FANUC Robot series

R-50*i*A CONTROLLER

iRPickTool

OPERATOR'S MANUAL

B-84454EN/04

© FANUC CORPORATION, 2024

Important notice

Translation of the original instruction
The original language of this manual is English. The other language manuals are a translation from the English one.

Thank you very much for purchasing FANUC Robot.

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

- No part of this manual may be reproduced in any form.
- The appearance and specifications of this product are subject to change without notice.
- Any copyright and other intellectual property rights regarding this manual shall belong to FANUC.
- The safety precautions described in this manual may no longer be accurate and up-to-date due to revisions or changes in laws and regulations. Before using our products, each user must confirm the latest laws and regulations.

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

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

Safety precautions

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

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

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

Personnel

Personnel can be classified as follows.

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

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

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

The operator is not allowed to work in the safeguarded space.
The programmer or teaching operator and maintenance technician are allowed to work in the safeguarded space.
Based on the "Ordinance on Industrial Safety and Health" in Japan or laws in each country, to work inside the robot set safeguarded space, the followings is required:

- **Perform risk assessment for the entire robot system.**
- **Based on the risk assessment result, all personnel must take appropriate professional safety training courses.**

Following table lists the work outside the safeguarded space. In this table, the symbol “○” means the work allowed to be carried out by the specified personnel.

Table 2-1 List of work outside the Safeguarded Space

	Operator	Programmer /Teaching operator	Maintenance technician
Turn power ON/OFF to Robot controller	○	○	○
Select operating mode (AUTO/T1/T2)		○	○
Select remote/local mode		○	○
Select robot program with teach pendant		○	○
Select robot program with external device		○	○
Start robot program with operator’s panel	○	○	○
Start robot program with teach pendant		○	○

	Operator	Programmer /Teaching operator	Maintenance technician
Reset alarm with operator's panel		○	○
Reset alarm with teach pendant		○	○
Set data on teach pendant		○	○
Teaching with teach pendant		○	○
Emergency stop with operator's panel	○	○	○
Emergency stop with teach pendant	○	○	○
Operator's panel maintenance			○
Teach pendant maintenance			○

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

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

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.

⚠WARNING

Used if a danger resulting in the death or serious injury of users are expected to occur if they fail to observe the approved procedure.

⚠CAUTION

Used if a danger resulting in the minor or moderate injury of users or equipment damage is expected to occur if they fail to observe the approved procedure.

NOTE

Used if a supplementary explanation not related to any of WARNING and CAUTION is to be indicated.

Preface

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

Overview of the manual

This manual describes how to operate the iRPickTool function in the R-50iA or R-50iA Mate Controller.

This manual summarizes the settings and the programming procedure in the iRPickTool function, assuming that the installation and setup of the robot to be used is completed. When you want to set up a robot system that uses the iRPickTool function, see the manuals introduced in "Related manuals(P.vi)".

⚠CAUTION

This manual is based on the R-50iA or R-50iA Mate Controller with software version 7DH1/28 (V10.10P/28). Note that functions and settings not described in this manual may be available, and some notation differences are present, depending on the software version. -

How to use this manual

This manual explains the procedures, etc. to program tracking systems that require vision or queue management.

For the 'iRPickTool Basic' setup procedures, see Chapter 2.

For the 'iRPickTool' setup procedures, see Chapter 3.

See "I.1.1. Configuration of software options(P.4)" for the difference between the 'iRPickTool Basic' and the 'iRPickTool'.

Chapters 4, 5, and 6 explain the details of the procedures and functions that are not explained in the setup procedures in Chapters 2 and 3.

Furthermore, Chapter 7 explains specific setting examples that suit each application, and Chapter 8 explains other useful functions of iRPickTool. After you have completed iRPickTool startup, refer to this manual in accordance with your intended use.

Contents of this manual

Chapter	Title	Description
	Preface	Gives an explanation of this manual and related manuals. In particular, please read the precautions.
Chapter 1	Overview	Gives an overview of iRPickTool and some configuration examples using iRPickTool.
Chapter 2	iRPickTool Basic (basic functions) startup procedures	Explains from preparation to setup for starting up iRPickTool Basic. Refer to this chapter if you are using iRPickTool Basic for the first time.
Chapter 3	iRPickTool (basic functions and Plug & Play) startup procedures	Explains from preparation to setup for starting up iRPickTool. Refer to this chapter if you are using an iRPickTool with Plug & Play for the first time.
Chapter 4	Other preparations before setup	Explains preparations other than the preparations in Chapters 2 and 3.
Chapter 5	Basic functions reference	Explains each function of the basic functions and details on screen items.
Chapter 6	Plug & Play reference	Explains each function of Plug & Play and gives details on tree view items.
Chapter 7	Setting example appropriate to the usage	Explains customizations for advanced robot functionality iRPickTool, and gives specific examples.
Chapter 8	Other useful functions	Explains useful functions of iRPickTool other than the functions explained up to Chapter 7.
Chapter 9	Troubleshooting	Explains common problems that may occur in tracking systems created using iRPickTool, and measures for resolving them.

Chapter	Title	Description
Appendix A to I	The appendices explain the following items. System variable files / Setting up a teaching PC / Pulsecoders and their connection / Correspondence table for KAREL used in the previous software and in iRPickTool / Sensor task customization / Vision I/F add-on	

Indications in this manual

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

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

Related manuals

This section introduces related manuals.

Manual (spec. no.)	Description
"OPERATOR'S MANUAL(Basic Operation)"(B-84464EN)	This is the main manual of the controller. This manual describes the following items for manipulating parts with the robot: • Setting up the robot system • Operating the robot • Creating and changing a program • Executing a program • Status indications • Backup and restore robot programs. This manual is used in applied design, installation, and teaching of robots.
"R-50iA CONTROLLER MAINTENANCE MANUAL"(B-84375EN)	This manual describes the maintenance and connection of the R-50iA Controller.
"OPERATOR'S MANUAL(Alarm code list)" (B-84464EN-1)	This manual describes the error code listings, causes, and remedies. It describes the causes of alarms and how to address them.
"Optional Function OPERATOR'S MANUAL"(B-84464EN-2)	This manual describes the software optional functions.
"Sensor Mechanical Unit / Control Unit OPERATOR'S MANUAL"(B-84544EN)	This manual describes the connection between sensors, which may be a camera or 3D Laser Sensor, and a controller, and maintenance of sensors.
"iRVision OPERATOR'S MANUAL (Reference)"(B-84434EN)	This manual is the reference manual for iRVision. This manual describes each function provided by iRVision. This manual describes the meanings (e.g., the items on the iRVision setup screen and the arguments of the instruction).
"iRVision 2D Camera Application OPERATOR'S MANUAL"(B-84434EN-2)	We ask that you refer to this manual first when you start up the iRVision 2D Compensation and 2.5D Compensation systems. This manual describes startup procedures, how to create programs, cautions, technical know-how, responses to several cases, and so on for the iRVision 2D Compensation and 2.5D Compensation systems.
"iRVision Inspection Application OPERATOR'S MANUAL"(B-84434EN-4)	We ask that you refer to this manual first when you start up inspection systems that use iRVision. This manual describes the startup procedures, how to create programs, cautions, technical know-how, responses to several cases, and so on for inspection systems that use iRVision.
"Network and Security Function OPERATOR'S MANUAL"(B-84604EN)	This manual describes robot networking options such as FTP, RIPE, iRConnect, and PC Share.
"Line Tracking OPERATOR'S MANUAL"(B-84464EN-10)	This manual describes the line tracking function that the tracking system with iRPickTool is based on.

Regarding Microsoft® products

The abbreviations below are used in this manual.

Microsoft® Windows® 10 Pro: Windows 10
Microsoft® Windows® 11 Pro: Windows 11
Windows® operating system: Windows

Regarding trademarks

Microsoft Edge, Windows, Windows 10 Pro, and Windows 11 Pro are registered trademarks or trademarks of Microsoft Corporation in the United States and in all other countries.
Google Chrome is a registered trademark of Google LLC.

Table of contents

Important notice ii
Safety precautions iii
Preface v
Overview of the manual v
How to use this manual v
Related manuals vi
Regarding Microsoft® products vii
I Main part 1
1. Overview 3
1.1. Configuration of software options 4
1.2. Configurations 4
1.2.1. Queue managed tracking 4
1.2.2. Visual tracking 6
1.3. Terms 7
1.4. Sample system configurations 12
1.5. Study for application 19
1.6. Before use 24
1.6.1. Restrictions 24
1.6.2. Restrictions 24
1.6.2.1. Restrictions of SpotTool+ 25
1.6.3. Recommendations 25
2. iRPickTool Basic (basic functions) startup procedures 26
2.1. Access to the robot homepage 27
2.2. Preparation before setting up the application 27
2.2.1. Connection and setup of Pulsecoders 27
2.2.1.1. Connecting Pulsecoders 28
2.2.1.2. Setting up Pulsecoders 28
2.2.1.3. Checking the Pulsecoder settings 32
2.2.2. Checking the connection of the camera 32
2.2.3. Checking the input of the trigger sensor 36
2.2.3.1. Using DI / RI 36
2.2.4. Setting up the tool frame of the pointer tool 37
2.2.5. Creating a robot program 42
2.2.6. Setting payloads (motion performance) 58
2.3. Setting up applications 60
2.3.1. Setting up a workcell 60
2.3.2. Setting up a tray 67
2.3.3. Setting up a conveyor 71
2.3.3.1. Setting up an infeed conveyor 71
2.3.3.2. Setting up an outfeed conveyor 72
2.3.4. Setting a tracking frame 72
2.3.5. Setting up a sensor 80
2.3.5.1. Setting an infeed conveyor sensor 80
2.3.5.2. Setting up an outfeed conveyor sensor (In the case of [DI] / [RI] + [Put part in queue]) 81
2.3.5.3. Setting up an outfeed conveyor sensor (In the case of [HDI] + [Put part in queue]) 83
2.3.6. Setting the sensor position 84
2.3.7. Setting up the vision system 89
2.3.7.1. Camera calibration 89
2.3.7.2. Vision process 99
2.3.8. Setting up a conveyor station 106

2.3.9. Setting up a tool frame for the hand ... 111
2.3.10. Setting up a reference position and teaching a robot position ... 113
2.3.10.1. Setting up a reference position for the infeed conveyor and teaching a robot position ... 113
2.3.10.2. Setting up a reference position for the outfeed conveyor and teaching a robot position ... 122
2.3.11. Fine adjustment of the tracking motion ... 127
2.3.11.1. Checking and fine adjustment of the tracking motion ... 127
2.3.11.2. Adjustment of the tracking frame error ... 131
2.3.11.3. Adjustment of the dynamic error of the robot ... 139
3. iRPickTool (basic functions and Plug & Play) startup procedures ... 152
3.1. Access to the robot homepage ... 153
3.2. Preparation before setting up the application ... 153
3.2.1. Connection and setup of Pulsecoders ... 153
3.2.1.1. Connecting Pulsecoders ... 154
3.2.1.2. Setting up Pulsecoders ... 154
3.2.1.3. Checking the Pulsecoder settings ... 158
3.2.2. Checking the connection of the camera ... 158
3.2.3. Checking the input of the trigger sensor ... 162
3.2.3.1. Using DI / RI ... 162
3.2.4. Setting up the tool frame of the pointer tool ... 163
3.2.5. Setting payloads (motion performance) ... 168
3.3. Setting up applications ... 170
3.3.1. Setting up a workcell ... 170
3.3.2. Setting up a gripper ... 178
3.3.3. Setting up a tray ... 181
3.3.4. Setting up a conveyor ... 185
3.3.4.1. Setting up an infeed conveyor ... 185
3.3.4.2. Setting up an outfeed conveyor ... 186
3.3.5. Setting up a tracking frame ... 186
3.3.6. Setting up a sensor ... 194
3.3.6.1. Setting an infeed conveyor sensor ... 194
3.3.6.2. Setting up an outfeed conveyor sensor (In the case of [DI] / [RI] + [Put part in queue]) ... 195
3.3.6.3. Setting up an outfeed conveyor sensor (In the case of [HDI] + [Put part in queue]) ... 196
3.3.7. Setting up a sensor position ... 197
3.3.8. Setting up the vision system ... 203
3.3.8.1. Camera calibration ... 203
3.3.8.2. Vision process ... 213
3.3.9. Setting up a conveyor station ... 220
3.3.10. Setting up a robot operation ... 225
3.3.11. Setting up a tool frame for the hand ... 226
3.3.12. Setting up a reference position and teaching a robot position ... 230
3.3.12.1. Setting up a reference position for the infeed conveyor, and teaching a robot position ... 230
3.3.12.2. Setting up a reference position for the outfeed conveyor, and teaching a robot position ... 235
3.3.12.3. Teaching the robot home position ... 240
3.3.13. Fine adjustment of the tracking motion ... 240
4. Other preparations before setup ... 241
4.1. Network creation ... 241
4.1.1. Connecting a network cable ... 241
4.1.2. Caution on setting up a network ... 242
4.1.3. Setting up the IP address ... 242
4.1.4. Setting up the robot ring ... 246
4.2. Pulsecoder installation, connection and setup ... 252
4.2.1. Installing a Pulsecoder ... 253

4.2.2. Connecting the Pulsecoder ... 253
4.2.3. Setting up Pulsecoders that are connected to the robot controller ... 253
4.2.4. Setting up Pulsecoders for multiple robots ... 254
4.3. Camera installation and connection ... 255
4.4. Setting up the tool frame ... 257
4.5. Calibration grid ... 257
5. Basic functions reference ... 258
5.1. Setting up a workcell ... 266
5.1.1. Setting the number of objects ... 266
5.1.2. Recipe ... 268
5.2. Setting up a robot ... 268
5.3. Setting up a tray ... 269
5.3.1. Cell operation ... 272
5.3.2. Layer operation ... 273
5.3.3. Setting the reference cell ... 275
5.4. Setting up a conveyor ... 276
5.4.1. Setting up a tracking frame ... 283
5.4.1.1. In the case of a circular conveyor ... 283
5.5. Setting up a sensor ... 288
5.5.1. Setting a sensor in detail ... 289
5.5.1.1. Distance and Put part in queue ... 290
5.5.1.2. Distance and Find part by vision ... 291
5.5.1.3. Distance and Run program ... 291
5.5.1.4. DI and Put part in queue ... 292
5.5.1.5. DI and Find part by vision ... 293
5.5.1.6. DI and Find part by multiview ... 294
5.5.1.7. DI and Run program ... 295
5.5.1.8. HDI and Put part in queue ... 296
5.5.1.9. Flag and Put part in queue ... 297
5.5.1.10. RI and Put part in queue ... 297
5.5.1.11. RI and Find part by vision ... 298
5.5.1.12. RI and Find part by multiview ... 300
5.5.1.13. RI and Run program ... 301
5.5.2. Settings required for each combination of a trigger condition and a trigger action ... 302
5.5.3. Setting up a sensor position ... 302
5.5.3.1. When a tray is not used ... 303
5.5.4. Setting up the vision system ... 305
5.5.5. Setting the tray position ... 306
5.5.5.1. When [Distance] and [Find part by vision] are selected ... 306
5.5.5.2. When [DI] and [Find part by vision] are selected ... 309
5.5.6. Setting the target position ... 312
5.5.6.1. When a tray is not used ... 312
5.5.6.2. When a tray is used ... 314
5.5.7. Restriction concerning sensors that can be operated simultaneously ... 317
5.6. Setting up a conveyor station ... 318
5.6.1. Adding a tracking frame ... 326
5.6.1.1. In the case of a line conveyor ... 326
5.6.1.2. In the case of a circular conveyor ... 329
5.7. Setting up a fixed station ... 332
5.8. KAREL program ... 332
5.8.1. Workcell-related programs ... 333
5.8.1.1. PKWCSTART.PC ... 333
5.8.1.2. PKWCEND.PC ... 333
5.8.1.3. PKWCCHECK.PC ... 333
5.8.1.4. PKWCRSTPRDST.PC ... 334
5.8.1.5. PKPRGRGETEN.PC ... 334

5.8.1.6. PKRSTWC.PC 334
5.8.1.7. PKABO.PC 334
5.8.1.8. PKLABO.PC 335
5.8.1.9. PKPOSTERR.PC 335
5.8.1.10. PKCOPYPR2POS.PC 335
5.8.1.11. PKWCGETACTRC.PC 336
5.8.1.12. IRVGETVRWRKID.PC 336
5.8.1.13. PKSETSYNCINT.PC 336
5.8.1.14. PKENBSPFUNC.PC 337
5.8.1.15. PKDSBSPFUNC.PC 337
5.8.2. Robot-related programs 338
5.8.2.1. PKRBGETID.PC 338
5.8.2.2. PKRBGETSFLG.PC 338
5.8.2.3. PKRBSETSVDLY 338
5.8.2.4. PKRBSETSKPADJ 339
5.8.3. Conveyor-related programs 339
5.8.3.1. PKCVSETLBD.PC 339
5.8.3.2. PKCVSETTRNAM.PC 340
5.8.3.3. PKCVENBLB.PC 341
5.8.3.4. PKCVENBSC.PC 342
5.8.3.5. PKCVGETNAME.PC 343
5.8.3.6. PKCVIDXDELQ.PC 343
5.8.3.7. PKCVGETCNVSPD.PC 345
5.8.4. Sensor-related programs 345
5.8.4.1. PKSNENBSENS.PC 345
5.8.4.2. PKSNSETVPNAM.PC 346
5.8.4.3. PKSNSETVTOFS.PC 347
5.8.4.4. PKSNSETMTOFS.PC 348
5.8.4.5. PKSNSTART.PC 348
5.8.4.6. PKSNSTOP.PC 349
5.8.4.7. PKSNSETDBNC.PC 349
5.8.4.8. PKSNSETDURTN.PC 349
5.8.5. Conveyor station-related programs 350
5.8.5.1. PKCSGETID.PC 350
5.8.5.2. PKCSCLRQUE.PC 351
5.8.5.3. PKCSGETQUE.PC 351
5.8.5.4. PKCSGETQCELL.PC 352
5.8.5.5. PKCSACKQUE.PC 353
5.8.5.6. PKCSPUTQUE.PC 353
5.8.5.7. PKCSPUTQUE2.PC 354
5.8.5.8. PKCSGETTIME.PC 354
5.8.5.9. PKCSGETDIST.PC 355
5.8.5.10. PKCSGETPFRT.PC 356
5.8.5.11. PKCSGETNPRT.PC 357
5.8.5.12. PKCSSETDSLIN.PC 358
5.8.5.13. PKCSGETENC.PC 358
5.8.5.14. PKCSGETTRID.PC 359
5.8.5.15. PKCSCLRACK.PC 359
5.8.5.16. PKCSGETDSLIN.PC 359
5.8.5.17. PKCSGETPINFO.PC 359
5.8.6. Fixed station-related programs 360
5.8.6.1. PKFSGETID.PC 360
5.8.6.2. PKFSPUTQUE.PC 360
5.8.6.3. PKFSCLRQUE.PC 361
5.8.6.4. PKFSGETQUE.PC 361
5.8.6.5. PKFSACKQUE.PC 362
5.8.6.6. PKFSPUTBUF.PC 362

5.8.6.7. PKFSCLRBUF.PC ... 362
5.8.6.8. PKFSGETBUF.PC ... 363
5.8.6.9. PKFSACKBUF.PC ... 363
5.8.6.10. PKFSGETTRID.PC ... 364
5.8.6.11. PKFSGETUF.PC ... 364
5.8.6.12. PKFSCLRACK.PC ... 364
5.8.7. Tray-related programs ... 365
5.8.7.1. PKTRSETCLPOS.PC ... 365
5.8.7.2. PKTRGETNUMCL.PC ... 366
5.8.7.3. PKTRGETCL.PC ... 366
5.9. Robot programs ... 367
5.9.1. Tracking program ... 367
5.9.2. Sample programs for the conveyor-to-conveyor configuration ... 368
5.9.2.1. Main program ... 369
5.9.2.2. Pick program ... 370
5.9.2.3. Placement program ... 373
5.9.3. Sample programs for the conveyor-to-fixed station configuration ... 373
5.9.3.1. Main program ... 374
5.9.3.2. Pick pogram ... 375
5.9.3.3. Placement program ... 375
5.9.4. Sample programs for the conveyor-to-buffer-to-conveyor configuration ... 376
5.9.4.1. Main program ... 377
5.9.4.2. Pick pogram ... 378
5.9.4.3. Placement program ... 379
5.9.4.4. Pick program (buffer) ... 380
5.9.4.5. Placement program (buffer) ... 381
5.10. Reference position setting ... 382
5.10.1. DI / RI and Find part by vision ... 382
5.11. Teaching the position to robots ... 385
5.11.1. Conveyor station ... 385
5.11.2. Fixed station ... 386
5.12. Fine adjustment of the tracking motion ... 386
5.12.1. Checking and fine adjustment of the tracking motion ... 386
5.12.2. Adjustment of the tracking frame error ... 387
5.12.3. Adjustment of the dynamic error of the robot ... 387
5.12.4. Continuous termination type specification for a taught position ... 387
5.12.5. Linear distance specification ... 388
5.12.6. Linear face plate instruction ... 389
5.12.7. Maximum linear speed function ... 389
5.12.8. Time before instruction ... 390
5.13. Precautions for when starting up multiple robots ... 390
5.14. Production runs ... 391
5.14.1. Selecting the operator’s panel for starting production ... 391
5.14.1.1. Operator’s panel ... 391
5.14.1.2. Peripheral equipment ... 391
5.14.2. Pausing, resuming, and aborting production ... 392
5.14.2.1. Pausing production ... 392
5.14.2.2. Aborting production ... 392
5.14.2.3. Resuming production ... 392
5.14.3. In the case of dual arm robots ... 392
5.15. Setup of pre-grouping ... 393
5.15.1. Terms ... 396
5.15.2. Setup of a workcell and robot ... 396
5.15.3. Setting up a tray ... 396
5.15.4. Setting up a conveyor ... 397
5.15.5. Setting up a sensor ... 404

5.15.6. Setting up a conveyor station 405
5.15.7. Setting up a fixed station 411
5.15.8. KAREL program 411
5.15.8.1. PKCSGENVRTRY.PC 411
5.15.8.2. PKCSGETQPRGR.PC 411
5.15.8.3. PKCSTRGPRGR.PC 412
5.15.8.4. PKCSACKQPRGR.PC 412
5.15.9. Robot programs 413
5.15.9.1. Sample programs for the conveyor-to-same-conveyor configuration (the first robot) 414
5.15.9.2. Sample programs for the conveyor-to-same-conveyor configuration (downstream robots) 417
5.15.9.3. Sample programs for the conveyor-to-conveyor configuration 417
5.15.10. Reference position setting 419
5.15.11. Teaching the position to robots 419
5.15.12. Fine adjustment of the tracking motion 419
5.15.12.1. Adjustment of the tracking frame error 419
6. Plug & Play reference 421
6.1. Workcell setup screen 422
6.2. Setting up a gripper and a zone 422
6.2.1. Gripper and zone 423
6.2.1.1. Setting up the gripper in detail 425
6.2.1.2. Setting up the zone in detail 426
6.3. Setting up a robot operation 427
6.3.1. Conveyor station robot operation 428
6.3.1.1. Setting up conveyor station robot operation in detail 429
6.3.2. Fixed station robot operation 431
6.3.2.1. Setting up fixed station robot operation in detail 432
6.4. Setting up a fixed station 434
6.5. Robot programs 434
6.5.1. Robot behavior under the standard TP programs 435
6.5.2. Registers 436
6.5.3. Position Registers 436
6.5.4. String Registers 437
6.5.5. Vision Registers 438
6.5.6. Payload IDs 438
6.5.7. User alarms 438
6.5.8. User frames 438
6.5.9. Tool frames 438
6.6. Reference position setting 439
6.7. Teaching the position to robots 439
6.8. Fine adjustment of the tracking motion 439
6.8.1. Adjustment of the tracking frame error 439
6.8.2. Adjustment of the dynamic error of the robot 441
6.9. Executing production 442
6.9.1. Starting production with the operator’s panel 442
6.9.2. Starting production with peripheral equipment 442
6.9.3. In the case of dual arm robots 443
6.10. KAREL program 444
6.10.1. Workcell-related programs 444
6.10.2. Gripper-related programs 444
6.10.2.1. PKGRCHKPP.PC 445
6.10.2.2. PKGRCLOSE.PC 445
6.10.2.3. PKGRGETDPDLY.PC 446
6.10.2.4. PKGRGETID.PC 446
6.10.2.5. PKGRGETPKDLY.PC 446

6.10.2.6. PKGRGETZNCNT.PC 446
6.10.2.7. PKGRGETZNUT.PC 447
6.10.2.8. PKGRINIT.PC 447
6.10.2.9. PKGROPEN.PC 447
6.10.3. Robot-related programs 447
6.10.3.1. PKRBGETCVCNT.PC 448
6.10.3.2. PKRBGETFSCNT.PC 448
6.10.3.3. PKRBGETPRGRT.PC 448
6.10.3.4. PKRBGETPKCV.PC 449
6.10.3.5. PKRBGETDPCV.PC 449
6.10.3.6. PKRBGETPKFS.PC 450
6.10.3.7. PKRBGETDPFS.PC 450
6.10.4. Conveyor-related programs 450
6.10.5. Sensor-related programs 450
6.10.6. Conveyor station-related programs 450
6.10.6.1. PKCSCALWPOS.PC 451
6.10.6.2. PKCSGETAPPR.PC 452
6.10.6.3. PKCSGETOPPR.PC 452
6.10.6.4. PKCSGETPMUL.PC 452
6.10.6.5. PKCSGETPPCHK.PC 453
6.10.6.6. PKCSGETVR.PC 453
6.10.6.7. PKCSGETWPOS.PC 453
6.10.6.8. PKCSGETCLRQ.PC 453
6.10.7. Fixed station-related programs 454
6.10.7.1. PKFSGETAPPR.PC 454
6.10.7.2. PKFSGETIXDLY.PC 454
6.10.7.3. PKFSGETIXODO.PC 455
6.10.7.4. PKFSGETIXSIM.PC 455
6.10.7.5. PKFSGETOPPR.PC 455
6.10.7.6. PKFSGETPMUL.PC 456
6.10.7.7. PKFSGETPPCHK.PC 456
6.10.7.8. PKFSGETRDY.PC 456
6.10.7.9. PKFSGETRDYDI.PC 456
6.10.7.10. PKFSGETVR.PC 457
6.10.7.11. PKFSMONIDX.PC 457
6.10.7.12. PKFSSETRDY.PC 457
6.10.7.13. PKFSGETCLRQ.PC 458
6.10.7.14. PKFSSETIDXDO.PC 458
6.10.8. Tray-related programs 458
6.11. Standard TP programs 458
6.11.1. TP program naming conventions 459
6.11.2. Standard TP program details 460
6.11.2.1. List of all the Standard TP programs 460
6.11.2.2. PK_MAIN1.TP: Main program for group 1 461
6.11.2.3. PK_INIT1.TP 462
6.11.2.4. PK_PERCH1.TP 464
6.11.2.5. PK_PICK1.TP 465
6.11.2.6. PK_DROP1.TP 466
6.11.2.7. PK_GR_GET_UTOOLS1.TP 466
6.11.2.8. PK_CV_GET_PICK_DATA11.TP 467
6.11.2.9. PK_CV_GET_DROP_DATA11.TP 468
6.11.2.10. PK_FS_GET_PICK_DATA11.TP 469
6.11.2.11. PK_FS_GET_DROP_DATA11.TP 470
6.11.2.12. PK_CV_PICK11.TP 472
6.11.2.13. PK_CV_DROP11.TP 475
6.11.2.14. PK_CV_WAITPOS1.TP 478
6.11.2.15. PK_FS_PICK11.TP 479

6.11.2.16. PK_FS_DROP11.TP 481
6.11.2.17. PK_FS_INDEX.TP 484
6.11.2.18. PK_FS_READY.TP 485
6.11.2.19. PK_GR_GET_ENDZONE1.TP 486
6.11.2.20. PK_CYCSTOP1.TP 487
6.12. Setup of pre-grouping 487
6.12.1. Setting up a gripper and a zone 488
6.12.2. Setup of a robot operation 488
6.12.3. Fine adjustment of the tracking motion 488
6.12.3.1. Adjustment of the tracking frame error (When you use ADJ_OFS) 488
6.12.4. Standard TP programs 489
6.12.4.1. PK_PRGR_MAIN1.TP 489
6.12.4.2. PK_PRGR_INIT1.TP 491
6.12.4.3. PK_PRGR_CV_PICKDROP1.TP 491
6.12.4.4. PK_PRGR_CV_PICKVTRAY1.TP 494
6.13. Plug & Play buffer function 497
6.13.1. Robot pick/place operations in the buffer 498
6.13.2. Setting up a buffer 498
6.13.3. Buffer KAREL macros 499
6.13.3.1. PKRBGETBUFID.PC 499
6.13.3.2. PKFSGETBUFPT.PC 500
6.13.3.3. PKFSGETBUFDT.PC 500
6.13.4. Buffer TP programs 500
6.13.4.1. PK_BF_GET_DATA1.TP 500
6.13.4.2. PK_BF_PICK1.TP 501
6.13.4.3. PK_BF_DROP1.TP 503
6.13.4.4. PK_BF_OK2PICK1.TP 505
6.13.4.5. PK_BF_OK2DROP1.TP 505
6.13.4.6. PK_CV_OK2PICK1.TP 506
6.13.4.7. PK_CV_OK2DROP1.TP 507
6.13.4.8. PK_BUF_PICK1.TP 508
6.13.4.9. PK_BUF_DROP1.TP 509
6.14. iRPickTool safe retreat with skip outbound motion 509
6.15. Fixed approach height for trays with multiple layers of cells 511
6.15.1. PKCSFIXAPRZ 512
6.15.2. PKFSFIXAPRZ 513
6.16. iRPickTool robot maximum speed specification to prevent duty failures 514
6.17. Setup of multiple stations 515
6.17.1. Setting up the operation and loading programs 516
6.17.2. Setting the pick conveyor 517
6.17.3. Setting the place conveyor 519
6.17.4. Setup of a fixed station 521
7. Setting example appropriate to the usage 522
7.1. Using different robots to pick parts depending on their positions on the conveyor 522
7.2. Using different robots to pick parts depending on their characteristics 526
7.3. Stacking parts in multiple layers in a box 530
7.4. Placing parts in sequence from the end of the box 532
7.5. Placing parts in different directions in a box 533
7.6. Changing the placement position depending on the part type 534
7.7. Placing parts on different outfeed conveyors depending on the model ID 537
7.8. Picking parts from two or more infeed conveyors 539
7.9. Reporting failure to pick a part 541
7.10. Picking a tray only when all cells are filled 542
7.11. Handling parts detected before stop when restarting the system 544
8. Other useful functions 545

8.1. Exchange workcell with using recipe 545
8.1.1. Setting a recipe 545
8.1.2. Detail setting of a recipe 547
8.1.3. Recipe operation by means of a PLC 557
8.1.3.1. Settings required for recipe operation by means of a PLC 558
8.1.4. Switching recipe with TP program 563
8.1.5. KAREL program 563
8.1.5.1. PKRCCREATE.PC 563
8.1.5.2. PKRCDELETE.PC 564
8.1.5.3. PKRCSAVE.PC 564
8.1.5.4. PKRCRESTORE.PC 565
8.1.5.5. PKRCEXPORT.PC 565
8.1.5.6. PKRCIMPORT.PC 565
8.1.5.7. PKRCACTIVE.PC 566
8.1.5.8. PKRCGETPLCIO.PC 566
8.1.5.9. PKRCSETPLCIO.PC 566
8.2. Checking the production status 567
8.2.1. Checking the status of a conveyor 568
8.2.2. Checking the status of a sensor 568
8.2.3. Checking the status of a conveyor station 569
8.2.4. Checking the status of a fixed station 571
8.3. Limiting the movable range of the final axis 572
8.4. Skipping the tracking motion outside the area 574
8.5. Checking the robot torque 576
8.6. iRPickTool 4D graphics display 577
8.6.1. Displaying the 4D Workcell 578
8.6.2. Creating the 4D Workcell 578
8.6.3. Using the touch screen icons 589
8.6.4. Selective display of specific information 590
8.6.4.1. 4D pick boundaries 590
8.6.4.2. 4D pick frames 590
8.6.5. Deleting the 4D Workcell 591
8.6.6. Changing the workcell scene using recipes 591
8.6.7. Using user-defined object shapes 591
8.6.8. Restrictions 592
8.7. Servo conveyor tracking 592
8.7.1. Setup 592
8.7.1.1. Setting up an extended axis for an index conveyor 592
8.7.1.2. Setting up an index servo conveyor 594
8.7.1.3. How to create a TP program for index servo conveyor 597
8.7.1.4. Setting up tracking schedules 599
8.7.1.5. Example of creating a main program 599
8.7.2. Servo conveyor function 600
8.7.2.1. Servo conveyor error adjustment parameters 600
8.7.2.2. Servo conveyor stop-wait function 600
8.7.2.3. Pick or Drop Multiple 601
8.7.2.4. KAREL programs for servo conveyor Line Tracking 603
8.7.3. Servo conveyor tracking restrictions 611
8.8. Multi-view 611
8.8.1. Overview 611
8.8.2. Restrictions 612
8.8.3. Preparation for camera 612
8.8.4. Setup 613
8.8.4.1. Setting up the vision system 613
8.8.4.2. Setting up a sensor 616
8.8.4.3. Reference position setting and robot position teaching 618

8.8.5. Production Status 619
8.9. 3D Vision Sensor 620
8.9.1. Overview 620
8.9.2. Restrictions 621
8.9.3. Mounting position setup 621
8.9.4. Vision process (3D) 629
8.9.5. Vision process (2D) 634
8.10. Clearance check function 637
8.10.1. Overview 637
8.10.2. Restrictions 637
8.10.3. Restrictions 638
8.10.4. Setup 639
8.10.4.1. Setup of a workcell and robot 639
8.10.4.2. Setting up a tray 639
8.10.4.3. Setting up a conveyor 640
8.10.4.4. Setting up a sensor 640
8.10.4.5. Clearance guide 642
8.10.4.6. Setting up a conveyor station 648
8.10.4.7. Setup of a fixed station 649
8.11. Diagnostic function 649
8.11.1. Collecting the iRPickTool diagnostics data 649
8.11.2. Collecting debug data 650
8.12. Tray redetection 653
8.12.1. Possible cases 654
8.12.2. Restrictions 655
8.12.3. Startup procedure 656
8.13. Tracking using robot-mounted camera 658
8.13.1. Restrictions 658
8.13.2. Teaching method 658
8.13.2.1. Definitions of registers used 659
8.13.2.2. Definitions of coordinate systems used 660
8.13.2.3. Creating a robot program 660
8.13.2.4. Setting up an infeed conveyor sensor 661
8.13.2.5. Setting an infeed conveyor sensor position 662
8.13.2.6. Setting up a robot-mounted camera 665
8.13.2.7. Creating a vision process 668
8.13.2.8. Setting up a reference position for the infeed conveyor, and teaching a robot position 670
8.13.3. KAREL program 674
8.13.3.1. PKSETNOMFRM.PC 674
8.13.3.2. PKCALOFSPOS.PC 674
8.13.3.3. PKCALOFSPOS2.PC 675
8.14. Filtering for signals 675
8.14.1. Specified By 676
8.14.2. Debounce 676
8.14.3. Duration 676
8.14.4. Polarity Inversion 677
8.15. Using a conveyor as a buffer 678
8.15.1. Functional overview 678
8.15.2. Startup procedure 682
8.15.2.1. Sensor installation and connection 683
8.15.2.2. Setting up applications 683
8.15.2.3. Checking the production status 690
8.15.3. KAREL program 690
8.15.3.1. PKCVCLRBUF.PC 691
8.15.3.2. PKCSGETBUF.PC 691
8.15.3.3. PKCSACKBUF.PC 693

8.15.3.4. PKCSGETBFDIS.PC 693
8.15.4. Sample programs for the infeed-to-buffer-to-outfeed-conveyor configuration 694
8.15.4.1. Tracking program for placing workpieces on a forward-moving buffer 695
8.15.4.2. Tracking program for placing workpieces on a backward-moving buffer 695
8.15.4.3. Tracking program for picking workpieces from a forward-moving buffer 696
8.15.4.4. Tracking program for picking workpieces from a backward-moving buffer 696
9. Troubleshooting 697
9.1. When the robot does not pick up parts 697
9.2. When parts are missed frequently 697
9.3. When the same part is picked up twice 698
9.4. Inaccuracy when a part rotates 699
9.5. Inaccuracy when a part is near the conveyor edges 699
9.6. When robot motion is not smooth 699
9.7. The stop conveyor does not work 699
9.8. Slow vision processing due to delayed image snap timing 699
9.9. The overrun count continues to increase 699
9.10. When settings cannot be changed during production 700
9.11. CVIS-151 (No more vision offsets) alarm displayed during tracking motion with multi-view ... 700
Appendix 703
A. Setup data file 705
B. Setting a teaching PC 706
B.1. Operating system and browser 706
B.2. Communication cable 706
B.3. Connection of the communication cable 706
B.4. Setting an IP address for the PC 707
C. αiA4000S Pulsecoder 710
C.1. Requirements 710
C.2. Figures 711
C.3. How to connect 713
D. KAREL program list 715
E. Sensor customization 717
E.1. Overview 717
E.2. Procedure to customize a sensor task 717
E.3. Essential points of customization 722
E.4. Routines 726
E.4.1. put_part_cst 726
E.4.2. run_vis_cst 726
E.4.3. update_nfnd 726
E.4.4. put_queue 727
E.4.5. get_idx_ecnt 728
E.4.6. is_srvo_conv 728
E.4.7. is_cond_dist 728
E.4.8. get_enc_id 729
E.4.9. get_conv_id 729
E.4.10. get_cstn_id 730
E.4.11. get_work_id 730
E.4.12. get_sens_pos 730
E.4.13. get_vp_name 731
E.4.14. get_tray_id 731
E.4.15. get_tray_pos 731
E.4.16. update_wid 732
E.5. Built-in functions 732
E.5.1. LINE_COUNT 732
E.5.2. V_RUN_FIND 733

E.5.3. VT_GET_FOUND 733
E.5.4. P_CSGETID 734
E.5.5. P_CVGETID 734
F. Vision I/F add-on 736
F.1. Overview 736
F.2. Terms 736
F.3. Restrictions 737
F.3.1. Restrictions on the robot system 737
F.3.2. Restrictions on the external machine vision systems 738
F.4. About the support scope 738
F.5. Startup procedure 739
F.5.1. External machine vision system connection and settings 740
F.5.1.1. External machine vision system connection 740
F.5.1.2. External machine vision system settings 741
F.5.2. Communication settings 741
F.5.2.1. IP address setting 741
F.5.2.2. Setting up a client tag 741
F.5.3. Setting up iRPickTool 742
F.5.4. Creating a robot program 742
F.5.4.1. Main program 746
F.5.4.2. Action program 747
F.5.4.3. Initialization program 748
F.5.4.4. Detection program 749
F.5.4.5. Latch program 750
F.5.4.6. Coordinate conversion parameter calculation program 751
F.5.4.7. Reference position setting program 752
F.5.5. Setting up an infeed conveyor sensor 753
F.5.6. Position register operation settings 754
F.5.7. Calculating the coordinate conversion parameter 754
F.5.8. Setting up a reference position for the infeed conveyor, and teaching a robot position 755
F.6. KAREL program 755
F.6.1. Programs related to communication 756
F.6.1.1. PKMVCONNECT.PC 756
F.6.1.2. PKMVDISCO.PC 756
F.6.2. Programs related to initialization 756
F.6.2.1. PKMVINIACT.PC 756
F.6.3. Programs related to input message transmission 757
F.6.3.1. PKMVSETSNDPM.PC 757
F.6.3.2. PKMVDELSNDPM.PC 757
F.6.3.3. PKMVSENDMSG1~4.PC 758
F.6.4. Programs related to output message reception 758
F.6.4.1. PKMVSETRCVPM.PC 758
F.6.4.2. PKMVSENDMSG1~4.PC 758
F.6.5. Programs related to output message analysis 759
F.6.5.1. PKMVSETPRSPM.PC 759
F.6.5.2. PKMVDELPRSPM.PC 759
F.6.5.3. PKMVSENDMSG1~4.PC 760
F.6.6. Programs related to coordinate conversion parameter calculation 760
F.6.6.1. PKMVSCLPUTP1.PC 760
F.6.6.2. PKMVSCLPUTP2.PC 761
F.6.6.3. PKMVSCLPUTP3.PC 761
F.6.6.4. PKMVSCLSETP1.PC 761
F.6.6.5. PKMVSCLSETP2PC 761
F.6.6.6. PKMVSCLCALPM.PC 762
F.6.7. Programs related to detection position conversion 762
F.6.7.1. PKMVCNVPX2MM.PC 762

F.6.7.2. PKMVCALOFS2D.PC 762
F.6.7.3. PKMVSETVR.PC 763
F.6.8. Other programs 763
F.6.8.1. PKMVWAITTASK.PC 763
F.7. Setting example appropriate to the usage 764
F.7.1. Detecting multiple parts at one snap 764
F.7.2. Connecting multiple robots 767
F.7.2.1. Program for setting a trigger 768
F.8. Troubleshooting 768
F.8.1. The external machine vision cannot be connected 768
F.8.2. Snapping or detection is not executed in the external machine vision system 768
F.8.3. Timeout occurs in PKMVRCVMSG. 768
F.8.4. PKMVPRSMSG fails 769
F.8.5. Latch fails 769
F.8.6. The grip position is incorrect 769
Revision record 771

I Main part

1. Overview

iRPickTool is robot software that has been developed for applications where the robot picks/places parts on a conveyor while tracking the motion of the conveyor.

It enables the robot to intelligently pick/place parts on a moving conveyor.

This is achieved by combining motion synchronization with parts on a moving conveyor (linear or circular), detecting those parts with a camera or photo eye, and, if multiple robots are utilized, the load balancing of the incoming or outgoing part flow.

It also supports the following kinds of applications.

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

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

1. iRPickTool Basic
2. iRPickTool
3. iRPickTool Basic + iRVision options (Visual tracking)
4. iRPickTool + iRVision options (Visual tracking)

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

Visual tracking is an application that detects parts usually using a camera.

An application that detects parts using a sensor such as a photo eye instead of 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 "I.1.1. Configuration of software options(P.4)".

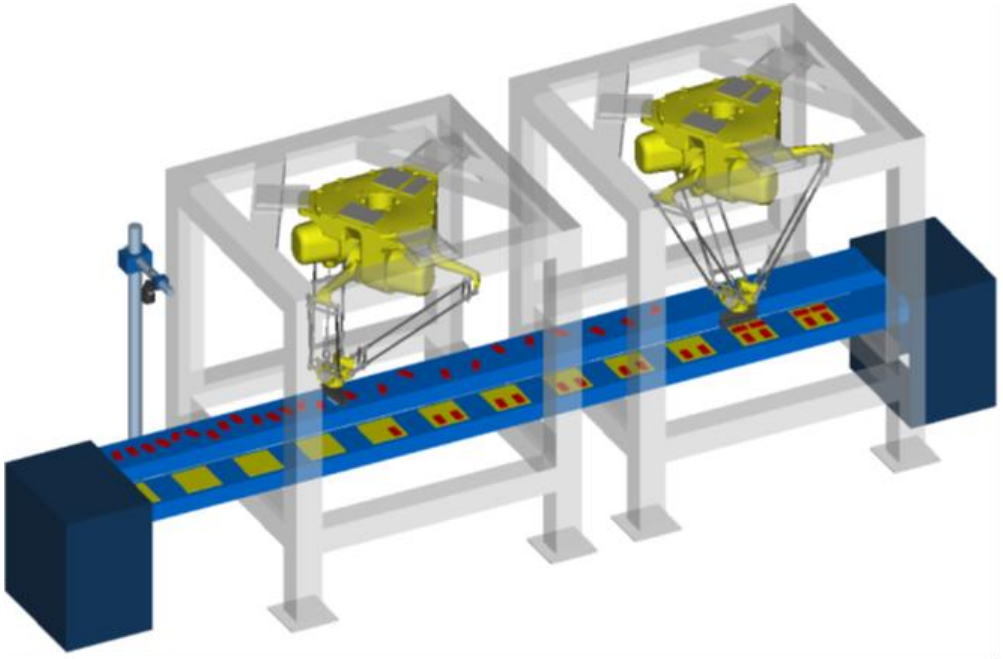


Fig. 1-1 Example use of visual tracking (The camera on the left side is used.)

1.1. Configuration of software options

iRPickTool has the following software configuration.

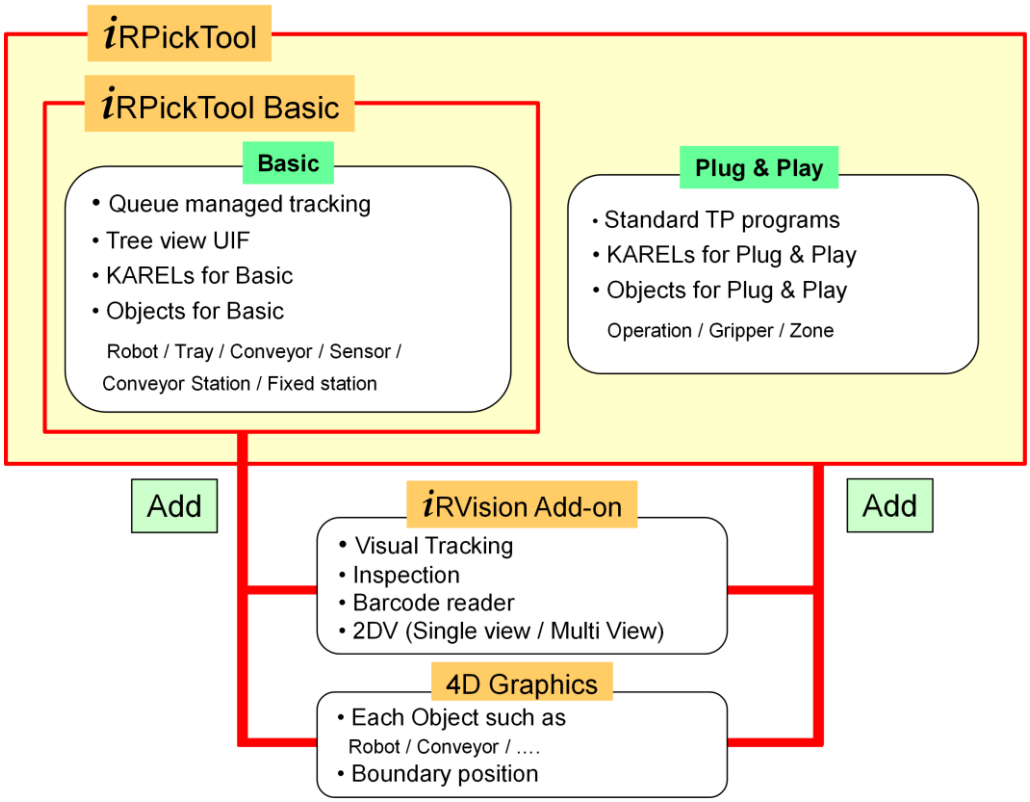


Fig. 1.1-1 iRPickTool software configuration diagram

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 among 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 that iRPickTool has the Plug & Play functions.

⚠CAUTION

- When iRPickTool (J945) is installed, iRPickTool Basic (J944) cannot be used. For example, it is not possible to use a backup of iRPickTool Basic (J944) on a robot controller in which iRPickTool (J945) is installed.
  When using the ROS Interface Packets Over Ethernet (RIPE), it is not possible to connect and use robot controllers in which iRPickTool Basic (J944) or iRPickTool (J945) is installed.
- To use the Ethernet Encoder with multiple robots, the Ethernet Encoder function (R762) must be installed on all robot controllers.

1.2. Configurations

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

1.2.1. Queue managed tracking

A queue managed tracking system consists of the following devices.

Devices required for the system to run

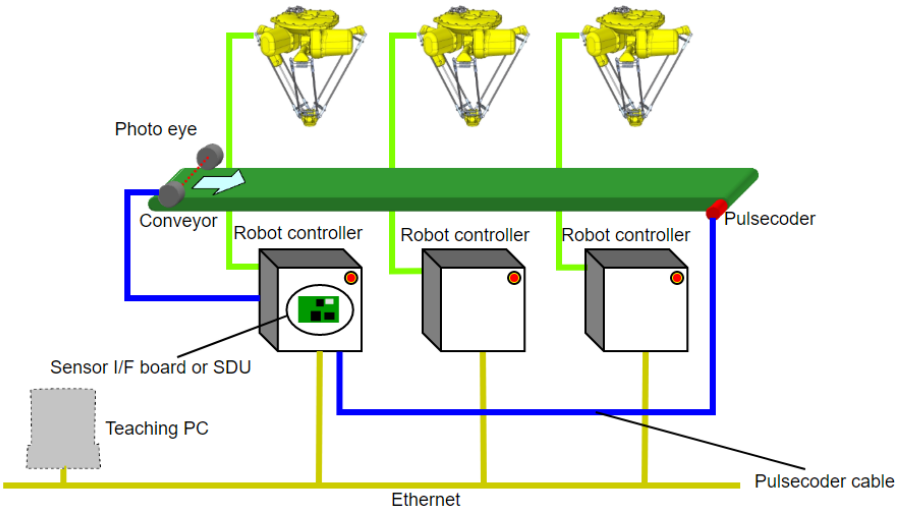
- Robot controller
- Robot
- Conveyor
- Photo eye
- Pulsecoder
- Sensor I/F board (installed in the robot controller. An SDU is required instead to use a pulse multiplexer.)
- Pulsecoder cable
- Pulse multiplexer (A cable to connect the pulse multiplexer and the Pulsecoder and a pulse multiplexer cable are also required at the same time in place of the Pulsecoder cable in order to use a pulse multiplexer.)
- Ethernet communication cable
- Ethernet hub

Devices only required for system setup

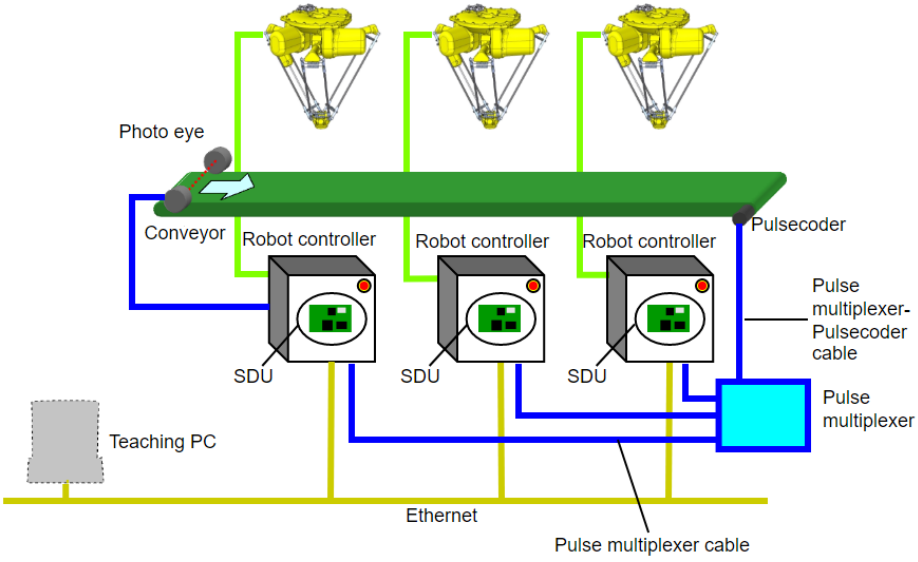
- A teaching PC and Ethernet communication cable
- Teach pendant (or tablet TP)

Sample configurations

When using the Ethernet Encoder



When using the pulse multiplexer



1.2.2. Visual tracking

A visual tracking system consists of the following devices.

Devices required for the system to run

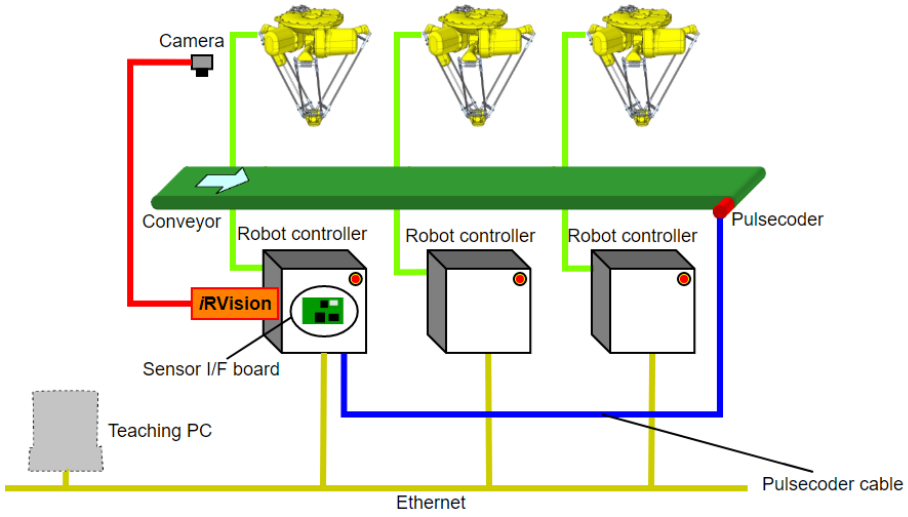
- Robot controller
- Robot
- Conveyor
- Photo eye (must be used as the trigger sensor for snapping an image using a camera)
- Camera
- Lens
- Camera cable
- Lighting
- Pulsecoder
- Sensor I/F board (installed in the robot controller.)
- Pulsecoder cable
- Pulse multiplexer (A cable to connect the pulse multiplexer and the Pulsecoder and a pulse multiplexer cable are also required at the same time in place of the Pulsecoder cable in order to use a pulse multiplexer. An SDU is required for each robot controller.)
- Ethernet communication cable
- Ethernet hub

Devices only required for system setup

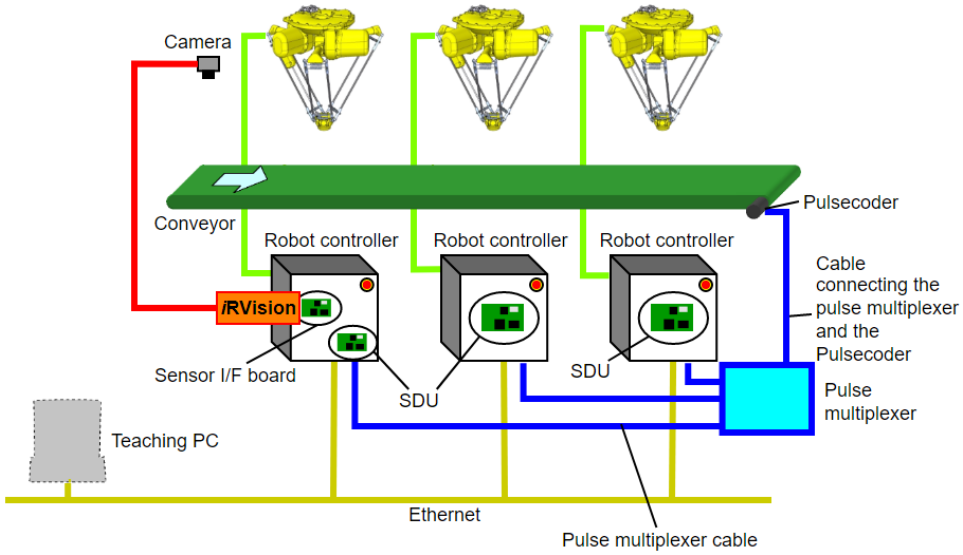
- A teaching PC and Ethernet communication cable
- Calibration grid (camera calibration plate)
- Teach pendant (or tablet TP)

Sample configurations

When using the Ethernet Encoder



When using the pulse multiplexer



1.3. Terms

This section describes terms used in this manual.

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

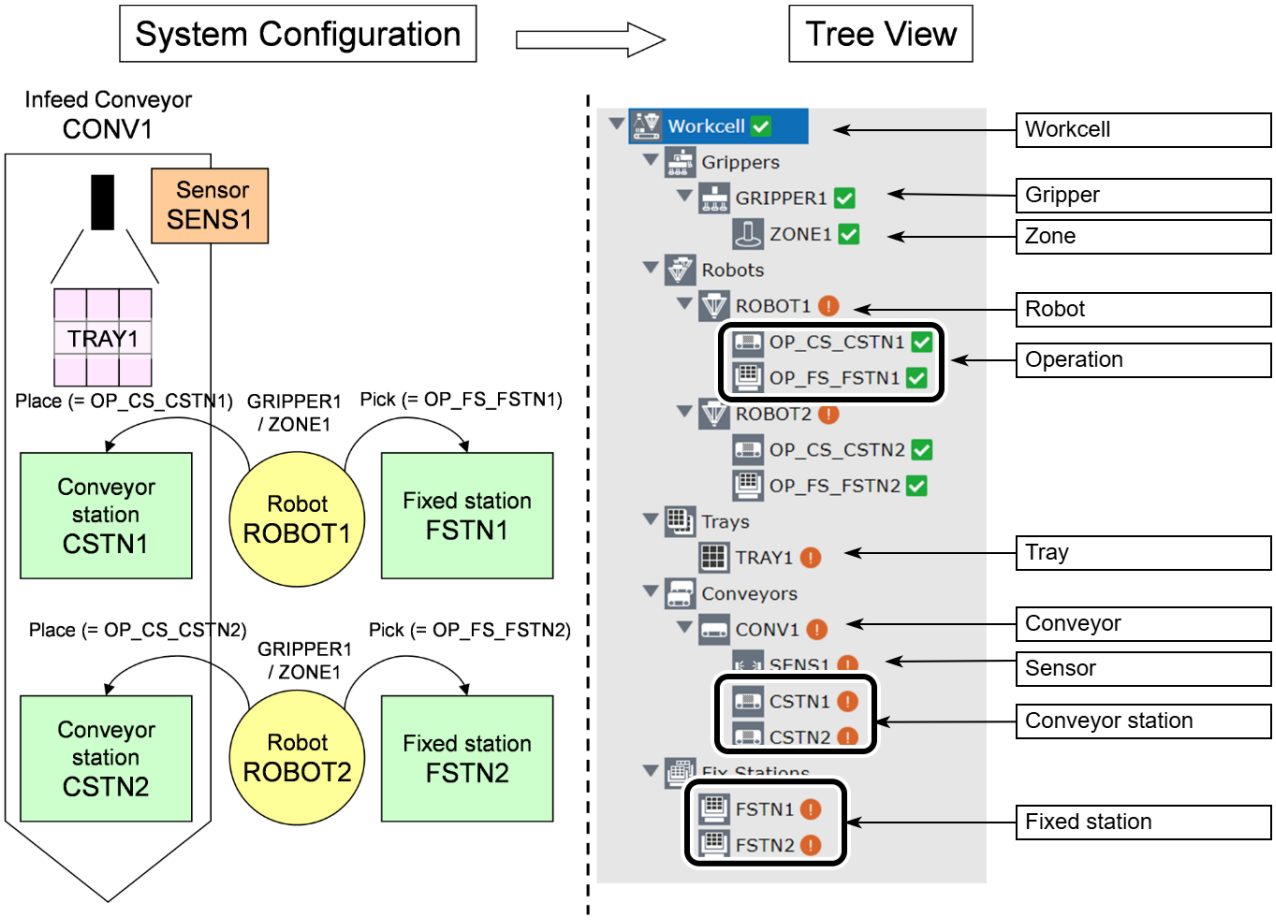


Fig. 1.3-1 System configuration and iRPickTool tree view display

With iRPickTool, set items using the tree view shown in Fig. 1.3-1. Each component used in this tree view is called an "object". Sequentially set objects from the top in the tree view.

[Workcell]

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

[Gripper]

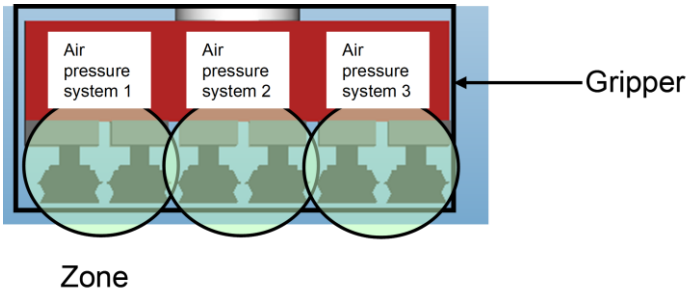
A gripper is an object indicating a gripper installed on the faceplate of a robot. It is contained only in Plug & Play and is referenced from a robot as described later.

When robots set in the same workcell 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.

[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.

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.



[Robot]

Robot refers to the devices performing tasks. It is necessary to create as many robot objects as the number of physical robots.

[Operation]

An operation is an object that belongs to the robot. It is included only in Plug & Play. It is an object that represents the pick/place operations of the robot at a conveyor station and fixed station, which are described later. There is a one-to-one correspondence between a conveyor station and an operation and 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” being carried on an outfeed conveyor. Generally, multiple parts are placed in a box. A tray is the tree view object that represents this box.

[Cell]	Each place where a part is placed (or picked up) is called a cell. For example, a box in which four parts can be laid out is defined by 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 them up 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 of one tray.

An example of a tray having four cells is given below to give you a better understanding.

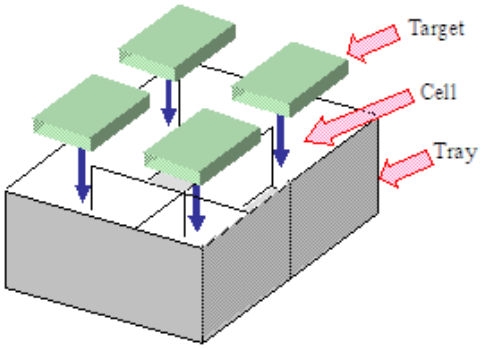


Fig. 1.3-2 Example: A tray that has four cells

In the example in Fig. 1.3-2, four parts can be placed in the box (or can be picked from the box).

The box itself is called a “tray” and each empty location in the tray in which a part can be placed or picked up is called a “cell”.

An occupied cell is called a “target”. This tray has four cells and can be represented as the figure below.

Fig. 1.3-3 is a top view of the tray in Fig. 1.3-2. In this example, a tray frame is defined at the lower-left corner of the box.

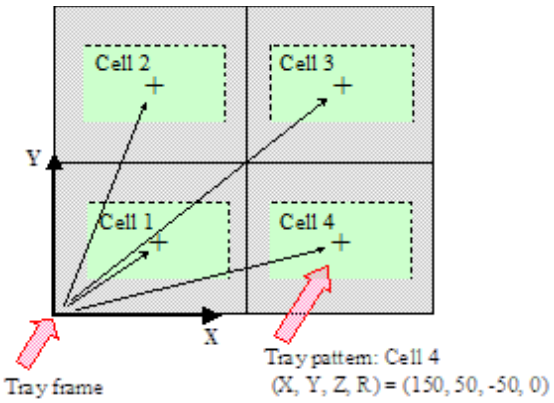


Fig. 1.3-3 An example of overlaying a frame on a tray that has four cells

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

The tray frame can be set anywhere in the tray. To teach a robot the position of the tray frame, however, set the position where the touch-up operation can be performed easily, such as the corner of a box, because the origin of the tray frame and a point along the X-axis must be touched up to set the TCP of each robot.

[Conveyor]

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

[Sensor]

A sensor is an object that detects parts moving along a conveyor and inputs the detection result to the most upstream conveyor station configured 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, see “E. Sensor customization(P.717)” or contact your local FANUC representative.

[Conveyor Station]

A conveyor station is an object indicating an area on a conveyor in which a robot performs an operation on a moving part. The robot picks up a part for a conveyor station and places it in another conveyor station or fixed station.

An example is given below to give you a better understanding of conveyors, sensors, and conveyor stations.

Fig. 1.3-4 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 part on the conveyor. One photo eye is installed at the upstream end of conveyor 2. This photo eye is used to detect each box on the conveyor. This system picks up parts from conveyor 1 and places them in boxes as they flow along conveyor 2.

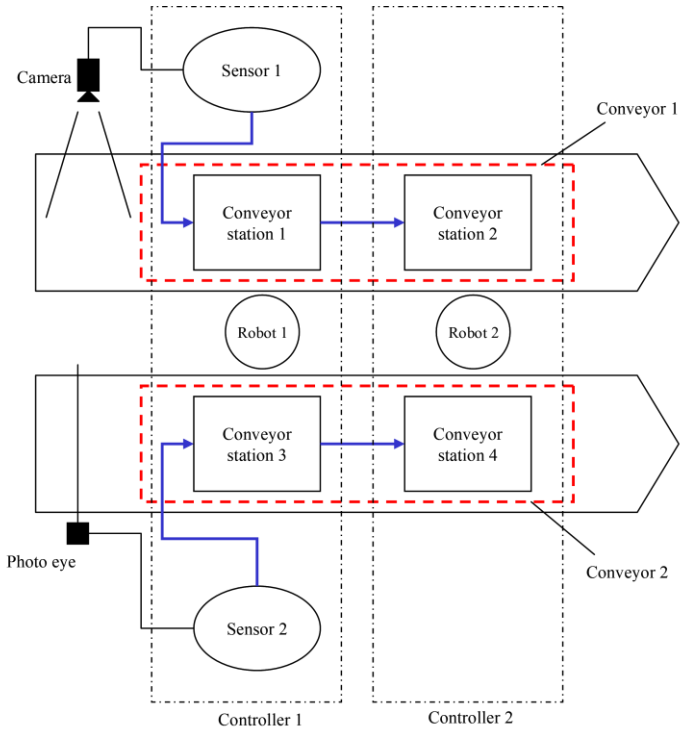


Fig. 1.3-4 Conveyor station system example 1

This system has two conveyors, conveyors 1 and 2, and four conveyor stations, conveyor stations 1 to 4.

Conveyor stations 1 and 2 are defined on conveyor 1 and conveyor stations 3 and 4 are defined on conveyor 2.

Robot 1 picks up parts from conveyor station 1 on conveyor 1 and places them in conveyor station 3 on conveyor 2.
Robot 2 picks up parts from conveyor station 2 on conveyor 1 and places them in conveyor station 4 on conveyor 2.

Sensor 1 detects parts on conveyor 1 using a camera. The detected part information is input to conveyor 1. Sensor 2 detects parts on conveyor 2 using a photo eye. The detected part information is input to conveyor 3.

The information of the part that sensor 1 inputs to conveyor station 1 flows from conveyor station 1 to conveyor station 2 as the part flows.

Similarly, the information of the part that sensor 2 inputs to conveyor station 3 flows from conveyor station 3 to conveyor station 4.

Each robot receives the information of the flowing parts from the conveyor station, and picks up a part or places it into a box.

If a part does not flow to a conveyor station, its information is not passed to the robot.

If a part passes through a conveyor station, its information is sent to the next conveyor station.

The information of the flowing parts is given to each robot based on information including the position of each part and predetermined allocation ratio set for each conveyor station.

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

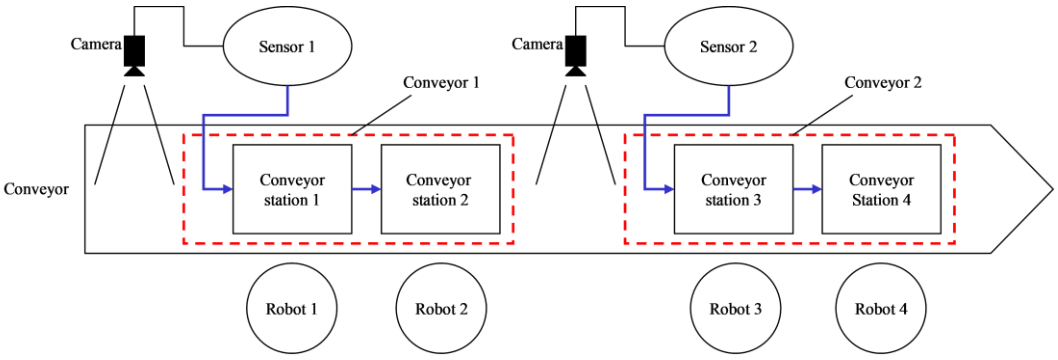


Fig. 1.3-5 Conveyor station system example 2

Line track schedule

This is data such as a tracking 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 place that does not move. 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. Because this is a nonmoving zone, the stations are not connected but are independent fixed stations, unlike conveyor stations.

In the following figure, the outfeed conveyor in the configuration in Fig. 1.3-4 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.

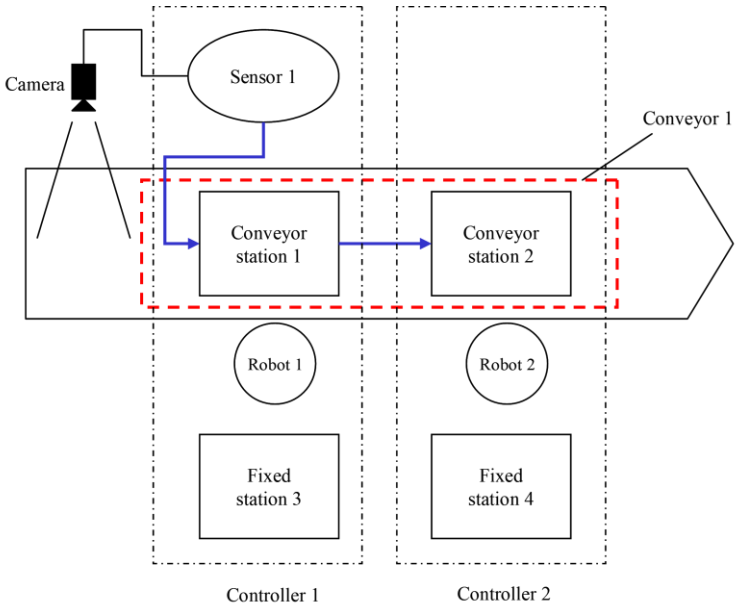


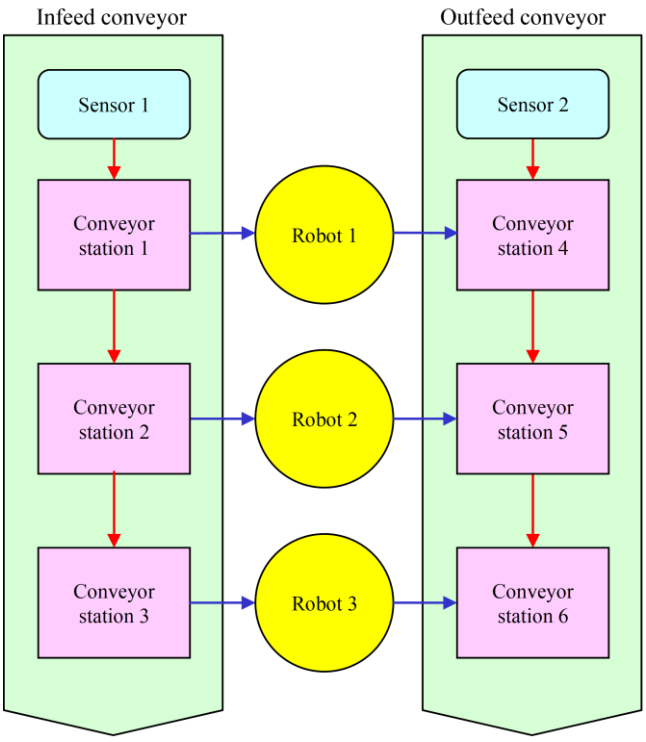
Fig. 1.3-6 Fixed station system example

1.4. Sample system configurations

This section introduces some sample system configurations to help you understand iRPickTool system configurations more specifically.

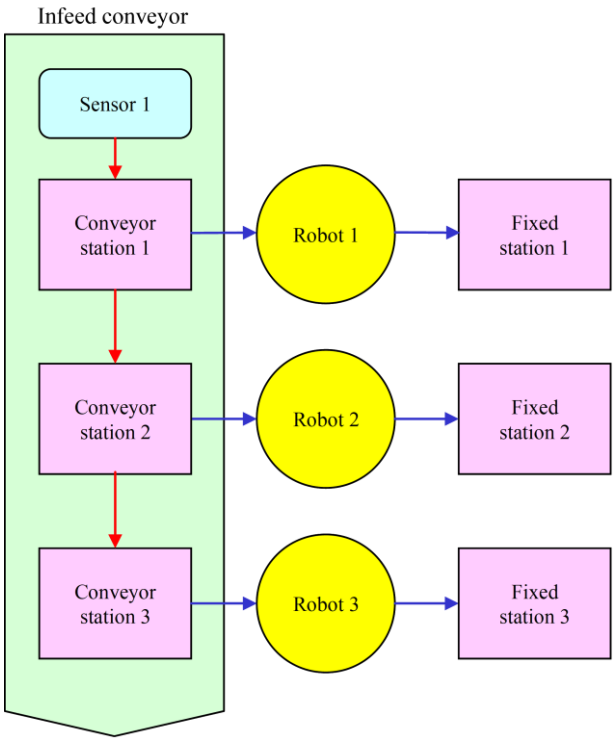
Sample configuration 1

This is the most basic configuration. Each robot picks up parts being carried on the infeed conveyor and places them on the outfeed conveyor.



Sample configuration 2

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

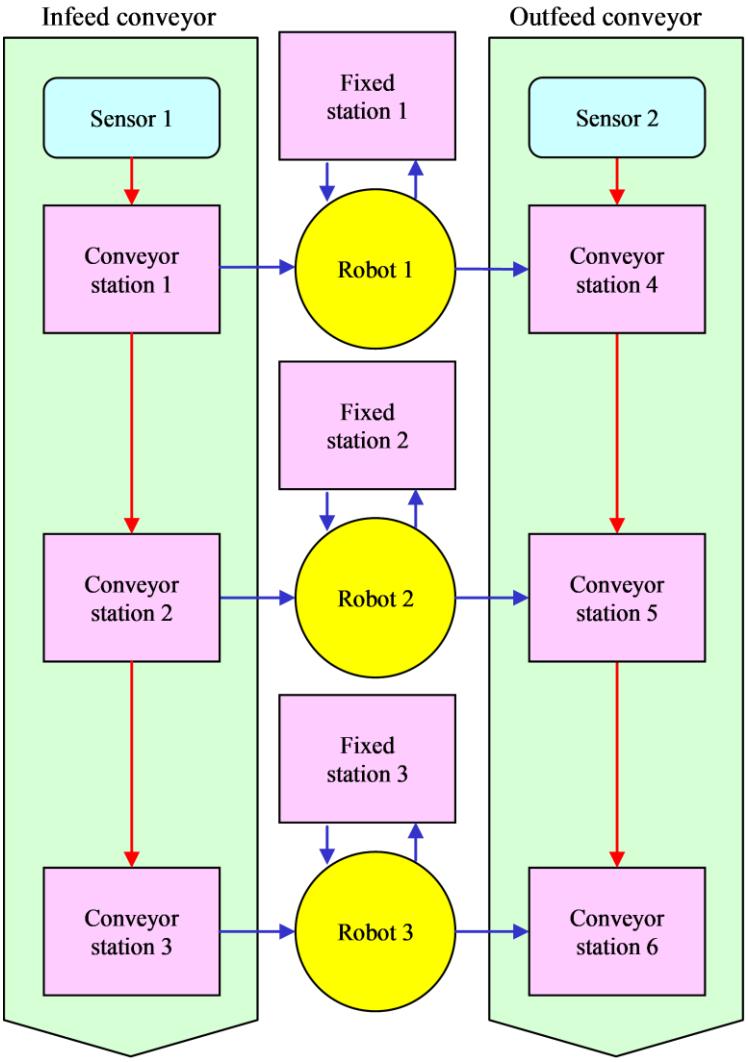


Sample configuration 3

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

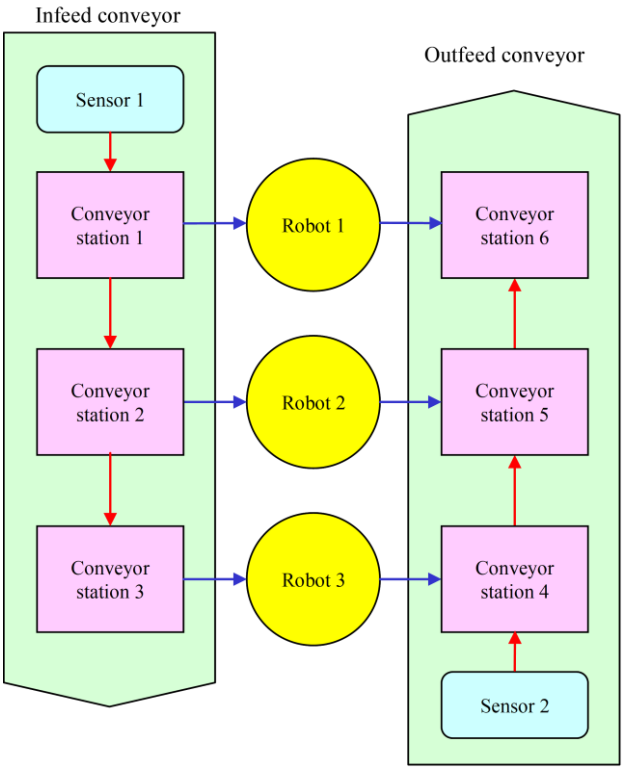
If parts are irregularly supplied from the infeed conveyor, then when too many are supplied, the parts are temporarily placed and stored at the fixed station. These fixed stations function as “buffers”.

When the supply of parts runs short, the robot picks up parts from the fixed station and places them in the conveyor station of the outfeed conveyor.



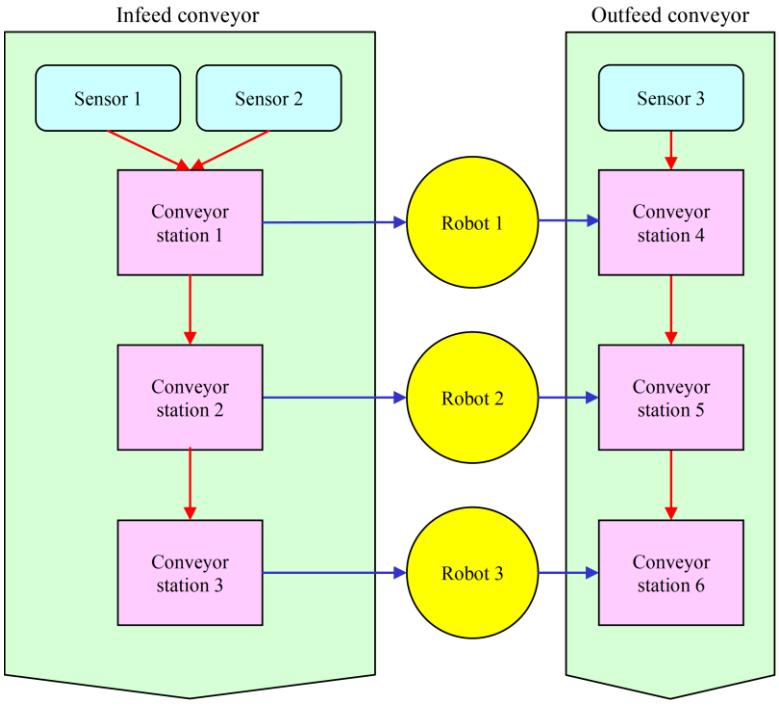
Sample configuration 4

This sample is basically the same as sample configuration 1. The flow direction is different between the infeed and outfeed conveyors.



Sample configuration 5

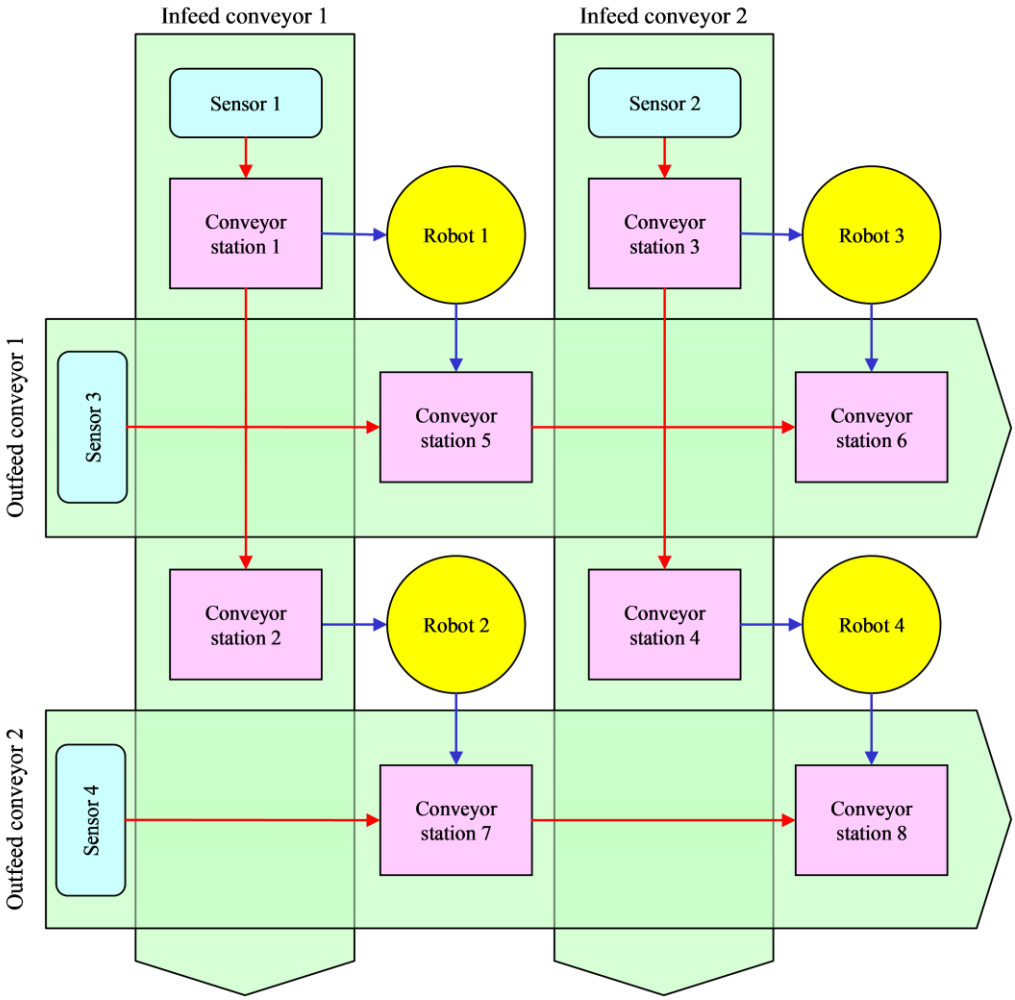
This sample is also basically the same as sample configuration 1. In this sample, conveyor 1 is wide and cannot be covered with one camera. In this case, install two cameras on the infeed conveyor, one covering the right half and the other covering the left half, and configure the system so that two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



Sample configuration 6

The configuration of this system is a little more complicated, in which two infeed and outfeed conveyors are installed and arranged so they intersect.

This system is seemingly complicated, but when you focus on the operation of each robot, you can see that the 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.



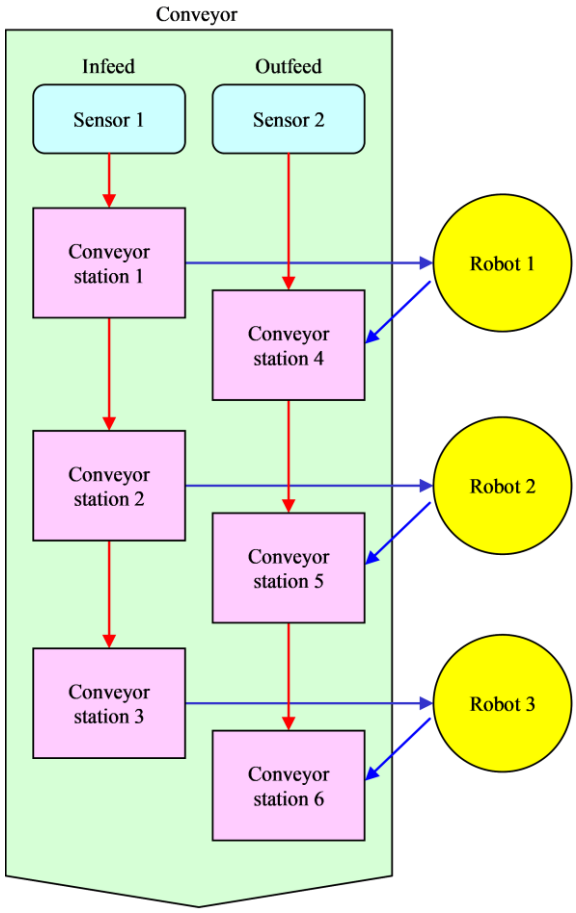
Sample configuration 7

In this system, the infeed and outfeed conveyors are physically the same conveyor.

This configuration applies to the case where parts being carried in a disorderly fashion 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 parts on the conveyor at equal spaces.

Although only one conveyor is used, it is necessary to create two conveyor objects on the infeed and outfeed sides.



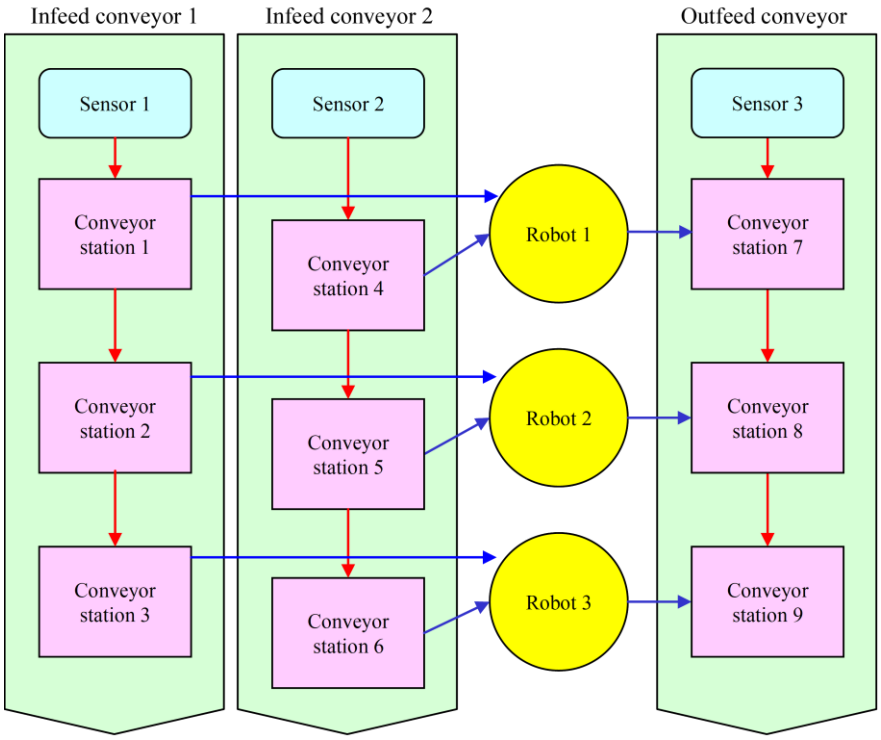
Sample configuration 8

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 being carried on the outfeed conveyor.

The robot program for this system is complex.

An example of the actual program is introduced in “I.7. Setting example appropriate to the usage(P.522)”.



1.5. Study for application

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

System layout

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. Design a layout of robots, conveyors, fixed stations, trays, and other devices to minimize the distance the robots move.

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, etc.

The maximum robot speed depends on the type of robot. The distance moved by a robot and the hand 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, ensure that the weight of the gripper plus parts does not exceed the transportable weight of the robot. Generally, the gripper is large in this case, so 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 being carried 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 parts being carried need to be constant.

Detection using a camera

A camera is installed in the upstream area of the conveyor, an image of 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. In one method, when the conveyor moves a certain distance, the system snaps an image using the camera to continuously monitor the conveyor. In the other method, the system only snaps an image to detect a part when a part passes in front of a photo eye.

⚠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.
- To allow multiple robots to retrieve workpieces detected by a single camera, the camera is typically connected to the most upstream robot controller.
- Only the robot controller connected to the camera requires the iRPickTool / iRVision Add-on (J946) and iRPickTool / Vision I/F Add-on (S521).
- Images of picked workpieces are snapped using a fixed camera, and the snapped images may then undergo tool offset processing. If both the fixed camera and the visual tracking camera are connected to the same robot controller and are both iRVision models, whichever process starts first, tool offset or visual tracking, takes priority. Depending on the processing timing, the system may wait for either tool offset image snapping and detection or visual tool tracking image snapping and detection.

Distribution of the pulse count

When multiple robots are used, how the pulse count of the Pulsecoder will be distributed to each robot is determined. The following two methods are available.

Distribution using a pulse multiplexer

The pulse count is directly input to each robot controller via the pulse distributor. Since the pulse count recognized by each robot controller is identical, high-precision tracking becomes possible.

However, the number of devices required to run the system will increase. An SDU is required for each robot controller.

In addition, a cable that connects each SDU and the pulse multiplexer and a cable that connects the pulse multiplexer and Pulsecoder are required.

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. This method eliminates the need for a pulse multiplexer, the cables connecting it to each SDU, and the SDUs for each robot controller, thereby reducing hardware costs.

However, some variations may occur in the way each robot controller recognizes the pulse count due to discrepancies in communication. Compared to distribution using a pulse multiplexer, variations 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 in 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 ring leader controller of the robot ring to reduce the communication traffic.

⚠CAUTION

2. When using the Ethernet Encoder, connect a camera or sensor to the robot controller connected to the Pulsecoder to obtain an accurate pulse count at the moment that parts are detected. Combined with CAUTION 1, connect the Pulsecoder and camera or sensor to the ring leader controller.
3. Using a pulse multiplexer is recommended in a system with two or more cameras. As mentioned in the previous caution, when using an Ethernet encoder, you must connect a camera to a robot controller that does not have a Pulsecoder connected.
   As a result, the variations in the handling position may be greater because a robot controller that does not have a Pulsecoder connected cannot obtain an accurate pulse count at the moment that parts are detected.

Conveyor speed

The 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: Image sensor size
L: Distance between the camera and the part
f: Focal distance

The following figure shows the positional relationship.

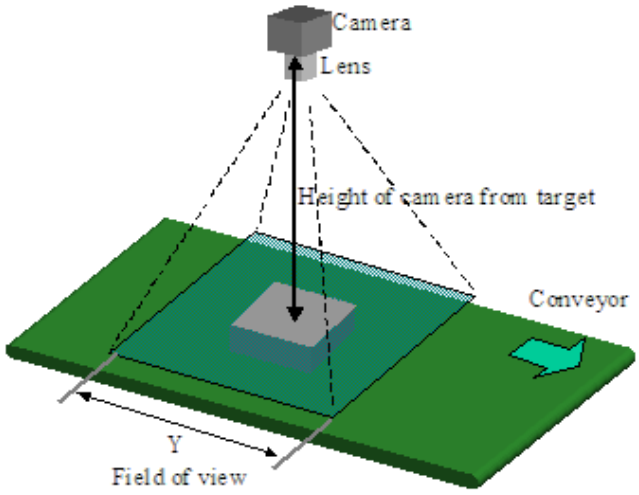


Fig. 1.5-1 Positional relationship that determines a camera's field of view

The image sensor size is the size of the charge-coupled device of the camera. It can be determined using the following expression:

Image sensor size = Cell size × Image size

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

Fig. 1.5-2 Types of camera and resolution

Camera	Cell size	Image size (in pixels)	Image sensor size
Digital camera (grayscale) SC500SF3	3.45 μm	2432 × 2048	8.390 mm × 7.065 mm
Digital camera (color) SC500SF3C	6.9 μm	1216 × 1024	8.390 mm × 7.065 mm

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

Horizontal direction: 8.390 (mm) −((1000 (mm) − 8 (mm)) ÷ 8 (mm)) = 1040.36 (mm)

Vertical direction: 7.065 (mm) −((1000 (mm) − 8 (mm)) ÷ 8 (mm)) = 876.06 (mm)

NOTE

It is better for the field of view to be 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, figure out an 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 amount of 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 gets higher, the exposure time has to be shorter. On the other hand, as the conveyor speed gets lower, the exposure time can be longer.

However, as the exposure time gets longer, the time required for detection by the vision system also gets longer. So, even when the conveyor moves slowly, a relatively short exposure time should be used.

Use the following formula 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 exposure time needed to make the image displacement 0.1 mm or less when a part is detected by the vision system is computed by the following expression:

0.1 (mm)÷ 100 (mm/s) = 1 (ms)

Accordingly, set an exposure time of 1 ms or less.

As the exposure time gets 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 per unit distance of the conveyor 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 gets lower, the image processing needs to be performed less frequently.

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

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

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

Handling precision

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

The handling precision can be improved by taking the following measures:

- Set the TCP for touch-up accurately (UTOOL should be precise).
- Decrease the conveyor speed to detect many parts at a time.
- Prepare sufficiently 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. The operation of changing the robot to which to pass the part information 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 are carried, however, the robot in the upstream area handles more parts and only few parts are carried 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 is to handle. For example, specification can be made so that robot 1 handles round parts and robot 2 handles triangular 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 along 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 "I.7. Setting example appropriate to the usage(P.522)".

Robot controller that 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 a 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 considerations are 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 is put 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.

1.6. Before use

1.6.1. Restrictions

When the tracking program is executed with the override set to less than 100%, the robot requires additional time to reach the target position.

However, its ability to track the conveyor is not affected by changes to the override value. If low speed operation checks are required, the conveyor speed must also be reduced accordingly.

If a tracking program is currently being executed at any override speed, and the conveyor begins to move, the robot motion will immediately stop; and the error "LNTK-041 Encoder is moved in T1 mode" will be displayed.

1.6.2. Restrictions

As of 7DH1/28 V10.10P/28), the following restrictions are imposed on queue-managed tracking and visual tracking.

- Cannot be used with software other than Handling Tool or Spot Tool+.
- Cannot be used with the spot welding function, dispense function, press machine function, and applications such as Pallet Tool.
- Does not support rail tracking.
- Does not support hot start mode. Disable the hot start.
- Does not support analog cameras.
- Cannot use the “Virtual” encoder type.
- Cannot use the Distance Before function. (Can use the Time Before function.)
- There are restrictions concerning the sensors that can be operated simultaneously. For details, see “I.5.5.7. Restriction concerning sensors that can be operated simultaneously(P.317)”.
- When using queue managed tracking, the encoder count command of the tracking command is not available.
- When using the ROS Interface Packets Over Ethernet (RIPE), the software series/versions of all the connected controllers need to match.

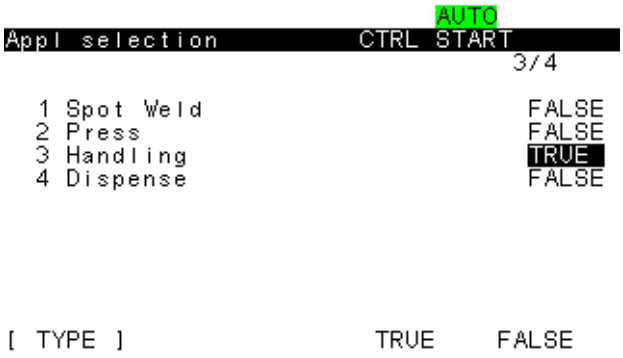
  Additionally, it is not possible to connect and use robot controllers in which iRPickTool Basic (J944) or iRPickTool (J945) is installed.
- The following functions cannot be used together. This is due to the limitations of the line tracking function.
  - Palletizing
  - Coordinated Motion
  - Space Check
  - Remote TCP

- Singularity Avoidance (Unavailable for tracking motion. Only available for normal motion.)
- Finishing Function Package
- Servo Gun Change function
- Robot Link Function
- Basic/Intelligent Interference Check
- Arc Sensor (TAST)
- Arc Voltage Control (AVC)
- Root Pass Memorization/Multi Pass Function (RPM)
- Touch Sensor
- Restart Position check function
- Weaving Function
- Continuous Turn function (This can be used as well if the motion group is different.)
- Softfloat Function
- Touch skip function
- MROT
- Distance Before function

- Different Pulsecoders cannot be used in multiple robot controllers that belong to the same conveyor object. Use the same Pulsecoder using the Ethernet Encoder or pulse multiplexer. For details on the object, see "I.5. Basic functions reference(P.258)".

1.6.2.1. Restrictions of SpotTool+

SpotTool+ (H590) supports iRPickTool Basic (J944) but does not support iRPickTool (J945). When iRPickTool Basic (J944) is ordered, the default application selected for Controlled Start is Handling. This selection cannot be changed.



For details, refer to the "Multiple applications" section in the appendix of "Spot Welding Function OPERATOR'S MANUAL"(B-84464EN-4).

1.6.3. Recommendations

The following settings are recommended in iRPickTool:

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

2. iRPickTool Basic (basic functions) startup procedures

This chapter explains the setup procedures for iRPickTool Basic (basic functions).

By performing the procedures as explained in this chapter, you will complete the setup of systems that use four-axis delta robots as shown in the figure below.

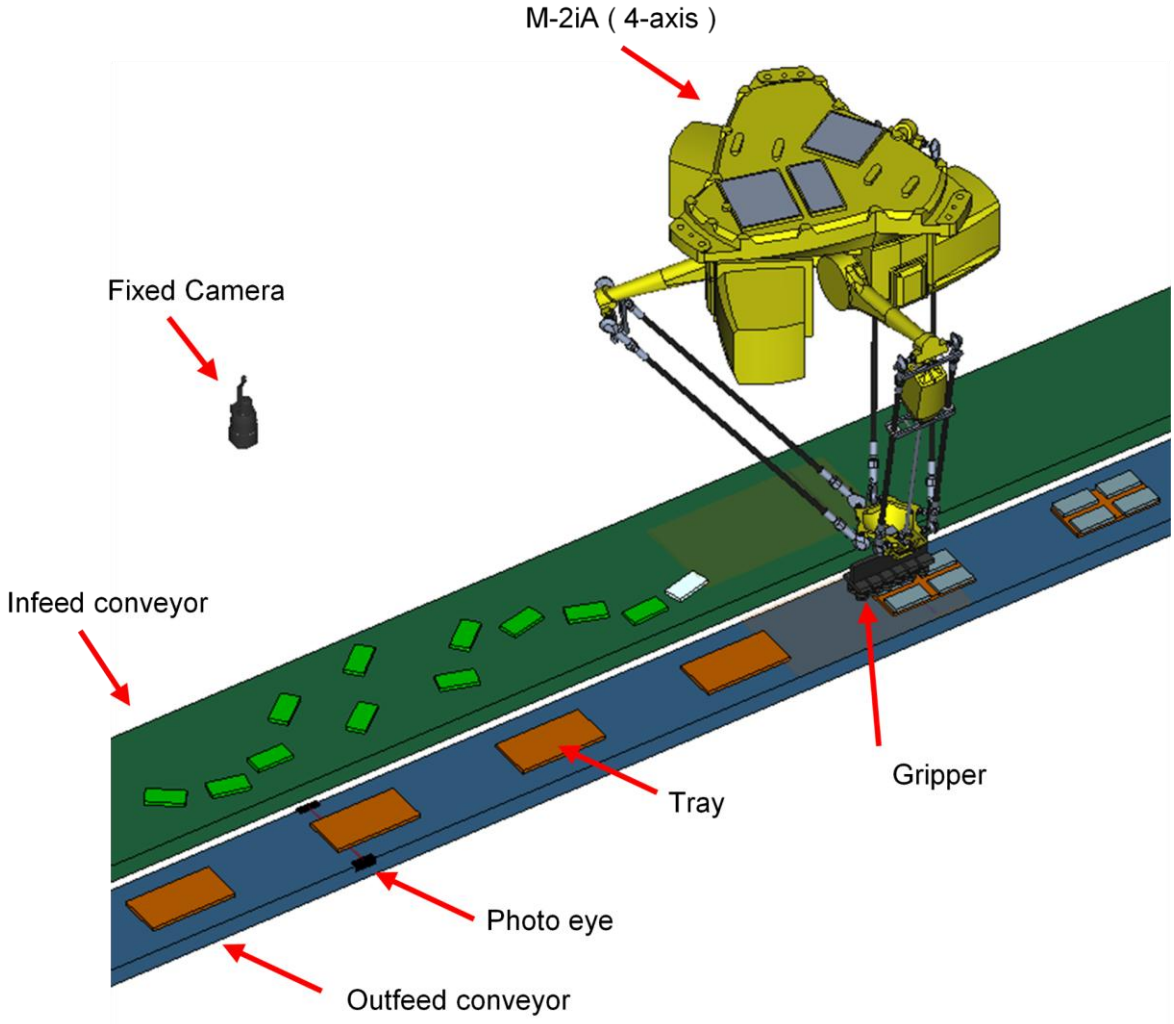


Fig. 2-1 System configuration

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

This chapter explains the following combinations as system configuration examples.

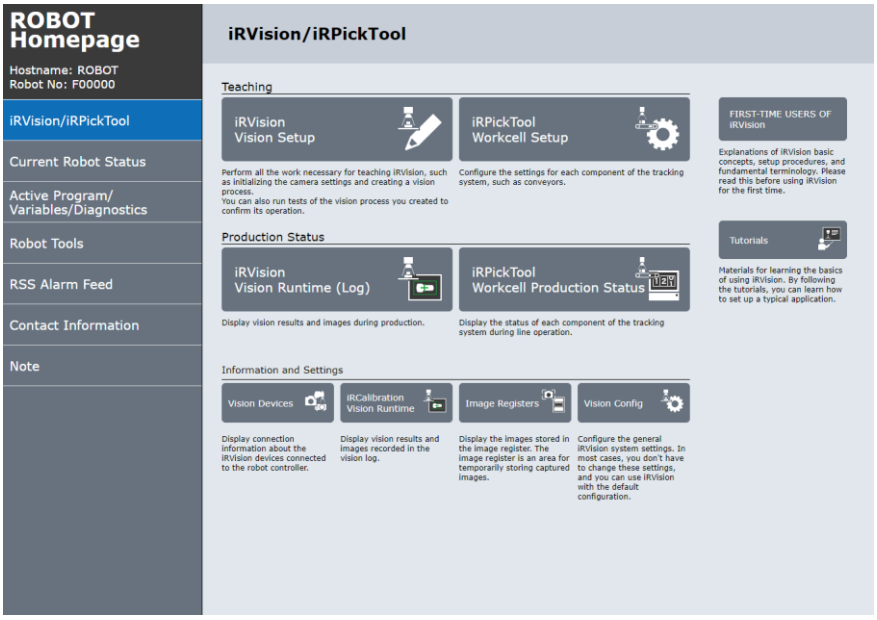
Infeed	: [Distance] + [Find part by vision]
Outfeed	: [DI] / [RI] / [HDI] + [Put part in queue] (Use tray)

For information on the setup procedures when conveyors and fixed stations in other combinations are used and details on each tree item, refer to “I.5. Basic functions reference(P.258)”.

2.1. Access to the robot homepage

Before opening the iRPickTool screen, follow the procedure below to display the robot homepage.

1. **Start an internet browser.**
2. **Enter the IP address of the robot controller in the address bar.**
   The homepage of the robot will be displayed.



All robot controllers have their robot homepages. When the robot controller has iRPickTool options, the following links related to iRPickTool appear on the robot homepage.

Setting up a workcell	Opens the edit screen for setting up iRPickTool. For details, see “I.5. Basic functions reference(P.258)” or “I.6. Plug & Play reference(P.421)”.
Workcell product status	Opens the edit screen for displaying the system production status set with iRPickTool. For details, see “I.8.2. Checking the production status(P.567)”.

2.2. Preparation before setting up the application

This section explains the preparation for the setup of tracking systems that use iRPickTool, such as the connection of hardware and creation of programs.

2.2.1. Connection and setup of Pulsecoders

Install a Pulsecoder and connect it to a robot controller.

After connecting, set parameters on the teach pendant of the 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”.

In the tracking system, the control unit may contain the sensor I/F board as the signal interface, and the sensor distribution board may be connected to it as needed. Alternatively, the separate interface device (SDU) may be attached in the cabinet. When the SDU is attached, an FSSB optical fiber cable for connection is also required.

In addition, cabling (encoder cables) for external hardware (encoder) is required to track conveyors, pallets, tables, or other items. Only the αiA4000S Pulsecoder A860-2052-T321 (both incremental and absolute) can be used in the R-50iA / R-50iA Mate Controller. For the specifications, see Fig. C.2-1(P.712), Table C.2-1(P.712), and Fig. C.2-2(P.713).

Hardware	Description
Interface device	Insert the sensor I/F board into the appropriate slot of the control unit. (See Fig. C.1-1(P.711)) When connecting two encoders to the sensor I/F board or when using the force sensor at the same time, attach the sensor distribution board in the cabinet. (See Fig. C.1-2(P.711)) For example, when you want to use three or more encoders, attach the separate interface device (SDU) in the cabinet.
Encoder	Use the αiA4000S Pulsecoder A860-2052-T321. To obtain a sufficient resolution, use an appropriate gear and reducer. (300 pluses/mm or more is recommended)

2.2.1.1. Connecting Pulsecoders

Connect the Pulsecoder to a robot controller.

Connecting the Pulsecoder

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

> The following figure assumes that the sensor I/F board and sensor distribution board are attached to the robot controller.
>
> **MEMO** For how to attach the sensor I/F board and sensor distribution board to the robot controller, see "Line Tracking OPERATOR'S MANUAL"(B-84464EN-10).

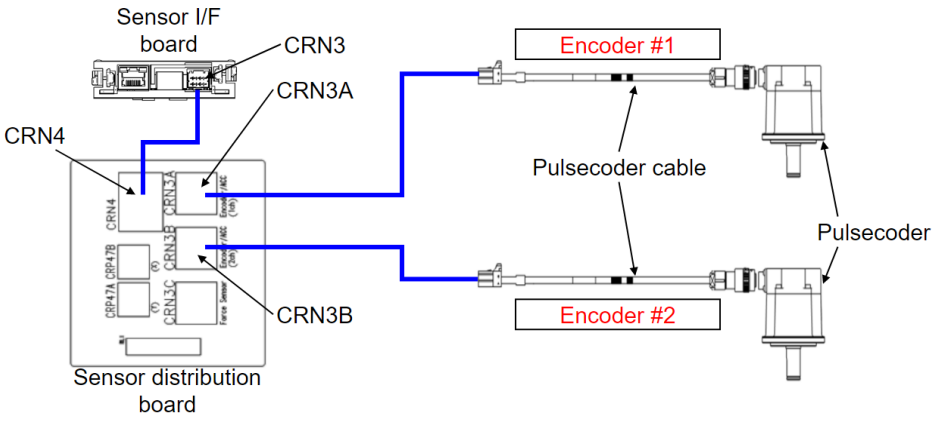


Fig. 2.2.1.1-1 Connection between the sensor I/F board, sensor distribution board, and Pulsecoders

2.2.1.2. Setting up Pulsecoders

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

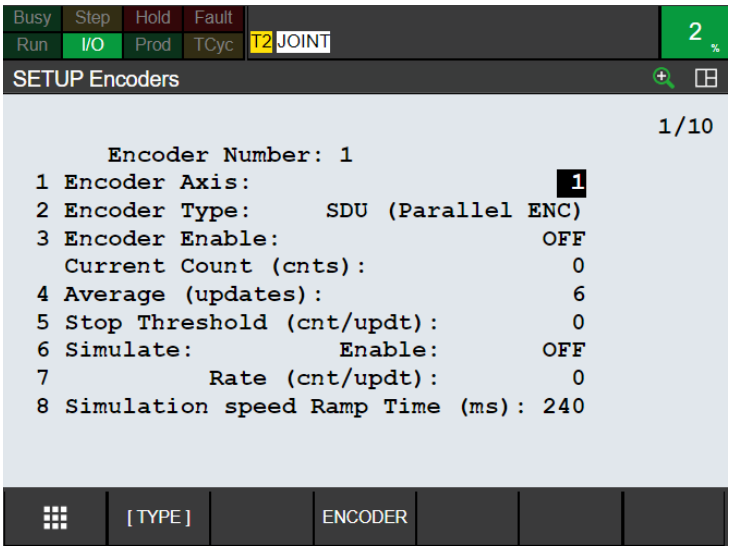
- [Encoder Axis]
- [Encoder Type]
- [Average (updates)]

1 **Press the [MENU] key on the robot controller's teach pendant.**
The menu appears.
2 **Select [SETUP] → [Encoders] from the menu.**

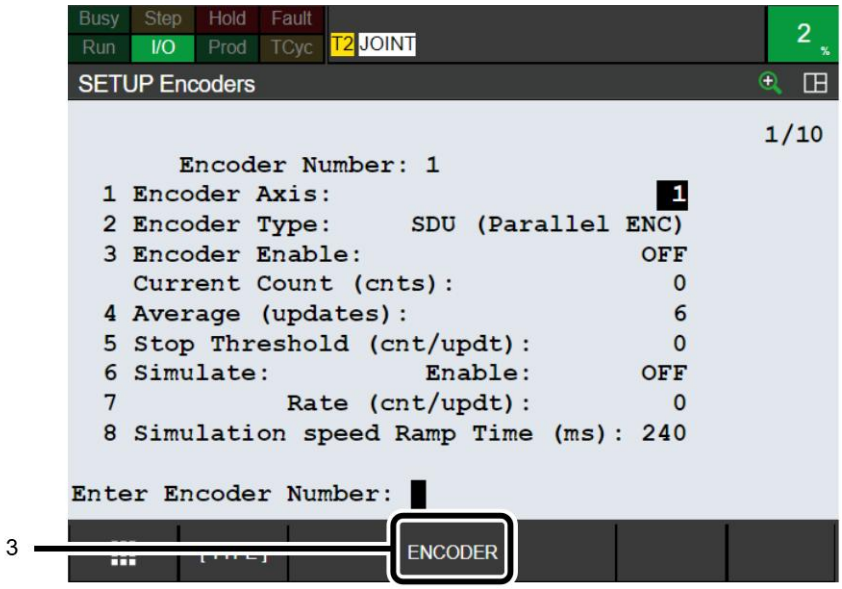


After a while, the following screen appears.

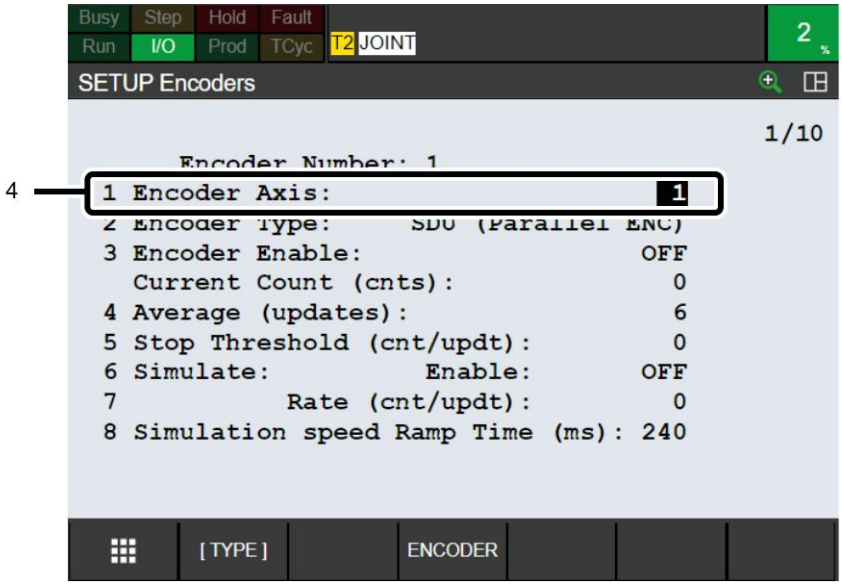
Number [9] and after in the setting items are the menu for optional functions. These will not be set at this time.



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



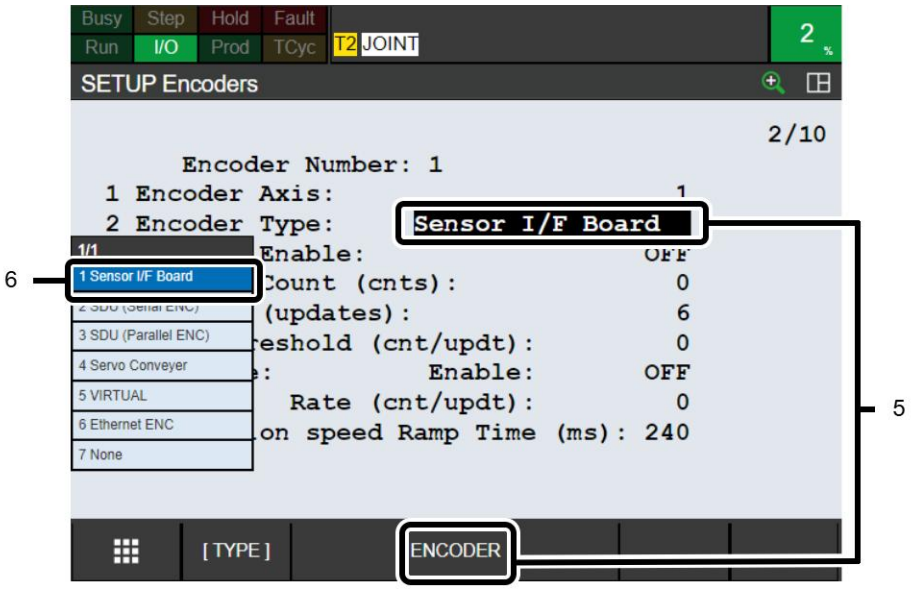
4 Move the cursor to [Encoder Axis] and input 1.



5 **Move the cursor to [Encoder Type] and press F4 [CHOICE].**

The menu appears.

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



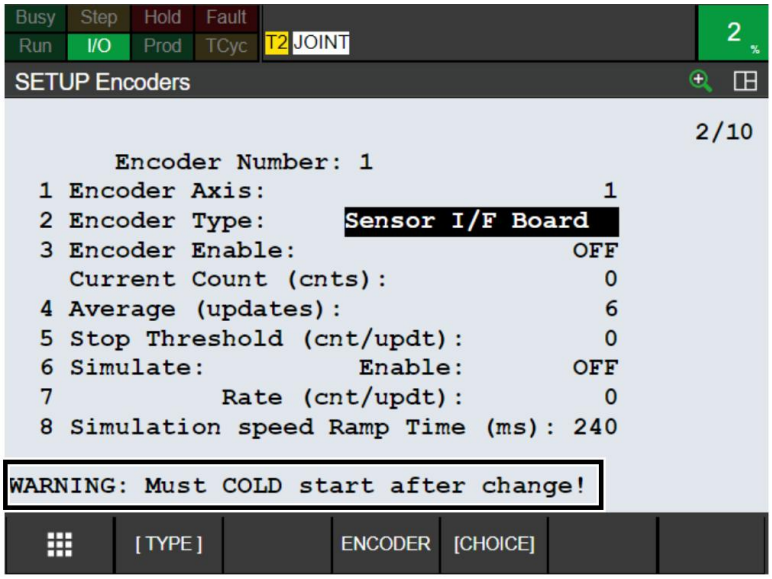
Encoder type

There are the following types for encoders.

- αiA4000S Pulsecoder (red Pulsecoder): [sensor I/F board]
- Incremental Pulsecoder (black Pulsecoder): [parallel incremental]
  - ∗ Cannot be connected to the sensor I/F board

	αiA4000S (A860-2052-T321)		Incremental Pulsecoder (A860-0301-T001 to T004)

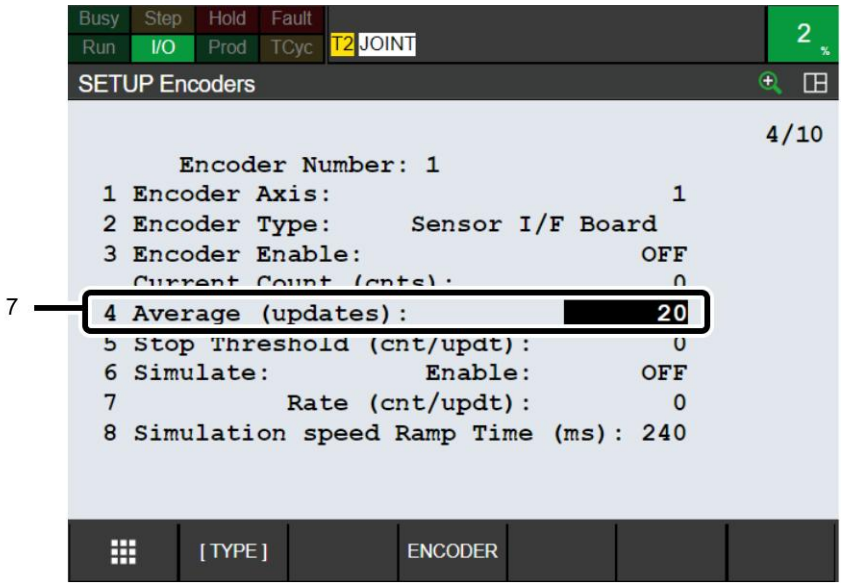
If encoder axes or encoder types are changed, a warning prompting to cold start the robot will appear.



7 Move the cursor to [Average (updates)] and input the cumulative accumulated number of instantaneous speeds (width of moving average).

By setting the accumulated number of instantaneous speeds, the robot will stop smoothly rather than suddenly, even if the conveyor suddenly stops while the robot is tracking.

Normally, enter “20”.

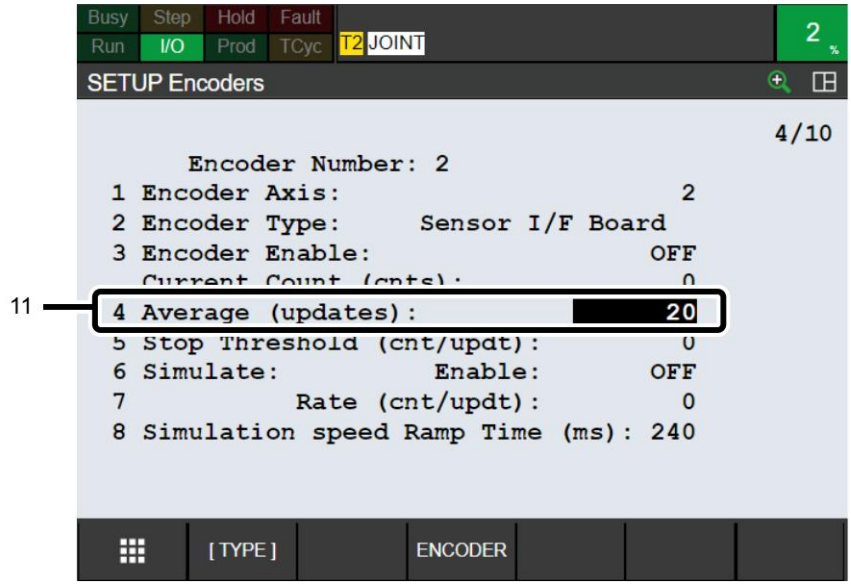


8 Next, set the parameters for the Pulsecoder that is installed at the outfeed conveyor. Press F3 [ENCODER], then input 2.

9 Move the cursor to [Encoder Axis] and input 2.

10 Move the cursor to [Encoder Type], press F4 [CHOICE], and select the encoder type that corresponds to the Pulsecoder that is to be connected.

11 Move the cursor to [Average (updates)] and input a value. This is the same as step 7. Normally, enter “20”.

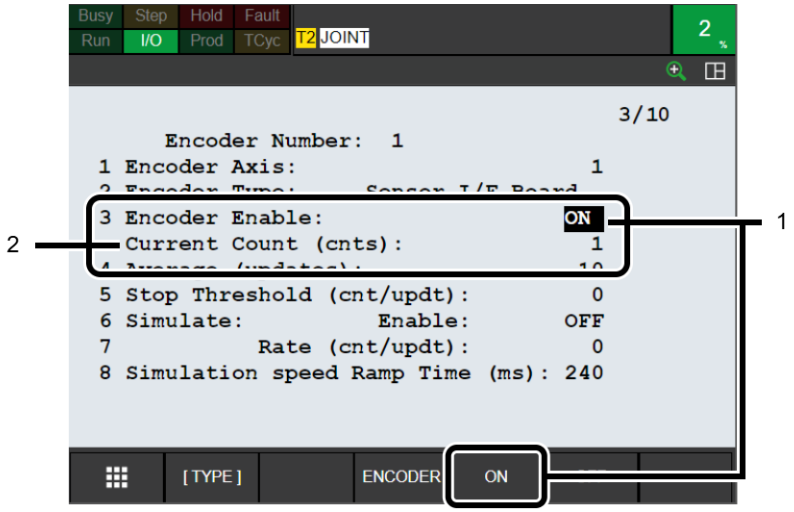


12 Finally, cold start the robot controller.

2.2.1.3. Checking the Pulsecoder settings

1 Open the [Encoders] screen and move the cursor to [Encoder Enable], then press F4 [ON].

2 Move the conveyor and confirm that the pulse count is displayed at [Current Count (cnts)].



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

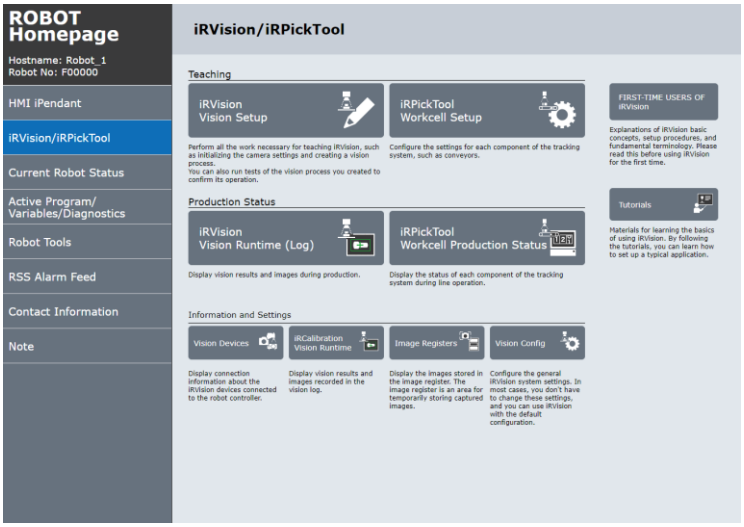
2.2.2. Checking the connection of the camera

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

For details on the creation of camera data and setting items, see "iRVision OPERATOR'S MANUAL (Reference)"(B-84434EN).

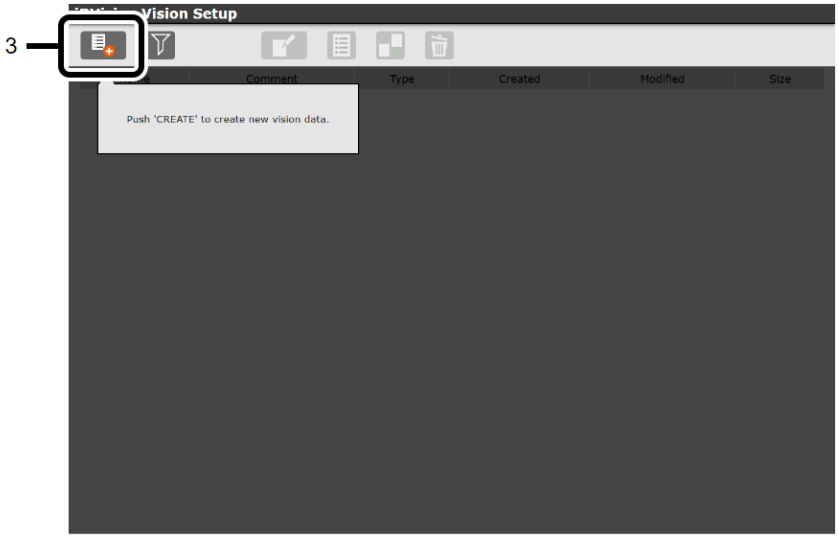
1 Open the robot homepage.

2 Select [iRVision/iRPickTool] in the menu area and click [iRVision Vision Setup] in the content display area.

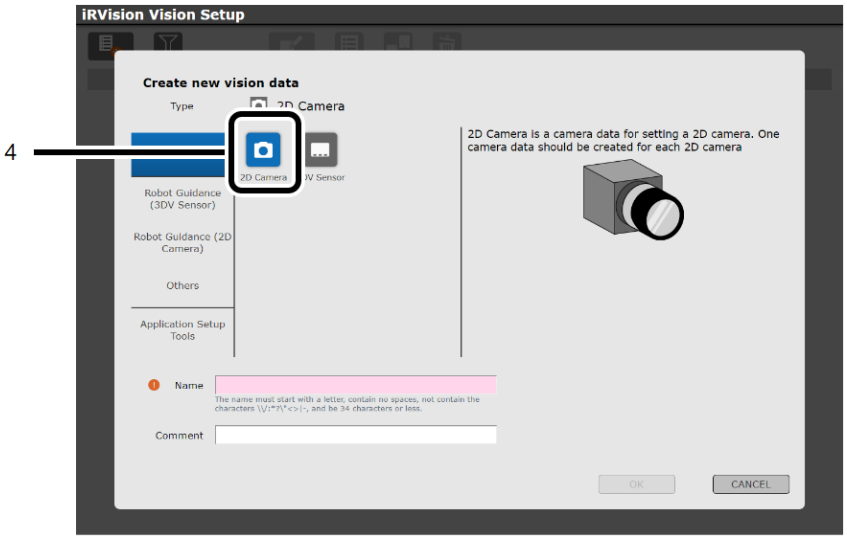


The [iRVision Vision Setup] screen appears.

3 Click [CREATE].



4 Select [2D Camera] in the [Camera Data] category in the displayed pop-up.

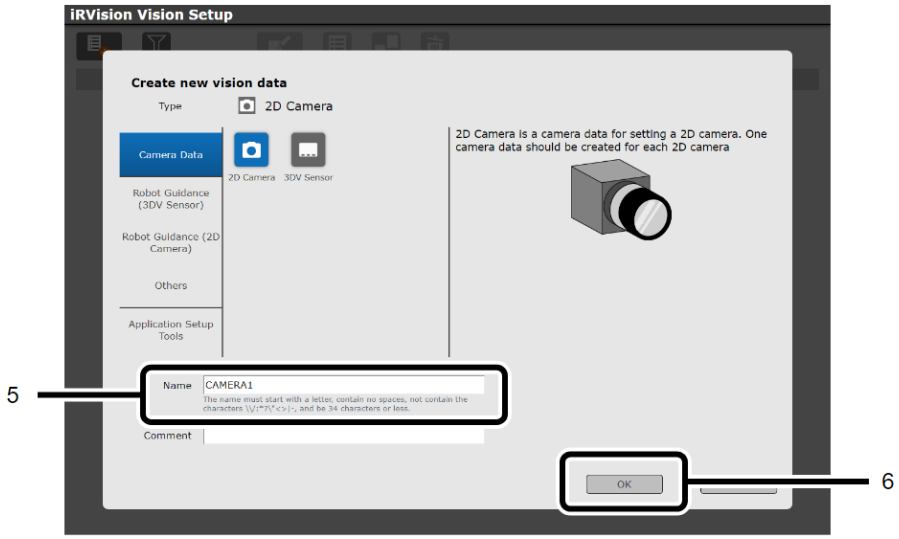


5 Input the name of the camera in [Name].

Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

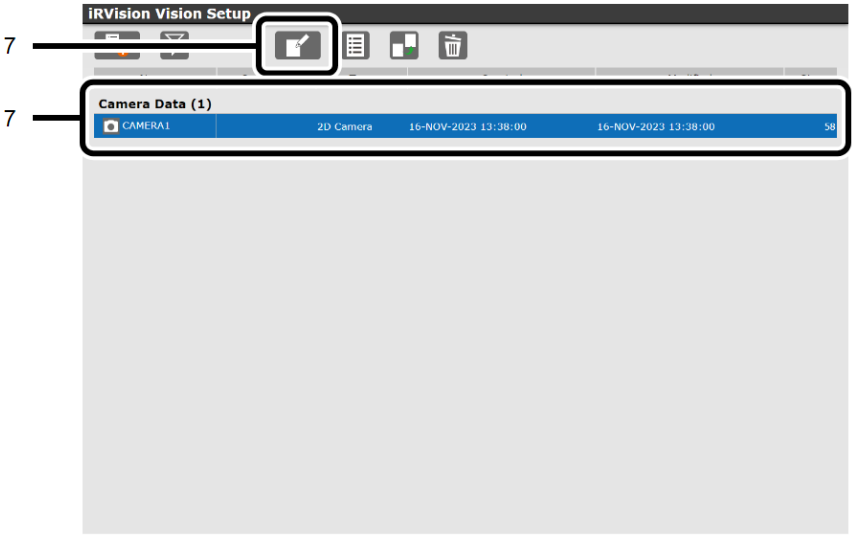
Example: CAMERA1

6 Click [OK].

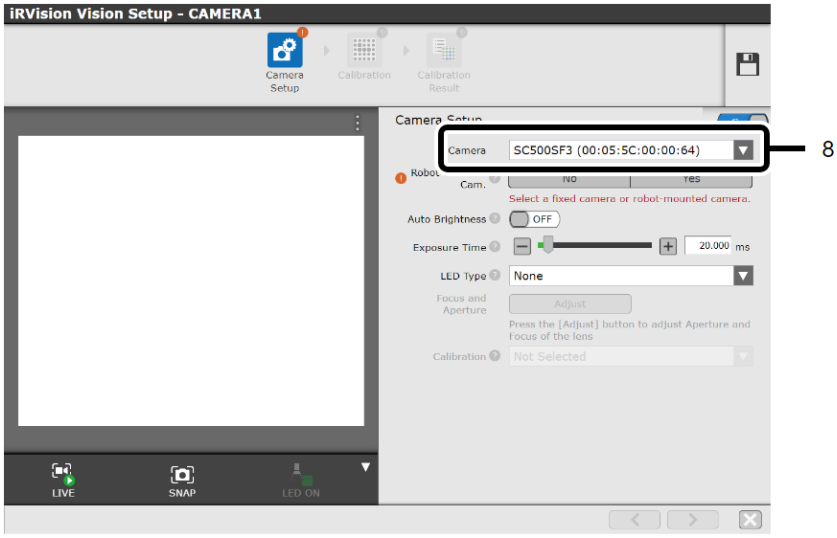


The created camera data is displayed in the list.

7 Select the created camera data and click [EDIT].



8 Select the connected camera in the [Camera] drop-down list.



9 To adjust the exposure, place something that displays high contrast characters within the field of view of the camera on the conveyor surface, then click [LIVE].

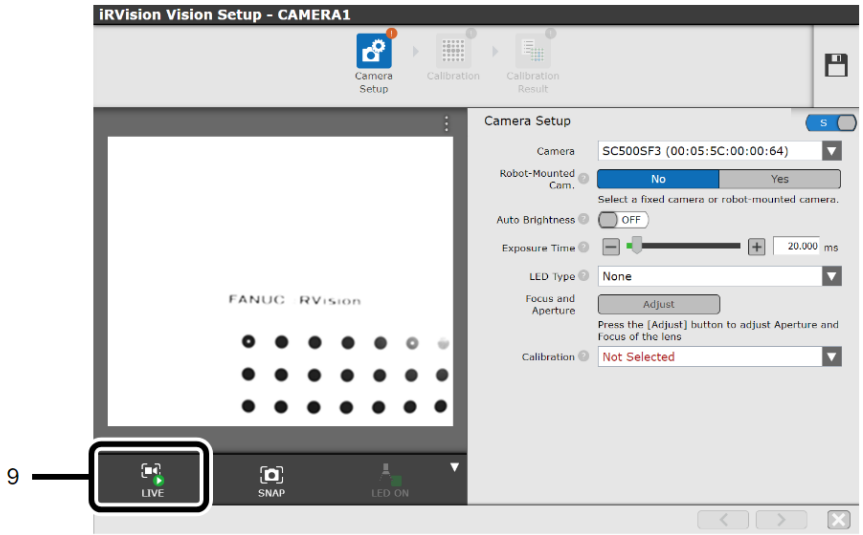
Confirm that the item that was placed has been captured within the frame on the left side of the screen.

If the screen is dark, adjust the following.

- Open the diaphragm of the lens.
- Adjust the lighting brightness.

10 Check the field of view of the camera.

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



11 When the camera position is determined, click the [Adjust] button in [Aperture and focus] to open the focus adjustment guide and adjust the aperture and focus according to the instructions in the guide.



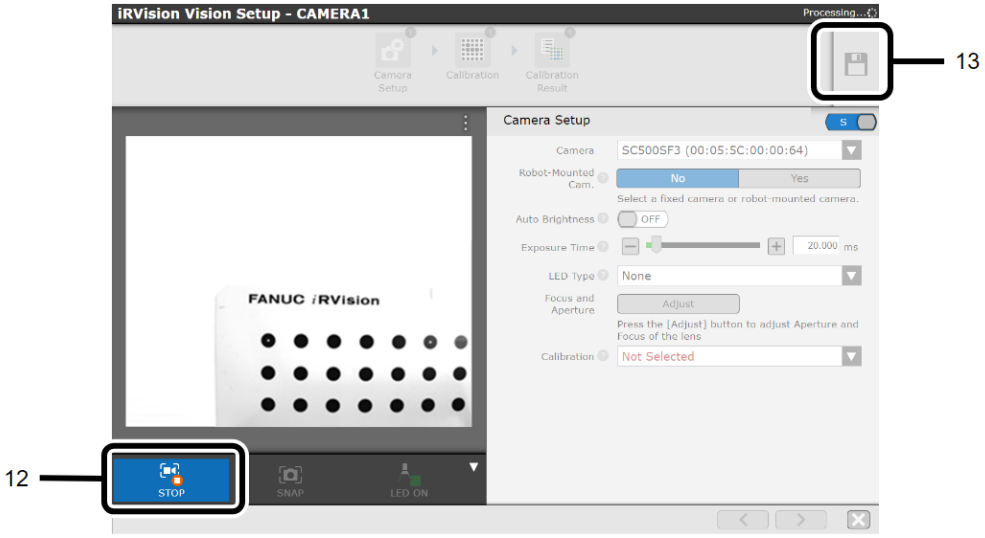
Fig. 2.2.2-1 Focus adjustment

12 Click [STOP].

Live will stop.

13 Click [SAVE].

The edit screen will close.



2.2.3. Checking the input of the trigger sensor

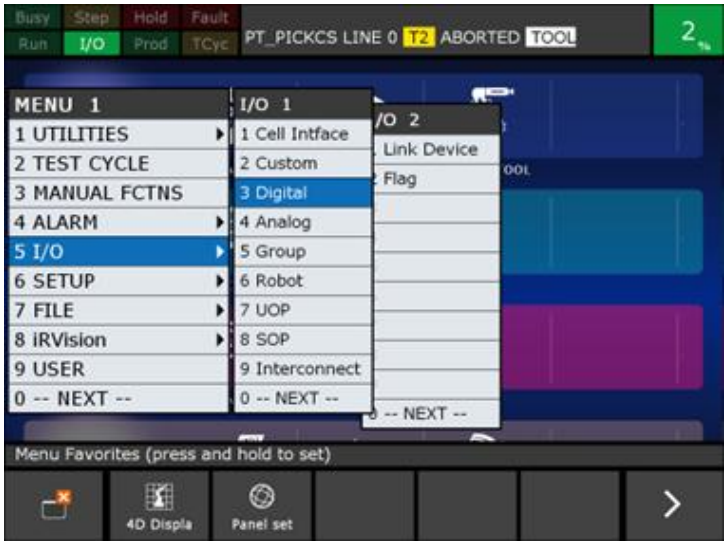
Check the input of the trigger sensor.

2.2.3.1. Using DI / RI

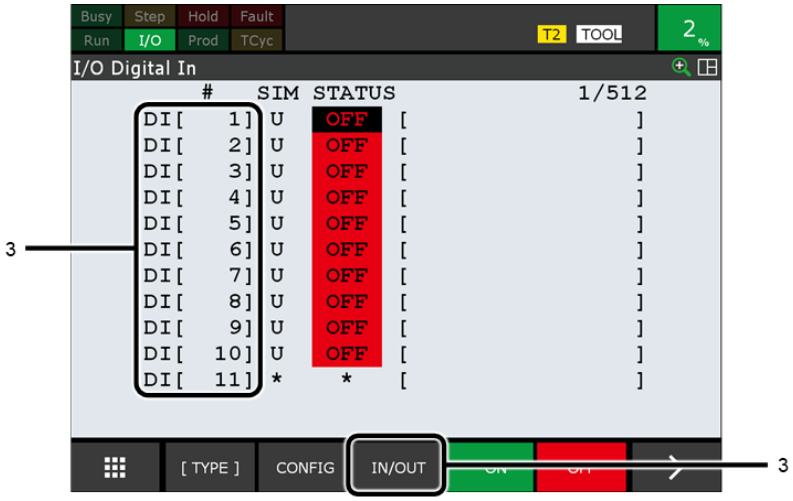
Check the input of the trigger sensors when DI / RI are used.

1 **Press the [MENU] key on the robot controller's teach pendant.**
The menu appears.

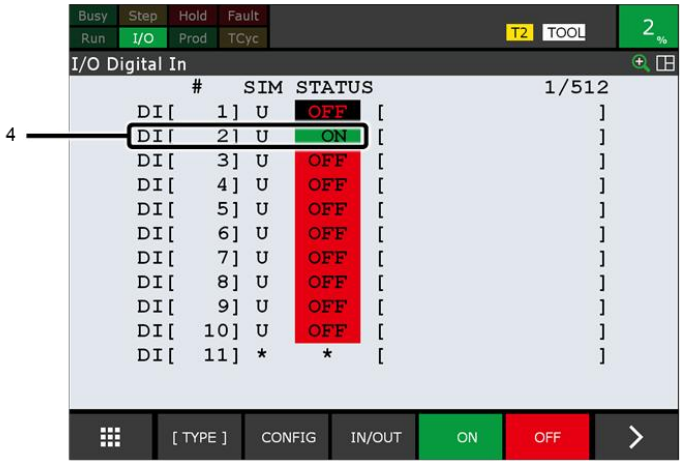
2 **If you are using DI, select [I/O] → [Digital] from the menu. If you are using RI, select [I/O] → [Robot] from the menu.**
For example, when DI is used the screen will look like this.



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



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



2.2.4. Setting up the tool frame of the pointer tool

Set up the tool frame of the pointer tool.

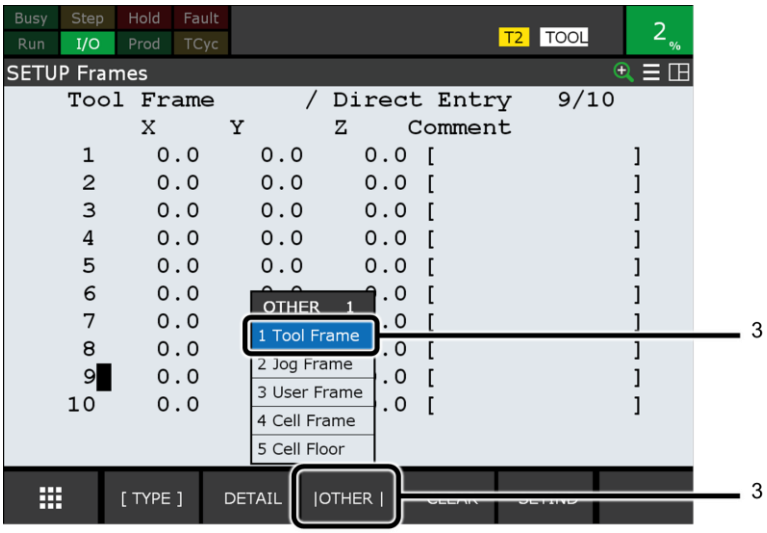
1 Press the [MENU] key on the robot controller's teach pendant.
The menu appears.

2 From the menu, select [SETUP] → [Frames].



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

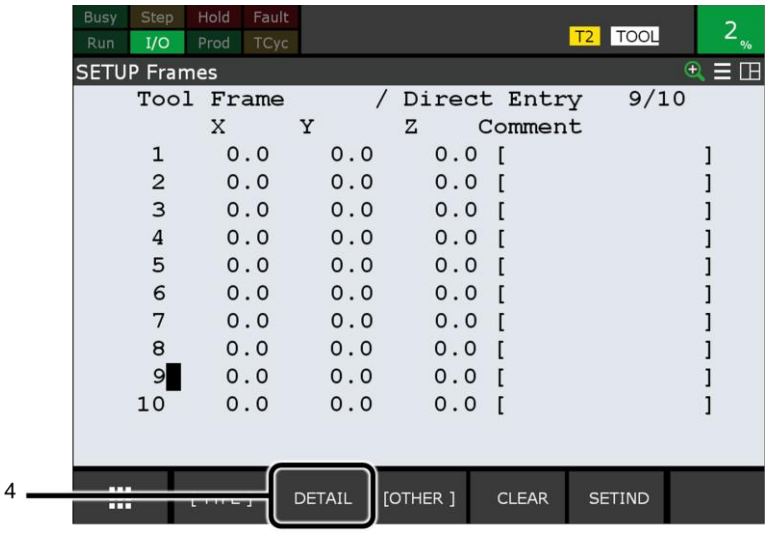
The menu appears.



4 Move the cursor to the line of the tool frame number to be set and press F2 [DETAIL].

The setup screen for the tool frame for the selected frame number will appear.

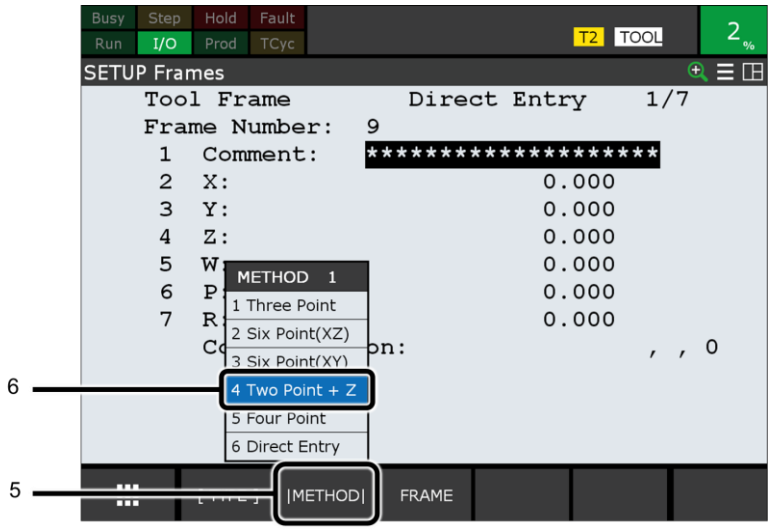
This screen is an example of when performing setup using Tool 9.



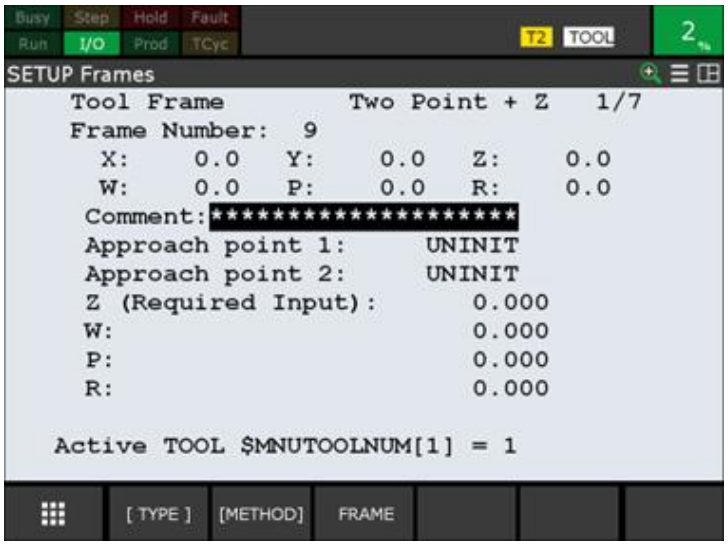
5 Press F2 [METHOD].

The menu appears.

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



After a while, the following screen appears.

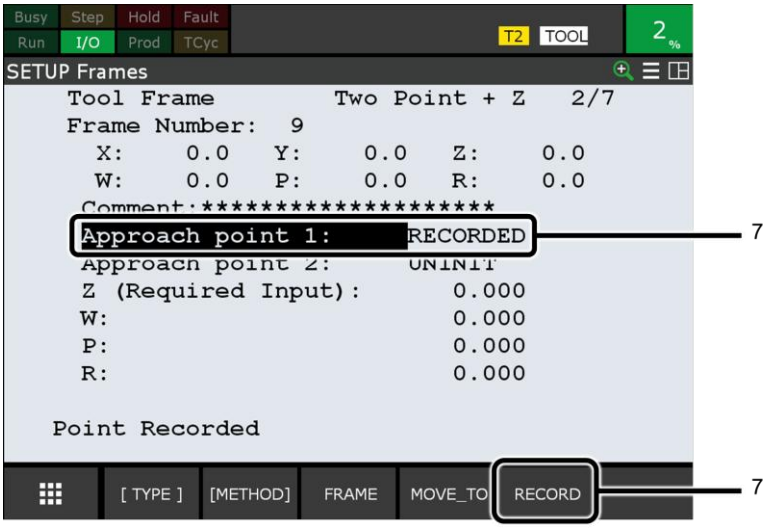


7 Move the cursor to [Approach point 1] and input approach point 1.

Jog the robot to the approach point to be recorded.

While holding down the [SHIFT] key on the teach pendant, press F5 [RECORD] to record the current position as the approach point.

'RECORDED' will be displayed.

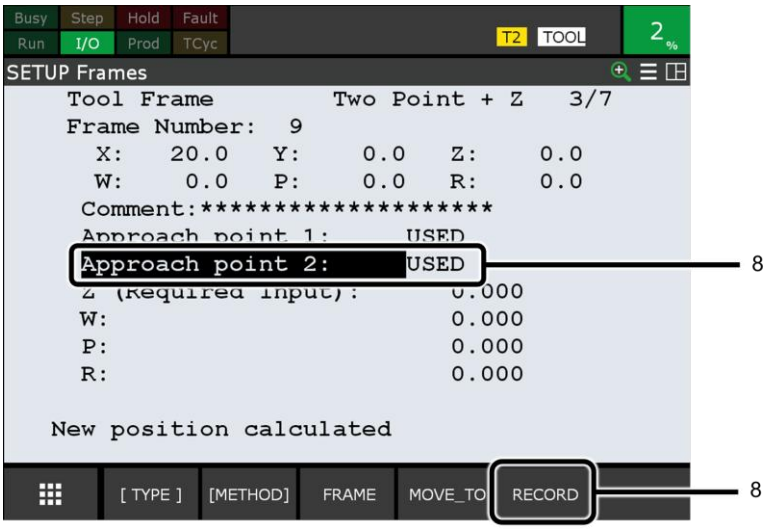


8 Move the cursor to [Approach point 2] and input approach point 2.

Jog the robot to the approach point to be recorded.

While holding down the [SHIFT] key on the teach pendant, press F5 [RECORD] to record the current position as the approach point.

When all the approach points are input, 'USED' will be displayed and the X and Y of the tool frame will be set.



CAUTION

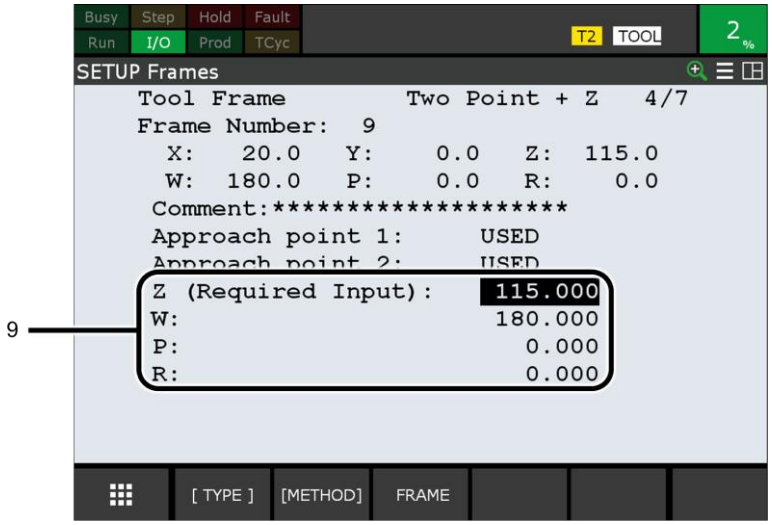
1. When inputting approach points, have the flange surface facing down.
2. Teach approach points 1 and 2 at different robot posture.
3. If the above conditions are not met, a message saying 'Invalid set of input points' will appear.

9 Move the cursor to [Z], [W], [P] and [R] and enter a value for each.

For [Z], input a numerical value measured with a ruler, etc.

Input [W], [P] and [R] directly.

* (W, P, R) = (180, 0, 0) is recommended.



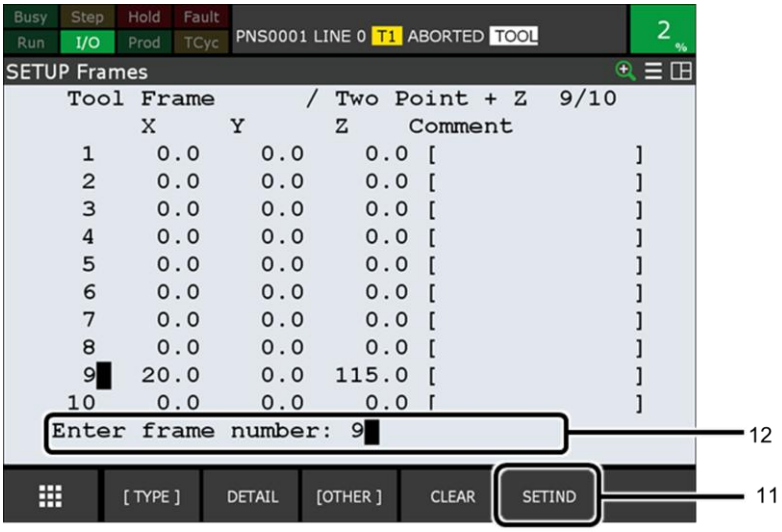
10 Press the [PREV] key on the teach pendant of the robot controller.

The Tool Frame list screen appears.

The setting values of all the tool frames can be checked.

11 Press F5 [SETIND].

12 Input the frame number of the tool frame.

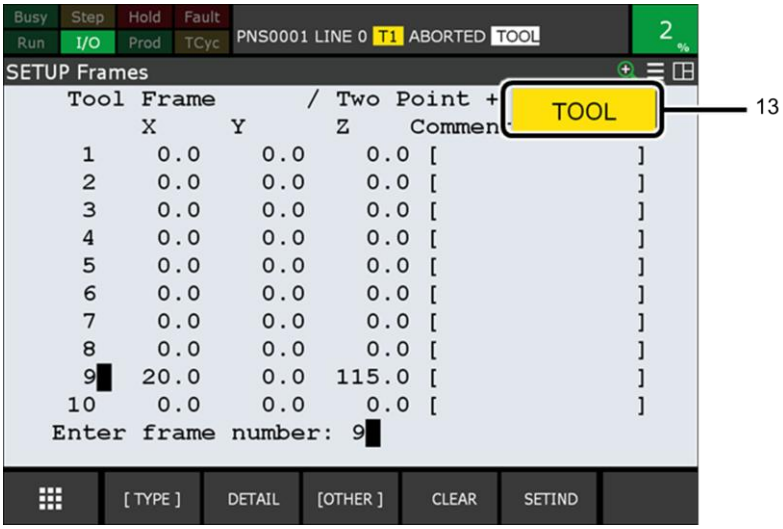


The tool frame will be enabled.

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

13 Set the manual continuous feed of the robot in ‘TOOL’.

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



14 Press the -Z (J6) key or +Z (J6) key on the teach pendant of the robot controller.

The robot jogs when it rotates around the Z axis.

Verify that the tool frame (TCP) has been set accurately.

2.2.5. Creating a robot program

Create a robot program.

Create the following robot programs.

- Main program
- Tracking program
- Vacuum ON/OFF program

Creating a main program

Create a main program.

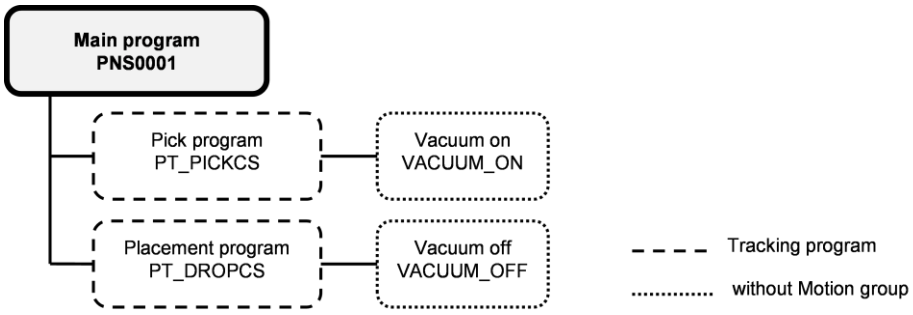


Fig. 2.2.5-1 Main program positional relationship

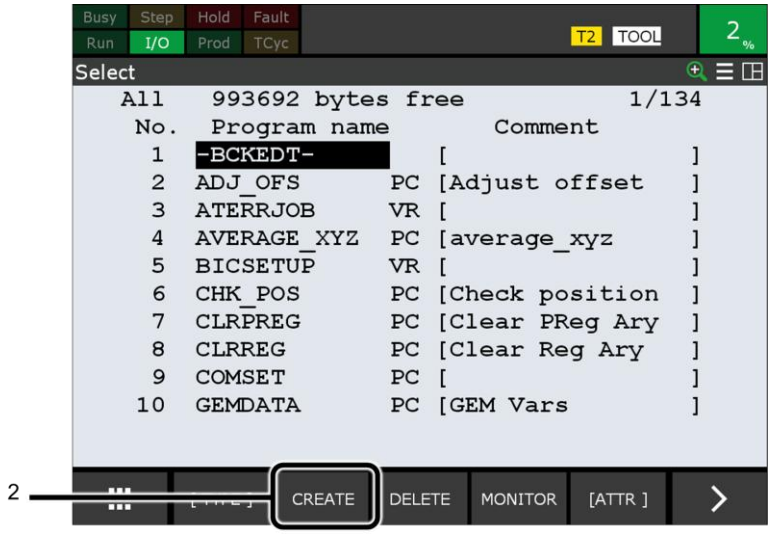
1 Press the [SELECT] key on the teach pendant of the robot controller.

A program list screen will appear.

2 Press F2 [CREATE].

A screen to create programs will appear.

If [CREATE] is not displayed, press [> (NEXT)] to switch screens.

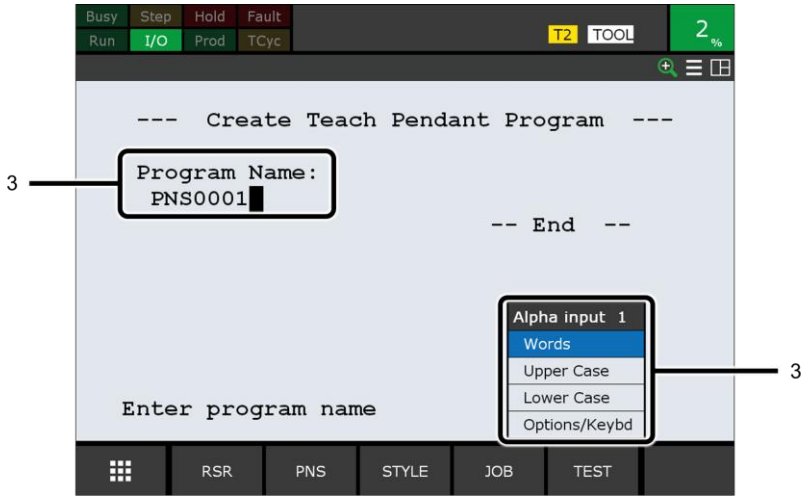


3 Move the cursor to [Program Name] and input a program name.

The input method for program names can be selected from among [Words], [Upper Case], [Lower Case] and [Options/Keybd], which are displayed in a pop-up.

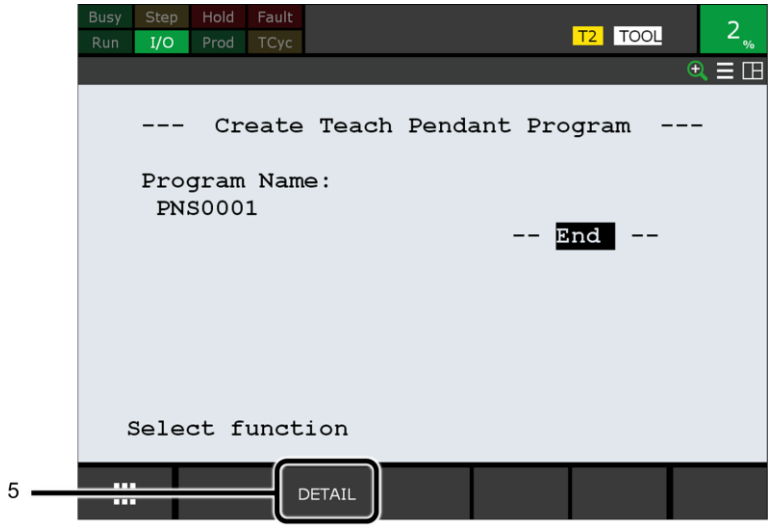
Example: Here, the name is set to PNS0001, on the assumption that PNS will start up.

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



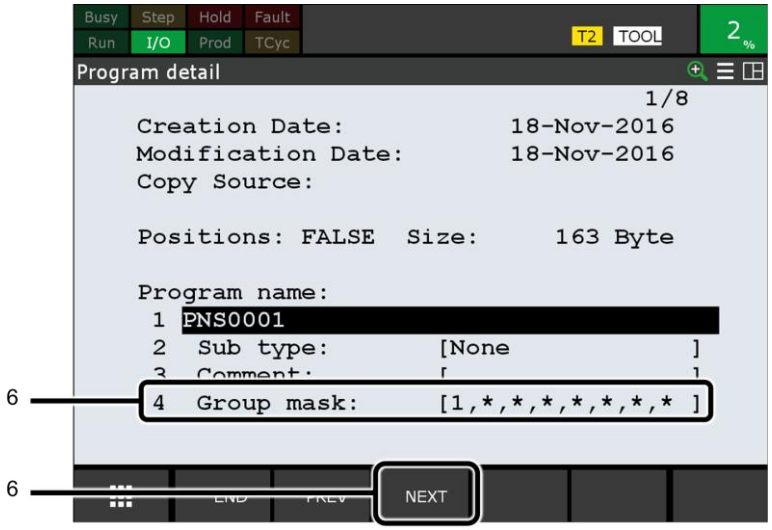
5 Press F2 [DETAIL].

A program details screen will appear.



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

A tracking program details screen will appear.



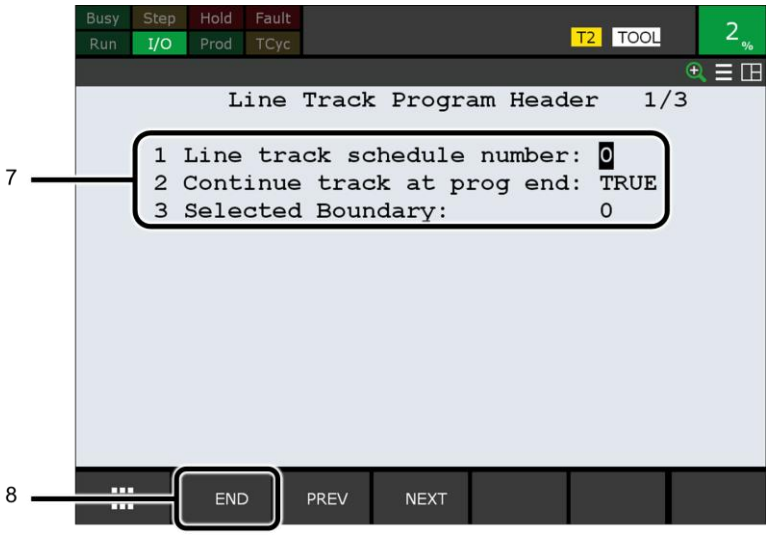
7 Confirm that '0' has been set in [Line Tracking Schedule].

8 Press [END].

A main program will be created.

* [Group mask]: [1,*,*,*,*,*,*,*]
[Line Tracking Schedule]: 0

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



Creating a tracking program

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

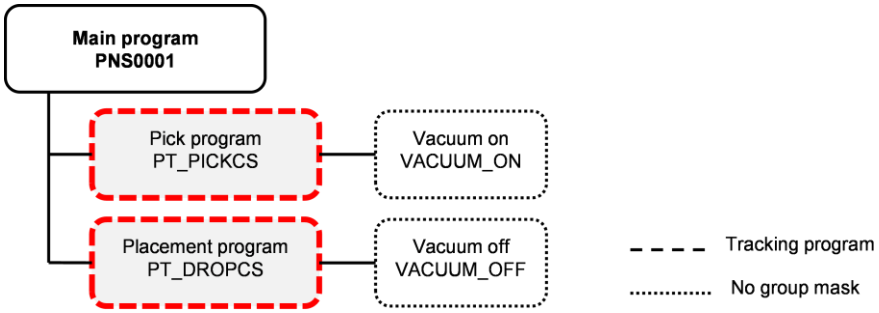


Fig. 2.2.5-2 Tracking program positional relationship

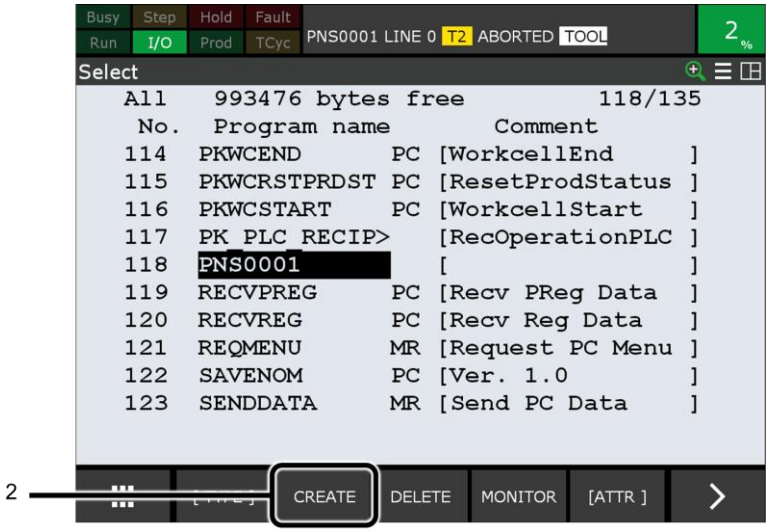
1 Press the [SELECT] key on the teach pendant of the robot controller.

A program list screen will appear.

2 Press F2 [CREATE].

A screen to create programs will appear.

If F2 [CREATE] is not displayed, press [> (NEXT)] to switch screens.

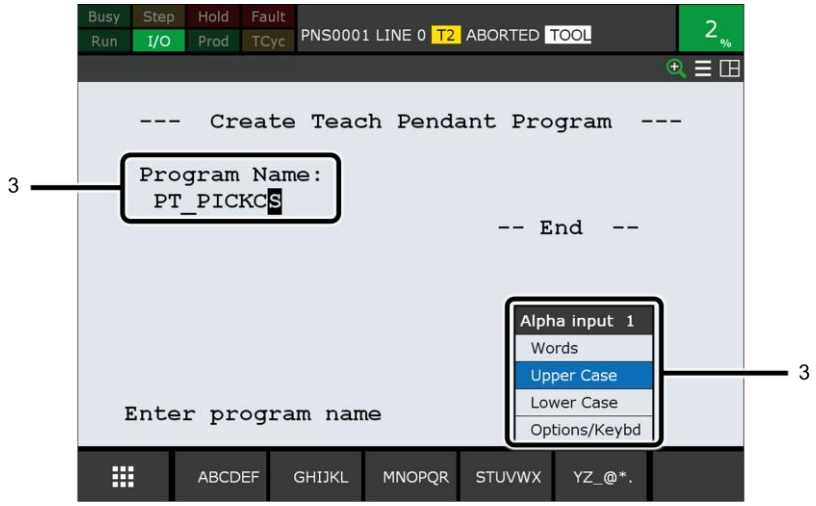


3 Move the cursor to [Program Name] and input a program name.

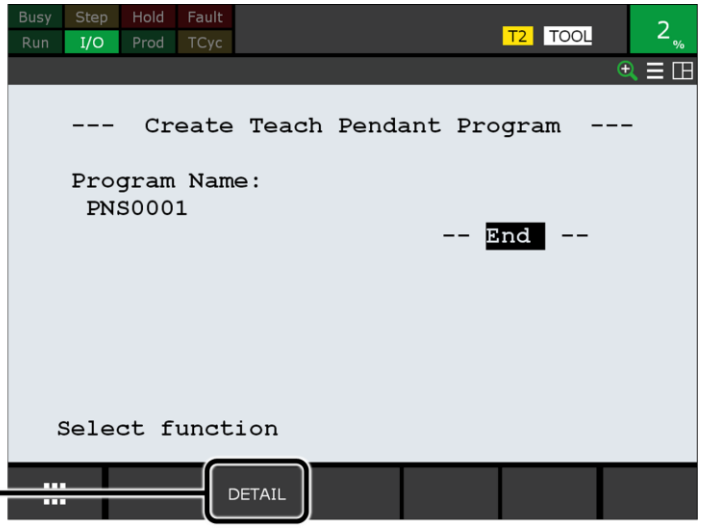
The input method for program names can be selected from among [Words], [Upper Case], [Lower Case] and [Options/Keybd], which are displayed in a pop-up.

Example: PT_PICKCS, PT_DROPCS

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

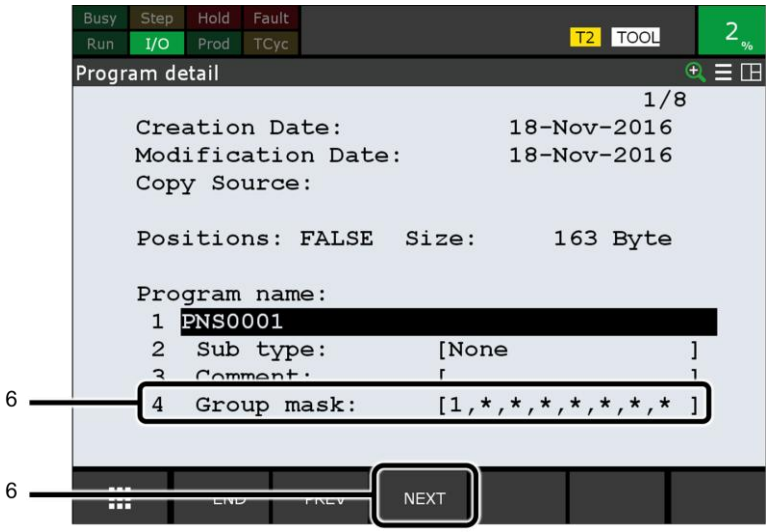


5 Press F2 [DETAIL].



A program details screen will appear.

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



The following tracking program details screen will appear.

7 Set [1] or [2] in [Line Tracking Schedule].

In the case of picking (PICK): 1

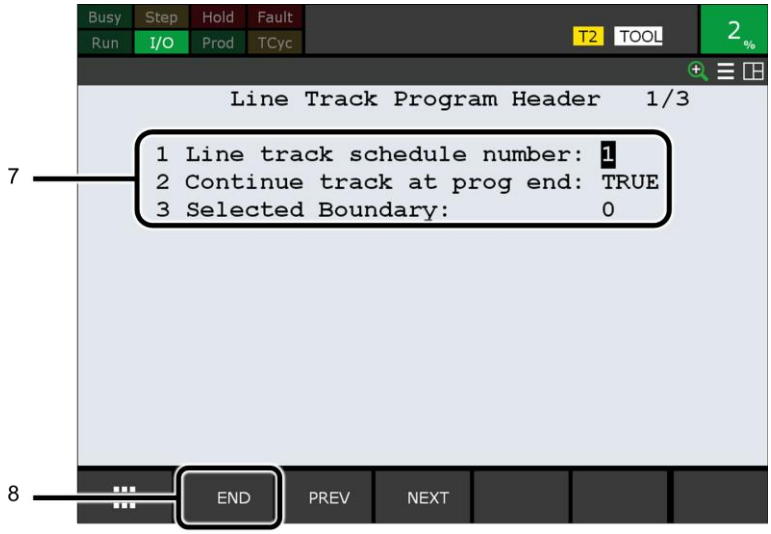
In the case of placing (DROP): 2

8 Press F1 [END].

A tracking program will be created.

*	[Group mask]:		[1,*,*,*,*,*,*]
	[Line Tracking Schedule]	Picking (PICK):	1
		Placing (DROP):	2

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



Creating a vacuum ON/OFF program

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

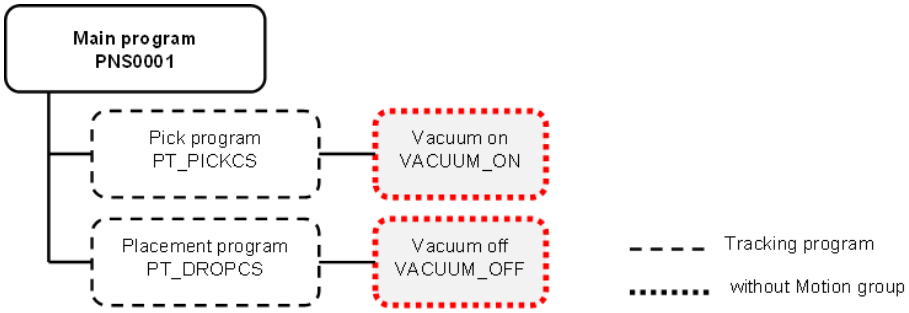


Fig. 2.2.5-3 Vacuum ON/OFF program positional relationship

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

A program list screen will appear.

2 Press F2 [CREATE].

A screen to create programs will appear.

If F2 [CREATE] is not displayed, press [> (NEXT)] to switch screens.

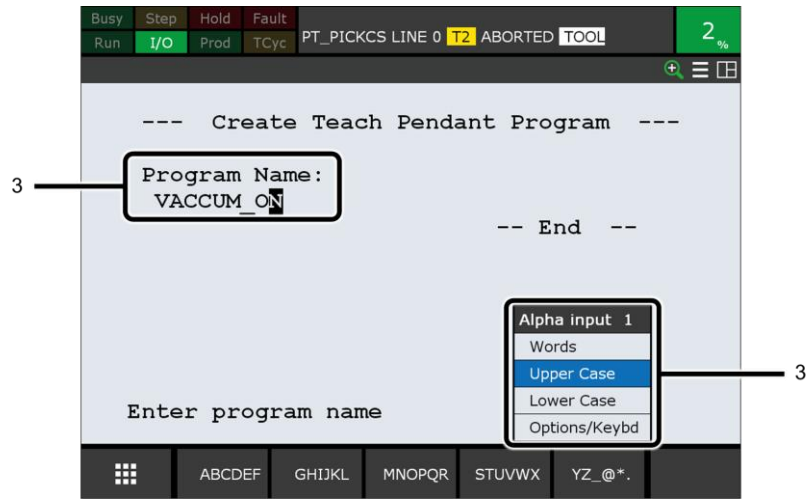


3 Move the cursor to [Program Name] and input a program name.

The input method for program names can be selected from among [Words], [Upper Case], [Lower Case] and [Options/Keybd], which are displayed in a pop-up.

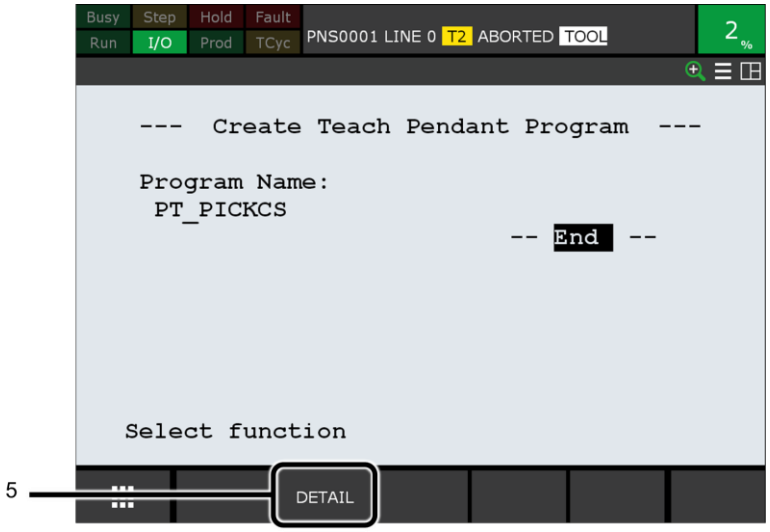
Example: VACUUM_ON, VACUUM_OFF

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



5 Press F2 [DETAIL].

A program details screen will appear.



6 Move the cursor to [Group mask] and change all of the group mask to '*'.

You can change it by pressing F5 [*] after moving the cursor to the location you want to change.

7 Press F1 [END].

A vacuum ON/OFF program will be created.

* [Group mask]: [*,*,*,*,*,*,*,*]
[Line Tracking Schedule]: 0

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

Sample program

Here are examples for each of the programs described in this section.

For information on KAREL programs, refer to "Using KAREL programs" below.

An example of a main program (PNS0001) is shown below.

PNS0001

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

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

PT_PICKCS

```
  1:   UFRAME_NUM=0
  2: UTOOL_NUM=1
  3:
  4: LBL[101:QUE check]
  5: STOP_TRACKING
  6: !PKCSGETQUE (Get part information)
  7: !*1:Register number for conveyor station ID
  8: !*2:Usually 1. 2 or more in the case of consecutive acquisition
  9: !*3:Index delay msec
 10: !*4:Vision Register number for part information
 11: !*5:Process status register number
 12: !*6:Model ID (Can be omitted)
 13: CALL PKCSGETQUE
       (CStn ID=R[11:CONV1 ID],
       Consec Flag=1,Timeout (ms)=100,
       Offset VR=1,Stat Reg=2)                                  KAREL program
```

```
14: IF R[2:GETQ Status Pick]=0,JMP LBL[111]
15: IF DI[1:CYCLE_STOP]=ON,JMP LBL[901]
16: JMP LBL[101]
17:
18: LBL[111:Pick operation]
19: !ADJ_OFS (Adjust offset)
20: CALL ADJ_OFS                                          KAREL program
      (vision register,offset position/VR Index=1,
      Position Register number for adjustment amount=21,
      Adjusted offset position/VR Index=1)
21: L PR[7:Pick Pos]                                      Pick approach position
      R[26:Pick Speed1]mm/sec
      CNT100 VOFFSET,VR[1]
      Offset,PR[11:Pick Apr Z]
22: L PR[7:Pick Pos] R[27:Pick Speed2]mm/sec               Pick position
      CNT R[24:Pick CNT]
      VOFFSET,VR[1]
      AP_LDR[21]
      TB R[25]sec, CALL VACUUM_ON
23: WAIT R[23]
24: PAYLOAD[2]
25: L PR[7:Pick Pos] R[28:Pick Speed3]mm/sec               Pick retract position
      CNT100 VOFFSET,VR[1]
      Offset,PR[11:Pick Apr Z]
      RT_LDR[22]
26:
27: !PKCSACKQUE (Part information ! process acknowledgement)
28: !*1:Register number for conveyor station ID
29: !*2:Processing results. Normally 1
30: CALL PKCSACKQUE
      (CStn ID=R[11:CONV1 ID],Success=1)                  KAREL program
31:
32: LBL[901:Pick finished]
[End]
```

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

PT_DROPCS

```
 1:   UFRAME_NUM=0
 2: UTOOL_NUM=1
 3:
 4: LBL[101:QUE check]
 5: STOP_TRACKING
 6: !PKCSGETQUE (Get part information)
 7: !*1:Register number for conveyor station ID
 8: !*2:Usually 1. 2 or more in the case of consecutive acquisition
 9: !*3:Index delay msec
10: !*4:Vision Register number for part information
11: !*5:Process status register number
12: !*6:Model ID (Can be omitted)
13: CALL PKCSGETQUE
      (CStn ID=R[12:CONV2 ID],
      Consec Flag=1,Timeout (ms)=100,
      Offset VR=2,Stat Reg=3)                             KAREL program
```

```
14: IF R[3:GETQ Status Drop]=0,JMP LBL[111]
15: IF DI[1:CYCLE_STOP]=ON,JMP LBL[901]
16: JMP LBL[101]
17:
18: LBL[111:Place operation]
19: L PR[8:Drop Pos] R[36:Drop Speed1]mm/sec          Place approach position
      CNT100 VOFFSET,VR[2]
      Offset,PR[12:Drop Apr Z] Wjnt
20: L PR[8:Drop Pos] R[37:Drop Speed2]mm/sec          Drop position
      CNT R[34:Drop CNT]
      VOFFSET,VR[2]
      AP_LDR[31]
      TB R[35]sec, CALL VACUUM_OFF
21: WAIT R[33]
22: PAYLOAD[1]
23: L PR[8:Drop Pos] R[38:Drop Speed3]mm/sec          Place retract position
      CNT100 VOFFSET,VR[2]
      Offset,PR[12:Drop Apr Z]
      RT_LDR[32]
24:
25: !PKCSACKQUE (Part information ! process acknowledgement)
26: !*1:Register number for conveyor station ID
27: !*2:Processing results. Normally 1
28: CALL PKCSACKQUE
      (CStn ID=R[12:CONV2 ID],Success=1)             KAREL program
29:
30: LBL[901:Place finished]
[End]
```

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

VACUUM_ON

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

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

VACUUM_OFF

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

Register list

Display a register list screen using the following procedure.

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

Table 2.2.5-1 Register list

R[2:GETQ Status Pick]=0	Pick part processing status

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

Display a Position Register list screen using the following procedure.

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

Table 2.2.5-2 Position Register list

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

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



Robot operation

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

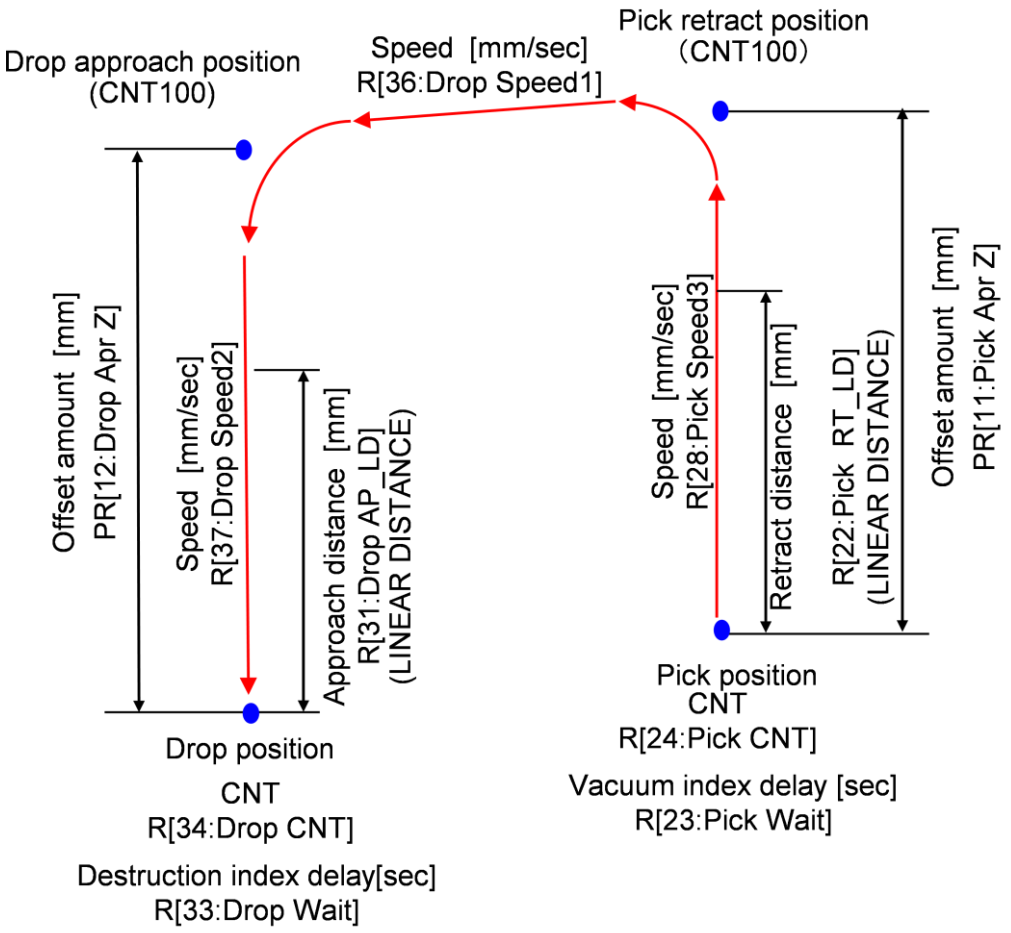


Fig. 2.2.5-4 Robot operation (Pick → Place)

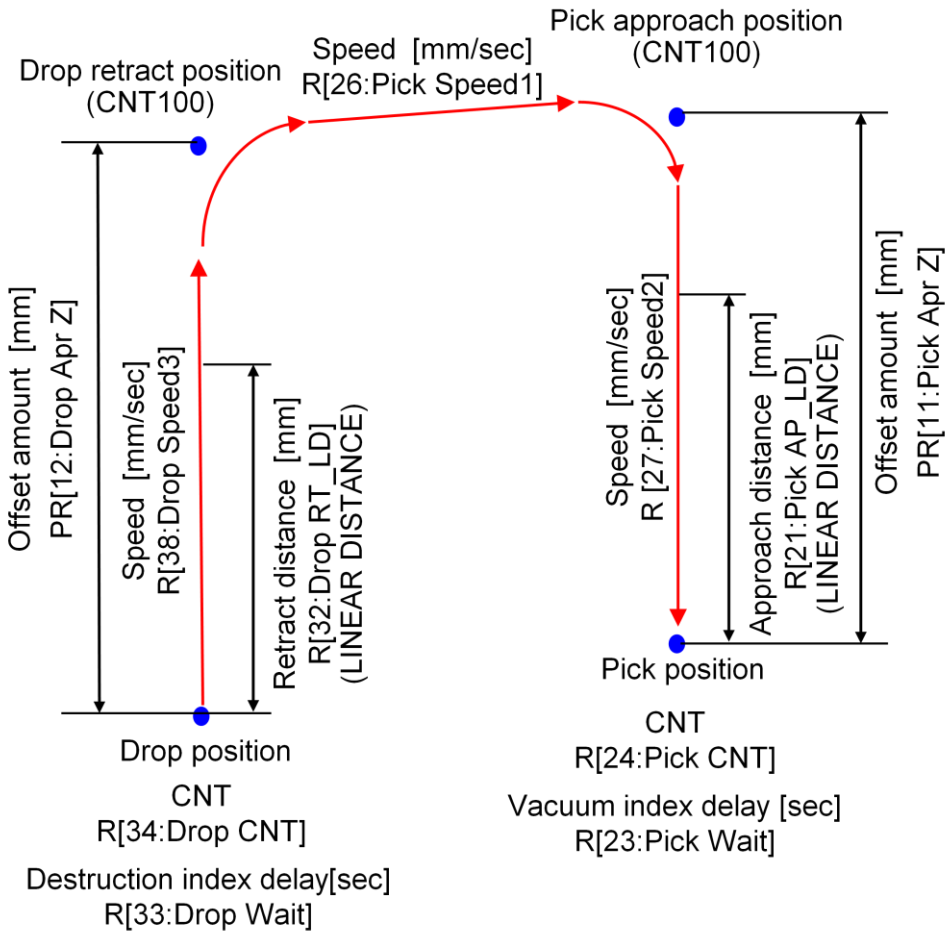


Fig. 2.2.5-5 Robot operation (Place → Pick)

Using KAREL programs

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

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



3. **Select [CALL program] from the menu.**



4 Press F3 [COLLECT].

'KAREL Progs' will be displayed in a list.

5 Select 'KAREL Progs' and press the [ENTER] key on the teach pendant of the robot controller.

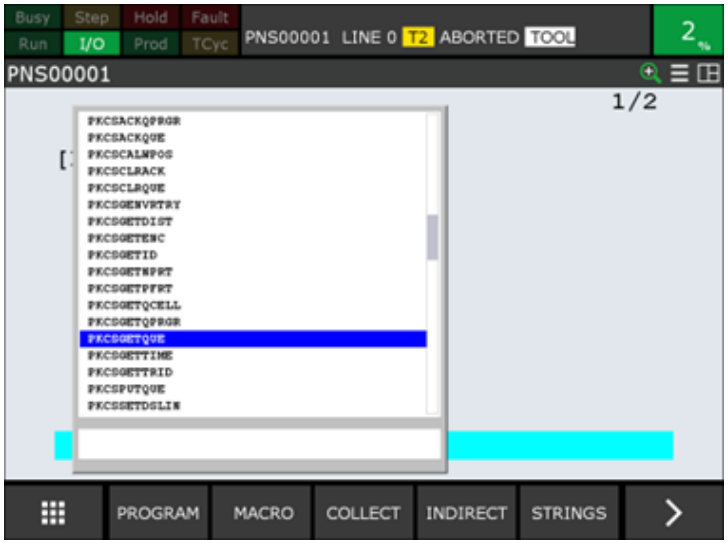
A KAREL program list screen will appear.



6 Select the KAREL program to call.

The KAREL program will be added on the program editing screen.

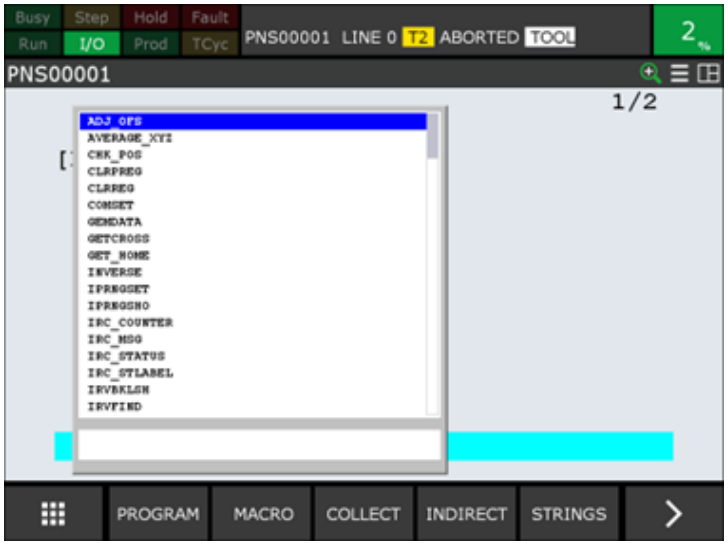
The following screen is an example of calling a KAREL program that starts with 'PK'.



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



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



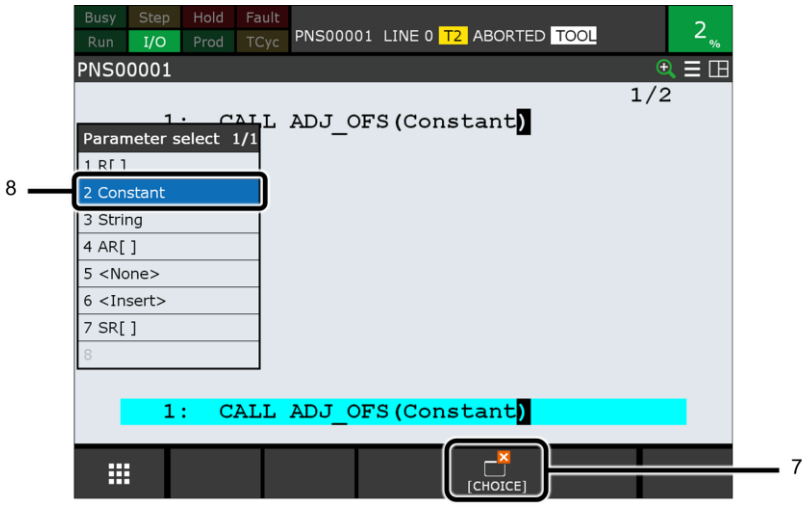
The following procedure is required only when you call a KAREL program other than one that starts with ‘PK’.
This procedure is explained using a call to ‘ADJ_OFS’ as an example.

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

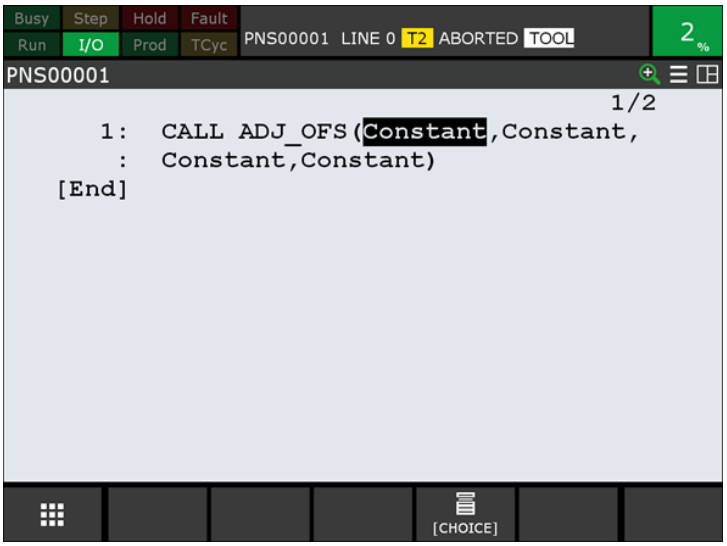


If you want to add an argument, move the cursor to the ')' on the right and press F4 [CHOICE].

8 Select [Constant] from the menu.



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



2.2.6. Setting payloads (motion performance)

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

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

The menu appears.

2 Select [SYSTEM] → [Motion] from the menu.



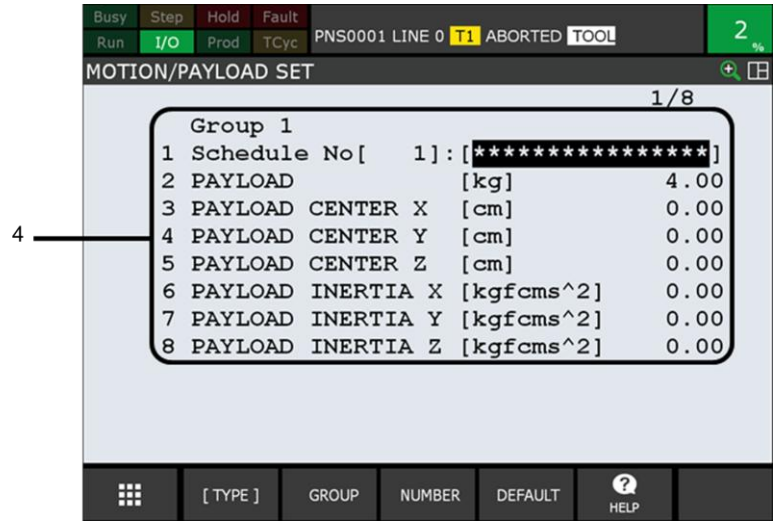
3 Move the cursor to [1] and press F3 [DETAIL].



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

4 Input the [PAYLOAD]. Not all of the items necessarily need to be input.

Like the sample programs (PT_PICKCS、PT_DROPCS) given above as programs for pick / place, if Set Payload [1] is used when the robot is not holding a part, set a payload for 'Hand only' in this step.



5 Press the [PREV] key on the teach pendant of the robot controller.

Return to the list on the motion performance screen.

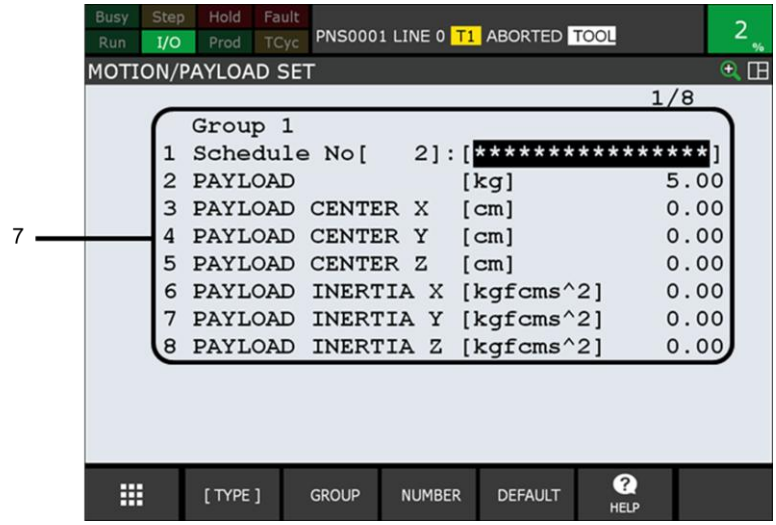
6 Move the cursor to [2] and press F3 [DETAIL].



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

7 Input the [PAYLOAD]. Not all of the items necessarily need to be input.

Like the sample programs (PT_PICKCS, PT_DROPCS) given above as programs for pick / place, if Set Payload [2] is used when the robot is holding a part, set a payload for 'Hand + part' in this step.



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

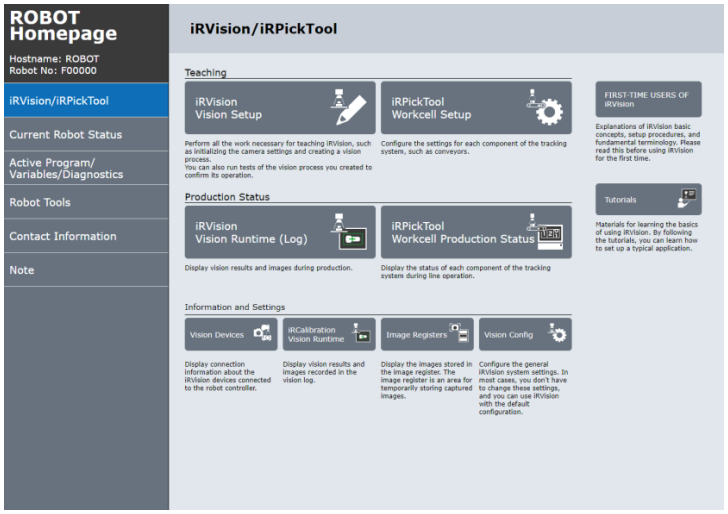
2.3. Setting up applications

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

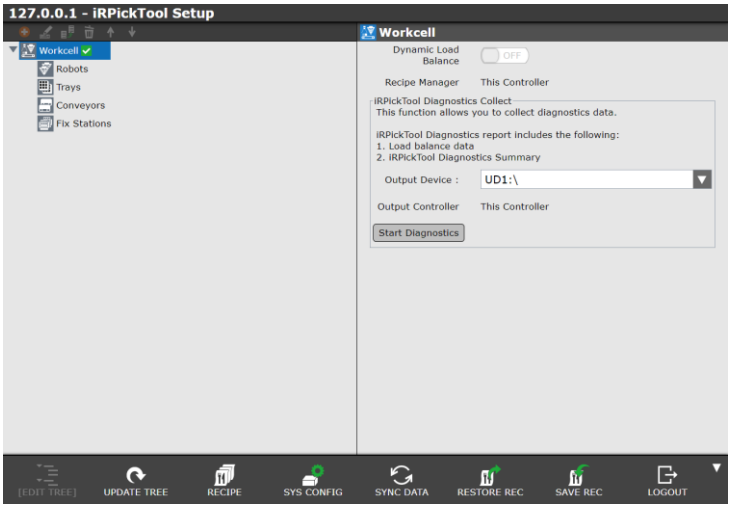
2.3.1. Setting up a workcell

Set up a workcell.

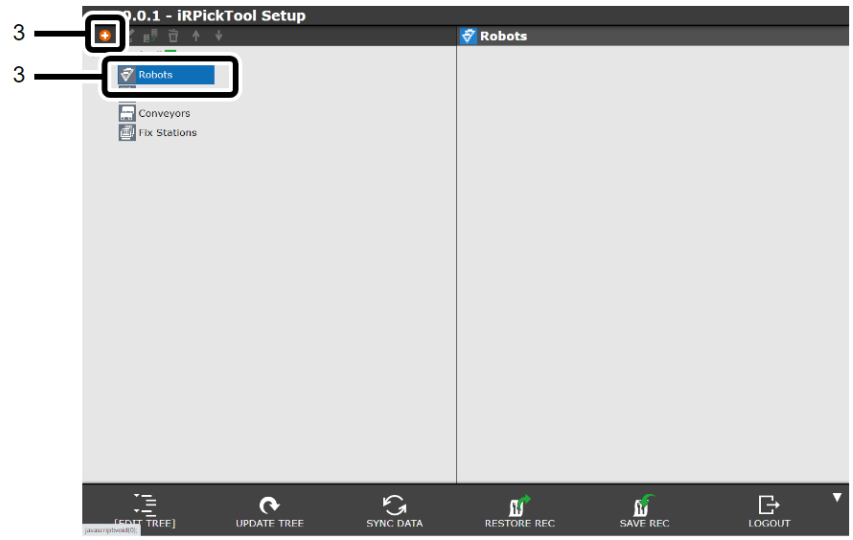
1. **Open the robot homepage.**
2. **Select [iRVision/iRPickTool] in the menu area and click [iRPickTool System Setup] in the content display area.**



The following iRPickTool setup screen will appear.



3. **Select [Robot], then click the [CREATE] button above the tree.**



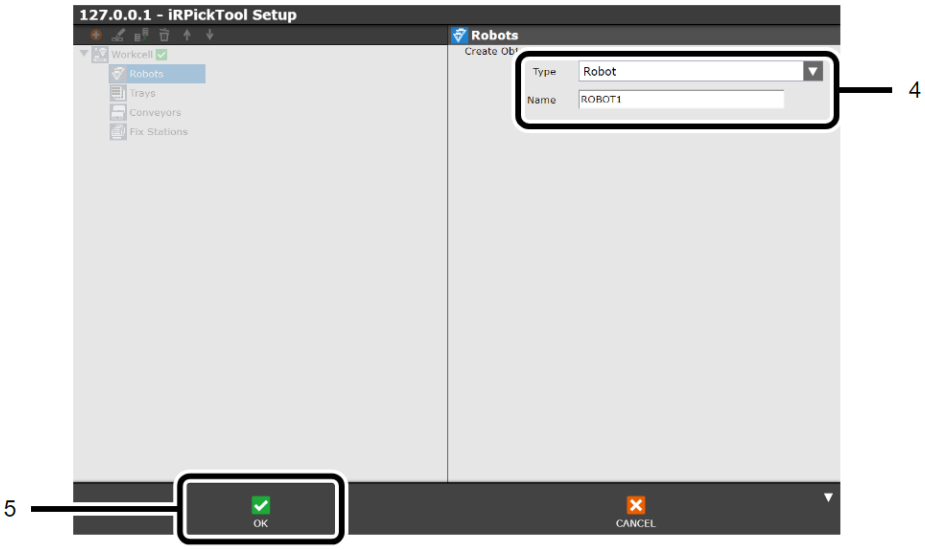
4 Input the name of the robot in [Name].

We recommend using the name that is displayed as the initial value just as it is.

Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

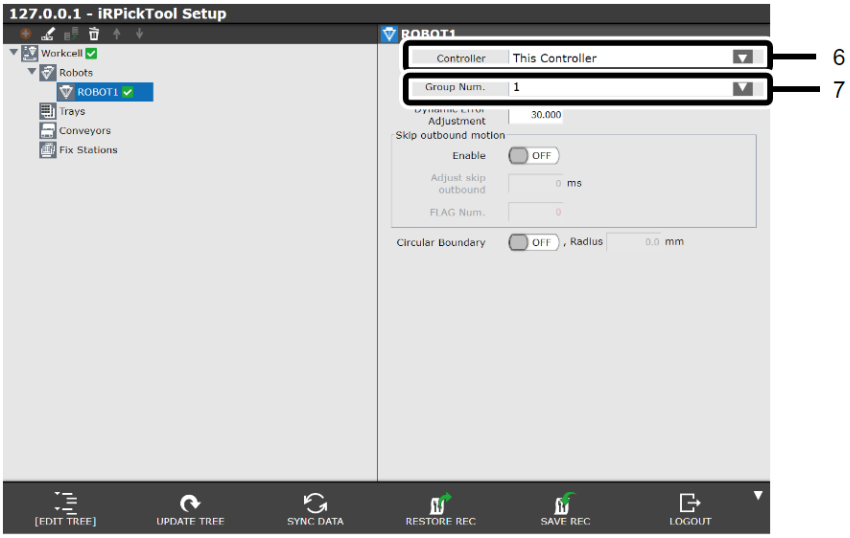
5 Click [OK].

The robot object is created.

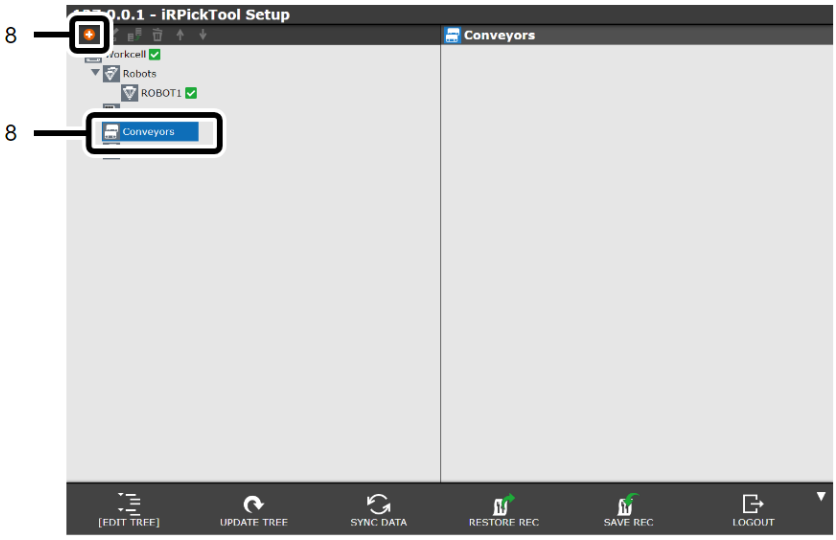


6 Select [This Controller] from the [Controller] drop-down list (if there is one robot).

7 Select [1] from the [Group Number] drop-down list.



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



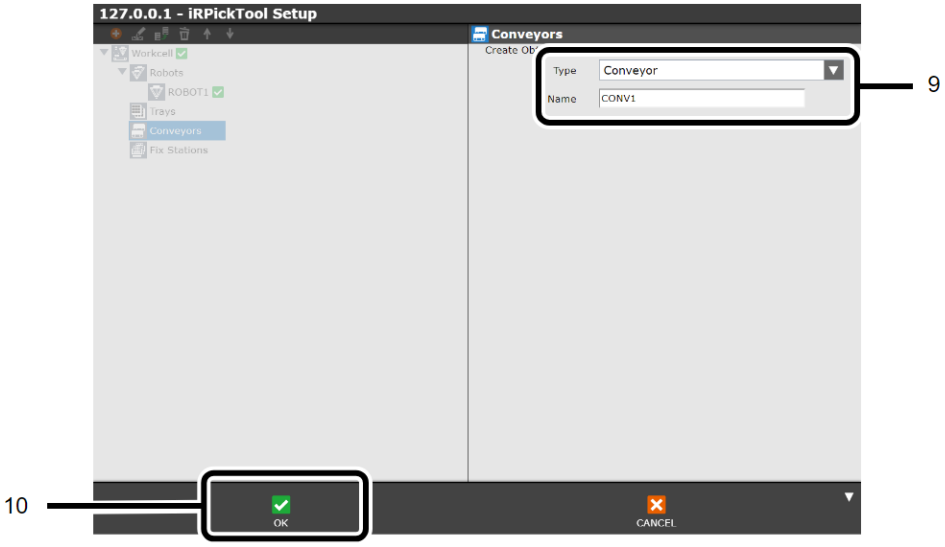
9 Input the name of the infeed conveyor in [Name].

We recommend using the name that is displayed as the initial value just as it is.

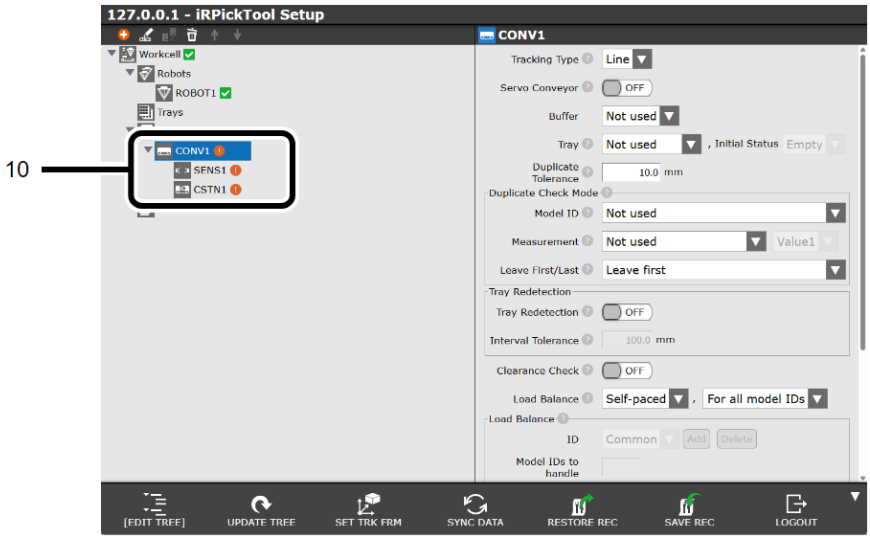
Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

10 Click [OK].

The robot object is created.



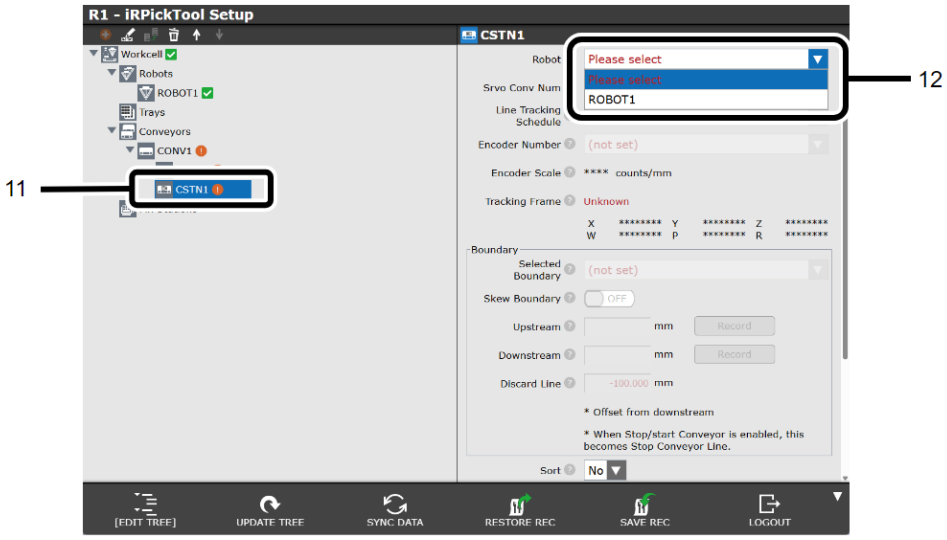
When a new conveyor is created, a sensor and conveyor station will be automatically created.



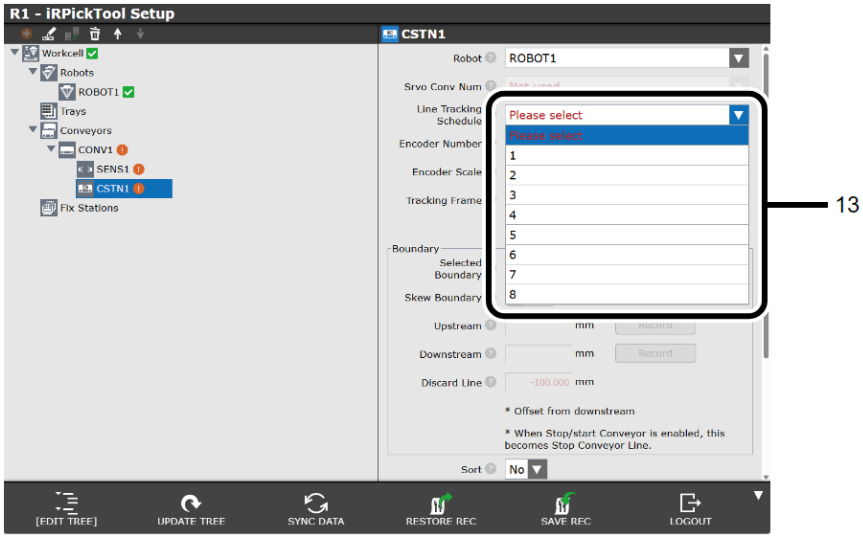
11 Select the conveyor station [CSTN1] that was automatically created in step 10.

12 Select a robot from the [Robot] drop-down list.

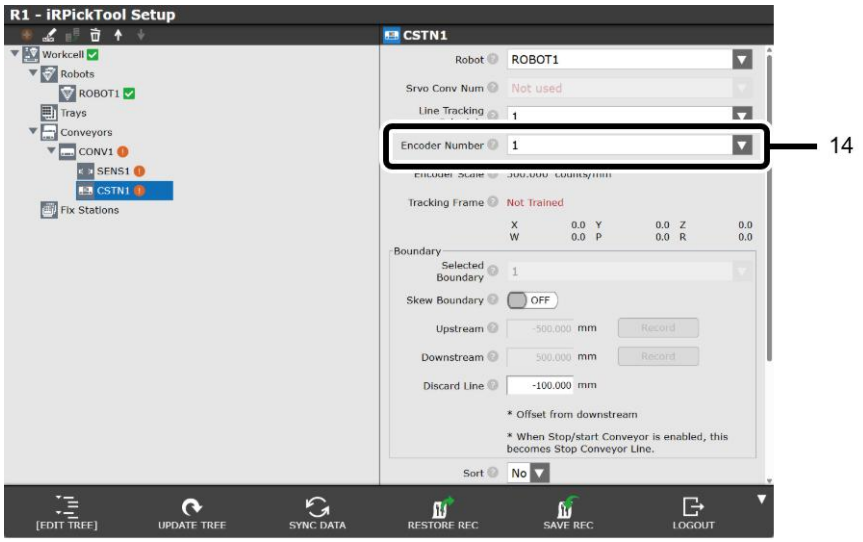
Select the robot name that was input in step 4.



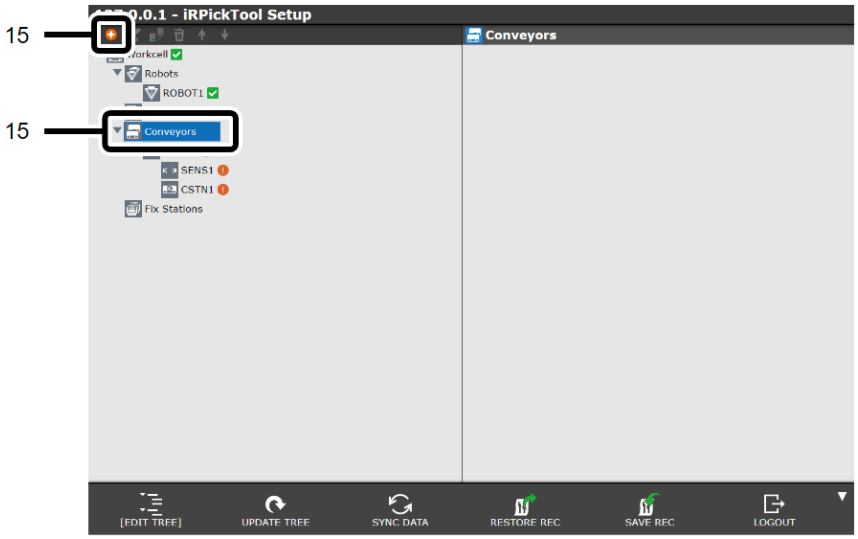
13 Select a number from the [Line Tracking Schedule] drop-down list. Select '1'.



14 **Select a number from the [Encoder Number] drop-down list. Select '1'.**



15 **Next, create an object for the outfeed conveyor. Select [Conveyor], then click the [CREATE] button above the tree.**



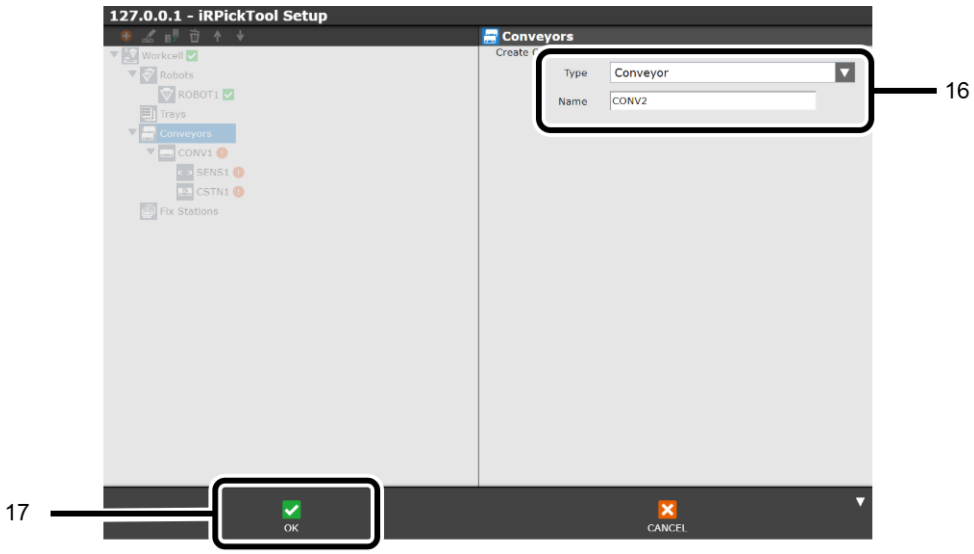
16 **Input the name of the outfeed conveyor in [Name].**

We recommend using the name that is displayed as the initial value just as it is.

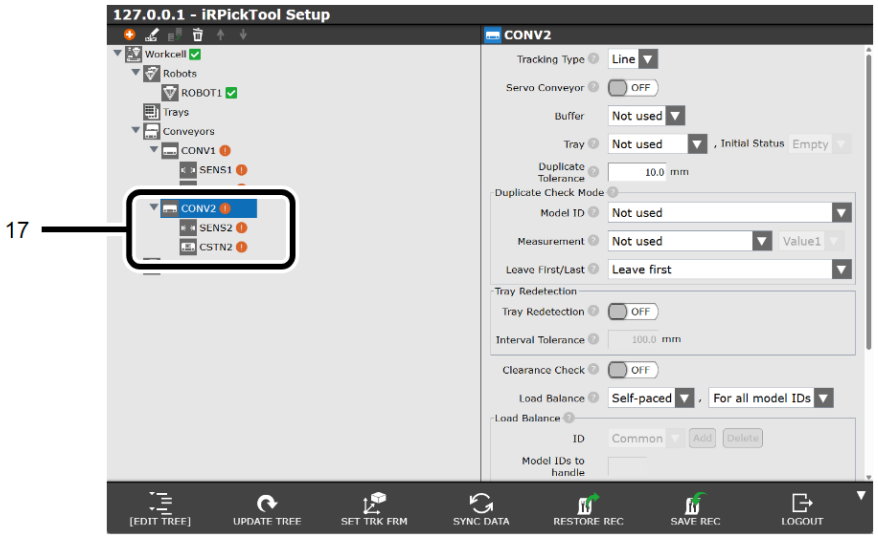
Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

17 Click [OK].

The robot object is created.



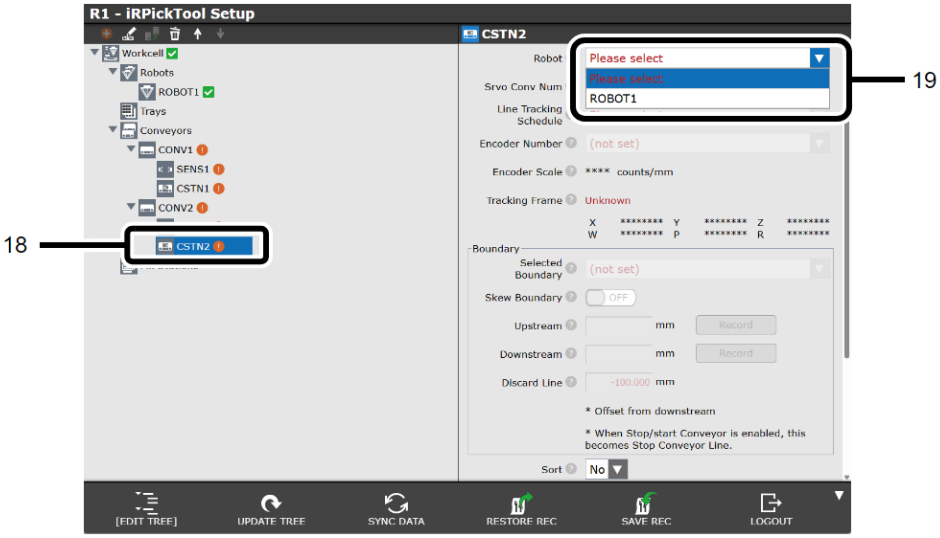
When a new conveyor is created, a sensor and conveyor station will be automatically created.



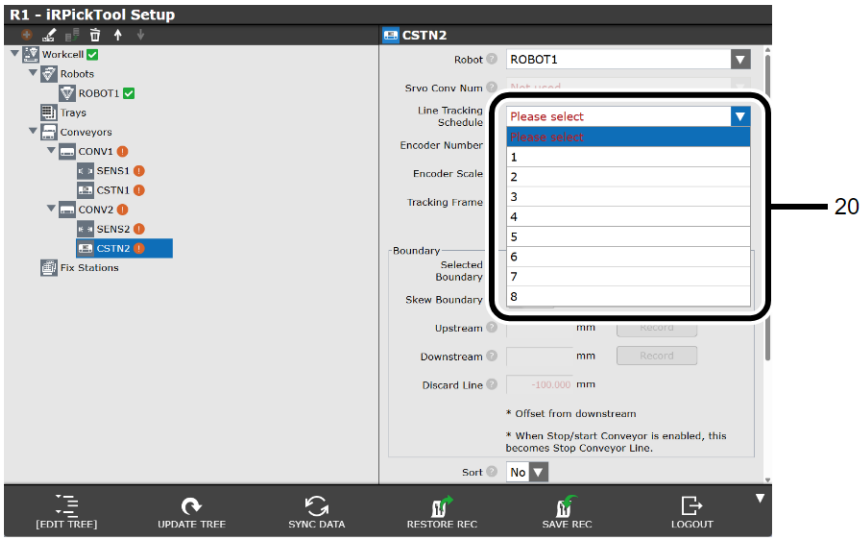
18 Select the conveyor station [CSTN2] that was automatically created in step 17.

19 Select a robot from the [Robot] drop-down list.

Select the robot name that was input in step 4.

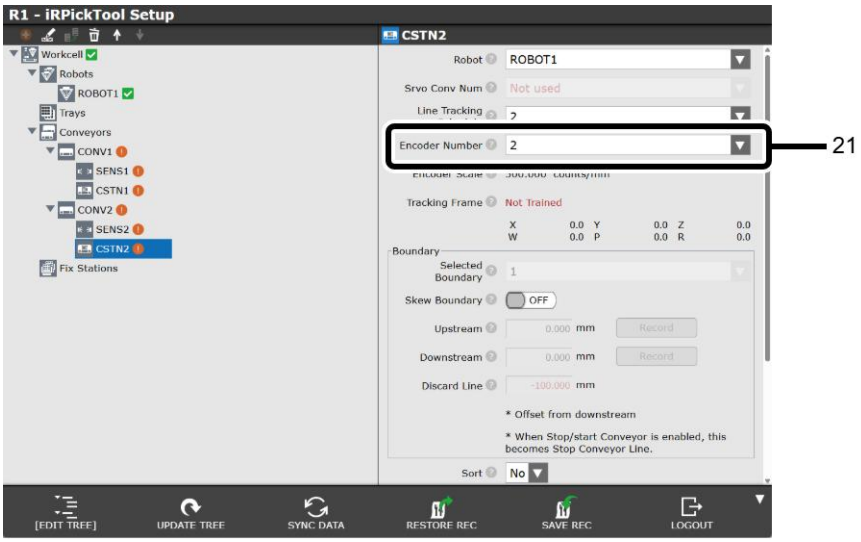


20 **Select a number from the [Line Tracking Schedule] drop-down list. Select ‘2’.**



21 **Select a number from the [Encoder Number] drop-down list.**

Select ‘2’.



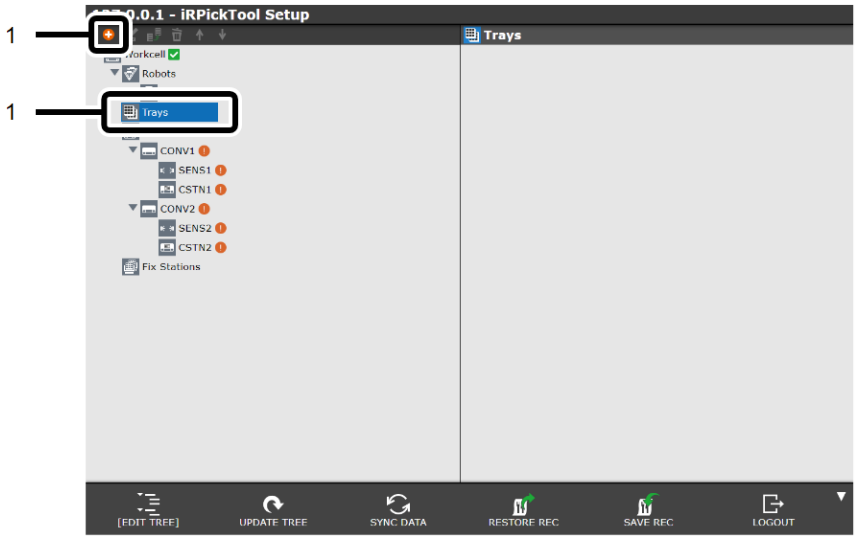
2.3.2. Setting up a tray

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

NOTE

When using trays on a conveyor, all cells within the same tray are sent to the downstream conveyor station when the most upstream cell crosses the Discard Line. Therefore, it is not possible to perform operations simultaneously at multiple conveyor stations on a long tray spanning several stations.

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



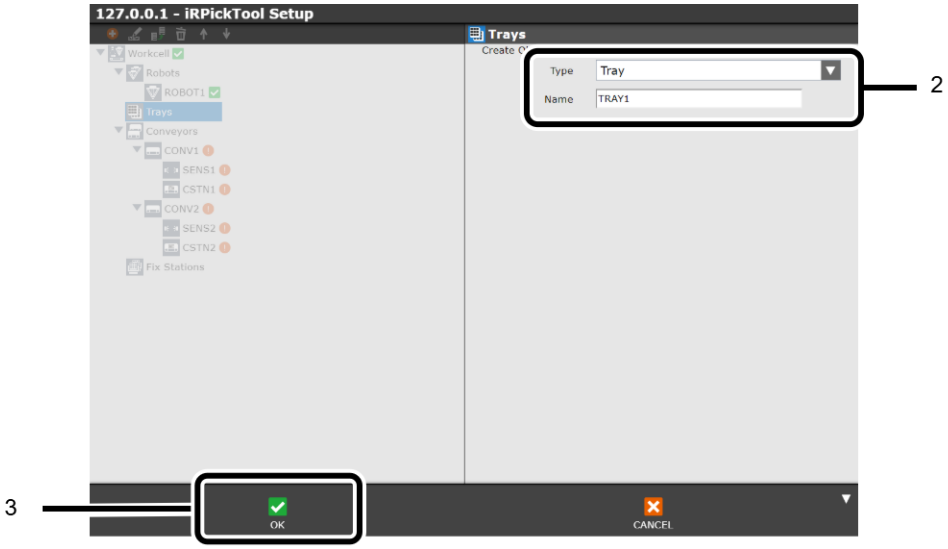
2 Enter the name of the tray in [Name].

We recommend using the name that is displayed as the initial value just as it is.

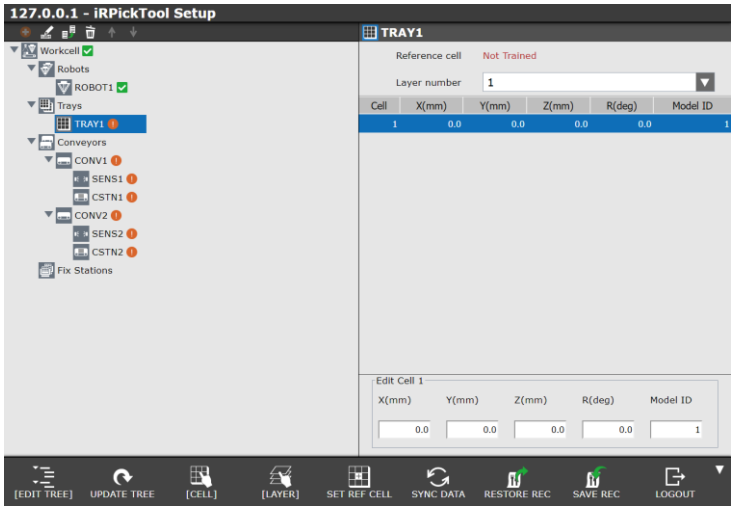
Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

3 Click [OK].

The robot object is created.



After a while, the following screen appears.



4 Decide on a tray frame.

The origin can be at any position, such as one corner of the box.

Aligning the flow direction of the conveyor with the X direction of the tray frame will make things easier to understand. (Other directions are also OK.)

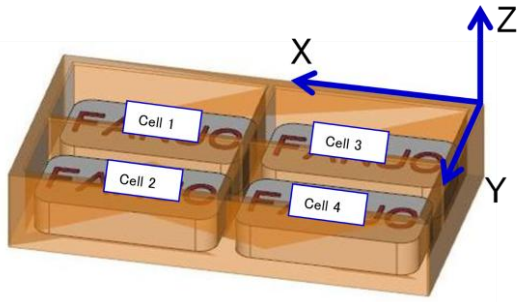
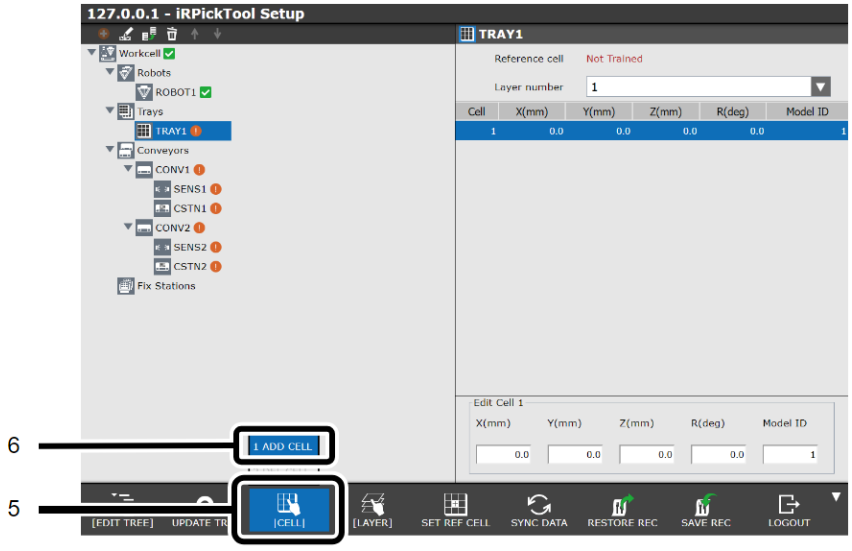


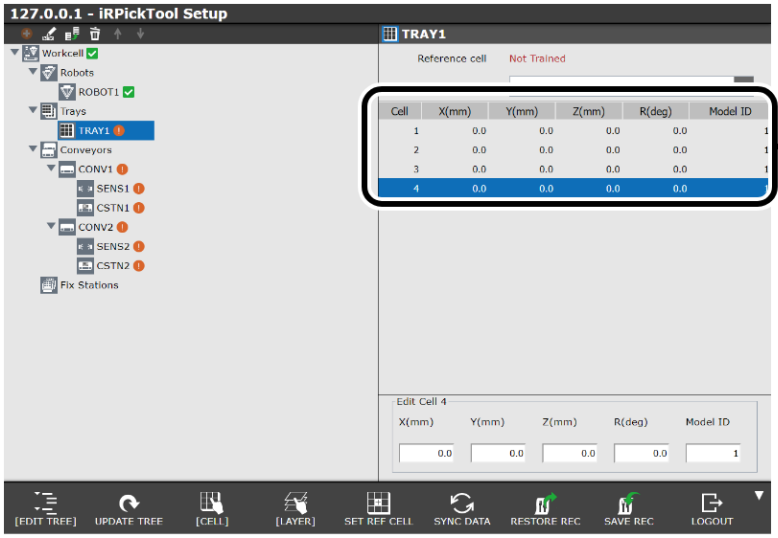
Fig. 2.3.2-1 Deciding on the origin of a tray

5 Click [CELL].

6 Select [ADD CELL] from the menu.

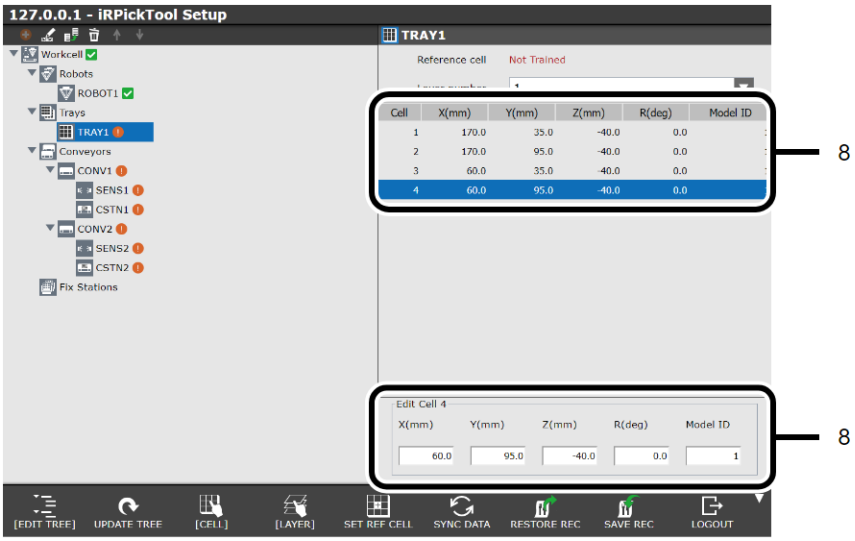


7 Add the cells for the first stage.



8 Enter the coordinates of each cell on the cell edit screen.

Select the row of the cell that you want to edit, and enter the coordinates in the frame at the bottom of the screen. Enter the coordinates of a cell in the tray frame coordinate system. For details, see "I.5.3. Setting up a tray(P.269)".



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

9 Click [LAYER].

10 Select [ADD LAYER] from the menu.



11 **Enter the number of cells in the second stage in [Number of cells].**

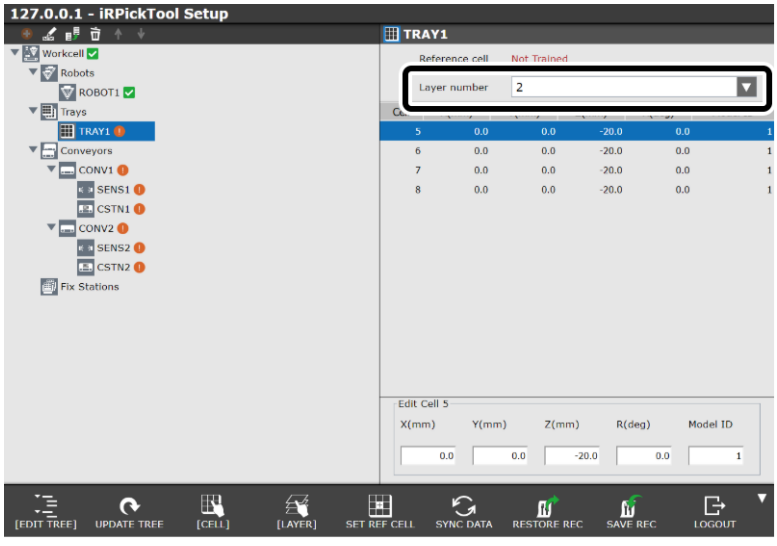
12 **Enter a Z value (mm) in [Default Z(mm)].**

13 **Click [OK].**



After a while, the following screen appears.

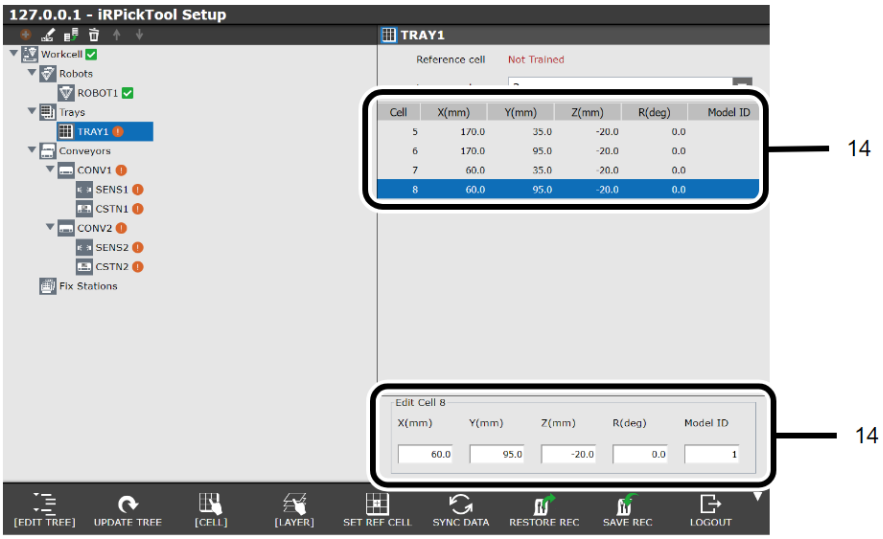
Layer '2' is created in [Layer number].



14 Enter the coordinates of each cell in Layer 2 on the cell edit screen.

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

The reference cell can remain untrained at this point. It is taught automatically in "I.2.3.10. Setting up a reference position and teaching a robot position(P.113)".



2.3.3. Setting up a conveyor

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

2.3.3.1. Setting up an infeed conveyor

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

1 Select conveyor [CONV1].



The conveyor setup screen will appear.

To set up a conveyor, the following data that appears as the initial values can be used as it is.

[Tracking Type]:	Line
[Servo Conveyor]:	[Disable]
[Tray]:	Not used

[Duplicate Tolerance]: 10.0 mm
[Load Balance]: Self-paced, For all model IDs

2.3.3.2. Setting up an outfeed conveyor

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

1 Select conveyor [CONV2].



The conveyor setup screen will appear.

2 Select [TRAY1] for [Tray] and [Empty] for [Initial Status].

For other settings for conveyors, the following data that appears as the initial values can be used as it is.

[Tracking Type]: Line
[Servo Conveyor]: [Disable]
[Duplicate Tolerance]: 10.0 mm (default)
[Load Balance]: Self-paced, For all model IDs

2.3.4. Setting a tracking frame

Set up a tracking frame.

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

The menu appears.

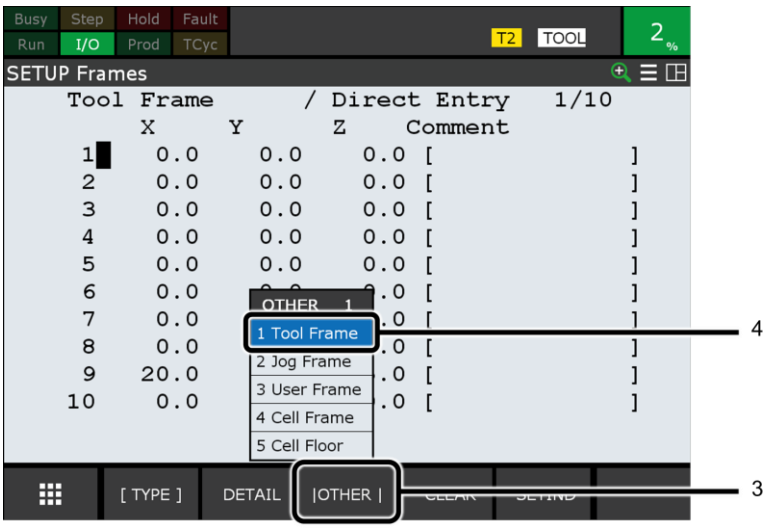
2 Select [SETUP] → [Frames] from the menu.



3 Press F3 [OTHER].

4 Select [Tool Frame] from the menu.

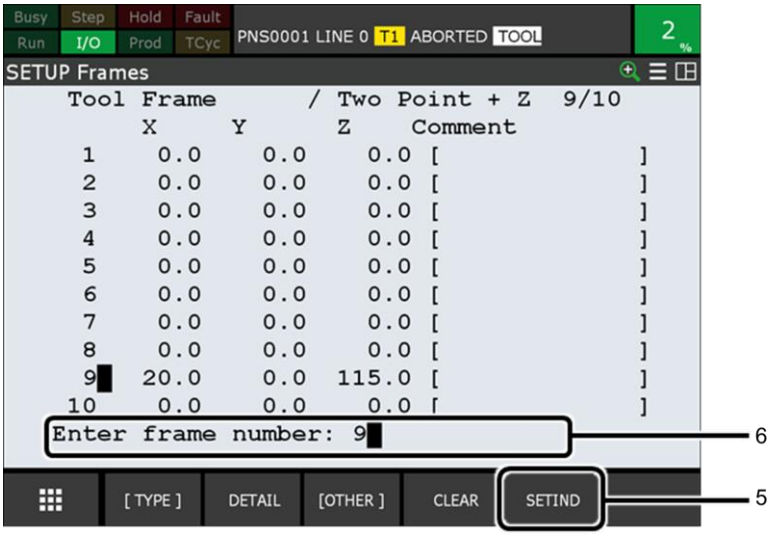
The Tool Frame list screen appears.



5 Press F5 [SETIND].

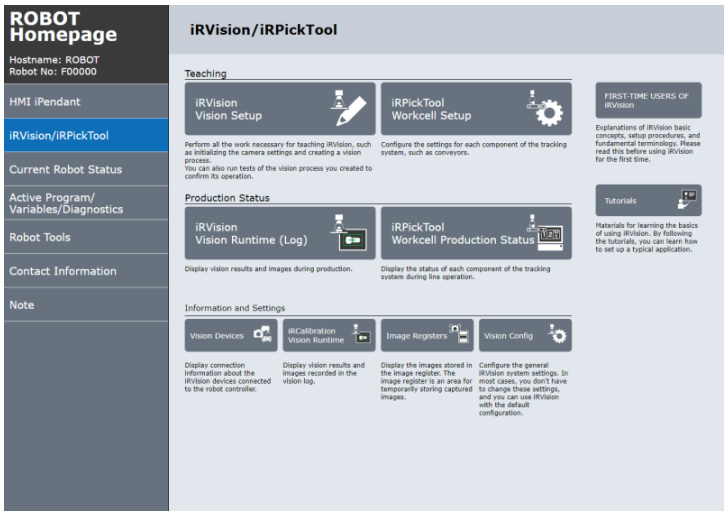
6 Enter the frame number that has been used for the pointer tool.

This procedure is explained using Tool 9 as an example.



7 **Open the robot homepage.**

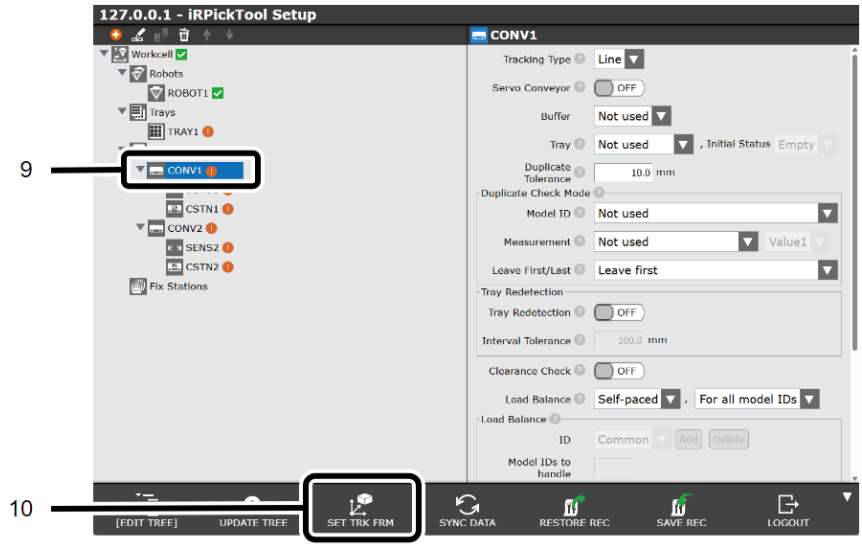
8 **Select [iRVision/iRPickTool] in the menu area and click [iRPickTool System Setup] in the content display area.**



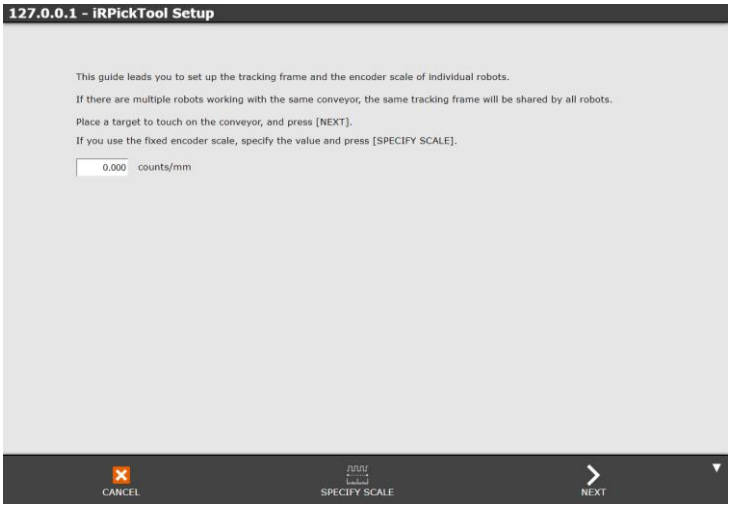
9 **First, set up the tracking frame of the infeed conveyor.**

Select conveyor [CONV1].

10 **Click [SET TRK FRM].**



After a while, the following screen appears.



⚠CAUTION

When using the αiA4000S, “IRPK-063 Encoder count does not match on all robots” may occur even when the conveyor is stationary, due to high-resolution encoder pulses. In this case, stop the conveyor and retry the operation several times.

11 Set a calibration grid on the upstream side of the conveyor, in accordance with the procedure shown on the screen.

Fix it with tape to prevent it from moving.

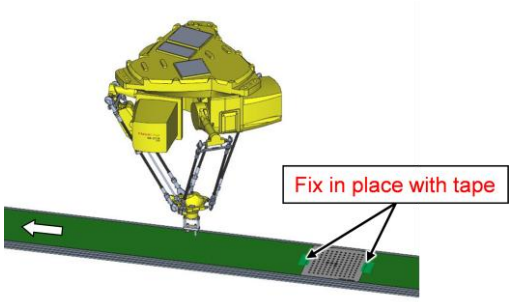


Fig. 2.3.4-1 Setting a calibration grid on the upstream side

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

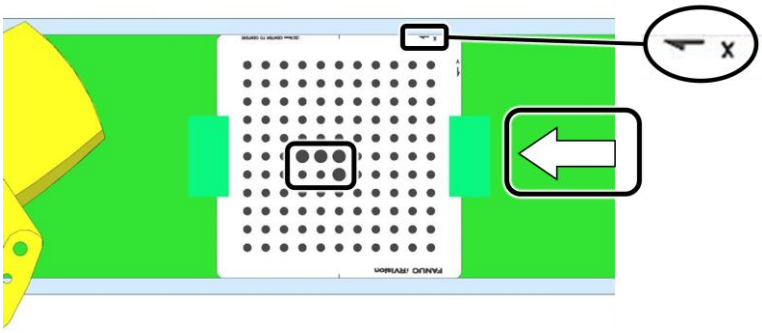
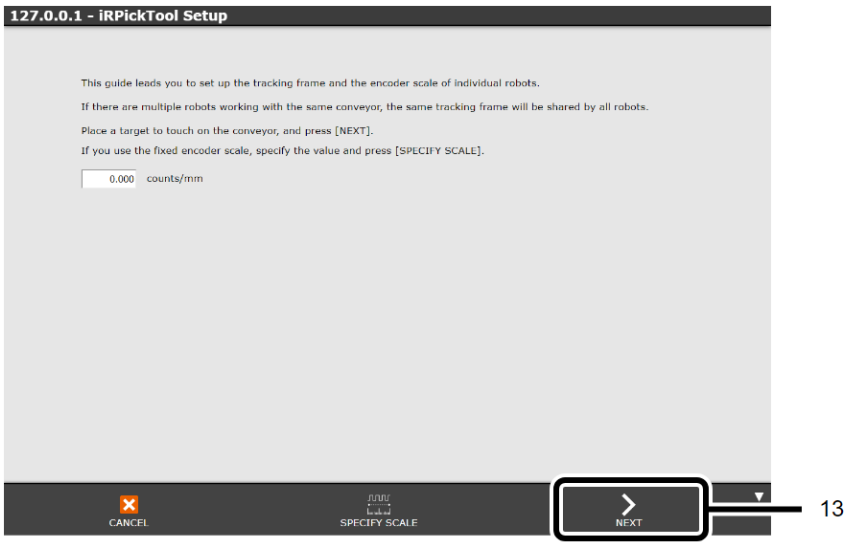
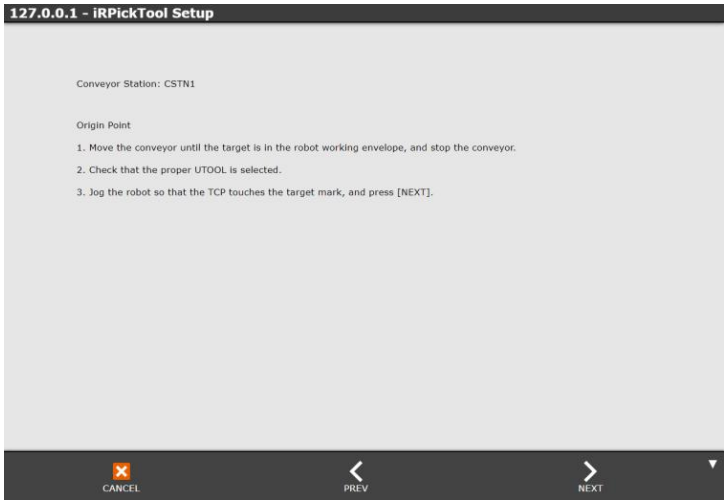


Fig. 2.3.4-2 Align the setting direction

13 Click [NEXT].



After a while, the following screen appears.



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

15 **Touch up the origin of the calibration grid.**

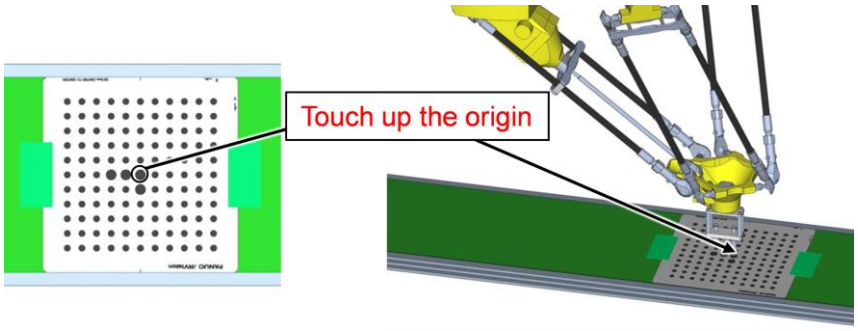
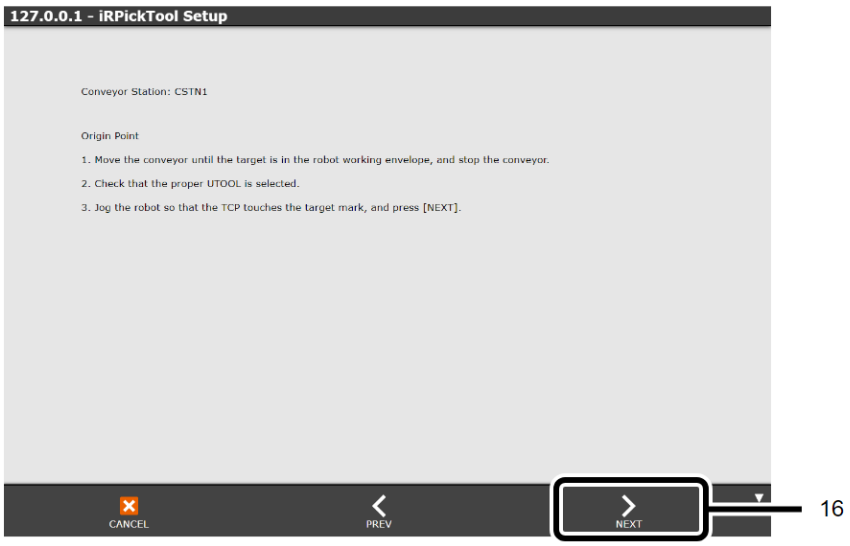
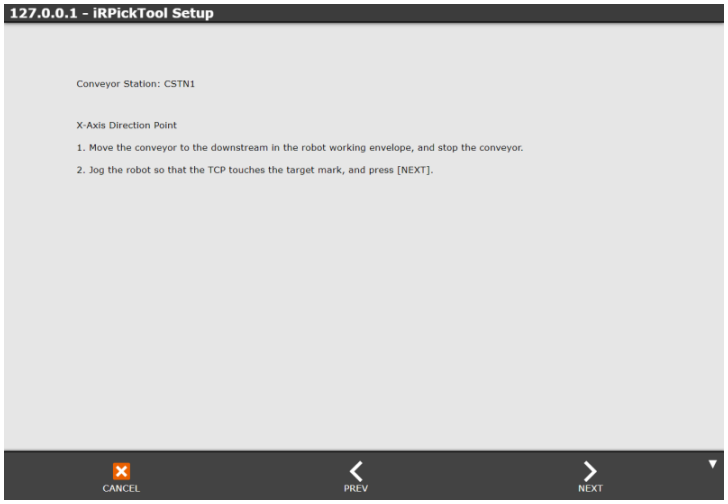


Fig. 2.3.4-3 Touching up the origin of the calibration grid

16 **Click [NEXT].**



After a while, the following screen appears.



17 **Move the robot up until it is above the surface of the conveyor.**

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

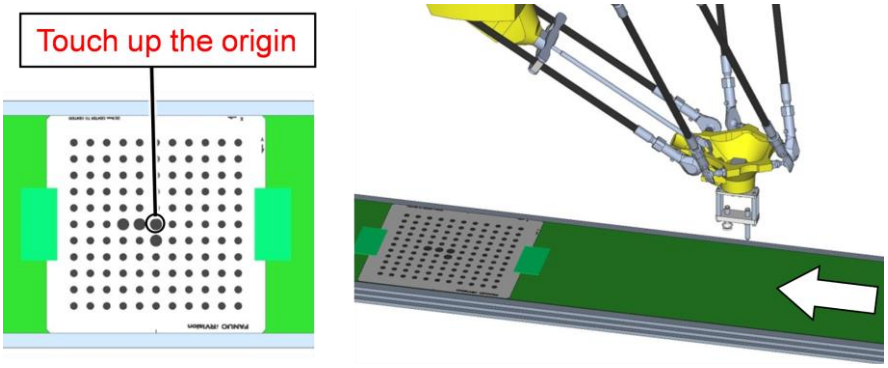


Fig. 2.3.4-4 Stopping the conveyor

19 **Touch up the origin of the calibration grid.**

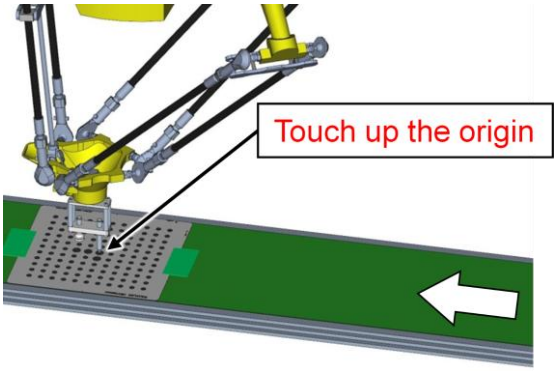
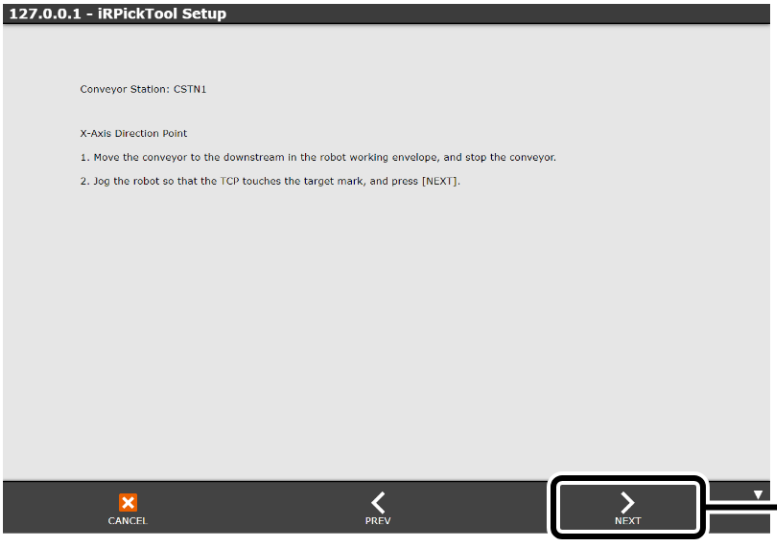
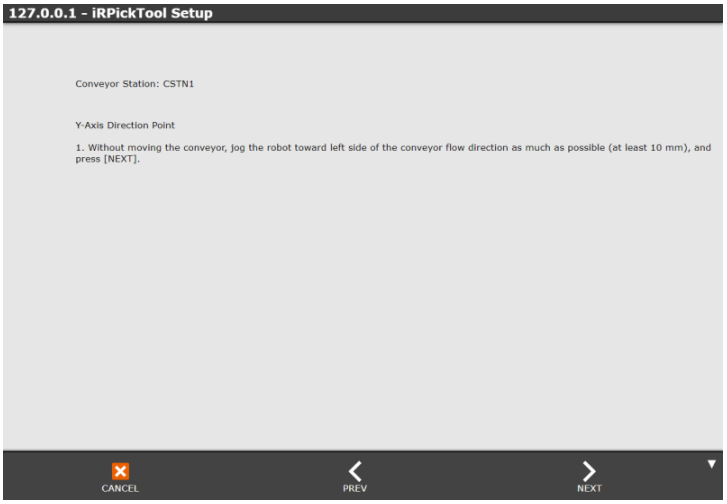


Fig. 2.3.4-5 Touching up the origin of the calibration grid

20 **Click [NEXT].**



After a while, the following screen appears.



21 Touch up the Y direction of the calibration grid.

Do not move the conveyor at this time.

This is not the Y direction of the robot. It is on the left side relative to the conveyor's movement direction.

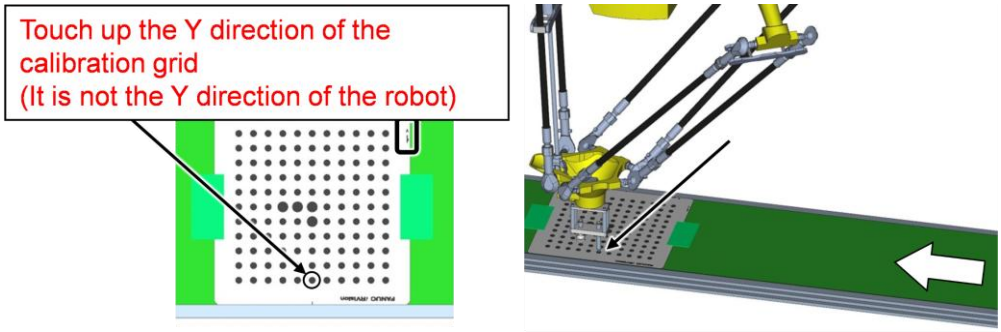
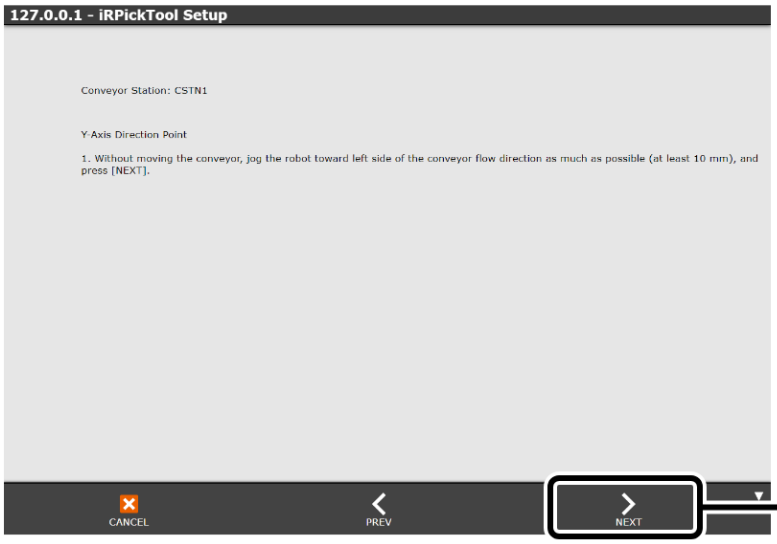
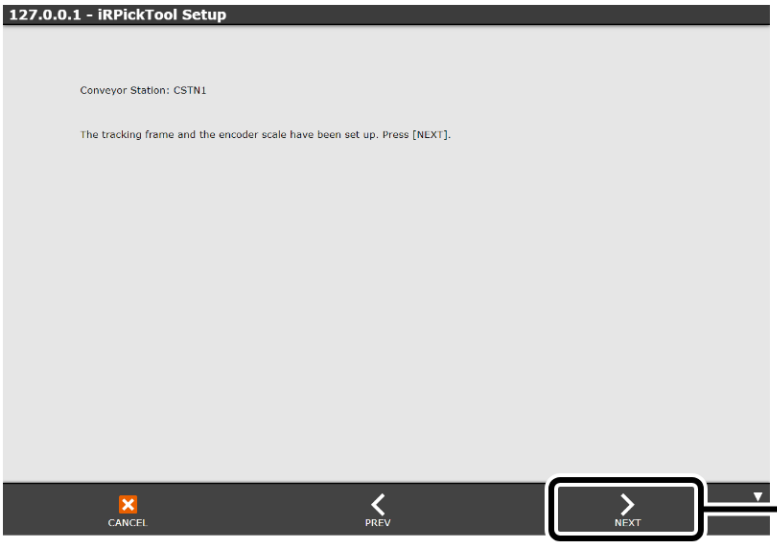


Fig. 2.3.4-6 Touching up the Y direction of the calibration grid.

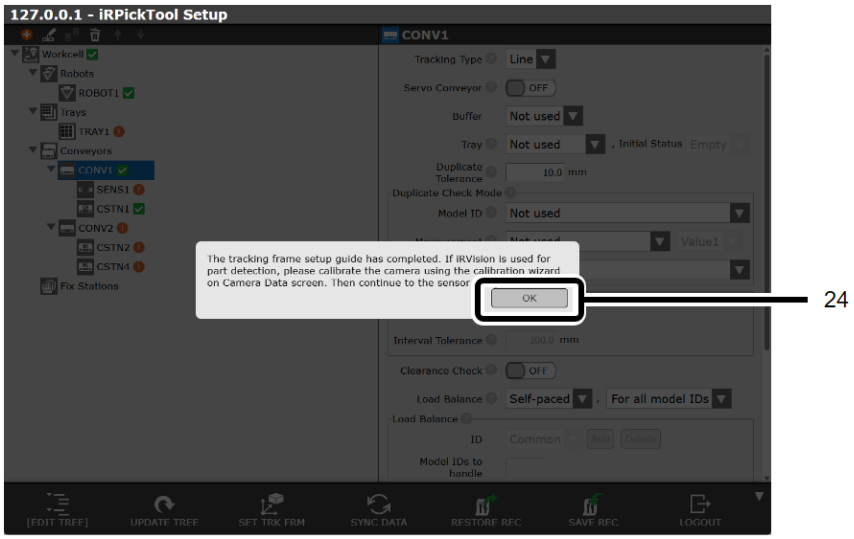
22 Click [NEXT].



23 **Click [NEXT].**



24 **Click [OK].**

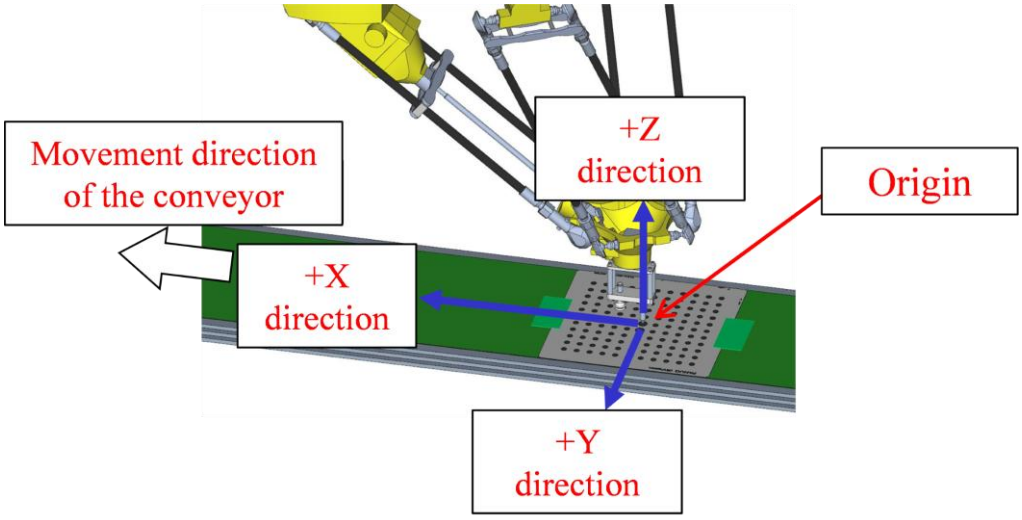


The tracking frame of the infeed conveyor has been set up.

The tracking frame of the infeed conveyor is set as follows.

Origin : touch-up point in step 15

+X direction : the movement direction of the conveyor
+Y direction : left side relative to the movement direction of the conveyor
+Z direction : an upward relative to the conveyor surface



25 Next, set up the tracking frame of the outfeed conveyor.

Select conveyor [CONV2].

26 Repeat steps 10 to 24 to set up the tracking frame of the outfeed conveyor.

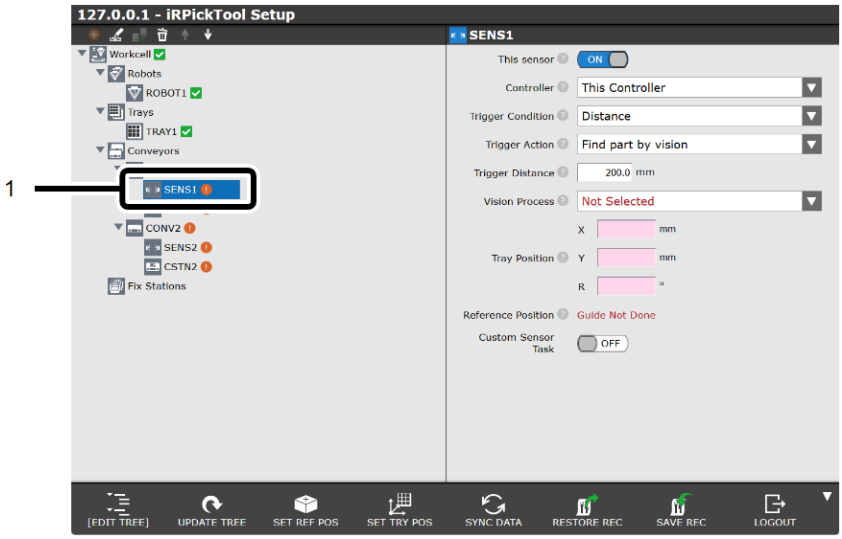
2.3.5. Setting up a sensor

Set up a sensor.

2.3.5.1. Setting an infeed conveyor sensor

Set up an infeed conveyor sensor. These are the settings for [Distance] + [Find part by vision].

1 Select the sensor [SENS1].



2 Set the following items.

[This Sensor]: [Enable]
[Controller]: [This Controller]
[Trigger Condition]: [Distance]

[Action]: [Find part by vision]

When [Trigger Condition] and [Action] are changed, setting items change accordingly.

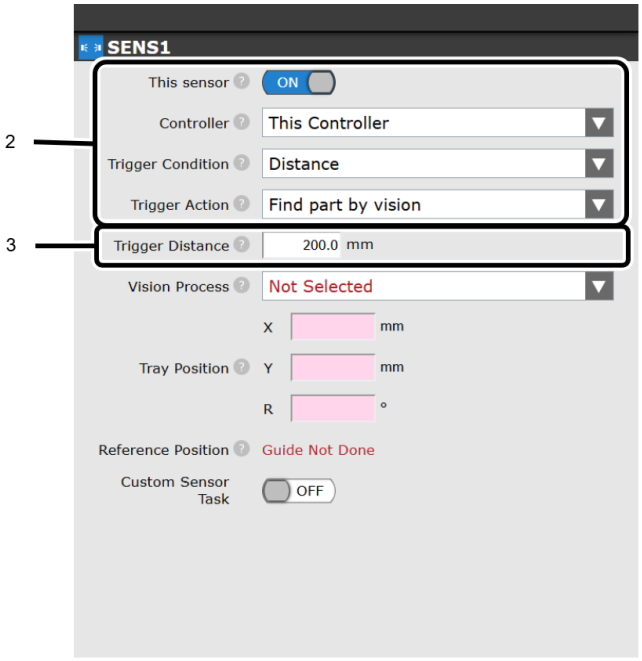
3 Set the following items.

[Trigger Distance]: Set how many mm the conveyor advances each time for vision detection to be performed.

It is recommended that the trigger distance be set such that a workpiece can be detected twice in the camera FOV.

[Vision Process]: You do not have to set this at this stage.

* Set this up immediately before setting up reference positions. For the settings, see "I.2.3.10.1. Setting up a reference position for the infeed conveyor and teaching a robot position(P.113)".



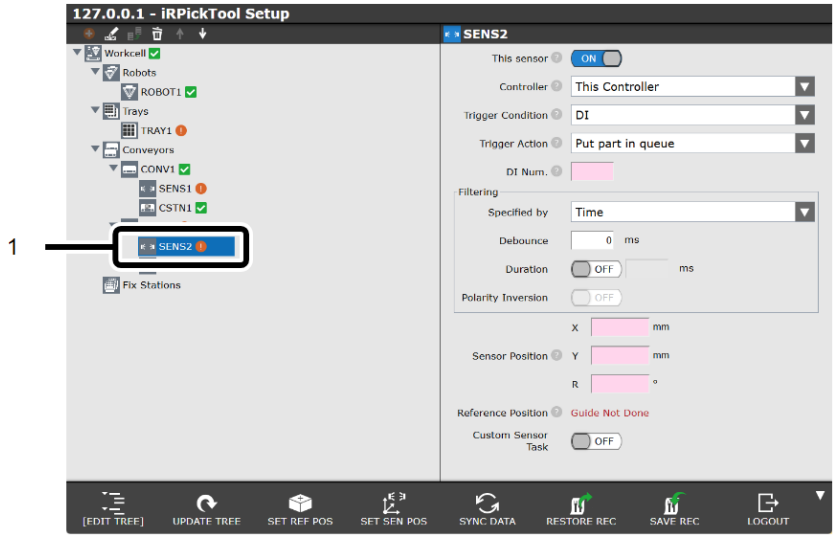
2.3.5.2. Setting up an outfeed conveyor sensor (In the case of [DI] / [RI] + [Put part in queue])

Set up an outfeed conveyor sensor. The following explanation uses the sensor settings for [DI] + [Put part in queue] as an example.

For [RI] + [Put part in queue], interpret "DI" as "RI" in the explanation.

In the case of [HDI] + [Put part in queue], skip this section and go to the next section, "I.2.3.5.3. Setting up an outfeed conveyor sensor (In the case of [HDI] + [Put part in queue])(P.83)".

1 Select the sensor [SENS2].



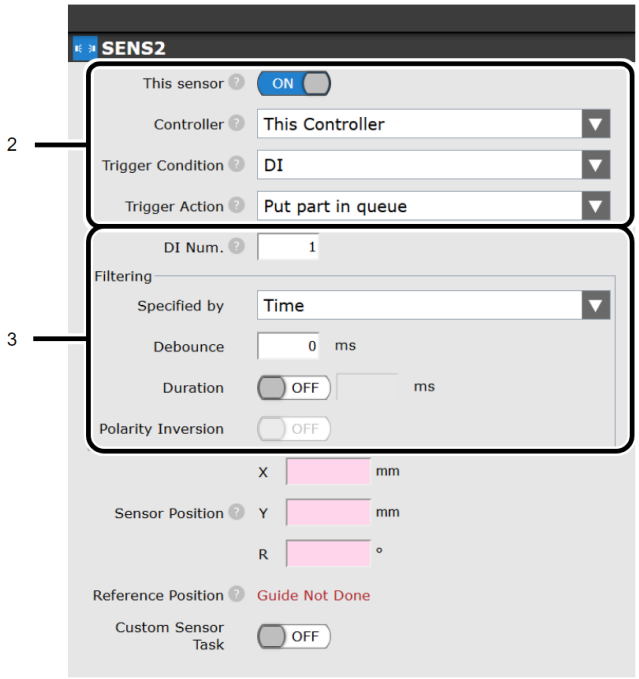
2 Set the following items.

[This Sensor]: [ON]
[Controller]: [This Controller]
[Trigger Condition]: [DI]
[Trigger Action]: [Put part in queue]
When [Trigger Condition] and [Action] are changed, setting items change accordingly.

3 Set the following items.

[DI Num]: Enter the DI number that has been assigned to the sensor input.
[Specified By]: Select the unit for specifying Debounce and Duration.
[Debounce]: Enter the time you want to disable input for (to prevent chattering).
[Duration]: Enable this setting to remove incidental noise, and enter a value longer than the duration or distance over which the noise occurs (for details, refer to "I.8.14.3. Duration(P.676) ").
[Polarity Inversion]: Enable this setting to use polarity inversion (for details, refer to "I.8.14.4. Polarity Inversion(P.677) ").
[Sensor Position]: You do not have to set this at this stage.

* This is set in the setup of the sensor position. For the settings, see "I.2.3.6. Setting the sensor position(P.84)".

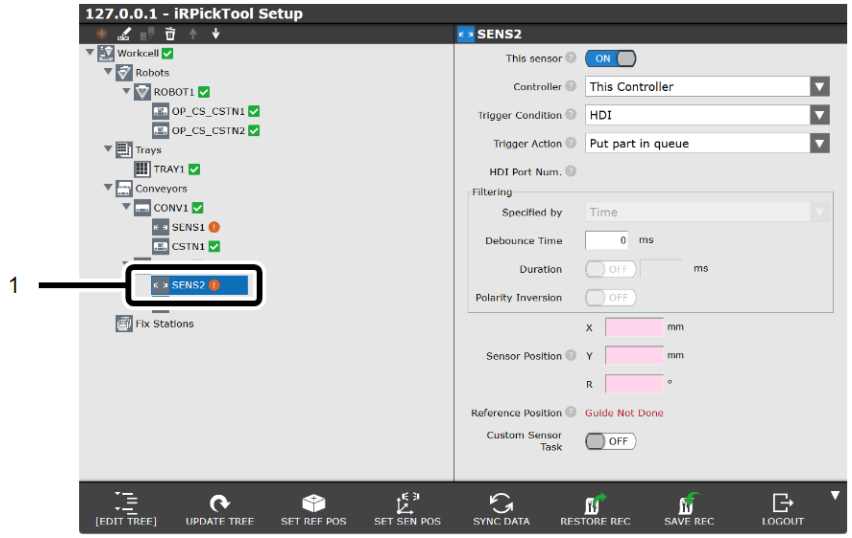


Skip the next section, “I.2.3.5.3. Setting up an outfeed conveyor sensor (In the case of [HDI] + [Put part in queue])(P.83)” and go to “I.2.3.6. Setting the sensor position(P.84)”.

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

Set up an outfeed conveyor sensor. These settings are for a sensor in the case of [HDI] + [Put part in queue]. In the case of [DI] / [RI] + [Put part in queue], see the previous section, “I.2.3.5.2. Setting up an outfeed conveyor sensor (In the case of [DI] / [RI] + [Put part in queue])(P.81)”, not this section.

1 Select the sensor [SENS2].



2 Set the following items.

[This Sensor]: [ON]
[Controller]: [This Controller]
[Trigger Condition]: [HDI]
[Trigger Action]: [Put part in queue]
When [Trigger Condition] and [Action] are changed, setting items change accordingly.

3 Set the following items.

[HDI Port Num.]: The number that was specified in the encoder setup will appear.
[Debounce Time]: Enter the time you want to disable input for (to prevent chattering).
[Sensor Position]: You do not have to set this at this time.

* This is set in the setup of the sensor position. For the settings, see “I.2.3.6. Setting the sensor position(P.84)”.



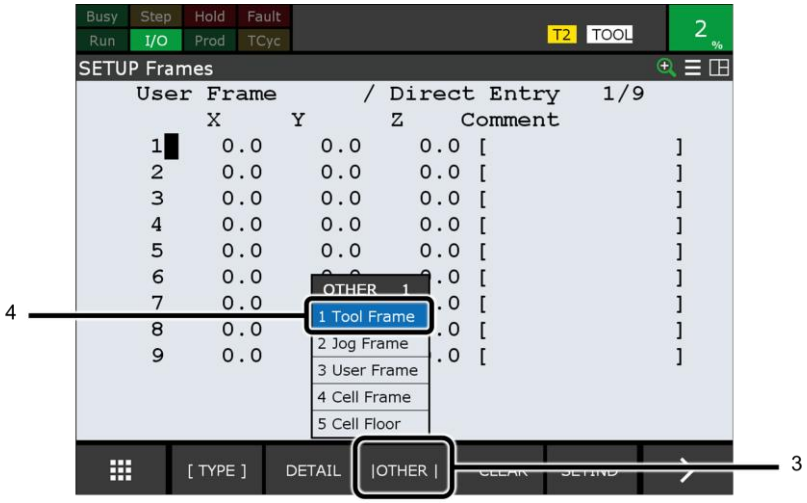
2.3.6. Setting the sensor position

Set up the sensor position of the outfeed conveyor. This setup will use trays.

1. **Press the [MENU] key on the robot controller's teach pendant.**
   The menu appears.
2. **From the menu, select [SETUP] → [Frames].**



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

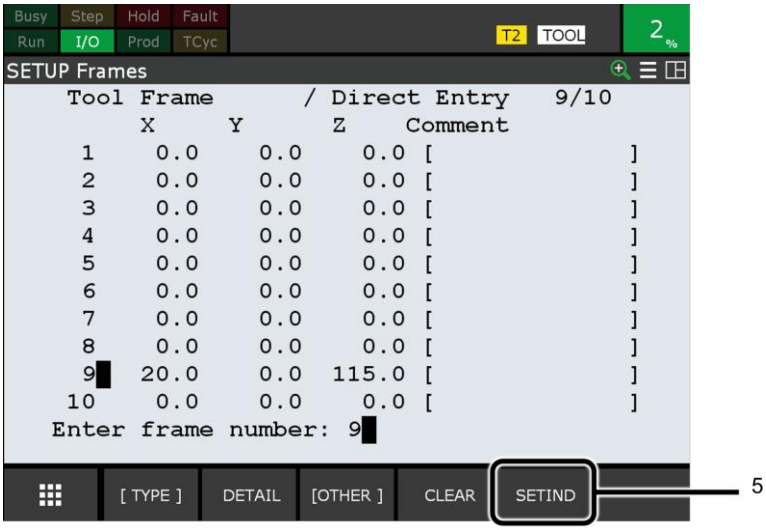


The Tool Frame list screen appears.

5 Press F5 [SETIND].

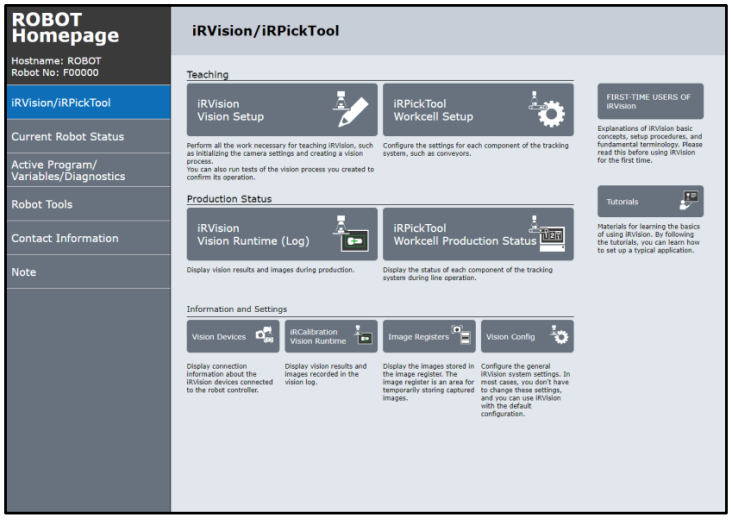
6 Enter the frame number that has been used for the pointer tool.

This procedure is explained using Tool 9 as an example.



7 Open the robot homepage.

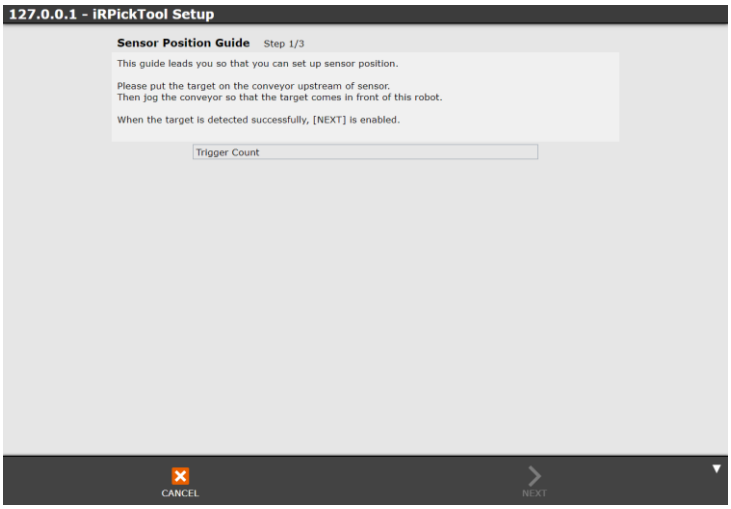
8 Select [iRVision/iRPickTool] in the menu area and click [iRPickTool Workcell Setup] in the content display area.



9 Select [SENS2] in [Conveyor], then click [SET SEN POS].



After a while, the following screen appears.



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

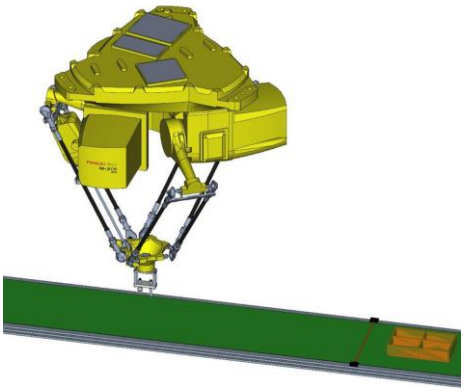


Fig. 2.3.6-1 Setting a tray upstream from the sensor

11 Move the conveyor. When the tray comes within the operating range of the robot, stop the conveyor.

The trigger count at the instant the sensor found a part will appear

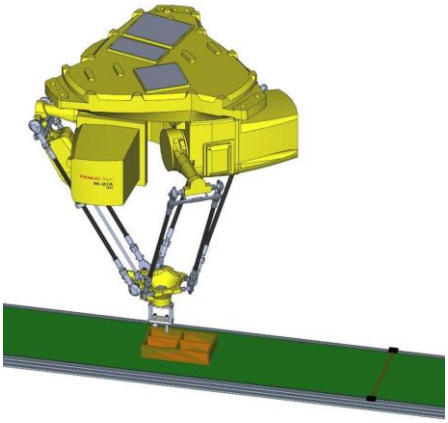
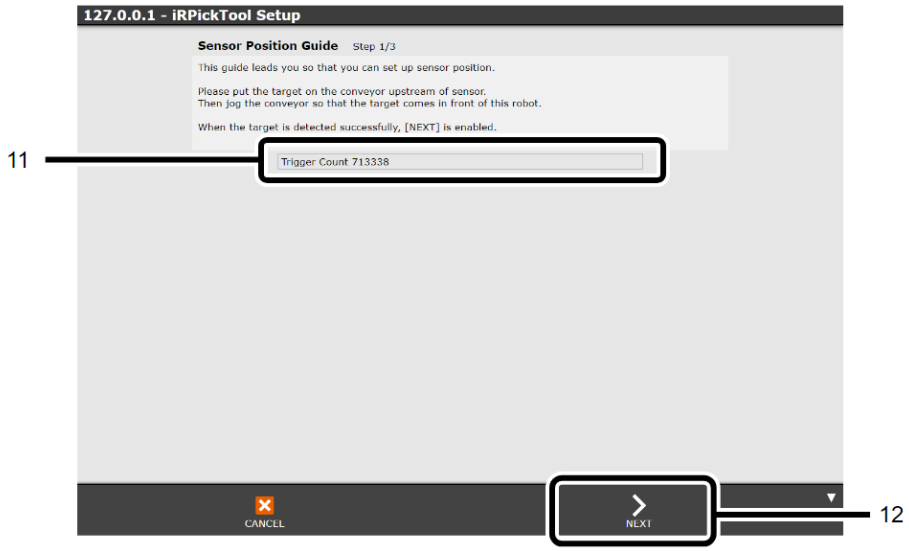
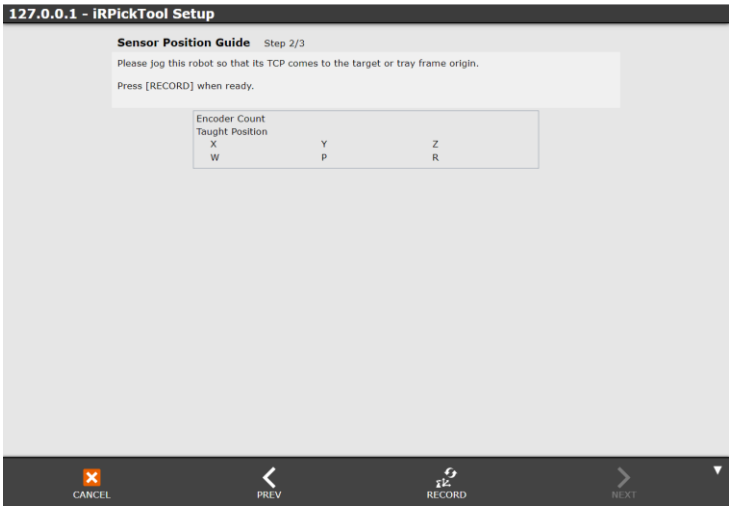


Fig. 2.3.6-2 Stopping when the tray comes within the operating range of the robot

12 Click [NEXT].



After a while, the following screen appears.



13 Touch up the origin of the tray.

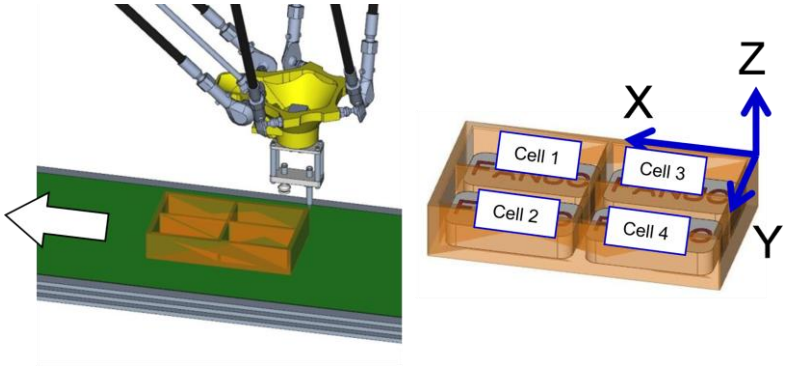


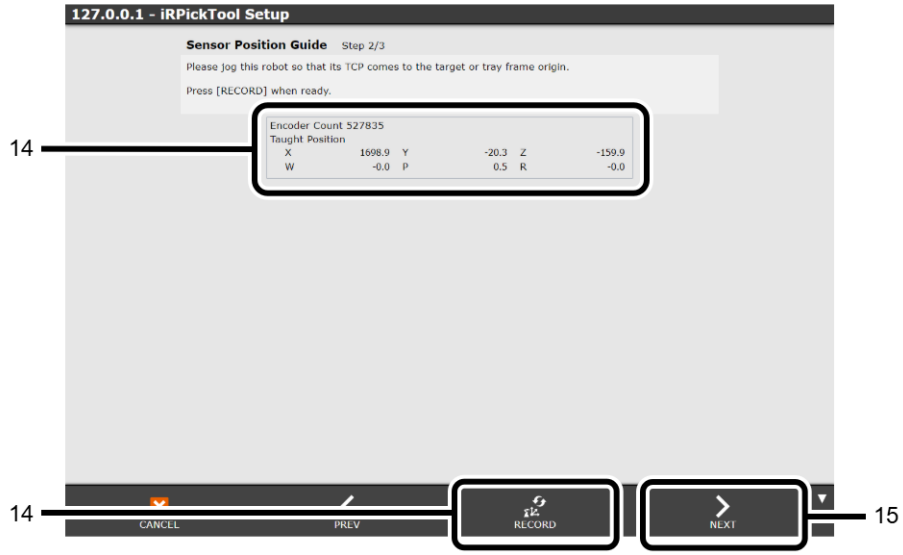
Fig. 2.3.6-3 Touching up the origin of the tray

14 Click [Record].

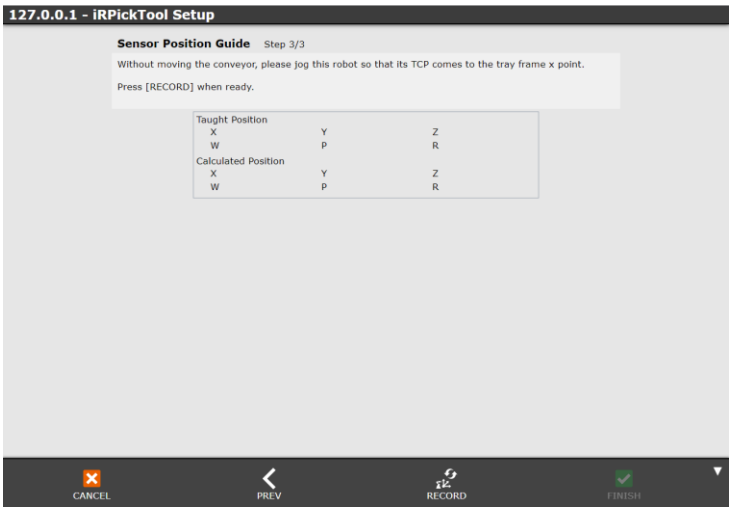
The current encoder count and taught position will appear.

NOTE
In the case of a multi group setup, select the robot group that was touched up in Step 13 and click [Record]. This applies to all subsequent steps.

15 Click [NEXT].



After a while, the following screen appears.



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

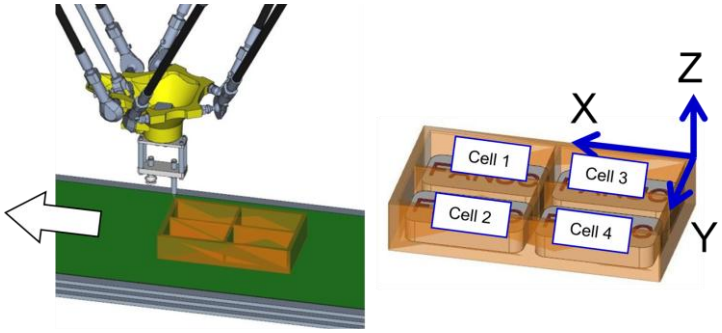
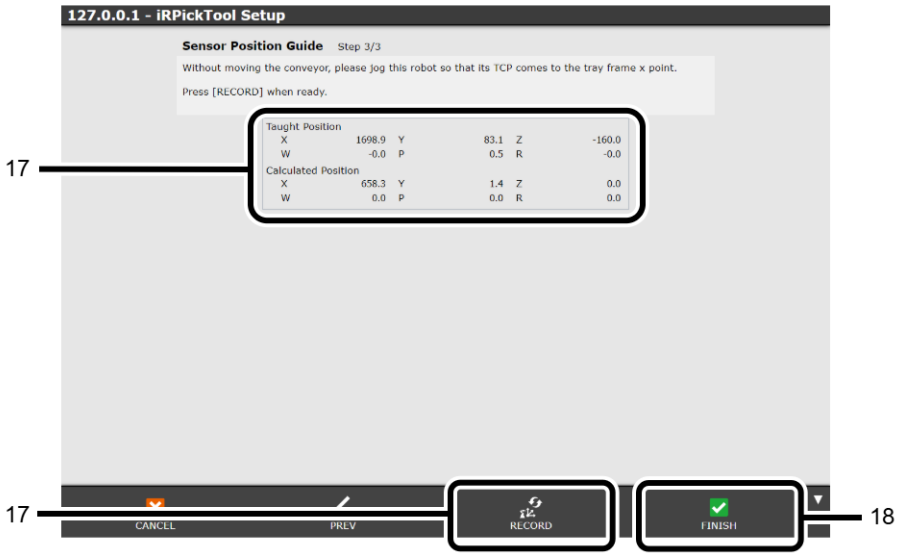


Fig. 2.3.6-4 Touching up the X direction of the tray

17 Click [Record].

The current encoder count and taught position will appear.

18 Click [FINISH].



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

2.3.7. Setting up the vision system

Set up the infeed conveyor vision.

2.3.7.1. Camera calibration

Perform camera calibration.

1 Prepare a calibration grid that is appropriate for the field of view of the camera.

Check the grid spacing at the lower right corner of the grid.

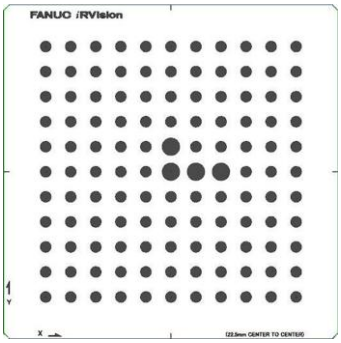


Fig. 2.3.7.1-1 Calibration grid

⚠CAUTION

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

NOTE

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

2 Set the calibration grid directly below the camera.

Fix it with tape to prevent it from moving.

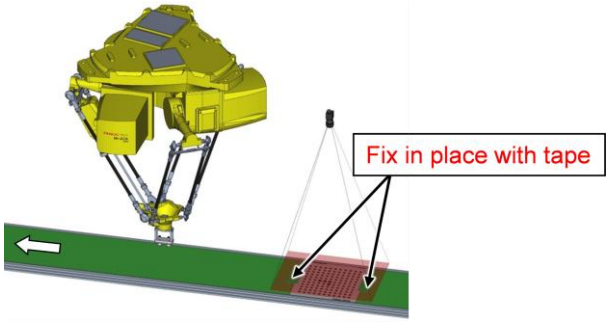


Fig. 2.3.7.1-2 Setting a calibration grid

3 The calibration grid can be set in any orientation.

For example, for line tracking, match the flow direction of the conveyor and the X direction of the calibration grid.

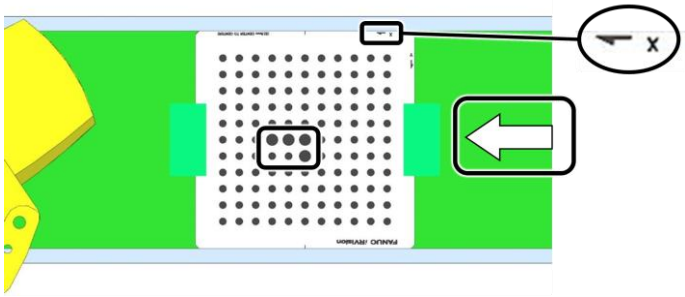


Fig. 2.3.7.1-3 Align the setting direction

4 Press the [MENU] key on the robot controller's teach pendant.

The menu appears.

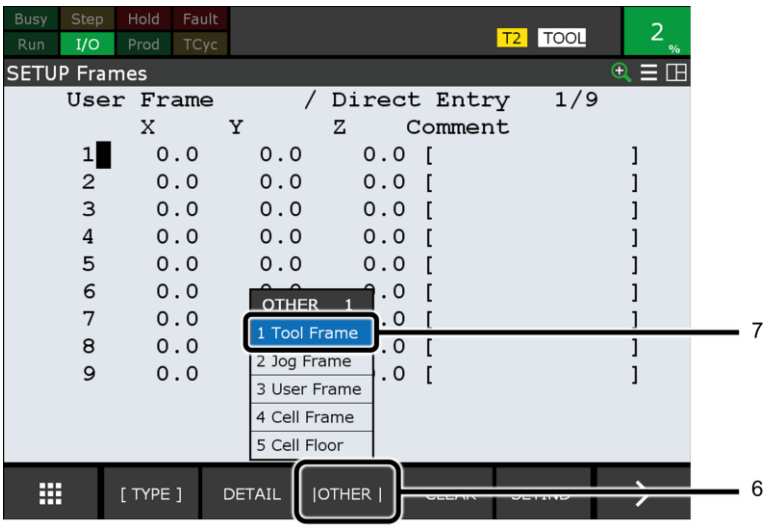
5 From the menu, select [SETUP] → [Frames].



6 Press F3 [OTHER].

7 Select [Tool Frame] from the menu.

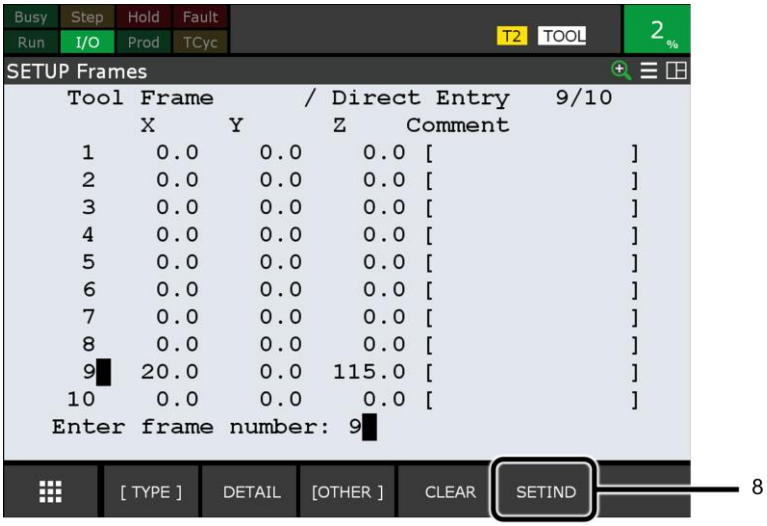
The Tool Frame list screen appears.



8 Press F5 [SETIND].

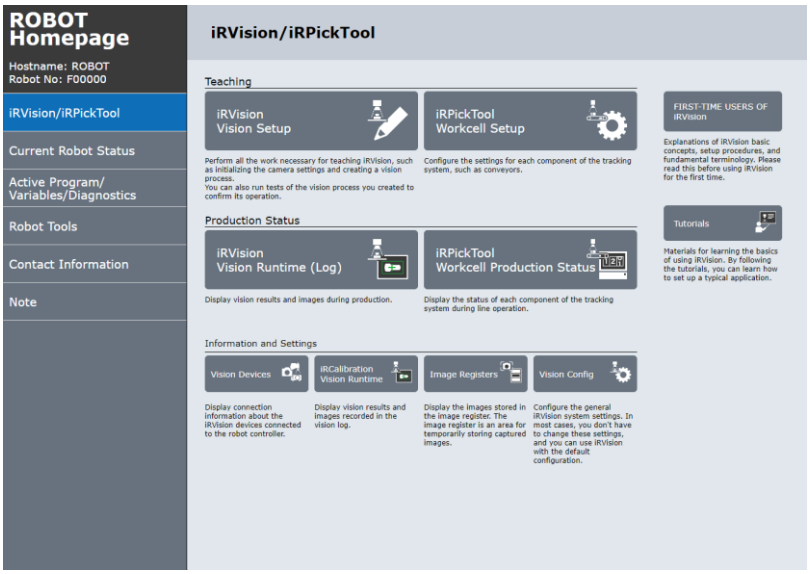
9 Enter the frame number that has been used for the pointer tool.

This procedure is explained using Tool 9 as an example.



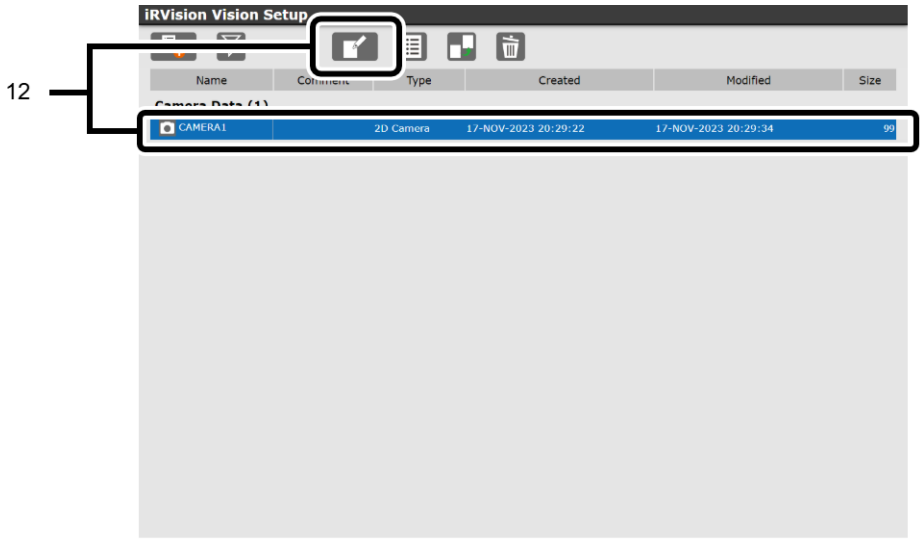
10 Open the robot homepage.

11 Select [iRVision/iRPickTool] in the menu area and click [iRVision Vision Setup] in the content display area.



The [iRVision Vision Setup] screen appears.

12 Select the camera data created in "I.2.2.2. Checking the connection of the camera(P.32)" and click [EDIT].



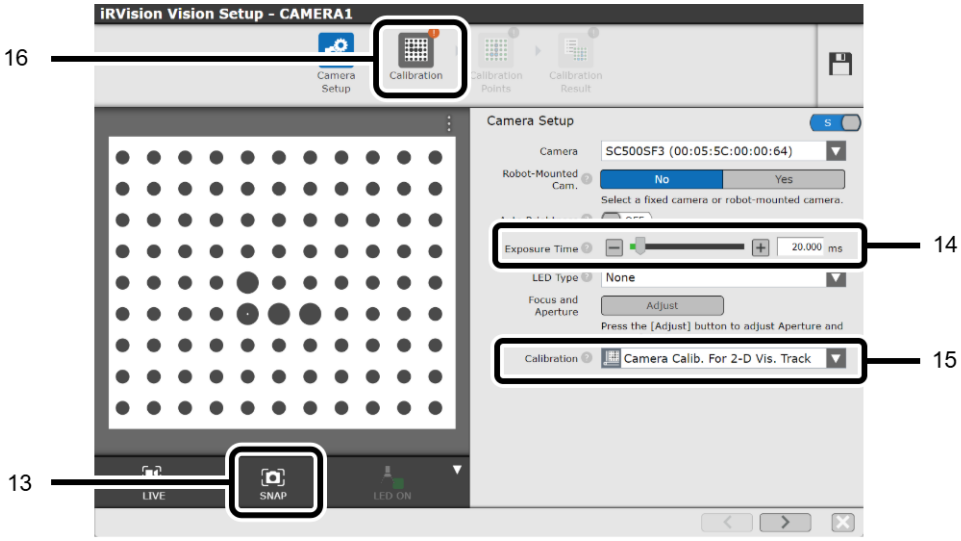
The camera setup screen will appear.

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

14 Enter an [Exposure Time].

15 Select [Calibration Calib. For 2-D Vis. Track] from the [Calibration] drop-down list.

16 Click [Calibration] in Program Line.



The calibration setup screen will appear.

17 Select [Conveyor Name].

18 Enter [Grid Spacing] of the grid pattern.

19 Select [Perspective] in [Projection].

[Perspective] has been selected as the initial value.

CAUTION

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

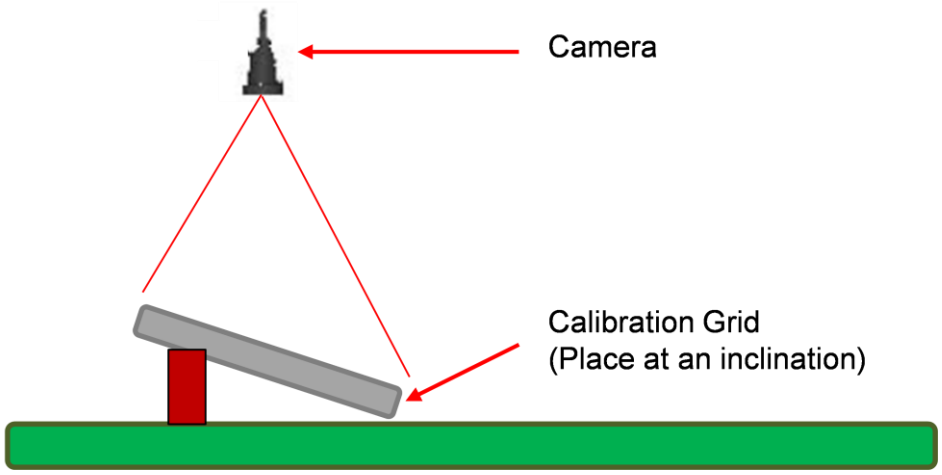
20 In [Camera Distance], enter the focal distance of the lens that is being used.

CAUTION

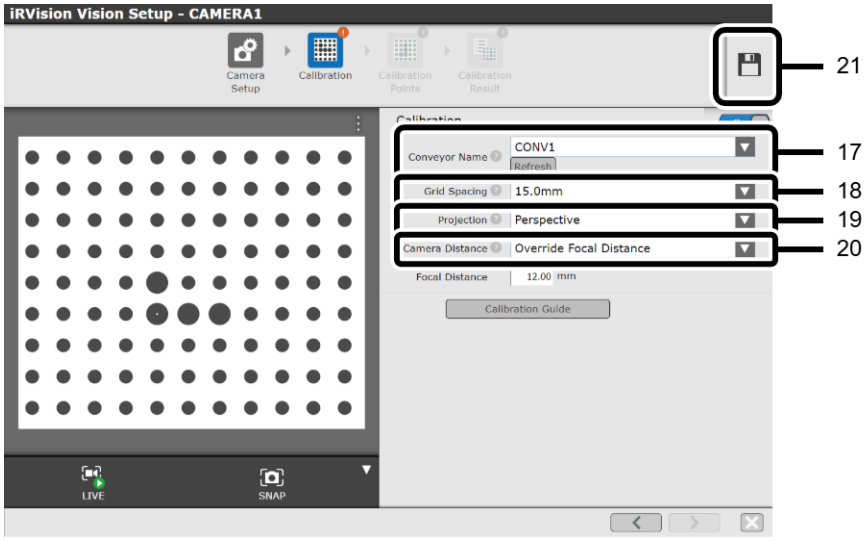
With regard to the calculation method for [Camera Distance], when [Auto] is selected, it is in principle possible that the focal distance might not be calculated accurately because the calibration grid on the conveyor and the camera's light-sensitive surface are nearly parallel.

NOTE

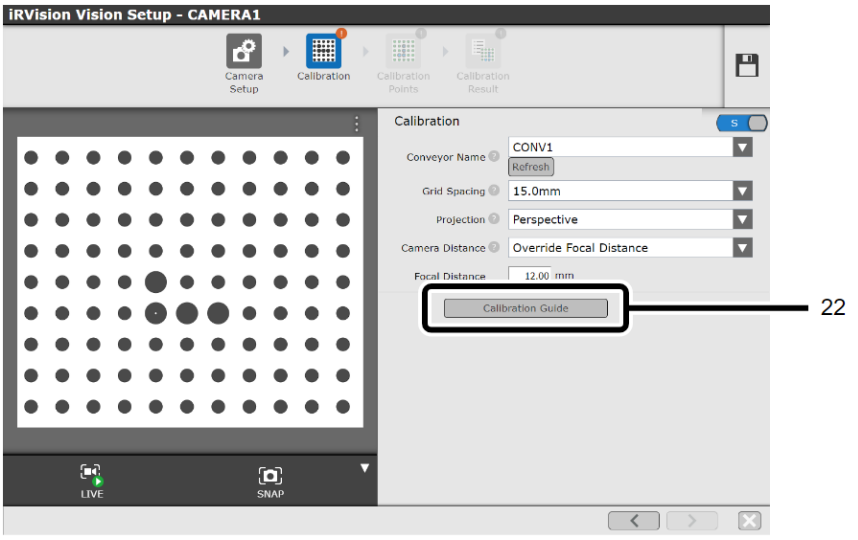
By placing the calibration grid at an inclination, an accurate [Camera Distance] can be calculated using [Auto].



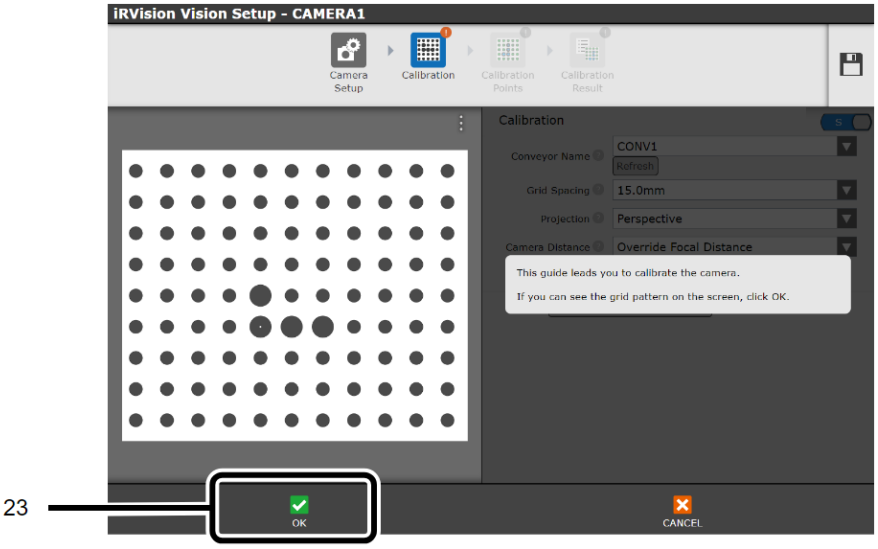
21 Click [SAVE].



22 Click [Calibration Guide].

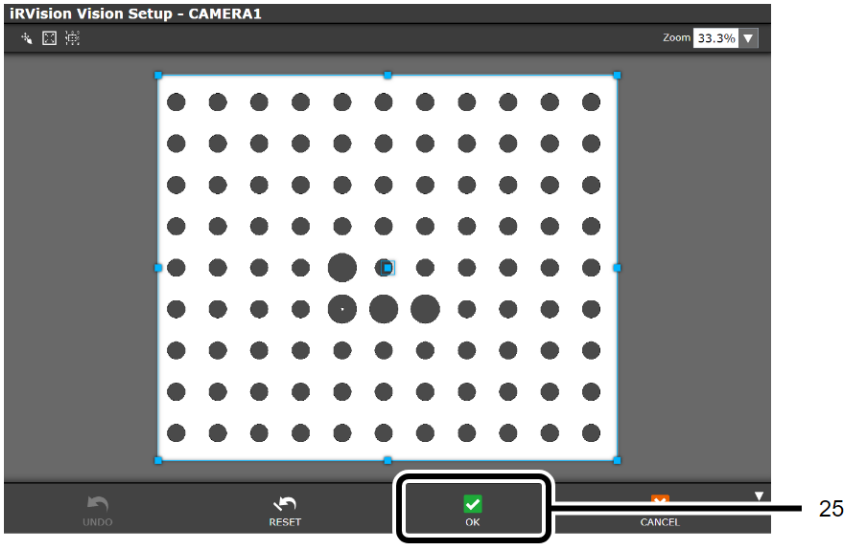


23 Confirm that the calibration grid is displayed on the screen and click [OK].

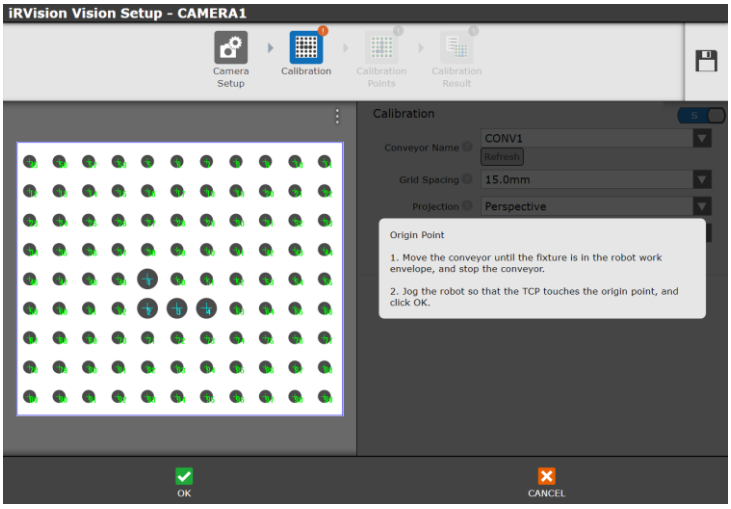


24 Enclose the dot pattern that appears on the screen with a magenta rectangular window.

25 Click [OK].



26 **Move the conveyor, and stop when the calibration grid gets near the center of the robot's operating range.**



27 **Touch up the origin of the calibration grid.**

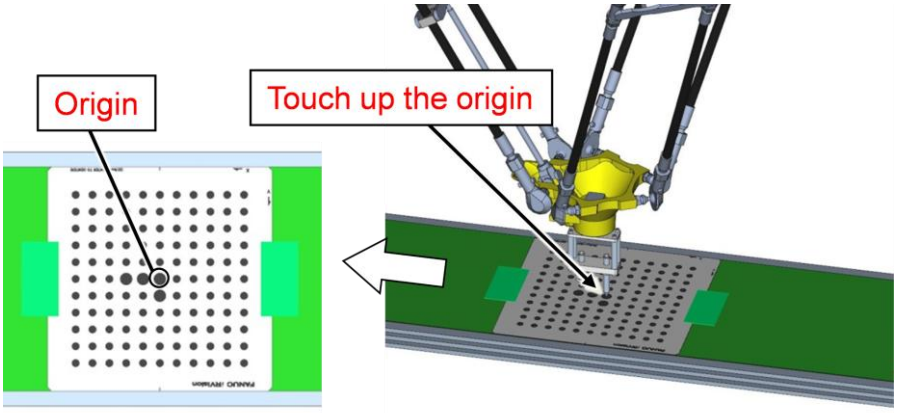
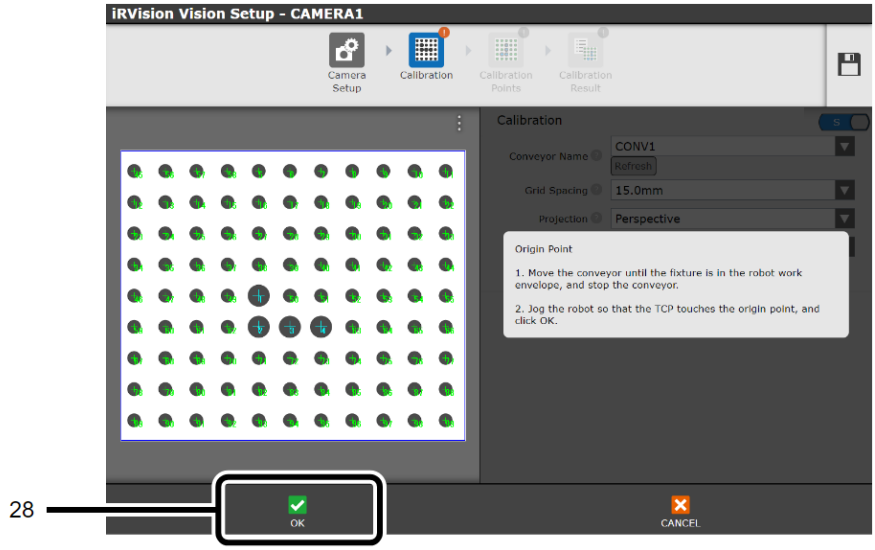
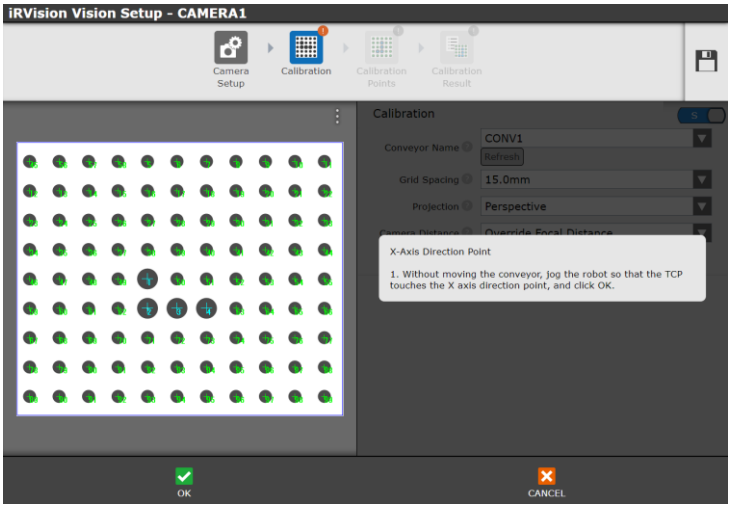


Fig. 2.3.7.1-4 Touching up the origin of the calibration grid

28 **Click [OK].**



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



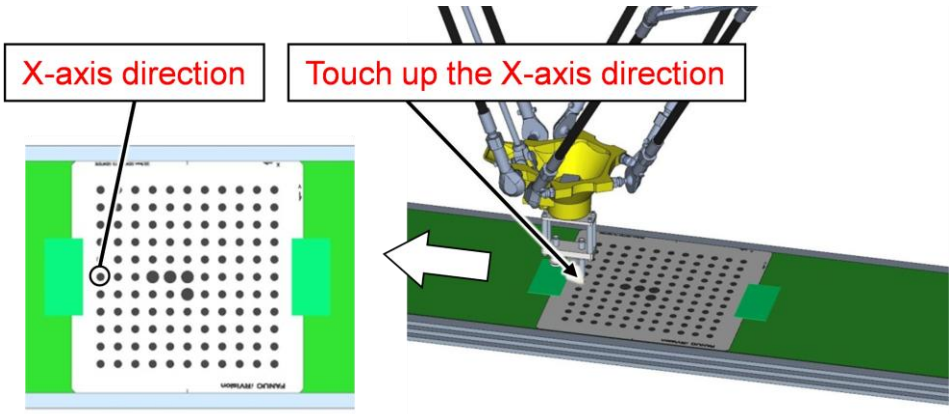
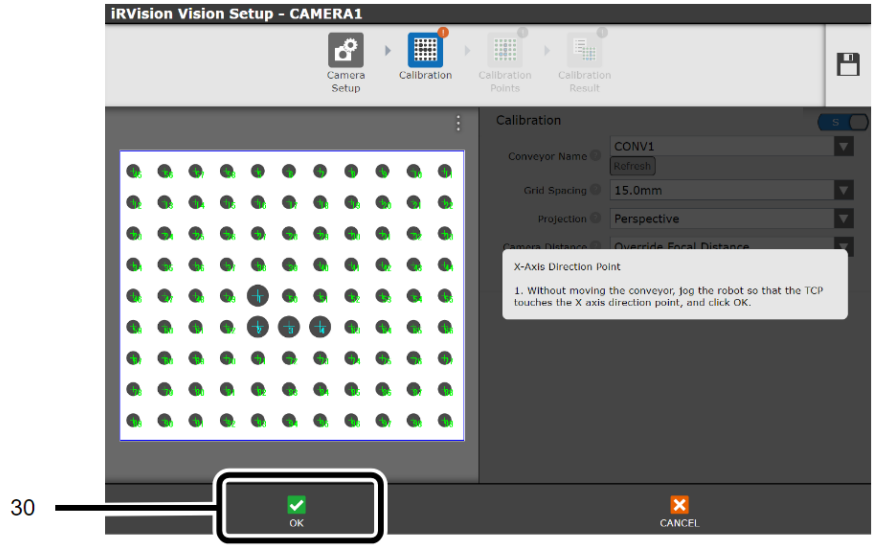
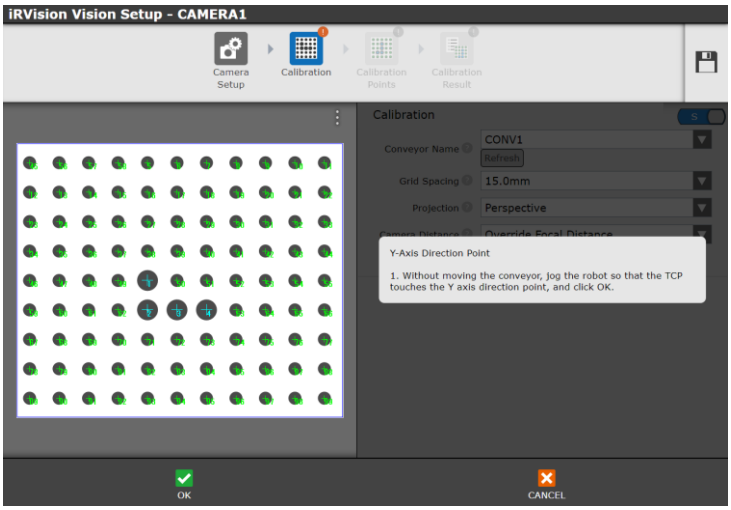


Fig. 2.3.7.1-5 Touching up the X-axis direction of the calibration grid

30 **Click [OK].**



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



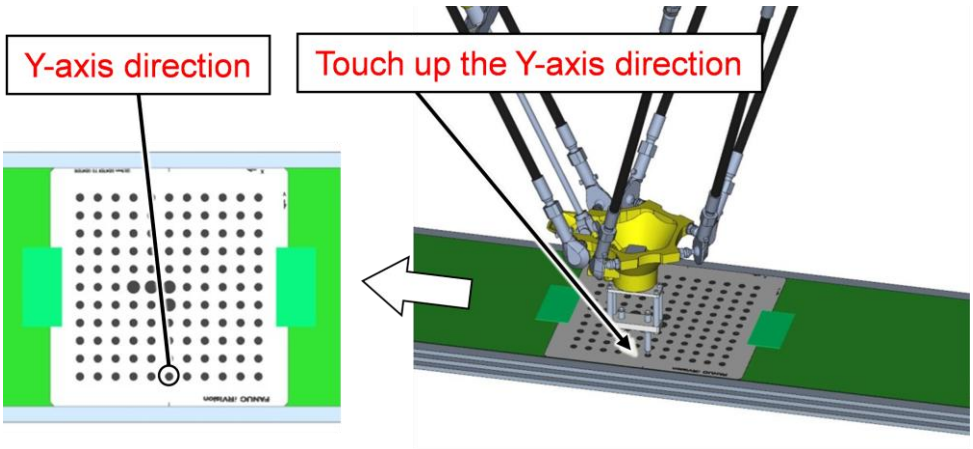
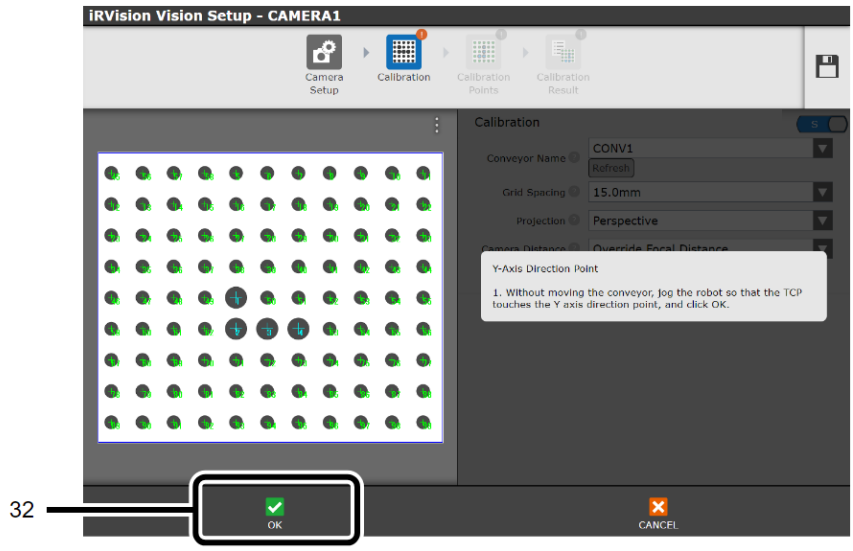


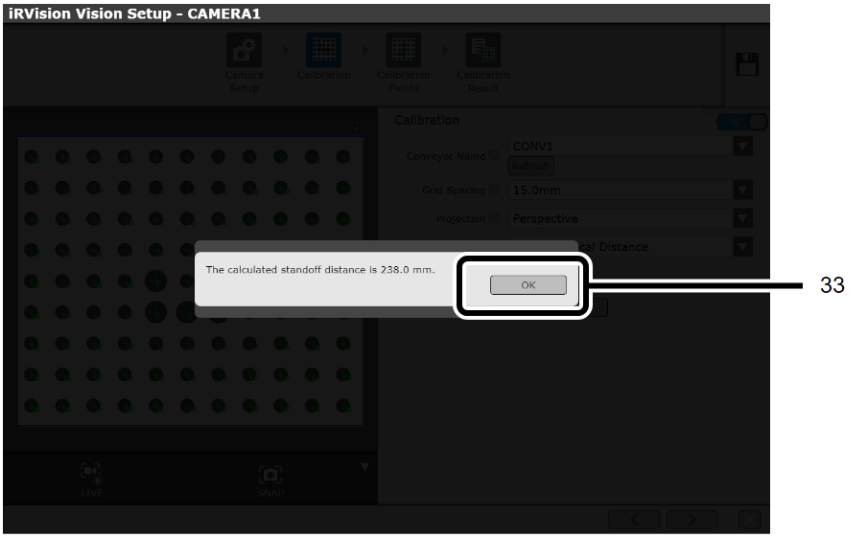
Fig. 2.3.7.1-6 Touching up the Y-axis direction of the calibration grid

32 **Click [OK].**

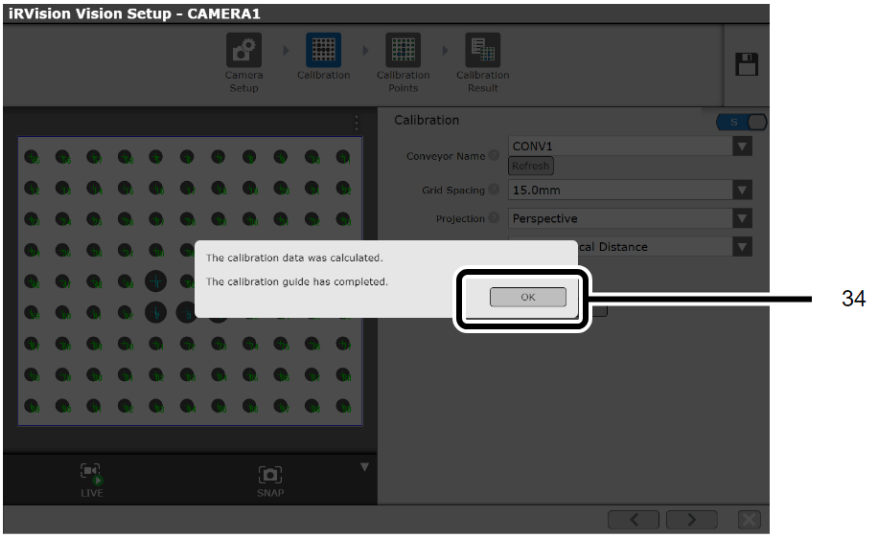


33 The calculated standoff distance is displayed. Confirm that the value is suitable for the actual standoff distance, and click [OK].

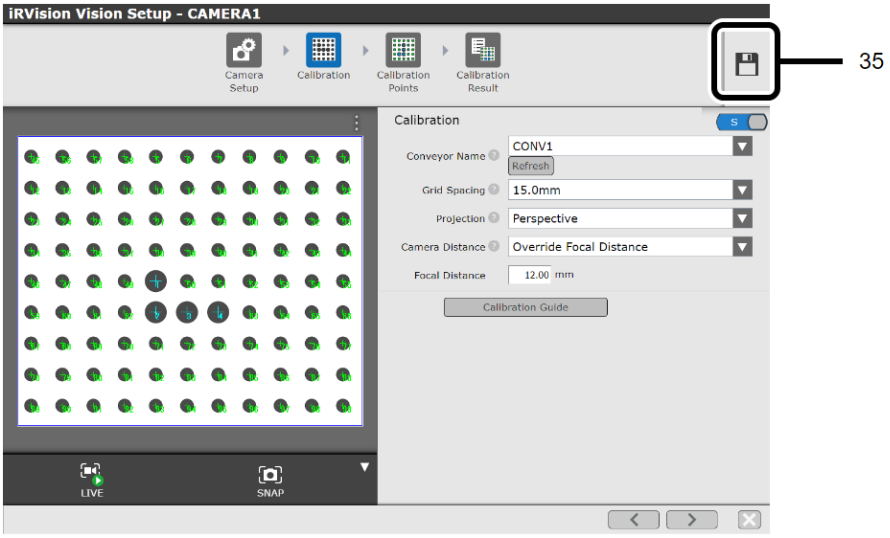
If the value is unsuitable, check whether the setting procedure and set parameters are correct.



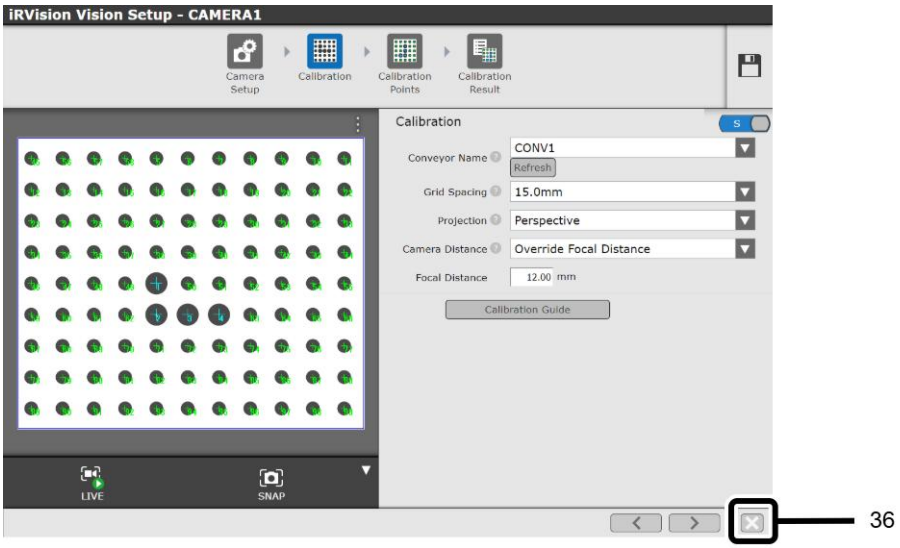
34 Click [OK].



35 Click [SAVE].



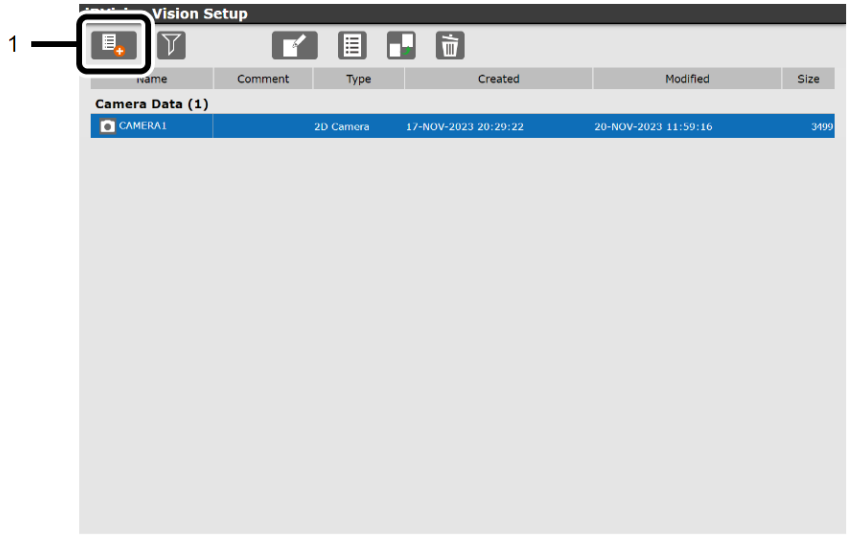
36 **Click [×].**



2.3.7.2. Vision process

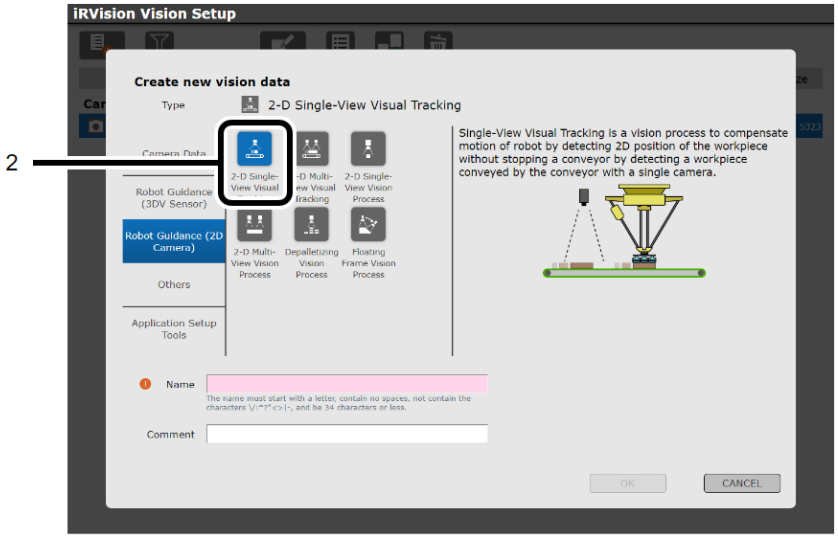
Set up a vision process.

1 **Click [CREATE].**



The [Create new vision data] screen appears.

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



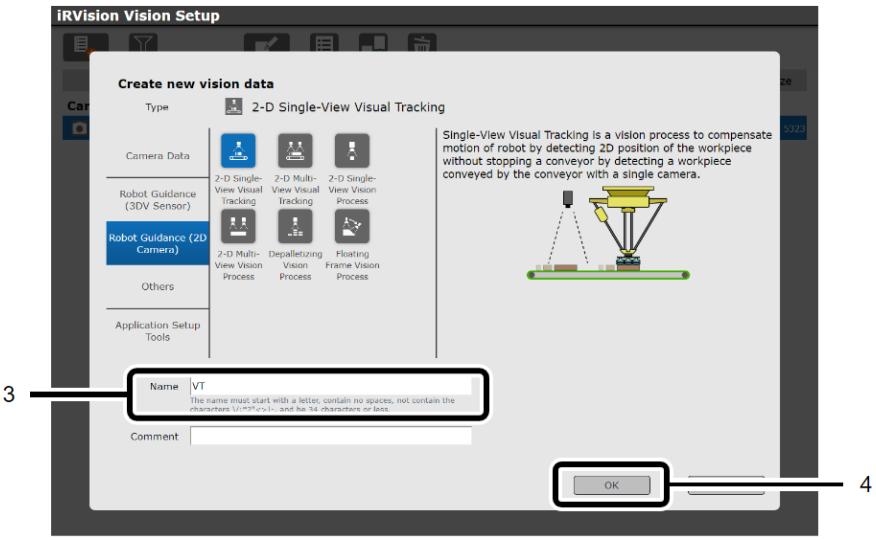
After a while, the following screen appears.

3 Enter the vision process name in [Name].

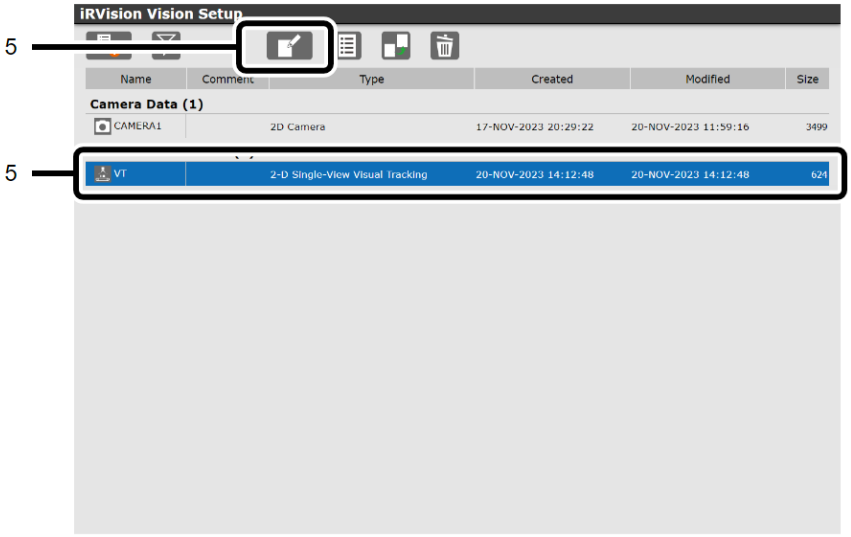
Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

Example: VT

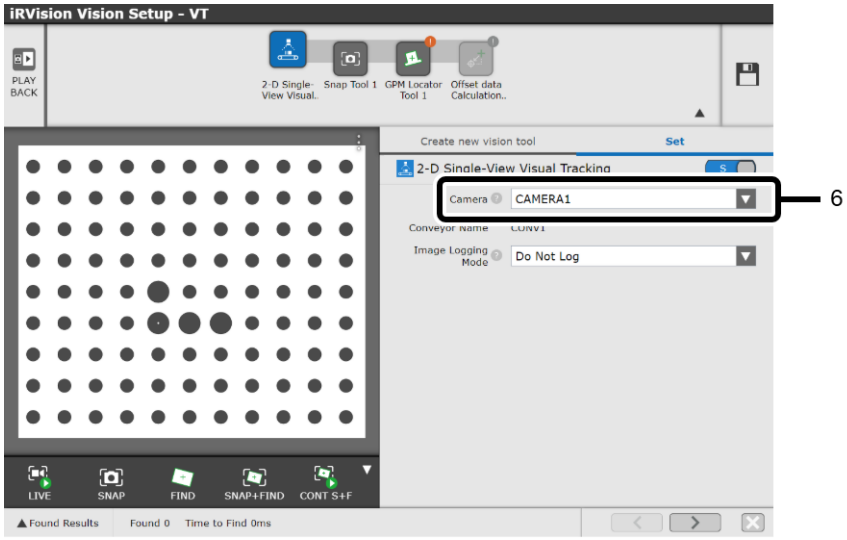
4 Click [OK].



5 On the vision data list screen, select a created vision process, and click [Edit].



6 Select the connected camera in the [Camera] drop-down list.

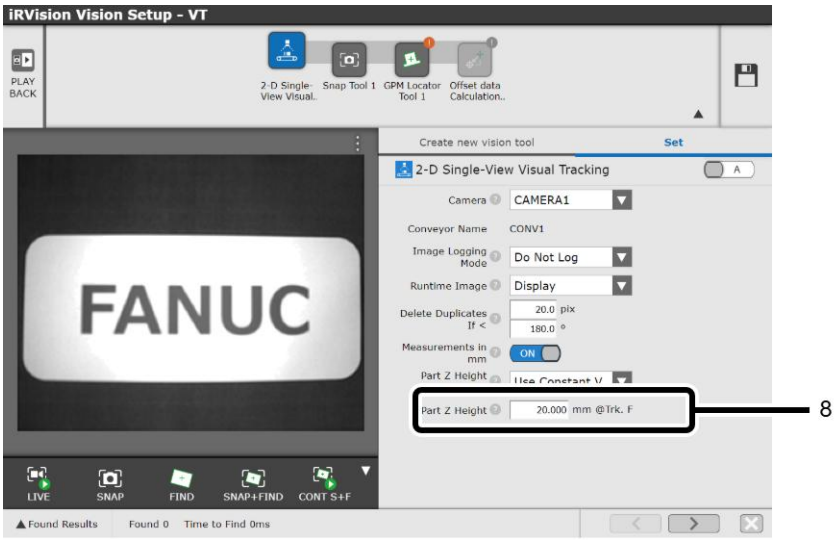


7 Measure the thickness of a part.

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

8 Enter the thickness of the part measured in step 7 into [Part Z Height].

Enter the distance from the plane at which the tracking frame was taught (the calibration grid surface) to the plane where the features that are to be found by vision will be.



NOTE

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

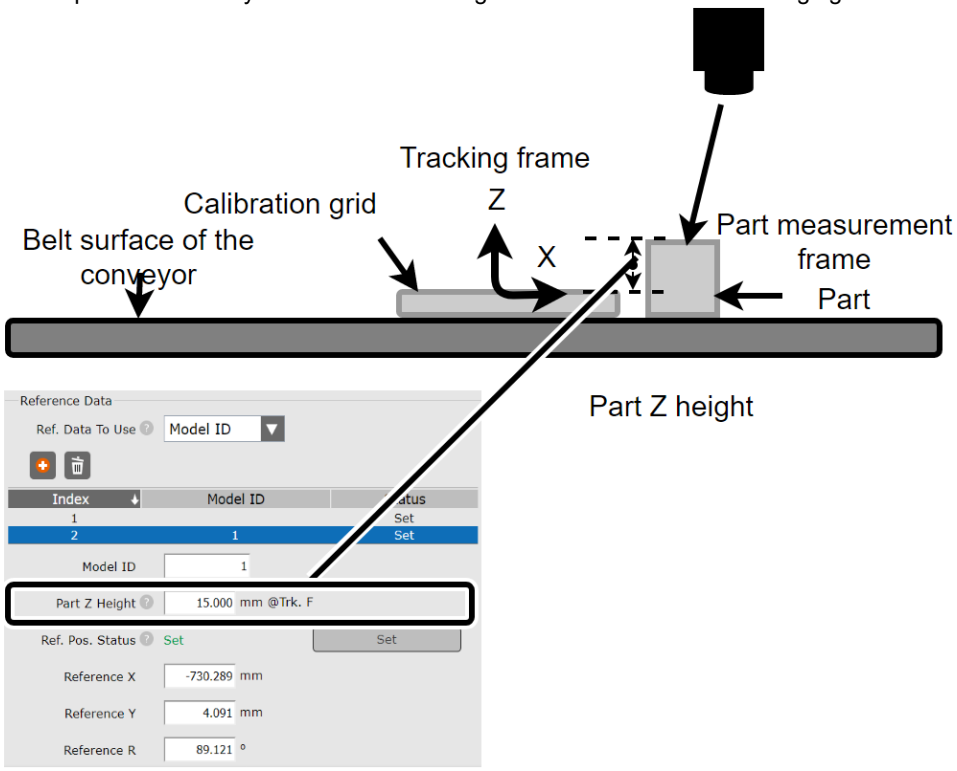
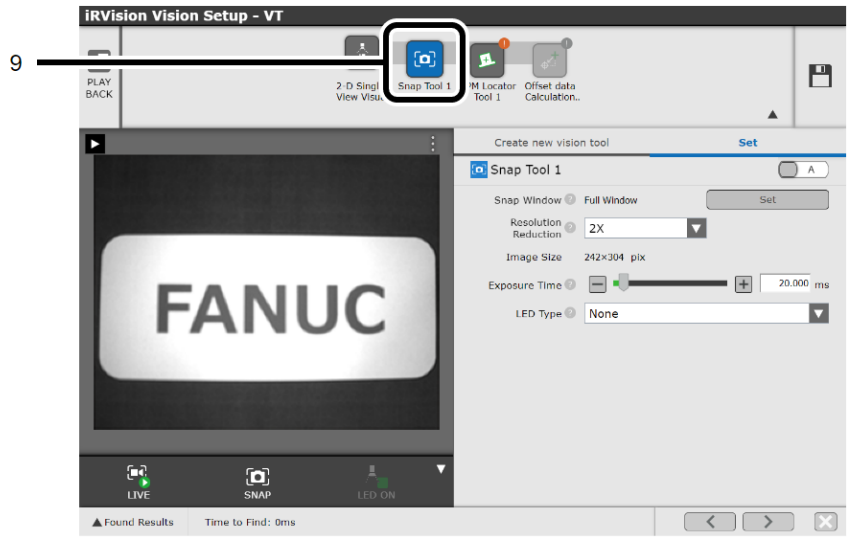


Fig. 2.3.7.2-1 Relationship between the system and 'Part Z Height'

9 Select [Snap Tool 1] in Program Line.



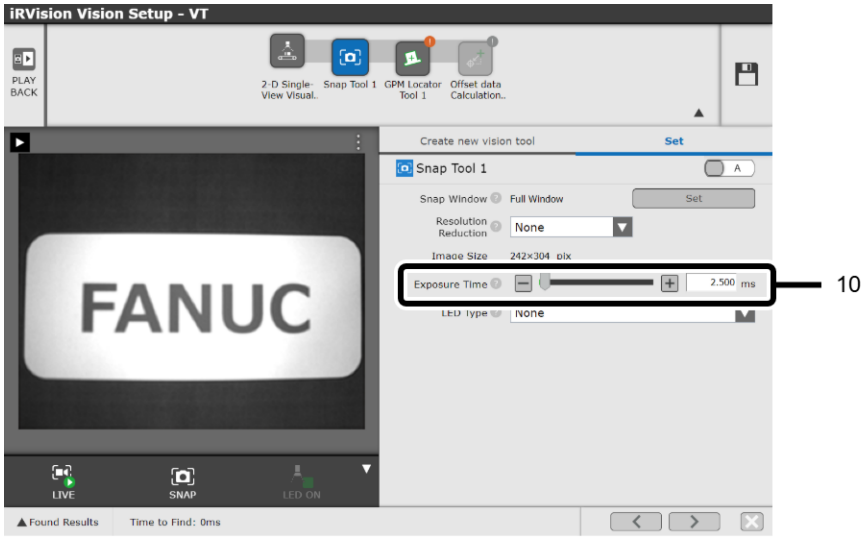
10 Enter an [Exposure Time].

Set the exposure time so that an image of a part that moves during exposure will have as little blurring as possible.

For, example, taking into account image blurring, in order to suppress misalignment during finding by vision to less than 0.1 mm when the conveyor speed is 100 mm/s, set a value that is less than 1 ms as the exposure time.

0.1 (mm) ÷ 100 (mm/s) = 1 (ms)

For information on the calculation of appropriate exposure times to suit the conveyor speed, refer to "Exposure time(P.22)".



11 Set a part in the field of view of the camera.

If the supply direction is fixed, set it by aligning it.

If the feed part variation is ±180 degrees, teaching may be easier if the orientation is aligned with the placing operation.

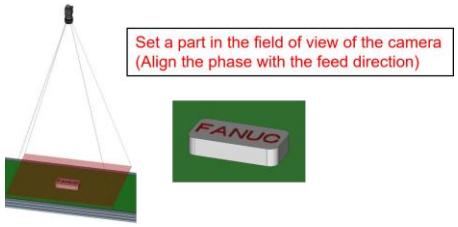
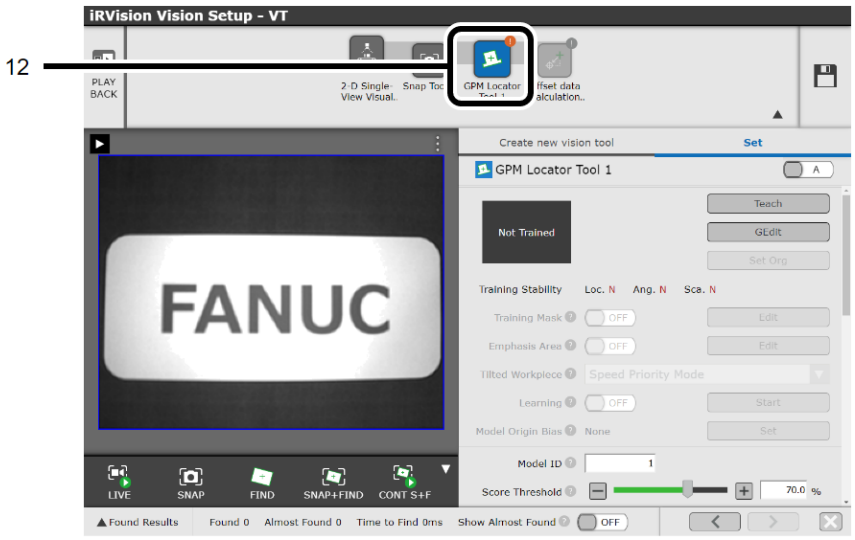


Fig. 2.3.7.2-2 Setting a part

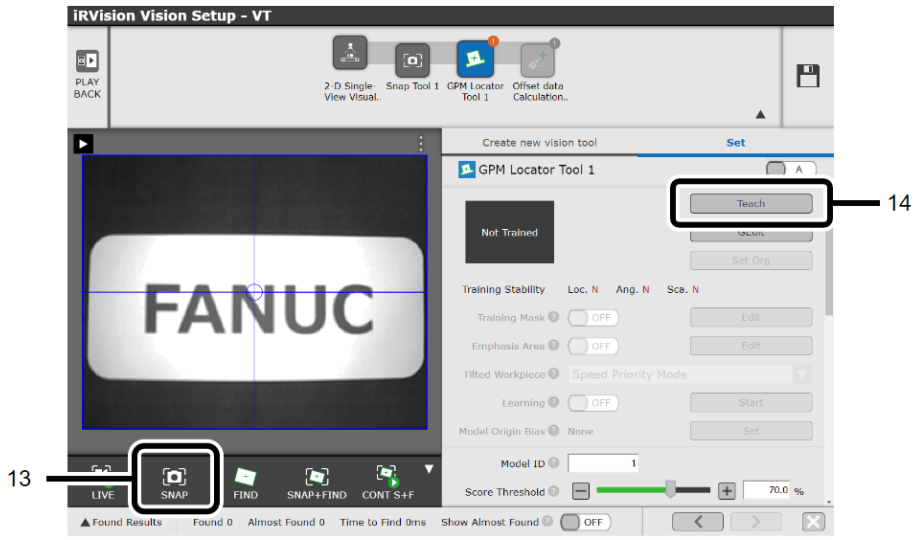
12 Select [GPM Locator Tool 1] in Program Line.



13 Click [SNAP].

Verify that the part appears on the screen.

14 Click [Teach].



15 Enclose the part with a magenta rectangular window.

16 Click [OK].



17 Click [Edit] in [Training Mask].



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

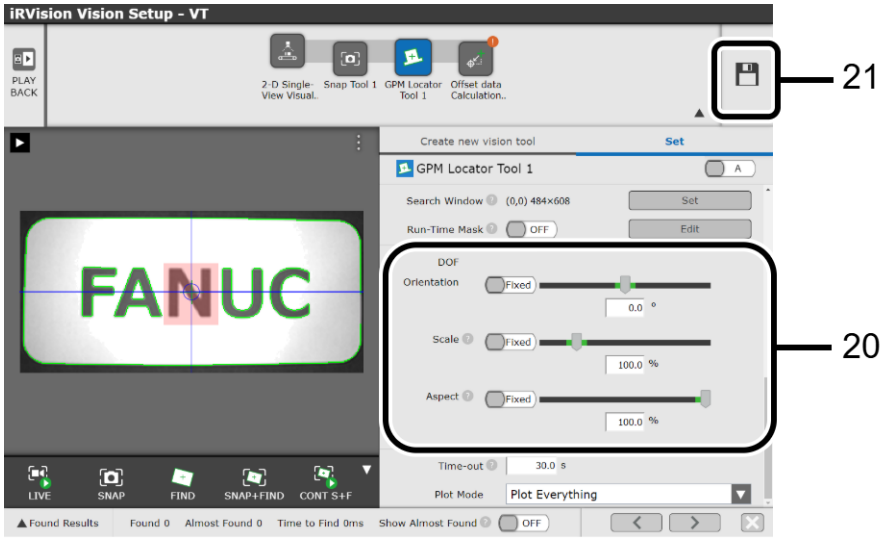
19 Click [OK].



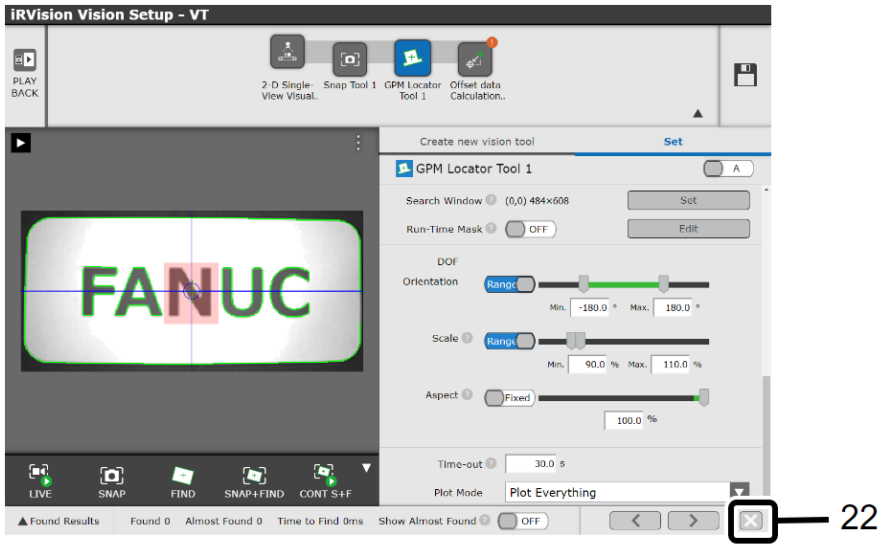
20 Set up the DOF as necessary.

For details of the setup, see the section “GPM Locator Tool” in the chapter “Command tools” in the "iRVision OPERATOR'S MANUAL (Reference)"(B-84434EN).

21 Click [SAVE].



22 **Click [×].**



2.3.8. Setting up a conveyor station

Set up a conveyor station.

1 **Press the [MENU] key on the robot controller's teach pendant.**
The menu appears.

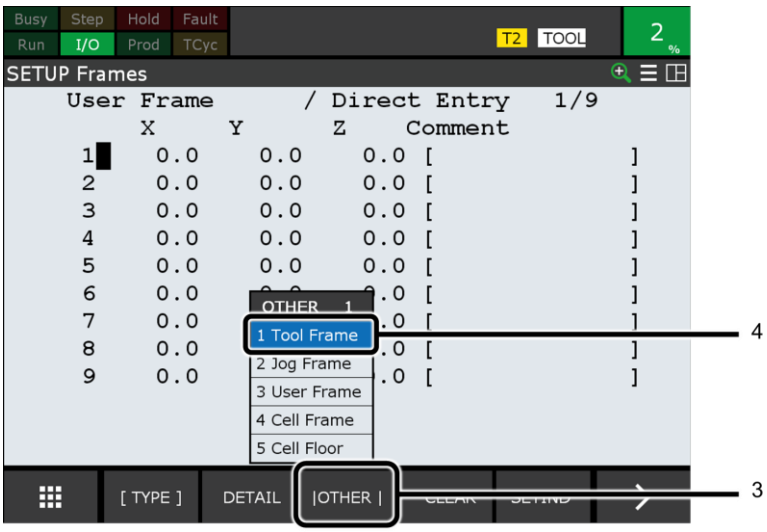
2 **From the menu, select [SETUP] → [Frames].**



3 Press F3 [OTHER].

4 Select [Tool Frame] from the menu.

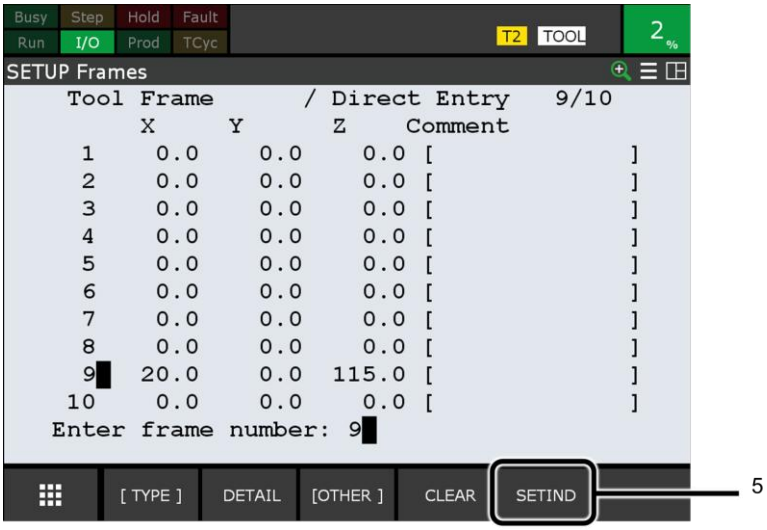
The Tool Frame list screen appears.



5 Press F5 [SETIND].

6 Enter the frame number that has been used for the pointer tool.

This procedure is explained using Tool 9 as an example.



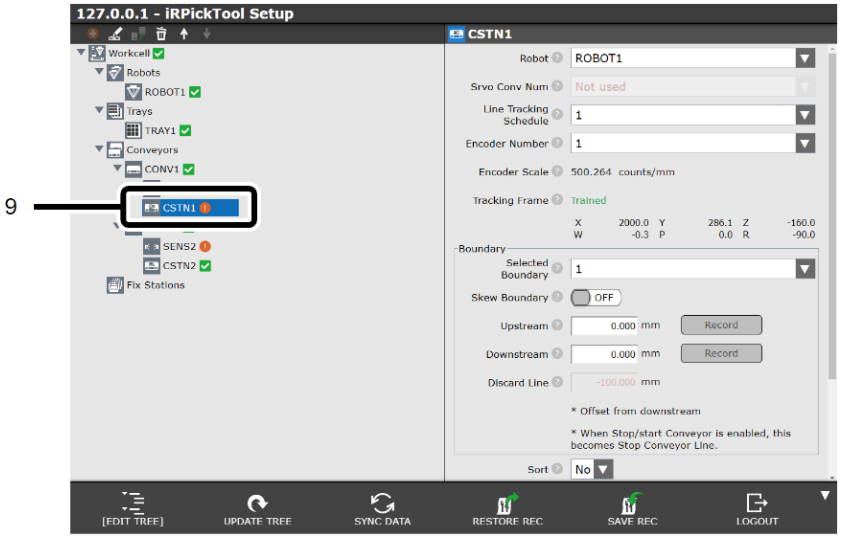
7 Press the [MENU] key on the robot controller's teach pendant.

The menu appears.

8 Select [SETUP] → [iRPickTool] from the menu.



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



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

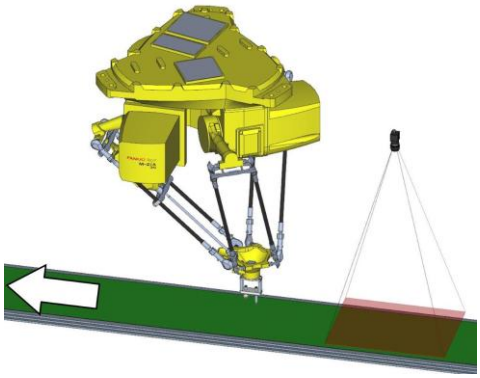
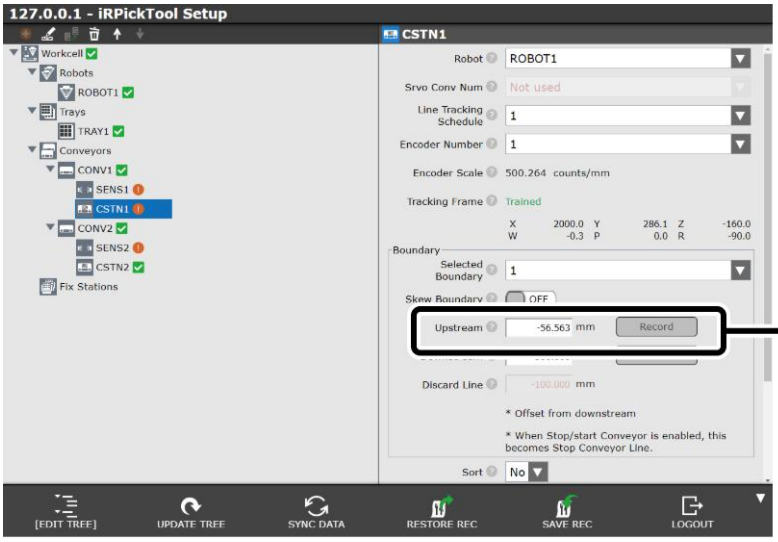


Fig. 2.3.8-1 Jogging the robot

11 Click [Record] for [Upstream Boundary].

The current position in the tracking frame will be saved.



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

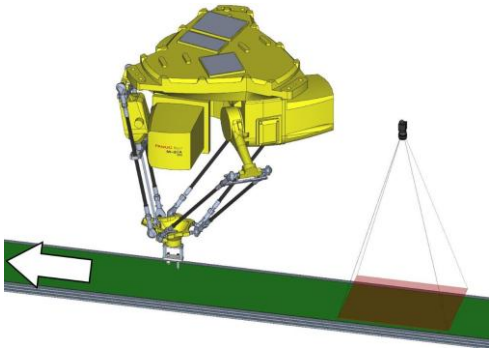
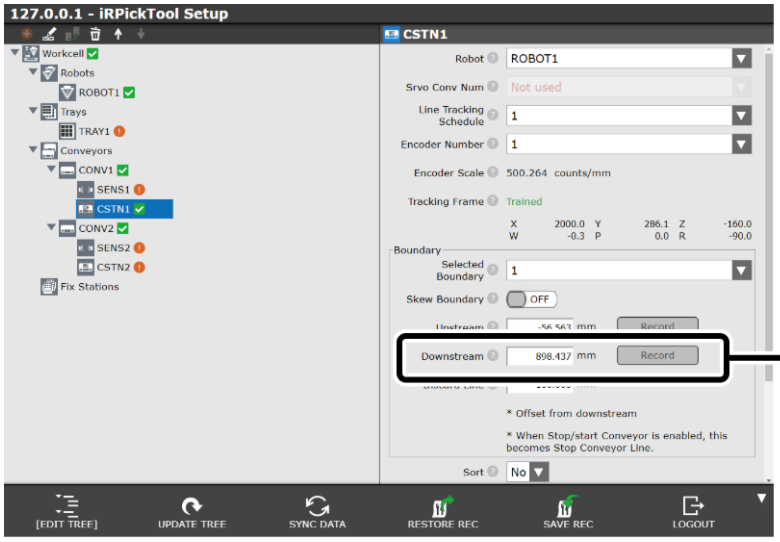


Fig. 2.3.8-2 Jogging the robot

13 Click [Record] for [Downstream].

The current position in the tracking frame will be saved.



14 Enter [Discard Line].

The discard line value must be larger than the value as calculated below.

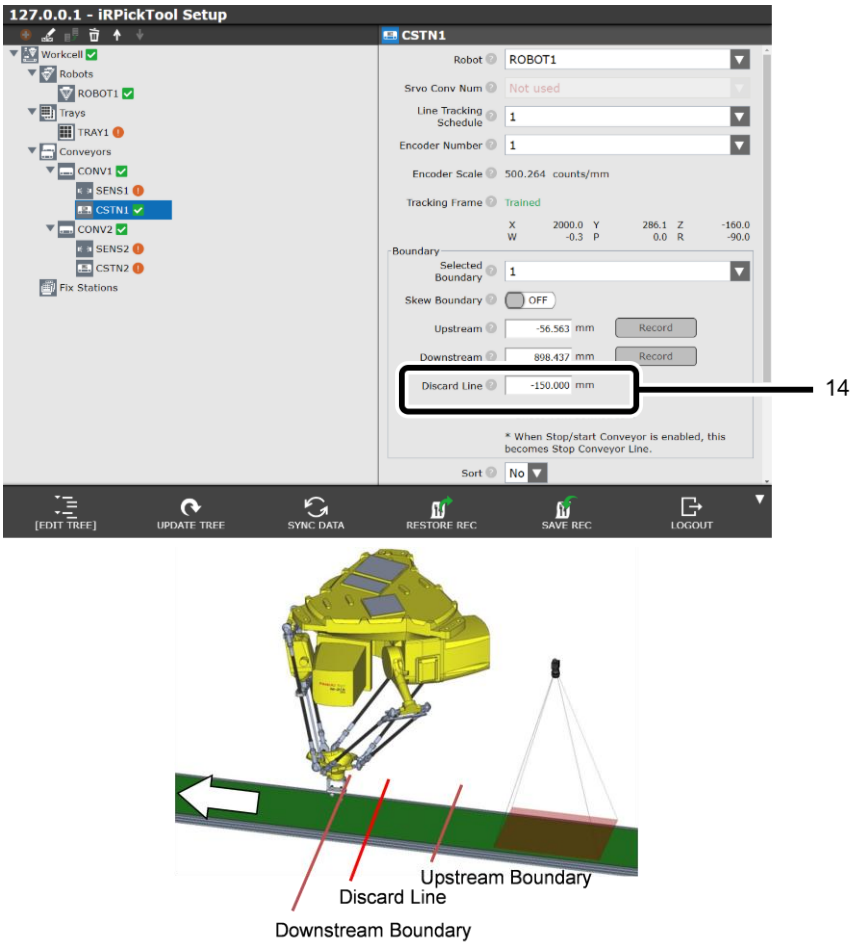
Time required for tracking* (sec) × Conveyor speed (mm/sec)

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

For example, if the robot operation time is approximately 0.8 seconds and the conveyor speed is 150 mm/sec, the free-running distance of the conveyor is:0.8 (sec) × 150 (mm/sec) = 120 (mm)

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

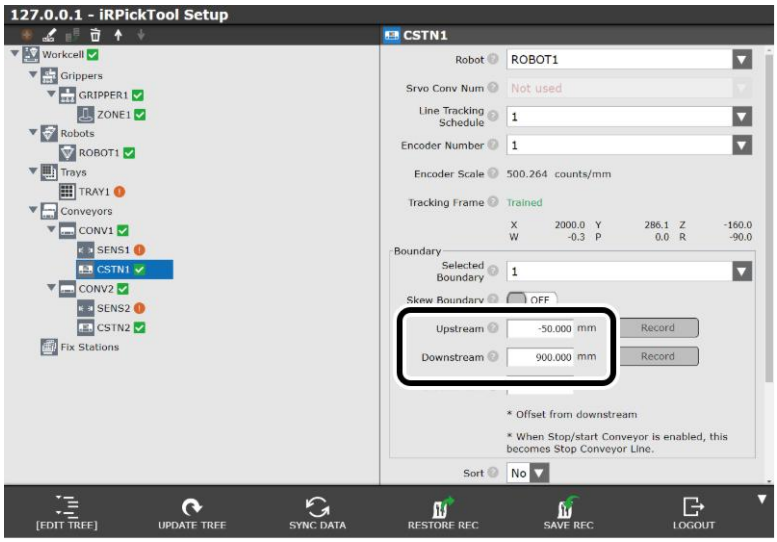
For details on discard lines, see "I.5.6. Setting up a conveyor station(P.318)".



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

Fig. 2.3.8-3 Setting up a discard line

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



15 Next, set up the conveyor station of the outfeed conveyor. Select the conveyor [CSTN2].

16 **Repeat steps 10 to 14 to set [Upstream], [Downstream], and [Discard Line] for the conveyor station of the outfeed conveyor.**

2.3.9. Setting up a tool frame for the hand

Set up a tool frame for the hand.

In this section, a four-axis delta robot is used as an example.

With a four-axis delta robot, we recommend that X and Y of the tool frame be set to 0, even if (X, Y) of the tool center point (TCP) of the hand in the mechanical interface frame is not (0, 0). This is to prevent the cycle time from being delayed or the robot's failing to stay in its operating range, as a result of large movement by the major axes (J1 to J3 axes) when a part is picked from the conveyor.

To make the tool's Z-axis face upward, directly input the tool's orientation. Directly input the values of the rotation angles W, P, and R of the tool frame around the mechanical interface frame's X, Y, and Z axes.

1 **Press the [MENU] key on the robot controller's teach pendant.**
The menu appears.

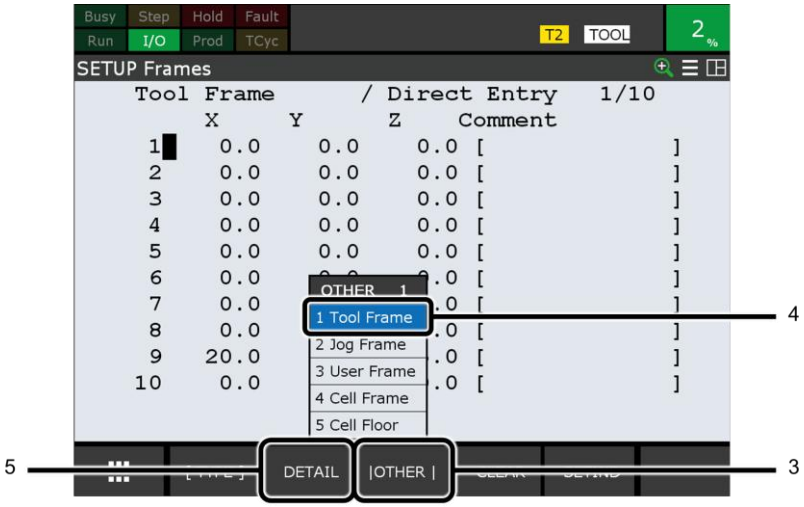
2 **From the menu, select [SETUP] → [Frames].**



3 **Press F3 [OTHER].**

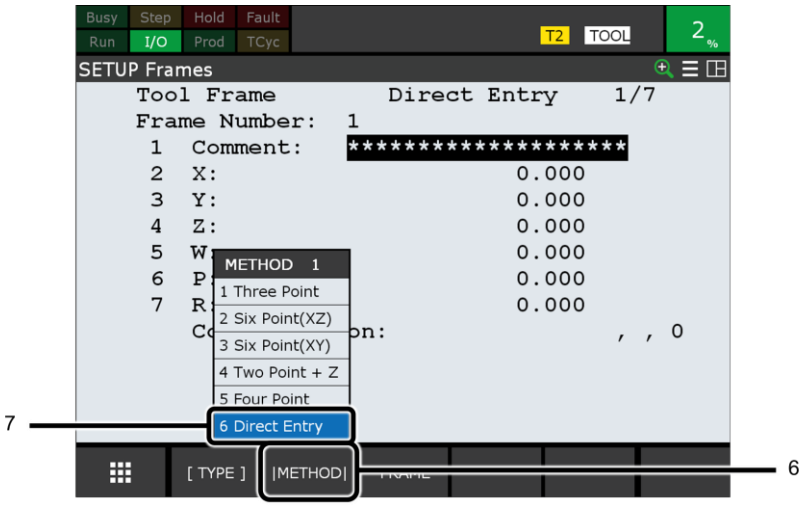
4 **Select [Tool Frame] from the menu.**
The Tool Frame list screen appears.

5 **Move the cursor to the line of the tool frame number to be set and press F2 [DETAIL].**
The screen is an example of when performing setup using Tool 1.
The setup screen for the tool frame for the selected frame number will appear.



6 Press F2 [METHOD].

7 Select [Direct Entry] from the menu.

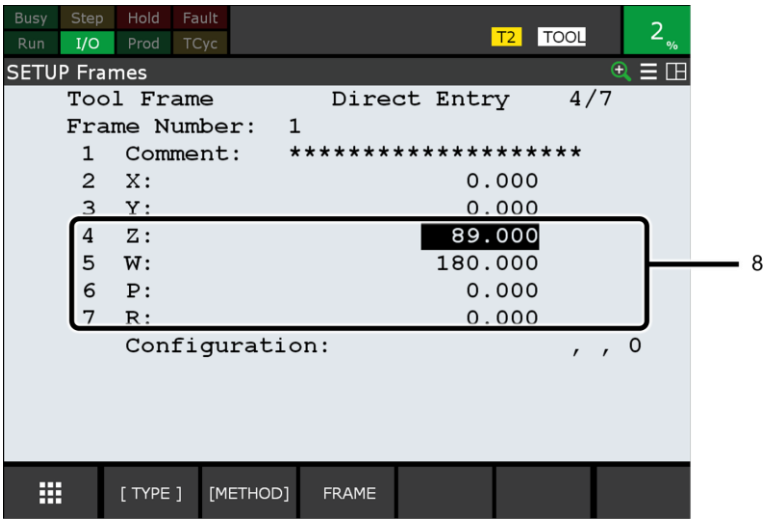


8 Move the cursor to [Z], [W], [P] and [R] and enter a value for each.

For [Z], enter a numerical value measured with a ruler, etc., or a hand design value.

Input [W], [P] and [R] directly.

* (W, P, R) = (180, 0, 0) is recommended.

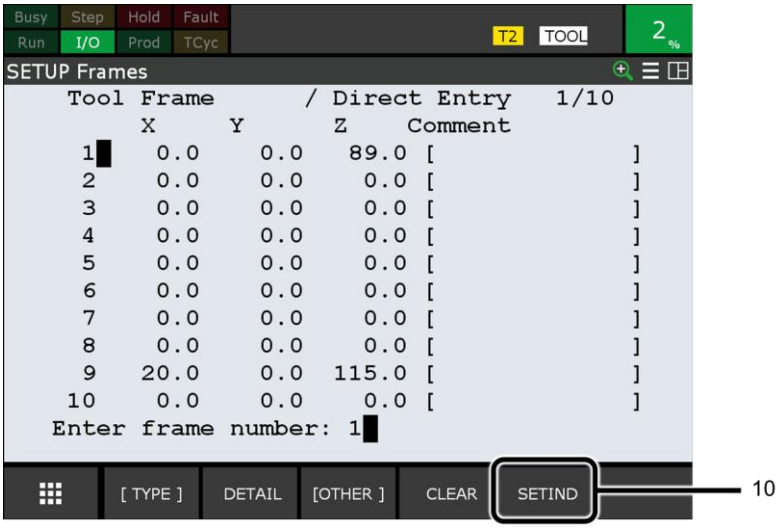


9 Press the [PREV] key on the teach pendant of the robot controller.

The Tool Frame list screen appears.

The setting values of all the tool frames can be checked.

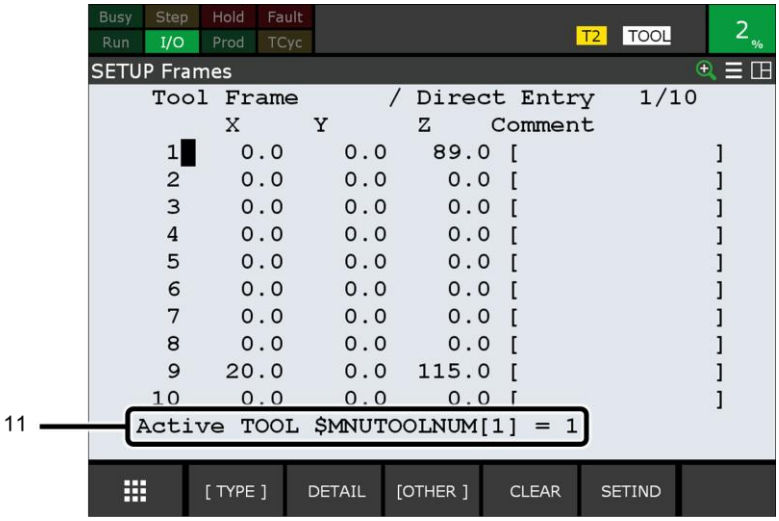
10 Press F5 [SETIND].



11 Enter a frame number.

The tool frame will be enabled.

* If you do not carry out steps 10 and 11, the frame that has been set up will not be active.



2.3.10. Setting up a reference position and teaching a robot position

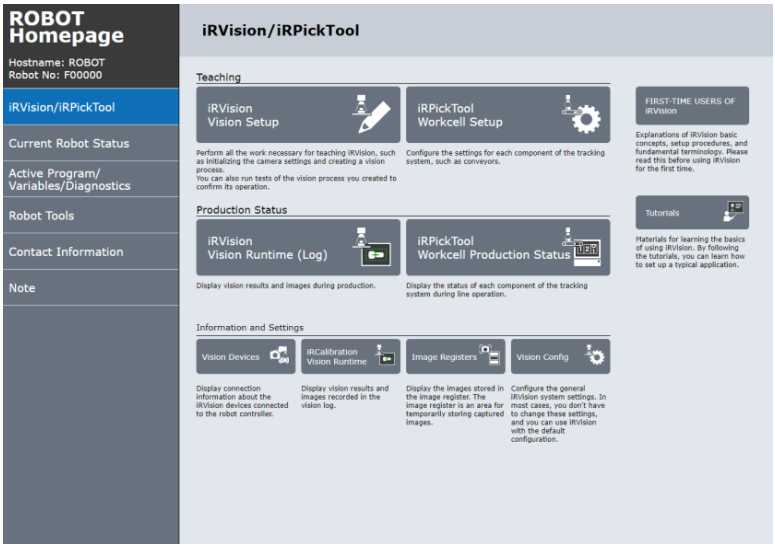
This procedure is for setting the reference position and teaching positions to the robot.

2.3.10.1. Setting up a reference position for the infeed conveyor and teaching a robot position

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

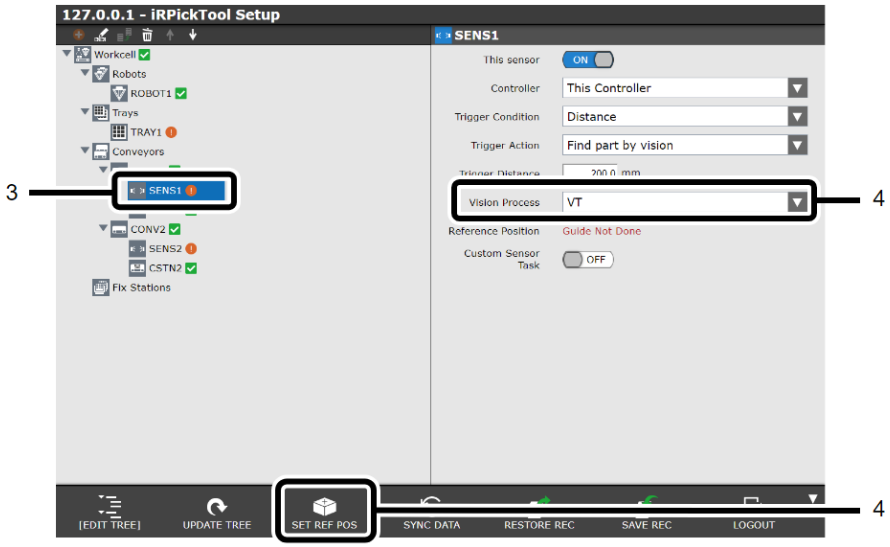
1 Open the robot homepage.

2 Select [iRVision/iRPickTool] in the menu area and click [iRPickTool System Setup] in the content display area.

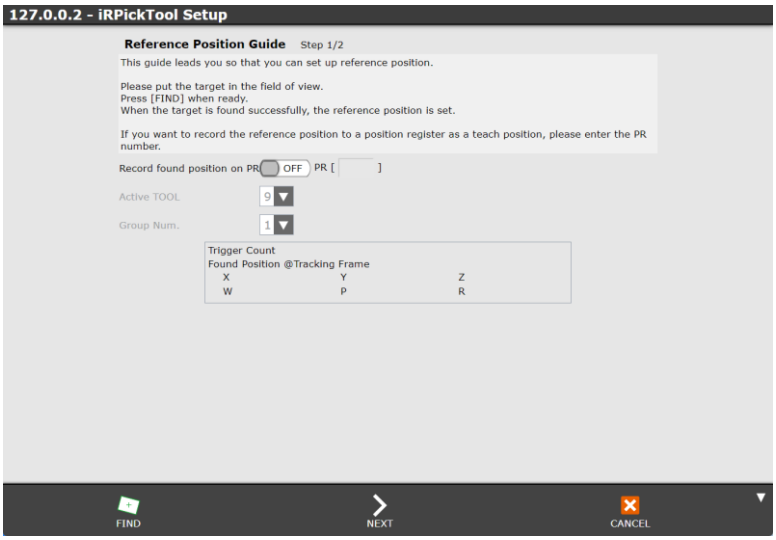


3 Select [SENS1] in [CONV1] and select the name of the vision process (VT) that you created in "I.2.3.7.2. Vision process(P.99)" from the [Vision Process] drop-down list.

Click [Reference Position].



The following [Reference Position Guide Step: 1/2] will appear.



NOTE

When [Record found position on PR] is available, the robot position teaching described later can be simplified by using this function. For details, see "I.2.3.10.1. Setting up a reference position for the infeed conveyor and teaching a robot position(P.113)" described later.

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

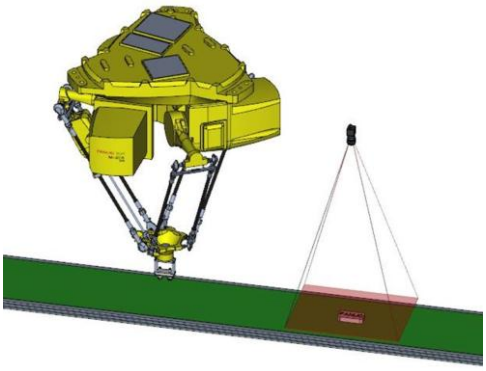


Fig. 2.3.10.1-1 Placing a part inside the field of view of the camera

6 Click [FIND].

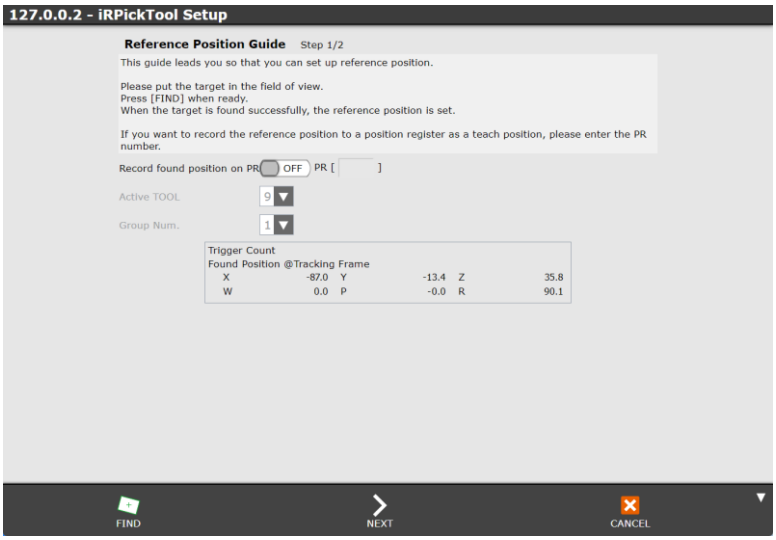
If detection succeeds, [Found Position] will be displayed.

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

NOTE

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

8 Click [NEXT].



9 Click [FINISH].

Check that [Guide Done] appears in [Reference Position].



10 Move the conveyor, and then when the part enters the robot's moving range, stop it.

* Be careful that the part does not become misaligned.

11 Move the robot in jog mode to the part pick position.

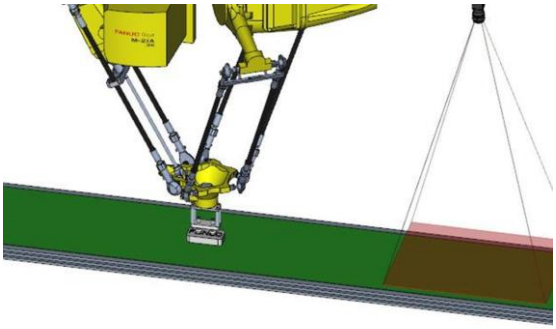


Fig. 2.3.10.1-2 Moving the robot to the part pick position

12 Display the pick program PT_PICKCS, and move the cursor to the line where the pick position is taught.

13 While holding down the [SHIFT] key, press F5 [TOUCHUP].

If F5 [TOUCHUP] is not displayed, press [> (NEXT)] to switch screens.



14 When the message saying "VR is UNINIT, continue?" is displayed, press F4 [YES] to teach the position.

NOTE

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



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

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

2 The menu appears.

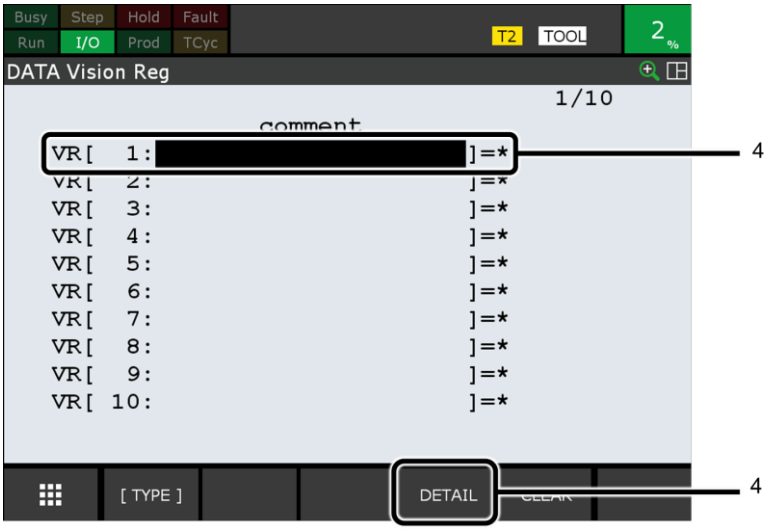


3 Select [DATA] → [Part Data VR] from the menu.

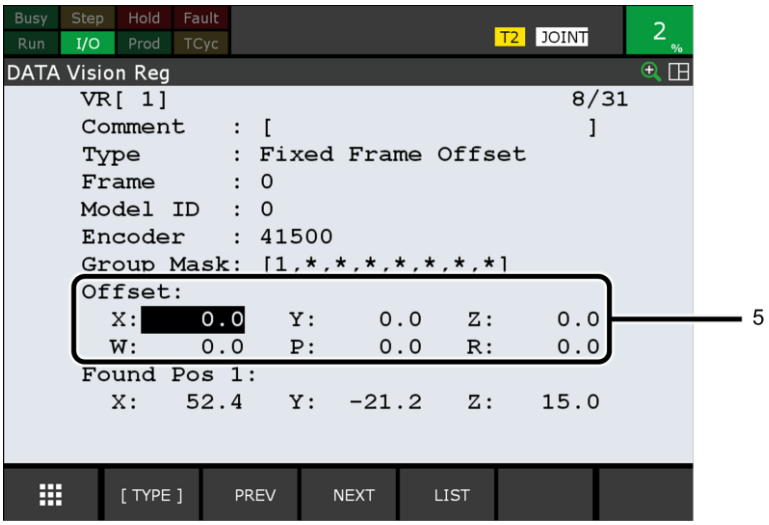


4 Place the cursor over '1' in [VR], then press F4 [DETAIL].

NOTE



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



Using a Position Register number for [Record found position on PR]

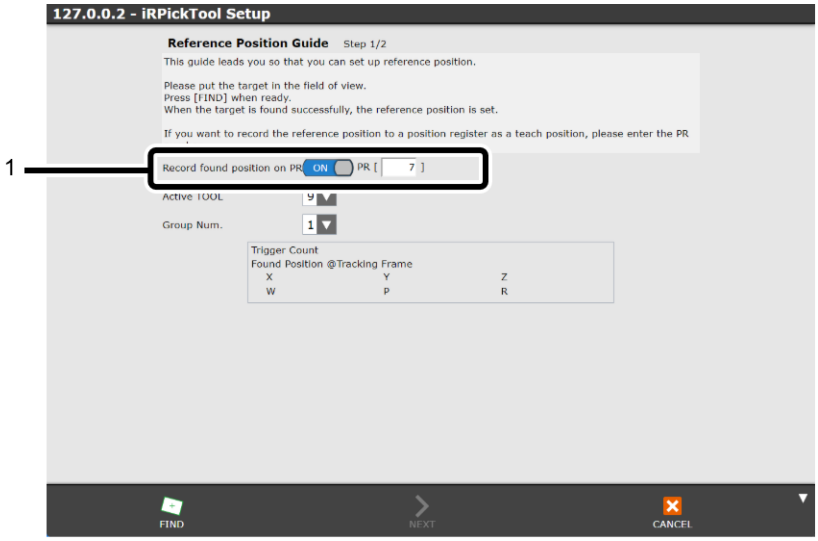
In the [Reference Position Guide Step 1/2] displayed in step 4, if a Position Register is entered for [Record found position on PR], the found position will be automatically set as the reference position in the Position Register, and the robot position teaching can be simplified.

This function can be used under the following conditions.

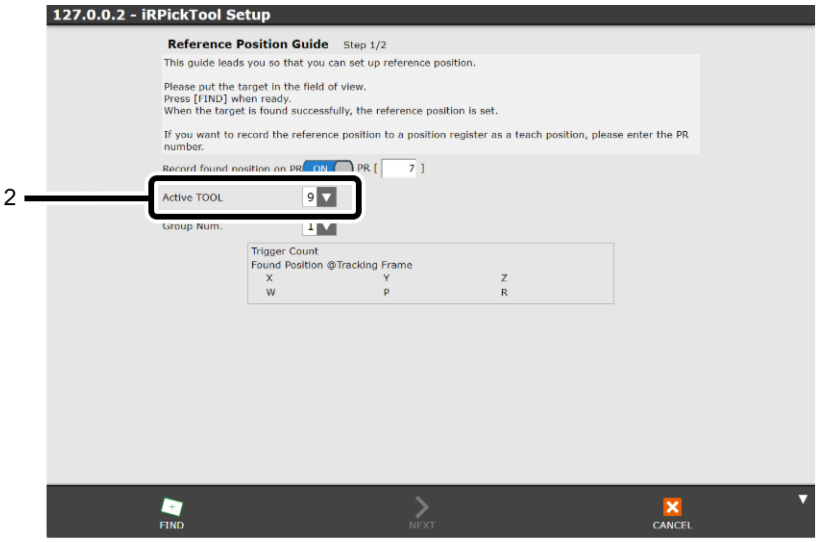
- The vision model origin is correct.
- The part Z height is correct.
- The tool frame setup for the hand is correct.

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

1 Set the [Record found position on PR] toggle switch to [Enable] in [Reference Position Guide Step: 1/2]. Enter the number of the Position Register to be used as the pick position in [PR]. Enter 7 if you are going to use position register [7] as in the sample program (PT_PICKCS) given above as a pick program.



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



CAUTION

The posture and configuration of the robot that will be stored in the position registers are determined from the current position of the selected tool. If a tool frame number that differs from the one that was set in "I.2.3.9. Setting up a tool frame for the hand(P.111)" is specified in [Active TOOL], the robot may not be in the intended posture or configuration, which may be dangerous. Carry out the operations with regard to the setup of the reference position with the utmost caution.

3 After moving the robot by jogging, adjust the tool position so that the robot will be in the posture for picking up a part.

CAUTION

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

4 Click [FIND].



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

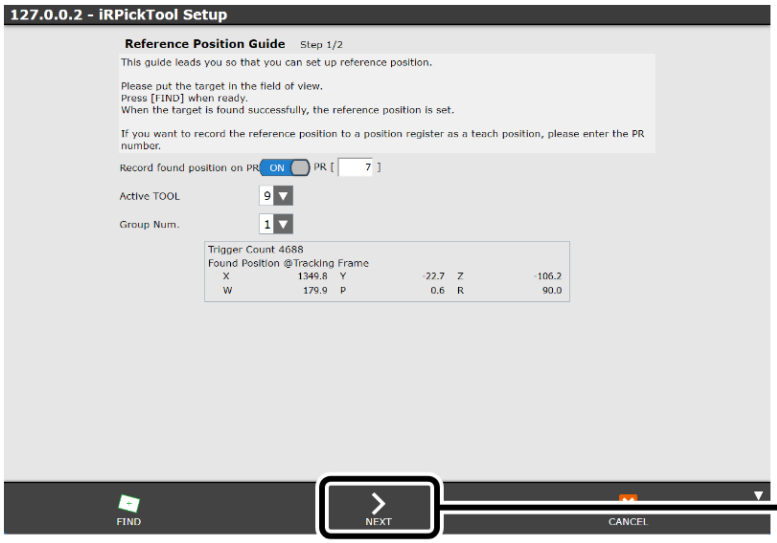
5 Check to see if detection has gone well using Runtime Image.

NOTE

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

6 Click [NEXT].

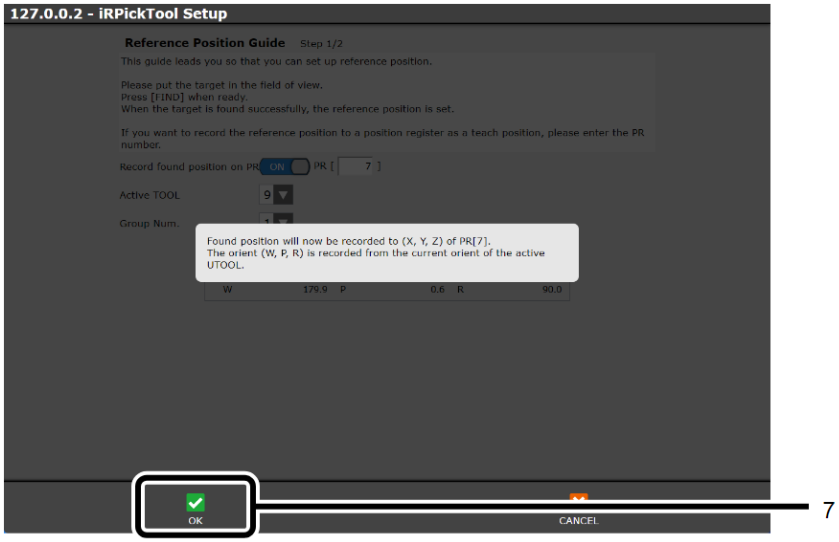
A message to confirm the position register that was set in step 1 will appear.



7 Click [OK].

NOTE

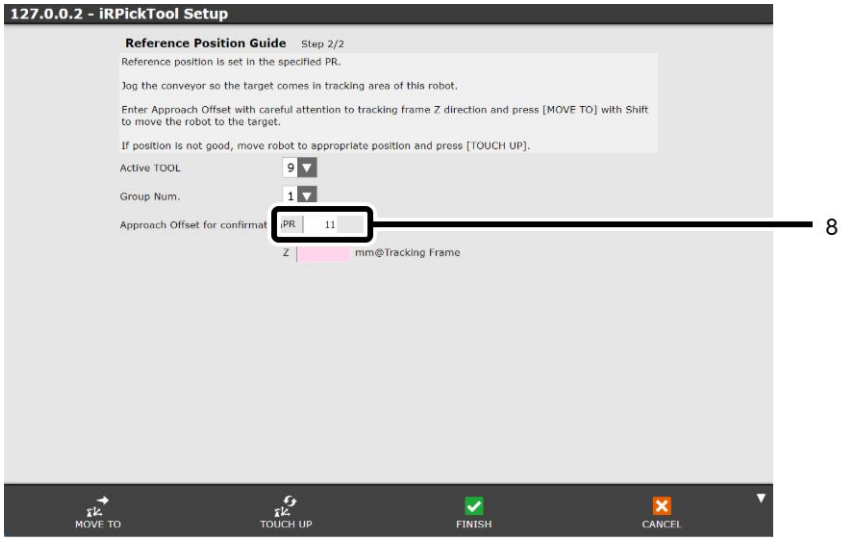
The currently selected tool frame number and group number need to match those selected on the screen.



The found position is recorded to the Position Register and [Reference Position Guide Step: 2/2] appears.

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

> **NOTE**
>
> It is assumed that [PR] will be used as a fixed frame offset for a tracking frame.



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

Move the conveyor until the part is directly in front of the robot.

Disable the alarm on the teach pendant. Then, on the iRPickTool setting screen displayed in the browser, hold down the [Shift] key and click [Move].

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



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

> **NOTE**
> In order for the tip point to touch up an accurate position, the model origin position for the vision process and the tool frame must be set up correctly.
> If the touch-up position is not correct, jog the robot to an appropriate position and click [Record]. When doing so, ensure that the currently selected tool frame number and group number match those selected on the screen.

11 Click [FINISH].



Setup of a reference position that uses a Position Register will finish.

[Guide Done] will appear in [Reference Position].

2.3.10.2. Setting up a reference position for the outfeed conveyor and teaching a robot position

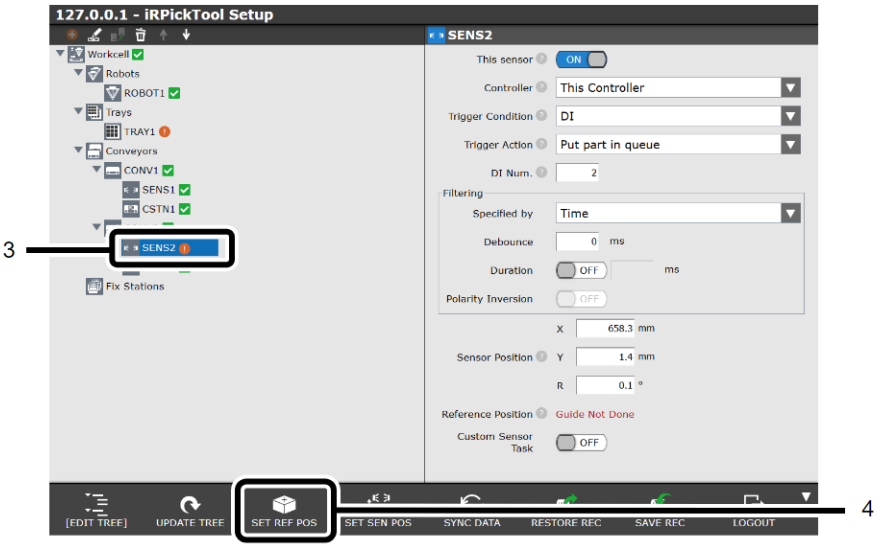
Set up a reference position for the outfeed conveyor, and teach a robot position. This procedure is for the case of [DI] / [RI] / [HDI] + [Put part in queue].

1 Open the robot homepage.

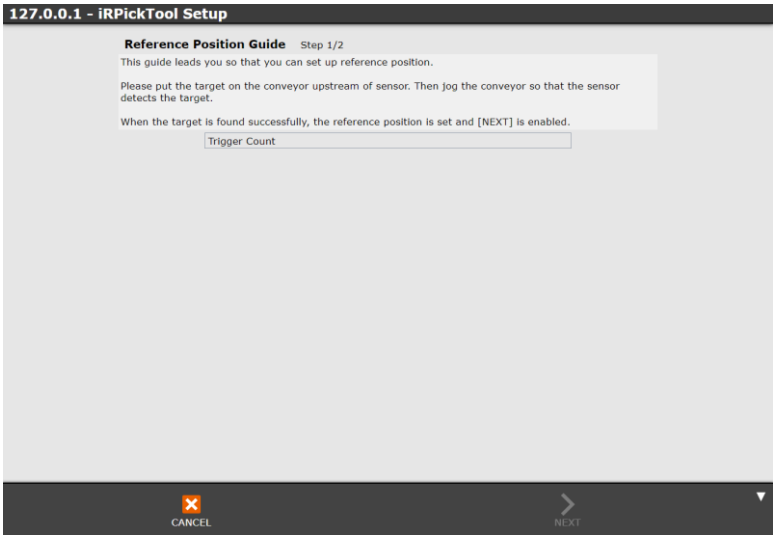
2 Select [iRVision/iRPickTool] in the menu area and click [iRPickTool Workcell Setup] in the content display area.



3 Select [SENS2] for the sensor, and click [SET REF POS].



After a while, the following screen appears.



4 Place a tray upstream of the trigger sensor.

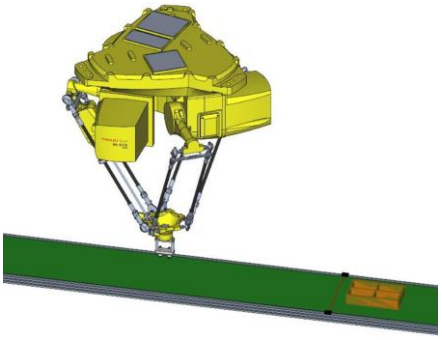


Fig. 2.3.10.2-1 Placing a tray upstream of the trigger sensor.

5 Move the conveyor. When the tray comes within the operating range of the robot, stop the conveyor.

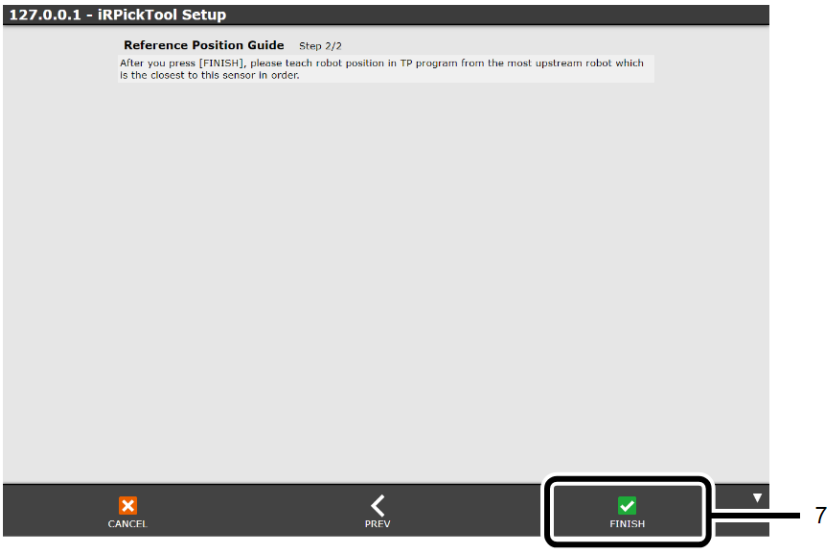
If detection is successful, [Trigger Count] will appear.

6 Click [NEXT].

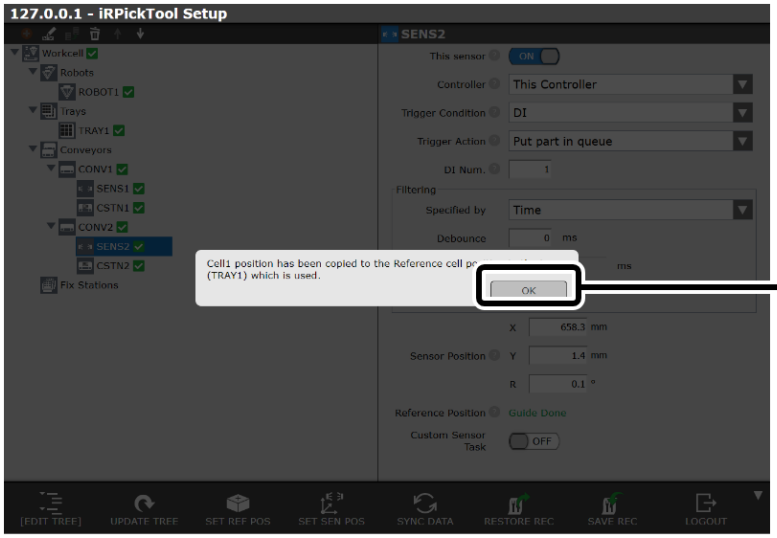
If detection succeeds, [NEXT] will become active.



7 Click [FINISH].

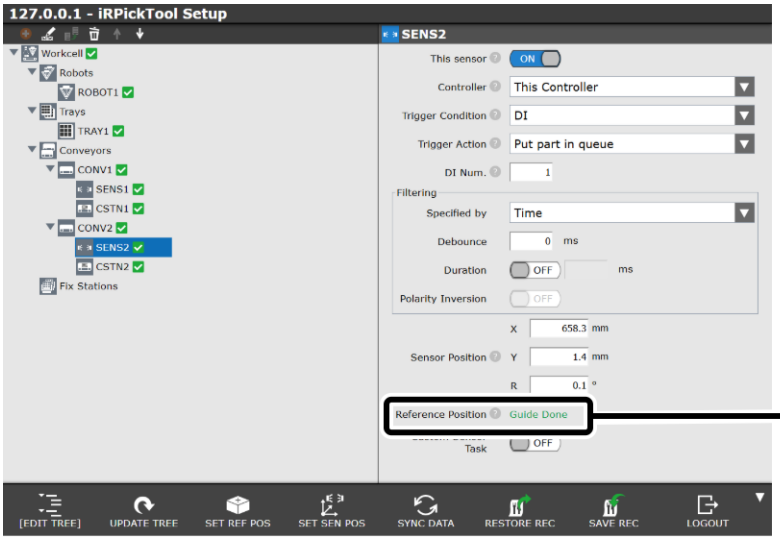


8 The following message will appear. Click [OK].



After a while, the following screen appears.

Check that [Guide Done] appears in [Reference position].



9 Move the conveyor. When the tray comes within the operating range of the robot, stop the conveyor.

* Be careful that the tray does not become misaligned.

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

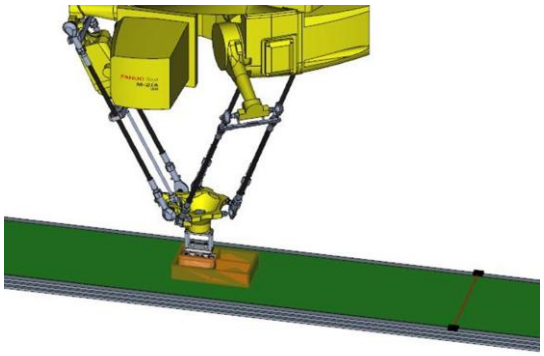


Fig. 2.3.10.2-2 Moving the robot to the drop position of cell 1 on the tray

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

12 While holding down the [SHIFT] key, press F5 [TOUCHUP].

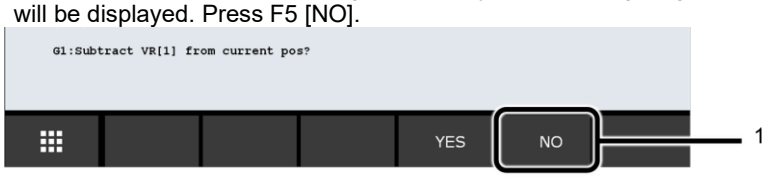
If F5 [TOUCHUP] is not displayed, press [> (NEXT)] to switch screens.



A message saying “VR is UNINIT, continue?” will be displayed, but press F4 [YES] to teach the position.

NOTE

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



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

1 Press the [MENU] key on the robot controller’s teach pendant.
The menu appears.

2 Select [NEXT] from the menu.

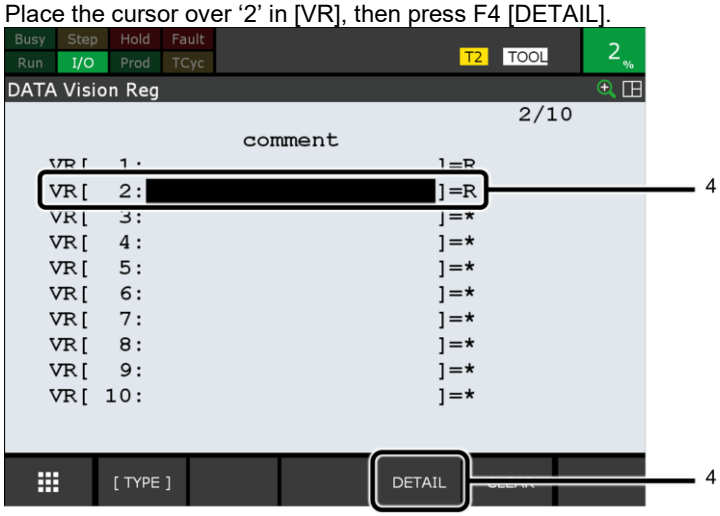


3 Select [DATA] → [Vision Reg] from the menu.

NOTE



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



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

2.3.11. Fine adjustment of the tracking motion

In systems that require precision, fine adjustments are made to the 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 adjustment procedures for each situation are described in the following section.

2.3.11.1. Checking and fine adjustment of the tracking motion

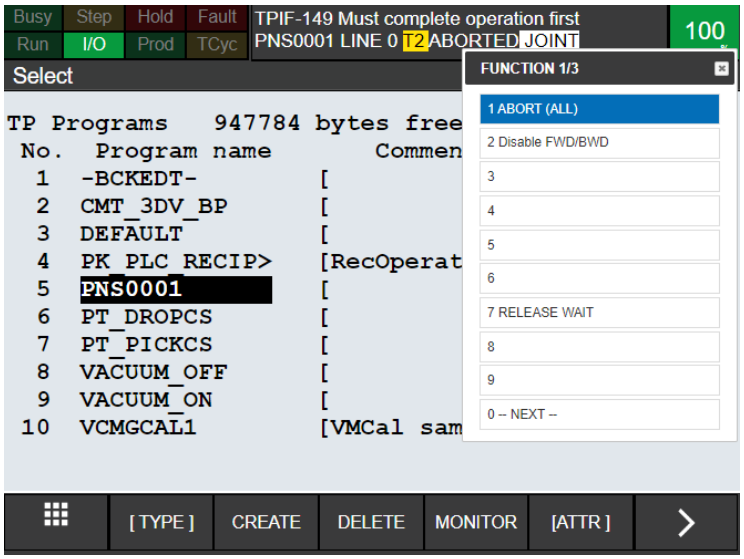
After teaching the robot position, adjust the tracking motion first. After moving the conveyor with the workpiece to the robot motion range, stop the conveyor to check the robot pick (place) position and make a minor adjustment to the teach position. Repeat this adjustment until there is no offset in each position. This adjustment should be performed by each robot.

The workpiece should be placed on the conveyor with the same phase as taught.

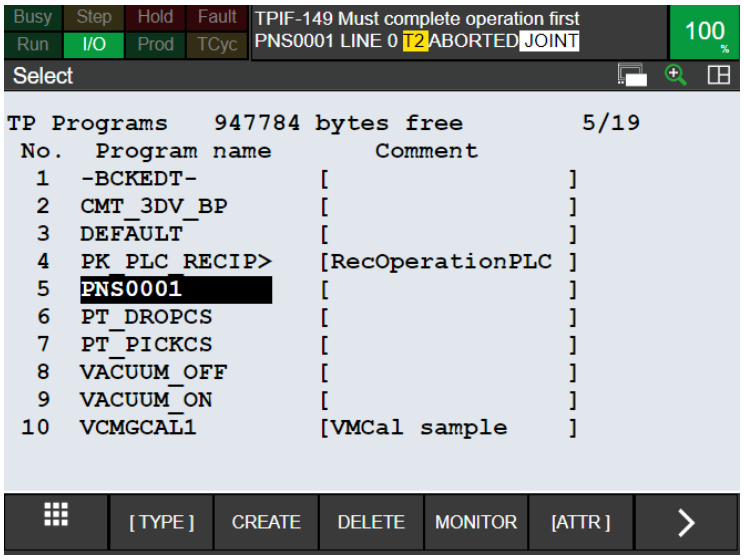
If you use a camera, you can skip this section as the same steps are described in the next section, "I.2.3.11.2. Adjustment of the tracking frame error(P.131)".

Checking the tracking motion

1 Press [FCTN] on the teach pendant, and select [1 ABORT (ALL)].



2 Press [SELECT] to display the program list screen.



3 Select [Main Program (PNS0001)] and press [ENTER].

4 Teach the perch position for all groups (PR[1: HOME POS]).



5 **Press [STEP] to enter step execution mode.**

6 **Place the cursor over the first line and press [SHIFT] and [FWD] to execute the program.**

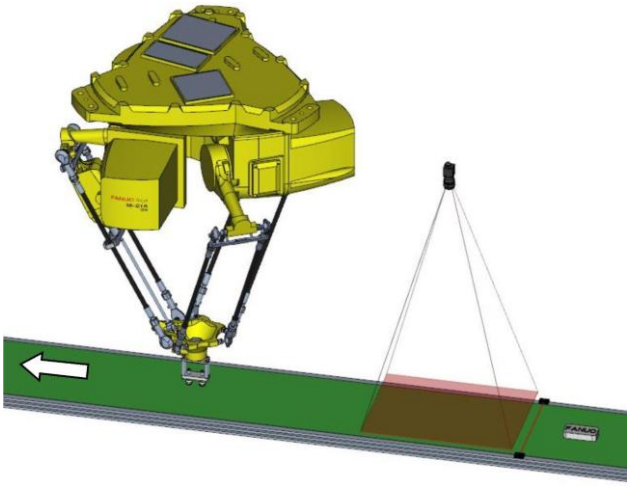
This will loop the program on the LBL[101:QUE check] of the pick program (PT_PICKCS).

```
 1: UFRAME_NUM=0
 2: UTOOL_NUM=1
 3:
 4: LBL[101:QUE check]
 5: STOP_TRACKING
 6: !PKCSGETQUE (Get part information)
 7: !*1:Register number for conveyor station ID
 8: !*2:Usually 1. 2 or more in the case of consecutive acquisition
 9: !*3:Index delay msec
10: !*4:Vision Register number for part information
11: !*5:Process status register number
12: !*6:Model ID (Can be omitted)
13: CALL PKCSGETQUE
      (CStn ID=R[11:CONV1 ID],
      Consec Flag=1,Timeout(ms)=100,
      Offset VR=1,Stat Reg=2)
14: IF R[2:GETQ Status Pick]=0,JMP LBL[111]
15: IF DI[1:CYCLE_STOP]=ON,JMP LBL[901]
16: JMP LBL[101]
...
```

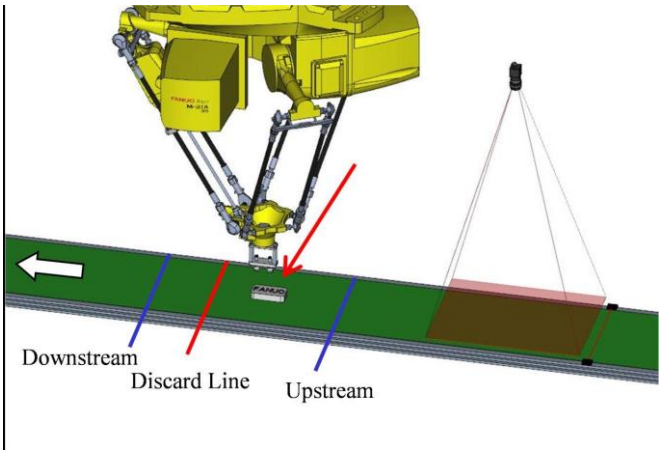
7 **Take your hand off the teach pendant of the robot and prepare the workpiece.**

8 **Set the workpiece upstream from the trigger sensor.**

Set the workpiece to be the same phase as the reference position previously set.



9 **Move the conveyor, and stop it when the workpiece enters the robot motion range (between the upstream boundary and the discard line).**



10 **Press [SHIFT] and [FWD] and then resume the program where the pick program (PT_PICKS) stops.**

The speed override should be slow at around 2%.

11 **Execute up to the motion program of the pick position, and check if there is any offset between the workpiece and the hand.**

If no offset is found, the tracking motion check is complete.

If any offset occurs, adjust the teach position according to the following procedures.

Adjust the teach position

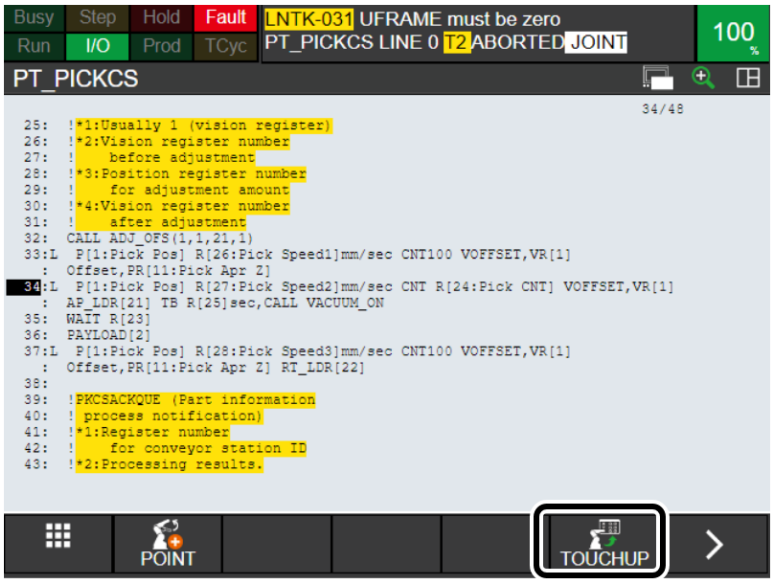
1 **Move the robot by jog operation.**

2 **Place the cursor over the line where you teach the pick position.**

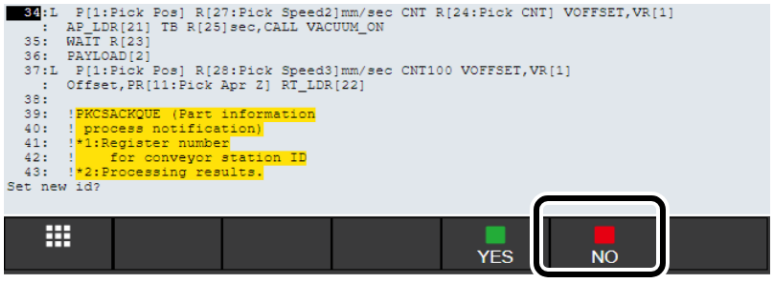
```
...
25: L PR[7:Pick Pos]
      R[26:Pick Speed1]mm/sec
      CNT100 VOFFSET,VR[1]
      Offset,PR[11:Pick Apr Z]
26: L PR[7:Pick Pos] R[27:Pick Speed2]mm/sec
      CNT R[24:Pick CNT]
      VOFFSET,VR[1]
      AP_LDR[21]
      TB R[25]sec, CALL VACUUM_ON
...
```

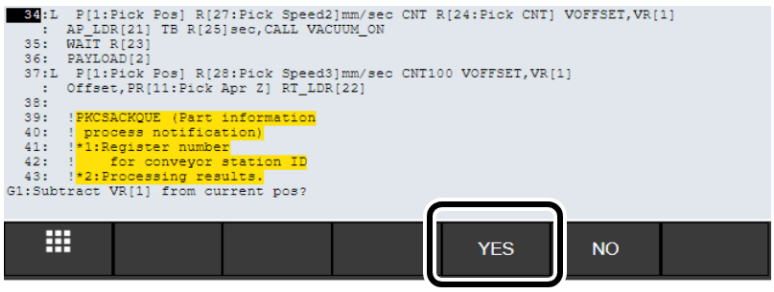
3 **Press [SHIFT] and [TOUCHUP].**

Switch the display by pressing [F->] if [TOUCHUP] is not displayed.



4 When the message “Set new ID?” appears, press F5 [NO]. When the message “Subtract VR from current pos?” appears, press F4 [YES].





5 Check the tracking again from step 1 of “Checking the tracking motion”. Teaching the home position in step 4 is not necessary.

> **⚠CAUTION**
>
> When you adjust the teach position, be sure to perform step 1 through step 11 of “Checking the tracking motion” to finish the procedures.

2.3.11.2. Adjustment of the tracking frame error

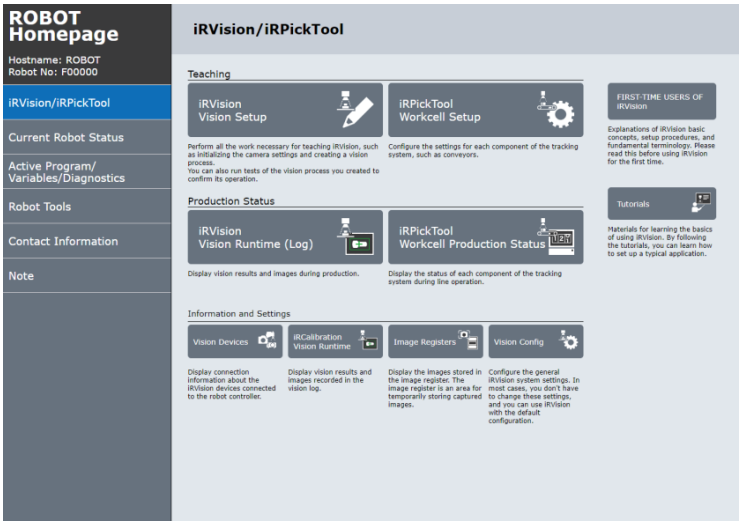
When a camera is used to detect parts, the tracking motion is susceptible to error if the tracking frame recognized by the camera does not match that recognized by each individual robot. This error manifests as the robot being unable to pick a rotated workpiece at the correct position even though it can pick a non-rotated workpiece at the correct position.

This error occurs because the tracking frame is not set up correctly. The best way would be to start over from setting the tool frame and the tracking frame, but it can be improved by adding the amount of adjustment to the offset data calculated with iRVision in the following procedure. Make this adjustment for each robot individually.

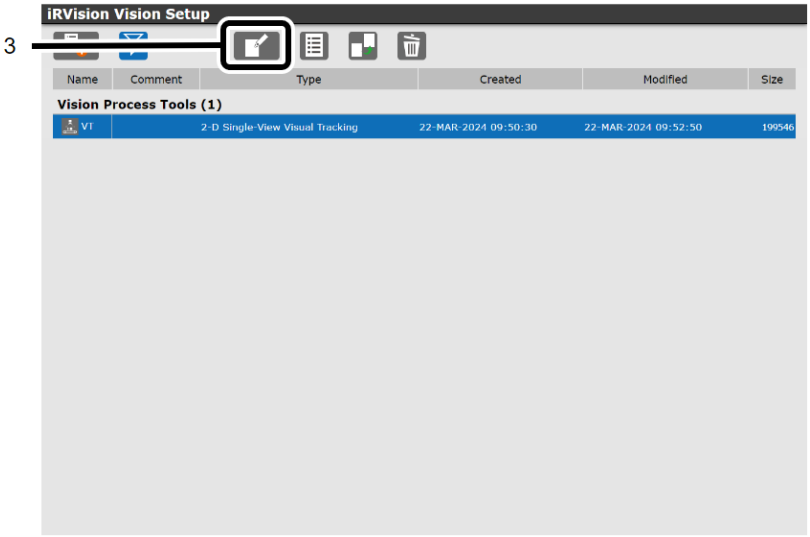
Before you start this adjustment, make sure that the robot can pick a workpiece at the correct position if a workpiece has the same phase as taught.

1 **Open the robot homepage.**

2 **Select [iRVision/iRPickTool] in the menu area and click [iRVision Vision Setup] in the content display area.**



3 **Select a vision process and click [Edit].**



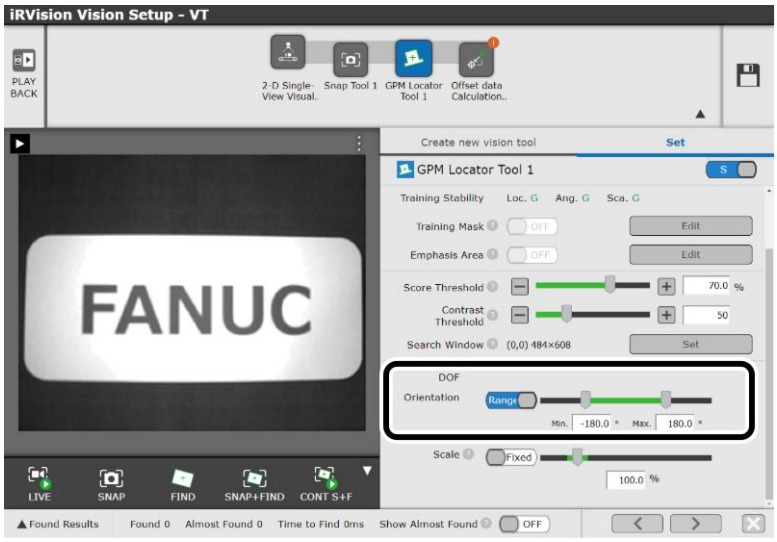
4 **Select [GPM Locator Tool 1] in Program Line.**



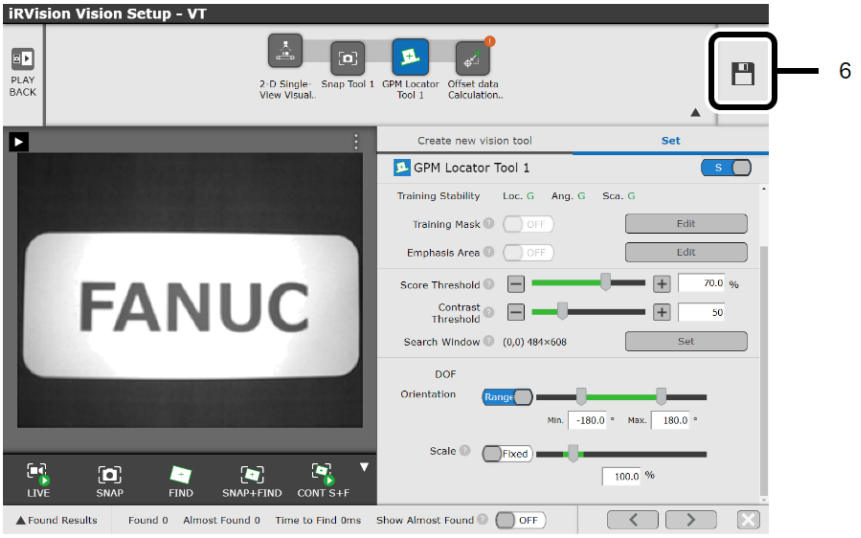
5 Set the DOF of the orientation to [Min.] -180 degrees, [Max.] 180 degrees.

If the model does not have the angle of ± 180 degrees (like a square or rectangle), set the DOF to [Min.] 170 degrees, [Max.] 190 degrees. When the reference position is around 0 degrees, the model will be force detected around 180 degrees.

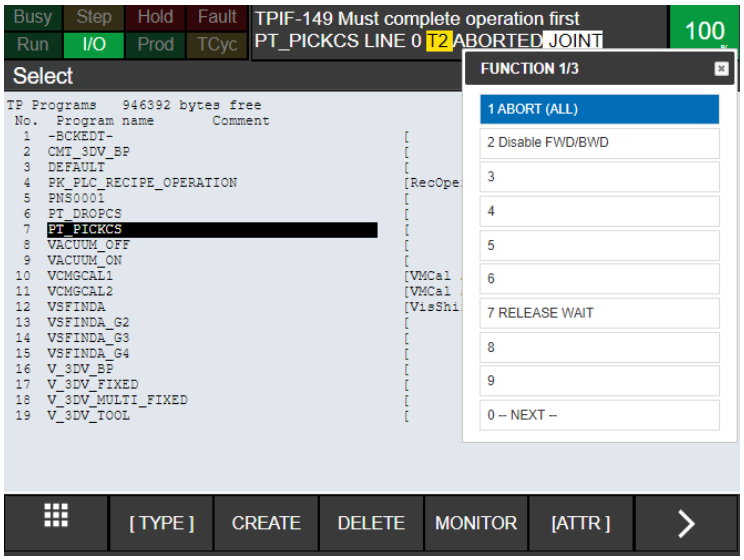
Take note of the values before the change as they need to be put back to their original settings after this adjustment.



6 Click [SAVE] and click [×].

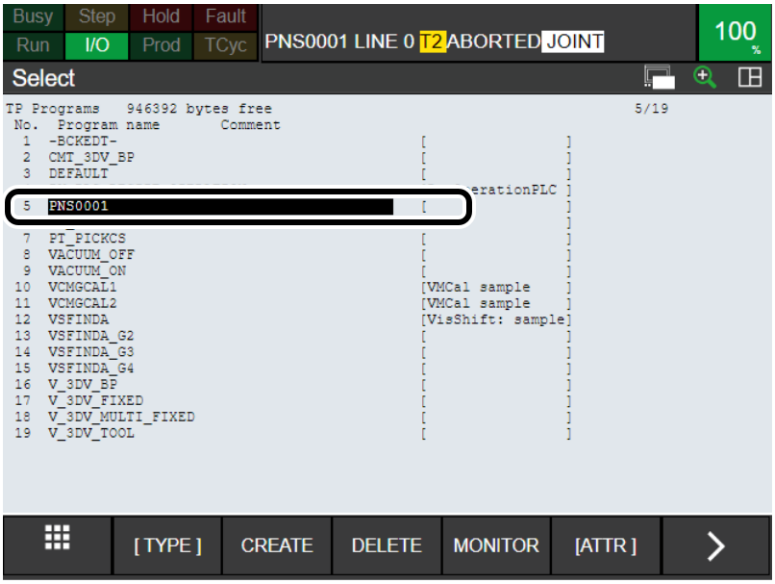


7 Press [FCTN] and select [1 ABORT (ALL)].



8 Press [SELECT] to display the program list screen.

9 Select [Main program (PNS0001)] and press [ENTER].



10 Press [STEP] to enter step execution mode.

11 Place the cursor over the first line and press [SHIFT] and [FWD] to execute the program.

This will loop the program on the LBL[101:QUE check] of the pick program (PT_PICKCS).

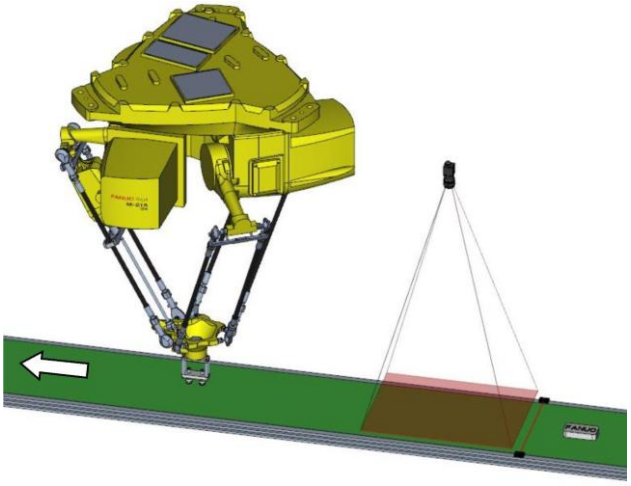
```
 1: UFRAME_NUM=0
 2: UTOOL_NUM=1
 3:
 4: LBL[101:QUE check]
 5: STOP_TRACKING
 6: !PKCSGETQUE (Get part information)
 7: !*1:Register number for conveyor station ID
 8: !*2:Usually 1. 2 or more in the case of consecutive acquisition
 9: !*3:Index delay msec
10: !*4:Vision Register number for part information
11: !*5:Process status register number
12: !*6:Model ID (Can be omitted)
13: CALL PKCSGETQUE
      (CStn ID=R[11:CONV1 ID],
      Consec Flag=1,Timeout(ms)=100,
```

```
     Offset VR=1,Stat Reg=2)
14: IF R[2:GETQ Status Pick]=0,JMP LBL[111]
15: IF DI[1:CYCLE_STOP]=ON,JMP LBL[901]
16: JMP LBL[101]
...
```

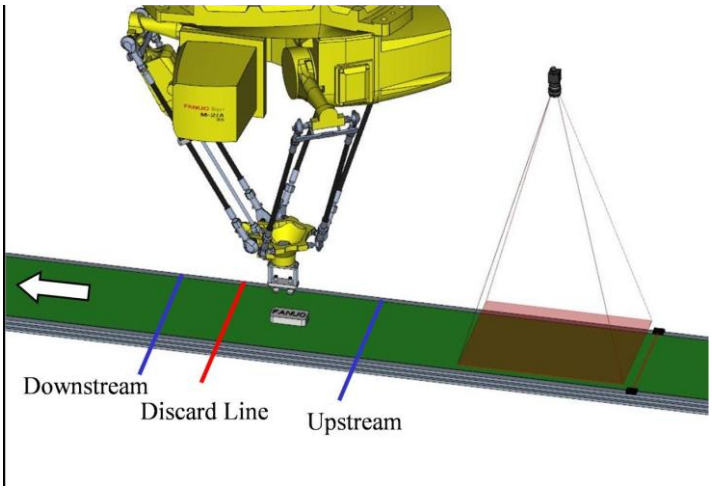
12 Take your hand off the teach pendant of the robot and prepare the workpiece.

13 Set the workpiece upstream of the trigger sensor or the camera.

Set the workpiece 180 degrees in the opposite direction from the reference position previously set.



14 Move the conveyor, and stop it when the workpiece enters the robot motion range (between the upstream boundary and the discard line).



15 Press [SHIFT] and [FWD] and then resume the program where the pick program (PT_PICKS) stops.

The speed override should be slow at around 2%.

⚠CAUTION
Be careful when handling the air piping and cables, as the axis of the robot wrist rotates 180 degrees.

16 Execute up to the motion program of the pick position, and check if there is any offset between the workpiece and the hand.

The tracking motion check is complete if no offset is found.

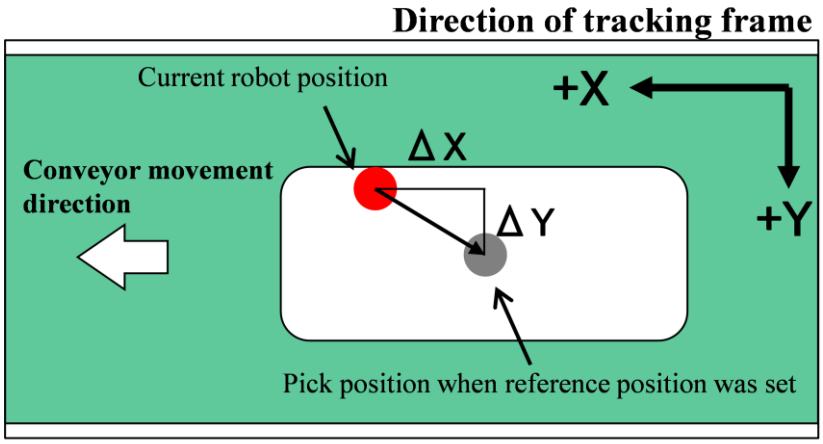
Restore the DOF values of the orientation changed in step 5 to the original values.

If any offset occurs, adjust it in the following procedures.

17 Measures the amount of offset (ΔX, ΔY).

Check the direction from the current position to the pick position when the reference position was set.

The following figure shows that ΔX has a negative value and ΔY has a positive value.



+X direction of the tracking frame accords with the conveyor movement direction

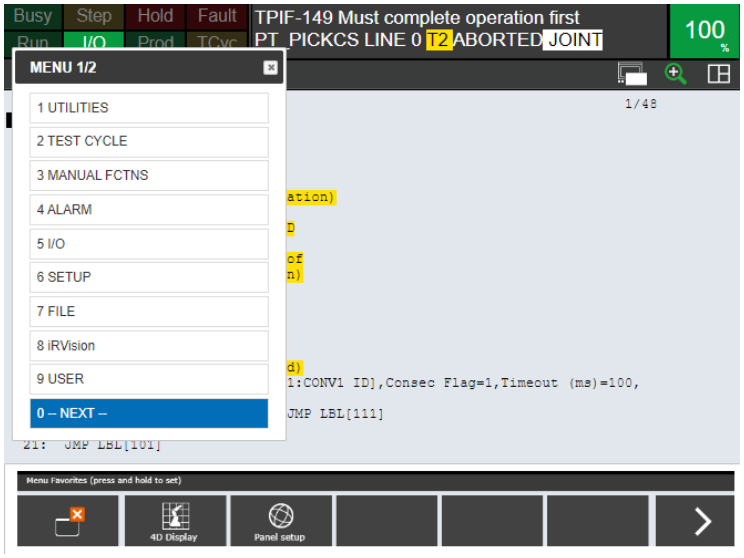
18 Check argument 3 of the robot program "CALL ADJ_OFS".

"21" for the sample program.

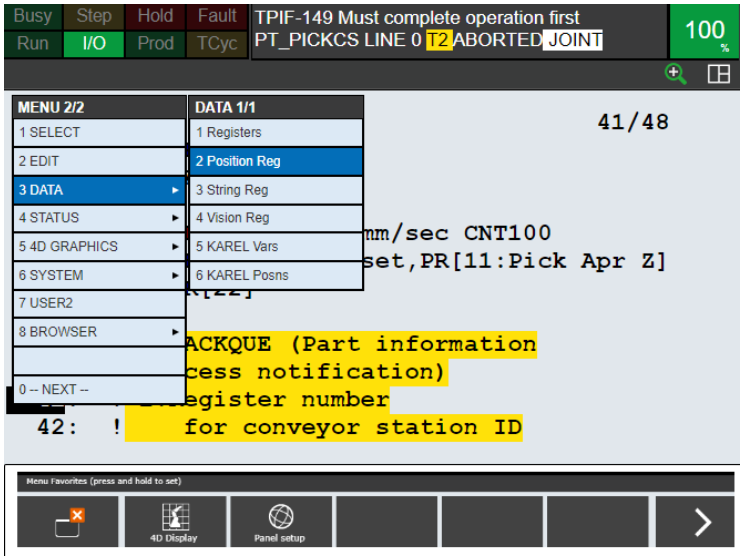
If the sample program is not used, add ADJ_OFS as described in "I.2.2.5. Creating a robot program(P.42)".

```
...
19: !ADJ_OFS (Adjust offset)
20: !*1:Usually 1 (vision register)
21: !*2:Vision Register number before adjustment
22: !*3:Position Register number for adjustment amount
23: !*4:Vision Register number after adjustment
24: CALL ADJ_OFS(1,1,21,1)
...
```

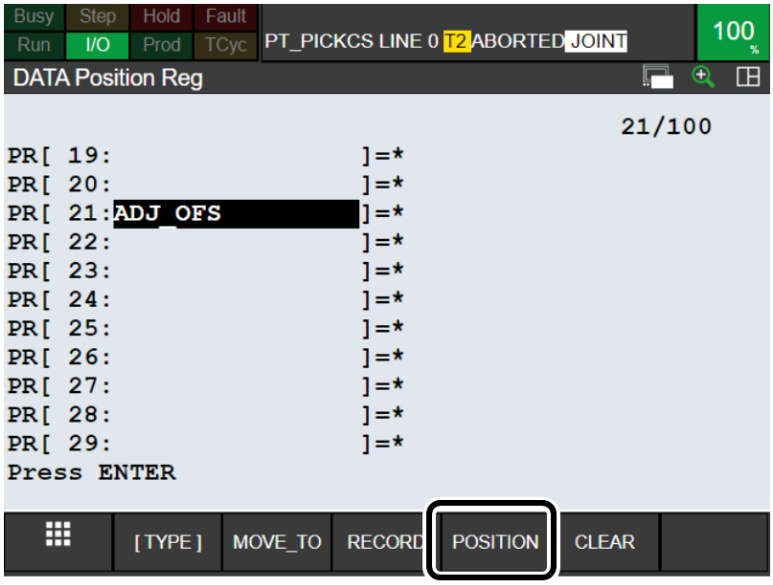
19 Press [MENU] in the teach pendant, select [0 - NEXT --] and then press [ENTER].



20 Select [3 DATA] → [Position Reg].



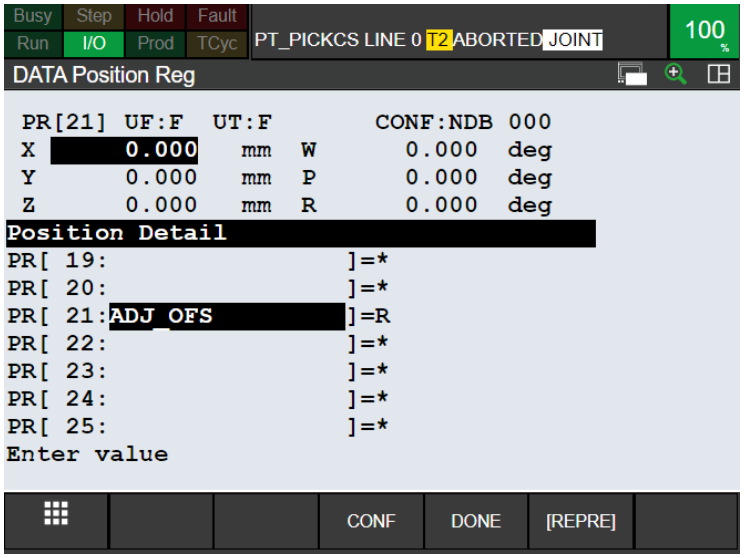
21 **Place the cursor over [PR[21:ADJ_OFS]], and press F4 [POSITION].**



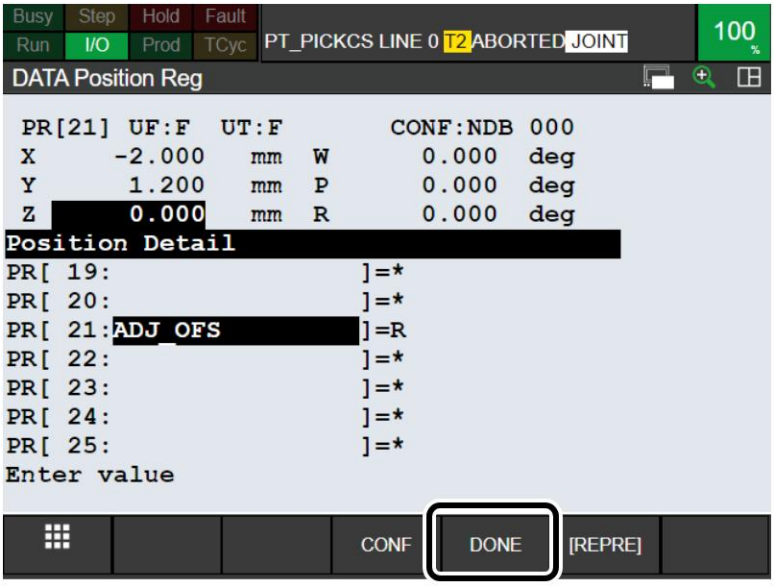
22 **Enter the half value of the offset amount in X, Y and press [ENTER].**

(X,Y) = (ΔX/2 , ΔY/2)

Refer to step 17 for the sign of each value.



22 **Press F4 [DONE].**



23 **Repeat the check process from step 7.**

> **⚠ CAUTION**
>
> When you adjust the setting error of the tracking frame, be sure to start checking from step 7 and finish with step 16.

About ADJ_OFS

ADJ_OFS is a KAREL program that only adds the specified amount of adjustment to the offset data stored in the Vision Register. The meanings of its arguments are described below. See the section “Vision support tools” in the chapter “Other options” in "iRVision OPERATOR'S MANUAL (Reference)"(B-84434EN).

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.

2.3.11.3. 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 "I.2.3.11.1. Checking and fine adjustment of the tracking motion(P.127)" to "I.2.3.11.2. Adjustment of the tracking frame error(P.131)".
2. If the robot works on multiple conveyors, adjust the error on the conveyor with the fastest speed.
3. The following procedures are for when the T2 mode is available.

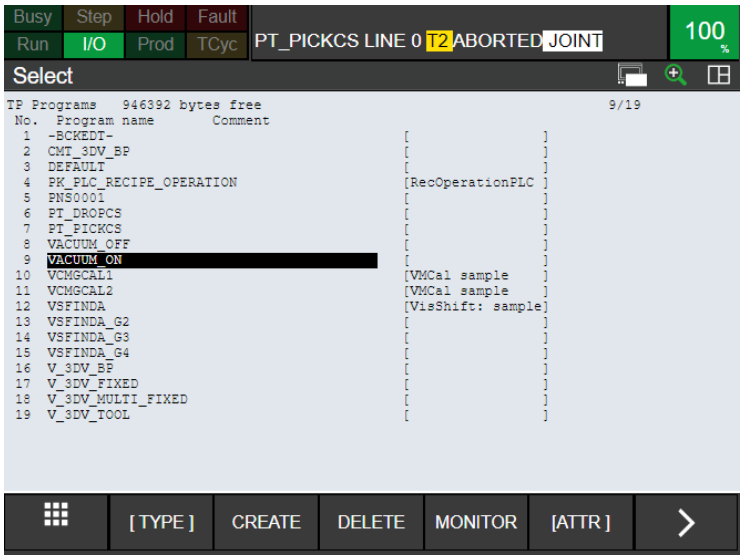
Adjust this error in the following procedures.

Comment out of tool control program

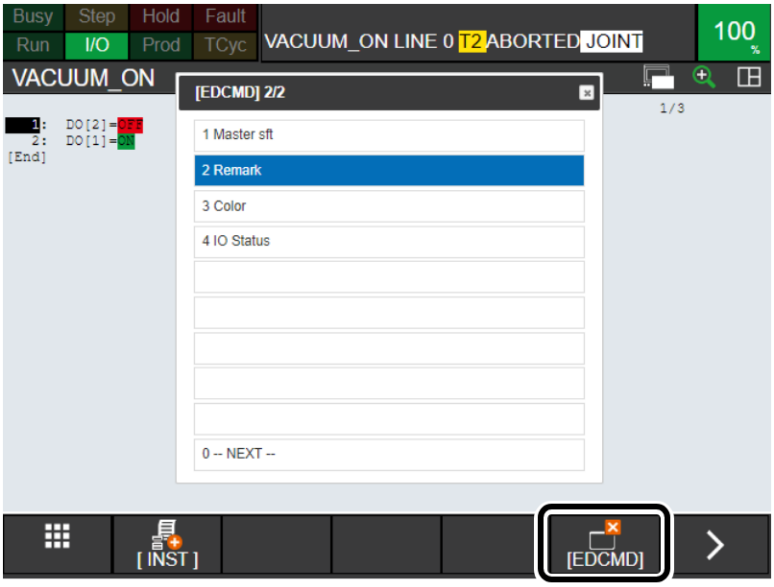
1 Press [SELECT] to display the program list screen.

2 Select the tool control program to adjust the error, and press [ENTER].

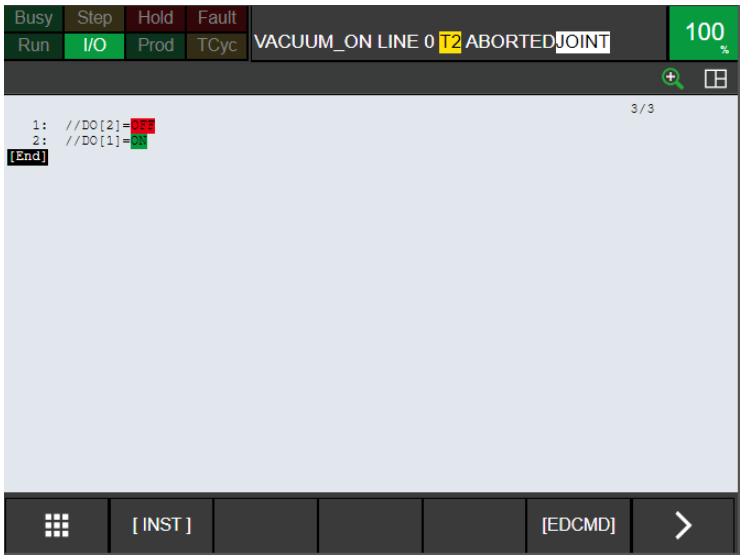
With the tool control program, select [VACUUM ON] for pickup motion, and [VACUUM OFF] for placement motion.



3 Place the cursor over the first line and press [Remark] in the edit commands of the icon editor.



4 Select the range to comment out and press F4 [REMARK]. '//' is added in front of the instruction specified. The lines beginning with '//' are comments.

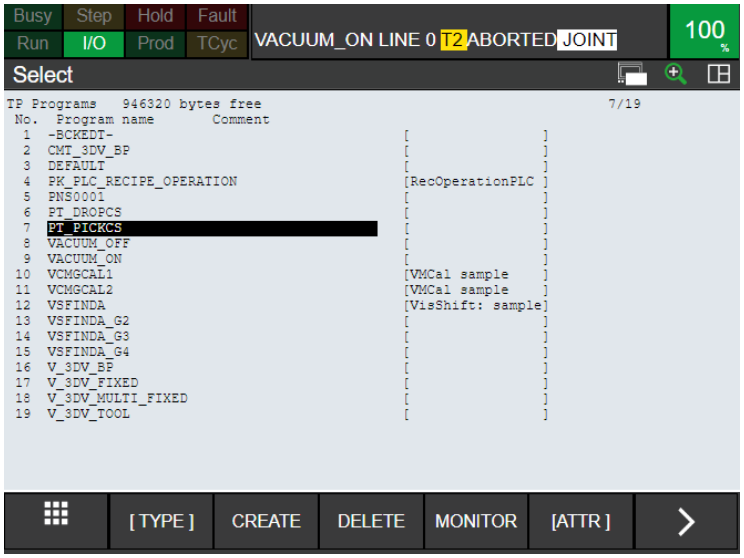


Changing the program

1 Press [SELECT] to display the program list screen.

2 Select the tracking program to adjust the error and press [ENTER].

With the tracking program, select [PT_PICKCS] for picking motion, and [PT_DROPCS] for placement motion.



3 Place the cursor over the end of the motion instruction (the end of the time before function) of the pick (place) position.

```
...
25: L PR[7:Pick Pos]
      R[26:Pick Speed1]mm/sec
      CNT100 VOFFSET,VR[1]
      Offset,PR[11:Pick Apr Z]
26: L PR[7:Pick Pos] R[27:Pick Speed2]mm/sec
      CNT R[24:Pick CNT]
      VOFFSET,VR[1]
      AP_LDR[21]
      TB R[25]sec,CALL VACUUM_ON
27: WAIT R[23]
...
```

4 Press F4 [CHOICE] and select [Offset, PR[ ]].

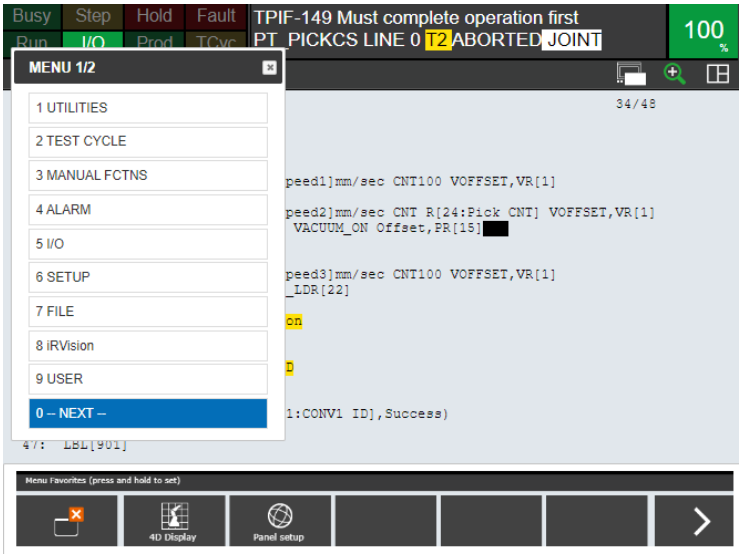
Offset, PR[ ] is added.

5 Enter '15' for the number of Offset, PR[ ].

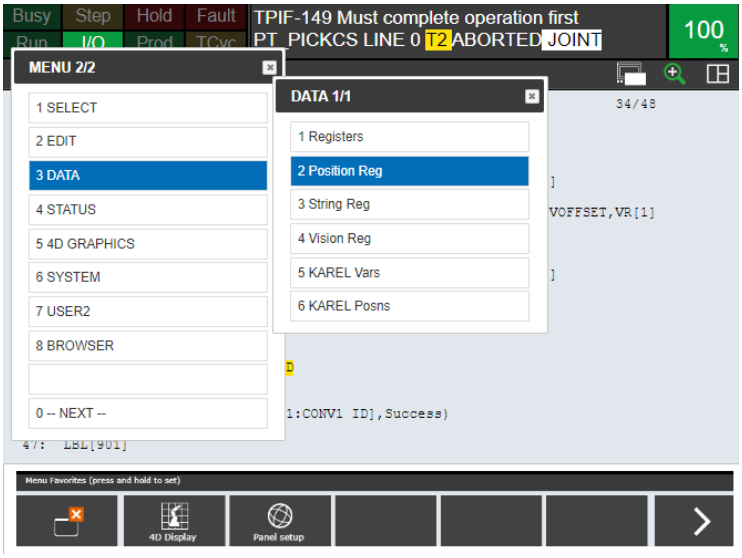
```
...
25: L PR[7:Pick Pos]
      R[26:Pick Speed1]mm/sec
      CNT100 VOFFSET,VR[1]
      Offset,PR[11:Pick Apr Z]
26: L PR[7:Pick Pos] R[27:Pick Speed2]mm/sec
      CNT R[24:Pick CNT]
      VOFFSET,VR[1]
      AP_LDR[21]
      TB R[25]sec, CALL VACUUM_ON
      Offset,PR[15]
27: WAIT R[23]
...
```

Setting the offset value

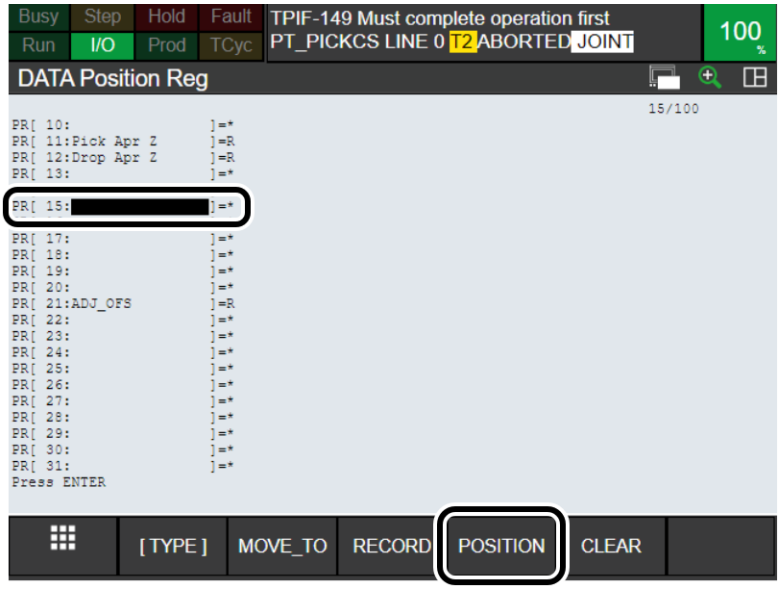
1 Press [MENU], select [0 - NEXT --] and press [ENTER].



2 **Select [3 DATA] → [Position Reg].**



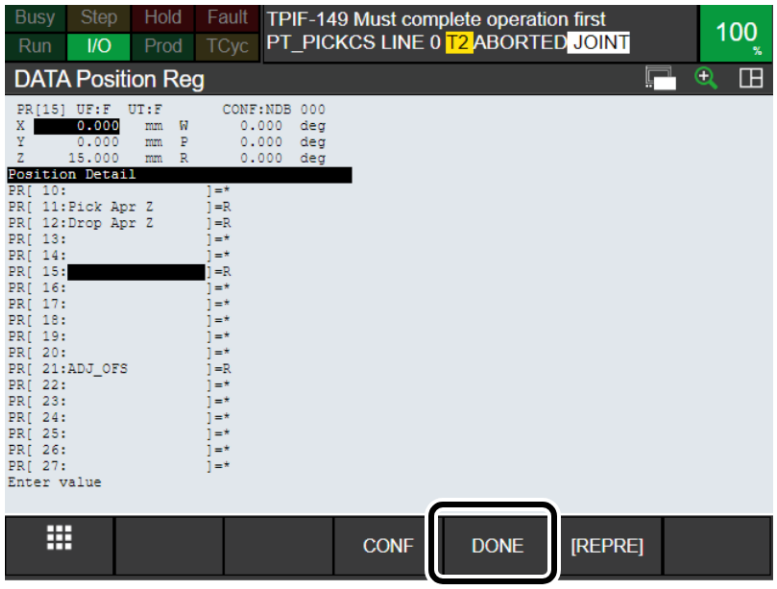
3 **Place the cursor over PR[15] and press F4 [POSITION].**



4 **Enter the value in the Position Register. Enter '0' for all items other than Z. For Z, enter the plus value for offset so that the robot will be positioned slightly upstream of the workpiece.**

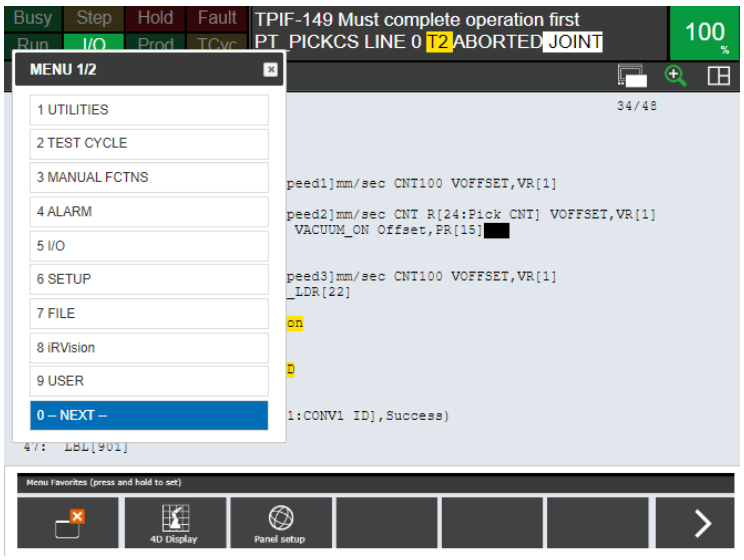


5 **After entering the values, press F4 [DONE].**

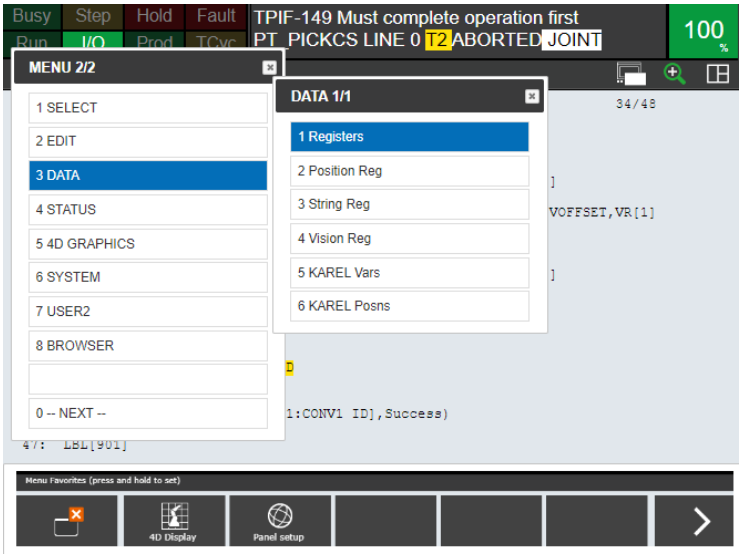


Changing the timer setting

1 **Press [MENU], select [0 - NEXT --] and press [ENTER].**

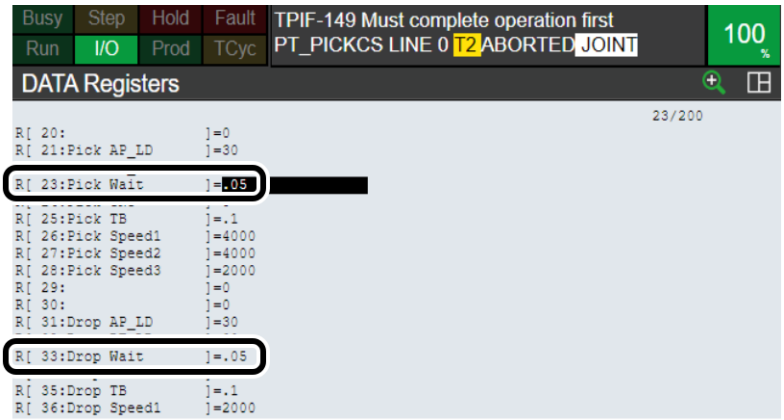


2 **Select [3 DATA] → [Registers].**



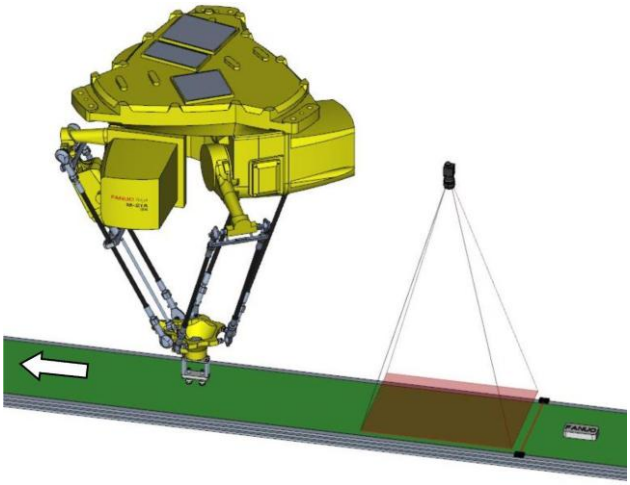
3 **Place the cursor over R[23:Pick Wait] to adjust with pick program, [33:Drop Wait] to adjust with the placement program and then enter the time for tracking the workpiece in seconds. At first, enter a value of around '3' (it depends on the speed of the conveyor among other things).**

Take note of the values before the change as they need to be put back to their original settings after this adjustment.



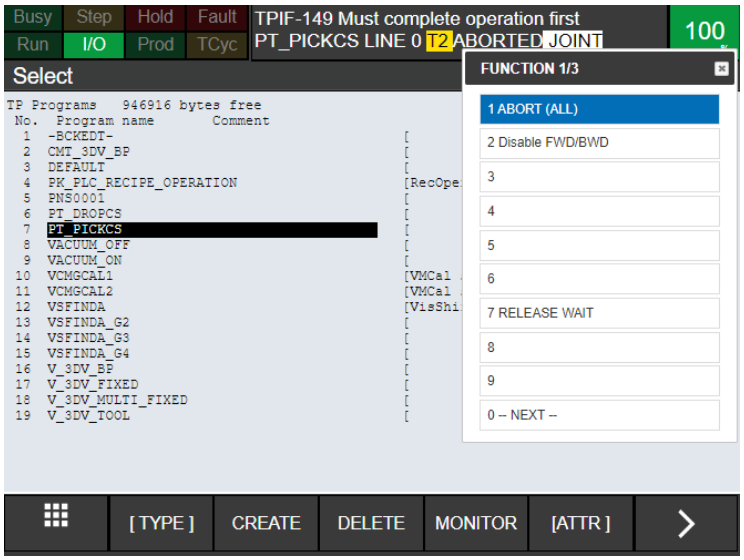
4 **Set the workpiece upstream of the trigger sensor or the camera.**

Set the workpiece to be the same phase as the reference position previously set.



Executing the program

1 Press [FCTN] and select [1 ABORT (ALL)].



2 Press [SELECT] to display the program list screen.

3 Select PNS0001 [MAIN PROGRAM] and press [ENTER].



4 Switch the mode switch to T2 of the robot controller.

5 Press [STEP] to release the step execution mode. Raise override to 100%.

Follow the procedures below if you feel uneasy about suddenly executing the program with 100% override.

- Perform the steps with the same procedures as "I.2.3.11.1. Checking and fine adjustment of the tracking motion(P.127)"
- Raise the override value gradually

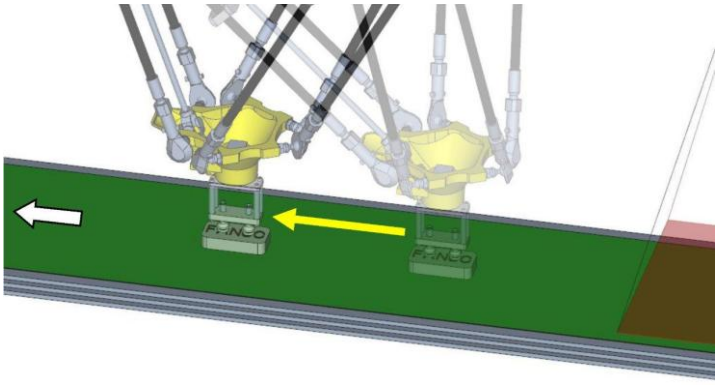
6 Place the cursor over the first line and press [SHIFT] and [FWD] to execute the program.

This will loop the program on the LBL[101:QUE check] of the pick program (PT_PICKCS).

```
 1: UFRAME_NUM=0
 2: UTOOL_NUM=1
 3:
 4: LBL[101:QUE check]
 5: STOP_TRACKING
 6: !PKCSGETQUE (Get part information)
 7: !*1:Register number for conveyor station ID
 8: !*2:Usually 1. 2 or more in the case of consecutive acquisition
 9: !*3:Index delay msec
10: !*4:Vision Register number for part information
11: !*5:Process status register number
12: !*6:Model ID (Can be omitted)
13: CALL PKCSGETQUE
      (CStn ID=R[11:CONV1 ID],
      Consec Flag=1,Timeout(ms)=100,
      Offset VR=1,Stat Reg=2)
14: IF R[2:GETQ Status Pick]=0,JMP LBL[111]
15: IF DI[1:CYCLE_STOP]=ON,JMP LBL[901]
16: JMP LBL[101]
```

...

7 **Keep your hand on the teach pendant of the robot and move the conveyor. The robot will track the workpiece during the time set in step 3 of "Changing the timer setting".**

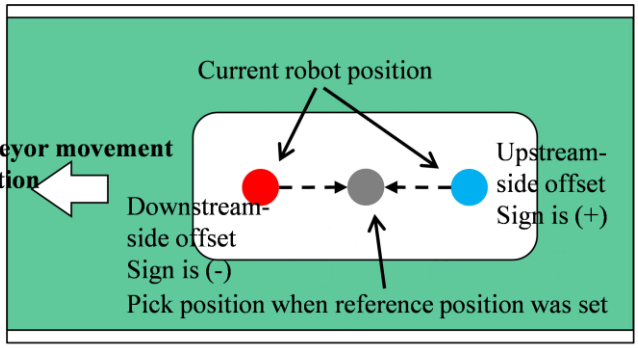


8 **Check the amount of offset visually. If there is no offset, this completes the adjustment of the robot motion error.**

Revert to the original value which was changed for error adjustment. For the procedures to revert to the original values, refer to "Uncomment the tool control program", "Revert the program to the original setting", "Remove the offset", and "Revert the timer setting" in the latter part of this section.

If offset exists, adjust with the following procedures.

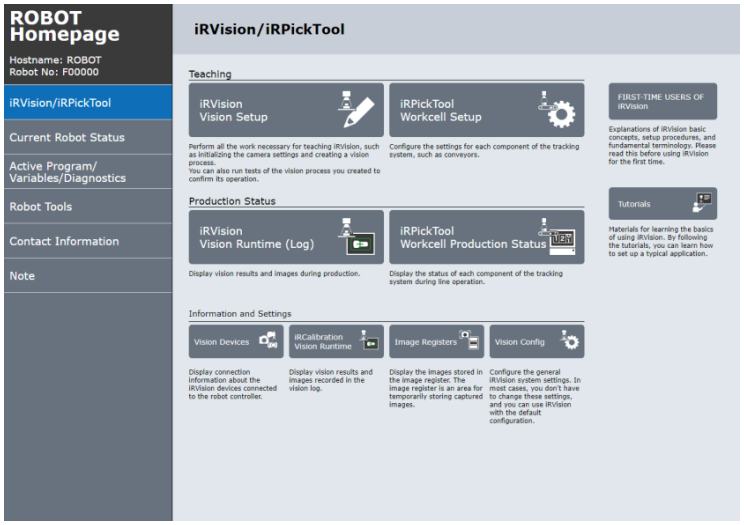
9 **Refer to the figure below and calculate the amount of offset. For the amount of offset, the signs are positive (+) when the robot is offset on the upstream side, and negative (-) when offset on the downstream side.**



Adjustment amount (ms) = (Offset amount (mm) / Conveyor speed (mm/sec))*1000
Add the calculated value to the original value if the adjustment amount have been set.

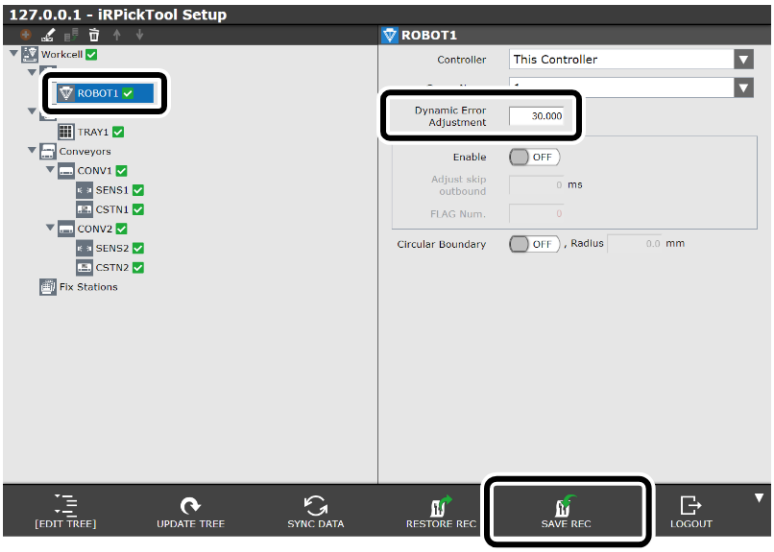
10 **Open the robot homepage.**

11 **Select [iRVision/iRPickTool] in the menu area and click [iRPickTool Workcell Setup] in the content display area.**



12 Select [ROBOT1] and enter the adjustment amount in [Dynamic Error Adjustment] calculated in step 9. If the adjustment amount is already set, add the calculated amount to the original value.

Click [SAVE REC].



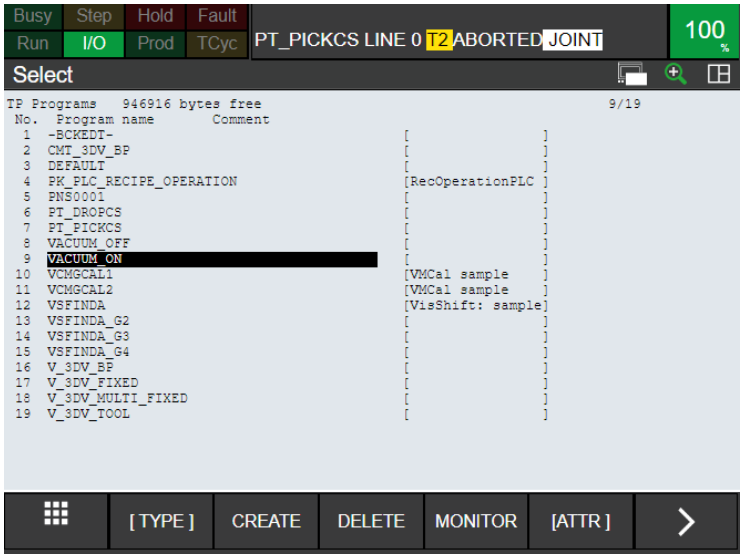
13 Repeat the adjustment of motion error from step 1.

⚠CAUTION

When you adjust the motion error, be sure to perform step 1 through step 8 of "Executing the program" to finish the procedures.

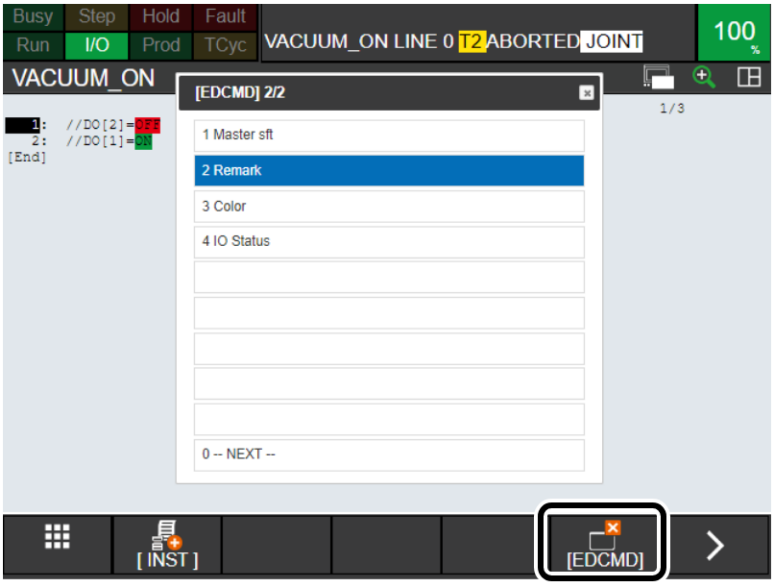
Uncomment the tool control program

1 Press [SELECT] to display the program list screen.



2 **Select the commented tool control program and press [ENTER].**

3 **Place the cursor over the first line and press [Remark] in the edit commands of the icon editor.**

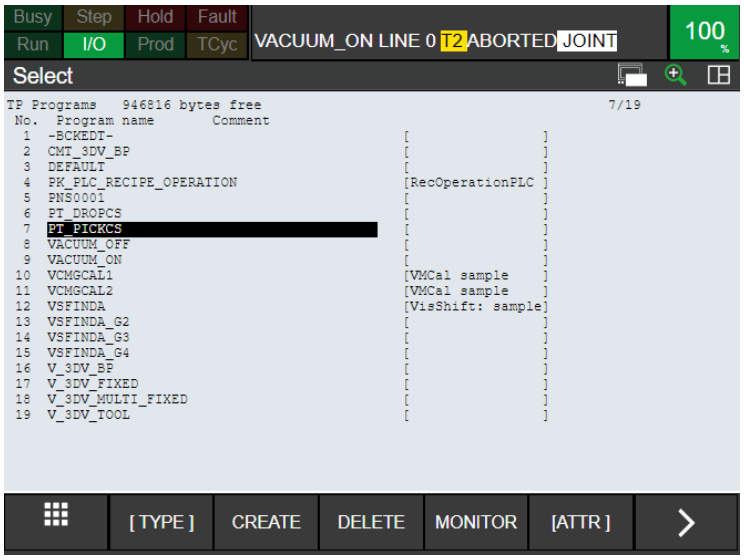


4 **Select the range to uncomment and press F4 [UNREMARK].**



Revert the program to the original setting

1 Press [SELECT] to display the program list screen.



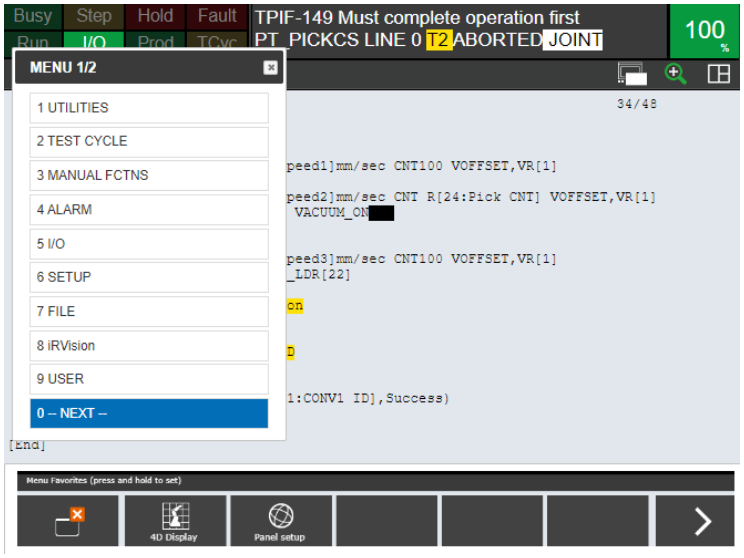
2 Select the tracking program that the position offset instruction is added and press [ENTER].

```
...
25: L PR[7:Pick Pos]
      R[26:Pick Speed1]mm/sec
      CNT100 VOFFSET,VR[1]
      Offset,PR[11:Pick Apr Z]
26: L PR[7:Pick Pos] R[27:Pick Speed2]mm/sec
      CNT R[24:Pick CNT]
      VOFFSET,VR[1]
      AP_LDR[21]
      TB R[25]sec, CALL VACUUM_ON
      Offset,PR[15]
27: WAIT R[23]
...
```

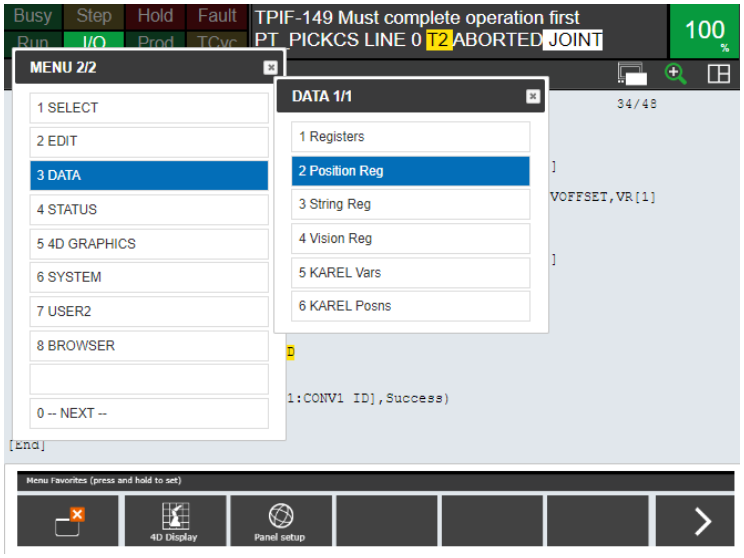
3 Place the cursor over the position offset instruction added, press F4 [CHOICE] and press ‘No option’.

Remove the offset

1 Press [MENU], select [0 - NEXT --] and press [ENTER].

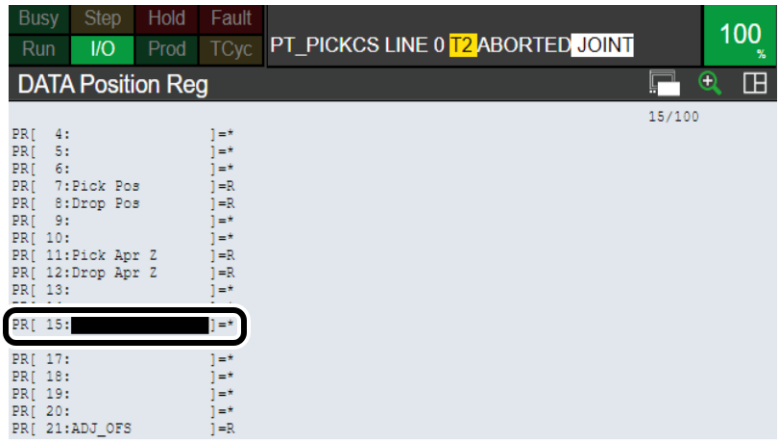


2 Select [3 DATA] → [Position Reg].



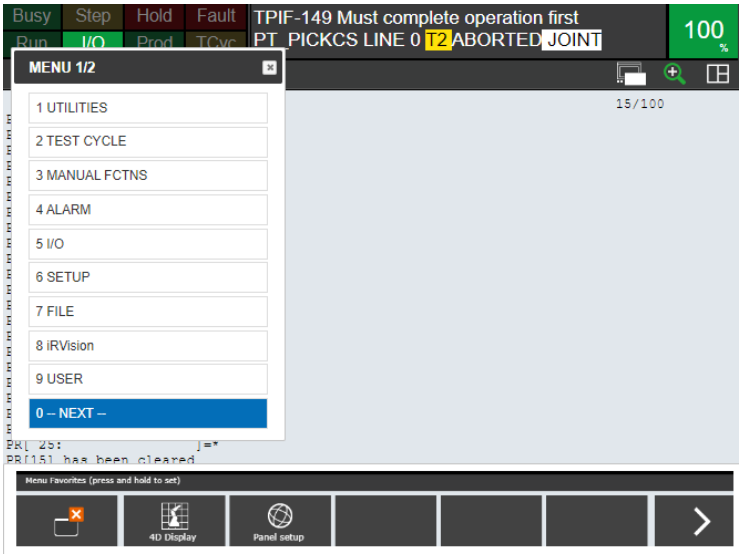
3 Place the cursor over PR[15], press [SHIFT] and F5 [CLEAR].

When the message “PR[15] will be cleared. OK?” appears, select F4 [YES].

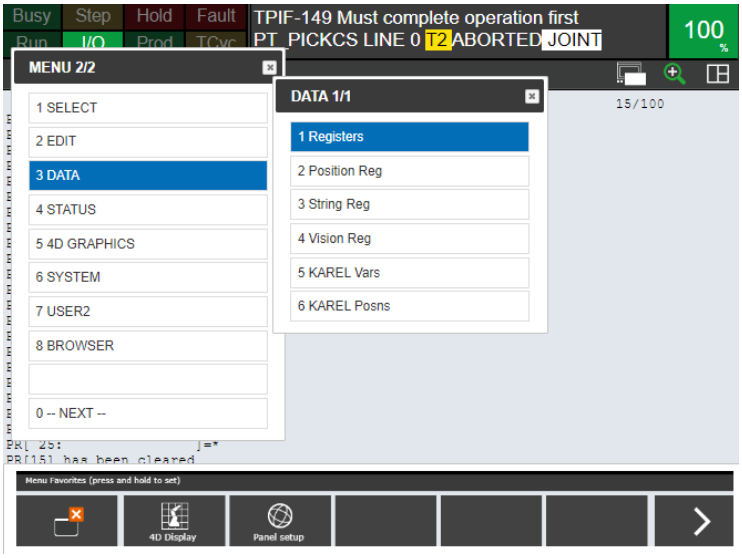


Revert the timer setting

1 Press [MENU], select [0 - NEXT --] and press [ENTER].



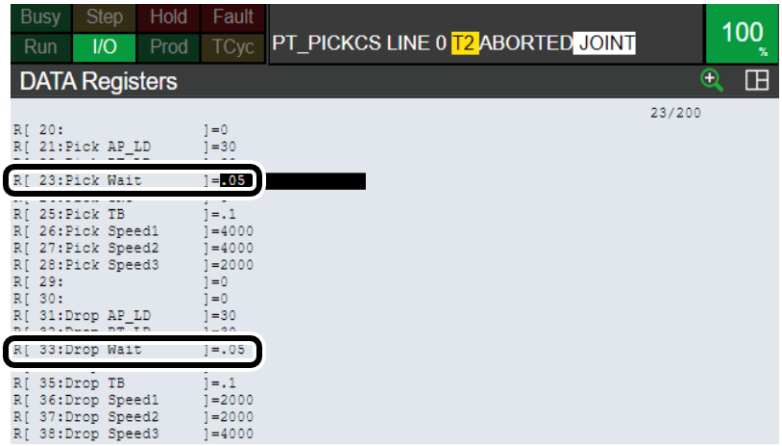
2 Select [3 DATA] → [Registers].



3 Enter the original value in the register changed in "Changing the timer setting".

R[23: Pick Wait] to adjust with pick program

R[33: Drop Wait] to adjust with placement program



3. iRPickTool (basic functions and Plug & Play) startup procedures

This chapter explains the setup procedures for iRPickTool (Basic Functions + Plug & Play).

By performing the procedures as explained in this chapter, you will complete the setup of systems that use four-axis delta robots as shown in the figure below.

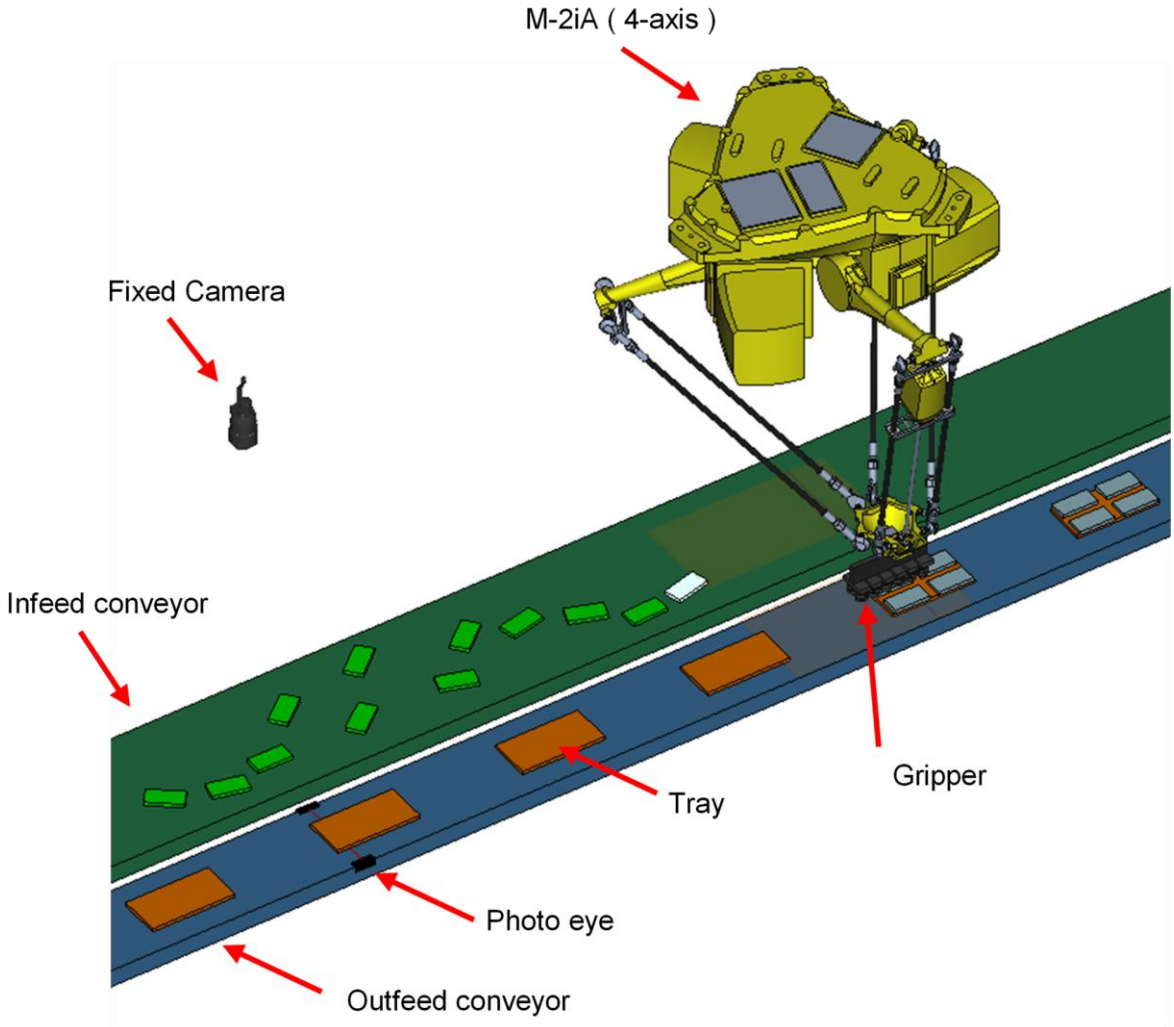


Fig. 3-1 System configuration

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

This chapter explains the following combinations as system configuration examples.

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

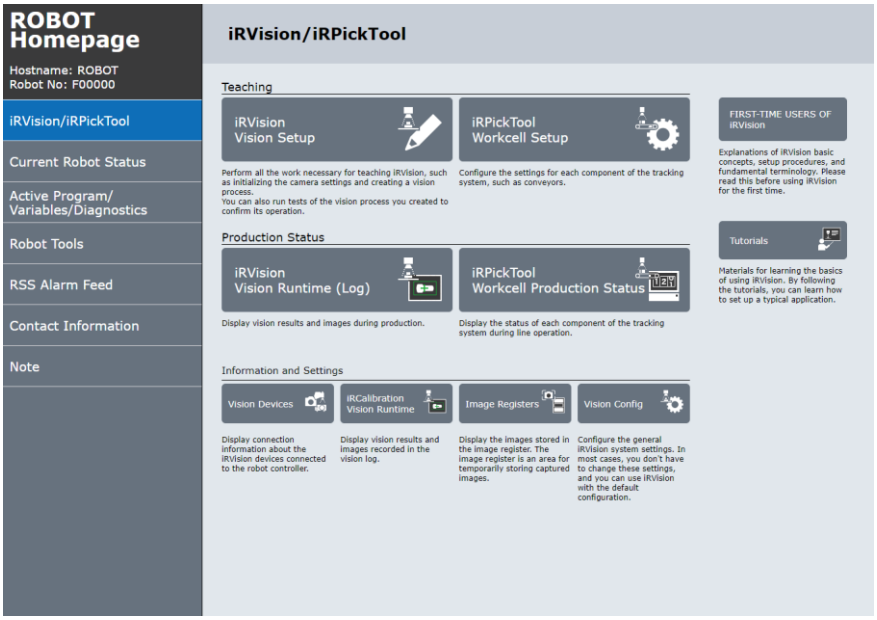
For information on the setup procedures when conveyors and fixed stations in other combinations are used and details on each tree item, refer to "I.5. Basic functions reference(P.258)". For the standard programs and objects for Plug & Play (gripper/zone/object), see "I.6. Plug & Play reference(P.421)".

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

3.1. Access to the robot homepage

Before opening the iRPickTool screen, follow the procedure below to display the robot homepage.

1 **Start an internet browser.**
2 **Enter the IP address of the robot controller in the address bar.**
The homepage of the robot will be displayed.



All robot controllers have their robot homepages. When the robot controller has iRPickTool options, the following links related to iRPickTool appear on the robot homepage.

Setting up a workcell	Opens the edit screen for setting up iRPickTool. For details, see “I.5. Basic functions reference(P.258)” or “I.6. Plug & Play reference(P.421)”.
Workcell product status	Opens the edit screen for displaying the system production status set with iRPickTool. For details, see “I.8.2. Checking the production status(P.567)”.

3.2. Preparation before setting up the application

This section explains the preparation for the setup of tracking systems that use iRPickTool, including the connection of hardware.

3.2.1. Connection and setup of Pulsecoders

Install a Pulsecoder and connect it to a robot controller.
After connecting, set parameters on the teach pendant of the 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”.

In the tracking system, the control unit may contain the sensor I/F board as the signal interface, and the sensor distribution board may be connected to it as needed. Alternatively, the separate interface device (SDU) may be attached in the cabinet. When the SDU is attached, an FSSB optical fiber cable for connection is also required.

In addition, cabling (encoder cables) for external hardware (encoder) is required to track conveyors, pallets, tables, or other items. Only the αiA4000S Pulsecoder A860-2052-T321 (both incremental and absolute) can be used in the R-50iA / R-50iA Mate Controller. For the specifications, see Fig. C.2-1(P.712), Table C.2-1(P.712), and Fig. C.2-2(P.713).

Hardware	Description
Interface device	Insert the sensor I/F board into the appropriate slot of the control unit. (See Fig. C.1-1(P.711)) When connecting two encoders to the sensor I/F board or when using the force sensor at the same time, attach the sensor distribution board in the cabinet. (See Fig. C.1-2(P.711)) For example, when you want to use three or more encoders, attach the separate interface device (SDU) in the cabinet.
Encoder	Use the αiA4000S Pulsecoder A860-2052-T321. To obtain a sufficient resolution, use an appropriate gear and reducer. (300 pluses/mm or more is recommended)

3.2.1.1. Connecting Pulsecoders

Connect the Pulsecoder to a robot controller.

Connecting the Pulsecoder

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

> The following figure assumes that the sensor I/F board and sensor distribution board are attached to the robot controller.
>
> **MEMO** For how to attach the sensor I/F board and sensor distribution board to the robot controller, see "Line Tracking OPERATOR'S MANUAL"(B-84464EN-10).

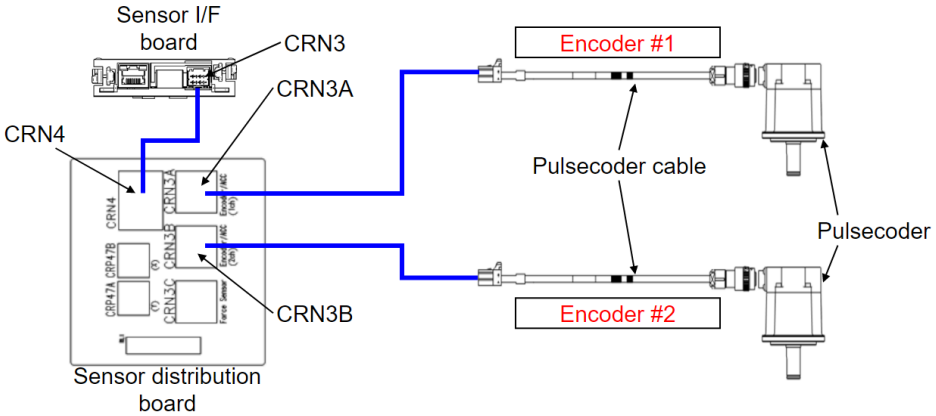


Fig. 3.2.1.1-1 Connection between the sensor I/F board, sensor distribution board, and Pulsecoders

3.2.1.2. Setting up Pulsecoders

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

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

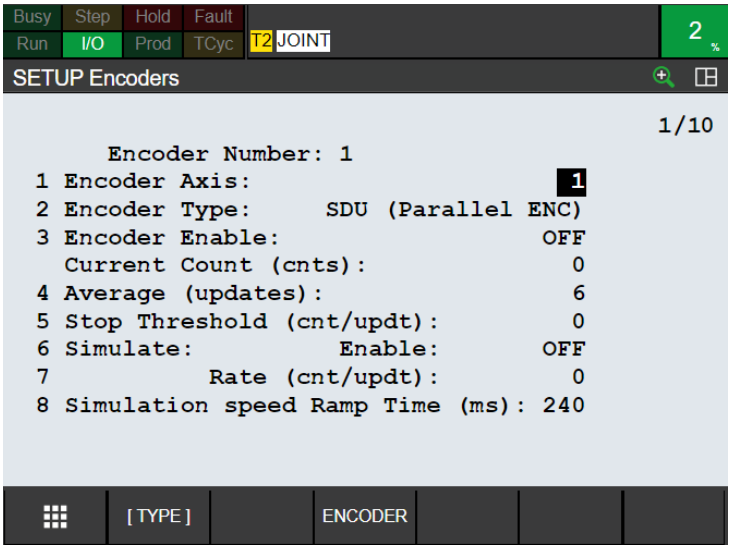
1 **Press the [MENU] key on the robot controller's teach pendant.**
The menu appears.

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

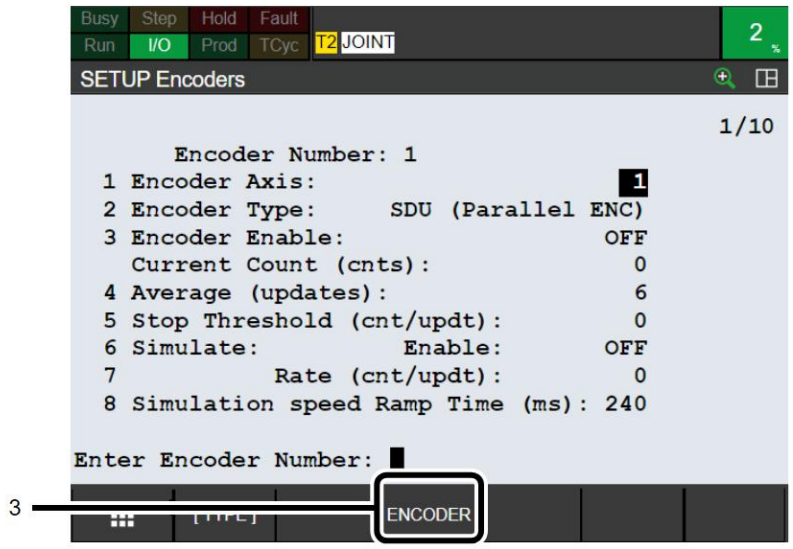


After a while, the following screen appears.

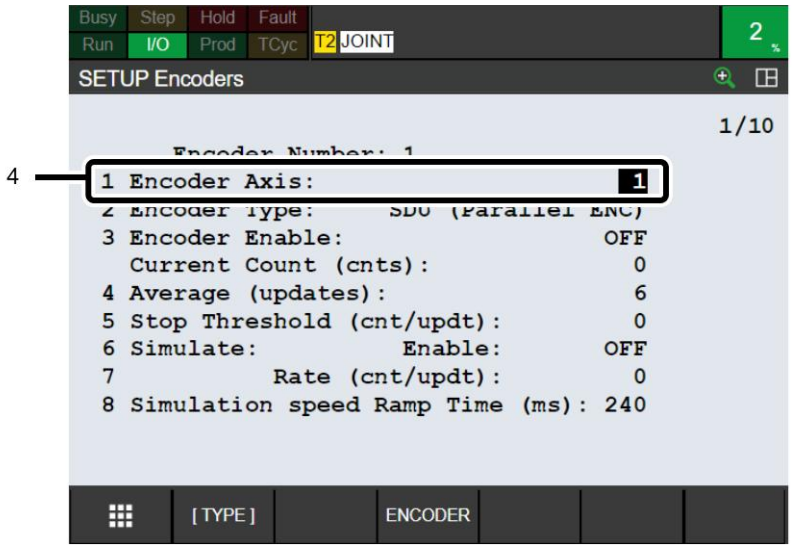
Number [9] and after in the setting items are the menu for optional functions. These will not be set at this time.



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



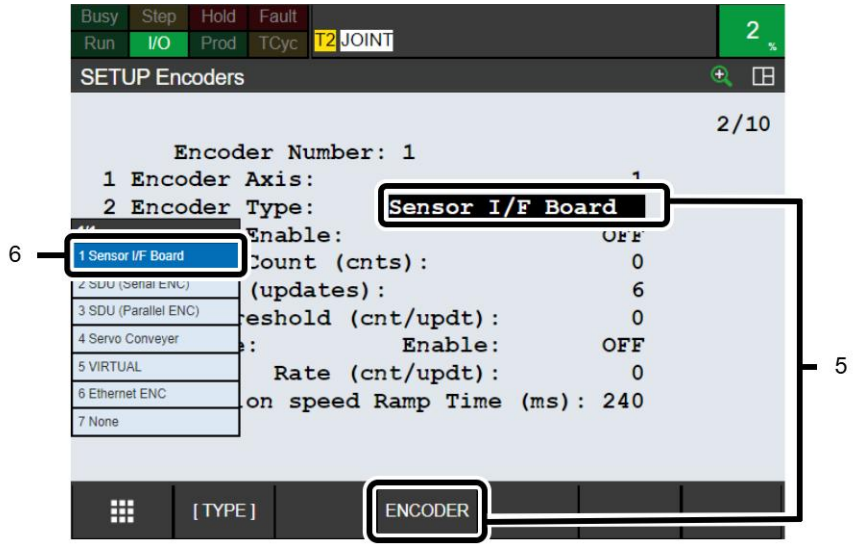
4 Move the cursor to [Encoder Axis] and input 1.



5 Move the cursor to [Encoder Type] and press F4 [CHOICE].

The menu appears.

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



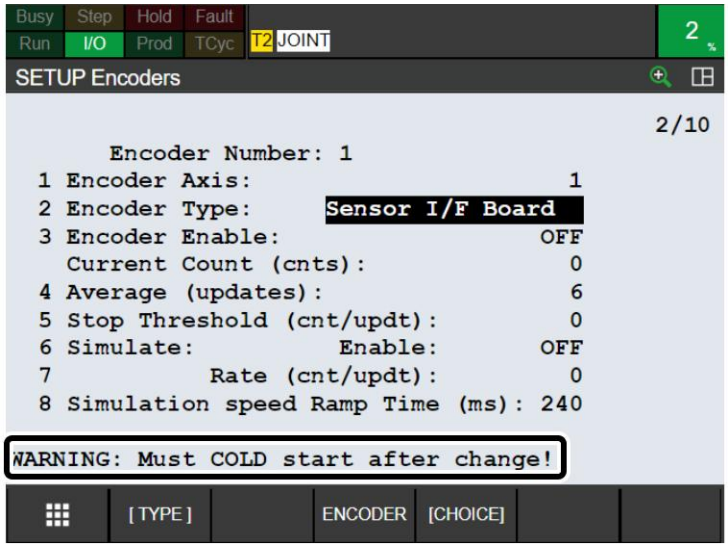
Encoder type

There are the following types for encoders.

- αiA4000S Pulsecoder (red Pulsecoder): [sensor I/F board]
- Incremental Pulsecoder (black Pulsecoder): [parallel incremental]
  - ∗ Cannot be connected to the sensor I/F board

	αiA4000S (A860-2052-T321)		Incremental Pulsecoder (A860-0301-T001 to T004)

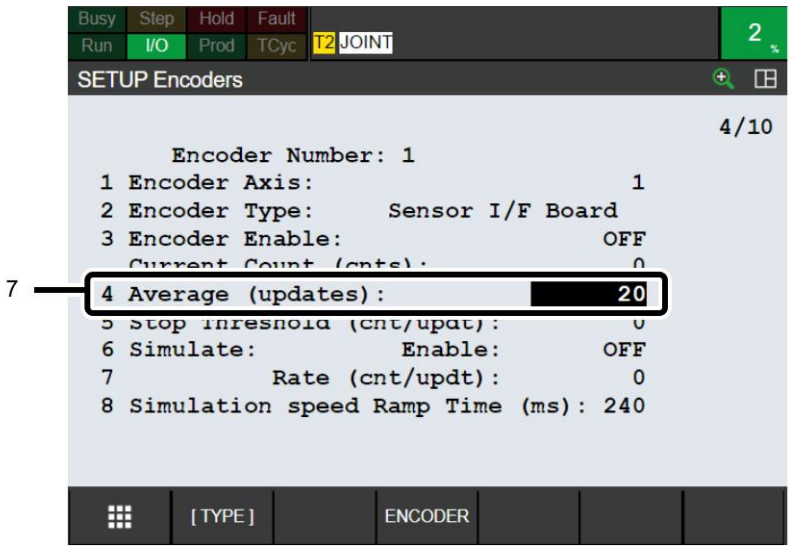
If encoder axes or encoder types are changed, a warning prompting to cold start the robot will appear.



7 Move the cursor to [Average (updates)] and input the cumulative accumulated number of instantaneous speeds (width of moving average).

By setting the accumulated number of instantaneous speeds, the robot will stop smoothly rather than suddenly, even if the conveyor suddenly stops while the robot is tracking.

Normally, enter “20”.

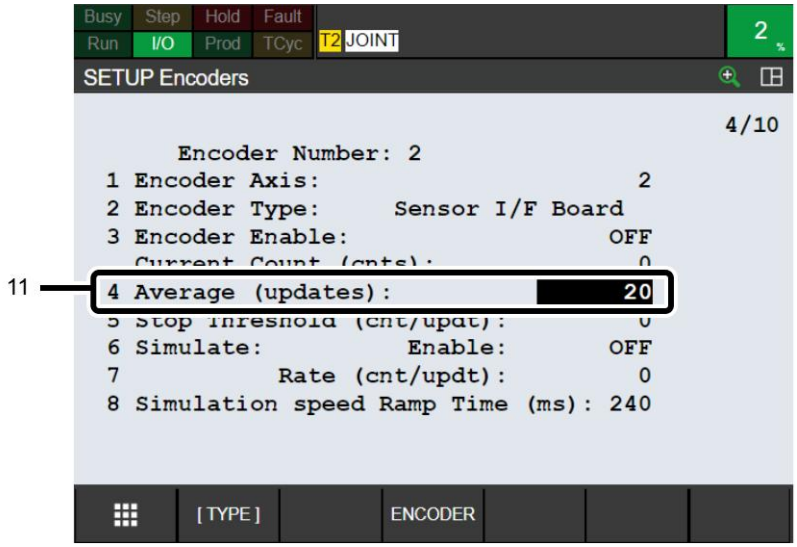


8 Next, set the parameters for the Pulsecoder that is installed at the outfeed conveyor. Press F3 [ENCODER], then input 2.

9 Move the cursor to [Encoder Axis] and input 2.

10 Move the cursor to [Encoder Type], press F4 [CHOICE], and select the encoder type that corresponds to the Pulsecoder that is to be connected.

11 Move the cursor to [Average (updates)] and input a value. This is the same as step 7. Normally, enter "20".

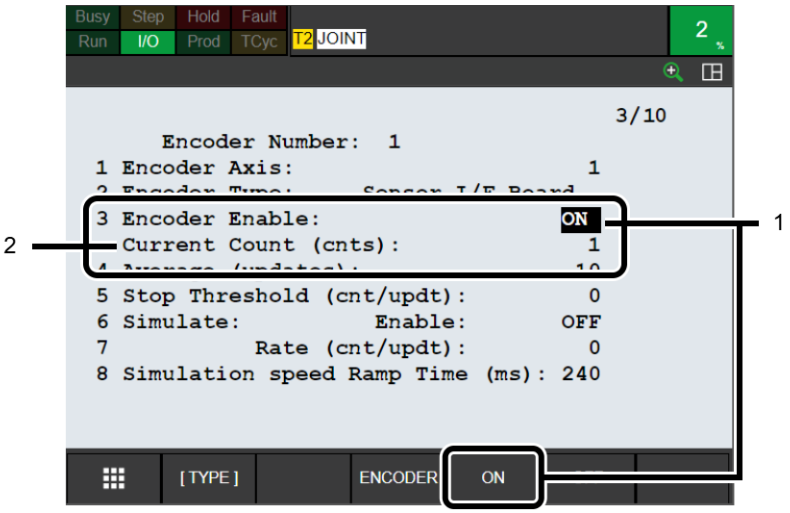


12 Finally, cold start the robot controller.

3.2.1.3. Checking the Pulsecoder settings

1 Open the [Encoders] screen and move the cursor to [Encoder Enable], then press F4 [ON].

2 Move the conveyor and confirm that the pulse count is displayed at [Current Count (cnts)].



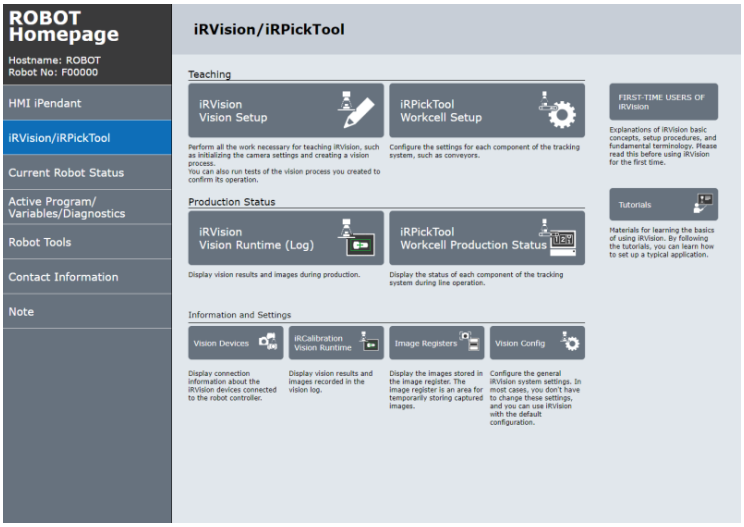
3 In the same manner, select Encoder Number: 2 for the Pulsecoder that is connected to the outfeed conveyor, then confirm.

3.2.2. Checking the connection of the camera

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

0 Open the robot homepage.

1 Select [iRVision/iRPickTool] in the menu area and click [iRVision Vision Setup] in the content display area.

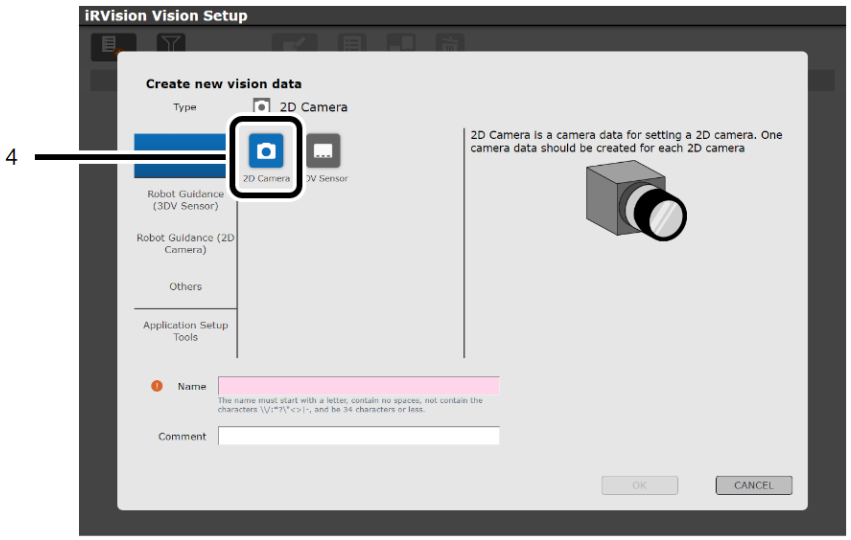


The [iRVision Vision Setup] screen appears.

2 Click [CREATE].



3 Select [2D Camera].



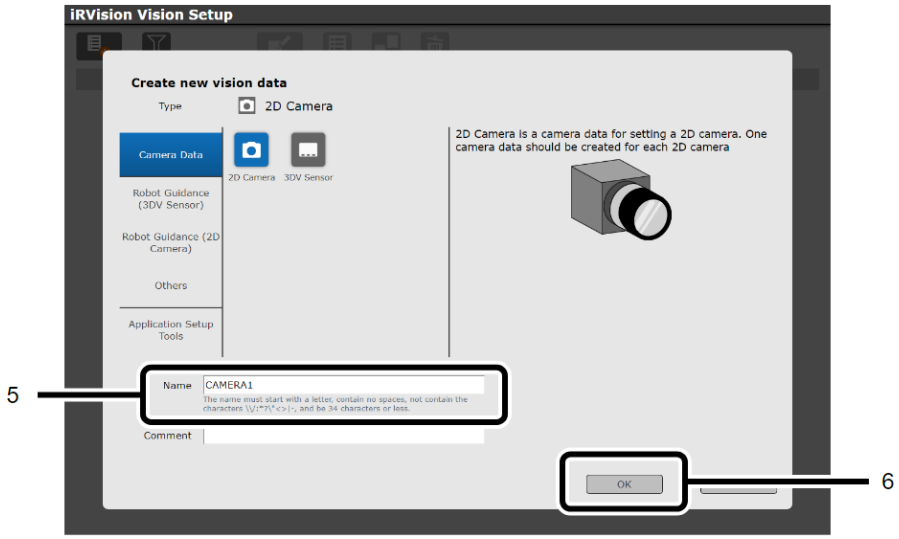
After a while, the following screen appears.

4 Input the name of the camera in [Name].

Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

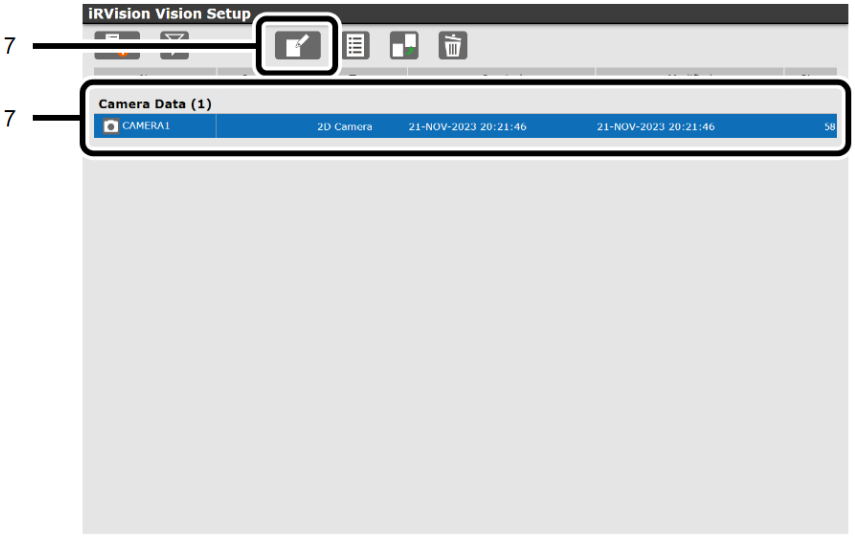
Example: CAMERA1

5 Click [OK].



The created camera data is displayed in the list.

6 Select the created camera data and click [EDIT].



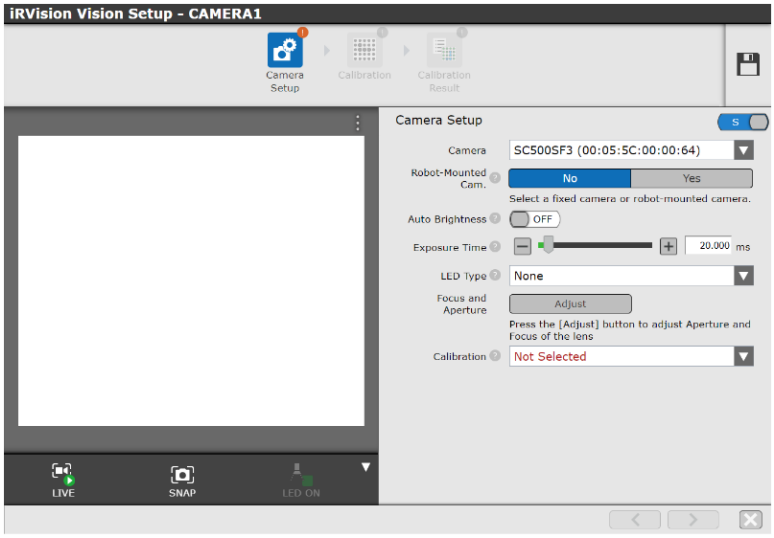
After a while, the following screen appears.

7 Switch the display mode by using the toggle switch at the upper right of the screen.

Simple: Hide less frequently used setting items.

Advanced: Display all setting items.

8 Select the connected camera in the [Camera] drop-down list.



9 To adjust the focus of the lens, place something that has high contrast characters within the field of view of the camera on the conveyor surface, then click [LIVE].

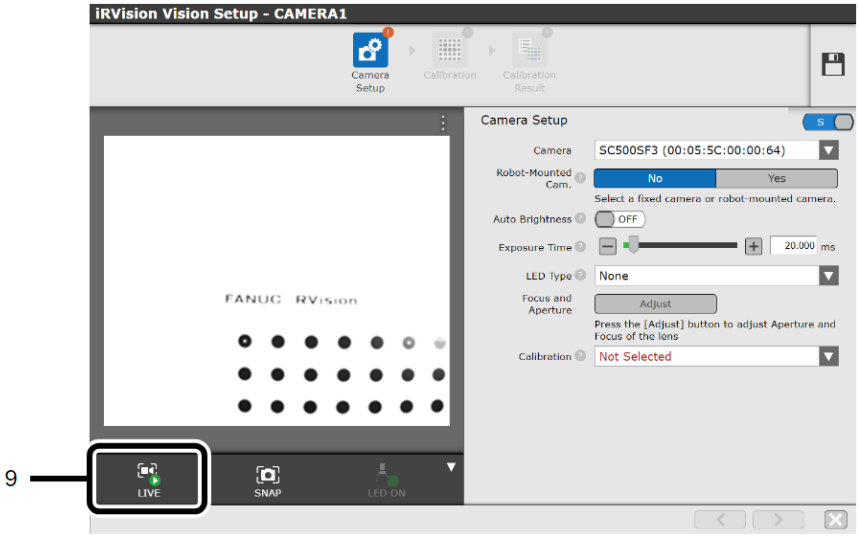
Confirm that the item that was placed has been captured within the frame on the left side of the screen.

* If the screen is dark, adjust the following.

- Open the diaphragm of the lens.
- Adjust the lighting.
- Adjust the exposure time.
- Switch (ON/OFF) Auto Brightness.

10 Check the field of view of the camera.

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



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



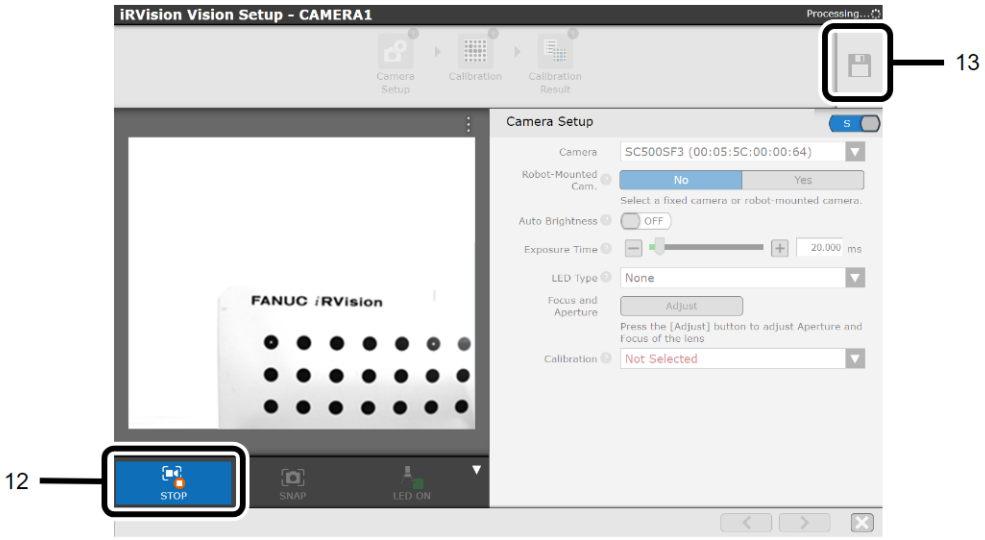
Fig. 3.2.2-1 Focus adjustment

12 Click [STOP].

Live will stop.

13 Click [SAVE].

The edit screen will close.



3.2.3. Checking the input of the trigger sensor

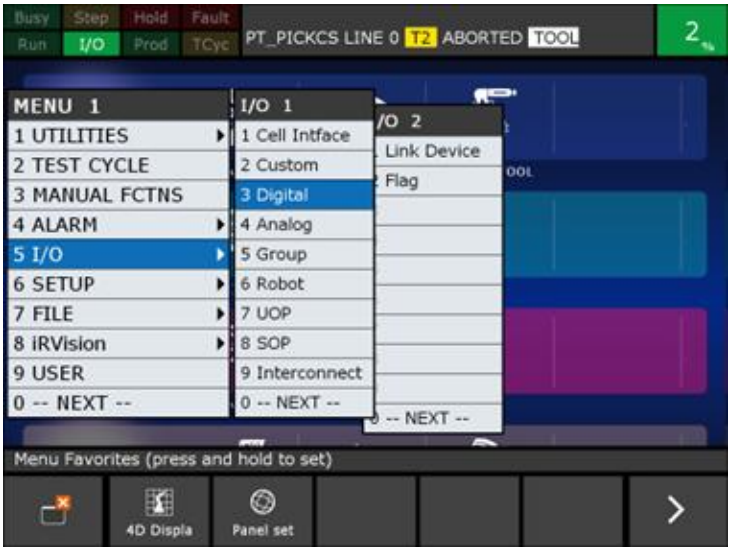
Check the input of the trigger sensor.

3.2.3.1. Using DI / RI

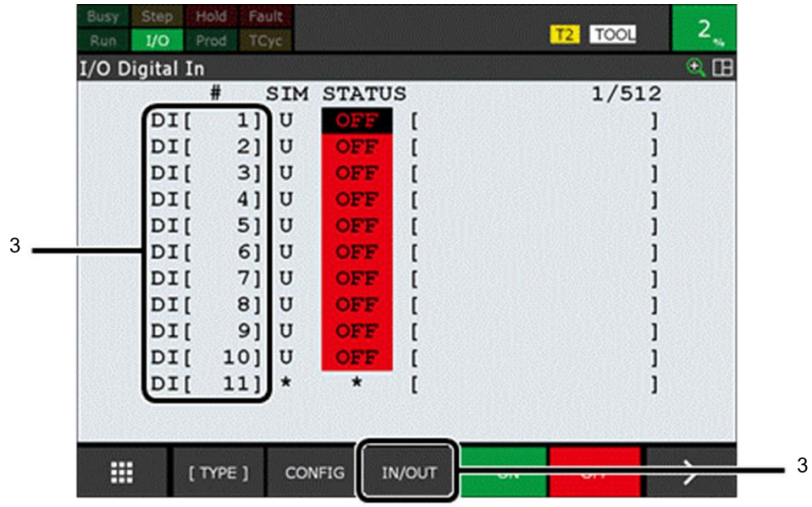
Check the input of the trigger sensors when DI / RI are used.

1 **Press the [MENU] key on the robot controller's teach pendant.**
The menu appears.

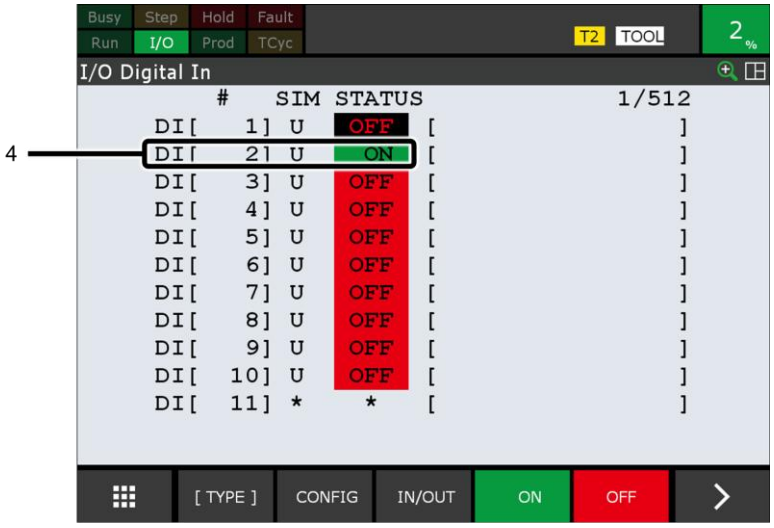
2 **If you are using DI, select [I/O] → [Digital] from the menu. If you are using RI, select [I/O] → [Robot] from the menu.**
For example, when DI is used the screen will look like this.



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



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



3.2.4. Setting up the tool frame of the pointer tool

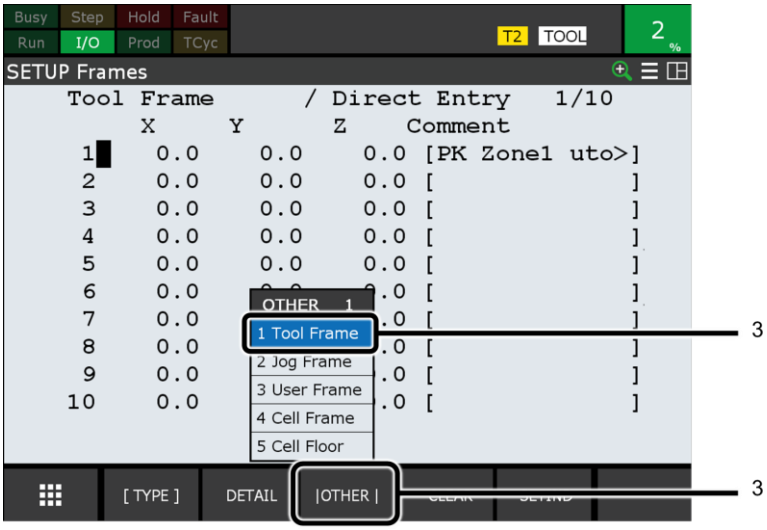
Set up the tool frame of the pointer tool.

1 Press the [MENU] key on the robot controller's teach pendant.
The menu appears.

2 From the menu, select [SETUP] → [Frames].



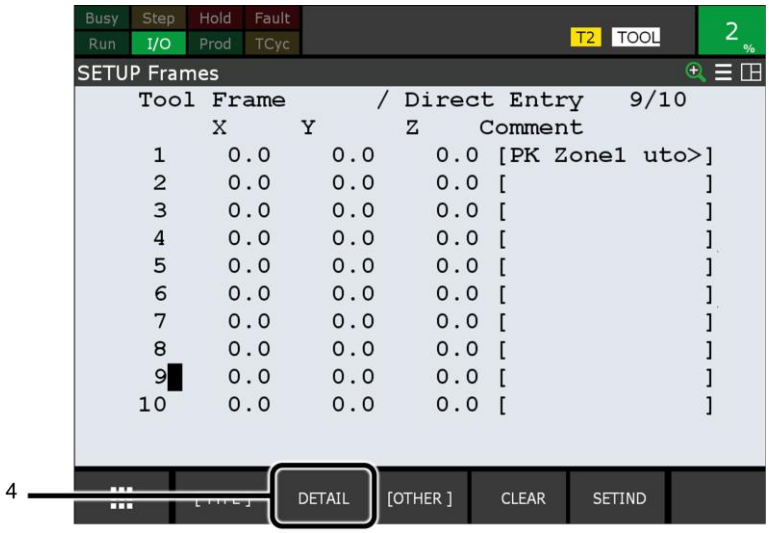
3 If frames other than the tool frames are displayed, press F3 [OTHER] and select [Tool Frame] from the menu.



4 Move the cursor to the line of the tool frame number to be set and press F2 [DETAIL].

The setup screen for the tool frame for the selected frame number will appear.

This screen is an example of when performing setup using Tool 9.



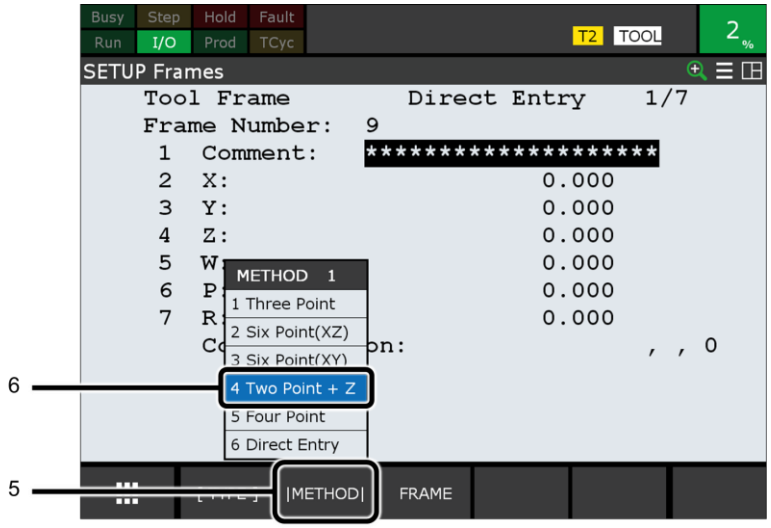
⚠CAUTION

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

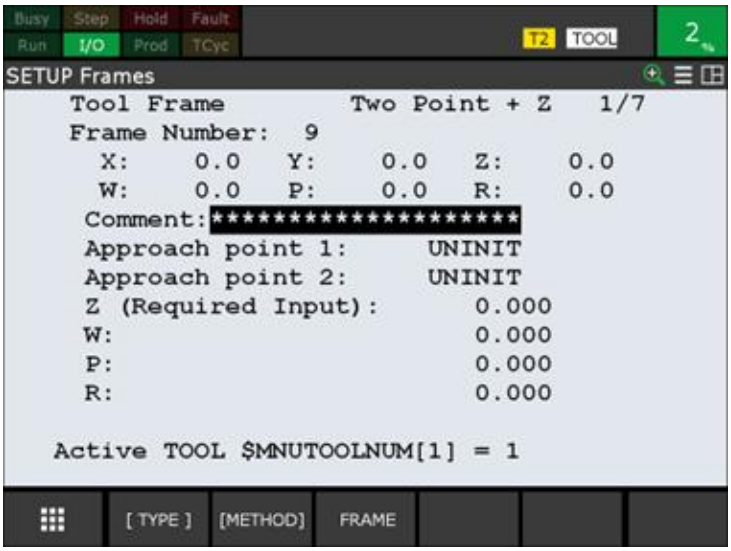
5 Press F2 [METHOD].

The menu appears.

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



After a while, the following screen appears.

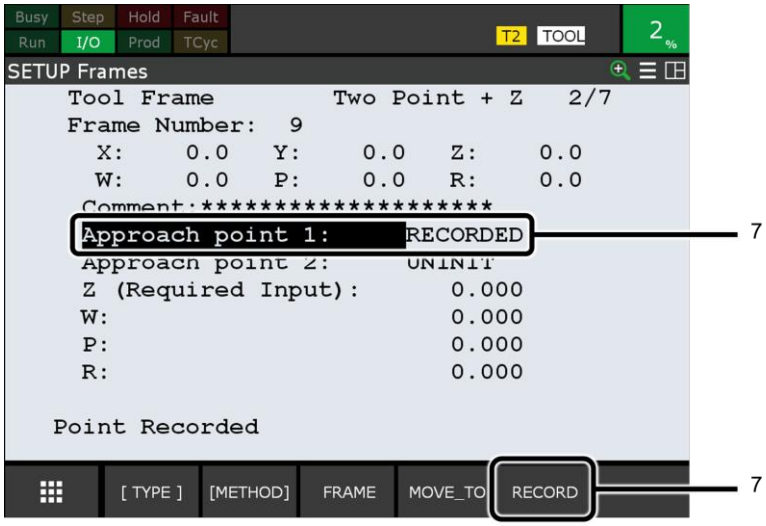


7 Move the cursor to [Approach point 1] and input approach point 1.

Jog the robot to the approach point to be recorded.

While holding down the [SHIFT] key on the teach pendant, press F5 [RECORD] to record the current position as the approach point.

'RECORDED' will be displayed.

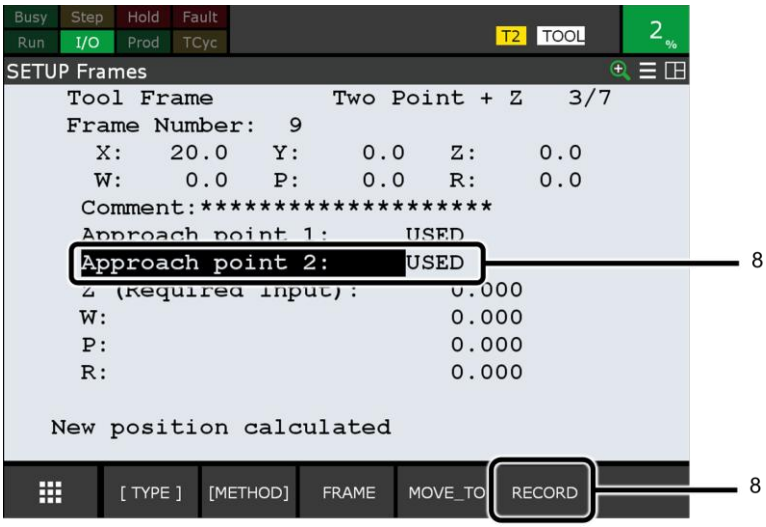


8 Move the cursor to [Approach point 2] and input approach point 2.

Jog the robot to the approach point to be recorded.

While holding down the [SHIFT] key on the teach pendant, press F5 [RECORD] to record the current position as the approach point.

When all the approach points are input, 'USED' will be displayed and the X and Y of the tool frame will be set.



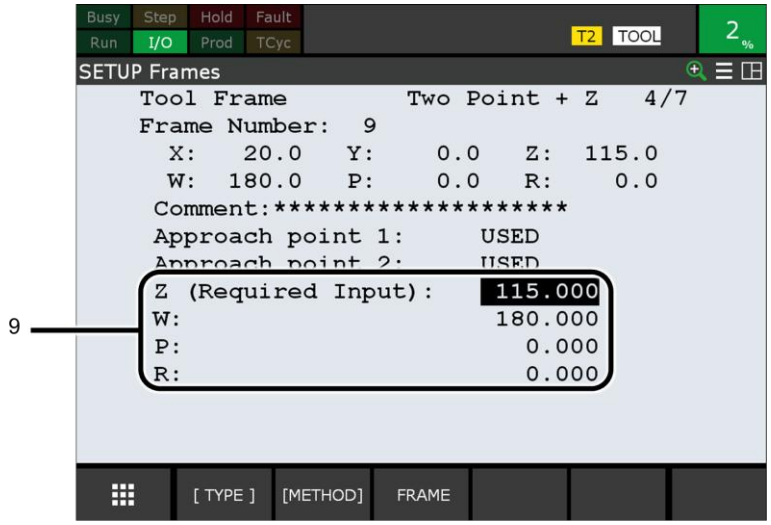
CAUTION

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

9 Move the cursor to [Z], [W], [P] and [R] and enter a value for each.

For [Z], input a numerical value measured with a ruler, etc.

Input [W], [P] and [R] directly.



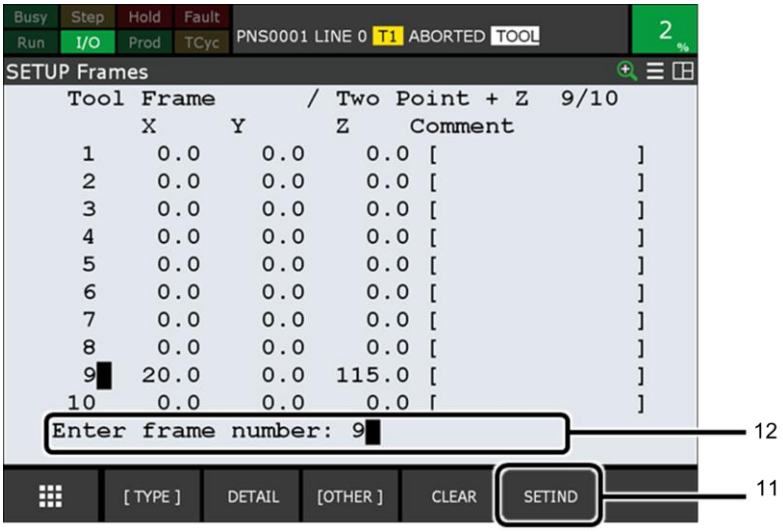
10 Press the [PREV] key on the teach pendant of the robot controller.

The Tool Frame list screen appears.

The setting values of all the tool frames can be checked.

11 Press F5 [SETIND].

12 Input the frame number of the tool frame.

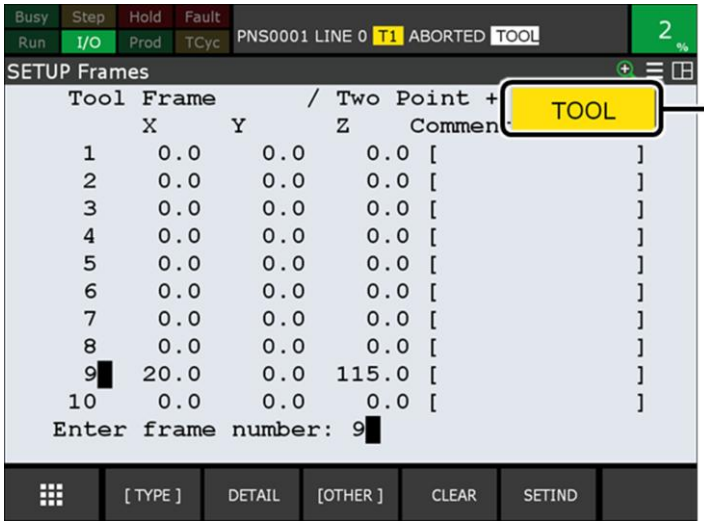


The tool frame will be enabled.

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

13 Set the manual continuous feed of the robot in ‘TOOL’.

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



14 Press the  **key or**  **key on the teach pendant of the robot controller.**

The robot jogs when it rotates around the Z axis.

Verify that the tool frame (TCP) has been set accurately.

3.2.5. Setting payloads (motion performance)

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

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

The menu appears.

2 Select [SYSTEM] → [Motion] from the menu.



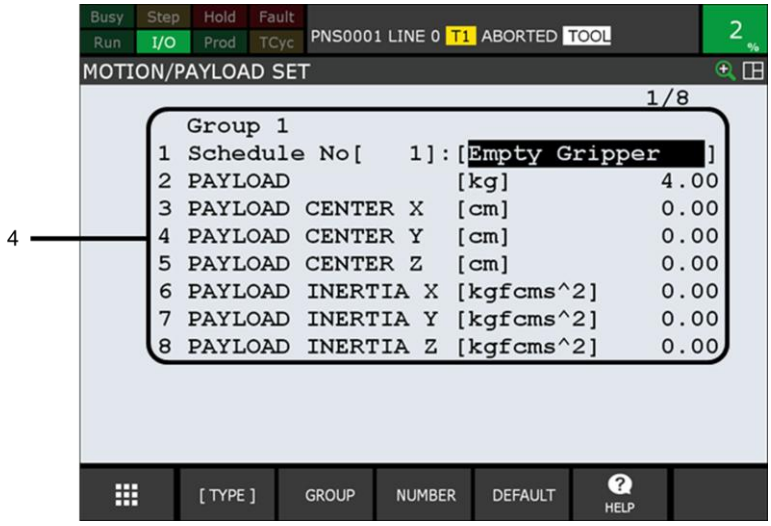
3 Move the cursor to [1] and press F3 [DETAIL].



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

4 Input the [PAYLOAD]. Not all of the items necessarily need to be input.

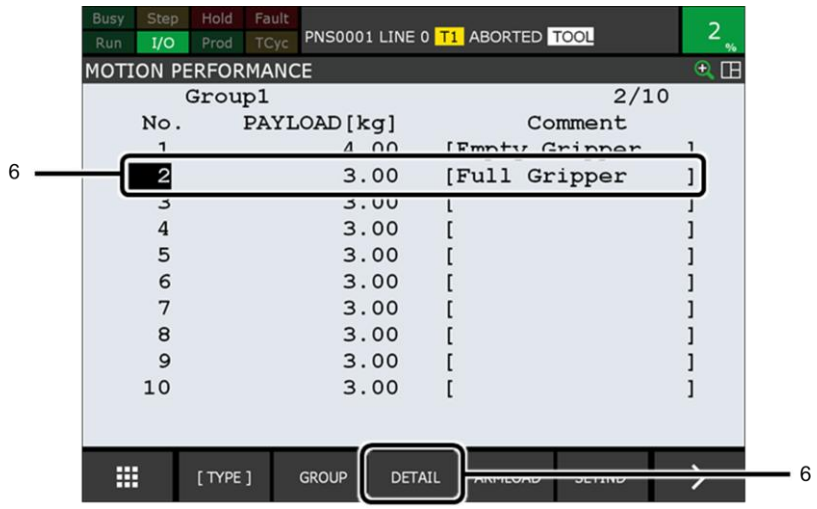
* Set a 'Hand only' payload.



5 Press the [PREV] key on the teach pendant of the robot controller.

Return to the list on the motion performance screen.

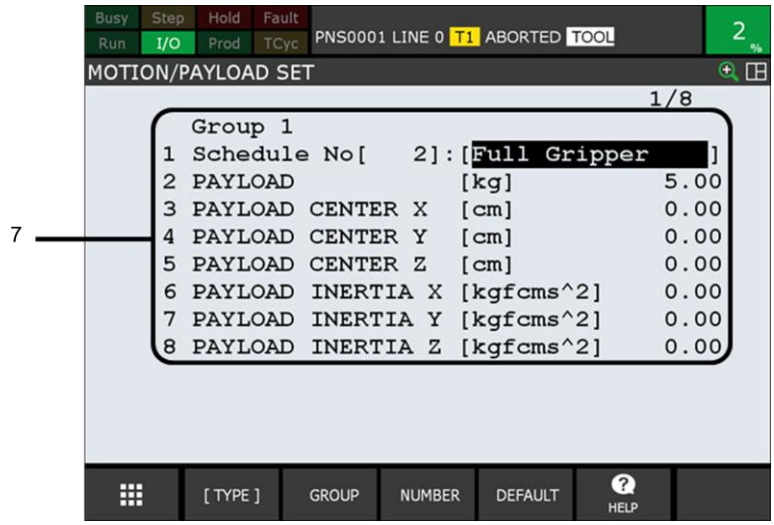
6 Move the cursor to [2] and press F3 [DETAIL].



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

7 Input the [PAYLOAD]. Not all of the items necessarily need to be input.

* Set a ‘Hand + part’ payload.



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

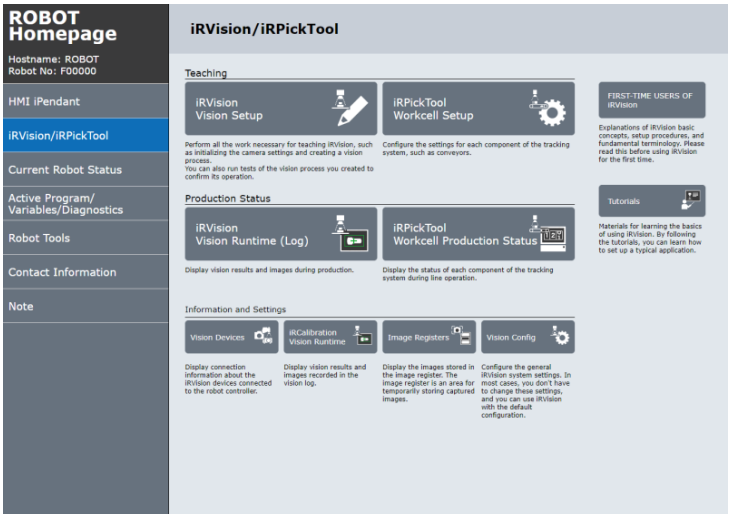
3.3. Setting up applications

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

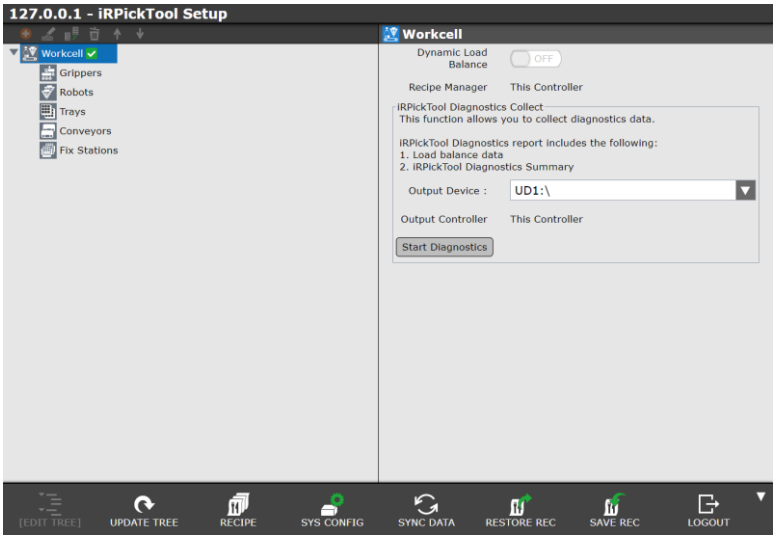
3.3.1. Setting up a workcell

Set up a workcell.

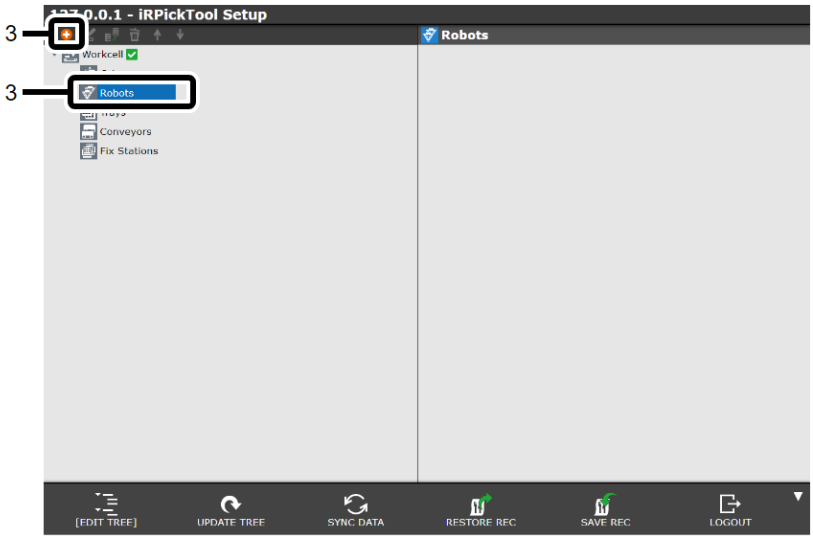
1. **Open the robot homepage.**
2. **Select [iRVision/iRPickTool] in the menu area and click [iRPickTool Workcell Setup] in the content display area.**



The following iRPickTool setup screen will appear.



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



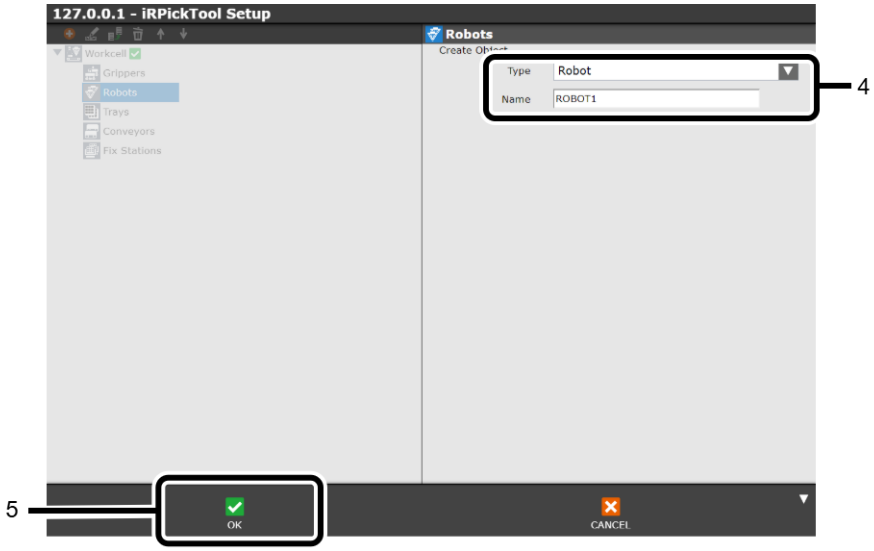
4 **Input the name of the robot in [Name].**

We recommend using the name that is displayed as the initial value just as it is.

Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

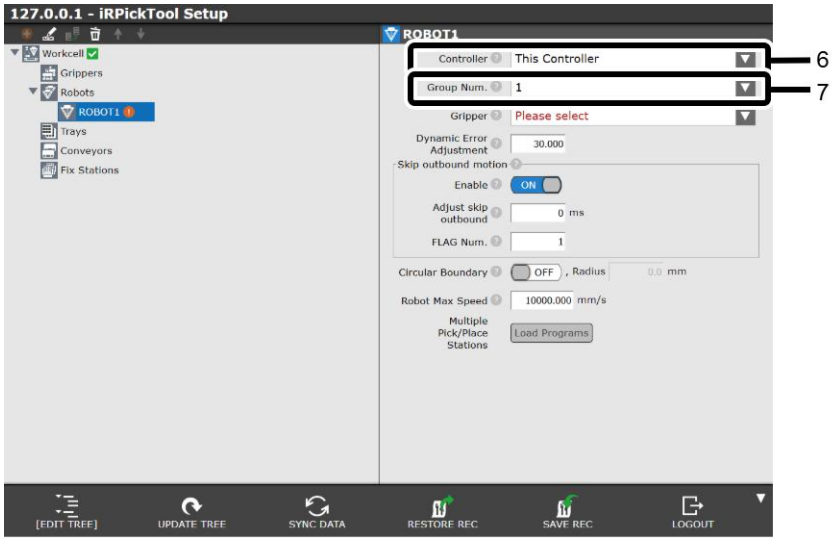
5 **Click [OK].**

The robot object is created.

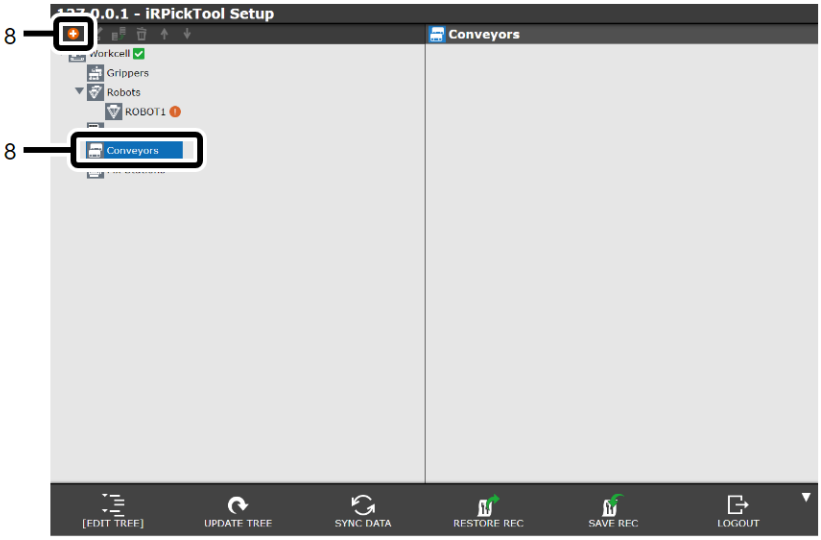


6 **Select [This Controller] from the [Controller] drop-down list (if there is one robot).**

7 **Select [1] from the [Group Num.] drop-down list.**



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



9 **Input the name of the infeed conveyor in [Name].**

We recommend using the name that is displayed as the initial value just as it is.

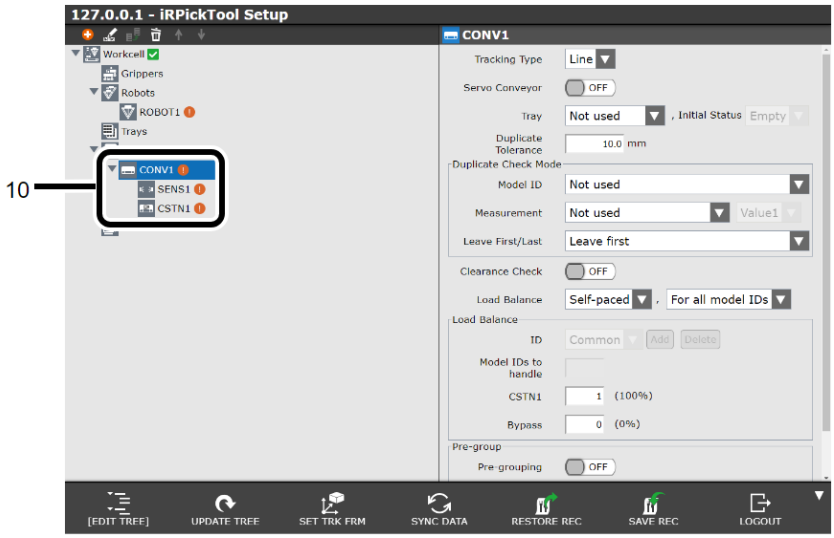
Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

10 Click [OK].

The robot object is created.



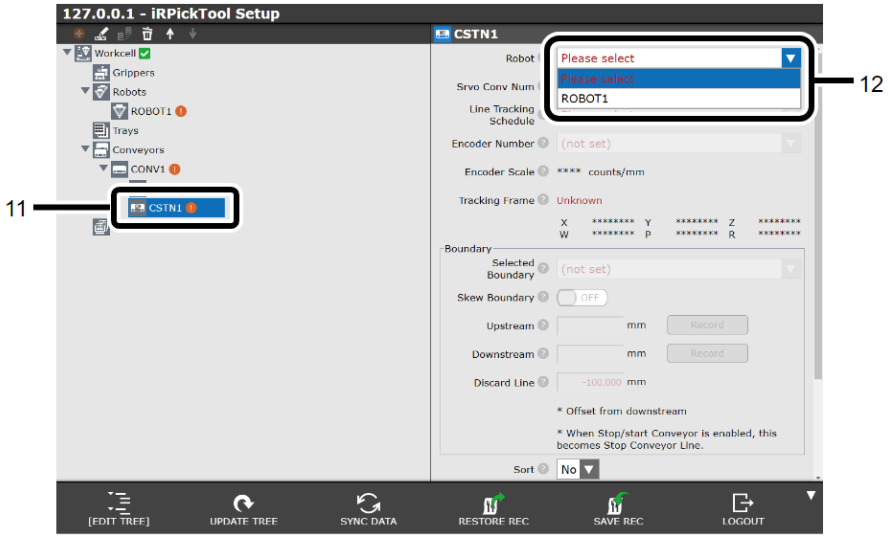
When a new conveyor is created, a sensor and conveyor station will be automatically created.



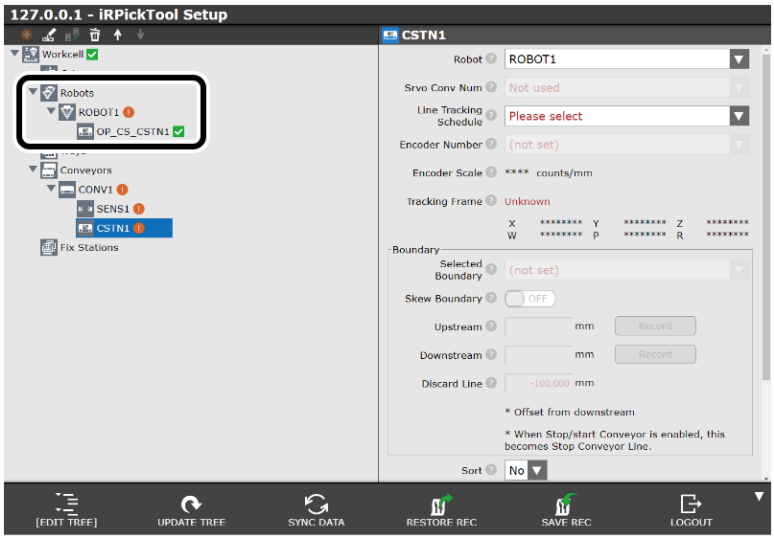
11 Select the conveyor station [CSTN1] that was automatically created in step 10.

12 Select a robot from the [Robot] drop-down list.

Select the robot name that was input in step 4.

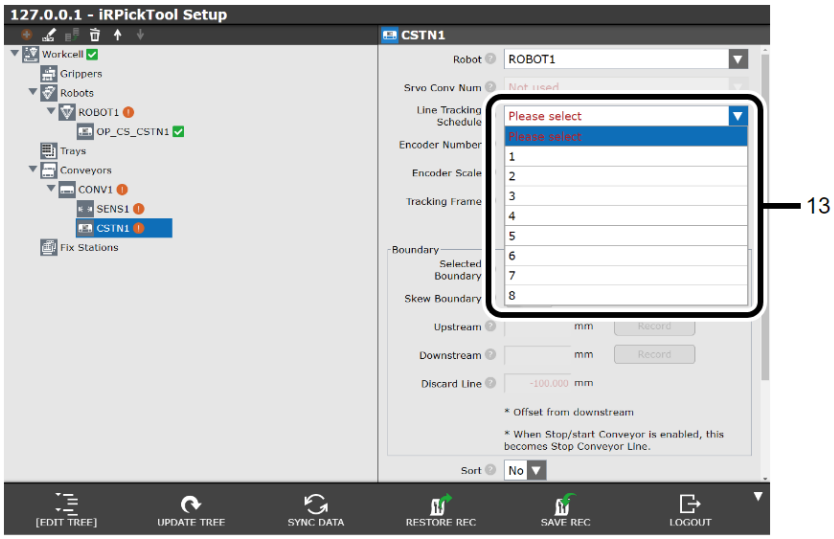


When the name of the robot is selected, the conveyor station operation [OP_CS_CSTN1] will be automatically added underneath the selected robot, as shown below.

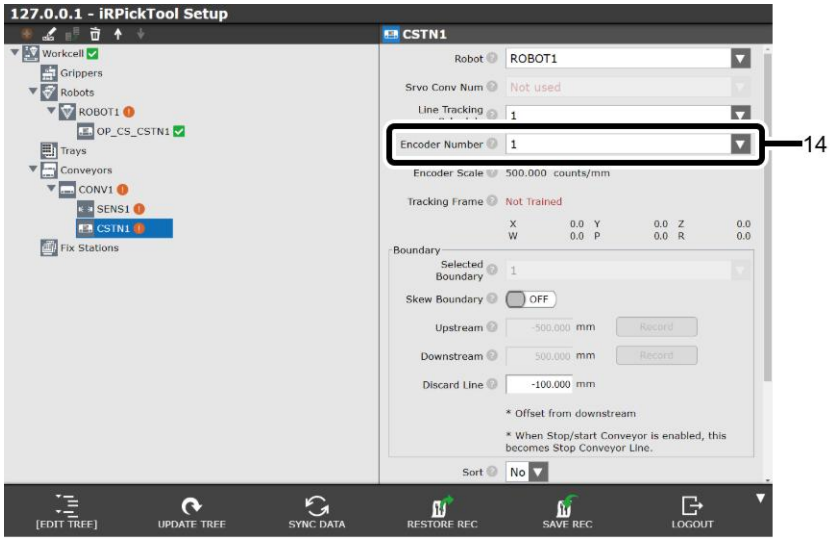


See "I.3.3.10. Setting up a robot operation(P.225)" for information on the setup of conveyor station operations.

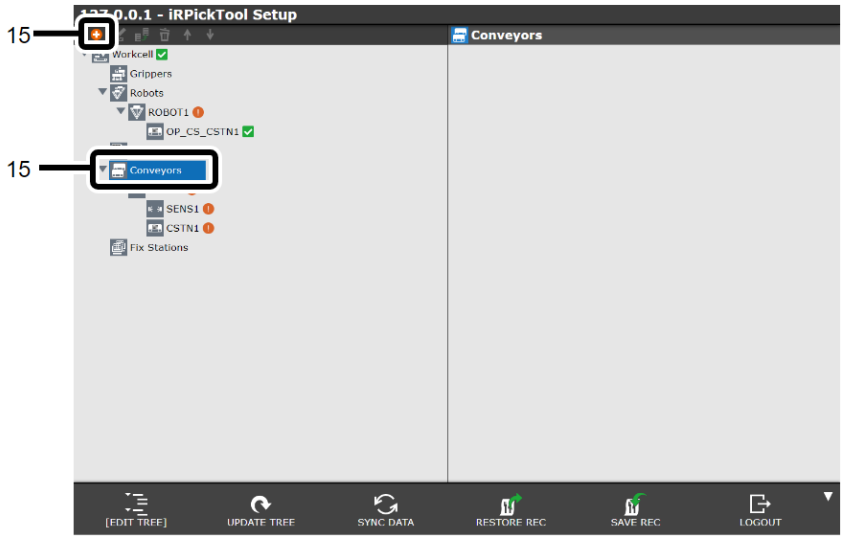
13 Select a number from the [Line Tracking Schedule] drop-down list. Select '1'.



14 Select a number from the [Encoder Number] drop-down list. Select '1'.



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



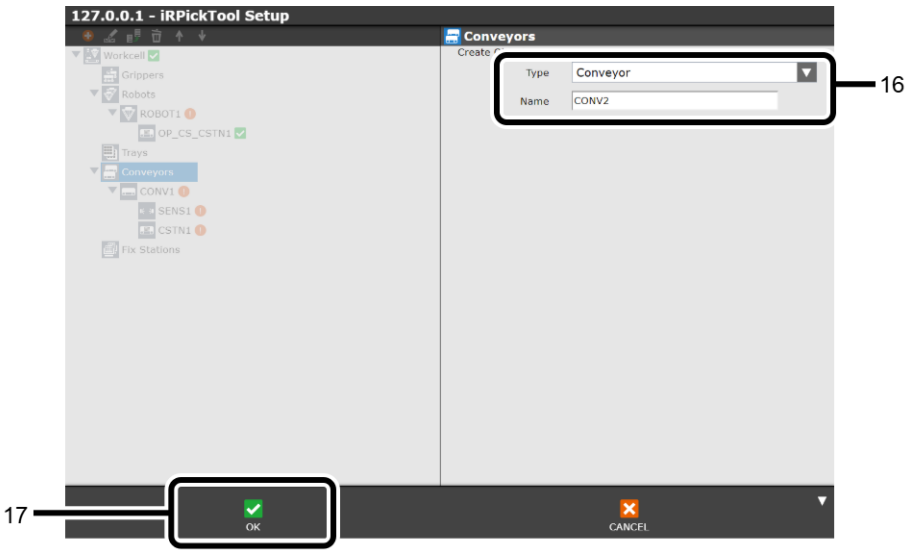
16 Input the name of the outfeed conveyor in [Name].

We recommend using the name that is displayed as the initial value just as it is.

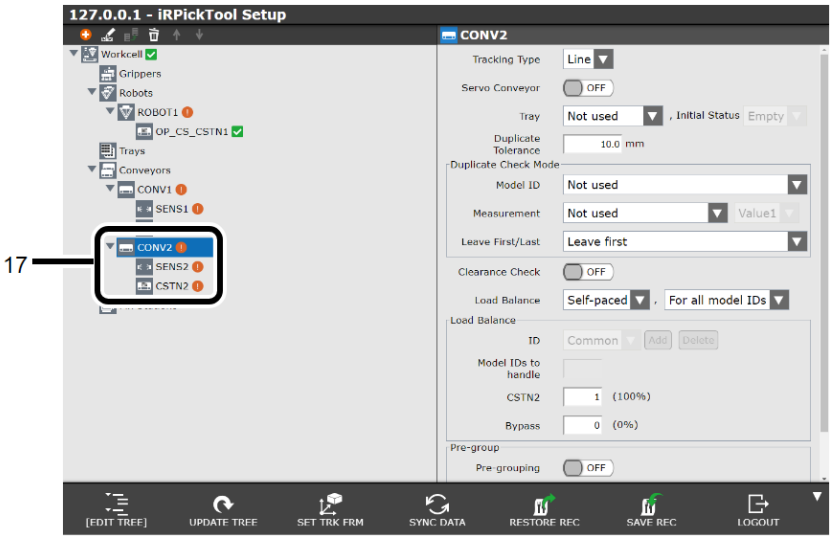
Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

17 Click [OK].

The robot object is created.



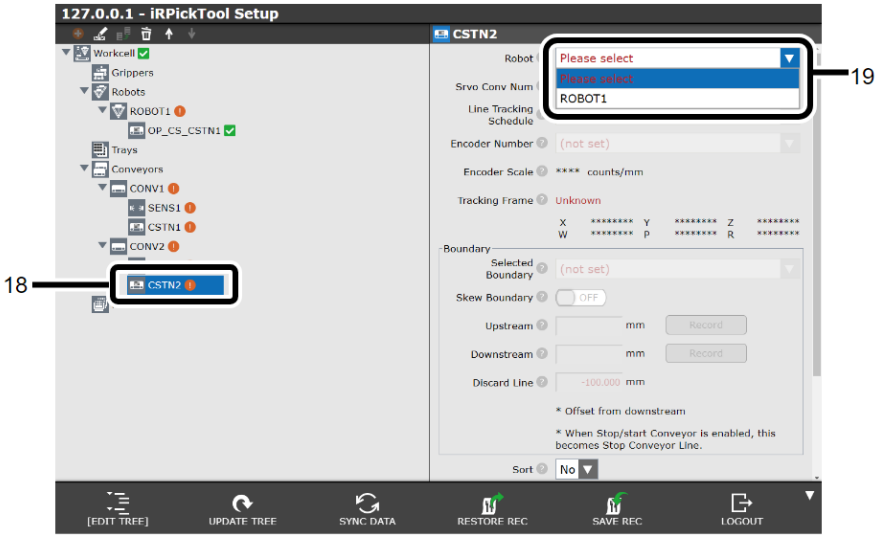
When a new conveyor is created, a sensor and conveyor station will be automatically created.



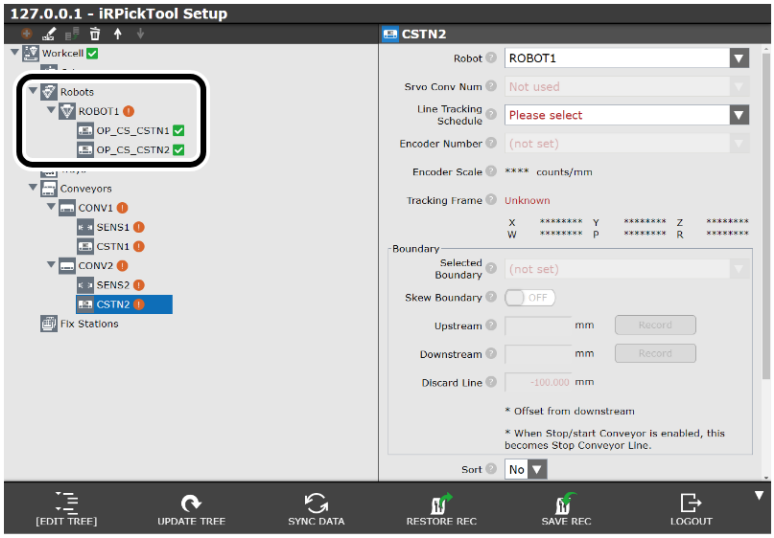
18 Select the conveyor station [CSTN2] that was automatically created in step 17.

19 Select a robot from the [Robot] drop-down list.

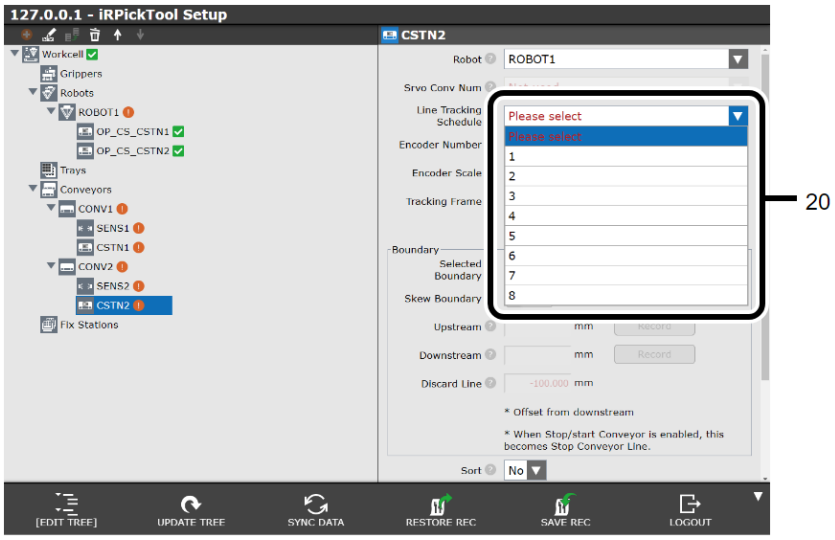
Select the robot name that was input in step 4.



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

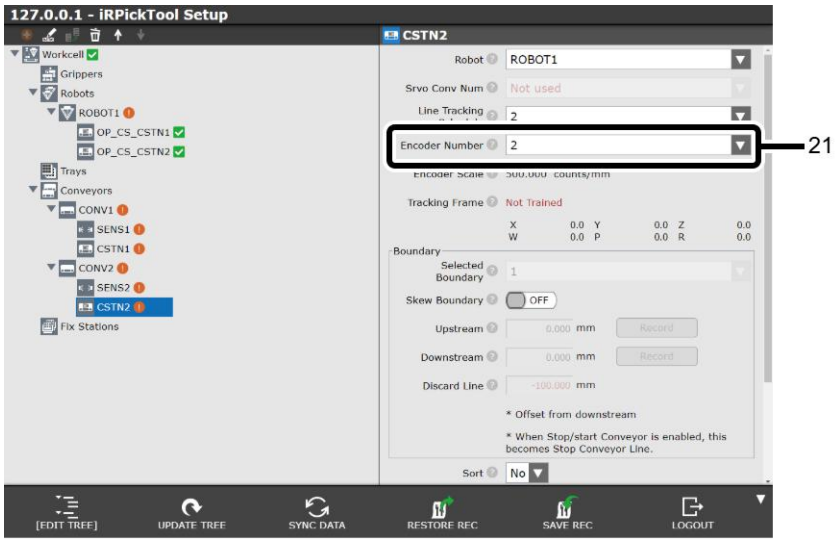


20 **Select a number from the [Line Tracking Schedule] drop-down list. Select ‘2’.**



21 **Select a number from the [Encoder Number] drop-down list.**

Select ‘2’.



3.3.2. Setting up a gripper

Set up a gripper and zones.

For the gripper, enter an index delay, etc. that is suitable for the pick/place operation. For the setup of a zone, which is a child object of a gripper, enter items such as tool frame, zone activation I/O, zone part presence I/O, etc.

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

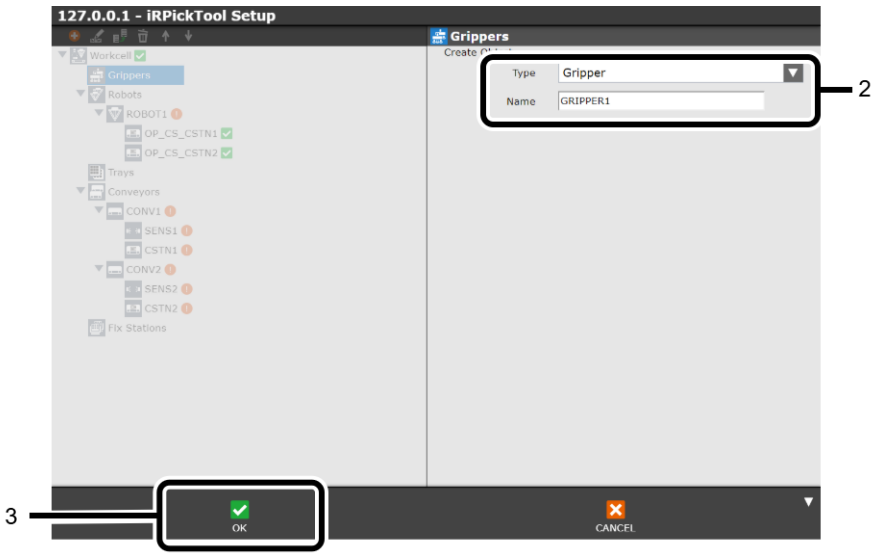


2 Input the name of the gripper in [Name].

We recommend using the name that is displayed as the initial value just as it is.

Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

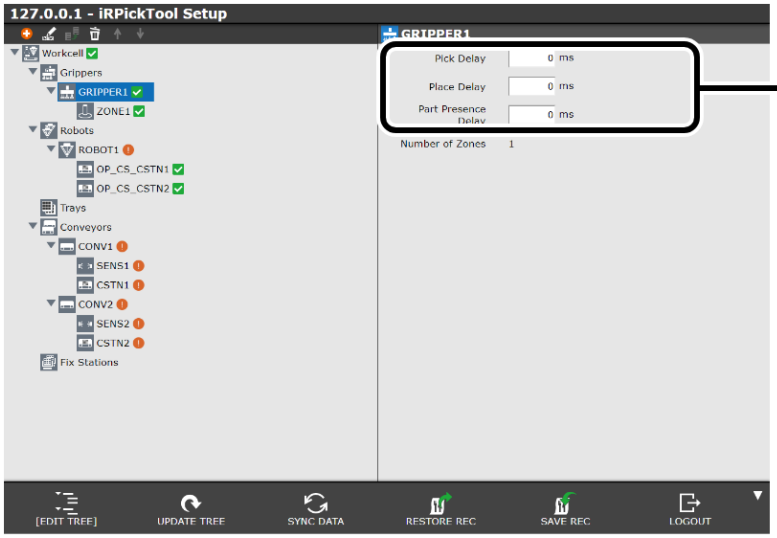
3 Click [OK].



A gripper is added. Under [Grippers] in the tree view, a zone will be automatically added.

4 Set [Pick Delay], [Place Delay], and [Part Presence Delay].

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



5 Select [ZONE].

The zone setting screen will appear.

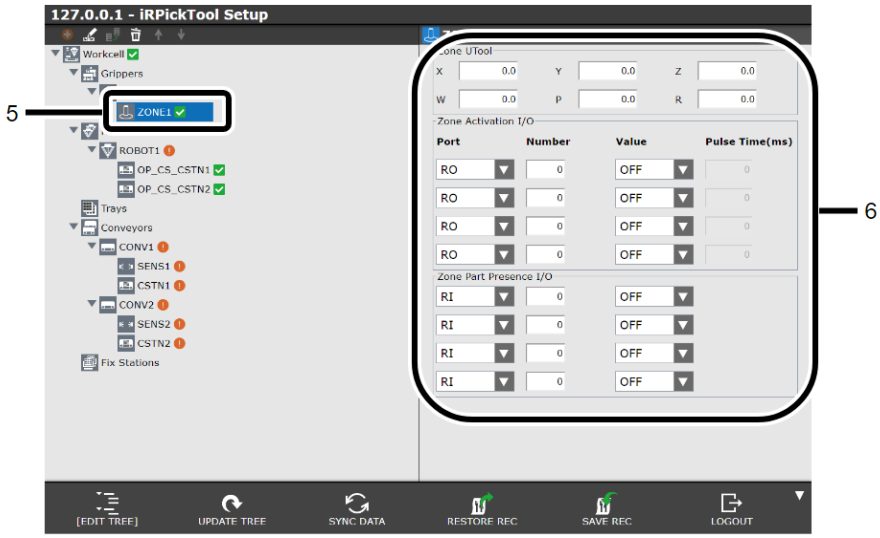
6 Set [Zone UTool], [Zone Activation I/O], and [Zone Part Presence I/O].

Zone UTool: Each zone of the gripper has a UTOOL value.

* It is OK to leave Zone UTool unset at this stage. This is set in "I.3.3.11. Setting up a tool frame for the hand(P.226)".

Zone Activation I/O: The output value is [ON], [OFF], or [Pulse]. If PULSE, is specified as the value of an output, then specify the duration of the pulse output in seconds.

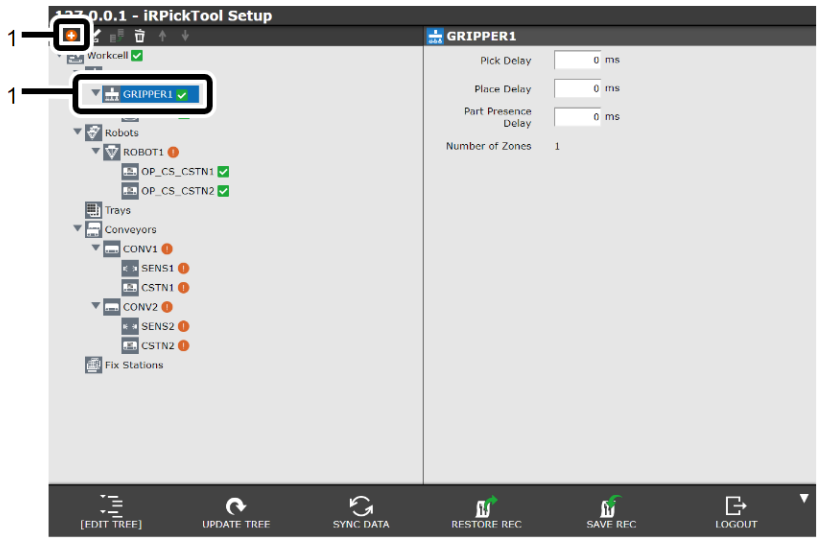
Zone Part Presence I/O: The value of the input signal is either [ON] or [OFF].



Adding a gripper zone

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

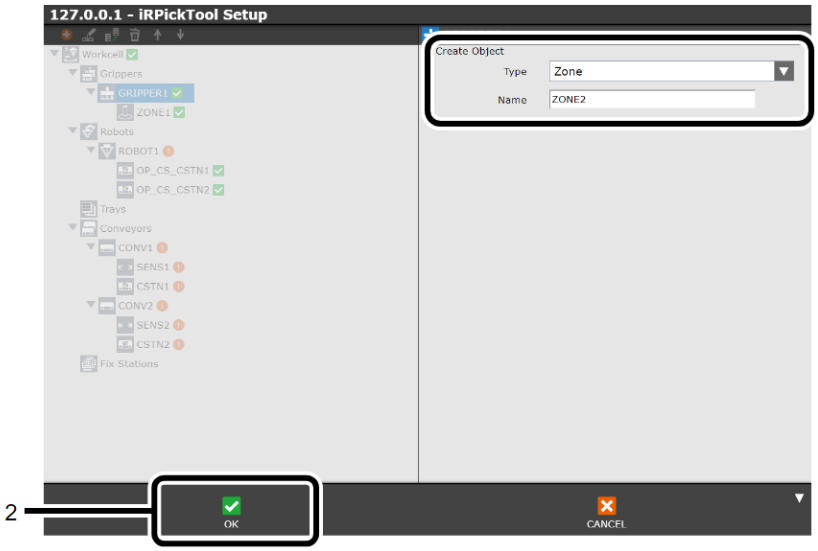
1 In the tree view, select the [GRIPPER] you want to add to, then click the [CREATE] button.



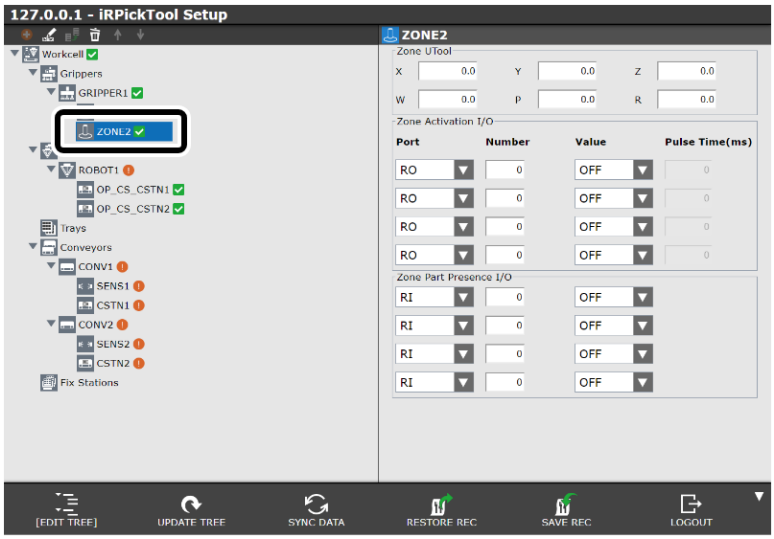
[ZONE 2] will be displayed on the create screen.

In [Name], a default name (e.g. ZONE2) is automatically provided. You can enter a different name if you wish to.

2 Click [OK].



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



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

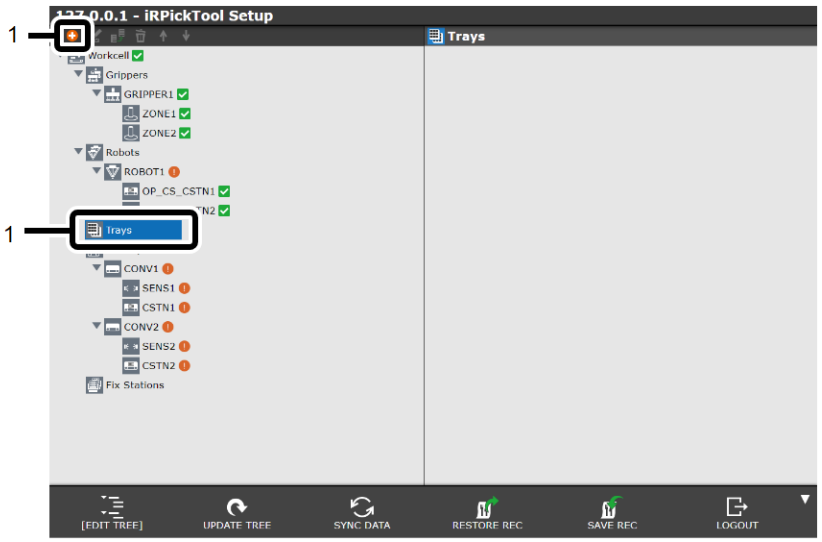
3.3.3. Setting up a tray

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

NOTE

When using trays on a conveyor, all cells within the same tray are sent to the downstream conveyor station when the most upstream cell crosses the Discard Line. Therefore, it is not possible to perform operations simultaneously at multiple conveyor stations on a long tray spanning several stations.

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



2 Enter the name of the tray in [Name].

We recommend using the name that is displayed as the initial value just as it is.

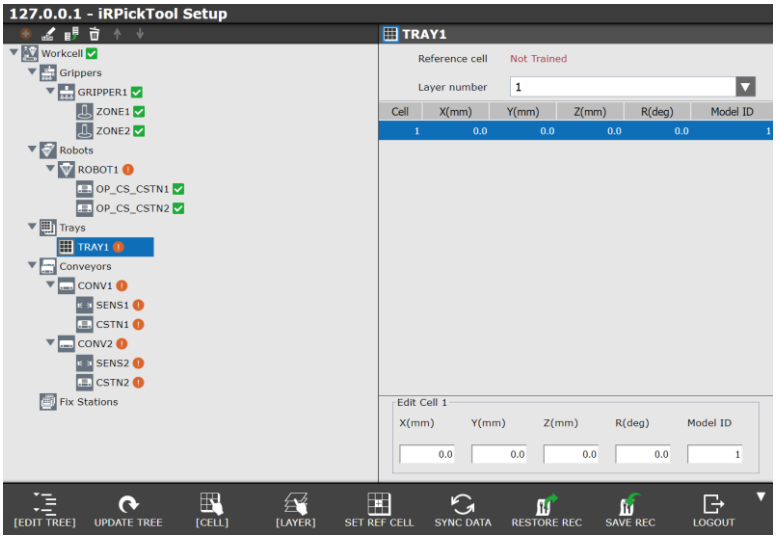
Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

3 Click [OK].

The robot object is created.



After a while, the following screen appears.



4 Decide on a tray frame.

The origin can be at any position, such as one corner of the box.

Aligning the flow direction of the conveyor with the X direction of the tray frame will make things easier to understand. (Other directions are also OK.)

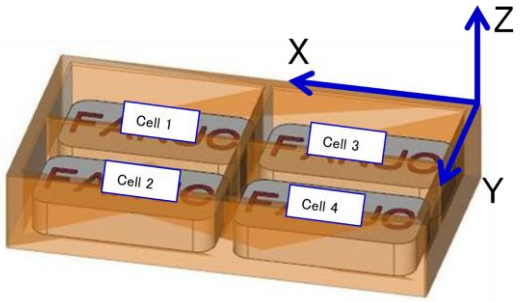
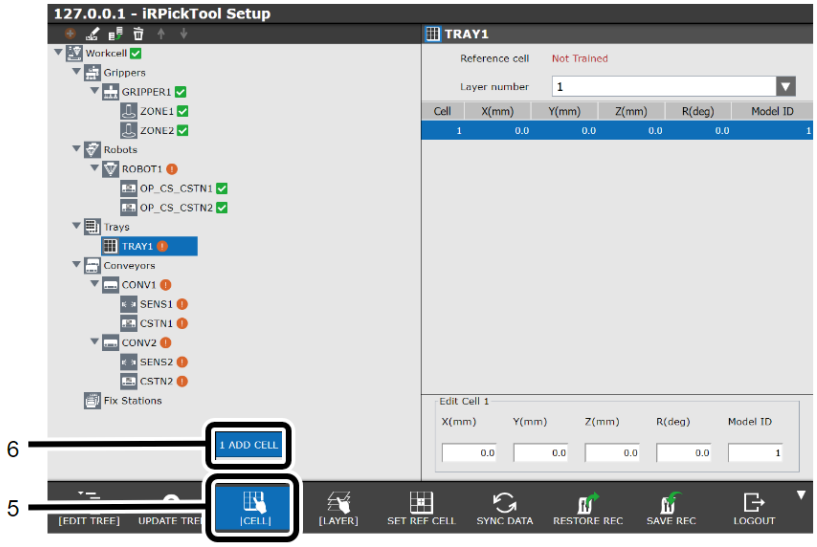


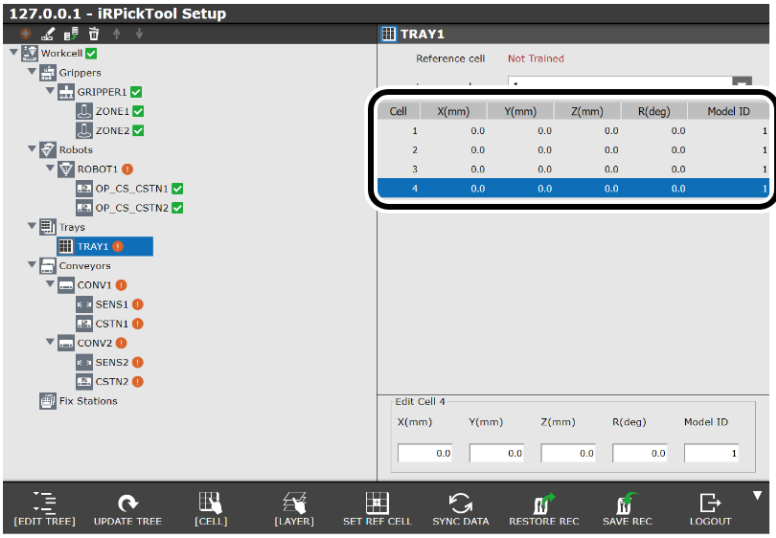
Fig. 3.3.3-1 Deciding on the origin of a tray

5 Click [CELL].

6 Select [ADD CELL] from the menu.

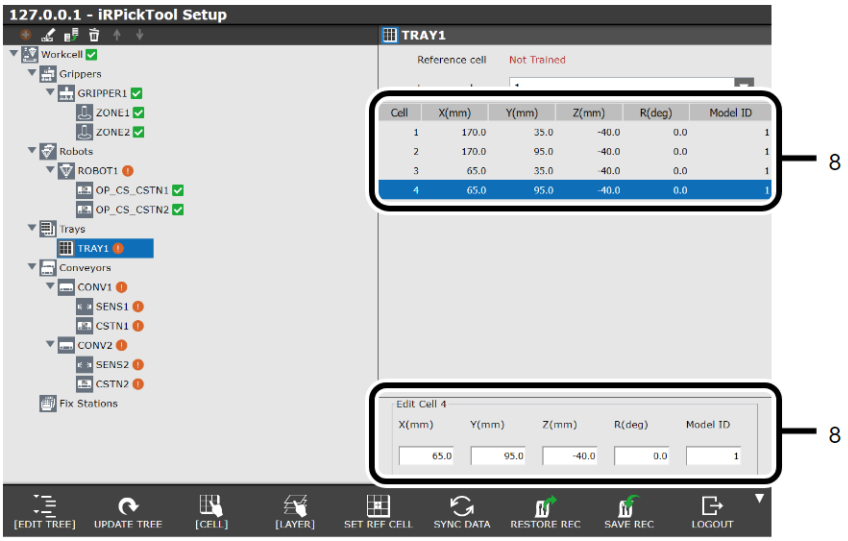


7 Add the cells for the first stage.



8 Enter the coordinates of each cell on the cell edit screen.

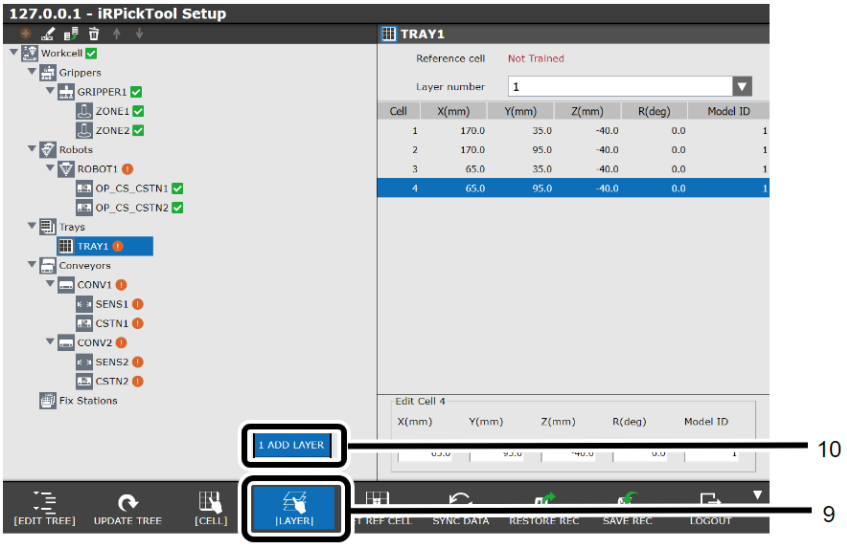
Select the row of the cell that you want to edit, and enter the coordinates in the frame at the bottom of the screen. Enter the coordinates of a cell in the tray frame coordinate system. For details, see "I.5.3. Setting up a tray(P.269)".



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

9 Click [LAYER].

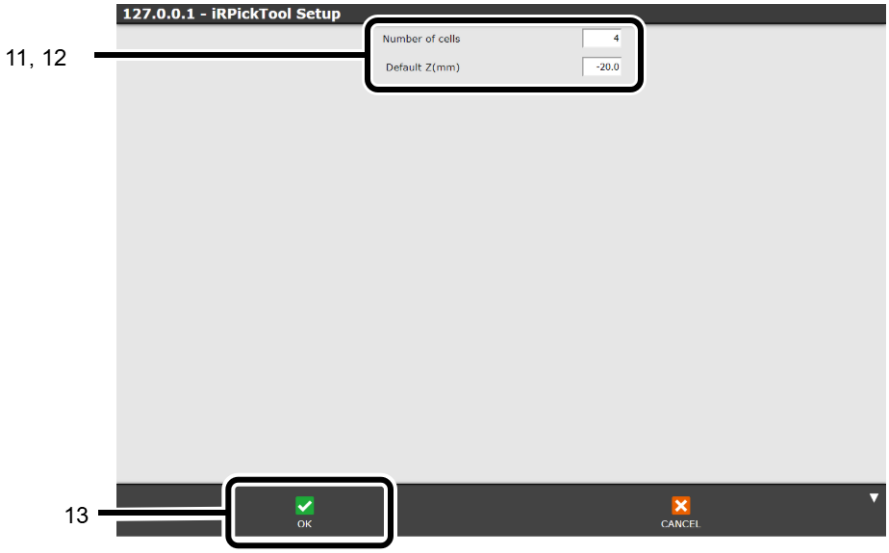
10 Select [ADD LAYER] from the menu.



11 **Enter the number of cells in the second stage in [Number of cells].**

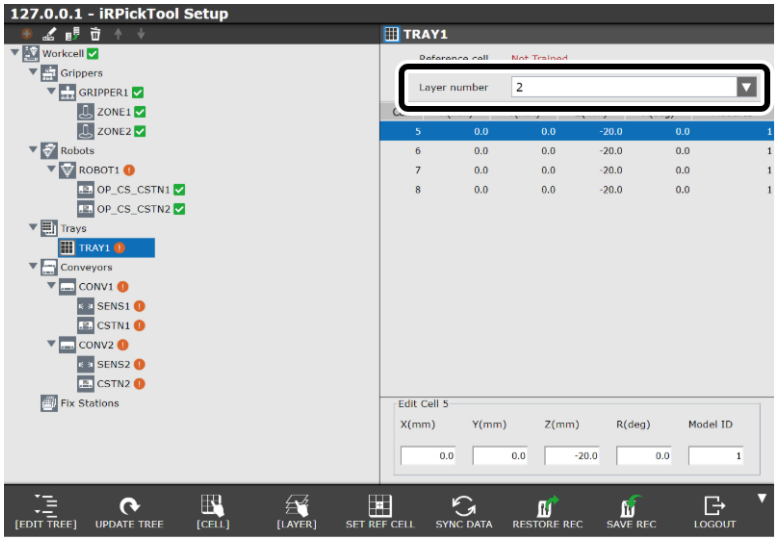
12 **Enter a Z value (mm) in [Default Z(mm)].**

13 **Click [OK].**



After a while, the following screen appears.

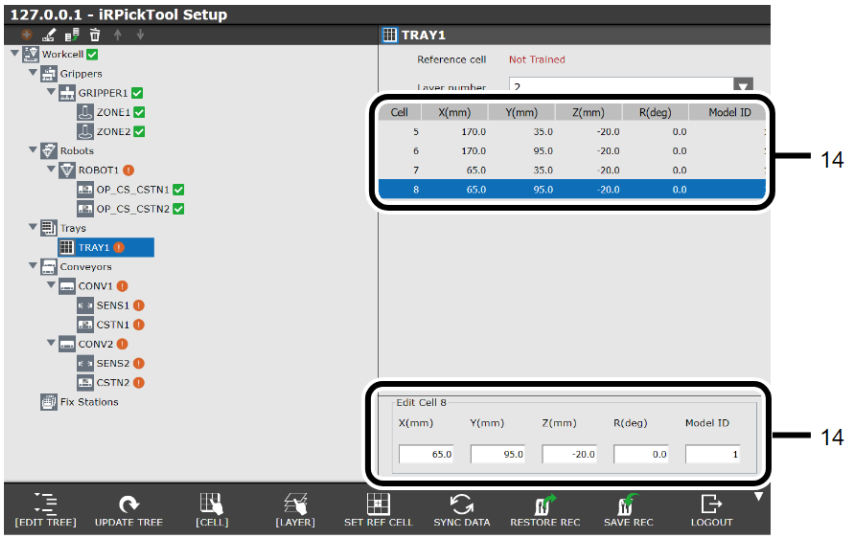
Layer ‘2’ is created in [Layer number].



14 Enter the coordinates of each cell in Layer 2 on the cell edit screen.

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

The reference cell can remain untrained at this point. It is taught automatically in "I.3.3.12. Setting up a reference position and teaching a robot position(P.230)".



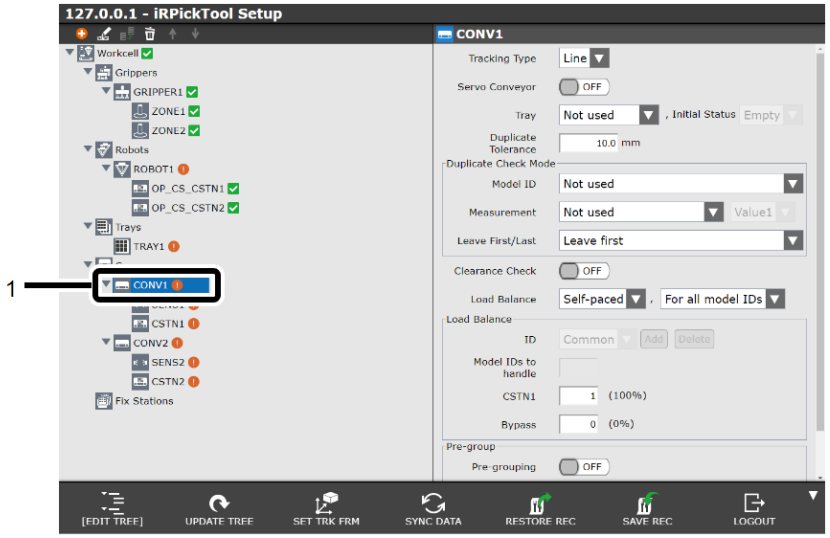
3.3.4. Setting up a conveyor

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

3.3.4.1. Setting up an infeed conveyor

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

1 Select the conveyor [CONV1].



The conveyor setup screen will appear.

To set up a conveyor, the following data that appears as the initial values can be used as it is.

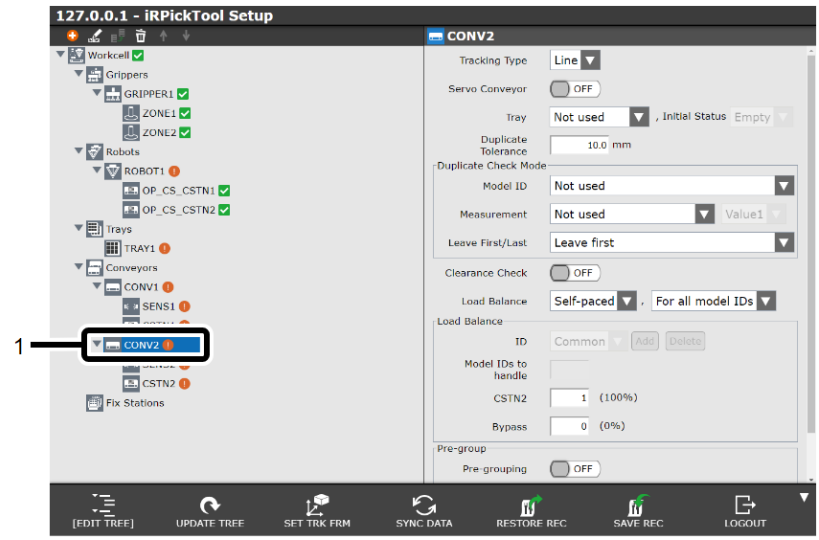
[Tracking Type]:	Line
[Servo Conveyor]:	[Disable]
[Tray]:	Not used
[Duplicate Tolerance]:	10.0 mm

[Load Balance]: Self-paced, For all model IDs

3.3.4.2. Setting up an outfeed conveyor

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

1 Select the conveyor [CONV2].



The conveyor setup screen will appear.

Select [TRAY1] in [Tray] and [Empty] in [Initial Status].

For other settings for conveyors, the following data that appears as the initial values can be used as it is.

[Tracking Type]: Line
[Servo Conveyor]: [Disable]
[Duplicate Tolerance]: 10.0 mm
[Load Balance]: Self-paced, For all model IDs

3.3.5. Setting up a tracking frame

Set up a tracking frame.

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

The menu appears.

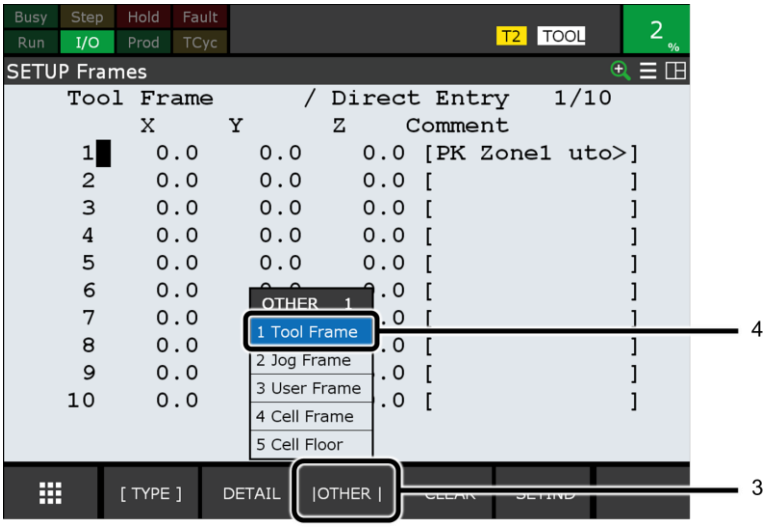
2 Select [SETUP] → [Frames] from the menu.



3 Press F3 [OTHER].

4 Select [Tool Frame] from the menu.

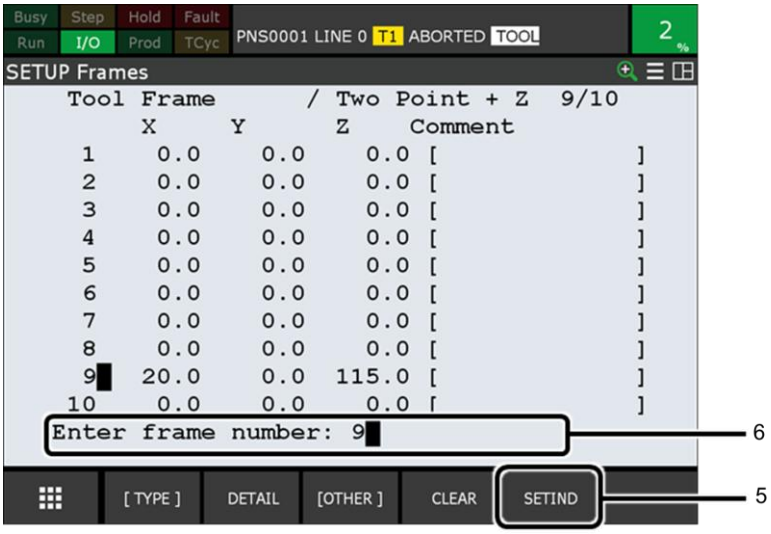
The Tool Frame list screen appears.



5 Press F5 [SETIND].

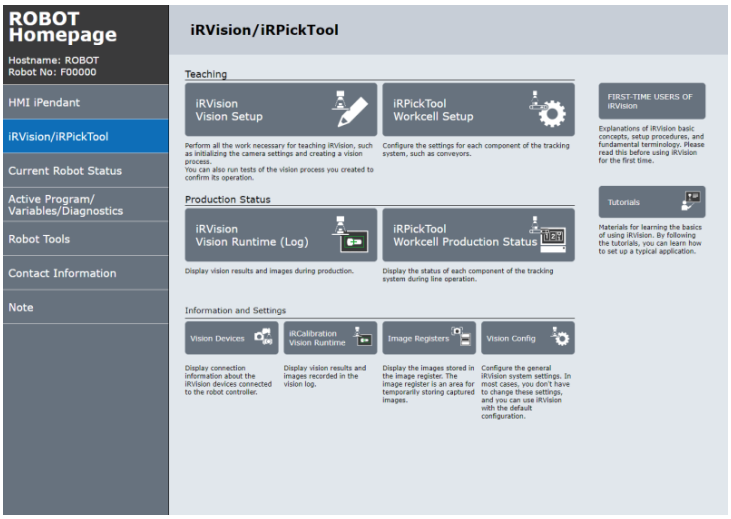
6 Enter the frame number that has been used for the pointer tool.

This procedure is explained using Tool 9 as an example.



7 **Open the robot homepage.**

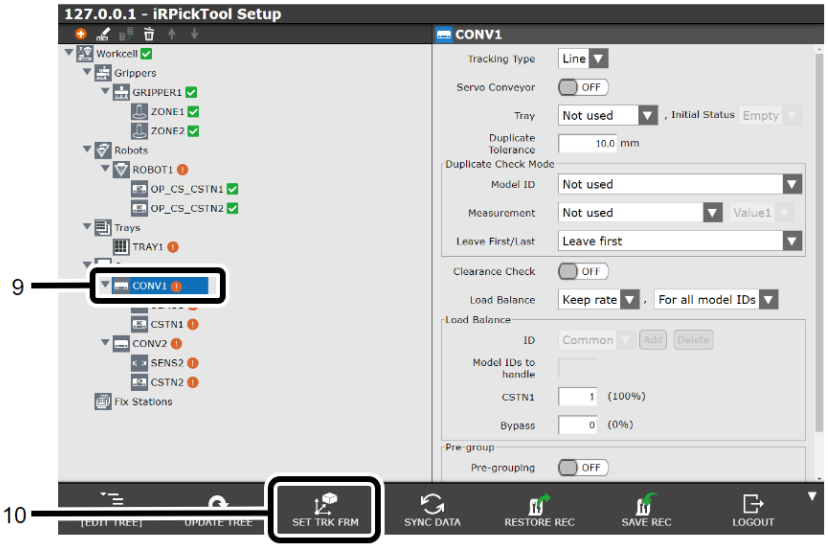
8 **Select [iRVision/iRPickTool] in the menu area and click [iRPickTool System Setup] in the content display area.**



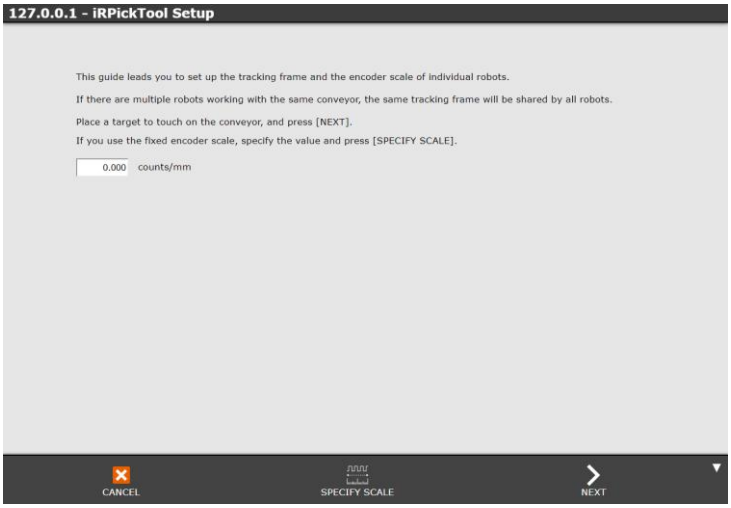
9 **First, set up the tracking frame of the infeed conveyor.**

Select conveyor [CONV1].

10 **Click [SET TRK FRM].**



After a while, the following screen appears.



⚠CAUTION

When using the αiA4000S, “IRPK-063 Encoder count does not match on all robots” may occur even when the conveyor is stationary, due to high-resolution encoder pulses. In this case, stop the conveyor and retry the operation several times.

11 Set a calibration grid on the upstream side of the conveyor, in accordance with the procedure shown on the screen.

Fix it with tape to prevent it from moving.

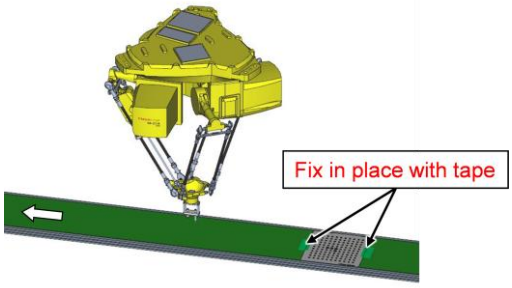


Fig. 3.3.5-1 Setting a calibration grid on the upstream side

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

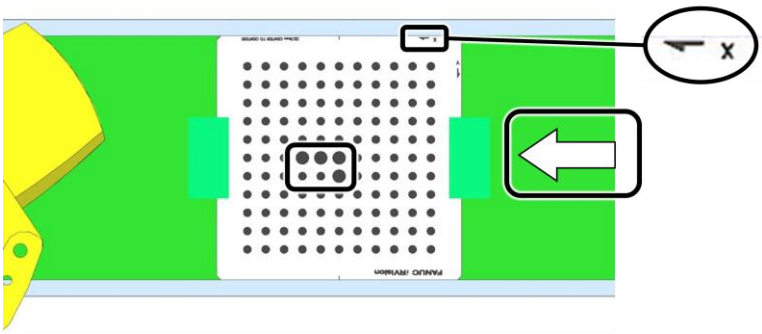
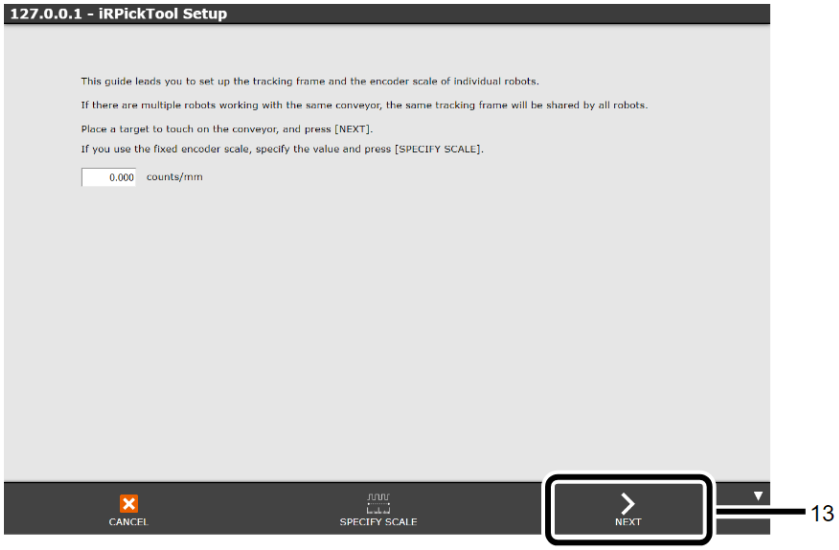
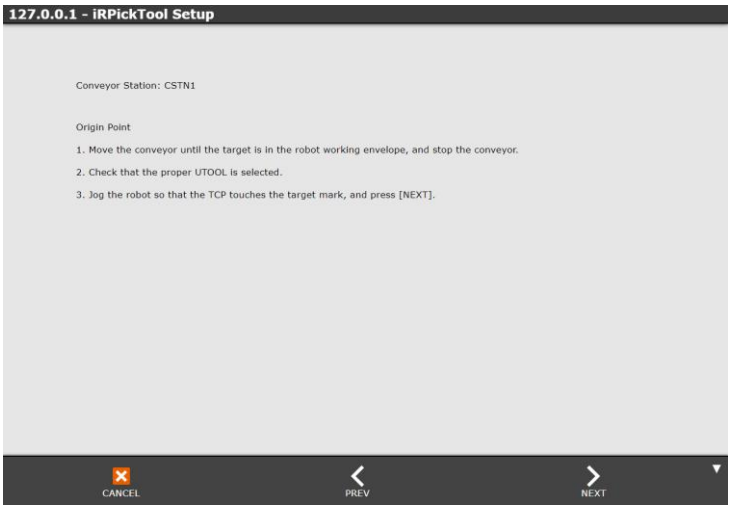


Fig. 3.3.5-2 Align the setting direction

13 Click [NEXT].



After a while, the following screen appears.



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

15 **Touch up the origin of the calibration grid.**

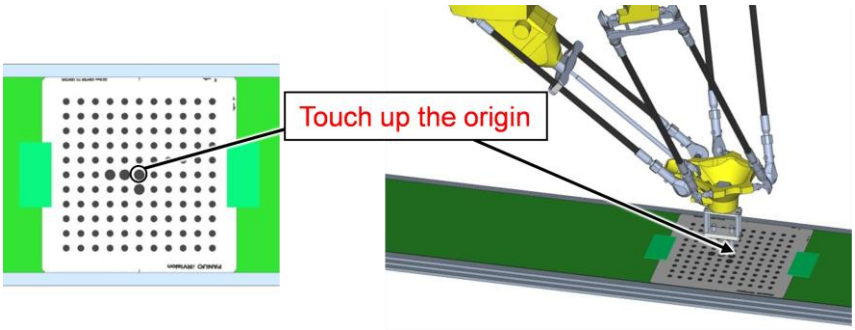
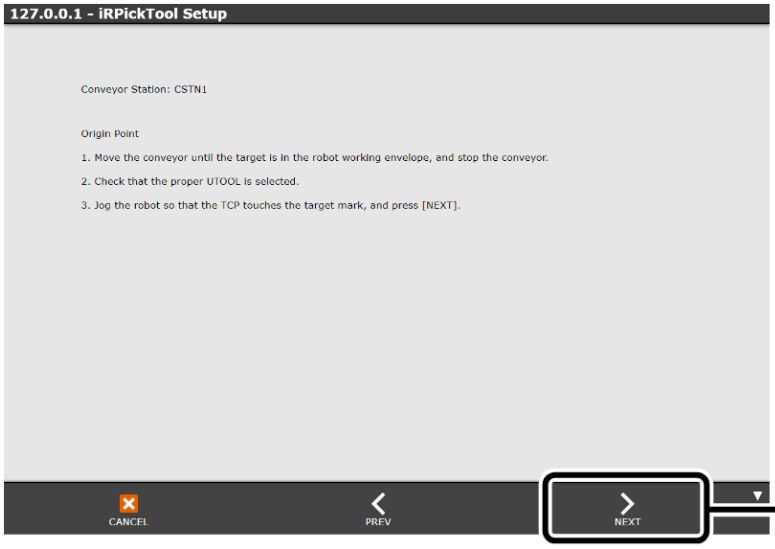
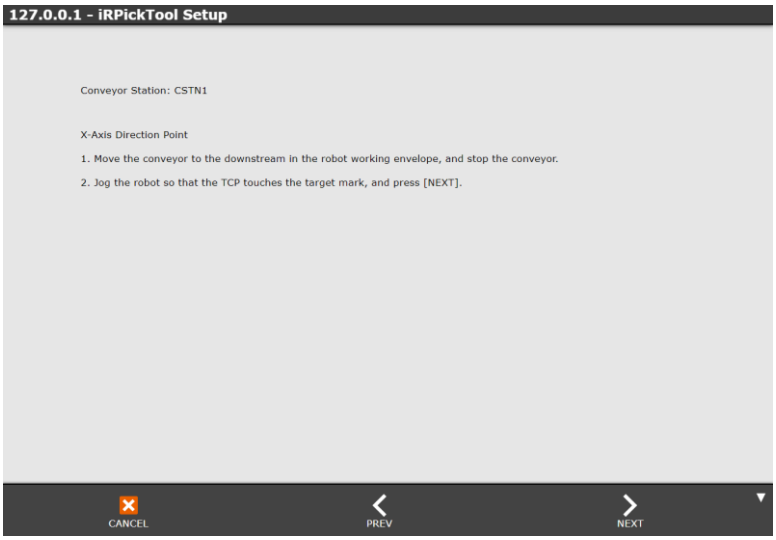


Fig. 3.3.5-3 Touching up the origin of the calibration grid

16 **Click [NEXT].**



After a while, the following screen appears.



17 Move the robot up until it is above the surface of the conveyor.

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

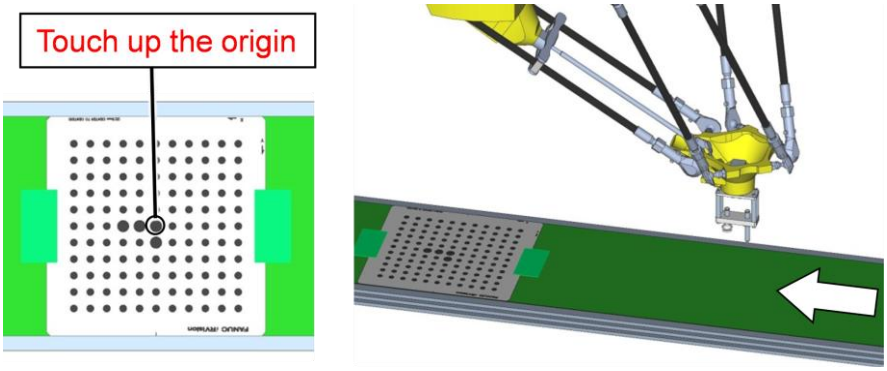


Fig. 3.3.5-4 Stopping the conveyor

19 Touch up the origin of the calibration grid.

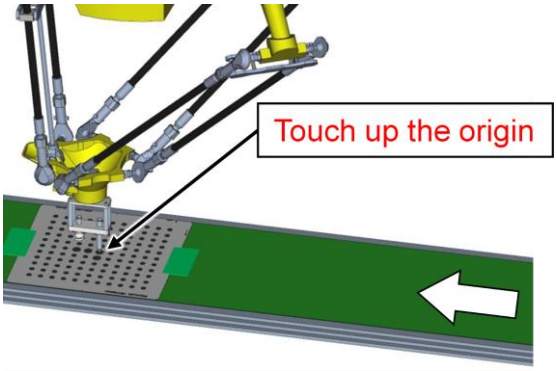
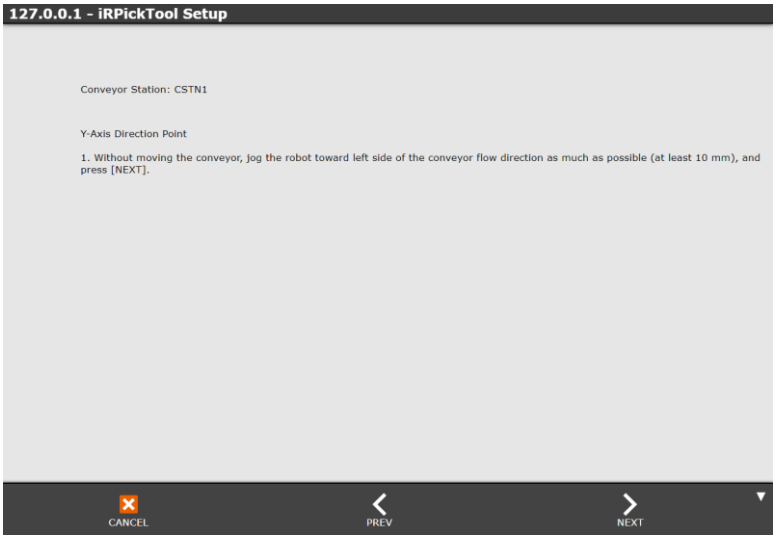


Fig. 3.3.5-5 Touching up the origin of the calibration grid

20 Click [NEXT].



After a while, the following screen appears.



21 Touch up the Y direction of the calibration grid.

Do not move the conveyor at this time.

This is not the Y direction of the robot. It is on the left side relative to the conveyor's movement direction.

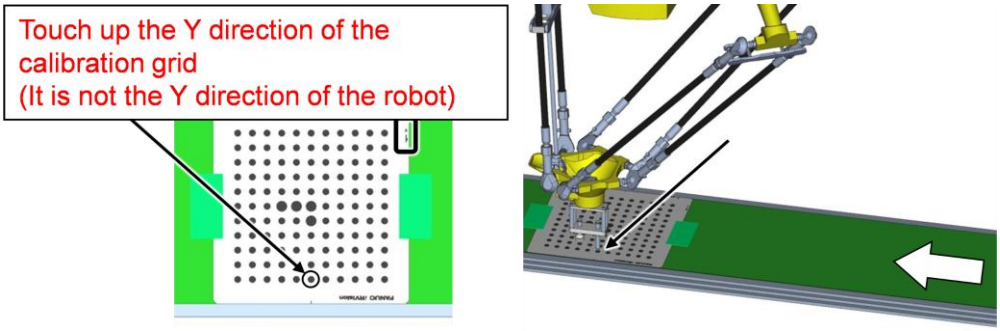


Fig. 3.3.5-6 Touching up the Y direction of the calibration grid.

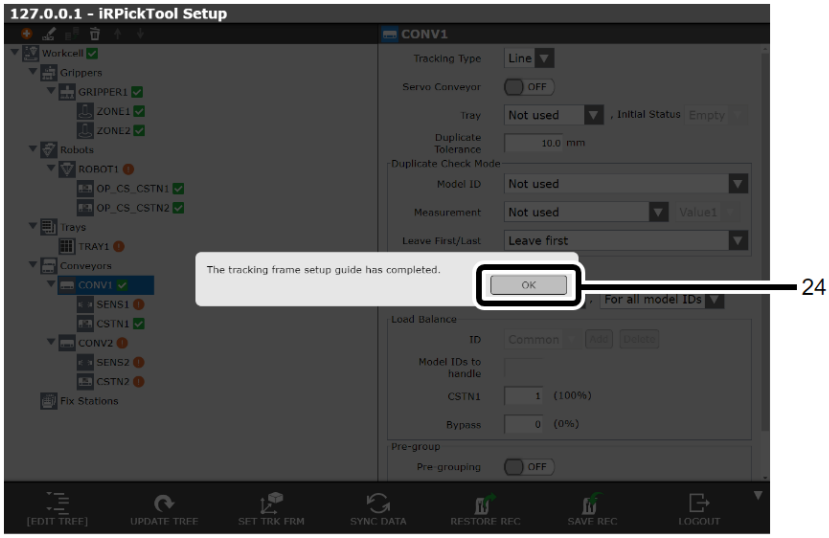
22 **Click [NEXT].**



23 **Click [NEXT].**



24 **Click [OK].**



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

25 Next, set up the tracking frame of the outfeed conveyor.

Select conveyor [CONV2].

26 Repeat steps 10 to 24 to set up the tracking frame of the outfeed conveyor.

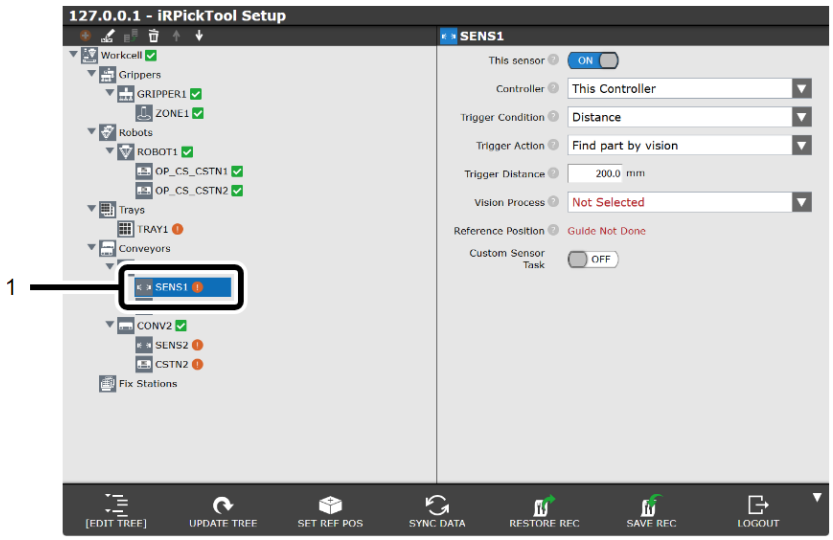
3.3.6. Setting up a sensor

Set up a sensor.

3.3.6.1. Setting an infeed conveyor sensor

Set up an infeed conveyor sensor. These are the settings for [Distance] + [Find part by vision].

1 Select the sensor [SENS1].



2 Set the following items.

[This Sensor]: [ON]
[Controller]: [This Controller]
[Trigger Condition]: [Distance]

[Trigger Action]: [Find part by vision]

When [Trigger Condition] and [Action] are changed, setting items change accordingly.

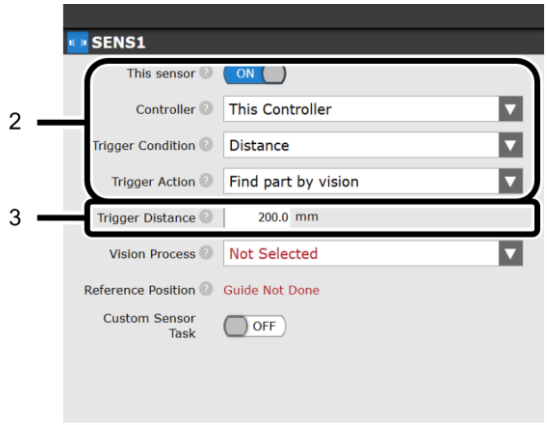
3 Set the following items.

[Trigger Distance]: Set how many mm the conveyor advances each time for vision detection to be performed.

* It is recommended that the trigger distance be set such that a workpiece can be detected twice in the camera FOV.

[Vision Process]: You do not have to set this at this stage.

* Set this up immediately before setting up reference positions. For the settings, see "I.3.3.12.1. Setting up a reference position for the infeed conveyor, and teaching a robot position(P.230)".



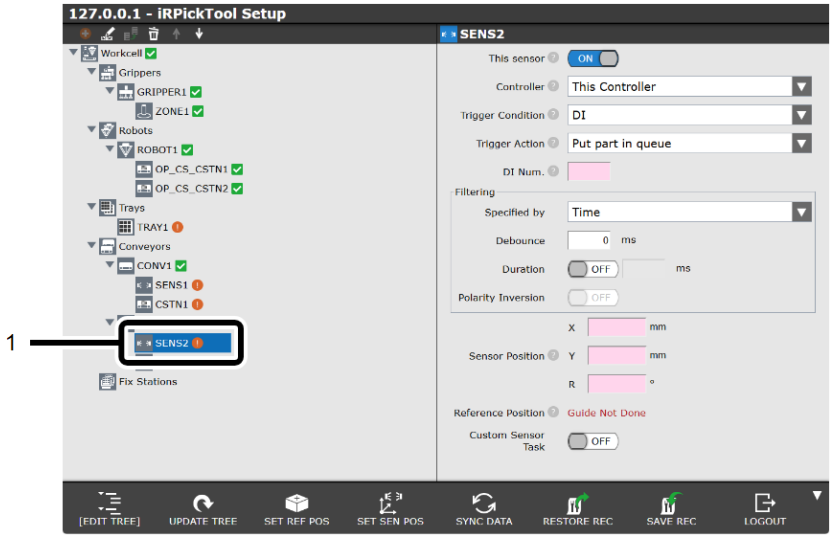
3.3.6.2. Setting up an outfeed conveyor sensor (In the case of [DI] / [RI] + [Put part in queue])

Set up an outfeed conveyor sensor. The following explanation uses the sensor settings for [DI] + [Put part in queue] as an example.

For [RI] + [Put part in queue], interpret "DI" as "RI" in the explanation.

In the case of [HDI] + [Put part in queue], skip this section and go to the next section, "I.3.3.6.3. Setting up an outfeed conveyor sensor (In the case of [HDI] + [Put part in queue])(P.196)".

1 Select the sensor [SENS2].



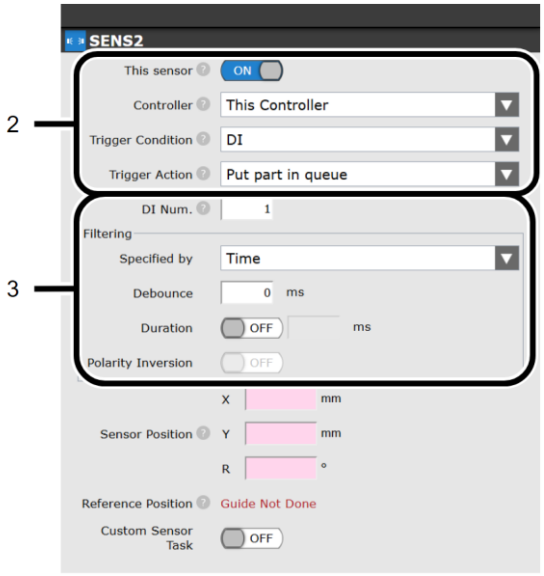
2 Set the following items.

[This Sensor] : [ON]
[Controller]: [This Controller]
[Trigger Condition]: [DI]
[Trigger Action]: [Put part in queue]
When [Trigger Condition] and [Action] are changed, setting items change accordingly.

3 Set the following items.

[DI Num]: Enter the DI number that has been assigned to the sensor input.
[Specified By]: Select the unit for specifying Debounce and Duration.
[Debounce]: Enter the time you want to disable input for (to prevent chattering).
[Duration]: Enable this setting to remove incidental noise, and enter a value longer than the duration or distance over which the noise occurs (for details, refer to "I.8.14.3. Duration(P.676) ").
[Polarity Inversion]: Enable this setting to use polarity inversion (for details, refer to "I.8.14.4. Polarity Inversion(P.677) ").
[Sensor Position]: You do not have to set this at this stage.

* This is set in the setup of the sensor position. For the settings, see "I.3.3.7. Setting up a sensor position(P.197)".

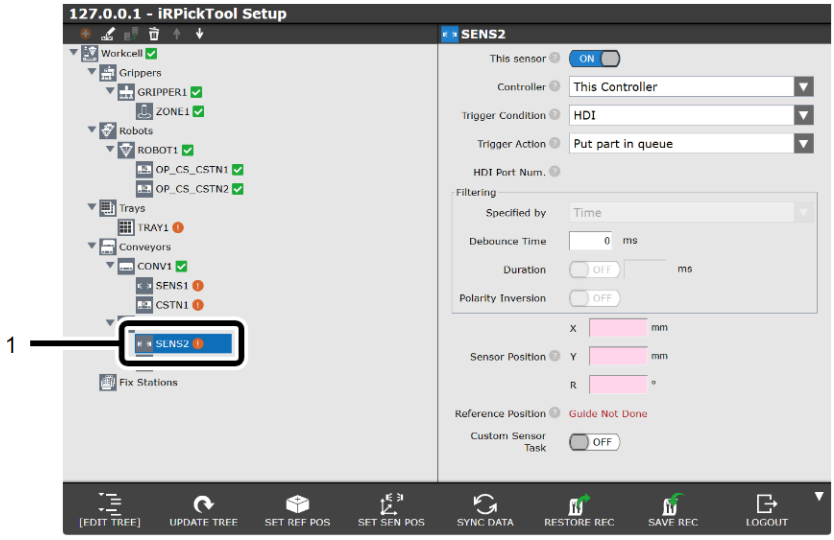


Skip the next section, "I.3.3.6.3. Setting up an outfeed conveyor sensor (In the case of [HDI] + [Put part in queue])(P.196)" and go to "I.3.3.7. Setting up a sensor position(P.197)".

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

Set up an outfeed conveyor sensor. These settings are for a sensor in the case of [HDI] + [Put part in queue]. In the case of [DI] / [RI] + [Put part in queue], see the previous section, "I.3.3.6.2. Setting up an outfeed conveyor sensor (In the case of [DI] / [RI] + [Put part in queue])(P.195)", not this section.

1 Select the sensor [SENS2].



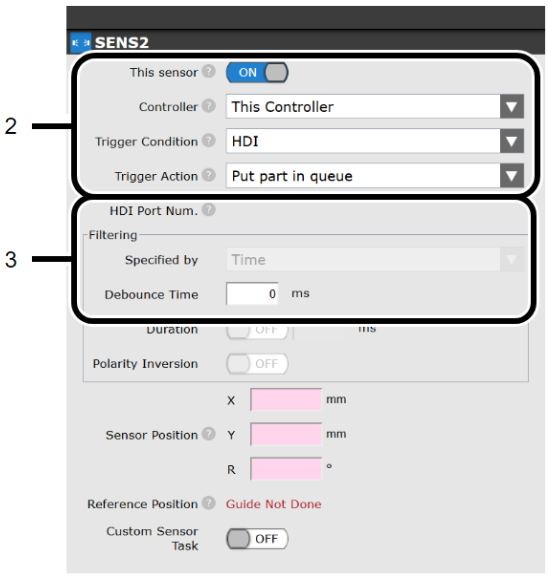
2 Set the following items.

[This Sensor]: [ON]
[Controller]: [This Controller]
[Trigger Condition]: [HDI]
[Trigger Action]: [Put part in queue]
When [Trigger Condition] and [Action] are changed, setting items change accordingly.

3 Set the following items.

[HDI Port Num.]: The number that was specified in the encoder setup will appear.
[Debounce Time]: Enter the time you want to disable input for (to prevent chattering).
[Sensor Position]: You do not have to set this at this time.

* This is set in the setup of the sensor position. For the settings, see "I.3.3.7. Setting up a sensor position(P.197)".



3.3.7. Setting up a sensor position

Set up the sensor position of the outfeed conveyor. This setup will use trays.

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

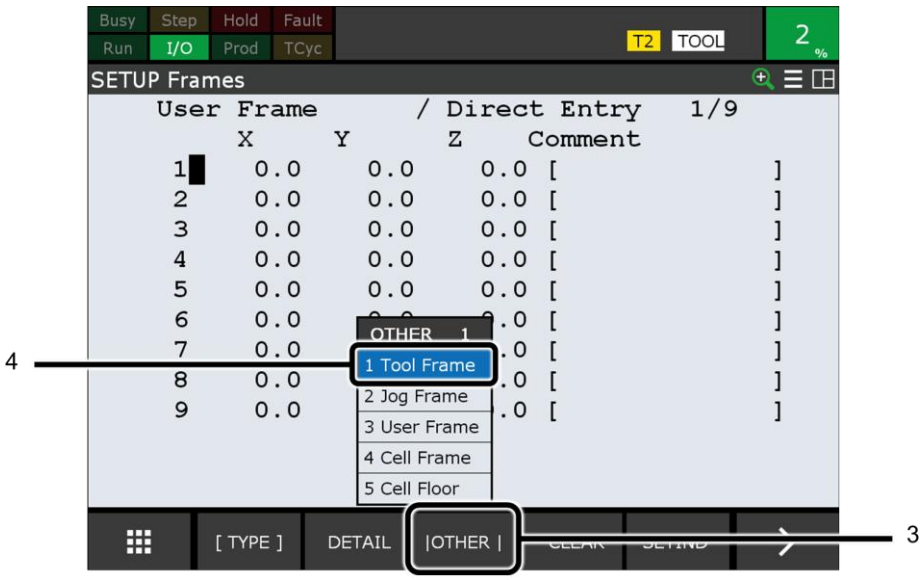
The menu appears.

2 **From the menu, select [SETUP] → [Frames].**



3 **Press F3 [OTHER].**

4 **Select [Tool Frame] from the menu.**

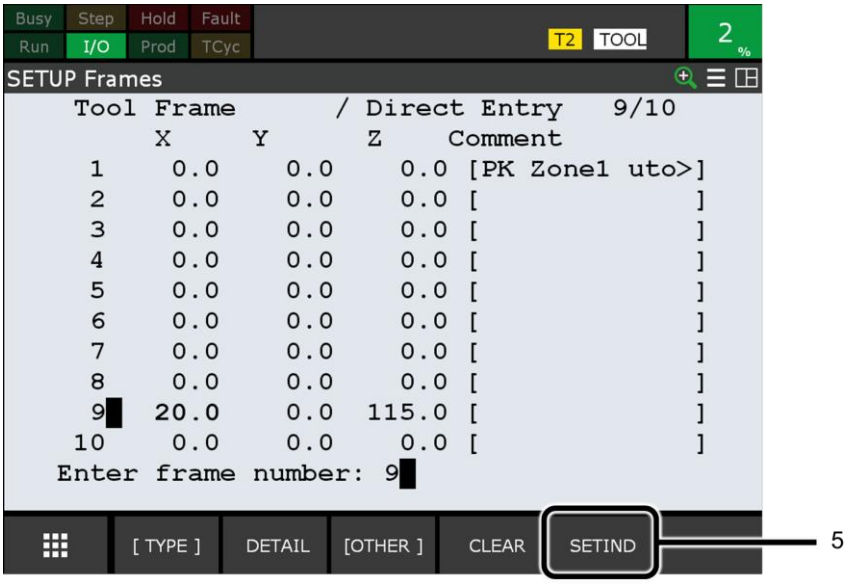


The Tool Frame list screen appears.

5 **Press F5 [SETIND].**

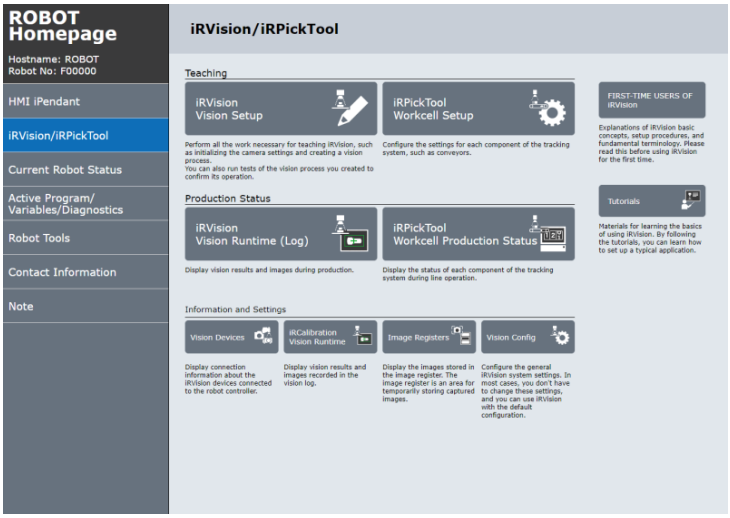
6 **Enter the frame number that has been used for the pointer tool.**

This procedure is explained using Tool 9 as an example.

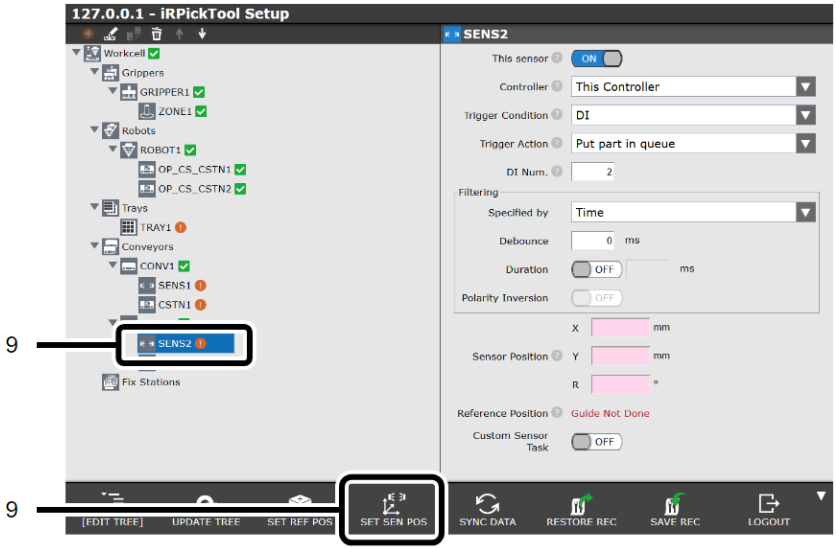


7 **Open the robot homepage.**

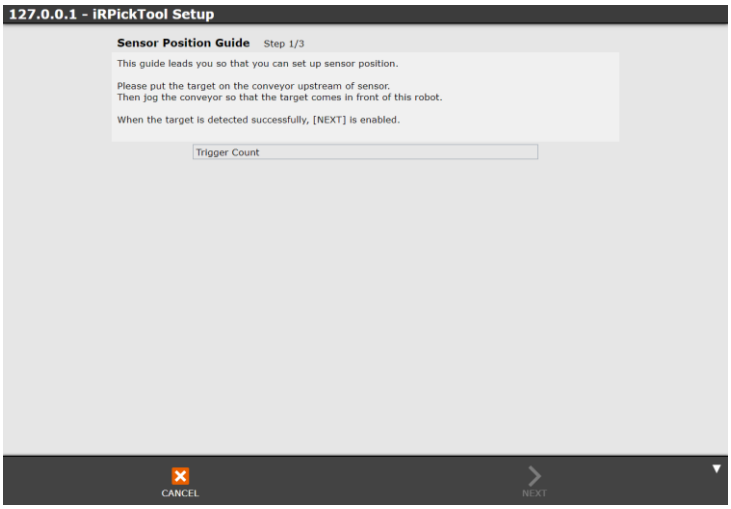
8 **Select [iRVision/iRPickTool] in the menu area and click [iRPickTool Workcell Setup] in the content display area.**



9 **Select [SENS2] in [Conveyor], then click [SET SEN POS].**



After a while, the following screen appears.



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

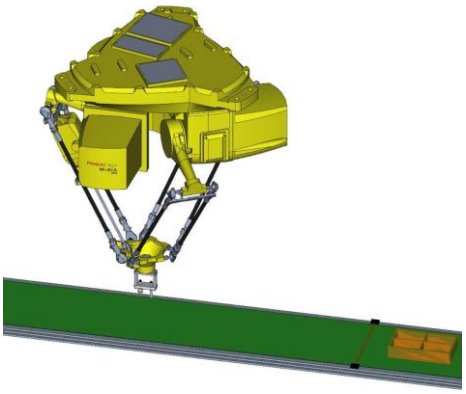


Fig. 3.3.7-1 Setting a tray upstream from the sensor

11 Move the conveyor. When the tray comes within the operating range of the robot, stop the conveyor.

The trigger count at the instant the sensor found a part will appear

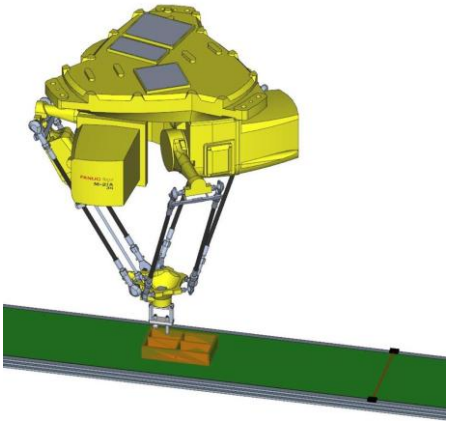
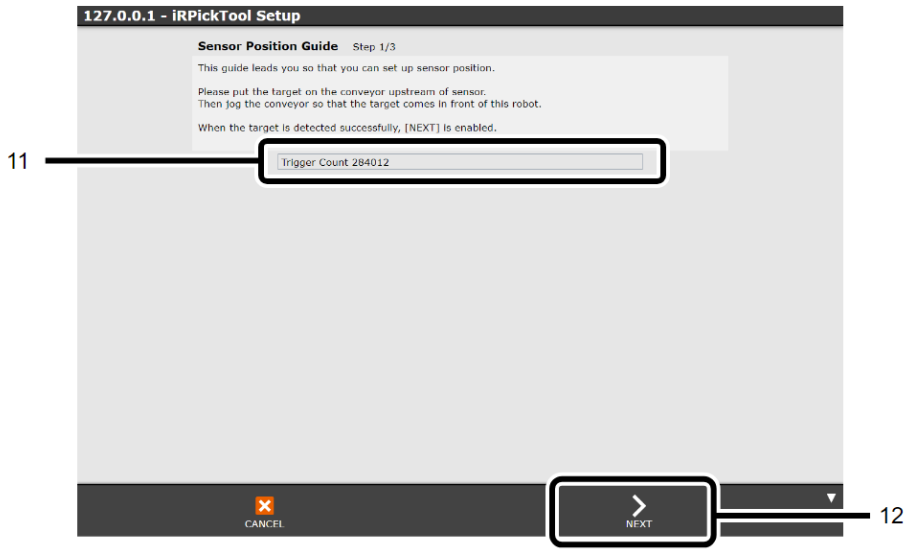
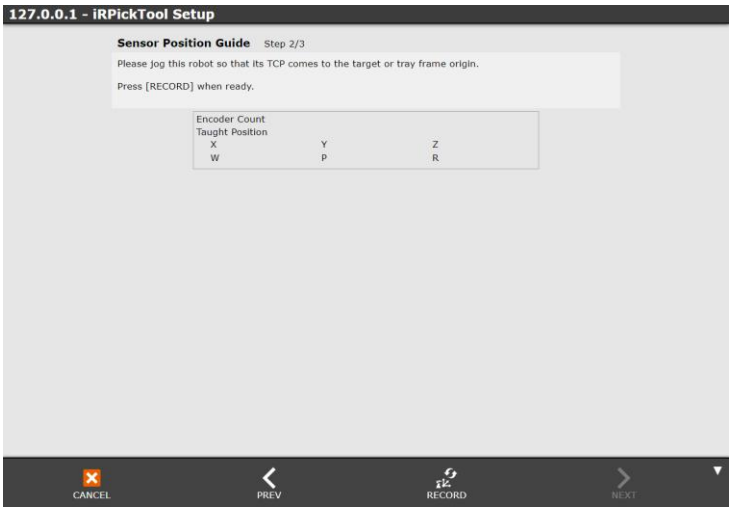


Fig. 3.3.7-2 Stopping when the tray comes within the operating range of the robot

12 Click [NEXT].



After a while, the following screen appears.



13 Touch up the origin of the tray.

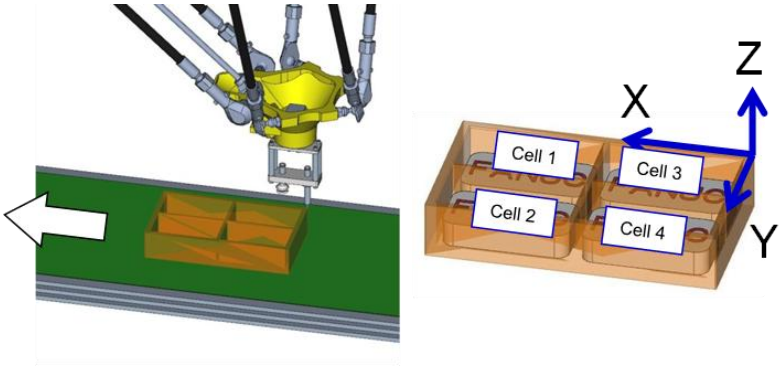


Fig. 3.3.7-3 Touching up the origin of the tray

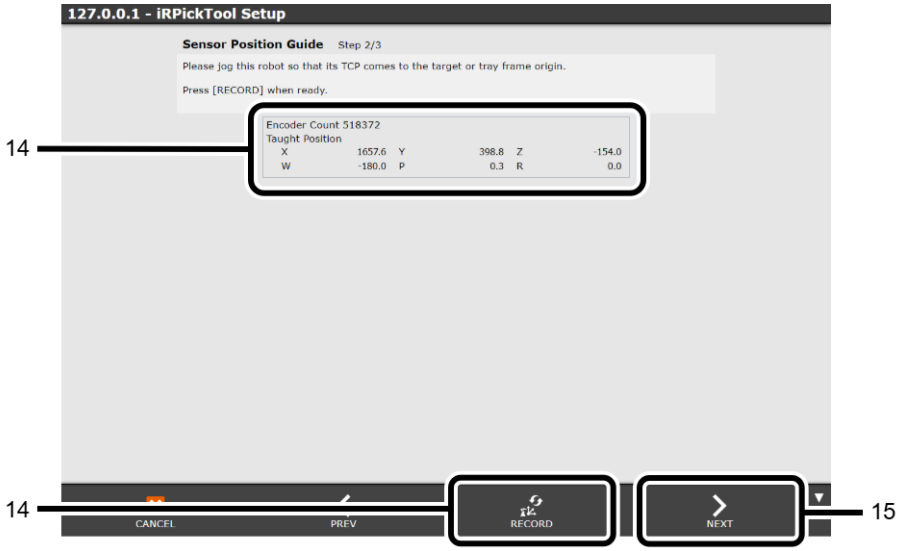
14 Click [Record].

The current encoder count and taught position will appear.

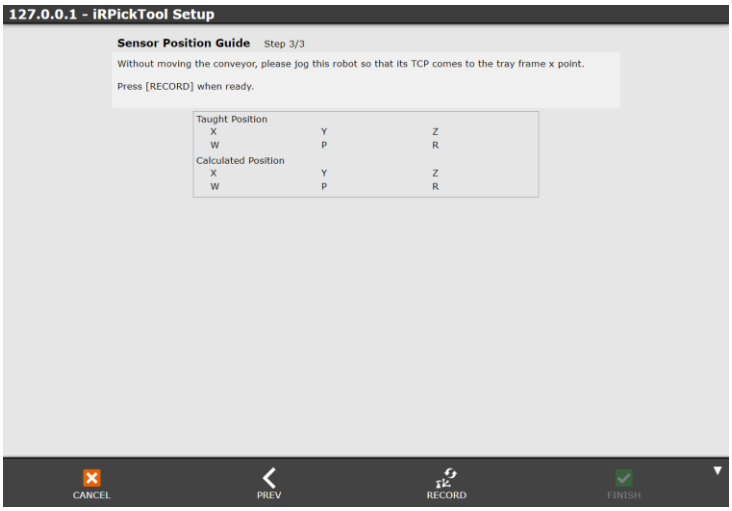
NOTE

In the case of a multi group setup, select the robot group that was touched up in Step 13 and click [Record]. This applies to all subsequent steps.

15 Click [NEXT].



After a while, the following screen appears.



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

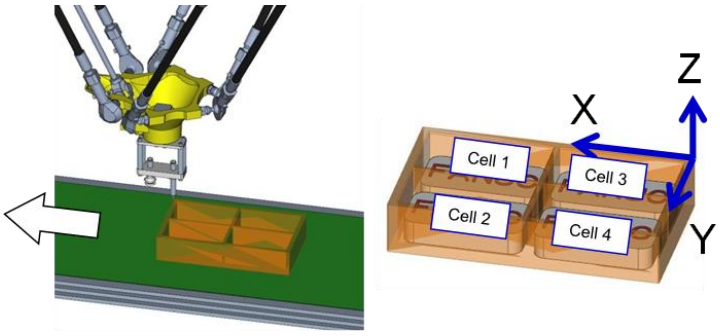
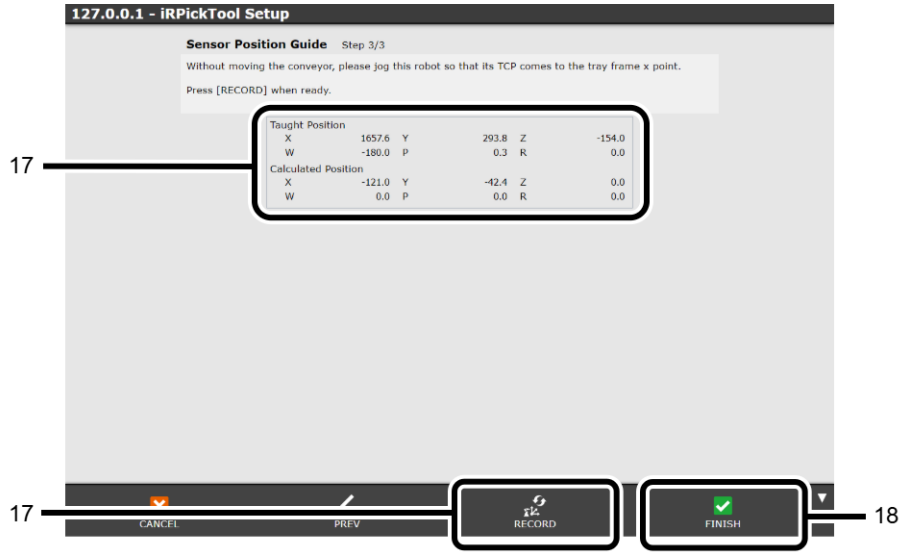


Fig. 3.3.7-4 Touching up the X direction of the tray

17 Click [Record].

The current encoder count and taught position will appear.

18 Click [FINISH].



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

3.3.8. Setting up the vision system

Set up the infeed conveyor vision.

3.3.8.1. Camera calibration

Perform camera calibration.

1 Prepare a calibration grid that is appropriate for the field of view of the camera.

Check the grid spacing at the lower right corner of the grid.

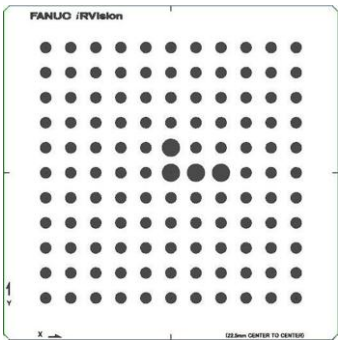


Fig. 3.3.8.1-1 Calibration grid

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

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

2 Set the calibration grid directly below the camera.

Fix it with tape to prevent it from moving.

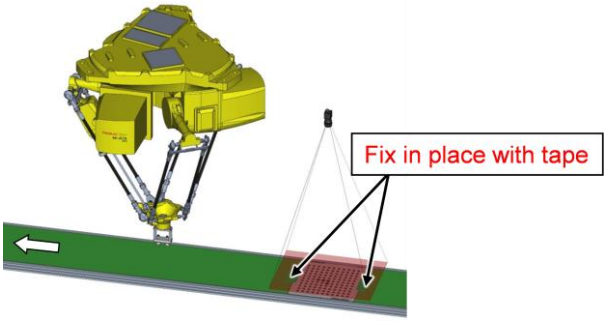


Fig. 3.3.8.1-2 Setting a calibration grid

3 The calibration grid can be set in any orientation.

For example, for line tracking, match the flow direction of the conveyor and the X direction of the calibration grid.

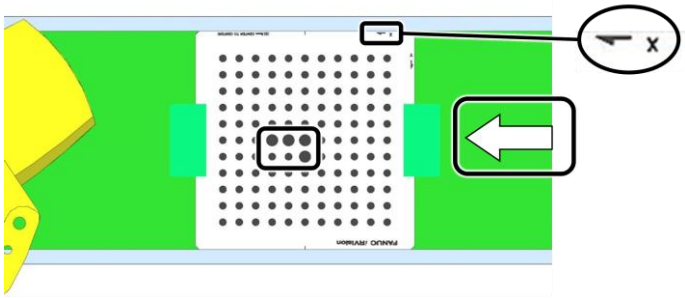


Fig. 3.3.8.1-3 Align the setting direction

4 Press the [MENU] key on the robot controller's teach pendant.

The menu appears.

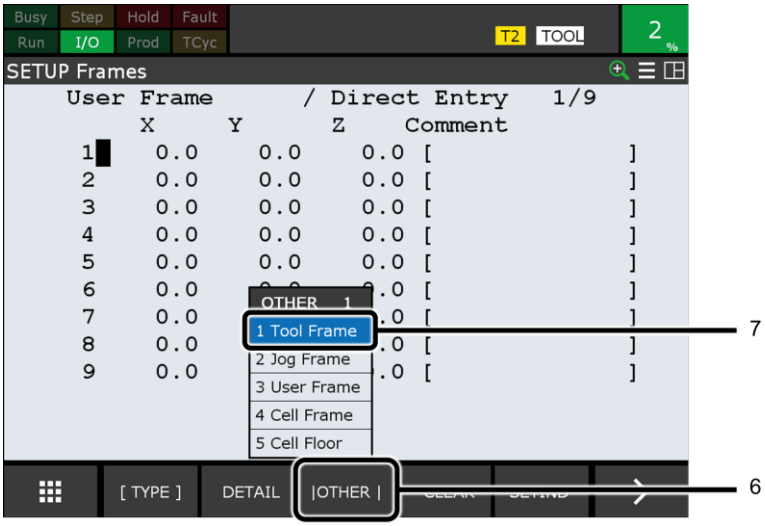
5 From the menu, select [SETUP] → [Frames].



6 Press F3 [OTHER].

7 Select [Tool Frame] from the menu.

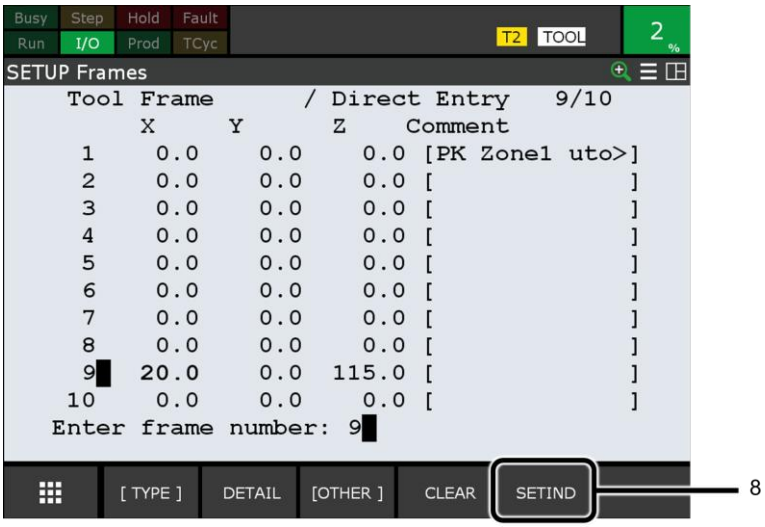
The Tool Frame list screen appears.



8 Press F5 [SETIND].

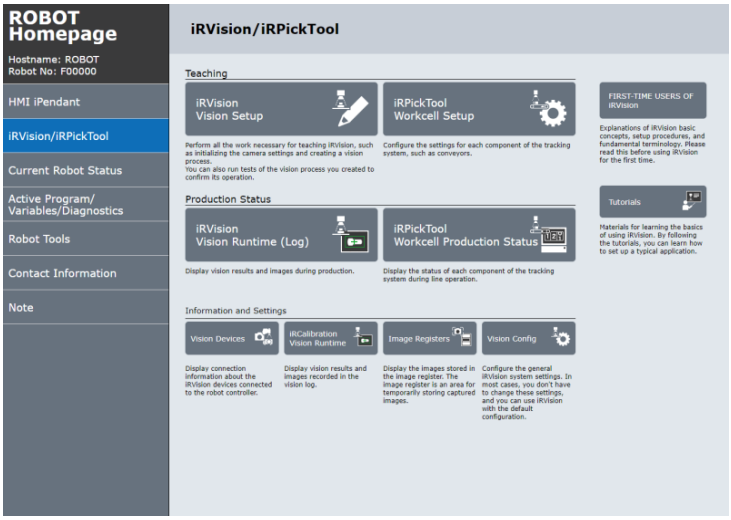
9 Enter the frame number that has been used for the pointer tool.

This procedure is explained using Tool 9 as an example.



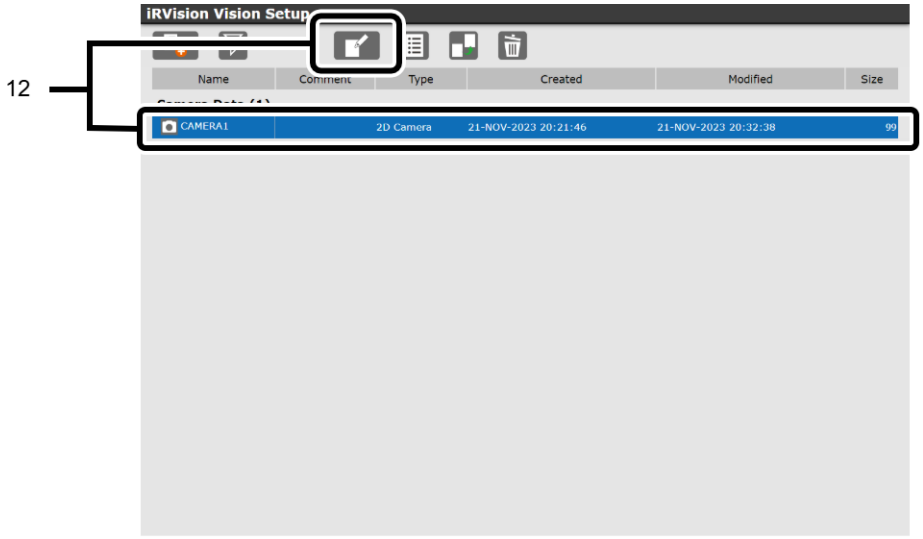
10 Open the robot homepage.

11 Select [iRVision/iRPickTool] in the menu area and click [iRVision Vision Setup] in the content display area.



The [iRVision Vision Setup] screen appears.

12 Select the camera data created in "I.3.2.2. Checking the connection of the camera(P.158)" and click [EDIT].



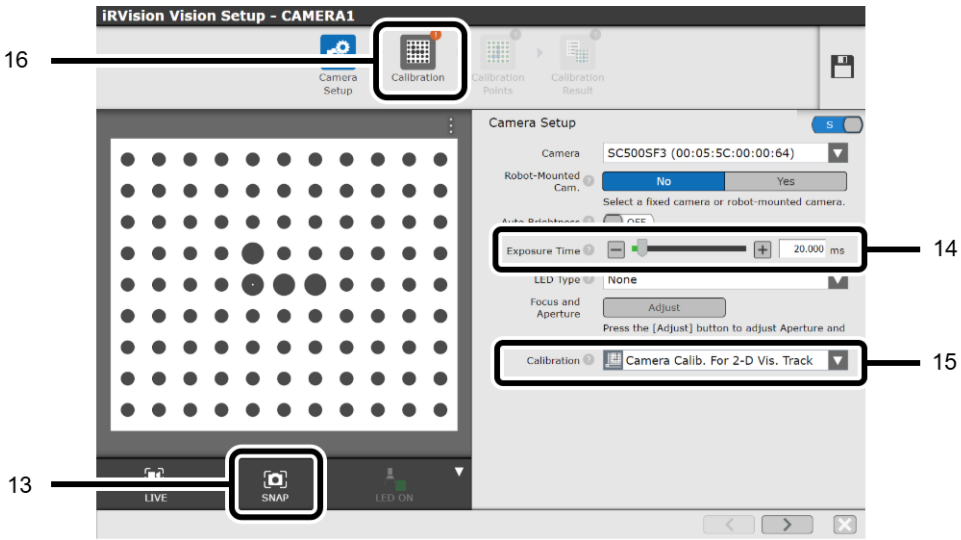
The camera setup screen will appear.

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

14 Enter an [Exposure Time].

15 Select [Calibration for 2D VT] from the [Calibration] drop-down list.

16 Click [Calibration] in Program Line.



The calibration setup screen will appear.

17 Select [Conveyor Name].

18 Enter [Grid Spacing] of the grid pattern.

19 Select [Perspective] in [Projection].

[Perspective] has been selected as the initial value.

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

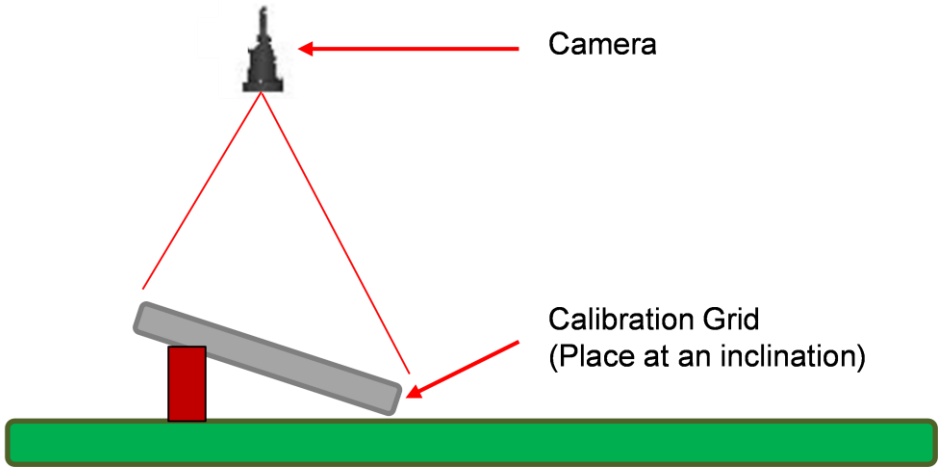
20 In [Camera Distance], enter the focal distance of the lens that is being used.

⚠CAUTION

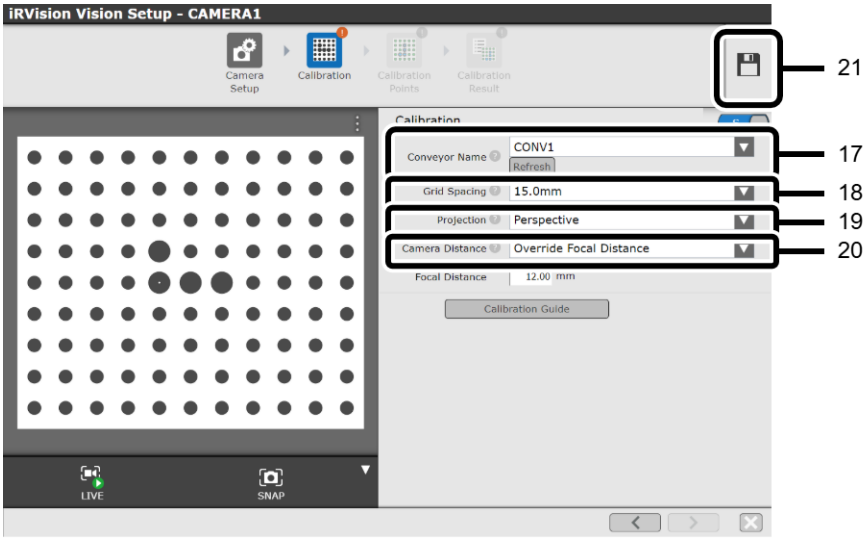
With regard to the calculation method for [Camera Distance], when [Auto] is selected, it is in principle possible that the focal distance might not be calculated accurately because the calibration grid on the conveyor and the camera's light-sensitive surface are nearly parallel.

NOTE

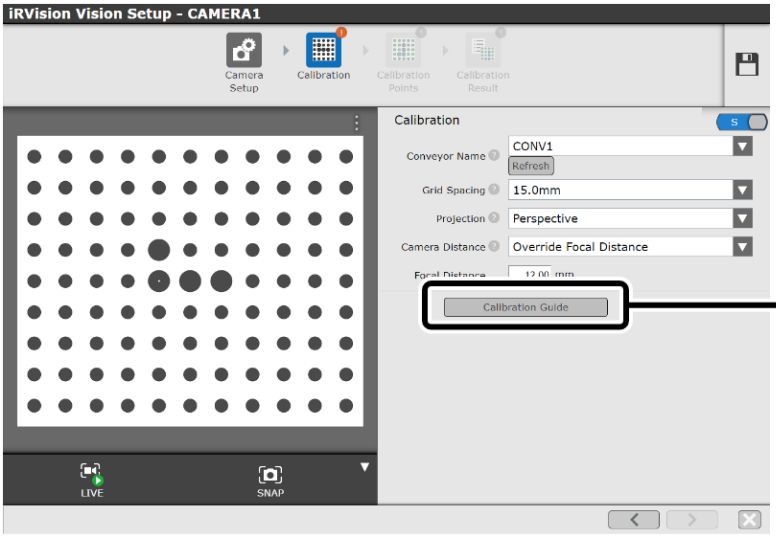
By placing the calibration grid at an inclination, an accurate [Camera Distance] can be calculated using [Auto].



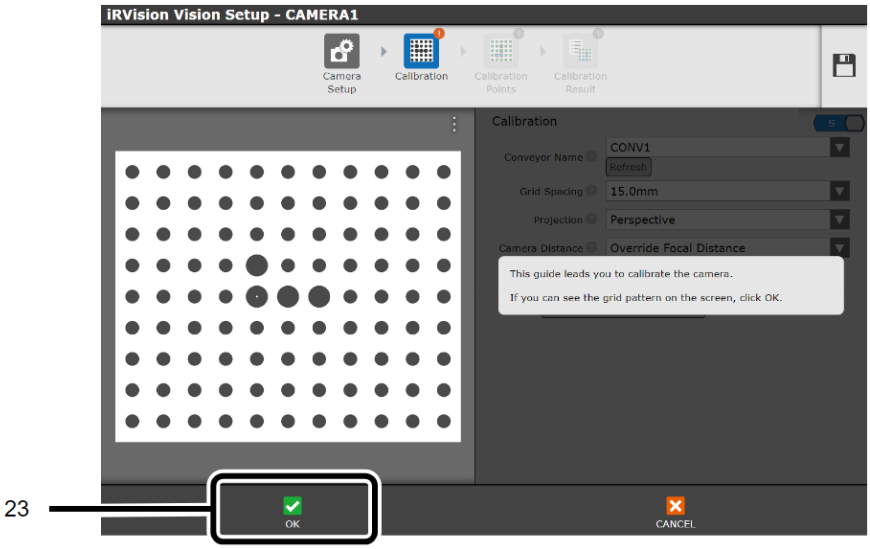
21 Click [>] and click [SAVE].



22 Click [Calibration Guide].

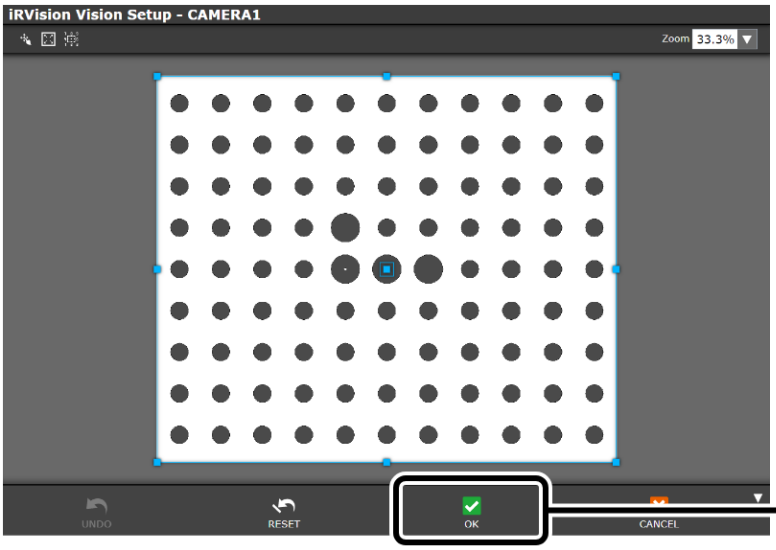


23 Confirm that the calibration grid is displayed on the screen and click [OK].

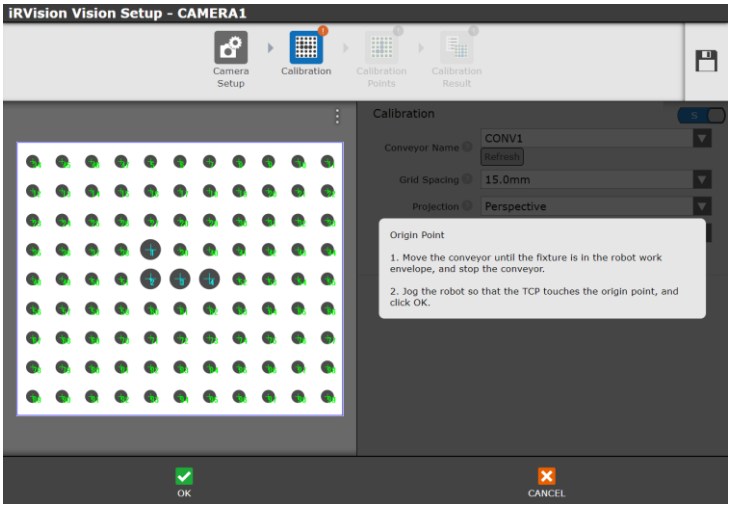


24 Enclose the dot pattern that appears on the screen with a magenta rectangular window.

25 Click [OK].



26 Move the conveyor, and stop when the calibration grid gets near the center of the robot's operating range.



27 Touch up the origin of the calibration grid.

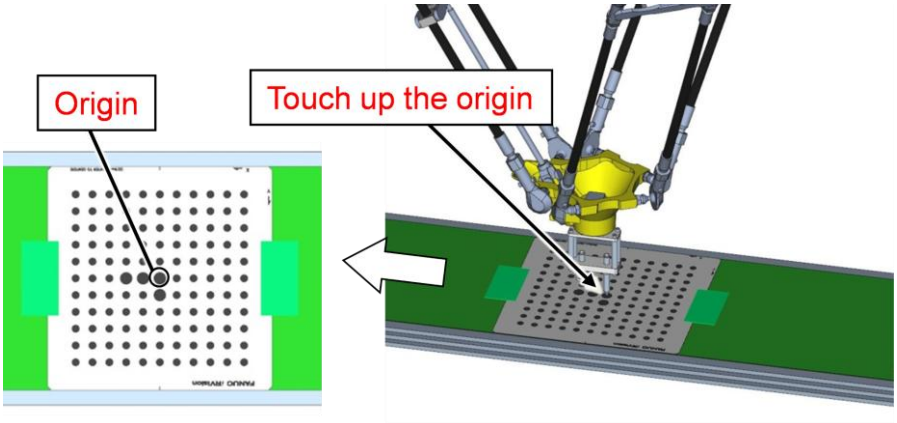
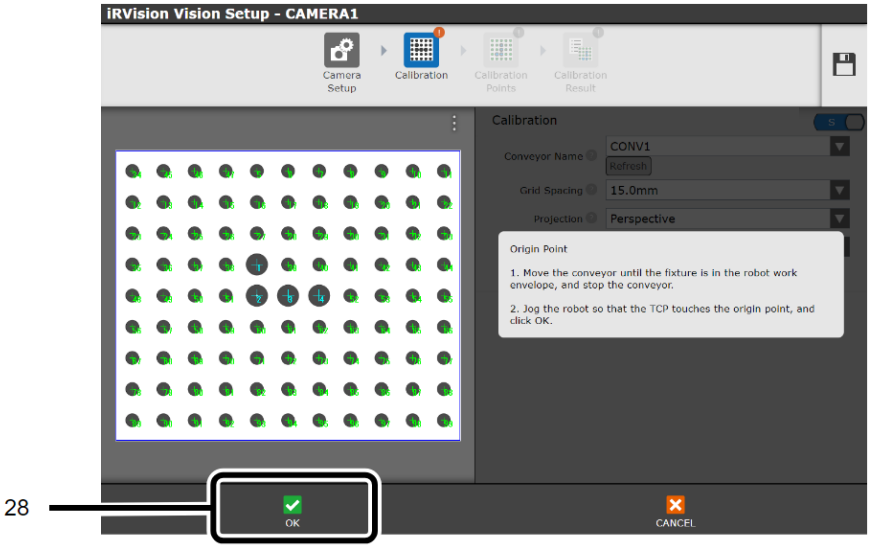


Fig. 3.3.8.1-4 Touching up the origin of the calibration grid

28 Click [OK].



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

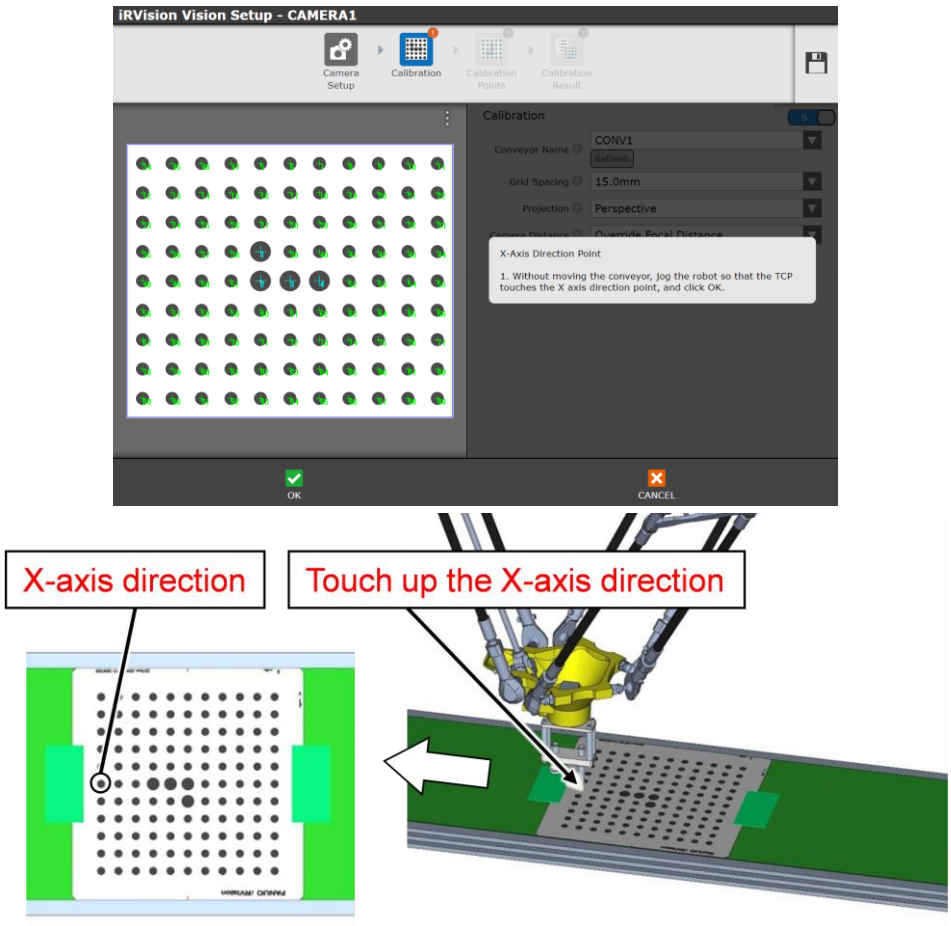
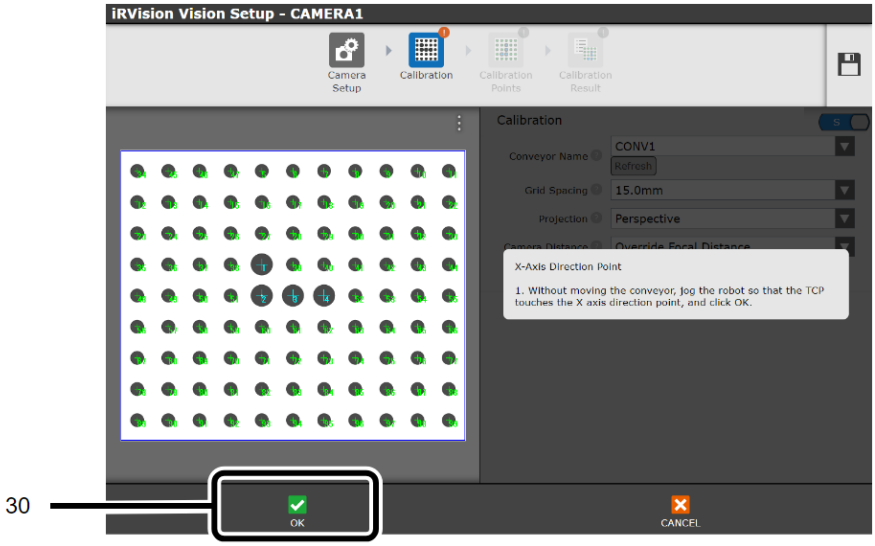


Fig. 3.3.8.1-5 Touching up the X-axis direction of the calibration grid

30 **Click [OK].**



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

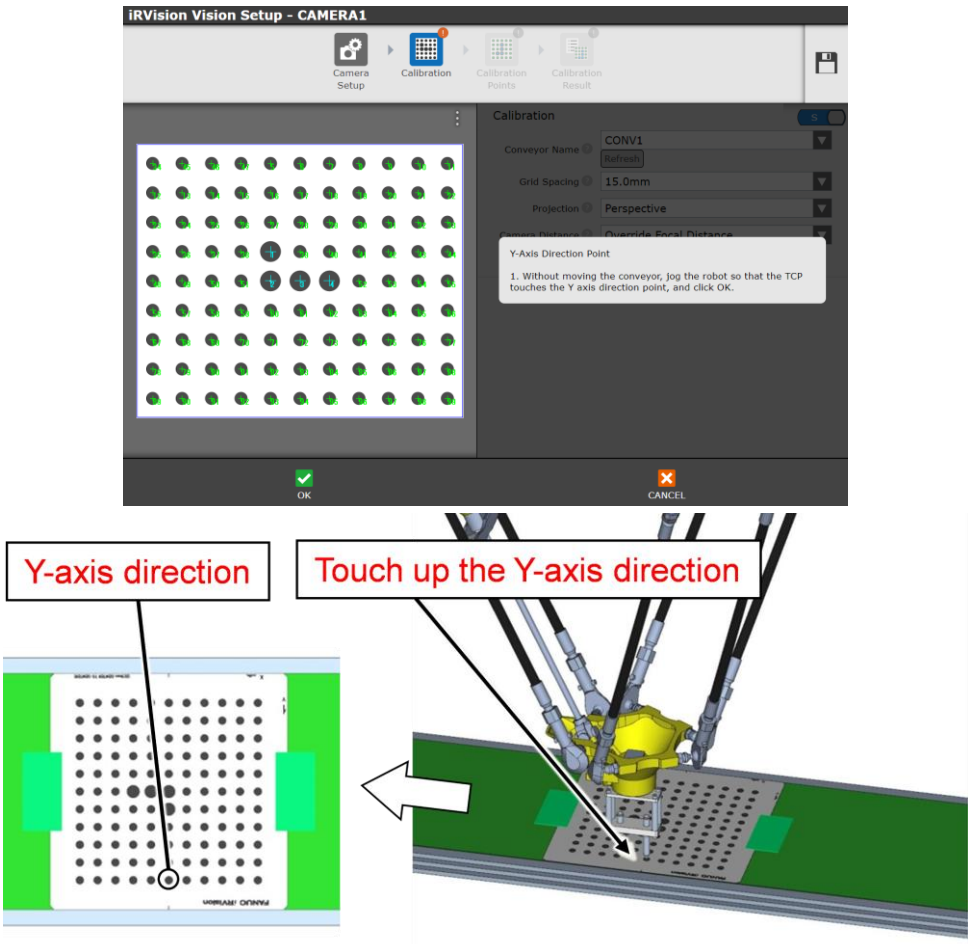
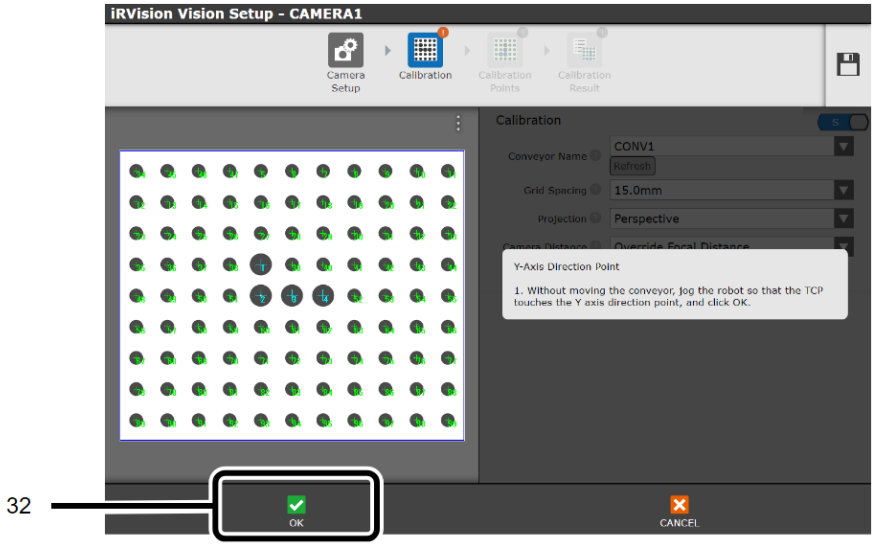


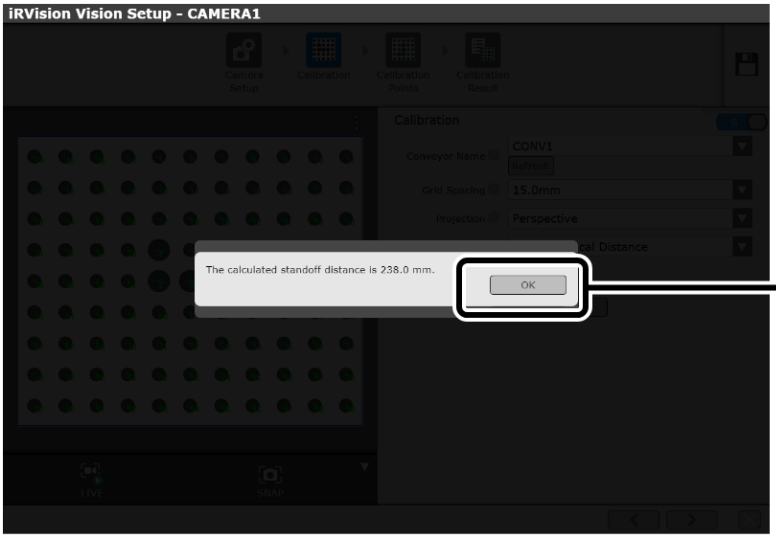
Fig. 3.3.8.1-6 Touching up the Y-axis direction of the calibration grid

32 Click [OK].

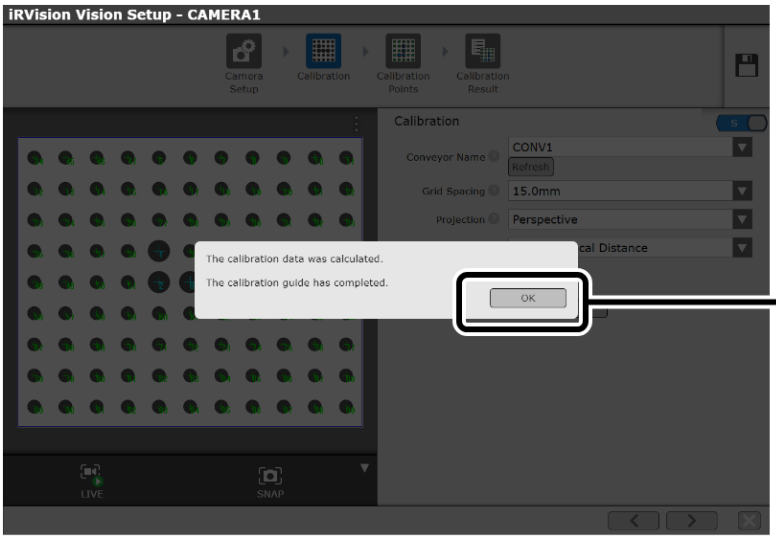


33 The calculated standoff distance is displayed. Confirm that the value is suitable for the actual standoff distance, and click [OK].

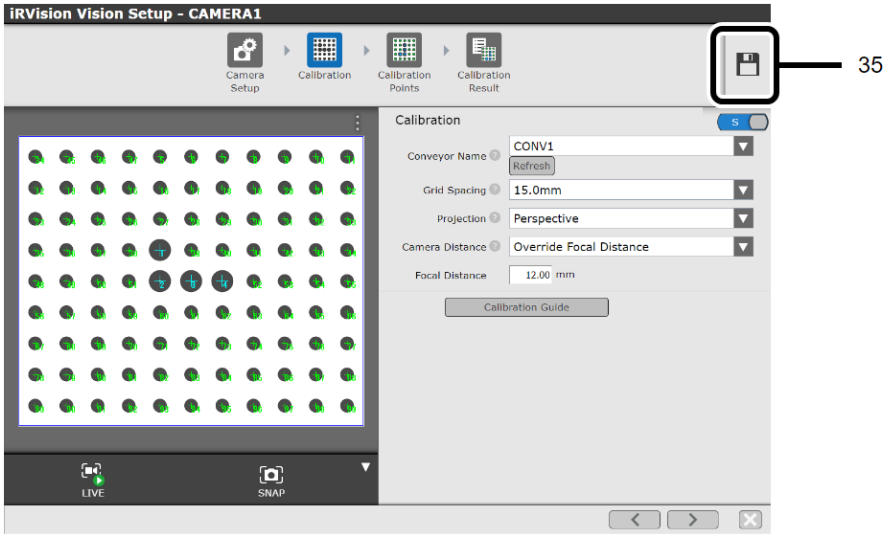
If the value is unsuitable, check whether the setting procedure and set parameters are correct.



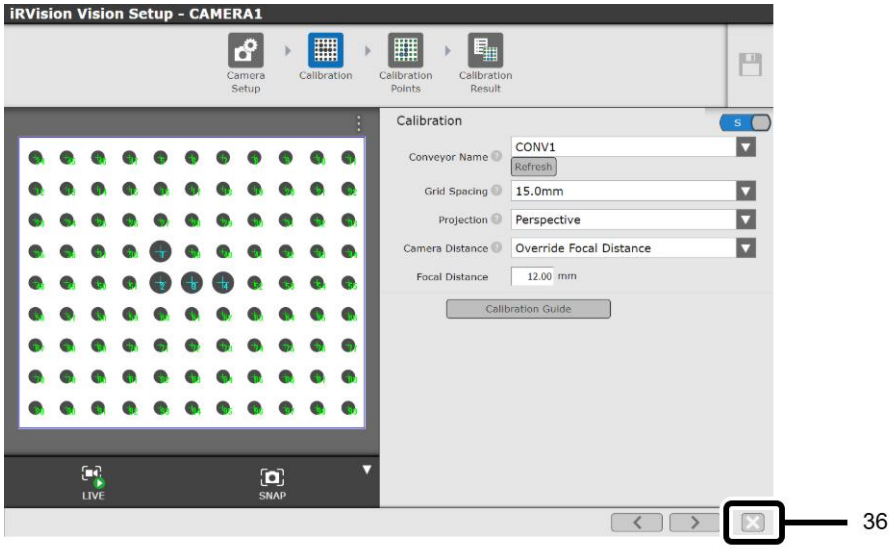
34 Click [OK].



35 Click [SAVE].



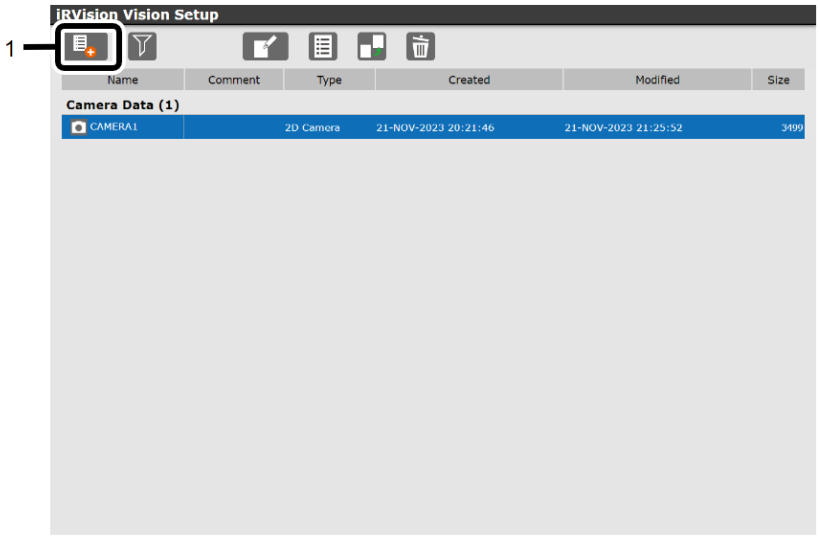
36 Click [×].



3.3.8.2. Vision process

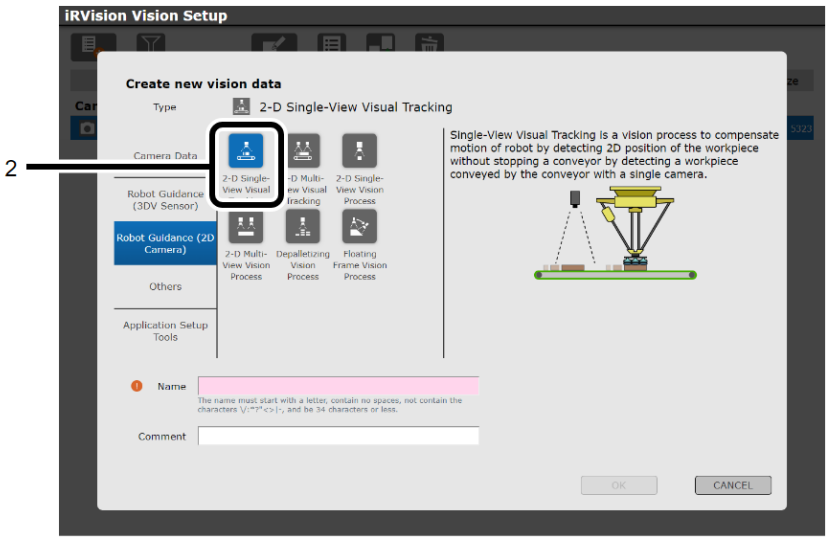
Set up a vision process.

1 Click [CREATE].



The [Create new vision data] screen appears.

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



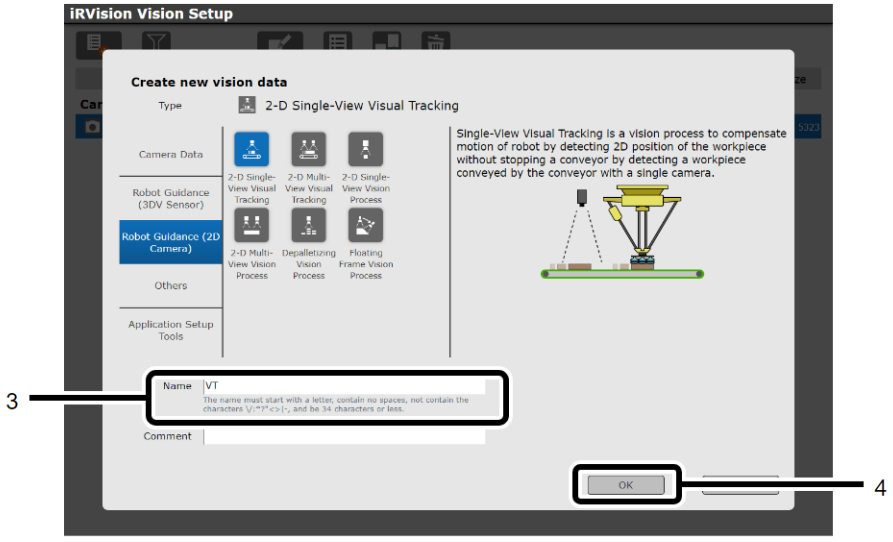
After a while, the following screen appears.

3 Enter the vision process name in [Name].

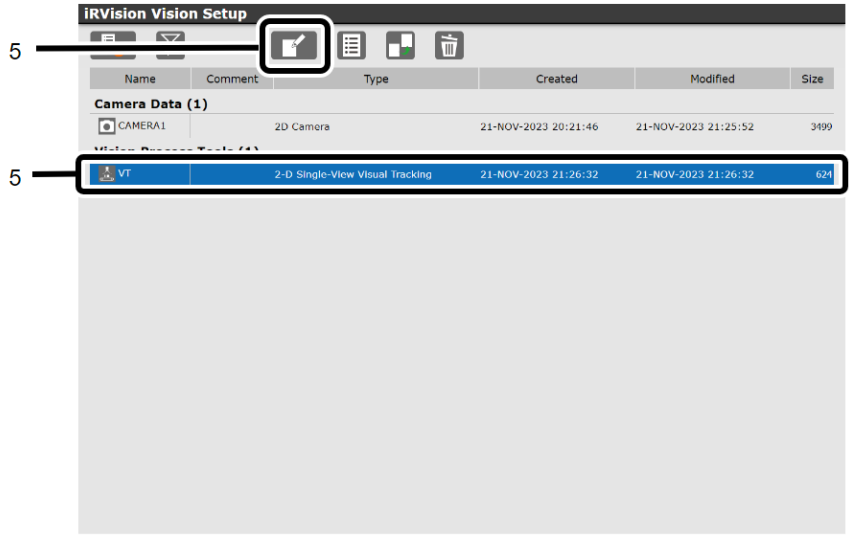
Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

Example: VT

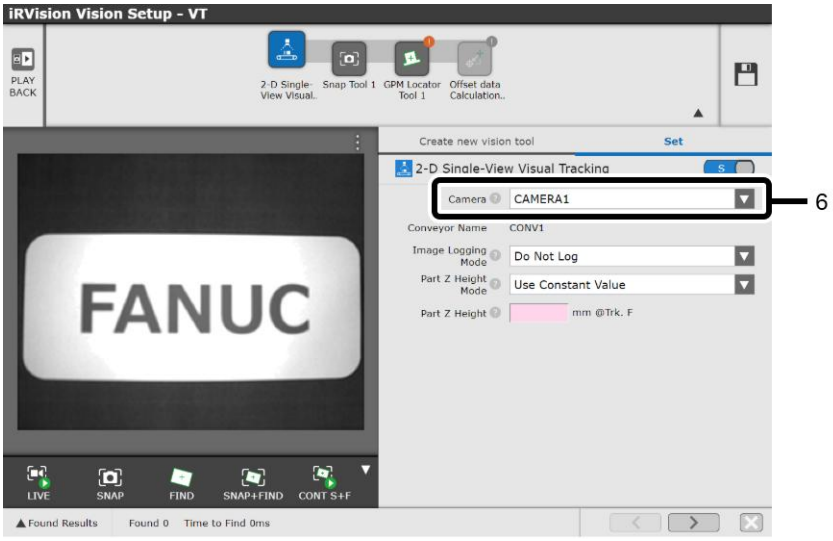
4 Click [OK].



5 On the vision data list screen, select a created vision process, and click [Edit].



6 Select the connected camera in the [Camera] drop-down list.

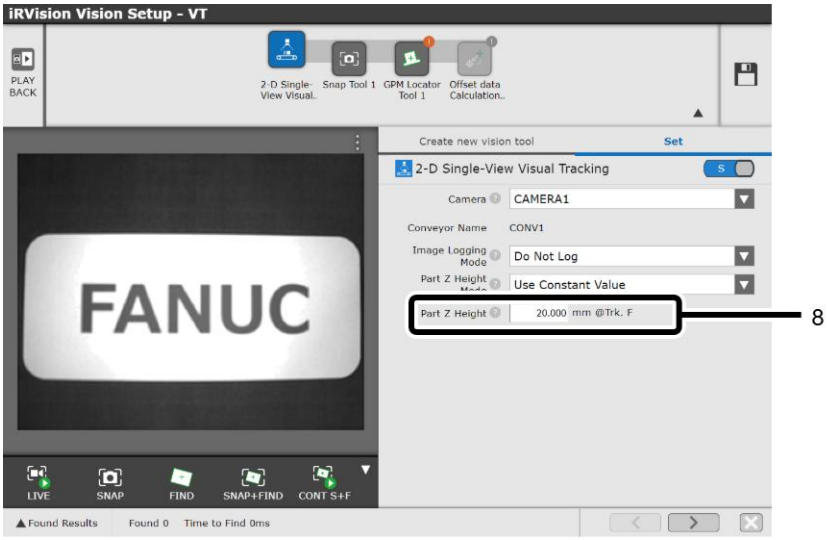


7 Measure the thickness of a part.

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

8 Enter the thickness of the part measured in step 7 into [Part Z Height].

Enter the distance from the plane at which the tracking frame was taught (the calibration grid surface) to the plane where the features that are to be found by vision will be.



NOTE

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

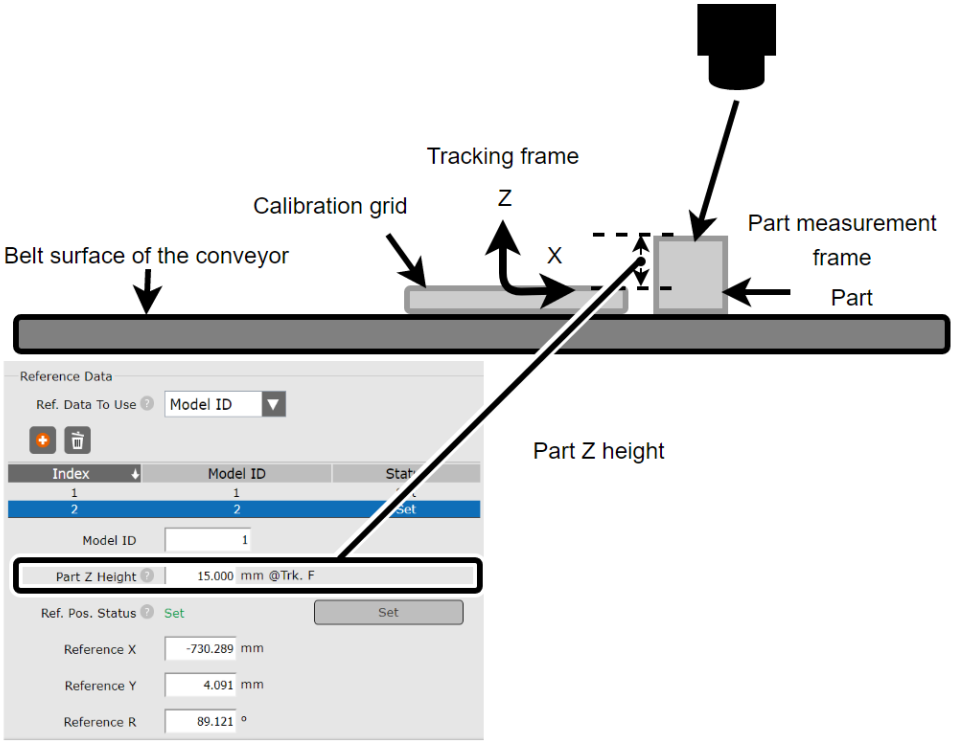
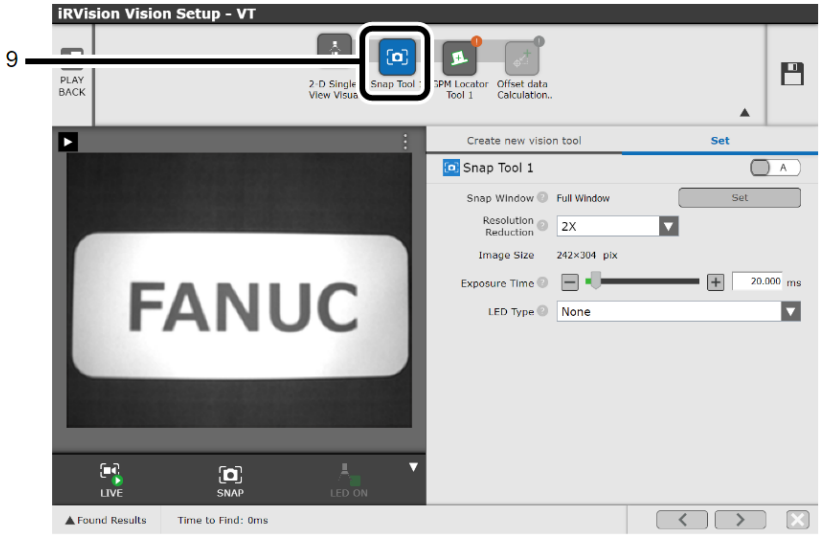


Fig. 3.3.8.2-1 Relationship between the system and 'Part Z Height'

9 Select [Snap Tool 1] in Program Line.



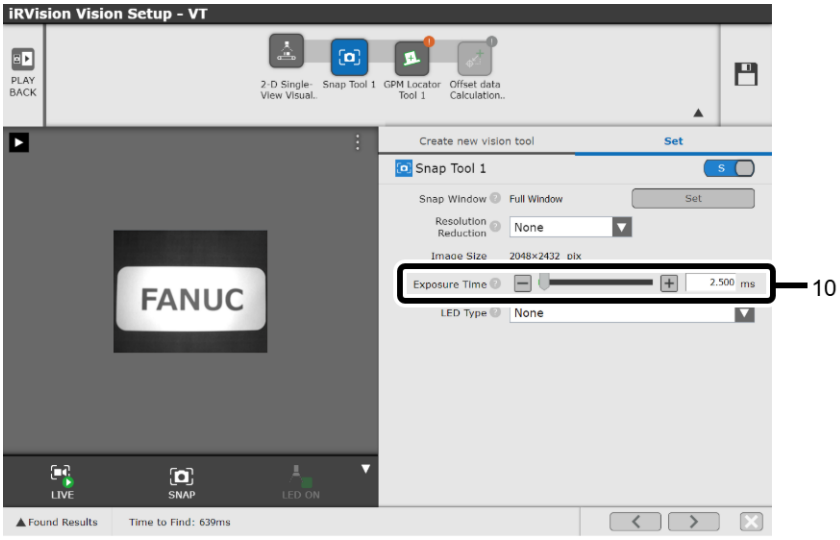
10 Enter an [Exposure Time].

Set the exposure time so that an image of a part that moves during exposure will have as little blurring as possible.

For, example, taking into account image blurring, in order to suppress misalignment during finding by vision to less than 0.1 mm when the conveyor speed is 100 mm/s, set a value that is less than 1 ms as the exposure time.

0.1 (mm) ÷ 100 (mm/s) = 1 (ms)

For information on the calculation of appropriate exposure times to suit the conveyor speed, refer to "Exposure time(P.22)".



11 Set a part in the field of view of the camera.

If the supply direction is fixed, set it by aligning it.

If the feed part variation is ±180 degrees, teaching may be easier if the orientation is aligned with the placing operation.

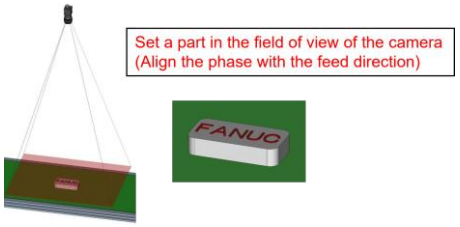


Fig. 3.3.8.2-2 Setting a part

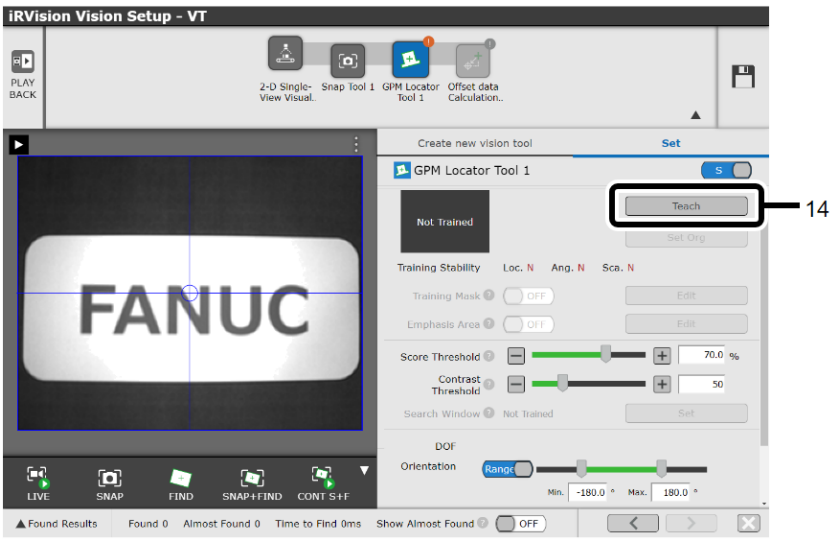
12 Select [GPM Locator Tool 1] in Program Line.



13 Click [SNAP].

Verify that the part appears on the screen.

14 Click [Teach].

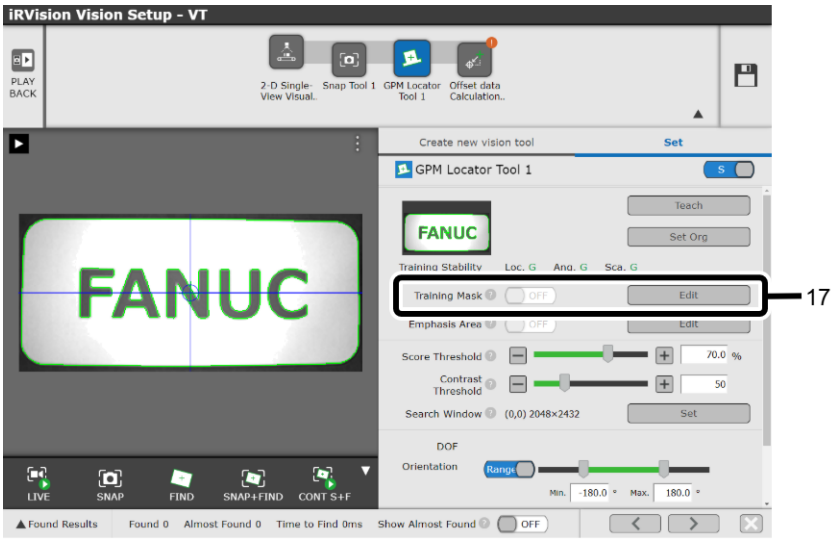


15 Enclose the part with a magenta rectangular window.

16 Click [OK].



17 Click [Edit] in [Training Mask].



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

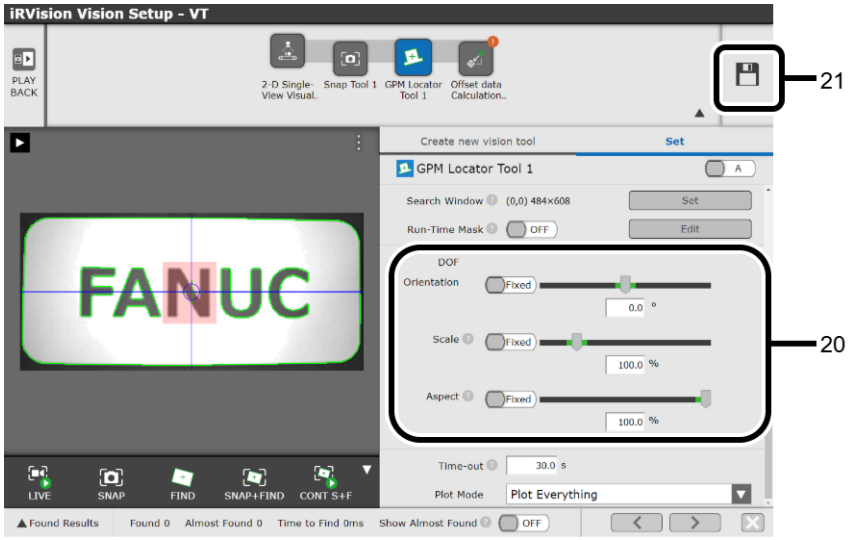
19 Click [OK].



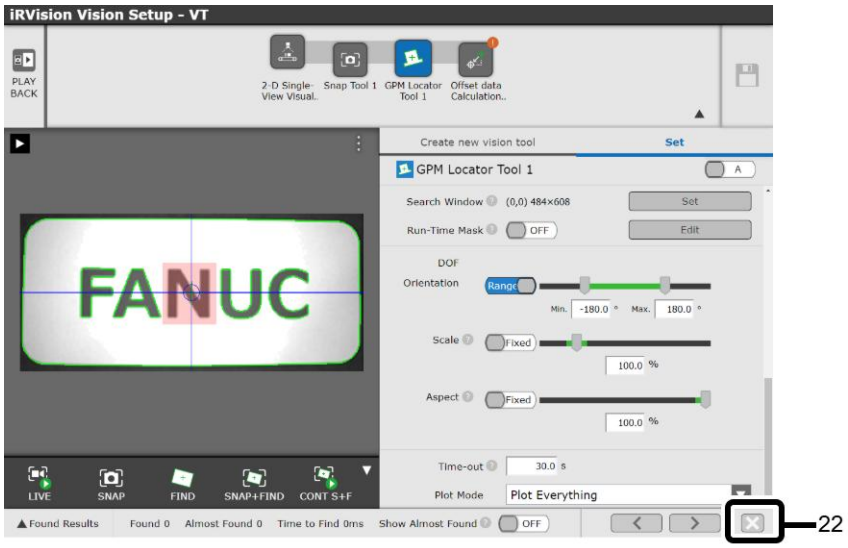
20 Set up the DOF as necessary.

For details of the setup, see the section “GPM Locator Tool” in the chapter “Command tools” in the "iRVision OPERATOR'S MANUAL (Reference)"(B-84434EN).

21 Click [SAVE].



22 **Click [×].**



3.3.9. Setting up a conveyor station

Set up a conveyor station.

1 **Press the [MENU] key on the robot controller's teach pendant.**
The menu appears.

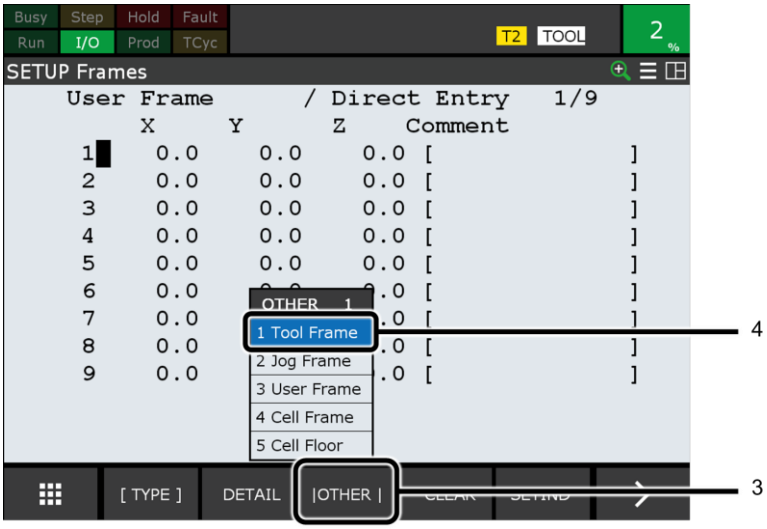
2 **From the menu, select [SETUP] → [Frames].**



3 **Press F3 [OTHER].**

4 **Select [Tool Frame] from the menu.**

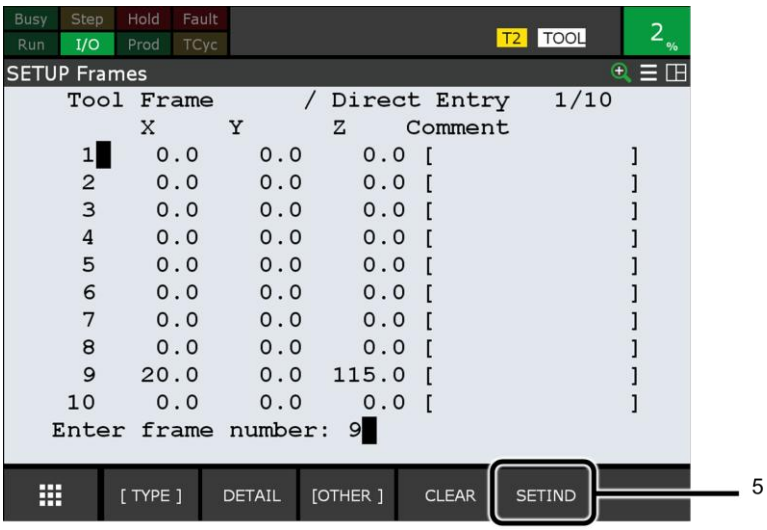
The Tool Frame list screen appears.



5 **Press F5 [SETIND].**

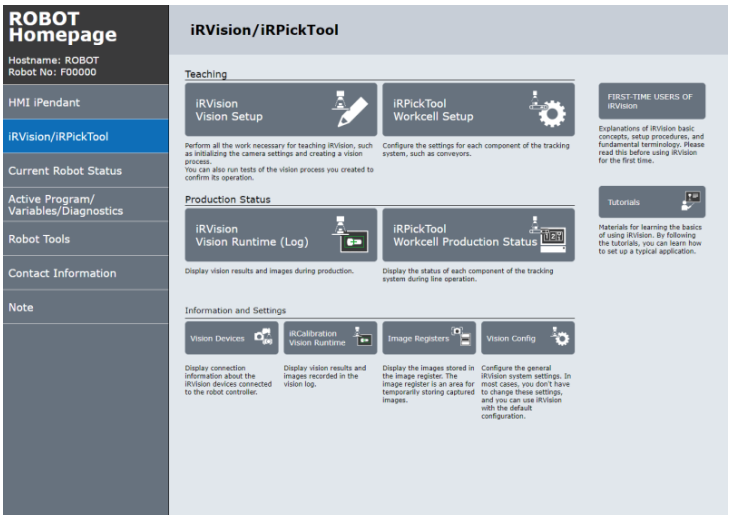
6 **Enter the frame number that has been used for the pointer tool.**

This procedure is explained using Tool 9 as an example.

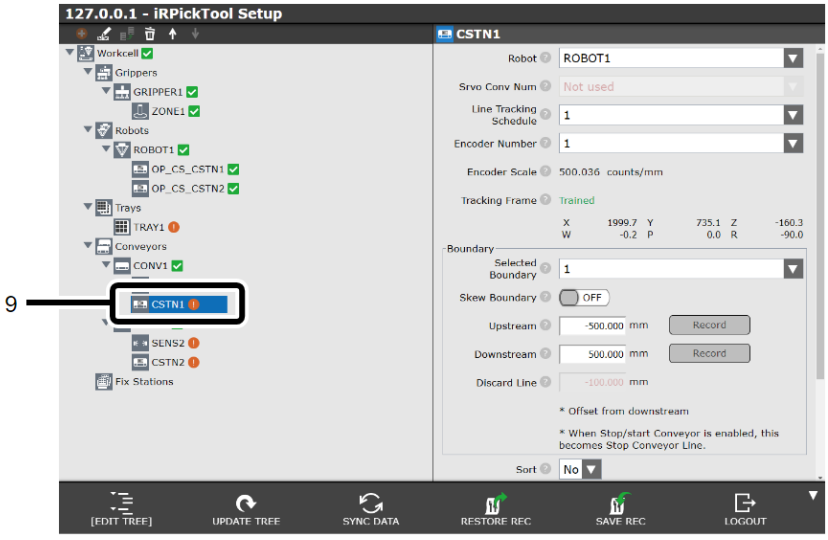


7 **Open the robot homepage.**

8 **Select [iRVision/iRPickTool] in the menu area and click [iRPickTool Workcell Setup] in the content display area.**



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



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

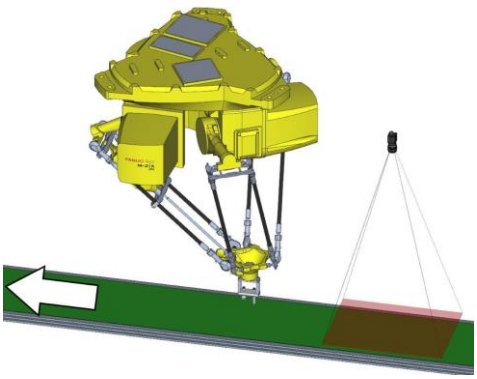
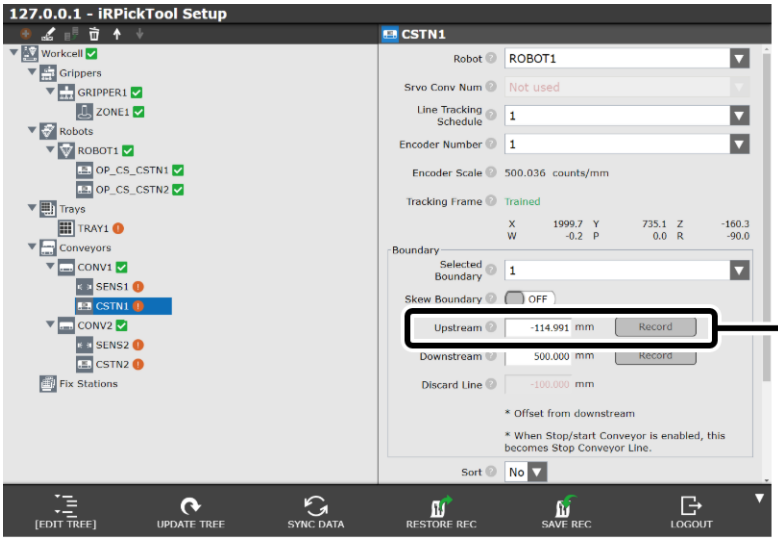


Fig. 3.3.9-1 Jogging the robot

11 **Click [Record] for [Upstream].**

The current position of the robot in the tracking frame will be saved in [Upstream].



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

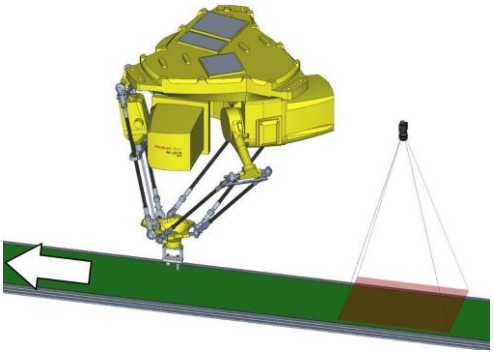
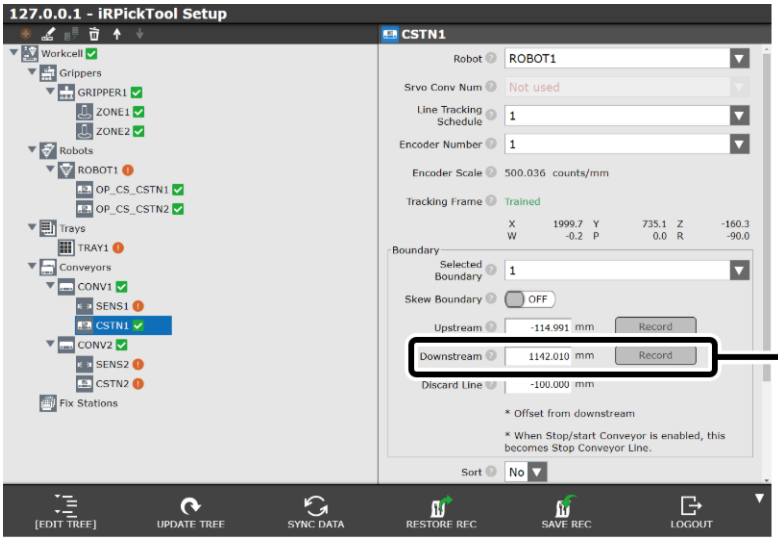


Fig. 3.3.9-2 Jogging the robot

13 Click [Record] for [Downstream].

The current position of the robot in the tracking frame will be saved in [Downstream].



14 Enter [Discard Line].

The discard line value must be larger than the value as calculated below.

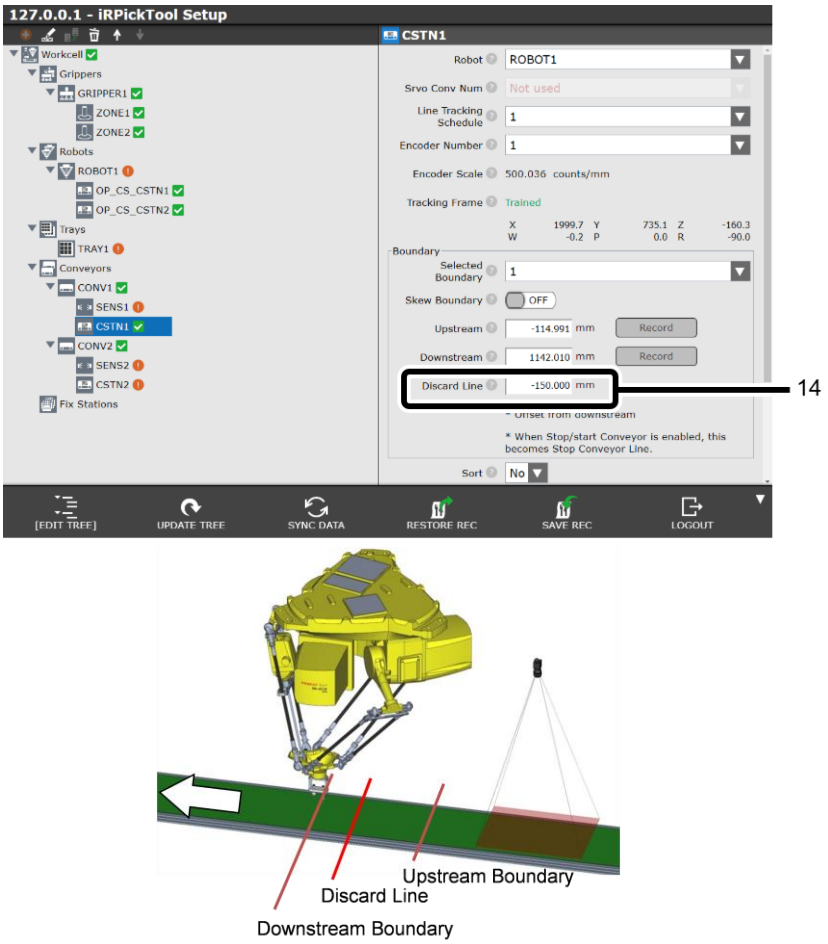
Time required for tracking* (sec) × Conveyor speed (mm/sec)

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

For example, if the robot operation time is approximately 0.8 seconds and the conveyor speed is 150 mm/sec, the free-running distance of the conveyor is:0.8 (sec) × 150 (mm/sec) = 120 (mm)

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

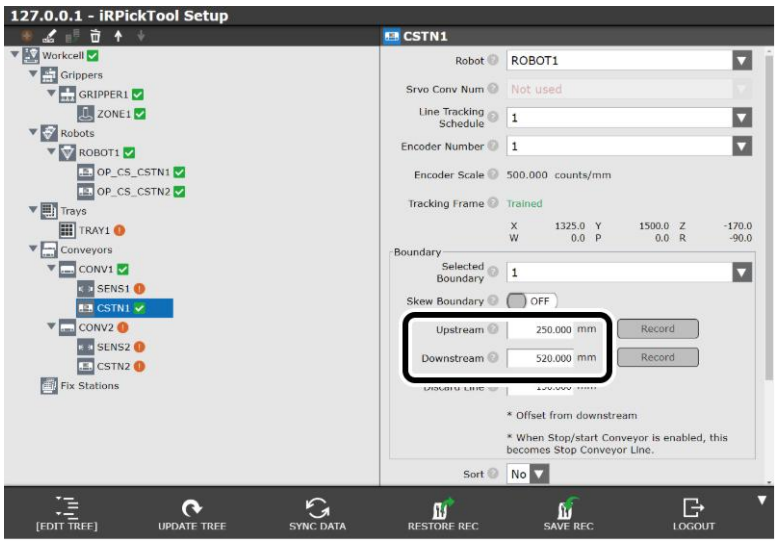
For details on discard lines, see "I.5.6. Setting up a conveyor station(P.318)".



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

Fig. 3.3.9-3 Setting up a discard line

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

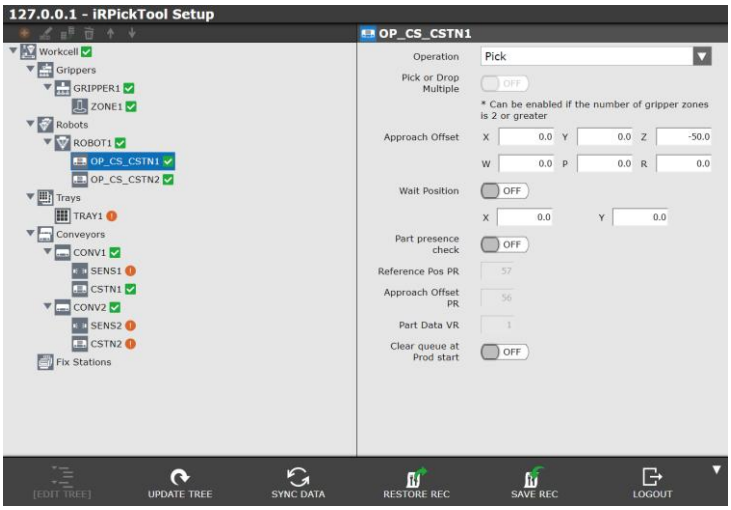


15 Next, set up the conveyor station of the outfeed conveyor. Select the conveyor [CSTN2].

16 Repeat steps 10 to 14 to set [Upstream], [Downstream], and [Discard Line] for the conveyor station of the outfeed conveyor.

3.3.10. Setting up a robot operation

Set up operations.



For a conveyor station operation, set the following items.

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

Item	Setting details
Vision Registers	This cannot be changed.
Clear queue at Prod start	Clears the remaining queues in the conveyor station when production starts.

3.3.11. Setting up a tool frame for the hand

Set up a tool frame for the hand.

In this section, a four-axis delta robot is used as an example.

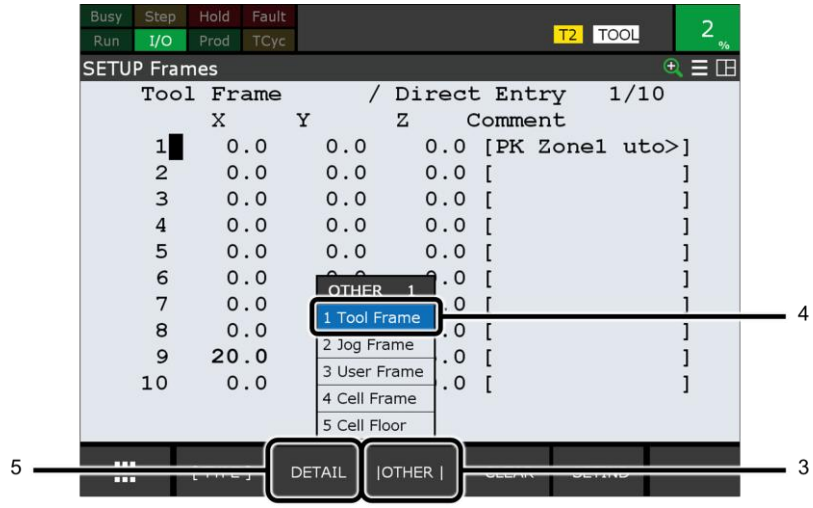
With a four-axis delta robot, we recommend that X and Y of the tool frame be set to 0, even if (X, Y) of the tool center point (TCP) of the hand in the mechanical interface frame is not (0, 0). This is to prevent the cycle time from being delayed or the robot's failing to stay in its operating range, as a result of large movement by the major axes (J1 to J3 axes) when a part is picked from the conveyor.

To make the tool's Z-axis face upward, directly input the tool's orientation. Directly input the values of the rotation angles W, P, and R of the tool frame around the mechanical interface frame's X, Y, and Z axes.

1. **Press the [MENU] key on the robot controller's teach pendant.**
   The menu appears.
2. **From the menu, select [SETUP] → [Frames].**

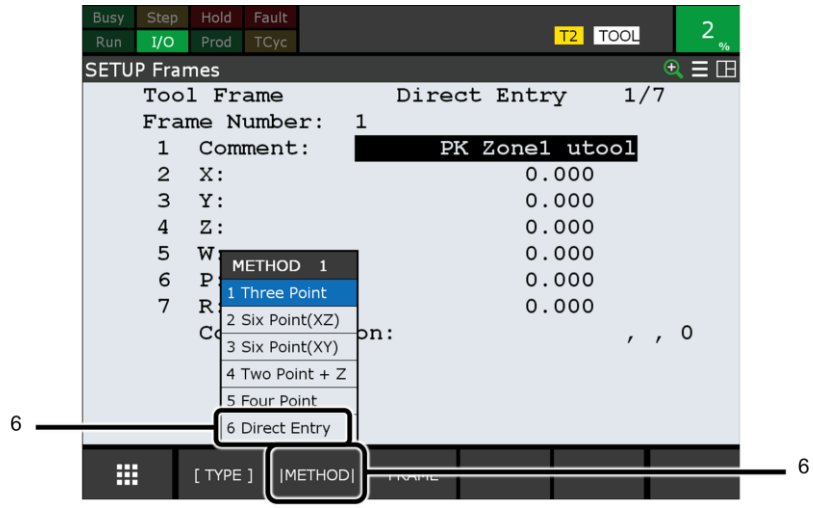


3. **Press F3 [OTHER].**
4. **Select [Tool Frame] from the menu.**
   The Tool Frame list screen appears.
5. **Place the cursor over Tool 1 and press F2 [DETAIL].**
   The tool frame setup screen for tool frame number 1 will appear.



6 Press F2 [METHOD].

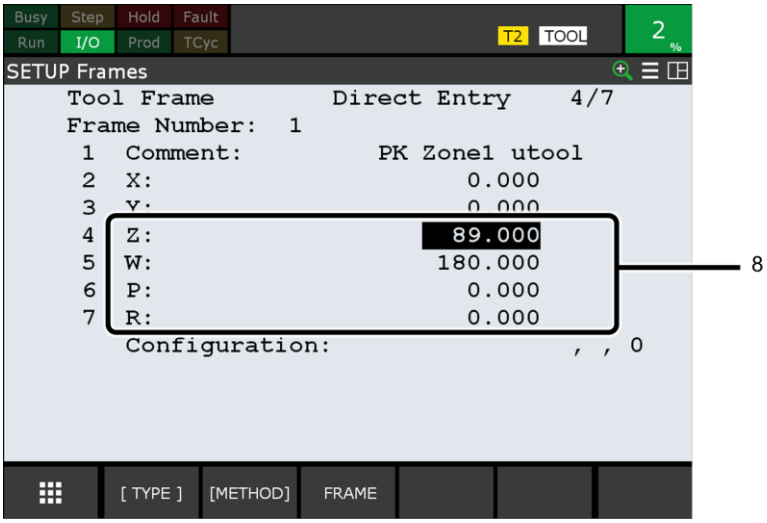
7 Select [Direct Entry] from the menu.



8 Move the cursor to [Z], [W], [P] and [R] and enter a value for each.

For [Z], input a numerical value measured with a ruler, etc.

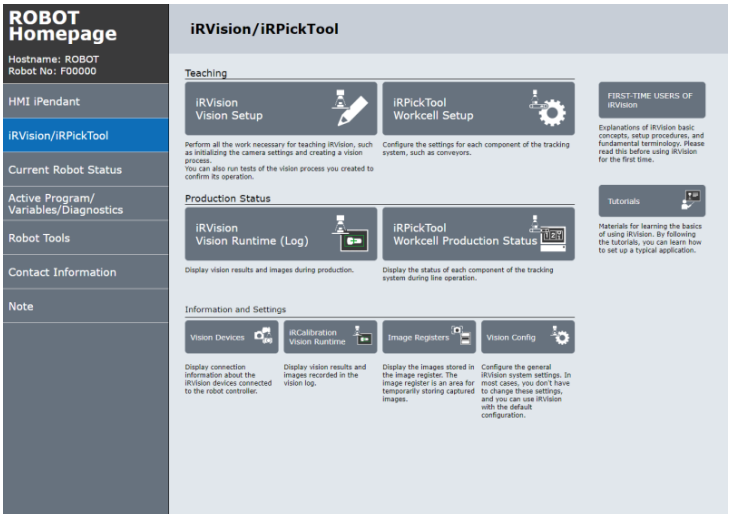
Input [W], [P] and [R] directly.



9 Make a note of the [Z], [W], [P] and [R] entered in step 8.

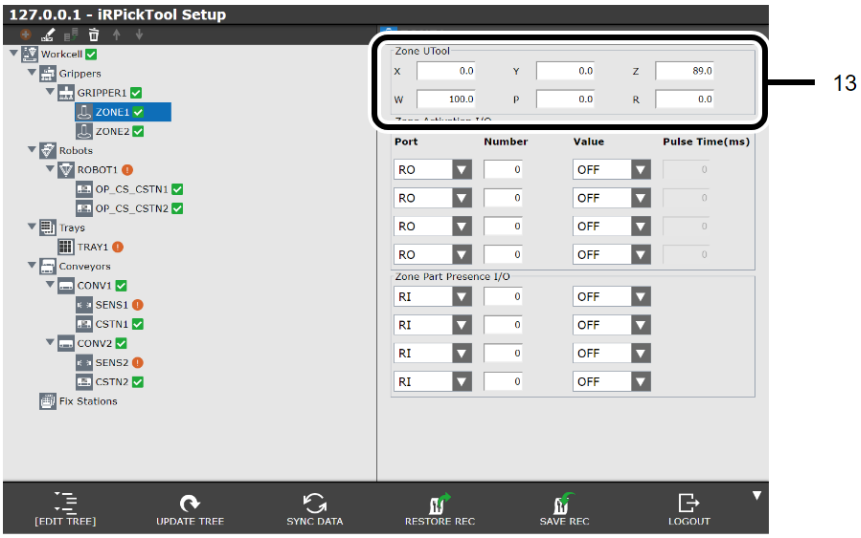
10 Open the robot homepage.

11 Select [iRVision/iRPickTool] in the menu area and click [iRPickTool Workcell Setup] in the content display area.



12 Select [Zone1] for [Gripper1].

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

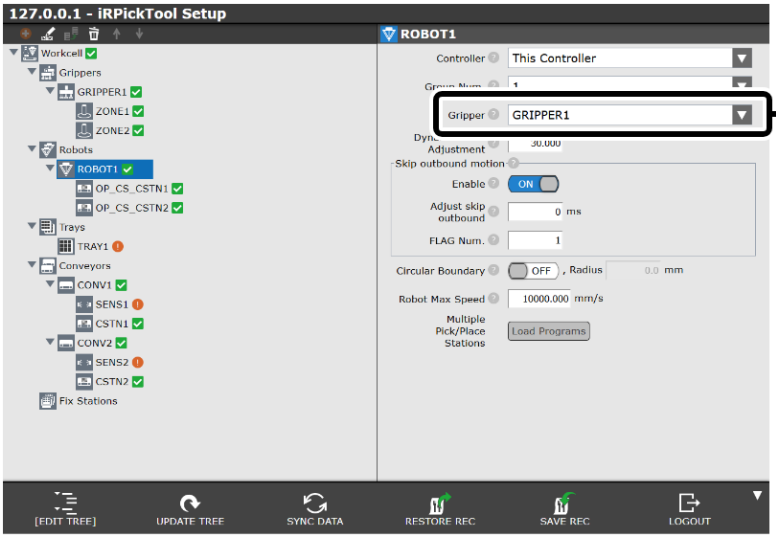


14 When using a multiple-item picking hand, set tool frame 2 and [Zone2] by following the same procedure as steps 1 to 13.

15 In the same manner, repeat the procedure for tool frame 3 and [Zone3], tool frame 4 and [Zone4], and so on up to the number of zones to be used.

16 Select [Robot1] for [Robots].

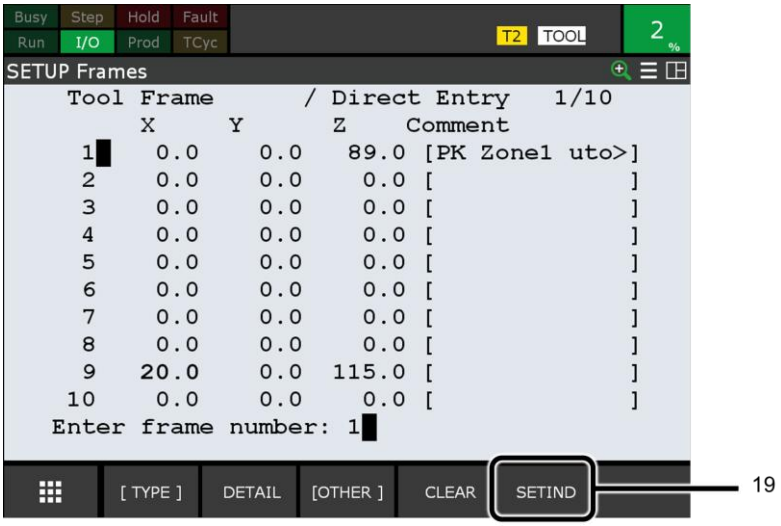
17 Select [Gripper1] from the [Gripper] drop-down list.



18 In the same way as steps 1 to 4, display the list screen for the tool frames.

The setting values of all the tool frames can be checked.

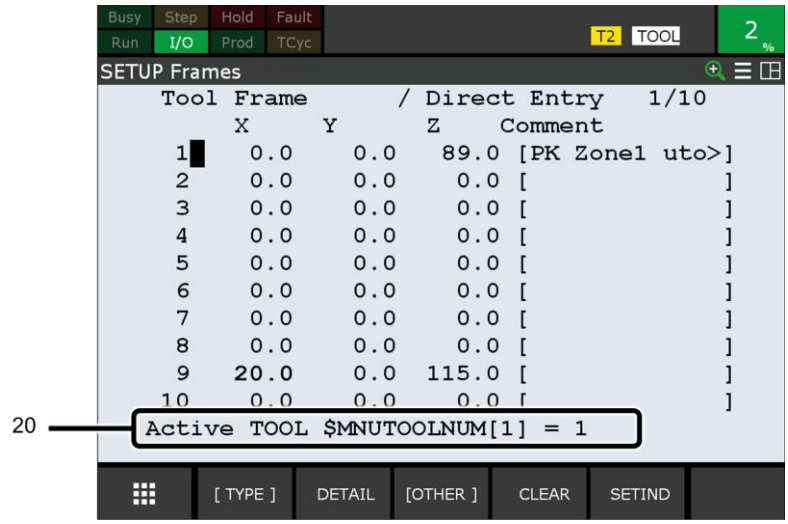
19 Press F5 [SETIND].



20 Enter tool frame number 1.

Tool frame 1 will be enabled.

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



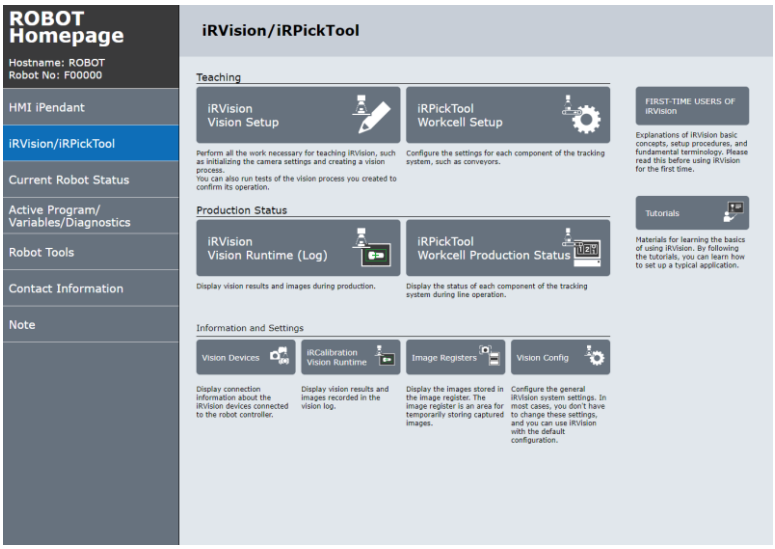
3.3.12. Setting up a reference position and teaching a robot position

This procedure is for setting the reference position and teaching positions to the robot.

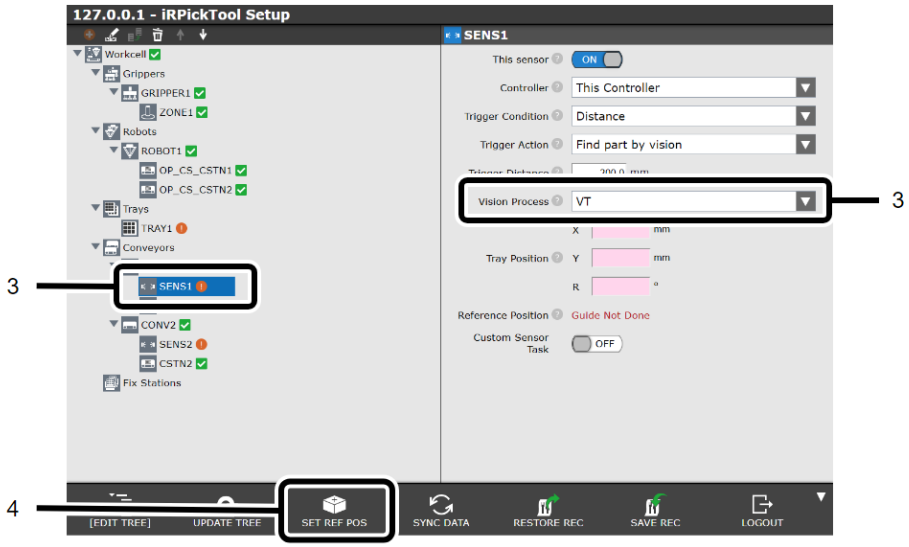
3.3.12.1. Setting up a reference position for the infeed conveyor, and teaching a robot position

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

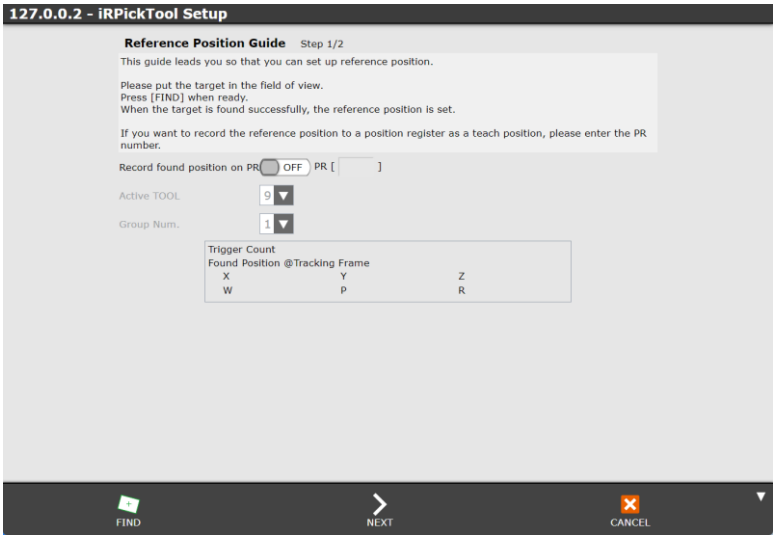
1. **Open the robot homepage.**
2. **Select [iRVision/iRPickTool] in the menu area and click [iRPickTool Workcell Setup] in the content display area.**



3. **Select [SENS1] in [CONV1] and select the name of the vision process (VT) that you created in "I.3.3.8.2. Vision process(P.213)" from the [Vision Process] drop-down list.**
4. **Click [SET REF POS].**



The following [Reference Position Guide Step: 1/2] will appear.



CAUTION

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

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

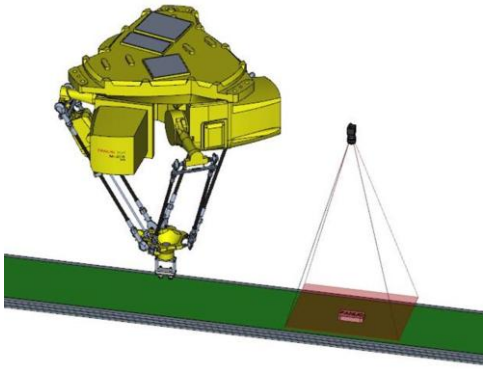


Fig. 3.3.12.1-1 Placing a part inside the field of view of the camera

6 Click [FIND].

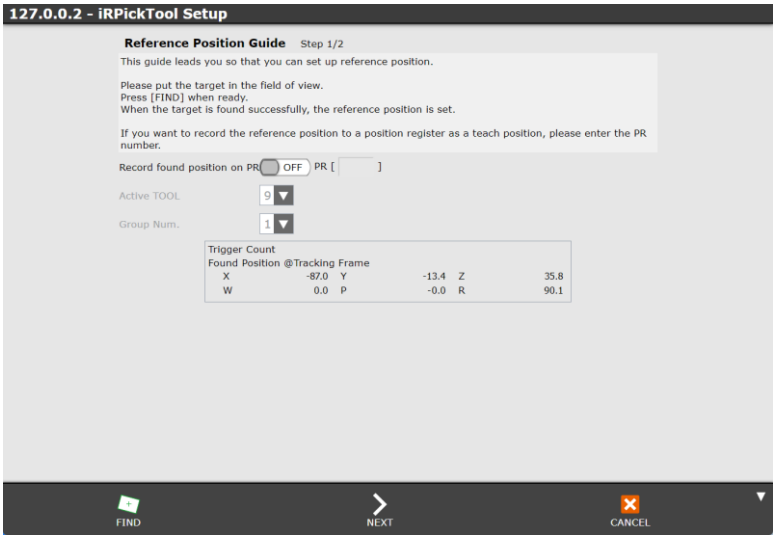
If detection succeeds, [Found Position] will be displayed.

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

NOTE

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

8 Click [NEXT].



9 Click [FINISH].

Check that [Guide Done] appears in [Reference position].



10 Move the conveyor, and then when the part enters the robot's moving range, stop it.

* Be careful that the part does not become misaligned.

11 Move the robot in jog mode to the part pick position.

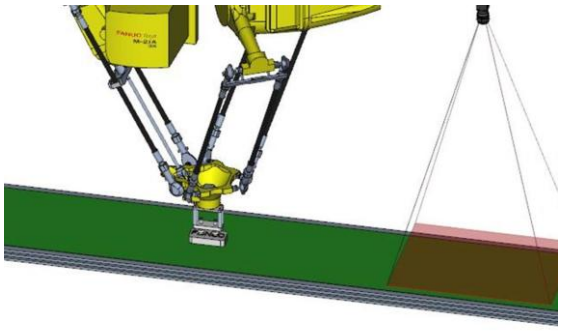


Fig. 3.3.12.1-2 Moving the robot to the part pick position

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

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

13 While holding down the [SHIFT] key, press F5 [TOUCHUP].

If F5 [TOUCHUP] is not displayed, press [> (NEXT)] to switch screens.



14 A message saying "VR is UNINIT, continue?" will be displayed, but press F4 [YES] to teach the position.

NOTE

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



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

1 Press the [MENU] key on the robot controller's teach pendant.
The menu appears.

2 Select [NEXT] from the menu.

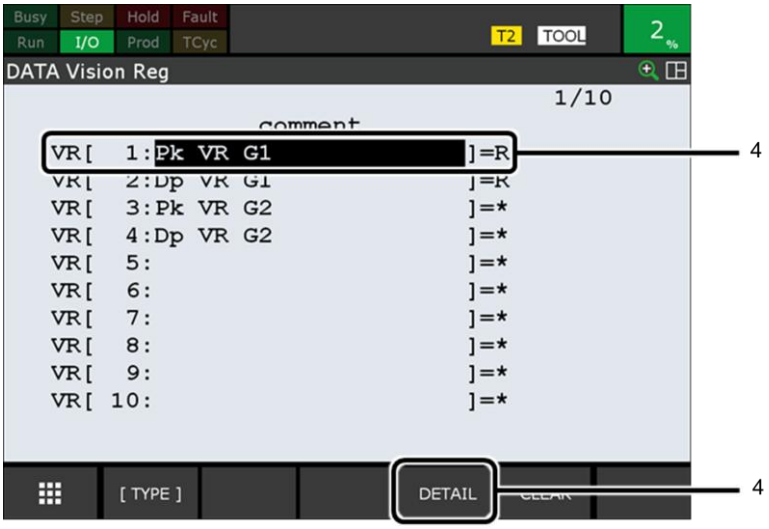


3 Select [DATA] → [Vision Reg] from the menu.

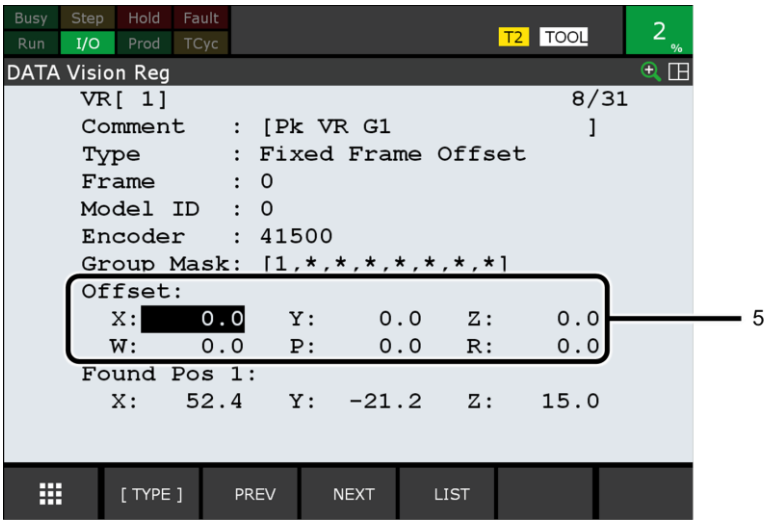
NOTE



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



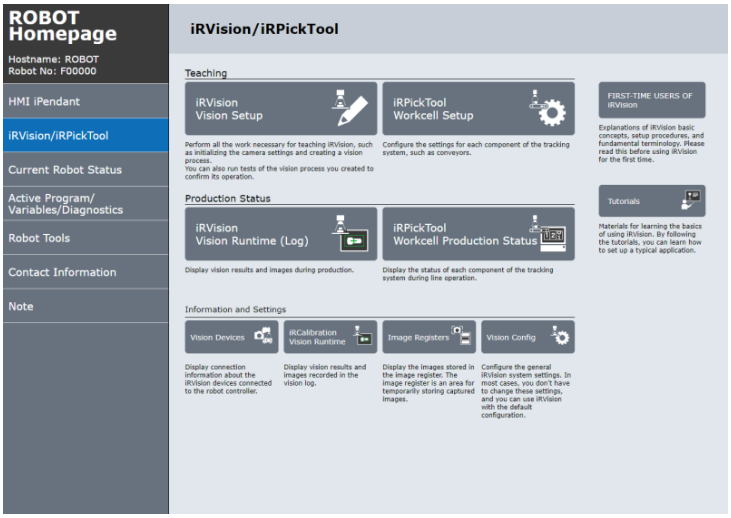
5 Enter '0' for all of the values [X], [Y], [Z], [W], [P] and [R] in [Offset].



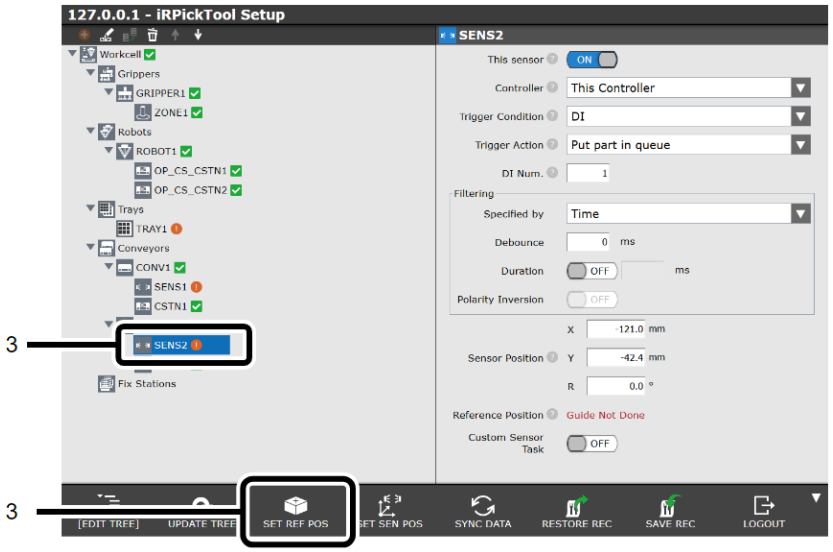
3.3.12.2. Setting up a reference position for the outfeed conveyor, and teaching a robot position

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

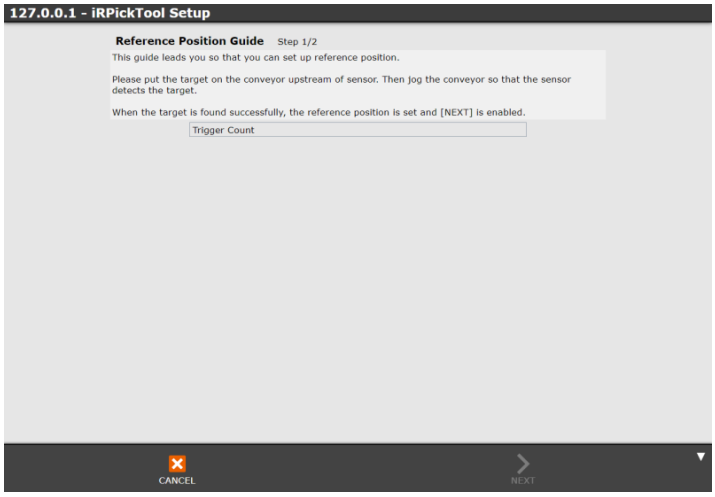
1. **Open the robot homepage.**
2. **Select [iRVision/iRPickTool] in the menu area and click [iRPickTool Workcell Setup] in the content display area.**



3. **Select [SENS2] for the sensor, and click [SET REF POS].**



After a while, the following screen appears.



4 Place a tray upstream of the trigger sensor.

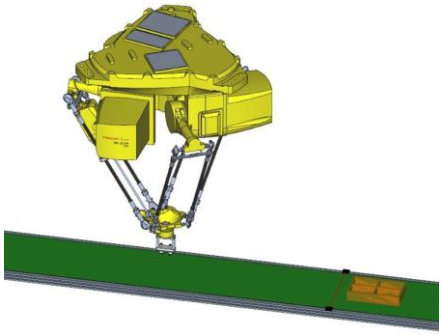


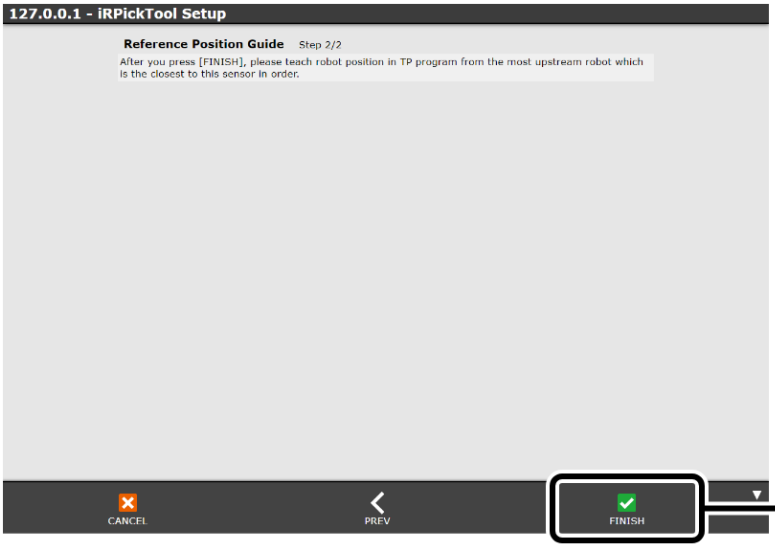
Fig. 3.3.12.2-1 Placing a tray upstream of the trigger sensor.

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

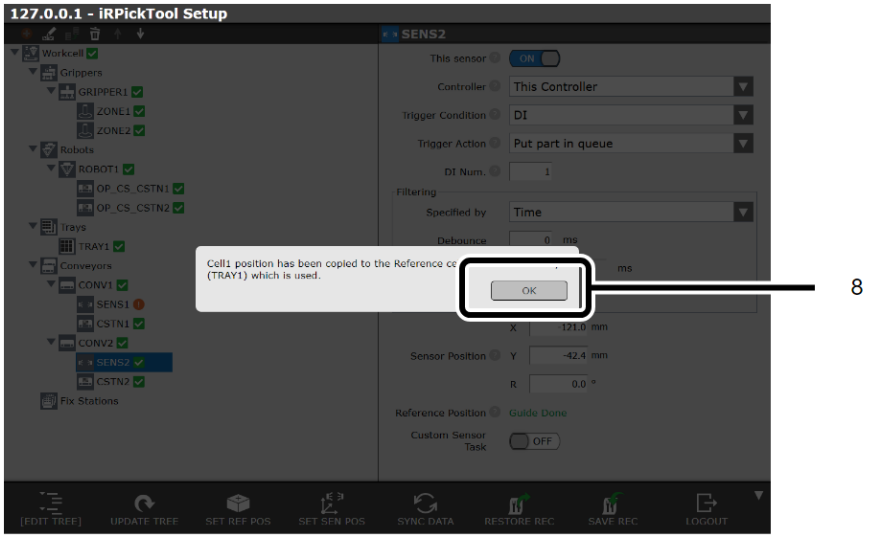
6 Click [NEXT].
If detection succeeds, [NEXT] will become active.



7 Click [FINISH].

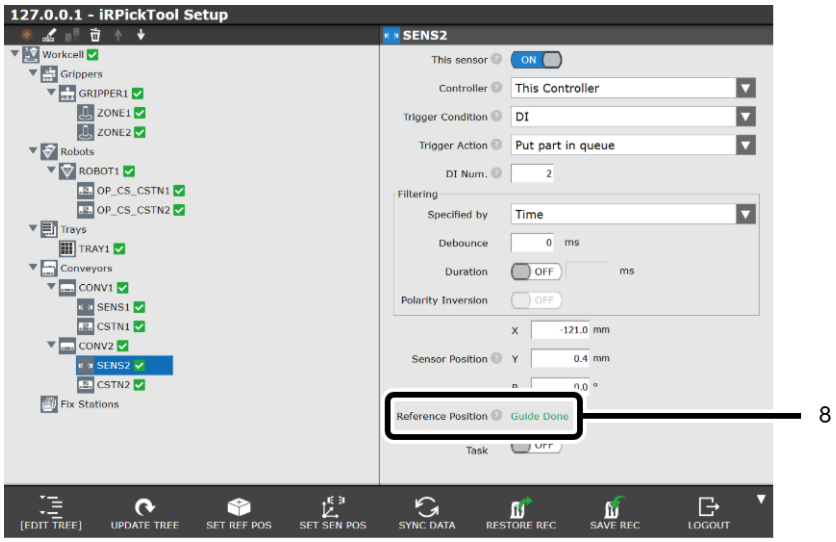


8 The following message will appear. Click [OK].



After a while, the following screen appears.

Check that [Guide Done] appears in [Reference position].



9 Move the conveyor. When the tray comes within the operating range of the robot, stop the conveyor.

* Be careful that the tray does not become misaligned.

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

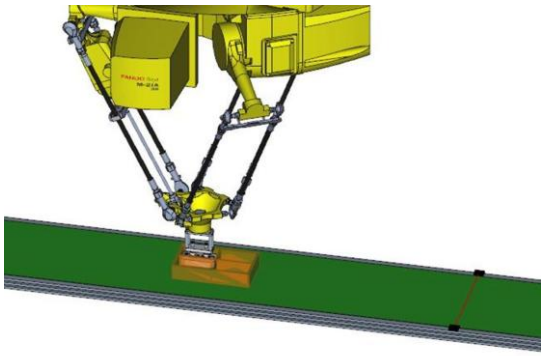


Fig. 3.3.12.2-2 Moving the robot to the drop position of cell 1 on the tray

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

```
69: L PR[59:Dp1 Cv Ref Pos] max_speed CNT0 VOFFSET,VR[2] TB    R[145]sec,
    CALL PKGROPEN("Gripper ID"=R[104:Gripper Id G],"Start Zone"=R[107:Counter G1],
    "End Zone"=R[106:Gr EndZone ])
```

12 While holding down the [SHIFT] key, press F5 [TOUCHUP].

If F5 [TOUCHUP] is not displayed, press [> (NEXT)] to switch screens.



13 A message saying "VR is UNINIT, continue?" will be displayed, but press F4 [YES] to teach the position.

NOTE

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



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

1 Press the [MENU] key on the robot controller's teach pendant.
The menu appears.

2 Select [NEXT] from the menu.

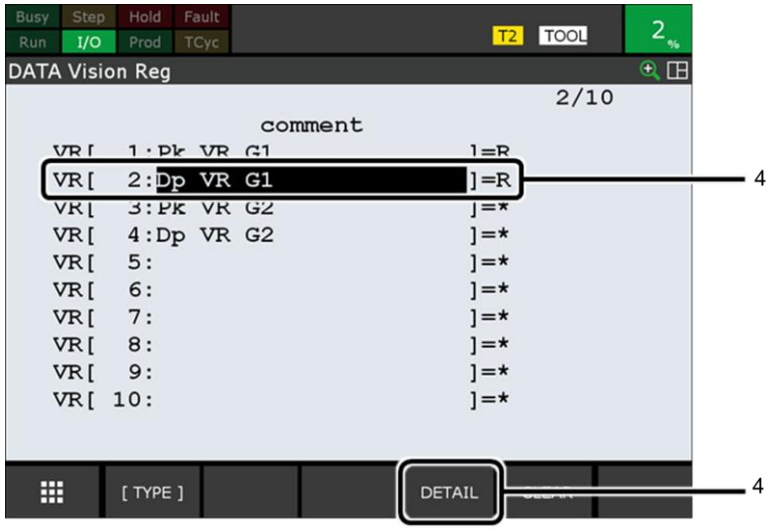
NOTE



3 Select [DATA] → [Vision Reg] from the menu.



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

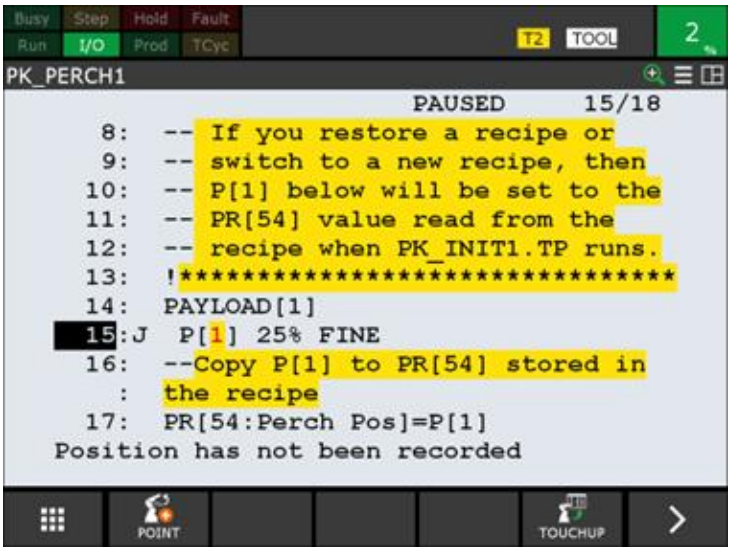


5 Enter ‘0’ for all of the values [X], [Y], [Z], [W], [P] and [R] in [Offset].

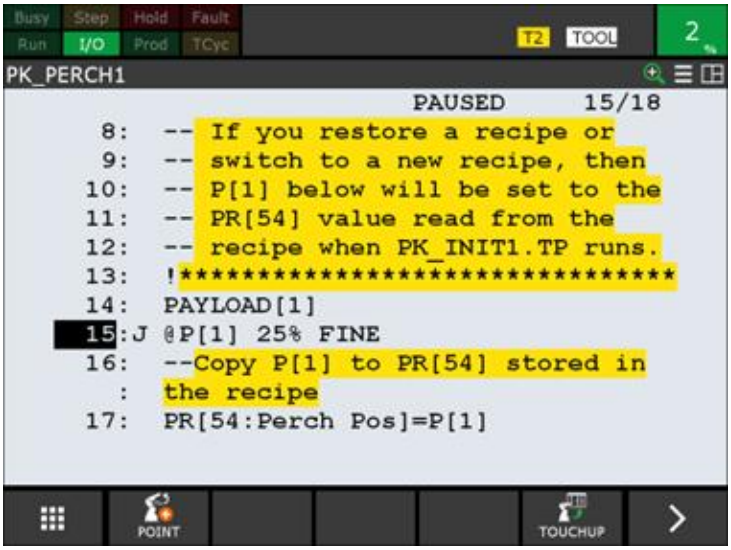
3.3.12.3. Teaching the robot home position

Teach the robot's home position.

1. **Display the main program PK_MAIN1.**
2. **Set the override to something less than 5%.**
3. **Run PK_MAIN1 from the beginning using teaching mode (T1, T2).**
   It will stop with an alarm, because P[1] in line 15 of PK_PERCH1 is untaught.



4. **Without changing the selected user frame or tool frame, jog the robot to the home position.**
5. **While holding down the [SHIFT] key, press F5 [TOUCHUP].**
   The current position will be taught to P[1].



3.3.13. Fine adjustment of the tracking motion

In systems that require precision, fine adjustments are made to the tracking motion. See "I.5.12. Fine adjustment of the tracking motion(P.386)".

4. Other preparations before setup

This chapter explains other preparations not explained in the setup procedures in Chapters 3 and 4, such as settings and precautions for creating a robot ring and using iRPickTool.

It also explains the preparation for when creating a robot ring and using iRPickTool when there are multiple robots.

4.1. Network creation

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

If you are setting up iRPickTool on a personal computer, even when there is only one robot controller in the system, network setup is necessary.

4.1.1. 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 dedicated local network for communication among robot controllers (“a robot ring” hereinafter) to prevent interference from other communication traffic.

The R-50iA controller has up to four Ethernet ports (two ports by default, and four ports when an optional board is added), allowing a single robot controller to belong to four different networks. If there is a need to connect to networks other than a robot ring, such as a factory network, the ports for the factory network and the network for the robot ring should be separated.

For example, when using port 1 to connect the robot controller to the factory network, use a port other than port 1 to connect to the robot ring network as shown in the figure below. In this case, only the robot controllers involved with iRPickTool and the teaching PC for iRPickTool should be connected to the network used for the robot ring.

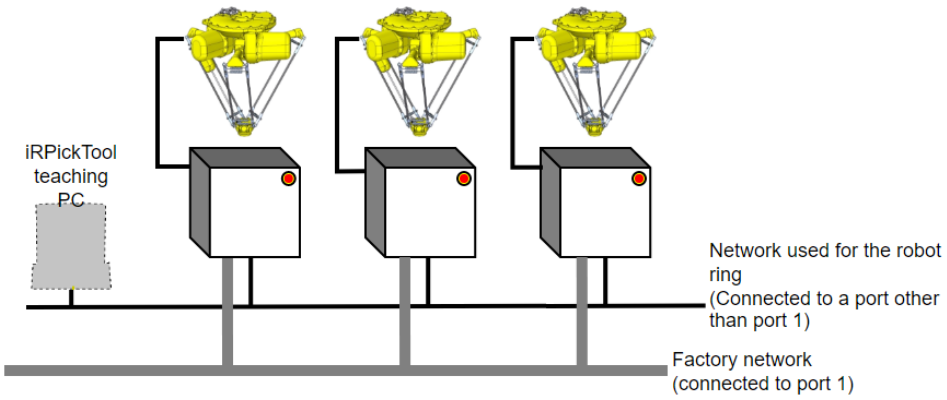


Fig. 4.1.1-1 Connection examples for two different networks

Connect the Ethernet cable to the port of the main board corresponding to the port number you want to use. For example, connect the cable to the CS01C port on the main board when using port 1 and to the CS01D port when using port 2.

NOTE

1. If the robot controller is only connected to the robot ring network, it doesn’t matter which Ethernet port is used.
2. Connect the teaching PC for setting up iRPickTool, which will be described later, to the network for the robot ring.

4.1.2. Caution on setting up a network

There are the following precautions on setting up a network. Observe these restrictions when performing setup.

- As described in "I.4.1.1. Connecting a network cable(P.241)", one robot controller has two ports. Set the IP address and subnet mask of each port used according to the instructions described in "I.4.1.3. Setting up the IP address(P.242)".
- To use both ports 1 and 2, set different network addresses for ports 1 and 2. For details, see "I.4.1.3. Setting up the IP address(P.242)".

Observe the following precautions when the system contains multiple robot controllers. When the system contains only one robot, skip them.

- When setting IP addresses, assign a unique robot name to each robot controller.
- When setting IP addresses, use consecutive numbers for the IP addresses of the port used for communication among robot controllers. For example, when the system contains four robot controllers and port 2 is used, the IP address of port 2 of each robot controller is set sequentially to 172.16.0.1, 172.16.0.2, 172.16.0.3, or 172.16.0.4.
- When the Ethernet Encoder is used, the IP address of the robot controller connected to a Pulsecoder must be the starting number. In the above example, set the IP address for port 2 of the robot controller connected to a Pulsecoder to 172.16.0.1.

4.1.3. Setting up the IP address

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

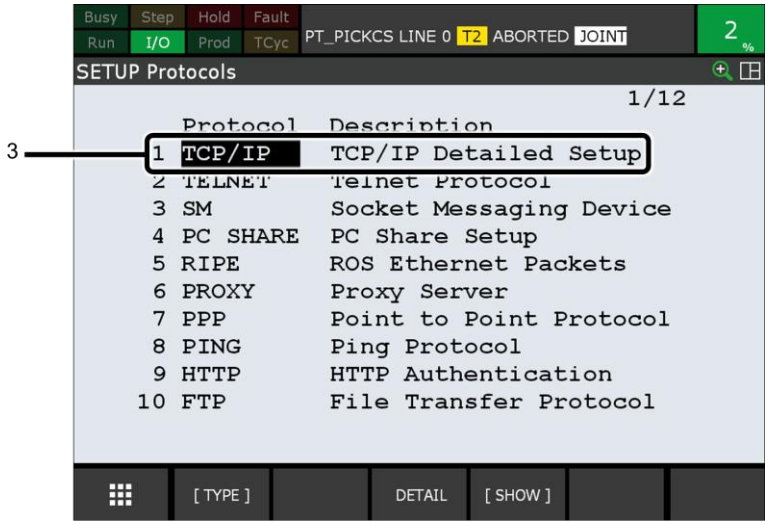
1 **Press the [MENU] key on the teach pendant of the robot controller.**
The menu appears.

2 **Choose [SETUP] → [Host Comm] from the menu.**

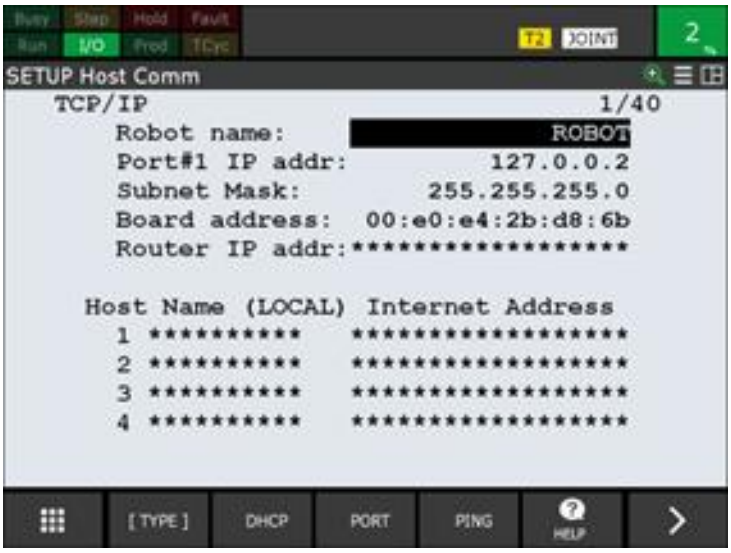


The [SETUP Protocols] screen will appear.

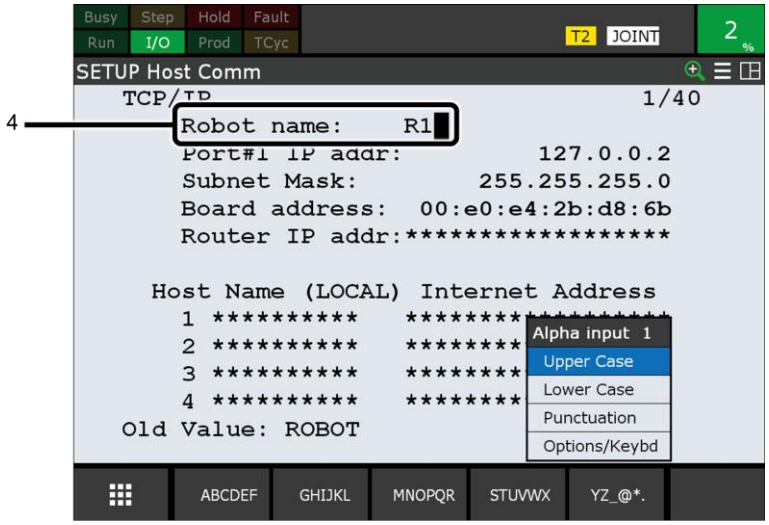
3 **Move the cursor to [TCP/IP] and press the [ENTER] key.**



The [SETUP Host Comm] screen will appear.



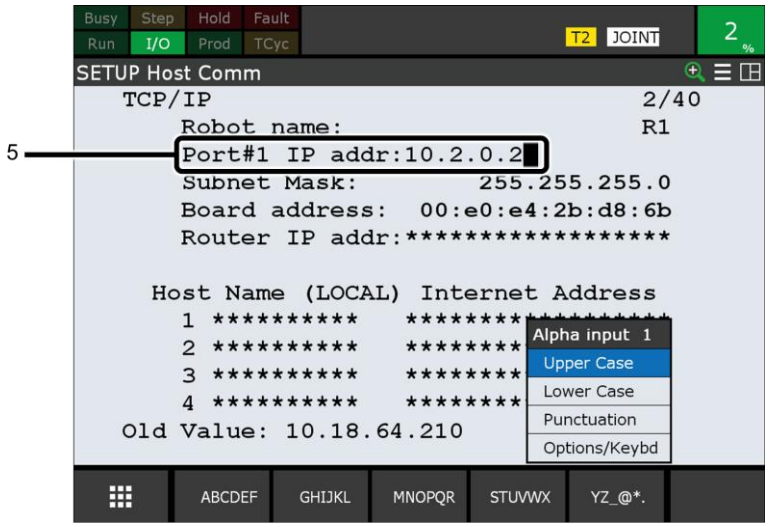
4 Enter the name of this robot controller in [Robot name]



⚠CAUTION

A robot name can contain only alphanumeric characters and minus sign. The first character must be an alphabetic character. The last character must not be a minus sign. The robot name cannot contain any spaces.

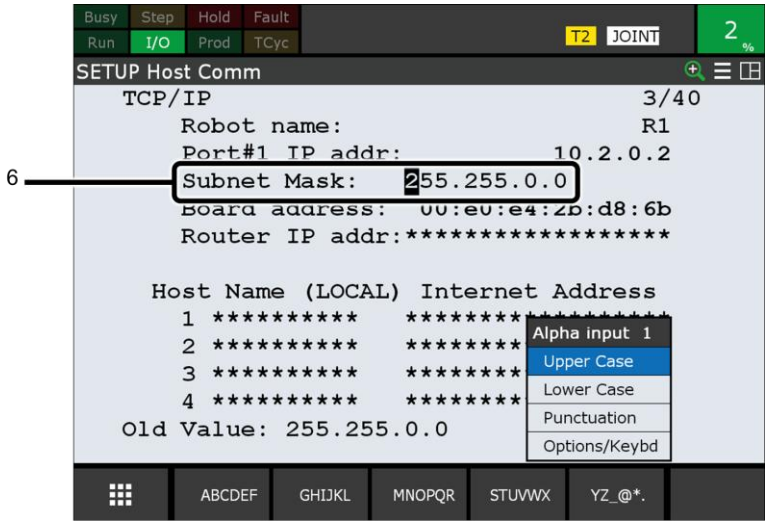
5 Enter the IP address of the robot controller in [Port#1 IP addr].



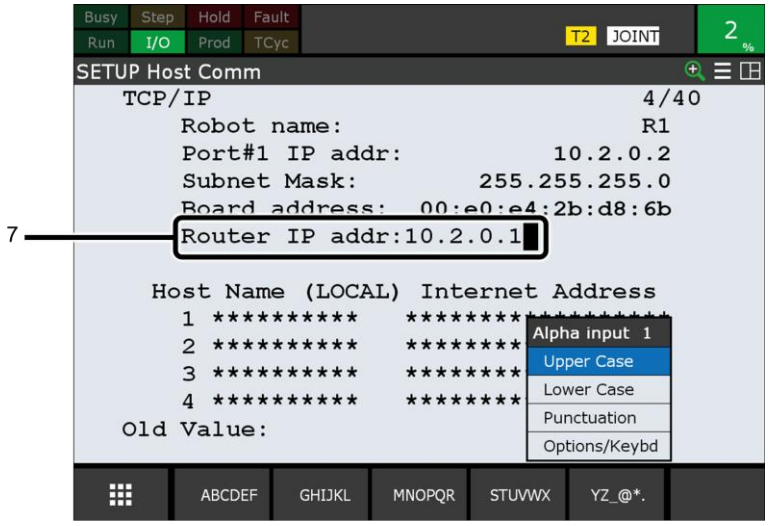
CAUTION

The IP address must not contain any space or unnecessary “0”. If there are any spaces or unnecessary 0’s in the IP address, correct communication is not possible.

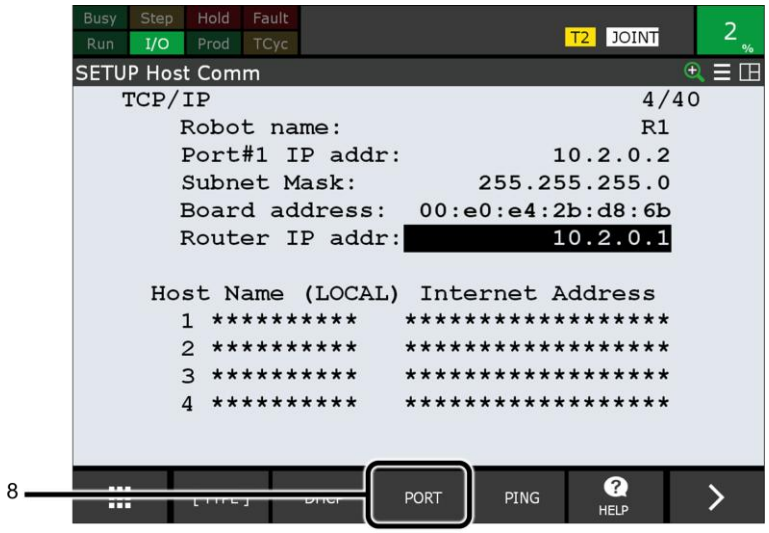
6 Enter a subnet mask in [Subnet Mask].



7 Enter the IP address of the default gateway in [Router IP addr].



8 Press the F3 [PORT] key and select a port.



9 Set the IP address and subnet mask for port 2 in the same way as in steps 5 and 6.

10 Reboot the robot controller.

⚠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.

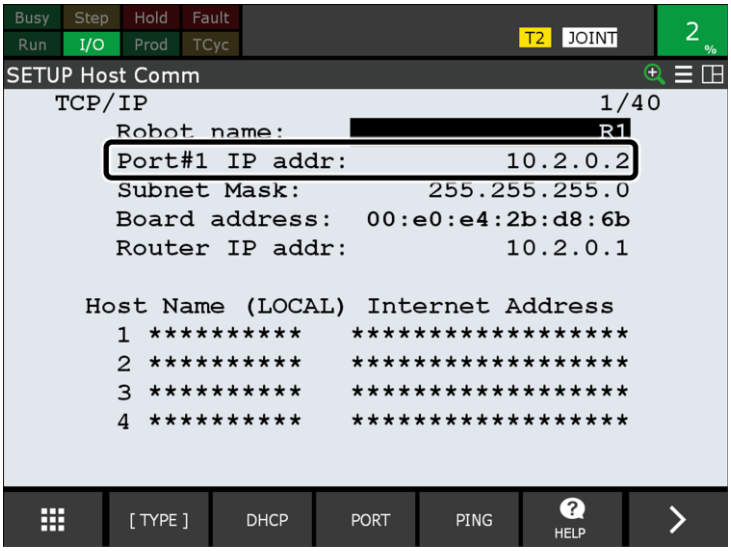


Fig. 4.1.3-1 Example of setting port 1

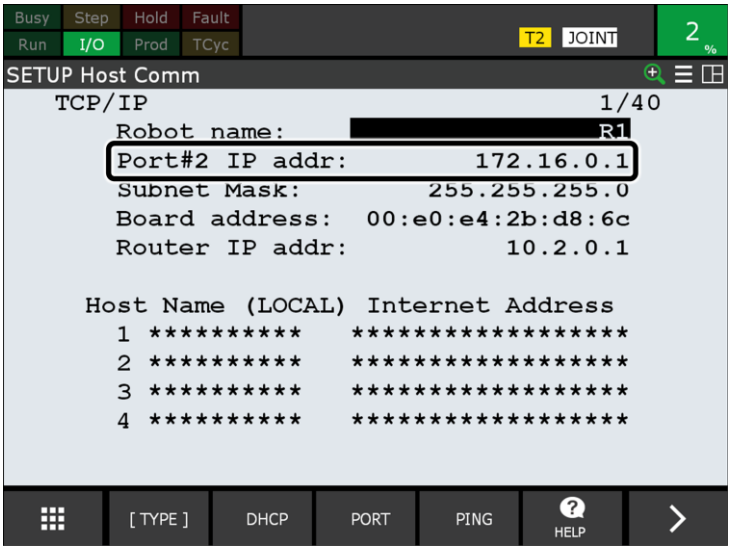


Fig. 4.1.3-2 Example of setting port 2

⚠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.

4.1.4. 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, see the chapter, “ROS interface packets over Ethernet (RIPE)” in the "Network and Security Function OPERATOR'S MANUAL"(B-84604EN).

⚠CAUTION

When using the ROS Interface Packets Over Ethernet (RIPE), the software series/versions of all the connected controllers need to match.

Ring leader and ring member

A robot ring consists of one ring leader controller and one or more ring members. With iRPickTool, any robot controller can be the ring leader. When the Ethernet Encoder is used, however, it is recommended to set the robot controller connected to a Pulsecoder as the ring leader controller.

When setting up a robot ring, set up the ring leader controller first, then set up all the ring member controllers.

⚠CAUTION

- For setting up a robot ring, an external memory device (a USB memory to be inserted into UD1 or an SD card to be inserted into MC) is required. For how to set up and use external memory devices, refer to "OPERATOR’S MANUAL(Basic Operation)"(B-84464EN).
- Use a network port that is not utilized for another network, such as the factory Ethernet, for the network port used in the robot ring. For details, refer to “I.4.1.1. Connecting a network cable(P.241)”.
  Additionally, it is recommended to configure the robot ring network as a dedicated local network.
- Prior to configuring the robot ring, set the TCP/IP for the port to be used for the robot ring on all robot controllers that will communicate. For the TCP/IP setting procedure , refer to “I.4.1.3. Setting up the IP address(P.242)”.
  The IP addresses set at this time must be configured in sequential order, starting with the ring leader controller.

⚠CAUTION

For example, if the IP address of port 2 on the ring leader controller is set to “192.168.0.101” and used for the robot ring, the IP address of port 2 on the first ring member controller should be “192.168.0.102”, the second ring member controller should be “192.168.0.103”, and so on, assigning sequential IP addresses.

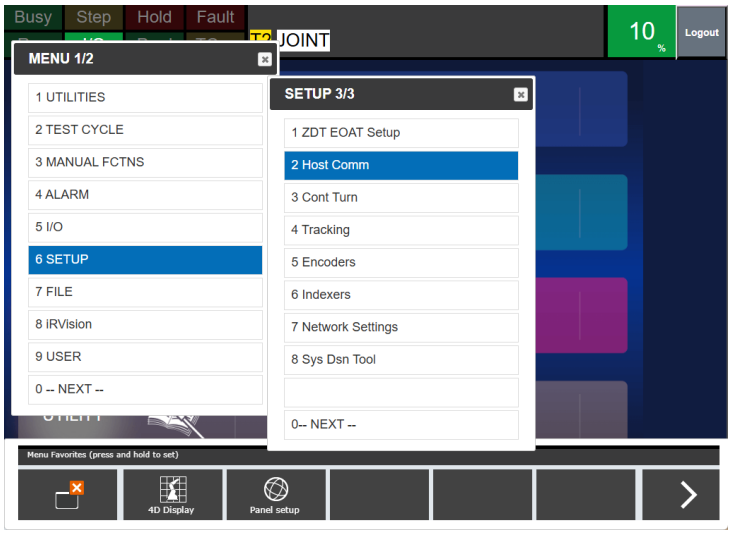
NOTE

On the teach pendant screens or other interfaces, the ring leader may be labeled as “Leader” and the ring members as “members”.

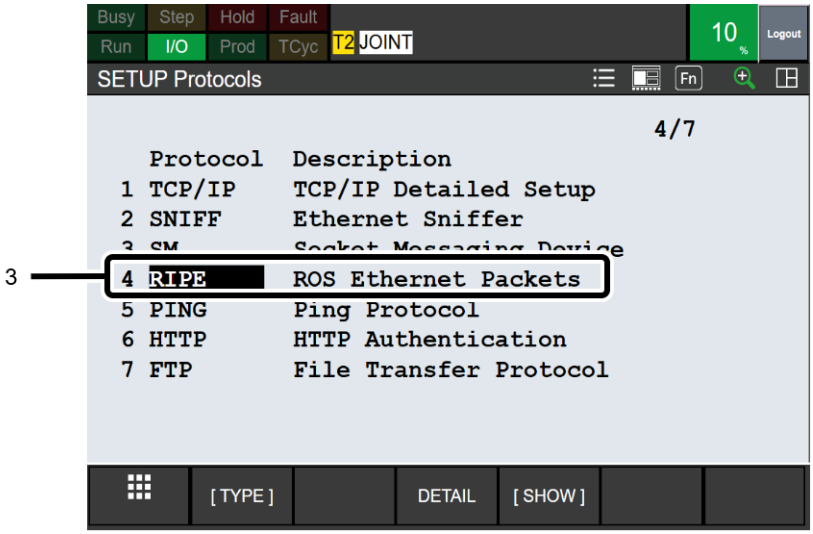
Ring leader settings

Follow the procedure below to set up the ring leader controller.

1. **Press the [MENU] key on the teach pendant of the robot controller.**
2. **Choose [SETUP] → [Host Comm] from the menu.**

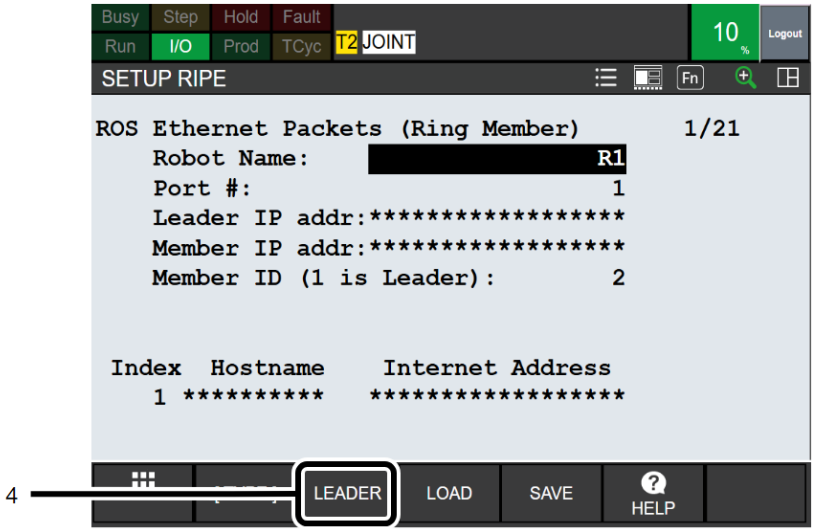


3. **Position the cursor on [RIPE] and press the [ENTER] key.**



4. **Press F2 [LEADER].**

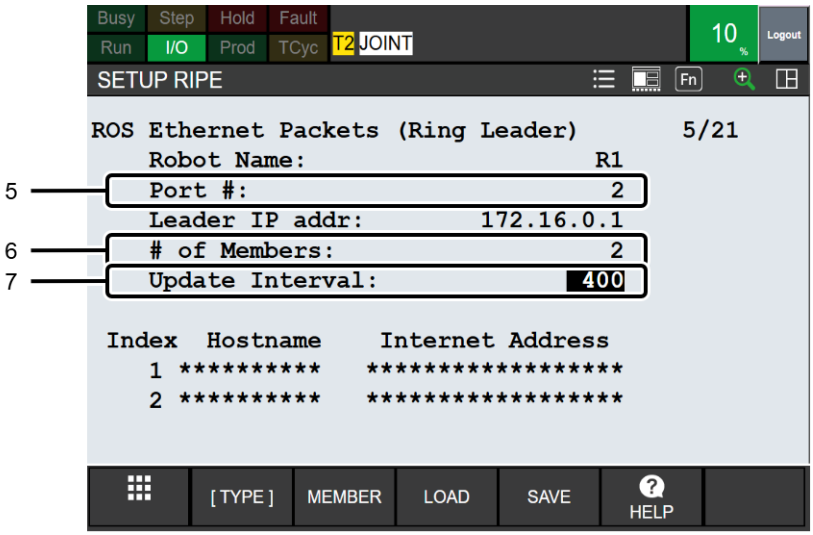
   This step is unnecessary when the screen for the ring leader is already open and F2 indicates [MEMBER].



5 **In [Port #], set the port number connected to the network used for the robot ring.**

6 **In [Number of Members], set the number of robot controllers that join the robot ring network.**

7 **Set [Update Interval] if necessary. At intervals of this setting, each robot is checked to see whether it is online.**



8 **In the table below [Update Interval], set the information of all the robot controllers connected to the robot ring.**

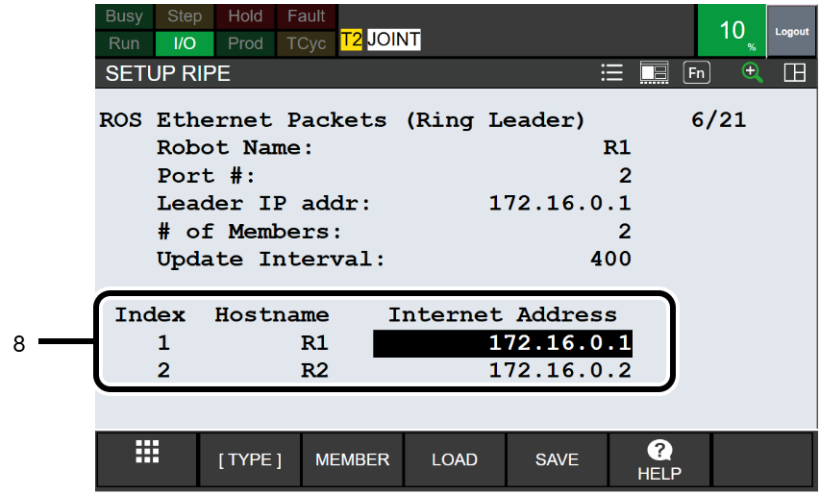
In [Host Name], set the robot name of each robot controller. In [Internet Address], set the IP address set in the robot ring communication port of each robot controller.

⚠CAUTION

In the first row of the table (the row whose [Index] is “1”), enter the information of the ring leader controller. Additionally, the configuration updates for each robot controller used in the robot ring will be applied in the index order listed in this table. Write down the contents of the table as needed.

NOTE

[Index] in this table is called the [Member Index].



9 Insert an external memory device into the robot controller. Insert the USB memory into port UD1 and the SD memory into port MC.

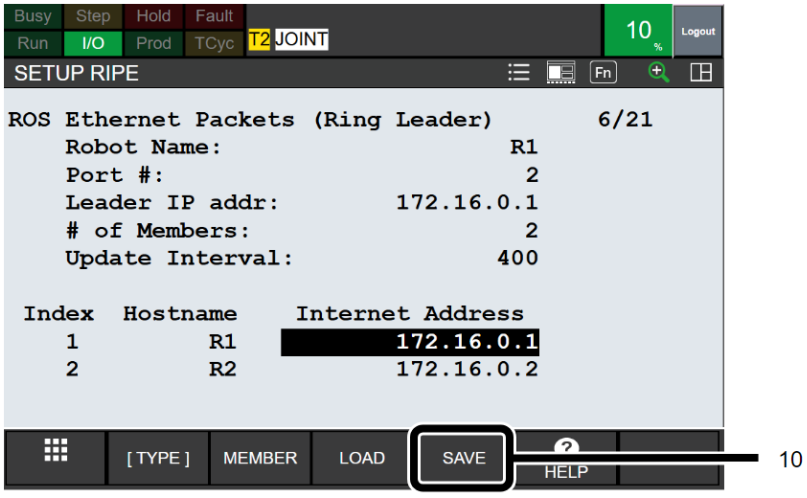
10 Press F4 [SAVE].

Save ROSIPCFG.XML to the inserted external memory device.

If F4 [SAVE] is not displayed, press [> (NEXT)] to switch screens.

NOTE

ROSIPCFG.XML will be saved in the root directory of the external memory device.

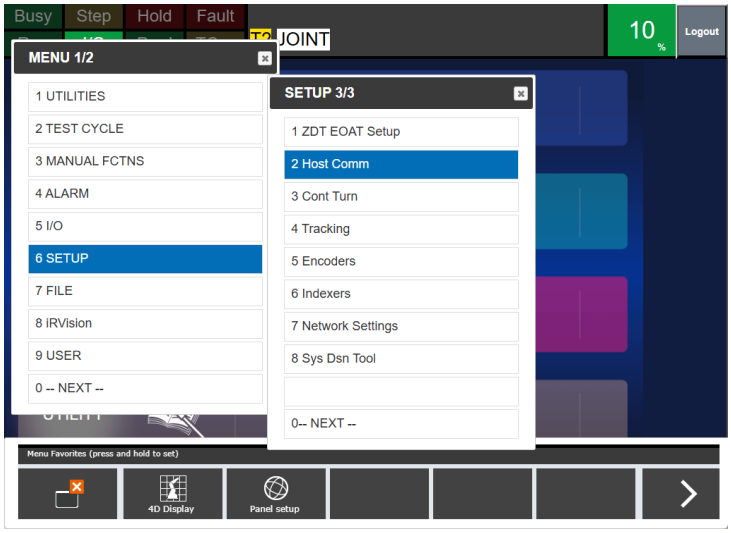


Ring member settings

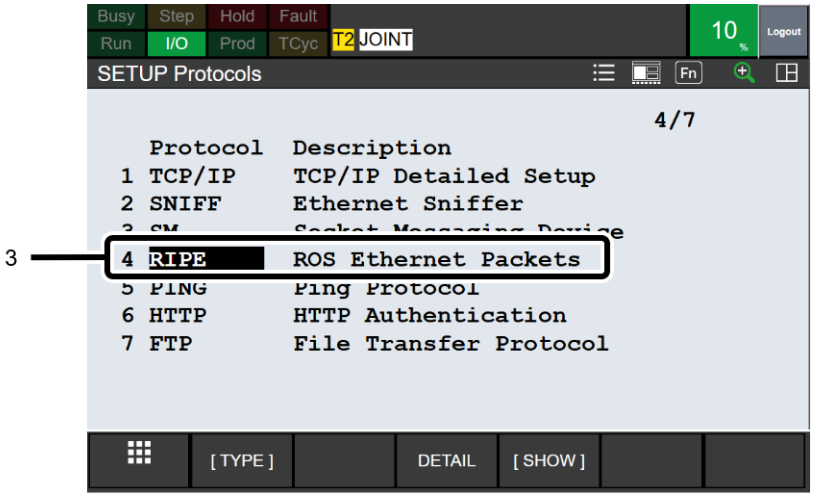
Follow the procedure below to set up each ring member controller.

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

2 Choose [SETUP] → [Host Comm] from the menu.

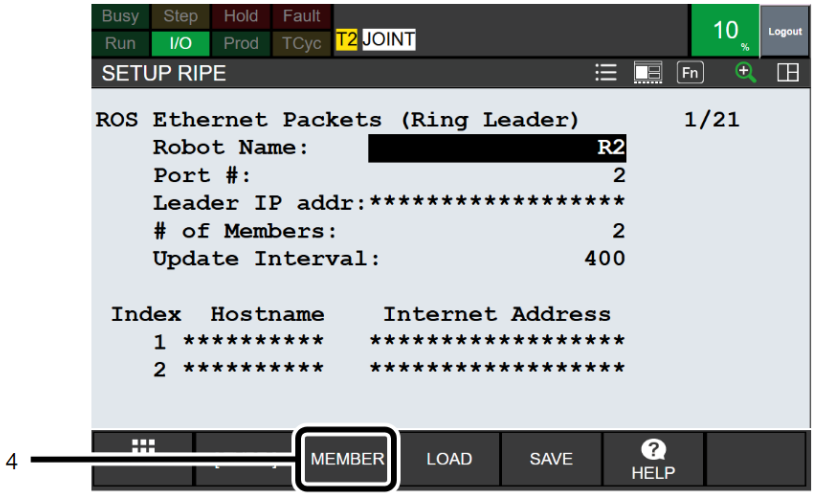


3 **Position the cursor on [RIPE] and press the [ENTER] key.**

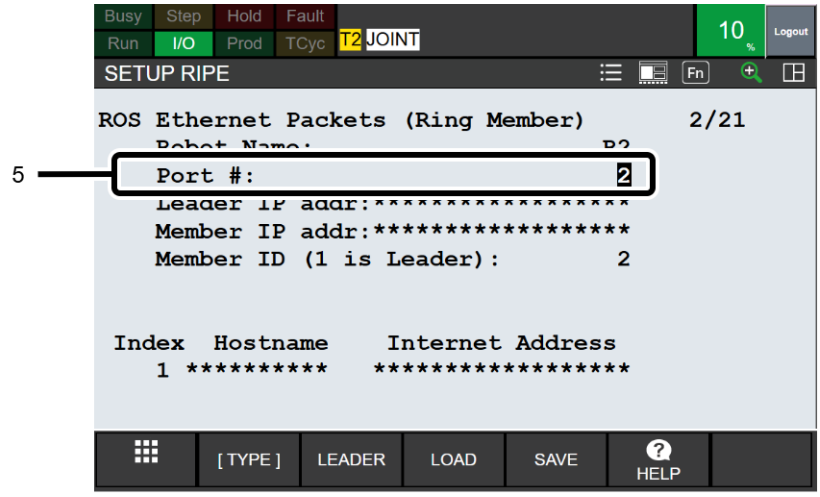


4 **Press F2 [MEMBER].**

This step is unnecessary when the screen for the ring member is already open and F2 indicates [LEADER].



5 **In [Port #], set the port number connected to the network used for the robot ring.**



6 Insert the external memory device on which ROSIPCFG.XML is saved into the robot controller. Insert the USB memory into port UD1 and the SD memory into port MC.

⚠CAUTION

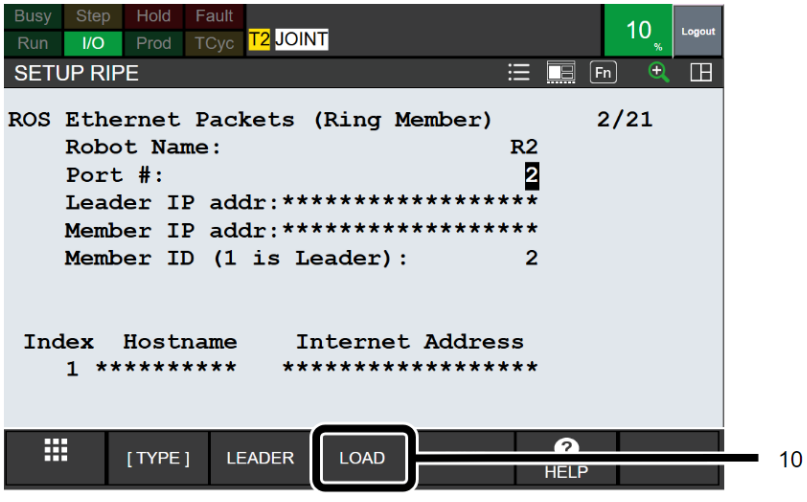
ROSIPCFG.XML must be saved in the root directory of the external memory device.

7 Press F3 [LOAD] to load ROSIPCFG.XML from the inserted external memory device.

If F3 [LOAD] is not displayed, press [> (NEXT)] to switch screens.

⚠CAUTION

The loaded settings will not be reflected on the screen until the robot controller is restarted.



8 Repeat the same procedure in all the ring member controllers.

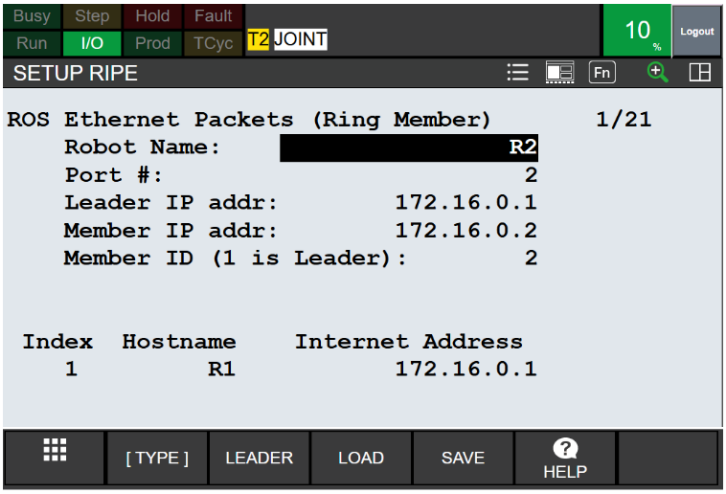
Applying and verifying the settings

Once setting is complete in the ring leader controller and all the ring member controllers, apply and verify the settings according to the steps below.

CAUTION

When each robot controller is restarted, the settings in ROSIPCFG.XML will be reflected. However, the robot controllers need to be restarted in the order of the index numbers listed in the table configured in step 8 of "Ring leader settings ".
First restart the robot controller with Index 1 (the ring leader) and verify its settings. After that, proceed to restart the robot controllers with Index 2, 3, and so on, in numerical order.

1. **Restart the ring leader controller and the ring member controllers in order of their index numbers.**
2. **On the ring leader controller, open the screen described in Step 4 of "Ring leader settings", and on the ring member controllers, open the screen described in Step 4 of "Ring member settings", and confirm that the displayed settings are as intended.**



4.2. Pulsecoder installation, connection and setup

Install a Pulsecoder and connect it to a robot controller.

This section explains the setup for when installing a Pulsecoder and connecting to main boards, and when using with multiple robots.

For connection and setup of the Pulsecoder and robot controller, see "I.2. iRPickTool Basic (basic functions) startup procedures(P.26)" and "I.3. iRPickTool (basic functions and Plug & Play) startup procedures(P.152)".

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".

In the tracking system, the control unit may contain the sensor I/F board as the signal interface and be connected to the sensor distribution board as needed. Alternatively, the separate interface device (SDU) may be attached in the cabinet. When the SDU is attached, an FSSB optical fiber cable for connection is also required.

In addition, cabling (encoder cables) for external hardware (encoder) is required to track conveyors, pallets, tables, or other items. Only the αiA4000S Pulsecoder A860-2052-T321 (both incremental and absolute) can be used in the R-50iA / R-50iA Mate Controller. For the specifications, see Fig. C.2-1(P.712), Table C.2-1(P.712), and Fig. C.2-2(P.713).

Hardware	Description
Interface device	Insert the sensor I/F board into the appropriate slot of the control unit. (See Fig. C.1-1(P.711)) When connecting two encoders to the sensor I/F board or when using the force sensor at the same time, attach the sensor distribution board in the cabinet. (See Fig. C.1-2(P.711)) For example, when you want to use three or more encoders, attach the separate interface device (SDU) in the cabinet.
Encoder	Use the αiA4000S Pulsecoder A860-2052-T321. To obtain a sufficient resolution, use an appropriate gear and reducer. (300 pluses/mm or more is recommended)

4.2.1. Installing a Pulsecoder

Install a Pulsecoder on a conveyor.

Pulsecoder

Install the Pulsecoder at a location outside the robot operation area so that the Pulsecoder does not interfere with the robot during operation. When a camera is used to detect parts, install the Pulsecoder outside the camera's field of view.

Attaching a rotary disk to the tip of the Pulsecoder so that the rotary disk rotates by directly touching the conveyor belt enables the conveyor travel distance to be measured more accurately than attaching the Pulsecoder directly to the drive or follower axis of the conveyor.

Ensure that no slippage occurs between the rotary disk and conveyor belt. If a slippage occurs, the conveyor travel distance cannot be measured accurately, resulting in degraded robot handling precision. Pressing the Pulsecoder against the conveyor using a spring is effective in securing the Pulsecoder.

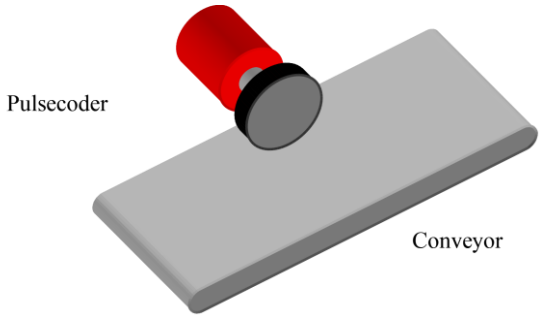


Fig. 4.2.1-1 Installation position of Pulsecoder

4.2.2. 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.

For how to connect the Pulsecoder, see "C. αiA4000S Pulsecoder(P.710)".

4.2.3. Setting up Pulsecoders that are connected to the robot controller

This section explains the setup for when using an αiA4000S Pulsecoder by connecting it to the sensor I/F board and sensor distribution board of the robot controller.

Set the following items.

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

1 **Press the [MENU] key on the robot controller's teach pendant.**

The menu appears.

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

3 **Press F3 [ENCODER] and the [Encoder Number].**

When other Pulsecoders are not connected, select encoder No. 1. When connecting two αiA4000S Pulsecoders to the robot controller, select 1 for the Pulsecoder connected to **CRN3A of the sensor distribution board** and 2 for the Pulsecoder connected to **CRN3B of the sensor distribution board**.

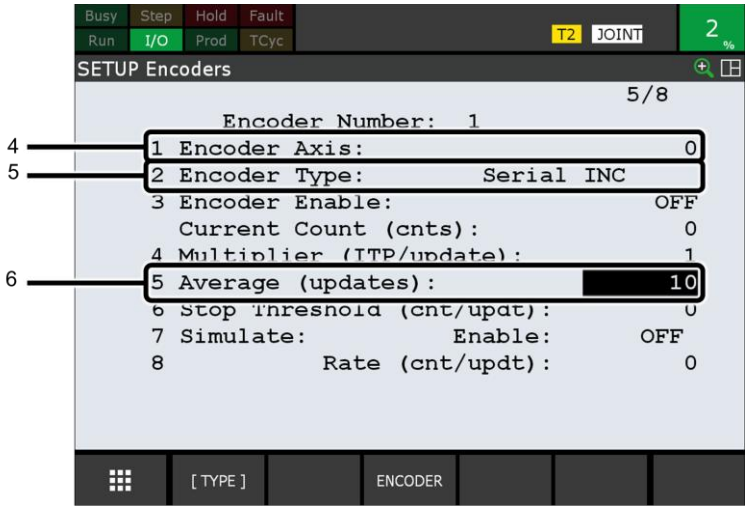
4 **Move the cursor to [Encoder Axis] and enter the same value as [Encoder Number].**

5 **Move the cursor to [Encoder Type], press F4 [CHOICE], and select [Sensor I/F Board].**

6 **Move the cursor to [Average (updates)] and input the cumulative accumulated number of instantaneous speeds (width of moving average).**

By setting the accumulated number of instantaneous speeds, the robot will stop smoothly rather than suddenly, even if the conveyor suddenly stops while the robot is tracking.

Normally, enter "20".



7 **Set up all the Pulsecoders connected to the robot controller in the same way.**

8 **Cold start the robot controller.**

4.2.4. Setting up Pulsecoders for multiple robots

If you add the Ethernet Encoder function (R762) option to your order, pulse count information can be provided to multiple robots via Ethernet instead of a pulse multiplexer. The robot controller connected to the Pulsecoder will act as the producer and transmit pulse count information to other robot controllers connected via Ethernet.

This section explains the setup for when using Ethernet Encoders with multiple robots.

If there is only one robot, see "I.2.2.1.2. Setting up Pulsecoders(P.28)".

The setup for when using Ethernet Encoders with multiple robots is as follows.

Settings in a robot controller that is connected to a Pulsecoder

The following describes how to set up a Pulsecoder connected to a robot.

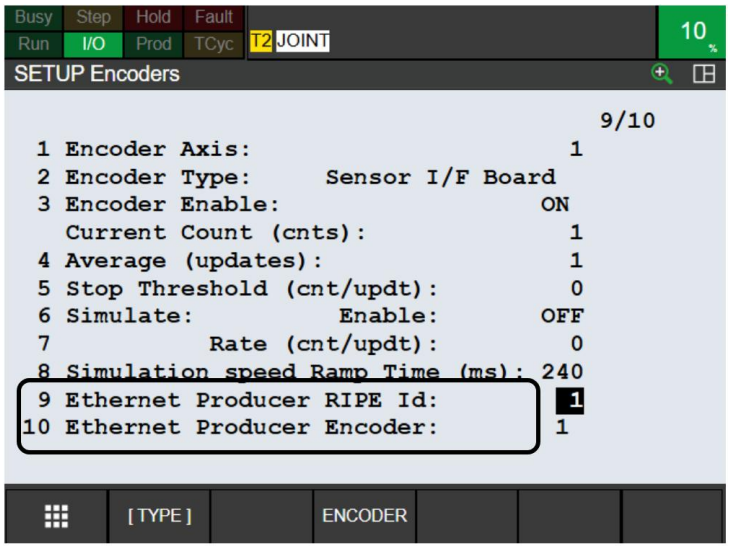
⚠CAUTION

It is recommended to connect the Pulsecoder to the ring leader controller of the robot ring to reduce the communication traffic in the robot ring.

1 **Perform setup steps 1 to 7 in the same way as “I.2.2.1.2. Setting up Pulsecoders(P.28)” or “I.3.2.1.2. Setting up Pulsecoders(P.154)”.**

The displayed Encoder setup screen is as shown below, however.

When a robot controller has the Ethernet Encoder option, items [Ethernet Producer RIPE Id] and [Ethernet Producer Encoder Number] are added.



2 **Position the cursor on [Ethernet Producer RIPE Id] and enter the RIPE ID of the robot controller.**

The RIPE ID for the ring leader controller of the robot ring is 1. For a ring member controller, it is the same as the member ID. For how to check the member ID, refer to “I.4.1.4. Setting up the robot ring(P.246)”.

3 **Position the cursor on [Ethernet Producer Encoder Number] and enter the number of the selected Encoder.**

The number to enter is the encoder number that is set.

For example, enter 1 when connecting the Pulsecoder to the CRN3A connector of the sensor distribution board and 2 when connecting it to CRN3B.

4 **Turn the power to the robot controller off, then on again.**

Settings in other robot controllers

1 **Open the Encoder setup screen.**

You can select any [Encoder Number].

2 **In [Ethernet Producer RIPE Id] and [Ethernet Producer Encoder Number], set the same values as those set in the robot controller connected to the Pulsecoder.**

3 **You do not need to change other items.**

4 **Turn the power to the robot controller off, then on again.**

4.3. Camera installation and connection

Install a camera and connect it to a robot controller.

For a queue managed tracking system, skip this section.

Selecting a camera to be used

For a visual tracking system, a digital camera (monochrome, color) and an analog camera are available.

Connecting the camera

Connect the camera to a robot controller. For details, see “2. Configuration” in "Sensor Mechanical Unit / Control Unit OPERATOR’S MANUAL"(B-84544EN).

⚠CAUTION

If you want to use the Ethernet Encoder, connect the camera to the ring leader 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 direction in which the camera image is larger. Generally, select the transverse direction. This ensures a wider field of view in the direction in which the conveyor moves.

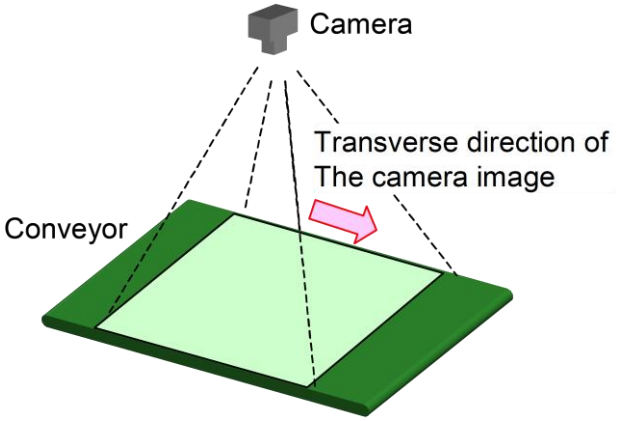


Fig. 4.3-1 Example of mounting a camera

Installing the camera and photo eye

When the system is designed to snap an image with a camera after the signal from a photo eye is input, the sensor monitors the DI (RI) signal. After the DI (RI) signal is input, the sensor waits for the conveyor to move a specified distance, then snaps an image with the camera.

The time from the input of the DI (RI) signal until an image is snapped by the vision system is about 0.1 seconds depending on the configuration of hardware used. For this reason, the expected timing of image snapping 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 × 0.1 sec.

You can install the camera farther downstream.

In this case, you can snap an image of a part in the desired position by adjusting the sensor setup parameter [Trigger Offset]. For details of [Trigger Offset], see “I.5.5. Setting up a sensor(P.288)”.

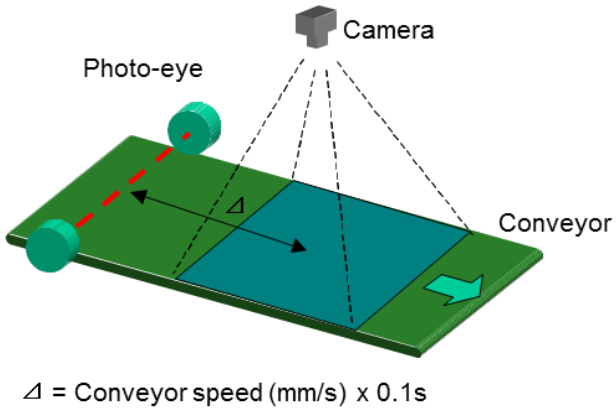


Fig. 4.3-2 Example of mounting a camera and a photo eye

4.4. Setting up the tool frame

Setup of a tool frame sets the tracking frame and defines the position of the tool center point (TCP) and tool posture with a pointer tool that is used in camera calibration.

When multiple robots are used, it is necessary to make this setting for all robots.

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, see the section, “Setting coordinate systems” in the chapter, “Setting up the robot system” in the "OPERATOR’S MANUAL(Basic Operation)"(B-84464EN).

The pointer is used for tracking frame setting, camera calibration, sensor/tray position setting, and other purposes. After the completion of required work, the pointer can be detached.

After the pointer is detached, however, the set tool center point is invalidated and it is necessary to set a tool center point again unless the pointer is precisely restored mechanically.

For this reason, it is recommended that the mechanical design for the pointer be made to be restored to the robot in preparation for operation using the pointer after that.

It is also recommended that the pointer be left mounted immediately before robot program position teaching described in “I.5.11. Teaching the position to robots(P.385)” or “I.6.7. Teaching the position to robots(P.439)”.

4.5. Calibration grid

For visual tracking camera calibration, a calibration grid is used. See the notes on creating a grid pattern described in calibration grid in the "iRVision OPERATOR'S MANUAL (Reference)"(B-84434EN).

FANUC provides a standard set of calibration grids.

5. Basic functions reference

This chapter explains the menu items and setup procedures when using iRPickTool Basic (Basic Functions) or iRPickTool (Basic Functions + Plug & Play).

Furthermore, examples of robot TP programs and KAREL programs are also given.

For the basic setup procedure for the iRPickTool Basic (basic function), see "I.2. iRPickTool Basic (basic functions) startup procedures(P.26)". For the setup procedures for iRPickTool (Basic Functions + Plug & Play), see "I.3. iRPickTool (basic functions and Plug & Play) startup procedures(P.152)".

The iRPickTool Basic menu and related items in this manual are shown in the figure below.

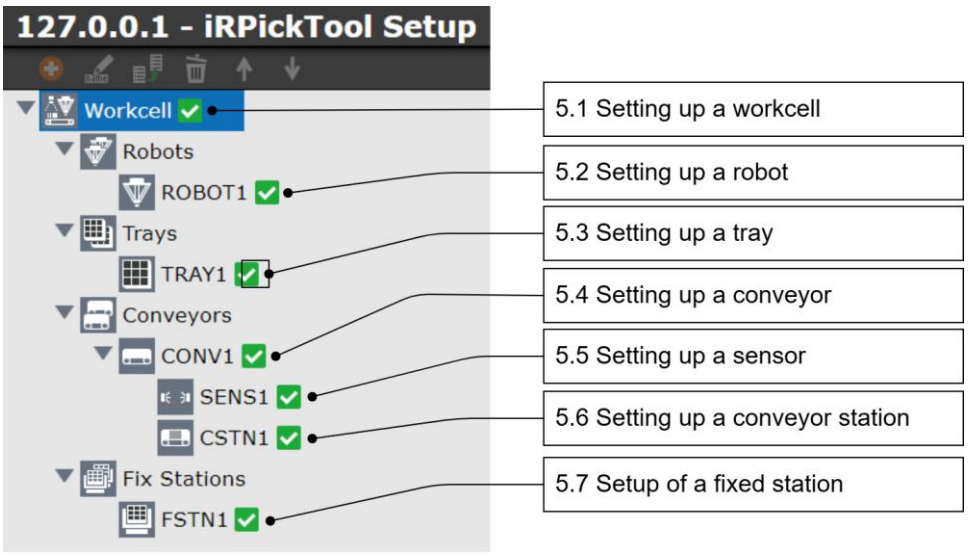


Fig. 5-1 iRPickTool Basic menu and related items

In addition to the iRPickTool Basic menu items in Fig. 5-1, "I.5.8. KAREL program(P.332)" and "I.5.9. Robot programs(P.367)" describe programs and "I.5.10. Reference position setting(P.382)" and "I.5.11. Teaching the position to robots(P.385)" describe teaching the robot position using specific examples as reference material for the basic functions.

Furthermore, "I.5.12. Fine adjustment of the tracking motion(P.386)" explains how to optimize the robot's tracking motions after finishing basic motion teaching.

"I.5.13. Precautions for when starting up multiple robots(P.390)" gives precautions for when starting production using multiple robots.

"I.5.15. Setup of pre-grouping(P.393)" 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.

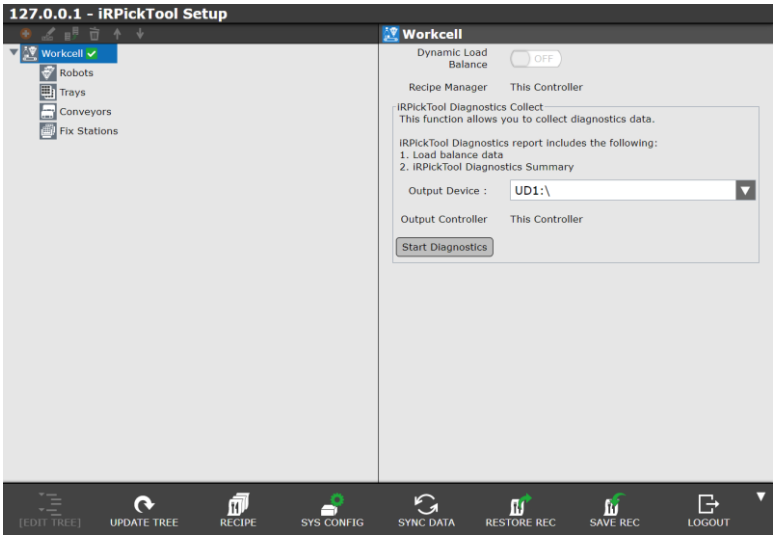
iRPickTool may be set up from any controller in a workcell.

However, you cannot setup iRPickTool simultaneously for two or more robot controllers with regard to robots connected by the robot ring.

Creating a new object

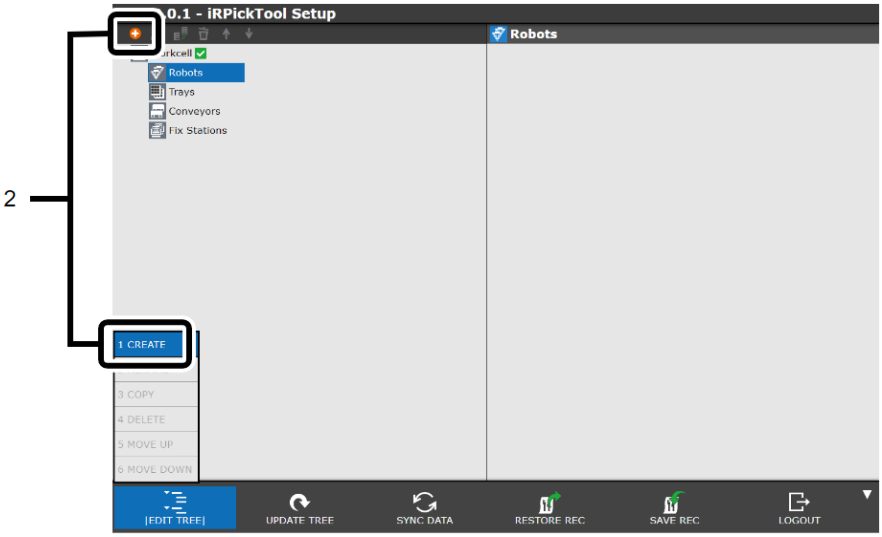
1. **Open the robot homepage.**
2. **Select [iRVision/iRPickTool] in the menu area and click [iRPickTool System Setup] in the content display area.**

   The following iRPickTool setup screen will appear.



On the iRPickTool setup screen, select the collection of objects that you want to create from the tree view on the left. For example, if you want to create a robot, follow the procedure below.

1. **Select [Robot].**
2. **Click the create button above the tree view, or select [CREATE] from the menu displayed by clicking [EDIT TREE].**



   A screen to create a new object will appear.

3. **On the Create Object screen, enter a name that is different from the other objects. The name displayed initially can also be used as it is.**

   When entering the name, note the following description.

   Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

4. **Click [OK] to create an object.**

   Click [CANCEL] to cancel creation of an object.

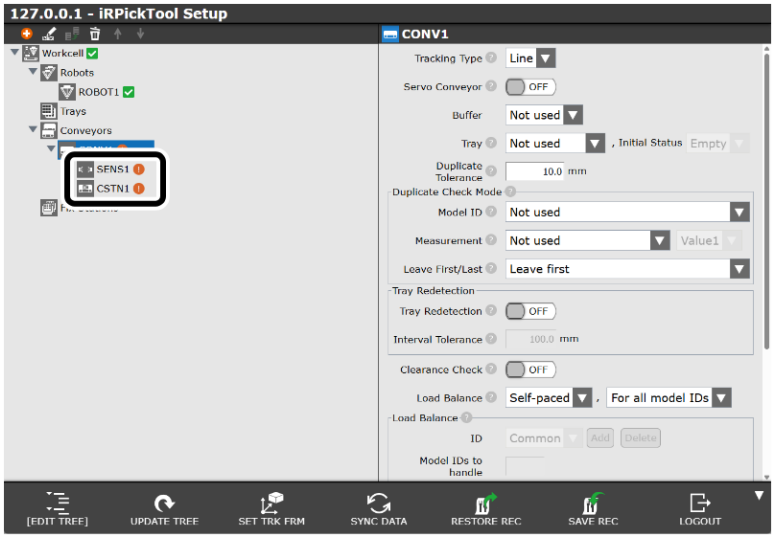


If you create a new object, the created object will be added to the tree view and its setup screen will appear.

With regard to the icon at the right of the objects that are displayed in the tree view, indicates the “untrained” status, and indicates the “trained” status.

Creating a new sensor and a new conveyor station

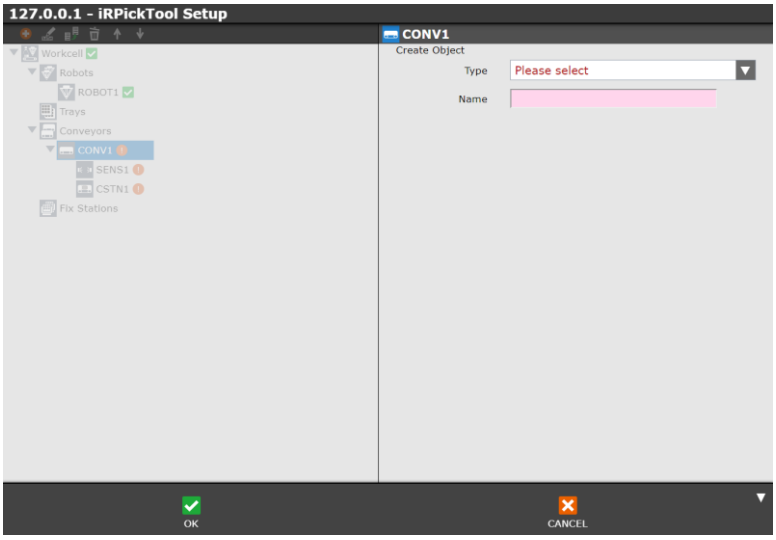
If you create a new conveyor, a sensor and a conveyor station are automatically created, as shown below.



If multiple robots perform a task on a conveyor, add a conveyor station (displayed as [CSTN1] in the tree view).
Furthermore, if there are multiple sensors on a conveyor, add a sensor (displayed as [SENS1] in the tree view).
If you want to create an object, follow the procedure below.

1 With the conveyor selected, click the create button above the tree view, or select [CREATE] from the menu displayed by clicking [EDIT TREE].

After a while, the following screen appears.



2 In [Type], specify whether the object you are adding is a conveyor station or sensor.

The default name appears in [Name], as when you create a new object of another type. Create a new object in the same way as with an object of another type.

Setting up an object individually

In the tree view on the left side of the iRPickTool setup screen, select the object you want to set up. Then, set each item on the setup screen on the right side.

With regard to the icon at the right of the objects that are displayed in the tree, indicates the “untrained” status, and indicates the “trained” status.

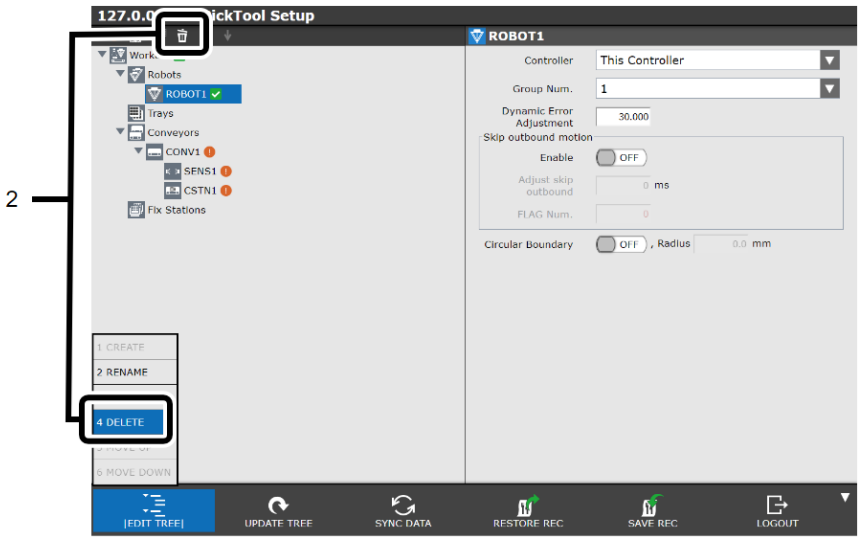
For details on each object, see “I.5.1. Setting up a workcell(P.266)” to “I.5.7. Setting up a fixed station(P.332)”. For Plug & Play, see “I.6.1. Workcell setup screen(P.422)” to “I.6.4. Setting up a fixed station(P.434)” as well.

Deleting an object

If you want to delete an object, follow the procedure below.

1 In the tree view at the left of the iRPickTool setup screen, select the object that you want to delete.

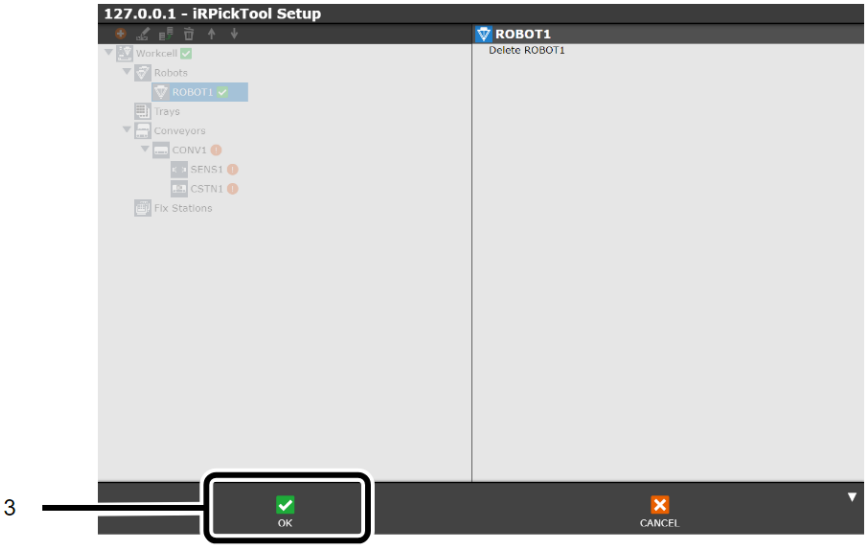
2 Click the delete button above the tree view, or select [Delete] from the menu displayed by clicking [EDIT TREE].



After a while, the following screen appears.

3 Click [OK] to delete the object.

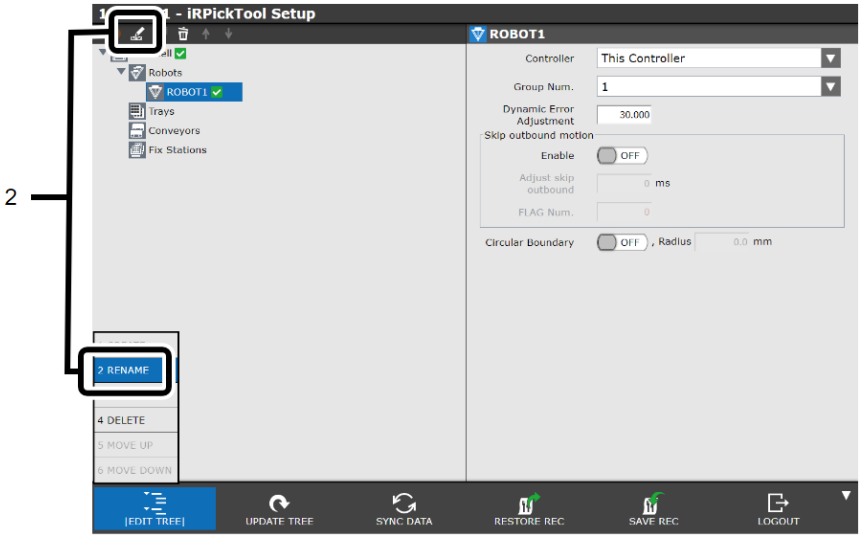
Click [Cancel] to cancel deletion of the object.



Renaming an object

If you want to rename an object, follow the procedure below.

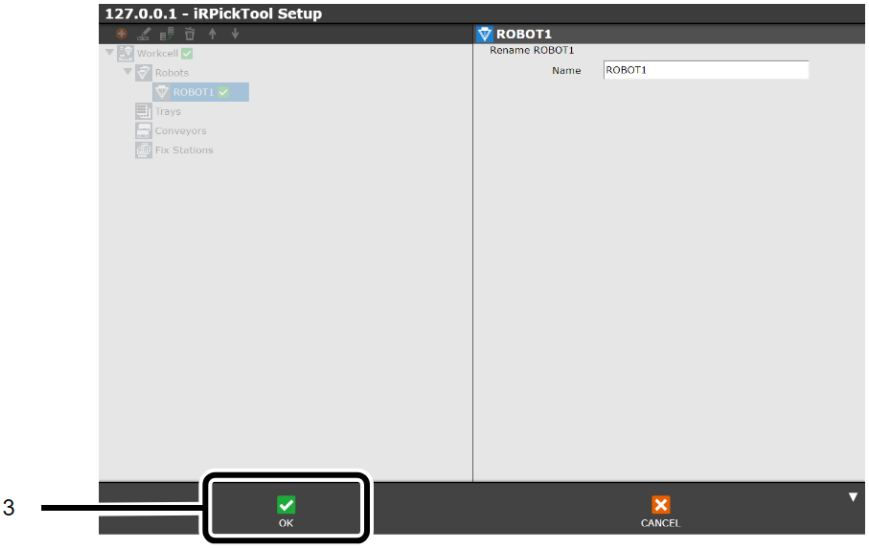
1. **In the tree view at the left of the iRPickTool setup screen, select the object that you want to rename.**
2. **Click the rename button**  **above the tree view, or select [RENAME] from the menu displayed by clicking [EDIT TREE].**



After a while, the following screen appears.

3. **Enter the new name of the object and click [OK].**

To cancel renaming the object, click [Cancel].



Moving an object in the order

To replace conveyor stations, change the order of the conveyor stations (objects).

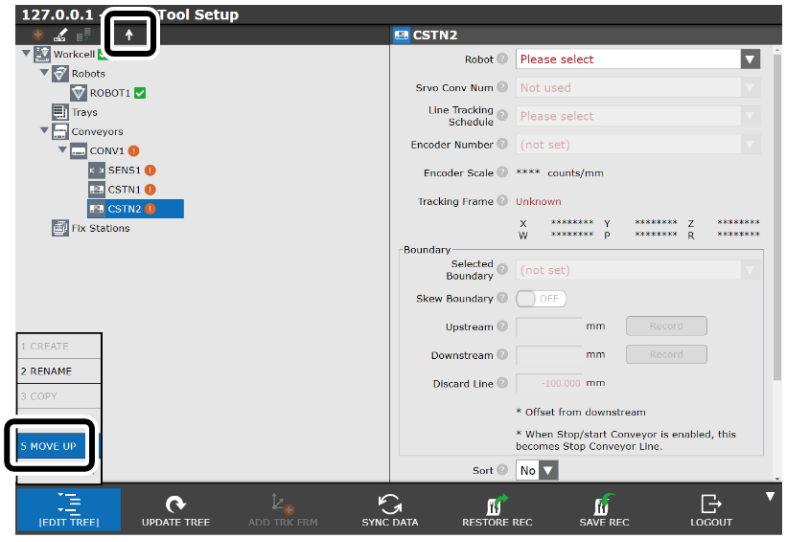
CAUTION

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

In the tree view on the left side of the iRPickTool setup screen, select the object you want to reorder.

To move it up in the order, press the move up button above the tree view, or select [Move up] from the menu displayed by clicking [EDIT TREE].

To move it down in the order, press the move down button above the tree view, or select [Move down] from the menu displayed by clicking [EDIT TREE].



Synchronizing data

For the iRPickTool settings, 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.

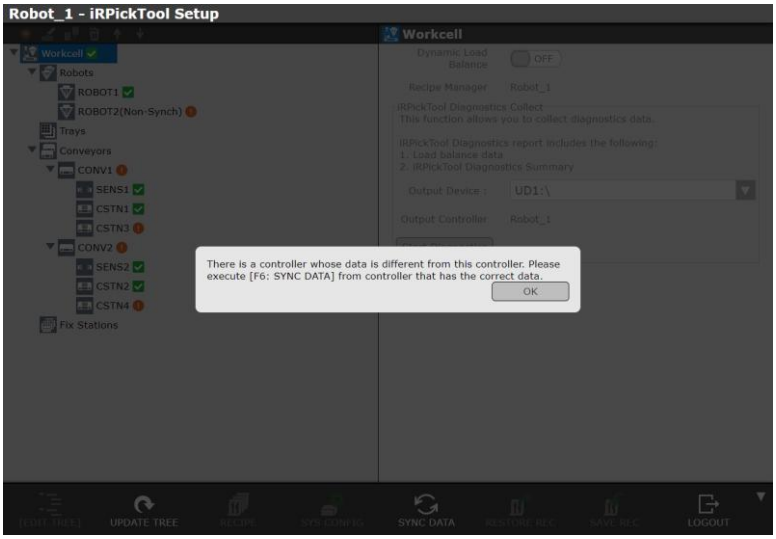
You can set up iRPickTool even if one of the robot controllers in the system is powered off.

However, if there is an offline robot controller when setting up iRPickTool, the settings will not be applied to the robot controller, resulting in 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 robot controllers.

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

2 **Click [OK].**

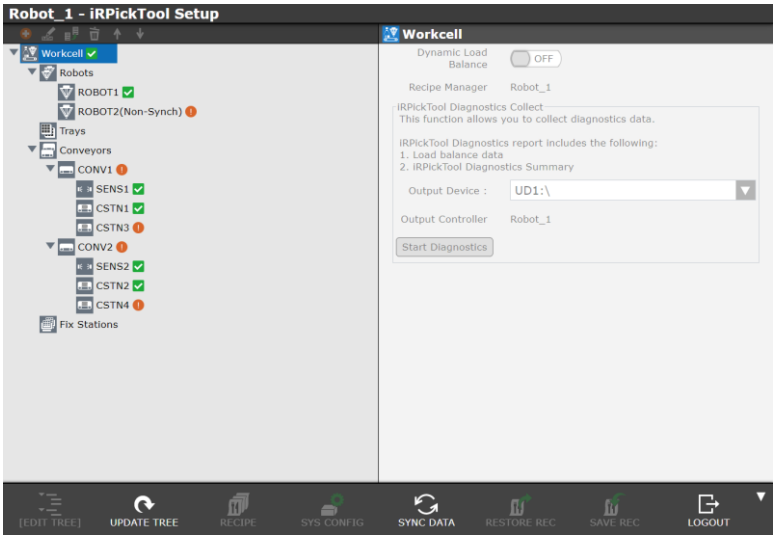


The screen enters a state where you cannot change the existing settings, as shown below.

If a robot controller that does not have the iRPickTool settings applied is selected for a robot, the robot will be displayed as “(Non-Synch)” in the tree view.

⚠CAUTION

A robot object displayed as “(Non-Synch)” indicates that a robot controller with settings that do not match has been selected.
Note that “(Non-Synch)” is not displayed on objects with settings that do not match.



3 **Check the settings of each individual object to see whether it is appropriate to transfer those settings to the other robot controllers.**

To perform synchronization based on the settings of another robot controller, click [LOGOUT] to close the iRPickTool setup screen. Then, open the iRPickTool setup screen for the other robot controller and check its settings as instructed above.

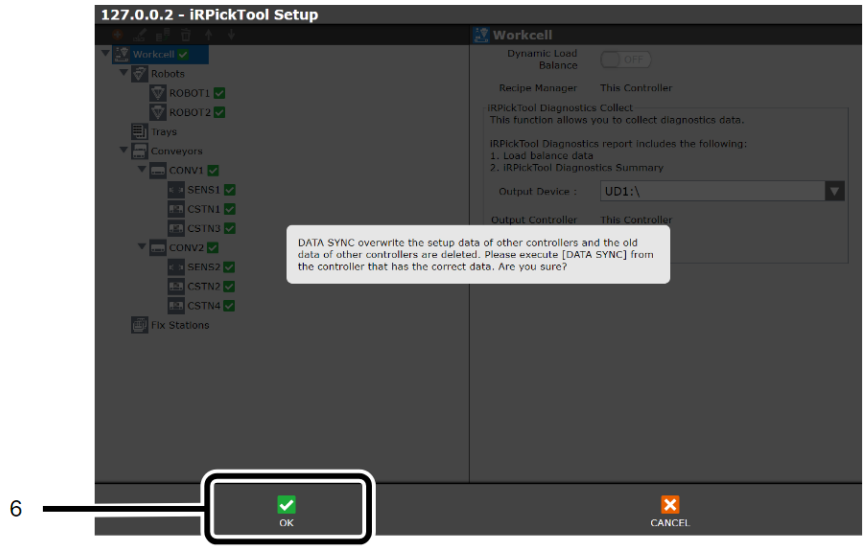
CAUTION

- Unless the specific differences between the non-synchronized robot controller and the currently open iRPickTool settings are clearly understood, verify the settings of all objects.
- Once you perform synchronization as instructed below based on unintended settings, you may be unable to restore the data to the original settings. Check carefully to make sure that it is appropriate to transfer the settings of the selected robot controller to the other robot controllers.

4 Check whether it is appropriate to transfer those settings to the other robot controllers and click [SYNC DATA].

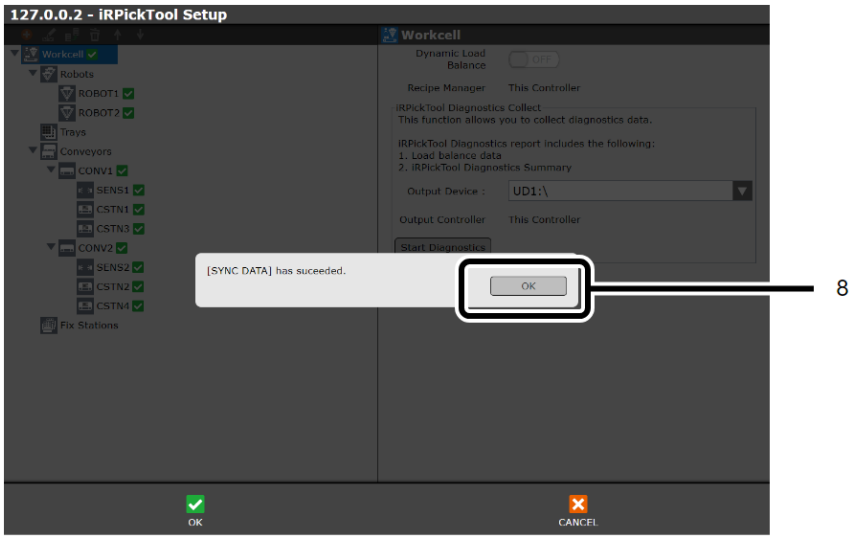
After a while, the following screen appears.

5 Click [OK].



6 When data synchronization is complete, the screen shown below is displayed.

7 Click [OK].



NOTE

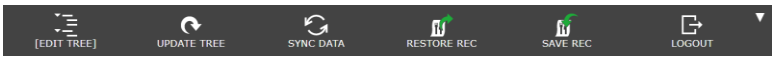
If data synchronization fails, such as when the RIPE-019 warning occurs on another robot controller, refer to "I.5.8.1.13. PKSETSYNCINT.PC(P.336)".

Refreshing the tree

Since the iRPickTool setup screen does not automatically reflect the latest status, you need to execute [UPDATE TREE] to apply the settings of the setup screen. 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 on the setup screen.

In these cases, click [UPDATE TREE] on the desired iRPickTool setup screen.



5.1. Setting up a workcell

The details of each item are as follows.

[Dynamic Load Balance]

If a robot goes offline on a conveyor for which load balancing is enabled, the parts to be handled by that robot are handled by other robots. If the robot goes online, it resumes the handling of the parts it is supposed to handle.

[Recipe Manager]

Recipe Manager is the controller in which the iRPickTool recipes are stored. A recipe is a summary of an entire workcell of iRPickTool. By using different recipes, you can replace an entire workcell. For details on the recipe, see “I.8.1. Exchange workcell with using recipe(P.545)”. All recipes will exist only in the “Recipe Manager” controller.

NOTE

Recipe Manager cannot be changed.

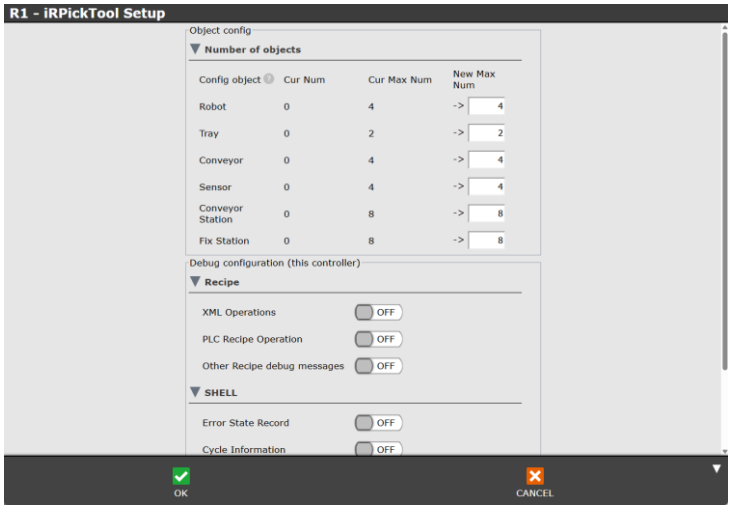
5.1.1. Setting the number of objects

On the workcell setup screen, you can modify the maximum number that can be set for each object.

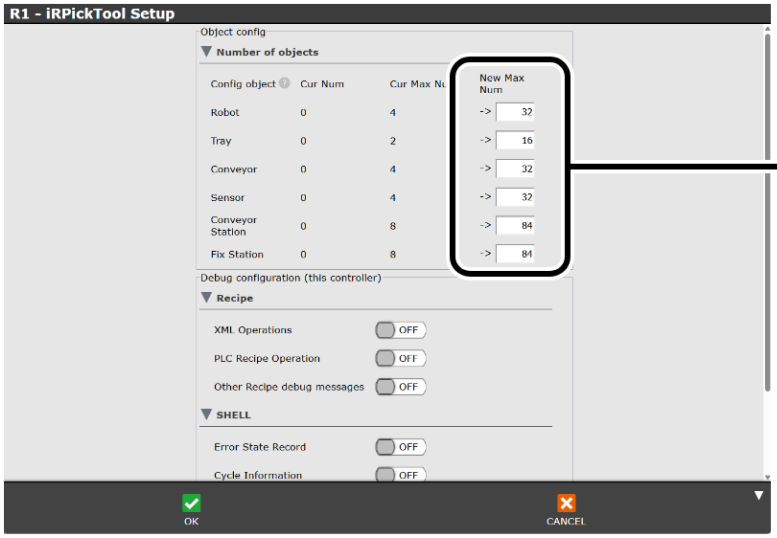
The procedure for changing the maximum number is as follows.

1 Press [OBJ CONFIG].

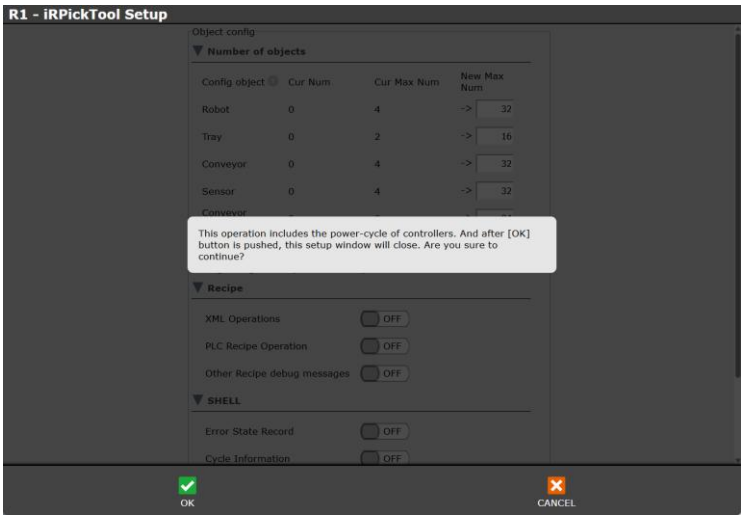
After a while, the following screen appears.



2 In [New Max Num] of the object that you want to increase, enter the new maximum number of objects after the increase.

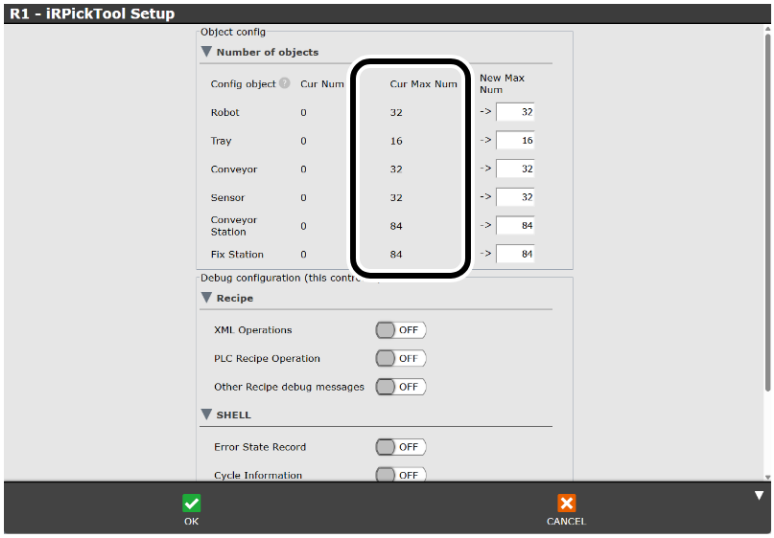


3 Click [OK]. The message shown below will be displayed.



4 Click [OK]. All the robot controllers for which RIPE is set will automatically restart.

If you press the [OBJ CONFIG] key again after the restart, the current maximum numbers of objects will have changed to the values you set, as shown below.



NOTE

1. Regardless of the maximum number of sensors set here, the maximum number of sensors that can be simultaneously operated by one robot controller is four.
   Regardless of the maximum number of conveyor stations set here, the maximum number of conveyor stations that can be added with one robot controller is eight. If you add nine or more conveyor stations, alarm INTP-310 “(PKSHCUST, number) Array index error” occurs when the power is turned on. In addition, functions such as Recipe Operation, Dynamic Load Balance, and check of whether a program is being executed, do not work.
2. The maximum number that can be modified for each object is shown in Table 5.1.1-1.

Table 5.1.1-1 Maximum number that can be modified for each object

Object	Maximum number
Robot	32
Tray	16
Sensor	32
Conveyor	32
Conveyor station	84
Fixed station	84

5.1.2. Recipe

A recipe is a summary of an entire workcell of iRPickTool.

By using different recipes, you can replace an entire workcell. For example, this is useful in cases such as:

- When changing from one product to another
- When running production for the same product using different grippers, trays, robots, sensors, conveyor stations and fixed stations

⚠CAUTION

When using the ROS Interface Packets Over Ethernet (RIPE), the software series/versions of all the connected controllers need to match.

When a recipe is used, one recipe (called the “active recipe”) is selected, and the workcell corresponding to its contents will be executed.

For details, see “I.8.1. Exchange workcell with using recipe(P.545)”.

5.2. Setting up a robot

The details of each item are as follows.

[Controller]

Select a robot controller that corresponds to the robot. If RIPE is not set, only “This Controller” is displayed.

[Group Number]

Select a group number that corresponds to the robot. If you did not order multiple groups for the robot controller selected in [Controller], only 1 is displayed.

⚠CAUTION

The group number specified here must be manually set in the [Group Mask] of the vision register to be used before operation. For details, see the description of the vision register details screen in "iRVision OPERATOR'S MANUAL (Reference)"(B-84434EN). The group mask is not overwritten even when the GET_OFFSET command is executed and the result is stored in the vision register.
The default configuration applies offset processing only to robots assigned to Group 1. Therefore, reconfiguration is typically not required.

[Dynamic Error Adjustment]

Set this item 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 “I.5.12. Fine adjustment of the tracking motion(P.386)”.

This means the error that is caused by tracking delay of the robot to the conveyor. Set values according to “I.5.12.3. Adjustment of the dynamic error of the robot(P.387)”. But this set value is enabled for a conveyor except the servo conveyor.

When using the servo conveyor, select [SYSTEM] → [Configuration] → [Tracking Function] → [Servo Conveyor Tracking] → [Dynamic Error Adjustment Value (ms)] from the menu and adjust the group number value of the servo conveyor for each servo conveyor.

[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 “I.8.4. Skipping the tracking motion outside the area(P.574)”.

[Circular Boundary]

If you want to set a circular boundary, enable this and set a radius of 0 to 5000.

The [Circular Boundary] item is displayed on the conveyor station setup screen only when circular boundary is enabled on the selected robot.

When circular boundary is enabled, you cannot use the pre-grouping function and sort function.

When the sort function is used at the conveyor station assigned to the selected robot, the circular boundary cannot be enabled.

5.3. Setting up 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.

NOTE

When using trays on a conveyor, all cells within the same tray are sent to the downstream conveyor station when the most upstream cell crosses the Discard Line. Therefore, it is not possible to perform operations simultaneously at multiple conveyor stations on a long tray spanning several stations.

The details of each item are as follows.

[Reference cell]

The offset for each cell is calculated from X, Y, Z and R for each cell and X, Y, Z and R for the reference cell.

If you press [SET REF CELL], the values for cell 1 will be set for the reference cell.

In the case of a conveyor that uses a tray, if you set the reference position on the sensor setup screen, the values for cell 1 will be set for the reference cell.

In the case of a fixed station, press [SET REF CELL] and set the reference cell, then teach the robot position for cell 1.

In the case of a conveyor that uses a tray, set the reference position on the sensor setup screen, then teach the robot position for cell 1. In the case of a conveyor that uses a tray, you can omit the step to set the reference cell on the tray setup screen.

If you want to make fine adjustments to the robot position for a specific cell after teaching the robot position for cell 1, change the coordinates for each cell.

After changing the reference cell, you need to set the reference position of the sensor and teach the robot position again.

[Layer number]

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

NOTE

1. When trays are handled by the conveyor, the order in which you add cells is not relevant to the order in which parts are put on (or taken from) trays during execution. Parts are handled sequentially from the cell on the downstream side as appropriate for the situation. To control the order intentionally, use “Layer” detailed in “I.5.3.2. Layer operation(P.273)”.
2. Up to 160 cells can be added per tray.

[Edit Cell 1] to [Edit Cell 160]

Edit cells. In accordance with the cell that is selected, the item name changes from [Edit Cell 1] to [Edit Cell 160].

[X(mm)], [Y(mm)], [Z(mm)] and [R(deg)]

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

The procedure to set coordinates is explained below, using the case where a tray is used to handle four parts.

First, decide on the position of the tray frame.

For example, suppose that you have decided to place it in a corner of the top of a box, as shown in Fig. 5.3-1.

NOTE

When trays are used on the conveyor, the posture of the tray frame is determined as follows.
The Z-axis direction of the tray frame is the same as that of the tracking frame. Also, the XY plane of the tray frame is the same as that of the tracking frame. Therefore, if you decide on the X-axis direction of the tray frame, the directions of all the other axes are automatically determined.

While you can define the tray frame freely on the tray, select a position that is easy to touch up. When the tray frame on the conveyor is taught to the robot, the origin of the tray frame and a point in the X-axis direction need to be touched up with the TCP of the robot.

Also, set the tray frame on the fixed station as the user frame.

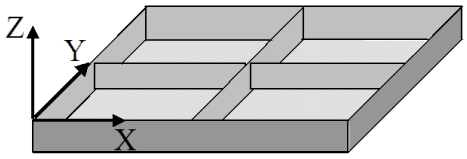


Fig. 5.3-1 Position of the tray frame

Add cells to a tray, and enter the coordinates of each cell.

For example, if the tray in Fig. 5.3-1 is viewed from above, it will look like the figure below.

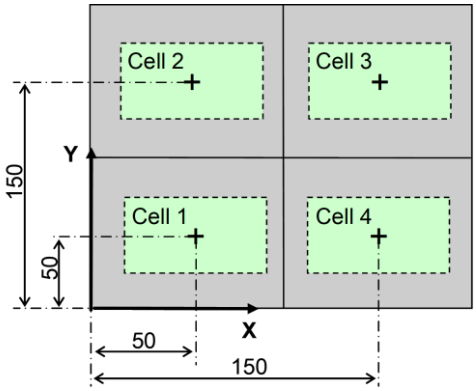
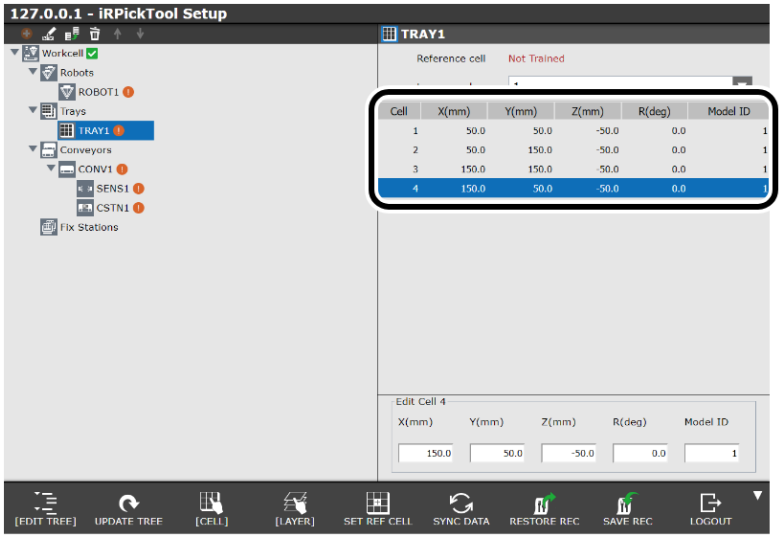


Fig. 5.3-2 Position setting example of the tray frame

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



Setting up a tray when parts are different in height

When parts to be handled are different in height, use the Z value of the cell.

For example, consider a case where there are two cells and the part to be placed in each of them has a different height, as shown in the figure below.

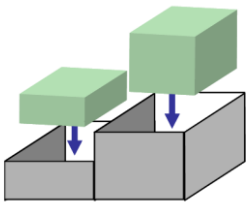
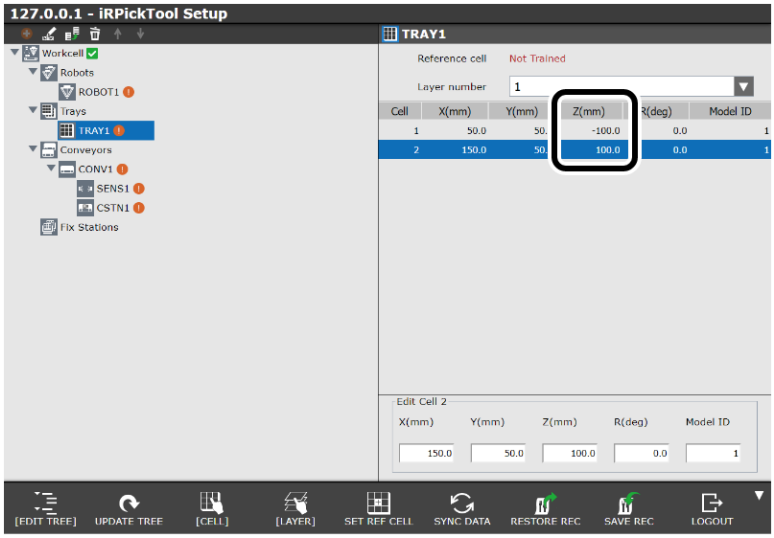


Fig. 5.3-3 Example of when parts have different heights

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



[Model ID]

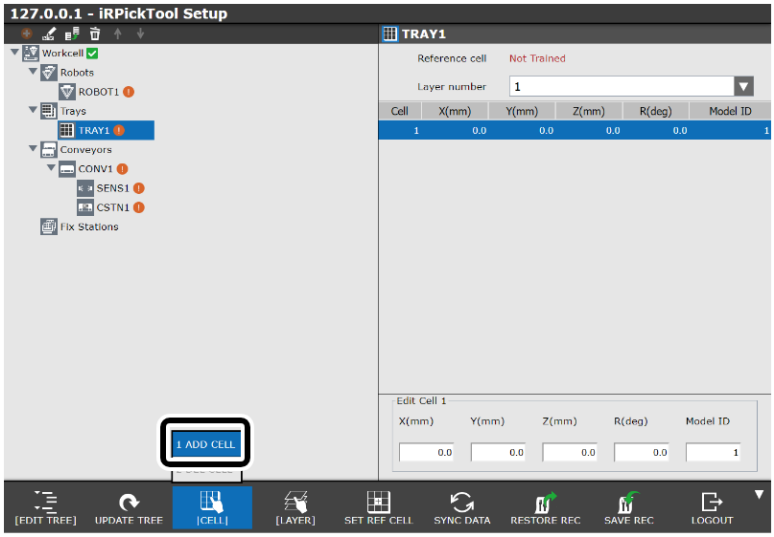
A model ID can be set for each cell. Model IDs are used in such cases as when you want the robot TP program to perform a different type of processing according to which cell is allocated.

There is no need to change the default value of 1 unless any other number is particularly required.

Examples of using model IDs are given in "I.7. Setting example appropriate to the usage(P.522)".

5.3.1. Cell operation

To add a cell to a tray, select [1. ADD CELL]. To delete a highlighted cell, select [2. DEL CELL] from the menu displayed by clicking [CELL].



NOTE

1. When trays are handled by the conveyor, the order in which you add cells is not relevant to the order in which parts are put on (or taken from) trays during execution. Parts are handled sequentially from the cell on the downstream side as appropriate for the situation. To control the order intentionally, use "Layer" described later.
2. Up to 160 cells can be added per tray.

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

When a robot hand that can hold multiple parts is used, the robot can handle more than one part in a single motion.

For example, consider a tray that can contain 12 parts with a robot hand capable of holding three parts, as shown in Fig. 5.3.1-1.

Each dotted-line circle represents a part.

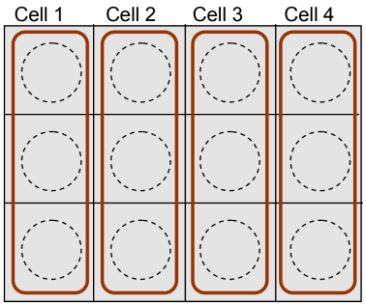
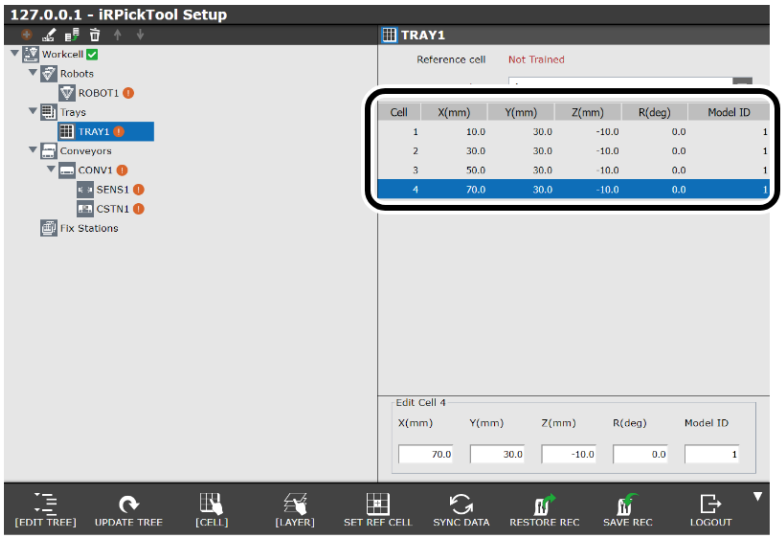


Fig. 5.3.1-1 Example of a tray for a hand that holds three parts

The robot handles three parts at a time. In such a case, three parts that can be handled at a time are considered to be a cell.

Therefore, the tray has four cells.

For example, the tray may be set as follows.



5.3.2. Layer operation

When parts are laid in layers in the tray, define each layer individually.

To add a layer, select [1. ADD LAYER].

To delete a layer, select the [Layer number] of the layer to be deleted and 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 the order in which parts are placed in the tray needs to be controlled - e.g., when the tray has multiple layers and a part cannot be placed in an upper layer unless the placement of parts in the lower layer is completed. When placing parts in a tray, set [Initial Status] to [Empty] for the conveyor and fixed station, as described later. This prevents a part from being placed in an upper-layer cell having a larger layer number when any lower-layer cell having a smaller layer number is empty.

The figure below shows an example of using layers. Parts are piled in three layers in a box. In this case, the tray has three cells. Priorities are given as follows. A part cannot be placed in the second or third layer unless a part is placed in the first layer. Likewise, a part cannot be placed in the third layer unless a part is placed in the second layer.

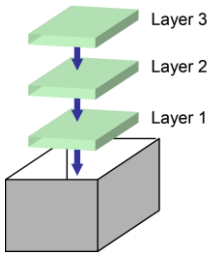
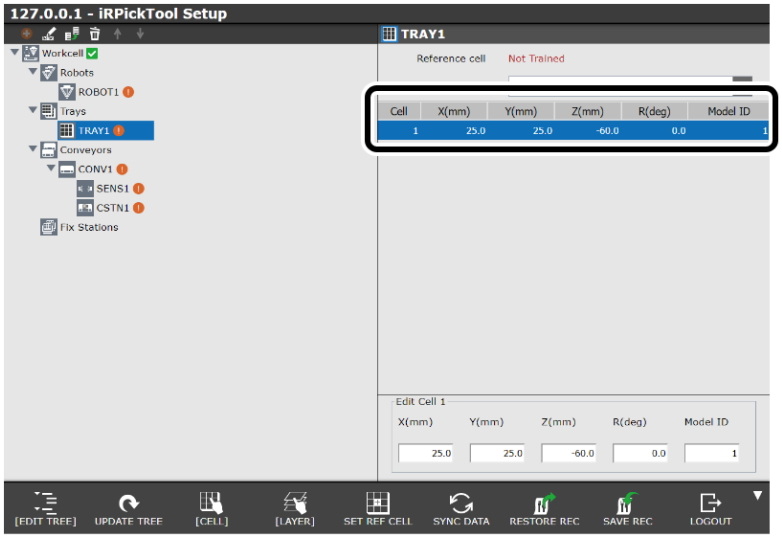


Fig. 5.3.2-1 Example of when parts are stacked in multiple layers in a tray

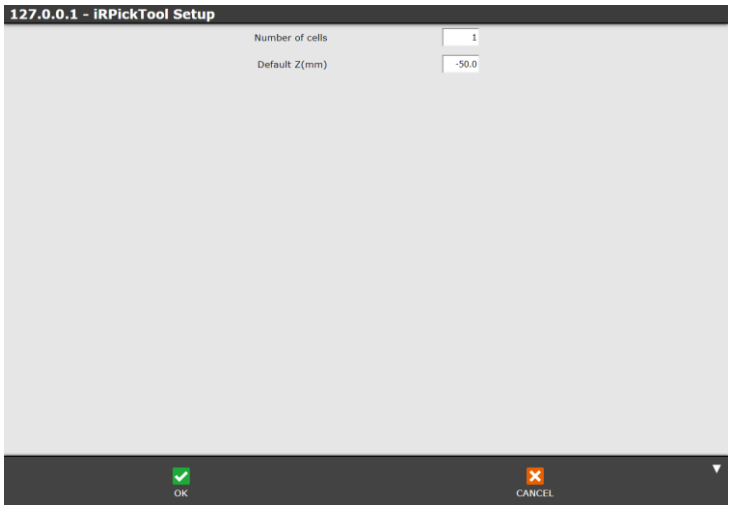
In this case, set the tray as instructed below.

1 First, set cell 1 in the first layer as follows.

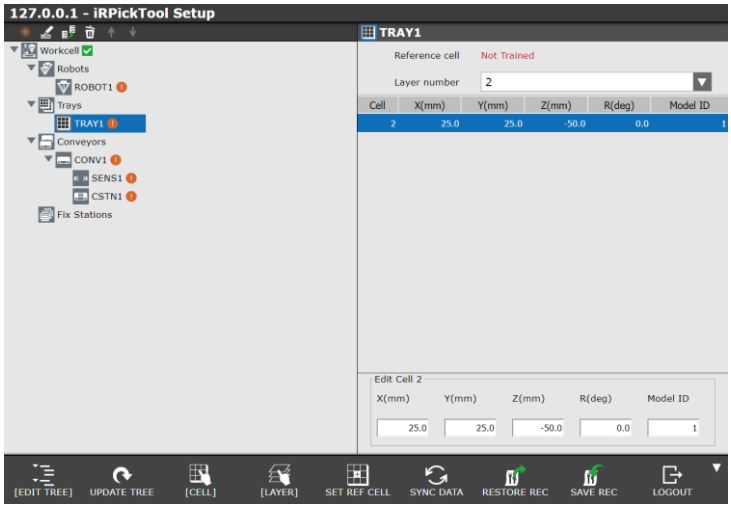


2 Next, to add the second layer, select [1. ADD LAYER] from [LAYER] key.

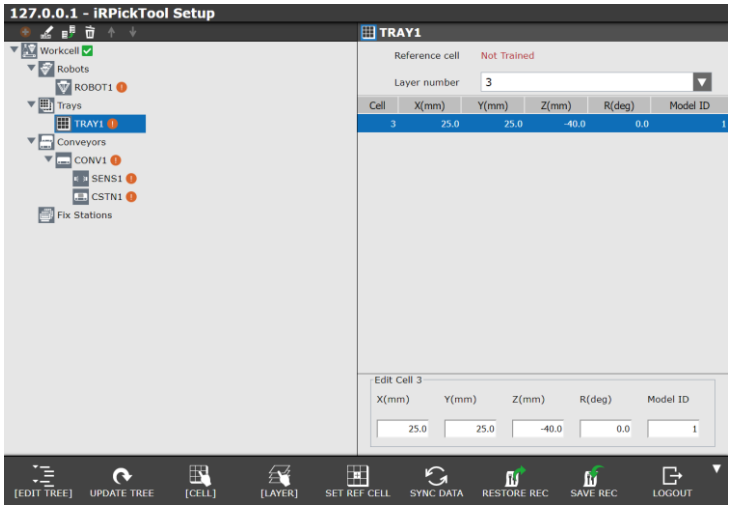
When the screen shown below is displayed, specify 1 as the number of cells for the second layer and –50 mm as the value of Z. Then, click [OK].



3 Layer 2 you have added is selected. Enter 25.0 mm as the values of X and Y for cell 2, as with cell 1.



4 Likewise, add layer 3 with the value of Z specified, and then set the values of X and Y for cell 3.



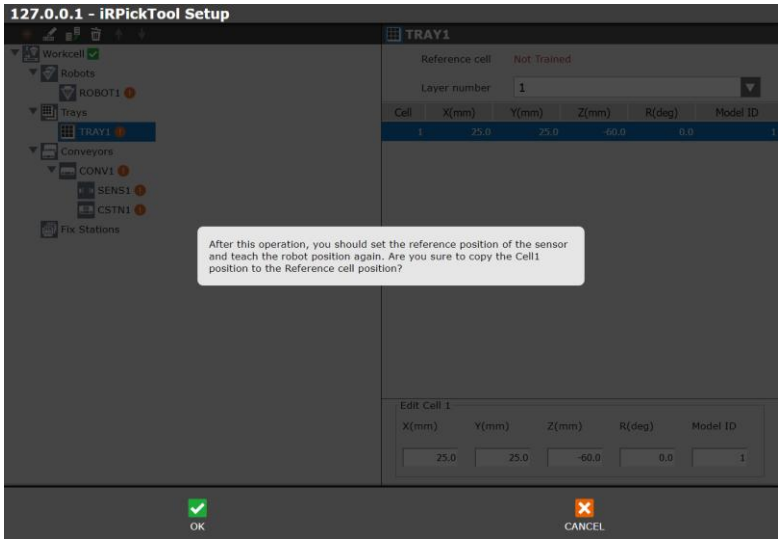
As described above, set cell 1 as layer 1 (bottom layer), cell 2 as layer 2 (middle layer), and cell 3 as layer 3 (top layer). The values of X and Y of each cell are the same, and a cell in a higher layer has a greater value of Z.

In addition, set [Initial Status] to [Empty] for the conveyor and fixed station that use this tray, as described later. If this item is set as mentioned here, the data of cell 2 or 3 is not allocated to the robot unless a part is placed in cell 1 having the smallest layer number, thus preventing a part from being placed in these locations.

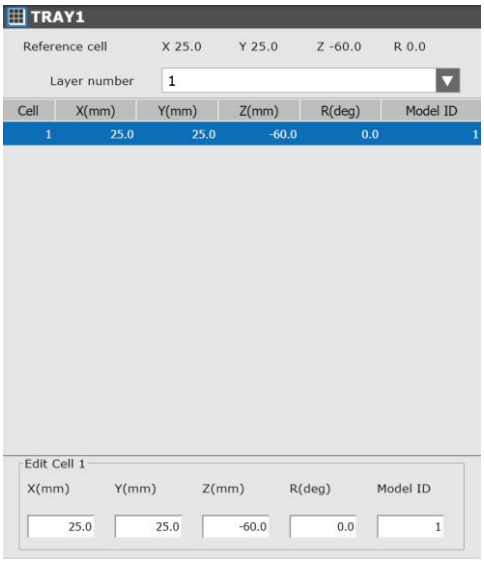
5.3.3. Setting the reference cell

Set a reference cell. Carry out setting of a reference cell immediately before “I.5.11. Teaching the position to robots(P.385)”.

If you press [SET REF CELL], a message that prompts you to set the sensor reference position and teach the robot position again will appear, as shown below.



If you click [OK], 'Not Trained (red letters)' disappears from [Reference cell] and the X, Y, Z and R values that you have set in [Edit Cell 1] will be displayed.



5.4. Setting up 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 "I.5.5. Setting up a sensor(P.288)" and "I.5.6. Setting up a conveyor station(P.318)".

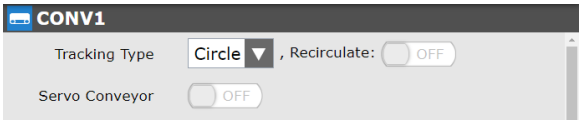
[Tracking Type]

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

[Recirculate]

If you set the [Recirculate] toggle switch, which is displayed only when [Arc] is selected for [Tracking Type], to [Enable], a tray for which the process is not yet complete will be taken over from the most downstream station by the most upstream station.

For example, in cases where a tray on a circular conveyor is not completely filled, you can recirculate it and fill the tray.



The conditions in which you can set [Recirculate] are as follows:

- “Arc” has been selected for [Tracking Type]
- [Tray] is being used
- Two or more conveyor stations have been added to the child object

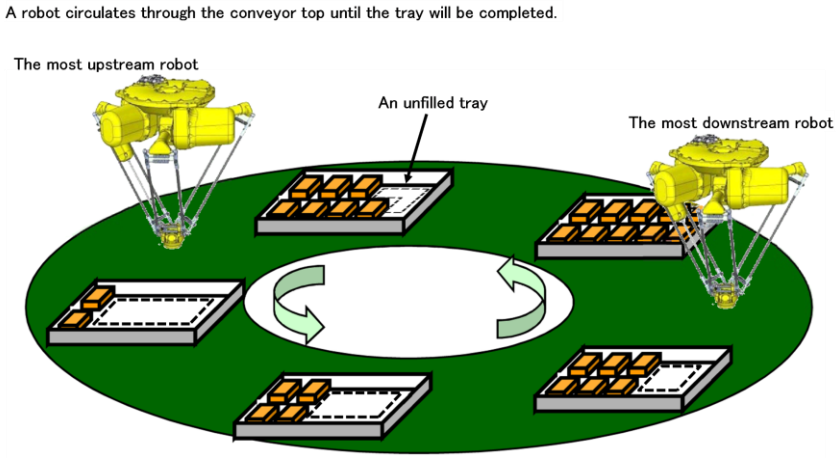


Fig. 5.4-1 Image of recirculation of an unfilled tray

[Servo Conveyor]

You can only set this item when the [Tracking Type] is set to “Line”. Set this item to [ON] if the conveyor is a servo driven conveyor. If you change it, child objects like the sensor and the conveyor station will be initialized.

To enable a servo conveyor, it needs to be set up in advance.

[Buffer]

This setting item specifies that the conveyor is a buffer conveyor.

For details, see “I.8.15. Using a conveyor as a buffer(P.678)”.

[Forward Direction Conveyor]

This item specifies the forward direction buffer conveyor.

This item is displayed when “Backward Direction” is selected for [Buffer].

For details, see “I.8.15. Using a conveyor as a buffer(P.678)”.

⚠CAUTION

When a conveyor is used as a forward or backward direction buffer conveyor, the following functions are unavailable.

- Sensor trigger conditions and actions other than DI, RI, distance, and put part in queue
- Duplicate check mode (only [Leave first] and [Not used] for both [Model ID] and [Measurement] are supported)
- Tray redetection
- Load balance (available when set to “Disable”)
- Pre-grouping
- Circular and skew tracking boundaries
- Sort (Y, measurement value)
- Stop/start conveyor

[Tray]

When using trays on the conveyor, select the tray to be used. When not using trays, select [Not Used].

[Initial Status]

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

[Duplicate Tolerance]

If the vision system detects more than one identical part, this tolerance is used to determine whether the detected parts are the same. Normally, when the same part is detected more than once, no current part position completely matches others due to detection error or some other factor. Set a tolerance so that the vision system retains only one of the multiple detected parts.

If the detected parts are identified as being the same, the vision system retains only one of the parts, in accordance with the setting in [Duplicate Check Mode]. Normally, set this value to 10 mm or so.

[Duplicate Check Mode] box

Specify which result is to be retained when the duplicate check identifies detected parts as being the same.

⚠CAUTION

When using trays, only the following settings are supported.

[Model ID]:	[Not used]
[Measurement]:	[Not used]
[Leave First/Last]:	[Leave first]

In addition, Duplicate Check is performed on a tray-by-tray basis.

[Model ID]

- Select from [Not used], [Leave max.], [Leave min.], [Overwrite only model ID by max. value], and [Overwrite only model ID by min. value].

[Measurement]

- When [Model ID] is set to [Not used], select from among [Not used], [Leave max.], and [Leave min.].
- When [Model ID] is [Leave max.] or [Leave min.], select from among [Not used], [Leave max. in identical model ID], and [Leave min. in identical model ID].
- When the [Model ID] is [Overwrite only model ID by max. value] or [Overwrite only model ID by min. value], no selection can be made. This item is automatically set to [Not used].
- When a setting other than [Not used] is selected, select which measurement value to use from [Value 1] to [Value 10].

[Leave First/Last]

- When [Model ID] is [Not used] and [Measurement] is [Not used], select from among [Leave first] and [Leave last].
- When [Model ID] is [Not used] and [Measurement] is [Leave max.] or [Leave min.], select from among [Leave first in identical meas.] and [Leave last in identical meas.].
- When [Model ID] is [Leave max.] or [Leave min.] and [Measurement] is [Not used], select from among [Leave first in identical model ID] and [Leave last in identical model ID].
- When [Model ID] is [Leave max.] or [Leave min.] and [Measurement] is [Leave max. in identical model ID] or [Leave min. in identical model ID], select from among [Leave first in identical model ID and meas.] and [Leave last in identical model ID and meas.].

- When the [Model ID] is [Overwrite only model ID by max. value] or [Overwrite only model ID by min. value], no selection can be made. This item is automatically set to [Leave first (other than model ID)].

The following list describes the behavior of each setting:

Table 5.4-1 Behavior when Model ID is "Not used"

Measurement	Leave First/Last	Behavior
Not used	Leave first	The first result is retained.
Not used	Leave last	The last result is retained.
Leave max.	Leave first if same meas.	The result for which the specified measurement value is largest is retained. If all the measurement values are the same, the first result is retained.
Leave max.	Leave last if same meas.	The result for which the specified measurement value is largest is retained. If all the measurement values are the same, the last result is retained.
Leave min.	Leave first if same meas.	The result for which the specified measurement value is the smallest is retained. If all the measurement values are the same, the first result is retained.
Leave min.	Leave last if same meas.	The result for which the specified measurement value is the smallest is retained. If all the measurement values are the same, the last result is retained.

Table 5.4-2 Behavior when Model ID is "Leave max."

Measurement	Leave First/Last	Behavior
Not used	Leave first if same model ID	The result with the larger model ID is retained. If all the model IDs are the same, the first result is retained.
Not used	Leave last if same model ID	The result with the larger model ID is retained. If all the model IDs are the same, the last result is retained.
Leave max. in identical model ID	Leave first in identical model ID and meas.	The result with the larger model ID is retained. If all the model IDs are the same, the result for which the specified measurement value is largest is retained. If all the model IDs and measurement values are the same, the first result is retained.
Leave max. in identical model ID	Leave last in identical model ID and meas.	The result with the larger model ID is retained. If all the model IDs are the same, the result for which the specified measurement value is largest is retained. If all the model IDs and measurement values are the same, the last result is retained.
Leave min. in identical model ID	Leave first in identical model ID and meas.	The result with the larger model ID is retained. If all the model IDs are the same, the result for which the specified measurement value is the smallest is retained. If all the model IDs and measurement values are the same, the first result is retained.
Leave min. in identical model ID	Leave last in identical model ID and meas.	The result with the larger model ID is retained. If all the model IDs are the same, the result for which the specified measurement value is the smallest is retained. If all the model IDs and measurement values are the same, the last result is retained.

Table 5.4-3 Behavior when Model ID is "Leave min."

Measurement	Leave First/Last	Behavior
Not used	Leave first if same model ID	The result with the smaller model ID is retained. If all the model IDs are the same, the first result is retained.
Not used	Leave last if same model ID	The result with the smaller model ID is retained. If all the model IDs are the same, the last result is retained.
Leave max. in identical model ID	Leave first in identical model ID and meas.	The result with the smaller model ID is retained. If all the model IDs are the same, the result for which the specified measurement value is largest is retained. If all the model IDs and measurement values are the same, the first result is retained.
Leave max. in identical model ID	Leave last in identical model ID and meas.	The result with the smaller model ID is retained. If all the model IDs are the same, the result for which the specified measurement value is

Measurement	Leave First/Last	Behavior
		largest is retained. If all the model IDs and measurement values are the same, the last result is retained.
Leave min. in identical model ID	Leave first in identical model ID and meas.	The result with the smaller model ID is retained. If all the model IDs are the same, the result for which the specified measurement value is the smallest is retained. If all the model IDs and measurement values are the same, the first result is retained.
Leave min. in identical model ID	Leave last in identical model ID and meas.	The result with the smaller model ID is retained. If all the model IDs are the same, the result for which the specified measurement value is the smallest is retained. If all the model IDs and measurement values are the same, the last result is retained.

Table 5.4-4 Behavior when Model ID is "Overwrite only model ID by max. value"

Measurement	Leave First/Last	Behavior
Not used	Leave first (other than model ID)	The first result is retained. However, the model IDs are overwritten with the largest result for the same part.

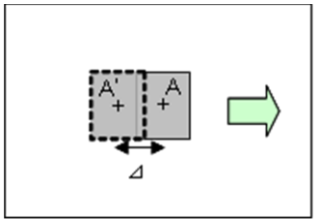
Table 5.4-5 Behavior when Model ID is "Overwrite only model ID by min. value"

Measurement	Leave First/Last	Behavior
Not used	Leave first (other than model ID)	The first result is retained. However, the model IDs are overwritten with the smallest result for the same part.

When the Duplicate Tolerance is not exceeded

In the default settings, [Not used] is selected for [Model ID], [Not used] is selected for [Measurement], and [Leave first] is selected for [Leave First/Last].

When ⊿ is less than or equal to the Duplicate Tolerance, only the (n–1)th detected A' workpiece is retained by default.



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

A : Current position of the nth workpeice detected

⊿ : Distance between A and A'

Fig. 5.4-2 Example of a case when the Duplicate Tolerance is not exceeded

⚠CAUTION

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

NOTE

In 7DF1/14 or older software, [Duplicate Tolerance] is displayed as [Overlap Tolerance]. In addition, [Duplicate Check Mode] is not supported. In the case of 7DF1/14 or older software, the behavior of the vision system is the same as that of the default settings in which [Not used] is selected for [Model ID], [Not used] is selected for [Measurement], and [Leave first] is selected for [Leave First/Last].

[Tray Redetection]

Configure the settings related to tray redetection.

For details on each setting item, refer to “I.8.12. Tray redetection(P.653)”.

[Tray Redetection]

Set whether or not to use tray redetection.

NOTE

When this function is enabled, the “Duplicate Check” function cannot be used.

Tray Redetection can only be enabled when the following conditions are met:

- [Tracking Type] is Line.
- Servo Conveyor is not used.
- Trays are used.

[Interval Tolerance]

Enter the maximum possible distance the tray can be displaced.

⚠CAUTION

For [Interval Tolerance], enter a value smaller than the spacing between trays. If trays arrive at intervals shorter than the Interval Tolerance, the downstream trays will be removed at the time of tray detection.

[Load Balance]

Specify how parts are to be allocated on the conveyor.

[Disable]

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]

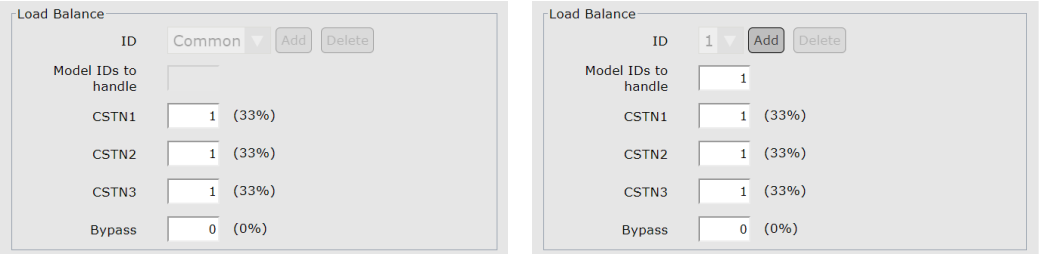
Up to eight sets of balance data can be specified. Which balance to use depends on the model ID of the part handled.

NOTE

- Some practical examples of specifying load balancing based on model IDs are given in “I.7. Setting example appropriate to the usage(P.522)”.
- Load balancing based on model IDs cannot be used in combination with the Sort function.

[Load Balance] box

The [Load Balance] box refers to the following parts in [System Setup].



When load balancing is not disabled, specify the ratio of parts to be handled by the robot of each conveyor station while changing the model ID.

If you specify [For all model IDs], the screen is as shown on the left. For example, suppose that the conveyor has three conveyor stations and that you want parts to be allocated evenly. In this case, specify 1 for each conveyor station, and parts will be allocated at a ratio of 1:1:1. 1:1:1 is exactly the same as a ratio of 2:2:2.

If you specify [For each model ID], the screen is as shown on the right. Set a ratio for each model ID while selecting a model ID individually. In [Model IDs to handle], specify a model ID. To specify the same balance for two or more model IDs, you can add model IDs by clicking the [Add] button. Up to eight model IDs can be specified. An unnecessary model ID can be deleted by clicking the [Delete] button.

If you set a value other than 0 in [Bypass], as many parts as the set value are intentionally bypassed 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]

If you enable the pre-grouping function which arrange parts on the same conveyor, you set items in this box.

For details, see “I.5.15. Setup of pre-grouping(P.393)”.

[Pre-grouping]

Set to [ON] when using the pre-grouping function that arranges parts on the same conveyor. Checking this item displays detailed setting items. For details on each item, see “I.5.15. Setup of pre-grouping(P.393)”.

[Log] box

When the toggle is set to [ON], each corresponding log is recorded. From the moment the toggle is set to [ON], corresponding information is recorded in the temporary memory. The information can be output to the output device specified at any timing even during execution.

[Part Event]

The maximum number of logged parts is 1000. If it goes over 1000, the data is deleted from the oldest one. This log can be utilized in order to play back the part flow in ROBOGUIDE. The playback function can be used in V8 Rev.L or later of ROBOGUIDE.

[Encoder Count]

The encoder count of all the conveyor stations set to this conveyor is recorded in a log at intervals of 16 milliseconds or less. The recording period can be adjusted between 10 to 60 seconds. When the log becomes full, the oldest data is deleted. This log can be used to detect whether the encoder has slipped or for other purposes.

[Output Device]

In this device, the recorded logs are output. You can select “UD1” or “MC”. When you press the [Output] button, all the currently recorded logs are output. You can press the [Output] button even during robot operation.

[Output Controller]

The robot controller where the logs are output is displayed. Logs are output to the controller of the robot that is online and set as the most upstream of this conveyor.

5.4.1. Setting up a tracking frame

Set a tracking frame and encoder scale.

You can set a tracking frame and encoder scale by using the tracking schedule setting screen of the teach pendant of the robot. However, the procedure described below is recommended particularly if multiple robots are used for tracking, because a tracking frame common to all the robots needs to be set.

⚠CAUTION

If you set a tracking frame for a 3-, 4- or 5-axis robot by using the tracking schedule setting screen of the teach pendant, a value other than 0 may be set as the value of W or P of the tracking frame. By contrast, if you set a tracking frame by using the procedure described below, 0 is always set as the values of W and P of the tracking frame for a 3-, 4- or 5-axis robot. Since a 3-, 4- or 5-axis robot can only move with its flange face facing downward (or upward) during tracking, set a tracking frame according to the procedure described below.
On the other hand, this procedure does not support tracking where the value of Z of the world frame changes for a 3-, 4- or 5-axis robot. If such tracking motion is involved, you need to use the tracking schedule setting screen of the teach pendant to set a tracking frame. Note that, even in this case, the flange face needs to keep facing downward (or upward) during tracking and it is therefore necessary to fine-adjust the frame by directly entering the frame components.

When setting a tracking frame, have a fixture carried on the conveyor. Any fixture may be used if it has a part that the robot can touch up accurately with the TCP. If you have purchased a calibration grid from FANUC, a dot on the calibration grid can be used as well.

The following description assumes the use of a calibration grid.

⚠CAUTION

1. If you change the tracking frame, you need to perform conveyor station setup, camera calibration, reference position setting, and robot position teaching again.
2. When there is more than one robot, you cannot set a tracking frame unless the encoder count values of all the controllers on the conveyor are identical. If they are not identical, turn the power of all the robot controllers off then on again.
3. When there is more than one robot, make sure that the fixture does not move on the conveyor belt. If the fixture moves because it comes into contact with the robot or other object while it is carried on the conveyor, you need to redo the tracking frame setting from the beginning.
4. The tracking frame affects the ultimate handling accuracy. Touch up the fixture using an accurately set TCP.
5. You cannot set a tracking frame correctly if the robot wrist posture (W, P and R) changes during the setting process. Jog the robot only in the X, Y and Z directions so that the robot wrist posture does not change.

The setting procedure is different depending on tracking type.

For the settings of the line conveyor, see “I.2. iRPickTool Basic (basic functions) startup procedures(P.26)”. If there are multiple robots on one conveyor, setup of the second robot’s tracking frame will start when the setup for the first robot’s tracking frame is finished.

If you want to redo the tracking frame setup for only a specific conveyor station for a conveyor for which the tracking frame has been set up, refer to “I.5.6.1. Adding a tracking frame(P.326)”.

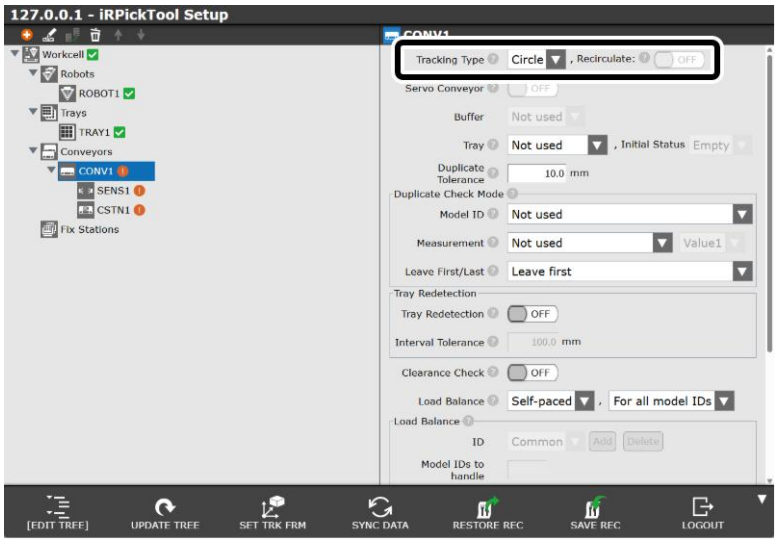
5.4.1.1. In the case of a circular conveyor

When the tracking type is “Circle”, the procedure is as follows.

Setup of circular tracking

The procedure to set up iRPickTool for when using circular tracking is explained below. Be careful of the following points as remarks for when using circular tracking.

Be sure to select [Arc] from the [Tracking Type] menu before proceeding to “Setup of a circular conveyor” given later.



Z-axis direction of the tracking frame

In circular tracking, the Z-axis direction of the tracking frame may be downward from the conveyor surface depending on the rotation direction of the conveyor. Specifically, the Z-axis direction of the tracking frame is determined as follows.

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

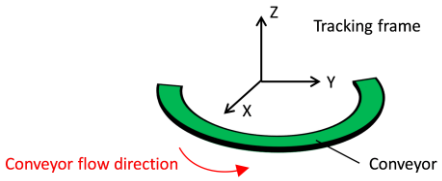


Fig. 5.4.1.1-1 Conveyor: Counterclockwise, Z-axis: Upward

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

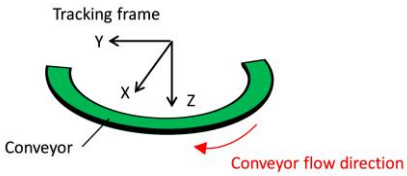


Fig. 5.4.1.1-2 Conveyor: Clockwise, Z-axis: Downward

The Z-axis direction of the tray frame is the same as that of the tracking frame. Check the Z-axis direction of the tracking frame before setting X, Y, Z and R of a tray. Refer to “I.5.3. Setting up a tray(P.269)”.

In addition, check the Z-axis direction of the tracking frame before setting [Part Z Height] in [Reference Data]. See “I.2.3.7.2. Vision process(P.99)”.

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

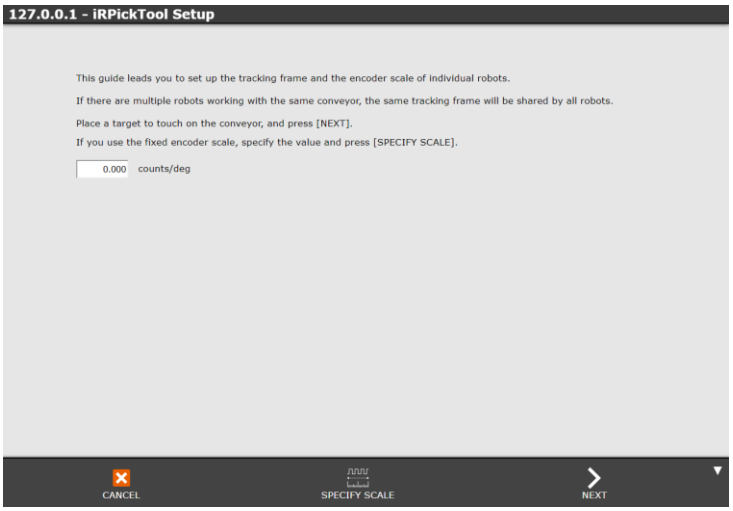
Setup of a circular conveyor

1 **Add as many conveyor stations as the number of robots that will operate on the conveyor. On the setup screen for each conveyor station, select a robot, tracking schedule number, and encoder number.**

For details, see "I.5.6. Setting up a conveyor station(P.318)".

2 **On the conveyor setup screen, click [SET TRK FRM].**

After a while, the following screen appears.



CAUTION

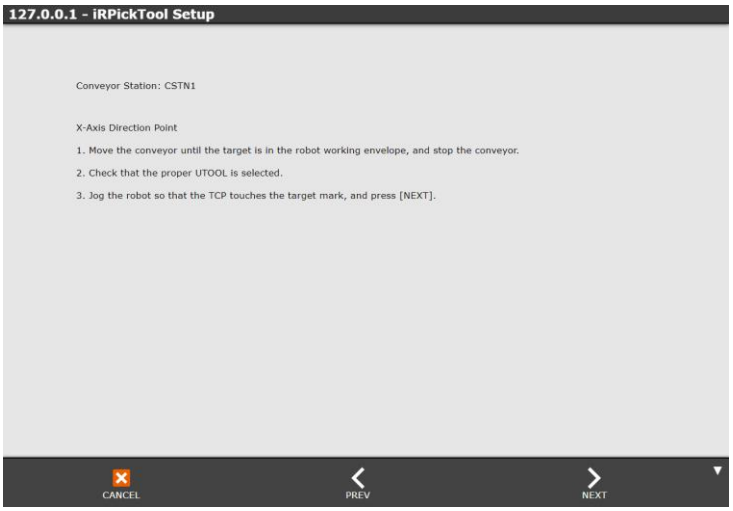
When using the αiA4000S, "IRPK-063 Encoder count does not match on all robots" may occur even when the conveyor is stationary, due to high-resolution encoder pulses. In this case, stop the conveyor and retry the operation several times.

3 **Place a fixture that can be touched up with the TCP of the robot at the most upstream area of the conveyor, and click [NEXT].**

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 click [SPECIFY SCALE].

In either case, the screen 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 **Select the user tool frame that you set in "I.4.4. Setting up the tool frame(P.257)" to be the jog feed frame.**

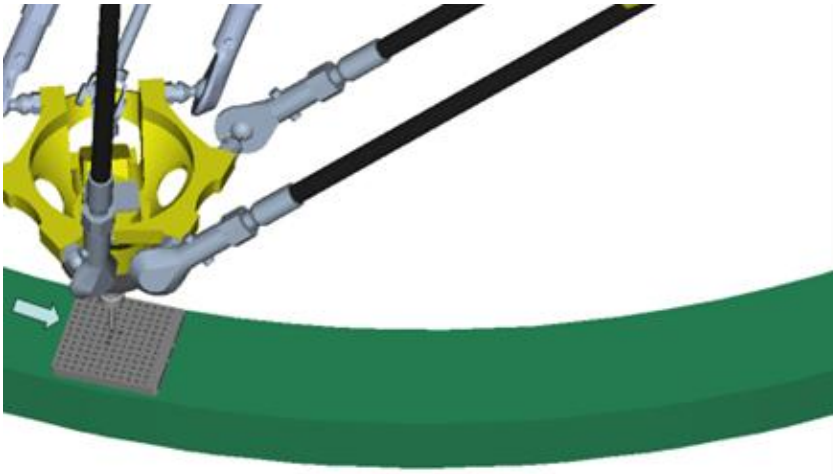
CAUTION

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

5 **Move the conveyor until the fixture reaches as far upstream as possible within the robot operation range and then stop it.**

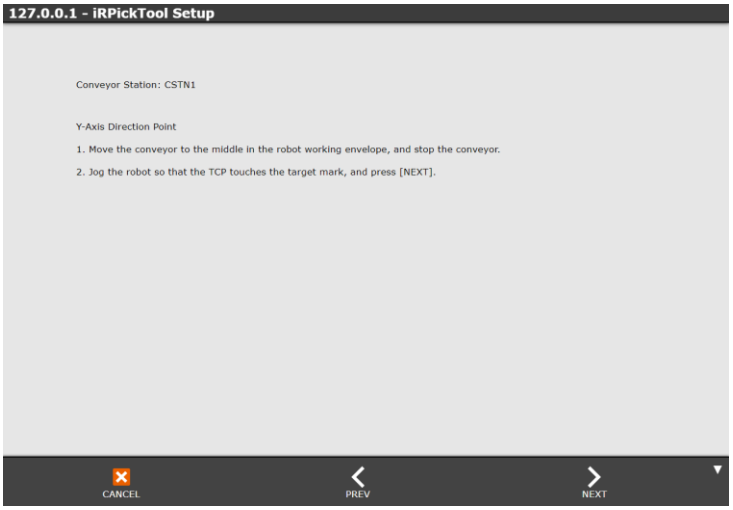
6 **Jog the robot to touch up the fixture with the TCP.**

Here, the dot at the intersection of the large dots in an L shape on the calibration plate is touched up.



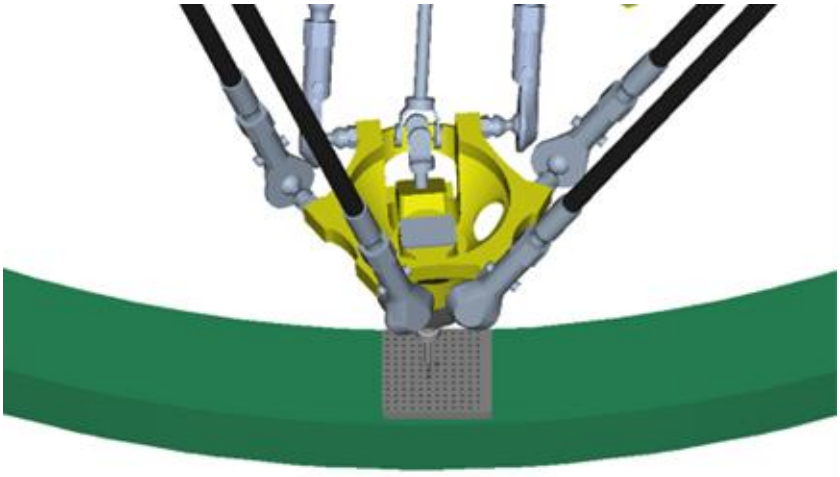
7 **With the fixture touched up, click [NEXT].**

After a while, the following screen appears.



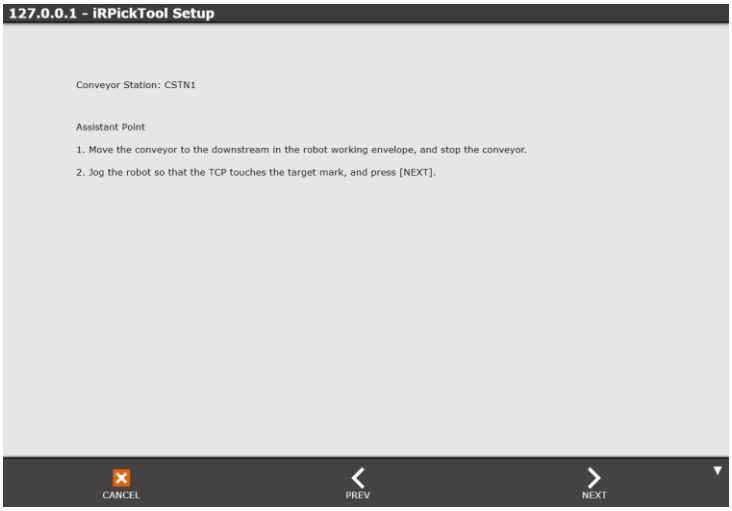
8 **After moving the robot to a position where the TCP does not interfere when the conveyor is moved, move the conveyor so that the fixture comes to the middle of the robot operation range and then stop it.**

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



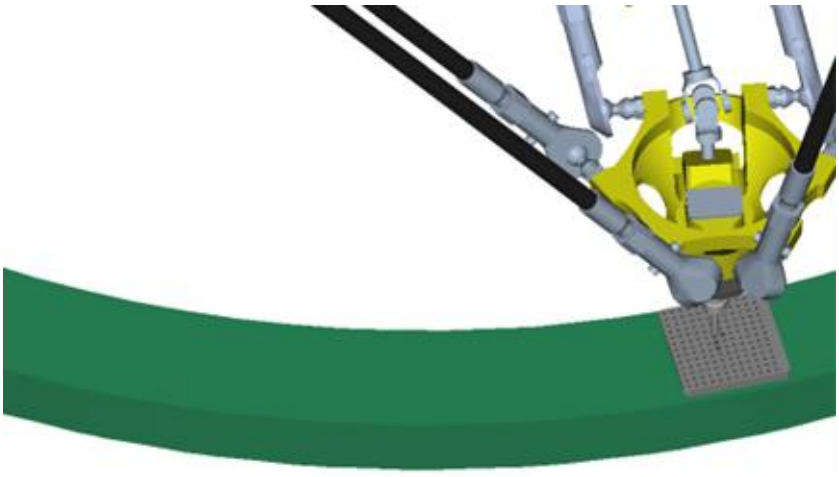
10 **With the fixture touched up, click [NEXT].**

After a while, the following screen appears.



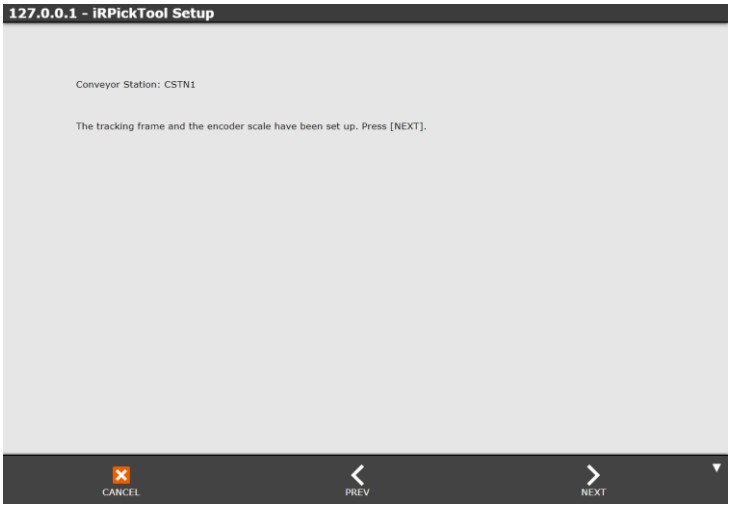
11 **After moving the robot to a position where the TCP does not interfere when the conveyor is moved, move the conveyor so that the fixture comes as far downstream as possible within the robot operation range and then stop it.**

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



13 **With the fixture touched up, click [NEXT].**

After a while, the following screen appears.

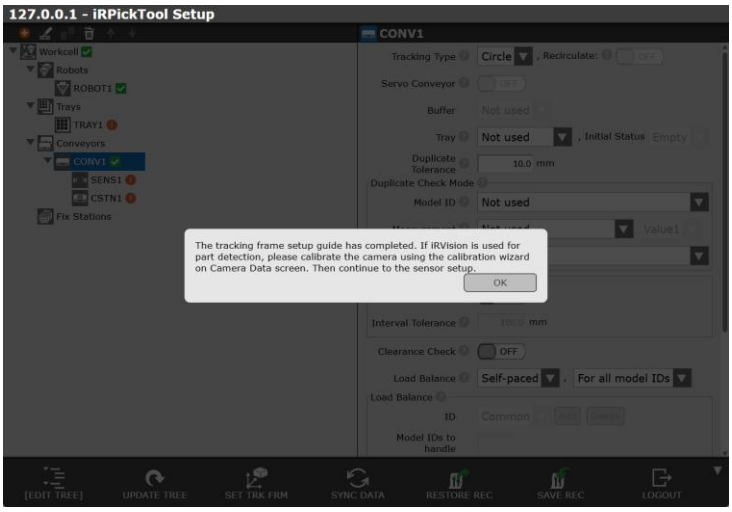


14 This completes the tracking frame setting procedure for the first robot, and a tracking frame and scale have now been set.

If you have more than one robot for the same conveyor, you can start the tracking frame setting procedure for the second robot by clicking [NEXT].

15 Repeat steps 4 through 12 for each robot.

When you have completed the procedure for all robots, the screen shown below is displayed.



16 Click [OK] 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 the F3 [PREV] key.

5.5. Setting up 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 sensor still needs to be taught even after you finish making these settings. The sensor object is trained after setting reference position (“I.5.10. Reference position setting(P.382)”).

5.5.1. Setting a sensor in detail

[This Sensor]

Specify whether to use the sensor selected on the screen. To use this sensor, set [Enable].

NOTE

In 7DH1/18 (V10.10P/18) or later, if the number of active sensors exceeds the maximum number (4) in each robot controller, “Limitation Exceeded” will be displayed in red to the right of the toggle switch.

[Controller]

If the robot ring is set, select the name of the robot controller to which the sensor is connected. If the robot ring is not set (i.e., there is only one robot controller), only [This Controller] is available.

NOTE

The robot for the robot controller selected in [Controller] must be selected at conveyor stations under the same conveyor and downstream of this sensor in the tree view.

[Trigger Condition]

Select the trigger condition from the following:

[Distance]

The action is executed each time the conveyor travels a specified distance.

[DI]

The action is triggered when the rising edge of the DI signal (state change from off to on) is detected.

[HDI]

The action is triggered when the input of the HDI signal is detected. This is a more precise trigger condition than DI.

[RI]

The action is triggered when the input of the RI signal is detected.

[FLAG]

The action is triggered when the flag turns ON. This is automatically selected when the servo conveyor is used.

[Trigger Action]

Select the action to be taken when the trigger condition is met from the following:

[Put part in queue]

A part is put into the queue.

The following restrictions are applied:

- The model ID is fixed as 0.
- [For each model ID] cannot be selected in [Load Balance] for a conveyor.

[Find part by vision]

The vision system searches for a part and puts the detected part into the queue. This action is not available when the trigger condition is [HDI] or [FLAG].

[Find part by multiview]

The vision system searches for a part and puts a detected part into the queue with a multi-view vision process. This action is not available when the trigger condition is [HDI] or [FLAG]. This action is only for a line conveyor.

[Run program]

The specified TP program is executed.

The following restrictions are applied:

- This action is not available when the trigger condition is [HDI] or [FLAG].
- This action can be used only on a linear conveyor.
- This action cannot be used on a conveyor using a tray.
- This action cannot be used for an infeed conveyor in pre-grouping.
- This action can only be used in the vision I/F add-on. For details, see “F. Vision I/F add-on(P.736)”.

NOTE

[Run program] cannot be used in ROBOGUIDE.

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

Table 5.5.1-1 Possible combinations of trigger condition and action

	Distance	DI	HDI	RI	FLAG
Put part in queue	✓	✓	✓	✓	✓
Find part by vision	✓	✓		✓	
Run program	✓	✓		✓	

For the setup screen that will be displayed depending on the combination of the trigger condition and action, see “I.5.5.1.1. Distance and Put part in queue(P.290)” to “I.5.5.1.13. RI and Run program(P.301)”.

5.5.1.1. Distance and Put part in queue

A workpiece is put into the queue each time the conveyor travels a specified distance.

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”, specify the angle (unit: degrees).

[Target Position]

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

[Reference Position]

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

5.5.1.2. Distance and Find part by vision

A vision process is executed each time the conveyor travels a specified distance.

[Trigger Distance]

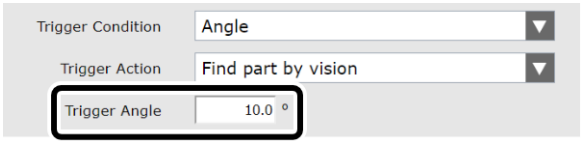
Specify the distance in mm that the conveyor is to travel before a vision process is executed. It is recommended that the trigger distance be set such that a workpiece can be detected twice in the camera FOV.

⚠CAUTION

If you set a [Trigger Distance] that is too short and the vision detection is often started before the previous vision detection completes, alarm 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. For more information, refer to [Action Time(P.569)].

In the case of a circular conveyor

For circular conveyors, specify the angle (units: degrees) for the item [Trigger Angle] in the case of [Distance] + [Find part by vision].



[Vision Process]

Enter the name of the vision process to be executed.

[Tray Position]

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

[Reference Position]

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

5.5.1.3. Distance and Run program

The specified TP program is executed each time the conveyor travels a specified distance.

⚠CAUTION

Use ACCUTRIG.
Use the encoder count command only in latch programs. Otherwise, accurate trigger values may not be obtained.
For details on latch programs, see “F.5.4.5. Latch program(P.750)” in “F. Vision I/F add-on(P.736)”.
For the combinations of the trigger conditions and actions for using ACCUTRIG, see “Table 5.5.7-1(P.317)”.

[Trigger Distance]

Specify the distance in mm that the conveyor is to travel before executing the TP program.

[DI/RI Num. to get Enc Cnt]

This is the number of the input signal used to inform the robot controller of the image capture timing for the external machine vision. The robot controller acquires the encoder count when this signal is input. Also, you cannot specify DI/RI numbers to get encoder counts that have already been specified for other sensors.

[Program Name]

Specify the name of the TP program to be executed. Programs using motion groups cannot be used. You cannot specify program names that have already been specified for other sensors.

5.5.1.4. DI and Put part in queue

A part is put into the queue when the rising edge of the DI signal (state change from off to on) is detected.

CAUTION

Use ACCUTRIG. Do not use the encoder count command of the tracking command. Doing so prevents workpiece detection.
For the combinations of the trigger conditions and actions for using ACCUTRIG, see "Table 5.5.7-1(P.317)".

[DI Num.]

Specify the number of the digital input signal to be detected.

CAUTION

This number is different from [Trigger INPUT number] displayed on the tracking schedule setting screen of the robot's teach pendant.

[Filtering] box

These items set signal filtering.

For details, see "I.8.14. Filtering for signals(P.675)".

NOTE

[Specified By], [Duration], and [Polarity Inversion] are supported in 7DH1/15 (V10.10P/15) and later versions. In versions prior to 7DH1/15 (V10.10P/15), the behavior is the same as when [Specified By] is set to [Time], and both [Duration] and [Polarity Inversion] are set to [Disable].

[Specified By]

Select whether to specify Debounce and Duration by [Time] or [Distance].

[Distance] can be selected only when [Tracking Type] is set to "Line".

[Debounce]

When a signal input is detected, if the next signal input is detected before the time or distance specified here has elapsed, that signal will be ignored as a mistaken input. By specifying a proper value, you can prevent the load on the CPU from increasing due to multiple detections caused by the debounce input of an external sensor.

[Duration]

Set whether or not to use the Duration function. To use the function, set the value for duration.

The Duration function executes an action only when the signal continues to be input for the duration or distance specified in [Duration] after the time or distance specified in [Debounce] has passed since the signal was first detected. It is considered valid only if the signal remains stable during this period.

If the signal is not detected during the "Duration" period, the signal is rejected. Also, as soon as the signal is rejected, the system begins waiting for the next signal input.

This function can only be used when [Tracking Type] is set to "Line".

[Polarity Inversion]

Set whether or not to use the Polarity Inversion function.

Polarity Inversion is a function that uses the moment the rear of the part passes the sensor as the trigger, which is the opposite of the normal setting.

This function can only be used when [Tracking Type] is set to “Line” or when the Duration function is enabled.

NOTE

When using Polarity Inversion, the trigger count is acquired when the rear of the part passes the sensor, so the signal must transition from OFF to ON at that moment. Therefore, it is necessary to either use a sensor that turns OFF when detecting a part or use the I/O attribute setting to reverse the polarity.

[Sensor Position]

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

[Reference Position]

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

5.5.1.5. DI and Find part by vision

When the rising edge of the DI signal (state change from off to on) is detected, a vision process is executed and the detected part is put into the queue.

[DI Num.]

Specify the number of the digital input signal to be detected.

⚠CAUTION

This number is different from [Trigger INPUT number] displayed on the tracking schedule setting screen of the robot’s teach pendant.

[Filtering] box

These items set signal filtering.

[Duration] and [Polarity Inversion] cannot be used with this combination of trigger condition and action.

For details, see “I.8.14. Filtering for signals(P.675)”.

NOTE

[Specified by] is supported in 7DH1/15 (V10.10P/15) and later versions. In versions prior to 7DH1/15 (V10.10P/15), the behavior is the same as when [Specified By] is set to [Time].

[Specified By]

Select whether to specify Debounce and Duration by [Time] or [Distance].

[Distance] can be selected only when [Tracking Type] is set to “Line”.

[Debounce]

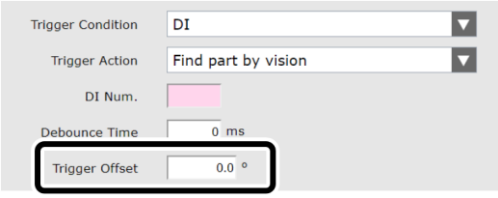
When a signal input is detected, if the next signal input is detected before the time or distance specified here has elapsed, that signal will be ignored as a mistaken input. By setting this value appropriately, you can prevent situations where the vision process is unintentionally executed multiple times due to chatter in the external sensor input, which would otherwise increase CPU load.

[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 cases such as when the camera is installed apart from the photo eye.

In the case of a circular conveyor

For circular conveyors, specify the angle (units: degrees) for the item [Trigger Offset] in the case of [DI]/[RI] + [Find part by vision].



[Vision Process]

Specify the name of the vision process to be executed.

[Tray Position]

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

[Reference Position]

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

5.5.1.6. DI and Find part by multiview

When the rising edge of the DI signal (state change from off to on) is detected, a multi-view vision process is executed and the detected part is put into the queue.

[DI Num.]

Specify the number of the digital input signal to be detected.

⚠CAUTION

This number is different from [Trigger INPUT number] displayed on the tracking schedule setting screen of the robot’s teach pendant.

[Filtering] box

These items set signal filtering.

[Duration] and [Polarity Inversion] cannot be used with this combination of trigger condition and action.

For details, see “I.8.14. Filtering for signals(P.675)”.

NOTE

[Specified by] is supported in 7DH1/15 (V10.10P/15) and later versions. In versions prior to 7DH1/15 (V10.10P/15), the behavior is the same as when [Specified By] is set to [Time].

[Specified By]

Select whether to specify Debounce and Duration by [Time] or [Distance].

[Distance] can be selected only when [Tracking Type] is set to “Line”.

[Debounce]

When a signal input is detected, if the next signal input is detected before the time or distance specified here has elapsed, that signal will be ignored as a mistaken input. By setting this value appropriately, you can prevent situations where the vision process is unintentionally executed multiple times due to chatter in the external sensor input, which would otherwise increase CPU load.

[Trigger Offset 1]

Specify a distance in mm by which the conveyer moves after the part passes by a sensor, such as a photo eye, until the first feature is snapped and found by camera view 1.

[Trigger Offset 2]

Specify a distance in mm by which the conveyer moves after the part passes by a sensor, such as a photo eye, until the second feature is snapped and found by camera view 2.

[Vision Process]

Specify the name of the vision process to be executed.

[Reference Position]

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

5.5.1.7. DI and Run program

The specified TP program is executed when the rising edge (from OFF to ON) of the DI signal is detected.

⚠CAUTION

Use ACCUTRIG.
Use the encoder count command only in latch programs. Otherwise, accurate trigger values may not be obtained.
For details on latch programs, see “F.5.4.5. Latch program(P.750)” in “F. Vision I/F add-on(P.736)”.
For the combinations of the trigger conditions and actions for using ACCUTRIG, see “Table 5.5.7-1(P.317)”.

[DI Num.]

Specify the number of the digital input signal to be detected.

⚠CAUTION

This number is different from [Trigger INPUT number] displayed on the tracking schedule setting screen of the robot’s teach pendant.

[Filtering] box

These items set signal filtering.

[Duration] and [Polarity Inversion] cannot be used with this combination of trigger condition and action.

For details, see “I.8.14. Filtering for signals(P.675)”.

NOTE

[Specified by] is supported in 7DH1/15 (V10.10P/15) and later versions. In versions prior to 7DH1/15 (V10.10P/15), the behavior is the same as when [Specified By] is set to [Time].

[Specified By]

Select whether to specify Debounce and Duration by [Time] or [Distance].

[Distance] can be selected only when [Tracking Type] is set to “Line”.

[Debounce]

When a signal input is detected, if the next signal input is detected before the time or distance specified here has elapsed, that signal will be ignored as a mistaken input. By setting this value appropriately, you can prevent situations where the vision process is unintentionally executed multiple times due to chatter in the external sensor input, which would otherwise increase CPU load.

[Trigger Offset]

The TP program is executed when the conveyor travels the distance specified here after the rising edge of the DI signal is detected.

[DI/RI Num. to get Enc Cnt]

This is the number of the input signal used to inform the robot controller of the image capture timing for the external machine vision. The robot controller acquires the encoder count when this signal is input. You cannot specify the same input signal as that specified for DI Num. Also, you cannot specify DI/RI numbers to get encoder counts that have already been specified for other sensors.

[Program Name]

Specify the name of the TP program to be executed. Program using motion groups cannot be used. You cannot specify program names that have already been specified for other sensors.

5.5.1.8. HDI and Put part in queue

When the input of the HDI signal is detected, a workpiece 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.

[Filtering] box

These items set noise removal for signals.

[Duration] and [Polarity Inversion] cannot be used with this combination of trigger conditions and actions.

For details, see “I.8.14. Filtering for signals(P.675)”.

[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 a proper value, you can prevent the load on the CPU from increasing due to multiple detections caused by the debounce input of an external sensor.

[Sensor Position]

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

[Reference Position]

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

5.5.1.9. Flag and Put part in queue

This is selected automatically only if a servo conveyor is used.

When the flag changes from OFF to ON, a workpiece is put into the queue. The flag is input automatically every time the servo conveyor moves by one pitch.

[FLAG Num.]

The flag number to be detected is displayed. The flag number is specified in the servo conveyor setup menu You cannot change this number in this menu.

[Target Position]

This is the part position that is generated when the flag turns ON. Set these values using the target position setting guide described later.

[Reference Position]

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

5.5.1.10. RI and Put part in queue

When the input of the RI signal is detected, a workpiece is put into the queue.

The procedure to start up the combination [RI] + [Put part in queue] is the same as that for [DI] + [Put part in queue].

For details, see “I.2. iRPickTool Basic (basic functions) startup procedures(P.26)” or “I.3. iRPickTool (basic functions and Plug & Play) startup procedures(P.152)”.

⚠CAUTION

Use ACCUTRIG. Do not use the encoder count command of the tracking command. Doing so prevents workpiece detection.
For the combinations of the trigger conditions and actions for using ACCUTRIG, see “Table 5.5.7-1(P.317)”.

[RI Num.]

Specify the number of the input signal to be detected.

⚠CAUTION

This number is different from [Trigger INPUT number] displayed on the tracking schedule setting screen of the robot’s teach pendant.

[Filtering] box

These items set signal filtering.

For details, see “I.8.14. Filtering for signals(P.675)”.

NOTE

[Specified By], [Duration], and [Polarity Inversion] are supported in 7DH1/15 (V10.10P/15) and later versions. In versions prior to 7DH1/15 (V10.10P/15), the behavior is the same as when [Specified By] is set to [Time], and both [Duration] and [Polarity Inversion] are set to [Disable].

[Specified By]

Select whether to specify Debounce and Duration by [Time] or [Distance].

[Distance] can be selected only when [Tracking Type] is set to “Line”.

[Debounce]

When a signal input is detected, if the next signal input is detected before the time or distance specified here has elapsed, that signal will be ignored as a mistaken input. By specifying a proper value, you can prevent the load on the CPU from increasing due to multiple detections caused by the debounce input of an external sensor.

[Duration]

Set whether or not to use the Duration function. To use the function, set the value for duration.

The Duration function executes an action only when the signal continues to be input for the duration or distance specified in [Duration] after the time or distance specified in [Debounce] has passed since the signal was first detected. It is only considered valid if the signal remains stable during this period.

If the signal is not detected during the “Duration” period, the signal is rejected. Also, as soon as the signal is rejected, the system begins waiting for the next signal input.

This function can only be used when [Tracking Type] is set to “Line”.

[Polarity Inversion]

Set whether or not to use the Polarity Inversion function.

Polarity Inversion is a function that uses the moment the rear of the part passes the sensor as the trigger, which is the opposite of the normal setting.

This function can only be used when [Tracking Type] is set to “Line” or when the Duration function is enabled.

NOTE

When using Polarity Inversion, the trigger count is acquired when the rear of the part passes the sensor, so the signal must transition from OFF to ON at that moment. Therefore, it is necessary to either use a sensor that turns OFF when detecting a part or use the I/O attribute setting to reverse the polarity.

[Sensor Position]

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

[Reference Position]

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

5.5.1.11. RI and Find part by vision

When the rising edge of the RI signal (state change from off to on) is detected, a vision process is executed and the detected part is put into the queue.

[RI Num.]

Specify the number of the input signal to be detected.

⚠CAUTION

This number is different from [Trigger INPUT number] displayed on the tracking schedule setting screen of the robot's teach pendant.

[Filtering] box

These items set signal filtering.

[Duration] and [Polarity Inversion] cannot be used with this combination of trigger condition and action.

For details, see "I.8.14. Filtering for signals(P.675)".

NOTE

[Specified by] is supported in 7DH1/15 (V10.10P/15) and later versions. In versions prior to 7DH1/15 (V10.10P/15), the behavior is the same as when [Specified By] is set to [Time].

[Specified By]

Select whether to specify Debounce and Duration by [Time] or [Distance].

[Distance] can be selected only when [Tracking Type] is set to "Line".

[Debounce]

When a signal input is detected, if the next signal input is detected before the time or distance specified here has elapsed, that signal will be ignored as a mistaken input. By setting this value appropriately, you can prevent situations where the vision process is unintentionally executed multiple times due to chatter in the external sensor input, which would otherwise increase CPU load.

[Trigger Offset]

A vision process is executed if the conveyor travels the distance specified here after the rising edge of the RI is detected. Use this offset in cases such as when the camera is installed apart from the photo eye.

In the case of a circular conveyor

For circular conveyors, specify the angle (units: degrees) for the item [Trigger Offset] in the case of [DI]/[RI] + [Find part by vision].



[Vision Process]

Specify the name of the vision process to be executed.

[Tray Position]

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

[Reference Position]

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

5.5.1.12. RI and Find part by multiview

When the rising edge of the RI signal (state change from off to on) is detected, a multi-view vision process is executed and the detected part is put into the queue.

[RI Num.]

Specify the number of the input signal to be detected.

⚠CAUTION

This number is different from [Trigger INPUT number] displayed on the tracking schedule setting screen of the robot's teach pendant.

[Filtering] box

These items set signal filtering.

[Duration] and [Polarity Inversion] cannot be used with this combination of trigger condition and action.

For details, see "I.8.14. Filtering for signals(P.675)".

NOTE

[Specified by] is supported in 7DH1/15 (V10.10P/15) and later versions. In versions prior to 7DH1/15 (V10.10P/15), the behavior is the same as when [Specified By] is set to [Time].

[Specified By]

Select whether to specify Debounce and Duration by [Time] or [Distance].

[Distance] can be selected only when [Tracking Type] is set to "Line".

[Debounce]

When a signal input is detected, if the next signal input is detected before the time or distance specified here has elapsed, that signal will be ignored as a mistaken input. By setting this value appropriately, you can prevent situations where the vision process is unintentionally executed multiple times due to chatter in the external sensor input, which would otherwise increase CPU load.

[Trigger Offset 1]

Specify a distance in mm by which the conveyer moves after the part passes by a sensor, such as a photo eye, until the first feature is snapped and found by camera view 1.

[Trigger Offset 2]

Specify a distance in mm by which the conveyer moves after the part passes by a sensor, such as a photo eye, until the second feature is snapped and found by camera view 2.

[Vision Process]

Specify the name of the vision process to be executed.

[Reference Position]

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

5.5.1.13. RI and Run program

The specified TP program is executed when an input of the RI signal is detected.

⚠CAUTION

Use ACCUTRIG.
Use the encoder count command only in latch programs. Otherwise, accurate trigger values may not be obtained.
For details on latch programs, see "F.5.4.5. Latch program(P.750)" in "F. Vision I/F add-on(P.736)".
For the combinations of the trigger conditions and actions for using ACCUTRIG, see "Table 5.5.7-1(P.317)".

[RI Num.]

Specify the number of the input signal to be detected.

⚠CAUTION

This number is different from [Trigger INPUT number] displayed on the tracking schedule setting screen of the robot's teach pendant.

[Filtering] box

These items set signal filtering.

[Duration] and [Polarity Inversion] cannot be used with this combination of trigger condition and action.

For details, see "I.8.14. Filtering for signals(P.675)".

NOTE

[Specified by] is supported in 7DH1/15 (V10.10P/15) and later versions. In versions prior to 7DH1/15 (V10.10P/15), the behavior is the same as when [Specified By] is set to [Time].

[Specified By]

Select whether to specify Debounce and Duration by [Time] or [Distance].

[Distance] can be selected only when [Tracking Type] is set to "Line".

[Debounce]

When a signal input is detected, if the next signal input is detected before the time or distance specified here has elapsed, that signal will be ignored as a mistaken input. By setting this value appropriately, you can prevent situations where the vision process is unintentionally executed multiple times due to chatter in the external sensor input, which would otherwise increase CPU load.

[Trigger Offset]

The TP program is executed when the conveyor travels the distance specified here after the rising edge of the DI signal is detected.

[DI/RI Num. to get Enc Cnt]

This is the number of the input signal used to inform the robot controller of the image capture timing for the external machine vision. The robot controller acquires the encoder count when this signal is input. You cannot specify the same input signal as that specified for RI Num. Also, you cannot specify DI/RI numbers to get encoder counts that have already been specified for other sensors.

[Program Name]

Specify the name of the TP program to be executed. Program using motion groups cannot be used. You cannot specify program names that have already been specified for other sensors.

5.5.2. Settings required for each combination of a trigger condition and a trigger action

The setup required will vary depending on the combination of trigger condition and action.

NOTE

For information on Multi-view, refer to "I.8.8. Multi-view"(P.611)"

Table 5.5.2-1 Settings required for each combination of a trigger condition and a trigger action

Trigger Condition	Action	Tray used / not used	Setting the sensor position	Setting up the vision system	Setting the tray position	Setting the target position
[Distance]	[Put part in queue]	Used				✓
		Not used				✓
	[Find part by vision]	Used		✓	✓	
		Not used *1		✓		
	[Run program]	Not used *2				
[DI]	[Put part in queue]	Used *1	✓			
		Not used	✓			
	[Find part by vision]	Used		✓	✓	
		Not used		✓		
	[Run program]	Not used *2				
[HDI]	[Put part in queue]	Used *1	✓			
		Not used	✓			
[FLAG]	[Put part in queue]	Used				✓
		Not used				✓
[RI]	[Put part in queue]	Used *1	✓			
		Not used	✓			
	[Find part by vision]	Used		✓	✓	
		Not used		✓		
	[Run program]	Not used *2				

*1 For the settings and the setup procedure, see "I.2. iRPickTool Basic (basic functions) startup procedures(P.26)" or "I.3. iRPickTool (basic functions and Plug & Play) startup procedures(P.152)".

*2 When [Run program] is specified as the action, these settings are configured in the TP program, not on the iRPickTool Setup screen. For details, see "F. Vision I/F add-on(P.736) ".

5.5.3. Setting up a sensor position

Teach the position where the photo eye is installed.

This setup is necessary when any of [DI]/[HDI]/[RI] is selected for the trigger condition and the action is "Put part in queue".

When a part crosses the photo eye, this position is loaded to the queue. The procedure slightly differs depending on whether a tray is used on the conveyor or not.

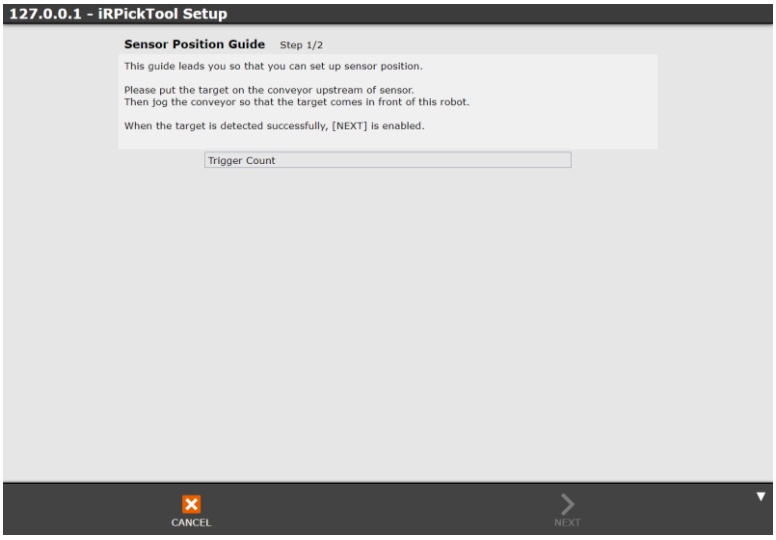
NOTE

To set the sensor position, you need to open the iRPickTool Setup screen with the robot controller selected for this sensor.

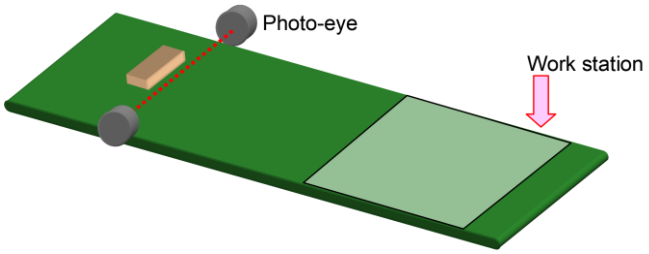
For how to set the sensor position when using the tray, see "I.2. iRPickTool Basic (basic functions) startup procedures(P.26)" or "I.3. iRPickTool (basic functions and Plug & Play) startup procedures(P.152)".

5.5.3.1. When a tray is not used

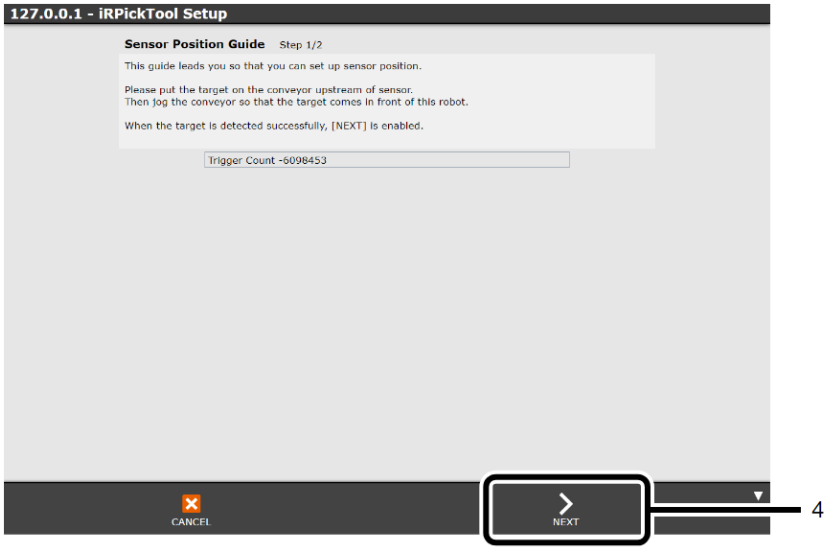
1. **Stop the conveyor.**
2. **Click [Sensor Position].**
   The following screen appears.



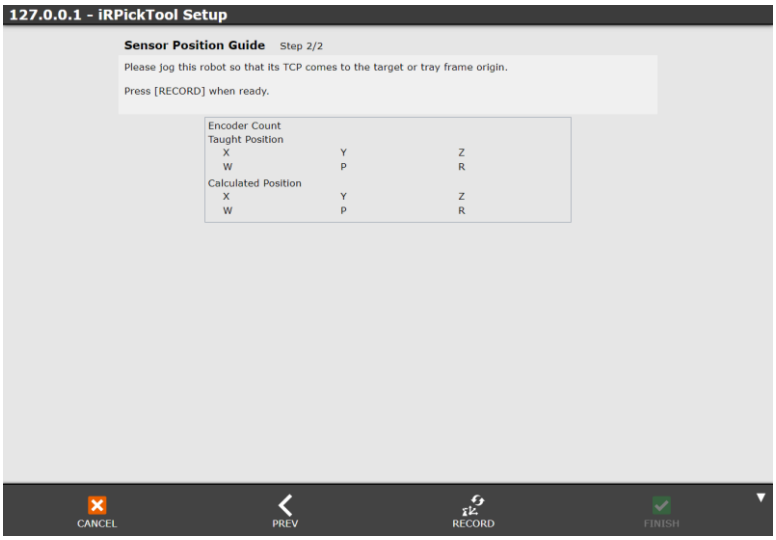
3. **Place a part upstream of the photo eye, and move the conveyor until the part is in front of the robot.**



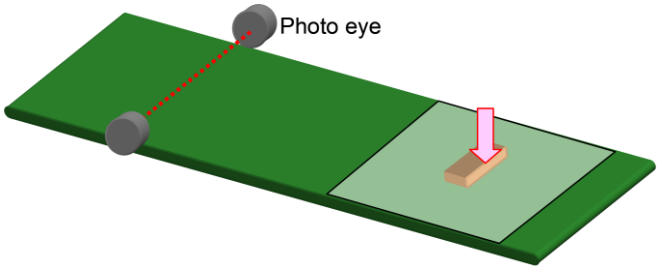
   When the sensor detects the part, the encoder value obtained at the time of detection is displayed and [NEXT] is enabled.
4. **Click [NEXT].**



The following screen appears.



5 Jog the robot and touch up the operation position of the workpiece.



NOTE

When the position touched up in this procedure passes the upstream boundary of the tracking area, the workpiece will be allocated to the robot. If the downstream end of the workpiece is touched up while the operation position is at the center, a situation may occur where the robot waits for the workpiece because it has been allocated to the robot, but its operation position has not yet passed the upstream boundary of the tracking area.

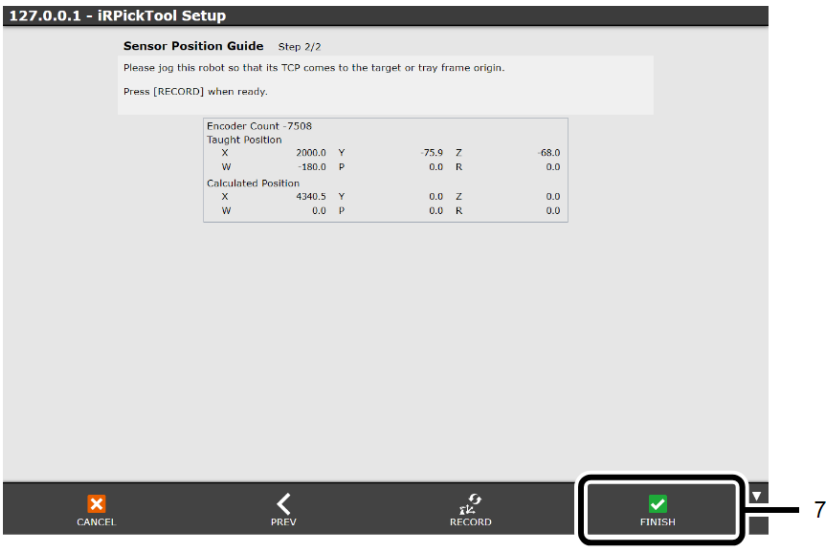
6 Click [Record].

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

NOTE

In the case of a multi group setup, select the robot group that was touched up in Step 5 and click [Record].

7 Click [FINISH].



The screen returns to the original setup screen. The calculated value is input for X of the sensor position (the values of Y and R remain unchanged from 0).

5.5.4. 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 robot controller to which a camera is connected.
- Limit check of [2-D Single-View Visual Tracking] and [2-D Multi-View Visual Tracking] cannot be used.

For how to set up the vision when using a 2D camera, see "I.2. iRPickTool Basic (basic functions) startup procedures(P.26)" or "I.3. iRPickTool (basic functions and Plug & Play) startup procedures(P.152)".

NOTE

Check the Z-axis direction of the tracking frame before setting [Part Z Height] in [Reference Data].



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

NOTE

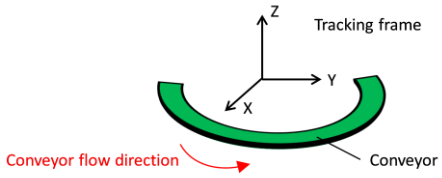


Fig. 5.5.4-1 Z-axis when the conveyor moves counterclockwise

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

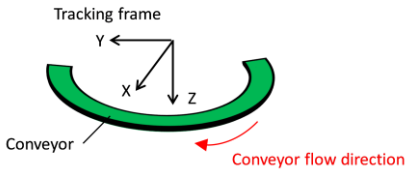


Fig. 5.5.4-2 Z-axis when the conveyor moves clockwise

For how to set up the vision when using a 3DV Vision Sensor, see "I.8.9. 3D Vision Sensor(P.620)".
For how to set up the vision when using multiple 2D cameras (multi-view), refer to "I.8.8. Multi-view(P.611)".

5.5.5. Setting the tray position

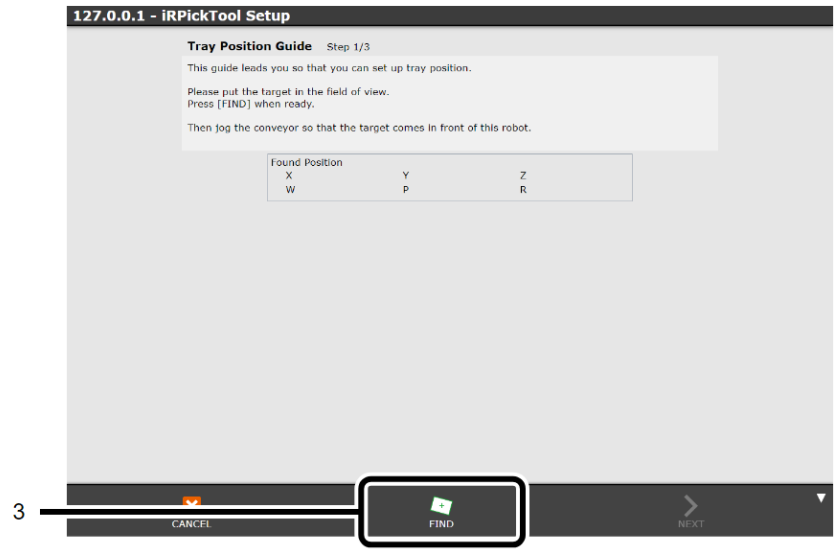
If [Find part by vision] is selected for [Trigger Action] and a tray is used, set the relative position of the origin and detection position of the tray. To do this requires that camera calibration and vision process teaching be completed in advance.

NOTE

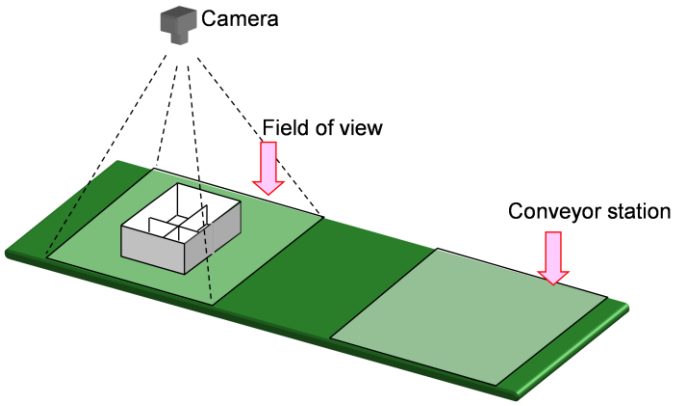
To set the tray position, you need to open the iRPickTool Setup screen with the robot controller selected for this sensor.

5.5.5.1. When [Distance] and [Find part by vision] are selected

1 **Stop the conveyor.**
2 **Click [SET TRY POS].**
The following screen appears.

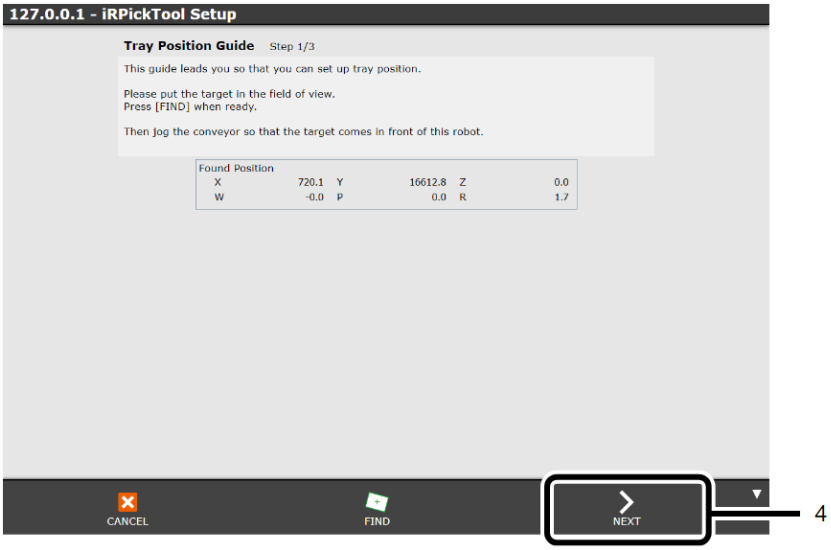


3 Place the tray inside the camera's field of view, and click [FIND].



When tray detection is successful, the detection position is displayed and [NEXT] is enabled.

4 Click [NEXT].

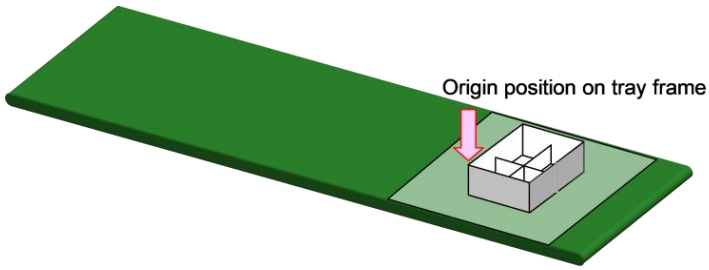


The following screen appears.



5 **Move the conveyor until the tray comes in front of the robot.**

6 **Jog the robot, and touch up the origin of the tray frame.**

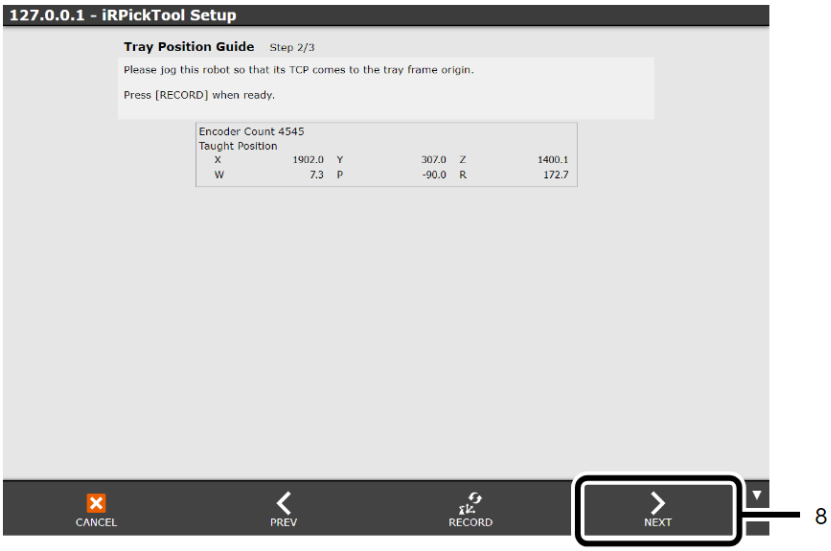


7 **Click [Record].**

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

> **NOTE**
>
> In the case of a multi group setup, select the robot group that was touched up in Step 6 and click [Record]. This applies to all subsequent steps.

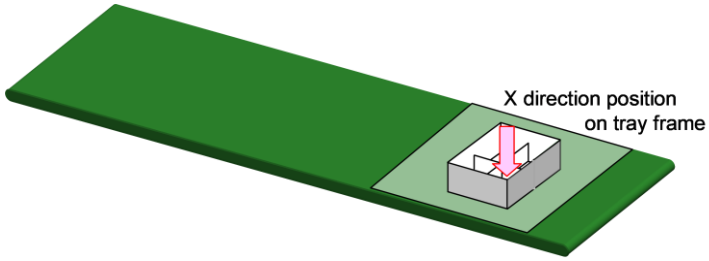
8 **Click [NEXT].**



The following screen appears.



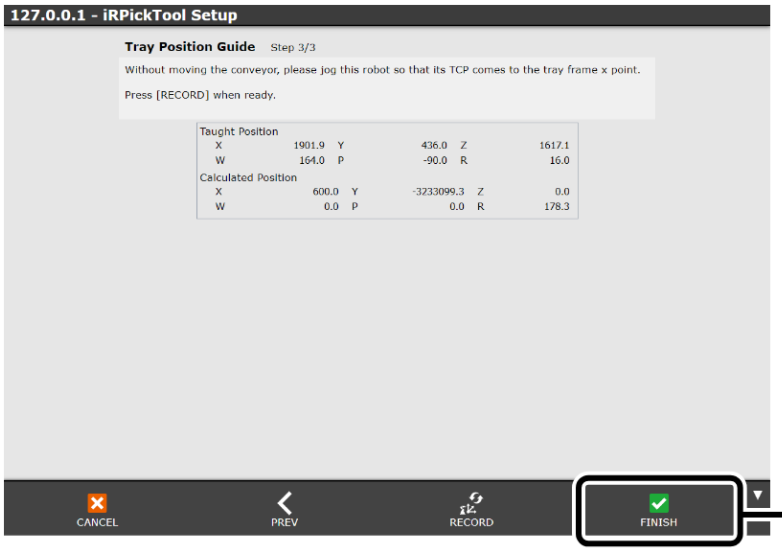
9 Jog the robot, and touch up the X direction point of the tray frame.



10 Click [Record].

The touch-up position and the calculated position are displayed, and [FINISH] is enabled.

11 Click [FINISH].



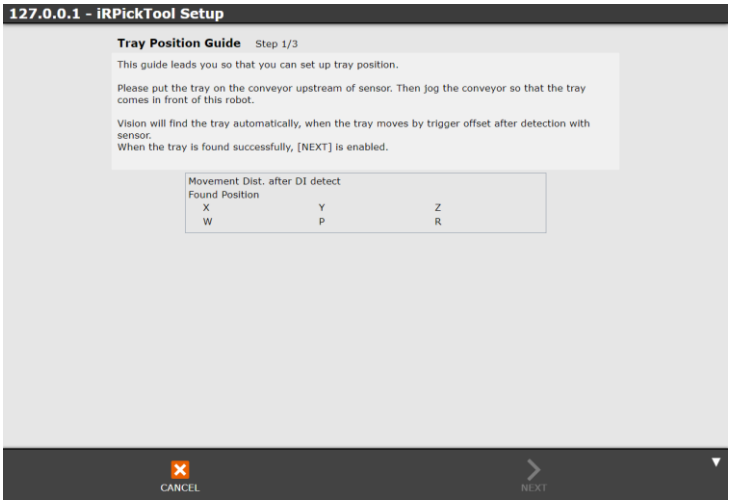
The screen returns to the original setup screen. Values are input for X, Y, and R of the tray position.

5.5.5.2. When [DI] and [Find part by vision] are selected

1 Stop the conveyor.

2 Click [Tray Position].

The following screen appears.



3 Place a tray on the most upstream section of the conveyor, and operate the conveyor until the tray is in front of the robot.

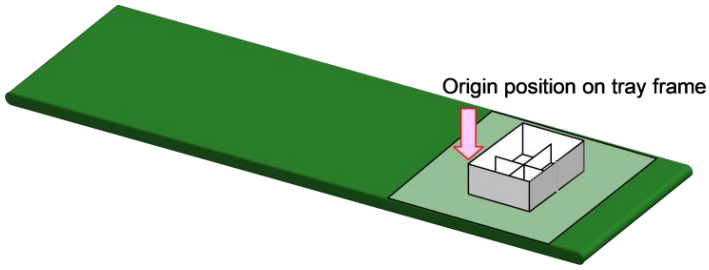
When the conveyor has moved by the trigger offset after the photo eye has detected the tray, iRVision will detect the part automatically. You do not have to stop the conveyor when iRVision detects the tray. When iRVision successfully detects the part, the detection position are displayed and [NEXT] is enabled.

4 Click [NEXT].

The following screen appears.



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



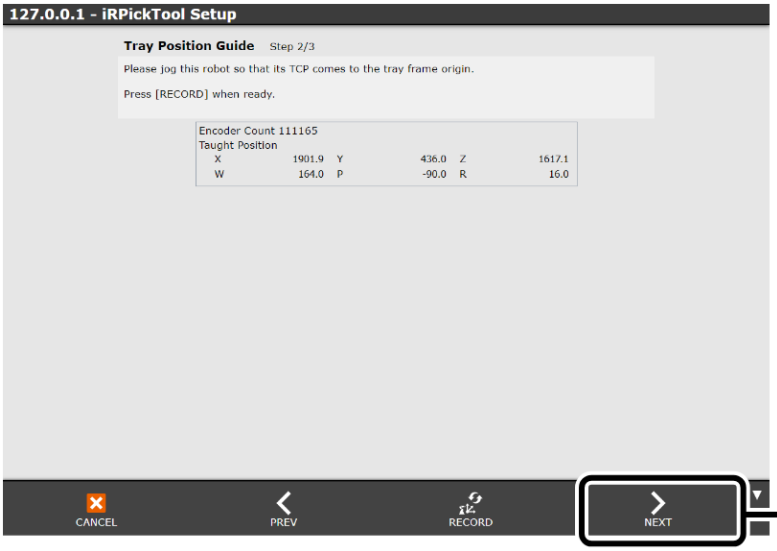
6 Click [Record].

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

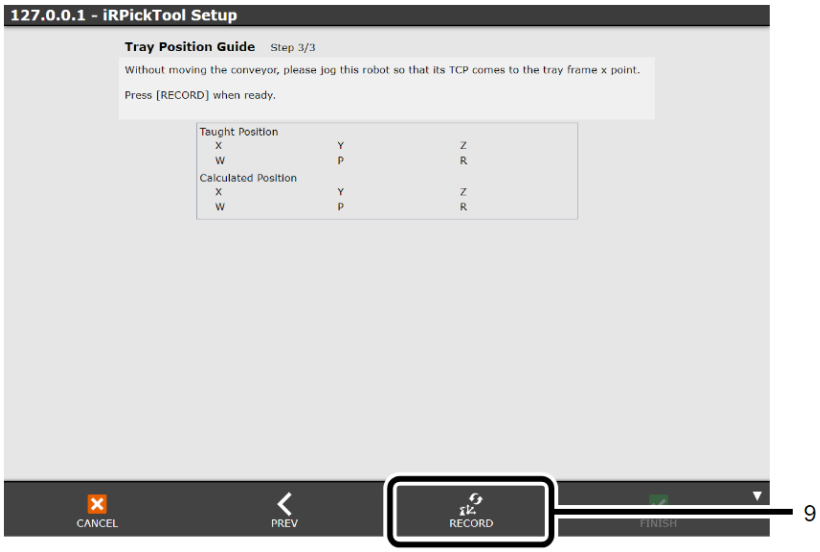
NOTE

In the case of a multi group setup, select the robot group that was touched up in Step 5 and click [Record]. This applies to all subsequent steps.

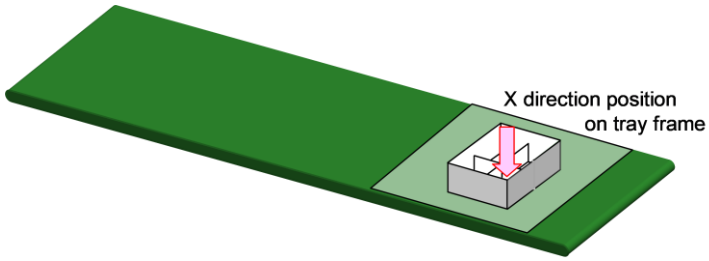
7 Click [NEXT].



The following screen appears.



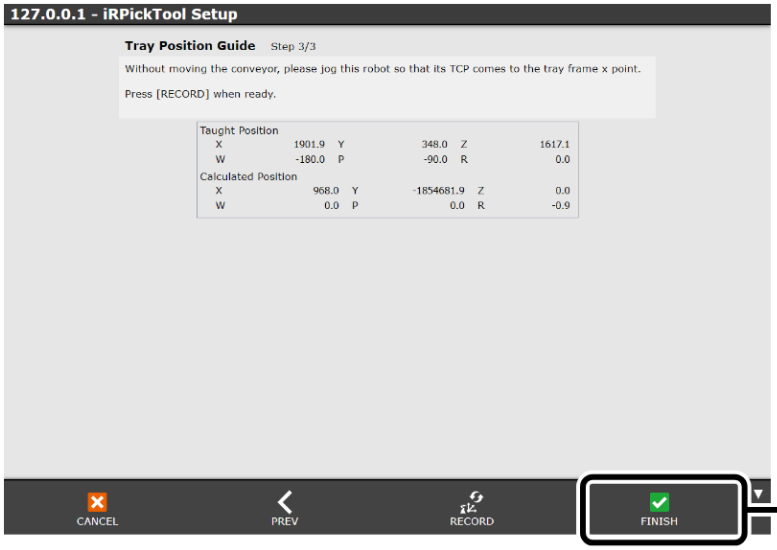
8 Jog the robot, and touch up the X direction point of the tray frame.



9 Click [Record].

The touch-up position and the calculated position are displayed, and [FINISH] is enabled.

10 Click [FINISH].



The screen returns to the original setup screen. Values are input for X, Y, and R of the tray position.

5.5.6. 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 trigger condition and action used.

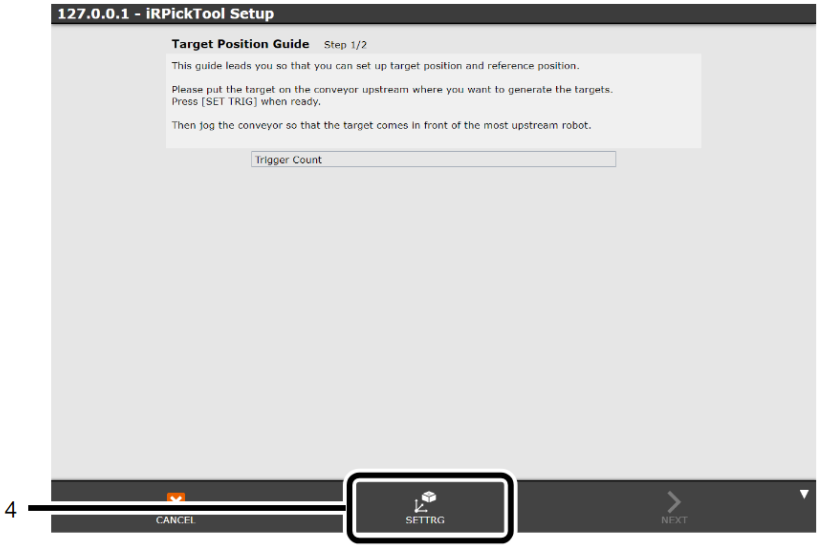
Teach the position where a part is to be generated. This position is put into the queue each time the conveyor moves a certain distance or the [Index Advance Trigger DI] of the servo conveyor is input. Since a part is not actually detected when these combinations are selected, perform robot position teaching at the same time as when you set the target position. The procedure slightly differs depending on whether a tray is used on the conveyor or not.

NOTE

To set the target position, you need to open the iRPickTool Setup screen with the robot controller selected for this sensor.

5.5.6.1. When a tray is not used

1. **Stop the conveyor.**
2. **Click [Target Position]. The following screen appears.**

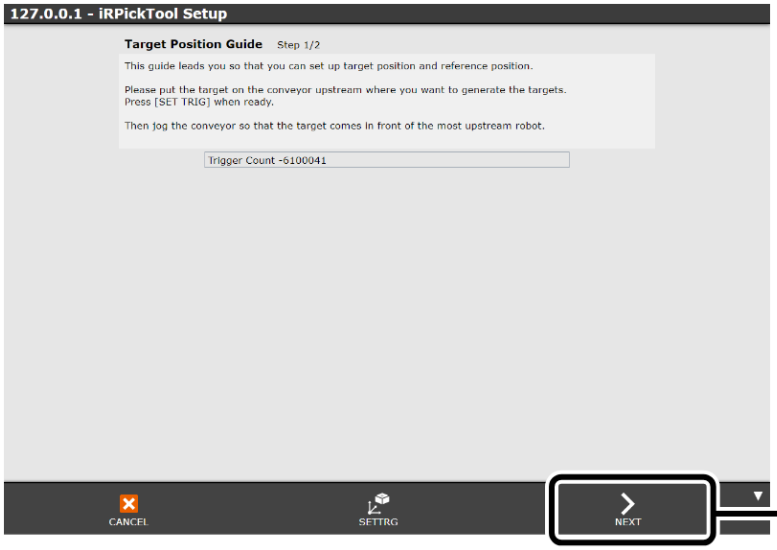


3 Place a part at the position where you want to generate it.

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 Click [SETTRG]. The current encoder value is displayed and [NEXT] is enabled.

5 Click [NEXT].

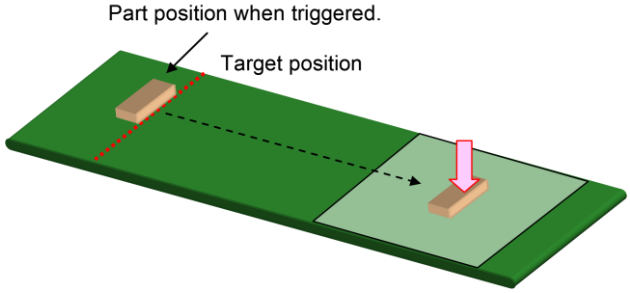


The following screen appears.



6 Move the conveyor, and perform robot position teaching sequentially, beginning with the most upstream robot. For details, see "I.5.11. Teaching the position to robots(P.385)".

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 click [RECORD].

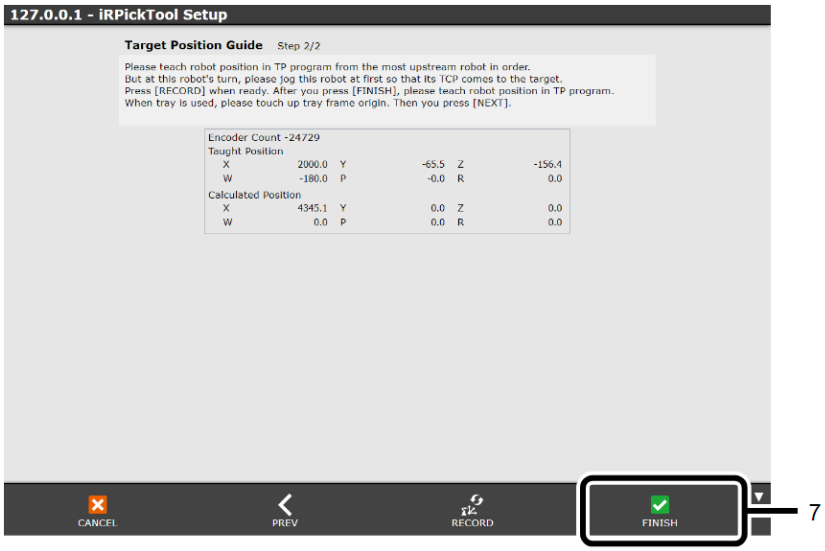


The encoder value obtained at the time of touch-up, the touch-up position, and the calculated position are displayed, and [FINISH] is enabled.

NOTE

- When the position touched up in this procedure passes the upstream boundary of the tracking area, the workpiece will be allocated to the robot. If the downstream end of the workpiece is touched up while the operation position is at the center, a situation may occur where the robot waits for the workpiece because it has been allocated to the robot, but its operation position has not yet passed the upstream boundary of the tracking area.
- In the case of a multi group setup, select the robot group that was touched up and click [Record].

7 Click [FINISH].



The screen returns to the original setup screen. The calculated value is input for X of the target position (the values of Y and R remain unchanged from 0).

8 Teach the position to this robot.

9 Then, move the conveyor further to teach the position to the downstream robots.

5.5.6.2. When a tray is used

1 Stop the conveyor.

2 Click [Target Position]. The following screen appears.

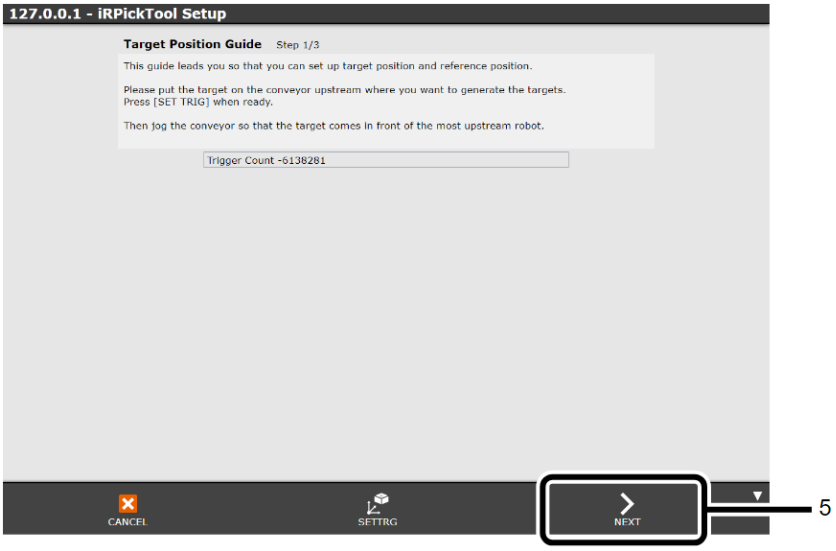


3 Place a tray at the position where you want to generate it.

In order to perform robot position teaching for all robots, the tray needs to be placed upstream of the most upstream robot operation range.

4 Click [SETTRG]. The current encoder value is displayed and [NEXT] is enabled.

5 Click [NEXT].



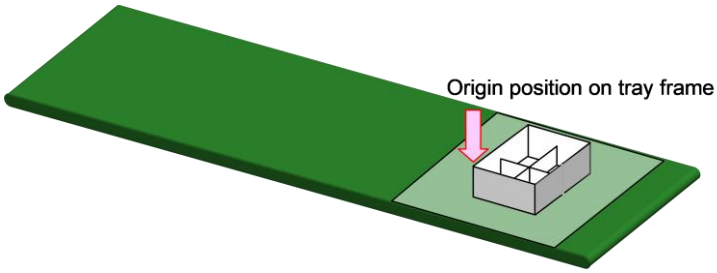
The following screen appears.



6 Move the conveyor, and perform robot position teaching sequentially, beginning with the most upstream robot.

For details, see “I.5.11. Teaching the position to robots(P.385)”.

When the part comes in front of this robot, first jog the robot and touch up the origin of the tray frame. Then, click [RECORD].

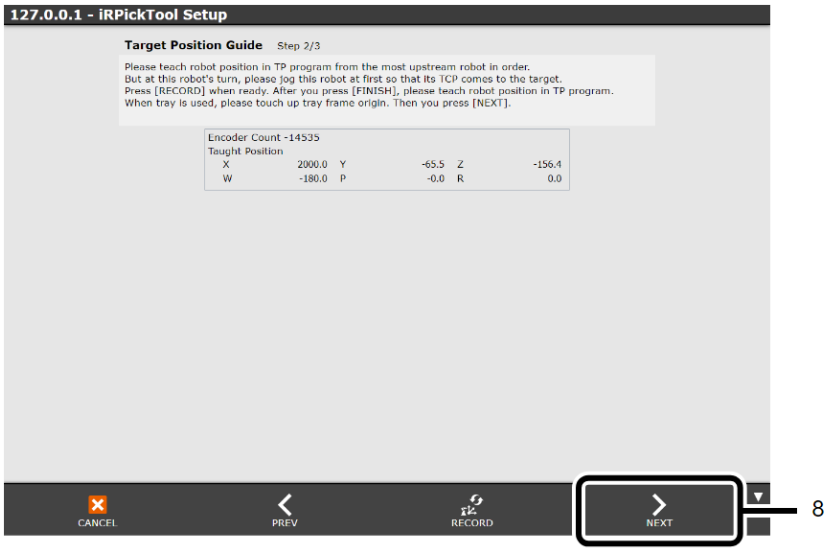


NOTE

In the case of a multi group setup, select the robot group that was touched up and click [Record]. This applies to all subsequent steps.

7 The encoder value obtained at the time of touch-up and the touch-up position are displayed, and [NEXT] is enabled.

8 Click [NEXT].



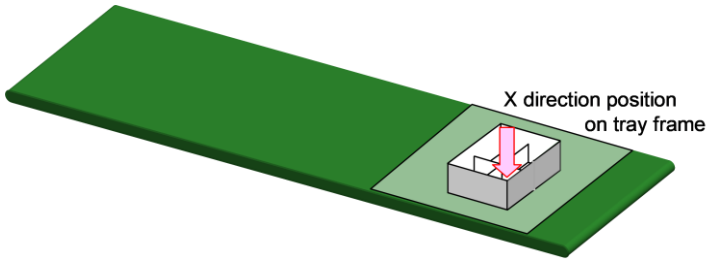
The following screen appears.



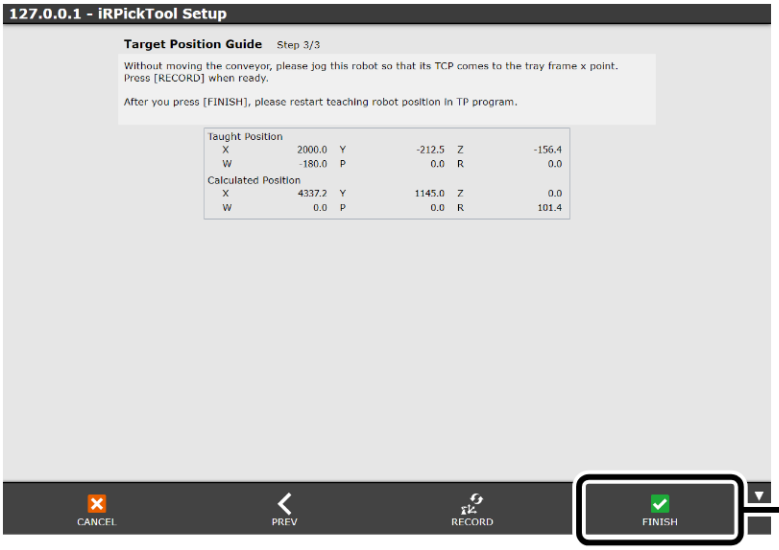
9 Jog the robot, and touch up the X direction point of the tray frame.

10 Click [Record].

The touch-up position and the calculated position are displayed, and [FINISH] is enabled.



11 Click [FINISH].



The screen returns to the original setup screen. Values are input for X, Y, and R of the target position.

12 Teach the position to this robot.

13 Then, move the conveyor further to teach the position to the downstream robots.

5.5.7. Restriction concerning sensors that can be operated simultaneously

Up to four sensors can be operated simultaneously by one robot controller. Do not add sensors for different actions to a single conveyor object.

NOTE

Use one task for each sensor.

When using the Multi Task function, it is recommended to increase the number of tasks.
Based on the maximum number set in the controlled start menu, increase the number of tasks according to the number of sensors executed by each robot controller.
For information about the controlled start menu, refer to the “Controlled start” section under the “Start mode” subsection in the “Special operations” chapter of the appendix in "OPERATOR’S MANUAL(Basic Operation)"(B-84464EN).

If there is a sensor combination that satisfies all the following conditions, the sensors cannot be started. In such a case, take measures such as changing the robot controller to which the sensors are connected.

- All sensors are connected to the same robot controller.
- All sensors use ACCUTRIG. *1
- The encoder used in the conveyor station corresponding to each sensor is the same. *2

*1 For the combinations of the trigger conditions and actions for using ACCUTRIG, see Table 5.5.7-1.

Table 5.5.7-1 Combinations of trigger conditions and actions for using ACCUTRIG

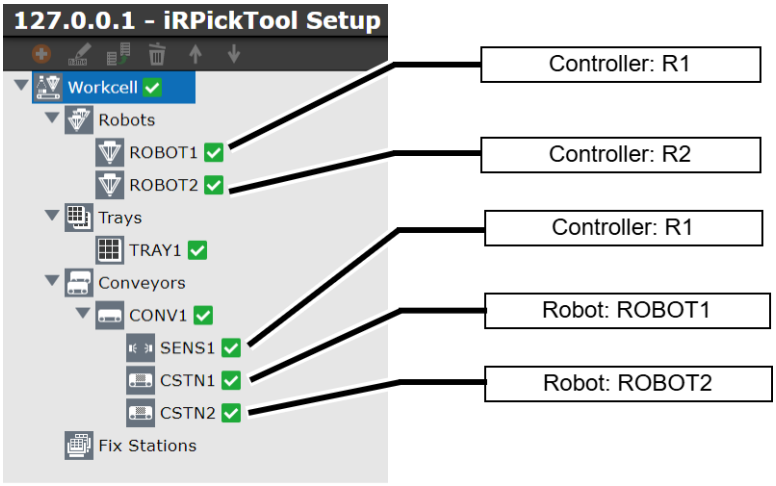
Trigger condition	Action	Use of ACCUTRIG
[Distance]	Put part in queue	
	Find part by vision	
	Run program	✓
[DI]	Put part in queue	✓
	Find part by vision	

Trigger condition	Action	Use of ACCUTRIG
	Run program	✓
	Find part by multiview	
[HDI]	Put part in queue	
[FLAG]	Put part in queue	
[RI]	Put part in queue	✓
	Find part by vision	
	Run program	✓
	Find part by multiview	

*2 The conveyor station corresponding to the sensor refers to the one that satisfies all of the following conditions:

- The conveyor station is under the same conveyor in the tree view and is located below this sensor.
- The robot controller corresponding to the robot at the conveyor station and the robot controller to which the sensor is connected are the same.

In the configuration shown in the figure below, CSTN1 is the conveyor station corresponding to SENS1.



5.6. Setting up 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.
The maximum number of conveyor stations that can be added with one robot controller is eight. If you add nine or more conveyor stations, alarm INTP-310 “(PKSHCUST, number) Array index error” occurs when the power is turned on. In addition, functions such as Recipe Operation, Dynamic Load Balance, and check of whether a program is being executed, do not work.

[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.

[Line Tracking Schedule]

Select the number of the tracking schedule to be used for this conveyor station. You specify the different number for each conveyor normally.

[Encoder Number]

Select the number of the encoder to be used for this conveyor station.

[Encoder Scale]

The currently set encoder scale is displayed.

[Tracking Frame]

The status and coordinates of the currently set tracking frame are displayed.

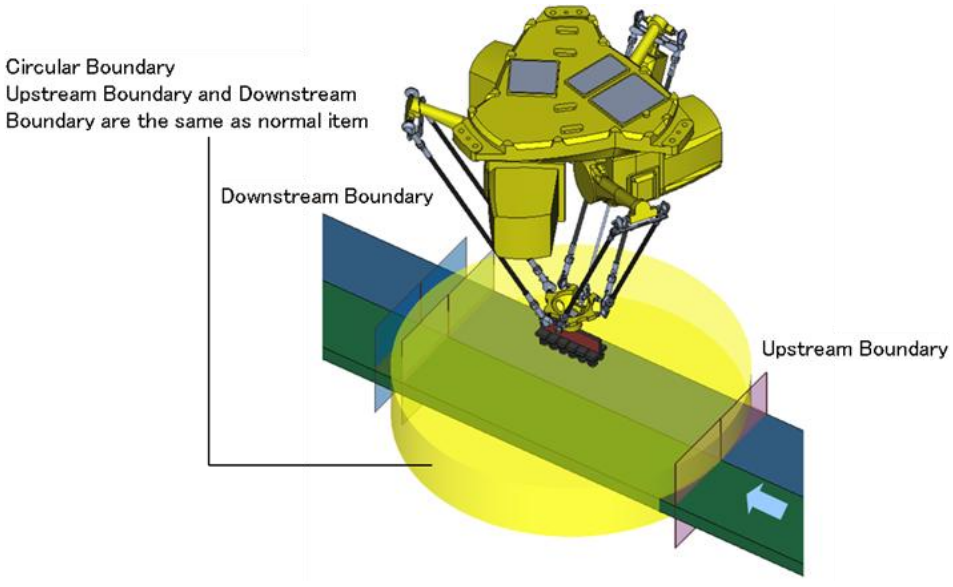
[Selected Boundary]

Select the number of the tracking area to be used. Normally, select 1.

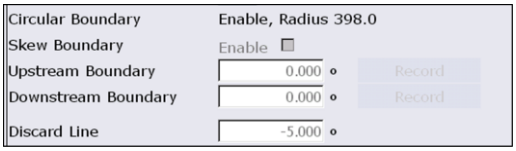
NOTE

To change the tracking area to be used by a TP program, use the SELBOUND tracking instruction. For details, see the description of tracking instructions in "Line Tracking OPERATOR'S MANUAL" (B-83474EN).
Changes made by the SELBOUND tracking instruction are not automatically reflected on the iRPickTool Setup screen. Close the Setup screen and reopen it, or refresh the tree.

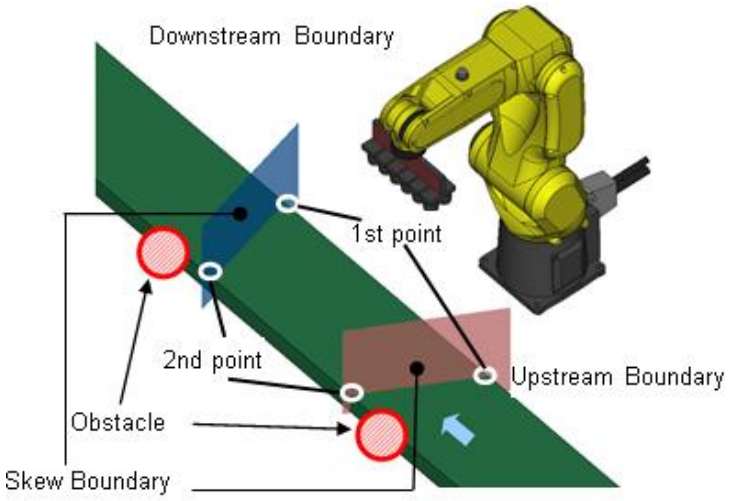
[Circular Boundary]



[Enable] and [Radius] will be displayed as shown below only when [Circular Boundary] for the selected robot is enabled.

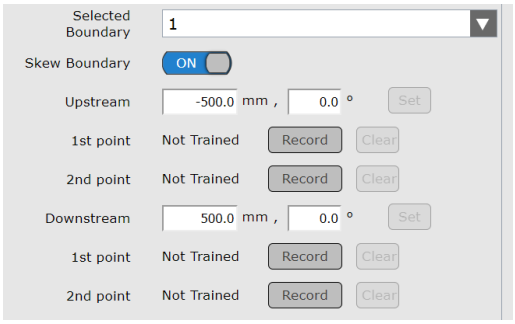


[Skew Boundary]



If you want to set a skew boundary, enable this.

If you enable skew boundary, skew positions on the upstream and downstream sides can be recorded, as shown below.



If you have selected circular tracking, this cannot be selected.

[1st point]

Touch up the 1st point on the upstream boundary and the 1st point on the downstream boundary, and set record.

[2nd point]

Touch up the 2nd point on the upstream boundary and the 2nd point on the downstream boundary, and set record.

[Record] button	Record the set positions for the 1st and 2nd points. When you press the [Record] button, “Not Trained” will change to “Trained”.
[Clear] button	Clear the set positions for the 1st and 2nd points.

[Upstream Boundary]

Specify the position of the upstream boundary of the tracking area. [Upstream Boundary] represents the most upstream position that the robot can reach.

When setting [Upstream Boundary] and [Downstream Boundary], make sure that the user tool frame is selected that is used by the program that causes the robot to follow the motion of the conveyor. See “I.5.9.2.2. Pick program(P.370)”.

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 degrees is set. The value can be changed manually as well.

For the positional relationship between [Upstream Boundary], [Downstream Boundary], and the [Destroy Boundary], refer to “Fig. 5.6-1”.

If you enable skew boundary, you will be able to set the 1st point and 2nd point for the skew boundary.

[Downstream Boundary]

Specify the position of the downstream boundary of the tracking area. [Downstream Boundary] represents the most downstream position that the robot can reach.

Move the TCP by jogging to the downstream boundary of the tracking area and click the [Record] button. The X coordinate value in the tracking frame is set. When the tracking type is “Circle”, the angle in degrees is set. The value can be changed manually as well.

For the positional relationship between [Downstream Boundary], [Upstream Boundary], and the [Destroy Boundary], refer to “Fig. 5.6-1”.

If you enable skew boundary, you will be able to set the 1st point and 2nd point for the skew boundary.

[Set] button

Set the mm value and deg value for the upstream boundary of the tracking area and the downstream boundary of the tracking area.

[Discard Line]

Specify the position where the robot decides to discard a part carried on the conveyor. This is an offset from the downstream boundary of the tracking area ([Downstream Boundary]). Normally, set a 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.

For the positional relationship between [Discard Line], [Downstream Boundary], and [Upstream Boundary], refer to “Fig. 5.6-1”.

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.

In the case of a line conveyor

If the robot operation time is approximately 0.8 seconds and the conveyor speed is 100 mm/s, the free-running distance of the conveyor is:

0.8 (s) × 100 (mm/s) = 80 (mm)

In this case, set the value of the discard line to −100 mm or so, allowing for some margin.

In the case of a circular conveyor

If the robot operation time is approximately 0.8 seconds and the conveyor speed is 5 deg/s, the free-running distance of the conveyor is:

0.8 (s) × 5 (deg/s) = 4 (deg).

In this case, set the value of the discard line to −5 deg or so, allowing for some margin.

⚠CAUTION

When the stop/start conveyor function described below is enabled, even if a part passes the discard line, a robot moves to pick it up within the area that does not extend beyond the downstream boundary. 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 needs to 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 too small, 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

NOTE

there is always at least one part between the upstream boundary of the tracking area and the discard line, the robot can pick up parts continuously. For example, if parts are supplied at almost even intervals, widening the space between the upstream boundary of the tracking area and the discard line makes it easier for the robot to pick up parts continuously, rather than supplying parts at even intervals.

Positional relationship of the items

The positional relationship of the upstream boundary of the tracking area, downstream boundary of the tracking area, and discard line is outlined below.

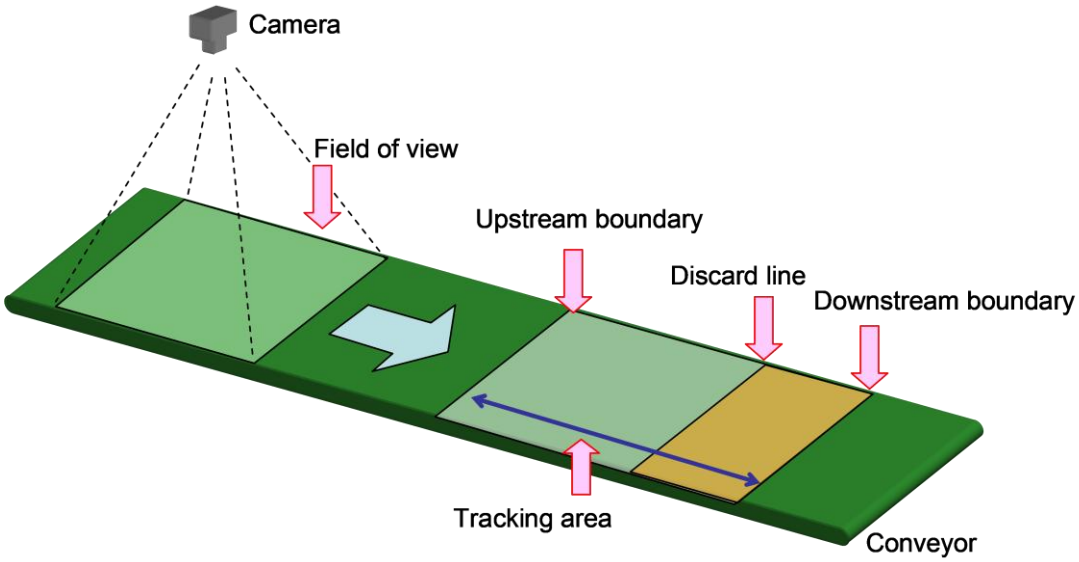


Fig. 5.6-1 Positional relationship between tracking area and discard line

[Buffer]

This item is displayed when “Forward Direction” is selected for [Buffer] on the conveyor setup screen.

For details, see “I.8.15. Using a conveyor as a buffer(P.678)”.

[Signal Output (Buffer)]

This item is displayed when “Forward Direction” is selected for [Buffer] on the conveyor setup screen.

For details, see “I.8.15. Using a conveyor as a buffer(P.678)”.

[Sort]

The sort function changes the order of parts on the conveyor to change the order of the pick or place operations. There are two sorting methods: [Y Sort] and [Meas 1 Sort]. Each method allows you to choose either ascending or descending order.

⚠CAUTION

Sorting cannot be performed in the following cases.

- A tray is set for the conveyor
- Skew Boundary is enabled
- Circular Boundary is enabled for the robot
- Load Balance for [For each model ID] is enabled
- Clearance check is used
- Pre-grouping is used
- Pick or Drop Multiple for Servo Conveyor is enabled

[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 cases where the parts are arranged side by side relative to the direction of conveyor movement, if a part is picked up from the farthest side from the destination, it may collide with nearby parts, resulting in a failure to pick it up. Such failures can be avoided by picking up the parts starting from the one closest to the destination.

In [X Tolerance], specify a range in 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.

[Meas 1 Sort]

This is a function that sorts information about the parts on the conveyor based on the priority set by the user. This function is useful when it is necessary to customize the pick-up order, such as when picking up bulk-loaded parts.

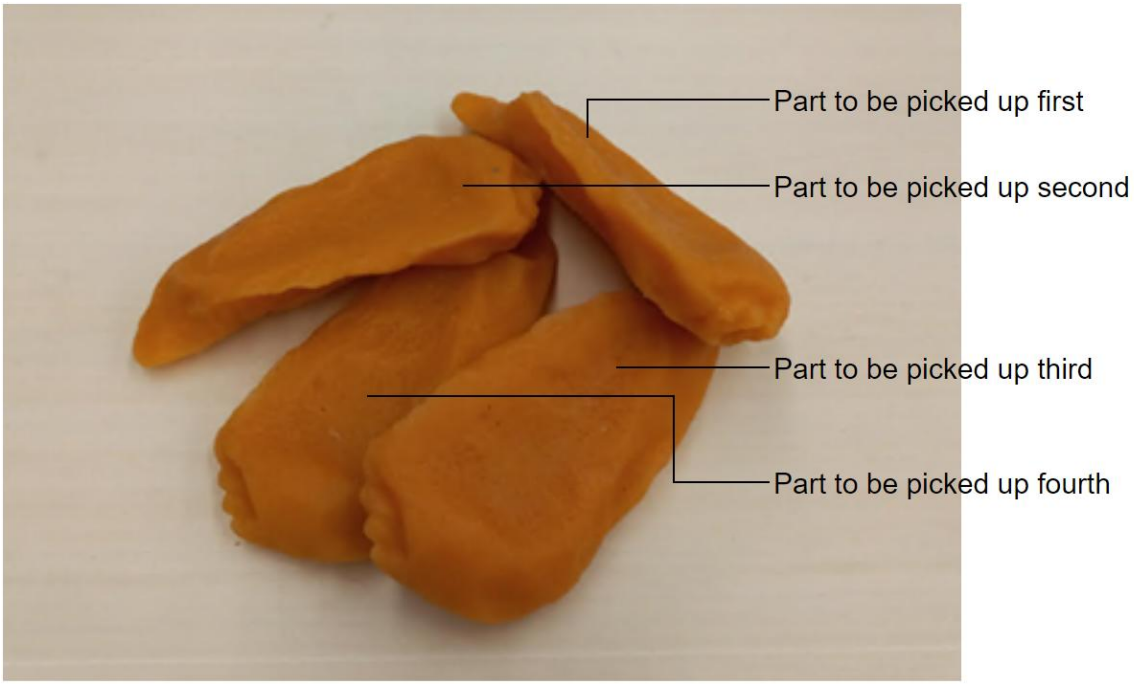


Fig. 5.6-2 Example of bulk-loaded parts

The priority of the parts to be picked can be freely set.

When [Meas 1 Sort] is enabled, it is necessary to set the order for Measurement Value 1 using one of the following methods.

When using iRVsion

When using iRVsion, the priority can be set using the measurement output tool. When setting a vision process, create the measurement output tool in the “Calculation, Logic” category. In Value 1, use the dropdown box on the left to select the command tool that outputs the desired value, and use the dropdown box on the right to select the value you want to output.

For details of the setup, refer to the “Measurement output tool” section in the “Command tools” chapter in the "iRVision OPERATOR'S MANUAL (Reference)"(B-84434EN).

When customizing a sensor task

Customize a sensor task to set the priority. For details, refer to “E. Sensor customization(P.717)”.

Turn Custom Sensor Task on. Before executing put_queue() within the run_vis_cst() routine, assign a priority to meas[1] of the detected part obtained using VT_GET_FOUND(). This process is repeated until no more parts are detected.

When using iRVision, the part detection order is based on the score determined by the selected vision tool.

Table 5.6-1 Insertion points for priority assignment code (excerpt from pksnliubcst.kl)

```
...
212: -- Find parts
213: V_RUN_FIND(vp_name, -1, stat)
214: WHILE (stat = ER_SUCCESS) DO
215:
216: -- Get information of found parts
217: VT_GET_FOUND(vp_name, model_id, enc_count, offset, found_pos, meas, stat)
218: IF (stat = ER_SUCCESS) THEN
219:
220: -- If tray, compute tray position on track frame
221: IF (tray_id <> 0) THEN
222: tray_pos = get_tray_pos(cond_id)
223: found_pos[1] = found_pos[1] : tray_pos
224: ENDIF
225:
226: -- Put part information to queue
227: -- If User Defined Priorities Sorting is enabled, assign customized
        picking priority here before put_queue.
228: -- meas[1] is reserved for users to assign priority (e.g. meas[1] = 2)
229: -- Priority assignments should be based on Ascend or Descent selection.
230: -- Consider to call chk_cstn_usort(cstn_id, BOOLEAN) to confirm the sorting
        method is correctly selected.
231:                                                    ← Insert the code here
232: put_queue(conv_id,
233: cstn_id,
234: work_id,
235: enc_count,
236: model_id,
237: offset,
238: found_pos,
239: meas,
240: duplicated,
241: stat)
...
```

To enable [Meas 1 Sort], it is necessary to set the “X tolerance”. For details, refer to “Y sort”. It is recommended to set [Meas 1 Sort] for all the conveyor stations linked to the same conveyor.



MEMO

[Meas 1 Sort] is available on robot controllers with software version 7DH1/17 (V10.10P/17) or later.

[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 from the time the conveyor stop signal is issued 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]

This is a function that sets the starting line of the area where a part needs to be removed before restarting the conveyor after the part has crossed the discard line and the conveyor has stopped.

When the part crosses the discard line and the conveyor stops, removing the relevant part allows the conveyor to restart. However, if another part is located near the discard line, the conveyor may stop again shortly after restarting.

To avoid such a situation, set the conveyor restart line with a sufficient margin from the discard line. If there are parts downstream of the restart conveyor line, the conveyor will not restart.

See the figure below. The conveyor stopped because part A passes the stop conveyor line (= discard line). Even if part A is picked up, because part B is in the downstream area of the restart conveyor line, the conveyor does not restart. Once part B is removed, the conveyor restarts.

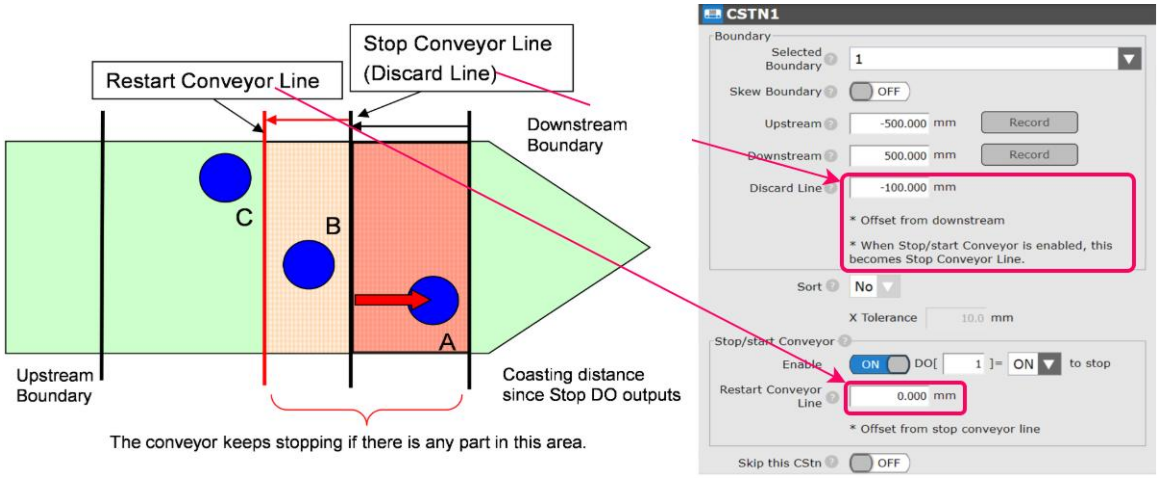


Fig. 5.6-3 Setup items for restart conveyor line and discard line

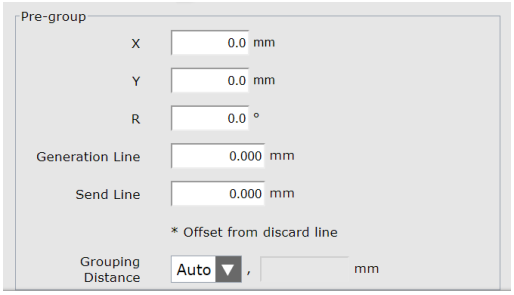
⚠CAUTION
1. In cases where multiple conveyor stations are defined with the same robot controller, if the same DO signal number is specified for the stop/start conveyor function and the polarity of the DO signal is reversed (i.e., the conveyor stops for some conveyor stations when DO is ON, while it stops for the other conveyor stations when DO is OFF), the stop/start conveyor function does not function properly. 2. To enable the stop conveyor function, set the average value (number of updates) after sufficient consideration. Set the value to 20 at first, then adjust it as necessary while performing the actual tracking operation. If the default value of 6 is used, “Move error excess” may occur because the robot attempts to precisely follow the motion of the conveyor as the conveyor stops or restarts. On the other hand, if the value is too large, there is a risk that the robots may not be able to follow the conveyor properly when it 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-group]

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



For details on pre-grouping, see “I.5.15. Setup of pre-grouping(P.393)”.

5.6.1. 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 settings.

The setting procedure is different with depending on the tracking type.

5.6.1.1. In the case of a line conveyor

When the tracking type is “Line”, see this procedure. When the tracking type is “Arc”, see “I.5.6.1.2. In the case of a circular conveyor(P.329)”.

1 On the conveyor station setup screen, click [ADD TRK FRM].

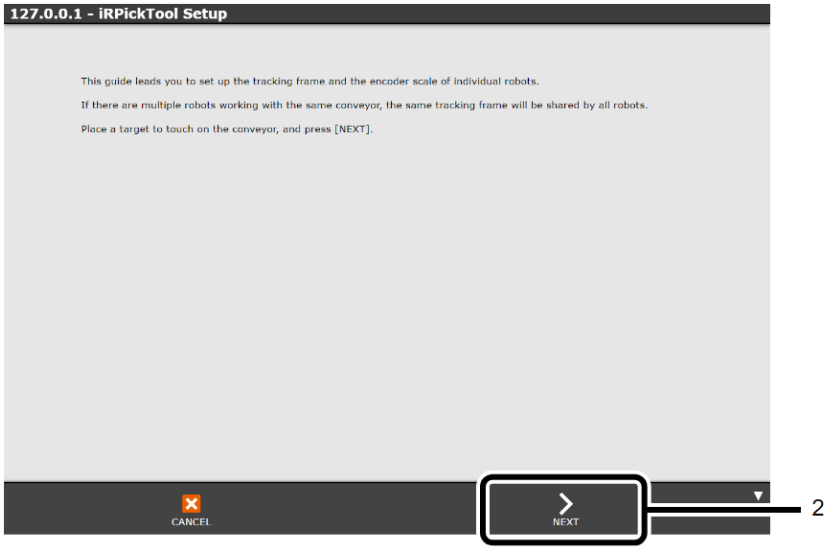


CAUTION

When using the αiA4000S, “IRPK-063 Encoder count does not match on all robots” may occur even when the conveyor is stationary, due to high-resolution encoder pulses. In this case, stop the conveyor and retry the operation several times.

2 The following screen appears.

Place a fixture that can be touched up with the TCP of the robot at the most upstream area of the conveyor, and click [NEXT].



CAUTION

1. The F4 [ADD TRK FRM] key is not displayed on the setup page for the most upstream conveyor station.
2. The 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.

3 For the most upstream robot, select the user tool frame set in "I.4.4. Setting up the tool frame(P.257)" as the jog feed frame.

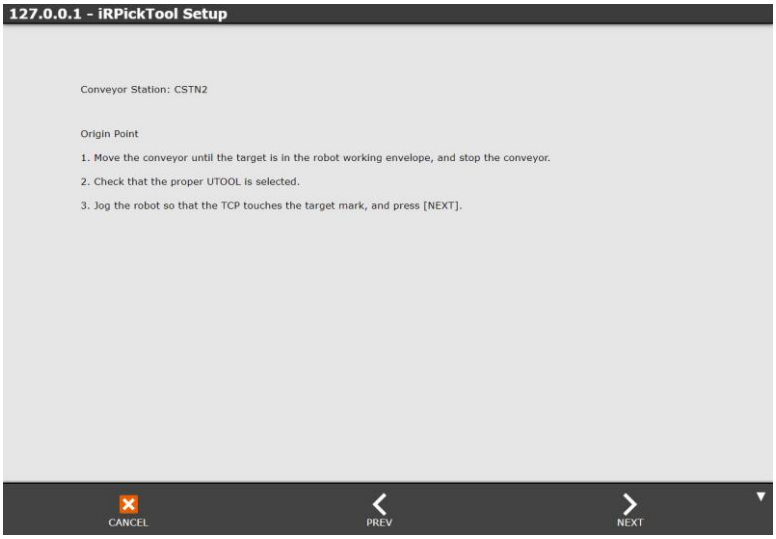
4 Move the conveyor until the fixture comes into the most upstream robot operation range, and stop it.

5 Jog the robot to touch up the fixture with the TCP.

6 With the fixture touched up, click [NEXT].



The following screen appears. The name of the conveyor station for which you are redoing the tracking frame setting is displayed on the first line.



7 **For the robot that you are redoing the tracking frame setup for, select the tool frame that you set up in "I.4.4. Setting up the tool frame(P.257)" as the jog feed frame.**

8 **For the conveyor station for which you are redoing the tracking frame setting, touch up the fixture as instructed in steps 14 through 22 in "I.2.3.4. Setting a tracking frame(P.72)".**

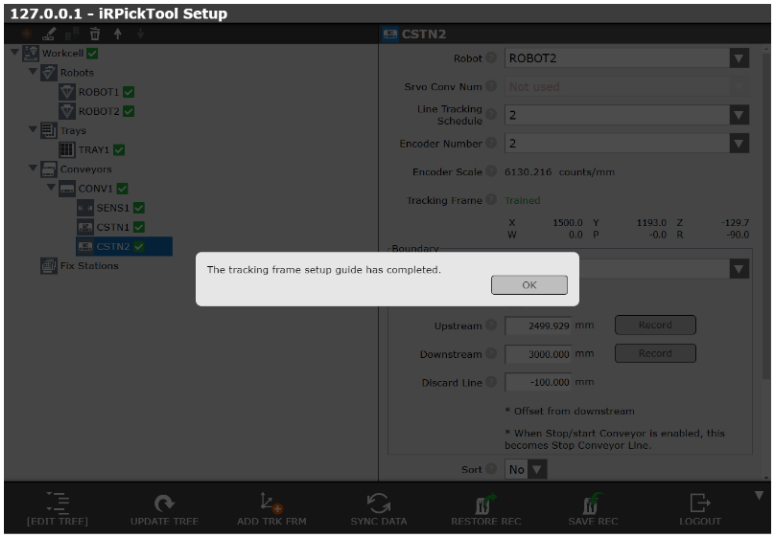
When you have proceeded to the step of setting the Y-axis direction point, the screen shown below is displayed.

When you have proceeded to the step of setting the assistant point, the screen shown below is displayed.

9 **Click [NEXT].**



After a while, the following screen appears.



5.6.1.2. In the case of a circular conveyor

When the tracking type is “Circle”, the procedure is as follows.

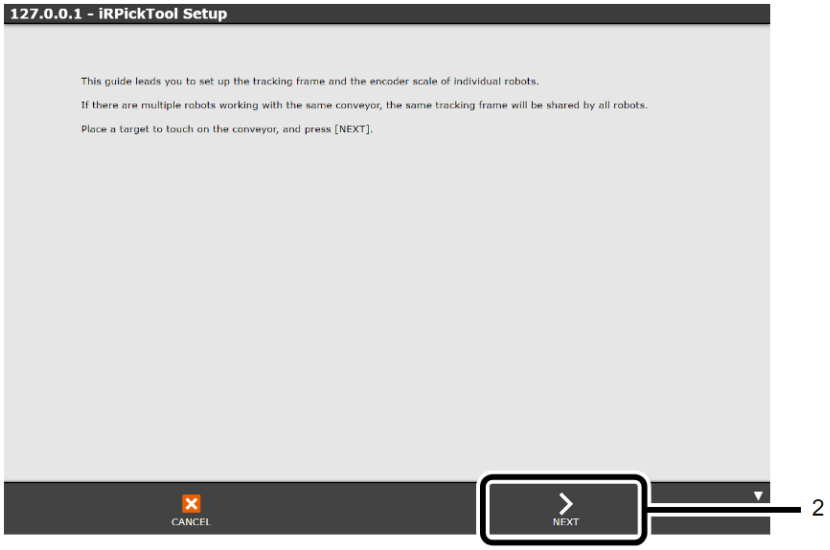
1 On the conveyor station setup screen, click [ADD TRK FRM].



CAUTION
When using the αiA4000S, “IRPK-063 Encoder count does not match on all robots” may occur even when the conveyor is stationary, due to high-resolution encoder pulses. In this case, stop the conveyor and retry the operation several times.

3 The following screen appears.

Place a fixture that can be touched up with the TCP of the robot at the most upstream area of the conveyor, and click [NEXT].



CAUTION

1. The F4 [ADD TRK FRM] key is not displayed on the setup page for the most upstream conveyor station.
2. The 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.

4 For the most upstream robot, select the user tool frame set in "I.4.4. Setting up the tool frame(P.257)" as the jog feed frame.

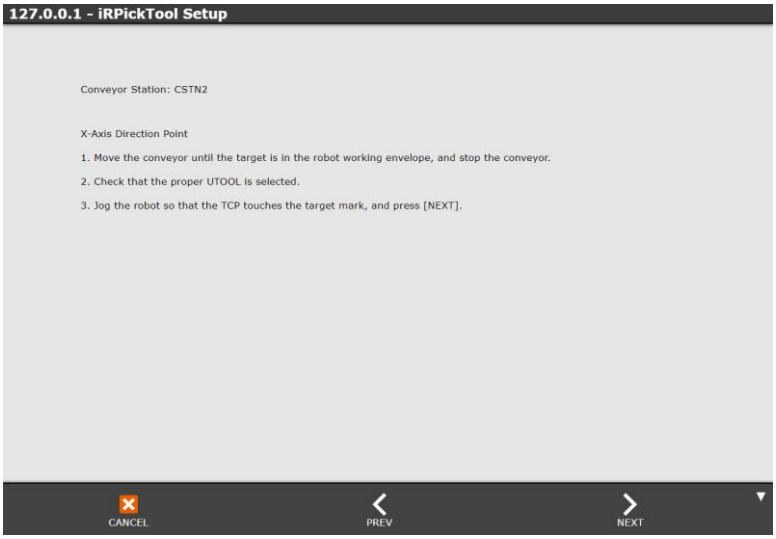
5 Move the conveyor until the fixture comes into the most upstream robot operation range, and stop it.

6 Jog the robot to touch up the fixture with the TCP.

7 With the fixture touched up, click [NEXT].



The following screen appears. The name of the conveyor station for which you are redoing the tracking frame setting is displayed on the first line.



7 For the conveyor station for which you are redoing the tracking frame setting, touch up the fixture as instructed in steps 4 through 13 in "I.5.4.1.1. In the case of a circular conveyor(P.283)".

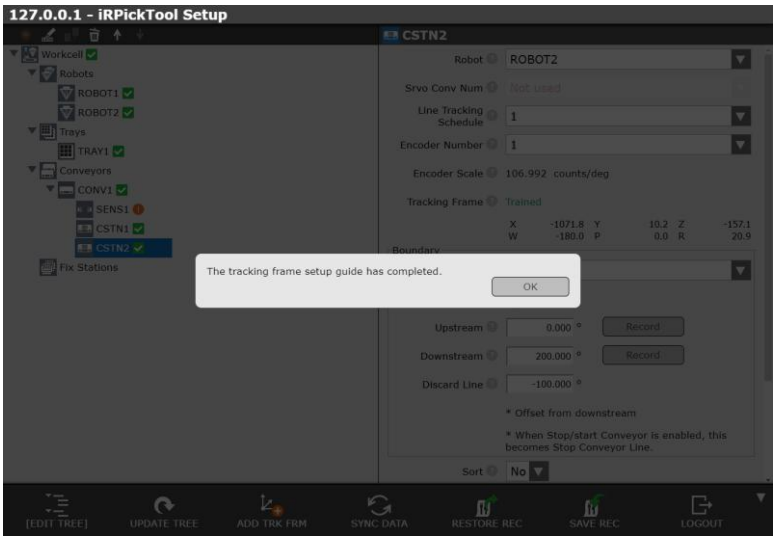
After auxiliary points are set, the following screen appears.

When you have proceeded to the step of setting the assistant point, the screen shown below is displayed.

8 Click [NEXT].



After a while, the following screen appears.



5.7. Setting up 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 [SETUP] and then [Frames] from the [MENU] key, instead of the iRPickTool setup screen. For details, see the “Setting a user coordinate system” subsection in the “Setting coordinate systems” section in the “Setting up the robot system” chapter in "OPERATOR’S MANUAL(Basic Operation)"(B-84464EN).

⚠CAUTION

User frame number can be selected at a fixed station is from 1 to 9.

[Buffer]

To use the tray as the buffer, set it to [Enable]. 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 in which parts are picked out of 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.

5.8. KAREL program

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.

5.8.1. 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.

5.8.1.1. 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 are any inconsistencies in the settings of the robot controllers within the workcell.
- Checks whether there is any invalid setting.
- Clears the part information from all the conveyor stations corresponding to the calling robot controller. The program executes the same processing as PKCSCLRQUE described later for all the conveyor stations corresponding to the calling robot controller. For details, see “I.5.8.5.2. PKCSCLRQUE.PC(P.351)”.

  If the workcell includes a buffer conveyor, the program executes the same processing as PKCVCLRBUF described later for all forward direction conveyors. For details, see “I.8.15.3.1. PKCVCLRBUF.PC(P.691)”.
- Starts the monitoring of the trigger condition that is set for each sensor of the calling robot controller. The program executes the same processing as PKSNSTART described later. For details, see “I.5.8.4.5. PKSNSTART.PC(P.348)”.

There is no argument.

Example:

```
CALL PKWCSTART
```

5.8.1.2. 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 robot controller. The program executes the same processing as PKSNSTOP described later

There is no argument.

Example:

```
CALL PKWCEND
```

5.8.1.3. PKWCCHECK.PC

This program checks whether there are any inconsistencies in the settings of the robot controllers within the workcell and whether there are any invalid settings. It is called instead of PKWCSTART when PKCSCLRQUE and PKSNSTART are called separately.

There is no argument.

Example:

```
CALL PKWCCHECK
```

5.8.1.4. PKWCRSTPRDST.PC

This program resets Production Status of iRPickTool. For details on Production Status, see "I.8.2. Checking the production status(P.567)".

There is no argument.

Example:

```
CALL PKWCRSTPRDST
```

5.8.1.5. PKPRGRGETEN.PC

This program returns whether pre-grouping can be used or not in this robot controller.

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 is stored in R[7] whether pre-grouping can be used in the robot controllers in which this program is being executed.

```
CALL PKPRGRGETEN(PreGrp Enb Reg=7)
```

5.8.1.6. 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 tree view.

- 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 robot controllers, then you need to run the program in each robot controller.
- This program only clears 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
```

5.8.1.7. PKABO.PC

This program is used to abort execution of TP programs and KAREL programs in each robot controller in the 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 robot controller to abort programs in all the robot controllers defined in the tree view.
- Important tasks such as sensor tasks will not be aborted. To stop a sensor task, refer to "I.5.8.4.6. PKSNSTOP.PC(P.349)" or "I.5.8.1.2. PKWCEND.PC(P.333)".

- This program uses Ethernet to communicate with the other controllers. Therefore, if a controller is offline, it will not be able to abort programs in that controller.

There is no argument.

Example:

```
CALL PKABO
```

5.8.1.8. PKLABO.PC

This program is used to abort all the TP programs and KAREL programs currently running in the robot controllers in which this program is executed, excluding important tasks such as sensor tasks.

There is no argument.

Example:

```
CALL PKLABO
```

5.8.1.9. 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, select one of the following options as the operation type.

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.

Usage:

To post error IRPK-232 “Invalid Position Register” as an error that should pause the robot

```
CALL PKPOSTERR(Error Code=127232, PAUSE)
```

5.8.1.10. PKCOPYPR2POS.PC

This program copies the value of a Position Register to the position variable of a TP program.

Argument 1:	Specify the number of the copy source Position Register.
Argument 2:	Specify the TP program name.
Argument 3:	Specify the number of the copy destination position variable.

Argument 4: Specify a group number.

Usage:

The value of PR [11] is copied to P[2] of MAIN1.TP regarding group 1.

```
CALL PKCOPYPR2POS(PR Num=11,Program Name='MAIN1',Position Num=2,Group Num=1)
```

5.8.1.11. PKWCGETACTRC.PC

This program copies an active recipe id.

Argument 1: Specify the register number to store the recipe ID of an active recipe.

Example:

```
CALL PKWCGETACTRC(RecipeID Reg=1)
```

5.8.1.12. IRVGETVRWRKID.PC

This program returns the part ID stored in the specified Vision Register.

Argument 1: Specify the number of the Vision Register.

Argument 2: Specify the register to return the part ID to.

Usage:

Part ID data is acquired from VR[1]. The data is stored in R[1].

```
CALL IRVGETVRWRKID(VR Index=1, WRK ID Reg=1)
```

5.8.1.13. PKSETSYNCINT.PC

When synchronizing data, all the iRPickTool settings are sent to each robot controller. At that time, the temporary increase in communication load may cause data synchronization failures. For example, a RIPE-019 warning may occur in another robot controller. Such failures can be avoided by using this program to increase the interval of data communication.

In addition, data synchronization is also performed when restoring, importing, or switching recipes. This program can be used when similar problems occur during these recipe operations.

However, the time required for data synchronization will increase. The longer the time interval specified here and the larger the current maximum number of objects set, the longer data synchronization takes.

When using this program, it is necessary to enable the function by specifying 2(Enable data synchronization communication time interval adjustment) for argument 1 of "I.5.8.1.14. PKENBSPFUNC.PC(P.337)". This function is enabled by default.

NOTE

This function is supported in 7DH1/16 (V10.10P/16) and later versions.

Argument 1: Specify which data communication interval to change.

- 1(Data synchronization on the settings screen):
  Specify when changing the data communication interval during data synchronization on the settings screen.
  If there are multiple robot controllers within the workcell, use this program in the robot controller in which the settings screen is opened.

- 2(Synchronization during recipe restoration):
  Specify when changing the data communication interval during data synchronization during recipe restoration (import, switching).
  If there are multiple robot controllers within the workcell, use this program in the robot controller assigned as the Recipe Manager. By default, the Recipe Manager is the ring leader robot controller within the RIPE robot ring network.

Argument 2: Set the time interval for data communication. You can specify a time interval between 0 milliseconds and 32 milliseconds. The default value is 8 milliseconds.

Example:

Change the data communication interval during data synchronization to 8 milliseconds.

```
CALL PKSETSYNCINT 1(Data synchronization on the settings screen), time interval (ms)=8)
```

5.8.1.14. PKENBSPFUNC.PC

This program enables a specific function related to data communication. To disable a function, use "I.5.8.1.15. PKDSBSPFUNC.PC(P.337)".

NOTE

This function is supported in 7DH1/16 (V10.10P/16) and later versions.

Argument 1: Specify the function to be enabled.

- 1(Enable tray communication volume reduction): Enables the function to reduce the communication volume when transmitting tray data from the upstream conveyor station to the downstream conveyor station. This function is disabled by default.

  ⚠CAUTION

  Tray communication volume reduction cannot be used in combination with tray redetection. For details on tray redetection, refer to "I.8.12. Tray redetection(P.653)".

- 2(Enable data synchronization communication time interval adjustment): Enables the function to adjust the data communication interval during data synchronization and recipe restoration (import, switching) on the settings screen. This function is enabled by default. Adjust the time interval using "I.5.8.1.13. PKSETSYNCINT.PC(P.336)".

Example:

Enable the function to adjust the data communication interval during data synchronization.

```
CALL PKENBSPFUNC(2(Enable data synchronization communication time interval adjustment))
```

5.8.1.15. PKDSBSPFUNC.PC

This program disables a specific function related to data communication. To enable a function, use "I.5.8.1.14. PKENBSPFUNC.PC(P.337)".

NOTE

This function is supported in 7DH1/16 (V10.10P/16) and later versions.

Argument 1: Specify the function to be disabled.

- 1(Disable tray communication volume reduction): Disables the function to reduce the communication volume when transmitting tray data from the upstream conveyor station to the downstream conveyor station. This function is disabled by default.
- 2(Disable data synchronization communication time interval adjustment): Disables the function to adjust the data communication interval during data synchronization and recipe restoration (import, switching) on the settings screen. This function is enabled by default. When this function is disabled, data is communicated without stopping for an interval.

Example:

Disable the function to adjust the data communication interval during data synchronization.

```
CALL PKDSBSPFUNC(2(Disable data synchronization communication time interval adjustment))
```

5.8.2. Robot-related programs

The two letters following [PK] are [RB] namely, the names of these programs begin with these four letters, [PKRB].

5.8.2.1. PKRBGETID.PC

This program gets the ID of the robot from a group number.

Argument 1:	Specify a 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.

Usage:

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)
```

5.8.2.2. PKRBGETSFLG.PC

This program gets the FLAG number in Skip outbound motion of the robot from the robot ID.

Argument 1:	Specify the ID of a Robot.
Argument 2:	Specify the register number in which the FLAG number is to be returned.

Usage:

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)
```

5.8.2.3. PKRBSETSVDLY

This program sets a servo delay and dynamically adjusts the tracking operation.

For details, see “Dynamic error fine adjustment parameters” in “Advanced functions” in "Line Tracking OPERATOR'S MANUAL" (B-83474EN).

Argument 1:	Specify the ID of a Robot.
Argument 2:	Specify a servo delay value (ms).

Usage:

Set a servo delay of 1.3 ms to tracking schedule 1.

```
CALL PKRBSETSVDLY(1, 1.3)
```

5.8.2.4. PKRBSETSKPADJ

This program sets the skip adjustment time to define the time margin to avoid “passing the target position” with the skip outbound motion.

For details, see “Skip outbound motion” in “Advance functions” in "Line Tracking OPERATOR'S MANUAL" (B-83474EN).

Argument 1:	Specify the ID of a Robot.
Argument 2:	Specify the skip adjustment time.

Usage:

Set a skip adjustment time of 100 milliseconds to tracking schedule 1.

```
CALL PKRBSETSKPADJ(1, 100)
```

5.8.3. Conveyor-related programs

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

5.8.3.1. PKCVSETLBD.PC

This program changes the load balance of each conveyor station on a conveyor.

This setting is reflected immediately.

When there are multiple robots, if you execute this in one robot, the setting is reflected in all robots.

Argument 1:	Specify the name of the conveyor using a character string.

NOTE

The number of displayed options is the maximum number of objects (32). In other words, those up to CONV32 are displayed as conveyor names.

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. The value of [Bypass] is set to 0. Do not specify a model ID. 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. Be sure to call this KAREL program at the beginning of the main program, that is, before KAREL programs PKWCSTART, PKSNSTART, or PKWCCHECK, so that the changed parameters are reflected correctly.
2. When you execute this program with the setup page open, the parameter to be changed might not be updated immediately. But they are updated automatically after a few seconds.
3. When calling KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKSNSETDBNC.PC, PKSNSETDURTN.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC) when using multiple robot controllers, be aware of the following points.
   - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all robot controllers in order to wait for the completion of the parameter changes. Set the wait time to 5 seconds or so. If parameter changes have not completed by the time PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
   - If you call more than one of the listed KAREL programs for setting parameters, do so from the same robot controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other robot controllers also. If the listed KAREL programs for setting parameters are called in multiple robot controllers, the following phenomena may occur:
     - IRPK-079 “Need to synchronize data” occurs when PKWCSTART or PKWCCHECK is called.
     - IRPK-082 “The setting cannot be changed because the data is different” will occur in one of the robot controllers.
4. When using a recipe and a KAREL program for setting parameters together, first change the recipe and then call the KAREL program.

5.8.3.2. PKCVSETTRNAM.PC

This program changes a tray in a conveyor. When there are multiple robots, if you execute this in one robot, the setting is reflected in all robots.

Argument 1: Specify the name of the conveyor using a character string.

NOTE

- The number of displayed options is the maximum number of objects (32). In other words, those up to CONV32 are displayed as conveyor names.
- When using PKCVSETTRNAM.PC for a buffer conveyor, specify Forward Direction Conveyor for Argument 1.

Argument 2: Specify the name of the tray using a character string.

NOTE

The number of displayed options is the maximum number of objects (16). In other words, those up to TRAY16 are displayed as tray names.

Example:

The following instruction changes the tray selected in “CONV1” to “TRAY2”.

```
CALL PKCVSETTRNAM (‘CONV1’,’TRAY2’)
```

⚠CAUTION

1. Please call this KAREL program at the beginning of the main program, that is, before KAREL programs PKWCSTART, PKWCCHECK, or PKSNSTART so that the changed parameters are reflected correctly.
2. When you execute this program with the setup page open, the parameter to be changed might not be updated immediately. But they are updated automatically after a few seconds.
3. When you use multiple robot controllers and call KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKSNSETDBNC.PC, PKSNSETDURTN.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC), be aware of the following points.
   - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all robot controllers in order to wait for the completion of the parameter changes. Set the wait time to 5 seconds or so. If parameter changes have not completed by the time PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.

⚠CAUTION

- If you call more than one of the listed KAREL programs for setting parameters, do so from the same robot controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other robot controllers also. If the listed KAREL programs for setting parameters are called in multiple robot controllers, the following phenomena may occur:
  - IRPK-079 “Need to synchronize data” occurs when PKWCSTART or PKWCCHECK is called.
  - IRPK-082 “The setting cannot be changed because the data is different” will occur in one of the robot controllers.

4. When using a recipe and a KAREL program for setting parameters together, first change the recipe and then call the KAREL program.

5.8.3.3. PKCVENBLB.PC

This program enables or disables the load balancing function. This setting is reflected immediately. When there are multiple robots, if you execute this in one robot, the setting is reflected in all robots.

Argument 1: Specify the name of the conveyor using a character string.

NOTE

The number of displayed options is the maximum number of objects (32). In other words, those up to CONV32 are displayed as conveyor names.

Argument 2: Specify “Enable” to enable the function or “Disable” to disable it.

NOTE

When the argument guidance is supported, select one of the following options as the operation type.

1. Enable
2. Disable

However, internally the sensor uses 1 for Enable and 0 for Disable.

So if argument guidance is not supported in your software version, specify 1 for Enable and 0 for Disable.

Example:

The following instruction enables the load balancing function for conveyor data CONV1.

```
CALL PKCVENBLB('CONV1',Enable)
```

⚠CAUTION

1. Be sure to call this KAREL program at the beginning of the main program, that is, before KAREL programs PKWCSTART, PKSNSTART, or PKWCCHECK, so that the changed parameters are reflected correctly.
2. 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 are carried 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 on the upstream side; for example, immediately after PKCSGETQUE is called.
3. When you execute this program with the setup page open, the parameter to be changed might not be updated immediately. But they are updated automatically after a few seconds.
4. When calling KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKSNSETDBNC.PC, PKSNSETDURTN.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC) when using multiple robot controllers, be aware of the following points.
   - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all robot controllers in order to wait for the completion of the parameter changes. Set the wait time to 5 seconds or so. If parameter changes have not completed by the time PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
   - If you call more than one of the listed KAREL programs for setting parameters, do so from the same robot controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other robot controllers also. If the listed KAREL programs for setting parameters are called in multiple robot controllers, the following phenomena may occur:
     - IRPK-079 “Need to synchronize data” occurs when PKWCSTART or PKWCCHECK is called.

⚠CAUTION

- IRPK-082 “The setting cannot be changed because the data is different” will occur in one of the robot controllers.

5. When using a recipe and a KAREL program for setting parameters together, first change the recipe and then call the KAREL program.

5.8.3.4. PKCVENBSC.PC

This program enables or disables the stop/start conveyor function for a specified conveyor station. This setting is reflected immediately. When there are multiple robots, if you execute this in one robot, the setting is reflected in all robots.

Argument 1: Specify the name of the conveyor using a character string.

NOTE

The number of displayed options is the maximum number of objects (32). In other words, those up to CONV32 are displayed as conveyor names.

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. Be sure to call this KAREL program at the beginning of the main program, that is, before KAREL programs PKWCSTART, PKSNSTART, or PKWCCHECK, so that the changed parameters are reflected correctly.
2. When you execute this program with the setup page open, the parameter to be changed might not be updated immediately. But they are updated automatically after a few seconds.
3. When calling KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKSNSETDBNC.PC, PKSNSETDURTN.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC) when using multiple robot controllers, be aware of the following points.
   - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all robot controllers in order to wait for the completion of the parameter changes. Set the wait time to 5 seconds or so. If parameter changes have not completed by the time PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
   - If you call more than one of the listed KAREL programs for setting parameters, do so from the same robot controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other robot controllers also. If the listed KAREL programs for setting parameters are called in multiple robot controllers, the following phenomena may occur:
     - IRPK-079 “Need to synchronize data” occurs when PKWCSTART or PKWCCHECK is called.
     - IRPK-082 “The setting cannot be changed because the data is different” will occur in one of the robot controllers.
4. When using a recipe and a KAREL program for setting parameters together, first change the recipe and then call the KAREL program.

5.8.3.5. PKCVGETNAME.PC

This program is used to get the name of a conveyor you defined in the tree view 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.

Usage:

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)
```

5.8.3.6. PKCVIDXDELQ.PC

This KAREL program can be used for conveyors for which [Servo Conveyor] is set to [On] on the conveyor setup screen.

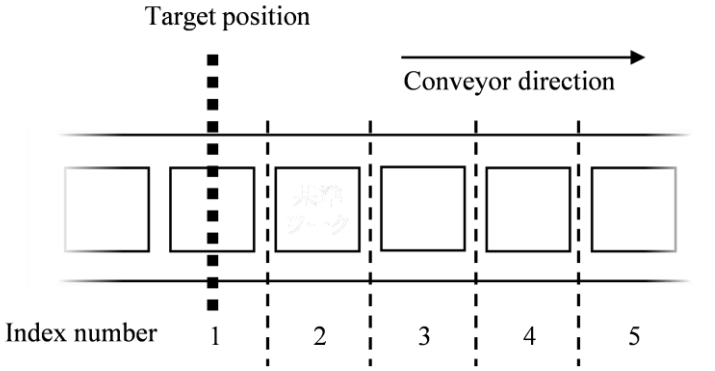
It deletes queue elements corresponding to a specified index. Multiple queue elements can be deleted at once.

Deleted queue elements are ignored by PKCSGETQUE, as well as by the load balance and product functions.

If the program fails to delete specified queue elements, it outputs the number of undeleted elements to the specified register.

Indexes are numbers that correspond to individual conveyor pitches.

The first index (target position) corresponds to the position where a workpiece is placed and a trigger is configured during target position setup. Subsequent indexes are assigned incrementally to each downstream pitch.



Argument 1:	Specify the name of the conveyor using a character string.
Argument 2:	Specify the start index from which queue elements are deleted, counting from the beginning. The first index corresponds to the target position and cannot be specified. Specify Index 2 or higher.
Argument 3:	Specify the number of elements to delete. Specify the number of elements to delete starting from the index specified in Argument 2 in the downstream direction.
Argument 4:	Specify the number of the register to which the number of undeleted queue elements is output.

Example:

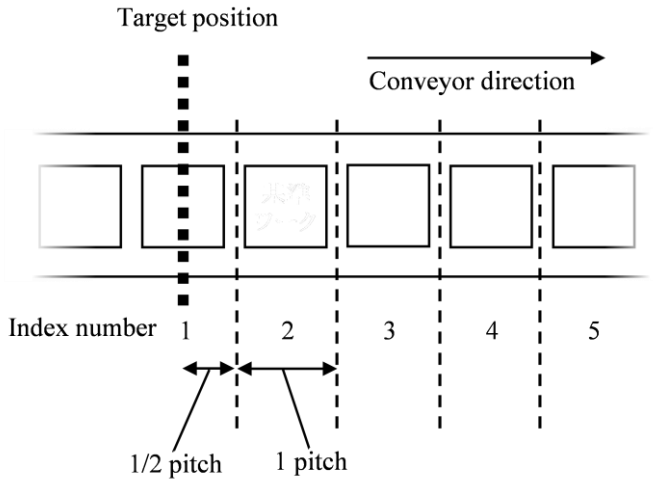
Deletes queue elements for three pitches starting from Index 2 for Conveyor Station 1.

The number of queue elements that could not be delete is stored in R[1].

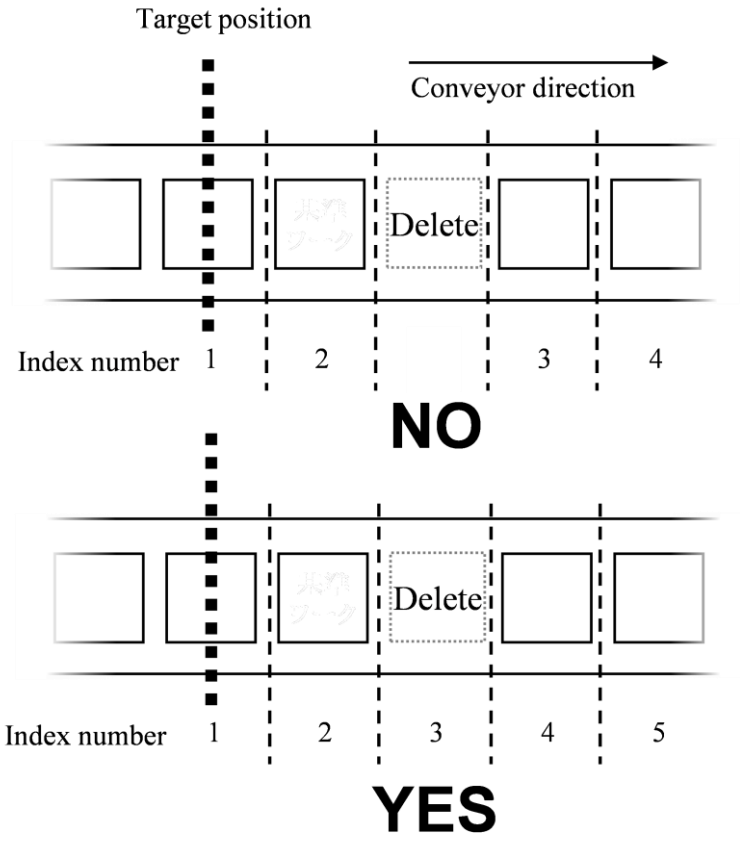
```
CALL PKCVIDXDELQ('CONV1',First Index=2,Num to Delete=3,Num Delete Reg=1)
```

NOTE

- Each index corresponds to a single pitch.
  However, Index 1 is offset by half a pitch downstream from the target position, and Indexes 2 and higher correspond to full pitches.



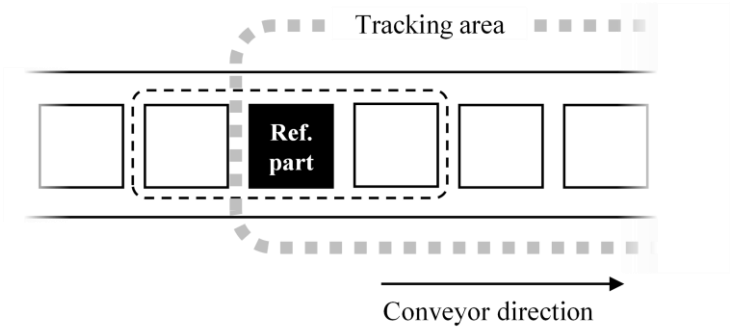
- Deleting queue elements does not shift the indexes.
  The positional relationship between indexes and conveyor pitches remains unchanged unless the target position or the servo conveyor pitch interval is modified.



- Deletion fails if any of the following conditions is met:
  - The specified queue elements have already been deleted.
  - The specified queue elements are within the tracking boundary.

  If Pick or Drop Multiple is enabled and the base part for the function is within the tracking boundary, deletion fails.
  For details on Pick or Drop Multiple, see "I.8.7.2.3. Pick or Drop Multiple(P.601)".

NOTE



5.8.3.7. PKCVGETCNVSPD.PC

This program acquires the conveyor speed in the tracking schedule.

Argument 1: Specify the tracking schedule number.

Argument 2: Specify a register number to store the conveyor speed.

Usage:

The speed of the conveyor with Line Tracking Schedule 1 is returned in R[10].

```
CALL PKCVGETCNVSPD(LNSCH Index=1,CV SPD Reg=10)
```

5.8.4. Sensor-related programs

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

5.8.4.1. PKSNENBSENS.PC

This program enable or disable a sensor. When there are multiple robots, if you execute this in one robot, the setting is reflected in all robots.

Argument 1: Specify the name of the sensor using a character string.

NOTE

The number of displayed options is the maximum number of objects (32). In other words, those up to SENS32 are displayed as sensor names.

Argument 2: Specify “Enable” to enable the function or “Disable” to disable it.

NOTE

When the argument guidance is supported, select one of the following options as the operation type.

1. Enable
2. Disable

However, internally the sensor uses 1 for Enable and 0 for Disable.

So if argument guidance is not supported in your software version, specify 1 for Enable and 0 for Disable.

Example:

The following instructions disable “SENS1”, enable “SENS2”, and start the sensors.

```
CALL PKSNENBSENS (‘SENS1’,Disable)
CALL PKSNENBSENS (‘SENS2’,Enable)
```

```
CALL PKWCSTART
```

CAUTION

1. Be sure to call this KAREL program at the beginning of the main program, that is, before KAREL programs PKWCSTART, PKSNSTART, or PKWCCHECK, so that the changed parameters are reflected correctly.
2. When you switch enable or disable after PKWCSTART or PKSNSTART, all sensors need to be enabled when the sensors start with PKWCSTART or PKSNSTART.
3. When you execute this program with the setup page open, the parameter to be changed might not be updated immediately. But they are updated automatically after a few seconds.
4. When calling KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKSNSETDBNC.PC, PKSNSETDURTN.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC) when using multiple robot controllers, be aware of the following points.
   - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all robot controllers in order to wait for the completion of the parameter changes. Set the wait time to 5 seconds or so. If parameter changes have not completed by the time PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
   - If you call more than one of the listed KAREL programs for setting parameters, do so from the same robot controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other robot controllers also. If the listed KAREL programs for setting parameters are called in multiple robot controllers, the following phenomena may occur:
     - IRPK-079 “Need to synchronize data” occurs when PKWCSTART or PKWCCHECK is called.
     - IRPK-082 “The setting cannot be changed because the data is different” will occur in one of the robot controllers.
5. When using a recipe and a KAREL program for setting parameters together, first change the recipe and then call the KAREL program.

5.8.4.2. PKSNSETVPNAM.PC

This program changes a vision process in a sensor. The vision process set for the sensor by [Find part by multiview] can also be changed. When there are multiple robots, if you execute this in one robot, the setting is reflected in all robots.

Argument 1: Specify the name of the sensor using a character string.

NOTE

The number of displayed options is the maximum number of objects (32). In other words, those up to SENS32 are displayed as sensor names.

Argument 2: Specify the name of the vision process using a character string.

Example:

The following instruction changes the vision process selected in “SENS1” to “VT2”.

```
CALL PKSNSETVPNAM (‘SENS1’,’VP Name=VT2’)
```

CAUTION

1. Be sure to call this KAREL program at the beginning of the main program, that is, before KAREL programs PKWCSTART, PKSNSTART, or PKWCCHECK, so that the changed parameters are reflected correctly.
2. When you execute this program with the setup page open, the parameter to be changed might not be updated immediately. But they are updated automatically after a few seconds.
3. When calling KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKSNSETDBNC.PC, PKSNSETDURTN.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC) when using multiple robot controllers, be aware of the following points.
   - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all robot controllers in order to wait for the completion of the parameter changes. Set the wait time to 5 seconds or so. If parameter changes have not completed by the time PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data”

⚠CAUTION

occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.

- If you call more than one of the listed KAREL programs for setting parameters, do so from the same robot controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other robot controllers also. If the listed KAREL programs for setting parameters are called in multiple robot controllers, the following phenomena may occur:
  - IRPK-079 “Need to synchronize data” occurs when PKWCSTART or PKWCCHECK is called.
  - IRPK-082 “The setting cannot be changed because the data is different” will occur in one of the robot controllers.

4. When using a recipe and a KAREL program for setting parameters together, first change the recipe and then call the KAREL program.

NOTE

To specify a position register for Argument 2, follow the procedure below:

1. Enter an appropriate character string in the argument.
2. Bring the cursor to the entered character string and press F4 [Select].
3. Select [String register].

5.8.4.3. PKSNSETVTOFS.PC

When you use [DI] + [Find part by vision], this program changes a trigger offset in a sensor. When there are multiple robots, if you execute this in one robot, the setting is reflected in all robots.

Argument 1: Specify the name of the sensor using a character string.

NOTE

The number of displayed options is the maximum number of objects (32). In other words, those up to SENS32 are displayed as sensor names.

Argument 2: Specify a value as the trigger offset.

Example:

The following instruction changes the trigger offset selected in “SENS1” to 100 mm.

```
CALL PKSNSETVTOFS(‘SENS1’,Trigger Offset=100)
```

⚠CAUTION

1. Be sure to call this KAREL program at the beginning of the main program, that is, before KAREL programs PKWCSTART, PKSNSTART, or PKWCCHECK, so that the changed parameters are reflected correctly.
2. When you execute this program with the setup page open, the parameter to be changed might not be updated immediately. But they are updated automatically after a few seconds.
3. When calling KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKSNSETDBNC.PC, PKSNSETDURTN.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC) when using multiple robot controllers, be aware of the following points.
   - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all robot controllers in order to wait for the completion of the parameter changes. Set the wait time to 5 seconds or so. If parameter changes have not completed by the time PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
   - If you call more than one of the listed KAREL programs for setting parameters, do so from the same robot controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other robot controllers also. If the listed KAREL programs for setting parameters are called in multiple controllers, the following phenomena may occur:
     - IRPK-079 “Need to synchronize data” occurs when PKWCSTART or PKWCCHECK is called.
     - IRPK-082 “The setting cannot be changed because the data is different” will occur in one of the robot controllers.
4. When using a recipe and a KAREL program for setting parameters together, first change the recipe and then call the KAREL program.

5.8.4.4. PKSNSETMTOFS.PC

When you use [DI] + [Find part by multiview], this program changes a trigger offset of each view in a sensor. When there are multiple robots, if you execute this in one robot, the setting is reflected in all robots.

Argument 1: Specify the name of the sensor using a character string.

NOTE

The number of displayed options is the maximum number of objects (32). In other words, those up to SENS32 are displayed as sensor names.

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 the trigger offsets of multi-view selected in “SENS1” to 100 and 200 mm.

```
CALL PKSNSETMTOFS(‘SENS1’,Trigger Offset 1=100,TriggerOffset 2=200)
```

CAUTION

1. Be sure to call this KAREL program at the beginning of the main program, that is, before KAREL programs PKWCSTART, PKSNSTART, or PKWCCHECK, so that the changed parameters are reflected correctly.
2. When you execute this program with the setup page open, the parameter to be changed might not be updated immediately. But they are updated automatically after a few seconds.
3. When calling KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKSNSETDBNC.PC, PKSNSETDURTN.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC) when using multiple robot controllers, be aware of the following points.
   - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all robot controllers in order to wait for the completion of the parameter changes. Set the wait time to 5 seconds or so. If parameter changes have not completed by the time PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
   - If you call more than one of the listed KAREL programs for setting parameters, do so from the same robot controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other robot controllers also. If the listed KAREL programs for setting parameters are called in multiple robot controllers, the following phenomena may occur:
     - IRPK-079 “Need to synchronize data” occurs when PKWCSTART or PKWCCHECK is called.
     - IRPK-082 “The setting cannot be changed because the data is different” will occur in one of the robot controllers.
4. When using a recipe and a KAREL program for setting parameters together, first change the recipe and then call the KAREL program.

5.8.4.5. PKSNSTART.PC

This program starts the monitoring of the trigger condition that is set for each sensor of the robot controller that calls the program. If the trigger condition is met, the specified action is automatically performed.

If this KAREL program is executed in a robot controller that does not have a sensor, no operation will occur.

There is no need to use the program if you will call PKWCSTART.

There is no argument.

Example:

```
CALL PKSNSTART
```

5.8.4.6. PKSNSTOP.PC

This program stops the monitoring of the trigger condition that is set for each sensor of the robot controller that calls the program.

There is no need to use this program if you will call PKWCEND.

There is no argument.

Example:

```
CALL PKSNSTOP
```

5.8.4.7. PKSNSETDBNC.PC

This program changes the debounce value set for the sensor. When there are multiple robots, if you execute this in one robot, the setting is reflected in all robots.

Argument 1: Specify the name of the sensor using a character string.
Argument 2: Specify the debounce value.

Example:

The following instruction changes the debounce value for “SENS1” to 100.

```
CALL PKSNSETDBNC(‘SENS1’,Debounce=100)
```

⚠CAUTION

1. Be sure to call this KAREL program at the beginning of the main program, that is, before KAREL programs PKWCSTART, PKSNSTART, or PKWCCHECK, so that the changed parameters are reflected correctly.
2. When you execute this program with the setup page open, the parameter to be changed might not be updated immediately. But they are updated automatically after a few seconds.
3. When calling KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKSNSETDBNC.PC, PKSNSETDURTN.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC) when using multiple robot controllers, be aware of the following points.
   - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all robot controllers in order to wait for the completion of the parameter changes. Set the wait time to 5 seconds or so. If parameter changes have not completed by the time PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
   - If you call more than one of the listed KAREL programs for setting parameters, do so from the same robot controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other robot controllers also. If the listed KAREL programs for setting parameters are called in multiple robot controllers, the following phenomena may occur:
     - IRPK-079 “Need to synchronize data” occurs when PKWCSTART or PKWCCHECK is called.
     - IRPK-082 “The setting cannot be changed because the data is different” will occur in one of the robot controllers.
4. When using a recipe and a KAREL program for setting parameters together, first change the recipe and then call the KAREL program.

5.8.4.8. PKSNSETDURTN.PC

This program changes the debounce value set for the sensor. When there are multiple robots, if you execute this in one robot, the setting is reflected in all robots.

Argument 1: Specify the name of the sensor using a character string.
Argument 2: Specify a value for duration.

Example:

The following instruction changes the duration value for “SENS1” to 100.

```
CALL PKSNSETDURTN(‘SENS1’,Duration=100)
```

⚠CAUTION

1. Be sure to call this KAREL program at the beginning of the main program, that is, before KAREL programs PKWCSTART, PKSNSTART, or PKWCCHECK, so that the changed parameters are reflected correctly.
2. When you execute this program with the setup page open, the parameter to be changed might not be updated immediately. But they are updated automatically after a few seconds.
3. When calling KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKSNSETDBNC.PC, PKSNSETDURTN.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC) when using multiple robot controllers, be aware of the following points.
   - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all robot controllers in order to wait for the completion of the parameter changes. Set the wait time to 5 seconds or so. If parameter changes have not completed by the time PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
   - If you call more than one of the listed KAREL programs for setting parameters, do so from the same robot controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other robot controllers also. If the listed KAREL programs for setting parameters are called in multiple robot controllers, the following phenomena may occur:
     - IRPK-079 “Need to synchronize data” occurs when PKWCSTART or PKWCCHECK is called.
     - IRPK-082 “The setting cannot be changed because the data is different” will occur in one of the robot controllers.
4. When using a recipe and a KAREL program for setting parameters together, first change the recipe and then call the KAREL program.

5.8.5. Conveyor station-related programs

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

5.8.5.1. PKCSGETID.PC

This program gets the ID of the conveyor station to which the robot that calls the program belongs in a specified conveyor.

Argument 1:	Specify a conveyor name.

NOTE

The number of displayed options is the maximum number of objects (32). In other words, those up to CONV32 are displayed as conveyor names.

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.

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)
```

5.8.5.2. PKCSCLRQUE.PC

This program clears the part information from a specified conveyor station. All the part information that is present in the conveyor station when the program is executed is cleared. Normally, call this program only once at system startup. There is no need to use the program if you will call PKWCSTART.

Argument 1: Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.

Usage:

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])
```

5.8.5.3. PKCSGETQUE.PC

This program gets the information of a part from a conveyor station. The acquired part information is stored in a Vision Register.

If the part whose information is acquired is within the tracking area, the value of the encoder is set as the trigger for the tracking motion. The robot waits for a specified time before the part whose information is acquired reaches the tracking area.

Argument 1: Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.

Argument 2: Normally, specify 1.
If you want the information of successive parts to be allocated regardless of the load balance for conveyor stations that is set during the conveyor setup, specify 2 or a greater value. For details, see "I.5.9.2.2. Pick program(P.370) ".

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 indefinitely. 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 information of the acquired part 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 register storing 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 "I.7. Setting example appropriate to the usage(P.522)".

Example:

The following instruction acquires the information on unprocessed parts at conveyor station 1.

IDs are stored in R[11]. The part information is stored in VR[1], and the processing status in R[2]. If cell 5 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 cell 1. So when you teach the robot position to a tray, you must make the taught position conform to the position of 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 cannot notify the conveyor station of the processing result described later.

5.8.5.4. PKCSGETQCELL.PC

This program gets the information of a cell from a conveyor station. 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 indefinitely. 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 register storing 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. A value is stored in the specified register only when the processing status is 0.

Example:

When the ID of conveyor station 1 is stored in R[11], the following instruction stores the information about cell 5 from conveyor station 1 in VR[1], the processing status in R[2], and the number of unfilled cells that should be filled before cell 5 in R[6]. If cell 5 has not reached the tracking area, the robot waits 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 cell 1. So when you teach the robot position to a tray, you must make the taught position conform to the position of cell 1.
2. If the processing status is 1, the cell information has not actually been allocated. Even if you immediately call PKCSGETQCELL described next, you may get the information on cells in different trays. Also, if the processing status is 1, PKCSACKQUE described next cannot notify the conveyor station of the processing result.

5.8.5.5. PKCSACKQUE.PC

After the processing status of PKCSGETQUE or PKCSGETQCELL becomes 0 and the part information is allocated, this program notifies the conveyor station of the processing that has been performed for the part.

Argument 1: Specify the ID of a conveyor station.
The ID can be acquired using PKCSGETID described earlier.

Argument 2: Specify one of the following values to indicate the processing result.

- 1: Success
  Specify this value when the part has been handled normally (e.g., picked or placed in a cell of the tray).
- 2: Return
  Specify this value when the part is returned to the conveyor station before being handled.
- 3: Skip
  Specify this value when the part has not been handled as scheduled.
  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.
  This applies, for example, when the pickup check is unacceptable for some reason such as because the part drops after being picked. In this case, since the robot has moved the part, a downstream robot cannot handle it properly even if the information of that part is returned to the conveyor station. Therefore, the part information is deleted.
  For cells in the tray, all other pending cells in the same tray are also be deleted.

If the stop/start conveyor function is enabled during the conveyor station setup and the conveyor is stopped because the part fails to be handled in time, the conveyor starts to move after PKCSACKQUE is executed with the processing result set to anything other than 2.

Usage:

The following instruction returns that the processing has succeeded. The ID is stored in R[11].

```
CALL PKCSACKQUE(CStn ID=R[11:CSTN1 ID],Success)
```

5.8.5.6. PKCSPUTQUE.PC

This program inputs part information to a conveyor station. Normally, PKCSPUTQUE.PC is not used. Use this program when you want to input part information to a conveyor station using a unique program that does not use the sensor function included with 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 register number for storing the result of the duplicate check. If the information is input with no duplicate parts, 0 is stored in the specified numeric register. If there are any duplicate parts and the information fails to be input, 1 is stored.

Example:

This program inputs part information to conveyor station 1.

Part information whose ID is input in R[11] is individually stored in VR[3]. The result of the duplicate check is returned to R[21].

```
CALL PKCSPUTQUE(CStn ID=R[11:CSTN1 ID],Offset VR=3,OverlapStat Reg=21)
```

5.8.5.7. PKCSPUTQUE2.PC

This program inputs part information to a conveyor station. Normally, PKCSPUTQUE2.PC is not used. Use this program when you want to input part information to a conveyor station using a unique program that does not use the sensor function included with iRPickTool. Compare this program to the PKCSPUTQUE KAREL program; the input arguments can be freely assigned and are not limited to a specified Vision Register

Argument 1:	Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.
Argument 2:	Specify the encoder count. The count can be acquired using the TP instruction VR[].ENC.
Argument 3:	Specify the model ID for the part.
Argument 4:	Specify the number of the Position Register that stores the position of the found part.
Argument 5:	Specify the number of the Position Register that stores the offset to the found part.
Argument 6:	Specify the starting index of the numeric register in which the measurements are stored.
Argument 7:	Specify the number of measurements to be stored in numeric registers. Measurements are stored in numeric registers in order, starting at the index specified in Argument 6.
Argument 8:	Specify the numeric register number for storing the result of the duplicate check. If the information is input with no duplicate parts, 0 is stored in the specified numeric register. If there are any duplicate parts and the information fails to be input, 1 is stored.

Example:

This program inputs part information to conveyor station 1.

Part information whose Conveyor Station ID is in R[11], encoder count is in R[12], Model ID is in R[13], part position is in PR[1], part offset is in PR[2]. The three measurements are stored in R[R[6]], R[R[6]+1] and R[R[6]+2]. The result of the duplicate check is returned to R[21].

```
CALL PKCSPUTQUE2(CStn ID=R[11:CSTN1 ID],Enc Cnt=R[12],Model ID=R[13],Fnd Pos PR=1,Ofst PR=2,Meas Reg=R[6],Meas Quantity=3,OverlapStat Reg=21)
```

5.8.5.8. PKCSGETTIME.PC

This program returns the estimated time required before a part to be picked up according to a specified order arrives at the upstream boundary of the tracking area. The estimated time is expressed in milliseconds. To estimate the time, the conveyor speed at the time when this function is called is used. If the part has not yet passed the upstream boundary of the tracking area, a positive value is stored. If it has passed the boundary, a negative value is stored. When the conveyor is stopped, the program returns 2147483646 if the part is upstream of the upstream boundary of the tracking area. The program returns -2147483647 if the part is downstream of the upstream boundary of the tracking area. In addition, the program also obtains information on the part to be picked up according to the specified order.

This KAREL program cannot be used from the time PKCSGETQUE returns 0 as the processing status until PKCSACKQUE is called. Calling the program during this period causes the error “No ACKQUE after the last GETQUE”. This KAREL program does not support circular conveyors.

NOTE

PKCSGETTIME.PC cannot be used in the following cases.

- “Circle” is set in [Tracking Type] for the conveyor
- [Clearance check] is used for the conveyor

Argument 1:	Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.
Argument 2:	Specify the order. For example, if you specify 3, the program returns the estimated time for the part whose information is to be acquired by the third subsequent PKCSGETQUE program. Since it is based on the queue information available when this program is executed, the order may change if the robot misses a part or detects a new one.

Argument 3: Normally, specify 1.
When a robot gripper capable of holding multiple parts is used and you want the robot to consecutively pick as many parts as the number of parts that its gripper can hold, specify which part to pick next. For example, if the gripper can hold four parts and it is now holding two parts, specify 3.

Argument 4: Normally, specify 1.
When a robot gripper capable of holding multiple parts is used and you want the robot to consecutively pick as many parts as the number of parts that its gripper can hold, specify the number of parts that the gripper can hold. For example, if the gripper can hold four parts, specify 4.

Argument 5: Specify the number of the register to store the estimated time.

Argument 6: Specify the number of the Vision Register to store the part information.

Argument 7: Specify the number of register storing 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.

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,Consec Flag=2,Quantity=4,
Time Reg=13,Offset VR=5,Stat Reg=12)
```

5.8.5.9. PKCSGETDIST.PC

This program returns the distance from the part to be picked up according to a specified order to the upstream boundary of the tracking area. The distance is expressed in millimeters. In addition, the program also obtains information on the part to be picked up according to the specified order.

This KAREL program cannot be used from the time PKCSGETQUE returns 0 as the processing status until PKCSACKQUE is called. Calling the program during this period causes the error “No ACKQUE after the last GETQUE”. This KAREL program does not support circular conveyors.

NOTE

PKCSGETDIST.PC cannot be used in the following cases.

- “Circle” is set in [Tracking Type] for the conveyor
- [Clearance check] is used for the conveyor

Argument 1: Specify the ID of a conveyor station.
The ID can be acquired using PKCSGETID described earlier.

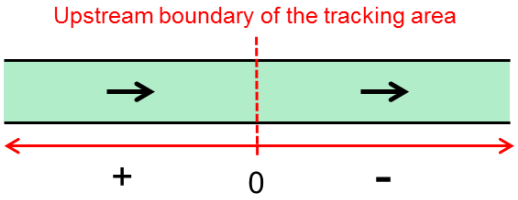
Argument 2: Specify the order.
For example, if you specify 3, the program returns the distance for the part whose information is to be acquired by the third subsequent PKCSGETQUE program. Since it is based on the queue information available when this program is executed, the order may change if the robot misses a part or detects a new one.

Argument 3: Normally, specify 1.

When a robot gripper capable of holding multiple parts is used and you want the robot to consecutively pick as many parts as the number of parts that its gripper can hold, specify which part to pick next.
For example, if the gripper can hold four parts and it is now holding two parts, specify 3.

Argument 4: Normally, specify 1.
When a robot gripper capable of holding multiple parts is used and you want the robot to consecutively pick as many parts as the number of parts that its gripper can hold, specify the number of parts that the gripper can hold. For example, if the gripper can hold four parts, specify 4.

Argument 5: Specify the number of the register to store the distance to the upstream boundary of the tracking area.
In the case of farther upstream than the upstream boundary of the tracking area, the sign is positive. In the case of farther downstream than the 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 register storing 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.

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), 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,Consec Flag=2,Quantity=4,
Time Reg=15,Offset VR=6,Stat Reg=14)
```

5.8.5.10. 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) × 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 [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 [Sort] is enabled, the upstream boundary is the position offset downstream by [X Tolerance] set in [Sort] from the discard line of the preceding conveyor station.

Expected number of parts

Of those parts located within the range used for the calculation, the expected number of parts refers to the number of parts that are expected to be processed by the conveyor station based on the load balance setting of the conveyor, the setting of [Sort] of the conveyor station, the layer and initial status settings of the tray, and the current status. The value is calculated not based on the conveyance capacity of the robot, but from the proportion that should be processed by the relevant robot based on the load balance setting and actual supply amount. In reality, however, the robot may not be able to process the full number of workpieces calculated as the expected number.

For example, when a load balance of 50% is set for the most upstream conveyor station, four of seven preceding parts have been picked, and three parts are located within the range used for the calculation, then the expected number of parts is 1 ((7 + 3) ÷ 2 - 4).

Distance to the conveyor station

The distance between the upstream and downstream boundaries of the range used for the calculation of the expected number of parts is defined as the distance to the conveyor station.

NOTE

PKCSGETPFRT.PC cannot be used in the following cases.

- “Circle” is set in [Tracking Type] for the conveyor
- [Clearance check] is used for the conveyor

Argument 1:	Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.
Argument 2:	Specify the number of the register to store the expected part flow rate.

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)
```

5.8.5.11. PKCSGETNPRT.PC

This program returns the number of parts that are held by a specified conveyor station. This is the number of parts detected with the sensor or the number of parts between the discard line of the upstream conveyor station and that of the current conveyor station. However, the following parts are excluded:

- Parts acquired with PKCSGETQUE for which PKCSACKQUE is not called yet
- Parts for which PKCSGETQUE is being called and whose arrival to the tracking area is waited for
- Parts that are determined not to be processed in the current conveyor station based on the load balance setting of the conveyor, the setting of [Sort] of the conveyor station, the layer and initial status settings of the tray, and the current status when PKCSGETQUE is called
- Parts for which 1 (Success), 3 (Skip), or 4 (Remove) is specified with PKCSACKQUE

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.

Usage:

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)
```

⚠CAUTION

PKCSGETNPRT cannot be used for buffer conveyors. If PKCSGETNPRT is used for a buffer conveyor, IRPK-065 “Missing my conveyor station” occurs.

5.8.5.12. PKCSSETDSLIN.PC

This program changes a discard line of the specified conveyor station. This setting is reflected immediately. When there are multiple robots, if you execute this in one robot, the setting is reflected in all robots.

Argument 1:	Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.
Argument 2:	Specify the discard line value.

Usage:

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. Be sure to call this KAREL program at the beginning of the main program, that is, before KAREL programs PKWCSTART, PKSNSTART, or PKWCCHECK, so that the changed parameters are reflected correctly.
2. When you execute this program with the setup page open, the parameter to be changed might not be updated immediately. But they are updated automatically after a few seconds.
3. When calling KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKSNSETDBNC.PC, PKSNSETDURTN.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC) when using multiple robot controllers, be aware of the following points.
   - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all robot controllers in order to wait for the completion of the parameter changes. Set the wait time to 5 seconds or so. If parameter changes have not completed by the time PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
   - If you call more than one of the listed KAREL programs for setting parameters, do so from the same robot controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other robot controllers also. If the listed KAREL programs for setting parameters are called in multiple robot controllers, the following phenomena may occur:
     - IRPK-079 “Need to synchronize data” occurs when PKWCSTART or PKWCCHECK is called.
     - IRPK-082 “The setting cannot be changed because the data is different” will occur in one of the robot controllers.
4. When using a recipe and a KAREL program for setting parameters together, first change the recipe and then call the KAREL program.

5.8.5.13. PKCSGETENC.PC

This program returns the encoder ID which is used in the specified conveyor station.

Argument 1:	Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.
Argument 2:	Specify the number of the register to store the encoder ID.

Usage:

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)
```

5.8.5.14. PKCSGETTRID.PC

This program returns the tray ID which is used in the specified conveyor station.

Argument 1:	Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described earlier.
Argument 2:	Specify the number of the register to store the tray ID.

Usage:

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)
```

5.8.5.15. 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 the ID of a conveyor station.

Usage:

To clear any pending acks for conveyor station whose ID is 1, use the following instruction:

```
CALL PKCSCLRACK(CStn ID=1)
```

5.8.5.16. PKCSGETDSLIN.PC

This program returns a discard line of the specified conveyor station.

Argument 1:	Specify the ID of a conveyor station. This ID can be acquired by using the above-mentioned PKCSGETID.
Argument 2:	Specify the register where the discard line value is stored.

Usage:

The discard line in conveyor station 1 is retrieved to R[2].

```
CALL PKCSGETDSLIN(CStn ID=1, DS Line Reg=2)
```

5.8.5.17. PKCSGETPINFO.PC

This program returns information about workpieces that passed through a conveyor station.

Argument 1:	Specify the ID of a conveyor station. This ID can be acquired by using the above-mentioned PKCSGETID.
Argument 2:	Specify the register that stores the total number of workpieces that passed through the conveyor station.
Argument 3:	Specify the register that stores the number of workpieces that were picked after passing through the conveyor station.

Argument 4: Specify the register that stores the number of workpieces that were missed after passing through the conveyor station.

Argument 5: Specify the register that stores the rate of workpieces that were picked after passing through the conveyor station (transfer rate, pieces/minute).

Argument 6: Specify the register that stores the flow rate (pieces/minute) of workpieces that passed through the conveyor station.

Argument 7: (optional) Specify a model ID.
If the argument is omitted, all workpieces that pass through the conveyor station are counted. In other cases, only workpieces with the specified model ID are counted.
Note that this argument can only be specified when [For each model ID] is selected as the conveyor's load balance option.

Example:

For the conveyor station specified by R[1], this example stores the number of passing workpieces in R[2], passing and picked workpieces in R[3], passing and missed workpieces in R[4], the transfer rate in R[5], and the workpiece flow rate in R[6].

```
CALL PKCSGETPINFO (Conveyor Station ID=R[1],Total Part Reg=2,Picked Part Reg=3,Missed Part Reg=4,Transfer Reg=5,Part Flow Rate Reg=6)
```

⚠CAUTION

PKCSGETPINFO cannot be used for buffer conveyors. If PKCSGETPINFO is used for a buffer conveyor, IRPK-065 "Conveyor station for this controller cannot be found" occurs.

5.8.6. Fixed station-related programs

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

5.8.6.1. PKFSGETID.PC

This program gets the ID of a specified fixed station.

Argument 1: Specify a fixed station name.

NOTE

35 options are displayed. In other words, up to FSTN35 are displayed as fixed station names.

Argument 2: Specify the number of the register to store the ID of the fixed station.

Example:

The following instruction stores the ID of fixed station 1 in R[12].

```
CALL PKFSGETID('FSTN1',FStn ID Reg=12)
```

5.8.6.2. PKFSPUTQUE.PC

This program places a new tray on a specified fixed station.

To specify the initial status of the placed tray ("Empty" or "Full"), use the fixed station setup screen.

To put another tray after all the cells in the placed tray are processed, call this KAREL program again.

Argument 1: Specify the ID of a fixed station.
The ID can be acquired by PKFSGETID described earlier.

Usage:

A new tray is placed on fixed station 1.
The ID is stored in R[12].

```
CALL PKFSPUTQUE(FStn ID=R[12: FSTN1 ID])
```

5.8.6.3. PKFSCLRQUE.PC

This program clears the queue information of the tray placed on a specified fixed station.

Argument 1:	Specify the ID of a fixed station. The ID can be acquired by PKFSGETID described earlier.

Usage:
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])
```

5.8.6.4. PKFSGETQUE.PC

This program gets the information about the unprocessed cells in the tray placed on specified fixed station. If the initial status of the tray is “Empty”, the robot places a part in each cell whose information is acquired. If it is “Full”, the robot picks the part from each cell.
The program gets the information about the unprocessed cells according to the following order of priority:

- Cells in a tray placed by PKFSPUTQUE
- Cell having the smallest layer number - namely, the cell in the bottom layer - if the initial status of the tray is “Empty”

  Cell having the largest layer number of the cells containing parts - namely, the cell in the top layer - if the initial status of the tray is “Full”
- Cell having a specified model ID if any is specified.
- Cell having the smallest cell number

Argument 1:	Specify the ID of a 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 register storing the processing status. 0: Success 1: Failure (The queue contains no cell that can be processed.)
Argument 4: (optional)	Specify a model ID.

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

The vision offset of each cell of a tray is calculated based on the relative position between this cell and cell 1. So when you teach the robot position to a tray, you must make the taught position conform to the position of cell 1.

5.8.6.5. PKFSACKQUE.PC

After the processing status of PKFSGETQUE becomes 0 and the unprocessed cell information is allocated, this program notifies the fixed station of the processing that has been performed for the cell.

Argument 1: Specify the ID of a 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.

Usage:

The following instruction returns that the processing has succeeded. The ID is stored in R[12].

```
CALL PKFSACKQUE(FStn ID=R[12: FSTN1 ID],Success)
```

5.8.6.6. PKFSPUTBUF.PC

This KAREL program is used only when a tray is used as a buffer.

It places a new buffer on a specified fixed station.

To specify the initial status of the placed buffer (“Empty” or “Full”), use the fixed station setup screen.

When a tray is used as a buffer, the robot can pick a part once placed in a cell again.

Argument 1: Specify the ID of a 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.

Usage:

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)
```

5.8.6.7. PKFSCLRBUF.PC

This KAREL program is used only when a tray is used as a buffer.

It clears the queue information of the buffer placed on a specified fixed station.

Argument 1: Specify the ID of a 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.

Usage:

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)
```

5.8.6.8. PKFSGETBUF.PC

This KAREL program is used only when a tray is used as a buffer.

It gets the target (empty cell information or information of the part placed in a cell) from the buffer placed on a specified fixed station. When a tray is used as a buffer, you need to specify the target you want to get, because the robot can either place a part in an empty cell or pick a part from a cell.

The program gets the information about the target according to the following order of priority:

- Cell in the buffer having the specified identification number
- Cell having the smallest layer number of the empty cells - namely, the cell in the bottom layer - if the target is a cell

  Part having the largest layer number of the cells containing parts - namely, the part in the top layer - if the target is a part
- Cell having a specified model ID if any is specified. A cell is selected based on the model ID even if the target is a part.
- Cell having the smallest cell number if the target is a cell

  When FIFO is used, First part that is placed in the cell, if the target is a part When LIFO is used, Last part that is placed in the cell, if the target is a part

Argument 1:	Specify the ID of a 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 register storing the processing status. 0: Success 1: Failure (The queue contains no target that can be processed.)
Argument 6: (optional)	Specify a model ID.

Example:

The following instruction stores the information about the unprocessed cell of the buffer whose identification number is 1 from fixed station 1 in VR[1] and the processing status in R[2]. The ID is stored in R[12].

```
CALL PKFSGETBUF (FStn ID=R[12: FSTN1 ID],Index =1,Cell,Offset VR=1,
Stat Reg=2)
```

⚠CAUTION

The vision offset of each cell of a tray is calculated based on the relative position between this cell and cell 1. So when you teach the robot position to a tray, you must make the taught position conform to the position of cell 1.

5.8.6.9. PKFSACKBUF.PC

This KAREL program is used only when a tray is used as a buffer.

After the processing status of PKFSGETBUF becomes 0 and the target information is allocated, this program notifies the fixed station of the processing that has been performed for the target.

Argument 1:	Specify the ID of a 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.

Usage:

The following instruction returns that the processing result is “Success”. The ID is stored in R[12].

```
CALL PKFSACKBUF(FStn ID=R[12: FSTN1 ID],Success)
```

5.8.6.10. PKFSGETTRID.PC

This program gets the ID of the tray that you assigned to a fixed station in the iRPickTool tree view.

Argument 1: Specify the ID of a fixed station.

Argument 2: Specify the register number in which to return the “Tray ID”.

Usage:

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)
```

5.8.6.11. PKFSGETUF.PC

This program gets the UFrame (UF) number that you assigned to a Fixed Station in the iRPickTool tree view.

Argument 1: Specify the ID of a fixed station.

Argument 2: Specify the number of the register to store the UF number.

Usage:

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)
```

5.8.6.12. PKFSCLRACK.PC

This program clears any pending ACK for a fixed conveyor station. Sometimes when you abort your robot program, a PKFSGETQUE (or PKFSGETBUF) call may have been made but not the PKFSACKQUE (or PKFSACKBUF) call. In this case, if you restart your main program, then if a PKFSGETQUE (or PKFSGETBUF) call is made prior to handling

the pending PKFSACKQUE (or PKFSACKBUF) call, then you will get alarm IRPK-051 “No ACKQUE after the last GETQUE.” However, if you execute PKFSCLRACK prior to the PKFSGETQUE (or PKFSGETBUF) call, then there will be no error. However, note that the system will permanently lose the part that was gotten by the PKFSGETQUE (or PKFSGETBUF) call prior to the Abort.

Argument 1: Specify the ID of a fixed station.

Usage:

The following instruction clears any pending ACKs for a fixed conveyor station whose ID is 1.

```
CALL PKFSCLRACK(FStn ID=1)
```

5.8.7. Tray-related programs

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

5.8.7.1. PKTRSETCLPOS.PC

This program changes a pattern in a tray. When there are multiple robots, if you execute this in one robot, the setting is reflected in all robots.

Argument 1: Specify the name of the tray using a character string.

NOTE

The number of displayed options is the maximum number of objects (16). In other words, those up to TRAY16 are displayed as tray names.

Argument 2: Specify the cell number from 1 to 160.

Argument 3: Specify the value of the target parameter with a number. The change targets are as follows.

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. Be sure to call this KAREL program at the beginning of the main program, that is, before KAREL programs PKWCSTART, PKSNSTART, or PKWCCHECK, so that the changed parameters are reflected correctly.
2. When you execute this program with the setup page open, the parameter to be changed might not be updated immediately. But they are updated automatically after a few seconds.
3. When calling KAREL programs for setting parameters (PKCVSETLBD.PC, PKCVSETTRNAM.PC, PKCVENBLB.PC, PKCVENBSC.PC, PKSNENBSENS.PC, PKSNSETVPNAM.PC, PKSNSETVTOFS.PC, PKSNSETMTOFS.PC, PKSNSETDBNC.PC, PKSNSETDURTN.PC, PKCSSETDSLIN.PC and PKTRSETCLPOS.PC) when using multiple robot controllers, be aware of the following points.
   - Insert WAIT instruction before PKWCSTART or PKWCCHECK in all robot controllers in order to wait for the completion of the parameter changes. Set the wait time to 5 seconds or so. If parameter changes have not completed by the time PKWCSTART or PKWCCHECK is called, IRPK-079 “Need to synchronize data” occurs. If IRPK-079 occurs even when the wait time is 5 sec, increase the time so that IRPK-079 does not occur.
   - If you call more than one of the listed KAREL programs for setting parameters, do so from the same robot controller. The listed KAREL programs can set the parameters of sensors or conveyor stations for other robot controllers also. If the listed KAREL programs for setting parameters are called in multiple robot controllers, the following phenomena may occur:
     - IRPK-079 “Need to synchronize data” occurs when PKWCSTART or PKWCCHECK is called.
     - IRPK-082 “The setting cannot be changed because the data is different” will occur in one of the robot controllers.
4. When using a recipe and a KAREL program for setting parameters together, first change the recipe and then call the KAREL program.

5.8.7.2. PKTRGETNUMCL.PC

This program returns the number of cells in the specified layer of the specified tray.

Argument 1:	Specify the ID of a tray. The ID of the tray which is used in a conveyor can be acquired by PKCSGETTRID described earlier.
Argument 2:	Specify the layer number. If you specify 0, this program returns the total number of cells in all layers of the tray.
Argument 3:	Specify the number of the register to store the number of cells.

Usage:

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)
```

5.8.7.3. PKTRGETCL.PC

This program returns data on the specified layer in the specified tray.

Argument 1:	Specify the ID of a tray. The ID of the tray used in a conveyor can be acquired by PKCSGETTRID described earlier.
Argument 2:	Specify the cell number from 1 to 160.
Argument 3:	Specify the register to return the model ID to.
Argument 4:	Specify the register to return the position R of the cell.
Argument 5:	Specify the register to return the position X of the cell.
Argument 6:	Specify the register to return the position Y of the cell.
Argument 7:	Specify the register to return the position Z of the cell.

Argumen t 8: Specify the register to return the layer number of the cell.

Usage:

Data is acquired from cell 1 in tray 1. Cell data is saved from R[1] to R[6].

```
CALL PKTRGETCL(Tray Index=1,Cell Index=1,Model ID Reg=1,R Reg=2,X Reg=3,Y Reg=4,Z Reg=5,Layer Reg=6)
```

5.9. Robot programs

Robot programs create programs that match the system.

This section contains examples of the basic programs and describes the concepts and other details of these programs. Tailor the sample programs provided herein to suit your system.

"I.5.9.1. Tracking program(P.367)" explains the details of tracking programs.

Programs that perform pickup and placement motions on a conveyor need to be created as tracking programs. Tracking programs require different settings from those of ordinary programs.

Subsequent subsections explain sample programs for each configuration.

Note that this section mainly discusses only the creation of the program logic.

For information about the actual task of robot position teaching, see "I.5.11. Teaching the position to robots(P.385)".

⚠CAUTION

By default, iRPickTool has hot start disabled.

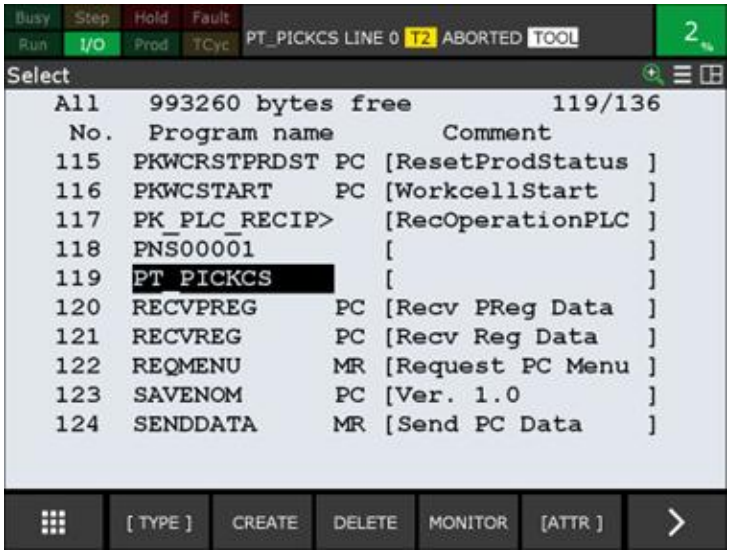
5.9.1. Tracking program

A program that causes the robot to follow the motion of the conveyor is called a tracking program. Unlike ordinary programs, a tracking program requires that a tracking schedule number and other data be set in advance using the program detail screen.

The setting procedure is as follows.

1 Press the [Select] key on the teach pendant.

A program list screen will appear.

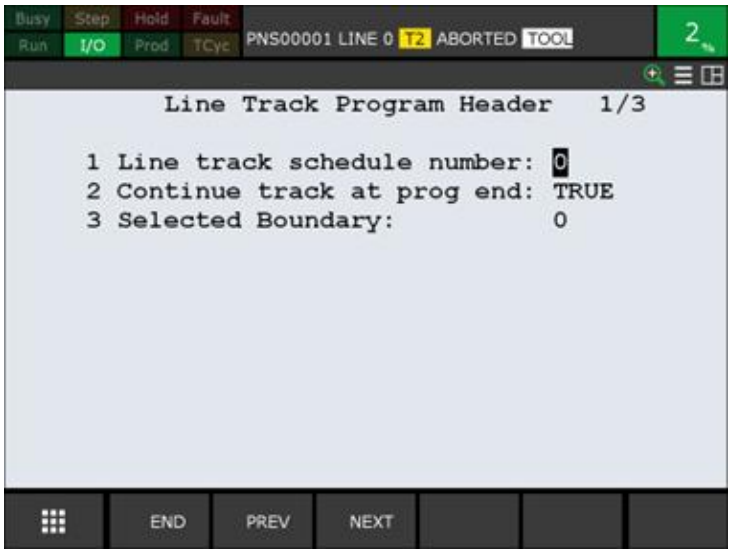


2 **From the list, select a tracking program.**

3 **Press [> (next page)], and then press F2 [DETAIL].**

4 **Press F3 [NEXT].**

The following screen appears.



5 **In [Line track schedule number], set the tracking schedule number specified for the conveyor station.**

6 **In [Continue track at prog end], set [TRUE].**

NOTE

If you set [FALSE] in [Continue track at prog end], the robot stops the tracking motion when the tracking program ends. As a result, the robot cannot move smoothly, even if it resumes tracking immediately when the next tracking program starts. A visual tracking system requires the robot to continue to move fast. Therefore, set [TRUE] in [Continue track at prog end] to ensure smooth robot motion.

7 **Leave the value in [Selected Boundary] unchanged from 0.**

If 0 is set, the tracking area specified for the conveyor station is used.

5.9.2. 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 needed.

Table 5.9.2-1 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 5.9.2-2 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

5.9.2.1. Main program

This is the main program of the system.

Setup example for a main program

MAIN1.TP

```
  1: R[1:CYCLE_STOP]=0
  2: CALL PKCSGETID('CONV1',CStn ID Reg=11)
  3: CALL PKCSGETID('CONV2',CStn ID Reg=12)
  4:
  5: UTOOL_NUM=1
  6: UFRAME_NUM=0
  7: J P[1:Perch Pos] 50% FINE
  8:
  9: CALL PKWCSTART
 10:
 11: LBL[100]
 12: CALL PICK1
 13: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 14: CALL DROP1
 15: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 16: JMP LBL[100]
 17:
 18: LBL[900]
 19: CALL PKWCEND
[End]
```

PICK1.TP is called on line 12 to pick a part, and DROP1.TP is called on line 14 to place the part. The main program mainly repeats the PICK1.TP and DROP1.TP calls alternately. When a hand that can hold multiple parts is used, PICK1.TP first picks all parts and then DROP1.TP places them at once.

R[1] serves as a cycle stop flag. The cycle stop flag is checked between a pickup motion and a placement motion and, if the flag is set to 1, the cycle is stopped.

PKCSGETID is called on lines 2 and 3 to store the IDs of the conveyor station in R[11] and R[12]. These are used in PICK1.TP and DROP1.TP, respectively.

PKWCSTART is called on line 9 to initialize the conveyor station and start a sensor task. If the cycle is stopped, PKWCEND is called on line 19 to stop the sensor task.

The initialization of the conveyor station and the startup of a sensor task can be performed separately by using PKCSCLRQUE and PKSNSTART. To do so, replace line 9 with those shown below.

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

First, PKWCCHECK is called to check for any invalid 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.

5.9.2.2. Pick program

This program performs a pickup motion on an infeed conveyor.

Setup example for a pick program

PICK1.TP

```
 1: UTOOL_NUM=1
 2: UFRAME_NUM=0
 3: R[3:ZONE] =1
 4:
 5: LBL[100]
 6: STOP_TRACKING
 7: CALL PKCSGETQUE(CStn ID=R[11:CSTN1],Consec Flag=R[3:ZONE],
    Timeout (ms)=100,Offset VR=1,Stat Reg=2)
 8: IF R[2:GETQ Status]=0,JMP LBL[200]
 9: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
10: JMP LBL[100]
11:
12: LBL[200]
13: L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
14: L PR[R[3:ZONE]] 4000mm/sec CNT0 VOFFSET,VR[1]
15: CALL VACUUM_ON
16: L PR[R[3:ZONE]] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
17: CALL PKCSACKQUE(CStn ID=R[11:CSTN1],Success)
```

```
  18:
  19: R[3:ZONE]=R[3:ZONE]+1
  20: IF R[3:ZONE]<=R[4:NUM_ZONE],JMP LBL[100]
  21: LBL[900]
[End]
```

In [Line track schedule number] of the program details screen, set the tracking schedule number specified for the conveyor station for which this robot is selected on conveyor CONV1.

In line 1, the tool frame number is selected. The tool frame number specifies the number of the tool frame that is set for the gripper.

As an example, suppose that the gripper tool frame has been set to number 1.

In this example, the gripper tool frame is common, and the position register is changed for each gripper zone. As an alternative way, you can do the same thing if you set a common position register and change the tool frame that is set for each gripper zone.

In line 2, the user frame number is selected. For the user frame number, always select 0.

In a tracking program, the robot may make an unexpected motion if a user frame is specified. (In a tracking program, when the user frame number is 0, it refers to the robot's tracking frame and not the world frame.)

PKCSGETQUE is called on line 7 to make the robot wait for up to 100 milliseconds before the part to be picked next enters the tracking area. When the part enters the tracking area, the vision offset is stored in VR[1] and 0 is stored in R[2]. After the part enters the tracking area, the program jumps to line 12 and performs a pickup motion. Then, PKCSACKQUE is called on line 17 to notify the conveyor station that the pickup motion is complete.

⚠CAUTION

When a hand that can hold multiple parts is used, parts are handled one by one sequentially as with PICK1.TP. Each time the status of PKCSGETQUE becomes 0, the trigger for the tracking motion is set. If you call PKCSGETQUE and PKCSACKQUE in advance as many times as the number of parts to be picked by the hand to get the information of those parts at once, the tracking motion trigger set in the last PKCSGETQUE program is used, potentially making the robot unable to pick the earlier acquired parts at the correct position.

Since the wait time is set to 100 milliseconds in the third argument of PKCSGETQUE, a timeout occurs in 100 milliseconds if there is no part to pick in the tracking area when PKCSGETQUE is called. In that case, R[2] contains a value other than 0, and PICK1.TP checks the cycle stop flag, calls PKCSGETQUE again, and waits for a part to be assigned.

⚠CAUTION

The robot controller that executes the 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 “I.5.12.4. Continuous termination type specification for a taught position(P.387)”. You can optimize the tracking motion further by using the linear distance specification instruction and time before instruction. For details, see “I.5.12.5. Linear distance specification(P.388)” and “I.5.12.8. Time before instruction(P.390)”.

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 “I.5.9.1. Tracking program(P.367)”, the sample programs presented herein have [Continue track at prog end] enabled. This setting is intended to allow the robot to move as fast as possible when performing successive pickup motions and placement motions.

If a part to pick is not conveyed for a certain period of time, however, the robot may go outside the conveyor station as it continues the previous tracking motion while waiting for a part.

To prevent the robot from going outside the operating space, the instruction stops the tracking motion. Even in this case, the robot starts the next tracking motion smoothly if a tracking trigger is set.

NOTE

When creating a pick program exclusively for a hand that can hold one part, you can make the program a little simpler than PICK1.TP. For information, read the following description of the placement program.

5.9.2.3. Placement program

This program performs a placement motion on an outfeed conveyor.

Setup example for a placement program

DROP1.TP

```
  1: UTOOL_NUM=1
  2: UFRAME_NUM=0
  3:
  4: LBL[100]
  5: STOP_TRACKING
  6: CALL PKCSGETQUE(CStn ID=R[12:CSTN2],Consec Flag=1, Timeout(ms)=100,
     Offset VR=2, Stat Reg=2)
  7: IF R[2:GETQ Status]=0,JMP LBL[200]
  8: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
  9: JMP LBL[100]
 10:
 11: LBL[200]
 12: L PR[11] 4000mm/sec CNT100 VOFFSET,VR[2] Tool_Offset,PR[12]
 13: L PR[11] 4000mm/sec CNT0 VOFFSET,VR[2]
 14: CALL VACUUM_OFF
 15: L PR[11] 4000mm/sec CNT100 VOFFSET,VR[2] Tool_Offset,PR[12]
 16: CALL PKCSACKQUE(CStn ID=R[12:CSTN2],Success)
 17:
 18: LBL[900]
[End]
```

This program is basically the same as the pick program.

In [Line track schedule number] of the program details 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.

5.9.3. 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 “I.5.9.2. Sample programs for the conveyor-to-conveyor configuration(P.368)” as well.

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

Table 5.9.3-1 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 5.9.3-2 Position Registers

PR[2]	Pickup position on the conveyor
PR[3]	Placement position on the fixed station
PR[11]	Tool offset to create an approach point for the pickup motion
PR[12]	Tool offset to create an approach point for the placement motion

5.9.3.1. Main program

This is the main program of the system.

Setup example for a main program

MAIN1.TP

```
  1: R[1:CYCLE_STOP]=0
  2: CALL PKCSGETID('CONV1',CStn ID Reg=6)
  3: CALL PKFSGETID('FSTN1',FStn ID Reg=7)
  4:
  5: ! Put a tray on a FSTN
  6: CALL PKFSPUTQUE(FStn ID=R[7:FSTN1_ID])
  7:
  8: UTOOL_NUM=1
  9: UFRAME_NUM=0
 10: J P[1:Perch Pos] 50% FINE
 11:
 12: CALL PKWCSTART
 13:
 14: LBL[100]
 15: CALL PICKCS
 16: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 17:
 18: CALL DROPFS
 19: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 20: JMP LBL[100]
 21:
 22: LBL[900]
 23: CALL PKWCEND
[End]
```

PICKCS.TP is called on line 13 to pick a part, and DROPFS.TP is called on line 16 to place the part. The main program mainly repeats the PICKCS.TP and DROPFS.TP calls alternately.

R[1] serves as a cycle stop flag. The cycle stop flag is checked between a pickup motion and a placement motion and, if the flag is set to 1, the cycle is stopped.

PKCSGETID is called on lines 2 and 3 to store the ID of the conveyor station in R[6] and the ID of the fixed station in R[7]. These are used in PICKCS.TP and DROPFS.TP, respectively.

PKFSPUTQUE is called on line 6 to place a tray on the fixed station.

PKWCSTART is called on line 10 to initialize the conveyor station and start a sensor task. If the cycle is stopped, PKWCEND is called on line 21 to stop the sensor task.

When multiple robots are used

In principle, the same programs may be copied for use on other robots.

Be careful about the following points.

- In MAIN1.TP, change the fixed station name to that of the fixed station where the relevant robot operates.
- PICKCS.TP and DROPFS.TP can be copied to other robots and used as they are. However, the tracking schedule number of the tracking program (PICKCS.TP) needs to be changed according to the conveyor station where the robot operates.

5.9.3.2. Pick pogram

This program performs a pickup motion on a conveyor.

Setup example for a pick program

PICKCS.TP

```
  1: UTOOL_NUM=1
  2: UFRAME_NUM=0
  3:
  4: LBL[100]
  5: ! Get a part from the 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 that the part has been picked from the CSTN
 16: CALL PKCSACKQUE(CStn ID=R[6:CSTN_ID],Success)
 17:
 18: LBL[900]
[End]
```

In [Line track schedule number] of the program details 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 “I.5.9.2.2. Pick program(P.370)”. Note that gripper zones are not used here.

5.9.3.3. Placement program

This program performs a placement motion on a fixed station for buffer.

Setup example for a placement program

DROPFS.TP

```
  1: UTOOL_NUM=1
  2: UFRAME_NUM=0
  3:
  4: LBL[100]
  5: ! Check if a tray is ready
  6: WAIT DI[21]=ON
  7:
  8: ! Get a cell from a 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 on 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 that the cell has been occupied
  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 acknowledge that the tray is full
  32: DO[21]=PULSE,1.0sec
  33:
  34: ! Prepare new tray
  35: CALL PKFSPUTQUE(FStn ID=R[7:FSTN1_ID])
  36:
  37: LBL[900]
[End]
```

A user frame number is specified on line 2. While the tracking program requires setting 0 for the user frame, any 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 “I.5.12.5. Linear distance specification(P.388)”, “I.5.12.7. Maximum linear speed function(P.389)”, and “I.5.12.8. Time before instruction(P.390)”.

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.

5.9.4. 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 “I.5.9.2. Sample programs for the conveyor-to-conveyor configuration(P.368)” as well.

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

Table 5.9.4-1 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 5.9.4-2 Position Registers

PR[2]	Pickup position on the conveyor
PR[3]	Placement position on the conveyor
PR[5]	Pickup position on the fixed station
PR[6]	Placement position on the fixed station
PR[12]	Tool offset to create an approach point for the pickup motion on the conveyor
PR[13]	Tool offset to create an approach point for the placement motion on the conveyor
PR[15]	Tool offset to create an approach point for the pickup motion on the fixed station
PR[16]	Tool offset to create an approach point for the placement motion on the fixed station

5.9.4.1. Main program

This is the main program of the system.

Setup example for a main program

MAIN1.TP

```
 1: R[1:CYCLE_STOP]=0
 2:
 3: ! Get station id
 4: CALL PKCSGETID('CONV1',CStn ID Reg=6)
 5: CALL PKCSGETID('CONV2',CStn ID Reg=7)
 6: CALL PKFSGETID('FSTN1',FStn ID Reg=8)
 7:
 8: ! Put a buffer on a FSTN
 9: CALL PKFSPUTBUF(FStn ID=R[8:BufferID],Index=1)
10:
11: UTOOL_NUM=1
12: UFRAME_NUM=0
13:J P[1:Perch Pos] 50% FINE
14:
15: CALL PKWCSTART
16:
17: R[10:Pick]=0
18: LBL[100]
19: ! Pick the part from the CSTN
20: CALL PICKCS
21: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
22: IF R[10:Pick]=1,JMP LBL[200]
23:
24: ! Pick the part from the buffer
25: CALL PICKBF
26: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
27: IF R[10:Pick]=0,JMP LBL[100]
```

```
  28:
  29: LBL[200]
  30: ! Drop a part on a CSTN
  31: CALL DROPCS
  32: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
  33: IF R[10:Pick]=0,JMP LBL[100]
  34:
  35: ! Drop a part on the buffer
  36: CALL DROPBF
  37: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
  38: IF R[10:Pick]=1,JMP LBL[200]
  39: JMP LBL[100]
  40:
  41: LBL[900]
  42: CALL PKWCEND
[End]
```

If there is no part when the robot attempts to pick one, it tries to pick a part from the buffer.

If there is no placement position when the robot attempts to place a part, it places that part in the buffer.

The robot checks the value of R[10:Pick] to determine whether to use the buffer.

R[10:Pick] is set to 1 when the robot hand is holding a part or to 0 when the robot hand is not holding a part.

In other words, if R[10:Pick] is 0 when the robot attempts to pick a part from the conveyor, the robot picks a part from the buffer.

If R[10:Pick] is 0 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.

5.9.4.2. Pick pogram

This program performs a pickup motion on a conveyor.

Setup example for a pick program

PICKCS.TP

```
  1: UTOOL_NUM=1
  2: UFRAME_NUM=0
  3:
  4: STOP_TRACKING
  5: ! Get a part from the 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 far upstream
  15: CALL PKCSGETTIME(CStn ID=R[6:CstnIDIn],Order=1,Consec 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: ! Get a 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 that the part has been picked from the CSTN
  35: CALL PKCSACKQUE(CStn ID=R[6:CstnIDIn],Success)
  36: R[10:Pick]=1
  37:
  38: LBL[900]
[End]
```

In [Line track schedule number] of the program details 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 R[10:Pick] set to 0, and accesses the buffer.

If the arrival time check reveals that the part will arrive soon, the robot repeatedly calls PKCSGETQUE on line 20 and waits for the part. Here, the wait time is set to 100 ms in order to reduce the CPU load.

5.9.4.3. Placement program

This program performs a placement motion on a conveyor.

Setup example for a placement program

DROPCS.TP

```
  1: UTOOL_NUM=1
  2: UFRAME_NUM=0
  3:
  4: STOP_TRACKING
  5: ! Get a cell from a tray
  6: CALL PKCSGETQUE(CStn ID=R[7:CstnIDOut],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 far upstream
 15: CALL PKCSGETTIME(CStn ID=R[7:CstnIDOut],Order=1,Consec 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: ! Get a cell from a tray
 20: CALL PKCSGETQUE(CStn ID=R[7:CstnIDOut],Consec Flag=1,Timeout(ms)=100,
     Offset VR=4,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 on 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 that the cell has been occupied
 35: CALL PKCSACKQUE(CStn ID=R[7:CstnIDOut],Success)
 36: R[10:Pick]=0
 37:
 38: LBL[900]
[End]
```

This program is basically the same as the pick program for picking parts from a conveyor.

In [Line track schedule number] of the program details screen, set the tracking schedule number specified for the conveyor station for which this robot is selected on conveyor CONV2.

5.9.4.4. Pick program (buffer)

This program performs a pickup motion on a fixed station for buffer.

Setup example of a pick program (buffer)

PICKBUF.TP

```
  1: UTOOL_NUM=1
  2: UFRAME_NUM=0
  3:
  4: ! Get a part from a buffer
  5: CALL PKFSGETBUF(FStn ID=R[8:BufferID],Index=1,Part,Offset VR=3,Stat Reg=5)
  6: IF R[5:FsGetQStat]<=0,JMP LBL[100]
  7: IF R[5:FsGetQStat]>0,JMP LBL[900]
```

```
   8:
   9: LBL[100]
  10: ! Pick the part from the buffer
  11: L PR[5] 10000mm/sec CNT100 VOFFSET,VR[3] Offset,PR[15]
  12: L PR[5] 10000mm/sec CNT0 VOFFSET,VR[3]
  13: CALL VACUUM_ON
  14: L PR[5] 10000mm/sec CNT100 VOFFSET,VR[3] Offset,PR[15]
  15:
  16: ! Ack that the part has been picked from the buffer
  17: CALL PKFSACKBUF(FStn ID=R[8:BufferID],Success)
  18: R[10:Pick]=1
  19:
  20: LBL[900]
[End]
```

A user frame number is specified on line 2. While the tracking program requires setting 0 for the user frame, any 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 “I.5.12.5. Linear distance specification(P.388)”, “I.5.12.7. Maximum linear speed function(P.389)”, and “I.5.12.8. Time before instruction(P.390)”.

5.9.4.5. Placement program (buffer)

This program performs a placement motion on a fixed station for buffer.

Setup example of a placement program (buffer)

DROPBUF.TP

```
   1: UTOOL_NUM=1
   2: UFRAME_NUM=0
   3:
   4: ! Get a cell from a buffer
   5: CALL PKFSGETBUF(FStn ID=R[8:BufferID],Index=1,Cell,Offset VR=2, Stat Reg=5)
   6: IF R[5:FsGetQStat]<=0,JMP LBL[100]
   7: IF R[5:FsGetQStat]>0,JMP LBL[900]
   8:
   9: LBL[100]
  10: ! Drop a part on the buffer
  11: L PR[6] 10000mm/sec CNT100 VOFFSET,VR[2] Offset,PR[16]
  12: L PR[6] 10000mm/sec CNT0 VOFFSET,VR[2]
  13: CALL VACUUM_OFF
  14: L PR[6] 10000mm/sec CNT100 VOFFSET,VR[2] Offset,PR[16]
  15:
  16: ! Ack that the cell has been occupied
  17: CALL PKFSACKBUF(FStn ID=R[8:BufferID],Success,2)
  18: R[10:Pick]=0
  19:
  20: LBL[900]
[End]
```

This program is basically the same as the program for picking parts from a buffer fixed station.

On line 5, an empty cell is picked from the buffer. On lines 11 to 14, a pickup motion is performed.

5.10. 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 pick-up/placement positions to all the robots that work on the same conveyor, with regard to a part for which a reference position has been specified.

To set the reference position, use the reference position setting guide on 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 the tracking motion. Then, set the reference position and perform robot position teaching in the placement program.
2. If [Ref. Data To Use] is set to [Model ID] in the vision process, repeat the procedure described below for each model ID. In such a case, set each model ID individually using a locator tool such as the GPM locator tool, and create the reference data for the number of model IDs by adding the reference data before setting the reference position.
3. You need to open the iRPickTool Setup screen with the controller selected for this sensor and configure the reference position.

The operation procedure slightly differs depending on the combination of sensor trigger condition and action.

For the operation procedures for the following combinations, see “I.2.3.10. Setting up a reference position and teaching a robot position(P.113)”.

- In the case of [Distance] + [Find part by vision]

  The procedure for an infeed conveyor is given, but it is the same in other cases, too.
- In the case of [DI]/[RI]/[HDI] + [Put part in queue]

  The procedure for when using a tray on an outfeed conveyor is given, but it is the same in other cases, too.

  In the case of the combination below, the part is not actually detected, so set the reference position at the same time as you set the target position. For details, see “I.5.5.6. Setting the target position(P.312)”.
- In the case of [Distance]/[FLAG] + [Put part in queue]

5.10.1. DI / RI and Find part by vision

This section explains the procedure to set the reference position for the case where you have set the trigger condition to [DI] or [RI] and the action to [Find part by vision] on the sensor setup screen.

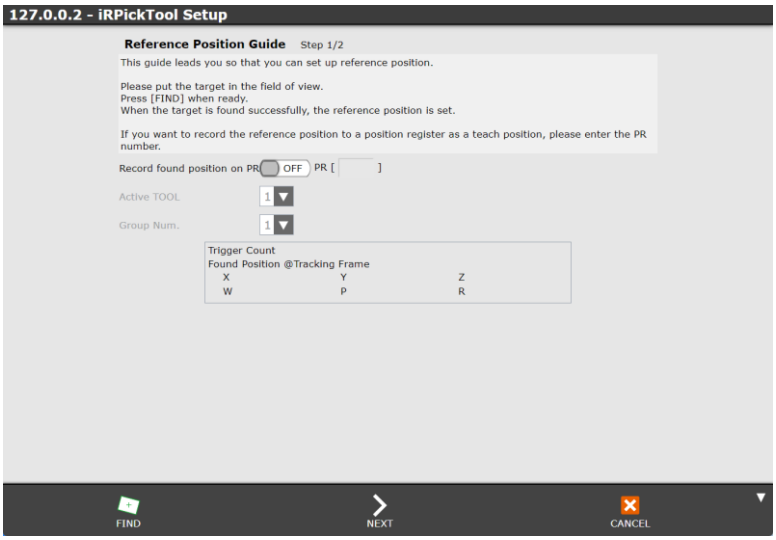
For the setup of a sensor, see “I.5.5. Setting up a sensor(P.288)”.

1 Stop the conveyor.

2 Select [SENS*] for the sensor, and click [SET REF POS].



The following [Reference Position Guide Step: 1/2] will appear.



NOTE

When [PR Num. to Set] is available, the robot position teaching described later can be simplified by using this function. For details, see “Using a Position Register number for [Record found position on PR(P.118)”.

3 Place a part upstream of the trigger sensor.

Place it around the center of the conveyor’s width direction.

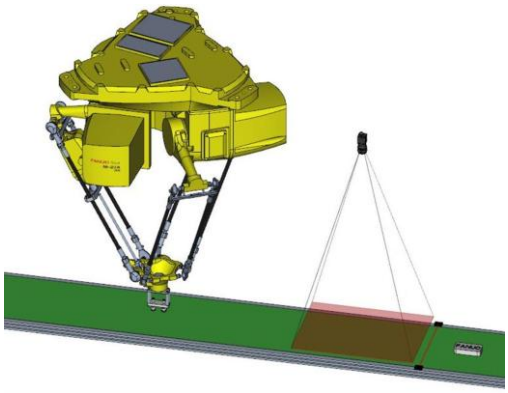


Fig. 5.10.1-1 Placing a part upstream of the trigger sensor

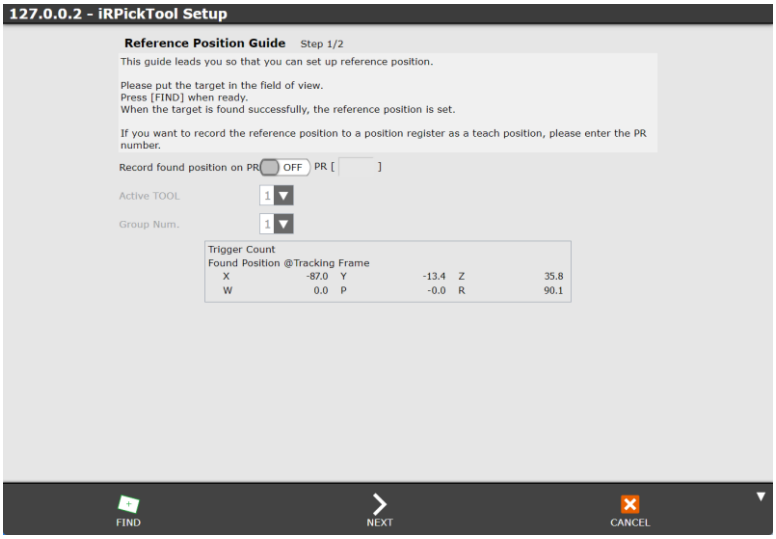
4 Move the conveyor, and then when the part enters the robot's moving range, stop it.

If detection succeeds, [Found Position] will be displayed.

* If detection does not succeed

- Open the vision process and select Runtime Image '100% Display'.
- Open the Runtime Image. Check the image.
- If the detection timing is off, change the value of [Trigger Offset].

If detection succeeds, [NEXT] will become active.



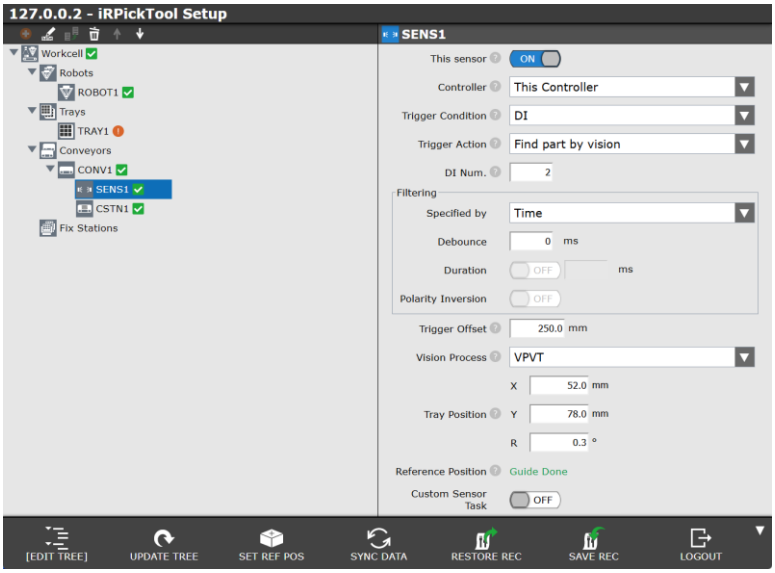
5 Click [NEXT].

After a while, the following screen appears.



6 Click [FINISH].

Check that [Guide Done] appears in [Reference position].



7 **Next, teach the robot position. (See “I.5.11. Teaching the position to robots(P.385)”.)**

5.11. Teaching the position to robots

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

For the operation procedures for the following combinations, see “I.2.3.10. Setting up a reference position and teaching a robot position(P.113)”.

- In the case of [Distance] + [Find part by vision]
  The procedure for an infeed conveyor is given, but it is the same in other cases, too.
- In the case of [DI]/[RI]/[HDI] + [Put part in queue]
  The procedure for when using a tray on an outfeed conveyor is given, but it is the same in other cases, too.

5.11.1. Conveyor station

1 **When the part enters the tracking area after the reference position is set on the sensor guide screen, stop the conveyor.**

⚠CAUTION

Be careful that the part does not become misaligned on the conveyor when the conveyor has been stopped.

2 **From the program list, open the tracking program.**

3 **Check that the user frame and user tool frame used in the tracking program are selected.**

4 **Jog the robot to teach the pickup position (or placement position).**

⚠CAUTION

1. Be sure to teach the pickup (or placement) position in the tracking program. This is necessary to teach the coordinates in the tracking frame. Even if a Position Register is used, do not teach the position on the Position Register screen. If you place the cursor on the line number of the motion instruction in the tracking program that moves the robot to the pickup position (or placement position), with the robot located at the pickup position (or placement position), and press the F5 [TOUCHUP] key while holding down the shift key the position is taught. When you are asked “Subtract VR[*] from current pos?”, select [NO]. Or when you are asked “VR[*] is UNINIT, continue?”, select [YES].
2. When using trays, teach the placement (or pickup) position for cell 1. The offset for each cell is calculated automatically from the coordinates of the cell and the coordinates of the reference cell. By using a vision correction instruction, the robot moves to each cell by referring to the offset stored in a Vision Register.

CAUTION

3. In the case of a 3-, 4- or 5-axis robot, you must teach the robot position as each W and P of it are 0 or 180 or -180.
4. In the case of a 5-axis robot, there are postures that the robot cannot take due to its structural limitations. Therefore, teach the pickup position (or placement position) so that the flange face faces downward (or upward). For details, refer to “Motion of 5-axis robot” in the subsection, “Motion format” in the section, “Motion instructions” in the chapter, “Program structure” in the ““OPERATOR’S MANUAL(Basic Operation)"(B-84464EN)”.

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.

5.11.2. Fixed station

On the tray setup screen, press [SET REF CELL] and set the values of cell 1 for the reference cell, then teach the placement/picking position for cell 1 only. For details, see “I.5.3.3. Setting the reference cell(P.275)”.

The position for each cell is created in relation to the reference cell, using a vision correction instruction.

NOTE

The placement/picking position for cell 1 can be taught in relation to any arbitrary user frame other than the user frame that indicates the tray frame.

5.12. 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 the 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.

5.12.1. Checking and fine adjustment of the tracking motion

After teaching the robot position, adjust the tracking motion first. After moving the conveyor with the workpiece to the robot motion range, stop the conveyor, to check the robot pick (place) position and make a minor adjustment to the teach position.

For the fine adjustment procedure, see “I.2.3.11.1. Checking and fine adjustment of the tracking motion(P.127)”.

5.12.2. Adjustment of the tracking frame error

When a camera is used to detect parts, the tracking motion is susceptible 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.

See “I.2.3.11.2. Adjustment of the tracking frame error(P.131)” for the error adjustment procedures.

5.12.3. 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.

See “I.2.3.11.3. Adjustment of the dynamic error of the robot(P.139)” for the error adjustment procedures.

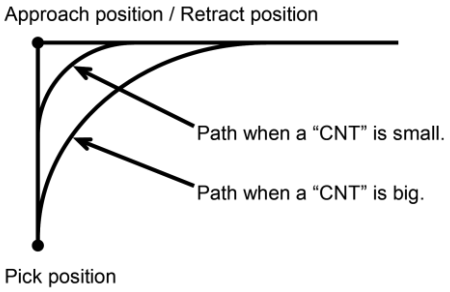
5.12.4. Continuous termination type specification for a taught position

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.



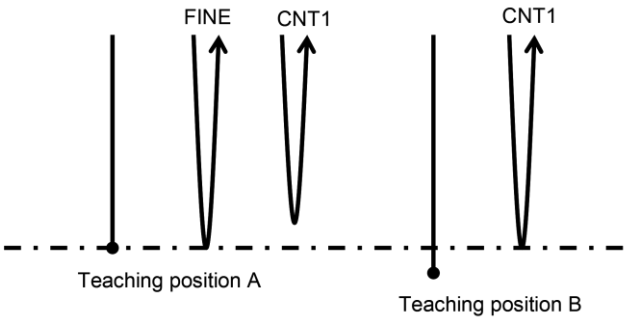
Pickup position and placement position

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.

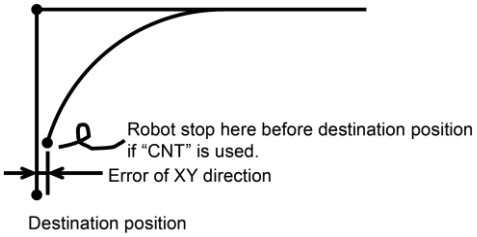
Note, however, that if CNT is specified, the robot moves slightly inward even when the value is small. This causes the robot to start moving toward the retract point before it arrives at the taught pickup position (or placement position). In the case of taught position A in the figure below, for example,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



5.12.5. Linear distance specification

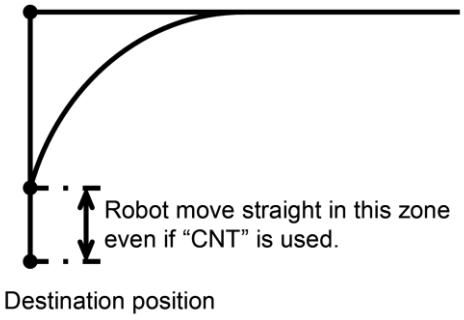
If a large CNT value is specified for the approach point or retract point and CNT is specified for the pickup position (or placement position) instead of FINE, the robot may fail to reach the destination position not only in height but also in the X and Y directions. The remaining distance appears to 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 "I.5.9.2.2. Pick program(P.370)".

```
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.

5.12.6. Linear face plate instruction

A linear face plate instruction is added to a motion instruction in a TP program. Add this instruction when the TCP is offset from the center of the face plate.

In a normal linear motion, control is exerted so that the TCP moves straight, as shown in the figure below. As a result, when the TCP is offset from the center of the face plate, particularly as in the case of a robot hand that can hold multiple parts, the center of the face plate zigzags and the robot has to move a longer distance, potentially resulting in the cycle time being not optimal.

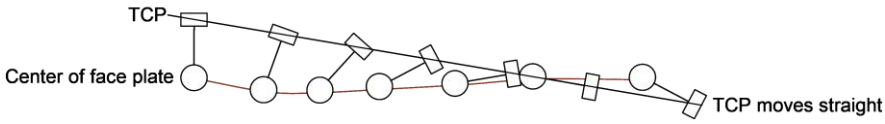


Fig. 5.12.6-1 Normal linear motion

If a linear face plate instruction is used, the center of the face plate becomes straight as shown in the figure below, although the linearity of the TCP is not maintained. This minimizes the robot movement distance and leads to a shorter cycle time.

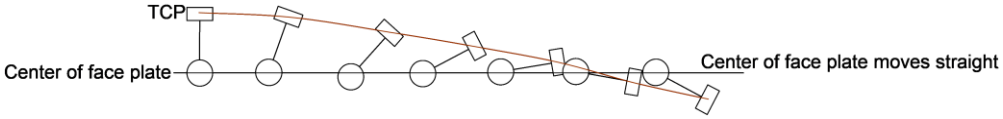


Fig. 5.12.6-2 When a linear face plate instruction is used

The instruction is expected to be effective particularly with a delta 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 "I.5.9.2.2. Pick program(P.370)".

```
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.

5.12.7. Maximum linear speed function

The maximum speed specified in a linear motion instruction (e.g., 2000 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 "I.5.9.3.2. Pick program(P.375)".

```
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.

5.12.8. 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 "I.5.9.2.2. Pick program(P.370)".

```
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.

5.13. Precautions for when starting up multiple robots

When performing visual tracking with multiple robots, if possible, start them with the conveyor stopped. Then, move the conveyor only after calling PKWCSTART or PKSNSTART on the robot controller where the sensor is running.

If the conveyor is moving at startup and the timing for calling PKWCSTART or PKCSCLRQUE differs between robots, the following phenomena may occur.

1. If a downstream robot calls PKWCSTART or PKCSCLRQUE before an upstream robot, part information from before the call may remain in the downstream robot.

   (If the following sequence occurs: the function is called by the downstream robot → workpiece data is sent from the upstream robot to the downstream robot → the function is then called by the upstream robot)
2. If PKWCSTART or PKCSCLRQUE is called on a robot downstream of the sensor after the sensor has been started up, the information of the detected parts will be lost.

5.14. Production runs

Production can be executed once test execution of the application is complete and both accuracy and processing volume have been properly adjusted.

5.14.1. Selecting the operator's panel for starting production

Production can be started either by pressing the start switch on the operator's panel or from peripheral equipment. In many cases, the peripheral equipment I/O signal is transmitted from the PLC.

5.14.1.1. Operator's panel

When using iRPickTool (J945), refer to "I.6.9.1. Starting production with the operator's panel(P.442)".

When you do not use a PLC and want to run iRPickTool Basic only with the operator's panel, perform the following operations.

- Set the remote/local setting to local. On the teach pendant of each robot controller, press the [MENU] key, navigate to [System] → [Configuration] screen, select the [Remote/Local Setting] item, and set it to [Local].
- In each robot controller, open the [LIST] screen and select the main program.
- Press the start switch on the operator's panel of each robot controller.

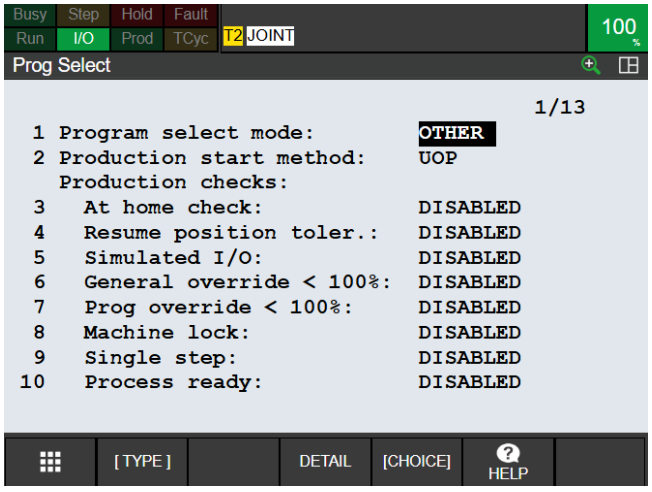
5.14.1.2. Peripheral equipment

When using iRPickTool (J945), refer to "I.6.9.2. Starting production with peripheral equipment(P.442)".

Normally, this method is used in the case where production is started from a PLC in a iRPickTool Basic workcell. The PLC and robot controller are connected via the peripheral equipment I/O signal. Connect the PLC to the robot controller so that the signal can be sent to each robot.

Follow the steps below when doing a production run in peripheral equipment mode.

- Set the remote/local setting to remote. On the teach pendant of each robot controller, press the [MENU] key, navigate to [System] → [Configuration] screen, select the [Remote/Local Setting] item, and set it to [Remote].
- Open the [Settings] → [Select Program] screen, and set [Automatic Operation Start Method] to "UOP". In addition, set [Program Selection Method] to "OTHER" (default setting), "PNS", or "RSR."



- When [Program Selection Method] is "OTHER", press F3 [DETAIL] and set the main program name to Data ID $UOP.PROGRAM_FOR_OTHER_MODE.
- When [Program Selection Method] is "PNS", create a TP program named "PNS0001.TP". At this time, set the group mask to <1,*,*,*,*,*,*,*>. This means that this program is specifically for Group 1. Note that the PNS program for motion Group 1 needs to be named PNS0001.TP.

- When [Program Selection Method] is “RSR”, create a TP program named “RSR0001.TP”. At this time, set the group mask to <1,*,*,*,*,*,*,*>. This means that this program is specifically for Group 1. Note that the RSR program for motion Group 1 needs be named RSR0001.TP.
- If a stack overflow alarm occurs during execution of the main TP program, increase the stack size of the main TP program.

For details on the [Select Program] screen, PNS, and RSR, refer to "OPERATOR’S MANUAL(Basic Operation)"(B-84464EN). Once these settings have been made in each robot controller, restart all the robot controllers.

5.14.2. Pausing, resuming, and aborting production

5.14.2.1. Pausing production

You can pause any robot during production by pressing the [HOLD] key on the teach pendant, enabling the teach pendant, or pressing the emergency stop button. Robots will also pause when the PAUSE alarm occurs during production. The HOLD signal can also be sent from a PLC via peripheral equipment I/O.

5.14.2.2. Aborting production

To abort production, perform the following operations.

- If production is being executed from the operation panel, press the [FCTN] key on the teach pendant of each robot controller, and select [END PROGRAM].
- If production is being executed from peripheral equipment, send the abort signal to each robot.

In order to prevent parts from being missed, stop the conveyor before aborting production.

During production, it is possible to pause only a specific robot at any time. Since sensor tasks are not stopped by the abort signal, even if production is aborted on the robot controller connected to the sensor, the iRPickTool application will continue running on other robots.

5.14.2.3. Resuming production

To resume production of a robot that has been paused, perform the following operations.

- If production is being executed from the operator’s panel, press the start switch on the operator’s panel of the paused robot.
- If production is being executed from peripheral equipment, send the start signal to the paused robot.

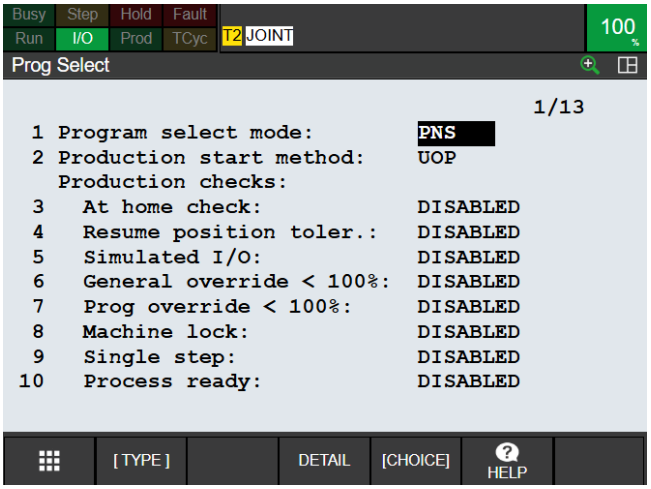
5.14.3. In the case of dual arm robots

When using iRPickTool (J945), refer to “I.6.9.3. In the case of dual arm robots(P.443)”.

If there is a robot controller for a dual robot group, you can start iRPickTool Basic using one of the following two methods.

- Using PNS:

  Press the [MENU] key, select [Settings] to display the [Select Program] screen, and set the [Program Selection Method] to “PNS”. If you have any questions regarding the PNS settings, refer to “"OPERATOR’S MANUAL(Basic Operation)"(B-84464EN)”.

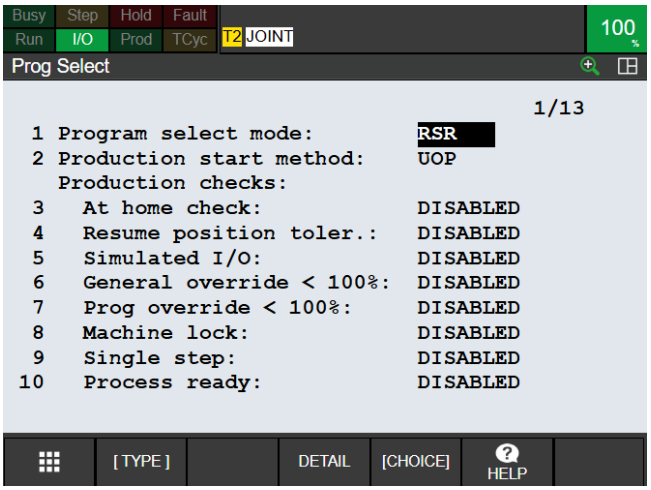


Create two programs: “PNS0001.TP” executed in Group 1 and “PNS0002.TP” executed in Group 2. In the header of the TP program, set the group mask to <1,*,*,*,*,*,*,*> for PNS0001.TP and to <*,1,*,*,*,*,*,*> for PNS0002.TP, respectively. Additionally, if a stack overflow alarm occurs during execution of the TP program, increase the stack size of PNS0001.TP and PNS0002.TP.

This allows each robot motion group to be executed, paused, resumed, and aborted independently.

- Using RSR:

  Press the [MENU] key, select [Settings] to display the [Select Program] screen, and set the [Program Selection Method] to “RSR”. If you have any questions regarding the RSR settings, refer to ““OPERATOR’S MANUAL(Basic Operation)"(B-84464EN)”.



Create two programs: “RSR0001.TP” executed in Group 1 and “RSR0002.TP” executed in Group 2. In the header of the TP program, set the group mask to <1,*,*,*,*,*,*,*> for RSR0001.TP and to <*,1,*,*,*,*,*,*> for RSR0002.TP, respectively. Additionally, if a stack overflow alarm occurs during execution of the TP program, increase the stack size of RSR0001.TP and RSR0002.TP.

This allows each robot motion group to be executed, paused, resumed, and aborted independently.

5.15. Setup of pre-grouping

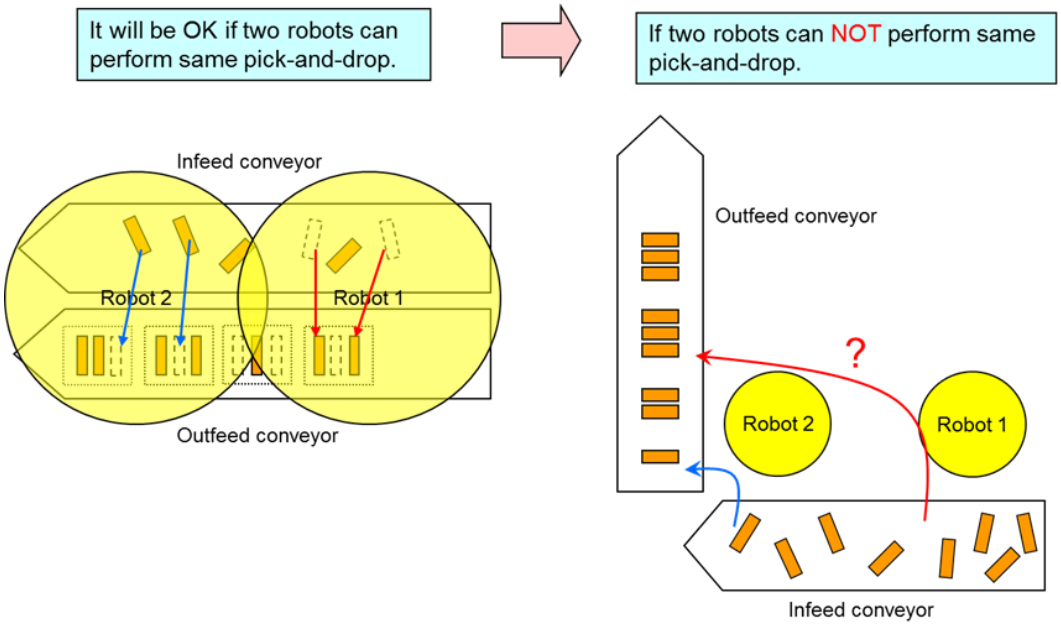
This section describes the setup procedure of the pre-grouping function which arranges parts on the same conveyor.

For settings specific to North America, in order to use the pre-grouping function, you will need the option for North America R856: iRPickTool / Pre-grouping.

Basically, you have pre-grouping robots which arrange random parts on the infeed conveyor into an arranged group. The arranged group is also sometimes called 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

This feature can be used effectively when two robots cannot perform the same pick-and-drop due to the customer's requirement such as a system layout.



Example 2

Even two robots can perform same pick-and-drop, due to too long path, cycle time for each robot is sometimes too slow. This feature may overcome the situation.

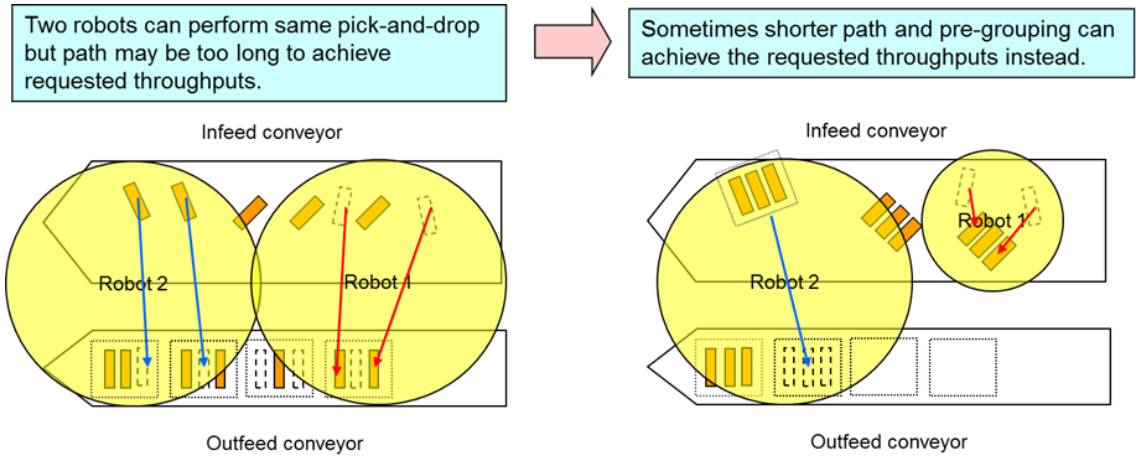


Fig. 5.15-1 Example of a configuration in which the pre-grouping function can be used effectively

Suppose there is a three-robot system that consists of two upstream robots that arrange parts on the same conveyor into virtual trays and a 3rd robot at the most downstream position that conveys arranged parts to another conveyor at one time.

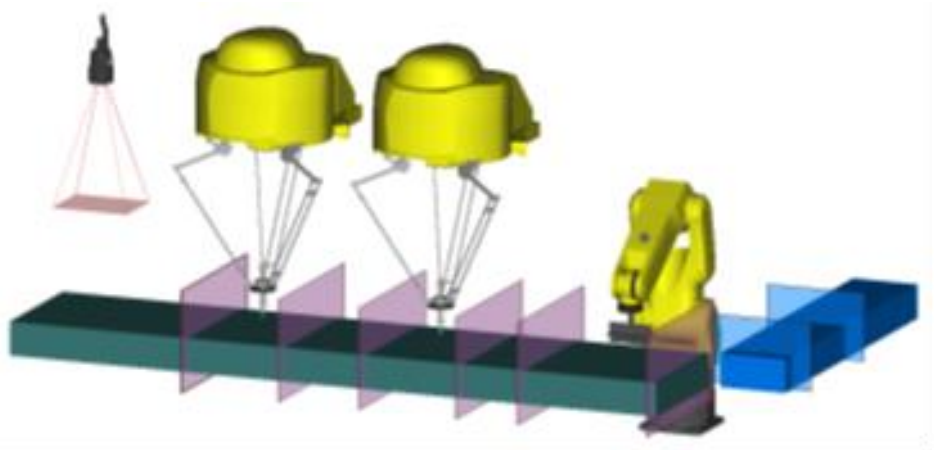


Fig. 5.15-2 Example of a system where the 3rd robot in the most downstream conveys arranged parts to another conveyor at one time

The system configuration and tree view composition in this case are shown in Fig. 5.15-3.

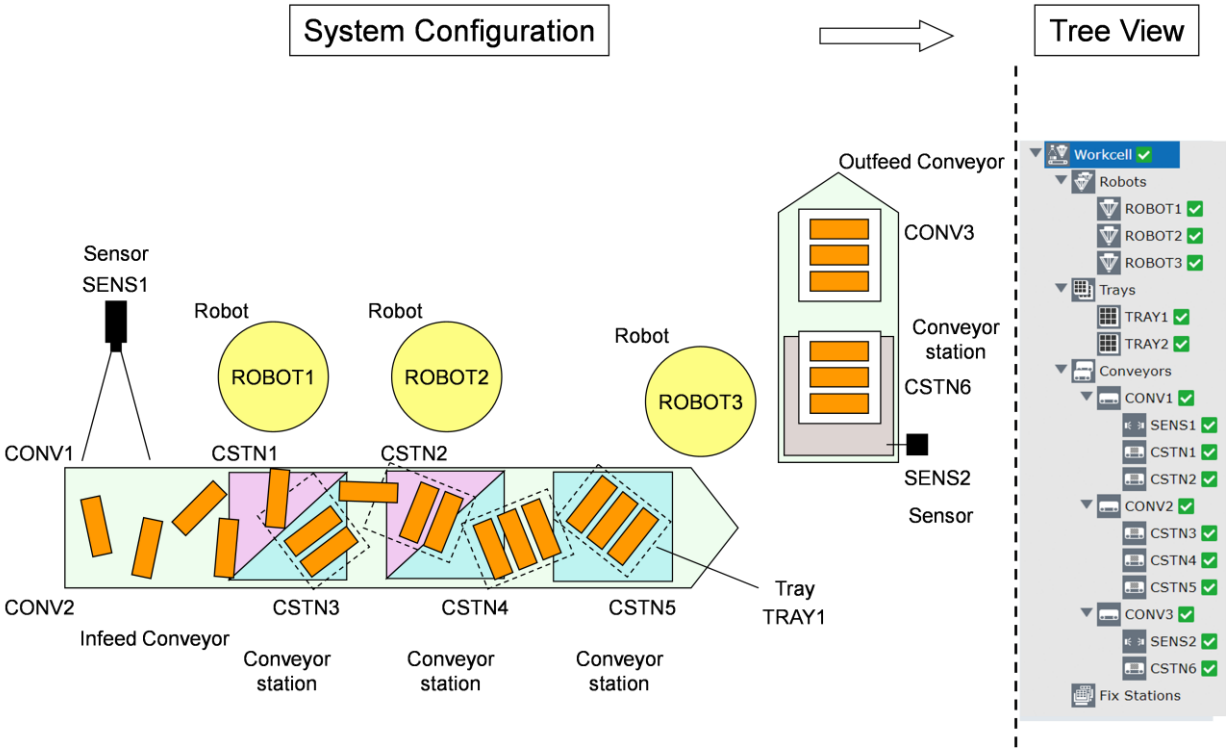


Fig. 5.15-3 System configuration and tree view configuration

As you can see, two conveyor station objects are created in the same area for arrangement on the same conveyor in the tree view. One is for picking operations and the other is for placement operations.

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. 5.15-3 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.

5.15.1. Terms

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

Pre-grouping conveyor

This is a conveyor object for arranged placement or pre-grouping operation.Fig. 5.15-3(P.395), CONV2 is a pre-grouping conveyor.

Infeed conveyor

This is a conveyor object for picking operation.Fig. 5.15-3(P.395), CONV1 is an infeed conveyor.

Virtual tray

If parts are arranged on the same conveyor, the tray is not actually moving, but parts seems to be arranged in line on a virtual tray. This is called a virtual tray.

This virtual tray is defined using a tray object as described below. 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. The orientations of the virtual trays can vary widely depending upon the randomness of the part flow and part density.

Base part

The base part is not moved when arranging.

5.15.2. Setup of a workcell and robot

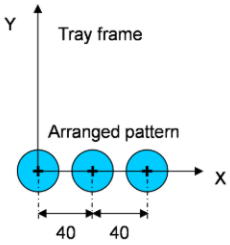
There are no special steps for pre-grouping. See “I.5.1. Setting up a workcell(P.266)” and “I.5.2. Setting up a robot(P.268)”.

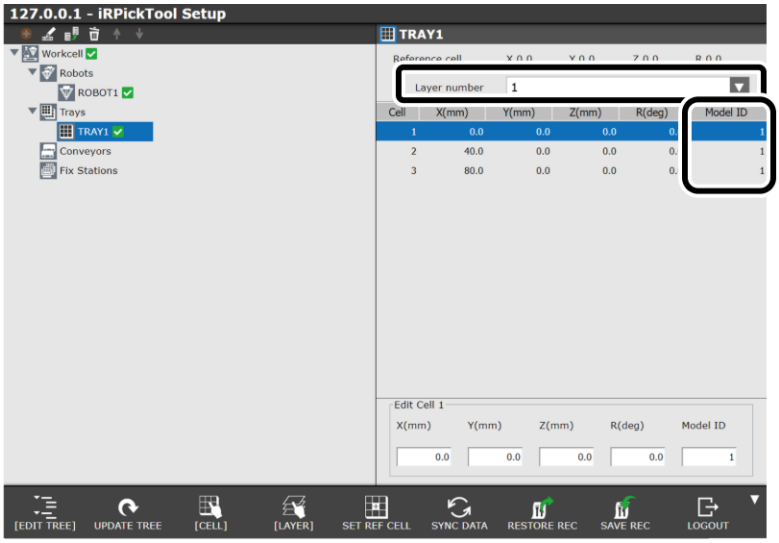
5.15.3. Setting up a tray

This section shows how to set up a virtual tray.

- The origin point of a tray frame can be set at any position desired, when treating the part of a [Circle (without Phase)].

  For example, it is set up so that X of 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 is the same as the X direction 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 layer 1.
- A model ID cannot be used by a pre-grouping function. The model ID of all cells in layer 1 must be set to 1.





CAUTION

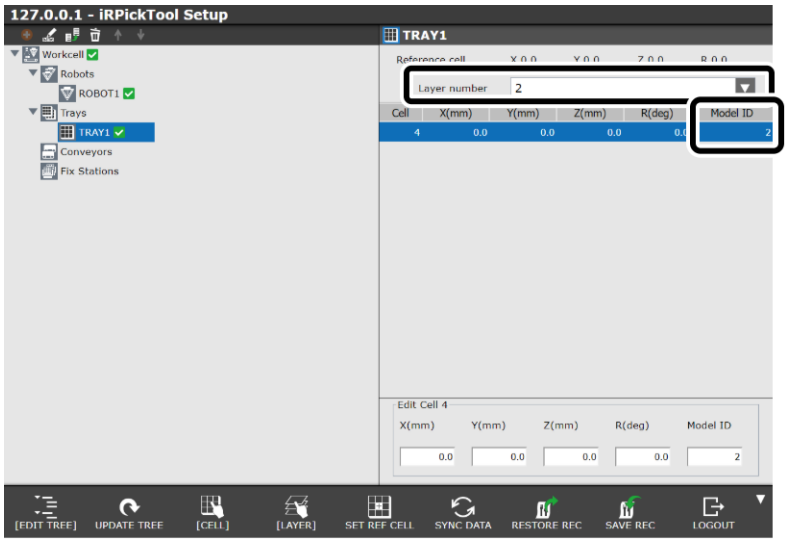
The center position of a part is set to the coordinates of each cell in layer 1. If the center position of a part is not set to the coordinates of a cell correctly, the part to be placed for arrangement may collide with the part on a conveyor because the interference check between the part on the conveyor and the part to be placed may not be calculated correctly.

NOTE

If cells are too close to each other, it may be judged that the base part and arranged parts will collide with each other when PKCSGENVRTRY is called, and the error IRPK-088 “There is an invalid pre-grouping setting.” may occur.

- Even though pre-grouping allows creating only single layer virtual trays, you can define one cell in layer 2.
- In order to pick only completely arranged parts, the cell of layer 2 should be set up with a model ID different from the cell of 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.



5.15.4. Setting up a conveyor

In pre-grouping you always setup two conveyors: the infeed conveyor and the pre-grouping conveyor.

1 The object of an infeed conveyor is set up.

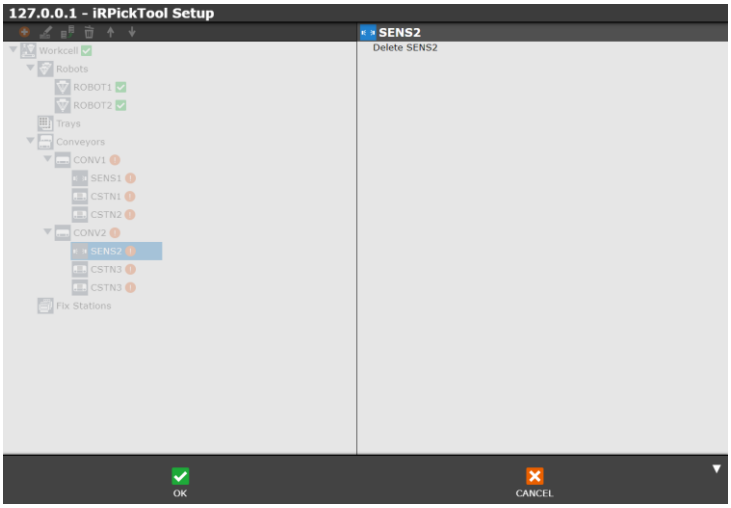
For details, see “I.5.4. Setting up a conveyor(P.276)”.

In the real world, both conveyors are the same physical conveyors. These two conveyors use the same tracking frame. As mentioned above, set up a tracking frame only for the infeed conveyor.

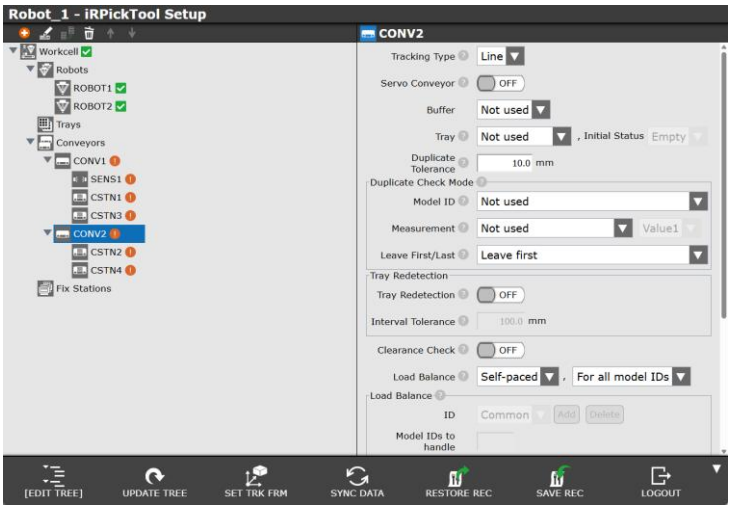
If the robot has been added only to the pre-grouping conveyor, set up a tracking frame as well.

2 The object of a pre-grouping conveyor is set up.

Under the object of a pre-grouping conveyor, a sensor object cannot be created. A setup of a pre-grouping cannot be performed in a conveyor object with a sensor object. When a conveyor object is created to use a pre-grouping function, the sensor object generated automatically must be deleted manually.



3 Then, add a required number of conveyor station objects.

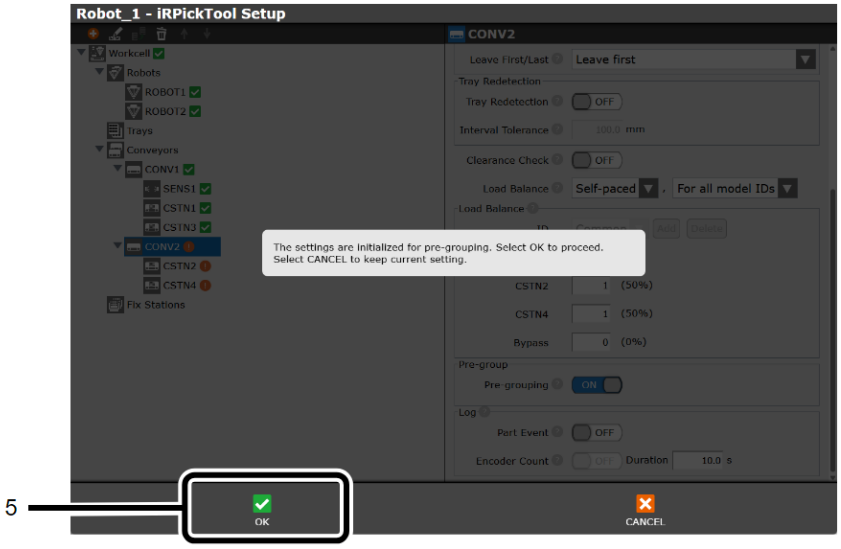


4 Set the [Pre-grouping] toggle switch to [Enable] in [Pre-group].

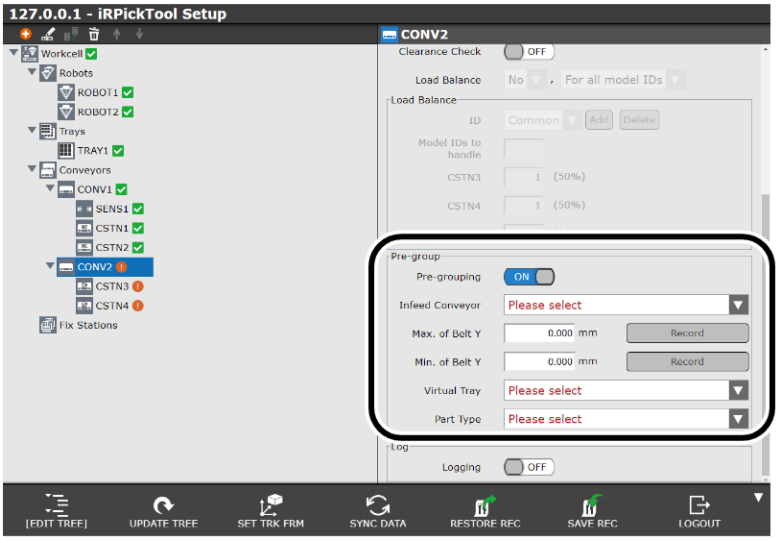
NOTE

When [Arc] is selected as the [Tracking Type] or the [Servo Conveyor] toggle switch is set to [Enable], the setting items for [Pre-group] are not displayed.

If the [Pre-grouping] toggle switch is set to [Enable], the following confirmation screens will be displayed.



5 Click [OK] to display the setting item of a pre-grouping.

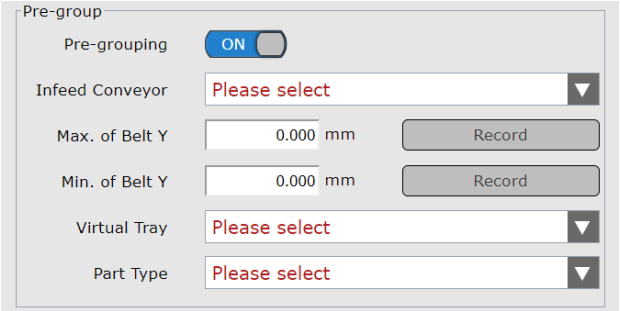


NOTE

1. Setting items initialized for pre-groupings are [Load Balance] and [Sort] of child conveyor station objects. The settings of [Load Balance] are initialized to [Disable] and [For all model IDs]. [Sort] of child conveyor station objects is initialized to Disabled.
2. In the pre-grouping conveyor, the [Tracking Type], [Servo Conveyor], [Tray] and [Load Balance] settings cannot be changed.
3. When multiple robots perform arrangement on the same conveyor using a load balance function, [Load Balance] must be set in the setting menu of an infeed conveyor.

6 Set items other than [Pre-group] according to "I.5.4. Setting up a conveyor(P.276)".

[Pre-group]



[Infeed Conveyor]

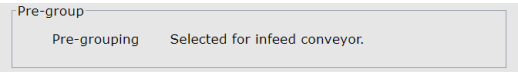
Select the infeed conveyor object.

NOTE

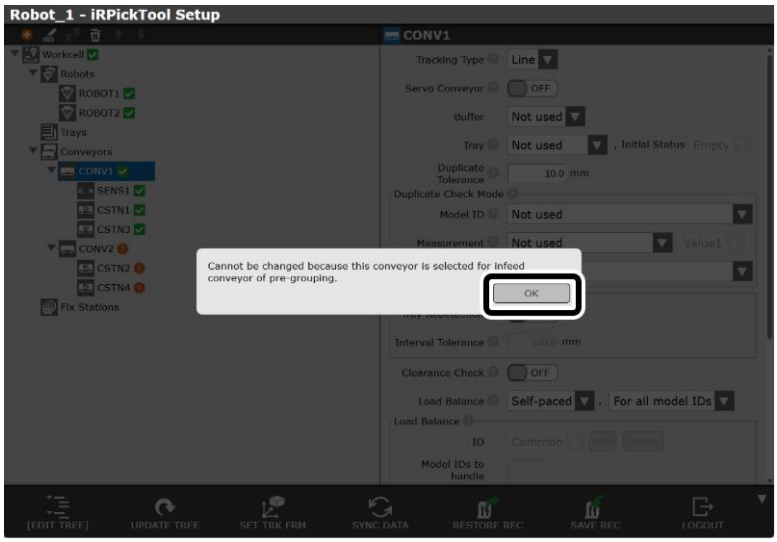
Only conveyor objects that meet the following conditions are displayed as options of [Infeed Conveyor].

- [Line] is selected as the [Tracking Type].
- The [Servo Conveyor] toggle switch is set to [Disable].
- [Not used] is selected for [Tracking Type].
- The [Pre-grouping] toggle switch is set to [Disable].

The setting of a [Pre-group] is shown in the setting menu of the conveyor object selected by [Infeed Conveyor], as shown in the next figure.



Moreover, in the setting menu of the conveyor object selected by [Infeed Conveyor], the [Tracking Type], [Servo Conveyor] and [Tray] settings cannot be changed. The following message will be displayed if you try to change anything. Click [OK] to restore the settings.



When [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 only be set in the setting menu of the child conveyor stations of the infeed conveyor.

⚠CAUTION

If [Infeed Conveyor] is selected, the [Line Tracking Schedule] of each child conveyor station of the pre-grouping conveyor will 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 for the infeed conveyor, press the F4 [SET TRK FRM] key to start setting up a tracking frame. See "I.2.3.4. Setting a tracking frame(P.72)". Setup of the second robot's tracking frame will start when the setup for the first robot's tracking frame is finished. If robots have been added only to the pre-grouping conveyor, set up a tracking frame as well.

[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 whole parts will not overrun this upper limit as opposed to the model origin point of the vision.

Select the tool frame set up by "I.4.4. Setting up the tool frame(P.257)" as a jog feed frame, then move the TCP by jogging and tap the [Record] button to set the Y value in a tracking frame. 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 tree view.

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

2 Click [OK] to set the new value.

Click [CANCEL] to keep the previous value.

NOTE

If you click [Record] when there is no conveyor station containing the robot connected to the robot controller whose teach pendant is being used to teach, the warning IRPK-065 "Conveyor station for this controller cannot be found" 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 whole parts will not overrun this lower limit as opposed to the model origin point of the vision.

Select the tool frame set up by "I.4.4. Setting up the tool frame(P.257)" as a jog feed frame, then move the TCP by jogging and tap the [Record] button to set the Y value in a tracking frame. 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 tree view.

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

2 Click [OK] to set the new value.

Click [CANCEL] to keep the previous value.

NOTE

If you click [Record] when there is no conveyor station containing the robot connected to the robot controller whose teach pendant is being used to teach, the warning IRPK-065 "Conveyor station for this controller cannot be found" will occur.

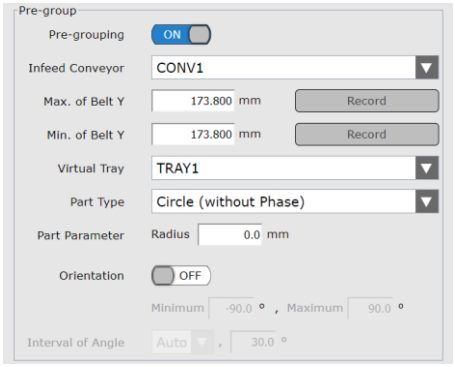
[Virtual Tray]

Select the tray which defines the arranged pattern.

[Part Type]

Select the form of the arranged part. The values that can be selected for [Part Type] are [Circle (without Phase)] and [Rectangle]. The setup screen switches as follows according to Part Type.

When [Part Type] is [Circle (without Phase)]



[Part Parameter]

Set the radius of a circle in [Radius].

CAUTION

If the radius value is smaller than the actual one, the part that 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 parts. When the form of the actual part is not the circle, you must set the radius of the circle which includes whole of the part.

NOTE

If the radius is too large for the cell interval of the tray for which the arrangement pattern is set, it may be judged that the base part and arranged parts will collide with each other when PKCSGENVRTRY is called, and error IRPK-088 “There is an invalid pre-grouping setting.” may occur.

[Orientation]

If [Circle (without Phase)] is selected, virtual trays can be generated in any direction desired.

When [Disable] is set, a pattern is generated only in one way so that the X direction of a tray frame may become the same as the X direction of a tracking frame as shown below.

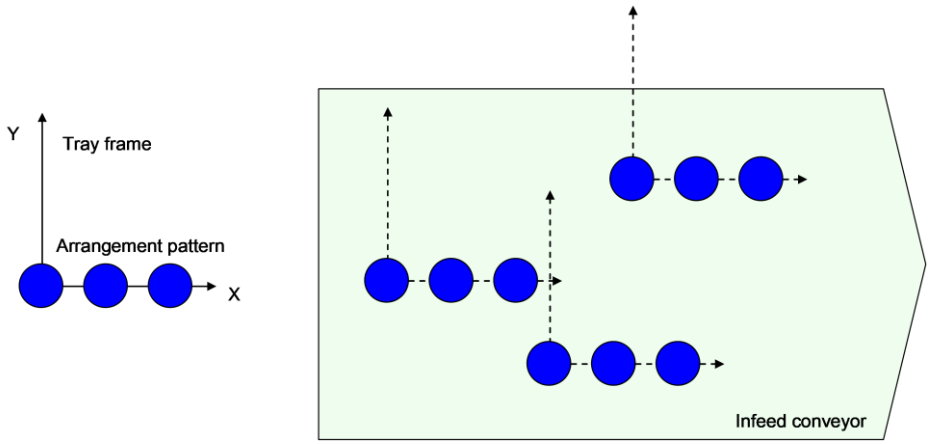


Fig. 5.15.4-1 When orientation is not [Enabled]

When [Enable] is set, the X direction of a tray frame will be searched from multiple directions. The X direction of a tracking frame is used as a standard (0 degrees), and a direction will be searched in the range that does not exceed the angle set as the [Minimum] and the [Maximum] with the interval of [Interval of Angle].

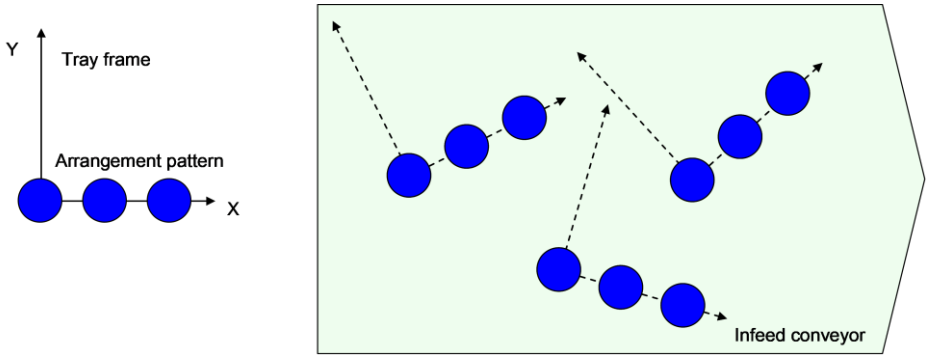


Fig. 5.15.4-2 When orientation is [Enabled]

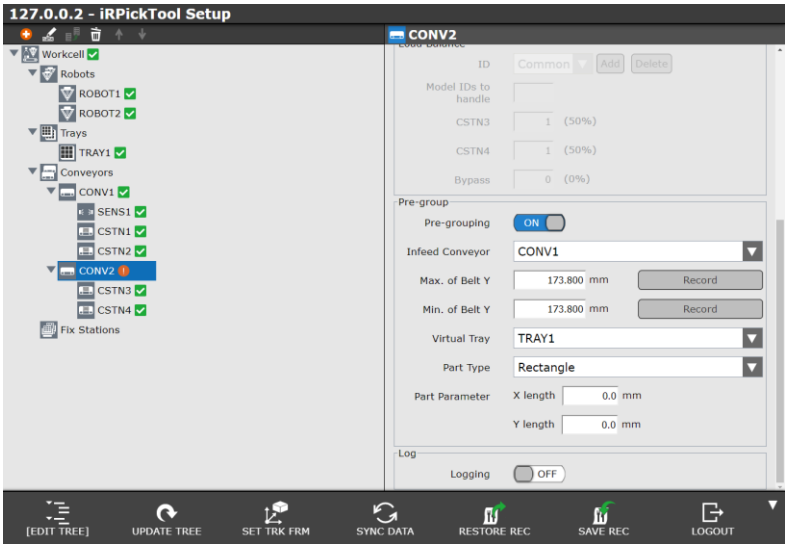
[Interval of Angle]

When [Enable] of [Orientation] is checked, [Interval of Angle] can be set. You can select [Auto] or [Specify].

When [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 [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 degrees.

When [Part Type] is [Rectangle]



[Part Parameter]

Set the length of the X side on the tray frame to [X length]. Set the length of the Y side on the tray frame to [Y length].

⚠CAUTION

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

NOTE

If the X length or Y length is too large for the cell interval of the tray for which the arrangement pattern is set, it may be judged that the base part and arranged parts will collide with each other when PKCSGENVRTRY is called, and error IRPK-088 “There is an invalid pre-grouping setting.” may occur.

This completes the setup of the pre-grouping conveyor.

Finally, set up the conveyor object of the most downstream robot which picks the arranged parts at one time if it places the picked parts on a conveyor.

Since all the parts on a tray are placed in one operation even when filling a part on a tray that is carried through a conveyor, select [Not used] for [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.

See "I.5.4. Setting up a conveyor(P.276)" for the other items.

5.15.5. Setting up 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 "I.5.5. Setting up a sensor_2(P.288)".

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 vision, the part that the robot picks and places for arrangement may collide with the part that is not detected.

NOTE

In the child sensor object of the infeed conveyor used for the arrangement on the same conveyor, when you select [Distance] as the trigger condition and select [Find part by vision] for [Trigger Action], set [Trigger Distance] to be as short as possible in order to prevent non-detection while following the CAUTION for [Trigger Distance(P.291)].

Next, if the position detected by vision or the photo eye does not match the center position of the actual part, the part that the robot picks and places for arrangement may collide with the part on a conveyor due to incorrect calculation of the interference check between both kinds of parts. When you see "I.2.3.7.2. Vision process(P.99)" and "I.5.5.3.1. When a tray is not used(P.303)", be careful when teaching the detection point.

⚠CAUTION

1. When you select [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 to be placed for arrangement may collide with the part on a conveyor because the interference check between the part on the conveyor and the part to be placed may not be calculated correctly.
2. When you select [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 the center position of a part, not the end of a part on the downstream side (which is equal to the position detected with a photo eye). If the touch-up position does not match the center position of a part, the part to be placed for arrangement may collide with the part on a conveyor because the interference check between the part on the conveyor and the part to be placed may not be calculated correctly.

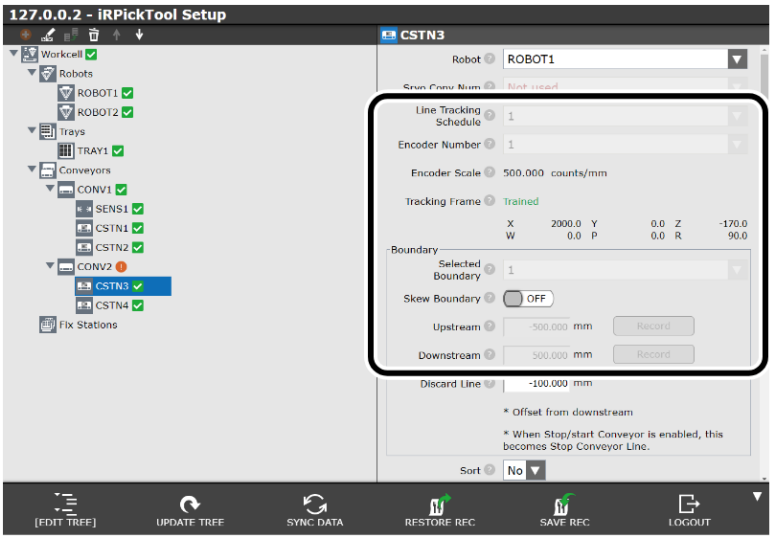
Additionally, if the position is not detected by the vision system and [Rectangle] is selected as the [Part Type] at the pre-grouping conveyor, you should pay attention to the order of the settings. Setting the sensor position or the target position must be performed after setting the [Virtual Tray Position]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

⚠CAUTION

When [Find part by vision] is not selected as the [Trigger Action] in the child sensor object of the infeed conveyor used for the arrangement on the same conveyor and [Rectangle] is selected 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.

5.15.6. Setting up 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 that judges whether the robot should give up the placement position on the generated a virtual tray flowing 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, see “[Discard Line](P.321)”.

[Sort]

The sort function cannot be used at the child conveyor station of the pre-grouping conveyor. When the [Pre-grouping] toggle switch is set to [Used] 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.

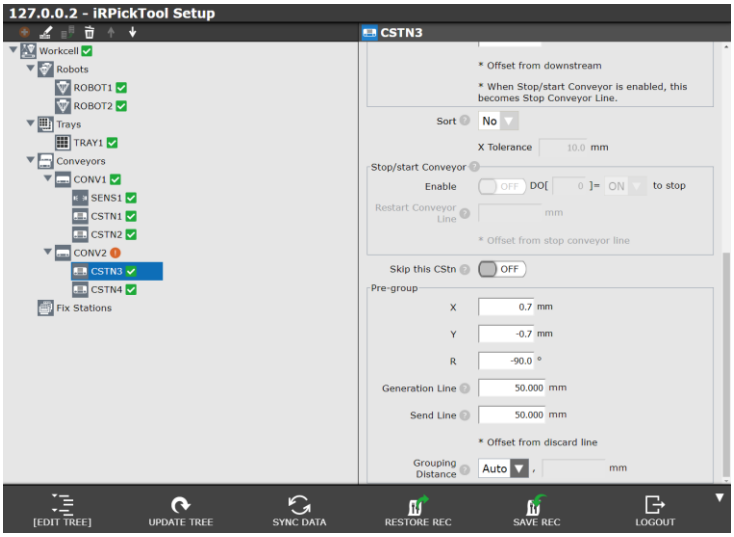
[Skip this CStn]

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 are also displayed on the child conveyor station where the same robot is selected for the 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.

If [Rectangle] is selected as the [Part Type] at the parent conveyor, the 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.

If [Circle (without Phase)] is selected as the [Part Type] at the parent conveyor, this item is not displayed because it is computed automatically.

If [Rectangle] is selected as the [Part Type] at the parent conveyor, perform the procedure described below.

CAUTION

If you set [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. **On the pre-grouping conveyor setup page, temporarily set the [Pre-grouping] toggle switch to [Disable] in [Pre-group]. (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 [Find part by vision] is set as the [Trigger Action] of the sensor in the infeed conveyor, set the tray position. See “I.5.5.5. Setting the tray position(P.306)”.
   - If you set the [Put part in queue] as the [Trigger Action] of the sensor in an infeed conveyor, set the sensor position when [DI] or [HDI] is set as trigger condition and set the target position when [Distance] is set as trigger condition. For details, see “I.2.3.6. Setting the sensor position(P.84)” or “I.5.5.6.2. When a tray is used(P.314)”.

   Here, touch up the origin or the X direction point of the tray frame as if the detected part is on cell 1 of the tray.

   CAUTION

   When touching up these points, you must consider that the detected part is on cell 1 of the tray. Otherwise, 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 them.
4. **After the guide completion, make a note of X, Y and R of the tray position, the sensor position or the target position.**
5. **In the infeed conveyor setup page, select [Not used] in [Tray] again.**
6. **On the pre-grouping conveyor setup page, set the [Pre-grouping] toggle switch to [Enable] in [Pre-group].**

   Then, click [OK] 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 [Find part by vision] as the [Trigger Action] of the sensor in an infeed conveyor, set this line beyond the end of the downstream side of the camera view. If you set [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.

- If [Circle (without Phase)] is selected as the [Part Type] at the parent conveyor, the value is the radius of the circle set for [Part Parameter].
- If [Rectangle] is selected as the [Part Type] at the parent conveyor, the value is half the length of the diagonal line on the rectangle. This value is computed from the value(x) set in [X length] of [Part Parameter] and the value(y) set in [Y length] of [Part Parameter] as shown in Fig. 5.15.6-1.

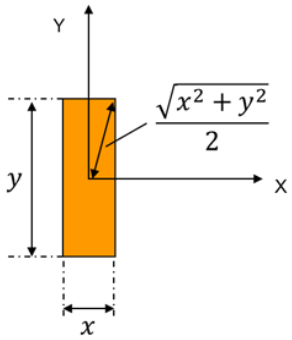
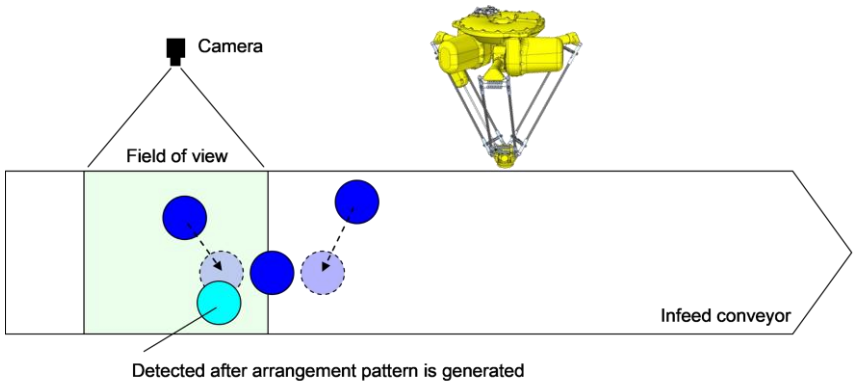


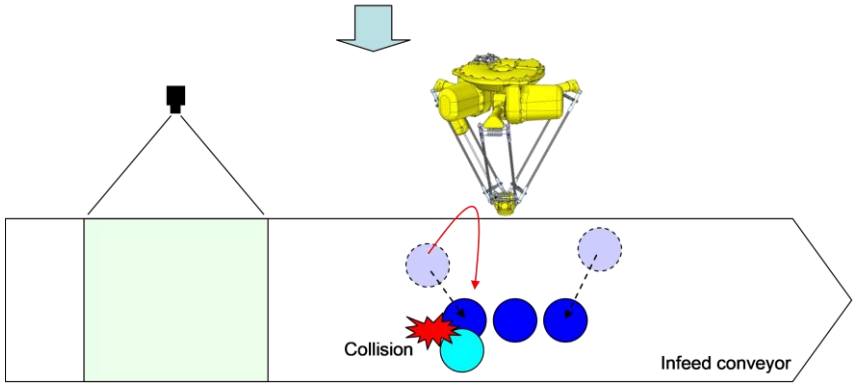
Fig. 5.15.6-1 Calculation of the generation line

CAUTION

1. When using a vision system at an infeed conveyor, if a pattern is generated in a camera view, a new part may be detected at a position that interferes with 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 system at an infeed conveyor, a new part may be detected at a position that interferes with 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.

⚠CAUTION

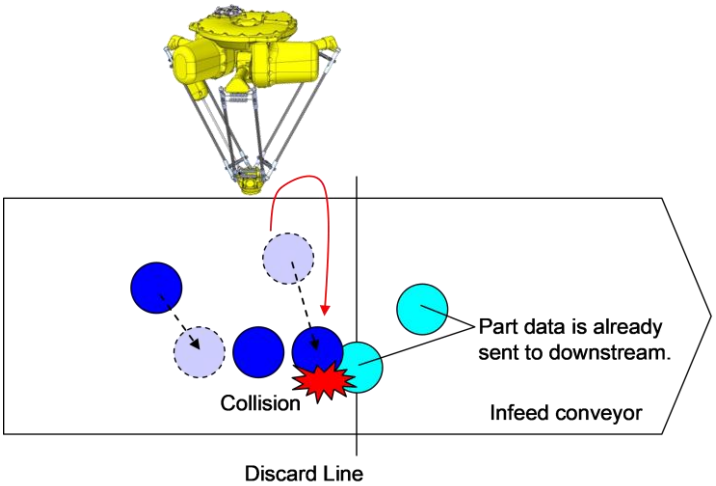




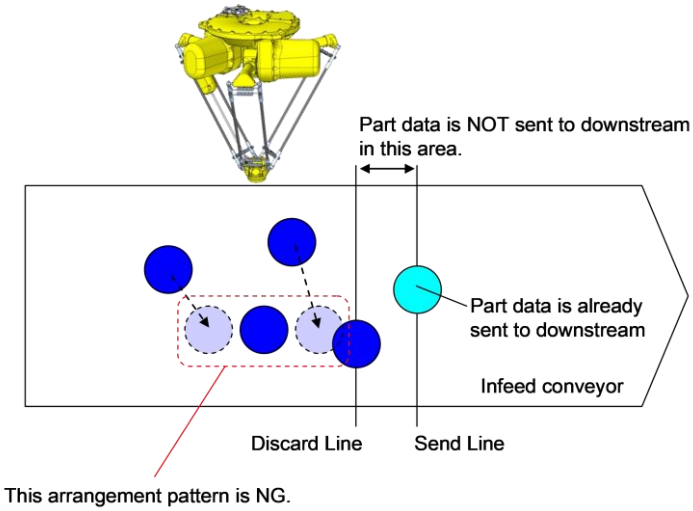
[Send Line]

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 [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 [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 shown below.



Part information is held until the Send line is passed even after passing the discard line so that the interference can be checked correctly for 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.

If [Circle (without Phase)] is set as the [Part Type], specify the radius of a part or larger value. If [Rectangle] is set as the [Part Type], specify half the length of the diagonal line on the rectangle or a larger value. For details, see Fig. 5.15.6-1.

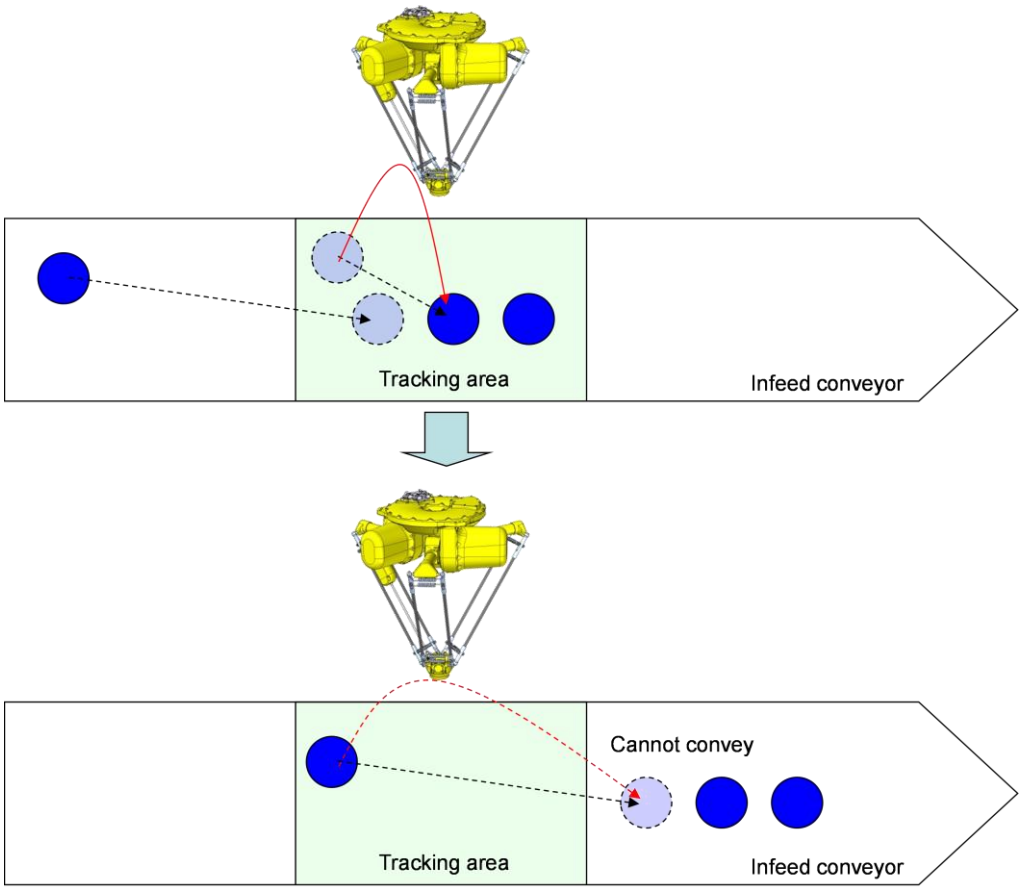
⚠CAUTION

If the send line is not set correctly, the part that the robot picks and places for arrangement may collide with another part 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.

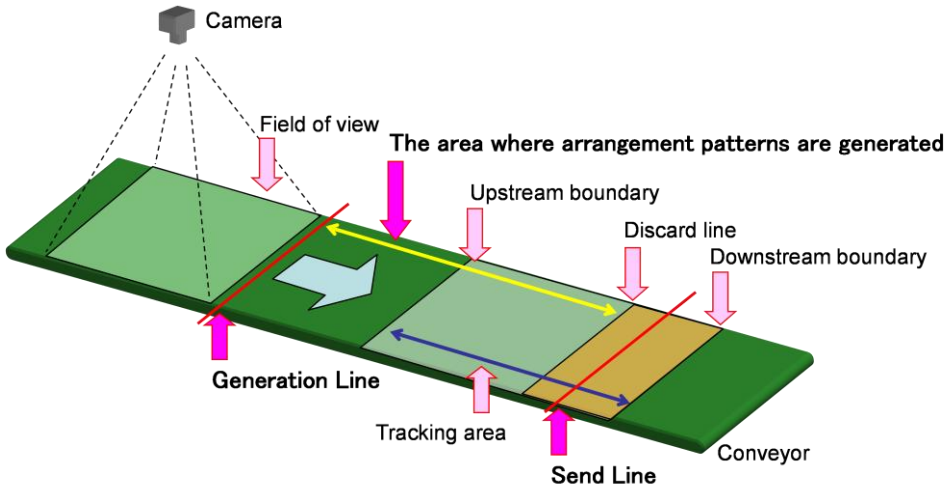


When [Auto] is selected, the distance between the discard line and the upstream boundary of the child conveyor station of the infeed conveyor is used.

When [Specify] is selected, the distance specified in the text box is used.

[Spatial relationship of each item]

The spatial relationship of the “Generation Line”, “Send Line”, “Upstream boundary”, “Downstream boundary” and “Discard Line” is summarized as the following figure. However, as a result of the setup according to this manual, the “Send Line” may be farther downstream than the “Downstream boundary”.



5.15.7. Setting up a fixed station

When you use a fixed station, see “I.5.7. Setting up a fixed station(P.332).”

5.15.8. KAREL program

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

5.15.8.1. 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 6.8.5.1 “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 a conveyor station that does not belong to a pre-grouping, it will cause error IRPK-087 “This is dedicated instruction for pre-grouping.” Moreover, when you specify the conveyor station for which the corresponding conveyor station of the infeed conveyor is not placed just behind the sensor object, it will cause error 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.

Usage:

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[4].

```
CALL PKCSGENVRTRY(CStn ID=R[12:CStnIDOut],Stat Reg=4)
```

5.15.8.2. 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 piece of 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 6.8.5.1 “PKCSGETID.PC”. You can specify only the conveyor station of the pre-grouping conveyor. When you specify a conveyor station that does not belong to a pre-grouping, it will cause error 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 argument 2, it will cause error 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.

Usage:

Information on unprocessed parts and the cell information of the corresponding placement position are acquired from the conveyor station of the infeed conveyor and the conveyor station of the pre-grouping conveyor. The ID of the conveyor station of the pre-grouping conveyor is stored in R[12]. The part information, the cell information of the corresponding placement position, and the processing status are stored in VR[1], VR[2], and R[3], respectively.

```
CALL PKCSGETQPRGR(CStn ID=R[12:CStnIDOut],Pick Offset VR=1,Drop Offset VR=2,
Stat Reg=3)
```

5.15.8.3. 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 6.8.5.1 “PKCSGETID.PC”.
You can only specify the conveyor station of the pre-grouping conveyor or the infeed conveyor. If you specify a different conveyor station, it will cause error 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 indefinitely. 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.

Usage:

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 cell 5 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)
```

5.15.8.4. PKCSACKQPRGR.PC

After the processing status of PKCSGETQPRGR becomes 0 and the part and 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 6.8.5.1 “PKCSGETID.PC”.

You can specify only the conveyor station of the pre-grouping conveyor. If you specify a conveyor station that does not belong to a pre-grouping, it will cause error IRPK-087 “This is dedicated instruction for pre-grouping.”

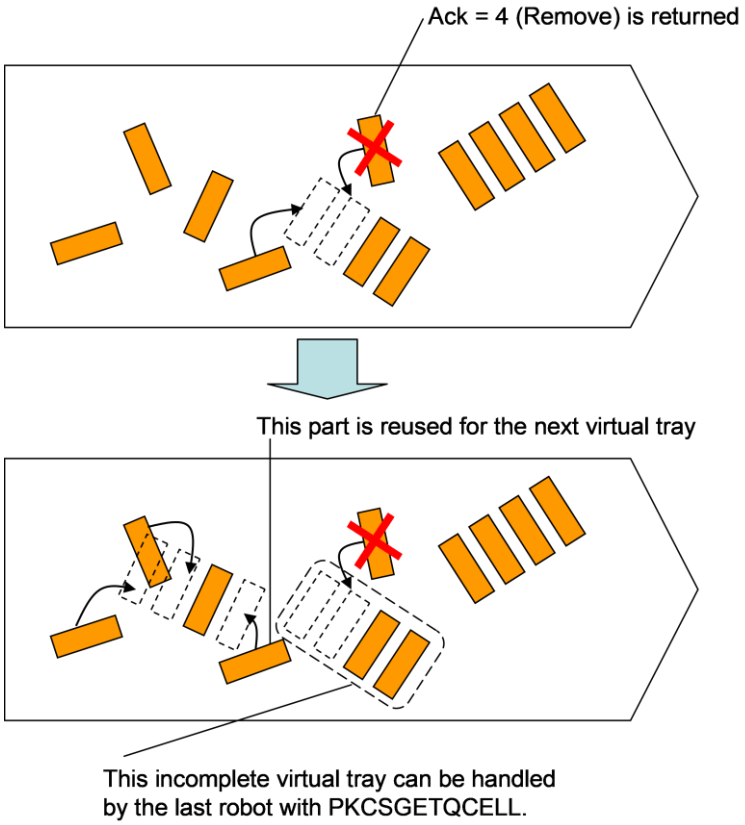
Argument 2: Specify one of the following values to indicate the processing result.

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.



Usage:

When the ID of the conveyor station is stored in R[12], the following instruction returns that the part and the cell to be placed have been processed successfully.

```
CALL PKCSACKQPRGR(CStn ID=R[12:CStnIDOut],Success)
```

5.15.9. 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.

5.15.9.1. Sample programs for the conveyor-to-same-conveyor configuration (the first robot)

This subsection contains a set of sample programs for the first robot whose position is immediately after a sensor on a conveyor.

Unlike in the case of the general tracking system, pre-grouping tracking system uses the following two types of robot programs.

- Main program
- Program that performs arrangement motion on the same conveyor (pre-grouping program)

The program that performs arrangement motion is a tracking program called from a main program.

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

Table 5.15.9.1-1 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 5.15.9.1-2 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',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 PKCSGENVRTRY(CStn ID=R[12:CStnIDOut],Stat Reg=4)
13: IF R[4:GENVRTRY STATUS]=0,JMP LBL[200]
14: IF R[4:GENVRTRY STATUS]=1,JMP LBL[300]
15: IF R[4:GENVRTRY STATUS]=2,JMP LBL[400]
16: WAIT .02(sec)
17: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
18: JMP LBL[100]
```

```
  19:
  20: LBL[200]
  21: CALL PREGROUPING1
  22: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
  23: JMP LBL[100]
  24:
  25: LBL[300]
  26: CALL PKCSGETQUE(CStn ID=R[11:CStnIDIn],Consec Flag=1,Timeout(ms)=0,
      Offset VR=1,Stat Reg=2)
  27: IF R[2:GETQ Status]<>0,JMP LBL[100]
  28: CALL PKCSACKQUE(CStn ID=R[11: CStnIDIn],Skip)
  29: JMP LBL[100]
  30:
  31: LBL[400]
  32: CALL PICKCS
  33: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
  34: CALL DROPFS
  35: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
  36: JMP LBL[100]
  37:
  38: LBL[900]
  39: CALL PKWCEND
[End]
```

A virtual tray is generated on line 12. If a virtual tray is generated, the program jumps to line 20 and calls PREGROUP1.TP which performs arrangement motion on a conveyor. PREGROUP1.TP ends only after performing the 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 discards the 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 handle all the parts, the program discards 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 will discard some parts even if the robot has a long motion path between the conveyor and the fixed station. See “I.5.9.3. Sample programs for the conveyor-to-fixed station configuration(P.373)” for the details about PICKCS.TP and DROPFS.TP.

When the processing status is 3, that is, the number of parts that have passed the generation line is less than that in the arranged pattern, the program repeats the instruction to generate a new virtual tray and the wait instruction until enough parts have passed the generation line.

Pre-grouping program

This program performs arrangement motion on the same conveyor.

PREGROUPING1.TP

```
  1: UTOOL_NUM=1
  2: UFRAME_NUM=0
  3:
  4: LBL[100]
  5: STOP_TRACKING
  6: CALL PKCSGETQPRGR(CStn ID=R[12:CStnIDOut],Pick Offset VR=1,Drop Offset VR=2,
     Stat Reg=2)
  7: IF R[2:GETQ Status]=0,JMP LBL[200]
  8: JMP LBL[800]
  9:
 10: LBL[200]
 11: CALL PKCSTRGPRGR(CStn ID=R[11:CStnIDIn],Timeout(ms)=100,Stat Reg=3)
 12: IF R[3:TRGPRGR STATUS]=0,JMP LBL[300]
```

```
  13: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
  14: JMP LBL[200]
  15:
  16: LBL[300]
  17: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
  18: L PR[1] 4000mm/sec CNT0 VOFFSET,VR[1]
  19: CALL VACUUM_ON
  20: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
  21:
  22: LBL[400]
  23: STOP_TRACKING
  24: CALL PKCSTRGPRGR(CStn ID=R[12:CStnIDOut],Timeout(ms)=100,Stat Reg=3)
  25: IF R[3:TRGPRGR STATUS]=0,JMP LBL[500]
  26: JMP LBL[400]
  27:
  28: LBL[500]
  29: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[2] Tool_Offset,PR[12]
  30: L PR[1] 4000mm/sec CNT0 VOFFSET,VR[2]
  31: CALL VACUUM_OFF
  32: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[2] Tool_Offset,PR[12]
  33: CALL PKCSACKQPRGR(CStn ID=R[12:CStnIDOut],Success)
  34: JMP LBL[100]
  35:
  36: LBL[800]
  37: END
  38:
  39: LBL[900]
  40: CALL PKCSACKQPRGR(CStn ID=R[12:CStnIDOut],Return)
[End]
```

On line 6, the part data and the corresponding cell data are obtained. If data are obtained successfully, the program jumps to line 10. If data are not obtained, the program jumps to line 37 and ends.

On line 11, the tracking trigger for pickup motion is set. Since the wait time is set to 100 milliseconds in the 2nd argument of PKCSTRGPRGR, a timeout occurs in 100 milliseconds if the part has not come into a tracking area when PKCSTRGPRGR is called. In that case, R[3] contains a value 1, and the program checks the cycle stop flag, calls PKCSTRGPRGR again, and waits for the part to come into a tracking area. When the cycle stop flag is 1, the program jumps to line 39, and ends after calling PKCSACKQPRGR for return the part data and corresponding cell data.

The pickup motion is performed on lines 17 to 20.

On line 24, the tracking trigger for placement motion is set. Since the wait time is set to 100 milliseconds in the 2nd argument of PKCSTRGPRGR, a timeout occurs in 100 milliseconds if the place where the picked part is placed has not come into a tracking area when PKCSTRGPRGR is called. In that case, R[3] contains a value 1, and the program checks the cycle stop flag, calls PKCSTRGPRGR again, and waits for the place where the picked part is placed to come into a tracking area.

The placement motion is performed on lines 29 to 32. The same Position Register is used as pickup motion.

PKCSACKQPRGR is called on line 33 to notify the conveyor station that the arrangement motion is complete. And then the program jumps to line 4 and obtains the next part data and corresponding cell data.

5.15.9.2. Sample programs for the conveyor-to-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 programs different from the sample programs shown in "I.5.15.9.1. Sample programs for the conveyor-to-same-conveyor configuration (the first robot)(P.414)".

Main program

This is the main program of the system.

MAIN1.TP

```
  1: R[1:CYCLE_STOP]=0
  2: CALL PKCSGETID('CONV1',CStn ID Reg=11)
  3: CALL PKCSGETID('CONV2',CStn ID Reg=12)
  4:
  5: UTOOL_NUM=1
  6: UFRAME_NUM=0
  7: J P[1:Perch Pos] 50% FINE
  8:
  9: CALL PKWCSTART
 10:
 11: LBL[100]
 12: CALL PREGROUPING1
 13: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 14:
 15: CALL PICKCS
 16: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 17: CALL DROPFS
 18: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
 19: JMP LBL[100]
 20:
 21: LBL[900]
 22: CALL PKWCEND
[End]
```

PREGROUP1.TP is called on line 12 to arrange parts on the same conveyor. PREGROUP1.TP ends only after performing the arrangement operation as much as possible. PREGROUP1.TP is similar to "Pre-grouping program(P.415)".

When there is no acquirable data and arrangement operation is ended, PICKCS.TP is called on line 15 to pick up a part discarded by the first robot. DROP.FS is called on line 17 to place the picked part on the fixed station. See "I.5.9.3. Sample programs for the conveyor-to-fixed station configuration(P.373)" 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.

5.15.9.3. 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 on another conveyor. This sub-subsection only contains the pickup motion programs different from the sample programs shown in "I.5.9.2. Sample programs for the conveyor-to-conveyor configuration(P.368)". See "I.5.9.2.1. Main program(P.369)" and "I.5.9.2.3. Placement program(P.373)" for the main program of the system and the program that performs placement motion on an outfeed conveyor, respectively.

Pick program

This section explains about a program that picks up only parts for which the alignment pattern is complete, and a program that also picks up parts for which the alignment pattern is not complete.

Example of a program that picks up only parts for which the alignment pattern is complete

PICK1.TP

```
  1: UTOOL_NUM=1
  2: UFRAME_NUM=0
  3:
  4: LBL[100]
  5: STOP_TRACKING
  6: CALL PKCSGETQUE(“CStn ID”=R[11:CStnIDIn],“Consec Flag”=1,“Timeout(ms)”=100,
       “Offset VR”=1,“Stat Reg”=2,2)
  7: IF R[2:GETQ Status]=0,JMP LBL[200]
  8: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
  9: JMP LBL[100]
 10:
 11: LBL[200]
 12: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
 13: L PR[1] 4000mm/sec CNT0 VOFFSET,VR[1]
 14: CALL VACUUM_ON
 15: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
 16: CALL PKCSACKQUE(CStn ID=R[11:CStnIDIn],Success)
 17:
 18: LBL[900]
[End]
```

The difference from the usual pick program in conveyor-to-conveyor configuration is that the model ID is specified by PKCSGETQUE on line 6. Here, model ID 1 is set to all cells in layer 1 of the virtual tray, and model ID 2 is set to the cell in layer 2. Information is acquired by specifying model ID 2 for argument 6 of PKCSGETQUE only when all the cells of layer 1 are filled.

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

Example of a program that also picks up parts for which the alignment pattern is not complete

PICK2.TP

```
  1: UTOOL_NUM=1
  2: UFRAME_NUM=0
  3:
  4: LBL[100]
  5: STOP_TRACKING
  6: CALL PKCSGETQCELL(CStn ID=R[11:CStnIDIn],Cell num=4,Timeout(ms)=100,
     Offset VR=1, Stat Reg=2, NumUnfilCel Reg=21)
  7: IF R[2:GETQ Status]=0,JMP LBL[200]
  8: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
  9: JMP LBL[100]
 10:
 11: LBL[200]
 12: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
 13: L PR[1] 4000mm/sec CNT0 VOFFSET,VR[1]
 14: CALL VACUUM_ON
```

```
 15: L PR[1] 4000mm/sec CNT100 VOFFSET,VR[1] Tool_Offset,PR[10]
 16: CALL PKCSACKQUE(CStn ID=R[11:CStnIDIn],Success)
 17:
 18: LBL[900]
[End]
```

The difference from PICK1.TP is the point of using PKCSGETQCELL instead of PKCSGETQUE on line 6. When the arranged pattern contains three parts, specify cell 4 that is added to layer 2 for pick 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].

5.15.10. 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 "I.5.10. Reference position setting(P.382)".

5.15.11. 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. See "I.5.11. Teaching the position to robots(P.385)".

5.15.12. Fine adjustment of the tracking motion

See "I.5.12. Fine adjustment of the tracking motion(P.386)". As special affairs in the case of a pre-grouping, care needs to be exercised when adjusting the tracking frame error.

5.15.12.1. Adjustment of the tracking frame error

When using ADJ_OFS, modify the program that performs arrangement motion shown in "I.5.15.9.1. Sample programs for the conveyor-to-same-conveyor configuration (the first robot)(P.414)" 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 to adjust 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
```

- For details, see “I.5.12.2. Adjustment of the tracking frame error(P.387)”.

6. Plug & Play reference

This chapter explains the details and setup procedures for Plug & Play menu items, for reference when using iRPickTool (Basic Functions + Plug & Play).

Furthermore, explanations of specific programs such as standard TP programs and KAREL programs are also given.

For the setup procedures for iRPickTool (Basic Functions + Plug & Play), see "I.3. iRPickTool (basic functions and Plug & Play) startup procedures(P.152)".

The iRPickTool screen menu and related items in this manual are as shown in the figure below.

Among the items below, "I.6.1. Workcell setup screen(P.422)" details the maximum number of Plug & Play objects, "I.6.4. Setting up a fixed station(P.434)" explains the settings for [Skip], which can only be set with Plug & Play, and "I.6.2. Setting up a gripper and a zone(P.422)" and "I.6.3. Setting up a robot operation(P.427)" are items that can only be set with Plug & Play. For other items, see the corresponding items in "I.5. Basic functions reference(P.258)".

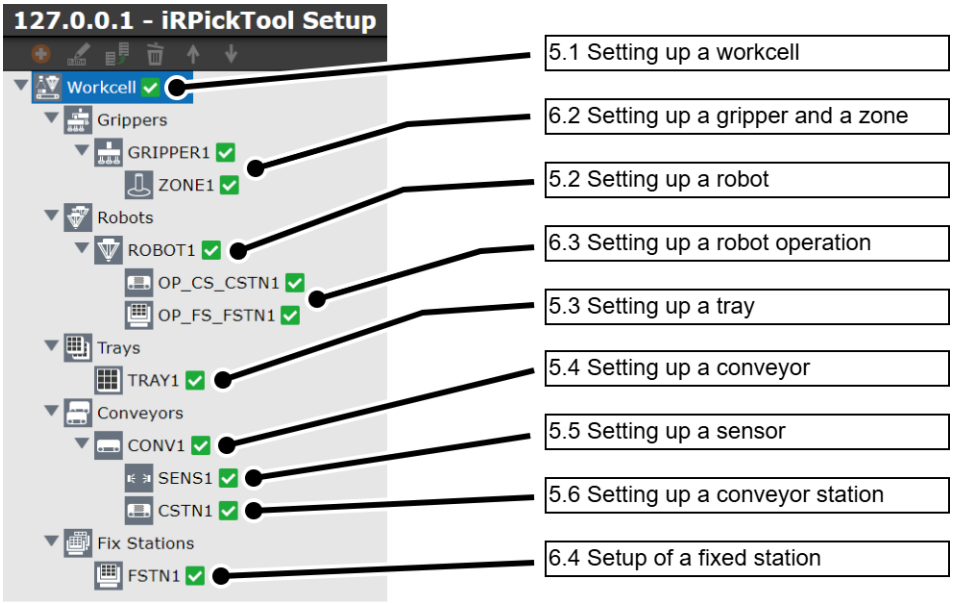


Fig. 6-1 iRPickTool menu and related items

Besides the menu items in Fig. 6-1, "I.6.5. Robot programs(P.434)", "I.6.10. KAREL program(P.444)", and "I.6.11. Standard TP programs(P.458)" explain their corresponding programs as a reference for Plug & Play. "I.6.6. Reference position setting(P.439)" and "I.6.7. Teaching the position to robots(P.439)" explain robot position teaching.

"I.6.8. Fine adjustment of the tracking motion(P.439)" describes the procedure to make error adjustments in a Plug & Play standard program in particular.

Besides these, "I.6.12. Setup of pre-grouping(P.487)" contains information specifically for the pre-grouping function.

"I.6.13. Plug & Play buffer function(P.497)" describes the setup and use of the Plug & Play buffer functions.

⚠CAUTION

As a software option for all robot controllers that are connected in the robot ring setup, order iRPickTool and not iRPickTool Basic.

Basic operations

With iRPickTool, do the setup separately for each configuration object, such as each robot and each conveyor.

For the basic operation of iRPickTool, see “Basic operations(P.258)”.

6.1. Workcell setup screen

For the workcell setup screen, see “I.5.1. Setting up a workcell(P.266)”.

Also, for the procedure to set up a Plug & Play workcell, see “I.3. iRPickTool (basic functions and Plug & Play) startup procedures(P.152)”.

Setting the number of objects

In a workcell, you can modify the maximum number of each object.

The maximum numbers that can be modified in Plug & Play are shown in Table 6.1-1.

For the procedure to modify the maximum number of each object, see “I.5.1.1. Setting the number of objects(P.266)”.

Table 6.1-1 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
Fixed station	84

6.2. Setting up a gripper and a zone

The gripper must be set up as the first step of iRPickTool setup if iRPickTool (J945) is installed. The Gripper Node is the topmost node in the tree view as shown below.

For the gripper, enter an index delay, etc. that is suitable for the pick/place operation. For the setup of a zone, which is a child object of a gripper, enter items such as tool frame, zone activation I/O, zone part presence I/O, etc.

If you enter these values correctly, you will not need to edit a standard TP program of Plug & Play.

A gripper is used by a Robot to pick or place parts.

A gripper represents the whole hand used to pick parts and its child object zone represents the section that actually picks parts. Following figure represents its concept.

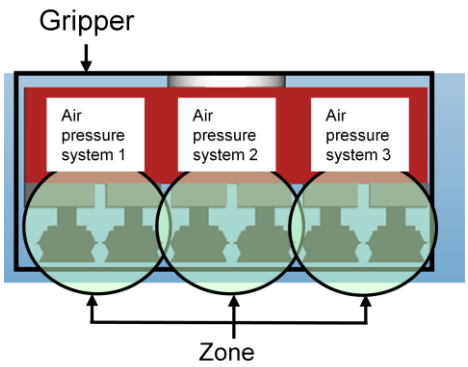


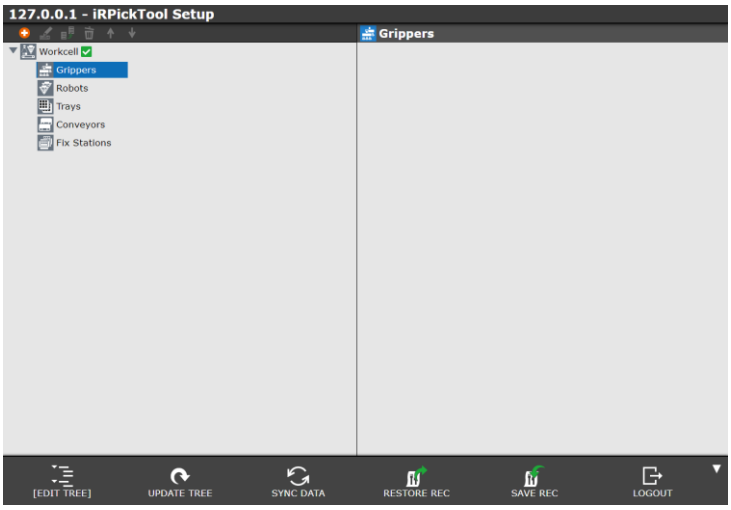
Fig. 6.2-1 Image of gripper and zones

A gripper contains up to 20 zones.

Each zone contains a UTOOL, up to 4 part presence inputs, and up to 4 zone activation I/Os. A zone can be used to pick up a part. A gripper may pick or place multiple parts using multiple zones.

Gripper data can be saved to recipes. This allows you to use different zone UTools, Pick/Place Delays, etc. for each recipe you are running without any custom programming on your part. For example, the common physical gripper shown in Fig. 6.2-1 above can be used with different settings between recipes. You can configure a recipe to put together all the paths in a single zone to pick a large part, and another recipe to set a zone to each path to pick three parts.

For details on the recipe, see "I.8.1. Exchange workcell with using recipe(P.545)".



6.2.1. Gripper and zone

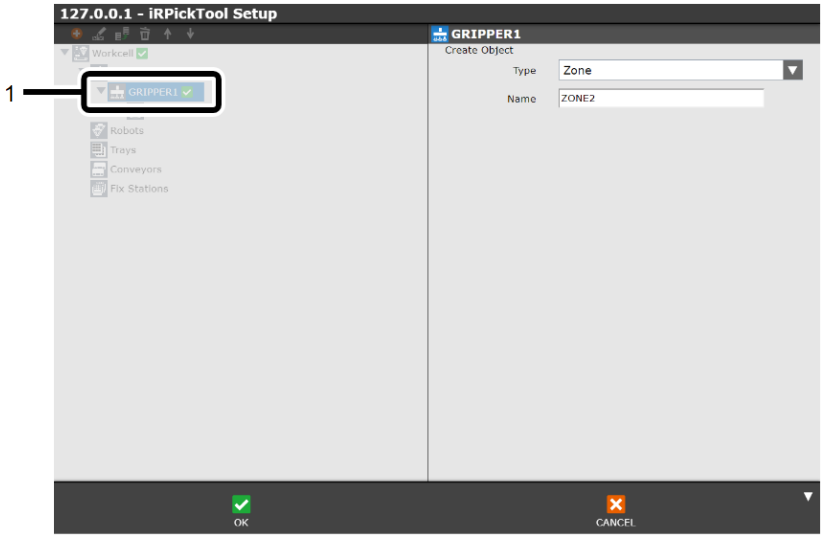
If you add a gripper, a zone will be added automatically.

Adding a gripper zone

Add a zone to the gripper using the following procedure.

1. **Select the [GRIPPER] that you want to add to the tree view.**
2. **Click the create button in the tree view, or select [CREATE] from the menu displayed by clicking [EDIT TREE].**

   The following screen appears.

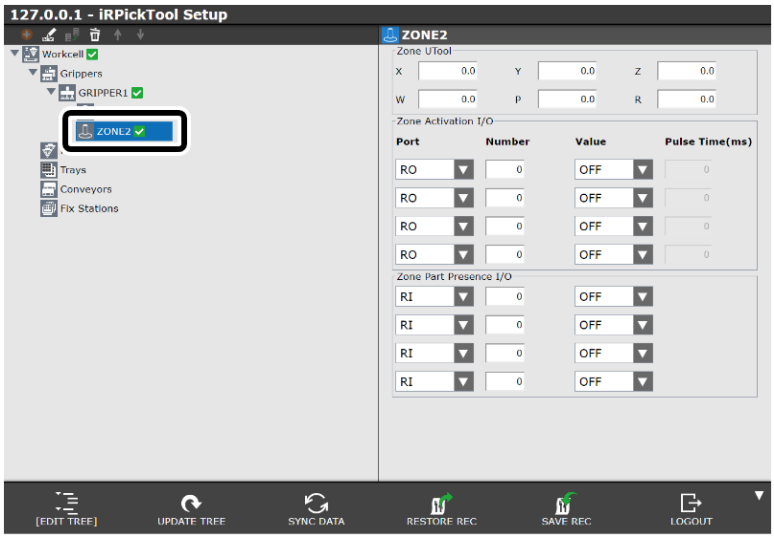


In [Name], a default name (e.g. ZONE2) is automatically provided. You can enter a different name if you wish to.

Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

3 Click [OK].

A zone will be added to the gripper.



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

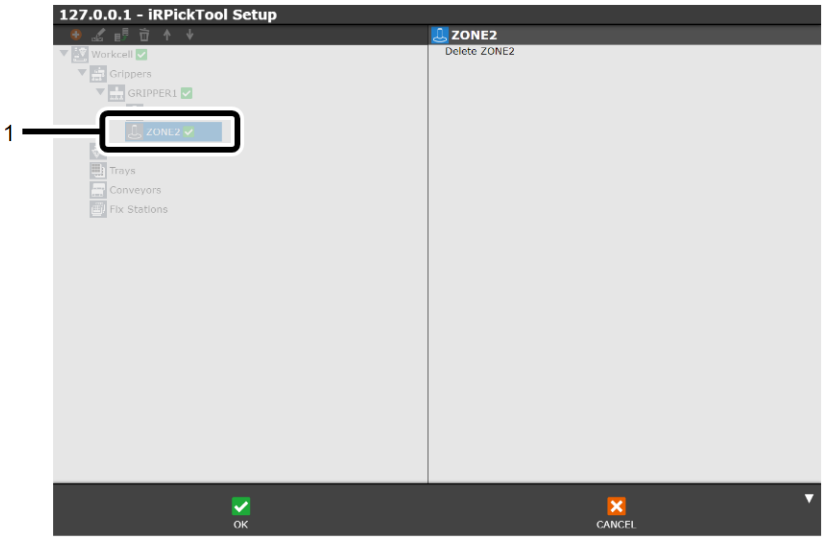
Deleting a gripper zone

Only when a gripper contains multiple zones, a zone can be deleted from the gripper. However, you can only delete the last zone of a gripper. You cannot delete the first zone since a gripper must always contain at least one zone. If you wish to delete the first zone, then delete the gripper itself.

1 In the tree view, select the [ZONE] (the last zone of a gripper) to be deleted.

2 Click the button above the tree view, or select [DELETE] from the menu displayed by clicking [EDIT TREE].

3 A confirmation message like the one shown below will appear, so click [OK].



The zone will be deleted from the gripper.

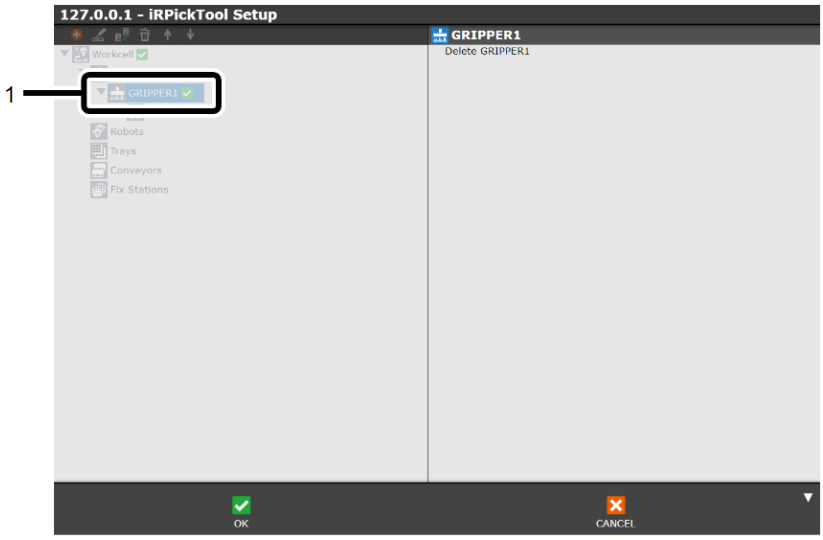
⚠CAUTION

If you try to delete a zone that is not the last one in the list, you will get a popup message saying “Only the last zone among multiple zones can be deleted.”

Deleting a gripper

A gripper can be deleted from the grippers collection.

1. **Select the [GRIPPER] that you want to delete from the tree view.**
2. **Click the button above the tree view, or select [DELETE] from the menu displayed by clicking [EDIT TREE].**
3. **A confirmation message like the one shown below will appear, so click [OK].**

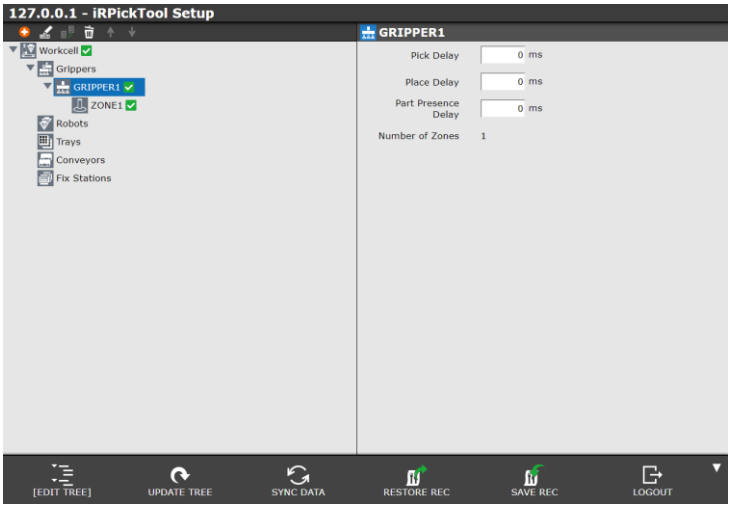


The gripper and all its zones will be deleted.

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.

[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.

iRPickTool calls PKGRGETDPDLY.PC to read the placement delay and set it to a register during production.

[Part Presence Delay]

Part presence check delay is the time after the zone activation I/O has been triggered 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.

iRPickTool calls PKGRCHKPP.PC to grip parts during production. PKGRCHKPP.PC queries and waits for the delay until the start of 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 a majority of the applications some small delay is usually required to pick and place the parts accurately.

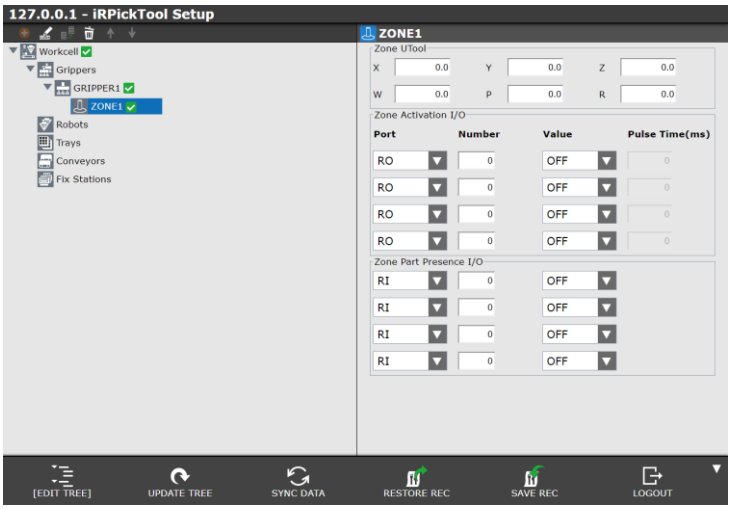
[Number of Zones]

The number of zones in the gripper is displayed here as a read-only property. This cannot be changed. 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.



[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.

MEMO

Because the tool frame data in the teach pendant is overwritten by values in zone settings, it is important to enter each value of the tool frame in iRPickTool zone settings.

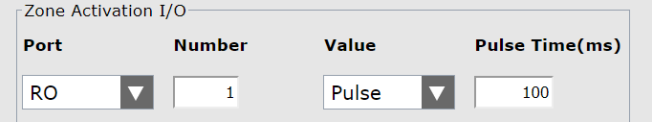
iRPickTool provides three macros for utilizing the zone UTOOL data. PKGRGETZNCNT puts the total zone count for a specified gripper in a Numeric Register.

PKGRGETZNUT puts the specified gripper's specified zone's UTOOL in a Position Register.

PK_GR_GET_UTOOLS1.TP, which uses both PKGRGETZNCNT.PC and PKGRGETZNUT.PC, gets the UTOOL specified for each zone and puts it in 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. To enable the zone activation I/O, select DO or RO in [Type] and enter a number other than 0 in [Number]. Up to four 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 milliseconds. To disable the zone activation I/O, set 0 in [Number] on the row.



The condition you specify in each row for the Zone Activation I/O is for the "Gripper Close" or "Pick part" operation. For example, assume that you specify RO[1] = ON while PKGRCLOSE.PC is executed during production. Then, the Robot Output RO[1] will be turned on when the PKGRCLOSE macro is executed. RO[1] will be turned off when the PKGROPEN macro is executed. In contrast, assume that you specify RO[1] = OFF while PKGROPEN.PC is executed during production. Then, the Robot Output RO[1] will be turned off when the PKGRCLOSE macro is executed. RO[1] will be turned on when the PKGROPEN macro is executed.

When handling large parts, it is important to enable multiple zone activation I/Os in a zone.

[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 a production run. If you specify RI[1] = OFF, then RI[1] will be checked for the OFF condition during gripper pick operation and the ON condition during gripper place operation.

When handling large parts, it is convenient to enable multiple zone activation I/Os in a zone.

6.3. Setting up a robot operation

An operation means the picking/placing motions that the robot executes at each station.

For stations, there are [Conveyor Stations] and [Fixed Stations], and for each station, there are the corresponding operations under the robot in the tree view.

For operations, there are setting items such as the type of task performed at the station (picking/placing), whether to pick up parts one by one or pick them up together in cases where there are multiple zones, and whether to execute a part present check after the operation.

If you set these correctly, you will be able to complete the task you are aiming for without editing a Plug & Play standard TP program.

The Robot Operations Setup is done after you create a conveyor station under the Conveyors node or a fixed station under the fixed stations node.

When you create a conveyor station or a fixed station, the robot operation node under the appropriate Robot in the tree is automatically created with default properties You cannot create it using the  button in the tree view toolbar or from the [EDIT TREE] CREATE function key.

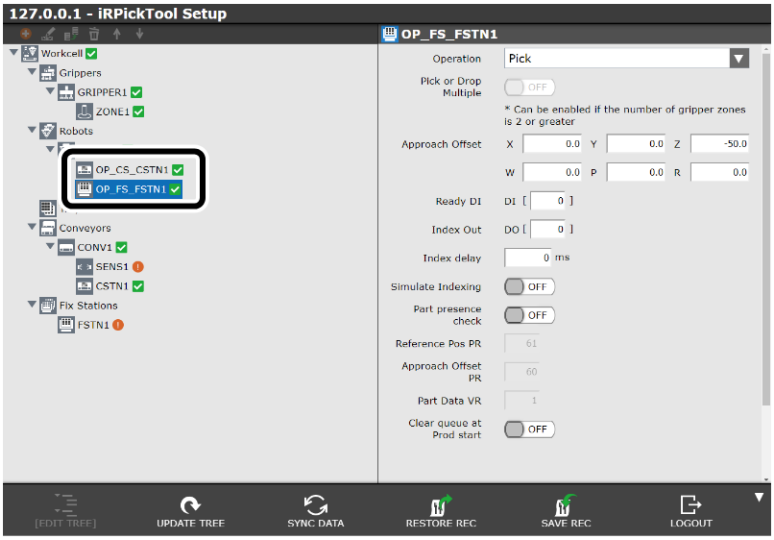
Similarly, you cannot delete a robot operation node using the  button in the tree or from the [EDIT TREE] DELETE function key.

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.



6.3.1. Conveyor station robot operation

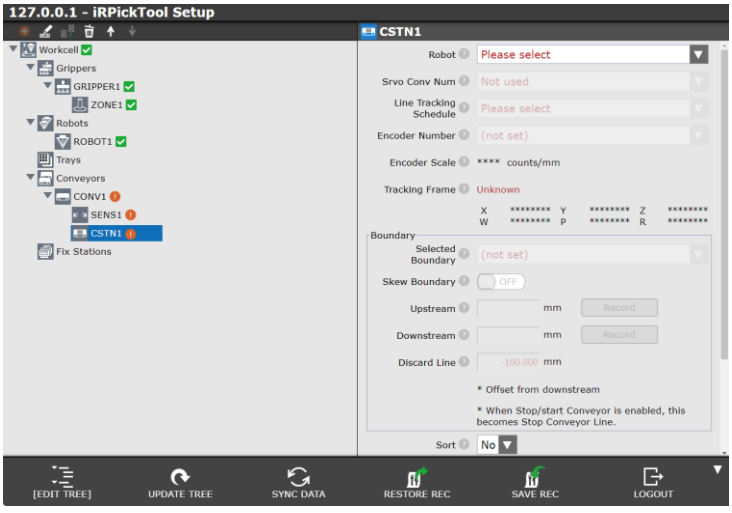
Set a Conveyor Station Robot Operation Node.

Creating a new Conveyor Station Robot Operation Node

Create a new Conveyor Station Robot Operation Node.

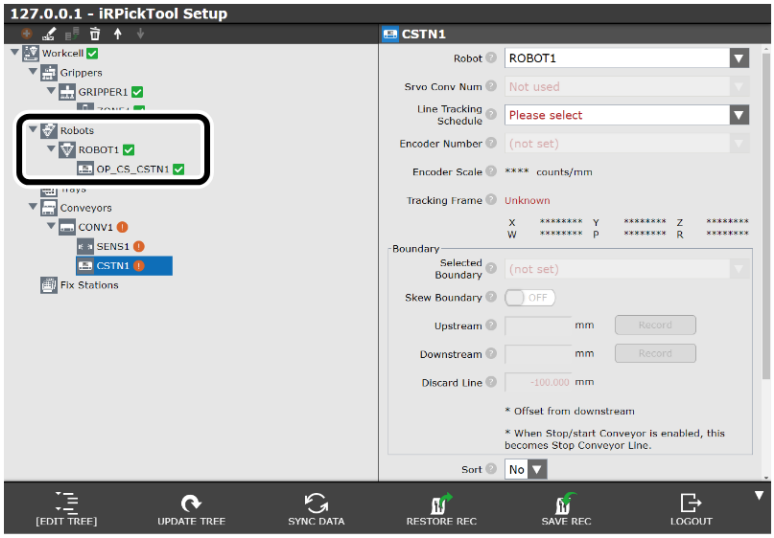
The screen below is the display screen for when creating a conveyor station.

Notice that there is no operation node under the robot yet.



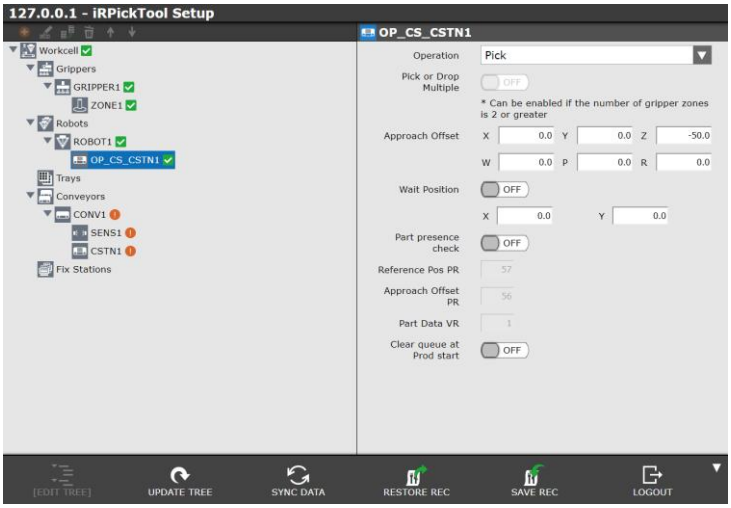
1 **Select [Robots] from among the conveyor station settings.**

Notice that as soon as you did that, the [OP_CS_CSTN1] node was immediately added under the Robot automatically.



2 Choose the operation [OP_CS_CSTN1] under the robot.

On the right hand side you will see all the properties as shown in the figure below.



6.3.1.1. Setting up conveyor station robot operation in detail

Set a conveyor station robot operation in detail.

In the tree view on the left side, select the Conveyor Station Robot Operation Node under the robot.

[Operation]

Select the “Pick” or the “Place” operation that the robot should perform at the conveyor station from the drop-down box.

[Conveyor ID]

The ID and name of the conveyor station used for this operation are displayed. This cannot be changed.

For details, see “I.6.17. Setup of multiple stations(P.515)”.

[Pick or Drop Multiple]

This item can only be set when there are two or more [Zones] for the [Gripper].

If you enable [Pick or Drop Multiple], the robot will simultaneously pick/place the same number of parts as the number of [Zones].

⚠WARNING

If you set a tray incorrectly, serious effects such that the collision between a robot gripper and the tray can occur when a robot places multiple parts into the tray or picks up multiple parts from the tray simultaneously by using a hand that can hold multiple parts. See "Setting of a tray when a robot hand that can hold multiple parts is used(P.272)".

It will pick the first part from zone 1 of the gripper and then go to the retreat position (or the Wait position if it is enabled and there is no part to pick). Then when the next part becomes available, it will go back to the pick position, pick the part and then return to the retreat position (or the Wait position if it is enabled and there is no part to pick). This continues until the last part has been picked.

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

You can use PKCSGETPMUL.PC to retrieve the quantity value to a register.

[Approach Offset]

Enter the x, y, z, w, p, r offset of the approach position relative to the pick/place position. In most applications, you will only use the Z offset because the approach position is in line with the pick or place position.

You can use PKCSGETAPPR.PC to assign the approach offset value to 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/place. The wait position is a position at the upstream boundary of the conveyor station at the height of approach offset.z value from the pick/place position. This position is calculated automatically by 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]

Set the shift amount for the wait position in millimeters. Internally in iRPickTool, the shift is along the track frame X of that Conveyor station. Enter a positive value in millimeters to shift the Wait Position downstream along the conveyor. A negative value will move the Wait position toward the upstream of the conveyor.

[Wait Position Y]

Enter a value in millimeters to shift the Wait Position perpendicular to conveyor flow direction. Internally in iRPickTool, the shift is along the track frame Y of that conveyor station.

[Part Presence Check]

Enable or disable the part presence check when the robot picks or places a part at this conveyor station.

PKGRCHKPP.PC performs a part presence check.

⚠CAUTION

If you enable [Part Presence Check] but you have not set up the part presence I/O in the robot's gripper menu, then the part presence check will not be done.

[Reference Pos PR]

The reference position is recorded in the Position Register with this number. This cannot be changed. For details on the Reference Position, see “I.3.3.12. Setting up a reference position and teaching a robot position(P.230)”.

[Approach Offset PR]

iRPickTool stores an approach offset in the Position Register with the number displayed here when a standard TP program is executed. This cannot be changed. An approach offset is used as a tool offset.

[Part Data VR]

iRPickTool acquires part information from the queue when a standard TP program is executed and stores it in the Vision Register with the number displayed here. This cannot be changed.

[Clear queue at Prod start]

Enable or disable the deletion of queue information from the station when production starts. If this is disabled, the robot tries to operate on parts remaining in the queue when production starts.

6.3.2. Fixed station robot operation

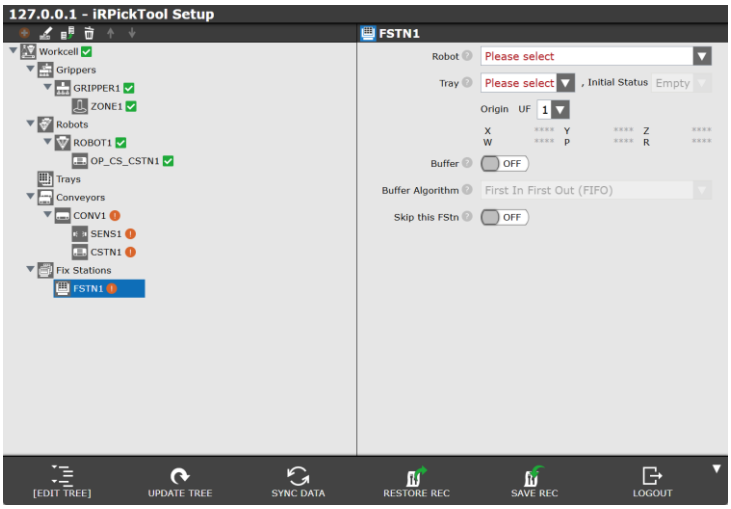
Set a Fixed Station Robot Operation Node.

Creating a new Fixed Station Robot Operation Node

Create a new Fixed Station Robot Operation Node.

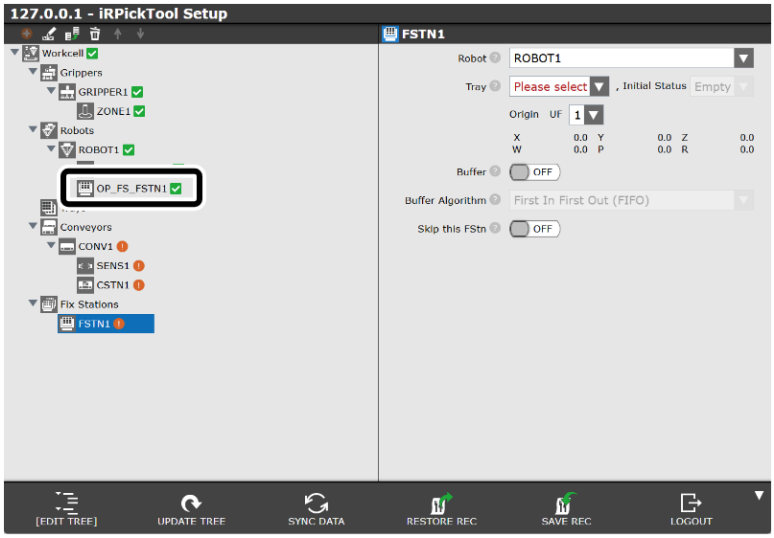
The screen below is the display screen for when creating a fixed station.

Notice that there is no fixed station operation node under the robot yet.



1 **Select [Robot] from among the fixed station settings.**

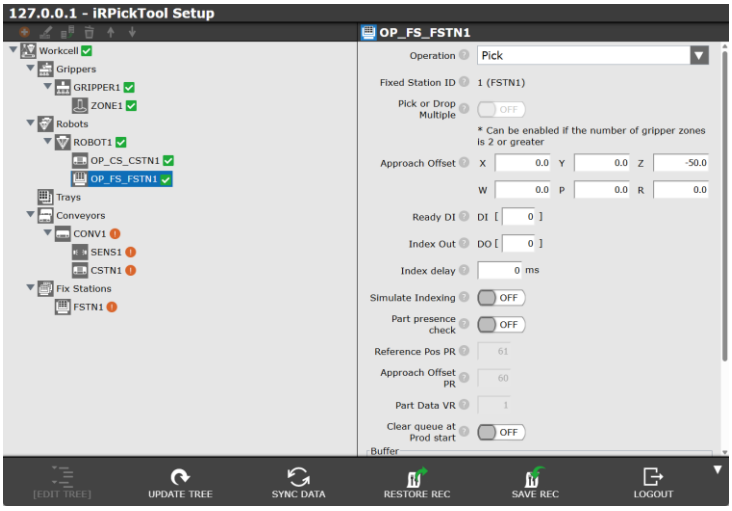
Notice that as soon as you did that, the [OP_FS_FSTN1] node was automatically added under the robot.



2 Choose the operation [OP_CS_FSTN1] under the robot.

On the right hand side you will see all the properties as shown in the figure below.

The buffer frame will be displayed only if R827: iRPickTool Production Control is installed. For details, see “I.6.13. Plug & Play buffer function(P.497)”.



6.3.2.1. Setting up fixed station robot operation in detail

Set a fixed station robot operation in detail.

In the tree view on the left side, select the fixed station robot operation node under the robot.

[Operation]

Select the “Pick” or the “Place” operation that the robot should perform at the fixed station from the drop-down box.

[Fixed Station ID]

The ID and name of the fixed station used for this operation are displayed. This cannot be changed.

For details, see “I.6.17. Setup of multiple stations(P.515)”.

[Pick or Drop Multiple]

This item can only be set when there are two or more [Zone] for the [Gripper].

If you enable [Pick or Drop Multiple], the robot will simultaneously pick/place the same number of parts as the number of [Zone].

Every time the robot picks a part, it goes to the retreat position and waits for the next part. Then when the next part becomes available, it will pick the part and then return to the retreat position. This continues until the last part has been picked.

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

You can use PKFSGETPMUL.PC to retrieve multiple property values of the operation to a register.

[Approach Offset]

Enter the x, y, z, w, p, r offset of the approach position relative to the pick/place position. In most applications, you will only use the Z offset because the approach position is in line with the pick or place position.

You can use PKFSGETAPPR.PC to assign the approach offset value to the special 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. For example, the DI set here is set to high when the tray delivery is completed and the robot is ready for operation in the system where the tray is delivered to the determined position before the robot operates. Set an appropriate DI number when using this function.

You can use PKFSGETRDYDI.PC to retrieve the port number value for the ready DI to a register.

To not use this function, set the port number to 0.

[Index Out]

In a fixed station, you can set the DO to notify the completion of pick-and-drop by the robot. For example, set this DO to high to eject the tray when the robot completes work in the system where the tray on which the robot has worked is ejected and another tray is delivered to the same position when the robot completes work.

Design a system that sends the station on which the robot has worked (Ready DI described above must be off during this period) and prepares a new empty station on which the robot should work if this DO goes high.

You can use PKFSGETIXODO.PC to retrieve the Index DO value to a register.

To not use this function, set the port number to 0.

[Simulate Indexing]

If you enable Indexing simulation, then after entering valid I/O values for the Ready DI and Index Out signals, iRPickTool will automatically simulate indexing using the 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.

You can use PKFSGETIXSIM.PC to simulate the initial state of the ready DI.

[Index Delay]

Specify the Index delay for simulation of indexing in milliseconds. This is the elapsed time after Ready DI temporarily falls and then rises again.

The value is used only if simulation of indexing is enabled.

[Part Presence Check]

Enable or disable the part presence check when the robot picks or places a part at this fixed station.

⚠CAUTION

If you enable [Part Presence Check] but you have not set up the part presence I/O in the robot's gripper menu, then the part presence check will not be done.

[Reference Pos PR]

Directly teach the reference position to the Position Register with this number. This cannot be changed.

[Approach Offset PR]

iRPickTool stores an approach offset in the Position Register with the number displayed here when a standard TP program is executed. This cannot be changed. An approach offset is used as a tool offset.

You can use PKFSGETAPPR.PC to assign the approach offset value to this dedicated position register.

[Part Data VR]

iRPickTool acquires part information from the queue when a standard TP program is executed and stores it in the Vision Register with the number displayed here. This cannot be changed.

[Clear queue at Prod start]

Enable or disable the deletion of queue information from the station when production starts. If this is disabled, the robot tries to operate on parts remaining in the queue when production starts.

6.4. Setting up a fixed station

Set up a fixed station.

This section explains the settings for [Skip], which can only be set with Plug & Play. For other settings, see "I.5.7. Setting up a fixed station(P.332)".

[Skip this FStn]

If you set [Enable], the robot does not work in this fixed station. If work on a specific fixed station becomes impossible due to reasons such as peripheral equipment failure when the robot works on multiple fixed stations, you can use this function to skip work on that fixed station.

6.5. Robot programs

A set of standard TP programs is provided for Plug & Play. If you execute the main program, PK_MAIN1.TP, the necessary robot operations are performed based on the data you set in "I.6.1. Workcell setup screen(P.422)" to "I.6.4. Setting up a fixed station(P.434)".

Also, with regard to the task of actually teaching the robot position, carry it out according to "I.3.3.12. Setting up a reference position and teaching a robot position(P.230)" for conveyor stations, and "I.5.10. Reference position setting(P.382)" and "I.5.11. Teaching the position to robots(P.385)" for fixed stations.

⚠CAUTION

iRPickTool does not support power failure handling. Power failure handling is disabled by default. After rebooting, be sure to execute the main program PK_MAIN1.TP from the beginning. Your system requirements may make it necessary to modify the robot programs.

"I.6.5.1. Robot behavior under the standard TP programs(P.435)" describes how the robot behaves with the standard TP programs.

The standard TP programs for Plug & Play exclusively use the registers: Position Registers, String registers, Vision Registers, load settings, user alarms, user frames, and user tool frames described in "I.6.5.2. Registers(P.436)" and later. Do not use these registers and settings for other purposes.

⚠WARNING
If any of the registers, Position Registers, character registers, Vision Registers, load settings, user alarms, user frames, and user tool frames exclusively intended for Plug & Play is used for any other purpose, the robot may show unexpected behavior or some other symptom may appear in the system.

6.5.1. Robot behavior under the standard TP programs

The standard TP programs assume that the robot behaves according to certain rules.
The assumed rules are described below.

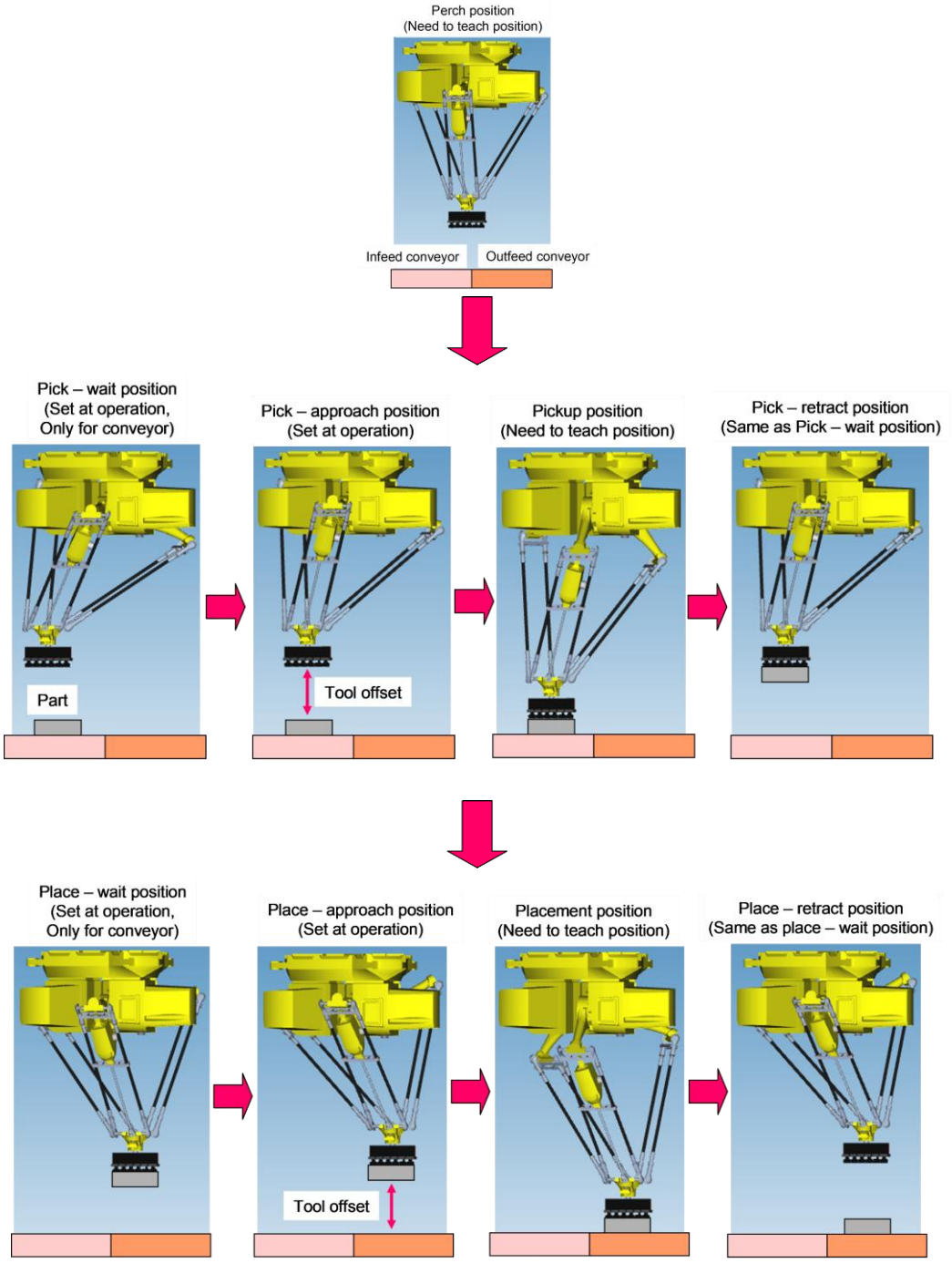


Fig. 6.5.1-1 Robot moving in accordance with a fixed rule

Once the placement motion is complete, the robot performs the pickup motion again, thus repeating the pickup and placement motions. The robot moves to the leftmost waiting position during the pickup or placement motion only

when it picks a part from a conveyor or places a part onto a conveyor. In the case of a fixed station, the robot does not move this way.

Regarding the behavior of the robot, the following three rules are particularly crucial:

1. The approach position is at the same location as the retract position.
2. The value to be set to teach the approach position is the tool offset value that is set with respect to the pickup or placement position.
3. The points that need to be taught during the entire flow are the origin, pickup position, and placement position only.

If the robot behaves in a way that violates any of these three assumed rules, modify the KAREL programs and standard TP programs as necessary after carefully reading "I.5.8. KAREL program(P.332)", "I.6.10. KAREL program(P.444)", and "I.6.11. Standard TP programs(P.458)".

6.5.2. Registers

iRPickTool uses the following Numeric Registers in the TP programs. Presently, Registers 95-230 are used.

Do not use these registers for other purposes. If you need more registers, increase the number of registers by using the Program Setup function of Controlled Start.

Table 6.5.2-1 Registers

R[95] to [98], R[100]	Temporary use registers. They are re-used in the various TP programs.
R[99]	Register for storing Active Recipe.
R[97] to [100]	Registers used commonly in all groups. These are used in programs for Group 1 and Group 2
R[101] to [108]	Registers global to group 1 only.
R[109] to [125]	Registers for the first pick operation in Group 1 (regardless of conveyor or fixed station drops)
R[126] to [127]	Registers for Group 1 Buffer operations
R[129] to [145]	Registers for the first drop operation (regardless of conveyor or fixed station drops)
R[147]	Register for Group 1 Buffer operations
R[148] to [150]	Temporary use registers. They are re-used in the various TP programs.
R[151] to [158]	Registers global to group 2 only.
R[159] to [175]	Registers for the second pick operation (regardless of conveyor or fixed station drops)
R[176] to [177]	Registers for Group 2 Buffer operations
R[179] to [195]	Registers for the second drop operation (regardless of conveyor or fixed station drops)
R[197]	Register for Group 2 Buffer operations
R[199] to [205]	Registers for Group 1 Buffer operations
R[209] to [215]	Registers for Group 2 Buffer operations
R[219] to [229]	Reserved
R[230]	Register for pre-grouping (This is used even when pre-grouping is not used. It is used for judging whether pre-grouping is used of not.)

6.5.3. Position Registers

iRPickTool uses the following Position Registers in the TP programs. Presently, Position Registers 51-90 are used.

Do not use these Position Registers for other purposes. If you need more Position Registers, increase the number of Position Registers using the Program Setup function of Controlled Start.

Table 6.5.3-1 Position Registers

PR[51]	Temporary 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

6.5.4. String Registers

iRPickTool uses the following String Registers in the TP programs. Do not use these String Registers for other purposes.

Table 6.5.4-1 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]	If the station in SR[1] is a conveyor station, name of the conveyor to which the station belongs

SR[5]	If the station in SR[2] is a conveyor station, name of the conveyor to which the station belongs

6.5.5. Vision Registers

iRPickTool uses the following Vision Registers in the TP programs. Do not use these Vision Registers for other purposes.

Table 6.5.5-1 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.

6.5.6. Payload IDs

iRPickTool uses the following payload IDs in the TP programs. Set the empty gripper payload and full gripper payload. Do not use these payload IDs for other purposes.

Table 6.5.6-1 Payload IDs

PAYLOAD[1]	Empty Gripper
PAYLOAD[2]	Full Gripper

6.5.7. User alarms

iRPickTool uses the following user alarms in the TP programs. Do not use these user alarms for other purposes.

Alarm No.	Alarm 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]

6.5.8. User frames

The TP programs use the user frames that you reference in your Fixed Station property page in the iRPickTool tree view. It does not use any dedicated user frames for internal use.

6.5.9. 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 also have 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.

6.6. Reference position setting

The reference position is the part detection position to be used when the pickup position or placement position is taught to the robot program.

Teach pick-up/placement positions to all the robots that work on the same conveyor, with regard to a part for which a reference position has been specified.

With regard to the setting of the reference position for Plug & Play, see "I.3.3.12. Setting up a reference position and teaching a robot position(P.230)" for conveyor stations, and "I.5.10. Reference position setting(P.382)" for fixed stations.

6.7. Teaching the position to robots

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

With regard to robot position teaching, see "I.3.3.12. Setting up a reference position and teaching a robot position(P.230)" for conveyor stations, and "I.5.11. Teaching the position to robots(P.385)" for fixed stations.

6.8. Fine adjustment of the tracking motion

The procedure to make fine adjustments to the robot's tracking motions with Plug & Play is the same as the procedure for the basic functions.

See "I.5.12. Fine adjustment of the tracking motion(P.386)".

This section mainly describes the procedure to make error adjustments in a Plug & Play standard program.

6.8.1. Adjustment of the tracking frame error

When a camera is used to detect parts, the tracking motion is susceptible 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 would be best 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 7.10.2.12 "PK_CV_PICK11.TP"

```
56: CALL ADJ_OFS(1,1,20,1)
57: --
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: --
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 G],
    Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone ])
74: --
```

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 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.
While the conveyor is operating, the robot cannot move even if the tracking program is run in T1 mode. In this case, you will see the alarm message, “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.**

If there is any displacement, jog the robot to the correct position. When asked about whether to subtract the offset value, select [YES].

9 **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.

10 **When the robot reaches the approach point, stop the conveyor.**

Leave the program running.

11 **When the conveyor stops completely, stop the program temporarily.**

12 **Check the amount of displacement between the robot position and the part position.**

If there is any displacement, measure the displacement in the X direction (parallel to the motion of the conveyor) and in the Y direction (orthogonal to the motion of the conveyor) 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.

13 **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.

14 **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, see "iRVision OPERATOR'S MANUAL (Reference)"(B-84434EN).

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.

6.8.2. Adjustment of the dynamic error of the robot

While the robot correctly reaches above the pickup position of a part when the conveyor is stopped, a certain amount of displacement occurs when the conveyor is moving. This error is due to a tracking delay in the robot motion.

NOTE

1. Before adjusting the dynamic error, make sure that no error occurs when the conveyor is stopped by making the error adjustments described in I.6.8.1. Adjustment of the tracking frame error(P.439).
2. If the robot works on multiple conveyors, adjust the error on the conveyor with the fastest speed.

To adjust this displacement, perform the procedure described below.

1 **Insert a motion instruction and a wait instruction immediately after the approach point so as to make the robot stop immediately above the pickup position set in the tracking program.**

Example: Standard TP program in Subsection 7.10.2.12 “PK_CV_PICK11.TP”

```
70: LBL[88]
71: --
72: L PR[57:Pk1 Cv Ref Pos] max_ speed CNT0 VOFFSET,VR[1] Tool_Offset,PR[15]
73: WAIT 10(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 G],
   Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone ])
75: --
```

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 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.
While the conveyor is operating, the robot cannot move even if the tracking program is run in T1 mode. In this case, you will see the alarm message, “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 “I.5.12.3. Adjustment of the dynamic error of the robot(P.387)”: Displacement amount (mm) ÷ conveyor speed (mm/s) × 1000 (unit: ms). The conveyor speed is that observed when the robot follows the part. The displacement amount has a positive sign when the robot is displaced from the part in the upstream direction of the conveyor and a negative sign when displaced in the downstream direction. If a value is already set in [Dynamic Error Adjustment], add the value calculated above to the existing value.**

6 **Delete the inserted motion instruction and wait instruction from the tracking program.**

Calculation example

When 30 is set in [Dynamic Error Adjustment], the conveyor speed is 500 mm/s, and the pickup position of the part is displaced by 3 mm in the upstream direction of the conveyor:

[Dynamic Error Adjustment] = +30(ms) + (3(mm) ÷ 500 (mm/s)) × 1000 = +36 (ms)

6.9. Executing production

Production can be executed once test execution of the application is complete and both accuracy and processing volume have been properly adjusted.

Refer to "I.5.14. Production runs(P.391)" for the settings common with "iRPickTool Basic (J944)". Refer to this section only for settings specific to iRPickTool (J945).

6.9.1. Starting production with the operator's panel

When you are not using a PLC and want to run iRPickTool only with the operator's panel, perform the following operations.

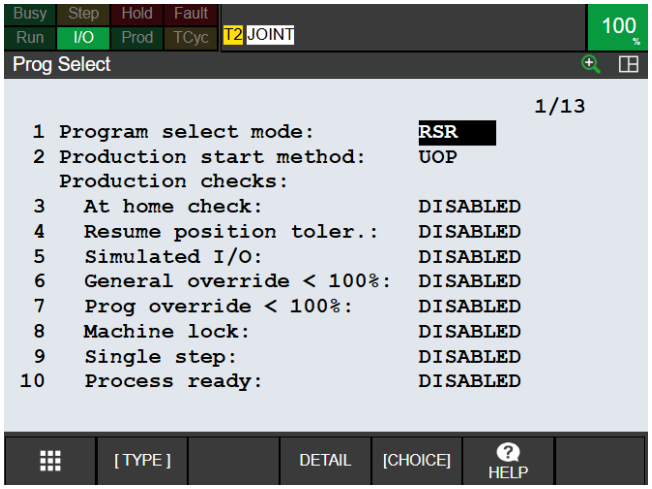
- Set the remote/local setting to local. On the teach pendant of each robot controller, press the [MENU] key, navigate to [System] → [Configuration] screen, select the "Remote/Local Setting" item, and set it to [Local].
- In each robot controller, open the [LIST] screen and select PK_MAIN1.
- Press the start switch on the operator's panel of each robot controller.

6.9.2. Starting production with peripheral equipment

Normally, this method is used in the case where production is started from a PLC in a iRPickTool workcell. The PLC and robot controller are connected via the peripheral equipment I/O signal. Connect the PLC to the robot controller so that the signal can be sent to each robot.

Follow the steps below when doing a production run in peripheral equipment mode.

- Set the remote/local setting to remote. On the teach pendant of each robot controller, press the [MENU] key, navigate to [System] → [Configuration] screen, select the "Remote/Local Setting" item, and set it to [Remote].
- Open the [Settings] → [Select Program] screen, and set [Automatic Operation Start Method] to "UOP". In addition, set [Program Selection Method] to "OTHER" (default setting), "PNS", or "RSR."



  - When [Program Selection Method] is "OTHER", press F3 [DETAIL] and set PK_MAIN1 to Data ID $UOP.PROGRAM_FOR_OTHER_MODE.
  - When [Program Selection Method] is "PNS", create a TP program named "PNS0001.TP". At this time, set the group mask to <1,*,*,*,*,*,*,*>. This means that this program is specifically for Group 1. Note that the PNS program for motion Group 1 needs to be named PNS0001.TP. For information on the program for Group 2, see "I.6.9.3. In the case of dual arm robots(P.443)".

- When [Program Selection Method] is “RSR”, create a TP program named “RSR0001.TP”. At this time, set the group mask to <1,*,*,*,*,*,*,*>. This means that this program is specifically for Group 1. Note that the RSR program for motion Group 1 needs be named RSR0001.TP. For information on the program for Group 2, see “I.6.9.3. In the case of dual arm robots(P.443)”.
- If a stack overflow alarm occurs during execution of the main TP program, increase the stack size of PNS0001.TP when the [Program Selection Method] is set to “PNS”, or RSR0001.TP when it is set to “RSR”.

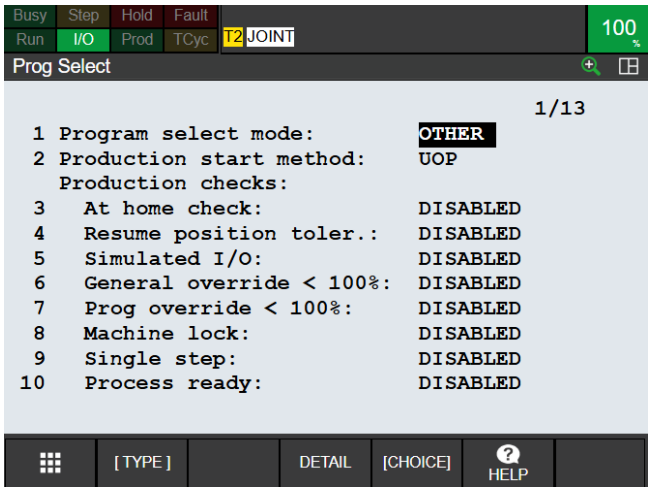
- For details on the [Select Program] screen, PNS, and RSR, refer to "OPERATOR’S MANUAL(Basic Operation)"(B-84464EN). Once these settings have been made in each robot controller, restart all the robot controllers.

6.9.3. In the case of dual arm robots

If there is a robot controller for a dual robot group, you can start iRPickTool using one of the following two methods.

- Using PNS:

  Press the [MENU] key, select [Settings] to display the [Select Program] screen, and set the [Program Selection Method] to “PNS”. If you have any questions regarding the PNS settings, refer to ““OPERATOR’S MANUAL(Basic Operation)"(B-84464EN)”.



  Create two programs: “PNS0001.TP” executed in Group 1 and “PNS0002.TP” executed in Group 2. In the header of the TP program, set the group mask to <1,*,*,*,*,*,*,*> for PNS0001.TP and to <*,1,*,*,*,*,*,*> for PNS0002.TP, respectively.

  PNS0001.TP must include the following line.

```
CALL PK_MAIN1
```

  PNS0002.TP must include the following line.

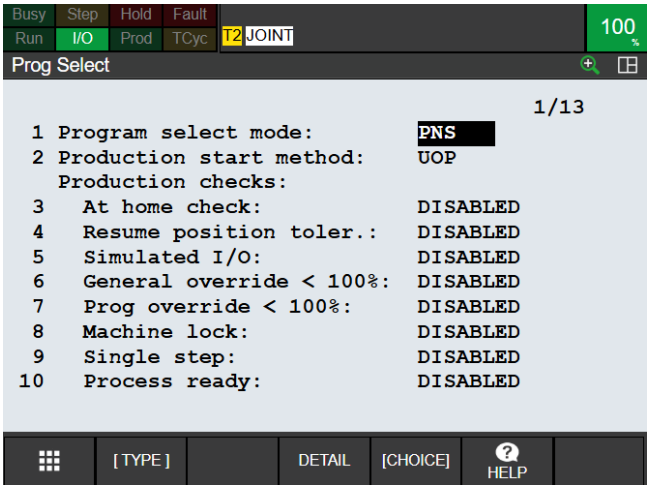
```
CALL PK_MAIN2
```

  If you do not want to run the iRPickTool standard TP program PK_MAIN2.TP on Group 2, do not create PNS0002.TP. Instead, create a PNS program that is neither PNS0001.TP nor PNS0002.TP. Use this method when you want to operate a servo axis in Group 2, for example, to drive the conveyor.

  This allows each robot motion group to be executed, paused, resumed, and aborted independently. Additionally, if a stack overflow alarm occurs during execution, increase the stack size of PNS0001.TP and PNS0002.TP.

- Using RSR:

  Press the [MENU] key, select [Settings] to display the [Select Program] screen, and set the [Program Selection Method] to “RSR”. If you have any questions regarding the RSR settings, refer to ““OPERATOR’S MANUAL(Basic Operation)"(B-84464EN)”.



Create two programs: “RSR0001.TP” executed in Group 1 and “RSR0002.TP” executed in Group 2. In the header of the TP program, set the group mask to <1,*,*,*,*,*,*,*> for RSR0001.TP and to <*,1,*,*,*,*,*,*> for RSR0002.TP, respectively.

RSR0001.TP must include the following line.

```
CALL PK_MAIN1
```

RSR0002.TP must include the following line.

```
CALL PK_MAIN2
```

If you do not want to run the iRPickTool standard TP program PK_MAIN2.TP on Group 2, do not create PNS0002.TP. Instead, create an RSR program that is neither RSR0001.TP nor RSR0002.TP. Use this method when you want to operate a servo axis in Group 2, for example, to drive the conveyor.

This allows each robot motion group to be executed, paused, resumed, and aborted independently. Additionally, if a stack overflow alarm occurs during execution, increase the stack size of RSR0001.TP and RSR0002.TP.

6.10. KAREL program

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.

6.10.1. Workcell-related programs

The two letters following [PK] are [WC] namely, the names of these programs begin with these four letters, [PKWC].

Note that the workcell-related KAREL program is the same for Plug & Play and iRPickTool Basic (J944). See “I.5.8. KAREL program(P.332)”.

6.10.2. Gripper-related programs

The two letters following [PK] are [GR] namely, the names of these programs begin with these four letters, [PKGR].

6.10.2.1. 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 set up in the Gripper and Zone menu in the tree view.

- 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 tree view. 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 the ID of a gripper.

Argument 2: Specify the operation, PICK or PLACE.

NOTE

When the argument guidance is supported, 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 register to return the status to. 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.

Usage:

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)
```

6.10.2.2. PKGRCLOSE.PC

This program performs the gripper close or pick operation for the specified zones of a gripper. This program uses the data you set up in the Gripper and Zone menus in the tree view.

Argument 1: Specify the 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 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)
```

6.10.2.3. PKGRGETDPDLY.PC

This program is used to get the drop or the place delay in seconds for a gripper that you specified as “Place Delay” in the Gripper setup menu in the tree view. This program uses the data you set up in the Gripper and Zone menus in the tree view.

Argument 1:	Specify the 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)
```

6.10.2.4. PKGRGETID.PC

This program gets the ID of the gripper used by a specified robot ID.

Argument 1:	Specify the ID of a robot.
Argument 2:	Specify the register to store the ID of the gripper set to the robot specified in argument 1.

Usage:

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)
```

6.10.2.5. PKGRGETPKDLY.PC

This program is used to get the pick delay in seconds for a gripper that you specified as “Pick Delay” in the Gripper setup menu in the tree view. This program uses the data you set up in the Gripper and Zone menus in the tree view.

Argument 1:	Specify the 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)
```

6.10.2.6. PKGRGETZNCNT.PC

This program is used to get the number of zones in a gripper. This program uses the data you set up in the Gripper and Zone menus in the tree view.

Argument 1:	Specify the 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)
```

6.10.2.7. PKGRGETZNUT.PC

This program is used to get the UTOOL of a Zone of a Gripper specified in the iRPickTool tree view into a Position Register. This program uses the data you set up in the Gripper and Zone menus in the tree view.

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 number of the Position Register to store the tool frame.

Example:

To get the UTOOL of 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)
```

6.10.2.8. PKGRINIT.PC

This program resets the gripper and zone data for all the grippers to factory defaults. You use this only in case you want to wipe out all the gripper data and start from scratch.

There is no argument.

Example:

```
CALL PKGRINIT
```

6.10.2.9. PKGROPEN.PC

This program performs the gripper open or place operation for the specified zones of a gripper. This program uses the data you set up in the Gripper and Zone menus in the tree view.

Argument 1:	Specify the 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)
```

6.10.3. Robot-related programs

The two letters following [PK] are [RB] namely, the names of these programs begin with these four letters, [PKRB].

Note that the following KAREL programs are the same for Plug & Play and iRPickTool Basic (J944). See "I.5.8. KAREL program(P.332)".

- PKRBGETID.PC
- PKRBGETSFLG.PC

6.10.3.1. 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 tree view data.

Argument 1: Specify the ID of a Robot.

Argument 2: Specify the operation, PICK or PLACE.

NOTE

When the argument guidance is supported, 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 number of the register to store the number of conveyors where the robot with the ID specified in argument 1 performs the operation specified in argument 2.

Usage:

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 Reg=7)
```

6.10.3.2. 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 tree view data.

Argument 1: Specify the ID of a Robot.

Argument 2: Specify the operation, PICK or PLACE.

NOTE

When the argument guidance is supported, 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 register number in which the fixed station count is to be returned.

Usage:

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)
```

6.10.3.3. PKRBGETPRGRT.PC

This program gets the type of pre-grouping. For details on pre-grouping, see “I.6.12. Setup of pre-grouping(P.487)”.

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 subsequent 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 the ID of an infeed conveyor.
Argument 2:	Specify the ID of a conveyor station on the infeed conveyor.
Argument 3:	Specify the ID of an outfeed conveyor.
Argument 4:	Specify the ID of a conveyor station on the outfeed conveyor.
Argument 5:	Specify the register number in which the type of pre-grouping is to be returned.

Usage:

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)
```

6.10.3.4. PKRBGETPKCV.PC

This program acquires the ID of an infeed conveyor.

Argument 1:	Specify the ID of a Robot.
Argument 2:	Specify the pick order number of a conveyor.
Argument 3:	Specify the register to return the conveyor ID to.

Usage:

The ID of the conveyor that the robot with ID 1 uses is acquired at the second pick operation. The conveyor ID is returned to register 7.

```
CALL PKRBGETPKCV(“Robot ID”=1,“ConvCount”=2,“Conv ID Reg”=7)
```

6.10.3.5. PKRBGETDPCV.PC

This program acquires the ID of an outfeed conveyor.

Argument 1:	Specify the ID of a Robot.
Argument 2:	Specify the placement order number of a conveyor.
Argument 3:	Specify the register to return the conveyor ID to.

Usage:

The ID of the conveyor that the robot with ID 1 uses is acquired at the first place operation. The conveyor ID is returned to register 7.

```
CALL PKRBGETDPCV(“Robot ID”=1,“ConvCount”=1,“Conv ID Reg”=7) ;
```

6.10.3.6. PKRBGETPKFS.PC

This program gets the ID of a supply fixed station.

Argument 1:	Specify the ID of a Robot.
Argument 2:	Specify the pick order number of a fixed station.
Argument 3:	Specify the register to return the fixed station ID to.

Usage:

The ID of the fixed station that the robot with ID 2 uses is acquired at the second pick operation. The conveyor ID is returned to register 7.

```
CALL PKRBGETPKFS(“Robot ID”=2,“ FStnCount ”=2,“FStn ID Reg”=7)
```

6.10.3.7. PKRBGETDPFS.PC

This program gets the ID of an outfeed fixed station.

Argument 1:	Specify the ID of a Robot.
Argument 2:	Specify the placement order number of a conveyor.
Argument 3:	Specify the register to return the fixed station ID to.

Usage:

The ID of the fixed station that the robot with ID 1 uses is acquired at the first place operation. The conveyor ID is returned to register 7.

```
CALL PKRBGETDPFS(“Robot ID”=1,“ FStnCount ”=2,“FStn ID Reg”=7)
```

6.10.4. Conveyor-related programs

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

Note that the conveyor-related KAREL program is the same for Plug & Play and iRPickTool Basic (J944). See “I.5.8. KAREL program(P.332)”.

6.10.5. Sensor-related programs

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

Note that the sensor-related KAREL program is the same for Plug & Play and iRPickTool Basic (J944). See “I.5.8. KAREL program(P.332)”.

6.10.6. Conveyor station-related programs

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

Note that the following KAREL programs are the same for Plug & Play and iRPickTool Basic (J944). See "I.5.8. KAREL program(P.332)".

- PKCSGETID.PC
- PKCSCLRQUE.PC
- PKCSGETQUE.PC
- PKCSGETQCELL.PC
- PKCSACKQUE.PC
- PKCSPUTQUE.PC
- PKCSGETTIME.PC
- PKCSGETDIST.PC
- PKCSGETPFRT.PC
- PKCSGETNPRT.PC
- PKCSSETDSLIN.PC
- PKCSGETENC.PC
- PKCSGETTRID.PC
- PKCSCLRACK.PC
- PKCSPUTQUE2.PC

6.10.6.1. PKCSCALWPOS.PC

This program uses part information acquired with PKCSGETQUE to calculate a wait position for the robot at the upstream boundary of the tracking area in the conveyor station to wait for the relevant part. This program outputs the decision regarding whether the robot moves to the wait position, along with the wait position. 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.

Argument 1:	Specify the ID of the conveyor station where you want the wait position to be calculated. The ID can be acquired using PKCSGETID. For details, refer to "I.5.8.5.1. PKCSGETID.PC(P.350)".
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 status of PKCSGETQUE.
Argument 5:	Specify the 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 the distance from the upstream boundary of the tracking area 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. If a part reaches the tracking area of the conveyor station by the time the robot arrives at the conveyor station from its current position, the robot does not need to move to the wait position. TThis setting value is intended to prevent unnecessary movements to the wait position. You may need to adjust this number depending upon the conveyor speed.

Usage:

In the conveyor station with ID 1, the reference position is taught to PR[56], PKCSGETQUE acquires part information to VR[1] and the status of PKCSGETQUE to R[10]. It is estimated that the conveyor moves by 150 mm while the robot moves from the current position to the wait position. The wait position is calculated with this condition. The decision on whether the robot moves to the wait position is returned to R[11]. The calculated wait position is returned to PR[40].

```
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)
```

6.10.6.2. PKCSGETAPPR.PC

This program gets and sets the approach offset of a conveyor station's reference position you specified in the OP_CS_cstnName menu in the iRPickTool tree view. It also returns the Position Register number used for the approach position.

Argument 1:	Specify the ID of a conveyor station.
Argument 2:	Specify the register number in which to return the Position Register number used to store the approach offset. (The number of the Position Register to store an approach offset is specified in OP_CS_cstnName in the iRPickTool tree view.)

Usage:

The approach offset set to the conveyor station operation corresponding to the conveyor station with ID 1 is acquired to the Position Register. It also returns the Position Register number used for the approach position to R[95].

```
CALL PKCSGETAPPR(CStn ID=1,ApproachPR Reg=95)
```

6.10.6.3. PKCSGETOPPR.PC

This program gets the "Reference Pos PR" of a conveyor station you specified in the OP_CS_cstnName menu in the iRPickTool tree view.

Argument 1:	Specify the ID of a conveyor station.
Argument 2:	Specify the register number in which to return the Position Register number of the Reference Position.

Usage:

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)
```

6.10.6.4. PKCSGETPMUL.PC

This program gets the "Pick or Drop Multiple" setting of a conveyor station (TRUE = 1, FALSE = 0) you specified in the OP_CS_cstnName menu in the iRPickTool tree view.

Argument 1:	Specify the ID of a conveyor station.
Argument 2:	Specify the register number in which to return the "Pick or Drop Multiple" enable or disable property. 1: Enable 0: Disable

Usage:

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)
```

6.10.6.5. PKCSGETPPCHK.PC

This program gets the “Part Presence Check” setting of a conveyor station (TRUE = 1, FALSE = 0) you specified in the OP_CS_cstnName menu in the iRPickTool tree view.

Argument 1:	Specify the ID of a conveyor station.
Argument 2:	Specify the register number in which to return the “Part Presence Check” enable or disable property. 1: Enable 0: Disable

Usage:

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)
```

6.10.6.6. PKCSGETVR.PC

This program gets the “Part Data VR” of a conveyor station you specified in the OP_CS_cstnName menu in the iRPickTool tree view.

Argument 1:	Specify the ID of a conveyor station.
Argument 2:	Specify the number of the register to store the value in [Vision Reg].

Usage:

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)
```

6.10.6.7. PKCSGETWPOS.PC

This program gets the “Wait Position” setting of a conveyor station (TRUE = 1, FALSE = 0) you specified in the OP_CS_cstnName menu in the iRPickTool tree view.

Argument 1:	Specify the ID of a conveyor station.
Argument 2:	Specify the register number in which to return the “Wait Position” enable or disable property. 1: Enable 0: Disable

Usage:

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)
```

6.10.6.8. PKCSGETCLRQ.PC

This program acquires the “Clear queue at Prod start” setting (TRUE = 1, FALSE = 0) for the conveyor station OP_CS_cstn displayed in the iRPickTool tree view.

Argument 1:	Specify the ID of a conveyor station.

Argument 2: Specify the register to which the property that indicates whether “Clear queue at Prod start” is enabled or disabled is returned.

Usage:

The “Clear queue at Prod start” property is acquired from the conveyor station with ID 1 and stored in register 8.

```
CALL PKCSGETCLRQ(CStn ID=1,CLRQ Reg=8)
```

6.10.7. Fixed station-related programs

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

Note that the following KAREL programs are the same for Plug & Play and iRPickTool Basic (J944). See “I.5.8. KAREL program(P.332)”.

- PKFSGETID.PC
- PKFSPUTQUE.PC
- PKFSCLRQUE.PC
- PKFSGETQUE.PC
- PKFSACKQUE.PC
- PKFSPUTBUF.PC
- PKFSCLRBUF.PC
- PKFSGETBUF.PC
- PKFSACKBUF.PC
- PKFSGETTRID.PC
- PKFSGETUF.PC

6.10.7.1. PKFSGETAPPR.PC

This program gets and sets the approach offset of a fixed station’s reference position you specified in the OP_FS_fstnName menu in the iRPickTool tree view. It also returns the Position Register number used for the approach position.

Argument 1: Specify the ID of a fixed station.

Argument 2: Specify the register number in which to return the Position Register number used to store the approach offset. (The number of the Position Register to store an approach offset is specified in OP_CS_cstnName in the iRPickTool tree view.)

Usage:

The approach offset set to the fixed station operation corresponding to the fixed station with ID 1 is acquired to the Position Register. It also returns the Position Register number used for the approach position to R[95].

```
CALL PKFSGETAPPR(FStn ID=1,ApproachPR Reg=95)
```

6.10.7.2. PKFSGETIXDLY.PC

This program gets the “Index delay” you specified in the OP_FS_fstnName menu in the iRPickTool tree view. The index delay is only used for simulated indexing of a fixed station. The units are milliseconds.

Argument 1: Specify the ID of a fixed station.

Argument 2: Specify the register number in which to return the “Index delay”.

Usage:

To get the index delay of a fixed station whose ID is 1 in Numeric Register 6, use the following instruction:

```
CALL PKFSGETIXDLY(FStn ID=1,IndexDelay Reg=6)
```

6.10.7.3. PKFSGETIXODO.PC

This program gets the “Index Out DO” or the Digital Output port number used for indexing out a fixed station you specified in the OP_FS_fstnName menu in the iRPickTool tree view.

Argument 1: Specify the ID of a fixed station.

Argument 2: Specify the register number in which to return the “Index Out DO”.

Usage:

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)
```

6.10.7.4. PKFSGETIXSIM.PC

This program gets the “Simulate Indexing” setting of a conveyor station (TRUE = 1, FALSE = 0) you specified in the OP_FS_fstnName menu in the iRPickTool tree view.

Argument 1: Specify the ID of a fixed station.

Argument 2: Specify the register number in which to return the “Simulate Indexing” enable or disable property.
1: Enable
0: Disable

Usage:

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)
```

6.10.7.5. PKFSGETOPPR.PC

This program gets the “Reference Pos PR” of a fixed station you specified in the OP_FS_fstnName menu in the iRPickTool tree view.

Argument 1: Specify the ID of a fixed station.

Argument 2: Specify the register number in which to return the Position Register number of the Reference Position.

Usage:

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)
```

6.10.7.6. PKFSGETPMUL.PC

This program gets the “Pick or Drop Multiple” setting of a fixed station (TRUE = 1, FALSE = 0) you specified in the OP_FS_fstnName menu in the iRPickTool tree view.

Argument 1:	Specify the ID of a fixed station.
Argument 2:	Specify the register number in which to return the “Pick or Drop Multiple” enable or disable property. 1: Enable 0: Disable

Usage:

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)
```

6.10.7.7. PKFSGETPPCHK.PC

This program gets the “Part Presence Check” setting of a fixed station (TRUE = 1, FALSE = 0) you specified in the OP_FS_fstnName menu in the iRPickTool tree view.

Argument 1:	Specify the ID of a fixed station.
Argument 2:	Specify the register number in which to return the “Part Presence Check” enable or disable property. 1: Enable 0: Disable

Usage:

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)
```

6.10.7.8. PKFSGETRDY.PC

This program gets the internal status of “fixed station ready” (TRUE = 0, FALSE = 1). When you run the robot with a fixed station, 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 the Station Ready DI to see when it goes from low to 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 the ID of a fixed station.
Argument 2:	Specify the register number in which to return the internal “Ready” status of a fixed station. 1: Ready 0: NotReady

Usage:

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)
```

6.10.7.9. PKFSGETRDYDI.PC

This program gets the “Ready DI” or the Digital Input port number used for checking if the fixed station is present (or ready to pick or place) that you specified in the OP_FS_fstnName menu in the iRPickTool tree view.

Argument 1:	Specify the ID of a fixed station.
Argument 2:	Specify the register number in which to return the “Ready DI” port number.

Usage:

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)
```

6.10.7.10. PKFSGETVR.PC

This program gets the “Part Data VR” of a fixed station you specified in the OP_CS_fstnName menu in the iRPickTool tree view.

Argument 1:	Specify the ID of a fixed station.
Argument 2:	Specify the number of the register to store the value in [Vision Reg].

Usage:

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)
```

6.10.7.11. PKFSMONIDX.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 Ready DI is high and Index DO is low, the program sets this fixed station to ready. (A robot will not be allowed inside a fixed station unless this status is TRUE).

Argument 1:	Specify the ID of a fixed station.
Argument 2:	Specify the register number in which to return the status. Values other than 0 indicate that the monitoring of the Station Ready DI has failed.

Usage:

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.

```
CALL PKFSMONIDX(FStn ID=1,Stat Reg=7)
```

6.10.7.12. PKFSSETRDY.PC

This program sets the internal status of “fixed station ready” (“Ready” or “NotReady”). Basically, use this program to prohibit the operation of a fixed station for safety. For what process internally checks availability, see “I.6.10.7.8. PKFSGETRDY.PC(P.456)”.

Argument 1:	Specify the ID of a fixed station.
Argument 2:	Specify “Ready” or “NotReady” to set the internal status for “fixed station ready”.

⚠CAUTION

When the argument guidance is supported, select one of the following options as the operation type.

1. Ready
2. Not Ready

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.

Usage:

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)
```

6.10.7.13. PKFSGETCLRQ.PC

This program acquires the “Clear queue at Prod start” setting (TRUE = 1, FALSE = 0) for the fixed station OP_FS_fstn displayed in the iRPickTool tree view.

Argument 1:	Specify the ID of a fixed station.
Argument 2:	Specify the register to which the property that indicates whether “Clear queue at Prod start” is enabled or disabled is returned.

Usage:

The “Clear queue at Prod start” property is acquired from the fixed station with ID 1 and stored in register 8.

```
CALL PKFSGETCLRQ(FStn ID=1,CLRQ Reg=8)
```

6.10.7.14. PKFSSETIDXDO.PC

This program sets the index output DO for the fixed station.

Argument 1:	Specify the ID of a fixed station.
Argument 2:	Specify the DO value of index output.

Usage:

Set DO[8] as the output DO index of fixed station 1.

```
CALL PKFSSETIDXDO(FStn ID=1,IDX DO=8)
```

6.10.8. Tray-related programs

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

Note that the tray-related KAREL program is the same for Plug & Play and iRPickTool Basic (J944). See “I.5.8. KAREL program(P.332)”.

6.11. Standard TP programs

This section describes the standard TP programs provided with the J945: iRPickTool option.

Users can customize these programs to meet the detailed requirements of the application. These programs improve user productivity because less code needs to be written to complete the pick application. As an SI, you can more easily and quickly develop solutions for a variety of customers to meet their requirements by using these programs that we have thoroughly tested and validated.

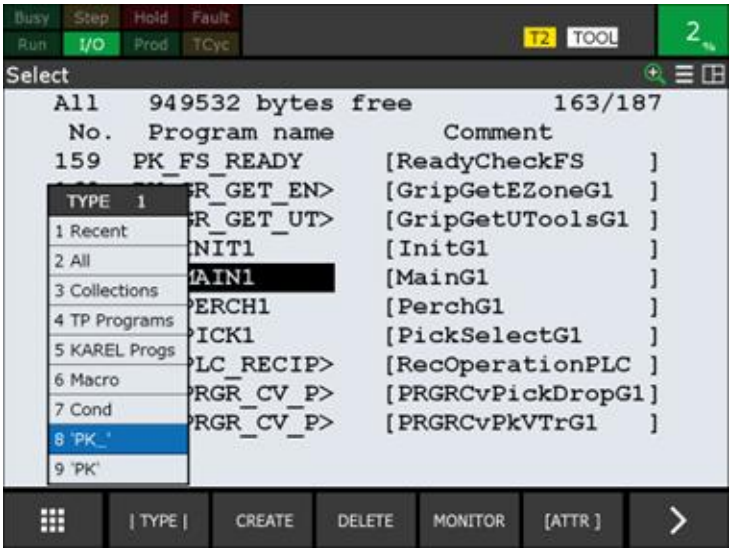
These standard TP programs have the following functions. See “I.6.5.1. Robot behavior under the standard TP programs(P.435)” for the actual robot behavior.

- All the FANUC robots. FANUC robots include 3-, 4-, 5-, or 6-axis robots such as delta robots, SCARA robots, and the LR Mate series.
- The following applications are supported:
  - Applications where the robots pick from a conveyor and place on a conveyor.
  - Applications where the robots pick from a conveyor and place on a fixed station or vice-versa.
  - Applications where the robots pick from a fixed station and place 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 (up to 20).
- Many other functions like automatic move to wait position when there are no parts, part presence checks for the gripper, etc.

6.11.1. 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 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 Standard 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 names end in 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 indicates that the robot in Group 1 picks a part for the first time.

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.

6.11.2. Standard TP program details

6.11.2.1. List of all the Standard TP programs

The following is a 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]

No.	Program name	Comment
18	PK_MAIN1	[MainG1]
19	PK_PERCH1	[PerchG1]
20	PK_PICK1	[PickSelectG1]

6.11.2.2. 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.

The initialization consists of:

1. **Initialize the system, including registers.**
   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.
2. **Start the sensor task to be ready to detect parts from the camera, DI or HDI sensors and put them into the part queue.**
3. **Move the robot to the perch position**
4. **When pre-grouping is used, the main program for pre-grouping PK_PRGR_MAIN1.TP is called.**
5. **When pre-grouping is not used, do the pick and place indefinite loop.**
   Always check if cycle stop is invoked at the end of the pick operation and the end of the place operation.
6. **If cycle stop is invoked, process the cycle stop function.**
7. **Upon exit from the pick and place loop (for example, via cycle stop) or exit from the main program for pre-grouping, stop the sensor task.**

PK_MAIN1.TP

```
 1: ! Main Prog for Grp:1
 2:
 3: LBL[10]
 4: --
 5: CALL PK_INIT1
 6: --
 7: CALL PKSNSTART
 8: --
 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: 00
18: LBL[100:pk n pl]
19: --
20: CALL PK_PICK1
21: IF R[101:Cycle Stop G1]>0,JMP LBL[10000]
22: --
23: CALL PK_DROP1
24: LBL[200]
25: IF R[101:Cycle Stop G1]>0,JMP LBL[10000]
26: JMP LBL[100]
```

```
27: 00:00
28: LBL[10000: End]
29: --
30: CALL PKSNSTOP
[End]
```

6.11.2.3. PK_INIT1.TP

PK_INIT1.TP is the initialization program for the group 1 main task which is PK_MAIN1.TP.

The initialization consists of:

1. **Checking if iRPickTool data is synchronized across all robot controllers.**
2. **Getting the active recipe id.**
3. **Determining the Robot ID for the group 1 robot.**
4. **Determining the Robot's gripper ID.**
5. **Getting the zone data of the gripper and assigning to UTOOLs.**
6. **Setting the P[1] position of the PK_PERCH1.TP program whenever a recipe is restored or switched to using the PKCOPYPR2POS KAREL macro.**
7. **Determining if the pick station is a conveyor station or a fixed station.**
8. **Determining if the place station is a conveyor station or a fixed station.**
9. **Getting all the data from the tree view for the robot and the required conveyor and fixed stations.**
10. **Getting the buffer id for the group 1 robot.**

    If buffer id > 0, that is, the Plug & Play buffer function is enabled, then PK_BF_GET_DATA1.TP is called, which gets buffer-related setting data that is in registers or Position Registers.

    The acquired setting data is referenced in programs to pick parts from the buffer or place parts in the buffer. The buffer function is included in R827: iRPickTool Production Control.
11. **Getting type for pre-grouping.**

    When pre-grouping is not used, the type is 0.
12. **Initializing or resetting 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: CALL PKWCGETACTRC("RecipeID Reg"=99)
 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 Utool 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: --
```

```
20: --
21: IF R[105:Recipe loaded G1]<>1,JMP LBL[25]
22: --
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]
```

6.11.2.4. 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: --
 2:
 3: UTOOL_NUM=0
 4: UFRAME_NUM=0
 5: !*********************************
 6: --
 7: --
 8: --
 9: --
10: --
11: --
12: --
13: !*********************************
14: PAYLOAD[1]
15: J P[1] 25% FINE
16: --
17: PR[54:Perch Pos]=P[1]
[End]
```

6.11.2.5. 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.

You can instead call PK_CV_DROP14.TP from PK_CV_DROP12.TP based on the pick order if there are multiple infeed conveyors. Up to four infeed conveyors are directly supported by PK_PICK11.TP. As with the conveyor, PK_FS_PICK11.TP is called for the pick motion for the first fixed station. Up to four pick motions on the fixed station are supported.

The buffer function is included in R827: iRPickTool Production Control. For details, see “I.6.13. Plug & Play buffer function(P.497)”.

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 “I.6.12. Setup of pre-grouping(P.487)”.

PK_PICK1.TP

```
  1: !G1 pick program
  2: !Determine tracking or non-tracking and call the corresponding program
  3:
  4: R[107:Counter G1]=1
  5: IF R[109:Pk1x CvId]=0,JMP LBL[1000]
  6: IF (R[199:Buf Id G1]<>0),JMP LBL[50]
  7: IF R[230:PRGR Robot Type]=3,JMP LBL[60]
  8: !Motion when picking from the conveyor
  9: !Check the conveyor switching flag
 10: SELECT R[216:Pick Stn]=1,CALL PK_CV_PICK11
 11: =2,CALL PK_CV_PICK12
 12: =3,CALL PK_CV_PICK13
 13: ELSE,CALL PK_CV_PICK14
 14: R[216:Pick Stn]=R[216:Pick Stn]+1
 15: CALL PKRBGETCVCNT(Robot ID=R[103:Robot Id G1],Pick=0,ConvCount Reg=95)
 16: IF (R[216:Pick Stn]>R[95:TemporaryUse1]),R[216:Pick Stn]=(1)
 17: JMP LBL[10000]
 18: !Buffering
 19: LBL[50: Buf loop]
 20: CALL PK_BUF_PICK1
 21: JMP LBL[10000]
 22: LBL[60:Virtual Tray]
 23: !Pick the virtual tray
 24: CALL PK_PRGR_CV_PICKVTRAY1
 25: JMP LBL[10000]
 26:
 27: LBL[1000: FStn]
 28: !Non-tracking pick motion on the fixed station
 29: !Check the fixed station switching flag
 30: SELECT R[216:Pick Stn]=1,CALL PK_FS_PICK11
 31: =2,CALL PK_FS_PICK12
 32: =3,CALL PK_FS_PICK13
 33: ELSE,CALL PK_FS_PICK14
 34: R[216:Pick Stn]=R[216:Pick Stn]+1
 35: CALL PKRBGETFSCNT(Robot ID=R[103:Robot Id G1],Pick=0,FStnCount Reg=95)
 36: IF (R[216:Pick Stn]>R[95:TemporaryUse1]),R[216:Pick Stn]=(1)
 37: LBL[10000: End]
[End]
```

6.11.2.6. 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.

You can call PK_CV_DROP14.TP from PK_CV_DROP12.TP based on the placement order if there are multiple outfeed conveyors. Up to four outfeed conveyors are directly supported by PK_DROP11.TP. As with the conveyor, PK_FS_DROP11.TP is called for placement motion for the first fixed station. Up to four placement motions on the fixed station are supported.

The buffer function is included in R827: iRPickTool Production Control. For details, see "I.6.13. Plug & Play buffer function(P.497)".

PK_DROP1.TP

```
  1: !G1 placement program
  2: !Determine tracking or non-tracking and call the corresponding program
  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: !Motion when placing on the conveyor
  8: !Check the conveyor switching flag
  9: SELECT R[217:Drop Stn]=1,CALL PK_CV_DROP11
 10: =2,CALL PK_CV_DROP12
 11: =3,CALL PK_CV_DROP13
 12: ELSE,CALL PK_CV_DROP14
 13: R[217:Drop Stn]=R[217:Drop Stn]+1
 14: CALL PKRBGETCVCNT(Robot ID=R[103:Robot Id G1],Place=1,ConvCount Reg=95)
 15: IF (R[217:Drop Stn]>R[95:TemporaryUse1]),R[217:Drop Stn]=(1)
 16: JMP LBL[10000]
 17: !Buffering
 18: LBL[50: Buffer drop]
 19: CALL PK_BUF_DROP1
 20: JMP LBL[10000]
 21:
 22: LBL[1000: FixedPos]
 23: !Non-tracking placement motion on the fixed station
 24: !Check the fixed station switching flag
 25: SELECT R[217:Drop Stn]=1,CALL PK_FS_DROP11
 26: =2,CALL PK_FS_DROP12
 27: =3,CALL PK_FS_DROP13
 28: ELSE,CALL PK_FS_DROP14
 29: R[217:Drop Stn]=R[217:Drop Stn]+1
 30: CALL PKRBGETFSCNT(Robot ID=R[103:Robot Id G1],Place=1,FStnCount Reg=95)
 31: IF (R[217:Drop Stn]>R[95:TemporaryUse1]),R[217:Drop Stn]=(1)
 32: LBL[10000: End]
[End]
```

6.11.2.7. PK_GR_GET_UTOOLS1.TP

This program sets the UTOOL for each zone of the gripper used by the group 1 robot.

1 Uses the KAREL macro PKGRGETZNCNT to get the zone count of the specified gripper

2 **Uses the KAREL macro PKGRGETZNUT to get the UTOOL value specified in the Zone menu of the Gripper in the iRPickTool tree view.**

3 **Sets the retrieved UTOOL of the zone to the system's UTOOL.**

This program is called by two arguments.

Argument 1:	Robot ID
Argument 2:	Gripper ID

PK_GR_GET_UTOOLS1.TP

```
 1: --
 2: --
 3: --
 4: --
 5: --
 6: --
 7:
 8: --
 9: CALL PKGRGETZNCNT(Gripper ID=AR[2],ZoneCount Reg=96)
10: FOR R[95:TemporaryUse1]=1 TO R[96:TemporaryUse2]
11: --
12: CALL PKGRGETZNUT(Robot ID=AR[1],Gripper ID=AR[2],Zone Index=R[95:TemporaryUse1],
    ZoneUTool PR=51)
13: --
14: UTOOL[R[95]]=PR[51:TemporaryUse1]
15: ENDFOR
16: ! PickQty = PlaceQty = ZoneCnt
17: R[111:Pk1x Qty]=R[96:TemporaryUse2]
18: R[131:Dp1x Qty]=R[96:TemporaryUse2]
[End]
```

6.11.2.8. PK_CV_GET_PICK_DATA11.TP

This program is used to get the data setup in the iRPickTool tree view for a pick operation at a conveyor station of a conveyor.

1 **Gets the conveyor ID (initialized when the KAREL macro PKRBGETCVCNT is executed) for the first pick operation using the KAREL macro PKRBGETPKCV.**

2 **Get the name of the first conveyor using the KAREL macro PKCVGETNAME.PC.**

3 **Gets the ID of the conveyor station on the conveyor using the KAREL macro PKCSGETID.PC.**

4 **Gets the "Pick or drop multiple" property for the pick operation using the KAREL macro PKCSGETPMUL.PC.**

5 **Gets the approach offset of the conveyor station from the tree view data and puts it in the approach offset Position Register using the KAREL macro PKCSGETAPPR.PC.**

6 **Gets the FLG used for checking the "Skip Outbound" condition in the tracking pick program using the KAREL macro PKRBGETSFLG.PC.**

7 **Gets the ID of the tray at the conveyor using the KAREL macro PKCSGETTRID.PC**

8 **Clears any pending ACK to avoid the "GetQueue before Ackque" error using the KAREL macro PKCSCLRACK.PC.**

9 **Gets a clear queue setting using the KAREL macro PKCSGETCLRQ and clears the queue using the KAREL macro PKCSCLRQUE.**

PK_CV_GET_PICK_DATA11.TP

```
  1: --
  2: --
  3: --
  4: --
  5:
  6: --
  7: CALL PKRBGETPKCV(Robot ID=R[103:Robot Id G1],ConvCount=1,Conv ID Reg=109)
  8: --
  9: CALL PKCVGETNAME(Conv ID=R[109:Pk1x CvId],ConvName SR=4)
 10: CALL PKCSGETID(SR[4],CStn ID Reg=110)
 11: CALL PKCSGETPMUL(CStn ID=R[110:Pk1x Stn],PikMultiple Reg=112)
 12: CALL PKCSGETAPPR(CStn ID=R[110:Pk1x Stn],ApproachPR Reg=95)
 13: CALL PKRBGETSFLG(Robot ID=R[103:Robot Id G1],SkipFlag Reg=102)
 14: !Get tray ID of conveyor station
 15: CALL PKCSGETTRID(CStn ID=R[110:Pk1x Stn],Tray ID Reg=121)
 16: --
 17: CALL PKCSCLRACK(CStn ID=R[110:Pk1x Stn])
 18: !Clear queue at the Group 1 conveyor station
 19: CALL PKCSGETCLRQ(CStn ID=R[110:Pk1x Stn],CLRQ Reg=95)
 20: IF R[95:TemporaryUse1]=0,JMP LBL[5000]
 21: CALL PKCSCLRQUE(CStn ID=R[110:Pk1x Stn])
 22:
 23: LBL[5000: Eff Tools]
 24: !Get PosCheck flag
 25: !PKSCSGETPOSCK(CStn ID,
 26: ! operation,PosCheck Reg)
 27: CALL PKCSGETPOSCK(CStn ID=R[110:Pk1x Stn],Pick=0,PosCheck Reg=115)
 28: LBL[10000:End]
[End]
```

⚠CAUTION

Although this program overwrites R[95:TemporaryUse1] multiple times, some values may be overwritten with other values without being used after being stored, like the value stored in R[95] on line 12. Such values are not referenced in standard TP programs and do not cause problems.

However, when changing a standard TP program before use, take actions such as saving these values in registers other than R[95] that are not overwritten, as needed.

6.11.2.9. PK_CV_GET_DROP_DATA11.TP

This program is used to get the data setup in the iRPickTool tree view for a drop or place operation at a conveyor station of a conveyor.

1. **Gets the conveyor ID (initialized when the KAREL macro PKRBGETCVCNT is executed) for the first drop operation using the KAREL macro PKRBGETDPCV.**
2. **Get the name of the first conveyor using the KAREL macro PKCVGETNAME.PC.**
3. **Gets the ID of the conveyor station on the conveyor using the KAREL macro PKCSGETID.PC.**
4. **Gets the "Pick or drop multiple" property for the pick operation using the KAREL macro PKCSGETPMUL.PC.**
5. **Gets the approach offset of the conveyor station from the tree view data and puts it in the approach offset Position Register using the KAREL macro PKCSGETAPPR.PC.**
6. **Gets the FLG used for checking the "Skip Outbound" condition in the tracking pick program using the KAREL macro PKRBGETSFLG.PC.**
7. **Gets the ID of the tray at the conveyor using the KAREL macro PKCSGETTRID.PC**

8 **Clears any pending ACK to avoid the "GetQueue before Ackque" error using the KAREL macro PKCSCLRACK.PC.**

9 **Gets a clear queue setting using the KAREL macro PKCSGETCLRQ and clears the queue using the KAREL macro PKCSGETCLRQ.**

PK_CV_GET_DROP_DATA11.TP

```
  1: --
  2: --
  3: --
  4: --
  5:
  6: --
  7: CALL PKRBGETDPCV(Robot ID=R[103:Robot Id G1],ConvCount=1,Conv ID Reg=129)
  8: --
  9: CALL PKCVGETNAME(Conv ID=R[129:Dp1x CvId],ConvName SR=5)
 10: CALL PKCSGETID(SR[5],CStn ID Reg=130)
 11: CALL PKCSGETPMUL(CStn ID=R[130:Dp1x Stn],PikMultiple Reg=132)
 12: CALL PKCSGETAPPR(CStn ID=R[130:Dp1x Stn],ApproachPR Reg=95)
 13: CALL PKRBGETSFLG(Robot ID=R[103:Robot Id G1],SkipFlag Reg=102)
 14: !Get tray ID of conveyor station
 15: CALL PKCSGETTRID(CStn ID=R[130:Dp1x Stn],Tray ID Reg=141)
 16: --
 17: CALL PKCSCLRACK(CStn ID=R[130:Dp1x Stn])
 18: !Clear queue at the Group 1 conveyor station
 19: CALL PKCSGETCLRQ(CStn ID=R[130:Dp1x Stn],CLRQ Reg=95)
 20: IF R[95:TemporaryUse1]=0,JMP LBL[5000]
 21: CALL PKCSCLRQUE(CStn ID=R[130:Dp1x Stn])
 22:
 23: LBL[5000: Eff Tools]
 24: !Get PosCheck flag
 25: !PKSCSGETPOSCK(CStn ID,
 26: ! operation,PosCheck Reg)
 27: CALL PKCSGETPOSCK(CStn ID=R[130:Dp1x Stn],Place=1,PosCheck Reg=135)
 28: LBL[10000:End]
[End]
```

⚠CAUTION

Although this program overwrites R[95:TemporaryUse1] multiple times, some values may be overwritten with other values without being used after being stored, like the value stored in R[95] on line 12. Such values are not referenced in standard TP programs and do not cause problems.

However, when changing a standard TP program before use, take actions such as saving these values in registers other than R[95] that are not overwritten, as needed.

6.11.2.10. PK_FS_GET_PICK_DATA11.TP

This program is used to get the data setup in the iRPickTool tree view for a pick operation at a fixed station.

1 **Gets the Fixed Station ID (initialized when the KAREL macro PKRBGETFSCNT is executed) for the first pick operation using the KAREL macro PKRBGETPKFS.**

2 **Gets the "Pick or drop multiple" property for the pick operation using the KAREL macro PKFSGETPMUL.PC.**

3 **Gets the approach offset of the fixed station from the tree view data and puts it in the approach offset Position Register using PKFSGETAPPR.PC.**

4 **Gets the Station Ready DI of the fixed station (for later station_ready? Check) using the KAREL macro PKFSGETRDYDI.PC.**

5 **Gets the Index Out DO digital output of the fixed station (for indexing the fixed station) using the KAREL macro PKFSGETIXODO.PC.**

6 **Sets the initial value of the internal station ready flag to FALSE. If the internal "ready" status of a fixed station is found during the station operation check, the station status is set to TRUE using the KAREL macro PKFSSETRDY.**

7 **The robot checks if [Simulate Indexing] set to the fixed station operation is enabled using the KAREL program PKFSGETIXSIM.**

8 **Gets the ID of the tray at the fixed station using the KAREL macro PKFSGETTRID.PC**

9 **Clears any pending ACK to avoid the "GetQueue before Ackque" error using the KAREL macro PKFSCLRACK.PC.**

10 **Initializes the tray using the KAREL macros PKFSCLRAUE and PKFSPUTQUE (the return values of the KAREL macro call PKFSGETCLRQ). Change this initialization process as appropriate based on the DI signal or other means.**

PK_FS_GET_PICK_DATA11.TP

```
  1: --
  2: --
  3: --
  4:
  5: --
  6: R[109:Pk1x CvId]=0
  7: CALL PKRBGETPKFS(Robot ID=R[103:Robot Id G1],FStnCount=1,
       FStn ID Reg=110)
 8: CALL PKFSGETPMUL(FStn ID=R[110:Pk1x StnI],PikMultiple Reg=112)
 9: CALL PKFSGETAPPR(FStn ID=R[110:Pk1x StnI],ApproachPR Reg=95)
 10: CALL PKFSGETRDYDI(FStn ID=R[110:Pk1x StnI],ReadyDI Reg=118)
 11: CALL PKFSGETIXODO(FStn ID=R[110:Pk1x StnI],IndexOutDO Reg=119)
 12: CALL PKFSSETRDY(FStn ID=R[110:Pk1x StnI],Ready=1)
 13: CALL PKFSGETIXSIM(FStn ID=R[110:Pk1x StnI],IndexSim Reg=95)
 14: CALL PKFSGETTRID(FStn ID=R[110:Pk1x StnI],Tray ID Reg=121)
 15: CALL PKFSCLRACK(FStn ID=R[110:Pk1x StnI])
 16: IF R[121:Pk1x TrayId]=0,JMP LBL[10000]
 17: CALL PKFSGETCLRQ(FStn ID=R[110:Pk1x StnI],CLRQ Reg=95)
 18: IF R[95:TemporaryUse1]=0,JMP LBL[1000]
 19: CALL PKFSCLRQUE(FStn ID=R[110:Pk1x StnI])
 20: CALL PKFSPUTQUE(FStn ID=R[110:Pk1x StnI])
 21: --
 22: R[122:Pk1x TrayCtr]=1
 23: LBL[1000]
 24:
 25: LBL[10000: End]
[End]
```

⚠CAUTION

Although this program overwrites R[95:TemporaryUse1] multiple times, some values may be overwritten with other values without being used after being stored, like the value stored in R[95] on line 9. Such values are not referenced in standard TP programs and do not cause problems.

However, when changing a standard TP program before use, take actions such as saving these values in registers other than R[95] that are not overwritten, as needed.

6.11.2.11. PK_FS_GET_DROP_DATA11.TP

This program is used to get the data setup in the iRPickTool tree view for a drop or place operation at a fixed station.

1. **Gets the fixed station ID (initialized when the KAREL macro PKRBGETFSCNT is executed) for the first drop operation using the KAREL macro PKRBGETDPFS.**
2. **Gets the "Pick or drop multiple" property for the pick operation using the KAREL macro PKFSGETPMUL.PC.**
3. **Gets the approach offset of the fixed station from the tree view data and puts it in the approach offset Position Register using PKFSGETAPPR.PC.**
4. **Gets the Station Ready DI of the fixed station (for later station_ready? Check) using the KAREL macro PKFSGETRDYDI.PC.**
5. **Gets the Index Out DO digital output of the fixed station (for indexing the fixed station) using the KAREL macro PKFSGETIXODO.PC.**
6. **Sets the initial value of the internal station ready flag to FALSE. If the internal "ready" status of a fixed station is found during the station operation check, the station status is set to TRUE using the KAREL macro PKFSSETRDY.**
7. **The robot checks if [Simulate Indexing] set to the fixed station operation is enabled using the KAREL program PKFSGETIXSIM.**
8. **Gets the ID of the tray at the fixed station using the KAREL macro PKFSGETTRID.PC**
9. **Clears any pending ACK to avoid the "GetQueue before Ackque" error using the KAREL macro PKFSCLRACK.PC.**
10. **Initializes the tray using the KAREL macros PKFSCLRAUE and PKFSPUTQUE (the return values of the KAREL macro call PKFSGETCLRQ).**

PK_FS_GET_DROP_DATA11.TP

```
  1: --
  2: --
  3: --
  4:
  5: --
  6: R[129:Dp1x CvId]=0
  7: CALL PKRBGETDPFS(Robot ID=R[103:Robot Id G1],FStnCount=1,
       FStn ID Reg=130)
  8: CALL PKFSGETPMUL(FStn ID=R[130:Dp1x StnI],PikMultiple Reg=132)
  9: CALL PKFSGETAPPR(FStn ID=R[130:Dp1x StnI],ApproachPR Reg=95)
  10: CALL PKFSGETRDYDI(FStn ID=R[130:Dp1x StnI],ReadyDI Reg=138)
  11: CALL PKFSGETIXODO(FStn ID=R[130:Dp1x StnI],IndexOutDO Reg=139)
  12: CALL PKFSSETRDY(FStn ID=R[130:Dp1x StnI],Ready=1)
  13: CALL PKFSGETIXSIM(FStn ID=R[130:Dp1x StnI],IndexSim Reg=95)
  14: CALL PKFSGETTRID(FStn ID=R[130:Dp1x StnI],Tray ID Reg=141)
  15: CALL PKFSCLRACK(FStn ID=R[130:Dp1x StnI])
  16: IF R[141:Dp1x TrayId]=0,JMP LBL[10000]
  17: CALL PKFSGETCLRQ(FStn ID=R[130:Dp1x StnI],CLRQ Reg=95)
  18: IF R[95:TemporaryUse1]=0,JMP LBL[1000]
  19: CALL PKFSCLRQUE(FStn ID=R[130:Dp1x StnI])
  20: CALL PKFSPUTQUE(FStn ID=R[130:Dp1x StnI])
  21: --
  22: R[142:Dp1x TrayCtr]=1
  23: LBL[1000]
  24:
  25: LBL[10000: End]
[End]
```

⚠CAUTION

Although this program overwrites R[95:TemporaryUse1] multiple times, some values may be overwritten with other values without being used after being stored, like the value stored in R[95] on line 9. Such values are not referenced in standard TP programs and do not cause problems.
However, when changing a standard TP program before use, take actions such as saving these values in registers other than R[95] that are not overwritten, as needed.

6.11.2.12. 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.

1. **Refresh register [109] which stores the conveyor ID using the PKRBGETPKCV macro.**
2. **Sets the skip outbound FLG to OFF before beginning the pick moves.**
3. **Gets a part from the part queue using PKCSGETQUE.**
4. **Regardless of getting or not getting a part from the queue, if the property "Wait Position" is enabled in the iRPickTool tree view for this conveyor station, it checks if the robot needs to wait at the Wait Position of the Conveyor station which is the Upstream boundary and calculates the wait position using PKCSCALWPOS.PC.**
5. **If the robot needs to wait because there is no part or the part is upstream of the upstream boundary, it moves to the wait position using PK_CV_WAITPOS1.TP.**
6. **If PKCSGETQUE did not return a part, it makes the request to get the part again.**
   If PKCSGETQUE returned a part, it proceeds to step 7.
7. **Sets the UTOOL to the gripper's first zone UTOOL.**
8. **Moves the robot to the approach position.**
9. **If the Skip Outbound FLG came on (implying that the robot cannot pick the part which just crossed the downstream boundary), then the part is returned to the queue with ACK=Skip(3) using the PKCSACKQUE macro and it goes back to the beginning of the program to request a new part to pick.**
10. **Moves the robot to the pick position while executing the "Gripper Close" operation using the Time Before (TB) condition. The gripper close operation is executed via the PKGRCLOSE.PC macro.**
11. **If the Skip Outbound FLG came on (implying that the robot cannot pick the part which just crossed the downstream boundary), then the part is returned to the queue with ACK=Skip(3) using the PKCSACKQUE macro and it goes back to the beginning of the program to request a new part to pick.**
12. **After moving to the pick position, the robot acquires [Pick Delay] using the KAREL program PKGRGETPKDLY and moves with the conveyor for the time in [Pick Delay].**
13. **Checks if the required parts are picked using the PKGRCHKPP.PC macro if the property "Part presence Check" is enabled for this conveyor station operation in the tree view.**
14. **If the part presence check fails, it returns the part to the queue with ACK=Remove (4) and goes back to the beginning of the program to request a part to pick the missing part again.**
15. **If the part presence check succeeds, it sets the payload to "full gripper" payload which is Payload[2].**
16. **Moves the robot to the retreat position.**
17. **If the robot is to pick multiple parts individually, it goes to the beginning of the program to request the next part to pick, otherwise the program ends.**

PK_CV_PICK11.TP

```
  1: --
  2: CALL PKRBGETPKCV(Robot ID=R[103:Robot Id G1],ConvCount=1,Conv ID Reg=109)
  3: CALL PKCVGETNAME(Conv ID=R[109:Pk1x CvId],ConvName SR=4)
 4: CALL PKCSGETID(SR[4],CStn ID Reg=110)
  5:
  6: --
  7: LBL[5]
  8: F[R[102]]=(OFF)
  9: UFRAME_NUM=0
 10: UTOOL_NUM=1
 11:
 12: LBL[10: Pick Loop]
 13: IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
 14: STOP_TRACKING
 15: --
 16: IF R[115:Pk1x UsePosChk]=0,JMP LBL[15]
```

```
 17: --
 18: --
 19: CALL PKCSGETQINRG(CStn ID=R[110:Pk1x Stn],Consec Flag=R[107:Counter G],
       Timeout (ms)=100,R[115:Pk1x UsePosChk],Ref Pos PR=57,Appr Ofst PR=56,Tool Offset=1
       Offset VR=1,Stat Reg=123)
 20: JMP LBL[20]
 21:
 22: LBL[15]
 23: --
 24: CALL PKCSGETQUE(CStn ID=R[110:Pk1x Stn],Consec Flag=R[107:Counter G],
       Timeout (ms)=100,Offset VR=1,Stat Reg=123)
 25:
 26: LBL[20]
 27:
 28:
 29:
 30: LBL[30]
 31: --
 32: CALL PKCSCALWPOS(CStn ID=R[110:Pk1x Stn],Ref Pos PR=57,
       Offset VR=1,GetQ stat=R[123:Pk1x GetQ ],Do WaitMove Reg=95,
       Wait Pos PR=55,NoMove Distance=150)
 33: --
 34: IF(R[95:TemporaryUse1]=0),CALL PK_CV_WAITPOS1(UTOOL Num=R[107:Counter ],
       Appr Ofst PR=56, WaitPos PR=55,PkReachable Reg=115)
 35:
 36: LBL[40:ChkGetQStat]
 37: IF((R[199:BUf Id G1]>0) AND (R[123:Pk1x GetQ stat]>0)),JMP LBL[10000]
 38: IF R[123:Pk1x GetQ stat]<>0,JMP LBL[10]
 39:
 40:
 41: LBL[50]
 42:
 43: LBL[60]
 44:
 45:
 46:
 47: LBL[70]
 48:
 49:
 50:
 51: LBL[80]
 52:
 53: --
 54: UTOOL_NUM=R[107:Counter G1]
 55: IF R[115:Pk1x UsePosChk]=0,JMP LBL[85]
 56: --
 57: CALL PKCSCHKPOS(CStn ID=R[110:Pk1x Stn],Offset VR=1,
       Ref Pos PR=57,Appr Ofst PR=56,Tool Offset=1,Stat Reg=108)
 58: IF R[108:PosNotReach G1]<>0,JMP LBL[1000]
 59:
 60: LBL[85]
 61: --
 62: LPR[57:Pk1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[1] Tool_Offset,
       PR[56:Pk1 Cv Ap Ofst]
 63: --
 64: IF(F[R[102]]),JMP LBL[1000]
 65:
 66: --
 67: CALL PK_GR_GET_ENDZONE1(PickDropCounter=R[107:Counter G],
       Multiple=R[112:Pk1x PkMult],Quantity=R[111:Pk1x Qty],End Zone Reg=106)
```

```
 68: IF R[115:Pk1x UsePosChk]=0,JMP LBL[88]
 69: --
 70: CALL PKCSCHKPOS(CStn ID=R[110:Pk1x Stn],Offset VR=1,
      Ref Pos PR=57,Appr Ofst PR=0,Tool Offset=1,Stat Reg=108)
 71: IF R[108:PosNotReach G1]<>0,JMP LBL[1000]
 72:
 73: LBL[88]
 74: --
 75: LPR[57:Pk1 Cv Ref Pos] max_speed CNT0 VOFFSET,VR[1] TB R[125]sec,
      CALL PKGRCLOSE(Gripper ID=R[104:Gripper Id G],Start Zone=R[107:Counter G1],
      End Zone=R[106:Gr EndZone ])
 76: --
 77: IF(F[R[102]]),JMP LBL[1000]
 78: --
 79: CALL PKGRGETPKDLY(Gripper ID=R[104:Gripper Id G],PickDelay Reg=113)
 80: WAIT R[113]
 81: --
 82: CALL PKCSGETPPCHK(CStn ID=R[110:Pk1x Stn],PartPresChk Reg=116)
 83: IF R[116:Pk1x UsePPChk]=0,JMP LBL[90]
 84: --
 85: --
 86: --
 87: CALL PKGRCHKPP(Gripper ID=R[104:Gripper Id G],Pick=1,Stat Reg=117,
      Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone ])
 88: LBL[90]
 89: PAYLOAD[2:Full Gripper]
 90: --
 91: L PR[57:Pk1 Cv Ref Pos] max_ speed CNT100 VOFFSET,VR[1] Tool_Offset,PR[56:Pk1 Cv Ap Ofst]
 92: IF R[117:Pk1x PPStat]<>0,JMP LBL[1500]
 93: JMP LBL[2000]
 94:
 95:
 96: LBL[1000: Missed part]
 97: F[R[102]]=(OFF)
 98: --
 99: CALL PKCSACKQUE(CStn ID=R[110:Pk1x Stn],Skip=3)
100: IF R[199:Buf Id G1]>0,JMP LBL[10000]
101: JMP LBL[10]
102:
103:
104: LBL[1200: Return part]
105: F[R[102]]=(OFF)
106: --
107: CALL PKCSACKQUE(CStn ID=R[110:Pk1x Stn],Return=2)
108:IF(R[199:Buf Id G1]>0),JMP LBL[10000]
109: JMP LBL[10]
110:
111:
112: LBL[1500: Dropped part]
113: F[R[102]]=(OFF)
114: --
115: CALL PKCSACKQUE(CStn ID=R[110:Pk1x Stn],Delete=4)
116: R[117:Pk1x PPStat]=0
117: IF R[199:Buf Id G1]>0,JMP LBL[10000]
118: JMP LBL[10]
119:
120:
121: LBL[2000: Pick OK]
122: --
123: CALL PKCSACKQUE(CStn ID=R[110:Pk1x Stn],Success=1)
```

```
124: --
125: IF R[112:Pk1x PkMult]=0,JMP LBL[2050]
126: R[107:Counter G1]=R[107:Counter G1]+R[111:Pk1x Qty]
127: JMP LBL[2100]
128:
129: LBL[2050]
130: R[107:Counter G1]=R[107:Counter G1]+1
131: IF R[107:Counter G1]<=R[111:Pk1x Qty],JMP LBL[10]
132:
133: LBL[2100]
134: R[205:Picked G1]=1
135: LBL[10000: End]
136: STOP_TRACKING
137: F[R[102]]=(OFF)
[End]
```

6.11.2.13. PK_CV_DROP11.TP

This program is used by the group 1 robot to perform the first drop or place operation at a conveyor station of a conveyor. This program works as follows.

1. **Refresh register [129] which stores the conveyor ID using the PKRBGETDPCV macro.**
2. **Sets the skip outbound FLG to OFF before beginning the place moves.**
3. **Gets a part from the part queue using PKCSGETQUE.**
4. **Regardless of getting or not getting a part from the queue, if the property "Wait Position" is enabled in the iRPickTool tree view for this conveyor station, it checks if the robot needs to wait at the Wait Position of the Conveyor station which is the Upstream boundary and calculates the wait position using PKCSCALWPOS.PC.**
5. **If the robot needs to wait because there is no part or the part is upstream of the upstream boundary, it moves to the wait position using PK_CV_WAITPOS1.TP.**
6. **Requests the PKCSGETQUE macro until the part is returned.**
7. **Sets the UTOOL to the gripper's first zone UTOOL.**
8. **Moves the robot to the approach position.**
9. **If the Skip Outbound FLG came on (implying that the robot cannot pick the part which just crossed the downstream boundary), then the part is returned to the queue with ACK=Skip(2) using the PKCSACKQUE macro and it goes back to the beginning of the program to request a new part to pick.**
10. **Moves the robot to the place position while executing the "Gripper Open" operation using Time Before (TB) condition.**

    The gripper open operation is executed via the PKGROPEN.PC macro.
11. **If the Skip Outbound FLG came on (implying that the robot cannot pick the part which just crossed the downstream boundary), then the part is returned to the queue with ACK=Skip(2) using the PKCSACKQUE macro and it goes back to the beginning of the program to request a new part to pick.**
12. **After moving to the place position, the robot acquires [Place Delay] using the KAREL program PKGRGETPKDLY and moves with the conveyor for the time in [Place Delay].**
13. **Checks if the required parts are placed using the PKGRCHKPP.PC macro if the property "Part presence Check" is enabled for this conveyor station operation in the tree view.**
14. **If the part presence check fails, it returns the part to the queue with ACK=Remove(4) and goes back to the beginning of the program to request a part to place into again.**
15. **If the part presence check succeeds, it sets the payload to "empty gripper" payload which is Payload[1].**
16. **Moves the robot to the retreat position.**
17. **If the robot is to place multiple parts individually, it goes to the beginning of the program to request the next part to place, otherwise the program ends.**

PK_CV_DROP11.TP

```
 1: --
 2: CALL PKRBGETDPCV(Robot ID=R[103:Robot Id G1],ConvCount=1,Conv ID Reg=129)
 3: CALL PKCVGETNAME(Conv ID=R[129:Dp1x CvId],ConvName SR=5)
 4: CALL PKCSGETID(SR[5],CStn ID Reg=130)
 5:
 6: --
 7: LBL[5]
 8: F[R[102]]=(OFF)
 9: UFRAME_NUM=0
10:
11: LBL[10: Pick Loop]
12: IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
13: STOP_TRACKING
14: --
15: IF R[135:Dp1x UsePosChk]=0,JMP LBL[15]
16: --
17: --
18: CALL PKCSGETQINRG(CStn ID=R[130:Dp1x Stn],Consec Flag=R[107:Counter G],
      Timeout (ms)=100,R[135:Dp1x UsePosChk],Ref Pos PR=59,Appr Ofst PR=58,
      Tool Offset=1,Offset VR=2,Stat Reg=143)
19: JMP LBL[20]
20:
21: LBL[15]
22: --
23: CALL PKCSGETQUE(CStn ID=R[130:Dp1x Stn],Consec Flag=R[107:Counter G],
      Timeout (ms)=100,Offset VR=2,Stat Reg=143 : )
24:
25: LBL[20]
26: --
27:
28:
29: LBL[30]
30: --
31: CALL PKCSCALWPOS(CStn ID=R[130:Dp1x Stn],Ref Pos PR=59,Offset VR=2,
      GetQ Stat=R[143:Dp1x GetQ ],Do WaitMove Reg=95, Wait Pos PR=55,NoMove Distance=150)
32: --
33: IF(R[95:TemporaryUse1]=0),CALL PK_CV_WAITPOS1(UTOOL Num=R[107:Counter ],
      Appr Ofst PR=58, WaitPos PR=55,PkReachable Reg=135)
34:
35: LBL[40:ChkGetQStat]
36: IF((R[199:BUf Id G1]>0) AND (R[143:Dp1x GetQ stat]>0)),JMP LBL[10000]
37: IF R[143:Dp1x GetQ stat]<>0,JMP LBL[10]
38:
39:
40: LBL[50]
41:
42: LBL[60]
43: --
44:
45:
46: LBL[70]
47: --
48:
49:
50: LBL[80]
51:
52: --
53: UTOOL_NUM=R[107:Counter G1]
54: IF R[135:Dp1x UsePosChk]=0,JMP LBL[85]
```

```
 55: --
 56: CALL PKCSCHKPOS(CStn ID=R[130:Dp1x Stn],Offset VR=2,Ref Pos PR=59,
     Appr Ofst PR=58,Tool Offset=1,Stat Reg=108)
 57: IF R[108:PosNotReach G1]<>0,JMP LBL[1000]
 58:
 59: LBL[85]
 60: --
 61: L PR[59:Dp1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[2] Tool_Offset,
     PR[58:Dp1 Cv Ap Ofst]
 62: --
 63: IF (F[R[102]]),JMP LBL[1000]
 64:
 65: --
 66: CALL PK_GR_GET_ENDZONE1(PickDropCounter=R[107:Counter G],Multiple=R[132:Dp1x DpMult],
     Quantity=R[131:Dp1x Qty],End Zone Reg=106)
 67: IF R[135:Dp1x UsePosChk]=0,JMP LBL[88]
 68: --
 69: CALL PKCSCHKPOS(CStn ID=R[130:Dp1x Stn],Offset VR=2,Ref Pos PR=59,
     Appr Ofst PR=0,Tool Offset=1,Stat Reg=108)
 70: IF R[108:PosNotReach G1]<>0,JMP LBL[1000]
 71:
 72: LBL[88]
 73: --
 74: 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 G],Start Zone=R[107:Counter G1],
     End Zone=R[106:Gr EndZone ])
 75: --
 76: IF (F[R[102]]),JMP LBL[1000]
 77: --
 78: CALL PKGRGETDPDLY(Gripper ID=R[104:Gripper Id G],DropDelay Reg=133)
 79: WAIT R[133]
 80: --
 81: CALL PKCSGETPPCHK(CStn ID=R[130:Dp1x Stn],PartPresChk Reg=136)
 82: IF R[136:Dp1x UsePPChk]=0,JMP LBL[90]
 83: --
 84: --
 85: --
 86: CALL PKGRCHKPP(Gripper ID=R[104:Gripper Id G],Place=1,Stat Reg=137,
     Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone ])
 87: LBL[90]
 88: PAYLOAD[1:Empty Gripper]
 89: --
 90: L PR[59:Dp1 Cv Ref Pos] max_speed CNT100 VOFFSET,VR[2] Tool_Offset,
     PR[58:Dp1 Cv Ap Ofst]
 91: IF R[137:Dp1x PPStat]<>0,JMP LBL[1500]
 92: JMP LBL[2000]
 93:
 94:
 95: LBL[1000: Missed part]
 96: F[R[102]]=(OFF)
 97: --
 98: CALL PKCSACKQUE(CStn ID=R[130:Dp1x Stn],Skip=3)
 99: IF R[199:Buf Id G1]>0,JMP LBL[10000]
100: JMP LBL[10]
101:
102: LBL[1200: Return part]
103: F[R[102]]=(OFF)
104: --
105: CALL PKCSACKQUE(CStn ID=R[130:Dp1x Stn],Return=2)
106: IF R[199:Buf Id G1]>0,JMP LBL[10000]
```

```
107: JMP LBL[10]
108:
109:
110: LBL[1500: Dropped part]
111: F[R[102]]=(OFF)
112: --
113: CALL PKCSACKQUE(CStn ID=R[130:Dp1x Stn],Delete=4)
114: R[137:Dp1x PPStat]=0
115: IF R[199:Buf Id G1]>0,JMP LBL[10000]
116: JMP LBL[10]
117:
118:
119: LBL[2000: Drop OK]
120:
121: --
122: CALL PKCSACKQUE(CStn ID=R[130:Dp1x Stn],Success=1)
123: --
124: IF R[132:Dp1x DpMult]=0,JMP LBL[2050]
125: R[107:Counter G1]=R[107:Counter G1]+R[131:Dp1x Qty]
126: JMP LBL[2100]
127:
128: LBL[2050]
129: R[107:Counter G1]=R[107:Counter G1]+1
130: IF R[107:Counter G1]<=R[131:Dp1x Qty],JMP LBL[10]
131:
132: LBL[2100]
133: R[205:Picked G1]=0
134: LBL[10000: End]
135: STOP_TRACKING
136: F[R[102]]=(OFF)
[End]
```

6.11.2.14. PK_CV_WAITPOS1.TP

This program is used by the group 1 robot to move to a wait position at the upstream boundary of a conveyor station of a conveyor when there are no parts to pick.

This program is called by four arguments.

Argument 1:	UTOOL_NUM which is usually the same as the pick or place counter.
Argument 2:	Position Register number of the approach offset of the conveyor station
Argument 3:	Position Register number used for the move to wait position.
Argument 4:	Numeric Register number that contains 0.

This program is used to move the robot to the wait position at the “pick” conveyor station or the “place” conveyor station by passing in different arguments.

PK_CV_WAITPOS1.TP

```
1: --
2: --
3: --
4: --
5: --
6: --
```

```
 7: --
 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: --
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: --
18: L PR[AR[3]] max_speed CNT100 Tool_Offset,PR[AR[2]]
19: LBL[100]
 [END]
```

6.11.2.15. PK_FS_PICK11.TP

This program is used by the group 1 robot to perform the first pick operation at a fixed station.

1. **Refresh register [110] which stores the fixed station ID using the PKRBGETPKFS macro.**
2. **Gets the UFRAME number of the fixed station that you set in the Fixed Station menu of the iRPickTool tree view using the PKFSGETUF.PC macro and sets the system's UFRAME_NUM to it.**
3. **Checks if the fixed station is Ready for pick using the TP macro PK_FS_READY.MR macro.**
4. **If the fixed station is not ready, then repeatedly re-executes the PK_FS_READY macro until the Station is Ready.**

   A station is ready when the following three conditions are TRUE.

   - The fixed station ready DI is ON
   - The Index Out DO is OFF
   - The station ready internal flag is TRUE

   These are interlocks needed for the robot to safely operate in the fixed station without collisions.
5. **Gets a part from the part queue using PKFSGETQUE.**
6. **Requests the PKCSGETQUE macro until the part is returned.**

   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 the 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, it proceeds to step 7.
7. **Sets the UTOOL to the gripper's first zone UTOOL.**
8. **Moves the robot to the approach position.**
9. **If the move to approach succeeded, then it moves the robot to the pick position while executing the "Gripper Close" operation using the Time Before (TB) condition.**

   The gripper close operation is executed via the PKGRCLOSE.PC macro.
10. **After moving to the pick position, the robot acquires [Pick Delay] using the KAREL program PKGRGETPKDLY and waits for the time in [Pick Delay].**
11. **Checks if the required parts have been picked using the PKGRCHKPP.PC macro if the property "Part presence Check" is enabled for this conveyor station in the tree view.**
12. **If the part presence check fails, the part information is returned to the queue with ACK=Return (2) and goes back to the beginning of the program to request a part to place again. (It would get the same part again).**

13 **If the part presence check succeeds, it sets the payload to “full gripper” payload which is Payload[2].**

14 **Moves the robot to the retreat position.**

15 **If the robot is to pick multiple parts individually, it goes to the beginning of the program to request the next part to pick, otherwise the program ends.**

PK_FS_PICK11.TP

```
 1: --
 2: CALL PKRBGETPKFS(Robot ID=R[103:Robot Id G1],FStnCount=1,FStn ID Reg=110)
 3:
 4: --
 5:
 6: R[107:Counter G1]=1
 7:
 8: LBL[5]
 9: IF R[101:Cycle Stop G1]=0,JMP LBL[10000]
10:
11: --
12: --
13: CALL PKFSGETTUF (FStn ID=R[110:Pk1x Stnl],UFrameNum Reg=95)
14: UFRAME_NUM=R[95:TemporaryUse1]
15:
16: LBL[10]
17: --
18: --
19: CALL PK_FS_READY(FStn ID=R[110:Pk1x Stnl],Stat Reg=95)
20: IF R[95:TemporaryUse1]=0,JMP LBL[20]
21: IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
22: WAIT .02(sec)
23: JMP LBL[10]
24:
25: LBL[20: Pick Loop]
26: IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
27:
28: --
29: --
30: CALL PKFSGETQUE(FStn ID=R[110:Pk1x Stnl],Offset VR=1,Stat Reg=123)
31: IF R[123:Pk1x GetQ stat]=0,JMP LBL[200]
32: !Replace FSTNs
33: CALL PK_FS_INDEX(FStn ID=R[110:Pk1x Stnl],CellCounter Reg=122,Stat Reg=95)
34: JMP LBL[5]
35:
36: LBL[200]
37:
38:
39: LBL[300]
40: UTOOL_NUM=R[107:Counter G1]
41: --
42: CALL PK_GR_GET_ENDZONE1(PickDropCounter=R[107:Counter G1],Multiple=R[112:Pk1x PkMult],
      Quantity=R[111:Pk1x Qty],End Zone Reg=106)
43: --
44:L PR[61:Pk1 FS Ref Pos] max_ speed CNT100 VOFFSET,VR[1] Tool_Offset,PR[60:Pk1 FS Ap Ofst]
45: --
46: 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 G],Start Zone=R[107:Counter G1],
      End Zone=R[106:Gr EndZone ])
47: --
48: CALL PKGRGETPKDLY(Gripper ID=R[104:Gripper Id G],PickDelay Reg=113)
49: WAIT R[113]
```

```
  50: --
  51: CALL PKFSGETPPCHK(FStn ID=R[110:Pk1x Stn],PartPresChk Reg=116)
  52: IF R[116:Pk1x UsePPChk]=0,JMP LBL[400]
  53: --
  54: --
  55: --
  56: CALL PKGRCHKPP(Gripper ID=R[104:Gripper Id G],Pick=0,Stat Reg=117,
       Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone ])
  57: LBL[400]
  58: --
  59: PAYLOAD[2:Full Gripper]
  60: --
  61: L PR[61:Pk1 FS Ref Pos] max_ speed CNT100 VOFFSET,VR[1] Tool_Offset,PR[60:Pk1 FS Ap Ofst]
  62: IF R[117:Pk1x PPStat]<>0,JMP LBL[1500]
  63: JMP LBL[2000]
  64:
  65:
  66: LBL[1500: PartGotdropped]
  67: --
  68: CALL PKFSACKQUE(FStn ID=R[110:Pk1x Stnl],Return=2)
  69: R[117:Pk1x PPStat]=0
  70: JMP LBL[20]
  71:
  72:
  73: LBL[2000: Pick ok]
  74: --
  75: CALL PKFSACKQUE(FStn ID=R[110:Pk1x Stnl],Success=1)
  76: --
  77: R[122:Pk1x TrayCtr]=R[122:Pk1x TrayCtr]+1
  78: --
  79: IF R[112:Pk1x PkMult]=0,JMP LBL[2050]
  80: R[107:Counter G1]=R[107:Counter G1]+R[111:Pk1x Qty]
  81: JMP LBL[3000]
  82:
  83: LBL[2050]
  84: R[107:Counter G1]=R[107:Counter G1]+1
  85: IF (R[107:Counter G1]<=R[111:Pk1x Qty],JMP LBL[20]
  86:
  87: LBL[3000: CheckIndex]
  88: !Check whether the trays can be replaced
  89: CALL PKFSGETQUE(FStn ID=R[110:Pk1x Stnl],Offset VR=1,Stat Reg=123)
  90: IF R[123:Pk1x GetQ stat]=0,JMP LBL[3500]
  91: !Replace FSTNs
  92: CALL PK_FS_INDEX(FStn ID=R[110:Pk1x Stnl],CellCounter Reg=122,Stat Reg=95)
  93: JMP LBL[10000]
  94: LBL[3500]
  95: --
  96: CALL PKFSACKQUE(FStn ID=R[110:Pk1x Stnl],Return=2)
  97:
  98:
  99: LBL[10000: End]
[End]
```

6.11.2.16. 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.

1. **Refresh register [130] which stores the fixed station ID using the PKRBGETDPFS macro.**

   Gets the UFRAME number of the fixed station that you set in the Fixed Station menu of the iRPickTool tree view using the PKFSGETUF.PC macro and sets the system's UFRAME_NUM to it.
2. **Checks if the fixed station is Ready for pick using the TP macro PK_FS_READY.MR macro.**
3. **If the fixed station is not ready, then repeatedly re-executes the PK_FS_READY macro until the station is Ready.**

   A station is ready when the following three conditions are TRUE.

   - The fixed station ready DI is ON
   - The Index Out DO is OFF
   - The station ready internal flag is TRUE

   These are interlocks needed for the robot to safely operate in the fixed station without collisions.
4. **Gets a part from the part queue using PKFSGETQUE.**
5. **Requests the PKCSGETQUE macro until the part is returned.**

   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 the 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, it proceeds to step 6.
6. **Sets the UTOOL to the gripper's first zone UTOOL.**
7. **Moves the robot to the approach position.**
8. **If the move to approach succeeded, then it moves the robot to the pick position while executing the "Gripper Open" operation using the Time Before (TB) condition. The gripper open operation is executed via the PKGROPEN.PC macro.**
9. **After moving to the place position, the robot acquires [Place Delay] using the KAREL program PKGRGETPKDLY and moves with the conveyor for the time in [Place Delay].**
10. **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 tree view.**
11. **If the part presence check fails, the part information is returned to the queue with ACK=Return (2) and goes back to the beginning of the program to request a part to place again. (It would get the same part again).**
12. **If the part presence check succeeds, it sets the payload to "empty gripper" payload which is Payload[1].**
13. **Moves the robot to the retreat position.**
14. **If the robot is to place multiple parts individually, it goes to the beginning of the program to request the next part to place, otherwise the program ends.**

PK_FS_DROP11.TP

```
 1: --
 2: CALL PKRBGETDPFS(Robot ID=R[103:Robot Id G1],FStnCount=1,FStn ID Reg=130)
 3:
 4: --
 5:
 6: R[107:Counter G1]=1
 7:
 8: LBL[5]
 9: IF R[101:Cycle Stop G1]=0,JMP LBL[10000]
10:
11: --
12: --
13: CALL PKFSGETTUF(FStn ID=R[130:Dp1x Stnl],UFrameNum Reg=95)
14: UFRAME_NUM=R[95:TemporaryUse1]
15:
16: LBL[10]
17: --
18: --
```

```
19: CALL PK_FS_READY(FStn ID=R[130:Dp1x StnI],Stat Reg=95)
20: IF R[95:TemporaryUse1]=0,JMP LBL[20]
21: IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
22: WAIT .05(sec)
23: JMP LBL[10]
24:
25: LBL[20: Drop Loop]
26: IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
27:
28: --
29: --
30: CALL PKFSGETQUE(FStn ID=R[130:Dp1x StnI],Offset VR=2,Stat Reg=143)
31: IF R[143:Dp1x GetQ stat]=0,JMP LBL[200]
32: !Replace FSTNs
33: CALL PK_FS_INDEX(FStn ID=R[130:Dp1x StnI],CellCounter Reg=142,Stat Reg=95)
34: JMP LBL[5]
35:
36: LBL[200]
37:
38:
39: LBL[300]
40: UTOOL_NUM=R[107:Counter G1]
41: --
42: CALL PK_GR_GET_ENDZONE1(PickDropCounter=R[107:Counter G],Multiple=R[132:Dp1x DpMult],
    Quantity=R[131:Dp1x Qty],End Zone Reg=106)
43: --
44: L PR[63:Dp1 FS Ref Pos] max_ speed CNT100 VOFFSET,VR[2] Tool_Offset,PR[62:Dp1 FS Ap Ofst]
45: --
46: 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 G],Start Zone=R[107:Counter G1],
    End Zone=R[106:Gr EndZone ])
47: --
48: CALL PKGRGETDPDLY(Gripper ID=R[104:Gripper Id G],DropDelay Reg=133)
49: WAIT R[133]
50: --
51: CALL PKFSGETPPCHK(FStn ID=R[130:Dp1x StnI],PartPresChk Reg=136)
52: IF R[136:Dp1x UsePPChk]=0,JMP LBL[400]
53: --
54: --
55: --
56: CALL PKGRCHKPP(Gripper ID=R[104:Gripper Id G],Place=1,Stat Reg=137,
    Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone ])
57: LBL[400]
58: --
59: PAYLOAD[1:Empty Gripper]
60: --
61: L PR[63:Dp1 FS Ref Pos] max_ speed CNT100 VOFFSET,VR[2] Tool_Offset,PR[62:Dp1 FS Ap Ofst]
62: IF R[137:Dp1x PPStat]<>0,JMP LBL[1500]
63: JMP LBL[2000]
64:
65:
66: LBL[1500: PartGotdropped]
67: --
68: CALL PKFSACKQUE(FStn ID=R[130:Dp1x StnI],Return=2)
69: R[137:Dp1x PPStat]=0
70: JMP LBL[20]
71:
72:
73: LBL[2000: Pick ok]
74: --
```

```
  75: CALL PKFSACKQUE(FStn ID=R[130:Dp1x Stnl],Success=1)
  76: --
  77: R[142:Dp1x TrayCtr]=R[142:Dp1x TrayCtr]+1
  78: --
  79: IF R[132:Dp1x DpMult]=0,JMP LBL[2050]
  80: R[107:Counter G1]=R[107:Counter G1]+R[131:Dp1x Qty]
  81: JMP LBL[3000]
  82:
  83: LBL[2050]
  84: R[107:Counter G1]=R[107:Counter G1]+1
  85: IF (R[107:Counter G1]<=R[131:Dp1x Qty]),JMP LBL[20]
  86:
  87: LBL[3000: CheckIndex]
  88: !Check whether the trays can be replaced
  89: CALL PKFSGETQUE(FStn ID=R[130:Dp1x Stnl],Offset VR=2,Stat Reg=143)
  90: IF R[143:Dp1x GetQ stat]=0,JMP LBL[3500]
  91: !Replace FSTNs
  92: CALL PK_FS_INDEX(FStn ID=R[130:Dp1x Stnl],CellCounter Reg=142,Stat Reg=95)
  93: JMP LBL[10000]
  94: LBL[3500]
  95: --
  96: CALL PKFSACKQUE(FStn ID=R[130:Dp1x Stnl],Return=2)
  97:
  98:
  99: LBL[10000: End]
[End]
```

6.11.2.17. 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:	Counter that shows a cell number in the tray at the fixed station
Argument 3:	Number of the register to store the status. If the value of status in this register is non-zero, then an error occurred during the indexing.

This program is run with the following processes.

1. **Determines the Index Out DO for the fixed station based on the operation parameter.**
2. **The program monitors that Ready DI goes low and then goes high using the KAREL program PKFSMONIDX.**
3. **When Ready DI goes high, the program ends.**

   If an error occurs in the KAREL program PKFSMONIDX before Ready DI goes high, the program stores 1, which indicates a fixed station replacement failure, in the argument 3 register as the status and ends.

PK_FS_INDEX.TP

```
  1: --
  2: --
  3: --
  4: --
  5: --
  6:
```

```
 7: R[AR[3]]=0
 8: R[96:TemporaryUse2]=0
 9:
10: --
11: CALL PKFSGETIXODO(FStn ID=AR[1],IndexOutDO Reg=96)
12:
13: --
14: IF R[96:TemporaryUse2]=0,JMP LBL[1500]
15:00
16:00
17: LBL[1000: Do_Idx]
18: --
19: DO[R[96]]=ON
20:
21:
22: --
23: --
24: --
25: --
26: CALL PKFSMONIDX(FStn ID=AR[1],Stat Reg=AR[3])
27: LBL[1500]
28: --
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]
```

6.11.2.18. 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:	Number of the register to store the Ready status. If the value of status in this register is non-zero, then the station is not ready. Not Ready is the status if at least one of the following three conditions is met: Ready DI set to the fixed station operation is low; Index DO is high; a fixed station is being replaced.

This program is run with the following processes.

1. **Determines the Index Out DO for the fixed station based on the operation parameter.**
2. **If the Index Out DO is high, then return “not ready” in the status register argument and return.**
3. **Then determine if the internal station ready flag is FALSE.**

   If replacement is not complete for some reasons, the program stores Not Ready in the argument 2 register as the status and ends.
4. **Then determine if the station ready DI is high.**

   If not high, then return “Not Ready” in the status register argument and return.

5 If all the station ready conditions are satisfied, then return 0 for success in the status register.

PK_FS_READY.TP

```
 1: --
 2: --
 3: --
 4: --
 5: --
 6:
 7: R[96:TemporaryUse2]=0
 8: R[AR[2]]=0
 9:
10: --
11: CALL PKFSGETIXODO(FStn ID=AR[1],IndexOutDO Reg=96)
12:
13: --
14: IF R[96:TemporaryUse2]<=0,JMP LBL[100]
15: IF DO[R[96]]=ON,JMP LBL[1000]
16:
17: LBL[100]
18: --
19: --
20: --
21: CALL PKFSGETRDY(FStn ID=AR[1],FStnReady Reg=97)
22: IF R[97:TemporaryUse3]<>1,JMP LBL[1000]
23:
24: --
25: CALL PKFSGETRDYDI(FStn ID=AR[1],ReadyDI Reg=96)
26:
27: --
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]
```

6.11.2.19. 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:	Number of the first zone
Argument 2:	The “Pick or drop multiple” property for the conveyor station or the fixed station in which the robot is operating. Gets it using the KAREL macro PKCSGETPMUL or PKFSGETPMUL.
Argument 3:	The number of zones that have been added to the gripper
Argument 4:	The end_zone register in which the value of the end zone is returned.

PK_GR_GET_ENDZONE1.TP

```
 1: --
 2: --
 3: --
 4: --
 5: --
 6: --
 7: --
 8: --
 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]
```

6.11.2.20. 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: --
2: --
3: PAUSE
[End]
```

6.12. Setup of pre-grouping

This section describes the procedure for setting up functions that arranges parts on the same conveyor (pre-grouping function) in connection with the Plug & Play gripper and operation.

The setup of the pre-grouping function in relation to other objects is the same as the setup for standard functions. See “I.5.15. Setup of pre-grouping(P.393)”.

For settings specific to North America, in order to use the pre-grouping function, you will need the option for North America R856: iRPickTool / Pre-grouping.

6.12.1. Setting up a gripper and a zone

Regarding the robot that performs arrangement on the same conveyor, there are no special steps for pre-grouping. See "I.6.2. Setting up a gripper and a zone(P.422)".

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. 5.15-3(P.395), add a zone per three parts.

6.12.2. Setup of a robot operation

Regarding the robot that performs arrangement on the same conveyor, there are no special steps for pre-grouping. See "I.6.3. Setting up a robot operation(P.427)".

Regarding the robot that picks all the arranged parts at one shot, you should give attention to conveyor station operation settings and fixed station operation settings.

In conveyor station operations for picking operations, parts that have been put in lines are picked together in one batch, but [Pick or Drop Multiple] is disabled.

In conveyor station operations or fixed station operations for placing operations, [Pick or Drop Multiple] is set up as follows.

In order to place all the parts in one go, enable [Pick or Drop Multiple].

If you will perform a placing operation after performing a picking operation several times, disable [Pick or Drop Multiple] if you will also perform a placing operation in units of parts that have been picked together in one batch.

6.12.3. Fine adjustment of the tracking motion

See "I.5.12. Fine adjustment of the tracking motion(P.386)". As special affairs in the case of a pre-grouping, care needs to be exercised when adjusting the tracking frame error.

6.12.3.1. Adjustment of the tracking frame error (When you use ADJ_OFS)

When using ADJ_OFS, modify the program that performs the arrangement motion shown in "pI.6.12.4.3. PK_PRGR_CV_PICKDROP1.TP(P.491)" 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 to adjust 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 G],Start Zone=1,End Zone=1)
```

- For details, see "I.6.8.1. Adjustment of the tracking frame error(P.439)".

6.12.4. Standard TP programs

This section describes the standard TP programs of pre-grouping.

They works depending on the type of pre-grouping defined as follows.

1. Pre-grouping is used and this is the first robot that performs arrangement on the same conveyor. Virtual trays are generated by this robot. (Robot1)
2. Pre-grouping is used and this is the second or subsequent robot that performs arrangement on the same conveyor. (Robot2)
3. Pre-grouping is used and this is the robot that picks all the arranged parts at one shot. (Robot3)

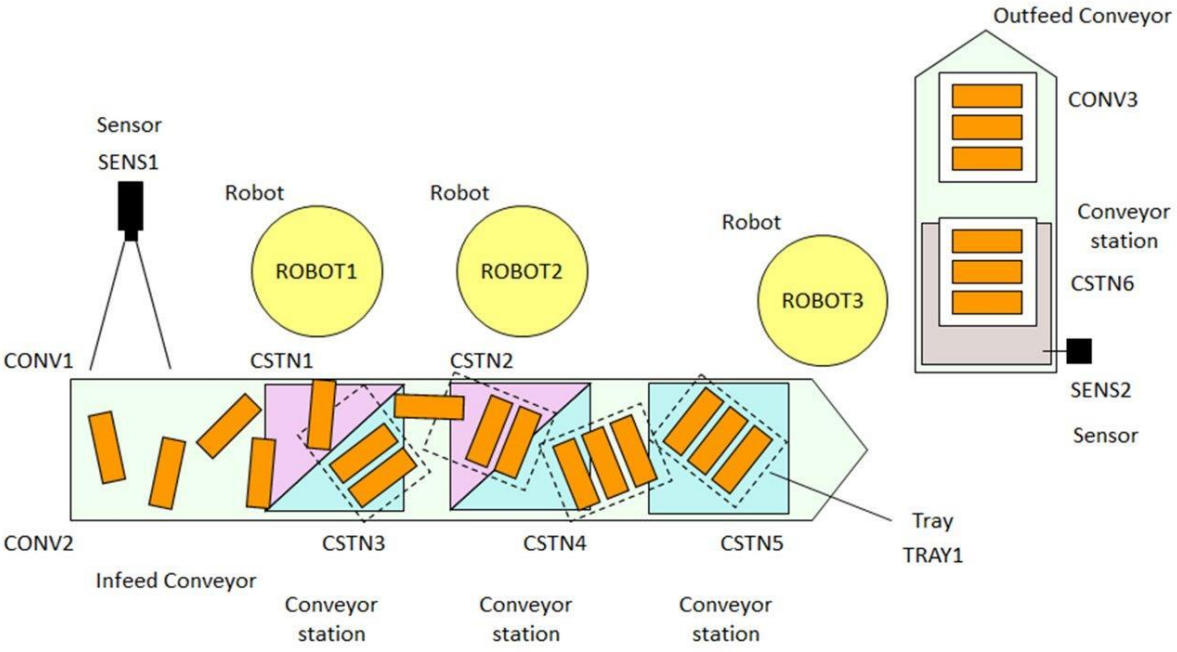


Fig. 6.12.4-1 System configuration

6.12.4.1. PK_PRGR_MAIN1.TP

This program is the main program of pre-grouping for the group 1 robot.

When the type of pre-grouping is 1, this program generates a virtual tray and calls PK_PRGR_CV_PICKDROP1.TP indefinitely. Before generating a virtual tray, always check if cycle stop is invoked. If a virtual tray cannot be generated because of interference, after PKCSGETQUE returns a part, the part is returned to the queue using ACK=Skip (3)

and then this program tries to generate another virtual tray. If a virtual tray cannot be generated because of other reasons, this program tries to generate a virtual tray again after waiting 0.02s.

When the type of pre-grouping is 2, this program calls PK_PRGR_CV_PICKDROP1.TP indefinitely. Before calling PK_PRGR_CV_PICKDROP1.TP, always check if cycle stop is invoked.

PK_PRGR_MAIN1.TP

```
 1: ! Main Prog for Grp:1
 2: ! for Pre-grouping
 3:
 4: IF R[230:PRGR Robot Type]<=0 OR R[230:PRGR Robot Type]>2,JMP LBL[10000]
 5: ;
 6: LBL[100:pk n pl]
 7: IF R[101:Cycle Stop G1]>0,JMP LBL[10000]
 8: ! Generate vtray only in the
 9: ! sensor controller.
10: IF R[230:PRGR Robot Type]<>1,JMP LBL[200]
11: ! Generate Virtual tray
12: ! PKCSGENVRTRY(cstn_id, stat_reg)
13: CALL PKCSGENVRTRY(CStn ID=R[130:Dp1x Stn],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 Stn],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 Stn],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]
```

6.12.4.2. PK_PRGR_INIT1.TP

This program is the initialization program of pre-grouping for the group 1 robot.

This program gets the type of pre-grouping.

PK_PRGR_INIT1.TP

```
1: ! PK_PRGR_INIT1.TP
2: !Initializations for pre-grouping
3:
4: !AR[1]: Register num for PRGR tye
5:
6: !Get PRGR type
7: CALL PKRBGETPRGRT(Pk Conv ID=R[109:Pk1x CvId],Pk CStn ID=R[110:Pk1x S],Dp Conv ID=R
     [129:Dp1x CvId],Dp CStn ID=R[130:Dp1x S],PreGrp Type Reg=AR[1])
8: LBL[10000:End]
[End]
```

6.12.4.3. PK_PRGR_CV_PICKDROP1.TP

This program is used by the group 1 robot to perform arrangement on the same conveyor.

1. **Gets a part from the part queue using PKCSGETQPRGR**
2. **Regardless of getting or not getting a part from the queue, if the property “Wait Position” is enabled in the iRPickTool tree view for this conveyor station, it checks if the robot needs to wait at the Wait Position of the Conveyor station which is the upstream boundary and calculates the wait position using PKCSCALWPOS.PC.**
3. **If the robot needs to wait because there is no part or the part is upstream of the upstream boundary, it moves to the wait position using PK_CV_WAITPOS1.TP.**

4 **If PKCSGETQPRGR did not return a part, the program ends. Here, if the type of pre-grouping is not 1, it waits 0.02 s before ending.**

5 **If PKCSGETQPRGR returned a part, it proceeds to step 6.**

6 **Sets a tracking trigger for the pickup motions using PKCSTRGPRGR**

7 **If PKCSTRGPRGR did not set a tracking trigger, it makes the request to set the tracking trigger again.**

8 **If PKCSTRGPRGR set a tracking trigger, it proceeds to step 9.**

9 **Moves the robot to the approach position.**

10 **Moves the robot to the pick position while executing the "Gripper Close" operation using the Time Before (TB) condition. The gripper close operation is executed via the PKGRCLOSE.PC macro.**

11 **After moving to the pick position, the robot acquires [Pick Delay] using the KAREL program PKGRGETPKDLY and moves with the conveyor for the time in [Pick Delay].**

12 **Sets the payload of the robot to "Full gripper" or Payload[2].**

13 **Moves the robot to the retreat position.**

14 **Sets a tracking trigger for the placement motions using PKCSTRGPRGR**

15 **If PKCSTRGPRGR did not set a tracking trigger, it makes the request to set the tracking trigger again.**

16 **If PKCSTRGPRGR set a tracking trigger, it proceeds to step 17.**

17 **Moves the robot to the approach position.**

18 **Moves the robot to the place position while executing the "Gripper Open" operation using Time Before (TB) condition.**

The gripper open operation is executed via the PKGROPEN.PC macro.

19 **After moving to the place position, the robot acquires [Place Delay] using the KAREL program PKGRGETPKDLY and moves with the conveyor for the time in [Place Delay].**

20 **Sets the payload of the robot to "Empty gripper" or Payload[1].**

21 **Moves the robot to the retreat position.**

22 **Goes to the beginning 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 Stn],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]=(1)),R[96:TemporaryUse2]=(1)
24: !Caclulate WAIT pos
```

```
25: CALL PKCSCALWPOS(CStn ID=R[110:Pk1x Stn],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 Stn],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 Stn],"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 G],Start Zone=1,End Zone=1)
46: ! Pick delay
47: CALL PKGRGETPKDLY(Gripper ID=R[104:Gripper Id G],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 Stn],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 Stn],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 G],Start Zone=1,End Zone=1)
69: ! Place delay
70: CALL PKGRGETDPDLY(Gripper ID=R[104:Gripper Id G],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 Stn],Success=1)
79: JMP LBL[10]
80:
81: LBL[9000]
82: WAIT .02(sec)
83:
84: LBL[10000: End]
85: STOP_TRACKING
[End]
```

6.12.4.4. PK_PRGR_CV_PICKVTRAY1.TP

This program is used by the group 1 robot to perform the pick operation at a conveyor station of a conveyor. The robot picks all the arranged parts at one shot. This program works as follows.

1. **Sets the skip outbound FLG to OFF before beginning the pick moves.**
2. **Gets the number of cells in a tray using PKTRGETNUMCL.**
3. **Gets a part from the part queue using PKCSGETQCELL. The number of cells is used for the ID of the cell in layer 2.**
4. **Regardless of getting or not getting a part from the queue, if the property "Wait Position" is enabled in the iRPickTool tree view for this conveyor station, it checks if the robot needs to wait at the Wait Position of the Conveyor station which is the Upstream boundary and calculates the wait position using PKCSCALWPOS.PC.**
5. **If the robot needs to wait because there is no part or the part is upstream of the upstream boundary, it moves to the wait position using PK_CV_WAITPOS1.TP.**
6. **If PKCSGETQCELL did not return a part, it makes the request to get the part again.**
7. **If PKCSGETQCELL returned a part, it checks whether all cells in layer 1 are filled or not.**

   If some cells in layer 1 are not filled, the program returns part information to the queue using ACK=Skip(3) in the LBL[1000] block, goes back to step 2, and attempts to acquire new part information. If all the cells in layer 1 are filled, the program proceeds to step 8.
8. **Sets the UTOOL to the gripper's first zone UTOOL.**
9. **Moves the robot to the approach position.**
10. **If the Skip Outbound FLG came on (implying that the robot cannot pick the part which just crossed the downstream boundary), then the part is returned to the queue with ACK=Skip(3) using the PKCSACKQUE macro and it goes back to step 2 to request a new part to pick.**
11. **Moves the robot to the pick position while executing the "Gripper Close" operation using the Time Before (TB) condition. The gripper close operation is executed via the PKGRCLOSE.PC macro.**
12. **If the Skip Outbound FLG came on (implying that the robot cannot pick the part which just crossed the downstream boundary), then the part is returned to the queue with ACK=Skip(3) using the PKCSACKQUE macro and it goes back to step 2 to request a new part to pick.**
13. **After moving to the pick position, the robot acquires [Pick Delay] using the KAREL program PKGRGETPKDLY and moves with the conveyor for the time in [Pick Delay].**
14. **Checks if the required parts are picked using the PKGRCHKPP.PC macro if the property "Part presence Check" is enabled for this conveyor station operation in the tree view.**
15. **If the part presence check fails, it returns the part to the queue with ACK=Remove (4) and goes back to step 2 to request a part to pick the missing part again.**
16. **If the part presence check succeeds, it sets the payload to "full gripper" payload which is Payload[2].**

17 **Moves the robot to the retreat position.**

18 **If the robot is to pick multiple virtual trays, it goes to step 2 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 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 Stn],Cell num=R[122:Pk1x TrayCtr],Timeout
    (ms)=100,Offset VR=1,Stat Reg=123,NumUnfilCel Reg=96)
33: CALL PKCSCALWPOS(Stn ID=R[110:Pk1x Stn],Ref Pos PR=57,Offset VR=1,GetQ Stat=R
 [123:Pk1x GetQ ],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 ],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 G],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 G],Start Zone=R[107:Counter G1],End Zone=R[106:Gr
    EndZone ])
55: ! Chk Skip Pick
56: IF (F[R[102]]),JMP LBL[1000]
57: ! Pick delay
58: CALL PKGRGETPKDLY(Gripper ID=R[104:Gripper Id G],PickDelay Reg=113)
59: WAIT R[113]
60: ! Check part pres
61: CALL PKCSGETPPCHK(CStn ID=R[110:Pk1x Stn],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 G],Pick=0,Stat Reg=117,Start Zone=R
  [107:Counter G1],End Zone=R[106:Gr EndZone ])
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 Stn],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 Stn],Delete=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 Stn],Success=1)
93: --
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]
```

6.13. Plug & Play buffer function

This section describes the setup and use of the Plug & Play buffer functions. This function is included in R827: iRPickTool Production Control, which contains the user interface, standard TP programs, and KAREL programs related to this function. The settings for this function are available on the fixed station operation menu. 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 the buffer. As long as there are filled cells, a robot can pick parts from the buffer. However, it does not support being used like an indexer, where a buffer is cleared when all cells are filled, and a new buffer is set. A static buffer with 4 tray cells is shown below:

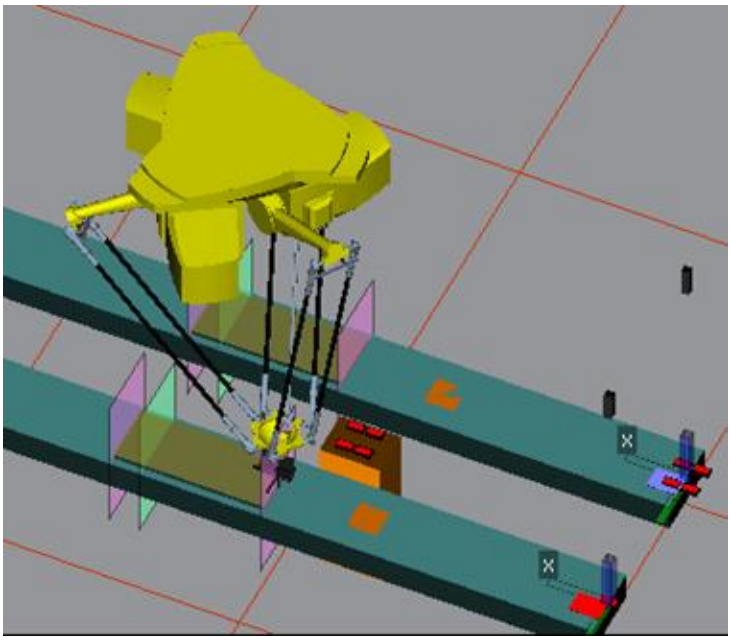


Fig. 6.13-1 A buffer that consists of 4 cells

iRPickTool buffers are supported only when the pick and place 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 place” applications.

6.13.1. Robot pick/place operations in the buffer

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 robot operates according to the following rules to pick parts from the buffer and place them in it.

- When a pick target is available, but a 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 a 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 the target in 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, 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.
- With [Enabled] set to [Wait Position] in the conveyor station operation settings, the robot waits at the wait position on the outfeed conveyor if there are no empty place positions on the outfeed conveyor and the buffer is full (after the robot picks a part).

6.13.2. Setting up a buffer

To create a buffer, create a fixed station in the iRPickTool tree view.

For details on the fixed station, refer to “I.5.7. Setting up a fixed station(P.332)”.

Configure the following settings on the fixed station setup screen.

- Select an appropriate tray in [Tray] and set [Empty] to [Initial Status].
- Set [Buffer] to [Enable].
- Select “First In First Out (FIFO)” or “Last In First Out (LIFO)” for the [Buffer Algorithm].

[Buffer]

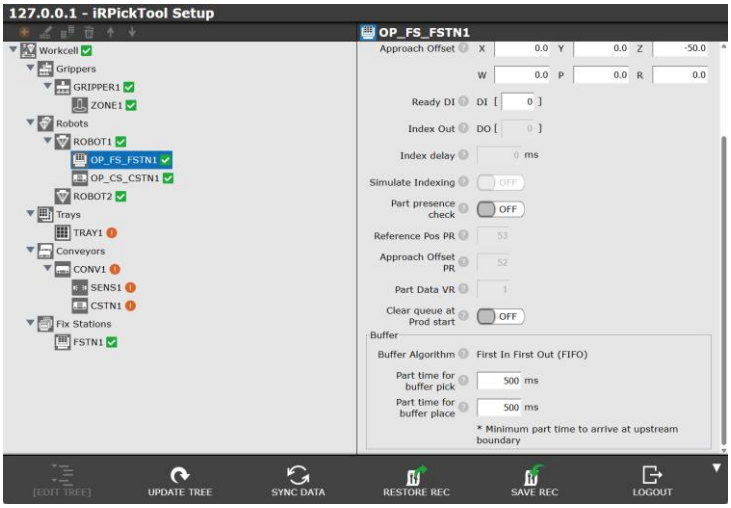
To use the tray as the buffer, set it to [Enable].

[Buffer Algorithm]

Specify the order in which parts are picked out of 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.

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. For details on fixed station robot operation, refer to "I.6.3.2. Fixed station robot operation(P.431)".

[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 based on 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 in a buffer if empty cells exist in it. Set the Part time for buffer place based on 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.

The settings in [Part time for buffer pick] and [Part time for buffer place] in [Buffer] can be changed as needed on the fixed station operation menu even during production. Adjust these times for the robot to pick parts from the buffer and place them in it at appropriate times.

6.13.3. Buffer KAREL macros

The buffer-related KAREL macros are defined in this section.

6.13.3.1. 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 the ID of a Robot.
Argument 2:	Specify the register number 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)
```

6.13.3.2. PKFSGETBUFPT.PC

This program is used to get the Part Time for Buffer pick in milliseconds for the buffer you set in the iRPickTool tree view in the OP_FS_ menu.

Argument 1:	Specify the ID of the fixed station used as a buffer.
Argument 2:	Specify the register number in which to return the Part Time for buffer place 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)
```

6.13.3.3. PKFSGETBUFDT.PC

This program is used to get the Part Time for Buffer place in milliseconds for the buffer you set in the iRPickTool tree view in the OP_FS_ menu.

Argument 1:	Specify the ID of the fixed station used as a buffer.
Argument 2:	Specify the register number 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)
```

6.13.4. Buffer TP programs

This section describes the buffer TP programs. In addition to these programs, you should also be 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 "I.6.11. Standard TP programs(P.458)" for details.

6.13.4.1. 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 tree view into Numeric and Position Registers.

- It clears any pending acks for the buffer using PKFSCLRACK.PC.
- Clears all empty cell information in the buffer using PKFSCLRBUF.PC.
- It initializes the buffer using PKFSPUTBUF.PC.

PK_BF_GET_DATA1.TP

```
1: ! PK_BF_GET_DATA1
```

```
 2: ! Set Num Reg for G1 Buffer
 3:
 4: CALL PKFSGETIXSIM(FStn ID=R[199:Buf Id G1],IndexSim Reg=95)
 5: CALL PKFSGETAPPR(FStn ID=R[199:Buf Id G1],ApproachPR Reg=95)
 6: CALL PKFSSETRDY(FStn ID=R[199:Buf Id G1],Ready)
 7: CALL PKFSCLRACK(FStn ID=R[199:Buf Id G1])
 8: CALL PKFSCLRBUF(FStn ID=R[199:Buf Id G1],Index=1)
 9: CALL PKFSPUTBUF(FStn ID=R[199:Buf Id G1],Index=1)
10: --
11: R[204:Buf Ctr G1]=0
12: R[205:Picked G1]=0
13: R[126:Buf PP Stat G1]=0
14: LBL[1000]
15:
16: LBL[10000: End]
14:
[End]
```

6.13.4.2. PK_BF_PICK1.TP

This program performs 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 “I.6.11.2.15. PK_FS_PICK11.TP(P.479)” to understand the logic of this program.

PK_BF_PICK1.TP

```
 1: --
 2:
 3: LBL[5]
 4: IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
 5:
 6: --
 7: --
 8: CALL PKFSGETTUF (FStn ID=R[199:Buf Id G1],UFrameNum Reg=95)
 9: UFRAME_NUM=R[95:TemporaryUse1]
10:
11: LBL[10]
12: --
13: --
14: CALL PK_FS_READY(FStn ID=R[199:Buf Id G1],Stat Reg=95)
15: IF R[95:TemporaryUse1]=0,JMP LBL[20]
16: IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
17: WAIT .02(sec)
18: JMP LBL[10]
19:
20: LBL[20]
21: IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
22:
23: --
24: --
25: CALL PKFSGETBUF(FStn ID=R[199:Buf Id G1],Index=1,Part,Offset VR=1,
```

```
Stat Reg=123)
26: IF R[123:Pk1x GetQ stat]=0,JMP LBL[200]
27: JMP LBL[10000]
28:
29: LBL[200]
30: CALL PK_GR_GET_ENDZONE1(PickDropCounter=R[107:Counter G],Multiple=0,Quantity=1,
    End Zone Reg=106)
31:
32:
33: LBL[300]
34: UTOOL_NUM=R[107:Counter G1]
35: --
36: L PR[53:Bf Ref Pos] max_speed CNT100 VOFFSET,VR[1] Tool_Offset,
    PR[52:Bf Ap Ofst]
37: --
38: L PR[53:Bf Ref Pos] max_speed CNT0 VOFFSET,VR[1] TB R[127]sec,
    CALL PKGRCLOSE(Gripper ID=R[104:Gripper Id G],Start Zone=R[107:Counter G1],
    End Zone=R[106:Gr EndZone ])
39: --
40: CALL PKGRGETPKDLY(Gripper ID=R[104:Gripper Id G],PickDelay Reg=95)
41: WAIT[95]
42: --
43: CALL PKFSGETPPCHK (FStn ID=R[199:Buf Id G1],PartPresChk Reg=95)
44: IF R[95:TemporaryUse1]=0,JMP LBL[400]
45: --
46: --
47: --
48: CALL PKGRCHKPP(Gripper ID=R[104:Gripper Id G],Pick=0,Stat Reg=126,
    Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone ])
49: LBL[400]
50: --
51: PAYLOAD[2]
52: --
53: L PR[53:Bf Ref Pos] max_ speed CNT100 VOFFSET,VR[1] Tool_Offset,PR[52:Bf Ap Ofst]
54: IF R[126:Buf PP Stat G1]<>0,JMP LBL[1500]
55: JMP LBL[2000]
56:
57:
58: LBL[1500:PartGotdropped]
59: --
60: CALL PKFSACKBUF(FStn ID=R[199:Buf Id G1],Return)
61: R[126:Buf PP Stat G1]=0
62: JMP LBL[10000]
63:
64:
65: LBL[2000: Pick ok]
66: --
67: ! PKFSACKBUF(buf_id, ack, vr (opt
68: CALL PKFSACKBUF(FStn ID=R[199:Buf Id G1],Success)
69: --
70: R[204:Buf Ctr G1]=R[204:Buf Ctr G1]-1
71: --
72: R[107:Counter G1]=R[107:Counter G1]+1
73: R[205:Picked G1]=1
```

```
74:
75: LBL[3000: CheckIndex]
76: ! For indexing buffers only
77:
78:
79: LBL[10000: End]
[End]
```

6.13.4.3. PK_BF_DROP1.TP

This program performs 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 “I.6.11.2.16. PK_FS_DROP11.TP(P.481)” to understand the logic of this program.

PK_BF_DROP1.TP

```
 1: --
 2:
 3: LBL[5]
 4: IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
 5:
 6: --
 7: --
 8: CALL PKFSGETTUF (FStn ID=R[199:Buf Id G1],UFrameNum Reg=95)
 9: UFRAME_NUM=R[95:TemporaryUse1]
10:
11: LBL[10]
12: --
13: --
14: CALL PK_FS_READY(FStn ID=R[199:Buf Id G1],Stat Reg=95)
15: IF R[95:TemporaryUse1]=0,JMP LBL[20]
16: IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
17: WAIT .02(sec)
18: JMP LBL[10]
19:
20: LBL[20]
21: IF R[101:Cycle Stop G1]=1,JMP LBL[10000]
22:
23: --
24: --
25: CALL PKFSGETBUF(FStn ID=R[199:Buf Id G1],Index=1,Cell=1,Offset VR=1,
      Stat Reg=143)
26: IF R[143:Dp1x GetQ stat]=0,JMP LBL[200]
27: JMP LBL[10000]
28:
29: LBL[200]
30: CALL PK_GR_GET_ENDZONE1(PickDropCounter=R[107:Counter G],Multiple=0,Quantity=1,
      End Zone Reg=106)
31:
32:
33: LBL[300]
```

```
34: UTOOL_NUM=R[107:Counter G1]
35: --
36: L PR[53:Bf Ref Pos] max_ speed CNT100 VOFFSET,VR[1] Tool_Offset,PR[52:Bf Ap Ofst]
37: --
38: L PR[53:Bf Ref Pos] max_speed CNT0 VOFFSET,VR[1] TB R[147]sec,
      CALL PKGROPEN(Gripper ID=R[104:Gripper Id G],Start Zone=R[107:Counter G1],
      End Zone=R[106:Gr EndZone ])
39: --
40: CALL PKGRGETDPDLY(Gripper ID=R[104:Gripper Id G],DropDelay Reg=95)
41: WAIT[95]
42: --
43: CALL PKFSGETPPCHK (FStn ID=R[199:Buf Id G1],PartPresChk Reg=95)
44: IF R[95:TemporaryUse1]=0,JMP LBL[400]
45: --
46: --
47: --
48: CALL PKGRCHKPP(Gripper ID=R[104:Gripper Id G],Place=1,Stat Reg=126,
      Start Zone=R[107:Counter G1],End Zone=R[106:Gr EndZone ])
49: LBL[400]
50: --
51: PAYLOAD[1]
52: --
53: L PR[53:Bf Ref Pos] max_ speed CNT100 VOFFSET,VR[1] Tool_Offset,PR[52:Bf Ap Ofst]
54: IF R[126:Buf PP Stat G1]<>0,JMP LBL[1500]
55: JMP LBL[2000]
56:
57:
58: LBL[1500:PartGotdropped]
59: --
60: CALL PKFSACKBUF(FStn ID=R[199:Buf Id G1],Return)
61: R[126:Buf PP Stat G1]=0
62: JMP LBL[10000]
63:
64:
65: LBL[2000: Pick ok]
66: --
67: ! PKFSACKBUF(buf_id, ack, vr (opt
68: CALL PKFSACKBUF(FStn ID=R[199:Buf Id G1],Success)
69: --
70: R[204:Buf Ctr G1]=R[204:Buf Ctr G1]+1
71: --
72: R[107:Counter G1]=R[107:Counter G1]+1
73: R[205:Picked G1]=0
74:
75: LBL[3000: CheckIndex]
76: ! For indexing buffers only
77:
78:
79: LBL[10000: End]
[End]
```

6.13.4.4. 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.PC, checking its status, and then returning the part back to the buffer.

This program is called by two arguments.

Argument 1:	ID of the fixed station used as buffer
Argument 2:	The register number in which the status is returned. If a part exists, 1 is returned, otherwise 0 is returned.

PK_BF_OK2PICK1.TP

```
 1: !PK_BF_OK2PICK1
 2: ! Arguments
 3: ! Arg 1: Buf Id
 4: ! Arg 2: ok2Pick reg (0 or 1)
 5:
 6: ! Get a part from buf
 7: ! PKFSGETBUF(buf_id, index,
 8: ! part_or_cell, vr, stat_reg,
 9: ! model_id (opt))
10: CALL PKFSGETBUF(FStn ID=AR[1],Index=2,Part,Offset VR=1,Stat Reg=123)
11: IF R[123:Pk1x GetQ stat]>0,JMP LBL[1000]
12: ! Ok
13: ! Return the part or cell
14: CALL PKFSACKBUF(FStn ID=AR[1],Return)
15: R[AR[2]]=1
16: JMP LBL[10000]
17:
18: LBL[1000: Not ok]
19: R[AR[2]]=0
20:
21: LBL[10000: End]
[End]
```

6.13.4.5. 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.PC, checking its status and then returning the part back to the buffer.

This program is called by two arguments.

Argument 1:	ID of the fixed station used as buffer
Argument 2:	The register number in which the status is returned. If an empty cell exists, 1 is returned, otherwise 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]
```

6.13.4.6. 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 can be performed only if the part on the infeed is upstream by at most the “part time for buffer pick”. Otherwise, it is necessary to check if a pick can be performed at the buffer. Use PKCSGETTIME.PC to acquire the time for a part to reach the tracking area.

This program is called by six arguments.

Argument 1:	ID of the conveyor to which the conveyor station belongs
Argument 2:	ID of the fixed station used as buffer
Argument 3:	The order (refer to Argument 2 in “I.5.8.5.8. PKCSGETTIME.PC(P.354)”)
Argument 4:	Consec Flag (refer to Argument 3 in “I.5.8.5.8. PKCSGETTIME.PC(P.354)”)
Argument 5:	The register number where “Part time for buffer pick” is returned
Argument 6:	The register number in which the status is returned. If a part exists, 1 is returned, otherwise 0 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(CONV1=SR[4],CStn ID Reg=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 Stn],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
64:
35: LBL[10000: End]
[End]
```

6.13.4.7. 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 can be performed only if the part on the outfeed is upstream by at most the “part time for buffer place”. Otherwise, it is necessary to check if a place can be performed at the buffer. Use PKCSGETTIME.PC to acquire the time for the place position to reach the tracking area.

This program is called by six arguments.

Argument 1:	ID of the conveyor to which the conveyor station belongs
Argument 2:	ID of the fixed station used as buffer
Argument 3:	The order (refer to Argument 2 in “I.5.8.5.8. PKCSGETTIME.PC(P.354)”)
Argument 4:	Consec Flag (refer to Argument 3 in “I.5.8.5.8. PKCSGETTIME.PC(P.354)”)
Argument 5:	The register number where “Part time for buffer place” is returned
Argument 6:	The register number in which the status is returned. If a drop position exists, 1 is returned, otherwise 0 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 Stn],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]
```

6.13.4.8. PK_BUF_PICK1.TP

This program is called from PK_PICK1.TP. This program decides if the part should be picked from a buffer (PK_BF_PICK1.TP) or from the infeed conveyor (PK_CV_PICK11.TP).

PK_BUF_PICK1.TP

```
  1: ! PK_BUF_PICK1
  2: ! Buffer Pick Grp1
  3:
  4: LBL[50: Buf loop]
  5: IF R[205:Picked G1]>0,JMP LBL[10000]
  6: R[202:Buf Available G1]=0
  7: R[203:CS Available G1]=0
  8: R[97:TemporaryUse3]=0
  9: LBL[100]
```

```
10: IF R[129:Dp1x CvId]>0,CALL PK_CV_OK2DROP1(Conv ID=R[129:Dp1x CvId],
    Buffer ID=R[199:Buf Id G1],Order=1,Drop Counter=1,BufDropTime Reg=201,Ok2Drop Reg=97)
11: CALL PK_BF_OK2PICK1(Buffer ID=R[199:Buf Id G1],Ok2Pick Reg=202)
12: CALL PK_CV_OK2PICK1(Conv ID=R[109:Pk1x CvId],Buffer ID=R[199:Buf Id G1],
    Order=1,Pick Counter=R[107:Counter G1],BufPickTime Reg=200,Ok2Pick Reg=203)
13: IF R[203:CS Available G1]>0,CALL PK_CV_PICK11
14: IF ((R[203:CS Available G1]<=0) AND (R[202:Buf Available G1]>0) AND
    (R[97:TemporaryUse3]>0)),CALL PK_BF_PICK1
15: IF R[205:Picked G1]<1,CALL PK_CV_PICK11
16: IF R[205:Picked G1]<1,JMP LBL[50]
17: LBL[10000]
[End]
```

6.13.4.9. PK_BUF_DROP1.TP

This program is called from PK_DROP1.TP. This program decides if the part should be placed in a buffer (PK_BF_DROP1.TP) or on the outfeed conveyor (PK_CV_DROP11.TP).

PK_BUF_DROP1.TP

```
 1: ! PK_BUF_DROP1
 2: ! Buffer Drop Grp1
 3:
 4: LBL[50: Buf loop]
 5: IF R[205:Picked G1]<1,JMP LBL[10000]
 6: R[202:Buf Available G1]=0
 7: R[203:CS Available G1]=0
 8: LBL[100]
 9: CALL PK_BF_OK2DROP1(Buffer ID=R[199:Buf Id G1],Ok2Drop Reg=202)
10: CALL PK_CV_OK2DROP1(Conv ID=R[129:Dp1x CvId],Buffer ID=R[199:Buf Id G1],
    Order=1,Drop Counter=[107:Counter G1],BufDropTime Reg=201,Ok2Drop Reg=203)
11: IF R[203:CS Available G1]>0,CALL PK_CV_DROP11
12: IF ((R[203:CS Available G1]<=0) AND (R[202:Buf Available G1]>0) AND (R[205:Picked G1]>0)),
    CALL PK_BF_DROP1
13: IF R[205:Picked G1]>0,CALL PK_CV_DROP11
14: IF R[205:Picked G1]>0,JMP LBL[50]
15: LBL[10000]
[End]
```

6.14. iRPickTool safe retreat with skip outbound motion

If the robot encounters the “LNTK-042 Skip outbound move” error during or after picking a part (or during/after placing a part) on a conveyor, it will skip the retreat motion that should follow.

As shown in the Fig. 6.14-1, if the robot is placing parts into a deep tray or box, skipping the retreat motion may cause the robot to collide with the side-wall of the current tray as it moves to the next taught position.

NOTE

The “LNTK-042 Skip outbound move” error occurs when the part is very close to the downstream boundary and the internal tracking software has determined that the robot does not have sufficient time to pick up the part before it exits the downstream boundary. Therefore, by default, the motions that would cause the robot to go beyond the downstream boundary are skipped, and the robot moves to pick the next part on the conveyor.

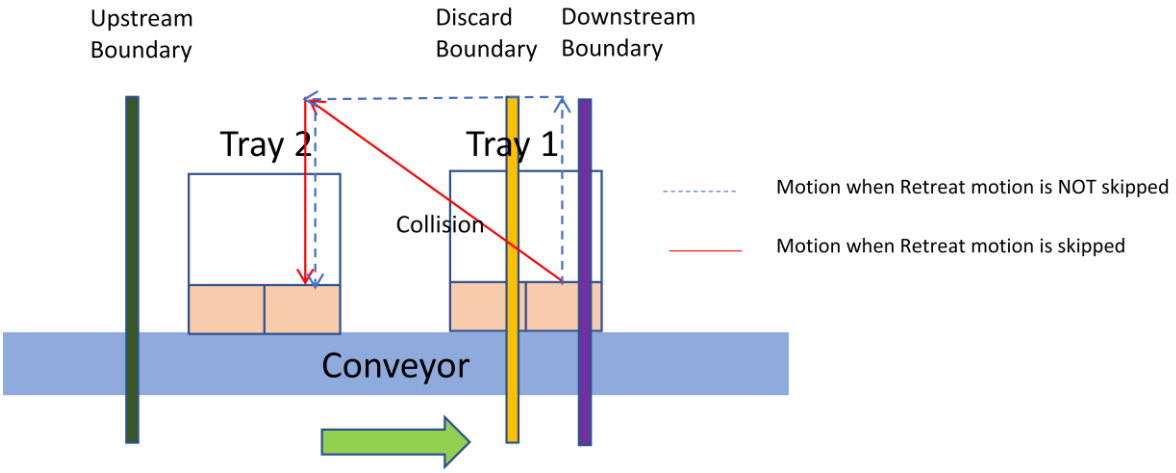


Fig. 6.14-1 "LNTK-042 Skip outbound move" error

The TP program PK_SAFE_RETREAT.TP has been provided with J945: iRPickTool to move the robot up perpendicular to the conveyor if the retreat motion is skipped. The retreat motion is indicated by a dotted line in Fig. 6.14-1. PK_SAFE_RETREAT.TP can be called in the tracking pick (PK_CV_PICK11.TP) or tracking place (PK_CV_DROP11.TP) program to avoid this issue.

PK_SAFE_RETREAT.TP program is shown below.

PK_SAFE_RETREAT.TP takes two arguments

Argument 1:	Utool number.
Argument 2:	Tool_offset that is used to safely retreat the robot.

When using PK_SAFE_RETREAT.TP, modify the program PK_CV_PICK11.TP if the Skip Outbound error occurs on the infeed conveyor, or PK_CV_DROP11.TP if the Skip Outbound error occurs while placing the part.

PK_SAFE_RETREAT.TP is available only with J945: iRPickTool.

The example below shows the usage of the PK_SAFE_RETREAT.TP program in the tracking pick program PK_CV_PICK11.TP. Note that if the "LNTK-042 Skip outbound move" error occurs then the Flag F[R[102]] comes on. You can call PK_SAFE_RETREAT.TP when the Flag turns on.

```
 73:  LBL[88]
 74:  --
 75: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])
   :
 76:  --
 77:  IF (F[R[102]]),
   :  CALL PK_SAFE_RETREAT(
   :  UT num=R[107:Counter G1],
   :  TOffst PR=56)
 78:  IF (F[R[102]]),JMP LBL[1000]
 79:  --
 80:  CALL PKGRGETPKDLY(
```

```
 89:  LBL[90]
 90:  PAYLOAD[2:Full Gripper]
 91:  --
 92:L  PR[57:Pk1 Cv Ref Pos] max_speed
   :  CNT100 VOFFSET,VR[1]
   :  Tool_Offset,PR[56:Pk1 Cv Ap Ofst]
   :
 93:  IF (F[R[102]]),
   :  CALL PK_SAFE_RETREAT(
   :  UT num=R[107:Counter G1],
   :  TOffst PR=56)
 94:  IF R[117:Pk1x PPStat]<>0,
```

6.15. Fixed approach height for trays with multiple layers of cells

When placing a part to a tray or box with multiple layers, the approach height will change as the robot moves to higher layers as shown in Fig. 6.15-1. The additional higher motion may result in inefficiency for tall trays. Also, robot motion limit errors may occur, which may cause the robot to pause.

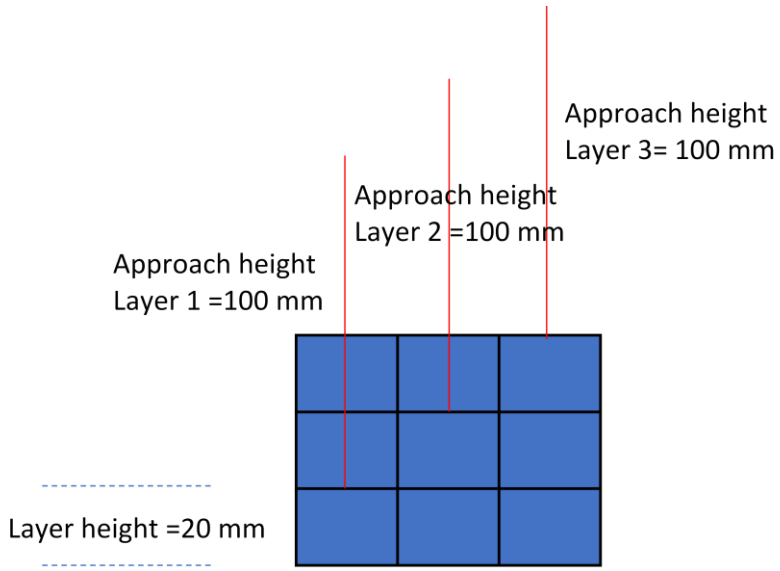


Fig. 6.15-1 The approach height the robot has to go to increases with each higher layer

The users may use the following KAREL programs to move the robot to a fixed approach height regardless of the number of layers in the tray.

- PKCSFIXAPRZ.PC
  For multi-layer trays on a conveyor.
- PKFSFIXAPRZ.PC
  For multi-layer trays on a fixed station.

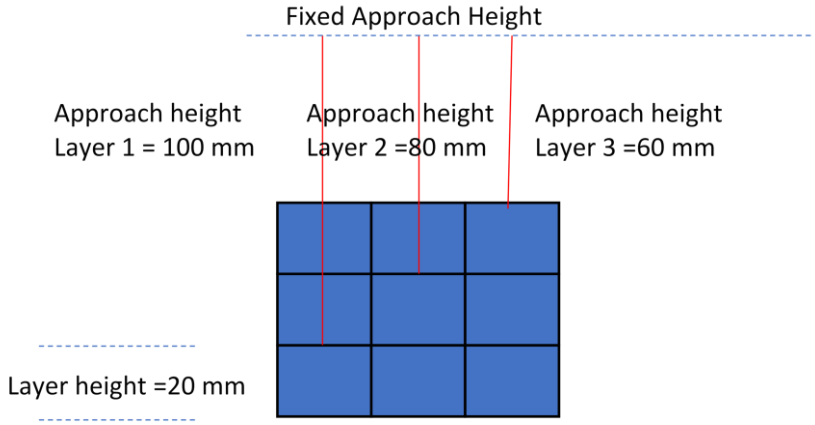


Fig. 6.15-2 Fixed approach height

This function is available only with J945: iRPickTool.

6.15.1. PKCSFIXAPRZ

The program PKCSFIXAPRZ.PC can be called from the conveyor picking program PK_CV_PICK11.TP or in the drop program PK_CV_DROP11.TP.

Argument 1:	Specify the ID of a conveyor station. The ID can be acquired using PKCSGETID described in 6.8.5.1 “PKCSGETID.PC”.
Argument 2:	Specify the offset type. “Tool Offset” is the default and it normally does not need to be modified. Offset is applied relative to the UTOOL with the motion to the approach position in PK_CV_PICK11.TP or PK_CV_DROP11.TP.

NOTE

To apply the offset to the tracking frame with the motion to the approach position in PK_CV_PICK11.TP or PK_CV_DROP11.TP, modify argument 2 to “Offset”. In addition, check and modify the TP program so that the offset can be applied to the tracking frame properly.

Argument 3: Specify the processing status of PKCSGETQUE.
The processing status of PKCSGETQUE is stored in the register whose number is specified in Argument 5 of PKCSGETQUE.

Argument 4: Specify the number of register storing the processing status.
If a value other than 0 is stored in the register with the specified number, it indicates that this function did not operate normally. In this case, the program is aborted. This does not usually happen if settings and arguments are not defective.

Example:

```
PKCSFIXAPRZ(CStn ID=R[110:Pk1x StndId], Tool Offset, GetQ Stat = R[123: Pk1x GetQ stat], Stat Reg=1)
```

Example of using in the PK_CV_PICK11.TP program. PKCSFIXAPRZ needs to be called after the PKCSGETQUE call. You should also call it before the move to the approach position.

```
 46:  LBL[80]
 47:  CALL PKCSFIXAPRZ(
   :  CStn ID=R[110:Pk1x StnId],
   :  Tool Offset,
   :  GetQ Stat=R[123:Pk1x GetQ stat],
   :  Stat Reg=1)
 48:  --
 49:  UTOOL_NUM=R[107:Counter G1]
 50:  IF R[115:Pk1x UsePosChk]=0,
   :  JMP LBL[85]
```

6.15.2. PKFSFIXAPRZ

The program PKFSFIXAPRZ.PC can be called from the fixed station picking program PK_FS_PICK11.TP or in the drop program PK_FS_DROP11.TP

Argument 1: Specify the ID of a fixed station.
The ID can be acquired using PKFSGETID.

Argument 2: Specify the offset type. “Tool Offset” is the default and it normally does not need to be modified.
Offset is applied relative to the UTOOL with the motion to the approach position in PK_FS_PICK11.TP or PK_FS_DROP11.TP.

NOTE

To apply the offset to the UFrame of the fixed station with the motion to the approach position in PK_FS_PICK11.TP or PK_FS_DROP11.TP, modify argument 2 to “Offset”. In addition, check and modify the TP program so that the offset can be applied to the UFrame of fixed station properly.

Argument 3: Specify the number of register storing the processing status.
If a value other than 0 is stored in the register with the specified number, it indicates that this function did not operate normally. In this case, the program is aborted. This does not usually happen if settings and arguments are not defective.

Example:

```
PKFSFIXAPRZ(FStn ID=R[110:Pk1x StnI],Tool Offset,Stat Reg=1)
```

Example of using in the PK_FS_PICK11.TP program. PKFSFIXAPRZ needs to be called before the move to the approach position.

```
39:  LBL[300]
40:  UTOOL_NUM=R[107:Counter G1]
41:  -- Set Pick/drop registers
42:  CALL PK_GR_GET_ENDZONE1(
  :  PickDropCounter=R[107:Counter G1]
  :  Multiple=R[112:Pk1x PkMult],
  :  Quantity=R[111:Pk1x Qty],
  :  End Zone Reg=106)
43:  -- Move to approach
44:  CALL PKFSFIXAPRZ(
  :  FStn ID=R[110:Pk1x StnId],
  :  Tool Offset,Stat Reg=1)
45:  -- Move to pick pos
46:L  PR[61:Pk1 FS Ref Pos] max_speed
```

6.16. iRPickTool robot maximum speed specification to prevent duty failures

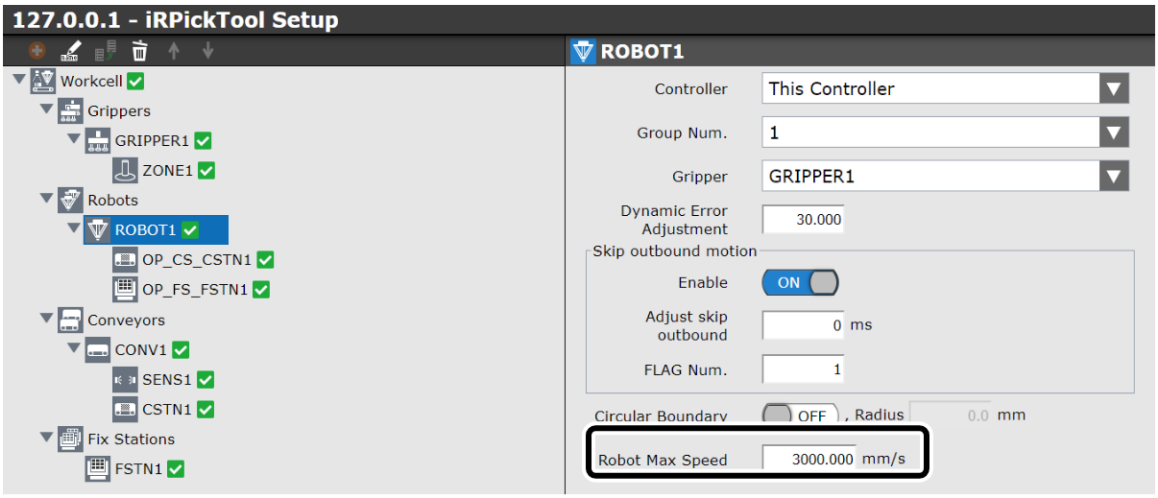
The J945: iRPickTool TP programs use the max_speed instruction for achieving the highest speed motions. An example of using max_speed is shown below.

```
86: L PR[57:Pk1 Cv Ref Pos] max_ speed CNT100 VOFFSET,VR[1]
     Tool_Offset,PR[56:Pk1 Cv Ap Ofst] ;
```

In iRPickTool high-speed applications users may run into duty cycle issues such as overheat alarms (which will stop the robot and require a power cycle) while running the robot at default max_speed for very long periods of time. To prevent the duty errors, you can set the proper value of the “Robot Max Speed” in the iRPickTool “Robot” object as shown in the figure below. Once it is set, this value will be applicable to all the TP programs that use the max_speed instruction.

The default value of the “Robot Max Speed” varies depending on the robot model. Refer to the value displayed on the setup screen.

The value of the “Robot Max Speed” is also saved to the recipes.



6.17. Setup of multiple stations

iRPickTool (J945) supports the operation of picking and placing parts from multiple conveyors or fixed stations. From this point forward, the term “station” will denote both “conveyor” and “fixed station”.

This feature simplifies the workflow for setting up these pick and place operations and reduces the user's program registration lead time. When multiple stations are set up, the pick or place sequence is handled based on the station ID. This allows the following robot systems to be set up:

- Multiple conveyors are used for pick and one fixed station is used for place.
- One conveyor is used for pick and multiple conveyors are used for place.

Multiple stations cannot be set up if any of the following conditions apply:

- Five or more stations are used for pick and place.
- Different types of stations are used in combination either for pick or place. For example, one conveyor and one fixed station are used in combination.
- A group 2 robot is present. Systems with multiple robot controllers are not supported.
- Pre-grouping is not supported.

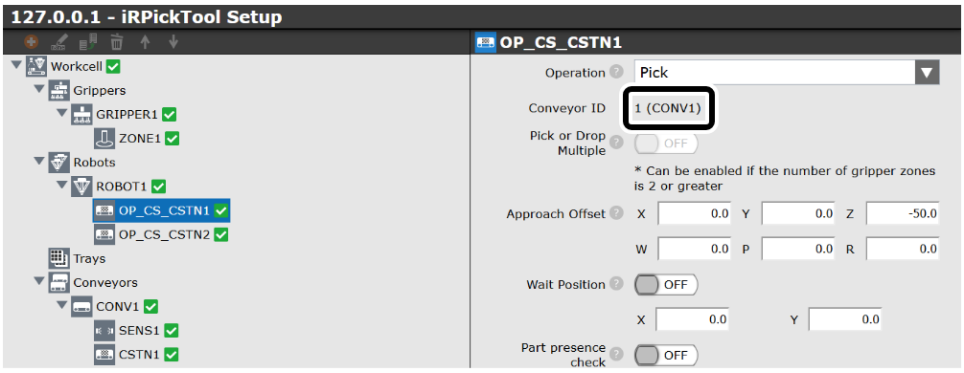
How to check station IDs

The ID and name of the station to be operated are displayed on the detailed settings screen for conveyor station operation or fixed station operation.

Multiple stations are processed based on this ID. This cannot be changed.

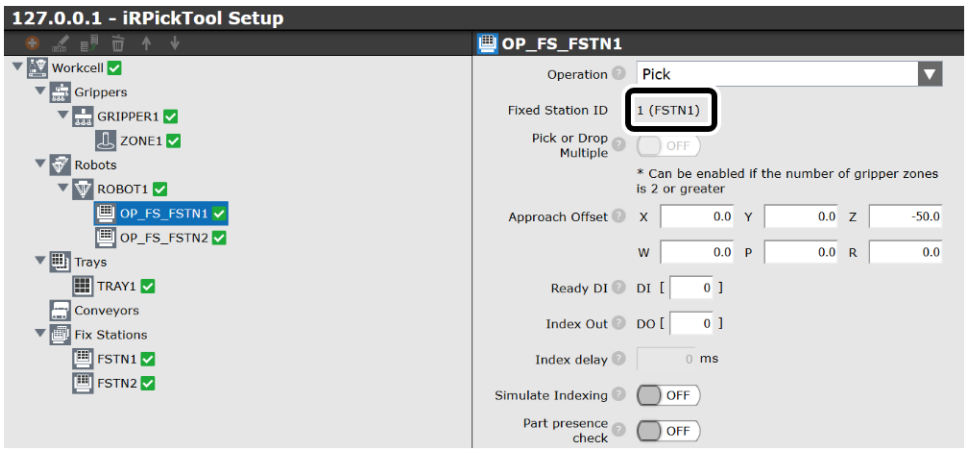
For conveyor station operations:

The [Conveyor ID] item displays the ID and name of the station performing the operation.



For fixed station operations:

The [Fixed Station ID] item displays the ID and name of the station performing the operation.

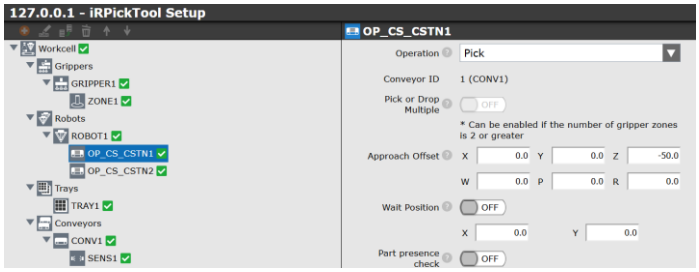


6.17.1. Setting up the operation and loading programs

To start setting up multiple stations, configure the same [Operation] setting for all stations.
The following example shows the configuration of two pick conveyors.

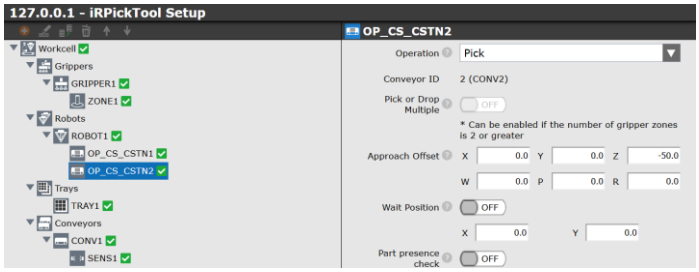
1 **In the tree view, click “OP_CS_CSTN1” under ROBOT1.**

2 **Select [Pick] from the [Operation] drop-down list.**



3 **In the tree view, click “OP_CS_CSTN2” under ROBOT1.**

4 **Select [Pick] from the [Operation] drop-down list.**

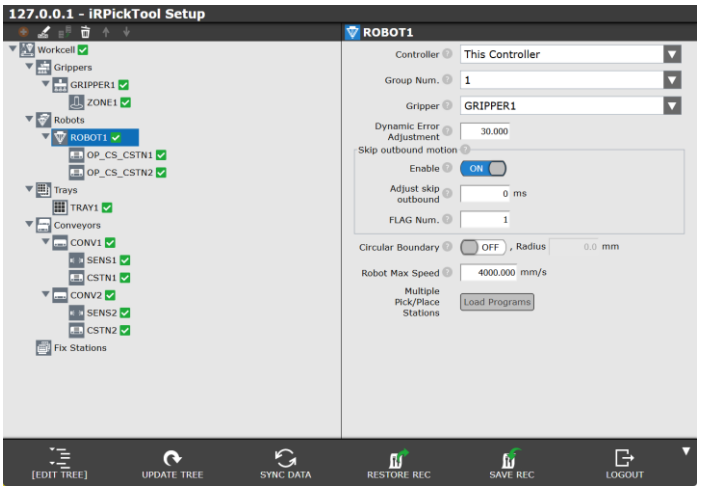


5 **For the place setting, as in the example above, select [Place] from the [Operation] drop-down list.**

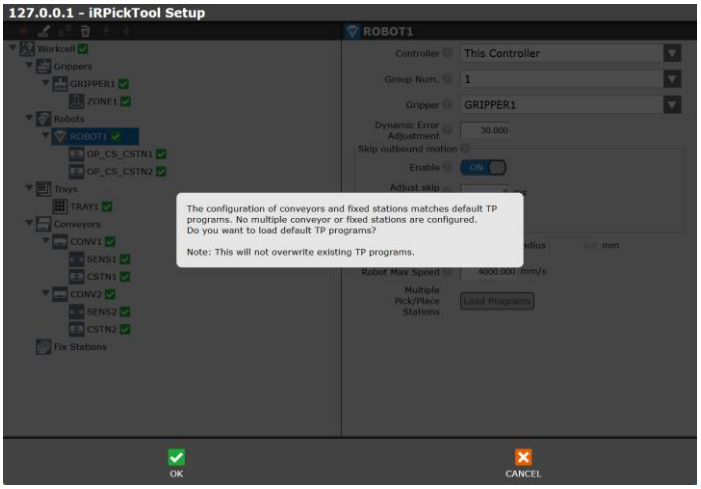
6 **Click on ROBOT1 in the tree view.**

7 **Click on [Load Programs] at the bottom of the screen, then click [OK] to confirm loading of the program.**

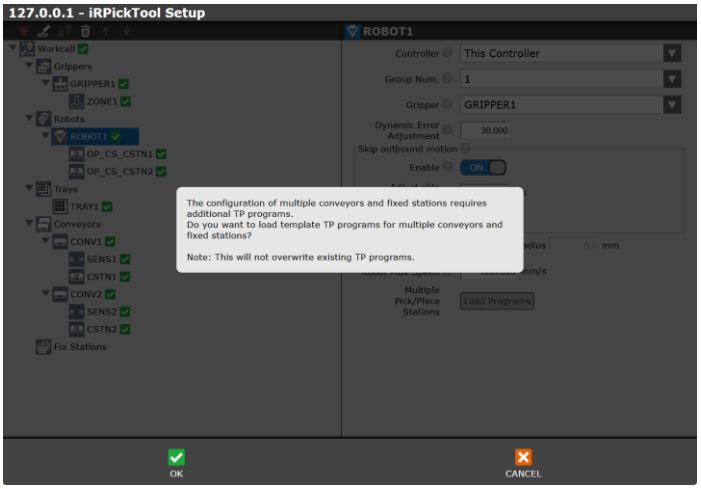
The supported TP programs are loaded into the robot controller. If another robot within the inter-robot communication is selected, the request is forwarded and the program is loaded there.



If you do not configure multiple stations by following the above procedure, the supported programs will not be loaded when you click [Load Programs]. However, this process will restore Group 1 and the 1st (single station) iRPickTool standard TP programs if they have been deleted. For the list of the restoration target TP programs, refer to “I.6.11.2.1. List of all the Standard TP programs(P.460)”. Note that the restoration target TP programs that already exist at the time of restoration will not be reverted to their initial state, even if their contents have been modified.



If multiple stations are configured, the loading process restores the standard iRPickTool TP program and loads TP programs that support multiple stations. When loading is complete, the number of TP programs loaded is displayed.



The following are programs that are loaded onto the robot controller. They are the TP programs for sequences 2 through 4 (2nd to 4th stations).

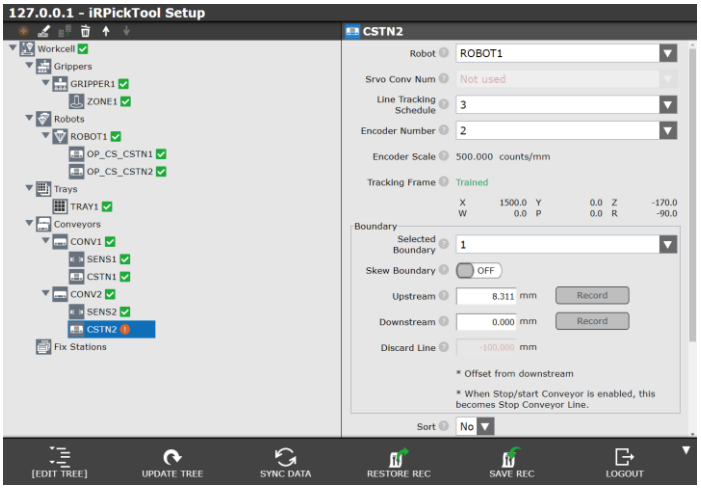
- PK_CV_GET_PICK_DATA1[2-4].TP
- PK_FS_GET_PICK_ DATA1[2-4].TP
- PK_CV_GET_DROP_ DATA1[2-4].TP
- PK_FS_GET_DROP_ DATA1[2-4].TP
- PK_CV_PICK1[2-4].TP
- PK_FS_ PICK1[2-4].TP
- PK_CV_ DROP1[2-4].TP
- PK_FS_DROP1[2-4].TP

6.17.2. Setting the pick conveyor

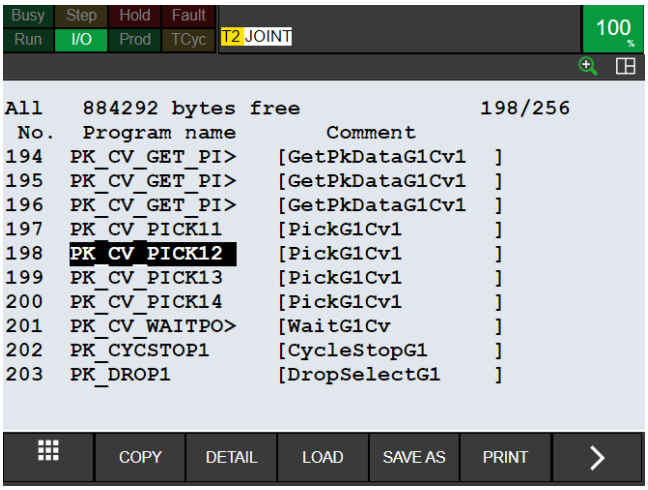
Set up the pick station using the following procedure:

1. **In the tree view, click “OP_CS_CSTN”.**
2. **Select [Pick] from the [Operation] drop-down list.**

The default [Line Tracking Schedule] for sequence 2 is 3. The default for “Selected Boundary” is 1.
Make sure the settings match the example for CSTN2 below.

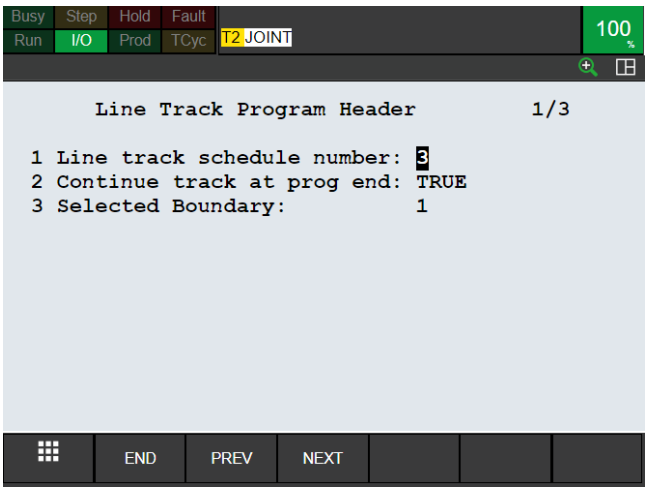


3 The correct encoder number must be set on the conveyor station and the tracking frame must be taught with CONV2. Before setting the sensor position in SENS2, press the [SELECT] key on the teaching pendant to go to the program list screen, and check [DETAIL] of PK_CV_PICK12.TP.



4 Click [NEXT] to view tracking program details.

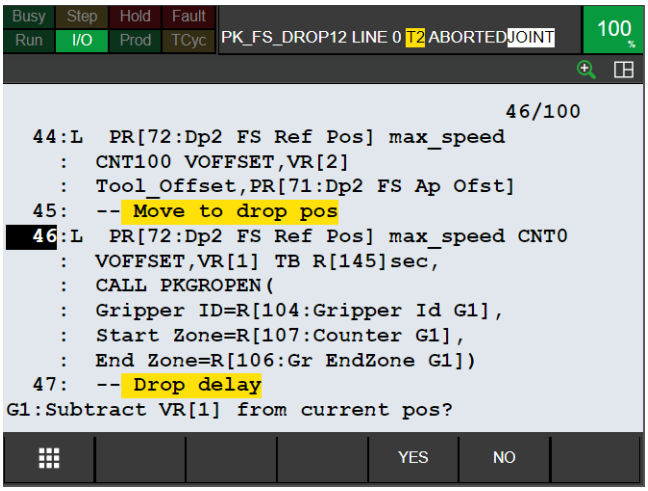
5 Verify that the line tracking schedule and the selected boundary match the CSTN2 settings.



6 Click [END].

7 Follow the guide to set the sensor position with SENS2. Teach the pick position according to the pick order.

In the following example, the order is 2. During teaching, if the question, “Do you want to subtract VR[1] to correct the position?”, appears, select [NO].



NOTE

To enable part picking from multiple conveyors, ensure that “Line Tracking Schedule” and “Selected Boundary” in the conveyor station settings match those specified in the pick TP program. Match the CSTN settings with those in the following pick TP program.

- In sequence 1, set 1 for [Line Tracking Schedule] and 1 for [Selected Boundary], the same as in PK_CV_PICK11.TP.
- In sequence 2, set 3 for [Line Tracking Schedule] and 1 for [Selected Boundary], the same as in PK_CV_PICK12.TP.
- In sequence 3, set 5 for [Line Tracking Schedule] and 1 for [Selected Boundary], the same as in PK_CV_PICK13.TP.
- In sequence 4, set 7 for [Line Tracking Schedule] and 1 for [Selected Boundary], the same as in PK_CV_PICK14.TP.

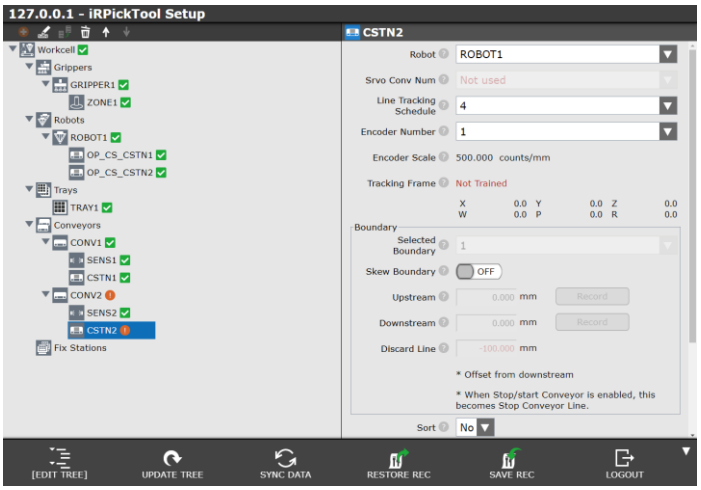
6.17.3. Setting the place conveyor

Set up the place station using the following procedure:

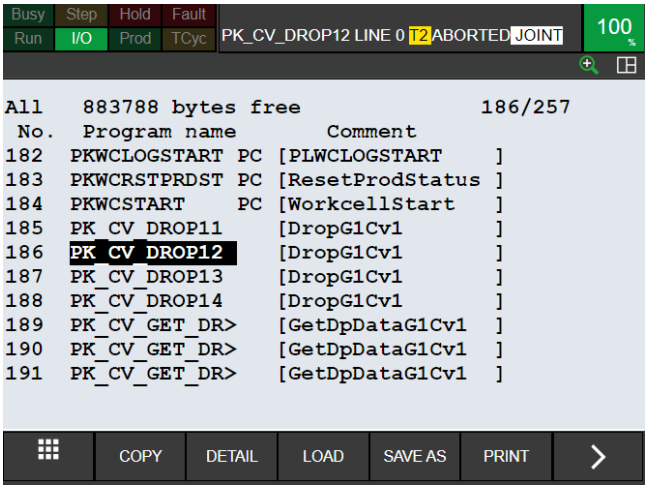
1. **In the tree view, click “OP_CS_CSTN”.**
2. **From [Operation], select [Place].**

   The default [Line Tracking Schedule] for sequence 2 is 4. The default for “Selected Boundary” is 1.

   Make sure the settings match the example for CSTN2 below.

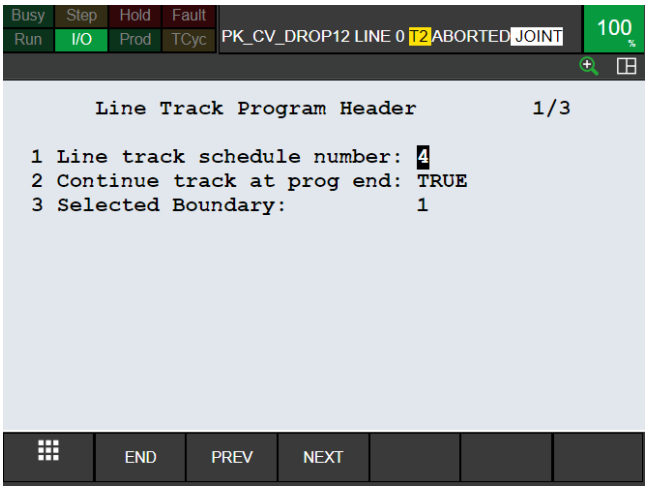


3. **The correct encoder number must be set on the conveyor station and the tracking frame must be taught with CONV2. Press the [SELECT] key on the teaching pendant to go to the program list screen, and check [DETAIL] of PK_CV_DROP12.TP.**



4 Click [NEXT] to view tracking program details.

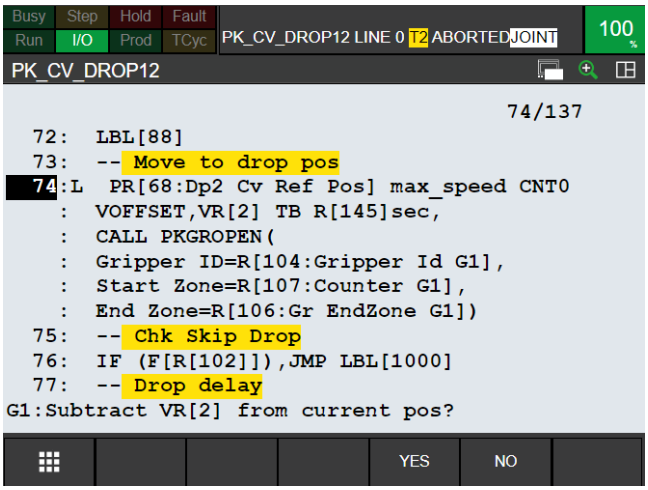
5 Verify that the line tracking schedule and the selected boundary match the CSTN2 settings.



6 Click [END].

7 Follow the guide to set the sensor position with SENS2. Teach the place position according to the place order.

In the following example, the order is 2. During teaching, if the question, “Do you want to subtract VR[2] to correct the position? ”, appears, select [NO].



NOTE

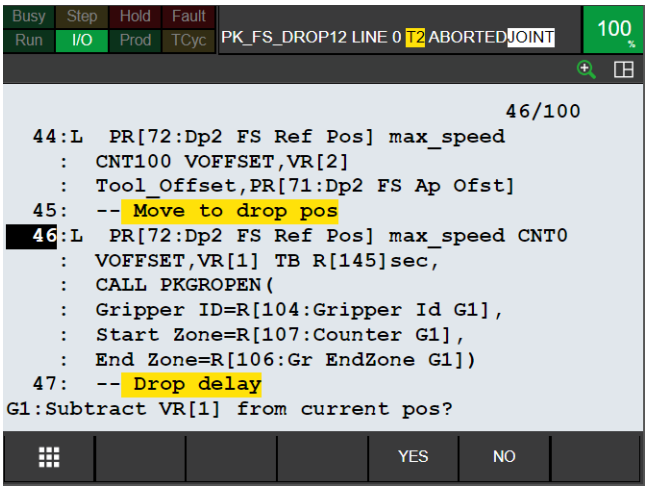
To enable part placement from multiple conveyors, ensure that “Line Tracking Schedule” and “Selected Boundary” in the conveyor station settings match those specified in the placement TP program. Match the CSTN settings with those in the following place TP program.

- In sequence 1, set 2 for [Line Tracking Schedule] and 1 for [Selected Boundary], the same as in PK_CV_DROP11.TP
- In sequence 2, set 4 for [Line Tracking Schedule] and 1 for [Selected Boundary], the same as in PK_CV_DROP12.TP.
- In sequence 3, set 6 for [Line Tracking Schedule] and 1 for [Selected Boundary], the same as in PK_CV_DROP13.TP.
- In sequence 4, set 8 for [Line Tracking Schedule] and 1 for [Selected Boundary], the same as in PK_CV_DROP14.TP.

6.17.4. Setup of a fixed station

Set the fixed station to pick or place using the following procedure:

1. **In the tree view, select OP_FS_FSTN.**
2. **Select [Pick] or [Place] for [Operation].**
3. **See “I.5.7. Setting up a fixed station(P.332)” to teach the position according to the procedure described.**
   For example, for placement order 2, teach the placement position in PK_FS_DROP12.TP.



7. Setting example appropriate to the usage

This chapter presents specific examples of setup in accordance with the use of iRPickTool Basic.

7.1. Using different robots to pick parts depending on their positions on the conveyor

This section describes the case in which different robots are used to pick parts depending on their positions on the conveyor.

In the example presented here, there are two robots, and the upstream robot picks parts on the right half of the conveyor while the downstream robot picks parts on the left half. A camera is used to detect parts on the conveyor.

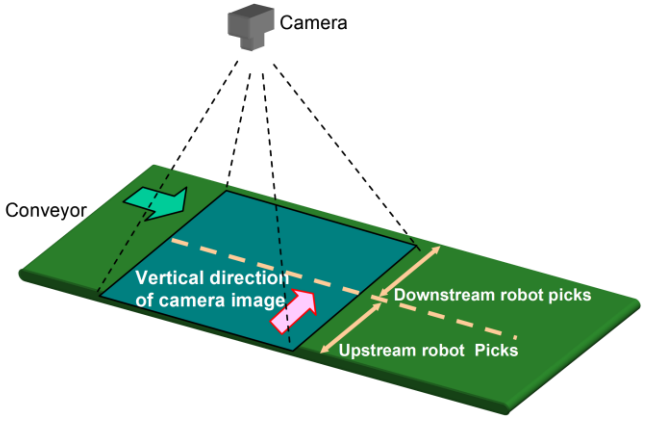


Fig. 7.1-1 Example of a setup where the robot that picks a part up is changed depending on the position of the part on the conveyor

In addition, the following assumptions are made.

- The conveyor configuration is defined as shown in Fig. 7.1-2.
  This is the same configuration as Configuration 1 shown in "I.1.4. Sample system configurations(P.12)".

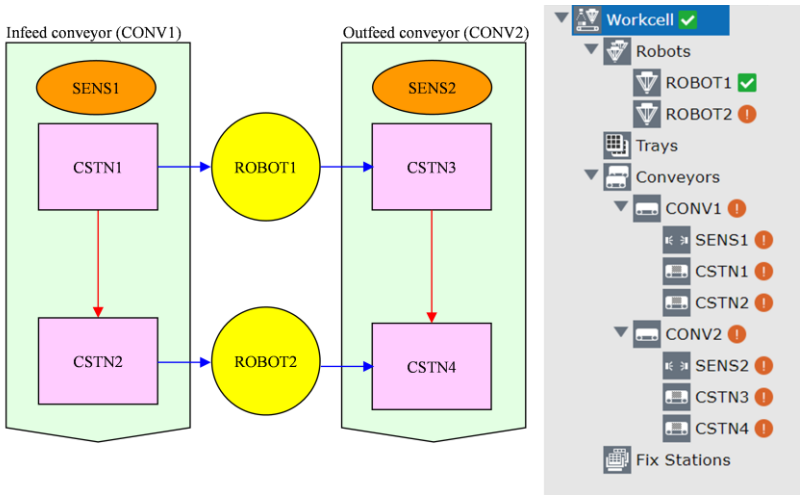
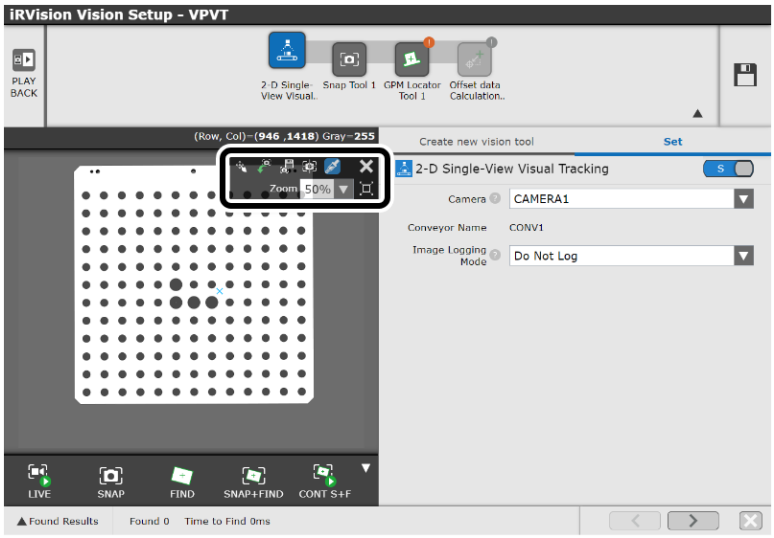


Fig. 7.1-2 Conveyor configuration example

- The camera is installed so that the horizontal direction of the camera image is parallel to the moving direction of the conveyor. That is, the width direction of the conveyor is aligned with the vertical direction of the camera image. In 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 Vt coordinate on the detected image is larger than the Vt coordinate of the conveyor center, select 2 for the model ID using a conditional execution tool. (For the Vt coordinate of the conveyor center, check using “Sampling Tool”.)

Checking the Vt coordinates of pixels with iRVision

You can check the position of a pixel that you have selected on an image using the iRVision sampling tool.



1 Click the button.

The sampling tool will be enabled, and will be displayed at the upper left.

2 Click any pixel on the image.

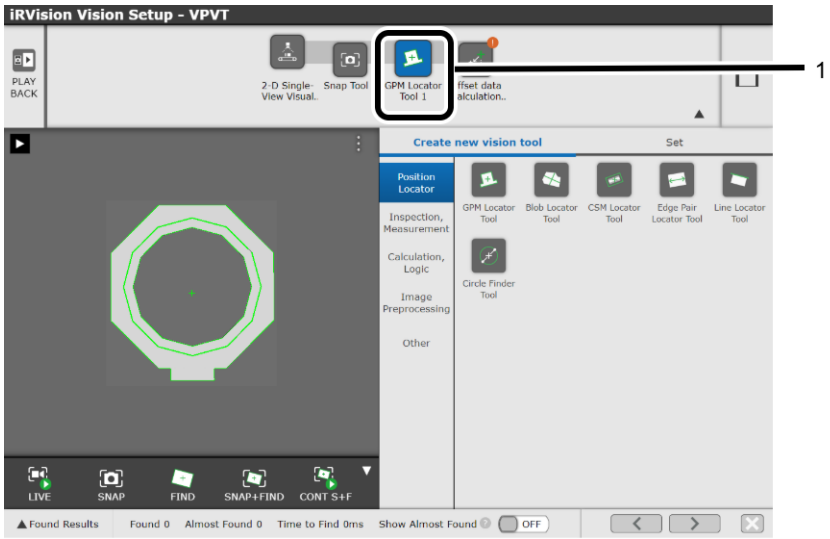
The mark moves and the position of the pixel (Vt, Hz) will be displayed at the upper right.

Inserting the conditional execution tool

Insert the conditional execution tool into the vision process.

1 Select the GPM Locator Tool icon in the Program Line and click the [Add new vision tool] tab.

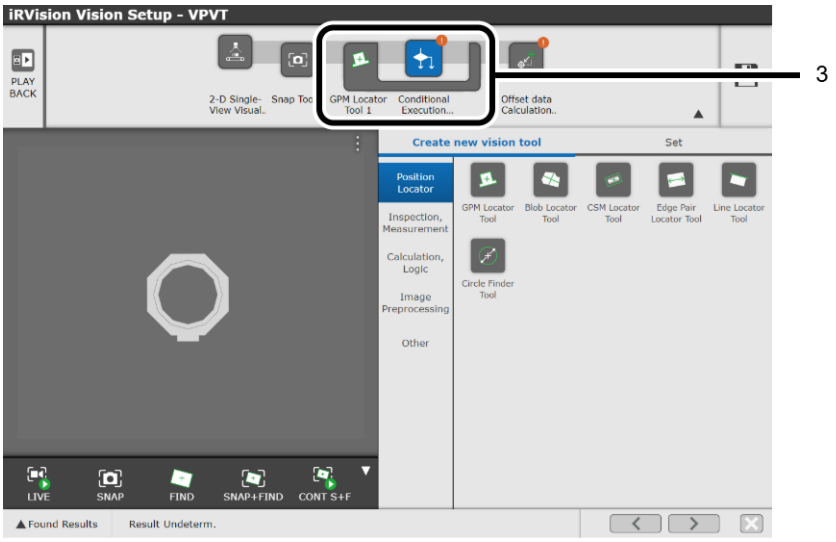
The following screen appears.



2 Select the [Conditional Execution Tool] icon in the [Calculation, Logic] category.



3 Drag and drop the [Conditional Execution Tool] icon onto the Program Line so that the Conditional Execution Tool becomes a child tool of the GPM Locator Tool.

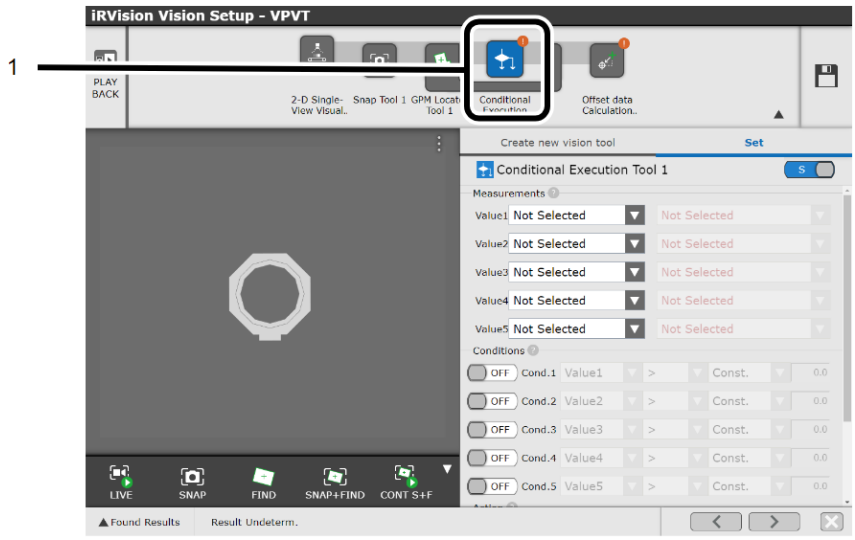


Setting the conditional execution tool

Set it so that if the Vt coordinate is larger than the Vt coordinate of the conveyor center (here, the explanation supposes it to be 240), 1 is added to the model ID.

1 Select [Conditional Execution Tool1].

The following screen appears.

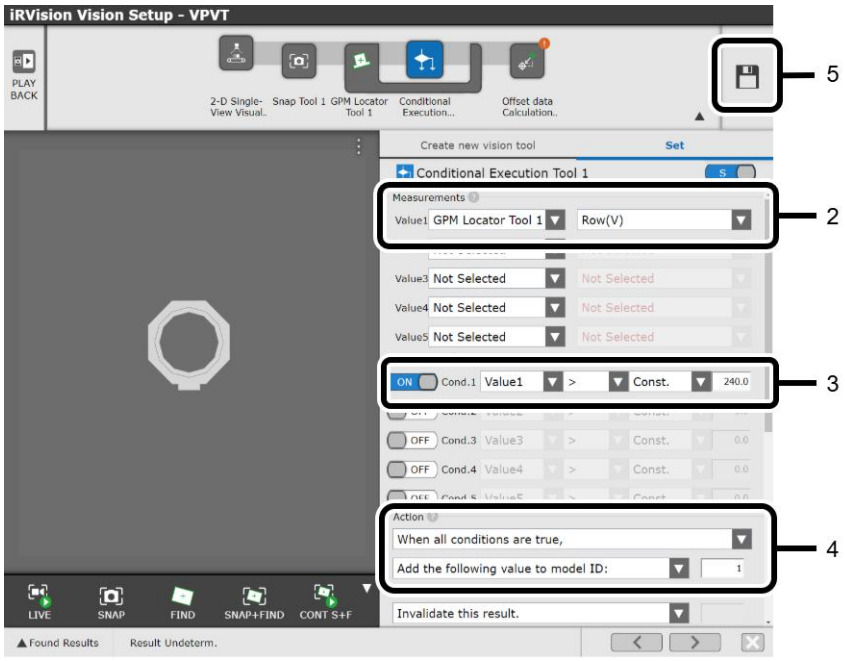


2 **Under [Measurements], select [GPM Locator Tool 1] and [Row(V)] next to [Value1].**

3 **Under [Conditions], check [Cond.1] and select [Value1], [>], [Const.], and [240].**

4 **Under [Action], select [Add the following value to model ID], and specify 1.**

5 **Click [SAVE].**



6 **Click [×].**

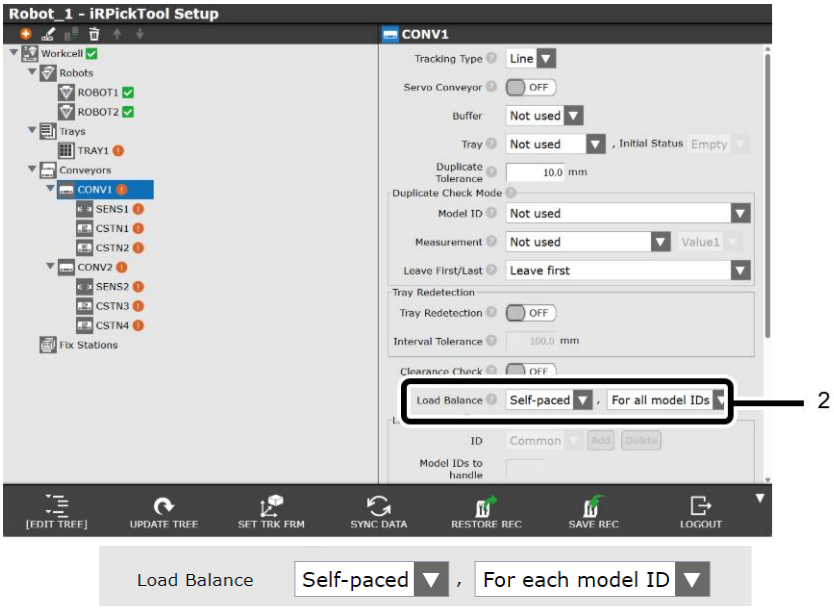
Setting the load balance

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.

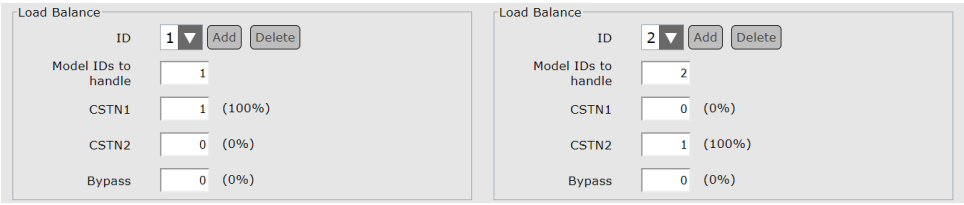
1 **Open the conveyor setup screen. After a while, the following screen appears.**

2 **Select [Self-paced] or [Keep rate] from the drop-down list on the left side of [Load Balance] and select [For each model ID] from the drop-down list on the right side.**



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.**



7.2. 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 one without a hole - are conveyed and allocated to different robots based on whether there is a hole or not. The following figure assumes that the target parts are nuts of two models.

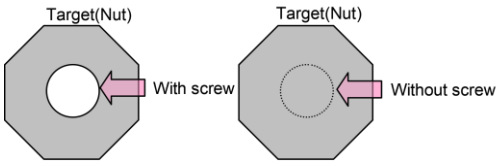


Fig. 7.2-1 Example of a partial characteristic of a part (a part with and without a hole in one section)

In addition, the following assumptions are made.

- Whether a hole is present is determined by measuring the standard deviation of brightness with a histogram tool. If a nut has an even surface, the standard deviation of brightness becomes large when a hole is present and becomes small when a hole is not present.

	With a hole	Without a hole
Standard deviation	Large	Small

⚠CAUTION

The tendency described above may not apply depending on differences in the background and other factors. The information provided here is for reference purposes only.

NOTE

Whether a hole is present can also be determined using the blob tool or GPM Locator Tool. Here, the method that uses a histogram tool is described as an example.

- The part model ID taught to the GPM Locator Tool is 1.
- If a hole is present, the model ID is set to 2 using the conditional execution tool.
- The conveyor is defined as shown in Fig. 7.2-2.
  This is the same as Configuration 1 shown in "I.1.4. Sample system configurations(P.12)".

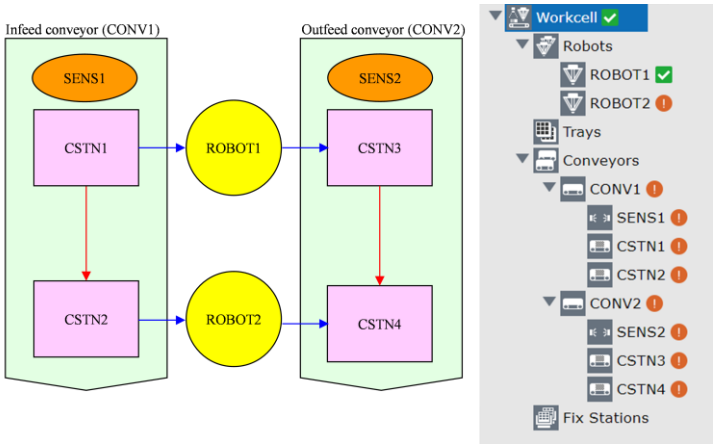


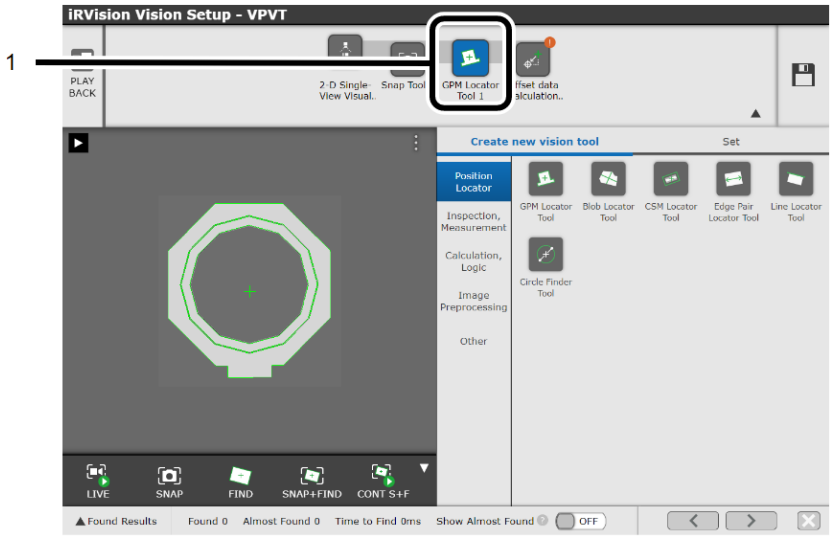
Fig. 7.2-2 Conveyor configuration example

Inserting the histogram tool

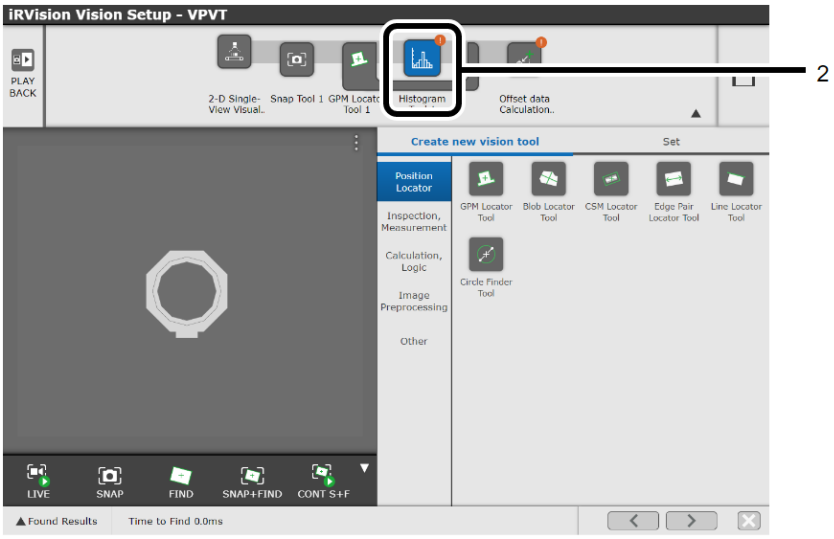
Insert the histogram tool to the vision process.

1 Select the GPM Locator Tool 1 icon in the Program Line and click the [Create new vision tool] tab.

The following screen appears.



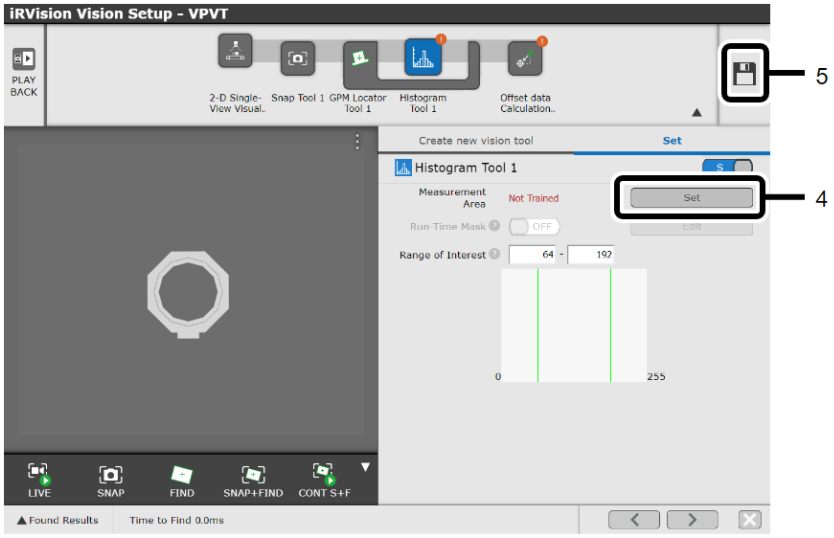
2 **Drag and drop the [Histogram Tool] icon onto the Program Line so that the Histogram Tool becomes a child tool of the GPM Locator Tool.**



3 **Select [Histogram Tool 1] in the Program Line.**

4 **Click the [SET] button to teach [Measurement Area].**

5 **Click [SAVE].**



6 **Click [×].**

Setting the conditional execution tool

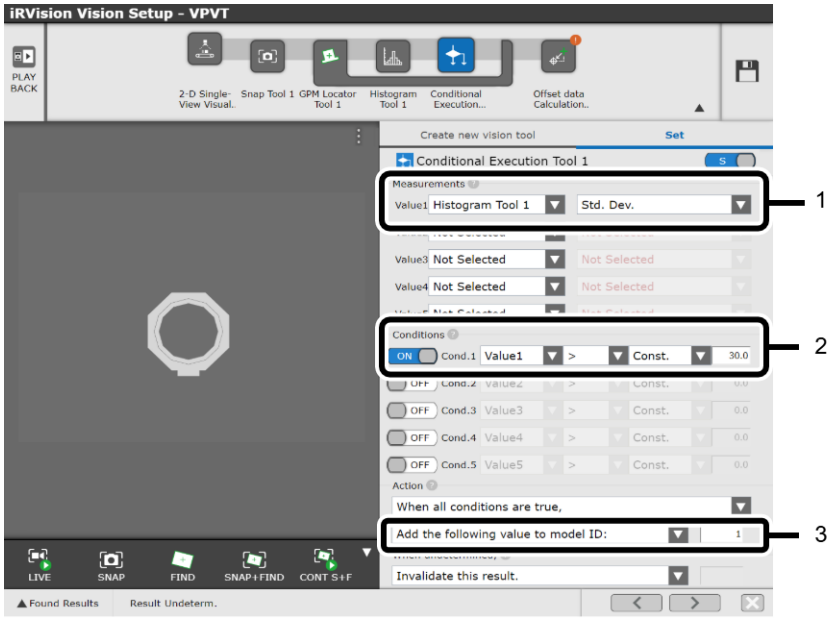
Set the conditional execution tool so that it sets the model ID to 2 if the part has a hole. See "I.7.1. Using different robots to pick parts depending on their positions on the conveyor(P.522)" for how to insert the conditional execution tool.

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 [Deviation] for [Value1].**

2 **Under [Conditions], check [Cond.1] and then select [Value1], [>], [Const.], and [30.0].**

3 **Under [Action], select [Add the following value to model ID] and specify 1.**

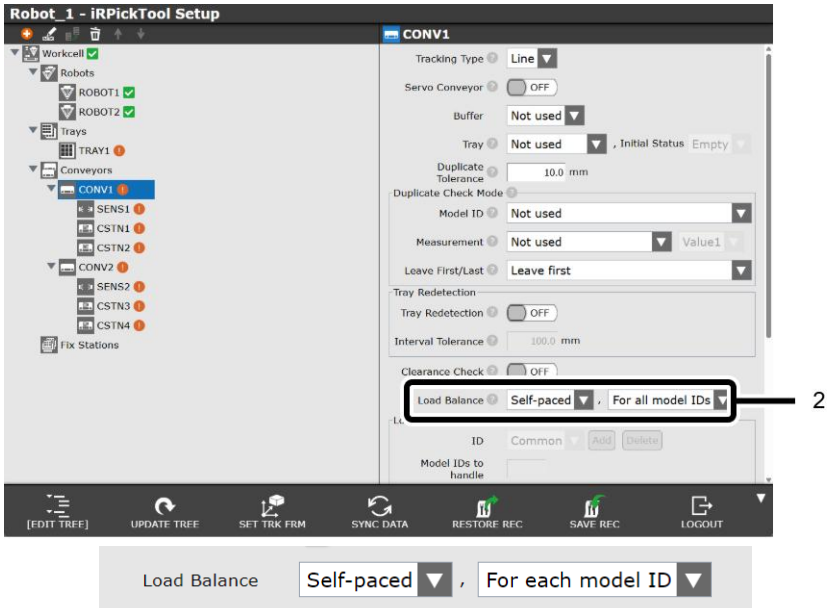


Setting the load balance

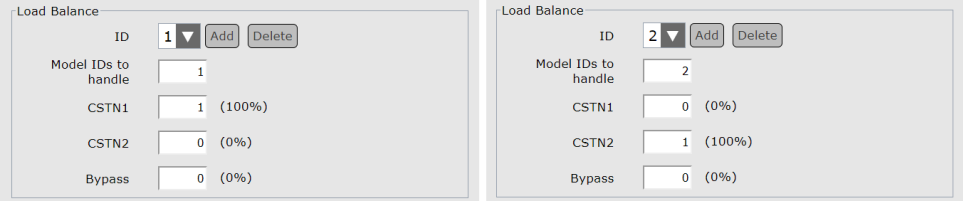
Set the load balance of the conveyor so that the upstream robot handles all parts whose model ID is 1 and the downstream robot handles all parts whose model ID is 2.

Open the conveyor setup screen, and set the load balance. Here, only the infeed conveyor is shown.

1. **Open the conveyor setup screen. After a while, the following screen appears.**
2. **Select [Self-paced] or [Keep rate] from the drop-down list on the left side of [Load Balance] and select [For each model ID] from the drop-down list on the right side.**



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.**



7.3. Stacking parts in multiple layers in a box

This refers to cases in which parts are stacked in multiple layers in a box.

This section describes the case in which parts are placed in two layers in a box where each layer contains four parts. While the example presented here uses a tray on the conveyor, the same procedure also applies when a fixed station is used.

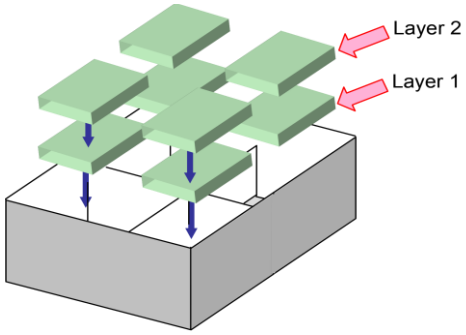


Fig. 7.3-1 Example of stacking parts in two layers in a box that can hold four parts in one layer

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

It is assumed that the conveyor configuration is defined as shown below. This is the same as Configuration 1 shown in "I.1.4. Sample system configurations(P.12)".

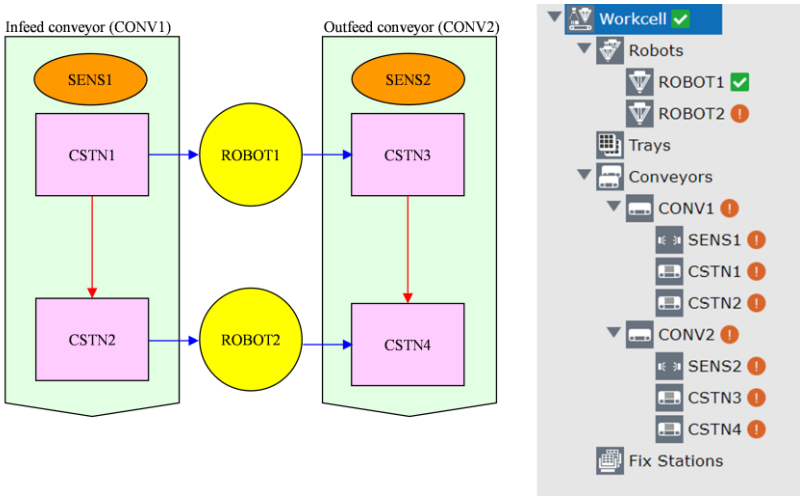
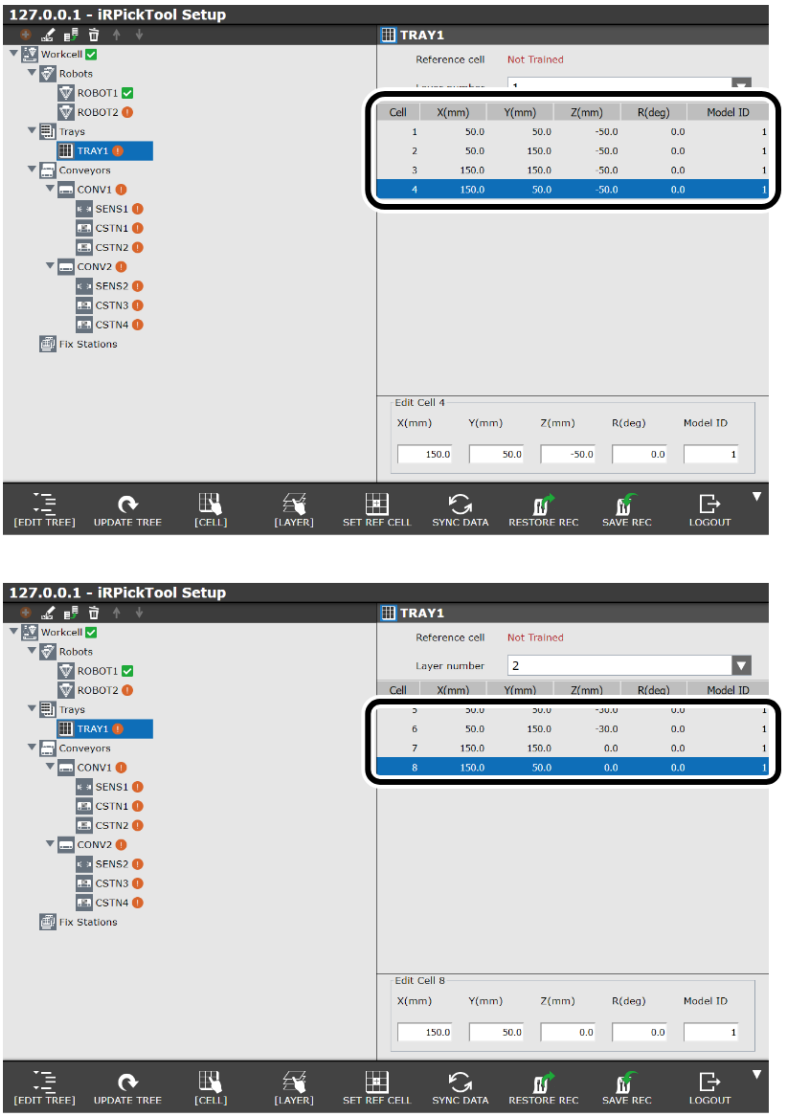


Fig. 7.3-2 Conveyor configuration example

Setting up a tray

Set the tray as shown below.

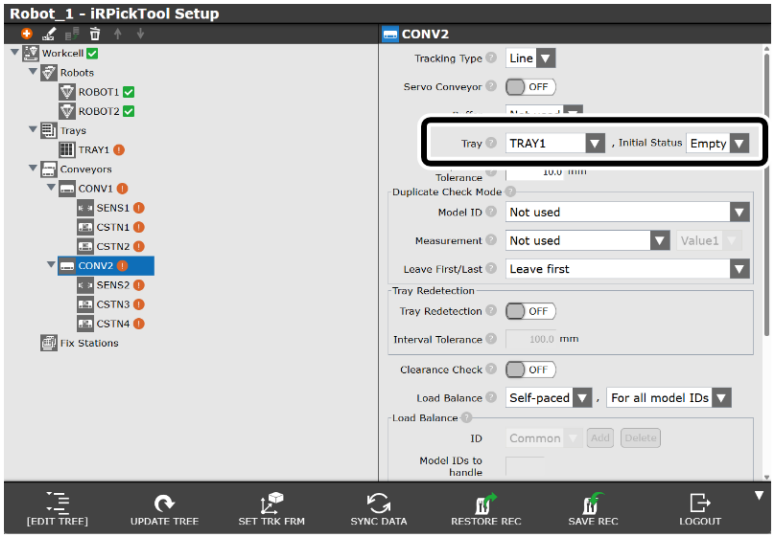


Cells 1 to 4 with layer number 1 belong to layer 1, and cells 5 to 8 with layer number 2 belong to layer 2. If you set the tray as shown above, the robot does not place a part in cells 5 to 8 of layer 2 until all the cells of layer 1 are filled.

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 up a 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.

7.4. Placing parts in sequence from the end of the box

This section describes the case in which parts are placed in sequence from the end of an unpartitioned box. While the example presented here uses a tray on the conveyor, the same procedure also applies when a fixed station is used.

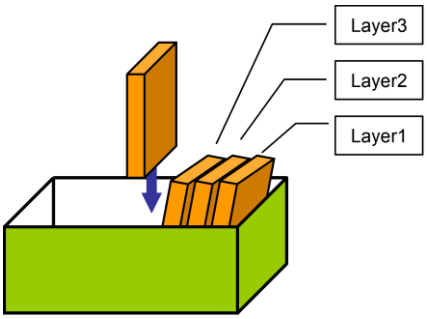
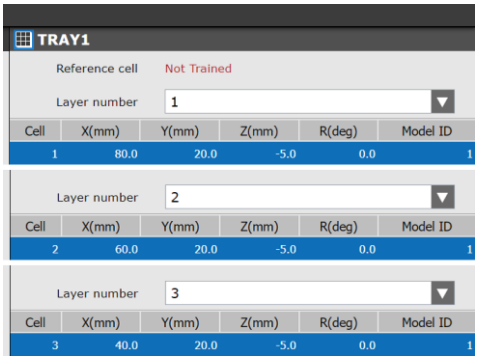


Fig. 7.4-1 Example of placing parts in a box in sequence from the end

In cases like this, set the tray using its “layer” function to prevent a part from being placed in the adjacent cell until the cell at the end is filled with a part.

Example of setting the tray

In the case mentioned above, each layer of the tray may be set as shown below.



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 "I.7.3. Stacking parts in multiple layers in a box(P.530)".

For details, see "I.7.3. Stacking parts in multiple layers in a box(P.530)".

7.5. Placing parts in different directions in a box

This section describes the case in which parts are placed in different directions in difference cells in a box.

In the example shown below, the direction of cells 2 and 4 is 90 degrees different from the other two cells.

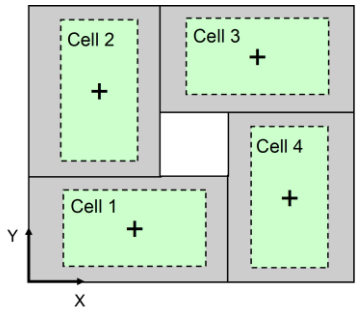
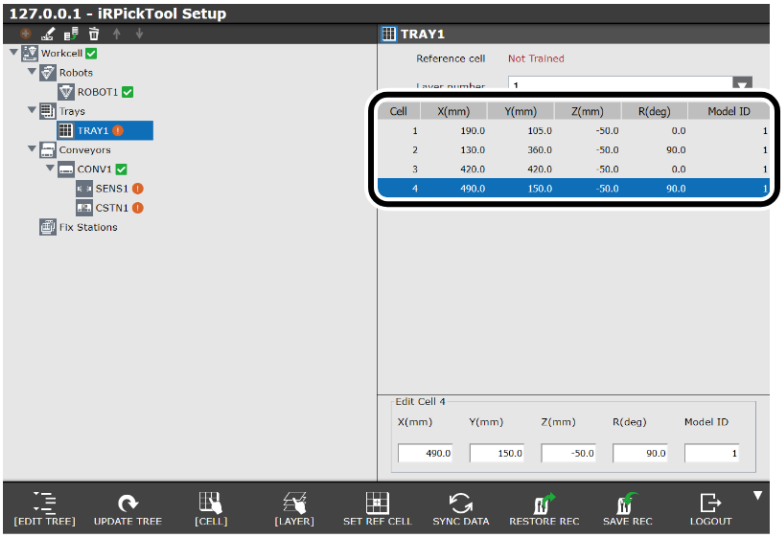


Fig. 7.5-1 Example of when the directions of the parts placed in a box are different depending on the cell

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

Setting up a tray

An example of setting the tray is given below.



The angle of cell 1 is 0 degrees, and the angle of the other cells 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.

7.6. Changing the placement position depending on the part type

This section describes the case of a system that places picked parts in a box in which parts of two models are placed in different predetermined places in the box depending on their model IDs. While the example presented here uses a tray on the conveyor, the same procedure also applies when a fixed station is used.

Consider a situation in which two part models (one large and the other small) are used with two types of boxes (one is the cell for storing the big part and the other is the cell for storing the small part), as in the example shown below. In this case, the cell in which a picked part is placed is determined by the model of that part.

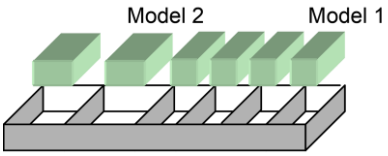


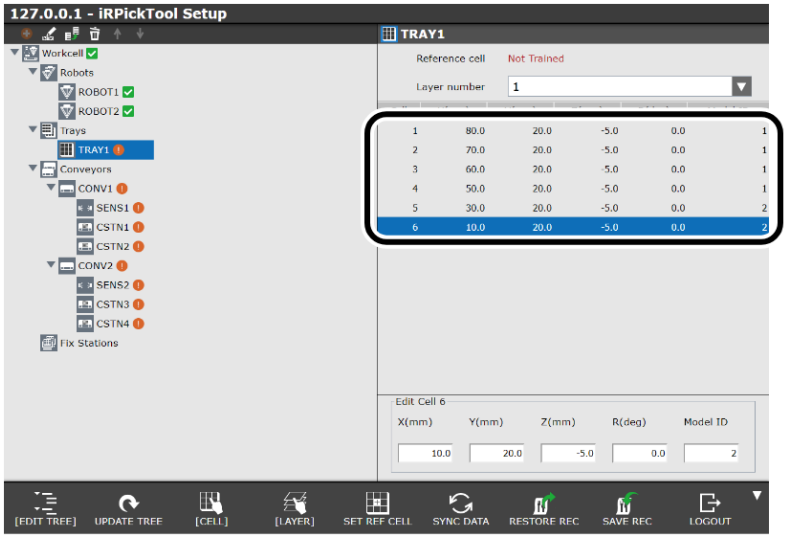
Fig. 7.6-1 Example of changing the placement position depending on the part type (large/small)

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

Here, it is assumed that you set the model ID of the small part to 1 and the model ID of the large one to 2 for the infeed conveyor.

Setting up a 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.

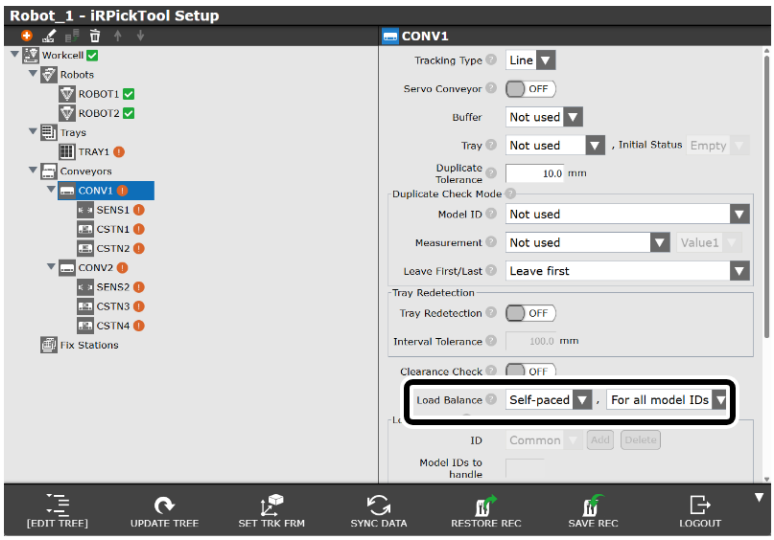
The model IDs of the cells are set to 1 and 2, as with the part model IDs for the infeed conveyor. Parts with model ID 1 are placed in the cells with model ID 1, and parts with model ID 2 are placed in the cells with model ID 2. The model IDs of cells do not necessarily need to match those of parts.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

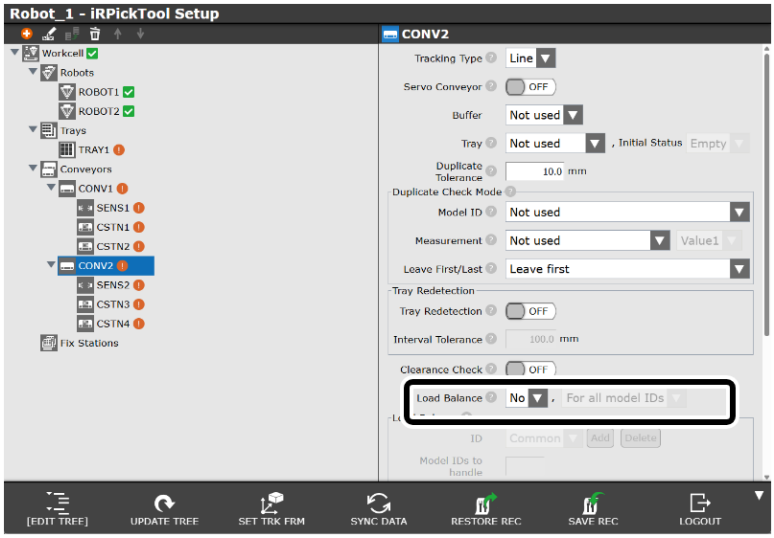
Setting up a conveyor

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 selection box next to [Load Balance] on the right, 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.

Examples of modifying the programs

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 “I.5.9.2.3. Placement program(P.373)” as follows. Major changes are underlined.

DROP1.TP

```
 1: UTOOL_NUM=1
 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 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.

7.7. Placing parts on different outfeed conveyors depending on the model ID

This section describes the case in which there is more than one outfeed conveyor and which outfeed conveyor to use is determined by the model of the part picked from the infeed conveyor.

Consider the case in which the system has one infeed conveyor and two outfeed conveyors, with each conveyor having one conveyor station, as shown in the figure below.

Parts of two models (model IDs 1 and 2) are conveyed from the infeed conveyor. Parts of model ID 1 are placed on outfeed conveyor 1, and those of model ID 2 are placed on outfeed conveyor 2.

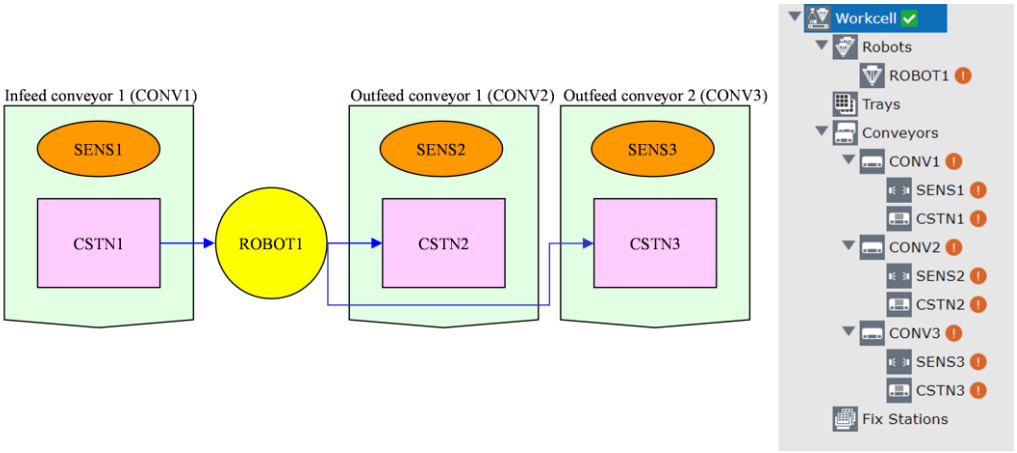


Fig. 7.7-1 Example of placing parts on different outfeed conveyors depending on the part type (large/small)

Setting up a 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 "I.5.9.2.1. Main program(P.369)" 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=1
 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 "I.5.9.2.3. Placement program(P.373)" 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.

7.8. 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 "I.1.4. Sample system configurations(P.12)".

Consider the case in which the system has two infeed conveyors and one outfeed conveyor, with each conveyor having one conveyor station, as shown in the figure below.

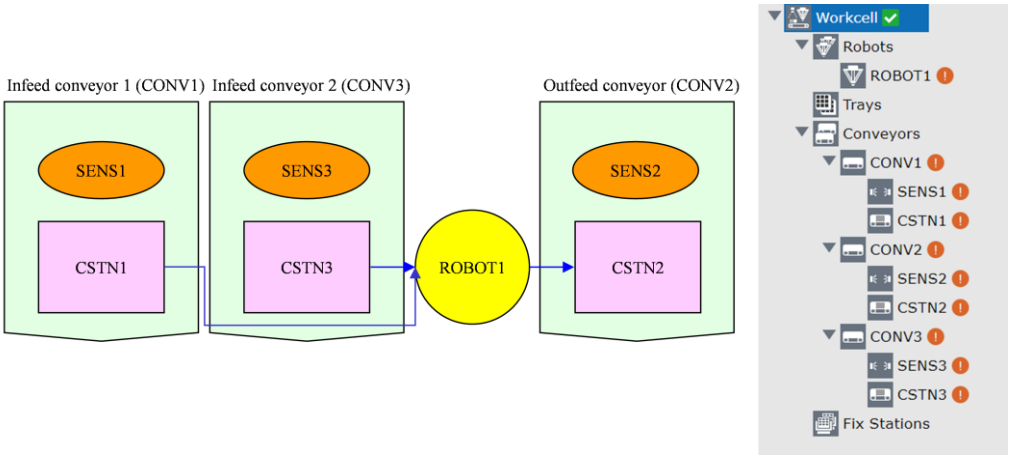


Fig. 7.8-1 Example of picking up parts from multiple infeed conveyors

It is assumed that the robot picks parts alternately from either conveyor. If the robot fails to pick a part from one conveyor, it attempts to pick one from the other.

Example of changing 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 changing the program MAIN1.TP described in "I.5.9.2.1. Main program(P.369)" and the program PICK1.TP described in "I.5.9.2.2. Pick program(P.370)" 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]=1,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 Stat]<>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:CSTN3],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:ZONE]+1]+20
 8: L PR[R[6:PR_NUM]] 4000mm/sec CNT100 VOFFSET,VR[3] Tool_Offset,PR[10]
 9: L PR[R[6:PR_NUM]] 4000mm/sec CNT3 VOFFSET,VR[3]
10: CALL VACUUM_ON
11: L PR[R[6:PR_NUM]] 4000mm/sec CNT100 VOFFSET,VR[3] 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.

7.9. 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 “I.5.9.2.2. Pick program(P.370)”.

An example of modifying this program is shown below. Major changes are underlined.

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 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],Delete)
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.

7.10. Picking a tray only when all cells are filled

This section describes the method to check whether all cells are filled with parts when the most downstream robot on a conveyor picks a tray.

Consider the case in which a tray has four cells and the most downstream robot picks the tray only when all its cells are filled with parts. It is assumed that all the most downstream robot does is to pick a tray.

Also, the most downstream robot is assumed to have a different hand from those of other robots.

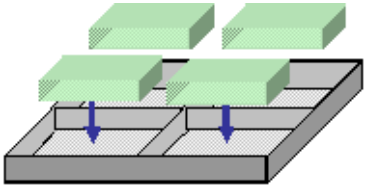


Fig. 7.10-1 Example of picking up a tray only when all the cells are full

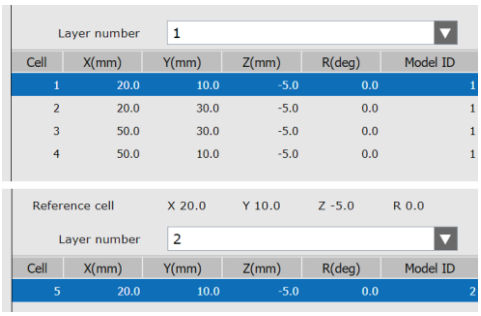
In such a case, it is necessary to set the tray and load balance differently.

In this particular case, define one extra cell for the tray. Add the last cell to the layer above that of the other cells so that the last cell is not allocated until all the other cells are filled.

Also, assign the last cell a different model ID from that of the other cells so that it is allocated only to the most downstream robot.

Setting up a tray

Set the tray as shown below.



The tray actually has only four cells, but define five cells. Set cell 5 as follows.

- Add it to the second layer.
- Assign a different model ID from that of the other cells.
- Make the position (X,Y,Z,R) the same as reference cell.

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

NOTE

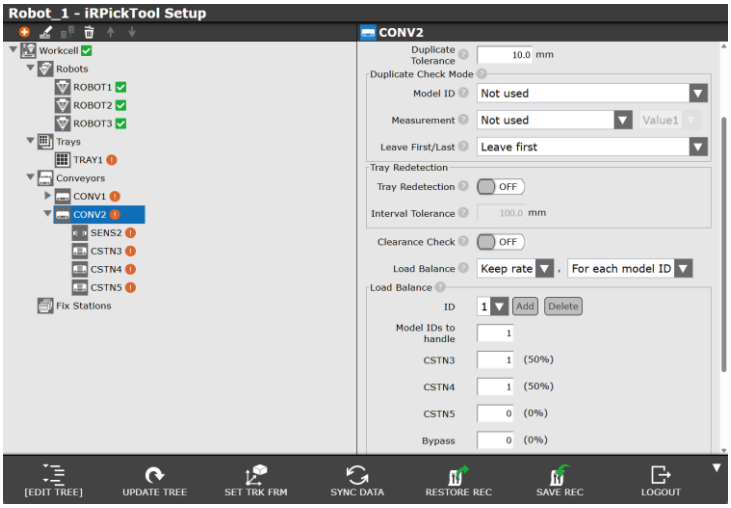
Normally, teach the robot position only for reference cell. The robot position for each cell is automatically calculated from the tray settings based on reference cell. However, since the operation for cell 5 (picking a tray) is different from the operation for the other cells (placing a part in a cell), the robot position needs to be able to be taught independently.

NOTE

In such a case, make the X, Y, Z, and R values of cell 5 the same as reference cell. This makes the position of cell 5 the same as reference cell, and the offset calculated based on cell 1 becomes 0. As a result, the robot position for cell 5 can be taught freely, as with reference cell.

Setting up a conveyor

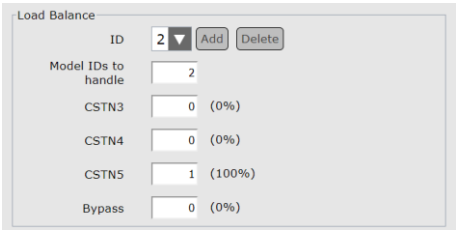
Set the load balance for the conveyor as follows.



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% is set for the first and second conveyor stations, and 0% is set for the third conveyor station).

On the other hand, since only those cells with model ID 2 need to be allocated to the last conveyor station, the load balance for model ID 2 is as shown below.



The load balance for model ID 2 in the last conveyor station is 100%. However, since [Empty] is selected for [Initial Status] for the tray, the cell in the second layer (cell 5) is not allocated until any of the cells in the first layer (cells 1 to 4) is empty. Therefore, the last robot can receive only the information of a tray where cells 1 to 4 are all filled.

Reference position setting and robot position teaching

When performing robot position teaching for the individual robots after setting the reference position on 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 the reference cell and cell 5 in the tray settings.

Changing the program

No program needs to be modified.

The most downstream robot gets the information of cell 5 using PKCSGETQUE, as the other robots do, and then picks a tray.

7.11. Handling parts detected before stop when restarting the system

This section describes the method to handle parts on the conveyor that have been detected before the system was stopped, but have not been processed when restarting the main program from the beginning after the system was stopped for an emergency or other reasons.

⚠CAUTION

When you turn the robot controller off and then on again to restart it, all part information is cleared. In this case, unprocessed parts on the conveyor detected before the system is stopped cannot be handled.

To avoid deleting information on parts that have been detected, call PKWCCHECK and PKSNSTART instead of calling PKWCSTART. Do not call PKCSCLRQUE. However, if the main program is restarted from the beginning when PKCSGETQUE has been executed before the system was stopped and the execution of PKCSACKQUE has not been completed, PKCSGETQUE is executed before PKCSACKQUE, which has not been executed, resulting in alarm IRPK-051 “No ACKQUE after the last GETQUE”. To avoid this alarm, execute PKCSCLRACK. If PKCSCLRACK is executed, the part information obtained by PKCSGETQUE before the system was stopped will be lost. If part information has been obtained by PKCSGETQUE, the system behavior will be the same as that when PKCSACKQUE is called by specifying “4:Remove”.

Example of changing the main program

An example of changing the program MAIN1.TP described in “I.5.9.2.1. Main program(P.369)” is shown below. The changed parts are underlined.

MAIN1.TP

```
 1: R[1:CYCLE_STOP]=0
 2: CALL PKCSGETID('CONV1',CStn ID Reg=11)
 3: CALL PKCSGETID('CONV2',CStn ID Reg=12)
 4:
 5: UTOOL_NUM=1
 6: UFRAME_NUM=0
 7: J P[1:Perch Pos] 50% FINE
 8:
 9: CALL PKWCCHECK
10: CALL PKCSCLRACK(CStn ID=R[11:CSTN1])
11: CALL PKCSCLRACK(CStn ID=R[12:CSTN2])
12: CALL PKSNSTART
13:
14: LBL[100]
15: CALL PICK1
16: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
17: CALL DROP1
18: IF R[1:CYCLE_STOP]=1,JMP LBL[900]
19: JMP LBL[100]
20:
21: LBL[900]
22: CALL PKWCEND
[End]
```

8. Other useful 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.

8.1. Exchange workcell with using recipe

This section describes the setup of the recipes.

When a recipe is used, one recipe (called the “active recipe”) is selected, and the workcell corresponding to its content will be executed.

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. Up to 99 recipes can be created.

Recipes have the following features:

1. Recipes may be changed by a PLC.
2. Recipes may be exported to an SD card or a USB device.
3. Recipes may be imported from an SD card or a USB device.

All the recipes are always stored in only one controller in the entire workcell to minimize recipe management and avoid confusion. The controller where all the recipes are stored is called the “Recipe Manager”.

Saving recipes can be done from the teach pendant.

On the [File] screen of the teach pendant, if you select [All] and save, the settings that are common to the recipes will be backed up to the IRPICKTOOL_CFG.ZIP file, and each recipe will be backed up to an IRPK_RECIPEx.ZIP (x is the recipe number) file.

The IRPICKTOOL_CFG.ZIP and IRPK_RECIPEx.ZIP files can also be loaded onto the robot controller. Its extracted contents will go in the FR: \IRPICKTOOL\ directory.

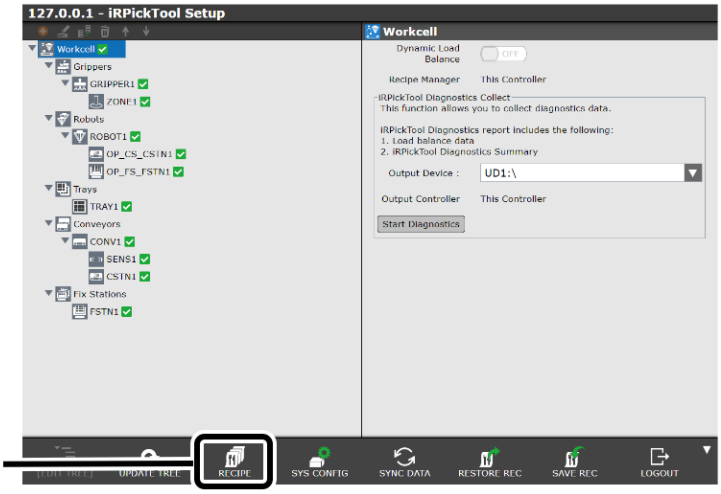
8.1.1. Setting a recipe

Recipe setup is performed in the Workcell property page. For details on workcells, see “I.5.1. Setting up a workcell(P.266)” or “I.6.1. Workcell setup screen(P.422)”.

The Workcell Page contains a function key called RECIPE. It also contains a property called the “Recipe Manager”.

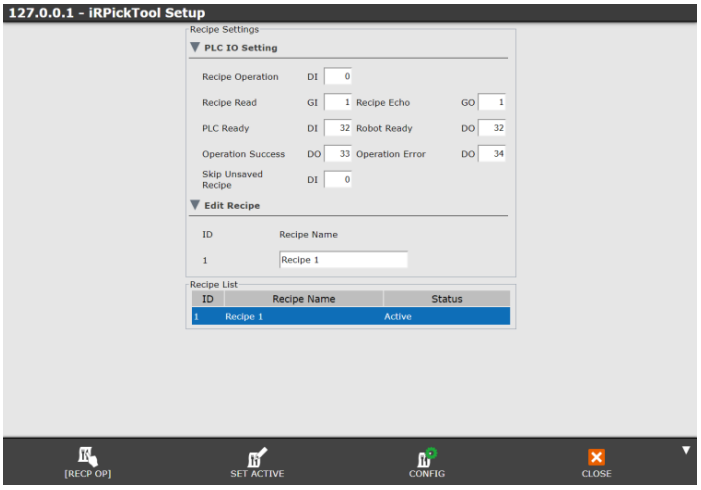
The Recipe Manager refers to the controller in which the recipes are stored. When you have multiple robot controllers in a workcell, the recipes are stored in the ring leader robot controller within the RIPE robot ring network by default. For details on robot rings, refer to “I.4.1.4. Setting up the robot ring(P.246)”.

1 On the [Workcell] settings screen, click [RECIPE].



2 The recipe settings screen will appear.

If there is already a recipe, the recipe name will be displayed under [Recipe List].



For details on how to use I/O in [I/O settings], refer to "I.8.1.3. Recipe operation by means of a PLC(P.557)".

[Recipe Operation DI]

When this DI is turned ON by external PLC, recipe will be operated.

[Recipe ID]

Set the GI number used to communicate function codes during recipe operations, as well as the recipe ID and string register number for creating, deleting, or switching recipes, from the PLC to the robot controller.

[Echo]

The value of the GI number set in [Recipe ID] is echoed to the GO with the number specified here.

[PLC ready]

Set the number of the DI that receives the PLC ready signal before reading the GI specified in [Recipe ID] and before continuing the operation afterward.

[Robot controller ready]

Set the number of the DO that sends the robot controller ready signal to the PLC.

[Recipe Operation Succeeded]

If the recipe operation succeeds, this DO turns ON.

[Recipe Fault DO]

If the recipe operation in the PLC fails, this DO turns ON.

[Skip Unsaved Recipe]

When the DI set here is ON, any unsaved recipe settings will be discarded during recipe switching via the PLC. When the DI is OFF, recipes will be automatically saved before recipe switching.

If the DI number is set to 0 and a recipe switch is performed with unsaved settings, error IRPK-277 will occur and the recipe switching process will stop.

[Recipe Name]

The current recipe name is displayed, and the recipe name can be directly edited. See “Renaming a recipe(P.548)” for details.

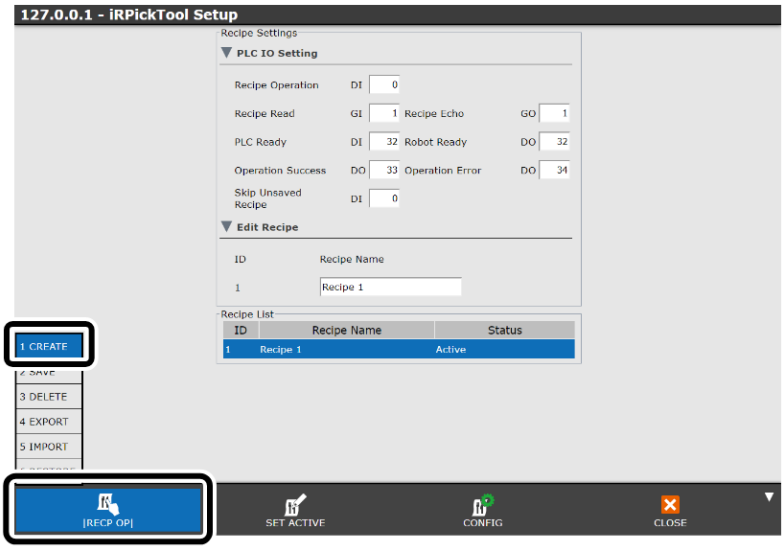
8.1.2. Detail setting of a recipe

Set a recipe.

Creating a new recipe

When you create a recipe you save the current 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. This XML file contains the tracking frames and user frames for fixed stations. In addition, when using Plug & Play, it also contains the taught robot positions required by iRPickTool, such as the perch position for all groups, reference position, and approach position.

1 Click [RECP OP] on the recipe settings screen and select [CREATE] from the menu that appears.

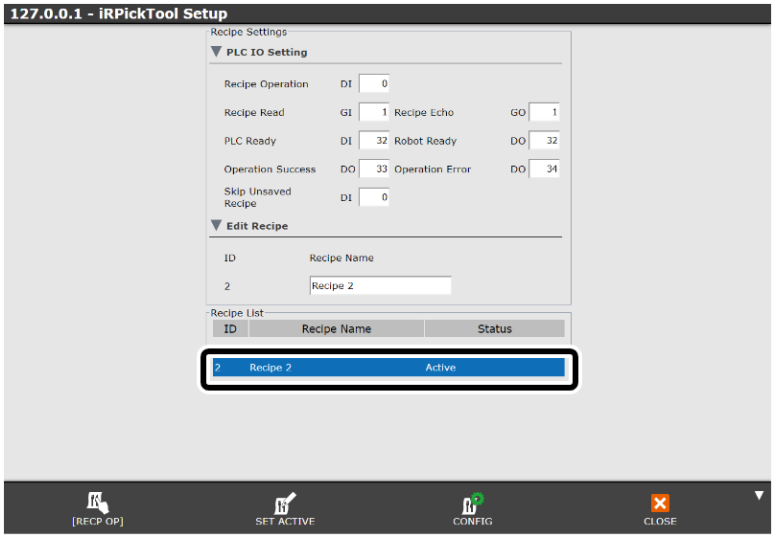


A new recipe with a default name of RecipeX where x is the recipe id is immediately created.

While a recipe is being created, “Now Creating Recipe” will be displayed on the screen.

When the recipe is successfully created, you will see a confirmation prompt (not shown). The newly created recipe contains all the current iRPickTool data you have setup using the tree view.

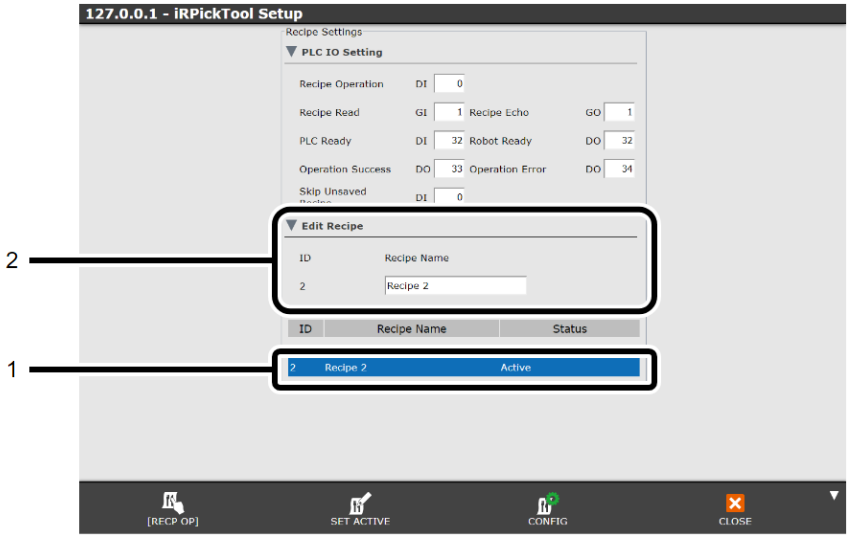
The created recipe will be added to [Recipe List] as shown on the screen below.



Renaming a Recipe

Rename a Recipe.

1. **From [Recipe List], select the recipe that you want to rename.**
2. **The current name of the recipe will be displayed under [Edit Recipe] at the bottom, so edit it directly.**



3. **Select the text box that contains the current name, and change the name.**
4. **If you press the 'ENTER' key on the teach pendant, the recipe name will be changed and the new name for the recipe will be displayed in [Recipe List].**

   Once you change the name, you should save the Recipe. See the “Saving a Recipe” section below.

   A recipe name can consist of up to 20 one-byte characters.

   You can include spaces in the recipe name. For example, “iRPickTool Recipe 2.” The name must not contain a “&”, “ ' ”, “ " ”, “/”, or “\.”

Deleting a recipe

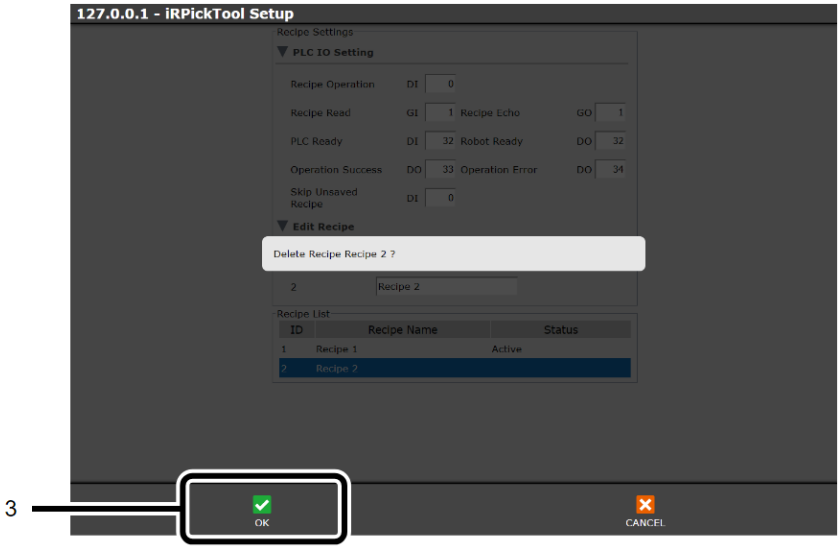
Delete a recipe.

When you delete the recipe, all the data associated with the recipe in XML files will be deleted from the FR:\IRPICKTOOL\IRPK_RECIPEx directory, where x is the Recipe ID that you see in the Recipe List window. The name of the relevant recipe will also be deleted from the Recipe List Window.

1 **Select the recipe that you want to delete from [Recipe List].**
The selected recipe will be highlighted.

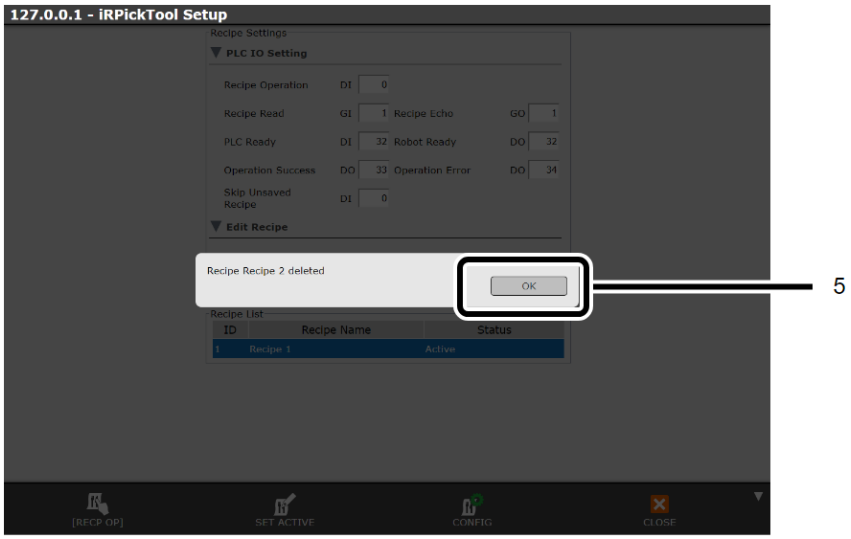
2 **Click [RECP OP] and select [DELETE] from the menu that appears.**
You will get a confirmation prompt as shown below.

3 **Click [OK].**



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 shown in the figure below.

5 **Click [OK].**



Saving a recipe

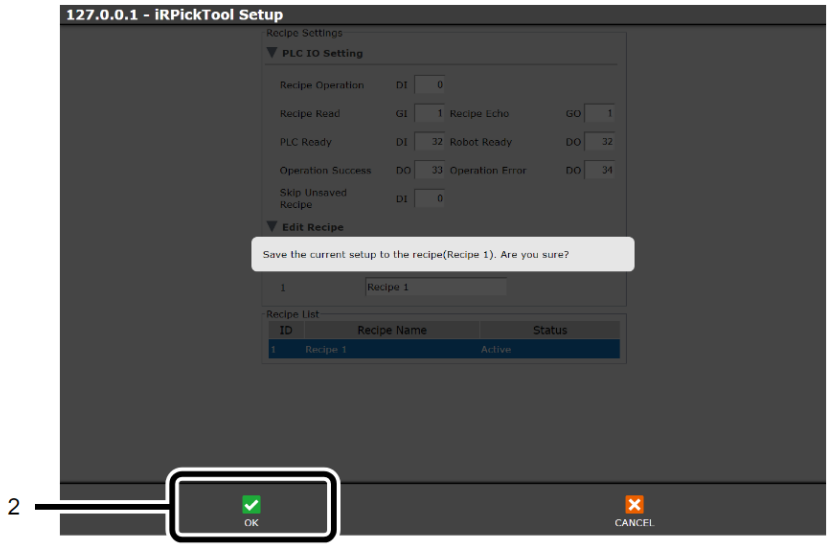
Save a recipe.

When you save a recipe, you save all the current iRPickTool data to the Active Recipe. [Status] for active recipes is [Active] in [Recipe List].

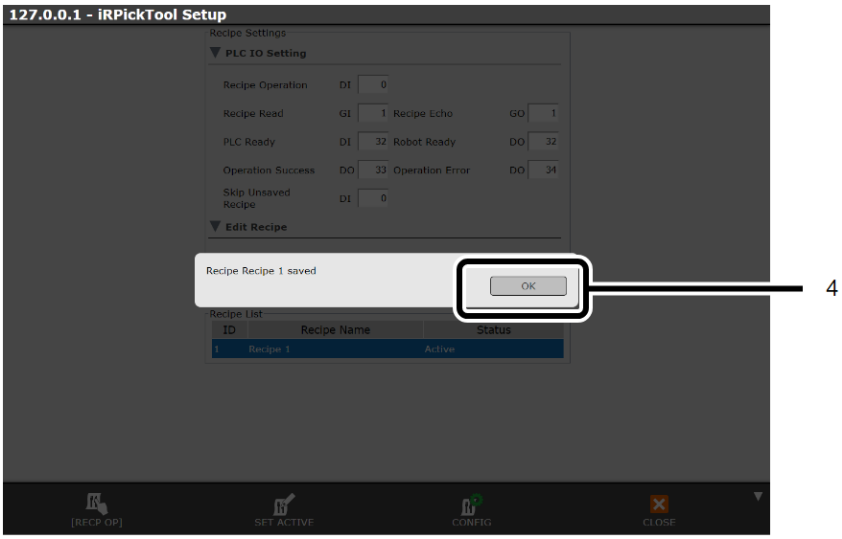
Save when you have made some changes to the iRPickTool data since you created the recipe or since you last saved the recipe.

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, note that the entire set of iRPickTool data will be completely saved.

1. **Click [RECP OP] and select [SAVE] from the menu that appears.**
   You will be asked for confirmation as shown in the figure below.
2. **Click [OK].**



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 shown in the figure below.
4. **Click [OK].**



Restoring a recipe

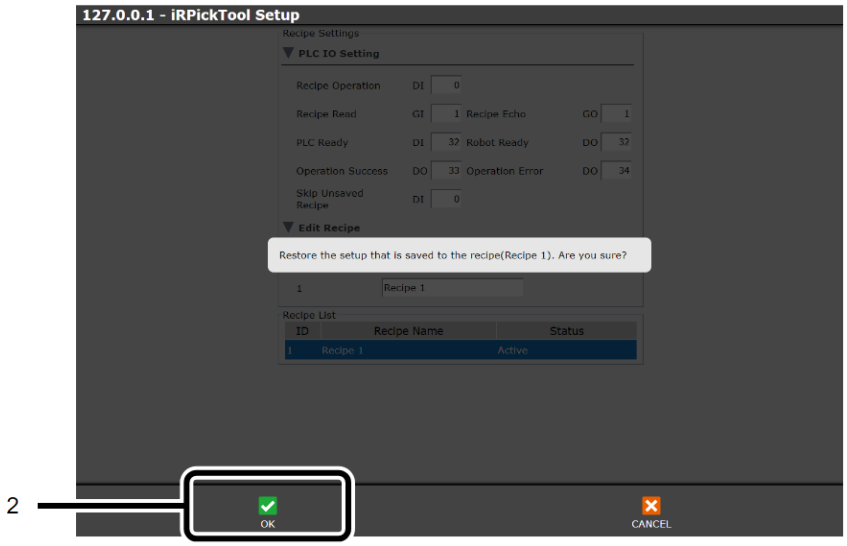
Restore a recipe.

When you restore a recipe, the last saved Active Recipe data will be loaded into the memory. The Restore operation is useful when you have made changes to the iRPickTool data since you last saved the recipe but wish to discard those changes and revert to the last saved version. It functions as a kind of “Exit Without Saving” 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, note that the entire set of iRPickTool data will be completely restored.

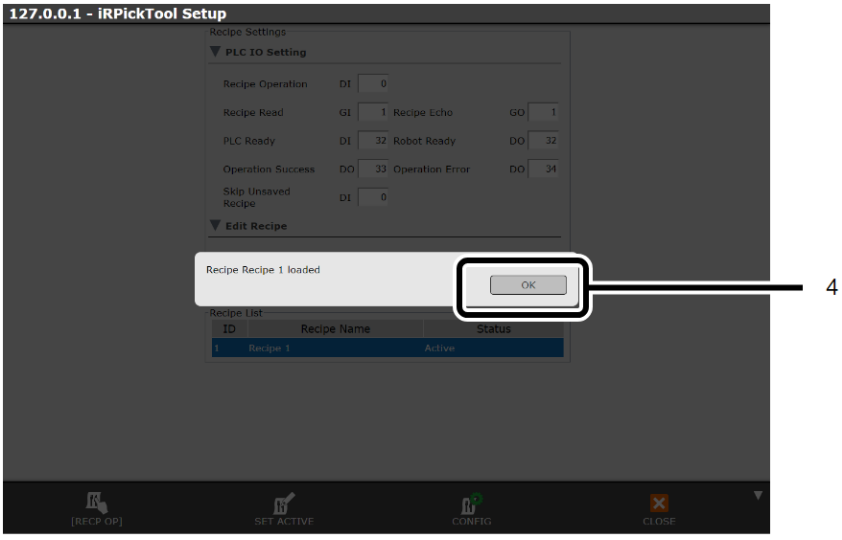
1 **Click [RECP OP] and select [RESTORE] from the menu that appears.**
You will be asked for confirmation as shown in the figure below.

2 **Click [OK].**



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 shown in the figure below.

4 **Click [OK].**



Exporting recipes

Export a recipe.

When you export recipes, the recipes stored in the FR:\IRPICKTOOL\ directory in the controller will be copied to an external storage device such as an SD card or USB stick. You can change the default device on the [File] screen. Find the [UTIL] function key and press it to display the pull-up. Select “Set Device” and then choose your default device such as “MC” or “UD1” or “UT1”. Once you have chosen the device, do the following:

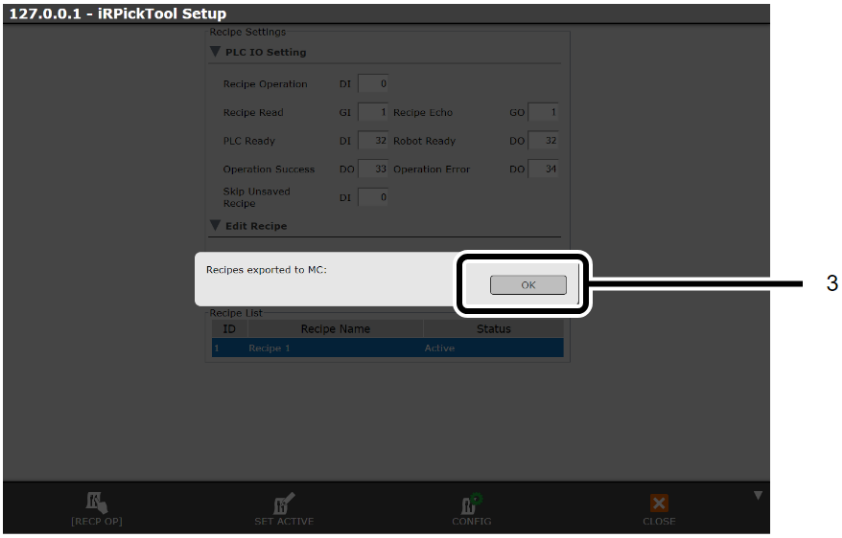
1 **Click [RECP OP] and select [EXPORT] from the menu that appears.**
You will be asked for confirmation as shown in the figure below.

2 **Click [OK].**



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 shown in the figure below.

3 **Click [OK].**



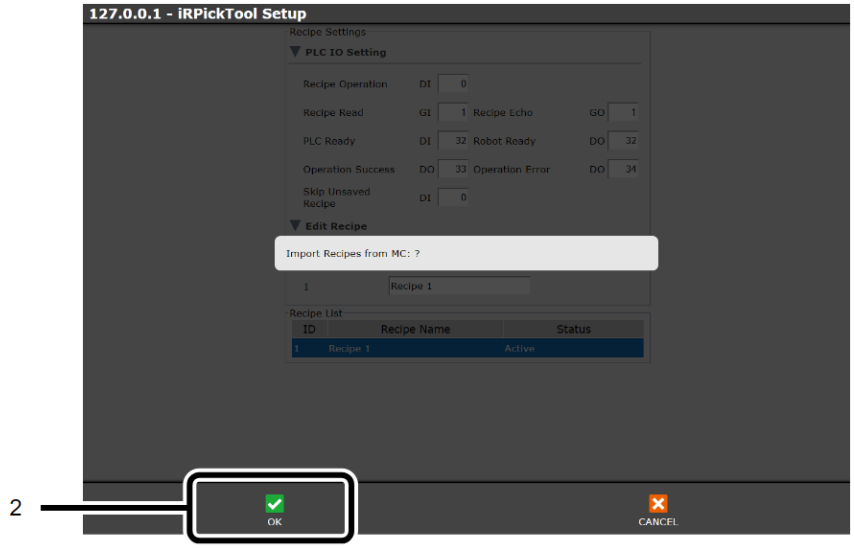
Importing recipes

Import a recipe.

When you import recipes, the recipes stored in an external storage device such as an SD card or USB stick will be copied to the FR:\IRPICKTOOL\ directory of the controller. You can change the default device on the [File] screen. Find the [UTIL] function key and press it to display the pull-up. Select “Set Device” and then choose your default device such as “MC” or “UD1” or “UT1”. Once you have chosen the device, do the following:

1 **Click [RECP OP] and select [IMPORT] from the menu that appears.**
You will be asked for confirmation as shown in the figure below.

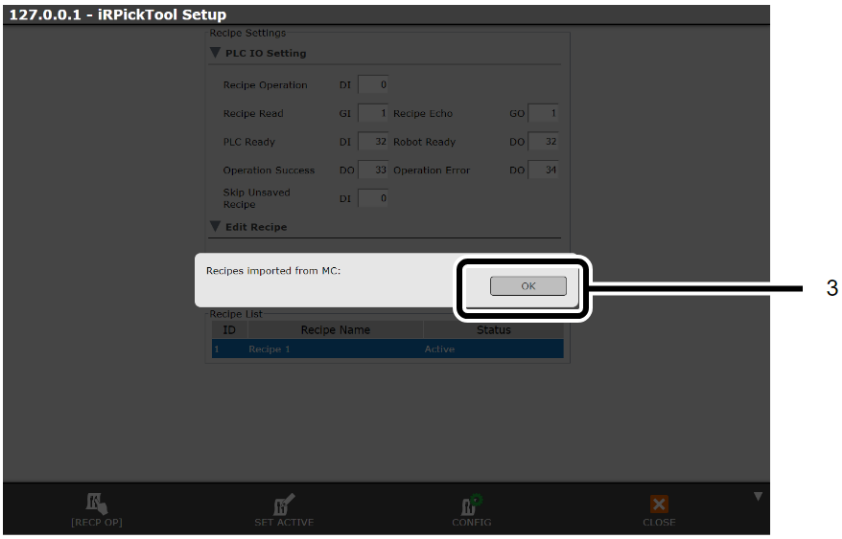
2 **Click [OK].**



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 shown in the figure below.

3 Click [OK].



Switching recipes

Switch to a different recipe from the current recipe.

Recipe switching is performed when you want to change from producing the current product to a different one using previously saved data. For example, switching from workpiece type A to workpiece type B.

1 Click the recipe you want to set as active in the [Recipe List] window.

The recipe row will be highlighted.

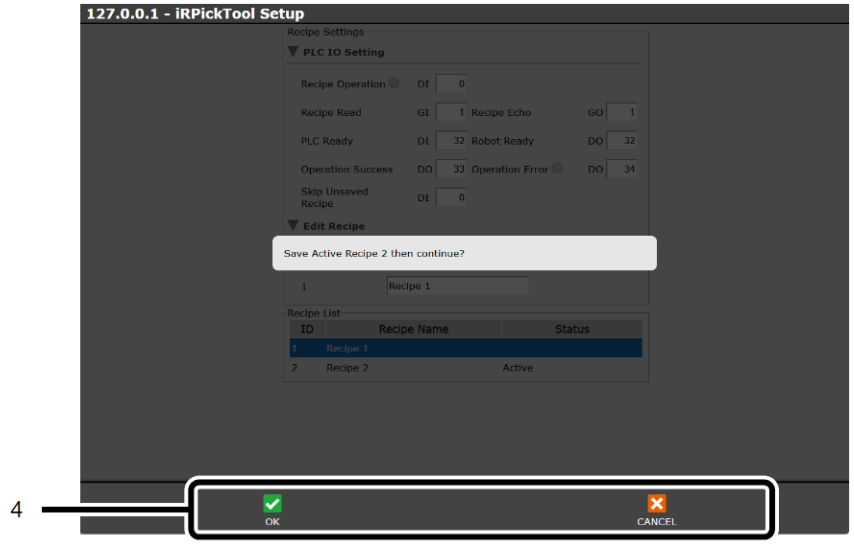
2 Click [SET ACTIVE].

You will get a confirmation prompt as shown below.

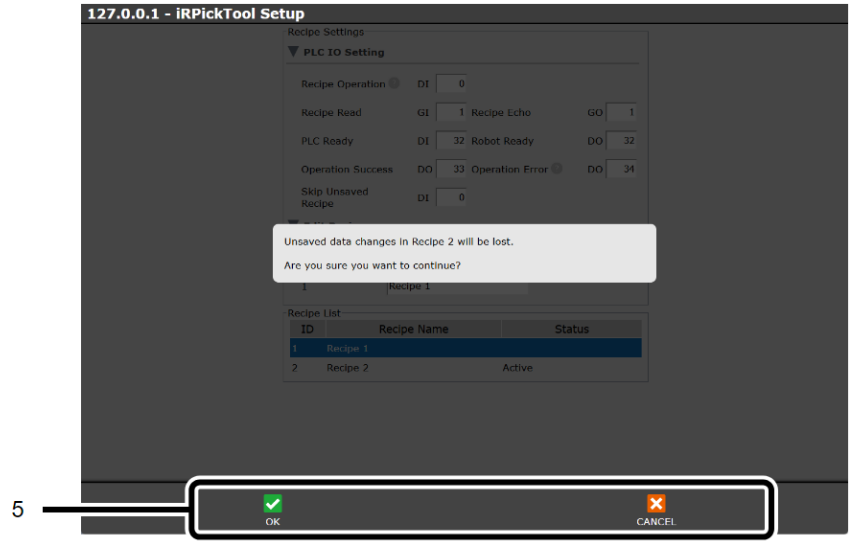
3 Click [OK].



4 **If there are unsaved settings, a message will appear asking whether you want to save the active recipe. Click [OK] to save the recipe and switch it, or click [Cancel] to continue switching without saving the recipe.**



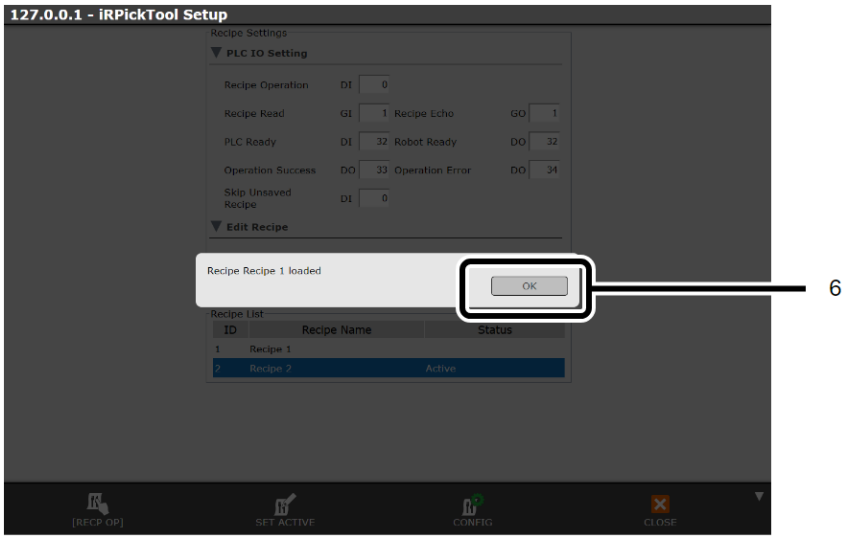
5 **If you click [Cancel], a final confirmation message will appear asking whether to skip saving the settings before switching the recipe. Click [OK] to start switching the recipe, or [Cancel] to abort the operation.**



6 **If the recipe switch is successful, a message like the one shown in the figure below will be displayed.**

If switching to the new recipe takes some time, the message “Now Switching Recipe” will be displayed before the recipe switch is successfully completed.

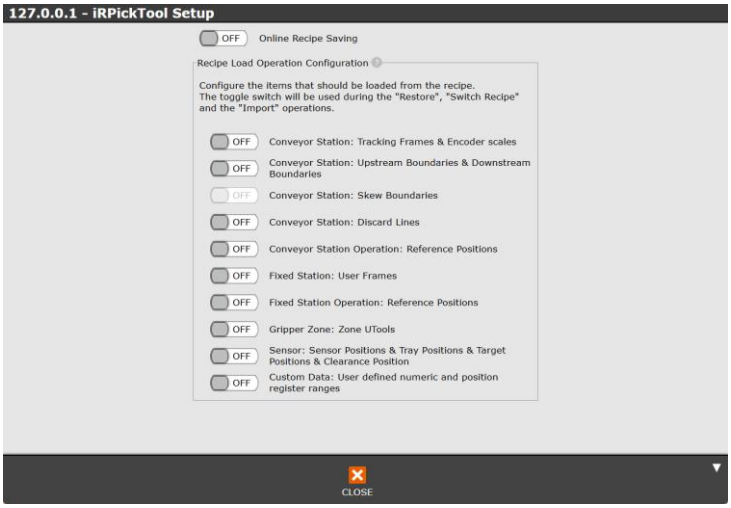
Click [OK].



Configuring Recipe Operations

You can configure a recipe operation using this function which is tied to the [CONFIG] key on the main Recipe menu.

At this time, you can only configure the “Load” function of the recipe which applies to the Restore, Switch Recipe and Import operations.



Editing recipe online

By enabling the [Online Recipe Saving] toggle switch at the top of the page, the active recipe can be stored while the robot is working on a production line. With this function, the parameter adjusted during the production process can be stored in the recipe file and can be used the next time this recipe becomes active. Storing recipe during the production process may take some time than normal recipe storing, still it has no impact on the robot cycle time. Since this screen cannot be accessed during production process, configuration changes should be made during the production is stopped.

Setting of restoring a recipe

Typically, when you switch to a new recipe, or restore 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 retaught the track frames for a conveyor or you retaught the Reference positions to be more accurate in the active recipe. Usually, 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.

Items for which it can be selected whether to use the current settings or to use the stored settings for the recipes to switch/restore are listed on the [Recipe Load Operation Configuration] screen.

Items set to [OFF] will maintain the currently used settings when switching recipes/restoring a recipe. Items set to [ON] use the settings saved in the recipe to be switched/restored.

If you do not want track frames to be loaded, then set the [Conveyor Station: Tracking Frames & Encoder scales] toggle switch to [OFF]. If there are other items in the list that you do not wish to load, then set those items to [OFF].

Just to be safe, we recommend saving the recipe to be used with the latest settings when you want to change the [Recipe Load Operation Configuration] toggle switch setting. As for the changed items, the current active settings will be discarded the next time the recipe is switched or restored.

When changing the settings, do so after storing the recipe by following the procedures.

1. **Perform a switching recipe on the Recipe list screen and switch to a different recipe.**
2. **Confirm that switching the recipe successfully completed except for the data of the items to be restored set to [ON] on [Recipe Load Operation Configuration].**

   NOTE

   The items with [Recipe Load Operation Configuration] set to [OFF] will not be restored from the switched recipe and will inherit the previous settings. Check if the inherit settings are appropriate for the switched recipe, and change the settings as needed.
   This is especially important for the items to be restored that you want to set to [ON] because the setting to restore each recipe setting is not enabled in step 2.
3. **Perform “Saving a Recipe” process.**

   The latest settings are stored on the recipe.
4. **Repeat steps 1 to 3 on all of the other recipes.**
5. **After completing all recipe settings, return to the [Recipe Load Operation Configuration] screen and set the items you want to restore in all the subsequent recipes to [ON].**

   After setting the rule for each item, click [CLOSE] to close the setting screen.

   NOTE

   - You can only control the loading of the items in the recipe that are listed in the Recipe Load Operation Configuration menu.
   - When saving a recipe, all the settings are saved. It is not possible to save only certain items.
   - If you always want the listed categories of data be exactly the same in all the recipes, it may be better to set all the items to [OFF].
   - By default, all the items are initially set to [OFF] and the listed categories of the recipe data are not loaded. In the case of the settings for North America, all the items displayed on this screen other than [Conveyor Station: Skew Boundary] are set to [ON] by default.

[Skew Boundary]

For [Skew Boundary], recipes will not be restored unless the [Conveyor Station: Skew Boundary] toggle switch is enabled, either in the standard settings or the North America settings.

When the [Conveyor Station: Upstream Boundaries & Downstream Boundaries] toggle switch is enabled and Skew Boundary (refer to “I.5.6. Setting up a conveyor station(P.318)”) is also enabled, the [Conveyor Station: Skew Boundary] toggle switch is enabled to allow the angle of the skewed boundary to be loaded during recipe operation.

The [Conveyor Station: Skew Boundary] toggle switch can be operated only when the [Conveyor Station: Upstream Boundaries & Downstream Boundaries] toggle switch is enabled. Additionally, when the [Conveyor Station: Upstream Boundaries & Downstream Boundaries] toggle switch is disabled, the [Conveyor Station: Skew Boundary] toggle switch will be automatically disabled.

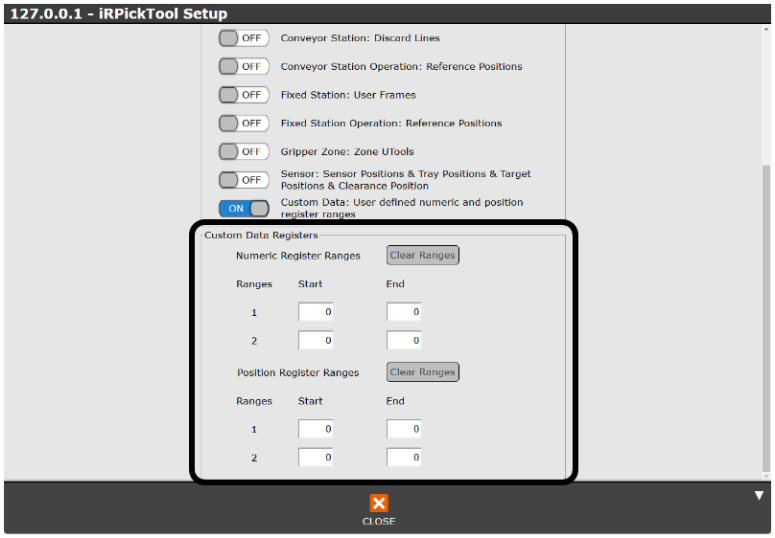
Custom Data Registers

Specify the ranges of the numeric registers and Position Registers.

This setting screen becomes enabled when the “Custom Data: User defined numeric and position register ranges” toggle switch in [Recipe Load Operation Configuration] is enabled. Enter numbers in the [Start] and [End] boxes to specify the ranges. You can specify up to two ranges. To cancel a range setting, enter 0 in both the [Start] and [End] boxes or click the [Clear] button.

NOTE

The specified ranges apply to all the robot controllers and robot groups. Registers in the ranges are saved in the recipe.
Use this setting for iRPickTool Basic (J944) or when customizing a standard TP program for iRPickTool (J945) and when numeric values and positions must be saved.
Switching between recipes loads data from registers that contain different values in the same range.



After specifying ranges, confirm that the register ranges have been correctly set before clicking the [Close] key to save the settings and returning to the main recipe page.

⚠CAUTION

If a warning appears, do not close the setting page and return to the main recipe page without solving it. An unclear error message may appear later during recipe operations. Clear the warning according to the following tips:

- Negative values cannot be set for ranges.
- The start number must not be larger than the end number.
- Settings where Range 1 and Range 2 overlap are not allowed.
- When iRPickTool (J945) is installed, 81 to 240 and 41 to 100 are reserved for the numeric register range and Position Register range, respectively.
  The entire register range can be used in iRPickTool Basic (J944).
- The specified Position Register Ranges must not contain any uninitialized position registers. All Position Registers within the range must be configured.
- A Position Register cannot be set as the position of each axis.

8.1.3. Recipe operation by means of a PLC

You can operate recipes from a PLC. This operation is performed using the following procedure.

Recipe operation by means of a PLC can be performed only with a recipe manager controller.

1. **On the [Recipe] settings screen for a robot controller that is a recipe manager, set the DI number for [Recipe Operation DI].**
2. **From the PLC, turn ON the DI that you have set in step 1.**
3. **When the DI is turned ON, it will be detected by the robot controllers and the program to perform a handshake with the PLC will be executed.**

   As a standard program for this purpose, PK_PLC_RECIPE_OPERATION.TP has been prepared.
4. **If you use a KAREL program, you can operate recipes. The arguments for KAREL programs are provided by group input signals in standard programs.**
5. **In the event of an error during recipe operation, the DO that has been set for [Recipe Fault DO] will turn ON.**

   If recipe operation is already being executed, recipe operation by means of a PLC is not allowed. For example, if [Recipe Operation DI] is turned ON while executing a recipe operation on the [Recipe] settings screen, the recipe operation will not be performed, so [Recipe Fault DO] will turn ON.

NOTE

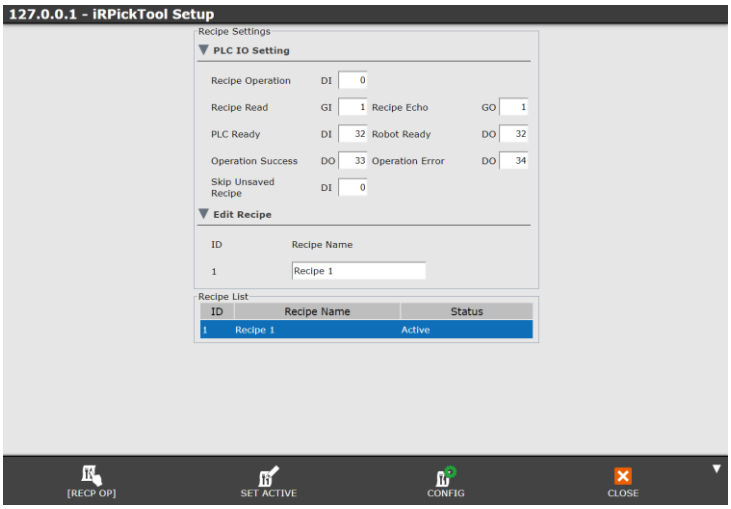
It sometime takes 5 and more seconds to complete this recipe operation by means of a PLC.

8.1.3.1. Settings required for recipe operation by means of a PLC

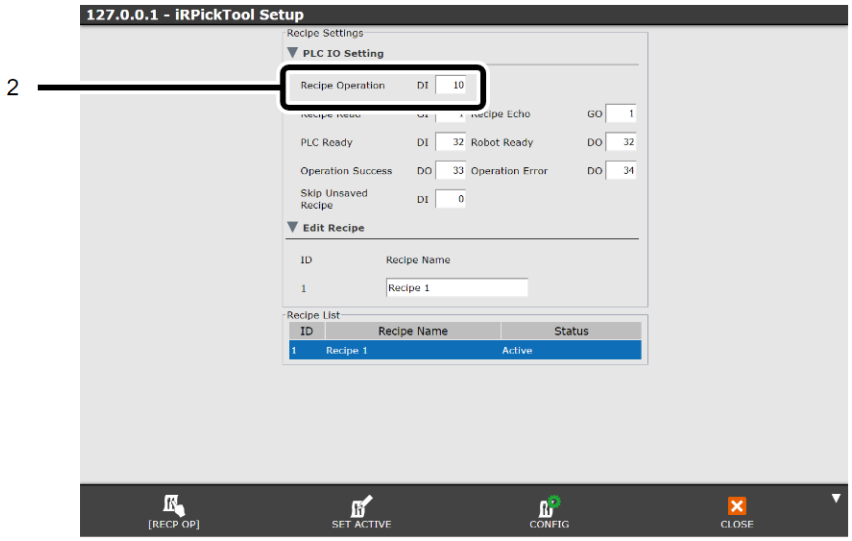
Set up settings that are required for recipe operation by PLC.

Settings required for recipe operation by means of a PLC

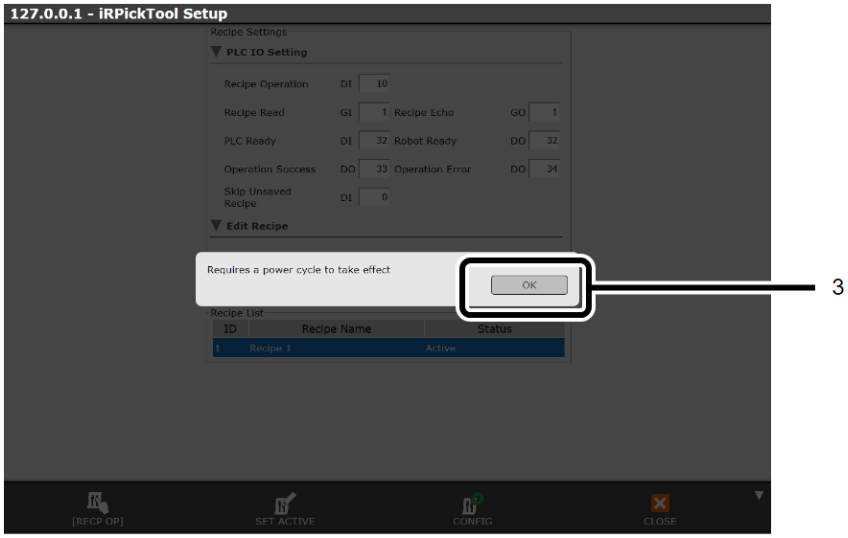
1 **Display the Recipe menu by clicking the [RECIPE] key in the Workcell menu for a robot controller that is a recipe manager.**



2 **In the Recipe Settings window, enter a valid Digital Input for the property “Recipe Change DI” as shown below:**



3 **The following confirmation message will appear, so click [OK].**



4 **Change other I/O setting items.**

When robots can receive data, the DO with the number corresponding to [Robot Controller Ready] will turn ON. When the [Robot Controller Ready] DO turns ON, the PLC sets the GI number corresponding to the [Recipe ID].

When the PLC sets data to the GI with the number corresponding to the [Recipe ID], it turns ON the DI set for [PLC Ready] on the PLC side, and the robot starts reading the [Recipe ID] and returns the same data to the GO with the number corresponding to [Echo]. The DO for [Robot Controller Ready] also turns OFF.

When the [Robot Controller Ready] DO turns OFF and the GO for [Echo] is confirmed, the PLC turns OFF the DI for [PLC Ready].

To automatically discard or save unsaved recipe settings during recipe operations, change the DI with the number corresponding to [Skip Unsaved Recipe].

When the DI with the number corresponding to [Skip Unsaved Recipe] is ON, unsaved changes are discarded when switching recipes. When it is OFF, unsaved changes are automatically saved before switching recipes. If this DI number is set to 0, the IRPK-277 alarm will occur if there are unsaved settings in the recipe and the recipe switching process will stop.

If it is necessary to check whether an error has occurred in the PLC, change the settings of [Recipe Operation Succeeded] and [Recipe Fault DO].

By default, [Robot Controller Ready] is assigned to DO[32], [PLC Ready] to DI[32], [Recipe ID] to GI[1], [Echo] to GO[1], [Recipe Operation Succeeded] to DO[33], and [Recipe Fault DO] to DO[34]. [Skip Unsaved Recipe] is disabled (DI number is set to 0). All the numbers in the I/O settings can be modified as necessary.

5 **Reboot the robot controller.**

Modify the PLC handshake program PK_PLC_RECIPE_OPERATION.TP

The standard PK_PLC_RECIPE_OPERATION.TP program is shown below. You can modify the program to suit your needs. The basic steps are:

1 **For the PLC, specify operations and parameters, depending on group input (GI), etc.**

2 **Call a KAREL program, depending on the specified operation.**

3 **Check the KAREL program status.**

This program obtains operations and parameters by communicating with a PLC, and then performs recipe operation. It is executed automatically when the [Recipe Operation DI] that was set on the [Recipe] settings screen under the tree view is turned ON.

This program can only be run when you are not running production, i.e., all the robots are aborted.

This program obtains operations and parameters by means of group I/O after performing a handshake with the PLC, but the means of communication can be changed to suit the application.

In this program:

1. All the I/O numbers can be configured on the [Recipe Settings] screen.
2. [Robot Controller Ready] is a DO signal that is output by the robot to check whether the PLC is ready to receive data.
3. [PLC Ready] is a DI signal that is output to the robot by the PLC when it is ready to receive data.
4. [Recipe ID] is used by the PLC to send operations and parameters to the robot.
5. [Echo] is used in order for the robot to resend the operations and parameters that have been received to the PLC to confirm receipt.
6. A KAREL program will be selected in accordance with the operation sent from the PLC. Depending on the operation, the parameters sent from PLC are specified as the arguments for a KAREL program, and the KAREL program is called.
7. The KAREL program status indicates whether the recipe operation was successful or not. When the recipe operation was successful, the [Recipe Operation Succeeded] DO turns ON. If the status is anything other than 0, the [Recipe Fault DO] turns ON.

PK_PLC_RECIPE_OPERATION.TP

```
 1: !-------------------------------- ;
 2: ! Communication program between a ;
 3: ! PLC and iRPickTool to perform ;
 4: ! Recipe operations. ;
 5: ! This program runs if the Recipe ;
 6: ! Change DI setup in the Recipe ;
 7: ! menu is turned on. ;
 8: ! Users can modify the PLC ;
 9: ! handshake - you can change ;
10: ! the DI/DO, GI/GO and registers ;
11: ;
12: !-------------------------------- ;
13: ;
14: !Recipe Operation Function Codes ;
15: ! CREATE = 1 ;
16: ! DELETE = 3 ;
17: ! SAVE_ACTIV_RECIPE = 4 ;
18: ! RESTORE_ACTIVE_RECIPE = 5 ;
19: ! EXPORT = 6 ;
20: ! IMPORT = 7 ;
21: ! SWITCH = 8 ;
22: ! ;
23: ! Numeric registers R[95]-R[98] ;
24: ! are used in this program. ;
25: ! Users can re-assign the ;
26: ! registers to other numbers ;
27: ! to suit their conveninence. ;
28: ! ;
29: !R[95] is used for storing status ;
30: ! ;
31: CALL PKRCGETPLCIO("PLC Ready"=2,"I/O Reg"=97) ;
32: CALL PKRCGETPLCIO("Recipe ID"=6,"I/O Reg"=96) ;
33: !Turn off Success DO ;
34: CALL PKRCSETPLCIO("Recipe Operation Successful"=4,"Value"=0) ;
35: ! Turn off Fault DO ;
36: CALL PKRCSETPLCIO("Recipe Fault DO"=5,"Value"=0) ;
37: ;
38: !Tell PLC robot ready to receive ;
39: CALL PKRCSETPLCIO("Robot Controller Ready"=3,"Value"=1) ;
40: ! Wait for PLC ready to send ;
41: WAIT DI[R [97]]=ON TIMEOUT,LBL[4000] ;
42: !Read function code ;
43: R[98:TemporaryUse4]=GI[R[96]] ;
44: ! Echo Recipe Operation to PLC ;
```

```
   45: CALL PKRCSETPLCIO("Echo"=7,"Value"=R[98:TemporaryUse4]) ;
   46: ! Signal PLC that data received. ;
   47: CALL PKRCSETPLCIO("Robot Controller Ready"=3,"Value"=0) ;
   48: ! Wait for PLC acknowledgement. ;
   49: WAIT DI[R[97]]=OFF TIMEOUT,LBL[4000] ;
   50: ;
  51: SELECT R[98:TemporaryUse4]=1,JMP LBL[100];
   52: =3,JMP LBL[300] ;
   53: =4,JMP LBL[400] ;
   54: =5,JMP LBL[500] ;
   55: =6,JMP LBL[600] ;
   56: =7,JMP LBL[700] ;
   57: =8,JMP LBL[800] ;
   58: ELSE,JMP LBL[4000] ;
   59: ;
   60: LBL[100:Create] ;
   61: !Tell PLC robot ready to receive ;
   62: CALL PKRCSETPLCIO("Robot Controller Ready"=3,"Value"=1) ;
   63: ! Wait for PLC ready to send ;
   64: WAIT DI[R [97]]=ON TIMEOUT,LBL[4000] ;
   65: ! Read recipe_id register ;
   66: R[98:TemporaryUse4]=GI[R[96]] ;
   67: ! Echo Recipe Id reg to PLC ;
   68: CALL PKRCSETPLCIO("Echo"=7,"Value"=R[98:TemporaryUse4]) ;
   69: ! Signal PLC that data received. ;
   70: CALL PKRCSETPLCIO("Robot Controller Ready"=3,"Value"=0) ;
   71: ! Wait for PLC acknowledgement. ;
  72: WAIT DI[R[97]]=OFF TIMEOUT,LBL[4000] ;
   73: !Tell PLC robot ready to receive ;
   74: CALL PKRCSETPLCIO("Robot Controller Ready"=3,"Value"=1) ;
   75: ! Wait for PLC ready to send ;
   76: WAIT DI[R [97]]=ON TIMEOUT,LBL[4000] ;
   77: ! Read string reg id for storing ;
   78: ! recipe name ;
   79: R[98:TemporaryUse4]=GI[R[96]] ;
   80: ! Echo string reg Id to PLC ;
   81: CALL PKRCSETPLCIO("Echo"=7,"Value"=R[98:TemporaryUse4]) ;
   82: ! Signal PLC that data received. ;
  83: CALL PKRCSETPLCIO("Robot Controller Ready"=3,"Value"=0) ;
   84: ! Wait for PLC acknowledgement. ;
  85: WAIT DI[R[97]]=OFF TIMEOUT,LBL[4000] ;
  86: CALL PKRCCREATE("RecipeID Reg"=R[98:TemporaryUse4],"RecipeNm StrReg"=R[96:TemporaryUse2],"Stat
Reg"=95)
   87: JMP LBL[5000] ;
   88: ;
  89: LBL[300:Delete] ;
   90: !Tell PLC robot ready to receive ;
  91: CALL PKRCSETPLCIO("Robot Controller Ready"=3,"Value"=1) ;
   92: ! Wait for PLC ready to send ;
   93: WAIT DI[R [97]]=ON TIMEOUT,LBL[4000] ;
   94: ! Read recipe id to delete ;
  95: R[98:TemporaryUse4]=GI[R[96]] ;
   96: ! Echo Recipe Id to PLC ;
  97: CALL PKRCSETPLCIO("Echo"=7,"Value"=R[98:TemporaryUse4]) ;
   98: ! Signal PLC that data received. ;
  99: CALL PKRCSETPLCIO("Robot Controller Ready"=3,"Value"=0) ;
  100: ! Wait for PLC acknowledgement. ;
  101: WAIT DI[R[97]]=OFF TIMEOUT,LBL[4000] ;
  102: CALL PKRCDELETE("Recipe ID"=R[98:TemporaryUse4],"Stat Reg"=95)
  103: JMP LBL[5000] ;
```

```
104: ;
105: LBL[400:Save];
106: CALL PKRCSAVE("Stat Reg"=95) ;
 107: JMP LBL[5000] ;
108: ;
109: LBL[500:Restore] ;
110: CALL PKRCRESTORE("Stat Reg"=95) ;
111: JMP LBL[5000] ;
112: ;
113: LBL[600:Export] ;
114: CALL PKRCEXPORT("Stat Reg"=95) ;
115: JMP LBL[5000] ;
116: ;
117: LBL[700:Import] ;
118: CALL PKRCIMPORT("Stat Reg"=95) ;
119: JMP LBL[5000] ;
120: ;
121: LBL[800:Switch] ;
122: !Tell PLC robot ready to receive ;
123: CALL PKRCSETPLCIO("Robot Controller Ready"=3,"Value"=1) ;
124: ! Wait for PLC ready to send ;
 125: WAIT DI[R [97]]=ON TIMEOUT,LBL[4000] ;
126: ! Read recipe id to switch to ;
127: R[98:TemporaryUse4]=GI[R[96]] ;
128: ! Echo Recipe Id to PLC ;
129: CALL PKRCSETPLCIO("Echo"=7,"Value"=R[98:TemporaryUse4]) ;
130: ! Signal PLC that data received. ;
 131: CALL PKRCSETPLCIO("Robot Controller Ready"=3,"Value"=0) ;
132: ! Wait for PLC acknowledgement. ;
133: WAIT DI[R[97]]=OFF TIMEOUT,LBL[4000] ;
134: LBL[801] ;
135: CALL PKRCACTIVE("Recipe ID"=R[98:TemporaryUse4],"Stat Reg"=95) ;
136: IF (R[95:TemporaryUse1]=127277) THEN ;
137: ! 127277 = Unsaved data exists ;
138: CALL PKRCGETPLCIO("Skip Unsaved Recipe"=8,"I/O Reg"=96) ;
139: IF R[96:TemporaryUse2]=0,JMP LBL[5000] ;
140: IF (DI[R[96]]=OFF) THEN ;
141: CALL PKRCSAVE("Stat Reg"=95) ;
142: ELSE ;
143: CALL PKRCRESTORE("Stat Reg"=95) ;
144: ENDIF ;
145: IF R[95:TemporaryUse1]<>0,JMP LBL[5000] ;
146: JMP LBL[801] ;
147: ENDIF ;
148: JMP LBL[5000] ;
149: ;
150: LBL[4000:PLC Recipe Error] ;
151: ! 127294 = PLC Rec Op failed ;
152: R[95:TemporaryUse1]=127294 ;
153: JMP LBL[5000] ;
154: ;
155: LBL[5000:Check status] ;
156: IF (R[95:TemporaryUse1]<>0) THEN ;
157: ! Turn Fault DO on ;
158: CALL PKRCSETPLCIO("Recipe Fault DO"=5,"Value"=0) ;
159: ! Post error and abort ;
160: CALL PKPOSTERR("Error Code"=R[95:TemporaryUse1],"ABORT"=2) ;
161: ELSE ;
162: ! Turn success DO on ;
163: CALL PKRCSETPLCIO("Recipe Operation Successful"=4,"Value"=1) ;
```

```
164: ENDIF ;
165: JMP LBL[10000] ;
166: ;
167: LBL[10000: End] ;
```

8.1.4. Switching recipe with TP program

By turning ON the DI of [Recipe Operation DI] set on the Recipe Settings screen, the TP program PK_PLC_RECIPE_OPERATION (dedicated for recipe operation) will automatically be executed.

PK_PLC_RECIPE_OPERATION is a sample program of the handshake with the PLC. Modify the program according to the actual operation. Switching recipe is performed by the PKRCACTIVE on line 135 of PK_PLC_RECIPE_OPERATION. After turning ON the DI of [Recipe Operation DI], confirm that the recipe operation is complete before starting production. For example, notify the main program via the register that the process is completed after the PKRCACTIVE program. For details, see "I.8.1.3. Recipe operation by means of a PLC(P.557)". (It can also be used without PLC.)

⚠CAUTION

- For changing recipes, avoid running PK_PLC_RECIPE_OPERATION or PKRCACTIVE directly using CALL or RUN of the TP program. When PK_PLC_RECIPE_OPERATION or PKRCACTIVE is run directly, the following alarm may occur.
  - "IRPK-075 This operation is allowed only when the program is aborted"
  - In addition, if a program stops during recipe switching, error IRPK-083 "Another recipe process is running." may occur the next time a recipe operation is performed. PK_PLC_RECIPE_OPERATION is designed not to stop in the middle of the process if you run it with the DI of [Recipe Operation DI].
- Perform error processing to not start production when the register value for the status specified in argument 2 of PKRCACTIVE becomes anything other than 0 (changing recipe has failed).
- The recipe operation and failure of the recipe change should be checked before PKWCSTART or PKWCCHECK.
- Do not run the program using motion group during recipe operation.
- To avoid starting production during recipe operation, call PKWCEND before turning on the DI of [Recipe Operation DI] or in front of the PK_PLC_RECIPE_OPERATION.

8.1.5. KAREL program

⚠CAUTION

When performing a recipe operation using the KAREL programs introduced here, call PKWCEND first to stop the sensors. In addition, if the recipe operation is forced to stop, error IRPK-083 "Another recipe process is running." may occur the next time a recipe operation is performed. To prevent the recipe from being forced to stop during operation, we recommend setting the TP programs that use the KAREL programs introduced here to ignore pause in the programs' detail settings.

8.1.5.1. PKRCCREATE.PC

This KAREL program is used only when creating a recipe. You can run the program only when all the robot controller programs that have been set in the tree view have finished.

When you run this program, it does the following:

- Turns [Recipe Fault DO] OFF if it is ON.
- Creates a recipe.
- Sets the ID of the created recipe.
- Sets the name of the created recipe.
- Sets the status.
- Turns [Recipe Fault DO] ON if the status is not 0.

Argument 1: Specify the register where the recipe ID is stored.

Argument 2: Specify the String Register where the recipe name is stored.

Argument 3: Specify the register to return the status to.

Usage:

Create a recipe. Return the recipe ID to R[2], the recipe name to SR[3], and the status to R[1].

```
CALL PKRCCREATE (RecipeID Reg=2,RecipeNm StrReg=3,Stat Reg=1)
```

8.1.5.2. PKRCDELETE.PC

This KAREL program is used only when deleting a recipe. You can run the program only when all the robot controller programs that have been set in the tree view have finished.

When you run this program, it does the following:

- Turns [Recipe Fault DO] OFF if it is ON.
- It then checks if the recipe is a valid recipe.
- If it is valid, it deletes the specified recipe.
- Sets the status.
- Turns [Recipe Fault DO] ON if the status is not 0.

Argument 1: Specify the ID of the recipe that you want to delete.

Argument 2: Specify the register to return the status to.

Usage:

Delete the recipe for which the ID is 3. Return the status to R[1].

```
PKRCDELETE(Recipe ID=3,Stat Reg=1)
```

8.1.5.3. PKRCSAVE.PC

This KAREL program is used only when saving a recipe. You can run the program only when all the robot controller programs that have been set in the tree view have finished.

When you run this program, it does the following:

- Turns [Recipe Fault DO] OFF if it is ON.
- Saves a recipe
- Sets the status.
- Turns [Recipe Fault DO] ON if the status is not 0.

Argument 1: Specify the register to return the status to.

Usage:

Save a recipe. Return the status to R[1].

```
PKRCSAVE(Stat Reg=1)
```

8.1.5.4. PKRCRESTORE.PC

This KAREL program is used only when loading a recipe. You can run the program only when all the robot controller programs that have been set in the tree view have finished.

When you run this program, it does the following:

- Turns [Recipe Fault DO] OFF if it is ON.
- Loads a recipe.
- Sets the status.
- Turns [Recipe Fault DO] ON if the status is not 0.

Argument 1: Specify the register to return the status to.

Usage:

Load a recipe. Return the status to R[1].

```
PKRCRESTORE(Stat Reg=1)
```

8.1.5.5. PKRCEXPORT.PC

This KAREL program is used only when exporting a recipe. You can run the program only when all the robot controller programs that have been set in the tree view have finished.

When you run this program, it does the following:

- Turns [Recipe Fault DO] OFF if it is ON.
- Exports a recipe.
- Sets the status.
- Turns [Recipe Fault DO] ON if the status is not 0.

Argument 1: Specify the register to return the status to.

Usage:

Export a recipe. Return the status to R[1].

```
PKRCEXPORT(Stat Reg=1)
```

8.1.5.6. PKRCIMPORT.PC

This KAREL program is used only when importing a recipe. You can run the program only when all the robot controller programs that have been set in the tree view have finished.

When you run this program, it does the following:

- Turns [Recipe Fault DO] OFF if it is ON.
- Imports a recipe.
- Sets the status.
- Turns [Recipe Fault DO] ON if the status is not 0.

Argument 1: Specify the register to return the status to.

Usage:

Import a recipe. Return the status to R[1].

```
PKRCIMPORT(Stat Reg=1)
```

8.1.5.7. PKRCACTIVE.PC

This KAREL program is used only when switching recipes. You can run the program only when all the robot controller programs that have been set in the tree view have finished.

When you run this program, it does the following:

- Turns [Recipe Fault DO] OFF if it is ON.
- It then checks if the recipe is a valid recipe.
- Loads the specified recipe if it is valid.
- Sets the status.
- Turns [Recipe Fault DO] ON if the status is not 0.

Argument 1: Specify an ID of the new recipe you wish to switch to.

Argument 2: Specify the register to return the status to.

Usage:

Switch to the recipe for which the ID is 3. Return the status to R[1].

```
CALL PKRCACTIVE(Recipe ID=3,Stat Reg=1)
```

8.1.5.8. PKRCGETPLCIO.PC

This KAREL program acquires the current I/O settings of the recipe for recipe operation by means of a PLC.

Argument 1: Select the I/O type from “Recipe Operation”, “PLC Ready”, “Robot Controller Ready”, “Recipe Operation Succeeded”, “Recipe Fault DO”, “Recipe ID”, “Echo”, and “Skip Unsaved Recipe”.

Argument 2: Specify the register to which the I/O number is returned.

Example:

The following instruction acquires the DI number for [Skip Unsaved Recipe] and stores it in Register [1].

```
CALL PKRCGETPLCIO("Skip Unsaved Recipe","I/O Reg"=1)
```

8.1.5.9. PKRCSETPLCIO.PC

This KAREL program configures I/O for recipe operation by means of a PLC.

Argument 1: Select the I/O type from “Robot Controller Ready”, “Recipe Operation Succeeded”, “Recipe Fault DO”, and “Echo”.

Argument 2: If argument 1 is “Echo”, set the GP value. Otherwise, set 1 to turn the DO ON, or 0 to turn the DO OFF.

Example:

The following instruction turns the DO for [Robot Controller Ready] ON.

```
CALL PKRCSETPLCIO("Robot Controller Ready","Value"=1)
```

8.2. Checking the production status

The production status of each iRPickTool object can be checked.

The screen to check the production status opens with the following procedure.

1. **Open the robot homepage.**
2. **Select [iRVision/iRPickTool] in the menu area and click [iRPickTool Workcell Production Status] in the content display area.**

MEMO

You can also check the production status on the teach pendant by following the procedure below.

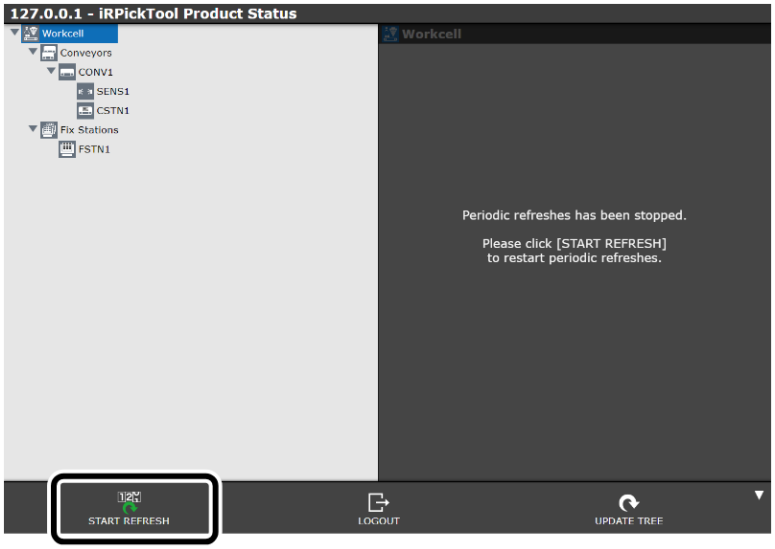
1. Press the [MENU] key on the teach pendant.
2. Select [NEXT].
3. Select [STATUS].
4. Press the F1 [TYPE] key and select [iRPickTool] from the options.

Basic operations

The operations common to all objects are described below.

Stopping and resuming start refresh

If you want to stop periodic refresh intentionally, click [STOP REFRESH] during a periodic refresh.



Click [START] to resume periodic refresh.

Resetting the production status

To reset the production time and statistic values and resume the collection of statistics from that point in time, click the [RESET] key.

Refreshing the tree

If you perform an operation related to the tree while logged in to the production status screen, such as adding a conveyor station on the iRPickTool setup screen, the change is not automatically reflected on the production status screen. In this case, click [UPDATE TREE]. 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.

8.2.1. Checking the status of a conveyor

The part processing status of an entire conveyor can be checked.

Select the desired conveyor from the tree.

[Run Time]

Time elapsed since production started

[Conveyor Speed(mm/s)]

Conveyor speed When the tracking type is “Circle”, the units are deg/s.

[Tray]

Name of the tray used on this 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.

[Total Pick Count]

Total number of parts processed on the selected conveyor. When trays are used, the total number of cells or trays processed on the selected conveyor is displayed, depending on the selected display unit.

[Pick Rate(counts/min)]

Average number of conveyed parts obtained by dividing the total number of parts processed on the selected conveyor by the production time. When trays are used, the average number of conveyed cells or trays obtained by dividing the total number of cells or trays processed on the selected conveyor by the production time, depending on the selected display unit.

[Miss Count]

Total number of parts not processed on any conveyor station. It does not include parts that have failed to be detected by vision.

[RESET]

Pressing [RESET] resets all the data. After the reset, recording starts again.

8.2.2. Checking the status of a sensor

The status of a sensor can be checked.

Select the desired sensor from the tree.

[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 at which a trigger occurs.

Specifically, make [Trigger Distance] set in “I.2.3.5. Setting up a sensor(P.80)”, “I.3.3.6. Setting up a sensor(P.194)”, or “I.5.5.1.2. Distance and Find part by vision(P.291)” longer than [Action Time(ms)] below.

[Action Time(ms)]

Time it takes before the selected sensor detects a part and that detected part is put on the most upstream conveyor station. If this value is larger than the trigger interval, the value in [Overrun Count] becomes larger.

[Interval(ms)]

Interval at which a trigger occurs

[Num. Found]

Number of parts detected each time the image processing is performed when the vision system is used (number of parts in the camera's field of view)

[RESET]

Pressing [RESET] resets all the data. After the reset, recording starts again.

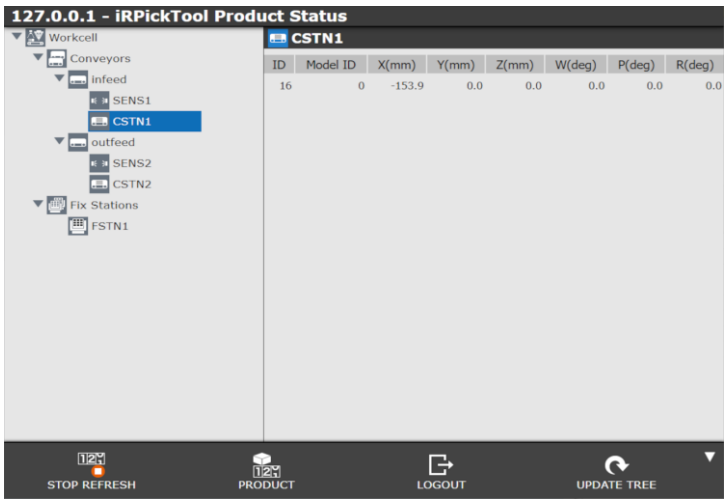
8.2.3. Checking the status of a conveyor station

The part processing status of a conveyor station can be checked.

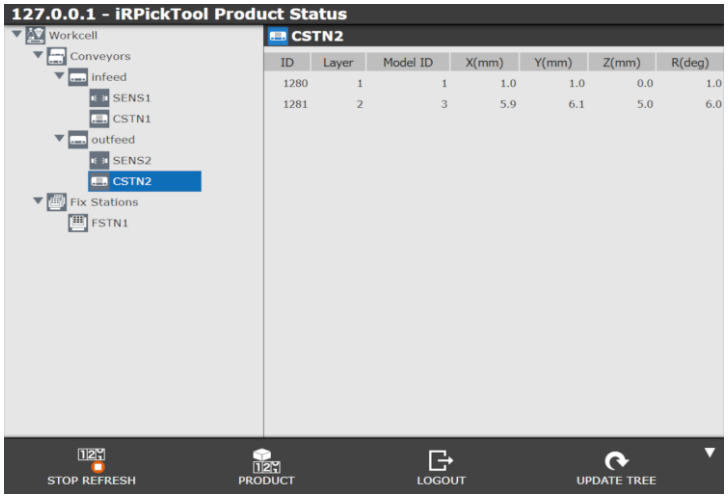
Select the desired conveyor station from the tree.

If you click the [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.

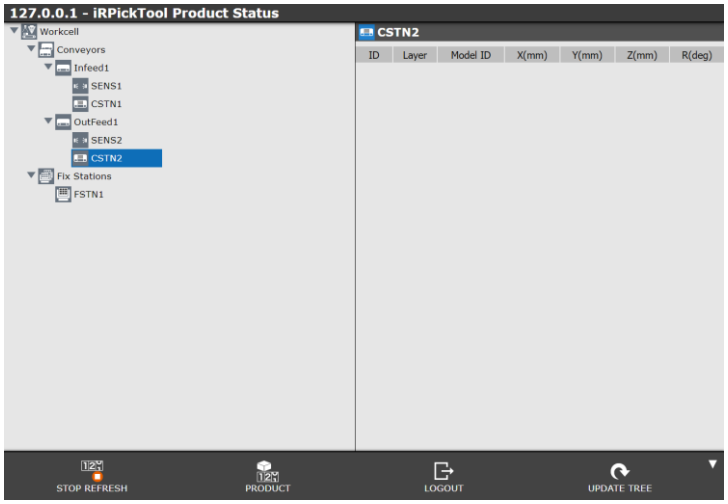
This screen is the one displayed when trays are not used. Each entry shows the distance of the part on the conveyor along the flow direction (x), and along the y, z, w, p and r directions from the nominal tracking frame.



When trays are used, the status of each cell in the conveyor station’s queue is displayed as shown below.



When a circular conveyor is used, the current position of the parts is displayed as an angle and radius instead of X and Y as shown below.



If you click the [PRODUCT] key, the processing status screen is displayed again.

Production Status

The information about the part processing status for each individual model ID is listed.

[Trays]

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 status of each part in the conveyor station is listed. 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.

When a circular conveyor is used, the current position of the part is displayed at an angle and radius instead of X and Y.

8.2.4. Checking the status of a fixed station

The cell or tray processing status of a fixed station can be checked.

Select the desired fixed station from the tree.

[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.

[Total Pick Count]

Total number of cells or trays processed by the selected fixed station

[Pick Rate(counts/min)]

Average number of conveyed cells or trays obtained by dividing the total number of cells or trays processed on this fixed station by the production time.

[Empty Cell / Total Cell]

Number of empty cells among all the cells in the currently processed tray. This represents the unfilled quantity of the tray.

[RESET]

Pressing [RESET] resets all the data. After the reset, recording starts again.

8.3. Limiting the movable range of the final axis

When performing a linear motion involving a posture change, the robot moves so as to minimize the rotation angle of each axis. Such a movement is called a minimum rotation. In a visual tracking system where the rotation angle of the final axis changes every time on the pickup and placement sides, the angle of the final axis may move out of the axis movable range as a minimum rotation is repeated, resulting in a stroke limit alarm.

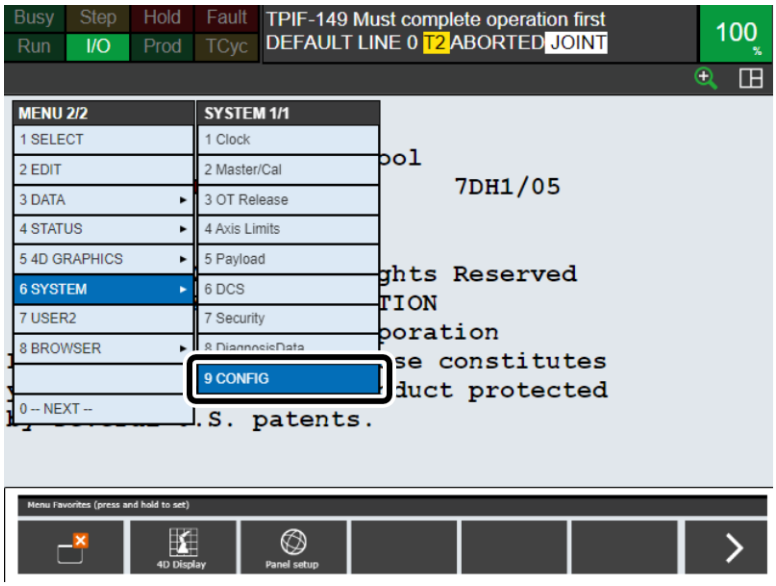
This function prevents the stroke limit of the final axis from being violated by going a long way intentionally in cases when a minimum rotation causes a stroke limit alarm.

The function takes effect when the following conditions are met:

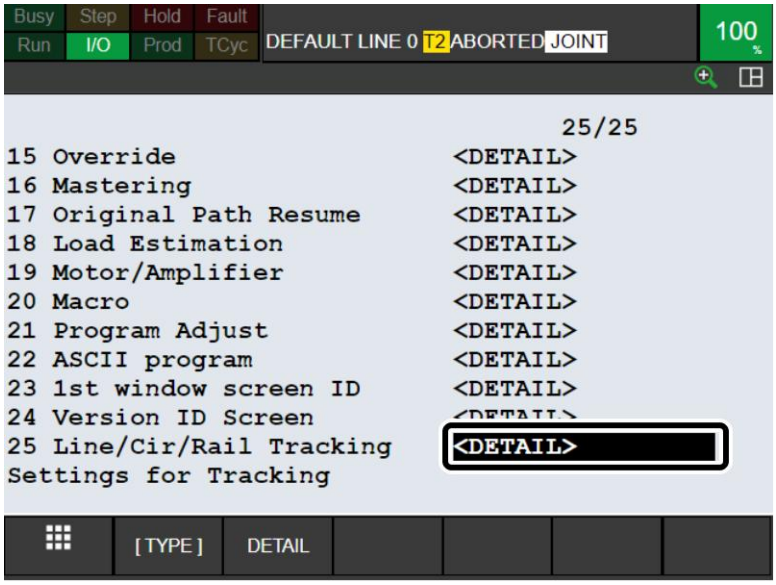
- The tracking type is [Line].
- The line tracking is in progress.
- A linear motion is performed.
- There is a wrist posture.

By default, this function is enabled.

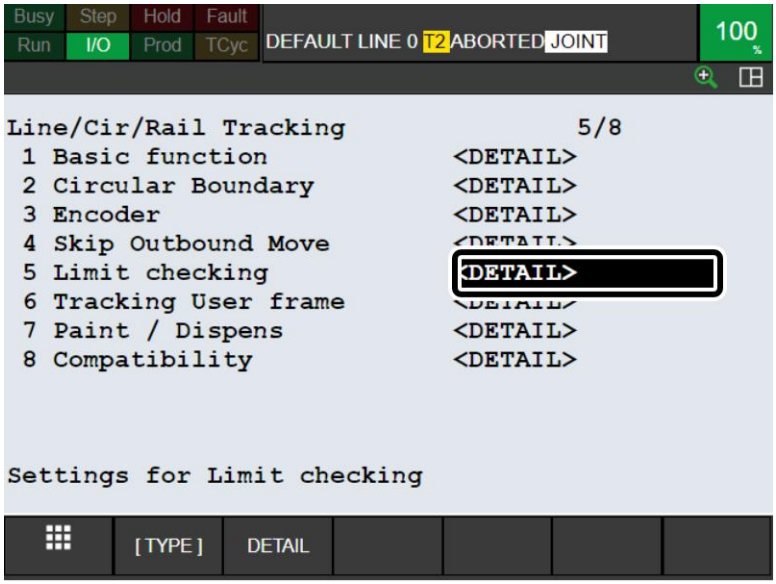
1 **Move to [Config].**



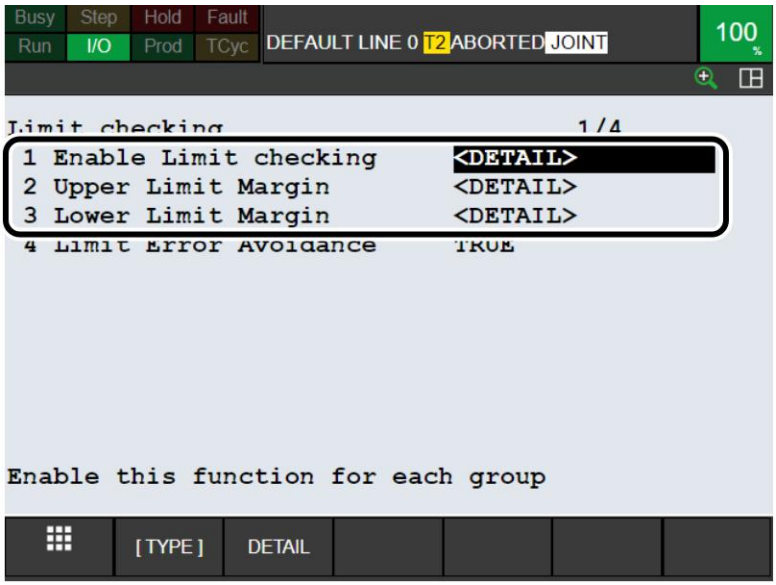
2 **Move to [Line/Cir/Rail Tracking].**



3 Enter [Limit Check].



4 Configure the three limit check settings.



1. Enable Limit Checking

   This is the flag indicating whether this function is enabled or disabled. To disable the function, set this variable to FALSE.

2. Upper Limit Margin (deg)

   This is the offset (units: degrees) from the upper limit angle of the final axis. Set this variable as necessary.

3. Lower Limit Margin (deg)

   This is the offset (units: degrees) from the lower limit angle of the final axis. Set this variable as necessary.

A change to any of these settings 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 Upper Margin and Lower Margin, the movable range of the J6 axis is limited to –180 to 180 degrees.

8.4. 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 too far 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.

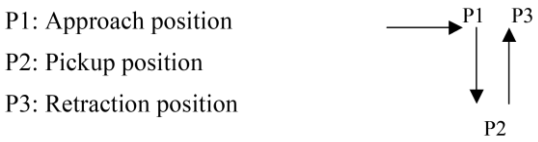
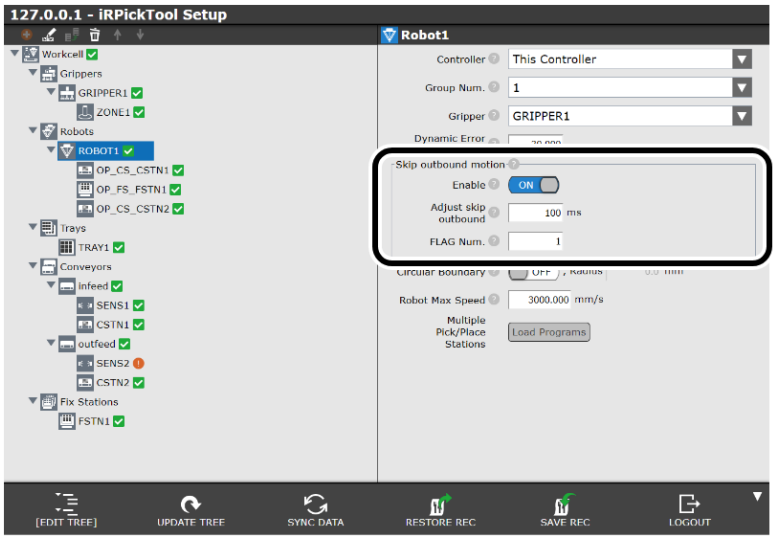


Fig. 8.4-1 Positions for a tracking program

This function determines whether the destination position is within the tracking area during the execution of a motion instruction for traveling to any of the above positions.

If it determines that the destination position is outside the tracking area, the function skips the instruction for traveling to that position and proceeds to the next instruction.

To use this function, set the items under [Skip outbound motion] on 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 units are milliseconds.

Suppose that the robot travels to pick a part in the above-mentioned motion pattern. If when it reaches to P1, it is still within the tracking area, but it goes out of the tracking area while it moves from P1 to P2 or it moves from P2 to P3, it is more efficient to go to pick the following part immediately, with giving up this part before going to P1.

In such a case, if you set the travel time from P1 to P3 via P2, you can save the unnecessary travel time to move to P1.

For example, measure the maximum travel time from P1 to P3 via P2 with the TIMER instruction. If the measured time is 100 ms, set 100 to [Adjust skip outbound].

Once the robot starts to travel to P1, whether the destination position is within the tracking area is determined at the position reached after only 100 milliseconds of travel. If the destination position is determined to be outside the tracking area, all the motion instructions for P1, P2, and P3 are skipped.

[FLAG Num.]

If a skip occurs, the flag corresponding to the number specified here is turned on. Note that the system software only turns the flag on. If the flag is turned on, it needs to be turned off by a robot program.

NOTE

This function is not effective if the destination position moves out of the tracking area when the robot continues the tracking motion while it waits for the next part after picking a part with a hand capable of holding multiple parts. This is because the continuation of the tracking motion while waiting for the next workpiece is not taken into consideration.
In this case, stop the tracking motion by executing the “STOP_TRACKING” instruction before calling PKCSGETQUE, as described in “I.5.9.2.2. Pick program(P.370)”.

Sample program

In order to use this function, the tracking program for the pickup and placement motions needs to be modified. An example of the modified program is shown below. Major changes are underlined. The flag number is 1, and the time required to arrive from P1 to P3 via P2 is 100 ms.

[FLAG Num.]= 1
[Adjust skip outbound] 100 (ms)

PICK1.TP

```
 1: UTOOL_NUM=0
 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.

8.5. Checking the robot torque

In the event of a motor overheat, the robot stops as a result of an alarm. Check the torque applied to each axis during a tracking motion to check if the motor is overheating.

You can display the torque monitor screen as follows.

1. **Press the [MENU] key on the teach pendant.**
2. **Select [NEXT].**
3. **Select [STATUS].**
4. **Press the F1 [TYPE] key, and select [Axis] from the options.**
5. **Press the [NEXT] key and then the F2 [MONITOR] key. The following screen appears.**



If the [Ave.] value for each axis displayed here increases gradually while the system is continuing constant operation, there is a risk of a motor overheat.

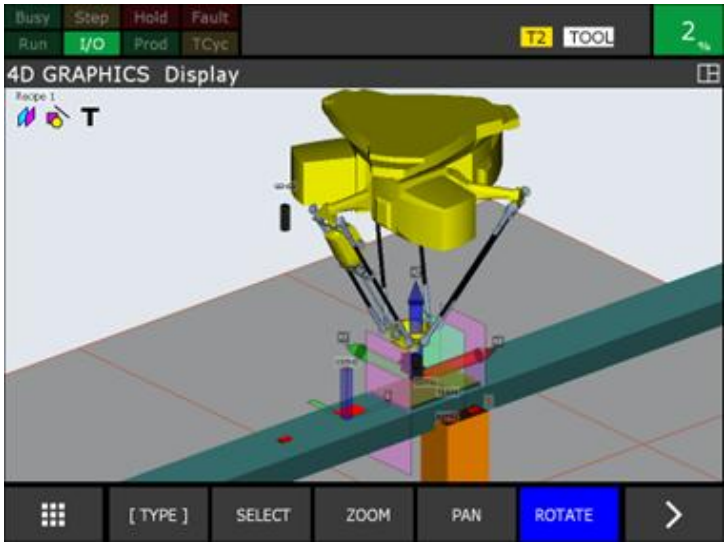
If so, decrease the speed of the tracking motion.

8.6. 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.

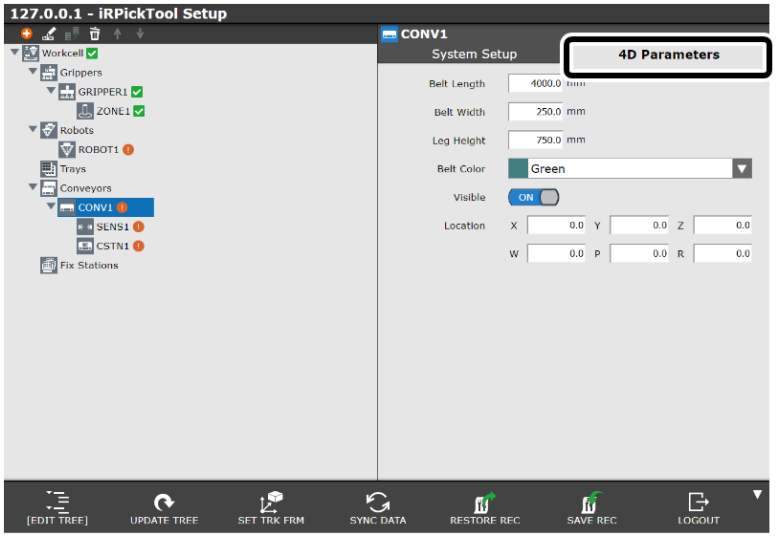
The figure below illustrates what you see in the 4D Display menu on the Teach Pendant when you have setup iRPickTool completely for a 2-robot workcell.



The 4D Setup is done in the iRPickTool tree view.

When the 4D Graphics (order number R764) is installed, the right pane of the tree view 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.



8.6.1. 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.

8.6.2. Creating the 4D Workcell

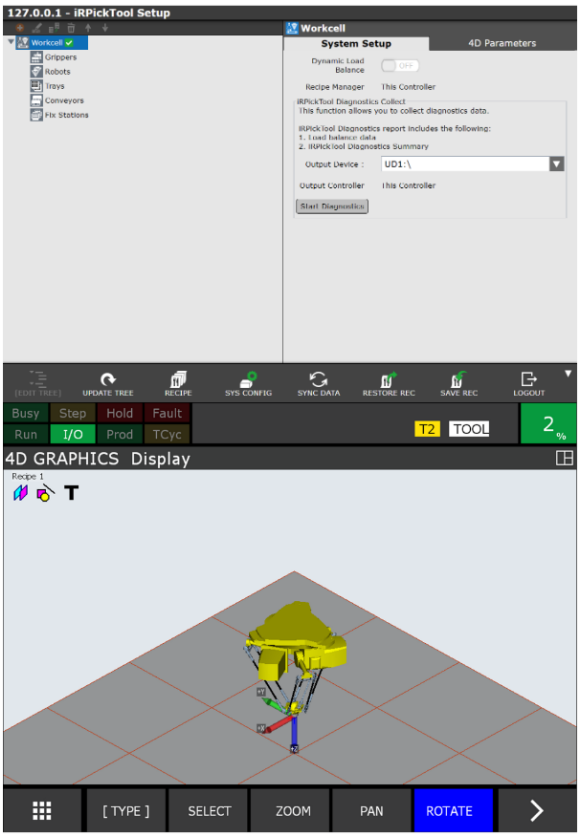
Empty workcell

When you have the R764: 4D Graphics option installed, and you have done no setup of iRPickTool in the tree view, 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 the RIPE Setup, then you will see all the robots in the RIPE ring. However, the robots may not be spaced correctly. They may all show up as being on top of each other at the same workcell origin. The robots will be spaced correctly only after you complete the track frame setup using the track guide in the Conveyor node of the tree view.

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.

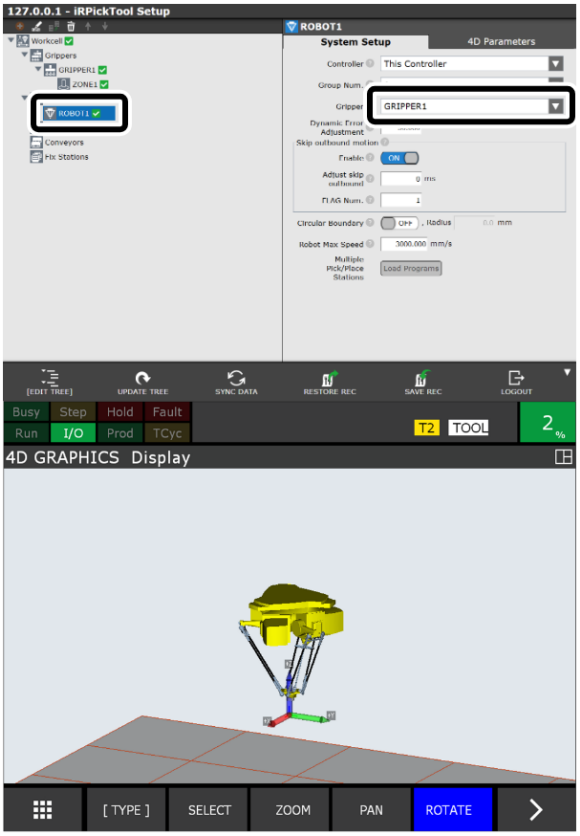


Creating 4D Workcell Objects

You create 4D Workcell Objects in the 4D Display menu automatically when you create that object in the tree view as you go through the setup steps.

Creating a new 4D Gripper

When you create a gripper in the tree view and assign it to a robot as shown in the figure below, the 3D gripper is created automatically. You can see the gripper at the end of the arm and also a text display above the robot showing the name of the robot.



Modifying the 4D Gripper

In the tree view on the left side, press a gripper, such as GRIPPER1, to setup in 4D. All the 4D Gripper properties that you can modify are shown in the right pane ([4D Parameters]). 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. If you want to change the file, refer to “I.8.6.7. Using user-defined object shapes(P.591)”.



[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 toggle switch.

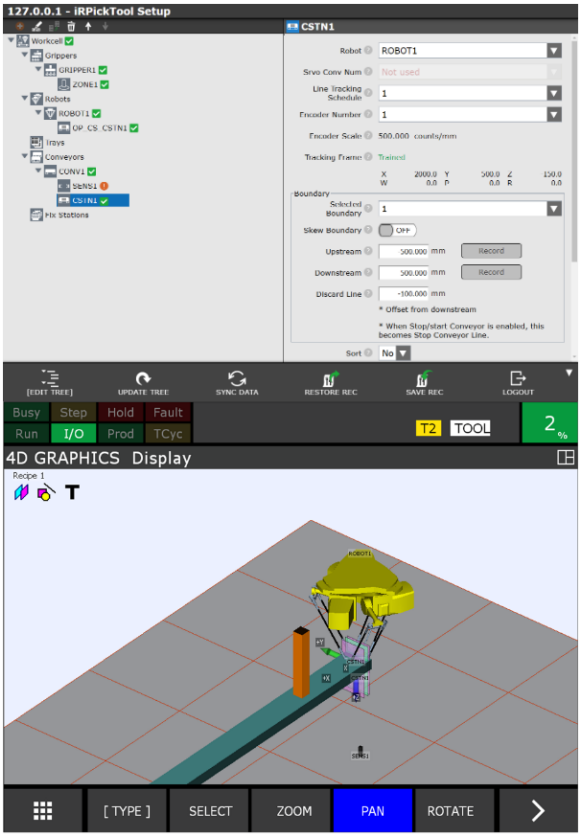
[Position]

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.

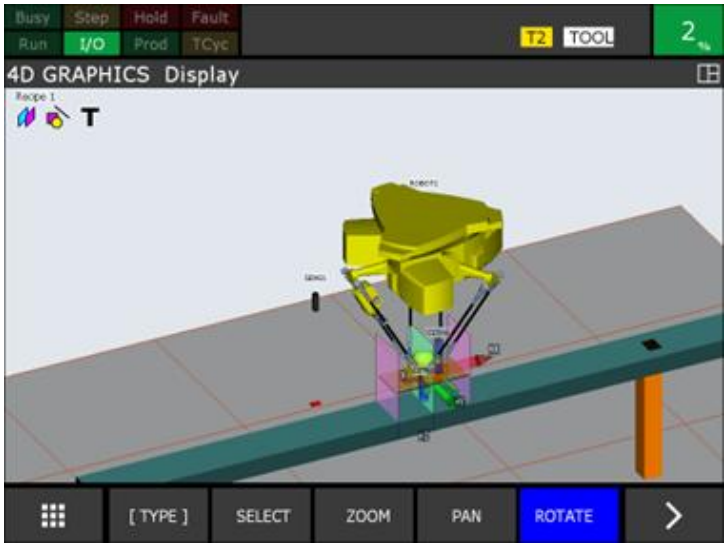
Creating a new 4D Conveyor

When you create a conveyor in the tree view, a 3D conveyor is automatically created with a default size of 4000 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 guide to teach the track frame, the conveyor will move to the proper location.

When you create a new conveyor in the tree view, 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. The sensor is displayed as a black rectangle box 25 mm x 25 mm x 100 mm in size at a default height of 900 mm from the track frame of the first conveyor station of the conveyor. 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. The discard boundary in light green is offset from the downstream boundary by a distance of -100 mm by default.

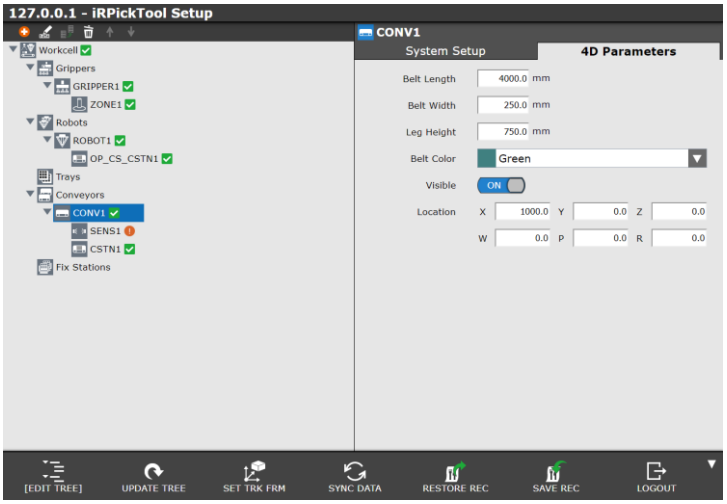


If you later go through the track guide in the tree view of the Conveyor setting screen and teach the track frame properly, and you go through the Sensor Guide in the Sensor menu, then the conveyor, the sensor and the conveyor station will all be located correctly in the workcell as shown in the figure below.



Modifying a 4D Conveyor

In the tree view on the left side, press a conveyor, such as CONV1, to setup in 4D. All the 4D Conveyor properties that you can modify are shown in the right pane ([4D Parameters]). 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. If you want to change the CAD file, see subsection "I.8.6.7. Using user-defined object shapes(P.591)".



[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]

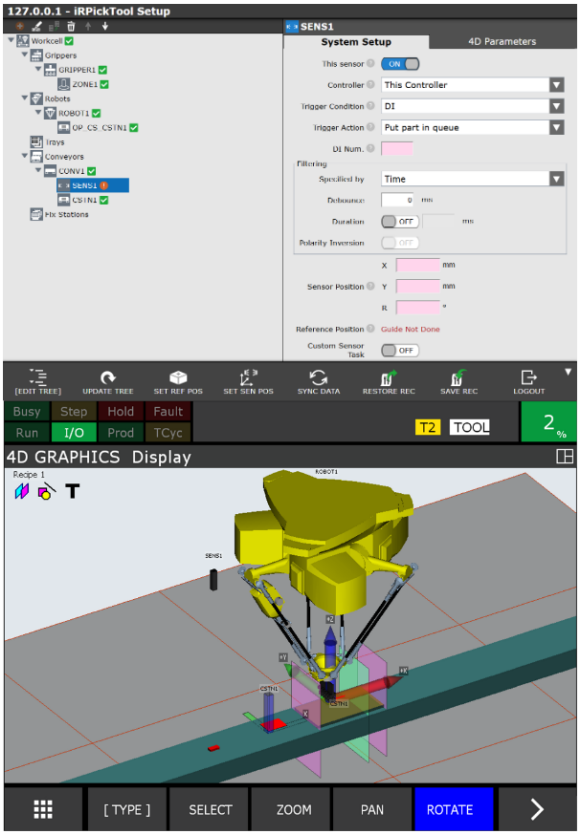
Enable or disable the visibility of the conveyor using the toggle switch.

[Position]

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.

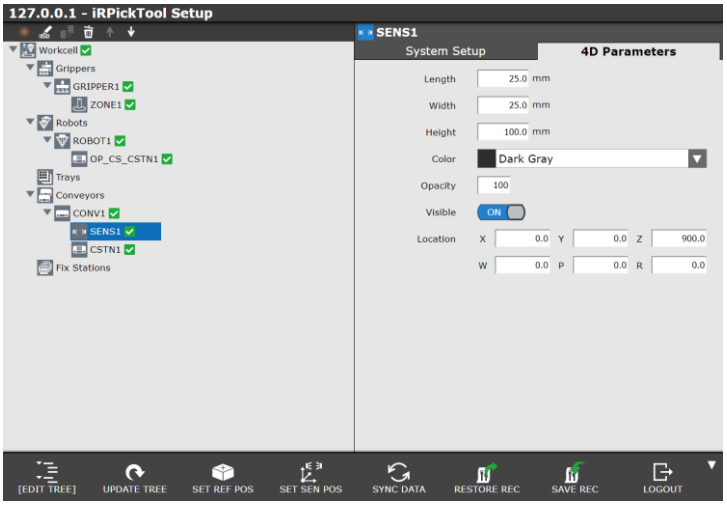
Creating a new 4D Sensor

When you create a Sensor in the tree view, 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, the sensor origin is set 900 mm above the tracking frame of the first conveyor station of the conveyor. When you set the sensor position by following the guide on the [Sensor] settings screen, the x and y coordinates of the sensor will be displayed at the proper positions. However, the displayed height will remain at 900 mm unless you correct it on the [4D Display] screen.



Modifying the 4D Sensor

In the tree view on the left side, press a Sensor, such as SENS1, to setup in 4D. All the 4D Sensor properties that you can modify are shown in the right pane ([4D Parameters]). 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. If you want to change the file, refer to “I.8.6.7. Using user-defined object shapes(P.591)”.



[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 of 0 to 100. 0 is completely transparent, and 100 is completely opaque.

[Visible]

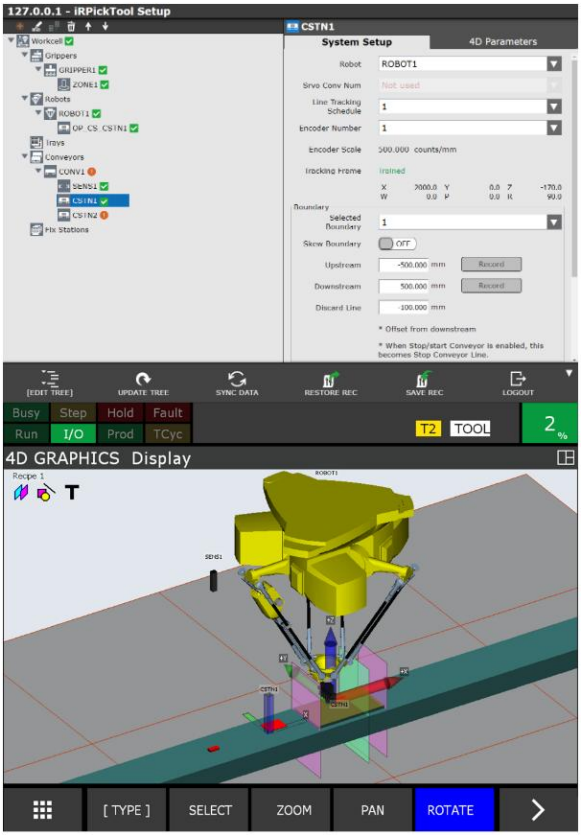
Enable or disable the visibility of the Sensor box using the toggle switch.

[Position]

The x, y, z, w, p, r offset from the track frame of the first conveyor station that the Sensor should be drawn.

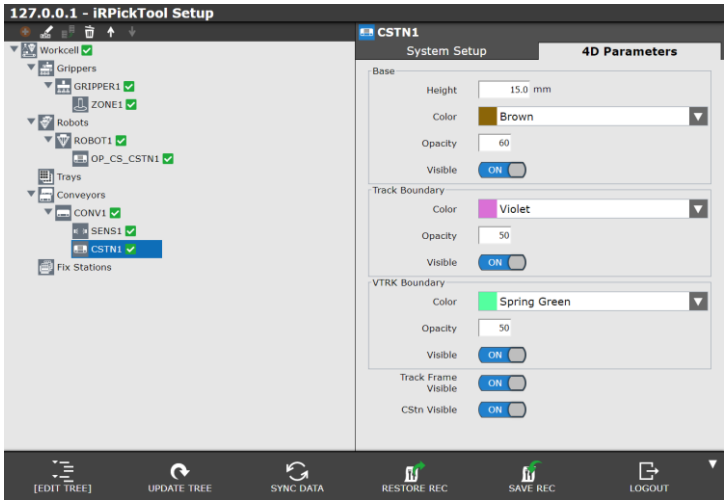
Creating a new 4D Conveyor Station

When you create a conveyor station in the tree view, 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. The discard boundary is displayed in Spring Green color. The tracking area between the upstream and downstream boundaries is displayed in brown color on the conveyor.



Modifying the 4D Conveyor Station

In the tree view on the left side, press a Conveyor Station, such as CSTN1, to setup in 4D. All the 4D Conveyor Station properties that you can modify are shown in the right pane ([4D Parameters]). 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.



[Base Height]

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 of 0 to 100. 0 is completely transparent, and 100 is completely opaque.

[Base Visible]

Enable or disable the visibility of the tracking area box or base using the toggle switch.

[Track Boundary Color]

The color of the tracking boundaries. 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 of 0 to 100. 0 is completely transparent, and 100 is completely opaque.

[Track Frame Visible]

Enable or disable the visibility of the tracking upstream and downstream boundaries using the toggle switch.

[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 of 0 to 100. 0 is completely transparent, and 100 is completely opaque.

[VTRK Boundary Visible]

Enable or disable the visibility of the discard boundary using the toggle switch.

[Track Frame Visible]

Enable or disable the visibility of the tracking frame box or base using the toggle switch.

[CStn Visible]

Enable or disable the visibility of the entire conveyor station using the toggle switch.

The entire conveyor station includes the tracking frame, the tracking upstream and downstream boundaries, the discard boundary and the base of the Conveyor station.

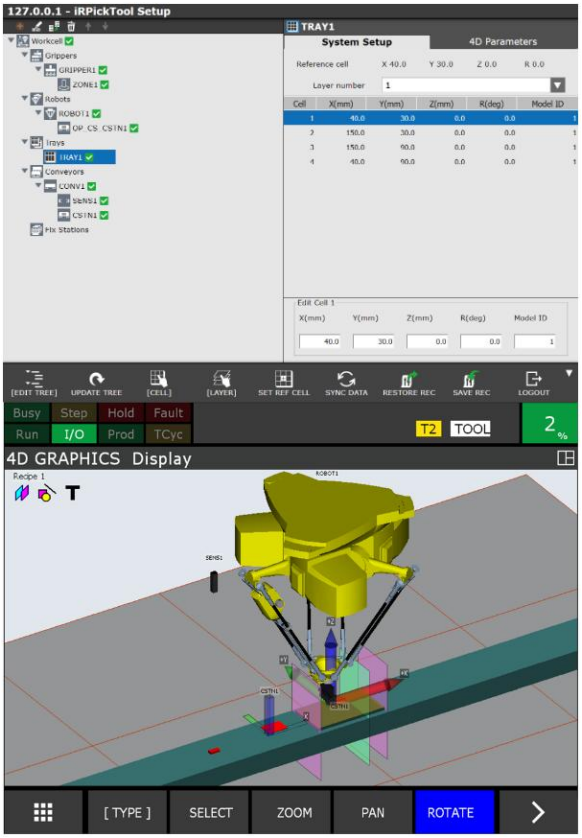
Creating a new 4D Tray

When you create a tray in the tree view 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 tree view. 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. In iRPickTool (J945), the reference position is also used and therefore tray cells are offset based on the reference position. iRPickTool Basic (J944) users are recommended to set the reference cell at the origin of the user frame to ensure that the 4D display aligns with the actual layout.

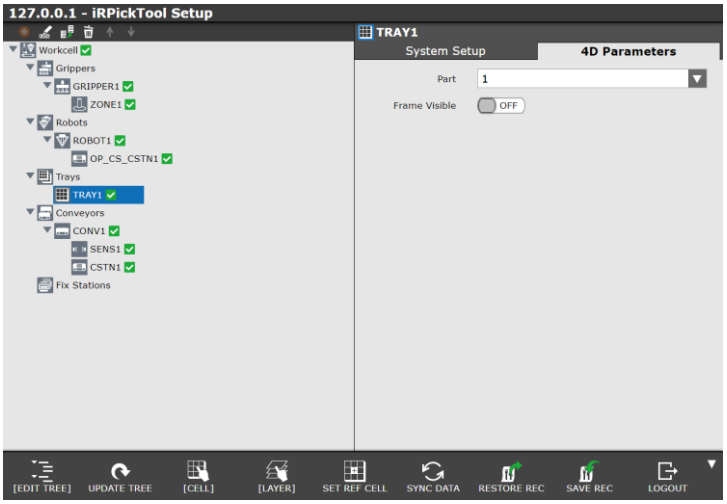
For a conveyor station tray, the tray frame is calculated by the sensor guide in the tree view as an offset from the track frame of the first conveyor station of the conveyor.

Note that trays will not be displayed simply by being defined under the [Tray] collection. They are displayed only when configured in either a fixed station or a conveyor.



Modifying the 4D Tray

In the tree view on the left side, press a Tray, such as TRAY1, to setup in 4D. All the 4D Tray properties that you can modify are shown in the right pane ([4D Parameters]). 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]

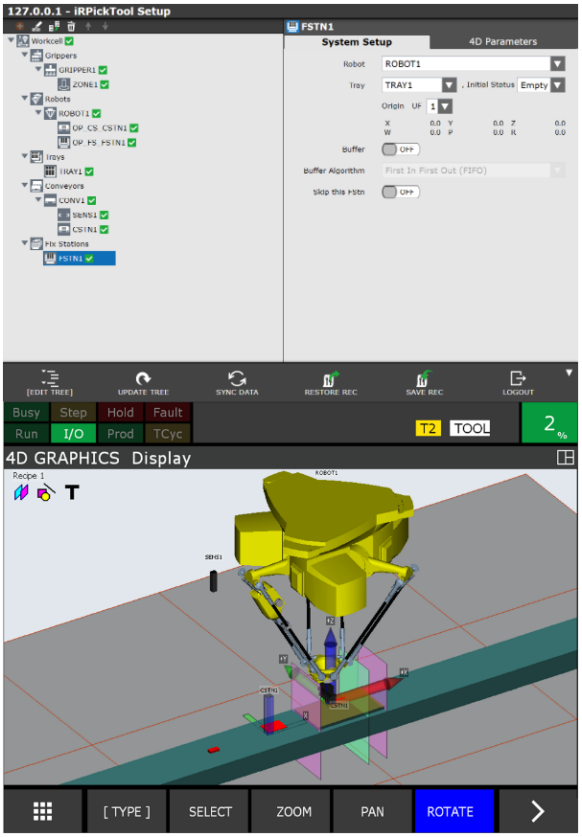
A tray contains one or more cells. The part that is used at each cell location can be selected in the drop-down box.

[Frame Visible]

Enable or disable the visibility of the tray frame box or base using the toggle switch. You will be able to view the names of the trays only when the tray frame is made visible.

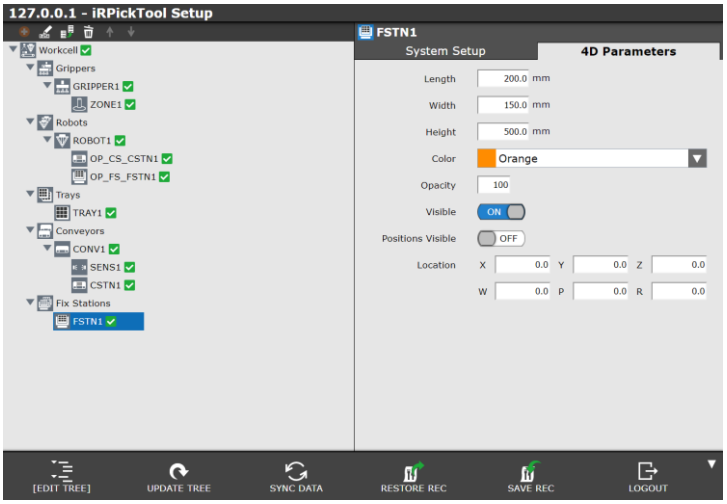
Creating a new 4D Fixed Station

When you create a fixed station in the tree view, 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 on the setting screen accessible from [Fixed Station] in the tree view menu. If the Uframe has not been taught, its value is nilpos which is the same as the robot's world zero position. Therefore, you will see the fixed station running through the robot zero position when you first create it. Later when you teach the Uframe correctly, the fixed station will be located correctly in the 3D workcell.



Modifying the 4D Fixed Station

In the tree view on the left side, press a Fixed Station, such as FSTN1, to setup in 4D. All the 4D Fixed Station properties that you can modify are shown in the right pane ([4D Parameters]). 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. If you want to change the file, refer to “I.8.6.7. Using user-defined object shapes(P.591)”.



[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 of 0 to 100. 0 is completely transparent, and 100 is completely opaque.

[Visible]

Enable or disable the visibility of the fixed station using the toggle switch.

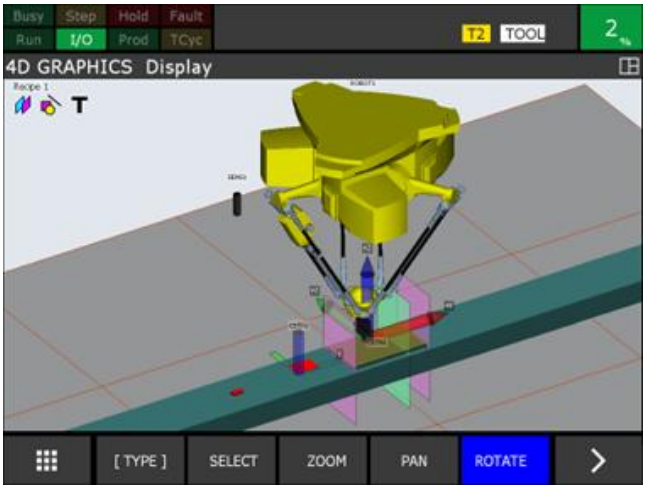
[Positions Visible]

Select or deselect the checkbox to view the approach and pick/place position of the fixed station in the Tray frame. It is displayed correctly only with Plug & Play.

[Position]

The position of the origin of the fixed station from the specified UFrame.

When you teach the Uframe correctly, your 4D display will look similar to the following.



8.6.3. 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 more quickly. In order to make the TP screen enabled for touch, you need to first click on the 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

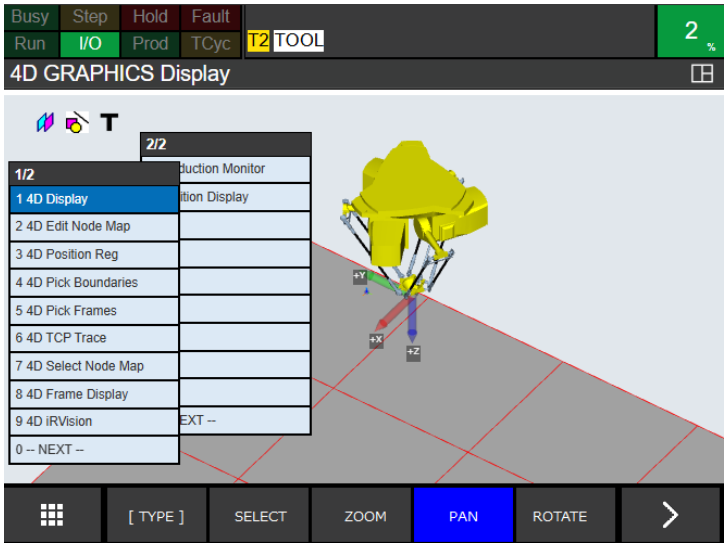
The iRPickTool icon is meant for showing and hiding all the iRPickTool visualizations. You may hide the iRPickTool visualizations when you want to get a clearer view of some other general purpose 4D Graphics function like “Visual Jog”, “4D Edit Node Map”, or “4D Position Reg” or “4D iRVision”.

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. This icon displays/hides them.

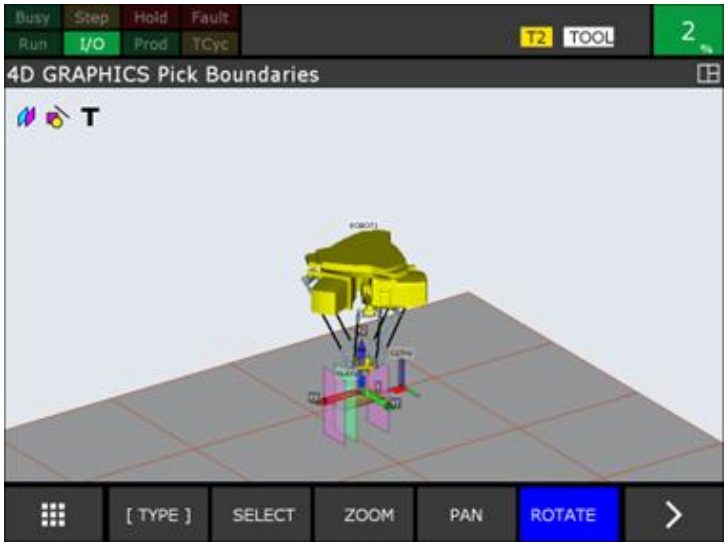
8.6.4. Selective display of specific information

When you press F1 [TYPE] on the [4D Display] screen, the options of [4D pick boundaries] and [4D pick frames] related to iRPickTool will be displayed. When you select these options, only specific information will be displayed for each option. You can explore all the scenes, but the following three scenes may be useful to you check your setup:



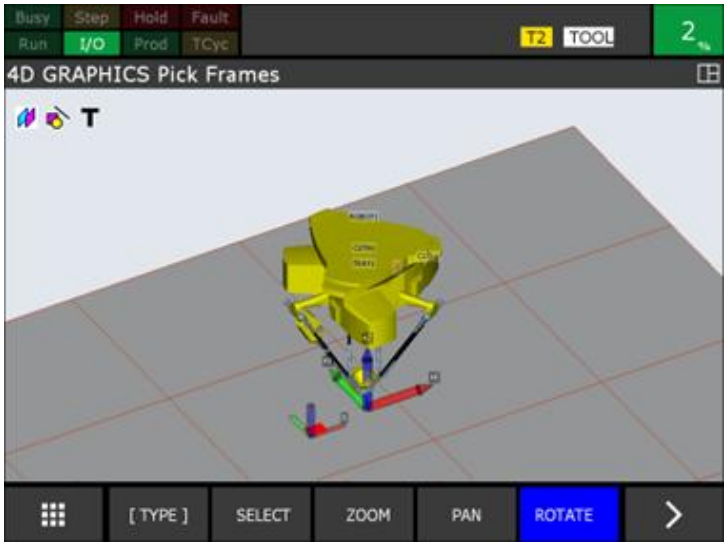
8.6.4.1. 4D pick boundaries

When you press the 4D Pick Boundaries scene, you will see a 3D scene containing only the boundaries and track frames on all the conveyors. This helps you to confirm the boundaries relative to the robots and also the location of your track frames.



8.6.4.2. 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.



8.6.5. Deleting the 4D Workcell

By deleting all objects from the iRPickTool tree view, you can delete the whole 4D Workcell.
You can delete individual objects such as sensors and conveyors from the tree view, but you cannot delete the robot from the 4D graphic display.

If you wish to temporarily turn off the visualizations of the iRPickTool objects you can use the iRPickTool show/hide icon .

8.6.6. 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. The text displayed on the following screen, “Recipe 1,” is the active recipe name.



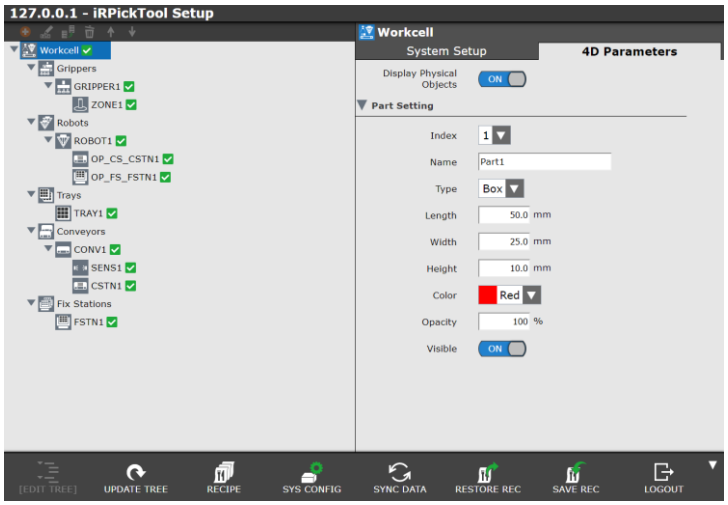
8.6.7. Using user-defined object shapes

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 tree view and then click on the 4D Parameters tab on 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.



8.6.8. Restrictions

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.

8.7. Servo conveyor tracking

8.7.1. Setup

Start the servo conveyor system using the following procedure.

1. **Setting up an index conveyor extended axis**
2. **Setting up an index servo conveyor**
3. **How to create a TP program for index servo conveyor**
4. **Setting up tracking schedules**
5. **Example of a main program**

8.7.1.1. Setting up an extended axis for an index conveyor

Independent extended axes are set up on the ROBOT MAINTENANCE MENU, which is accessible from the controlled start menu. For details of the setup procedure, see “Independent axis setup” in "OPERATOR’S MANUAL(Basic Operation)"(B-84464EN). When adding an extended axis, enter the values for some of the following setup items listed below as described for using extended axis as “Index” servo conveyor. For other items, see "OPERATOR’S MANUAL(Basic Operation)"(B-84464EN) and set up the appropriate values for each system configuration.

Axis type

Set the axis type to [Rotary Axis].

```
-- INDEPENDENT AXES TYPE --

  1. Linear Axis
  2. Rotary Axis

Select? 2
```

Gear ratio

Set [Motor revolutions] (M) and [Axis revolutions] (A) to integer values.

```
--- GEAR RATIO (Rotary axis) ---

Gear Ratio = A : M
  M ... Number of motor revolutions
  A ... Number of axis revolutions
Enter a positive integer value for each.

First, enter M 1000
```

```
--- GEAR RATIO (Rotary axis) ---

Gear Ratio = A : M
  M ... Number of motor revolutions
  A ... Number of axis revolutions
Enter a positive integer value for each.

Enter A 120
```

Set [Axis revolutions] (A) to a value that results in both M and A being integers based on [Rotor Output Teeth] described in "I.8.7.1.2. Setting up an index servo conveyor(P.594)".

For example, if 100 motor revolutions results in 1.2 axis revolutions, set M = 1000 and A = 12.

Maximum axis speed

Set the maximum axis speed to the recommended value. Select [2 : No Change].

```
-- MAX JOINT SPEED SETTING --

Suggested Speed = 3600.000(deg/s)
(Calculated with Max Motor Speed)

Enter (1:Change, 2:No Change)? 2
```

Rotary 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.

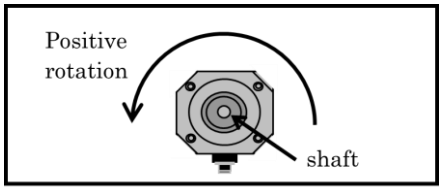


Fig. 8.7.1.1-1 Positive rotation of a motor when viewed from flange front

Movable range

Set the upper limit to 180 (deg) and the lower limit to −180 (deg).

```
-- UPPER LIMIT --
Enter Upper Limit ( deg)? 180

-- LOWER LIMIT --
Enter Lower Limit ( deg)? -180
```

Mastering position

Normally, set the mastering position to 0.

```
-- MASTER POSITION --
Enter Master Position (deg)? 0
```

Acceleration/deceleration time constants

Enter the 1st ACC/DEC time (acc_time1), 2nd ACC/DEC time (acc_time2), and Minimum Accel Time that can reach to the index conveyor speed required for maximum production. The time required for one index conveyor can be calculated by using the formula “operation time + stop time”. The calculation formula for the operation time is “acceleration time + deceleration time”. An acceleration time and deceleration time is the same at a normal operation of index conveyor. Set the acc_time2 to half value of the acc_time1. Minimum Accel Time is the sum of the acc_time1 and the acc_time2.

NOTE

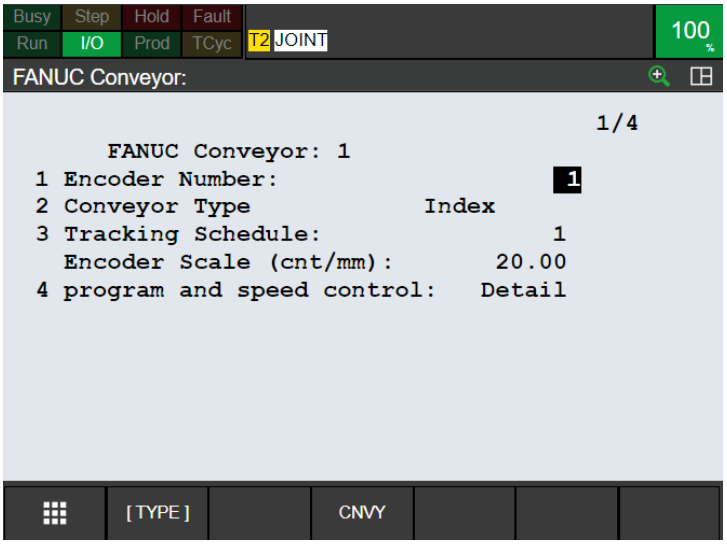
For example, an application with the number of maximum production is 200 counts/min (300ms each) with the stop time is 100ms, operation time of the motor is 200ms/pitch. Therefore, the acceleration time must be less than half of this value (100 ms). In this case, set the acc_time1 to 64 ms, the acc_time2 to 32 ms, and the Minimum Accel Time constant to 96 ms.

8.7.1.2. Setting up an index servo conveyor

When configuring an index servo conveyor, follow the procedure below. The settings configured in this procedure are automatically applied to encoder and continuous turn settings. No additional configuration is required.

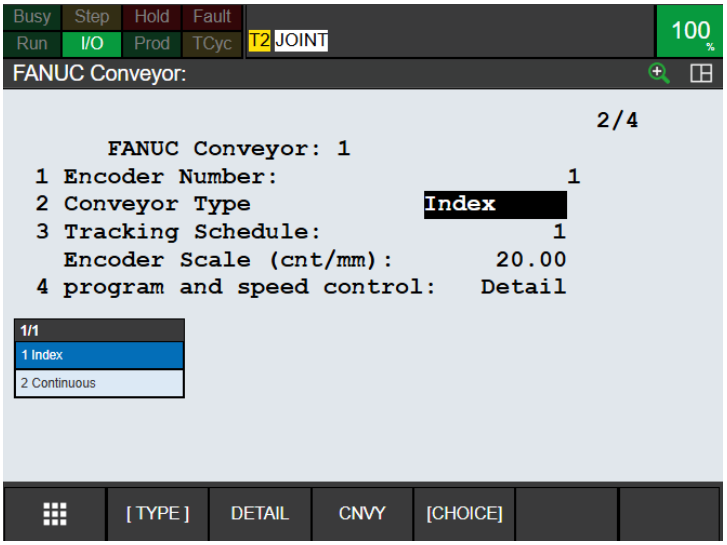
Setting up an index servo conveyor

1. **Press the [MENU] key to display the screen menu.**
2. **Select [SETUP].**
3. **Press F1 [TYPE] to display the screen switch menu.**
4. **Select [Indexers]. The following setting screen appears.**



5 **Set [1 Encoder Number] to the number of the encoder to be configured as a servo conveyor.**

6 **Move the cursor to [2 Conveyor Type], press [F4: CHOICE], and select “Index”.**



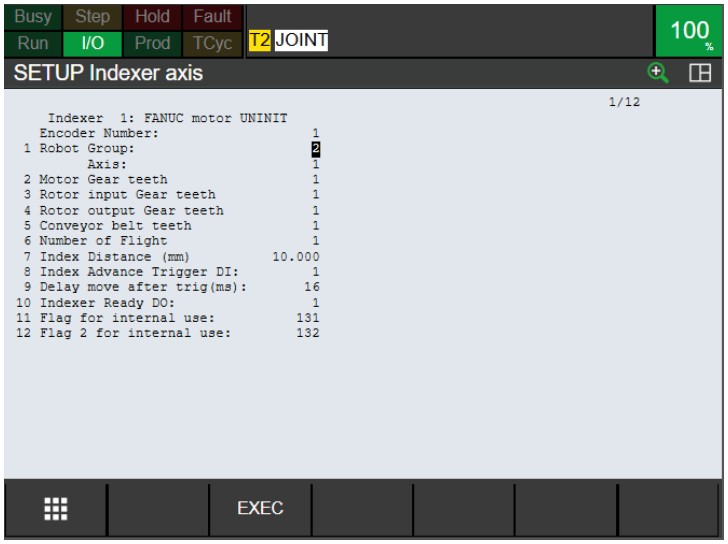
7 **Set [6 Tracking Schedule] to the line tracking schedule that uses the servo conveyor.**

8 **Move the cursor to the [2 Conveyor Type] line and press F2 [DETAIL] or [ENTER] to display the detailed setting screen.**

The screen for setting up axis differs depending on the indexer type.

9 **The following detailed settings screen appears.**

F1 [TYPE] does not appear. Press the [PREV] key to go to another screen.



10 Set [1 Robot Group] to the group number of the additional axis that drives the conveyor and specify the axis number.

11 Set [2 Motor Gear teeth], [3 Rotor input Gear teeth], [4 Rotor output Gear teeth], and [5 Conveyor belt teeth] to integer values.

Set [6 Number of Flights] to the number of buckets on the conveyor. The relationship between these settings is illustrated below. Enter appropriate values to configure the gear ratio of the servo conveyor.

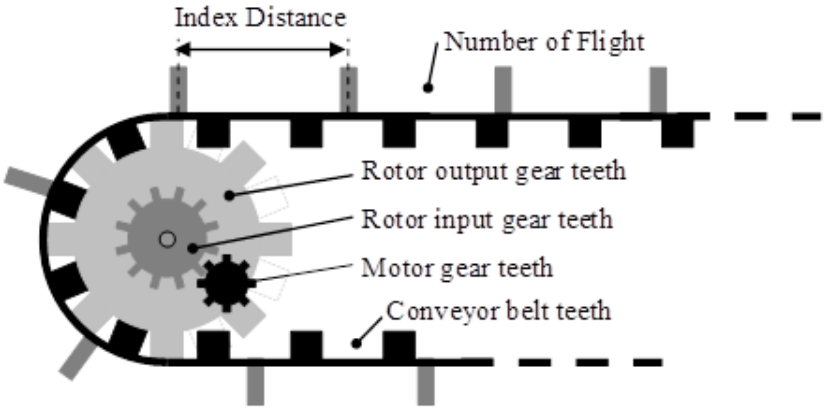


Fig. 8.7.1.2-1 Relationship between the motor, rotor, and conveyor

* If the conveyor rotates N times when the motor rotates M times, the following settings can also be used based on the rotary relationship between the motor and the conveyor. Here, M and N are integers.

- Set [2 Motor Gear teeth] to “N”.
- Set [3 Rotor input Gear teeth] to “M”.
- Set [4 Rotor output Gear teeth] and [5 Conveyor belt teeth] to “1”.

12 Set [7 Index Distance (mm)] to the distance the servo conveyor moves for one pitch.

13 Set [8 Index Advance Trigger DI] to the DI number used to drive the servo conveyor.

14 Set [9 Delay move after trig (ms)] to the delay time between the servo conveyor drive DI trigger and the start of conveyor motion.

15 Set [10 Indexer Ready DO] to the DO number that is output when a servo conveyor TP program is executed.

See INDXG*.TP, the servo conveyor TP program described below.

16 Set [11 Flag for internal use] to the flag number used for the servo conveyor. (Verify that the specified flag is not used for other purposes.)

17 Set [12 Flag2 for internal use] to the flag number used for the servo conveyor. (Verify that the specified flag is not used for other purposes.)

18 After completing this setup procedure, press F2 [EXEC] to cycle the power.

NOTE

By the above setting, Continuous Rotation setup of the conveyor and Encoder setup of the specified encoder are also done.

8.7.1.3. How to create a TP program for index servo conveyor

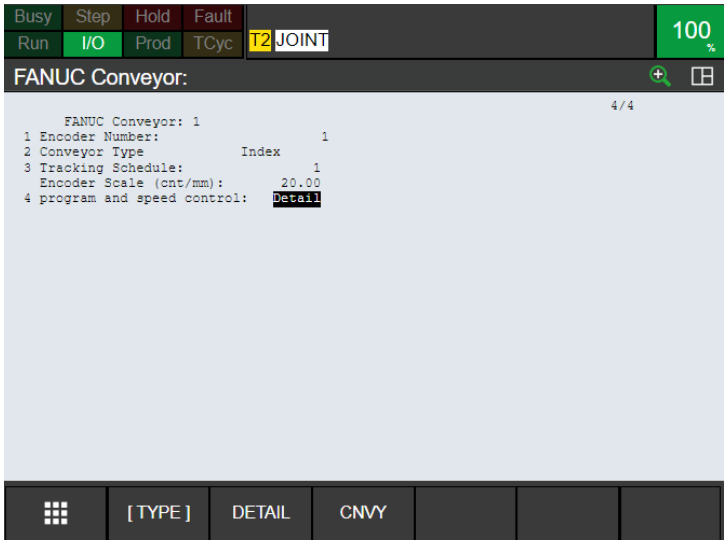
The servo conveyor is configured as an extended axis and therefore requires a TP program to operate it. Use the procedure below to generate a TP program for servo conveyor operation.

Generating a servo conveyor TP program

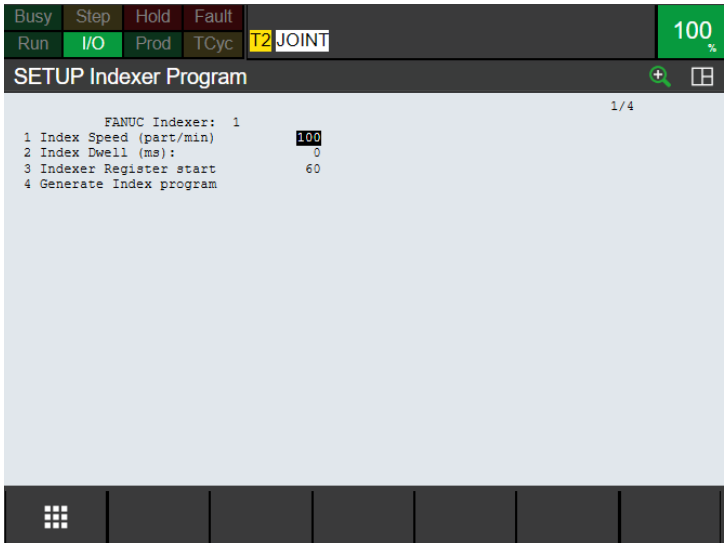
NOTE

Before generating a TP program that drives the servo conveyor, configure the tracking payload on the MOTION/PAYLOAD SET menu and switch to that payload.

1 On the servo conveyor setting screen, move the cursor to [program and speed control] and press F2 [DETAIL] or [ENTER].



The following screen appears. F1 [TYPE] does not appear. Press the [PREV] key to go to another screen.



2 Set [1 Index Speed (part/min)] to the servo conveyor speed as an integer.

Specify the number of pitches moved per minute.

3 **Set [2 Index Dwell (ms)] to the value of the duration of time the servo conveyor stops as an integer.**

The time required for one conveyor pitch to move is calculated based on the servo conveyor speed and stop time.

Move time per pitch [ms] = (60 * 1000 / servo conveyor speed) − servo conveyor stop time

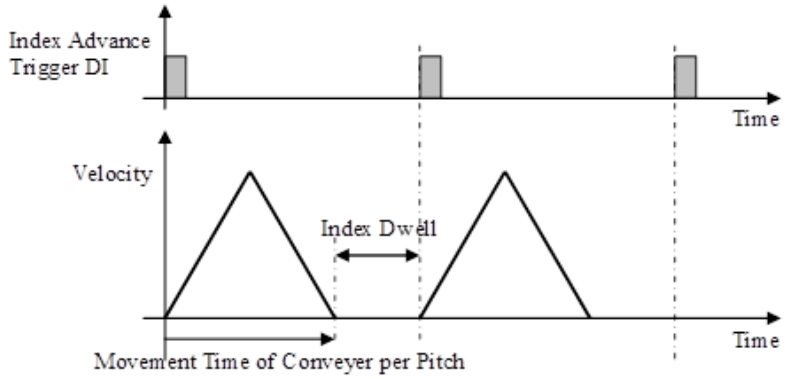


Fig. 8.7.1.3-1 Speed pattern

4 **Set [3 Indexer Register start] to the starting register number used in the TP program.**

TP programs use two registers.

For example, if “60” is specified, R[60] and R[61] are used. R[60] is used to specify the DO number used to verify that the conveyor has returned to the initial position. R[61] is used to determine the conveyor pitch. See the sample program on the following page.

5 **After completing the previous steps, move the cursor to [4 Generate Index Program] and press F2 [Create].**

If generation is successful, the message “Done” is displayed and INDXG*.TP is generated (“*” indicates the servo conveyor group number. If the servo conveyor is in the second group, the program name is INDXG2.TP).

Restrictions

- INDXG*.TP is the default system program. Do not change it directly. Always rename it before use.
- The message “Part Rate exceeds allowable value” may be displayed depending on the settings. In this case, the acceleration calculated from the servo conveyor settings exceeds the allowable acceleration of the conveyor or robot, and the TP program cannot be generated. Modify the servo conveyor speed or stop time.
- If INDXG*.TP already exists and needs to be changed, hold the [SHIFT] key and press F3 [REPLACE]. If successful, the message “Done” is displayed, and the existing INDXG*.TP is overwritten with a TP program generated using the current settings.
  - If only F3 [REPLACE] is pressed, the message “Press SHIFT + CHANGE to replace the program” is displayed. The TP program is not overwritten. Hold the [SHIFT] key and press F3 [CHANGE].
  - If INDXG*.TP is selected, it cannot be updated. The message “Processing failed” is displayed.
  - Depending on the settings, the message “Part Rate exceed allowable value” may be displayed. The TP program is not overwritten. Modify the servo conveyor speed or stop time.

Sample program

The code below is an example of INDXG*.TP, an automatically generated TP program.

Example :INDXG*.TP

```
1: J P[1] 50% CNT0 ; Move to the initial position
2: IF R[60:G2 Ready DO]=0,JMP LBL[3] ;
3: DO[R[60]]=ON ; Enable the DO specified by R[60]
4: LBL[3] ;
5: LBL[61:G2 cur slot ID]=0 ; Reset R[61] to “0”
6: LBL[1] ;
```

```
 7: J P[2] 180msec CNT100 INC INDXMV ; Moves the conveyor by one pitch when the servo DI input is on
 8: R[61:G2 cur slot ID]=R[61:G2 cur slot ID]+1 ; Counts conveyor operations
 9: IF R[61:G2 cur slot ID]=32,JMP LBL[2] ; Jump to label    [2] if conveyor is turned once
10: JMP LBL[1] ;
11: LBL[2] ;
12: J P[1] 100% CNT100 ; Error correction motion
13: R[61:G2 cur slot ID]=0 ;Reset R[61] to “0”
14: 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
};
```

- The motion statement on line 1 moves the conveyor to its initial position.
- Lines 2 through 4 output a DO to verify that the conveyor has moved to its initial position.
- The motion statement on line 7 line moves the conveyor by one pitch. The additional axis rotates 360 degrees.
  The “INC INDXMV” option enables monitoring of input from the DI configured for the servo.
- The “IF” on line 9 checks whether the conveyor makes one revolution. In this example, the conveyor completes a full rotation after 32 pitches.
- The motion statement on line 12 corrects for errors caused by conveyor motion and is executed once per conveyor revolution.
- R[60] and R[61] are assigned based on the servo conveyor register start number. R[60] is used to specify the DO number used to verify that the conveyor has returned to the initial position. R[61] is used to determine the conveyor pitch.

8.7.1.4. Setting up tracking schedules

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. For details, refer to “I.2.3.8. Setting up a conveyor station(P.106)” or “I.3.3.9. Setting up a conveyor station(P.220)”.

8.7.1.5. Example of creating a main program

The main program is almost the same as when a servo conveyor is not used. The difference from not using servo conveyor is that it is necessary to run TP program for moving a servo conveyor by multitask.

In the main program example below, the Run command on line 4 starts CONVEYOR.TP (renamed from INDXG*.TP), which is the servo conveyor TP program.

Sample main program

```
 1: R[1:CYCLE_STOP]=0
 2: CALL PKCSGETID('CONV1',CStn ID Reg=11)
 3: CALL PKCSGETID('CONV2',CStn ID Reg=12)
 4: Run CONVEYOR
 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 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
```

8.7.2. Servo conveyor function

8.7.2.1. Servo conveyor error adjustment parameters

For the servo conveyor tracking function, dynamic errors of tracking motion are adjusted using the procedure below, rather than by setting [Dynamic Error Adjustment], which is described in "I.5.2. Setting up a robot(P.268)".

1. **Press the [MENU] key on the teach pendant.**
2. **Select [SYSTEM] → [CONFIG] from the menu.**
3. **Move the cursor to [<DETAIL>] for [Line/Cir/Rail Tracking] and press F2 [DETAIL].**
4. **Move the cursor to [<DETAIL>] for [Servo Conveyor] and press F2 [DETAIL].**
5. **Move the cursor to [<DETAIL>] for [Dynamic Tune Variable(ms)] and press F2 [DETAIL].**
6. **Adjust the servo conveyor group number value for each servo conveyor.**

For details on the adjustment procedure, refer to the "High-speed tracking adjustment" section in "Line Tracking OPERATOR'S MANUAL"(B-84464EN-10). This setting can be adjusted on a per-conveyor basis.

NOTE

If a servo conveyor-specific KAREL program is used to configure the trigger value, Static Error (Line/Cir/Rail Tracking -> Basic function -> Static Tune Variable (ms)) does not need to be adjusted.

8.7.2.2. Servo conveyor stop-wait function

By default, this function is disabled. When this function is disabled, a robot starts/ends a tracking regardless of the motion of the servo conveyor. Therefore, for example, if a robot starts a tracking motion while a servo conveyor is accelerating, the acceleration of the robot is increased in order to catch up with 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

To enable this function, set the index dwell to a value greater than 0 and generate a servo conveyor TP program. If the servo conveyor never stops, tracking cannot be started or ended.

To enable or disable this function, perform the procedure below. The function can be enabled or disabled separately for tracking motion start and tracking motion end on a per-schedule-number basis.

For example, schedule number 1 can be configured to wait for the servo conveyor to stop when tracking motion starts, but not when tracking motion ends.

1. **Press the [MENU] key on the teach pendant.**
2. **Select [SYSTEM] → [CONFIG] from the menu.**
3. **Move the cursor to [<DETAIL>] for [Line/Cir/Rail Tracking] and press F2 [DETAIL].**
4. **Move the cursor to [<DETAIL>] for [Servo Conveyor] and press F2 [DETAIL].**
5. **Move the cursor to [<DETAIL>] for [Wait Indexer Stop (start)] or [Wait Indexer Stop (end)] and press F2 [DETAIL].**
6. **When enabling this function on a per-schedule-number basis, press F4 [TRUE]. When disabling it, press F5 [FALSE]**
7. **After completing the configuration, reboot the robot controller.**

NOTE

After changing this function setting, regenerate the servo conveyor drive TP program.

8.7.2.3. Pick or Drop Multiple

This function allows multiple workpieces to be picked or placed on a servo conveyor at one time. It reduces the number of pick-and-place motions for workpieces, enabling the robot to handle a larger transport quantity.

Enabling the Pick or Drop Multiple function does not require any modifications to TP programs.

⚠CAUTION

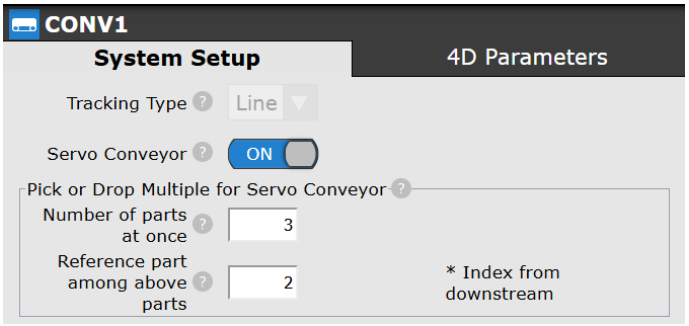
The following functions and objects cannot be used in combination with the Pick or Drop Multiple function. (This function is enabled when “Number of parts at once” is set to 2 or higher.)

- Tray
- Sort function (Y, measurement value 1)
- Stop Conveyor function
- Multi zone function of *i*RPickTool (J945)

Setting up the Pick or Drop Multiple function

On the conveyor setup screen, enter the number of workpieces to be picked or placed at one time and specify the reference workpiece.

1. **On the conveyor setup screen, enter the number of workpieces to be picked or dropped at one time and specify the reference workpiece.**



- [Number of parts at once]

  Enter the number of workpieces to be picked or dropped at one time.

- [Reference part among above parts]

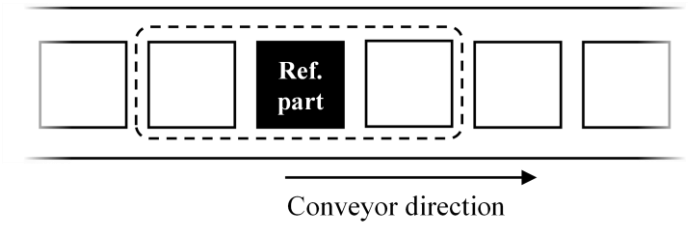
  Specify the position of the reference workpiece that is selected from among the workpieces to be picked or dropped (“reference workpiece”). The reference workpiece is used to distribute or ignore other workpieces. For details, see “Distribution” and “Discard” below.

  When specifying the reference workpiece position, count from the downstream side.

  - When selecting the middle workpiece from three workpieces as the reference workpiece

    [Number of parts at once]: 3

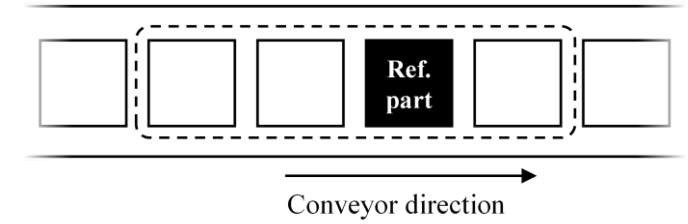
    [Reference part among above parts]: 2



- When selecting the second workpiece (counting from downstream) from four workpieces as the reference workpiece

  [Number of parts at once]: 4

  [Reference part among above parts]: 2



Setting the target position

Place one workpiece on the conveyor.

Set a trigger so that this workpiece is used as the reference workpiece for multiple workpieces to be picked or dropped at one time.

1 **Move the conveyor to position the reference workpiece within the tracking boundary and teach that position.**

For details on the position teaching procedure, see “I.5.5.6. Setting the target position(P.312)”.

Picking or dropping multiple motion

The taught workpiece serves as the reference for picking or placing multiple workpieces at one time.

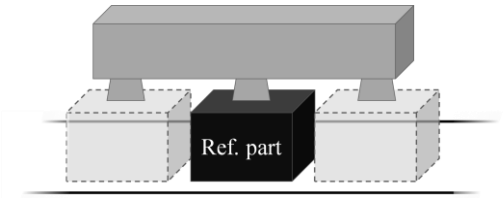


Fig. 8.7.2.3-1 <Three workpieces picked with the second workpiece set as the reference workpiece>

Distribution

When the reference workpiece passed the upstream tracking boundary, PKCSGETQUE distributes the information for the workpieces to be picked or dropped at one time.

Discard

When the reference workpiece moves downstream beyond the discard line by the number of pitches specified as [”Reference part among above parts” − 1], it is allowed to pass, and the corresponding workpiece information is handed over to the downstream conveyor station.

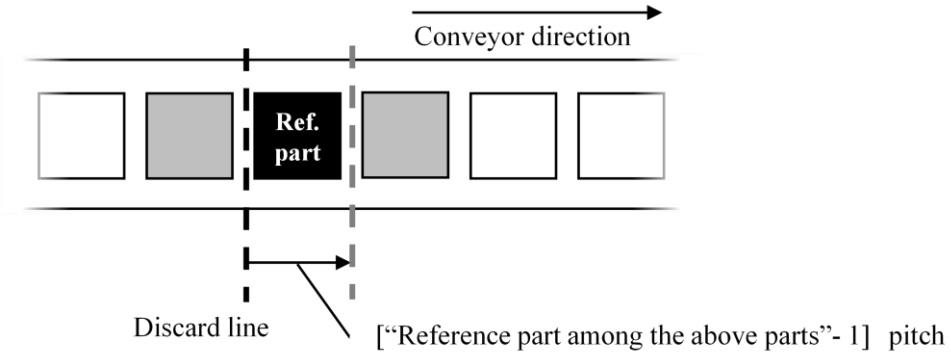


Fig. 8.7.2.3-2 <Three workpieces picked with the second workpiece set as the reference workpiece>

NOTE
When PKCSACKQUE is called, the workpieces picked or dropped at one time are treated as already picked, even if some of them have not passed the upstream tracking boundary.

8.7.2.4. KAREL programs for servo conveyor Line Tracking

For servo conveyors, the following KAREL programs can be used.

NOTE
KAREL programs provided by iRPickTool can also be used.

Use these programs to retrieve trigger values at regular motion intervals without using sensors and to save multiple trigger values. It is also possible to use Discard Line function and Stop Conveyor function and Distribution Line function.

- SLTKINIT
- SLTKREST
- SLTKPSHQ
- SLTKPOPQ
- SLTKDELQ
- SLTKRSTQ
- SLTKGTPP

Each KAREL program is described below. The Discard Line, Stop Conveyor, and Distribution Line functions are described afterward.

NOTE
Functions related to the KAREL programs for servo conveyors cannot be used in combination with iRPickTool functions. When using iRPickTool functions, use the KAREL programs provided by iRPickTool instead of the KAREL programs for servo conveyors.

SLTKINIT

This program is used to initialize trigger value information. Normally, this program is called once when the system is started.

Argument 1:	Specify a number of tracking schedule which Servo Conveyor Line Tracking uses.
Argument 2:	Specify the Position Register number that is saved an initial position of the conveyor.

Argument 3: Specify the position register number.

The distance between the current conveyor position and initial position (value in the position register specified by Argument 2) is set in the position register specified by Argument 3. The stored value does not exceed one pitch (within the 0-360° range). This can be used to align the conveyor position when motion starts.

SLTKREST

Servo conveyors have a fixed pitch-to-pitch distance, which allows encoder values to be retrieved at regular intervals as trigger values. This program is used to set the trigger reference value. 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.

Argument 1: Specify a number of tracking schedule which Servo Conveyor Line Tracking uses.

In the example below, SLTKINIT sets PR[2] to the distance between the current conveyor position and its initial position. The conveyor is moved to the initial position by Joint Motion. SLTKREST sets the trigger reference value.

```
1: CALL SLTKINIT (1,1,2) ;          Initialization and position register setup
2: J PR[2] 50% CNT0 INC     ;       Motion to the initial position
3: CALL SLTKREST (1) ;              Setting the trigger reference value
```

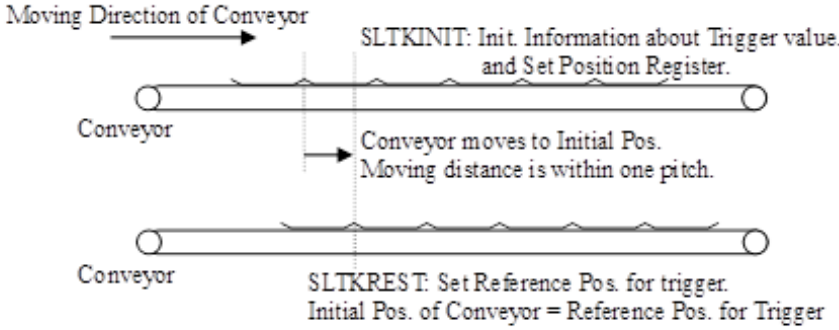


Fig. 8.7.2.4-1 SLTKINIT, SLTKREST

SLTKPSHQ

This program is used to save trigger values after the conveyor moves by a certain number of pitches.

Argument 1: Specify a number of tracking schedule which Servo Conveyor Line Tracking uses.

Argument 2: Specify a number of pitch.

The number of trigger values that can be saved is configured using the following setting. After changing this setting, the power needs to be cycled.

[Queue Size of Trigger]: Default 20, Maximum: 100

n indicates the schedule number that uses the servo conveyor

NOTE

When the number of saved trigger values reaches the value specified by [Queue Size of Trigger], the oldest saved trigger value is deleted.

When executed, SLTKPSHQ calculates a trigger value based on the trigger reference value set by SLTKREST and saves it. When SLTKPSHQ is executed for the first time, the trigger reference value itself is saved as a trigger value. Normally, this program is executed after the conveyor moves by the specified number of pitches.

For example, to feed one pitch and save one trigger value, specify “1” for Argument 2. To feed two pitches and save one trigger value, specify “2” for Argument 2. In the figure below, the conveyor feeds four pitches while trigger values are saved.

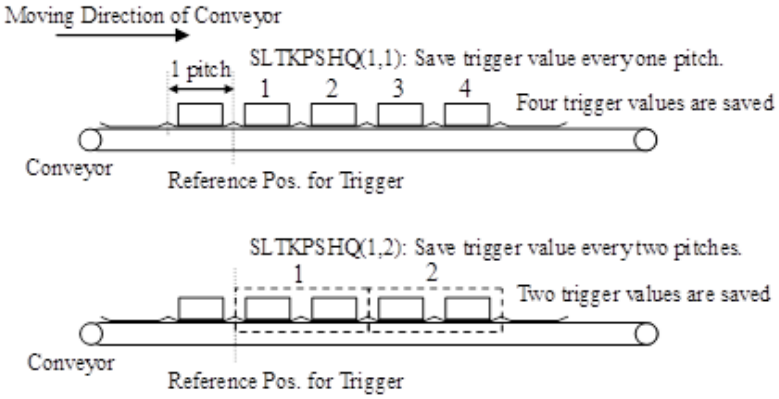


Fig. 8.7.2.4-2 SLTKPSHQ

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. The retrieved trigger value is deleted and cannot be retrieved again.

Argument 1:	Specify a number of tracking schedule which Servo Conveyor Line Tracking uses.
Argument 2:	Specify a register number. The register with the specified number stores the trigger value.
Argument 3:	Specify a register number. The register with the specified number stores the status. If SLTKPOPQ fails to get a trigger value, a value other than 0 is set.

In the example below, the queue contains four saved trigger values.

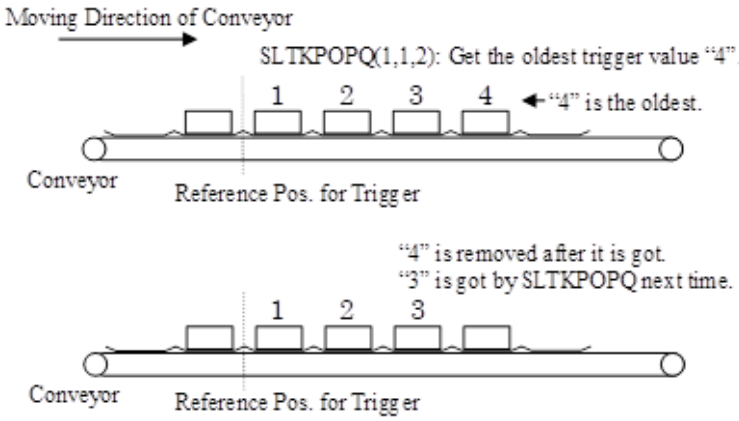


Fig. 8.7.2.4-3 SLTKPOPQ

SLTKDELQ

This KAREL program is used to delete a saved trigger value with the specified number. Once deleted, the trigger value can no longer be retrieved using SLTKPOPQ.

Argument 1:	Specify a number of tracking schedule which Servo Conveyor Line Tracking uses.

Argument 2: Specify the number of the trigger value to be deleted.

For example, if argument2 is “2”, SLTKDELQ deletes the second trigger value “2” from the newest trigger value.

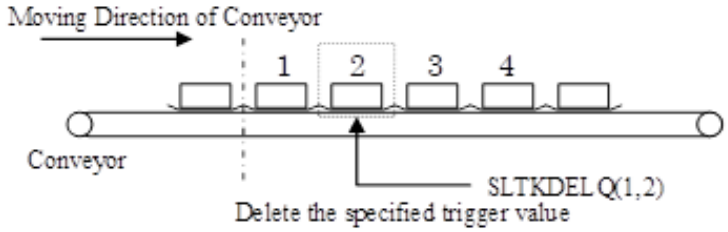


Fig. 8.7.2.4-4 SLTKDELQ

NOTE

The message “Trigger count is not existed” is displayed when there is no trigger value specified.

SLTKRSTQ

This program turns off the specified DO for Stop Conveyor. For details, refer to the section describing Stop Conveyor. See the Stop Conveyor section as well.

Argument 1: 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.

Argument 1: Specify a number of tracking schedule which Servo Conveyor Line Tracking uses.

Argument 2: Specify a register number. The calculated current X-axis position in the tracking frame of the trigger value is set.

Argument 3: Specify a register number. The register with the specified number stores the status. If position retrieval is successful, 0 is set. If it fails, a non-zero value is set. In the event of failure, the value in the register specified by Argument 2 is not updated.

For example, in the figure below, executing SLTKGTPP calculates the position of the fourth trigger value, which is the oldest.

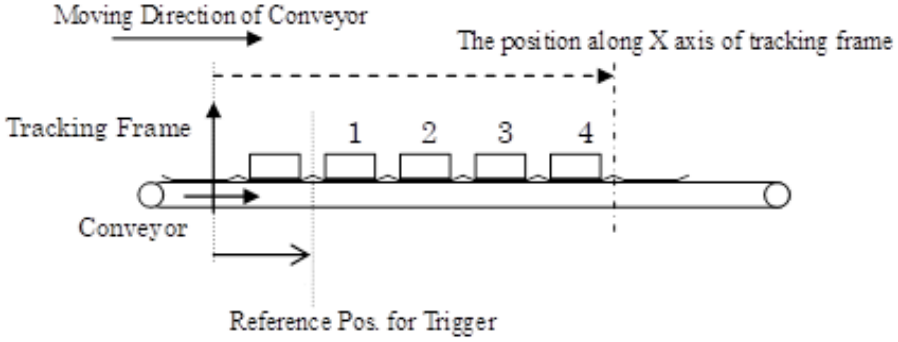


Fig. 8.7.2.4-5 SLTKGTPP

This program uses the oldest retrievable trigger value to calculate its position. For example, if SLTKDELQ has deleted “4”, the position corresponding to the next trigger “3” is calculated.

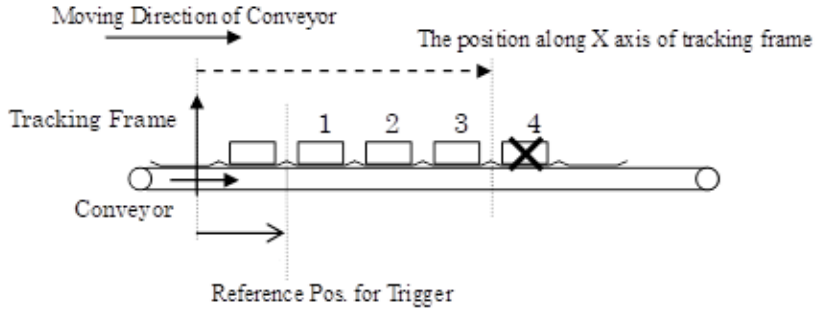


Fig. 8.7.2.4-6 SLTKGTPP and SLTKDELQ

When the discard line is configured and “4” passes the line, SLTKPOPQ can no longer retrieve “4”. In this case, the position corresponding to trigger value “3” (the next pitch) is calculated. See the section describing Discard Line.

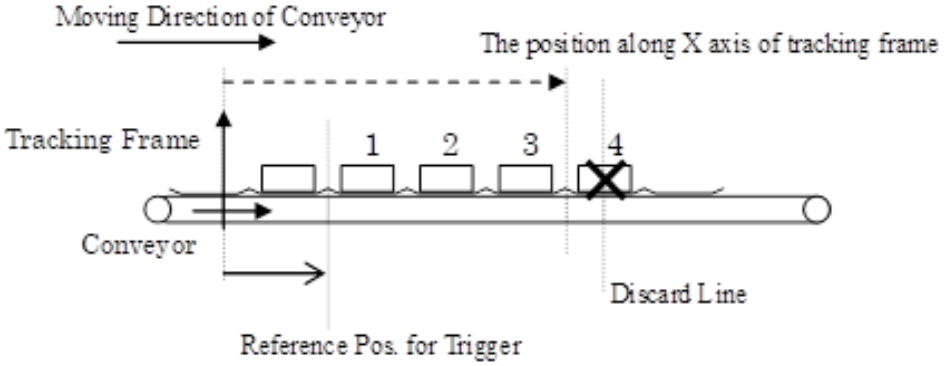


Fig. 8.7.2.4-7 SLTKGTPP and the Discard Line

If the stop conveyor is configured, and “4” passes the discard line, a DO is output to prompt the system to stop the conveyor. When the stop conveyor is used, trigger value “4” can still be retrieved using SLTKPOPQ, even after it passes the discard line. The position corresponding to trigger value “4” is calculated. See the Stop Conveyor section as well.

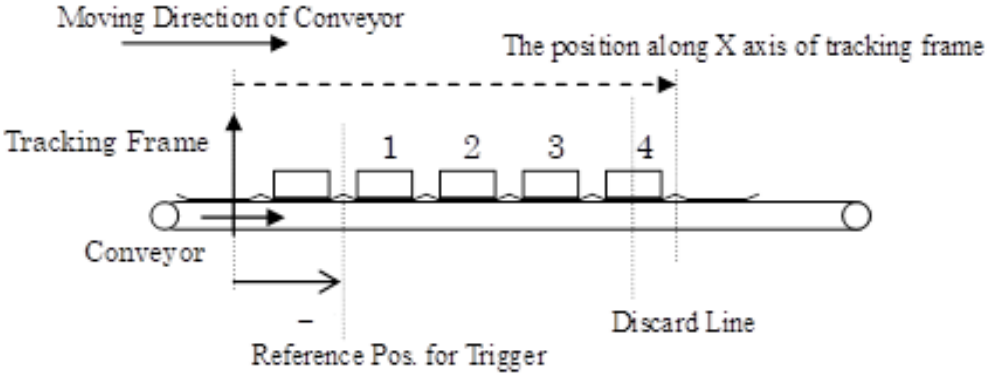


Fig. 8.7.2.4-8 SLTKGTPP and the Stop Conveyor

Discard Line

A trigger value can be discarded when its corresponding position passes the discard line. It is necessary to set Discard Line on the basis of downstream boundary of the tracking area.

To use this function, open [MENU] and navigate to [SYSTEM] → [CONFIG] → [Line/Circ/Rail Tracking] → [Servo Conveyor], then configure the following items in the detailed servo conveyor tracking settings.

[Enable Discard Line]	Enable
[Discard Line(mm)]	Position of the discard line within the tracking boundary, measured from the downstream end [mm]

[Reference Pos for Trigger] X-axis position of the trigger reference in the tracking frame [mm]

* These settings must be configured for each line tracking schedule number.

If the [Discard Line(mm)] value is negative, the discard line is positioned upstream from the downstream end. Normally, a negative value is specified.

Once the position corresponding to a trigger value crosses the discard line, that trigger value can no longer be retrieved using SLTKPOPQ. The oldest retained trigger value is used to determine whether it should be discarded.

In the example below, the position corresponding to trigger value “4” crosses the discard line and “4” is discarded. After discarding “4”, the system proceeds to determine whether the position corresponding to trigger value “3” should be discarded.

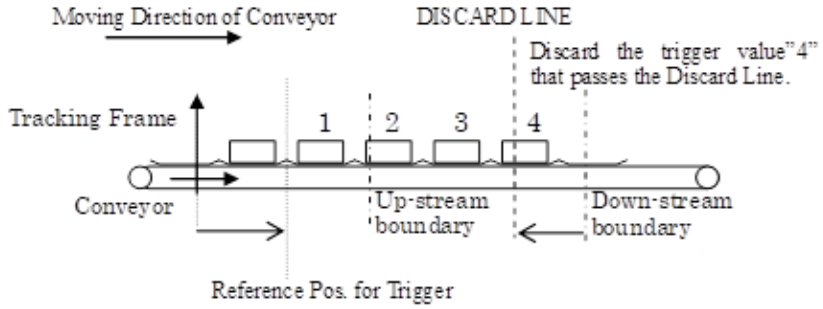


Fig. 8.7.2.4-9 Discard Line

Stop conveyor

If the stop conveyor is enabled, the DO is turned on when the position corresponding to a trigger value passes the Discard Line. This signal should be used to block the input of the servo conveyor drive DI. The Discard Line is configured to ensure that the desired tracking motion can be activated at the position where the conveyor stops.

To use this function, open [MENU] and navigate to [SYSTEM] → [CONFIG] → [Line/Circ/Rail Tracking] → [Servo Conveyor], then configure the following items in the detailed servo conveyor tracking settings.

[Enable Discard Line]	Enable
[Discard Line(mm)]	Position of the discard line within the tracking boundary, measured from the downstream end [mm]
[Reference Pos for Trigger]	X-axis position of the trigger reference in the tracking frame [mm]
[Enable Stop Conveyor]	Enable
[Stop Conveyor DO[]]	DO number to be used

* These settings must be configured for each line tracking schedule number.

When the Stop Conveyor is used, the trigger value used for Stop Conveyor judgment is retained even after it passes the discard line. While the DO remains on (until SLTKRSTQ is executed), that trigger value can be retrieved using SLTKPOPQ. Always use SLTKRSTQ to turn off the DO. After SLTKRSTQ is executed, the next trigger value is used for Stop Conveyor judgment.

In the example below, the position corresponding to trigger value “4” passes the discard line, and the specified DO is turned on. The conveyor should be stopped accordingly. Trigger value “4” can be retrieved using SLTKPOPQ. After processing at the position corresponding to trigger value “4” is complete, execute SLTKRSTQ to turn off the specified DO. The conveyor motion should be resumed accordingly. After SLTKRSTQ is executed, Stop Conveyor judgment is performed for the position corresponding to trigger value “3”.

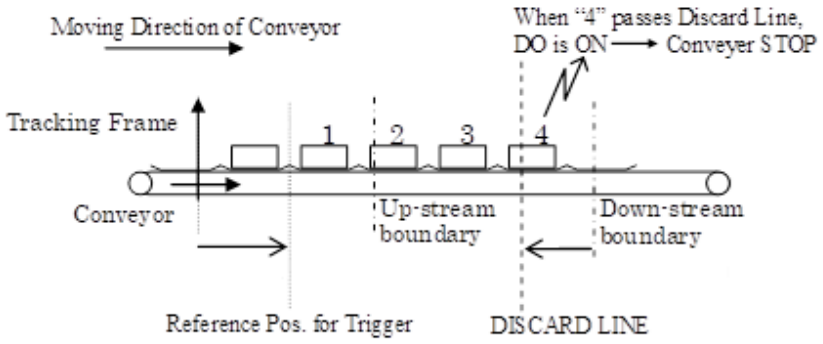


Fig. 8.7.2.4-10 Stop conveyor

NOTE

The discard line and the Stop Conveyor cannot be used simultaneously on the same servo conveyor.

Distribution start line

Until the position corresponding to a trigger value crosses the distribution start line, that trigger value cannot be retrieved using SLTKPOPQ. The distribution start line is configured with reference to the upstream end of the tracking boundary.

To use this function, open [MENU] and navigate to [SYSTEM] → [CONFIG] → [Line/Circ/Rail Tracking] → [Servo Conveyor], then configure the following items in the detailed servo conveyor tracking settings.

[Enable Distribution Line]	Enable or disable the distribution start line settings. The default value is disable.
[Distribution Line(mm)]	Position of the distribution start line within the tracking boundary, measured from the upstream end [mm]
[Reference Pos for Trigger]	X-axis position of the trigger reference in the tracking frame [mm]

* These settings must be configured for each line tracking schedule number.

If the [Distribution Line(mm)] value is negative, the distribution line is positioned upstream from the upstream end of the tracking boundary. Normally, a negative value is specified. Until the position corresponding to a trigger value passes the distribution line, that trigger value cannot be retrieved using SLTKPOPQ.

For example, when the position corresponding to trigger value "4" passes the distribution start line, the trigger value can be retrieved using SLTKPOPQ.

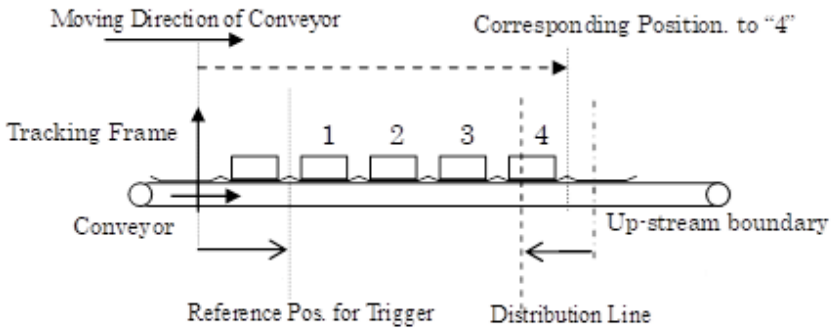


Fig. 8.7.2.4-11 Distribution line

Sample program

A sample main program that uses the Stop Conveyor and a sample servo conveyor drive program (CONVEYOR.TP) are provided below.

- These programs use line tracking schedule "1".

- The initial conveyor position is saved in PR[1].
- The trigger value is saved every time the conveyor moves 2 pitches.

At the beginning of the main program, trigger value information is initialized using SLTKINIT. The conveyor drive program is then executed.

Sample main program

```
 1: J P[1] 100% FINE      ;
 2:   Enc[1] =ON ;
 3:   CALL SLTKINIT(1,1,2);          Initializes trigger value information. Sets PR[2].
 4:   Run CONVEYOR ;                 Execute the conveyor drive program
 5:   LBL[1] ;
 6:   WAIT      .10(sec) ;
 7:   CALL SLTKPOPQ（1,1,2）;        Retrieves the trigger value
 8:   IF R[2]<>0,JMP LBL[1] ;        If the trigger value cannot be retrieved, jump to LBL[1]
 9:   Trigger setting tracking[1] R[1] ;   Sets R[1] as the trigger value
10:   CALL TRAK ;                    Calls the tracking program
11:   CALL SLTKRSTQ(1);              Turns off the Stop Conveyor DO
12:   CALL NORM ;                    Calls the normal motion program
13:   JMP LBL[1] ;
```

- On line 3, SLTKINIT is executed to initialize trigger value information. PR[2] is set to the distance the conveyor moves from the current position to the initial position specified by PR[1].
- On line 4, the conveyor drive program is executed using “run”.
- On lines 5 through 8, trigger values are retrieved every 100 [ms] using SLTKPOPQ. On line 8, R[2] is checked to determine whether a trigger value has been retrieved. If not, execution returns to line 5.
- On line 11, if the Stop Conveyor DO remains on, it is turned off.

In the servo conveyor drive program, trigger values are retrieved using SLTKPSHQ every two conveyor movements. R[3] is used to count conveyor operations and SLTKPSHQ is executed based on the value of R[3].

Sample CONVEYOR.TP

```
 1: J PR[2] 50% CNT0 INC      ;   Motion to the initial position
 2:   CALL SLTKREST(1) ;          Sets the trigger reference value
 3:   IF R[60:G2 Ready DO]=0,JMP LBL[3] ;
 4:   DO[R[60]]=ON ;              Turns on the DO specified by R[60]
 5:   LBL[3] ;
 6:   R[61:G2 cur slot ID]=0 ;    Resets R[61] to “0”
 7:   R[3]=0 ;                    R[3] is a counter for SLTKPSHQ
 8:   LBL[1] ;
 9: J P[2] 180msec CNT100 INC INDXMV ;   Motion for one pitch
10:   R[61:G2 cur slot ID]=R[61:G2 cur slot ID]+1 ;   Counts conveyor operations
11:   R[3]=R[3]+1 ;               Counts conveyor operations
12:   IF R[61:G2 cur slot ID]=32,JMP LBL[2] ;
13:   JMP LBL[4] ;                Jump to line 19 (LBL[4])
14:   LBL[2] ;
15: J PR[1] 100% CNT100 ;        Motion to correct errors
16:   R[61:G2 cur slot ID]=0 ;    Resets R[61] to “0”
17:   LBL[4] ;
18:   IF R[3]<2,JMP LBL[1];      If the conveyor has not moved twice, jump to LBL[1]
19:   CALL SLTKPSHQ(1,2)          If the conveyor has moved twice, save the trigger value
20:   R[3]=0 ;                    Resets R[3]
21:   JMP LBL[1] ;
/POS
```

- PR[2] is set to the distance the conveyor moves from the current position to the initial position specified by PR[1]. Accordingly, line 1 moves the conveyor from the current position to the position specified by PR[1].

- On line 2, SLTKREST sets the trigger reference value.
- On lines 7, 11, and 20, the counter (R[3]) is initialized, incremented, and reset, respectively.
- On line 15, the conveyor moves to the position specified by PR[1] to correct errors that occur during a full rotation.
- On line 18, the number of conveyor operations is checked. If the value is 2, SLTKPSHQ is executed to save the trigger value.
- To change the number of conveyor operations for each trigger, the value “2” on lines 18 and 19 needs to be changed to the desired number.

8.7.3. Servo conveyor tracking restrictions

- It cannot be used at the same time as the High Accuracy Line Tracking function (R884).
- iRPickTool Basic (J944) or iRPickTool (J945) must be ordered at the same time the Constant Path (R663). The Constant Path (R663) is included in the FRL Params (R651).
- It is necessary to separate a robot and an extended axis as a conveyor into different groups.
- Multiple servo conveyors are supported. Up to four axes can be added (Group 1: robot axes, Groups 2-5: conveyor axes).
- Servo Conveyor Line Tracking function can be used together with a traditional encoder conveyor.
- Servo Conveyor Line Tracking function supports HDI and ACCUTRIG instruction.
- It can be used in combination with visual tracking. Visual tracking cannot be used in conjunction with a servo-driven conveyor.
- Robots performing tracking cannot use the Continuous Turn function.
- The original path resume function (returning to the original path when resuming motion after a pause) cannot be used.
- It is not possible to execute a TP program whose motion mask has the tracking robot group and servo conveyor group. Motion group settings can be viewed in the TP program details.
- In a tracking program, add the wrist joint motion instruction (Wjnt) to all motion instruction or don’t add Wjnt to motion instruction. If there motion instructions with and without a Wjnt, TCP might be deviated from the destination during a tracking motion.
- In a tracking program of servo conveyor, robots must not change tools by using the tool frame select instruction. If the tracking robot changes tools, it may deviate from the target position during tracking.
- It is possible to specify flags on setup display.

8.8. Multi-view

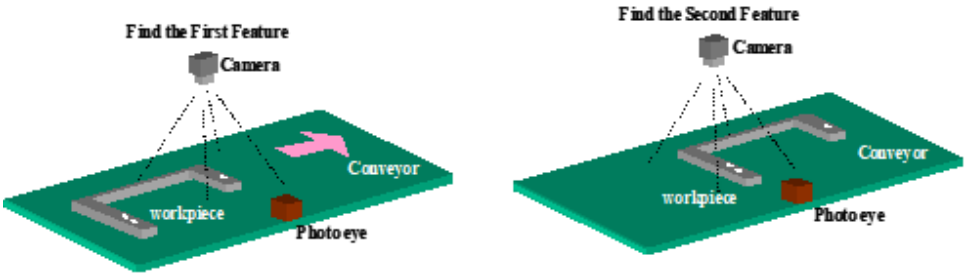
This section explains the setup of multi-view using one or two cameras in iRPickTool.

The multi-view function is useful in a case that a whole part cannot be detected by one measurement for iRPickTool application. A large part such as an automotive floor panel moving on a conveyer belt is given as an example.

8.8.1. Overview

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 a specified distance, the first feature on the part is automatically snapped and found. This detects the first feature on the object. After the part has further moved a predetermined distance, the second feature on the part is snapped and found. This detects the second feature on the object. The sensor task combines the found results and calculates the offset data.



8.8.2. Restrictions

- You can use multi-view function only in line conveyor. You cannot use this function in a circular conveyor or servo conveyor.
- When you use two cameras, you need to connect them to the same robot controller.
- You cannot use the multi-view and the tray function at the same time.
- You cannot use a single-view vision process and a multi-view vision process in a robot controller at the same time.

However you can execute them by switching between them using "I.5.8.4.1. PKSNENBSENS.PC(P.345)". For example, when you want to disable "SENS1" and enable "SENS2" in a TP program, call PKSNENBSENS.PC as shown below.

```
CALL PKSNENBSENS ('SENS1',Disable)
CALL PKSNENBSENS ('SENS2',Enable)
CALL PKSNSTART
```

After calling PKSNENBSENS.PC, you have to call PKWCSTART.PC (or PKSNSTART.PC).

- Multi-view function cannot find two or more parts at the same time.
- You cannot begin detecting another workpiece before finishing both measurements of view 1 and view 2 for a workpiece.
- External Model Train cannot be used on 2-D Multi-View Visual Tracking.
- Offset limit function cannot be used on the vision process of 2-D Multi-View Visual Tracking.

8.8.3. Preparation for camera

Number of cameras

You can use one or two cameras.

When you use one camera, the vision detects two different features on a part with the same camera. Therefore, these features go through a field of view for one camera to detect these features.

When you use two cameras, it detects the first feature on the part by one camera and the second one by the other camera. In this case, you can arrange the cameras layout.

Connection

When you use two cameras, you have to connect them to the same robot controller.

Camera layout

You can arrange the camera layout.

Note that the time until the camera snaps an image after the photo eye detects the target part is about 0.1 seconds, as mentioned in "I.4.3. Camera installation and connection(P.255)". Therefore, it is recommended to install the camera so

that the feature to be detected appears near the center of the camera's field of view approximately 0.1 seconds after the photo eye detects the target part.

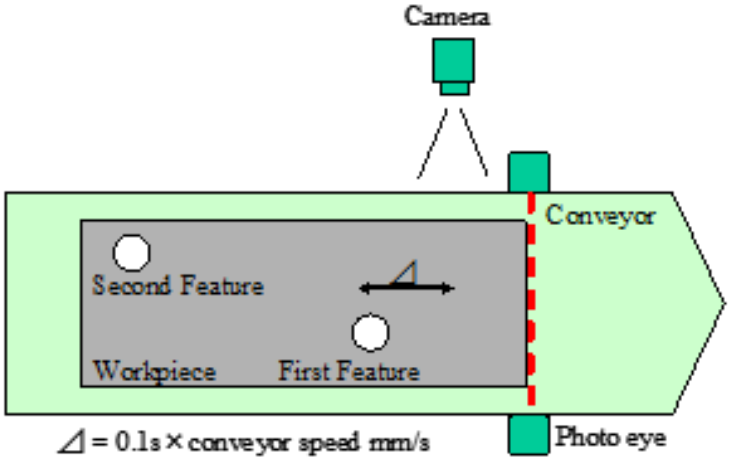


Fig. 8.8.3-1 Example camera layout

8.8.4. Setup

The setup procedures of Conveyor, Camera data, creation of robot program, and execution of robot program are the same as those for a single view configuration. Setup specific to the multi-view is described here.

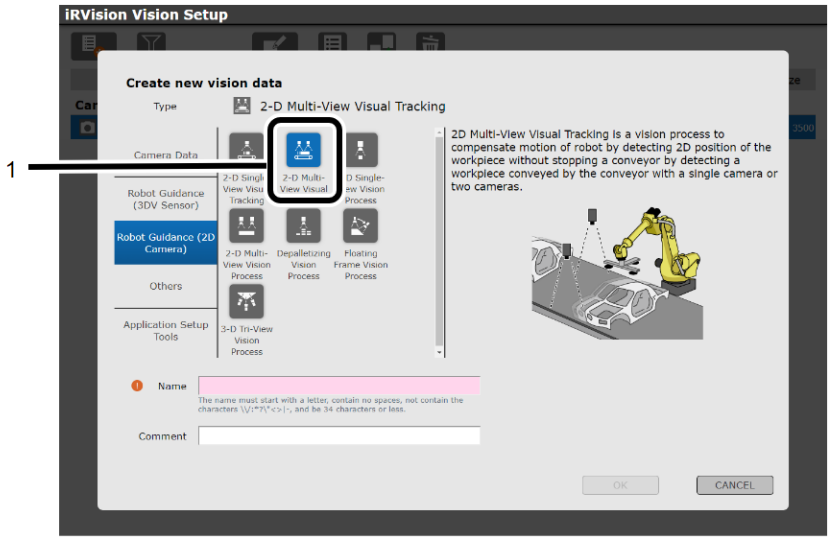
8.8.4.1. Setting up the vision system

In multi-view, you use 2-D Multi-View Visual Tracking vision process. Camera view 1 is for the first feature on a part, and camera view 2 is for the second feature on the same part.

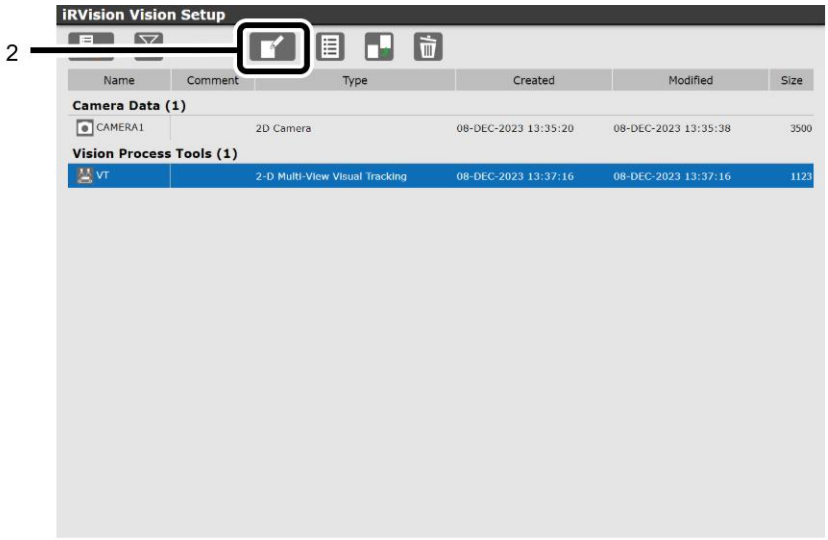
⚠CAUTION
It is necessary to set camera view 1 for the detection first (the first feature) and camera view 2 for the detection later (the second feature).

The setup procedure for setting each camera view is shown below.

1 Create new vision data of [2-D Multi-View Visual Tracking] on the iRVision vison setup.



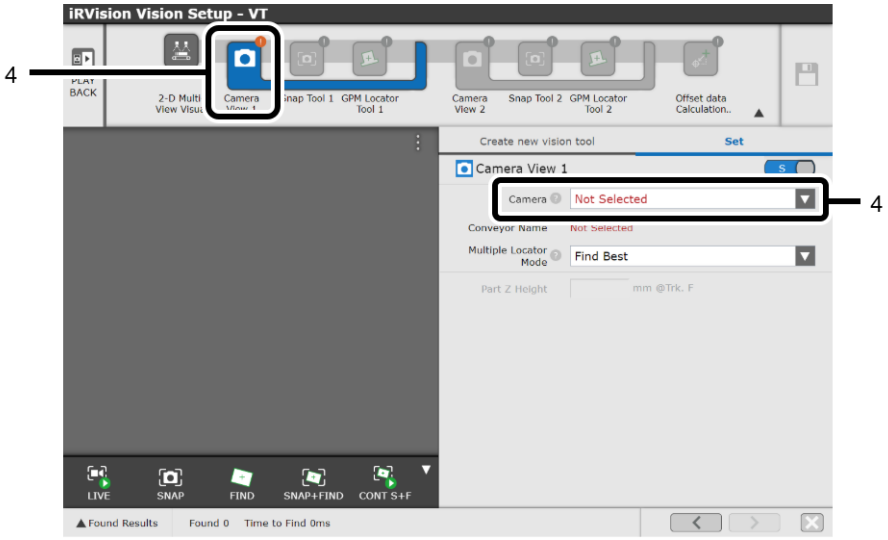
2 On the vision data list screen, select a created vision process, and click [Edit].



3 Set a part so that the first feature is inside camera view 1.

If the supply direction is fixed, set it by aligning it.

4 Click [Camera View 1] in Program Line and select the connected camera in the [Camera] drop-down list.



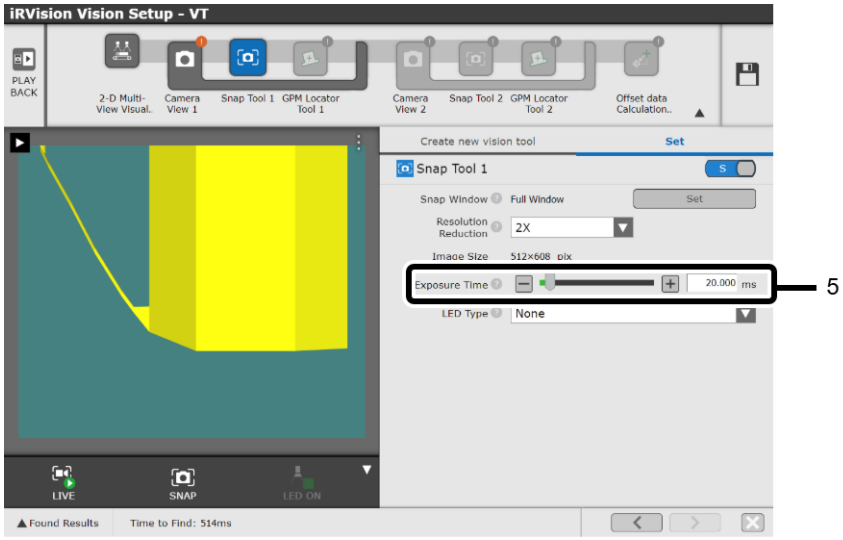
5 Click [Snap Tool 1] under [Camera View 1], and enter an [Exposure Time].

Set the exposure time so that an image of a part that moves during exposure will have as little blurring as possible.

For, example, taking into account image blurring, in order to suppress misalignment during finding by vision to less than 0.1 mm when the conveyor speed is 100 mm/s, set a value that is less than 1 ms as the exposure time.

0.1 (mm) ÷ 100 (mm/s) = 1 (ms)

For information on the calculation of appropriate exposure times to suit the conveyor speed, refer to “Exposure time” in “I.1.5. Study for application(P.19)”.

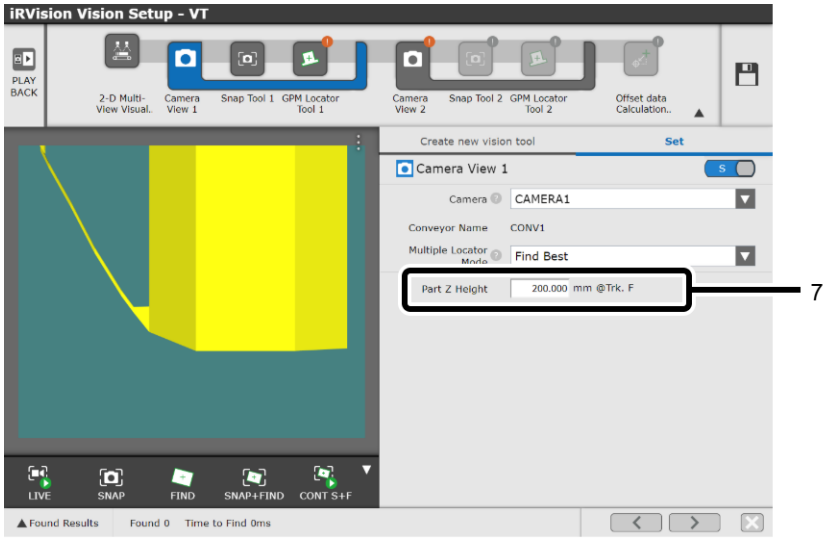


6 Measure the thickness of a part.

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

7 Select [Camera View 1] from the tree, and enter the thickness of a part in [Part Z Height].

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



NOTE

[Part Z Height] is the part height from the tracking frame. Measure it using a ruler, etc.
The part Z height is as shown in the following figure when the tracking frame is taught on the calibration grid surface.

NOTE

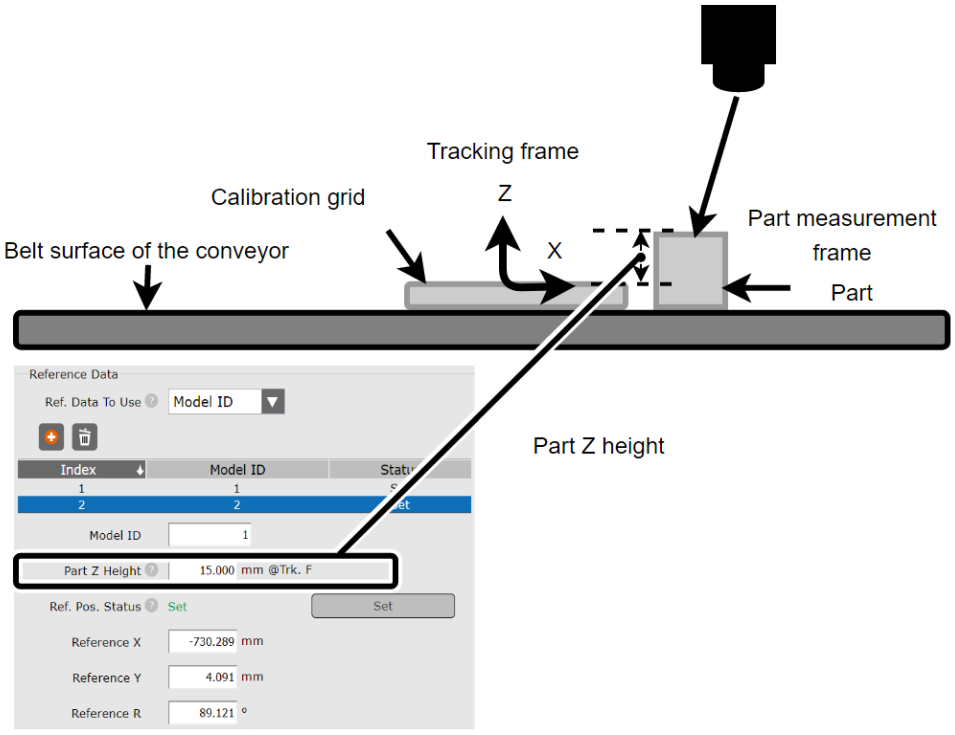


Fig. 8.8.4.1-1 Relationship between the system and 'Part Z Height'

8 **Set up the [GPM Locator Tool 1] under [Camera View 1].**

For information on setup GPM Locator Tool, refer to "I.2.3.7.2. Vision process(P.99)".

9 **Set a part so that the second feature is inside camera view 2.**

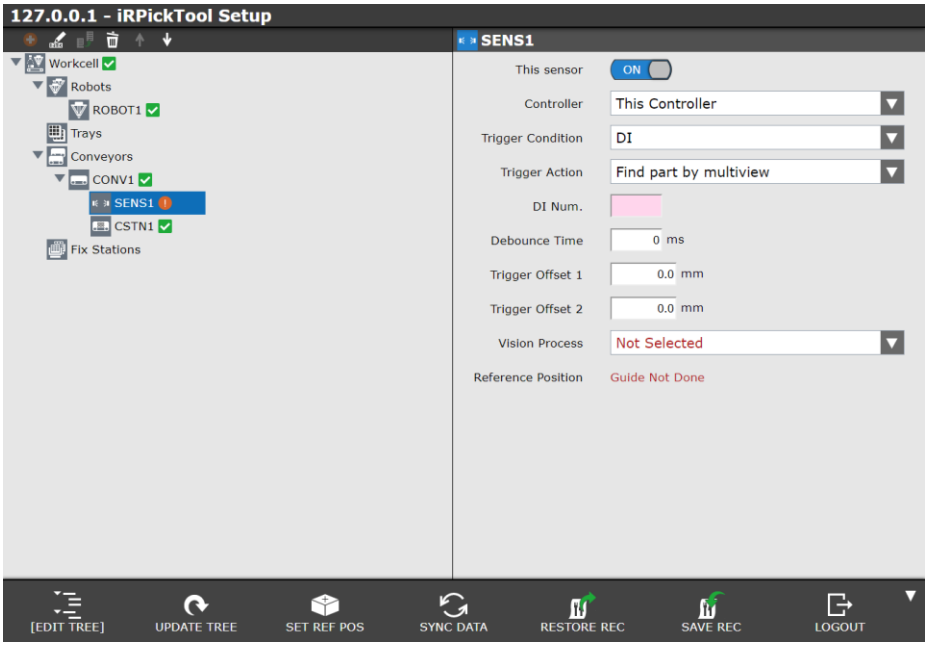
Make the same setting as in steps 4 to 8 for camera view 2.

8.8.4.2. Setting up a sensor

1 **Open the robot homepage.**

2 **Select [iRVision/iRPickTool] in the menu area and click [iRPickTool System Setup] in the content display area.**

3 **On the displayed screen, when you select a sensor object in a tree and set the cursor on [Trigger Condition] and select [DI] and set the cursor on [Trigger Action] and select [Find part by multiview], the following screen is displayed.**



Trigger Offset 1, Trigger Offset 2

[Trigger Offset 1] is a distance in mm by which the conveyer moves after the part passes by the photo eye until the first feature is snapped and found by camera view 1.

[Trigger Offset 2] is a distance in mm by which the conveyer moves after the part passes by the photo eye until the second feature is snapped and found by camera view 2.

Case of one camera

When you use one-camera configuration shown in the figure below, specify [Trigger Offset 1] and [Trigger Offset 2] as follows.

Trigger Offset 1 = D1 (mm) - L1 (mm) - 0.1 (s) × Conveyor speed (mm/s)
Trigger Offset 2 = D2 (mm) - L1 (mm) - 0.1 (s) × Conveyor speed (mm/s)

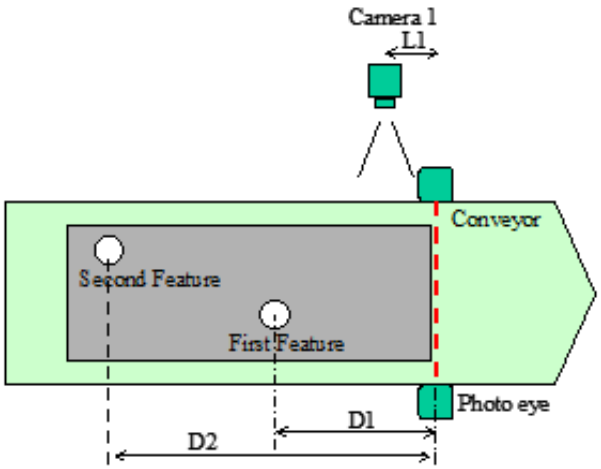


Fig. 8.8.4.2-1 Example of one-camera configuration

Case of two cameras

If you use the two-camera configuration shown in the figure below to have Camera 1 find the first feature and have Camera 2 find the second feature, specify [Trigger Offset 1] and [Trigger Offset 2] as follows.

Trigger Offset 1 = D1 (mm) - L1 (mm) - 0.1 (s) × Conveyor speed (mm/s)
Trigger Offset 2 = D2 (mm) + L2 (mm) - 0.1 (s) × Conveyor speed (mm/s)

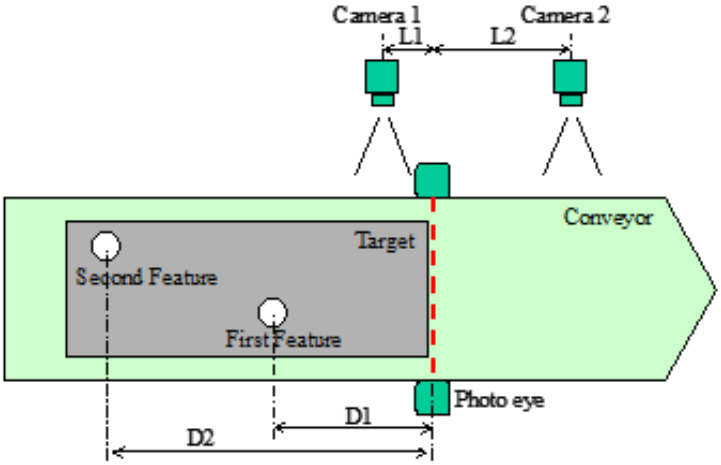


Fig. 8.8.4.2-2 Example of two-camera configuration

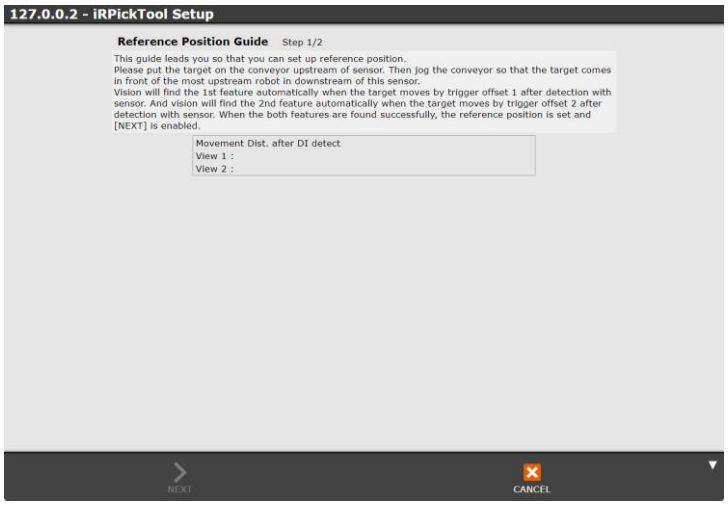
Vision process

Specify the vision process name of 2-D Multi-View Visual Tracking vision process.

8.8.4.3. Reference position setting and robot position teaching

You set reference positions according to the Guide instructions.

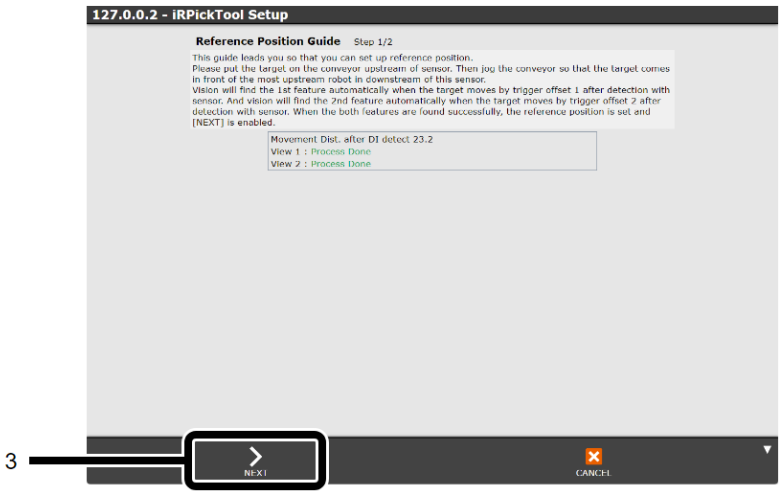
1. **Put a part on the upstream side of the sensor such as a photo eye position on the conveyor.**
2. **Click [SET REF POS] to start the Reference Position Guide.**
   The following screen is displayed.



3. **Move the conveyer to make the photo eye detect the part.**
   Stop the conveyor when the part reaches in front of each robot. Here, when the part moves by [Trigger offset 1] distance after the photo eye detection, the first feature will be snapped automatically by camera view 1. And when the part moves by [Trigger offset 2] distance after the photo eye detection, the second feature will be snapped automatically by camera view 2.
   When the part is detected by the photo eye successfully, the movement distance is displayed and updated. The detection results are also displayed for each camera view. 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. [NEXT] is then enabled. Click [NEXT].

MEMO

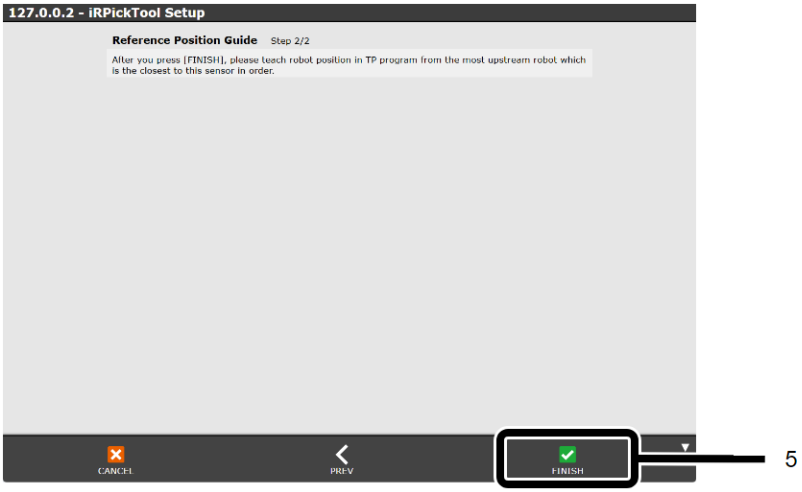
The trigger is set with the encoder value when it is detected by camera view 1



4 Teach the robot positions.

This teaching procedure is the same as that for a single view configuration.

5 Click [Finish] to finish the Guide.

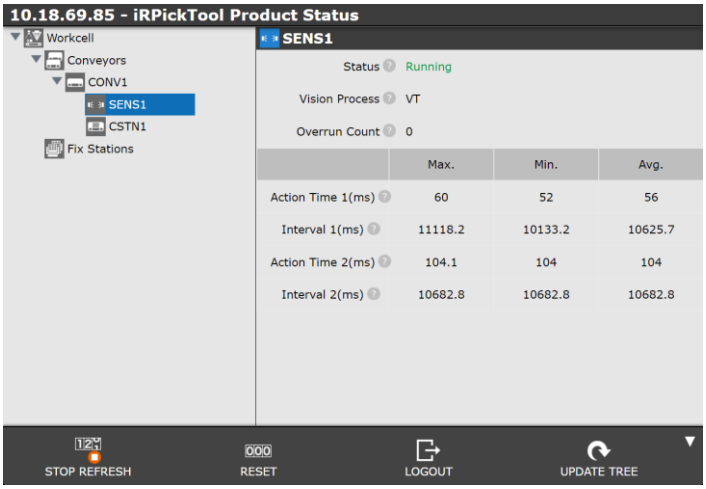


MEMO

The position information of the midpoint between the first feature and the second feature is stored in camera view 1 of vision register (VR[1]) by PKCSGETQUE. However, the robot starts tracking the part only after the teaching position of the robot exceeds upstream boundary.
When the midpoint between the first feature and the second feature exceeds the upstream boundary, PKCSGETQUE succeeds.

8.8.5. Production Status

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



Vision process

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

Action Time 1

Total process time from the time camera view 1 is executed until it snaps an image.

Action Time 2

Total process time from the time camera view 2 is executed until the vision process result is input to the queue.

Interval 1

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

Interval2

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

MEMO

After a DI is input, the next DI input is ignored until a conveyer travels a distance up to (Trigger Offset 2 - Trigger Offset 1). This is to prevents only camera view 1 from being executed continuously. In this case, the Overrun Count is not updated.

8.9. 3D Vision Sensor

This section describes the method for setting up iRPickTool when using the 3D Vision Sensor.

8.9.1. Overview

You can pick the workpiece on the upper side of overlapped workpieces or tilted workpiece using the 3D Vision Sensor. By using depth images obtained by the 3D Vison Sensor, you can also perform 2D visual tracking for workpieces that are difficult to detect using normal 2D images.

8.9.2. Restrictions

- The tray function cannot be used for conveyors using “3DV Single-View Visual Tracking”.
- The clearance check function cannot be used in conveyors using “3DV Single-View Visual Tracking”.
- Conveyors using “3DV Single-View Visual Tracking” cannot be selected as the infeed conveyor of the pre-grouping function.
- [Depth Snap Tool] of 3D Vision Sensor cannot be used on “2-D Multi-View Visual Tracking”.

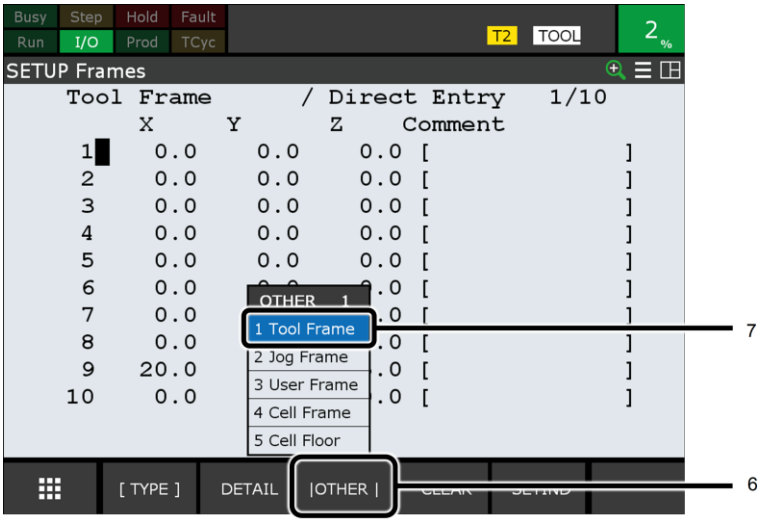
8.9.3. Mounting position setup

The following describes how to set up the mounting position.

1. **Prepare a calibration grid suitable for the 3D Vision Sensor's field of view.**
2. **Set the calibration grid directly below the 3D Vision Sensor.**
   Fix it with tape to prevent it from moving.
3. **The calibration grid can be set in any orientation. For example, for line tracking, match the flow direction of the conveyor and the X direction of the calibration grid.**
4. **Press the [MENU] key on the robot controller’s teach pendant.**
   The menu appears.
5. **From the menu, select [SETUP] → [Frames].**



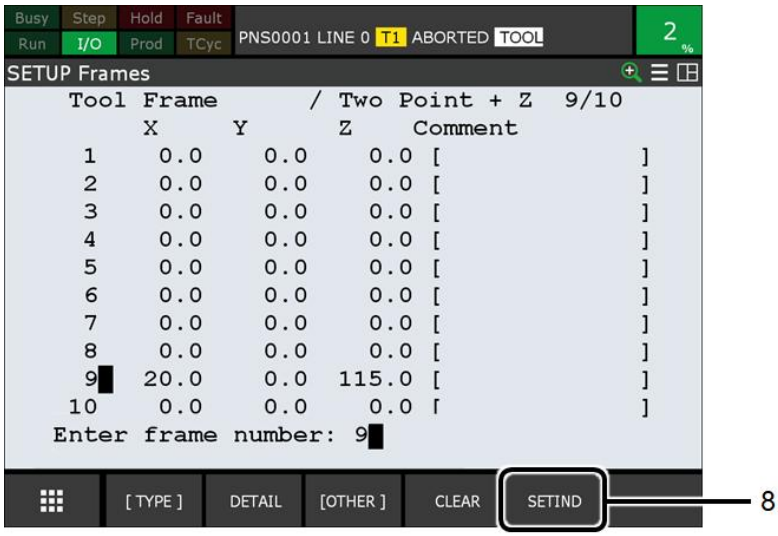
6. **Press F3 [OTHER].**
7. **Select [Tool Frame] from the menu.**
   The Tool Frame list screen appears.



8 Press F5 [SETIND].

9 Enter the frame number that has been used for the pointer tool.

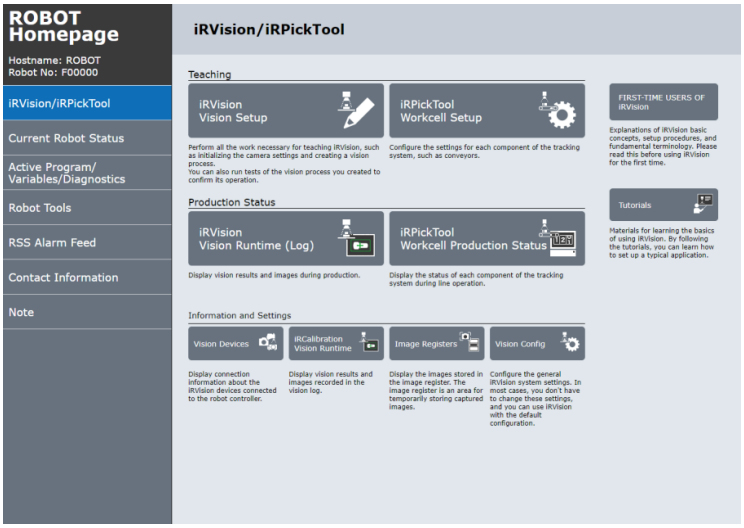
This procedure is explained using Tool 9 as an example.



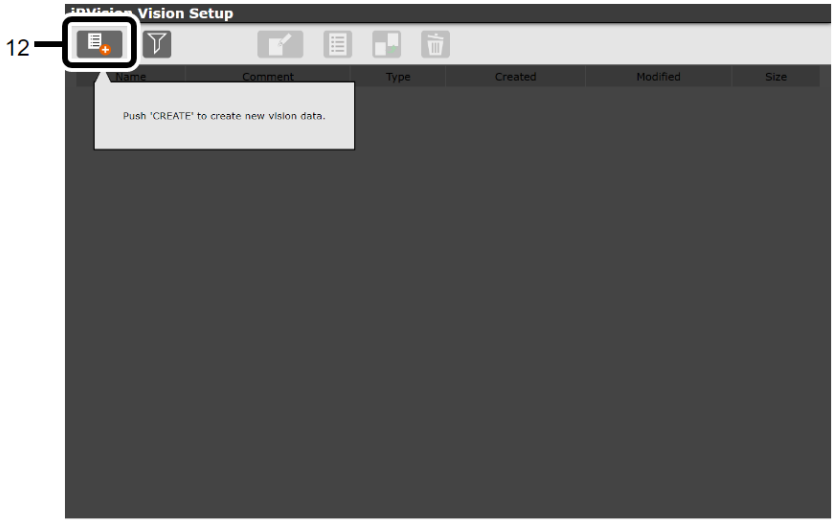
10 Open the robot homepage.

11 Select [iRVision/iRPickTool] in the menu area and click [iRVision Vision Setup] in the content display area.

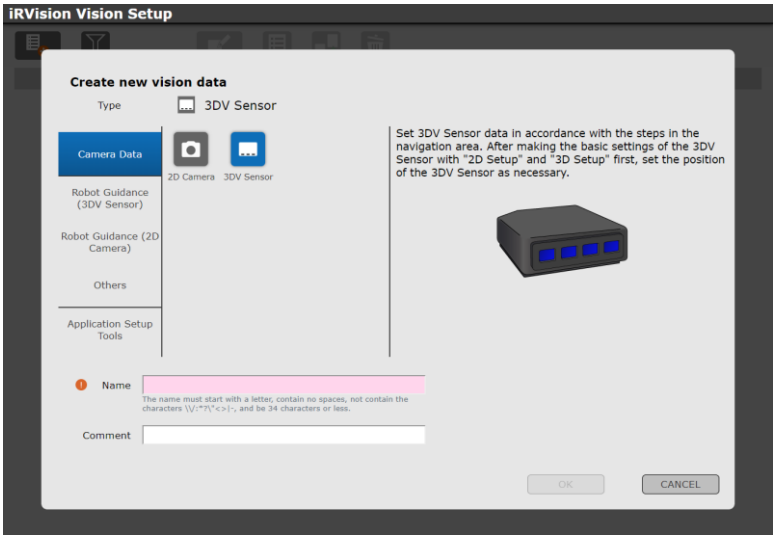
The [iRVision Vision Setup] screen appears.



12 Click [CREATE].



13 Select [3DV Sensor].



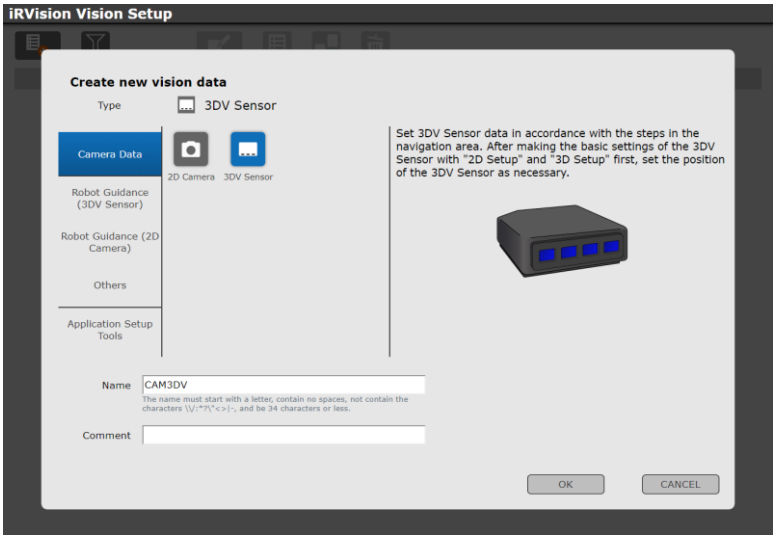
After a while, the following screen appears.

14 Enter the name of the 3D Vision Sensor in [Name].

Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

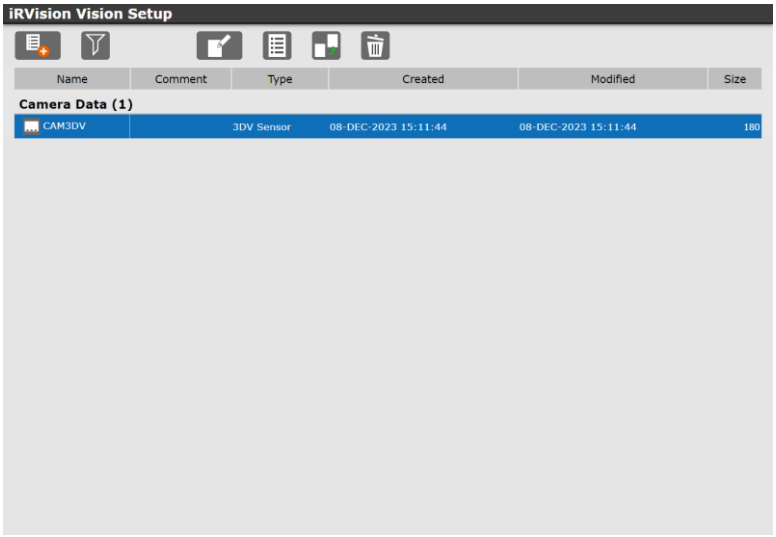
Example: CAM3DV

15 Click [OK].



The created camera data is displayed in the list.

16 Select the created camera data and click [EDIT].

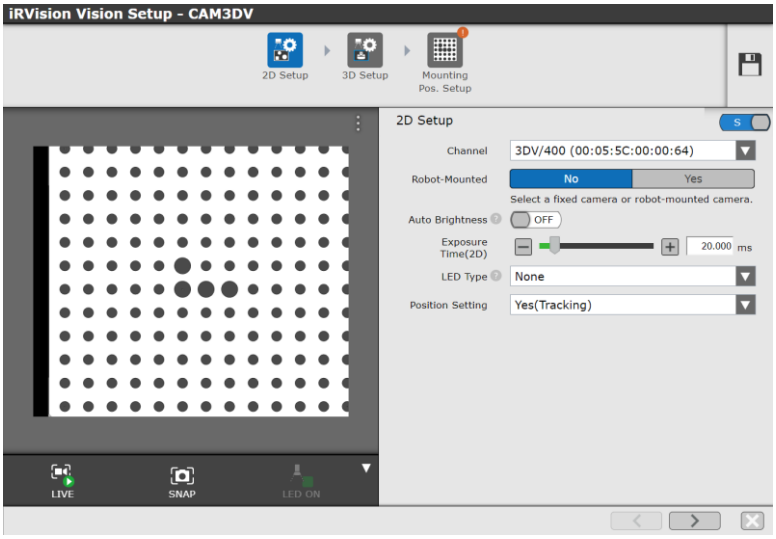


After a while, the following screen appears.

17 With [2D Setup] selected in the navigation area, select [Channel] from the drop-down list.

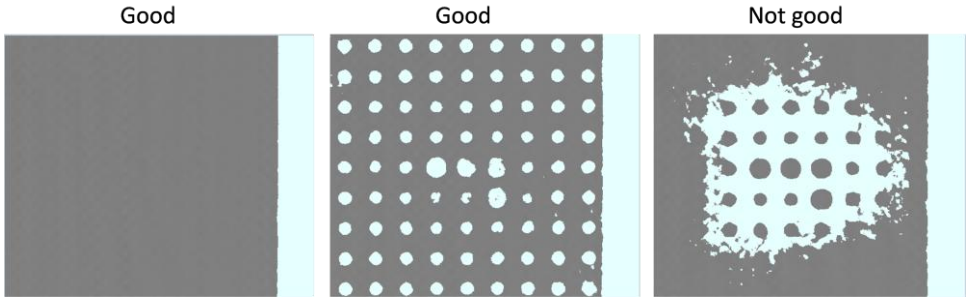
18 Enter [Exposure Time (2D)].

19 In [Position Setting], select [Yes (Tracking)] from the drop-down list.



20 **Select [3D Setup] in the navigation area.**

21 **Adjust exposure time, projector intensity and noise removal level so that depth data on surface of the calibration grid can be measured. The depth data on the grid points are not necessary. Adjust so that depth data around grid points can be measured.**

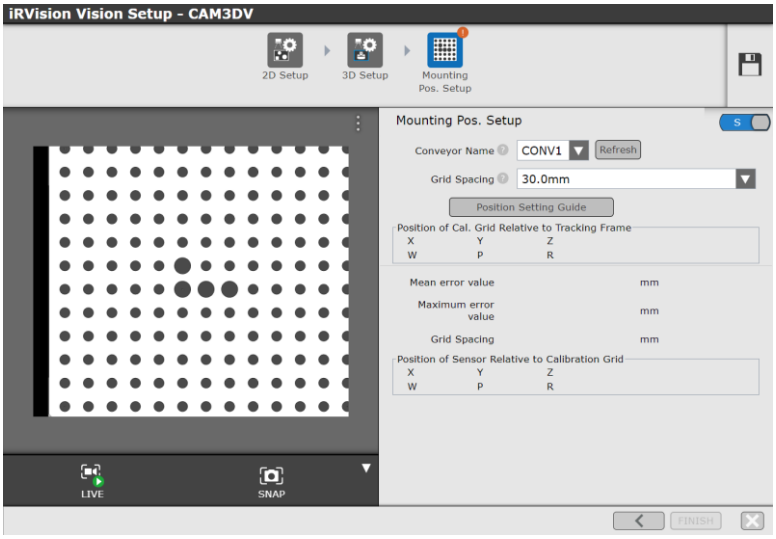


22 **Select [Mounting Pos. Setup] in the navigation area.**

23 **Select [Conveyor Name].**

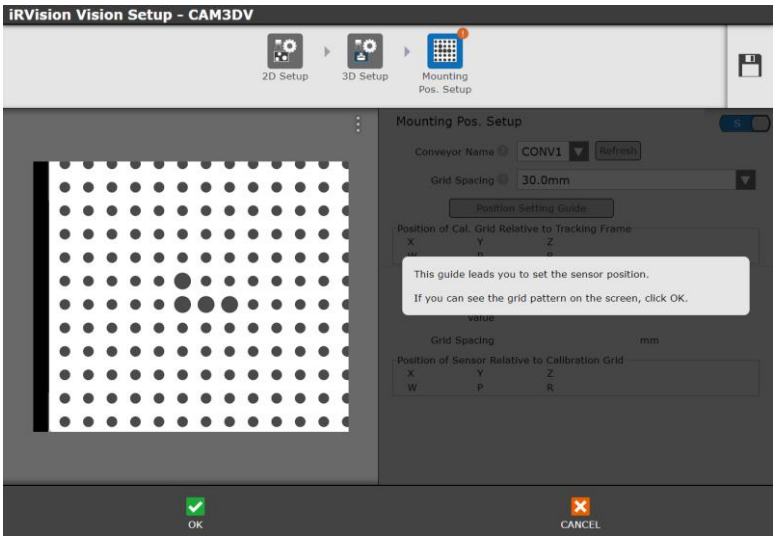
24 **Enter [Grid Spacing] of the grid pattern.**

25 **Click [SAVE].**



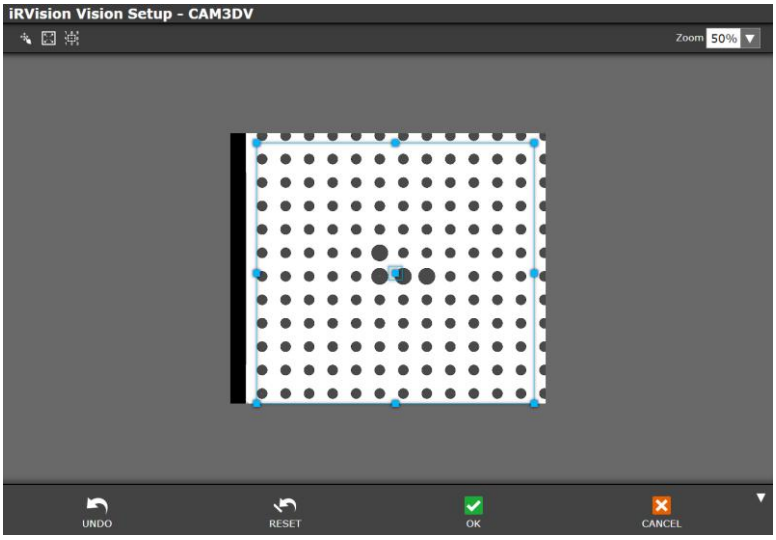
26 **Click [Position Setting Guide].**

27 **Confirm that the calibration grid is displayed on the screen and click [OK].**

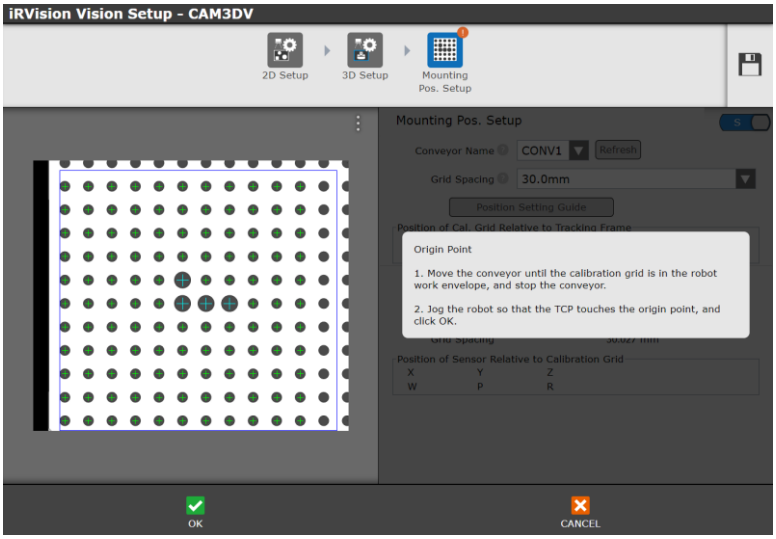


28 **Enclose the dot pattern that appears on the screen with a magenta rectangular window.**

29 **Click [OK].**



30 **Move the conveyor, and stop when the calibration grid gets near the center of the robot's operating range.**



31 **Touch up the origin of the calibration grid.**

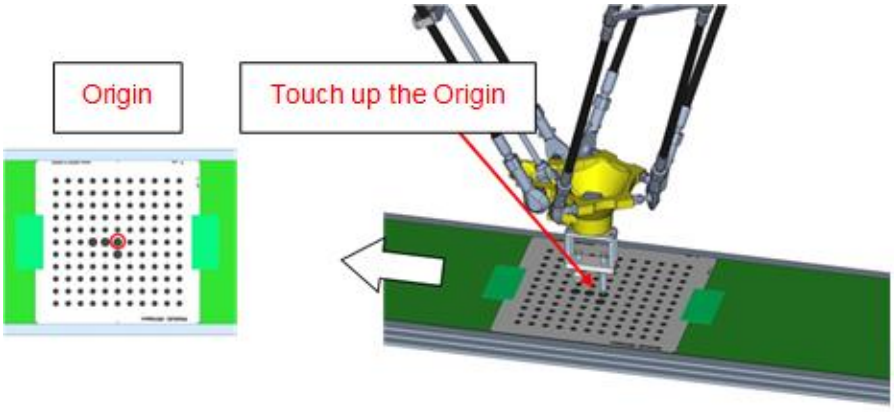


Fig. 8.9.3-1 Touching up the origin of the calibration grid

32 **Click [OK].**

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

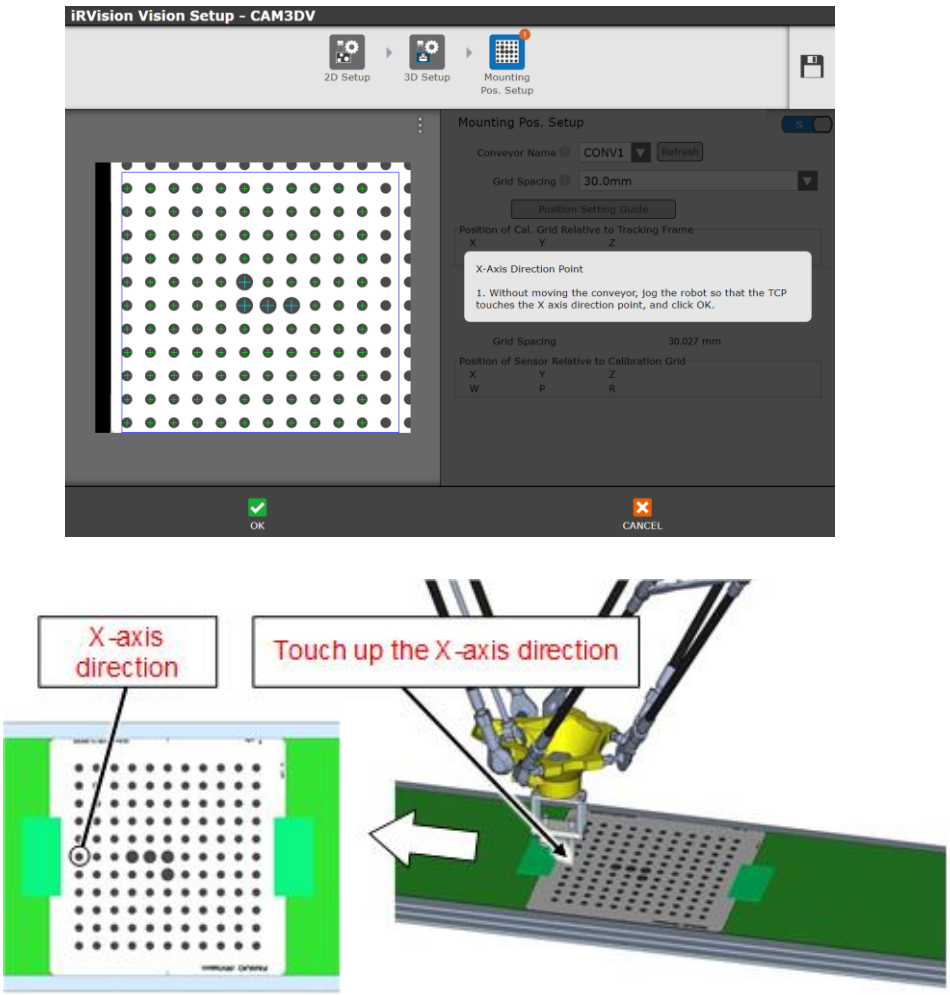
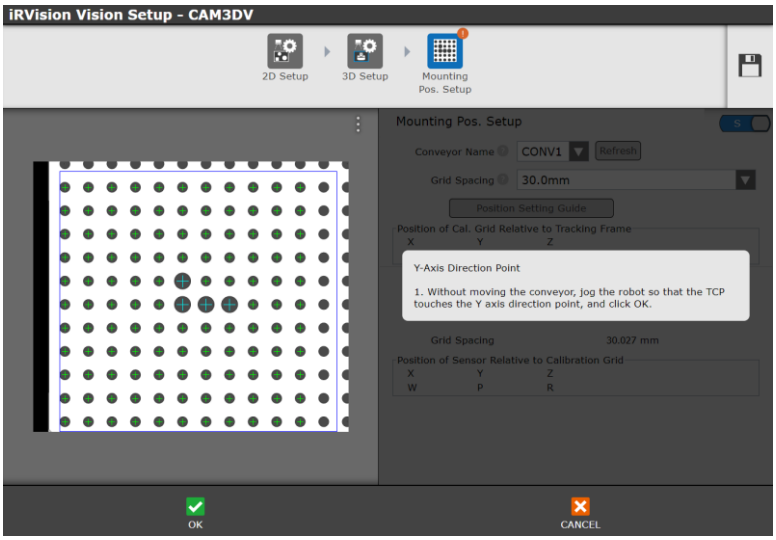


Fig. 8.9.3-2 Touching up the X-axis direction of the calibration grid

34 Click [OK].

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



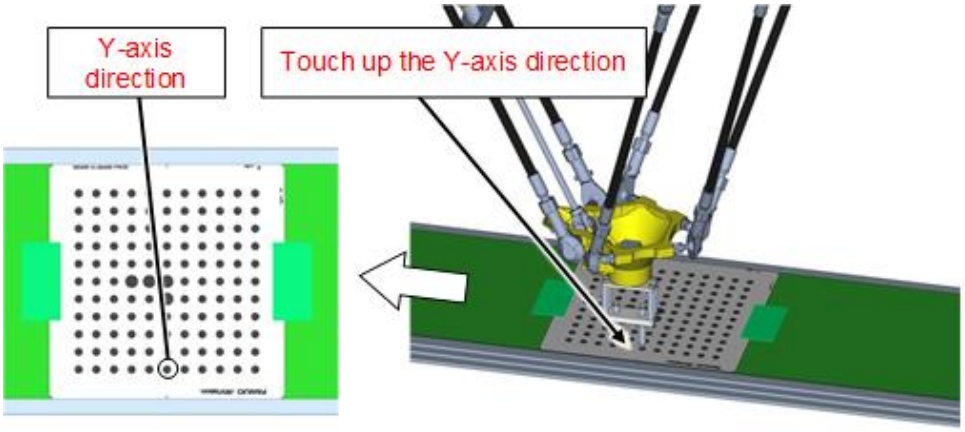
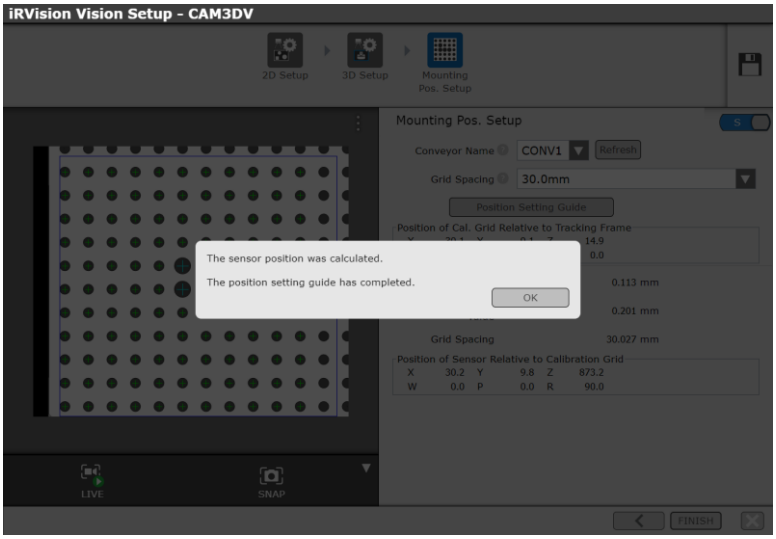


Fig. 8.9.3-3 Touching up the Y-axis direction of the calibration grid

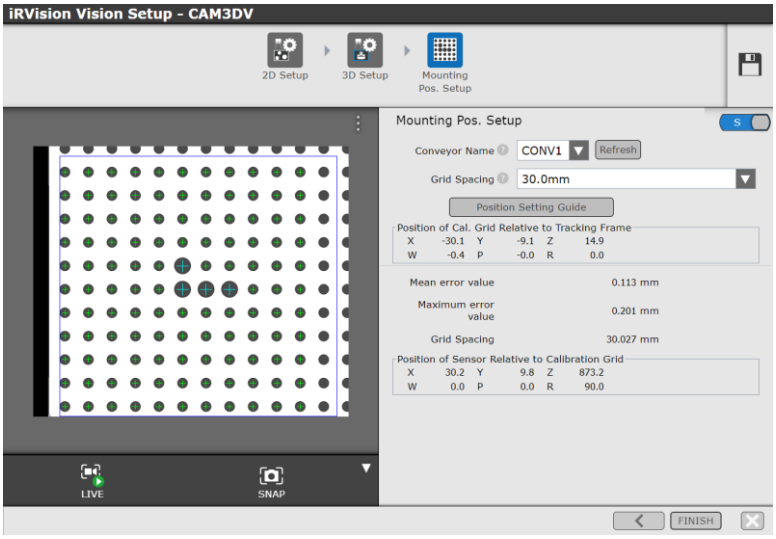
36 Click [OK].

The mounting position is calculated.

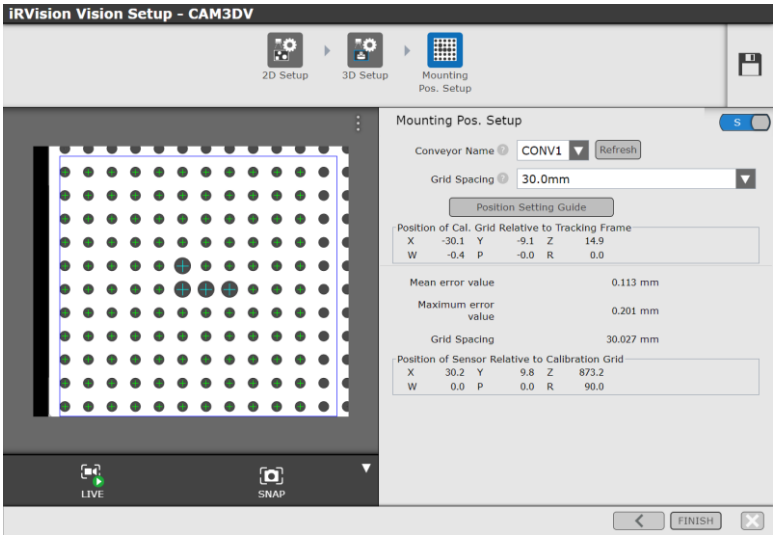


37 Click [OK].

38 Click [SAVE].



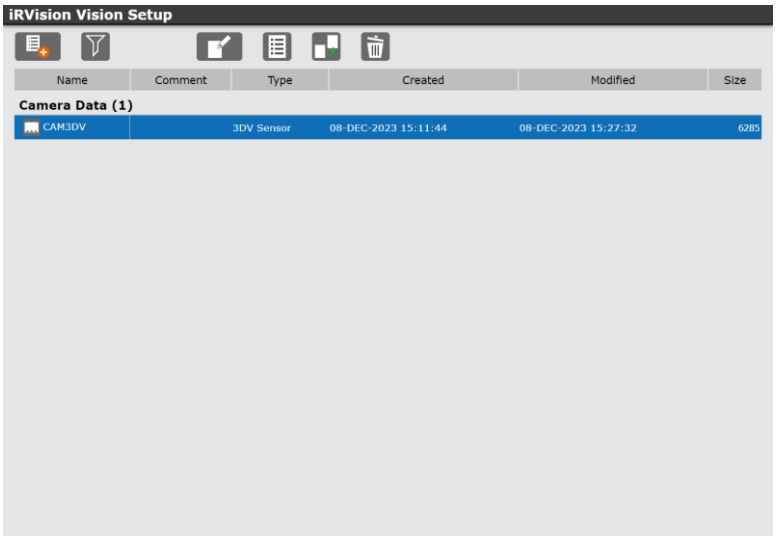
39 Click [FINISH].



8.9.4. Vision process (3D)

The following describes how to set up the vision process when picking up a part three-dimensionally, such as when picking up the part on the top of overlapped parts or a tilted part. For how to perform 2D visual tracking using depth images provided by the 3D Vison Sensor, refer to the following subsection, “Vision process (2D)”.

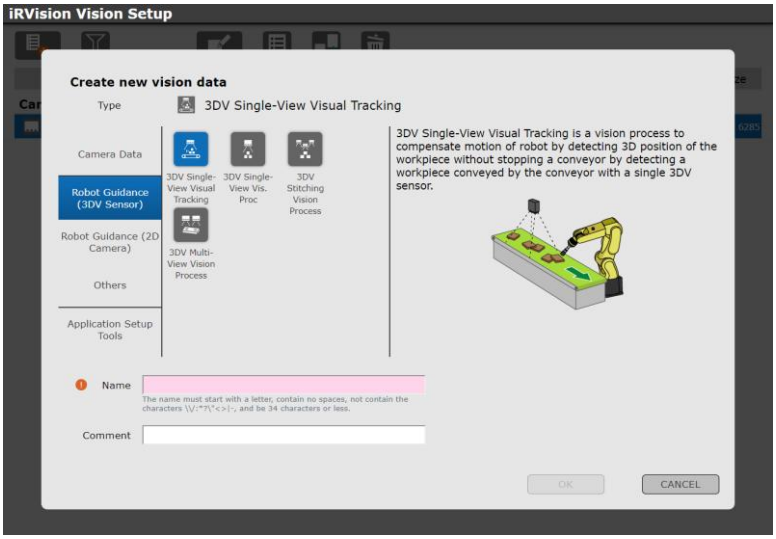
1 Click [CREATE].



The [Create new vision data] screen appears.

2 Select [3DV Single-View Visual Tracking].

After a while, the following screen appears.

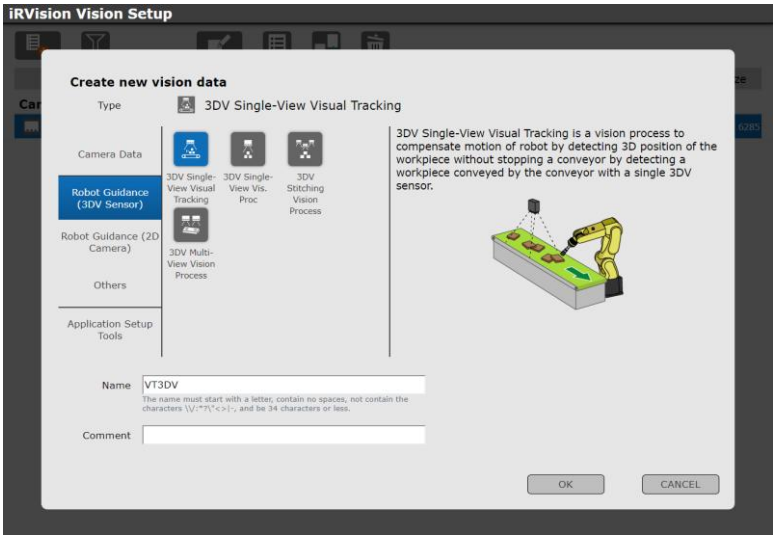


3 Enter the vision process name in [Name].

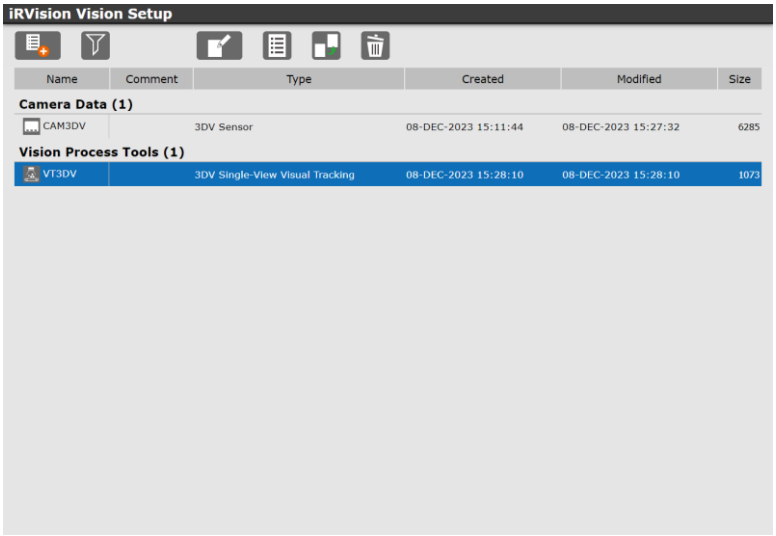
Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

Example: VT3DV

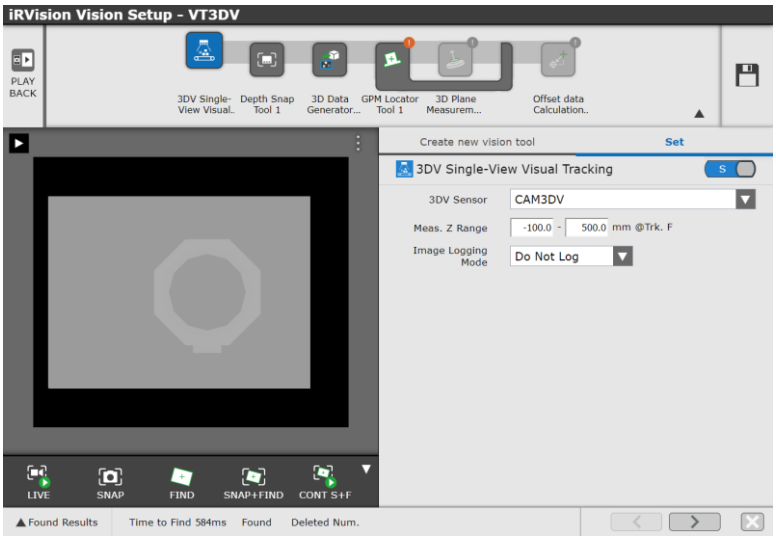
4 Click [OK].



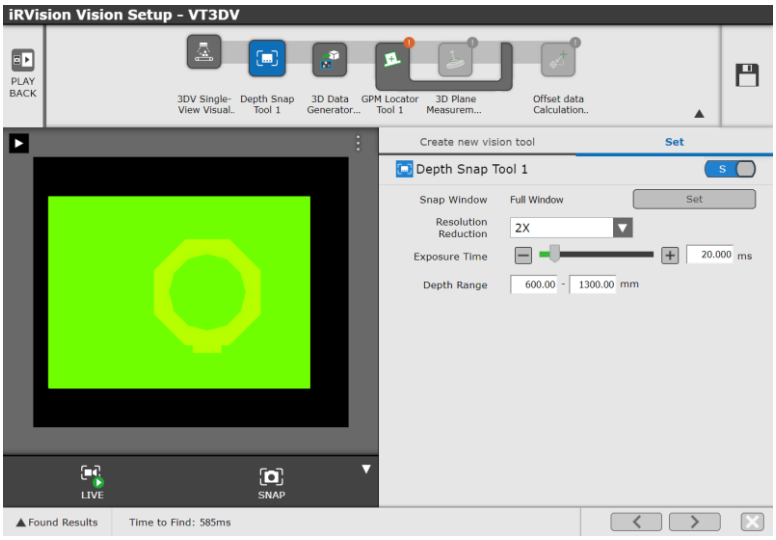
5 On the vision data list screen, select a created vision process, and click [Edit].



6 Select a sensor from the [3DV Sensor] drop-down list.



7 Select [Depth Snap 1] in Program Line.



8 When you wish to capture a partial image, rather than a full depth image, set [Snap Window].

9 You can reduce the image size when snapping the image to reduce the resolution. Select the degree of reduction in the drop-down box for [Resolution Reduction].

10 Set [Depth Range].

This is the depth range to be converted into a grayscale image. The three-dimensional points in the specified depth range will be converted into a grayscale image and output to a 2D command tool.

11 Adjust [Exposure Time], [Projector Intensity], and [Noise Removal Level] by following the procedure below.

12 Set [Color Mode] to Color. The pixels for which depth could not be measured are displayed in black.

13 Set [Projector Intensity] to 16 and [Noise Removal Level] to 10.

These are tentative settings for correctly adjusting [Exposure Time]. Black pixels may temporarily increase.



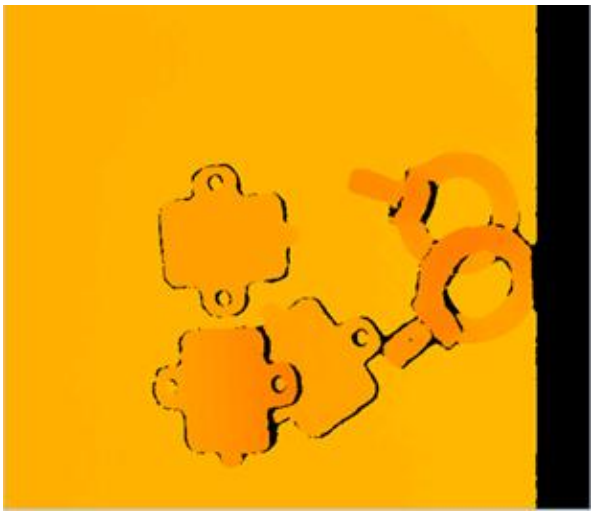
14 **Adjust [Exposure Time] so as to reduce black pixels to the extent possible.**

15 **Reduce [Projector Intensity] so as to reduce black pixels to the extent possible.**

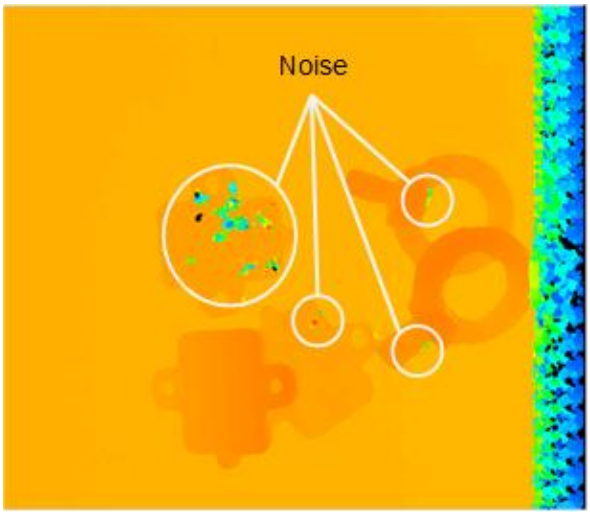
When there is no noticeable change in the image even after changing [Projector Intensity], set a large number for [Projector Intensity]. The larger the value, the more resistant the image is to ambient light.



16 **Reduce [Noise Removal Level] so as to reduce black pixels to the extent possible within a range in which noise does not increase.**



When a small value is set for [Noise Removal Level], snapped depth images will be noisy. The following is an example of an image with noise.



17 Select [3DV Single-View Visual Tracking] in Program Line.

18 Enter [Meas.Z Range]. Enter the range of Z in the tracking frame.

NOTE

When the Z-axis direction of the tracking frame is downward, enter the range of Z in the “detection frame” in [Meas.Z Range]. The “detection frame” refers to the frame in which the tracking frame is rotated 180 degrees around the X-axis, with the Z-axis facing upward.

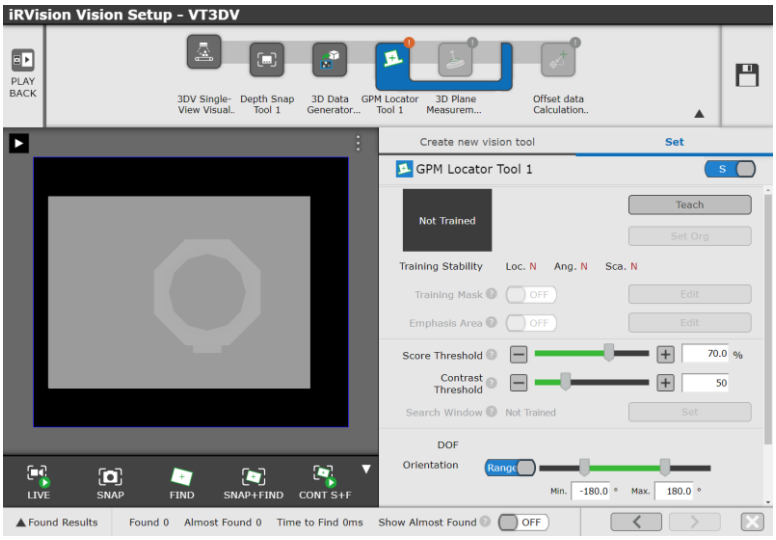
To be specific, when the calibration grid position viewed from the tracking frame at the time of mounting position setup is downward, the direction of the Z-axis of the tracking frame is downward. For example, the direction of the Z-Axis of the tracking frame is downward in the case of a circular conveyor that moves clockwise.

Each color represents the following.

Blue to red:	Three-dimensional points used for detection within the measured Z range.
Black:	Three-dimensional points that could not be measured in the image snapped by the 3D Vision Sensor. If the target part is displayed in black, redo the procedure from setup of the depth snap tool.
White:	Three-dimensional points that are located higher than the upper limit of the measured Z range. These points are not used for detection.
Gray:	Three-dimensional points that are located lower than the lower limit of the measured Z zone. These points are not used for detection.

Perform adjustment so that the conveyor surface become bluish and the top surface of the part becomes red.

19 Select [GPM Locator Tool 1] in Program Line.



20 Place the part so that it is inside the measurement area.

21 Click [SNAP] to capture an image.

22 Click [Teach].

23 Enclose the part with a magenta rectangular window.

24 Click [OK].

25 Click [Edit] in [Training Mask].

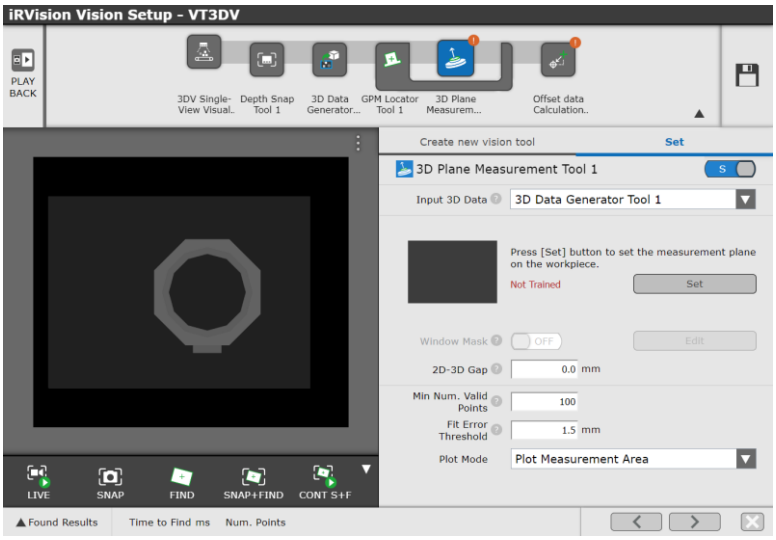
26 Select a tool at the top of the screen, and paint the section in red for which you do not want to make a feature for matching.

27 Click [OK].

28 Set up the DOF as necessary.

For details of the setup, refer to the section “GPM Locator Tool” in the chapter “Command tools” in the "iRVision OPERATOR'S MANUAL (Reference)"(B-84434EN).

29 Select [3D Plane Measurement Tool 1] in Program Line.



30 Place the part so that it is inside the measurement area.

31 Click [Teach].

32 Enclose the area for which the plane is measured with a reddish-purple rectangular window.

33 Click [OK].

34 Enter [2D-3D Gap].

[2D-3D Gap] is the height difference between the measured plane and 2D measurement surface. When the 2D measurement surface is closer to the camera than the measured plane, this is a positive value.

35 Click [Edit] in [Training Mask].

36 When there are sections that you wish to exclude from the measurement area, such as a stepped plane inside the measurement area, click the [Edit] button of [Window Mask] and teach the Window Mask.

Even when the Window Mask is taught, it will be ignored if you change the toggle switch to [Not used].

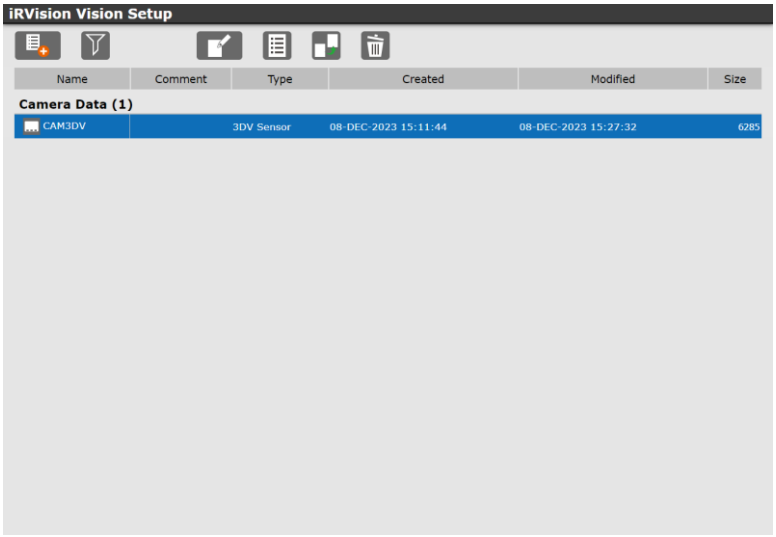
37 Click [SAVE].

38 Click [×].

8.9.5. Vision process (2D)

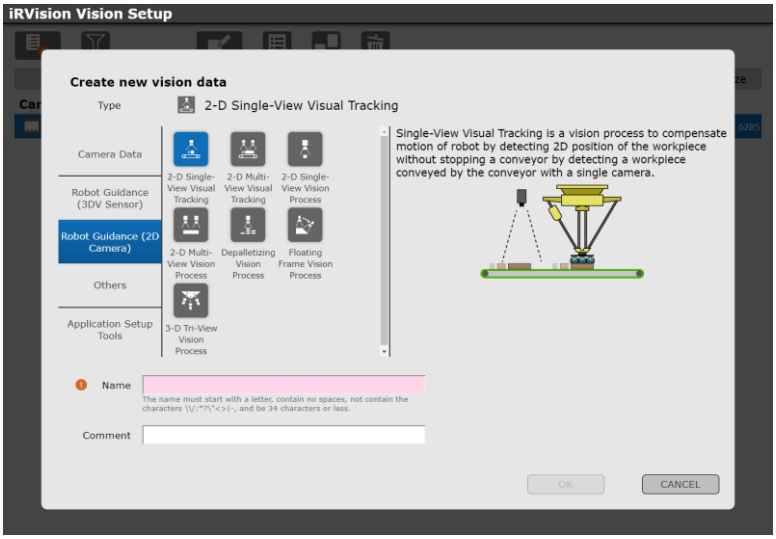
The following describes the method for performing 2D visual tracking using depth images obtained by the 3D Vision Sensor.

1 Click [CREATE].



The [Create new vision data] screen appears.

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



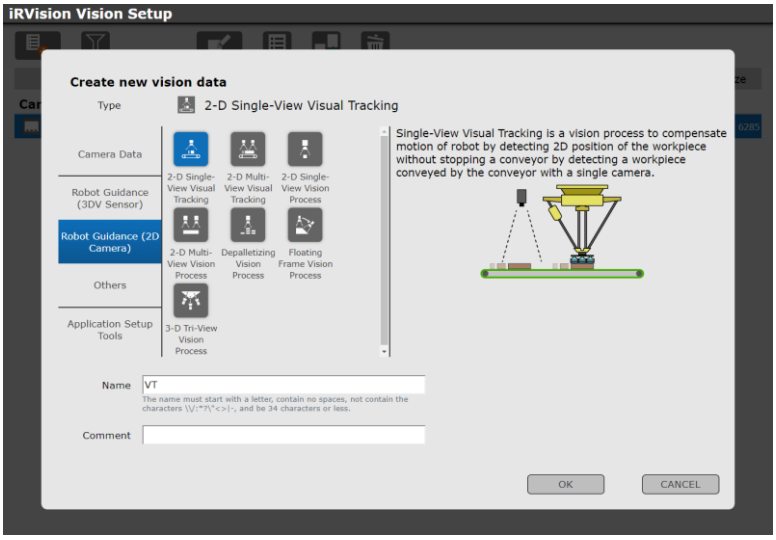
After a while, the following screen appears.

3 Enter the vision process name in [Name].

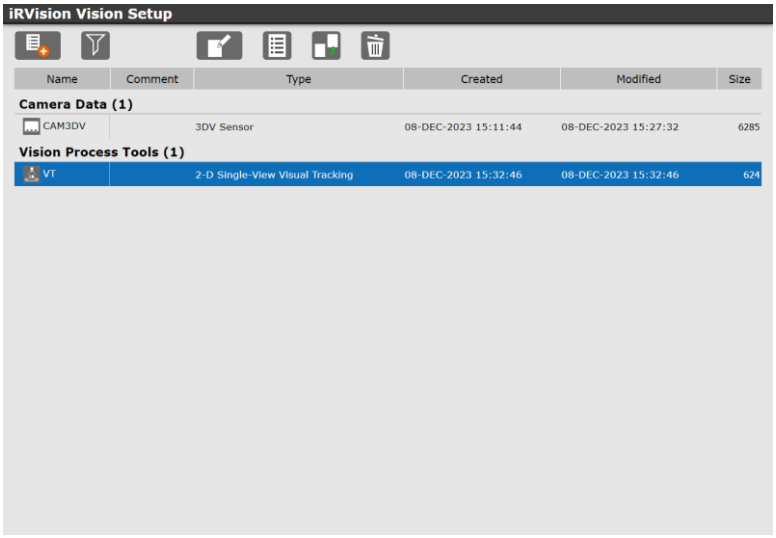
Enter the name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

Example: VT

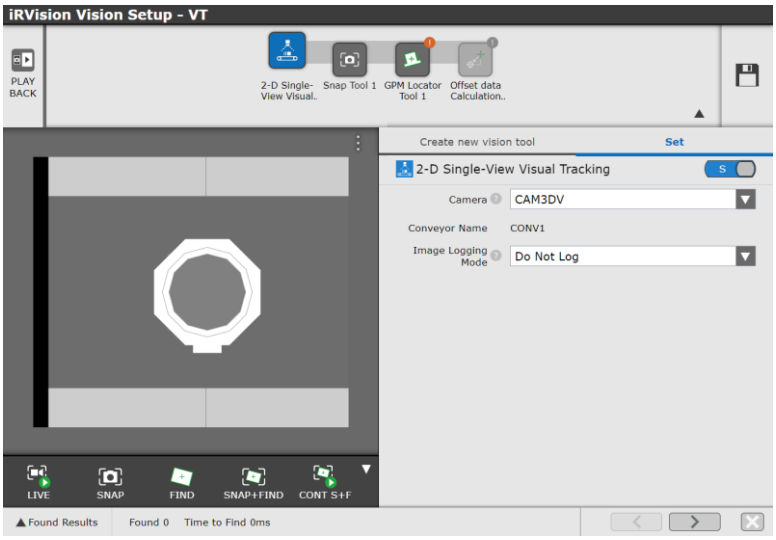
4 Click [OK].



5 **On the vision data list screen, select a created vision process, and click [Edit].**



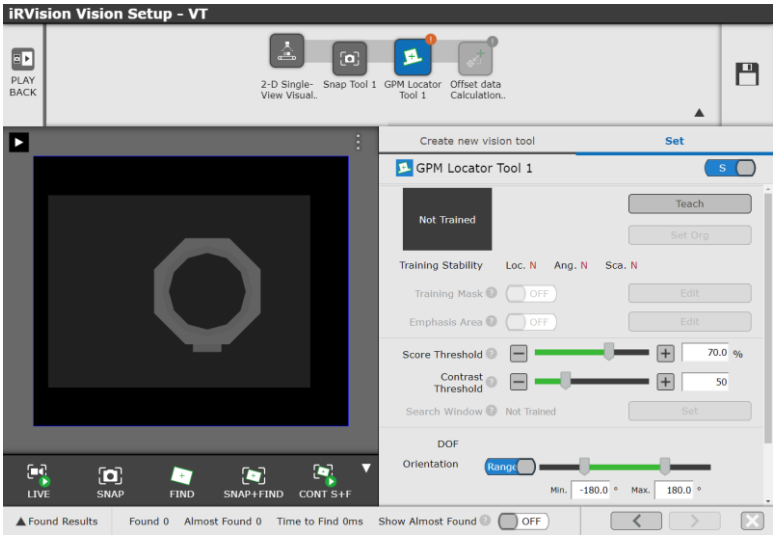
6 **Select the connected camera in the [Camera] drop-down list.**



7 **Drag and drop the [Depth Snap] icon from the [Add new vision tool] tab in the Program Line.**

8 **Select [Snap Tool 1] in Program Line, and drag and drop it to outside the Program Line area to delete it.**

9 **Select [GPM Locator 1] and select [Depth Snap Tool 1] for [Input Image].**



10 Set [Depth Snap Tool 1] and [GPM Locator Tool 1] by following the same procedures as those described in the previous subsection, "Vision Process (3D)".

11 Click [SAVE].

12 Click [×].

MEMO

[Depth Snap Tool] of 3D Vision Sensor cannot be used on 2-D Multi-View Visual Tracking.

8.10. Clearance check function

8.10.1. Overview

During visual tracking, you may accidentally move the parts around the target part when you are trying to pick a part from densely packed parts. In such a case, this function enables you to set an arbitrary range (clearance) around each part, check if there are other parts in the clearance of the part to be picked, and pick the part only when there are no other parts in the clearance.

For example, in a case such as described in the figure below, parts A and E can be picked because there is no other part in the clearance. However, parts B, C, and D cannot be picked because there is another part in the clearance. When part E is picked, however, there is no longer another part in the clearance of part D, and therefore part D can be picked.

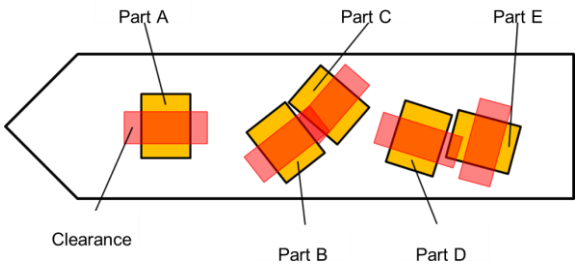


Fig. 8.10.1-1 Example of clearance check

8.10.2. Restrictions

Clearance check cannot be used if any of the following conditions apply.

- Two or more sensors are used for one conveyor station.

- Any of the following settings are enabled on the conveyor setup screen:
  - [Tray] is selected.
  - [Pre-grouping] is used.
  - [Servo Conveyor] is used.
  - [Sort] is used.
- Any of the following are enabled on the sensor setup screen:

When ROBOGUIDE is not used.
[Trigger Condition] and [Action] are not [Distance] and [Find part by vison].
When ROBOGUIDE is used.
[Trigger Condition] and [Action] are not [Distance] and [Find part by vision] or [DI] and [RG put part in queue].
A vision process of [3DV Single-View Visual Tracking] is selected.

8.10.3. Restrictions

A clearance check may not be performed correctly in the following cases:

- Two or more models of parts are simultaneously supplied to the conveyor.

  A clearance check can be set only for a single model of part. For this reason, when two types of parts are simultaneously supplied to the conveyor, as shown in the figure below, part B does not become the target of the clearance check, even though it is in the clearance of part A for which the clearance check is set, because the clearance check is not set for part B. Therefore, it is judged that part A can be picked.

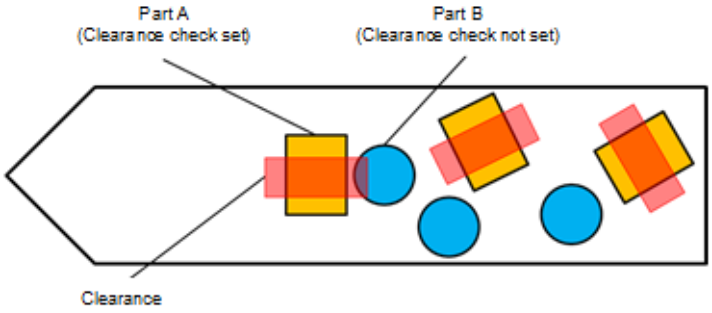


Fig. 8.10.3-1 Example in which two or more models of parts are supplied to a conveyor

NOTE

If the model and model origin are in the same position, a clearance check can be used even when multiple model IDs are used.

- When the field of view of the sensor is close to the upstream boundary of the tracking area of the most upstream robot.

  In the case of the figure below, part B is in the clearance of part A, but is not included in the target of the clearance check because it not detected. As a result, it is judged that part A can be picked.

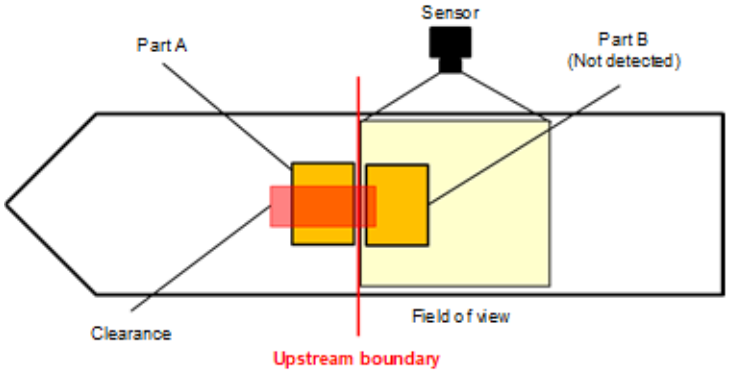


Fig. 8.10.3-2 Example in which the field of view of the sensor is close to the upstream boundary of the tracking area of the most upstream robot

- The send position (send line) of the upstream robot is close to the upstream boundary of the tracking area of the downstream robot.

  In the case of the figure below, part B is in the clearance of part A. However, part B has not crossed over the send position (send line) of robot 1 and data of part B has not been sent to robot 2. Therefore, part B is excluded from the clearance check target. As a result, it is judged that part A can be picked.

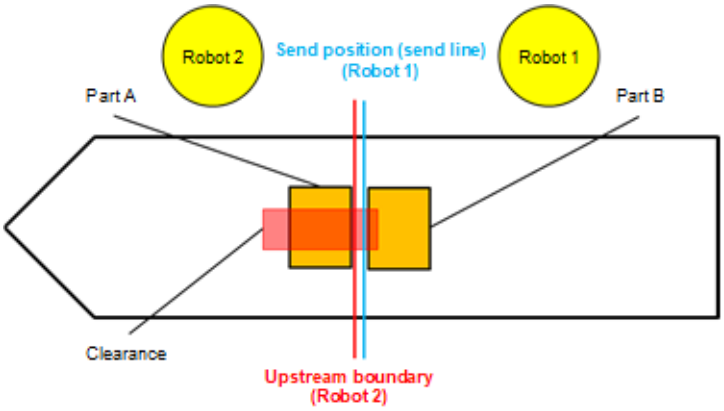


Fig. 8.10.3-3 Example in which the send position of the upstream robot is close to the upstream boundary of the tracking area of the downstream robot

8.10.4. Setup

Before using the clearance check function, first complete the procedures described in "I.4. Other preparations before setup(P.241)".

8.10.4.1. Setup of a workcell and robot

There are no special notes concerning the setup of a fixed station for a clearance check. See "I.5.1. Setting up a workcell(P.266)" and "I.5.2. Setting up a robot(P.268)".

8.10.4.2. Setting up a tray

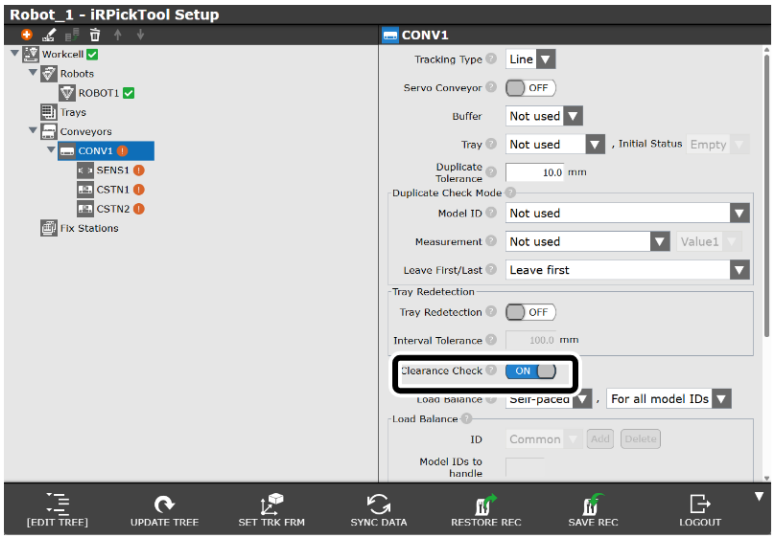
There are no special notes concerning the setup of a fixed station for a clearance check.

Trays cannot be used in a conveyor in which a clearance check is used. When using trays in a conveyor not using a clearance check, or when using a fixed station, refer to "I.5.3. Setting up a tray_2(P.269)".

8.10.4.3. Setting up a conveyor

On the setup screen of the conveyor in which a clearance check is to be used, set the [Clearance Check] toggle switch to [ON].

There are no special notes concerning the setup of a conveyor for a clearance check. See "I.5.4. Setting up a conveyor(P.276)".



8.10.4.4. Setting up a sensor

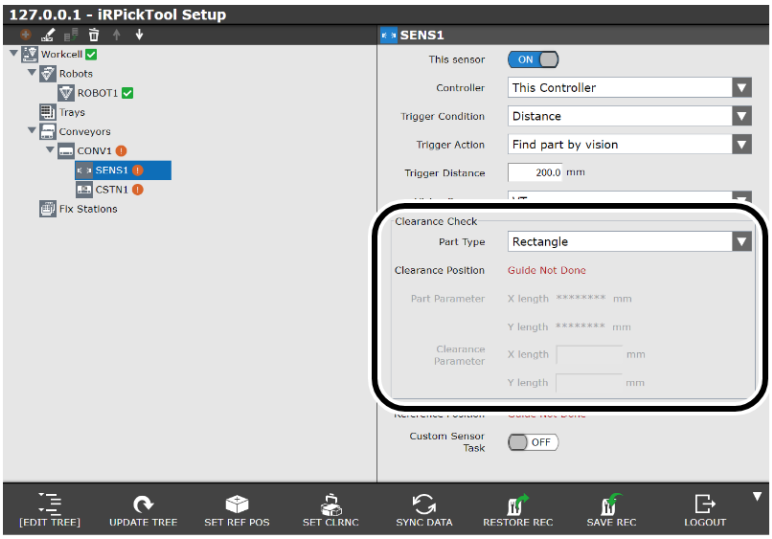
When clearance check is used in the parent conveyor, the clearance check setting item appears on the sensor setup screen.

[Part Type]

Select the shape of the part for which a clearance check is set. [Circle (without Phase)] or [Rectangle] can be selected. The setup screen switches as follows according to Part Type.

When [Part Type] is [Rectangle]

After a while, the following screen appears.



[Clearance Position]

When the procedure provided in "I.8.10.4.5. Clearance guide(P.642)" is complete, [Guide Complete] appears in green.

[Part Parameter]

Set the part length in the X direction in [X length]. Set the part length in the Y direction in [Y length]. This is set in "I.8.10.4.5. Clearance guide(P.642)".

[Clearance Parameter]

Set the clearance length of the part in the X direction in [X length]. Set the clearance length of the part in the Y direction in [Y length]. This is set in "I.8.10.4.5. Clearance guide(P.642)".

For example, the figure below illustrates a case in which [X length] and [Y length] of [Part parameter] are 30 mm and 50 mm, respectively, and [X length] and [Y length] of [Clearance Parameter] are 15 mm and −10 mm, respectively.

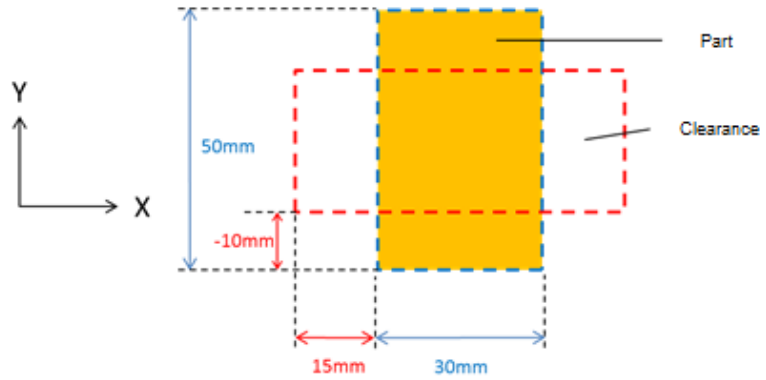


Fig. 8.10.4.4-1 Parameter setting example(when [Part Type] is [Rectangle])

[Set Clearance]

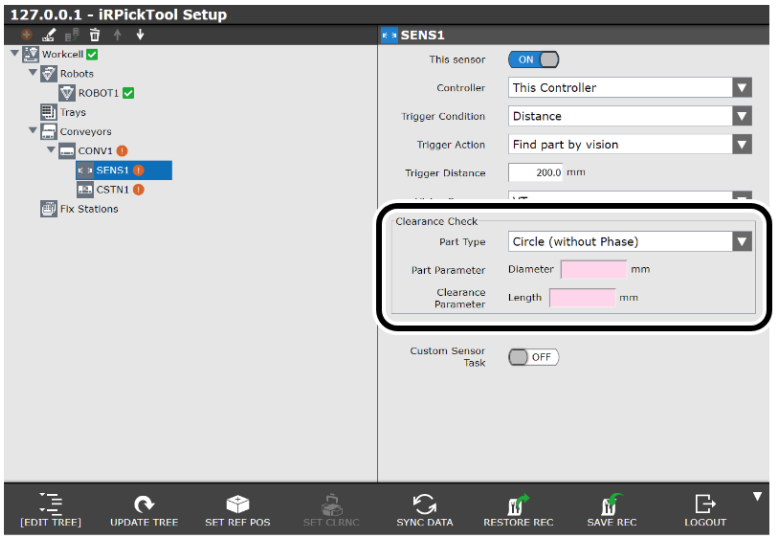
Pressing this key starts the procedure in "I.8.10.4.5. Clearance guide(P.642)".

⚠CAUTION

The procedure in "I.8.10.4.5. Clearance guide(P.642)" is performed with the pointer attached. Perform this procedure before reference position setup, which is performed with the hand attached.

When [Part Type] is [Circle (without Phase)]

After a while, the following screen appears.



[Part Parameter]

Set the diameter of the circle in [Diameter].

[Clearance Parameter]

Set the clearance length in [Length].

The figure below illustrates a case in which [Part Parameter] is 30 mm and [Clearance Parameter] is 10 mm, for example.

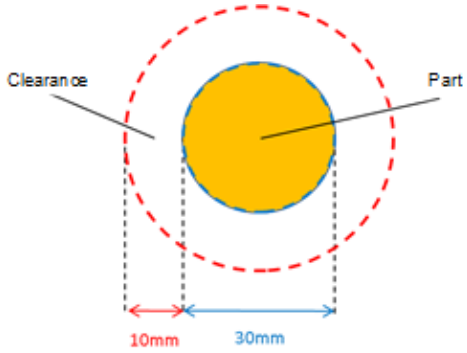


Fig. 8.10.4.4-2 Parameter setting example (When [Part Parameter] is [Circle (without Phase)])

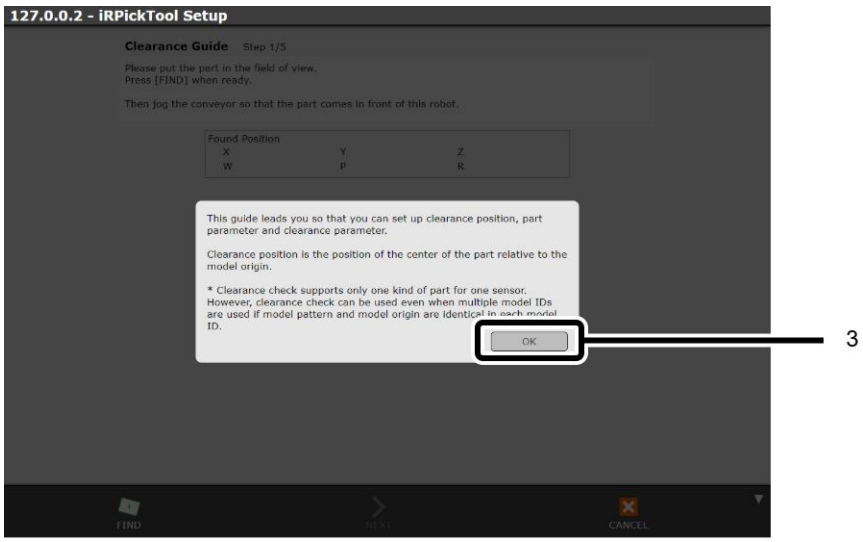
8.10.4.5. Clearance guide

Configure this setting only when [Rectangle] is selected for [Part Type] in "I.8.10.4.4. Setting up a sensor(P.640)".

1 **Stop the conveyor.**

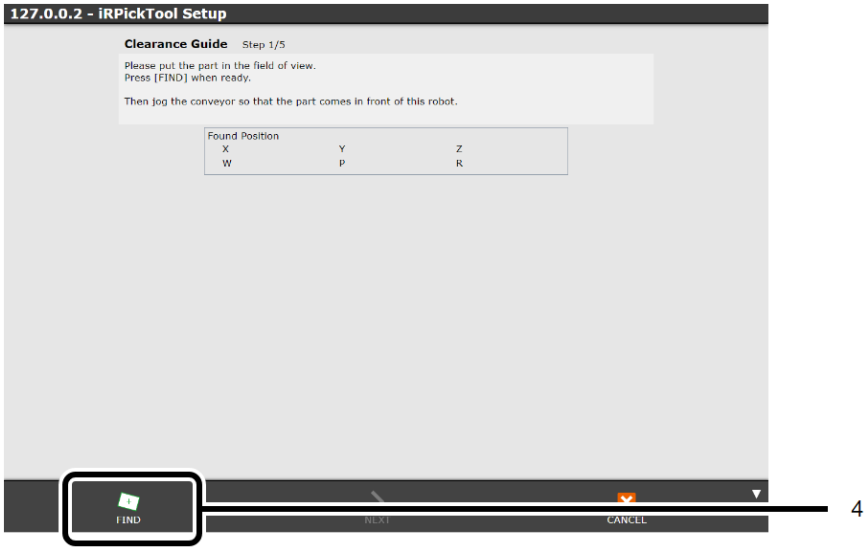
2 **Click [Set CLRNC].**

The following screen appears.



3 **Confirm the content of the displayed message, and click [OK].**

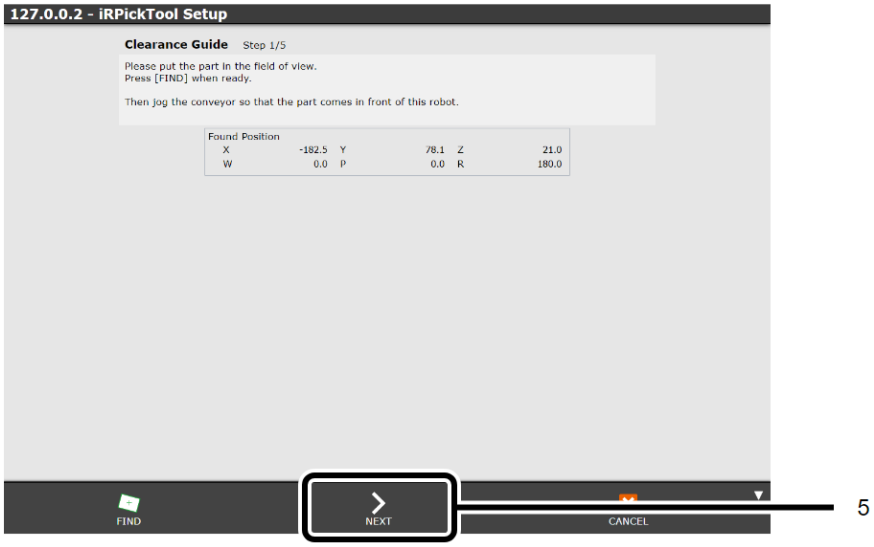
The following screen appears.



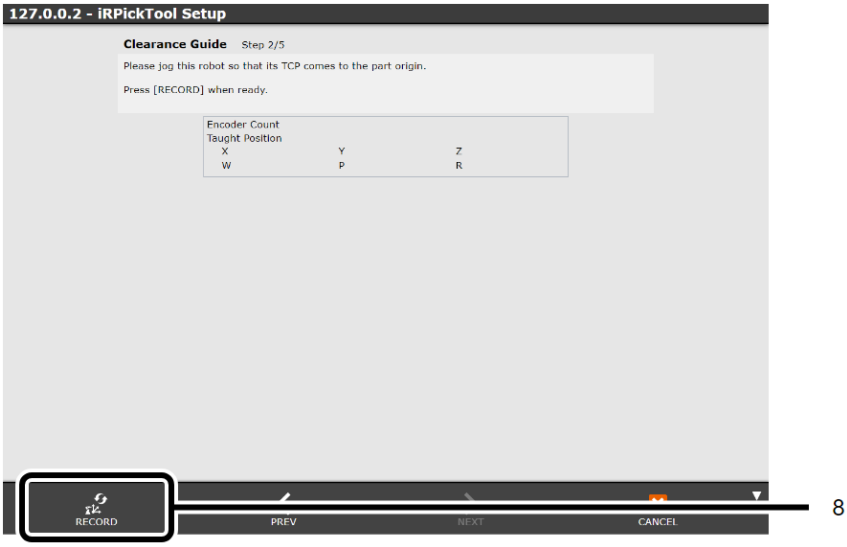
4 Place the part inside the camera's field of view and click [FIND].

When the part is detected successfully, the detection position is displayed and [NEXT] is enabled.

5 Click [NEXT].



The following screen appears.



6 Move the conveyor until the part is located in front of the robot.

7 Move the robot by jog operation, and touch up the origin of the part.

NOTE

Touch up the part at three points: the origin point, X direction point, and Y direction point.
Be careful about the following points.

- The origin of the part must be the vertex of the rectangle, not the model origin of the vision detection.
- The vertices in the x-direction/y-direction of the rectangle, starting from the origin of the part, must be the x-direction point and y-direction point, respectively.

The points to be touched up differ depending on the Z-axis direction of the tracking frame, as described in the following.

- When the +Z axis of the tracking frame is in the upward direction to the conveyor surface
  The Y direction point should be in the direction in which the X direction point is rotated counter-clockwise by 90°.

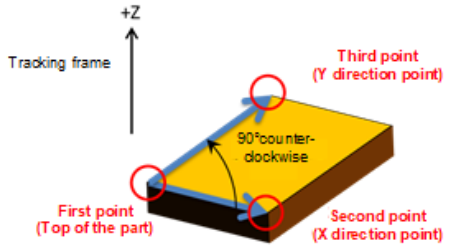


Fig. 8.10.4.5-1 When the +Z axis of the tracking frame is in the upward direction

- When the +Z axis of the tracking frame is in the downward direction the conveyor surface
  The Y direction point should be in the direction in which the X direction point is rotated clockwise by 90°.

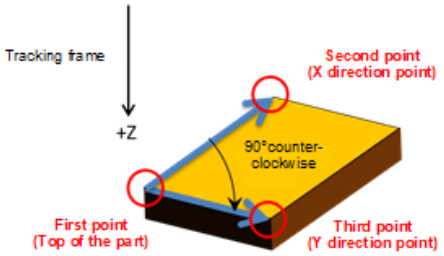


Fig. 8.10.4.5-2 When the +Z axis of the tracking frame is in the downward direction

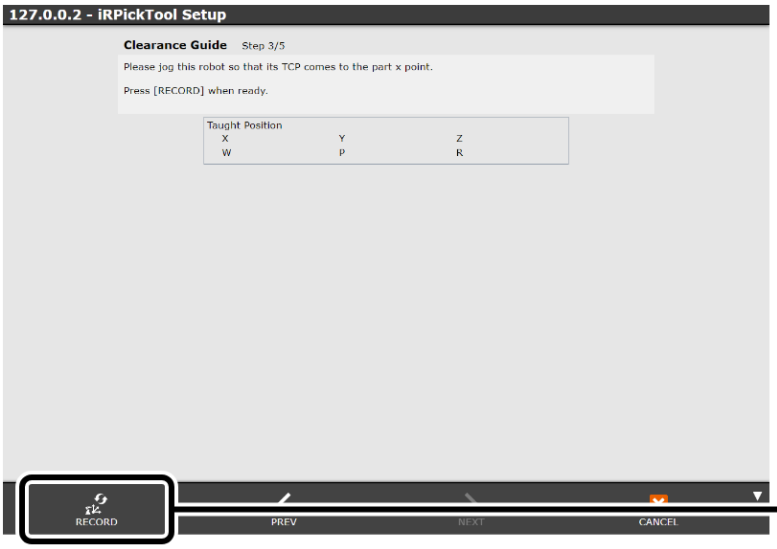
8 Click [Record].

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

9 Click [NEXT].



The following screen appears.

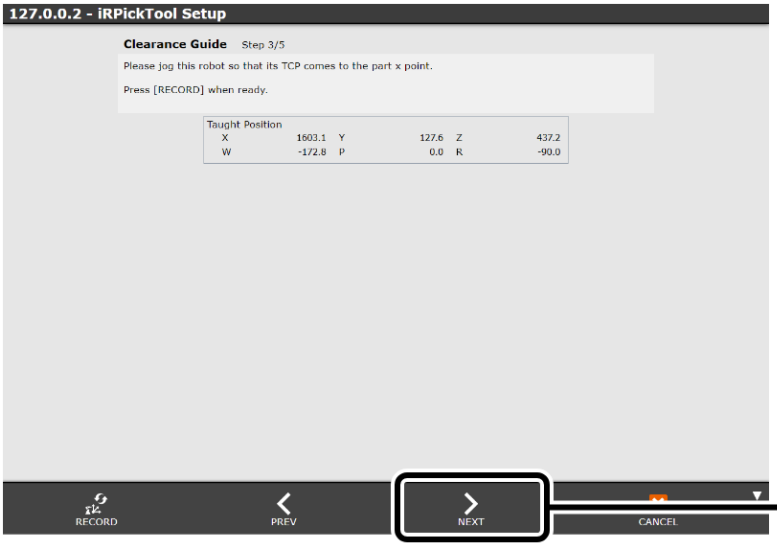


10 Move the robot by jog operation, and touch up the X direction of the part.

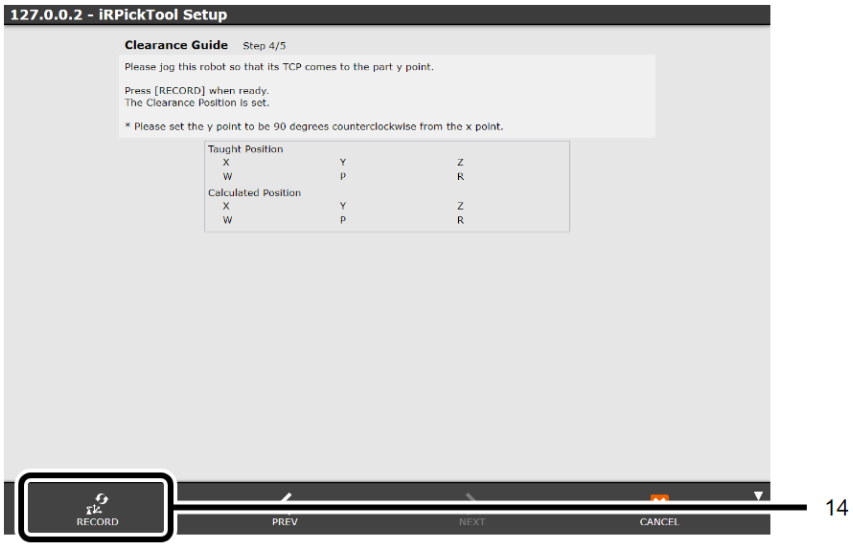
11 Click [Record].

The touch-up position is displayed and [NEXT] is enabled.

12 Click [NEXT].



The following screen appears.

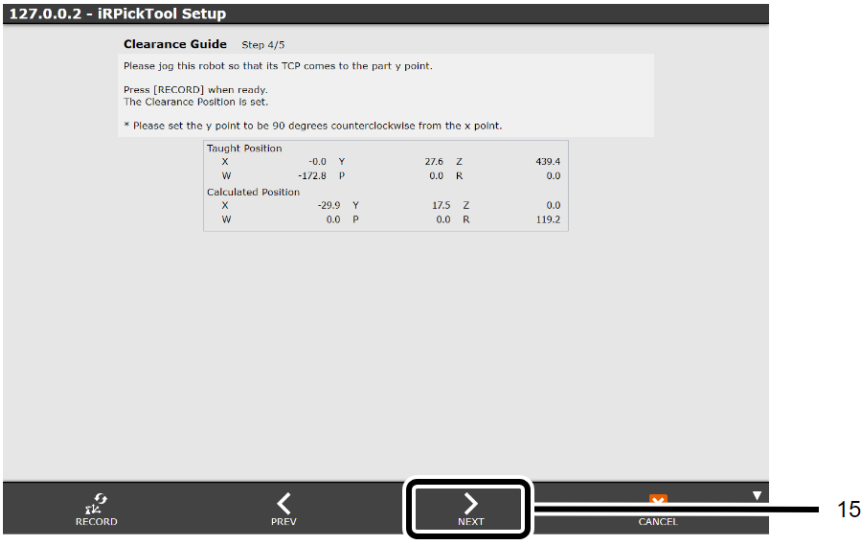


13 Move the robot by jog operation, and touch up the Y direction point of the part.

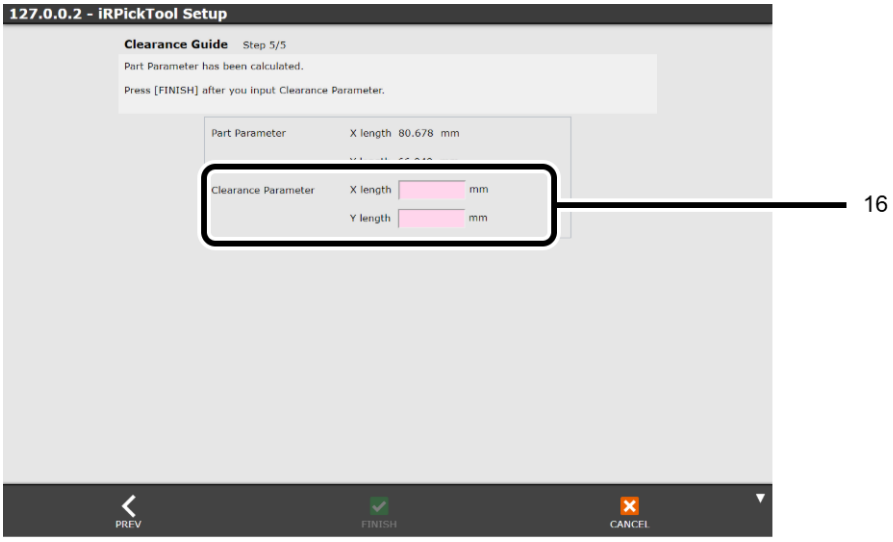
14 Click [Record].

The touch-up position and calculated position are displayed, and [NEXT] is enabled.

15 Click [NEXT].



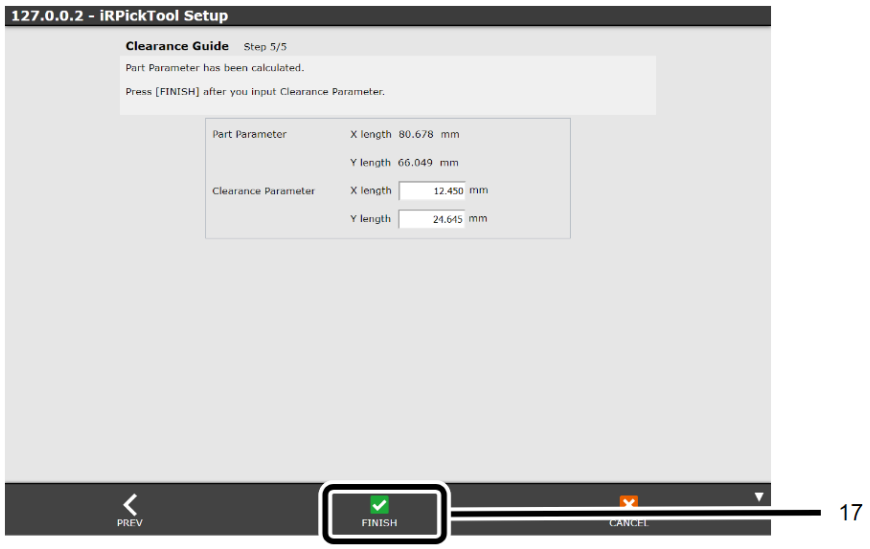
The following screen appears.



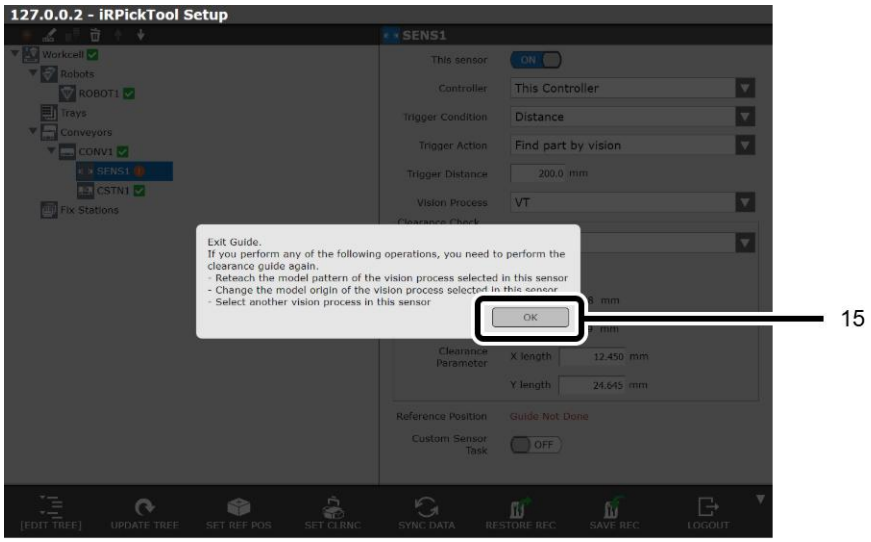
16 Enter [X length] and [Y length] of [Clearance Parameter].

[FINISH] is enabled.

17 Click [FINISH].



The following screen appears.



18 Confirm the content of the displayed message, and click [OK].

The screen returns to the original setup screen.

[Guide Complete] is displayed in green next to [Clearance Position], and the values set in the guide are entered in [Part Parameter] and [X length] and [Y length] of [Clearance Parameter].

CAUTION

When any of the following procedures is performed in a vision process used in the sensor for which a clearance check is set, it is necessary to perform clearance guiding again.

- Reteaching the model
- Changing the model origin

8.10.4.6. Setting up a conveyor station

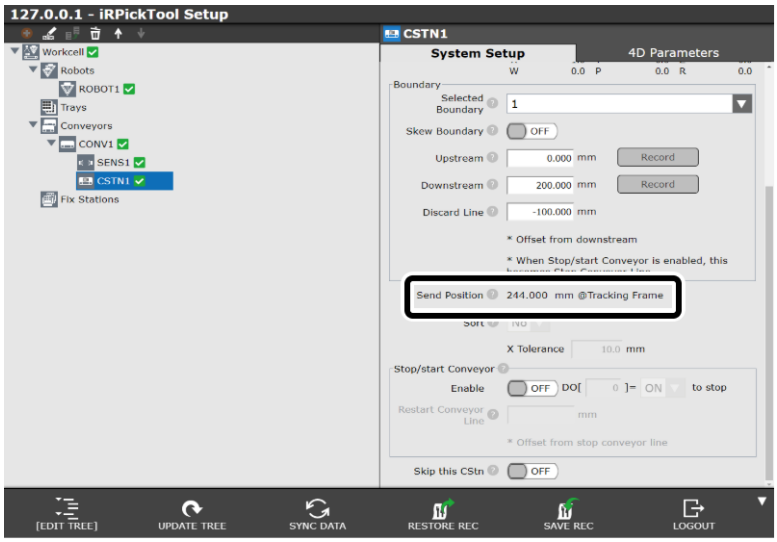
In the case of a line conveyor

[Send Position] is displayed.

The send position refers to the position on this conveyor station where the part data is sent to the downstream station, and is expressed as a position on the tracking frame.

If setup of the clearance check in "I.8.10.4.4. Setting up a sensor(P.640)", setup of the tracking area (upstream, downstream), and setup of the discard line have been completed, values are automatically set in [Send Position] for all the conveyor stations corresponding to the sensors for which clearance check is set.

There are no special notes concerning the setup of a conveyor for a clearance check. See "I.5.6. Setting up a conveyor station(P.318)".



In the case of a circular conveyor

[Send Line] is displayed as shown in the figure below.

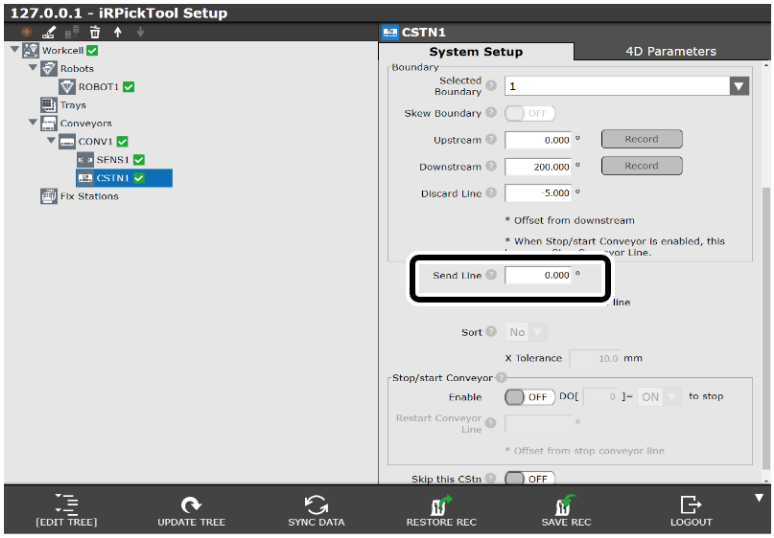
The send line refers to the position on this conveyor station, where the part data is sent to the downstream station, and is expressed as an offset from the discard line.

In the case of a circular conveyor, send line values are not automatically set, so refer to the following notes to set the values.

There are no special notes concerning the setup of a conveyor for a clearance check. See "I.5.6. Setting up a conveyor station(P.318)".

CAUTION

The send line position changes depending on whether parts flow on the inside or outside of the circular conveyor.
For example, when parts flow near the center of the conveyor, it is necessary to set a large value for the send line.



8.10.4.7. Setup of a fixed station

There are no special notes concerning the setup of a fixed station for a clearance check. See "I.5.7. Setting up a fixed station(P.332)".

8.11. Diagnostic function

This section describes the function that collects information necessary for investigation when issues occur in iRPickTool.

8.11.1. Collecting the iRPickTool diagnostics data

iRPickTool diagnostics helps users to collect several types of diagnostic data. You can check the diagnostic data log yourself, but for quicker issue resolution, we recommend sending the diagnosis log to us.

When users see an issue in the application such as robots not picking parts, picking the wrong parts (e.g., wrong model_id), they should first collect the diagnostics data. The data can be collected at any time including while production is running.

There are three types of logs that can be collected.

- Queue Log (PQLOG.TXT):

  Contains data on the sequence of events such as the id and timestamp of each part that was put in the queue, allocation of the part when the PKCSGETQ call is made, load balance details and the parts that are sent to the next conveyor station.

- Sensor Log (PSLOG.TXT):

  Shows starts, stops, etc. of the sensors.

- Diagnostics Log (IRPKDIAG.TXT):

  Checks the setup values of all the iRPickTool items in the tree view for each object such as robot, conveyor station, sensor, fixed station in detail.

You can collect iRPickTool Diagnostic data from the "Workcell object" in the iRPickTool tree view.

To run this feature do the following:

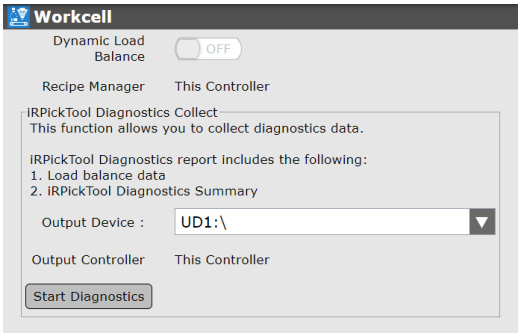
1 Select the output device.

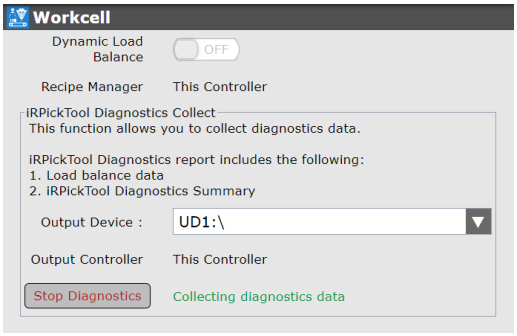
2 **Press the “Start Diagnostics” button. Once you start the diagnostics, the button will change to Stop Diagnostics.**

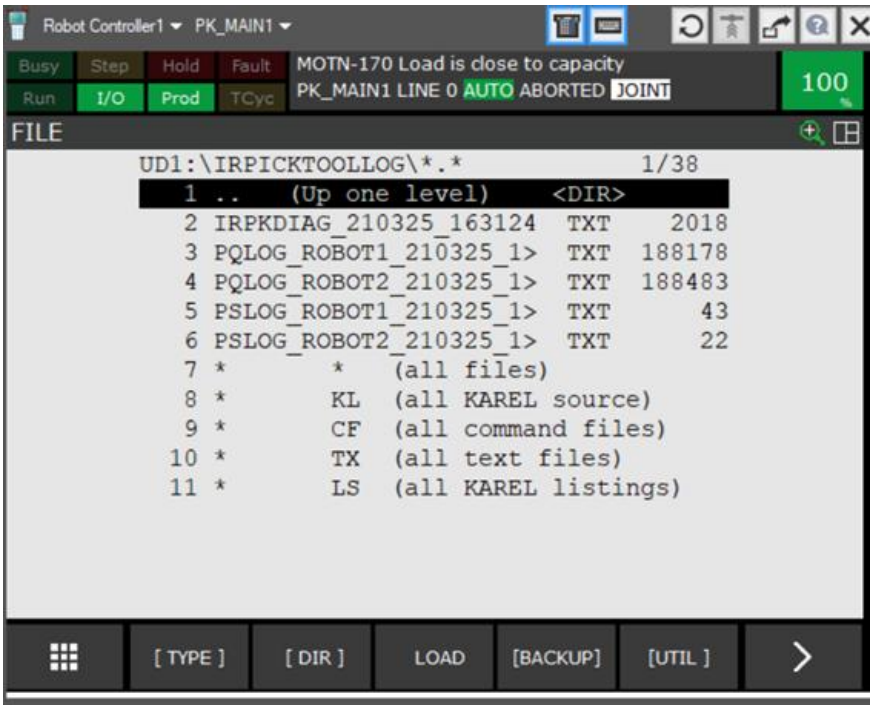
3 **Start production if you wish to collect data while production is running.**

4 **Press the “Stop Diagnostics” button.**

5 **The logs can be found in the output device**







Logs can be collected on the following devices: UD1, MC and UT1. iRPickTool creates a directory called irpicktoollog where all logs will be saved, in the root directory of the output device.

You can also collect data using the teach pendant of any of the robots in the workcell. iRPickTool diagnostic data obtained by each controller in the workcell will be output into the selected device on the controller which has been set in “Output Device” and “Output Controller”.

Each log file has a timestamp in the file name. Therefore, you can collect the data multiple times without overwriting previously collected logs.

A text editor can normally be used to open and review the log entries.

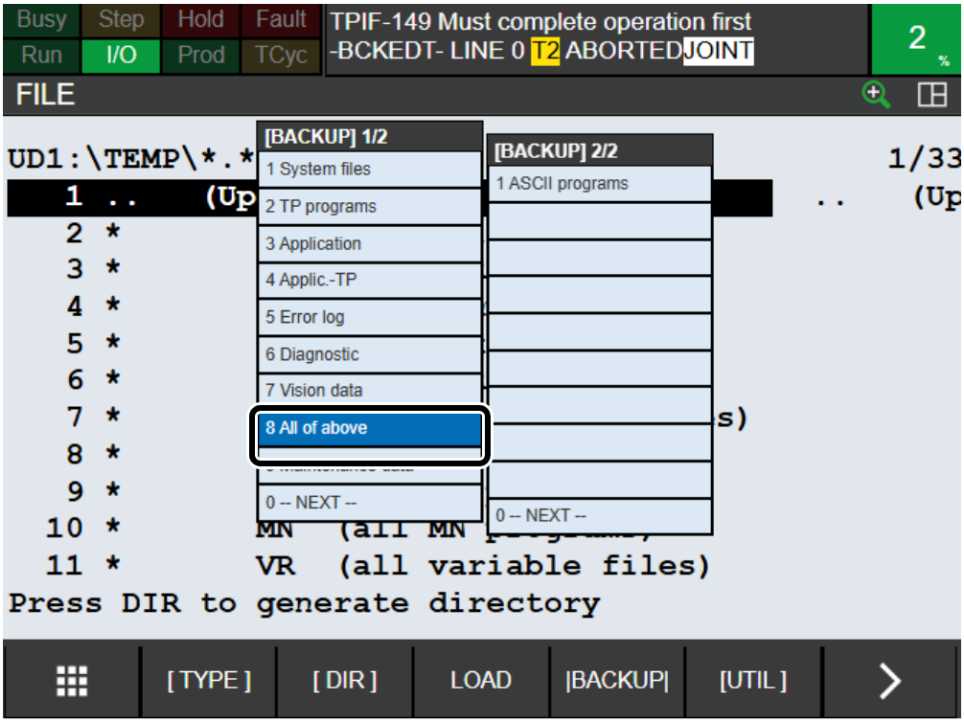
8.11.2. Collecting debug data

To debug shell-related issues involving recipe operations, Dynamic Load Balance, encoder synchronization, and more, iRPickTool provides a switch for recording debug messages.

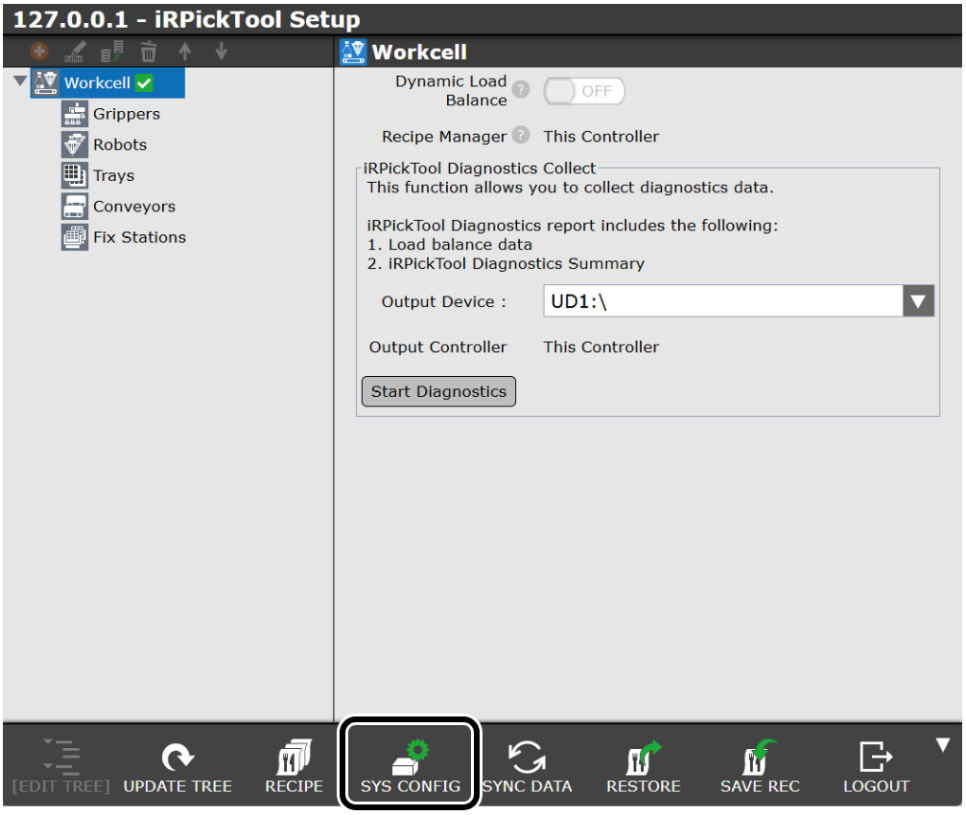
⚠CAUTION
Please contact FANUC before using this function.

Once recording starts, messages are continuously recorded, and the recorded information is output to the robot's backup.

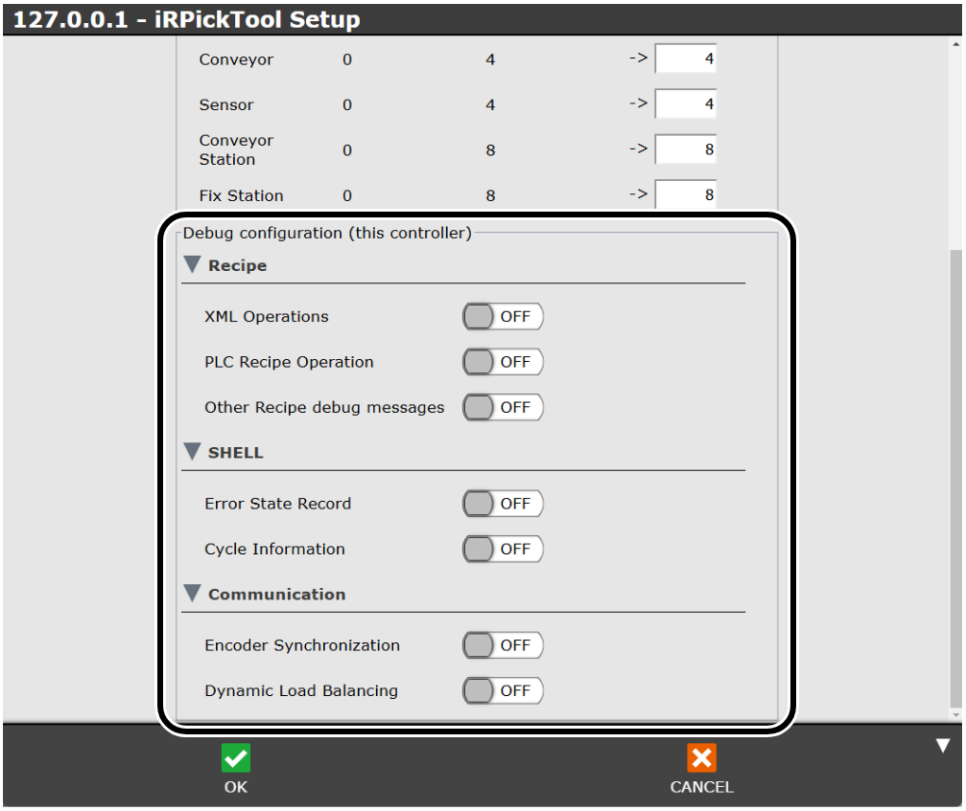
To create a robot backup, select [8 All] when creating the backup data on the [File] screen. Once the debugging is complete, disable the switch.



Click [Configuration] on the workcell setting screen.



Scroll to the [Debug Settings (Login Controller)] section and enable the toggle switch for the item you want to debug to start recording.



[Recipe]

If an error occurs during recipe operation by means of a PLC, enable the [PLC Recipe Operation] toggle switch. If the error or warning is related to file operations, enable the [XML Operation] toggle switch. For other cases, enable the [Other Recipe Debug Messages] toggle switch.

[Shell]

If issues such as Dynamic Load Balance not functioning or being able to perform setting operations from the iRPickTool settings screen even during production occur, it is possible that the shell is not operating correctly. Enable the [Error State Logging] toggle switch and the [Cycle Information] toggle switch.

[Communication]

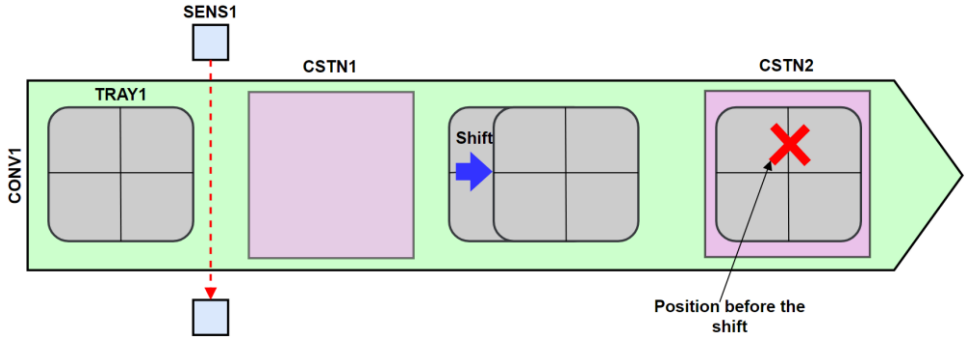
Record communication-related processes specific to RPickTool. If there are issues with shared encoder count discrepancies or updates, enable the [Encoder Synchronization] toggle switch. When an issue occurs with dynamic load balancing, enable the [Dynamic Load Balancing] toggle switch.

8.12. Tray redetection

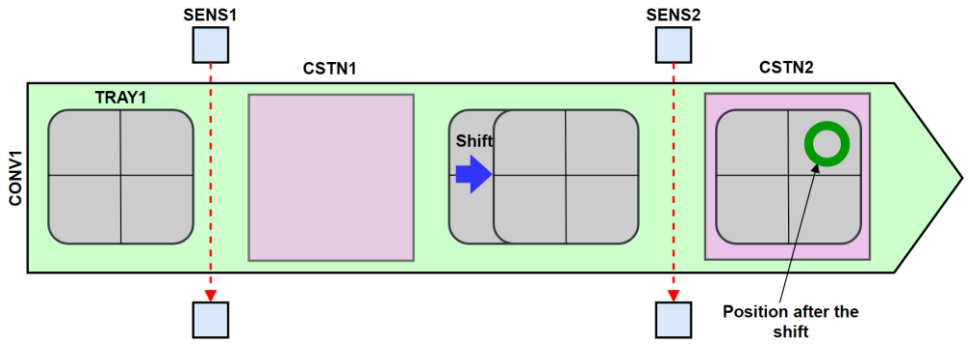
When performing tray tracking with multiple robots, if the position of the tray shifts during work on the upstream robot, it may cause failures in the downstream robot's operations.

To avoid this, a sensor is installed before the downstream robot to redetect the tray and correct the shift.

Without this function, if the tray shifts upstream as shown in the figure below, the downstream robot will perform tasks at the position before the shift, resulting in failure.



By using this function, even if the tray shifts upstream as shown in the figure below, the tray will be redetected by the downstream sensor, allowing the work to be performed at the correct position.



NOTE

This function is supported in 7DH1/15 (V10.10P/15) and later versions.

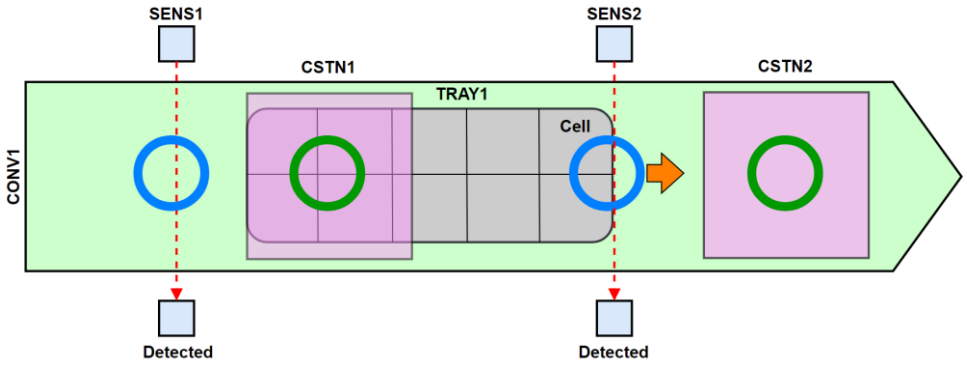
8.12.1. Possible cases

The following shows the possible tray detection cases for this function, along with the tracking availability and tracking accuracy for each case.

Case 1

This is the case where the tray is detected by the downstream sensor and then crosses the discard line of the upstream conveyor station.

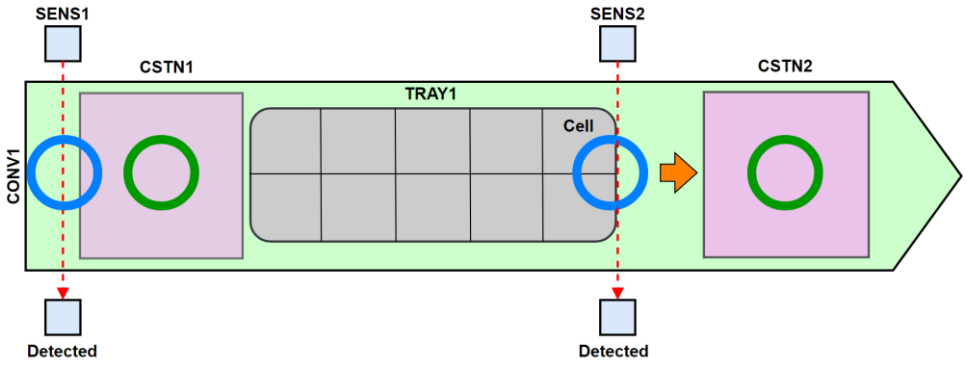
Tracking is possible, and there are no issues with accuracy.



Case 2

This is the case where the tray crosses the discard line of the upstream conveyor station before it is detected by the downstream sensor.

Tracking is possible, and there are no issues with accuracy.

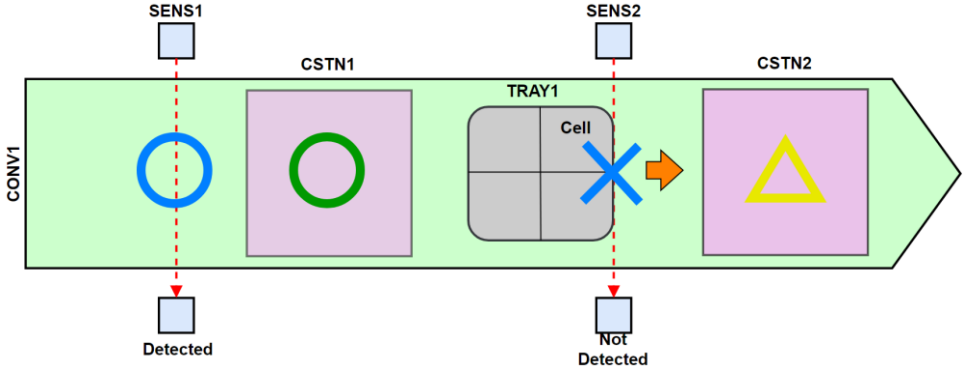


Case 3

This is the case where the tray is detected by the upstream sensor but not detected by the downstream sensor.

The position detected by the upstream sensor is tracked at the downstream conveyor station.

Tracking is possible, but the accuracy decreases.



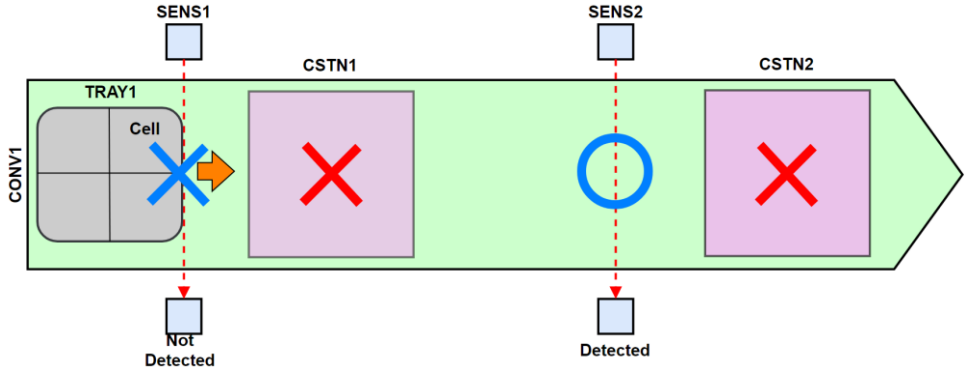
Case 4

This is the case where the tray is not detected by the upstream sensor but is detected by the downstream sensor.

If the tray is not detected by the most upstream sensor, it will be treated as not detected by all other sensors.

At this time, the tray detection count will be 0, and it will be processed as if no misses have occurred.

In this case, tracking cannot be performed.



8.12.2. Restrictions

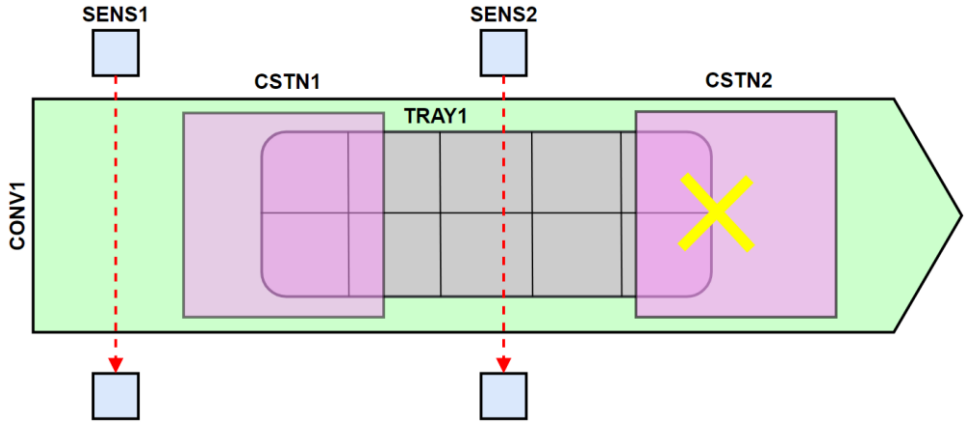
The following restrictions are applied to tray redetection.

- Supports only line conveyors. Circular conveyors or servo conveyors are not supported.
- Supports trays only. Parts are not supported.
- The only action supported is [Put part in queue]. [Find part by vision] is not supported.
- Only [DI] and [RI] are supported as trigger conditions.
- The sensors need to be connected to the robot controller of the conveyor station downstream from that sensor. Additionally, multiple sensors cannot be connected to the same robot controller.
- Tray redetection and duplicate check cannot be used together.
  - When tray redetection is enabled, if duplicate parts are detected by the sensor, the last detected part will be retained.
  - The tolerance for tray redetection is used to determine duplicate parts (refer to "I.5.4. Setting up a conveyor(P.276)").
- If the tray is not detected by the most upstream sensor, it will be treated as not detected by all other sensors.
- Tray communication volume reduction cannot be used in combination with tray redetection.

  For details on tray communication volume reduction, refer to "I.5.8.1.14. PKENBSPFUNC.PC(P.337)" and "I.5.8.1.15. PKDSBSPFUNC.PC(P.337)".

Invalid cases

- If the trays enter the downstream conveyor station before passing the discard line of the upstream conveyor station, the function will be invalid.



- If the trays move and the spacing between trays falls below the tolerance for tray redetection, the function will be invalid.

  When detected by the sensor, the downstream trays may be deleted.

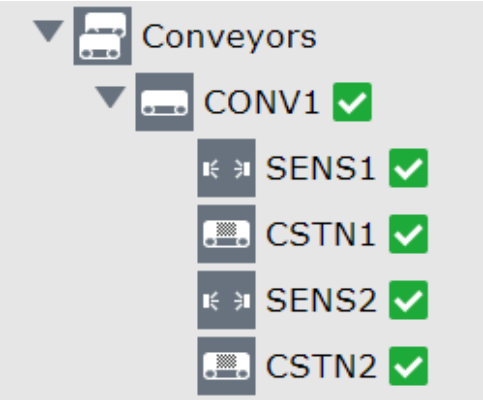
Conditions under which an error may occur when using PKWCSTART

In conveyors where tray redetection is enabled, an error occurs in the following cases while using PKWCSTART.

- When no conveyor station with a redetection sensor exists

  A conveyor station with a redetection sensor refers to the conveyor station located just below the sensor that performs tray redetection.

  In the following tree, CSTN2 is the conveyor station with the redetection sensor.



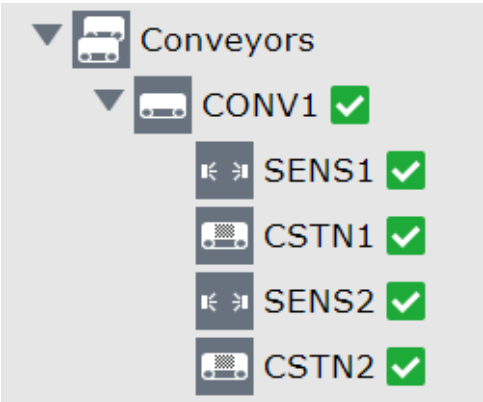
- When there is a sensor for which actions other than “Put part in queue” is specified

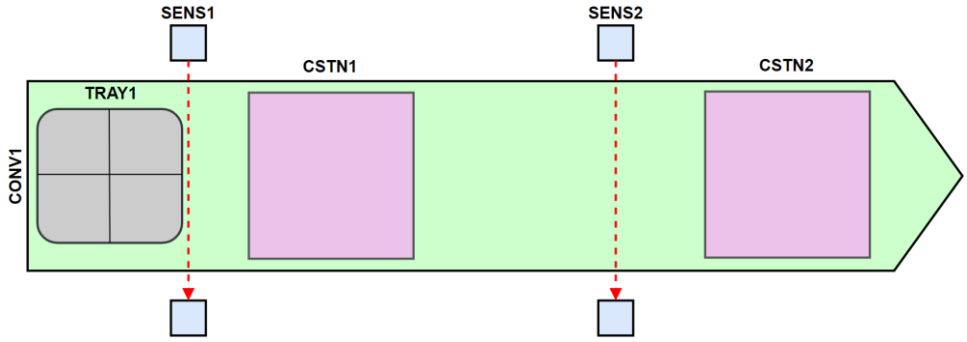
8.12.3. Startup procedure

All procedures other than the ones below are the same as the standard iRPickTool settings. The same TP programs as usual are also used.

1 **Create the conveyor object according to the system.**

In the case of the figure below, the tree structure is as follows.

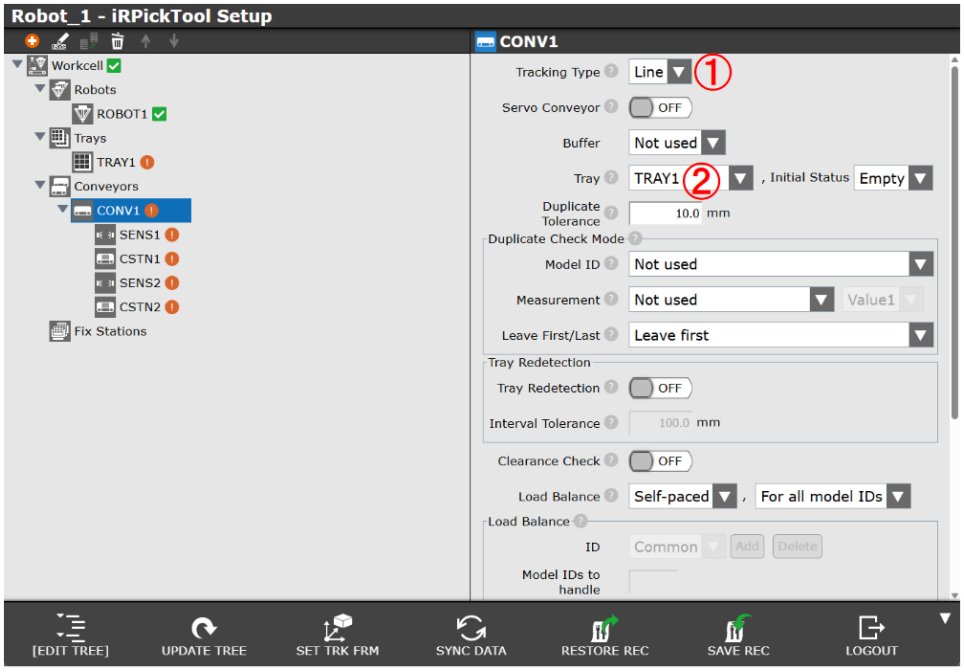




2 **Configure the following settings in the [System Setup] sheet.**

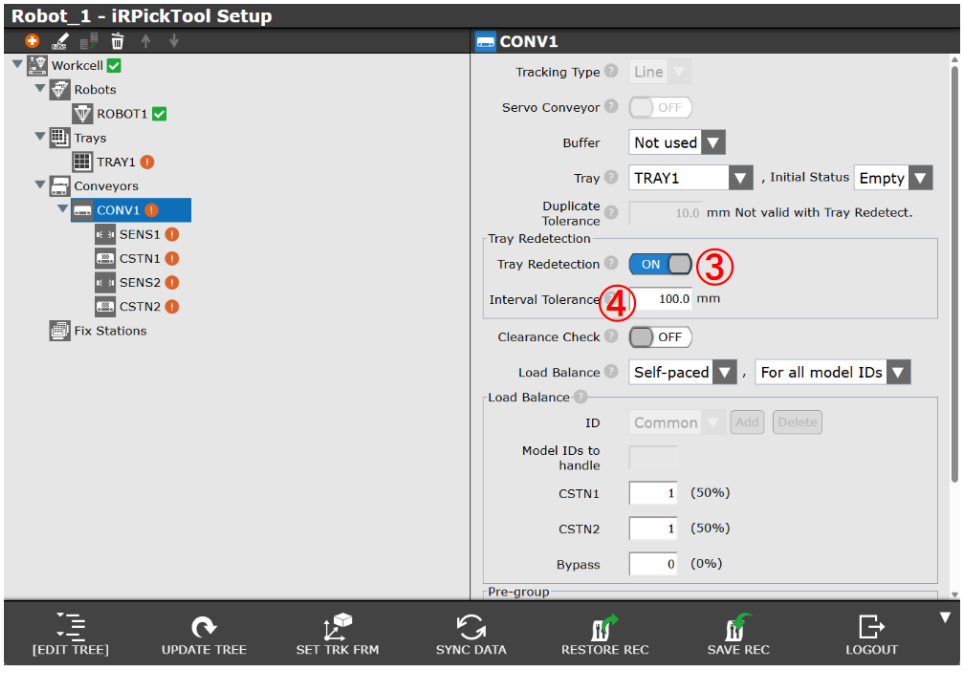
1 Select [Line] for [Tracking Type].

2 Select the tray to be used.



3 In the “Tray Redetection” area, set “Tray Redetection” to “Enable”.

4 Enter the maximum possible displacement distance of the tray in [Interval Tolerance].



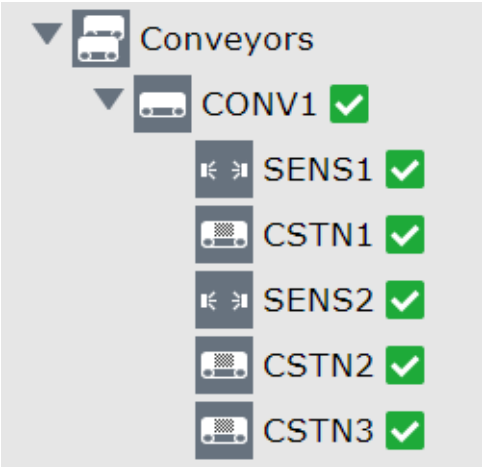
CAUTION

For [Interval Tolerance], enter a value smaller than the spacing between trays. If trays arrive at intervals shorter than the Interval Tolerance, the downstream trays will be removed at the time of tray detection.

3 Perform robot position teaching relative to the reference position of the nearest upstream sensor for each conveyor station.

In the configuration shown below, perform reference position teaching for SENS1 using the robot assigned to CSTN1. Then, perform position teaching for the robot selected in CSTN1.

Subsequently, perform reference position teaching for SENS2 using the robot assigned to CSTN2. Then, perform position teaching for the robots selected in CSTN2 and CSTN3.



8.13. Tracking using robot-mounted camera

This function can be used to detect and track parts transferred on the conveyor using robot-mounted cameras.

2D cameras and 3D vision sensors can be used for tracking using robot-mounted cameras..

NOTE

This function is supported in 7DH1/10 (V10.10P/10) and later versions.

8.13.1. Restrictions

The following restrictions are imposed on tracking using robot-mounted cameras.

- Supports only line conveyors.
- Does not support multiple robots.
  - This function cannot pass a part found with a robot-mounted cameras to a downstream robot. Therefore, tracking cannot be performed using multiple robots.
- Cannot use trays.
- The following trigger conditions and action are supported.
  - Only [DI] and [RI] are supported as trigger conditions.
  - Only [Put part in queue] is supported as action.

8.13.2. Teaching method

This section describes how to start up the system described in "I.2. iRPickTool Basic (basic functions) startup procedures(P.26)", which has been changed to detect parts through tracking using 2D robot-mounted cameras on the infeed conveyor.

When using 3D vision sensors, create [3DV Sensor] instead of [2D Camera] in the procedure described in "I.8.13.2.6. Setting up a robot-mounted camera(P.665)" and [3DV Single-View Vis. Proc] instead of [2-D Single-View Vision Process] in the procedure described in "I.8.13.2.7. Creating a vision process(P.668)" and perform teaching with the same procedure.

MEMO

When using both the Snap Tool and Depth Snap Tool in 3D Vision sensors, create a TP program to call IRVRPOSCHK(0) and execute it in advance.

This section describes procedures different from "I.2. iRPickTool Basic (basic functions) startup procedures(P.26)". Perform teaching, referring to the following.

STEP	Item name	Procedure
1	Creating a robot program	Follow the procedure described in "I.8.13.2.3. Creating a robot program(P.660)".
2	Setting up a workcell	The conventional procedure remains unchanged. Follow the procedure described in "I.2.3.1. Setting up a workcell(P.60)".
3	Setting up a tray	The conventional procedure remains unchanged. Follow the procedure described in "I.2.3.2. Setting up a tray(P.67)".
4	Setting up a conveyor	The conventional procedure remains unchanged. Follow the procedure described in "I.2.3.3. Setting up a conveyor(P.71)".
5	Setting a tracking frame	The conventional procedure remains unchanged. Follow the procedure described in "I.2.3.4. Setting a tracking frame(P.72)".
6	Setting up a sensor	For the infeed conveyor, follow the procedure described in "I.8.13.2.4. Setting up an infeed conveyor sensor(P.661)". For the outfeed conveyor, the conventional procedure remains unchanged. Follow the procedure described in "I.2.3.5. Setting up a sensor(P.80)".
7	Setting the sensor position	For the infeed conveyor, follow the procedure described in "I.8.13.2.5. Setting an infeed conveyor sensor position(P.662)". For the outfeed conveyor, the conventional procedure remains unchanged. Follow the procedure described in "I.2.3.6. Setting the sensor position(P.84)".
8	Setting up the vision system	Follow the procedures described in "I.8.13.2.6. Setting up a robot-mounted camera(P.665)" and "I.8.13.2.7. Creating a vision process(P.668)".
9	Setting up a conveyor station	The conventional procedure remains unchanged. Follow the procedure described in "I.2.3.8. Setting up a conveyor station(P.106)".
10	Setting up a tool frame for the hand	The conventional procedure remains unchanged. Follow the procedure described in "I.2.3.9. Setting up a tool frame for the hand(P.111)".
11	Setting up a reference position and teaching a robot position	For the infeed conveyor, follow the procedure described in "I.8.13.2.8. Setting up a reference position for the infeed conveyor, and teaching a robot position(P.670)". For the outfeed conveyor, the conventional procedure remains unchanged. Follow the procedure described in "I.2.3.10. Setting up a reference position and teaching a robot position(P.113)".

8.13.2.1. Definitions of registers used

In this example, the following registers are used. Change the definitions of register numbers as required.

Table 8.13.2.1-1 Register list

R[11:CONV1 ID]=1	CONV1 (pick) ID
R[12:CONV2 ID]=2	CONV2 (place) ID
R[21:Pick AP_LD]=30	Pick approach distance specification [mm]
R[22:Pick RT_LD]=30	Pick retract distance specification [mm]
R[23:Pick Wait]=.05	Index delay at pick position [sec]
R[24:Pick CNT]=0	Pick position CNT specification
R[25:Pick TB]=.1	TIME BEFORE time (VACUUM_ON) [sec]
R[26:Pick Speed1]=4000	Pick program speed 1 [mm/sec]
R[27:Pick Speed2]=4000	Pick program speed 2 [mm/sec]
R[28:Pick Speed3]=2000	Pick program speed 3 [mm/sec]
R[29:Pick Speed3]=4000	Pick program speed 0 (for movement to snap position) [mm/sec]
R[31:Drop AP_LD]=30	Place approach distance specification [mm]
R[32:Drop RT_LD]=30	Place retract distance specification [mm]
R[33:Drop Wait]=.05	Index delay at drop position [sec]
R[34:Drop CNT]=0	Drop position CNT specification

R[35:Drop TB]=.1	TIME BEFORE time (VACUUM_OFF) [sec]
R[36:Drop Speed1]=2000	Placement program speed 1 [mm/sec]
R[37:Drop Speed2]=2000	Placement program speed 2 [mm/sec]
R[38:Drop Speed3]=4000	Placement program speed 3 [mm/sec]

Table 8.13.2.1-2 Position Register list

PR[1:HOME POS]	Perch Position for all groups
PR[5:Snap Pos]	Snap position
PR[6:Vofs Pick Pos]	Position of pick position multiplied by vision offset
PR[7:Pick Pos]	Pick position
PR[8:Drop Pos]	Drop position
PR[11:Pick Apr Z]	Offset from pick position
PR[12:Drop Apr Z]	Offset from drop position

Table 8.13.2.1-3 Vision Register list

VR[1]	For getting the pick position from a pick program
VR[2]	For getting the drop position from a drop program
VR[3]	For getting the detection result of the Robot-Mounted Camera

8.13.2.2. Definitions of coordinate systems used

In this example, the following coordinate systems are used. Change the definitions of the coordinate system numbers as required.

Table 8.13.2.2-1 User frame list

User frame [9]	Vision Process offset frame



MEMO

Since the values for the Vision Process offset frame are set in the subsequent procedures, initial values do not need to be set.

Table 8.13.2.2-2 Tool frame list

Tool frame [1]	Hand TCP

8.13.2.3. Creating a robot program

Create the following programs as described in "I.2.2.5. Creating a robot program(P.42)".

- Main program
- Tracking program
- Vacuum ON/OFF program

For the pick program (PT_PICKCS), change the program, referring to the following.

```
 1: UFRAME_NUM=0 ;
 2: UTOOL_NUM=1 ;
 3: ;
 4: LBL[101:QUE check] ;
 5: STOP_TRACKING ;
 6: !PKCSGETQUE (Get part information)
 7: !*1:Register number for conveyor station ID
 8: !*2:Usually 1. 2 or more in the case of consecutive acquisition
 9: !*3:Index delay msec
10: !*4:Vision Register number for part information
11: !*5:Process status register number
```

```
12: !*6:Model ID (Can be omitted)
13: CALL PKCSGETQUE                                  Wait until a part arrives.
      (CStn ID=R[11:CONV1 ID],
      Consec Flag=1,Timeout(ms)=100,
      Offset VR=1,Stat Reg=2) ;
14: IF R[2:GETQ Status Pick]=0,JMP LBL[111] ;
15: IF DI[1:CYCLE_STOP]=ON,JMP LBL[901] ;
16: JMP LBL[101] ;
17: ;
18: LBL[111:Vision Operation] ;
19:L PR[5:Snap Position] R[29:Pick Speed0]mm/sec FINE ;
20: ;
21: !Sets the offset frame as the nominal frame ;
22: CALL PKSETNOMFRM(Trk Sch Num=1,                   Write the nominal frame into the user frame
      Ofs Frm Num=9) ;                               used as the offset frame for the vision
                                                     process.
23: VISION RUN_FIND 'VP2S' ;                         Executes a vision process. Writes the offset
24: VISION GET_OFFSET 'VP2S' VR[3] JMP LBL[902] ;    into Vision Register [3].
25: ;
26: !Calculates the position after VOFFSET;
27: CALL PKCALOFSPOS(Src Pos Reg Num=7,Vis Reg Num=3, Writes the position to which the offset for
      Dst Pos Reg Num=6) ;                           Vision Register [3] is applied into Position
28: ;                                                Register [6].
29: JMP LBL[200] ;
30: !Used for pick position teaching ;
31:L PR[7:Pick Pos] 100mm/sec FINE ;
32: LBL[200:Pick operation] ;
33: ;
34:L PR[6:Vofs Pick Pos] R[26:Pick Speed1]mm/sec
      CNT100 Offset,PR[11:Pick Apr Z] ;
35:L PR[6:Vofs Pick Pos] R[27:Pick Speed2]mm/sec
      CNT R[24:Pick CNT] AP_LDR[21]
      TB R[25]sec,CALL VACUUM_ON ;
36: WAIT R[23] ;
37: PAYLOAD[2] ;
38:L PR[6:Vofs Pick Pos] R[28:Pick Speed3]mm/sec
      CNT100 Offset,PR[11:Pick Apr Z]RT_LDR[22] ;
39: ;
40: !PKCSACKQUE (Part information ! process acknowledgement)
41: !*1:Register number for conveyor station ID KAREL program
42: !*2:Processing results. Normally 1
43: CALL PKCSACKQUE(CStn ID=R[11:CONV1 ID],
      Success=1)
44: LBL[901:Pick finished]
45: END ;
46: ;
47: LBL[902:Detection Failed] ;
48: CALL PKCSACKQUE(CStn ID=R[11:CONV1 ID],
       Return=2) ;
```

8.13.2.4. Setting up an infeed conveyor sensor

Set up an infeed conveyor sensor. The following explanation uses the sensor settings for [DI] + [Put part in queue] as an example.

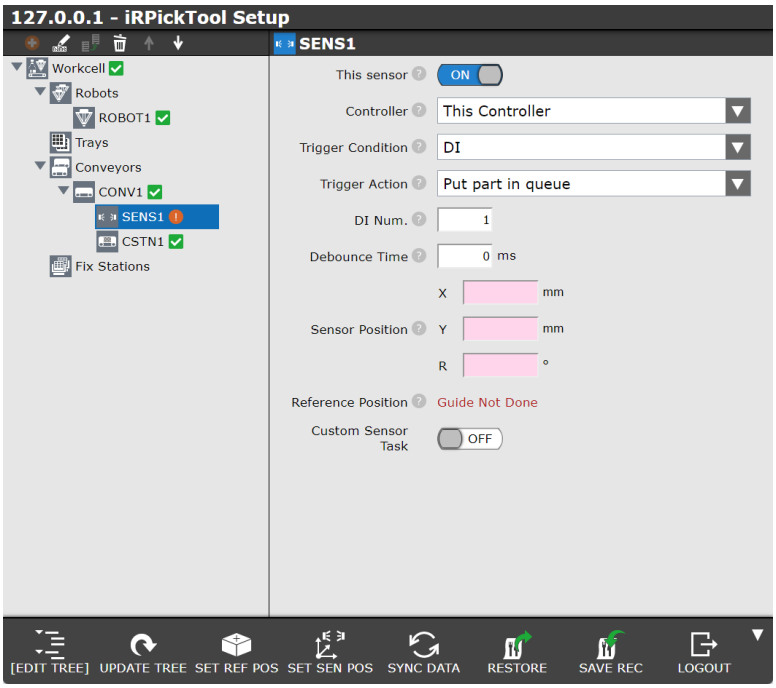
For [RI] + [Put part in queue], interpret “DI” as “RI” in the explanation.

1. **Set up SENS1.**
2. **Set the following items.**

[This Sensor]: Select the [Enable] checkbox
[Controller]: [This Controller]
[Trigger Condition]: [DI]
[Action]: [Put part in queue]
When [Trigger Condition] and [Action] are changed, setting items change accordingly.

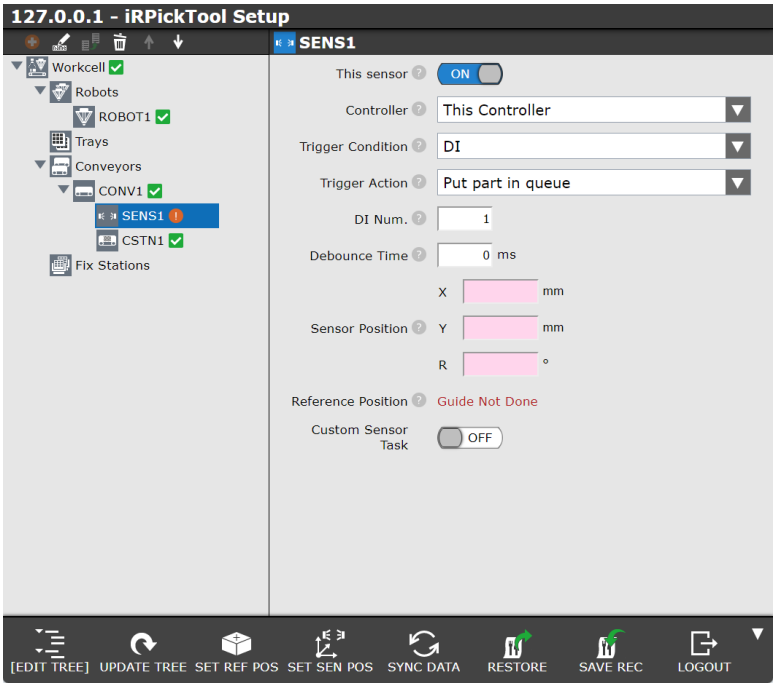
3 Set the following items.

[DI Num]: Enter the DI number that has been assigned to the sensor input.
[Debounce Time]: Enter the time you want to disable input for (to prevent chattering).



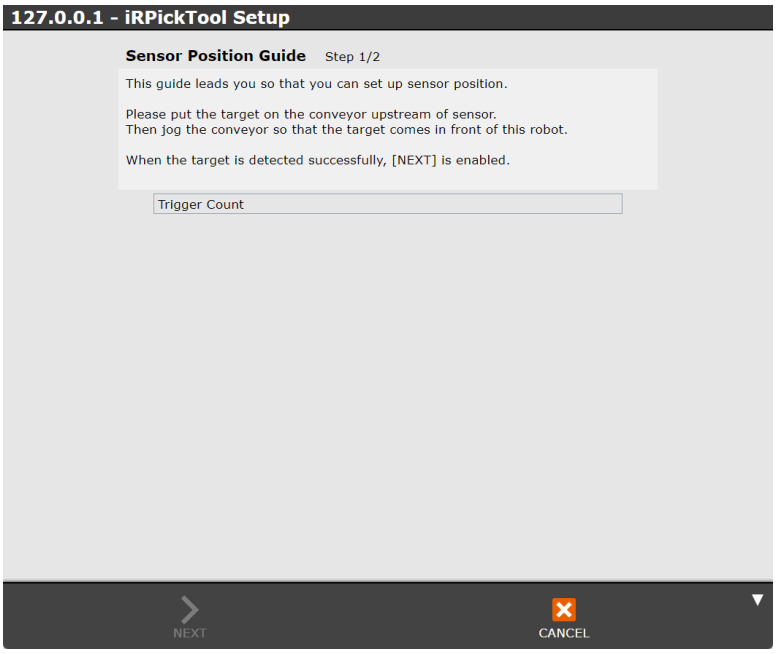
8.13.2.5. Setting an infeed conveyor sensor position

1 Set up SENS1.

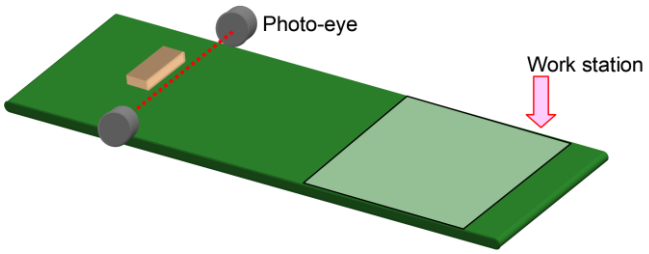


2 Press [SET SEN POS] .

The following screen appears.

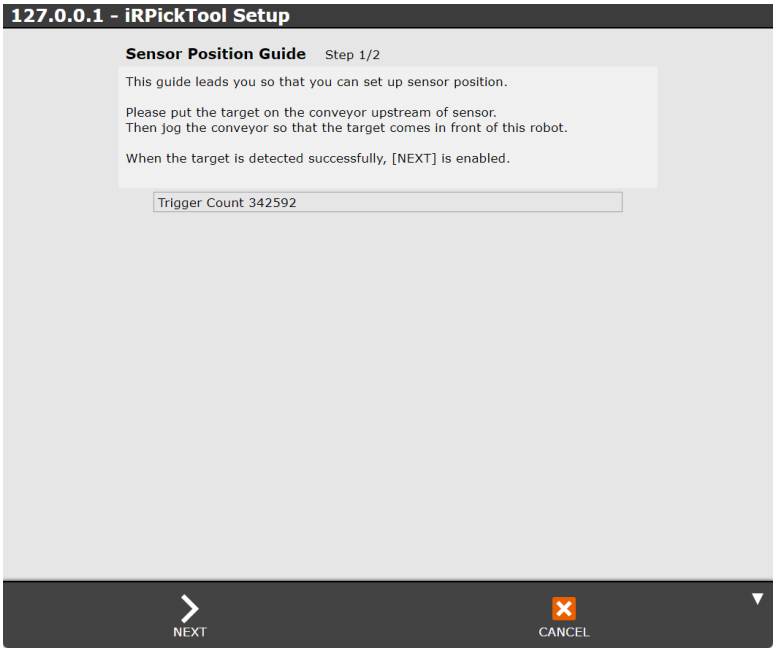


3 Place a part upstream of the photo eye, and move the conveyor until the part is in front of the robot.

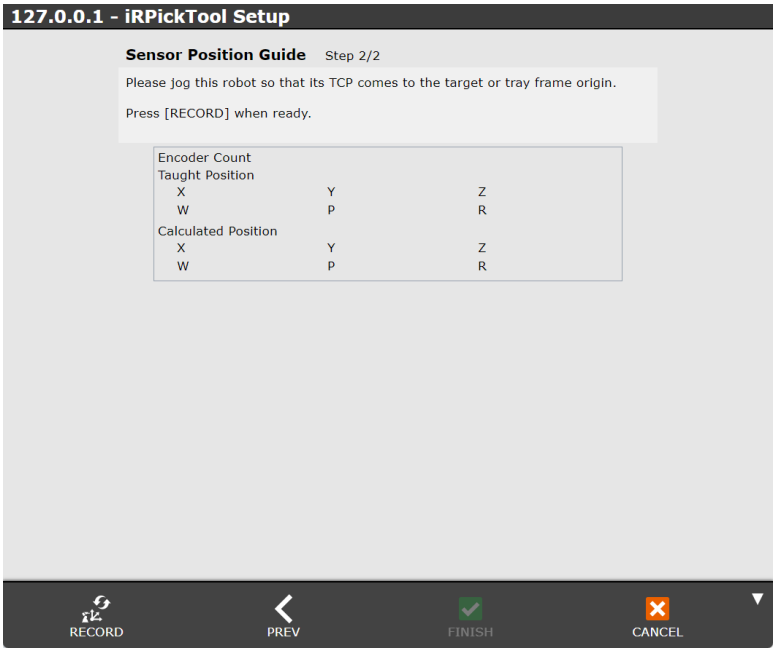


When the sensor detects the part, the encoder value obtained at the time of detection is displayed and [NEXT] is enabled.

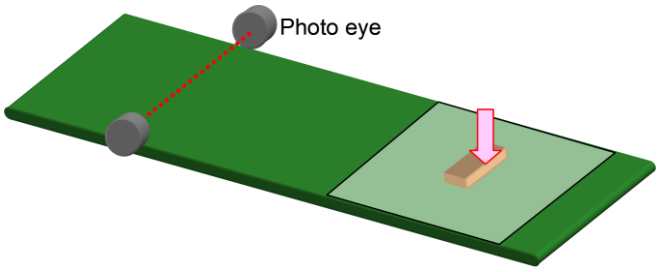
4 Press [NEXT].



The following screen appears.



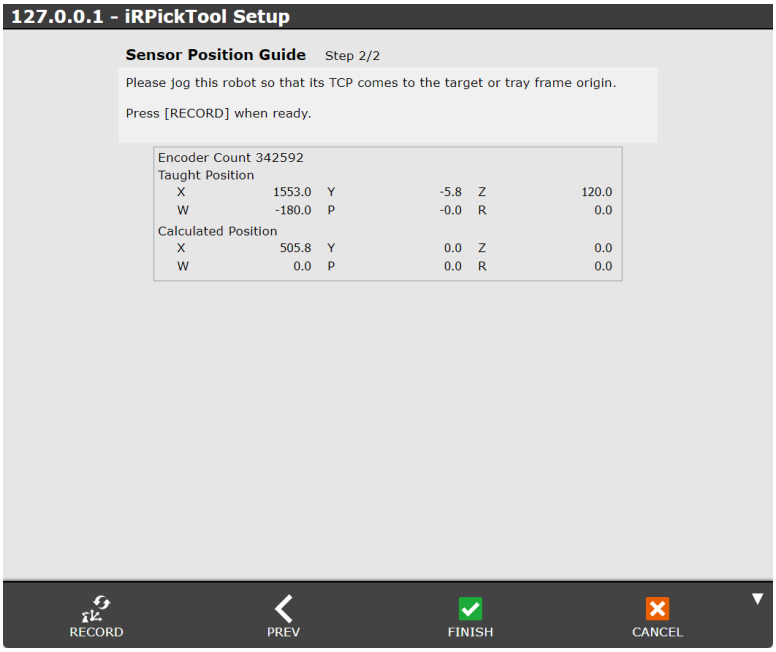
5 Move the conveyor until the part comes in front of the robot. Jog the robot, and touch up the downstream end of the part (the part detected by the photo eye) with the TCP.



6 Press [RECORD].

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

7 Press [FINISH].

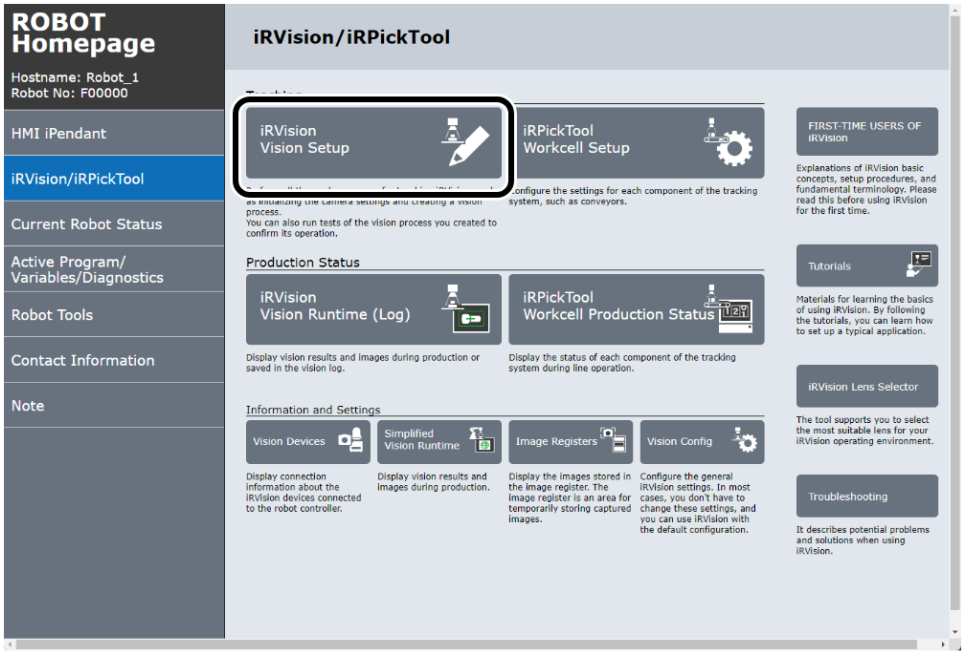


The screen returns to the original setup screen. The calculated value is input for X of the sensor position (the values of Y and R remain unchanged from 0).

8.13.2.6. Setting up a robot-mounted camera

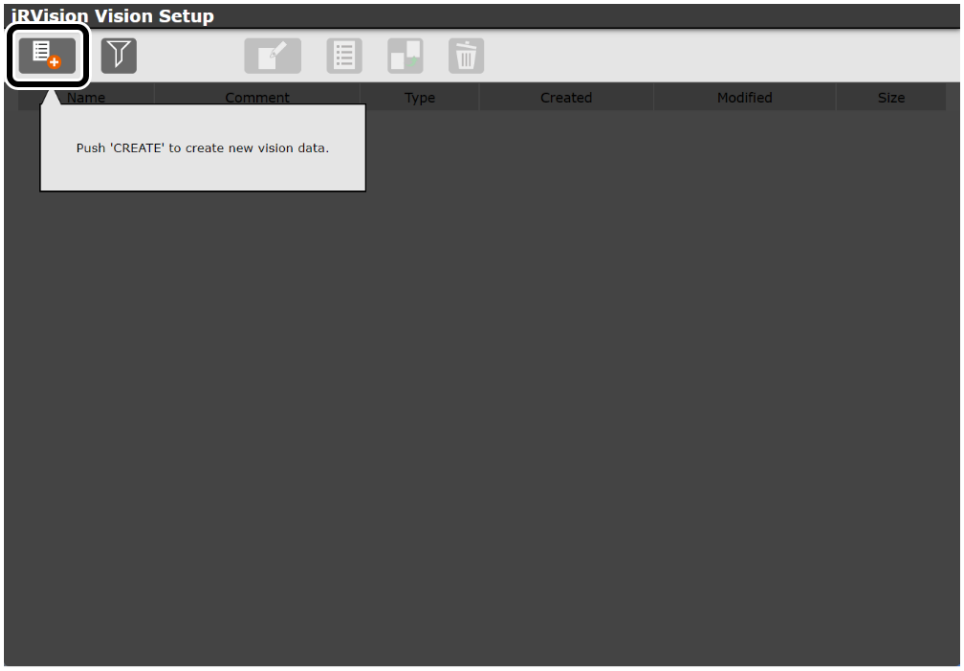
1 Open the robot homepage.

2 Select [iRVision/iRPickTool] in the menu area and click [iRVision Vision Setup] in the content display area.



The [iRVision Vision Setup] screen appears.

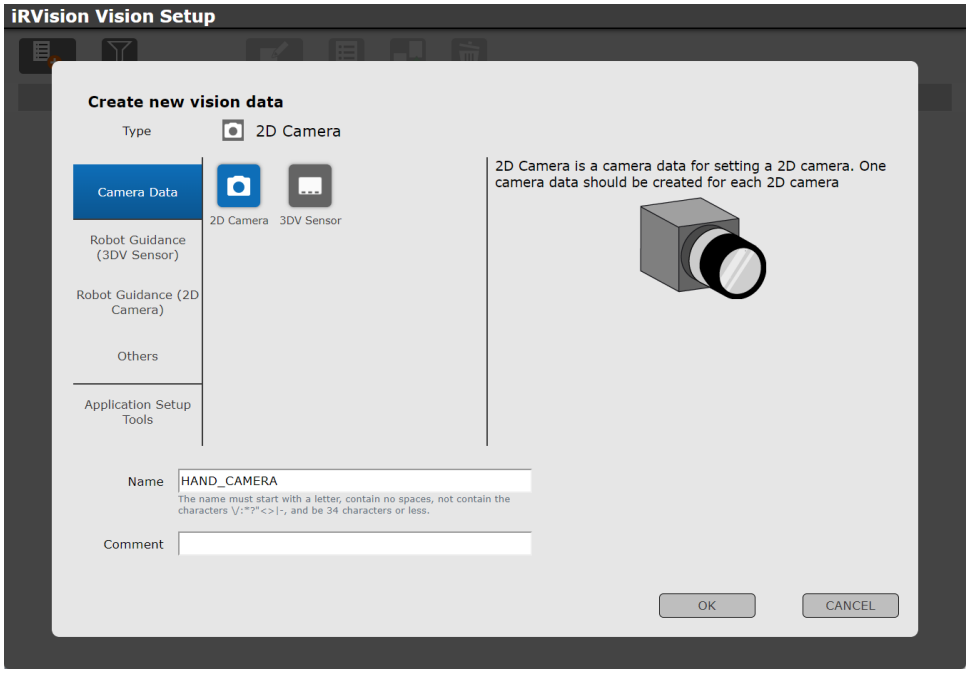
3 Click [CREATE].



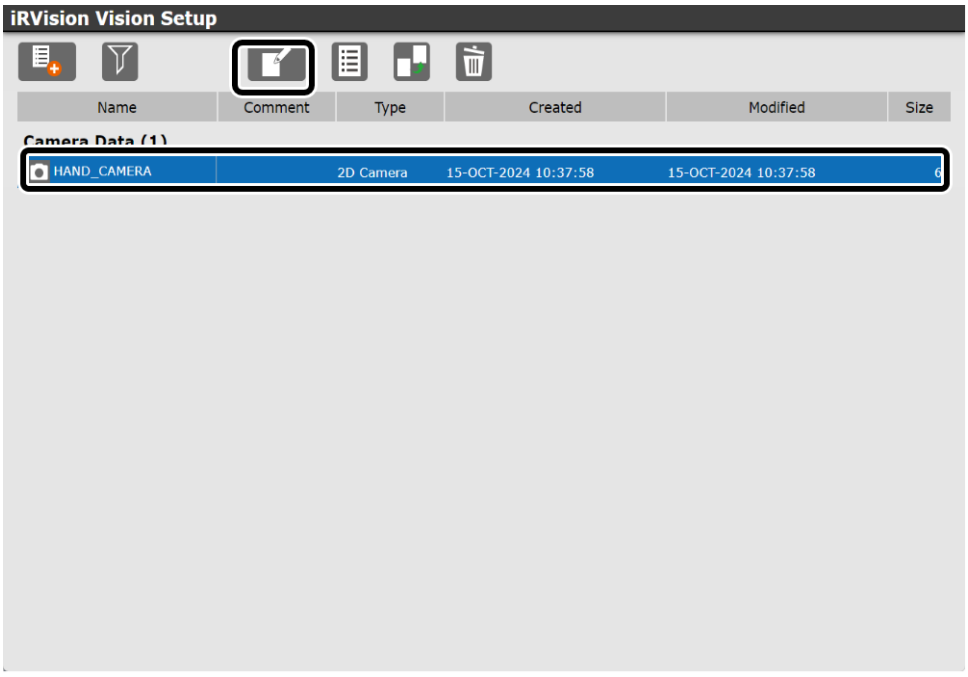
4 Create camera data for a 2D camera.

Enter the camera data name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

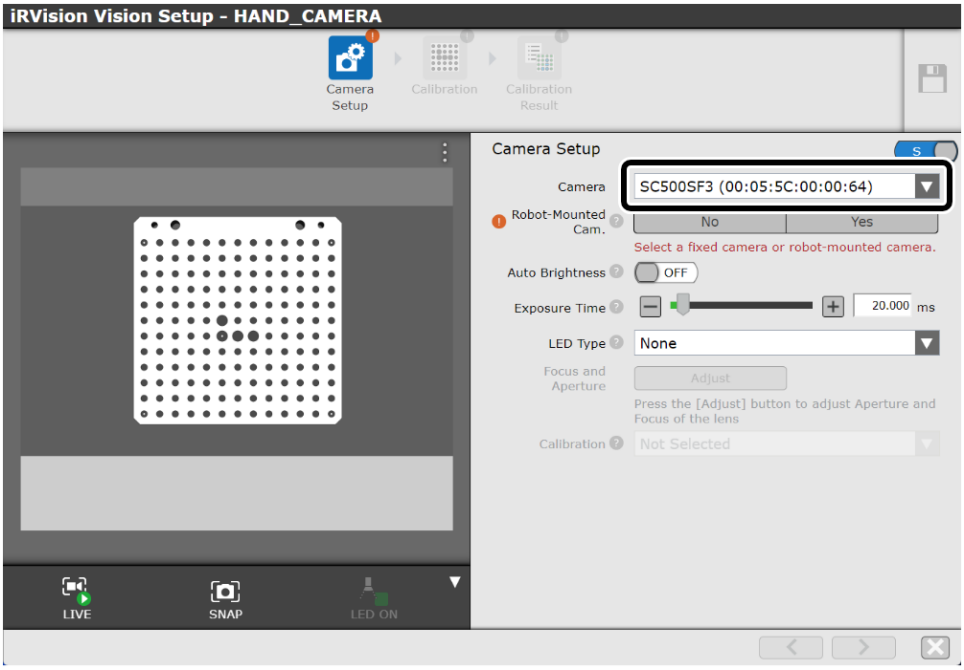
Example: HAND_CAMERA



5 **Select the created camera data and click [EDIT].**



6 **In [Camera], select the camera to be used.**



7 To adjust the focus of the lens, place something that has high-contrast characters within the field of view of the camera on the conveyor surface, then click [LIVE].

Confirm that the item that was placed has been snapped within the frame on the left side of the screen.

If the screen is dark, adjust the following.

- Open the diaphragm of the lens.
- Adjust the lighting.

8 Check the field of view of the camera.

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

9 Once the camera position is determined, click the [Adjust] button in [Aperture and focus] to open the focus adjustment guide and adjust the aperture and focus according to the instructions in the guide.



10 Click [STOP].

Live will stop.

11 Set as follows.

[Robot-Mounted Cam.]: Robot
[Calibration]: Grid pattern calibration

12 Click [Calibration Setup] in the navigation area.

The grid pattern calibration setup screen will appear.

13 Select the grid spacing of the calibration grid in the [Grid Spacing] drop-down box.

14 Select [Fixed] in the [Robot-Held Cal. Grid] drop-down box.

15 Select [Auto Measurement] in the [Setting Cal. Grid Pos. Method] drop-down box.

MEMO

In Simple Mode, the world frame is set as the application frame. If you want to change the application frame to another user frame, switch to Advanced Mode and change the application frame.

16 Select [Cal. Grid Location] in the navigation area.

The screen to specify the calibration grid installation position will appear.

17 Jog the robot to the auto measurement start position.

Set such a distance that the camera center roughly faces the calibration grid center and that the calibration grid center comes into focus as the measurement start position.

18 Press the [RECORD] button in [Measurement Start Position].

[Set] is displayed in [Measurement Start Position].

19 Press the [Execute] button in [Measure Cal. Grid Position] while pressing the [SHIFT] key on the teach pendant.

When measurement is completed normally, the normal completion message pops up.

20 Check position information displayed in [Measured Cal. Grid Position] for problems.

21 Select [Calibration] in the navigation area.

The screen for setting up grid pattern calibration will appear.

22 Select [2] in the [Number of Planes] drop-down box.

23 Detect Calibration plane 1.

Move the calibration grid away from the camera by an appropriate distance, click [SNAP], and then click the [FIND] button in [1st Plane].

24 Detect Calibration plane 2.

Change the distance between the calibration grid and the camera, click [SNAP], and then click the [FIND] button in [2nd Plane].

25 Click [Calibration Points] in the navigation area.

The Calibration Points screen will appear.

26 Confirm that no points are detected other than the dot positions on the calibration grid on the displayed screen.

If improper points are detected, delete them with the following procedure.

1. Select the improper detected points from the list of detected points or enter the numbers of the detected points in the [Target] text box.
2. Click the [Delete] button to delete the improper detected points.

27 Click [Calibration Result] in the navigation area.

The calibration result screen will appear.

28 On the displayed screen, confirm that the focal length is roughly correct and that the mean error value and the maximum error value are not too large.

[Scale] indicates how many millimeters each pixel on an image represents. This value can be calculated by dividing the size of the field of view by the image size.

For example:

When the size of the field of view is 262 x 169 mm and the image size is 640 x 480 pixels, the scale is calculated as follows:

0.409 mm/pixel = 262 mm/640 pixels

If the error in [Scale] is large, check that the calibration jig installation information is properly set or that a proper value is entered in [Grid Spacing]. Note that [Scale] is not even within the field of view because it changes depending on the distance from the camera. The value displayed in [Scale] is the average value around the calibration plane.

If calibration points and calibration results are normal, calibration ends.

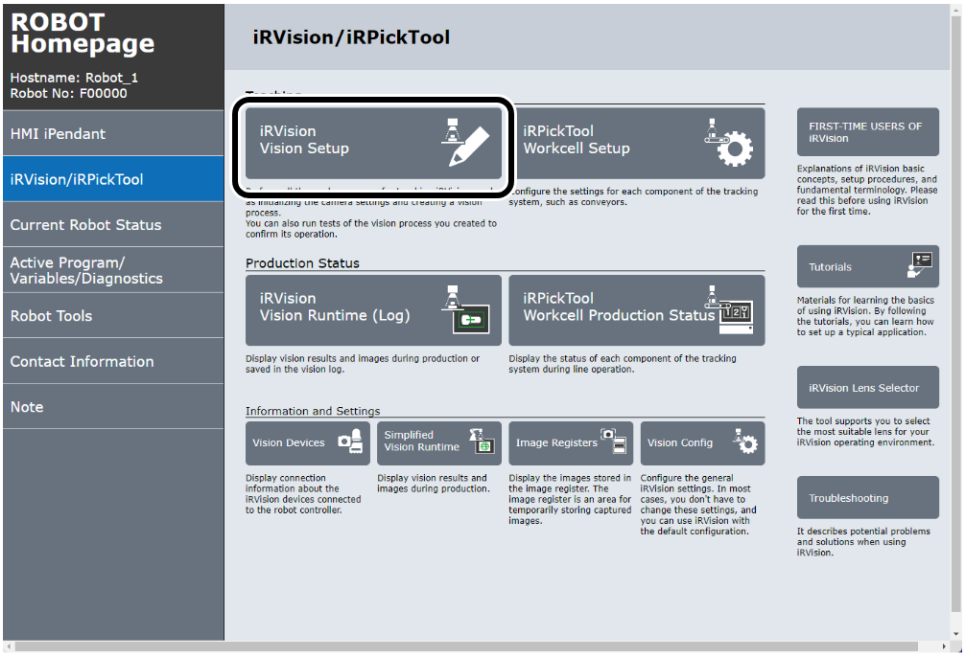
29 Click [SAVE] to save data and click [END EDIT].

Note that the calibration grid can be removed.

8.13.2.7. Creating a vision process

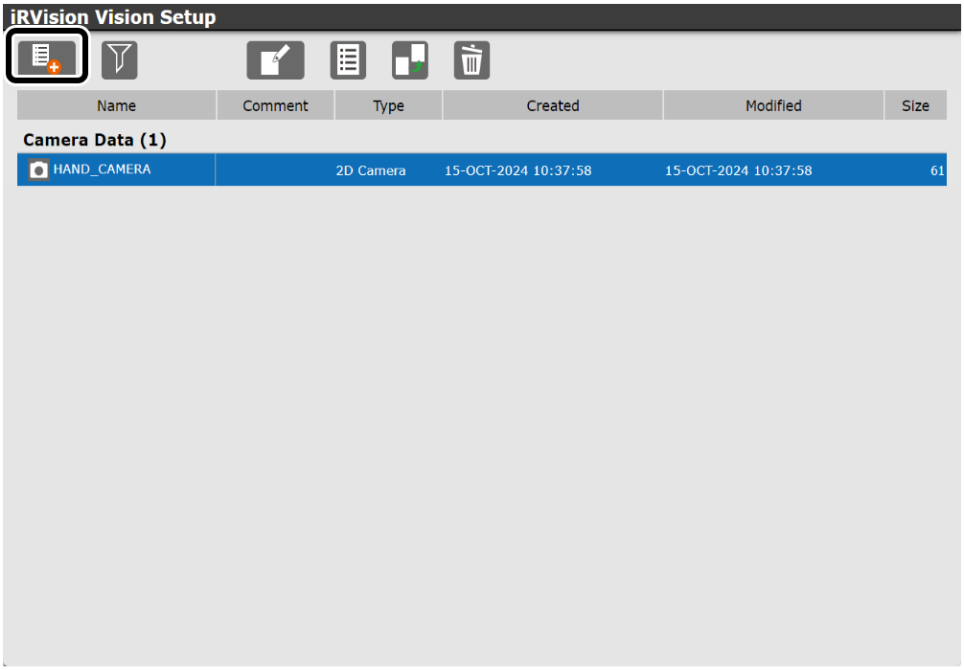
1 Open the robot homepage.

2 Select [iRVision/iRPickTool] in the menu area and click [iRVision Vision Setup] in the content display area.



The [iRVision Vision Setup] screen appears.

3 Click [CREATE].



4 Create a [2-D Single-View Vision Process] vision process.

Enter the vision process name using half-width alphanumeric characters and half-width katakana. Spaces and symbols other than underscores cannot be used. The first character must be a half-width alphanumeric character.

Example: VP2S



NOTE

Select [2-D Single-View Vision Process], not [2-D Single-View Visual Tracking].

5 On the vision data list screen, select a created vision process, and click [Edit].

6 Press the tree tab, select [2-D Single-View Vision Process], and set the following items.

- In [Camera], select the camera created in "I.8.13.2.6. Setting up a robot-mounted camera(P.665)".
- Select [Fixed Frame Offset] in [Offset Mode].
- Select [Use Frame] in [Offset Frame] and select user frame 9.

MEMO

In the pick program in [Offset Frame], specify the user frame specified in Argument 2 of PKSETNOMFRM.PC.

7 Select [Snap Tool 1] in Program Line.

8 Enter an [Exposure Time].

Set the exposure time so that an image of a part that moves during exposure will have as little blurring as possible.

For example, taking into account image blurring, in order to suppress misalignment during finding by vision to less than 0.1 mm when the conveyor speed is 100 mm/s, set a value that is less than 1 ms as the exposure time.

0.1 (mm) ÷ 100 (mm/s) = 1 (ms)

For information on the calculation of appropriate exposure times to suit the conveyor speed, refer to "Exposure time(P.22)".

9 Set a part in the field of view of the camera.

10 Select [GPM Locator Tool 1] in Program Line and perform teaching to detect parts.

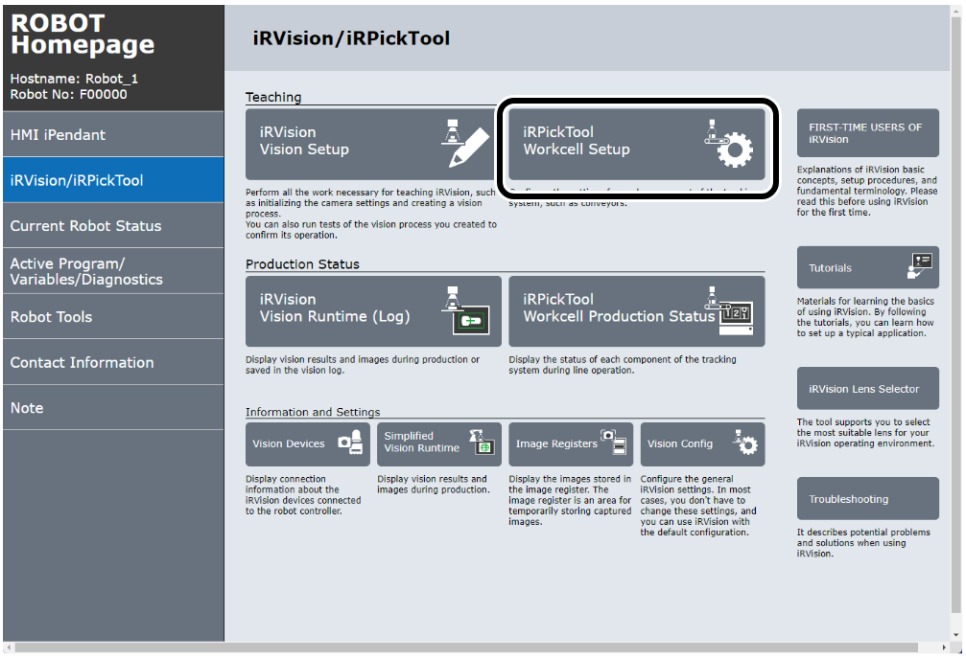
For details of the setup, refer to the section "GPM Locator Tool" in the chapter "Command tools" in the "iRVision OPERATOR'S MANUAL (Reference)"(B-84434EN).

11 Click [SAVE] to save data and click [END EDIT].

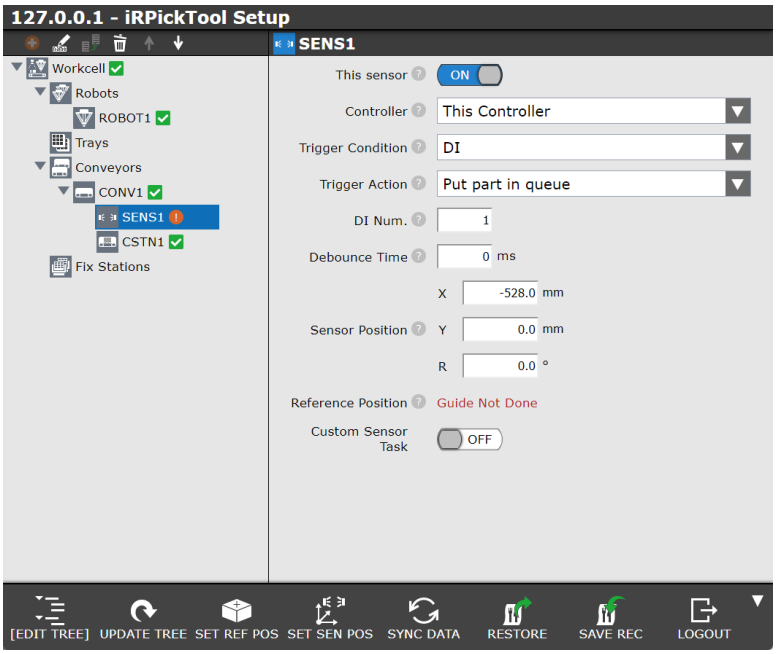
8.13.2.8. Setting up a reference position for the infeed conveyor, and teaching a robot position

1 Open the robot homepage.

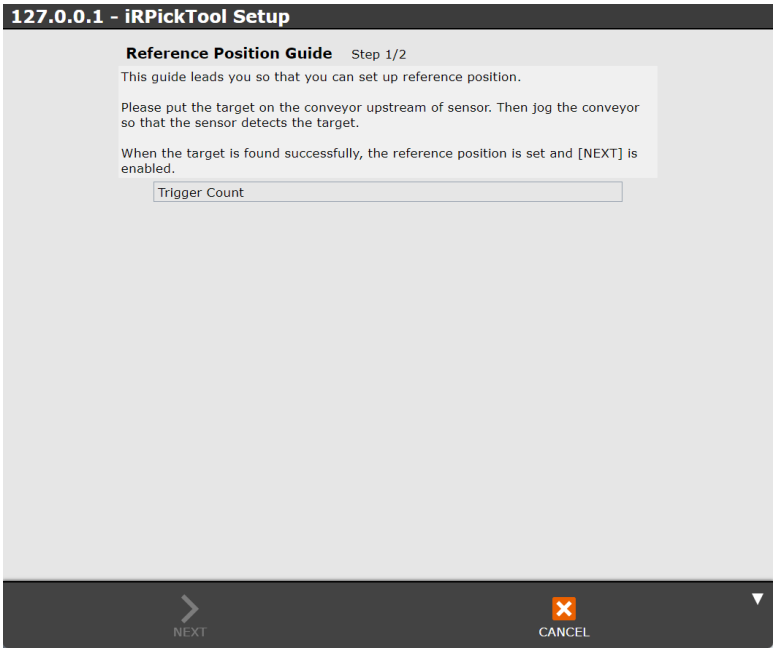
2 Select [iRVision/iRPickTool] in the menu area and click [iRPickTool System Setup] in the content display area.



3 Select [SENS1] for the sensor and press [Reference Position].



After a while, the following screen appears.

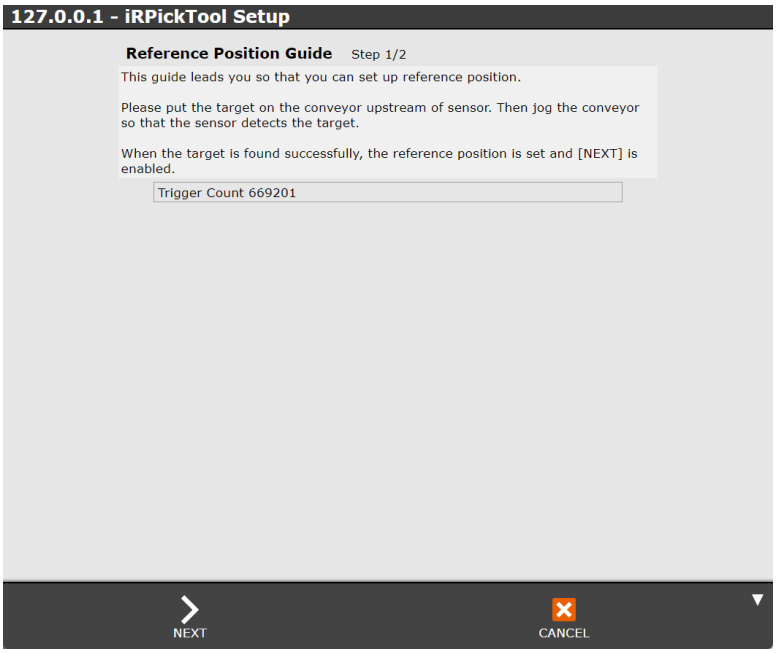


4 **Move the conveyor, and then when the part enters the robot's moving range, stop it.**

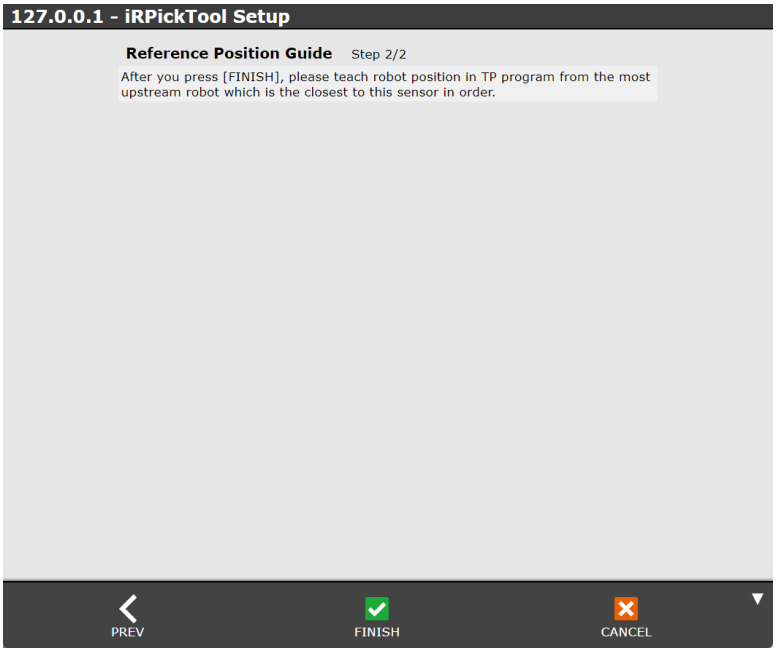
If detection is successful, [Trigger Count] will appear.

5 **Press [NEXT].**

If detection succeeds, [NEXT] will become active.

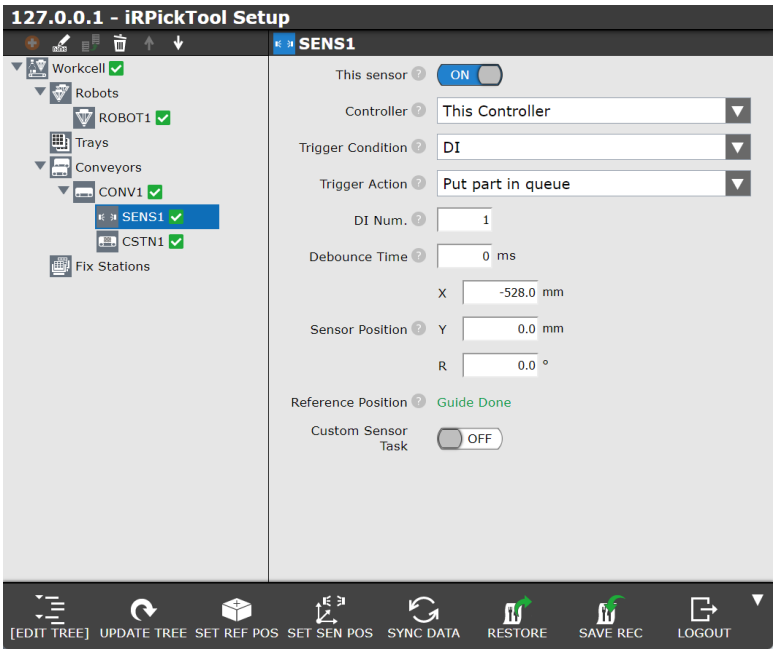


6 **Press [FINISH].**



After a while, the following screen appears.

Check that [Guide Done] appears in [Reference Position].



7 **Move the conveyor, and then when the part enters the robot's moving range, stop it.**

8 **Jog the robot to the part snap position.**

9 **Display the pick program (PT_PICKCS), and move the cursor to the line where the snap position is taught.**

When you use the sample program, note that the snap position is taught on line 19.

10 **While holding down the [SHIFT] key, press F5 [TOUCHUP].**

If F5 [TOUCHUP] is not displayed, press [> (NEXT)] to switch screens.

11 **Call PKSETNOMFRM (Trk Sch Num=1, Ofs Frm Num=9).**

When you use the sample program, note that this KAREL program is executed on line 22 in the pick program (PT_PICKCS). Therefore, place the cursor over this line and execute it in step mode.

CAUTION

Be sure to execute this KAREL program to set the values of the user frame used as the offset frame before teaching the reference position of the vision process.
Do not move the conveyor after executing this KAREL program until you teach the reference position for the vision process.

MEMO

- Use the user frame number specified in the offset frame of the vision process in Argument 2.
- When this KAREL program is executed, the nominal frame overwrites this user frame.

13 Open the vision process created in "I.8.13.2.7. Creating a vision process(P.668)".

14 Select [Offset Data Calculation] in the Program Line.

15 Enter the part thickness in the [Part Z Height] text box.

Enter the distance from the plane at which the tracking frame was taught (the calibration grid surface) to the plane where the features that are to be found by vision will be.

16 Click [SNAP] to snap images and click [FIND] to detect parts.

17 Click the [Set] button in [Reference Position].

18 Confirm that [Reference Position] is [Set].

19 Click [SAVE] and then click [END EDIT].

20 Jog the robot to the pick position.

21 Place the cursor over the line where you teach the pick position.

When you use the sample program, note that the pick position is taught on line 31.

22 While holding down the [SHIFT] key, press F5 [TOUCHUP].

If F5 [TOUCHUP] is not displayed, press [> (NEXT)] to switch screens.

8.13.3. KAREL program

This section describes the specifications of the KAREL programs that are provided.

8.13.3.1. PKSETNOMFRM.PC

Overwrite the user frame with the nominal frame for line tracking.

Argument 1:	Specify the tracking schedule number.
Argument 2:	Specify a user frame number. It is used for the Vision Process offset frame.
Argument 3: (optional)	Specify a group number. Group 1 is used if this specification is omitted.

NOTE

When the group number specified with argument 3 does not match the group number for tracking schedule specified with argument 1, IRPK-118 WARN KAREL program error 2 occurs.

Example:

Overview of the Vision Process offset frame (user frame number 1 in group 1) with the nominal frame for tracking schedule number 1.

```
Call PKSETNOMFRM(Trk Sch Num=1,Ofs Frm Num=1)
```

8.13.3.2. PKCALOFSPOS.PC

Outputs the specified position multiplied by the Vision Program offset.

Argument 1: Specify the number of the Position Register before compensation.
Argument 2: Specify the number of the Vision Register.
Argument 3: Specify the number of the Position Register that is used to output the position after compensation.
Argument 4: (optional) Specify a group number. Group 1 is used if this specification is omitted.

NOTE

An alarm occurs in the following cases:

- When the position register specified with argument 1 is not initialized
- When the format of the position register specified with argument 1 is not XYZWPR or XYZWPREXT (IRPK-118 WARN KAREL program error 1)
- When the Vision Register offset specified with argument 2 is not initialized
- When the Vision Register offset mode specified with argument 2 is not Offset Instruction
- When the Vision Register offset frame number specified with argument 2 does not match the number of the second argument of PKSETNOMFRM.PC that was executed just before

Example:

Multiply the position of PR[1] by the vision offset stored in VR[1] and output the result to PR[21]. All the registers belong to group number 1.

```
Call PKCALOFSPOS(Src Pos Reg Num=1,Vis Reg Num=1,Dst Pos Reg Num=21)
```

8.13.3.3. PKCALOFSPOS2.PC

Output the position determined by applying the offset for the Position Register to the specified position.

Argument 1: Specify the number of the Position Register before compensation.
Argument 2: Specify the number of the Position Register that contains the offset.
Argument 3: Specify the number of the Position Register that is used to output the position after compensation.
Argument 4: (optional) Specify a group number. Group 1 is used if this specification is omitted.

NOTE

An alarm occurs in the following cases:

- If the Position Register specified in Argument 1 or 2 has not been initialized yet
- If the format of the Position Register specified in Argument 1 or 2 is neither XYZWPR nor XYZWPREXT (IRPK-118 WARN KAREL program error 1)

Example:

Apply the vision offset stored in PR[2] to the position in PR[1] and output the output result to PR[21]. All the registers belong to group number 1.

```
CALL   PKCALOFSPOS2(Src Pos Reg Num=1,Ofst Pos Reg Num=2,Dst Pos Reg Num=21)
```

8.14. Filtering for signals

This function can remove chatter or noise from signals.

Although the teaching procedure for iRPickTool does not change when using this function, the execution timing of actions may be delayed. For the execution timing of actions, refer to the description of each function.

NOTE

[Specified By], [Duration], and [Polarity Inversion] are supported in 7DH1/15 (V10.10P/15) and later versions. In versions prior to 7DH1/15 (V10.10P/15), the behavior is the same as when [Specified By] is set to [Time], and both [Duration] and [Polarity Inversion] are set to [Disable].

8.14.1. Specified By

Select the unit for specifying Debounce and Duration.

[Distance] can be selected only when the [Tracking Type] is “Line” and the Trigger Condition is [DI] or [RI].

[Time]

Judgment is made based on the elapsed time. The units are [ms].

[Distance]

Judgment is made based on the distance the conveyor moves. The units are [ms].

8.14.2. Debounce

When a signal input is detected, if the next signal input is detected before the time or distance specified here has elapsed, that signal will be ignored as a mistaken input. By specifying a proper value, you can prevent the load on the CPU from increasing due to multiple detections caused by the debounce input of an external sensor.

When this function is used alone, the action is triggered upon detection of the signal input.

8.14.3. Duration

The Duration function works in conjunction with the Debounce function to eliminate chatter and accidental noise in signals. In a system where accidental noise does not occur, this function does not need to be used.

This function is available only when the [Tracking Type] is set to [Line], the Trigger Condition is [DI] or [RI], and the action is set to [Put part in queue].

NOTE

When using the Duration function, the action will not be executed until the time or distance corresponding to the combined values of Debounce and Duration has elapsed. Even when the Duration value is set to 0, the action will not be executed until the time or distance specified for Debounce has elapsed.
If the encoder type is Virtual, Duration cannot be used.

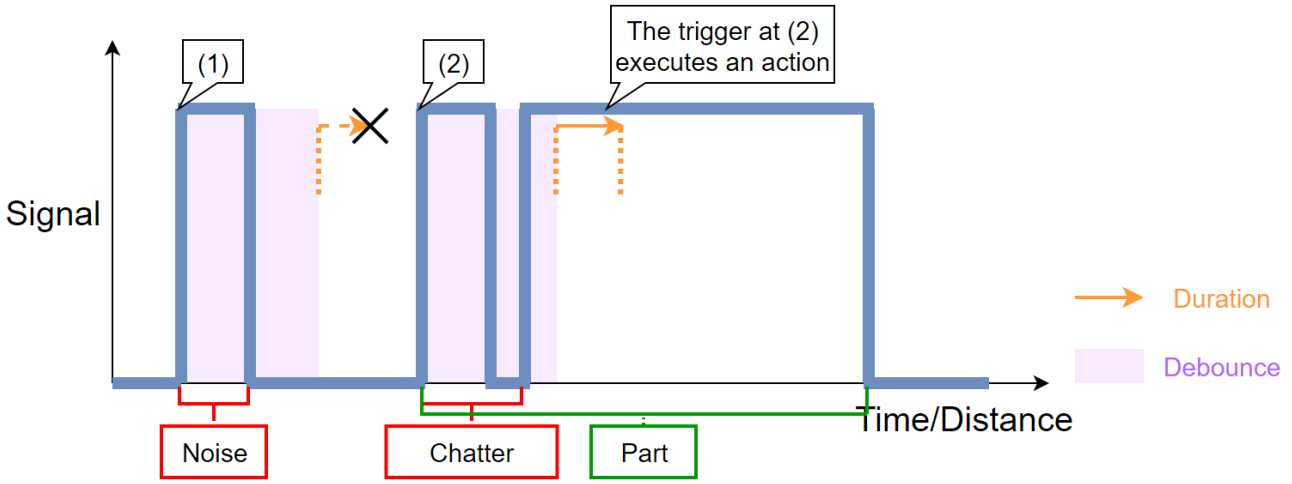
The Duration function executes an action only when the signal continues to be input for the duration or distance specified in [Duration] after the time or distance specified in [Debounce] has passed since the rising edge of signal was first detected. Only if the signal remains stable during this period, it is considered valid.

If the signal is not detected during the “Duration” period, the signal is rejected. Also, as soon as the signal is rejected, the system begins waiting for the rising edge of the next signal.

The above is explained using the figure below as an example.

(1) is an example of noise. After Debounce ends, the signal is rejected during the Duration period as the signal turns OFF.

(2) is an example of chatter that occurs when an actual part is detected. Although the signal turns OFF midway, it is ignored because it is during Debounce. During Duration, the signal remains ON, so it is judged to be a valid signal. The action is executed with the rise of (2) as the trigger at the end of Duration.



To use Duration, set each item as follows:

[Debounce]

Specify a value greater than the time or distance at which chatter occurs.

[Duration]

Specify a value greater than the time or distance at which noise occurs.

Also, set the combined value of [Debounce] and [Duration] to be shorter than the time or distance at which the part is detected.

8.14.4. Polarity Inversion

Polarity Inversion is a function that uses the moment when the rear of the part passes the sensor as the trigger, which is the opposite of the normal setting. Use this function when the front and back sides of the part have different shapes, and the back side generates less signal noise.

This function is available only when the [Tracking Type] is set to [Line], the Trigger Condition is [DI] or [RI], the action is set to [Put part in queue], and the Duration function is enabled.

NOTE

When using the Polarity Inversion function, the action is executed when the rear of the part passes the sensor. Therefore, even if the part enters the robot's boundary, it will not be allocated while it is passing the sensor. It is necessary to adjust the position of the sensor considering this.

NOTE

When using Polarity Inversion, the trigger count is acquired when the rear of the part passes the sensor, so the signal must transition from OFF to ON at that moment. Therefore, it is necessary to either use a sensor that turns OFF when detecting a part or use the I/O attribute setting to reverse the polarity.

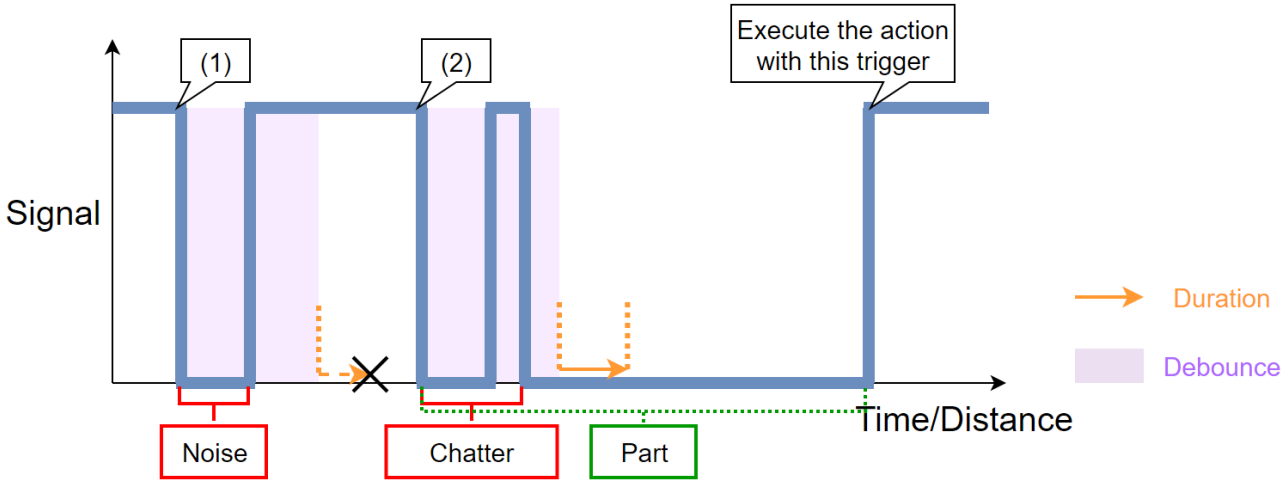
The signal is judged to be valid only when the signal continues to be OFF for the duration or distance specified in [Duration] after the time or distance specified in [Debounce] has passed since the falling edge of signal was detected. If the signal is judged to be valid, the action will be executed using the signal's rising edge at the moment when the rear section of the part passes the sensor as the trigger.

If the signal turns ON during the "Duration" period, the signal is rejected. Also, as soon as the signal is rejected, the system begins waiting for the falling edge of the next signal.

The above is explained using the figure below as an example. This figure is an example of using a sensor that turns the signal OFF when a part is detected.

(1) is an example of noise. After Debounce ends, the signal is rejected during the Duration period as the signal turns ON.

(2) is an example of chatter that occurs when an actual part is detected. Although the signal turns ON midway, it is ignored because it is during Debounce. During Duration, the signal remains OFF, so it is judged to be a valid signal. After that, the action will be executed using the signal's rising edge at the moment when the rear section of the part passes the sensor as the trigger.



8.15. Using a conveyor as a buffer

8.15.1. Functional overview

This function allows a conveyor that is not an infeed or outfeed conveyor to operate as a buffer conveyor.

If workpieces are irregularly supplied from the infeed conveyor, then when too many are supplied, the parts are temporarily placed and stored on the buffer conveyor.

When the supply of workpieces runs short, the robot picks up parts from the buffer conveyor and places them in the conveyor station of the outfeed conveyor.

For details on the buffer function, see "I.6.13. Plug & Play buffer function(P.497)".

As long as the buffer conveyor has vacant cells, robots can place workpieces on it. Conversely, if the buffer conveyor has workpieces, robots can pick workpieces from it.

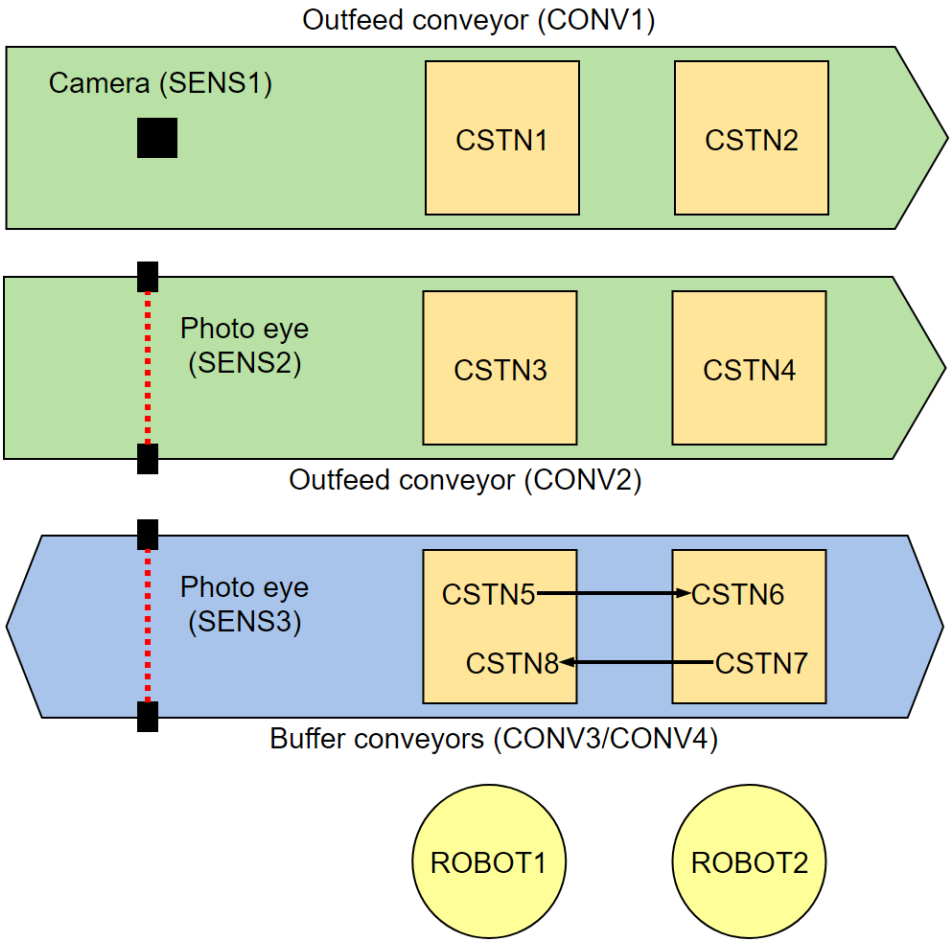
Other features include:

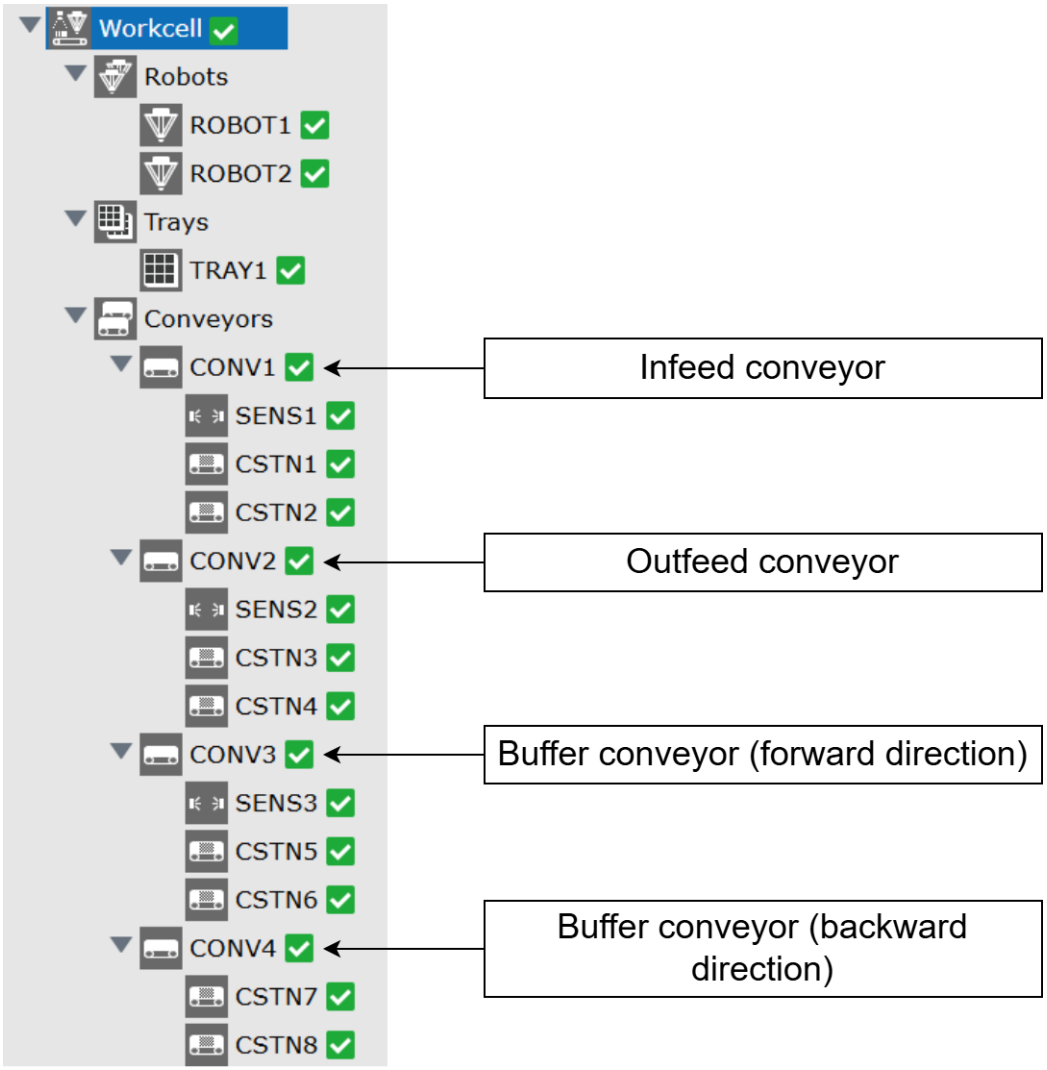
- A robot other than the one that placed a workpiece on the buffer conveyor can pick that workpiece.
  In this case, workpiece detection information can be stored and shared among robots.
- The buffer conveyor can move workpieces in the forward or backward direction so that they are positioned in front of the workpiece picking robot, improving buffer efficiency.

MEMO

The remainder of this section describes this function using a two-robot configuration as an example.

System configuration overview


Outfeed conveyor (CONV1)
Camera (SENS1)
CSTN1
CSTN2
Photo eye
(SENS2)
CSTN3
CSTN4
Outfeed conveyor (CONV2)
Photo eye
(SENS3)
CSTN5
CSTN6
CSTN8
CSTN7
Buffer conveyors (CONV3/CONV4)
ROBOT1
ROBOT2



Infeed conveyor

Install the vision sensor (SENS1) upstream.

Configure the conveyor stations (CSTN1, CSTN2) under the conveyor object (CONV1).

Outfeed conveyor

Install the photo eye (SENS2) upstream.

Configure the conveyor stations (CSTN3, CSTN4) under the conveyor object (CONV2).

Buffer conveyor

Install the photo eye (SENS3) upstream in the forward direction.

Configure separate tracking schedules for forward and backward conveyor movements.

To achieve this, configure conveyor objects used for forward conveyor movement (CONV3) and backward conveyor movement (CONV4).

Configure the conveyor stations (CSTN5, CSTN6) under CONV3.

Configure the conveyor stations (CSTN7, CSTN8) under CONV4.

Restrictions

- The buffer conveyor is supported in 7DH1 series 22 (V10.10P/22) and later. Earlier versions do not support this function.

  The buffer conveyor signal control function is supported in 7DH1 series 23 (V10.10P/23) and later.

- This function can only be used on a linear conveyor. You cannot use this function in a circular conveyor or servo conveyor.
- The supported combinations of trigger conditions and actions are [Distance], [DI], or [RI] combined with [Put part in queue].
- This function can only be used when all robots on the buffer conveyor are online.
- Unlike fixed-station buffers, the [Buffer Algorithm] function is not supported.
- The initial [Tray] status cannot be set to [Full].
- The Plug & Play function is not supported.

Sensor restrictions

- The sensor trigger condition [HDI] cannot be used.
- The sensor actions [Find part by vision], [Find part by multiview], and [Run program] cannot be used.

Conveyor restrictions

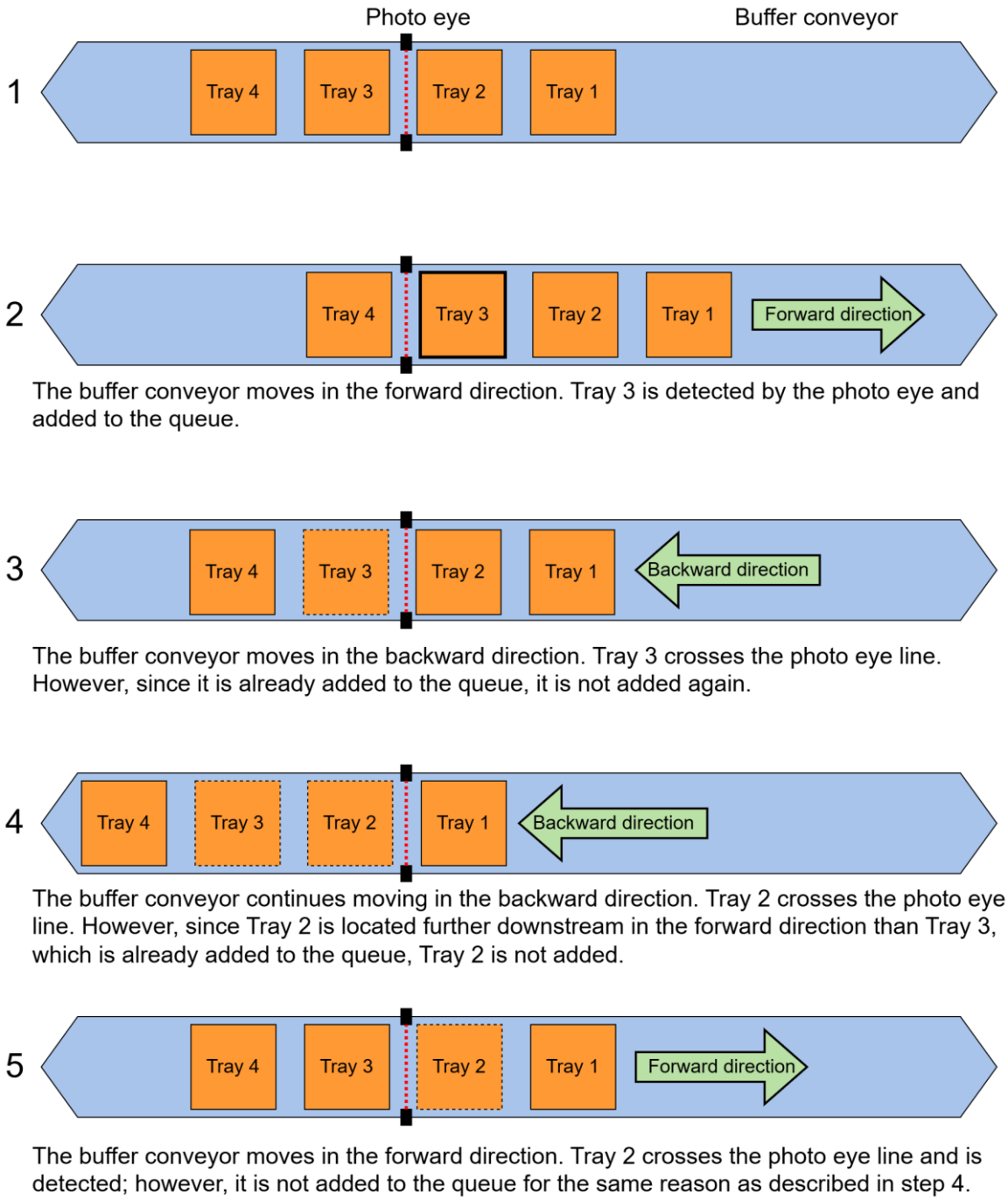
- The [Duplicate Check Mode] conveyor mode cannot be used.

  [Model ID] and [Measurement] can only be set to [Not used]. [Leave First/Last] can only be set to [Leave first].
- The [Tray Redetection], [Load Balance], and [Pre-grouping] conveyor settings cannot be modified. The [Load Balance] functions as it is set to "No".

Conveyor station restrictions

- The [Circular Boundary] (configured on selected robots) and [Skew Boundary] conveyor station settings cannot be used.
- The [Sort (Y, measurement value 1)] conveyor station setting cannot be used.
- The [Stop/start Conveyor] conveyor station setting cannot be used.

Information on a tray detected in the forward direction downstream from the existing queue element is not pushed into the queue again.



The buffer conveyor moves in the forward direction. Tray 3 is detected by the photo eye and added to the queue.

The buffer conveyor moves in the backward direction. Tray 3 crosses the photo eye line. However, since it is already added to the queue, it is not added again.

The buffer conveyor continues moving in the backward direction. Tray 2 crosses the photo eye line. However, since Tray 2 is located further downstream in the forward direction than Tray 3, which is already added to the queue, Tray 2 is not added.

The buffer conveyor moves in the forward direction. Tray 2 crosses the photo eye line and is detected; however, it is not added to the queue for the same reason as described in step 4.

1, 2: The buffer conveyor moves in the forward direction. Tray 3 is detected by the photo eye and added to the queue.

3: The buffer conveyor moves in the backward direction. Tray 3 is detected by the photo eye; however, since it is already added to the queue, it is not added again.

4: The buffer conveyor continues moving in the backward direction. Tray 2 is detected by the photo eye. However, since it is located further downstream in the forward direction than Tray 3, which is already added in the queue, it is not added.

5: The buffer conveyor moves in the forward direction again. Tray 2 is detected by the photo eye; however, it is not added to the queue for the same reason described in step 4.

8.15.2. Startup procedure

This section describes the setup procedure for a system that uses a buffer conveyor.

8.15.2.1. Sensor installation and connection

Install a vision sensor upstream of the infeed conveyor.

Refer to "I.2.2.2. Checking the connection of the camera(P.32)" to verify the camera connection.

Install photo eyes upstream of the outfeed conveyor and upstream of the buffer conveyor in the forward direction.

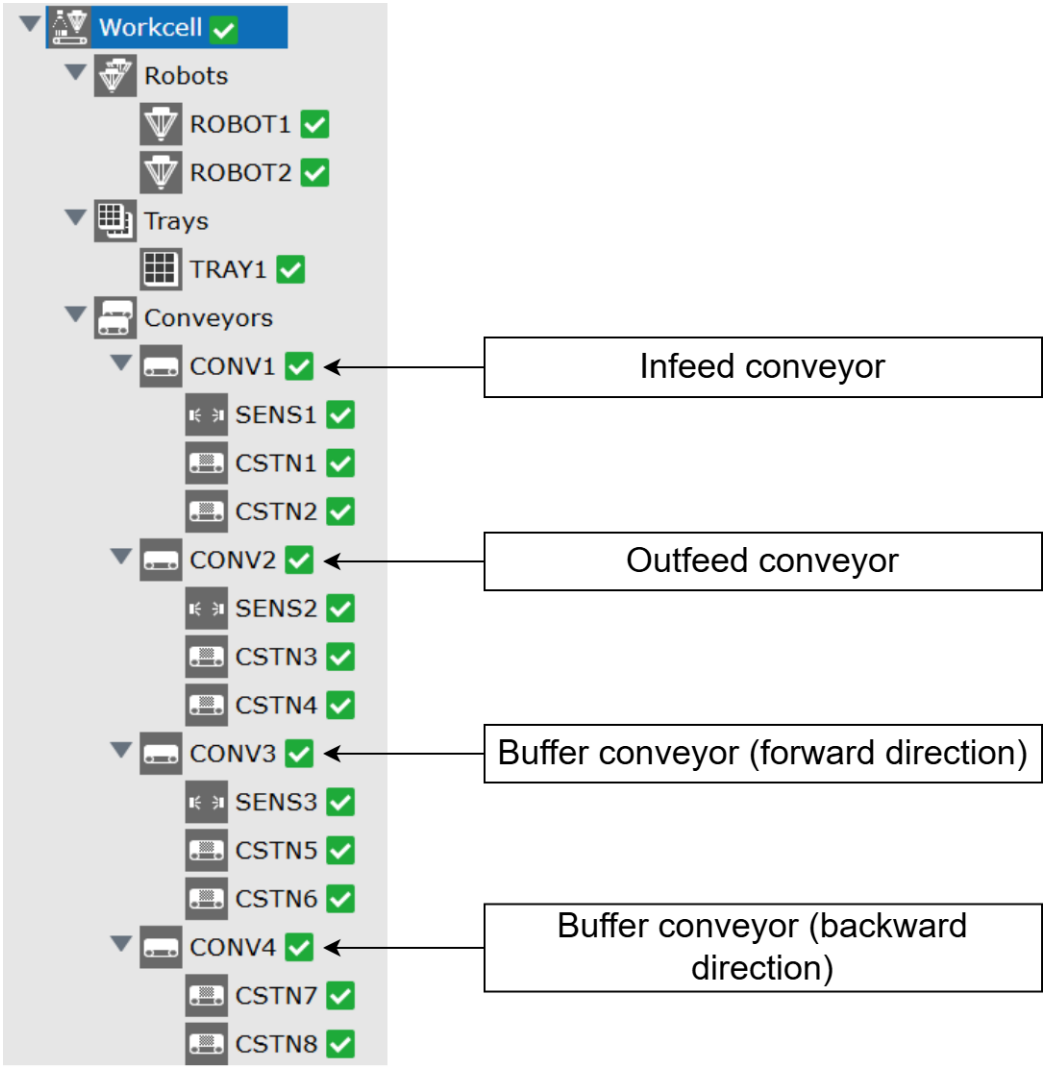
Refer to "I.2.2.3. Checking the input of the trigger sensor(P.36)" to verify the sensor connections.

8.15.2.2. Setting up applications

MEMO

Buffer conveyors move in both the forward and backward directions. In the description below, "upstream" and "downstream" are used relative to the forward direction unless otherwise specified.

Reference the configuration described in "Basic operations" of "I.5. Basic functions reference(P.258)" to configure the workcell in the system as shown below.



Setting up a robot

Add [ROBOT1] and [ROBOT2] as child objects under [Robots].

The procedures below are for a two-robot system. When using three or more robots, create additional [ROBOT] child objects under [Robot] to match the number of actual robots.

MEMO

Buffer conveyors do not support Circular Boundary. In the robot settings, set [Circular Boundary] to [Disable].

Setting up a conveyor

Set up the infeed, outfeed, and buffer conveyors.

In the procedures below, the infeed conveyor is referred to as [CONV1], the outfeed conveyor as [CONV2], the forward buffer conveyor as [CONV3], and the backward buffer conveyor as [CONV4].

Infeed conveyor

1. **Select [CONV1]. The conveyor setup screen will appear.**
2. **Leave [Buffer] set to [Not Used].**
3. **Configure other settings according to the actual conveyor setup by referring to the description in "I.2.3.3. Setting up a conveyor(P.71)".**

Outfeed conveyor

1. **Select [CONV2]. The conveyor setup screen will appear.**
2. **Leave [Buffer] set to [Not Used].**
3. **Configure other settings according to the actual conveyor setup by referring to the description in "I.2.3.3. Setting up a conveyor(P.71)".**

Setting up a tray

Configure the tray by referring to the description in "I.2.3.2. Setting up a tray(P.67)".

The trays used on the buffer conveyor may be identical to those used on the outfeed conveyor. However, it is recommended to set up separate trays, even if the cell layout is identical, to prevent the reference cell settings from being overwritten during configuration.

Setting up a buffer conveyor

Create [CONV3] and [CONV4] as child objects under [Conveyors], and configure them as described below.

Setting the buffer conveyor (forward direction)

1. **Select [CONV3]. The conveyor setup screen will appear.**
2. **Select [Forward Direction] for [Buffer]. When this setting is applied, the settings of the child objects [SENS] and [CSTN] are all reset to their default values.**

   [Duplicate Check Mode], [Tray Redetection], [Load Balance], and [Pre-grouping], which are not supported for buffer conveyors, can no longer be modified. In addition, [Servo Conveyor] is fixed to [Off], and [Circular Tracking] is fixed to [Not Used].
3. **Select the tray configured in "Setting up a tray" for [Tray].**

Setting up the buffer conveyor (backward direction)

1. **A [CONV] object that has a [SENS] child object cannot be configured as a backward conveyor. First, delete [SENS4], the child object of [CONV4].**
2. **Select [CONV4]. The conveyor setup screen will appear.**
3. **Select [Backward Direction] for [Buffer]. When this setting is applied, the settings of the child object [CSTN] are all reset to their default values.**

   [Duplicate Check Mode], [Tray Redetection], [Load Balance], and [Pre-grouping], which are not supported for buffer conveyors, can no longer be modified. In addition, [Servo Conveyor] is fixed to [Off], and [Circular Tracking] is fixed to [Not Used].
4. **The [Tray] settings for the backward buffer conveyor are linked to those for the forward buffer conveyor and cannot be modified.**

Setting up a conveyor station

When using a buffer conveyor, create the same number of conveyor stations as robots.

The procedures below are for a two-robot system. Therefore, two [CSTN] child objects are created under each [CONV] object.

When using three or more robots, create additional [CSTN] child objects under [CONV] to match the number of actual robots.

Configure [CSTN5] and [CSTN6], the child objects under the [CONV3] buffer conveyor (forward direction) and [CSTN7] and [CSTN8], the child objects under the [CONV4] buffer conveyor (backward direction).

1. **Select [CSTN] in the tree view. The conveyor station setup screen will appear.**
2. **Configure the settings using the description in "I.2.3.8. Setting up a conveyor station(P.106)" as a reference.**

MEMO

The following functions are not supported for buffer conveyor stations and cannot be configured on the conveyor station setup screen.
- Circular Boundary
- Skew Boundary
- Sort (Y, measurement value 1)
- Stop/start Conveyor

If Circular Boundary is enabled for a robot selected for a conveyor station, [Circular Boundary] is labelled as unavailable on the conveyor station setup screen. On the robot setup screen, set Circular Boundary to Disable.

For a backward buffer conveyor station, the following settings are automatically configured based on the settings of the corresponding forward buffer conveyor station according to the procedure described in "Setting up the backward buffer conveyor station ".
- Encoder Number
- Selected Boundary
- Upstream
- Downstream

Setting up the forward buffer conveyor station

A buffer setting area is displayed on the forward buffer conveyor station setup screen.

Configure the following items in this area.

[Send Position(Upstream)]

This setting item controls when information about a detected tray is sent to the upstream conveyor station.

Specify the X value in the tracking frame.

When all cells of a tray are located further upstream than the [Send Position(Upstream)] value, the tray information is sent to the upstream conveyor station.

In the figure below, information about a tray located further upstream than the [Send Position(Upstream)] value (blue) for CSTN6 is sent to CSTN5, the upstream conveyor station.

If a conveyor station is the most upstream station on the buffer conveyor, there is no upstream conveyor station to send the information to, so the tray information is discarded. When the tray later moves in the forward direction and returns, the tray information cannot be retrieved.

In the figure below, since CSTN5 is the most upstream station on the buffer conveyor, information about a tray located further upstream than the [Send Position(Upstream)] value (red) is discarded.

If the send position is not configured, trays are not added to the queue.

This item can be modified during production.

[Send Position(Downstream)]

This setting item controls when information about a detected tray is sent to the downstream conveyor station.

Specify the X value in the tracking frame.

When all cells of a tray are located further downstream than the [Send Position(Downstream)] value, the tray information is sent to the downstream conveyor station.

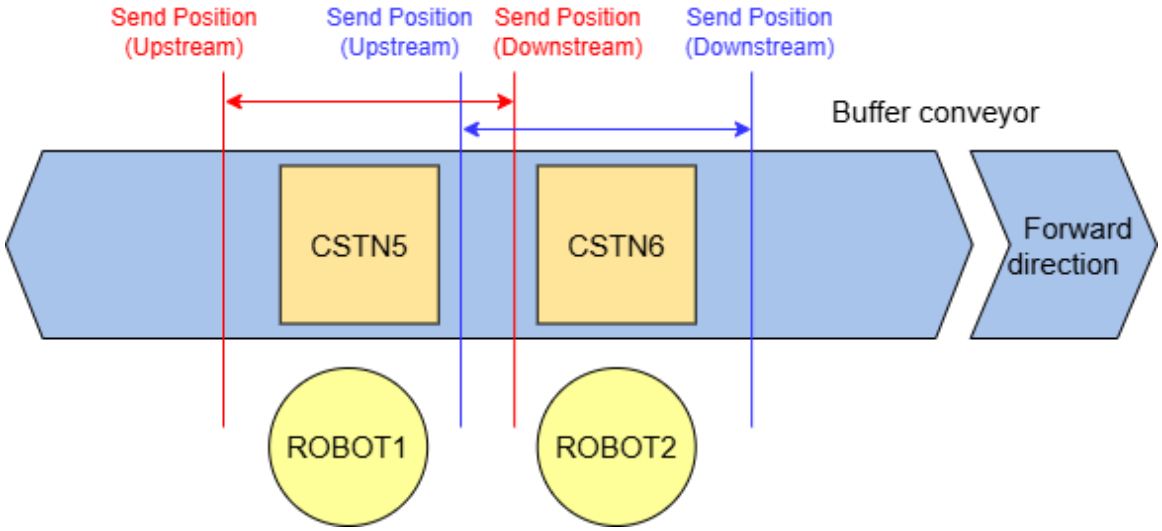
In the figure below, information about a tray located further downstream than the [Send Position(Downstream)] value (red) for CSTN5 is sent to CSTN6, the downstream conveyor station.

If a conveyor station is the most downstream station on the buffer conveyor, there is no downstream conveyor station to send the information to, so the tray information is discarded. When the tray later moves in the backward direction and returns, the tray information cannot be retrieved.

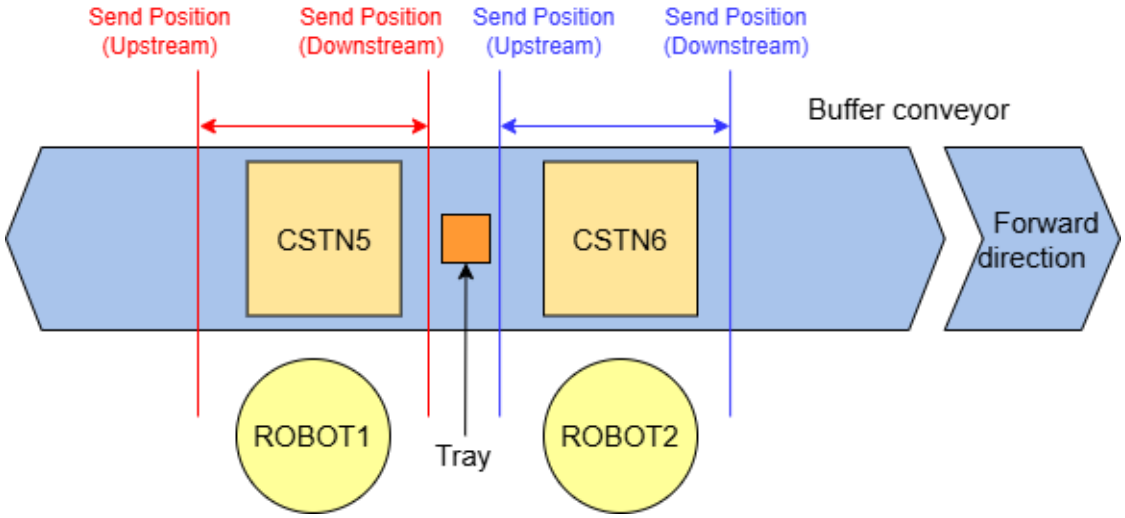
In the figure below, since CSTN6 is the most downstream station on the buffer conveyor, information about a tray located further downstream than the [Send Position(Downstream)] value (blue) is discarded.

If the send position is not configured, trays are not added to the queue.

This item can be modified during production.



Assume that the area between the send position (downstream) of conveyor station CSTN5 (red in the figure below) and the send position (upstream) of the neighboring conveyor station CSTN6 (in blue in the figure below) is at least one tray length. When a tray crosses the send position and its information is sent to the neighboring conveyor station, the tray may be immediately sent back. In this case, information is actually sent when some of the cells enter the area between the send position (upstream) (CSTN6 in blue in the figure below) and send position (downstream) (CSTN6 in blue in the figure below) of the neighboring conveyor station.



[Signal Output]

This setting configures output signals that control buffer conveyor movement. For example, these signals can stop or reverse the buffer conveyor to prevent trays or placed workpieces from passing through while the buffer conveyor continues moving.

A signal is output when a tray or workpiece crosses [Send Position(Upstream)] or [Send Position(Downstream)]. The signal output position (signal control position) is configured using an offset from the corresponding [Send Position].

Enable or disable output signals, enter DO numbers, and turn them on or off. When turning off a DO, it must be set to on when production starts.

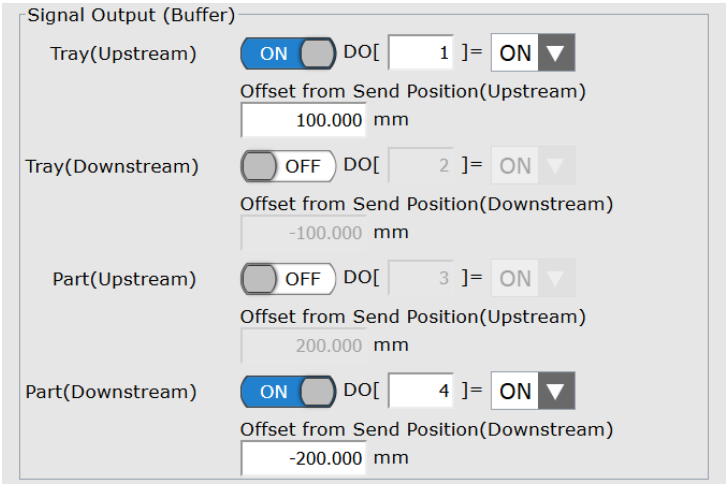
Specify unique [DO] numbers for [Tray(Upstream)], [Tray(Downstream)], [Part(Upstream)], and [Part(Downstream)].

Set [Offset from Send Position] to a positive value for upstream settings or to a negative value for downstream settings.

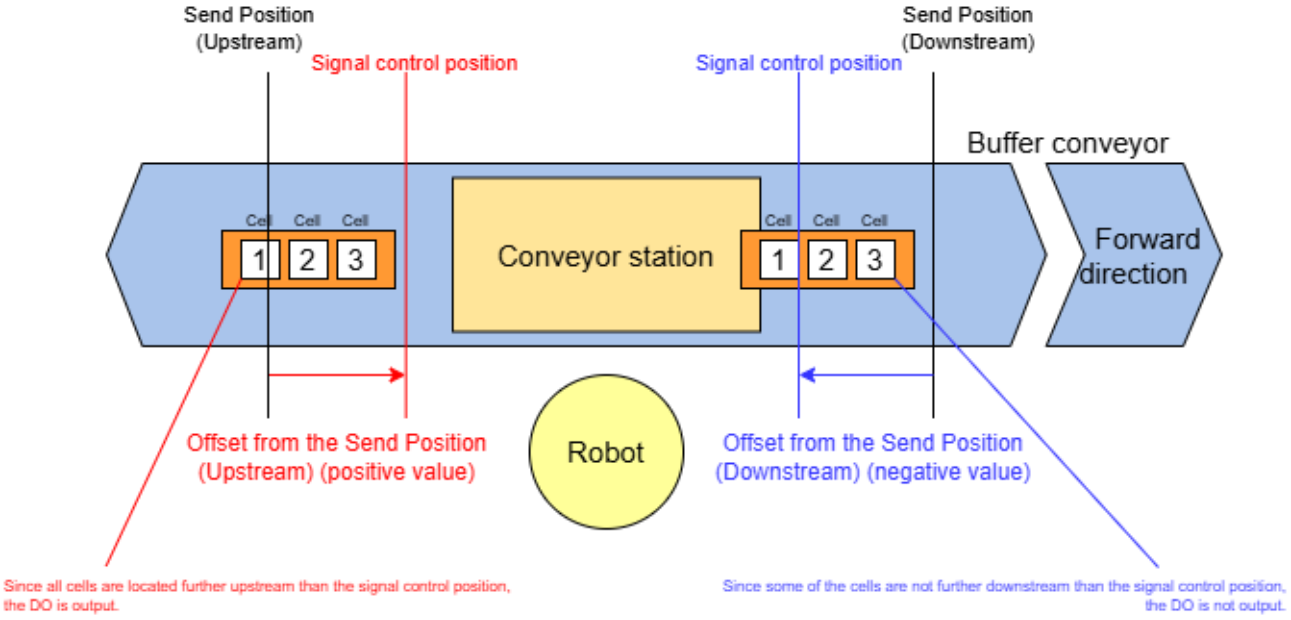
If either [DO] or [Offset from Send Position] is not configured, targets are not added to the queue.

If signal control is switched to [ON] by the toggle switch after a target is added to the queue, the target is added to the queue, but signal control will not function while either [DO] or [Offset from Send Position] remains unconfigured.

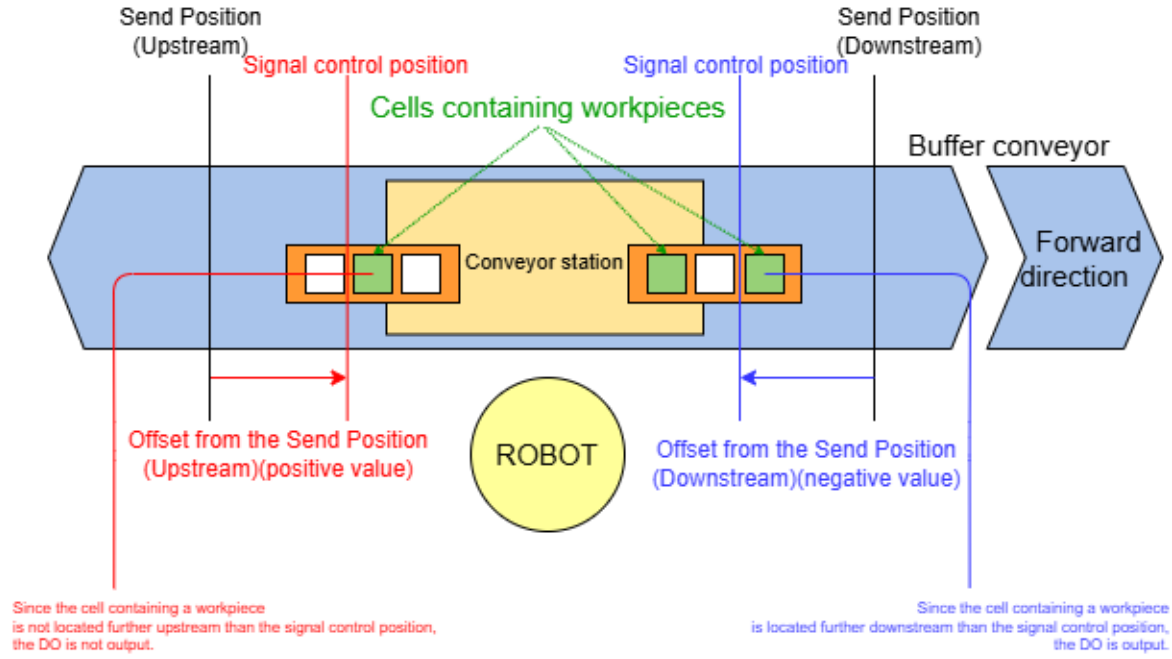
These items can be modified during production.



If Tray(Upstream) or Tray(Downstream) is ON by the toggle switch, when all cells of a tray are located closer to the send position than the signal control position, the signal is output regardless of whether the cells contain workpieces.



If Part(Upstream) or Part(Downstream) is ON by the toggle switch, when a cell containing a workpiece is located closer to the send position than the signal control position, the signal is output.



If the area between [Send Position(Upstream)] and [Send Position(Downstream)] does not overlap or overlaps only slightly with the neighboring conveyor station (Fig. No overlap with the neighboring conveyor station), the signal is output immediately after information is received from the neighboring conveyor station. In this case, the signal can be suppressed by expanding the area between [Send Position(Upstream)] and [Send Position(Downstream)] so that it sufficiently overlaps the neighboring conveyor station (Fig. Sufficient overlap with the neighboring conveyor station).

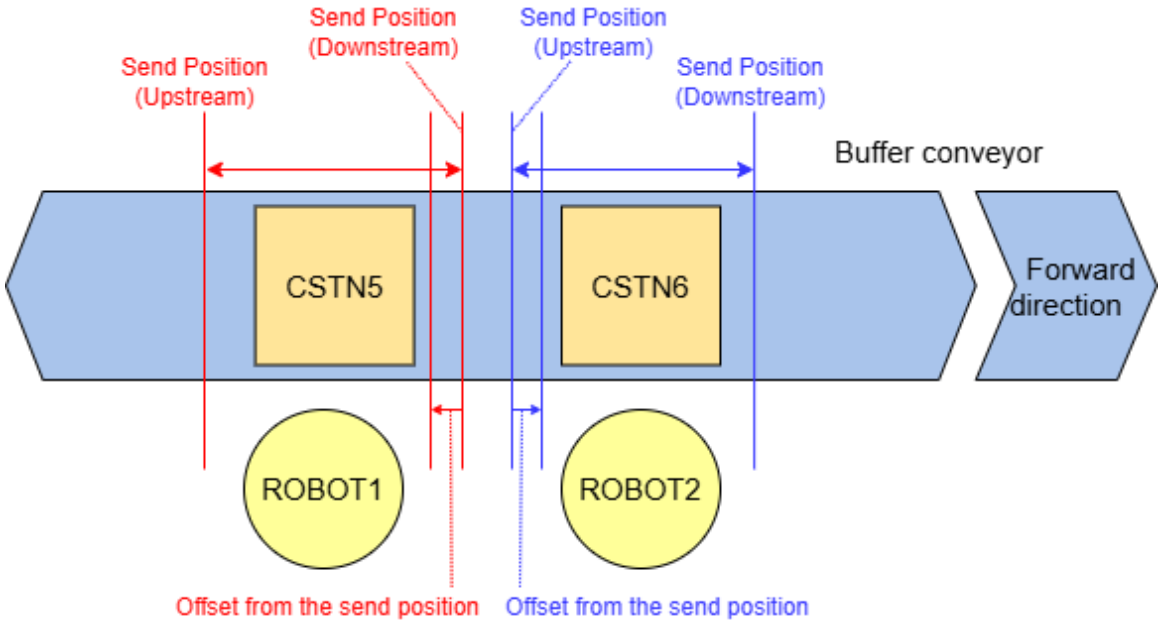


Fig. 8.15.2.2-1 No overlap with the neighboring conveyor station

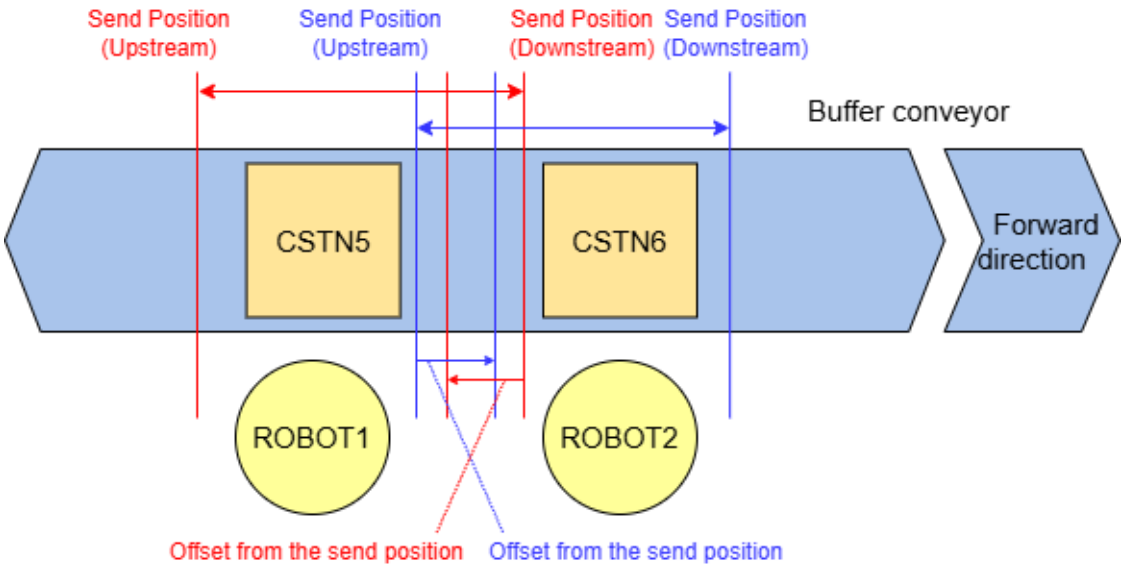


Fig. 8.15.2.2-2 Sufficient overlap with the neighboring conveyor station

Setting up the backward buffer conveyor station

This procedure sets up the backward buffer conveyor station.

Select [Robot] on the backward buffer conveyor station setup screen.

Then, reverse the order based on the [Robot] settings of the forward buffer conveyor station.

Proceed to select [Line Tracking Schedule]. Select a different [Line Tracking Schedule] number from that used by the forward conveyor station for the same robot.

If the forward buffer conveyor station has already been configured, its settings can be applied to a backward buffer conveyor station.

Click the [SET TRK FRM] function key on the buffer conveyor (backward direction) setup screen for [CONV4]. A confirmation screen appears with the message “Set line tracking schedule based on Forward Direction Conveyor ”.

Select [OK] to automatically configure the following items:

- Tracking Frame: X-axis value reversed by 180°
- Encoder Scale: Same as the forward-direction value with the opposite sign
- Backward “Upstream Boundary”: Forward “Downstream Boundary” value with the opposite sign
- Backward “Downstream Boundary”: Forward “Upstream Boundary” value with the opposite sign

Setting up a sensor

1 **Configure the [SENS1] child object under [CONV1], [SENS2] under [CONV2], and [SENS3] under [CONV3] by referring to the description in “I.5.5. Setting up a sensor(P.288)”.**

Setting up a reference position and teaching a robot position

Teach a robot position by referring to the description in “I.2.3.10. Setting up a reference position and teaching a robot position(P.113)”.

When the reference position is configured for SENS3, the triggers for the forward and backward line tracking schedules are configured.

MEMO

Of the [Trigger Condition] options, [HDI] cannot be used with buffer conveyors. Of the [Trigger Action] options, [Find part by vision], [Find part by multiview], and [Run program] cannot be used with buffer conveyors.

After configuring the reference position for SENS3, teach robot positions for the tracking programs that place workpieces on the forward and backward buffers.

The tracking program that picks workpieces from the forward-moving buffer uses the same robot position (position register) as the tracking program that places workpieces on the forward-moving buffer.

The tracking program that picks workpieces from the backward-moving buffer uses the same robot position (position register) as the tracking program that places workpieces on the backward-moving buffer.

For examples of configuring these tracking programs, refer to "I.8.15.4. Sample programs for the infeed-to-buffer-to-outfeed-conveyor configuration(P.694)".

8.15.2.3. Checking the production status

The production status of a buffer conveyor or conveyor station can be checked.

1. **Open the robot homepage.**
2. **Select [iRVision/iRPickTool] in the menu area and click [iRPickTool Workcell Production Status] in the content display area.**
3. **Select a forward conveyor or conveyor station in the tree view.**
4. **The product status screen appears.**

   For details on the production status screen, see "I.8.2. Checking the production status(P.567)".

   When a conveyor is selected, the following cell-level items appear:

   [Run Time], [Conveyor Speed (mm/s)], [Tray], [Total Pick Count], [Pick Rate (counts/min)]

   [Total Pick Count] displays the total number of placed and picked workpieces for this conveyor.

   [Pick Rate (counts/min)] displays the average [Total Pick Count] value per minute.

   When a conveyor station is selected, the following cell-level items appear:

   [Run Time], [Tray], [Total Pick Count], [Pick Rate (counts/min)]

   The queue does not appear for buffer conveyor stations.

   If a backward conveyor or conveyor station is selected, a message prompts selection of a forward conveyor or conveyor station.

8.15.3. KAREL program

This section describes which standard conveyor KAREL programs can be used with buffer conveyors.

For the specifications of standard conveyor KAREL programs that can be used with buffer conveyors, refer to the corresponding pages.

In addition, the specifications of buffer conveyor KAREL programs are described on the following pages.

Standard conveyor KAREL programs that can be used with buffer conveyors

- I.5.8.3.2. PKCVSETTRNAM.PC(P.340)
  (When using a buffer conveyor, a forward conveyor must be specified for Argument 1.)
- I.5.8.3.5. PKCVGETNAME.PC(P.343)
- I.5.8.3.7. PKCVGETCNVSPD.PC(P.345)
- I.5.8.5.1. PKCSGETID.PC(P.350)
- I.5.8.5.12. PKCSSETDSLIN.PC(P.358)
- I.5.8.5.13. PKCSGETENC.PC(P.358)
- I.5.8.5.14. PKCSGETTRID.PC(P.359)
- I.5.8.5.16. PKCSGETDSLIN.PC(P.359)

Standard conveyor KAREL programs that cannot be used with buffer conveyors

- I.5.8.3.1. PKCVSETLBD.PC(P.339)
- I.5.8.3.3. PKCVENBLB.PC(P.341)

- I.5.8.3.4. PKCVENBSC.PC(P.342)
- I.5.8.5.2. PKCSCLRQUE.PC(P.351)
- I.5.8.5.3. PKCSGETQUE.PC(P.351)
- I.5.8.5.4. PKCSGETQCELL.PC(P.352)
- I.5.8.5.5. PKCSACKQUE.PC(P.353)
- I.5.8.5.6. PKCSPUTQUE.PC(P.353)
- I.5.8.5.7. PKCSPUTQUE2.PC(P.354)
- I.5.8.5.9. PKCSGETDIST.PC(P.355)
- I.5.8.5.8. PKCSGETTIME.PC(P.354)
- I.5.8.5.10. PKCSGETPFRT.PC(P.356)
- I.5.8.5.11. PKCSGETNPRT.PC(P.357)
- I.5.8.5.17. PKCSGETPINFO.PC(P.359)
- I.5.8.5.15. PKCSCLRACK.PC(P.359)
- Plug & Play I.6.10. KAREL program(P.444)

Buffer conveyor KAREL programs

- I.8.15.3.1. PKCVCLRBUF.PC(P.691)
- I.8.15.3.2. PKCSGETBUF.PC(P.691)
- I.8.15.3.3. PKCSACKBUF.PC(P.693)
- I.8.15.3.4. PKCSGETBFDIS.PC(P.693)

8.15.3.1. PKCVCLRBUF.PC

This program deletes queue information from a conveyor buffer.

Specify a conveyor as the argument. Queue information is deleted from all robot controllers on the conveyor, regardless of whether a forward or backward conveyor is specified. It is not necessary to call this program separately for forward and backward conveyors.

When PKWCSTART is called during system startup, queue information is deleted from all buffer conveyors in the workcell. In contrast, PKCVCLRBUF deletes queue information from the specified conveyor buffer.

This KAREL program is comparable to PKFSCLRBUF, which is used when a fixed station is used as a buffer; however, PKCVCLRBUF does not require the specification of an identification number.

Argument 1: Specify the conveyor name.

Example:

Specify the conveyor name for Argument 1.

```
CALL PKCVCLRBUF('CONV3')
```

8.15.3.2. PKCSGETBUF.PC

It gets the target (empty cell information or information of the part placed in a cell) from the buffer placed on a specified conveyor 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:

In the description below, the terms “most downstream” and “discard line” assume that a forward conveyor station is specified. When a backward conveyor station is specified, interpret the terms “most downstream” and “discard line” relative to the backward direction.

- If no model ID is specified for a cell target, the target is the most downstream vacant cell with the lowest layer number (the lowest layer) of the most downstream tray at the specified conveyor station.

  If all vacant cells with the lowest layer number (the lowest layer) of the most downstream tray have crossed the discard line, the target is selected from the second most downstream tray.

- If no model ID is specified for a workpiece target, the target is the workpiece in the most downstream cell with the highest layer number (the highest layer) of the most downstream tray at the specified conveyor station.

  If all occupied cells with the highest layer number (the highest layer) of the most downstream tray have crossed the discard line, the target is selected from the second most downstream tray.

- If a model ID is specified for a cell target, the target is the most downstream vacant cell with the lowest layer number (the lowest layer) of the most downstream tray at the specified conveyor station that matches the specified model ID.

  If all vacant cells with the lowest layer number (the lowest layer) of the most downstream tray that match the specified model ID have crossed the discard line, the target is selected from the second most downstream tray.

- If a model ID is specified for a workpiece target, the target is the workpiece in the most downstream cell with the highest layer number (the highest layer) of the most downstream tray at the specified conveyor station that matches the specified model ID.

  The target is selected based on the cell’s model ID, rather than the workpiece’s model ID. If all occupied cells with the highest layer number (the highest layer) of the most downstream tray that match the specified model ID have crossed the discard line, the target is selected from the second most downstream tray.

Argument 1: Specify the ID of a conveyor station.

Argument 2: Specify the target type.
- 1: Cell
- 2: Part (placed in a cell)

Argument 3: Specify the number of the Vision Register to store the vision offset.

Argument 4: Specify the number of the Vision Register to store workpiece detection information.
This argument can be specified only when the target is a workpiece.
If the target is a cell, or if the target is a workpiece but the information is not required, specify 0.
If a non-zero value is specified for Argument 4, specify Argument 6.
Four types of information can be retrieved:
- Encoder count
- Model ID
- Found position
- Measurement value

The offset data remains uninitialized to prevent the specified vision register from being used for vision offset processing.

Argument 5: Specify the number of register storing the processing status.
- 0: Success. The information of the target within the tracking area is allocated and the trigger for the tracking motion is set.
- 1: Failure. No processable target exists between the upstream boundary and the discard line, and no value is set in the vision register.

Argument 6: Specify the number of the register that stores the retrieval status of the workpiece detection information.
Specify this argument when a non-zero value is specified for Argument 4.
If 0 is specified for argument 4, specify 0 for Argument 6 as well.
- 0: Success.
- 1: Failure. If detection information is not stored in the buffer, the NG status is stored.

Argument 7: Specify a model ID.
This argument is optional and normally does not need to be specified.

⚠CAUTION

If the target is a cell and Argument 4 or 6 is non-zero, error IRPK-057 “Specified value in argument is invalid” occurs.
If the processing status stored in the register specified by Argument 5 is 1 (NG), no values are output to the registers specified by Arguments 4 and 6.

Example:

The ID stored in R[13] is specified as the conveyor station ID, a cell is specified as the target, and 4 is specified as the vision register number for the offset. Since the target is a cell, 0 is specified for Argument 4, which specifies the vision register number for workpiece detection information. 3 is specified as the number of the register for the processing status. Since the target is a cell, 0 is specified for Argument 6, which specifies the register number for the retrieval status of workpiece detection information.

```
CALL PKCSGETBUF(CStn ID=R[13],Cell,Offset VR=4,Buffer VR=0,Stat Reg=3,Buffer Stat Reg=0)
```

8.15.3.3. PKCSACKBUF.PC

After the processing status of PKCSGETBUF becomes 0 and the target information is allocated, this program notifies the conveyor of the processing that has been performed for the target.

This KAREL program is comparable to PKFSACKBUF, which is used when a fixed station is used as a buffer.

Argument 1: Specify the ID of a conveyor station.

Argument 2: Specify the processing result.
- 1: Success
- 2: Return

Argument 3: Specify the number of the Vision Register storing the detection information. This argument can only be specified when the target is a cell. The stored detection information can be retrieved when calling PKCSGETBUF later with a workpiece specified as the target.
This argument is optional and normally does not need to be specified.

⚠CAUTION

If the vision register used by PKCSGETBUF to store workpiece detection information is specified for Argument 3, error IRPK-234 “Invalid vision register” occurs.

Example:

The ID stored in R[13] is specified as the conveyor station ID and the processing result is specified for Argument 2.

```
CALL PKCSACKBUF(CStn ID=R[13],Success)
```

8.15.3.4. PKCSGETBFDIS.PC

This program returns the distance to the upstream boundary of the tracking area. The distance is expressed in millimeters.

This KAREL program is comparable to PKCSGETDIST, which is used for standard conveyor stations.

Argument 1: Specify the ID of a conveyor station.

Argument 2: Specify the target type.
- 1: Cell
- 2: Part (placed in a cell)

Argument 3: Specify the order. Only 1 can be specified for this KAREL program.

Argument 4: This argument controls the picking order when using a hand that can hold multiple workpieces. Only 1 can be specified for this KAREL program.

Argument 5: This argument controls the picking order when using a hand that can hold multiple workpieces. Only 1 can be specified for this KAREL program.

Argument 6: Specify the number of the register to store the distance to the upstream boundary of the tracking area. In the case of farther upstream than the upstream boundary of the tracking area, the sign is positive. In the case of farther downstream than the upstream boundary of the tracking area, the sign is negative.

Argument 7: Specify the number of the Vision Register to store the vision offset.

Argument 8: Specify the number of the Vision Register to store workpiece detection information.
This argument can be specified only when the target is a workpiece.
If the target is a cell, or if the target is a workpiece but the information is not required, specify 0.

If a non-zero value is specified for Argument 8, specify Argument 10.
Four types of information can be retrieved:

- Encoder count
- Model ID
- Found position
- Measurement value

The offset data remains uninitialized to prevent the specified vision register from being used for vision offset processing.

Argument 9: Specify the number of register storing the processing status.

- 0: Success.
- 1: Failure.

Argument 10: Specify the number of the register that stores the retrieval status of the workpiece detection information.
Specify this argument when a non-zero value is specified for Argument 8.
If 0 is specified for argument 8, specify 0 for Argument 10 as well.

- 0: Success.
- 1: Failure. No value is output to the register used to store the vision offset.

Argument 11: If you want to know the distance for a part with a specific model ID, specify the model ID.
This argument is optional and normally does not need to be specified.

CAUTION

If the target is a cell and Argument 8 or 10 is non-zero, error IRPK-057 “Specified value in argument is invalid” occurs.
If the processing status stored in the register specified by Argument 9 is 1 (NG), no values are output to the registers specified by Arguments 8 and 10.

Example:

The ID stored in R[13] is specified as the conveyor station ID and a cell is specified as the target. For this KAREL program, only 1 can be specified for Arguments 3, 4, and 5. 15 is specified as the register number used to store the distance to the upstream boundary. 6 is specified as the vision register number used to store the offset. 1 is specified as the vision register number used to store the workpiece detection information. 3 is specified as the register number used to store the processing status. 14 is specified as the register number used to store the retrieval status of detection information. Argument 11, which specifies a workpiece type and stores its expected distance, is omitted in this example.

```
CALL PKCSGETBFDIS(CStn ID=R[13], Cell, Order=1, Consec Flag=1, Quantity=1, Distance Reg=15, Offset VR=6,
Buffer VR=1, Stat Reg=3, Buffer Stat Reg=14)
```

8.15.4. Sample programs for the infeed-to-buffer-to-outfeed-conveyor configuration

This subsection provides a set of sample programs for a robot that uses a conveyor as a buffer.

These programs are similar to those used when a fixed station is used as a buffer. See “I.5.9.4. Sample programs for the conveyor-to-buffer-to-conveyor configuration(P.376)” as well.

When a conveyor is used as a buffer, either a forward or backward conveyor station is designated according to the buffer conveyor’s direction of movement.

To determine whether a workpiece should be placed on the buffer, PKCSGETBFDIS is used. Specify “Cell” for Argument 2 (target) of PKCSGETBFDIS so that the distance to the upstream boundary is returned.

To determine whether a workpiece should be picked from the buffer, call PKCSGETBFDIS in advance with “workpiece” specified for Argument 2 (target).

The sample programs presented here use the registers, Position Registers, and Vision Registers listed below. Change the register numbers as needed.

Table 8.15.4-1 Registers

R[2]	Workpiece picking status for PKCSGETBUF
R[3]	Workpiece placing status for PKCSGETBUF
R[13]	Stores the conveyor station ID of the buffer conveyor (forward direction)
R[14]	Stores the conveyor station ID of the buffer conveyor (backward direction).

R[40]	Retrieval status of workpiece detection information

Table 8.15.4-2 Position Registers

PR[31]	Pick and place position when the buffer moves in the forward direction
PR[41]	Pick and place position when the buffer moves in the backward direction

Table 8.15.4-3 Vision Registers

VR[1]	Retrieves the pick position vision offset for the pick program (specified when storing workpiece detection information in the buffer)
VR[3]	Retrieves the pick position vision offset for the pick program
VR[4]	Retrieves the placement position vision offset for the placement program
VR[5]	Retrieves the workpiece detection information stored in the buffer

8.15.4.1. Tracking program for placing workpieces on a forward-moving buffer

This program is used to place workpieces on a forward-moving buffer conveyor.

Example configuration of the tracking program used to place workpieces on a forward-moving buffer

```
1: CALL PKCSGETBUF(CStn ID=R[13],Cell,Offset VR=4,Buffer VR=0,
   Stat Reg=3,Buffer Stat Reg=0)
2: IF R[3:GETQ Status Drop]<>0,JMP LBL[900]
3: L PR[31] 10000mm/sec CNT100 VOFFSET,VR[4] Offset,PR[12:Drop Apr Z]
4: L PR[31] 10000mm/sec CNT0 VOFFSET,VR[4]
5: L PR[31] 10000mm/sec CNT100 VOFFSET,VR[4] Offset,PR[12:Drop Apr Z]
6: CALL PKCSACKBUF(CStn ID=R[13],Success)
7: LBL[900]
```

Retrieve the ID of the forward conveyor station in advance using PKCSGETID.

On the program details screen, set [Line Tracking Schedule] to the tracking schedule number specified for the forward conveyor station.

To store workpiece detection information in the buffer, on line 6 specify the register number storing detection information for Argument 3 of PKCSACKBUF as shown below.

```
6: CALL PKCSACKBUF(CStn ID=R[13],Success,1)
```

8.15.4.2. Tracking program for placing workpieces on a backward-moving buffer

This program is used to place workpieces on a backward-moving buffer conveyor.

Example configuration of the tracking program used to place workpieces on a backward-moving buffer

```
1: CALL PKCSGETBUF(CStn ID=R[14],Cell,Offset VR=4,Buffer VR=0,
   : Stat Reg=3,Buffer Stat Reg=0);
2: IF R[3:GETQ Status Drop]<>0,JMP LBL[900]
3: L PR[41] 10000mm/sec CNT100 VOFFSET,VR[4] Offset,PR[12:Drop Apr Z]
4: L PR[41] 10000mm/sec CNT0 VOFFSET,VR[4]
5: L PR[41] 10000mm/sec CNT100 VOFFSET,VR[4] Offset,PR[12:Drop Apr Z]
6: CALL PKCSACKBUF(CStn ID=R[14],Success)
7: LBL[900]
```

Retrieve the ID of the backward conveyor station in advance using PKCSGETID.

On the program details screen, set [Line Tracking Schedule] to the tracking schedule number specified for the backward conveyor station.

To store workpiece detection information in the buffer, on line 6 specify the register number storing detection information for Argument 3 of PKCSACKBUF as shown below.

```
6: CALL PKCSACKBUF(CStn ID=R[14],Success,1)
```

8.15.4.3. Tracking program for picking workpieces from a forward-moving buffer

This program is used to pick workpieces from a forward-moving buffer conveyor.

Example configuration of the tracking program used to pick workpieces from a forward-moving buffer

```
1: CALL PKCSGETBUF(CStn ID=R[13],Part,Offset VR=3,Buffer VR=0,
   : Stat Reg=2,Buffer Stat Reg=0);
2: IF R[2:GETQ Status Pick]<>0,JMP LBL[900]
3: L PR[31] 10000mm/sec CNT100 VOFFSET,VR[3] Offset,PR[11:Pick Apr Z]
4: L PR[31] 10000mm/sec CNT0 VOFFSET,VR[3]
5: L PR[31] 10000mm/sec CNT100 VOFFSET,VR[3] Offset,PR[11:Pick Apr Z]
6: CALL PKCSACKBUF(CStn ID=R[13],Success)
7: LBL[900]
```

On the program details screen, set [Line Tracking Schedule] to the tracking schedule number specified for the forward conveyor station.

To retrieve workpiece detection information stored in the buffer, on line 1 specify the vision register number storing workpiece detection information for Argument 4 of PKCSGETBUF as shown below. In addition, use Argument 6 to specify the number of the register that stores the retrieval status of the workpiece detection information.

```
1: CALL PKCSGETBUF(CStn ID=R[13],Part,Offset VR=3,Buffer VR=5,
   : Stat Reg=2,Buffer Stat Reg=40);
```

8.15.4.4. Tracking program for picking workpieces from a backward-moving buffer

This program is used to pick workpieces from a backward-moving buffer conveyor.

Example configuration of the tracking program used to pick workpieces from a backward-moving buffer

```
1: CALL PKCSGETBUF(CStn ID=R[14],Part,Offset VR=3,Buffer VR=0,
   : Stat Reg=2,Buffer Stat Reg=0);
2: IF R[2:GETQ Status Pick]<>0,JMP LBL[900]
3: L PR[41] 10000mm/sec CNT100 VOFFSET,VR[3] Offset,PR[11:Pick Apr Z]
4: L PR[41] 10000mm/sec CNT0 VOFFSET,VR[3]
5: L PR[41] 10000mm/sec CNT100 VOFFSET,VR[3] Offset,PR[11:Pick Apr Z]
6: CALL PKCSACKBUF(CStn ID=R[14],Success)
7: LBL[900]
```

On the program details screen, set [Line Tracking Schedule] to the tracking schedule number specified for the backward conveyor station.

To retrieve workpiece detection information stored in the buffer, on line 1 specify the vision register number storing workpiece detection information for Argument 4 of PKCSGETBUF as shown below. In addition, use Argument 6 to specify the number of the register that stores the retrieval status of the workpiece detection information.

```
1: CALL PKCSGETBUF(CStn ID=R[13],Part,Offset VR=3,Buffer VR=5,
   : Stat Reg=2,Buffer Stat Reg=40);
```

9. Troubleshooting

This chapter describes common issues in a tracking system using iRPickTool and their countermeasures.

9.1. When the robot does not pick up parts

Cause 1: The conveyer station is disabled.

Remedy 1: If the conveyer station has “Skip this CStn” enabled, part queue information will not be sent to this conveyor station’s robot. Disable “Skip this CStn”.

Cause 2: The load balance is set improperly.

Remedy 2: Confirm that the pickup ratio or load balance quota of the robot is not set to zero in the Conveyor menu. If the load balance quota for putting down parts is incorrect, pickup motion may not operate properly. Confirm that the load balance quota for picking up and putting down is set properly.

Cause 3: The model ID is incorrect.

Remedy 3: Confirm that the model ID of the part detected by the vision process matches the model ID specified by the load balance in the conveyer setting. If using a tray, confirm that the model ID of a cell matches the model ID of the load balance. When a model ID is specified as an argument in PKCSGETQ, PKFSGETQ, or PKFSGETBUF call, confirm that the model ID specified in the call matches the model ID set in the vision process or cell.

Cause 4: No part is detected by the vision process.

Remedy 4: Confirm that parts are detected correctly using the iRVision Runtime Image.

Cause 5: The sensor task is not started.

Remedy 5: Confirm that PKWCSTART.PC (or PKSNSTART.PC) has been executed. You may also confirm the execution status by checking the sensor or conveyor station statuses on the production status screen. For details, refer to Section 9.2 “Checking the production status”.

9.2. When parts are missed frequently

Cause 1: Processing by the robot is too slow.

Remedy 1: Disable the hot start when it is enabled.

⚠CAUTION

The iRPickTool tracking does not support hot start.

Disable Automatic Backup. In the FILE menu press the F1 [TYPE] key and select “Auto Backup”. Press the F5 key “DISABLE” in the “Automatic Backup” menu.

Cause 2: The motion of the robot is too slow.

Remedy 2: If the motion of the robot is too slow, the robot will not be able to pick parts on a moving conveyor. Adjust the speed in the motion program. When the motion speed is set to “max_speed” in the motion program, the maximum speed that can be set for the robot is automatically specified. Confirm that an appropriate override is also selected.

Cause 3: No part is detected by the vision process.

Remedy 3: If parameters of the vision process are set inappropriately, no parts may be detected. Adjust the parameters of the GPM Locator Tool or blob tool so that parts are detected correctly.

Cause 4: Vision detection is too slow.

Remedy 4: If vision processing is too slow relative to the conveyor speed, the next vision process cannot be executed at the requested timing and parts may pass by the field of view without being detected. On the iRPickTool Product Status screen, check the sensor menu to see if the vision execution is too slow (i.e., whether the Overrun Count is greater than or equal to 1). If it is too slow, implement measures such as reducing the conveyor speed or expanding the camera's field of view. When multiple vision systems are connected to one robot controller, distribute the sensors among multiple robot controllers.

Cause 5: Vision detection time is too long.

Remedy 5: 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 cases, it is necessary to pause the TP program execution and perform the vision detection process. Specifically, the third argument of PKCSGETQUE is set to a positive value, as described in "I.5.9.2.2. Pick program(P.370)". When the third argument of PKCSGETQUE is set to a positive value, PKCSGETQUE works as follows, as described in "I.5.8.5.3. PKCSGETQUE.PC(P.351)".

- If there are pickable parts, it does not wait.
- If there are no pickable parts, it waits for the 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 "I.5.9.2.2. Pick program(P.370)") is executed repeatedly, thereby extending the time before completion of vision detection.

Cause 6: The setting of the discard line is inappropriate.

Remedy 6: If the position of the discard line is too far upstream, a pickable part is determined to not be pickable and is missed. Adjust the appropriate position of the discard line by actually operating the system.

Cause 7: The tracking area is set inappropriately.

Remedy 7: Specify proper values in [Upstream] and [Downstream] in the conveyor station settings.

Cause 8: The model ID is incorrect.

Remedy 8: 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.

9.3. When the same part is picked up twice

Cause 1: Camera calibration is inappropriate.

Remedy 1: Confirm that the tool frame number that was selected during calibration and the tool frame set to the pointer are correct. Confirm that the grid spacing and focal distance of calibration data are correct.

Cause 2: The [Duplicate Tolerance] setting is inappropriate.

Remedy 2: Confirm that this setting is not too small.

Cause 3: A phantom part is falsely detected.

Remedy 3: Confirm that a wrong part is not detected in its vicinity using Vision run-time window.

9.4. Inaccuracy when a part rotates

Cause: Tracking frame has not been set correctly.

Remedy: For details, see “I.5.12.2. Adjustment of the tracking frame error(P.387)”.

9.5. Inaccuracy when a part is near the conveyor edges

Cause 1: The calibration grid used for camera calibration may be too small to calculate the lens distortion correctly.

Remedy 1: If calibration is performed with the calibration grid captured only by part of the camera's field of view, the part position detected in the section where the grid pattern is not captured will not be calculated correctly. Make the calibration grid larger so that it is fully covers the field of view, and calibrate it again.

Cause 2: If the [Part Z Height] setting specified by the vision process is incorrect, this symptom may appear.

Remedy 2: Set [Part Z Height] to the height of the feature to be detected in the tracking frame properly.

9.6. When robot motion is not smooth

Cause: The value of the average speed (updates) is small.

Remedy: If the conveyor in use does not move smoothly, the robot also may not move smoothly because it follows the motion of the conveyor. In such a case, increase [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 6. Set the value to 20 initially, and check the robot’s operation.

9.7. The stop conveyor does not work

Cause: The setup of DO values may be inconsistent.

Remedy: In multiple conveyer stations defined in the same robot controller, check if the same DO number is not specified for the stopped conveyor, and the polarities of DO are not set opposite to each other. If the polarities of DO are 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

9.8. Slow vision processing due to delayed image snap timing

Cause: Vision processing may be too slow because the history of images is getting stored in the memory device.

Remedy: Disable [Constant Logging] in [Vision Settings] in iRVision so that vision logs will not be saved.

9.9. The overrun count continues to increase

Cause: If the overrun count continues to increase on the sensor menu of the production status sensor screen, it means vision processing is too slow because the vision processing time is longer than the interval at which a vision start trigger occurs. The system will continue to operate even in this condition. However, since vision

performs the next processing after the previous detection is completed, some parts may pass by the field of view of the camera. In such a case, the setting needs to be changed.

Remedy: When detection is made each time the conveyor moves a certain distance, open the sensor setting screen and set [Trigger Distance (mm)] to the largest value possible and check if this symptom persists.

When detection is made each time the DI signal is input, adjust the interval at which parts are fed to make the interval at which the DI signal is input as long 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 [Always Save Vision Log] in [Vision Settings] in iRVision or open the vision process and set [Do Not Log] to [Image Logging Mode].
- If the Runtime Display is displayed, disable [Runtime Display] in [Vision Settings] in iRVision so it does not display the image.
- Adjust the location parameters of the 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 than vision could also reduce the vision processing time. For example, if multitasking is used, try to stop using it.

9.10. When settings cannot be changed during production

Cause: There may be an item whose settings cannot be changed during production on the setting screen.

Remedy: Due to safety concerns, iRPickTool does not allow some iRPickTool setup items to be changed during production on the setup screen. There are three types of setup items.

1. Setup items that can be changed. For example, you can change the load-balance quota even during production.
2. Setup items that can be changed but require confirmation. For example, if you want to change the upstream boundary of a conveyor station, a confirmation prompt will be displayed if you want to make this change.
3. Setup items that cannot be changed during production. For example, you will not be allowed to change the “Trigger action” of a sensor during a production run. If you try to change the trigger action of a sensor, the warning IRPK-075 “This operation is allowed only when the program is aborted” will appear. Regarding the item, “3. Setup items that cannot be changed”, these items cannot be changed during production on the setup screen. You must stop production and abort all robot programs to change the settings. All robots in the workcell should be in an aborted state.
   If any robot is paused or running, the workcell will assume the system is in production and you will not be able to make the changes.

⚠CAUTION

This troubleshooting is for settings change on the setup screen. This is different from when using a KAREL program to change the settings. To do so, see the chapter on KAREL programs.

9.11. CVIS-151 (No more vision offsets) alarm displayed during tracking motion with multi-view

Cause 1: Undetected camera view exists in multi-view vision process.

Remedy 1: Perform test execution for the vision process which is used on tracking, and check whether all camera views can detect a part. If there are any camera views that fail to detect a part, adjust the parameter of the vision process or command tool.
If an alarm occurs during production even if the test execution succeeded, the snap environment may be changed during teaching and in production. Perform teaching according to the actual situation at production as much as possible.

Cause 2:	Offset data was not calculated as the combine error exceeds the combine error limit in the multi-view vision process.
Remedy 2:	Check whether the combine error exceeds combine error limit. Adjust the combine error limit, or parameter of the vision process or command tool to reduce the combine error.

Appendix

A. Setup data file

iRPickTool data is saved in these files.

SYSPKCFG.SV	All iRPickTool data are saved in these files.
LNTKBCK.SV	All line tracking data are saved in this file.

B. Setting a teaching PC

In iRPickTool, all settings can generally be made with the touch-panel-enabled iPendant, 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

Connect the personal computer for teaching to the robot controller to prepare to start teaching the iRVision. The personal computer is only used when teaching an iRVision. It can be removed during line operation.

⚠CAUTION

This section only describes Windows and browser settings that must be changed when using a personal computer for teaching, assuming that your personal computer has standard settings. This means that you may not be able to teach an iRVision if settings not mentioned in this section are not standard.

See the following table for the tested OSes and browsers.

OS	Windows 10 Pro (64-bit) Windows 11 Pro (64-bit)
Browser	Google Chrome 76 or later Microsoft Edge

⚠CAUTION

1. The tested languages 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 communicate with the robot normally. 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 cable that meets the specifications in the following table.

Cable	Twisted part
Shield	Shielded

B.3. Connection of the communication cable

Connect the robot controller to a teach PC via the Ethernet cable. One end of the cable is connected to the Ethernet connector on the front face of the main board of the robot controller. The other end is connected to the connector of the PC marked with 品.

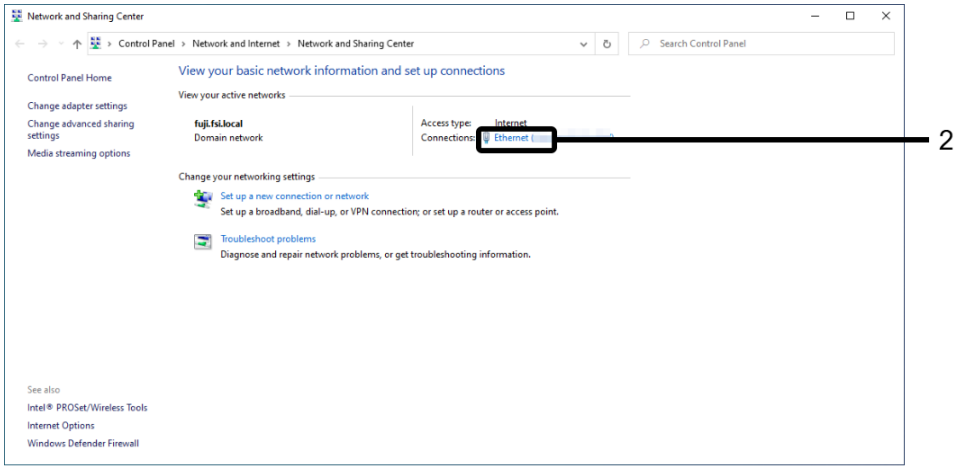
B.4. Setting an IP address for the PC

Set an IP address for the teach PC. When three robot controllers and the teach PC are connected and not connected to another network, the IP addresses can be set as shown below.

Robot controller 1	172.16.0.1
Robot controller 2	172.16.0.2
Robot controller 3	172.16.0.3
PC	172.16.0.4
Gateway	172.16.0.5
Subnet mask	255.255.0.0

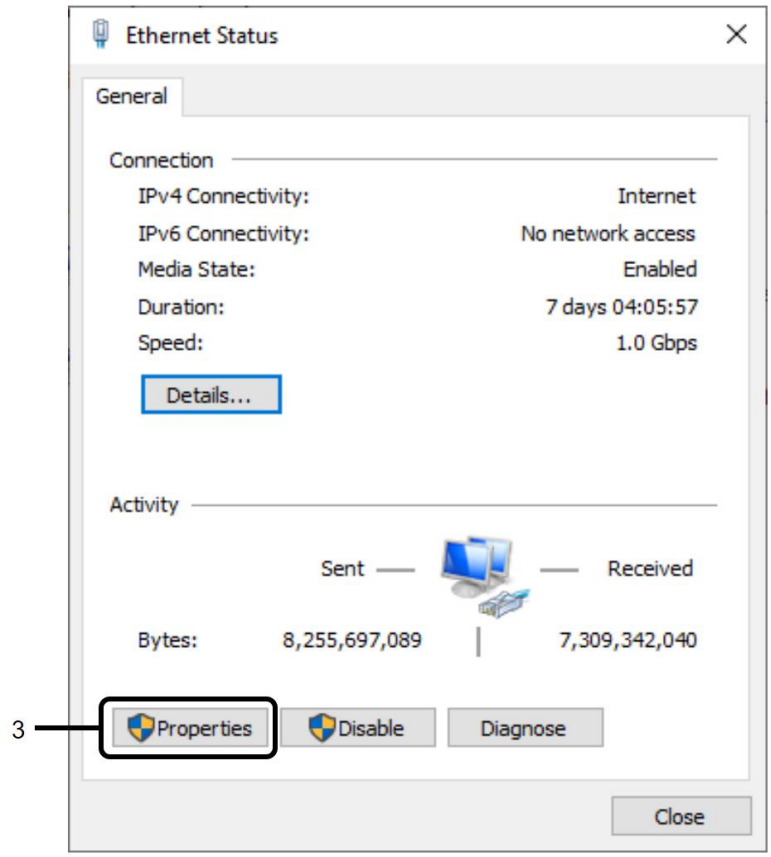
Set up the IP address using the following procedure.

1. **From [Start] → [Control Panel] of a PC, open [Network and Sharing Center].**
2. **Click the network displayed in [Connections] under [View your active networks].**

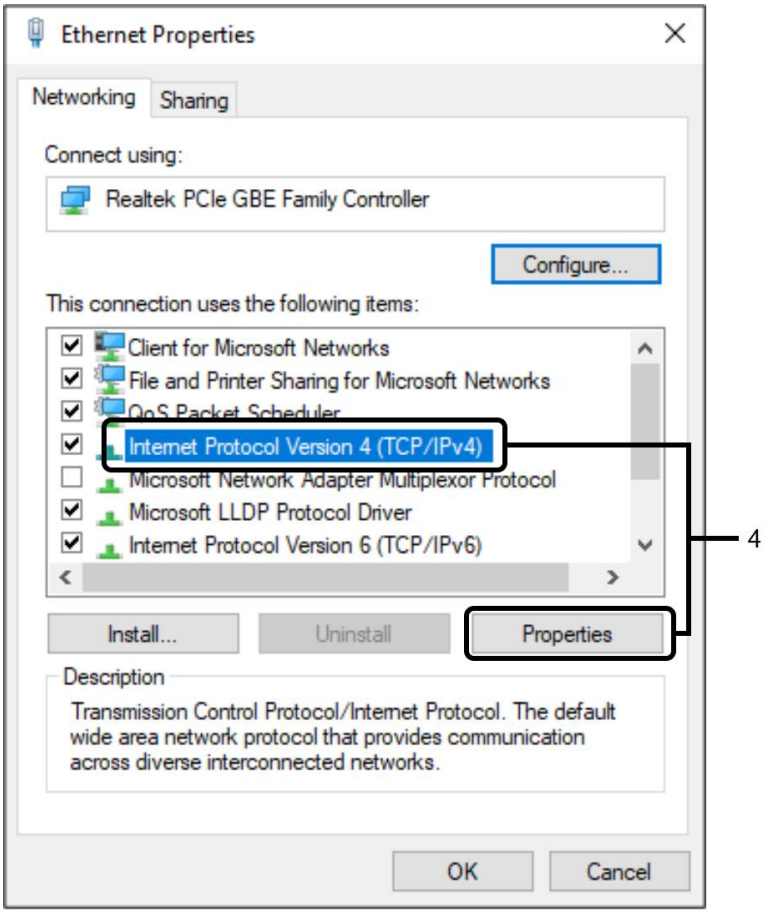


The dialog box showing the network status will appear.

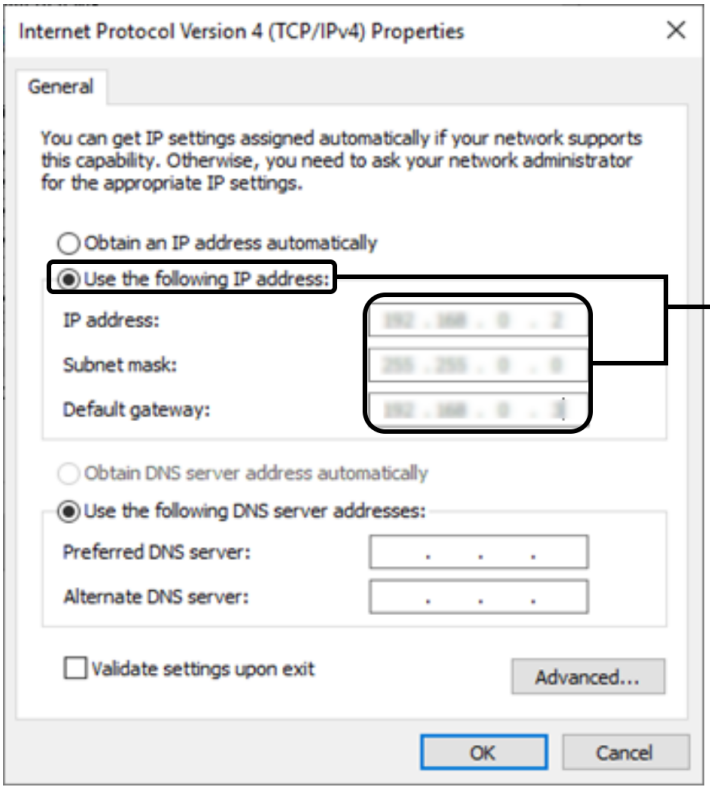
3. **Click the [Properties] button.**



4 Select [Internet Protocol Version 4 (TCP/IPv4)] and click the [Properties] button.



5 Check [Use the following IP address], enter values in [IP address], [Subnet mask], and [Default gateway], and click the [OK] button to close the window.



6 Click the [OK] button to close the window.

C. αiA4000S Pulsecoder

This chapter contains the usage of the αiA4000S Pulsecoder (A860-2052-T321).

C.1. Requirements

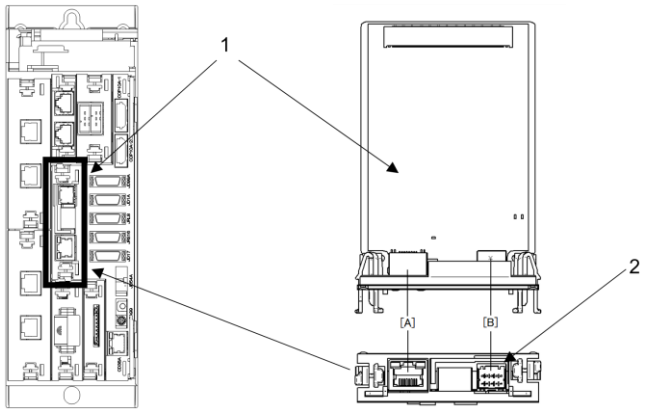
For devices required for line tracking using an R-50iA / R-50iA Mate, see each of the following tables.

Table C.1-1 Requirements for each robot controller and encoder

Used encoder	R-50iA Controller	R-50iA Mate Controller
A860-2052-T321 αiA4000S Pulsecoder (both incremental and absolute)	See Table C.1-2 .	

Table C.1-2 Requirements (when using the αiA4000S Pulsecoder A860-2052-T321).

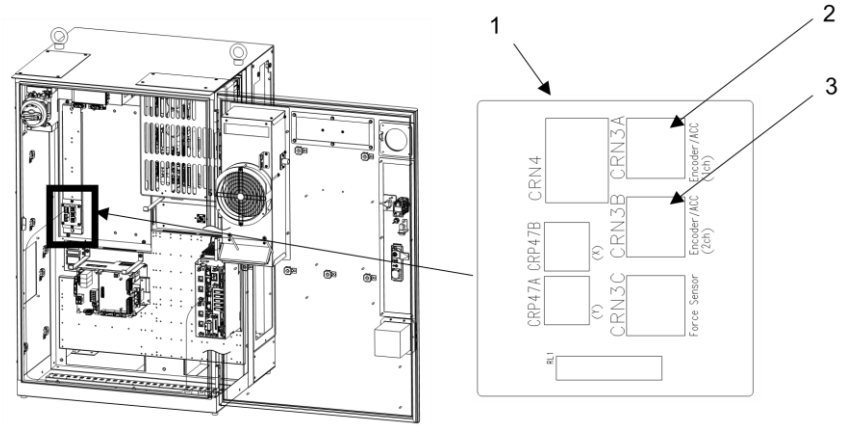
Required components	R-50iA Controller A-cabinet	R-50iA Controller B-cabinet	R-50iA Mate Controller	Comment
Hardware				
Interface device	(When using the sensor I/F board) Order specification: A05B-2700-H050#S Board specification: A20B-2103-0710			• When using one αiA4000S Pulsecoder, select the sensor I/F board. • When using two αiA4000S Pulsecoders, order the sensor distribution board in addition to the sensor I/F board.
	(When using the sensor distribution board) Order specification: A05B-2701-J031#1 Board specification: A20B-2005-0670	(When using the sensor distribution board) Order specification: A05B-2703-J031#1 Board specification: A20B-2005-0670	—	
Pulsecoder cable (when using the αiA4000S Pulsecoder in incremental mode)	A05B-2701-J210 #07MA (7M) #14MA (14M) #20MA (20M) #30MA (30M)	A05B-2703-J210 #07MD (7M) #14MD (14M) #20MD (20M) #25MD (25M) #30MD (30M)	A05B-2701-J210 #07M (7M) #14M (14M) #20M (20M) #30M (30M)	When using the αiA4000S Pulsecoder in absolute mode, the cables in this table cannot be used.



No.	Name	Comment
1	Sensor I/F board	Order specification: A05B-2700-H050#S

No.	Name	Comment
		Board specification: A20B-2103-0710
2	CRN3	Force sensor I/F 1 channel, acceleration sensor and Pulsecoder I/F 2 channels

Fig. C.1-1 Sensor I/F board



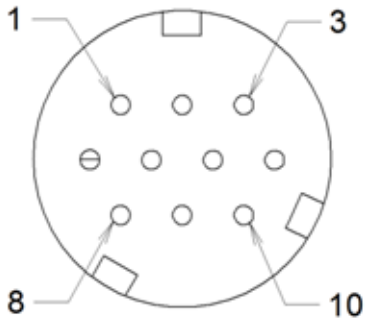
No.	Name	Comment
1	Sensor distribution board	Order specification: A05B-2701-J031#1 (A-cabinet) Order specification: A05B-2703-J031#1 (B-cabinet) Board specification: A20B-2005-0670
2	CRN3A	First channel of Pulsecoder I/F
3	CRN3B	Second channel of Pulsecoder I/F

Fig. C.1-2 Sensor distribution board

NOTE

See "C.2. Figures(P.711)" for Pulsecoder signal information, specifications, and images containing the dimensions of the Pulsecoders.

C.2. Figures



Signal name	Pin No.
SD	2
*SD	1

Signal name	Pin No.
REQ *REQ	6 5
+5 V 0 V	8, 9 7, 10
FG +6 V	3 4

Fig. C.2-1 αiA4000S Pulsecoder (A860-2052-T321) connection signal information

Table C.2-1 αiA4000S Pulsecoder (A860-2052-T321) specifications

Item		Specification
Power voltage		5 [V]+5%/-10%
Current consumption		Up to 0.2 [A]
Working temperature range		0 to +60 [℃]
Resolution		4,000,000 [/rev.]
Maximum rotation speed		6000 [min^{-1}]
Inertia moment of the input shaft		Up to 9.8×10^{-3} [$kg \cdot m^2$]
Input shaft startup torque		Up to 0.098 [N·m]
Ratio loads	Radial	98 [N]
	Axial	49 [N]
Shaft diameter runout		0.02×10^{-3} [m]
Configuration		Dust proof and drip-proof (equivalent to IP65: when using a waterproof connector)
Vibration proof acceleration		5 [G] (50 to 2,000 [Hz])
Weight		Approx.0.6 [kg]

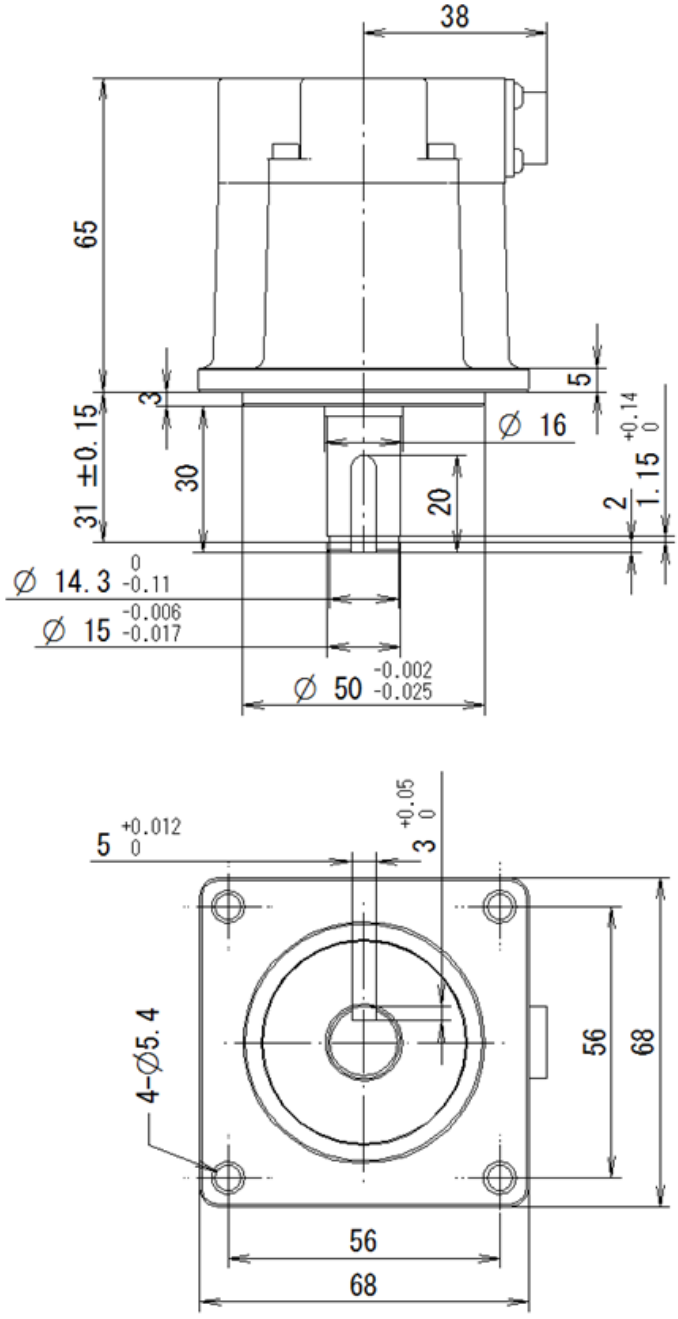


Fig. C.2-2 αiA4000S Pulsecoder (A860-2052-T321) dimensions

C.3. How to connect

The encoder configuration shown in Table C.3-1 can be used for line tracking using an R-50iA or R-50iA Mate.

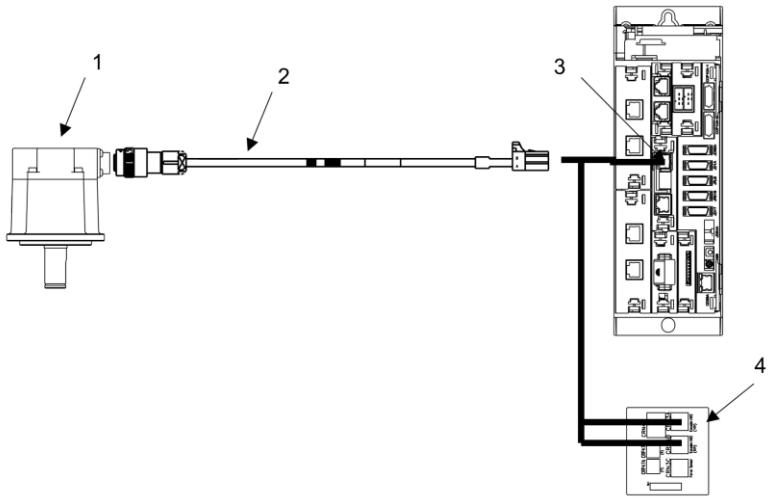
Table C.3-1 Example of encoder configuration (with αiA4000S Pulsecoder)

Number of encoders	Example of encoder configuration
1	• **Encoder #1:** Connected to the encoder terminal CRN3 of the sensor I/F board
2	• **Encoder #1, 2:**

Number of encoders	Example of encoder configuration
	Connected to encoder terminals CRN3A to CRN3B of the sensor distribution board

NOTE

- The only encoder that can be connected to the encoder terminal of the sensor I/F board or sensor distribution board is the α*i*A4000S Pulsecoder (A860-2052-T321).



No.	Name	Order specification/board specification	I/F side connector name
1	αiA4000S Pulsecoder	A860-2052-T321	
2	Line tracking cable	A-cabinet: A05B-2701-J210 #07MA (7M) #14MA (14M) #20MA (20M) #30MA (30M) B-cabinet: A05B-2703-J210 #07MD (7M) #14MD (14M) #20MD (20M) #25MD (25M) #30MD (30M) Mate controller: A05B-2701-J210 #07MA (7M) #14MA (14M) #20MA (20M) #30MA (30M)	
3	Sensor I/F board	A20B-2103-0710	CRN3
4	Sensor distribution board	A20B-2005-0670	CRN3A (first channel) CRN3B (second channel)

Fig. C.3-1 Cable connection to the sensor I/F board and sensor distribution board

D. KAREL program list

This is a correspondence table of KAREL programs of the old visual tracking software on and before the 7DC1 series and iRPickTool.

No.	Old KAREL	Description	Argument	New KAREL	Argument	Comment
1	VSTKCLRQ.PC	Clear a queue.	1: Workarea name	PKCSCLRQUE.PC	1: Conveyor station ID	ID is acquired by PKCSGETID.
2	VSTKGETQ.PC	Get a part from a queue.	1: Workarea name 2: Vision Register number to output part 3: Time to wait for part 4: Register number for status 5: Flag to handle part consecutively. 6: Model ID (optional)	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: Model ID (optional)	ID is acquired by PKCSGETID. The order of argument is changed.
3	VSTKACKQ.PC	Return acknowledgement for the part acquired from the queue.	1: Workarea name 2: Vision Register number specified in GETQ 3: Processing result	PKCSACKQUE.PC	1: Conveyor station ID 2: Processing result	You do not need to specify a vision register number as an argument.
4	VSTKSTRT.PC	Start sensor.	None	PKSNSTART.PC	None	
5	VSTKSTOP.PC	Stop sensor.	None	PKSNSTOP.PC	None	
6	VSTKSTLB.PC	Change load balance.	1: Line name 2: Register number to specify load balance 3: Number of workarea 4: Bypass 5: Model ID (optional)	PKCVSETLBD.PC	1: Conveyor name 2: Register number to specify load balance 3: Number of conveyor station 4: Bypass 5: Model ID (optional)	
7	VSTKENLB.PC	Switch load balance enabled/disabled.	1: Line name 2: Enable/Disable	PKCVENBLB.PC	1: Conveyor name 2: Enable/Disable	
8	VSTKENSC.PC	Switch stop conveyor enabled/disabled.	1: Line name 2: Register number to specify enable/disable 3: Number of workarea	PKCVENBSC.PC	1: Conveyor name 2: Register number to specify enable/disable 3: Number of conveyor station	
9	VSTKSTVN.PC	Change the vision process name.	1: Sensor number 2: Vision process name	PKSNSETVPNAM.PC	1: Sensor name 2: Vision process name	
10	VSTKSTTN.PC	Change tray name.	1: Sensor number 2: Tray name	PKCVSETTRNAM.PC	1: Conveyor name 2: Tray name	
11	VSTKGETT.PC	Output estimated time for part to arrive at the upstream boundary.	1: Workarea name 2: Register number to output estimated time 3: Flag to handle part consecutively. 4: Model ID (optional)	PKCSGETTIME.PC	1: Conveyor station ID 2: Order 3: Flag to handle part consecutively. 4: Quantity of pick	ID is acquired by PKCSGETID. Though some arguments are added, what

No.	Old KAREL	Description	Argument	New KAREL	Argument	Comment
					5: Register number to output estimated time 6: Vision Register number to output part 7: Register number for status 8: Model ID (optional)	we can do increase by them.
12	VSTKPFRT.PC	Return the prediction number of a part to be processed per minute immediately after PKCSGETPFRT.	1: Workarea name 2: Register number to output prediction pick rate	PKCSGETPFRT.PC	1: Conveyor station ID 2: Register number to output prediction pick rate	ID is acquired by PKCSGETID.
13	VSTKNPRT.PC	Return the number of parts in a queue.	1: Workarea name 2: Register number to output number of part in queue	PKCSGETNPRT.PC	1: Conveyor station ID 2: Register number to output number of part in queue	ID is acquired by PKCSGETID.

E. Sensor customization

E.1. Overview

Sensors have the following role in the workcell of iRPickTool. A sensor monitors the conveyor movement and an external signal like DI, RI or HDI. When a specified condition is met, the sensor executes an action such as execution of a vision process and puts the information of found parts in a queue.

Application engineers can customize this action.

For example, they can add their own processes unique to their application in between finding parts and putting their information into queues by customizing the sensors.

Here, the procedure of customizing a sensor and the essential points of the customization for the action are explained.

⚠CAUTION

The customer is responsible for the use of any customized programs. Be sure to perform sufficient operation checks on the actual machine before using customized programs.

E.2. Procedure to customize a sensor task

Sensor should be customized using ROBOGUIDE.

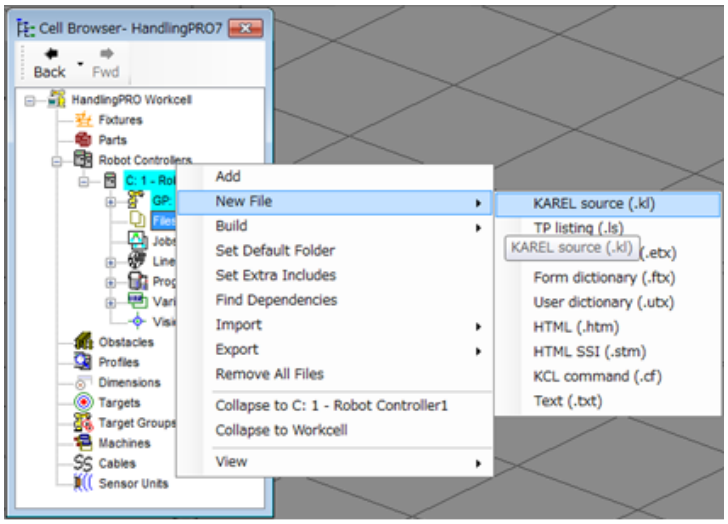
The following steps are performed in ROBOGUIDE.

1 **Make a workcell in ROBOGUIDE.**

You have to order the following option at least when you create a virtual robot.

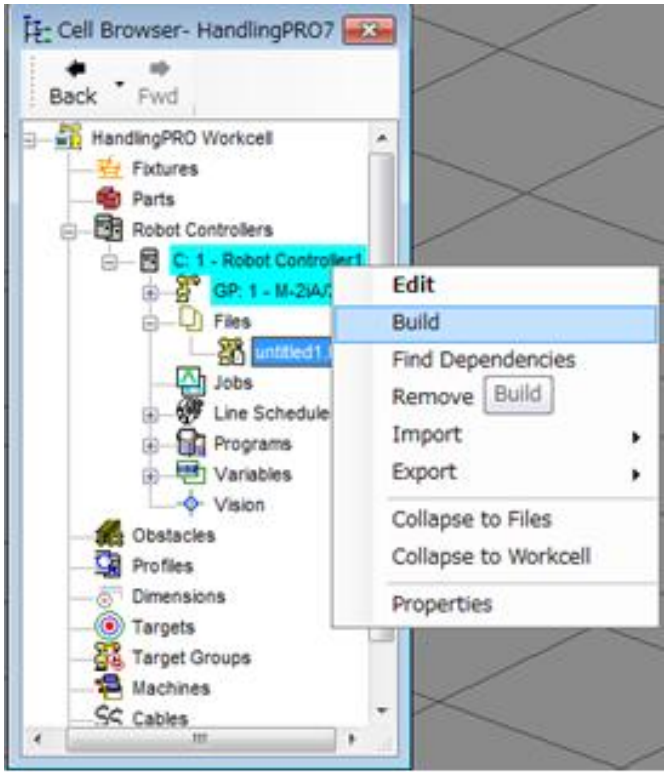
iRPickTool Basic (J944)

2 **Right click [Files] under Cell Browser and select [New File] → [KAREL source (.kl)] from the displayed menu.**



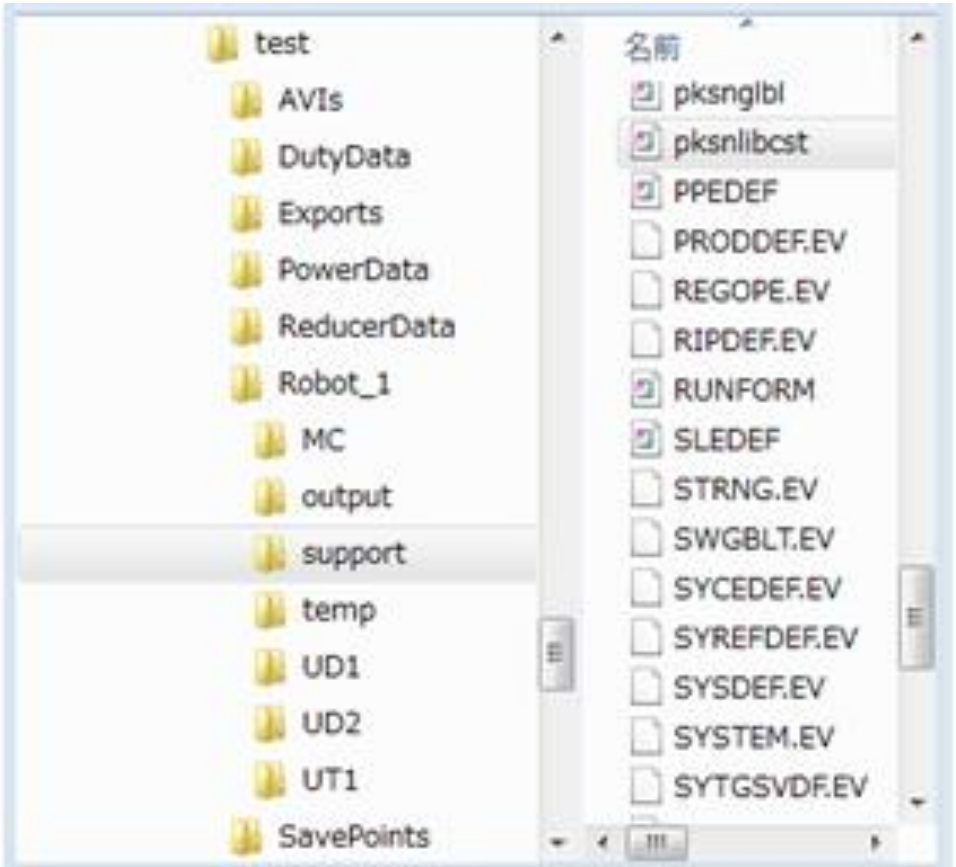
3 **Build the created KAREL source without editing anything.**

“File cannot be opened or created. …” might be displayed in another window, but the message can be ignored.



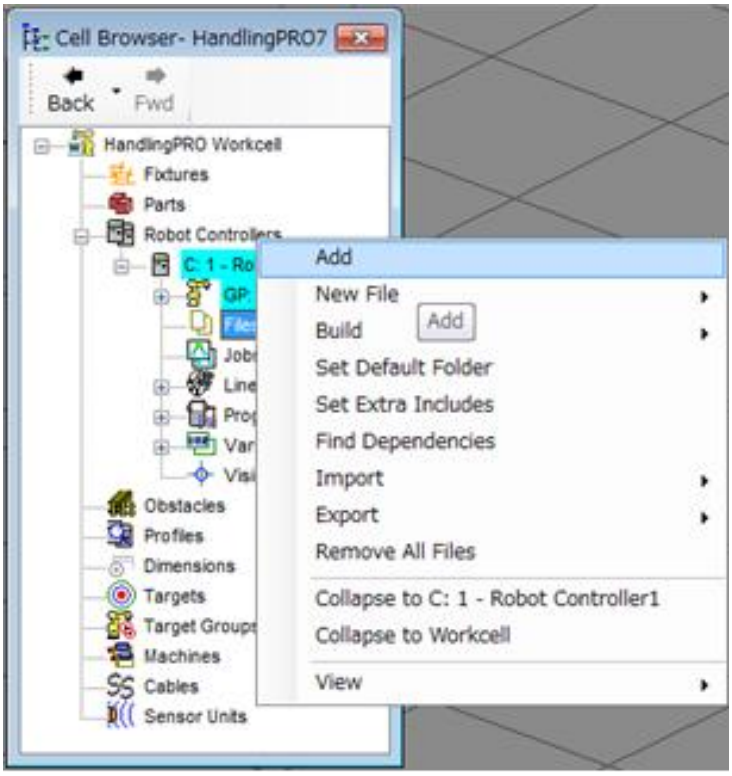
4 Click [Tool Frame] in the menu to go to the project folder.

Open the Robot_1 folder to find the sample KAREL program, PKSNLIBCST.KL, created in the [support] folder as shown in the figure below.



5 Add PKSNLIBCST.KL in the support folder to Cell Brower as shown below.

PKSNLIBCST.KL is a sample KAREL program for customization.



6 Customize PKSNLIBCST.KL.

> **⚠CAUTION**
>
> - You cannot change the program name and the function name defined in this KAREL program.
> - On the R-50iA, sample KAREL programs created for R-30iB Plus or earlier versions are not supported.

7 Build the customized PKSNLIBCST.KL.

The above-mentioned steps are performed in ROBOGUIDE.

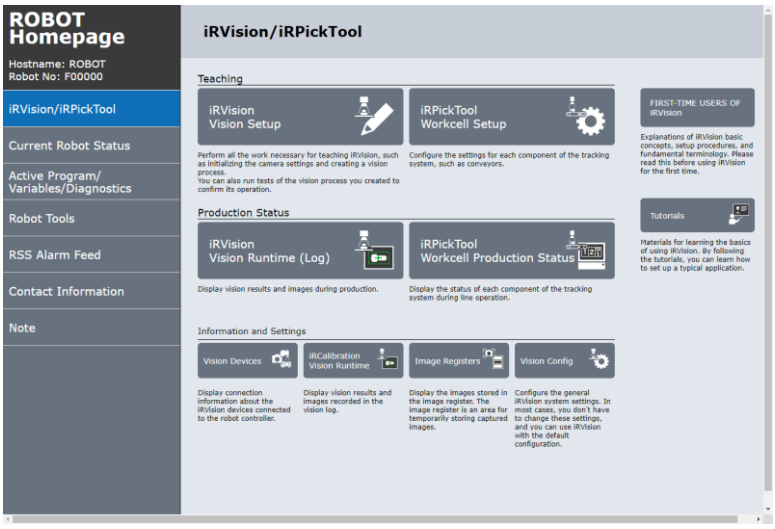
Here, all of following steps are performed in a real robot controller.

Load a compiled and built KAREL program (.PC) onto a real robot controller to use it.

8 Load the built PKSNLIBCST (.PC) onto the real controller from the FILE menu of TP in cold start mode.

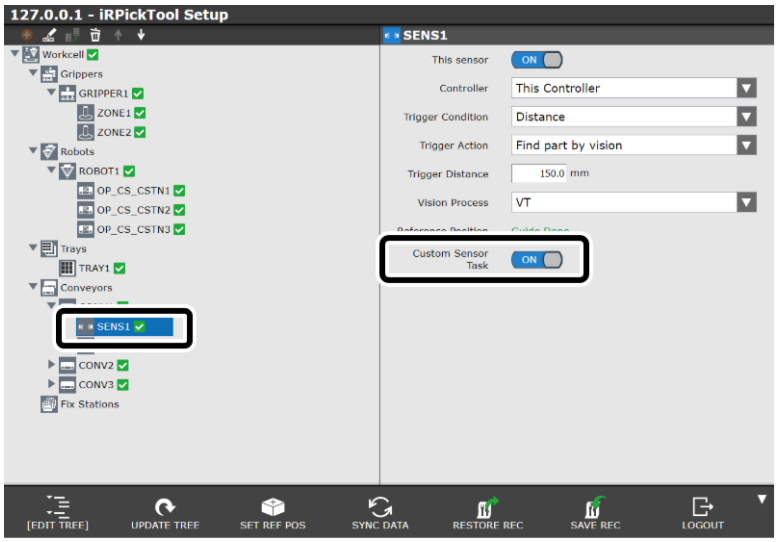
> **NOTE**
>
> The built KAREL sources loaded from the FILE menu during the cold start mode can be saved by “All of above” backup.

9 Open the robot homepage.



10 Select [iRVision/iRPickTool] in the menu area and click [iRPickTool System Setup] in the content display area.

11 Click the sensor you want to customize and set the [Customize Sensor Task] toggle switch to [ON].



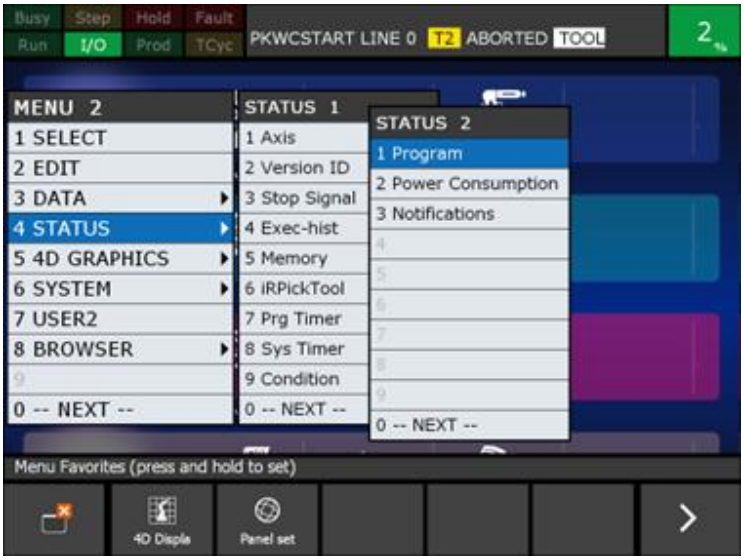
12 Start/Stop sensor by PKWCSTART/PKWCEND or PKSNSTART/PKSNSTOP in TP program as usual.

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 shown below.

13 Execute the action one or more times.

Press the [MENU] key on the teach pendant of the robot controller and select [NEXT] → [STATUS] → [Program] on the menu.



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 an ID for the system software to identify the enabled sensor.

For example, assume that SENS1 is enabled, SENS2 is disabled, SENS3 is enabled and that the system software has assigned No. 1 to SENS1 and No. 2 to SENS3.

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.

Note that IDs are not necessarily assigned from the smallest value according to the order of sensor object name (such as SENS1) numbers, though the system software automatically assigns IDs to enabled sensors.

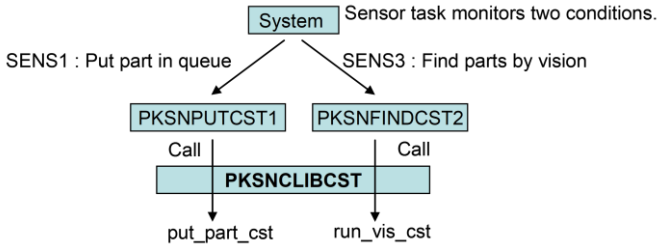


Fig. E.2-1 The flow of calling when ‘SENS1’ and ‘SENS3’ are enabled and their conditions are met.

E.3. Essential points of customization

In PKSNLIBCST.KL, the actions performed by the sample program [Put part in queue] and [Find part by vision] are described. In the sensor for which [Customize Sensor Task] is enabled in "E.2. Procedure to customize a sensor task(P.717)", the routine described here is called from the system software. For details on each function, see "E.4. Routines(P.726)" or "E.5. Built-in functions(P.732)" described later. Here the essential points of customization are explained.

Definition Part:

(Do NOT change this.)

```
--------------------------------------------------------------------------
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 -- Built-in functions for vision
%ENVIRONMENT ltrk -- Built-in functions for line tracking
%ENVIRONMENT pksndef -- Variable for sensor
%INCLUDE pksnglbl
%INCLUDE vierrdef -- Vision error definition. VEXE_E_NOMOR
--------------------------------------------------------------------------
ROUTINE update_nfnd(cond_id : INTEGER; num_found : 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
--------------------------------------------------------------------------
--------------------------------------------------------------------------
ROUTINE get_act_enc_cnt(p_cond_id : INTEGER; p_enccnt : INTEGER; p_status :
INTEGER) FROM pksnlib
--------------------------------------------------------------------------
--------------------------------------------------------------------------
ROUTINE get_idx_ecnt(cond_id : INTEGER) : INTEGER FROM pksnlib
--------------------------------------------------------------------------
--------------------------------------------------------------------------
ROUTINE is_srvo_conv(cond_id : INTEGER) : BOOLEAN FROM pksnlib
--------------------------------------------------------------------------
--------------------------------------------------------------------------
ROUTINE is_cond_dist(cond_id : INTEGER) : BOOLEAN FROM pksnlib
--------------------------------------------------------------------------
--------------------------------------------------------------------------
ROUTINE get_enc_id(cond_id : INTEGER) : INTEGER FROM pksnlib
--------------------------------------------------------------------------
--------------------------------------------------------------------------
ROUTINE get_conv_id(cond_id : INTEGER) : INTEGER FROM pksnlib
--------------------------------------------------------------------------
--------------------------------------------------------------------------
ROUTINE get_cstn_id(cond_id : INTEGER) : INTEGER FROM pksnlib
--------------------------------------------------------------------------
--------------------------------------------------------------------------
ROUTINE get_work_id(cond_id : INTEGER) : INTEGER FROM pksnlib
```

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
ROUTINE get_sens_pos(cond_id : INTEGER) : ARRAY OF XYZWPR FROM pksnlib
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
ROUTINE get_vp_name(cond_id : INTEGER) : STRING FROM pksnlib
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
ROUTINE get_tray_id(cond_id : INTEGER) : INTEGER FROM pksnlib
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
ROUTINE get_tray_pos(cond_id : INTEGER) : XYZWPR FROM pksnlib
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
ROUTINE update_wid(cond_id : 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
   conv_id : INTEGER
   cstn_id : INTEGER
   work_id : INTEGER
   sens_pos : ARRAY[MAX_NUM_VIEW] OF XYZWPR
   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
   stat = ER_SUCCESS
   -- Read the triggered encoder count
   IF is_srvo_conv(cond_id) THEN
      -- For servo conveyor (indexer)
      enc_count = get_idx_ecnt(cond_id)
   ELSE
      IF is_cond_dist(cond_id) THEN
         -- For belt conveyor where part is put every the specified distance
         get_act_enc_cnt(cond_id, enc_count, stat)
      ELSE
         -- For belt conveyor where part is put when DI or HDI is input
         enc_count = LINE_COUNT(get_enc_id(cond_id))
      ENDIF
   ENDIF

   -- Get the parameters
   conv_id = get_conv_id(cond_id)
   cstn_id = get_cstn_id(cond_id)
   work_id = get_work_id(cond_id)
```

```
   sens_pos = get_sens_pos(cond_id)

   -- Set dummy results. You can customize them before they are put
   dmy_ofs.x = 0.0                                          Here you can store the original
   dmy_ofs.y = 0.0                                          value of your application.
   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
   IF (stat = ER_SUCCESS) THEN
      put_queue(conv_id,
                cstn_id,
                work_id,
                enc_count,
                model_id,
                dmy_ofs,
                sens_pos,
                dmy_meas,
                duplicated,
                stat)
   ENDIF
   IF (stat = ER_SUCCESS) THEN
      nfound = 1
      -- Update work id. Work id is updated as below. Work id = 1,2,3...
      update_wid(cond_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
   vp_name : STRING[MAX_PROG_NAM]
   tray_id : INTEGER
   conv_id : INTEGER
   cstn_id : INTEGER
   work_id : INTEGER
   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

   -- Get the parameters
   vp_name = get_vp_name(cond_id)
   tray_id = get_tray_id(cond_id)
   conv_id = get_conv_id(cond_id)
   cstn_id = get_cstn_id(cond_id)
   work_id = get_work_id(cond_id)

   -- Find parts
   V_RUN_FIND(vp_name, -1, stat)
   WHILE (stat = ER_SUCCESS) DO

      -- Get information of found parts
      VT_GET_FOUND(vp_name, model_id, enc_count, offset, found_pos, meas,
stat)
      IF (stat = ER_SUCCESS) THEN

         -- If tray, compute tray position on track frame
         IF (tray_id <> 0) THEN
            tray_pos = get_tray_pos(cond_id)
            found_pos[1] = found_pos[1] : tray_pos
         ENDIF
                                                          Here you can store the original
                                                          value of your application.
         -- Put part information to queue                 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 the P_CVGETID and
                                                          P_CSGETID built-in functions
                                                          described later.
         put_queue(conv_id,
                   cstn_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...
            update_wid(cond_id)
         ENDIF
      ENDIF
   ENDWHILE
   IF stat = VEXE_E_NOMOR THEN
      stat = ER_SUCCESS
   ENDIF

   -- Notify of the number of found part. You must call this function in the end of this
routine.
   update_nfnd(cond_id, nfound)
END run_vis_cst
```

E.4. Routines

E.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:	cond_id is defined by system software to identify the enabled sensor. It is different from sensor id.
status:	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.

E.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:	cond_id is defined by system software to identify the enabled sensor. This may not necessarily match the sensor number.
status:	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.

E.4.3. 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:	cond_id is defined by system software to identify the enabled sensor. Although this function is used in put_part_cst and run_vis_cst, use these arguments as is. This may not necessarily match the sensor number.
num_found:	Number of found items.

E.4.4. put_queue

Purpose:

This puts part information into a tracking queue. It simultaneously performs a duplicate check, and outputs a status (= stat in Syntax below) when a part is identified as a duplicate.

When the robot of the target conveyor station is off-line, it puts a part to the next conveyor station automatically. When all robots are offline, the IRPK-210 “Controller off-line” warning occurs and SUCCESS (= 0) is returned as the 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:

cnv_id:	Specify a conveyor id where parts flow.
stn_id :	Specify a conveyor station id. The part information is put into the queue of the specified conveyor station.
work_id :	Specify the unique identifier of the part when needed. Specify 0 when you do not use the IDs.
enc_count :	This specifies an encoder value. It is the encoder value at the moment when the image is acquired in order to find the part or when a sensor such as a photo eye detects the part.
model id :	Specify the model ID for the part.
offset :	Specify the offset data of the part.
found_pos[1] to found_pos [4]:	Specify the found positions for the parts detected in camera views 1 to 4.
meas[1] to meas[10] :	Specify the measurement value for the part.
enc_count to meas :	Usually, specify the values acquired by the VT_GET_FOUND built-in function described later.

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.

stat : 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.

E.4.5. get_idx_ecnt

Purpose:

This acquires the encoder value when the servo conveyor is used.

Syntax:

get_idx_ecnt(cond_id)

Function Return Type:	INTEGER
[in] cond_id :	INTEGER

Details:

cond_id: cond_id is defined by system software to identify the enabled sensor.
Although this function is used in put_part_cst, use this argument as is.
This may not necessarily match the sensor number.

This built-in function returns an encoder value.

E.4.6. is_srvo_conv

Purpose:

This determines whether the servo conveyor is being used.

Syntax:

is_srvo_conv(cond_id)

Function Return Type:	BOOLEAN
[in] cond_id :	INTEGER

Details:

cond_id: cond_id is defined by system software to identify the enabled sensor.
Although this function is used in put_part_cst, use this argument as is.
This may not necessarily match the sensor number.

When the servo conveyor is being used, the return value is TRUE. Otherwise, the return value is FALSE.

E.4.7. is_cond_dist

Purpose:

This determines whether the trigger condition is distance.

Syntax:

is_cond_dist(cond_id)

Function Return Type:	BOOLEAN
[in] cond_id :	INTEGER

Details:

cond_id:	cond_id is defined by system software to identify the enabled sensor. Although this function is used in put_part_cst, use this argument as is. This may not necessarily match the sensor number.

When the trigger condition is distance, the return value is TRUE. Otherwise, the return value is FALSE.

E.4.8. get_enc_id

Purpose:

This acquires an encoder number.

Syntax:

get_enc_id(cond_id)

Function Return Type:	INTEGER
[in] cond_id :	INTEGER

Details:

cond_id:	cond_id is defined by system software to identify the enabled sensor. Although this function is used in put_part_cst, use this argument as is. This may not necessarily match the sensor number.

This built-in function returns an encoder number.

E.4.9. get_conv_id

Purpose:

This acquires the conveyor ID.

Syntax:

get_conv_id(cond_id)

Function Return Type:	INTEGER
[in] cond_id :	INTEGER

Details:

cond_id:	cond_id is defined by system software to identify the enabled sensor. Although this function is used in put_part_cst and run_vis_cst, use these arguments as is. This may not necessarily match the sensor number.

This built-in function returns a conveyor ID.

E.4.10. get_cstn_id

Purpose:

This acquires a conveyor station ID.

Syntax:

get_cstn_id(cond_id)

Function Return Type:	INTEGER
[in] cond_id :	INTEGER

Details:

cond_id:	cond_id is defined by system software to identify the enabled sensor. Although this function is used in put_part_cst and run_vis_cst, use these arguments as is. This may not necessarily match the sensor number.

This built-in function returns a conveyor station ID.

E.4.11. get_work_id

Purpose:

This acquires the unique identifier for the part when needed.

Syntax:

get_work_id(cond_id)

Function Return Type:	INTEGER
[in] cond_id :	INTEGER

Details:

cond_id:	cond_id is defined by system software to identify the enabled sensor. Although this function is used in put_part_cst and run_vis_cst, use these arguments as is. This may not necessarily match the sensor number.

This built-in function returns a part ID.

E.4.12. get_sens_pos

Purpose:

This acquires the set sensor position.

Syntax:

get_sens_pos(cond_id)

Function Return Type:	ARRAY OF WYZWPR
[in] cond_id :	INTEGER

Details:

cond_id: cond_id is defined by system software to identify the enabled sensor.
Although this function is used in put_part_cst, use this argument as is.
This may not necessarily match the sensor number.

This built-in function returns a sensor position.

E.4.13. get_vp_name

Purpose:

This acquires the set vision process name.

Syntax:

get_vp_name(cond_id)

Function Return Type:	STRING
[in] cond_id :	INTEGER

Details:

cond_id: cond_id is defined by system software to identify the enabled sensor.
Although this function is used in run_vis_cst, use this argument as is.
This may not necessarily match the sensor number.

This built-in function returns a vision process name.

E.4.14. get_tray_id

Purpose:

This acquires the tray ID of the tray set on the conveyor.

Syntax:

get_tray_id(cond_id)

Function Return Type:	INTEGER
[in] cond_id :	INTEGER

Details:

cond_id: cond_id is defined by system software to identify the enabled sensor.
Although this function is used in run_vis_cst, use this argument as is.
This may not necessarily match the sensor number.

This built-in function returns a tray ID. The function returns 0 if the tray is not used.

E.4.15. get_tray_pos

Purpose:

This acquires the set tray position.

Syntax:

get_tray_pos(cond_id)

Function Return Type:	ARRAY OF WYZWPR
[in] cond_id :	INTEGER

Details:

cond_id: cond_id is defined by system software to identify the enabled sensor.
Although this function is used in run_vis_cst, use this argument as is.
This may not necessarily match the sensor number.

This built-in function returns a tray position.

E.4.16. update_wid

Purpose:

This function updates the part ID.

This function usually increases the part ID by 1. 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 part ID for each cell is assigned as follow: Cell part ID = Tray part ID * 160 + Cell index − 1 The maximum part ID is 13421771 because above cell part ID should not exceed the INTEGER maximum.

Syntax:

update_wid(cond_id)

Function Return Type:	None
[in] cond_id :	INTEGER

Details:

cond_id: cond_id is defined by system software to identify the enabled sensor.
Although this function is used in put_part_cst and run_vis_cst, use these arguments as is.
This may not necessarily match the sensor number.

E.5. Built-in functions

E.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:

encoder_id: Specify an encoder number.

This built-in function returns an encoder value.

If used in put_part_cst, the accurate encoder value when the rising edge (state change from off to on) of the DI (RI, HDI) signal is detected is returned.

E.5.2. V_RUN_FIND

Purpose:

This executes a vision process.

Syntax:

V_RUN_FIND(vp_name, camera_view, status)

[in] vp_name:	STRING
[in] camera_view:	INTEGER
[out] status:	INTEGER
%ENVIRONMENT GROUP:	CVIS

Details:

vp_name:	Specify the name of the vision process.
camera_view:	Specify a camera view number. If 0 is specified, all the camera views of the vision process are executed.
status :	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 function finishes just after the vision process finishes acquiring an image. The vision process 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.

E.5.3. VT_GET_FOUND

Purpose:

This outputs the information of a part that a vision process 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:

vp_name :	Specify the name of the vision process.
model_id :	The model ID of the found part is returned to model_id.
enc_count :	The encoder value retrieved at the moment the image is acquired to find the part is returned to enc_count.
offset :	The offset data of the found part is returned to offset.
found_pos[1] to found_pos [4]:	The found positions for the parts detected in camera views 1 to 4 are returned to found_pos[1] to found_pos[4]. iRPickTool only uses camera view 1, except for multi-view.
meas_val[1] to meas_val[10] :	The measurement value of the found part is returned to model_id. The measurement values are specified in the measurement output tool of a vision process.
status :	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 function is used after the V_RUN_FIND built-in function.

If vision process finds more than one part, call this built-in function repeatedly until its status is not 0. When there is no more part to get, the status becomes 117151.

E.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:

conv_name :	Specify a conveyor name.
group_num:	Specify a robot group number.

This built-in function returns the ID of the conveyor station where the robot specified with group_num operates.

E.5.5. P_CVGETID

Purpose:

It returns the conveyor ID acquired from a specified conveyor name.

Syntax:

P_CVGETID(conv_name)

Function Return Type:	INTEGER
[in] conv_name:	STRING
%ENVIRONMENT Group:	PKQU

Details:

conv_name : Specify a conveyor name.

This built-in function returns the conveyor number.

F. Vision I/F add-on

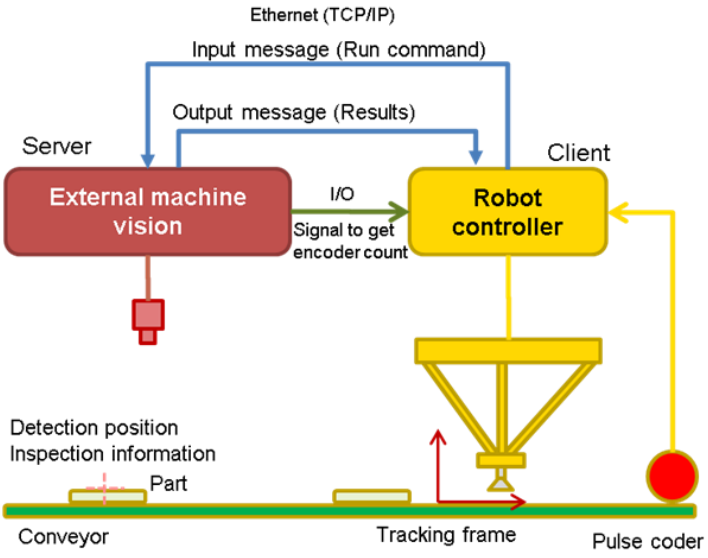
This appendix describes the iRPickTool Vision I/F add-on.

F.1. Overview

The iRPickTool Vision I/F add-on is a function to perform 2D visual tracking from the part detection result acquired from the external machine vision through TCP/IP communication. The TP program that communicates with the external machine vision is executed as an action of the iRPickTool sensor. This TP program executes the following processes:

- Send the start command to the external machine vision and receive the detection result.
- Receive the signal the external machine vision outputs when snapping (encoder acquisition signal) to acquire the encoder count.
- Convert the detection position into the coordinates of a position on the tracking frame.
- Input the detection result into the queue.

Use the KAREL program to exchange messages with the external machine vision. The customer determines the message format in line with their external machine vision and system.



The following software options are required to use the iRPickTool Vision I/F add-on.

- J944 (iRPickTool Basic) or J945 (iRPickTool)
- R648 (user socket message function)
- S521 (iRPickTool Vision I/F add-on)

F.2. Terms

This appendix describes terms specific to the iRPickTool Vision I/F add-on.

Data

Each element of a message separated with a separator.

Input message

Message that the robot controller sends to the external machine vision. The message ends with a line feed character. The input message contains the following elements.

Start command

Snap, detection, and detection result transmission commands for the external machine vision. For details, see the operator's manual from each vision manufacturer.

Output message

Message that the robot controller receives from the external machine vision. The message ends with a line feed character. The output message contains the following elements.

Detection result

Part information that the external machine vision outputs to the robot controller. The customer selects necessary information depending on the system and determines the message format.

Response

Status output when the external machine vision receives a command. For details, see the operator's manual from each vision manufacturer.

Action program

Program that communicates with the external machine vision and inputs the detection result into the queue.

Latch program

Program that acquires the encoder count at the moment when the external machine vision snaps an image.

Encoder acquisition signal

Signal used for latch. The robot controller acquires the encoder count when this signal is input. A signal that synchronizes with snapping, such as the strobe signal, is used.

F.3. Restrictions

This section describes restrictions on the robot system and external machine vision that the iRPickTool Vision I/F add-on can use.

F.3.1. Restrictions on the robot system

- This is only applicable to 2D visual tracking.
- The external machine vision system is only applicable to linear conveyors.
- When multiple external machine vision systems are used for the same robot controller, you must use different encoders for each of the conveyor stations that support the sensors using the external machine vision systems.
- Trays cannot be used with conveyors that use external machine vision systems.
- External machine vision systems cannot be connected to a dual arm robot.
- External machine vision systems cannot be used with pre-grouped infeed conveyors.

F.3.2. Restrictions on the external machine vision systems

- Communication must be possible with the TCP/IP protocol via Ethernet.
- Communication must be established in TCP/IP ASCII format.
- The external machine vision system must be able to act as a TCP/IP server. The robot controller cannot act as a server.
- About input messages
  - The external machine vision system must be able to execute snapping and detection by using an input message sent from the robot controller.
  - The character length of each data item in the input message must be 34 characters or less.
  - The line feed characters, LF, CR, CR+LF, or none must be supported.
- About output messages
  - When detection execution is complete or an input message for obtaining detection results is received, an output message (detection results) must be sent to the robot controller.
  - At least the workpiece detection position (X, Y, R) must be sent as the detection results.
  - The character length of each data item in the output message must be 254 characters or less.
  - Each data item in the detection results must be delimited by a “,” (comma) or “ ” (space).
  - It must be possible to append a line feed character, LF, CR, or CR+LF to the end of the detection results.
- The third party vision must be able to output signals to the robot controller when snapping takes place.

F.4. About the support scope

This section describes the support scope.

Software part

- FANUC supports only the functions of the robot controller.
- To check for support for camera snapping, detection of targets in images, inspection sections, etc., check with each vision manufacturer.
- FANUC’s robot controller operates when the external machine vision sends a correct detection result and sends signals at correct times.

Hardware part

- FANUC supports only the robot and robot controller.
- For vision system parts including camera, lens, and cable, check with each vision manufacturer.

For details on the support scope, see Fig. F.4-1 below.

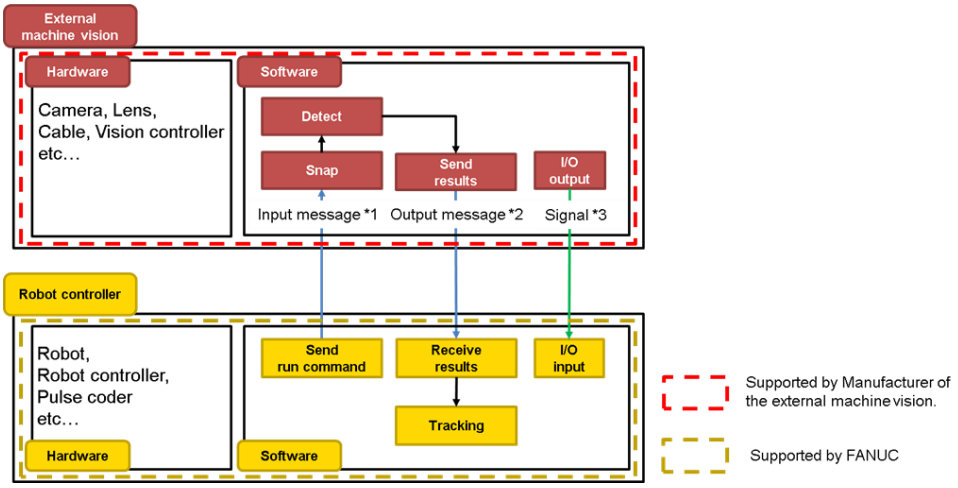


Fig. F.4-1 Support scope

*1 FANUC supports the scope up to sending the determined input message to the external machine vision via TCP/IP.
The external machine vision manufacturer supports processing such as detection based on the input message that the external machine vision receives.

*2 The customer that operates the robot determines the output message format in line with their system.
The external machine vision manufacturer supports the creation of output messages using the format and their transmission via TCP/IP from the external machine vision vision manufacturer.
FANUC supports the extraction of the detection result from the output message that the robot controller receives and tracking.

*3 The external machine vision manufacturer supports the transmission of signals used to acquire the conveyor position when snapping images (signals that synchronize with snapping, such as the strobe signal) and when sending the signals.
FANUC supports the reception of these signals and the acquisition of the conveyor position at that time.

F.5. Startup procedure

This section describes how to start up the iRPickTool Vision I/F add-on.

This section describes a sample system that uses the same configuration as the one in "I.2. iRPickTool Basic (basic functions) startup procedures(P.26)". To use the iRPickTool Vision I/F add-on, change the combination of infeed conveyors as follows.

Infeed: [Distance] + [Run program]

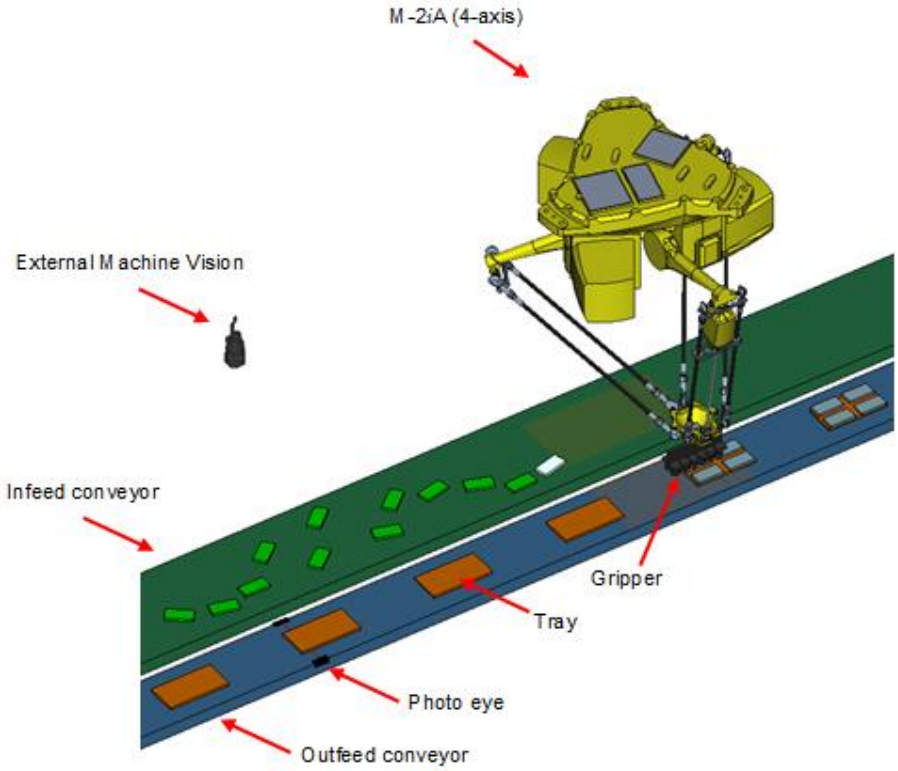


Fig. F.5-1 System configuration

Here, a case where the external machine vision and the robot controller communicate with each other using messages described in "Table F.5-1" is described as an example. When actually applying these message formats, change them according to the specifications of your external machine vision.

Table F.5-1 Message formats

Input message (Robot controller → External machine vision)	RUNFIND(string)<CR>
Response to an input message (External machine vision → Robot controller)	Normal : OK(string)<CR> Abnormal : ER(string)<CR>
Output message (External machine vision → Robot controller)	Num. Found(integer number),Model ID(integer number),X(real number),Y(real number),R(real number),Measurement 1(real number)<CR>

F.5.1. External machine vision system connection and settings

Make connection settings so that the external machine vision system and the robot controller can communicate with each other. For the setting procedures, see the operator's manual from the relevant vision system manufacturer.

F.5.1.1. External machine vision system connection

- Connect the external machine vision system and the robot controller via the Ethernet cable.
- Connect the signals that synchronize with snapping images such as the strobe signal to the DI or RI of the robot controller as an encoder acquisition signal.

NOTE

Use an independent Ethernet port, which is not used for other communications, for connection with the external machine vision system.

F.5.1.2. External machine vision system settings

- Make settings so that the external machine vision system can be connected to the robot controller by using a TCP/IP protocol via Ethernet.
- Teach it so that workpiece detection can be performed.
- Make settings to send the detection results to the robot controller in the format of the output message.
- Set the encoder acquisition signal.

Record the following items as they will be used in the robot controller settings.

- IP address and port number to be used for TCP/IP.
- Start command for the external machine vision system to send snap, detection, and detection results.
- DI number or RI number of the robot controller to which the encoder acquisition signal is connected.

F.5.2. Communication settings

Make settings related to the external machine vision system.

F.5.2.1. IP address setting

1. **Press the [MENU] key on the robot controller's teach pendant.**
2. **Select [SETUP] → [Host Comm] from the menu.**
3. **Select TCP/IP and press F3 [DETAIL].**
4. **Set an IP address and subnet mask.**

F.5.2.2. Setting up a client tag

Setting up the host communication screen

1. **Press the [MENU] key on the robot controller's teach pendant.**
2. **Select [SETUP] → [Host Comm] from the menu.**
3. **Select F4 [Display] → [2 Client].**
4. **Move the cursor to the client tag to be used, and press the [ENTER] key.**
5. **Move the cursor to [Protocol] to be used and press the [ENTER] key.**
6. **Select [SM].**
7. **Move the cursor to [Server IP/host name] to be used, and press the [ENTER] key.**
8. **Enter the IP address of the external machine vision.**
9. **Place the cursor over [Remote port], enter the port number of the external machine vision, and press the [ENTER] key.**
10. **Move the cursor to [Status at startup] and press the [ENTER] key.**
11. **Select [Define].**
12. **Press F2 [OPERATION] and select [Define].**

F.5.3. Setting up iRPickTool

Set items common to "I.2. iRPickTool Basic (basic functions) startup procedures(P.26)". Use the same settings for the following items when using the iRPickTool Vision I/F add-on as well. Set them, seeing each item in "I.2. iRPickTool Basic (basic functions) startup procedures(P.26)".

- "I.2.2.1. Connection and setup of Pulsecoders(P.27)"
- "I.2.2.3. Checking the input of the trigger sensor(P.36)"
- "I.2.2.4. Setting up the tool frame of the pointer tool(P.37)"
- "I.2.2.6. Setting payloads (motion performance)(P.58)"
- "I.2.3.1. Setting up a workcell(P.60)"
- "I.2.3.2. Setting up a tray(P.67)"
- "I.2.3.3. Setting up a conveyor(P.71)"
- "I.2.3.4. Setting a tracking frame(P.72)"
- "I.2.3.5.2. Setting up an outfeed conveyor sensor (In the case of [DI] / [RI] + [Put part in queue])(P.81)" or "I.2.3.5.3. Setting up an outfeed conveyor sensor (In the case of [HDI] + [Put part in queue])(P.83)"
- "I.2.3.6. Setting the sensor position(P.84)"
- "I.2.3.8. Setting up a conveyor station(P.106)"
- "I.2.3.9. Setting up a tool frame for the hand(P.111)"
- "I.2.3.10.2. Setting up a reference position for the outfeed conveyor and teaching a robot position(P.122)"

F.5.4. Creating a robot program

Create a robot program. For how to create a program, see "I.2.2.5. Creating a robot program(P.42)".

Create the following programs for robot programs. The tracking program and vacuum ON/OFF program are the same as those created in "I.2.2.5. Creating a robot program(P.42)" and not described here. Create a program as described in "I.2.2.5. Creating a robot program(P.42)".

- Main program
- Tracking program
- Vacuum ON/OFF program

The iRPickTool Vision I/F add-on requires a program that communicates with the external machine vision and acquires detection results. This program is executed from the sensor task as an action in iRPickTool every time the trigger condition is met. For the relationship with the main program, see Fig. F.5.4-1.

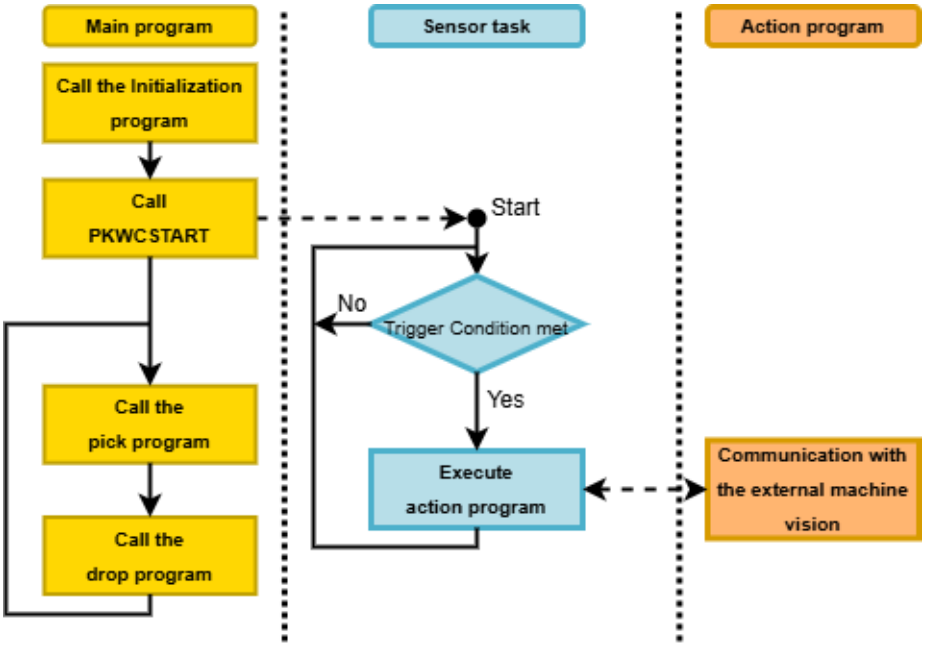


Fig. F.5.4-1 Program to be executed

Create the following program that communicates with the external machine vision and acquires detection result.

- Action program
- Latch program
- Detection program

NOTE

Enable the setting to ignore pause in the above programs. If a program ends due to an emergency stop or other cause during communication, part information could be erased before input.

These programs execute the following processes. For the program flowchart, see Fig. F.5.4-2.

1. **Send the start command to the external machine vision and receive the detection result.**
2. **Receive the signal the external machine vision outputs when snapping (encoder acquisition signal) to acquire the encoder count.**
3. **Convert the detection position into the coordinates of a position on the tracking frame.**
4. **Input the detection result into the queue.**

Create the following program to initialize connection with the external machine vision, etc.

- Initialization program



Fig. F.5.4-2 Flowchart of the action program

The iRPickTool Vision I/F add-on creates a program to calculate the coordinate conversion parameter and set the reference position.

- Coordinate conversion parameter calculation program
- Reference position setting program

Defining registers

These programs use the following registers. Change the register numbers as needed.

Table F.5.4-1 Registers

R[2:GETQ Status Pick]=0	Pick part processing status
R[3:GETQ Status Drop]=0	Place part processing status
R[11:CONV1 ID]=1	CONV1 (pick) ID
R[12:CONV2 ID]=2	CONV2 (place) ID
R[21:Pick AP_LD]=30	Pick approach distance specification [mm]
R[22:Pick RT_LD]=30	Pick retract distance specification [mm]
R[23:Pick Wait]=.05	Index delay at pick position [sec]
R[24:Pick CNT]=0	Pick position CNT specification
R[25:Pick TB]=.1	TIME BEFORE time (VACUUM_ON) [sec]

R[26:Pick Speed1]=4000	Pick program speed 1 [mm/sec]
R[27:Pick Speed2]=4000	Pick program speed 2 [mm/sec]
R[28:Pick Speed3]=2000	Pick program speed 3 [mm/sec]
R[31:Drop AP_LD]=30	Place approach distance specification [mm]
R[32:Drop RT_LD]=30	Place retract distance specification [mm]
R[33:Drop Wait]=.05	Index delay at drop position [sec]
R[34:Drop CNT]=0	Drop position CNT specification
R[35:Drop TB]=.1	TIME BEFORE time (VACUUM_OFF) [sec]
R[36:Drop Speed1]=2000	Placement program speed 1 [mm/sec]
R[37:Drop Speed2]=2000	Placement program speed 2 [mm/sec]
R[38:Drop Speed3]=4000	Placement program speed 3 [mm/sec]
R[40:Num. to get Enc Cnt]=2	DI/RI number to acquire the encoder count (Specify the number to be used)
R[41:Client number]=1	Client tag number (Specify the number to be used)
R[43:SENDMSG Status]=0	Status of PKMVSENDMSG
R[44:RCVMSG Status]=0	Status of PKMVRCVMSG
R[45:PRSMSG Status]=0	Status of PKMVPRSMSG
R[46:MVSFIND1 Status]=0	Status of the detection result of the external machine vision *1
R[47:End MVSACTION1]=0	Flag indicating that MVS ACTION1 ended
R[48:LATCH succeeded]=0	Flag indicating that latch has succeeded
R[49:OverlapStat Reg]=0	Duplication result when a detection result is input into the queue
R[51:x0]=0	x0 (position at which the calculation of the coordinate conversion parameter is detected)
R[52:y0]=0	y0 (position at which the calculation of the coordinate conversion parameter is detected)
R[53:x1]=0	x1 (position at which the calculation of the coordinate conversion parameter is detected)
R[54:y1]=0	y1 (position at which the calculation of the coordinate conversion parameter is detected)
R[55:x2]=0	x2 (position at which the calculation of the coordinate conversion parameter is detected)
R[56:y2]=0	y2 (position at which the calculation of the coordinate conversion parameter is detected)
R[60:Enc Cnt]=0	Encoder count (detection result)
R[61:Num. Found]=0	Number of detections (detection result)
R[62:Model ID]=0	Model ID (detection result)
R[63:x]=0	x (detection result)
R[64:y]=0	y (detection result)
R[65:r]=0	r (detection result)
R[66:Measurement Value 1]=0	Measurement Value 1 (detection result)

* 1 The status of the detection result in the external machine vision is defined as follows in the TP program.

0: PUTQUE succeeded
1: An error occurred during communication with the external machine vision
2: The number of data items in the message analyzed with PKMVPRSMSG is different
3: Response error
4: Zero detections
5: No encoder acquisition signal input

Table F.5.4-2 Position Registers

PR[1:HOME POS]=R	Perch Position for all groups
PR[7:Pick Pos]=R	Pick position
PR[8:Drop Pos]=R	Drop position
PR[11:Pick Apr Z]=R	Offset from pick position

PR[12:Drop Apr Z]=R	Offset from drop position
PR[21:ADJ_OFS]=R	For pick position offset adjustment
PR[26:Ref Pos]=R	Reference position
PR[27:Detection Pos]=R	Detection position
PR[28:Offset]=R	Offset

Table F.5.4-3 String Registers

SR[6 :Response]	Response
SR[7 :OK Response]	Response at normal time (Specify response at normal time)

Table F.5.4-4 Vision Registers

VR[1]	For pick offset instruction
VR[2]	For drop offset instruction

F.5.4.1. Main program

This section describes a sample main program for the system.

PNS0001

```
  1:   !PKCSGETID (Get conveyor station ID)
  2:   !*1:Conveyor name
  3:   !*2:Register number for conveyor station ID
  4:   !*3:Group number (Can be omitted)
  5:   CALL PKCSGETID('CONV1',CStn ID Reg=11)                KAREL program
  6:   CALL PKCSGETID('CONV2',CStn ID Reg=12)                KAREL program
  7:
  8: ! Initializes the external machine vision
  9: CALL MVSINIT1                                          Initialization
 10:
 11:   UFRAME_NUM=0
 12:   UTOOL_NUM=1
 13: PAYLOAD[1]
 14: L PR[1:HOME POS] 500mm/sec FINE Wjnt
 15:
 16: !--- System startup ---
 17: !PKWCSTART (System startup, including check and clear)
 18: CALL PKWCSTART                                         KAREL program
 19:
 20: LBL[101]
 21: !--- Pick program ---
 22: CALL PT_PICKCS
 23: IF DI[1:CYCLE_STOP]=ON,JMP LBL[901]
 24: !--- Placement program ---
 25: CALL PT_DROPCS
 26: IF DI[1:CYCLE_STOP]=ON,JMP LBL[901]
 27: JMP LBL[101]
 28:
 29: LBL[901]
 30: UFRAME_NUM=0
 31: UTOOL_NUM=1
 32: L PR[1:HOME POS] 500mm/sec FINE Wjnt
 33:
 34: !PKWCEND (System end)
```

```
35: CALL PKWCEND
[End]
```

This program initializes iRPickTool and the external machine vision, and performs the tracking operation.

The TP program is called on line 9 to initialize connection with the external machine vision, etc. For details, see "F.5.4.3. Initialization program(P.748)".

Other lines are the same as the main program in "I.2.2.5. Creating a robot program(P.42)". The programs called on lines 22 and 25 are the same as the pick program and placement program in "I.2.2.5. Creating a robot program(P.42)". Create the pick program and placement program as described in "I.2.2.5. Creating a robot program(P.42)".

F.5.4.2. Action program

This section describes a sample program executed as an action in iRPickTool.

MVSACTION1

```
 1: R[47:End MVSACTION1]=0
 2:
 3: !Uses an accurate trigger
 4: Accurate trigger tracking[1]                          Uses an accurate
                                                          trigger
 5:
 6: R[48:LATCH succeeded]=0
 7: Run LATCH                                             Runs the latch
                                                          program
 8:
 9: CALL MVSFIND1                                         Acquires the
                                                          detection result
10: IF R[46:MVSFIND1 Status]<>0,JMP LBL[990]
11:
12: IF R[48:LATCH succeeded]=0,JMP LBL[900]               Checks whether
                                                          LATCH succeeded or
                                                          failed
13:
14: ! Stores the detection position in the Position Register   Stored in the Position
                                                                Register
15: PR[27,1:Detection Pos]=R[63:x]
16: PR[27,2:Detection Pos]=R[64:y]
17: PR[27,3:Detection Pos]=0
18: PR[27,4:Detection Pos]=0
19: PR[27,5:Detection Pos]=0
20: PR[27,6:Detection Pos]=R[65:r]
21:
22: ! Converts the detection position into coordinates and calculates offset   Converts the
                                                                               detection position
                                                                               into coordinates
23: CALL PKMVCNVPX2MM("Action ID"=1,"Fnd Pos PR"=27,"Cnv Pos PR"=27)
24: CALL PKMVCALOFS2D("Fnd Pos PR"=27,"Ref Pos PR"=26,"Offset PR"=28)
25:
26: ! Inputs the detection result into the queue
27: CALL PKCSPUTQUE2("CStn ID"=R[11:CONV1 ID],"Enc Cnt"=R[60],   Inputs the detection
                                                                result into the queue
  "Model ID R"=R[62],"Fnd Pos PR"=27,"Offset PR"=28,"Meas. R"=66,
  "Meas.Cnt"=1,"OverlapStat Reg"=49)
28: JMP LBL[990]
```

```
29:
30: LBL[900:Latch error]                                    Error processing
31: R[46:MVSFIND1 Status]=5
32:
33: LBL[990]
34: R[47:End MVSACTION1]=1
35: CALL PKMVWAITTASK("Task Name"='LATCH')
 [END]
```

This program is executed as an action in iRPickTool. This program acquires the detection result from the external machine vision and inputs it into the queue.

The accurate trigger is set up on line 4. When using a tracking schedule number other than 1, change this argument to the tracking schedule number to be used.

The latch program to acquire the encoder count when a signal is input from the external machine vision is executed on line 7. For details, see "F.5.4.5. Latch program(P.750)".

The program to acquire the detection result from the external machine vision is called on line 9. For details, see "F.5.4.4. Detection program(P.749)".

Whether LATCH succeeded is checked on line 12. If the signal for acquiring the encoder count is output late or if the output width is long, the signal could not be detected before this check, and as a result, latch may be determined as having failed. In this case, use the signal output at the same time as snapping images to acquire the encoder count and set a short signal output width. Or, add the wait instruction before this line.

Detection positions are stored in Position Registers on lines 15 to 20.

The detection result is converted into coordinates on line 23 and offset is calculated on line 24.

The result is input into the queue as a part on line 27.

The labels to which the processing jumps if an error occurs are defined on lines 30 and 31. Specify error processing here if necessary.

1 is stored in R[47] to notify the latch program of the end of the action program, which ends on line 34. The processing waits until the latch program ends on line 35.

⚠CAUTION

Execute PKMVSENDMSG and PKMVRSVMSG only in the action program.
If multiple programs attempt to communicate at the same time, alarm INTP-326 " (PKMVLIB, 207) File variable is already in use" occurs. However, this does not cause problems if the action program does not use the relevant communication port.

F.5.4.3. Initialization program

This section describes a sample initialization program.

MVSINIT1

```
1: CALL PKMVCONNECT("Client ID"=R[41:Client number])          Connects with the external
                                                              machine vision
2:
3: CALL PKMVINIACT("Action ID"=1,
                   "Client ID"=R[41:Client number],           Initializes the action
                    "In Msg Delimiter"=',',
                    "In Msg Newline char"='CR',
                    "In Msg Max num data"=10,
                    "Out Msg Delimiter"=',',
```

```
                                        "Out Msg Newline char"='CR',
                                        "Out Msg Max num data"=20)
4:
5: CALL PKMVSETSNDPM("Action ID"=1,"Stat Reg"=43,'RUNFIND')      Sets the transmission parameter
6: CALL PKMVSETRCVPM("Action ID"=1,"Timeout(ms)"=5000,"Stat Reg"=44)  Sets the reception parameter
7:
8: LBL[990]
[End]
```

The external machine vision is connected on line 1. Enter the client number to be used in R[41] in advance.

The action is initialized on line 3. For details on the arguments, see “F.6.2.1. PKMVINIACT.PC(P.756)”.

The input message is set up on line 5. Specify the start command for the external machine vision in the input message.

The parameters at reception are set up on line 6. Specify a timeout longer than the time taken for the snap and detection by the external machine vision.

NOTE

In some external machine visions, the above program alone may not complete connection. For example, login may be required at the time of connection. In this case, add processing necessary for your external machine vision.

F.5.4.4. Detection program

This section describes a sample program for communicating with the external machine vision and acquiring the detection result.

MVSFIND1

```
 1: R[46:MVSFIND1 Status]=1
 2:
 3: !Sends the start command
 4: CALL    PKMVSENDMSG1                                  Sends the start command
 5:
 6: !Receives a response
 7: CALL    PKMVRCVMSG1                                   Receives a response
 8:
 9: CALL    PKMVDELPRSPM("Action ID"=1)
10: CALL PKMVSETPRSPM("Action ID"=1,"Stat Reg"=45,        Sets the analysis parameter
    3,6)
11: CALL PKMVPRSMSG1                                      Analyzes the response
12: IF R[45:PRSMSG Status]<>0,JMP LBL[900]
13: IF SR[6]<>SR[7],JMP LBL[910]
14:
15: !Receives the detection result
16: CALL PKMVRCVMSG1                                      Receives the detection result
17:
18: CALL PKMVDELPRSPM("Action ID"=1)
19: CALL PKMVSETPRSPM("Action ID"=1,"Stat Reg"=45,        Sets the analysis parameter
    1,61,1,62,2,63,2,64,2,65,2,66)
20: CALL PKMVPRSMSG1                                      Analyzes the detection result
21: IF R[45:PRSMSG Status]<>0,JMP LBL[900]
22:
23: IF R[61:Num. Found]<=0,JMP LBL[920]
```

```
   24:
   25: R[46:MVSFIND1 Status]=0
   26: END
   27:
   28: LBL[900:PKMVPRSMSG error]                    Error processing
   29: R[46:MVSFIND1 Status]=2
   30: JMP LBL[990]
   31:
   32: LBL[910:Response error]
   33: R[46:MVSFIND1 Status]=3
   34: JMP    LBL[990]
   35:
   36: LBL[920: Num. Found 0]
   37: R[46:MVSFIND1 Status]=4
   38:
   39: LBL[990]
[End]
```

The start command is sent to the external machine vision on line 4. The start command sent here was set up in "F.5.4.3. Initialization program(P.748)".

The response to the start command is received on line 7. PKMVRCVMSG receives one message separated with a line feed character every time it is executed. Even if the external machine vision has sent both a response and message and a detection result message, only the response is received in this processing.

The response analysis processing parameter is set on line 10. For details on the parameters, see "F.6.5.1. PKMVSETPRSPM.PC(P.759)".

The response is analyzed and stored in string R[6] on line 11.

Whether the response is normal is determined on line 13. Enter the response when the external machine vision normally receives the start command in string R[7] in advance.

The detection result is received on line 16.

The detection result analysis processing parameter is set on line 19.

The detection result is analyzed and stored in each register set in PKMVSETPRSPM on line 20.

The labels to which the processing jumps if an error occurs in each KAREL program are defined on lines 28 to 37. Specify error processing here if necessary.

NOTE

Some external machine vision systems may execute snap, detection, and transmission of detection results with different commands. In this case, add the processing to send the respective commands.

F.5.4.5. Latch program

This section describes a sample latch program.

LATCH

```
1: ! Check input of the encode acquisition signal
2: WAIT DI[R[40]]=ON OR R[47:End MVSACTION1]=1     Detect signal input
3: WAIT DI[R[40]]=OFF OR R[47:End MVSACTION1]=1
4:
5: !Obtain the encoder count
6: IF (R[47:End MVSACTION1]=0) THEN
7: Enc Cnt[1] R[60]                               Acquiring the encoder count
```

```
 8: R[48:LATCH succeeded]=1
 9: ENDIF
  [END]
```

The encoder count is stored in the register when a signal is input from the external machine vision system.

Wait until input of the encoder acquisition signal is detected on lines 2 or 3. Enter the DI/RI Num. to get the Enc Cnt to be used for R[40] in advance. When RI is used for DI/RI Num. to get the Enc Cnt, change the condition for this wait command to RI.
For the case when an action program ends without signal input, wait will also end when R[47] is 1, which indicates that the action program ends.

Store the encoder count in R[60] on line 7.
Store 1 in R[48] on line 8 to indicate that latch is successful.

F.5.4.6. Coordinate conversion parameter calculation program

This section describes a sample program for calculating coordinate conversion parameters of the external machine vision system.

CALC_EXTPM

```
 1: CALL PKWCEND
 2:
 3: CALL PKCSGETID('CONV1',"CStn ID Reg"=11)
 4:
 5: PAUSE
 6: !Enter x0, y0, x1, and y1 and continue                    Specify the detection position
 7: CALL PKMVSCLPUTP1("Action ID"=1,"CStn ID"=R[11:CONV1 ID],
          "X0"=R[51:x0],"Y0"=R[52:y0])
 8:
 9: CALL PKMVSCLPUTP2("Action ID"=1,"CStn ID"=R[11:CONV1 ID],
          "X1"=R[53:x1],"Y1"=R[54:y1])
10:
11: PAUSE
12: !Move the conveyer, enter x2 and y2, and continue        Specify the position after
                                                             movement
13: CALL PKMVSCLPUTP3("Action ID"=1,"CStn ID"=R[11:CONV1 ID],
            "X2"=R[55:x2],"Y2"=R[56:y2])
14:
15: PAUSE
16: ! Touch-up target 1 and continue                         Touch-up target 1
17: CALL PKMVSCLSETP1("Action ID"=1,"CStn ID"=R[11:CONV1 ID])
18:
19: PAUSE
20: ! Touch-up target 2 and continue                         Touch-up target 2
21: CALL PKMVSCLSETP2("Action ID"=1,"CStn ID"=R[11:CONV1 ID])
22:
23: CALL PKMVSCLCALPM("Action ID"=1)                         Calculate the parameter
  [END]
```

Calculate the coordinate conversion parameter of the external machine vision. The coordinate conversion parameter is used to convert the detection position in the external machine vision system into the coordinates of a position in the tracking frame.
This program assumes that the target detection position is manually stored in the register.

The sensors are shut down on line 1 so that no actions are executed during teaching.

Give the detection position of target 1, P0 (x0, y0), and that of target 2, P1 (x1, y1), to the program on lines 7 and 9. The detection positions obtained before this line are executed and stored in the register.

After that, move the conveyor in the range where the target does not move out of the external machine vision system's field of view, store detection position P2 (x2, y2) of target 2 after movement in the register, and then execute line 13.

Move the conveyor to the position where target touch-up is possible, execute line 17 with target 1 touched-up and line 21 with target 2 touched-up, and teach the position on the target tracking frame.

Calculate the coordinate conversion parameter of the external machine vision system based on the position information specified on line 23.

F.5.4.7. Reference position setting program

This section describes a sample TP program for setting the reference position.

SET_REFPOS

```
  1: CALL PKWCEND
  2:
  3: CALL MVSINIT1                                                    Initialization
  4:
  5: CALL MVSFIND1                                                    Acquires the
                                                                      detection result
  6: IF R[46:MVSFIND1 Status]<>0,JMP LBL[900]
  7: IF R[61:Num. Found]<=0,JMP LBL[900]
  8:
  9: Enc Cnt [1] R[60]
 10: Trigger setting tracking[1] R[60]                                Sets a trigger
 11:
 12: ! Stores the reference position in the Position Register         Stored in the Position
                                                                      Register
 13: PR[27,1:Detection Pos]=R[63:x]
 14: PR[27,2:Detection Pos]=R[64:y]
 15: PR[27,3:Detection Pos]=0
 16: PR[27,4:Detection Pos]=0
 17: PR[27,5:Detection Pos]=0
 18: PR[27,6:Detection Pos]=R[65:r]
 19:
 20: ! Converts the reference position into coordinates               Converts the
                                                                      reference position
                                                                      into coordinates
 21: CALL PKMVCNVPX2MM("Action ID"=1,"Fnd Pos PR"=27,"Cnv Pos PR"=26)
 22:
 23: END
 24: LBL[900]
 25: UALM[1]
  [END]
```

This program acquires the detection position from the external machine vision, converts it into coordinates, and stores them as the reference position in the Position Register.

After executing this program, teach the position to the robot with a pick operation program.

The sensors are shut down on line 1 so that no actions are executed during teaching.

The TP program is called on line 3 to initialize connection with the external machine vision, etc. For details, see "F.5.4.3. Initialization program(P.748)" above.

The TP program for acquiring the detection result from the external machine vision is called on line 5. For details, see "F.5.4.4. Detection program(P.749)".
If Num. Found is 0 on line 7, the processing jumps to line 24, generating a user alarm.

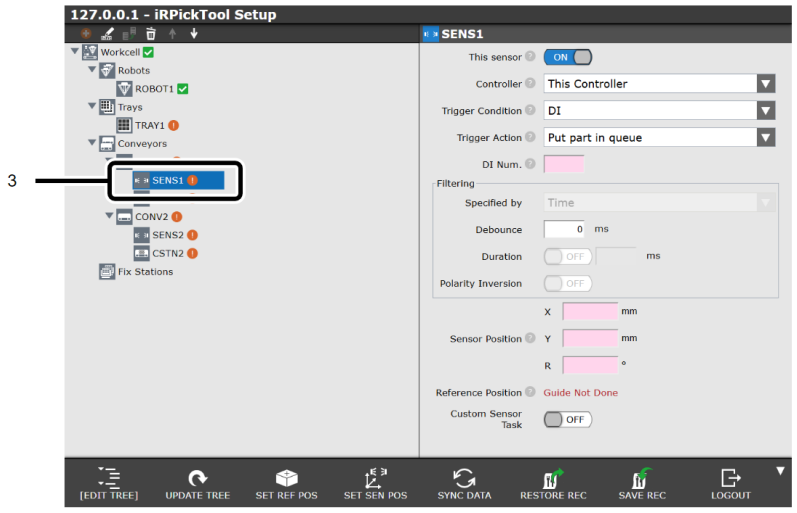
The encoder count is acquired and a trigger is set to the tracking schedule on lines 9 and 10. When using an encoder number other than 1 and a tracking schedule number other than 1, change these arguments to the numbers to be used.

Acquired results are stored in Position Registers on lines 13 to 18.
The detection position is converted into coordinates and stored as the reference position in PR[26] on line 21.

F.5.5. Setting up an infeed conveyor sensor

Set up an infeed conveyor sensor.

1. **Open the robot homepage.**
2. **Select [iRVision/iRPickTool] in the menu area and click [iRPickTool System Setup] in the content display area.**
3. **Select [SENS1].**

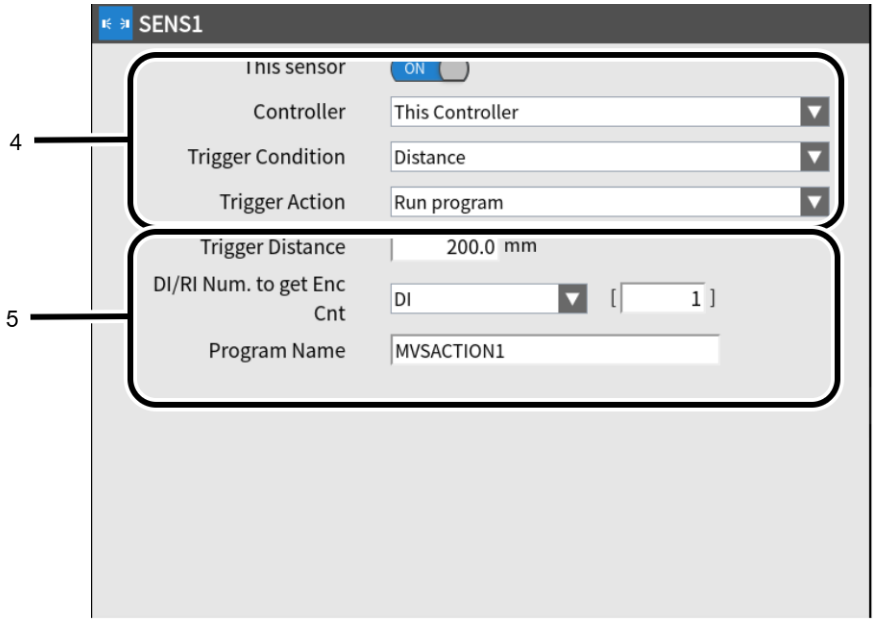


4. **Set the following items.**

   [This Sensor]: [Enable]
   [Controller]: [This Controller]
   [Trigger Condition]: [Distance]
   [Action]: [Run program]
   When [Trigger Condition] and [Action] are changed, setting items change accordingly.

5. **Set the following items.**

   [Trigger Distance]: Enter how many millimeters the conveyor advances before the action program is executed
   [DI/RI Num. to get Enc Cnt]: Enter the DI or RI number at which the signal to acquire the encoder count is input
   [Program Name]: [MVSACTION1]



CAUTION

Although the sensors enter teaching complete status after this item is set, the setup is not yet completed. Do not start the sensors until the procedures described in "F.5.7. Calculating the coordinate conversion parameter(P.754) " and "F.5.8. Setting up a reference position for the infeed conveyor, and teaching a robot position(P.755) " later are complete.

F.5.6. Position register operation settings

1. **Press the [MENU] key on the robot controller's teach pendant.**
2. **Select [SYSTEM] →[Configuration] from the menu.**
3. **Place the cursor over the [Program Command] line, then press F3 [DETAIL].**
4. **Enable [Position Register Operation without Motion Group].**

CAUTION

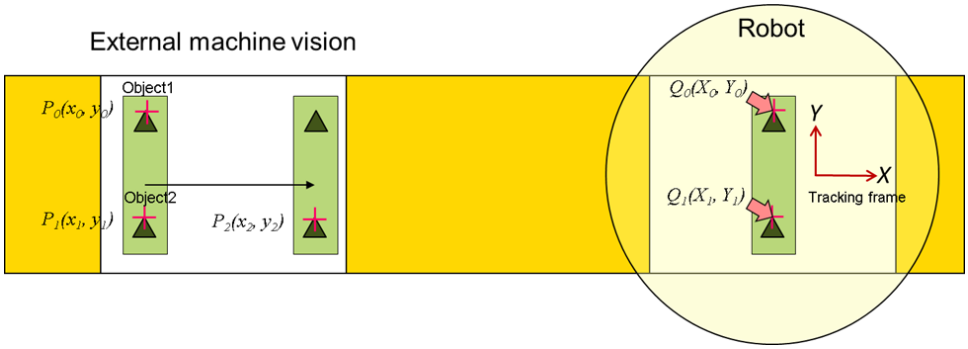
When [Position Register Operation without Motion Group] is enabled, you can change the position register value from the TP program for which motion group is not set. This setting is required because position register operation is performed in the action program. To change the position register value from the TP program for which motion groups other than an action program are not set, check that the change will not affect the robot motion.

F.5.7. Calculating the coordinate conversion parameter

Calculate the parameter to convert the detection position in the external machine vision into the coordinates of a position in the tracking frame. Execute the coordinate conversion parameter calculation program with the procedure below. For the coordinates conversion parameter calculation program, see "F.6.6. Programs related to coordinate conversion parameter calculation(P.760)".

1. **Switch to the tool frame set to the tip of the touch-up pin.**
2. **Fix two targets so that they are positioned perpendicular to the flow direction of the conveyor in the external machine vision's field of view.**
3. **Execute the coordinate conversion parameter calculation program with the procedure below.**
    1. Record the detection position of target 1, P0 (x0,y0), and that of target 2, P1 (x1,y1), in the external machine vision.

2 Specify P0 in the argument of PKMVSCLPUTP1 (KAREL program to manually enter the detection position) and execute the program.
3 Specify P1 in the argument of PKMVSCLPUTP2 without moving the conveyor and execute the program.
4 Move the conveyor in the flow direction in the range where the target does not move out of the external machine vision's field of view, and record the detection position P2 (x2,y2) of target 2 after movement.
5 Specify P2 in the argument of PKMVSCLPUTP3.
6 Move the conveyor to the position at which the target can be touched up and touch up the detection position of target 1, Q0 (X0, Y0).
7 Execute PKMVSCLSETP1 (KAREL program to acquire the robot position).
8 Touch up the detection position of target 2, Q1 (X1, Y1), without moving the conveyor.
9 Execute PKMVSCLSETP2.
10 Execute PKMVSCLCALPM.



NOTE

1. To improve the accuracy, calculate the coordinate conversion parameter using the target that has a height close to the part pick position.
2. If the position at which the robot touches up is different from the detection position in the external machine vision, the coordinate conversion parameter is not accurately calculated. Use targets whose detection positions can be easily determined.

F.5.8. Setting up a reference position for the infeed conveyor, and teaching a robot position

This procedure is for setting the reference position and teaching positions to the robot. For the reference position setting program, see "F.5.4.7. Reference position setting program(P.752)".

1 Fix a part in the external machine vision's field of view.
2 Execute the reference position setting program. The following setup is performed.
- The detection result is acquired from the external machine vision and stored in the Position Register.
- The detection result is converted into coordinates using PKMVCNVPX2MM.
- Trigger setting is executed and a trigger value is set to the tracking schedule.

3 Move the conveyor, and then when the part enters the robot's moving range, stop it.
4 Move the robot in jog mode to the part pick position.
5 Display a pick operation program and place the cursor over the line where the pick position is taught.
6 While holding down the [SHIFT] key, press F5 [TOUCHUP].

F.6. KAREL program

This section describes KAREL programs used in the iRPickTool Vision I/F add-on.

NOTE
The alarm specific to this function that occurs in KAREL programs in the iRPickTool Vision I/F add-on are displayed as IRPK-201. The cause of this alarm is displayed in the alarm message text. Refer to the KAREL program where the alarm occurred and the alarm message text to solve the problem.

F.6.1. Programs related to communication

F.6.1.1. PKMVCONNECT.PC

Connect to the external machine vision system by using the specified client tag. Communication is disconnected temporarily when this program is executed. Therefore, PKMVDISCO does not need to be executed in advance even during reconnection.

Argument 1:	Specify the number of the client tag for which the external machine vision is set.

Example:

```
CALL PKMVCONNECT("Client ID"=2) ;
```

F.6.1.2. PKMVDISCO.PC

Disconnect the communication with the external machine vision system connected by using the specified client tag. An alarm occurs when this KAREL program is executed by using a client tag that is not connected yet.

Argument 1:	Specify the number of the client tag for which the external machine vision is set.

Example:

```
CALL PKMVDISCO("Client ID"=2) ;
```

F.6.2. Programs related to initialization

F.6.2.1. PKMVINIACT.PC

Make initial settings for the specified action. Execute the program for communicating with the external machine vision system and acquiring the detection result first. In the subsequent KAREL programs, specify the action ID set in this program before executing them.

Argument 1:	Specify the ID of the action to be set from 1 to 4. Specify different action IDs for each external machine vision system for communication.
Argument 2:	Specify the number of the client tag for which the external machine vision is set. A client ID that is already set for another action ID cannot be specified.
Argument 3:	Specify a character that separates each data item in the input message.
Argument 4:	Specify a line feed character to be appended to the end of the input message. Select a line feed character from among the following: “LF” "CR” "CRLF” “” (none)
Argument 5:	Specify the maximum number of data items in the input message. When multiple input messages are to be sent, specify the biggest number of data items. When the number of data items set in

PKMVSETSNDPM exceeds this value, the warning “IRPK-201 max_data_in is insufficient” occurs. The maximum settable number of data items is 500.
For example, when the following input message is set, specify 9.
"SUCCESS,FIND,0,0,0,0,0,0,0<LF>"

Argument 6: Specify a character that separates each data item in the output message. Do not specify any of the characters that are contained in the data.

Argument 7: Specify a line feed character to be appended to the end of the output message.
Select a line feed character from among the following:
"LF”
"CR”
"CRLF”

Argument 8: Specify the maximum number of data items in the output message. When multiple output messages are to be received, specify the biggest number of data items. When the number of data items received when PKMVRCVMSG is executed exceeds this value, the warning “IRPK-201 max_data_out is insufficient” occurs. The maximum settable number of data items is 500.

Example:

```
CALL PKMVINIACT("Action ID"=1,"Client ID"=2,"In Msg Delimiter"=',',"In Msg Newline char"='LF',"In Msg Max num data"=1,"Out Msg Delimiter"=',',"Out Msg Newline char"='LF',"Out Msg Max num data"=11) ;
```

F.6.3. Programs related to input message transmission

F.6.3.1. PKMVSETSNDPM.PC

Set the data in the input message to be sent with PKMVSENDMSG. To set more than 28 data items, which is the number of arguments for specifying data, continue to execute this program, which will set the data subsequent to the previously set data. When more data than the maximum number of data items in an input message set in PKMVINIACT is set, the warning “IRPK-201 max_data_in is insufficient” occurs. The data set in this program will not be deleted even when PKMVSENDMSG is executed. When the same input message is to be sent, it is okay to execute this program only once at the beginning. To set different input messages, execute this program after executing PKMVDELSNDPM and deleting the input message that has been set.

Argument 1:	Specify an action ID.
Argument 2:	Specify a register number to store the status of PKMVSENDMSG.
Arguments 3 to 30: (Arguments 4 to 30 can be omitted)	Specify the data in the input message. It can be specified by using integers, real numbers, and strings. When a real number is specified, up to the third decimal place is sent. When a string is specified, the maximum settable string length is 34.

Example:

When the following program is executed, “RUNFIND” and “1.234” are set as the data of the input message.

```
CALL PKMVSETSNDPM("Action ID"=1,"Stat Reg"=10,'RUNFIND',1.2345) ;
```

F.6.3.2. PKMVDELSNDPM.PC

Delete the input message to be sent with PKMVSENDMSG. To set different input messages, execute this program before executing PKMVSETSNDPM.

Argument 1: Specify an action ID.

Example:

```
CALL PKMVDELSNDPM("Action ID"=1) ;
```

F.6.3.3. PKMVSENDMSG1~4.PC

Send the input message to the external machine vision system. The number at the end of the program name corresponds to the action ID.

Separate the data that has been set with PKMVSETSNDPM with a separator and append a line feed character to the end before sending. Use PKMVINIACT to set the separator and the line feed character.

One of the following values is stored in the register as the status after execution.

0:	Success
12:	Communication error

There is no argument.

Example:

```
CALL PKMVSENDMSG1 ;
```

F.6.4. Programs related to output message reception

F.6.4.1. PKMVSETRCVPM.PC

Set the timeout time for PKMVRCVMSG. The data set in this program will not be deleted even when PKMVRCVMSG is executed. When the timeout time is not to be changed, it is okay to execute this program only once at the beginning.

Argument 1:	Specify an action ID.
Argument 2:	Specify the timeout time for output message reception. The units are [ms].
Argument 3:	Specify a register number to store the status of PKMVRCVMSG.

Example:

```
CALL PKMVSETRCVPM("Action ID"=1,"Timeout(ms)"=5000,"Stat Reg"=11) ;
```

F.6.4.2. PKMVSENDMSG1~4.PC

Receive the output message sent from the external machine vision system. The number at the end of the program name corresponds to the action ID.

PKMVRCVMSG receives one output message separated by a line feed character when this program is executed. Set the line feed character with PKMVINIACT. At execution, the previously received message will be deleted. To store the received output message in a register, execute PKMVPRSMSG. When more data than the maximum number of data items in an output message set in PKMVINIACT is received, the warning “IRPK-201 max_data_out is insufficient” occurs. When an output message cannot be received within the timeout time after this program is executed, the warning “IRPK-201 Receive message timeout” occurs. When an invalid line feed character or data with 255 or more characters is received, the warning “IRPK-201 Receive data is invalid” occurs.

One of the following values is stored as the status after execution.

0:	Success
11:	Timeout occurred
12:	Communication error

There is no argument.

Example:

```
CALL PKMVRCVMSG1 ;
```

F.6.5. Programs related to output message analysis

F.6.5.1. PKMVSETPRSPM.PC

Set the parameter for analysis with PKMVPRSMSG. For the parameter, set a combination of the data type in the output message to be received and the register number in which to store it. To set more than 14 parameters, which is the number of arguments for specifying data, continue to execute this program, which will set the parameters subsequent to the previously set data. The maximum settable number of data items is 500 (the maximum settable number of data items is different from the maximum number of data items in the output message that has been set with PKMVINIACT.). The data set in this program will not be deleted even when PKMVPRSMSG is executed. To set different parameters, execute this program after executing PKMVDELPRSPM and deleting the parameters that have been set.

Argument 1:	Specify an action ID.
Argument 2:	Specify the register number in which to store the status of PKMVPRSMSG.
Arguments 3, 5, 7, ..., 29: (optional)	Specify the data type. Specify it from the following values: This argument and the register number, which will be specified next, will be a combination of parameters for a single data item. 1: Integer 2: Real number 3: String
Arguments 4, 6, 8, ..., 30: (optional)	Specify the register number in which to store data. When the data type specified in the previous argument is an integer or real number, data will be stored in a register. When it is a string, data will be stored in a string register.

Example:

Set the parameters for storing the data in the output message “SUCCESS (string), Model ID (Integer), x (Real number) <LF>” in String R[5], R[10], and R[11].

```
CALL PKMVSETPRSPM("Action ID"=1,"Stat Reg"=12,3,5,1,10,2,11) ;
```

F.6.5.2. PKMVDELPRSPM.PC

Delete the parameters that have been set with PKMVSETPRSPM. To set different parameters for received messages, execute this program before executing PKMVSETPRSPM.

Argument 1:	Specify an action ID.

Example:

```
CALL PKMVDELPRSPM("Action ID"=1) ;
```

F.6.5.3. PKMVSENDMSG1~4.PC

Analyze the output message received with PKMVRCVMSG and store the data in a register. The number at the end of the program name corresponds to the action ID.

Every time it is executed, the data in the output message that has been received with PKMVRCVMSG is retrieved in sequence from the beginning, and their types are converted in the order of the parameters set with PKMVSETPRSPM and stored in the register. When the set data type is different from the analyzed data type, the warning “IRPK-201 Data type is invalid” occurs. When empty data is analyzed, the warning “IRPK-201 Receive data is invalid” occurs.

⚠CAUTION

When the number of analyzed data items in an output message is smaller than the number of parameters that have been set with PKMVSETPRSPM, analysis will be performed by using the parameters subsequent to the previously used ones the next time PKMVRCVMSG is executed. Therefore, if analysis ends part way through due to an error, the parameters to be used for analysis will shift and an error may occur. Basically, before executing PKMVPRSMSG, execute PKMVDELPRSPM and PKMVSETPRSPM to set the parameters again.

One of the following values is stored as the status after execution.

0:	Success
1:	The number of analyzed data items in the output message is greater than the number of parameters set with PKMVSETPRSPM
2:	The number of analyzed data items in the output message is smaller than the number of parameters set with PKMVSETPRSPM
3:	No data in the output message has been analyzed
4:	NG

There is no argument.

Example:

```
CALL PKMVPRSMSG1 ;
```

F.6.6. Programs related to coordinate conversion parameter calculation

F.6.6.1. PKMVSCLPUTP1.PC

Input the detection position for the first target.

Argument 1:	Specify an action ID.
Argument 2:	Specify the ID of the conveyor station.
Argument 3:	Specify a value for x0.
Argument 4:	Specify a value for y0.

Example:

```
CALL PKMVSCLPUTP1("Action ID"=1,"CStn ID"=R[6:CSTN1 ID], "X0"=R[51:x0],"Y0"=R[52:y0] ;
```

F.6.6.2. PKMVSCLPUTP2.PC

Input the detection position for the second target. Execute this program without moving the conveyor after PKMVSCLPUTP1 is executed.

Argument 1:	Specify an action ID.
Argument 2:	Specify the ID of the conveyor station.
Argument 3:	Specify a value for x1.
Argument 4:	Specify a value for y1.

Example:

```
CALL PKMVSCLPUTP2("Action ID"=1,"CStn ID"=R[6:CSTN1 ID], "X1"=R[53:x1],"Y1"=R[54:y1]) ;
```

F.6.6.3. PKMVSCLPUTP3.PC

Input the detection position for the second target after moving the conveyor. After executing PKMVSCLPUTP2, move the conveyor in the range where the target does not move out of the external machine vision system's field of view, specify the detection position of the second target with an argument, and then execute this program.

Argument 1:	Specify an action ID.
Argument 2:	Specify the ID of the conveyor station.
Argument 3:	Specify a value for x2.
Argument 4:	Specify a value for y2.

Example:

```
CALL PKMVSCLPUTP3("Action ID"=1,"CStn ID"=R[6:CSTN1 ID], "X2"=R[55:x2],"Y2"=R[56:y2]) ;
```

F.6.6.4. PKMVSCLSETP1.PC

Teach the position of the first target on the tracking frame. Execute it with the first target's detection position touched-up.

Argument 1:	Specify an action ID.
Argument 2:	Specify the ID of the conveyor station.

Example:

```
CALL PKMVSCLSETP1("Action ID"=1,"CStn ID"=R[6:CSTN1 ID]) ;
```

F.6.6.5. PKMVSCLSETP2PC

Teach the position of the second target on the tracking frame. Execute this program with the second target touched-up, without moving the conveyor after PKMVSCLSETP1 is executed.

Argument 1:	Specify an action ID.
Argument 2:	Specify the ID of the conveyor station.

Example:

```
CALL PKMVSCLSETP2("Action ID"=1,"CStn ID"=R[6:CSTN1 ID]) ;
```

F.6.6.6. PKMVSCLCALPM.PC

Calculate the coordinate conversion parameter of the external machine vision. Execute this program after executing all the programs related to coordinate conversion parameter calculation.

Argument 1:	Specify an action ID.

Example:

```
CALL PKMVSCLCALPM("Action ID"=1) ;
```

F.6.7. Programs related to detection position conversion

F.6.7.1. PKMVCNVPX2MM.PC

Convert the detection position in the external machine vision system into the coordinates of a position in the tracking frame. It can be executed after the coordinate conversion parameter calculation ends.

Argument 1:	Specify an action ID.
Argument 2:	Specify the number of the Position Register that contains the detection position in the external machine vision system.
Argument 3:	Specify the number of the Position Register that contains the position after coordinate conversion. When the same value as that of argument 2 is specified, the specified detection position will be overwritten with the value after coordinate conversion.

Example:

```
CALL PKMVCNVPX2MM("Action ID"=1,"Fnd Pos PR"=2,"Cnv Pos PR"=1) ;
```

F.6.7.2. PKMVCALOFS2D.PC

This program calculates offset based on the detection position and reference position. Specify the position converted into coordinates using PKMVCNVPX2MM. Offset can be calculated only in the iRPickTool Vision I/F add-on.

Argument 1:	Specify the number of the Position Register that contains the detection position.
Argument 2:	Specify the number of the Position Register that contains the reference position.

Argument 3:	Specify the number of the Position Register to store offset.

Example:

```
CALL PKMVCALOFS2D("Fnd Pos PR"=2,"Ref Pos PR"=1,"Offset PR"=3) ;
```

F.6.7.3. PKMVSETVR.PC

Store the specified value in the Vision Register. This is used to input a workpiece into the queue by using PKCSPUTQUE.

NOTE

Using PKCSPUTQUE2 allows you to input a workpiece into the queue without using a Vision Register. For details, refer to "I.5.8.5.7. PKCSPUTQUE2.PC(P.354) ".

Argument 1:	Specify the number of the Vision Register that contains values.
Argument 2:	Specify the number of the Position Register that contains the detection position.
Argument 3:	Specify the number of the Position Register that contains the offset.
Argument 4:	Specify the number of the register that contains the model ID.
Argument 5:	Specify the number of the register that contains the encoder count.
Arguments 6–15: (optional)	Specify the number of the register that contains measurement values 1 to 10. When the measurement values are omitted, 0.0 is stored.

Example:

Set the following values for Vision Register 1.

Detection position:	PR[2]
Offset:	PR[3]
Model ID:	R[31]
Encoder count:	R[28]
Measurement value 1:	R[38]

```
CALL PKMVSETVR("VR"=1,"Fnd Pos PR"=2,"Offset PR"=3,"Model ID Reg"=31,"Enc Count Reg"=28,38) ;
```

F.6.8. Other programs

F.6.8.1. PKMVWAITTASK.PC

Wait until the specified task ends. This is used to wait for the program running in the background to end. When the specified task does not exist, this program will end immediately.

Argument 1:	Name of the task the end of which is being waited for.

Example:

```
CALL PKMVWAITTASK("Task Name"='LATCH') ;
```

F.7. Setting example appropriate to the usage

F.7.1. Detecting multiple parts at one snap

This section describes how to start up the external machine vision system that sends up to two detection results at one detection as in the following message format.

Message formats

Input message (Robot controller → External machine vision)	RUNFIND(string)<CR>
Response to an input message (External machine vision → Robot controller)	Normal : OK(string)<CR> Abnormal : ER(string)<CR>
Output message (External machine vision → Robot controller)	Num. Found(integer number),Model ID 1(integer number),X1(real number),Y1(real number),R1(real number),Measurement Value 1.1(real number) ,Model ID 2(integer number),X2(real number),Y2(real number),R2(real number),Measurement Value 2.1(real number)<CR>

In this message format, even if Num. Found is 1, the output message must be sent without omitting the second detection result data. Therefore, the following processing is required if Num. Found is 1.

- Store 1 in Num. Found
- Store 0 as data after Model ID 2

Although the general startup procedure is the same as the one described in "F.5. Startup procedure(P.739)", change the action program and detection program to the following ones.
Additionally, the following registers will be used. Change the register numbers as needed.

Defining registers

Table F.7.1-1 Additional registers to be used

R[72:Model ID]=0	Model ID (detection result 2)
R[73:x]=0	x (detection result 2)
R[74:y]=0	y (detection result 2)
R[75:r]=0	r (detection result 2)
R[76:Measurement Value 1]=0	Measurement Value 1 (detection result 2)

Action program

MVSACTION1

```
 1: R[47:End MVSACTION1]=0 ;
 2: ;
 3: !Uses an accurate trigger ;
 4: Accurate trigger tracking[1] ;
 5: ;
 6: R[48:LATCH succeeded]=0 ;
 7: Run LATCH ;
 8: ;
 9: CALL MVSFIND1 ;
10: IF R[46:MVSFIND1 Status]<>0,JMP LBL[990] ;
11: ;
12: IF R[48:LATCH succeeded]=0,JMP LBL[900] ;
13: ;
```

```
14: !Stores Detection result 1 in the Position Register ;
15: PR[27,1:Detection Pos]=R[63:x1] ;
16: PR[27,2:Detection Pos]=R[64:y1] ;
17: PR[27,3:Detection Pos]=0 ;
18: PR[27,4:Detection Pos]=0 ;
19: PR[27,5:Detection Pos]=0 ;
20: PR[27,6:Detection Pos]=R[65:r1] ;
21: ;
22: !Converts the detection position into coordinates ;
23: CALL PKMVCNVPX2MM("Action ID"=1,"Fnd Pos PR"=27,"Cnv Pos PR"=27) ;
24: !Calculates offset ;
25: CALL PKMVCALOFS2D("Fnd Pos PR"=27,"Ref Pos PR"=26,"Offset PR"=28) ;
26: ;
27: !Inputs the detection result into the queue ;
28: CALL PKCSPUTQUE2("CStn ID"=R[11],"Enc Cnt”=R[60],
   "Model ID R"=R[62],"Fnd Pos PR"=27,"Offset PR"=28,"Meas. R"=66,
   "Meas.Cnt"=1,"OverlapStat Reg"=49)
29: ;
30: IF R[61:Num. Found]<=2,JMP LBL[990] ;
31: !Stores Detection result 2 in the Position Register ;
32: PR[27,1:Detection Pos]=R[73:x2] ;
33: PR[27,2:Detection Pos]=R[74:y2] ;
34: PR[27,3:Detection Pos]=0 ;
35: PR[27,4:Detection Pos]=0 ;
36: PR[27,5:Detection Pos]=0 ;
37: PR[27,6:Detection Pos]=R[75:r2]
38: ;
39: !Converts the detection position into coordinates and calculates offset;
40: CALL PKMVCNVPX2MM("Action ID"=1,"Fnd Pos PR"=27,"Cnv Pos PR"=27)
41: !Calculates offset ;
42: CALL PKMVCALOFS2D("Fnd Pos PR"=27,"Ref Pos PR"=26,"Offset PR"=28)
43: ;
44: !Inputs the detection result into the queue ;
45: CALL PKCSPUTQUE2("CStn ID"=R[11],"Enc Cnt"=R[60],
   "Model ID R"=R[62],"Fnd Pos PR"=27,"Offset PR"=28,"Meas. R"=76,
   "Meas.Cnt"=1,"OverlapStat Reg"=49)
46: JMP LBL[990] ;
47: ;
48: LBL[900:Latch error] ;
49: R[46:MVSFIND1 Status]=5 ;
50: ;
51: LBL[990] ;
52: R[47:End MVSACTION1]=1 ;
53: CALL PKMVWAITTASK("Task Name"='LATCH') ;
[End]
```

Inputs Detection result 1 into the queue

Ends when Num. Found is 1

Inputs Detection result 2 into the queue

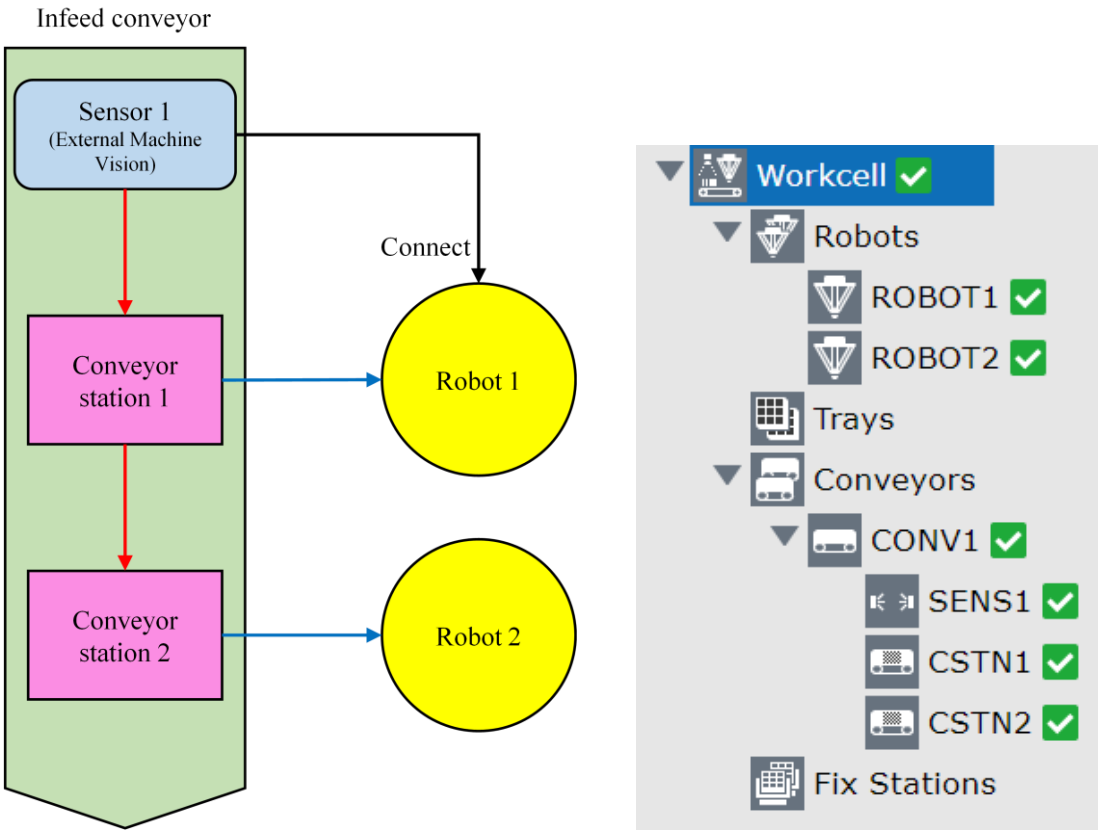
Detection program

MVSFIND1

```
 1: R[46:MVSFIND1 Status]=1 ;
 2: ;
 3: !Sends the start command ;
 4: CALL PKMVSENDMSG1 ;
 5: ;
 6: !Receives a response ;
 7: CALL    PKMVRCVMSG1 ;
 8: ;
 9: CALL    PKMVDELPRSPM("Action ID"=1) ;
10: CALL PKMVSETPRSPM("Action ID"=1,"Stat Reg"=45,3,6) ;
11: CALL PKMVPRSMSG1 ;
12: IF R[45:PRSMSG Status]<>0,JMP LBL[900] ;
13: IF SR[6]<>SR[7],JMP LBL[910] ;
14: ;
15: !Receives the detection result ;
16: CALL PKMVRCVMSG1 ;
17: ;
18: !Analyzes the first detection result ;                    Analyzes the
                                                             first detection
                                                             result
19: CALL PKMVDELPRSPM("Action ID"=1) ;
20: CALL PKMVSETPRSPM("Action ID"=1,"Stat Reg"=45,1,61,1,62,2,63,2,64,2,65,2,66) ;
21: CALL PKMVPRSMSG1 ;
22: ;
23: !Analyzes the second detection result ;                   Analyzes the
                                                             second
                                                             detection result
24: CALL PKMVDELPRSPM("Action ID"=1) ;
25: CALL PKMVSETPRSPM("Action ID"=1,"Stat Reg"=45,1,72,2,73,2,74,2,75,2,76) ;
26: CALL PKMVPRSMSG1 ;
27: IF R[45:PRSMSG Status]<>0,JMP LBL[900] ;
28: ;
29: IF R[61:Num. Found]<=0,JMP LBL[920] ;
30: ;
31: R[46:MVSFIND1 Status]=0 ;
32: END ;
33: ;
34: LBL[900:PKMVPRSMSG error] ;
35: R[46:MVSFIND1 Status]=2 ;
36: JMP LBL[990] ;
37: ;
38: LBL[910:Response error] ;
39: R[46:MVSFIND1 Status]=3 ;
40: JMP LBL[990] ;
41: ;
42: LBL[920:Num. Found 0] ;
43: R[46:MVSFIND1 Status]=4 ;
44: ;
45: LBL[990] ;
[End]
```

F.7.2. Connecting multiple robots

The following describes the configuration method when picking parts detected by an external machine vision system by multiple robots, as shown in the figure below.



Connect the external machine vision system to the most upstream robot controller.

The following software options are required only for robot controllers that communicate with external machine vision systems. In the example in this section, the following software options are required only for the most upstream robot controller.

- R648 (user socket message function)
- S521 (iRPickTool Vision I/F add-on)

The following software options are required for all robot controllers.
J944 (iRPickTool Basic) or J945 (iRPickTool)

Perform the startup by following the procedures from "F.5.1. External machine vision system connection and settings(P.740)" to "F.5.7. Calculating the coordinate conversion parameter(P.754)" on the most upstream robot. For all other robots, follow the standard iRPickTool startup procedure.
Next, set the reference position and teach the positions to the robots according to the following procedures.

1. **Fix a part in the external machine vision's field of view.**
2. **Execute the reference position setting program on the most upstream robot. The following setup is performed.**
   1. The detection result is acquired from the external machine vision and stored in the Position Register.
   2. The detection result is converted into coordinates using PKMVCNVPX2MM.
   3. Trigger setting is executed and a trigger value is set to the tracking schedule.
3. **Execute the trigger setting program on all downstream robots.**
4. **Move the conveyor, and then when the part enters the moving range of the most upstream robot, stop it.**

5 **Move the robot in jog mode to the part pick position.**
6 **Display a pick operation program and place the cursor over the line where the pick position is taught.**
7 **While holding down the [SHIFT] key, press F5 [TOUCHUP].**
8 **Move the robot to a position where it will not interfere with the conveyor during its movement. Then, start the conveyor and stop it once the part enters the operating range of the next robot.**
9 **Move the robot in jog mode to the part pick position.**
10 **Display a pick operation program and place the cursor over the line where the pick position is taught.**
11 **While holding down the [SHIFT] key, press F5 [TOUCHUP].**
12 **Perform steps 8 through 11 on all robots, in order from upstream to downstream.**

F.7.2.1. Program for setting a trigger

```
1: Enc Cnt[1] R[60]
2: Trigger setting tracking[1] R[60]
```

The encoder count is acquired on line 1 and a trigger is set to the tracking schedule on line 2. When using an encoder number other than 1 and a tracking schedule number other than 1, change the corresponding argument according to the number to be used.

F.8. Troubleshooting

This section describes common issues in a system using the iRPickTool Vision I/F add-on and countermeasures for them.

F.8.1. The external machine vision cannot be connected

Cause 1: The client tag settings are incorrect.
Remedy 1: Confirm that the IP address and port number of the external machine vision are properly set to the client tag, referring to "F.5.2.2. Setting up a client tag(P.741) ". Confirm that [Current statue] in the client tag is [Define].

Cause 2: The Ethernet port is not set up.
Remedy 2: Confirm that the TCP/IP settings for the Ethernet port connected to the external machine vision are configured.

F.8.2. Snapping or detection is not executed in the external machine vision system

Cause 1: The input message (start command) is incorrect.
Remedy 1: Check that the input message specified with PKMVSETSENDPM is the same as the start command of the external machine vision system. Also, check that the separator and line feed character in the input message specified with PKMVINIACT can be recognized by the external machine vision system.

Cause 2: The external machine vision system is not ready to receive commands.
Remedy 2: The external machine vision system is not ready to receive commands. For the setting procedures, see the operator's manual from the relevant vision manufacturer.

F.8.3. Timeout occurs in PKMVRCVMSG.

Cause 1: The line feed character in the output message is incorrect.
Remedy 1: Check that the line feed character used in the output message matches that of the output message specified with PKMVINIACT.

Cause 2:	The timeout time is too short.
Remedy 2:	Set the timeout time to the one longer than the detection time of the external machine vision system.
Cause 3:	The input message for sending detection results has not been sent.
Remedy 3:	Some external machine vision systems may execute snap, detection, and transmission of detection results with different commands. Check that detection results can be sent by using the input message sent from the robot controller.

F.8.4. PKMVPRSMSG fails

Cause 1:	The analysis parameter in the output message is incorrect.
Remedy 1:	Check that the parameter set with PKMVSETPRSPM is the same as the format of the output message to be sent from the external machine vision system. Also, check that the separator in the output message specified with PKMVINIACT is correct.
Cause 2:	The analysis parameter has shifted.
Remedy 2:	When the number of analyzed data items in an output message is smaller than the number of parameters that have been set with PKMVSETPRSPM, analysis will be performed by using the parameters subsequent to the previously used ones the next time PKMVRCVMSG is executed. Therefore, if analysis ends part way through due to an error, the parameters to be used for analysis will shift and an error may occur. Basically, before executing PKMVPRSMSG, execute PKMVDELPRSPM and PKMVSETPRSPM to set the parameters again.
Cause 3:	The received output message is different from the output message for which parameters have been set.
Remedy 3:	The robot controller may receive output messages other than those for detection results because, for example, some external machine vision systems return responses as output messages when it receives a command. Check the output messages sent from the external machine vision system.

F.8.5. Latch fails

Cause 1:	The DI/RI Num. to get Enc Cnt is incorrect.
Remedy 1:	Check that the number of the encoder acquisition signal and the number of the signal input that is detected in the latch program are correct.
Cause 2:	The encoder acquisition signal is not output.
Remedy 2:	Check that the encoder acquisition signal is output from the external machine vision system during snapping. Also, check that the wiring between the external machine vision system and the robot controller is correct.
Cause 3:	The time until the encoder acquisition signal falls is too long.
Remedy 3:	When the encoder acquisition signal output timing is late or the encoder acquisition signal output width is long, analysis of detection results may end before the signal falling is detected in the latch program, which causes a misjudgment that the latch has failed. Use the signal output at the same time as snapping as the encoder acquisition signal, and set a short output width. Or, add a wait command before the command for judging if latch is successful in the action program.

F.8.6. The grip position is incorrect

Cause 1:	The coordinate conversion parameter was not properly calculated.
Remedy 1:	Calculate the coordinate conversion parameter again, referring to “F.5.7. Calculating the coordinate conversion parameter(P.754) ”. After calculating the coordinate conversion parameter, set the reference position again.
Cause 2:	The correct reference position is not set.

Remedy 2:	Set the reference position again, referring to “F.5.8. Setting up a reference position for the infeed conveyor, and teaching a robot position(P.755) ”. Execute PKWCEND to confirm that the sensors are off as the first step of the reference position setting.
Cause 3:	The time for the signal to acquire the encoder count varies.
Remedy 3:	If the signal to acquire the encoder count is output at different times, the proper conveyor position when an image is snapped is not acquired. Consider using other signals output at a determined time.
Cause 4:	The accurate trigger is disabled.
Remedy 4:	You must execute the accurate trigger instruction to enable the accurate trigger before sending the input message. Confirm that DI/RI Num. to get Enc Cnt for the sensor object is correct.
Cause 5:	Tracking frame has not been set correctly.
Remedy 5:	For details, see “I.5.12.2. Adjustment of the tracking frame error(P.387)”.

Revision record

Edition	Date	Contents
04	Mar., 2026	• Changed descriptions to include software version number 7DH1 series 28 (V10.10P/28) *( ) indicates the software version in which the function was implemented in the following. • Added a description of the servo conveyor tracking function (7DH1 series 19, V10.10P/19) • Added a description of the conveyor buffer function (7DH1 series 22, V10.10P/22)
03	Jul.,2025	• Changed descriptions to include software version number 7DH1 series 18 (V10.10P/18) *( ) indicates the software version in which the function was implemented in the following. • Updated the descriptions of the recipe operation by means of a PLC (7DH1 series 09) • Added descriptions related to execution of production (7DH1 series 09) • Added restrictions for Spot Tool+ (7DH1 series 15) • Added description of signal noise removal (7DH1 series 15) • Added descriptions of tray redetection (7DH1 series 15) • Added descriptions for cases involving multiple robots (7DH1 series 15) • Added descriptions of the KAREL programs: PKSETSYNCINT.PC, PKENBSPFUNC.PC, and PKDSBSPFUNC.PC (7DH1 series 16) • Added a description of the “Sort by Measurement Value 1” function (7DH1 series 17) • Added descriptions regarding debug data collection (7DH1 series 17) • Added a description regarding the display that appears when the number of active sensors exceeds the maximum limit while setting the detailed settings of a sensor (7DH1 series 18)
02	Jan., 2025	• Changed descriptions to include software version number 7DH1 series 12 (V10.10P/12) *( ) indicates the software version in which the function was implemented in the following. • Added descriptions of the KAREL program PKCSPUTQUE2.PC (7DH1 series 10) • Added a description of the hand camera tracking (7DH1 series 10) • Added a description of the Vision I/F add-on (7DH1 series 11) • Changed a description of the sensor customization (7DH1 series 12)
01	Jun., 2024	

Index

< α >

αiA4000S Pulsecoder 710

<Number>

3D Vision Sensor 620
4D pick boundaries 590
4D pick frames 590

<A>

Aborting production 392
About the support scope 738
Access to the robot homepage 27,153
Action program 747
Adding a tracking frame 326
Adjustment of the dynamic error of the robot 139,387,441
Adjustment of the tracking frame error ... 131,387,419,439
Adjustment of the tracking frame error (When you use ADJ_OFS) 488
Appendix 703

<B>

Basic functions reference 258
Before use 24
Buffer KAREL macros 499
Buffer TP programs 500
Built-in functions 732

<C>

Calculating the coordinate conversion parameter 754
Calibration grid 257
Camera calibration 89,203
Camera installation and connection 255
Caution on setting up a network 242
Cell operation 272
Changing the placement position depending on the part type 534
Changing the workcell scene using recipes 591
Checking and fine adjustment of the tracking motion 127,386
Checking the connection of the camera 32,158
Checking the input of the trigger sensor 36,162
Checking the production status 567,690
Checking the Pulsecoder settings 32,158
Checking the robot torque 576
Checking the status of a conveyor 568
Checking the status of a conveyor station 569
Checking the status of a fixed station 571
Checking the status of a sensor 568
Clearance check function 637
Clearance guide 642
Collecting debug data 650
Collecting the iRPickTool diagnostics data 649
Communication cable 706
Communication settings 741
Configuration of software options 4
Configurations 4
Connecting a network cable 241
Connecting multiple robots 767
Connecting Pulsecoders 28,154
Connecting the Pulsecoder 253
Connection and setup of Pulsecoders 27,153
Connection of the communication cable 706
Continuous termination type specification for a taught position 387
Conveyor station 385
Conveyor station robot operation 428
Conveyor station-related programs 350,450
Conveyor-related programs 339,450
Coordinate conversion parameter calculation program 751
Creating a robot program 42,660,742
Creating a vision process 668
Creating the 4D Workcell 578
CVIS-151 (No more vision offsets) alarm displayed during tracking motion with multi-view 700

<D>

Debounce 676
Definitions of coordinate systems used 660
Definitions of registers used 659
Deleting the 4D Workcell 591
Detail setting of a recipe 547
Detecting multiple parts at one snap 764
Detection program 749
DI / RI and Find part by vision 382
DI and Find part by multiview 294
DI and Find part by vision 293
DI and Put part in queue 292
DI and Run program 295
Diagnostic function 649
Displaying the 4D Workcell 578
Distance and Find part by vision 291
Distance and Put part in queue 290
Distance and Run program 291
Duration 676

<E>

Essential points of customization 722
Example of creating a main program 599
Exchange workcell with using recipe 545
Executing production 442
External machine vision system connection 740
External machine vision system connection and settings 740
External machine vision system settings 741

<F>

Figures 711
Filtering for signals 675
Fine adjustment of the tracking motion 127,240,386,419,439,488

Fixed approach height for trays with multiple layers of cells....511
Fixed station....386
Fixed station robot operation....431
Fixed station-related programs....360,454
Flag and Put part in queue....297
Functional overview....678

<G>

get_conv_id....729
get_cstn_id....730
get_enc_id....729
get_idx_ecnt....728
get_sens_pos....730
get_tray_id....731
get_tray_pos....731
get_vp_name....731
get_work_id....730
Gripper and zone....423
Gripper-related programs....444

<H>

Handling parts detected before stop when restarting the system....544
HDI and Put part in queue....296
How to connect....713
How to create a TP program for index servo conveyor 597
How to use this manual....v

<I>

I Main part....1
Important notice....ii
In the case of a circular conveyor....283,329
In the case of a line conveyor....326
In the case of dual arm robots....392,443
Inaccuracy when a part is near the conveyor edges....699
Inaccuracy when a part rotates....699
Initialization program....748
Installing a Pulsecoder....253
IP address setting....741
iRPickTool (basic functions and Plug & Play) startup procedures....152
iRPickTool 4D graphics display....577
iRPickTool Basic (basic functions) startup procedures..26
iRPickTool robot maximum speed specification to prevent duty failures....514
iRPickTool safe retreat with skip outbound motion....509
IRVGETVRWRKID.PC....336
is_cond_dist....728
is_srvo_conv....728

<K>

KAREL program....332,411,444,563,674,690,755
KAREL program list....715
KAREL programs for servo conveyor Line Tracking..603

<L>

Latch fails....769
Latch program....750
Layer operation....273
Limiting the movable range of the final axis....572
LINE_COUNT....732
Linear distance specification....388
Linear face plate instruction....389
List of all the Standard TP programs....460

<M>

Main program....369,374,377,746
Maximum linear speed function....389
Mounting position setup....621
Multi-view....611

<N>

Network creation....241

<O>

Operating system and browser....706
Operator's panel....391
Other preparations before setup....241
Other programs....763
Other useful functions....545
Overview....3,611,620,637,717,736
Overview of the manual....v

<P>

P_CSGETID....734
P_CVGETID....734
Pausing production....392
Pausing, resuming, and aborting production....392
Payload IDs....438
Peripheral equipment....391
Pick or Drop Multiple....601
Pick pogram....375,378
Pick program....370
Pick program (buffer)....380
Picking a tray only when all cells are filled....542
Picking parts from two or more infeed conveyors....539
PK_BF_DROP1.TP....503
PK_BF_GET_DATA1.TP....500
PK_BF_OK2DROP1.TP....505
PK_BF_OK2PICK1.TP....505
PK_BF_PICK1.TP....501
PK_BUF_DROP1.TP....509
PK_BUF_PICK1.TP....508
PK_CV_DROP11.TP....475
PK_CV_GET_DROP_DATA11.TP....468
PK_CV_GET_PICK_DATA11.TP....467
PK_CV_OK2DROP1.TP....507
PK_CV_OK2PICK1.TP....506
PK_CV_PICK11.TP....472
PK_CV_WAITPOS1.TP....478
PK_CYCSTOP1.TP....487
PK_DROP1.TP....466
PK_FS_DROP11.TP....481
PK_FS_GET_DROP_DATA11.TP....470
PK_FS_GET_PICK_DATA11.TP....469
PK_FS_INDEX.TP....484
PK_FS_PICK11.TP....479

PK_FS_READY.TP....485
PK_GR_GET_ENDZONE1.TP....486
PK_GR_GET_UTOOLS1.TP....466
PK_INIT1.TP....462
PK_MAIN1.TP: Main program for group 1....461
PK_PERCH1.TP....464
PK_PICK1.TP....465
PK_PRGR_CV_PICKDROP1.TP....491
PK_PRGR_CV_PICKVTRAY1.TP....494
PK_PRGR_INIT1.TP....491
PK_PRGR_MAIN1.TP....489
PKABO.PC....334
PKCALOFSPOS.PC....674
PKCALOFSPOS2.PC....675
PKCOPYPR2POS.PC....335
PKCSACKBUF.PC....693
PKCSACKQPRGR.PC....412
PKCSACKQUE.PC....353
PKCSCALWPOS.PC....451
PKCSCLRACK.PC....359
PKCSCLRQUE.PC....351
PKCSFIXAPRZ....512
PKCSGENVRTRY.PC....411
PKCSGETAPPR.PC....452
PKCSGETBFDIS.PC....693
PKCSGETBUF.PC....691
PKCSGETCLRQ.PC....453
PKCSGETDIST.PC....355
PKCSGETDSLIN.PC....359
PKCSGETENC.PC....358
PKCSGETID.PC....350
PKCSGETNPRT.PC....357
PKCSGETOPPR.PC....452
PKCSGETPFRT.PC....356
PKCSGETPINFO.PC....359
PKCSGETPMUL.PC....452
PKCSGETPPCHK.PC....453
PKCSGETQCELL.PC....352
PKCSGETQPRGR.PC....411
PKCSGETQUE.PC....351
PKCSGETTIME.PC....354
PKCSGETTRID.PC....359
PKCSGETVR.PC....453
PKCSGETWPOS.PC....453
PKCSPUTQUE.PC....353
PKCSPUTQUE2.PC....354
PKCSSETDSLIN.PC....358
PKCSTRGPRGR.PC....412
PKCVCLRBUF.PC....691
PKCVENBLB.PC....341
PKCVENBSC.PC....342
PKCVGETCNVSPD.PC....345
PKCVGETNAME.PC....343
PKCVIDXDELQ.PC....343
PKCVSETLBD.PC....339
PKCVSETTRNAM.PC....340
PKDSBSPFUNC.PC....337
PKENBSPFUNC.PC....337
PKFSACKBUF.PC....363
PKFSACKQUE.PC....362
PKFSCLRACK.PC....364
PKFSCLRBUF.PC....362
PKFSCLRQUE.PC....361
PKFSFIXAPRZ....513
PKFSGETAPPR.PC....454
PKFSGETBUF.PC....363
PKFSGETBUFDT.PC....500
PKFSGETBUFPT.PC....500
PKFSGETCLRQ.PC....458
PKFSGETID.PC....360
PKFSGETIXDLY.PC....454
PKFSGETIXODO.PC....455
PKFSGETIXSIM.PC....455
PKFSGETOPPR.PC....455
PKFSGETPMUL.PC....456
PKFSGETPPCHK.PC....456
PKFSGETQUE.PC....361
PKFSGETRDY.PC....456
PKFSGETRDYDI.PC....456
PKFSGETTRID.PC....364
PKFSGETUF.PC....364
PKFSGETVR.PC....457
PKFSMONIDX.PC....457
PKFSPUTBUF.PC....362
PKFSPUTQUE.PC....360
PKFSSETIDXDO.PC....458
PKFSSETRDY.PC....457
PKGRCHKPP.PC....445
PKGRCLOSE.PC....445
PKGRGETDPDLY.PC....446
PKGRGETID.PC....446
PKGRGETPKDLY.PC....446
PKGRGETZNCNT.PC....446
PKGRGETZNUT.PC....447
PKGRINIT.PC....447
PKGROPEN.PC....447
PKLABO.PC....335
PKMVCALOFS2D.PC....762
PKMVCNVPX2MM.PC....762
PKMVCONNECT.PC....756
PKMVDELPRSPM.PC....759
PKMVDELSNDPM.PC....757
PKMVDISCO.PC....756
PKMVINIACT.PC....756
PKMVPRSMSG fails....769
PKMVSCLCALPM.PC....762
PKMVSCLPUTP1.PC....760
PKMVSCLPUTP2.PC....761
PKMVSCLPUTP3.PC....761
PKMVSCLSETP1.PC....761
PKMVSCLSETP2PC....761
PKMVSENDMSG1~4.PC....758,760
PKMVSETPRSPM.PC....759
PKMVSETRCVPM.PC....758
PKMVSETSNDPM.PC....757
PKMVSETVR.PC....763

PKMVWAITTASK.PC....763
PKPOSTERR.PC....335
PKPRGRGETEN.PC....334
PKRBGETBUFID.PC....499
PKRBGETCVCNT.PC....448
PKRBGETDPCV.PC....449
PKRBGETDPFS.PC....450
PKRBGETFSCNT.PC....448
PKRBGETID.PC....338
PKRBGETPKCV.PC....449
PKRBGETPKFS.PC....450
PKRBGETPRGRT.PC....448
PKRBGETSFLG.PC....338
PKRBSETSKPADJ....339
PKRBSETSVDLY....338
PKRCACTIVE.PC....566
PKRCCREATE.PC....563
PKRCDELETE.PC....564
PKRCEXPORT.PC....565
PKRCGETPLCIO.PC....566
PKRCIMPORT.PC....565
PKRCRESTORE.PC....565
PKRCSAVE.PC....564
PKRCSETPLCIO.PC....566
PKRSTWC.PC....334
PKSETNOMFRM.PC....674
PKSETSYNCINT.PC....336
PKSNENBSENS.PC....345
PKSNSETDBNC.PC....349
PKSNSETDURTN.PC....349
PKSNSETMTOFS.PC....348
PKSNSETVPNAM.PC....346
PKSNSETVTOFS.PC....347
PKSNSTART.PC....348
PKSNSTOP.PC....349
PKTRGETCL.PC....366
PKTRGETNUMCL.PC....366
PKTRSETCLPOS.PC....365
PKWCCHECK.PC....333
PKWCEND.PC....333
PKWCGETACTRC.PC....336
PKWCRSTPRDST.PC....334
PKWCSTART.PC....333
Placement program....373,375,379
Placement program (buffer)....381
Placing parts in different directions in a box....533
Placing parts in sequence from the end of the box....532
Placing parts on different outfeed conveyors depending on the model ID....537
Plug & Play buffer function....497
Plug & Play reference....421
Polarity Inversion....677
Position register operation settings....754
Position Registers....436
Possible cases....654
Precautions for when starting up multiple robots....390
Preface....v
Preparation before setting up the application....27,153
Preparation for camera....612
Procedure to customize a sensor task....717
Production runs....391
Production Status....619
Program for setting a trigger....768
Programs related to communication....756
Programs related to coordinate conversion parameter calculation....760
Programs related to detection position conversion....762
Programs related to initialization....756
Programs related to input message transmission....757
Programs related to output message analysis....759
Programs related to output message reception....758
Pulsecoder installation, connection and setup....252
put_part_cst....726
put_queue....727

<Q>

Queue managed tracking....4

<R>

Recipe....268
Recipe operation by means of a PLC....557
Recommendations....25
Reference position setting....382,419,439
Reference position setting and robot position teaching 618
Reference position setting program....752
Regarding Microsoft® products....vii
Registers....436
Related manuals....vi
Reporting failure to pick a part....541
Requirements....710
Restriction concerning sensors that can be operated simultaneously....317
Restrictions....24,592,612,621,637,638,655,658,737
Restrictions of SpotTool+....25
Restrictions on the external machine vision systems....738
Restrictions on the robot system....737
Resuming production....392
Revision record....771
RI and Find part by multiview....300
RI and Find part by vision....298
RI and Put part in queue....297
RI and Run program....301
Robot behavior under the standard TP programs....435
Robot pick/place operations in the buffer....498
Robot programs....367,413,434
Robot-related programs....338,447
Routines....726
run_vis_cst....726

<S>

Safety precautions....iii
Sample programs for the conveyor-to-buffer-to-conveyor configuration....376
Sample programs for the conveyor-to-conveyor configuration....368,417

Sample programs for the conveyor-to-fixed station configuration 373
Sample programs for the conveyor-to-same-conveyor configuration (downstream robots)....417
Sample programs for the conveyor-to-same-conveyor configuration (the first robot) 414
Sample programs for the infeed-to-buffer-to-outfeed-conveyor configuration....694
Sample system configurations....12
Selecting the operator's panel for starting production..391
Selective display of specific information 590
Sensor customization....717
Sensor installation and connection 683
Sensor-related programs....345,450
Servo conveyor error adjustment parameters 600
Servo conveyor function 600
Servo conveyor stop-wait function....600
Servo conveyor tracking....592
Servo conveyor tracking restrictions 611
Setting a recipe....545
Setting a sensor in detail....289
Setting a teaching PC 706
Setting a tracking frame 72
Setting an infeed conveyor sensor....80,194
Setting an infeed conveyor sensor position 662
Setting an IP address for the PC 707
Setting example appropriate to the usage 522,764
Setting payloads (motion performance)....58,168
Setting the number of objects 266
Setting the pick conveyor....517
Setting the place conveyor 519
Setting the reference cell 275
Setting the sensor position....84
Setting the target position....312
Setting the tray position....306
Setting up a buffer....498
Setting up a client tag 741
Setting up a conveyor....71,185,276,397,640
Setting up a conveyor station 106,220,318,405,648
Setting up a fixed station....332,411,434
Setting up a gripper 178
Setting up a gripper and a zone 422,488
Setting up a reference position and teaching a robot position....113,230
Setting up a reference position for the infeed conveyor and teaching a robot position....113
Setting up a reference position for the infeed conveyor, and teaching a robot position....230,670,755
Setting up a reference position for the outfeed conveyor and teaching a robot position....122
Setting up a reference position for the outfeed conveyor, and teaching a robot position 235
Setting up a robot 268
Setting up a robot operation 225,427
Setting up a robot-mounted camera....665
Setting up a sensor....80,194,288,404,616,640
Setting up a sensor position....197,302
Setting up a tool frame for the hand 111,226
Setting up a tracking frame 186,283
Setting up a tray 67,181,269,396,639
Setting up a workcell....60,170,266
Setting up an extended axis for an index conveyor 592
Setting up an index servo conveyor....594
Setting up an infeed conveyor....71,185
Setting up an infeed conveyor sensor....661,753
Setting up an outfeed conveyor....72,186
Setting up an outfeed conveyor sensor (In the case of [DI] / [RI] + [Put part in queue]) 81,195
Setting up an outfeed conveyor sensor (In the case of [HDI] + [Put part in queue])....83,196
Setting up applications 60,170,683
Setting up conveyor station robot operation in detail ...429
Setting up fixed station robot operation in detail....432
Setting up iRPickTool....742
Setting up Pulsecoders 28,154
Setting up Pulsecoders for multiple robots....254
Setting up Pulsecoders that are connected to the robot controller 253
Setting up the gripper in detail 425
Setting up the IP address 242
Setting up the operation and loading programs 516
Setting up the robot ring....246
Setting up the tool frame 257
Setting up the tool frame of the pointer tool....37,163
Setting up the vision system....89,203,305,613
Setting up the zone in detail 426
Setting up tracking schedules....599
Settings required for each combination of a trigger condition and a trigger action 302
Settings required for recipe operation by means of a PLC 558
Setup 592,613,639
Setup data file....705
Setup of a fixed station....521,649
Setup of a robot operation 488
Setup of a workcell and robot....396,639
Setup of multiple stations....515
Setup of pre-grouping 393,487
Skipping the tracking motion outside the area 574
Slow vision processing due to delayed image snap timing 699
Snapping or detection is not executed in the external machine vision system....768
Specified By....676
Stacking parts in multiple layers in a box....530
Standard TP program details 460
Standard TP programs....458,489
Starting production with peripheral equipment 442
Starting production with the operator's panel 442
Startup procedure 656,682,739
String Registers....437
Study for application 19
Switching recipe with TP program....563

<T>

Teaching method....658

Teaching the position to robots 385,419,439
Teaching the robot home position 240
Terms 7,396,736
The external machine vision cannot be connected 768
The grip position is incorrect.....769
The overrun count continues to increase 699
The stop conveyor does not work.....699
Time before instruction 390
Timeout occurs in PKMVRCVMSG.....768
Tool frames 438
TP program naming conventions.....459
Tracking program.....367
Tracking program for picking workpieces from a backward-moving buffer 696
Tracking program for picking workpieces from a forward-moving buffer 696
Tracking program for placing workpieces on a backward-moving buffer 695
Tracking program for placing workpieces on a forward-moving buffer 695
Tracking using robot-mounted camera.....658
Tray redetection 653
Tray-related programs 365,458
Troubleshooting 697,768

<U>

update_nfnd.....726
update_wid 732
User alarms.....438
User frames 438
Using a conveyor as a buffer 678
Using DI / RI.....36,162
Using different robots to pick parts depending on their characteristics 526
Using different robots to pick parts depending on their positions on the conveyor 522
Using the touch screen icons 589
Using user-defined object shapes 591

<V>

V_RUN_FIND 733
Vision I/F add-on 736
Vision process 99,213
Vision process (2D).....634
Vision process (3D).....629
Vision Registers 438
Visual tracking 6
VT_GET_FOUND 733

<W>

When [DI] and [Find part by vision] are selected 309
When [Distance] and [Find part by vision] are selected.....306
When a tray is not used 303,312
When a tray is used 314
When parts are missed frequently 697
When robot motion is not smooth 699
When settings cannot be changed during production ...700
When the robot does not pick up parts 697
When the same part is picked up twice 698
Workcell setup screen 422
Workcell-related programs.....333,444

B-84454EN/04