EPSON

Epson RC+ 8.0 选件
Part Feeding 8.0
导入&软件篇

翻译版

Rev.4
SCM259S7759F

目录

1. 前言

1.1 前言

感谢您购买本公司的机器人系统。
本手册记载了正确使用Epson RC+ Part Feeding选件所需的事项。
使用系统之前，请仔细阅读本手册与相关手册，正确地进行使用。
阅读之后请妥善保管，以便随时取阅，如有不明之处，请再次阅读。

本公司的产品均通过严格的测试和检查，以确保机器人系统的性能符合本公司的标准。但是如果在超出本手册所描述的环境中使用本产品，则可能会影响产品的基本性能。

本手册阐述了本公司可以预见的危险和问题。请务必遵守手册中所述的安全注意事项，已确保安全正确地使用我们的机器人系统。

1.2 商标

Microsoft、Windows、Windows商标、Visual Basic、Visual C++为美国Microsoft Corporation在美国与其它国家的注册商标或商标。其它公司名称、商标名称、产品名称均为各公司的注册商标或商标。

1.3 关于标记

Microsoft® Windows® 10 Operating system
Microsoft® Windows® 11 Operating system
本使用说明书将上述操作系统分别标记为Windows 10、Windows 11。另外，有时可能将Windows 10、Windows 11统一标记为Windows。

1.4 注意

严禁擅自复制或转载本使用说明书的部分或全部内容。
本说明书记载的内容将来可能会随时更改，恕不事先通告。
如发现本说明书内容有何错误或不妥之处，请与本公司联系。

1.5 制造商

SEIKO EPSON CORPORATION

1.6 联系方式

有关联系方式的详细信息，请参阅以下手册中的“销售商”。
“安全手册”

1.7 阅读本手册之前

本节介绍了您在阅读本手册之前应了解的事项。

关于Epson RC+ 8.0的安装文件夹

Epson RC+ 8.0的安装文件夹可改到任意位置。本手册中的说明是以Epson RC+ 8.0被安装到“C:\EpsonRC80”中为前提条件的。

2. 导入篇

2.1 前言

2.1.1 关于Part Feeding

可利用Epson RC+ 8.0选件Part Feeding（以下简称Part Feeding选件），简单地开发通过送料器分离部件并由机器人从送料器中拾取部件的系统。

2.1.1.1 背景

为了应对产品的短寿命化、多品种化潮流与JIT（少批次、短交货期）等，制造的生产形态呈现出多样化趋势。一方面，随着租金持续上涨、制造现场人员流失以及劳动人员高龄化等状况的不断加剧，依赖于人的制造活动已难以为继。另外，如何实现可应对多样化的灵活性，已成为相关行业当前面临的课题。
作为构成自动化生产线的要素之一，让我们设想一下部件供给的场景。部件供给是生产设备的重要因素，可防止产量超出部件供给能力。部件供给方式的主流是造价便宜且选项较多的振动型碗式送料器。但需要根据供给部件制作振动碗，并按部件手动更换振动碗，还需要专业技术人员进行调整。这需要高级的工程技术能力和经验，而且，还面临着难以应对多品种和短交货期的课题。
作为克服振动型碗式送料器缺点的装置，近年来市场推出了“智能送料器”（以下简称送料器）。该装置的特点在于，可通过更改送料器的设定，简单地应对各种供给部件。另外，市场也推出了与机器人（可进行送料器与部件识别的图像处理并搬运识别部件）组合使用的产品。
这种产品只能通过机器人系统进行送料器的单独动作，还需要客户在送料器的调整、操作、图像处理以及机器人搬运部件方面进行设计。综合探讨并设计部件供给需要时间与经验。比如，循环时间会因部件投放数量或机器人拾取部件的位置而发生较大的变化。如果未适当地进行设计，即使送料器具有高性能与高功能，也无法充分发挥其功能或性能，进而提高循环时间。就原来的产品而言，不可能做到任何人都可以马上使用送料器的程度。

2.1.1.2 导入Part Feeding选件的优点

如果导入Part Feeding选件，则具有下述优点。

- 送料器、视觉系统与机器人的完美整合
  一站式提供部件供给所需的要素。将送料器、视觉系统与机器人完全整合在一起。与单独准备并进行评估时相比，可免除导入时的繁琐。

- 削减装置开发与启动工时
  自动进行送料器/视觉系统运作无需客户编程。虽然由客户来记述机器人动作，但仅为写入到模板代码中。自动进行送料器/视觉系统与机器人运作的同步控制。可通过最低限度的编程开发装置。也可以简单地与客户当前使用的环境整合。可利用GUI简单地进行送料器、机器人与视觉系统方面的设定。

- 削减停工时间与运行成本
  可利用Part Feeding选件应对各种部件。部件切换时无需更改设备，可缩短生产时的停工时间。另外，无需制作新设备，可控制长期运行成本。

2.1.1.3 Part Feeding选件的功能

可利用下述典型功能，简单、充分地发挥送料器的功能/性能。另外，也可以构建高效的部件供给系统。

- 与送料器之间的通信I/F
  本系统内置有用于进行送料器设定/控制的通信程序。无需客户进行通信编程。
- 送料器的自动调整功能
  内置有根据部件自动调整送料器参数（振幅、振动时间等）的功能。客户只需按简单的步骤进行操作，即使没有送料器相关知识，也可以简单地进行调整作业。
- 送料器的控制算法
  内置有送料器的控制算法。该算法主要用于尽可能缩短机器人拾取部件所需的时间。即使客户未加关注，也可以实现高效的运作。

- 用于掌握机器人/送料器的运转状况的循环时间日志输出功能
  内置有用于将机器人、送料器、视觉系统的运作时间输出为文件的功能。通过更改参数设定、获取日志并进行分析，可实现高效运作。要使用该功能时，需要将装有Epson RC+ 8.0的PC连接至控制器。
- 多送料器运作
  可将最多4台送料器连接至1台控制器进行控制。T/VT系列时，最多可控制2台。要协同进行多个送料器运作时，由于可使用1台控制器进行控制，因此程序较为简单。
- 多部件运作
  可同时将最多4种部件装入1台送料器中进行处理。可减少送料器的设置台数，因此可实现低成本化或省空间化。多部件运作时，1台送料器最多可使用2台机器人。
- 清除运作
  内置有用于排出送料器上的部件的运作功能。可用于因切换品种而要自动排出送料器上的部件，或排出不良部件与过多投放的部件。为IF-80时，带有用于进行清除运作的平台以及用于存放排出部件的容器。

2.1.2 作为前提的Epson RC+ 8.0的基本知识

Part Feeding选件以Epson RC+ 8.0环境为核心。
要使用Part Feeding选件时，需要具备有关Epson RC+ 8.0开发环境、Epson机器人、Epson RC+ 8.0选件Vision Guide 8.0方面的知识。本手册内容是以具备下述事项知识的人员为对象进行说明的。

- Epson RC+ 8.0项目管理概念与使用方法
- 利用Epson RC+ 8.0创建并编辑SPEL+程序的方法
- 通过运行窗口执行SPEL+程序的方法
- SPEL+的基本语言结构、功能与使用方法
- Vision Guide 8.0的功能与使用方法

2.1.3 相关手册

使用Part Feeding选件时，请参阅以下手册。

- “Epson RC+ 8.0选件 Part Feeding 8.0 IF-***篇”
  ***：送料器机型名称（IF-80、IF-240或IF-380/530）
  “Epson RC+ 8.0选件 Part Feeding 8.0 IF-A1520 & IF-A2330篇”
  记载了各送料器使用方法的说明。
- “Epson RC+ 8.0选件 Part Feeding 8.0 Hopper篇”
  记载了料斗使用方法的说明。
- “Epson RC+ 8.0用户指南”
  记载了Epson机器人控制系统使用方法的说明。
- “SPEL+语言参考”
  记载了SPEL+语言的命令的说明。
- 各机器人手册
  记载了有关机器人的各种说明。

2.1.4 关于正文中的符号

正文中使用几个标记来记载重要事项。如下所述为各标记的含义。

⚠ 警告

该符号表示如果无视该标识进行错误使用，则可能会导致死亡或重伤的内容。

⚠ 警告

该符号表示如果无视该标识进行错误使用，则可能会因触电而导致受伤的内容。

⚠ 注意

该符号表示如果无视该标识进行错误使用，则可能会导致受伤或只发生物品损坏的内容。

要点

在操作机器人时，必须遵守的事项等各项须知事项。

提示

如何简化操作以及有关操作方法的提示内容。

2.2 安全

使用之前，请仔细阅读本手册，正确地进行使用。
阅读之后请妥善保管，以便随时取阅，如有不明之处，请再次阅读。

2.2.1 安全注意事项

⚠ 警告

- 请勿将本产品用于确保安全的用途。

  该符号表示如果无视该标识进行错误使用，则可能会导致死亡或重伤的内容。
- 请在手册记载的使用条件下使用本产品。如果在未满足条件的环境中使用，则不仅会缩短产品的使用寿命，还可能会造成严重的安全问题。

⚠ 注意

- 请从本公司销售商采购送料器。
- 请从本公司销售商采购相机与相机电缆。

如果是从本公司销售商之外采购的，则不被视为保修对象。

2.2.2 机器人的安全

让机器人或其他自动装置进行运作时，请以安全为最优先事项。控制器或Epson RC+ 8.0内置有许多安全功能。请使用紧急停止或安全门输入等各种安全功能。请在设计机器人单元时使用这些安全功能。
有关安全信息与指南，请参阅以下手册。
“Epson RC+8.0 用户指南 - 关于安全”

2.2.3 视觉系统的安全

有关视觉系统的安全，请参阅以下手册。
“Vision Guide 8.0硬件&设置篇 - 安全注意事项”

2.2.4 送料器的安全

有关送料器的安全，请参阅以下某手册。

- “Epson RC+ 8.0选件 Part Feeding 8.0 IF-80篇 - 安全注意事项”
- “Epson RC+ 8.0选件 Part Feeding 8.0 IF-240篇 - 安全注意事项”
- “Epson RC+ 8.0选件 Part Feeding 8.0 IF-380篇 IF-530篇 - 安全注意事项”
- “Epson RC 8.0选件 Part Feeding 8.0 IF-A1520 & IF-A2330篇 - 关于安全”
- “安全手册(Part Feeding)”

2.2.5 料斗的安全

有关料斗的安全，请参阅以下手册。
“Epson RC+ 8.0选件 Part Feeding 8.0 Hopper篇”

2.3 术语定义

下面说明术语。

硬件

术语	描述
送料器	是通过振动分离散装部件，以便于机器人搬送的设备。
平台	为送料器的构成部件，属于部件托盘部分。
部件	是机器人处置的部件。请客户自行准备。
添加进给	为通过料斗向送料器添加部件的进给方式，可确保送料器上的部件始终处于最佳数量状态。
隔开进给	为隔开送料器上的所有部件之后，通过料斗向送料器投放部件的进给方式。
全面拾取	以送料器上分散的部件中所有可拾取的部件为对象进行拾取动作。
部分拾取	以送料器上分散的部件中位于特定区域的部件为对象进行拾取动作。可利用Part Feeding选件，选择在送料器上定义的4个区域，并从中拾取部件。
自定义照明	是客户准备的照明。用于识别送料器背光灯无法识别的部件或部件姿势（表里等）。

术语	描述
料斗	是用于向送料器平台供给部件的装置。
清除	是指排出残留在送料器中的部件。
清除门	该开闭门用于排出残留在送料器中的部件。 连接送料器后，可利用命令控制开闭。 可用于切换多品种少量生产的部件品种或排出不良部件。
送料器校准	是调整送料器参数的作业，以确保部件在送料器上适当地移动。
多送料器	是指将多台送料器连接至1台控制器的构成。本选件支持最多4台送料器。
多部件	是指将多种类型的部件放入到1台送料器中进行同时处理的构成。本选件支持最多32个部件。

软件

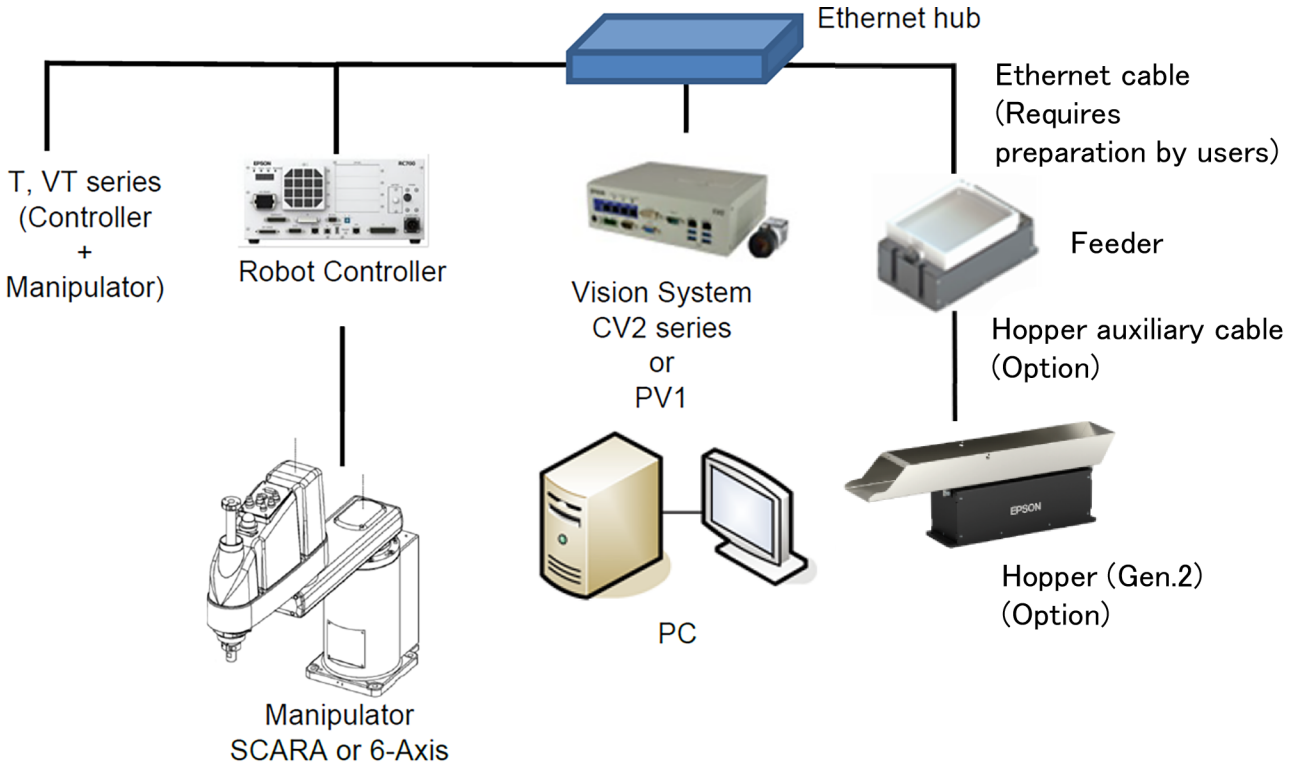
术语	描述
拾取	是指通过机器人抓取送料器上的部件。
放置	是指将机器人抓取的部件投放或搁在指定场所。
Part Feeding进程	是对Part Feeding选件内置的视觉系统或送料器的运作进行自动化并调用机器人运作的自动进程。
回调函数	是用于在指定条件下调用Part Feedeing进程的SPEL+函数。客户记述函数的内容。记述客户装置固有的处理（例：利用机器人抓取、放置部件）。
部件坐标队列	用于获取送料器上的部件的坐标。按本地坐标处理坐标。
UPM	Unit per minute 是机器人每分钟处理的部件数。
有效部件	是多部件运作时的主体部件。使用该部件的送料器运作参数。

2.4 系统概述

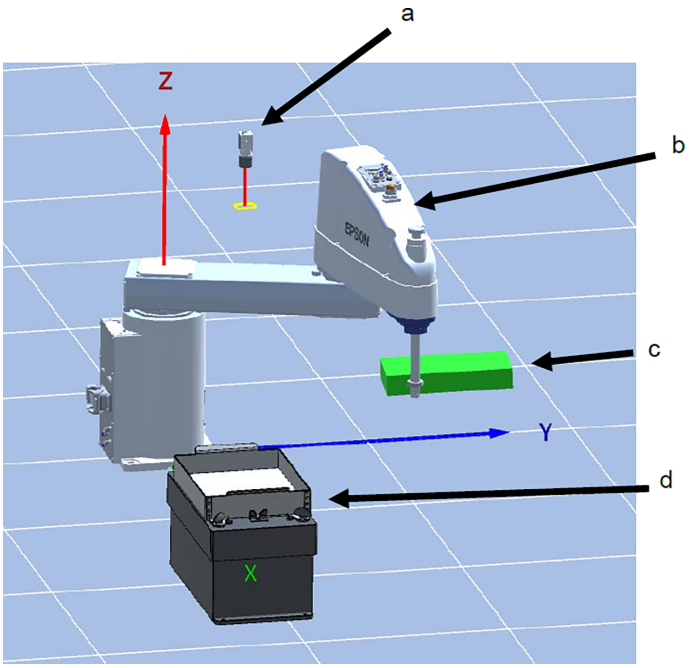
如果使用Part Feeding选件，则可简单地构建取放部件的系统。下面说明系统构成。

2.4.1 全体构成

如下所述为使用Part Feeding选件的系统构成示例。
送料器IF-80附带有内置料斗。



如下所述为机械手周边的配置示例。



符号	项目
a	相机
b	机械手
c	部件放置场所（托盘等）

符号	项目
d	送料器

2.4.2 送料器

送料器为本系统必需品。请务必准备。
可使用IF-80、IF-240、IF-380、IF-530、IF-A1520或IF-A2330送料器。不支持其他送料器。
可利用1台控制器控制多台送料器。
请使用从销售商采购的送料器。
如果未使用从销售商采购的送料器，则可能无法连接至控制器或无法充分发挥性能。

2.4.3 机器人

机器人为本系统必需品。请务必准备。

2.4.3.1 机械手

RC700系列、RC90系列、可连接至RC800-A控制器的SCARA型与6轴机械手以及T/VT系列可使用。不支持X5系列、PG。
1台送料器最多可设置2台连接至同一控制器上的机械手。

2.4.3.2 末端夹具

末端夹具用于抓取部件。包括真空/压缩空气吸附型与卡盘机构型。请根据客户的部件或需求进行选择。
本公司不销售末端夹具。请客户自行准备末端夹具。

2.4.4 视觉系统

视觉系统为本系统必需品。请务必准备。

2.4.4.1 视觉系统

可使用以下某种视觉系统。

- PC视觉PV1
  有关所需规格，请参阅以下内容。
  PC
- 紧凑型视觉CV2-SA/HA（固件Ver.3.0.0.0以后版本）或CV2-HB/SB/LB（固件Ver.3.2.0.0以后版本）
  不支持CV1与CV2-S/H/L。
  不支持其他公司视觉系统。

2.4.4.2 相机

可连接视觉系统的相机均可使用。
准备1台用于识别送料器上的部件的相机。作为向下固定相机或安装到机器人移动轴上的移动相机设置该相机。
可根据需要，追加用于进行已抓取部件的位置补偿的向上相机或用于对部件放置场所进行定位的相机。

2.4.5 照明

需要使用照明，以便正确识别平台上的部件。
可使用以下某种或两种照明。

- 内置于送料器中的背光灯
- 客户准备的自定义照明（I/O控制、以太网控制等）

2.4.6 PC

进行以下作业时，需要使用PC。

- 浏览或编辑Part Feeding选件设定时
- 进行送料器校准时
- 进行SPEL+编程时
- 获取日志时

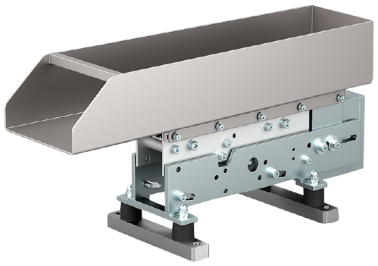
可在不连接PC的状态下进行Part Feeding进程运作。如下所述为PC所需的规格。

- 使用CV时：应可安装 RC+
- 使用PV时：请参阅以下内容。
  “Vision Guide 8.0硬件&设置篇 - 系统条件”

2.4.7 料斗

料斗是用于向送料器供给部件的装置。属于选件。
可将从销售商采购的料斗连接至送料器。
通过IO或Ethernet等，将客户准备的送料器连接至控制器。
送料器IF-80为料斗一体型型号。

料斗包括第1代 (Gen.1) 与第2代 (Gen.2) 2种系列。

料斗 (Gen.2)	料斗 (Gen.1)
	
■ 是比料斗 (Gen.1) 更新的型号。 ■ 通过Epson RC+ 8.0调整振幅。	■ 利用安装在附带的料斗控制器上的电位器调整振幅。

2.5 硬件

2.5.1 确认随附品

根据订购内容，交付Part Feeding选件时随附下述物品。到货后，请立即确认是否有缺货。另外，请确认是否损坏。

- Part Feeding许可证（有另寄的情况）

- 送料器主体、背光灯（内置）
- 平台
- 送料器电源电缆
- 以太网电缆

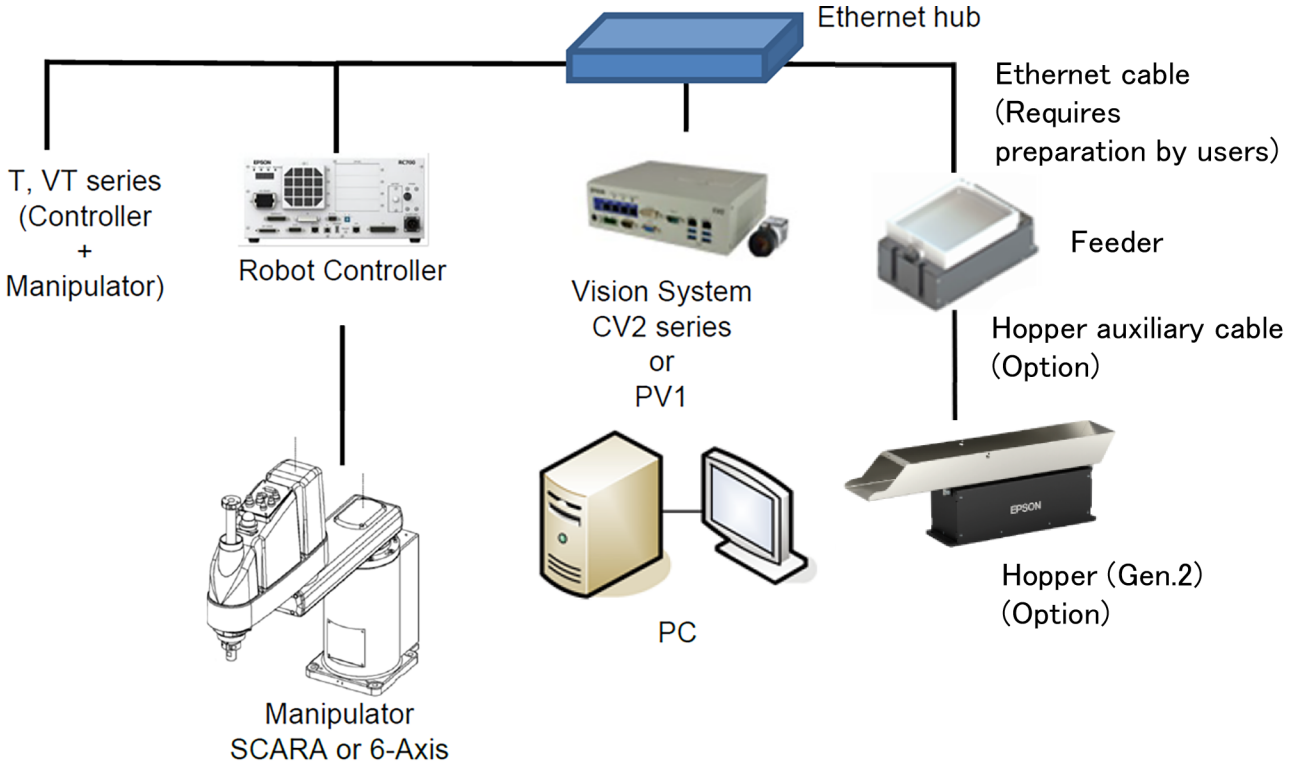
以下为选件。

- 料斗
- 料斗控制器（料斗 (Gen.1) 用）
- 料斗附带电缆
- 其他选件

2.5.2 系统构成

2.5.2.1 构成示例

如下所述为使用Part Feeding选件的系统构成示例。
送料器IF-80附带有内置料斗。



1台控制器最多可设置4台送料器。T/VT系列时，可同时控制最多2台送料器。
多部件运作时，1台送料器最多可设置2台机械手。
请使用以太网集线器连接以太网。请使用下述视觉系统。

紧凑型视觉系统 CV2系列	CV2-HA/SA（固件Ver.3.0.0.0以后版本）
	CV2-HB/SB/LB（固件Ver.3.2.0.0以后版本）
PC视觉PV1	

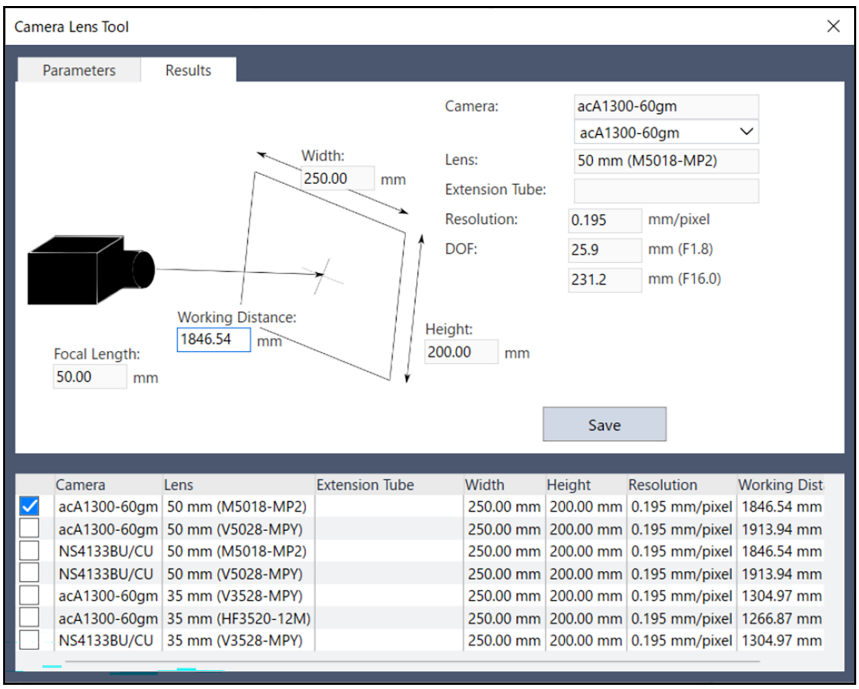
不支持CV1与CV2-S/H/L。
不支持其他公司视觉系统。

2.5.2.2 构成选择相关考虑事项

- Part Feeding选件可使用的机械手为SCARA型机器人与6轴型机器人。不支持X5系列、PG。
- 使用PV时，请勿利用同一网络端口与送料器以及相机进行通信。如果发生同时与送料器以及相机进行通信的现象，则可能会对拍摄或送料器的运作产生不良影响。请在PC中增设网卡，或使用多端口网卡直接连接相机与网络端口。
- 从送料器的规格方面看，即使多个机器人控制器连接1台送料器，也不会发生错误。进行网络设定时，请注意IP地址的分配，不要将多台机器人控制器连接至1台送料器。
- 1台送料器最多可设置2台料斗。
- 可在模拟器中显示送料器与料斗的CAD数据。CAD数据文件位于以下文件夹中。
  `C:\EpsonRC80\Simulator\CAD\PartFeeder`

2.5.2.3 选择相机镜头

使用光学选择工具，根据视野、工作距离选择镜头以及延伸管的厚度。
可通过Epson RC+ 8.0菜单 -［设置］-［系统设置］-［视觉］-［相机］-［相机镜头工具］，启动光学选择工具。
有关详细信息，请参阅以下内容。
“Vision Guide 8.0硬件&设置篇 - 设置篇 - 光学选择功能”



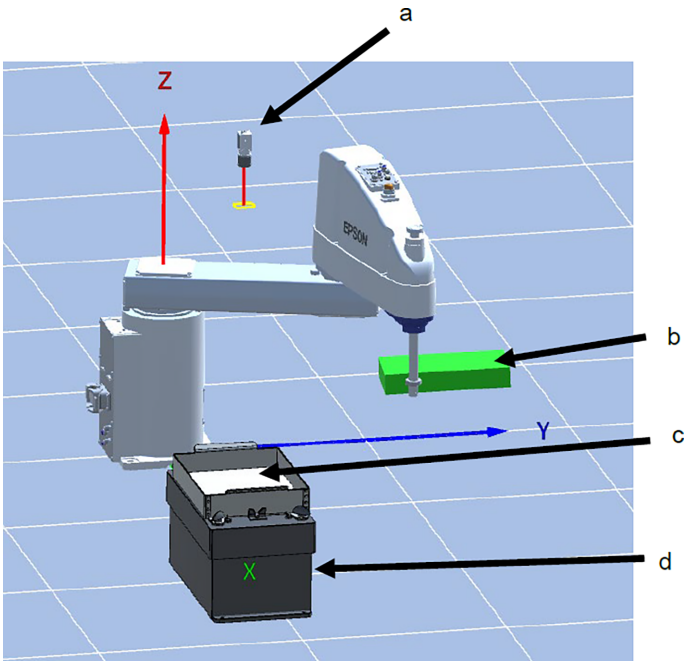
2.5.3 设置与调整

2.5.3.1 机械手与控制器

有关机械手的安装，请参阅各机械手手册。尤其请根据机器人系统安全手册，在注意安全的基础上进行设置。
请客户自行制作或采购末端夹具。有关控制器的设置，请参阅控制器手册。

2.5.3.2 相机与镜头

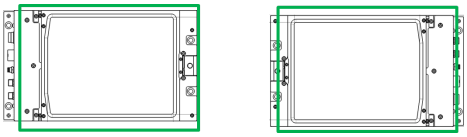
向下设置相机，以便可环视整个送料器平台。
建议向下固定相机。也可以使用移动相机。



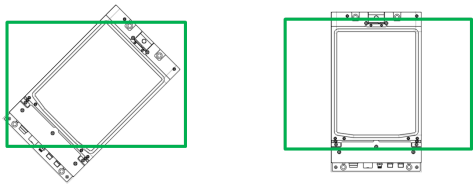
符号	项目
a	相机（向下固定）
b	放置位置
c	平台
d	送料器

需要将送料器的纵向与相机视野（下图中的绿框）的水平方向对准。否则，Part Feeding系统不会正常运作。送料器的方向左右均可。

良好示例



不良示例



即使是移动相机，也如上图所示，对平台的纵向与相机视野的水平方向进行示教（创建点数据），作为拍摄位置。
利用相机拍摄部件时，如下图所示，调整镜头的焦点与光圈，以便可清晰地识别部件并确保平台面具有均匀的亮度。

2.5.3.3 送料器与料斗

安装送料器与料斗时，请注意下述事项。

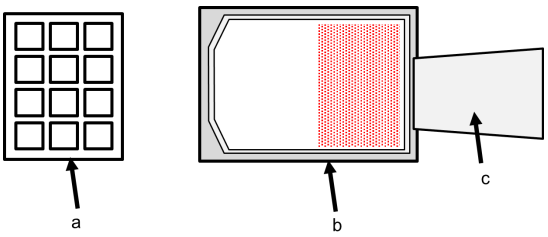
- 设置在水平面上。
- 设置在高刚性台座上。
- 利用螺栓牢固地固定。

如果安装到不平的场所或低刚性台座上，部件的分散则会产生偏移，或部件无法充分地分散，导致可拾取的部件减少，造成循环时间过低。有关安装的详细信息，请参阅以下各送料器手册。
“Epson RC+ 8.0选件 Part Feeding 8.0 IF-***篇 - 环境与设置”
***：送料器机型名称（IF-80、IF-240、IF-380、IF-530）
“Epson RC+ 8.0选件 Part Feeding 8.0 IF-A1520 & IF-A2330篇 - 设置”

将料斗设置在水平面上。利用螺栓牢固地固定到高刚性台座上。请将朝向料斗平台方向的伸出控制在最低限度。通过尽可能扩大平台面，增加可拾取的部件，以提高循环时间。有关安装的详细信息，请参阅以下手册。
“Epson RC+ 8.0选件 Part Feeding 8.0 Hopper篇”

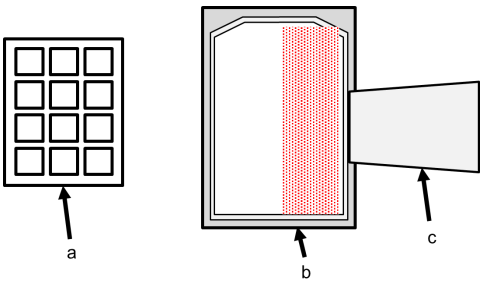
送料器与料斗的位置关系：配置料斗时，使部件在送料器上的投放位置位于放置位置的相对侧。设置为将部件投放到下图红色阴影区域。如果采取这样的配置，则可利用Part Feeding选件的并行进给功能，提高系统的循环时间。

配置示例1



a	放置位置
b	送料器
c	料斗

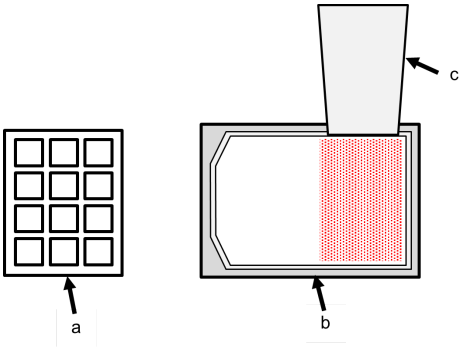
配置示例2



a	放置位置
b	送料器
c	料斗

如果部件在料斗进给期间滚入到机器人要拾取部件的区域，则会对部件拾取产生不良影响。
为了防止发生这种现象，按以下方式配置料斗的话，可能会有效果。

配置示例3



a	放置位置
b	送料器
c	料斗

要通过料斗适当地投放部件时，可将适当数量的部件投放到适当的位置是至关重要的。为此，请探讨对料斗进行如下所述的追加加工。

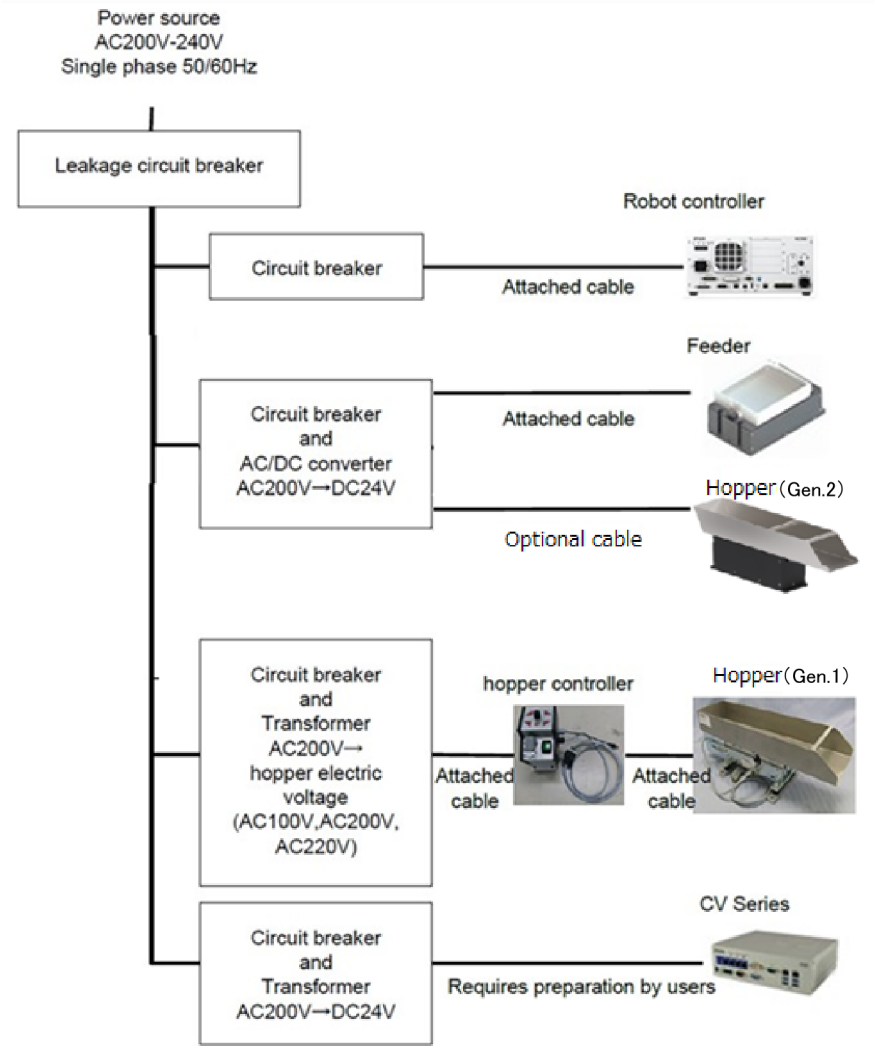
- 在料斗中间设置用于调整部件流量的阻挡机构
- 在料斗出口设置用于限制部件排出位置的导件

2.5.4 电气配线

2.5.4.1 供电注意事项

下面说明向机器人控制器、CV2、送料器与料斗供电的注意事项。
请根据“JIS B 9960-1(IEC 60204-1)机械类的安全性-机械的电气装置- 5.1 输入电源导体的连接”，将客户设计机械的电气装置连接至单一电源。要在装置的特定部分（比如，送料器或料斗）使用与输入电源不同的电源时，请通过机械的电气装置内的变压器或转换器等进行供电。

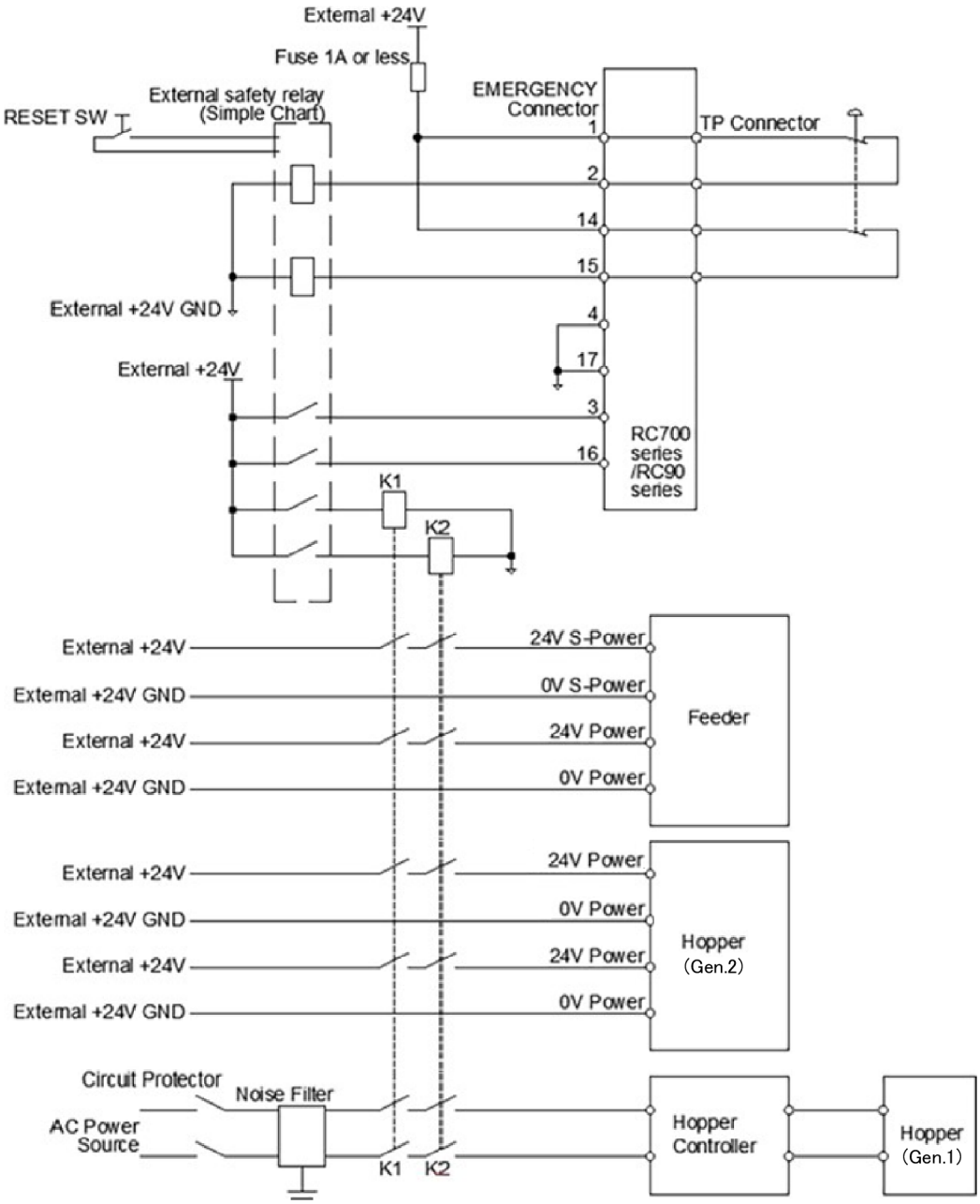
如下所述为连接单一AC200V电源时，需要客户设计的电气装置的概略示例。有关详细信息，请依据“JIS B 9960-1(IEC 60204-1)机械类的安全性-机械的电气装置”。



2.5.4.2 向送料器供电

有关配线的详细信息，请参阅以下手册。
“Epson RC+ 8.0选件 Part Feeding 8.0 IF-***篇 - 电缆连接”
***：送料器机型名称（IF-80、IF-240、IF-380、IF-530）
“Epson RC+ 8.0选件 Part Feeding 8.0 IF-A1520 & IF-A2330篇 - 电气规格”

设计电路时，请参考控制器手册中的“EMERGENCY”-“电路图与配线示例”，确保按下紧急停止开关时，会通过外部安全继电器将送料器与料斗电源置为OFF。下图所示为概略参考电路图。



2.5.4.3 向料斗供电

有关配线的详细信息，请参阅以下手册。
“Epson RC+ 8.0选件 Part Feeding 8.0 Hopper篇”

2.5.4.4 对机器人进行配线

根据控制器手册，对各机器人进行配线。

2.5.4.5 对相机进行配线

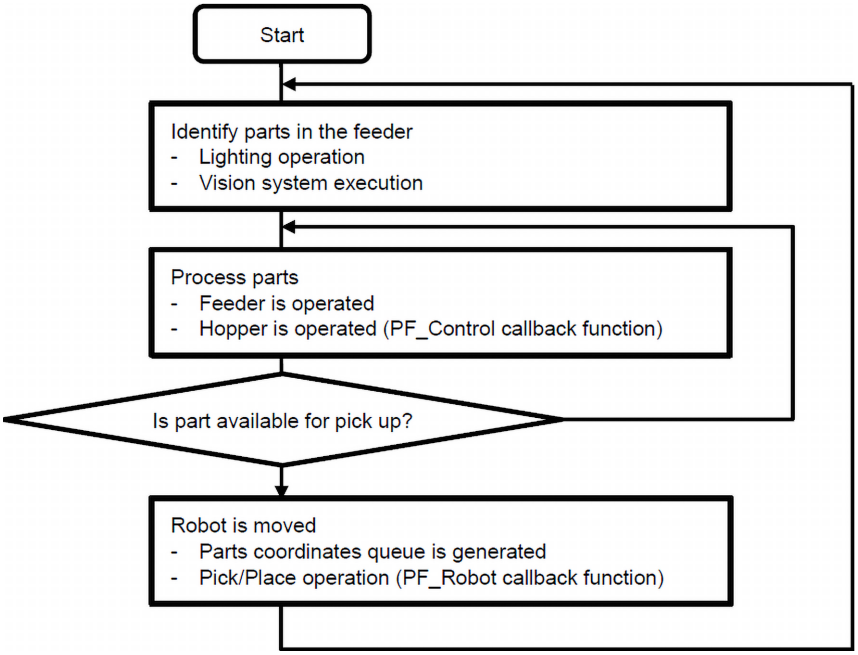
有关配线的详细信息，请参阅以下手册。
“Epson RC+ 8.0选件 Vision Guide 硬件篇”

2.6 运作概述

下面说明Part Feeding选件的运作概述。

2.6.1 Part Feeding进程

Part Feeding进程是指，自动进行视觉系统与送料器的控制并内置于Part Feeding系统中的进程运作。
要开始Part Feeding进程时，需要通过客户程序执行PF_Start命令。
如下所述为Part Feeding进程的内容。



1. 识别送料器上的部件
   通过视觉系统识别平台上部件的数量或分布。
2. 处理部件
   为便于机器人抓取部件，控制送料器以移动部件。部件数量较少或较多时，调用PF_Control回调函数，通过料斗供给部件。
3. 移动机器人
   生成部件坐标队列（送料器上的部件坐标列表）。调用PF_Robot回调函数，进行部件的取放动作。

Part Feeding进程因客户程序调用PF_Stop命令而停止。

2.6.2 向送料器供给部件

按下述方法向送料器供给部件。

- 使用料斗
- 人工进给

2.6.2.1 部件供给数

向平台的部件供给数是确定运作的循环时间的重要因素。

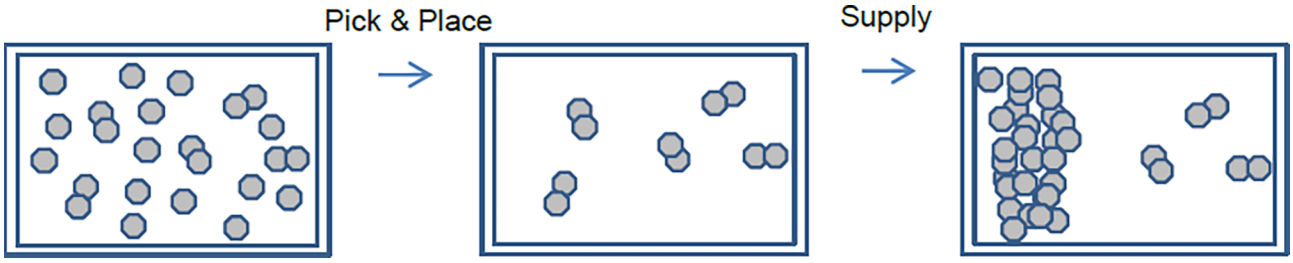
- 部件数过多时：
  会因发生部件重叠而必须进行好几次送料器运作而导致循环时间过低。
- 部件数过少时：
  需要多次向平台供给部件。导致循环时间过低。

有适量的供给数（送料器运作后平台上的部件数）。可通过送料器校准求出该数量。

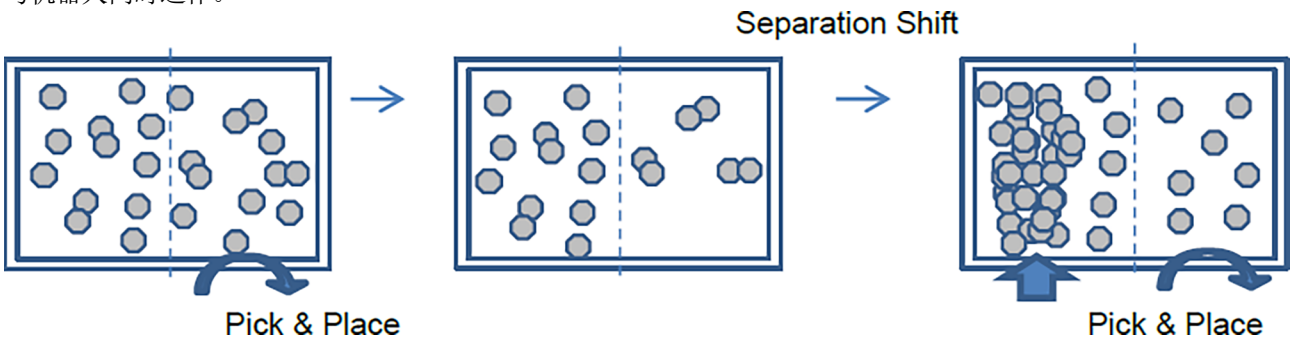
向送料器供给部件的方式（时序）包括以下3种。

1. 隔开进给
   是指将送料器上的所有部件隔开后供给部件。
   由于送料器上部件的滞留时间几乎固定，因此，对于对振动敏感（不耐振动）的部件，建议采用这种方式。但由于机器人通过1次送料器运作可获取的平均部件数较少，因此循环时间会延长。

2. 添加进给
   按送料器上可获取部件为零的时序，追加供给部件。
   由于机器人通过1次送料器运作可获取的平均部件数相对较多，因此循环时间会缩短，提高了生产率。但由于部件的滞留时间相对较长，因此这种方式不适合于对振动敏感的部件。



3. 并行进给
   与部件拾取位置指定功能组合使用这种方式。
   在机器人拾取部件期间，将部件追加到部件拾取位置的相反区域。由于机器人通过1次送料器运作可获取的平均部件数较多，因此循环时间会缩短。另外，可同时进行料斗与机器人的运作，因此循环时间会进一步缩短。但由于部件会在送料器上滞留较长的时间，因此这种方式不适合于对振动敏感的部件。另外，需要客户记述程序，以确保料斗与机器人同时运作。

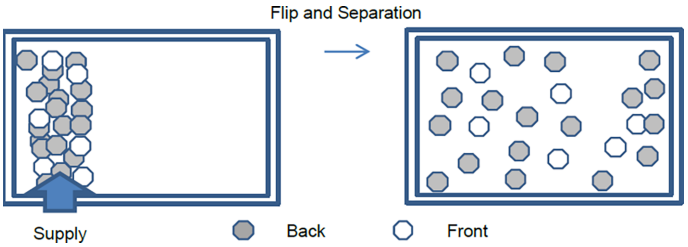


2.6.3 送料器运作

有Part Feeding选件时，会根据送料器上的部件状态自动选择并执行送料器运作。这样的话便于机器人抓取部件。
如下所述为送料器运作情况。说明图为概述。可能会与实际运作不同。

2.6.3.1 翻转与分离

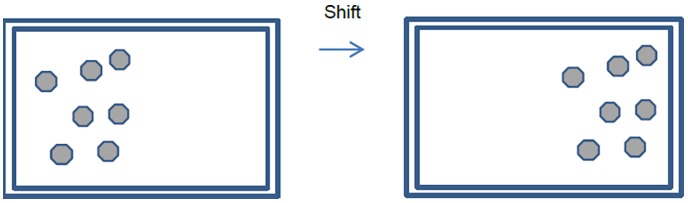
使部件均匀地分散在平台上。通过适当地空开部件之间的间隔，以便于机器人抓取部件。



分离之前，可能会加入将部件靠向中心的动作。这种动作被称之为定芯。

2.6.3.2 移动

在保持部件间隔（分布）的状态下使部件全体向某一方向移动。
通过将部件靠近放置位置，可缩短机器人的移动距离，以提高循环时间。



移动包括向前（靠近拾取位置的方向）与向后（远离拾取位置的方向）。

2.6.4 平台上的部件拾取位置

机器人拾取部件的位置包括“全面拾取”与“部分拾取”2种。
部分拾取与全面拾取究竟哪种方式效率高，这取决于客户部件、末端夹具与料斗等装置构成。请实际让装置进行运作，获取日志后进行确认。

2.6.4.1 全面拾取

以平台的整个面为对象进行拾取。
部件大于平台尺寸时（IF-240时，大致标准为大于等于2平方厘米），选择全面拾取。

2.6.4.2 部分拾取

以接近放置位置的区域为对象进行拾取。
采用这种方式时，会根据部件分布状况，自动进行将部件移动至拾取对象区域的操作。另外，可在机器人运作的同时向非拾取对象区域进给部件。（需要客户记述程序。）通过运用这些功能，通常相较于全面拾取，更能缩短机器人的循环时间。

2.6.5 避免末端夹具与平台之间产生干扰

为了防止末端夹具与平台之间产生物理干扰，需要将送料器上的部件可拾取范围设定在平台外圈之内。可利用Part Feeding选件简单地指定该距离。

2.7 部件

下面说明Part Feeding选件可使用的部件。
每个项目最多可注册32个部件。

要点

销售商具备用于评估客户部件是否符合Part Feeding选件的体制。有关详细信息，请咨询当地销售商。

2.7.1 可处理部件的条件

在送料器可处理的部件方面存在下述条件。

2.7.1.1 视觉符合性

需要能通过视觉系统正确地识别部件。

- 由透明树脂成型的部件可能会因透光而无法识别其形状。在这种情况下，通过将照明更改为非可见光或使用反射照明，有时可解决这种问题。
- 识别部件的表里时，可能会因部件的形状而无法识别表里。在这种情况下，通过追加反射照明，有时可解决这种问题。

2.7.1.2 大小与重量

部件越大，进入平台的部件数量（可铺满而非重叠的数量）越少。该数量较小时，送料器的运作次数会增加，并且机器人运作时间也会相对减少，因此会导致循环时间过低。作为部件数量的大致标准，最好在送料器中放入大于等于50个的部件。
部件的总重量（1个部件的重量×可不重叠进入平台的部件数）需小于等于送料器的可搬重量。如果超过该重量，则可能会因送料器过载而导致部件分离能力过低、循环时间过低或送料器的使用寿命缩短。
有关送料器的可搬重量，请参阅各送料器手册。

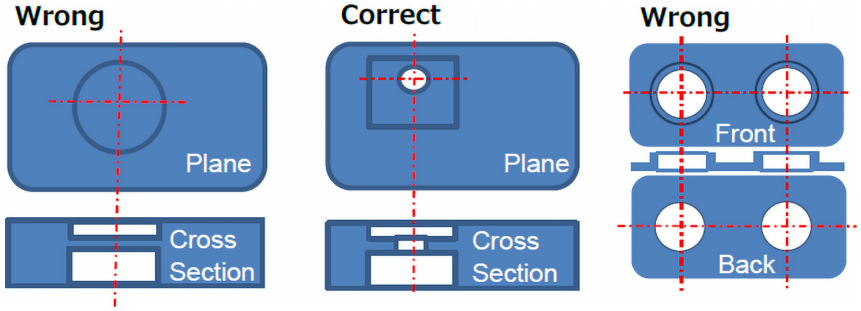
2.7.1.3 部件的材质与状态

以下部件不适合于Part Feeding选件。

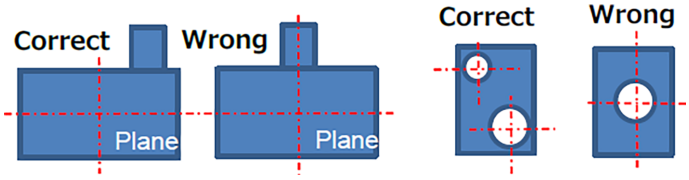
- 柔性材质或较轻材质的部件
  例：纸、纤维质地物品
- 会因振动而导致损坏/变形或摩擦后产生粉尘的部件
  例：粉末固化物品、涂装物品
- 粘性物品、泄漏液体的部件
  例：食品

2.7.1.4 部件的形状等

- 球形部件不会在送料器上静止不动，难以拾取。请使用防滚动平台选件。
  例：轴承钢珠
- 易于缠绕的部件难以分离。
  例：盘簧
- 在透光状态下，截面方向上形状不同、不透光材质的部件可能会无法识别表里。如下所述为这种部件的示例。

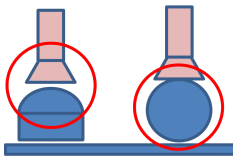


无法分选截面方向上形状不同的部件

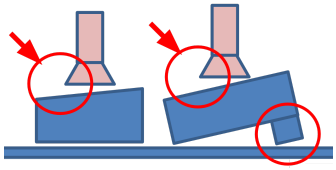


平面上的外形形状不同时，可进行表里分选

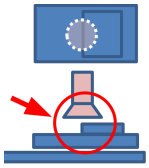
- 如果采用吸附方式进行拾取，拾取部件时，部件吸附面则会有平行于送料器底面的面，并且吸附垫等会产生垂直下降，因此，建议使用可充分确保吸附垫面积的部件。如下所述为这种部件的示例。



球体（Ball）

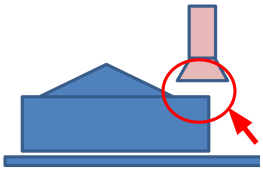


送料器底面与吸附面不平行的部件

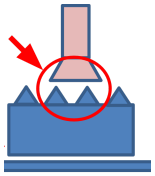


吸附位置有高差时

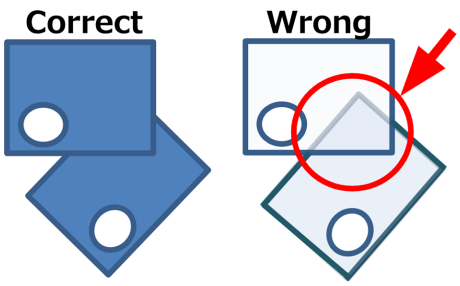
- 采用吸附方式进行进给时，请确保重心没有高差。如下所述为这种部件的示例。



无法确保足够吸附面积的部件



无法确保平滑面的部件（采用吸附方式进行进给时）



从下部透光的部件

- 要在组装工序中使用时，请采取在拾取后防止部件错位的措施。这种措施就是在部件上设置导孔，另外，在末端夹具侧设置销以控制错位。其他措施包括设置向上固定相机，并对拾取后的部件进行定位。

2.7.2 部件示例

如下所述为Part Feeding选件可使用的部件示例。
No.1～3的部件适合于IF-240及IF-A1520。No.4与5因相对于IF-240及IF-A1520来说过大或过重而不适合。No.6与7因相对于IF-240、IF-380、IF-530、IF-A1520与IF-A2330来说过小或过轻而不适合。

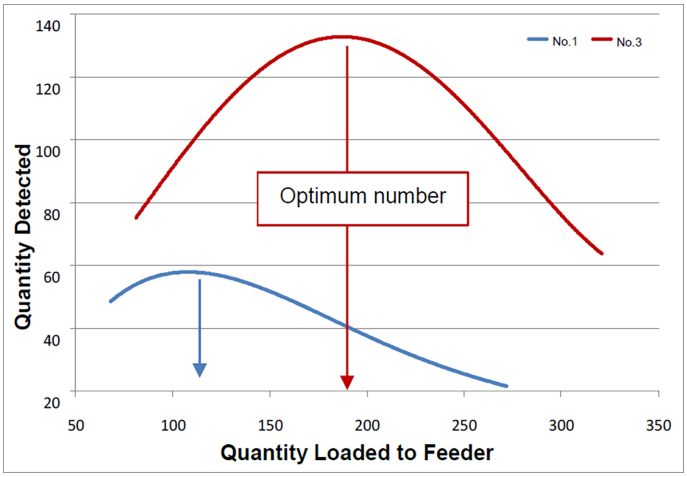
No	图	特质	尺寸[mm]	重量[g]	注释
1		金属压制部件	10 × 10 × 0.2	0.088	适合于IF-240、IF-A1520
2		金属压制部件	11 × 5.5 × 0.2	0.029	适合于IF-240、IF-A1520
3		树脂部件	10 × 9 × 2.1	0.127	适合于IF-240、IF-A1520
4		尼龙连接器	21 × 29.9 × 21	7.1	适合于IF-380、IF-A2330
5		高螺母	36 × 11 × 9.5	14	适合于IF-380、IF-530、IF-A2330
6		IC	5 × 4.4 × 1.5	0.082	适合于IF-80
7		金属衬件	ø4 × 1	0.102	适合于IF-80

仅利用背光灯无法识别No.1与2的表里。
仅利用背光灯即可识别No.3的表里。

2.7.2.1 向送料器的投放数量与图像处理检测数量的关系

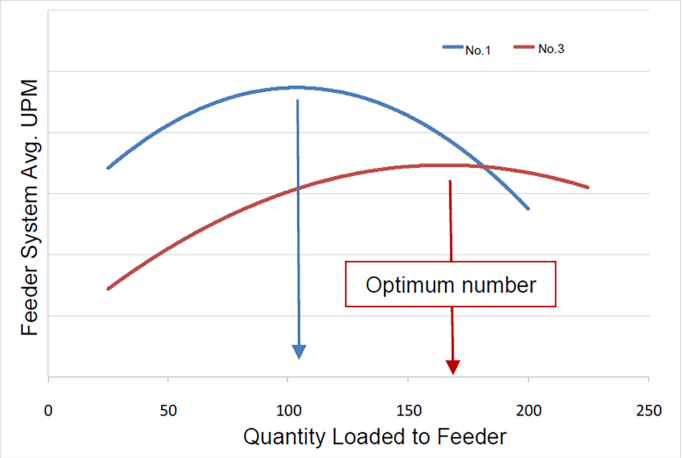
向送料器的部件投放数量与图像处理检测数量存在向上凸起的关系。
根本无法检测送料器上接触相邻部件的部件或重叠投放的部件。与相邻部件的易接触性、易重叠性因部件而异，图形的形状也不同。有Part Feeding选件时，请根据销售商的实验，通过校准求出最佳部件投放数量。

如下所示为部件No.1与3的向送料器投放数量与图像处理检测数量的关系图。
请关注以下事项：无法检测已投放到送料器上的所有部件；检测数量因投放数量而异。



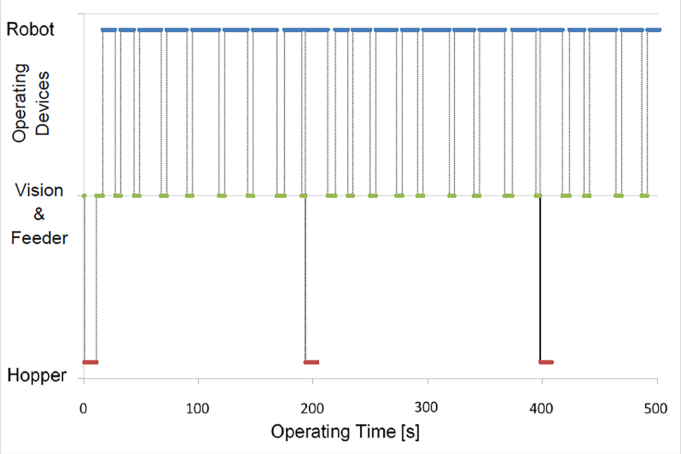
2.7.2.2 向送料器投放数量与平均UPM（Unit Per Minute）的关系

将进行某段时间拾取动作时的单位时间的平均拾取数量设为平均UPM。
如下所示为向部件No.1与3的送料器投放数量与平均UPM的关系图。部件No.1与3均为向上凸起的图形。平均UPM因机器人的速度、加速度、行进距离而异，因此并未显示纵轴数值。除机器人的运作条件以外，请注意：UPM会因向送料器的部件投放数量而异；存在向增大UPM的最佳送料器的部件投放数量。



2.7.2.3 送料器运作与UPM（Unit Per Minute）的关系

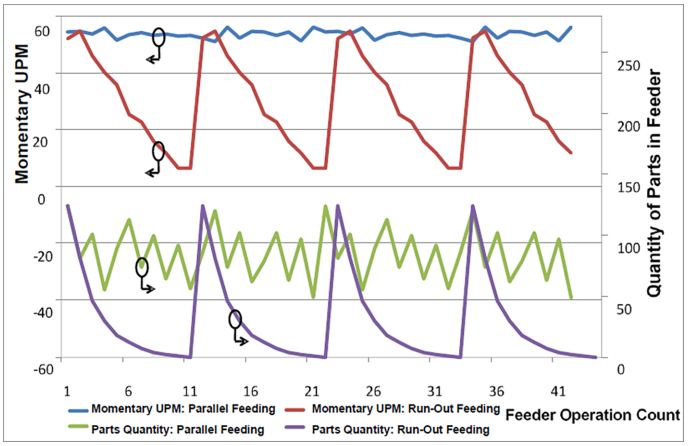
以时间为横轴，将要运作的设备设为机器人、视觉送料器与料斗。如果绘制各设备的运作时序，则如下图所示。开始运作时，通过视觉送料器运作检测送料器上没有部件，然后开动料斗，将部件投放到送料器上。



接下来，送料器运作，使部件分散开来，然后通过Vision进行部件检测，机器人进行拾取动作。如果没有可拾取的部件，则再次通过视觉送料器的运作分散与检测部件。接着再次进行机器人的运作。
如果重复进行视觉送料器与机器人的运作，送料器上的部件则会逐渐减少。根据设定的阈值，按照料斗运作时序来进行料斗运作，投放部件。如下所示为以并行进给运作为前提的图形。
重复进行视觉送料器与机器人的运作，因此视觉送料器运作瞬间的UPM会变为“0”。机器人运作瞬间的UPM变为大于“向送料器投放数量与平均UPM（Unit Per Minute）的关系”中所示的平均UPM的值。请注意，在平均UPM方面，如果视觉送料器运作瞬间的UPM=0，则会变为机器人运作瞬间的UPM平均时间。
另外，图中所示的机器人运作蓝线长度并不固定。因为可拾取的部件数量因送料器上的部件分散状况而异；另外，重复进行拾取动作后，送料器上的部件数量减少，可拾取的部件也随之减少。
为了进行稳定的部件供给，需要尽可能将送料器上的部件数量保持恒定。

2.7.2.4 送料器上部件数量与料斗运作的关系

如下所示为，为了进行稳定的部件供给，通过料斗进行隔开供给时，以及将最佳投放数量设为180，将料斗投放数量设为90并进行并行进给时，各送料器运作次数的瞬时UPM（Unit Per Minute）与送料器上部件数量的图形。



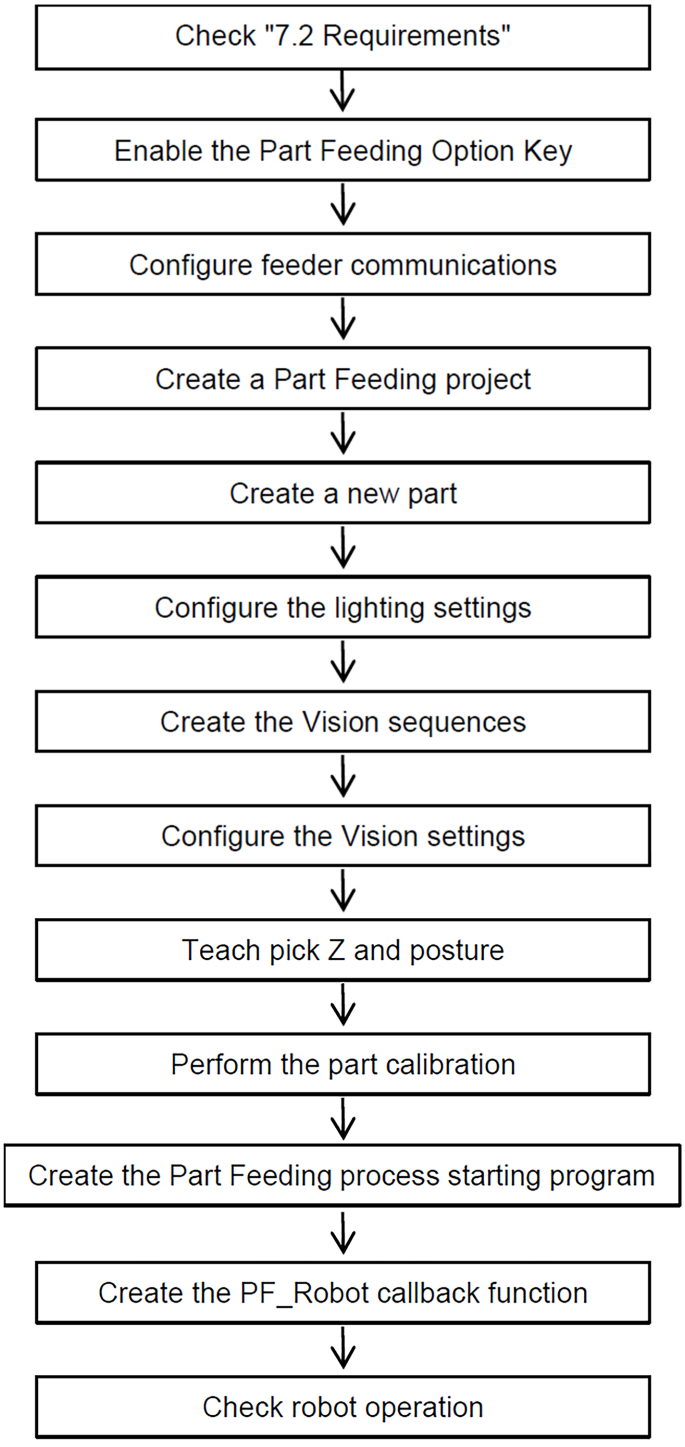
隔开进给时，如果未隔开部件，则无法通过料斗进行进给，瞬时UPM会逐渐下降；如果在部件为零后通过料斗供给部件，瞬时UPM也会复原。
并行进给时，料斗每进行2～4次运作，料斗就会进行1次运作，因此，送料器上的部件数量也不会低于下限值，这样的话，瞬时UPM的波动也会减小。

2.8 试着使用

试着使用一下Part Feeding选件，构建取放部件的系统。

2.8.1 作业流程

如下所述为作业进程。

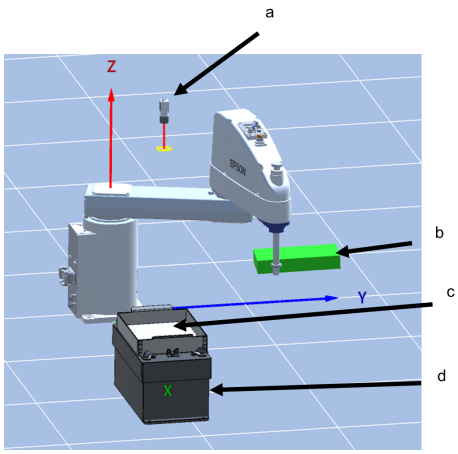


2.8.2 前提条件

2.8.2.1 装置构成

- 机械手使用SCARA型机器人。
  即使是6轴机器人，作业内容也相同。
  应连接适合于部件的末端夹具。
- 相机为向下固定相机。
- 使用送料器的背光灯。

- 不使用料斗。



符号	项目
a	相机（向下固定）
b	机械手
c	部件放置场所（托盘等）
d	送料器

2.8.2.2 连接与调整。

请确认以下事项。

- Epson RC+应被连接至控制器
- 机械手应被连接至控制器
- 送料器应被连接至控制器
- Vision Guide（PV或CV）应被连接至控制器
- 机械手、相机与送料器应被正确设置
- 应完成相机位置调整、焦点调整与亮度调整

2.8.2.3 部件

- 部件ID为“1”。
- 是部件形状没有表里之分的部件。利用视觉的Blob对象（通过面积值检测）进行检测。

2.8.2.4 设置

请确认以下事项。

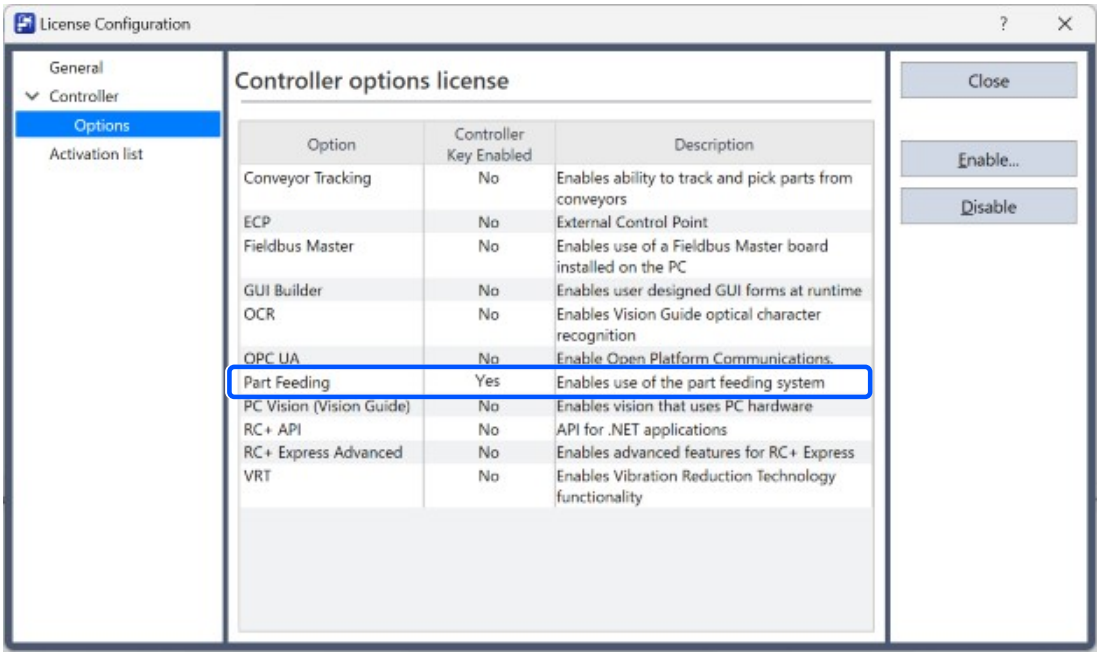
- 机械手应被注册到系统配置中
- 应完成视觉校准
- 应完成工具坐标系的设置

2.8.2.5 其他

错误处理是指执行回调函数的模板代码中记述的内容。

2.8.3 启用Part Feeding选件

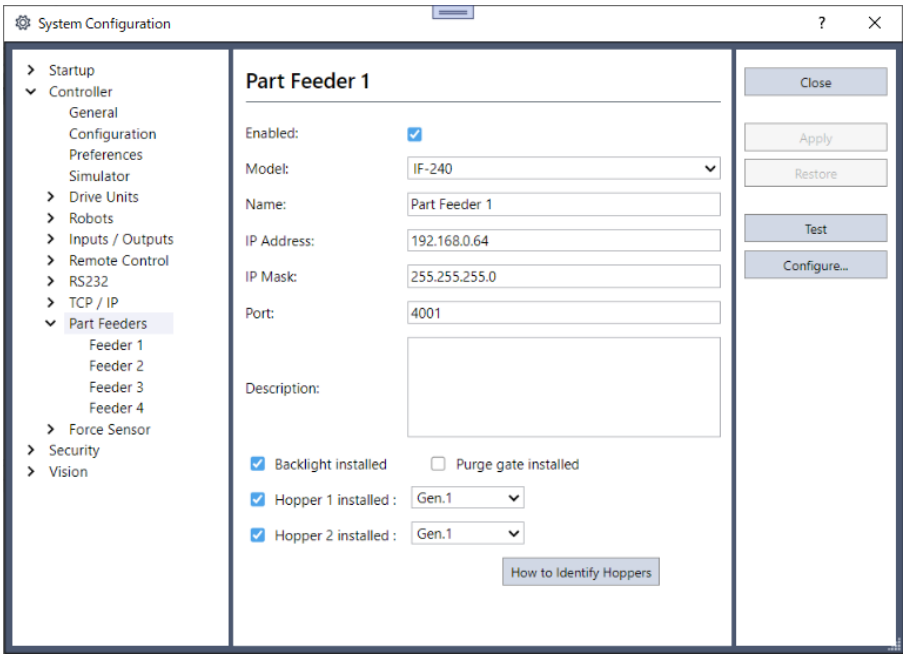
要使用Part Feeding功能时，需要启用该选件。
如果连接控制器，并从Epson RC+ 8.0菜单 - [设置] - [许可证设置]画面的树形图中选择[控制器] - [选件]，则会显示下述画面。



选件启用步骤因控制器的型号而异。有关详细信息，请参阅以下手册。
“Epson RC+ 8.0用户指南 - 安装控制器许可证”

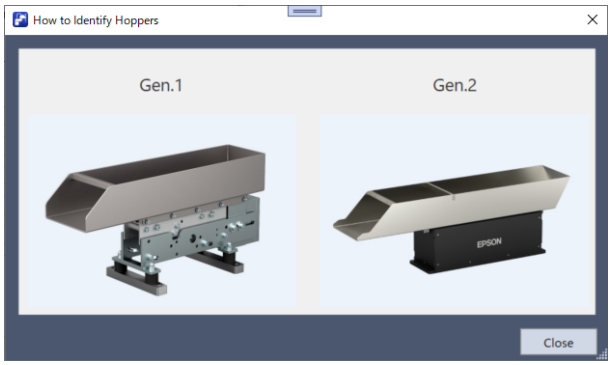
2.8.4 进行送料器的初始设定

1. 选择Epson RC+ 8.0菜单 - [设置] - [系统配置]。
2. 选择树[控制器] - [零件送料器] - [送料器1]。



3. 设定以下项目。

项目	描述
启用	要启用送料器时，勾选复选框。

项目	描述
型号	选择送料器型号。
送料器名称	设定任意名称。（半角字母数字与下划线。最多32个字符）
IP地址	输入当前送料器中设定的IP地址。 默认IP地址为192.168.0.64。
子网掩码	输入当前送料器中设定的子网掩码。 默认子网掩码为255.255.255.0。
端口	输入当前送料器中设定的端口编号。 默认端口编号为4001。
注释	填写送料器的说明（注释）。属于选件。（半角字母数字与下划线。最多32个字符）
安装背光灯	已安装送料器内置背光灯时，勾选复选框。
安装清除门	使用送料器选件的清除门时，勾选复选框。（仅限于IF-240，IF-380，IF-530、IF-A1520、IF-A2330）
安装料斗1 安装料斗2	连接料斗时，勾选复选框。从Gen.1/Gen.2中选择料斗类型。有关料斗的类型，请参阅以下内容。 料斗
料斗的区分方法	显示料斗的类型。 

4. 设定结束后，单击[应用]按钮。

要点

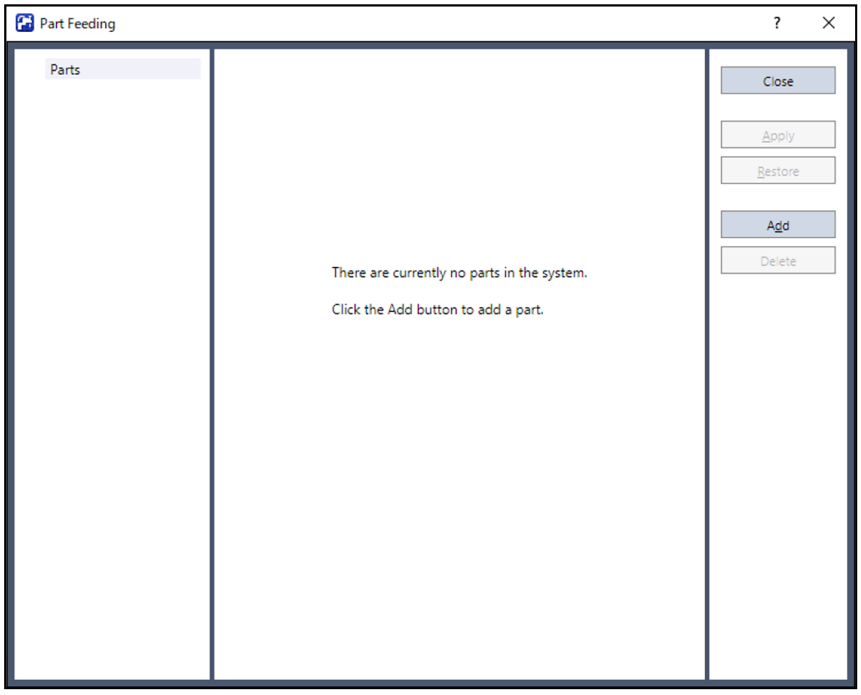
- 要更改送料器的IP地址时，请参阅以下内容。
  零件送料器画面
- T/VT系列控制器时，可设定4台部分的送料器，但可同时控制的送料器最多为2台。
- 根据要使用的送料器与料斗，存在自动设定的项目。有关详细信息，请参阅以下内容。
  操作入门

2.8.5 准备Part Feeding用项目

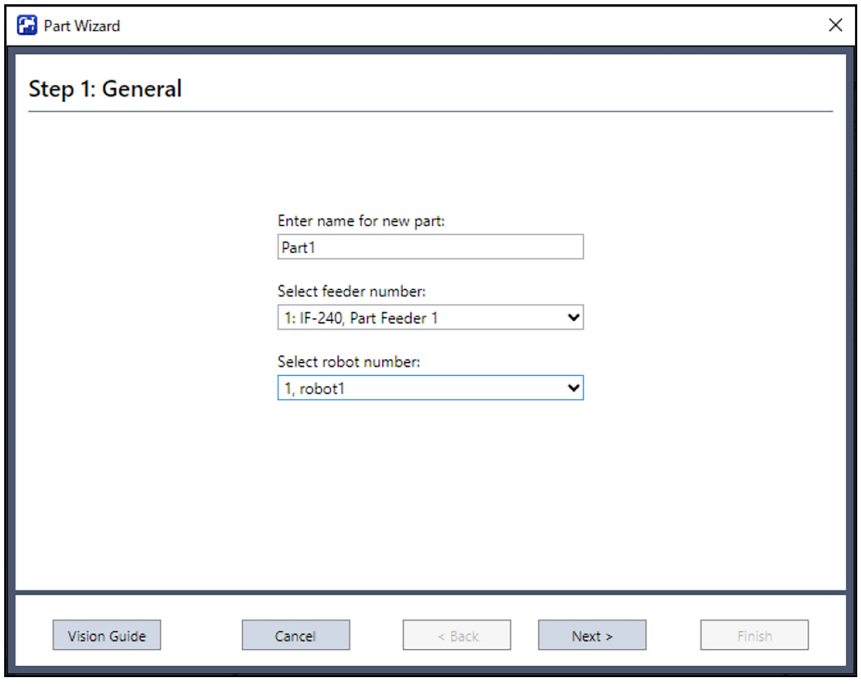
选择Epson RC+ 8.0菜单 - [项目] - [新建项目]，新建项目。
或打开现有项目，创建副本。

2.8.6 新建部件

1. 选择Epson RC+ 8.0菜单 - [工具] - [上料]。单击[添加]按钮。



2. 启动部件向导。



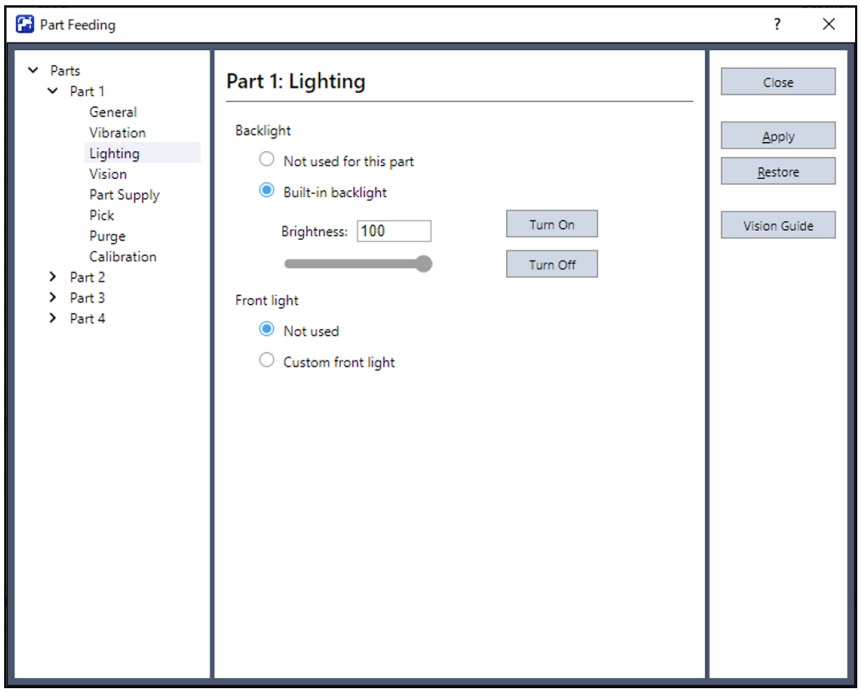
3. 根据部件向导设定部件。有关详细的设定内容，请参阅以下内容。
   部件向导
   向导用于进行基本设定。在本教程中，退出向导并完成设定。

提示

程序文件PartFeeding.prg与包含文件PartFeeding.inc被添加至项目中。

2.8.7 进行照明设定

1. 选择树 - [照明]。



2. 单击[On]按钮，确认送料器的背光灯点亮。

2.8.8 创建视觉序列

创建2个视觉序列。
除部件检测用以外，也创建送料器校准用视觉序列。

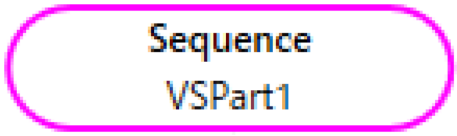
2.8.8.1 创建部件检测用视觉序列

创建用于检测部件的视觉序列。

1. 单击[Vision Guide]按钮，显示视觉引导画面。

2. 将1个部件放到送料器上。

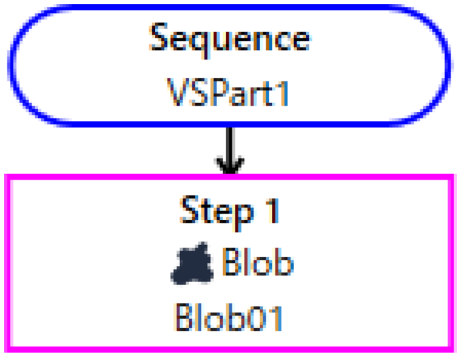
3. 新建1个视觉序列。按如下所述设定属性。

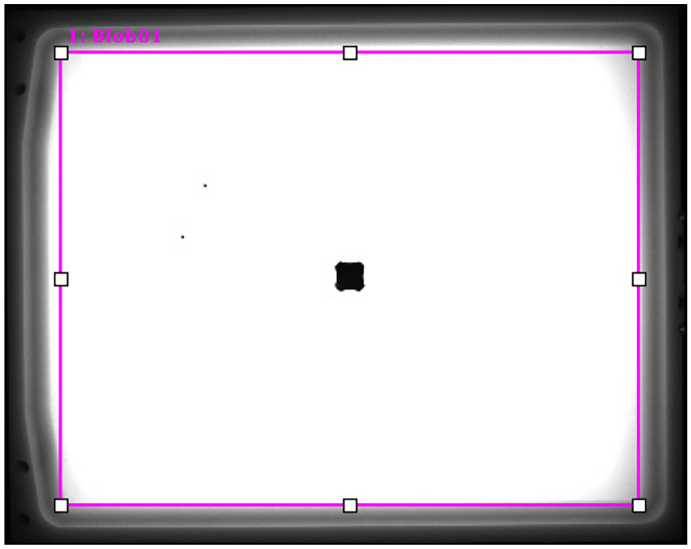


属性	设定方法
Name	设定任意名称。（例：VS_Part1）
Calibration	设定视觉校准。

属性	设定方法
ExposureTime	设定时注意以下事项。 - 应可清晰地识别部件 - 平台周围与中央的亮度应基本相同

4. 添加1个Blob对象。按如下所述设定属性。



属性	设定方法
SearchWindow	将平台全体设为检测区域。 
NumberToFind	设为“All”。
MaxArea	设为约为部件面积值的1.3倍。
MinArea	设为约为部件面积值的0.7倍。
ThresholdHigh	进行可以可靠地检测部件的设定。进行免于错误地检测平台周边的黑暗部分的设定。

提示

无法正确地检测部件时，也请重新调整除此之外的属性或视觉序列的属性。

5. 设定结束后，单击[Run]按钮，确认可正确地识别部件以及不会错误地检测背景。
6. 设定结束后，单击Vision Guide菜单 - [File] - [Save]按钮。设定被保存。

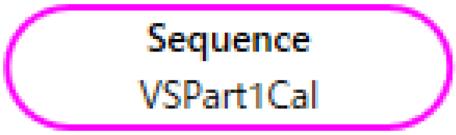
提示

有关创建部件检测用视觉序列的详细信息，请参阅以下内容。

视觉校准

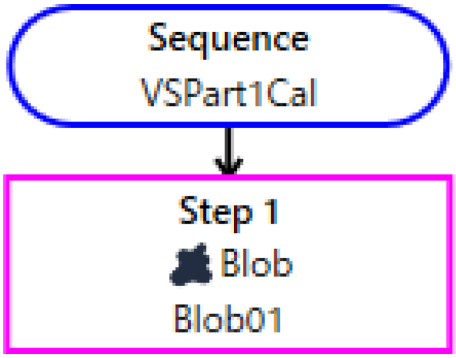
2.8.8.2 创建送料器校准用视觉序列

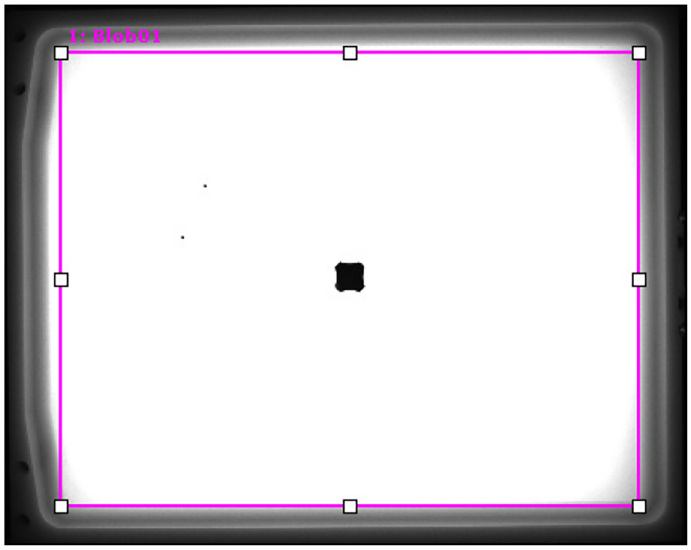
1. 新建1个视觉序列。按如下所述设定属性。



属性	设定方法
Name	设定任意名称。（例：VS_Part1_Cal）
Calibration	设定视觉校准。
ExposureTime	进行可以清晰地识别部件的设定。

2. 添加1个Blob对象。按如下所述设定属性。



属性	设定方法
SearchWindow	将平台全体设为检测区域。 
MaxArea	保持默认值。
MinArea	设为约为部件面积的0.9倍的值。按以下步骤调查部件面积。 1. 点亮送料器的背光灯 2. 将数个部件不重叠地放在平台上 3. 执行视觉序列 4. 将Blob结果的Area平均值设为部件面积

NumberToFind	设为“All”。
ThresholdHigh	进行可以可靠地检测部件的设定。进行免于错误地检测平台周边的黑暗部分的设定。

提示

无法正确地检测部件时，也请重新调整除此之外的属性或视觉序列的属性。

送料器上有黑暗部分或脏污等难以检测的部分时，可利用“检测掩膜”从检测对象中排除。有关详细信息，请参阅以下内容。

程序示例 8.3

3. 设定结束后，单击[Run]按钮，确认可正确地识别部件以及不会错误地检测背景。
4. 设定结束后，单击Vision Guide菜单 - [File] - [Save]按钮。设定被保存。
5. 关闭Vision Guide画面。

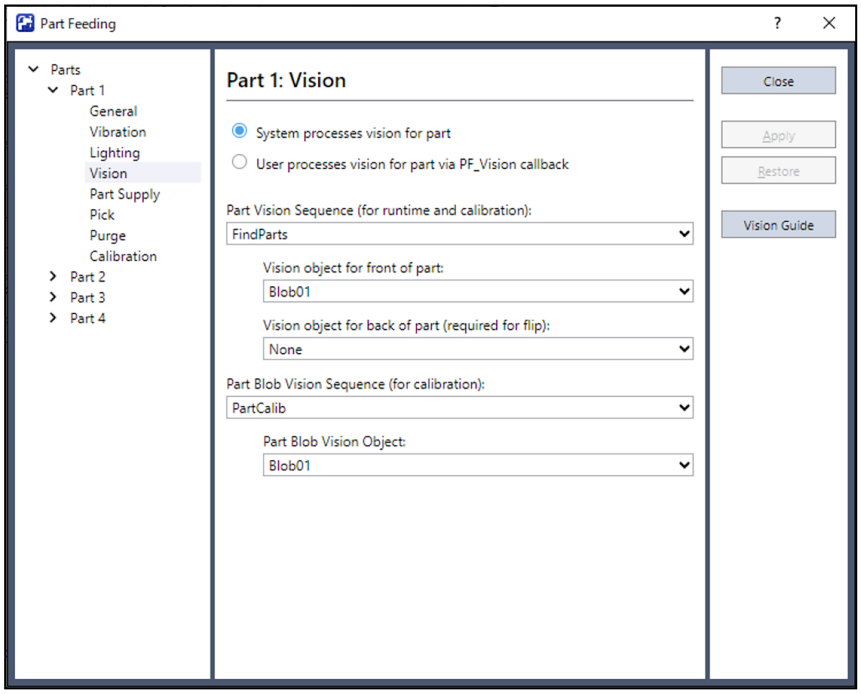
提示

有关创建送料器校准用视觉序列的详细信息，请参阅以下内容。

视觉校准

2.8.9 进行视觉设定

1. 选择树中的[视觉]。



2. 设定以下项目。

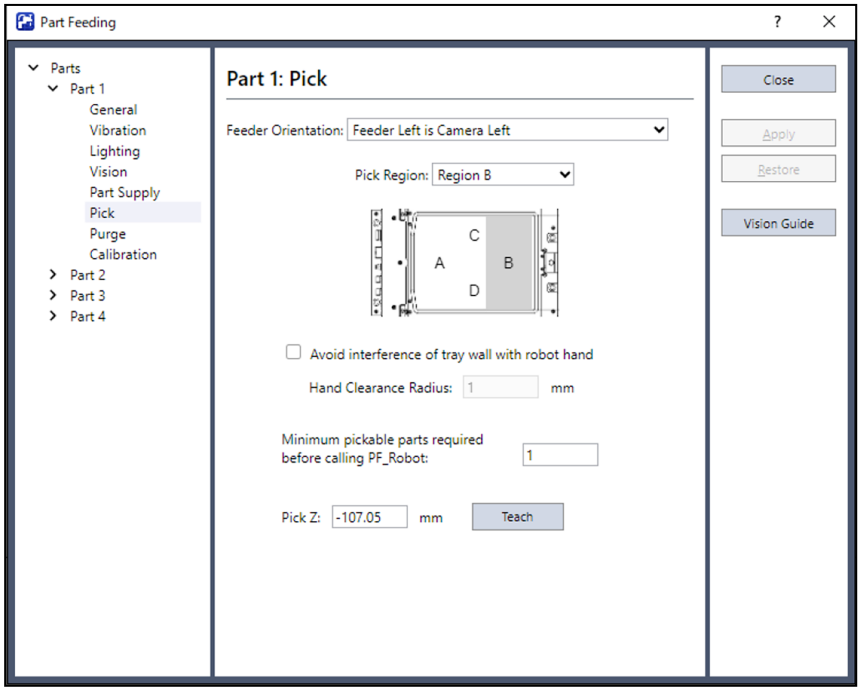
项目	设定方法
部件视觉序列	指定下述已创建的视觉序列。 创建视觉序列

项目	设定方法
表面部件的视觉对象	指定下述已创建的视觉序列中包含的Blob对象。 创建视觉序列
部件Blob视觉序列	指定下述已创建的视觉序列。 创建视觉序列
部件Blob视觉对象	指定下述已创建的视觉序列中包含的Blob对象。 创建视觉序列

3. 单击[应用]按钮。

2.8.10 进行拾取设定

1. 选择树中的[拾取]。



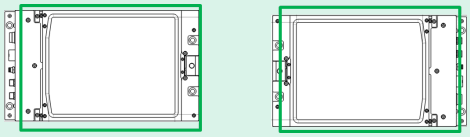
2. 设定以下项目。

项目	设定方法
送料器的方向	选择用相机查看时的送料器设置方向。请正确地进行设定，以确保送料器的方向对准用相机查看时的方向。
拾取区域	选择接近放置位置的位置。 此时选择B。 可选择的项目因送料器而异。

3. 单击[示教]按钮。

提示

送料器的方向可选择某个方向。



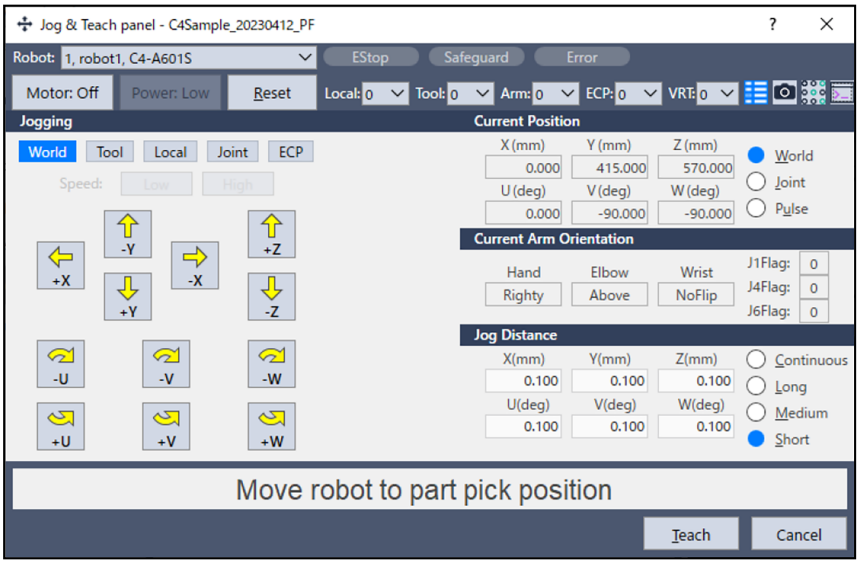
在拾取区域方面，请选择A～D中接近放置位置的位置。

可从A～D的4处与全体中选择1处IF-240、IF-A1520、IF-A2330的拾取区域。

可从A～B的2处与全体中选择1处IF-380、IF-530的拾取区域。

仅可从全体中选择IF-80的拾取区域。

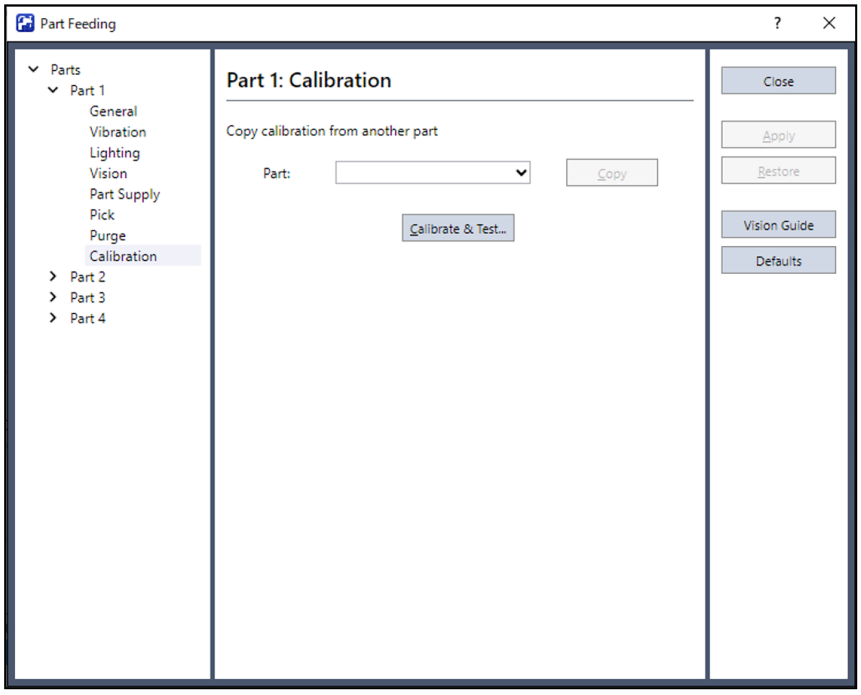
2.8.11 进行拾取Z坐标与姿势的示教



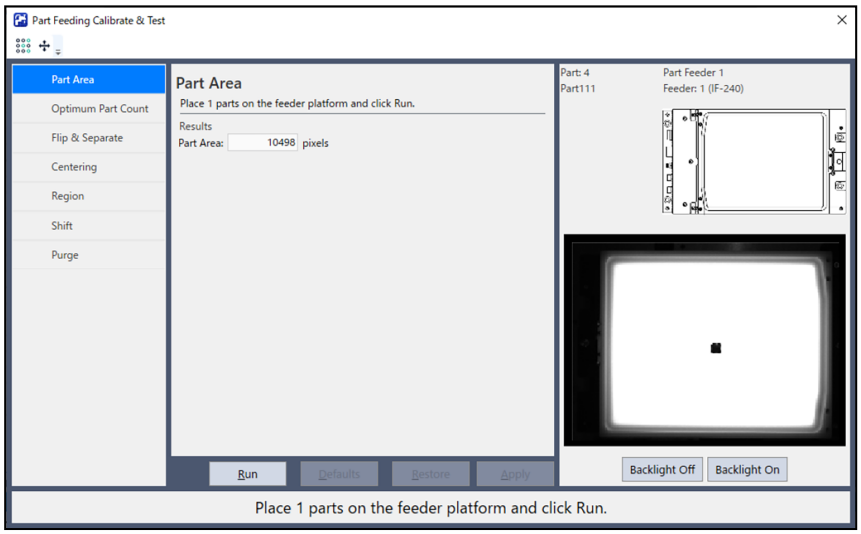
1. 将1个部件放到送料器上。
2. 通过步进操作等移动机器人，使末端夹具接触送料器上的部件。
   为卡盘机构的末端夹具时，用于抓取部件。
3. 单击[OK]按钮。
   Z坐标被保存。
4. 单击[拾取]对话框中的[应用]按钮。

2.8.12 校准&测试

1. 选择树中的[校准]。
   单击[校准&测试]按钮。



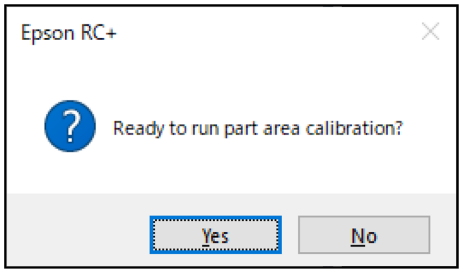
2. 显示[校准&测试]画面。
   选择[部件区域]选项卡。



3. 将1个部件放到平台中央。

4. 单击[执行]按钮。

5. 显示以下消息。
   单击[是]按钮。



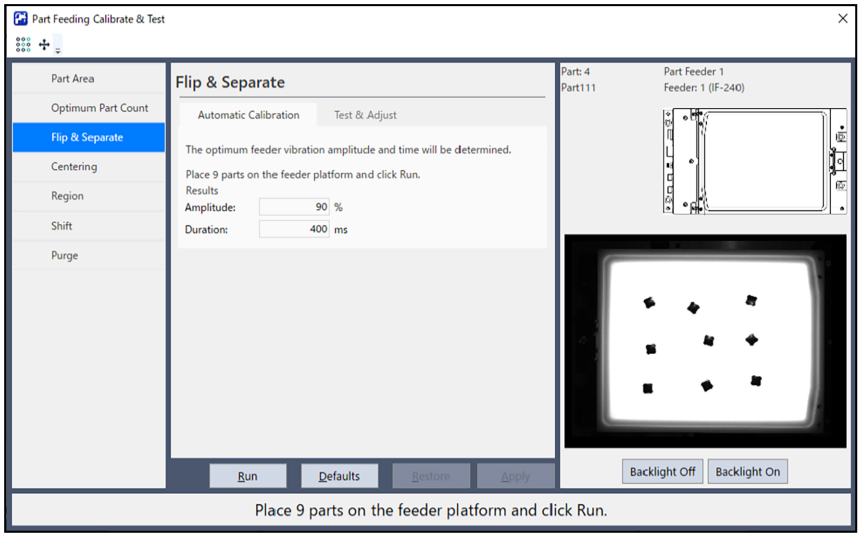
6. 确认“部件面积”中显示数值。

7. 单击[应用]按钮。

要点

下述（8）[翻转&分离]、(14) [区域]的校准并非必须项目，但建议执行。

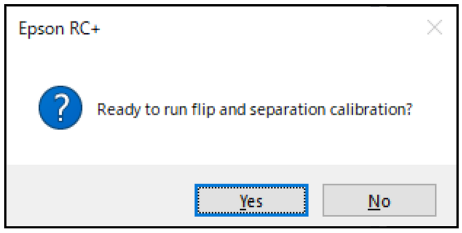
8. 选择[翻转&分离]选项卡。



9. 将显示数量的部件放到平台上。

10. 单击[执行]按钮。

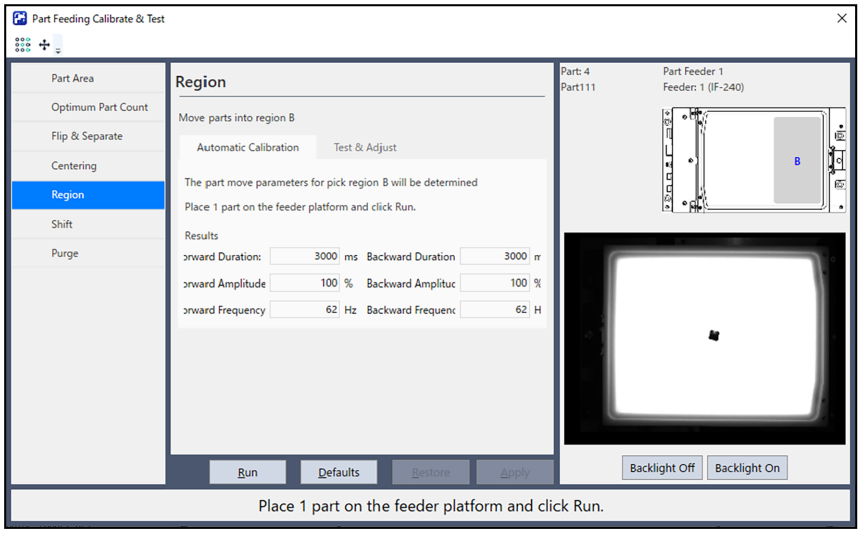
11. 显示以下消息。
    单击[是]按钮。



12. 尝试几次送料器振动与视觉处理后，确认各[结果]值已被更新。

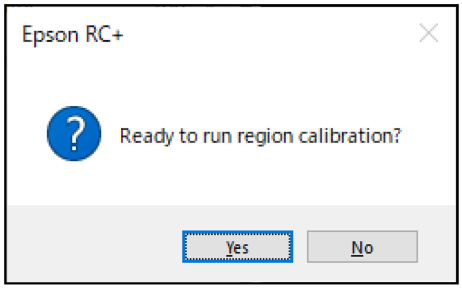
13. 单击[应用]按钮。

14. 选择[区域]选项卡。



15. 将投放部件数量设为1个。

16. 显示以下消息。
单击[执行]按钮。



17. 尝试几次送料器振动与视觉处理后，确认各[结果]值已被更新。

18. 单击[应用]按钮。

2.8.13 创建Part Feeding进程开始程序

Part Feeding进程开始程序用于记述进行机器人初始化并开始部件拾取与放置的处理。如下所述为记述内容的具体示例。

1. 切换为用于拾取&放置部件的机器人。
2. 将机器人的电机置为On。
3. 设定机器人的速度、加速度与功率模式等。
使用Speed、Accel、Power等。
设定LimZ时，请考虑Z坐标、平台深度（28mm）、越过平台时的Z方向余量以及设置在送料器周边的单元的Z方向高度。
Z坐标为下述已示教的坐标。
进行拾取Z坐标与姿势的示教
4. 将机器人移动至可进行视觉拍摄的位置。
（为固定向下相机时，移动至不影响拍摄的位置。）
使用Home等。
5. 进行客户装置的设备设定。
例：设定料斗、设定自定义照明等
6. 要获取日志时，执行PF_InitLog。

7. 指定部件ID，执行PF_Start。
   如果执行PF_Start，则会生成任务32，并立即将控制返回至调用侧。

如下所述为程序示例。下面在程序文件main.prg中创建以下函数。（省略5与6的记述）

```
Function test

  Robot 1
  Motor On
  Speed 100
  Accel 100, 100
  Power High
  LimZ -80.0
  Home

  PF_Start(1)

Fend
```

2.8.14 创建PF_Robot回调函数

在PF_Robot回调函数中记述机器人拾取与放置部件的运作。该函数按如下所述进行运作。（重复1～7。）

1. 从部件坐标队列中获取平台上的部件坐标。
   使用PF_QueGet。
2. 将机器人移至部件位置。
   使用Jump等。（为SCARA型机器人时）
3. 将吸附置为On等，然后抓取部件。
4. 将机器人移至部件放置位置。
   使用Jump等。（为SCARA型机器人时）
5. 将吸附置为Off等，然后释放部件。
6. 删除1个部件坐标队列中的数据。
   使用PF_QueRemove。
7. 确认是否产生停止指令。如果产生，则退出循环。
   使用PF_IsStopRequested。
8. 作为返回值，返回PF_Robot = PF_CALLBACK_SUCCESS。

如下所述为PF_Robot回调函数示例。
另外，由于自动生成的PartFeeding.prg中记述有循环处理与1、6、7的处理，因此会记述剩余处理。

如下所述为程序中使用的标签等。
IO标签： Chuck（部件吸附）、UnVacumm（部件释放）
点标签：PlacePos（放置坐标）

```
Function PF_Robot(partID As Integer) As Integer

    Do While PF_QueLen(partID) > 0

        ' Pick
        P0 = PF_QueGet(partID)
        Jump P0 ! Wait 0.1; Off UnVacumm !
        On Chuck
        Wait 0.1

        ' Place
        Jump PlacePos
        Off Chuck
```

```
        On UnVacumm
        Wait 0.1

        ' Deque
        PF_QueRemove partID

        ' Check Cycle stop
        If PF_IsStopRequested = True Then
            Exit Do
        EndIf

    Loop
    Off UnVacumm
    PF_Robot = PF_CALLBACK_SUCCESS

Fend
```

2.8.15 进行运作确认

开始主程序，确认动作。

要点

初次请以低输出与低速度进行机器人运作，充分确认机器人是否按预期运作。

1. 创建项目。
2. 启动运行窗口。
3. 执行test函数。
4. 确认机器人正确进行拾取与放置部件的运作。

用于执行部件拾取&放置的系统设置至此结束。

3. 软件篇

3.1 前言

下面说明Epson RC+ 8.0 Part Feeding 8.0软件概述。

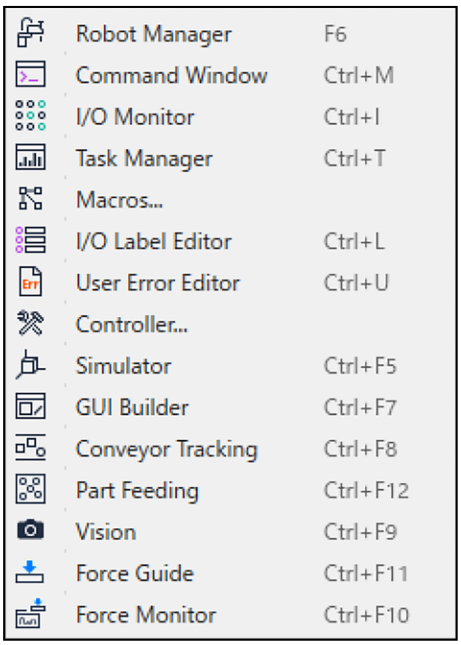
3.1.1 Part Feeding软件构成

Part Feeding软件主要由以下3部分构成。

- Part Feeding窗口
- Part Feeding SPEL+命令
- 回调函数

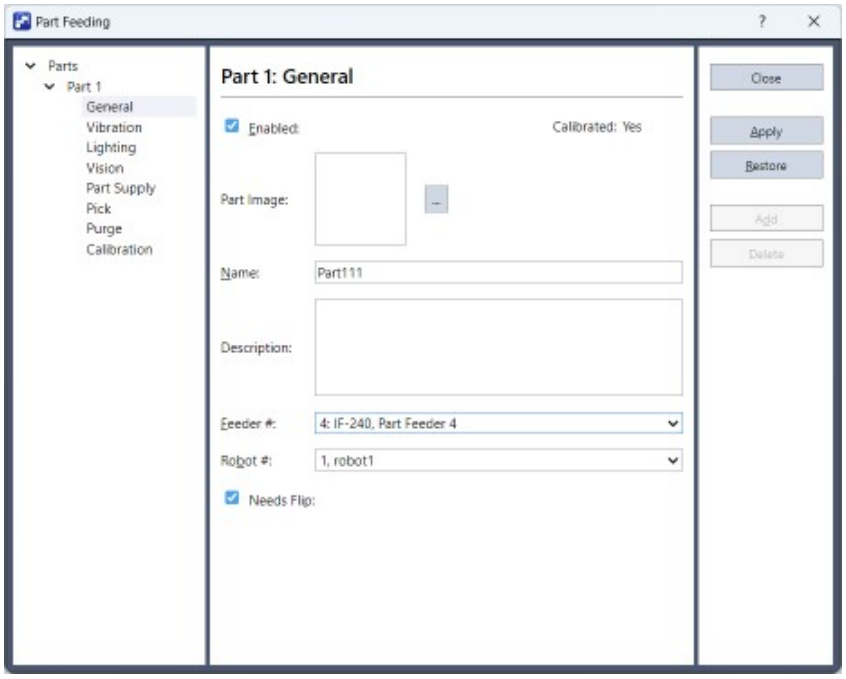
3.1.1.1 [上料]窗口

如果单击Epson RC+ 8.0菜单 -[工具] -[上料]，则会显示[上料]窗口。

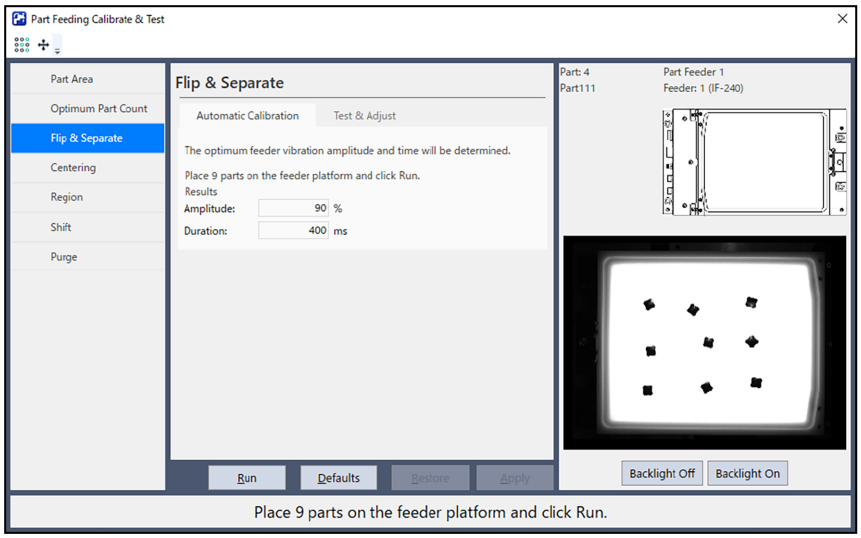


可进行以下操作。

1. 设定部件的各种参数。



2. 进行送料器的校准、手动调整/测试。只需根据画面中的指示，任何人都可以简单地进行校准作业。可细致地调整送料器的参数，只需操作一个按钮，即可进行手动调整/测试。



要点

如果未处于将RC+连接至控制器的状态，则不会显示上料窗口。通过虚拟控制器或以离线方式打开窗口的话，会发生错误。

3.1.1.2 Part Feeding SPEL+命令

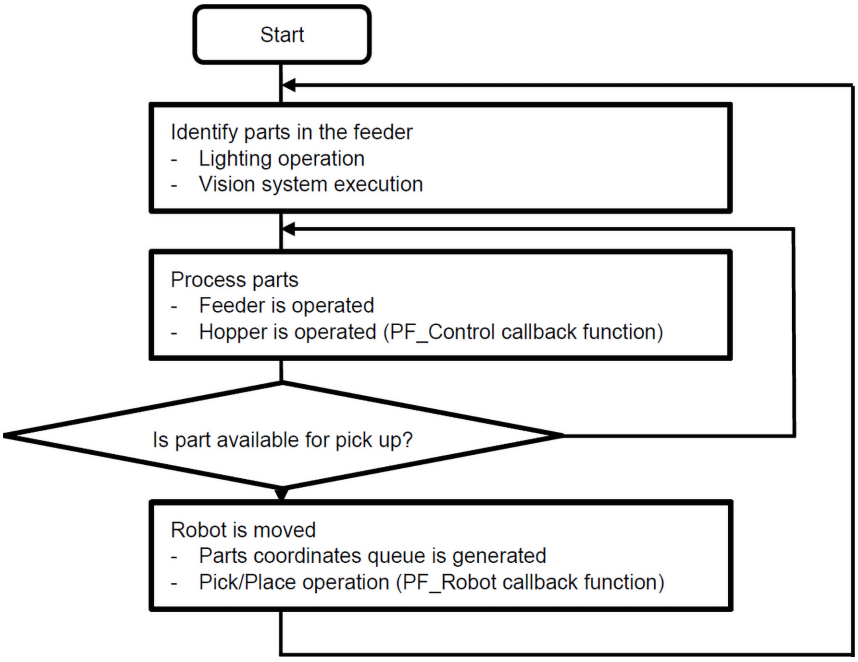
Part Feeding SPEL+命令是为了通过客户程序执行并控制Part Feeding选件而准备的SPEL+命令。如下所述为典型的命令。

命令	描述与用途
PF_Start	开始Part Feeding进程

命令	描述与用途
PF_Stop	发行Part Feeding进程停止请求
PF_Abort	强制结束Part Feeding进程
PF_QueGet函数	返回已注册至部件坐标队列（送料器上的部件坐标队列）的坐标数据
PF_InitLog	指定日志文件的输出路径
PF_Center	执行送料器的定芯运作
PF_Flip	执行送料器的翻转运作
PF_Shift	执行送料器的移动动作

3.1.1.3 Part Feeding进程

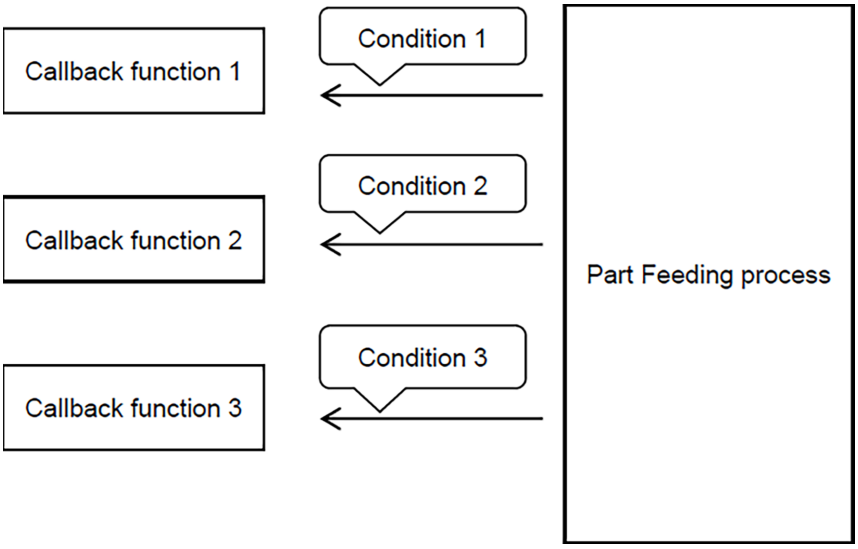
Part Feeding进程是指，自动进行视觉系统与送料器的控制并内置于Part Feeding系统中的进程运作。
利用PF_Start命令开始Part Feeding进程。利用PF_Stop命令等停止进程。
如下所述为Part Feeding进程的内容。



1. 识别送料器上的部件
   通过视觉系统识别平台上部件的数量或分布。
2. 处理部件
   为便于机器人抓取部件，控制送料器以移动部件。部件数量较少或较多时，调用PF_Control回调函数，通过料斗供给部件。
3. 移动机器人
   生成部件坐标队列（送料器上的部件坐标列表）。调用PF_Robot回调函数（请参阅下节），进行部件的取放动作。

3.1.1.4 回调函数

回调函数是用于在指定条件下由Part Feeding进程调用的SPEL+函数。通过添加部件，会在项目内自动生成回调函数。客户根据自己的装置，记述回调函数的内容。



比如，在PF_Robot回调函数中记述机器人拾取与放置部件的运作。如果适当地分散送料器上的部件，并且机器人进入可拾取部件的状态，Part Feeding进程则会调用PF_Robot回调函数。这样的话，回调函数就变为Part Feeding进程自动调用的函数，而非由客户创建函数调用的函数，敬请注意。

如下所述为回调函数一览与记述内容。

函数名称	记述内容
PF_Robot	部件取放动作
PF_Control	客户装置的设备（料斗、自定义照明）操作
PF_Status	错误处理
PF_MobileCam	使用移动相机时的机器人移动
PF_Vision	客户独自的视觉处理
PF_Feeder	客户独自的送料器运作
PF_CycleStop	发生停止指令时的处理

如下所述为回调函数被调用的条件。

条件	函数名称
可拾取部件	PF_Robot
部件为零	PF_Status
点亮自定义照明	PF_Control
将移动相机移动至拍摄位置	PF_MobileCam
发生错误	PF_Status

有关回调函数的详细信息，请参阅以下内容。

Part Feeding回调函数

3.1.2 Part Feeding用项目

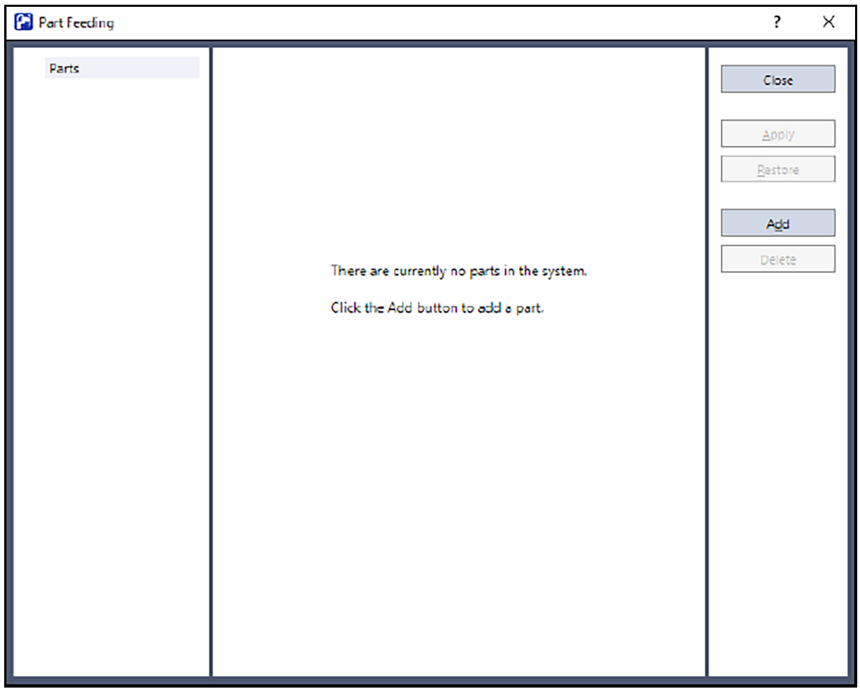
下面说明Part Feeding用SPEL项目。

3.1.2.1 将Part Feeding选件适用于项目

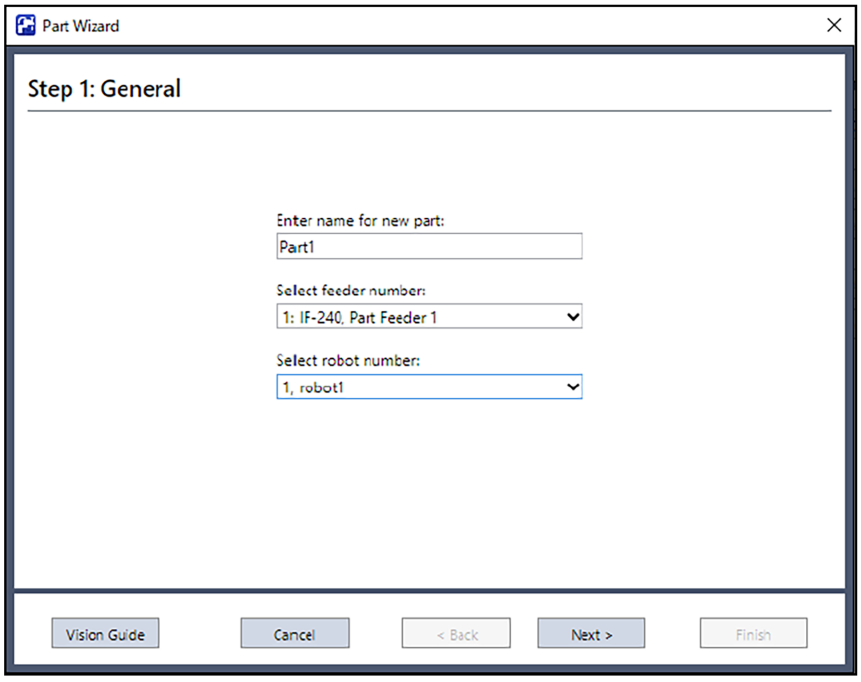
要将Part Feeding选件适用于项目时，按如下所述进行操作。

1. 打开要适用选件的现有项目。或新建项目。

2. 单击Epson RC+ 8.0菜单 - [工具] - [上料]。
   显示以下画面。



3. 单击[添加]按钮。
   启动部件向导。

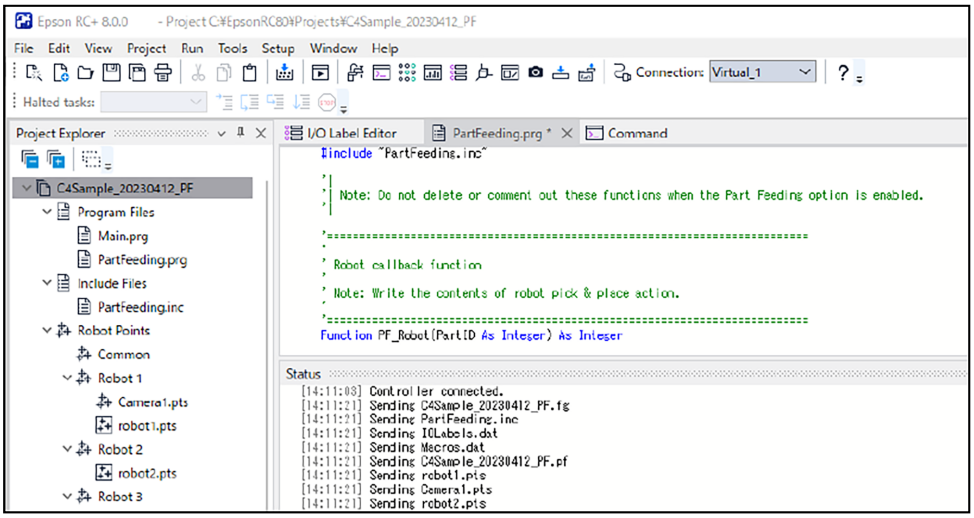


4. 根据部件向导设定部件。
   有关详细的设定内容，请参阅以下内容。
   部件向导
   创建1个部件。

   此时，Part Feeding选件专用的程序文件会被添加至项目中。请参阅以下内容。
   项目构成

3.1.2.2 项目构成

下面说明启用Part Feeding选件时被添加的部分。
如果在Part Feeding窗口中新设部件，则会添加以下程序文件。



- PartFeeding.prg
  记述有回调函数的模板代码。
- PartFeeding.inc
  记述有进行使用Part Feeding选件的编程时使用的常数。

要点

请勿从项目中排除或删除这些文件。否则会发生创建错误。

要点

进行下述语言设定时，程序中的注释会变为与语言设定相同的语言。

- 日文
- 德文
- 西班牙文

为除此以外的语言设定时，注释会变为英文。

3.1.2.3 构成文件

添加以下项目构成文件。
PF文件
是记述部件设定的文件。文件名为[项目名称].pf。

要点

请勿利用编辑器直接编辑文件。否则会导致无法读取文件并发生错误。

3.1.2.4 导入文件

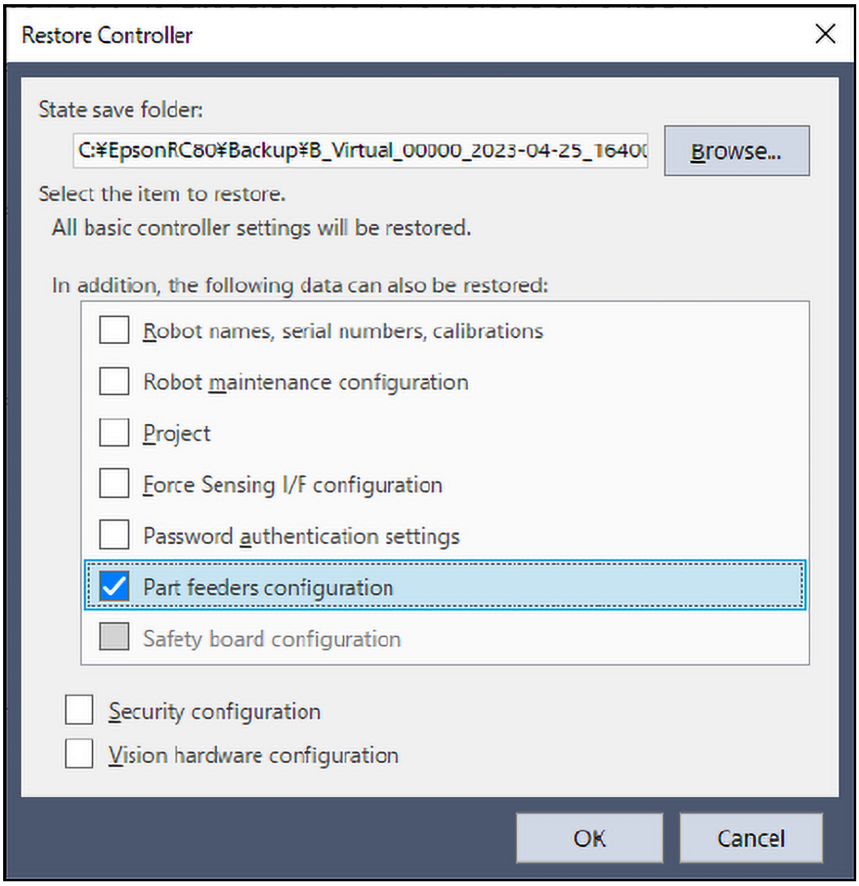
可导入PF文件。这样的话即可将部件信息复制到其他项目中。
有关详细信息，请参阅以下内容。
[导入](文件菜单)

3.1.2.5 控制器的备份和恢复

要备份控制器设定时，选择Epson RC+ 8.0菜单 - [工具] - [维护]中的[备份控制器]。
此时，Part Feeding选件的数据也会被同时备份。
要恢复已备份的控制器设定时，选择Epson RC+ 8.0菜单 - [工具] - [维护]中的[恢复控制器]。
此时，如果在以下画面中勾选[零件送料器配置]，Part Feeding选件的数据也会被同时恢复。

要点

因进行故障排除而将备份发送到本公司时，请通过Epson RC+8.0获取备份。使用控制器主体的存储器端口进行备份时，并不会对送料器使用状况相关数据进行备份。



3.1.3 SPEL编程

3.1.3.1 编程概述

下面说明Part Feeding选件的概述。在本章节以后将说明详细内容。
在几乎所有的应用中，都由系统自动进行送料器运作。此外，也会自动生成基本的SPEL程序。由已生成的名为回调函数的SPEL函数构成。
执行时，系统会调用回调函数。符合下述条件时会进行调用。

PF_Robot函数	送料器上有可拾取的部件时
PF_Status函数	状态发生变化时
PF_Vision函数	通过PF_Vision callback进行视觉处理时
PF_Feeder函数	进行独自的送料器控制时
PF_Control函数	控制料斗或前灯等时
PF_MobileCam函数	因使用移动相机检测部件而将相机移动至送料器上方时
PF_CycleStop函数	停止Part Feeding的运作时

为了处理特定要件，需要更改回调函数。
几乎所有的应用都需要使用PF_Robot与PF_Status回调函数。
如下所述为各回调函数的概述。

PF_Robot回调函数
如果完成送料器运作与视觉处理并判断可拾取部件，则会自动执行PF_Robot回调函数。记述实现部件取放动作的代码。在单纯的应用中，记述用于实现机器人取放动作与夹爪（部件抓取机构）动作的数行代码。如果PF_Robot回调函数结束，则会执行PF_Status回调函数。

PF_Status回调函数

如果各回调函数完成，则会自动执行PF_Status回调函数。PF_Status时，根据各回调函数的返回值向系统发出后续方法指示。另外，PF_Status时也向操作员发出发生错误状态的通知。错误包括机器人过载转矩错误等系统级错误、料斗进给过少或过多这样的Part Feeding错误等。为PF_Robot回调函数时，记述应对发生的错误的方法。在单纯的应用中，无需更改PF_Status回调函数的代码。

在更复杂的应用中，也许需要客户记述视觉处理。在这种情况下，需要使用PV_Vision回调函数。如下所述为PF_Vision回调函数的概述。

PF_Vision回调函数

客户进行零件送料器的视觉处理（而非系统准备的视觉处理）时，记述PF_Vision回调函数所需的处理。需要利用相机拍摄图像，启动部件检测序列，获取结果或将结果添加至部件坐标队列。在单纯的应用中，由系统进行视觉处理，因此无需记述PF_Vision回调函数。
有关详细信息，请参阅以下内容。
PF_Vision

偶尔需要使用客户准备的照明（而非内置的背光灯）。PF_Control回调函数用于控制客户准备的照明。也通过PF_Control回调函数控制客户准备的料斗。另外，也通过PF_Control回调函数控制本公司的料斗选件。如下所述为PF_Control回调函数的概述。

PF_Control回调函数

系统判断需要通过料斗进行进给时，或系统判断需要进行灯的ON/OFF运作时，会自动执行PF_Control回调函数。使用Epson的料斗选件时，需要将通过PF_Control回调函数注释的PF_OutputONOFF命令，进行非注释化处理。

PF_MobileCam回调函数

为设置在机器人上的移动相机（而非设置在送料器上方的向下相机）时，会自动执行PF_MobileCam回调函数。在PF_MobileCam回调函数中记述机器人将相机移动至送料器上方的代码。此外，检测到部件并将部件数据添加至Part Feeding队列后，也记述机器人从送料器上方的转移动作。使用向下固定相机时，无需记述PF_MobileCam回调函数。

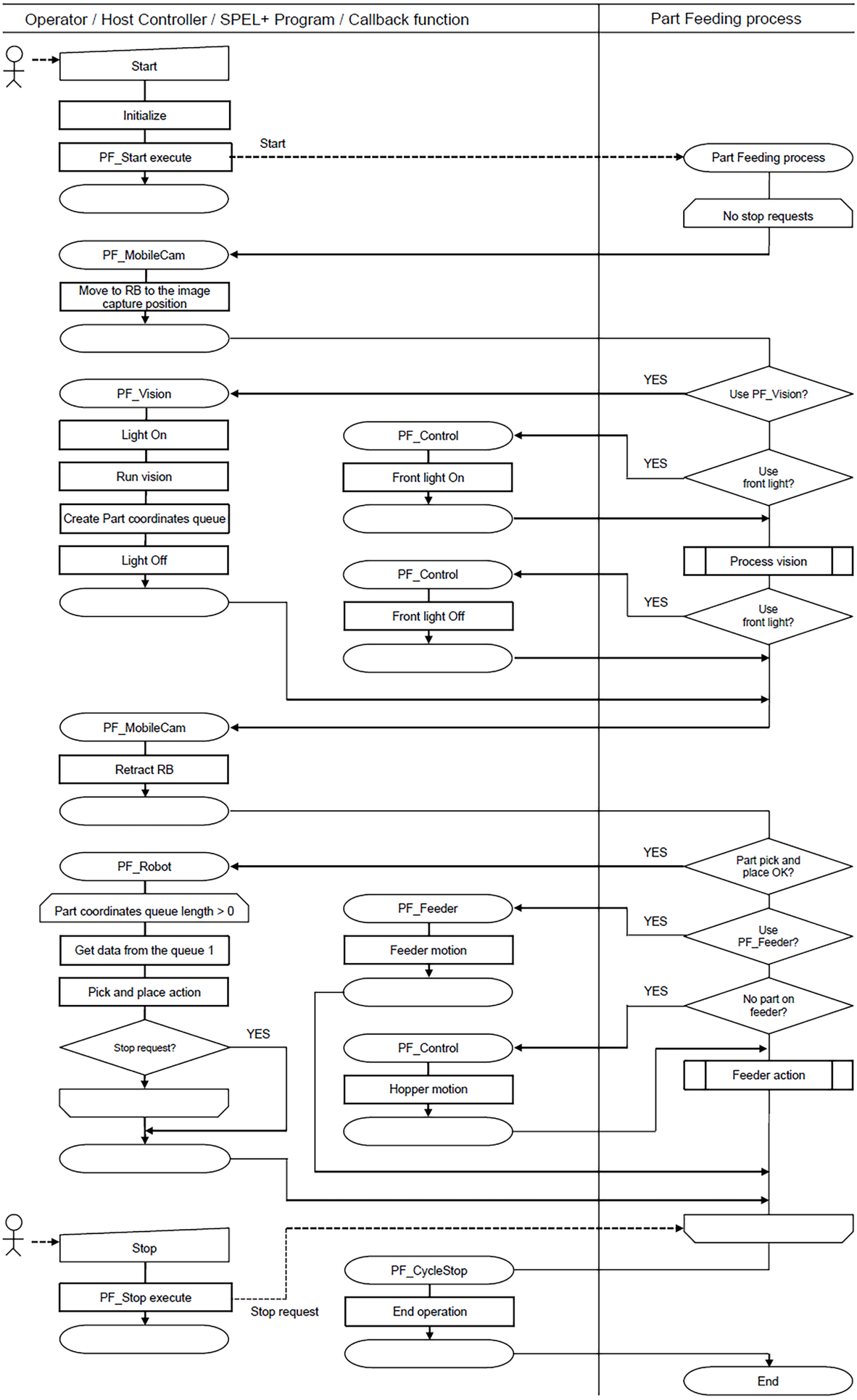
PF_CycleStop回调函数

如果执行PF_Stop，则自动执行PF_CycleStop回调函数。该PF_CycleStop回调函数用于将机器人移动到安全位置或将输出置为ON或OFF那样的结束处理。在单纯的应用中，无需记述PF_CycleStop回调函数。

PF_Feeder回调函数

客户的装置或部件无法顺利进行自动送料器控制处理时，可使用PF_Flip或PF_Shift命令等，在PF_Feeder回调函数内通过SPEL代码记述送料器运作。一般来说，无需记述PF_Feeder回调函数。

所有的回调函数都需要设定返回值。要通过函数返回值时，将值分配给函数名称（比如，PF_Robot = PF_CALLBACK_SUCCESS，其中，PF_CALLBACK_SUCCESS为定义值0的常数）。正常结束时，会将PF_CALLBACK SUCCESS作为返回值。PF_Status时，会在某个回调函数完成后或发生错误后，向系统发出后续方法的指示。系统需要识别要执行的运作（继续、结束、重新启动等）。比如，可将返回值设为常数PF_EXIT并结束。在简单的应用中，可直接使用自动生成的回调函数的返回值。
有关详细信息，请参阅以下内容。
Part Feeding回调函数
下页所述为进行拾取&放置的程序的全体图。


Operator / Host Controller / SPEL+ Program / Callback function
Part Feeding process
Start
Initialize
PF_Start execute
Start
Part Feeding process
No stop requests
PF_MobileCam
Move to RB to the image capture position
PF_Vision
YES
Use PF_Vision?
Light On
Run vision
Create Part coordinates queue
Light Off
PF_Control
Front light On
YES
Use front light?
Process vision
PF_Control
Front light Off
YES
Use front light?
PF_MobileCam
Retract RB
PF_Robot
YES
Part pick and place OK?
Part coordinates queue length > 0
Get data from the queue 1
Pick and place action
Stop request?
YES
PF_Feeder
Feeder motion
YES
Use PF_Feeder?
YES
No part on feeder?
PF_Control
Hopper motion
Feeder action
Stop
PF_Stop execute
Stop request
PF_CycleStop
End operation
End

3.1.3.2 开始Part Feeding进程

如下所述为开始Part Feeding进程所需的处理。

1. 将机器人移动至可进行视觉拍摄的位置。
   （为固定向下相机时，移动至不影响拍摄的位置。）
2. 执行PF_Start命令，开始Part Feeding进程。

如下所述为程序示例。

```
Function PickParts

   Home
   PF_InitLog 1, "C:\log.csv", True


Fend
```

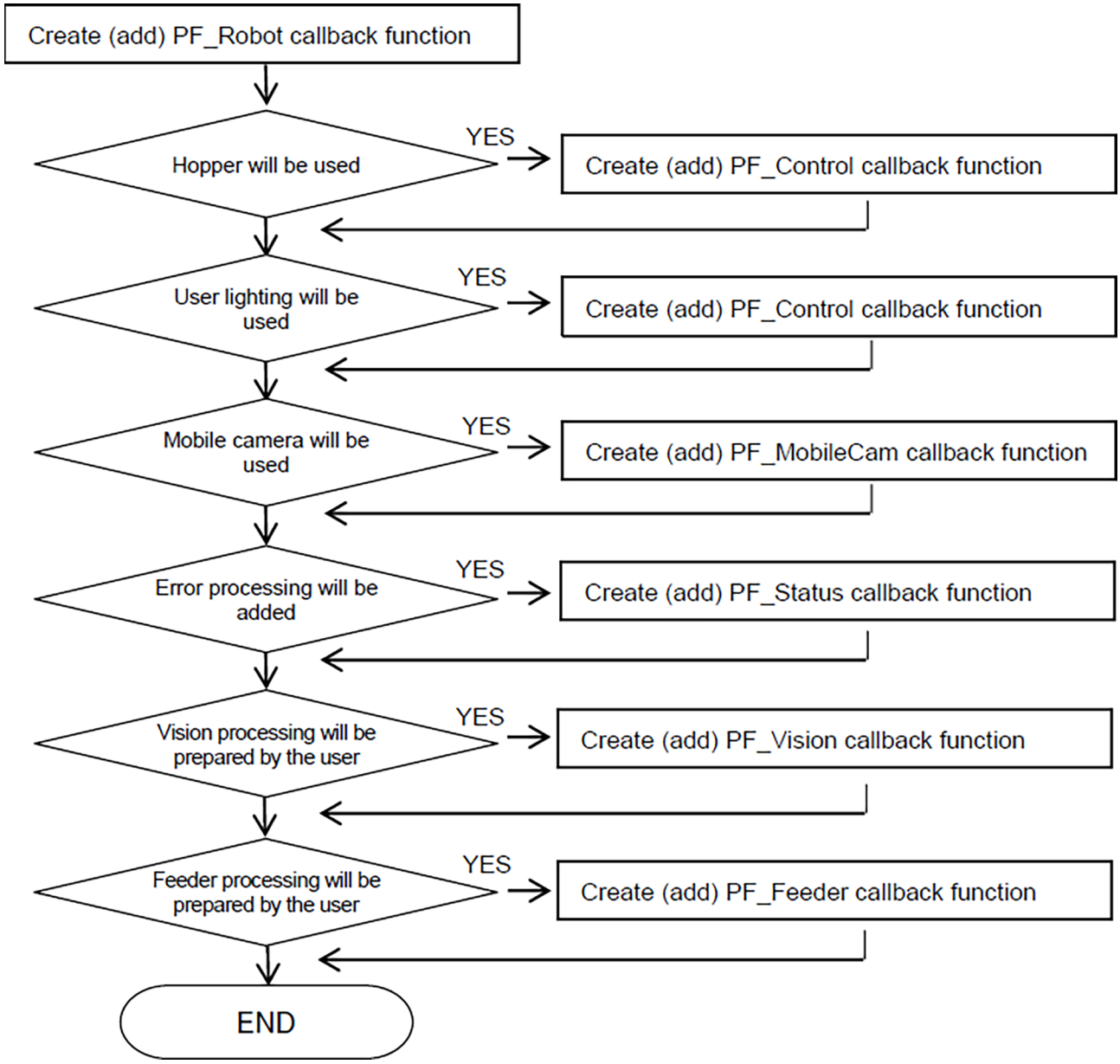
要点

记述以下处理，以便在装置启动时执行。

- 设定用于部件取放的机器人的速度、加速度与功率模式等
- 设定料斗，设定自定义照明，进行末端夹具姿势初始化等

3.1.3.3 拾取&放置处理

如下所述为拾取&放置处理的程序创建作业进程。



创建（添加）PF_Robot回调函数为必须作业。有关除此之外的项目，按客户装置规格在需要时进行作业。

PF_Robot回调函数

在PF_Robot回调函数中记述机器人拾取与放置部件的运作。该函数按如下所述进行运作。（重复1～7。）

1. 从部件坐标队列中获取平台上的部件坐标。
   使用PF_QueGet。
2. 将机器人移至部件位置。
   使用Jump等。（为SCARA型机器人时）
3. 将吸附置为On等，然后抓取部件。
4. 将机器人移至部件放置位置。
   使用Jump等。（为SCARA型机器人时）
5. 将吸附置为Off等，然后释放部件。
6. 删除1个部件坐标队列中的数据。
   使用PF_QueRemove。
7. 确认是否产生停止指令。如果产生，则退出循环。
   使用PF_IsStopRequested。
   有关编程示例，请参阅以下内容。

- 创建PF_Robot回调函数
- 应用程序示例

向送料器供给部件（PF_Control）选件

编写使用料斗向送料器供给部件时的运作程序。在PF_Control回调函数（PartFeeding.prg）中记述。
如下所述为程序内容。

1. 记述平台上没有任何部件时的料斗运作（部件进给）。
   此时，PF_Control回调函数的自变量Control为PF_CONTROL_SUPPLY_FIRST。
2. 记述向平台上添加部件时的料斗运作（部件进给）。
   此时，PF_Control回调函数的自变量Control为PF_CONTROL_SUPPLY。
   有关编程示例，请参阅以下内容。
   创建PF_Robot回调函数

自定义照明的操作（PF_Control）选件

编写使用自定义照明时的运作程序。
在PF_Control回调函数（PartFeeding.prg）中记述。
如下所述为程序的内容。

1. 记述点亮自定义照明的运作。
   此时，PF_Control回调函数的自变量Control为PF_CONTROL_LIGHT_ON。
2. 记述熄灭自定义照明的运作。
   此时，PF_Control回调函数的自变量Control为PF_CONTROL_LIGHT_OFF。
   有关编程示例，请参阅以下内容。
   创建PF_Robot回调函数

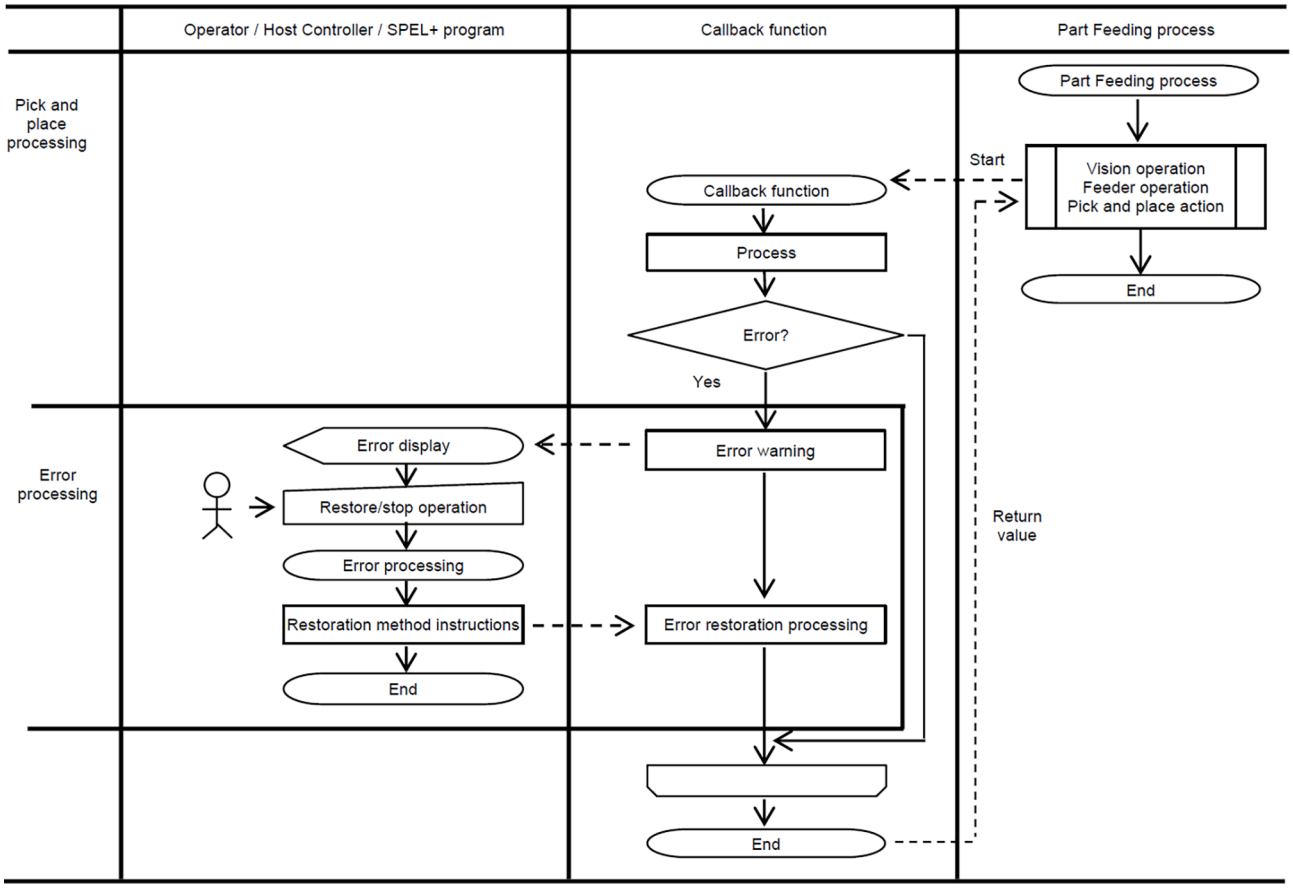
要点

请记述点亮与熄灭自定义照明的两种运作。仅记述点亮运作时，可能会导致无法顺利地通过视觉系统识别部件并发生错误。

3.1.3.4 错误处理

下面说明Part Feeding选件运作期间发生错误的处理方法。

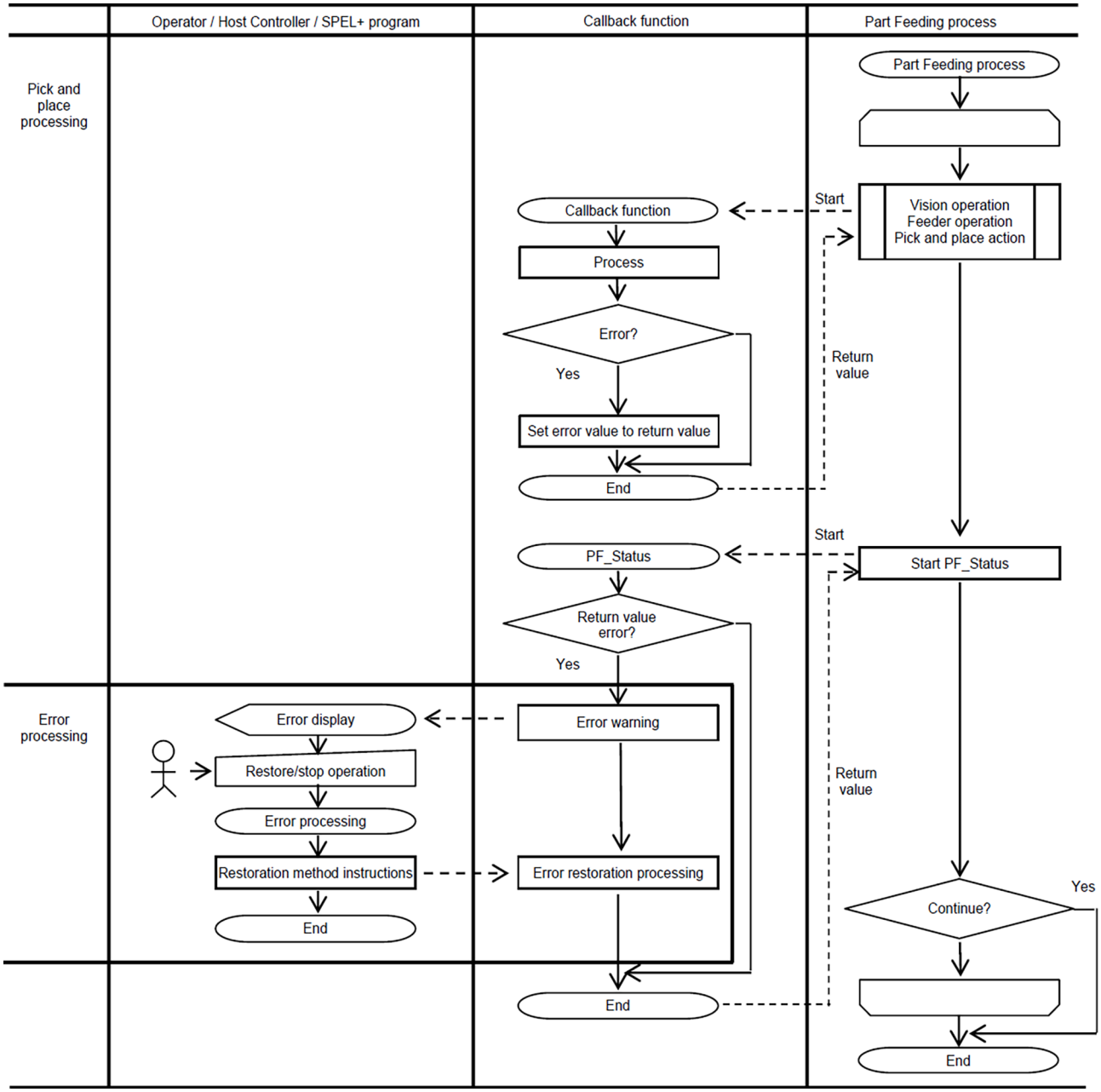
1. 在回调函数中处理错误



在回调函数中检测和处理在回调函数中发生的错误。不将控制返还给Part Feeding进程而要继续进行操作时，按该方法进行处理。

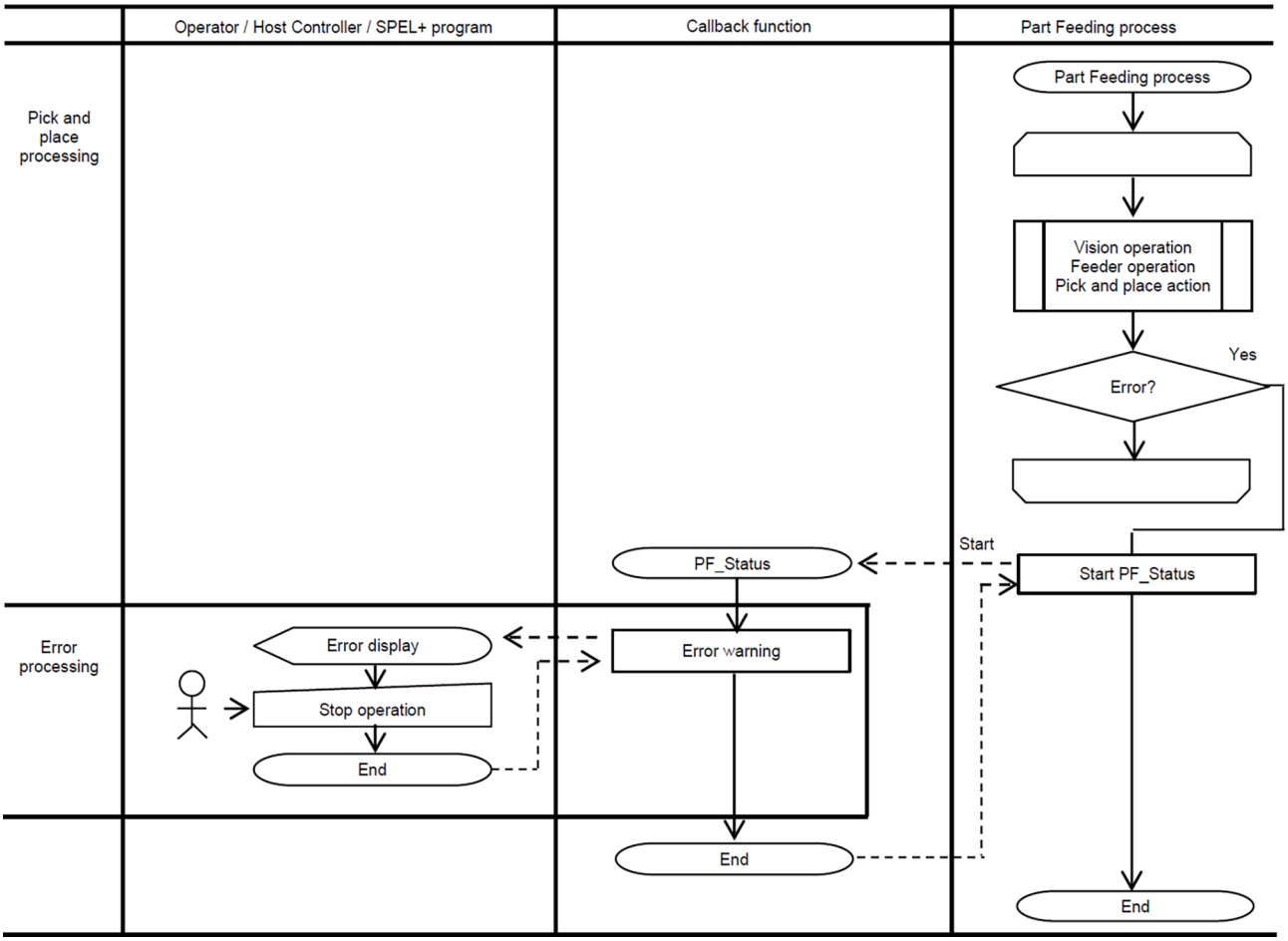
例：在PF_Robot回调函数中发生部件吸附错误，并进行重试处理

2. 在PF_Status回调函数中处理错误



在回调函数中发生错误时，将返回值设为PF_CALLBACK_SUCCESS以外的值（用户错误8000-8999）。
Part Feeding进程时，会将该值设为PF_Status回调函数的自变量Status并进行启动。
要将错误处理通用化时，按该方法进行处理。
有关详细信息，请参阅以下内容。
PF_Status

3. 处理Part Feeding进程内部发生的错误



可能会因Part Feeding参数设定不善（PF_Start自变量指定不善）或视觉设定不善等而导致在Part Feeding进程内部发生错误。
Part Feeding进程时，会将PF_STATUS_ERROR设为自变量Status并进行启动。
如果PF_Status函数结束，Part Feeding进程则会结束。
有关详细信息，请参阅以下内容。

- PF_Status
- 错误处理

3.1.3.5 结束处理

要通过客户程序结束Part Feeding进程时，可采用以下某种方法。

1. 执行PF_Stop命令
   通过客户程序，执行PF_Stop命令。
   Part Feeding进程时，会在当前处理的回调函数结束并且PF_CycleStop回调函数结束后结束运作。
   有关详细信息，请参阅以下手册。
   "Part Feeding SPEL+命令参考 - PF_Stop"
2. 执行PF_Abort命令
   通过客户程序，执行PF_Abort命令。
   Part Feeding进程会立即结束。
   此时，当前处理的回调函数也会立即结束。
   有关详细信息，请参阅以下手册。
   "Part Feeding SPEL+命令参考 - PF_Abort"

要点

如果打开安全门或执行Pause，送料器与料斗振动则会停止。即使关闭安全门或执行Continue继续操作，也不重新开始振动。

3.1.3.6 Part Feeding进程使用的功能

Part Feeding进程运作期间，会占用以下功能。请进行适当的安装，确保Part Feeding进程正在运作时，客户程序不会使用这些功能。

送料器编号	任务	计时器	SyncLock
1	32	63	63
2	31	62	62
3	30	61	61
4	29	60	60
系统（已预约）	28	-	59

为下述条件时，可使用这些功能。

- Part Feeding进程处于停止状态时
- 未使用多台送料器时（比如，仅连接1台送料器时，可使用送料器编号2以后占用的功能。）

要点

请勿使用已预约的系统任务与SyncLock命令。

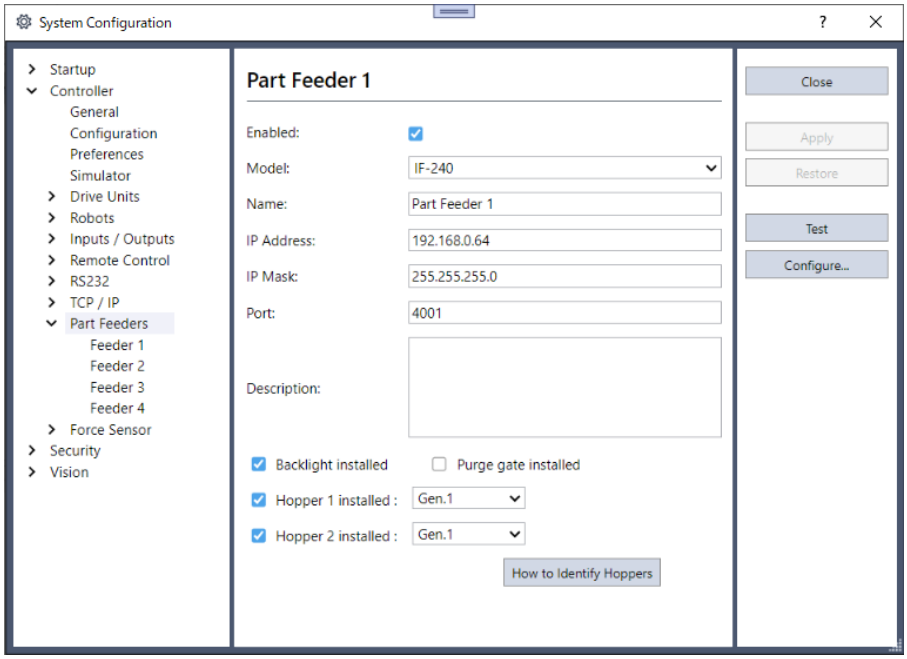
3.2 Part Feeding GUI

下面说明Epson RC+ 8.0 Part Feeding 8.0的GUI。

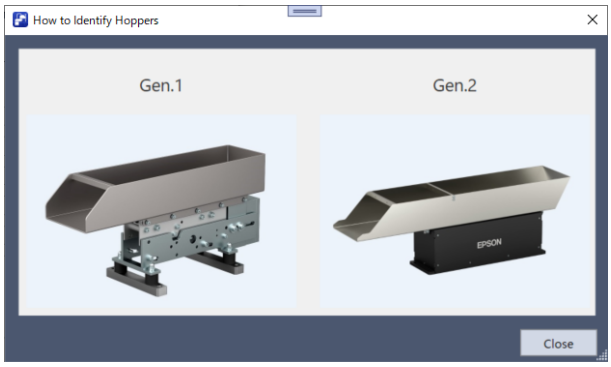
3.2.1 操作入门

在Epson RC+ 8.0菜单 -［设置］-［系统配置］中进行将送料器连接至控制器的设定。

3.2.1.1 零件送料器画面



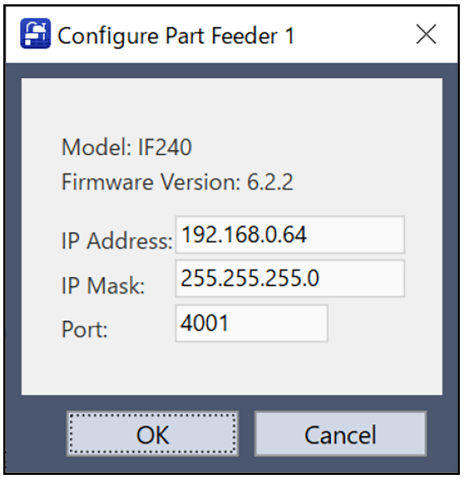
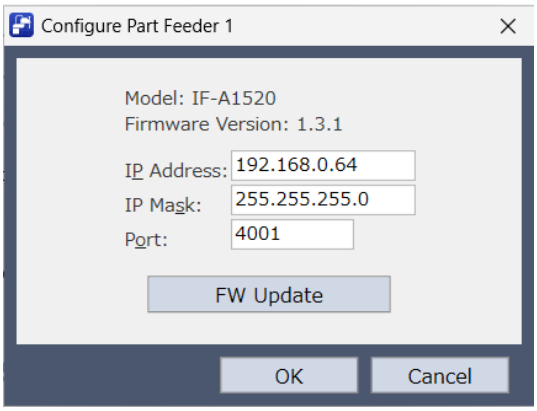
项目	描述
启用	要启用送料器时，勾选复选框。
型号	选择送料器型号。
送料器名称	设定任意名称。（半角字母数字与下划线。最多32个字符）
IP地址	输入当前送料器中设定的IP地址。 默认IP地址为192.168.0.64。
子网掩码	输入当前送料器中设定的子网掩码。 默认子网掩码为255.255.255.0。
端口	输入当前送料器中设定的端口编号。 默认端口编号为4001。
注释	填写送料器的说明（注释）。属于选件。（半角字母数字与下划线。最多32个字符）
安装背光灯	已安装送料器内置背光灯时，勾选复选框。
安装清除门	使用送料器选件的清除门时，勾选复选框。（仅限于IF-240，IF-380，IF-530、IF-A1520、IF-A2330）
安装料斗1 安装料斗2	连接料斗时，勾选复选框。从Gen.1/Gen.2中选择料斗类型。有关料斗的类型，请参阅以下内容。 料斗

项目	描述
料斗的区分方法	显示料斗的类型。 

要点

- 要更改送料器的网络设定时，单击下面说明的[设置]按钮。
- 如果输入了当前送料器中未设定的IP地址、子网掩码与端口并与送料器进行通信，则会发生错误。已单击[应用]按钮时，不发生错误。
- “安装清除门”设定会对振动参数产生影响。使用清除门时，在上料对话框中添加新部件之前，将“安装清除门”设定设为ON。如果在添加新部件后勾选复选框，送料器则不能正确进行运作。
- 使用IF-80时，安装料斗与料斗类型会被自动设为以下内容。不能更改设定。
  - 安装料斗1：选择；安装料斗2：解除选择
  - 料斗1类型：Gen.2；料斗2类型：空栏
- 使用IF-240并选择安装料斗1与安装料斗2其中之一时，如果选择安装清除门，则会显示只能使用1台料斗的通知信息。如果选择[否]，则恢复设定；如果选择[是]，则解除安装料斗2选项。
- 使用IF-240并选择安装清除门与安装料斗1时，如果选择安装料斗2，则会在安装清除门无效时显示可使用2台料斗的通知信息。如果选择[否]，则恢复设定；如果选择[是]，则解除安装清除门选项。
- 打开利用EPSON RC+ 7.0创建的Part Feeding项目时，如果连接料斗，则会被识别为料斗1，并且按如下所述设定料斗1类型。
  - IF-80：Gen.1
  - IF-80以外：Gen.2

按钮	描述
关闭	关闭对话框。
应用	应用编辑内容。
恢复	恢复编辑内容。
测试	进行送料器的通信确认。

按钮	描述
设置	更改送料器的网络设定。  - IP地址 设置送料器的新的IP地址。 - 子网掩码 设置送料器的新的子网掩码。 - 端口 设置送料器的新的端口编号。 设置后，单击[OK]按钮。 选择IF-A1520/2330时，如果PC和IF-A1520/2330直接用以太网电缆连接，会显示固件更新[FW Update]按钮。 

⚠ 注意

- 使用时，请在送料器中设置以下专用IP地址。

  Class A：10.0.0.0 - 10.255.255.255

  Class B：172.16.0.0 - 172.31.255.255

  Class C：192.168.0.0 - 192.168.255.255

- 当需设置控制器全局IP地址时，请务必在使用前充分了解有关未经授权访问等风险。

要点

如果未将控制器与送料器设为同一网段，则不能进行通信。

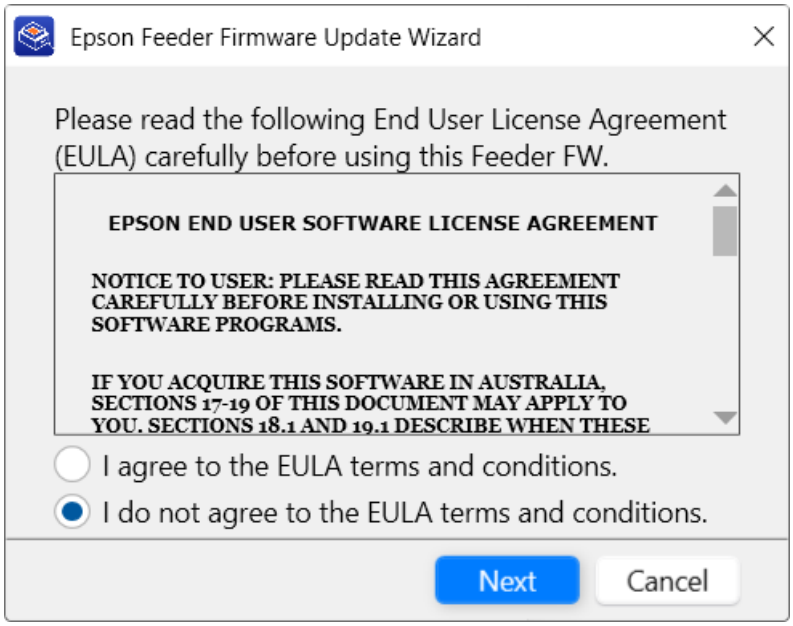
设定了与控制器不同的网络时，无法与送料器进行通信。在这种情况下，需要更改控制器的IP地址或子网掩码。

该设定位于Epson RC+ 8.0菜单 - [设置] - [系统配置] - [控制器] - [配置]中。
有关详细信息，请参阅以下手册。
“Epson RC+ 8.0 用户指南 - Ethernet通信”

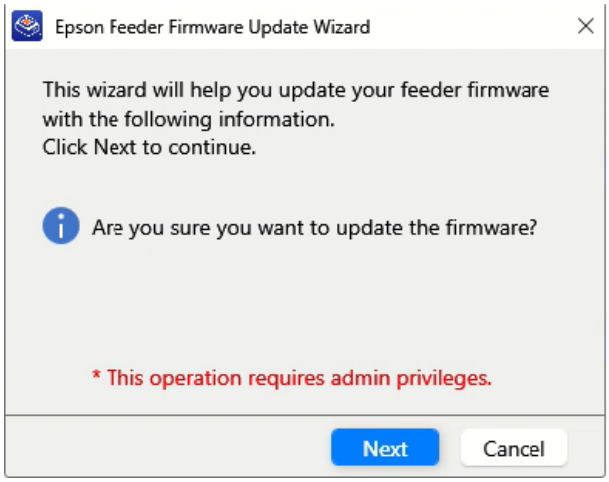
3.2.1.2 固件更新

在IF-A1520/2330中，PC和送料器通过以太网连接至同一网络时，[送料器的网络设定]画面中会显示[FW Update]按钮。

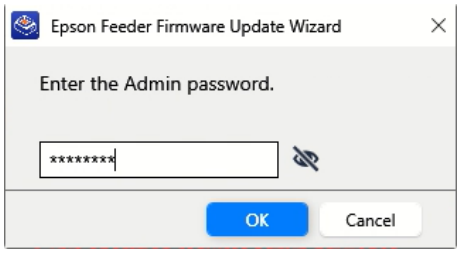
1. 显示固件的使用许可协议。
   请仔细阅读内容，只有在同意时选择[I agree to the EULA terms and conditions]，然后单击[Next]。



2. 显示更新向导。要开始固件更新时，单击[Next]。



3. 在密码输入画面中输入密码，然后单击[OK]。



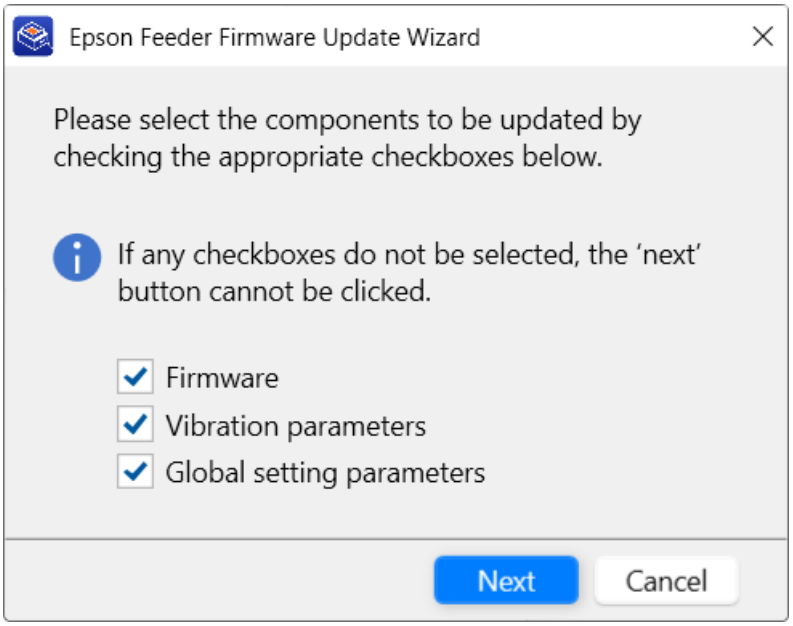
要点

- 初始密码为“12345678”。
- 输入了初始密码时，会显示更改密码对话框。请输入2次新密码。

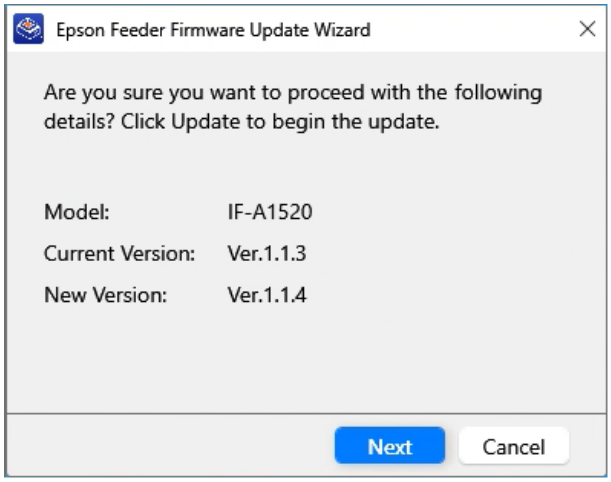
- 密码用1至16个半角字母数字字符指定。无法设置与初始密码相同的密码。
- 请务必记下更改后的密码，以免遗忘。

3-2 选择要更新哪个项目，然后单击[Next]。默认设置为勾选所有复选框。

- Firmware： 更新送料器的固件（执行文件）时选择
- Vibration parameters： 要将供料器中存储的振动/序列参数重置为初始值时选择。
- Global setting parameters： 要将各种内部参数重置为初始值时选择。
  初始值可能会随固件更新而改变。

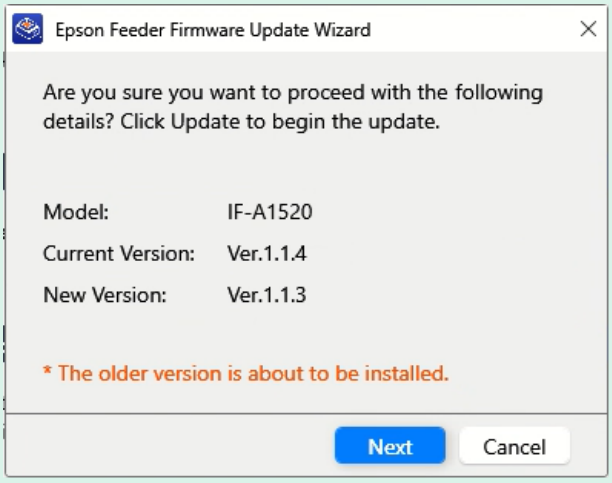


4. 显示更新内容。如无问题，单击[Next]。



要点

如果要安装比已安装固件更早的版本时，会出现警告消息“* The older version is about to be installed.”。

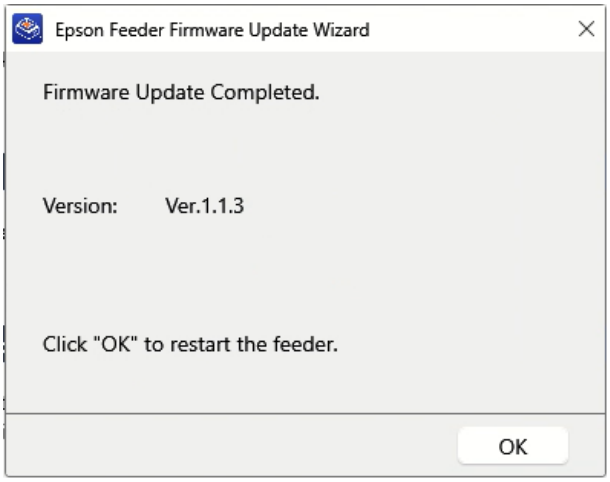


5. 进行固件写入。
   固件写入工作大约需要10秒钟。

⚠ 注意

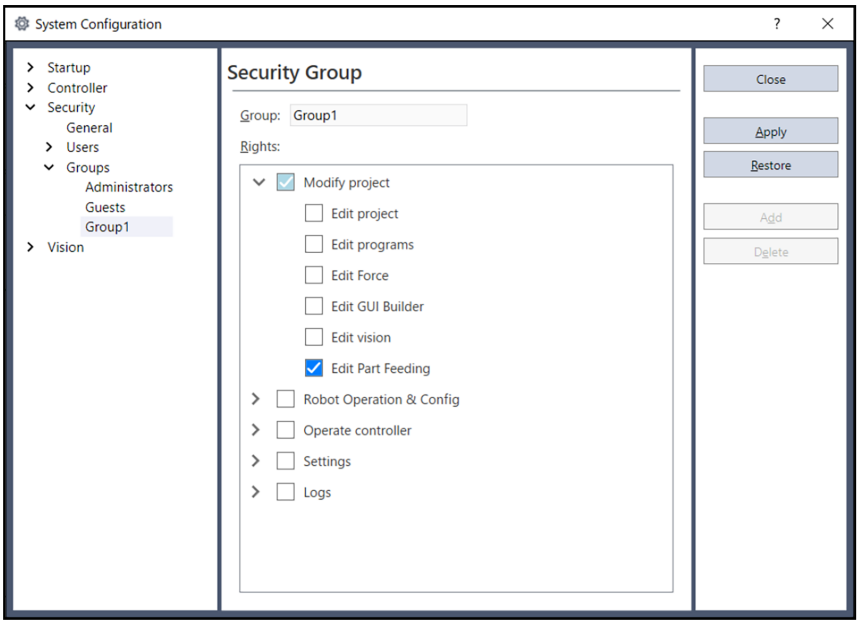
显示以下画面期间，千万不要切断送料器主体的电源。此外，如果断开与PC的连接，固件写入可能会失败，并导致送料器无法启动。

6. 显示更新向导的完成画面。如果单击[OK]，送料器主体会自动重新启动。



3.2.1.3 安全画面

通过进行安全设定，可限制浏览/编辑Part Feeding选件的设定的用户。

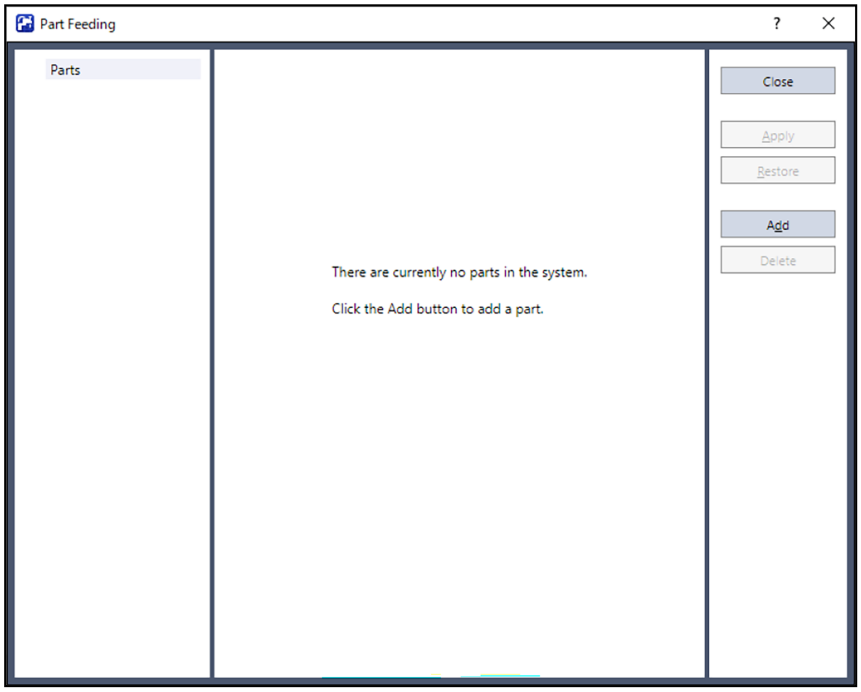


勾选[Part Feeding编辑]复选框时，属于该组的用户可显示[上料]窗口。
未勾选[Part Feeding编辑]复选框时，如果属于该组的用户要显示[上料]窗口，则会发生错误。
有关详细信息，请参阅以下手册。
“Epson RC+ 用户指南 - 安全”

3.2.2 部件向导

3.2.2.1 新建部件

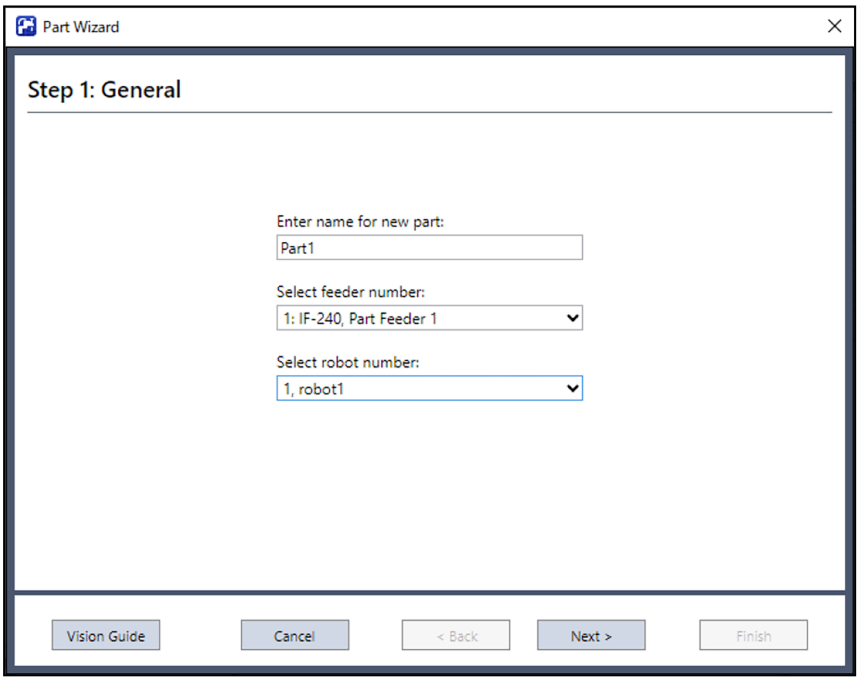
1. 单击Epson RC+ 8.0菜单 - [工具] - [上料]。



2. 单击[添加]按钮。启动部件向导。

3.2.2.2 常规

进行基本设定。



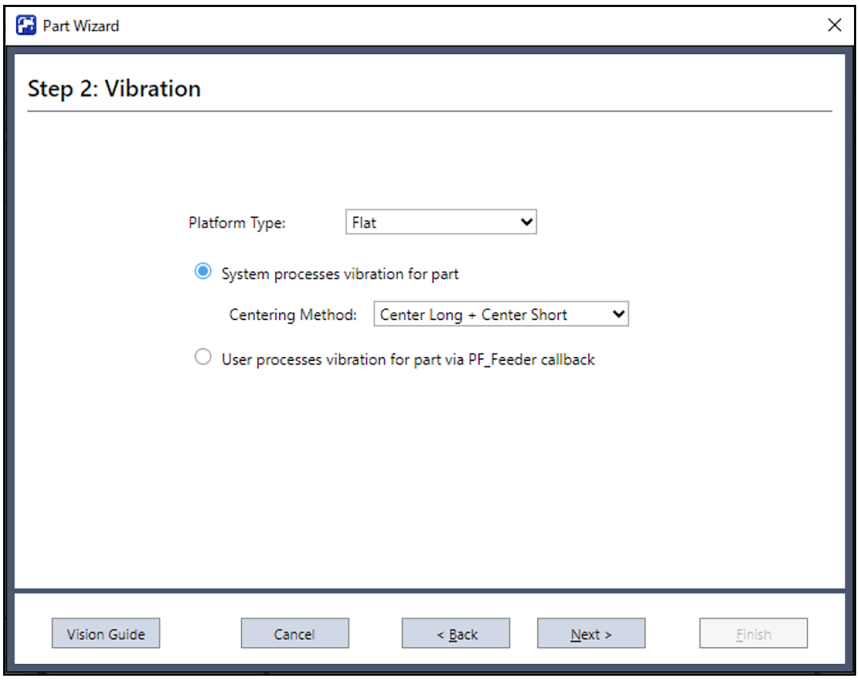
项目	描述
新部件名称	记述部件名称。（半角字母数字与下划线。最多32个字符）
选择送料器编号	选择该部件使用的送料器编号。在系统配置画面中确认送料器编号。
选择机器人编号	选择机器人编号。

下面说明部件向导的基本操作。通过窗口下部的5个按钮操作部件向导。

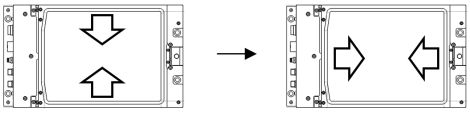
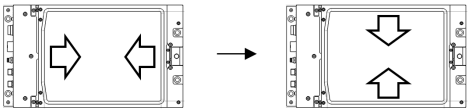
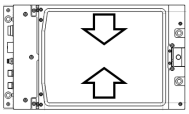
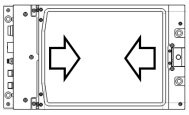
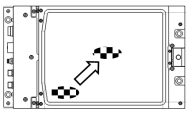
按钮	说明
视觉指南	显示Vision Guide画面。可创建视觉序列。
取消	中止部件向导。
返回	返回上一个部件向导。
下一步	进入下一个部件向导。
完成	退出部件向导。

3.2.2.3 振动

进行送料器的振动设定。



项目	描述
平台类型	指定平台的类型。 a) 是可从本公司采购的标准平台。 平面：平坦的平台 防滚动：已进行防滚动加工的平台 防粘贴：已进行防粘贴加工的平台（仅限于IF-80/240） b) 是客户独自制作的自定义平台。 长槽：已进行部件竖立用长槽加工的平台 孔：已进行部件竖立用孔加工的平台 定位槽：已进行部件方向对齐用孔加工的平台
由系统进行振动处理	由系统进行送料器控制。 已选择上述b)时，不能使用该选项。
进行PF_Feeder callback振动处理	使用PF_Feeder回调函数。

项目	描述
定芯方法	已选择[由系统进行振动处理]时，选择部件的定芯运作（投放部件时等部件分布偏差较大时，使部件靠向中央以进行均等分散的运作）类型。 无： 不进行定芯。 长轴定芯 + 短轴定芯： 向长轴方向进行定芯，然后向短轴方向进行定芯。  短轴定芯 + 长轴定芯： 向短轴方向进行定芯，然后向长轴方向进行定芯。  长轴定芯： 仅向长轴方向进行定芯。  短轴定芯： 仅向短轴方向进行定芯。  移动定芯： 通过移动动作进行定芯。 

要点

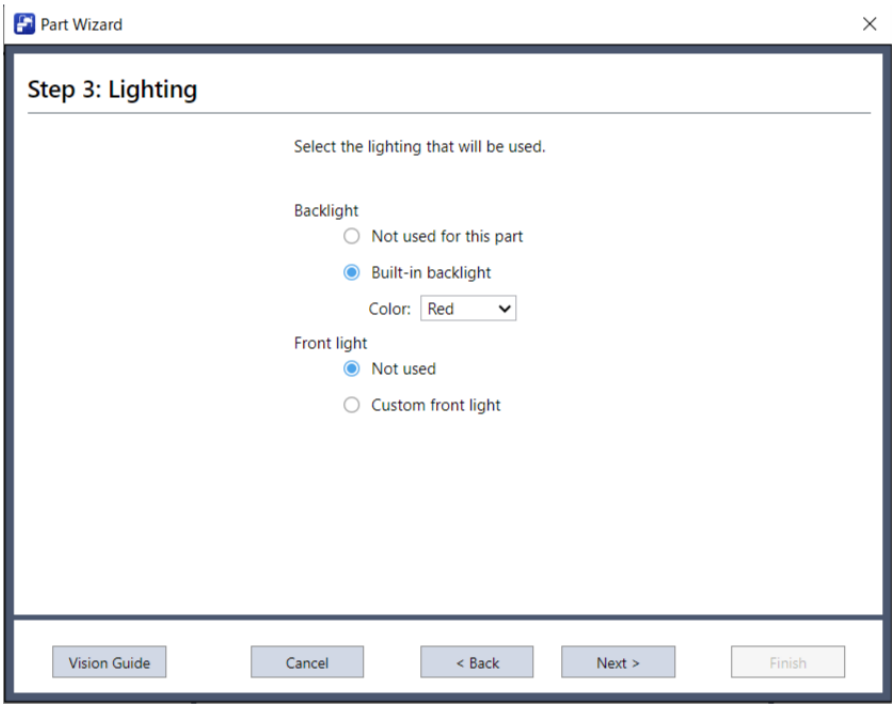
定芯方法的最佳选项因部件类型或料斗位置而异。下述某种方法效果最佳。

- 长轴定芯 + 短轴定芯
- 短轴定芯 + 长轴定芯

但与其他定芯方法（或“无”时）相比，送料器运作时间会延长。如果选择适当分散部件并且定芯所需时间最短的选项，则会有一定的效果。

3.2.2.4 照明

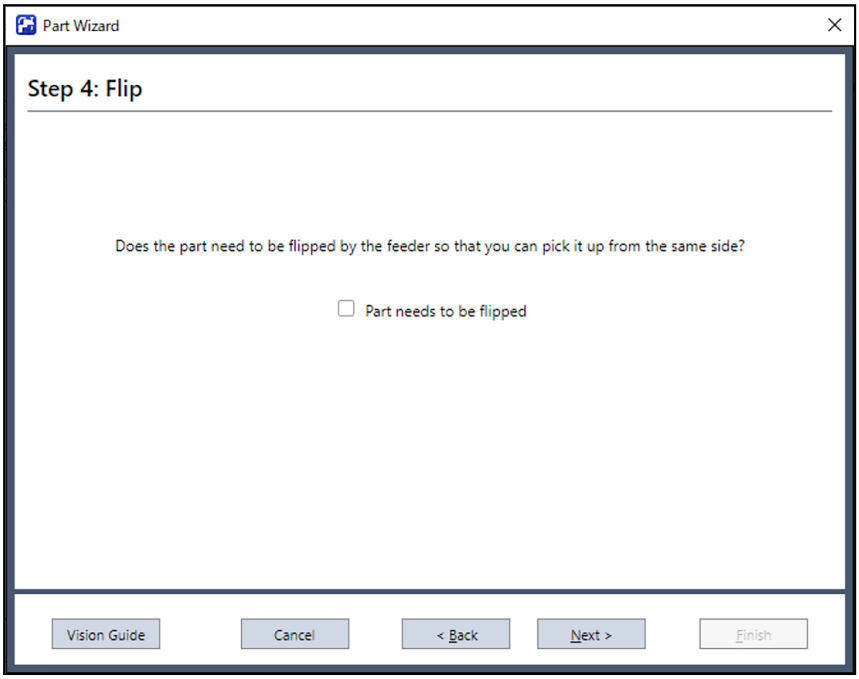
进行照明设定。



项目		描述
背光灯		指定背光灯的控制方法。
	未使用	视觉系统拍摄时，在不使用送料器背光灯的情况下选择。属于选件。
	内置背光灯	视觉系统拍摄时，在使用送料器背光灯的情况下选择。没有背光灯时不能选择。
	颜色	从“白色”、“红色”中选择背光灯的颜色。 (仅限于IF-A1520/23300)
前灯		指定前灯（选件）的控制方法。
	不使用	不使用前灯。
	自定义前灯	调用PF_Control回调函数，控制客户准备的自定义照明。属于选件。 有关PF_Control回调函数，请参阅以下内容。 PF_Control

3.2.2.5 翻转

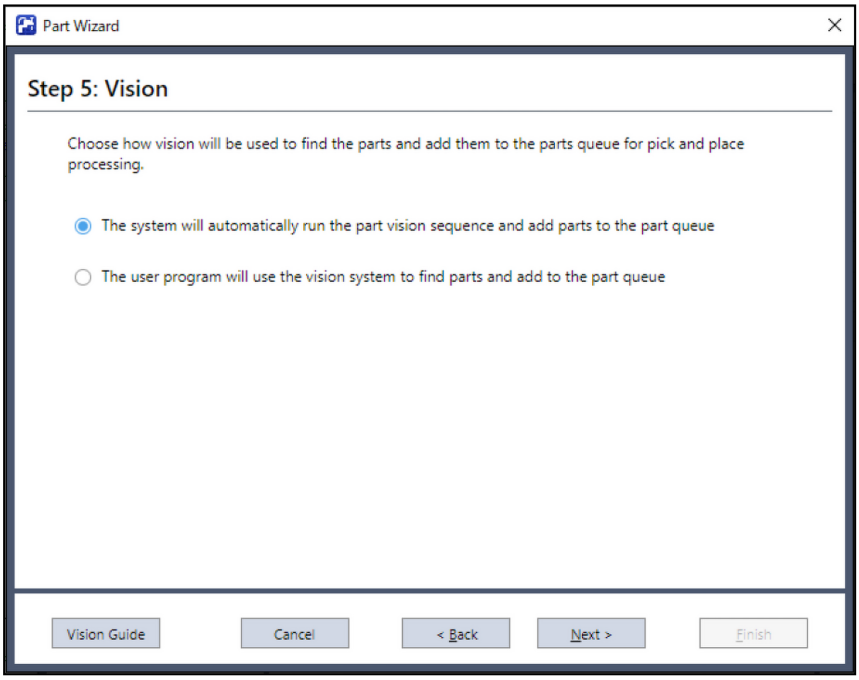
部件需要翻转时，进行翻转设定。需要拾取朝向特定面的部件时，勾选“需要翻转”。



项目	描述
需要翻转	部件具有表里等姿势时选择。启用更改拾取姿势的翻转动作。这样的话，1次送料器运作可获取的部件数可能会增加，并提高循环时间。

3.2.2.6 视觉

进行视觉设定。

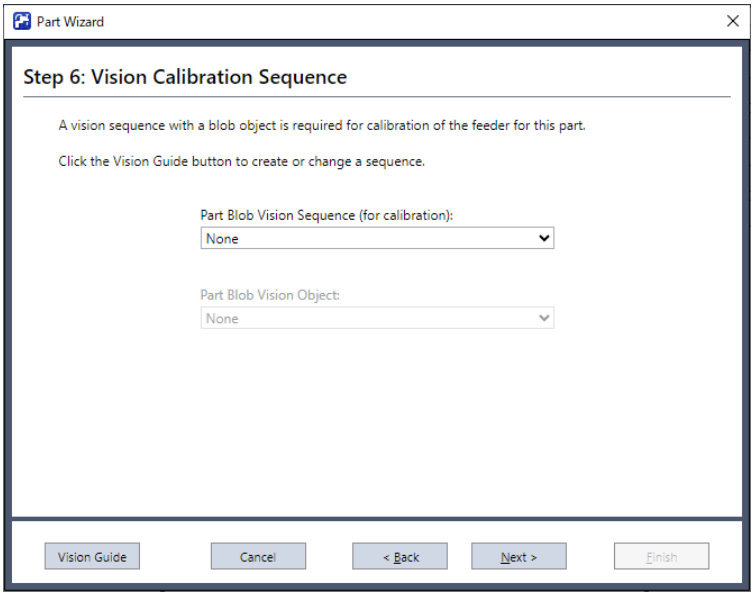


项目	描述
系统自动执行部件视觉序列，并将部件添加至部件队列。	系统自动进行视觉处理。通常选择该项目。

项目	描述
记述用户执行PF_Vision Callback所需的处理（执行照明的运作和视觉处理，检测部件并添加至部件队列）。系统自动执行PF_Vision Callback。	属于选件。 启用PF_Vision回调函数，并对视觉操作进行自定义。 有关PF_Vision回调函数，请参阅以下内容。 PF_Vision

3.2.2.7 视觉校准序列

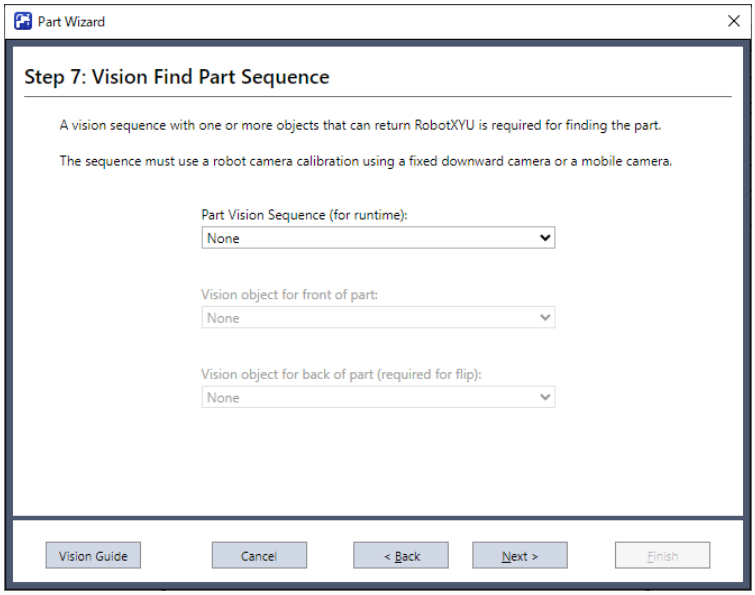
设定用于送料器校准的视觉序列。



项目	描述
部件Blob视觉序列	必须设置：请务必设置该项目。 选择送料器校准使用的视觉序列名称。
部件Blob视觉对象	必须设置：请务必设置该项目。 选择送料器校准时使用送料器背光灯以检测部件的视觉对象名称。 仅可指定Blob。

3.2.2.8 部件检测视觉序列

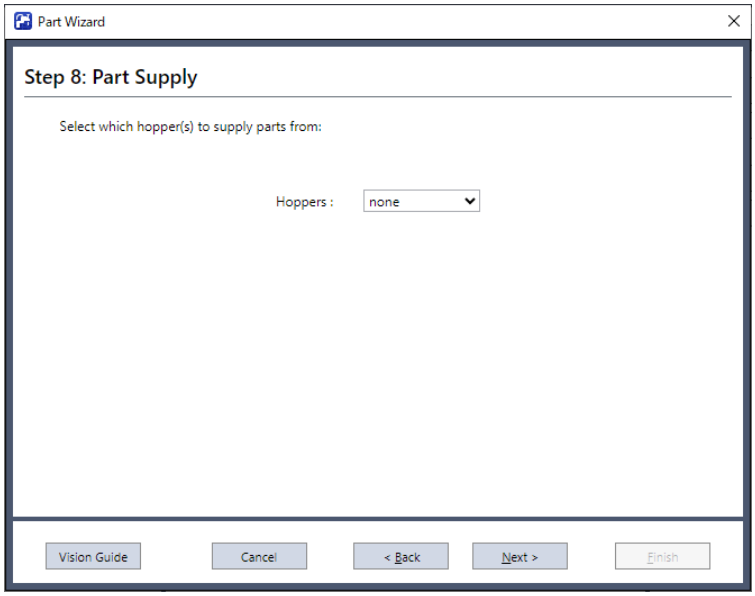
进行用于部件检测的视觉设定。



项目	描述
部件视觉序列（for runtime）	必须设置：请务必设置该项目。 选择用于检测部件的视觉序列名称。 仅显示已进行机器人校准的视觉序列。
检测表面部件的视觉对象	必须设置：请务必设置该项目。 选择用于检测拾取部件的视觉对象名称。 仅显示返回RobotXYU结果的对象。
检测背面部件的视觉对象	要进行翻转时（请参阅“常规”），请务必设定该项目。 选择用于检测因背向等而处于无法拾取姿势的部件的视觉对象名称。 仅显示返回RobotXYU结果的对象。

3.2.2.9 供给部件

进行料斗设定。



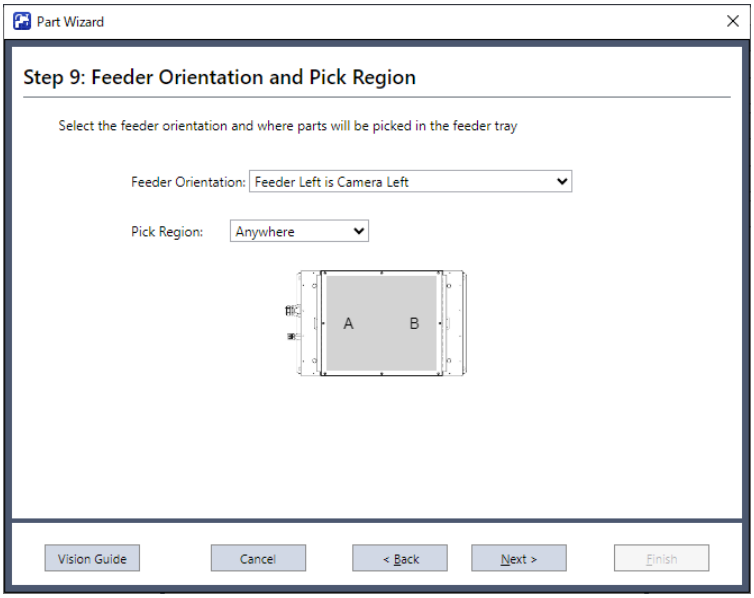
项目	描述
料斗	选择要使用的料斗。

要点

如果在未勾选“启用料斗”复选框的状态下选择要使用的料斗，则会显示料斗未连接的通知信息。

3.2.2.10 送料器的方向与拾取区域

进行送料器的配置与部件拾取相关设定。

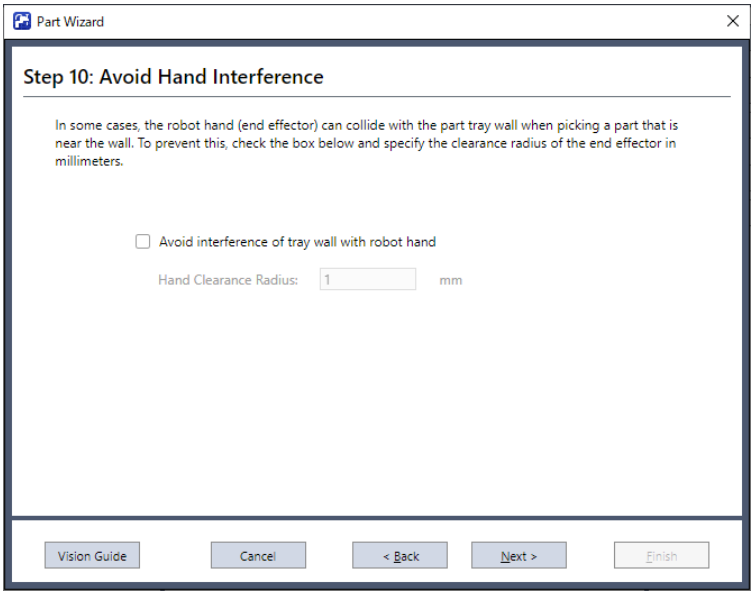


项目	描述
送料器的方向	设定通过相机图像查看时的送料器设置方向。

项目	描述
拾取区域	设定通过相机图像查看时的部件拾取区域。 选择全体或区域A、B、C、D之一。 可设定的区域因送料器机型而异。 IF-80：全体 IF-240、IF-A1520、IF-A2330：全体、区域A、区域B、区域C、区域D IF-380、IF-530：全体、区域A、区域B

3.2.2.11 防止末端夹具干扰

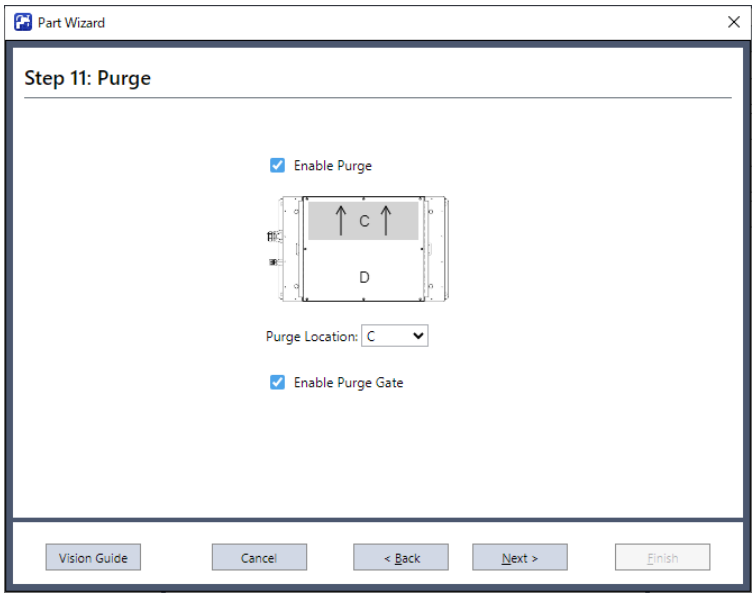
进行防止末端夹具与平台干扰回避功能相关的设定。



项目	描述
防止末端夹具碰撞	要启用防止末端夹具与平台碰撞功能时勾选。
末端夹具的旋转半径	与平台外圈（在视觉的搜索窗口中设定）之间处在该距离以内的部件不会进入坐标队列。 单位：mm

3.2.2.12 清除

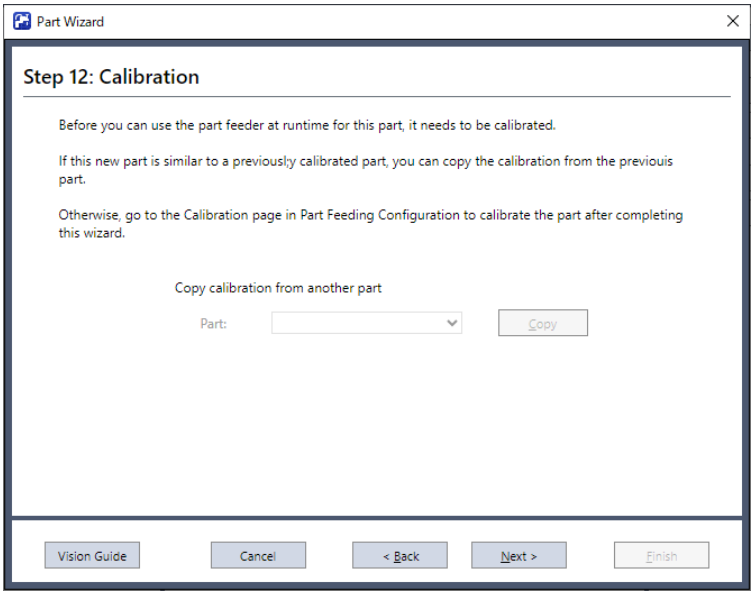
进行清除相关设定。



项目	描述
启用清除	要启用清除时勾选。
启用清除门	要使用清除门时勾选。
清除位置	设定清除方向。进行清除运作时，部件会向箭头方向移动。清除位置方面可选择A、C、D之一。可设定的项目因送料器类型而异。 IF-80：A IF-240，IF-380，IF-530，IF-A1520，IF-A2330：C，D

3.2.2.13 送料器校准

进行校准设定。

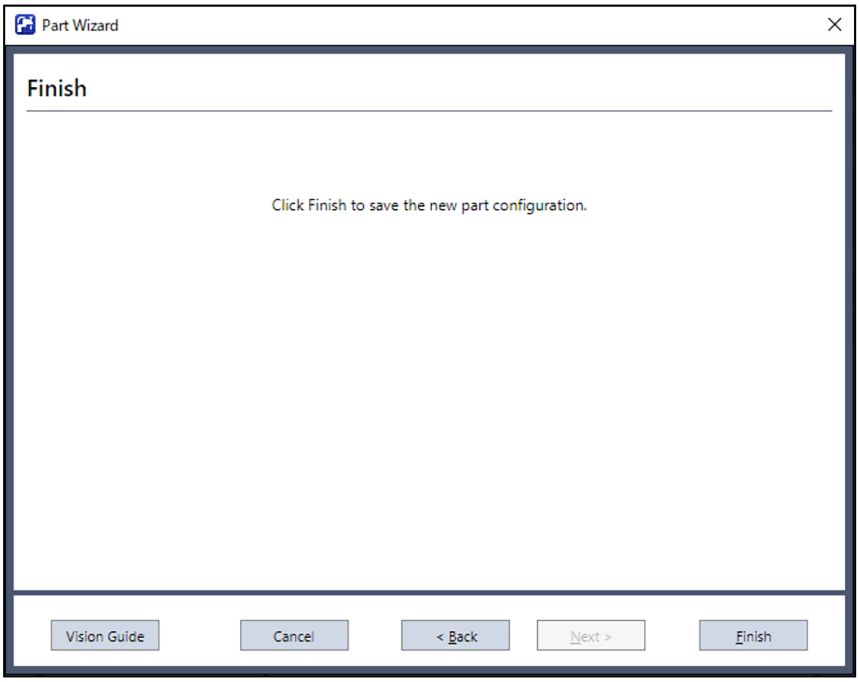


项目	描述
从其他部件复制校准结果	从其他部件向该部件复制校准结果。

项目		描述
	部件	指定复制源部件。
	复制	执行复制。

3.2.2.14 完成

单击[完成]按钮，退出部件向导。



3.2.3 上料对话框

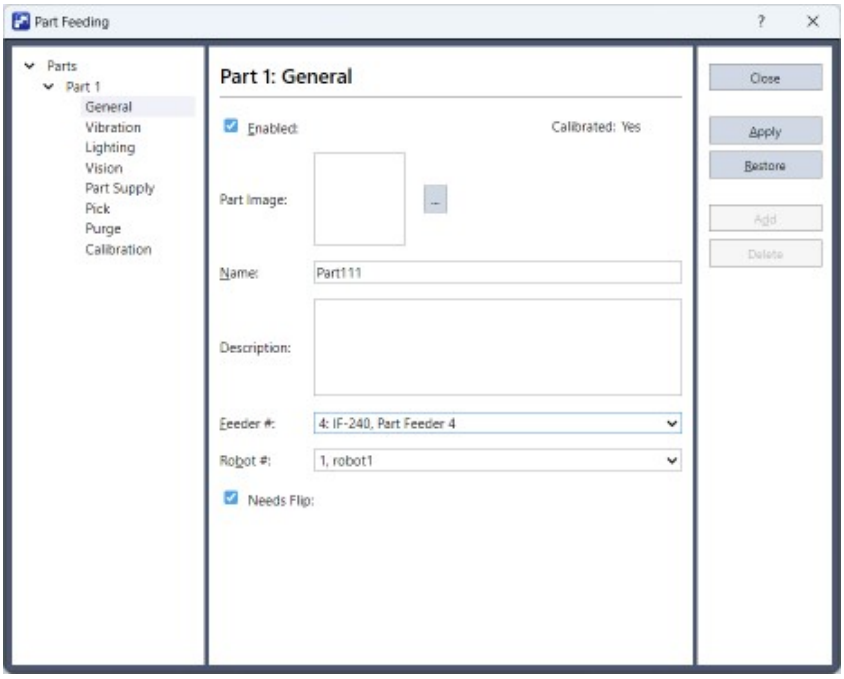
在Epson RC+ 8.0菜单 - [工具] - [上料]中进行Part Feeding的各种设定、送料器校准、调整与测试。

要点

请将Epson RC+连接至控制器。如果未处于将Epson RC+连接至控制器的状态，则不会显示上料窗口。如果在虚拟控制器中或离线状态下打开Part Feeding窗口，会发生错误。需要启用Part Feeding许可证。

3.2.3.1 常规

进行基本设定。



项目	描述
启用	要启用该部件时，勾选复选框。如果指定无效部件并执行PF_Start命令，会发生错误。
部件图像	可注册部件的图像。 单击[…]按钮，选择部件的图像文件。 此处注册的图像并不用于图像处理。可注册客户易于识别的图像。
校准	需要：需要进行送料器校准。 完成：已完成送料器校准。
名称	记述部件名称。（半角字母数字与下划线。最多16个字符）
注释	填写部件的说明（注释）。 属于选件。 （最多256个字符）
送料器#	选择该部件使用的送料器编号。 在系统配置画面中确认送料器编号。
机器人	选择机器人编号。
需要翻转	部件具有表里等姿势时选择。启用更改拾取姿势的翻转动作。这样的话，1次送料器运作可获取的部件数可能会增加，并提高循环时间。

按钮	描述
关闭	关闭画面。
应用	应用编辑内容。
恢复	恢复编辑内容。
添加	将部件添加到树中。最多可添加32个部件。

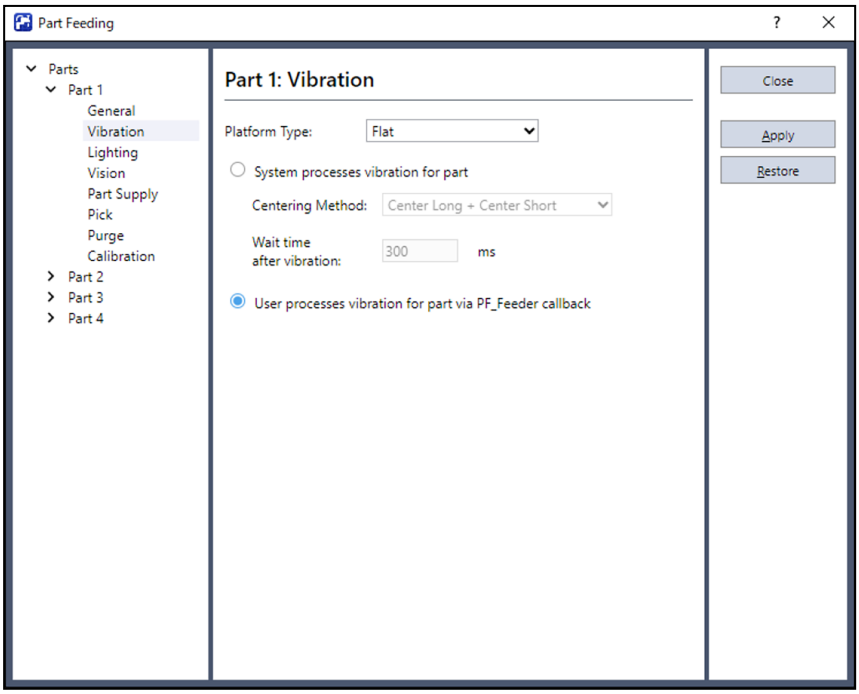
按钮	描述
删除	从树中删除部件。仅可删除树末尾的部件。 不能删除树中间的部件。取消勾选[启用]复选框，设为禁用。

要点

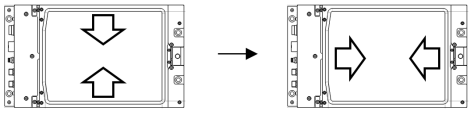
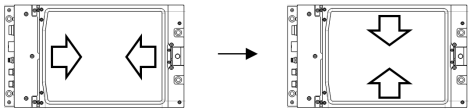
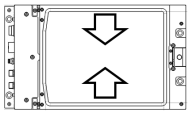
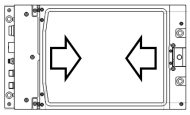
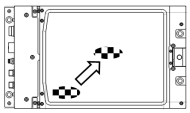
如果删除所有部件并进行创建，则会发生创建错误。这是因为无法使用上料命令或函数的缘故。在这种情况下，请对状态窗口中显示的相应行的注释进行非注释化处理。

3.2.3.2 振动

进行振动设定。



项目	描述
平台类型	指定平台的类型。 a) 是可从本公司采购的标准平台。 平面：平坦的平台 防滚动：已进行防滚动加工的平台 防粘贴：已进行防粘贴加工的平台（仅限于IF-80/240） b) 是客户独自制作的自定义平台。 长槽：已进行部件竖立用长槽加工的平台 孔：已进行部件竖立用孔加工的平台 定位槽：已进行部件方向对齐用孔加工的平台
由系统进行振动处理	由系统进行送料器控制。 已选择上述b)时，不能选择该选项。
通过PF_Feeder callback进行振动处理	使用PF_Feeder回调函数。

项目	描述
定芯方法	已选择[由系统进行振动处理]时，选择部件的定芯运作（投放部件时等部件分布偏差较大时，使部件靠向中央以进行均等分散的运作）类型。 无： 不进行定芯。 长轴定芯 + 短轴定芯： 向长轴方向进行定芯，然后向短轴方向进行定芯。  短轴定芯 + 长轴定芯： 向短轴方向进行定芯，然后向长轴方向进行定芯。  长轴定芯： 仅向长轴方向进行定芯。  短轴定芯： 仅向短轴方向进行定芯。  移动定芯： 通过移动动作进行定芯。 
振动后等待时间	指定送料器振动停止～通过视觉系统进行拍摄的等待时间（单位：毫秒）。无法通过视觉系统顺利地识别部件时，或机器人抓取送料器上的部件时产生位置偏差时，如果增大该值，则可能会对其有改善。

要点

定芯方法的最佳选项因部件类型或料斗位置而异。下述某种方法效果最佳。

- 长轴定芯 + 短轴定芯
- 短轴定芯 + 长轴定芯

但与其他定芯方法（或“无”时）相比，送料器运作时间会延长。如果选择适当分散部件并且定芯所需时间最短的选项，则会有一定的效果。

要点

选择“移动定芯”时，自动校准功能不起作用。

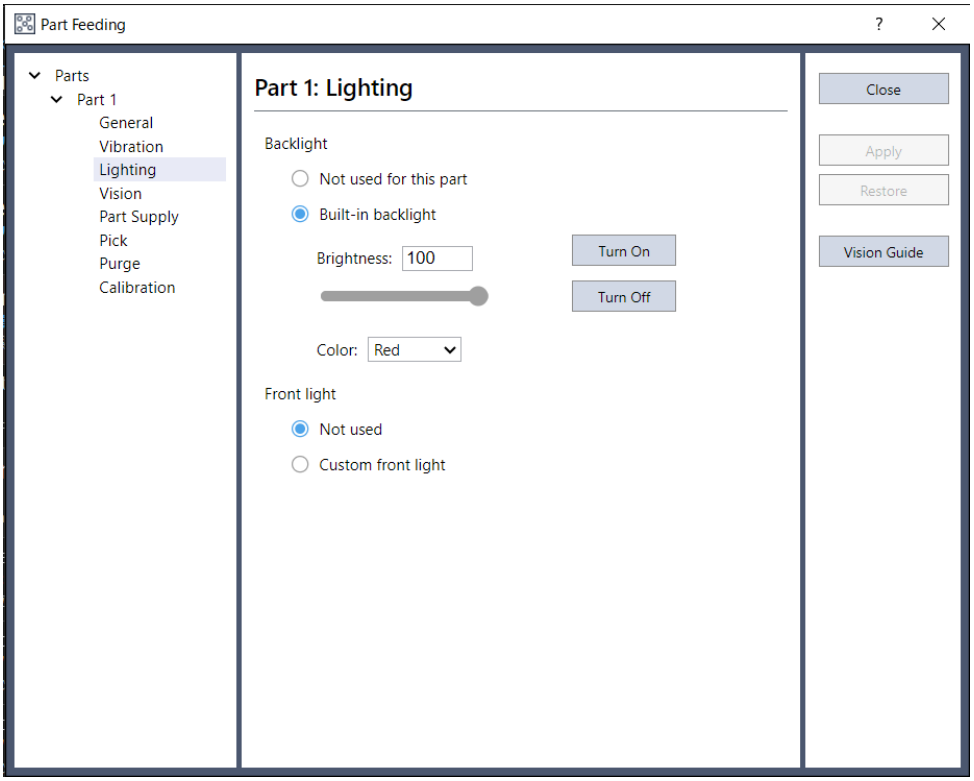
请参阅以下内容。

校准&测试

按钮	描述
关闭	关闭画面。
应用	应用编辑内容。
恢复	恢复编辑内容。

3.2.3.3 照明

进行照明设定。



项目		描述
背光灯		指定背光灯的控制方法。
	未使用	视觉系统拍摄时，在不使用送料器背光灯的情况下选择。 属于选件。
	内置背光灯	视觉系统拍摄时，在使用送料器背光灯的情况下选择。 没有背光灯时不能选择。
On		将送料器的背光灯设为ON。
Off		将送料器的背光灯设为OFF。
照度		设定送料器背光灯的亮度。 指定0～100%的值。
颜色		从“白色”、“红色”中选择背光灯的颜色。 (仅限于IF-A1520/23300)

前灯		指定前灯（选件）的控制方法。
	不使用	不使用前灯。
	自定义前灯	调用PF_Control回调函数，控制客户准备的自定义照明。属于选件。 有关PF_Control回调函数，请参阅以下内容。 PF_Control

要点

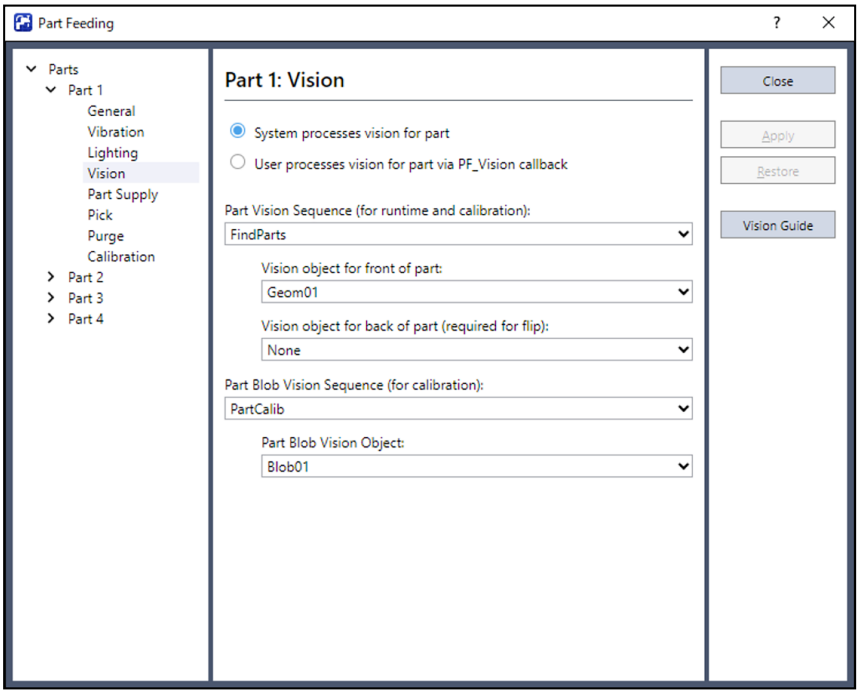
可利用百分比在0～100%的范围内设定照度，但可能会因送料器的型号而限制实际最小照度。请参阅以下内容。

PF_BacklightBrightness

按钮	描述
关闭	关闭画面。
应用	应用编辑内容。
恢复	恢复编辑内容。
视觉指南	显示Vision Guide画面。 可创建视觉序列。

3.2.3.4 视觉

进行视觉设定。
有关视觉序列的创建方法，请参阅以下内容。
Part Feeding选件使用的视觉序列



项目	描述
由系统进行视觉处理	系统自动进行视觉处理。 通常选择该项目。

项目	描述
通过PF_Vision callback进行视觉处理	属于选件。 启用PF_Vision回调函数，并对视觉操作进行自定义。有关PF_Vision回调函数，请参阅以下内容。 PF_Vision
部件检测视觉序列	必须设置：请务必设置该项目。 选择用于检测部件的视觉序列名称。仅显示已进行机器人校准的序列。
检测表面部件的视觉对象	必须设置：请务必设置该项目。 选择用于检测拾取部件的视觉对象名称。仅显示可返回RobotXYU结果的对象。
检测背面部件的视觉对象	要进行翻转时（请参阅“常规”），请务必设定该项目。选择用于检测因背向等而处于无法拾取姿势的部件的视觉对象名称。拾取表里双方时，也进行设定。仅显示可返回RobotXYU结果的对象。
部件Blob视觉序列	必须设置：请务必设置该项目。 选择送料器校准使用的视觉序列名称。仅显示已进行机器人校准的序列。
部件Blob视觉对象	必须设置：请务必设置该项目。 选择送料器校准时使用送料器背光灯以检测部件的视觉对象名称。仅可指定Blob。

要点

请务必指定必须设定的项目。

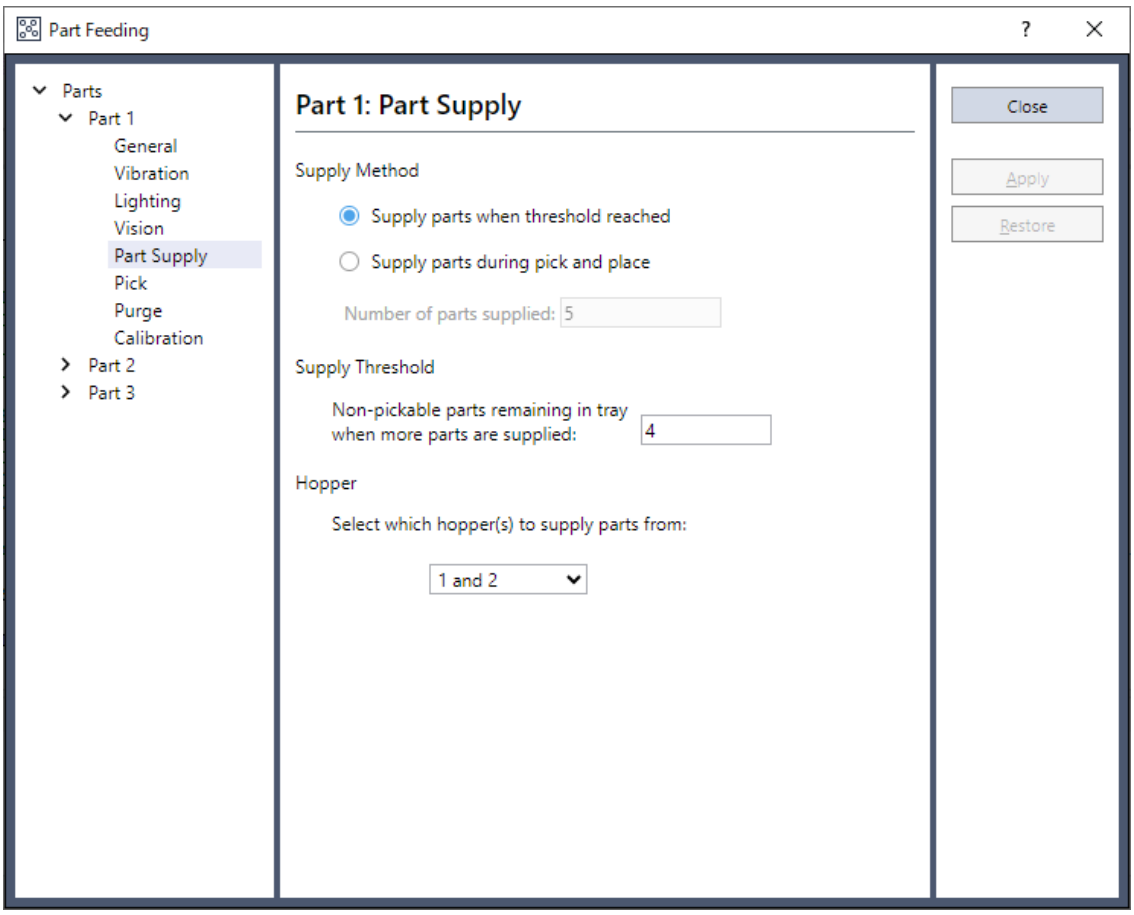
未设定时，校准运作或进程运作会发生错误。

按钮	描述
关闭	关闭对话框。
应用	应用编辑内容。 选件以外的项目中有未指定项目时，不能进行操作。
恢复	恢复编辑内容。
视觉指南	显示Vision Guide对话框 可创建视觉序列。

3.2.3.5 供给部件

设定向送料器供给部件的方式。由客户在PF_Control回调函数范围内编写料斗进给运作程序。有关PF_Control回调函数，请参阅以下内容。

PF_Control



项目	说明
达到供给阈值时供给部件	送料器上的部件数达到阈值时，供给部件。
拾取&放置期间供给部件	添加部件，以确保送料器上的部件始终处于最佳数量。 与隔开部件时相比，机器人的循环时间会缩短。
供给部件数	已选择“拾取&放置期间供给部件”时，输入从料斗添加部件的数量。 默认值为10。 有关部件数，请参阅以下内容。 料斗的调整方法
判断为需要供给的送料器上的部件余量	如果托盘上残留的不可拾取的部件数低于该值，则供给部件。 默认值为4。 已使用0时，会在部件为零之后供给部件。
料斗	选择要使用的料斗。

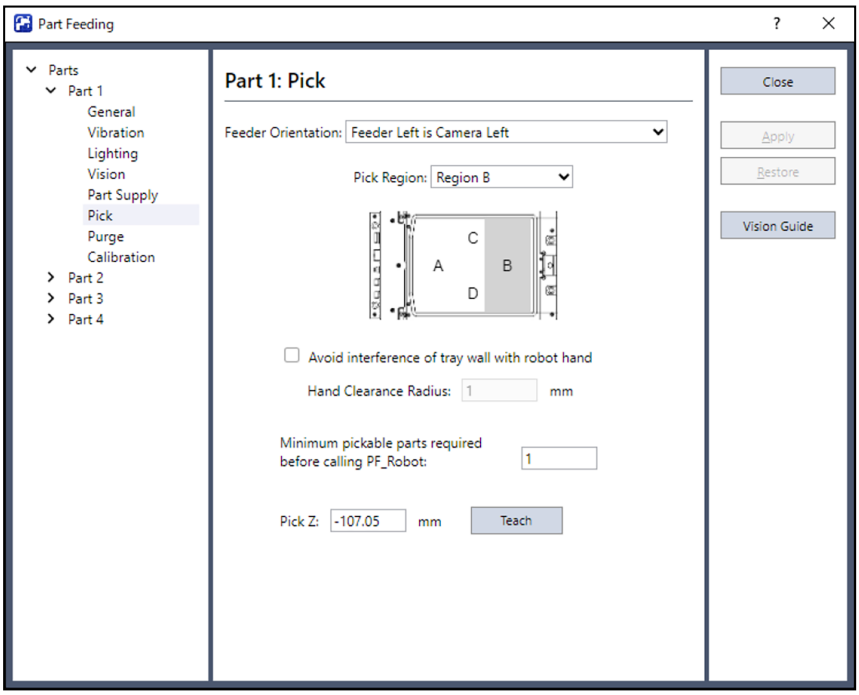
要点

- 如果在未勾选“启用料斗”复选框的状态下选择要使用的料斗，则会显示料斗未连接的通知信息。

- 可通过以下画面执行料斗测试。
  校准&测试

3.2.3.6 拾取

进行送料器的配置与部件拾取相关设定。



项目	描述
送料器的方向	设定通过相机图像查看时的送料器设置方向。
拾取区域	设定通过相机图像查看时的部件拾取区域。 选择全体或区域A、B、C、D之一。 如果选择区域A、B、C、D，则即使在校准期间也进行移动。 IF-80：全体 IF-240、IF-A1520、IF-A2330：全体、区域A、区域B、区域C、区域D IF-380、IF-530：全体、区域A、区域B
防止末端夹具碰撞	要启用防止末端夹具与平台碰撞功能时勾选。
末端夹具的旋转半径	与平台外圈（在视觉的搜索窗口中设定）之间处在该距离以内的部件不会进入坐标队列。单位：mm
调用PF_Robot之前所需的最小可拾取部件数	通常，即使只找到1个可拾取部件，也会调用PF_Robot callback。默认值为1。一般来说，送料器的振动可能花费时间，由于振动后未必能可靠地找到可选择的部件，因此无需更改设定。与该设定无关，通过视觉系统检测的实际部件数始终加载至部件队列。根据部件的数量与分布状况，确定振动方法。未达到调用PF_Robot之前所需的最小可拾取部件数时，系统会执行适当的运作。调用PF_Robot之前，如果送料器上有可拾取的最小数量的部件，有时可能会提高系统性能。
示教	显示示教画面，进行拾取部件时的Z坐标示教。 使用6轴型机器人时，除Z坐标外，也进行V坐标与W坐标的示教。 请参阅以下内容。 示教窗口

项目	描述
Z位置	是拾取部件时的Z坐标（本地坐标）。包括进行示教时的Z坐标。可输入数值进行更改。 单位：mm

要点

已勾选[防止末端夹具碰撞]复选框并调整[末端夹具半径]时，请确认末端夹具与平台之间不会发生干扰。

要点

在视觉画面中选择了[使用PF_Vision callback的系统视觉]时，即使启用[防止末端夹具碰撞]，距离平台末端[末端夹具半径]以内的部件，也可能会进入部件坐标队列。请注意。

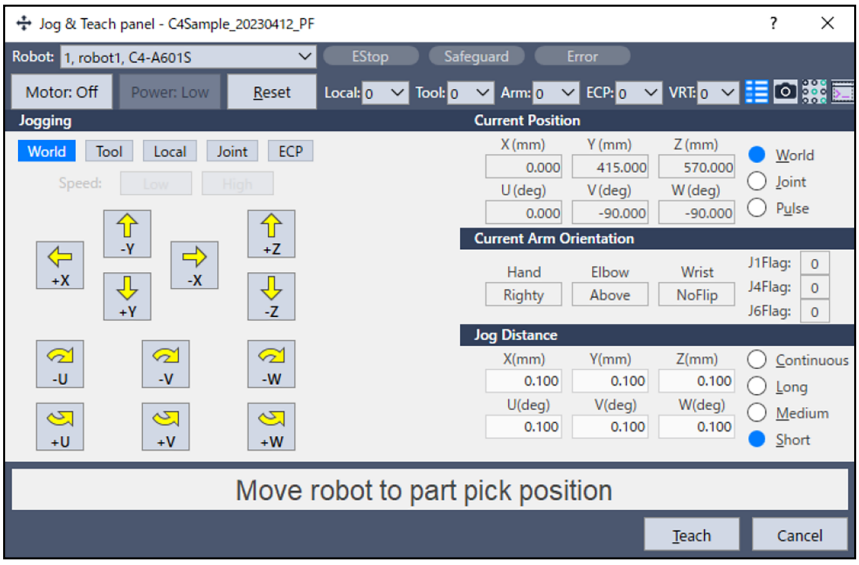
提示

使用6轴型机器人时的V坐标值与W坐标值不会显示在对话框中，但会被保存在内部。

按钮	描述
关闭	关闭画面。
应用	应用编辑内容。
恢复	恢复编辑内容。

3.2.3.7 示教窗口

进行拾取部件时的Z坐标示教。



1. 将1个部件放到平台上。
2. 操作Jog按钮（+X、-X、+Y、-Y），将机器人置于获取部件时的Z坐标与姿势。
3. 单击[OK]按钮。Z坐标、V姿势与W姿势则会被注册。

要点

此处选择的机器人是部件检测视觉序列中设定的视觉校准参照的机器人。

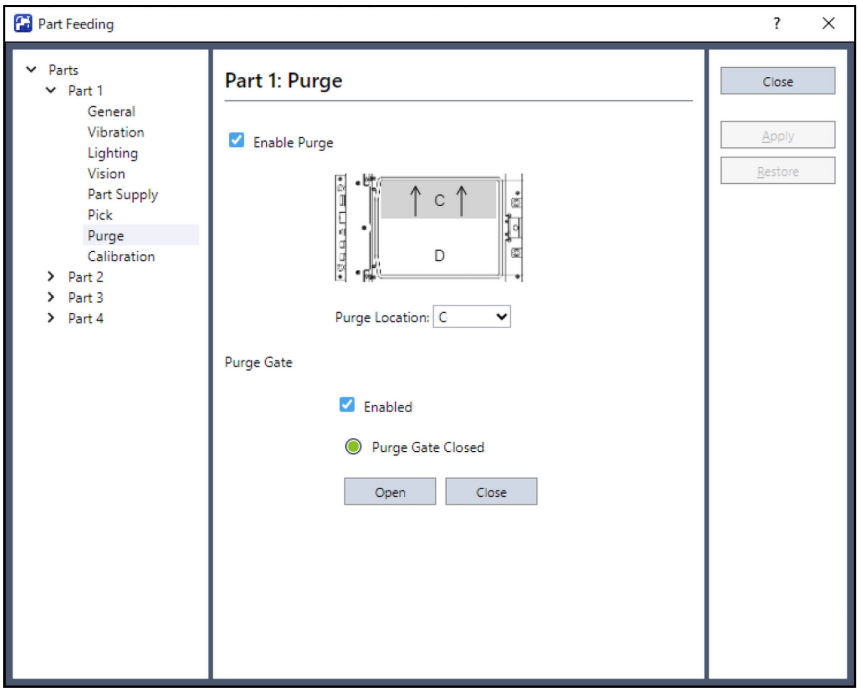
要点

在视觉画面中选择了[使用PF_Vision callback的系统视觉]时，需要客户将Z坐标的处理填写到PF_Vision回调函数内。

有关详细信息，请参阅以下内容。
PF_Vision

3.2.3.8 清除

进行清除设定。



项目		描述
启用清除		要启用清除功能时勾选。
清除位置		设定清除位置。进行清除运作时，部件会向箭头方向移动。清除位置方面可选择A、C、D之一。可设定的项目因送料器类型而异。 IF-80：A IF-240，IF-380，IF-530，IF-A1520，IF-A2330：C，D
清除门		进行清除门设定。
	启用	要启用部件的清除门运作时，勾选该复选框。默认设定时，如果Epson RC+ 8.0菜单 - [设置] - [系统设置] - [控制器] - [零件送料器]中的[安装清除门]复选框为ON，本复选框也会置为ON。
	关闭清除门	显示清除门传感器的状态。 关闭：绿色 未关闭：灰色

项目		描述
	打开/关闭	进行清除门的开闭运作。

要点

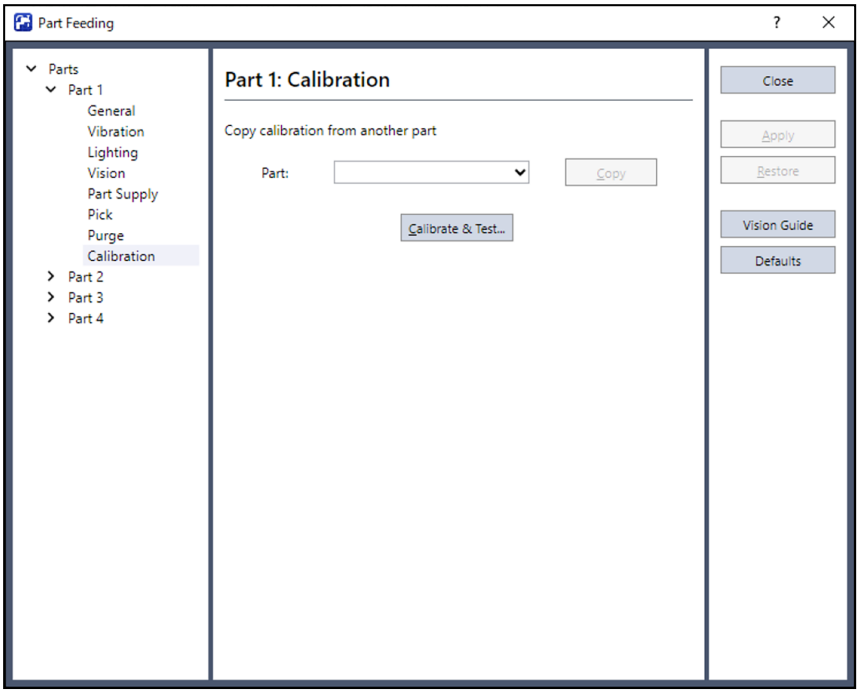
仅在Epson RC+ 8.0菜单 - [设置] - [系统设置] - [控制器] - [零件送料器]中的[安装清除门]复选框为ON时，才会显示[清除门]组框。

要点

如果要在清除门未关闭时切换为其他画面，则会显示对话框“关闭清除门。是否继续？”。如果按下OK，清除门则会关闭。

3.2.3.9 校准

进行送料器的校准、参数编辑与测试。



项目		说明
从其他部件复制校准结果		从其他部件向该部件复制校准结果。
	部件	指定复制源部件。
	复制	执行复制。
校准&测试		进行送料器的校准、参数编辑与测试。

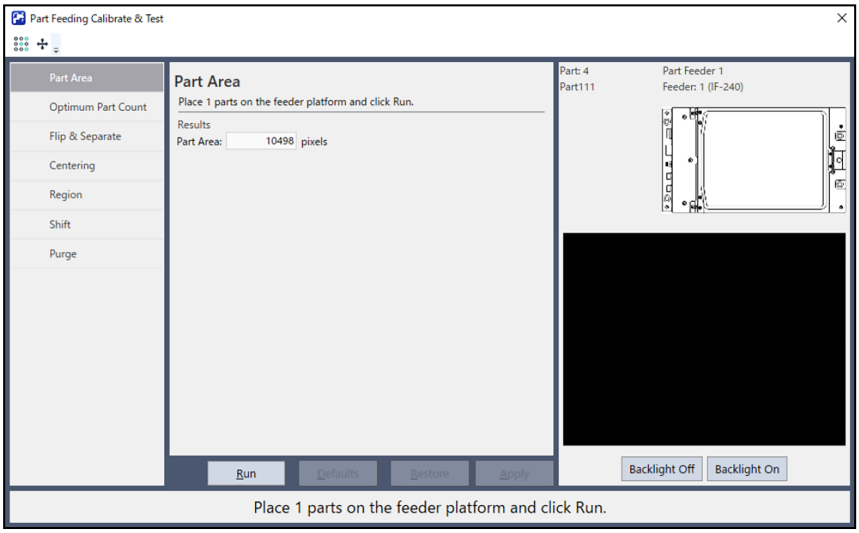
3.2.4 校准&测试

如果执行校准&测试，则可进行以下作业。

- 送料器的校准

- 送料器参数的调整
- 送料器运作测试

首先从左侧选项卡中选择要调整的送料器运作。



各选项卡的说明

选项卡	说明	校准必须/任意
部件区域	执行部件区域的校准。	必须
最佳投放部件数	进行最佳投放部件数的调整。	可选
分离	进行分离（使部件分散的运作）调整或测试。	可选
翻转&分离	进行翻转（改变部件姿势的运作）与分离（使部件分散的运作）调整或测试。 翻转有效时（请参阅“常规”）会显示。	可选
定芯	进行定芯（汇集部件的运作）调整或测试。	可选
区域	进行拾取区域（指定部件区域进行拾取时的部件移动运作）调整或测试。 在拾取画面（请参阅“拾取”）中将部件区域选为A、B、C、D区域之一时会显示。	可选
移动	进行移动（部件移动运作）调整或测试。	可选
料斗	进行料斗调整与测试。使用IF-80或已勾选“连接料斗”复选框时会显示。	（无）
清除	进行清除（从送料器中排出部件的运作）调整（仅限于IF-80）或测试。 在清除画面（请参阅“清除”）中启用清除时会显示。	可选

要点

新注册时，只能选择[部件区域]选项卡与[料斗]选项卡。

请首先选择[部件区域]选项卡，然后实施校准。

选择“拾取区域”时：请执行“区域”的自动校准。

使用IF-80并“启用”清除时：请执行“清除”的自动校准。

未进行自动校准时，使用默认值。

按钮的功能

按钮	说明
关闭	关闭窗口。
执行/中断	开始当前画面中的校准或测试。执行期间，显示会变为“中断”。如果单击[中断]按钮，则中断当前执行的运作。
应用	保存更改。
恢复	恢复已更改的值。
默认	将当前选择画面中的值恢复为默认值。
背光灯 ON/OFF	点亮或熄灭送料器的背光灯
 （I/O监视器）	启动I/O监视器。 控制自定义照明等的控制。
 （步进运作）	显示机器人步进进给运作窗口。 用于将移动相机移动至图像读取位置。

要点

在使用安全门的装置中单击[执行]按钮，并在运作期间打开安全门时，会停止送料器的运作。向导画面保持不变。

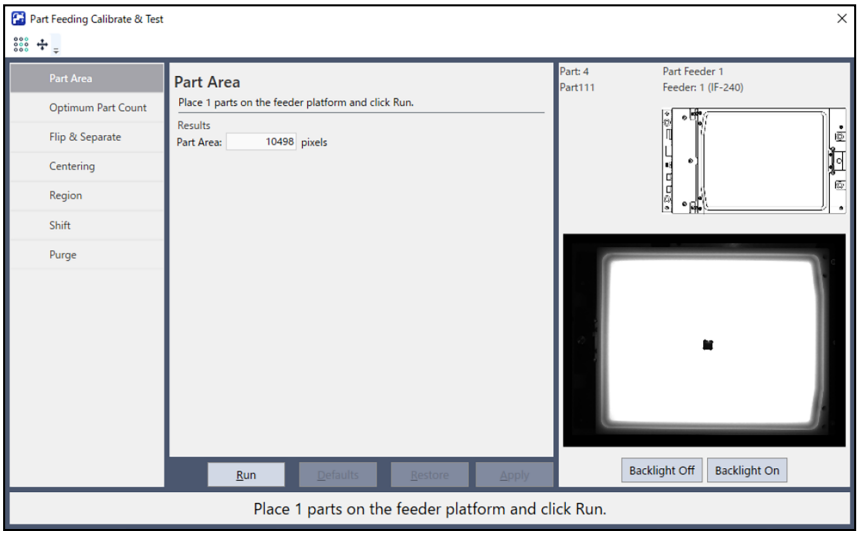
在这种情况下，请单击[中断]按钮，中断校准并关闭安全门，然后重新开始校准。

准备

- 请准备作为送料器校准对象的部件。
  部件需要具备一定的数量。有关数量，请参阅以下内容。
  最佳投放部件数
- 使用移动相机时：
  单击[步进动作]按钮，将机器人移动至图像读取位置。
- 使用IO控制的自定义照明时：
  请单击向导中显示的[I/O监视器]按钮，点亮照明。
- 新注册时：
  请首先执行“部件区域”校准。
  选择[部件区域]选项卡，然后单击[执行]按钮，实施校准。

3.2.4.1 部件区域

执行部件区域（每个部件的像素数）的校准。



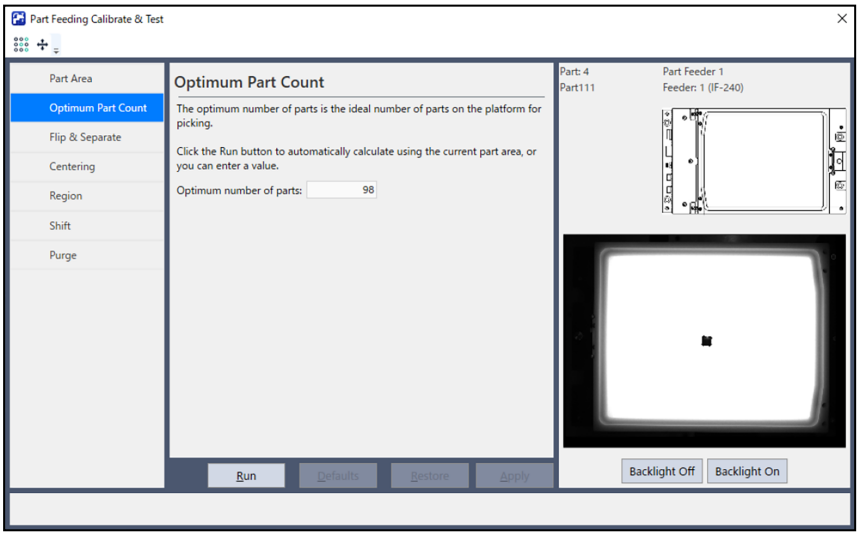
1. 显示项目的说明

项目	说明
部件面积	部件的像素数

2. 校准步骤
   (1) 将1个部件放到平台上。
   (2) 单击[执行]按钮。测量部件区域的像素数。
   (3) 单击[应用]按钮。结果会被保存。

3.2.4.2 最佳投放部件数

计算最佳投放部件数。



显示项目的说明

项目	说明
最佳投放部件数	最佳投放部件数的计算值 可编辑

调整指南

本参数是假设部件接近正方形或圆形计算出来的。因此，为细长形状或中间有空洞部分时，设定比校准值小的值。
与料斗一起使用时，根据该值判断是否需要添加部件。感觉投放部件的数量较少时，设定比当前大的值。

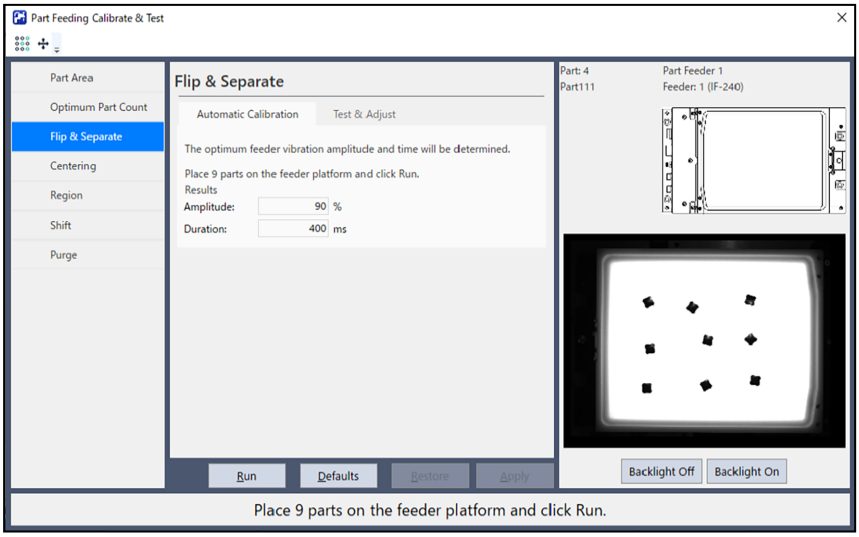
校准步骤

1. 单击[执行]按钮。计算最佳投放部件数。
2. 单击[应用]按钮。结果会被保存。

3.2.4.3 翻转&分离 - 自动校准

执行翻转与分离的校准。翻转无效时，该项目会显示为“分离”。
有关详细信息，请参阅以下内容。
常规

请在以下画面中选择“自动校准”选项卡。



要点

将平台类型设为“长槽”、“孔”、“定位槽”时，不显示[自动校准]选项卡。
有关详细信息，请参阅以下内容。
常规

显示项目的说明

项目	说明
振动振幅	显示通过自动校准获取的振动的振幅强度。 单位：%
振动时间	显示通过自动校准获取的振动持续时间。 单位：ms

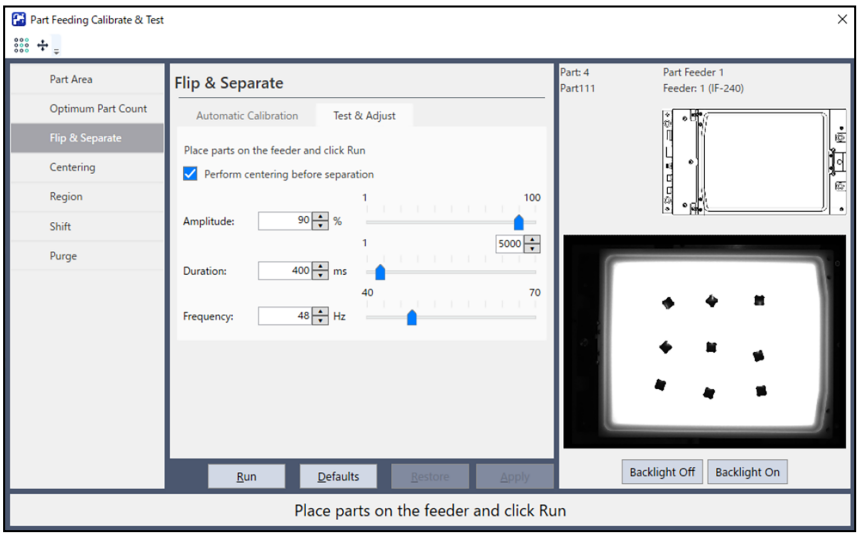
校准步骤

1. 投放显示数量的部件。
2. 单击[执行]按钮。
   尝试进行送料器振动运作，显示最佳的振动振幅与振动时间。
   运作时间受部件的物理特性（重量、大小、材质、表面摩擦等）的影响而需要数分钟左右。

3. 单击[应用]按钮。保存结果。

3.2.4.4 翻转&分离 – 测试与调整

调整翻转与分离的运作参数。翻转无效时（请参阅常规），该项目会显示为“分离”。
选择[测试与调整]选项卡。



显示项目的说明

项目	说明
分离之前定芯	如果勾选复选框，则会在进行测试运作时，于分离运作之前，执行定芯运作。 仅适用于测试运作。
振动振幅	设定振动的振幅强度。 单位：%
振动时间	设定振动的持续时间。 单位：ms
振动频率	设定振动的频率。 单位：Hz

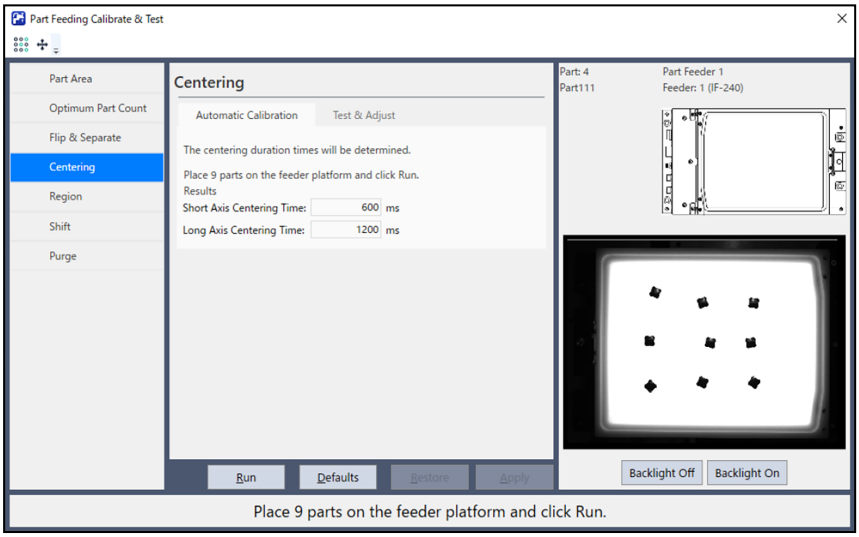
调整指南
调整振幅与频率，以确保部件可尽快分散并更改姿势，并且部件不会从平台弹出。
将完全更改分散或姿势的最短时间设为振动时间。

测试步骤

1. 投放适当数量（例：最佳投放部件数）的部件。
2. 单击[执行]按钮。
3. 确认运作。
   根据需要调整参数，然后再次单击[执行]按钮。
   请参阅以下内容。
   送料器参数的调整方法

3.2.4.5 定芯 - 自动校准

执行定芯校准。
选择[自动校准]选项卡。



要点

- 将平台类型设为“长槽”、“孔”、“定位槽”时，不显示[自动校准]选项卡。有关详细信息，请参阅以下内容。
  振动
- IF-80时不显示[自动校准]选项卡。

显示项目的说明

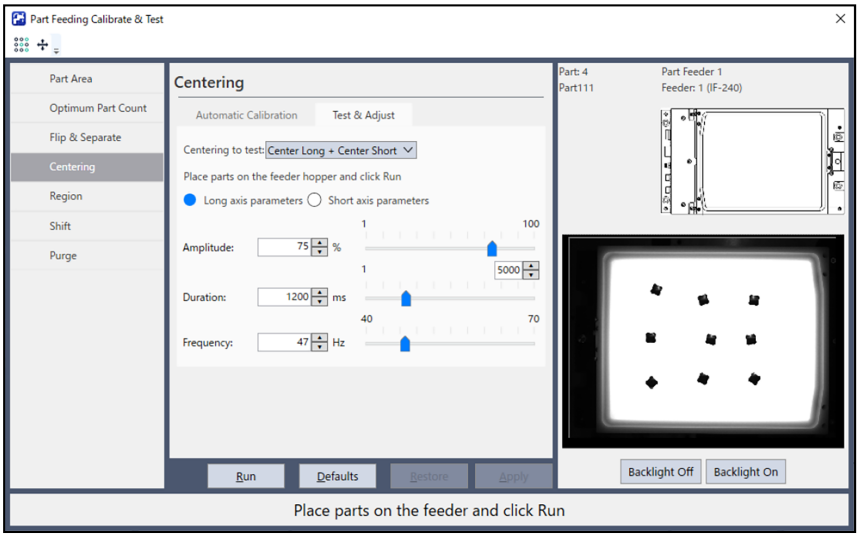
项目	说明
短轴定芯时间	显示短轴定芯的持续时间。 单位：ms
长轴定芯时间	显示长轴定芯的持续时间。 单位：ms

校准步骤

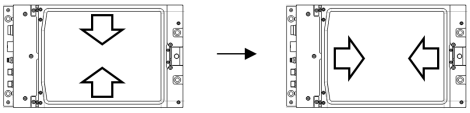
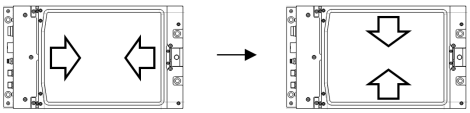
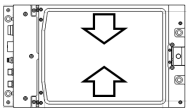
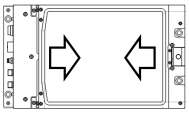
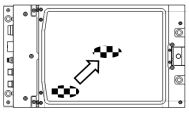
1. 投放显示数量的部件。
2. 单击[执行]按钮。
   尝试进行送料器振动运作，显示最佳的振动振幅与振动时间。
3. 单击[应用]按钮。结果会被保存。

3.2.4.6 定芯 - 测试与调整

调整定芯的运作参数。
请选择[测试与调整]选项卡。



显示项目的说明

项目	说明
定芯测试	选择要测试的定芯运作的类型。 长轴 + 短轴  短轴 + 长轴  长轴  短轴  移动定芯 
长轴参数与短轴参数	调整长轴与短轴中的某个运作的参数并选择。 选择“移动定芯”时，不会显示。
振动振幅	设定振动的振幅强度。 单位：% 选择“移动定芯”时，不会显示。

项目	说明
振动时间	设定振动的持续时间。 单位：ms 选择“移动定芯”时，不会显示。
振动频率	设定振动的频率。 单位：Hz 选择“移动定芯”时，不会显示。

调整指南

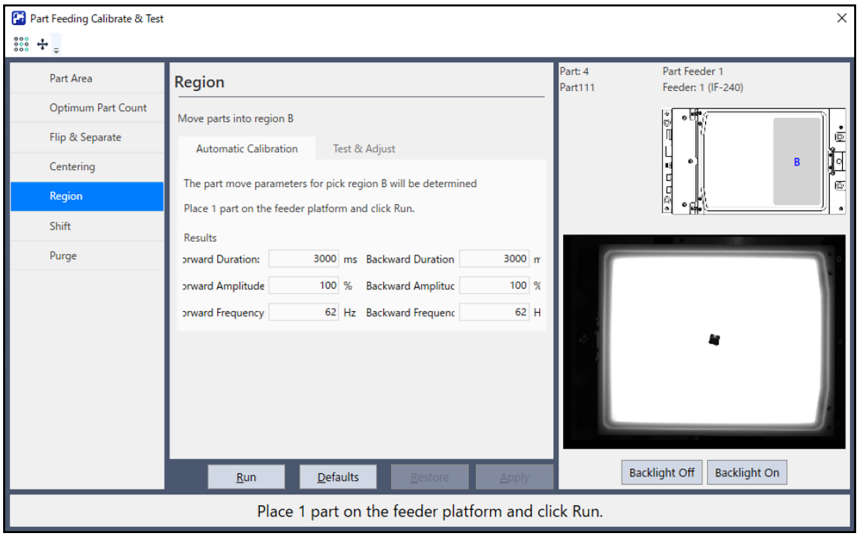
调整振幅与频率，以确保部件尽可能尽早向指定方向汇集。将汇集完成时间设为振动时间。

测试步骤

1. 投放适当数量（例：最佳投放部件数）的部件。
2. 单击[执行]按钮。
3. 确认运作。根据需要调整参数，然后再次单击[执行]按钮。
   请参阅以下内容。
   送料器参数的调整方法

3.2.4.7 区域 - 自动校准

执行拾取区域移动的校准。已指定拾取区域时，会显示该项目。
有关详细信息，请参阅以下内容。
拾取
请选择[自动校准]选项卡。



要点

将平台类型设为“长槽”、“孔”、“定位槽”时，不显示[自动校准]选项卡。
有关详细信息，请参阅以下内容。
振动

显示项目的说明

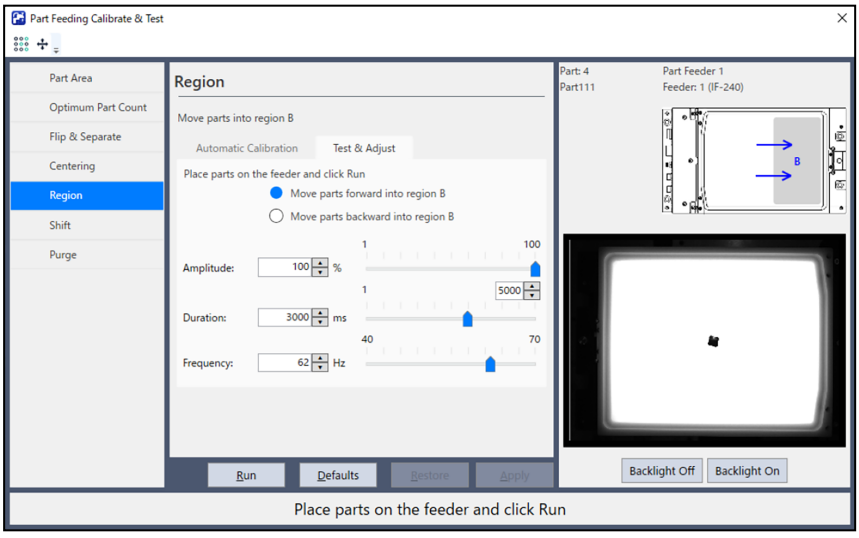
项目	说明
前移时间	显示前移动作的时间。 单位：ms
前移振幅	设定振动的振幅强度。 单位：%
前移频率	设定振动的频率。 单位：Hz
后移时间	显示后移运作的时间。 单位：ms
后移振幅	设定振动的振幅强度。 单位：%
后移频率	设定振动的频率。 单位：Hz

校准步骤

1. 投放1个部件。
2. 单击[执行]按钮。
   尝试进行送料器振动运作，显示最佳的振动振幅、振动时间与振动频率。
3. 单击[应用]按钮。结果会被保存。

3.2.4.8 区域 – 测试与调整

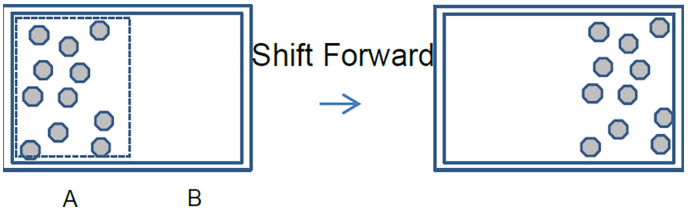
调整拾取区域移动的运作参数。已指定拾取区域时，会显示该项目。
有关详细信息，请参阅以下内容。
拾取
请选择[测试与调整]选项卡。



显示项目的说明

项目	说明
进行前移运作，移动至区域x；进行后移运作，移动至区域x	选择要测试移动的类型。x表示在拾取画面中选择的部件区域（A、B、C、D区域之一）。 移动方向因送料器的设置方向而异。 画面右上方会显示实际的移动方向。 有关详细信息，请参阅以下内容。 拾取
振动振幅	设定振动的振幅强度。 单位：%
振动时间	设定振动的持续时间。 单位：ms
振动频率	设定振动的频率。 单位：Hz

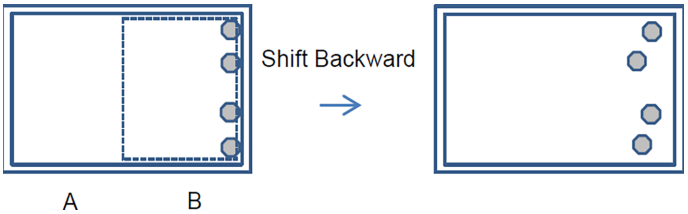
前移调整指南、测试步骤
前移是指从指定区域持续拾取部件，并在区域内的部件减少时，通过移动动作将部件移向指定区域的运作。
下图所示为向区域B前移的理想运作结果。



调整振动时间，以确保部件移动适当的距离（平台边长的一半）。如果过度移动部件，部件则会滞留在平台末端。另外，如果移动不足，则无法向区域内供给足够的部件，从而导致效率降低。调整振幅或频率，以确保部件适当地移动（在保持移动前部件分布的状态下移动）。如果部件的移动不顺利，则会碰到区域内的其余部件或产生重叠，导致效率降低。

1. 向显示区域投放适当数量（例：最佳部件投放部件数的1/2）的部件。
2. 单击[执行]按钮。
3. 确认运作。
   根据需要调整参数，然后再次单击[执行]按钮。请参阅以下内容。
   送料器参数的调整方法

后移调整步骤指南、测试步骤　后移是指重复进行前移运作，并在部件滞留于平台边角时，通过移动运作将部件移向（返回）指定区域的运作。下图所示为向区域B后移的理想运作结果。



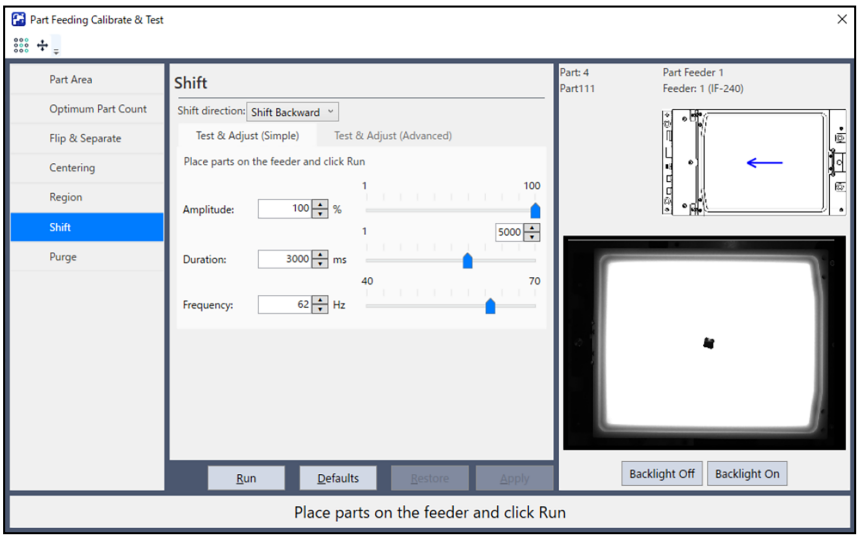
调整振动时间（与振幅或频率），以确保部件移动适当的距离（碰到平台末端的部件返回至可拾取位置的程度）。如果过度移动部件，部件则会被移出区域，导致效率降低。

1. 向显示区域的边角（平台的边角）投放适当数量（例：4个左右）的部件。
2. 单击[执行]按钮。
3. 确认运作。
   根据需要调整参数，然后再次单击[执行]按钮。

请参阅以下内容。
送料器参数的调整方法

3.2.4.9 移动 – 测试与调整（简易）

调整移动的运作参数。
请选择[测试与调整（简易）]选项卡。



显示项目的说明

项目	说明	
移动方向	选择要测试移动的方向。	
	移动方向	**部件移动方向**
	前	
	左前	
	右前	
	左	
	右	
	后	
	左后	
	右后	
	移动方向因送料器的设置方向而异。画面右上方会显示实际的移动方向。	
振动振幅	设定振动的振幅强度。 单位：%	

项目	说明
振动时间	设定振动的持续时间。 单位：ms
振动频率	设定振动的频率。 单位：Hz

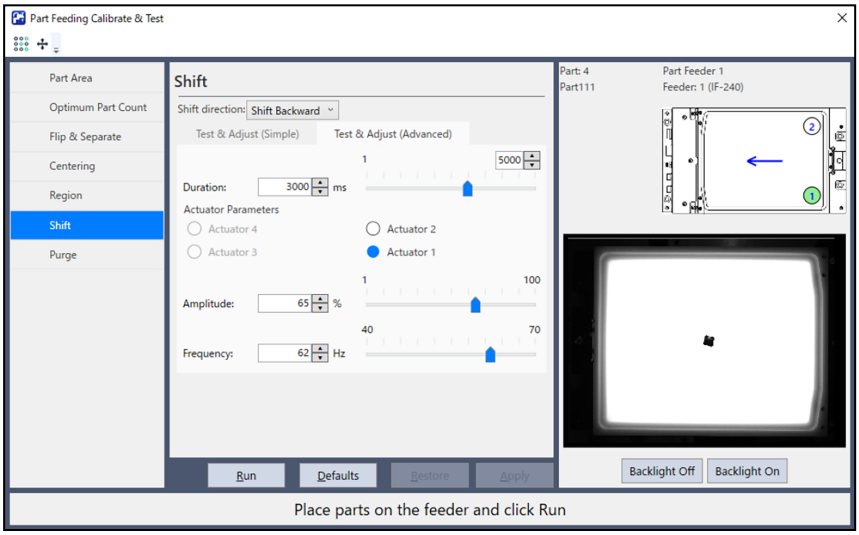
调整指南
调整振幅与频率，以确保部件尽快且顺利地移向指定方向。
将完成预期运作的时间设为振动时间。

测试步骤

1. 投放适当数量（例：4个左右）的部件。
2. 单击[执行]按钮。
3. 确认运作。
   根据需要调整参数，然后再次单击[执行]按钮。
   请参阅以下内容。
   送料器参数的调整方法

3.2.4.10 移动 - 测试与调整（详细）

调整移动的运作参数。
请选择[测试与调整（详细）]选项卡。



显示项目的说明

<table>
<tr><th>项目</th><th colspan="2">说明</th></tr>
<tr><td rowspan="10">移动方向</td><td colspan="2">选择要测试移动的方向。</td></tr>
<tr><td>部件移动方向</td><td>运作</td></tr>
<tr><td>前</td><td>  </td></tr>
<tr><td>左前</td><td>  </td></tr>
<tr><td>右前</td><td>  </td></tr>
<tr><td>左</td><td>  </td></tr>
<tr><td>右</td><td>  </td></tr>
<tr><td>后</td><td>  </td></tr>
<tr><td>左后</td><td>  </td></tr>
<tr><td>右后</td><td>  </td></tr>
<tr><td colspan="2">移动方向因送料器的设置方向而异。画面右上方会显示实际的移动方向。</td></tr>
<tr><td>振动时间</td><td colspan="2">设定振动的持续时间。
所有执行器通用该值。
单位：ms</td></tr>
<tr><td>执行器1
执行器2
执行器3
执行器4</td><td colspan="2">选择要设定的执行器（送料斗内部的振动体）。
画面右上方会显示执行器的位置。</td></tr>
<tr><td>振动振幅</td><td colspan="2">设定振动的振幅强度。
单位：%</td></tr>
<tr><td>振动频率</td><td colspan="2">设定振动的频率。所有执行器通用该值。
单位：Hz</td></tr>
</table>

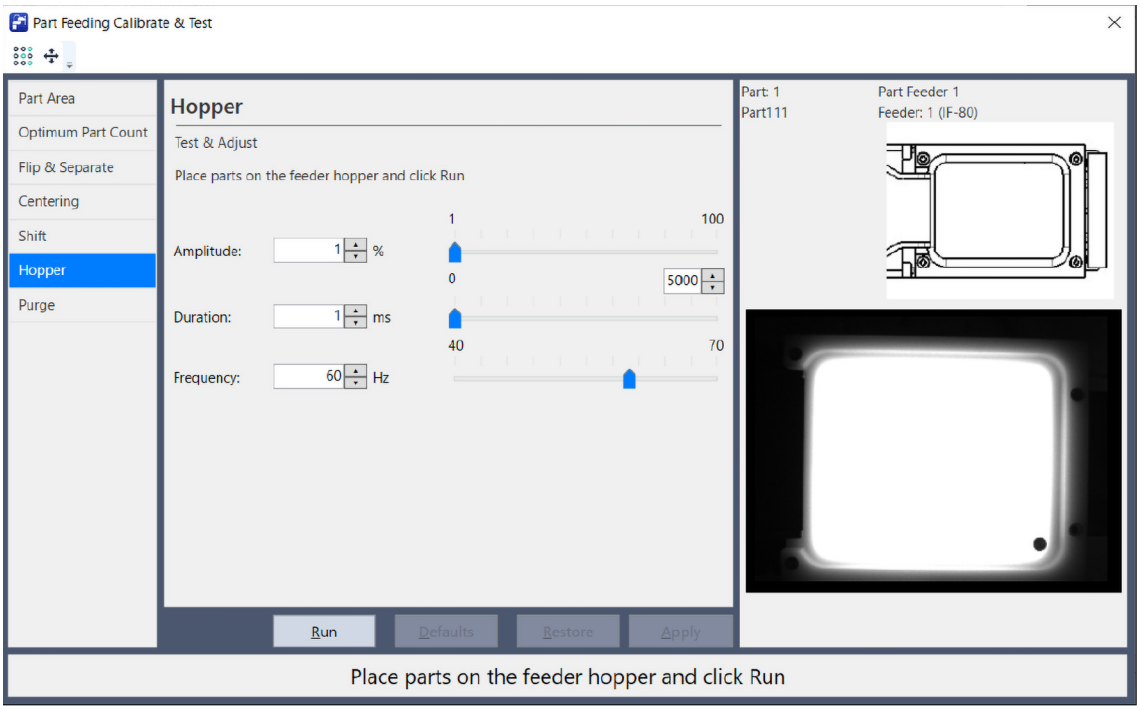
调整指南　调整振幅与频率，以确保部件尽快且顺利地移向指定方向。将完成预期运作的时间设为振动时间。

测试步骤

1. 投放适当数量（例：4个左右）的部件。
2. 单击[执行]按钮。
3. 确认运作。
   根据需要调整参数，然后再次单击[执行]按钮。
   请参阅以下内容。
   送料器参数的调整方法

3.2.4.11 料斗 - 测试与调整

使用大于等于1个的料斗时，可使用料斗 - 测试与调整画面。
请参阅以下内容。
料斗的调整方法



显示项目的说明

项目	说明
振动振幅（仅限于IF-80料斗 (Gen.2)）	设定振动的振幅强度。 单位：%
振动时间	设定测试时的振动持续时间。 单位：ms
振动频率（仅限于IF-80）	设定振动的频率。 单位：Hz

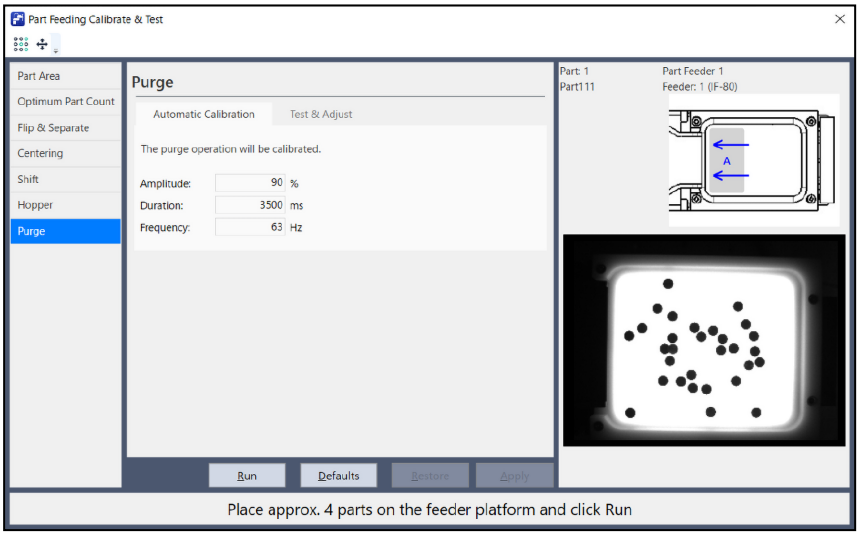
调整指南
有关调整的详细信息，请参阅以下内容。
料斗的调整方法

要点

- 连接2个料斗时，料斗画面中会显示测试与调整使用哪个料斗的选择下拉菜单。
- 已更改设定时，在更改料斗编号之前单击[应用]按钮。适用更改之前，料斗编号已被更改时，会显示适用更改或恢复原样的选择信息。

3.2.4.12 清除 - 自动校准（仅限于IF-80）

执行IF-80的清除（排出送料器上的部件）校准。启用清除时，会显示该项目。
有关详细信息，请参阅以下内容。
清除
请选择[自动校准]选项卡。



要点

将平台类型设为“长槽”、“孔”、“定位槽”时，不显示[自动校准]选项卡。
有关详细信息，请参阅以下内容。
振动

显示项目的说明

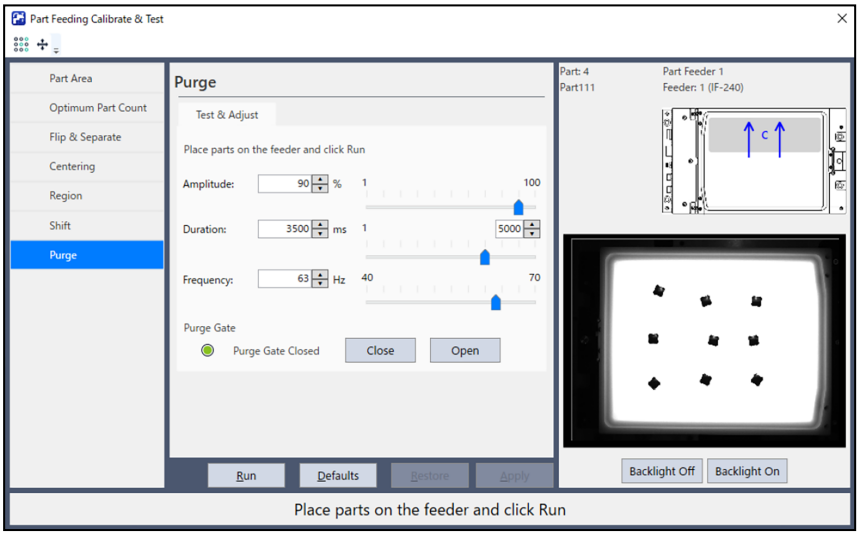
项目	说明
振动振幅	设定振动的振幅强度。 单位：%
振动时间	设定振动的持续时间。 单位：ms
振动频率	设定振动的频率。 单位：Hz

校准步骤

1. 清空清除箱（选件）。
2. 投放显示数量的部件。
3. 单击[执行]按钮。
   尝试进行送料器振动运作，显示最佳的振动振幅与振动时间。
   运作时间受部件的物理特性（重量、大小、材质、表面摩擦等）的影响而需要数分钟左右。
4. 单击[应用]按钮。结果会被保存。

3.2.4.13 清除 - 测试与调整

调整清除（排出送料器上的部件）的运作参数。启用清除时，会显示该项目。
有关详细信息，请参阅以下内容。
清除



显示项目的说明

项目		说明
振动振幅		设定振动的振幅强度。 单位：%
振动时间		设定振动的持续时间。 单位：ms
振动频率		设定振动的频率。 单位：Hz
清除门		进行清除门的操作。
	关闭清除门	显示清除门传感器的状态。 关闭：绿色；未关闭：灰色
	打开/关闭	进行清除门的开闭运作。

要点

即使将振动振幅设为0%，送料器也进行若干振动。

这是规格。

要点

仅在Epson RC+ 8.0菜单 - [设置] - [系统设置] - [控制器] - [零件送料器]中的[安装清除门]复选框为ON时，才会在门关闭的状态下显示清除。

要点

如果要在清除门未关闭时切换为其他画面，则会显示对话框“关闭清除门。是否继续？”。如果按下OK，清除门则会关闭。

调整指南　调整振幅与频率，以确保尽快从送料器上排出部件。
将部件完全排出之前的时间设为振动时间。

测试步骤

1. 投放适当数量（例：最佳投放部件数）的部件。
2. 打开清除门（客户准备）。
   IF-80时，清空清除箱（选件）。
3. 单击[执行]按钮。
4. 确认运作。
   根据需要调整参数，然后再次单击[执行]按钮。
   请参阅以下内容。
   送料器参数的调整方法

3.2.4.14 送料器参数的调整方法

要点

已事先将各参数设为最佳值。因此，通常无需手动调整这些参数。

- 振动振幅
  通过设定较大的值，加快部件的移动速度。这样可缩短部件分散或移动的完成时间，有助于缩短循环时间。
  另一方面，如果过度增大数值，则可能会导致部件从平台弹出。建议设定可充分判断部件移动或分散状况的最小值。

- 振动时间
  通过设定较长的时间，增大部件的分散量或移动量。这样的话，有助于增加1次送料器运作可获取的部件数。
  另一方面，如果过度增大数值，循环时间则会延长。建议设定可充分判断部件移动或分散状况的最小值。

- 振动频率
  该值被预设为由平台重量与送料器内置弹簧的弹簧常数确定的共振频率。因此，处理有一定重量的部件（金属制产品等）时，通过设定略小于设定值的值，有时可能会改善运作。另外，使用客户制作的平台时，有时通过更改频率，可能会改善部件的动作状态。
  但如果更改该值，则可能会发生部件动作状态发生变化等副作用，因此请慎重更改该值。
  如果更改频率，则可能会导致振动声音增大。这不是故障。
  在意振动声音时，请更改频率，从噪音最大的频率（=共振频率）调低数Hz。

3.2.5 [文件]菜单

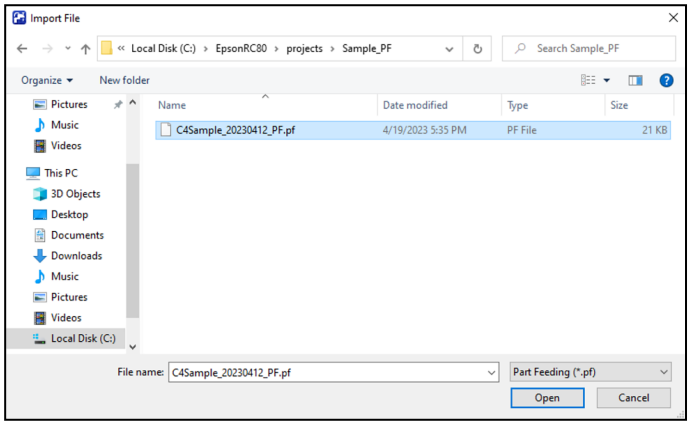
在Epson RC+ 8.0菜单 - [文件]中进行Part Feeding的各种文件操作。

3.2.5.1 [导入]（文件菜单）

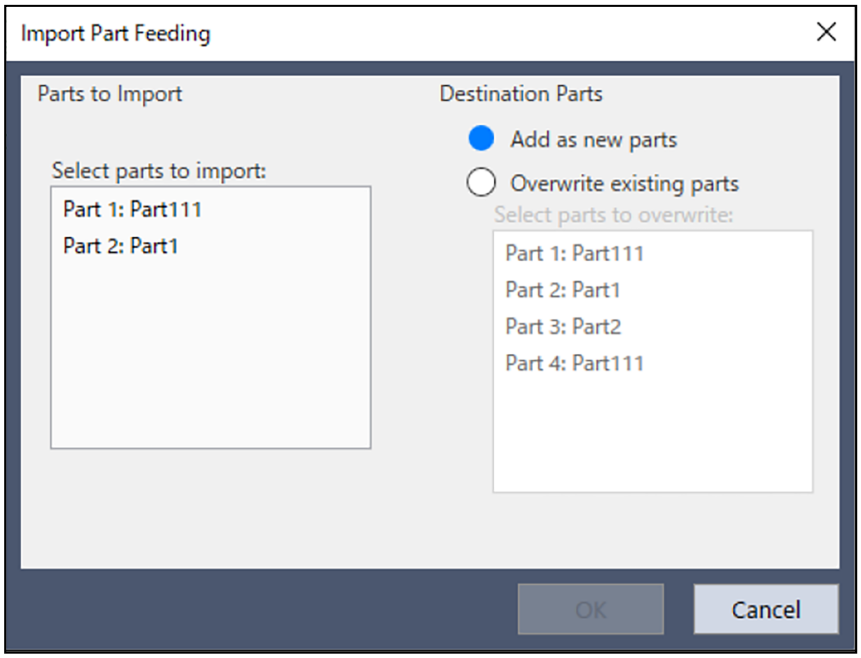
可从其他Epson RC+ 8.0项目中导入部件设定。

1. 选择Epson RC+ 8.0菜单 - [文件] - [导入]。

2. 在[文件类型]中选择“上料 (*.pf)”。



3. 在以下画面中进行导入操作。



项目	说明
选择要导入的部件	选择导入来源的部件。
添加新部件	要作为新部件添加时选择。
覆盖存在的部件	要将数据覆盖至现有部件时选择。
选择要覆盖的部件	已选择[覆盖存在的部件]时，选择要覆盖的部件。

3.3 Part Feeding SPEL+命令参考

下面说明客户的SPEL+程序可使用的Part Feeding SPEL+命令。如下所述为命令一览。

命令/函数	描述/用途
PF_Abort	强制结束Part Feeding的进程运作
PF_AccessFeeder	获取机器人访问送料器的锁定
PF_ActivePart	在多部件运作时切换有效部件
PF_Backlight	进行背光灯的ON/OFF运作

命令/函数	描述/用途
PF_BacklightBrightness	设定背光灯的亮度
PF_BacklightColor	更改送料器内置背光灯的颜色
PF_Center	执行定芯运作
PF_CenterByShift	执行移动定芯运作
PF_Flip	执行翻转运作
PF_Hopper	控制料斗
PF_Info函数	获取Part Feeding选件的属性
PF_InitLog	启用日志文件输出并指定输出目标路径
PF_IsStopRequested函数	返回是否发行PF_Stop
PF_Name$函数	从部件ID中获取部件名称
PF_Number函数	从部件名称中获取部件ID
PF_Purge函数	执行清除运作
PF_Output	控制送料器的输出端子（使用2台料斗 (Gen.1) 时）
PF_OutputOnOff	控制送料器的输出端子（使用1台料斗 (Gen.1) 时）
PF_PurgeGate	控制清除门的开闭运作
PF_PurgeGateStatus函数	获取清除门关闭传感器的状态
PF_QtyAdjHopperTime函数	返回适当的料斗运作时间
PF_QueAdd	在 坐标队列中 添加数据（点数据、姿势与用户数据）
PF_QueAutoRemove	设定坐标队列的自动删除功能
PF_QueAutoRemove函数	返回坐标队列自动删除功能的状态
PF_QueGet函数	从坐标队列中获取点数据
PF_QueLen函数	返回已注册到坐标队列中的数据数
PF_QueList	显示坐标队列的数据一览
PF_QuePartOrient	设定或显示坐标队列的部件姿势
PF_QuePartOrient函数	从坐标队列中获取部件姿势
PF_QueRemove	删除坐标队列的数据
PF_QueSort	设定或显示坐标队列的Sort方法
PF_QueSort函数	返回坐标队列的Sort方法
PF_QueUserData	设定或显示坐标队列的用户数据
PF_QueUserData函数	从坐标队列中获取用户数据
PF_ReleaseFeeder	解除通过PF_AccessFeeder获取的锁定
PF_Shift	执行移动运作

命令/函数	描述/用途
PF_Start	开始指定部件的Part Feeding进程
PF_Stop	发出Part Feeding进程的结束请求

3.3.1 PF_Abort

强制结束指定部件的Part Feeding进程运作。

格式
PF_Abort 部件ID

参数

- 部件ID
  指定部件ID（整数值1～32）。

返回值
无

描述
立即中断指定部件的Part Feeding进程。
与PF_Stop不同，正在执行的回调函数被中断。
即使未通过PF_Start开始Part Feeding运作进程，也使用PF_Abort停止平台或料斗的振动。
要在多部件运作中使用PF_Abort时，通过设定由PF_Start设定的某个ID，即会中断Part Feeding进程。
不能通过虚拟控制器与命令窗口执行。

使用示例

```
PF_Abort 1
```

3.3.2 PF_AccessFeeder

PF_AccessFeeder用于在多台机器人/1台送料器的系统中防止机器人之间的碰撞。2台机器人同时共享同一送料器时，需要使用该命令。PF_AccessFeeder用于获取访问送料器的机器人的锁定。
已获取锁定时，PF_AccessFeeder用于在锁定被解除之前或达到指定超时（选件）之前暂停任务。
机器人结束送料器的使用后，会使用PF_ReleaseFeeder语句解除锁定，以便将送料器释放给其他机器人。
有关详细信息，请参阅以下内容。
PF_ReleaseFeeder

示例：
在任务1中执行PF_AccessFeeder 1。继续执行任务1。
如果在任务2中执行PF_AccessFeeder 1，则会暂停任务2。
如果在任务1中执行PF_ReleaseFeeder 1，则会重新开始任务2。

本命令用于在多机器人构成中进行排他控制。

格式
PF_AccessFeeder 送料器编号/送料器名称[, 超时]

参数

- 送料器编号
  以表达式或数值（整数值1～4）指定送料器编号。

- 送料器名称
  指定送料器标签（字符串）。
- 超时
  以表达式或数值（整数值）指定等待解除锁定的超时（秒）。可省略。

返回值
无

描述
如果已指定超时，则可利用TW函数的返回值判断结果。
有关详细信息，请参阅以下手册。
“Epson RC+ 8.0 SPEL+语言参考 - TW函数”

- 锁定被解除或已获取锁定 (False)
- 达到超时 (True)

送料器的运作不会受锁定获取/解除的影响。
任务结束之后，通过该任务获取的锁定会被自动解除。
如果在同一任务中对同一送料器连续2次执行PF_AccessFeeder，则会发生错误。
不能通过虚拟控制器与命令窗口执行。

使用示例
下例所述为通过2台机器人拾取1台送料器上的部件的情况。
机器人1获取送料器上的部件，然后移动至点place1。
机器人1到达移动路径的50%位置时，机器人2开始移动，以获取部件。

```
Function Main

    MemOff PartsToPick

    Motor On

    PF_Start 1

    Xqt Robot1PickPlace
    Xqt Robot2PickPlace
Fend

Function Robot1PickPlace
    Robot 1
    Do
        If MemSw(PartsToPick) = On Then
            If PF_QueLen(1) > 0 Then
                PF_AccessFeeder 1
                P0 = PF_QueGet(1)
                PF_QueRemove 1
                Jump P0 /R
                On 5
                Wait 0.5
                Jump Place ! D30; PF_ReleaseFeeder 1 !
                Off 5
                Wait 0.25
            Else
                MemOff PartsToPick
            EndIf
        EndIf
        Wait 0.1
    Loop
Fend

Function Robot2PickPlace
```

```
    Robot 2
    Do
        If MemSw(PartsToPick) = On Then
            If PF_QueLen(2) > 0 Then
                PF_AccessFeeder 1
                P0 = PF_QueGet(2)
                PF_QueRemove (2)
                Jump P0 /R
                On 5
                Wait 0.5
                Jump Place ! D30; PF_ReleaseFeeder 1 !
                Off 5
                Wait 0.25
            Else
                MemOff PartsToPick
            EndIf
        EndIf
        Wait 0.1
    Loop
Fend

Function PF_Robot(PartID As Integer) As Integer
    Select PartID
        Case 1
            MemOn PartsToPick
            Wait MemSw(PartsToPick) = Off
        Case 2
            MemOn PartsToPick
            Wait MemSw(PartsToPick) = Off
    Send
    PF_Robot = PF_CALLBACK_SUCCESS
Fend
```

3.3.3 PF_ActivePart

在多部件运作时切换有效部件。PF_ActivePart语句用于向系统传递目前应处理哪个部件。由系统来进行部件振动或供给运作，以便机器人拾取由PF_ActivePart指定的部件。

格式

PF_ActivePart 部件ID

参数

- 部件ID
  指定部件ID（整数值1～32）。

返回值

无

描述

多部件运作时，PF_Start语句第1自变量的部件ID为最初的有效部件。要使用其他部件时，可使用PF_ActivePart语句切换有效部件。系统会针对有效部件进行运作（使用送料器振动参数，通过料斗供给部件等）。
通常在结束PF_Robot回调函数之前设定PF_ActivePart语句。
未开始Part Feeding运作进程时，即使执行本命令，也不会进行任何运作。
未进行多部件运作时（执行PF_Start时仅指定1个ID的情况），即使执行本命令，也不会进行任何运作。
指定了未在多部件运作中指定的部件ID时，即使执行本命令，也不会进行任何运作。
不能通过虚拟控制器与命令窗口执行。

使用示例

本例所述为使用PF_ActivePart切换部件1与部件2的方法。

```
Function PF_Robot(PartID As Integer) As Integer
   Select PartID
       Case 1
           If PF_QueLen(1) > 0 Then
               MemOn PartsToPick1
               Wait MemSw(PartsToPick1) = Off
               PF_ActivePart 2 'Switch to Part 2
           Else
               PF_ActivePart 1 'Part 1 is still needed
           EndIf
       Case 2
           If PF_QueLen(2) > 0 Then
               MemOn PartsToPick2
               Wait MemSw(PartsToPick2) = Off
               PF_ActivePart 1 'Switch to Part 1
           Else
               PF_ActivePart 2 'Part 2 is still needed
           EndIf
   Send
   PF_Robot = PF_CALLBACK_SUCCESS
Fend
```

3.3.4 PF_Backlight

将送料器内置的背光灯设为ON或OFF。

格式

PF_Backlight 送料器编号 | 送料器名称, On | Off

参数

- 送料器编号
  以表达式或数值指定送料器编号（整数值1～4）。
- 送料器名称
  以字符串指定送料器名称。
- On/Off
  指定On/Off。

返回值

无

描述

系统自动进行视觉处理时，背光灯会自动进行On/Off。
使用PF_Vision回调函数时，使用该命令进行背光灯的On/Off。
IF-80时，背光灯会在点亮30后自动熄灭。（时间因亮度而异。）
自动熄灭时，如果要再次点亮，则需执行PF_Backlight Off命令。
不能通过虚拟控制器与命令窗口执行。

使用示例

```
PF_Backlight 1, On
```

3.3.5 PF_BacklightBrightness

设定送料器内置背光灯的亮度。

格式
PF_BacklightBrightness 送料器编号 | 送料器名称, 亮度

参数
- 送料器编号
  以表达式或数值指定送料器编号（整数值1～4）。
- 送料器名称
  以字符串指定送料器名称。
- 照度
  以0%～100%的值（整数值）指定照度。

返回值
无

描述
通常在[上料]对话框中设定内置背光灯的照度。如果执行时需要更改亮度，则可利用该命令进行更改。
不能通过虚拟控制器与命令窗口执行。

要点

可利用百分比在0～100%的范围内设定照度，但使用IF-240时，即使设定小于等于25%的照度，实际最小照度值也会被限制为25%。这是因为小于等于一定照度时无法获得均匀的亮度。

如果要将照度设为0%，则请使用PF_Backlight语句将背光灯设为OFF。

使用示例

```
PF_BacklightBrightness 1, 80
```

3.3.6 PF_BacklightColor

更改送料器内置背光灯的颜色。
本命令专用于IF-A1520与IF-A2330。

格式
PF_BacklightColor 送料器编号 | 送料器名称, 颜色

参数
- 送料器编号
  以表达式或数值指定送料器编号（整数值1～4）。
- 送料器名称
  以字符串指定送料器名称。
- 颜色
  指定颜色。

值	颜色
PF_BACKLIGHTCOLOR_WHITE	白色
PF_BACKLIGHTCOLOR_RED	红色

返回值
无

描述
在[上料]对话框中设定内置背光灯的颜色。如果执行时需要更改颜色，则可利用该命令进行更改。不能通过虚拟控制器执行。

使用示例

```
PF_BacklightColor 1, 1
```

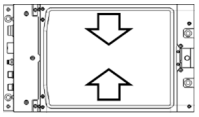
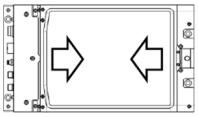
3.3.7 PF_Center

执行送料器的定芯运作。
可在IF-240、IF-380、IF-530、IF-A1520、IF-A2330使用。不能在IF-80中使用。

格式
PF_Center 部件ID, 方向 [, 运作时间]

参数

- 部件ID
  指定部件ID（整数值1～32）。
- 方向
  指定定芯方向。

方向	值（由PartFeeding.inc定义）	部件的移动方向
长轴	PF_CENTER_LONG_AXIS	
短轴	PF_CENTER_SHORT_AXIS	

- 运作时间
  以毫秒指定运作时间（整数值1～30000）。省略时，使用通过送料器校准计算出来的值。
  指定-1时，会进行与省略时相同的运作。

返回值
无

描述
执行IF系列送料器的定芯运作。
定芯运作用于下述情况。

- 分散部件前靠向中央
- 对齐细长部件的方向

可在客户的函数内直接使用本命令。另外，可在执行PF_Start命令期间调用的各回调函数内部使用。

不能在下述条件下执行。

- 通过用户函数执行时：通过命令指定的送料器（由部件ID指定的送料器）被Part Feeding进程（PF_Start命令）使用时（错误7733）
- 通过回调函数执行时：
  指定未通过PF_Start命令指定的部件ID时（错误7733）
- 通过虚拟控制器与命令窗口执行（错误3804）

本命令用于在内部处理中使用SyncLock。有关详细信息，请参阅以下内容。
Part Feeding进程使用的功能

使用示例

```
PF_Center 1,1
```

3.3.8 PF_Flip

执行翻转运作。

格式
PF_Flip 部件ID [, 运作时间]

参数

- 部件ID
  指定部件ID（整数值1～32）。
- 运作时间
  以毫秒指定运作时间（整数值1～30000）。
  省略时，使用通过送料器校准计算出来的值。
  指定-1时，会进行与省略时相同的运作。

返回值
无

描述
执行翻转运作。
翻转运作用于下述情况。

- 分散部件（运作时间较长）
- 更改部件姿势（运作时间较短）

不能在下述条件下执行。

- 通过用户函数执行时：通过本命令指定的送料器（由部件ID指定的送料器）被Part Feeding进程（PF_Start命令）使用时（错误7733）
- 通过回调函数执行时：指定未通过PF_Start命令指定的部件ID时（错误7733）
- 通过虚拟控制器与命令窗口执行（错误3804）

本命令用于在内部处理中使用SyncLock。有关详细信息，请参阅以下内容。
Part Feeding进程使用的功能

使用示例

```
PF_Flip 1,500
```

3.3.9 PF_Hopper

控制料斗。

格式
PF_Hopper 料斗编号_A, 部件ID_A [,运作时间_A] [,料斗编号_B] [,部件ID_B] [运作时间_B]

参数

- 料斗编号_A
  指定料斗编号（整数值1～2）。
- 部件ID_A
  指定部件ID（整数值1～32）。
- 运作时间_A
  可省略。指定料斗A的振动时间（整数值1～30000）。（毫秒单位）
  已省略运作时间_A或指定-1时，使用在上料校准画面中设定的值。已指定0时，料斗A不振动。已进行运作时停止振动。
- 料斗编号_B
  可省略。指定料斗编号（整数值1～2）。
- 部件ID_B
  可省略。指定部件ID（整数值1～32）。
- 运作时间_B
  可省略。指定料斗B的振动时间（整数值1～30000）。（毫秒单位）
  已省略运作时间_B或指定-1时，使用在上料校准画面中设定的值。已指定0时，料斗B不振动。已进行运作时停止振动。

返回值
无

描述
执行本命令后立即返还控制。
设为在部件ID_A中设定的部件使用料斗编号_A，在部件ID_B中设定的部件使用料斗编号_B。在上料对话框的部件供给画面中设定要用于部件的料斗。

执行PF_Abort命令时料斗停止。执行PF_Stop命令时不停止。

可在送料器振动的同时执行PF_Hopper。
IF-80时仅1个料斗对应于送料器，因此将使用IF-80时的料斗编号设为1。

不能通过虚拟控制器与命令窗口执行。
已选择系统配置中未启用的料斗编号时，会发生错误。

Note：PF_OutputOnOff、PF_Output仅针对料斗 (Gen.1) 起作用。
料斗 (Gen.2) 的控制需要PF_Hopper，其也可以用于料斗 (Gen.1) 的控制。

使用示例1：
针对部件9，进行3秒钟的料斗 (Gen.2) 1运作。在上料校准画面中设定料斗 (Gen.2) 的振动振幅。

```
PF_Hopper 1,9,3000
Wait 3  '等待至料斗运作结束。
```

使用示例2：
针对部件1，按上料校准对话框的料斗画面中指定的时间进行料斗2的运作。

```
PF_Hopper 2,1
```

使用示例3：
立即停止使用示例2的料斗2。

```
PF_Hopper 2,1,0
```

使用示例4：
同时进行下述运作：针对部件2，使料斗1进行500 ms的运作；针对部件3，使料斗2进行400 ms的运作。

```
PF_Hopper 1,2,500,2,3,400
```

使用示例5：
立即停止使用示例4的料斗1与2。

```
PF_Hopper 1,1,0,2,3,0
```

3.3.10 PF_CenterByShift

执行移动定芯运作。

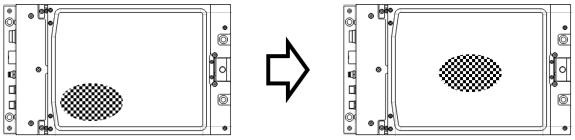
格式
PF_CenterByShift 部件ID

参数

- 部件ID
  指定部件ID（整数值1～32）。

返回值
无

描述
执行移动定芯运作。



定芯运作用于下述情况。

- 分散部件前靠向中央
- 末端夹具干扰平台时，将部件靠向中央

如果执行本命令，部件Blob视觉序列则会运作，测量送料器上部件的重心。进行移动运作，以确保部件重心位于送料器中央。
执行本命令时，如果部件已分布在送料器中央附近，有时送料器可能会停止而不进行运作。

不能在下述条件下执行。

- 通过用户函数执行时：
  通过本命令指定的送料器（由部件ID指定的送料器）被Part Feeding进程（PF_Start命令）使用时（错误7733）
- 通过回调函数执行时：
  指定未通过PF_Start命令指定的部件ID时（错误7733）
- 通过虚拟控制器与命令窗口执行（错误3804）

本命令用于在内部处理中使用SyncLock。有关详细信息，请参阅以下内容。
Part Feeding进程使用的功能

使用示例

```
PF_CenterByShift 1
```

3.3.11 PF_Info函数

获取部件的属性。

格式

(1) PF_Info（部件ID, 属性ID）
(2) PF_Info（部件名称, 属性ID）

参数

- 部件ID
  指定部件ID（整数值1～32）。
- 部件名称
  指定部件的名称（字符串）。
- 属性ID
  指定属性ID。
  如下所述为可获取的属性。

属性ID	内容
PF_INFO_ID_FEEDER_CALIB_CORRECT_MAXNUM	用于计算最佳数量校准的结果部件投放数量。
PF_INFO_ID_FEEDER_NO	返回用于部件的送料器编号。
PF_INFO_ID_ROBOT_NO	返回用于部件的机器人编号。

返回值

返回指定的属性值。

描述

返回部件的属性。在回调函数中使用该值自定义各运作。
不能通过虚拟控制器与命令窗口执行。

使用示例

请参阅以下内容。
PF_Control

3.3.12 PF_InitLog

启用Part Feeding日志文件输出并指定输出目标路径。
开始PF_Start之前执行。
要输出日志文件时，需要将装有RC+的PC连接至控制器。
有关Part Feeding日志文件，请参阅以下内容。
Part Feeding日志文件

格式

PF_InitLog部件, ID 输出目标路径, 添加

参数

- 部件ID
  指定部件ID（整数值1～32）。
- 输出目标路径
  指定日志文件输出目标路径（PC上的路径 + 文件名）。
- 添加
  如果设定True，则会在有特定的输出目标路径时添加数据。
  如果设定False，则覆盖文件。

返回值
无

描述
请通过与执行PF_Start时相同的任务执行。如果通过不同的任务执行，则不输出日志。
未开始Part Feeding进程时，不进行任何运作。
控制器未连接PC时，即使执行本函数，也不输出日志。另外，也不发生错误。
路径不存在或写入文件失败时，如果执行PF_Start，则会发生错误。执行本函数时，不发生错误。
不能通过虚拟控制器与命令窗口执行。
通过PF_Start命令由回调函数执行时，会记录送料器控制命令（PF_Center、PF_CenterByShift、PF_Flip、PF_Shift）的运作日志。在除此之外的用户函数内执行时，不记录日志。
本命令用于在内部处理中使用计时器。有关详细信息，请参阅以下内容。
Part Feeding进程使用的功能

使用示例
请参阅以下内容。
PF_Start / PF_Start函数

3.3.13 PF_IsStopRequested函数

调查有无Part Feeding进程的结束请求（是否执行PF_Stop）。
通常在回调函数内部使用。

格式
PF_IsStopRequested（部件ID）

参数
- 部件ID
  指定部件ID（整数值1～32）。

返回值
Part Feeding进程执行期间，如果PF_Stop被调用，则返回True。
除此之外时返回False。

描述
用于调查在回调函数内部有无Part Feeding进程的结束请求。编写程序，确保在进行循环处理等情况下，按循环调用该函数，有结束请求时，退出循环并结束回调函数。
多部件运作时，通过指定由PF_Start指定的某个部件ID，可调查结束请求。比如，按部件ID 1、2、3、4开始多部件运作并执行PF_Stop 2时，即使在本函数中指定部件ID 1、2、3、4之一，也可以调查有无结束请求。
不能通过虚拟控制器与命令窗口执行。

使用示例
请参阅以下内容。
PF_Robot

3.3.14 PF_Name$函数

通过部件ID返回部件名称。

格式
PF_Name$（部件ID）

参数

- 部件ID
  指定部件ID（整数值1～32）。

返回值
以字符串返回指定部件ID的名称。

描述
指定的部件ID无效时返回“ ”（空格）。
不能通过虚拟控制器与命令窗口执行。

使用示例

```
Print PF_Name$(1)
```

3.3.15 PF_Number函数

通过部件名称返回部件ID。

格式
PF_Number（部件名称）

参数

- 部件名称
  指定部件名称（字符串）。

返回值
返回指定部件名称的部件ID（整数值1～32）。

描述
没有部件名称时，返回-1。
有多个同名部件时，返回ID最小的那一个。
不能通过虚拟控制器与命令窗口执行。

使用示例

```
Print "Part1 part number = ", PF_Number("Part1")
```

3.3.16 PF_Output

控制送料器的输出端子。
如果将2台料斗连接至送料器，通过各自的料斗供给部件，则可指定各自料斗的ON时间。

格式
PF_Output 送料器编号/送料器名称, Out端子1 ON时间, Out端子2 ON时间

参数

- 送料器编号
  以表达式或数值指定送料器编号（整数值1～4）。
- 送料器名称
  以字符串指定送料器名称。
- Out端子1 ON时间
  指定ON时间（整数值）。可指定1 - 30000[ms]。
  为0时，保持OFF状态。
- Out端子2 ON时间
  指定ON时间（整数值）。可指定1 - 30000[ms]。
  为0时，保持OFF状态。

返回值

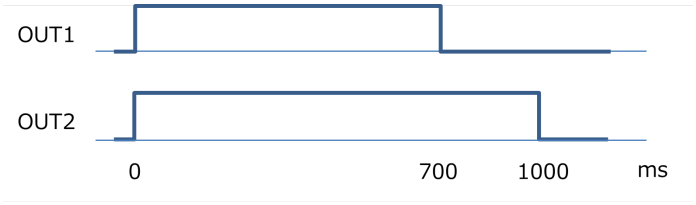
无

描述

执行本命令后立即返还控制。要等待料斗运作结束时，请在执行本命令之后立即使用Wait命令。
如果在送料器振动期间或料斗运作期间执行本命令，则会停止这些运作，并开始由本命令指定的运作。
要同时执行送料器振动与料斗运作时，请使用PF_OutputOnOff命令。
执行PF_Abort命令时料斗停止。执行PF_Stop命令时不停止。
不能通过虚拟控制器与命令窗口执行。
未在[系统配置] - [控制器] - [零件送料器]中勾选“安装料斗”时，会发生错误2584（指定了料斗安装设定无效的送料器）。
不能在IF-80中使用（会发生错误2589“调用送料器不可能执行的运作命令”）。请使用PF_OutputOnOff命令。
在IF-240中使用清除门时，Out端子2会用于清除门控制，因此，即使将Out端子2 ON时间设为0以外的值，Out端子2的输出也保持不变。但由于不能省略，因此请指定适当的值（0等）。

使用示例

PF_Output 1, 700, 1000
如下所述为从输出端子的输出。



PF_Output 1, 0, 700
如下所述为从输出端子的输出。

3.3.17 PF_OutputOnOff

控制送料器的输出端子。
要通过料斗供给部件时，将该命令置为On并予以执行。要停止从料斗的部件供给时，将该命令置为Off并予以执行。
IF-80内置有料斗。其他送料器备有料斗选件。

格式

(1) PF_OutputONOFF 送料器编号/送料器名称, On, 输出编号 [, 运作时间] [, 等待设定]
(2) PF_OutputONOFF 送料器编号/送料器名称, Off

参数

- 送料器编号
  以表达式或数值指定送料器编号（整数值1～4）。
- 送料器名称
  以字符串指定送料器名称。
- On/Off
  指定On(1) / Off (0)。
- 输出编号
  指定输出目标。为On时可指定。
  1：Out端子1
  2：Out端子2
- 运作时间
  指定On时间（整数值）。可指定1 - 30000[ms]。
  0或省略时，为连续On指定。
- 等待设定
  设定是否加入等待。仅限于IF-80可设定。
  0或省略时：不等待料斗运作完成。
  1：等待料斗运作完成。

返回值

无

描述

执行本命令后，会立即返还控制（包括在IF-80中将等待设定设为0或省略时的情况）。要等待料斗运作结束时，如使用示例所述，请使用Wait命令。
不能将输出端子1与2同时设为On。比如，已将端子1设为On时，端子2会置为Off。
已指定Off时，端子1与2会同时置为Off。
执行PF_Abort命令时料斗停止。执行PF_Stop命令时不停止。
未在[系统配置] - [控制器] - [零件送料器]中勾选“安装料斗”时，会发生错误2584（清除门无效）。
不能通过虚拟控制器与命令窗口执行。
在IF-240中使用清除门时，Out端子2会用于清除门控制。因此，即使将输出编号设为2，也可以执行，但Out端子2的输出不会发生变化。

使用示例

请参阅以下内容。
PF_Control

3.3.18 PF_PurgeGate

控制清除门（选件）的开闭运作。

格式

PF_PurgeGate 送料器编号/送料器名称, On/Off

参数

- 送料器编号
  以表达式或数值指定送料器编号（整数值1～4）。

- 送料器名称
  以字符串指定送料器名称。
- On/Off
  指定On(1) / Off (0)。如果指定On，清除门则会打开。
  如果指定Off，清除门则会关闭。

返回值
无

描述
执行本命令后立即返还控制。
要等待料斗运作结束时，请参阅以下使用示例。
PF_PurgeGate
未在[系统配置] - [控制器] - [零件送料器]中勾选“安装清除门”时，会发生错误2593（送料器的清除输出无效）。
不能通过虚拟控制器与命令窗口执行。
即使紧急停止、安全门打开、程序停止或执行PF_Stop时，也不会立即停止清除门的开闭运作。

使用示例1
等待清除门打开，然后进行后续运作。

```
PF_PurgeGate 1, On
Wait 5.0

' 后续运作
```

使用示例2
等待清除门关闭，然后进行后续运作。
注：如果清除门进行关闭运作时有过载（部件被夹住等），则会自动切换为打开运作。以下程序为使用
PF_PurgeGateStatus检测上述现象的示例。

```
Integer looplim
looplim = 50
PF_PurgeGate 1, Off
Do While PF_PurgeGateStatus(1) = True And looplim > 0
     Wait 0.1
     looplim = looplim -1
Loop
If looplim <= 0 Then
   ' 错误处理 清除门未关闭
EndIf

' 后续运作
```

3.3.19 PF_PurgeGateStatus函数

确认清除门（选件）关闭传感器的状态。

格式
PF_PurgeGateStatus（送料器编号/送料器名称）

参数
- 送料器编号
  以表达式或数值指定送料器编号（整数值1～4）。
- 送料器名称
  以字符串指定送料器名称。

返回值
如果关闭传感器置为On（门关闭），则返回False。
如果关闭传感器置为Off（门未关闭），则返回True。

描述
清除门带有关闭传感器，但没有打开传感器。也就是说，即使处于稍稍打开清除门的状态，本函数也会返回True。
执行本函数后立即返还控制。
未在[系统配置] - [控制器] - [零件送料器]中勾选“安装清除门”时，会发生错误2593（送料器的清除输出无效）。
不能通过虚拟控制器与命令窗口执行。

使用示例
请参阅以下使用示例。
PF_PurgeGate

3.3.20 PF_Purge / PF_Purge函数

执行清除（排出送料器上的部件）运作。
如果执行清除运作，则不论成功/失败与否，由部件ID指定的部件坐标队列都会被清除。

格式
PF_Purge 部件ID, 运作类型[, 振动时间[, 余量阈值[, 重试次数]]]
PF_Purge（部件ID, 运作类型[, 振动时间[, 余量阈值[, 重试次数]]]）

参数
- 部件ID
  指定部件ID（整数值1～32）。
- 运作类型
  指定运作的类型。

值（由PartFeeding.inc定义）	内容
PF_PURGETYPE_NOVISION	不进行基于视觉系统的反馈。 不进行部件余量计数。
PF_PURGETYPE_VISION	通过视觉系统进行反馈。 使用视觉系统进行部件余量计数。

- 振动时间
  指定清除运作时间。（毫秒单位）
  可省略。
  如果省略，则使用以下设定的值。
  清除 - 测试与调整
  指定-1时，会进行与省略时相同的运作。
- 余量阈值
  指定重试清除运作的部件余量（如大于等于该数，则进行重试）。
  可省略。
  运作类型为PF_PURGETYPE_NOVISION时，即使设定，运作也没有变化。
  运作类型为PF_PURGETYPE_VISION时，如果省略余量阈值，则会进行重试，直至余量为0。
- 重试次数
  指定重试清除运作次数的上限值。（初次清除运作也包含在次数中。）
  可省略。
  运作类型为PF_PURGETYPE_NOVISION时，即使设定，运作也没有变化。
  运作类型为PF_PURGETYPE_VISION时，如果省略重试次数，“余量阈值”的部件则会残留在平台上，或者在所有部件从平台上被除去之前进行清除运作（省略“余量阈值”与“重试”时）。

返回值

如果作为函数使用，则会在正常排出部件以及在运作类型中指定PF_PURGETYPE_NOVISION（没有部件余量计数）时，返回True。在运作类型中指定了PF_PURGETYPE_VISION（有部件余量计数）时，即使达到重试次数，也会在部件余量不低于阈值时返回False。

描述

IF-80时可选择清除对应平台与清除容器（选件）。
IF-240、IF-380、IF-530、IF-A1520、IF-A2330时可选择清除门（选件）。在[设置] - [系统配置] - [控制器] - [零件送料器] - [送料器]中设定清除门的有无。可按部件设定清除门的使用/不使用。使用清除门时，会根据PF_Purge语句，自动进行清除门的开闭运作，并通过开闭传感器进行判断的处理。
使用客户自制的清除门时，PF_Purge语句仅在平台上移动部件。
运作类型为PF_PURGETYPE_VISION时，使用部件Blob视觉序列，检测平台上的大致部件数。
有关清除的有效/无效以及清除部件的方向，请参阅以下内容。
防止末端夹具干扰
IF-80需要实施清除校准。
请参阅以下内容。
清除 - 自动校准（仅限于IF-80）
如果在清除无效时执行本函数，则会发生错误，并且由PF_STATUS_PURGENOTENABLED调用PF_Startus回调函数。
不能通过虚拟控制器与命令窗口执行。
在EPSON RC+ 7.5.0中，即使运作类型= 1，也不能省略余量阈值与重试。在这种情况下，将余量阈值与重试指定为值“0”。

使用示例

例1：
如下所述为使用视觉反馈进行清除的示例。各清除的振动时间为1500毫秒。尝试进行运作，直至部件小于等于3个。尝试进行5次重试。

```
Boolean purgeStatus
purgeStatus = PF_Purge(1, PF_PURGETYPE_VISION, 1500, 3, 5)
Print purgeStatus
```

例2：
如下所述为在没有视觉反馈的状态下进行清除的示例。进行2000 msec的清除运作。

```
PF_Purge 1, PF_PURGETYPE_NOVISION, 2000
```

3.3.21 PF_QtyAdjHopperTime函数

返回投放适量部件量所需的料斗运作时间。

格式

PF_QtyAdjHopperTime（部件ID, 供给量, 供给时间）

参数

- 部件ID
  指定部件ID（整数值1～32）。
- 供给量
  指定按供给时间提供的部件的量（数量）。
- 供给时间
  指定供给按供给量指定的部件的量（数量）所需的料斗运作时间[ms]。

返回值
返回料斗运作时间[ms]。返回值范围为1～30000[ms]。
发生错误时返回1。

描述
PF_QtyAdjHopperTime时，使用视觉系统计算供给最佳部件数（通过送料器校准确定）所需的料斗运作时间。返回值可用作PF_OutputOnOff语句的持续时间参数。
由开发人员调整并指定SupplyQty与SupplyTime。调整作业时可使用Test Hopper按钮。这些值表示在指定时间内通过料斗供给部件的数量。有关详细信息，请参阅以下内容。
校准

多部件运作时，计算的运作时间以同量（同数）投放各部件为前提。PF_Start运作期间执行PF_QtyAdjHopperTime时，执行在执行PF_Start时指定的部件Blob视觉序列。如果在PF_Start未运作时执行PF_QtyAdjHopperTime，则会仅使用由自变量指定的部件ID的部件Blob视觉序列。

不能通过虚拟控制器或命令窗口执行该命令。
移动相机不能使用该命令。
使用PF_Vision回调函数时，PF_QtyAdjHopperTime的作用在于仅使用部件Blob视觉序列，以便确定由部件ID指定的部件料斗运作时间。
选择“拾取&放置期间供给部件”时，如果在机器人机械臂进入相机视野的状态下执行本命令，则可能会将机械臂识别为部件，返回错误的料斗运作时间，敬请主持。
有关详细信息，请参阅以下内容。
供给部件

使用示例
在本例中使用PF_Control回调函数，在适量供给部件所需的推算时间内进行料斗选件的运作。料斗被调整为每秒钟(1000ms) 供给10个部件。

```
Function PF_Control(PartID As Integer, Control As Integer) As Integer
   Integer hopperOnTime

   Select Control
       'Request for part supply (add up to optimum number)
       Case PF_CONTROL_SUPPLY
           hopperOnTime = PF_QtyAdjHopperTime(PartID, 10, 1000)
           PF_OutputOnOff 1, On, 1, hopperOnTime
           Wait hopperOnTime / 1000
   Send
   PF_Control = PF_CALLBACK_SUCCESS
Fend
```

3.3.22 PF_QueAdd

在部件坐标队列中添加数据（点数据、部件姿势、用户数据）。

格式
PF_QueAdd 部件ID, 点数据 [, 部件姿势 [, 用户数据]]

参数
- 部件ID
  指定部件ID（整数值1～32）。
- 点数据
  指定点数据。
- 部件姿势
  以整数值指定与点数据一起注册的部件姿势。

可省略。
PF_PARTORIENT_FRONT=1: 表面部件姿势
PF_PARTORIENT_BACK=2: 背面部件姿势

- 用户数据
  以实数值指定与点数据一起注册的用户数据。
  可省略。

返回值
无

描述
用于将任意数据（点数据、部件姿势、用户数据）注册至部件坐标队列。
在Part Feeding进程中自动生成部件坐标队列。通常无需使用PF_QueAdd。用于客户要进行独自的视觉处理（使用PF_Vision回调函数）之时。
数据会被添加至指定部件ID的部件坐标队列的末尾。但是，如果已通过PF_QueSort设置了Sort方法，则按按照设置的Sort方法注册。
部件坐标队列的数据保持数的上限值为“1000”。

使用示例
请参阅以下内容。
PF_Vision

3.3.23 PF_QueAutoRemove

设定自动删除指定部件坐标队列的功能。

格式
PF_QueAutoRemove 部件ID, {True | False}

参数

- 部件ID
  指定部件ID（整数值1～32）。
- True | False
  False：禁用自动删除功能（默认）。
  True：启用自动删除功能。

返回值
无

描述
设定自动删除部件坐标队列的功能如果启用自动删除功能，则会在通过PF_QueGet从部件坐标队列获取点数据时，自动从部件坐标队列删除点数据。
如果禁用自动删除功能，则不删除点数据。要删除时，使用PF_QueRemove。
可按部件ID设定自动删除功能的启用/禁用。

使用示例

```
PF_QueAutoRemove 1, True
```

3.3.24 PF_QueAutoRemove函数

返回在部件坐标队列中设定的自动删除功能的状态。

格式
PF_QueAutoRemove（部件ID）

参数
- 部件ID
  指定部件ID（整数值1～32）。

返回值
启用部件坐标队列的自动删除功能时返回“True”；禁用时返回“False”。

描述
请参阅以下内容。
PF_QueAutoRemove

使用示例

```
Boolean autoremove
autoremove = PF_QueAutoRemove (1)
```

3.3.25 PF_QueGet函数

从部件坐标队列返回点数据。

格式
PF_QueGet（部件ID [, 索引] ）

参数
- 部件ID
  指定部件ID（整数值1～32）。
- 索引
  以整数值指定要获取的队列数据索引。
  开头索引编号为0。可省略。

返回值
返回点数据。

描述
PF_QueGet用于从部件坐标队列获取点数据。如果省略索引，将返回队列数据的开头数据。如果已设置索引，将返回设置的索引的点数据。
如果通过PF_QueAutoRemove，启用队列数据的自动删除功能，将通过PF_QueGet删除点数据。
如果禁用，将不删除点数据。要删除时，使用PF_QueRemove。

使用示例
请参阅以下内容。
PF_Robot

3.3.26 PF_QueLen函数

返回注册至部件坐标队列的部件坐标队列数据的数值。

格式
PF_QueLen（部件ID）

参数

- 部件ID
  指定部件ID（整数值1～32）。

返回值

以整数值返回有效部件坐标队列数据的注册数。

描述

返回部件坐标队列数据数的当前注册数。
还可以作为Wait命令的自变量使用。

使用示例

请参阅以下内容。
PF_Robot

3.3.27 PF_QueList

显示指定部件坐标队列的部件坐标队列数据一览（点数据）。

格式

PF_QueList 部件ID, [显示数]

参数

- 部件ID
  指定部件ID（整数值1～32）。
- 显示数
  以整数值指定显示数的数据量。
  可省略。如果省略，将显示所有队列数据。

描述

仅可在命令窗口中使用。

使用示例

利用命令窗口的操作示例

```
> PF_QueList 1
Queue 0   = XY(   1.000,   1.000,   0.000,   0.000 )   /R /0   ( 1 ) ( 0.000)
Queue 1   = XY(   3.000,   1.000,   0.000,   0.000 )   /R /0   ( 1 ) ( 2.000)
Queue 2   = XY(   4.000,   1.000,   0.000,   0.000 )   /R /0   ( 1 ) ( 3.000)
Queue 3   = XY(   5.000,   1.000,   0.000,   0.000 )   /R /0   ( 2 ) ( 4.000)
Queue 4   = XY(   6.000,   1.000,   0.000,   0.000 )   /R /0   ( 2 ) ( 5.000)
```

3.3.28 PF_QuePartOrient

显示、重新设定指定部件ID与索引的部件姿势（整数）。

格式

PF_QuePartOrient 部件ID [,索引 [, 部件姿势]]

参数

- 部件ID
  指定部件ID（整数值1～32）。

- 索引
  以整数值指定部件坐标队列数据的索引。
  （开头索引编号为“0”。）通过命令窗口执行时可省略。
- 部件姿势
  以整数值指定要重新设定的部件姿势。
  通过命令窗口执行时可省略。如果省略，将显示当前用户数据。
  PF_PARTORIENT_FRONT=1: 表面部件姿势
  PF_PARTORIENT_BACK=2: 背面部件姿势

描述
重新设定和显示部件坐标队列中注册的部件姿势。

通过PF_QueSort设定了以下Sort方法或设定了Sort方法时，部件坐标队列的排列顺序则会因所设定的Sort方法而异。
QUE_SORT_POS_PARTORIENT：部件姿势升序
QUE_SORT_INV_PARTORIENT：部件姿势降序

请参阅
PF_QuePartOrient函数

使用示例

```
PF_QuePartOrient 1, 1, PF_PARTORIENT_FRONT
```

3.3.29 PF_QuePartOrient函数

返回指定部件ID与索引的部件姿势。

格式
PF_QuePartOrientt（部件ID [, 索引] ）

参数

- 部件ID
  指定部件ID（整数值1～32）。
- 索引
  以整数值指定要获取的队列数据索引。
  开头索引编号为“0”可省略。

返回值
返回部件姿势 (integer)。

描述
PF_QuePartOrient用于从部件坐标队列获取部件姿势。如果省略索引，将返回部件坐标队列的开头数据。如果已设置索引，将返回设置的索引的部件姿势。

使用示例
请参阅以下内容。
PF_Robot

3.3.30 PF_QueRemove

从指定的部件坐标队列中删除部件坐标队列数据（点数据）。

格式
PF_QueRemove 部件ID [, 索引 | All]

参数

- 部件ID
  指定部件ID（整数值1～32）。
- 索引
  以整数值指定要删除的部件坐标队列数据的索引。
  开头索引编号为“0”
  可省略。要删除所有数据时，指定“All”。

返回值
无

描述
从部件坐标队列中删除数据（点数据）。要从部件坐标队列中删除已使用的数据时使用。

使用示例
请参阅以下内容。
PF_Robot

3.3.31 PF_QueSort

设定和显示指定部件坐标队列的Sort方法。

格式
PF_QueSort 部件ID [,Sort方法]

参数

- 部件ID
  指定部件ID（整数值1～32）。
- Sort方法
  以整数值（0～8）或下述常数指定Sort方法。
  通过命令窗口执行时可省略。
  如果省略，将显示当前Sort方法。

常数	值	内容
QUE_SORT_NONE	0	无排序（注册至部件坐标队列）
QUE_SORT_POS_X	1	X坐标升序
QUE_SORT_INV_X	2	X坐标降序
QUE_SORT_POS_Y	3	Y坐标升序
QUE_SORT_INV_Y	4	Y坐标降序
QUE_SORT_POS_USER	5	用户数据（实数）升序
QUE_SORT_INV_USER	6	用户数据（实数）降序
QUE_SORT_POS_PARTORIENT	7	部件姿势升序
QUE_SORT_INV_PARTORIENT	8	部件姿势降序

返回值
无

描述
设定部件坐标队列的Sort方法。通过PF_QueAdd添加点数据时，会根据已设定的Sort注册到部件坐标队列中。
使用PF_QueAdd之前，需要执行PF_QueSort。即使在添加数据后使用本函数，也不会进行数据排序。

使用示例

```
PF_QueSort 1, QUE_SORT_POS_X
```

3.3.32 PF_QueSort函数

返回在指定部件坐标队列中设定的Sort方法。

格式
PF_QueSort（部件ID）

参数

- 部件ID
  指定部件ID（整数值1～32）。

返回值
以整数值返回在部件坐标队列中设定的Sort方法。

值	内容
0	无排序（注册至部件坐标队列）
1	X坐标升序
2	X坐标降序
3	Y坐标升序
4	Y坐标降序
5	用户数据（实数）升序
6	用户数据（实数）降序
7	部件姿势升序
8	部件姿势降序

使用示例

```
Integer quesort

quesort = PF_QueSort(1)
```

3.3.33 PF_QueUserData

显示、重新设定指定部件ID与索引的用户数据（实数）。

格式
PF_QueUserData 部件ID [,索引 [, 用户数据]]

参数

- 部件ID
  指定部件ID（整数值1～32）。
- 索引
  以整数值指定部件坐标队列数据的索引。（开头索引编号为“0”。）
  通过命令窗口执行时可省略。
- 用户数据
  以实数值指定要重新设置的用户数据。
  通过命令窗口执行时可省略。
  如果省略，将显示当前用户数据。

描述

重新设定和显示部件坐标队列中注册的用户数据。

通过PF_QueSort设定了以下Sort方法或设定了Sort方法时，部件坐标队列的排列顺序则会因所设定的Sort方法而异。
QUE_SORT_POS_USER：用户数据（实数）升序
QUE_SORT_INV_USER：用户数据（实数）降序

请参阅

PF_QueUserData函数

使用示例

```
Real r
r = 0.1
PF_QueUserData 1, 1, r
```

3.3.34 PF_QueUserData函数

返回指定部件ID（与索引）的用户数据。

格式

PF_QueUserData（部件ID [, 索引] ）

参数

- 部件ID
  指定部件ID（整数值1～32）。
- 索引
  以整数值指定要获取的队列数据索引。
  开头索引编号为“0”可省略。

返回值

返回用户数据 (Real)。

描述

PF_QueUserData函数用于从PF队列获取用户数据（实数）。如果省略索引，将返回部件坐标队列的开头数据。如果已设置索引，将返回设定的索引的用户数据。

使用示例

```
Real r
r = PF_QueUserData(1, 1)
```

3.3.35 PF_ReleaseFeeder

解除通过PF_AccessFeeder获取的锁定（所有权）。
PF_AccessFeeder与PF_ReleaseFeeder用于在多台机器人/1台送料器系统中锁定/解锁对送料器的访问，以防止机器人之间的碰撞。2台机器人同时共享同一送料器时，需要使用这些命令。

示例：
在任务1中执行PF_AccessFeeder 1。继续执行任务1。
如果在任务2中执行PF_AccessFeeder 1，则会暂停任务2。
如果在任务1中执行PF_ReleaseFeeder 1，则会重新开始任务2。

本命令可记述在运作命令的并行处理 (!...!) 中。

格式

格式1：PF_ReleaseFeeder 送料器编号/送料器名称
格式2：运作命令 ! [Dn;] PF_ReleaseFeeder 送料器编号/送料器名称 !

参数

- 送料器编号
  以表达式或数值（整数：1～4）指定送料器编号。
- 送料器名称
  指定送料器标签（字符串）。
- 运作命令
  Arc、Arc3、Go、Jump、Jump3、Jump3CP、Move、BGo、BMove、TGo、TMove之一
- Dn
  以%指定按机器人移动路径上哪个位置开始并行处理
  （请参阅“Epson RC+ 8.0 SPEL+语言参考 !...! 并行处理”）

返回值

无

描述

仅可在同一任务内解锁。
不能通过虚拟控制器与命令窗口执行。

使用示例

下例所述为获取送料器上最后1个部件并到达移动到放置位置的路径的50%位置时，将控制返还给PF_Start，开始视觉系统拍摄于送料器运作的情况。

```
Function main
    Motor On
    Do while True
        PF_AccessFeeder 1
        PF_Start(1)
    Loop
Fend

Function PF_Robot(partID As Integer)
    Xqt Task_PF_Robot(partID)
    PF_Robot = PF_SUCCESS
Fend

Function Task_PF_Robot(partID As Integer)
    PF_AccessFeeder 1

    Integer i
    For i = 1 to numToPick
```

```
        pick = PF_GetQue(PartID)
        PF_QueRemove(PartID)
        Jump pick
        On gripperOutput
        Wait 0.1
        If i < numToPick And PF_QueLen(PartID) > 0 Then
            Jump place
        Else
            ' Last part, so release the feeder at 50%
            Jump place !D50; PF_ReleaseFeeder 1!
        EndIf
        Off gripperOutput
        Wait 0.1
    Next i
Fend
```

3.3.36 PF_Shift

执行移动运作。

格式

PF_Shift 部件ID, 方向 [, 振动时间]

参数

- 部件ID
  指定部件ID（整数值1～32）。
- 方向
  指定移动方向。

方向	值（由PartFeeding.inc定义）	运作
前	PF_SHIFT_FORWARD	
左前	PF_SHIFT_FORWARD_LEFT	
右前	PF_SHIFT_FORWARD_RIGHT	
左	PF_SHIFT_LEFT	
右	PF_SHIFT_RIGHT	
后	PF_SHIFT_BACKWARD	
左后	PF_SHIFT_BACKWARD_LEFT	
右后	PF_SHIFT_BACKWARD_RIGHT	

- 振动时间
  指定清除运作时间。（毫秒单位）
  可省略。如果省略，则使用通过移动校准设定的值。
  请参阅以下内容。
  - 移动 － 测试与调整（简易）

- 移动 - 测试与调整（详细）
  指定-1时，会进行与省略时相同的运作。

返回值
无

描述
执行IF系列送料器的移动运作。

移动运作用于下述情况。

- 将部件移动至放置位置侧，提高机器人的取放效率
- 使用自定义平台时，将部件直立落入槽或孔中或对准方向

不能在下述条件下执行。

- 通过用户函数执行时，通过本命令指定的送料器（由部件ID指定的送料器）被Part Feeding进程（PF_Start命令）使用时（错误7733）
- 通过回调函数执行时，指定未通过PF_Start命令指定的部件ID时（错误7733）
- 通过虚拟控制器与命令窗口执行（错误3804）

本命令用于在内部处理中使用SyncLock。有关详细信息，请参阅以下内容。
Part Feeding进程使用的功能

使用示例

```
PF_Shift 1, PF_SHIFT_FORWARD, 500
```

3.3.37 PF_Start / PF_Start函数

开始指定部件的Part Feeding进程。

格式
PF_Start 部件ID1 [, 部件ID2 [, 部件ID3 [, 部件ID4]]]
PF_Start（部件ID1 [, 部件ID2 [, 部件ID3 [, 部件ID4]]]）

参数

- 部件ID
  指定部件ID（整数值1～32）。将变量用作自变量时，如果未指定部件ID，则将值设为0。

返回值
无

描述
开始PF_Start之前，请执行下述处理。
有关执行方法，请参阅使用示例。

- 选择要使用的机器人
- 设定机器人（Power、Speed、Accel等）
- 将电机设为On
- 要输出日志时，执行PF_InitLog

如果执行PF_Start，则会生成新任务（任务编号32）并开始Part Feeding进程。
不等待Part Feeding进程结束，将控制返还给调用来源。

在下述条件下会发生错误，并执行Status回调函数。未开始Part Feeding进程。

条件	Status回调函数自变量Status的值
没有部件ID 要利用不同送料器中的部件开始多部件运作 重复设定部件ID	PF_STATUS_BAD_ID
部件的参数设定无效	PF_STATUS_BAD_PARAMETER
送料器校准未完成	PF_STATUS_CAL_NOT_COMPLETE
部件无效	PF_STATUS_PARTNOTENABLED
送料器已被使用	PF_STATUS_FEEDERINUSE_ERROR
发生系统错误	PF_STATUS_ERROR

- 多送料器运作
  可利用属于不同送料器的部件同时执行Part Feeding进程。比如，部件1属于送料器1，部件2属于送料器2时，可在执行PF_Start 1后执行PF_Start 2。
  PF_Start用于按送料器创建任务。任务编号始于“32”，每执行一次PF_Start，编号都会减小1。使用的任务编号取决于送料器编号。

送料器编号	任务编号
1	32
2	31
3	30
4	29

各回调函数（PF_Robot、PF_Control、PF_Status. PF_Vision、PF_MobileCam）用于创建任务并在同一任务中执行。
T/VT系列控制器时，可同时控制的送料器最多为2台。如果要进行大于等于3台的运作，则会发生“错误7731：超过控制器类型的最大同时送料器数”。

- 多部件运作
  要进行多部件运作时，在自变量中指定多个ID。最多可指定4个部件ID。此时，送料器按由PF_Start的第1自变量指定的部件ID（有效部件）的校准参数进行运作。要切换有效部件时，使用PF_ActivePart命令。
  可在多部件运作中指定的部件仅为属于同一送料器的部件。如果指定不同送料器的部件并执行PF_Start，则会发生错误，并且由PF_STATUS_BAD_ID调用PF_Status回调函数。
  多部件运作时，1台送料器最多可使用2台机器人。如果按使用大于等于3台的机器人设定执行PF_Start，则会发生错误，并且由PF_STATUS_**调用PF_Status回调函数。

- 其他注意事项
  执行Part Feeding进程期间，不能针对同一送料器执行新的Part Feeding进程。比如，部件1属于送料器1，部件2属于送料器1时，如果执行PF_Start 1后执行PF_Start 2，则会发生错误，并且由PF_STATUS_FEEDERINUSE_ERROR调用PF_Status回调函数。已执行的PF_Start会继续进行处理。
  通常请通过任务执行PF_Start。通过后台任务执行时，会发生错误。
  不能通过虚拟控制器与命令窗口执行。

使用示例

```
Robot 1
Motor On
Power High
Speed 100
Accel 100, 100
```

```
LimZ -80.0

PF_InitLog("C:\log.csv", True)
PF_Start(1)
```

3.3.38 PF_Stop

发出Part Feeding进程的结束请求。
有正在执行的回调函数时，等待其结束。
然后执行PF_CycleStop回调函数，进程停止。

格式
PF_Stop 部件ID

参数

- 部件ID
  指定部件ID（整数值1～32）。

返回值
无

描述
停止Part Feeding进程。
与PF_Abort命令不同，会等待当前执行的回调函数的结束。
回调函数结束后，执行PF_CycleStop回调函数。
未开始Part Feeding进程时，不进行任何运作。
要在多部件运作中使用PF_Stop时，通过设定由PF_Start设定的某个ID，即会停止Part Feeding进程。
不能在执行PF_Stop后立即执行PF_Start。如果在PF_CycleStop回调函数完成前执行PF_Start，则会发生PF_STATUS_FEEDERINUSE_ERROR。这是因为在当前的Part Feeding进程完成前，要执行新的Part Feeding进程的缘故。
要修正这种状况时，添加下述代码。

```
PF_Stop 1     ' 在本例中，由使用任务32的送料器1执行部件1。
TaskWait 32   ' 等待处理结束。
PF_Start 2    ' 开始新部件。
```

不能通过虚拟控制器与命令窗口执行。

使用示例

```
PF_Stop 1
```

3.4 Part Feeding回调函数

回调函数是指，在指定条件成立时由PF_Start命令的运作进程自动调用的SPEL函数。
新注册第1个部件时，会将记述有各回调函数模板的程序文件（PartFeeding.prg、PartFeeding.inc）添加至项目中。
客户根据自己的装置规格，用SPEL语言记述所需的运作。

函数	描述/用途
PF_Robot	记述机器人操作（拾取、放置）。
PF_Control	记述料斗、自定义照明的操作。
PF_Status	记述错误处理。

函数	描述/用途
PF_MobileCam	记述朝移动相机拍摄位置的移动和避让。
PF_Vision	记述独自的视觉处理（例：根据多个视觉序列的结果判定部件）。
PF_Feeder	记述独自的送料器处理（例：在客户制作的平台上处理部件）。
PF_CycleStop	记述执行PF_Stop后的处理。

3.4.1 通用事项

回调函数内的机器人编号为已指定的机器人编号。
有关指定方法，请参阅以下内容。
常规
执行Part Feeding进程期间，回调函数会被作为常规任务予以执行。使用的任务编号始于“32”。每添加一次送料器，都会减少。
请勿删除PartFeeding.prg、PartFeeding.inc。另外，请勿删除未使用的回调函数或进行非注释化处理。否则会发生创建错误。
执行Part Feeding进程期间，请勿通过客户代码调用回调函数。否则在这种情况下，可能会导致意外运作。可单独执行回调函数，以便进行测试。

3.4.2 PF_Robot

记述机器人从送料器中获取部件，然后放到指定位置的运作。
请务必记述该函数的内容。

格式

```
Function PF_Robot（部件ID As Integer）As Integer
’ 机器人的运作
Fend
```

参数

- 部件ID
  输入部件ID（整数值1～32）。
  在多部件运作时有效部件进入。

返回值

要在PF_Robot函数结束后控制Part Feeding进程时，请指定下述值。（由PartFeeding.inc定义各常数。）

- PF_CALLBACK_SUCCESS
  通常指定该值。部件坐标队列中未残留数据时（多部件运作或没有有效部件的数据时），会继续进行Part Feeding处理。
  队列中残留数据时（多部件运作或有有效部件的数据时），会重启PF_Robot回调函数，而非更改部件坐标队列的内容。

- PF_CALLBACK_RESTART
  不论坐标队列中是否残留有数据，都继续执行Part Feeding进程，重新生成坐标队列。这适用于每次拾取都进行拍摄，以便高精度地拾取部件的情况。

- PF_CALLBACK_RESTART_ACTIVEPART
  不论坐标队列中是否残留有数据，都继续执行Part Feeding进程，并且仅生成有效部件的坐标队列。不更改有效部件以外的坐标队列。这适用于希望省略有效部件以外的视觉处理并进行高速化的情况。

- 用户定义错误编号（8000 - 8999）
  要启动PF_Status回调函数进行错误处理时设定。已设定的值会被作为PF_Status回调函数的自变量交接。

描述

本函数用于记述以下处理。

1. 从坐标队列中获取部件坐标（使用PF_QueGet函数）
2. 将机器人移至送料器上的部件坐标位置
3. 操作末端夹具等抓取部件
4. 将机器人移至部件配置坐标
5. 操作末端夹具等释放部件
6. 删除1个坐标队列（使用PF_QueRemove命令）

一般来说，会通过循环处理执行1～6，对坐标队列内的所有数据进行处理。坐标队列中包括通过此前视觉运作检测的部件坐标列表。坐标为本地坐标。多部件运作时，按部件ID生成坐标队列。
启动本函数时选择的机器人是由部件的Robot#指定的机器人。要多机器人使用时，根据需要使用Robot语句切换机器人。
有关详细信息，请参阅以下手册。
“Epson RC+ 8.0 SPEL+语言参考 - Robot”

⚠ 注意

- **建议通过参数获取部件ID。**
- **在多机器人环境下要更改选择的机器人编号时，请注意不要更改为错误的机器人编号。**

程序示例

下述程序是使用单纯的真空末端夹具处理所有部件的示例。
拾取部件时，通过真空传感器监视吸附状态。
不能吸附时，进行最多3次重试。即使这样仍不吸附时，会作为返回值返回用户错误代码8001。

```
' ** IO Label (Output) **
' O_Adsorb 吸附
' O_Desorb 脱离
'
' ** IO Label (Input) **
' I_Adsorbed 吸附传感器
'
' ** Point **
' PlacePos 部件的放置位置
'
' ** User Error **
' 8001 发生吸附超时
'
Function PF_Robot(partID As Integer) As Integer

    Byte retry

    ' 处理所有坐标队列或没有停止请求期间进行循环
    Do While PF_QueLen(partID) > 0 And PF_IsStopRequested(partID) = False

        '将机器人移至拾取位置
        Jump PF_QueGet(partID)

        ' 真空ON & 吸附确认（3次重试）
        On O_Adsorb
        retry = 3
        Do While retry > 0
```

```
        Wait Sw(I_Adsorbed) = On, 0.5
        If TW = True Then
            ' 如果超时，则将吸附设为OFF，然后再次设为ON
            Off O_Adsorb
            Wait 0.1
            On O_Adsorb
        EndIf
    Loop
    If TW = True Then
        ' 即使重试也不能吸附，因错误而结束
        ' 由Status回调函数进行错误处理
        PF_Robot = 8001
        Exit Function
    EndIf

    '将机器人移至放置位置
    Jump PlacePos

    ' 真空OFF & 真空破坏
    Off O_Adsorb
    On O_Desorb
    Wait 0.1
    Off O_Desorb

    ' 从坐标队列中删除1个数据
    PF_QueRemove partID

    Loop

    PF_Robot = PF_CALLBACK_SUCCESS

Fend
```

3.4.3 PF_Control

系统需要进行料斗运作或照明操作时，会调用PF_Control回调函数。该回调函数中记述有客户装置上安装的以下设备的操作。

- 料斗
- 自定义照明

要使用自定义照明时，请在Epson RC+ 8.0菜单 - [工具] - [上料] - [照明]中选择[自定义前灯]。

格式

```
Function PF_Control（部件ID As Integer, Control As Integer）As Integer
'（料斗操作）
'（外部照明操作）
Fend
```

参数

- 部件ID
  输入部件ID（整数值1～32）。
  在多部件运作时有效部件进入。
- Control
  输入操作类型（整数值）。

操作	值（由PartFeeding.inc定义）
料斗操作 （送料器上没有部件时）	PF_CONTROL_SUPPLY_FIRST
料斗操作 （送料器上有部件，可添加时）	PF_CONTROL_SUPPLY
自定义照明On	PF_CONTROL_LIGHT_ON
自定义照明Off	PF_CONTROL_LIGHT_OFF

返回值

正常时，请设定常数PF_CALLBACK_SUCCESS（由PartFeeding.inc定义）。
发生错误时，请设定用户定义错误编号 (8000 - 8999)。该值会被作为PF_Status回调函数的自变量交接。

描述

使用Select...Send语句记述各设备的处理。

- 料斗操作（送料器上没有部件时）
  是送料器上没有任何部件的状态。使料斗运作，向送料器供给部件。
  如果将此时供给的部件数设为通过最佳投放部件数校准获取的值（通过PF_Info命令获取），则会最大限度提高机器人的运作效率。

- 料斗操作（送料器上有部件，可添加时）
  是送料器上有部件，但处于可添加的状态。可按该时序添加部件，增加通过视觉系统检测的部件，提高机器人的运作效率。
  比如，可将此时的供给部件数作为机器人取走的部件数。

- 自定义照明On
  是点亮自定义照明以进行视觉系统拍摄的时序。操作并点亮自定义照明。

- 自定义照明Off
  是熄灭自定义照明以结束视觉系统拍摄的时序。操作并熄灭自定义照明。
  请在客户的代码中记述下述处理。请在执行PF_Start之前进行操作。
  -开始与料斗之间的通信、设定调整等
  -开始与外部照明之间的通信、设定调整（亮度）等

程序示例

下述程序用于控制料斗与外部照明。
将1台料斗 (Gen.2) 连接至IF-240，然后利用PF_Hopper命令进行控制。本例中的最佳部件数为60。在料斗校准画面中进行调整，以确保料斗每进行3秒钟运作供给60个部件。
外部照明的规格应可连接至标准IO。

```
' ** IO Label (Output) **
' O_FrontLight 外部照明

Function PF_Control(partID As Integer, Control As Integer) As Integer
    Integer hopperOnTime

    Select Control

        ' 料斗运作  送料器上没有部件时
        Case PF_CONTROL_SUPPLY_FIRST
            PF_Hopper 1,partID, 3000
            Wait 3

        ' 料斗运作  向送料器上添加部件时
```

```
        Case PF_CONTROL_SUPPLY
            hopperOnTime = PF_QtyAdjHopperTime(partID, 60, 3000)
            PF_Hopper 1, partID, hopperOnTime
            Wait hopperOnTime / 1000

        ' 自定义照明ON
        Case PF_CONTROL_LIGHT_ON
            On O_FrontLight

        ' 自定义照明OFF
        Case PF_CONTROL_LIGHT_OFF
            Off O_FrontLight

    Send

    PF_Control = PF_CALLBACK_SUCCESS
Fend
```

3.4.4 PF_Status

记述回调函数的返回值以及与系统状态（未供给部件等）相应的处理（主要是错误处理）。

格式

Function PF_Status（部件ID As Integer, Status As Integer）As Integer
’（错误处理）
Fend

参数

- 部件ID
  输入部件ID（整数值1～32）。
  在多部件运作时有效部件进入。
- Status
  输入状态。
  是回调函数的返回值或系统设定的值（请参阅：描述）。

返回值

如果未指定返回值，则会视为已指定PF_EXIT。请务必设定返回值，以结束Part Feeding进程。

- PF_CONTINUE
  继续Part Feeding进程。通常请指定该值。
- PF_EXIT
  结束Part Feeding进程。如果指定该值，Part Feeding进程会立即结束。要指定该值时，根据需要记述将装置恢复为初始状态的处理（机器人的原点返回、电机的Off等）。
- PF_RESTART
  重置Part Feeding进程并利用视觉系统重新开始。
- PF_RESTART_ACTIVEPART
  重置Part Feeding进程并利用视觉系统重新开始。仅重新生成有效部件的队列。

描述

在其他回调函数（PF_CycleStop函数与PF_Status函数除外）之后执行该函数。

对于自变量Status而言，被设为此前执行的回调函数的返回值或Part Feeding进程内部发生的错误。记述与这些值相应的处理。

- PF_CALLBACK_SUCCESS
  回调函数已正常结束。通常不进行任何处理。

- PF_CALLBACK_RESTART
  PF_Robot回调函数正常结束，返回值为PF_CALLBACK_RESTART。通常不进行任何处理。

- PF_CALLBACK_RESTART_ACTIVEPART
  PF_Robot回调函数正常结束，返回值为PF_CALLBACK_RESTART_ACTIVEPART。通常不进行任何处理。

- PF_STATUS_NOPART
  是未从料斗供给部件的状态。记述向料斗供给部件的操作。

- PF_STATUS_TOOMANYPART
  是从料斗供给过多部件且无法拾取部件的状态。记述通过操作员调用等除去料斗上的部件的操作。该状态频繁发生时，请重新评估料斗设定。

- PF_STATUS_BAD_ID
  执行PF_Start命令时指定的部件ID错误。请确认是否正确指定已注册的部件ID。多部件运作时，要通过多台送料器开始运作。请重新评估各部件的设定。Part Feeding进程会立即结束。

- PF_STATUS_BAD_PARAMETER
  执行PF_Start命令时指定的部件参数错误。Part Feeding进程会立即结束。

- PF_STATUS_CAL_NOT_COMPLETE
  未完成执行PF_Start命令时指定的部件校准。Part Feeding进程会立即结束。
  有关送料器校准的执行方法，请参阅以下内容。
  校准

- PF_STATUS_WRONGPART
  无法检测到送料器上的部件。请确认可正确检测部件视觉序列。另外，请确认是否进入其他类型的部件或损坏的部件。
  多次尝试通过视觉系统检测部件，虽然部件Blob视觉序列检测到拾取区域内有几个部件，但部件检测视觉序列无法识别表面部件或背面部件时，会产生该状态值。

- PF_STATUS_PARTBLOB_ERROR
  用于部件Blob检测的视觉序列或对象无效。Part Feeding进程会立即结束。请确认用于部件的部件Blob序列与对象。

- PF_STATUS_PARTSEQ_ERROR
  用于部件检测的视觉序列或对象无效。Part Feeding进程会立即结束。请确认用于部件的部件序列与对象。

- PF_STATUS_ERROR
  执行PF_Start命令期间发生错误（系统错误）。Part Feeding进程会立即结束。请确认通过“视觉”指定的视觉序列是否正常运作。请确认各回调函数的运作是否正确，或单独执行回调函数进行调试。另外，如果大于等于2台的机器人共享送料器，则会发生错误7730“1台供料器的机器人数超过最大值。”。
  可利用Status回调函数的返回值，指定运作进程的处理。
  如果未指定返回值，则会视为已指定PF_EXIT。请务必设定返回值，以结束Part Feeding进程。

- PF_STATUS_FEEDERINUSE_ERROR
  针对同一送料器多次启动Part Feeding进程。Part Feeding进程会立即结束。（继续执行已启动的Part Feeding进程。）请重新评估程序。

- PF_STATUS_PARTNOTENABLED
  部件无效。
  请在Epson RC+ 8.0菜单 - [工具] - [上料] - [部件] - [部件**] - [常规]中确认[启用]复选框已被勾选。

- PF_STATUS_PURGENOTENABLED
  部件无效，但已执行PF_Purge函数。
  请在Epson RC+ 8.0菜单 - [工具] - [上料] - [部件] - [部件**] - [常规]中确认[启用]复选框已被勾选。

注意

- 请在PF_Status函数内部记述程序，以免发生新的错误。可能是递归执行PF_Status函数，导致错误处理未结束。
- 不建议在PF_Status函数内记述送料器控制命令（PF_Center、PF_CenterByShift、PF_Flip、PF_Shift）。
- PF_Status具有的作用为，在此前执行的回调函数完成后，指示系统继续执行的方法（也就是在PF_Robot、PF_Vision、PF_Control、PF_MobileCam、PF_Feeder之后继续执行的方法）。通过将PF_Status的返回值设为PF_CONTINUE、PF_RESTART、PF_RESTART_ACTIVEPART、PF_EXIT之一的返回值，实现这一运作。有关详细信息，请参阅以下程序示例。

程序示例

下述程序记述有错误处理。
用户错误为Robot回调函数的程序示例中所述的吸附超时。
发生该错误时，将机器人的电机置为Off。

```
' ** User Error **
' 8001 发生吸附超时

Function PF_Status(PartID As Integer, Status As Integer) As Integer

    Select Status

        Case PF_CALLBACK_SUCCESS
            ' 成功（通常不进行任何操作）
        Case PF_CALLBACK_RESTART
            ' 利用视觉系统重新开始
            PF_Status = PF_RESTART
            Exit Function

        Case PF_CALLBACK_RESTART_ACTIVEPART
            ' 利用视觉系统重新开始 -
            ' 仅生成有效部件的坐标队列并获取图像
            PF_Status = PF_RESTART_ACTIVEPART
            Exit Function

        Case PF_STATUS_NOPART
            ' 料斗上没有部件
            MsgBox "Hopper empty."

        Case PF_STATUS_TOOMANYPART
            ' 送料器上的部件过多
            MsgBox "Too many parts on Feeder."

        Case PF_STATUS_BAD_ID
            ' 没有指定的部件ID
            MsgBox "Bad PartID."

        Case PF_STATUS_BAD_PARAMETER
            ' 部件参数错误
            MsgBox "Bad parameter."

        Case PF_STATUS_CAL_NOT_COMPLETE
            ' 校准未完成
            MsgBox "Calibration incomplete."

        Case PF_STATUS_WRONGPART
            ' 可能是送料器平台上的部件有误
```

```
            MsgBox "Wrong Part."

        Case PF_STATUS_ERROR
            ' 错误
            MsgBox "Error!! (code: " + Str$(Err) + " ) " + ErrMsg$(Err)

        Case PF_STATUS_PARTBLOB_ERROR
            ' 部件Blob视觉序列错误
            MsgBox "Part Blob vision error."

        Case PF_STATUS_PARTSEQ_ERROR
            ' 部件检测视觉序列错误
            MsgBox "Part Sequence vision error."

        Case PF_STATUS_FEEDERINUSE_ERROR
            ' 送料器正在使用
            MsgBox "Feeder is already in use."

         Case PF_STATUS_PARTNOTENABLED
            ' 部件无效
             MsgBox "Part is disabled."

        Case PF_STATUS_PURGENOTENABLED
            ' 清除无效
            MsgBox "Purge is disabled."

        Case 8001
            ' 例：吸附超时
            MsgBox " Vacuum Error!!"
            Motor Off
            PF_Status = PF_EXIT
            Exit Function

    Send

    PF_Status = PF_CONTINUE

Fend
```

3.4.5 PF_MobileCam

记述使用移动相机时将机器人移至拍摄位置的处理以及拍摄后的避让处理。不论是否使用移动相机，始终执行本函数。

格式

```
Function PF_MobileCam（部件ID As Integer, Action As Integer）As Integer
’（将机器人移至拍摄位置）
’（避让机器人）
Fend
```

参数

- 部件ID
  输入部件ID（整数值1～32）。
  在多部件运作时有效部件进入。

- Action
  输入移动的类型。

移动的类型	常数（由PartFeeding.inc定义）
将机器人移至拍摄位置 （此后会进行视觉系统拍摄与送料器运作。）	PF_MOBILECAM_BEFORE
避让机器人 （此后会执行PF_Robot回调函数。）	PF_MOBILECAM_AFTER

返回值
正常时，请设定常数PF_CALLBACK_SUCCESS（由PartFeeding.inc定义）。要结束本函数并进行错误处理时，请设定用户定义错误编号 (8000 - 8999)。该值会被作为PF_Status回调函数的自变量交接。

描述
在拍摄之前以及调用PF_Robot回调函数之前执行该函数。
记述本函数之后，请充分进行测试，确认机器人可否安全地移动。

程序示例 下述程序所述为将移动相机移至拍摄位置的处理以及进行避让处理的示例。位置被定义为点数据。标签的拍摄位置为VisionPos、避让位置为HomePos。

```
' ** Point **
' VisionPos  移动相机拍摄位置
' HomePos  移动相机避让位置
'
Function PF_MobileCam(partID As Integer, Action As Integer) As Integer

    Select Action

        Case PF_MOBILECAM_BEFORE ' 将机器人移至拍摄位置
            Jump VisionPos
        Case PF_MOBILECAM_AFTER ' 避让机器人
            Jump HomePos
    Send

    MobileCam = PF_CALLBACK_SUCCESS
Fend
```

3.4.6 PF_Vision

客户要记述拍摄处理（照明控制、视觉序列的执行、SPEL程序处理）时使用。仅通过执行视觉序列难以检测部件或识别姿势（表里等）时使用。要使用本功能时，请在Epson RC+ 8.0菜单 - [工具] - [Part Feeding] - [Vision]中，选择[通过PF_Vision callback进行视觉处理]。

格式
Function PF_Vision（部件ID As Integer, ByRef numBack As Integer）As Integer
’（视觉处理）
Fend

参数

- 部件ID
  输入部件ID（整数值1～32）。
  在多部件运作时有效部件进入。
- numBack
  代入不可拾取部件（背面方向等）的数量。（进行ByRef指定。）

Part Feeding进程使用该值判定是否需要翻转。
部件没有表里时，设定“0”。

返回值

要继续处理时，请设定PF_CALLBACK_SUCCESS。
要进行错误处理时，请设定用户定义错误编号 (8000 - 8999)。
该值会被作为PF_Status回调函数的自变量交接。

描述

该函数用于组合多个视觉序列与外部照明控制检测部件坐标。客户记述下述处理。

- 自定义照明控制
- 执行视觉处理（执行VGet语句）
- 管理部件坐标队列（初始化与注册点数据）

程序示例

下述程序所述为使用2个视觉序列VSeq1（包括1个Geom对象）与VSeq2（包括1个Geom对象）检测部件的示例。同时使用VSeq1与自定义照明1，以检测部件的位置，但不能判定姿势。同时使用VSeq2与自定义照明2，仅检测表面部件。
通过PF_QueAdd命令，已检测部件的坐标（设为本地编号1）被放入到坐标队列中。此时，将按如下所述获取的值代入Z坐标。
示教窗口
要启用翻转时，请参阅以下内容。
常规
在[部件视觉序列]中指定VSeq2，点亮自定义照明2。请参阅以下内容。
视觉
进行翻转校准。请参阅以下内容。
校准

```
' ** IO Label (Output) **
' O_FrontLight1 自定义照明1
' O_FrontLight2 自定义照明2
Function PF_Vision(partID As Integer, ByRef NumBack As Integer) As Integer

    Boolean found
    Integer i, numFound
    Real PX_X, PX_Y, PX_U
    Real RB_X, RB_Y, RB_U, RB_Z

    ' 拾取z坐标
    RB_Z = -80.0

    ' 对坐标队列进行初始化
    PF_QueRemove partID, All

    ' 点亮自定义照明1
    On O_FrontLight1

    ' 检测部件（执行UsrVisionSeq1）
    VRun VSeq1
    VGet VSeq1.Geom01.NumberFound, numFound

    ' 熄灭自定义照明1
    Off O_FrontLight1
    ' 点亮自定义照明2
    On O_FrontLight2

    numBack = 0
    For i = 1 To numFound
        '获取通过 VSeq1.Geom01检测的部件的XY像素坐标
```

```
        ' 设定VSeq2.Geom01的搜索窗口，以围起该部件
        ' 部件尺寸 = 100 × 100 p × l
        VGet VSeq1.Geom01.PixelXYU(i), found, PX_X, PX_Y, PX_U
        VSet VSeq2.Geom01.SearchWinLeft, (PX_X - 50)
        VSet VSeq2.Geom01.SearchWinTop, (PX_Y - 50)

        '执行VSeq2
        VRun VSeq2
        VGet VSeq2.Geom01.Found, found

        If found = True Then
            ' 可检测时为表面部件，将其注册至坐标队列中
            ' 本地编号为1
            VGet VSeq1.Geom01.RobotXYU(i), found, RB_X, RB_Y, RB_U
            PF_QueAdd partID, XY(RB_X, RB_Y, RB_Z, RB_U) /1

        Else
            ' 无法检测时为背面部件
            numBack = numBack + 1

        EndIf

    Next

    ' 熄灭外部照明2
    Off O_FrontLight2

    PF_Vision = PF_CALLBACK_SUCCESS

Fend
```

3.4.7 PF_Feeder

客户要使用送料器控制命令（PF_CenterByShift、PF_Center、PF_Flip、PF_Shift的各命令）记述送料器控制运作时使用。

格式

Function PF_Feeder（部件ID As Integer, 表面部件数 As Integer, 背面部件数, 状态 As Integer）As Integer
’（部件状态判定）
’（送料器运作）
Fend

参数

- 部件ID
  输入部件ID（整数值1～32）。
  在多部件运作时有效部件进入。
- 表面部件数
  输入通过视觉系统检测并进入部件坐标队列的表面部件数。
- 背面部件数
  输入通过视觉系统检测的后向部件数。

- 状态
  表示通过系统检测的状态或由系统确定的推荐运作。
  输入下表所述的1个值。

状态/推荐运作	常数（由PartFeeding.inc定义）	备注
可取放	PF_FEEDER_PICKOK	可取放部件。
可添加供给部件	PF_FEEDER_SUPPLY	部件供给方法为[拾取&放置期间供给部件]时，可通过料斗添加供给部件。 有关详细信息，请参阅以下内容。 供给部件
需要翻转运作	PF_FEEDER_FLIP	推荐翻转运作（分散部件或更改姿势）。
移动至拾取区域	PF_FEEDER_SHIFT	推荐移动运作（将部件移动至拾取区域）。
定芯与翻转	PF_FEEDER_CENTER_FLIP	推荐定芯与翻转运作。
料斗变空	PF_FEEDER_HOPPER_EMPTY	料斗内没有部件。通过操作员调用等向料斗供给部件。
后移至拾取区域	PF_FEEDER_SHIFT_BACKWARDS	推荐后移运作（将滞留在平台边角的部件返回至拾取区域）。
通过料斗进给并进行定芯与分离	PF_FEEDER_SUPPLY_CENTER_FLIP	需要通过料斗供给部件，推荐进行部件定芯与分离运作。
部件过多	PF_FEEDER_TOO_MANY	平台上的部件数量过多。通过操作员调用等除去平台上的部件。
混入其他类型部件	PF_FEEDER_WRONGPART	可能是混入了错误类型的部件。通过操作员调用等进行恢复。
不能判定	PF_FEEDER_UNKNOWN	系统无法对设置长槽或孔的平台判断最佳振动。 请参阅以下内容。 振动

返回值

根据PF_Feeder结束后要让系统执行的进程，指定以下值。

常数（由PartFeeding.inc定义）	Part Feeding进程的运作
PF_CALLBACK_SUCCESS	判断为可拾取部件且无必要进行更多送料器运作时指定。系统调用PF_Robot回调函数。注意：不重新生成部件坐标队列。如果在临时进行送料器运作之后恢复该值，送料器上的部件坐标与部件坐标队列的数据则会产生偏差。
PF_CALLBACK_RESTART	在进行送料器运作之后指定。系统重新生成所有的部件坐标队列，并再次调用PF_Feeder回调函数。
PF_CALLBACK_RESTART_ACTIVEPART	在进行送料器运作之后指定。系统重新生成仅限于有效部件的部件坐标队列，并再次调用PF_Feeder回调函数。

描述

用户可利用该函数处理独自的送料器运作。通常，系统会判断送料器振动并执行处理。如果选择[上料] - [部件] - [振动]中的“通过PF_Feeder回调函数控制振动”，系统则会按指定的时序执行PF_Feeder回调函数。可使用PF_Feeder回调函数解决难以适用Part Feeding进程的应用。

例：Part Feeding进程无法判断适合于自定义平台（客户实施孔、长槽等加工的特殊平台）的处理。取而代之，客户可在PF_Feeder回调函数中记述独自的送料器处理。

可利用自变量“表面部件数”与“背面部件数”，判断可否切换为拾取运作或确定适当的振动运作。比如，表面部件数>0时可拾取部件，可判断为不需要更大程度的送料器振动。
自变量“状态”表示推荐的运作或装置状态。这有助于确定送料器的运作类型或条件。

返回值需要返回上表所述的某个值。返回值表示要让系统执行的进程。

程序示例1：
下例所述为利用自变量“状态”按条件，执行送料器振动运作的方法。

```
Function PF_Feeder(PartID As Integer, NumFrontParts As Integer, NumBackParts As
Integer, state As Integer) As Integer

    Select state

        Case PF_FEEDER_PICKOK
            ' Call PF_Robot because there are parts ready to pick
            PF_Feeder = PF_CALLBACK_SUCCESS

        Case PF_FEEDER_SUPPLY
            ' Need to supply more parts
            PFControlReturnVal = PF_Control(PartID, PF_CONTROL_SUPPLY_FIRST)
            ' Shift forward and then Flip
            PF_Shift PartID, PF_SHIFT_FORWARD, 500
            PF_Flip PartID
            ' Restart and re-acquire images
            PF_Feeder = PF_CALLBACK_RESTART

        Case PF_FEEDER_FLIP
            ' Flip the parts
            PF_Flip PartID
            ' Restart and re-acquire images
            PF_Feeder = PF_CALLBACK_RESTART

        Case PF_FEEDER_CENTER_FLIP
            ' Center, Flip and Separate the parts
            PF_Center PartID, PF_CENTER_LONG_AXIS, 900
            PF_Center PartID, PF_CENTER_SHORT_AXIS
            PF_Flip PartID
            ' Restart and re-acquire images
            PF_Feeder = PF_CALLBACK_RESTART

        Case PF_FEEDER_TOO_MANY
            ' Notify operator that there are too many parts on the feeder
            PFStatusReturnVal = PF_Status(PartID, PF_STATUS_TOOMANYPART)
            ' Restart and re-acquire images
            PF_Feeder = PF_CALLBACK_RESTART
    Send

Fend
```

程序示例2：
下例所述为利用自变量“表面部件数”与“背面部件数”，执行送料器振动运作的方法。

```
Function PF_Feeder(PartID As Integer, NumFrontParts As Integer, NumBackParts As
Integer, state As Integer) As Integer
    Integer PFControlReturnVal

    Select True

        Case NumFrontParts = 0 And NumBackParts <> 0
            PF_CenterByShift PartID
            PF_Flip PartID
            ' Restart and re-acquire images
```

```
            PF_Feeder = PF_CALLBACK_RESTART

        Case NumFrontParts = 0 And NumBackParts = 0
            PFControlReturnVal = PF_Control(PartID, PF_CONTROL_SUPPLY_FIRST)
            ' Center, Flip and Separate
            PF_Center 1, PF_CENTER_LONG_AXIS, 900
            PF_Center 1, PF_CENTER_SHORT_AXIS, 700
            PF_Flip 1, 600
            PF_Feeder = PF_CALLBACK_RESTART 're-acquire images

        Default
            ' Call PF_Robot because there are parts ready to pick
            PF_Feeder = PF_CALLBACK_SUCCESS

    Send

Fend
```

3.4.8 PF_CycleStop

记述执行PF_Stop命令后的处理。

格式

Function PF_CycleStop（部件ID As Integer）
’（停止时的处理）
Fend

参数

- 部件ID
  输入部件ID（整数值1～32）。
  在多部件运作时有效部件进入。

返回值

无

描述

执行PF_Stop之后，正在执行的回调函数结束后执行本函数。下面记述装置的处理等。作为记述的内容，包括机器人的原点返回或将机器人的电机置为Off等处理。

程序示例

执行PF_Stop命令之后，使机器人进行原点返回运作，然后进行将电机置为Off的处理。

```
Function PF_CycleStop(partID As Integer)

    Home
    Motor Off

Fend
```

3.5 Part Feeding日志文件

3.5.1 概述

Part Feeding日志文件用于记录通过运作进程进行的以下运作的运作时间与结果。

- 视觉处理

- 送料器运作
- 回调函数运作 (Robot、Status)

Part Feeding日志文件包括下述用途。

- 调查拾取部件的循环时间。
- 调查每次的可拾取部件数量，以用于调整料斗投放数量。
- 根据各运作的处理时间，调查并改善需要时间的运作。

要使用该功能时，需将PC连接至控制器。

3.5.2 启用日志功能

将PC连接至控制器。进行确保在PF_Start之前执行PF_InitLog的记述。
有关详细信息，请参阅以下内容。
PF_InitLog

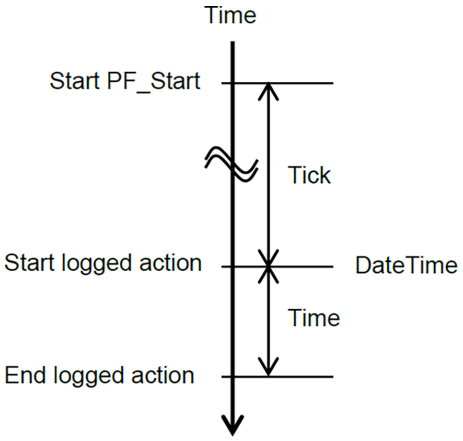
3.5.3 日志文件的格式

3.5.3.1 通用事项

文件为CSV格式。由PF_InitLog的自变量指定文件名。
按时间系列，将以下数据记录到1个日志文件中。“数据”字段因日志类型而异。其他字段均通用。

列	列名	类型	内容
1	DateTime	字符串	运作开始时间 (yyyy/mm/dd hh:MM:ss)
2	Tick	实数	PF_Start开始后的经过时间 [秒] (s.sss)
3	Time	实数	处理时间[秒] (s.sss)
4	Type	字符串	运作类型
5	ID	整数	部件ID
6	Data1	因数据类型 (Type) 而异。	
7	Data2		
8	Data3		
9	PartName	字符串	部件名称
10	RobotNo	整数	分配给部件的机器人编号
11	FeederNo	整数	分配给部件的送料器编号
12	Project	字符串	Epson RC+项目名称

下图所示为DateTime时间与Tick/Time时间的关系。



3.5.3.2 视觉序列运作日志

是记述Part Feeding进程处理由[部件视觉序列]指定的视觉序列所需的时间，以及已检测表面部件与背面部件的数量的日志。
有关详细信息，请参阅以下内容。
视觉

列	列名	类型	内容
4	Type	字符串	日志类型 (“UserVision”)
5	ID	整数	部件ID
6	NumFront	整数	已检测的表面部件数量
7	NumBack	整数	已检测的背面部件数量

3.5.3.3 系统视觉序列运作日志

是记述为了检测部件分布等，Part Feeding进程处理内部生成的视觉序列所需的时间的日志。

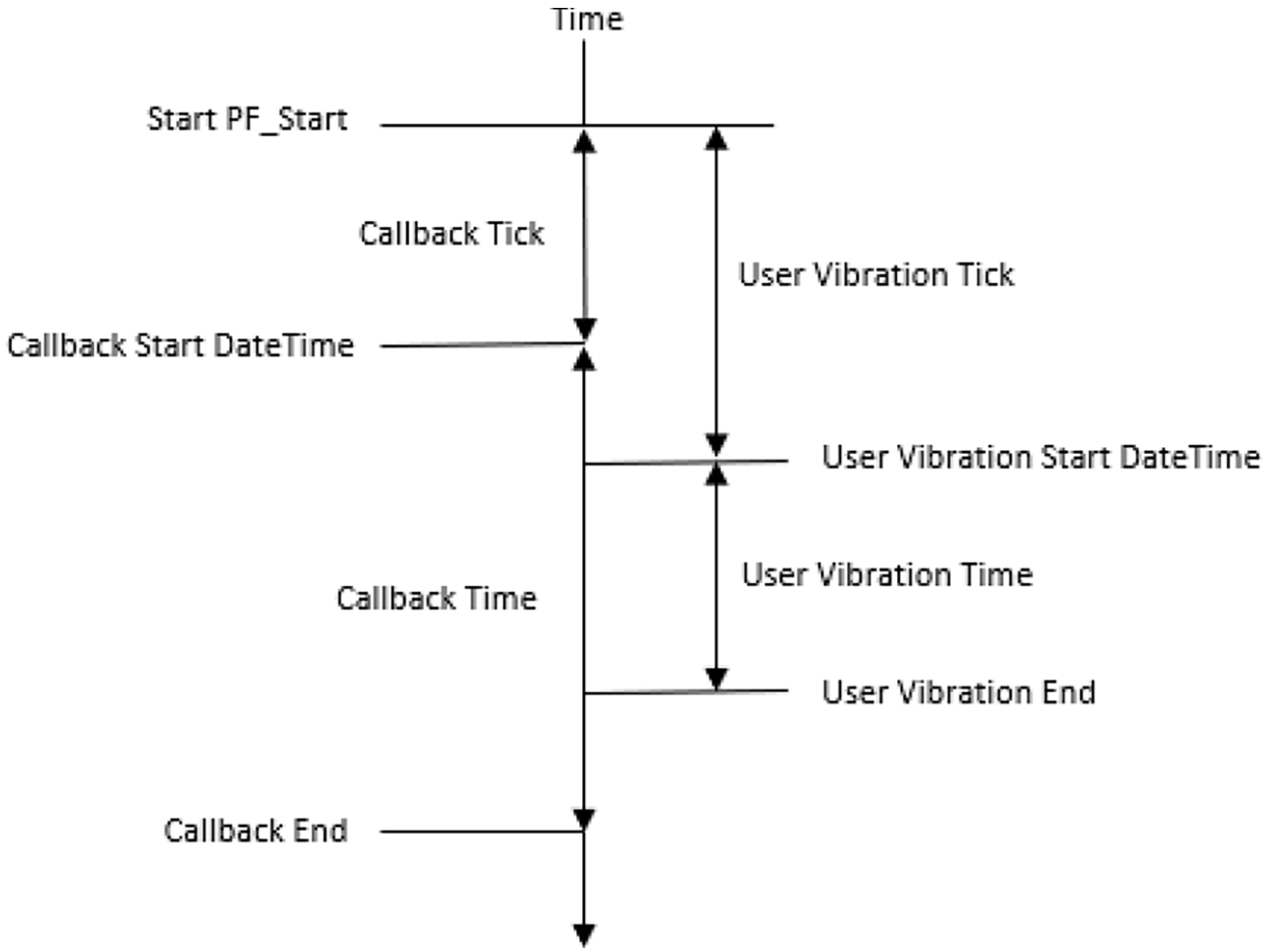
列	列名	类型	内容
4	Type	字符串	日志类型 (“SystemVision”)
5	ID	整数	部件ID

3.5.3.4 振动日志

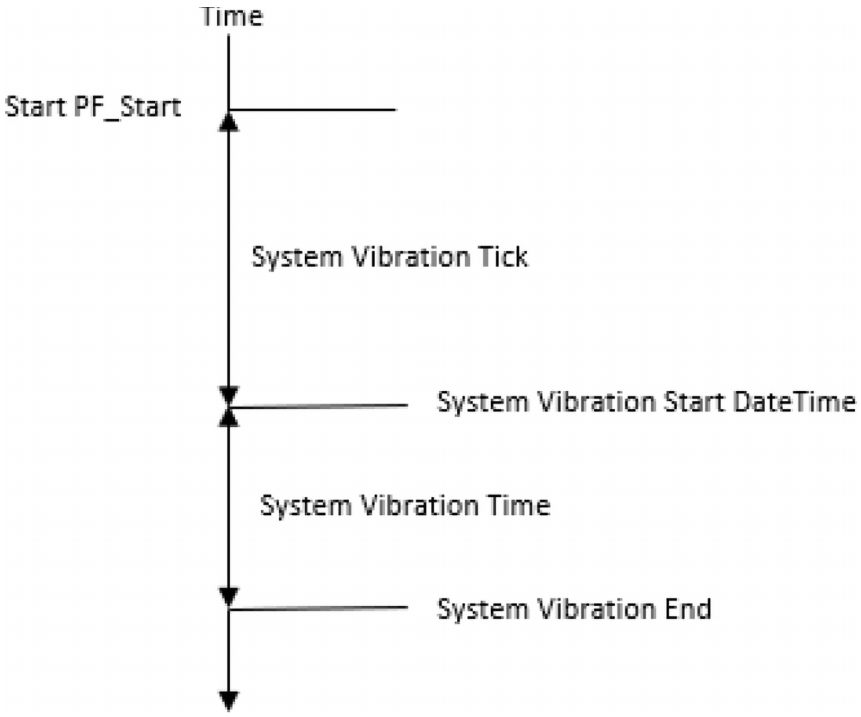
该日志记录有由系统或用户执行的送料器振动运作的类型。

列	列名	类型	内容	
4	Type	字符串	运作类型：	
			Separation	分离
			Centering	定芯
			Shift	移动
			BackShift	后移
			Flip	翻转
			CenterByShift	移动定芯

列	列名	类型	内容	
			Purge	清除
			QtyAdjHopperTime	QtyAdjHopperTime
5	ID	整数	部件ID	
6	Callback Name	字符串	已执行振动的回调函数（或系统）的名称：	
			System	送料器
			Robot	视觉
			Control	MobileCam
			CycleStop	状态



时序图：在回调函数内执行的通过用户操作产生的振动



时序图：系统产生的振动

要点

通过系统进行振动时，日志文件的Data1列中会显示“System”。除此以外时，Data1中显示由用户进行振动的回调函数的名称。

3.5.3.5 PF_Robot回调函数运作日志

是由PF_Robot回调函数处理的部件数的日志。

列	列名	类型	内容
4	Type	字符串	运作类型 (“Robot”)
5	ID	整数	部件ID
6	Num	整数	部件处理数（仅限于有效部件） （调用前的坐标队列注册数 - 调用后的坐标队列注册数）

3.5.3.6 PF_MobileCam回调函数运作日志

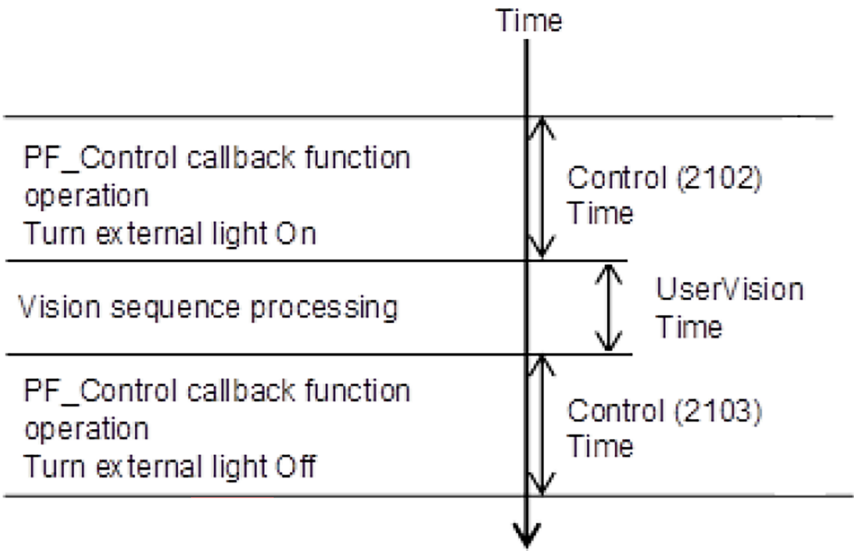
是PF_MobileCam回调函数的运作类型的日志。

列	列名	类型	内容	
4	Type	字符串	运作类型 (“MobileCam”)	
5	ID	整数	部件ID	
6	Action	整数	将机器人移至拍摄位置	2001
			避让机器人	2002

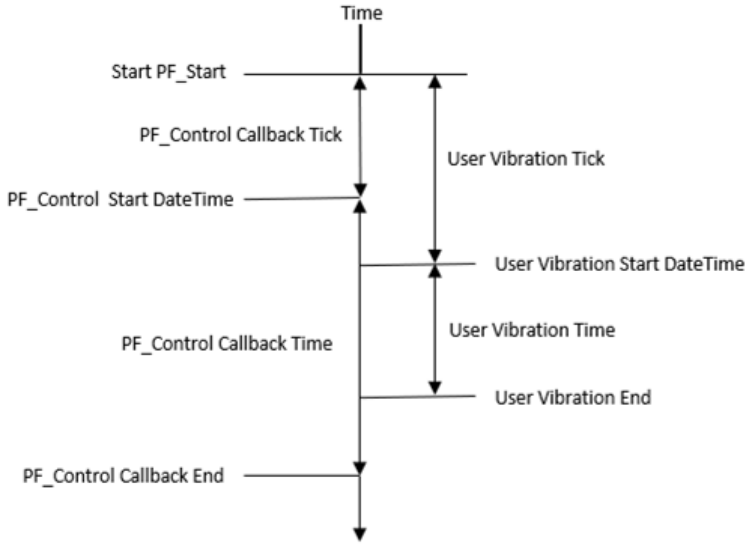
3.5.3.7 PF_Control回调函数运作日志

是PF_Control回调函数的运作类型的日志。

列	列名	类型	内容	
4	Type	字符串	运作类型 (“Control”)	
5	ID	整数	部件ID	
6	Action	整数	料斗操作（部件数0）	2100
			料斗操作（添加部件）	2101
			自定义照明On	2102
			自定义照明Off	2103



时序图：使用前灯的视觉系统



时序图：在PF_Control回调函数内执行的通过用户操作产生的振动

3.5.3.8 PF_Status回调函数运作日志

是PF_Status回调函数的状态类型日志。

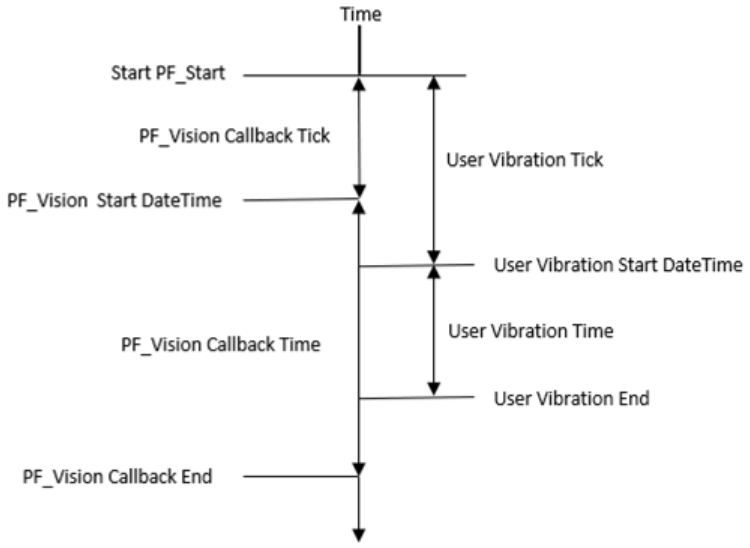
列	列名	类型	内容
4	Type	字符串	运作类型 (“Status”)
5	ID	整数	部件ID

列	列名	类型	内容	
6	Status	整数	已发生的状态或用户错误	
			正常	0
			未供给部件	2200
			部件过多	2201
			ID错误	2202
			参数错误	2203
			校准未完成	2204
			系统错误	2205
			有无法检测的部件	2206
			部件Blob序列异常	2207
			部件视觉序列异常	2208
			送料器-正在使用	2209
			部件无效	2210
			清除无效	2211
			用户错误	8000～8999

3.5.3.9 PF_Vision回调函数运作日志

是通过PF_Vision回调函数检测的表面部件与背面部件的次数的日志。

列	列名	类型	内容
4	Type	字符串	运作类型 (“VisionCallback”)
5	ID	整数	部件ID
6	NumFront	整数	表面部件检测次数
7	NumBack	整数	背面部件检测次数

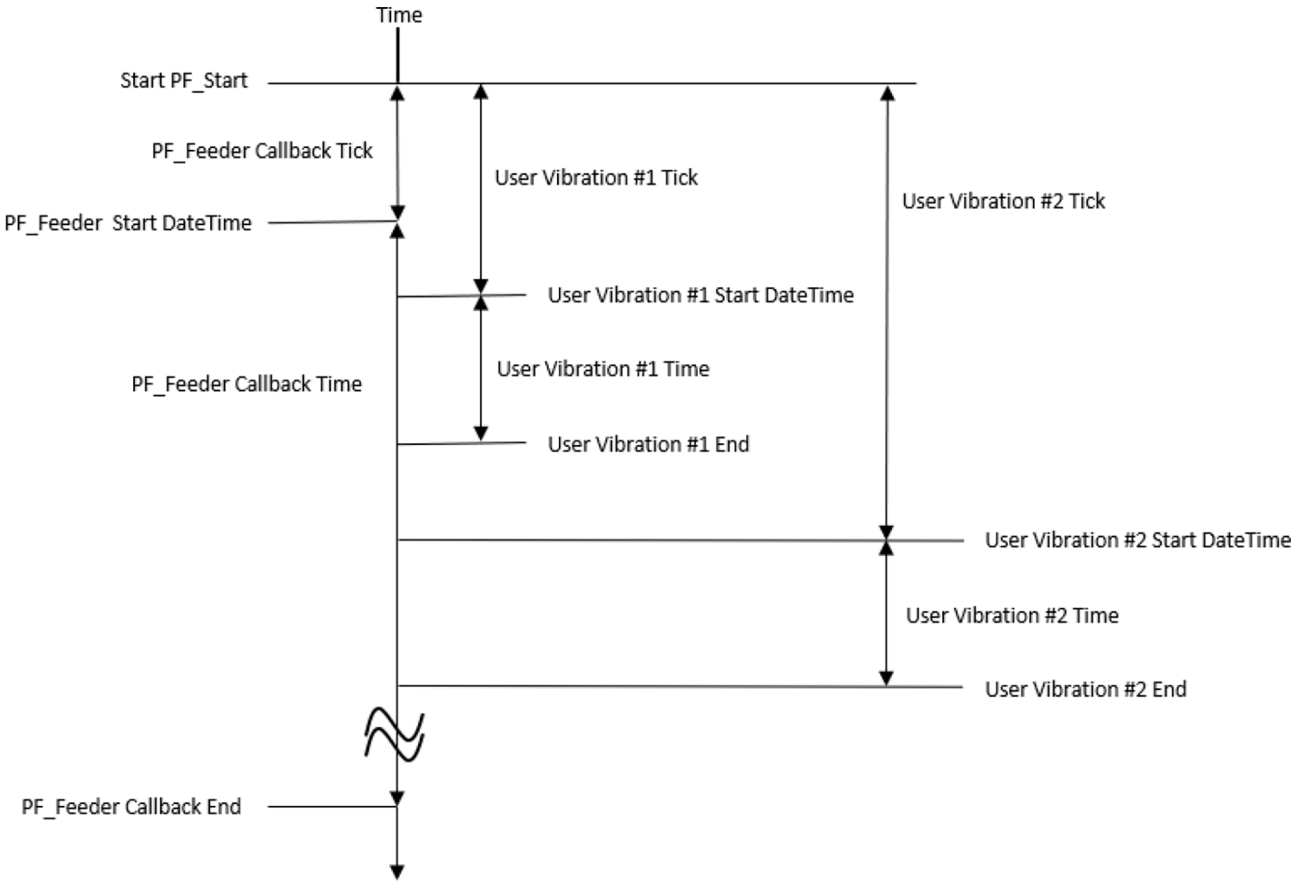


时序图：在PF_Vision回调函数内执行的通过用户操作产生的振动

3.5.3.10 PF_Feeder回调函数运作日志

是PF_Feeder回调函数的姿态日志。

列	列名	类型	内容	
4	Type	字符串	运作类型 (“Feeder”)	
5	ID	整数	部件ID	
6	State	整数	推荐运作：	
			不能判定	0
			可取放	1
			可添加供给部件	2
			翻转	3
			移动	4
			定芯与翻转	5
			料斗变空	6
			向后移动	7
			通过料斗进给并进行定芯与翻转	8
			部件过多	9
			混入其他类型部件	10



时序图：在PF_Feeder回调函数内执行的通过用户操作产生的振动

3.5.3.11 PF_CycleStop回调函数运作日志

为PF_CycleStop回调函数的状态日志。

列	列名	类型	内容
4	Type	字符串	日志类型 (“CycleStop”)
5	ID	整数	部件ID

3.5.4 日志样本

如下所述为Part Feeding日志的样本。

```
DateTime,Tick,Time,Type,ID,Data1,Data2,Data3
2018/04/07 11:13:44,0.199,0.010,MobileCam,1,2001
2018/04/07 11:13:44,0.220,0.000,Status,1,0
2018/04/07 11:13:44,0.531,0.257,SystemVision,1
2018/04/07 11:13:44,0.798,0.512,UserVision,1,0,0
2018/04/07 11:13:45,1.483,10.232,Control,1,2100
2018/04/07 11:13:55,11.725,-0.000,Status,1,0
2018/04/07 11:13:55,11.736,3.740,Separation,1
2018/04/07 11:13:59,15.486,0.011,MobileCam,1,2001
2018/04/07 11:13:59,15.508,-0.000,Status,1,0
2018/04/07 11:13:59,15.819,0.259,SystemVision,1
2018/04/07 11:14:00,16.088,4.236,UserVision,1,1,24,57
2018/04/07 11:14:04,20.545,13.274,Control,1,2101
2018/04/07 11:14:17,33.829,0.000,Status,1,0
2018/04/07 11:14:17,33.840,0.011,MobileCam,1,2002
2018/04/07 11:14:17,33.862,0.000,Status,1,0
2018/04/07 11:14:17,33.873,22.792,Robot,1,24
2018/04/07 11:14:40,56.675,0.000,Status,1,0
2018/04/07 11:14:40,56.686,0.011,MobileCam,1,2001
2018/04/07 11:14:40,56.708,-0.000,Status,1,0
2018/04/07 11:14:40,57.019,0.257,SystemVision,1
2018/04/07 11:14:41,57.495,1.687,Shift,1
2018/04/07 11:14:43,59.192,0.010,MobileCam,1,2001
2018/04/07 11:14:43,59.214,-0.000,Status,1,0
2018/04/07 11:14:43,59.524,0.259,SystemVision
2018/04/07 11:14:43,59.793,4.208,UserVision,1,1,25,61
2018/04/07 11:14:48,64.213,1.600,Control,1,2101
2018/04/07 11:14:49,65.823,0.000,Status,1,0
2018/04/07 11:14:49,65.834,0.011,MobileCam,1,2002
2018/04/07 11:14:49,65.856,-0.000,Status,1,0
2018/04/07 11:14:49,65.867,24.044,Robot,1,25
2018/04/07 11:15:13,89.921,0.000,Status,1,0
```

3.6 Part Feeding选件使用的视觉序列

使用Part Feeding时，需要创建以下2个视觉序列。

- 送料器校准用视觉序列
- 部件检测用视觉序列

有关视觉引导、视觉序列对象的详细信息，请参阅以下手册。
“Vision Guide - 软件篇”
有关视觉属性的详细信息，请参阅以下手册。
“Vision Guide - 属性&结果参考”

3.6.1 视觉校准

在送料器的平台面上进行视觉校准。
有关视觉校准的方法，请参阅以下手册。
“Vision Guide 8.0软件篇 - 视觉校准”

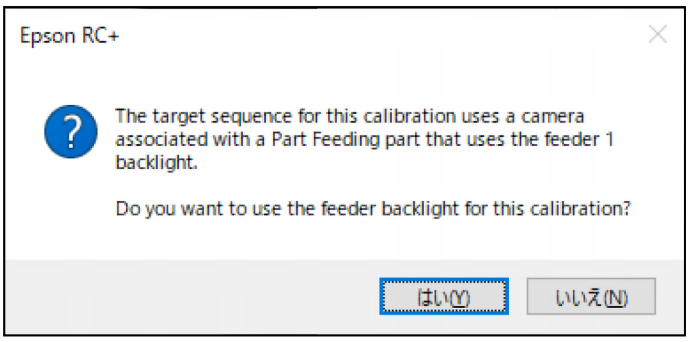
如下所述为视觉校准的重要属性。

属性	设定方法
Camera	设定相机编号。
CameraOrientation	设定相机的固定方法。 向下固定相机-Fixed downward 移动相机-Mobile J4或Mobile J6
RobotLocal	指定机器人本地编号。
RobotNumber	指定机器人编号。 对准以下指定的机器人编号。 常规
RobotTool	指定机器人工具编号。

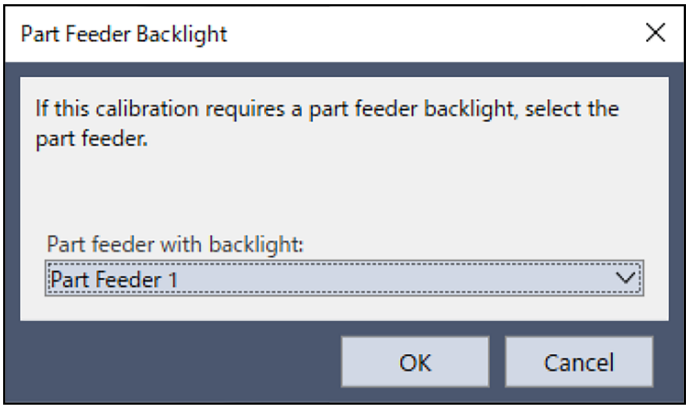
带有背光灯的送料器大于等于1个时：

要点

校准的TargetSequence相机与部件序列相同时，会显示下述信息。



校准的TargetSequence相机未用于部件序列时，会显示下述对话框。



3.6.2 部件检测视觉序列

部件检测视觉序列用于检测部件并获取部件坐标。按下述要领创建视觉序列。

要点

请在送料器校准用视觉序列之外，另行创建部件检测视觉序列。

3.6.2.1 单纯的部件

下面说明在不考虑姿势（表里）和旋转量的前提下，可通过单纯的抓取来检测部件的视觉序列的创建示例。

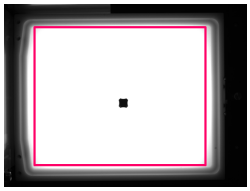
- 视觉序列
  设定以下属性。请务必设定或确认以下项目。根据需要，也设定除此之外的属性。

属性	设定方法
Calibration	设定送料器拍摄相机的视觉校准。 指定与由校准用视觉序列指定的内容相同的内容。 需要完成视觉校准。
Camera	设定送料器拍摄相机的相机编号。 指定与由校准用视觉序列指定的内容相同的内容。
ExposureTime	在将送料器背光灯置为On的状态下进行调整，以确保可顺利地拍摄部件。另外，进行旨在不会使送料器周围显得黑暗的调整。

- 视觉对象
  注册1个可获取机器人坐标的下述某个对象。
  - ArcFinder
  - ArcInspector
  - Blob
  - BoxFinder
  - ColorMatch
  - CornerFinder
  - Correlation
  - DefectFinder
  - Edge

- Geometric
- LineInspector
- Point
- Polar

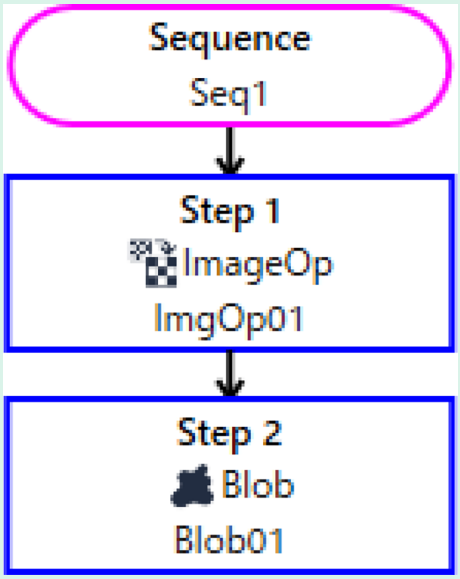
在对象中设定以下属性。请务必设定或确认以下项目。根据对象的类型，也设定除此之外的属性。

属性	设定方法
NumberToFind	设为All。
SearchWindow	对准平台的内圈。另外，进行旨在使平台周边的黑暗部分不会进入的设定。 

要点

也可以并用其他项目。

例：要进行二值化处理时，使用ImageOp对象



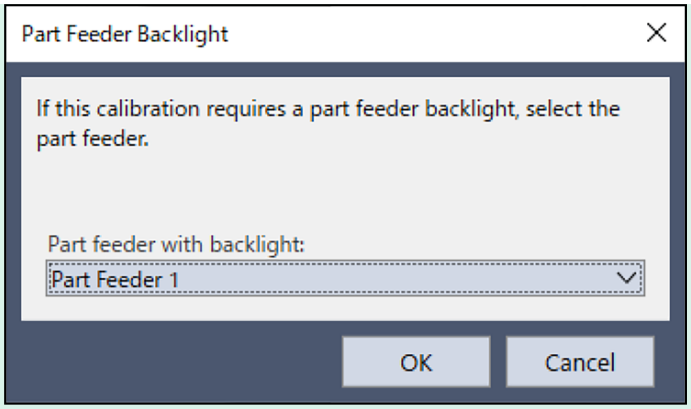
要点

创建了新序列，带有背光灯的零件送料器大于等于1个时，会根据需要，显示带有背光灯的送料器选择对话框。

要通过Epson RC+ Vision Guide窗口执行视觉序列或对象时，背光灯会根据选择自动置为ON。

带有背光灯的送料器为IF-80时，背光灯会在拍摄图像后置为OFF，视频也会临时静止。通过静止，即使背光灯置为OFF，也可以确认图像。如果单击视频区域，则可切换为直播视频。

主要在创建新项目后却未添加供给部件时会发生这种现象。



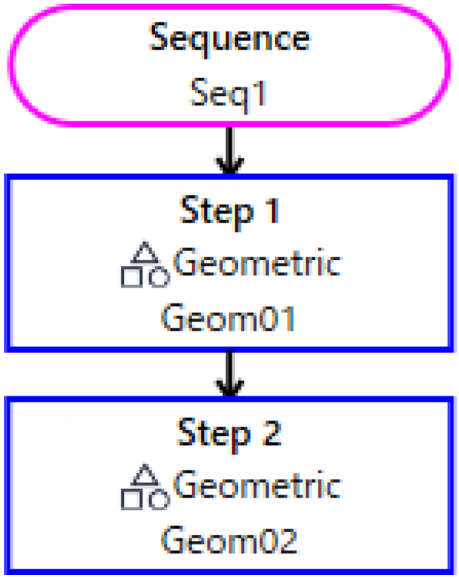
3.6.2.2 带有表里的部件

属于带有表里的部件，而要获取某1方时，按如下所述进行设定。

- 视觉序列
  参阅以下内容，按相同的方式进行创建。
  单纯的部件

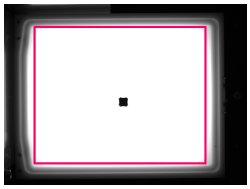
- 视觉对象

1. 注册1个用于检测部件表面的对象（例：Geometric）。
2. 注册1个用于检测部件背面的对象（例：Geometric）。
   例：要进行表里识别时，创建2个Geometric。（Geom01表面检测用、Geom02背面检测用）



如下所述为Geometric的属性设定示例。

属性	设定方法
Accept	进行可以可靠地检测部件的设定。
NumberToFind	设为All。

属性	设定方法
SearchWindow	对准平台的内圈。 
ModelWindow	对准部件的外圈，然后对部件进行示教。

要点

要使用Geometric或Correlation对象时，将Timeout属性设定得略长于执行序列所需的常规时间，是一种有效的方法。未设定时，处理时间最长可能需要2000毫秒（默认）。

示例：

在Geometric对象中设为NumberToFind = 9

有多个部件

执行VRun

→ 视觉系统时间为75毫秒（例）

设为NumberToFind = All

执行VRun

→ 视觉系统时间约为2000毫秒（超时）

将Timeout属性设为100毫秒

执行VRun

→ 视觉系统时间约为100毫秒（超时）

这样的话，在需要时间的应用中正确设定超时就显得至关重要。

3.6.2.3 考虑机器人末端夹具的空间时

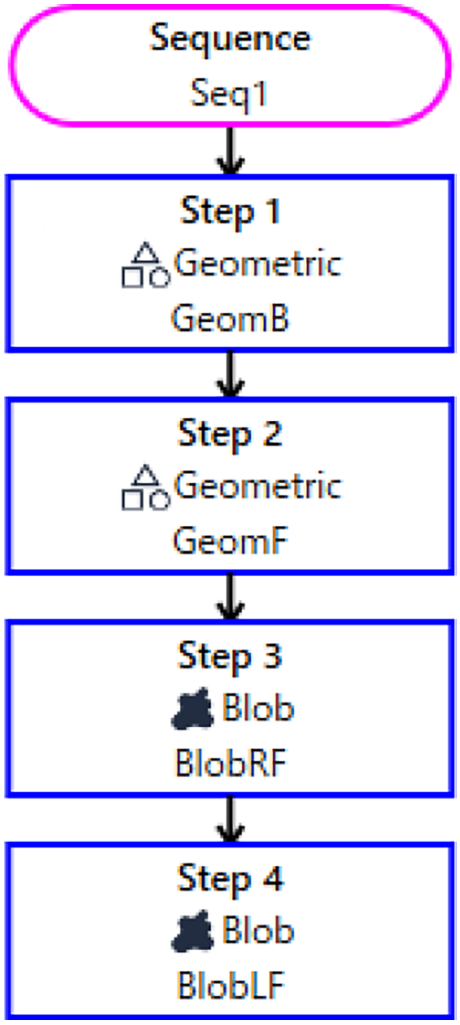
通过模式匹配（Geometric、Correlation）进行检测时，即使部件部分重叠也进行检测，但可能会对机器人的拾取带来影响。要将这类部件除外时，按如下所述进行设定。

- 视觉序列
  参阅以下内容，按相同的方式进行创建。
  单纯的部件

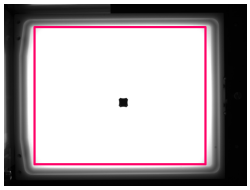
- 视觉对象

1. 注册2个用于检测部件的对象。（例：Geometric）
   例：创建用于表面检测的GeomF以及用于背面检测的GeomB。

2. 添加用于在部件的周围确认机器人末端夹具进入空间的视觉对象。
例：创建Blob对象以用于确认空间。设定Blob对象的CheckClearanceFor属性。



Geometric的属性设定示例

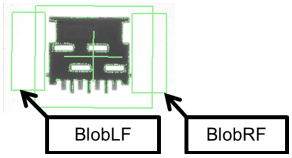
属性	设置示例
Accept	进行可以可靠地检测部件的设定。
NumberToFind	设为“All”。
SearchWin	对准平台的内圈。 
ModelWin	对准部件的外圈，然后对部件进行示教。
Name	GeomB：检测背面部件的视觉对象 GeomF：检测表面部件的视觉对象

Blob的属性设定示例

属性	设置示例
CheckClearanceFor	GeomF 检测表面部件的视觉对象

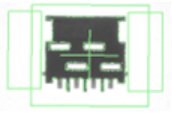
属性	设置示例
ClearanceCondition	NotFound
FailColor	LightGreen
Name	BlobRF：用于确认机器人末端夹具右爪进入空间的对象 BlobLF：用于机器人末端夹具的左爪
PassColor	Red
SearchWin Type	RotatedRectangle
ThresholdHigh	192

本例所述是确认仅表面部件时机器人末端夹具手爪进入的空间的情况。

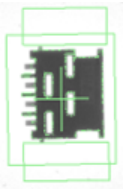


1. 将1个部件放到送料器的中央。

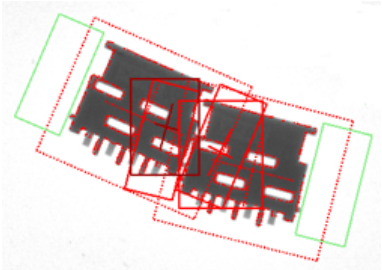
2. 执行视觉序列。



3. 将放到送料器中央的部件转动90°，然后再次执行视觉序列。



请确认Blob的SearchWindow随着部件的旋转而旋转。上图所示为有机器人末端夹具手爪进入空间时的检测结果。没有空间时，会得到下述检测结果。



3.6.2.4 构成特殊视觉

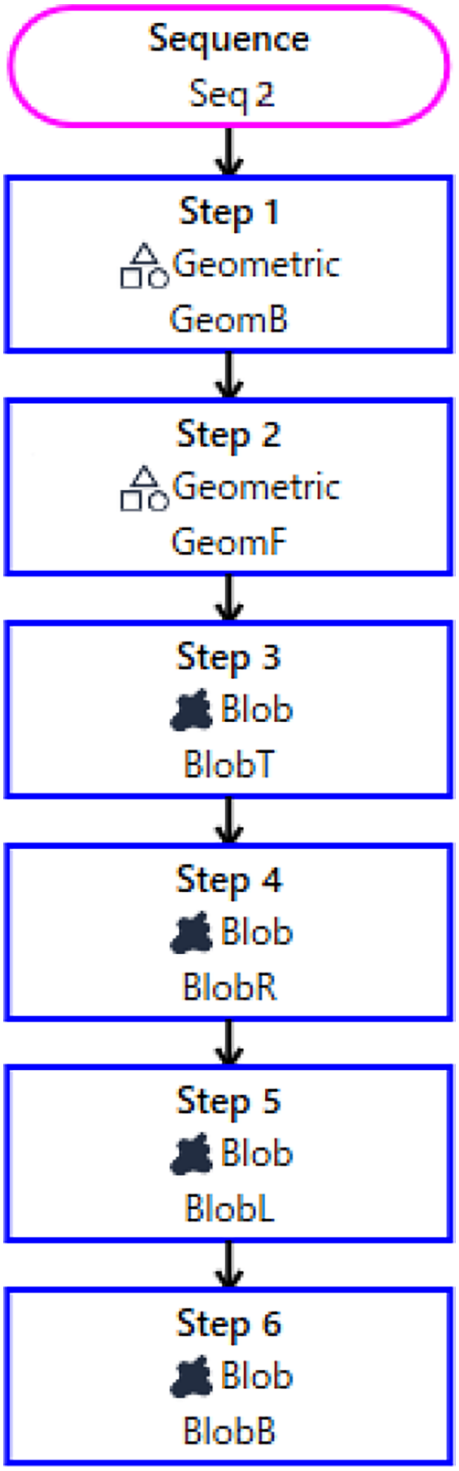
使用大于等于2个的视觉序列或多个照明进行部件检测时，可记述启用视觉回调函数、控制照明与检测部件的所有内容。请参阅以下内容。

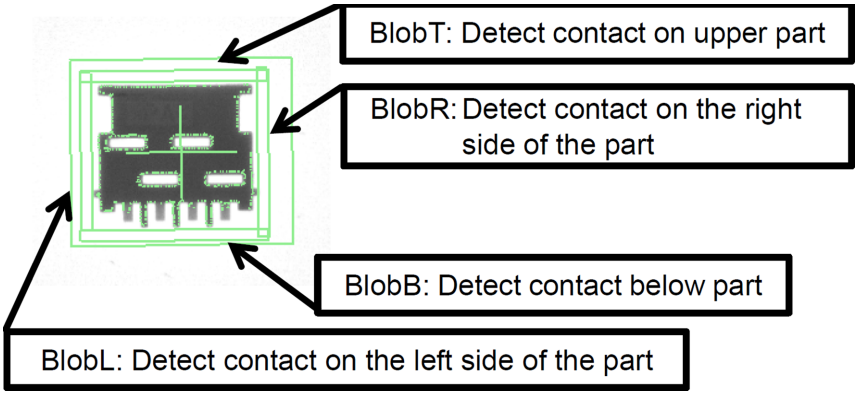
PF_Vision

3.6.2.5 碰到相邻部件时的不拾取示例

如下所述为检测到碰到相邻部件或相互间隙较小并设为拾取对象之外的示例。参阅以下内容，按相同的方式创建视觉序列。

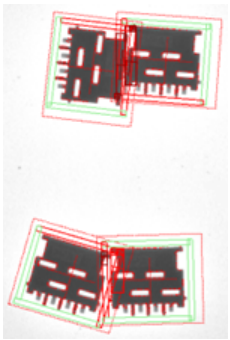
考虑机器人末端夹具的空间时





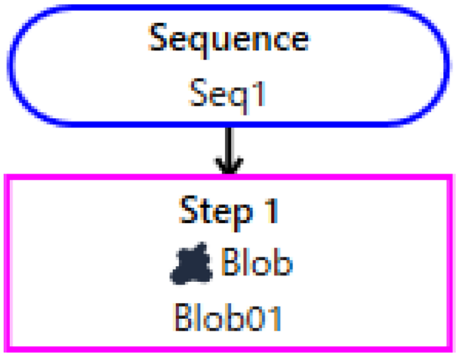
未碰到相邻部件时的检测示例：

碰到相邻部件时的检测示例：



3.6.3 部件Blob视觉序列

创建送料器校准使用的视觉序列。



要点

请在部件检测用视觉序列之外，另行创建送料器校准用视觉序列。

3.6.3.1 视觉序列

设定以下属性。请务必设定或确认以下项目。请将除此之外的属性设为默认值。

属性	设定方法
Calibration	参阅以下内容，设定已创建的视觉校准。 视觉校准 需要完成视觉校准（CalComplete结果为“True”）。
Camera	设定送料器拍摄相机的相机编号。
ExposureTime	在将送料器背光灯置为On的状态下进行调整，以确保可适当地拍摄部件。另外，进行旨在不会使平台周边显得黑暗的调整。 易于透光的部件需要慎重设定该值。

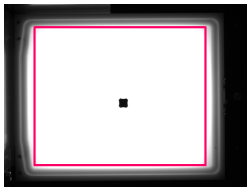
要点

送料器上有黑暗部分或脏污等难以检测的部分时，可利用“检测掩膜”从检测对象中排除。

有关详细信息，请参阅以下内容。
程序示例 8.3

3.6.3.2 视觉对象

设定以下属性。请务必进行设定或确认。请将下述以外的属性设为默认值。

属性	设定方法
MaxArea	保持默认值（相机的width × height）。
MinArea	设为约为部件面积的0.9倍的值。 按以下步骤调查部件面积。 1. 点亮送料器的背光灯 2. 将几个部件放到平台上，注意不要产生重叠 3. 执行视觉序列 4. 将Blob结果的Area平均值设为部件面积
NumberToFind	设为“All”。
SearchWin	尽可能准确地对准平台的内圈。另外，进行旨在使平台周边的黑暗部分不会进入的设定。  要旋转搜索窗口（在[Type]中选择[Rotated Rectangle]）时，请将旋转角度设为±45° 以内。
ThresholdColor	设为“Black”。

属性	设定方法
ThresholdHigh	进行可检测部件并且不检测平台外圈等黑暗部分的设定。 易于透光的部件需要慎重设定该值。
ThresholdLow	设为“0”。

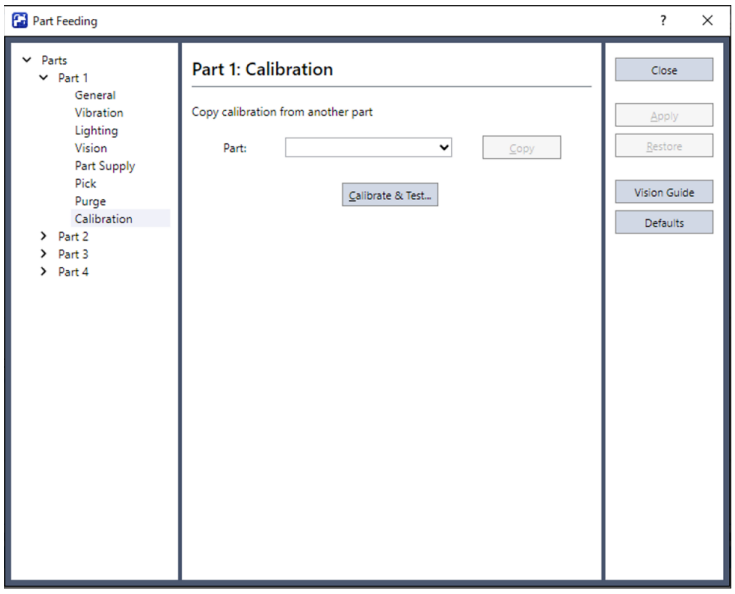
3.7 料斗的调整方法

下面说明料斗的调整方法示例。

3.7.1 料斗（Gen.1）的调整方法

料斗与送料器的设置以及连接应已完成。

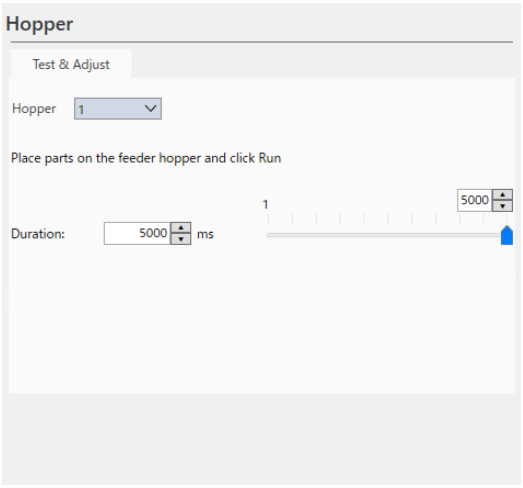
1. 在上料对话框中选择[校准]，然后单击[校准&测试]。



2. 选择料斗选项卡。

3. 向料斗投放最佳投放部件数的5～10倍的部件。可在[最佳投放部件数]选项卡中确认最佳投放部件数。

4. 仅可在测试与调整画面中调整振动时间。通过料斗控制器手动调整振动振幅。



5. 逐渐增大料斗控制器的振动振幅刻度盘，以确保部件的投放速度保持恒定。调整振动时间，以确保按下[执行]按钮时供给适当数量的部件。

可使用PF_Hopper、PF_OutputOnOff命令控制料斗 (Gen.1)。任何一个命令都会进行相同的运作。

要连接2台料斗 (Gen.1) 时，使用PF_Output命令。
有关命令示例，请参阅以下内容。
Part Feeding SPEL+命令参考

测试步骤

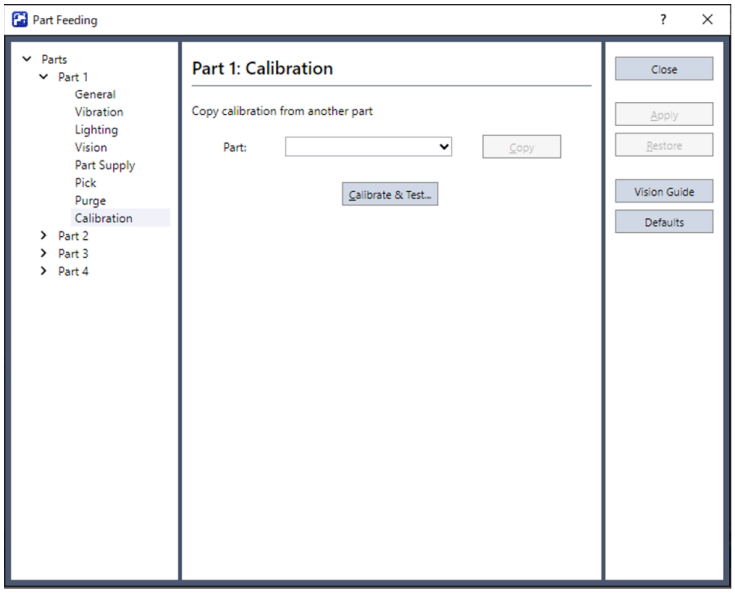
1. 向料斗中投放部件。
2. 单击[执行]按钮。
3. 确认是否供给最佳数量的部件。

根据需要调整参数，然后将部件送回料斗，再次单击[执行]按钮。

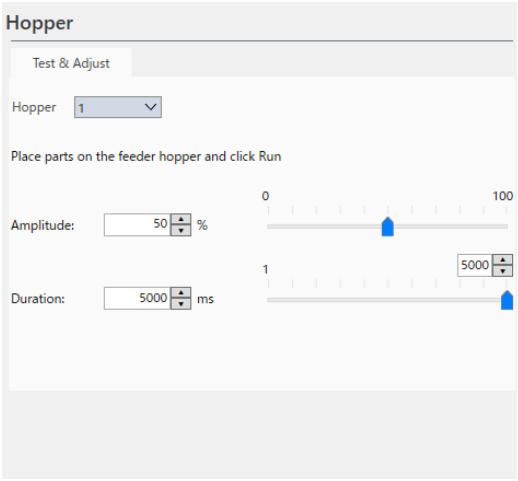
3.7.2 料斗 (Gen.2) 的调整方法

料斗与送料器的设置以及连接应已完成。

1. 在上料对话框中选择[校准]，然后单击[校准&测试]。



2. 选择料斗选项卡。
3. 向料斗投放最佳投放部件数的5～10倍的部件。可在[最佳投放部件数]选项卡中确认最佳投放部件数。
4. 可在测试与调整画面中调整振动振幅与振动时间。



5. 调整振幅，以确保来自料斗的部件的投放速度保持恒定。调整振动时间，以确保按下[Run]按钮时供给适当数量的部件。

可使用PF_Hopper命令控制料斗 (Gen.2)。
有关命令示例，请参阅以下内容。
Part Feeding SPEL+命令参考

如果设定的值超过下表推荐的最大振动振幅值，则控制器无法达到目标振幅。随着料斗变空，振动振幅会增大。

要点

- 调整振动振幅与振动时间之前实施校准。有关详细信息，请参阅以下手册。
  “Epson RC+ 8.0选件 Part Feeding 8.0 Hopper篇”
- 料斗 (Gen.2) 内置有智能传感器，用于调整振动振幅。料斗检测当前的部件数量，并自动调整振动振幅，以确保稳定地供给部件。

料斗S (1L/2L) 的最大振动振幅

部件数量	振动振幅推荐值
< 0.5 kg	小于等于100%
< 1 kg	小于等于75%
< 1.5 kg	小于等于50%
< 2 kg	小于等于25%

料斗M (3L/7L) 的最大振动振幅

部件数量	振动振幅推荐值
< 4 kg	小于等于100%
< 6 kg	小于等于75%
< 9 kg	小于等于50%
< 12 kg	小于等于25%

料斗L (14L) 的最大振动振幅

部件数量	振动振幅推荐值
< 5 kg	小于等于100%
< 10 kg	小于等于75%
< 15 kg	小于等于50%
< 20 kg	小于等于25%

测试步骤

1. 向料斗中投放部件。
2. 单击[执行]按钮。
3. 确认是否供给最佳数量的部件。

根据需要调整参数，然后将部件送回料斗，再次单击[执行]按钮。

3.7.3 IF-80料斗的调整方法

IF-80为料斗一体型型号。下面说明料斗的调整示例。
实施料斗校准。有关详细信息，请参阅以下内容。
料斗 - 测试与调整

修正程序。可使用PF_Hopper、PF_OutputOnOff命令控制IF-80料斗。任何一个命令都会进行相同的运作。PF_Hopper命令对应于料斗 (Gen.1)、料斗 (Gen.2) 以及IF-80所有类型的料斗。

有关命令的详细信息，请参阅以下内容。
Part Feeding SPEL+命令参考

3.8 RC+操作期间发生的错误

信息	原因与措施
未设定相机。 请在[设置] - [Vision Guide设定]中设定相机。	未将相机注册至系统中。 请注册相机。
虚拟控制器不支持Part Feeding。	Part Feeding选件不支持虚拟控制器。 请连接至控制器。
未安装或未启用上料选件。	未启用Part Feeding选件。 本选件收费。 请从销售商采购选件密钥代码，然后进行设定。
未设定有效的送料器。	未将送料器注册至系统中。或未启用。 请注册或启用送料器。
未设定校准视觉序列。	未指定校准用视觉序列。 请在上料窗口的视觉页面中指定校准用视觉序列。
不能添加更多的部件。	可注册至1个项目中的部件类型最多为32种。 使用之前，请删除未使用的部件或覆盖未使用部件的参数。
需要对错误的视觉序列机器人相机进行校准。	未在部件检测用视觉序列或送料器校准用视觉序列中设定视觉校准。 请在各视觉序列的Calibration属性中，设定有效的视觉校准。

信息	原因与措施
校准错误：部件过多。	送料器校准运作期间，部件的投放数量过多。 或因视觉设定不当而错误地检测部件。请正确地投放画面中显示数量的部件。或重新评估视觉设定。
校准错误：部件过少。	送料器校准运作期间，部件的投放数量过少。或因视觉设定不当而无法检测部件。请正确地投放画面中显示数量的部件。或重新评估视觉设定。
无法检测部件。	送料器校准运作期间，未投放部件。或因视觉设定不当而无法检测部件。 请正确地投放画面中显示数量的部件。或重新评估视觉设定。
送料器通信端口的打开/关闭失败。请确认送料器的连接。	1） 送料器的连接已被切断。请确认送料器与控制器之间的以太网连接是否正常（有无电缆断线、集线器故障或未供电现象）。请确认送料器的电源。 2） 送料器的通信设定（送料器的型号、IP地址、子网掩码、端口）有误。请重新评估设定。
无法连接至送料器。	（同上）
已指定未定义函数。	请退出[上料]窗口，重建项目。
在当前的设定值的条件下，无法连接送料器。	送料器的通信设定（送料器的型号、IP地址、子网掩码、端口）有误。请重新评估设定。
要使用上料功能时，该版本的Epson RC+ 8.0中的控制器固件需为 ** 以后版本。	控制器版本与RC+版本不一致。请更新控制器固件。
已将上料部件**设为可供零件送料器**（型号**）使用，但控制器的送料器**为型号**。如果更改零件送料器的型号，则需要重新校准。是否继续？	系统配置中的送料器型号已被更改，导致与部件的送料器设定不匹配。 请将系统配置中的送料器型号恢复原样，或实施对象部件的重新校准。
零件送料器固件版本（ ** ）不支持该版本的Epson RC+ 8.0。该型号**零件送料器的固件版本需为**以后版本。	1） 上料功能不支持该送料器的固件版本。请更新送料器的固件。 2） 已连接其他公司送料器。请使用从本公司采购的送料器。
对上料部件使用了固件版本为**的紧凑型视觉系统。要使用上料功能时，紧凑型视觉系统的固件版本应为**以后版本。	1） 要在上料功能中使用CV时，CV的固件版本需为3.0.0.0以后版本。请更新CV固件。 2） 要在上料功能中使用CV时，请使用CV2-SA/HA、CV2-HB/SB/LB。不支持CV1与CV2-S/H/L。

3.9 应用程序示例

下面说明适用于各种状况的应用程序示例。

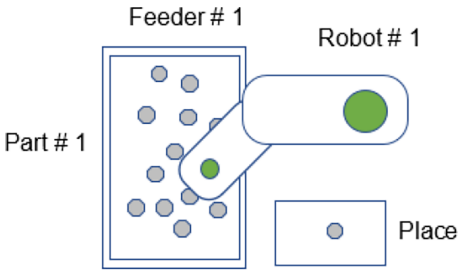
3.9.1 每台送料器1台机器人&每台送料器1种部件

3.9.1.1 程序示例 1.1

示例类型：
将PF_Robot回调函数用于运动

构成

- 机器人数量：1
- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：1
  相机的方向：送料器#1上的固定向下相机



描述

将部件从送料器移动至箱中。移动完所有部件时，Control回调函数用于发出部件请求。如果通过操作员或料斗向送料器补充部件，则会继续执行移动循环。如果执行PF_Stop命令，则停止当前的循环并结束应用。（以下样本代码未使用PF_Stop命令。）

样本代码

Main.prg

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park
    PF_Start 1
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    Do While PF_QueLen(PartID) > 0
        P0 = PF_QueGet(PartID)
        Jump P0
        On Gripper; Wait 0.2
        Jump Place
        Off Gripper; Wait 0.2
        PF_QueRemove PartID
        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf
    Loop
    PF_Robot = PF_CALLBACK_SUCCESS
 Fend
```

3.9.1.2 程序示例 1.2

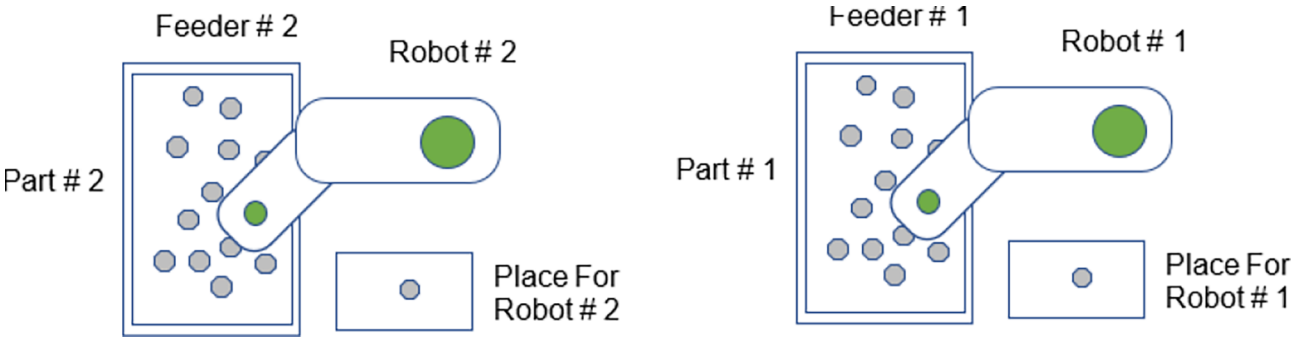
示例类型：

多机器人系统 - 每台送料器1台机器人&每台送料器1种部件

构成

- 机器人数量：2
  将机器人1连接至控制单元。将机器人2连接至驱动单元。

- 送料器数量：2
- 送料器上的部件类型数量：1
- 配置位置数量：每台机器人1处
- 相机的方向：在各送料器上配置固定向下相机



描述

各机器人拾取&放置各部件。每台机器人都有专用的送料器与放置位置。另外，机器人1与机器人2的运作范围不重叠。
各机器人的各个点文件中带有已进行“Park”、“Pick”、“Place”示教的点。

样本代码

Main.prg

```
Function main
    Robot 1
    If Motor = Off Then
         Motor On
    EndIf
    Power Low
    Jump Park

    Robot 2
    If Motor = Off Then Motor On
    Power Low
    Jump Park

    PF_Start 1
    PF_Start 2
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    Integer gripperOutput

    Select PartID
        Case 1
            Robot 1
            gripperOutput = 1
        Case 2
            Robot 2
            gripperOutput = 2
    Send

    Do While PF_QueLen(PartID) > 0
        Pick = PF_QueGet(PartID)
        Jump pick
        On gripperOutput; Wait 0.2
        Jump Place
        Off gripperOutput; Wait 0.2
        PF_QueRemove PartID
```

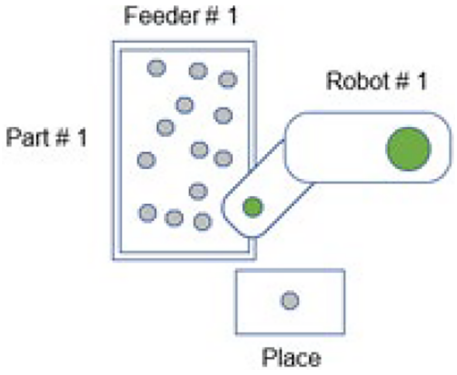
```
        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf
    Loop
    PF_Robot = PF_CALLBACK_SUCCESS

Fend
```

3.9.1.3 程序示例 1.3

示例类型：
通过机器人运作进行视觉系统与振动的并行处理

构成

- 机器人数量：1
- 送料器数量：1
- 部件类型数量：1
- 配置位置数量：1
- 相机的方向：固定向下相机



描述
机器人从零件送料器中拾取部件#1，放置到固定夹具上。该运作持续到从送料器移动完所有的“表面”部件为止。最后的部件被放置时（送料器运作的90%阶段），送料器振动，并根据需要通过料斗向送料器补充部件。
本例所述为与机器人运动并行运作送料器以优化处理能力的具体内容。

样本代码
Main.prg

```
Function main
    MemOff PartsToPick
    Off Gripper

    Robot 1
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park

    PF_Start(1)

    Xqt RobotPickPlace
Fend

Function RobotPickPlace
    Do
        Wait MemSw(PartsToPick) = On
```

```
        If PF_QueLen(1) > 0 Then
            Do
                Pick = PF_QueGet(1)
                PF_QueRemove 1
                Jump pick
                On Gripper; Wait 0.2
                If PF_QueLen(1) = 0 Then
                    Jump Place ! D90; MemOff PartsToPick !
                    Off Gripper; Wait 0.2
                    Exit Do
                Else
                    Jump Place
                    Off Gripper; Wait 0.2
                EndIf
            Loop
        Else
            MemOff PartsToPick
        EndIf
    Loop
Fend
```

PartFeeding.prg

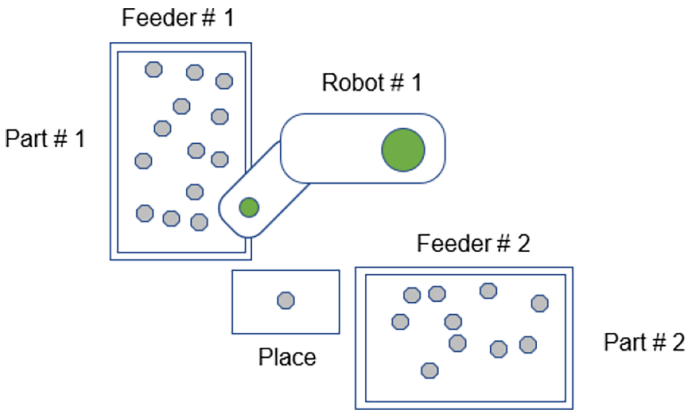
```
Function PF_Robot(PartID As Integer) As Integer
    MemOn PartsToPick
    Wait MemSw(PartsToPick) = Off

    PF_Robot = PF_CALLBACK_SUCCESS
Fend
```

3.9.1.4 程序示例 1.4

示例类型：
每台送料器1台机器人；每2台送料器1种部件

构成

- 机器人数量：1
- 送料器数量：2
- 各送料器上的部件类型数量：1
- 配置位置数量：1
- 相机的方向：在各送料器上配置固定向下相机



描述
机器人从送料器#1中拾取部件#1，将其放置到箱中。该运作持续到从送料器#1移动完所有的“表面”部件为止。此后，机器人从送料器#2上拾取“表面”部件，将其移动至箱中。相机与送料器并行运作。

样本代码

Main.prg

```
Function main
    MemOff PartsToPick1; MemOff PartsToPick2
    Off Gripper

    Robot 1
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park

    PF_Start(1)
    PF_Start(2)

    Xqt rbt1
Fend

Function rbt1
    Do
        Call RobotPickPlace(1)
        Call RobotPickPlace(2)
    Loop
Fend

Function RobotPickPlace(PartID As Integer)
    Integer partsToPickMembit

    Select PartID
        Case 1
            partsToPickMembit = IONumber("PartsToPick1")
        Case 2
            partsToPickMembit = IONumber("PartsToPick2")
    Send

    Wait MemSw(partsToPickMembit) = On
    If PF_QueLen(PartID) > 0 Then

         Do
            Pick = PF_QueGet(PartID)
            PF_QueRemove PartID
            Jump pick
            On Gripper; Wait 0.2
            If PF_QueLen(PartID) = 0 Then
                Jump Place ! D90; MemOff partsToPickMembit !
                Off Gripper; Wait 0.2
                Exit Do
            Else
                Jump Place
                Off Gripper; Wait 0.2
            EndIf

            If PF_IsStopRequested(PartID) = True Then
                MemOff partsToPickMembit
                Exit Do
            EndIf
        Loop
    Else
        MemOff partsToPickMembit
    EndIf
Fend
```

PartFeeding.prg

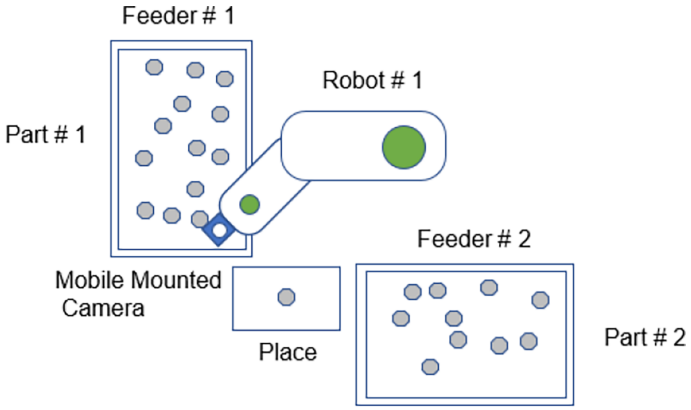
```
Function PF_Robot(PartID As Integer) As Integer

    Select PartID
        Case 1
            MemOn PartsToPick1
            Wait MemSw(PartsToPick1) = Off
        Case 2
            MemOn PartsToPick2
            Wait MemSw(PartsToPick2) = Off
    Send

    PF_Robot = PF_CALLBACK_SUCCESS
Fend
```

3.9.1.5 程序示例 1.5

示例类型：
每台送料器1台机器人；每2台送料器1种部件 - 移动支架安装相机

构成

- 机器人数量：1
- 送料器数量：2
- 送料器上的部件类型数量：1
- 配置位置数量：1
- 相机的方向：各送料器使用机器人的移动支架安装相机。
  （本例未使用固定向下相机。）



描述
机器人将移动支架安装相机移动至送料器#1上拍摄图像。机器人从送料器#1中拾取各部件，将其放置到箱中。该运作持续到从送料器移动完所有的“表面”部件为止。机器人将移动支架安装相机移动至送料器#2上拍摄图像。机器人从送料器#2中拾取朝向正确的部件，将其放置到箱中。

要点

使用装有多个带背光灯送料器的移动相机时，其他视觉序列用于与各送料器关联的各部件。

样本代码
Main.prg

```
Function main
    MemOff PartsToPick1; MemOff PartsToPick2
```

```
    MemOff mobileCamBefore1; MemOff mobileCamAfter1
    MemOff mobileCamBefore2; MemOff mobileCamAfter2
    MemOff mobileCamInPos1; MemOff mobileCamInPos2

    Robot 1
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park

    PF_Start(1)
    PF_Start(2)
    Xqt rbt1
Fend

Function rbt1
    Do
        Wait MemSw(mobileCamBefore1) = On
        Jump MobileCamShotFeeder1; MemOn mobileCamInPos1
        Wait MemSw(mobileCamAfter1) = On
        MemOff mobileCamInPos1

        Call RobotPickPlace(1)

        Wait MemSw(mobileCamBefore2) = On
        Jump MobileCamShotFeeder2; MemOn mobileCamInPos2
        Wait MemSw(mobileCamAfter2) = On
        MemOff mobileCamInPos2

        Call RobotPickPlace(2)
    Loop
Fend

Function RobotPickPlace(PartID As Integer)
    Integer partsToPickMembit, partCnt, gripperOutput, toolNum

    Select PartID
        Case 1
            partsToPickMembit = IONumber("PartsToPick1")
        Case 2
            partsToPickMembit = IONumber("PartsToPick2")
    Send

    Wait MemSw(partsToPickMembit) = On
    Do While PF_QueLen(PartID) > 0
        P0 = PF_QueGet(PartID)
        Jump P0
        On Gripper; Wait 0.2
        Jump Place
        Off Gripper; Wait 0.2
        PF_QueRemove PartID
        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf
    Loop
    MemOff partsToPickMembit
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer

    Select PartID
        Case 1
```

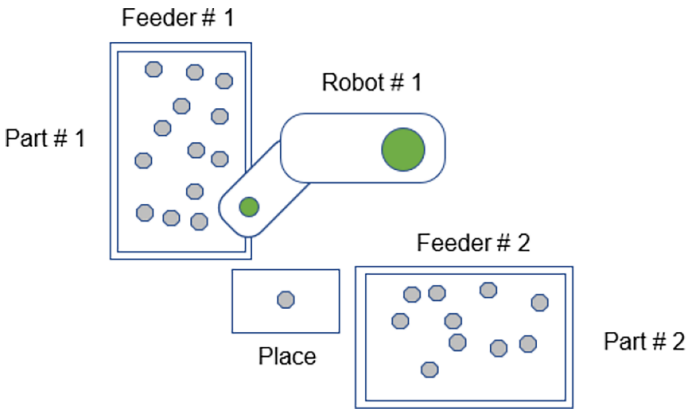
```
            MemOn PartsToPick1
            Wait MemSw(PartsToPick1) = Off
        Case 2
            MemOn PartsToPick2
            Wait MemSw(PartsToPick2) = Off
    Send

    PF_Robot = PF_CALLBACK_SUCCESS
Fend

Function PF_MobileCam(PartID As Integer, Action As Integer) As Integer
    Integer mobileCamBeforeMembit, mobileCamAfterMembit, mobileCamInPosMembit

    Select PartID
        Case 1
            mobileCamBeforeMembit = IONumber("mobileCamBefore1")
            mobileCamAfterMembit = IONumber("mobileCamAfter1")
            mobileCamInPosMembit = IONumber("mobileCamInPos1")
        Case 2
            mobileCamBeforeMembit = IONumber("mobileCamBefore2")
            mobileCamAfterMembit = IONumber("mobileCamAfter2")
            mobileCamInPosMembit = IONumber("mobileCamInPos2")
    Send

    Select Action
        Case PF_MOBILECAM_BEFORE
            ' Request for robot move to camera position
            MemOff mobileCamAfterMembit
            MemOn mobileCamBeforeMembit
            Wait MemSw(mobileCamInPosMembit) = On
        Case PF_MOBILECAM_AFTER
            ' Request for robot move after part vision acquisition
            MemOff mobileCamBeforeMembit
            MemOn mobileCamAfterMembit
            Wait MemSw(mobileCamInPosMembit) = Off
    Send

    PF_MobileCam = PF_CALLBACK_SUCCESS
Fend
```

3.9.1.6 程序示例 1.6

示例类型：
指定要拾取部件的数量。

构成

- 机器人数量：1
- 送料器数量：2
- 各送料器上的部件类型数量：1
- 配置位置数量：1
- 相机的方向：在各送料器上配置固定向下相机



描述

机器人从送料器#1中拾取4个部件#1，将其分别放置。此后，机器人从送料器#2中拾取5个部件#2，将其放置。相机与送料器并行运作。
本例说明机器人从各送料器拾取特定数量部件的方法。为样本代码时，通过将numToPick参数设为“ALL_AVAILABLE”，可拾取所有部件。
拾取所需量的部件后，某个送料器中残留有“表面”部件时，送料器不振动。

样本代码

Main.prg

```
###define ALL_AVAILABLE -1

Function main
    MemOff PartsToPick1; MemOff PartsToPick2
    Off Gripper

    Robot 1
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park

    PF_Start(1)
    PF_Start(2)

    Xqt rbt1
Fend

Function rbt1
    Do
        Call RobotPickPlace(1, 4) 'part 1...pick & place 4 times
        Call RobotPickPlace(2, 5) 'part 2...pick & place 5 times
    Loop
Fend

Function RobotPickPlace(PartID As Integer, numToPick As Integer)
    Integer partsToPickMembit, partCnt

    Select PartID
        Case 1
            partsToPickMembit = IONumber("PartsToPick1")
        Case 2
            partsToPickMembit = IONumber("PartsToPick2")
    Send

    partCnt = 0
    Do
        Wait MemSw(partsToPickMembit) = On
```

```
        If PF_QueLen(PartID) > 0 Then
            Pick = PF_QueGet(PartID)
            PF_QueRemove PartID
            Jump pick
            On Gripper; Wait 0.2
            partCnt = partCnt + 1
            If PF_QueLen(PartID) = 0 Then
                Jump Place ! D90; MemOff partsToPickMembit !
                Off Gripper; Wait 0.2
                If (partCnt = numToPick) Or (numToPick = ALL_AVAILABLE) Then
                    Exit Do
                EndIf
            Else
                Jump Place
                Off Gripper; Wait 0.2
                If (partCnt = numToPick) Then
                     Exit Do
                EndIf
             EndIf
         Else
             MemOff partsToPickMembit
         EndIf
         If PF_IsStopRequested(PartID) = True Then
            MemOff partsToPickMembit
            Exit Do
         EndIf
    Loop
Fend
```

PartFeeding.prg

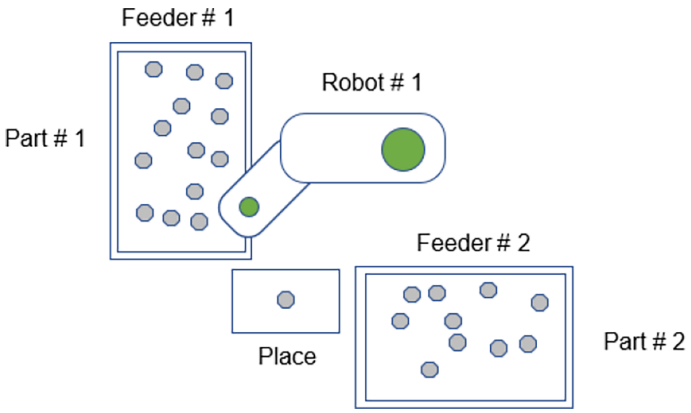
```
Function PF_Robot(PartID As Integer) As Integer

    Select PartID
        Case 1
            MemOn PartsToPick1
            Wait MemSw(PartsToPick1) = Off
        Case 2
            MemOn PartsToPick2
            Wait MemSw(PartsToPick2) = Off
     Send

    PF_Robot = PF_CALLBACK_SUCCESS
Fend
```

3.9.1.7 程序示例 1.7

示例类型：
确认机器人编号与用于部件的机器人编号一致。

构成

- 送料器数量：大于等于1
- 各送料器上的部件类型数量：1
- 相机的方向：在各送料器上配置固定向下相机



描述

使用多台送料器时，通常在其他任务内记述机器人的运作，而非在PF_Robot回调函数内记述。也就是说，PF_Robot回调函数仅用于将部件的位置（点）放入部件坐标队列中。

在进行机器人的运动任务时，使用部件坐标队列中的这些点。采用该方法构成代码时，请在机器人的运动任务中，确认当前机器人编号与部件选择的机器人编号是否一致。

通过该确认，可防止机器人移动至错误的点或发生碰撞。

样本代码

Main.prg

```
Function RobotPickPlace(PartID As Integer, numToPick As Integer)

    If PF_Info(PartID, PF_INFO_ID_ROBOT_NO) <> Robot Then
        Print "Robot does not match the robot # for the current part"
        Quit All
    EndIf

    'Robot Motion Code
Fend
```

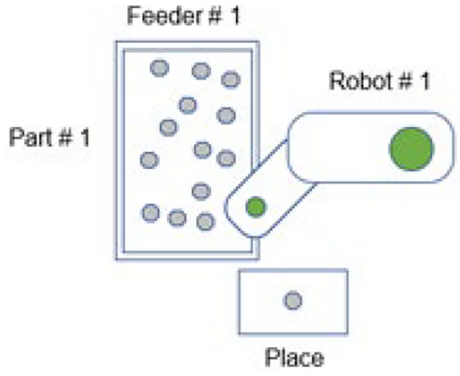
3.9.1.8 程序示例 1.8

示例类型：

拾取之后，获取新图像并加载队列。

构成

- 机器人数量：1
- 送料器数量：1
- 部件类型数量：1
- 配置位置数量：1
- 相机的方向：固定向下相机



描述

机器人从零件送料器中拾取部件#1，放置到固定夹具上。进行拾取&放置操作后，获取新图像并重新加载队列。
在本例中，拾取期间周围的部件移动令人担忧。PF_Robot回调函数的返回值“PF_CALLBACK_RESTART”用于对所有部件重新强制执行视觉处理，并重新加载所有的部件队列。
这种方法没有效率可言，但如果按循环获取图像，在提高性能精度的特定状况下，“PF_CALLBACK_RESTART”有用。与机器人运作并行发生视觉处理和送料器振动。

样本代码

Main.prg

```
Function main
    MemOff PartsToPick
    Off Gripper

    Robot 1
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park

    PF_Start(1)

    Xqt RobotPickPlace
Fend

Function RobotPickPlace
    Do
        Wait MemSw(PartsToPick) = On
        If PF_QueLen(1) > 0 Then
            Pick = PF_QueGet(1)
            PF_QueRemove 1
            Jump pick
            On Gripper; Wait 0.2
            Jump Place ! D90; MemOff PartsToPick !
            Off Gripper; Wait 0.2
        Else
            MemOff PartsToPick
        EndIf
    Loop
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    MemOn PartsToPick
    Wait MemSw(PartsToPick) = Off

    PF_Robot = PF_CALLBACK_RESTART
Fend
```

3.9.1.9 程序示例 1.9

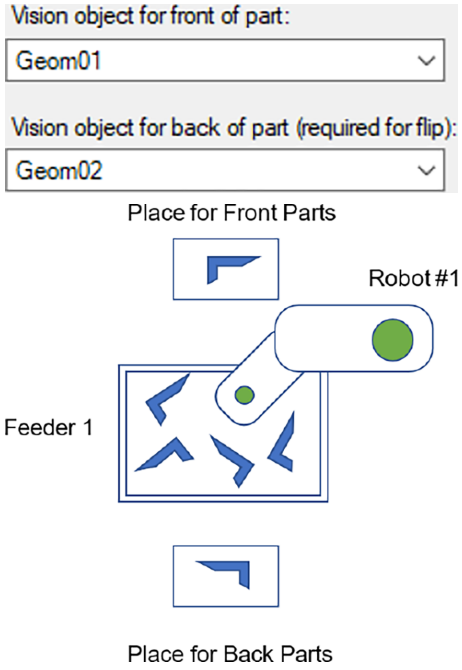
示例类型：
按表里对部件进行排序

构成

- 机器人数量：1
- 送料器数量：1

- 送料器上的部件类型数量：1
- 配置位置数量：2（1个为“表面”方向，另1个为“背面”方向）
- 相机的方向：送料器#1上的固定向下相机
- 部件1常规：

- 部件1视觉：



描述

在该应用中，机器人根据部件的“表面”与“背面”的方向，对部件进行排序。表面方向的部件被配置到带有“PlaceFront”标签的点上；背面方向的部件被配置到带有“PlaceBack”标签的其他点上。
在该应用中，拾取顺序并不重要。机器人会尽可能多地拾取&放置表面方向的部件，尽可能多地拾取&放置背面方向的部件。
在未勾选“需要翻转”复选框的情况下，选择“检测表面部件的对象”与“检测背面部件的对象”的视觉对象时，系统会同时将表里方向的部件加载至坐标队列中，据此设定Part Orientation的值。
“需要翻转”设定会向系统传递要将部件置于特定方向的信息。如果未勾选“需要翻转”复选框，则会向系统传递要将部件置于表里两个方向的信息。

在这种情况下，表面方向部件的方向数据（部件坐标队列内），会被自动设为常数“PF_PARTORIENT_FRONT”。
背面方向部件的数据，会被自动设为常数“PF_PARTORIENT_BACK”。

样本代码

Main.prg

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park
    PF_Start 1
Fend
```

PartFeeding.prg

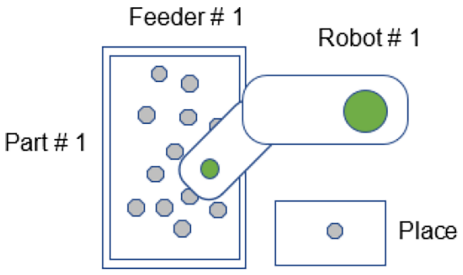
```
Function PF_Robot(PartID As Integer) As Integer
    Do While PF_QueLen(PartID) > 0
        P0 = PF_QueGet(PartID)
        Jump P0
        On Gripper; Wait 0.2
        If PF_QuePartOrient(PartID) = PF_PARTORIENT_FRONT Then
            Jump PlaceFront
        ElseIf PF_QuePartOrient(PartID) = PF_PARTORIENT_BACK Then
            Jump PlaceBack
        EndIf
        Off Gripper; Wait 0.2
        PF_QueRemove PartID
        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf
    Loop
    PF_Robot = PF_CALLBACK_SUCCESS

Fend
```

3.9.1.10 程序示例 1.10

示例类型：
使用PF_Vision回调函数

构成

- 机器人数量：1
- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：1
- 相机的方向：送料器#1上的固定向下相机



描述
本例所示为使用PF_Vision回调函数获取图像，并加载包括视觉结果在内的部件坐标队列的方法。要使用该功能时，在Epson RC+ 8.0菜单 - [工具] - [上料] - [视觉]中，选择[通过PF_Vision callback进行视觉处理]。

样本代码
Main.prg

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park
    PF_Start 1
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    Do While PF_QueLen(PartID) > 0
        P0 = PF_QueGet(PartID)
        Jump P0
        On Gripper; Wait 0.2
        Jump Place
        Off Gripper; Wait 0.2
        PF_QueRemove PartID
        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf
    Loop
    PF_Robot = PF_CALLBACK_SUCCESS

Fend

Function PF_Vision(PartID As Integer, ByRef numBack As Integer) As Integer
    Boolean found
    Integer i, numFront
    Real RB_X, RB_Y, RB_U, RB_Z

    ' Pick Z coordinate
    RB_Z = -132.0

    ' Initialize coordinates queue
    PF_QueRemove PartID, All

    PF_Backlight 1, On

    ' Detect the parts
    VRun UsrVisionSeq
    PF_Backlight 1, Off

    VGet UsrVisionSeq.Geom01.NumberFound, numFront 'Front Parts
    VGet UsrVisionSeq.Geom02.NumberFound, numBack 'Back Parts
    If numFront <> 0 Then
        For i = 1 To numFront
            VGet UsrVisionSeq.Geom01.RobotXYU(i), found, RB_X, RB_Y, RB_U
            If found Then
                PF_QueAdd PartID, XY(RB_X, RB_Y, RB_Z, RB_U)
            EndIf
        Next
    EndIf

    PF_Vision = PF_CALLBACK_SUCCESS

Fend
```

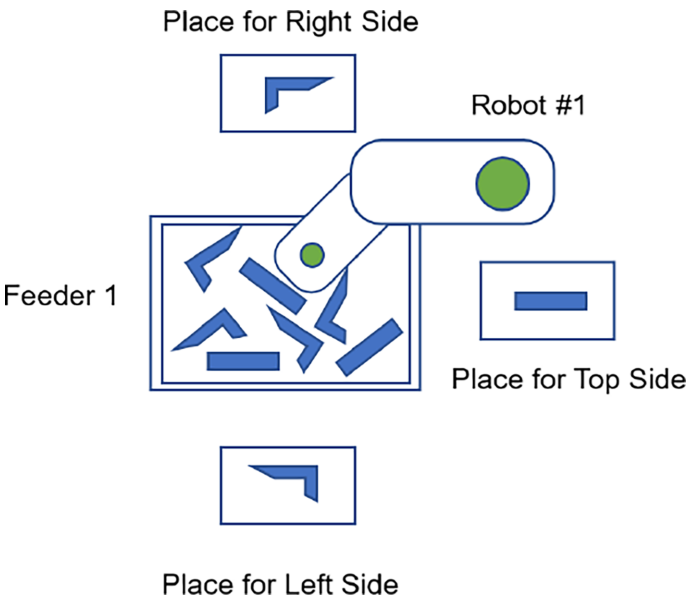
3.9.1.11 程序示例 1.11

示例类型：
选择多面部件

构成

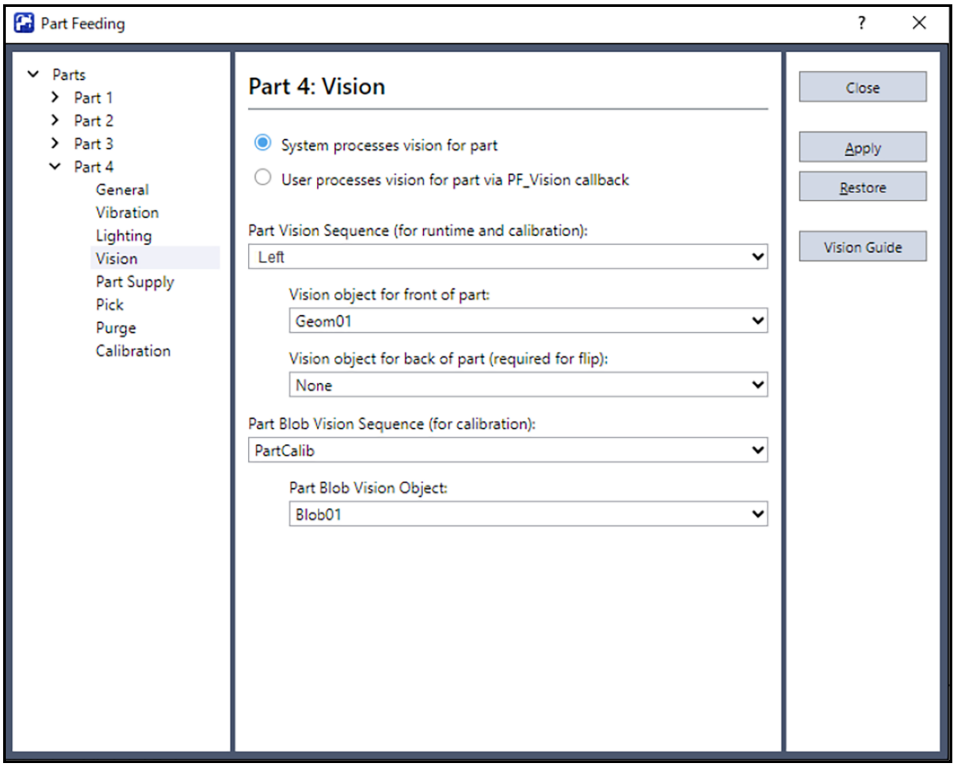
- 机器人数量：1
- 送料器数量：1
- 送料器上的部件类型数量：1
- 部件姿势数量：3
- 配置位置数量：3（部件各方向都有1个配置位置）

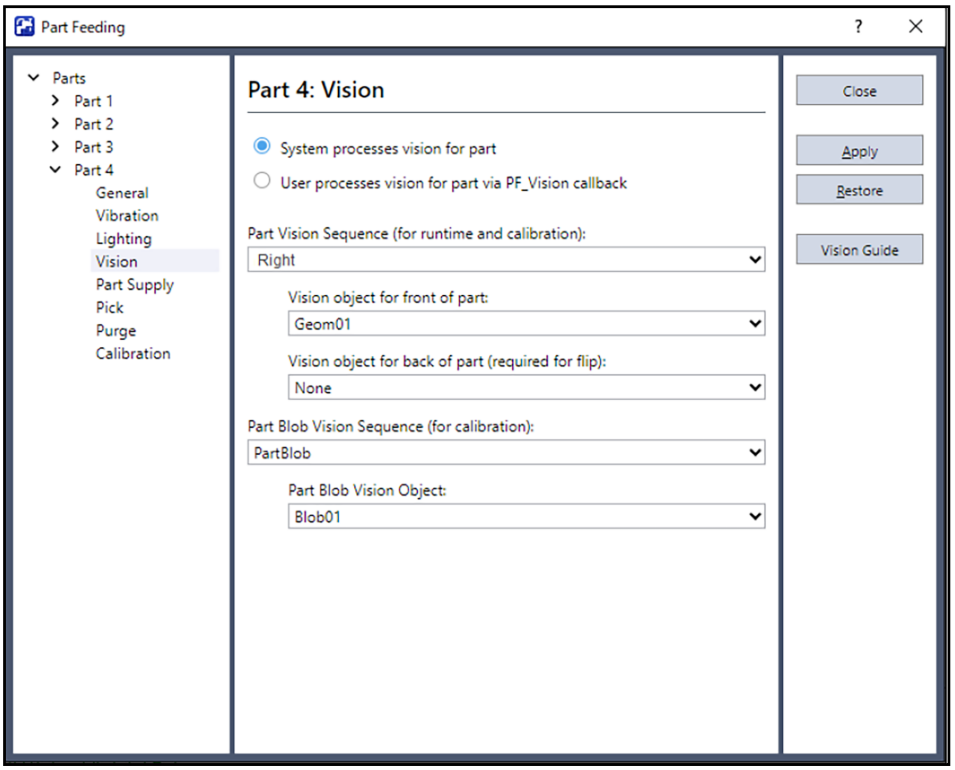
- 相机的方向：送料器#1上的固定向下相机



描述

本例中的送料器上有1种物理部件。部件包括3种姿势（右、左、上）。机器人可拾取3种姿势部件的一种。
实际上是1个部件，但如果从视觉系统的角度来看，则为各姿势分别不同的部件。上料时，支持用同一送料器同时执行的4个部件。因此，创建各姿势不同的部件。创建3个部件视觉序列，分别命名为“Left”、“Right”与“Top”。
为各视觉序列时，使用Geometric对象将部件配置为特定姿势。在[上料]对话框中创建部件1、2与3共3个部件。
部件1时，将部件配置为“左侧”姿势。
部件2时，将部件配置为“右侧”姿势。
部件3时，将部件配置为“上侧”姿势。
所有3个部件均未勾选“需要翻转”复选框。3个部件均使用与“PartBlob”同名的部件Blob视觉序列。


Part Feeding
Parts
Part 1
Part 2
Part 3
Part 4
General
Vibration
Lighting
Vision
Part Supply
Pick
Purge
Calibration
Part 4: Vision
System processes vision for part
User processes vision for part via PF_Vision callback
Part Vision Sequence (for runtime and calibration):
Left
Vision object for front of part:
Geom01
Vision object for back of part (required for flip):
None
Part Blob Vision Sequence (for calibration):
PartCalib
Part Blob Vision Object:
Blob01
Close
Apply
Restore
Vision Guide


Part Feeding
Parts
Part 1
Part 2
Part 3
Part 4
General
Vibration
Lighting
Vision
Part Supply
Pick
Purge
Calibration
Part 4: Vision
System processes vision for part
User processes vision for part via PF_Vision callback
Part Vision Sequence (for runtime and calibration):
Right
Vision object for front of part:
Geom01
Vision object for back of part (required for flip):
None
Part Blob Vision Sequence (for calibration):
PartBlob
Part Blob Vision Object:
Blob01
Close
Apply
Restore
Vision Guide

对各部件的拾取Z进行示教（拾取高度可能因部件各姿势而异）。左侧部件被配置到带有“PlaceLeft”标签的点上。右侧部件被配置到带有“PlaceRight”标签的点上。上侧部件被配置到带有“PlaceTop”标签的点上。机器人在左向拾取&放置所有部件，然后在右向拾取&放置所有部件，最后在上向拾取&放置所有部件。

样本代码

Main.prg

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park
    PF_Start 1, 2, 3
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer

    Do While PF_QueLen(PartID) > 0
        P10 = PF_QueGet(PartID)
        Jump P10
        On Vacuum; Wait 0.2
        Select PartID
            Case 1
                Jump PlaceLeft
            Case 2
                Jump PlaceRight
            Case 3
                Jump PlaceTop
        Send
        Off Vacuum; Wait 0.2
        PF_QueRemove PartID
    Loop

    PartID = PartID + 1
    If PartID > 3 Then PartID = 1
    PF_ActivePart PartID
    PF_Robot = PF_CALLBACK_SUCCESS

Fend
```

要点

本例所示为使用多部件功能处理多面部件的简单方法。另1种方法是通过PF_Vision回调函数进行“User to Process Vision”的方法。如果使用PF_QueAdd，则可将用户数据包括在部件坐标中。可将用户数据设为表示部件姿势的数值（1 =左、2 =右、3 =上）。使用PF_QueUserData函数获取方向值，以便将部件配置在正确的位置上。使用该代码时，也需要考虑部件侧面的高度差异。

3.9.1.12 程序示例 1.12

示例类型：

使用PF_Vision回调函数与多搜索功能的部件检测视觉序列

构成

- 机器人数量：1
- 送料器数量：1

- 送料器上的部件类型数量：1
- 配置位置数量：1
- 相机的方向：送料器#1上的固定向下相机

描述

凭借部件检测视觉序列难以进行处理时（比如，为检测部件而需要多个视觉序列，或需要特殊的外部照明控制时），使用PF_Vision回调函数。

本例所示为使用PF_Vision回调函数获取图像，并将视觉结果加载至部件坐标队列中的方法。要使用该功能时，选择Epson RC+ 8.0菜单 - [工具] - [上料] - [视觉]中的[通过PF_Vision callback进行视觉处理]。

在本例中使用了视觉处理的多搜索功能。多搜索功能用于搜索对象的CenterPointObject属性或Frame对象的各结果。使用多搜索功能时，找到的结果可能会不按预期顺序排列。PF_Vision用于反复处理所有结果，如下所述为对找到结果的处理方法。

- 使用CenterPointObject进行多搜索的示例

1. 创建用于搜索多个部件的Blob等对象。
2. 使用Blob的CenterPointObject，创建Polar等其他对象。
3. 将CenterPntObjResult属性设为ALL。
4. 执行序列。按找到的Blob结果，显示Polar对象的实例。

- 使用Frame进行多搜索的示例

1. 创建用于搜索多个部件的Blob等对象。
2. 创建Frame对象，将OriginPoint属性设为Blob。
3. 将OriginPntObjResult属性设为All。
4. 使用Frame创建Polar等其他对象。
5. 将FrameResult属性设为All。
6. 执行序列。按找到的Blob结果，显示Frame对象的实例；按Frame结果，显示Polar对象的实例。

样本代码

Main.prg

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park
    PF_Start 1
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer

    Do While PF_QueLen(PartID) > 0
        P10 = PF_QueGet(PartID)

        Select PF_QuePartOrient(PartID)
            Case PF_PARTORIENT_FRONT ' Front
                Tool 1 ' Tool to pick Front
                ' Pick
                Jump P10 ! D90; On Vacuum !
                Wait 0.1
                ' Place
                Jump FrontPlace
                Off Vacuum
                Wait 0.1
```

```
            Case PF_PARTORIENT_BACK ' Back
                Tool 2 ' Tool to pick Back
                ' Pick
                Jump P10 ! D90; On Vacuum !
                Wait 0.1
                ' Place
                Jump BackPlace
                Off Vacuum
                Wait 0.1
        Send

        PF_QueRemove PartID

        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf

    Loop

    PF_Robot = PF_CALLBACK_SUCCESS

Fend


Function PF_Vision(PartID As Integer, ByRef numBack As Integer) As Integer

    Integer i, numFront, numBack, numResults, count
    Boolean found
    Real x, y, z, u

    z = -170 ' Set the pick Z coordinate to the desired height for your application

    PF_QueRemove 1, All

    PF_Backlight 1, On
    VRun FindPart

    VGet FindPart.Front.NumberFound, numFront
    VGet FindPart.Front.NumberOfResults, numResults
    count = 0
    For i = 1 To numResults
        VGet FindPart.Front.RobotXYU(i), found, x, y, u
        If found Then
            count = count + 1
            P999 = XY(x, y, z, u) /R
            PF_QueAdd 1, P999, PF_PARTORIENT_FRONT
        EndIf
        If count = numFront Then
            Exit For
        EndIf
    Next

    VGet FindPart.Back.NumberFound, numBack
    VGet FindPart.Back.NumberOfResults, numResults
    count = 0
    For i = 1 To numResults
        VGet FindPart.Back.RobotXYU(i), found, x, y, u
        If found Then
            count = count + 1
            P999 = XY(x, y, z, u) /R
            PF_QueAdd 1, P999, PF_PARTORIENT_BACK
        EndIf
        If count = numBack Then
            Exit For
        EndIf
```

```
    Next

    PF_Backlight 1, Off
    PF_Vision = PF_CALLBACK_SUCCESS

Fend
```

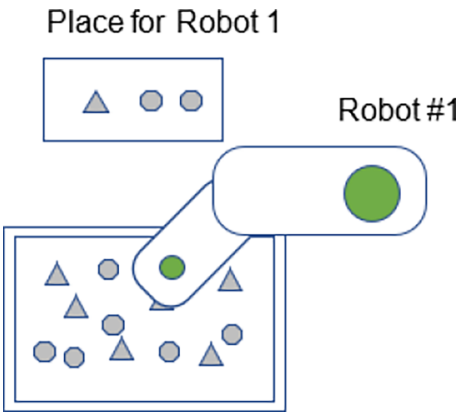
3.9.2 1台机器人 - 多部件

3.9.2.1 程序示例 2.1

示例类型：
通过1台机器人与多部件-机器人运作，进行视觉系统与振动的并行处理

构成

- 机器人数量：1
- 送料器数量：1
- 送料器上的部件类型数量：2
- 配置位置数量：1
- 相机的方向：固定向下相机



描述
包括2种部件（物理方面存在差异）。机器人连续拾取&放置2个部件#1，然后拾取&放置1个部件#2。这是通过交互执行“PF_ActivePart”来实现的。在该应用中，拾取顺序非常重要（例：装入部件时）。如果配置好最后的部件，“PF_ActivePart”则会被更改，供给预期的部件，并向系统发送振动信号，最后根据需要获取图像。这是通过并行进行机器人运作（达到“place”前的移动路径的30%位置时）来实现的。

样本代码
Main.prg

```
Function Main
    Integer numToPick1, numToPick2, i

    Robot 1
    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Park

    MemOff PartsToPick1
    MemOff PartsToPick2
    numToPick1 = 2
```

```
    numToPick2 = 1

    PF_Start 1, 2

    Do
        i = 0
        Do
            Wait MemSw(PartsToPick1) = On
            P0 = PF_QueGet(1)
            PF_QueRemove (1)
            Jump P0 /R
            On Gripper
            Wait 0.25
            i = i + 1
            If i < numToPick1 And PF_QueLen(1) > 0 Then
                Jump Place
            Else
                'Last part or no more parts available to pick
                If i = numToPick1 Then
                    PF_ActivePart 2
                EndIf
                Jump Place ! D30; MemOff PartsToPick1 !
            EndIf
            Off Gripper
            Wait 0.25
        Loop Until i = numToPick1
        i = 0
        Do
            Wait MemSw(PartsToPick2) = On
            P0 = PF_QueGet(2)
            PF_QueRemove (2)
            Jump P0 /R
            On Gripper
            Wait 0.25
            i = i + 1
            If i < numToPick2 And PF_QueLen(2) > 0 Then
                Jump Place
            Else
                'Last part or no more parts available to pick
                If i = numToPick2 Then
                    PF_ActivePart 1
                EndIf
                Jump Place ! D30; MemOff PartsToPick2 !
            EndIf
            Off Gripper
            Wait 0.25
        Loop Until i = numToPick2
    Loop
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    Select PartID
        Case 1
            MemOn PartsToPick1
            Wait MemSw(PartsToPick1) = Off
        Case 2
            MemOn PartsToPick2
            Wait MemSw(PartsToPick2) = Off
    Send
```

```
    PF_Robot = PF_CALLBACK_SUCCESS
Fend
```

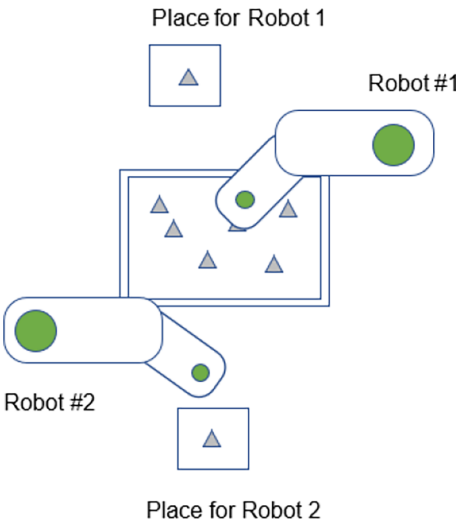
3.9.3 2台机器人 - 1种部件

3.9.3.1 程序示例 3.1

示例类型：
2台机器人与1种物理部件- PF_Robot回调函数的运动-按特定顺序拾取

构成

- 机器人数量：2
- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：2
- 相机的方向：固定向下相机



描述
包括2台机器人与1台送料器。物理部件仅为1种。各机器人都有独自的相机校准，因此包括2种（机器人1的部件1与机器人2的部件2）逻辑部件。
机器人从送料器中依次进行拾取运作。在该应用中，拾取顺序非常重要。由“PF_ActivePart”执行交互拾取顺序。
在PF_Robot回调函数内执行机器人的运作。
在本例中没有送料器与机器人运作的并行处理。代码虽然单纯，但没有效率。各机器人包括带有“park”标签的点与带有“place”标签的点。本例的重要概念为PF_Robot回调函数的返回值“PF_CALLBACK_RESTART_ACTIVEPART”。
不可能利用该返回值在双方的部件队列中调整部件，但可确保多台机器人使用同一送料器。利用返回值强制获取PF_ActivePart的新图像，并且加载PF_ActivePart的队列。

样本代码
Main.prg

```
Function Main
    Robot 1
    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Park
    Robot 2
```

```
    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Park

    PF_Start 1, 2
Fend
```

PartFeeding.prg

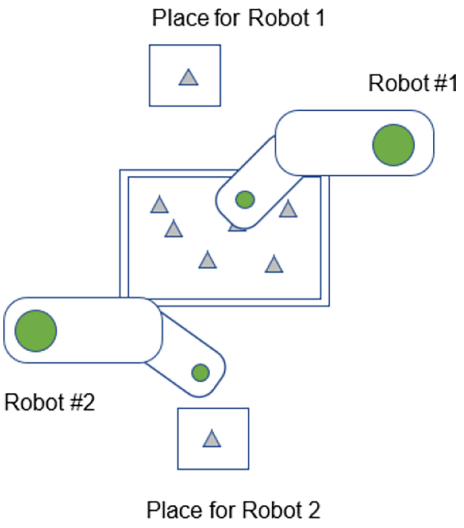
```
Function PF_Robot(PartID As Integer) As Integer
    If PF_QueLen(PartID) > 0 Then
        Select PartID
            Case 1
                Robot 1
                P0 = PF_QueGet(1)
                PF_QueRemove (1)
                Jump P0 /R
                On rbt1Gripper
                Wait 0.25
                Jump Place
                Off rbt1Gripper
                Wait 0.25
                PF_ActivePart 2
            Case 2
                Robot 2
                P0 = PF_QueGet(2)
                PF_QueRemove (2)
                Jump P0 /L
                On rbt2Gripper
                Wait 0.25
                Jump Place
                Off rbt2Gripper
                Wait 0.25
                PF_ActivePart 1
        Send
    EndIf

    PF_Robot = PF_CALLBACK_RESTART_ACTIVEPART

Fend
```

3.9.3.2 程序示例 3.2

示例类型：
2台机器人与1种物理部件 - 其他任务的运动- 拾取顺序没有关系 - 按特定的顺序拾取

构成

- 机器人数量：2
- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：2
- 相机的方向：固定向下相机



描述

包括2台机器人与1台送料器。物理部件仅为1种。各机器人都有独自的相机校准，因此包括2种（机器人1的部件1与机器人2的部件2）逻辑部件。拾取顺序没有关系-先到顺序。

本例中的差异在于按循环获取各部件的视觉。这对拾取期间周围的部件可能移动的情况有用。

PF_Robot回调函数的返回值“PF_CALLBACK_RESTART”用于对所有部件重新强制执行视觉处理，并重新加载所有的部件队列。

这种方法没有效率可言，但“PF_CALLBACK_RESTART”在特定状况下有用。

样本代码

Main.prg

```
Function Main
    Robot 1
    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Park
    Robot 2
    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Park
    MemOff PartsToPick
    PF_Start 1, 2
    Xqt Robot1PickPlace
    Xqt Robot2PickPlace
Fend

Function Robot1PickPlace
    Robot 1

    Do
        PF_AccessFeeder (1)
        Wait MemSw(PartsToPick) = On
        If PF_QueLen(1) > 0 Then
            P0 = PF_QueGet(1)
            PF_QueRemove (1)
            Jump P0 /R
            On 5
             Wait 0.5
            Jump Place ! D30; MemOff PartsToPick; PF_ReleaseFeeder 1 !
            Off 5
            Wait 0.25
```

```
        Else
            MemOff PartsToPick; PF_ReleaseFeeder 1
        EndIf
    Loop
Fend

Function Robot2PickPlace
    Robot 2

    Do
        PF_AccessFeeder (1)
        Wait MemSw(PartsToPick) = On
        If PF_QueLen(2) > 0 Then
            P0 = PF_QueGet(2)
            PF_QueRemove (2)
            Jump P0 /L
            On 2
            Wait 0.5
            Jump Place ! D30; MemOff PartsToPick; PF_ReleaseFeeder 1 !
            Off 2
            Wait 0.25
        Else
            MemOff PartsToPick; PF_ReleaseFeeder 1
        EndIf
    Loop
Fend
```

PartFeeding.prg

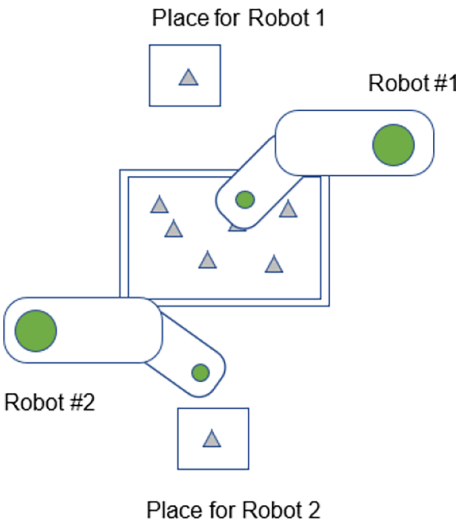
```
Function PF_Robot(PartID As Integer) As Integer
    MemOn PartsToPick
    Wait MemSw(PartsToPick) = Off

    PF_Robot = PF_CALLBACK_RESTART 'Force vision and vibration to refresh
Fend
```

3.9.3.3 程序示例 3.3

示例类型：
2台机器人与1种物理部件 - 其他任务的运动 -先到顺序-被模拟进程的延迟

构成

- 机器人数量：2
- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：2
- 相机的方向：固定向下相机



描述

包括2台机器人与1台送料器。物理部件仅为1种。各机器人都有独自的相机校准，因此包括2种（机器人1的部件1与机器人2的部件2）逻辑部件。在本例中，可调整各机器人的进程时间（按随机待机时间被模拟）。
各机器人从送料机中拾取部件后，因执行其他操作　而进入忙碌状态。
机器人从送料器中拾取部件后，结束其他操作，并在完成从送料器中拾取其他部件的准备时，使用存储器位“Rbt1Complete”与“Rbt2Complete”发送信号。PF_Robot回调函数用于在预期部件（PF_ActivePart）与当前部件不同时（也就是其他机器人拾取部件的情况下），返回“PF_CALLBACK_RESTART_ACTIVEPART”的值。这样可防止机器人队列内的点重复。
获取PF_ActivePart的新图像，并且加载PF_ActivePart的队列。但下一部件与当前部件相同时（也就是同一机器人从送料器中拾取的情况下），PF_Robot回调函数的返回值为“PF_CALLBACK_SUCCESS”。PF_AccessFeeder与PF_ReleaseFeeder用于避免机器人访问送料器时发生碰撞。

样本代码

Main.prg

```
Function Main
    Robot 1
    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Place
    Robot 2
    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Place
    MemOff PartsToPick1
    MemOff PartsToPick2

    PF_Start 1, 2
    Xqt Robot1PickPlace
    Xqt Robot2PickPlace
Fend

Function Robot1PickPlace
    Integer randomTime

    Robot 1
    MemOn Rbt1Complete

    Do
        Wait MemSw(PartsToPick1) = On
```

```
        PF_AccessFeeder (1)
        MemOff Rbt1Complete
        P0 = PF_QueGet(1)
        PF_QueRemove (1)
        Jump P0 /R
        On rbt1Gripper
        Wait 0.25
        Jump Place ! D30; MemOff PartsToPick1; PF_ReleaseFeeder 1 !
        Off rbt1Gripper
        Wait 0.25
        'Test long process time - robot is doing something else
        Randomize
        randomTime = Int(Rnd(9)) + 1
        Wait randomTime
        MemOn Rbt1Complete
    Loop
Fend

Function Robot2PickPlace
    Integer randomTime

    Robot 2
    MemOn Rbt2Complete

    Do
        Wait MemSw(PartsToPick2) = On
        PF_AccessFeeder (1)
        MemOff Rbt2Complete
        P0 = PF_QueGet(2)
        PF_QueRemove (2)
        Jump P0 /L
        On rbt2Gripper
        Wait 0.25
        Jump Place ! D30; MemOff PartsToPick2; PF_ReleaseFeeder 1 !
        Off rbt2Gripper
        Wait 0.25
        'Test long process time - robot is doing something else
        Randomize
        randomTime = Int(Rnd(9)) + 1
        Wait randomTime
        MemOn Rbt2Complete
    Loop
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    Integer nextPart

    Select PartID
        Case 1
            MemOn PartsToPick1
            Wait MemSw(PartsToPick1) = Off
        Case 2
            MemOn PartsToPick2
            Wait MemSw(PartsToPick2) = Off
    Send

    Wait MemSw(Rbt1Complete) = On Or MemSw(Rbt2Complete) = On
    If MemSw(Rbt1Complete) = On Then
        nextPart = 1
    ElseIf MemSw(Rbt2Complete) = On Then
        nextPart = 2
    EndIf
```

```
    PF_ActivePart nextPart

    If nextPart = PartID Then
        'Same part so no need to re-acquire an image and reload the queue
        PF_Robot = PF_CALLBACK_SUCCESS
    Else
        'Restart from vision -
        'Acquire image and load queue for only the Active Part
        PF_Robot = PF_CALLBACK_RESTART_ACTIVEPART
    EndIf

Fend
```

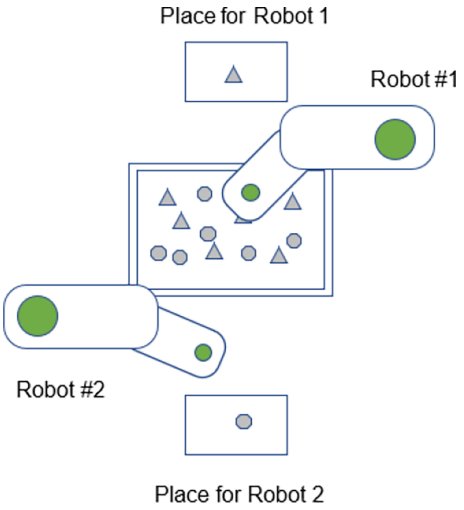
3.9.4 2台机器人 - 多部件

3.9.4.1 程序示例 4.1

示例类型：
2台机器人、1台送料器、多部件 - PF_Robot callback的运动 - 按特定顺序拾取

构成

- 机器人数量：2
- 送料器数量：1
- 送料器上的部件类型数量：2
- 配置位置数量：2
- 相机的方向：固定向下相机



描述
包括2台机器人与1台送料器。各机器人拾取固有的（物理方面存在差异）部件。机器人从送料器中依次进行拾取运作。在该应用中，拾取顺序非常重要。
由“PF_ActivePart”执行交互拾取顺序。在PF_Robot回调函数内执行机器人的运作。在本例中没有送料器与机器人运作的并行处理。代码虽然单纯，但没有效率。
机器人1拾取并配置部件#1。机器人2拾取并配置部件#2。各机器人包括带有“park”标签的点与带有“place”标签的点。

样本代码
Main.prg

```
Function Main
    Robot 1
    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Park
    Robot 2
    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Park
    PF_Start 1, 2
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    If PF_QueLen(PartID) > 0 Then
        Select PartID
            Case 1
                Robot 1
                P0 = PF_QueGet(1)
                PF_QueRemove (1)
                Jump P0 /R
                On rbt1Gripper
                Wait 0.25
                Jump Place
                Off rbt1Gripper
                Wait 0.25
                PF_ActivePart 2
            Case 2
                Robot 2
                P0 = PF_QueGet(2)
                PF_QueRemove (2)
                Jump P0 /L
                On rbt2Gripper
                Wait 0.25
                Jump Place
                Off rbt2Gripper
                Wait 0.25
                PF_ActivePart 1
        Send

    EndIf

    PF_Robot = PF_CALLBACK_SUCCESS

Fend
```

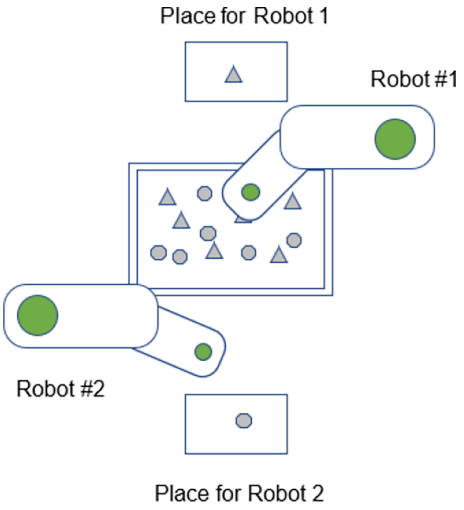
3.9.4.2 程序示例 4.2

示例类型：
2台机器人、1台送料器、多部件 - 其他任务的运动 - 按特定顺序拾取

构成

- 机器人数量：2
- 送料器数量：1
- 送料器上的部件类型数量：2
- 配置位置数量：2

- 相机的方向：固定向下相机



描述

包括2台机器人与1台送料器。各机器人拾取固有的（物理方面存在差异）部件。机器人从送料器中依次进行拾取运作。这是通过交互执行“PF_ActivePart”来实现的。
1台机器人要拾取的部件为零时，另一台机器人可继续从送料器中拾取，直至部件为零。机器人1拾取并配置部件#1。机器人2拾取并配置部件#2。
各机器人包括带有“park”标签的点与带有“place”标签的点。“PF_AccessFeeder”与“PF_ReleaseFeeder”可用于防止两台机器人同时访问送料器。如果某台机器人移动与配置位置之间的距离的30%，另一台机器人则可访问送料器。

样本代码

Main.prg

```
Function Main
    Robot 1
    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Park
    Robot 2
    Motor On
    Power Low
    Speed 50
    Accel 50, 50
    Jump Park
    MemOff PartsToPick1
    MemOff PartsToPick2

    PF_Start 1, 2
    Xqt Robot1PickPlace
    Xqt Robot2PickPlace
Fend

Function Robot1PickPlace
    Robot 1
    Do
        Wait MemSw(PartsToPick1) = On
        PF_AccessFeeder 1
        P0 = PF_QueGet(1)
        PF_QueRemove (1)
        Jump P0 /R
        On 5
        Wait 0.5
        Jump Place ! D30; MemOff PartsToPick1; PF_ReleaseFeeder 1 !
```

```
        Off 5
        Wait 0.25
    Loop
Fend

Function Robot2PickPlace
    Robot 2
    Do
        Wait MemSw(PartsToPick2) = On
        PF_AccessFeeder 1
        P0 = PF_QueGet(2)
        PF_QueRemove (2)
        Jump P0 /L
        On 2
        Wait 0.5
        Jump Place ! D30; MemOff PartsToPick2; PF_ReleaseFeeder 1 !
        Off 2
        Wait 0.25
    Loop
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    Select PartID
        Case 1
            If PF_QueLen(1) > 0 Then
                MemOn PartsToPick1
                Wait MemSw(PartsToPick1) = Off
                PF_ActivePart 2
            Else
                PF_ActivePart 1
            EndIf
        Case 2
            If PF_QueLen(2) > 0 Then
                MemOn PartsToPick2
                Wait MemSw(PartsToPick2) = Off
                PF_ActivePart 1
            Else
                PF_ActivePart 2
            EndIf
    Send
    PF_Robot = PF_CALLBACK_SUCCESS
Fend
```

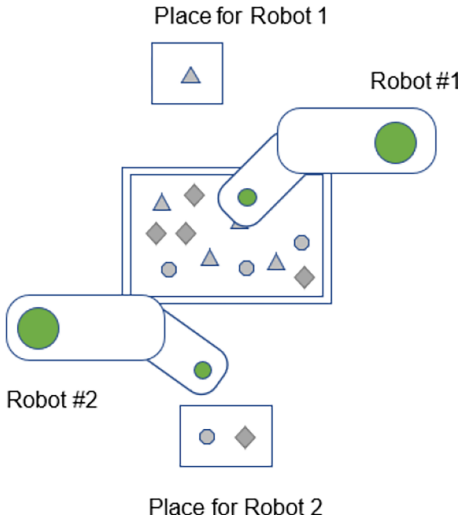
3.9.4.3 程序示例 4.3

示例类型：
2台机器人、1台送料器、多部件 - 其他任务的运动 - 按特定顺序拾取

构成

- 机器人数量：2
- 送料器数量：1
- 送料器上的部件类型数量：3
- 配置位置数量：2

- 相机的方向：固定向下相机



描述

包括2台机器人与1台送料器。各机器人拾取不同的部件。
机器人1拾取&放置部件#1中的一个。
机器人2拾取&放置部件4与部件5中的一个。在该应用中，拾取顺序非常重要。由“PF_ActivePart”执行交互拾取顺序。与送料器的振动并行进行机器人的运作。

样本代码

Main.prg

```
Function Main
    Robot 1
    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Park
    Robot 2
    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Park
    MemOff PartsToPick1
    MemOff PartsToPick4
    MemOff PartsToPick5

    PF_Start 1, 4, 5
    Xqt Robot1PickPlace
    Xqt Robot2PickPlace
Fend

Function Robot1PickPlace
    Robot 1
    Do
        Wait MemSw(PartsToPick1) = On
        PF_AccessFeeder (1)
        P0 = PF_QueGet(1)
        PF_QueRemove (1)
        Jump P0 /R
        On rbt1Gripper; Wait .25
        Jump Place ! D30; MemOff PartsToPick1; PF_ReleaseFeeder 1 !
        Off rbt1Gripper
        Wait 0.25
    Loop
```

```
Fend

Function Robot2PickPlace
    Robot 2
    Do
        Wait MemSw(PartsToPick4) = On
        PF_AccessFeeder (1)
        P0 = PF_QueGet(4)
        PF_QueRemove (4)
        Jump P0 /L
        On rbt2Gripper; Wait .25
        Jump Place ! D30; MemOff PartsToPick4 !
        Off rbt2Gripper; Wait 0.25
        Wait MemSw(PartsToPick5) = On
        P0 = PF_QueGet(5)
        PF_QueRemove (5)
        Jump P0 /L
        On rbt2Gripper; Wait 0.25
        Jump Place ! D30; MemOff PartsToPick5; PF_ReleaseFeeder 1 !
        Off rbt2Gripper; Wait 0.25
    Loop
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    Select PartID
        Case 1
            MemOn PartsToPick1
            Wait MemSw(PartsToPick1) = Off
            PF_ActivePart 4
        Case 4
            MemOn PartsToPick4
            Wait MemSw(PartsToPick4) = Off
            PF_ActivePart 5
        Case 5
            MemOn PartsToPick5
            Wait MemSw(PartsToPick5) = Off
        PF_ActivePart 1
    Send
    PF_Robot = PF_CALLBACK_SUCCESS
Fend
```

3.9.5 通过PF_Feeder回调函数控制振动

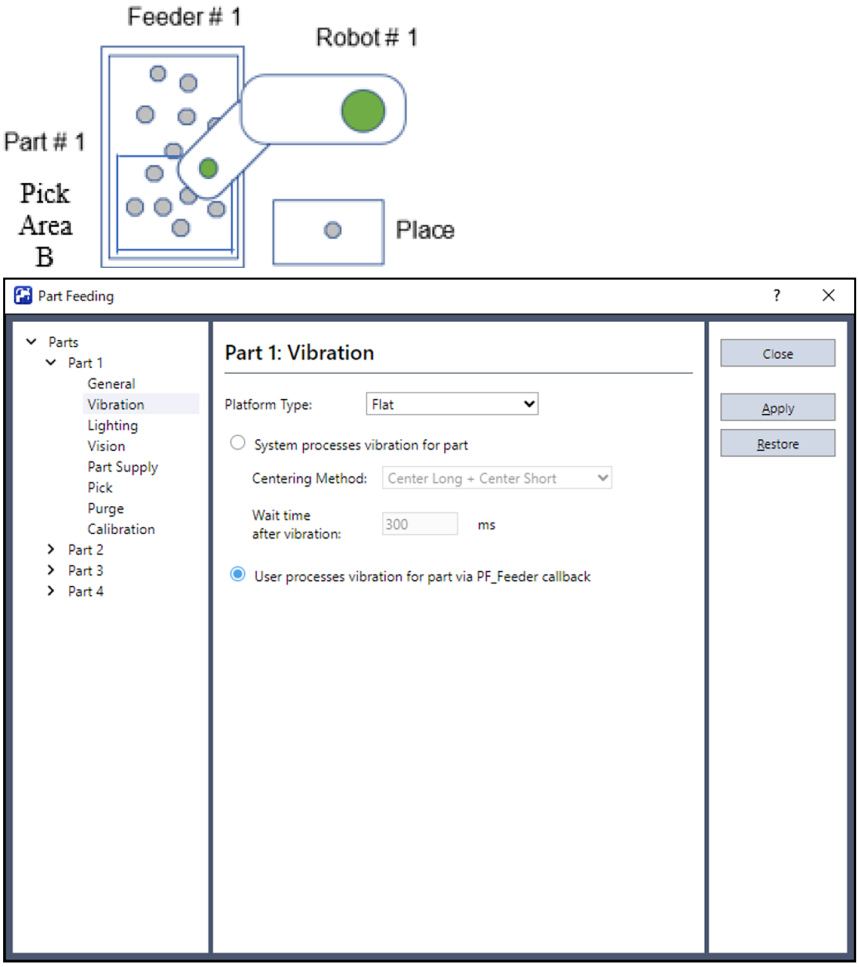
3.9.5.1 程序示例 5.1

示例类型：
平面平台 - 通过PF_Feeder回调函数控制振动

构成

- 机器人数量：1
- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：1
- 平台类型：平面
- 拾取区域：区域B

- 相机的方向：固定向下相机



描述

在本例中使用标准的平面平台。使用平面平台时，通常选择菜单 - [工具] - [上料] - [部件] - [振动]中的[由系统控制振动]按钮。

本例介绍选择[通过PF_Feeder callback控制振动]并独自记述振动处理的方法。在PF_Feeder回调函数内执行用户的振动处理。需要与系统提供的振动处理不同的振动处理时，选择[通过PF_Feeder callback控制振动]。使用自定义平台（孔、长槽、定位槽等）时，需要通过PF_Feeder回调函数自行处理振动。

有关详细信息，请参阅以下内容。

程序示例 5.2

即使选择[通过PF_Feeder callback控制振动]（平面、防粘贴、防滚动的各标准平台），系统也会判断是否对处于各种状况的部件进行了最佳处理。通过自变量“state”将判断结果提供给PF_Feeder回调函数。由“PartFeeding.inc”文件定义各种状态的常数值。

比如，常数“PF_FEEDER_PICKOK”表示机器人可拾取部件。作为其他示例，系统判断翻转部件为最佳处理时，会将常数“PF_FEEDER_FLIP”交接给PF_Feeder回调函数。具体如何处理自变量“state”的值，则委托给用户。

从概念上来讲，用户可使用PF_Feeder回调函数的自变量“state”与适当的送料器控制命令，重新创建系统处理。虽然会重复进行，但平面平台时，通常会选择[由系统控制振动]。也就是说，本例所述为使用PF_Feeder回调函数与送料器控制命令模拟系统处理的方法。

样本代码

Main.prg

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
```

```
    Power Low
    Jump Park
    PF_Start 1
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    Do While PF_QueLen(PartID) > 0
        P0 = PF_QueGet(PartID)
        Jump P0
        On Gripper; Wait 0.2
        Jump Place
        Off Gripper; Wait 0.2
        PF_QueRemove PartID
        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf
    Loop
    PF_Robot = PF_CALLBACK_SUCCESS

Fend

Function PF_Feeder(PartID As Integer, NumFrontParts As Integer, NumBackParts As
Integer, state As Integer) As Integer

    Select state

        ' OK to Pick
        Case PF_FEEDER_PICKOK
             ' Call PF_Robot because there are parts ready to pick
            PF_Feeder = PF_CALLBACK_SUCCESS

         ' Supply more parts
        Case PF_FEEDER_SUPPLY
            PFControlReturnVal = PF_Control(PartID, PF_CONTROL_SUPPLY_FIRST)
            PF_Feeder = PF_CALLBACK_RESTART ' Restart and re-acquire images

        ' Parts are spread out but need to be flipped
         Case PF_FEEDER_FLIP
            PF_Flip PartID
            ' Restart and re-acquire images
            PF_Feeder = PF_CALLBACK_RESTART

        ' Shift parts into pick region
        Case PF_FEEDER_SHIFT
            PF_Shift PartID, PF_SHIFT_FORWARD
            PF_Feeder = PF_CALLBACK_RESTART ' Restart and re-acquire images

        ' Center, Flip and Separate
        Case PF_FEEDER_CENTER_FLIP
            PF_Center PartID, PF_CENTER_LONG_AXIS
            PF_Center PartID, PF_CENTER_SHORT_AXIS
            PF_Flip PartID
            PF_Feeder = PF_CALLBACK_RESTART ' Restart and re-acquire images

        ' Hopper is empty
        Case PF_FEEDER_HOPPER_EMPTY
            PFStatusReturnVal = PF_Status(PartID, PF_STATUS_NOPART)
            PFControlReturnVal = PF_Control(PartID, PF_CONTROL_SUPPLY_FIRST)
            ' Center, Flip and Separate
            PF_Center PartID, PF_CENTER_LONG_AXIS
            PF_Center PartID, PF_CENTER_SHORT_AXIS
            PF_Flip PartID
            PF_Feeder = PF_CALLBACK_RESTART ' Restart and re-acquire images
```

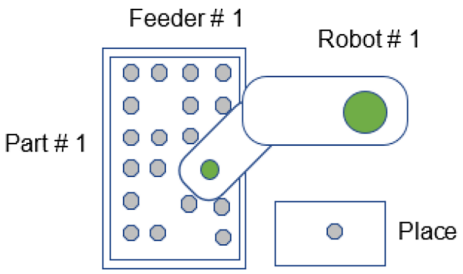
```
        ' Parts have gathered against the platform wall
        Case PF_FEEDER_SHIFT_BACKWARDS
            PF_Shift PartID, PF_SHIFT_BACKWARD
            PF_Feeder = PF_CALLBACK_RESTART

        ' Hopper Supply, Center, Flip and Separate
        Case PF_FEEDER_SUPPLY_CENTER_FLIP
            PFControlReturnVal = PF_Control(PartID, PF_CONTROL_SUPPLY)
            PF_Center PartID, PF_CENTER_LONG_AXIS
            PF_Center PartID, PF_CENTER_SHORT_AXIS
            PF_Flip PartID
            PF_Feeder = PF_CALLBACK_RESTART ' Restart and re-acquire images

        ' Too many parts
        Case PF_FEEDER_TOO_MANY
            PFStatusReturnVal = PF_Status(PartID, PF_STATUS_TOOMANYPART)
            PF_Feeder = PF_CALLBACK_RESTART ' Restart and re-acquire images

        ' Wrong part
        Case PF_FEEDER_WRONGPART
            PFStatusReturnVal = PF_Status(PartID, PF_STATUS_WRONGPART)
            PF_Feeder = PF_CALLBACK_RESTART ' Restart and re-acquire images

    Send

Fend
```

3.9.5.2 程序示例 5.2

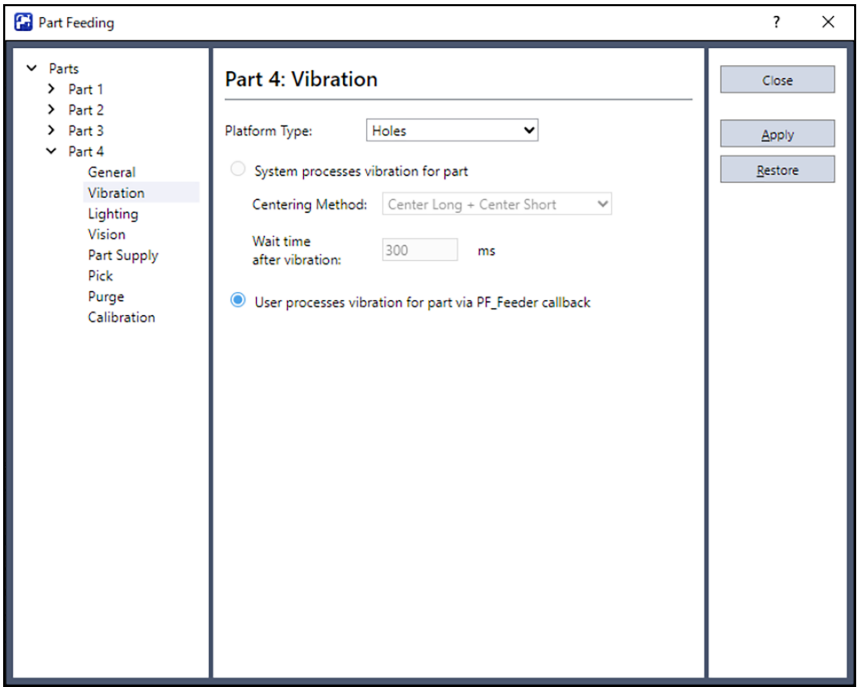
示例类型：
自定义平台（带孔）- 通过PF_Feeder回调函数控制振动

构成

- 机器人数量：1
- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：1
- 平台类型：孔
- 拾取区域：全面
- 相机的方向：固定向下相机



描述
由于是自定义平台，因此自动选择[通过PF_Feeder回调函数控制振动]。



视觉系统获取图像并读入部件队列后，会调用PF_Feeder回调函数。用户代码用于判断在PF_Feeder回调函数内使送料器进行振动的方法。表面方向与背面方向的部件数由自变量提供给PF_Feeder回调函数。这些自变量为“NumFrontParts”与“NumBackParts”。在本例中，NumFrontParts大于“0”时，由于机器人可取放部件，因此无需送料器振动。在这种情况下，回调函数的返回值为“PF_CALLBACK_SUCCESS”。该返回值用于指示系统调用PF_Robot回调函数。

NumFrontParts为“0”时，样本代码用于执行VRun，然后执行部件Blob序列，以判断部件是否处于聚集状态，或者根本没有部件。如果通过部件Blob序列找不到部件，则会将料斗设为ON。部件Blob序列找到部件时，送料器会进行翻转运作、前移运作与后移运作，使部件落入孔中。部件在送料器上振动时，系统必定重新获取视觉图像。（这是因为部件位置因振动而发生了变化。）这是通过将返回值设为“PF_CALLBACK_RESTART”来实现的。这用于在重新获取新的图像并重新加载部件坐标队列后，判断是否需要再次调用PF_Feeder回调函数并进一步进行运作。

提示

翻转、长时间前移、短时间后移是各种自定义平台通用的有效运作。

要点

平台类型为孔、长槽或定位槽时，会将常数PF_FEEDER_UNKNOWN交接给PF_Feeder回调函数。这是因为自定义平台时，系统无法确定适当的送料器运作。

样本代码

Main.prg

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park
    PF_Start 1
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    Do While PF_QueLen(PartID) > 0
        P0 = PF_QueGet(PartID)
        Jump P0
        On Gripper; Wait 0.2
        Jump Place
        Off Gripper; Wait 0.2
        PF_QueRemove PartID
        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf
    Loop
    PF_Robot = PF_CALLBACK_SUCCESS

Fend


Function PF_Feeder(PartID As Integer, NumFrontParts As Integer, NumBackParts As
Integer, state As Integer) As Integer

' Example for Structured Platform with holes state = PF_FEEDER_UNKNOWN

    Integer PFControlReturnVal
    Integer numFound

    Select True

        ' OK to Pick
        Case NumFrontParts > 0
            ' Call PF_Robot because there are parts ready to pick
            PF_Feeder = PF_CALLBACK_SUCCESS '

        ' No Front parts were found but there are Back parts
        Case NumFrontParts = 0 And NumBackParts <> 0

            ' Flip, long Shift Forward and short Shift Backward
            PF_Flip PartID, 500
            PF_Shift PartID, PF_SHIFT_FORWARD, 1000
            PF_Shift PartID, PF_SHIFT_BACKWARD, 300

            PF_Feeder = PF_CALLBACK_RESTART ' Restart and re-acquire images

        ' There are no Front or Back parts found
        ' Either there is a clump of parts or there are no parts on the tray
        ' Acquire an image from the Part Blob sequence to make a determination
        Case NumFrontParts = 0 And NumBackParts = 0

            PF_Backlight 1, On ' Backlight on
            VRun PartBlob ' Acquire Image
            PF_Backlight 1, Off 'Backlight off
            VGet PartBlob.Blob01.NumberFound, numFound ' Were any Blobs found?

            If numFound > 0 Then ' Clump of parts found

                ' Flip, long Shift Forward and short Shift Backward
                PF_Flip PartID, 500
                PF_Shift PartID, PF_SHIFT_FORWARD, 1000
                PF_Shift PartID, PF_SHIFT_BACKWARD, 300

            Else ' No parts found

                ' Call the Control callback to supply more parts
                PFControlReturnVal = PF_Control(PartID, PF_CONTROL_SUPPLY_FIRST)
```

```
            EndIf

            PF_Feeder = PF_CALLBACK_RESTART ' Restart and re-acquire images

    Send

Fend
```

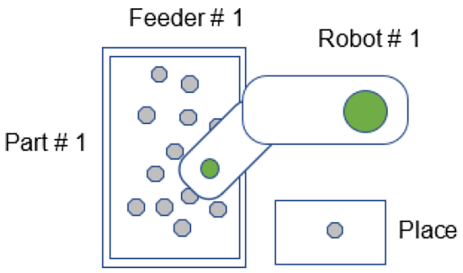
3.9.6 错误处理

3.9.6.1 程序示例 6.1

示例类型：
回调函数内的潜在错误状态的处理

构成

- 机器人数量：1
- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：1
- 相机的方向：送料器#1上的固定向下相机



描述
在本例中，检测并处理回调函数内潜在的错误状态，以免在Part Feeding进程循环内发生错误。在本例中使用PF_Vision回调函数获取图像，并将视觉结果加载至部件坐标队列中。机器人方面有较大的工具偏移。机器人会因情况而超出作业范围，因此可能无法将工具对准部件角度（通过视觉系统检测）。
如果未对错误进行处理，则会发生“坐标转换”错误。该样本代码用于在将坐标加载至部件坐标队列之前，确认是否处于机器人可拾取部件的角度。这是通过TargetOK语句来实现的。

样本代码
Main.prg

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park
    PF_Start 1
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer

    ' Tool 1 will be used to pick up the part
    Tool 1
```

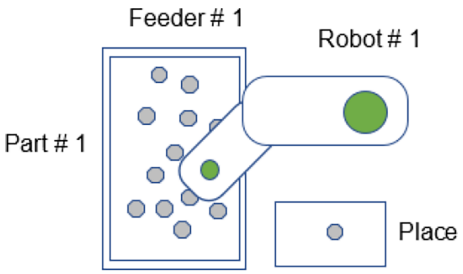
```
    Do While PF_QueLen(PartID) > 0
        P0 = PF_QueGet(PartID)
        Jump P0
        On Gripper; Wait 0.2
        Jump Place
        Off Gripper; Wait 0.2
        PF_QueRemove PartID
        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf
    Loop
    PF_Robot = PF_CALLBACK_SUCCESS

Fend

Function PF_Vision(PartID As Integer, ByRef numBack As Integer) As Integer
    Boolean found
    Integer i, numFront
    Real RB_X, RB_Y, RB_U, RB_Z

    ' Tool 1 will be used to pick up the part
    Tool 1

    ' Pick Z coordinate
    RB_Z = -132.0

    ' Initialize coordinates queue
    PF_QueRemove PartID, All
    PF_Backlight 1, On
    ' Detect the parts
    VRun UsrVisionSeq
    PF_Backlight 1, Off

    VGet UsrVisionSeq.Geom01.NumberFound, numFront 'Front Parts
    VGet UsrVisionSeq.Geom02.NumberFound, numBack 'Back Parts
    If numFront <> 0 Then
        For i = 1 To numFront
            VGet UsrVisionSeq.Geom01.RobotXYU(i), found, RB_X, RB_Y, RB_U
            If found Then
                If TargetOK(XY(RB_X, RB_Y, RB_Z, RB_U)) Then
                    PF_QueAdd PartID, XY(RB_X, RB_Y, RB_Z, RB_U)
                EndIf
            EndIf
         Next
    EndIf

    PF_Vision = PF_CALLBACK_SUCCESS

Fend
```

3.9.6.2 程序示例 6.2

示例类型：
回调函数内部处理时发生错误的处理

构成

- 机器人数量：1
- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：1
- 相机的方向：送料器#1上的固定向下相机



描述

在本例中使用PF_Vision回调函数获取图像，并将视觉结果加载至部件坐标队列中。在本例中，为了加载部件坐标队列，需要送料器上有最小数量的可拾取部件（本例为5个）。即使尝试3次也未加载部件坐标队列时，会显示向操作员询问是继续搜索部件还是停止的信息。

样本代码

Main.prg

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park
    PF_Start 1
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer

    ' Tool 1 will be used to pick up the part
    Tool 1

    Do While PF_QueLen(PartID) > 0
        P0 = PF_QueGet(PartID)
        Jump P0
        On Gripper; Wait 0.2
        Jump Place
        Off Gripper; Wait 0.2
        PF_QueRemove PartID
        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf
    Loop
    PF_Robot = PF_CALLBACK_SUCCESS

Fend

Function PF_Vision(PartID As Integer, ByRef numBack As Integer) As Integer
    Boolean found
    Integer i, numFront
    Real RB_X, RB_Y, RB_U, RB_Z
    Integer RetryCount
    String msg$
    Integer mFlags, answer

    ' Pick Z coordinate
    RB_Z = -132.0

    ' Initialize coordinates queue
    PF_QueRemove PartID, All
    RetryCount = 0
```

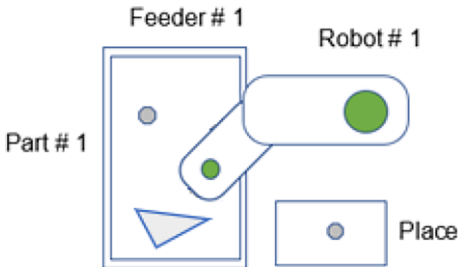
```
    Do
        PF_Backlight 1, On
        ' Detect the parts
        VRun UsrVisionSeq
        PF_Backlight 1, Off

        VGet UsrVisionSeq.Geom01.NumberFound, numFront 'Front Parts
        VGet UsrVisionSeq.Geom02.NumberFound, numBack 'Back Parts
        If numFront >= 5 Then 'Min number of parts = 5 for this example
            For i = 1 To numFront
                VGet UsrVisionSeq.Geom01.RobotXYU(i), found, RB_X, RB_Y, RB_U
                If found Then
                    PF_QueAdd PartID, XY(RB_X, RB_Y, RB_Z, RB_U)
                EndIf
            Next
            Exit Do
        Else
            If RetryCount < 3 Then
                PF_Center 1, PF_CENTER_LONG_AXIS
                PF_Center 1, PF_CENTER_SHORT_AXIS
                PF_Flip 1, 500
                RetryCount = RetryCount + 1
             Else
                msg$ = PF_Name$(PartID) + CRLF + CRLF
                msg$ = msg$ + "Min Number of Parts Cannot be Loaded." + CRLF
                msg$ = msg$ + "Do you want to Continue trying?"
                mFlags = MB_YESNO + MB_ICONQUESTION
                MsgBox msg$, mFlags, "Minimum Number Parts", answer
                If answer = IDNO Then
                    PF_Stop(PartID)
                    Exit Do
                Else
                    RetryCount = 0
                EndIf
            EndIf
        EndIf
    Loop

    PF_Vision = PF_CALLBACK_SUCCESS

Fend
```

3.9.6.3 程序示例 6.3

示例类型：
PF_Status回调函数的状态错误的处理

构成

- 机器人数量：1
- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：1
- 相机的方向：送料器#1上的固定向下相机



描述

在本例中，托盘上的最后部件被检测为不良部件。托盘上安装有清除门选件且已启用。从托盘上排出检测到的“不良部件”，然后通过料斗供给新的部件，执行定芯与翻转运作。

样本代码

PartFeeding.prg

```
Function PF_Status(PartID As Integer, Status As Integer) As Integer

    Select Status

        ' Other Status Cases have been removed from this sample code for simplicity

        Case PF_STATUS_WRONGPART
            ' There may be a wrong part on the feeder platform.
            ' Purge Part 1 without vision feedback. Purge duration is default.
            ' The Purge Gate automatically opens and closes
            PF_Purge 1, PF_PURGETYPE_NOVISION
            ' Turn on the hopper for 3 sec
            PF_OutputOnOff 1, On, 1, 3000
            Wait 3.0
            PF_Center 1, PF_CENTER_LONG_AXIS
            PF_Center 1, PF_CENTER_SHORT_AXIS
            PF_Flip 1, 500

        ' Other Status Cases have been removed from this sample code for simplicity

     Send

    PF_Status = PF_CONTINUE
Fend
```

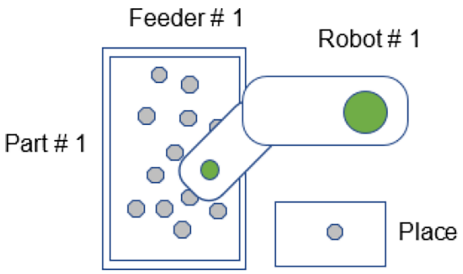
3.9.6.4 程序示例 6.4

示例类型：

PF_Status回调函数的用户错误的处理

构成

- 机器人数量：1
- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：1
- 相机的方向：送料器#1上的固定向下相机



描述

机器人配备带真空开关的真空杯型夹爪，以用于检测部件是否被适当拾取。真空传感器检测部件失败时，机器人会尝试再次拾取部件。如果失败3次，则生成用户错误（8000）。
通过将PF_Robot的返回值设为用户错误编号，将用户错误发送至PF_Status回调函数。PF_Status回调函数用于显示操作员继续执行应用或结束应用的信息框。

样本代码

Main.prg

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park
    PF_Start 1
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer

    Integer PickRetryCount ' Pick Retry Count

    Do While PF_QueLen(PartID) > 0

        ' Get position of part to be picked
        P10 = PF_QueGet(PartID)

        PickRetryCount = 0
        Do
            Jump P10
            On Vacuum
            Wait Sw(VacOn), 0.5 ' 0.5 second timeout on
            Vacuum switch
            If TW = False Then ' Vacuum successful
                Exit Do ' Exit Do Loop and place the part
            EndIf
            Off Vacuum
            PickRetryCount = PickRetryCount + 1 ' Increment retry count
            If PickRetryCount = 3 Then
                ' Vacuum retries were not successful
                Jump Park
                PF_QueRemove PartID
                PF_Robot = 8000 ' Set the return value to user
                Error 8000
                ' PF_Status callback will be called with status value 8000
                 Exit Function
             EndIf
         Loop

        ' Part detected in vacuum gripper
        Jump Place
```

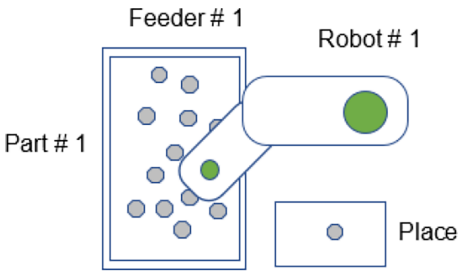
```
        Off Vacuum
        Wait 0.25

        ' Deque
        PF_QueRemove PartID

        'Check Cycle stop
        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf

     Loop

    PF_Robot = PF_CALLBACK_SUCCESS

Fend

Function PF_Status(PartID As Integer, Status As Integer) As Integer
    String msg$
    Integer mFlags, answer

    Select Status
        ' Other Status Cases have been removed from this sample code for simplicity

        Case 8000 ' User Error 8000 occured.

            msg$ = PF_Name$(PartID) + CRLF + CRLF
            msg$ = msg$ + "Vacuum Pick error has occurred." + CRLF
            msg$ = msg$ + "Do you want to Continue?"
            mFlags = MB_YESNO + MB_ICONQUESTION
            MsgBox msg$, mFlags, "Vacuum Pick Error", answer
            If answer = IDNO Then
                PF_Status = PF_EXIT
            Else
                PF_Status = PF_CONTINUE
            EndIf

            Exit Function
    Send

Fend
```

3.9.6.5 程序示例 6.5

示例类型：
Part Feeding回调函数内的控制器错误的处理

构成

- 机器人数量：1
- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：1
- 相机的方向：送料器#1上的固定向下相机



描述

通常，发生控制器错误时，会通过Part Feeding进程自动将常数PF_STATUS_ERROR设为Status参数，并调用PF_Status回调函数。没有添加的用户代码时，会发生这种现象。PF_Status回调函数用于输出或显示错误编号与信息。如果PF_Status回调函数结束，Part Feeding进程则会结束。
但在本例中，需要处理PF_Robot回调函数内的特定控制器错误。
发生其他所有的控制器错时，在将PF_STATUS_ERROR被设为Status参数的状态下，调用PF_Status回调函数。
在这种情况下，为了防止损坏夹爪的电气配线与气压配管，需要避免U轴（SCARA型机器人）进行大于等于+/-360° 的旋转动作。
通过Epson RC+的机器人管理器限制轴4的运作范围。如果限制轴4的运作范围，则可防止损坏电气配线与气压配管。送料器上的部件要对轴4进行超出其运作范围的旋转动作时，会发生错误4001“机械臂已达到运作范围的极限”。错误处理器从队列中删除部件，机器人也会重新开始拾取剩余的所有部件。
视觉系统重新获取相同被拒部件的图像，然后，选择所有的部件并在队列清空后执行PF_Flip，以防止再次添加到队列中。

样本代码

Main.prg

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park
    PF_Start 1
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    Integer errNum

    OnErr GoTo ehandle ' Error Handler

retry:
    Do While PF_QueLen(PartID) > 0

        ' Get position of part to be picked
        P10 = PF_QueGet(PartID)

        ' Error 4001 can occur if the part's angle
        ' causes Joint 4 to rotate beyond its motion range
        Jump P10

        On Gripper
        Wait 0.25
        Jump Place
        Off Gripper
        Wait 0.25

        'Deque
        PF_QueRemove PartID
```

```
        'Check Cycle stop
        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf

    Loop

    PF_Flip PartID

    PF_Robot = PF_CALLBACK_SUCCESS
    Exit Function

ehandle:

    errNum = Err
    If errNum = 4001 Then ' Example of Handled error
        Print "Error 4001: Arm reached the limit of motion range"
        PF_QueRemove PartID ' Remove the part from the queue
        EResume retry ' Continue picking the remaining parts in the queue
    Else
        ' Other unhandled errors
        ' PF_Status is called with the PF_STATUS_ERROR status parameter
        PF_Robot = PF_STATUS_ERROR
    EndIf
Fend

Function PF_Status(PartID As Integer, Status As Integer) As Integer

    Select Status

        ' Other Status Cases have been removed from this sample code for simplicity

        Case PF_STATUS_ERROR ' Error.
            msg$ = PF_Name$(PartID) + CRLF
            msg$ = msg$ + "Error!! (code: " + Str$(Err) + " ) " + ErrMsg$(Err)
            MsgBox msg$, MB_ICONSTOP

        ' Other Status Cases have been removed from this sample code for simplicity

     Send

    If Status = PF_STATUS_ERROR Then
        ' A controller error occurred. Terminate the Part Feeding Process.
        PF_Status = PF_EXIT
    Else
        ' Otherwise Continue running the Part Feeding Process.
        PF_Status = PF_CONTINUE
    EndIf

Fend
```

3.9.7 使用多台相机

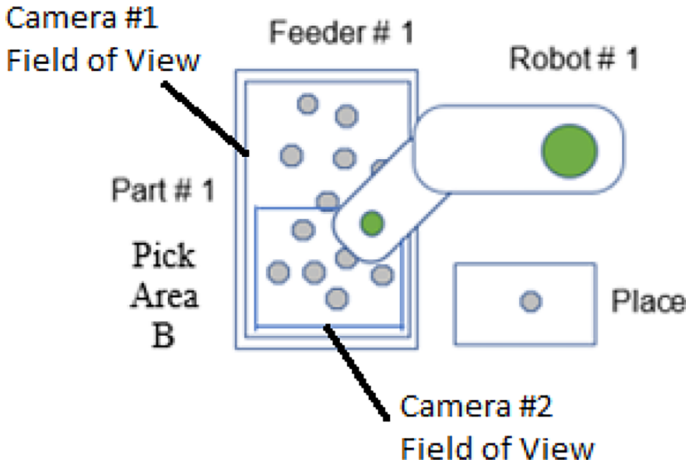
本章节说明使用多台相机以提高拾取精度的方法。

3.9.7.1 程序示例 7.1

示例类型：
使用多台固定向下相机以提高拾取区域的精度

构成

- 机器人数量：1
- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：1
- 平台类型：平面
- 拾取区域：区域B
- 相机的方向
  - 相机#1：向下固定。视野为送料器托盘全体。用于部件Blob序列
  - 相机#2：向下固定。视野与区域B相同（托盘的一半）。用于部件检测序列

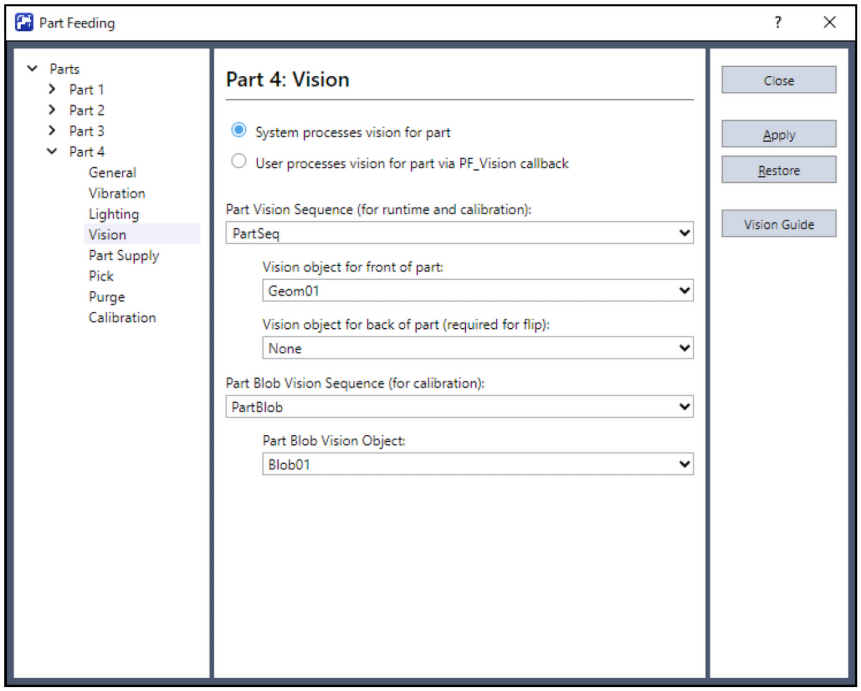


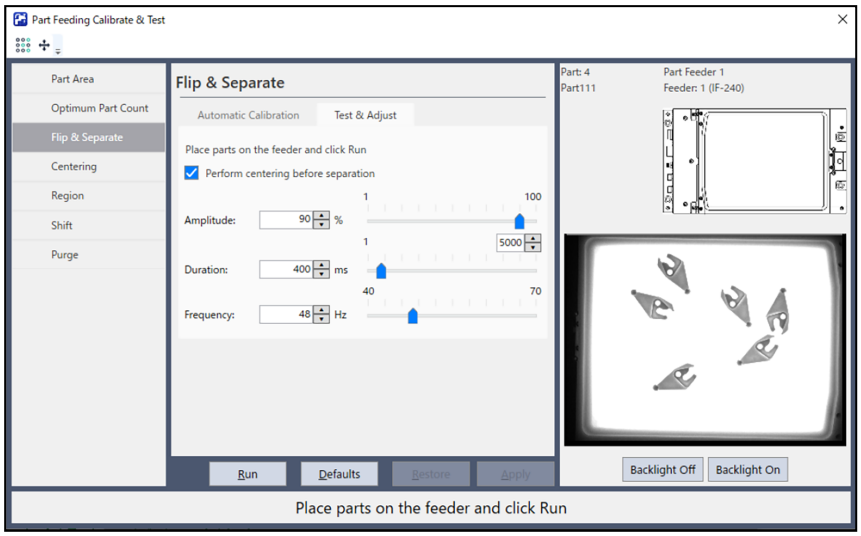
描述

将相机#1的视野设为映照送料器托盘全体。部件Blob序列使用相机#1。部件Blob序列用于根据托盘内的部件数与分布情况，确定送料器的振动类型。将相机#2的视野设为映照拾取区域B全体。为了有效地使用相机视野，需要将相机#2相对于相机#1旋转90度（也就是说，相机相互正交）。相机#2用于部件检测序列。
由部件检测序列检测的部件被用于生成部件坐标队列。由于视野大小为相机#1视野的一半，因此可大幅提高分辨率（mm/像素）。
此外，相机#1被用于判断送料器的振动方法，故此可使用分辨率低于相机#2的相机。比如，相机#1的分辨率为640×480像素，相机#2的分辨率为5472×3648像素。
通过缩窄视野，提高相机的分辨率，机器人的拾取精度得以提高。

翻转&分离的自动校准时，使用部件检测序列，确认找到可拾取的部件。部件检测序列被用于视野较小的相机时，如果未实施添加步骤，翻转&分离的自动校准则不会起到正确作用。
请实施下述某种方法：

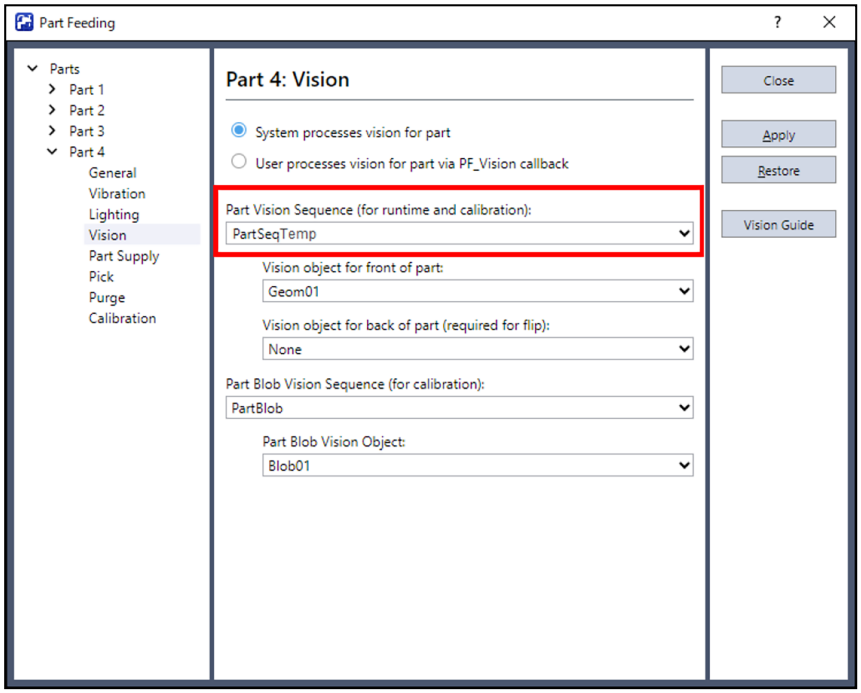
1. 跳过自动校准，手动调整翻转&分离校准参数。



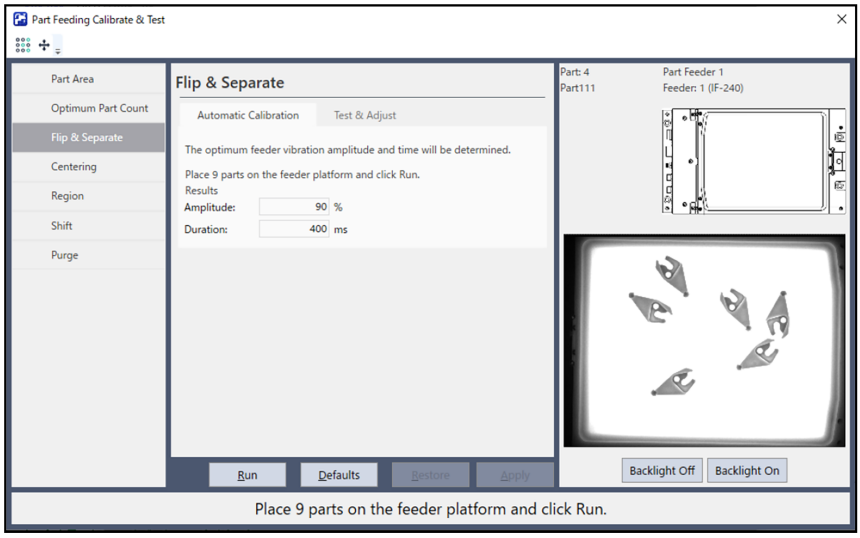


2. 创建仅用于自动校准且使用相机#1（较宽的视野）的临时性部件检测序列。
在本例中，使用相机#2（较窄的视野）的部件检测序列的名称为“PartSeq”。使用相机#1（较宽的视野）的临时性部件检测序列的名称为“PartSeqTemp”。“PartSeqTemp”时，使用Geometric对象搜索部件。

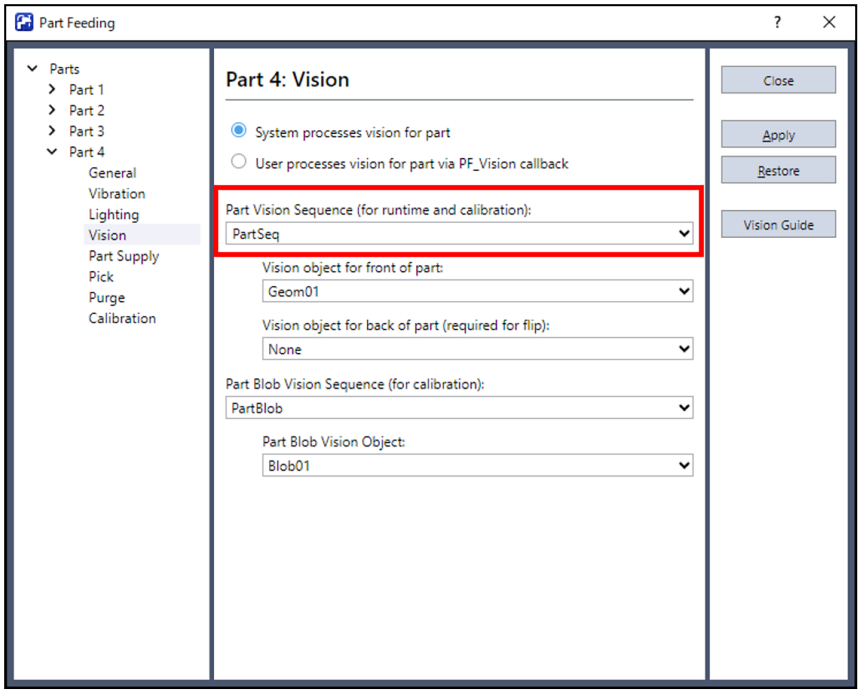
“PartSeqTemp”时，仅临时用于送料器的校准。作为部件检测视觉序列，选择“PartSeqTemp”（请参阅以下内容）。



3. 切换为Part Feeding对话框的校准&测试页面，执行分离的自动校准。



4. 完成所需的所有校准后，关闭校准&测试对话框。将部件检测视觉序列返还给“PartSeq”（使用较窄视野相机#2）。执行时，使用“PartSeq”的结果生成部件坐标队列。



无需特别代码。

样本代码

Main.prg

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park
    PF_Start 1
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    Do While PF_QueLen(PartID) > 0
        P0 = PF_QueGet(PartID)
        Jump P0
        On Gripper; Wait 0.2
        Jump Place
        Off Gripper; Wait 0.2
        PF_QueRemove PartID
        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf
    Loop
    PF_Robot = PF_CALLBACK_SUCCESS

Fend
```

3.9.7.2 程序示例 7.2

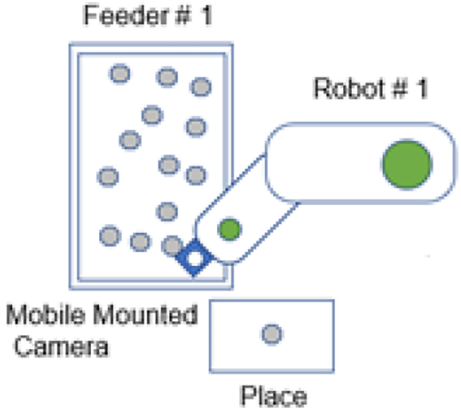
示例类型：

使用固定向下相机与移动相机双方，提高拾取精度

构成

- 机器人数量：1

- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：1
- 平台类型：平面
- 拾取区域：全面
- 相机的方向
  - 相机#1：向下固定。视野为送料器托盘全体。用于部件Blob序列与部件检测序列
  - 相机#2：移动（J2轴）。视野略大于部件。拾取前获取部件的二次图像



描述

移动相机（轴2）被定义为“机械臂”（仅限于SCARA型机器人）。使用6轴机器人时，可对移动相机（轴6）定义工具（而非机械臂）。

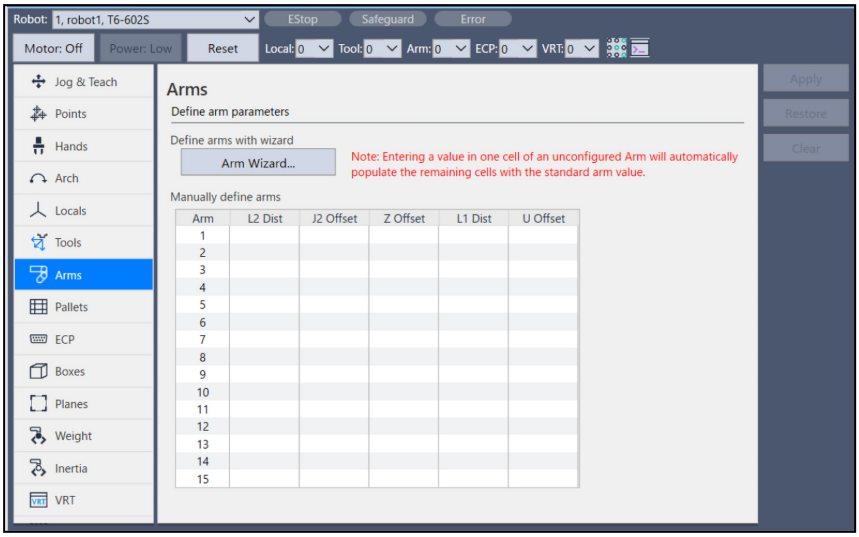
将相机#2移动至部件上部，以替代发出直接将夹爪移动至部件坐标的命令。执行添加的视觉序列，通过部件确定准确的拾取位置。

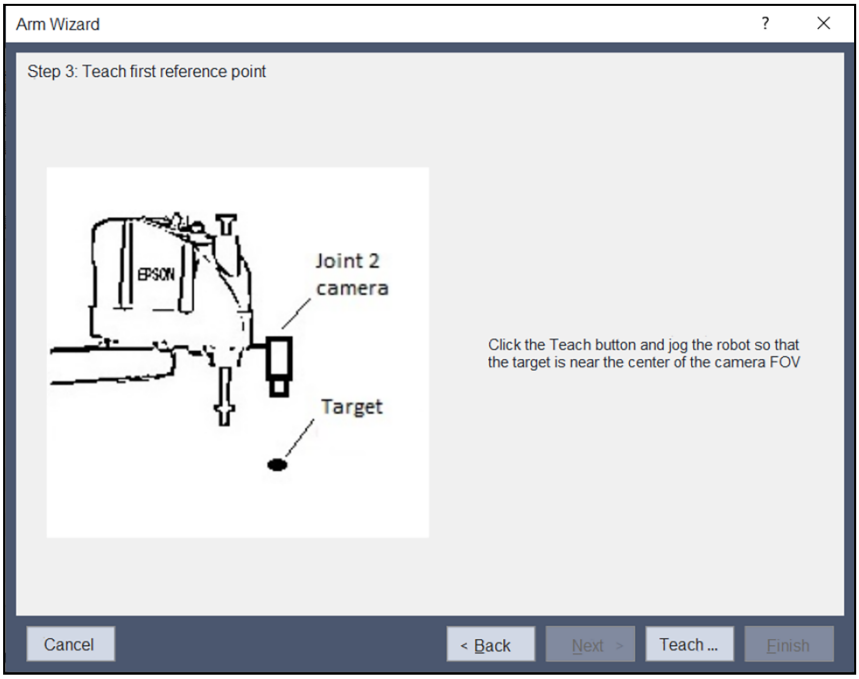
移动相机的视野远小于固定向下相机，因此拾取精度得以提高。因为要进行将相机移动至部件上方所需的追加运作与视觉处理，整个循环时间会延长。

要自动定义机械臂时，切换为Epson RC+ - 工具 - 机器人管理器。选择机器人管理器中的[增设手臂]选项卡。在本例中，选择Arm 1，然后单击[Arm 向导]按钮。执行向导的各步骤。

有关向导的详细信息，请参阅以下手册。

“Vision Guide 8.0软件篇 - 设定相机设置位置的机械臂”





创建可对移动相机进行校准且可找到送料器上单一部件的视觉序列。为该序列时，在PF_Robot回调函数内执行(VRun)。本例中的序列名称为“MobileCam”。未在上料对话框中选择“MobileCam”。部件Blob序列与部件检测序列使用相机#1，如通常一样，在上料对话框中进行选择。

样本代码

Main.prg

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park
    PF_Start 1
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer

    Boolean found
    Real x, y, u
```

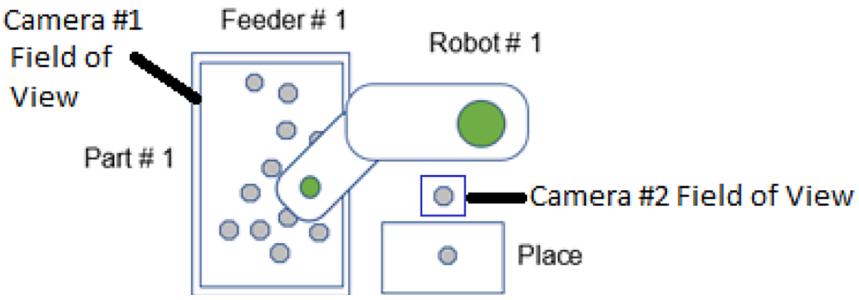
```
    Do While PF_QueLen(PartID) > 0
        P0 = PF_QueGet(PartID)
        Arm 1 'Select the Arm that is defined for the Mobile Camera
        Jump P0 :Z(0) 'Position the Mobile camera over the part
        VRun MobileCam
        VGet MobileCam.Geom01.RobotXYU, found, x, y, u
        Arm 0 'Select default robot arm
        If found Then
            Jump XY(x, y, PICKZ, u) /R
            On Gripper; Wait 0.2
            Jump Place
            Off Gripper; Wait 0.2
        EndIf
        PF_QueRemove PartID
        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf
    Loop
    PF_Robot = PF_CALLBACK_SUCCESS

Fend
```

3.9.7.3 程序示例 7.3

示例类型：
使用多台固定相机以提高拾取精度

构成

- 机器人数量：1
- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：1
- 平台类型：平面
- 拾取区域：全面
- 相机的方向
  - 相机#1：向下固定。视野为送料器托盘全体。用于部件Blob序列与部件检测序列
  - 相机#2：向上固定。用于创建机器人夹爪保持部件的工具偏移



描述
相机#1位于送料器托盘之上，处于向下固定状态。相机#2处于向上固定状态。相机#1的视野覆盖送料器的托盘全体。部件Blob视觉序列与部件检测序列使用相机#1。
相机#2的视野略大于部件。机器人从送料器上拾取部件，然后它会将该部件移动至相机#2的上方。相机#2被用于动态创建夹爪保持部件的工具偏移。机器人使用新定义的工具偏移来配置部件。工具偏移用于对从送料器拾取的不准确性进行补偿。
相机#2仅被用于PF_Robot回调函数。未在Epson RC+ 8.0菜单 - [工具] - [上料] - [视觉]中设定相机#2的视觉序列。样本代码时，使用相机#2的VGet RobotToolXYU结果确定工具偏移。

该PF_Robot函数首先执行利用夹爪抓取部件的步骤。接下来，使用VGet RobotToolXYU获取工具偏移，并使用TLSet定义工具。然后，机器人使用新的工具偏移来配置部件。

在本例中，需要对向上固定相机进行校准。有关向上固定相机校准方法的详细信息，请参阅以下手册。
“Vision Guide 8.0软件篇 - 校准步骤：向上固定相机”

样本代码

Main.prg

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park
    PF_Start 1
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
   Boolean found
   Real xTool, yTool, uTool

   Do While PF_QueLen(PartID) > 0
       P0 = PF_QueGet(PartID)
       Tool 0 ' Select the correct Tool number for the Gripper
       Jump P0
       On Gripper; Wait 0.2
       Jump upCam
       VRun findPartInGripper
       VGet findPartInGripper.Geom01.RobotToolXYU, found, xTool, yTool, uTool
       If found Then
           TLSet 1, XY(xTool, yTool, 0, 0)
           Tool 1
           Jump Place
       Else
            Jump reject ' Part not found in gripper - reject part
       EndIf
       Off Gripper; Wait 0.2
       PF_QueRemove PartID
       If PF_IsStopRequested(PartID) = True Then
           Exit Do
       EndIf
   Loop
   PF_Robot = PF_CALLBACK_SUCCESS
Fend
```

3.9.8 改善视觉结果

本章节说明改善视觉结果的方法。

3.9.8.1 程序示例 8.1

示例类型：
使用图像缓冲区与ImageOp、SubtractAbs

构成

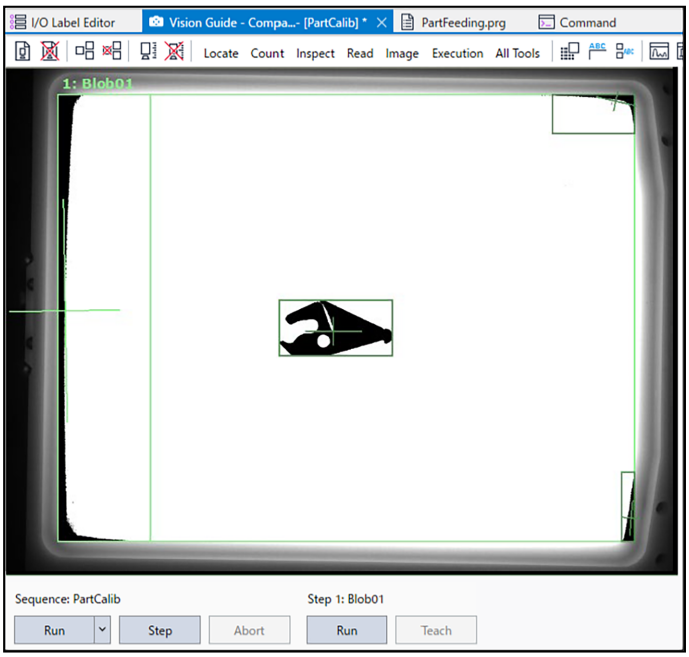
- 机器人数量：1

- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：1
- 相机的方向：送料器#1上的固定向下相机

描述

即使将背光灯设为ON，也可能会在托盘边角上留下影子。Blob对象的搜索窗口中包含有托盘的边角，并且未适当调整阈值时，可能会误将影子识别为部件。
部件Blob视觉序列被用于检测单个部件或部件总数。部件Blob视觉序列将影子视为部件时，系统会错误确定振动方法或判断托盘上没有足够的部件，因此，可能导致PF_Control回调函数不将料斗设为ON。

如下所述为误将托盘边角的影子视为部件的示例。

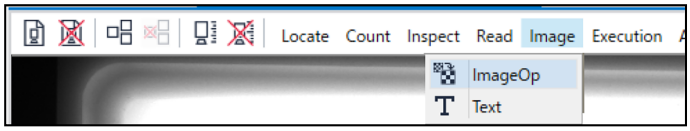


如果将ImageOp视觉对象添加至PartBlob序列，则可能有助于解决该问题。使用ImageOp的SubtractAbs处理。SubtractAbs用于输出2个图像缓冲区的差异。在本例中，ImageBuffer1属性为空白文件夹的图像文件，ImageBuffer2属性为通过相机获取的图像（值“0”表示相机的图像缓冲区）。如果扣除图像缓冲区，则会有效地删除文件夹中的图像。

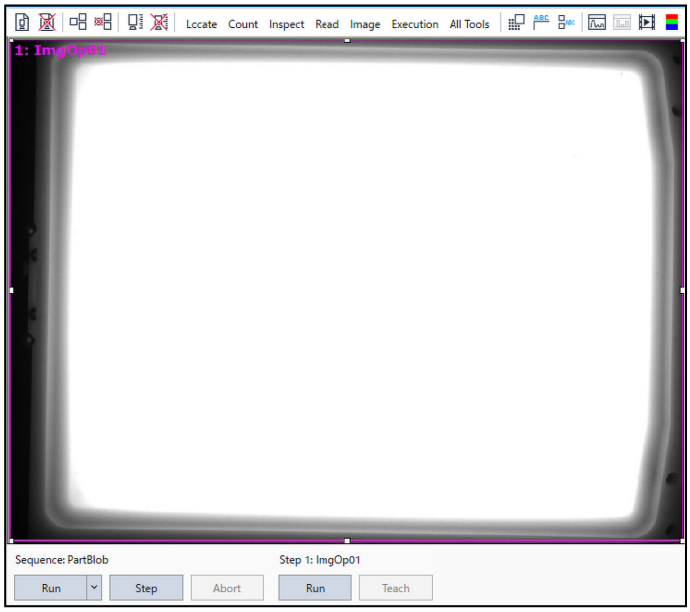
如下所述为其步骤。

1. 创建新的视觉序列“PartBlob”。
2. 将送料器的背光灯设为ON，除去送料器托盘上的所有部件。
3. 单击PartBlob的SaveImage属性中的[Click to save]按钮。在本例中，将图像文件命名为“EmptyIF-240”。如下所示为文件的图像。

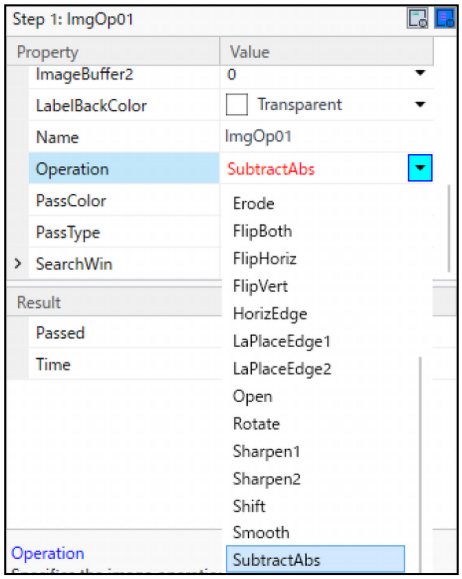
4. 通过视觉引导的工具栏，拖放ImageOp对象。



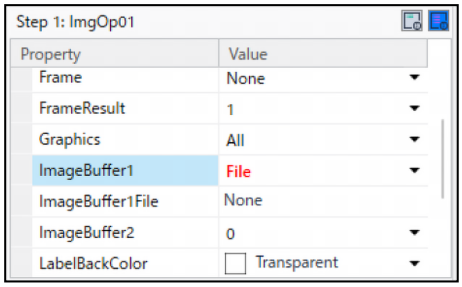
5. 更改ImageOp的尺寸，设为相机视野全体。



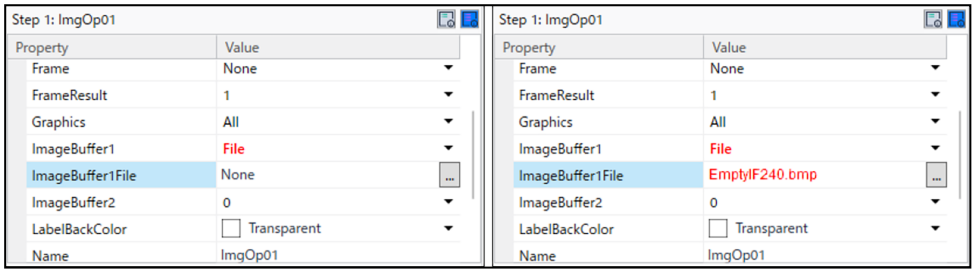
6. 将ImageOp的Operation属性更改为“SubtractAbs”。



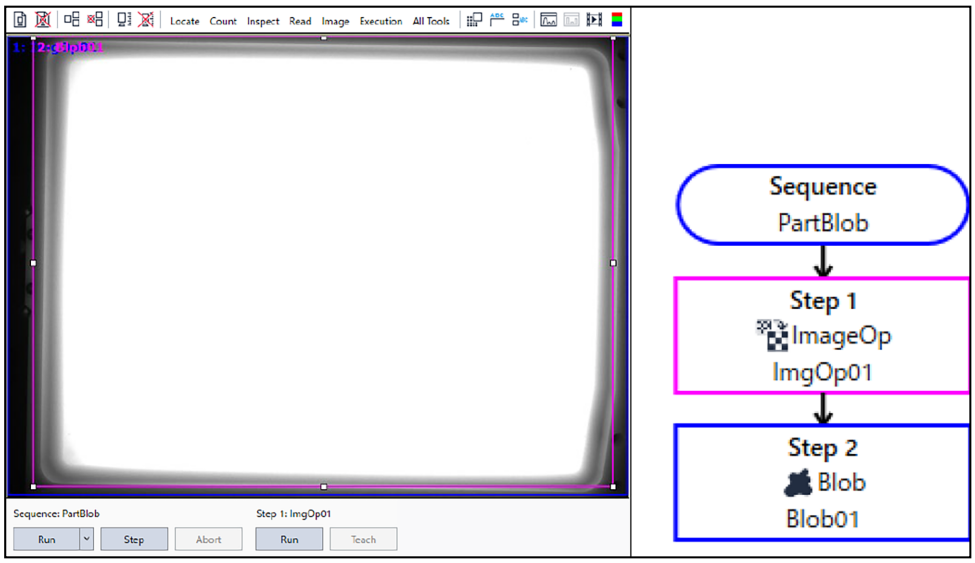
7. 将ImageOp的ImageBuffer1属性更改为“File”。



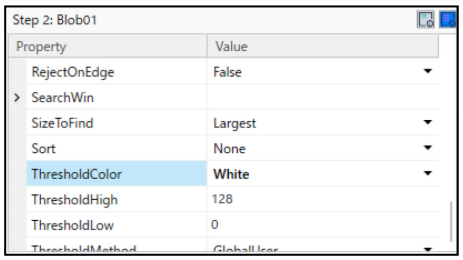
8. 单击ImageBufferFile按钮，选择以前作为“EmptyIF-240”保存的文件。



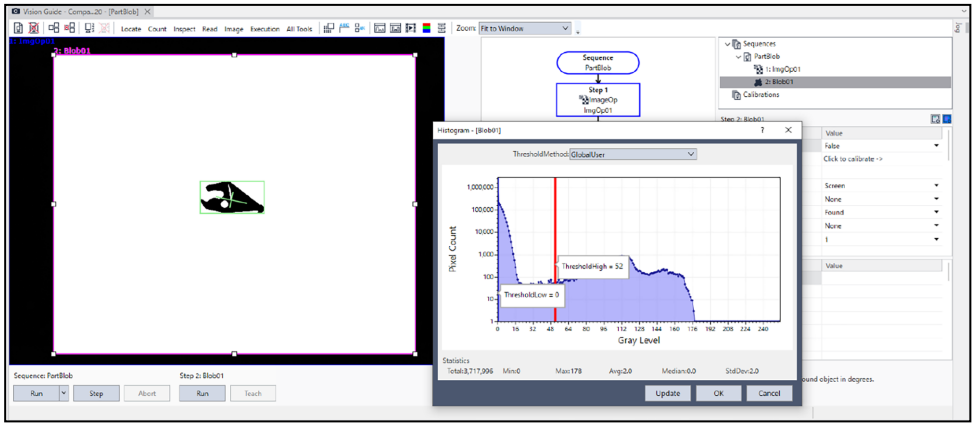
9. 通过视觉引导的工具栏，添加Blob对象。更改Blob搜索窗口的尺寸，设为大于送料器托盘。



10. 将Blob的ThresholdColor属性更改为“White”。

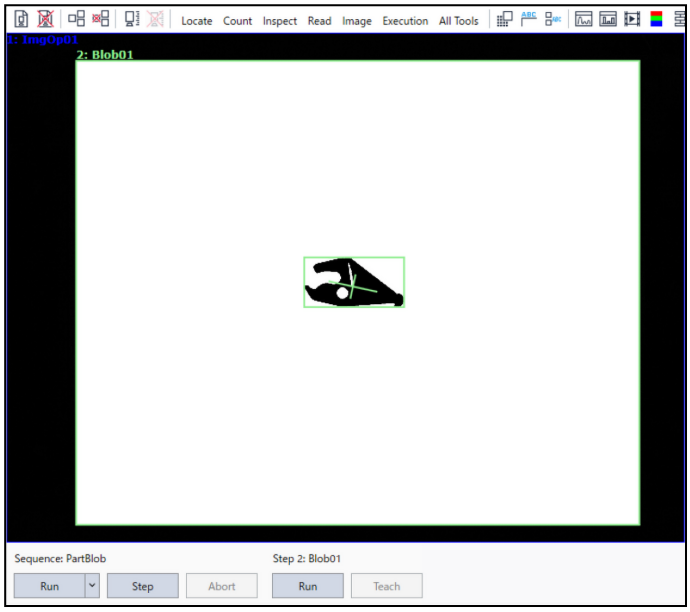


11. 将1个部件放到托盘上，然后按下Vision Guide工具栏中的[直方图]。

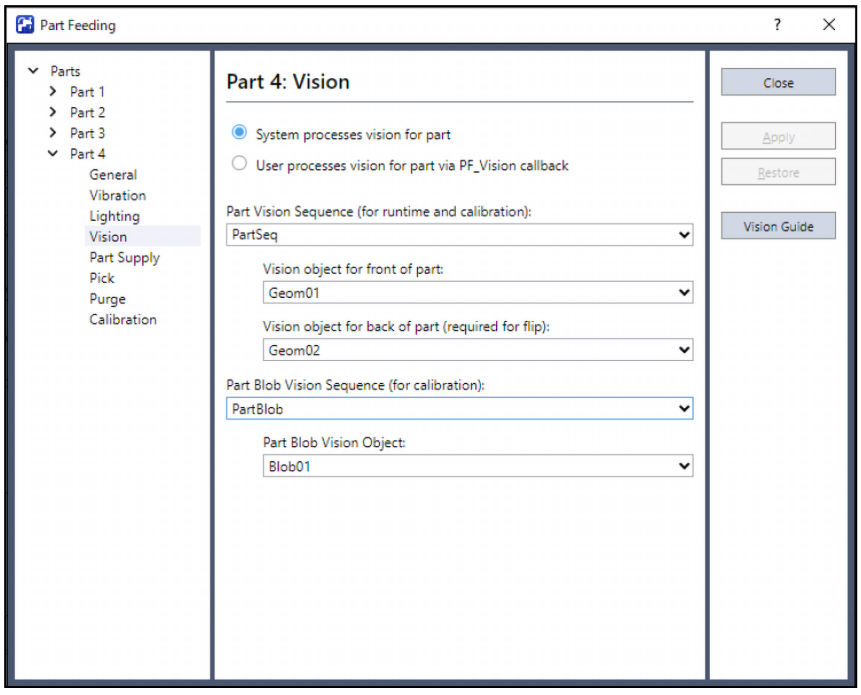


拖动直方图中的红色ThresholdHigh条，直至部件被适当地二值化。根据需要单击[更新]，结束之后单击直方图窗口中的[OK]按钮。

通过上述操作后，如果执行PartBlob，则会删除所有的送料器图像。条也会在全白背景中显示为黑色。送料器被PartBlob序列有效地屏蔽。



12. 在Epson RC+ 8.0菜单 - [工具] - [上料]视觉页面中，作为目标部件的部件Blob视觉序列，设定“PartBlob”。



无需特别代码。

样本代码

Main.prg

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park
    PF_Start 1
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    Do While PF_QueLen(PartID) > 0
        P0 = PF_QueGet(PartID)
        Jump P0
        On Gripper; Wait 0.2
        Jump Place
        Off Gripper; Wait 0.2
        PF_QueRemove PartID
        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf
    Loop
    PF_Robot = PF_CALLBACK_SUCCESS

Fend
```

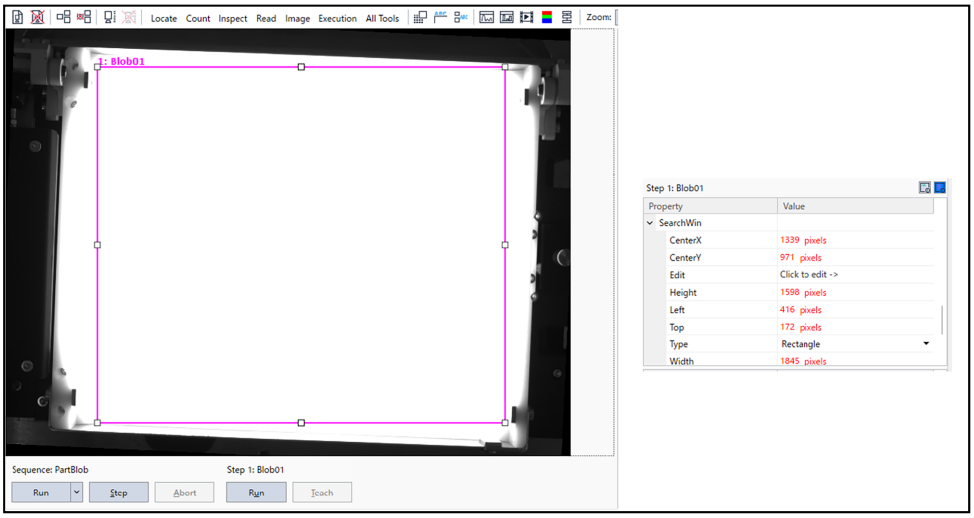
3.9.8.2 程序示例 8.2

示例类型：
将RotatedRectangle用于部件Blob视觉序列的SearchWinType

构成

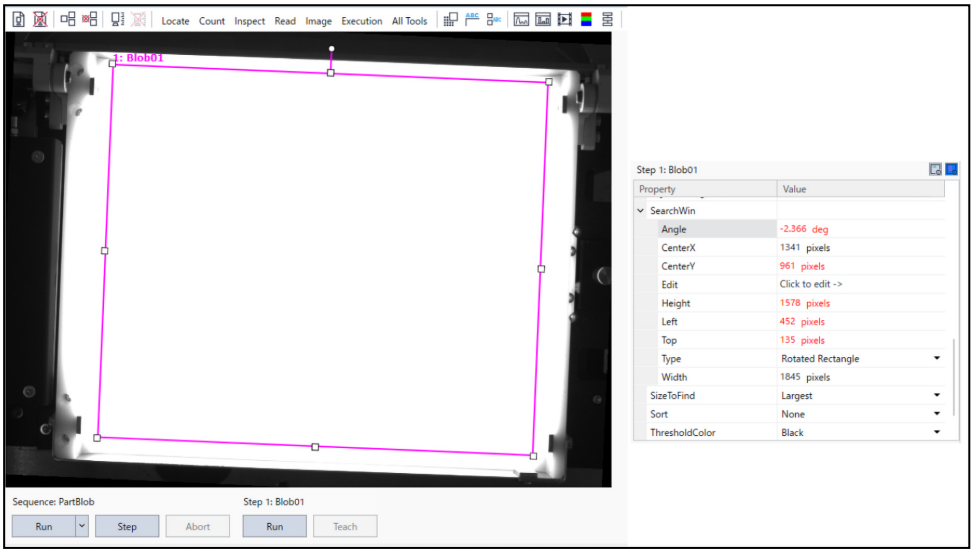
- 机器人数量：1
- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：1
- 相机的方向：送料器#1上的固定向下相机

Description
相机视野与送料器托盘未相互平行时，可能难以适当地设定PartBlob搜索窗口的尺寸。如果搜索窗口过大，PartBlob则可能会将托盘检测为部件。如果搜索窗口过小，系统则无法适当地判断托盘上的部件的数量与分散状态。EPSON RC+ 7.5.2以后版本时，PartBlob对象可将“RotatedSearchWin”用于SearchWinType属性。



如下所述为SearchWinType被设为“Rectangle”时的示例。

Part Blob的SearchWinType被设为“RotatedRectangle”时，可对齐相机视野与送料器托盘。



要点

请将SearchWin Angle属性设为+/- 45° 。

无需特别代码。

样本代码

Main.prg

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park
    PF_Start 1
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    Do While PF_QueLen(PartID) > 0
        P0 = PF_QueGet(PartID)
        Jump P0
        On Gripper; Wait 0.2
        Jump Place
        Off Gripper; Wait 0.2
        PF_QueRemove PartID
        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf
    Loop
    PF_Robot = PF_CALLBACK_SUCCESS

Fend
```

3.9.8.3 程序示例 8.3

示例类型：
将检测掩模用于部件Blob视觉序列的搜索窗口

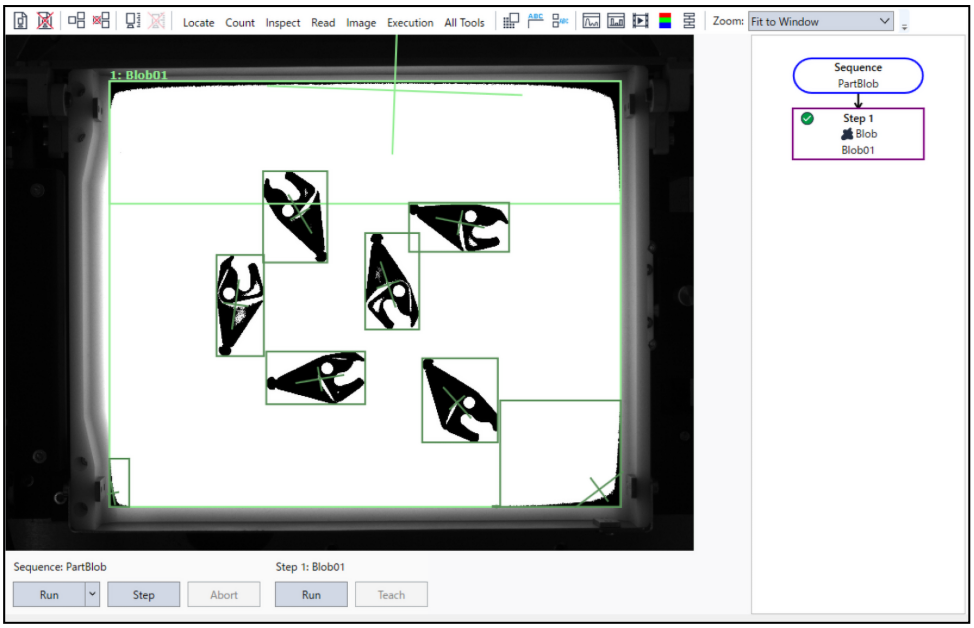
构成

- 机器人数量：1
- 送料器数量：1
- 送料器上的部件类型数量：1
- 配置位置数量：1
- 相机的方向：送料器#1上的固定向下相机

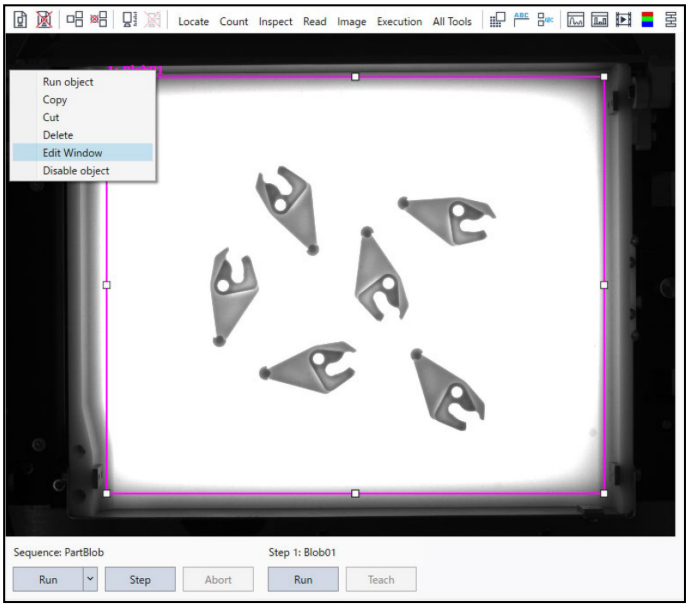
描述

即使将背光灯设为ON，也可能会在托盘边角上留下影子。Blob对象的搜索窗口中包含有托盘的边角，并且未适当调整阈值时，可能会误将影子识别为部件。部件Blob视觉序列被用于检测单个部件或部件的堆积状况。部件Blob将影子视为部件时，系统可能会错误确定振动方法。

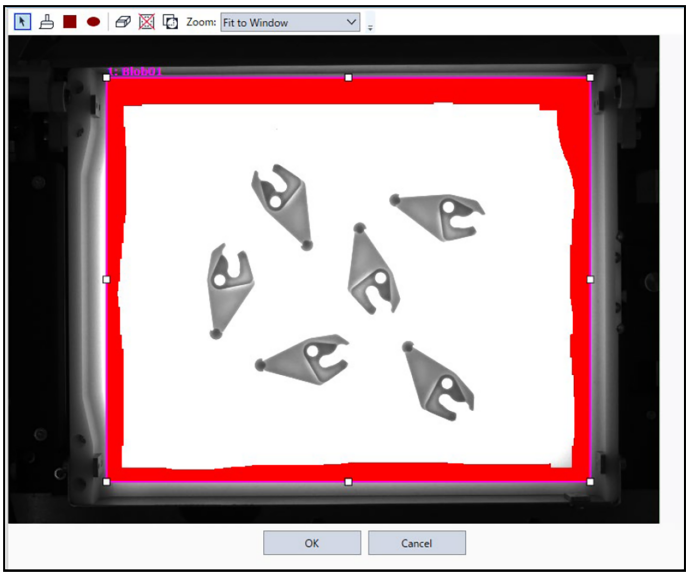
如下所述为误将托盘边角的影子视为部件的示例。



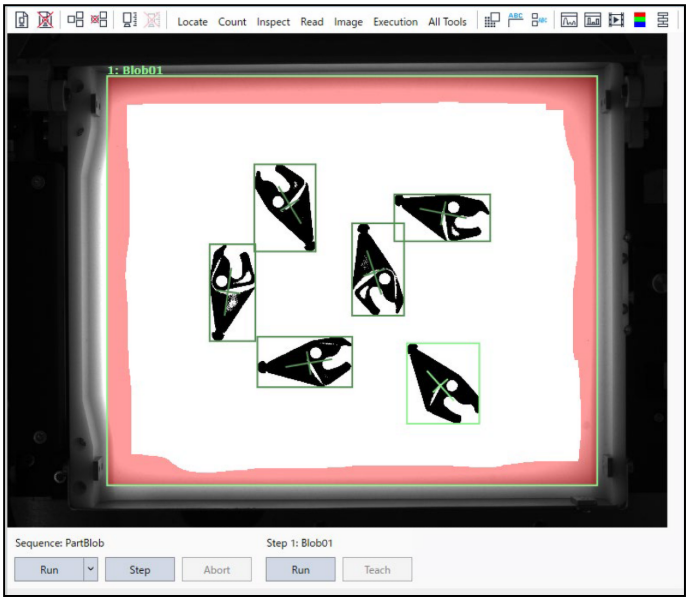
通过在Blob搜索窗口中的非识别区域（此时为托盘边角）中设定检测掩模，可进行屏蔽。
右键单击Blob，然后从弹出菜单中选择[窗口的编辑]，或选择Blob的SearchWin的[编辑窗口]属性。



使用漆刷进行涂刷，以便从搜索窗口中删除托盘。



如果此时执行Blob，搜索窗口中则不包含托盘（请参阅以下内容）。



无需特别代码。

样本代码

Main.prg

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
    Power Low
    Jump Park
    PF_Start 1
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    Do While PF_QueLen(PartID) > 0
        P0 = PF_QueGet(PartID)
        Jump P0
        On Gripper; Wait 0.2
        Jump Place
        Off Gripper; Wait 0.2
        PF_QueRemove PartID
        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf
    Loop
    PF_Robot = PF_CALLBACK_SUCCESS

Fend
```

4. 发展篇

4.1 多部件 & 多机器人

本章节说明在同一送料器上同时使用多个部件的方法。此外，也说明按组处理部件时，多台机器人如何安全地访问同一送料器。

4.1.1 多部件 & 多机器人的规格与要件

- 1台机器人控制器最多可使用4台零件送料器。
- 机器人控制器之间不能共享零件送料器。
  比如，2台T/VT系列机器人不能共享同一送料器。要利用多台机器人控制1台送料器时，需要使用1台支持多台机器人的机器人控制器。
- 可在1个系统中任意组合送料器的型号。
  比如，分别用1台IF-80、IF-240、IF-380、IF-530等进行各种组合。
- 可同时向同一送料器送入最多4种部件。
  要同时向同一送料器送入多种部件时，部件需要具备相似的物理特征。也就是说，重量相同，尺寸或材质相似，面积也必须基本相同。各部件使用固有的振动参数。因此，需要通过部件的振动参数，确保其他部件不会从送料器弹出。
- 最多2台机器人可同时使用同一送料器。
  如果针对部件组执行PF_Start (PF_Start 1、2、5) 并将部件分配给大于等于2台上的机器人，则会发生错误。
- 如果执行PF_Start，单一部件或部件组则会被送至送料器上。
  如果针对已分配给同一送料器编号的部件多次执行PF_Start，则会显示“送料器已被使用”的信息（由PF_Status回调函数发行），不会执行第2次的PF_Start。
- 要针对部件组执行PF_Start时 (PF_Start 1、2、5)，需要将所有的部件分配给同一送料器编号。
- PF_Start用于将单一部件或部件组送至送料器上。
  固有的控制器任务编号会被分配到正在运作的各送料器上。（请参阅下表）
  请勿在用户的应用代码中使用被确保用于送料器的任务。未使用送料器编号时，可将其任务用于用户代码。使用PF_InitLog语句获取Part Feeding的数据日志时，由于特定的计时器编号被确保用于系统，因此不能用于用户代码。使用送料器控制命令 (PF_Center、PF_ConterByShift、PF_Flip、PF_Shift) 时，由于特定的SyncLock编号被确保用于系统，因此不能用于用户代码。任务编号、计时器编号与SyncLock编号是送料器编号固有的。（请参阅下表）

送料器编号	任务	计时器	SyncLock
1	32	63	63
2	31	62	62
3	30	61	61
4	29	60	60

- PF_Start语句中记载的第1个部件是最初的“有效部件”（最初请求的部件）。使用PF_ActivePart语句选择下一个请求的部件。本章节详细说明PF_ActivePart。
- 针对当前的有效部件执行进给运作（振动、拾取区域、供给方式）。PF_Start语句中记载的所有部件被选为有效部件时，会使用各自的设定。比如，应用需要时，PF_Start组的各部件可能具有不同的拾取区域。
- 交接给各回调函数的部件ID (PartID) 为当前有效部件（目前请求的部件）的部件编号。

- 针对PF_Start语句中记载的各部件，获取图像，并读入各部件的队列与视觉结果。各部件使用各自的视觉设定与照明设定。比如，PF_Start列表中的某部件使用“system processes vision”，其他部件使用“user processes vision”等，所有的部件都使用各自的设定。同样地，各部件可使用前灯、不带背光灯、不同的亮度等独自的照明条件。
- Part Feeding日志文件（通过PF_InitLog语句开始）中包含有1台送料器上的所有部件的数据。
- 多台机器人要访问同一送料器时，需要在用户的应用代码中使用PF_AccessFeeder与PF_ReleaseFeeder语句，以免机器人发生碰撞。本章节将详细说明这方面的情况。
- 所有部件都需要单独进行校准。假设同时送入1台送料器中的部件都具有相似的特征（尺寸、重量、材质等）。

4.1.2 多部件 & 多机器人的主要概念

4.1.2.1 PF_ActivePart

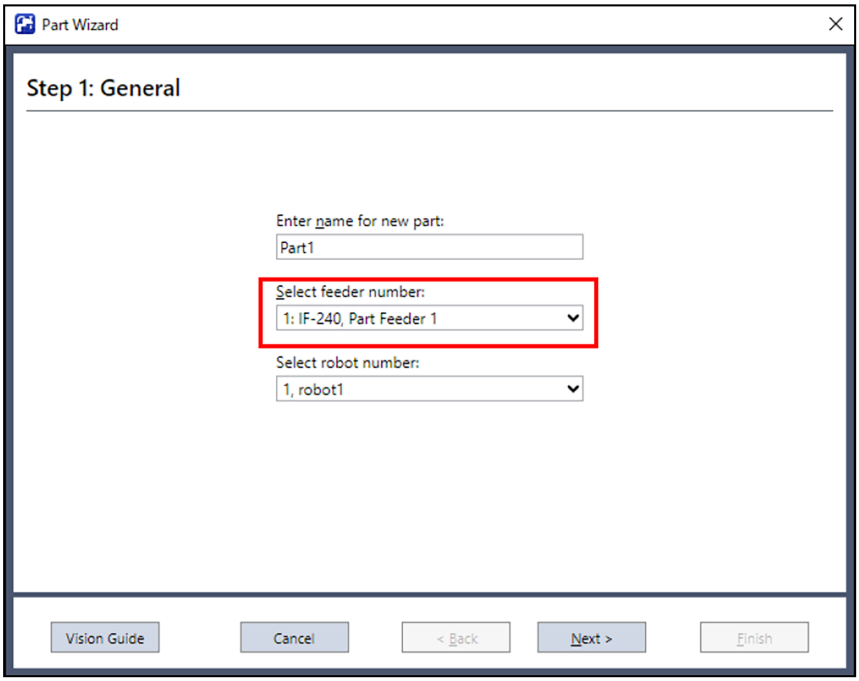
PF_ActivePart语句用于在执行时将用户意图通知给Part Feeding进程。系统需要掌握哪个部件被请求。系统会投放部件，以确保可使用被请求的部件。（使用适当的振动设定并从料斗供给部件等。）

下面简单说明执行时需要PF_ActivePart的理由。首先说明所有部件被同一送料器投放的“配套”应用。空箱被分发给传送带上的机器人。箱上的条形码表示要配置到箱中的部件类型。箱子被提示给条形码阅读器之前，并不了解所需的部件类型与拾取数。此时零件送料器会振动，找到所需的部件并完成命令。

作为使用PF_ActivePart的另1个使用示例，下面说明2个部件的组装操作。2种部件位于同一零件送料器上。在该应用中，需要机器人拾取1个部件#1并放置到固定件上。接下来，机器人拾取2个部件#2并插入到部件#1中。PF_ActivePart用于向系统发出需要投放哪个部件的指示，以完成组装进程。为了进一步实现该应用，假设组装前通过视觉系统检查各部件。PF_ActivePart仅限于部件#1，以确保部件#1的测试不合格时，机器人可返回到送料器获取适当的部件#1。部件#1的测试合格时，PF_ActivePart用于切换为部件#2。
PF_Start语句的第1个部件ID是最初的有效部件。通常在PF_Robot回调函数结束前设定PF_ActivePart，以确保专门针对请求进给运作的部件。
本章节详细说明PF_ActivePart。

4.1.2.2 PF_Start

如上所述，可在同一送料器上同时处理最多4种部件。这可通过对各部件ID指定执行PF_Start语句的时序予以实现。
比如，部件1、2、3、4全部都使用送料器#1。最初创建各部件时，在[部件向导]中指定送料器编号。



通过PF_Start同时送入仅分配给同一送料器的部件。如下所述为同时将全部4种部件送至送料器#1时的编码。

```
PF_Start 1, 2, 3, 4
```

在本例中，部件#1为最初的有效部件。送料器最初使用部件#1的振动参数。在本例中，值“1”会被作为各回调函数的PartID参数发送，直至PF_ActivePart变为其他部件ID。

要点

PF_Start列表中的第1个部件为有效部件，直至按其他部件ID执行PF_ActivePart。

如果用户代码用于开始同一送料器上的其他部件，则会发生错误“送料器正在使用”。由PF_Status回调函数处理错误，状态值变为常数PF_D_STATUS_FEEDERINUSE_ERROR。
如下所述为发生“送料器正在使用”错误的示例。

部件1、2、3、4全部都正在使用送料器#1

```
PF_Start 1, 3, 4      ‘利用送料器#1开始部件编组
```

如果执行下述代码行，则会发生“送料器正在使用”错误

```
PF_Start 2
```

要同时将全部4种部件送至送料器#1时，还需要执行下述语句。

```
PF_Start 1, 2, 3, 4
```

最多2台机器人可同时使用同一送料器。如果PF_Start时要将同时使用大于等于2台机器人的部件送至同一送料器，则会发生“每台送料器的最大机器人数”错误。
如下所述为发生错误的示例。

部件1：使用机器人#1与送料器#1
部件3：使用机器人#2与送料器#1

部件5：使用机器人#3与送料器#1

```
PF_Start 1,3,5 <-
```

Error 7731超过控制器类型的最大同时送料器数。

4.1.2.3 读入视觉与队列

默认设定为针对送入同一送料器的所有部件获取视觉图像。也包括仅针对有效部件获取图像的机制（在下一部分对其进行说明）。各部件使用各自的照明要件（使用前灯/背光灯、移动相机、PF_Vision回调函数等）。

如果所有的部件视觉序列完成，则读入视觉结果与全部队列。

提示

为了提高效率，可对所有的部件视觉序列设定相同的CameraBrightness与CameraContrast。在PF_Start列表的最初部件的视觉序列中，RuntimeAcquire属性被设为“Stationary”。在使用同一送料器的剩余的部件视觉序列中，RuntimeAcquire被设为“None”。这样的话，排除了多余的图像获取时间。该方法以在相同照明条件下找到所有部件为前提。

4.1.2.4 PF_Robot返回值

所有的回调函数都要求返回值。通常，返回值表示操作已正常完成。（常数PF_CALLBACK_SUCCESS）返回值用于更改系统的运作方法。PF_Robot返回值用于更改主要的进程流程，并根据“使用场景”提供不同的运作。根据不同的返回值，开发人员可在各种状况下控制Part Feeding的进程流程。

下面说明PF_Robot回调函数的各返回值。

PF_CALLBACK_SUCCESS:

PF_Robot回调函数结束并且“PF_Robot = PF_CALLBACK_SUCCESS”时，系统会确认下述有效部件（请求的部件）可否在其队列中使用。可使用有效部件时，系统会再次调用PF_Robot函数。交接给PF_Robot的PartID参数为该有效部件的部件编号。可使用有效部件时，由于请求的部件已位于队列中，因此无需获取新图像或使送料器振动。不能使用有效部件时，系统会获取通过PF_Start语句执行的所有部件的新图像。视觉结果被读入各部件队列中，系统判断是否需要使送料器振动。

PF_CALLBACK_RESTART:

PF_Robot回调函数结束并且“PF_Robot = PF_CALLBACK_RESTART”时，不论有效部件队列中是否有可使用的部件，系统都会获取通过PF_Start语句执行的所有部件的新图像。视觉结果被读入所有的部件队列中，系统判断是否需要使送料器振动。PF_CALLBACK_RESTART返回值的主要目的是，从主要Part Feeding进程循环的最初，强制进行新图像的获取、队列的重新读入、重新判断与重新投放等运作。要按拾取&放置的循环获取新图像时，该返回值非常实用。比如，拾取&放置期间错误地切断周边的部件时，如果获取新图像并更新队列，则非常便利。

PF_CALLBACK_RESTART_ACTIVEPART:

PF_Robot回调函数结束并且“PF_Robot = PF_CALLBACK_RESTART_ACTIVEPART”时，系统仅获取有效部件（并非PF_Start语句中记载的所有部件）的新图像。多台机器人要共享同一送料器时，该返回值非常有用。如果使用PF_CALLBACK_RESTART_ACTIVEPART，则可防止部件在多个队列中重复。根据该返回值，仅对有效部件强制获取新图像，并且仅读入有效部件的队列。PF_Start语句中记载的所有其他部件的部件队列会被删除。下面举例说明这方面的情况。

假设送料器平台上有1种物理部件，2台机器人正在从送料器中拾取部件。在该应用中，2种部件被添加至Part Feeding对话框中。不同的机器人编号被分配各部件。此外，各部件的视觉序列带有不同的视觉校准。（相机需要针对特定的机器人进行校准。）即使部件的物理类型仅为1种，也需要创建2个逻辑工具。（每台机器人1个。）PF_Start语句中包含有双方的部件编号。针对双方的逻辑部件获取视觉，然后读入双方的队列。任一视觉序列均用于查找平台上的相同物理部件，因此队列中就会发生坐标重复现象。即使机器人#1拾取部件并从队列中删除，该部件也残留在另一台机器人的队列

中。结果，机器人#2就会尝试拾取已被机器人#1删除的部件。PF_CALLBACK_RESTART_ACTIVEPART用于确保仅将部件读入1个队列中。这样的话，就无需特别排序或特别的队列分布。有关使用场景的编程方法详细信息，请参阅以下内容。
2台机器人 - 1种部件

4.1.2.5 PF_AccessFeeder / PF_ReleaseFeeder

PF_AccessFeeder / PF_ReleaseFeeder用于通过对送料器访问的锁定与解锁，防止多机器人单一送料器系统的碰撞。2台机器人同时共享同一送料器时，需要使用这些命令。PF_AccessFeeder用于相互排他锁定。已获取锁定时，PF_AccessFeeder用于在锁定被解除之前或达到指定时间（选件）之前暂停（在轮到之前待机）任务。机器人结束送料器的使用后，由于另一台机器人要使用送料器，因此需要使用PF_ReleaseFeeder语句解除锁定。可在运动命令的“!...!”并行处理语句内使用PF_ReleaseFeeder。这样可确保一台机器人离开送料器后，另一台机器人靠近送料器。如下所述为访问送料器期间的防止机器人发生碰撞的代码。

```
PF_AccessFeeder 1
Pick = PF_QueGet(1)
PF_QueRemove (1)
Jump Pick
On gripper
Wait .25
Jump Place ! D80; PF_ReleaseFeeder 1 !
```

4.1.2.6 PF_Stop

PF_Start语句中包含有PartID参数。为单一部件系统时，PartID与送入的部件是相同的。多部件系统时，PartID为送入送料器上的部件的某个部件编号。也就是说，可指定PF_Start列表中的某个部件。但交接给PF_CycleStop回调函数的PartID为当前有效部件的部件ID，敬请注意。

4.1.2.7 PF_InitLog

PF_InitLog用于执行Part Feeding日志文件。PF_Start语句中记载的所有部件的数据被记录为文件。日志文件中的各条目包含有当前有效部件的部件ID。比如，针对当前的有效部件，记录有由PF_Robot回调函数处理的部件数。有关详细信息，请参阅以下内容。
Part Feeding日志文件

4.1.2.8 PF_QtyAdjHopperTime

PF_QtyAdjHopperTime函数用于计算供给最佳数量部件所需的料斗运作时间。根据特定时间内供给部件的数量、当前送料器平台上的某部件概数与针对部件进行“部件区域”校准期间所分配的“最佳部件数”，计算时间。可在用户代码的任意位置执行PF_QtyAdjHopperTime函数。比如，要在运作之前供给部件时，可在PF_Start语句之前执行。Part Feeding系统不能判断哪个部件组被用于送料器。包括将1种部件送入送料器，或同时将4种不同的部件送入送料器的情况等。在PF_Start之前执行了PF_QtyAdjHopperTime时，只能通过计算推测由PartID参数指定的部件为平台上的唯一部件。仅使用（其PartID的）Part Blob视觉序列。在PF_Start之后执行了PF_QtyAdjHopperTime时，Part Feeding系统可掌握送料器上有何种部件。（因为已在PF_Start语句中指定部件。）一起执行已供给PartID的Part Blob视觉序列与各自的部件序列。同时送入多个部件时，系统把各种类型的部件保持相同数量作为最佳方案。但可能不适合于部分案例。比如，也有希望部件#2的数量为部件#1的2倍的情况。在这种情况下，为了通过料斗供给所需量的部件，需要在用户代码中对由PF_QtyAdjHopperTime函数返回的计算时间进行缩放处理（乘法或除法）。

4.1.3 教程

本部分具体说明安装多部件应用/多部件与多机器人应用这两种应用的方法。多数案例只简单说明步骤，而不说明详细的执行内容。本节以用户理解新部件的创建方法、部件校准的执行方法、1台机器人1种部件应用的程序记述方法为前

提。
有关详细信息，请参阅以下内容。
试着使用

4.1.3.1 教程1：1台机器人、1台送料器、2种部件

在本教程中，2种部件被同时送至同一送料器上。部件存在物理方面的差异。使用部件#1与部件#2。送料器上带有向下固定相机。机器人拾取&放置2个部件#1，然后拾取&放置1个部件#2。循环执行该操作。

使用不同的末端夹具（输出位）拾取各部件。在该组装应用中，部件数与拾取顺序非常重要。使用PF_ActivePart进行部件#1与部件#2的切换处理。记述旨在确保最先在PF_Robot回调函数内执行机器人运作的代码。接下来通过单独的多任务执行运作，对送料器的振动看下并行处理，以提高机器人单位时间的处理能力。

1. 创建Epson RC+的新项目。

2. 对向下固定相机进行校准。

3. 通过Vision Guide创建Part Blob序列。
   部件校准期间或执行时的反馈会使用“Part Blob序列”，以决定部件的最佳投放方法。Part Blob序列用于在执行时检测各自的部件或部件群。通常，Part Blob序列中包含有单一的Blob视觉对象。Part Blob序列中需要分配给Calibration属性的相机校准。
   将Blob对象的NumberToFind属性设为“All”。将Blob的ThresholdAuto属性设为“False”（默认值）。
   在平台中配置部件，然后打开[直方图]窗口，调整ThresholdHigh与ThresholdLow，直至检测到部件。将Blob的MinArea设为部件区域MinArea的约0.9倍。部件有表里时，设定确保不论哪个方向都会检测到部件的MinArea。可分别对（送料器上的）2种部件设定单独的Part Blob序列，但通常可对所有部件，使用同一Part Blob序列。
   确认可利用Part Blob序列检测送入该送料器上的各部件。Blob对象的搜索窗口需要覆盖尽可能广范围的平台区域。但Blob对象仅限于检测部件，不检测送料器托盘自身，这点至关重要。如果利用Part Blob序列检测到部分托盘，系统则无法正确运作。

4. 通过Vision Guide创建在该送料器中运作的2种部件各自的部件序列。
   确认已针对各序列设定Calibration属性。一般来说，Geometric对象用于确定表面（第2个Geometric对象用于在需要翻转时确定部件的背面）。将视觉对象的NumberToFind属性设为“All”。

5. 显示[工具] - [上料]，然后根据“部件向导”添加新部件。
   有关部件向导的详细信息，请参阅以下内容。
   试着使用

6. 单击新部件的[校准] - [校准]按钮，开始“校准向导”。
   有关校准向导的详细使用方法，请参阅以下内容。
   校准&测试

7. 单击部件#1の[拾取] - [示教]按钮。
   通过步进运作将机器人移动至部件拾取高度，然后进行“Pick Z”示教。

8. 单击[上料] - [添加]按钮，添加部件#2。
   使用“部件向导”设定部件。

9. 单击[校准] - [校准]按钮，开始校准向导。

10. 单击部件#2の[拾取] - [示教]按钮。
    通过步进运作将机器人移动至部件拾取高度，然后进行“Pick Z”示教。

11. 关闭[上料]对话框。
    会自动创建上料模板代码（上料回调函数）。

12. 对机器人进行停止（park）与放置（place）示教，并分别命名为“park”与“place”。

13. 按如下所述修正模板代码。

Main.prg

```
Function Main
    Robot 1
    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Park

    PF_Start 1, 2
Fend
```

PartFeeding.prg

```
Global Integer numPicked ' number of parts that have been picked

Function PF_Robot(PartID As Integer) As Integer

    Integer numRequired, gripperOutput

    Select PartID
        Case 1
            numRequired = 2 ' number of parts required
            gripperOutput = 1
        Case 2
            numRequired = 1
            gripperOutput = 2
    Send
    Do
        If PF_QueLen(PartID) > 0 Then
            P0 = PF_QueGet(PartID)
            PF_QueRemove (PartID)
            Jump P0
            On gripperOutput
            Wait 0.1
            Jump Place
            Off gripperOutput
            Wait 0.1
            numPicked = numPicked + 1
        Else
            ' Not enough parts were picked
            PF_ActivePart PartID ' No change in Active Part
            PF_Robot = PF_CALLBACK_SUCCESS
            Exit Function
        EndIf
    Loop Until numPicked = numRequired

    numPicked = 0
    ' select the next Active Part
    If PartID = 1 Then
        PF_ActivePart 2
    Else
        PF_ActivePart 1
    EndIf
    PF_Robot = PF_CALLBACK_SUCCESS
Fend
```

在上述样本代码中，如果机器人完成“放置”运作并且PF_Robot函数结束，送料器则会根据需要进行振动。可重构该代码，以确保与机器人的运作并行让送料器进行振动。
要将可拾取部件事宜通知给运动任务时，使用PF_Robot回调函数（例：Function Main）。
如果可使用部件，则使用远程IO（命名为“PartsToPick1”、“PartsToPick2”）发送信号。

如果已放置好可使用的最后一个部件（本例的运作完成80%的状态），运动任务则会向PF_Robot函数发送信号并结束，然后返回一个值。根据该返回值，系统会掌握可获取新图像、振动以及通过料斗供给部件等事宜。在这里，修正教程代码，以确保可并行执行送料器运作与机器人运作。

Main.prg

```
Function Main
    Integer numToPick1, numToPick2, numPicked

    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Park

    MemOff PartsToPick1
    MemOff PartsToPick2
    numToPick1 = 2
    numToPick2 = 1

    PF_Start 1, 2

    Do
        numPicked = 0

        Do
            Wait MemSw(PartsToPick1) = On
            pick = PF_QueGet(1)
            PF_QueRemove (1)
            Jump pick
            On gripper1
            Wait 0.1
            numPicked = numPicked + 1
            If numPicked < numToPick1 And PF_QueLen(1) > 0 Then
                Jump Place
            Else
                ' Last part or no more parts available to pick
                If numPicked = numToPick1 Then
                    ' Select the next part
                    PF_ActivePart 2
                EndIf
                Jump Place ! D80; MemOff PartsToPick1 !
            EndIf
            Off gripper1
            Wait 0.1
        Loop Until numPicked = numToPick1

        numPicked = 0
        Do
            Wait MemSw(PartsToPick2) = On
            pick = PF_QueGet(2)
            PF_QueRemove (2)
            Jump pick
            On gripper2
            Wait 0.1
            numPicked = numPicked + 1
            If numPicked < numToPick2 And PF_QueLen(2) > 0 Then
                Jump Place
            Else
                ' Last part or no more parts available to pick
                If numPicked = numToPick2 Then
                    ' Select the next part
                    PF_ActivePart 1
                EndIf
```

```
                Jump Place ! D80; MemOff PartsToPick2 !
            EndIf
            Off gripper2
            Wait 0.1
        Loop Until numPicked = numToPick2
    Loop
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    Select PartID
        Case 1
            MemOn PartsToPick1
            Wait MemSw(PartsToPick1) = Off
        Case 2
            MemOn PartsToPick2
            Wait MemSw(PartsToPick2) = Off
    Send

    PF_Robot = PF_CALLBACK_SUCCESS
Fend
```

4.1.3.2 教程2：2台机器人、1台送料器、2种部件

在本教程中，2种部件被同时送至同一送料器上。部件存在物理方面的差异。使用部件#1与部件#2。送料器上带有向下固定相机。2台机器人共享同一送料器。2台机器人交互从送料器中进行拾取运作。各机器人拾取&放置各部件。机器人#1拾取1个部件#1，然后机器人#2拾取1个部件#2。
在该应用中，拾取顺序非常重要。可利用“PF_ActivePart”调换拾取顺序。记述旨在确保最先在PF_Robot回调函数内执行机器人运作的代码。在本案例中，依次进行机器人运作。一次仅1台机器人进行运作。
接下来通过单独的多任务执行运作，以提高机器人单位时间的处理能力。已修正的教程中也包含有PF_AccessFeeder / PF_ReleaseFeeder。PF_AccessFeeder / PF_ReleaseFeeder用于防止从送料器中拾取时，机器人发生碰撞。

1. 创建Epson RC+的新项目。
2. 对机器人#1用向下固定相机进行校准。
3. 对机器人#2用向下固定相机进行校准。
4. 通过Vision Guide创建部件#1的Part Blob序列。通常，Part Blob序列中包含有单一的Blob视觉对象。序列中需要针对分配给Calibration属性的机器人#1执行的相机校准。将Blob对象的NumberToFind属性设为“All”。
   将Blob的ThresholdAuto属性设为“False”（默认值）。在平台中配置部件，然后打开[直方图]窗口，调整ThresholdHigh与ThresholdLow，直至检测到部件。将Blob的MinArea设为部件区域MinArea的约0.9倍。
   部件有表里时，设定确保不论哪个方向都会检测到部件的MinArea。确认Part Blob序列可检测到部件#1。
   Blob对象的搜索窗口需要覆盖尽可能广范围的平台区域。但Blob对象仅限于检测部件，不检测送料器托盘自身，这点至关重要。
5. 通过Vision Guide创建部件#2的Part Blob序列。通常，Part Blob序列中包含有单一的Blob视觉对象。序列中需要针对分配给Calibration属性的机器人#2执行的相机校准。将Blob对象的NumberToFind属性设为“All”。
   将Blob的ThresholdAuto属性设为“False”（默认值）。
   在平台中配置部件，然后打开[直方图]窗口，调整ThresholdHigh与ThresholdLow，直至检测到部件。
   将Blob的MinArea设为部件区域MinArea的约0.9倍。
   部件有表里时，设定确保不论哪个方向都会检测到部件的MinArea。确认Part Blob序列可检测到部件#2。Blob对象的搜索窗口需要覆盖尽可能广范围的平台区域。
   但Blob对象仅限于检测部件，不检测送料器托盘自身，这点至关重要。
6. 通过Vision Guide创建用于检测部件#1的部件序列。
   请确认Calibration属性已被设为针对机器人#1执行的校准。
   一般来说，Geometric对象用于确定表面。（第2个Geometric对象用于在需要翻转时确定部件的背面。）将视觉对象的NumberToFind属性设为“All”。

7. 通过Vision Guide创建用于检测部件#2的部件序列。
   请确认Calibration属性已被设为针对机器人#2执行的校准。
   一般来说，Geometric对象用于确定表面。（第2个Geometric对象用于在需要翻转时确定部件的背面。）将视觉对象的NumberToFind属性设为“All”。
8. 显示[工具] - [上料]，然后添加部件#1的新部件。根据“部件向导”添加部件。在部件向导的第一页，确认已选择机器人#1。
   有关部件向导的详细信息，请参阅以下内容。
   试着使用
9. 单击新部件#1的[校准] - [校准]按钮，开始“校准向导”。
   有关校准向导的详细使用方法，请参阅以下内容。
   校准&测试
10. 单击部件#1の[拾取] - [示教]按钮。
    通过步进运作将机器人#1移动至部件拾取高度，然后进行“Pick Z”示教。
11. 单击[上料] - [添加]按钮，添加部件#2。使用“部件向导”设定部件。
    在部件向导的第一页，确认已选择机器人#2。
12. 单击[校准] - [校准]按钮，开始校准向导。
13. 单击部件#2の[拾取] - [示教]按钮。通过步进运作将机器人#2移动至部件拾取高度，然后进行“Pick Z”示教。
14. 关闭[上料]对话框。
    会自动创建上料模板代码（Part Feeding的回调函数）。
15. 按如下所述修正模板代码。

Main.prg

```
Function Main
    Robot 1
    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Park
    Robot 2
    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Park

    PF_Start 1, 2
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    If PF_QueLen(PartID) > 0 Then
        Select PartID
            Case 1
                Robot 1
                P0 = PF_QueGet(1)
                PF_QueRemove (1)
                Jump P0 /R
                On rbt1Gripper
                Wait 0.25
                Jump Place
                Off rbt1Gripper
                Wait 0.25
                PF_ActivePart 2
            Case 2
                Robot 2
```

```
                P0 = PF_QueGet(2)
                PF_QueRemove (2)
                Jump P0 /L
                On rbt2Gripper
                Wait 0.25
                Jump Place
                Off rbt2Gripper
                Wait 0.25
                PF_ActivePart 1
        Send

    EndIf

    PF_Robot = PF_CALLBACK_SUCCESS

Fend
```

在这里，修正样本代码，以确保通过其他多任务执行机器人运作。确保1台机器人离开送料器后，其他机器人可开始向送料器移动。通过并行运作，多台机器人共享1台送料器时，必须使用PF_AccessFeeder命令与PF_ReleaseFeeder命令，以免机器人发生碰撞。
此外，已修正的代码用于并行处理送料器的振动与机器人运作。在本教程中，可在各机器人到达各自放置位置的80%地点时，确保机器人可删除相机视野并获取图像。
另外，各机器人到达各自放置位置的80%地点时，即使另一台机器人开始向送料器移动，也可以确保安全。实际运作的百分比取决于特定状况下的机器人速度与相对定位。各机器人带有“park”点与“place”点。在本教程中，机器人#1以右手姿态从送料器中进行拾取动作，机器人#2以左手姿态从送料器中进行拾取动作。
如下所述为已修正的模板代码。

Main.prg

```
Function Main
    Robot 1
    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Park
    Robot 2
    Motor On
    Power High
    Speed 50
    Accel 50, 50
    Jump Park
    MemOff PartsToPick1
    MemOff PartsToPick2

    PF_Start 1, 2
    Xqt Robot1PickPlace
    Xqt Robot2PickPlace
Fend

Function Robot1PickPlace
    Robot 1
    Do
        Wait MemSw(PartsToPick1) = On
        PF_AccessFeeder 1
        P0 = PF_QueGet(1)
        PF_QueRemove (1)
        Jump P0 /R
        On rbt1Gripper
        Wait 0.25
        Jump Place ! D80; MemOff PartsToPick1; PF_ReleaseFeeder 1 !
        Off rbt1Gripper
        Wait 0.25
```

```
    Loop
Fend

Function Robot2PickPlace
    Robot 2
    Do
        Wait MemSw(PartsToPick2) = On
        PF_AccessFeeder 1
        P0 = PF_QueGet(2)
        PF_QueRemove (2)
        Jump P0 /L
        On rbt2Gripper
        Wait 0.25
        Jump Place ! D80; MemOff PartsToPick2; PF_ReleaseFeeder 1 !
        Off rbt2Gripper
        Wait 0.25
    Loop
Fend
```

PartFeeding.prg

```
Function PF_Robot(PartID As Integer) As Integer
    Select PartID
        Case 1
            If PF_QueLen(1) > 0 Then
                MemOn PartsToPick1
                Wait MemSw(PartsToPick1) = Off
                PF_ActivePart 2
            Else
                PF_ActivePart 1
            EndIf
        Case 2
            If PF_QueLen(2) > 0 Then
                MemOn PartsToPick2
                Wait MemSw(PartsToPick2) = Off
                PF_ActivePart 1
            Else
                PF_ActivePart 2
            EndIf
    Send
    PF_Robot = PF_CALLBACK_SUCCESS
Fend
```

4.1.4 多部件 & 多机器人的总结

下面介绍多部件、多机器人的送料器应用所需的主要概念的概述。

- **通过使用PF_Start**，可指定最多4种部件并同时将其送至同一送料器。
- **如果使用PF_ActivePart**，则可选择当前所需的部件。系统进行振动运作，将请求的部件置于可使用状态。
- 最多2台机器人可同时访问同一送料器。
  PF_AccessFeeder与**PF_ReleaseFeeder**可用于确保两台机器人安全地从送料器中拾取部件而不会发生碰撞。
- **PF_Robot返回值**用于控制主要的进程流程。
  以下返回值用于提供各种功能。
  PF_CALLBACK_SUCCESS
  可使用PF_ActivePart（请求的部件）时，系统会再次调用PF_Robot函数，而不重新获取图像或重新读入部件队列。
  不能使用PF_ActivePart时，会针对各部件获取新图像并重新读入部件队列。
  PF_CALLBACK_RESTART
  系统针对通过PF_Start语句执行的各部件获取新图像并重新读入所有的部件队列。

PF_CALLBACK_RESTART_ACTIVEPART
系统仅针对PF_ActivePart获取新图像。读入PF_ActivePart队列与视觉结果，然后删除所有其他部件队列。

4.2 平台的类型

本章节说明平台的类型。

4.2.1 标准平台的类型

IF-80、IF-240、IF-380、IF-530中包括平面 (Flat)、防粘贴 (Anti-stick)、防滚动 (Anti-roll) 3 种标准平台。不论哪种平台，均可使用白色与黑色。
IF-A1520、IF-A2330中包括平面 (Flat)、防滚动 (Anti-roll) 2种标准平台。不论哪种平台，均为白色。

4.2.1.1 平台的颜色

在几乎所有的应用中，通过使用送料器的内置背光灯，都可以实现最佳的图像处理。使用背光灯时，应使用白色半透明的平台。背光灯用于对部件外圈投影。
无法通过视觉系统看到部件表面的特征。另外，要将对比度最大化时，也可以使用黑色平台与自定义前灯予以实现。但一般来说，系统都被设计为与背光灯选件一起使用。点亮背光灯，检测部件。(右上方、|上下、边角方向等。)
使用背光灯时，透明对象或半透明对象的对比度可能会降低。要针对透明部件获得对比度时，白色以外的背光灯可能会有效果。

4.2.1.2 平台的材质

根据送料器型号，可在防带电 (ESD) 或医疗应用场景中使用特殊材料。有关详细信息，请参阅各送料器的硬件手册。

4.2.1.3 标准平台的使用方法

- **平面：**
  具有稳定姿势的部件可使用平面平台。需要部件在送料器振动后较短时间内达到静止状态。由于是多品种少量生产，几乎所有的应用都是用平面平台。

外观

截面图

a	部件
b	平台

Epson提供的平台满足平坦度与平行度规格，以确保下表所述的拾取精度。

	IF-80	IF-240	IF-380 / IF-530	IF-A1520 / IF-A2330
表面的平坦度 [mm]	0.1	0.2	0.6	0.3

	IF-80	IF-240	IF-380 / IF-530	IF-A1520 / IF-A2330
表面与基准之间的平行度［mm］	0.1	0.5	0.6	0.6

- **防粘贴：**
  防粘贴平台带有窄槽，以用于减少表面的接触阻力。这样的话，可降低摩擦力并改善送料器表面的部件形态。防粘贴平台非常适合于因运动摩擦（滑动摩擦或动摩擦）而导致难以分散的部件。

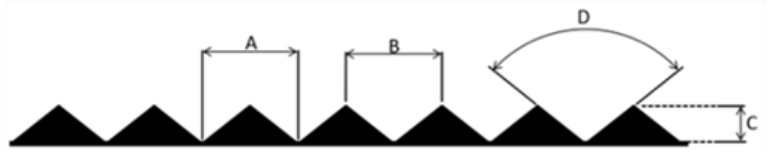
表面

截面图

a	部件
b	平台

	A	B	C	D
IF-80	0.4	0.4	0.2	90

小型部件用标准防粘贴平台的结构

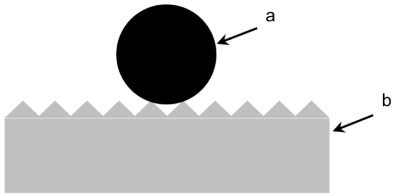


	A	B	C
IF-240	0.7	1.3	0.5

大型部件用标准防粘贴平台的结构

- **防滚动：**
  防滚动平台带有经过机械加工的结构面，可使易滚动部件变为静止状态。防滚动平台非常适合于供给圆柱形成分。防滚动平台还有通过防止部件滚动来缩短稳定时间的特点。

表面

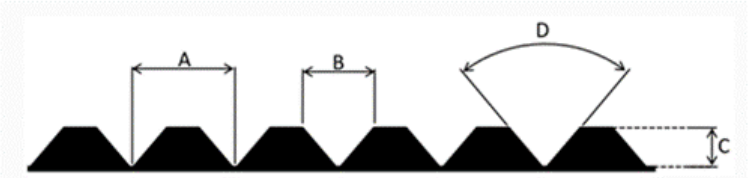


截面图

a	部件
b	平台

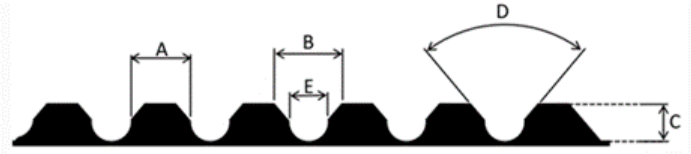
	A	B	C	D	适合的部件
IF-80	1.25	1	0.5	90	ø 0.7mm - ø 1.5mm
IF-80	2.75	2.5	1.25	90	ø 1.5mm - ø 3.5mm
IF-240	3	2.5	1.25	90	ø 1.7mm - ø 3.5mm
IF-240	5.5	5	2.5	90	ø 3.5mm - ø 7mm
IF-240	10.5	10	5	90	ø 7mm - ø 14mm
IF-380	3	2.5	0.722	120	ø 3mm - ø 5mm
IF-380	5.5	5	1.443	120	ø 5mm - ø 10mm
IF-530	6.5	6	1.732	120	ø 6mm - ø 12mm
IF-A1520	7	5	2	90	ø 3.5mm - ø 7mm
IF-A2330	7	5	2	90	ø 3.5mm - ø 7mm

小型部件用标准防滚动平台的结构



	A	B	C	D	E	适合的部件
IF-380	10.5	12	5.31	120	2	ø 10mm - ø 24mm
IF-530	12.5	14	4.9	120	4	ø 12mm - ø 28mm

大型部件用标准防滚动平台的结构



4.2.2 自定义平台

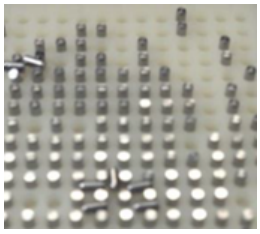
要点

请客户设计并制作自定义平台。

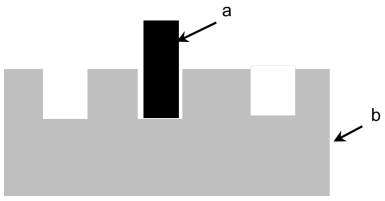
4.2.2.1 自定义平台的基本设计

自定义平台包括孔、长槽、定位槽3种基本设计。自定义平台的目的在于利用孔、长槽或定位槽排列部件，以便获得预期的循环时间。部件的自然静止位置与拾取方向不一致时，如果要正确配置部件，则需要使用自定义平台。
如下所述为自定义平台一般概述。

- **孔：**



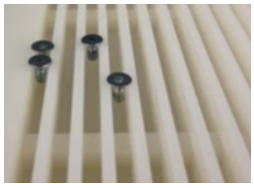
表面



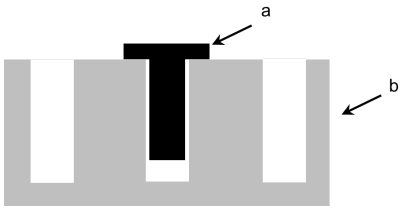
截面图

a	部件
b	平台

- **长槽：**



表面

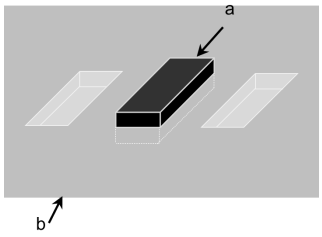


截面图

a	部件
b	平台

- **定位槽：**

表面



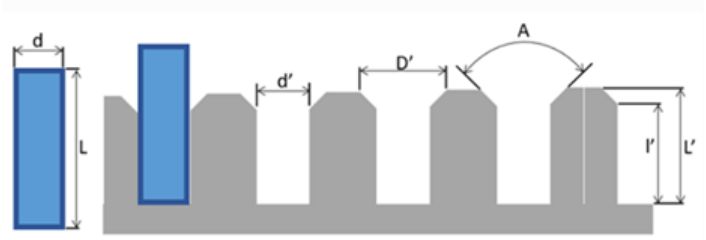
斜视图

a	部件
b	平台

4.2.2.2 自定义平台设计指南

- **孔：**
  要供给圆柱形部件并置于直立状态时，孔加工是非常便利的手段。
  在带孔平台的基本设计中，请考虑下述事项。
  - 一般来说采用单纯的结构。
  - 孔直径（d’）是板中最重要的尺寸。
    需要足够大的直径，以确保将部件笔直地立在孔内侧。为最大直径3.5mm的部件时，在最大部件直径（d）加上0.05mm。但如果是直径超过3.5mm的大型部件，或圆锥形部件等圆柱形以外的部件时，需要进一步增大直径。
  - 需要开出具有给足够深度的孔（l’），以便在必要时可沿着壁面引导部件。
    工件被放到底部时，不需要较长的导件。
  - 最好在大于等于部件高度（L）3分之1（如果可能，为2分之1）的位置引导部件，以确保尽可能使部件保持笔直状态。
  - 另外，也请注意板（L’）上的部件剩余高度。
  - 倒角（A）对于将部件放入孔中是不可或缺的。
    大多数情况下的倒角角度为60°。
  - 平台的重量与标准平面平台相同。如果重量发生大幅度变化，共振频率会随之发生变化，这可能会对送料器的运作产生影响。在这种情况下，可通过调整送料器的运作频率进行应对。

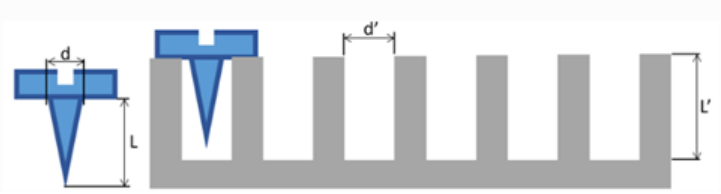
	D’	d’	l’	A
IF-80	>0.5*L	d+0.05mm	0.5*L	60
IF-240	>0.5*L	d+0.1mm	0.5*L	60
IF-380	>0.5*L	d+0.5mm	0.5*L	60
IF-530	>0.5*L	d+1.0mm	0.5*L	60
IF-A1520	>0.5*L	d+1.0mm	0.5*L	60
IF-A2330	>0.5*L	d+1.5mm	0.5*L	60



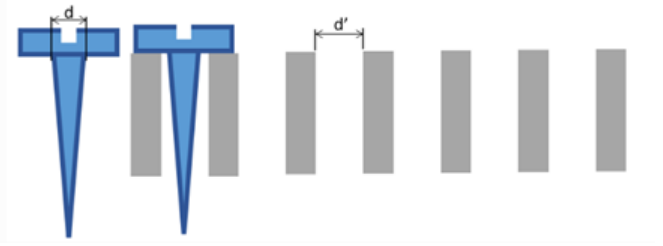
- **长槽**

已设计长槽的平台用于供给垂直安装的螺钉类部件。
在带长槽平台的基本设计中，请考虑下述事项。

- 一般来说采用单纯的结构。
- 如果使用不带贯通长槽的平台，则可供给最长60mm的部件。部件长度更大时，请在设计长槽时考虑板厚。
- 根据组件直径（d）确定插槽的宽度（d’）。一般来说，在最大直径中加上0.05mm ～ 0.1mm（考虑容许误差）。也请考虑长槽的宽度公差。这会因加工而异，对于所有尺寸的送料器来说并不相同。推荐采用0/+公差。
- 工件未被放到底部，因此大多数情况下，长槽的深度（L’）并不重要。因此，需要具有足够的大小，以免部件倾斜接触底部。通常适用0/+公差。



最大长度为60mm的大型部件时，无需贯通长槽。



超过60mm的部件时，设计贯通长槽时请考虑板厚。

	d’	L’
IF-80	d+0.05mm	L’ >L
IF-240	d+0.1mm	L’ >L

	d’	L’
IF-380	d+0.5mm	-
IF-530	d+1.0mm	-
IF-A1520	d+1.0mm	L’>L
IF-A2330	d+0.5mm	-

- 长槽贯通平台底部时，需要使用“内部扩散板”，使操作员直接看不到LED背光灯以及背光灯的光线不直接进入相机内。“内部扩散板”被配置在背光灯上。
- 平台的重量与标准平面平台相同。如果重量发生大幅度变化，共振频率会随之发生变化，这可能会对送料器的运作产生影响。在这种情况下，可通过调整送料器的运作频率进行应对。

- **定位槽**

  在带定位槽平台的基本设计中，请考虑下述事项。
  - 在平台上设计定位槽，以确保机器人可简单地抓取部件。理想情况为部件的平面朝上，以确保可利用真空夹爪进行拾取动作。
  - 无需按定位槽完整地排列（定位）部件。视觉系统的目的是在2D平面中检测到部件的坐标与 旋转角。如果平台的结构比较单纯，则可低成本地制作平台。另外，一般来说平台设计越单纯，越会有好的效果。

上面所述不过是一般意义上的指南。需要根据实际情况进行应对。

4.2.2.3 平台的容许重量

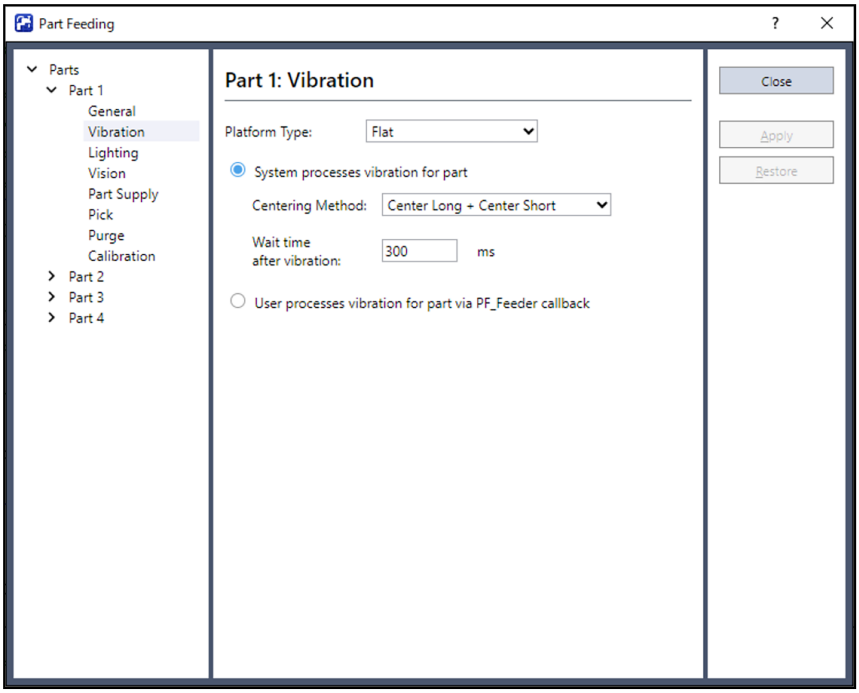
	平台的最大重量 **1	部件的最大重量 **2
IF-80	150 g	50 g
IF-240	800 g	400 g
IF-380	4 kg	1.5 kg
IF-530	5 kg	2 kg
IF-A1520	950 kg	400 kg
IF-A2330	2.5 kg	1.5 kg

**1
平台表示在送料器上部振动的部分（不含部件）。平台由板部分和机架部分构成。

**2
表示可搭载在平台上的部件的最大重量。

4.2.3 平台的选择

在Epson RC+ 8.0菜单 - [工具] - [上料] - [振动]中选择平台类型。有关详细信息，请参阅以下内容。
振动

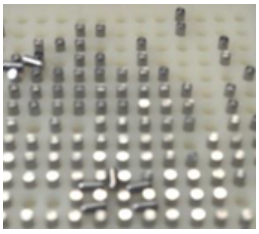


如果选择标准平台（平面、防粘贴、防滚动），则可选择让系统处理送料器振动，或通过PF_Feeder回调函数处理送料器振动。通常由系统处理送料器振动为最佳选择。
如果在振动页面中选择“通过PF_Feeder callback控制振动”，用户则可独自记述送料器的振动控制。系统以“state”参数的形式，在PF_Feeder回调函数中提供推荐的送料器振动类型。有关这些内容，将在下一章节进行详细说明。

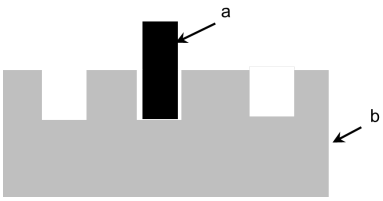
如果选择自定义平台（长槽、孔、定位槽），用户则需要通过PF_Feeder回调函数处理部件振动。自定义平台时，不判断系统推荐的送料器振动类型。（这是因为最佳振动类型因平台的加工而异。）

4.2.3.1 自定义平台处理的程序示例

在带孔自定义平台中，将销配置在纵向上。



表面



截面图

a	部件
b	平台

用户要执行视觉处理并重新加载部件坐标队列时，选择菜单 - [工具] - [上料] - [部件] - [视觉]中的[通过PF_Vision callback进行视觉处理]。在本例中，由系统进行视觉处理。
在PF_Robot回调函数内执行机器人的拾取&放置。平台类型选择“孔”。

由于这是自定义平台，因此仅可选择“通过PF_Feeder callback控制振动”。由部件检测序列检测销的上部，坐标会被自动加载至部件坐标队列中。
在本例中，没有查找背面部件的视觉对象。
如果视觉系统获取图像并将表面部件加载至部件坐标队列中，则会调用PF_Feeder回调函数。用户代码用于判断并执行在PF_Feeder回调函数内使送料器进行振动的方法。
作为自变量“NumFrontParts”，表面部件的数量被交接给PF_Feeder回调函数。在本例中，自变量“NumFrontParts”大于“0”时，由于机器人可取放部件，因此无需送料器振动。在这种情况下，需要将PF_Feeder回调函数的返回值设为“PF_CALLBACK_SUCCESS”。
该返回值用于指示系统调用PF_Robot回调函数。“NumFrontParts”等于“0”时，样本代码用于针对部件Blob序列执行VRun，判断部件是否处于聚集状态，或者根本没有部件。如果通过部件Blob序列找不到部件，则会将料斗设为ON，并向送料器供给部件。如果通过部件Blob序列找到部件，送料器则会进行翻转运作，先向前移动，然后向后移动，以确保销落入孔中。
每进行一次送料器振动，系统都需要重新获取新的视觉图像。
因此，将返回值设为“PF_CALLBACK_RESTART”。系统重新获取新的图像并重新加载部件队列后，会再次调用PF_Feeder回调函数。用户代码用于判断是否需要进一步进行送料器运作。

提示

翻转、长时间前移、短时间后移是自定义平台的典型有效运作。
有关详细信息，请参阅以下内容。
程序示例 5.2

要点

平台类型为孔、插槽或定位槽时，会将常数PF_FEEDER_UNKNOWN交接给PF_Feeder回调函数的“state”自变量。这是因为自定义平台时，系统无法确定适当的送料器运作。
有关详细信息，请参阅以下内容。
程序示例 5.2

```
Function PF_Feeder(PartID As Integer, NumFrontParts As Integer, NumBackParts As
Integer, state As Integer) As Integer

    ' 带孔结构平台示例 state = PF_FEEDER_UNKNOWN

    Integer PFControlReturnVal
    Integer numFound

    Select True

        ' 可拾取时
        Case NumFrontParts > 0
            ' 因有可拾取的部件而调用PF_Robot
            PF_Feeder = PF_CALLBACK_SUCCESS

        ' 没有表面方向的部件，但有背面方向的部件时
        Case NumFrontParts = 0 And NumBackParts <> 0

            ' 翻转、前移、后移
            PF_Flip PartID, 500
            PF_Shift PartID, PF_SHIFT_FORWARD, 1000
            PF_Shift PartID, PF_SHIFT_BACKWARD, 300

            PF_Feeder = PF_CALLBACK_RESTART ' 重启后，重新获取图像

        ' 找不到表面方向或背面方向的部件时
```

```
        ' 部件聚集或托盘上根本没有
        ' 通过Part Blog序列重新获取图像并进行判断
        Case NumFrontParts = 0 And NumBackParts = 0

            PF_Backlight 1, On ' 背光灯 On
            VRun PartBlob ' 获取图像
            PF_Backlight 1, Off ' 背光灯 Off
            VGet PartBlob.Blob01.NumberFound, numFound ' 是否找到Blob?

            If numFound > 0 Then ' 发现部件聚集时

                ' 翻转、前移、后移
                PF_Flip PartID, 500
                PF_Shift PartID, PF_SHIFT_FORWARD, 1000
                PF_Shift PartID, PF_SHIFT_BACKWARD, 300

            Else ' 找不到部件时

                '调用部件供给用Control回调函数
                PFControlReturnVal = PF_Control(PartID, PF_CONTROL_SUPPLY_FIRST)

            EndIf

            PF_Feeder = PF_CALLBACK_RESTART ' 重启后，重新获取图像
    Send

Fend
```

4.2.3.2 使用PF_Feeder回调函数的标准平面平台示例

使用标准的平面平台。通常，系统可对平面平台的振动进行最佳处理。
但本例情况下用于判断是否需要进行特殊的振动处理。平台类型为“平面”、“防粘贴”或“防滚动”时，系统也可以最适当地让部件进行振动。通过自变量“state”，系统判断被提供给PF_Feeder回调函数。“PartFeed.inc”文件将几种状态定义为常数。
比如，常数“PF_FEEDER_PICKOK”表示机器人可拾取部件。作为其他示例，系统判断翻转部件为最佳运作时，会将常数“PF_FEEDER_FLIP”交接给PF_Feeder回调函数。可使用PF_Feeder回调函数的自变量“state”与送料器控制命令，创建替代系统处理的独自处理。
下述样本所述为使用自变量“state”执行推荐操作的方法的基本概念。本例并不包括全部内容。
有关更完整的示例，请参阅以下内容。
程序示例 5.1

```
Function PF_Feeder(PartID As Integer, NumFrontParts As Integer, NumBackParts As
Integer, state As Integer) As Integer
    Integer PFControlReturnVal, PFStatusReturnVal
    Boolean PFPurgeStatus

    Select state

        Case PF_FEEDER_PICKOK
            PF_Feeder = PF_CALLBACK_SUCCESS ' 因有可拾取的部件

            ' 而调用PF_Robot

        Case PF_FEEDER_SUPPLY
            ' 通过料斗供给
            PFControlReturnVal = PF_Control(PartID, PF_CONTROL_SUPPLY_FIRST)
            PF_CenterByShift PartID ' 通过料斗供给后，对部件进行定芯
            PF_Feeder = PF_CALLBACK_RESTART ' 重启后，重新获取图像

        Case PF_FEEDER_FLIP
            PF_Flip PartID
```

```
            PF_Feeder = PF_CALLBACK_RESTART ' 重启后，重新获取图像

        Case PF_FEEDER_CENTER_FLIP
            PF_Center PartID, PF_CENTER_LONG_AXIS, 900
            PF_Center PartID, PF_CENTER_SHORT_AXIS
            PF_Flip PartID
            PF_Feeder = PF_CALLBACK_RESTART ' 重启后，重新获取图像

        Case PF_FEEDER_HOPPER_EMPTY
            ' 向用户发出料斗已空的通知
            PFStatusReturnVal = PF_Status(PartID, PF_STATUS_NOPART)
            ' 通过料斗供给部件
            PFControlReturnVal = PF_Control(PartID, PF_CONTROL_SUPPLY_FIRST)
            PF_Center PartID, PF_CENTER_LONG_AXIS
            ' 定芯、翻转与部件分离
            PF_Center PartID, PF_CENTER_SHORT_AXIS
            PF_Flip PartID
            PF_Feeder = PF_CALLBACK_RESTART ' 重启后，重新获取图像

        Case PF_FEEDER_WRONGPART
            If PF_Info(PartID, PF_INFO_ID_OBJECT_PURGE_ENABLED) Then
                ' 启用清除时，使用Vision反馈进行清除
                ' 各清除尝试运作持续1500ms
                ' 在平台上留下3个部件
                ' 进行5次重新尝试运作
                PFPurgeStatus = PF_Purge(1, 2, 1500, 3, 5)
                If PFPurgeStatus = False Then
                    Print "Purge was not successful"
                    Quit All
                EndIf
            Else ' 向用户发出送料器上也许有错误部件的通知
                Print "Wrong part may be on the feeder"
                Quit All
            EndIf

    Send
Fend
```

5. 故障排除

5.1 故障排除一览

5.1.1 不了解送料器中设定的IP地址

1. 对IP地址进行初始化。
   有关步骤，请参阅以下内容。
   “Epson RC+ 8.0选件 Part Feeding 8.0 IF-***篇 - IP地址的初始化”
   ***：送料器机型名称（IF-80、IF-240或IF-380/530） “Epson RC+ 8.0选件 Part Feeding 8.0 IF-A1520 & IF-A2330篇 - 重置”
2. 指定已进行初始化的IP地址并连接至送料器。
   然后更改IP地址。
   有关步骤，请参阅以下内容。
   操作入门

5.1.2 送料器不振动或振动较弱

- 送料器的LED显示是点亮还是熄灭？
  未点亮或熄灭时，可能是未供电。
- Epson RC+上显示信息时，请采取各错误信息的对策措施。
- 可能是送料器校准失败。
  请再次执行送料器校准。
  有关执行方法，请参阅以下内容。
  校准&测试
  即使这样仍不正常运作时，请加载默认值并再次执行校准。

为上述以外情况时，可能是送料器发生故障。请咨询销售商。

5.1.3 送料器上的部件运作不顺畅或有偏移

- 可能是送料器安装台架的刚性较低，无法正常地将振动能量传递至平台。请更改为刚性较高的台架。
- 可能是送料器校准失败。
  请再次执行送料器校准。
  有关执行方法，请参阅以下内容。
  校准&测试
  即使这样仍不正常运作时，请加载默认值并再次执行校准。
- 可能是平台因长时间使用而导致磨损。
  平台属于耗材。请更换平台。

为上述以外情况时，可能是送料器发生故障。请咨询销售商。

5.1.4 料斗不振动

- 料斗购自销售商时，请确认料斗与送料器的连接状况。
  有关详细信息，请参阅以下内容。
  送料器与料斗
- 请确认是否正确进行与料斗之间的通信设定。
- 请确认是否正确进行料斗操作编程。
  有关确认方法，请参阅以下内容。
  PF_Control

5.1.5 平台已堆满部件

- 可能是1次料斗运作导致投放至送料器的部件数过多。请适当调整料斗的投放数量。

5.1.6 平台上没有部件

- 请确认部件是否在料斗中。
- 可能是来自料斗的部件未正常投放。
  请调整料斗的投放数量。
- 料斗购自销售商时，请确认料斗与送料器的连接状况。

5.1.7 不了解备份的获取方法

- 请参阅以下内容。
  控制器的备份和恢复

5.2 故障受理表

[Basics] month/date/year:

Your company	Department
Name	TEL : E-MAIL :
Manipulator model/Serial number /	Controller model name/Serial number /
Date of trouble occurred	Occasion Tooling/during production others ( )

[Troubleshooting report]

No.	Check item	Countermeasure (summary)	Result	Notes
1	Vision error is occurred?	Cancel vision error then try reactivating the feeder system.	Trouble solved Yes/No Not applicable	
2	Feeder error occurred during Epson RC+ operation.	Refer to *Errors that Occur While Using Epson RC+* section and try resolve the obstacle. Check the countermeasure is working by testing feeder communication from Epson RC+.	Trouble solved Yes/No Not applicable	
3	Feeder error 2582 occurred.	Perform measures of troubleshooting in the manual and try resolve the obstacles. Check the countermeasure is working by testing feeder communication from Epson RC+.	Trouble solved Yes/No Not applicable	If error occurs even after taking all the measures described in 1 to 3, there is a problem with the feeder body. Try 4 and after.
4	Epson RC+LED (Power and S-Power) of the feeder does not light up.	Check for the feeder power supply.	Trouble solved Yes/No Not applicable	If there is no problem with the feeder power supply, there is a problem with the feeder body. Try 5 and after.
5	Backlight will not light up.	Make sure that the backlight is set to "red" or "white" and that the brightness is not set to 0% (for IF-A series). Exchange the backlight then try testing backlight with Epson RC+. (other than the above models)	Trouble solved Yes/No Not applicable	If not light up even after backlight test, there is a problem with the feeder body. Try 6 and after.
6	In other case	Test calibration from Epson RC+ and check the motion of the feeder.	Trouble solved Yes/No Not applicable	If not vibrate as configured, there is a problem with the feeder body.

[Other notices, questions for Epson]
