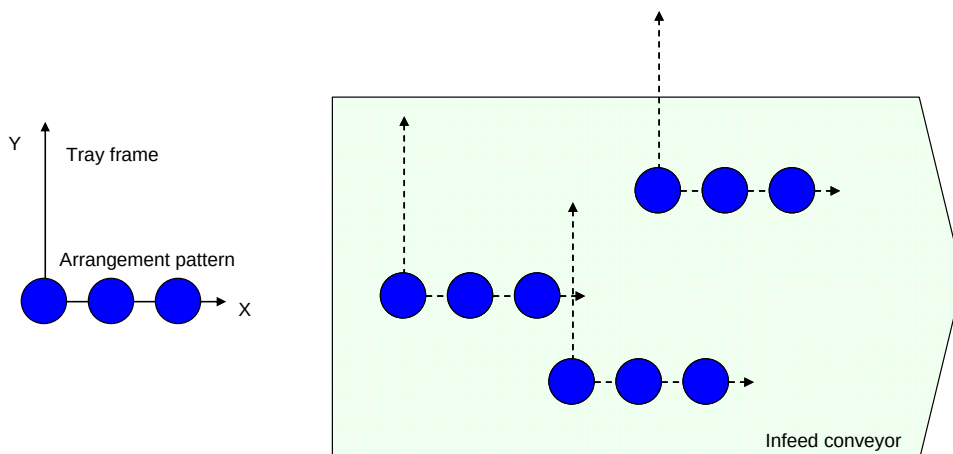


CAUTION

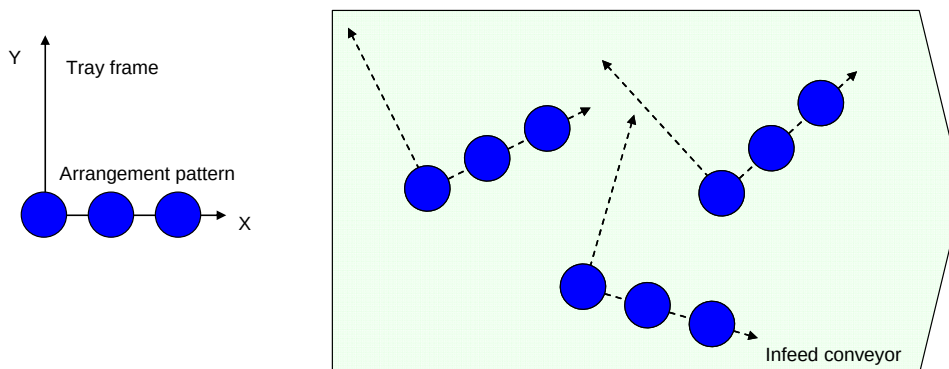
If the radius value is smaller than the actual one, the part which the robot picks and places for arrangement may collide with the part on a conveyor due to the incorrect calculation of the interference check between both kinds of the parts. When the form of the actual part is not the circle, you must set the radius of the circle which includes whole of the part.

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.



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].



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],

Pre-group	Used ☒	
Pre-grouping	CONV1	
Infeed Conveyor		
Max. of Belt Y	173.800 mm	Record
Min. of Belt Y	-136.000 mm	Record
Virtual Tray	TRAY1	
Part Type	Rectangle	
Part Parameter	X length 0.0 mm	
	Y length 0.0 mm	

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 uncorrect 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.

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 6.6 "SETUP OF A CONVEYOR" for the other items.

6.16.8 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 6.7 "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 6.7.1 "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 uncorrect calculation of the interference check between both kinds of parts. When you refer to 6.7.3.1 "When a tray is not used" under 6.7.3 "Setting the Sensor Position" and 6.7.4.5 "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 uncorrect 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 uncorrect calculation of the interference check between both kinds of the parts.

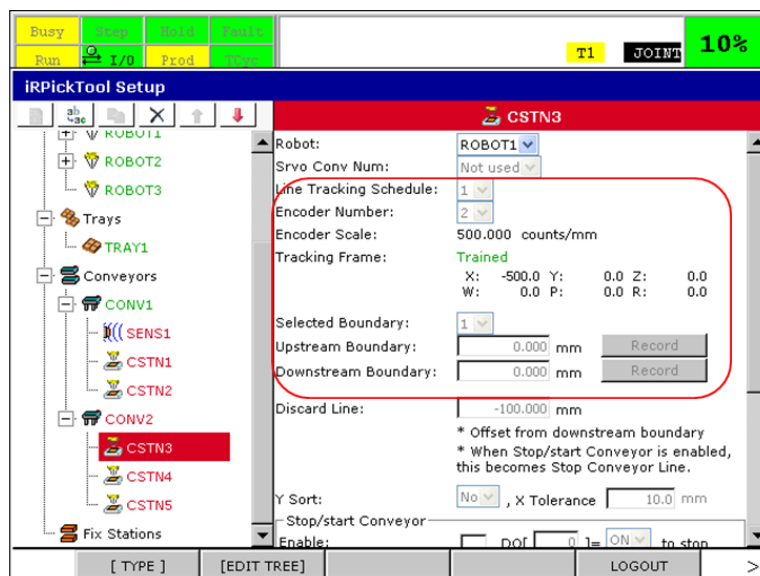
Additionally, if the position is not detected by the vision and the [Rectangle] is selected as the [Part Type] at the pre-grouping conveyor, you should give attention to the order of settings. Setting of the sensor position or the target position must be performed after setting of the [Virtual Tray Position] 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.

6.16.9 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 6.8 "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.

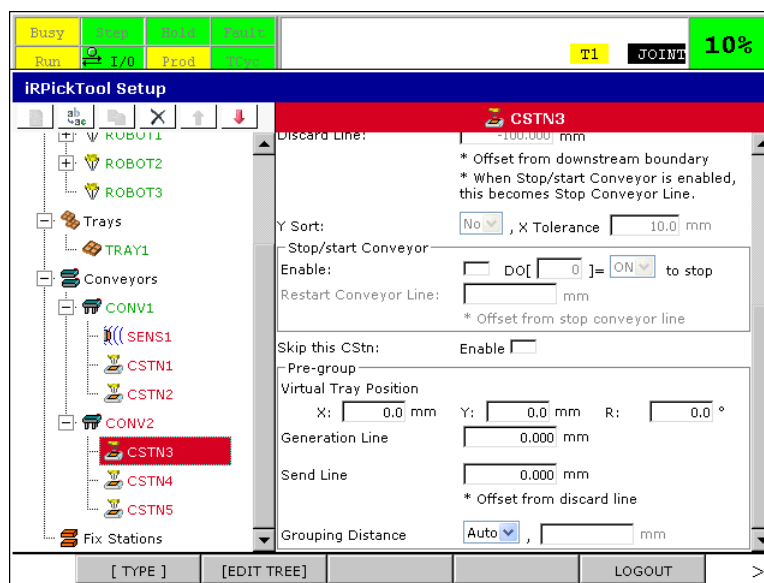
Skip Station

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 6.7.5 “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 6.7.3.2 "When a tray is used" under "6.7.3 Setting the Sensor Position". When the [Distance] is set as the [Trigger Action], set the target position. Refer to 6.7.6.2 "When a tray is used" under "6.7.6 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 wizard 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.

Generation 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 length] of [Part Parameter] and the value(y) set at [Y length] of [Part Parameter] as shown in Fig. 6.16 (b).

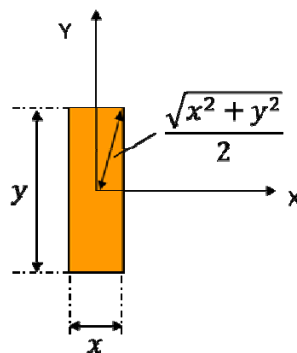
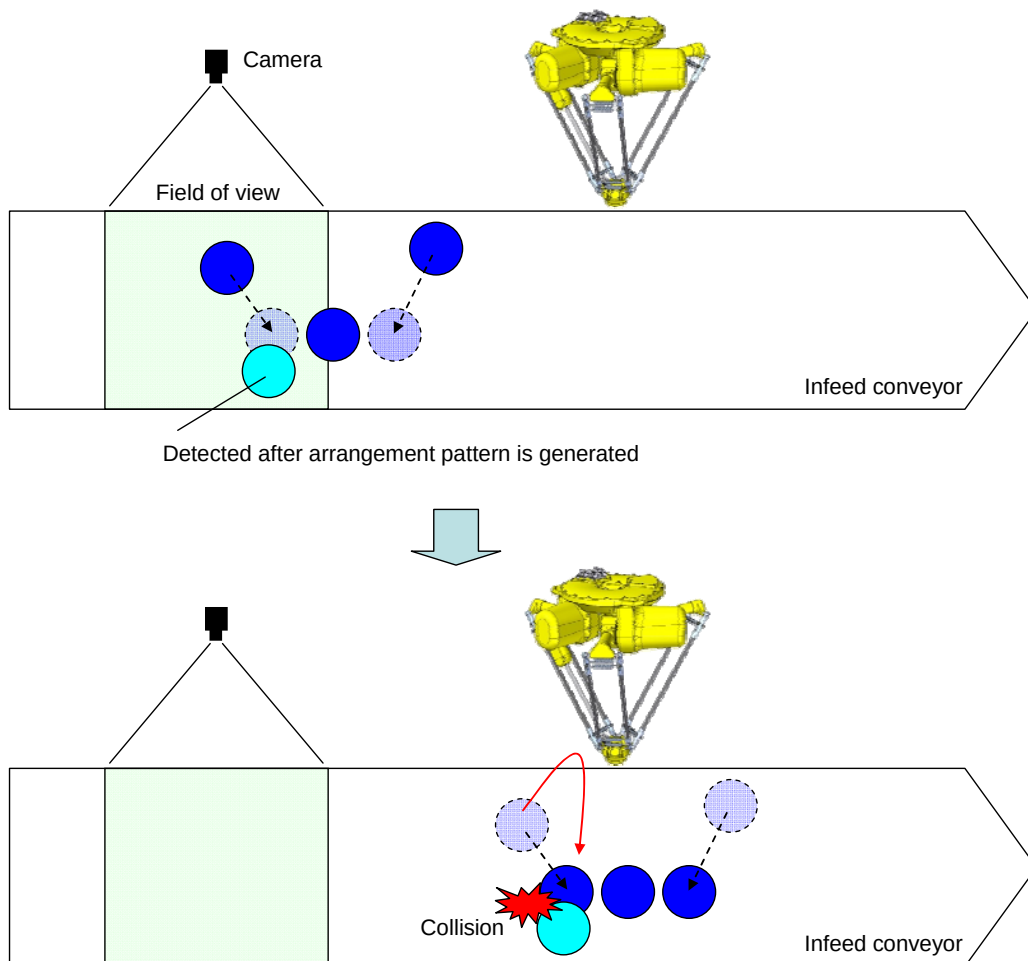


Fig. 6.16 (b)

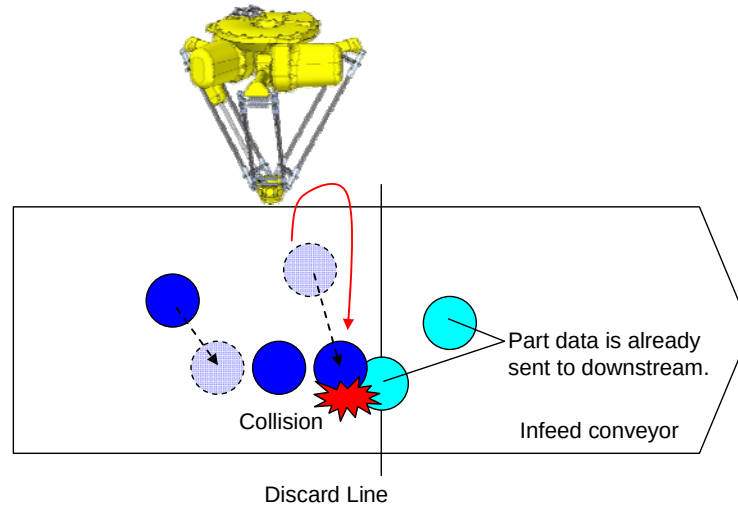
⚠ CAUTION

- 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.

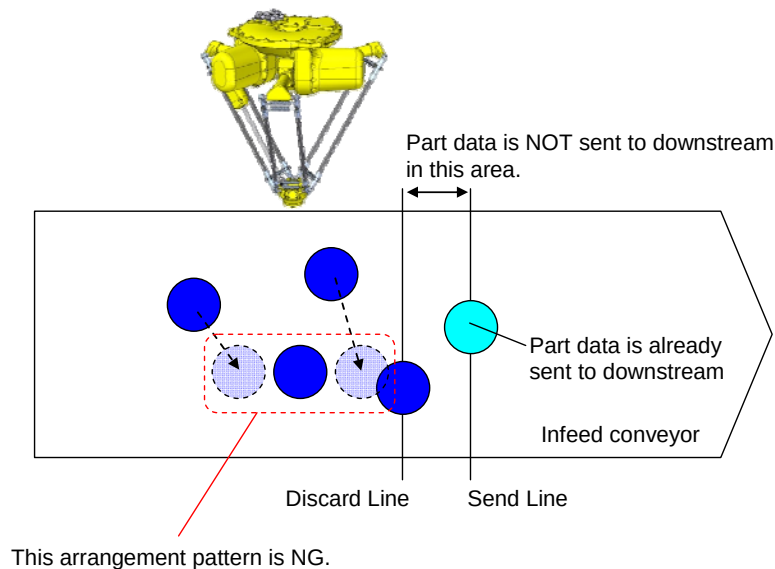


Send Line

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.



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.



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. 6.16 (b).



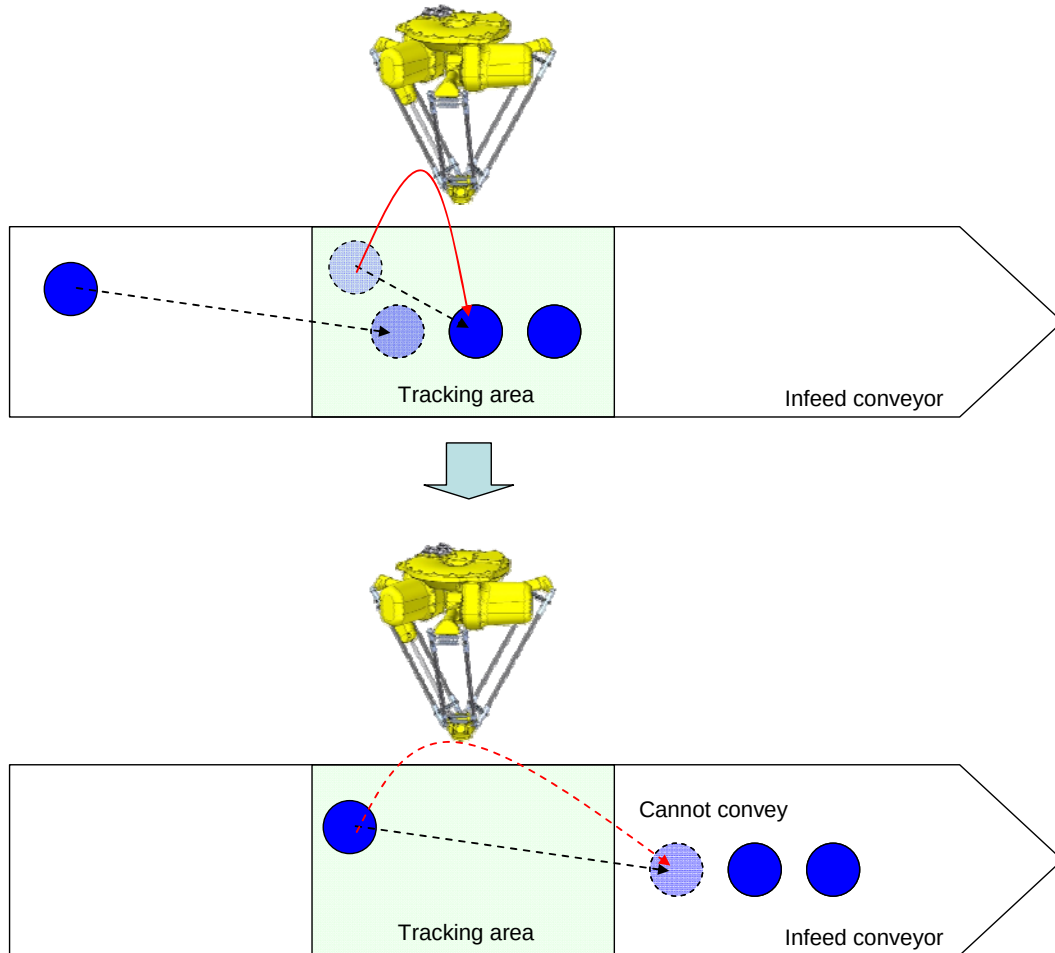
CAUTION

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.

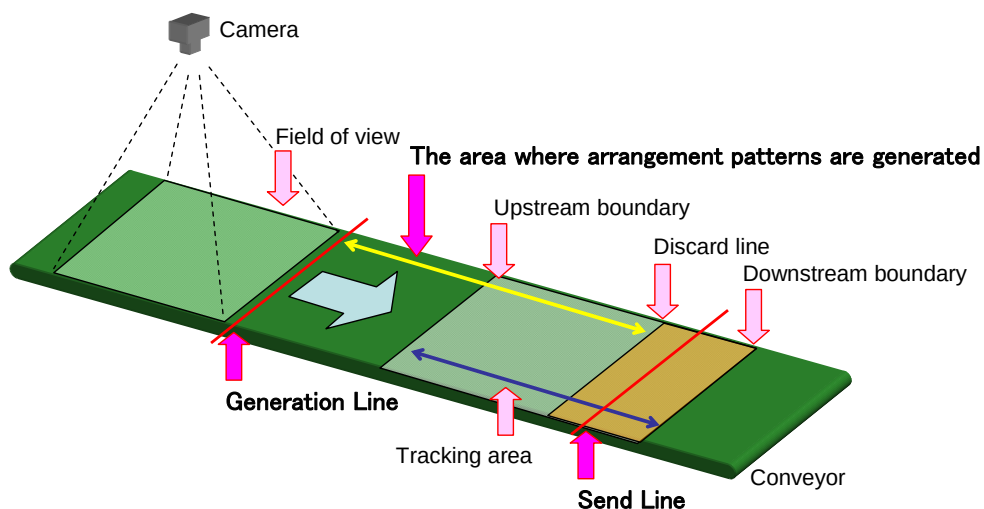
If parts which constitutes a virtual tray separates too much each other about the direction of movement of a conveyor, even if the robot picks up a part for arrangement, the corresponding placement position can already pass a tracking area and the robot cannot place the part.



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"



6.16.10 Setup of a Fixed Station

When you use a fixed station, refer to 6.9 "SETUP OF A FIXED STATION".

6.16.11 Robot Programs

The points that need to be taught during the entire flow are the origin and pickup position only. Since the same position data is used, it is unnecessary to teach the placement position. Refer to 6.10 "ROBOT PROGRAMS".

6.16.12 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 6.11 "REFERENCE POSITION SETTING".

6.16.13 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 6.12 "TEACHING THE POSITION TO ROBOTS".

6.16.14 Fine Adjustment of The Tracking Motion

Refer to 6.13 "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.

6.16.14.1 Adjustment of the tracking frame error

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 6.13.2 "Adjustment of the Tracking Frame Error" for details.

6.16.15 KAREL Programs

This subsection explains the KAREL programs for a pre-grouping function. All of them are related to a conveyor station.

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.

6.16.15.1 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 6.14.6.1 "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 belong 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)
```

NOTE

If you specify Argument 3 (model ID), the argument guidance is not displayed.

6.16.15.2 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 6.14.6.1 "PKCSGETID.PC". You can specify only the conveyor station of the pre-grouping conveyor. When you specify the conveyor station which does not belong 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)
```

6.16.15.3 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 6.14.6.1 "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)

6.16.15.4 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 6.14.6.1 "PKCSGETID.PC". You can specify only the conveyor station of the pre-grouping conveyor. When you specify the conveyor station which does not belong 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"

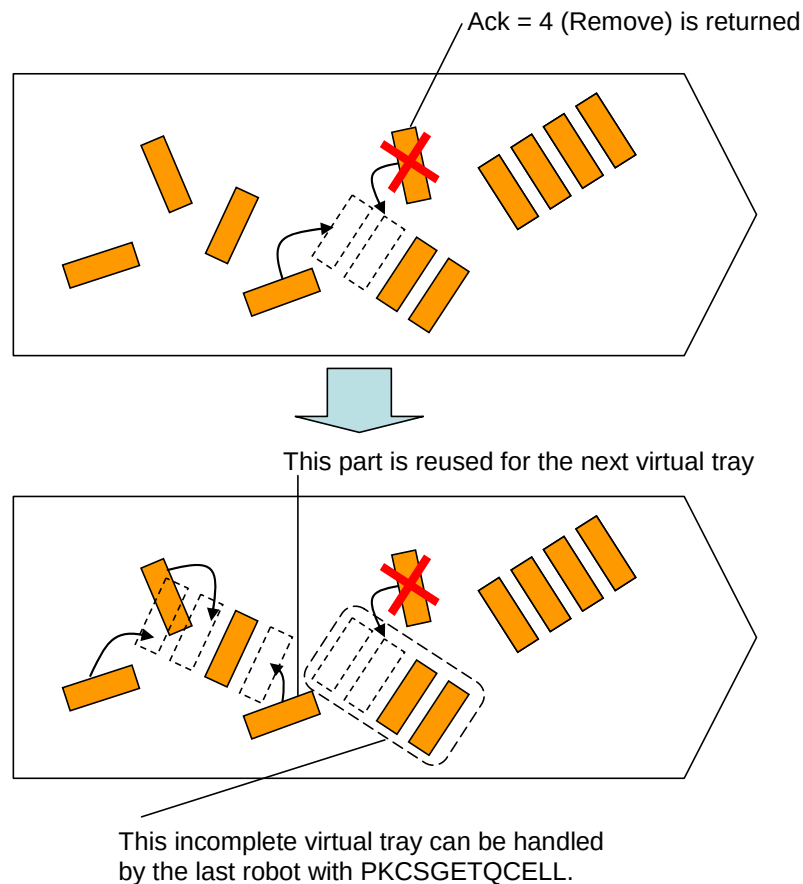
Specify this value when the part is returned to the conveyor station before being handled.

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.

**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)
```

6.16.16 Standard TP programs

This section describes the standard TP programs of pre-grouping.

They work depending on the type of pre-grouping defined as follows.

- 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.

6.16.16.1 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]

```

6.16.16.2 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 Cvld],Pk CStn ID=R[110:Pk1x
   StnId],Dp Conv ID=R[129:Dp1x Cvld],Dp CStn ID=R[130:Dp1x StnId],PreGrp Type
   Reg=AR[1])
8: LBL[10000:End]
[End]

```

6.16.16.3 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.

Gets a part from the part queue using PKCSGETQPRGR

Regardless of getting or not getting a part from the queue, if the property “Wait Position” is enabled in the iRPickTool 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.

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.

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.

If PKCSGETQPRGR returned a part,
 Sets a tracking trigger for the pickup motions using PKCSTRGPRGR
 If PKCSTRGPRGR did not set a tracking trigger, it makes the request to set the tracking trigger again.
 Moves the robot to the approach position.
 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.
 If the move to the pick position succeeded, then it gets the pick delay using the PKGRGETPKDLY.PC macro.
 Sets the payload of the robot to “Full gripper” or Payload[2].
 Moves the robot to the retreat position.
 Sets a tracking trigger for the placement motions using PKCSTRGPRGR
 If PKCSTRGPRGR did not set a tracking trigger, it makes the request to set the tracking trigger again.
 Moves the robot to the approach position.
 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.
 If the move to the place position succeeded, then it gets the place delay using the PKGRGETDPDLY.PC macro.
 Sets the payload of the robot to “Empty gripper” or Payload[1].
 Moves the robot to the retreat position.
 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]

```

6.16.16.4 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.

Sets the skip outbound FLG to OFF before beginning the pick moves.

Gets the number of cells in a tray using PKTRGETNUMCL.

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.

Regardless of getting or not getting a part from the queue, if the property "Wait Position" is enabled in the iRPickTool 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 PKSCALWPOS.PC.

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.

If PKCSGETQCELL did not return a part, it makes the request to get the part again.

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.

Sets the UTOOL to the gripper's first zone's UTOOL.

Moves the robot to the approach position.

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.

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.

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.

If the move to the pick position succeeded, then it gets the pick delay using the PKGRGETPKDLY.PC macro.

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.

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.

If the part presence check succeeds, it sets the payload of the robot to "Full gripper" or Payload[2].

It moves the robot to the retreat position.

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 OTHER FUNCTIONS

This chapter describes the functions that are useful in the *iRPickTool* tracking system. These functions can be used to construct a more stable and flexible system.

7.1 EXCHANGE WORKCELL WITH USING RECIPE

This chapter describes the setup of the Recipes.

A Recipe is a collection of *iRPickTool* 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}

Recipes can be used for the following purposes:

- To change from one product to another
- To run with a different set of gripper, tray, robot, sensor, Conveyor station, Fixed Station or quotas for the same product.

iRPickTool 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. *iRPickTool* will then prepare the workcell to run the new recipe as the active recipe.

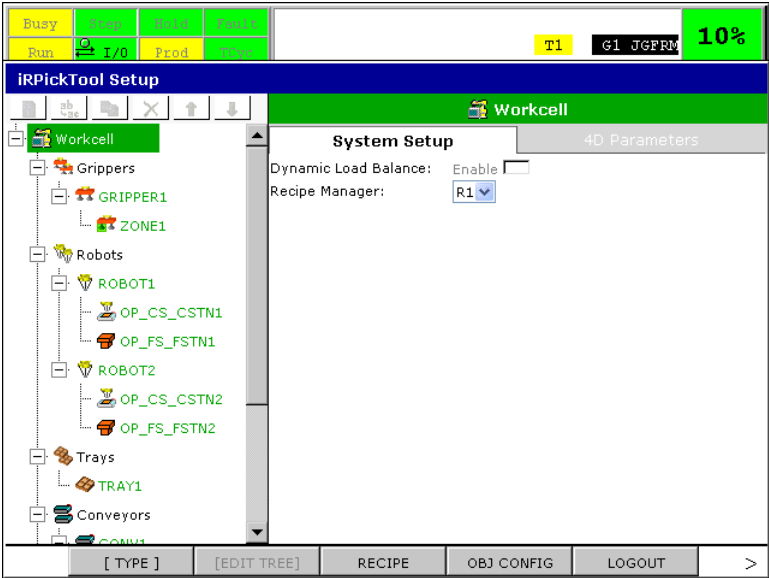
1. Recipes may be changed by a PLC.
2. Recipes may be exported to a memory card or a USB device.
3. Recipes may be imported from a memory card or a USB device.

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”. However, Recipes may be moved from one controller to another.

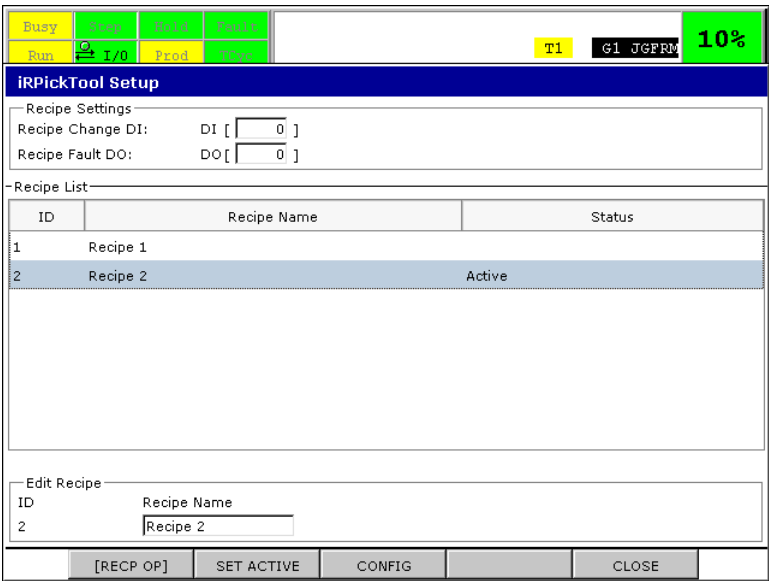
Recipes are backed up into IRPICKTOOL.ZIP files when you do a File Backup All from the File menu on the pendant. The *iRPickTool.zip* file can also be restored to the controller. Its extracted contents will go the FR:\IRPICKTOOL\ directory.

7.1.1 Setting a Recipe

Recipe setup is performed in the Workcell property page. 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.



When you click the RECIPE function key from the Workcell menu, you will see the Recipe Menu as shown in the figure below. You will see the names of the recipes listed in the Recipe List listview.



Recipe Change DI
When this DI is turned ON by external PLC, Recipe will be changed. See “7.1.3 PLC Recipe Changeovers” in detail.

Recipe Fault DO
If the recipe change fails due to some reason, this DO turns ON.

Recipe Name
You can change the name of recipe which is shown here. See “Renaming a recipe” in detail.

7.1.2 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 *iRPickTool* 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 *iRPickTool* such as the Perch Position, Reference Position, Approach position, etc.

- 1 Click on F1 [RECP OP] key in the Recipe page. See figure below. Select [1 CREATE] from the pull-up.

The screenshot shows the **iRPickTool Setup** window. At the top, there are status buttons: Busy, Stop, Run, Fault, and a progress bar showing 10%. Below this is the **Recipe Settings** section with fields for Recipe Change DI and Recipe Fault DO. The **Recipe List** section contains a table with columns ID, Recipe Name, and Status. A pull-up menu is open over the table, showing options: 1. CREATE, 2. SAVE, 3. DELETE, 4. EXPORT, 5. IMPORT, and 6. RESTORE. The 'Edit Recipe ID' field shows '1' and the 'Name' field is empty.

ID	Recipe Name	Status
1	Recipe 1	Active

- 2 A new recipe with a default name of RecipeX where x is the recipe id is immediately created. You may see the screen momentarily display “Now creating Recipe”. When the recipe is successfully created, you will see a confirmation prompt (not shown). The newly created recipe contains all the current *iRPickTool* data you have setup using the treeview. The Recipe is shown in the Recipe List window as shown in the following figure.

Busy	Stop	Hold	Fault			T1	G1 JGPRM	10%
Run	Q I/O	Prod	Stop					

iRPickTool Setup

Recipe Settings

Recipe Change DI: DI [0]
Recipe Fault DO: DO [0]

Recipe List

ID	Recipe Name	Status
1	Recipe 1	
2	Recipe 2	Active

Edit Recipe

ID	Recipe Name
2	Recipe 2

[RECP OP]

SET ACTIVE

CONFIG

CLOSE

Renaming a Recipe

Rename a Recipe.

- Click on the Recipe you wish to rename in the Recipe list window. Immediately, the current name of the recipe is updated in the Edit Recipe window at the bottom. Click on the textbox containing the current name and change it. When you press Enter, the recipe is renamed and you will see the new name of the Recipe in the Recipe List window. 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 may not begin with a numeric character or contain a “-” (minus sign).

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\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.

- Click on the Recipe you wish to delete in the Recipe list window. The Recipe row will be highlighted.
- Click on FI [RECP OP]. You will see a pull-up containing the DELETE choice. Select [3 DELETE].

- 3 The screen will become dark. You will get a confirmation prompt as shown below. Click F4 [OK] key.

The screenshot shows the iRPickTool Setup screen. At the top, there are status buttons: Busy (yellow), Stop (green), Hold (green), Fault (green), Run (yellow), I/O (green), Prod (yellow), and Stop (green). To the right, it displays 'TEST LINE 0 T1 ABORTED G1 JGFRM' and a green progress bar at 10%. The main area is titled 'iRPickTool Setup' and contains sections for 'Recipe Settings' (Recipe Change DI: DI [0], Recipe Fault DO: DO [0]), 'Recipe List' (a table with columns ID, Recipe Name, and Status), and 'Edit Recipe' (ID: 2, Recipe Name: Recipe 2). A white dialog box is overlaid on the screen with the text 'Delete Recipe Recipe 2 ?'. At the bottom, there are 'OK' and 'CANCEL' buttons.

- 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. Click F4 [OK] key.

The screenshot shows the iRPickTool Setup screen, similar to the previous one. The 'Recipe List' table now shows 'Recipe 2' as deleted. A white dialog box is overlaid on the screen with the text 'Recipe Recipe 2 deleted'. The 'Edit Recipe' section still shows ID: 1 and Recipe Name: Recipe 2. At the bottom, there are 'OK' and 'CANCEL' buttons.

Saving a Recipe

Save a Recipe.

When you save a recipe, you save all the current iRPickTool 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 iRPickTool 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.

**CAUTION**

When Plug & Play is used, IRPK-084 “Recipe operation time out.” occurs if position register for perch position or reference position is uninitialized. In this case, turn the power to the robot controller off, then on again and save a recipe after teaching the positions.

- 1 Click on F1 [RECP OP] key. You will see a pull-up containing the SAVE choice. Select [2 SAVE].
- 2 You will be asked to confirm as shown in the figure below. Click F4 [OK] key.

The screenshot shows the **iRPickTool Setup** screen. At the top, there are status indicators: **Busy**, **Stop**, **Ready**, **Teach**, **Run**, **I/O**, **Prod**, **Stop**. The main display area shows **TEST LINE 0 T1 ABORTED G1 JGFRM** and a green bar with **10%**. Below this, the **iRPickTool Setup** window is open. It has sections for **Recipe Settings** (Recipe Change DI: DI [0], Recipe Fault DO: DO [0]), **Recipe List** (ID 1, Recipe 1, Status), and **Edit Recipe** (ID 1, Recipe Name Recipe 1). A white dialog box is centered on the screen with the text: "Save the current setup to the recipe(Recipe 1). Are you sure?". At the bottom of the setup window are **OK** and **CANCEL** buttons.

- 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. Click F4 [OK] key.

The screenshot shows the **iRPickTool Setup** screen, similar to the previous one. The status indicators and main display area are the same. The **iRPickTool Setup** window is open. A white dialog box is centered on the screen with the text: "Recipe Recipe 1 saved". At the bottom of the setup window are **OK** and **CANCEL** buttons.

Restoring a Recipe

Restore a Recipe.

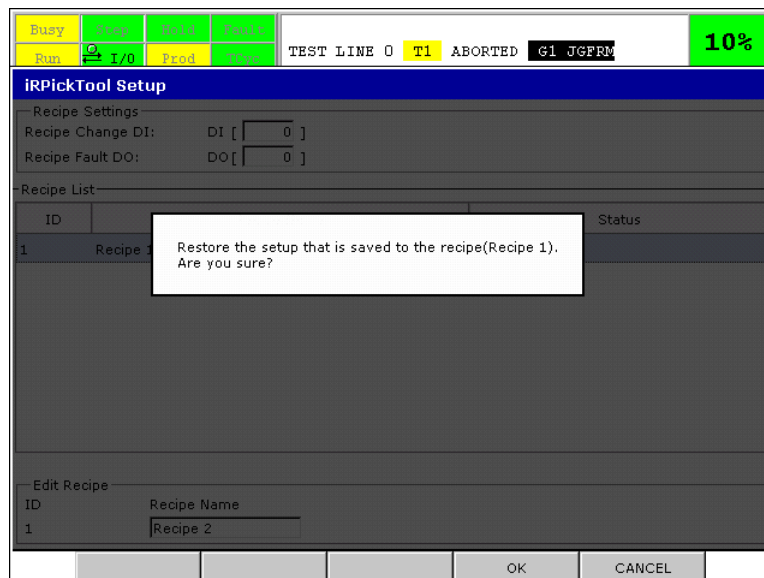
When you restore a recipe, you load the last saved Active Recipe data to the memory to make it current. You do the Restore operation when you have made some changes to the *iRPickTool* data since you since you last saved the recipe but wish to get rid of those changes and go back to the last saved version. 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 *iRPickTool* data is completely restored.

- 1 Click on F1 [RECP OP] key. You will see a pull-up containing the RESTORE choice. Select [6 RESTORE].
- 2 You will be asked to confirm as shown in the figure below. Click 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. Click F4 [OK] key.

Busy	Stop	Hold	Fault	IRPK-261 Recipe Recipe 1 loaded	10%
Run	Start	Prod	Reset	TEST LINE 0 T1 ABORTED G1 JGFRM	

iRPickTool Setup

Recipe Settings
 Recipe Change DI: DI [0]
 Recipe Fault DO: DO [0]

Recipe List

ID	Recipe	Status
1	Recipe 1	Recipe Recipe 1 loaded

Edit Recipe
 ID: 1
 Recipe Name: Recipe 1

OK

**CAUTION**

A recipe that is saved in a controller with the newer version of software must not be restored into one with an older version of software. It may be restored incorrectly.

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 click on it to display the pull-up. Select “Set Device” and then choose your default device such as “MC” or “UD1” or “UT1:”.

- 1 Click on F1 [RECP OP] key. in the Recipe page. Select [4 EXPORT] from the pull-up menu.
- 2 You will be asked to confirm as shown in the figure below. Click F4 [OK] key.

Busy	Stop	Hold	Fault	TEST LINE 0 T1 ABORTED G1 JGFRM	10%
Run	Start	Prod	Reset		

iRPickTool Setup

Recipe Settings
 Recipe Change DI: DI [0]
 Recipe Fault DO: DO [0]

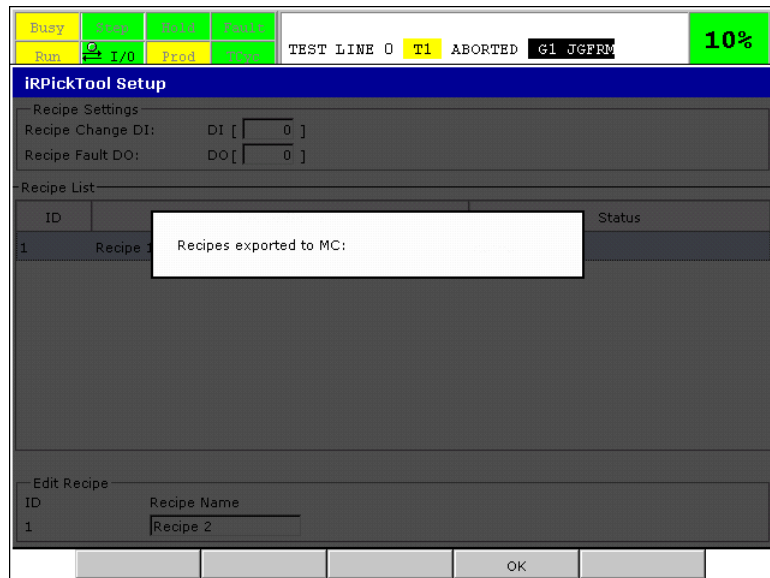
Recipe List

ID	Recipe	Status
1	Recipe 1	Export Recipes to MC: ?

Edit Recipe
 ID: 1
 Recipe Name: Recipe 2

OK CANCEL

- 3 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. Click F4 [OK] key.

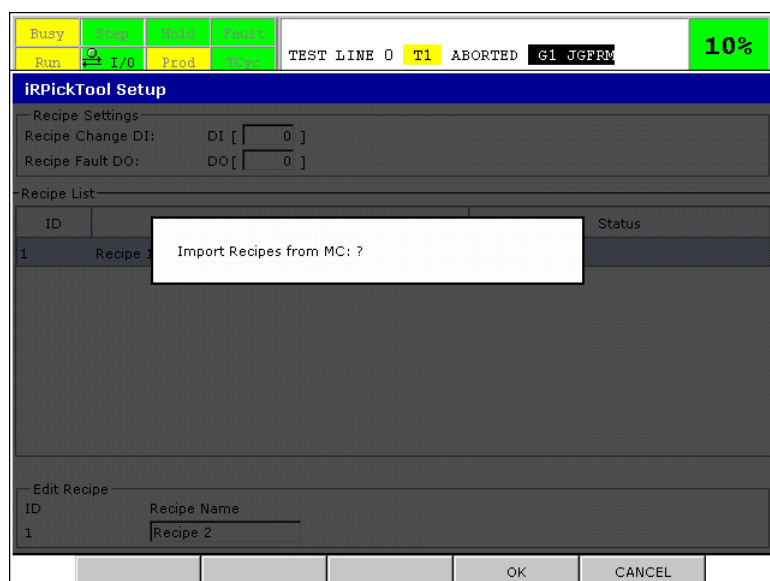


Importing Recipes

Import Recipes.

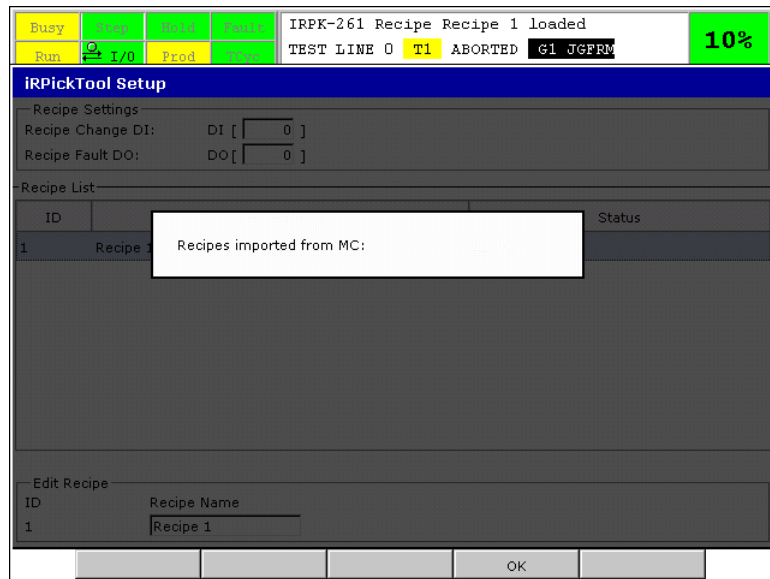
When you Import Recipes you copy the recipes stored in an external storage device such as the memory card or the USB stick into the controller's FR:\IRPICKTOOL\ directory. The device you are importing from is the same as \$DEVICE. You can change the default device from the File menu – find the [UTIL] function key and click on it to display the pull-up. Select “Set Device” and then choose your default device such as “MC” or “UD1” or “UT1:”.

- 1 Click on F1 [RECP OP] key in the Recipe page. Select [5 IMPORT] from the pull-up menu.
- 2 You will be asked to confirm as shown in the figure below. Click F4 [OK] key.



- 3 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. Click F4 [OK] key.

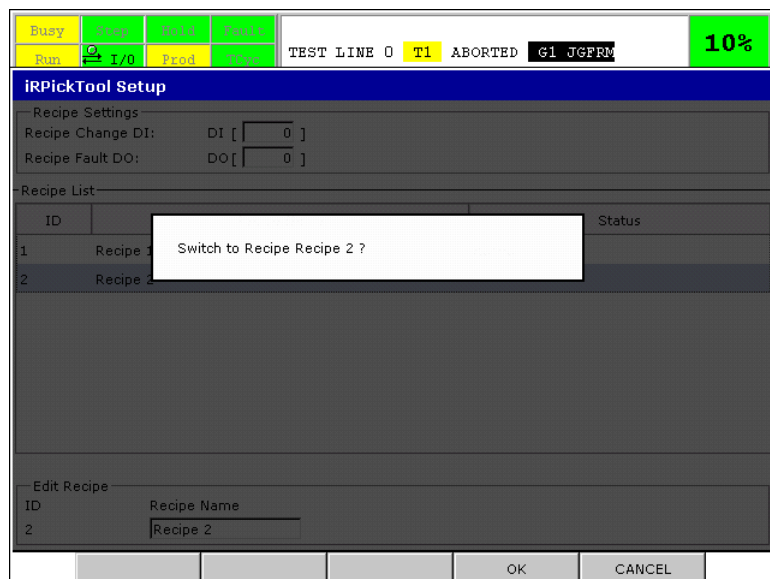


Switching Recipes

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 Click on the Recipe you wish to switch to in the Recipe list window. The Recipe row will be highlighted.
- 2 Use F2 [SET ACTIVE] key.
- 3 The screen will become dark. You will get a confirmation prompt as shown below. Click F4 [OK] key.



- 4 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. Click F4 [OK] key.

Busy	Stop	Hold	Fault	IRPK-261 Recipe Recipe 2 loaded	10%
Run	0 I/O	Prod	Prog	TEST LINE 0 T1 ABORTED G1 JGPRM	

iRPickTool Setup		
Recipe Settings		
Recipe Change DI:	DI [0]	
Recipe Fault DO:	DO [0]	
Recipe List		
ID	Recipe Name	Status
1	Recipe 1	
2	Recipe 2	Recipe Recipe 2 loaded
Edit Recipe		
ID	Recipe Name	
2	Recipe 2	
OK		

Moving Recipes

Move Recipes.

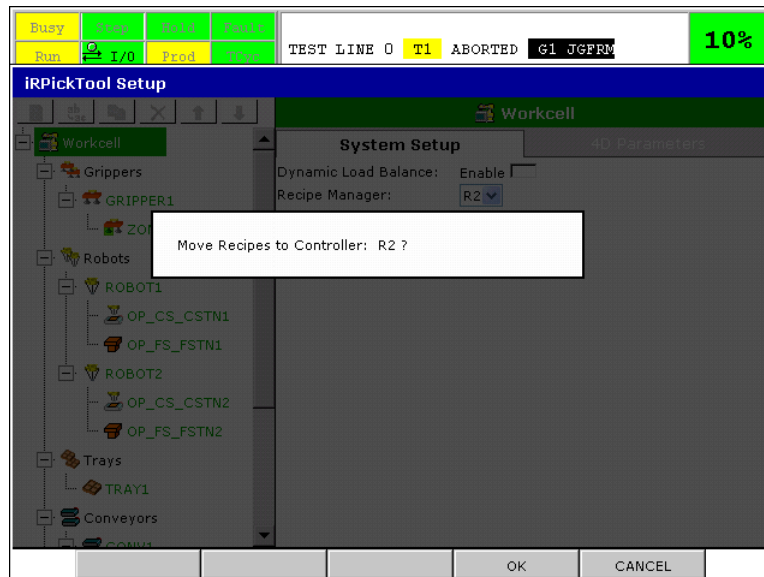
You may sometimes wish to move the recipes from the controller where they are stored to a different controller. You do this because you know that the current controller is going to be powered off for a while because of maintenance needs or other reasons. In this case, from an *iRPickTool*'s standpoint, you need to make another controller the "Recipe Manager" instead of the current controller. You accomplish this in the Workcell Menu.



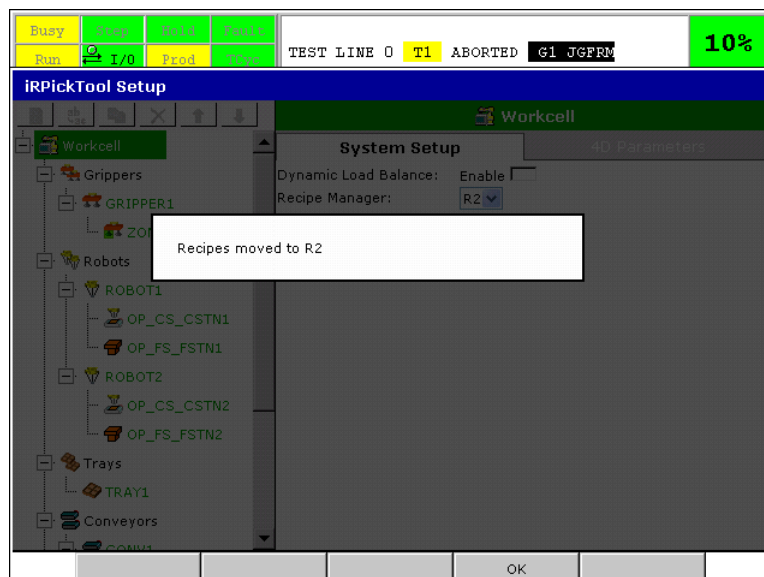
CAUTION

When *iRPickTool* moves recipes to a different controller, it will first move them to the FR:\IRPICKTOOL\ directory in that controller and completely delete the Recipes from the FR:\IRPICKTOOL\ directory of the current controller. Since this operation happens over Ethernet which could go down in the middle of the move, there is the danger that you may lose the Recipes you have so painstakingly saved. Therefore, it is recommended that you first Export the Recipes to an external storage device like the memory card or the USB stick before initiating the Move function. This function is disabled by default. If you need this function, please contact your local FANUC representative.

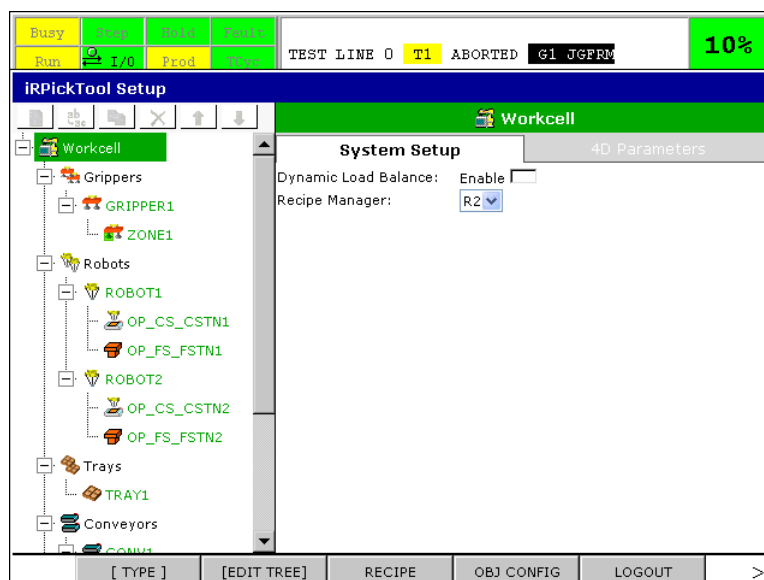
- 1 In the Workcell menu's Recipe Manager property, select a different controller to move all the Recipes to. You will see a prompt as shown below. Click F4 [OK] key.



- 2 You will see the screen display “Now Moving Recipes” if it takes some time to Move the Recipes. When all the recipes have been successfully moved to the new recipe manager controller, you will see the confirmation prompt as shown in the figure below. Click F4 [OK] key.



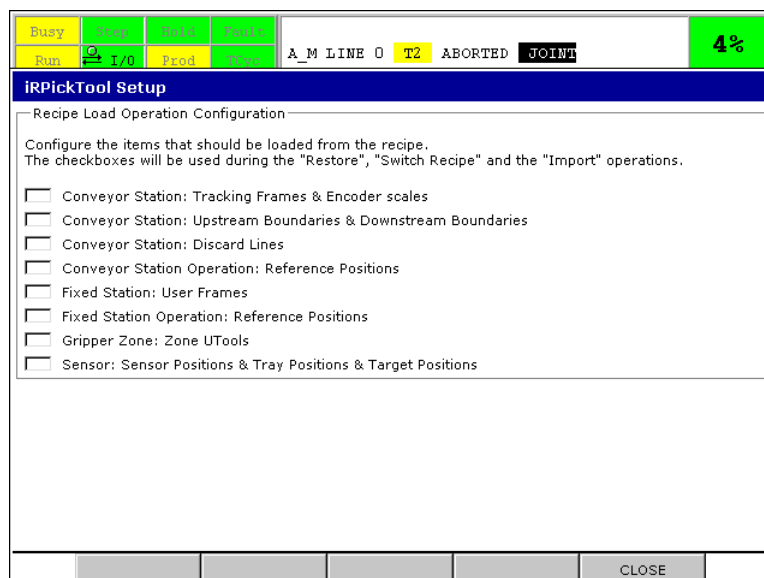
- 3 The Workcell menu will display the new Recipe Manager controller as shown below.



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.

7.1.3 PLC Recipe Changeovers

You can switch Recipes using a PLC. The procedure involves the following steps:

- 1 Set “Recipe Change DI” in recipe setting menu in the recipe manager controller.
- 2 PLC sets a DI signal to high in the Recipe Manager controller.
- 3 When the DI goes high, the controller detects it and runs a PLC handshake program. The handshake program is called PK_PLC_REC_CHG.TP.
- 4 The PKRCSWICH macro that is called in the above TP program allows you to load a specified recipe by its id. The id is provided by a Group Input signal by default in the handshake program.
- 5 If an error occurs during the recipe switching, a fault DO is turned on that the PLC can monitor.

A PLC recipe change will not be allowed if another recipe change is already running in the controller. For example, if you are using the Recipe UI to perform some recipe operation and the PLC signal goes high, the request will not be completed but the fault DO will be turned on.

The PLC recipe change must be performed only via the Recipe Manager controller.

NOTE

It sometime takes 5 and more seconds to complete this PLC recipe changeover.

7.1.3.1 Setting of PLC recipe change data

Set the PLC Recipe Change Data.

Setup the PLC Recipe Change Data

- 1 Display the Recipe menu by clicking on F2 [RECIPE] key from the Workcell menu. The Recipe menu will be displayed as shown below. Notice the Recipe Settings Window at the top.

The screenshot shows the iRPickTool Setup window. At the top, there are status indicators: Busy (yellow), Run (green), I/O (green), Prod (yellow), and a green bar showing T2, G1 JGFRM, and 10%. Below this is the 'iRPickTool Setup' title bar. The 'Recipe Settings' section contains two fields: 'Recipe Change DI:' with a value of 0 and 'Recipe Fault DO:' with a value of 0. Below this is a 'Recipe List' table with columns ID, Recipe Name, and Status. The table contains one entry: ID 1, Recipe Name 'Recipe 1', and Status 'Active'. At the bottom, there is an 'Edit Recipe' section with fields for ID (1) and Recipe Name ('Recipe 1'). The bottom of the window has buttons: [RECP OP], SET ACTIVE, CONFIG, and CLOSE.

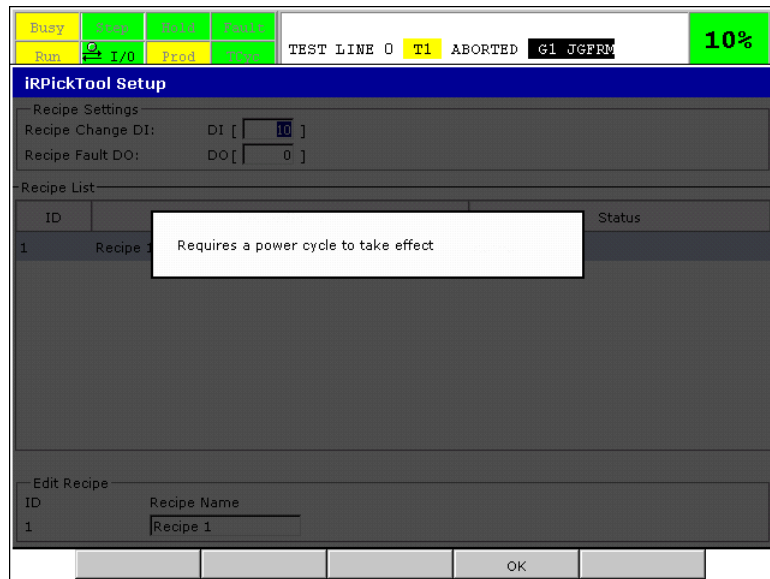
ID	Recipe Name	Status
1	Recipe 1	Active

- 2 In the Recipe Settings window, enter a valid Digital Input for the property “Recipe Change DI” as shown below:

This screenshot is similar to the previous one, but the 'Recipe Change DI' field in the 'Recipe Settings' section now has the value 10. All other elements, including the 'Recipe List' table and the 'Edit Recipe' section, remain the same. The status bar at the top shows T1, G1 JGFRM, and 10%.

ID	Recipe Name	Status
1	Recipe 1	Active

- 3 When you press Enter you will get the message below. Click F4 [OK] key.



- 4 Enter the Recipe Fault DO port number if the PLC wants to know if an error occurred or not.
5 Power cycle the controller as required by step 3.

Modify the PLC handshake program PK_PLC_REC_CHG.TP

The standard PK_PLC_REC_CHG.TP program is shown below. You can modify the program to suit your need. The basic steps are:

- 1 Provide the Recipe id via a group input or other mechanisms
- 2 Call PKRCSWITCH with the recipe id.
- 3 Check the fault DO and if necessary the status returned by PKRCSWITCH.

This program communicates with a PLC to get the recipe ID and then switches the Recipe used by all the robots defined in the iRPickTool treeview (or your workcell).

This program is automatically run by iRPickTool when the “Recipe Change DI” defined in the Recipe menu of the treeview goes high.

This program can only be run when you are not running production – i.e., all the robots are aborted.

This program uses a default handshake with the PLC using Group Inputs and Group Outputs to get the recipe ID. You may change it to other handshake methods.

In this program:

DO[32] is used by the robot to ack the PLC that is ready to receive data.

DI[32] is used to receive the ACK from the PLC that is ready to send the data.

GI[1] is used to receive the recipe ID from the PLC.

GO[1] is used by the robot to send the received recipe ID back to the PLC for verification.

The KAREL macro PKRCSWITCH is used to switch the recipe to the new recipe ID sent by the PLC.

The status register of PKRCSWITCH indicates if the recipe switching was successful.

If you assigned “Recipe Fault DO” in the Recipe menu, then if the status was non-zero, the output would be turned high by PKRCSWITCH macro.

PK_PLC_REC_CHG.TP

```

1:  !-----
2:  ! Communication program between a
3:  ! PLC and iRPickTool to switch
4:  ! Recipes.
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: ! But you must call PKRCSWITCH
12:
13: !-----
14:
15: ! Ready to receive Recipe Id
16: !   Min. = 1, Max. = 99.
17:
18: DO[32]=ON
19: ! Wait for PLC ready to send data
20: WAIT DI[32]=ON
21: R[96:TemporaryUse2]=GI[1]
22: ! Echo Recipe number back to PLC.
23: GO[1]=R[96:TemporaryUse2]
24: ! Signal PLC that data received.
25: DO[32]=OFF
26: ! Wait for PLC acknowledgement.
27: WAIT DI[32]=OFF
28:
29: ! PKRCSWITCH(recipe_id, status) s
30: ! recipes. R[95] will be non-zero
31: ! an error occurred. If you speci
32: ! valid DO in the Recipe UI for F
33: ! Change DO, then it will be turn
34: ! if there is an error.
35:
36: CALL PKRCSWITCH(Recipe ID=R[96:TemporaryUse2],Stat Reg=95)

```

7.1.4 KAREL Program

7.1.4.1 PKRCSWITCH.PC

This program is used to switch recipes. It will run only if all the controllers defined in the tree-view are aborted because recipe switching is allowed only under that condition.

When you run this program, it does the following:

- If the recipe fault DO is ON, it first turns it off.
- It then validates if the recipe is a valid recipe.
- Loads the new recipe.
- Sets the status.
- If the status is non-zero, it will turn the Recipe Fault DO to ON.

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)

Example:

To switch to a recipe whose ID is 3, use the following instruction. The status of the call is returned in Numeric Register 1. If the value of register 1 is non-zero, then an error occurred while switching recipes.

```
CALL PKRCSWITCH(Recipe ID=3,Stat Reg=1)
```

7.2 CHECKING THE PRODUCTION STATUS

The production status of each iRPickTool object can be checked.

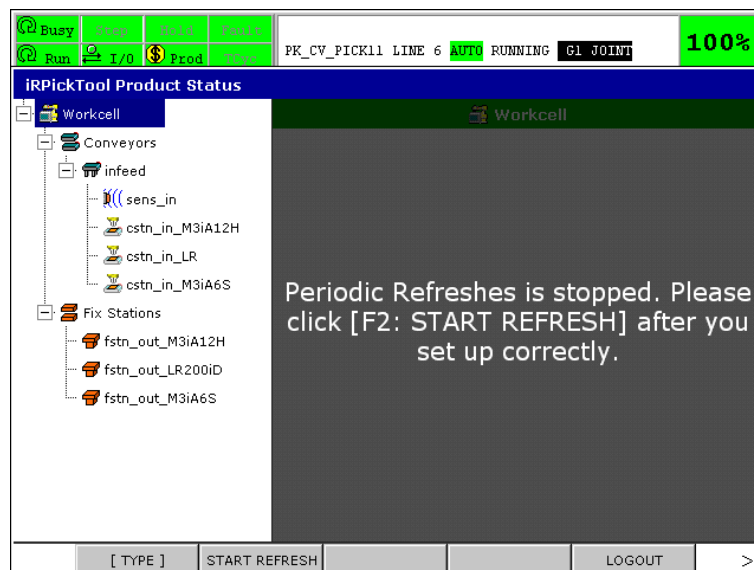
- 1 Press the [MENU] key on the teach pendant.
- 2 Select [0 - - - NEXT - - -].
- 3 Select [4 STATUS].
- 4 Press the F1 [TYPE] key, and select [iRPickTool] from the options.

Basic operations

The operations common to all objects are described below.

Resuming continuous update

If an error occurs during production, a screen as shown below may be displayed indicating that continuous update has been stopped.



In this case, you can resume continuous update by pressing F2 [START REFRESH] key after clearing the error.

To stop continuous update intentionally, press F2 [STOP REFRESH] key during continuous update.

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

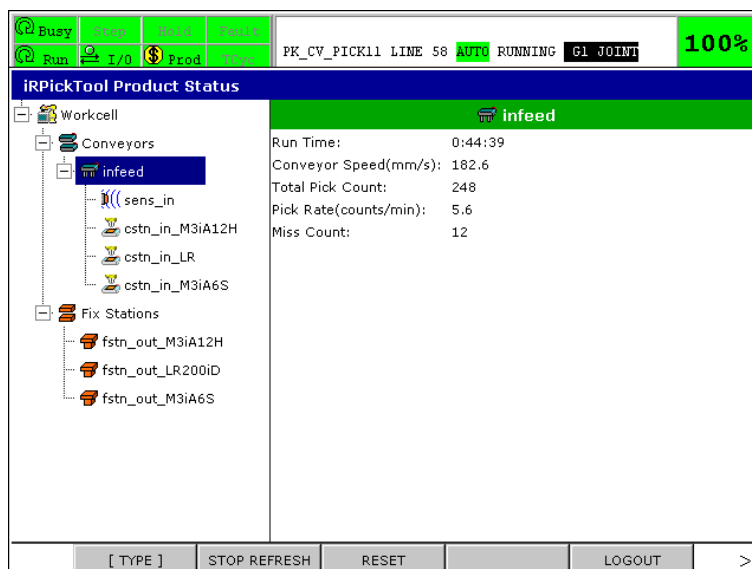
If you perform an operation related to the tree, such as adding a conveyor station in the *iRPickTool* setup screen, while logged in to the production status screen, you cannot have the change reflected in the production status screen without doing anything. To have the change reflected, press the [NEXT] 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.

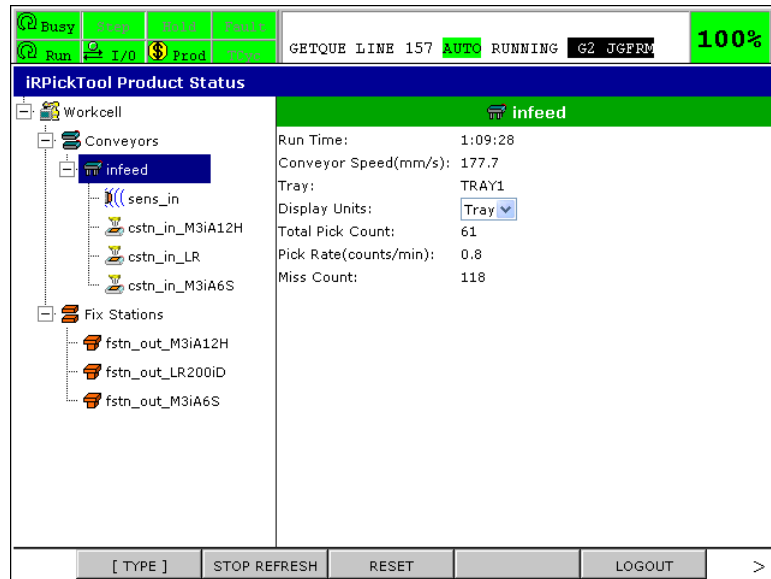
7.2.1 Checking the Status of a Conveyor

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

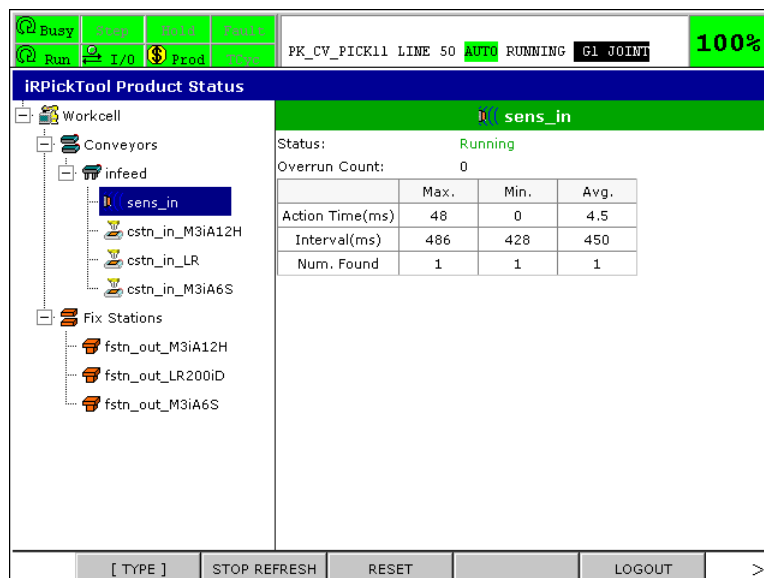
RESET (F3 key):

Pressing this key resets all the data. After the reset, recording starts again.

7.2.2 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 [Trigger Distance] longer than the [Action Time] below, which is mentioned in “When [Distance] and [Find part by vision] are selected”, 4,5.1 (or 6.7.1) “Setting a Sensor in Detail”.

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)

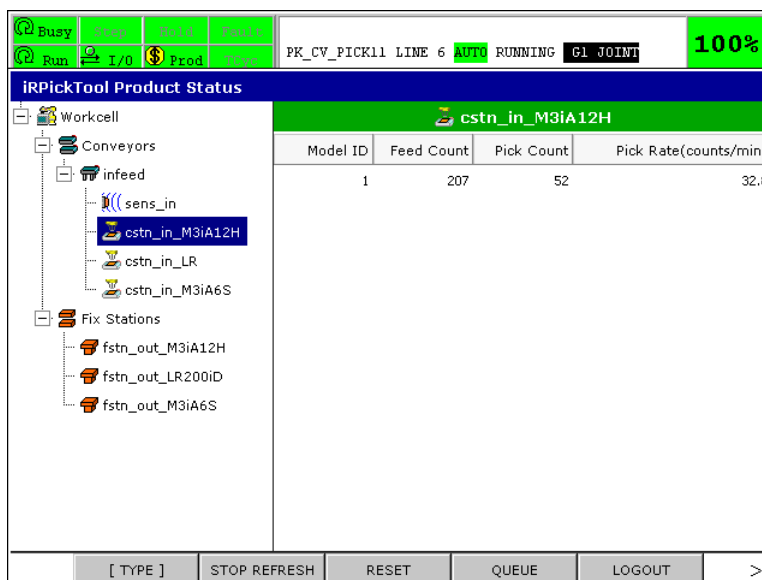
RESET (F3 key):

Pressing this key resets all the data. After the reset, recording starts again.

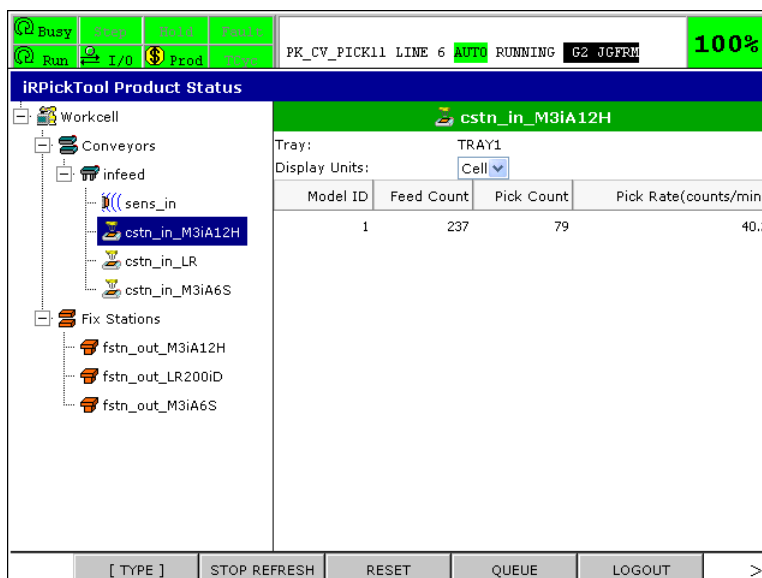
7.2.3 Checking the Status of a Conveyor Station

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.



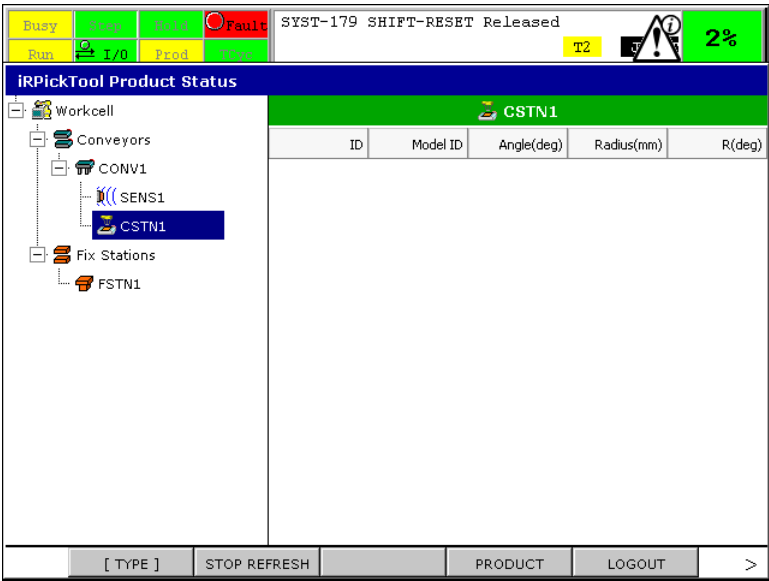
If you press F4 [QUEUE] key, the status, you can view the queue of parts or parts that is yet to be processed by the conveyor station. You can view the leading 10 entries in the queue. Each entry shows the distance of the part on the conveyor along the flow direction (x), and along y and r directions from the nominal tracking frame. This screen is the one displayed when trays are not used.

	Busy		Stop		Hold		Pause	PK_CV_PICK11 LINE 6		AUTO	RUNNING	G1 JOINT	100%															
	Run		I/O		Prod		View																					
iRPickTool Product Status																												
Workcell - Conveyors - infeed - sens_in cstn_in_M3iA12H - cstn_in_LR - cstn_in_M3iA6S - Fix Stations - fstn_out_M3iA12H - fstn_out_LR200iD - fstn_out_M3iA6S				cstn_in_M3iA12H																								
				<table><thead><tr><th>ID</th><th>Model ID</th><th>X(mm)</th><th>Y(mm)</th><th>R(deg)</th></tr></thead><tbody><tr><td>287</td><td>0</td><td>-343.9</td><td>1.0</td><td>0.0</td></tr><tr><td>288</td><td>0</td><td>-418.9</td><td>1.0</td><td>0.0</td></tr></tbody></table>										ID	Model ID	X(mm)	Y(mm)	R(deg)	287	0	-343.9	1.0	0.0	288	0	-418.9	1.0	0.0
				ID	Model ID	X(mm)	Y(mm)	R(deg)																				
				287	0	-343.9	1.0	0.0																				
				288	0	-418.9	1.0	0.0																				
				[TYPE]																								
				STOP REFRESH																								
				PRODUCT																								
				LOGOUT																								
				>																								

When trays are used, the status of each cell in the conveyor station’s queue is displayed as shown below.

Busy	Stop	Hold	Pause	PK_CV_PICK11 LINE 58	AUTO	RUNNING	G2 JGFRM	100%		
Run	I/O	Prod	Setup							
iRPickTool Product Status										
Workcell				cstn_in_M3iA12H						
Conveyors				ID	Layer	Model ID	X(mm)	Y(mm)	Z(mm)	R(deg)
infeed				8001	1	1	216.9	1.6	0.0	-83.0
sens_in				8002	1	1	216.9	1.6	0.0	-83.0
cstn_in_M3iA12H				8003	1	1	216.9	1.6	0.0	-83.0
cstn_in_LR				8160	1	1	136.8	1.6	0.0	-83.0
cstn_in_M3iA6S				8161	1	1	136.8	1.6	0.0	-83.0
Fix Stations				8162	1	1	138.9	1.6	0.0	-83.0
fstn_out_M3iA12H				8163	1	1	138.9	1.6	0.0	-83.0
fstn_out_LR200iD				8320	1	1	58.2	1.6	0.0	-83.0
fstn_out_M3iA6S				8321	1	1	58.2	1.6	0.0	-83.0
				8322	1	1	58.2	1.6	0.0	-83.0
[TYPE]				STOP REFRESH		PRODUCT		LOGOUT		>

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.

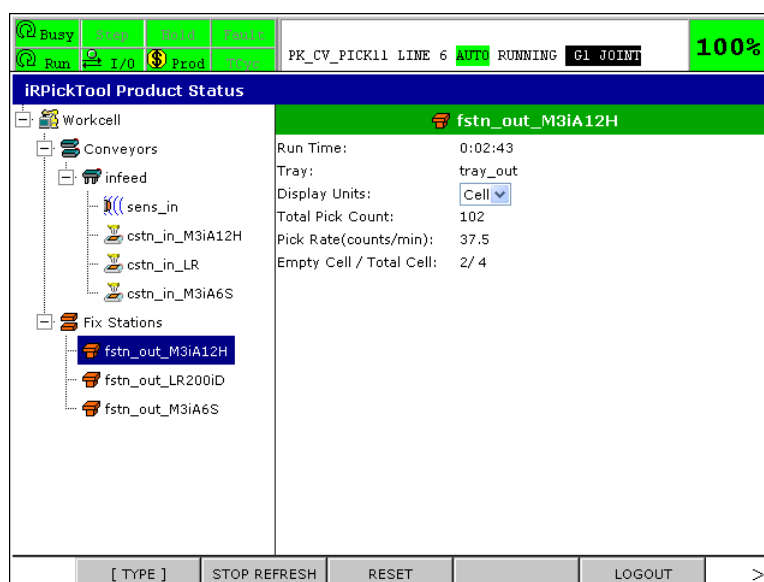
X, Y, Z, R:

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.

7.2.4 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 (F3 key):

Pressing this key resets all the data. After the reset, recording starts again.

7.3 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.

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 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 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.

7.4 SKIPPING THE TRACKING MOTION OUTSIDE THE AREA

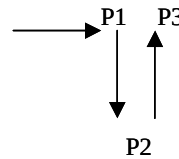
If [Discard Line] is properly set in the conveyor station settings, the robot judges that it cannot catch up with a part when that part passes the "discard line". The information about that part is passed to the next conveyor station. However, the robot may be unable to catch up with a part even when 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

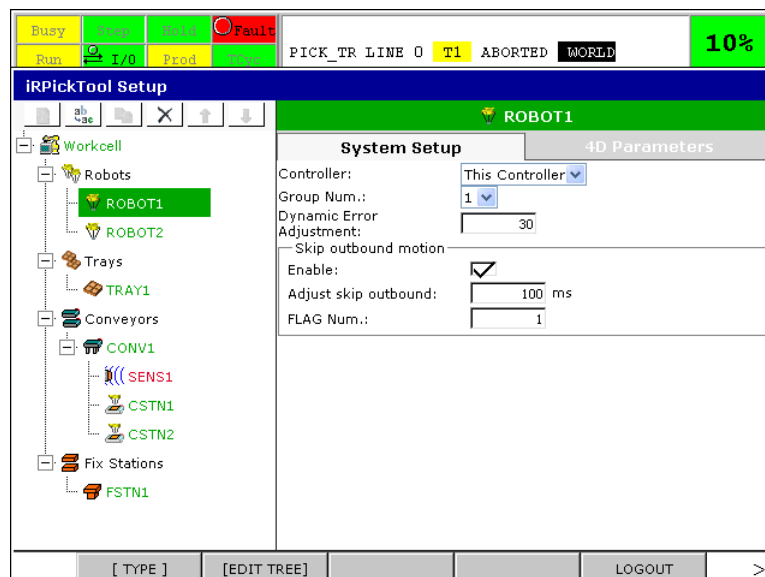
P2: Pickup position

P3: Retraction position



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.

In this case, stop the tracking motion by executing the "STOP_TRACKING" instruction before calling PKCSGETQUE, as described in Subsection 4.9.2.2, "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 maximum travel distance to P1 is 400 mm.

[FLAG Num.] = 1

[Adjust skip outbound] = $400 \text{ (mm)} \div 4000 \text{ (mm/s)} = 0.1 \text{ (s)} = 100 \text{ (ms)}$

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.

7.5 CHECKING THE ROBOT TORQUE

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 [0 - - - NEXT - - -].
- 3 Select [4 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.

Busy	Stop	Reset	Start	Z_MAIN LINE 0 T2 ABORTED WORLD				10%	
Run	I/O	Prod	Cont						

STATUS Axis

Torque Monitor

GRP[1]

	Ave. / Max.	Inpos	OT	VRDY
J1 :	0.000/	.390 1	0	ON
J2 :	.390/	.617 1	0	ON
J3 :	0.000/	.478 1	0	ON
J4 :	.110/	.333 1	0	ON
J5 :	.156/	.324 1	0	ON
J6 :	0.000/	.390 1	0	ON
J7 :	.390/	.617 1	0	ON

[TYPE]

MONITOR

TRACKING

DISTURB

[UTIL]

>

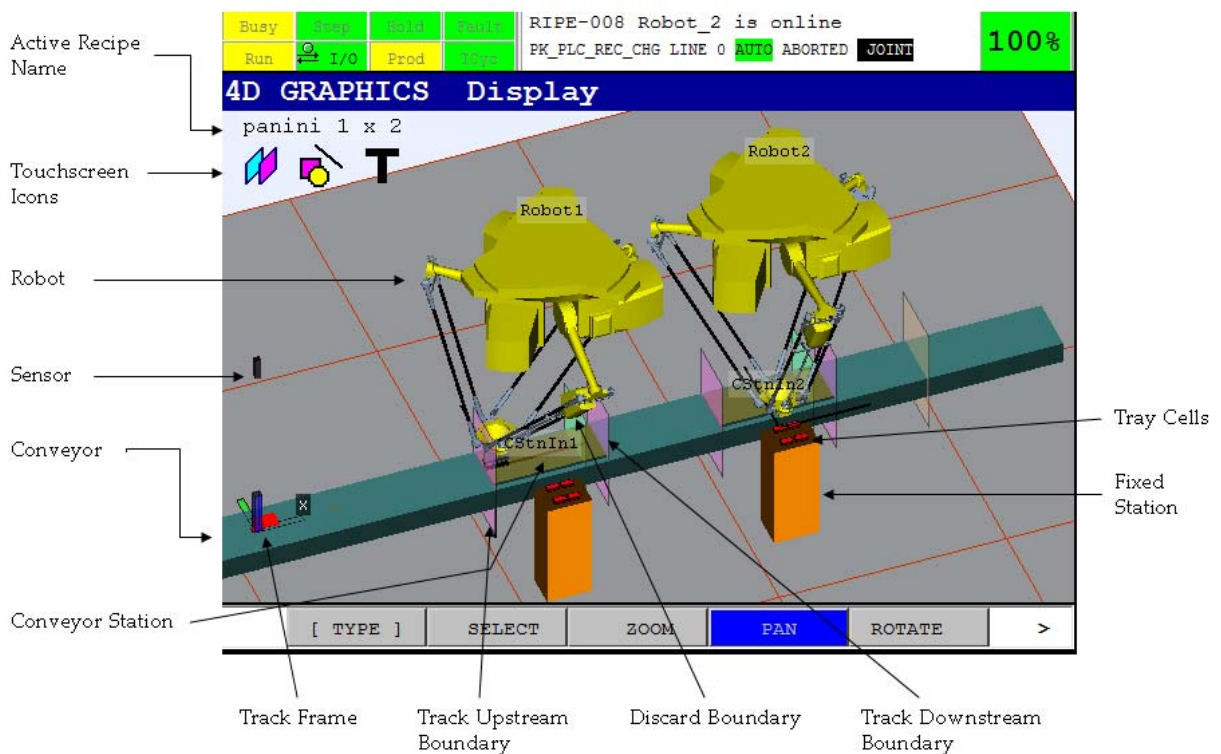
While the system continues steady operation, if the value of [Ave.] for each axis displayed in this screen gradually becomes greater, the motor may be overheating. If so, decrease the speed of the tracking motion.

7.6 iRPickTool 4D GRAPHICS DISPLAY

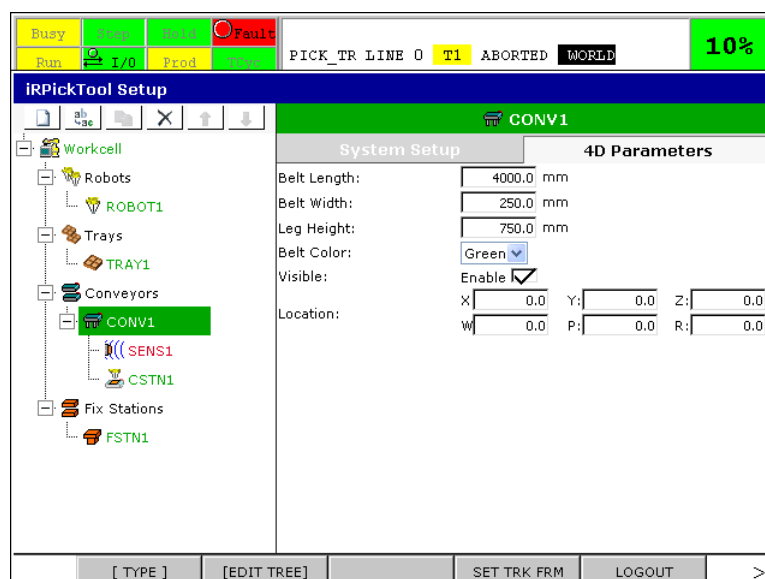
This chapter describes the setup and use of the R764: 4D Graphics option.

When the R764: 4D Graphics option is installed, you can visualize all your workcell objects in 3-dimensions on the iPendant. 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 iRPickTool completely for a 2-robot workcell.



The 4D Setup is done in the *iRPickTool* 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.



7.6.1 Displaying the 4D Workcell

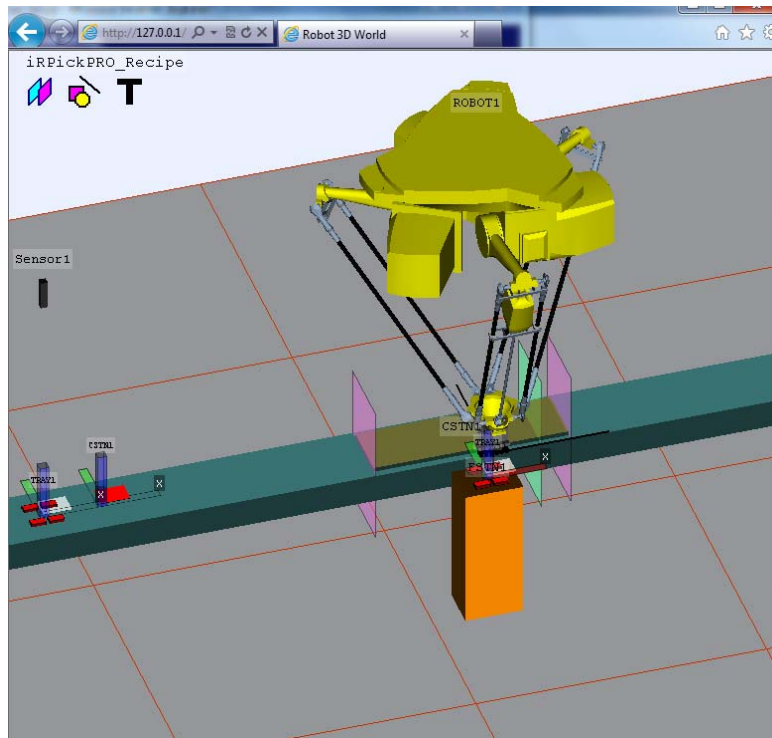
Once you change the 4D properties of an object, you will then need to go to the [MENU] key – 4D Graphics – 4D Display menu on the iPendant. In the 4D Display menu, your workcell will be displayed in 3D and you can see the effect of the change you made.

If you wish to see the effect of the changes to the properties you make in the treeview immediately, then you will need to do have a web browser window open on your laptop or a PC connected to the controller. Then you will be able to do your treeview setup on the iPendant and simultaneously look at the 4D Graphics window on the laptop or PC. To do so, you will need to make sure that you have installed the R558: Internet Connectivity/Custo option.

As well as iPendant, when you browse to the controller's home page, you will be able to find the "Full Screen 4D Display" link as shown below:

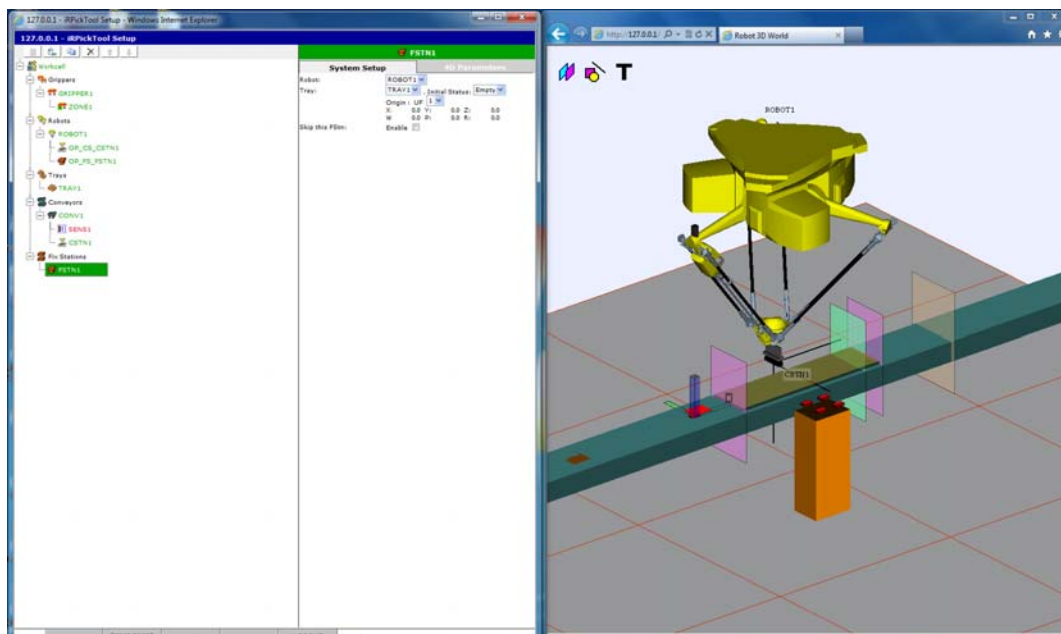


When you click on the 4D display link you will see the 4D menu in the Browser as shown below.



You can also display both the treeview as well as the 4D display in two separate windows on your laptop or PC connected to the controller as shown below. To see the two windows side-by-side:

- 1 Minimize all open windows.
- 2 Open an Internet Explorer window and browse to the robot home page. Click on the “Workcell Setup” link under iRPickTool.
- 3 Open another new Internet Explorer window and browse to the robot home page. Click on the “Full Screen 4D Display” link.
- 4 Right click on an empty space on the Microsoft Windows taskbar and select “Show windows side by side” from the pop-up menu.



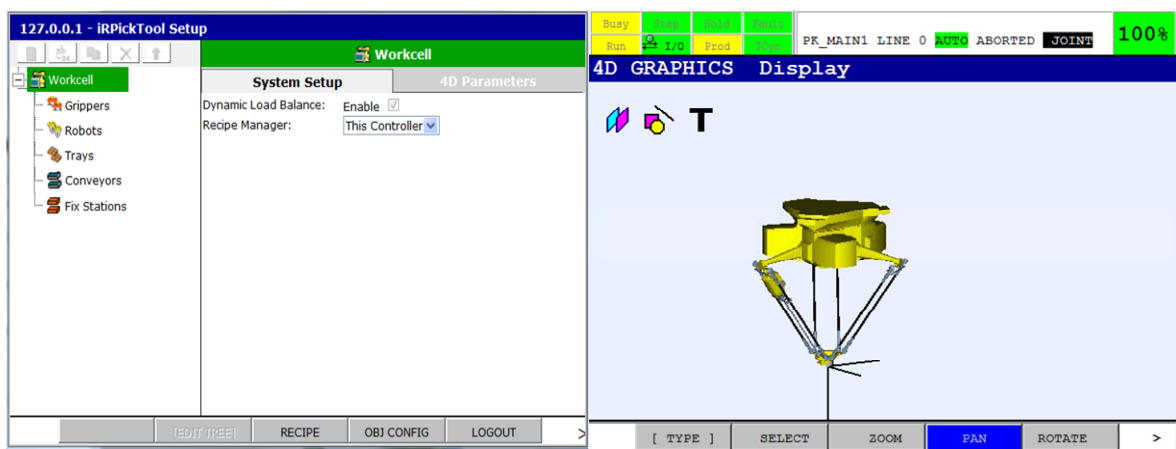
7.6.2 Creating the 4D Workcell

Empty Workcell

When you have the R764: 4D Graphics option installed, and you have done no setup of *iRPickTool* 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 wizard in the Conveyor node of the treeview.

Any empty workcell contains no other *iRPickTool* objects such as the gripper, the conveyor or the fixed stations. See the empty workcell in the figure below.

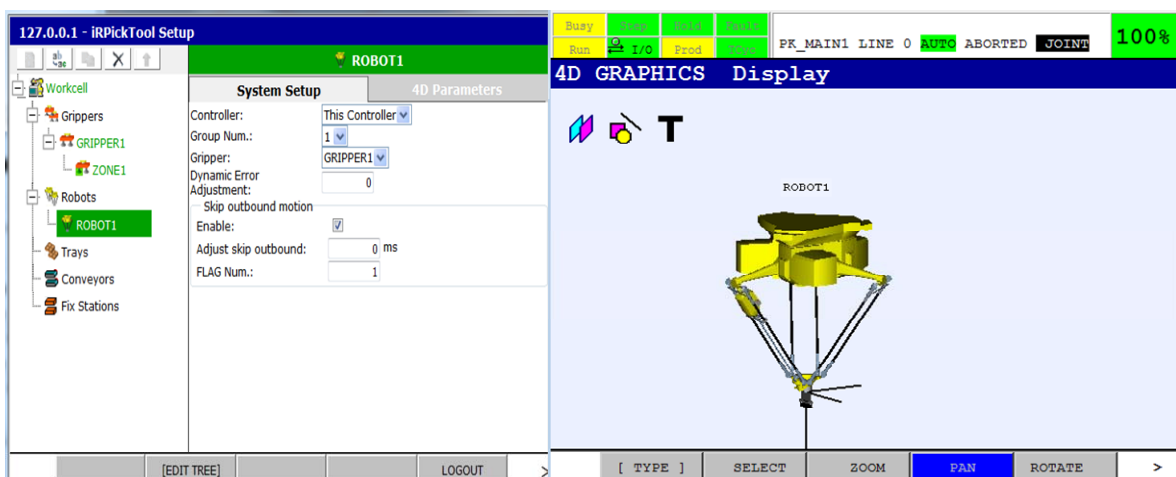


Creating 4D Workcell Objects

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.

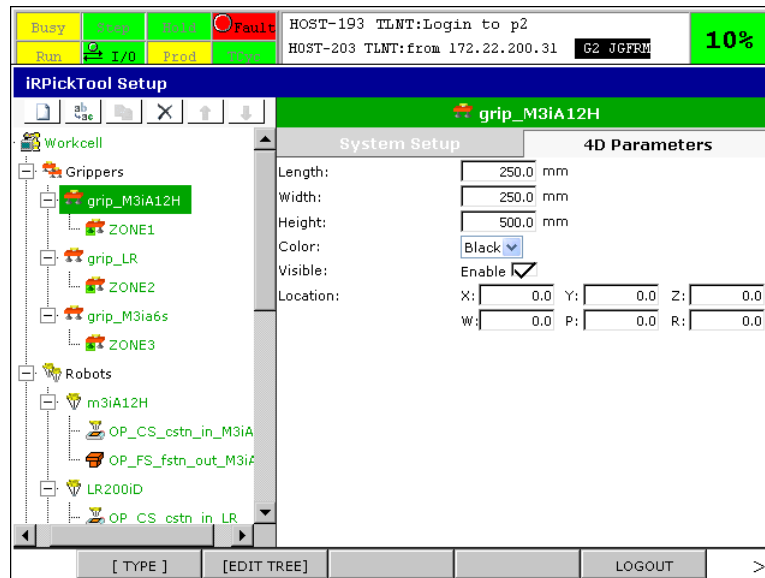
Creating a new 4D Gripper

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, click on 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 section 7.6.7, “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.

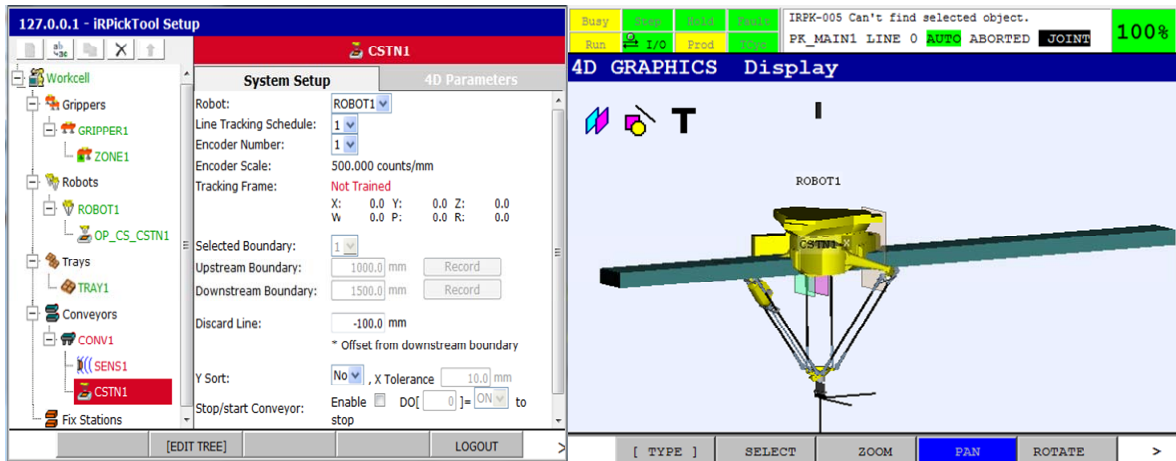
Creating a new 4D Conveyor

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, iRPickTool 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 wizard to teach the track frame, the conveyor will move to the proper location.

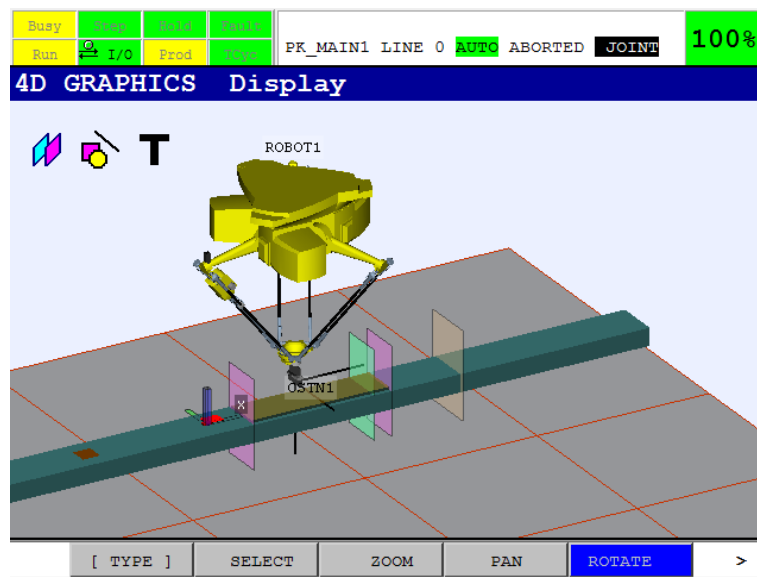
When you create a new conveyor in the treeview, it also creates a sensor and a conveyor station by default. The 4D display menu will also attempt to display the sensor and the conveyor station. However,

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.

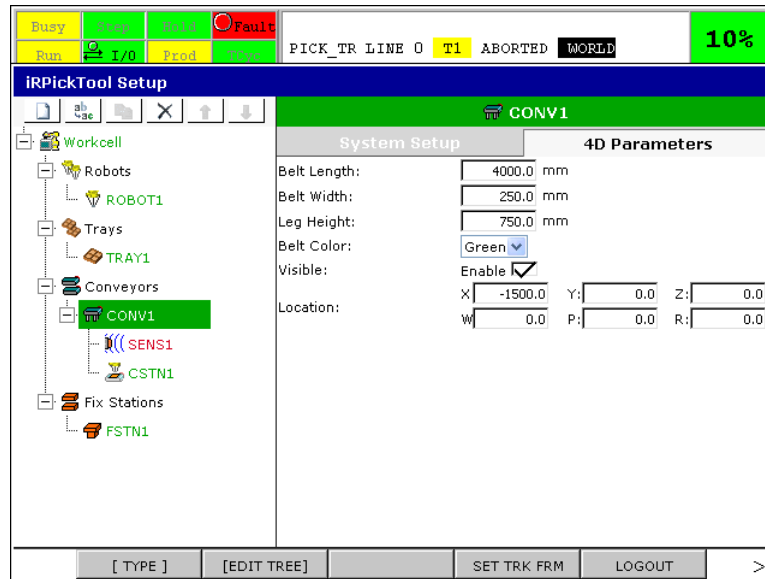


If you later go through the track wizard in the treeview and teach the track frame properly, and you go through the Sensor Wizard in the Sensor menu, then the conveyor, the sensor and the conveyor station will all be located correctly in the workcell as shown in the figure below.



Modifying a 4D Conveyor

In the tree view on the left side, click on 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 section 7.6.7, “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

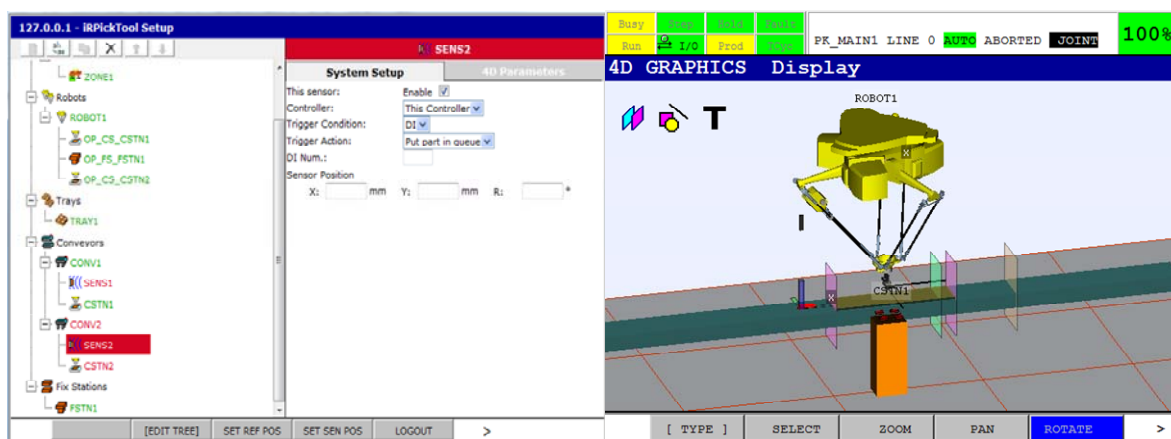
Enable or disable the visibility of the conveyor using the check box.

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.

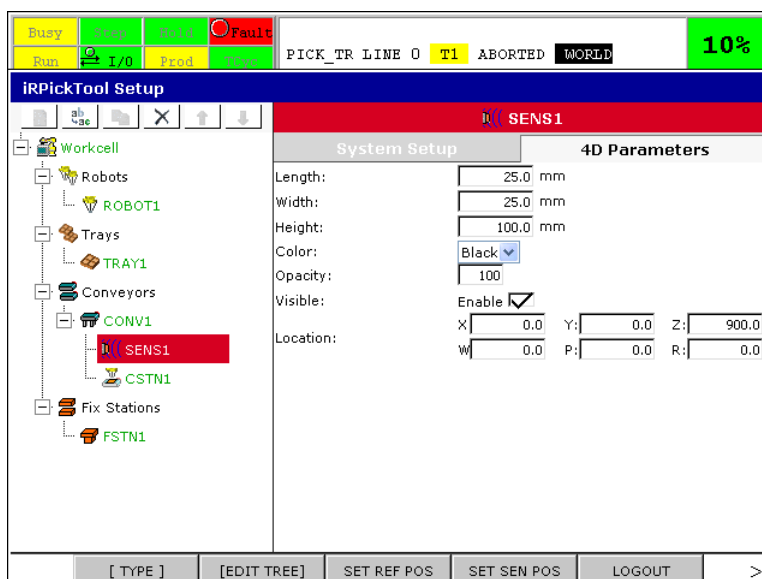
Creating a new 4D Sensor

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, iRPickTool 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 wizard 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, click on 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 section 7.6.7, “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

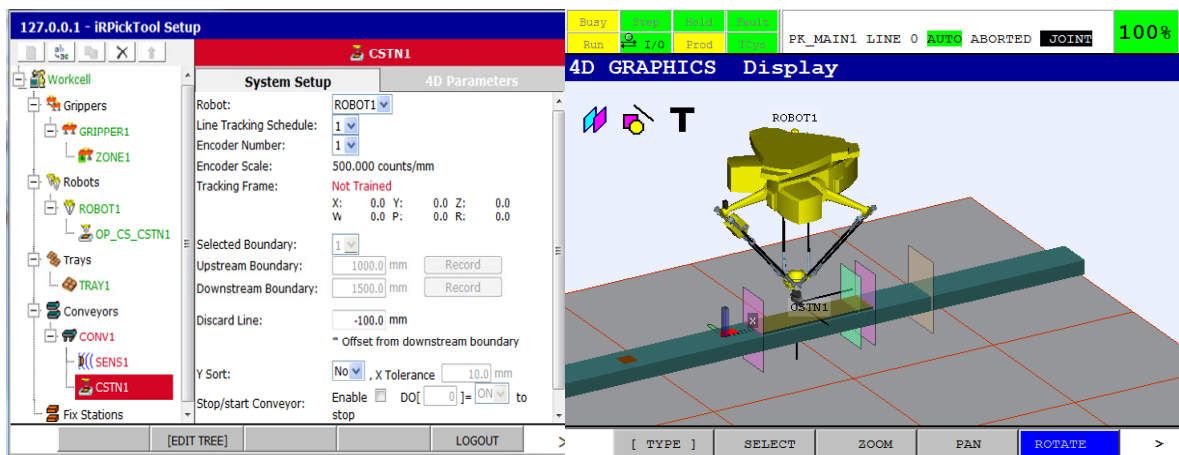
When you create a Conveyor Station in the treeview, a 3D Conveyor Station is automatically created as follows:

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.

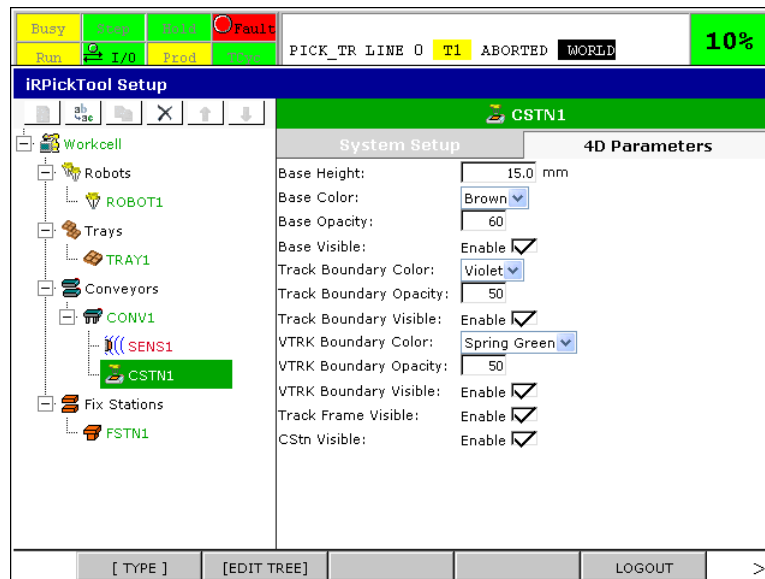
The discard boundary is displayed in Spring Green color.

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, click on 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.

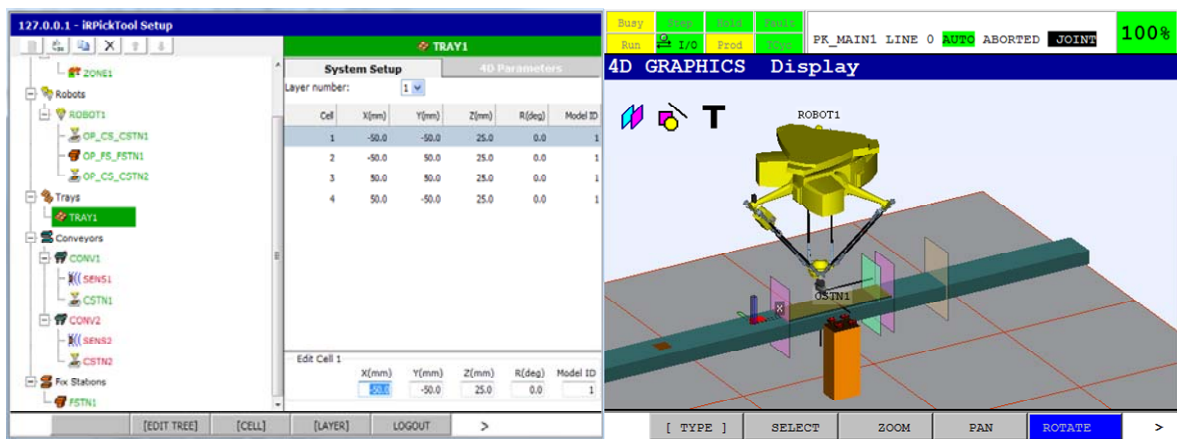
CStn Visible

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

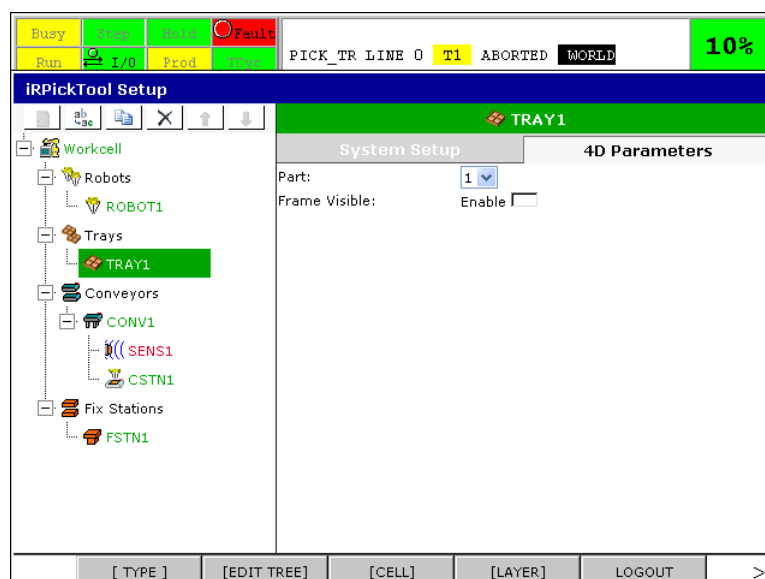
When you create a tray in the treeview and assign it to a Fixed Station as shown in the figure below, the 3D tray cells are created automatically at the Fixed Station. You can also assign trays to conveyors in the conveyor menu of the treeview. The tray cells are always drawn in the tray frame. For a 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 wizard 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, click on 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

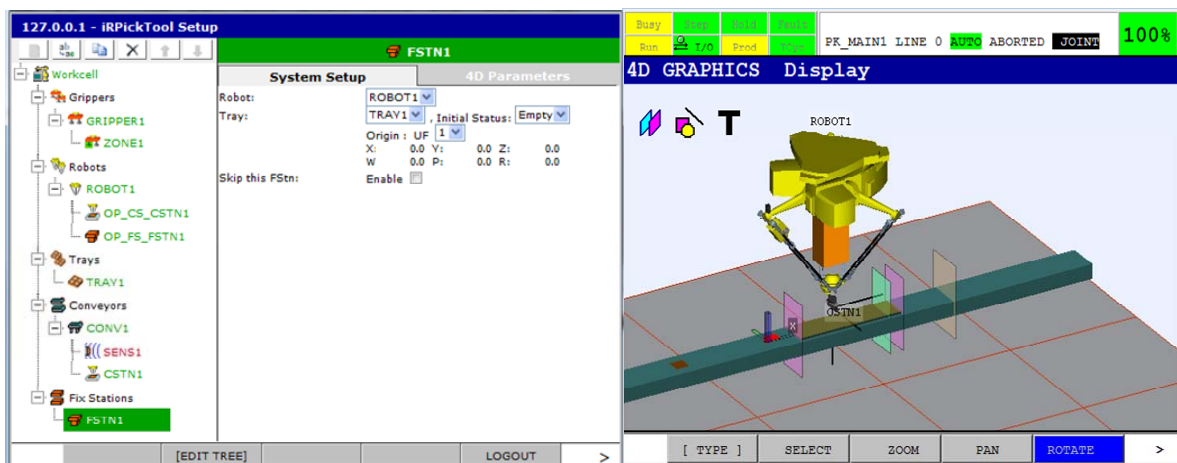
A tray contains one or more cells. The part that is used at each cell location can be selected in the drop-down box. In *iRPickTool*, there is no User interface to setup the part properties. You can setup the size and color of the part from the system variable \$pkpt4d.

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.

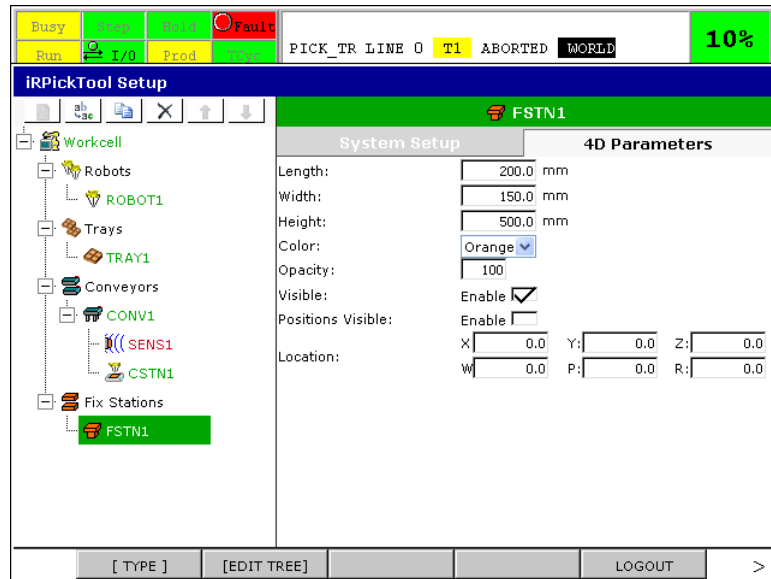
Creating a new 4D Fixed Station

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, *iRPickTool* 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, click on 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 section 7.6.7, “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.

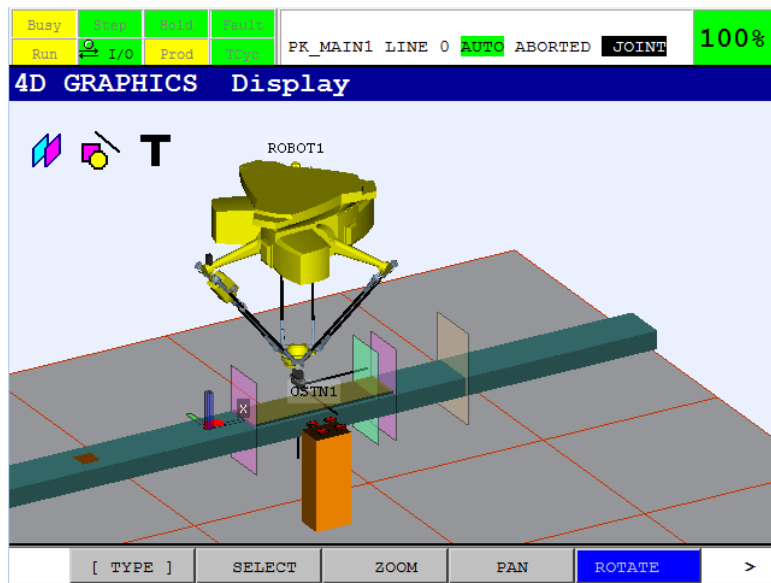
Positions Visible

Enable or disable the checkbox to view the approach and pick (or 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:



7.6.3 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 *iPendant* 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.

iRPickTool Show/Hide Icon

The *iRPickTool* icon  is meant for showing and hiding all the *iRPickTool* visualizations. You may hide the *iRPickTool* 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 *iRVision*”.

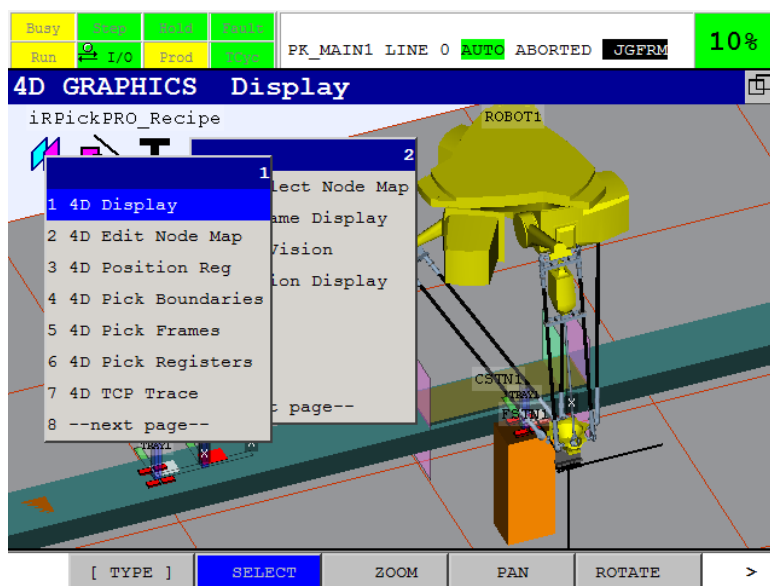
Text Show/Hide Icon

The Text icon  is meant for showing and hiding Text entries in the 4D scene. *iRPickTool* allows you to display the names of the Robots, conveyor stations, fixed stations, sensors, tray frames and track frames.

7.6.4 Using the 4D [TYPE] Scenes

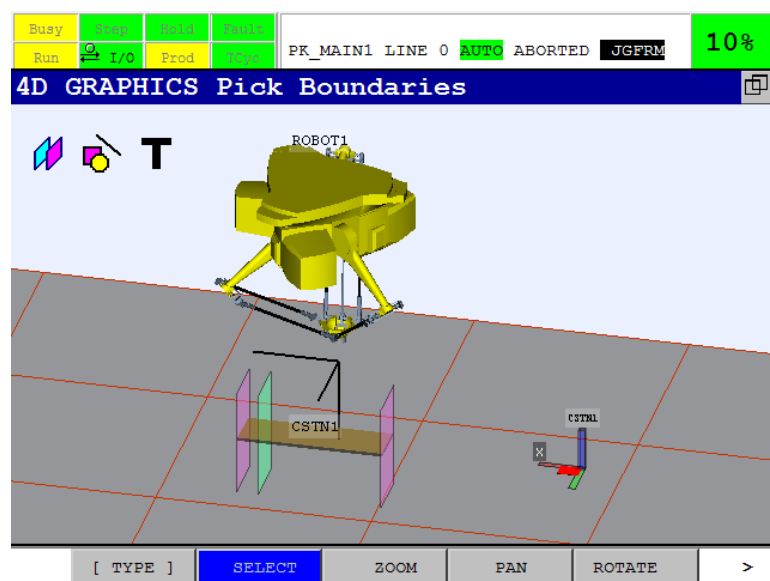
In the 4D Display menu, the F1 function key is assigned to [TYPE]. When you click on F1 [TYPE] key, a subwindow will open up containing additional scenes that you can view as shown below. You can explore all the scenes, but the following three scenes may be useful to you check your setup:

4. 4D Pick Boundaries
5. 4D Pick Frames
6. 4D Pick Registers (Not implemented).



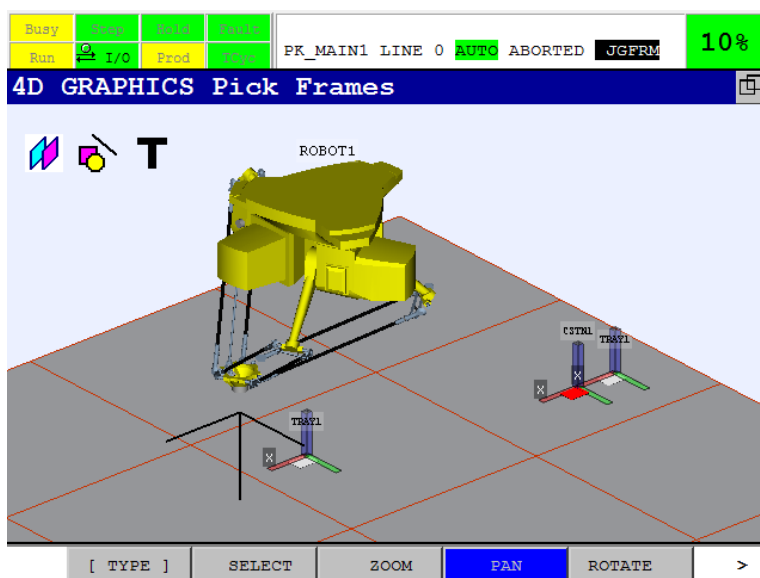
7.6.4.1 4D pick boundaries

When you click on 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.



7.6.4.2 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.



7.6.5 Deleting the 4D Workcell

You can delete the entire 4D Workcell only when you delete all the objects from the *iRPickTool* preview. You can delete individual objects such as a sensor or a conveyor station by deleting them in the treeview. You cannot delete the robots from the scene.

If you wish to temporarily turn off the visualizations of the *iRPickTool* objects you can use the

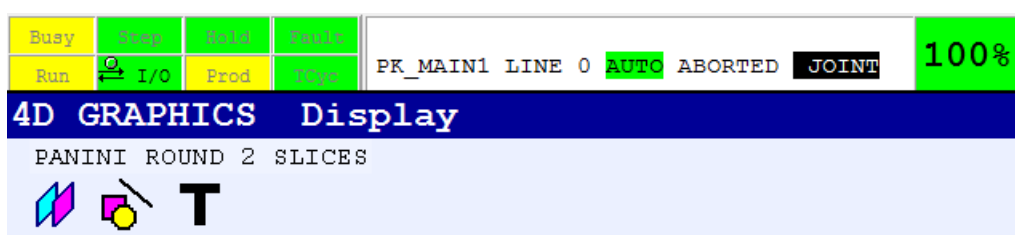
iRPickTool Show/Hide icon



7.6.6 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 *iRPickTool* icons. See figure below. The text “PANINI ROUND 2 SLICES” is the name of the active recipe.

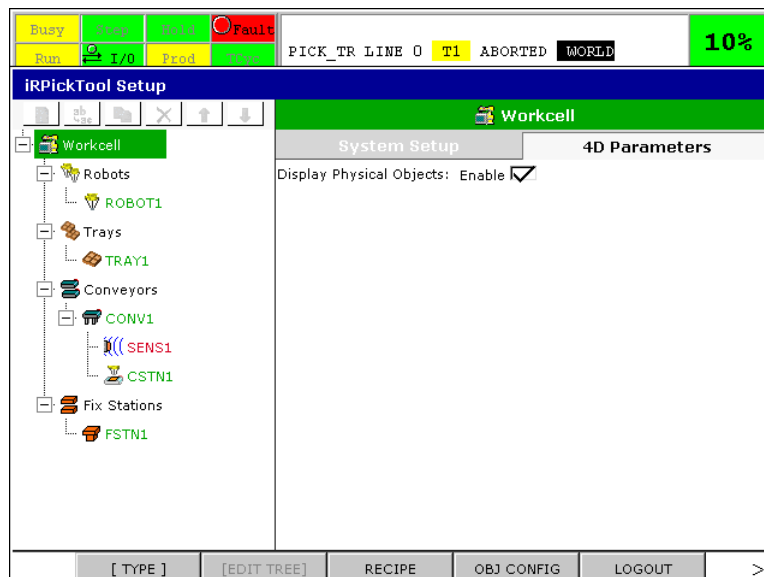


7.6.7 Loading Custom CAD Files

By default, *iRPickTool* 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 *iRPickTool* will not try to create the physical objects in the workcell such as the grippers, sensor boxes, conveyors and Fixed Station boxes. To prevent *iRPickTool* 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, *iRPickTool* will only create logical objects such as conveyor stations with boundaries or tray cells. See figure below.



7.6.8 Limitations

Do not use the 4D Graphics for *iRPickTool* 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.

7.7 BUFFER FUNCTION

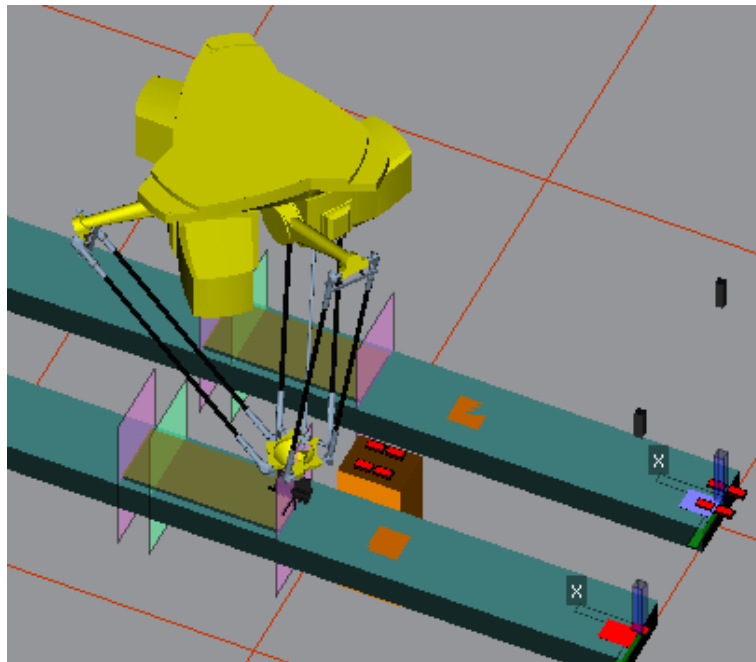
This section describes the setup and use of the Plug & Play buffer functions. This option is included in the R857: *iRPickTool* / Option Package Add-on option, and it contains the User Interface for the buffer operations in the OP_FS_xxx menu, TP programs for buffer handling and some KAREL macros to 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: *iRPickTool Basic*. However, *iRPickTool 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:



iRPickTool 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 *iRPickTool* robot (group).

The *iRPickTool* Buffer operations will work well for only “single pick” and “single drop” applications.

7.7.1 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:

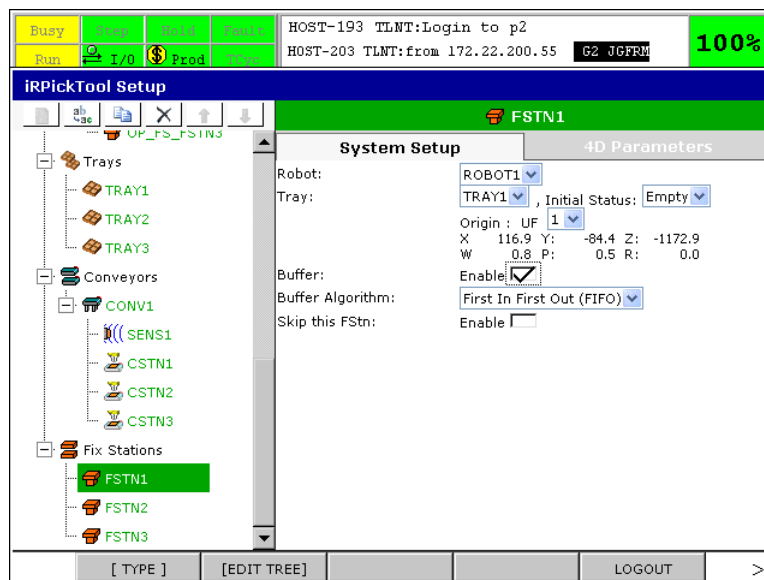
- When a pick target is available, but drop target is not available, and the buffer has empty cells, the robot places the part in the buffer.
- When a pick target is known to exist but it is upstream of the tracking upstream boundary, then a place target is available, and, the robot may still pick from 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.

7.7.2 Setting up a Buffer

To create a buffer, create a Fixed Station in the *iRPickTool* tree-view. Section 6.9, "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

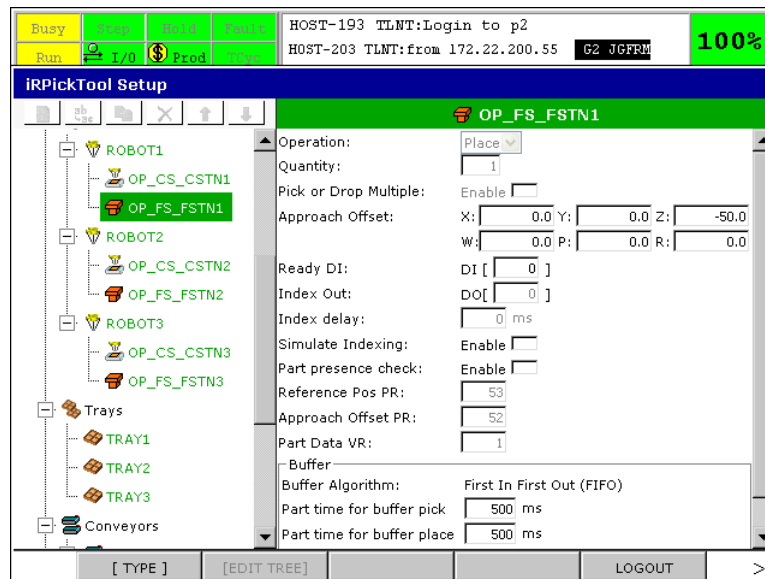
If "Enable" is checked, this Fixed Station is used for a buffer.

Buffer Algorithm

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 6.4.2, “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.

The pick quantity of the Fixed Station Operation will be disabled when buffer is selected in the Fixed Station menu.

NOTE

The buffer does not have its own independent pick quantity. During the pick cycle, it uses the pick quantity of the pick conveyor station. During the drop cycle, it uses the drop quantity of the drop conveyor station.

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.

7.7.3 Buffer KAREL Macros

The buffer related KAREL macros are defined in this section.

7.7.3.1 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)

7.7.3.2 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 *iRPickTool* 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)

7.7.3.3 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 *iRPickTool* 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)

7.7.4 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 6.15, “STANDARD TP PROGRAMS” in detail.

7.7.4.1 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 iRPickTool 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: --: Reset tray cell counter
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]
```

7.7.4.2 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 6.5.2.15 “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: -- 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: Pick Loop]
21: IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
22:
23: -- Get a part 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,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 PKFSGETPMUL(FStn ID=R[199:Buf Id G1],PikMultiple Reg=95)
31: -- Find the last grip zone
32: CALL PK_GR_GET_ENDZONE1(PickDropCounter=R[107:Counter ],
   Multiple=R[95:TemporaryUse1],Quantity=R[111:Pk1x Qty],End Zone Reg=106)
33: IF R[95:TemporaryUse1]=0,JMP LBL[300]
34: IF R[107:Counter G1]=1,JMP LBL[300]
35: JMP LBL[2000]
36:
37:
38: LBL[300]
39: UTOOL_NUM=R[107:Counter G1]
40: -- Move to approach
41: L PR[53:Bf Ref Pos] max_speed CNT100 VOFFSET,VR[1] Tool_Offset,
   PR[52:Bf Ap Ofst]
42: -- Move to pick pos
43: 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])
44: -- Pick delay
45: CALL PKGRGETPKDLY(Gripper ID=R[104:Gripper Id G1],PickDelay Reg=95)
46: WAIT R[95]
47: -- Check part pres
48: CALL PKFSGETPPCHK(FStn ID=R[199:Buf Id G1],PartPresChk Reg=95)
49: IF R[95:TemporaryUse1]=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,Stat Reg=126,
    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[53:Bf Ref Pos] max_speed CNT100 VOFFSET,VR[1]
    Tool_Offset,PR[52:Bf Ap Ofst]
59: IF R[126:Buf PP Stat G1]<>0,JMP LBL[1500]
60: JMP LBL[2000]
61:
62:
63: LBL[1500: PartGotdropped]
64: -- Return part in VR1 to Queue
65: CALL PKFSACKBUF(FStn ID=R[199:Buf Id G1],Return)
66: R[126:Buf PP Stat G1]=0
67: JMP LBL[10000]
68:
69:
70: LBL[2000: Pick ok]
71: -- Ack the part has been picked
72: ! PKFSACKBUF(buf_id, ack, vr (opt
73: CALL PKFSACKBUF(FStn ID=R[199:Buf Id G1],Success)
74: -- Update tray cell count
75: R[204:Buf Ctr G1]=R[204:Buf Ctr G1]-1
76: -- Update pick counter
77: R[107:Counter G1]=R[107:Counter G1]+1
78: IF (R[107:Counter G1]<=R[111:Pk1x Qty]),JMP LBL[20]
79: R[205:Picked G1]=1
80:
81: LBL[3000: CheckIndex]
82: ! For indexing buffers only
83:
84:
85: LBL[10000: End]
[End]

```

7.7.4.3 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 6.5.2.16 “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: Drop Loop]
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,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: Multidrop]
30:  CALL PKFSGETPMUL(FStn ID=R[199:Buf Id G1],PikMultiple Reg=95)
31:  -- Find the last grip zone
32:  CALL PK_GR_GET_ENDZONE1(PickDropCounter=R[107:Counter ],
   Multiple=R[95:TemporaryUse1],Quantity=R[131:Dp1x Qty],End Zone Reg=106)
33:  IF R[95:TemporaryUse1]=0,JMP LBL[300]
34:  IF R[107:Counter G1]=1,JMP LBL[300]
35:  JMP LBL[2000]
36:
37:
38:  LBL[300]
39:  UTOOL_NUM=R[107:Counter G1]
40:  -- Move to approach
41:L PR[53:Bf Ref Pos] max_speed CNT100 VOFFSET,VR[1] Tool_Offset,
   PR[52:Bf Ap Ofst]
42:  -- Move to drop pos
43: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])
44:  -- Drop delay
45:  CALL PKGRGETDPDLY(Gripper ID=R[104:Gripper Id G1],DropDelay Reg=95)
46:  WAIT R[95]
47:  -- Check part pres
48:  CALL PKFSGETPPCHK(FStn ID=R[199:Buf Id G1],PartPresChk Reg=95)
49:  IF R[95:TemporaryUse1]=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,Stat Reg=126,
   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[53:Bf Ref Pos] max_speed CNT100 VOFFSET,VR[1]
    Tool_Offset,PR[52:Bf Ap Ofst]
59:  IF R[126:Buf PP Stat G1]<>0,JMP LBL[1500]
60:  JMP LBL[2000]
61:
62:
63:  LBL[1500: PartGotdropped]
64:  -- Return part in VR1 to Queue
65:  CALL PKFSACKBUF(FStn ID=R[199:Buf Id G1],Return)
66:  R[126:Buf PP Stat G1]=0
67:  JMP LBL[10000]
68:
69:
70:  LBL[2000: Pick ok]
71:  -- Ack the part has been picked
72:  ! PKFSACKBUF(buf_id, ack, vr (opt
73:  CALL PKFSACKBUF(FStn ID=R[199:Buf Id G1],Success)
74:  -- Update tray cell count
75:  R[204:Buf Ctr G1]=R[204:Buf Ctr G1]+1
76:  -- Update pick counter
77:  R[107:Counter G1]=R[107:Counter G1]+1
78:  IF (R[107:Counter G1]<=R[131:Dp1x Qty]),JMP LBL[20]
79:  R[205:Picked G1]=0
80:
81:  LBL[3000: CheckIndex]
82:  ! For indexing buffers only
83:
84:
85:  LBL[10000: End]
[End]

```

7.7.4.4 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=1,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]

```

7.7.4.5 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]

```

7.7.4.6 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 (or the quantity in the gripper)

Argument 4: The pick counter (next pick number)

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]
```

7.7.4.7 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 (or the quantity in the gripper)

Argument 4: The drop counter (next drop number)

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]

```

7.7.4.8 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 Cvld]>0,CALL PK_CV_OK2DROP1(Conv ID=R[129:Dp1x Cvld],
    Buffer ID=R[199:Buf Id G1],Order=1,Drop Counter=0,
    BufDropTime Reg=201,Ok2Drop Reg=97)
11: CALL PK_BF_OK2PICK1(Buffer ID=R[199:Buf Id G1],Ok2Pick Reg=202)
12: R[98:TemporaryUse4]=R[107:Counter G1]-1
13: CALL PK_CV_OK2PICK1(Conv ID=R[109:Pk1x Cvld],
    Buffer ID=R[199:Buf Id G1],Order=R[107:Counter G1],
    Pick Counter=R[98:TemporaryUse],BufPickTime Reg=200,Ok2Pick Reg=203)
14: IF R[203:CS Available G1]>0,CALL PK_CV_PICK11
15: IF ((R[203:CS Available G1]<=0) AND (R[202:Buf Available G1]>0)
    AND (R[97:TemporaryUse3]>0)),CALL PK_BF_PICK1
16: IF R[205:Picked G1]<1,CALL PK_CV_PICK11
17: IF R[205:Picked G1]<1,JMP LBL[50]
18: LBL[10000]
[End]

```

7.7.4.9 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: R[98:TemporaryUse4]=R[107:Counter G1]-1
11: CALL PK_CV_OK2DROP1(Conv ID=R[129:Dp1x Cvld],
    Buffer ID=R[199:Buf Id G1],Order=R[107:Counter G1],
    Drop Counter=R[98:TemporaryUse],BufDropTime Reg=201,
    Ok2Drop Reg=203)

```

```
12: IF R[203:CS Available G1]>0,CALL PK_CV_DROP11
13: IF ((R[203:CS Available G1]<=0) AND (R[202:Buf Available G1]>0)
    AND (R[205:Picked G1]>0)),CALL PK_BF_DROP1
14: IF R[205:Picked G1]>0,CALL PK_CV_DROP11
15: IF R[205:Picked G1]>0,JMP LBL[50]
16: LBL[10000]
[End]
```

7.8 EFFICIENCY TOOLS

This section describes the setup and use of the Efficiency Tools function. This function is included in the R857: iRPickTool / Option Package Add-on option.

The goal of the Efficiency Tools option is to provide functions that help you to create your picking application more efficiently.

7.8.1 Pick Only Reachable Parts

Use the “Pick only reachable parts” feature to prevent “Position not reachable” errors during the robot motions to and from conveyors.

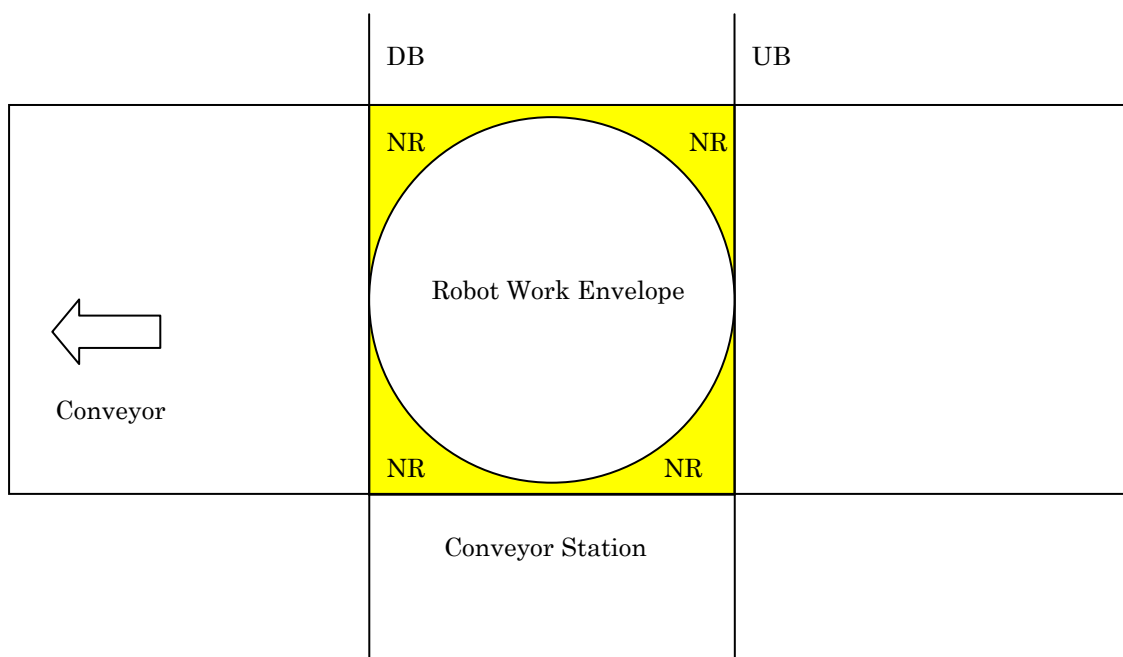
This function eliminates the “MOTN-017 Limit error” and the “MOTN-018 Position not reachable” errors completely during picking from and placing to conveyors (in tracking TP programs). When this error occurs in the tracking programs, without this function, you have no choice except to move the tracking upstream or downstream boundaries such that the area in which the robot picks is limited. However, with this function, you don’t have to move boundaries or reduce the area in which the robot picks. The robot will never go to pick or place a part that is not reachable. In the meantime if there are other reachable parts in the work envelope, the robot will pick them. It will pick or place only when the target part is in its reachable area. See figure below. The regions labeled “NR” are not reachable. By default, the robot would try to pick up a part in the NR areas because it is in between the Upstream Boundary UB and the Downstream Boundary DB. This stops the robot and causes down-time and missed parts. By using the new function, the robot will not pick in the “NR” areas. It will only pick inside the circular work envelope. The robot may even wait until the part flows from the “NR” areas to the inside of the circular work envelope. The position check is always made with the current UTOOL so it is considered. There it also works when the robot needs to pick or drop multiple parts.



CAUTION

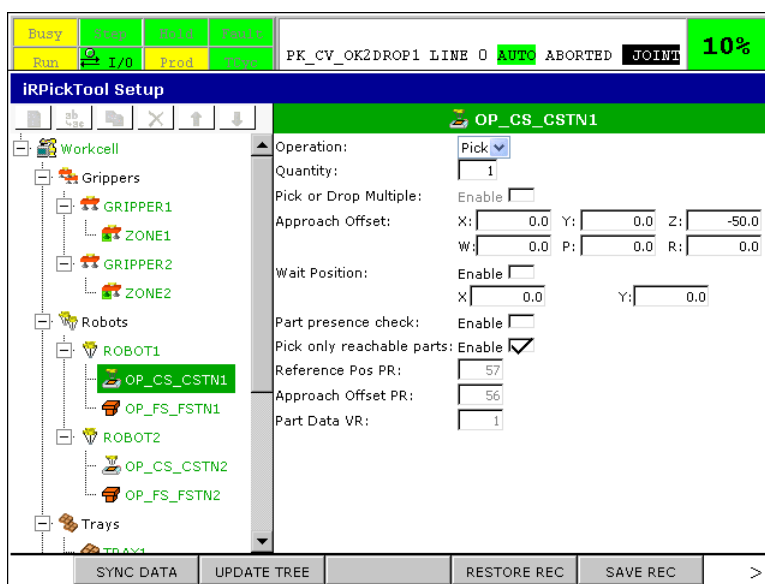
You cannot use the following functions with “Pick only reachable parts” feature.

- Tray
- Y Sort
- Consecutive pick when the robot picks and places multiple parts.



7.8.2 Setting up Pick Only Reachable Parts

This feature can be enabled and disabled for each conveyor station in the Conveyor Station Operations menu in the *iRPickTool* treeview as shown in the figure below.



Enable the checkbox for “Pick only reachable parts”.

Note that you will see this item only if you purchased the R857: *iRPickTool* / Option Package Add-on option.

Limitations

When this feature is used, you cannot specify `model_id` which is an option argument to `PKCSGETQINRG.PC`.

7.8.3 KAREL Macros

This option provides KAREL macros that allow you to get reachable parts from the queue and also to check if any tracking position is reachable at any time.

7.8.3.1 PKCSGETPOSCK.PC

This program is used to check if “Pick only reachable parts” is enabled for a conveyor station operation corresponding to specified conveyor station. If it returns zero, then the feature is not enabled. It is the dedicated KAREL program for Plug & Play.

Argument 1:

Specify the Id of the conveyor station.

Argument 2:

Specify the operation at this conveyor station. PICK = 0, PLACE = 1.

NOTE

When the argument guidance is supported, you select one of the following options.

1 Pick

2 Place

However, the operation 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 enabled (=1) or disabled (=0).

Example:

To check if “Pick Only Reachable Parts” is enabled for a conveyor station whose Id=1 where a pick operation is done, use the following call to return the value of “Pick only reachable parts” in Register 95.

```
CALL PKCSGETPOSCK(CStn ID=1,Pick,PosCheck Reg=95)
```

7.8.3.2 PKCSGETQINRG.PC

This program must be used when you wish to use the “Pick only reachable parts” function. This program is integrated with the standard TP programs provided by Plug & Play. If you are a Basic function user, then you need to call this program in your tracking pick and place programs instead of the standard PKCSGETQUEUE.PC.

When you run this program, it does the following:

- It checks if the allocated part on the conveyor is in the area between upstream and downstream boundaries. If so, it checks if it is reachable. If reachable, it then allocates it.
- If the allocated part is not reachable, it then scans the queue further to see if there is another part that is reachable. If there is one, it allocates the part.
- If no part in the queue is reachable, it continues to check the queue until it gets a part that is reachable.

NOTE

If you use PKCSGETQINRG, you will not be able to request an allocation for a part based on model_id. In that case, you should use the standard PKCSGETQUE.

Argument 1:

Specify an ID of the conveyor station.

Argument 2:

Specify 1.

**CAUTION**

“Pick only reachable parts” function doesn’t support the consecutive pick.

Argument 3:

Specify the timeout value as in the standard PKCSGETQUE.

Argument 4:

Specify whether position check should be made (=“Do Pos Check”) or not (=“No Pos Check”). Usually, specify “Do Pos Check”.

NOTE

If the argument guidance is not supported in your software version, specify 1 as “Do Pos Check”.

Argument 5:

Specify the Position Register number of the Reference Position used for the conveyor station.

Argument 6:

Specify the approach offset Position Register number used for the conveyor station.

Argument 7:

Specify whether the approach offset used in Argument 6 is a tool offset (=1) or a offset (=0).

NOTE

When the argument guidance is supported, you select one of the following options.

1 Tool Offset

2 Offset

However, the approach offset is internally set as Tool Offset=1 and Offset=0. So if the argument guidance is not supported in your software version, specify 1 as Tool Offset or 0 as Offset.

Argument 8:

Specify the vision register number to return the part data from the queue.

Argument 9:

Specify the Numeric Register in which to return the status. The following statuses may be stored.

0: The information of the parts 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.

Example:

Make the following call to get a reachable part at a conveyor station whose Id=1 for the first pick. Timeout in 100 ms if none found. The reference Position Register is 57, the approach offset register is 56, and the approach offset is actually a tool offset. Return the part data in VR[1] and the status of the call in R[95].

```
CALL PKCSGETQINRG(CStn ID=1,Consec Flag=1,Timeout (ms)=100,
Do Pos Check,Ref Pos PR=57,Appr Ofst PR=56,Tool Offset,Offset VR=1,Stat Reg=95)
```

7.8.3.3 PKCSCHKPOS.PC

This program is used to check if a tracking position used by a conveyor station (for example, the Reference Position and the Approach position) is reachable. In other words, it is used to prevent the robot from going to positions that are not reachable. (Prevents MOTN-018 “Position not reachable”).

When you run this program, it does the following:

- It checks if a specified position is reachable or not. The Position Register id containing the position to be checked is passed in as an input argument. Optionally, you can specify an offset_preg which contains an offset value that should be applied to the position before doing the reachability check.
- In order to prevent the robot from moving to the position that is not reachable during tracking, it also checks if the position which is downstream from the specified position by the distance specified by Discard Line (the absolute value of Discard Line) is reachable or not.
- It returns the check result to a Numeric Register.

Argument 1:

Specify the Id of the conveyor station.

Argument 2:

Specify the Vision Register (VR) number used by the conveyor station for storing the part data from the queue.

Argument 3:

Specify the Position Register number (PR) used as the Reference Position.

Argument 4:

Specify the Position Register number (PR) used to store the Approach offset relative to the Reference Position.

If this is specified, then the reachability check will be made after applying the offset in this PR to the reference position specified in Argument 3.

This argument should be zero when you want to check the reachability of only the Reference Position.

This argument should be non-zero when you want to check the reachability of the approach position.

Argument 5:

If the approach offset used with Argument number 4 is stored as a tool_offset, then specify 1.

If the approach offset used with Argument number 4 is NOT stored as a tool_offset, but a offset, specify 0.

NOTE

When the argument guidance is supported, you select one of the following options.

1 Tool Offset

2 Offset

However, the approach offset is internally set as Tool Offset=1 and Offset=0. So if the argument guidance is not supported in your software version, specify 1 as Tool Offset or 0 as Offset.

Argument 6:

Specify the Numeric Register in which to return the status. If the status is zero, then the position is reachable. If non-zero, then it is not reachable.

Example:

To check if the Reference Position PR[57] (usually the pick or the place position) of a conveyor station whose Id=1 is reachable, and which uses VR[1] to store the part data from a PKCSGETQUE or PKCSGETQINRG call, use the macro as follows. Whether the position which is downstream from the Reference Position by the distance specified by Discard Line (the absolute value of Discard Line) is reachable or not is also checked. If the value of Register 95 in which the result is returned is zero, the position is reachable. Otherwise it is not reachable.

```
CALL PKCSCHKPOS(CStn ID=1,Offset VR=1,Ref Pos PR=57,Appr Ofst PR=0,Offset ,
Stat Reg=95)
```

To check if the Approach Position of a conveyor station whose Id=1 is reachable, and which uses VR[1] to store the part data from a PKCSGETQUE or PKCSGETQINRG call, and whose Reference PR is 57 and approach offset PR is 56, and the approach offset is specified as a tool_offset, use the macro as follows. Whether the position which is downstream from the Approach Position by the distance specified by Discard Line (the absolute value of Discard Line) is reachable or not is also checked. If the value of Register 95 in which the result is returned is zero, the position is reachable. Otherwise it is not reachable.

```
CALL PKCSCHKPOS(CStn ID=1,Offset VR=1,Ref Pos PR=57,Appr Ofst PR=56,
Tool Offset,Stat Reg=95)
```

7.8.3.4 PKOPPOSCHK.PC

This program is used to check if a non-tracking position is reachable.

When you run this program, it does the following:

- It checks if a specified position in a given Uframe and Utool is reachable or not. The position register id containing the position to be checked is passed in as an input argument. Optionally, you can specify an offset_preg which contains an offset value that should be applied to the position before doing the reachability check.
- It returns the check result to a Numeric Register.

Argument 1:

Specify the position register to check for reachability.

Argument 2:

Specify the group number of the robot. A position register can store position data for multiple motion groups. You need to indicate which group is used when the robot moves to the Position Register in Argument 1.

Argument 3:

Specify the Uframe used by the robot while moving to the position.

Argument 4:

Specify the UTool used by the robot while moving to the position.

Argument 5:

Specify the Numeric Register in which to return the status. If the status is zero, then the position is reachable. If non-zero, then it is not reachable.

Argument 6 (optional):

Specify the Position Register number (PR) used to store the offset to be applied to the position of Argument 1 before doing the reachability check.

Example:

To check if the position in PR[50] is reachable by a group 1 robot in Uframe #1 and UTool #1 after applying the offset in PR[51] specified as an Offset, use the macro as follows. If the value of Register 95 in which the result is returned is zero, the position is reachable. Otherwise it is not reachable.

```
CALL PKOPPOSCHK(Position PR=50,Group Num=1,Uframe Num=1,Tool Frame
Num=1,Stat Reg=95,Offset PR=51)
```

7.8.4 Sample Program

The sample program shown below is from Plug & Play standard TP program. The program is meant for picking from a conveyor.

Normally, you use PKCSGETQUE as shown in line 19. However, this program branches to use PKCSGETQUE when the “Pick only reachable parts” is disabled. The enable/disable flag is stored in R[115]. On the other hand, when the “Pick only reachable parts” is enabled, PKCSGETQINRG is executed which only returns reachable parts.

Also note the use of PKCSCHKPOS.PC macro on lines 52 and 65. This program checks if the tracking pick position is reachable again. You may need to use it because the parts are always moving and depending upon the conveyor speed and the robot time to move to pick from its current location (for example, from the place location), the part may be unreachable. If PKCSCHKPOS.PC macro returns a non-zero status, the program branches to LBL[1000] where the part is returned to the queue by doing a PKCSACKQUE with ack value of “Skip”(3). This is followed by returning to the top of the loop LBL[10: Pick loop] on line 7 to request a reachable part again.

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:
36:  LBL[50]
37:
38:  LBL[60]
39:  //-- Smart Pick
40:
41:
42:  LBL[70]
43:  //-- Check if a reject part
44:
45:
46:  LBL[80]
47:
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=1,Stat Reg=108)
53:  IF R[108:PosNotReach G1]<>0,JMP LBL[1000]
54:
55:  LBL[85]
56:  -- Move to approach
57:  L PR[57:Pk1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[1]
    Tool_Offset,PR[56:Pk1 Cv Ap Ofst]

```

```

58:  -- Flag on when Skip outbound true
59:  IF (F[R[102]]),JMP LBL[1000]
60:
61:  -- Set Pick/drop registers
62:  CALL PK_GR_GET_ENDZONE1(PickDropCounter=R[107:Counter ],
    Multiple=R[112:Pk1x PkMult],Quantity=R[111:Pk1x Qty],End Zone Reg=106)
63:  IF R[115:Pk1x UsePosChk]=0,JMP LBL[88]
64:  -- Check if tracking position is reachable
65:  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)
66:  IF R[108:PosNotReach G1]<>0,JMP LBL[1000]
67:
68:  LBL[88]
69:  -- Move to pick pos
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])
71:  -- Chk Skip Pick
72:  IF (F[R[102]]),JMP LBL[1000]
73:  -- Pick delay
74:  CALL PKGRGETPKDLY(Gripper ID=R[104:Gripper Id G1],PickDelay Reg=113)
75:  WAIT R[113]
76:  -- Check part pres
77:  CALL PKCSGETPPCHK(CStn ID=R[110:Pk1x StnId],PartPresChk Reg=116)
78:  IF R[116:Pk1x UsePPChk]=0,JMP LBL[90]
79:  -- PKGRCHKPP grip_id, operation,
80:  -- stat_reg, StartZone,
81:  -- EndZone (opt)
82:  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])
83:  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]
```

8 TROUBLESHOOTING

This chapter describes relatively common troubles in a tracking system using *iRPickTool* and their measures.

8.1 WHEN THE ROBOT DOES NOT PICK UP PARTS

Possible causes are described below.

- The conveyer station is disabled.
If the conveyer station is set to enable the skip station, information about parts is not provided for the robot of the conveyer station. Disable the skip station.
- The load balance is set improperly.
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 improper, pickup motion does not operate properly. Confirm that the load balance quota for picking up and putting down is set properly.
- The model ID is incorrect.
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.
In the case of the tray, confirm that the model ID of a cell matches the model ID of the load balance. When a model ID is specified in a 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.
- Parts are not detected by the vision process.
Confirm that parts are detected correctly with the *iRVision* runtime image.
- The sensor task is not started.
Confirm that PKWCSTART.PC (or PKSNSTART.PC) has been executed. You can go to the *iRPickTool* Production Status menu and check the status of the sensor. You may also check the part queues in the conveyor station page of the Status menu. Please see “7.2 CHECKING THE PRODUCTION STATUS” in detail.

8.2 WHEN PARTS ARE MISSED FREQUENTLY

Possible causes are described below.

- Processing by the robot is too slow.
Disable the hot start when it is enabled.



CAUTION

The *iRPickTool* tracking does not support the hot start.

Disable the Automatic Backup. Press F1[TYPE] key in FILE menu and select “Auto Backup”. Press F5 key “DISABLE” in “Automatic Backup” item.

- The motion of the robot is too slow.
If the motion of the robot is too slow, some of travelling parts cannot be picked. 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 selected.

- No part is detected by the vision process.
If parameters of the vision process are set inappropriately, no parts may be detected. Adjust the parameters of the GPM locator tool so that parts are detected correctly.
- Vision detection is too slow.
If vision processing is too slow as compared with the conveyor speed, the next vision process cannot be executed at the requested timing and parts pass by the field of view without being detected. Confirm that vision execution is not slow (the overrun count is less than 1) with the sensor menu of the iRPickTool production status screen and, if it is too slow, decrease the speed of the conveyor or enlarge the field of view.
When multiple sensors with vision systems are operated on one robot controller, distribute the sensors over multiple robot controllers.
- Vision detection time is too long.
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 4.9.2.2, "Pick program". When the third argument of PKCSGETQUE is set to a positive value, PKCSGETQUE works as follows, as described in "4.8.4.3 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 "4.9.2.2 Pick program") is executed repeatedly, thereby extending the time before completion of vision detection.
- The setting of the discard line is inappropriate.
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.
- The tracking area is set inappropriately.
Check that the set tracking area is appropriate.
- The model ID is incorrect.
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.

8.3 WHEN THE SAME PART IS PICKED UP TWICE

Possible causes are described below.

- Camera calibration is inappropriate.
Confirm that the pointer TCP used for camera calibration is correct. Confirm that the grid spacing and focal distance of calibration data are correct.
- The [Upstream Boundary] setting of the conveyer station is inappropriate.
Confirm that [Upstream Boundary] is not set within the field of view. If [Upstream Boundary] is set within the field of view, information of a part is delivered to the conveyer station even though the part is located within the field of view. Since the part is located within the field of view yet, it may be detected again. The robot cannot make an overlap check because the information of part was already delivered to the conveyer station, so the robot tries to pick up the same part twice.
- The [Overlap Tolerance] setting is inappropriate.
Confirm that this setting is not too small.
- A phantom part is falsely detected.
Confirm that a wrong part is not detected in its vicinity using Vision run-time window.

8.4 COMPENSATION DEVIATES WHEN A PART ROTATES

The tracking frame is not set correctly. See Subsection 4.12.2, "Adjustment of the Tracking Frame Error".

Or, the camera calibration is not correct.

8.5 COMPENSATION DEVIATES FOR A PART NEAR THE CONVEYOR EDGES

The calibration grid used for camera calibration may be too small to calculate the lens distortion correctly. If the lens distortion cannot be calculated correctly in camera calibration, the position of a part at the end of the field of view cannot be calculated correctly. Make the calibration grid larger so that it is fully covers the field of view, and calibrate it again.

If the [Part Z Height] setting specified by the vision process is incorrect, this symptom may appear. Set [Part Z Height] to the height of the feature to be detected in the tracking frame properly.

8.6 WHEN ROBOT MOTION IS NOT SMOOTH

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.

8.7 THE STOP CONVEYOR DOES NOT WORK

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.

8.8 IMAGE SNAP TIMING IS DELAYED

Vision processing may be too slow because the execution history of images etc. is stored in the memory card. Please uncheck [Enable Logging] in the [iRVision Configuration] screen of iRVision so as not to save execution history.

8.9 THE OVERRUN COUNT CONTINUES TO INCREASE

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.

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 iRVision 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.

8.10 WHEN SETTINGS CANNOT BE CHANGED DURING PRODUCTION

In consideration of safety, iRPickTool manage changes of setting items in the menus during production in the following three levels depending on the characteristics of setting items.

1. Setting items that can be changed. For example, you can change the load-balance quota at any time.
2. Setting items that can be changed but with a confirmation prompt. For example, if you wish to change the “tracking upstream boundary” of a conveyor station, you will be first prompted to confirm that you wish to make this change.
3. Setting item that cannot be changed. For example, you will not be allowed to change the “Trigger action” of a sensor during production run. A message box will be displayed that you cannot do so.

In the above levels, "3. Setting items that cannot be changed" cannot be changed during production. First stop production and then change settings. The following conditions are satisfied, the system is regarded as “during production”.

- Selecting “AUTO” mode with the mode switch
- Disabling the teach pendant.

APPENDIX

A

SYSTEM VARIABLE FILES

System variable file for *iRPickTool* is shown below.

SYSPKCFG.SV

IRPKTLCFG.SV

In this file, all *iRPickTool* data are saved

LNTKBCK.SV

In this file, all line tracking data are saved.

B SETTING A TEACHING PC

In *iRPickTool*, all settings can be made with the touch-panel-enabled *iPendant* 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 *iPendant*, it is not necessary to set a teaching PC.

A teaching PC is required only when settings of *iRPickTool* 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 *iRPickTool*. Select a 10BASE-T or 100BASE-T cable that meets the following specifications.

Cable	Twisted pair
Shield	Shielded

B.3 CONNECTION OF THE COMMUNICATION CABLE

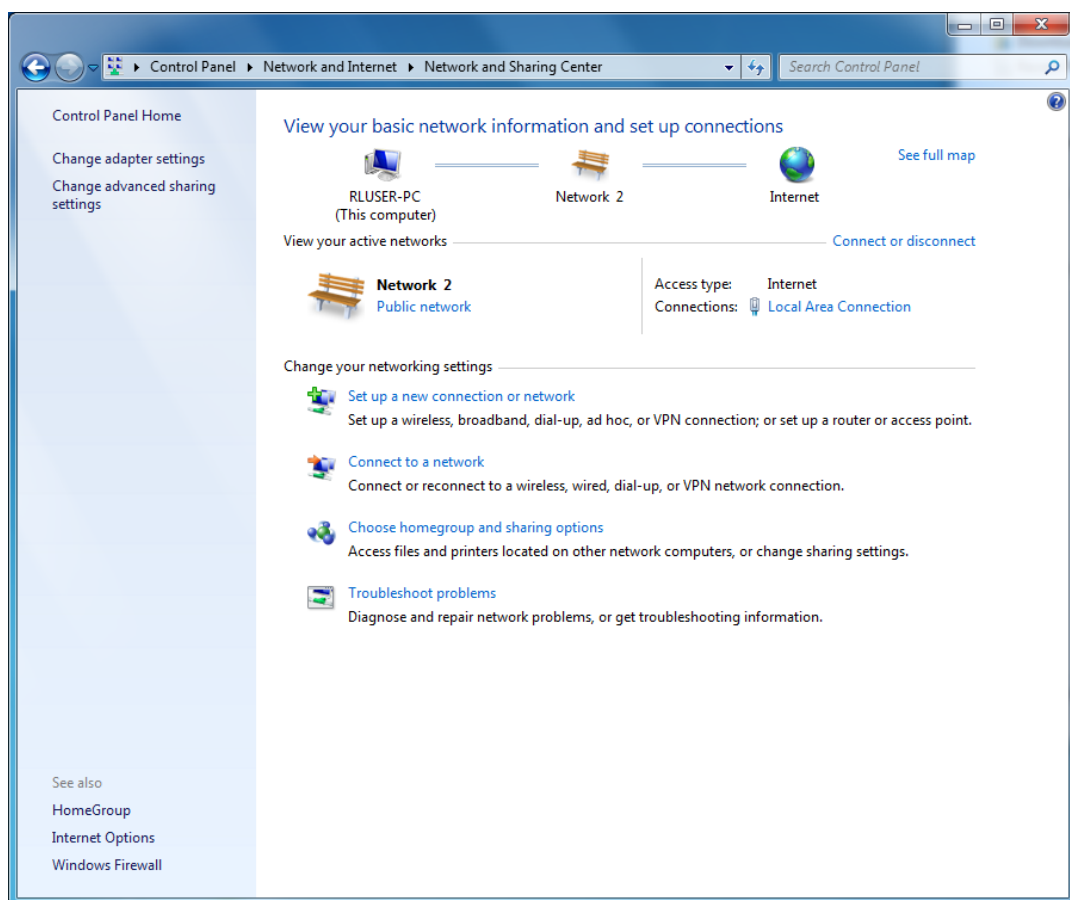
Connect the robot controller to a teach PC via the Ethernet cable. One end of the cable is connected to the Ethernet connector on the front face of the main board of the robot controller. The other end is connected to the connector of the PC marked with .

B.4 SETTING AN IP ADDRESS FOR THE PC

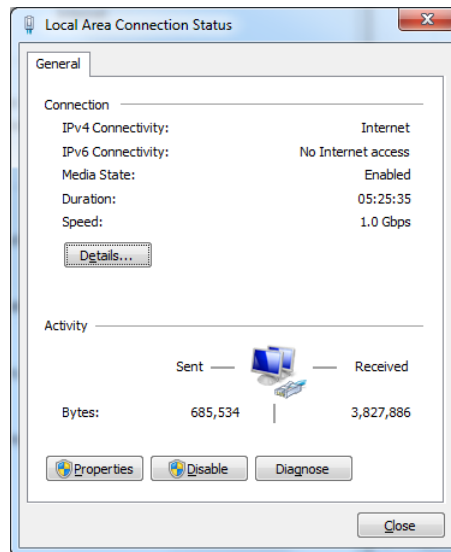
Set an IP address for the teach PC. When three robot controllers and the teach PC are connected and not connected to another network, the IP addresses as shown below can be set, for example.

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

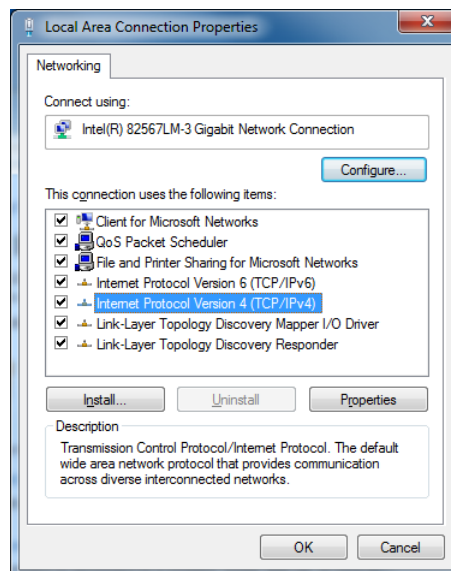
In the Control Panel, open [Network and Sharing Center].



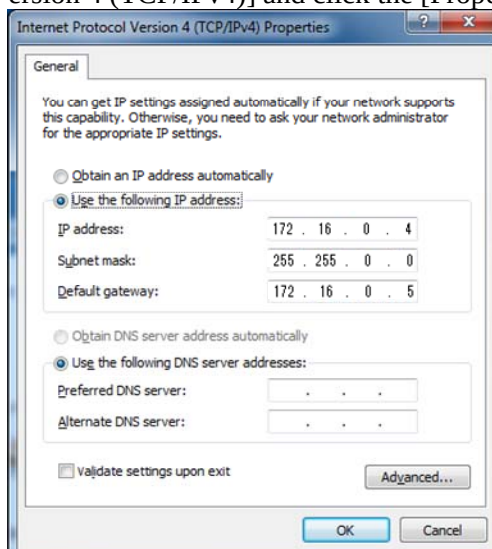
- 2 Click [Local Area Connections] under [View your active networks].



- 3 Click the [Properties] button.



- 4 Select [Internet Protocol Version 4 (TCP/IPv4)] and click the [Properties] button.

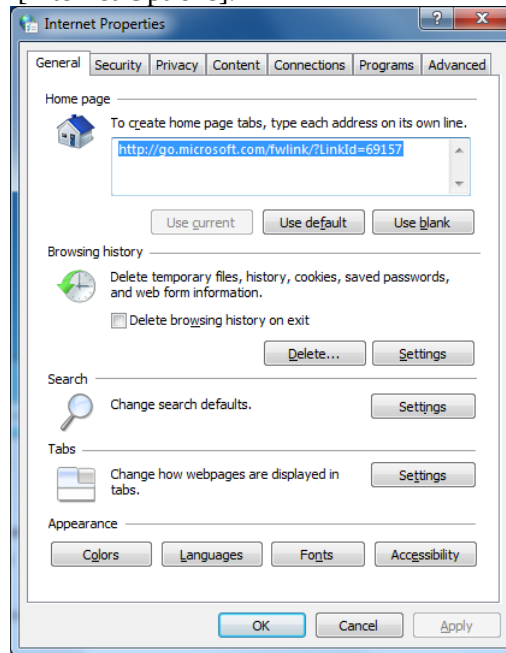


- 5 Check [Use the following IP address], enter values in [IP address], [Subnet mask], and [Default gateway], and click the [OK] button to close the window.

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.

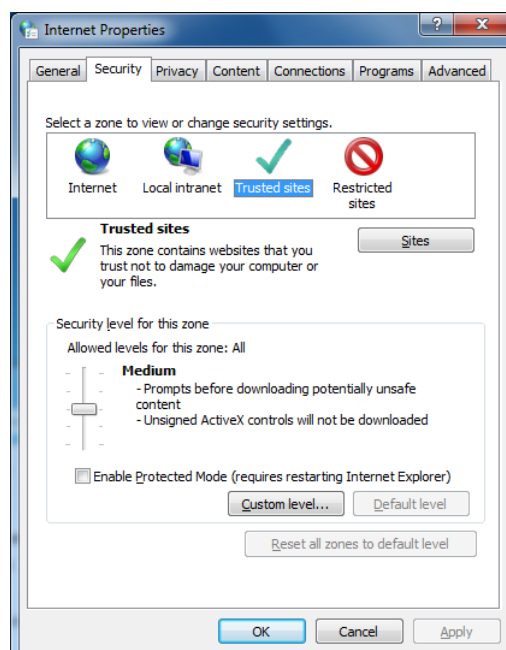
- 1 In the Control Panel, open [Internet Options].



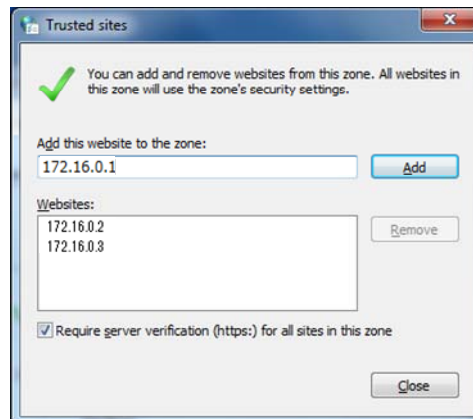
Trusted Site

The following procedure registers the robot controller as a trusted site.

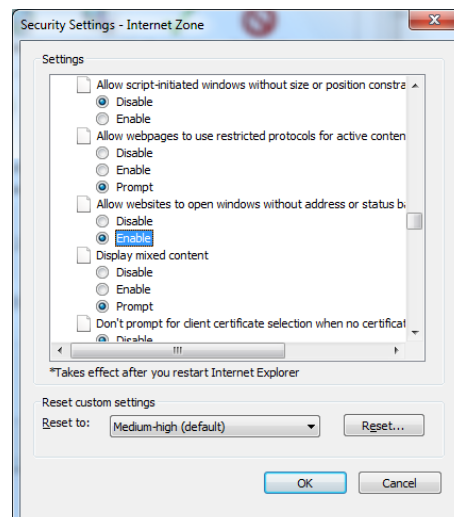
- 1 Select the [Security] tab.



- 2 Select [Trusted Site] as the zone and click the [Sites] button.



- 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”.

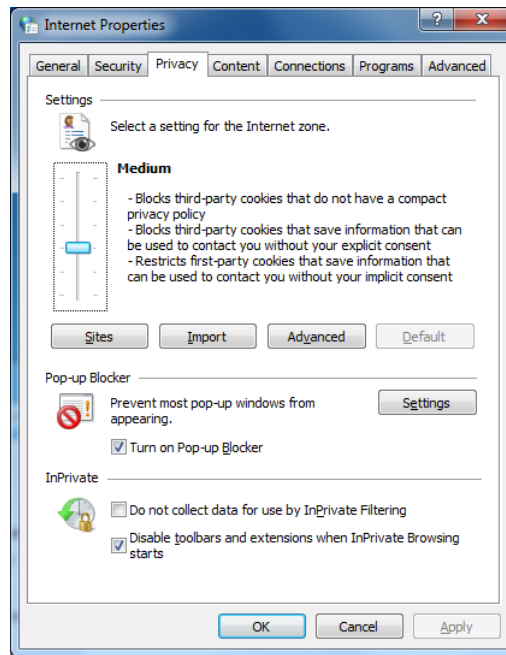


- 7 Press the [Close] button to close the dialog box.
- 8 Re-open the Internet Explorer.

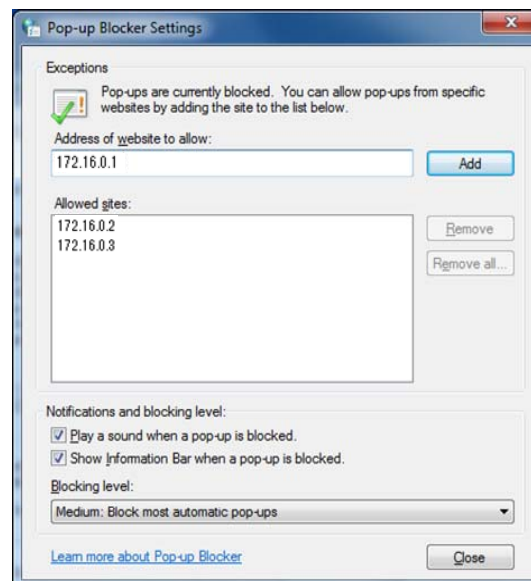
Preventing Popups from being Blocked

The following procedure prevents popups from being blocked during access to the homepage in the robot controller.

- 1 Select the [Privacy] tab.



- 2 Click the [Settings] button under [Pop-up Blocker].

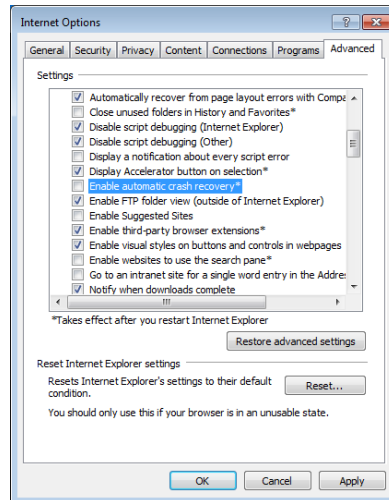


- 3 Enter the IP address of the robot controller in the [Address of Web site to allow] text box and click the [Add] button.
- 4 Press the [Close] button to close the dialog box.

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.

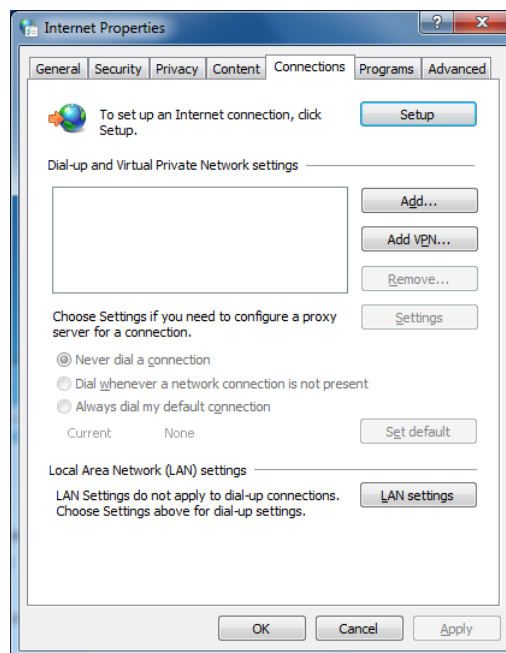


- 3 Click [OK]. When you changed the item, restart your PC.

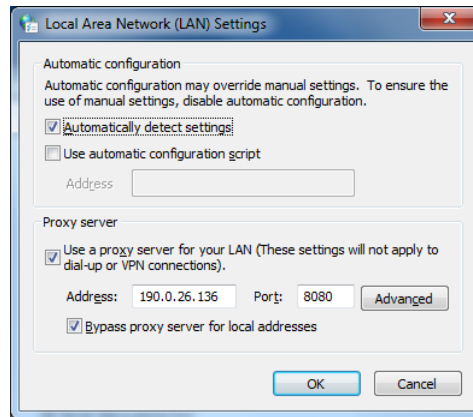
Proxy Server

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.

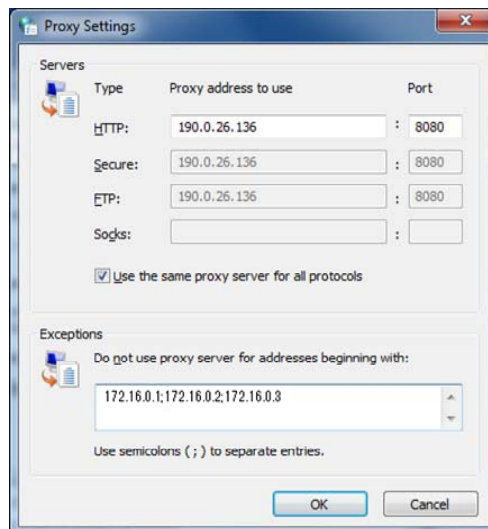
- 1 Select the [Connections] tab.



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].

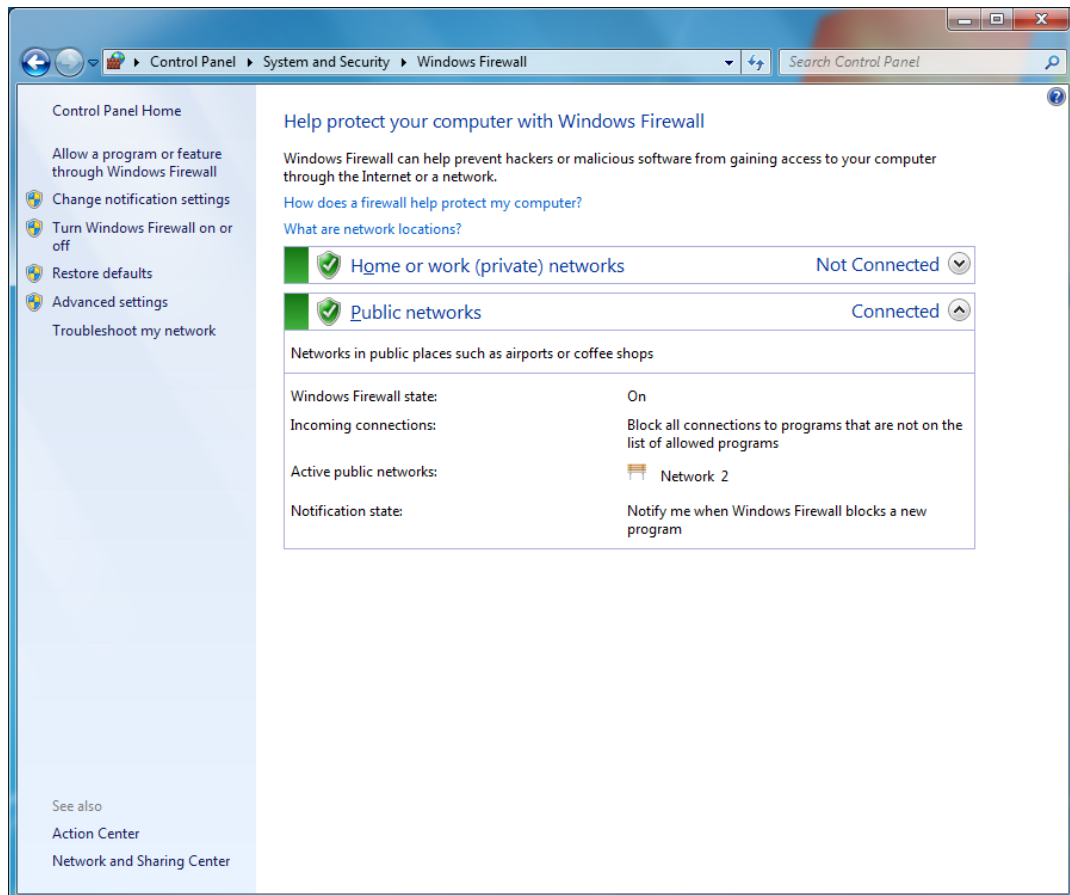


- 5 Enter the IP address of the robot controller in the [Exceptions] text box.
- 6 Click the [OK] button to close the dialog box.
- 7 Click the [OK] button to close the window.

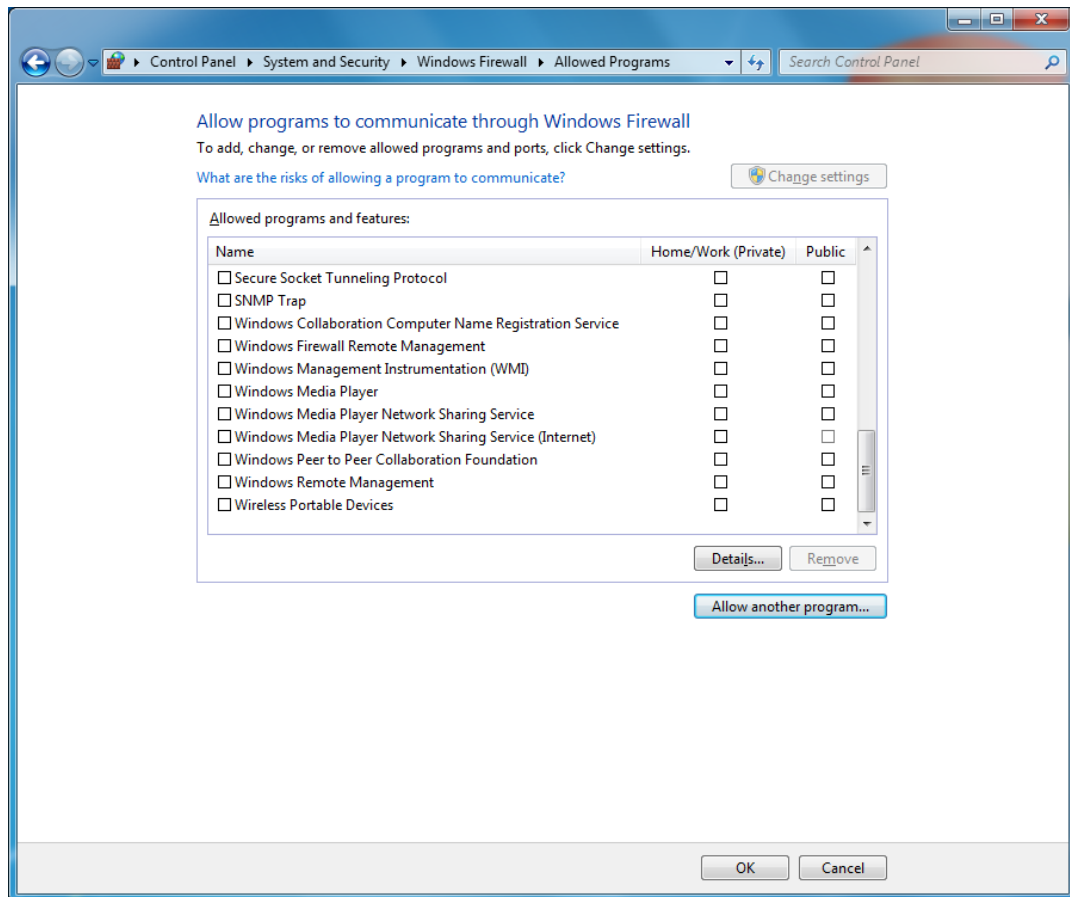
B.6 CHANGING SETTINGS OF WINDOWS FIREWALL

The following procedure sets Windows Firewall to prevent Windows Firewall from blocking communication with the robot controller.

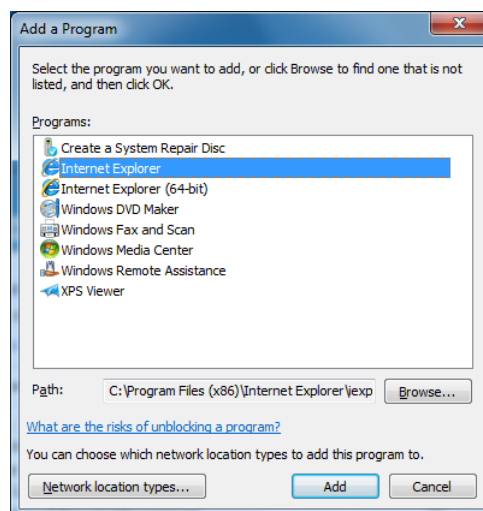
- 1 In the Control Panel, open [Windows Firewall].



- 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 iRPickTool, refer to Section B.9, "TROUBLESHOOTING" first.

B.7 OPENING *iRPickTool* SCREENS

Before opening the *iRPickTool* 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.

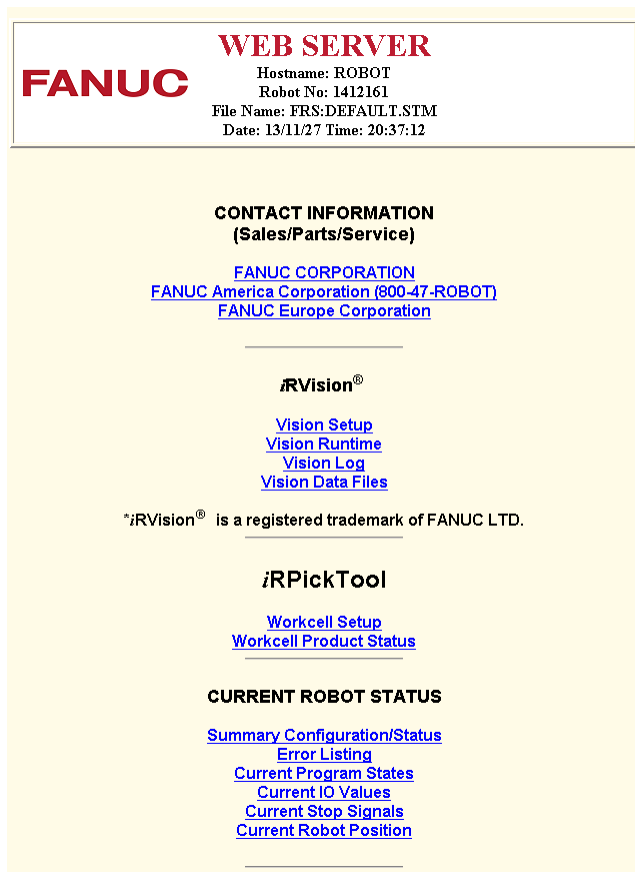


Fig. B.7

All robot controllers have their robot homepages. When the robot controller has *iRPickTool* options, the following links related to *iRPickTool* appear on the robot homepage.

Workcell Setup

Opens the edit screen for setting *iRPickTool*. For details, refer to Chapter 4, "SETUP OF THE BASIC FUNCTIONS" or Chapter 6, "SETUP PLUG & PLAY FUNCTIONS".

Workcell Product Status

Opens the edit screen for displaying the system production status set with *iRPickTool*. For details, see Section 7.2, "CHECKING THE PRODUCTION STATUS".

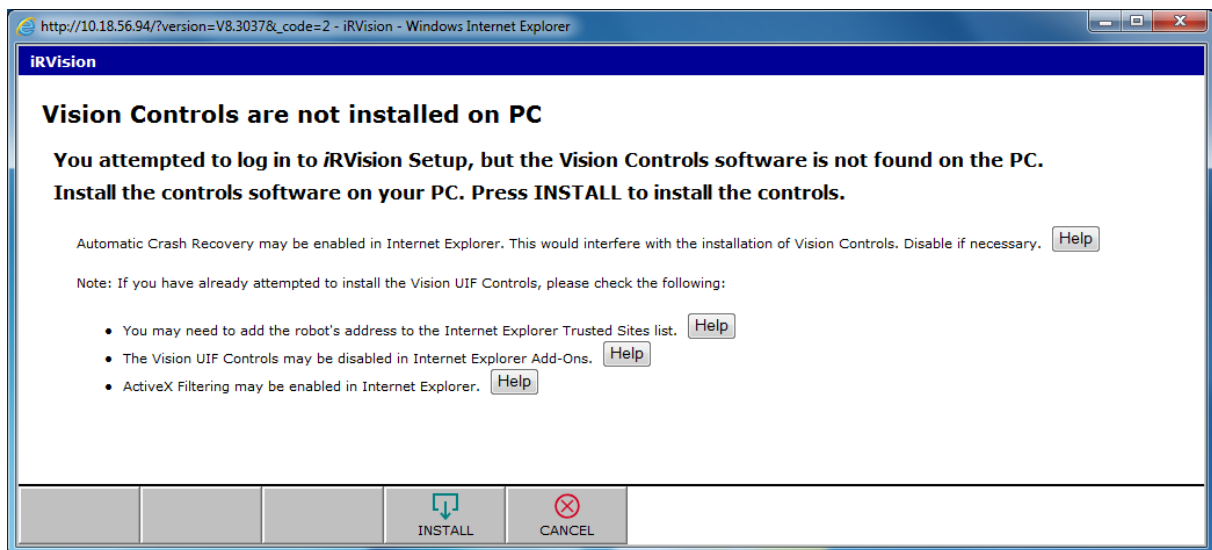
B.8 OPENING *iR*Vision SCREENS

In visual tracking, camera setting, camera calibration setting, and vision program setting need to be performed on the *iR*Vision setup screen. In *iR*PickTool queue managed tracking, these settings are not necessary.

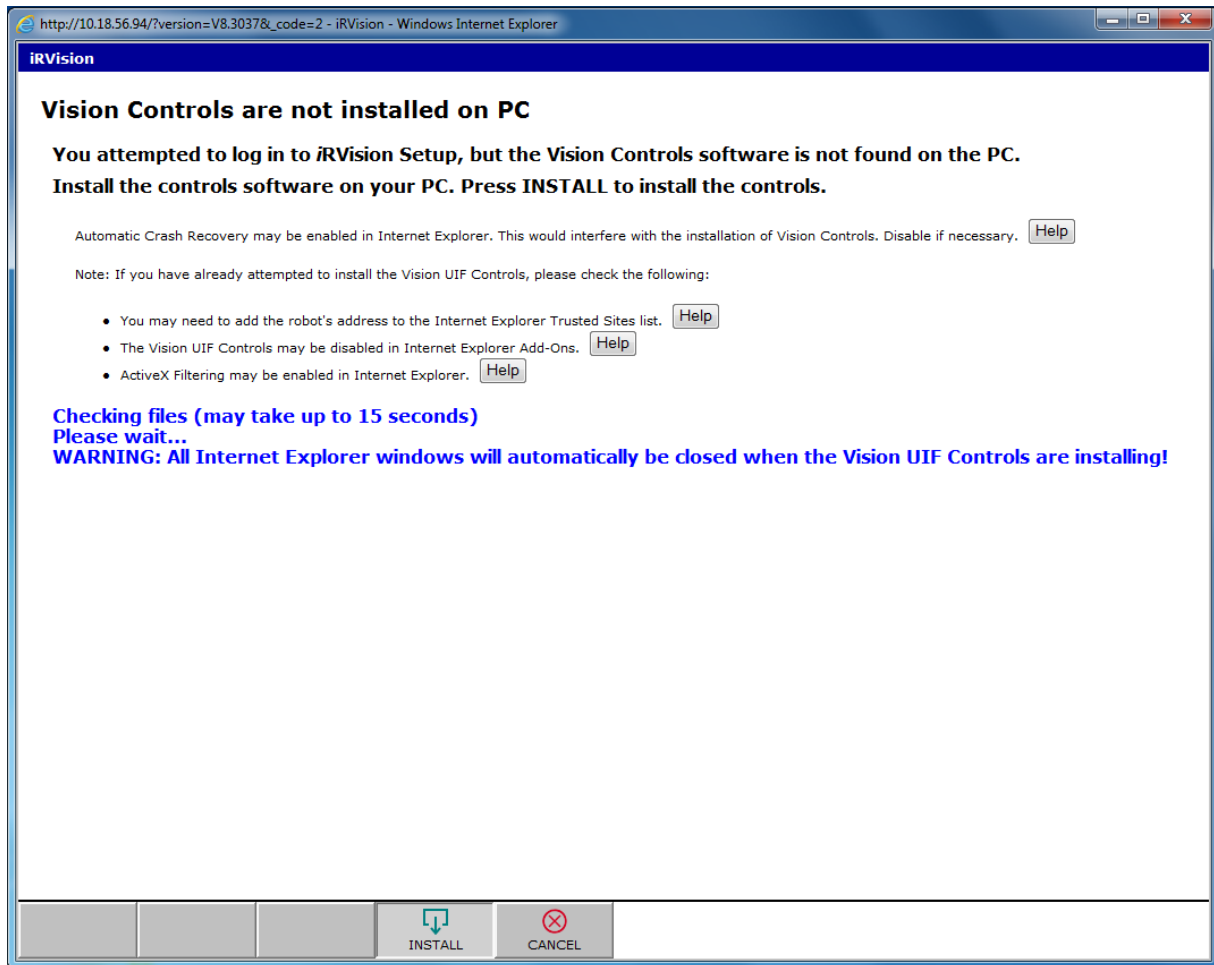
Installing Vision UIF Controls

Vision UIF Controls need to be installed in your PC to open the *iR*Vision 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 *iR*Vision on the screen in Fig. B.7. 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.



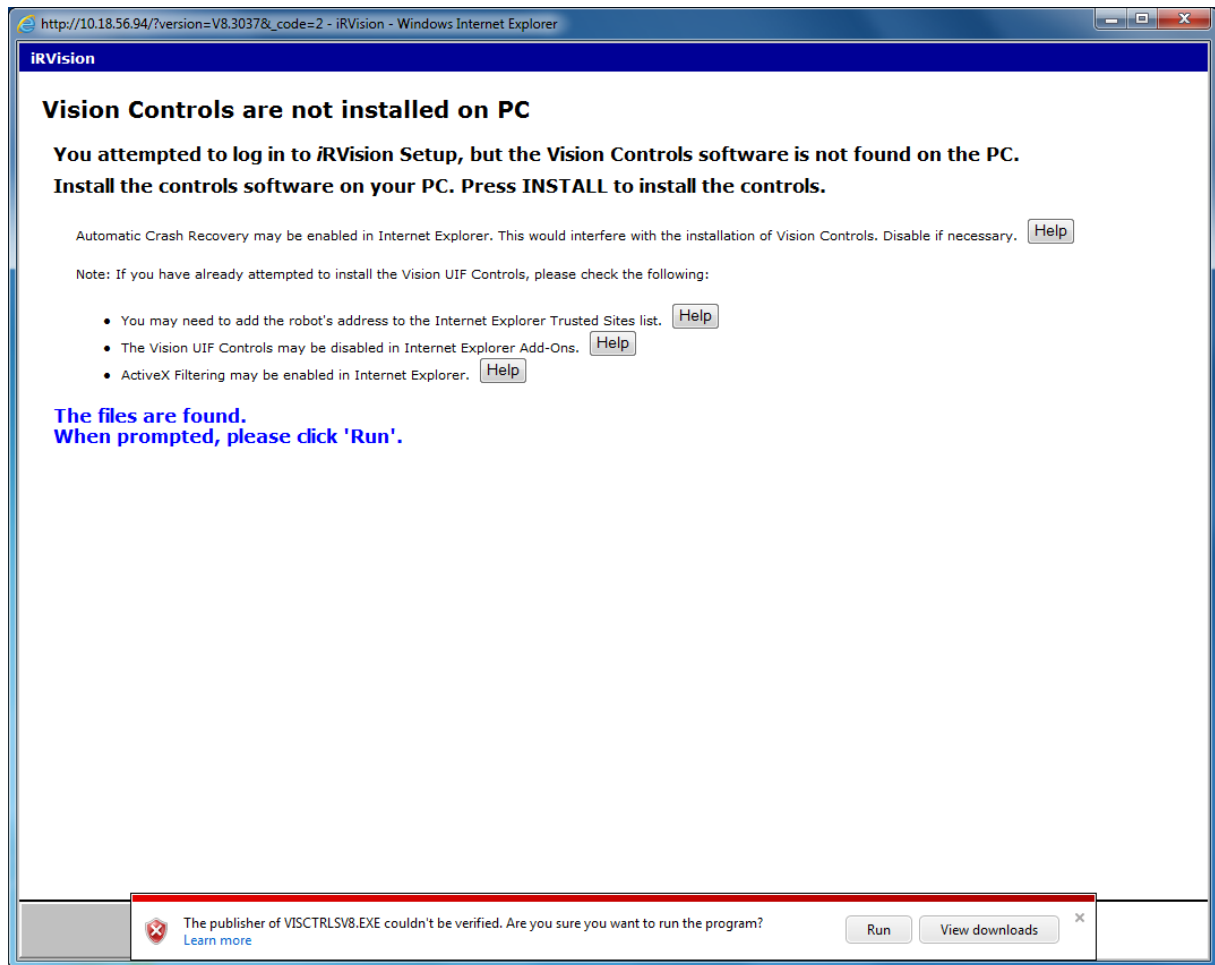
- 2 Click the [INSTALL] button.



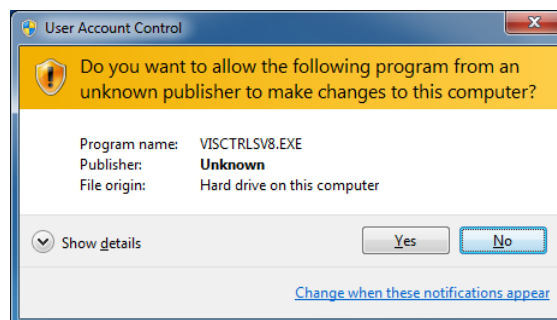
3 After a while, the following screen appears.



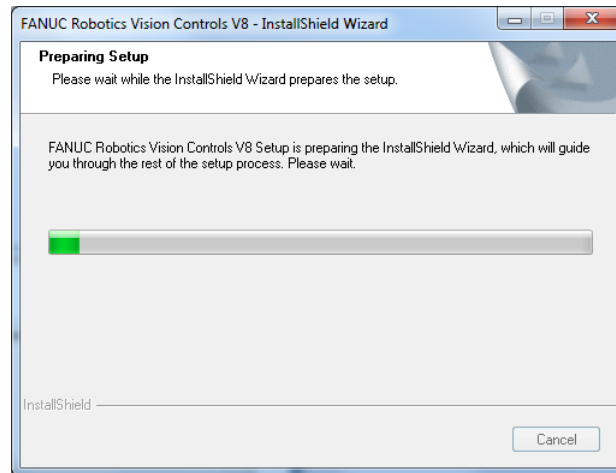
- 4 Click the [Run] button.
- 5 After a while, the following screen appears.



- 6 Click the [Run] button.
- 7 After a while, the following screen appears.



- 8 Click the [Yes] button.
- 9 Installation of Vision UIF Controls starts.



- 10 When the installation is completed, all windows of Internet Explorer are closed.
- 11 Start Internet Explorer again and open the homepage of the robot.

B.9 TROUBLESHOOTING

If there is a problem with *iRPickTool* setup on a PC, first check the following description.

The robot homepage cannot be opened.

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. Set it as described in B.5, "CHANGING SETTINGS OF INTERNET EXPLORER".

Links to *iR*Vision pages are not shown on the robot homepage.

Navigation links to *iR*Vision pages such as Vision Setup and Vision Runtime are not shown on the robot homepage. The problem can happen when a backup of a controller without *iR*Vision option is loaded to a controller with *iR*Vision option. Set the system variable \$VISION_CFG.\$PC_SETUP to TRUE and reopen the robot homepage.

When you click [Workcell Setup] or [Workcell Product Status] of *iRPickTool*, [Failed to login Vision Setup] appears.

Windows firewall may be set incorrectly. Set it as described in B.6, "CHANGING SETTINGS OF WINDOWS FIREWALL".

When you click [Workcell Setup] or [Workcell Product Status] of *iRPickTool*, [Enables popup on Internet Explorer] appears.

Internet Explorer may be set incorrectly. 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.

Internet Explorer may be set incorrectly. Set it as described in B.5, "CHANGING SETTINGS OF INTERNET EXPLORER".

When you click [Workcell Setup] or [Workcell Product Status] of *iRPickTool*, [70: Cannot write] appears.

Internet Explorer may be set incorrectly. Set it as described in B.5, "CHANGING SETTINGS OF INTERNET EXPLORER".

Even when you click [Workcell Setup] or [Workcell Product Status] of iRPickTool, no window opens.

Verify the following points.

Setting Windows Firewall

Windows firewall may be set incorrectly. Set it as described in B.6, "CHANGING SETTINGS OF WINDOWS FIREWALL".

Security Software

If security software is installed in your PC, communication may be blocked by the security software. Disable the security software.

Internet Explorer

An issue that Internet Explorer fails to open a new window on Windows 7 is reported. 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

If the setting of Trusted Sites in Internet Explorer is incorrect, you may not be able to install the Vision UIF Controls. Refer to Subsection A.1.7, “Modifying Settings of Internet Explorer”, and make sure that of the robot IP address is in Trusted Sites.

Add-ons

If the Vision UIF Controls is disabled in [Manage add-ons] of Internet Explorer, the problem can occur. 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

If the ActiveX Filtering is enabled in Internet Explorer, the problem can occur. 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.

Windows firewall may be set incorrectly. 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 *iRPickTool*, Internet Explorer is closed with the message [A problem occurred].

Communication with the robot controller may not be performed normally due to the influence of the add-on software of Internet Explorer.

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 *iRPickTool* setup operation can be performed normally. If no problem arises, enable the disabled add-ons one at a time while checking that the *iRPickTool* setup operation is not affected.

No image is displayed on the *iR*Vision teach screen.

Verify the following points.

Restarting PC

Cases restarting PC resolves the problem have been reported. Restart your PC and re-open the *iR*Vision setup screen.

Administrator password

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.

Internet Information Server

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. Disable the Worldwide Web Publishing Service.

Temporary Internet Files

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 cached by IE, and the cached page file is used for Vision Setup of another controller. 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.

When Internet Information Services (IIS) is enabled, communication with the robot controller may not be performed correctly. 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

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. Refer to Subsection A.1.7, “Modifying Settings of Internet Explorer”, and make sure that of the robot IP address is in Trusted Sites.

Address Bar, Status Bar

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. Refer to Subsection A.1.7, “Modifying 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.

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. Refer to Subsection A.1.7, “Modifying 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.

When Internet Information Services (IIS) is enabled, communication with the robot controller may not be performed correctly. 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

Confirm that the "iRVision 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.

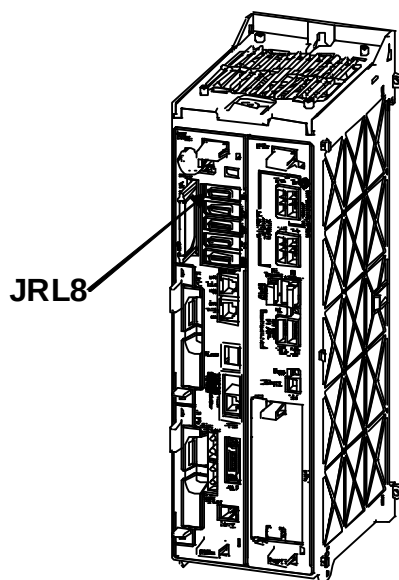
**CAUTION**

The HDI signals cannot be used as general-purpose DIs.

C.1 HARDWARE CONNECTION OF R-30iB

R-30iB supports up to 5 High Speed Digital Input (HDI) #1 to #5, located on the JRL8 connector of the controller.

R-30iB



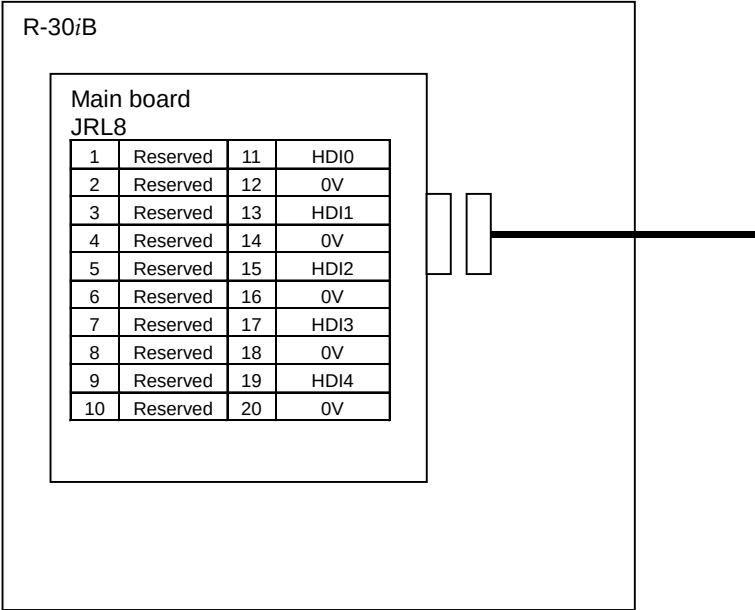


Fig. C.1 (a) R-30iB HDI interface

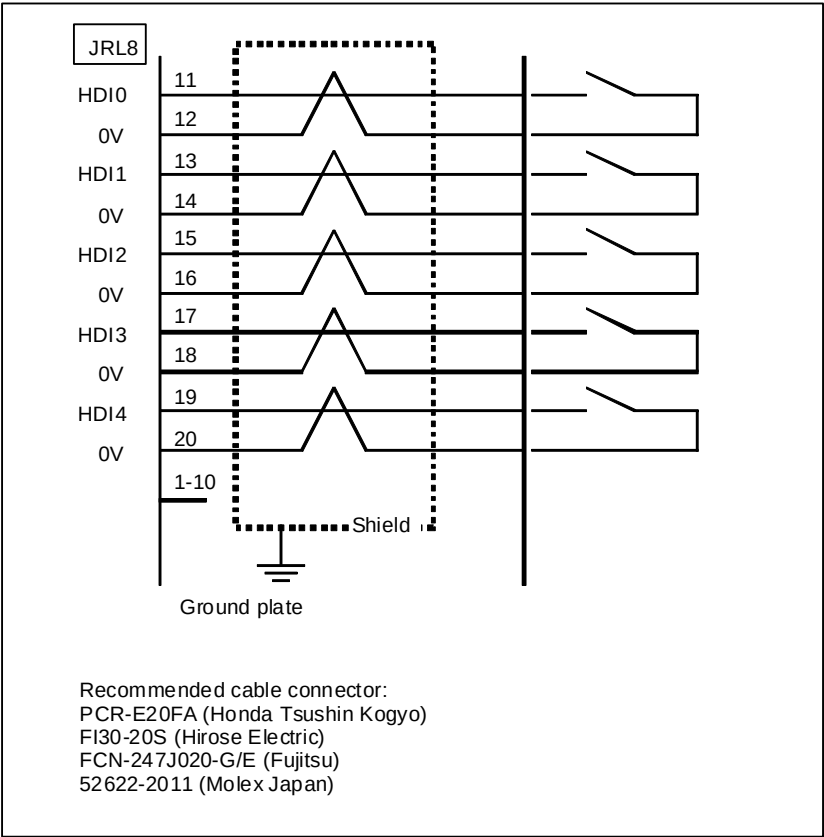


Fig. C.1 (b) R-30iB HDI cable connections

C.2 HARDWARE CONNECTION OF R-30iB Mate

R-30iB Mate supports up to 2 High Speed Digital Input (HDI) #1 to #2, located on the CRL3 connector of the controller.

R-30iB Mate

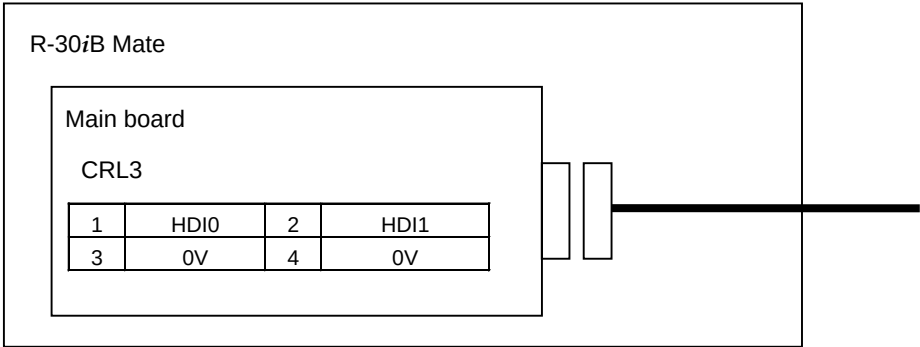
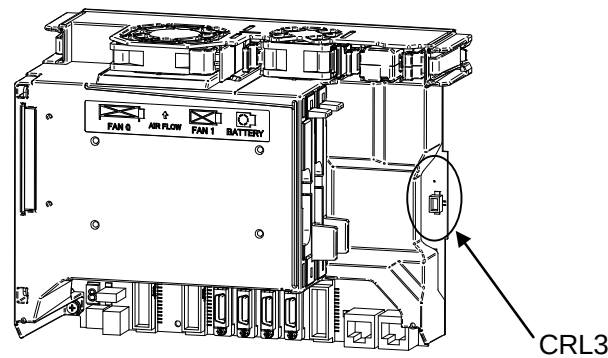


Fig. C.2 (a) R-30iB Mate HDI interface

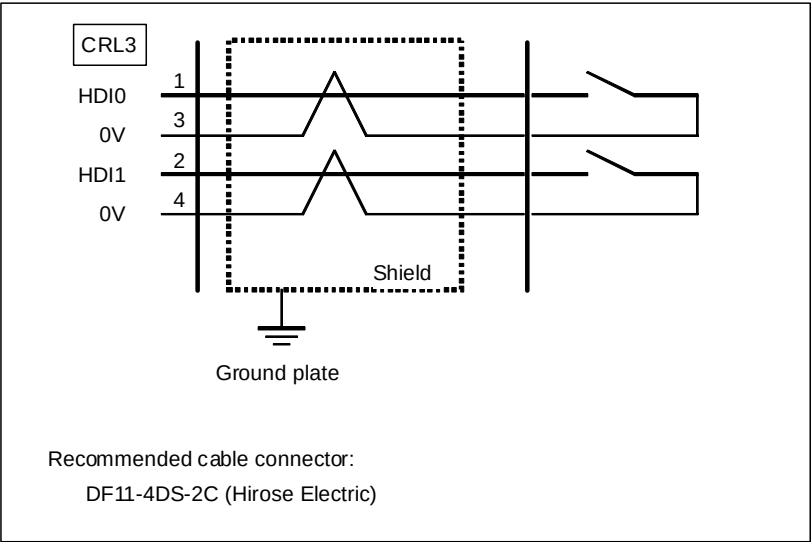


Fig. C.2 (b) R-30iB Mate HDI cable connections

NOTE

In R-30iB Mate controller, HDI interface is not available with Main board A (A20B-8200-0790) or Main board B (A20B-8200-0791). Main board C (A20B-8200-0792) is required to enable HDI for Line Tracking feature with R-30iB Mate controller. See Table C.2 (a) for detail.

Table C.2 Requirements for R-30iB Mate Main board (HDI interface)

Main board	Board Specification	Available / Unavailable for HDI interface	Comments
Main board A	A20B-8200-0790	Unavailable	The Main board in the left column is included in the following order specification. A05B-2650-H001, A05B-2661-H001 (R-30iB Mate) A05B-2655-H001, A05B-2662-H001 (R-30iB Mate (Open Air))
Main board B	A20B-8200-0791	Unavailable	The Main board in the left column is included in the following order specification. A05B-2650-H002, A05B-2661-H002 (R-30iB Mate) A05B-2655-H002, A05B-2662-H002 (R-30iB Mate (Open Air))
Main board C	A20B-8200-0792	Available	The Main board in the left column is included in the following order specification. A05B-2650-H003, A05B-2661-H003 (R-30iB Mate) A05B-2655-H003, A05B-2662-H003 (R-30iB Mate (Open Air))

C.3 INPUT SIGNAL

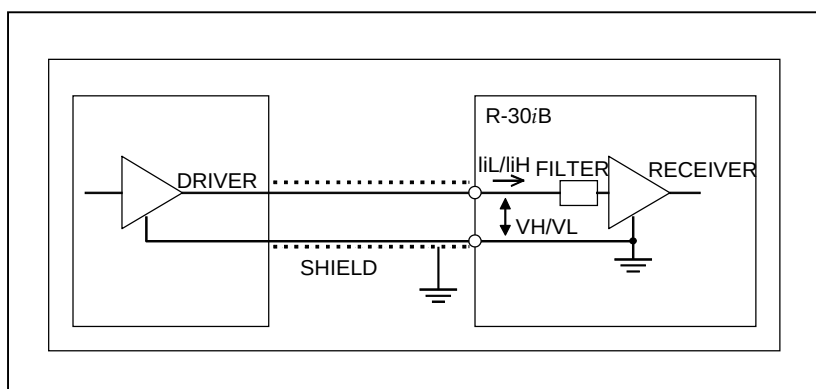


Fig. C.3 HDI input signal

Absolute maximum rating

Input voltage range V_{in} : -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	$V_{in}=5\text{ V}$
		11 max	mA	$V_{in} = 10\text{ V}$
Low level input current	IiL	-8.0 max	mA	$V_{in} = 0\text{ V}$
Input signal pulse duration		20 min	ms	
Input signal delay or variations		0.02(max)	ms	

NOTE

- 1 The plus (+) sign of liH/liL represents the direction of flow into the receiver. The minus (-) sign of liH/liL 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.4 SETUP OF HDI

C.4.1 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.4.2 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.3)
- R-30iB supports up to 5 High Speed Digital Input (HDI) #1 to #5, located on the JRL8 connector of the controller.
- R-30iB Mate supports up to 2 High Speed Digital Input (HDI) #1 to #2, located on the CRL3 connector of the controller.

Steps

1. Press the [MENU] key.
2. Select [0 NEXT].
3. Select [6 SYSTEM].
4. Press F1 key, [TYPE] and select [2 Variables].
5. Move the cursor to the following variables and check they are set to the following values. When High Speed Scanning option has been loaded, they are set to the following values by default.
 - \$HSLTENBL = TRUE
 - \$LNCFG.\$HSDI_ENABLE = TRUE
6. After changing above variables, turn off the controller, and then turn it on again to accept the new setting.
7. Use the Encoder Setup Menu to input used HDI port id to HDI port id.
8. Turn off the controller, and then turn it on again to accept the new setting.

SETUP Encoders	
	9/9
Encoder Number : 1	
1 Encoder Axis :	1
2 Encoder Type :	Serial INC
3 Encoder Enable :	OFF
Current Count (cnts) :	0
4 Multiplier (ITP/update) :	1
5 Average (updates) :	1
6 Stop Threshold (cnt/updt) :	0
7 Simulate :	Enable : OFF
8 Rate (cnt/updt) :	0
9 HDI port Id :	1

	[TYPE]		ENCODER			

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

SERVO CONVEYOR LINE TRACKING FUNCTION

D.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.

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.

NOTE

On 7DC3 series and later, "Continuous" has been added to Servo Conveyor type. It is possible to use it for moving a conveyor at constant speed. If you would like to move and stop a conveyor every constant distance again and again, please select "Index".

D.2 SETUP

Please set up Servo Conveyor Line Tracking system by the following step.

- D.2.1 Independent Extended Axis Setup
- D.2.2 Servo Conveyor Setup
- D.2.3 TP Program for Servo Conveyor
- D.2.4 Tracking Schedule Setup
- D.2.5 Example of TP Program

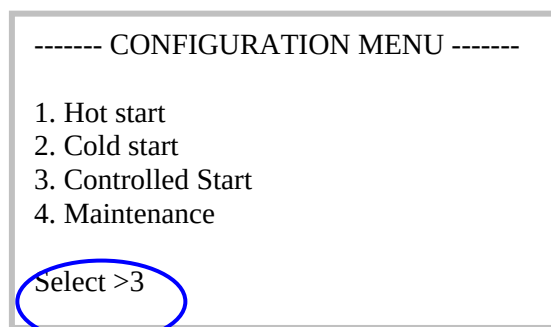
D.2.1 Independent Extended Axis Setup

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.

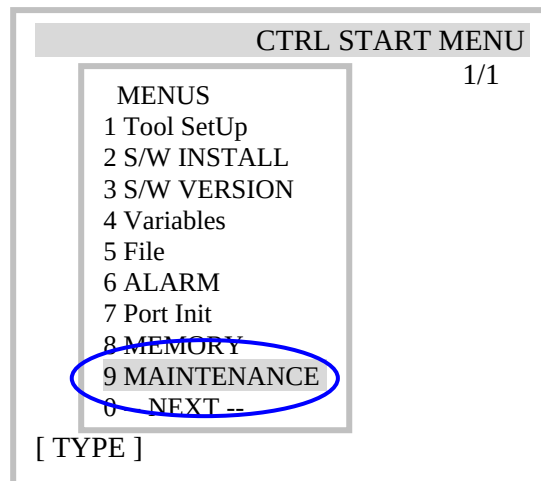


- 2 Select “3. Controlled Start” and press the [ENTER] key. After a while a Control Start Menu appears.

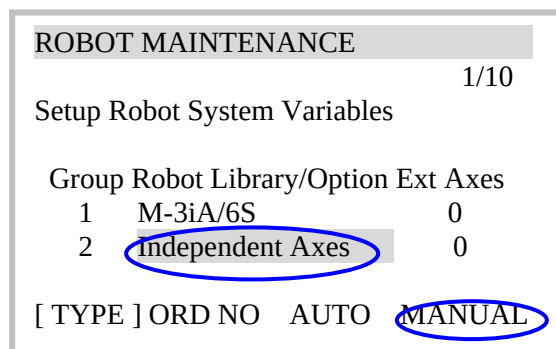
Procedure: Robot Maintenance

Step

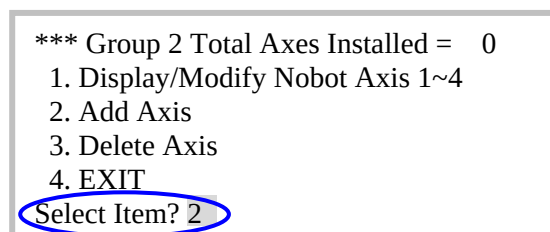
- 1 Press the [MENU] key to display the screen menu. Select [9 MAINTENANCE] and press the [ENTER] key. ROBOT MAINTENANCE screen appears.



- 2 Select [Independent Axes] and press F4 [MANUAL] key.



- 3 “Independent Axes Setup Menu” appears. If you add axis, please select [2: 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].



NOTE

Please refer to the mechanical specification for the following procedure.

- 4 Select a servo motor which is used as Independent Axis. Select [1 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 iS2/4000i, 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 on 7DC2 series or 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.

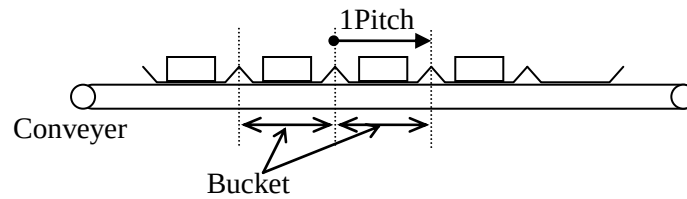


Fig. D.2.1(a) Bucket and Pitch

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.

$$\text{GearRatio} = \frac{M}{N \times \text{Number of Buckets}}$$

- 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.

$$\text{GearRatio} = \frac{\text{Rotor Input Gear Teeth} \times \text{Conveyer Belt Teeth}}{\text{Motor Gear Teeth} \times \text{Rotor Output Gear Teeth} \times \text{Number of Flight}}$$

- (b) In the case of setting a servo conveyor as “Continuous” on 7DC3 and later
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.

$$\text{GearRatio} = \frac{M \times \text{Belt Section Teeth}}{N \times \text{Belt Teeth of the conveyor}}$$

- 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.

$$\text{GearRatio} = \frac{\text{Rotor Input Gear Teeth} \times \text{Belt Section Teeth}}{\text{Motor Gear Teeth} \times \text{Rotor Output Gear Teeth}}$$

- 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.

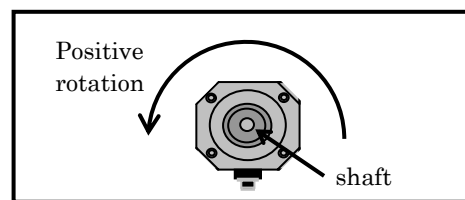


Fig. D.2.1(b) Positive rotation of a motor

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 LIMIT--
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 --
$$\text{Load Ratio} = \frac{\text{Load Inertia} + \text{Motor Inertia}}{\text{Motor Inertia}}$$

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 “4 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.

```
ROBOT MAINTENANCE
1/10
Setup Robot Sys
Group Robot L
1 M-3iA/
2 Indepen
[ TYPE ] ORD NO  AUTO  MANUAL

FUNCTION
1 START(COLD)
2 RESTORE/BACKUP
3 PRINT SCREEN
4 UNSIM ALL I/O
```

D.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

On 7DC2 series, the servo conveyor line tracking function uses flags from 131 to 138. You must not use these flags. A flag is an internal I/O port that can be read and set.

On 7DC3 series and later, it is possible to specify a flag to use on Servo Conveyor Setup display.

Procedure: Servo Conveyor Setup (7DC2 series)

Step

- 1 Press the [MENU] key.
- 2 Select [6 SETUP].
- 3 Press F1 [TYPE] key.
- 4 Select "Indexers". You will see a screen as following.

SETUP Indexers		2/7
Indexer Number: 1		
1	Encoder Number:	1
2	Indexer type	FANUC
3	Index Advance Trigger DI:	1
4	Delay move after trig(ms):	0
5	Indexer Ready DO:	2
6	Tracking Schedule:	1
7	Create Index program:	Detail
Encoder Scale (cnt/mm):		524288
[]	[TYPE]	[DETAIL]
[]	[INDEX]	[CHOICE]

- 5 Move the cursor to [1 Encoder Number] and enter the number of the encoder that is used as servo conveyor.
- 6 Move the cursor to [2 Indexer Type] and select "FANUC".
- 7 Move the cursor to [3 Index Advance Trigger DI] and enter the index of DI is used for moving the servo conveyor.
- 8 Move the cursor to [4 Delay move after trig (ms)] and enter the value of delay time until the servo conveyor starts after trigger.
- 9 Move the cursor to [5 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).
- 10 Move the cursor to [6 Tracking Schedule] and enter the value of the tracking schedule that is used for servo conveyor.
- 11 Move the cursor to [2 Indexer Type] 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.

SETUP Indexer axis	
	1/7
Indexer 1: FANUC motor UNINIT	
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	1
7 Index Distance (mm)	10.000
EXEC	

- 12 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.
- 13 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.

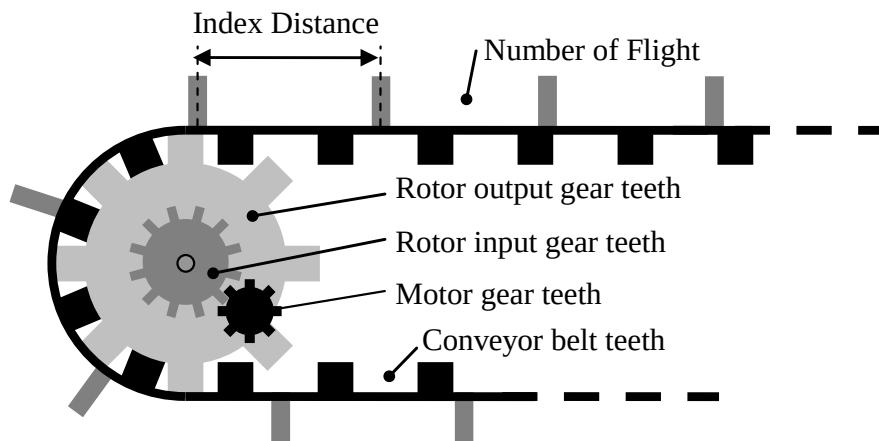


Fig. D.2.2(a) Relationship between motor, rotor and conveyor

- 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. Each of "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]
- 14 Move the cursor to [7 Index Distance (mm)] and the distance a one pitch on the servo conveyor.
 - 15 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.

Procedure: Servo Conveyor Setup (7DC3 series and later)

Step

- 1 Press the [MENU] key.
- 2 Select [6 SETUP].
- 3 Press F1 [TYPE] key.
- 4 Select "Indexers". You will see a screen as following.

FANUC Conveyor:		1/4
FANUC Conveyor: 1		
1	Encoder Number:	1
2	Conveyor Type:	Index
3	Tracking Schedule:	1
	Encoder Scale (cnt/mm):	524288
4	program and speed control:	Detail
[]	[TYPE]	[]

- 5 Move the cursor to [1 Encoder Number] and enter the number of the encoder that is used as servo conveyor.
- 6 Move the cursor to [2 Conveyor Type] and select "Index" or "Continuous".
 Index: If you would like to move and stop a conveyor every constant distance again and again, please select "Index".
 Continuous: If you would like to move a conveyor continuously at constant speed, please select "Continuous".

FANUC Conveyor:		2/4
FANUC Conveyor: 1		
1	Encoder Number:	1
1	Conveyor Type:	Index
2	Tracking Schedule:	1
3	Encoder Scale (cnt/mm):	524288
4	program and speed control:	Detail
5		
[]	[TYPE]	[]

- 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 Conveyor 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 conveyor type.

In the case of that Conveyor 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.

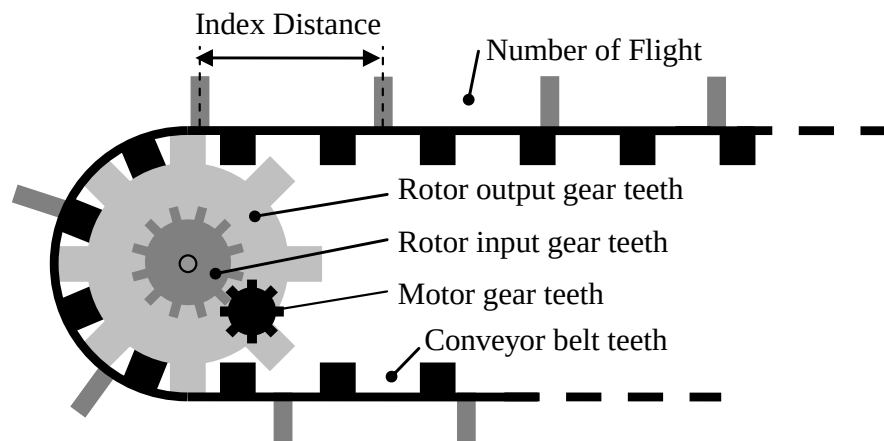


Fig. D.2.2(b) Relationship between motor, rotor and conveyor

- 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 "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.

- After the above setup, press F2 [EXEC] key and be sure to repower.

By the above setting, Continuous Rotation setup of the conveyor and Encoder setup of the specified encoder are also done.

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.

- 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.

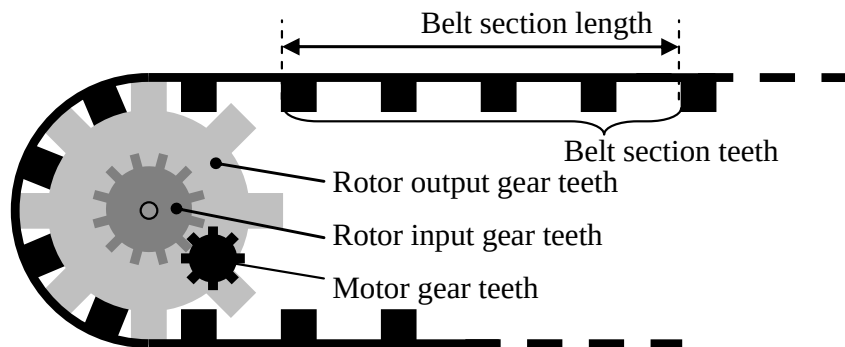


Fig. D.2.2(c) Relationship between motor, rotor and conveyor

- 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.

D.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 (7DC2 series)

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.

SETUP Indexers		7/7
Indexer Number: 1		
1	Encoder Number:	1
2	Indexer type	FANUC
3	Index Advance Trigger DI:	1
4	Delay move after trig(ms):	0
5	Indexer Ready DO:	2
6	Tracking Schedule:	1
7	Create Index program:	Detail
Encoder Scale (cnt/mm):		524288
[TYPE] DETAIL INDEX		

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 Motn		1/4
FANUC Indexer: 1		
1	Index Speed (part/min)	100
2	Index Dwell (ms):	0
3	Indexer Register start	60
4	Generate Index program:	

- 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:

$$\text{Movement Time of Conveyor per Pitch [ms]} = (60 \times 1000 / \text{Index Speed}) - \text{Index Dwell}$$

- Index Dwell is “-1”:

$$\text{Movement Time of Conveyor per Pitch [ms]} = (2 \times 60 \times 1000 / \text{Index Speed})$$

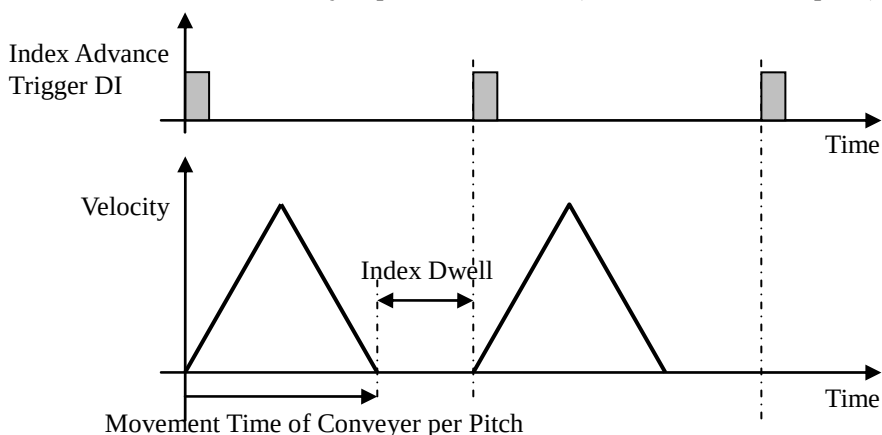


Fig. D.2.3(a) Speed pattern if Index Dwell is “0” and over

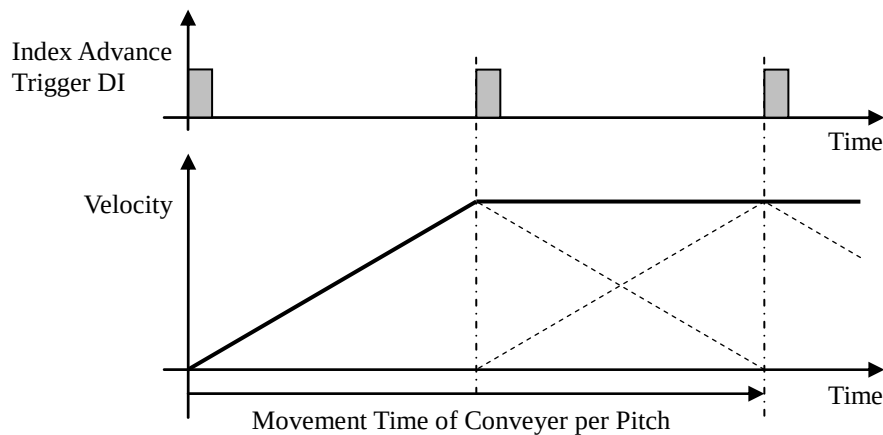


Fig. D.2.3(b) Speed pattern if Index dwell is "-1"

- 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 INDYG*.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 line 11 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 "Index" Conveyor (7DC3 and later)

Step

NOTE

Before generating indexer program, please set max payload which is used in an indexer tracking motion on MOTION/PAYLOAD SET display.

- 1 Move the cursor to [4 Program and speed control] in "SETUP Indexers" and press F2 [DETAIL] key or ENTER key.

FANUC Conveyor:		4/4
FANUC Conveyor: 1		
1	Encoder Number:	1
2	Conveyor Type:	Index
3	Tracking Schedule:	1
	Encoder Scale (cnt/mm):	524288
4	program and speed control:	Detail
	[TYPE]	[DETAIL] [CNVY]

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 PREV key.

SETUP Indexer Program:		1/4
FANUC Indexer: 1		
1	Index Speed (part/min)	100
2	Index Dwell (ms):	0
3	Indexer Register start	60
4	Generate Index program:	

- 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:

$$\text{Movement Time of Conveyor per Pitch [ms]} = (60 \times 1000 / \text{Index Speed}) - \text{Index Dwell}$$

- Index Dwell is "-1":

$$\text{Movement Time of Conveyor per Pitch [ms]} = (2 \times 60 \times 1000 / \text{Index Speed})$$

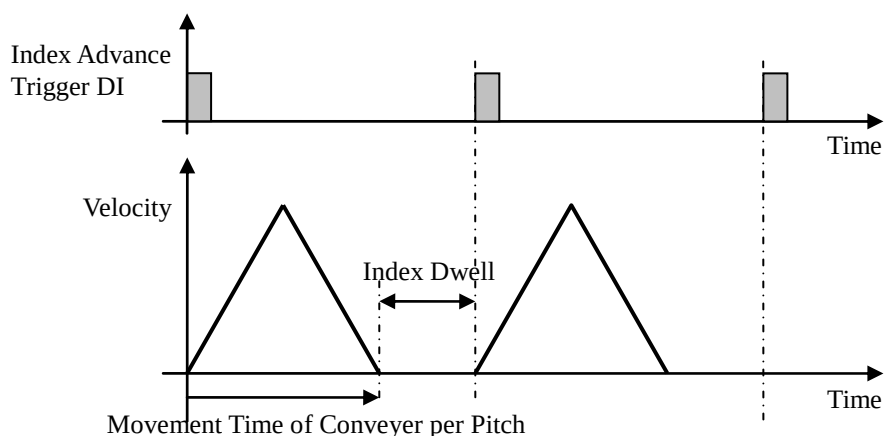


Fig. D.2.3(c) Speed pattern if Index Dwell is "0" and over

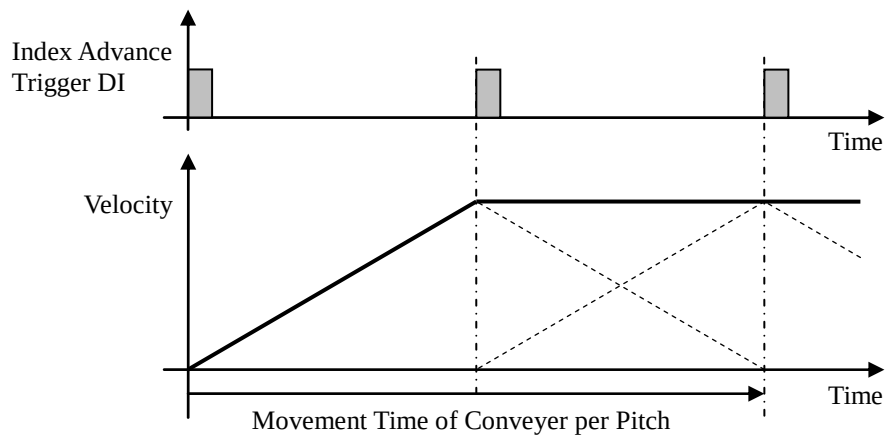


Fig. D.2.3(d) Speed pattern if Index dwell is "-1"

- 4 Move the cursor to [3 Indexer Register start] and enter the start index of the 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 would like to change 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 line 11 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 (7DC3 and later)

Step

- 1 Move the cursor to [4 Program and speed control] in "SETUP Indexers" and press F2 [DETAIL] key or the [ENTER] key.

FANUC Conveyor:		4/4
FANUC Conveyor: 1		
1	Encoder Number:	1
2	Conveyor Type:	Continuous
3	Tracking Schedule:	1
	Encoder Scale (cnt/mm):	524288
4	program and speed control:	Detail
[TYPE]	DETAIL	CNVY

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 Program	
	1/2
FANUC Indexer: 1	
1	Conveyor Speed (mm/sec) 100
2	Generate Conveyor program

- 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.

D.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 “4.6 SETUP OF A CONVEYOR STATION” or “6.8 SETUP OF A CONVEYOR STATION”.

D.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 “5.10.2.3 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.

D.3 FUNCTION FOR SERVO CONVEYOR LINE TRACKING

D.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 \$LNCFG_GRP[.].\$SOFT_DELAY, \$SRVO_DELAY.

Please refer to 5.4 FINE TUNING HIGH SPEED ACCURACY of Line Tracking OPERATOR’S MANUAL. It is possible to adjust a dynamic error on every servo conveyor.

\$SLTK_GRP[number of servo conveyor group].\$SRVO_DELAY

D.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.

D.3.3 KAREL Program for Servo Conveyor Line Tracking

If you set “FANUC” as [2 Indexer Type] on 7DC2 series or select “Index” as [2 Conveyor Type] on 7DC3 series or later, it is possible to use the following KAREL programs for servo conveyor line tracking. 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 iRPickTool KAREL programs.

- SLTKINIT
- SLTKREST
- SLTKPSHQ
- SLTKPOPQ
- SLTKDELQ
- SLTKRSTQ
- SLTKGTPP

This subsection explains these KAREL programs and Discard Line, Stop Conveyor and Distribution Line.

NOTE

- 1 If you set "FANUC" as [2 Indexer Type] on 7DC2 series or select "Index" as [2 Conveyor Type] on 7DC3 series or later, it is possible to use these KAREL programs for servo conveyor line tracking.
- 2 Distribution Line function and SLTKGTPP is available since 7DC2/15.

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.

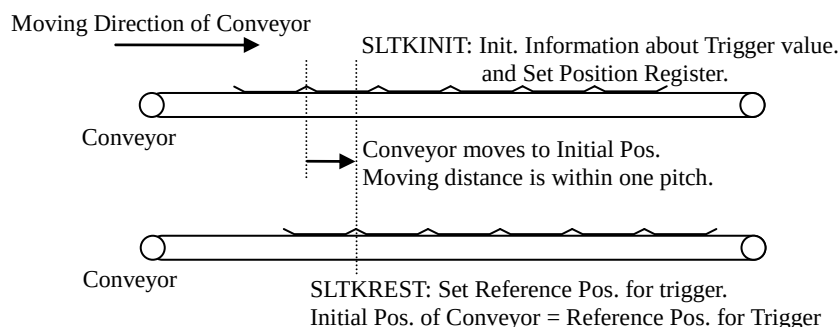


Fig. D.3.3(a) SLTKINIT, SLTKREST

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.

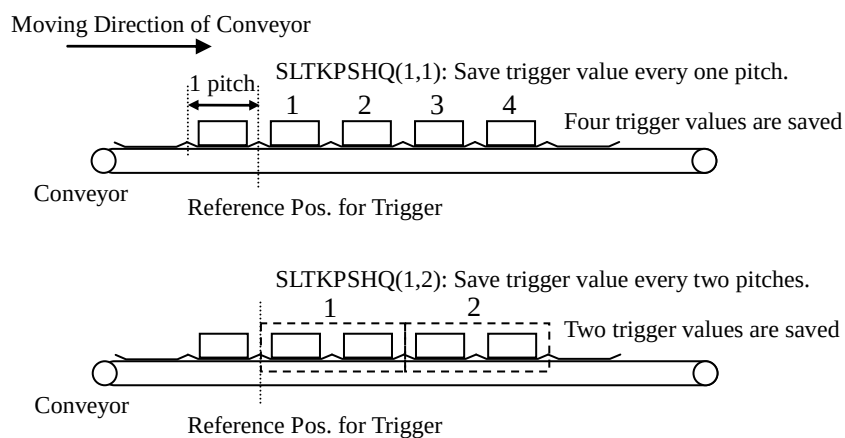


Fig. D.3.3(b) SLTKPSHQ

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.

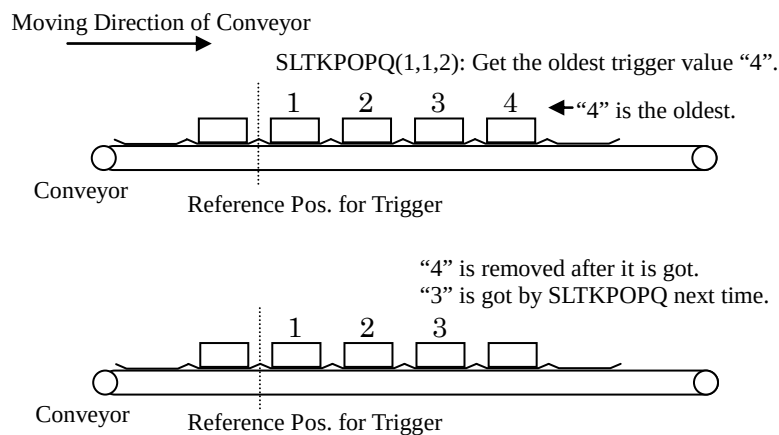


Fig. D.3.3(c) SLTKPSHQ

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.

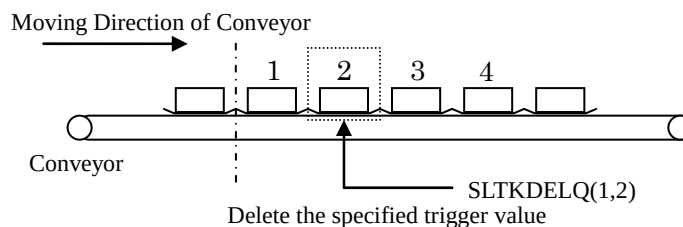


Fig. D.3.3(d) SLTKDELQ

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.

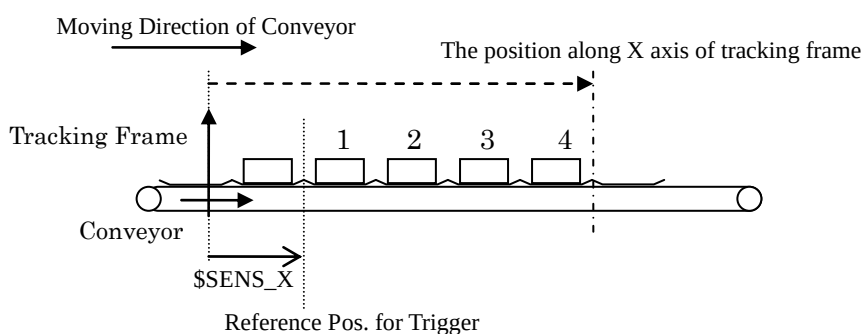


Fig. D.3.3(e) SLTKGTPP

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.

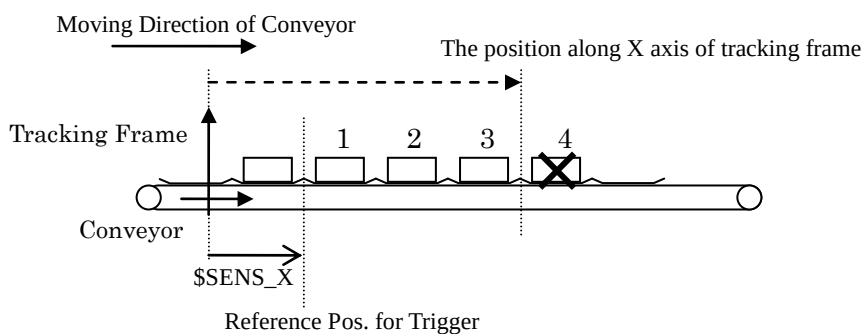


Fig. D.3.3(f) SLTKGTPP and SLTKDELQ

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.

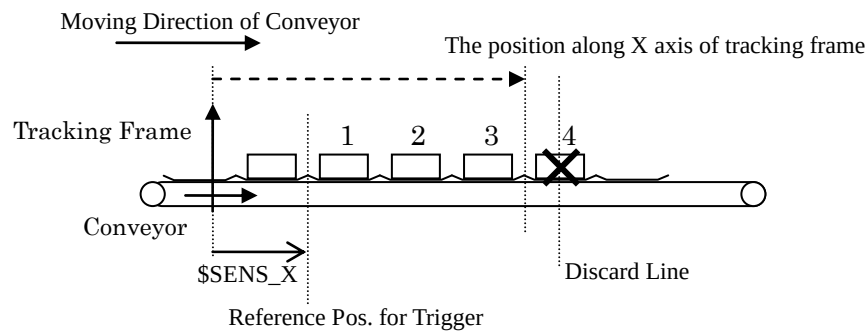


Fig. D.3.3(g) SLTKGTPP and Discard Line

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.

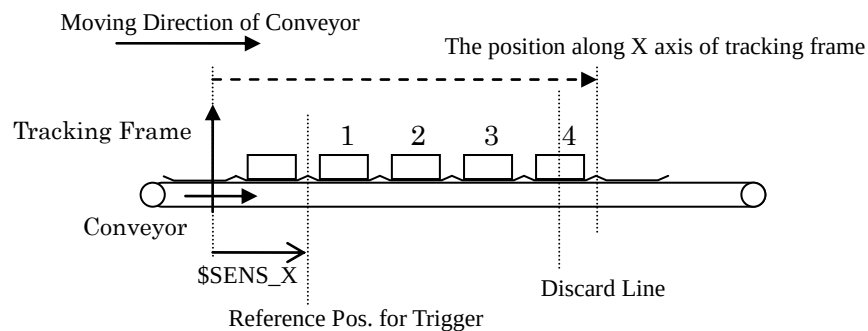


Fig. D.3.3(h) SLTKDELQ and Stop Conveyor

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.

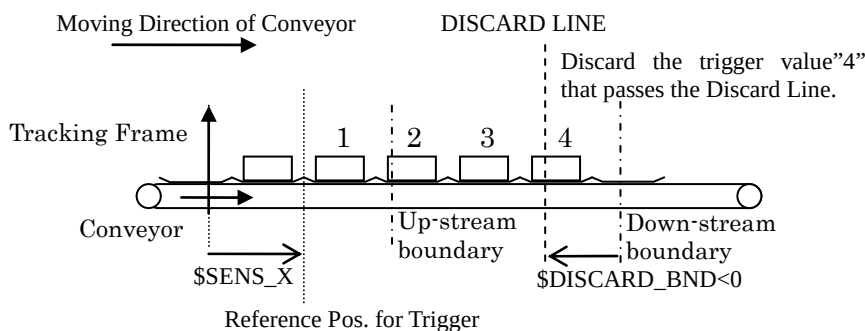


Fig. D.3.3(i) Discard line

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.

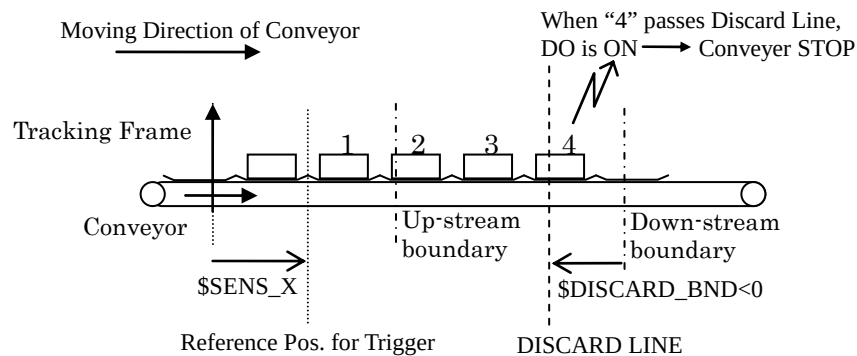


Fig. D.3.3(j) Stop conveyor

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.

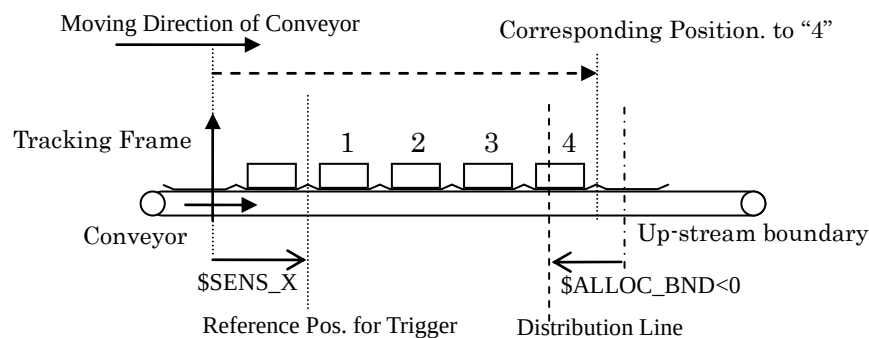


Fig. D.3.3(k) Distribution Line

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[msec]. 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.

D.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.
- Servo conveyor line tracking function on 7DC2 series uses flags from 131 to 138. You must not use these flags for another use. (A flag is an internal I/O port that can be read and set.)
It is possible to specify flags on setup display on 7DC3 series and later

E α A1000S PULSECODER

This chapter contains the usage of α A1000S Pulsecoder (A860-0372-T001).

E.1 REQUIREMENTS

E.1.1 R-30iB Controller

Table E.1.1 R-30iB: α A1000S Pulsecoder : A860-0372-T001

Required Components	R-30iB Controller A-Cabinet	R-30iB 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 G "SEPARATE DETECTOR UNIT (SDU)".) αA1000S Pulsecoder can be also connected to Pulsecoder terminal (JD17) on Main 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-30iB Controller A-Cabinet	R-30iB Controller B-Cabinet	Comments
Hardware			
Pulsecoder Cable (In case of an α A1000S 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 α A1000S Pulsecoder as an absolute Pulsecoder, cables in this table can not be used. When the pulse multiplexer is used, α A1000S Pulsecoder cannot be used. If you want to use multiple robots with α A1000S 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)	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)	

Refer to Fig. E.2 (a) to Fig. E.2 (c) for Pulsecoder signal information, specifications, and images containing the dimensions of the Pulsecoders.

Refer to Fig. E.3 (a) to Fig. E.3 (e) for information on connections of the cables.

E.1.2 R-30iB Mate Controller

Table E.1.2 (a) R-30iB Mate: α A1000S Pulsecoder : A860-0372-T001

Required Components	R-30iB Mate Controller	R-30iB Mate 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-30iB Mate, mini slot) A05B-2655-J035, A05B-2662-J035 (R-30iB Mate (Open Air), mini slot) αA1000S Pulsecoder can be also connected to Pulsecoder terminal (CRS41) on Main board. In this case, Line tracking interface board is made redundant, but Learning Vibration Control function (A05B-2660-J573) (option) cannot be used.
Fiber Optic (FSSB) Cable	A66L-6001-0026	A66L-6001-0023	The Line tracking interface board in the left column is included in the following order specification. A05B-2650-J035, A05B-2661-J035 (R-30iB Mate, mini slot) A05B-2655-J035, A05B-2662-J035 (R-30iB Mate (Open Air), mini slot)
Pulsecoder Cable (In case of an α A1000S 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 α A1000S Pulsecoder as an absolute Pulsecoder, cables in this table can not be used.
	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)	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)	When the pulse multiplexer is used, α A1000S Pulsecoder cannot be used. If you want to use multiple robots with α A1000S Pulsecoder, use Ethernet Encoder function (A05B-2660-R762) (option).
	For Main 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 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)	α A1000S Pulsecoder cannot be connected to Main board A (A20B-8200-0790) for R-30iB Mate. See Table 2.1.1 (d).

Refer to Fig. E.2 (a) to Fig. E.2 (c) for Pulsecoder signal information, specifications, and images containing the dimensions of the Pulsecoders.

Refer to Fig. E.4 (a) to Fig. E.4 (d) for information on connections of the cables.

Table E.1.2 (b) R-30iB Mate Main board which can connect to α A1000S Pulsecoder

Main board	Board Specification	Comments
Main board B	A20B-8200-0791	The Main board in the left column is included in the following order specification. A05B-2650-H002, A05B-2661-H002 (R-30iB Mate) A05B-2655-H002, A05B-2662-H002 (R-30iB Mate (Open Air))
Main board C	A20B-8200-0792	The Main board in the left column is included in the following order specification. A05B-2650-H003, A05B-2661-H003 (R-30iB Mate) A05B-2655-H003, A05B-2662-H003 (R-30iB Mate (Open Air))



CAUTION

Main board A (A20B-8200-0790) cannot be connected to α A1000S Pulsecoder.

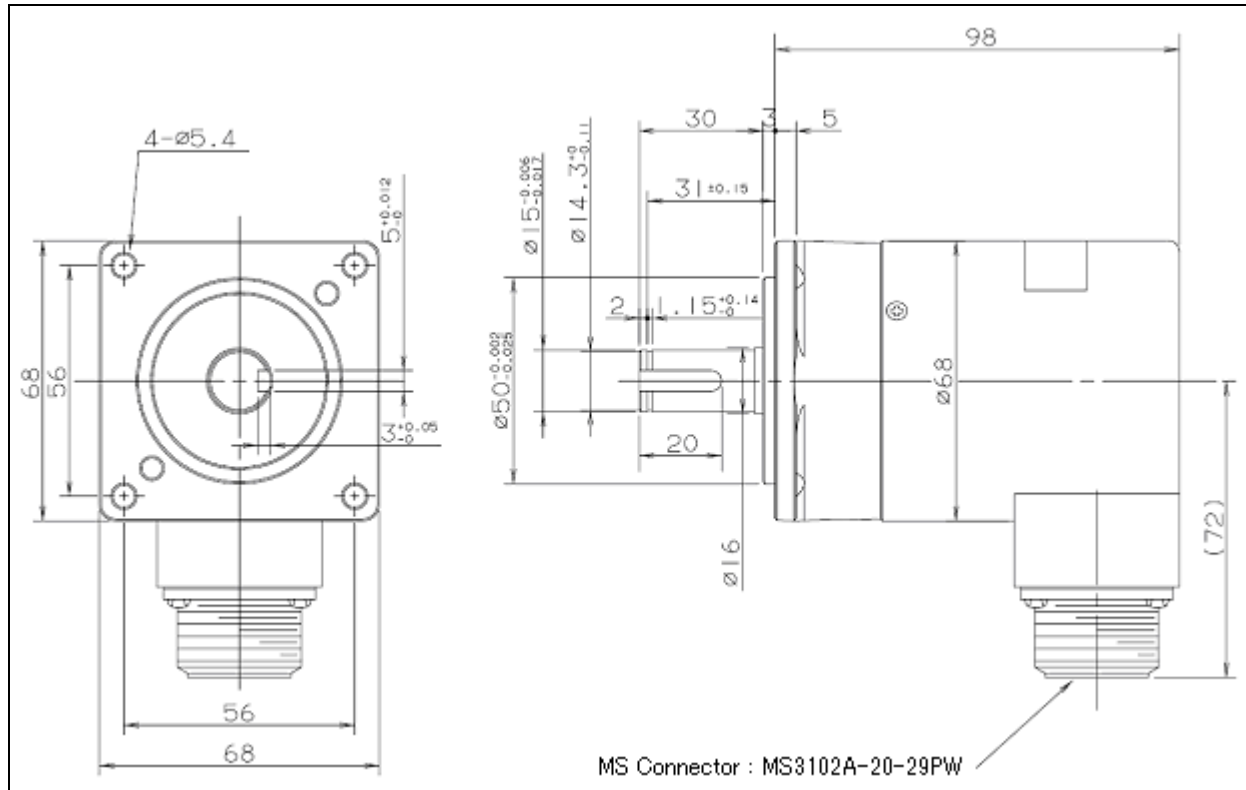
E.2 FIGURES

A860-0372-T001 3102A20-29PW	Signal Name	Pin No.
		α A1000S 3102A20-29P
	SD	A
	*SD	D
	REQ	F
	*REQ	G
	+5V	J,K
	0V	N,T
	Shield	H
	+6VA	R
	0VA	S

Fig. E.2 (a) α A1000S Pulsecoder (A860-0372-T001) connection signal information

Item		Specification
Power voltage		5 [V] $\pm$ 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 1×10^{-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		0.02×10^{-3} [m]

Item	Specification
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. E.2 (b) α A1000S Pulsecoder (A860-0372-T001) specificationsFig. E.2 (c) α A1000S Pulsecoder dimensions (A860-0372-T001)

E.3 HOW TO CONNECT TO R-30iB

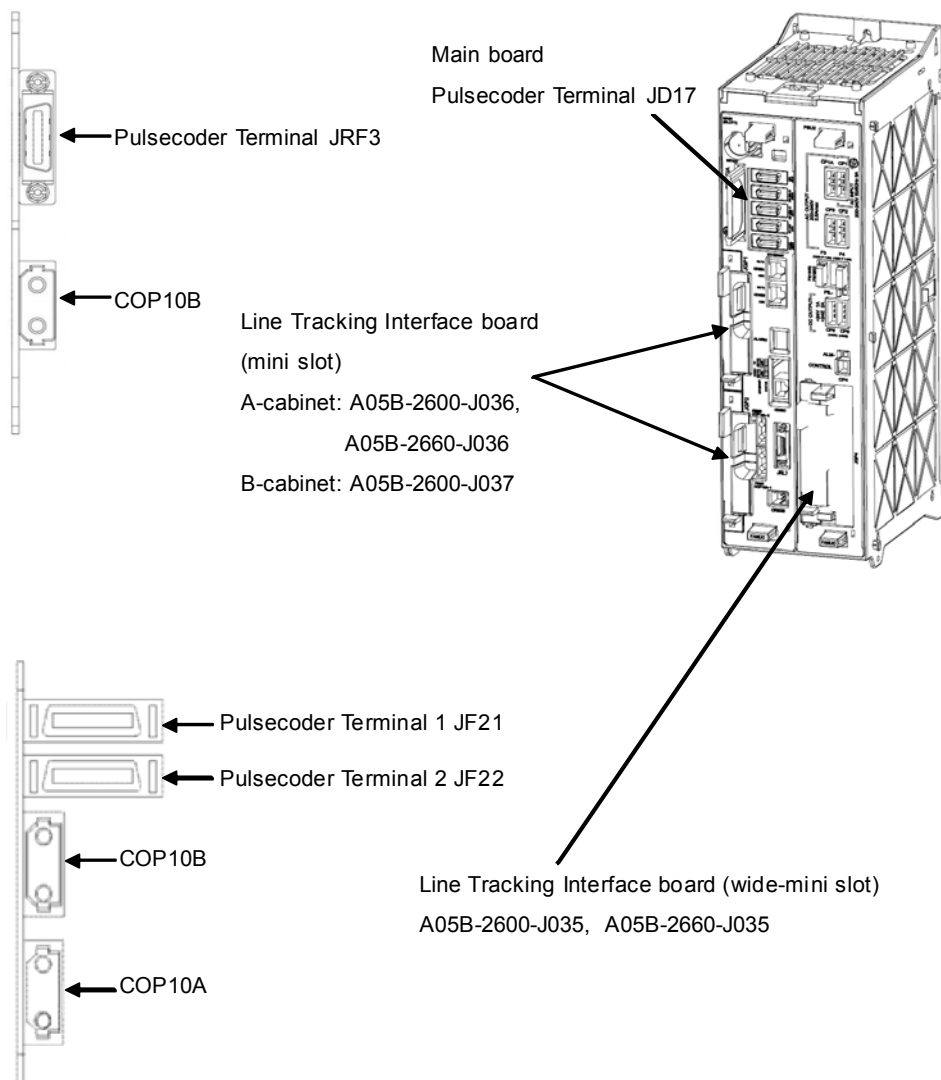


Fig. E.3 (a) R-30iB Controller with line tracking connections

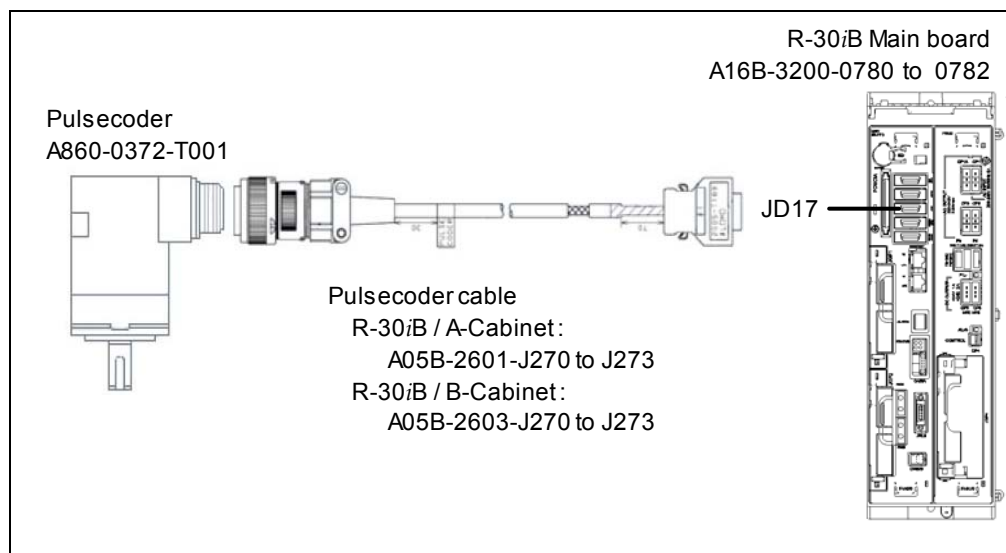


Fig. E.3 (b) Connecting cables with R-30iB Main board (one α A1000S Pulsecoder)

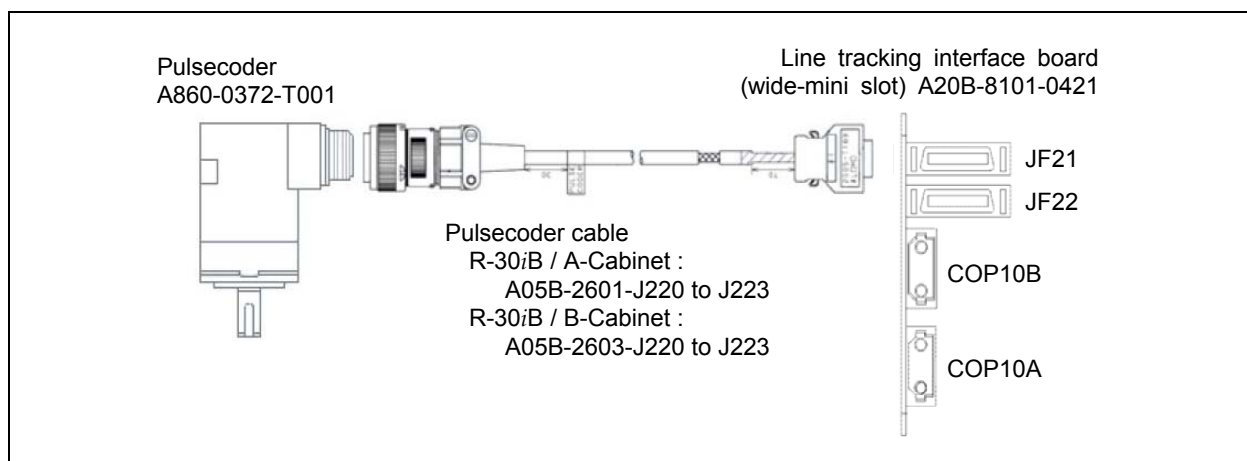


Fig. E.3 (c) Connecting cables with Line tracking interface board A05B-2600-J035 or A05B-2660-J035 (one α A1000S Pulsecoder)

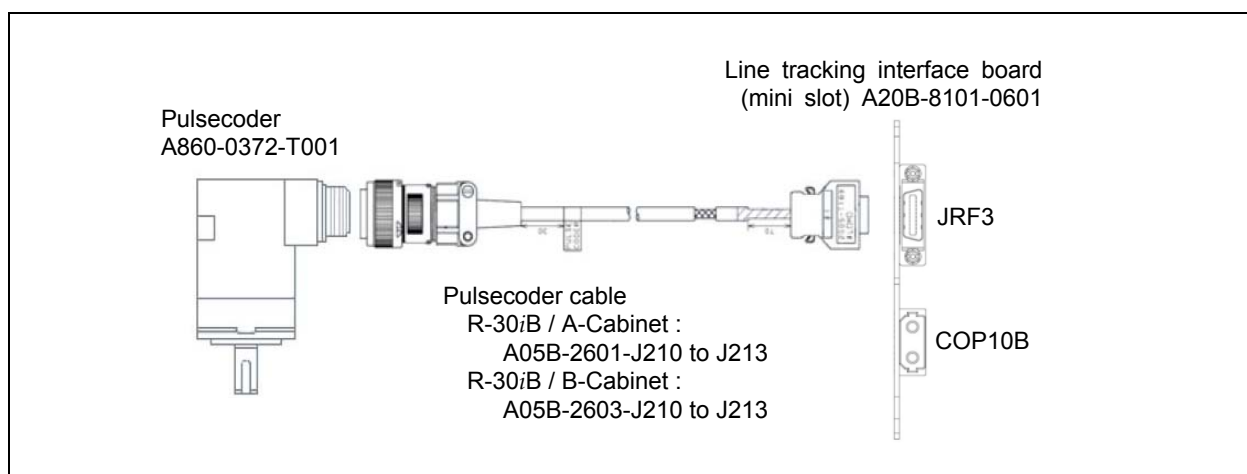


Fig. E.3 (d) Connecting cables with Line tracking interface board A05B-2600-J036, A05B-2660-J036 or A05B-2600-J037 (one α A1000S Pulsecoder)

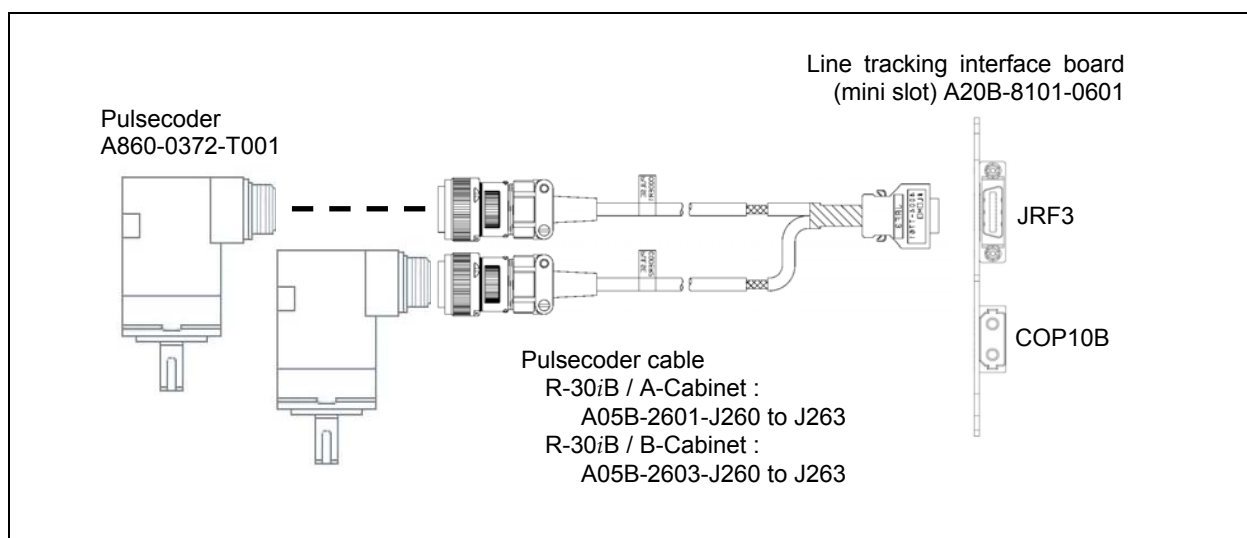


Fig. E.3 (e) Connecting cables with Line tracking interface board A05B-2600-J036, A05B-2660-J036 or A05B-2600-J037 (two α A1000S Pulsecoders)

E.4 HOW TO CONNECT TO R-30iB Mate

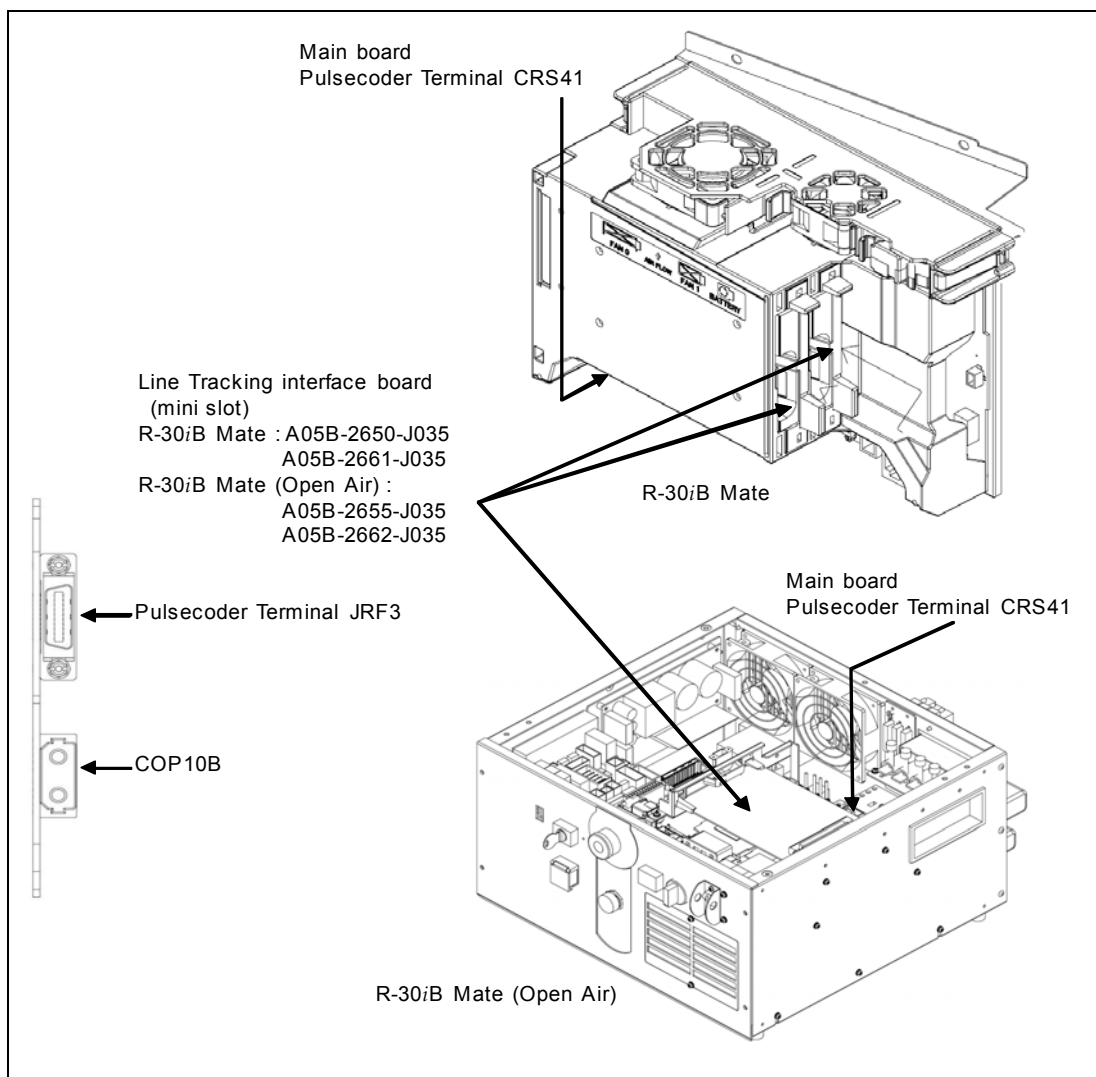


Fig. E.4 (a) R-30iB Mate Controller with line tracking connections

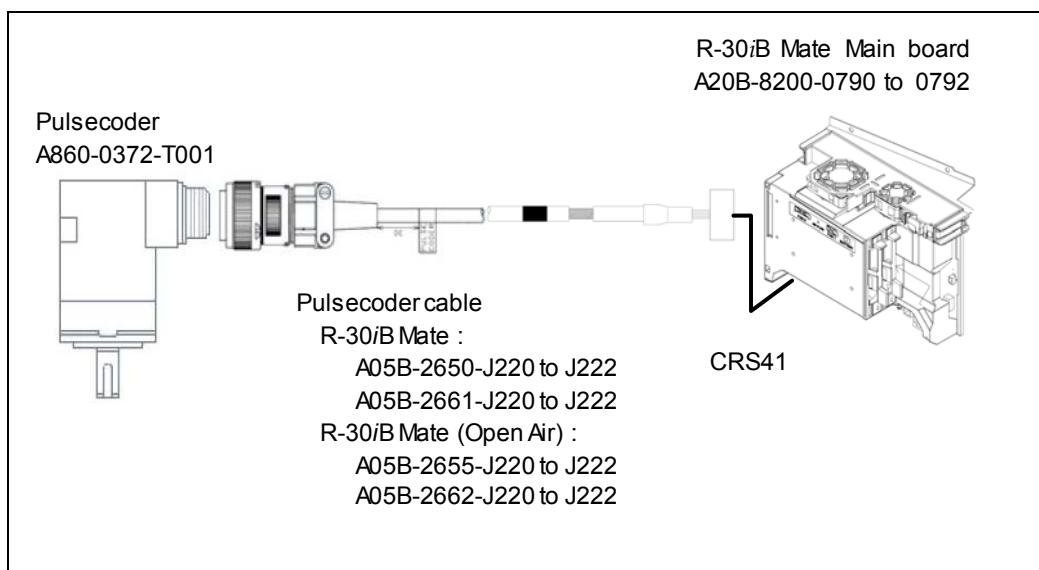


Fig. E.4 (b) Connecting cables with R-30iB Mate Main board (one α A1000S Pulsecoder)

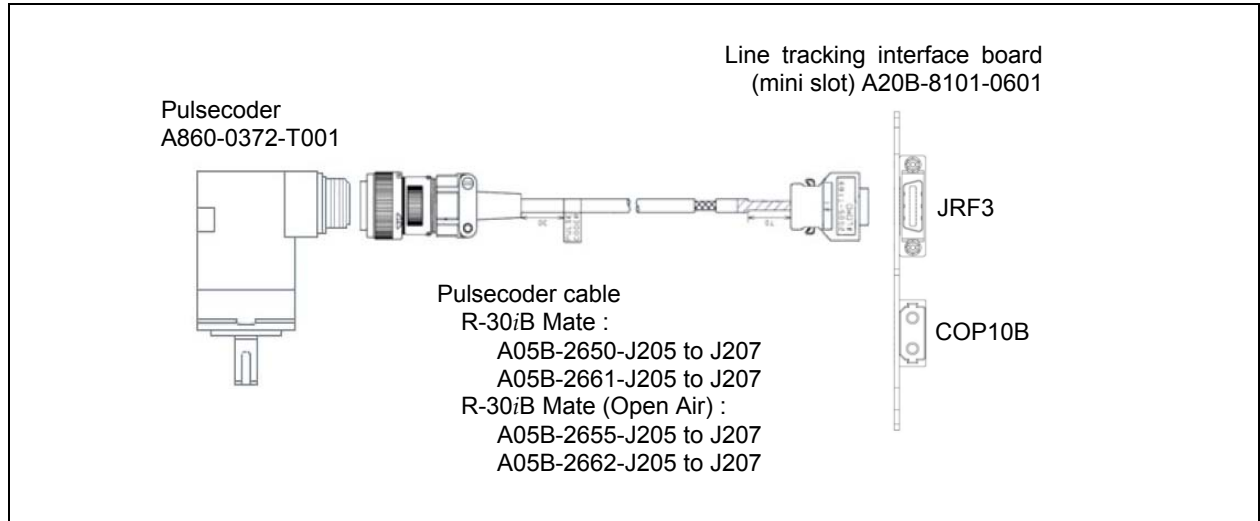


Fig. E.4 (c) Connecting cables with Line tracking interface board A05B-2650-J035, A05B-2655-J035, A05B-2661-J035 or A05B-2662-J035 (one α A1000S Pulsecoder)

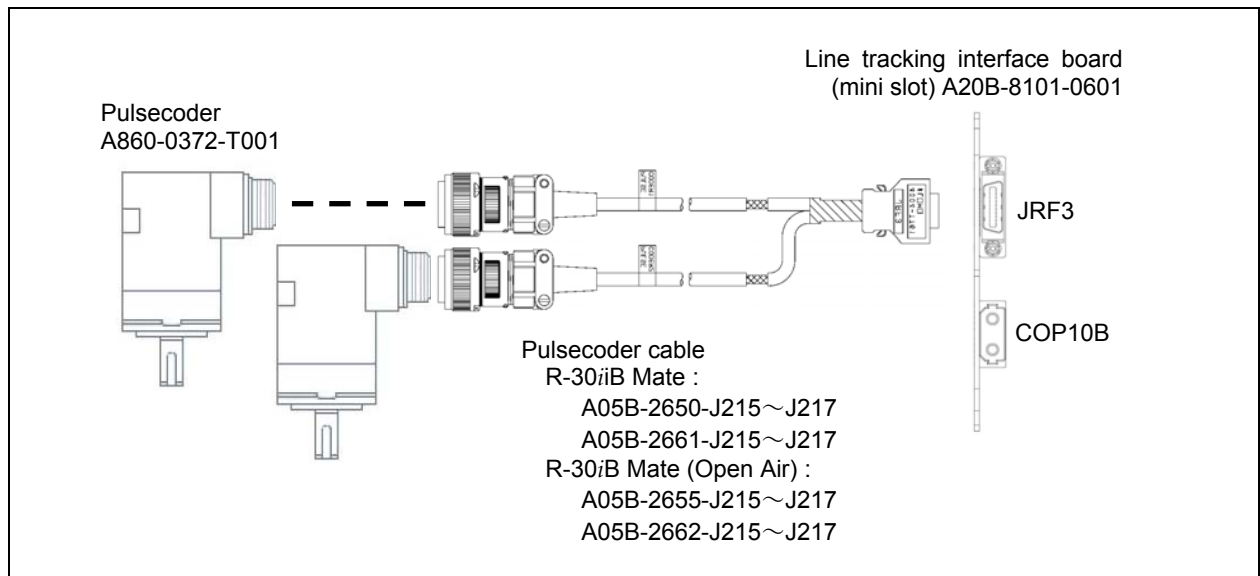


Fig. E.4 (d) Connecting cables with Line tracking interface board A05B-2650-J035, A05B-2655-J035, A05B-2661-J035 or A05B-2662-J035 (two α A1000S Pulsecoders)

NOTE

If the line tracking interface board or the Main board Pulsecoder terminal cannot be used or is not available, you can use the SDU shown in Fig. G (a) to Fig. G (i).

F INCREMENTAL PULSECODER

This chapter contains the usage of Incremental Pulsecoder (A860-0301-T001 to T004).

F.1 REQUIREMENTS

F.1.1 R-30iB Controller

Table F.1.1 R-30iB: Incremental Pulsecoder A860-0301-T001 to T004

Required Components	R-30iB Controller A-Cabinet	R-30iB 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 G "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-30iB Controller A-Cabinet	R-30iB 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 Pulsecoder terminal on Main 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)	
Pulsecoder to Multiplexer cable	A05B-2451-K102 (7M) A05B-2451-K103 (14M)		

Refer to Fig. F.2 (a) to Fig. F.2 (c) for Pulsecoder signal information, specifications, and images containing the dimensions of the Pulsecoders.

Refer to Fig. F.3.1 (a) to Fig. F.3.2 (b) and Fig. F.3.2 (a) to Fig. F.3.2 (b) for information on connections of the cables.

F.1.2 R-30iB Mate Controller

Table F.1.2 (a) R-30iB Mate: Incremental Pulsecoder A860-0301-T001 to T004

Required Components	R-30iB Mate Controller	R-30iB Mate 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-30iB Mate, mini slot) A05B-2655-J035, A05B-2662-J035 (R-30iB Mate (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-30iB Mate, mini slot) A05B-2655-J035, A05B-2662-J035 (R-30iB Mate (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 Pulsecoder terminal on Main board. On R-30iB Mate, the pulse multiplexer cannot be used. If you want to use multiple robots with α A1000S Pulsecoder, use Ethernet Encoder function (A05B-2660-R762) (option).
	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)	

Refer to Fig. F.2 (a) to Fig. F.2 (c) for Pulsecoder signal information, specifications, and images containing the dimensions of the Pulsecoders.

Refer to Fig. F.3.1 (c) to Fig. F.3.2 (d) and Fig. F.3.2 (a) to Fig. F.3.2 (b) for information on connections, of the cables.

F.2 FIGURES

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. F.2 (a) Incremental Pulsecoder (A860-0301-T001) connection signal information

Item		Specification
Power voltage		5 (V) 5%
Current consumption		Up to 0.35 (A)
Working temperature range		0 to 60 (°C)
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. F.2 (b) Incremental Pulsecoder (A860-0301-T001) specifications

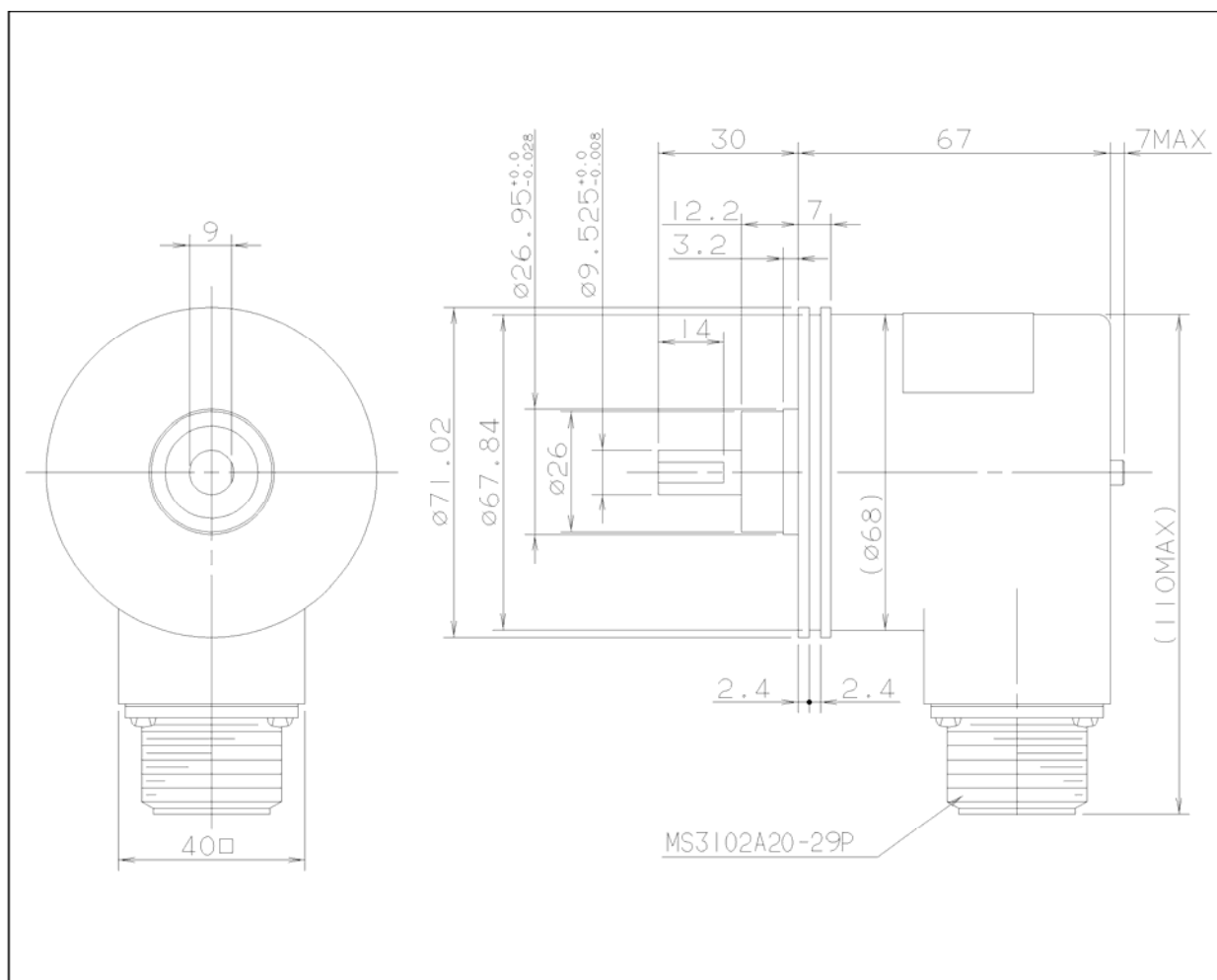


Fig. F.2 (c) Incremental Pulsecoder dimensions (A860-0301-T001)

F.3 HOW TO CONNECT

F.3.1 Connecting to Robot Controller Directly

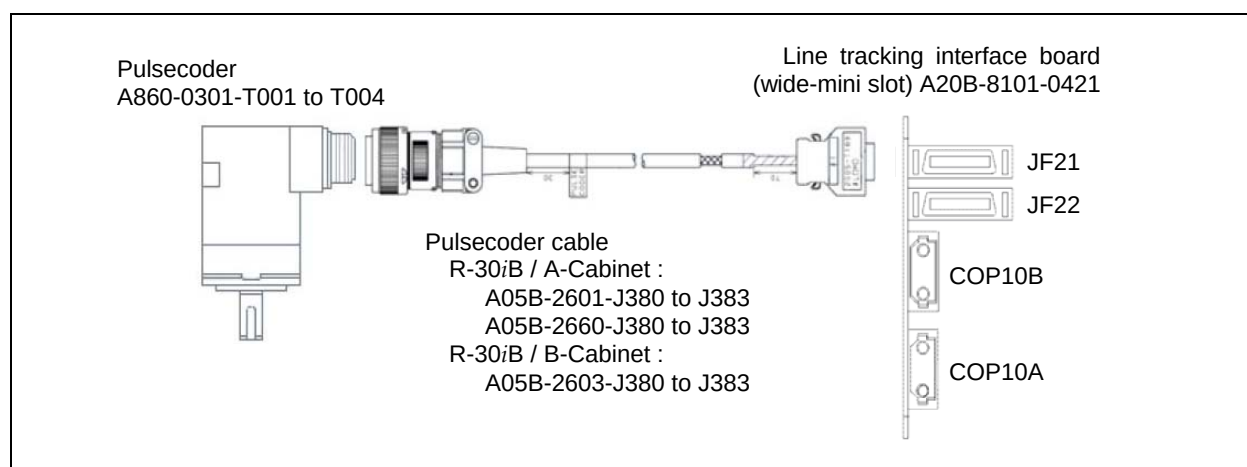


Fig. F.3.1 (a) Connecting cables with Line tracking interface board A05B-2600-J035, A05B-2660-J035 (one Pulsecoder)

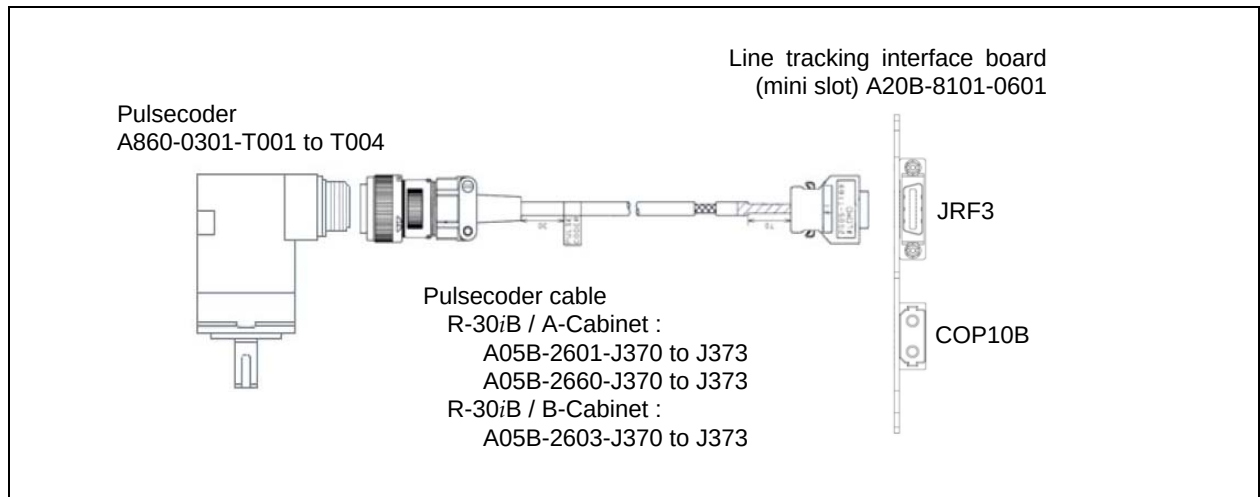


Fig. F.3.1 (b) Connecting cables with Line tracking interface board A05B-2600-J036, A05B-2660-J036 or A05B-2600-J037 (one Pulsecoder)

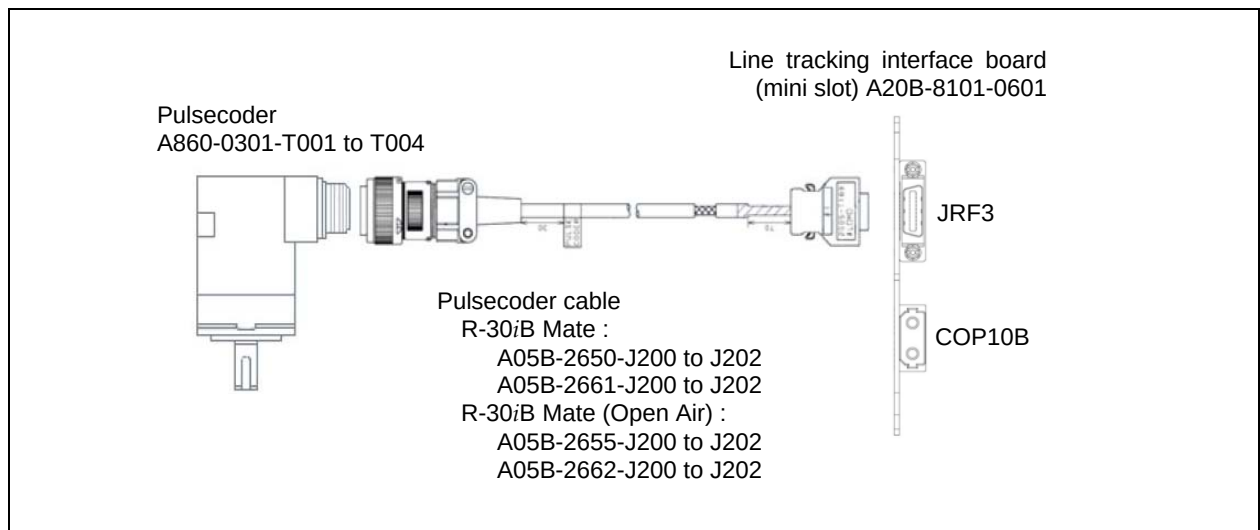


Fig. F.3.1 (c) Connecting cables with Line tracking interface board A05B-2650-J035, A05B-2655-J035, A05B-2661-J035 or A05B-2662-J035 (one Pulsecoder)

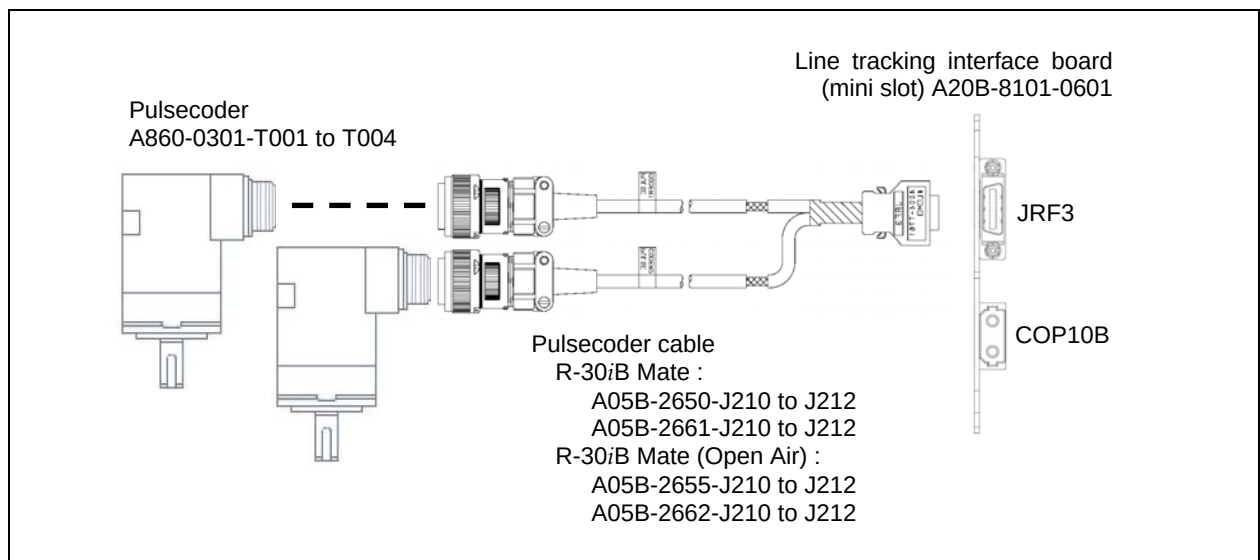


Fig. F.3.1 (d) Connecting cables with Line tracking interface board A05B-2650-J035, A05B-2655-J035, A05B-2661-J035 or A05B-2662-J035 (two Pulsecoders)

F.3.2 Connecting to Pulse Multiplexer

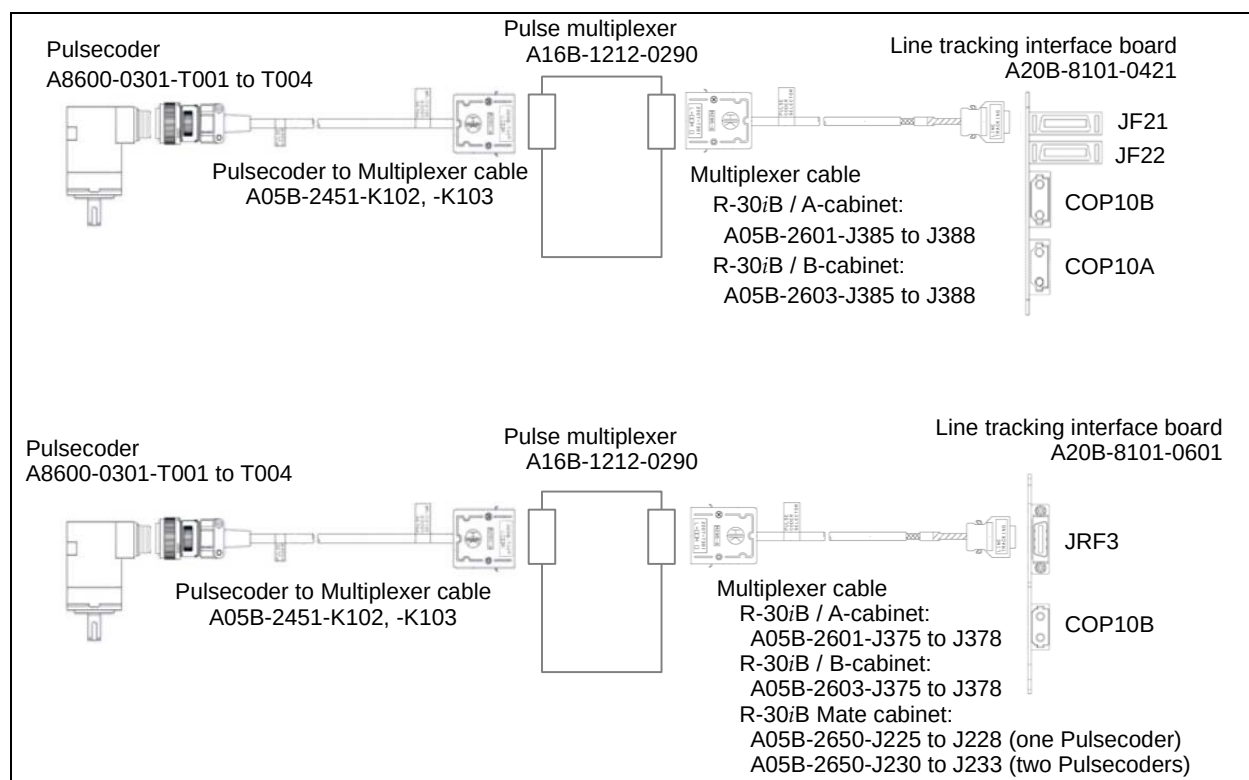


Fig. F.3.2 (a) Pulse multiplexer and connecting cables 1 (Pulsecoder, A860-0301-T001 to T004)

NOTE

If the line tracking interface board cannot be used or is not available, you can use the SDU shown in Fig.G (a) to Fig. G (i).

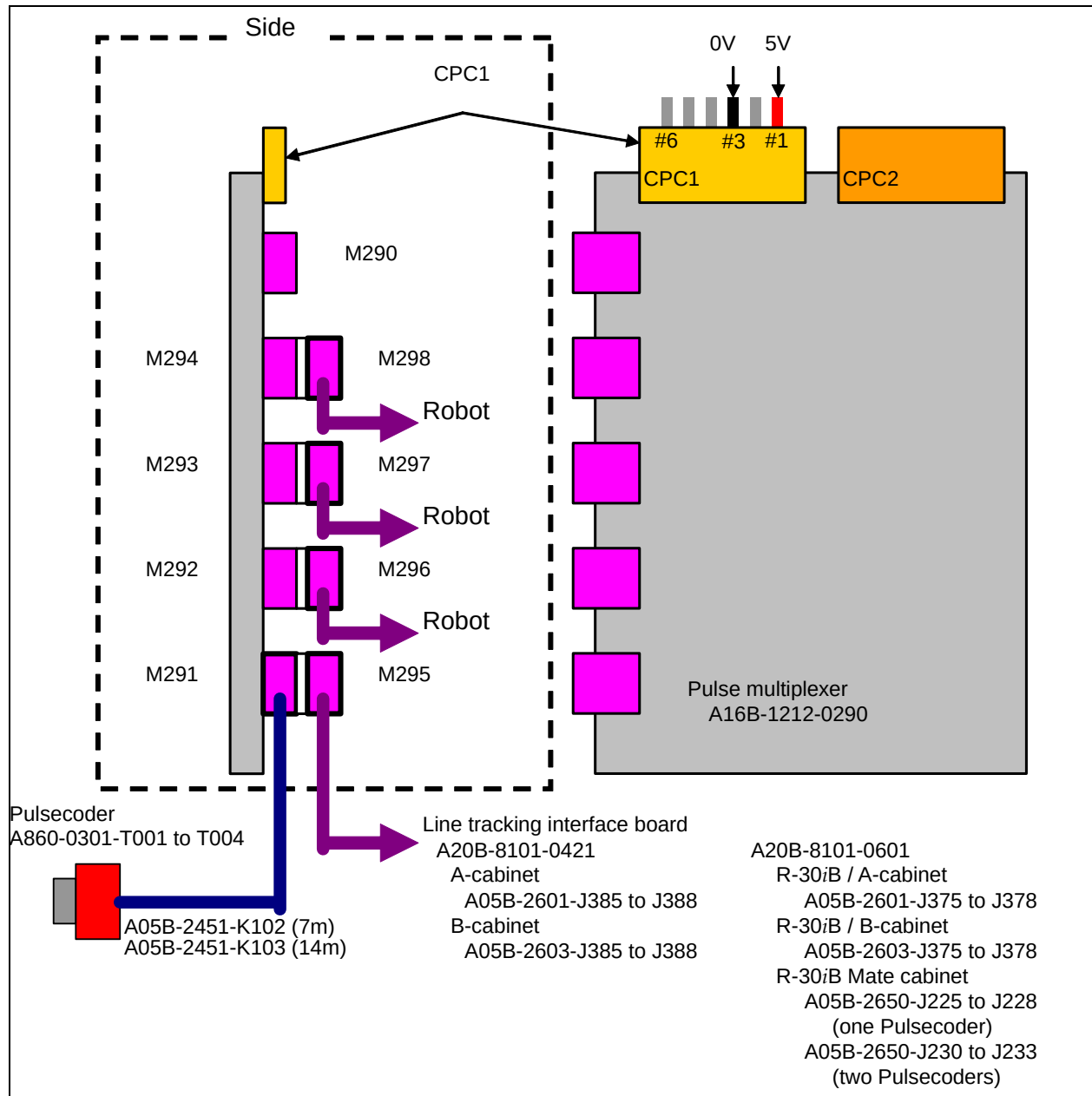


Fig. F.3.2 (b) Connecting cables to the multiplexer 2 (Pulsecoder, A860-0301-T001 to T004)

NOTE

1. You can connect up to four cables (pulse multiplexer to/from controller) to one pulse multiplexer.
2. You can connect five or more controllers to one Pulsecoder by using two or more Pulse multiplexers and connecting each other with a Pulse multiplexer connection cable(A05B-2600-K155,156 / A05B-2660-K155,K156).



CAUTION

Power to Multiplexer should be supplied before power on to robot controller. Otherwise, "SRVO-082 DAL alarm" will occur and the value of Pulsecoder cannot be read in robot controller.

G SEPARATE DETECTOR UNIT (SDU)

If the line tracking interface board and main board cannot be used or is not available, such as more than 4 Pulsecoders should be connected to only one controller, you can use the SDU. Because SDU requires retrofit work to mount in the container, please refer Fig. G (a) to Fig. G (i) and work properly.

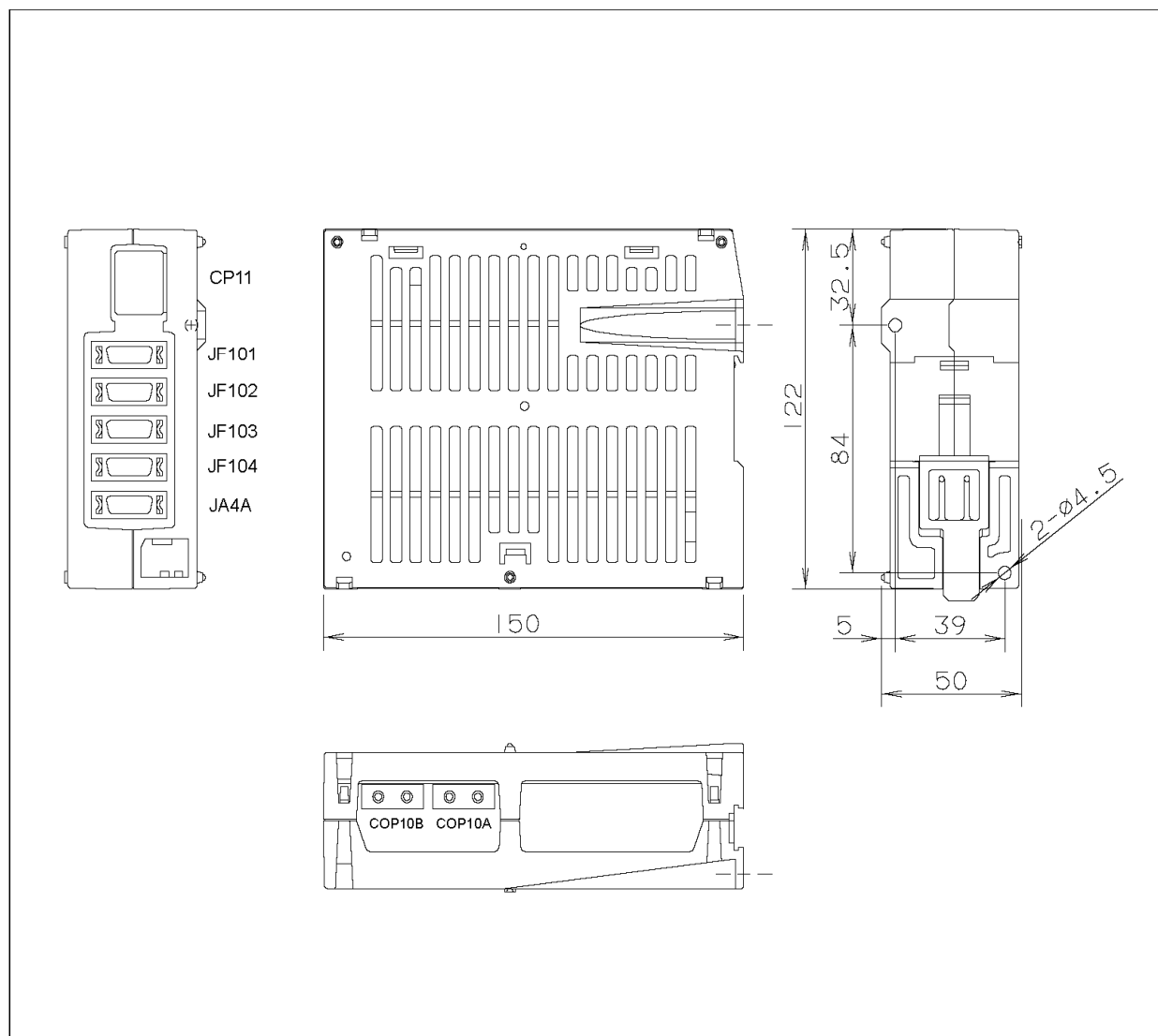


Fig. G (a) External dimensions of separate detector unit (SDU)

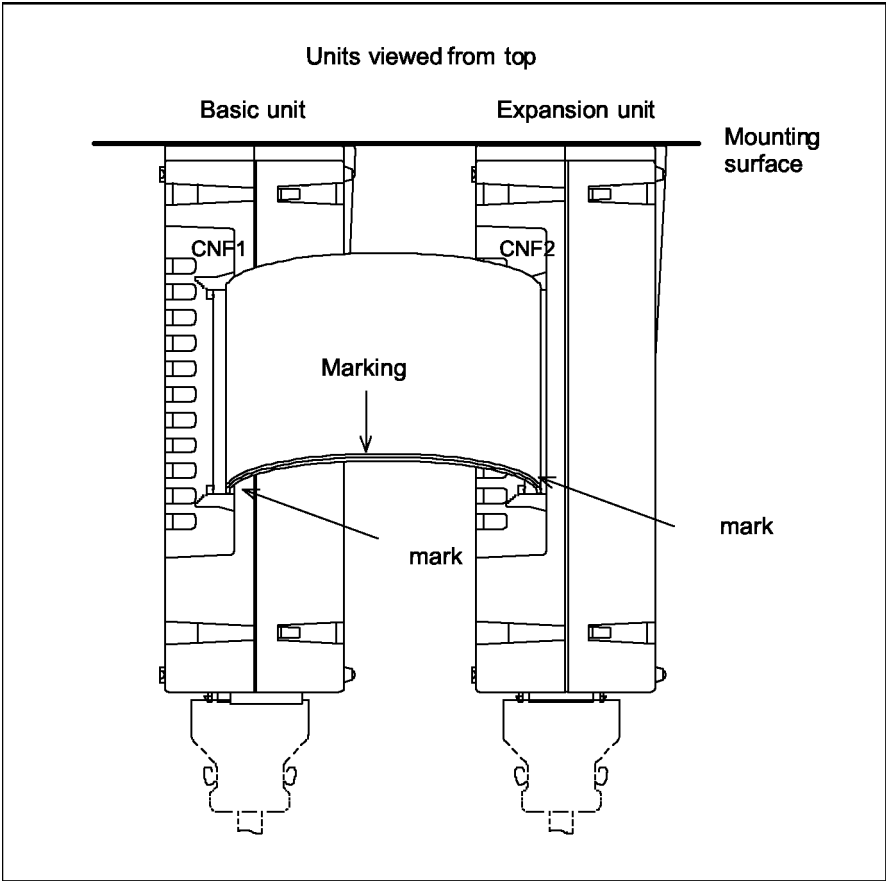


Fig. G (b) Cable connection between basic unit and expansion unit

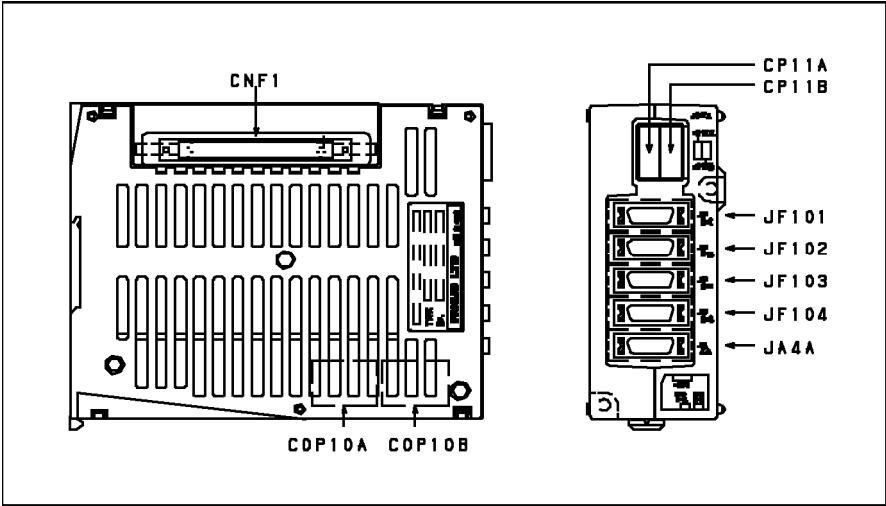


Fig. G (c) Connector locations on the basic unit

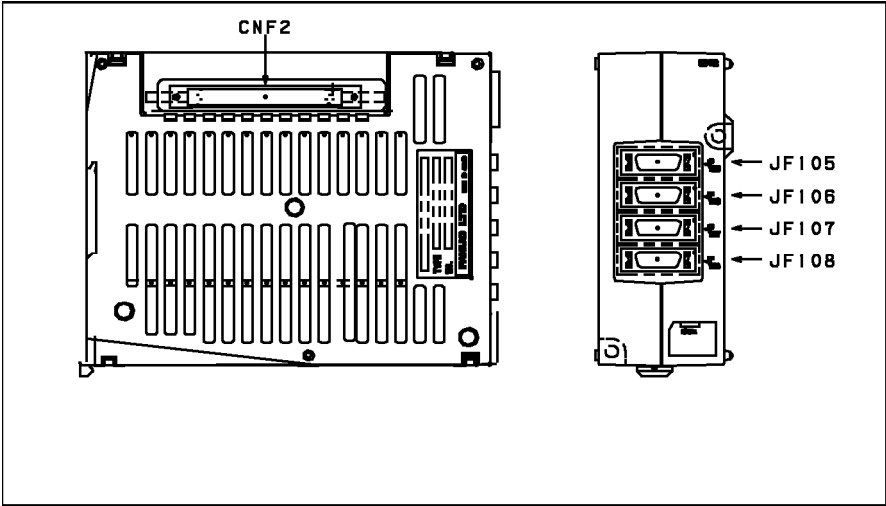


Fig. G (d) Connector locations on the expansion unit

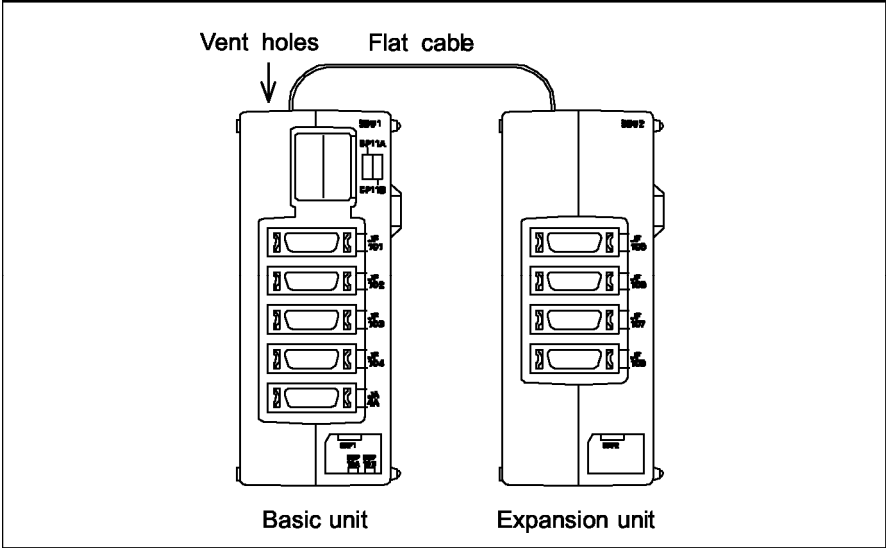


Fig. G (e) Flat cable placement during installation

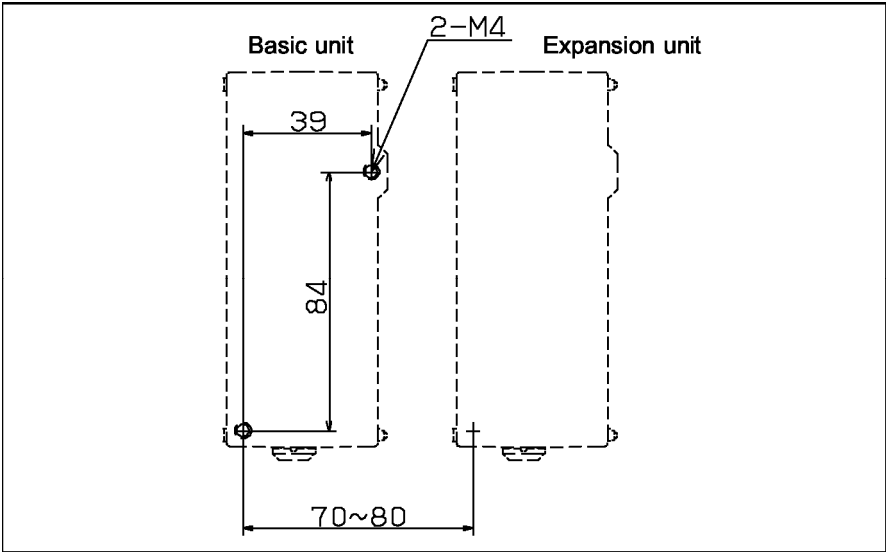
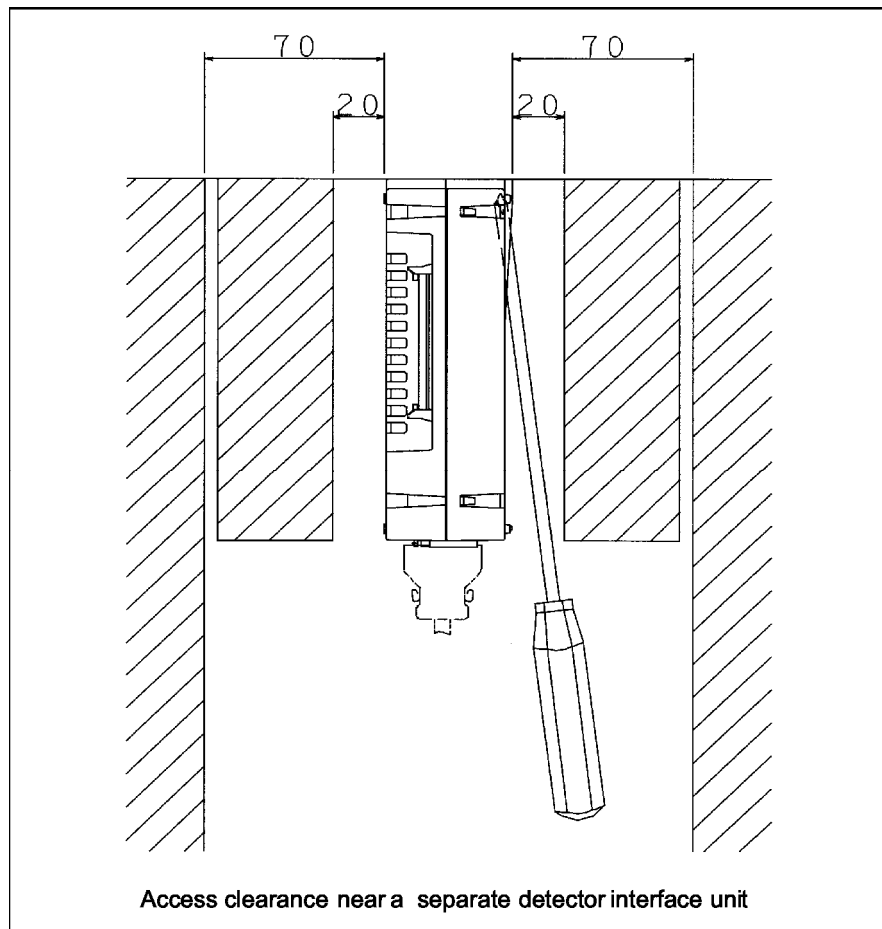


Fig. G (f) Horizontal separation of mounting holes during installation

⚠ 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.

**Fig. G (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).

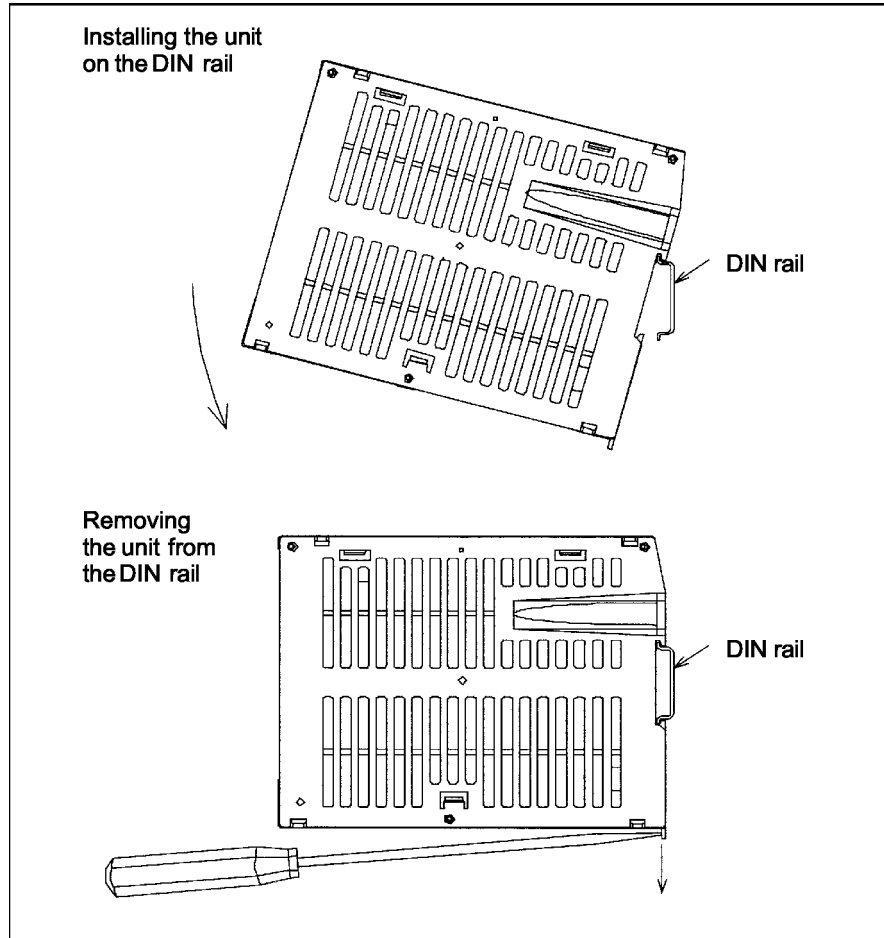


Fig. G (h) Installing and removing the unit

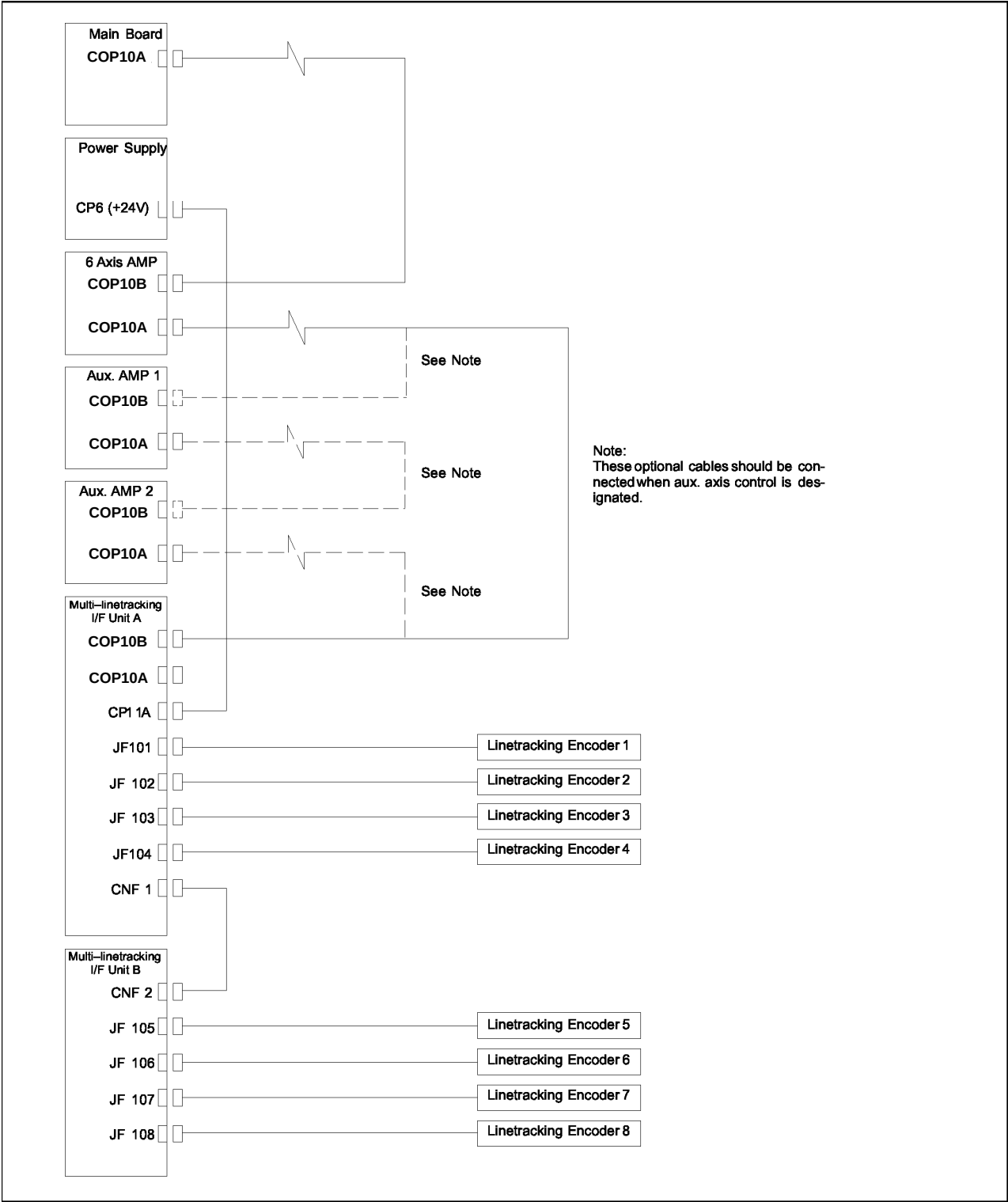


Fig.G (i) Connection diagram

H

KAREL PROGRAM LIST

This is a correspondence table of KAREL programs of the old visual tracking software on and before 7DC1 series and *iRPickTool*.

No.	Old KAREL	Description	Argument	Version	New KAREL	Argument	Comment	Version
1	VSTKCLRQ.PC	Clear a queue.	1: Workarea name	V7.40P/01	PKCSCLRQUE.PC	1: Conveyor Station ID	ID is gotten by PKCSGETID.	V8.20P/01
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	V7.40P/01	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.	V8.20P/01
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	V7.40P/01	PKCSACKQUE.PC	1: Conveyor Station ID 2: Processing result	You need not to specify vision register number as argument.	V8.20P/01
4	VSTKSTRT.PC	Start sensor.	None	V7.40P/01	PKSNSTART.PC	None		V8.20P/01
5	VSTKSTOP.PC	Stop sensor.	None	V7.40P/01	PKSNSTOP.PC	None		V8.20P/01
6	VSTKSTLB.PC	Change loadbalance.	1: Line name 2: Register number to specify loadbalance 3: Number of workarea 4: Bypass 5(Optional): Model ID	V7.50P/07	PKCVSETLBD.PC	1: Conveyor name 2: Register number to specify loadbalance 3: Number of conveyor station 4: Bypass 5(Optional): Model ID		V8.20P/01
7	VSTKENLB.PC	Switch load balance enabled/disabled.	1: Line name 2: Enable/Disable	V7.50P/07	PKCVENBLB.PC	1: Conveyor name 2: Enable/Disable		V8.20P/01
8	VSTKENSC.PC	Switch stop conveyor enabled/disabled.	1: Line name 2: Register number to specify enable/disable 3: Number of workarea	V7.50P/07	PKCVENBSC.PC	1: Conveyor name 2: Register number to specify enable/disable 3: Number of conveyor station		V8.20P/01
9	VSTKSTVN.PC	Change vision program name.	1: Sensor number 2: Vision process name	V7.50P/07	PKSNSETVPNAM.P	1: Sensor name 2: Vision process name		V8.20P/14
10	VSTKSTTN.PC	Change tray name.	1: Sensor number 2: Tray name	V7.70P/09	PKCVSETTRNAM.P	1: Conveyor name 2: Tray name		V8.20P/14
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	V7.50P/01	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.	V8.20P/01
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	V7.70P/17	PKCSGETPFRT.PC	1: Conveyor Station ID 2: Register number to output prediction pick rate	ID is gotten by PKCSGETID.	V8.20P/01
13	VSTKNPRT.PC	Return the number of part in an queue.	1: Workarea name 2: Register number to output number of part in queue	V7.40P/02	PKCSGETNPRT.PC	1: Conveyor Station ID 2: Register number to output number of part in queue	ID is gotten by PKCSGETID.	V8.20P/01

SENSOR CUSTOMIZATION

I.1 OVERVIEW

Sensor takes the following role in the workcell of *iRPickTool*.

- A sensor monitors a conveyor movement and an external signal like DI 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.

I.2 CUSTOMIZATION STEPS

Sensor should be customized using ROBOGUIDE. The following steps are performed in ROBOGUIDE.

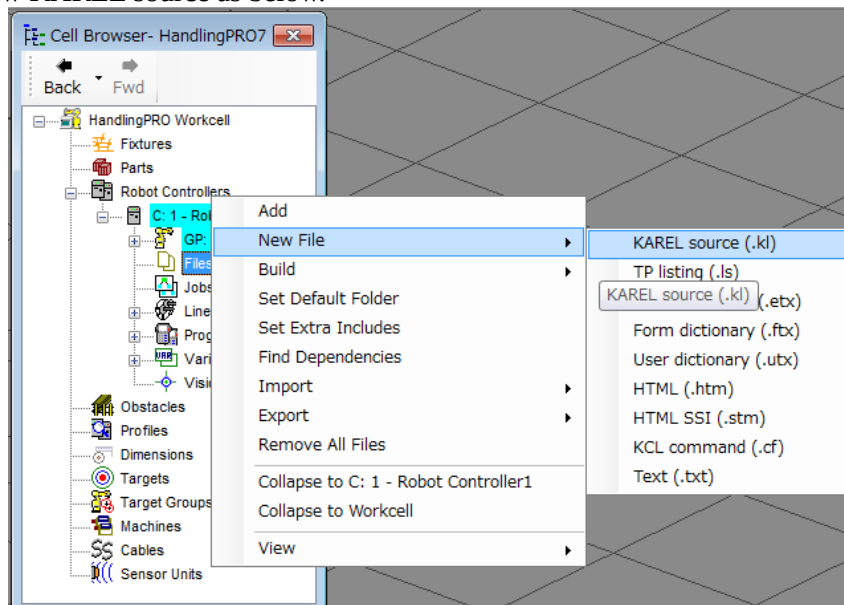
- 1 Make a workcell in ROBOGUIDE. You have to order the following option at least when you create a virtual robot.
 - *iRPickTool* Basic (J944)



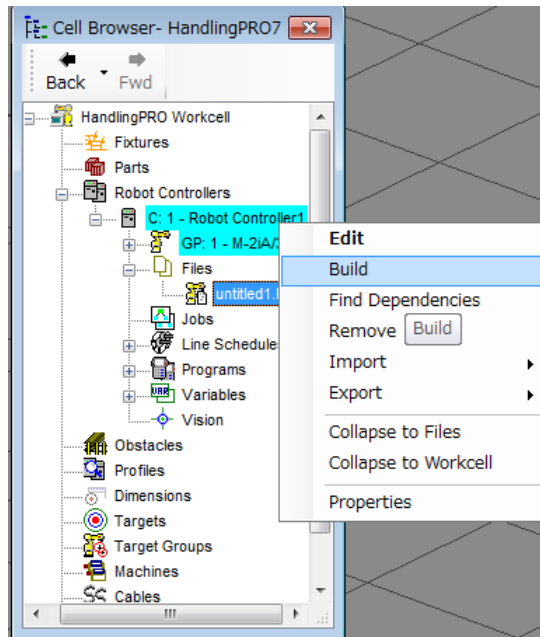
CAUTION

You need the virtual robot of V8.20P/10(7DC2/10) or later in order to build (translate) the customized sensor.

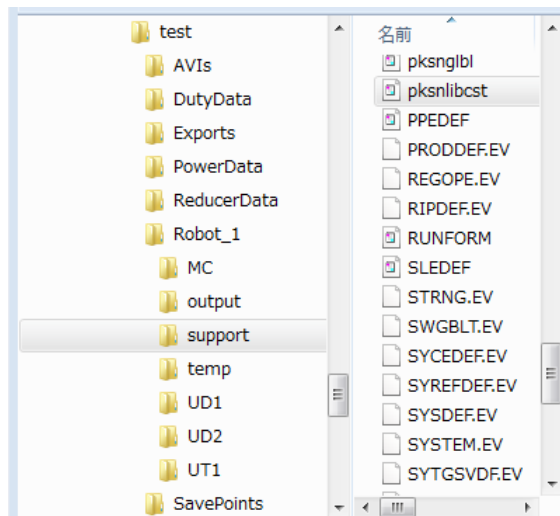
- 2 Create a new KAREL source as below.



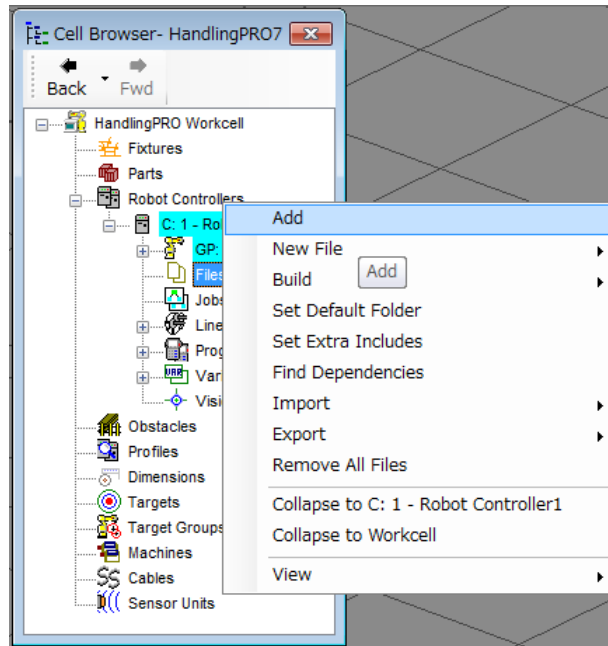
- 3 Build the created KAREL source without editing anything.



The “support” folder which includes the sample KAREL program (pksnlibcst.kl) is created in Robot_1 folder of the workcell as blow.



- 4 Add psnlibcst.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.

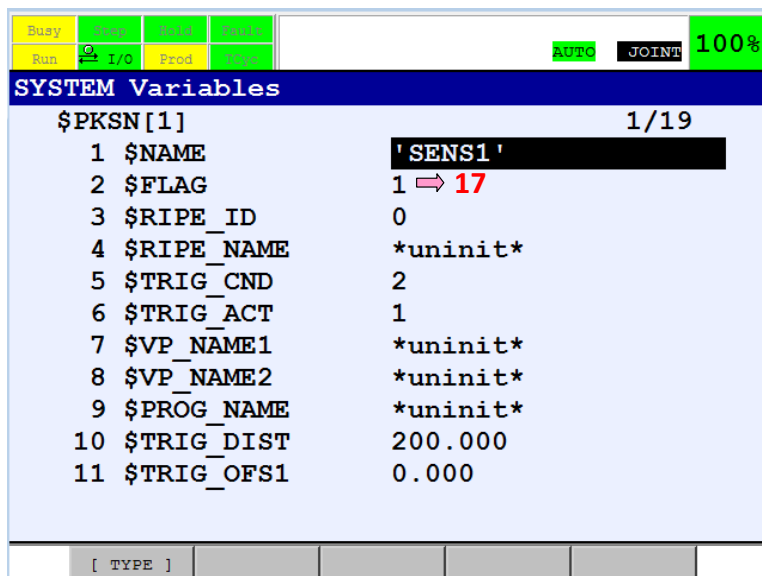
Here, all of following steps are performed in a real robot controller.

- 7 Load the built pksnlibcst(.PC) to the real controller from the FILE menu of TP during the cold start mode.

NOTE

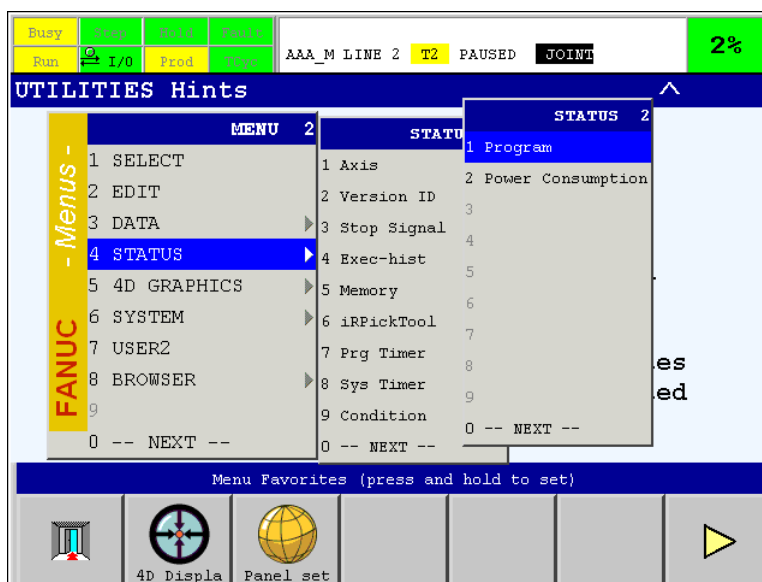
The built KAREL sources loaded from the FILE menu during the cold start mode can be saved by “All of above” backup.

- 8 Perform the following settings in order to execute the customized program as an action.
If you log in the iRPickTool setup page, please log out it before the following procedure.
If you do not create sensor object in iRPickTool setup page, please create it before the following procedure. You set BIT4(=16) in \$PKSN[n].\$FLAG. 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**

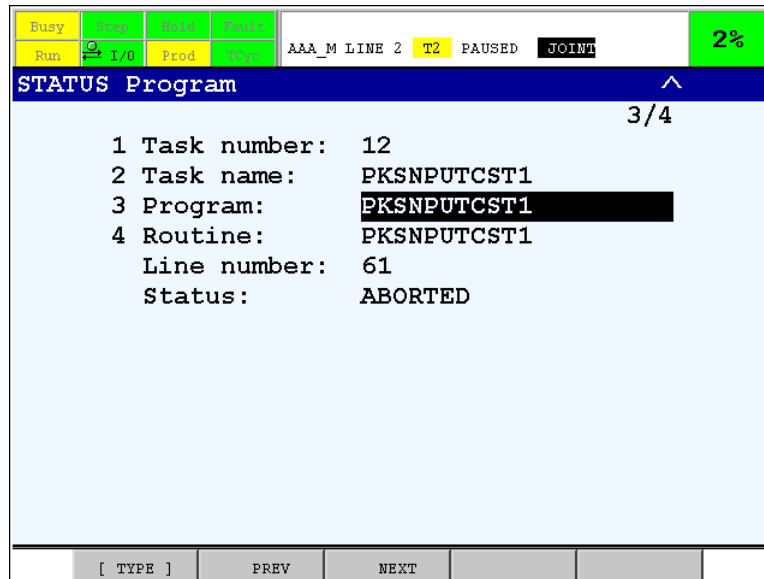
If you set the above bit with keeping logging in iRPickTool setup page, the bit can be reset when you change the sensor parameter in the setup page. If the bit is reset, please set the bit again after you log out the setup page.

- 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.
Select “Program” in “STATUS” menu as below.

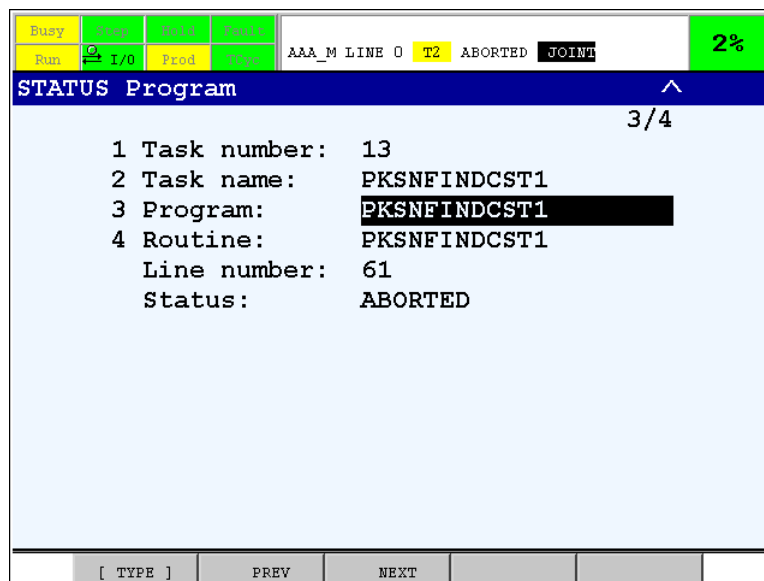


Execute the action one or more times.

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 disabled and “SENS3” is enabled and their settings are stored in the system variable as below

\$PKSN[1].\$NAME = “SENS1” (Enabled)

\$PKSN[2].\$NAME = “SENS2” (Disabled)

\$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.

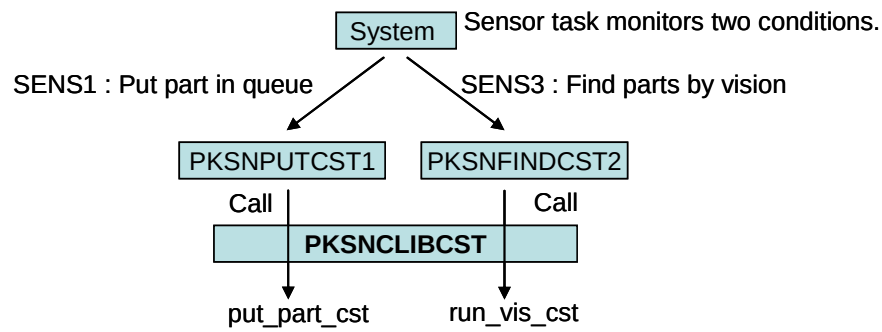


Fig. I.2 The flow of calling when “SENS1” and “SENS3” are enabled and their conditions are met.

I.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 ROUTINES or 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 pksnndef      -- Variable for sensor

%INCLUDE pksnlibl
%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

```

Here you can store the original value of your application.

```

    put_queue(sens_cfg[cond_id].cnv_id,
              sens_cfg[cond_id].stn_id,
              sens_cfg[cond_id].work_id,
              enc_count,
              model_id,
              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
      

Here you can store the original value  
of your application.


```

```

-- 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 = VE_XE_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

```

When you want to put a part to another conveyor according to model ID, you branch a process before put_queue. Regarding getting conveyor ID and conveyor station ID, you can use P_CVGETID and P_CSGETID BUILT-In described later.

I.4 ROUTINES

I.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.

I.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.

I.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 encoder count.

I.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.

I.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.

I.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.

I.4.7 put_queue

Purpose:

This puts part information into a tracking queue and outputs a duplication status that indicates whether the new part is put into the queue or not put because of double detection. 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 iRVision 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.

I.5 BUILT-INS

I.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.

I.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.

I.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.

I.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.

I.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.

INDEX

<Number>

- 4D pick boundaries 506
- 4D pick frames 507

<A>

- αA1000S PULSECODER.....596
- Adding a Tracking Frame 120,330
- Adjustment of the Dynamic Error of the Robot 181,349
- Adjustment of the Tracking Frame Error 179,215,347,450

- BASIC START-UP PROCEDURE.....27
- BUFFER FUNCTION.....508
- Buffer KAREL Macros 511
- Buffer TP Programs 513
- BUILT-INS 631

<C>

- Calibrating the camera 97,308
- CAMERA INSTALLATION AND CONNECTION.....48
- Case of circular conveyor.....333
- Caution on Setting up a Network 29
- CHANGING SETTINGS OF INTERNET EXPLORER
.....543
- CHANGING SETTINGS OF WINDOWS FIREWALL548
- Changing the Workcell Scene Using Recipes 507
- CHECKING THE PRODUCTION STATUS 480
- CHECKING THE ROBOT TORQUE 491
- Checking the Status of a Conveyor 481
- Checking the Status of a Conveyor Station.....484
- Checking the Status of a Fixed Station 487
- Checking the Status of a Sensor 483
- Circular Conveyor.....78,122,290
- CIRCULAR TRACKING 244
- COMMUNICATION CABLE 540
- COMPENSATION DEVIATES FOR A PART NEAR
THE CONVEYOR EDGES 534
- COMPENSATION DEVIATES WHEN A PART
ROTATES 534
- CONFIGURATION OF SOFTWARE OPTIONS 4
- CONFIGURATIONS 5
- Connecting a Network Cable 28
- Connecting the Pulsecoder 38
- Connecting to Pulse Multiplexer 611
- Connecting to Robot Controller Directly 609
- CONNECTION OF THE COMMUNICATION
CABLE.....540
- Continuous Termination Type Specification for a
Taught Position.....183
- Conveyor Station.....177,345
- Conveyor Station Robot Operation 268
- Conveyor Station-Related Programs 140,373
- Conveyor-Related Programs 130,362
- Creating the 4D Workcell 495

CUSTOMIZATION STEPS 620

<D>

- Deleting the 4D Workcell 507
- Detail Setting of a Recipe 465
- Displaying the 4D Workcell.....493
- Dynamic Error Tune Variable.....585

<E>

- EFFICIENCY TOOLS 522
- Enabling High Speed Scanning.....563
- ESSENTIAL POINTS OF CUSTOMIZATION 625
- Example of Main Program 585
- EXCHANGE WORKCELL WITH USING RECIPE..463

<F>

- FIGURES.....599,608
- FINE ADJUSTMENT OF THE TRACKING MOTION
.....179,215,347,450
- Fixed Station 178,346
- Fixed Station Robot Operation.....272
- Fixed Station-Related Programs.....151,387
- FUNCTION FOR SERVO CONVEYOR LINE
TRACKING 585

<G>

- get_ccnt_idx.....629
- GRID PATTERN 50
- Gripper and Zone 260
- Gripper-Related Programs 355

<H>

- HARDWARE CONNECTION OF R-30iB 559
- HARDWARE CONNECTION OF R-30iB Mate 561
- Hardware to which to connect a Pulsecoder 38
- HIGH SPEED DI (HDI).....559
- HOW TO CONNECT 609
- HOW TO CONNECT TO R-30iB 601
- HOW TO CONNECT TO R-30iB Mate.....603
- How to Create TP Program for Servo Conveyor.....577

</>

- IMAGE SNAP TIMING IS DELAYED 535
- incre_wid 630
- INCREMENTAL PULSECODER.....605
- Independent Extended Axis Setup 565
- Initial Fine Adjustment 179,347
- INPUT SIGNAL 562
- Installing a Pulsecoder 37
- iRPickTool 4D GRAPHICS DISPLAY 491

<K>

- KAREL Macros 524
- KAREL Program.....479
- KAREL Program for Servo Conveyor Line Tracking..586
- KAREL PROGRAM LIST 619

KAREL PROGRAMS.....	126,206,350,451	PK_CYCSTOP1.TP.....	429
KEY CONCEPTS	9,189,432	PK_DROP1.TP	407
<L>		PK_FS_DROP11.TP.....	424
Limitations	239,244,508	PK_FS_GET_DROP_DATA11.TP	412
LIMITATIONS ON SERVO CONVEYOR LINE		PK_FS_GET_PICK_DATA11.TP.....	411
TRACKING	595	PK_FS_INDEX.TP	427
LIMITING THE MOVABLE RANGE OF THE FINAL		PK_FS_PICK11.TP	422
AXIS.....	488	PK_FS_READY.TP.....	428
Line Conveyor.....	74,120,286,330	PK_GR_GET_ENDZONE1.TP.....	429
LINE_COUNT.....	631	PK_GR_GET_UTOOLS1.TP	407
Linear Distance Specification	184	PK_INIT1.TP.....	403
Linear Face Plate Instruction.....	184	PK_MAIN1.TP: Main Program for Group 1.....	402
List of all the TP Programs	402	PK_PERCH1.TP	405
Loading Custom CAD Files.....	508	PK_PICK1.TP.....	406
<M>		PK_PRGR_CV_PICKDROP1.TP.....	456
Main program.....	160,165,168	PK_PRGR_CV_PICKVTRAY1.TP	459
Maximum Linear Speed Function.....	185	PK_PRGR_INIT1.TP.....	456
MULTI-VIEW	239	PK_PRGR_MAIN1.TP.....	454
<N>		PKABO.PC	352
NETWORK CREATION.....	28	PKCSACKQPRGR.PC	208,453
Numeric Registers	339	PKCSACKQUE.PC	143,376
<O>		PKCSCALWPOS.PC.....	383
OPENING iRPickTool SCREENS.....	550	PKCSCHKPOS.PC	526
OPENING iRVision SCREENS.....	551	PKCSCLRACK.PC.....	384
OPERATING SYSTEM AND BROWSER.....	540	PKCSCLRQUE.PC.....	141,373
OTHER FUNCTIONS	463	PKCSGENVRTRY.PC	206,451
OVERVIEW	3,563,565,620	PKCSGETAPPR.PC	384
OVERVIEW OF THE MANUAL	1	PKCSGETDIST.PC	146,379
<P>		PKCSGETENC.PC	150,387
P_CSGETID.....	632	PKCSGETID.PC.....	140,373
P_CVGETID	633	PKCSGETNPRT.PC.....	149,381
Part Detection Procedure.....	239	PKCSGETOPPR.PC	385
Payload IDs.....	341	PKCSGETPFRT.PC.....	148,380
Pick Only Reachable Parts	522	PKCSGETPMUL.PC	385
Pick program	161,166,170	PKCSGETPOSCK.PC	524
Pick program (Buffer).....	172	PKCSGETPPCHK.PC	385
PICKING A TRAY ONLY WHEN ALL CELLS ARE		PKCSGETQCELL.PC	142,375
FILLED	237	PKCSGETQINRG.PC	524
PICKING PARTS FROM TWO OR MORE INFEED		PKCSGETQPRGR.PC	206,452
CONVEYORS.....	233	PKCSGETQTY.PC.....	386
PK_BF_DROP1.TP.....	515	PKCSGETQUE.PC.....	141,374
PK_BF_GET_DATA1.TP	513	PKCSGETTIME.PC	145,377
PK_BF_OK2DROP1.TP.....	518	PKCSGETTRID.PC.....	150,387
PK_BF_OK2PICK1.TP	517	PKCSGETVR.PC	386
PK_BF_PICK1.TP.....	513	PKCSGETWPOS.PC	386
PK_BUF_DROP1.TP.....	521	PKCSPUTQUE.PC	144,377
PK_BUF_PICK1.TP	521	PKCSSETDSLIN.PC.....	149,382
PK_CV_DROP11.TP.....	417	PKCSTRGPRGR.PC	207,452
PK_CV_GET_DROP_DATA11.TP	409	PKCVENBLB.PC	132,364
PK_CV_GET_PICK_DATA11.TP.....	408	PKCVENBSC.PC	133,365
PK_CV_OK2DROP1.TP	519	PKCVGETNAME.PC.....	366
PK_CV_OK2PICK1.TP.....	518	PKCVSETLBD.PC	130,362
PK_CV_PICK11.TP	413	PKCVSETTRNAM.PC.....	131,363
PK_CV_WAITPOS1.TP.....	421	PKFSACKBUF.PC	155,392
		PKFSACKQUE.PC.....	153,389
		PKFSCLRBUF.PC.....	154,390
		PKFSCLRQUE.PC	151,388
		PKFSGETAPPR.PC.....	393

PKFSGETBUF.PC	154,391	PKWCSTART.PC.....	126,350
PKFSGETBUFD.T.PC.....	512	Placement program	164,166,171,172
PKFSGETBUFP.T.PC	512	PLACING PARTS IN DIFFERENT DIRECTIONS IN	
PKFSGETID.PC	151,387	A BOX	227
PKFSGETIXDLY.PC	393	PLACING PARTS IN DIFFERENT PLACES OF A	
PKFSGETIXODO.PC	393	BOX DEPENDING ON THE MODEL ID	228
PKFSGETIXSIM.PC	394	PLACING PARTS IN LAYERS	224
PKFSGETMONIDX.PC	397	PLACING PARTS IN SEQUENCE FROM THE END	
PKFSGETOPPR.PC.....	394	OF THE BOX.....	226
PKFSGETPMUL.PC.....	394	PLACING PARTS ON DIFFERENT OUTFEED	
PKFSGETPPCHK.PC.....	395	CONVEYORS DEPENDING ON THE MODEL ID	231
PKFSGETQTY.PC	395	PLC Recipe Changeovers	476
PKFSGETQUE.PC	152,388	Position Registers.....	339
PKFSGETRDY.PC	395	PREFACE	1
PKFSGETRDYDI.PC	396	PREPARATION	28
PKFSGETTRID.PC	396	Preparation for Camera	239
PKFSGETUF.PC	396	Production Status	244
PKFSGETVR.PC	397	PULSECODER INSTALLATION, CONNECTION	
PKFSPUTBUF.PC	153,390	AND SETUP	37
PKFSPUTQUE.PC.....	151,388	put_part_cst.....	628
PKFSSETRDY.PC.....	397	put_queue.....	630
PKGETVAR.PC.....	128,353		
PKGRCHKPP.PC	355	<Q>	
PKGRCHKQTY.PC.....	356	Queue Managed Tracking	5
PKGRCLOSE.PC.....	357		
PKGRGETDPDLY.PC	357	<R>	
PKGRGETID.PC	357	R-30iB Controller	596,605
PKGRGETPKDLY.PC	358	R-30iB Mate Controller	598,607
PKGRGETZNCNT.PC	358	REFERENCE POSITION SETTING	173,215,341,450
PKGRGETZNUT.PC	358	Reference position setting and robot position teaching	242
PKGRINIT.PC	359	RELATED MANUALS.....	1
PKGROPEN.PC.....	359	REPORTING FAILURE TO PICK A PART	236
PKLABO.PC.....	352	REQUIREMENTS.....	596,605
PKOPGETEN.PC.....	353	RESTRICTIONS	27
PKOPPOSCHK.PC.....	527	Robot Behavior Under the Standard TP Programs.....	337
PKPOSTERR.PC	352	Robot program	241
PKPRGRGETEN.PC	130,355	ROBOT PROGRAMS	158,209,337,450
PKRBGETBUFID.PC.....	512	Robot-Related Programs	359
PKRBGETCVCNT.PC	359	ROUTINES.....	628
PKRBGETFSCNT.PC	360	Run Multi-View System	243
PKRBGETID.PC	360	run_vis_cst.....	628
PKRBGETPRGRT.PC.....	361		
PKRBGETSFLG.PC	361	<S>	
PKRCSWITCH.PC	479	SAFETY PRECAUTIONS	s-1
PKRSTWC.PC	351	Sample Program.....	528
PKSETVAR.PC	129,354	Sample Programs for the	
PKSNENBSENS.PC	134,367	Conveyor-to-Buffer-to-Conveyor Configuration	168
PKSNSETMTOFS.PC	138,371	Sample Programs for the Conveyor-to-Conveyor	
PKSNSETMVNAM.PC	136,369	Configuration	159,213
PKSNSETVPNAM.PC	135,368	Sample Programs for the Conveyor-to-Fixed Station	
PKSNSETVTOFS.PC	137,370	Configuration	164
PKSNSTART.PC	139,372	Sample programs for the conveyor-to-the same	
PKSNSTOP.PC	139,372	conveyor configuration (downstream robots).....	212
PKTRGETNUMCL.PC.....	157,400	Sample programs for the conveyor-to-the same	
PKTRSETCLPOS.PC	156,398	conveyor configuration (the first robot)	209
PKWCHECK.PC.....	127,351	SAMPLE SYSTEM CONFIGURATIONS.....	14
PKWCEND.PC	127,351	SENSOR CUSTOMIZATION.....	620
PKWCRSTPRDST.PC.....	127,351	Sensor-Related Programs	134,367

SEPARATE DETECTOR UNIT (SDU)	613
SERVO CONVEYOR LINE TRACKING FUNCTION	565
Servo Conveyor Setup	572
Setting a Recipe	463
Setting a Sensor in Detail	83,295
SETTING A TEACHING PC	540
Setting a Tracking Frame	73,285
SETTING AN IP ADDRESS FOR THE PC	541
Setting IP Addresses	30
Setting of PLC recipe change data	477
SETTING OF THE TCP	49
Setting the Sensor Position	92,303
Setting the Target Position	112,322
Setting the Tray Position	108,318
Setting up a Buffer	510
Setting up conveyor station robot operation in detail	270
Setting up fixed station robot operation in detail	273
Setting up Pick Only Reachable Parts	523
Setting up the camera	96,307
Setting up the gripper in detail	264
Setting up the Pulsecoder	45
Setting up the Robot Ring	31
Setting Up the Vision System	96,307
Setting up the zone in detail	265
Settings of sensor task	241
Settings Required for Each Combination of a Trigger	
Condition and a Trigger Action	91,303
Setup	240,245,565
SETUP OF A CONVEYOR	70,191,282,435
SETUP OF A CONVEYOR STATION	116,199,326,444
SETUP OF A FIXED STATION	125,206,336,450
SETUP OF A GRIPPER AND A ZONE	260,433
SETUP OF A ROBOT	63,267,433
SETUP OF A ROBOT OPERATION	268,433
SETUP OF A SENSOR	83,198,295,443
SETUP OF A TRAY	64,189,276,434
SETUP OF A WORKCELL	60,257,433
Setup of a Workcell and a Robot	189
SETUP OF HDI	563
SETUP OF PLUG & PLAY FUNCTIONS	248
SETUP OF PRE-GROUPING	187,430
SETUP OF THE BASIC FUNCTIONS	51
SKIPPING THE TRACKING MOTION OUTSIDE	
THE AREA	488
STANDARD TP PROGRAMS	400,454
start_srv	629
String Registers	340
STUDY FOR APPLICATION	21
SYSTEM STARTUP	186
SYSTEM VARIABLE FILES	539
<T>	
Teaching a GPM locator tool	107,317
Teaching a vision process	105,316
TEACHING THE POSITION TO ROBOTS	177,215,345,450
The Buffer Algorithm	509
THE OVERRUN COUNT CONTINUES TO	
INCREASE	535
THE STOP CONVEYOR DOES NOT WORK	535
Time Before Instruction	186
Tool Frames	341
TP Program Details	402
TP Program Naming Conventions	400
Tracking Program	158
Tracking Schedule Setup	585
Tray-Related Programs	156,398
TROUBLESHOOTING	532,555
<U>	
update_nfind	629
User Alarms	341
USING DIFFERENT ROBOTS TO PICK PARTS	
DEPENDING ON THEIR CHARACTERISTICS	220
USING DIFFERENT ROBOTS TO PICK PARTS	
DEPENDING ON THEIR POSITIONS ON THE	
CONVEYOR	217
Using the 4D [TYPE] Scenes	506
Using the Touch Screen Icons	505
Usrer Frames	341
<V>	
V_RUN_FIND	631
VARIATIONS	217
Vision process	105,240,315
Vision Registers	340
Visual Tracking	7
VT_GET_FOUND	632
<W>	
Wait Indexer Stop Function	585
When [DI] and [Find part by vision] are selected	110,175,320,343
When [DI] or [HDI] and [Put part in queue] Are	
Selected	176,344
When [Distance] and [Find part by vision] are selected	
.....	108,174,318,342
When [Distance] or [Flag] and [Put part in queue] Are	
Selected	177,345
When a tray is not used	92,112,303,323
When a tray is used	94,114,305,324
When multiple robots are used	44
When one robot is used	42
WHEN PARTS ARE MISSED FREQUENTLY	532
WHEN ROBOT MOTION IS NOT SMOOTH	534
WHEN SETTINGS CANNOT BE CHANGED	
DURING PRODUCTION	536
WHEN THE ROBOT DOES NOT PICK UP PARTS	532
WHEN THE SAME PART IS PICKED UP TWICE	534
When the vision system is used	246
Workcell-Related Programs	126,350

REVISION RECORD

Edition	Date	Contents
04	Sep., 2016	• Addition of 6.16 "SETUP OF PRE-GROUPING" and etc.
03	Jun., 2016	• Addition of 5.11 "MULTI-VIEW" and etc.
02	Sep., 2014	• Addition of 4.14 "SETUP OF PRE-GROUPING" and etc.
01	Dec., 2013	

B-83604EN/04



* B - 8 3 6 0 4 E N / 0 4 *