EPSON

Epson RC+ 8.0 選配件
Part Feeding 8.0
導入&軟體篇

翻譯版

Rev.4
TCM259S7760F

目錄

1. 前言

1.1 前言

感謝您選購本公司的機器人產品。
本手冊記載了正確使用Epson RC+ Part Feeding選配件所需的必要事項。
在安裝機器人系統之前，請務必詳閱本手冊及其他相關手冊。
請將本手冊放在方便隨時取用的地方。

所有機器人產品都經過嚴格的測試和檢查，以確保性能符合我們的標準。但請注意，如果超出手冊中所描述的使用條件來使用我們的機器人系統，產品的基本功能可能無法正常發揮。

本手冊的內容包括我們能夠預見到的危險和問題。請務必遵守本文檔中所述的安全注意事項，以確保安全並正確的使用我們的機器人系統。

1.2 商標

Microsoft、Windows、Windows標誌、Visual Basic和Visual C++是Microsoft Corporation在美國和/或其他國家的註冊商標或商標。其他的公司名稱、品牌名稱及產品名稱均為其各公司的註冊商標或商標。

1.3 本手冊中的商標註釋

Microsoft® Windows® 10作業系統
Microsoft® Windows® 11作業系統
於本手冊中，Windows 10、Windows 11分別表示上述作業系統。某些情況下，Windows意指Windows 10與Windows 11。

1.4 注意事項

本使用手冊的部分或全部內容，均不得未經許可複製或轉載。
本文件中記載的內容，將來可能會未經預告而變更。
若您發現本文件內容有誤或有任何疑問，敬請聯絡我們。

1.5 製造商

SEIKO EPSON CORPORATION

1.6 聯絡方式

關於詳細聯絡方式，請參閱以下手冊中的「供應商」。
「安全手冊」

1.7 閱讀本手冊之前

以下為閱讀本手冊之前須了解的資訊。

Epson RC+ 8.0的安裝資料夾

Epson RC+ 8.0的安裝資料夾路徑可以更改為任何位置。本手冊假定Epson RC+ 8.0安裝在`C:\EpsonRC80`上進行說明。

2. 導入篇

2.1 前言

2.1.1 關於Part Feeding

Epson RC+ 8.0的Part Feeding選配件（以下簡稱Part Feeding選配件）可讓您輕鬆開發一套透過送料器分離零件、由機器人從送料器取出零件的系統。

2.1.1.1 背景

為因應產品壽命縮短、多樣化、JIT（小批量、短交期）等要求的趨勢，製造業的生產形態正在多樣化發展中。另一方面，由於持續上漲的薪資、製造現場人力短缺、勞動者高齡化等因素，依賴人力的製造模式變得越來越困難。此外，如何實現能夠因應多樣化的彈性（靈活性）也成為一大課題。
先來關注自動化生產線組成元素之一的零件供應吧。由於無法讓生產量超過零件供應能力，零件供應是生產設備的重要元素。以零件供應方式而言，主流上會採用價格相對便宜且選擇豐富的振動送料器。然而，振動盤需要根據供應零件進行定製，每次更換零件時都需要手動更換振動盤，並且須由專業技術人員進行調整。這需要高階的工程能力和豐富的經驗，因此難以因應多樣化和短交期的需求。
近年來，「智慧型零件送料器」（以下簡稱送料器）作為解決振動送料器缺點的裝置應運而生。該裝置只需變更送料器的設定，就能輕鬆因應各種供應零件。市場上，也出現結合了該送料器與機器人（用於處理識別零件的影像以及已識別的零件）的產品。
這類產品僅允許從機器人系統執行送料器的個別動作，而送料器的調整、操作、影像處理及機器人對零件的處理，都需由您自行設計。若要全面地規劃及設計零件的供應，需要時間和經驗。例如，光是零件放入數量和機器人拾取位置等因素，就會大幅影響週期時間。如果設計不當，即使是高性能、高功能的送料器，也無法充分發揮其功能和性能，無法提升週期時間的效率。如果是傳統產品，要讓任何人都能立即使用送料器是不可能的。

2.1.1.2 導入Part Feeding選配件的優點

導入Part Feeding選配件具有以下優點：

- 可完全整合送料器、視覺系統、機器人
  以整套方式提供零件供應所需要素。送料器、機器人、視覺系統已完全整合。相較於分別準備和評估這些設備，導入時可節省許多麻煩。

- 可減少設備開發及啟動的工時
  送料器、視覺系統的運作會自動執行。您無需進行程式設計。機器人運作需由您編寫，但只需填入範本程式碼即可。送料器、視覺系統與機器人運作的同步控制會自動執行。採取最低限度的程式設計就能進行設備開發。也能輕鬆整合到您目前使用的環境中。送料器、機器人、視覺系統的設定可透過GUI輕鬆完成。

- 可削減停機時間、運行成本
  Part Feeding選配件可因應各種零件。無需因零件切換而變更設備，可縮短生產時的停機時間。此外，也不需要再製作新的設備，可抑制長期運行成本。

2.1.1.3 Part Feeding選配件的功能

透過以下代表性功能，即可輕鬆地充分發揮送料器的功能和性能。並且同時實現高效率的零件供應系統。

- 與送料器的通訊介面
  用於設定和控制送料器的通訊程式已整合於本系統中。您無需為了通訊而進行程式設計。
- 送料器的自動調整功能
  系統內建了根據零件自動調整送料器參數（振幅、振動時間等）的功能。您只需依照簡單的步驟操作，即使沒有送料器相關知識也能輕鬆進行調整作業。
- 送料器的控制演算法
  系統內建了控制送料器的演算法。此演算法旨在計算如何盡可能地縮短機器人拾取零件的所需時間。即使您沒有特別留意，也能實現高效率的運作。

- 掌握機器人、送料器運作狀況的週期時間日誌輸出功能
  系統內建了將機器人、送料器、視覺系統的運作時間輸出至檔案的功能。透過變更參數設定、擷取並分析日誌，運作起來即可更加有效率。使用此功能時，需要將安裝了Epson RC+ 8.0的PC連接至控制器。
- 多台送料器運作
  最多可將4台送料器連接至1個控制器來進行控制。T/VT系列最多可控制2台。要讓多台送料器的運作相互配合時，也可以只使用1個控制器進行控制，讓程式設計變得簡單。
- 多種零件運作
  最多可同時將4種零件放入1台送料器進行處理。可減少送料器的設置台數，從而實現低成本化和空間節省。在多種零件運作中，每1台送料器最多可使用2台機器人。
- 清除動作
  系統內建了排出送料器上之零件的功能。可用於品項切換時自動排出送料器上的零件，或排出不良零件及過量放入的零件。IF-80配備了用於清除動作的平台以及用於存放排出零件的容器。

2.1.2 必備的Epson RC+ 8.0基本知識

Part Feeding選配件是以Epson RC+ 8.0環境為核心的選配件。
若要使用Part Feeding選配件，則需要具備Epson RC+ 8.0開發環境、EPSON機器人及Epson RC+ 8.0選配件Vision Guide 8.0的相關知識。本手冊內容是針對具備以下知識之人員的說明：

- Epson RC+ 8.0專案管理的概念與使用方法
- 在Epson RC+ 8.0中建立和編輯SPEL+程式的方法
- 從Run視窗執行SPEL+程式的方法
- SPEL+的基本語言結構、功能與使用方法
- Vision Guide 8.0的功能與使用方法

2.1.3 相關手冊

使用Part Feeding選配件時，請參閱以下手冊。

- 「Epson RC+ 8.0選配件 Part Feeding 8.0 IF-***篇」
  ***：送料器機型名稱（IF-80、IF-240或IF-380/530）
  「Epson RC+ 8.0選配件 Part Feeding 8.0 IF-A1520 & IF-A2330篇」
  記載了各送料器的使用說明。
- 「Epson RC+ 8.0選配件 Part Feeding 8.0 Hopper篇」
  記載了料斗的使用說明。
- 「Epson RC+ 8.0使用指南」
  記載了EPSON機器人控制器系統的使用說明。
- 「SPEL+語言參考」
  記載了SPEL+語言的命令說明。
- 各機器人手冊
  記載了與機器人相關的各種說明。

2.1.4 關於符號

本文中使用了數種標示來說明重要事項。各標示具有以下含義：

⚠ 警告

此符號代表若不正確遵守相關指示，可能會有重傷或死亡的危險。

⚠ 警告

此符號代表若不正確遵守相關指示，可能會有導致人員觸電、受傷的危險。

⚠ 注意

此符號代表若不正確遵守相關指示，可能會有人員受傷或設備及設施受損的危險。

提示

提供操作機器人系統時須遵照的重要資訊。

TIP

提供簡化或替代操作方式的建議。

2.2 安全

使用前請詳閱本手冊，並正確使用。
請將本手冊放在方便隨時取用的地方。

2.2.1 安全注意事項

⚠ 警告

- 請勿將本產品用於確保安全的目的。

  此符號代表若不正確遵守相關指示，可能會有重傷或死亡的危險。
- 請依照手冊中記載的使用條件使用本產品。若在不符合使用條件的環境中使用本產品，不僅會縮短產品壽命，還可能引發嚴重的安全問題。

⚠ 注意

- 請透過本公司產品的供應商購買送料器。
- 請透過本公司產品的供應商購買攝影機、攝影機電纜。

從本公司產品的供應商以外之處所購買的產品，則不在保固範圍內。

2.2.2 機器人的安全

操作機器人或其他自動裝置時，請將安全視為最優先的考量項目。控制器和Epson RC+ 8.0內建了許多安全功能。可使用緊急停止和安全門輸入等各種安全功能。設計機器人單元時，請使用這些安全功能。
關於安全資訊和指南，請參閱以下手冊。
「Epson RC+ +8.0 User's Guide - 安全」

2.2.3 視覺系統的安全

關於視覺系統的安全，請參閱以下手冊。
「Vision Guide 8.0 Hardware & Setup- 安全性注意事項」

2.2.4 送料器的安全

關於送料器的安全，請參閱以下任一手冊。

- 「Epson RC+ 8.0選配件 Part Feeding 8.0 IF-80篇 - 安全注意事項」
- 「Epson RC+ 8.0選配件 Part Feeding 8.0 IF-240篇 - 安全注意事項」
- 「Epson RC+ 8.0選配件 Part Feeding 8.0 IF-380篇 IF-530篇 - 安全注意事項」
- 「Epson RC 8.0選配件 Part Feeding 8.0 IF-A1520 & IF-A2330篇 - 安全」
- 「安全手冊(Part Feeding)」

2.2.5 料斗的安全

關於料斗的安全相關資訊，請參閱以下手冊。
「Epson RC+ 8.0選配件 Part Feeding 8.0 Hopper篇」

2.3 用語的定義

以下說明本節所用的關鍵用語。

硬體

用語	說明
送料器	透過振動分離散裝零件，藉此讓機器人更方便搬運零件的設備。
平台	送料器的組成零件之一，屬於用於承接零件的部分。
零件	交由機器人處理的零件。請由您自行準備。
追加供料	為確保送料器上的零件隨時保持在最佳數量，而將零件從料斗追加到送料器上的供料方式。
拾取完後供料	在將送料器上的所有零件拾取完後，再將零件從料斗放入送料器的供料方式。
全面拾取	針對送料器上所有可拾取的分散零件進行拾取。
部分拾取	針對送料器上位於特定區域的分散零件進行拾取。Part Feeding選配件可從送料器上定義的4個區域中選擇要拾取的區域。

用語	說明
自訂照明	由您準備的照明。用於識別送料器的背光無法識別之零件，以及識別零件的姿態（如正反面）。
料斗	向送料器平台供應零件的裝置。
清除	意指排出送料器中剩餘的零件。
清除閘門	用於排出送料器上剩餘零件的開關門。 與送料器連接後，可透過命令控制開關。 可用於多品項小批量生產中的零件品項切換以及不良零件的排出。
送料器校準	用於調整送料器參數以確保零件能在送料器上適當移動的作業。
多台送料器	將多台送料器連接到1個控制器的配置。本選配件最多可支援4台送料器。
多種零件	在1台送料器中放入多種零件同時進行處理的配置。本選配件最多支援32個零件。

軟體

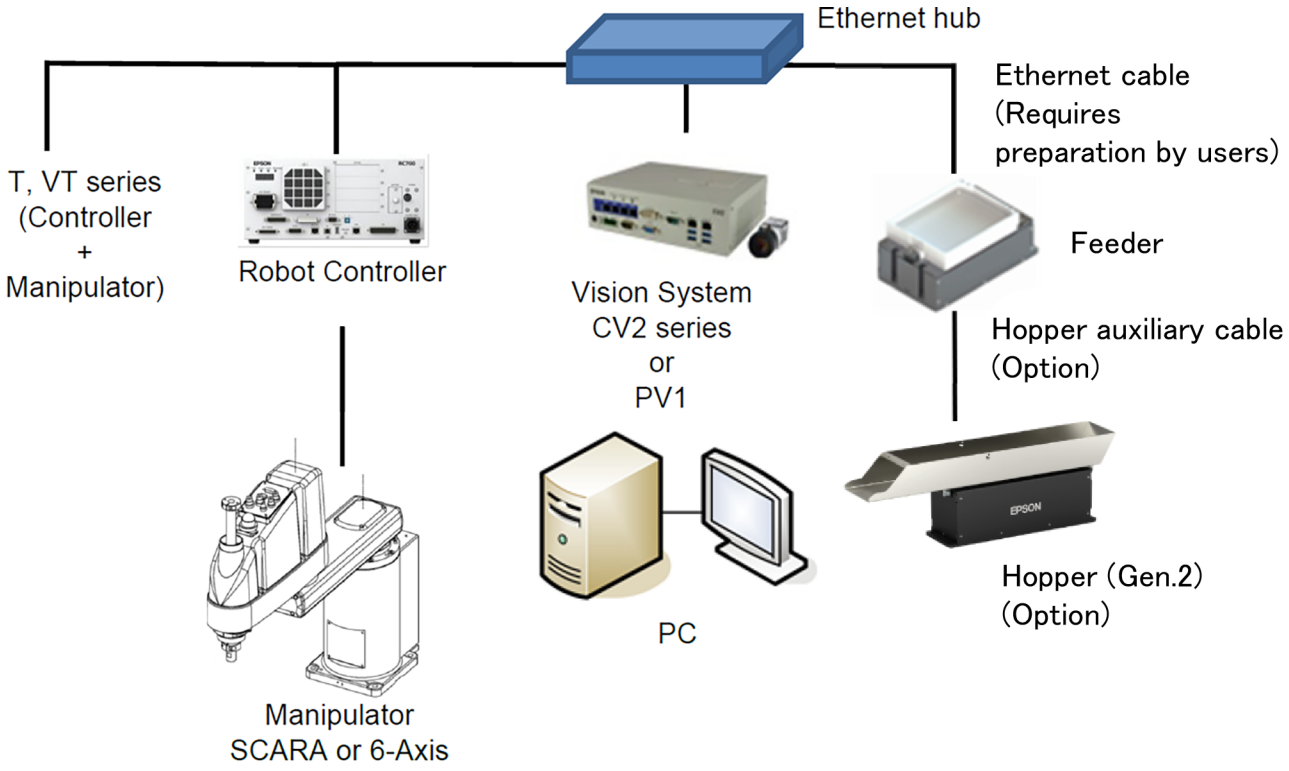
用語	說明
拾取	機器人抓住送料器上的零件之動作。
放置	機器人將抓住的零件放下或擺放到指定位置的動作。
Part Feeding程序	Part Feeding選配件內建的自動程序，可自動化視覺系統的運作和送料器的運作、並呼叫機器人運作。
回呼函數	Part Feeding程序會在特定條件下呼叫的SPEL+函數。函數內容請由您自行編寫。請編寫您的設備固有的處理內容（例如：使用機器人拾取或放置零件）。
零件座標佇列	用於取得送料器上的零件座標。座標以局部座標處理。
UPM	Unit per minute 機器人每1分鐘處理的零件數量。
主動零件	多種零件運作時作為主體的零件。屆時將會使用此零件的送料器運作參數。

2.4 系統概述

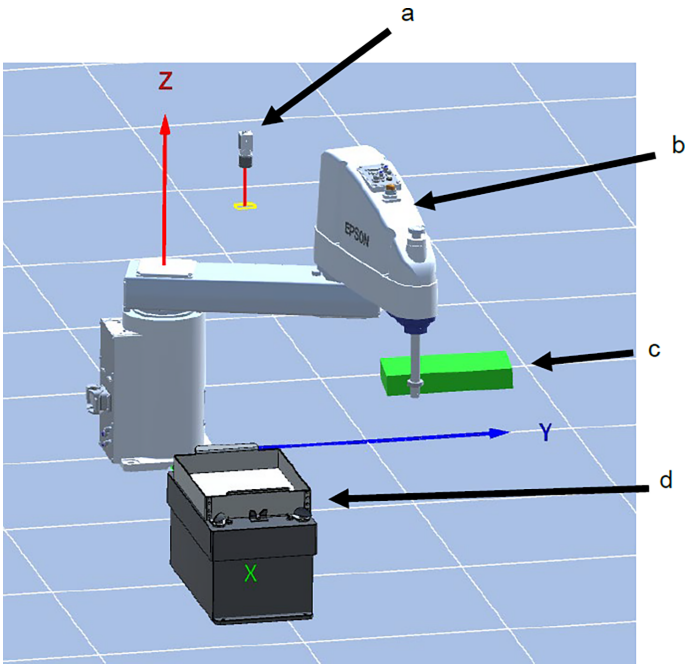
使用Part Feeding選配件，即可輕鬆實現拾取和放置零件的系統。以下為系統配置的說明。

2.4.1 整體配置

以下範例為使用Part Feeding選配件時的系統配置。
送料器IF-80附帶內建料斗。



以下為機械臂周邊的配置範例。



符號	項目
a	照相機
b	機械臂
c	零件的放置位置（棧板、托盤等）

符號	項目
d	送料器

2.4.2 送料器

送料器是本系統的必備項目。請務必準備。
可使用的送料器包括IF-80、IF-240、IF-380、IF-530、IF-A1520或IF-A2330。不支援其他的送料器。
1個控制器可控制多台送料器。
請使用從供應商購買的送料器。
若使用不是從供應商購買的送料器，則可能無法連接至控制器或無法充分發揮性能。

2.4.3 機器人

機器人是本系統的必備項目。請務必準備。

2.4.3.1 機械臂

可使用RC700系列、RC90系列、RC800-A控制器可連接的SCARA型及6軸型機械臂，以及T/VT系列。不支援X5系列和PG。
可在1台送料器上安裝最多2台連接至同一個控制器的機械臂。

2.4.3.2 夾具

末端夾具用於拾取零件。有使用真空、壓縮空氣吸附的類型，以及使用夾頭結構的類型。請您根據要使用的零件和需求選擇適合的類型。
本公司並未販售末端夾具。請您自行準備末端夾具。

2.4.4 視覺系統

視覺系統是本系統的必備項目。請務必準備。

2.4.4.1 視覺系統

可使用以下任一視覺系統：

- PC視覺系統PV1
  關於必要規格的資訊，請參閱以下內容。
  PC
- 緊湊型視覺CV2-SA/HA（韌體Ver.3.0.0.0或以上），或CV2-HB/SB/LB）韌體Ver.3.2.0.0或以上）
  不支援CV1及CV2-S/H/L。
  不支援其他廠牌的視覺系統。

2.4.4.2 攝影機

可使用任何可連接到視覺系統的攝影機。
需準備1台用於識別送料器上之零件的攝影機。此攝影機可安裝為朝下固定的攝影機、或安裝為機器人活動軸上的移動式攝影機。
如有必要，可再增設用於補償已抓住零件位置的朝上固定的攝影機，或用於對零件放置位置進行定位的攝影機。

2.4.5 照明

照明是用於正確識別平台上零件的必備項目。
可使用以下任一種或兩種照明：

- 送料器內建的背光
- 您自行準備的自訂照明（I/O控制、乙太網路控制等）

2.4.6 PC

在執行以下作業時需要使用電腦：

- 瀏覽或編輯Part Feeding的選配件設定時
- 執行送料器校準時
- 執行SPEL+程式設計時
- 擷取日誌時

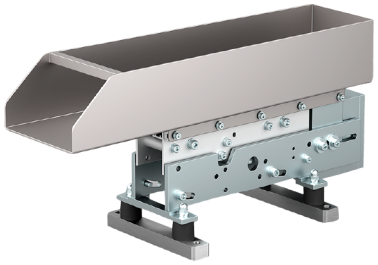
Part Feeding的程序運作無需連接至電腦即可進行。電腦的必要規格如下：

- 使用CV時：可安裝RC+
- 使用PV時：請參閱以下內容。
  「Vision Guide 8.0 Hardware & Setup - 系統需求」

2.4.7 料斗

料斗是向送料器供應零件的裝置。屬於選購品。
從供應商購買的料斗可連接到送料器。
您自行準備的送料器可透過IO或乙太網路等方式連接到控制器。
送料器IF-80為料斗一體型。

料斗有第1代（Gen.1）和第2代（Gen.2）這兩種系列。

料斗（Gen.2）	料斗（Gen.1）
	
■ 比料斗（Gen.1）更新的型號。 ■ 透過Epson RC+ 8.0調整振幅。	■ 透過隨附之料斗控制器上的電位計調整振幅。

2.5 硬體

2.5.1 確認隨附品的內容

Part Feeding選配件會依據訂購內容，於包裝內隨附以下物品。到貨後，請立即確認是否有缺少的物品。同時，請確認是否有損壞。

- Part Feeding授權（可能會單獨寄送）
- 送料器主體、背光（內建）
- 平台
- 送料器電源線
- 乙太網路線

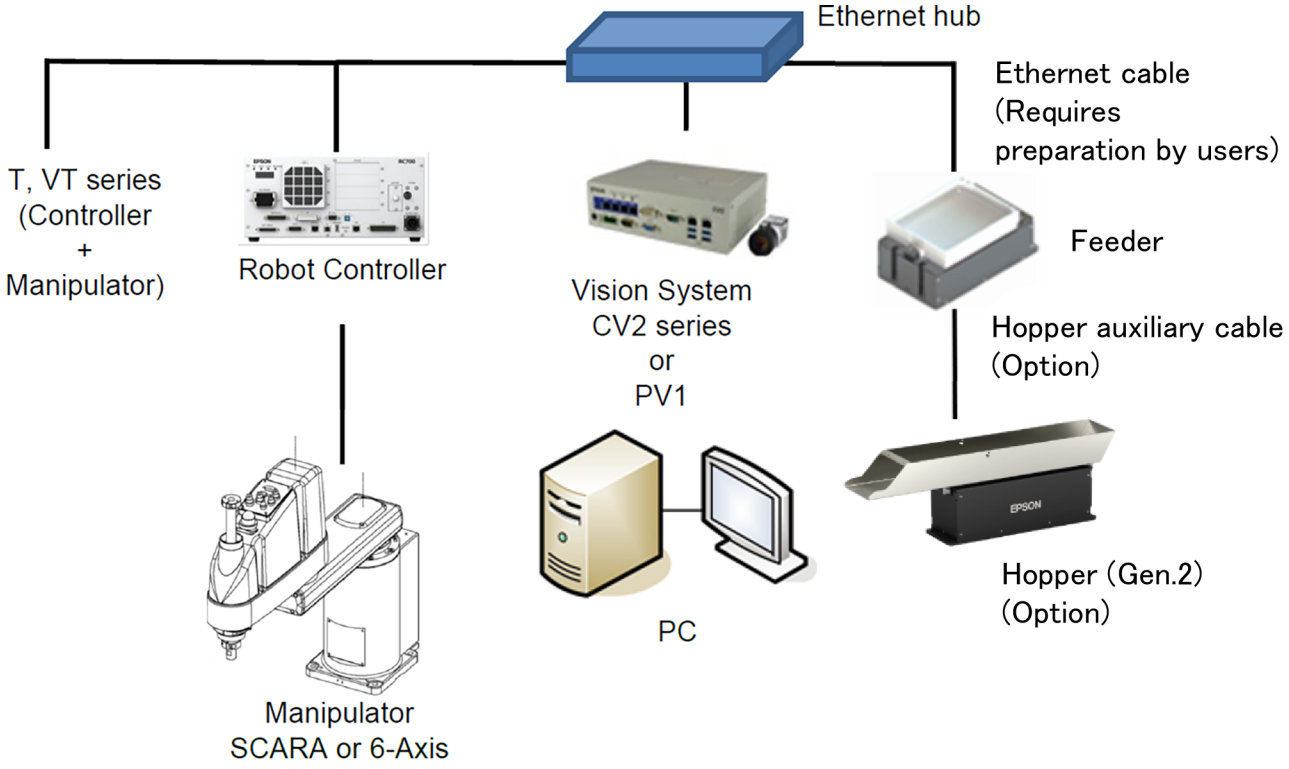
以下為選購品：

- 料斗
- 料斗控制器（料斗（Gen.1）用）
- 料斗隨附之電纜
- 其他選購品

2.5.2 系統配置

2.5.2.1 配置範例

以下為使用Part Feeding選配件的系統配置範例。
送料器IF-80附帶內建料斗。



每1台控制器最多可安裝4台送料器。T/VT系列最多可同時控制2台送料器。
在多種零件運作的情況下，每1台送料器最多可安裝2台機械臂。
連接乙太網路時，請使用乙太網路集線器。請使用以下視覺系統：

緊湊型視覺CV2系列	CV2-HA/SA（韌體Ver.3.0.0.0或以上）
	CV2-HB/SB/LB（韌體Ver.3.2.0.0或以上）
PC視覺系統PV1	

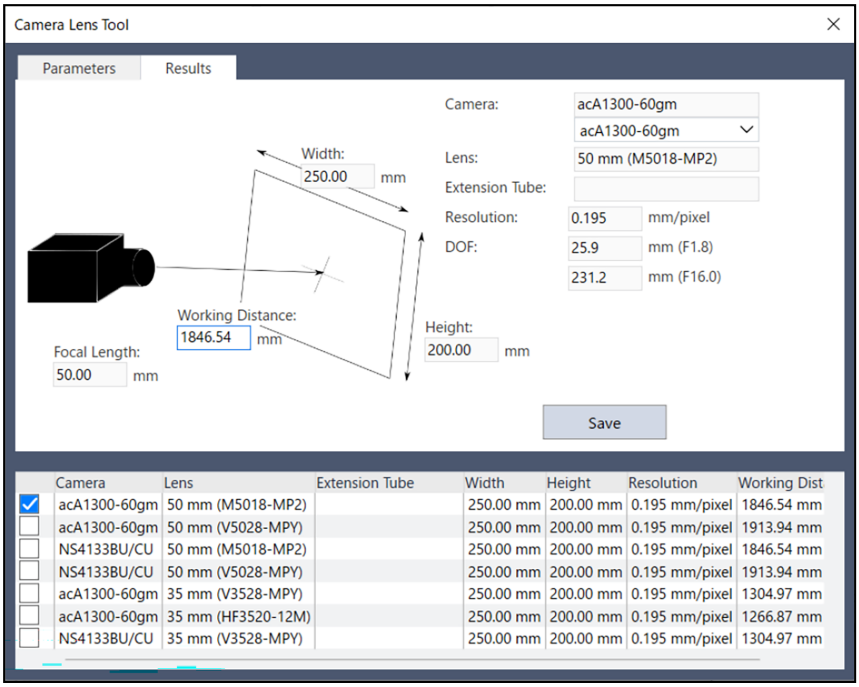
不支援CV1及CV2-S/H/L。
不支援其他廠牌的視覺系統。

2.5.2.2 配置選擇的考量事項

- Part Feeding選配件可使用的機械臂有SCARA型和6軸型機器人。不支援X5系列和PG。
- 使用PV時，請勿在同一個網路埠執行送料器通訊和攝影機通訊。若送料器通訊和攝影機通訊同時發生，則可能會影響成像或送料器的運作。請在電腦上增設網路卡或使用多埠網路卡，並直接連接攝影機和網路埠。
- 依據送料器的規格，即使從多台機器人控制器連接到1台送料器也不會產生錯誤。進行網路設定時，請注意IP位址的配置，避免將多台機器人控制器連接到1台送料器。
- 每1台送料器最多可安裝2台料斗。
- 可在模擬器上顯示送料器和料斗的CAD資料。CAD資料檔案位於以下資料夾：
  `C:\EpsonRC80\Simulator\CAD\PartFeeder`

2.5.2.3 選擇攝影機鏡頭

使用光學選擇工具，根據視野、工作距離選擇鏡頭和延長管的厚度。
可從Epson RC+ 8.0功能表 - [設置] - [系統配置] - [視覺系統] - [攝影機] - [光學選擇工具]啟動光學選擇工具。
如需詳細資訊，請參閱以下內容。
「Vision Guide 8.0 Hardware & Setup - 設置篇 - 光學選擇功能」



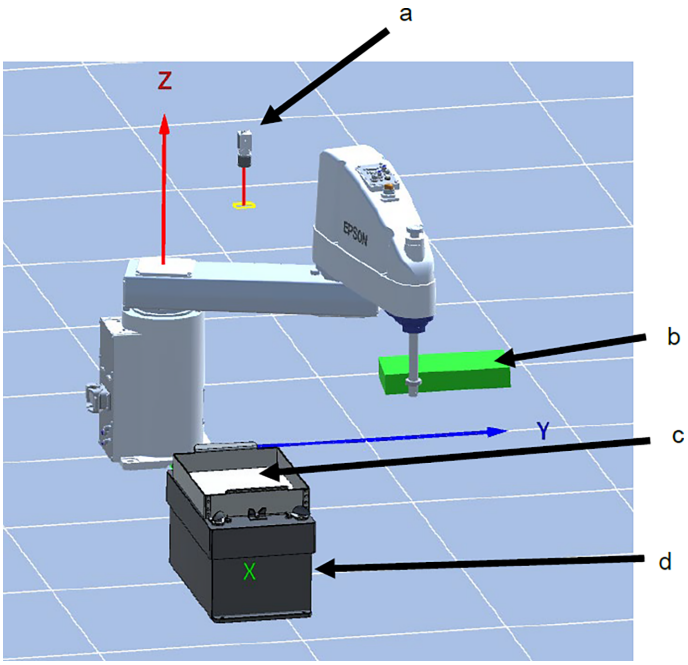
2.5.3 安裝、調整

2.5.3.1 機械臂和控制器

關於機械臂的安裝，請參閱各機械臂的手冊。請尤其要按照機器人系統安全手冊的指示，於安裝時注意安全。
末端夾具請由您自行製作或購買。關於控制器的安裝，請參閱控制器的手冊。

2.5.3.2 攝影機、鏡頭

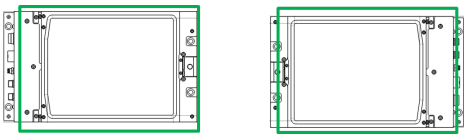
將攝影機朝下安裝，使其能夠俯視整個送料器平台。
建議將攝影機固定朝下。也可使用移動式攝影機。



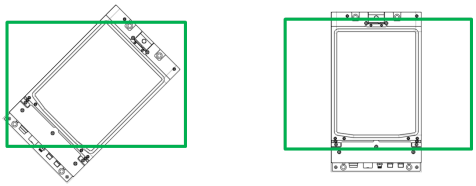
符號	項目
a	攝影機（朝下固定）
b	放置位置
c	平台
d	送料器

需要將送料器的長度方向與攝影機視野（下圖綠色框）的水平方向對齊。若未對齊，Part Feeding系統將無法正常運作。送料器的方向可以是左右任一方向。

正常範例



不良範例



使用移動式攝影機時，也請如上圖所示，針對平台的長度方向與攝影機視野水平方向對齊的位置進行教導（建立點資料）、以其作為成像位置。

當使用攝影機對零件進行成像時，請依下圖所示，調整鏡頭的焦距及光圈，以確保零件能清晰識別且平台表面的亮度均勻。

2.5.3.3 送料器、料斗

安裝送料器和料斗時，請注意以下幾點：

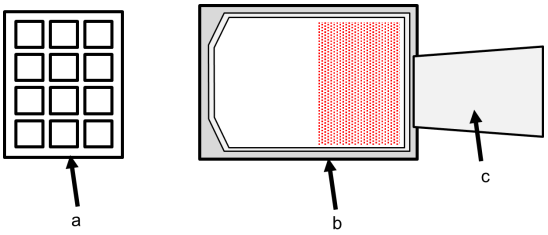
- 安裝在水平面上。
- 安裝在高剛性的底座上。
- 用螺栓確實地固定。

若安裝在不平坦的位置或低剛性的底座上，可能會導致零件分散不均或零件無法充分分散，使得可拾取的零件數量減少、週期時間效率變差。安裝詳情請參閱以下各送料器的手冊。
「Epson RC+ 8.0選配件 Part Feeding 8.0　IF-***篇 環境與設置」
***：送料器機型名稱（IF-80、IF-240、IF-380或IF-530）
「Epson RC+ 8.0選配件 Part Feeding 8.0 IF-A1520 & IF-A2330篇 設置」

將料斗安裝在水平面上。用螺栓牢牢地固定在高剛性的底座上。料斗突出平台的部分應盡量減少。盡量讓平台面積保持寬廣，藉此增加可拾取的零件數量並提升週期時間的效率。安裝詳情請參閱以下手冊。
「Epson RC+ 8.0選配件 Part Feeding 8.0 Hopper篇」

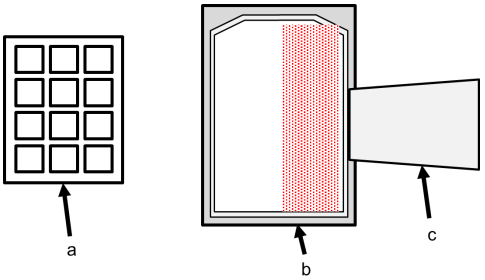
關於送料器和料斗的位置關係，在配置料斗時，應確保送料器上的零件放入位置與放置位置呈對向側。請將零件放入到下圖紅色網格標示的區域。此配置可利用Part Feeding選配件的並行供料功能，提升系統的週期時間效率。

配置範例1



a	放置位置
b	送料器
c	料斗

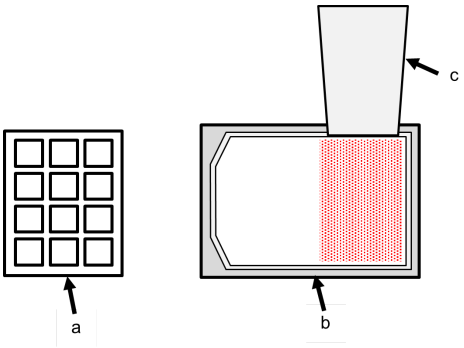
配置範例2



a	放置位置
b	送料器
c	料斗

料斗的供料過程中，若有零件滾入機器人正在拾取的區域，將對零件的拾取產生不良影響。
採取下列料斗配置的話，則可有效防止這類情況：

配置範例3



a	放置位置
b	送料器
c	料斗

要從料斗適當放入零件時，將適量的零件放入到適當位置非常重要。因此，請考量針對料斗進行以下額外加工：

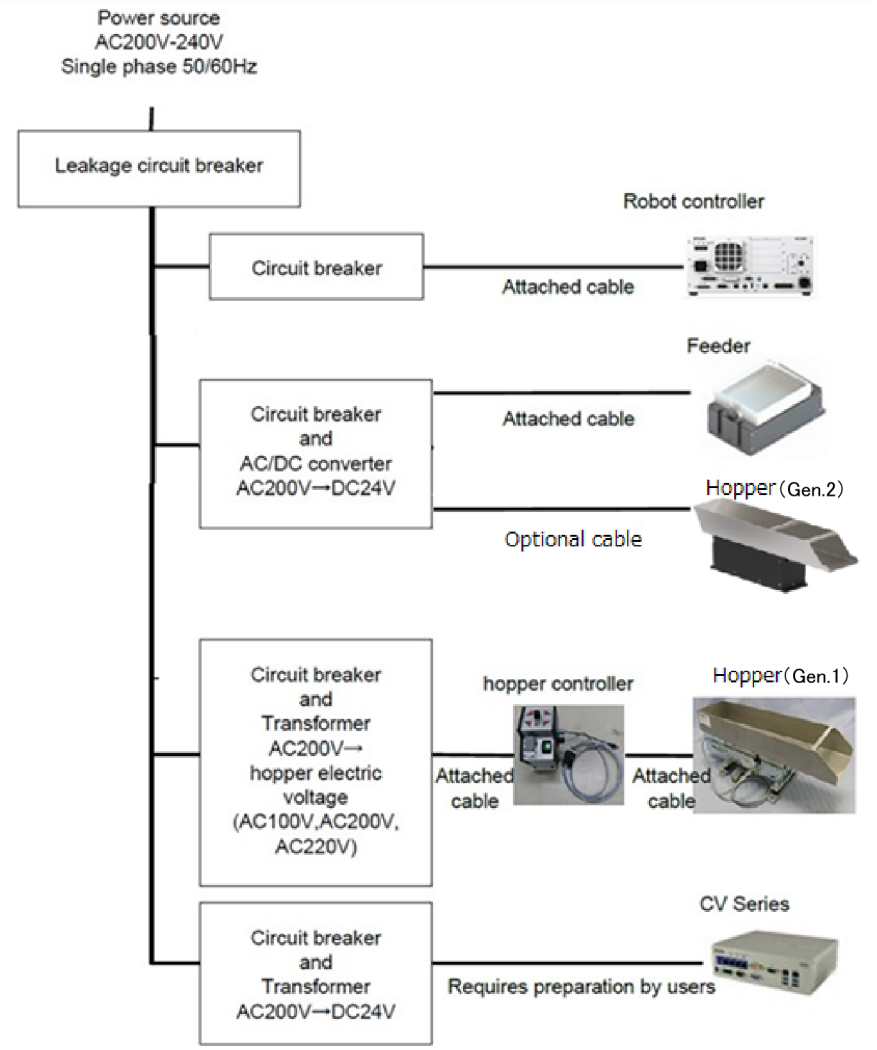
- 在料斗中途設置用於調整零件流量的擋板結構
- 在料斗出口設置用於限制零件排出位置的導引裝置

2.5.4 電氣佈線

2.5.4.1 電源供應的注意事項

下面說明機器人控制器、CV2、送料器、料斗的電源供應注意事項。
請根據「JIS B 9960-1（IEC 60204-1）機械類的安全性-機械的電氣裝置- 5.1輸入電源導體的連接」，將您設計的機械電氣設備連接到單一電源。若設備的特定部分（例如送料器和料斗）要使用與輸入電源不同的電源，請透過機械電氣設備內的變壓器或轉換器等設備進行供應。

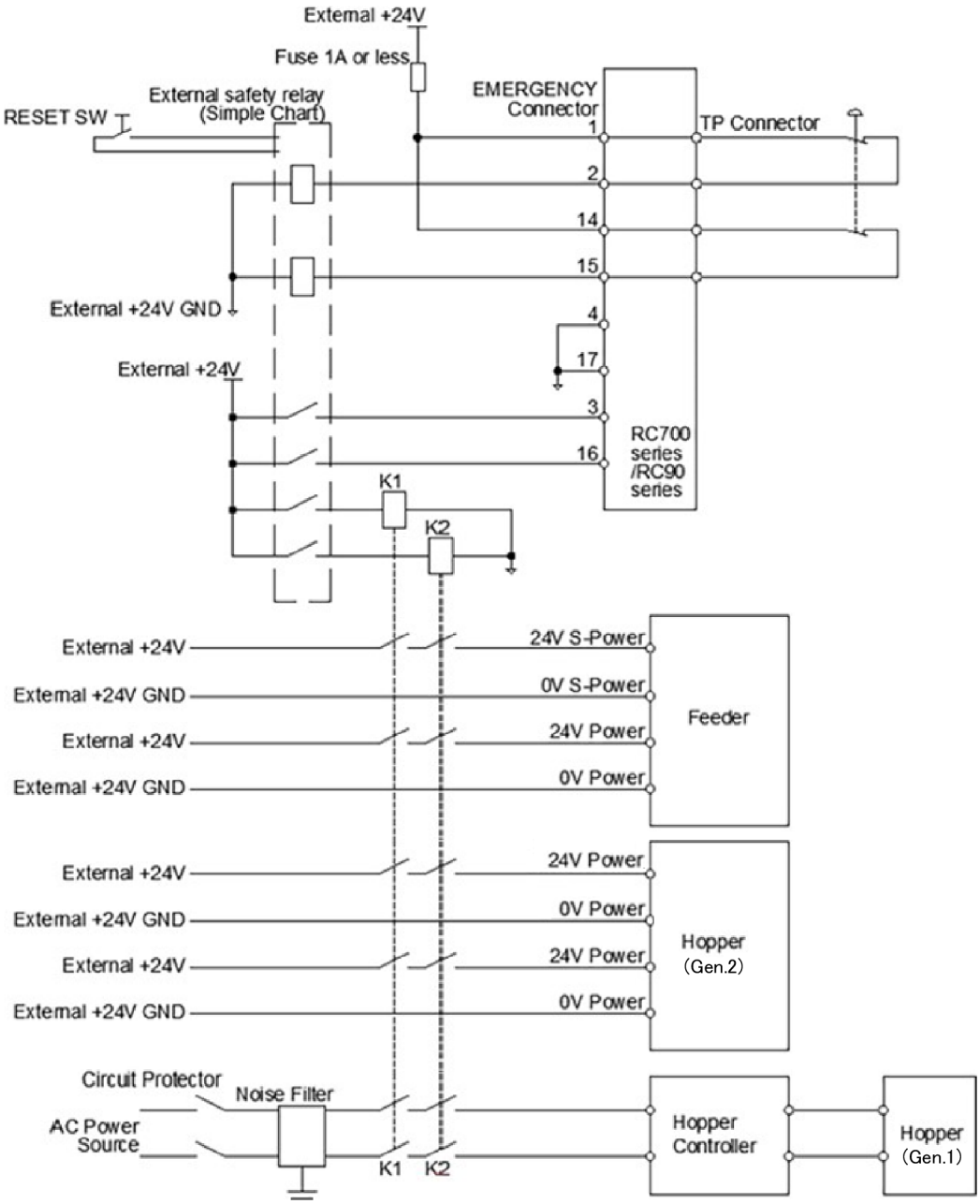
以下內容為連接到單一的AC200V電源時，您需設計的電氣設備之概略範例。詳情請遵循「JIS B 9960-1（IEC 60204-1）機械類的安全性-機械的電氣裝置-」。



2.5.4.2 向送料器供電

如需佈線的詳細資訊，請參閱以下手冊。
「Epson RC+ 8.0選配件 Part Feeding 8.0 IF-***篇 - 電纜連接」
***：送料器機型名稱（IF-80、IF-240、IF-380或IF-530）
「Epson RC+ 8.0選配件 Part Feeding 8.0 IF-A1520 & IF-A2330篇 - 電氣規格」

請參考控制器手冊的「EMERGENCY」-「電路圖和佈線範例」，設計一個在按下緊急停止開關時，會透過外部安全繼電器來關閉送料器及料斗電源的電路。概略參考電路圖如下所示。



2.5.4.3 向料斗供電

如需佈線的詳細資訊，請參閱以下手冊。
「Epson RC+ 8.0選配件 Part Feeding 8.0 Hopper篇」

2.5.4.4 機器人的佈線

根據各機器人、控制器的手冊進行佈線。

2.5.4.5 攝影機的佈線

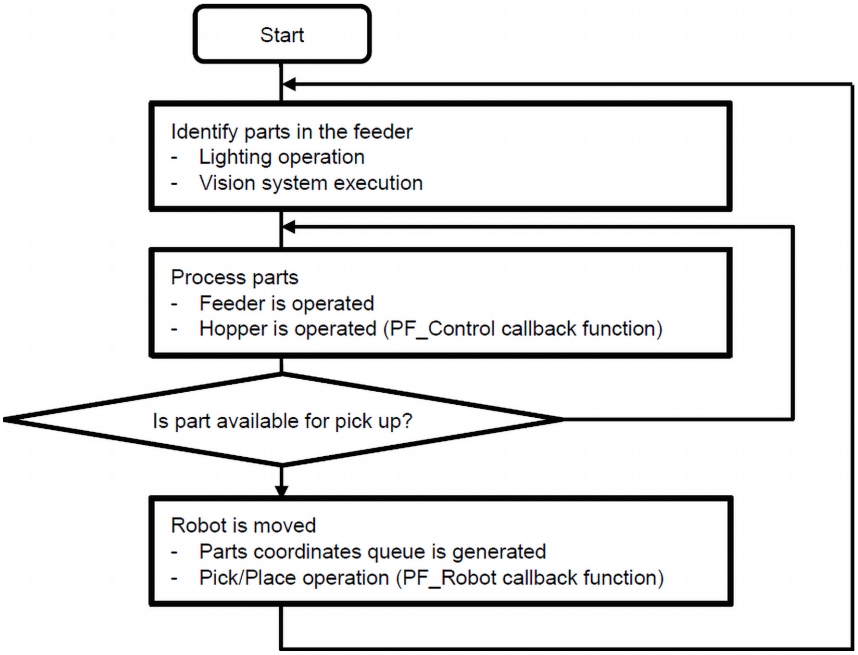
關於佈線的詳細資訊，請參閱以下手冊。
「Epson RC+ 8.0選配件Vision Guide 硬體篇」

2.6 運作概要

下面說明Part Feeding選配件的運作概要。

2.6.1 Part Feeding程序

Part Feeding程序是Part Feeding系統內建的運作程序，可自動執行視覺系統和送料器的控制。
若要啟動Part Feeding程序，請透過您的程式執行PF_Start命令。
Part Feeding程序的內容如下：



1. 識別送料器上的零件
   使用視覺系統來識別平台上的零件數量和分布。
2. 處理零件
   透過控制送料器並移動零件，讓機器人能夠更容易地抓住零件。當零件數量少或沒有零件時，則呼叫PF_Control回呼函數，透過料斗供應零件。
3. 移動機器人
   生成零件座標佇列（送料器上零件座標的列表）。呼叫PF_Robot回呼函數，執行零件的取放動作。

透過您的程式呼叫PF_Stop命令，即可停止Part Feeding程序。

2.6.2 向送料器供應零件

向送料器供應零件的方法如下：

- 使用料斗
- 人工供料

2.6.2.1 零件的供應數量

向平台供應零件的數量是決定運作週期時間的重要因素。

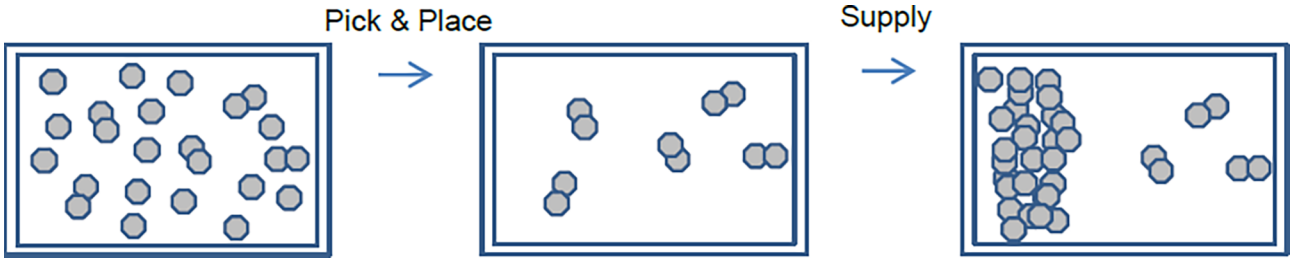
- 零件數量過多時：
  會導致零件重疊、使得送料器必須運作多次，導致週期時間效率變差。
- 零件數量過少時：
  需要多次向平台供應零件。週期時間效率變差。

有適當的供應數量（送料器運作後平台上的零件數量）。此數量可透過送料器校準來進行計算。

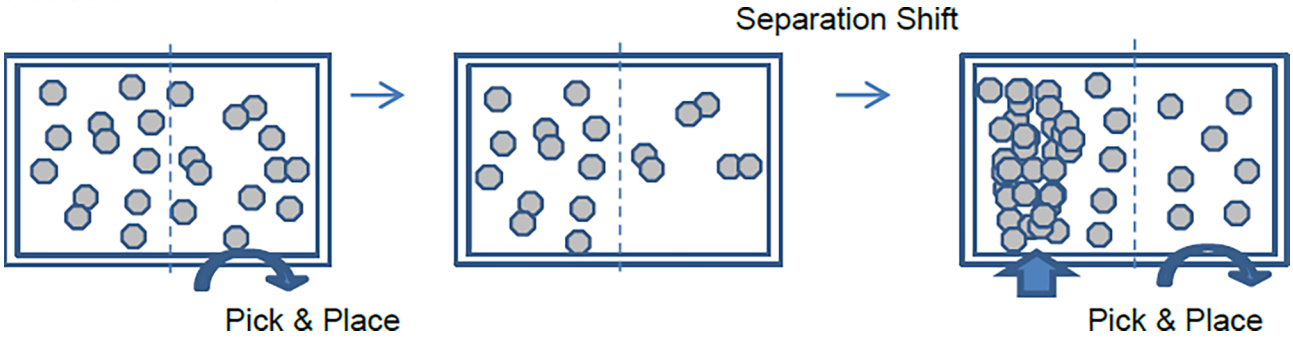
向送料器供應零件的方式（時機）有以下3種：

1. 拾取完後供料
   將送料器上的零件全部拾取完後再供應零件。
   由於送料器上零件的滯留時間幾乎保持一定，建議對振動敏感性（不耐振動）零件採用此方式。然而，每1次送料器運作時，機器人能取得的平均零件數量較少，因此會拉長週期時間。

2. 追加供料
   在送料器上可取得的零件用盡後，再追加零件。
   每1次送料器運作時，機器人能取得的平均零件數量較多，因此週期時間會縮短，生產效率提高。但是，零件的滯留時間會有差異，因此振動敏感性零件不適合使用此方式。



3. 並行供料
   此方式需要搭配指定零件拾取位置的功能使用。
   於機器人拾取的同時，在零件拾取位置的對面區域追加零件。每1次送料器運作時，機器人能取得的平均零件數量較多，因此週期時間會縮短。此外，由於料斗和機器人可同時運作，因此會更進一步縮短週期時間。然而，會產生在送料器上停留較長時間的零件，因此振動敏感性零件不適合使用此方式。而且也需要由您自行編寫程式來確保料斗和機器人同時運作。

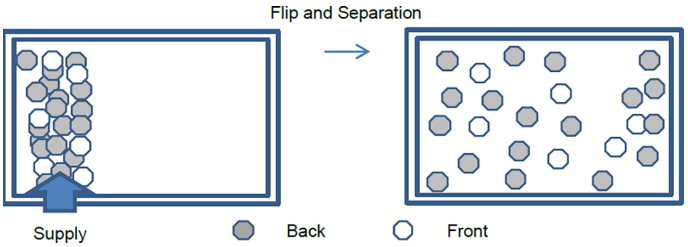


2.6.3 送料器的運作

Part Feeding選配件會根據送料器上的零件狀態，自動選擇並執行送料器的運作。這讓機器人更容易抓住零件。
送料器的運作如下所示：說明圖僅屬於大致參考。有可能會不同於實際運作狀況。

2.6.3.1 翻轉和分離

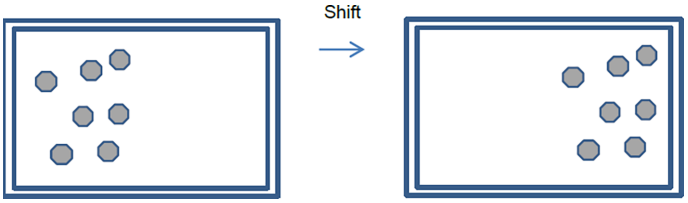
讓零件均勻分散在平台上。在零件之間保持適當的間隔，可讓機器人更容易抓住零件。



在分離之前，可能會先執行將零件移動到中心的動作。此動作稱為「集中」。

2.6.3.2 位移

保持零件的間隔（分布）不變，將全體零件向一個方向移動。
讓零件靠近放置位置，藉此可縮短機器人的移動距離，提升週期時間的效率。



位移方向包括前向（接近拾取位置的方向）和後向（遠離拾取位置的方向）。

2.6.4 平台上的零件拾取位置

機器人拾取零件的位置有「全面拾取」和「部分拾取」2種。
部分拾取和全面拾取哪種效率更好，取決於您的零件、末端夾具、料斗等設備的配置。請實際操作設備，並擷取日誌以進行確認。

2.6.4.1 全面拾取

針對整個平台進行拾取。
若零件相對於平台尺寸較大（以IF-240而言，大約2平方公分以上），則選擇全面拾取。

2.6.4.2 部分拾取

針對靠近放置位置的區域進行拾取。
採用此方式時，會根據零件分布，自動執行將零件位移到拾取目標區域的操作。此外，可在非拾取目標區域同時執行機器人運作和零件的供料。（需要由您自行編寫程式。）一般而言，使用這些功能可比全面拾取縮短更多機器人的週期時間。

2.6.5 迴避末端夾具與平台的干擾

為了防止末端夾具與平台發生物理性干擾，需要將送料器上的零件可拾取範圍設定在平台外周的內側。而Part Feeding選配件可讓使用者輕鬆指定這段距離。

2.7 零件

下面說明可在Part Feeding選配件中使用的零件。
每1個專案最多可註冊32種零件。

提示

供應商已建立評估您的零件是否適合Part Feeding選配件的體制。欲了解詳情，請聯繫您的供應商。

2.7.1 可處理零件的條件

送料器可處理的零件有以下條件：

2.7.1.1 視覺系統的適合性

需要能透過視覺系統正確識別零件。

- 透明樹脂成形的零件會使光線穿透，因此可能無法被正確地識別形狀。於此情況下，則可能透過將照明變更為非可見光或使用反射照明來解決。
- 辨別零件正反面時，有時會因為零件形狀的緣故而無法區分正反面。於此情況下，可能透過追加反射照明來解決。

2.7.1.2 大小、重量

零件越大，平台可容納的零件數量（不重疊且可鋪滿的數量）就越少。若此數量較小，送料器的運作次數將會增加，機器人運作時間相對減少，因而導致週期時間效率變差。針對零件數量的參考標準，則建議送料器能容納50個以上的零件。
零件的總重量（1個零件的重量×平台上不重疊情況下可放入的零件數量）需要小於送料器的可承載重量。若超過此重量，送料器將會過載、導致零件分離能力下降、週期時間效率變差，甚至縮短送料器的壽命。
關於送料器的可承載重量，請參閱各送料器的手冊。

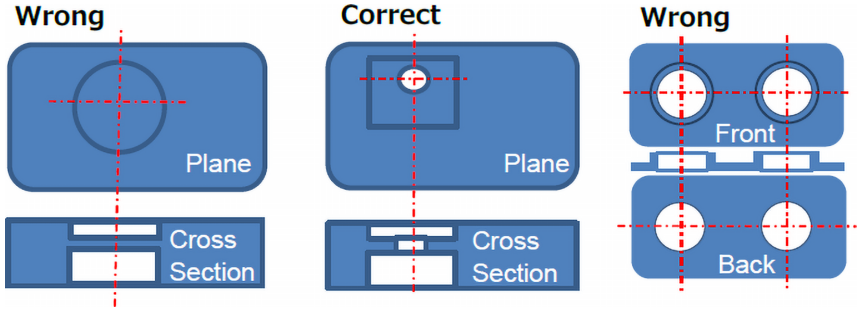
2.7.1.3 零件的材質、狀態

以下零件不適合用於Part Feeding選配件：

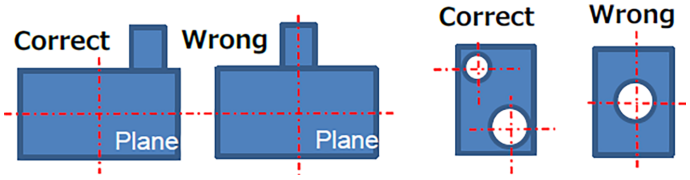
- 材質柔軟或輕盈的零件
  例如：紙、纖維材質物
- 受到振動會破損或變形、摩擦後會產生粉塵的零件
  例如：粉末壓製物、塗裝物
- 具黏著性或會滲出液體的零件
  例如：食品

2.7.1.4 零件的形狀、其他

- 球形的零件無法在送料器上停止移動，因此難以進行拾取。請使用防滾動平台的選配件。
  例如：軸承鋼球
- 容易纏繞起來的零件難以進行分離。
  例如：螺旋彈簧
- 剖面形狀不同且材質不透光的零件，可能會在穿透光的影響之下，變得無法辨別正反面。以下為此類零件的例子：

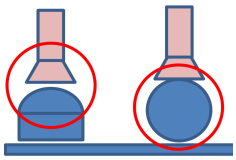


每個剖面方向的形狀不同的零件無法進行分類

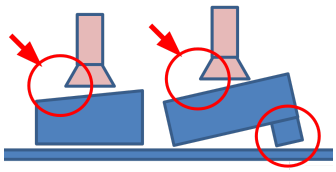


每個平面的外形形狀不同時，則可識別正反面

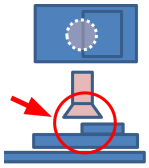
- 要採取吸附式拾取時，則建議選擇如下零件：吸附面與送料器底面呈平行，能讓吸盤垂直降下且能確保足夠吸盤面積的零件。以下為此類零件的例子：



球體（Ball）

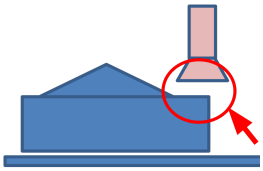


吸附面與送料器底面無法呈平行的零件

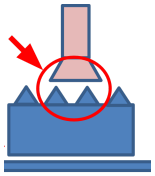


吸附位置有高低差時

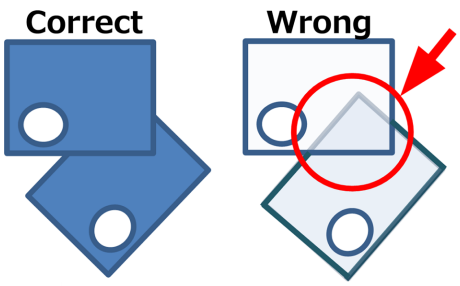
- 要使用吸附方式供料時，請確保重心處無高低差。以下為此類零件的例子：



無法確保足夠吸附面積的零件



無法確保表面平滑的零件（使用吸附方式供料時）



能從底部透光的零件

- 要在組裝工序使用時，請採取能在拾取動作後防止零件位置偏移的對策。例如在零件上設置導孔、在末端夾具側設置定位銷等用於抑制位置偏移的對策。另一種對策則是設置朝上固定的攝影機，用於對拾取後的零件進行定位。

2.7.2 零件範例

以下為Part Feeding選配件可使用的零件範例。
IF-240及IF-A1520適合No.1～3的零件。No.4、5的零件對於IF-240及IF-A1520而言過大或過重，因此不適合。
No.6、7的零件對於IF-240、IF-380、IF-530、IF-A1520、IF-A2330而言過小或過輕，因此不適合。

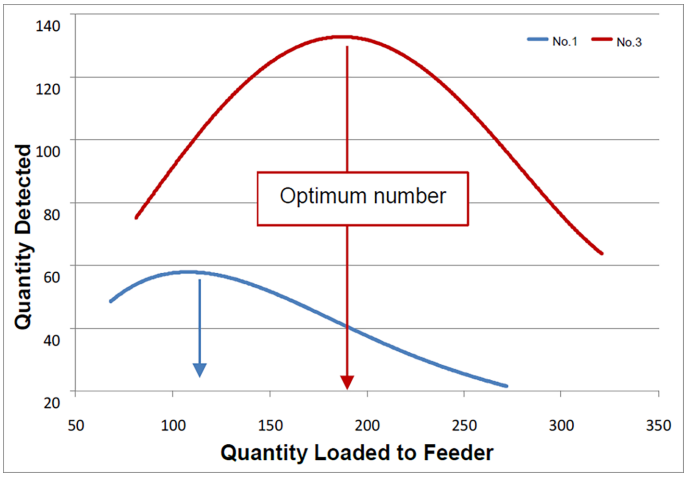
No.	圖	特質	尺寸 [mm]	重量 [g]	註解
1		金屬沖壓零件	10 × 10 × 0.2	0.088	適合IF-240、IF-A1520
2		金屬沖壓零件	11 × 5.5 × 0.2	0.029	適合IF-240、IF-A1520
3		樹脂零件	10 × 9 × 2.1	0.127	適合IF-240、IF-A1520
4		尼龍連接器	21 × 29.9 × 21	7.1	適合IF-380、IF-A2330
5		加高型螺帽	36 × 11 × 9.5	14	適合IF-380、IF-530、IF-A2330
6		IC	5 × 4.4 × 1.5	0.082	適合IF-80
7		金屬襯套	ø4 × 1	0.102	適合IF-80

No.1、2無法僅使用背光來判別正反面。
No.3可以僅使用背光來判別正反面。

2.7.2.1 放入送料器的數量與影像處理檢測數量的關係

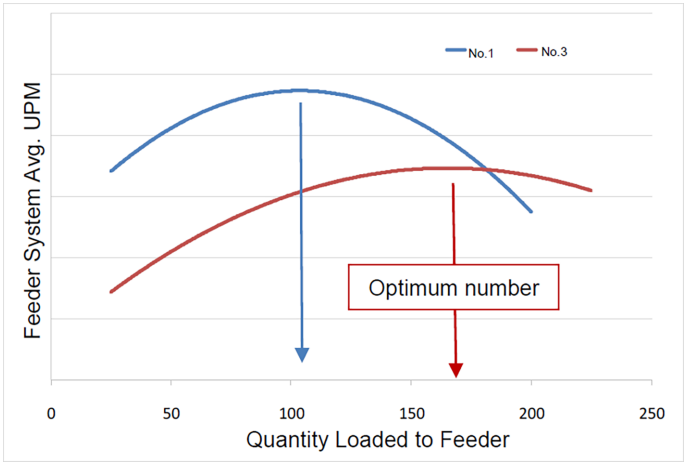
放入送料器的零件數量與影像處理檢測數量之間，在關係圖上呈現上凸的曲線。
送料器上的零件會與相鄰的零件接觸或重疊，因此無法檢測到所有放入的零件。與相鄰零件的接觸程度和重疊傾向會隨著零件的不同而變化，在圖表上呈現的形狀也會不同。使用Part Feeding選配件，即可參考供應商的實驗資料，透過校準來尋求最佳的零件放入數量。

以下呈現的是放入送料器的零件No.1和3數量與影像處理檢測數量的關係圖。
請注意，並無法檢測到所有放入送料器的零件，且檢測數量會隨放入數量而變化。



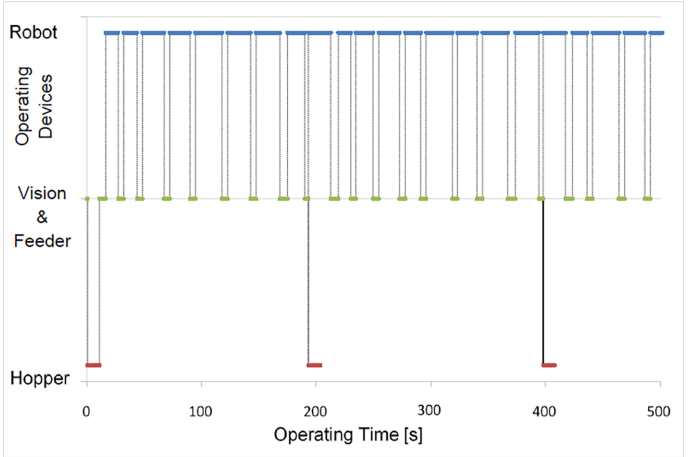
2.7.2.2 放入送料器的數量與平均UPM（Unit Per Minute）之關係

平均UPM是指在特定時間內執行拾取動作時，每單位時間的平均拾取數量。
以下呈現的是放入送料器的零件No.1和3數量與平均UPM的關係圖。零件No.1和3的圖表都呈現上凸的曲線。由於平均UPM會隨機器人的速度、加速度、移動量而變化，故於此處特意不呈現縱軸的數值。請注意，除了機器人的運作條件外，UPM還會隨著放入送料器的零件數量而變化，且存在能增大UPM的最佳放入數量。



2.7.2.3 送料器的運作與UPM（Unit Per Minute）的關係

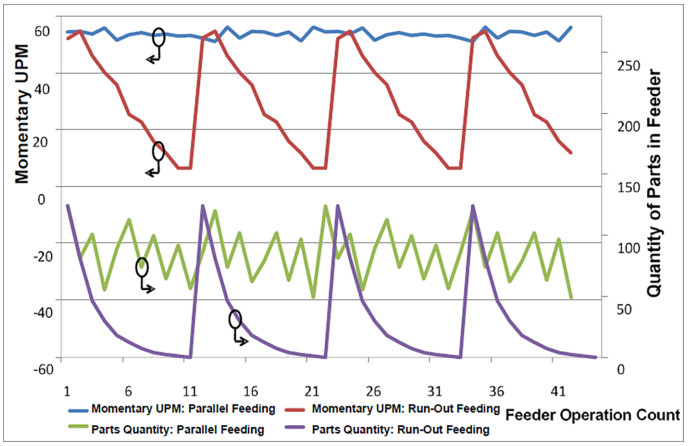
以時間為橫軸，以機器人、視覺系統與送料器、料斗為運作設備。繪製各設備運作的時機之後，可呈現如下圖所示的結果。開始運作時，視覺系統和送料器會進行運作，並且在檢測到送料器上沒有零件後驅動料斗、將零件放入送料器。



接著，送料器繼續運作，分散零件，透過視覺系統檢測零件，讓機器人執行拾取動作。當可拾取的零件用盡時，會再透過視覺系統和送料器的運作，分散、檢測零件。接著，再次執行機器人的運作。
重複視覺系統、送料器和機器人的運作期間，送料器上的零件數量將會逐漸減少。依照設定的閾值，在料斗運作的時機驅動料斗以放入零件。此處的圖表內容是以並行供料作為前提。
由於需重複視覺系統、送料器和機器人的運作，在視覺系統、送料器運作的瞬間，UPM會變為「0」。機器人運作瞬間的UPM會比「放入送料器的數量與平均UPM（Unit Per Minute）的關係」中所顯示的平均UPM值更大。請注意，平均UPM是視覺系統和送料器運作瞬間的UPM=0與機器人運作瞬間的UPM之時間的平均值。
此外，圖中機器人運作的藍線長度並不固定。這是因為可拾取的零件數量會因送料器上的零件分散狀況而異，且送料器上的零件數量以及可拾取零件也會隨著重複拾取動作而減少的緣故。
為了穩定地供應零件，需要盡可能讓送料器上的零件數量保持穩定。

2.7.2.4 送料器上的零件數量與料斗運作的關係

在為了穩定地供應零件，透過料斗進行拾取完後供料的情況下，和以最佳放入數180、料斗放入數90進行並行供料的情況下，每次送料器運作的瞬時UPM（Unit Per Minute）和送料器上的零件數量之關係如下列圖表所示。



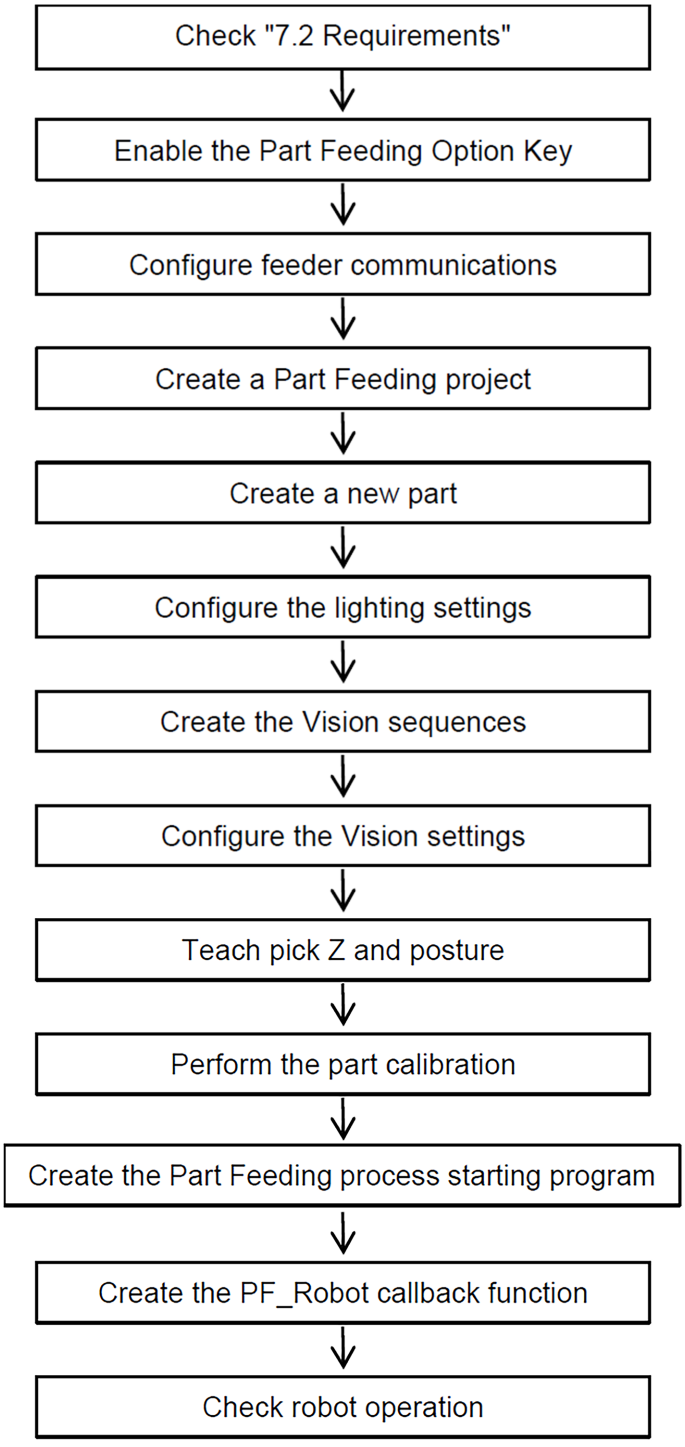
在拾取完後供料中，由於必須拾取完零件才會再透過料斗供料，所以瞬時UPM會不斷下降。當零件用盡後從料斗供應零件時，瞬時UPM會恢復原狀。
在並行供料中，料斗每運作2～4次，料斗就會運作1次，因此送料器上的零件數量不會低於下限值，瞬時UPM的變動也較小。

2.8 開始使用

請試著使用Part Feeding選配件來建立一個進行零件取放的系統。

2.8.1 作業流程

作業程序如下所示：

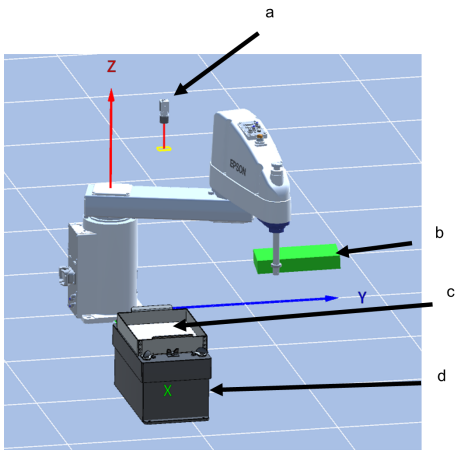


2.8.2 前提條件

2.8.2.1 設備配置

- 機械臂使用SCARA型機器人。
  使用6軸型機器人時的作業內容也相同。
  假設已連接了適合零件的末端夾具。
- 攝影機為朝下固定的攝影機。
- 使用送料器的背光。

- 不使用料斗。



符號	項目
a	攝影機（朝下固定）
b	機械臂
c	零件的放置位置（棧板、托盤等）
d	送料器

2.8.2.2 連接、調整

參閱以下要點。

- Epson RC+已和控制器連接
- 機械臂已和控制器連接
- 送料器已和控制器連接
- Vision Guide（PV或CV）已和控制器連接
- 機械臂、攝影機、送料器已正確安裝
- 攝影機位置、焦距、亮度的調整已完成

2.8.2.3 零件

- 零件ID為「1」。
- 形狀簡單且無正反面之分的零件。使用視覺系統的Blob物件（透過面積值檢測）進行檢測。

2.8.2.4 設定

參閱以下要點。

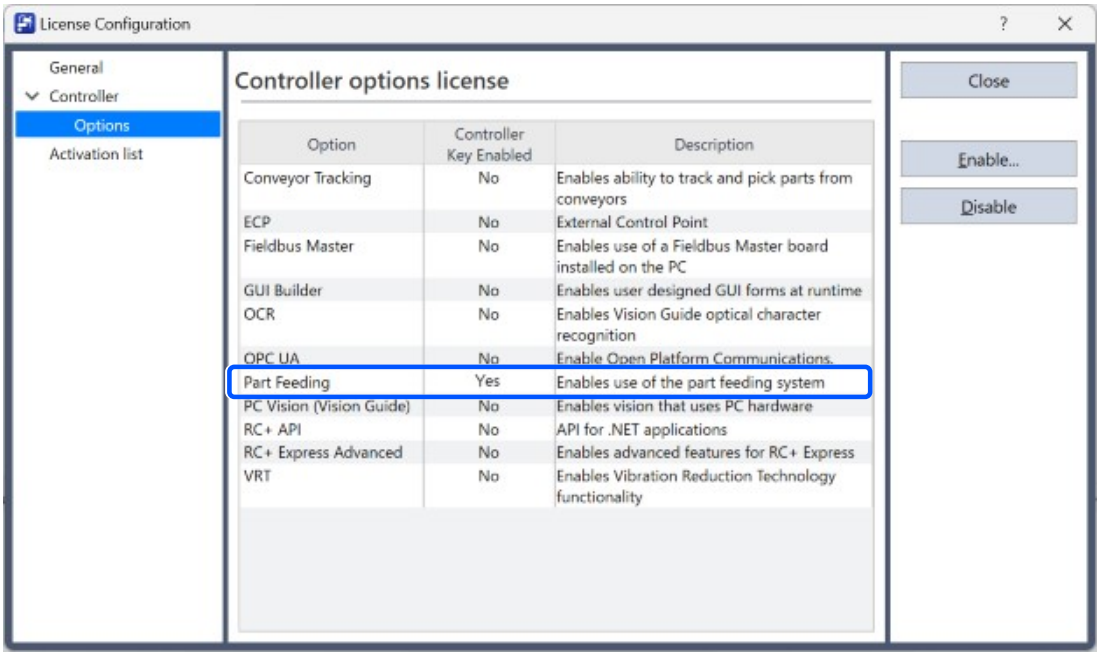
- 機械臂已註冊在系統配置中
- 視覺系統校準已完成
- 工具座標系統的設定已完成

2.8.2.5 其它

錯誤處理將執行回呼函數的範本程式碼中所編寫的內容。

2.8.3 啟用Part Feeding選配件

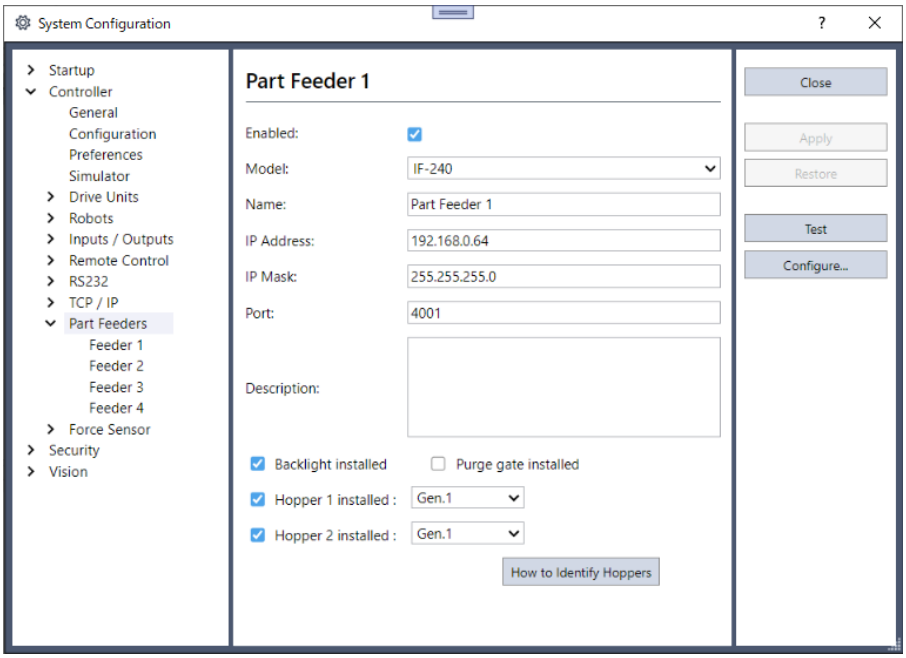
要使用Part Feeding功能，則需要啟用選配件。
連接控制器，從Epson RC+ 8.0功能表 - [設置] - [授權設定]畫面的樹狀檢視中選擇[控制器] - [選配件]後，將會顯示以下畫面。



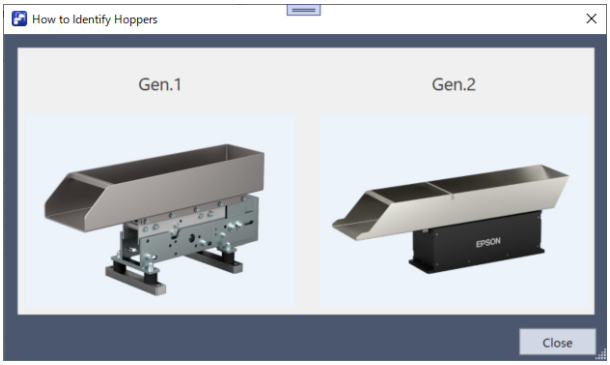
啟用選配件的步驟因控制器型號而異。詳細資訊請參閱以下手冊。
「Epson RC+ 8.0 使用指南 - 安裝控制器授權」

2.8.4 送料器的初始設定

1. 選擇Epson RC+ 8.0功能表 - [設置] - [系統配置]。
2. 選擇樹狀清單[控制器] - [零件送料器] - [送料器 1]。



3. 設定以下項目。

項目	描述
啟用	若要啟用送料器，則勾選核取方塊。
型號	選擇送料器的型號。
送料器名稱	設定任意名稱。（半形英數字和底線。最多32個字元）
IP位址	輸入目前設定在送料器上的IP位址。 預設IP位址為192.168.0.64。
子網路遮罩	輸入目前設定在送料器上的子網路遮罩。 預設的子網路遮罩為255.255.255.0。
連接埠	輸入目前設定在送料器上的連接埠編號。 預設的連接埠編號為4001。
註解	填寫送料器的說明（註解）。屬於選項。（半形英數字和底線。最多256個字元）
安裝背光	若有安裝送料器內建的背光，則勾選核取方塊。
安裝清除閘門	若要使用送料器選配件的清除閘門，則勾選核取方塊。（僅適用於IF-240、IF-380、IF-530、IF-A1520、IF-A2330）
安裝料斗1 安裝料斗2	有連接料斗時，則勾選核取方塊。從Gen.1/Gen.2中選擇料斗類型。關於料斗類型的詳細資訊，請參閱以下內容。 料斗
料斗的區分方法	顯示料斗的類型。 

4. 設定完成後，點擊[套用]按鈕。

提示

- 若要變更送料器的IP位址，請參閱以下內容。

  零件送料器畫面
- T/VT系列控制器雖然可設定4台送料器，但最多只能同時控制2台。
- 部分項目會根據使用的送料器、料斗自動進行設定。如需詳細資訊，請參閱以下內容。

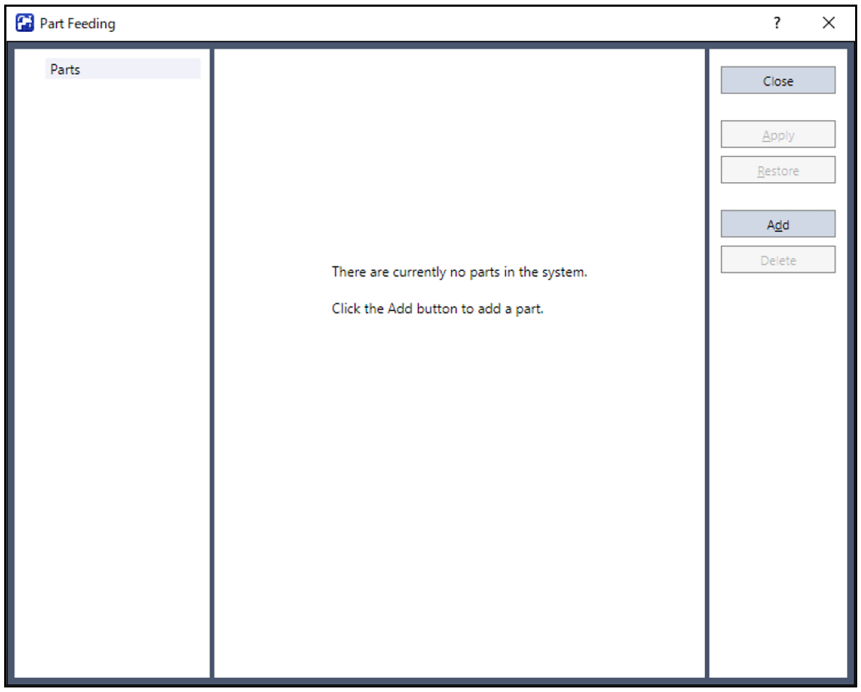
  開始使用

2.8.5 準備Part Feeding用專案

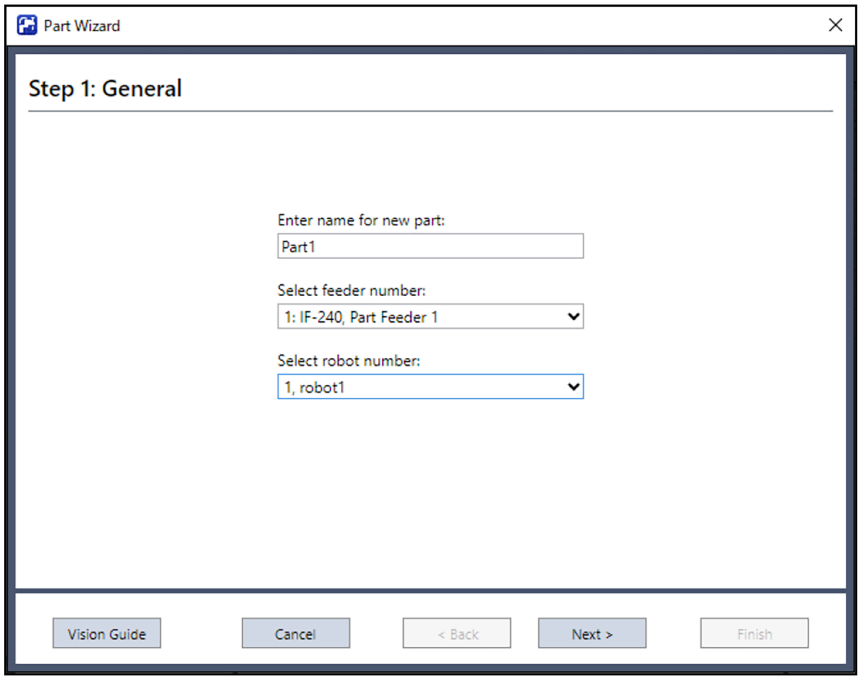
選擇Epson RC+ 8.0功能表 - [專案] - [新建]以建立新專案。
或是開啟現有的專案並建立副本。

2.8.6 建立新零件

1. 選擇Epson RC+ 8.0功能表 - [工具] - [料件送料]。點擊[增加]按鈕。



2. 零件精靈將會啟動。



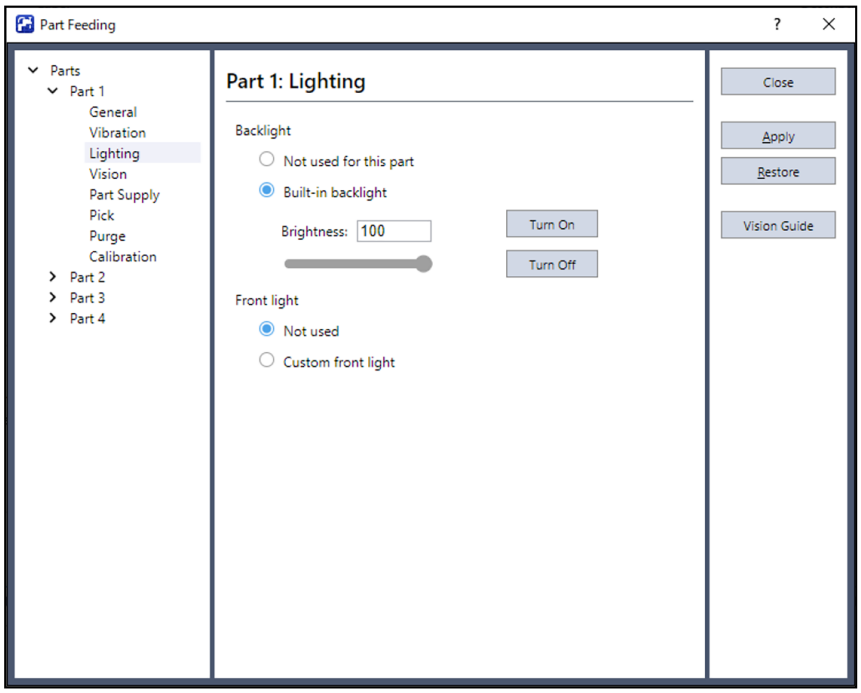
3. 按照零件精靈的指示進行零件設定。關於詳細的設定內容，請參閱以下內容。
   零件精靈
   精靈將會進行基本設定。本教學中，將就此結束精靈並完成設定。

TIP

程式檔案PartFeeding.prg和包含檔案PartFeeding.inc將會添加到專案中。

2.8.7 照明設定

1. 選擇樹狀清單 - [照明]。



2. 點擊[On]按鈕，以確認送料器的背光亮起。

2.8.8 建立視覺序列

建立2個視覺序列。
除了零件檢測用序列外，也請建立送料器校準用的視覺序列。

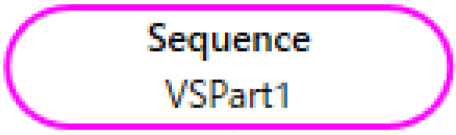
2.8.8.1 建立零件檢測用視覺序列

建立用於檢測零件的視覺序列。

1. 點擊[Vision Guide]按鈕，以顯示Vision Guide畫面。

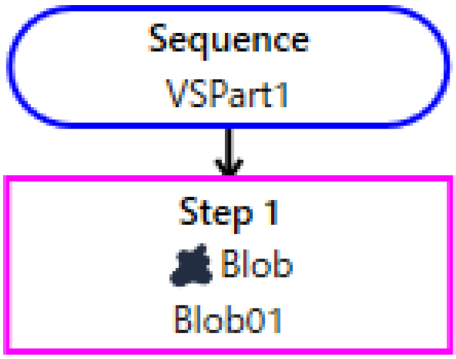
2. 將1個零件放置在送料器上。

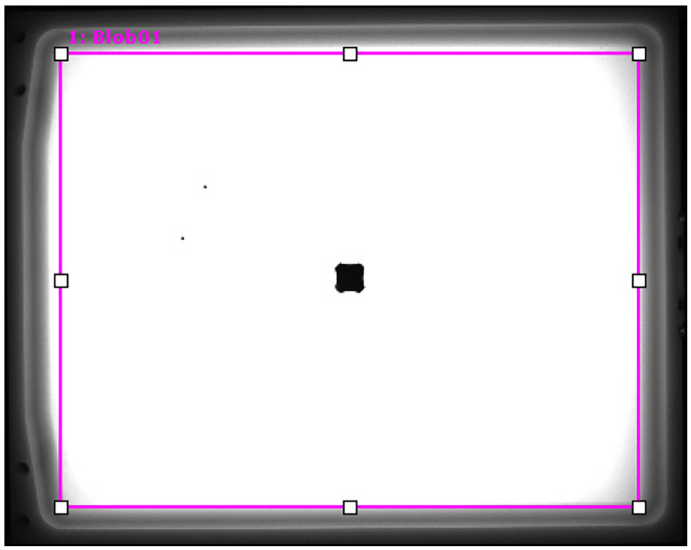
3. 新建1個視覺序列。依照以下內容設定屬性：



屬性	如何設定使用者定義的遠程輸出I/O
Name	設定任意名稱。（例如：VS_Part1）
Calibration	設定視覺校準。
ExposureTime	注意以下事項進行設定： - 零件能清楚識別 - 平台周圍與中央的亮度大致相同

4. 新增1個Blob物件。依照以下內容設定屬性：



屬性	如何設定使用者定義的遠程輸出I/O
SearchWindow	將整個平台設為檢測區域。 
NumberToFind	設定為「All」。
MaxArea	設定為零件面積值的1.3倍左右。
MinArea	設定為零件面積值的0.7倍左右。
ThresholdHigh	設定為能確實檢測零件的值。設定為不會誤檢測平台周邊暗部的值。

TIP

若無法正確檢測零件，請重新調整其他屬性或視覺序列的屬性。

5. 設定完成後點擊[Run]按鈕，並確認可正確識別零件且沒有誤檢測到背景。
6. 設定完成後，點擊Vision Guide功能表 - [File] - [Save]按鈕。將會儲存設定。

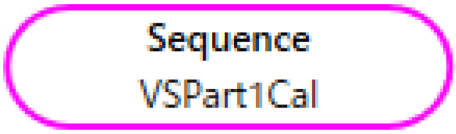
TIP

關於建立零件檢測用視覺序列的詳細資訊，請參閱以下內容。

視覺校準

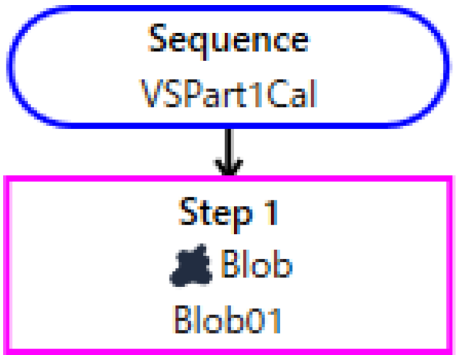
2.8.8.2 建立送料器校準用視覺序列

1. 新建1個視覺序列。依照以下內容設定屬性：

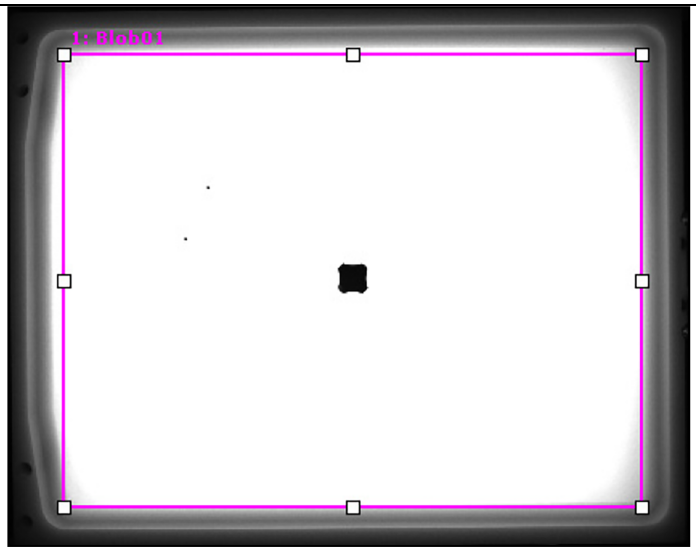


屬性	如何設定使用者定義的遠程輸出I/O
Name	設定任意名稱。(例如：VS_Part1_Cal)
Calibration	設定視覺校準。
ExposureTime	設定時，確保能清楚識別零件。

2. 新增1個Blob物件。依照以下內容設定屬性：



屬性	如何設定使用者定義的遠程輸出I/O
SearchWindow	將整個平台設為檢測區域。

屬性	如何設定使用者定義的遠程輸出I/O
	
MaxArea	保持預設值。
MinArea	將數值設定為零件面積的約0.9倍。按照以下步驟估算零件面積： 1. 讓送料器的背光亮起 2. 在平台上放置數個不重疊的零件 3. 執行視覺序列 4. 以Blob結果中的Area之平均值作為零件面積
NumberToFind	設定為「All」。
ThresholdHigh	設定為能確實檢測零件的值。設定為不會誤檢測平台周邊暗部的值。

TIP

若無法正確檢測零件，請重新調整其他屬性或視覺序列的屬性。

若送料器上有不想檢測的暗部或污漬等，可使用「檢測遮罩」將其從檢測對象中排除。如需詳細資訊，請參閱以下內容。

程式範例 8.3

3. 設定完成後點擊[Run]按鈕，並確認可正確識別零件且沒有誤檢測到背景。
4. 設定完成後，點擊Vision Guide功能表 - [File] - [Save]按鈕。將會儲存設定。
5. 關閉Vision Guide畫面。

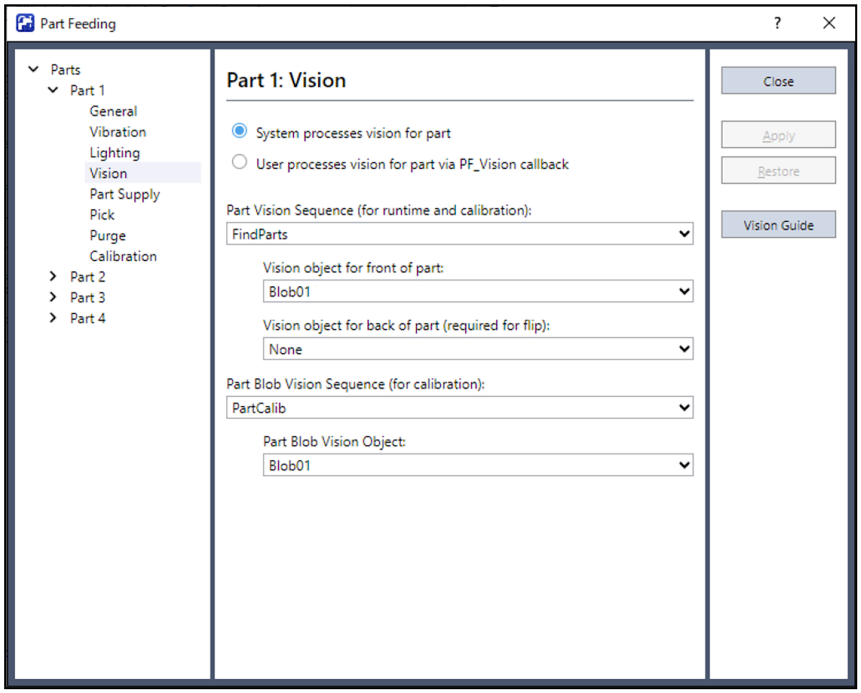
TIP

關於建立送料器校準用視覺序列的詳細資訊，請參閱以下內容。

視覺校準

2.8.9 設定視覺系統

1. 選擇樹狀清單中的[視覺系統]。



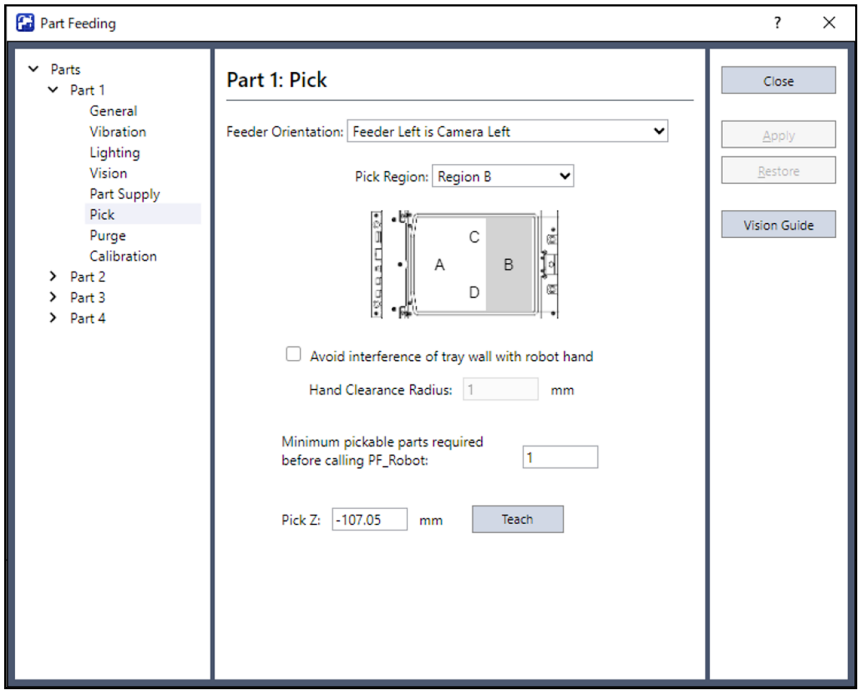
2. 設定以下項目。

項目	如何設定使用者定義的遠程輸出I/O
零件視覺序列	指定在以下項目中建立的視覺序列。 **建立視覺序列**
正面零件的視覺物件	指定包含在以下項目中建立之視覺序列中的Blob物件。 **建立視覺序列**
零件Blob視覺序列	指定在以下項目中建立的視覺序列。 **建立視覺序列**
零件Blob視覺物件	指定包含在以下項目中建立之視覺序列中的Blob物件。 **建立視覺序列**

3. 點擊[套用]按鈕。

2.8.10 拾取設定

1. 選擇樹狀清單中的[拾取]。



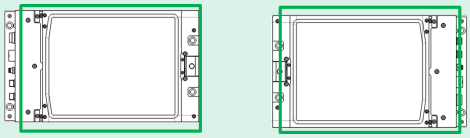
2. 設定以下項目。

項目	如何設定使用者定義的遠程輸出I/O
送料器的方向	選擇攝影機視圖中的送料器設置方向。請正確地設定，以確保攝影機的視圖與送料器方向一致。
拾取區	選擇靠近放置位置之處。 在本步驟中選擇B。 可選擇的項目會因送料器而異。

3. 點擊[示教]按鈕。

TIP

送料器方向可選擇任一方向。



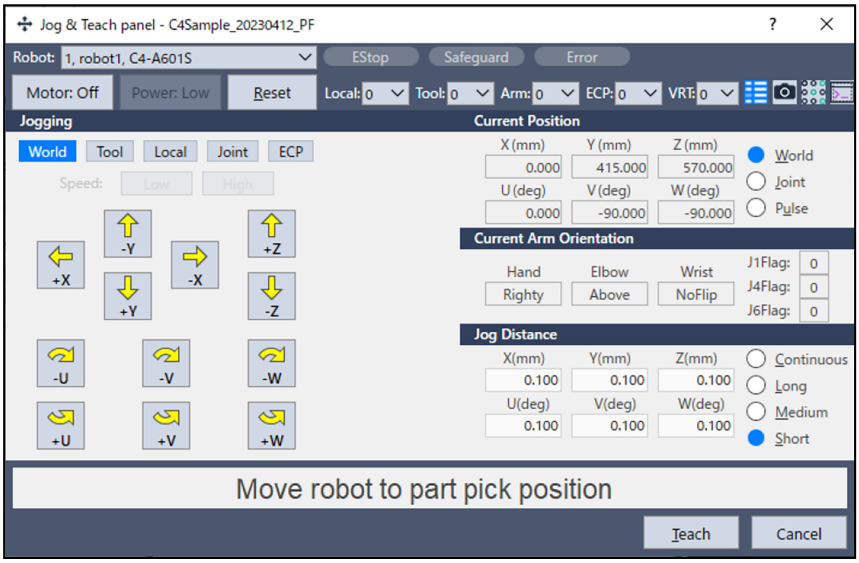
請從A～D中選擇靠近放置位置的拾取區。

IF-240、IF-A1520、IF-A2330的拾取區可從A～D的4個區域與全體中選擇1個。

IF-380、IF-530的拾取區可從A、B的2個區域和全體中選擇1個。

IF-80的拾取區僅能選擇全體。

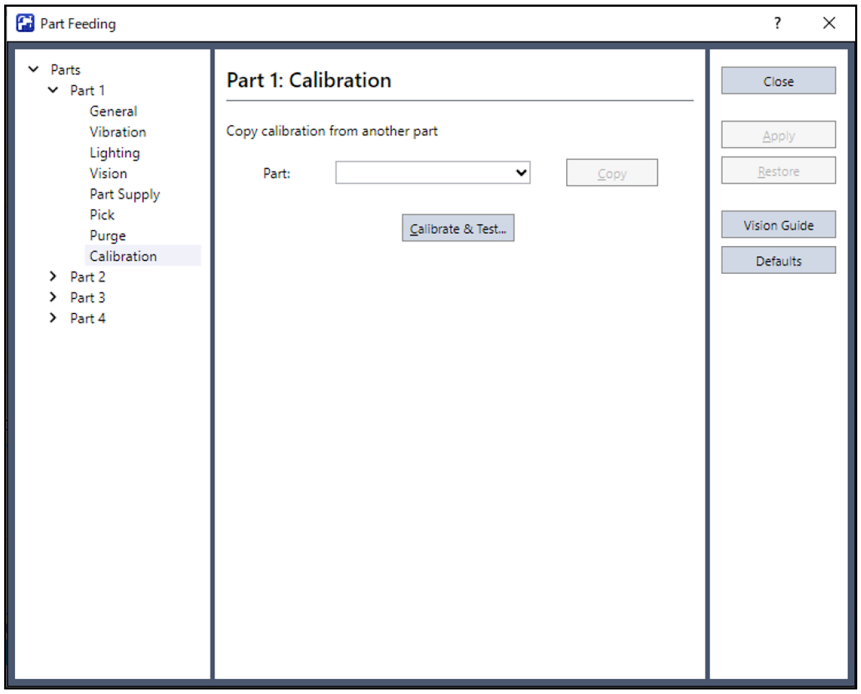
2.8.11 教導拾取Z座標和姿態



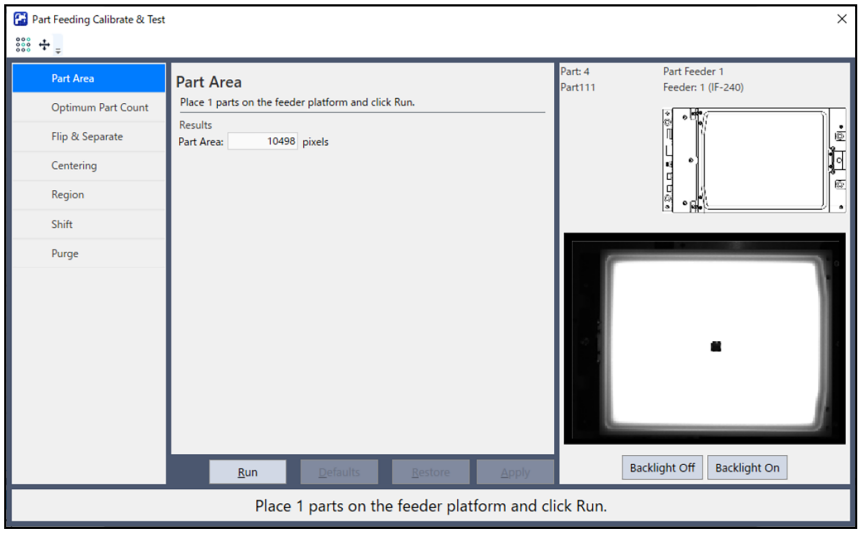
1. 在送料器上放置1個零件。
2. 透過點動操作等方式移動機器人，讓末端夾具接觸送料器上的零件。
   使用夾頭結構的末端夾具將會拾取零件。
3. 點擊[確定]按鈕。
   Z座標將被儲存下來。
4. 點擊[拾取]對話方塊中的[套用]按鈕。

2.8.12 校準&測試

1. 選擇樹狀清單中的[校準]。
   點擊[校準&測試]按鈕。



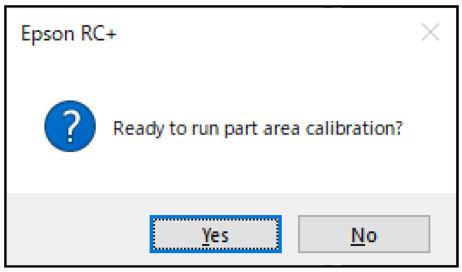
2. 將會顯示[校準&測試]畫面。
   選擇[零件區域]標籤。



3. 在平台中央放置1個零件。

4. 點擊[執行]按鈕。

5. 顯示以下訊息。
   點擊[是(Y)]按鈕。



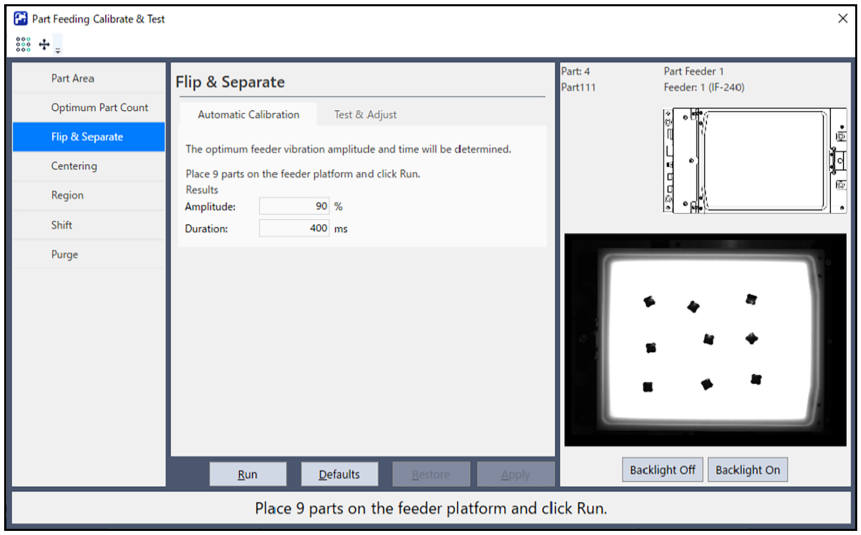
6. 確認「零件面積」欄位所顯示的值。

7. 點擊[套用]按鈕。

提示

下一步的(8)[翻轉&分離]、(14)[區域]的校準雖然並非必要事項，但仍建議執行。

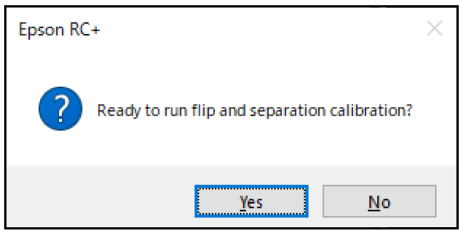
8. 選擇[翻轉&分離]標籤。



9. 在平台上放置所顯示數量的零件。

10. 點擊[執行]按鈕。

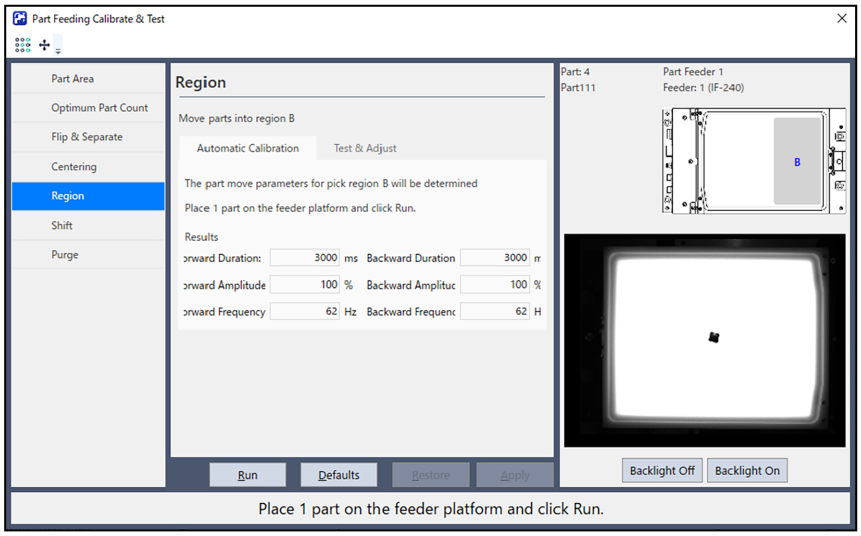
11. 顯示以下訊息。
點擊[是(Y)]按鈕。



12. 在嘗試多次送料器振動和視覺處理後，確認[結果]中的各值已更新。

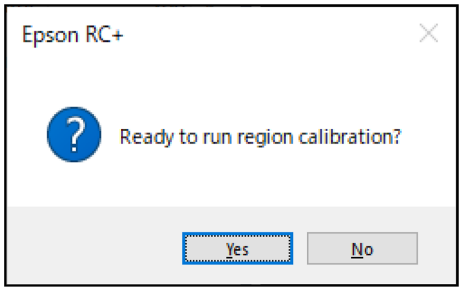
13. 點擊[套用]按鈕。

14. 選擇[區域]標籤。



15. 將放入的零件數量設為1個。

16. 顯示以下訊息。
    點擊[執行]按鈕。



17. 在嘗試多次送料器振動和視覺處理後，確認[結果]中的各值已更新。

18. 點擊[套用]按鈕。

2.8.13 建立Part Feeding程序啟動程式

Part Feeding程序啟動程式中，需編寫機器人的初始化以及開始零件取放的處理。記載內容的具體範例如下所示：

1. 切換到用於零件取放的機器人。
2. 啟動機器人的馬達。
3. 設定機器人的速度、加速度、功率模式等。
   使用Speed、Accel、Power等命令。
   在設定LimZ時，應考量到Z座標、平台的深度（28mm）、越過平台時Z方向的餘裕、送料器周邊設置的單元之Z方向高度。
   Z座標是下列章節中教導的座標。
   教導拾取Z座標和姿態
4. 將機器人移動到可進行視覺成像的位置。
   （使用朝下固定的攝影機時，則移動到不妨礙成像的位置。）
   使用Home等命令。
5. 設定您所用設備的機器。
   例如：料斗、自訂照明的設定等
6. 若要擷取日誌，則執行PF_InitLog。
7. 指定零件ID並執行PF_Start。
   執行PF_Start後會生成任務32，呼叫方將立即恢復控制。

以下為程式範例：在本章節中，在程式檔案main.prg中建立以下函數。（省略5、6的記載）

```
Function test

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

  PF_Start(1)

Fend
```

2.8.14 建立PF_Robot回呼函數

在PF_Robot回呼函數中，編寫機器人拾取和放置零件的動作。此函數的運作方式如下所示：(重複1～7。)

1. 透過零件座標佇列取得平台上的零件座標。
   使用PF_QueGet。
2. 將機器人移動到零件位置。
   使用Jump等命令。(使用SCARA機器人時)
3. 透過啟動吸附等方式來抓住零件。
4. 將機器人移動到放置零件的位置。
   使用Jump等命令。(使用SCARA機器人時)
5. 透過關閉吸附等方式來釋放零件。
6. 刪除零件座標佇列中的1筆資料。
   使用PF_QueRemove。
7. 確認是否發生停止命令。若發生則跳出迴圈。
   使用PF_IsStopRequested。
8. 回傳PF_Robot = PF_CALLBACK_SUCCESS作為回傳值。

以下為PF_Robot回呼函數的範例。
迴圈處理以及1、6、7的處理已記載於自動生成的PartFeeding.prg中，故此處僅記載其餘的處理內容。

程式中使用的標籤等內容如下所示：
IO標籤：Chuck（零件吸附)、UnVacumm（零件釋放）
點位標籤：PlacePos（放置座標）

```
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 進行動作確認

啟動主程式並進行動作確認。

> 提示
>
> 初次運行時，請以低功率、低速度移動機器人，充分確認機器人是否按照指示進行動作。

1. 建立專案。
2. 啟動Run視窗。
3. 執行test函數。
4. 確認機器人有正確地執行拾取和放置零件的動作。

執行零件取放的系統就此設定完成。

3. 軟體篇

3.1 前言

下面說明Epson RC+ 8.0 Part Feeding 8.0 的軟體概要。

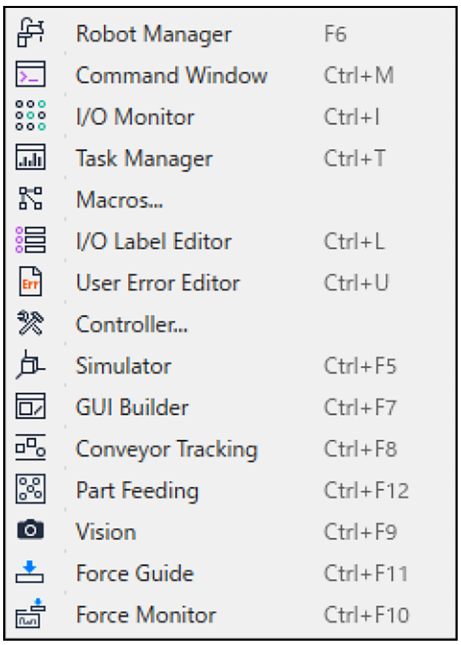
3.1.1 Part Feeding軟體配置

Part Feeding軟體主要由以下3個部分組成：

- Part Feeding視窗
- Part Feeding SPEL+命令
- 回呼函數

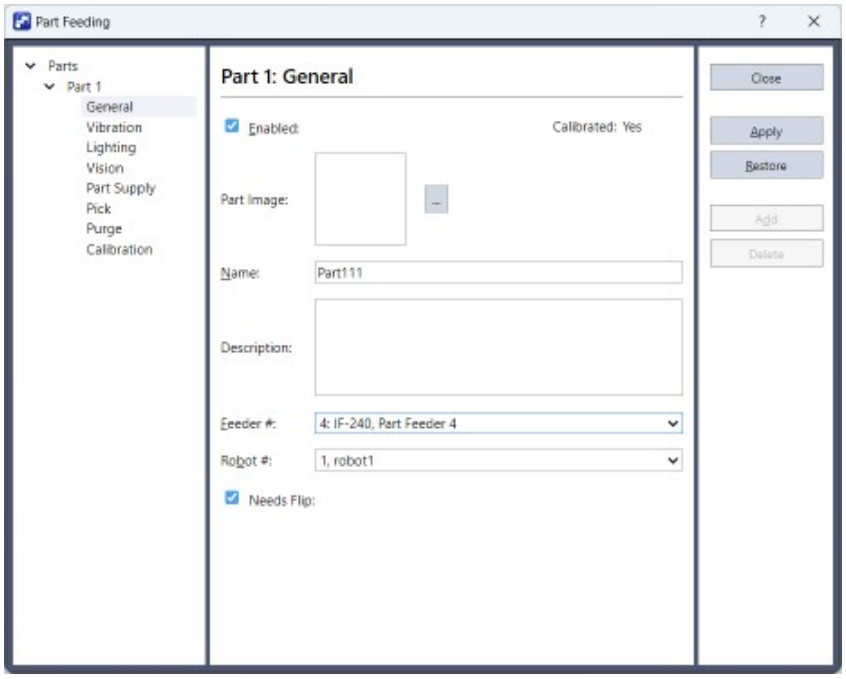
3.1.1.1 [料件送料]視窗

點擊Epson RC+ 8.0功能表 - [工具] - [料件送料]，將會顯示[料件送料]視窗。

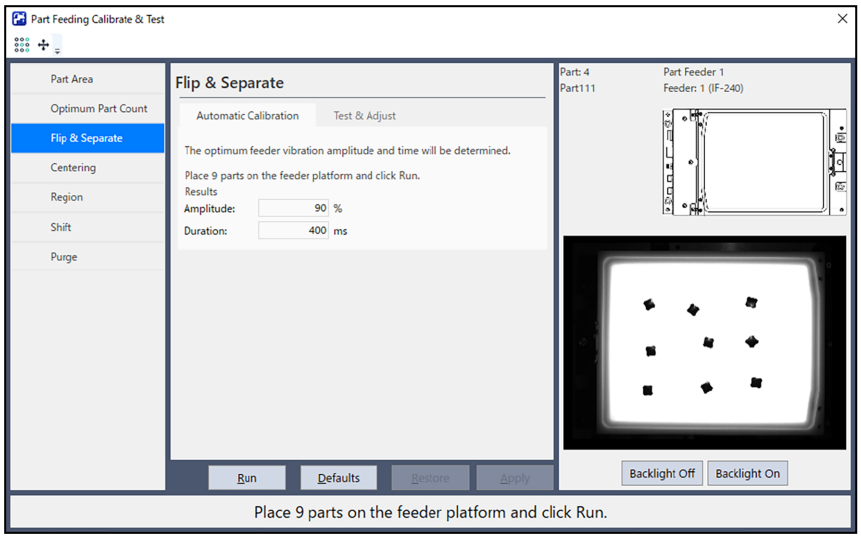


可進行以下操作：

1. 設定零件的各種參數。



2. 執行送料器的校準、手動調整/測試。只要按照畫面上的指示，任何人都能輕鬆完成校準作業。手動調整/測試可以精細地調整送料器參數，只需按一下按鈕即可輕鬆測試。



提示

料件送料視窗只有在RC+連接到控制器的狀態下才會顯示。若在使用虛擬控制器或離線的狀態下嘗試開啟視窗，則會出現錯誤。

3.1.1.2 Part Feeding SPEL+命令

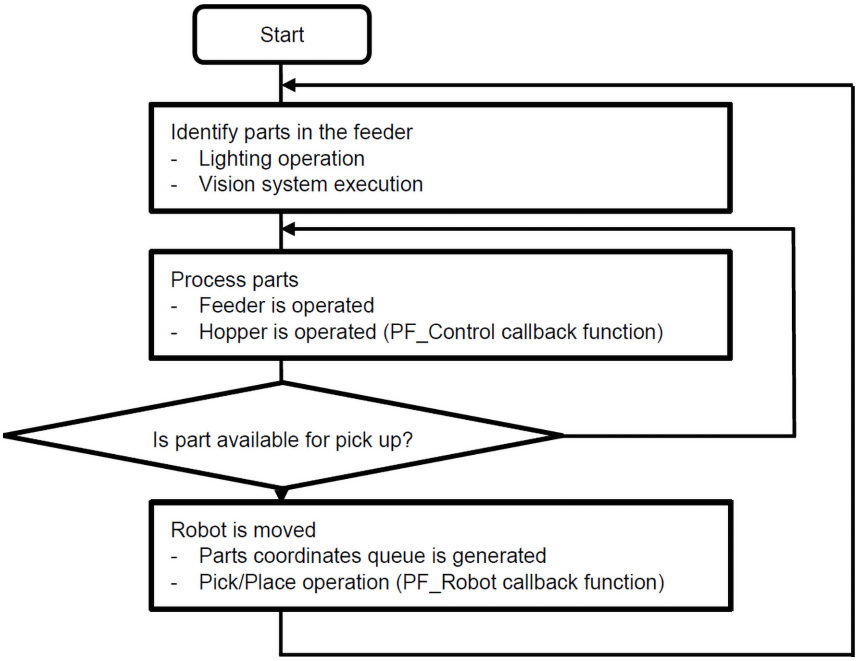
Part Feeding SPEL+命令是為了透過您的程式來執行和控制Part Feeding選配件而提供的SPEL+命令。以下為其中的代表性命令：

命令	描述/用法
PF_Start	啟動Part Feeding程序

命令	描述/用法
PF_Stop	發出Part Feeding程序停止請求
PF_Abort	強制終止Part Feeding程序
PF_QueGet函數	回傳註冊在零件座標佇列（送料器上的零件座標列表）中的座標資料
PF_InitLog	指定日誌檔的輸出路徑
PF_Center	執行送料器的集中動作
PF_Flip	執行送料器的翻轉動作
PF_Shift	執行送料器的位移動作

3.1.1.3 Part Feeding程序

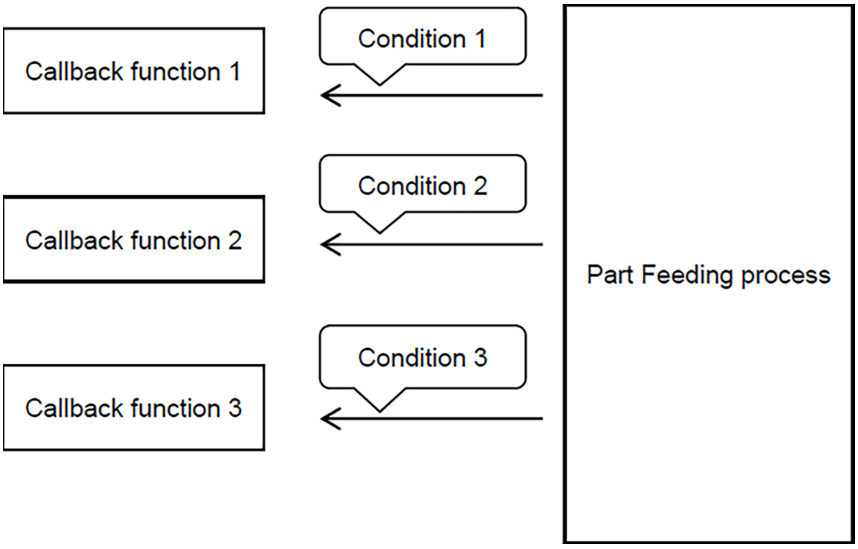
Part Feeding程序是Part Feeding系統內建的運作程序，可自動執行視覺系統及送料器的控制。
Part Feeding程序可透過PF_Start命令啟動。可透過PF_Stop命令等方式停止。
Part Feeding程序的內容如下所示：



1. 識別送料器上的零件
   使用視覺系統來識別平台上的零件數量和分布。
2. 處理零件
   透過控制送料器並移動零件，讓機器人能夠更容易地抓住零件。當零件數量少或沒有零件時，則呼叫PF_Control回呼函數，透過料斗供應零件。
3. 移動機器人
   生成零件座標佇列（送料器上零件座標的列表）。呼叫PF_Robot回呼函數（參閱下一節），執行零件的取放動作。

3.1.1.4 回呼函數

回呼函數是在特定條件下透過Part Feeding程序進行呼叫的SPEL+函數。新增零件後，會自動在專案中生成回呼函數。應由您根據其設備需求編寫回呼函數的內容。



例如，在PF_Robot回呼函數中編寫拾取和放置零件的機器人動作。當送料器上的零件適當分散且機器人處於可拾取狀態時，Part Feeding程序將會呼叫PF_Robot回呼函數。請注意，回呼函數並不是透過您所建立的函數進行呼叫，而是由Part Feeding程序自動呼叫。

以下列出的是回呼函數的清單及其內容說明：

函數名稱	內容說明
PF_Robot	零件的取放動作
PF_Control	您所用設備的機器（料斗、自訂照明）之操作
PF_Status	錯誤處理
PF_MobileCam	使用移動式攝影機時的機器人移動
PF_Vision	您自訂的視覺處理
PF_Feeder	您自訂的送料器動作
PF_CycleStop	發生停止指令時的處理

以下列舉的是呼叫回呼函數的條件：

條件	函數名稱
零件變得可拾取	PF_Robot
零件用盡	PF_Status
自訂照明亮起	PF_Control
將移動式攝影機移動到成像位置	PF_MobileCam
發生錯誤	PF_Status

關於回呼函數的詳細資訊，請參閱以下內容。
Part Feeding回呼函數

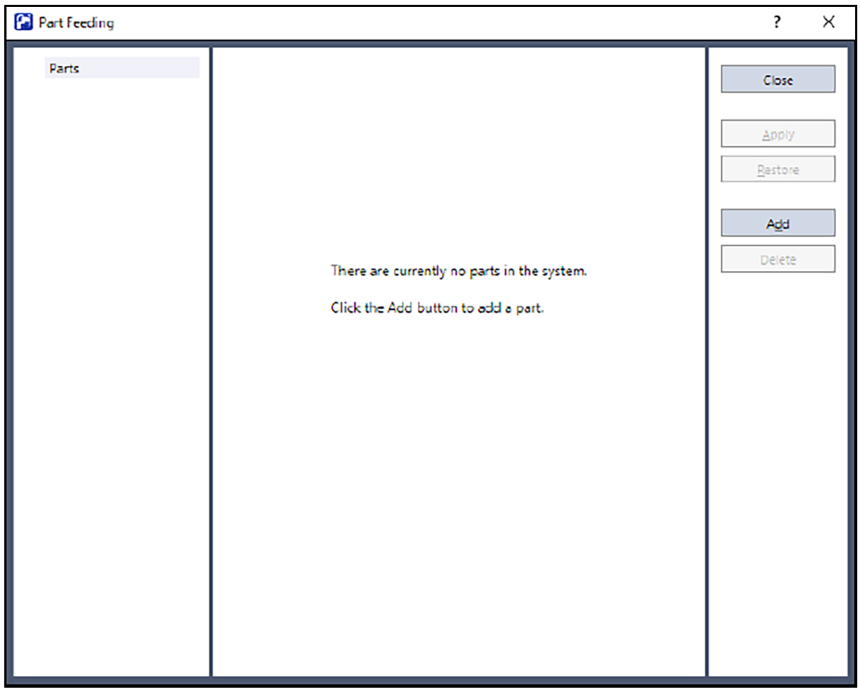
3.1.2 Part Feeding用專案

下面說明Part Feeding用SPEL專案。

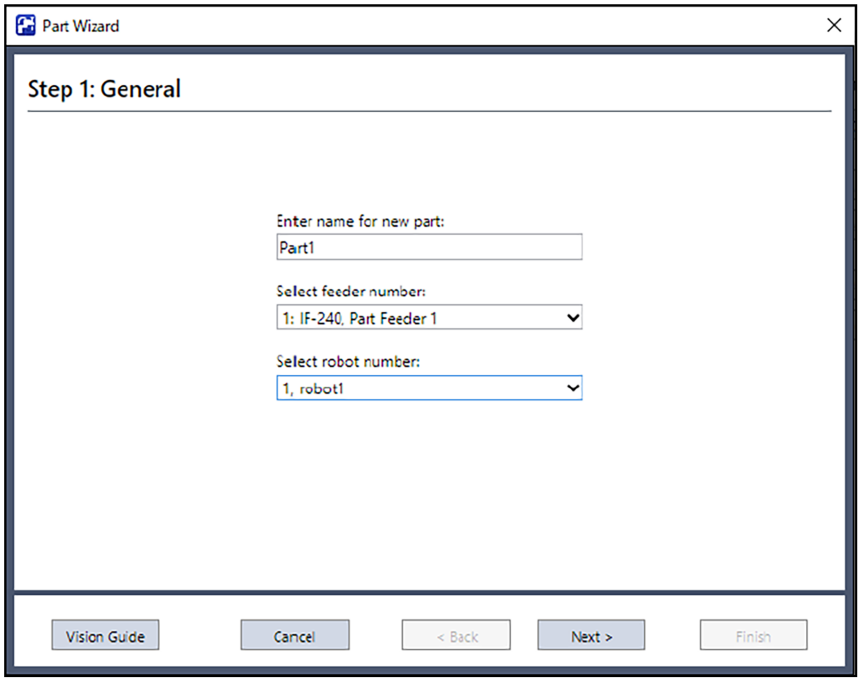
3.1.2.1 將Part Feeding選配件套用到專案中

若要將Part Feeding選配件套用到專案中，請按照以下步驟操作：

1. 開啟要套用選配件的現有專案。或是建立新專案。
2. 點擊Epson RC+ 8.0功能表 - [工具] - [料件送料]。
   將會顯示以下畫面。



3. 點擊[增加]按鈕。
   零件精靈將會啟動。

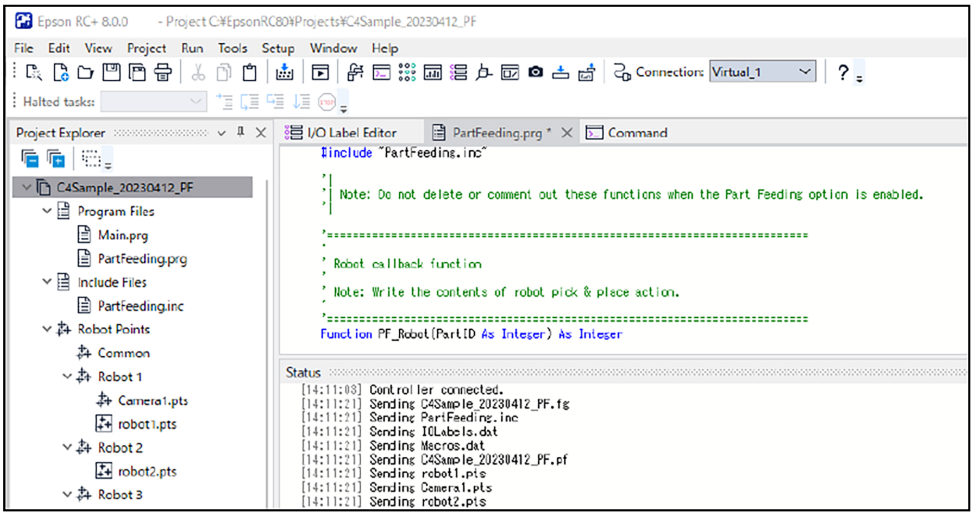


4. 按照零件精靈的指示進行零件設定。
   關於詳細的設定內容，請參閱以下內容。
   零件精靈
   將建立1個零件。

   此時將會把Part Feeding選配件專用的程式檔案新增到專案中。請參閱以下內容。
   專案的配置

3.1.2.2 專案的配置

下面說明啟用Part Feeding選配件時所新增的部分。
在Part Feeding視窗中新設定零件時，會新增以下程式檔案：



- PartFeeding.prg
  記載了回呼函數的範本程式碼。
- PartFeeding.inc
  記載了在使用Part Feeding選配件進行程式設計時所使用的常數。

提示

請勿從專案中排除或刪除這些檔案。否則將導致建置錯誤。

提示

在使用以下語言設定時，程式中的註解會使用相同於語言設定的語言：

- 日文
- 德文
- 西班牙文

而在其他語言設定下，註解將會使用英文。

3.1.2.3 組態檔案

作為組織專案的檔案，將新增以下檔案：
PF檔案
記載零件設定的檔案。檔案名稱為[專案名稱].pf。

提示

請勿直接使用編輯器編輯檔案。否則檔案可能會變得無法讀取並出現錯誤。

3.1.2.4 匯入檔案

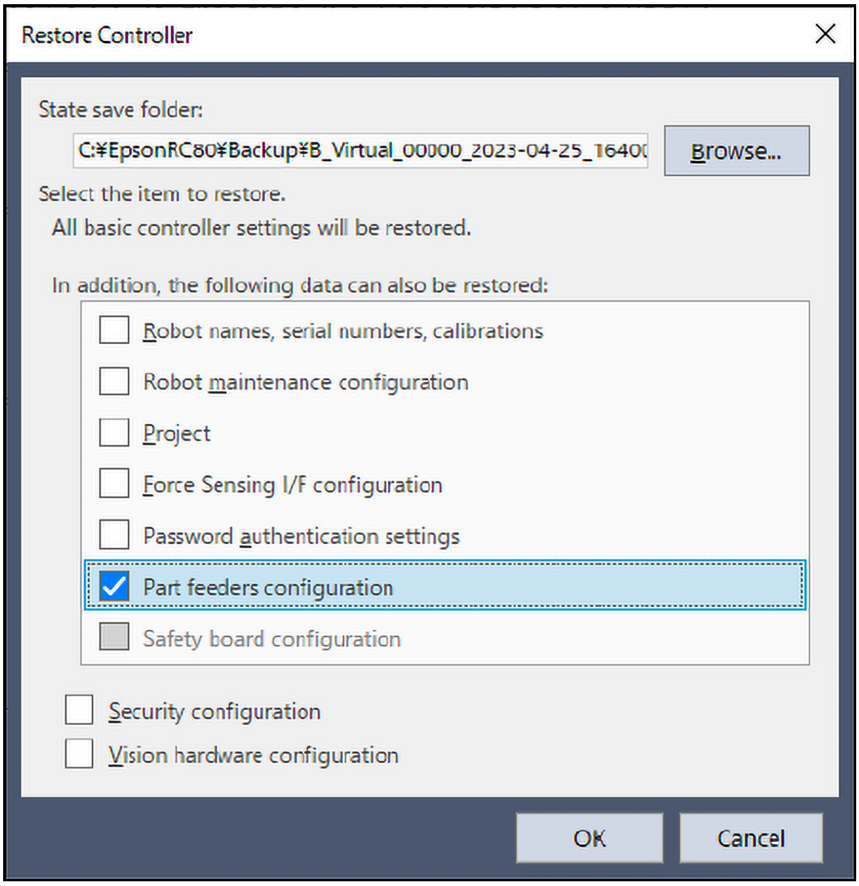
可匯入PF檔案。透過此功能，即可將零件資訊複製到其他專案中。
如需詳細資訊，請參閱以下內容。
[匯入](檔案功能表)

3.1.2.5 控制器的備份、還原

若要備份控制器設定，則選擇Epson RC+ 8.0功能表 - [工具] - [維護]中的[控制器設定備份]。
此時也會一起備份Part Feeding選配件的資料。
若要還原已備份的控制器設定，則選擇Epson RC+ 8.0功能表 - [工具] - [維護]中的[還原控制器設定]。
若有在以下畫面中勾選[零件送料器配置]，Part Feeding選配件的資料也會一併還原。

提示

為了故障排除而需要將備份發送給本公司時，請透過Epson RC+8.0來取得備份。若使用控制器主體的記憶體埠之備份，則不會備份與送料器使用狀況相關的資料。



3.1.3 SPEL程式設計

3.1.3.1 程式設計的概要

下面說明Part Feeding程式設計的概要。詳細內容將在本章之後說明。
在大多數的應用程式中，送料器的運作是由系統自動執行的。此外，基本的SPEL程式是自動生成的。該程式由生成的SPEL函數（稱為回呼函數）所組成。
執行時，由系統呼叫回呼函數。滿足以下條件時，會進行呼叫：

PF_Robot函數	送料器上有可拾取的零件時
PF_Status函數	狀態改變時
PF_Vision函數	在使用PF_Vision callback進行視覺處理時
PF_Feeder函數	進行自訂的送料器控制時
PF_Control函數	控制料斗或前側光等時
PF_MobileCam函數	使用移動式攝影機，為了檢測零件而將攝影機移動到送料器上方時
PF_CycleStop函數	停止Part Feeding的運作時

若為了處理特定需求，則需要修改回呼函數。
在大多數應用程式中，需要使用PF_Robot和PF_Status回呼函數。
以下為各回呼函數的概要：

PF_Robot回呼函數
當送料器的運作和視覺處理完成且判斷可以拾取零件時，會自動執行PF_Robot回呼函數。編寫實現零件取放動作的

程式碼。在單純的應用程式中，會編寫幾行實現機器人取放動作和夾爪（抓住零件的結構）運作的程式碼。
PF_Robot回呼函數結束後，將會執行PF_Status回呼函數。

PF_Status回呼函數
各回呼函數完成後，會自動執行PF_Status回呼函數。PF_Status將會根據各回呼的回傳值，向系統指示繼續執行的方法。或是PF_Status將向操作員通知發生錯誤狀態。該錯誤內容可能是機器人過載扭矩錯誤等系統層級的錯誤，或是料斗供料太少、太多等Part Feeding的錯誤。PF_Robot回呼函數中需記載如何因應發生的錯誤。針對簡單的應用程式，無需修改PF_Status回呼函數的程式碼。

而針對更複雜的應用程式，則可能需由您自行編寫視覺處理的程序。於此情況下，則需要使用PV_Vision回呼函數。
以下為PF_Vision回呼函數的概要：

PF_Vision回呼函數
若要由您進行零件送料器的視覺處理而非由系統進行視覺處理，則在PF_Vision回呼函數中編寫必要的處理。需要從攝影機擷取影像、執行檢測零件的序列、取得結果，並將結果新增到零件座標佇列中。在簡單的應用程式中，會由系統進行視覺處理，因此無需編寫PF_Vision回呼函數。
如需詳細資訊，請參閱以下內容。
PF_Vision

在極少數情況下，需要使用您自行準備的照明而非內建的背光。PF_Control回呼函數用於控制您準備的照明。而您所準備料斗的控制也是透過PF_Control回呼函數執行的。此外，本公司料斗（選購品）的控制也是透過PF_Control回呼函數執行的。以下為PF_Control回呼函數的概要：

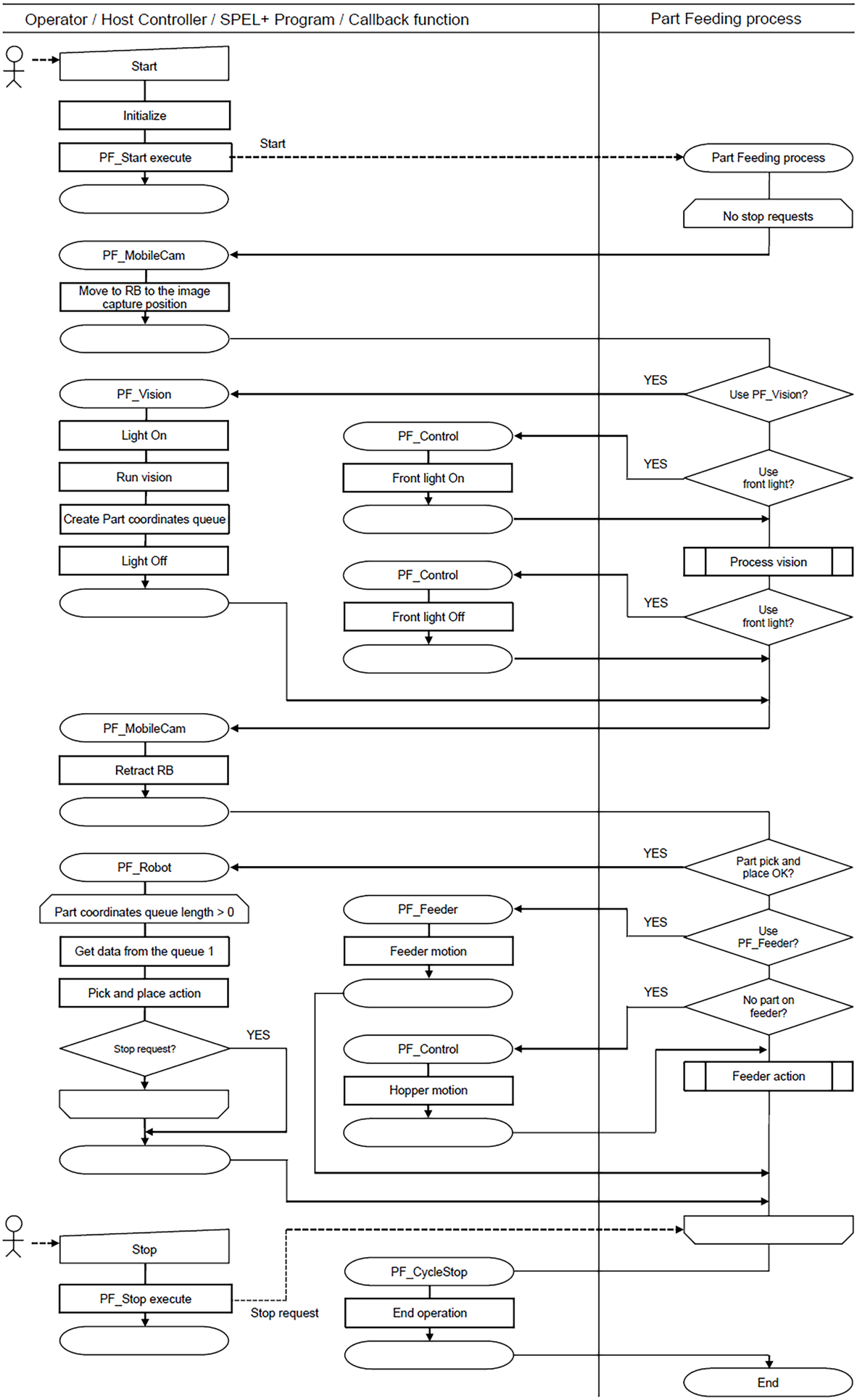
PF_Control回呼函數
當系統判斷需要從料斗供料，或判斷需要開啟/關閉照明時，將會自動執行PF_Control回呼函數。使用EPSON的料斗（選購品）時，需要對PF_Control回呼函數中註解掉的PF_OutputONOFF命令取消註解。

PF_MobileCam回呼函數
若要使用安裝在機器人上的移動式攝影機，而非安裝在送料器上方的朝下攝影機，則會自動執行PF_MobileCam回呼函數。在PF_MobileCam回呼函數中，編寫讓機器人將攝影機移動到送料器上方的程式碼。此外，也要編寫在檢測到零件並將零件資料新增到Part Feeding佇列後，讓機器人從送料器上方退避的動作。使用朝下固定的攝影機時，無需編寫PF_MobileCam回呼函數。

PF_CycleStop回呼函數
執行PF_Stop後，會自動執行PF_CycleStop回呼函數。PF_CycleStop回呼函數可用於編寫結束處理的程序，例如將機器人退避到安全位置或開啟/關閉輸出等操作。在簡單的應用程式中，無需編寫PF_CycleStop回呼函數。

PF_Feeder回呼函數
當您的設備或零件無法透過自動送料器控制進行處理時，則可在PF_Feeder回呼函數中使用PF_Flip命令或PF_Shift命令等，以SPEL程式碼編寫送料器運作。一般而言，不需要編寫PF_Feeder回呼函數。

所有回呼函數都需要設定回傳值。若要透過函數回傳數值，則將數值指派給函數名稱（例如，PF_Robot = PF_CALLBACK_SUCCESS，其中PF_CALLBACK_SUCCESS是定義為值0的常數）。正常結束時，回傳值為PF_CALLBACK_SUCCESS。PF_Status會在任一回呼函數完成或發生錯誤之後，向系統指示繼續執行的方法。系統需要辨別欲執行的操作（繼續、結束、重新啟動等）。例如，可藉由將回傳值設為常數PF_EXIT的方式來結束。在簡單的應用程式中，可以直接使用自動生成的回呼函數之回傳值。
如需詳細資訊，請參閱以下內容。
Part Feeding回呼函數
下一頁為執行取放時的程式整體流程圖。


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
YES
Use PF_Vision?
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
YES
Part pick and place OK?
PF_Robot
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 啟動Part Feeding程序

啟動Part Feeding程序所需的處理如下所示：

1. 將機器人移動到可進行視覺成像的位置。
   （使用朝下固定的攝影機時，則移動到不妨礙成像的位置。）
2. 執行PF_Start命令，啟動Part Feeding程序。

以下為程式範例：

```
Function StartPickPlace()

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

Fend
```

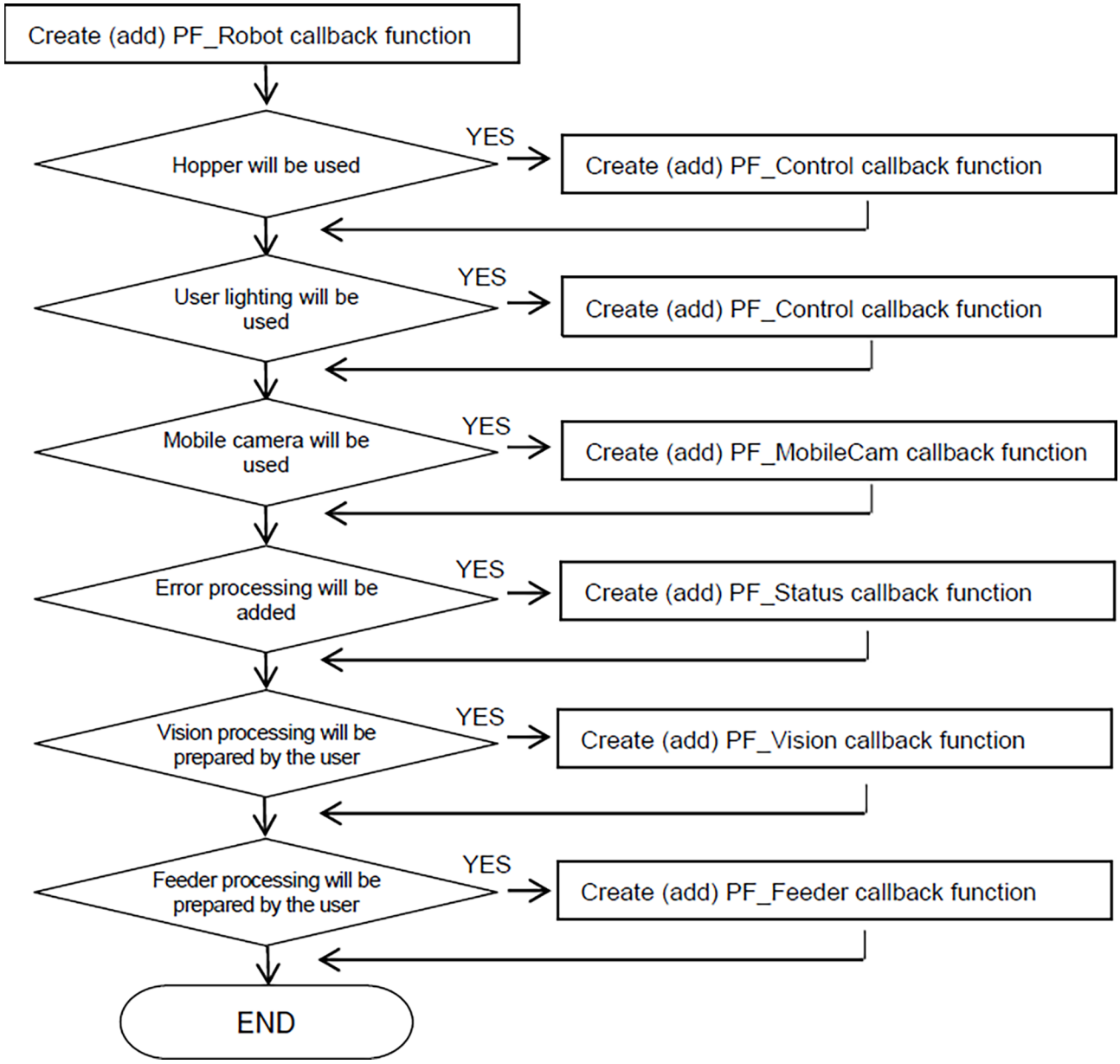
提示

編寫以下處理時，應確保在啟動設備時執行：

- 零件取放時的機器人速度、加速度、功率模式等的設定
- 料斗的設定、自訂照明的設定、末端夾具姿態的初始化等

3.1.3.3 取放處理

以下是建立取放處理程式的程序：



PF_Robot回呼函數的建立（追加）屬於必要作業。其他項目則會根據您的設備規格需求進行。

PF_Robot回呼函數

在PF_Robot回呼函數中，編寫機器人拾取和放置零件的動作。此函數的運作方式如下所示：(重複1～7。)

1. 透過零件座標佇列取得平台上的零件座標。
   使用PF_QueGet。
2. 將機器人移動到零件位置。
   使用Jump等命令。(使用SCARA機器人時)
3. 透過啟動吸附等方式來抓住零件。
4. 將機器人移動到放置零件的位置。
   使用Jump等命令。(使用SCARA機器人時)
5. 透過關閉吸附等方式來釋放零件。
6. 刪除零件座標佇列中的1筆資料。
   使用PF_QueRemove。
7. 確認是否發生停止命令。若發生則跳出迴圈。
   使用PF_IsStopRequested。
   程式設計範例請參閱以下內容。

- 建立PF_Robot回呼函數
- 應用程式範例

向送料器供應零件（PF_Control）的選配功能

編寫使用料斗向送料器供應零件時的運作程式。在PF_Control回呼函數（PartFeeding.prg）中編寫。
程式內容如下所示：

1. 編寫平台上沒有任何零件時的料斗運作（零件供料）。
   此時，PF_Control回呼函數的引數Control為PF_CONTROL_SUPPLY_FIRST。
2. 編寫將零件放入平台時的料斗運作（零件供料）。
   此時，PF_Control回呼函數的引數Control為PF_CONTROL_SUPPLY。
   程式設計範例請參閱以下內容。
   建立PF_Robot回呼函數

自訂照明的操作（PF_Control）選配功能

編寫使用自訂照明時的運作程式。
在PF_Control回呼函數（PartFeeding.prg）中編寫。
程式內容如下所示：

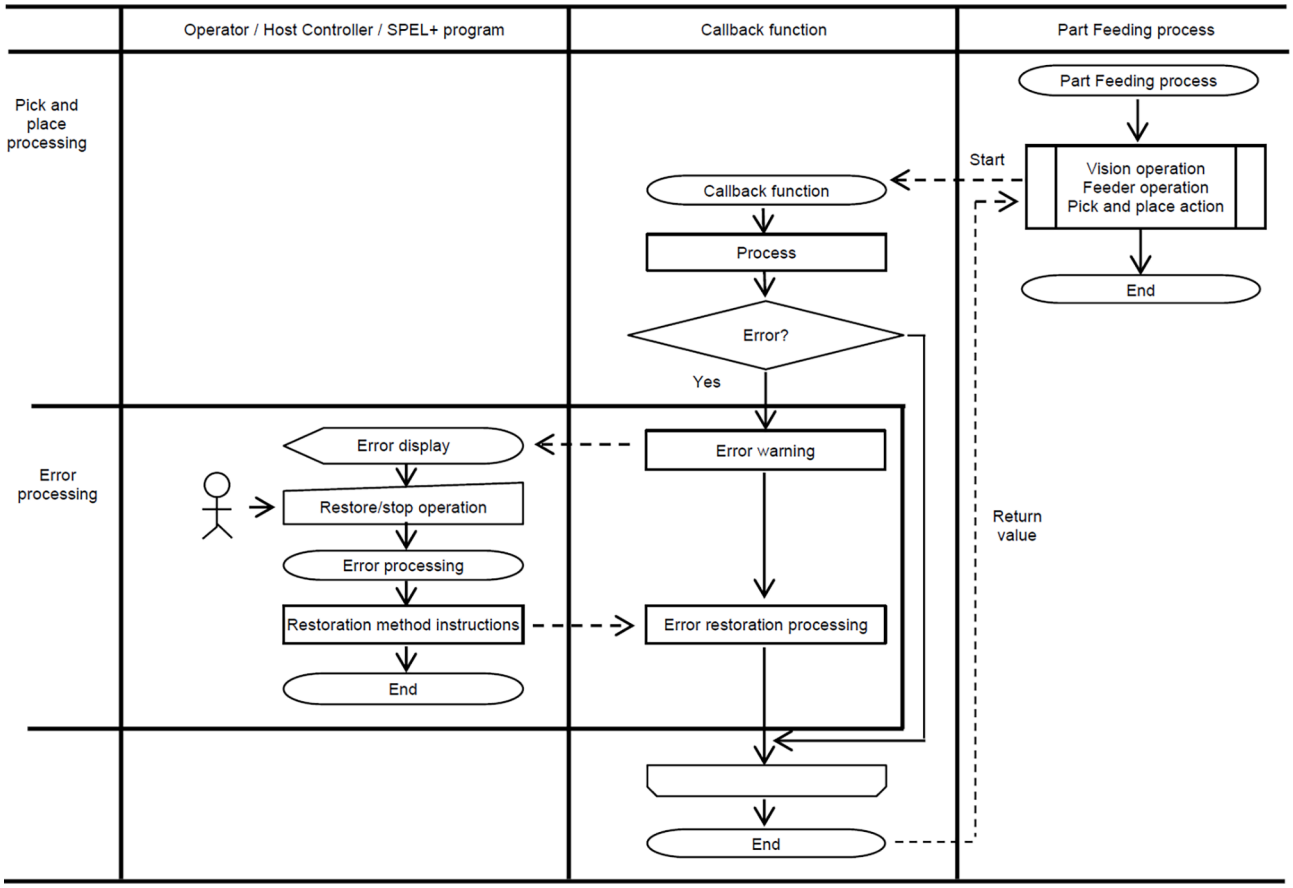
1. 編寫自訂照明亮起的動作。
   此時，PF_Control回呼函數的引數Control為PF_CONTROL_LIGHT_ON。
2. 編寫關閉自訂照明的動作。
   此時，PF_Control回呼函數的引數Control為PF_CONTROL_LIGHT_OFF。
   程式設計範例請參閱以下內容。
   建立PF_Robot回呼函數

> 提示
>
> 請同時編寫自訂照明亮起和關閉的動作。若只編寫亮起動作，則可能會使視覺系統無法正確識別零件而出現錯誤。

3.1.3.4 錯誤處理

下面說明Part Feeding選配件運作中發生錯誤時的處理方法。

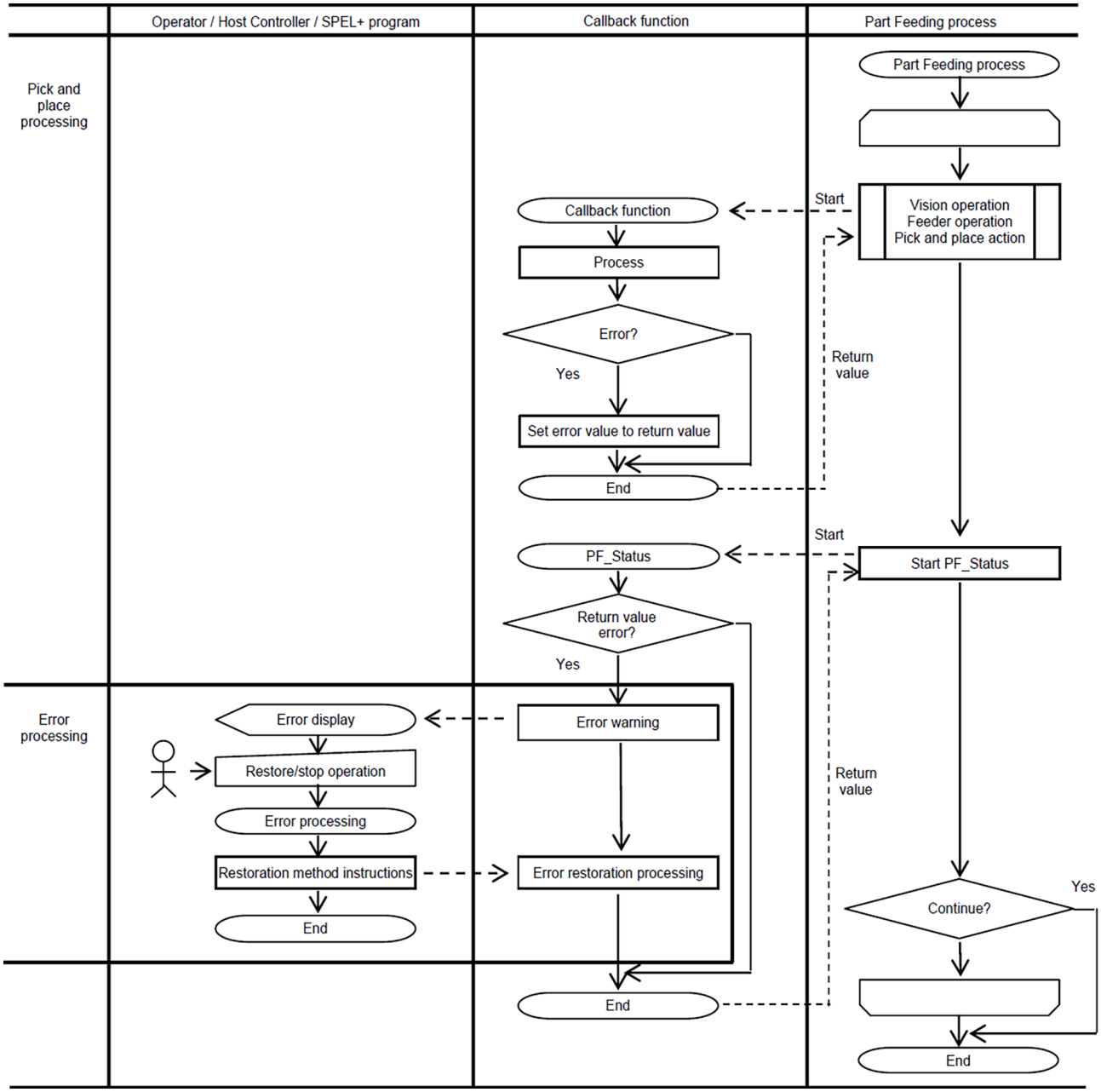
1. 在回呼函數中處理錯誤



在回呼函數中檢測並處理回呼函數中發生的錯誤。若要在不使Part Feeding程序恢復控制的情況下繼續執行，請使用此方法處理。

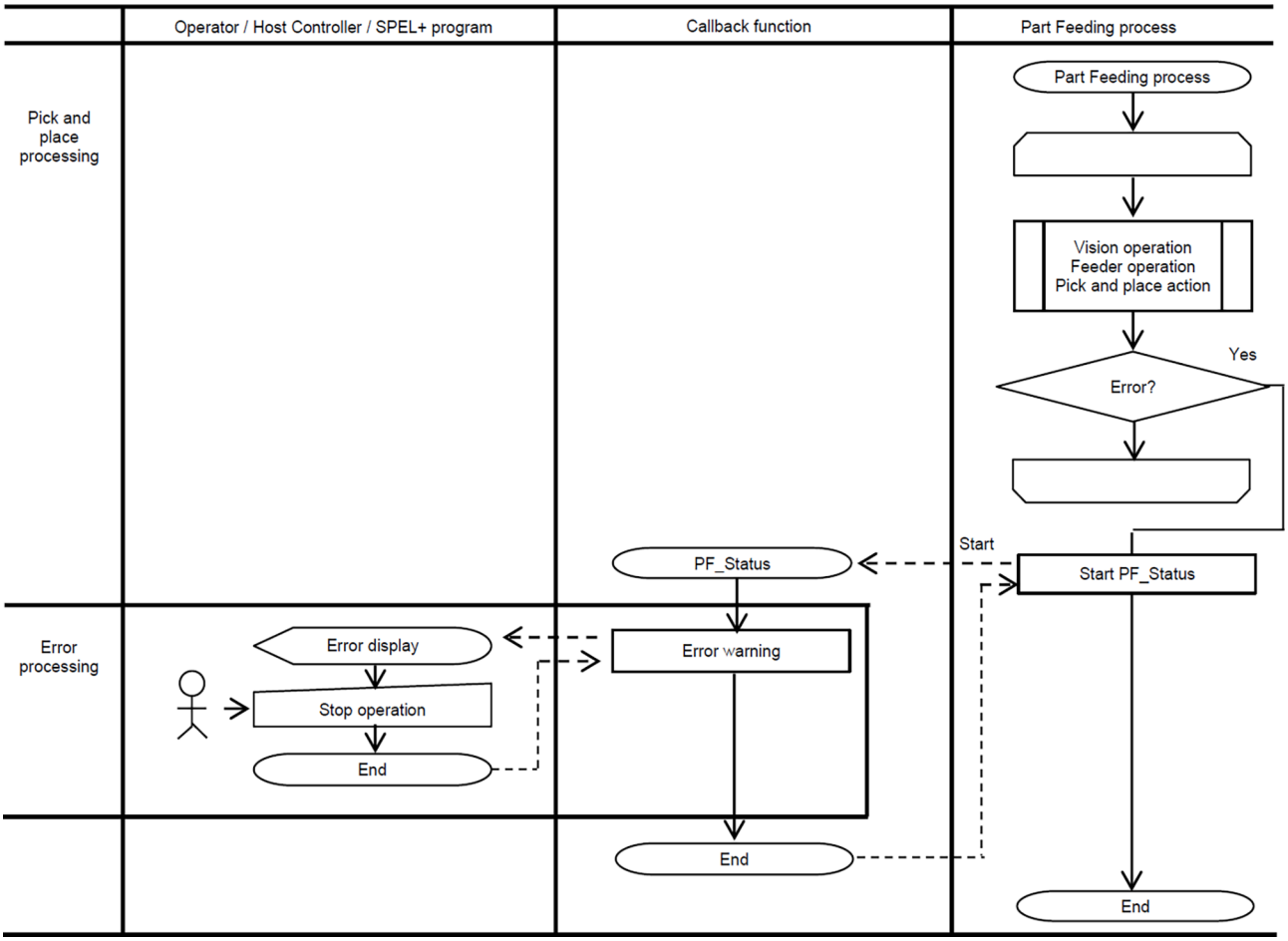
例如：在PF_Robot回呼函數中發生零件吸附錯誤時執行重試

2. 在PF_Status回呼函數中處理錯誤



當回呼函數中發生錯誤時，則將回傳值設為非PF_CALLBACK_SUCCESS的值（使用者錯誤8000-8999）。
Part Feeding程序會將此值設為PF_Status回呼函數的引數Status並啟動。
若想統一錯誤處理，請使用此方法處理。
如需詳細資訊，請參閱以下內容。
PF_Status

3. 處理Part Feeding程序內部發生的錯誤



可能會因為Part Feeding參數設定不當（PF_Start引數的指定不當）或視覺設定不當等因素，使Part Feeding程序內部發生錯誤。
Part Feeding程序會將PF_STATUS_ERROR設為引數Status並啟動。
PF_Status函數結束後，Part Feeding程序也會結束。
如需詳細資訊，請參閱以下內容。

- PF_Status
- 錯誤處理

3.1.3.5 終止處理

若要透過您的程式終止Part Feeding程序，可採取下列任一方法：

1. 執行PF_Stop命令
   從您的程式執行PF_Stop命令。
   Part Feeding程序將會在目前處理中的回呼函數及PF_CycleStop回呼函數結束之後終止。
   詳細資訊請參閱以下手冊。
   「Part Feeding SPEL+命令參考手冊 - PF_Stop」
2. 執行PF_Abort命令
   透過您的程式執行PF_Abort命令。
   Part Feeding程序將會立即終止。
   此時，目前處理中的回呼函數也會立即終止。
   詳細資訊請參閱以下手冊。
   「Part Feeding SPEL+命令參考手冊 - PF_Abort」

提示

開啟安全門或執行Pause時，送料器和料斗的振動會停止。即使關閉安全門或執行Continue以繼續操作，振動也不會重新開始。

3.1.3.6 Part Feeding程序所使用的功能

Part Feeding程序運作期間會占用以下功能。當Part Feeding程序運作時，請避免透過您的程式使用這些功能。

送料器編號	任務	計時器	SyncLock
1	32	63	63
2	31	62	62
3	30	61	61
4	29	60	60
系統（已預約）	28	-	59

滿足以下條件時可使用這些功能：

- Part Feeding程序停止時
- 不使用多台送料器時（例如：僅連接1台送料器時，可使用占用送料器編號2以後之裝置的功能。）

提示

請勿使用已預約的系統任務和SyncLock命令。

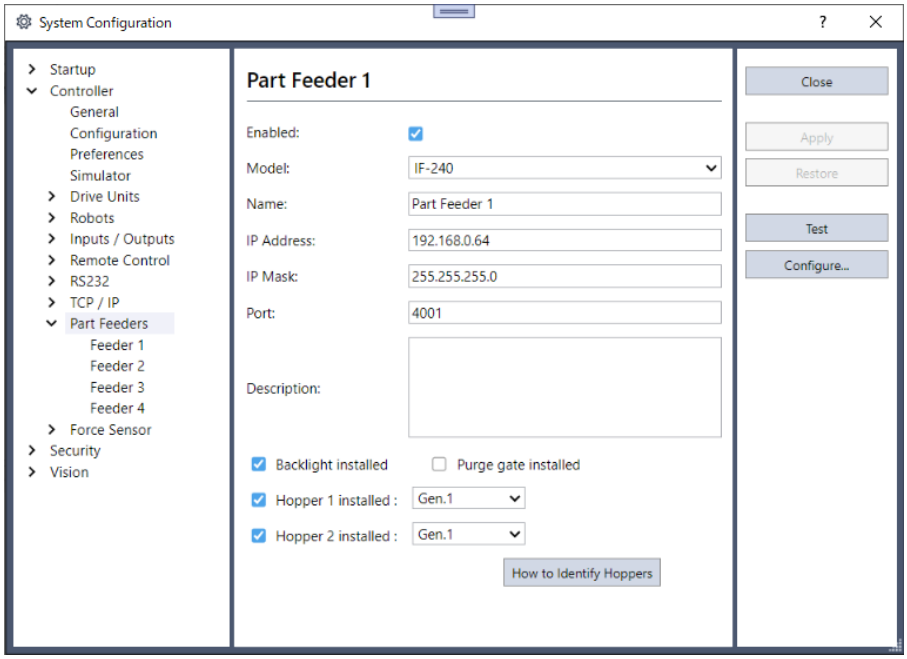
3.2 Part Feeding GUI

下面說明Epson RC+ 8.0 Part Feeding 8.0的GUI。

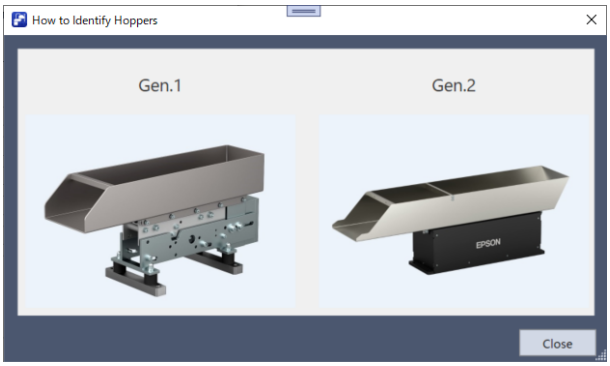
3.2.1 開始使用

在Epson RC+ 8.0功能表 - [設置] - [系統配置]中，進行將送料器連接到控制器的設定。

3.2.1.1 零件送料器畫面



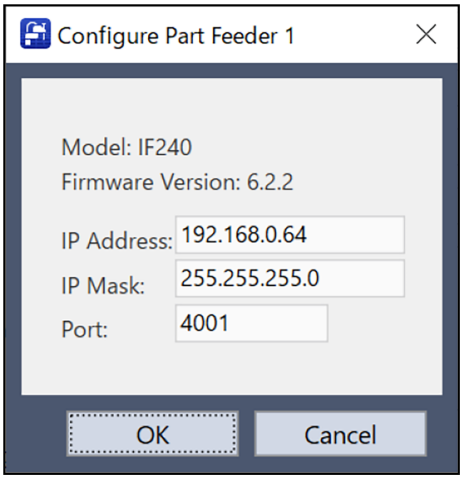
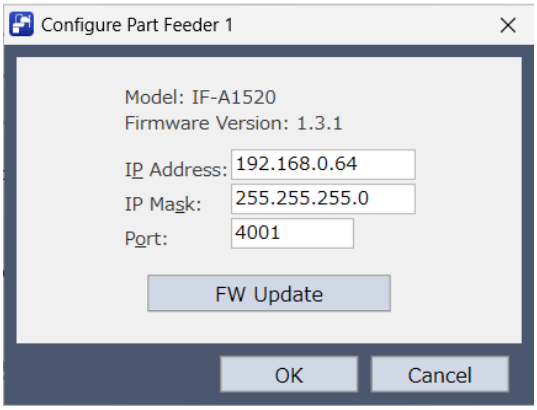
項目	描述
啟用	若要啟用送料器，則勾選核取方塊。
型號	選擇送料器的型號。
送料器名稱	設定任意名稱。（半形英數字和底線。最多32個字元）
IP位址	輸入目前設定在送料器上的IP位址。 預設IP位址為192.168.0.64。
子網路遮罩	輸入目前設定在送料器上的子網路遮罩。 預設的子網路遮罩為255.255.255.0。
連接埠	輸入目前設定在送料器上的連接埠編號。 預設的連接埠編號為4001。
註解	填寫送料器的說明（註解）。屬於選項。（半形英數字和底線。最多256個字元）
安裝背光	若有安裝送料器內建的背光，則勾選核取方塊。
安裝清除閘門	若要使用送料器選配件的清除閘門，則勾選核取方塊。（僅適用於IF-240、IF-380、IF-530、IF-A1520、IF-A2330）
安裝料斗1 安裝料斗2	有連接料斗時，則勾選核取方塊。從Gen.1/Gen.2中選擇料斗類型。關於料斗類型的詳細資訊，請參閱以下內容。 料斗

項目	描述
料斗的區分方法	顯示料斗類型。 

提示

- 若要變更送料器的網路設定，請點擊以下說明的[設定]按鈕。
- 若輸入目前未設定在送料器上的IP位址、子網路遮罩、連接埠，與送料器通訊時將會出現錯誤。點擊[套用]按鈕時不會發生錯誤。
- 「安裝清除閘門」的設定會影響振動參數。使用清除閘門時，請在透過料件送料對話方塊新增零件之前，開啟「安裝清除閘門」的設定。新增零件後再勾選核取方塊，可能會使送料器無法正常運作。
- 使用IF-80時，「安裝料斗」和「料斗類型」會自動採取以下設定：設定無法變更。
  - 安裝料斗1：已選；安裝料斗2：取消選擇
  - 料斗1類型：Gen.2；料斗2類型：空白
- 使用IF-240且同時選擇「安裝料斗1」、「安裝料斗2」時，若選擇「安裝清除閘門」，則會顯示出只能使用1台料斗的通知訊息。若選擇[否]，將會恢復為原設定；若選擇[是]，將會取消選擇「安裝料斗2」。
- 使用IF-240且選擇「安裝清除閘門」和「安裝料斗1」時，若選擇「安裝料斗2」，則會顯示出在停用清除閘門時才能使用2台料斗的通知訊息。若選擇[否]，將會恢復為原設定；若選擇[是]，將會取消選擇「安裝清除閘門」。
- 當開啟在EPSON RC+ 7.0中建立的Part Feeding專案時，連接的料斗會被識別為料斗1，料斗1的類型則會採取下列設定：
  - IF-80：Gen.1
  - IF-80以外：Gen.2

按鈕	描述
關閉	關閉對話方塊。
套用	套用編輯內容。
恢復	將編輯內容恢復原狀。
測試	進行送料器的通訊確認。

按鈕	描述
設定	變更送料器的網路設定。  - IP位址 設定送料器的新IP位址。 - 子網路遮罩 設定送料器的新子網路遮罩。 - 連接埠 設定送料器的新連接埠編號。 在設定之後點擊[確定]按鈕。 選擇IF-A1520/2330時，若PC和IF-A1520/2330直接透過乙太網路電纜連接，將會顯示韌體更新[FW Update]按鈕。 

⚠ 注意

- 使用時，請在送料器中設定以下私人IP位址。

  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
- 為控制器設置全域IP位址時，請務必在使用前充分瞭解未經授權訪問等風險。

提示

控制器和送料器必須設定在相同的網路區段才能進行通訊。

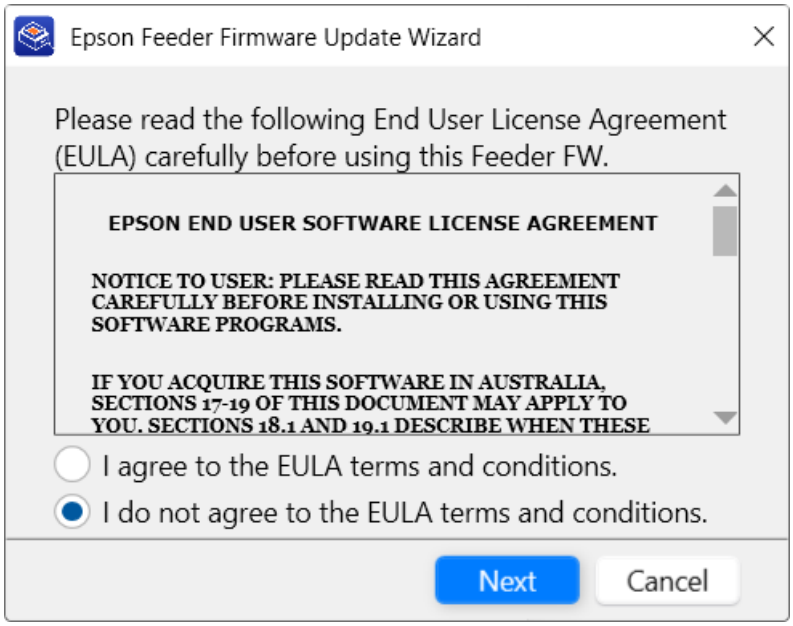
若設定與控制器不同的網路，將無法與送料器進行通訊。於此情況下，則需要變更控制器的IP位址或子網路遮罩。

該設定可在Epson RC+ 8.0功能表 - [設置] - [系統配置] - [控制器] - [設定]中進行。
詳細資訊請參閱以下手冊。
「Epson RC+ 8.0 使用指南 - 乙太網路通訊」

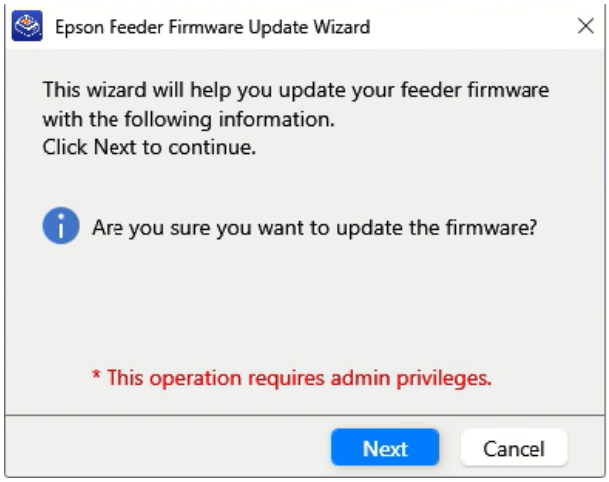
3.2.1.2 更新韌體

在IF-A1520/2330上，PC和送料器透過乙太網路連接至同一個網路時，[送料器的網路設定]畫面將會顯示[FW Update]按鈕。

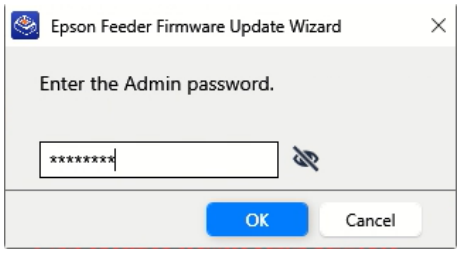
1. 將會顯示韌體的授權合約。
   請仔細閱讀內容，僅在接受時選擇[I agree to the EULA terms and conditions]，然後點擊[Next]。



2. 將會顯示更新精靈。要開始韌體更新，點擊[Next]。



3. 在密碼輸入畫面輸入密碼，然後點擊[OK]。



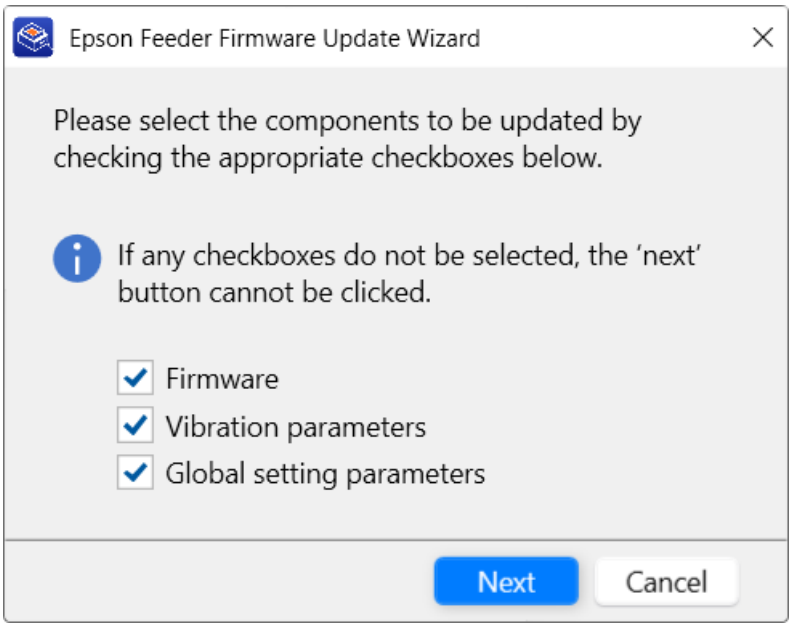
提示

- 初始密碼為「12345678」。
- 輸入了初始密碼時，將會顯示變更密碼對話方塊。請輸入兩次新密碼。

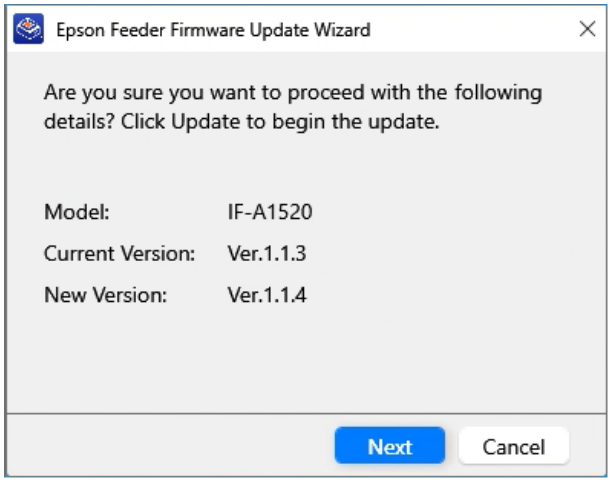
- 密碼指定為1至16個半形英數字元。無法設定與初始密碼相同的密碼。
- 請務必記錄變更後的密碼，以免遺忘。

3-2 選擇更新哪個項目，然後點擊[Next]。在預設狀態下選擇所有核取方塊。

- Firmware: 更新送料器的韌體（執行檔）時選擇
- Vibration parameters: 要將給料器中儲存的振動/序列參數重設為預設值時選擇。
- Global setting parameters: 要將各種內部參數重設為預設值時選擇。
  預設值可能會隨韌體更新而改變。

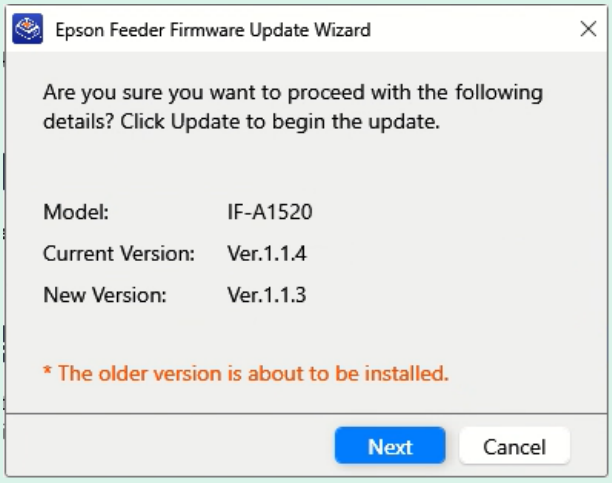


4. 將會顯示更新內容。若無問題，點擊[Next]。



提示

嘗試安裝早於已安裝韌體的版本時，將會顯示「* The older version is about to be installed.」警告訊息。

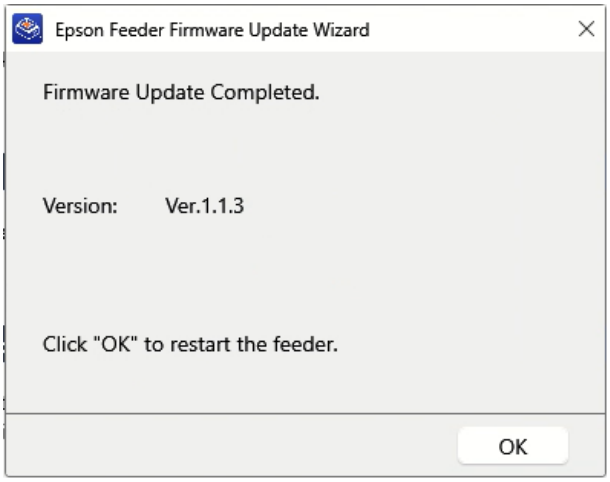


5. 寫入韌體。
   寫入韌體的工作約需10秒。

⚠ 注意

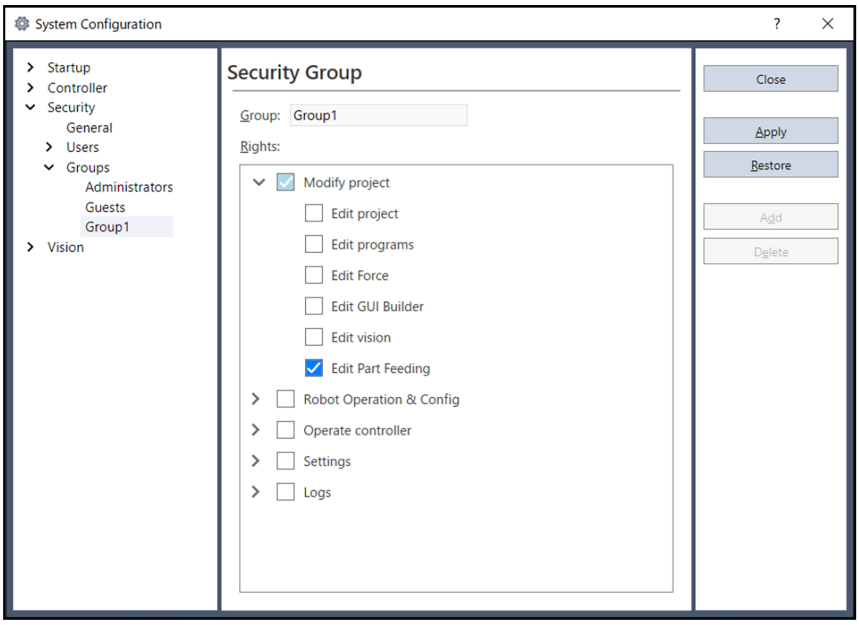
以下畫面顯示期間，都不要切斷送料器主體的電源。此外，如果斷開與PC的連接，可能會造成韌體寫入失敗，從而導致送料器無法啟動。

6. 將會顯示更新精靈的完成畫面。若點擊[OK]按鈕，送料器主體將會自動重新啟動。



3.2.1.3 安全畫面

透過安全設定，可限制能夠瀏覽或編輯Part Feeding選配件設定的使用者。

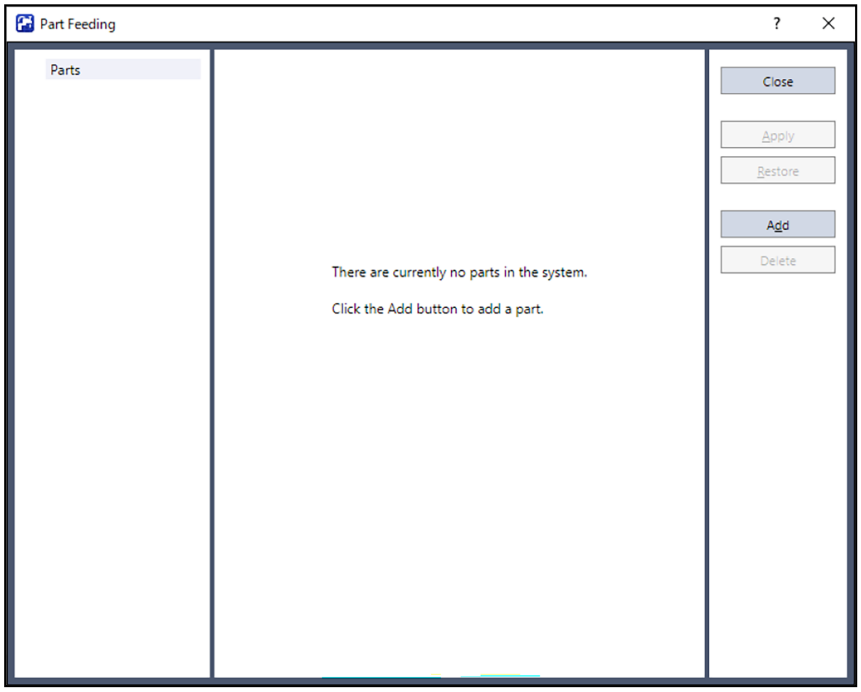


若有勾選[編輯Part Feeding]核取方塊，屬於此群組的使用者可顯示[料件送料]視窗。
若未勾選[編輯Part Feeding]核取方塊，屬於此群組的使用者嘗試顯示[料件送料]視窗時，將會出現錯誤。
詳細資訊請參閱以下手冊。
「Epson RC+使用指南 - 安全」

3.2.2 零件精靈

3.2.2.1 建立新零件

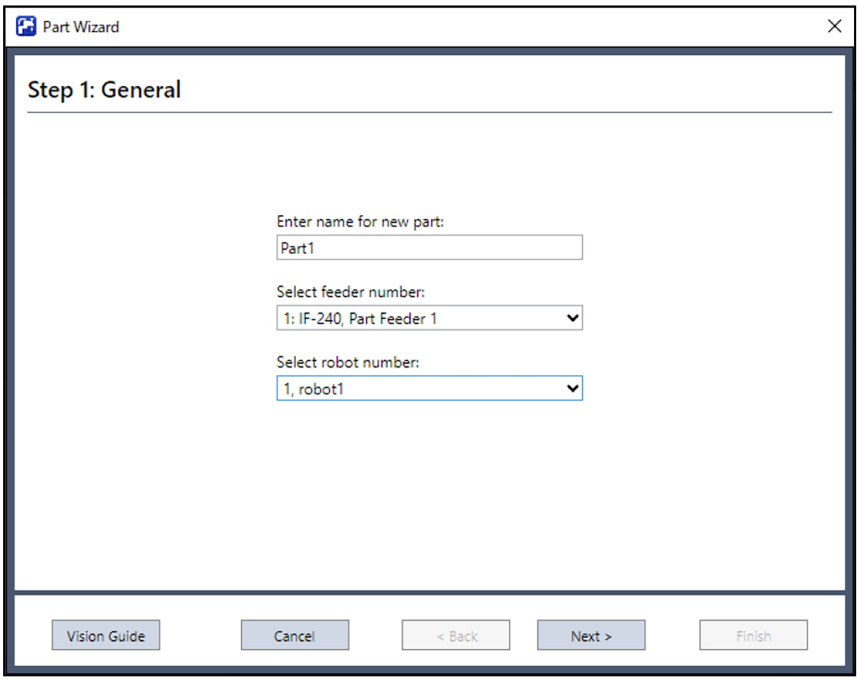
1. 點擊Epson RC+ 8.0功能表 - [工具] - [料件送料]。



2. 點擊[增加]按鈕。零件精靈將會啟動。

3.2.2.2 一般設定

進行基本設定。



項目	描述
新零件的名稱	記載零件名稱。(半形英數字和底線。最多32個字元)
選擇送料器編號	選擇此零件要使用的送料器編號。在系統配置畫面中確認送料器編號。

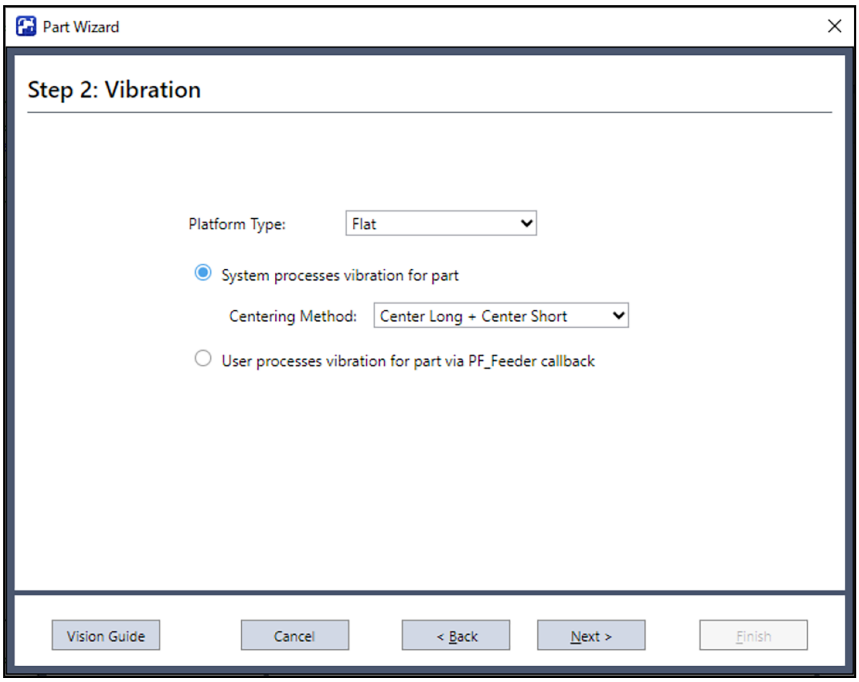
項目	描述
選擇機器人編號	選擇機器人編號。

下面說明零件精靈的基本操作。零件精靈需要透過視窗下方的5個按鈕操作。

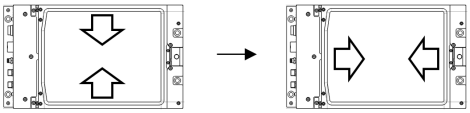
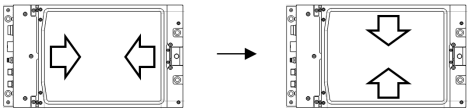
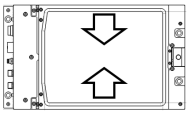
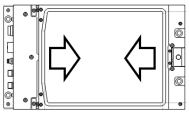
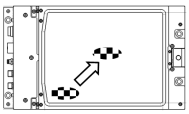
按鈕	說明
Vision Guide	顯示Vision Guide畫面。可建立視覺序列。
取消	中止零件精靈。
上一頁	返回前一頁零件精靈。
下一頁	切換到下一頁零件精靈。
完成	結束零件精靈。

3.2.2.3 振動

設定送料器的振動。



項目	描述
平台類型	指定平台的類型。 a) 可從本公司購買的標準平台。 平面：平坦的平台 防滾動：採用防滾動加工的平台 防黏著：採用防黏著加工的平台（僅限IF-80/240） b) 您自行製作的自訂平台。 溝槽：採用溝槽加工，藉此讓零件垂直立起的平台 孔：採用孔加工，藉此讓零件垂直立起的平台 凹槽：採用孔加工，藉此讓零件方向一致的平台

項目	描述
由系統進行振動處理	由系統控制送料器。 選擇上述b） 時，此選項則無法使用。
由PF_Feeder callback進行振動處理	使用PF_Feeder回呼函數。
集中方法	選擇[由系統進行振動處理]時，需要選擇零件的集中動作（在放入零件時等零件分布偏差較大的情況下，將零件集中到中央並均勻分散的動作）類型。 無： 不進行集中。 長軸集中+短軸集中： 先進行長軸方向的集中，再進行短軸方向的集中。  短軸集中+長軸集中： 先進行短軸方向的集中，再進行長軸方向的集中。  長軸集中： 僅進行長軸方向的集中。  短軸集中： 僅進行短軸方向的集中。  透過位移進行集中： 使用位移動作進行集中。 

提示

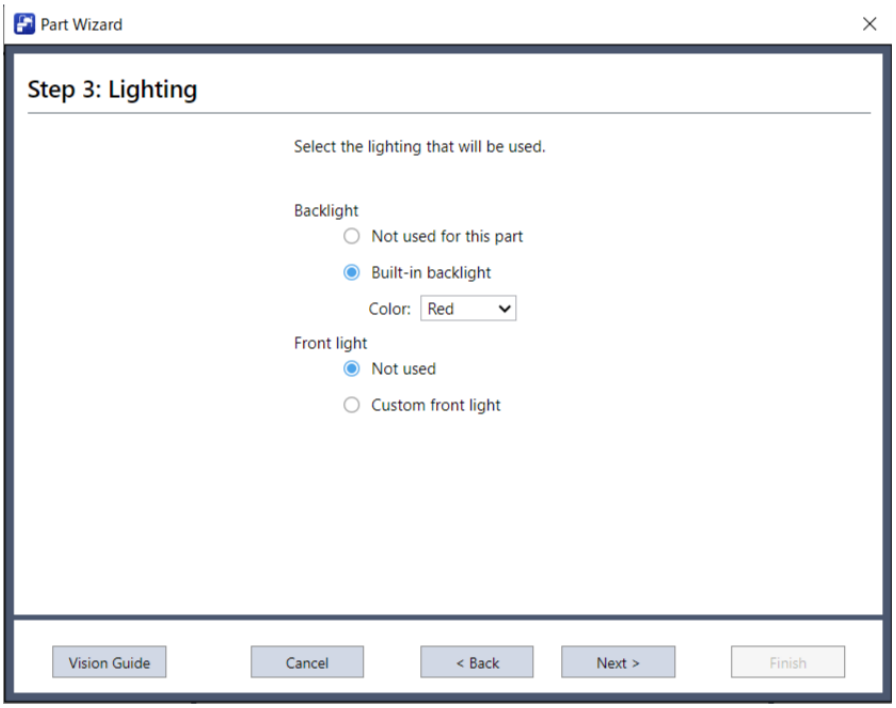
集中方法會依據零件類型和料斗位置而改變最佳選項。最有效的是以下兩種方法之一：

- 長軸集中+短軸集中
- 短軸集中+長軸集中

但與其他集中方法（或「無」時）相比，採取這些方法時的送料器運作時間較長。選擇能讓零件適當分散、且集中所需時間最短的方法最為有效。

3.2.2.4 照明

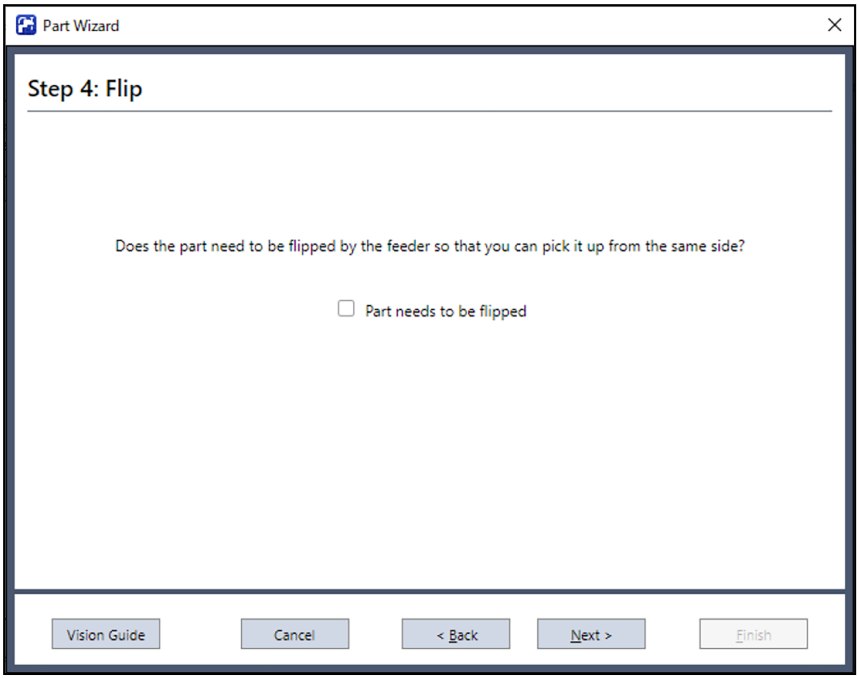
進行照明的設定。



項目		描述
背光		指定背光的控制方法。
	未使用	若未在視覺系統成像時使用送料器的背光，則選擇此項。屬於選項。
	內建背光	若要在視覺成像時使用送料器的背光，則選擇此項。沒有背光時則無法選擇。
	顏色	從「白」「紅」中選擇背光的顏色。 （僅限IF-A1520/2330）
前側光		指定前側光（選購品）的控制方法。
	不使用	不使用前側光。
	自訂前側光	呼叫PF_Control回呼函數，控制您所準備的自訂照明。屬於選項。 關於PF_Control的回呼函數，請參閱以下內容。 PF_Control

3.2.2.5 翻轉

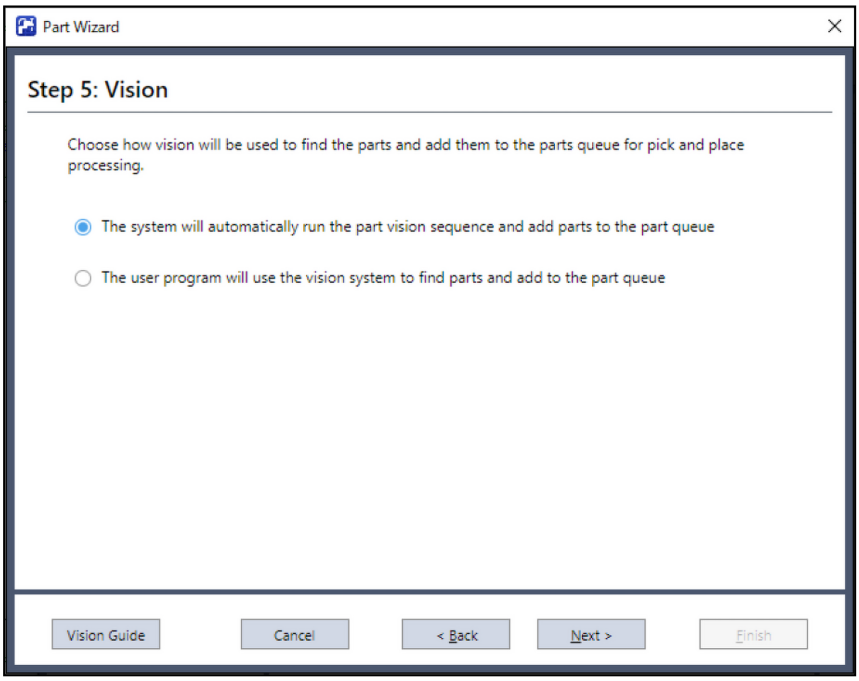
需要翻轉零件時，進行翻轉設定。若需要拾取朝向特定面的零件，則勾選「需要翻轉」。



項目	描述
需要翻轉	在零件有正反面等姿態時選擇。將啟用改變拾取姿態的翻轉動作。此操作可能會增加1次送料器運作過程中可取得的零件數量，從而提升週期時間的效率。

3.2.2.6 視覺系統

進行視覺系統的設定。

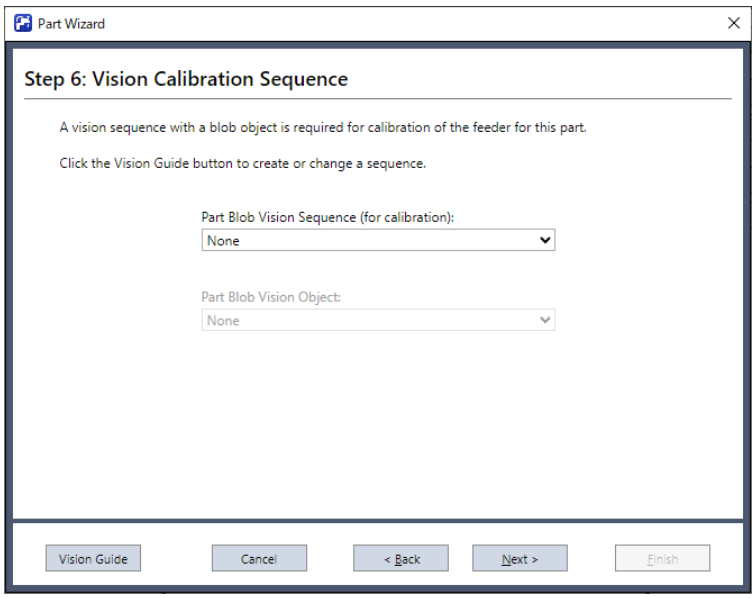


項目	描述
系統會自動執行零件視覺序列，並將零件新增到零件佇列。	由系統自動進行視覺處理。通常會選擇此項。

項目	描述
使用者需要在PF_Vision Callback中編寫必要的處理程序（操作照明、執行視覺系統以檢測零件、新增到零件佇列）。系統將會自動執行PF_Vision Callback。	屬於選項。 啟用PF_Vision回呼函數，並自訂視覺系統的操作。 關於PF_Vision回呼函數，請參閱以下內容。 PF_Vision

3.2.2.7 視覺校準序列

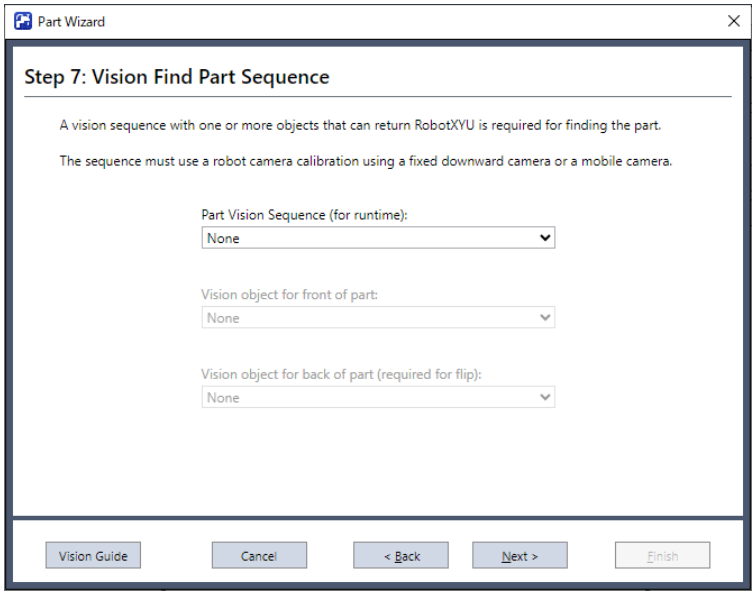
對用於送料器校準的視覺序列進行設定。



項目	描述
零件Blob視覺序列	必須設定：請務必設定此項目。 選擇用於送料器校準的視覺序列名稱。
零件Blob視覺物件	必須設定：請務必設定此項目。 選擇視覺物件名稱，其用於在送料器校準中，使用送料器背光來檢測零件。僅可指定Blob。

3.2.2.8 零件檢測視覺序列

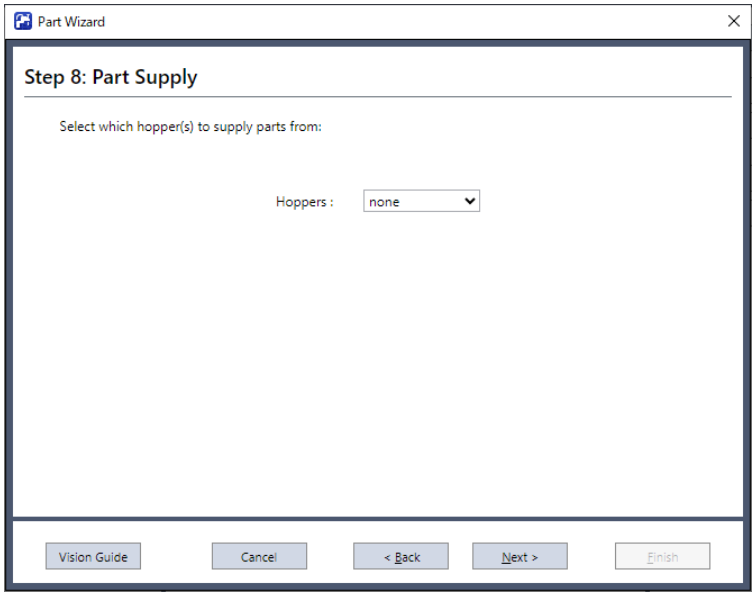
對用於零件檢測的視覺系統進行設定。



項目	描述
零件視覺序列（for runtime）	必須設定：請務必設定此項目。 選擇用於檢測零件的視覺序列名稱。 僅顯示已執行機器人校準的視覺序列。
用於檢測正面零件的視覺物件	必須設定：請務必設定此項目。 選擇用於檢測所拾取零件的視覺物件名稱。 僅顯示回傳RobotXYU結果的物件。
用於檢測背面零件的視覺物件	要進行翻轉時（參考「一般設定」），請務必設定此項目。 選擇視覺物件名稱，其用於檢測背面等姿態上無法拾取之零件。 僅顯示回傳RobotXYU結果的物件。

3.2.2.9 供應零件

進行料斗的設定。



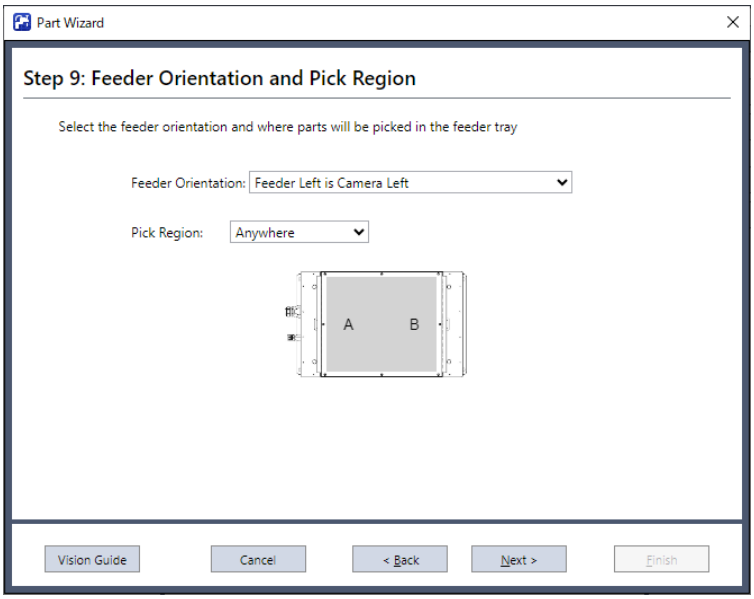
項目	描述
料斗	選擇使用的料斗。

提示

未勾選「啟用料斗」核取方塊的狀態下若選擇要使用的料斗，將會顯示未連接料斗的通知訊息。

3.2.2.10 送料器的方向和拾取區

進行送料器配置和零件拾取相關的設定。

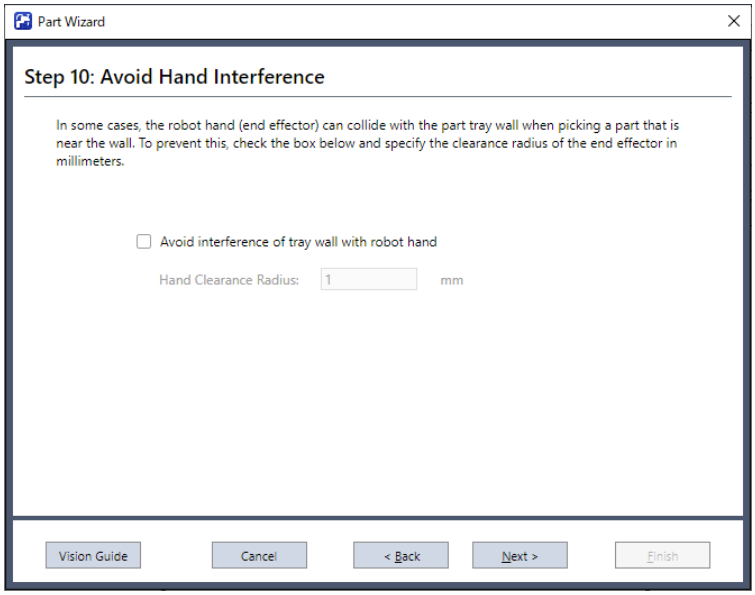


項目	描述
送料器的方向	設定透過攝影機畫面看到的送料器設置方向。

項目	描述
拾取區	設定透過攝影機畫面看到的零件拾取區域。 選擇全體或A、B、C、D中的任一區域。 可設定的區域可能會依送料器機型而不同。 IF-80：全體 IF-240、IF-A1520、IF-A2330：全體、區域A、區域B、區域C、區域D IF-380、IF-530：全體、區域A、區域B

3.2.2.11 防止末端夾具的干擾

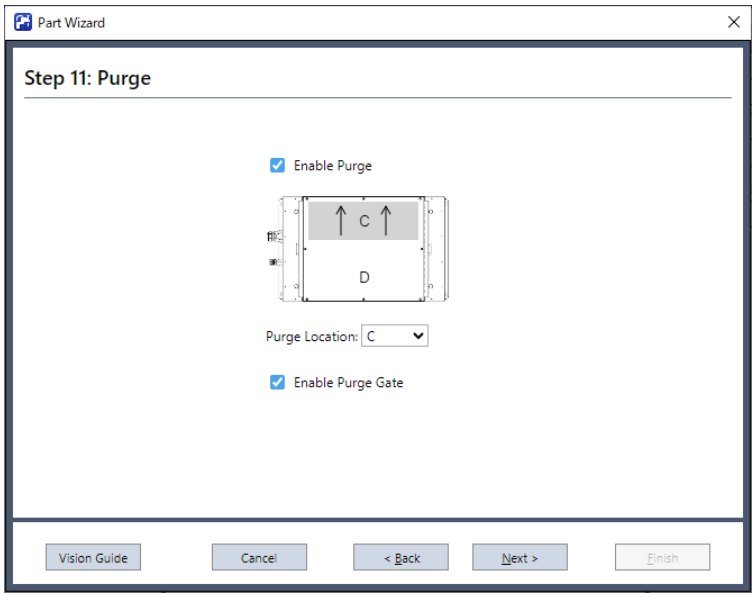
進行末端夾具與平台的干擾迴避功能相關設定。



項目	描述
防止末端夾具的碰撞	啟用末端夾具與平台的碰撞防止功能時，勾選此核取方塊。
末端夾具的旋轉半徑	位於平台外周（透過視覺系統的搜尋視窗進行設定）距離此值內的零件不會進入座標佇列。 單位：mm

3.2.2.12 清除

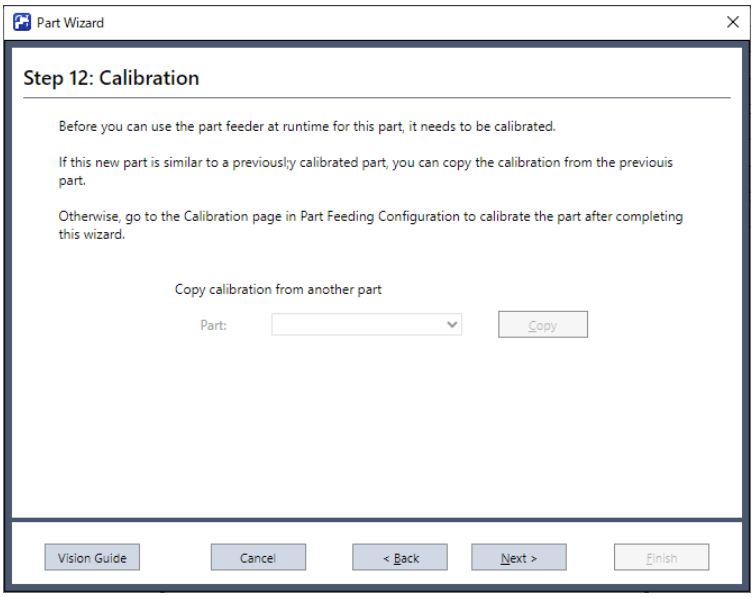
進行清除的相關設定。



項目	描述
啟用清除	啟用清除功能時勾選此核取方塊。
啟用清除閘門	使用清除閘門時勾選此核取方塊。
清除位置	設定清除的方向。在執行清除動作時，零件會朝箭頭的方向移動（位移）。清除位置可選擇A、C或D中的任一位置。 可設定的項目依送料器類型而不同。 IF-80: A IF-240、IF-380、IF-530、IF-A1520、IF-A2330: C、D

3.2.2.13 送料器校準

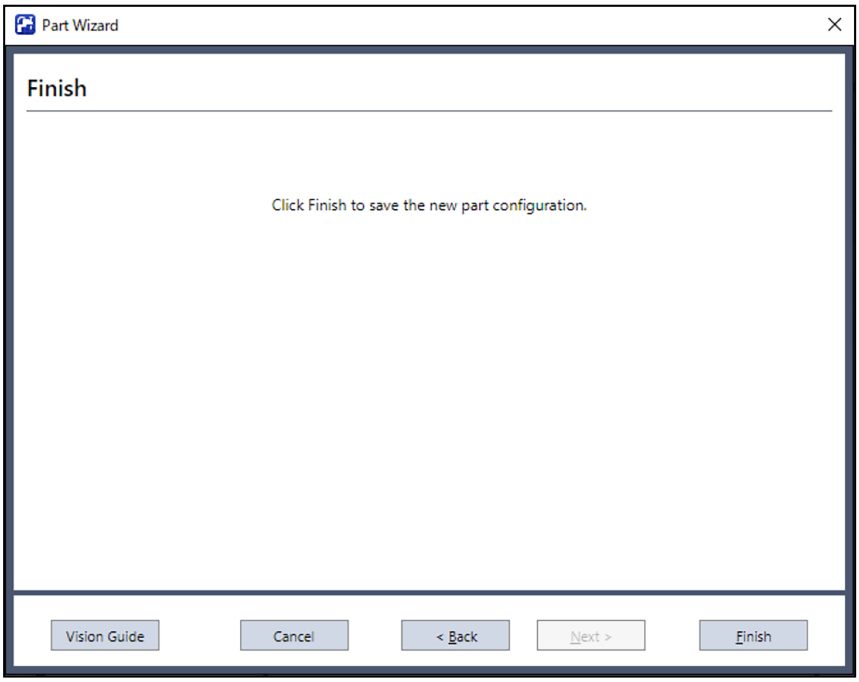
進行與校準相關的設定。



項目		描述
從其他零件複製校準結果		將校準結果從另一個零件複製到此零件。
	零件	指定被複製的零件。
	複製	執行複製。

3.2.2.14 完成

點擊[完成]按鈕以結束零件精靈。



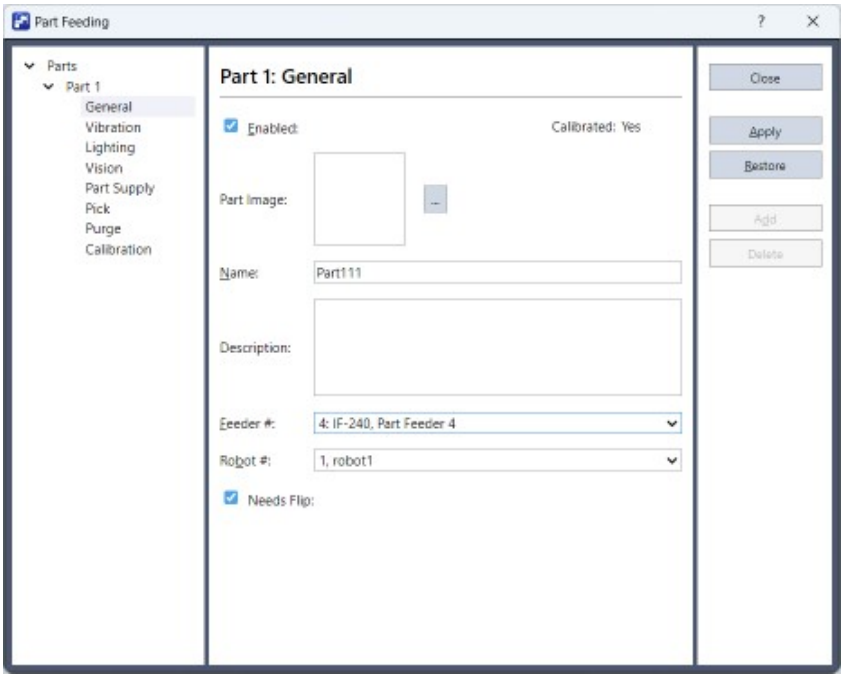
3.2.3 料件送料對話方塊

在Epson RC+ 8.0功能表 - [工具] - [料件送料]中，可進行Part Feeding的各種設定、送料器校準、調整及測試。

> 提示
>
> 請將Epson RC+連接到控制器。料件送料視窗只有在Epson RC+連接到控制器的狀態下才會顯示。若在使用虛擬控制器或離線狀態下嘗試開啟Part Feeding視窗，則會出現錯誤。需要啟用Part Feeding授權。

3.2.3.1 一般設定

進行基本設定。



項目	描述
啟用	若要啟用此零件，則勾選核取方塊。若指定未啟用的零件並執行PF_Start命令，將會出現錯誤。
零件影像	可註冊零件的影像。 點擊[…]按鈕，選擇零件的影像檔案。 此處註冊的影像不會被用於影像處理。可註冊容易識別的影像。
校準	需要：需要執行送料器校準。 完成：送料器校準已完成。
名稱	記載零件名稱。（半形英數字和底線。最多16個字元）
註解	填寫零件的說明（註解）。 屬於選項。 （最多256個字元）
送料器#	選擇此零件要使用的送料器編號。 在系統配置畫面中確認送料器編號。
機器人	選擇機器人編號。
需要翻轉	在零件有正反面等姿態時選擇。將啟用改變拾取姿態的翻轉動作。此操作可能會增加1次送料器運作過程中可取得的零件數量，從而提升週期時間的效率。

按鈕	描述
關閉	關閉畫面。
套用	套用編輯內容。
恢復	將編輯內容恢復原狀。

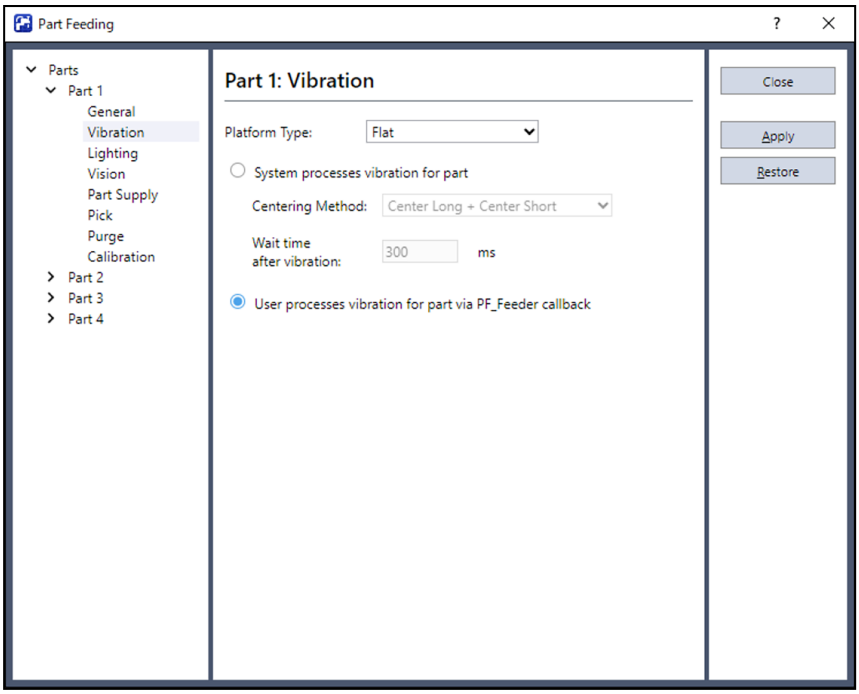
按鈕	描述
增加	將零件新增到樹狀清單。最多可新增32個零件。
刪除	從樹狀清單刪除零件。僅可刪除樹狀清單末端的零件。 無法刪除樹狀清單中間的零件。取消勾選[啟用]核取方塊來停用。

提示

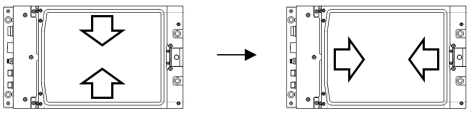
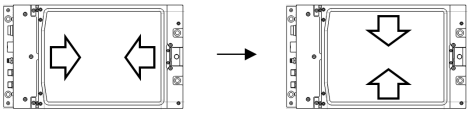
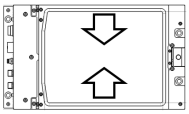
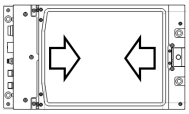
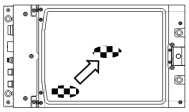
若刪除所有零件後進行建置，則會發生建置錯誤。此錯誤是由於無法使用料件送料的命令和函數所導致。於此情況下，請將狀態視窗中顯示的相關行註解掉。

3.2.3.2 振動

進行振動設定。



項目	描述
平台類型	指定平台的類型。 a) 可從本公司購買的標準平台。 平面：平坦的平台 防滾動：採用防滾動加工的平台 防黏著：採用防黏著加工的平台（僅限IF-80/240） b) 您自行製作的自訂平台。 溝槽：採用溝槽加工，藉此讓零件垂直立起的平台 孔：採用孔加工，藉此讓零件垂直立起的平台 凹槽：採用孔加工，藉此讓零件方向一致的平台
由系統進行振動處理	由系統控制送料器。 若已選擇上述b)，則無法選擇此項。

項目	描述
由PF_Feeder callback進行振動處理	使用PF_Feeder回呼函數。
集中方法	選擇[由系統進行振動處理]時，需要選擇零件的集中動作（在放入零件時等零件分布偏差較大的情況下，將零件集中到中央並均勻分散的動作）類型。 無： 不進行集中。 長軸集中+短軸集中： 先進行長軸方向的集中，再進行短軸方向的集中。  短軸集中+長軸集中： 先進行短軸方向的集中，再進行長軸方向的集中。  長軸集中： 僅進行長軸方向的集中。  短軸集中： 僅進行短軸方向的集中。  透過位移進行集中： 使用位移動作進行集中。 
振動後等待時間	指定從送料器振動停止後到視覺系統拍攝之前的等待時間（單位：毫秒）。若視覺系統無法識別零件，或是機器人會在抓住送料器上的零件時發生位置偏移，增加此數值可能有助於改善這類狀況。

提示

集中方法會依據零件類型和料斗位置而改變最佳選項。最有效的是以下兩種方法之一：

- 長軸集中+短軸集中
- 短軸集中+長軸集中

但與其他集中方法（或「無」時）相比，採取這些方法時的送料器運作時間較長。選擇能讓零件適當分散、且集中所需時間最短的方法最為有效。

提示

選擇「使用位移的集中」時，自動校準則無法正常運作。

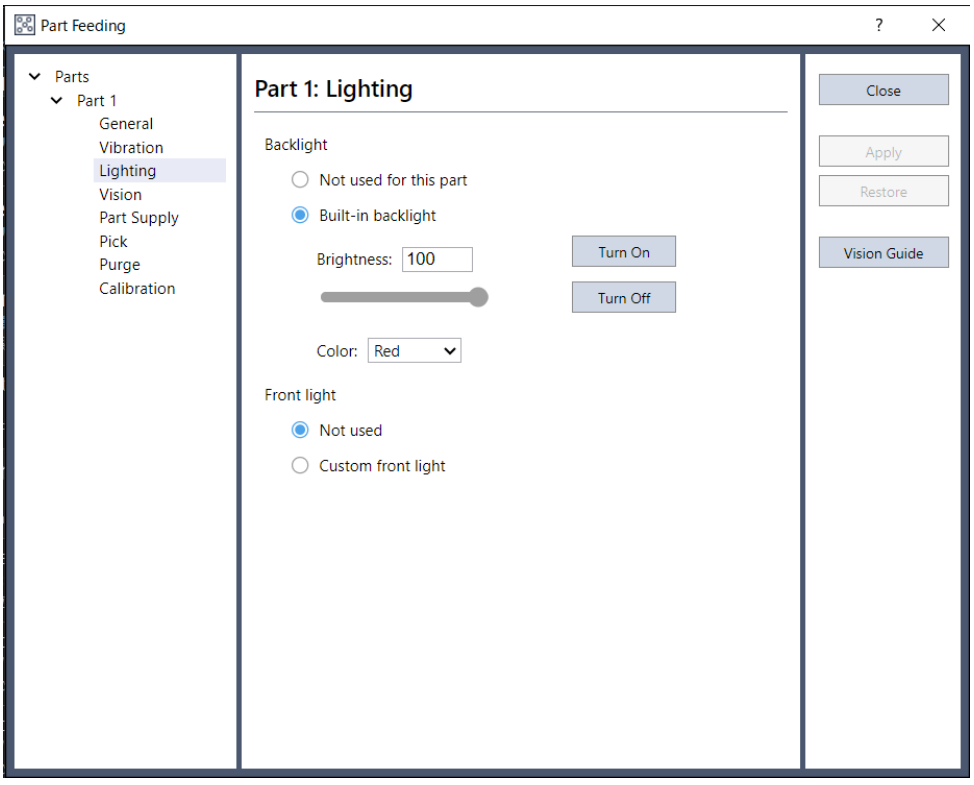
請參閱以下內容。

校準&測試

按鈕	描述
關閉	關閉畫面。
套用	套用編輯內容。
恢復	將編輯內容恢復原狀。

3.2.3.3 照明

進行照明的設定。



項目		描述
背光		指定背光的控制方法。
	未使用	若未在視覺系統成像時使用送料器的背光，則選擇此項。 屬於選項。
	內建背光	若要在視覺成像時使用送料器的背光，則選擇此項。 沒有背光時則無法選擇。
On		開啟送料器的背光。

項目		描述
Off		關閉送料器的背光。
照度		設定送料器背光的亮度。 指定0 ～ 100%的數值。
顏色		從「白」「紅」中選擇背光的顏色。 （僅限IF-A1520/2330）
前側光		指定前側光（選購品）的控制方法。
	不使用	不使用前側光。
	自訂前側光	呼叫PF_Control回呼函數，控制您所準備的自訂照明。屬於選項。 關於PF_Control的回呼函數，請參閱以下內容。 PF_Control

提示

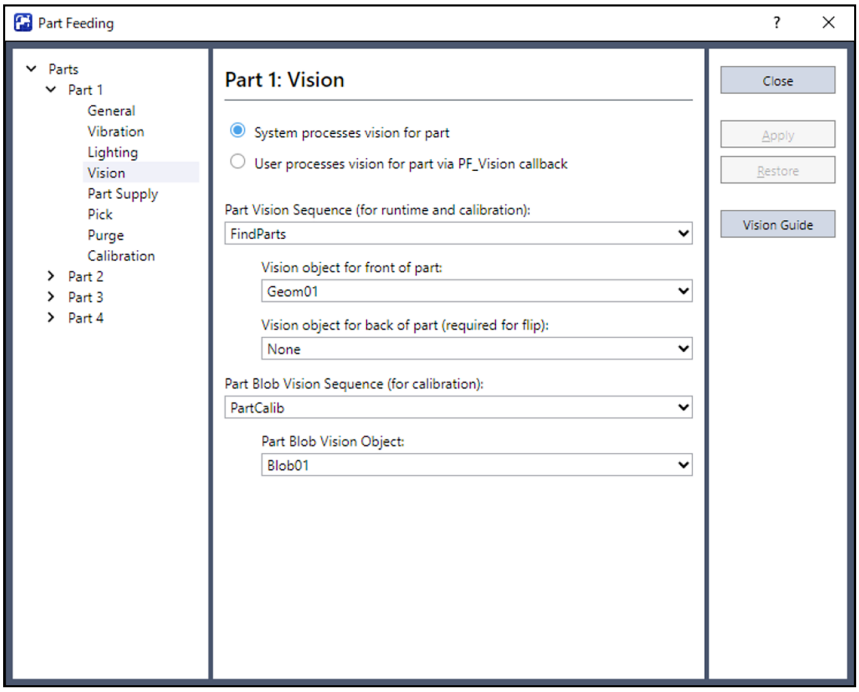
照度可在0～100%的範圍內以百分比設定，但實際照度的最小值可能會因送料器的型號而受限。請參閱以下內容。

PF_BacklightBrightness

按鈕	描述
關閉	關閉畫面。
套用	套用編輯內容。
恢復	將編輯內容恢復原狀。
Vision Guide	顯示Vision Guide畫面。 可建立視覺序列。

3.2.3.4 視覺系統

進行視覺系統的設定。
關於視覺序列的建立方法，請參閱以下內容。
Part Feeding選配件所使用的視覺序列



項目	描述
由系統進行視覺處理	由系統自動進行視覺處理。 通常會選擇此項。
使用PF_Vision callback進行視覺處理	屬於選項。 啟用PF_Vision回呼函數，並自訂視覺系統的操作。關於PF_Vision回呼函數，請參閱以下內容。 PF_Vision
零件檢測視覺序列	必須設定：請務必設定此項目。 選擇用於檢測零件的視覺序列名稱。僅顯示已執行機器人校準的序列。
用於檢測正面零件的視覺物件	必須設定：請務必設定此項目。 選擇用於檢測所拾取零件的視覺物件名稱。僅顯示能回傳RobotXYU結果的物件。
用於檢測背面零件的視覺物件	要進行翻轉時（參考「一般設定」），請務必設定此項目。選擇視覺物件名稱，其用於檢測背面等姿態上無法拾取之零件。針對表面和背面都拾取時也需進行設定。僅顯示能回傳RobotXYU結果的物件。
零件Blob視覺序列	必須設定：請務必設定此項目。 選擇用於送料器校準的視覺序列名稱。僅顯示已執行機器人校準的序列。
零件Blob視覺物件	必須設定：請務必設定此項目。 選擇視覺物件名稱，其用於在送料器校準中，使用送料器背光來檢測零件。僅可指定Blob。

提示

請務必針對標示為必須設定的項目進行指定。

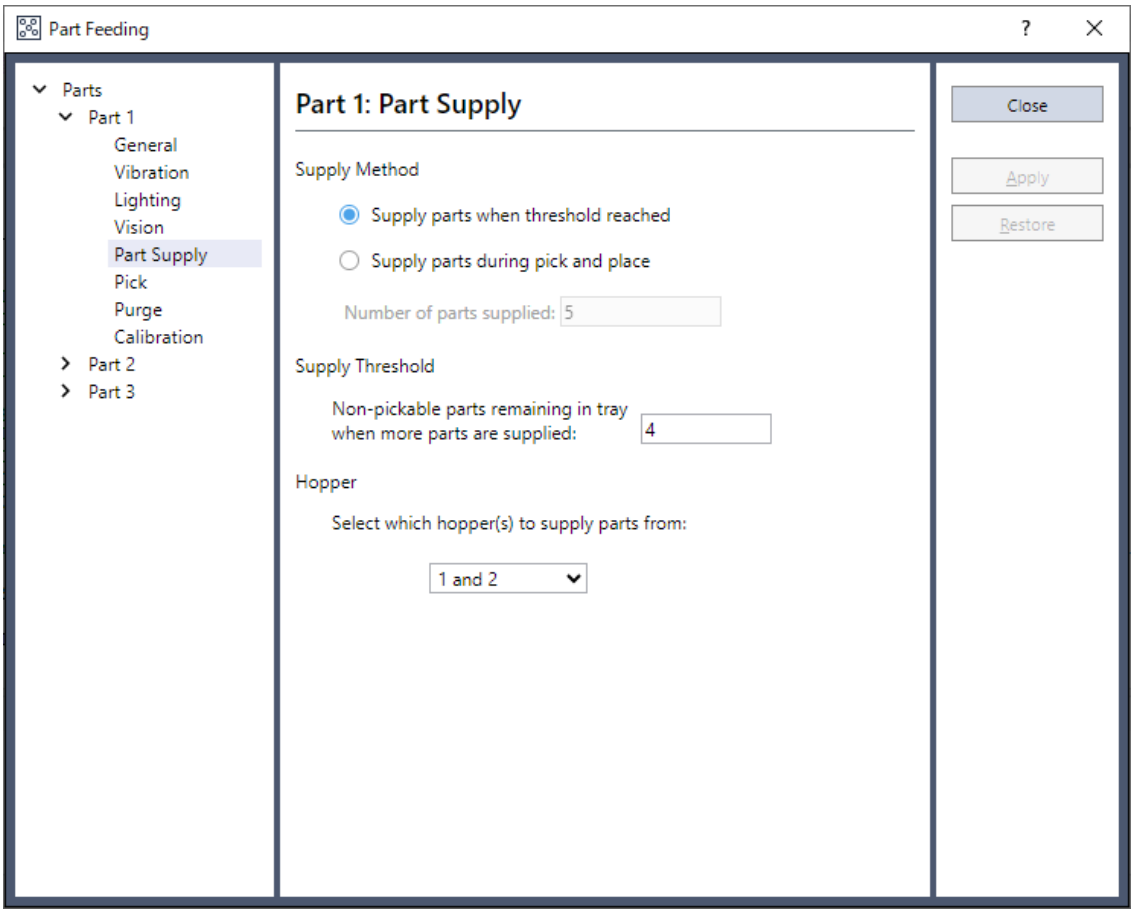
若未設定，校準動作或程序運作時將會發生錯誤。

按鈕	描述
關閉	關閉對話方塊。
套用	套用編輯內容。 除了選項以外的項目中如有未指定的項目，則無法操作。
恢復	將編輯內容恢復原狀。
Vision Guide	顯示Vision Guide對話方塊。 可建立視覺序列。

3.2.3.5 供應零件

設定向送料器供應零件的方式。料斗的供料動作由您在PF_Control回呼函數內進行程式設計。關於PF_Control的回呼函數，請參閱以下內容。

PF_Control



項目	說明
在達到供應閾值時供應零件	在送料器上的零件數量達到閾值時供應零件。

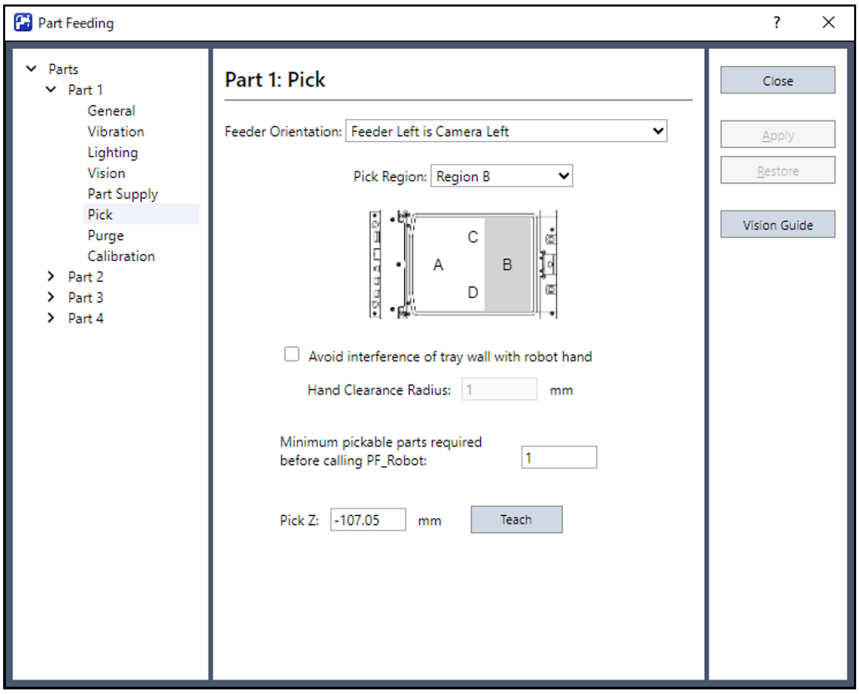
項目	說明
在取放期間供應零件	添加零件，讓送料器上的零件數量達到最佳數量。 與拾取完零件後再供應時相比，機器人的週期時間較短。
供應零件數量	在選擇「在取放期間供應零件」時，輸入要透過料斗添加的零件數量。 預設值為10。 關於零件數量，請參閱下列內容。 **料斗的調整方法**
判斷為需要供應的送料器上的剩餘零件數量	當托盤上剩餘的不可拾取零件數量低於此值時，將會供應零件。 預設值為4。 使用0時，則會在零件用盡後再供應零件。
料斗	選擇使用的料斗。

提示

- 未勾選「啟用料斗」核取方塊的狀態下若選擇要使用的料斗，將會顯示未連接料斗的通知訊息。
- 料斗的測試可透過以下畫面執行：
  校準&測試

3.2.3.6 拾取

進行送料器配置和零件拾取相關的設定。



項目	描述
送料器的方向	設定透過攝影機畫面看到的送料器設置方向。

拾取區	設定透過攝影機畫面看到的零件拾取區域。 選擇全體或A、B、C、D中的任一區域。 若選擇區域A、B、C、D，在校準時也會進行位移。 IF-80：全體 IF-240、IF-A1520、IF-A2330：全體、區域A、區域B、區域C、區域D IF-380、IF-530：全體、區域A、區域B
防止末端夾具的碰撞	啟用末端夾具與平台的碰撞防止功能時，勾選此核取方塊。
末端夾具的旋轉半徑	位於平台外周（透過視覺系統的搜尋視窗進行設定）距離此值內的零件不會進入座標佇列。單位：mm
呼叫PF_Robot前所需的可拾取零件的最小數量	一般狀況下，即使只搜尋到1個可拾取零件，也會呼叫PF_Robot callback。預設值為1。一般來說，送料器的振動需要花費時間，且振動後不一定能確實搜尋到可選擇的零件，因此無需變更設定。無論此設定如何，零件佇列中都會載入視覺系統檢測到的實際零件數量。依據零件的數量和分布決定振動方法。若未滿足「呼叫PF_Robot前所需的可拾取零件的最小數量」，系統將會執行適當的動作。在呼叫PF_Robot前，若送料器上有最小數量的可拾取零件，在某些情況下可能會提高系統效能。
示教	顯示教導畫面，教導拾取零件時的Z座標。 使用6軸型機器人時，除了Z座標外，也需要教導V座標、W座標。 請參閱以下內容。 教導視窗
Z位置	拾取零件時的Z座標（局部座標）。輸入教導時的Z座標。可輸入數值進行變更。 單位：mm

提示

勾選[防止末端夾具的碰撞]核取方塊並調整[末端夾具的半徑]時，請確認夾具與平台不會產生干擾。

提示

在視覺畫面選擇「使用PF_Vision callback的系統視覺」時，即使啟用[防止末端夾具的碰撞]，位於平台邊緣距離[末端夾具的半徑]內的零件也可能進入到零件座標佇列。應小心。

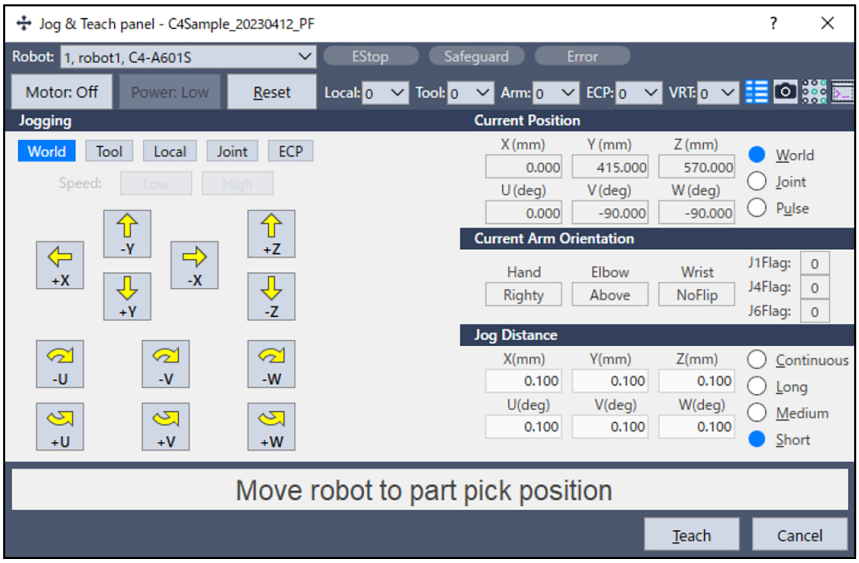
TIP

使用6軸型機器人時的V座標、W座標之值，雖然不會顯示在對話方塊中，但會被儲存在內部中。

按鈕	描述
關閉	關閉畫面。
套用	套用編輯內容。
恢復	將編輯內容恢復原狀。

3.2.3.7 教導視窗

教導拾取零件時的Z座標。



1. 在平台上放置1個零件。
2. 操作Jog按鈕（+X、-X、+Y、-Y），將機器人移動到取得零件時的Z座標和姿態。
3. 點擊[確定]按鈕。Z座標、V姿態、W姿態將會被註冊。

提示

此處選擇的機器人是零件檢測視覺序列中設定的視覺校準所參照的機器人。

提示

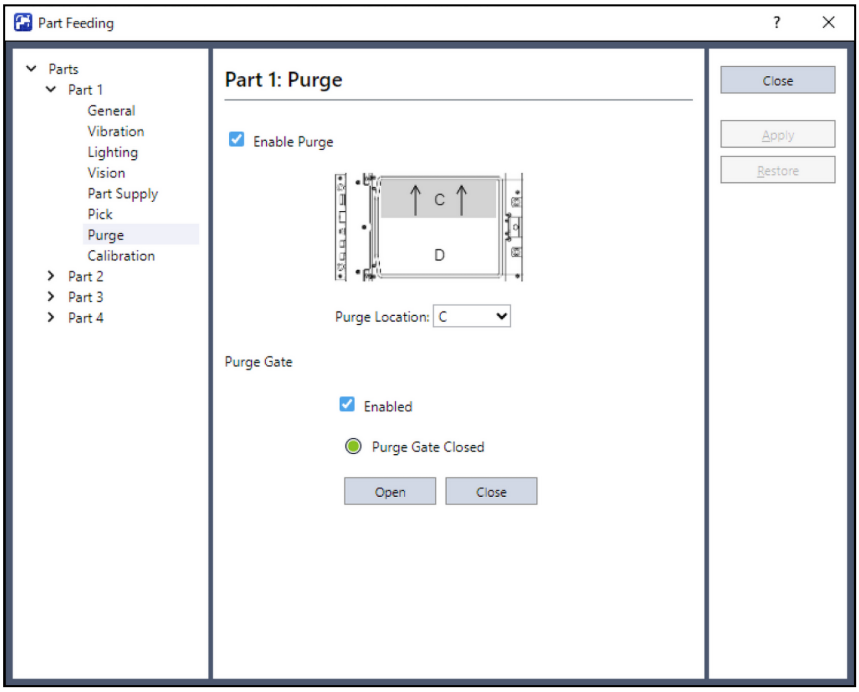
在視覺畫面選擇「使用PF_Vision callback的系統視覺」時，Z座標的處理內容需由您編寫在PF_Vision回呼函數內。

如需詳細資訊，請參閱以下內容。

PF_Vision

3.2.3.8 清除

進行清除的設定。



項目		描述
啟用清除		啟用清除功能時勾選此核取方塊。
清除位置		設定清除的位置。進行清除動作時，零件會朝箭頭的方向移動（位移）。清除位置可選擇A、C或D中的任一位置。可設定的項目依送料器類型而不同。 IF-80: A IF-240、IF-380、IF-530、IF-A1520、IF-A2330: C、D
清除閘門		進行清除閘門的設定。
	啟用	若要啟用零件的清除閘門動作，則勾選此核取方塊。在預設狀況下，若在Epson RC+ 8.0功能表 - [設置] - [系統配置] - [控制器] - [零件送料器]中有勾選[安裝清除閘門]核取方塊，則啟用本核取方塊。
	關閉清除閘門	顯示清除閘門感測器的狀態。 關閉：綠色 未關閉：灰色
	開啟/關閉	開啟或關閉清除閘門。

提示

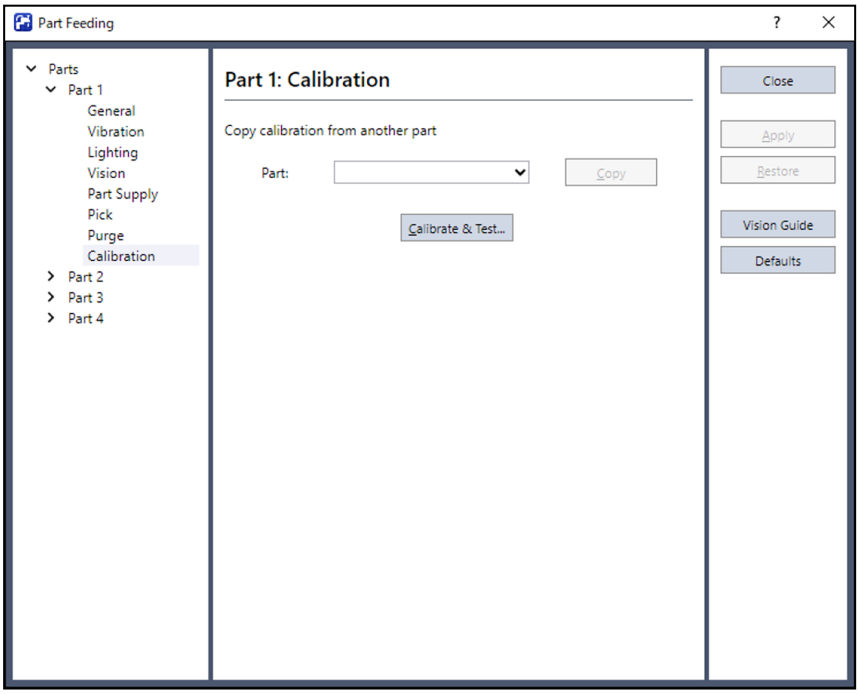
只有在Epson RC+ 8.0功能表 - [設置] - [系統配置] - [控制器] - [零件送料器]中有勾選[安裝清除閘門]核取方塊時，才會顯示[清除閘門]群組方塊。

提示

在未關閉清除閘門時，若要切換到其他畫面，則會顯示「清除閘門將會關閉。是否繼續？」的對話方塊。按下「確定」之後，將會關閉清除閘門。

3.2.3.9 校準

進行送料器的校準、參數編輯與測試。



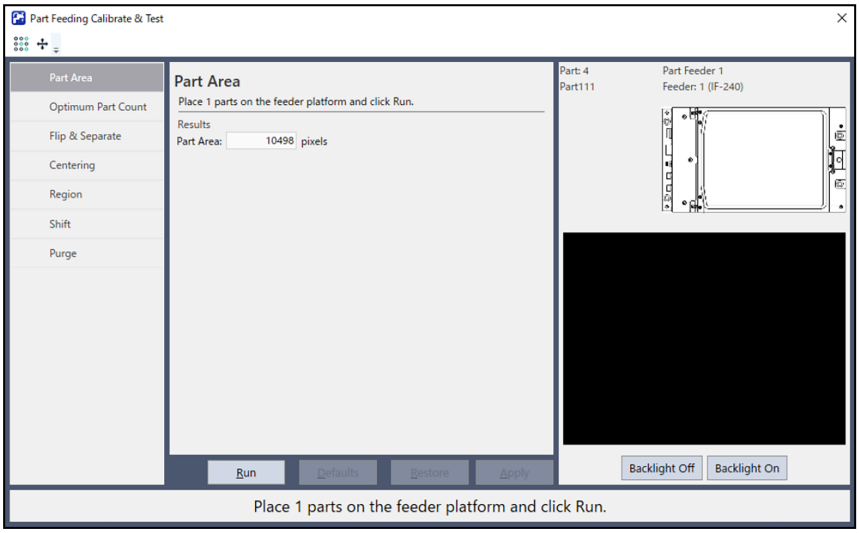
項目		說明
從其他零件複製校準結果		將校準結果從另一個零件複製到此零件。
	零件	指定被複製的零件。
	複製	執行複製。
校準&測試		進行送料器的校準、參數編輯與測試。

3.2.4 校準&測試

執行校準&測試後，可執行以下作業：

- 送料器的校準
- 送料器參數的調整
- 送料器的運作測試

首先，從左側標籤選擇要調整的送料器運作。



各標籤的說明

標籤	說明	必須 / 任意執行校準
零件區域	執行零件區域的校準。	必須
最佳放入零件數量	調整最佳放入零件數量。	任意
分離	進行分離（讓零件分散的動作）的調整和測試。	任意
翻轉&分離	進行翻轉（改變零件姿態的動作）和分離（讓零件分散的動作）的調整和測試。 在啟用翻轉時（參考「一般設定」）顯示。	任意
集中	進行集中（聚集零件的動作）的調整和測試。	任意
區域	進行拾取區域（指定零件區進行拾取時的零件移動動作）的調整和測試。 在拾取畫面（參考「拾取」）中選擇A、B、C、D中的任一區域作為零件區時顯示。	任意
位移	進行位移（零件的移動動作）的調整和測試。	任意
料斗	進行料斗的調整和測試。在使用IF-80或已勾選「連接料斗」核取方塊時顯示。	（無）
清除	進行清除（從送料器排出零件的動作）的調整（僅限IF-80）和測試。 在清除畫面（參考「清除」）中啟用清除功能時顯示。	任意

提示

進行新註冊時，除了[零件區域]標籤和[料斗]標籤以外，無法選擇其他標籤。

請先選擇[零件區域]標籤以執行校準。

選擇了「拾取區域」時：請執行「區域」的自動校準。

使用IF-80且將清除設為「啟用」時：請執行「清除」的自動校準。

不執行自動校準時，將會使用預設值。

按鈕的功能

按鈕	說明
關閉	關閉視窗。
執行/暫停	開始目前畫面的校準或測試。在執行時，顯示會變更為「暫停」。點擊[暫停]按鈕，將會暫停目前執行的動作。
套用	保存改變。
恢復	將變更的值恢復原狀。
預設	將選擇中的畫面之值恢復為預設值。
背光 開啟 / 關閉	亮起或關閉送料器的背光。
 （I/O監視器）	啟動I/O監視器。 用於控制自訂照明等。
 （點動動作）	顯示用於點動機器人的視窗。 在將移動式攝影機移動到影像擷取位置時使用。

提示

若在使用安全門的設備上點擊[執行]按鈕，並在運作中開啟安全門，則會停止送料器的運作。精靈畫面不會產生變化。

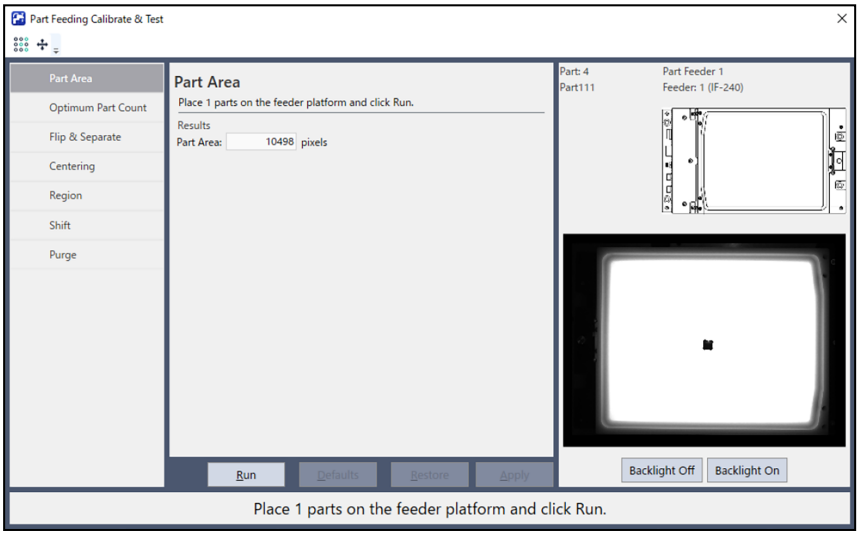
此時，請點擊[暫停]按鈕來暫停校準，關閉安全門後重新開始校準。

準備

- 請準備送料器校準的目標零件。
  需要一定數量的零件。關於數量，請參閱以下內容。
  最佳放入零件數量
- 使用移動式攝影機時：
  點擊[點動動作]按鈕，將機器人移動到影像擷取位置。
- 使用IO控制的自訂照明時：
  點擊精靈中顯示的[I/O監視器]按鈕，讓照明亮起。
- 新註冊時：
  請先執行「零件區域」校準。
  選擇[零件區域]標籤，並點擊[執行]按鈕進行校準。

3.2.4.1 零件區域

進行零件區域（每1個零件的像素數量）的校準。



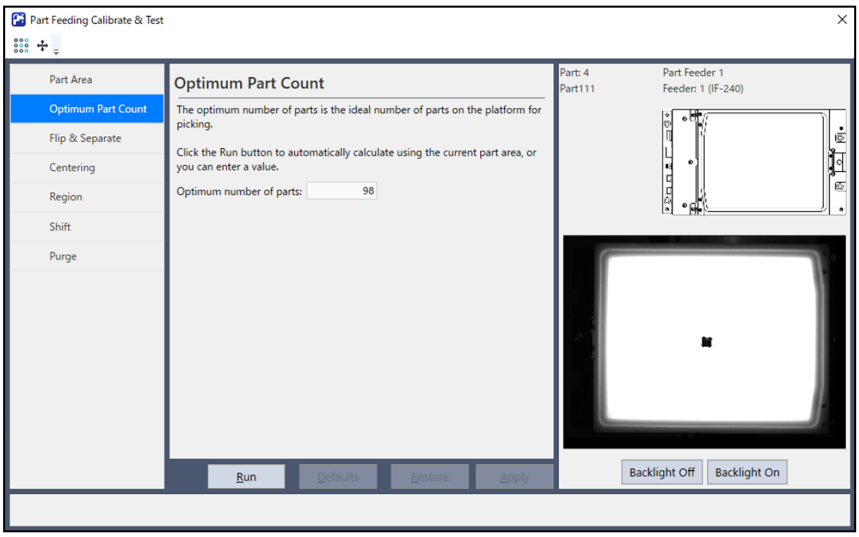
1. 顯示項目的說明

項目	說明
零件面積	零件的像素數量

2. 校準步驟
   (1) 在平台上放置1個零件。
   (2) 點擊[執行]按鈕。將會測量零件區域的像素數量。
   (3) 點擊[套用]按鈕。將會儲存結果。

3.2.4.2 最佳放入零件數量

計算最佳放入零件數量。



顯示項目的說明

項目	說明
最佳放入零件數量	最佳放入零件數量的計算值 可編輯

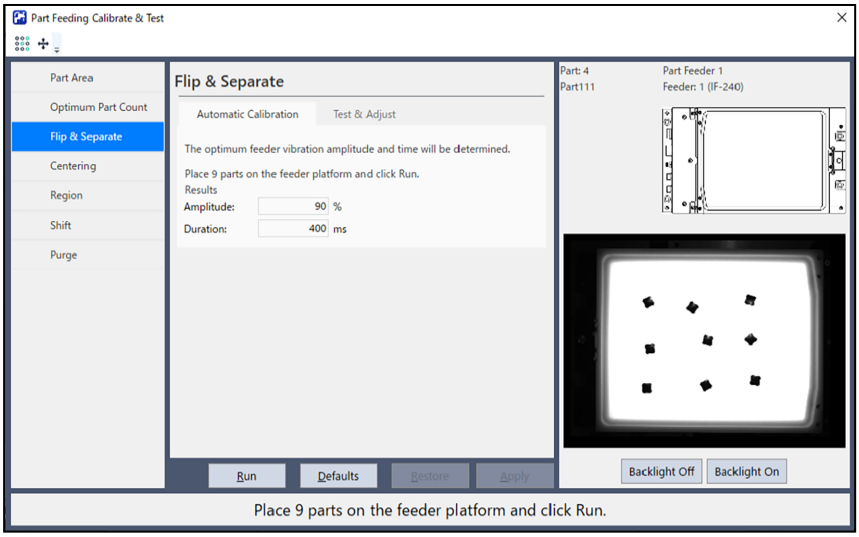
調整指南
本參數是基於零件為正方形或接近圓形的假設所計算而出的。因此，若零件形狀細長或中央有空洞部分，則應設定為小於校準參數的值。
若要與料斗一起使用，則根據該數值判斷是否需要添加零件。若感覺放入的零件數量太少，則設定為大於目前數值的值。

校準步驟

1. 點擊[執行]按鈕。將計算出最佳放入零件數量。
2. 點擊[套用]按鈕。將會儲存結果。

3.2.4.3 翻轉&分離 - 自動校準

執行翻轉和分離的校準。在未啟用翻轉時，此項目將會顯示為「分離」。
如需詳細資訊，請參閱以下內容。
一般設定
請在以下畫面選擇「自動校準」標籤。



提示

將平台類型設為「溝槽」、「孔」、「凹槽」時，不會顯示[自動校準]標籤。
如需詳細資訊，請參閱以下內容。
一般設定

顯示項目的說明

項目	說明
振動振幅	顯示透過自動校準取得的振動振幅強度。 單位：%
振動時間	顯示透過自動校準取得的振動持續時間。 單位：ms

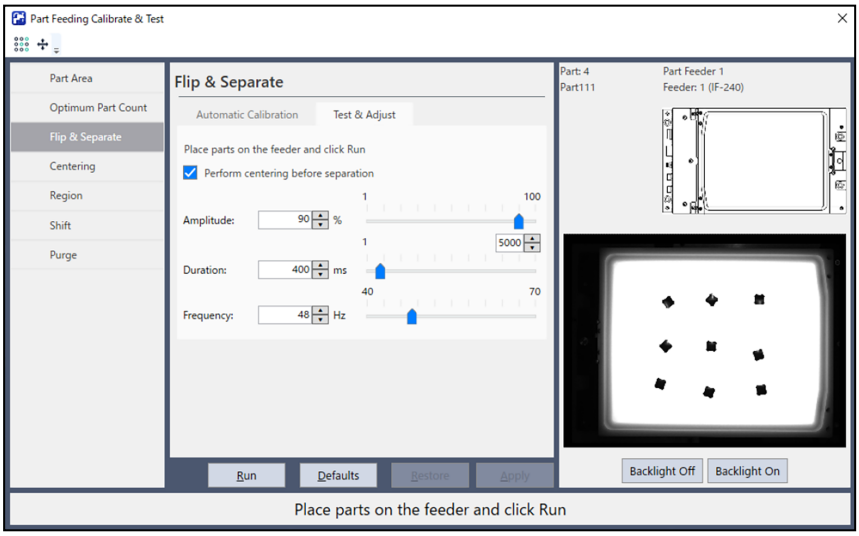
校準步驟

1. 放入所顯示數量的零件。

2. 點擊[執行]按鈕。
   將會嘗試進行送料器振動，並顯示出最佳的振動振幅、振動時間。
   可能會視零件的物理特性（重量、大小、材質、表面摩擦等）而花費數分鐘的動作時間。
3. 點擊[套用]按鈕。將會儲存結果。

3.2.4.4 翻轉&分離 - 測試和調整

調整翻轉和分離的動作參數。在未啟用翻轉時（參照一般設定），此項目將會顯示為「分離」。
選擇[測試和調整]標籤。



顯示項目的說明

項目	說明
在分離前進行集中	若有勾選此核取方塊，則會在測試動作時且在分離動作之前執行集中動作。 僅套用於測試動作。
振動振幅	設定振動振幅的強度。 單位：%
振動時間	設定振動的持續時間。 單位：ms
振動頻率	設定振動的頻率。 單位：Hz

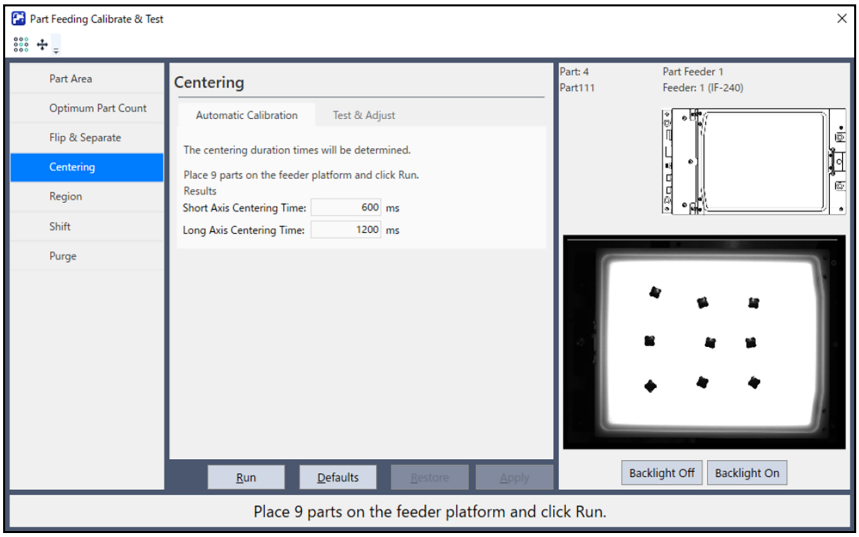
調整指南
調整振幅及頻率，讓零件能盡可能快速分散或改變姿態，進而避免零件飛出平台。
將能充分進行分散或姿態變更的最短時間設定為振動時間。

測試步驟

1. 放入適當數量（例如：最佳放入零件數量）的零件。
2. 點擊[執行]按鈕。
3. 確認運作。
   根據需要調整參數，並再次點擊[執行]按鈕。
   請參閱以下內容。
   送料器參數的調整方法

3.2.4.5 集中 - 自動校準

執行集中校準。
選擇[自動校準]標籤。



提示

- 將平台類型設為「溝槽」、「孔」、「凹槽」時，不會顯示[自動校準]標籤。如需詳細資訊，請參閱以下內容。
  振動
- 使用IF-80時，則不會顯示[自動校準]標籤。

顯示項目的說明

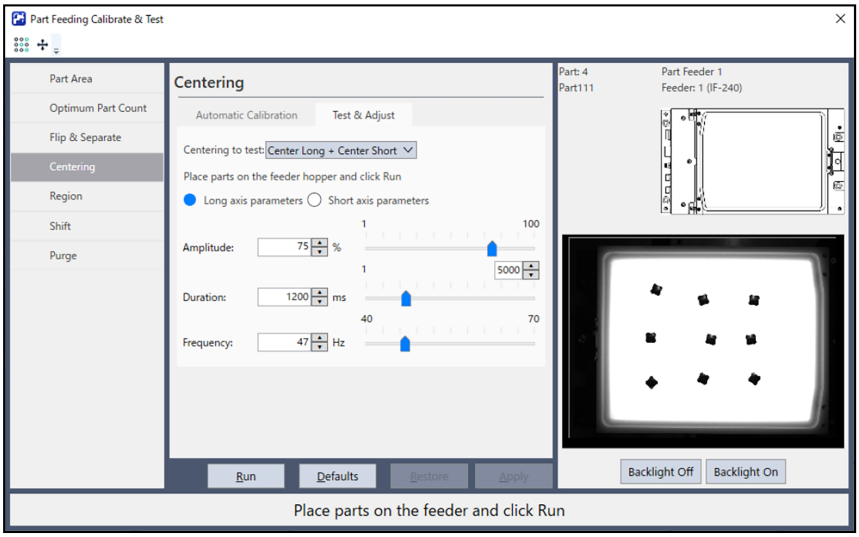
項目	說明
短軸集中時間	顯示短軸集中的持續時間。 單位：ms
長軸集中時間	顯示長軸集中的持續時間。 單位：ms

校準步驟

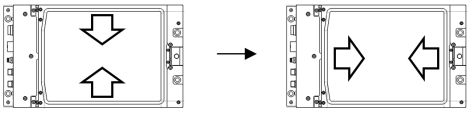
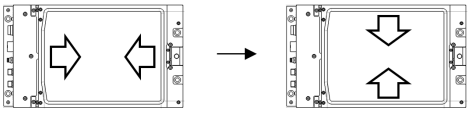
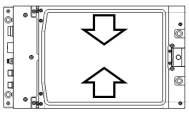
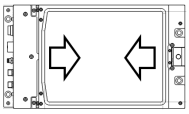
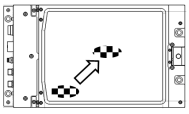
1. 放入所顯示數量的零件。
2. 點擊[執行]按鈕。
   將會嘗試進行送料器振動，並顯示出最佳的振動振幅、振動時間。
3. 點擊[套用]按鈕。將會儲存結果。

3.2.4.6 集中 - 測試和調整

調整集中的動作參數。
請選擇[測試和調整]標籤。



顯示項目的說明

項目	說明
測試集中	選擇要測試的集中動作類型。 長軸 + 短軸  短軸 + 長軸  長軸  短軸  使用位移的集中 
長軸的參數/短軸的參數	選擇要調整長軸或短軸的動作參數。 已選擇「使用位移的集中」時則不會顯示。
振動振幅	設定振動振幅的強度。 單位：% 已選擇「使用位移的集中」時則不會顯示。

項目	說明
振動時間	設定振動的持續時間。 單位：ms 已選擇「使用位移的集中」時則不會顯示。
振動頻率	設定振動的頻率。 單位：Hz 已選擇「使用位移的集中」時則不會顯示。

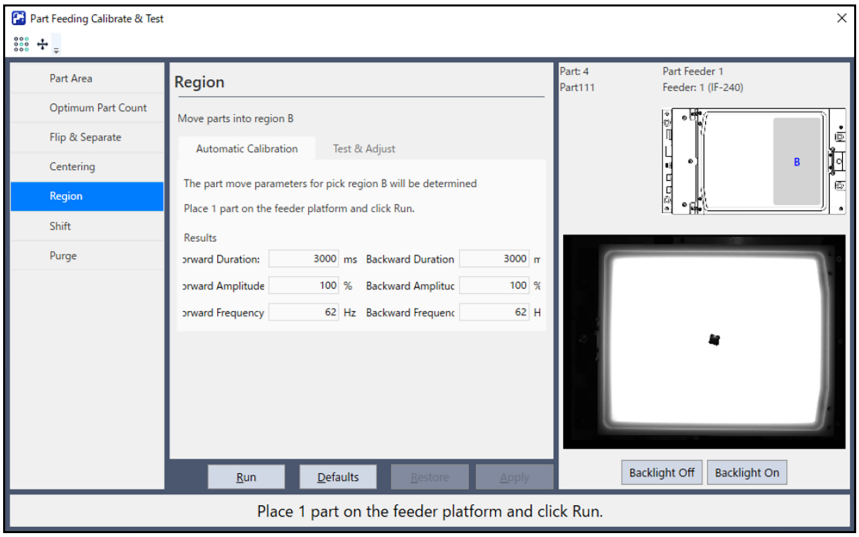
調整指南
調整振幅及頻率，讓零件能盡可能快速朝指定方向集中。將完成集中的時間設定為振動時間。

測試步驟

1. 放入適當數量（例如：最佳放入零件數量）的零件。
2. 點擊[執行]按鈕。
3. 確認運作。根據需要調整參數，並再次點擊[執行]按鈕。
   請參閱以下內容。
   送料器參數的調整方法

3.2.4.7 區域 - 自動校準

執行拾取區域位移的校準。指定拾取區域時會顯示此項目。
如需詳細資訊，請參閱以下內容。
拾取
請選擇[自動校準]標籤。



提示

將平台類型設為「溝槽」、「孔」、「凹槽」時，不會顯示[自動校準]標籤。
如需詳細資訊，請參閱以下內容。
振動

顯示項目的說明

項目	說明
前向位移時間	顯示前向位移動作的時間。 單位：ms
前向位移振幅	設定振動振幅的強度。 單位：%
前向位移頻率	設定振動的頻率。 單位：Hz
後向位移時間	顯示後向位移的時間。 單位：ms
後向位移振幅	設定振動振幅的強度。 單位：%
後向位移頻率	設定振動的頻率。 單位：Hz

校準步驟

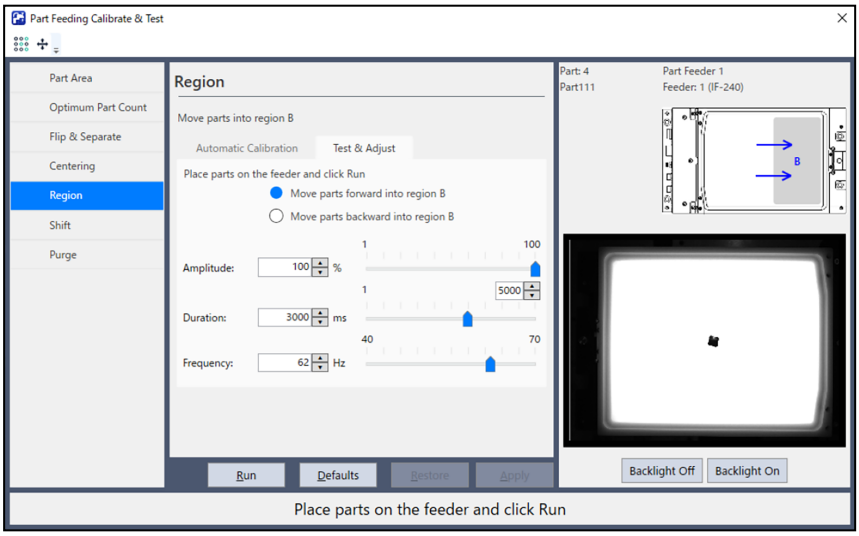
1. 放入1個零件。
2. 點擊[執行]按鈕。
   將會嘗試進行送料器振動，並顯示出最佳的振動振幅、振動時間與振動頻率。
3. 點擊[套用]按鈕。將會儲存結果。

3.2.4.8 區域 - 測試和調整

調整拾取區域位移的動作參數。指定拾取區域時會顯示此項目。
如需詳細資訊，請參閱以下內容。

拾取

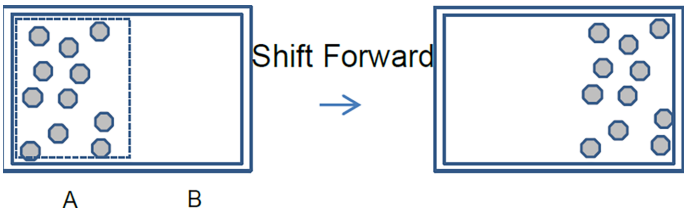
請選擇[測試和調整]標籤。



顯示項目的說明

項目	說明
向前位移後移動至區域x/向後位移後移動至區域x	選擇要測試的位移類型。x則代表在拾取畫面中選擇的零件區（A、B、C、D中任一區域）。 位移方向會根據送料器的設置方向而改變。 實際位移方向將會顯示在畫面右上方。 如需詳細資訊，請參閱以下內容。 拾取
振動振幅	設定振動振幅的強度。 單位：%
振動時間	設定振動的持續時間。 單位：ms
振動頻率	設定振動的頻率。 單位：Hz

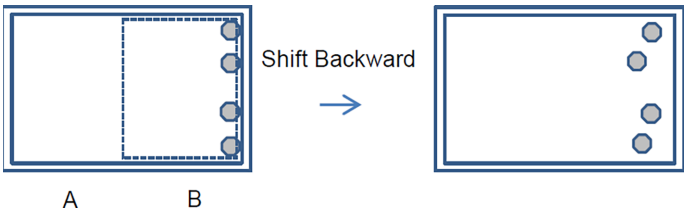
前向位移的調整指南、測試步驟
前向位移是指從指定區域開始持續保持拾取動作，區域內的零件減少時，透過位移動作將零件位移到指定區域的操作。
向前位移至區域B時的理想動作結果如下圖所示。



調整振動時間，以使零件移動適當距離（平台邊長的一半）。若零件移動過多，將會導致零件滯留在平台邊緣。若移動量不足，則無法朝區域內充分供應零件、使效率降低。調整振幅和頻率，以使零件移動適當距離（移動時，保持移動前的零件分布狀態）。若零件移動不順暢，將會導致區域內的零件互相接觸或重疊、使效率降低。

1. 將適當數量（例如：最佳放入零件數量的1/2）的零件放入所顯示區域。
2. 點擊[執行]按鈕。
3. 確認運作。
   根據需要調整參數，並再次點擊[執行]按鈕。請參閱以下內容。
   送料器參數的調整方法

後向位移的調整指南、測試步驟 後向位移是指在重複進行前向位移後、零件滯留在平台角落時，透過位移讓零件位移（返回）到指定區域的操作。向後位移至區域B時的理想動作結果如下圖所示。



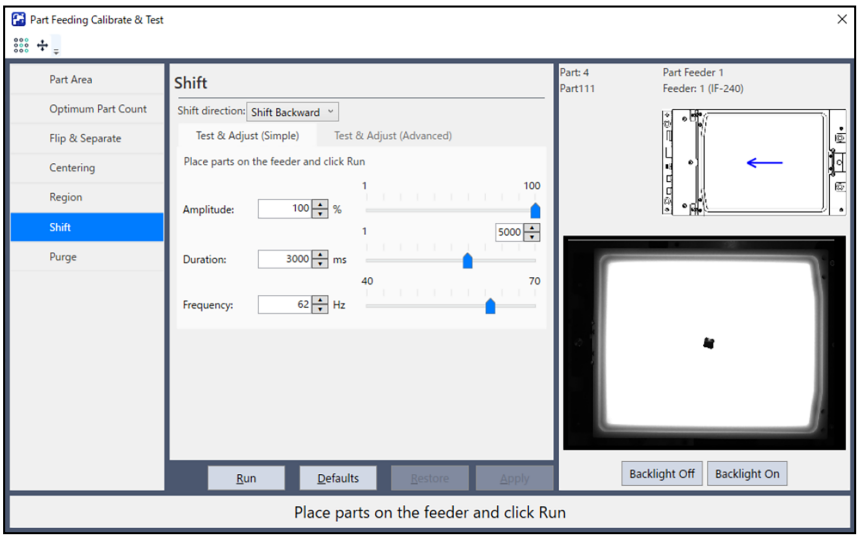
調整振動時間（以及振幅、頻率），以使零件移動適當距離（讓接觸到平台邊緣的零件返回到可拾取位置的距離）。若零件移動過多，將會讓零件超出區域、使效率降低。

1. 將適當數量（例如：約4個）的零件放入所顯示區域的角落（平台的角落）。
2. 點擊[執行]按鈕。
3. 確認運作。
   根據需要調整參數，並再次點擊[執行]按鈕。

請參閱以下內容。
送料器參數的調整方法

3.2.4.9 位移 - 測試和調整（簡易）

調整位移的動作參數。
請選擇[測試和調整（簡易）]標籤。



顯示項目的說明

項目	說明	
位移方向	選擇要測試的位移方向。	
	位移方向	**零件移動方向**
	前	
	左前	
	右前	
	左	
	右	
	後	
	左後	
	右後	
	位移方向會根據送料器的設置方向而改變。實際位移方向將會顯示在畫面右上方。	

項目	說明
振動振幅	設定振動振幅的強度。 單位：%
振動時間	設定振動的持續時間。 單位：ms
振動頻率	設定振動的頻率。 單位：Hz

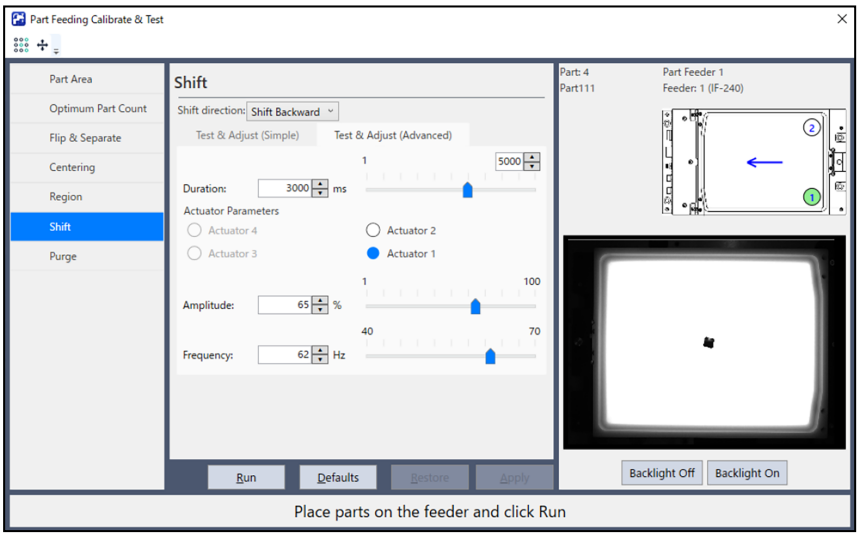
調整指南
調整振幅及頻率，讓零件盡可能快速且順暢地朝指定方向移動。
將完成預期動作的時間設定為振動時間。

測試步驟

1. 放入適當數量（例如：約4個）的零件。
2. 點擊[執行]按鈕。
3. 確認運作。
   根據需要調整參數，並再次點擊[執行]按鈕。
   請參閱以下內容。
   送料器參數的調整方法

3.2.4.10 位移 - 測試和調整（詳細）

調整位移的動作參數。
請選擇[測試和調整（詳細）]標籤。



顯示項目的說明

<table>
<tr><td>項目</td><td colspan="2">說明</td></tr>
<tr><td rowspan="10">位移方向</td><td colspan="2">選擇要測試的位移方向。</td></tr>
<tr><td>零件移動方向</td><td>動作</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>後</td><td>  </td></tr>
<tr><td>左後</td><td>  </td></tr>
<tr><td>右後</td><td>  </td></tr>
<tr><td colspan="2">位移方向會根據送料器的設置方向而改變。實際位移方向將會顯示在畫面右上方。</td></tr>
<tr><td>振動時間</td><td colspan="2">設定振動的持續時間。
此數值為所有致動器通用。
單位：ms</td></tr>
<tr><td>致動器1
致動器2
致動器3
致動器4</td><td colspan="2">選擇要設定的致動器（送料器內部的振動體）。
致動器的位置將顯示在畫面右上方。</td></tr>
<tr><td>振動振幅</td><td colspan="2">設定振動振幅的強度。
單位：%</td></tr>
<tr><td>振動頻率</td><td colspan="2">設定振動的頻率。此數值為所有致動器通用。
單位：Hz</td></tr>
</table>

調整指南 調整振幅及頻率，讓零件盡可能快速且順暢地朝指定方向移動。將完成預期動作的時間設定為振動時間。

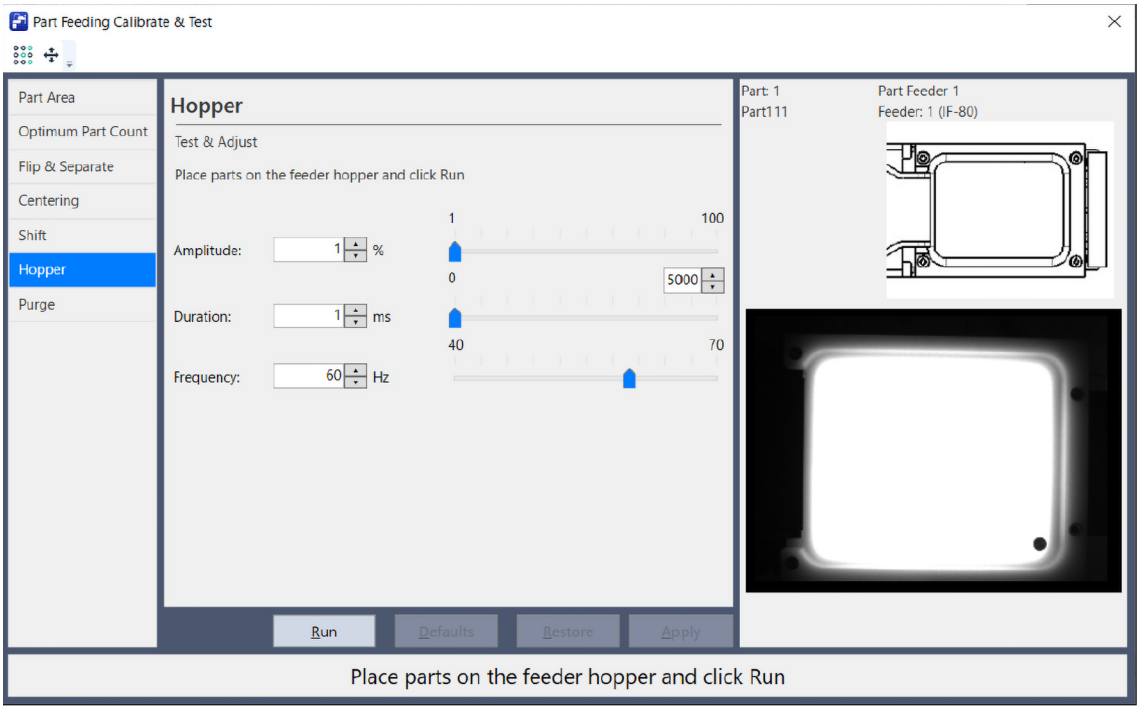
測試步驟

1. 放入適當數量（例如：約4個）的零件。
2. 點擊[執行]按鈕。
3. 確認運作。
   根據需要調整參數，並再次點擊[執行]按鈕。

請參閱以下內容。
送料器參數的調整方法

3.2.4.11 料斗 - 測試和調整

使用至少1個料斗時，可使用「料斗 - 測試和調整」畫面。
請參閱以下內容。
料斗的調整方法



顯示項目的說明

項目	說明
振動振幅（僅限IF-80料斗（Gen.2））	設定振動振幅的強度。 單位：%
振動時間	設定測試時的振動持續時間。 單位：ms
振動頻率（僅限IF-80）	設定振動的頻率。 單位：Hz

調整指南
關於調整的詳細資訊，請參閱以下內容。
料斗的調整方法

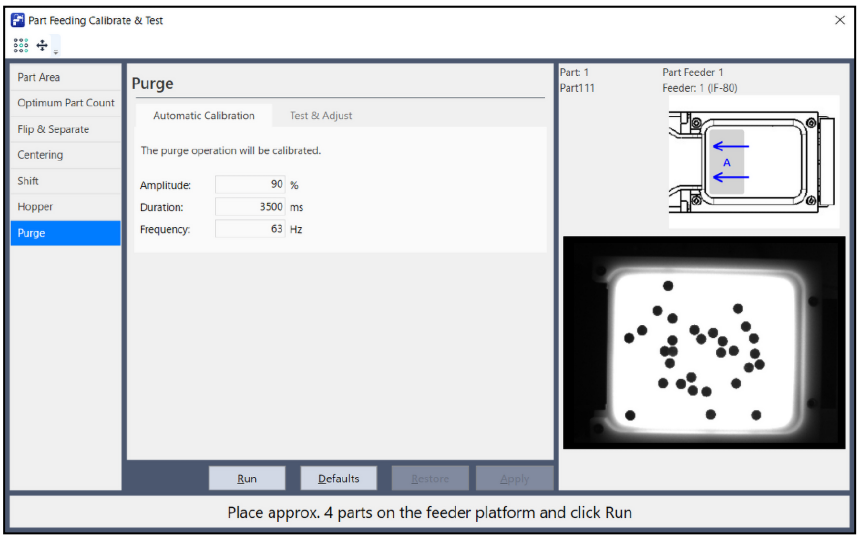
提示

- 連接2個料斗時，料斗畫面上會顯示下拉式功能表，用於選擇要用在測試和調整的料斗。

- 變更設定時，請在變更料斗編號前點擊[套用]按鈕。在套用變更之前，若有變更了料斗編號，則會顯示訊息讓您選擇是否套用變更或恢復成原本的設定。

3.2.4.12 清除 - 自動校準（僅IF-80）

執行IF-80的清除（排出送料器上的零件）校準。在啟用清除功能時會顯示此項目。
如需詳細資訊，請參閱以下內容。
清除
請選擇[自動校準]標籤。



提示

將平台類型設為「溝槽」、「孔」、「凹槽」時，則不會顯示[自動校準]標籤。
如需詳細資訊，請參閱以下內容。
振動

顯示項目的說明

項目	說明
振動振幅	設定振動振幅的強度。 單位：%
振動時間	設定振動的持續時間。 單位：ms
振動頻率	設定振動的頻率。 單位：Hz

校準步驟

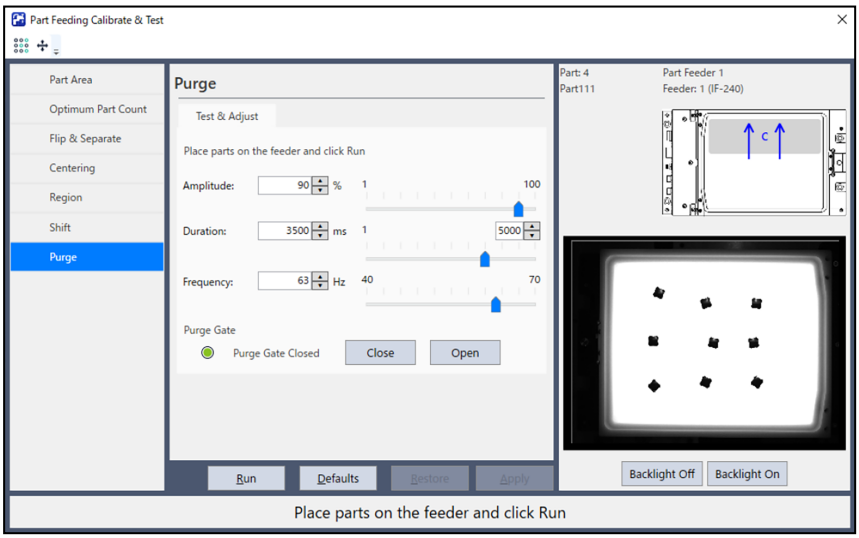
1. 清空清除箱（選購品）。
2. 放入所顯示數量的零件。
3. 點擊[執行]按鈕。
   將會嘗試進行送料器振動，並顯示出最佳的振動振幅、振動時間。
   可能會視零件的物理特性（重量、大小、材質、表面摩擦等）而花費數分鐘的動作時間。

4. 點擊[套用]按鈕。將會儲存結果。

3.2.4.13 清除 - 測試和調整

調整清除（排出送料器上的零件）的動作參數。在啟用清除功能時會顯示此項目。
如需詳細資訊，請參閱以下內容。

清除



顯示項目的說明

項目		說明
振動振幅		設定振動振幅的強度。 單位：%
振動時間		設定振動的持續時間。 單位：ms
振動頻率		設定振動的頻率。 單位：Hz
清除閘門		操作清除閘門。
	關閉清除閘門	顯示清除閘門感測器的狀態。 關閉：綠色；未關閉：灰色
	開啟/關閉	開啟或關閉清除閘門。

提示

即使將振動振幅設為0%，送料器仍會稍微振動。

此屬規格內的正常現象。

提示

只有在Epson RC+ 8.0功能表 - [設置] - [系統配置] - [控制器] - [零件送料器]中有勾選[安裝清除閘門]核取方塊時，才會在閘門關閉狀態下顯示清除標籤。

提示

在未關閉清除閘門時，若要切換到其他畫面，則會顯示「清除閘門將會關閉。是否繼續執行？」的對話方塊。按下「確定」之後，將會關閉清除閘門。

調整指南 調整振幅及頻率，讓零件盡可能快速地從送料器上排出。
將完全排出零件的所需時間設定為振動時間。

測試步驟

1. 放入適當數量（例如：最佳放入零件數量）的零件。
2. 開啟清除閘門（由您自行準備）。
   使用IF-80時，則清空清除箱（選購品）。
3. 點擊[執行]按鈕。
4. 確認運作。
   根據需要調整參數，並再次點擊[執行]按鈕。
   請參閱以下內容。
   送料器參數的調整方法

3.2.4.14 送料器參數的調整方法

提示

各參數已預先設定最佳值。因此在一般狀況下，不需要手動調整這些參數。

- 振動振幅
  設定較大的值可加快零件的移動速度。藉此可縮短零件分散或移動的完成時間，有效縮短週期時間。
  但若設定過大的值，零件可能會飛出平台。建議設定為能充分讓零件移動及分散的最小值。

- 振動時間
  設定較長的時間可增加零件的分散量和移動量。可藉此增加1次送料器運作中可取得的零件數量。
  但若設定過長的時間，將會加長週期時間。建議設定為能充分讓零件移動及分散的最小值。

- 振動頻率
  此值已預設為由平台重量和送料器內建彈簧之常數所決定的共振頻率。因此，在處理較重的零件（如金屬製品）時，設定為略小於預設值的數值，可能會改善動作。此外，使用您製作的平台時，變更頻率後可能改善零件的移動行為。
  然而，變更此數值可能會產生零件移動行為改變等副作用，因此請謹慎變更數值。
  變更頻率可能會讓產生的振動聲音變大。此變化並不屬於故障現象。
  若擔心振動聲音，請針對會產生最大噪音的頻率（=共振頻率）變更幾赫茲。

3.2.5 [檔案]功能表

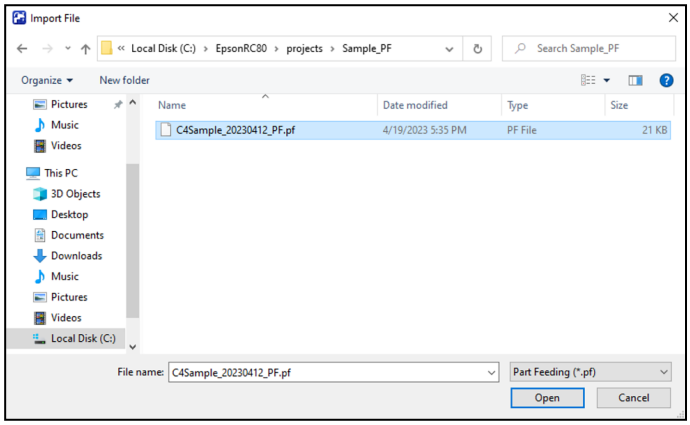
在Epson RC+ 8.0功能表 - [檔案]中，進行Part Feeding的各種檔案操作。

3.2.5.1 [匯入](檔案功能表)

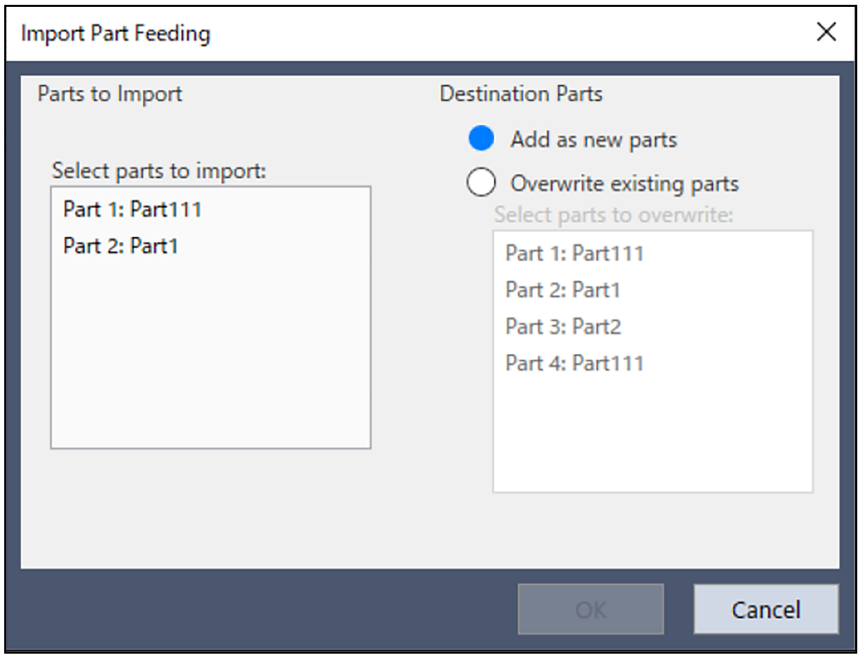
可從其他的Epson RC+ 8.0專案匯入零件設定。

1. 選擇Epson RC+ 8.0功能表 - [檔案] - [匯入]。

2. 在[檔案類型]中，選擇「料件送料（*.pf)」。



3. 在以下畫面進行中進行匯入操作。



項目	說明
選擇要匯入的零件	選擇匯入來源的零件。
新增新零件	在要新增為新零件時選擇。
覆蓋現有的零件	在要覆蓋現有的零件資料時選擇。
選擇要覆蓋的零件	在有選擇[覆蓋現有的零件]時，選擇要覆蓋的零件。

3.3 Part Feeding SPEL+命令參考手冊

下面說明您的SPEL+程式中可使用的Part Feeding SPEL+命令。以下為命令列表：

命令/函數	描述/用法
PF_Abort	強制終止Part Feeding的程序運作
PF_AccessFeeder	取得讓機器人存取送料器的鎖定
PF_ActivePart	在採取多種零件運作時切換主動零件

命令/函數	描述/用法
PF_Backlight	開啟/關閉背光
PF_BacklightBrightness	設定背光亮度
PF_BacklightColor	變更送料器內建背光的顏色
PF_Center	執行集中動作
PF_CenterByShift	執行使用位移的集中動作
PF_Flip	執行翻轉動作
PF_Hopper	控制料斗
PF_Info函數	取得Part Feeding選配件的屬性
PF_InitLog	啟用日誌檔輸出並指定輸出目標路徑
PF_IsStopRequested函數	回傳是否已發出PF_Stop
PF_Name$函數	透過零件ID取得零件名稱
PF_Number函數	透過零件名稱取得零件ID
PF_Purge函數	執行清除動作
PF_Output	控制送料器的輸出端子（使用2台料斗（Gen.1）時）
PF_OutputOnOff	控制送料器的輸出端子（使用1台料斗（Gen.1）時）
PF_PurgeGate	控制清除閘門的開關
PF_PurgeGateStatus函數	取得清除閘門關閉感測器的狀態
PF_QtyAdjHopperTime函數	回傳適當的料斗運作時間
PF_QueAdd	在座標佇列中新增資料 （點資料、姿態、使用者資料）
PF_QueAutoRemove	設定座標佇列的自動刪除功能
PF_QueAutoRemove函數	回傳座標佇列自動刪除功能的狀態
PF_QueGet函數	從座標佇列取得點資料
PF_QueLen函數	回傳座標佇列中已註冊資料的數量
PF_QueList	顯示座標佇列的資料列表
PF_QuePartOrient	設定、顯示座標佇列的零件姿態
PF_QuePartOrient函數	從座標佇列取得零件姿態
PF_QueRemove	刪除座標佇列的資料

命令/函數	描述/用法
PF_QueSort	設定、顯示座標佇列的Sort方法
PF_QueSort函數	回傳座標佇列的Sort方法
PF_QueUserData	設定、顯示座標佇列的使用者資料
PF_QueUserData函數	透過座標佇列取得使用者資料
PF_ReleaseFeeder	解除透過PF_AccessFeeder取得的鎖定
PF_Shift	執行位移動作
PF_Start	啟動指定零件的Part Feeding程序
PF_Stop	發出Part Feeding程序的終止請求

3.3.1 PF_Abort

強制終止指定零件的Part Feeding程序運作。

格式
PF_Abort 零件ID

參數

- 零件ID
  指定零件ID（整數值1～32）。

傳回值
無

描述
立即中止指定零件的Part Feeding程序。
與PF_Stop不同，正在執行的回呼函數會被暫停。
即使未使用PF_Start啟動Part Feeding運作程序，也能使用PF_Abort停止平台或料斗的振動。
在多種零件運作中使用PF_Abort時，設定PF_Start中設定的任一ID，即可暫停Part Feeding程序。
無法從虛擬控制器及命令視窗執行。

使用範例

```
PF_Abort 1
```

3.3.2 PF_AccessFeeder

PF_AccessFeeder用於多台機器人/1台送料器的系統中，防止機器人之間的碰撞。在有2台機器人同時共享同一個送料器時，必須使用此命令。PF_AccessFeeder取得存取送料器的機器人鎖定。
若已取得鎖定，PF_AccessFeeder會暫停任務，直到鎖定被釋放或達到指定的超時時間（選項）。
機器人完成使用送料器後，使用PF_ReleaseFeeder陳述式解除鎖定，以將送料器釋放給其他機器人。
如需詳細資訊，請參閱以下內容。
PF_ReleaseFeeder

範例：
在任務1中執行PF_AccessFeeder 1。任務1繼續執行。

在任務2中執行PF_AccessFeeder 1時，任務2會暫停。
在任務1中執行PF_ReleaseFeeder 1時，任務2會恢復執行。

本命令用於多台機器人配置的互斥控制。

格式
PF_AccessFeeder 送料器編號/送料器名稱[, 超時]

參數

- 送料器編號
  以運算式或數值（1～4的整數值）指定送料器編號。
- 送料器名稱
  指定送料器標籤（字串）。
- 超時
  以運算式或數值（整數）指定鎖定解除等待的超時時間（秒）。選填。

傳回值
無

描述
若指定了超時，可透過TW函數的回傳值判斷結果。
有關詳細資訊，請參閱以下手冊。
「Epson RC+ 8.0 SPEL+語言參考- TW 函數」

- 鎖定已釋放或已取得鎖定（False）
- 已達到超時（True）

送料器的動作不受取得/釋放鎖定的影響。
任務結束時，該任務中取得的鎖定會自動釋放。
在同一任務中，若對同一送料器連續執行兩次PF_AccessFeeder，將會出現錯誤。
無法從虛擬控制器及命令視窗執行。

使用範例
以下是2台機器人拾取1台送料器上的零件的範例。
機器人1取得送料器上的零件，並移動到點位place1。
當機器人1達到移動路徑的50%時，機器人2開始移動以取得零件。

```
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

在採取多種零件運作時切換主動零件。PF_ActivePart陳述式告知系統目前應處理哪個零件。系統會振動或供應零件，使機器人能拾取由PF_ActivePart指定的零件。

格式

PF_ActivePart 零件ID

參數

- 零件ID
  指定零件ID（整數值1～32）。

傳回值

無

描述
多種零件運作時，PF_Start陳述式的第1個引數的零件ID是最初的主動零件。若想使用其他零件，可使用PF_ActivePart陳述式切換主動零件。系統對主動零件執行動作（使用送料器振動參數、從料斗供應零件等）。
PF_ActivePart陳述式通常在PF_Robot回呼函數結束前設定。
若未啟動Part Feeding運作程序，即使執行本命令也不會有任何反應。
非多種零件運作時（執行PF_Start時僅指定1個ID），即使執行本命令也不會有任何反應。
若指定了未在多種零件運作中被指定的零件ID，即使執行本命令也不會有任何反應。
無法從虛擬控制器及命令視窗執行。

使用範例
此例展示如何使用PF_ActivePart在零件1和零件2之間切換。

```
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

開啟/開關送料器內建的背光。

格式
PF_Backlight 送料器編號 | 送料器名稱, On | Off

參數

- 送料器編號
  以運算式或數值指定送料器編號（1～4的整數值）。
- 送料器名稱
  以字串指定送料器名稱。
- On/Off
  指定On/Off。

傳回值
無

描述
當系統自動進行視覺處理時，背光會自動開啟/關閉。
使用PF_Vision回呼函數時，使用此命令開啟/關閉背光。
IF-80的情況下，背光點亮後30秒即會自動熄滅。（依亮度不同，時間會有變化。）

自動熄滅的情況下，再次點亮前需執行PF_Backlight Off命令。
無法從虛擬控制器及命令視窗執行。

使用範例

```
PF_Backlight 1, On
```

3.3.5 PF_BacklightBrightness

設定送料器內建背光的亮度。

格式
PF_BacklightBrightness 送料器編號 | 送料器名稱, 亮度

參數

- 送料器編號
  以運算式或數值指定送料器編號（1～4的整數值）。
- 送料器名稱
  以字串指定送料器名稱。
- 照度
  以0%～100%的值（整數值）指定照度。

傳回值
無

描述
通常，內建背光的照度在[料件送料]對話方塊中設定。執行時若需變更照度，可使用此命令進行變更。
無法從虛擬控制器及命令視窗執行。

> 提示
>
> 照度可在0～100%的範圍內以百分比設定，但使用IF-240時，即使設定25%以下的照度，實際照度的最小值也會限制在25%。這是因為在某一照度以下無法得到均勻的明亮度。

若想將照度設為0%，請使用PF_Backlight陳述式以關閉背光。

使用範例

```
PF_BacklightBrightness 1, 80
```

3.3.6 PF_BacklightColor

變更送料器內建背光的顏色。
本命令專用於IF-A1520及IF-A2330。

格式
PF_BacklightColor 送料器編號 | 送料器名稱, 顏色

參數

- 送料器編號
  以運算式或數值指定送料器編號（1～4的整數值）。

- 送料器名稱
  以字串指定送料器名稱。

- 顏色
  指定顏色。

值	顏色
PF_BACKLIGHTCOLOR_WHITE	白
PF_BACKLIGHTCOLOR_RED	紅

傳回值
無

描述
內建背光的顏色在[料件送料]對話方塊中設定。執行時若需變更顏色，可使用此命令進行變更。無法從虛擬控制器執行。

使用範例

```
PF_BacklightColor 1, 1
```

3.3.7 PF_Center

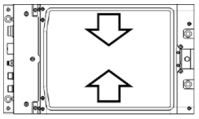
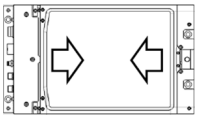
執行送料器的集中動作。
IF-240、IF-380、IF-530、IF-A1520、IF-A2330可使用。IF-80不可使用。

格式
PF_Center 零件ID, 方向 [, 動作時間]

參數

- 零件ID
  指定零件ID（整數值1～32）。

- 方向
  指定集中方向。

方向	值（PartFeeding.inc中定義）	零件移動方向
長軸	PF_CENTER_LONG_AXIS	
短軸	PF_CENTER_SHORT_AXIS	

- 動作時間
  以毫秒指定動作時間（整數值1～30000）。省略時，使用透過送料器校準算出的值。
  指定-1時，將進行與省略時相同的動作。

傳回值
無

描述
執行IF系列送料器的集中動作。
在以下情況下使用集中動作：

- 分散零件前將零件集中到中央
- 使細長零件的方向一致

本命令可在您的函數中直接使用。也可在執行PF_Start命令時呼叫的各回呼函數內部使用。

在以下條件下無法執行：

- 從使用者函數執行時：命令中指定的送料器（以零件ID指定的送料器）正在Part Feeding程序（PF_Start命令）中使用（錯誤7733）
- 在回呼函數中執行時：
  指定了未在PF_Start命令中指定的零件ID（錯誤7733）
- 從虛擬控制器及命令視窗執行（錯誤3804）

本命令在內部處理中使用SyncLock。如需詳細資訊，請參閱以下內容。
Part Feeding程序所使用的功能

使用範例

```
PF_Center 1,1
```

3.3.8 PF_Flip

執行翻轉動作。

格式
PF_Flip 零件ID [, 動作時間]

參數

- 零件ID
  指定零件ID（整數值1～32）。
- 動作時間
  以毫秒指定動作時間（整數值1～30000）。
  省略時，使用透過送料器校準算出的值。
  指定-1時，將進行與省略時相同的動作。

傳回值
無

描述
執行翻轉動作。
翻轉動作用於以下情況：

- 分散零件（使用較長的動作時間）
- 改變零件姿態（使用較短的動作時間）

在以下條件下無法執行：

- 從使用者函數執行時：本命令中指定的送料器（以零件ID指定的送料器）正在Part Feeding程序（PF_Start命令）中使用（錯誤7733）

- 在回呼函數中執行時：指定了未在PF_Start命令中指定的零件ID（錯誤7733）
- 從虛擬控制器及命令視窗執行（錯誤3804）

本命令在內部處理中使用SyncLock。如需詳細資訊，請參閱以下內容。
Part Feeding程序所使用的功能

使用範例

```
PF_Flip 1,500
```

3.3.9 PF_Hopper

控制料斗。

格式
PF_Hopper 料斗編號_A, 零件ID_A [,動作時間_A] [,料斗編號_B] [,零件ID_B] [動作時間_B]

參數

- 料斗編號_A
  指定料斗編號（整數值1～2）。
- 零件ID_A
  指定零件ID（整數值1～32）。
- 動作時間_A
  選填。指定料斗A的振動時間（整數值1～30000）。（單位：毫秒）
  省略動作時間_A或指定-1時，使用料件送料校準畫面中設定的值。指定0時，料斗A不會振動。若已在動作，則會停止振動。
- 料斗編號_B
  選填。指定料斗編號（整數值1～2）。
- 零件ID_B
  選填。指定零件ID（整數值1～32）。
- 動作時間_B
  選填。指定料斗B的振動時間（整數值1～30000）。（單位：毫秒）
  省略動作時間_B或指定-1時，使用料件送料校準畫面中設定的值。指定0時，料斗B不會振動。若已在動作，則會停止振動。

傳回值
無

描述
執行本命令後，立即恢復控制。
讓設定在零件ID_A中的零件使用料斗編號_A；讓設定在零件ID_B中的零件使用料斗編號_B。在料件送料對話方塊的零件供應畫面中設定用於零件的料斗。

執行PF_Abort命令時，料斗會停止。執行PF_Stop命令時，不會停止。

PF_Hopper可與送料器振動同時執行。
IF-80對送料器僅支援1個料斗，因此使用IF-80時將料斗編號設為1。

無法從虛擬控制器及命令視窗執行。
選擇系統配置中未啟用的料斗編號時，會出現錯誤。

Note：PF_OutputOnOff、PF_Output僅在使用料斗（Gen.1）時進行動作。
PF_Hopper需用於控制料斗（Gen.2），也可用於控制料斗（Gen.1）。

使用範例1：
使零件9的料斗（Gen.2）1動作3秒。料斗（Gen.2）的振動振幅已在料件送料校準畫面中設定。

```
PF_Hopper 1,9,3000
Wait 3  '等待料斗運作結束。
```

使用範例2：
依照料件送料校準對話方塊的料斗畫面中所指定的時間使零件1的料斗2進行動作。

```
PF_Hopper 2,1
```

使用範例3：
立即停止使用範例2的料斗2。

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

使用範例4：
同時使零件2的料斗1動作500 ms，使零件3的料斗2動作400 ms。

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

使用範例5：
立即停止使用範例4的料斗1、2。

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

3.3.10 PF_CenterByShift

執行使用位移的集中動作。

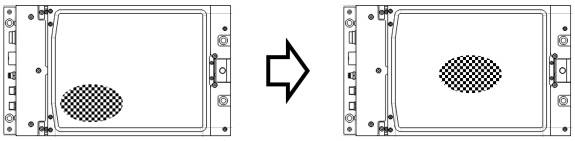
格式
PF_CenterByShift 零件ID

參數

- 零件ID
  指定零件ID（整數值1～32）。

傳回值
無

描述
執行使用位移的集中動作。



在以下情況下使用集中動作：

- 分散零件前將零件集中到中央
- 當末端夾具與平台發生干擾時，將零件集中到中央

執行本命令，零件Blob視覺序列即會運作，並測量送料器上零件的重心。之後，將會進行位移動作，使零件重心位於送料器中央。
執行本命令時，若零件已分布在送料器的中央附近，送料器可能結束而不會進行動作。

在以下條件下無法執行：

- 從使用者函數執行時：
  本命令中指定的送料器（以零件ID指定的送料器）正在Part Feeding程序（PF_Start命令）中使用（錯誤7733）
- 在回呼函數中執行時：
  指定了未在PF_Start命令中指定的零件ID（錯誤7733）
- 從虛擬控制器及命令視窗執行（錯誤3804）

本命令在內部處理中使用SyncLock。如需詳細資訊，請參閱以下內容。
Part Feeding程序所使用的功能

使用範例

```
PF_CenterByShift 1
```

3.3.11 PF_Info函數

取得零件的屬性。

格式
(1) PF_Info(零件ID, 屬性ID)
(2) PF_Info(零件名稱, 屬性ID)

參數

- 零件ID
  指定零件ID（整數值1～32）。
- 零件名稱
  指定零件名稱（字串）。
- 屬性ID
  指定屬性ID。
  可取得的屬性如下：

屬性ID	內容
PF_INFO_ID_FEEDER_CALIB_CORRECT_MAXNUM	用於計算最佳數量校準結果的零件放入數量。
PF_INFO_ID_FEEDER_NO	傳回零件使用的送料器編號。
PF_INFO_ID_ROBOT_NO	傳回零件使用的機器人編號。

傳回值
傳回指定的屬性值。

描述
傳回零件的屬性。在回呼函數中，使用此值自訂各動作。
無法從虛擬控制器及命令視窗執行。

使用範例
請參閱以下內容。
PF_Control

3.3.12 PF_InitLog

啟用Part Feeding日誌檔輸出並指定輸出目標路徑。
在啟動PF_Start前執行。
若要輸出日誌檔，需要將安裝有RC+的PC連接到控制器。
關於Part Feeding日誌檔，請參閱以下內容。
Part Feeding日誌檔

格式
PF_InitLog零件, ID 輸出目標路徑, 追加

參數
- 零件ID
  指定零件ID（整數值1～32）。
- 輸出目標路徑
  指定日誌檔輸出目標路徑（PC上的路徑+檔案名稱）。
- 追加
  設定True時，若特定輸出目標路徑已存在，則追加資料。
  設定False，即會覆寫檔案。

傳回值
無

描述
請從執行PF_Start的相同任務執行。從不同任務執行時，不會輸出日誌。
若未啟動Part Feeding程序，則不執行任何動作。
控制器未連接PC時，即使執行本函數也不會輸出日誌，也不會出現錯誤。
若路徑不存在或寫入檔案失敗，執行PF_Start時會出現錯誤。執行本函數時不會出現錯誤。
無法從虛擬控制器及命令視窗執行。
在PF_Start命令的回呼函數內執行時，會記錄送料器控制命令（PF_Center、PF_CenterByShift、PF_Flip、PF_Shift）的運作日誌。在其他使用者函數內執行時，不會記錄日誌。
本命令在內部處理中使用計時器。如需詳細資訊，請參閱以下內容。
Part Feeding程序所使用的功能

使用範例
請參閱以下內容。
PF_Start / PF_Start函數

3.3.13 PF_IsStopRequested函數

檢查Part Feeding程序是否有終止請求（是否執行了PF_Stop）。
通常在回呼函數內部使用。

格式
PF_IsStopRequested(零件ID)

參數

- 零件ID
  指定零件ID（整數值1～32）。

傳回值
若在執行Part Feeding程序時呼叫了PF_Stop，則傳回True。
在其他情況下，會傳回False。

描述
在回呼函數內部使用，檢查Part Feeding程序是否有終止請求。進行如下程式設計：在執行迴圈處理等情況下，每次迴圈時呼叫此函數，並在有終止請求時跳出迴圈，結束回呼函數。
多種零件運作時，可指定PF_Start中指定的任一零件ID來檢查終止請求。例如，以零件ID 1、2、3、4開始多種零件運作，執行PF_Stop 2時，不論在本函數中指定零件ID 1、2、3、4的哪一個，都能檢查是否有終止請求。
無法從虛擬控制器及命令視窗執行。

使用範例
請參閱以下內容。
PF_Robot

3.3.14 PF_Name$函數

透過零件ID傳回零件名稱。

格式
PF_Name$(零件ID)

參數

- 零件ID
  指定零件ID（整數值1～32）。

傳回值
以字串傳回指定零件ID的名稱。

描述
指定的零件ID無效時，傳回「」（空字串）。
無法從虛擬控制器及命令視窗執行。

使用範例

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

3.3.15 PF_Number函數

透過零件名稱傳回零件ID

格式
PF_Number(零件名稱)

參數

- 零件名稱
  指定零件名稱（字串）。

傳回值
傳回指定零件名稱的零件ID（整數值1～32）。

描述
零件名稱不存在時，會傳回-1。
有多個同名零件時，傳回最小的零件ID。
無法從虛擬控制器及命令視窗執行。

使用範例

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

3.3.16 PF_Output

控制送料器的輸出端子。
當送料器與2台料斗連接，從各料斗供應零件時，可指定各料斗的ON時間。

格式
PF_Output 送料器編號/送料器名稱, Out端子1 ON時間, Out端子2 ON時間

參數

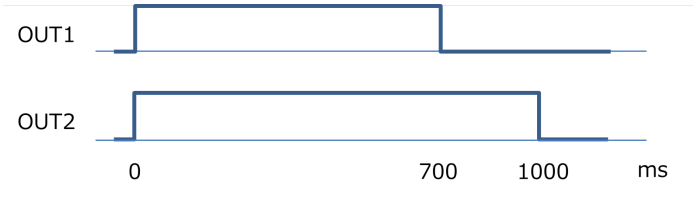
- 送料器編號
  以運算式或數值指定送料器編號（1～4的整數值）。
- 送料器名稱
  以字串指定送料器名稱。
- Out端子1 ON時間
  指定ON時間（整數值）。可指定為1 - 30000[ms]。
  ON時間為0時，將保持OFF狀態。
- Out端子2 ON時間
  指定ON時間（整數值）。可指定為1 - 30000[ms]。
  ON時間為0時，將保持OFF狀態。

傳回值
無

描述
執行本命令後，立即恢復控制。若要等待料斗動作結束，請在執行本命令後立即使用Wait命令。
在送料器振動期間或料斗運作期間執行本命令，這些動作即會停止，並開始執行本命令所指定的動作。
若要同時執行送料器振動和料斗運作，請使用PF_OutputOnOff命令。
執行PF_Abort命令時，料斗會停止。執行PF_Stop命令時，不會停止。
無法從虛擬控制器及命令視窗執行。
若在在[系統配置] - [控制器] - [零件送料器]中未勾選「安裝料斗」，將發生錯誤2584（指定了安裝料斗設定無效的零件送料器）。
IF-80無法使用（會出現錯誤2589「呼叫送料器不可執行的動作命令」）。請使用PF_OutputOnOff命令。
IF-240使用清除閘門時，Out端子2用於控制清除閘門，因此即使將Out端子2 ON時間設為非0值，Out端子2的輸出也不會變化。但ON時間不能省略，請指定適當的值（如0）。

使用範例

PF_Output 1, 700, 1000
輸出端子的輸出如下：



PF_Output 1, 0, 700
輸出端子的輸出如下：

3.3.17 PF_OutputOnOff

控制送料器的輸出端子。
從料斗供應零件時，以On執行此命令。停止從料斗供應零件時，以Off執行此命令。
IF-80內建料斗。其他送料器可選購料斗。

格式

(1) PF_OutputONOFF 送料器編號/送料器名稱, On, 輸出編號 [, 動作時間] [, 等待設定]
(2) PF_OutputONOFF 送料器編號/送料器名稱, Off

參數

- 送料器編號
  以運算式或數值指定送料器編號（1～4的整數值）。
- 送料器名稱
  以字串指定送料器名稱。
- On/Off
  指定On（1）/Off (0)。
- 輸出編號
  指定輸出目標。於開啟狀態時可指定。
  1：Out端子1
  2：Out端子2
- 動作時間
  指定On時間（整數值）。可指定為1 - 30000[ms]。
  On時間為0或省略時，指定為持續開啟。
- 等待設定
  設定是否加入等待。僅IF-80可設定。
  0或省略時：不等待料斗運作完成。
  1：等待料斗運作完成。

傳回值

無

描述
執行本命令後，會立即恢復控制（包含IF-80的等待設定為0或省略的情況）。若要等待料斗運作結束，請使用使用範例所示的Wait命令。
無法同時開啟輸出端子1和2。例如，開啟端子1時，端子2會關閉。
指定關閉時，端子1和2都會關閉。
執行PF_Abort命令時，料斗會停止。執行PF_Stop命令時，不會停止。
若在[系統配置] - [控制器] - [零件送料器]中未勾選「安裝實裝」，將發生錯誤2584（未啟用清除閘門）。
無法從虛擬控制器及命令視窗執行。
在IF-240中，使用清除閘門時，Out端子2用於控制清除閘門。因此，將輸出編號設為2時可以執行，但是Out端子2的輸出仍不會改變。

使用範例
請參閱以下內容。
PF_Control

3.3.18 PF_PurgeGate

控制清除閘門（選購品）的開關。

格式
PF_PurgeGate 送料器編號/送料器名稱, On/Off

參數

- 送料器編號
  以運算式或數值指定送料器編號（1～4的整數值）。
- 送料器名稱
  以字串指定送料器名稱。
- On/Off
  指定On（1）/Off (0)。指定On時，清除閘門會打開。
  指定Off時，清除閘門會關閉。

傳回值
無

描述
執行本命令後，立即恢復控制。
若要等待料斗運作結束，請參考以下使用範例。
PF_PurgeGate
若在[系統配置] - [控制器] - [零件送料器]中未勾選「安裝清除閘門」，將發生錯誤2593（未啟用送料器的清除輸出）。
無法從虛擬控制器及命令視窗執行。
清除閘門的開關動作在緊急停止、安全防護已打開、程式停止、執行PF_Stop的任何情況下，都不會立即停止。

使用範例1
等待清除閘門開啟後，進行下一個動作。

```
PF_PurgeGate 1, On
Wait 5.0

' 下一個動作
```

使用範例2
等待清除閘門關閉後，進行下一個動作。

備註：清除閘門關閉動作時若負載過載（如零件卡住等），會自動切換為開啟動作。以下程式是使用PF_PurgeGateStatus來檢測出此情況的範例。

```
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
   ' 錯誤處理 清除閘門未關閉
EndIf

' 下一個動作
```

3.3.19 PF_PurgeGateStatus函數

確認清除閘門（選購品）的關閉感測器狀態。

格式
PF_PurgeGateStatus(送料器編號/送料器名稱)

參數

- 送料器編號
  以運算式或數值指定送料器編號（1～4的整數值）。
- 送料器名稱
  以字串指定送料器名稱。

傳回值
關閉感測器為On（閘門已關閉）時傳回False。
關閉感測器為Off（閘門未關閉）時傳回True。

描述
清除閘門有關閉感測器，但沒有開啟感測器。也就是說，當清除閘門處於稍微開啟的狀態時，本函數會傳回True。
執行本函數後，會立即恢復控制。
若在[系統配置] - [控制器] - [零件送料器]中未勾選「安裝清除閘門」，將發生錯誤2593（未啟用送料器的清除輸出）。
無法從虛擬控制器及命令視窗執行。

使用範例
請參閱以下使用範例。
PF_PurgeGate

3.3.20 PF_Purge / PF_Purge函數

執行清除（排出送料器上的零件）動作。
執行清除動作時，無論成功或失敗，以零件ID指定的零件座標佇列都會被清除。

格式
PF_Purge 零件ID, 動作類型[, 振動時間[, 剩餘數量閾值[, 重試次數]]]
PF_Purge(零件ID, 動作類型[, 振動時間[, 剩餘數量閾值[, 重試次數]]])

參數

- 零件ID
  指定零件ID（整數值1～32）。
- 動作類型
  指定動作的類型。

值（PartFeeding.inc中定義）	內容
PF_PURGETYPE_NOVISION	不進行視覺回饋。 不計算零件剩餘數量。
PF_PURGETYPE_VISION	進行視覺回饋。 使用視覺系統，計算零件剩餘數量。

- 振動時間
  指定清除動作時間。（單位：毫秒）
  選填。
  省略時，使用以下設定的值。
  清除 - 測試和調整
  指定-1時，將進行與省略時相同的動作。
- 剩餘數量閾值
  指定要重試清除動作的零件剩餘數量（大於或等於此數量時會重試）。
  選填。
  動作類型為PF_PURGETYPE_NOVISION時，即使設定也不會改變動作。
  動作類型為PF_PURGETYPE_VISION時，若省略剩餘數量閾值，則會重試直到剩餘數量變為0為止。
- 重試次數
  指定清除動作重試次數的上限值。（初次清除動作也計入次數中。）
  選填。
  動作類型為PF_PURGETYPE_NOVISION時，即使設定也不會改變動作。
  動作類型為PF_PURGETYPE_VISION時，若省略重試次數，則會進行清除直到平台上剩餘的零件數量為「剩餘數量閾值」，或所有零件都從平台上移除為止（「剩餘數量閾值」和「重試」都省略的情況）。

傳回值

作為函數使用時，若零件正常排出以及將動作類型指定為PF_PURGETYPE_NOVISION（不計算零件剩餘數量），則傳回True。將動作類型指定為PF_PURGETYPE_VISION（計算零件剩餘數量）時，若達到重試次數但零件剩餘數量仍未低於閾值，則傳回False。

描述

IF-80可選擇支援清除的平台及清除容器（選購品）。
IF-240、IF-380、IF-530、IF-A1520及IF-A2330可選擇清除閘門（選購品）。清除閘門的有無在[設置] - [系統配置] - [控制器] - [零件送料器] - [送料器]中設定。清除閘門的使用/不使用可按每個零件進行設定。使用清除閘門時，清除閘門的開啟/關閉動作及開啟/關閉感測器的判斷會由PF_Purge陳述式自動處理。
使用您自製的清除閘門時，PF_Purge陳述式僅在平台上使零件位移（移動）。
動作類型為PF_PURGETYPE_VISION時，使用零件blob視覺序列來檢測平台上大約的零件數量。
關於清除的啟用/停用以及零件的清除方向，請參閱以下內容。
防止末端夾具的干擾
IF-80需要進行清除校準。
請參閱以下內容。
清除 - 自動校準（僅IF-80）
清除無效時執行本函數會發生錯誤，並以PF_STATUS_PURGENOTENABLED呼叫PF_Startus回呼函數。
無法從虛擬控制器及命令視窗執行。
在EPSON RC+ 7.5.0中，即使動作類型= 1，也無法省略剩餘數量閾值及重試。在這種情況下，請將剩餘數量閾值及重試指定為值「0」。

使用範例

範例1：
使用視覺回饋的清除範例。每次清除的振動時間為1500毫秒。嘗試執行直到零件數量為3個或以下。嘗試5次重試。

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

範例2：
不使用視覺回饋的清除範例。進行2000毫秒的清除。

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

3.3.21 PF_QtyAdjHopperTime函數

傳回放入適當零件量所需的料斗運作時間。

格式

PF_QtyAdjHopperTime(零件ID, 供應量, 供應時間)

參數

- 零件ID
  指定零件ID（整數值1～32）。
- 供應量
  指定供應時間內提供的零件量（個數）。
- 供應時間
  指定料斗運作時間[ms]，用於供應在供應量中所指定數量（個數）的零件。

傳回值

傳回料斗運作時間[ms]。傳回值為1～30000[ms]。
發生錯誤時，傳回1。

描述

PF_QtyAdjHopperTime使用視覺系統計算供應最佳零件數量（透過送料器校準進行決定）所需的料斗操作時間。回傳值可作為PF_OutputOnOff陳述式的持續時間參數使用。
SupplyQty和SupplyTime由開發者調整並指定。調整作業可使用Test Hopper按鈕。這些值表示在特定時間內從料斗供應的零件數量。如需詳細資訊，請參閱以下內容。
校準

多種零件運作時，計算的動作時間是以各零件放入同等數量為前提。PF_Start運作中執行PF_QtyAdjHopperTime時，會執行PF_Start執行中所指定的零件Blob視覺序列。在PF_Start未運作時，若執行PF_QtyAdjHopperTime，則僅使用以引數指定的零件ID之零件Blob序列。

此命令無法從虛擬控制器或命令視窗執行。
此命令無法用於移動式攝影機。
使用PF_Vision回呼函數時，PF_QtyAdjHopperTime僅使用零件Blob視覺序列來決定以零件ID指定的零件之料斗運作時間。
請注意，選擇「在取放期間供應零件」時，若在機器人手臂進入攝影機視野的狀態下執行本命令，可能會將手臂誤認為零件，導致傳回不正確的料斗運作時間。
如需詳細資訊，請參閱以下內容。
供應零件

使用範例

本用法範例中，使用PF_Control回呼，在為供應適當數量零件所需的估計時間內操作料斗（選購品）。假設料斗被調整為每秒（1000ms）供應10個零件。

```
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

在零件座標佇列上新增資料（點資料、零件姿態、使用者資料）。

格式

PF_QueAdd 零件ID, 點資料 [, 零件姿態 [, 使用者資料]]

參數

- 零件ID
  指定零件ID（整數值1～32）。
- 點資料
  用於指定點資料。
- 零件姿態
  以整數值指定與點資料一起註冊的零件姿態。
  選填。
  PF_PARTORIENT_FRONT=1：正面零件姿態
  PF_PARTORIENT_BACK=2：背面零件姿態
- 用戶資料
  以實數值指定和點資料一起註冊的使用者資料。
  選填。

傳回值

無

描述

用於在零件座標佇列中註冊任意資料（點資料、零件姿態、使用者資料）。
零件座標佇列會在Part Feeding程序中自動生成。通常無需使用PF_QueAdd。您進行自訂的視覺處理（使用PF_Vision回呼函數）時會使用本函數。
資料會被新增到指定零件ID的零件座標佇列末尾。但是，若透過PF_QueSort設定了Sort方法，則依該Sort方法進行註冊。
零件座標佇列的資料保持數量上限為「1000」。

使用範例

請參閱以下內容。
PF_Vision

3.3.23 PF_QueAutoRemove

用於在指定的零件座標佇列上設定自動刪除功能。

格式
PF_QueAutoRemove 零件ID, {True | False}

參數

- 零件ID
  指定零件ID（整數值1～32）。
- True | False
  False：停用自動刪除功能（預設）。
  True：啟用自動刪除功能。

傳回值
無

描述
用於在零件座標佇列上設定自動刪除功能。若啟用自動刪除功能，則在透過PF_QueGet從零件座標佇列中取得點資料時，從零件座標佇列自動刪除點資料。
若停用自動刪除功能，則不刪除點資料。若要刪除，則使用PF_QueRemove。
可在每個零件ID上設定啟用/停用自動刪除功能。

使用範例

```
PF_QueAutoRemove 1, True
```

3.3.24 PF_QueAutoRemove函數

用於傳回在零件座標佇列上設定的自動刪除功能的狀態。

格式
PF_QueAutoRemove (零件ID)

參數

- 零件ID
  指定零件ID（整數值1～32）。

傳回值
啟用零件座標佇列的自動刪除功能時，傳回「True」；停用時，傳回「False」。

描述
請參閱以下內容。
PF_QueAutoRemove

使用範例

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

3.3.25 PF_QueGet函數

用於從零件座標佇列傳回點資料。

格式
PF_QueGet (零件ID [, 指數])

參數

- 零件ID
  指定零件ID（整數值1～32)。
- 指數
  以整數值指定要讀取的佇列資料指數。
  開頭指數編號為0。選填。

傳回值
傳回點資料。

描述
PF_QueGet從零件座標佇列取得點資料。有省略指數時，則傳回佇列資料的開頭資料。有設定指數時，則傳回設定指數的點資料。
若透過PF_QueAutoRemove啟用佇列資料的自動刪除功能，則透過PF_QueGet刪除點資料。
停用時，不刪除點資料。若要刪除，則使用PF_QueRemove。

使用範例
請參閱以下內容。
PF_Robot

3.3.26 PF_QueLen函數

傳回零件座標佇列中已註冊零件座標佇列資料的數量。

格式
PF_QueLen (零件ID)

參數

- 零件ID
  指定零件ID（整數值1～32)。

傳回值
以整數值傳回有效的零件座標佇列資料之註冊數量。

描述
傳回零件座標佇列資料數量的目前註冊資料數量。
還可用作Wait命令的引數。

使用範例
請參閱以下內容。
PF_Robot

3.3.27 PF_QueList

顯示指定零件座標佇列的零件座標佇列資料列表（點資料)。

格式
PF_QueList 零件ID, [顯示數]

參數

- 零件ID
  指定零件ID（整數值1～32）。
- 顯示數
  以整數值指定要顯示的資料數量。
  選填。若省略，則顯示所有佇列資料。

描述
僅可在命令視窗中使用。

使用範例
命令視窗中的操作範例

```
> 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

顯示及重新設定指定零件ID及指數的零件姿態（整數）。

格式
PF_QuePartOrient 零件ID [,指數 [, 零件姿態]]

參數

- 零件ID
  指定零件ID（整數值1～32）。
- 指數
  以整數值指定零件座標佇列資料的指數。
  （開頭指數編號為「0」。）若在命令視窗中執行，則可省略。
- 零件姿態
  以整數值指定要重新設定的零件姿態。
  若在命令視窗中執行，則可省略。若省略，則顯示目前的使用者資料。
  PF_PARTORIENT_FRONT=1：正面零件姿態
  PF_PARTORIENT_BACK=2：背面零件姿態

描述
重新設定和顯示在零件座標佇列上註冊的零件姿態。

若透過PF_QueSort設定了以下Sort方法，則依該設定的Sort方法改變零件座標佇列的排列順序。
QUE_SORT_POS_PARTORIENT：零件姿態升序
QUE_SORT_INV_PARTORIENT：零件姿態降序

參照
PF_QuePartOrient函數

使用範例

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

3.3.29 PF_QuePartOrient函數

傳回指定零件ID及指數的零件姿態。

格式
PF_QuePartOrientt (零件ID [, 指數])

參數

- 零件ID
  指定零件ID（整數值1～32)。
- 指數
  以整數值指定要讀取的佇列資料指數。
  開頭指數編號為「0」。選填。

傳回值
傳回零件姿態（integer)。

描述
PF_QuePartOrient用於從零件座標佇列取得零件姿態。有省略指數時，則傳回零件座標佇列的開頭資料。有設定指數時，則傳回設定指數的零件姿態。

使用範例
請參閱以下內容。
PF_Robot

3.3.30 PF_QueRemove

從指定的零件座標佇列刪除零件座標佇列資料（點資料)。

格式
PF_QueRemove 零件ID [, 指數 | All]

參數

- 零件ID
  指定零件ID（整數值1～32)。
- 指數
  以整數值指定要刪除的零件座標佇列資料之指數。
  開頭指數編號為「0」。
  選填。若要刪除所有資料，請指定「All」。

傳回值
無

描述
從零件座標佇列刪除資料（點資料)。用於從零件座標佇列刪除已使用的資料。

使用範例
請參閱以下內容。
PF_Robot

3.3.31 PF_QueSort

用於在指定的零件座標佇列上設定和顯示Sort方法。

格式
PF_QueSort 零件ID [,Sort方法]

參數

- 零件ID
  指定零件ID（整數值1～32）。
- Sort方法
  以整數值（0～8）或如下所示的常數指定Sort方法。
  若在命令視窗中執行，則可省略。
  若省略，則顯示目前Sort方法。

預先定義的常數	值	內容
QUE_SORT_NONE	0	無排序（按註冊到零件座標佇列的順序）
QUE_SORT_POS_X	1	X坐標升序
QUE_SORT_INV_X	2	X坐標降序
QUE_SORT_POS_Y	3	Y坐標升序
QUE_SORT_INV_Y	4	Y坐標降序
QUE_SORT_POS_USER	5	使用者資料(實數)升序
QUE_SORT_INV_USER	6	使用者資料(實數)降序
QUE_SORT_POS_PARTORIENT	7	零件姿態升序
QUE_SORT_INV_PARTORIENT	8	零件姿態降序

傳回值
無

描述
在零件座標佇列上設定Sort方法。透過PF_QueAdd新增點資料時，則依設定的Sort方法在零件座標佇列上進行註冊。
使用PF_QueAdd前，需要執行PF_QueSort。即使在新增資料後使用本函數，也不會對資料進行重新排序。

使用範例

```
PF_QueSort 1, QUE_SORT_POS_X
```

3.3.32 PF_QueSort函數

用於傳回在指定的零件座標佇列上設定的Sort方法。

格式
PF_QueSort (零件ID)

參數

- 零件ID
  指定零件ID（整數值1～32)。

傳回值
用於以整數值傳回在零件座標佇列上設定的Sort方法。

值	內容
0	無排序（按註冊到零件座標佇列的順序)
1	X坐標升序
2	X坐標降序
3	Y坐標升序
4	Y坐標降序
5	使用者資料(實數)升序
6	使用者資料(實數)降序
7	零件姿態升序
8	零件姿態降序

使用範例

```
Integer quesort

quesort = PF_QueSort(1)
```

3.3.33 PF_QueUserData

顯示及重新設定指定零件ID及指數的使用者資料（實數)。

格式
PF_QueUserData 零件ID [, 指數] [, 使用者資料]

參數

- 零件ID
  指定零件ID（整數值1～32)。
- 指數
  以整數值指定零件座標佇列資料的指數。(開頭指數編號為「0」。)
  若在命令視窗中執行，則可省略。
- 用戶資料
  以實數值指定要重新設定的用戶資料。
  若在命令視窗中執行，則可省略。
  若省略，則顯示目前的使用者資料。

描述
重新設定和顯示在零件座標佇列上註冊的使用者資料。

若透過PF_QueSort設定了以下Sort方法，則依該設定的Sort方法改變零件座標佇列的排列順序。
QUE_SORT_POS_USER：使用者資料（實數）升序
QUE_SORT_INV_USER：使用者資料（實數）降序

參照
PF_QueUserData函數

使用範例

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

3.3.34 PF_QueUserData函數

傳回指定零件ID（及指數）的使用者資料。

格式
PF_QueUserData (零件ID [, 指數])

參數

- 零件ID
  指定零件ID（整數值1～32）。
- 指數
  以整數值指定要讀取的佇列資料指數。
  開頭指數編號為「0」。選填。

傳回值
傳回使用者資料（Real）。

描述
PF_QueUserData函數用於從PF佇列取得使用者資料（實數）。有省略指數時，則傳回零件座標佇列的開頭資料。有設定指數時，則傳回設定指數的使用者資料。

使用範例

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

3.3.35 PF_ReleaseFeeder

解除透過PF_AccessFeeder取得的鎖定（所有權）。
PF_AccessFeeder及PF_ReleaseFeeder用於多台機器人/1台送料器的系統中，透過鎖定或解除鎖定對送料器的存取來防止機器人之間的碰撞。在有2台機器人同時共享同一個送料器時，必須使用這些命令。

範例：
在任務1中執行PF_AccessFeeder 1。任務1繼續執行。
在任務2中執行PF_AccessFeeder 1時，任務2會暫停。
在任務1中執行PF_ReleaseFeeder 1時，任務2會恢復執行。

本命令可以編寫在動作命令的平行處理 (!...!) 中。

格式

格式1：PF_ReleaseFeeder 送料器編號/送料器名稱
格式2：動作命令 ! [Dn;] PF_ReleaseFeeder 送料器編號/送料器名稱 !

參數

- 送料器編號
  以運算式或數值（1～4的整數值）指定送料器編號。
- 送料器名稱
  指定送料器標籤（字串）。
- 動作命令
  Arc、Arc3、Go、Jump、Jump3、Jump3CP、Move、BGo、BMove、TGo、TMove中的任一命令
- Dn
  以%指定機器人移動路徑中開始平行處理的位置
  （請參閱「Epson RC+ 8.0 SPEL+ 語言參考 !...! 平行處理」）

傳回值

無

描述

只能在同一任務內解除鎖定。
無法從虛擬控制器及命令視窗執行。

使用範例

本用法範例描述了取得送料器上最後1個零件，並在移動到放置位置的路徑達到50%時，讓PF_Start恢復控制，開始視覺成像和送料器運作。

```
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
```

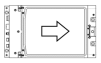
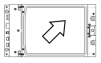
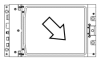
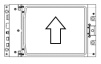
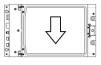
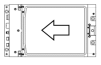
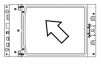
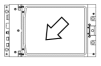
3.3.36 PF_Shift

執行位移動作。

格式
PF_Shift 零件ID, 方向 [, 振動時間]

參數

- 零件ID
  指定零件ID（整數值1～32）。
- 方向
  指定位移方向。

方向	值（PartFeeding.inc中定義）	動作
前	PF_SHIFT_FORWARD	
左前	PF_SHIFT_FORWARD_LEFT	
右前	PF_SHIFT_FORWARD_RIGHT	
左	PF_SHIFT_LEFT	
右	PF_SHIFT_RIGHT	
後	PF_SHIFT_BACKWARD	
左後	PF_SHIFT_BACKWARD_LEFT	
右後	PF_SHIFT_BACKWARD_RIGHT	

- 振動時間
  指定清除動作時間。（單位：毫秒）
  選填。省略時，會使用透過位移校準設定的值。
  請參閱以下內容。
  - **位移 - 測試和調整（簡易）**
  - **位移 - 測試和調整（詳細）**
    指定-1時，將進行與省略時相同的動作。

傳回值
無

描述
執行IF系列送料器的位移動作。

位移動作用於以下情況：

- 將零件移向放置位置側，以提高機器人地取放效率
- 使用自訂平台時，使零件落入溝槽或孔中以使其立起，或對齊方向

在以下條件下無法執行：

- 從使用者函數執行時，本命令中指定的送料器（以零件ID指定的送料器）正在Part Feeding程序（PF_Start命令）中使用（錯誤7733）
- 在回呼函數中執行時，指定了未在PF_Start命令中指定的零件ID（錯誤7733）
- 從虛擬控制器及命令視窗執行（錯誤3804）

本命令在內部處理中使用SyncLock。如需詳細資訊，請參閱以下內容。
Part Feeding程序所使用的功能

使用範例

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

3.3.37 PF_Start / PF_Start函數

啟動指定零件的Part Feeding程序。

格式
PF_Start 零件ID1 [, 零件ID2 [, 零件ID3 [, 零件ID4]]]
PF_Start (零件ID1 [, 零件ID2 [, 零件ID3 [, 零件ID4]]])

參數

- 零件ID
  指定零件ID（整數值1～32）。作為引數使用變數且不指定零件ID時，將數值設為0。

傳回值
無

描述
在啟動PF_Start前，請執行以下處理。
關於執行方法，請參閱使用範例。

- 選擇使用的機器人
- 設定機器人（Power、Speed、Accel等）
- 馬達On
- 若要輸出日誌，則執行PF_InitLog

執行PF_Start，將會生成新任務（任務編號32），並啟動Part Feeding程序。
讓呼叫源恢復控制，不等待Part Feeding程序結束。

在以下條件下會出現錯誤，並執行Status回呼函數。不會啟動Part Feeding程序。

條件	Status回呼函數的引數Status值
零件ID不存在 嘗試使用不同送料器的零件開始多種零件運作 重複設定零件ID	PF_STATUS_BAD_ID
零件的參數設定無效	PF_STATUS_BAD_PARAMETER
送料器校準未完成	PF_STATUS_CAL_NOT_COMPLETE

條件	Status回呼函數的引數Status值
零件已停用	PF_STATUS_PARTNOTENABLED
送料器已被使用	PF_STATUS_FEEDERINUSE_ERROR
發生系統錯誤	PF_STATUS_ERROR

- 多台送料器運作
  不同送料器的零件可以同時執行Part Feeding程序。例如，若零件1屬於送料器1，零件2屬於送料器2，則可以在執行PF_Start 1後執行PF_Start 2。
  PF_Start會為每個送料器建立任務。任務編號從「32」開始，每次執行PF_Start時編號減1。使用的任務編號取決於送料器編號。

送料器編號	任務編號
1	32
2	31
3	30
4	29

各回呼函數（PF_Robot、PF_Control、PF_Status、PF_Vision、PF_MobileCam）在與建立PF_Start的相同任務中執行。
T/VT系列控制器最多可同時控制2台送料器。若嘗試使用3台或更多送料器運作，將發生「錯誤7731：超過控制器類型的最大同時送料器數量」。

- 多種零件運作
  執行多種零件運作時，在引數中指定多個零件ID。最多可指定4個零件ID。此時，送料器會使用以PF_Start中第1個引數指定的零件ID（主動零件）的校準參數進行運作。若要切換主動零件，則使用PF_ActivePart命令。
  多種零件運作只能指定屬於同一送料器的零件。指定不同送料器的零件執行PF_Start時，會發生錯誤，並以PF_STATUS_BAD_ID呼叫PF_Status回呼函數。
  多種零件運作中，1個送料器最多可使用2台機器人。在使用3台或更多機器人的設定下執行PF_Start時，會發生錯誤，並以PF_STATUS_**呼叫PF_Status回呼函數。

- 其他注意事項
  執行Part Feeding程序時，無法對同一送料器重新執行Part Feeding程序。例如，在零件1和零件2均屬於送料器1時，若在執行PF_Start 1後執行PF_Start 2，則會發生錯誤，並以PF_STATUS_FEEDERINUSE_ERROR呼叫PF_Status回呼函數。已執行的PF_Start會繼續處理。
  請從一般任務執行PF_Start。從背景任務執行時會發生錯誤。
  無法從虛擬控制器及命令視窗執行。

使用範例

```
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

發出Part Feeding程序的終止請求。
若有正在執行的回呼函數，則會等待其終止。
之後，會執行PF_CycleStop回呼函數，並停止程序。

格式
PF_Stop 零件ID

參數
- 零件ID
  指定零件ID（整數值1～32）。

傳回值
無

描述
停止Part Feeding程序。
與PF_Abort命令不同，會等待執行中的回呼函數終止。
回呼函數終止後，會執行PF_CycleStop回呼函數。
若未啟動Part Feeding程序，則不執行任何動作。
在多種零件運作中使用PF_Stop時，設定PF_Start中設定的任一ID，即可停止Part Feeding程序。
無法在執行PF_Stop後立即執行PF_Start。在PF_CycleStop回呼函數完成前執行PF_Start，會發生PF_STATUS_FEEDERINUSE_ERROR。這是因為在當前Part Feeding程序完成前嘗試執行新的Part Feeding程序。
若要修正此情況，則追加以下程式碼：

```
PF_Stop 1     ' 在此範例中，零件1在使用任務32的送料器1上執行。
TaskWait 32   ' 等待處理結束。
PF_Start 2    ' 開始新零件。
```

無法從虛擬控制器及命令視窗執行。

使用範例

```
PF_Stop 1
```

3.4 Part Feeding回呼函數

回呼函數是指在特定條件下從PF_Start命令的運作程序中自動呼叫的SPEL函數。
新註冊第1個零件時，專案中會新增各回呼函數範本的程式檔案（PartFeeding.prg、PartFeeding.inc）。您可根據自身設備的規格，使用SPEL語言編寫必要的動作。

函數	描述/用法
PF_Robot	編寫機器人操作（拾取、放置）。
PF_Control	編寫料斗、自訂照明的操作。
PF_Status	編寫錯誤處理。
PF_MobileCam	編寫移動或退避到移動式攝影機成像位置的操作。
PF_Vision	編寫自訂的視覺處理（例如：依據多項視覺序列結果來判別零件）。

函數	描述/用法
PF_Feeder	編寫自訂的送料器處理（例如：在您製作的平台上處理零件）。
PF_CycleStop	編寫執行PF_Stop後的處理。

3.4.1 通用事項

回呼函數內的機器人編號為指定的機器人編號。
關於指定方法，請參閱以下內容。
一般設定
在Part Feeding程序的執行期間，作為一般任務執行回呼函數。使用的任務編號從「32」開始，每新增一個送料器則遞減。
請勿刪除PartFeeding.prg及PartFeeding.inc。另外，請勿將未使用的回呼函數刪除或註解掉。否則會發生建置錯誤。
在Part Feeding程序的執行期間，請勿從您的程式碼呼叫回呼函數。這可能會引起非預期的動作。若要進行測試，可以單獨執行回呼函數。

3.4.2 PF_Robot

編寫機器人從送料器取得零件並放置到指定位置的動作。
請務必編寫此函數的內容。

格式

```
Function PF_Robot(零件ID As Integer) As Integer
' 機器人運作
Fend
```

參數

- 零件ID
  輸入零件ID（整數值1～32）。
  多種零件運作時，輸入主動零件。

傳回值

請指定以下值來控制PF_Robot函數結束後的Part Feeding程序。（各常數在PartFeeding.inc中定義。）

- PF_CALLBACK_SUCCESS
  通常指定此值。當零件座標佇列中沒有剩餘資料時（多種零件運作時、主動零件沒有資料時），會繼續Part Feeding處理。
  佇列中有剩餘資料時（多種零件運作時、主動零件有資料時），會在不改變零件座標佇列內容的情況下重新啟動PF_Robot回呼函數。

- PF_CALLBACK_RESTART
  無論座標佇列中是否有剩餘資料，都會繼續Part Feeding程序，並重新生成座標佇列。這於每次拾取零件時進行成像以提高零件拾取精度的情況下有效。

- PF_CALLBACK_RESTART_ACTIVEPART
  無論座標佇列中是否有剩餘資料，都會繼續Part Feeding程序，生成僅限主動零件的座標佇列。非主動零件的座標佇列不會變更。這於省略非主動零件的視覺處理以加快速度的情況下有效。

- 使用者定義錯誤編號（8000 - 8999）
  當需要啟動PF_Status回呼函數進行錯誤處理時設定。設定的值會作為PF_Status回呼函數的引數被傳遞。

描述

本函數中編寫以下處理：

1. 從座標佇列取得零件座標（使用PF_QueGet函數）
2. 將機器人移動至送料器上的零件座標位置
3. 操作末端夾具等來抓住零件
4. 將機器人移動至零件放置座標
5. 操作末端夾具等來釋放零件
6. 刪除1個座標佇列（使用PF_QueRemove命令）

一般來說，會在迴圈處理中執行步驟1～6，處理座標佇列中的所有資料。座標佇列中包含前一次視覺動作檢測到的零件座標列表。座標為局部座標。多種零件運作時，會為每個零件ID生成座標佇列。
本函數啟動時選擇的機器人是在零件的Robot#中指定的機器人。使用多台機器人運作時，如有需要，可使用Robot陳述式切換機器人。
有關詳細資訊，請參閱以下手冊。
「Epson RC+ 8.0 SPEL+語言參考 - Robot」

⚠ 注意

- 建議從參數取得零件ID。
- 在多台機器人環境中變更所選機器人編號時，請注意不要變更為錯誤的機器人編號。

程式範例

以下程式是使用簡單的真空末端夾具處理所有零件的範例。
拾取零件時，使用真空感測器監視吸附狀態。
無法吸附時，最多重試3次。仍無法吸附時，傳回使用者錯誤碼8001作為回傳值。

```
' ** IO Label (Output) **
' O_Adsorb 吸附
' O_Desorb 釋放
'
' ** IO Label (Input) **
' I_Adsorbed 吸附感測器
'
' ** Point **
' PlacePos 零件的放置位置
'
' ** User Error **
' 8001 發生吸附超時
'
Function PF_Robot(partID As Integer) As Integer

    Byte retry

    ' 處理所有座標佇列，或進行迴圈處理直到有停止請求
    Do While PF_QueLen(partID) > 0 And PF_IsStopRequested(partID) = False

        ' 將機器人移動至拾取位置
        Jump PF_QueGet(partID)

        ' 真空ON & 確認吸附（重試3次）
        On O_Adsorb
        retry = 3
        Do While retry > 0
```

```
        Wait Sw(I_Adsorbed) = On, 0.5
        If TW = True Then
            ' 超時時，關閉吸附後重新開啟
            Off O_Adsorb
            Wait 0.1
            On O_Adsorb
        EndIf
    Loop
    If TW = True Then
        ' 重試後仍無法吸附，因此錯誤結束
        ' 錯誤處理在Status回呼函數中進行
        PF_Robot = 8001
        Exit Function
    EndIf

    ' 將機器人移動至放置位置
    Jump PlacePos

    ' 真空OFF & 釋放真空
    Off O_Adsorb
    On O_Desorb
    Wait 0.1
    Off O_Desorb

    ' 從座標佇列刪除1個資料
    PF_QueRemove partID

    Loop

    PF_Robot = PF_CALLBACK_SUCCESS

Fend
```

3.4.3 PF_Control

在系統需要料斗運作或照明操作時會呼叫PF_Control回呼函數。在此回呼函數中，編寫安裝在您設備上的以下機器的操作：

- 料斗
- 自訂照明

使用自訂照明時，請在Epson RC+ 8.0功能表 - [工具] - [料件送料] - [照明]中，選擇[自訂前側光]。

格式

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

參數

- 零件ID
  輸入零件ID（整數值1～32）。
  多種零件運作時，輸入主動零件。
- Control
  輸入操作的類型（整數值）。

操作	值（PartFeeding.inc中定義）
料斗操作 （送料器上沒有零件時）	PF_CONTROL_SUPPLY_FIRST
料斗操作 （送料器上有零件，且可以追加時）	PF_CONTROL_SUPPLY
自訂照明On	PF_CONTROL_LIGHT_ON
自訂照明Off	PF_CONTROL_LIGHT_OFF

傳回值
在正常情況下，請設定常數PF_CALLBACK_SUCCESS (PartFeeding.inc中定義)。
錯誤時，請設定使用者定義錯誤編號（8000 - 8999)。此值會作為PF_Status回呼函數的引數被傳遞。

描述
使用Select...Send陳述式編寫各設備的處理。

- 料斗操作 (送料器上沒有零件時)
  送料器上完全沒有零件的狀態。操作料斗，向送料器供應零件。
  若此時供應的零件數量為透過最佳放入零件數量校準取得的值（透過PF_Info命令取得)，對提高機器人的動作效率最為有效。

- 料斗操作 (送料器上有零件，且可以追加時)
  送料器上存在零件，但可以追加的狀態。在此時機追加零件，可增加視覺系統可檢測的零件，提高機器人的運作效率。
  此時供應的零件數量可設為如機器人取走的零件數量等。

- 自訂照明On
  視覺成像時開啟自訂照明的時機。操作自訂照明使其開啟。

- 自訂照明Off
  視覺成像結束後關閉自訂照明的時機。操作自訂照明使其關閉。
  請在您的程式碼中編寫以下處理。請確保這些處理在執行PF_Start前進行：
  -開始與料斗的通訊、調整設定等
  -開始與外部照明的通訊、調整設定（亮度）等

程式範例
以下程式用於控制料斗和外部照明。
將1台料斗（Gen.2）連接到IF-240，並使用PF_Hopper命令控制。在此範例中，最佳零件數量為60。在料斗校準畫面中，料斗已調整為在3秒的動作時間內供應60個零件。
假設外部照明已連接到標準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

        ' 料斗運作 送料器上沒有零件時
        Case PF_CONTROL_SUPPLY_FIRST
            PF_Hopper 1,partID, 3000
            Wait 3
```

```
        ' 料斗運作 在送料器上追加零件時
        Case PF_CONTROL_SUPPLY
            hopperOnTime = PF_QtyAdjHopperTime(partID, 60, 3000)
            PF_Hopper 1, partID, hopperOnTime
            Wait hopperOnTime / 1000

        ' 自訂照明ON
        Case PF_CONTROL_LIGHT_ON
            On O_FrontLight

        ' 自訂照明OFF
        Case PF_CONTROL_LIGHT_OFF
            Off O_FrontLight

    Send

    PF_Control = PF_CALLBACK_SUCCESS
Fend
```

3.4.4 PF_Status

編寫回呼函數的回傳值，以及對應系統狀態（如未供應零件等）的處理（主要是錯誤處理）。

格式

Function PF_Status(パーツID As Integer, Status As Integer) As Integer
'（錯誤處理）
Fend

參數

- 零件ID
  輸入零件ID（整數值1～32）。
  多種零件運作時，輸入主動零件。
- Status
  輸入狀態。
  回呼函數的回傳值或系統設定的值（請參閱「說明」）。

傳回值

若不指定回傳值，將視為指定了PF_EXIT，這會導致Part Feeding程序終止，因此請務必設定回傳值。

- PF_CONTINUE
  繼續Part Feeding程序。在一般情況下，請指定此值。
- PF_EXIT
  Part Feeding程序將會終止。指定此值，Part Feeding程序將會立即終止。若要指定此值，視需要編寫將裝置恢復到初始狀態的處理（將機器人原點復歸、關閉馬達等）。
- PF_RESTART
  重設Part Feeding程序並重新從視覺開始執行。
- PF_RESTART_ACTIVEPART
  重設Part Feeding程序並重新從視覺開始執行。僅重新生成主動零件的佇列。

描述

此函數會在其他回呼函數（PF_CycleStop函數及PF_Status函數除外）之後執行。

引數Status中設置前一個回呼函數的回傳值或在Part Feeding程序內部發生的錯誤。編寫與這些值相應的處理。

- PF_CALLBACK_SUCCESS
  回呼函數正常結束。通常不進行任何處理。

- PF_CALLBACK_RESTART
  PF_Robot回呼函數正常結束，回傳值為PF_CALLBACK_RESTART。通常不進行任何處理。

- PF_CALLBACK_RESTART_ACTIVEPART
  PF_Robot回呼函數正常結束，回傳值為PF_CALLBACK_RESTART_ACTIVEPART。通常不進行任何處理。

- PF_STATUS_NOPART
  料斗未供應零件的狀態。編寫向料斗供應零件的操作。

- PF_STATUS_TOOMANYPART
  從料斗過量供應零件，無法拾取零件的狀態。編寫透過操作員呼叫等方式，從料斗上移除零件的操作。若此狀態頻繁發生，請重新設定料斗。

- PF_STATUS_BAD_ID
  執行PF_Start命令時指定的零件ID不正確。請確認是否正確指定了已註冊的零件ID。多種零件運作時，嘗試在多台送料器上開始。請重新設定各零件。Part Feeding程序將會立即終止。

- PF_STATUS_BAD_PARAMETER
  執行PF_Start命令時指定的零件參數不正確。Part Feeding程序將會立即終止。

- PF_STATUS_CAL_NOT_COMPLETE
  執行PF_Start命令時指定的零件校準未完成。Part Feeding程序將會立即終止。
  關於執行送料器校準的方法，請參閱以下內容。
  校準

- PF_STATUS_WRONGPART
  無法檢測到送料器上的零件。請確認零件視覺序列能正確檢測到零件。或者，請確認是否混入了其他種類的零件或損壞的零件。
  當視覺多次嘗試檢測零件，零件Blob視覺序列檢測到拾取區域內有物體，但零件檢測視覺序列無法識別為正面零件或背面零件時，會出現此狀態值。

- PF_STATUS_PARTBLOB_ERROR
  用於檢測零件Blob的視覺序列或物件無效。Part Feeding程序將會立即終止。請確認用於零件的零件Blob序列和物件。

- PF_STATUS_PARTSEQ_ERROR
  用於檢測零件的視覺序列或物件無效。Part Feeding程序將會立即終止。請確認用於零件的零件序列和物件。

- PF_STATUS_ERROR
  執行PF_Start命令時發生錯誤（系統錯誤）。Part Feeding程序將會立即終止。請確認「視覺」中指定的視覺序列是否正確運作。請單獨執行回呼函數進行偵錯，以確認各回呼函數的動作是否正確。此外，如果嘗試讓2台以上的機器人共享1台送料器，將發生錯誤7730「每台送料器的機器人數量超過最大值。」。
  可透過Status回呼函數的回傳值指定運作程序的處理。
  若不指定回傳值，將視為指定了PF_EXIT，這會導致Part Feeding程序終止，因此請務必設定回傳值。

- PF_STATUS_FEEDERINUSE_ERROR
  對同一台送料器多次啟動了Part Feeding程序。Part Feeding程序將會立即終止。（已啟動的Part Feeding程序會繼續。）請檢查程式。

- PF_STATUS_PARTNOTENABLED
  零件已停用。
  請確認在Epson RC+ 8.0功能表 - [工具] - [料件送料] - [零件] - [零件**] - [一般]中，[啟用]核取方塊已勾選。

- PF_STATUS_PURGENOTENABLED
  清除功能已停用，但執行了PF_Purge函數。
  請確認在Epson RC+ 8.0功能表 - [工具] - [料件送料] - [零件] - [零件**] - [一般]中，[啟用]核取方塊已勾選。

注意

- 在PF_Status函數內，請編寫不會發生新錯誤的程式。否則可能會遞迴執行PF_Status函數，導致錯誤處理無法結束。
- 不建議在PF_Status函數內編寫送料器控制命令（PF_Center、PF_CenterByShift、PF_Flip、PF_Shift）。
- PF_Status的作用是指示系統如何在上一個回呼函數完成後繼續執行（也就是，在PF_Robot、PF_Vision、PF_Control、PF_MobileCam、PF_Feeder之後如何繼續執行）。這是透過將PF_Status的回傳值設為PF_CONTINUE、PF_RESTART、PF_RESTART_ACTIVEPART或PF_EXIT中的任一回傳值來實現的。詳細資訊請參考以下程式範例。

程式範例
以下程式用於編寫錯誤處理。
使用者錯誤是Robot回呼函數程式範例所示的吸附超時。
發生此錯誤時，將關閉機器人的馬達。

```
' ** User Error **
' 8001 發生吸附超時

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

    Select Status

        Case PF_CALLBACK_SUCCESS
            ' 成功（通常不進行任何處理）
        Case PF_CALLBACK_RESTART
            ' 從視覺重新開始
            PF_Status = PF_RESTART
            Exit Function

        Case PF_CALLBACK_RESTART_ACTIVEPART
            ' 從視覺重新開始 -
            ' 僅生成主動零件的座標佇列和取得影像
            PF_Status = PF_RESTART_ACTIVEPART
            Exit Function

        Case PF_STATUS_NOPART
            ' 料斗上沒有零件
            MsgBox "Hopper empty."

        Case PF_STATUS_TOOMANYPART
            ' 送料器上的零件過多
            MsgBox "Too many parts on Feeder."

        Case PF_STATUS_BAD_ID
            ' 指定的零件ID不存在
            MsgBox "Bad PartID."

        Case PF_STATUS_BAD_PARAMETER
            ' 零件參數不正確
            MsgBox "Bad parameter."

        Case PF_STATUS_CAL_NOT_COMPLETE
```

```
            ' 校準未完成
            MsgBox "Calibration incomplete."

        Case PF_STATUS_WRONGPART
            ' 送料器平台上的零件可能不正確
            MsgBox "Wrong Part."

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

        Case PF_STATUS_PARTBLOB_ERROR
            ' 零件Blob視覺序列錯誤
            MsgBox "Part Blob vision error."

        Case PF_STATUS_PARTSEQ_ERROR
            ' 零件檢測視覺序列錯誤
            MsgBox "Part Sequence vision error."

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

         Case PF_STATUS_PARTNOTENABLED
            ' 零件已停用
             MsgBox "Part is disabled."

        Case PF_STATUS_PURGENOTENABLED
            ' 清除已停用
            MsgBox "Purge is disabled."

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

    Send

    PF_Status = PF_CONTINUE

Fend
```

3.4.5 PF_MobileCam

編寫使用移動式攝影機時，將機器人移動到成像位置的處理，以及成像後退避的處理。無論是否使用移動式攝影機，都會執行本函數。

格式

Function PF_MobileCam(零件ID As Integer, Action As Integer) As Integer
' (將機器人移動至成像位置)
' (讓機器人退避)
Fend

參數

- 零件ID
  輸入零件ID（整數值1～32）。
  多種零件運作時，輸入主動零件。
- Action
  輸入移動類型。

移動類型	常數（PartFeeding.inc中定義）
將機器人移動至成像位置 （之後會進行視覺成像及送料器運作。）	PF_MOBILECAM_BEFORE
讓機器人退避 （之後會執行PF_Robot回呼函數。）	PF_MOBILECAM_AFTER

傳回值
在正常情況下，請設定常數PF_CALLBACK_SUCCESS（PartFeeding.inc中定義）。若要結束本函數並進行錯誤處理，請設定使用者定義錯誤編號（8000 - 8999）。此值會作為PF_Status回呼函數的引數被傳遞。

描述
在成像前及呼叫PF_Robot回呼函數前執行本函數。
編寫本函數後，請充分測試機器人是否能安全移動。

程式範例 以下程式是將移動式攝影機移動到成像位置的處理以及退避處理的範例。位置以點資料定義。標籤的成像位置為VisionPos，退避位置為HomePos。

```
' ** Point **
' VisionPos 移動式攝影機成像位置
' HomePos 移動式攝影機退避位置
'
Function PF_MobileCam(partID As Integer, Action As Integer) As Integer

    Select Action

        Case PF_MOBILECAM_BEFORE ' 將機器人移動到成像位置
            Jump VisionPos
        Case PF_MOBILECAM_AFTER ' 將機器人退避
            Jump HomePos
    Send

    MobileCam = PF_CALLBACK_SUCCESS
Fend
```

3.4.6 PF_Vision

要編寫成像處理（照明控制、執行視覺、SPEL程式處理）時使用。在僅執行視覺難以檢測到零件或判別姿態（如正反面）時使用。若要使用本功能，請在Epson RC+ 8.0功能表 - [工具] - [Part Feeding] - [Vision]中，選擇[使用PF_Vision callback進行視覺處理]。

格式
Function PF_Vision(零件ID As Integer, ByRef numBack As Integer) As Integer
'（視覺處理）
Fend

參數

- 零件ID
  輸入零件ID（整數值1～32）。
  多種零件運作時，輸入主動零件。
- numBack
  指派不可拾取的零件（如背面朝上等）數量。（以ByRef指定。）
  此值用於在Part Feeding程序中判斷是否需要翻轉。
  對於沒有正反面的零件，設定為「0」。

傳回值

若要繼續處理，請設定PF_CALLBACK_SUCCESS。
若要進行錯誤處理，請設定使用者定義錯誤編號（8000 - 8999）。
此值會作為PF_Status回呼函數的引數被傳遞。

描述

此函數用於結合多個視覺序列和外部照明控制來檢測零件座標的情況。您可編寫如下處理：

- 自訂照明控制
- 執行視覺（執行VGet陳述式）
- 零件座標佇列管理（初始化及點資料註冊）

程式範例

以下程式是使用2個視覺序列VSeq1（包含1個Geom物件）及VSeq2（包含1個Geom物件）檢測零件的範例。VSeq1與自訂照明1一起使用，可檢測到零件位置，但無法判定姿態。VSeq2與自訂照明2一起使用，僅檢測正面零件。
使用PF_QueAdd命令將檢測到的零件座標（設為本地編號1）放入座標佇列。此時，將透過以下方式獲得的值指派給Z座標。

教導視窗

若要啟用翻轉功能，請參閱以下內容。

一般設定

在[零件視覺序列]中指定VSeq2，以開啟自訂照明2。請參閱以下內容。

視覺系統

進行翻轉的校準。請參閱以下內容。

校準

```
' ** IO Label (Output) **
' O_FrontLight1 自訂照明1
' O_FrontLight2 自訂照明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座標
    RB_Z = -80.0

    ' 初始化座標佇列
    PF_QueRemove partID, All

    ' 開啟自訂照明1
    On O_FrontLight1

    ' 檢測零件（執行UsrVisionSeq1）
    VRun VSeq1
    VGet VSeq1.Geom01.NumberFound, numFound
```

```
    ' 關閉自訂照明1
    Off O_FrontLight1
    ' 開啟自訂照明2
    On O_FrontLight2

    numBack = 0
    For i = 1 To numFound
        ' 取得VSeq1.Geom01檢測到的零件XY像素座標
        ' 並設定VSeq2.Geom01的搜尋視窗以圍繞該零件
        ' 零件大小 = 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)

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

        If found = True Then
            ' 若能檢測到則為正面零件，註冊至座標佇列
            ' 本地編號為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
            ' 若無法檢測到則為背面零件
            numBack = numBack + 1

        EndIf

    Next

    ' 關閉外部照明2
    Off O_FrontLight2

    PF_Vision = PF_CALLBACK_SUCCESS

Fend
```

3.4.7 PF_Feeder

要使用送料器控制命令（PF_CenterByShift、PF_Center、PF_Flip、PF_Shift等各種命令）編寫送料器控制動作時使用。

格式

Function PF_Feeder(零件ID As Integer, 正面零件數量 As Integer, 背面零件數量 As Integer, 狀態 As Integer) As Integer
'（判定零件狀態）
'（送料器運作）
Fend

參數

- 零件ID
  輸入零件ID（整數值1～32）。
  多種零件運作時，輸入主動零件。

- 正面零件數量
  輸入視覺檢測到並放入零件座標佇列中的正面零件數量。
- 背面零件數
  輸入視覺檢測到的背面零件數量。
- 狀態
  表示系統檢測到的狀態或系統決定的建議動作。
  輸入下表中的其中1個值。

狀態/建議動作	常數（PartFeeding.inc中定義）	備註
可進行取放	PF_FEEDER_PICKOK	零件可進行取放。
可追加供應零件	PF_FEEDER_SUPPLY	零件供應方法為[在取放期間供應零件]時，可從料斗追加供應零件。 如需詳細資訊，請參閱以下內容。 **供應零件**
需要翻轉動作	PF_FEEDER_FLIP	建議進行翻轉動作（分散零件或改變姿態）。
位移至拾取區域	PF_FEEDER_SHIFT	建議進行位移動作（將零件移至拾取區域）。
集中及翻轉	PF_FEEDER_CENTER_FLIP	建議進行集中及翻轉動作。
料斗已空	PF_FEEDER_HOPPER_EMPTY	料斗內沒有零件。透過操作員呼叫等方式，向料斗供應零件。
向拾取區域反向位移	PF_FEEDER_SHIFT_BACKWARDS	建議進行反向位移動作（將滯留在平台角落的零件移回拾取區域）。
從料斗供料，並進行集中與分離	PF_FEEDER_SUPPLY_CENTER_FLIP	需要從料斗供應零件，並建議對零件進行集中及分離動作。
零件過多	PF_FEEDER_TOO_MANY	平台上的零件數量過多。透過操作員呼叫等方式，移除平台上的零件。
混入其他種類的零件	PF_FEEDER_WRONGPART	可能混入了錯誤種類的零件。透過操作員呼叫等方式進行恢復。
無法判定	PF_FEEDER_UNKNOWN	系統無法對設有溝槽或孔的平台判斷最佳振動方式。 請參閱以下內容。 **振動**

傳回值

根據在PF_Feeder結束後讓系統執行的程序，指定以下值。

常數（PartFeeding.inc中定義）	Part Feeding程序的運作
PF_CALLBACK_SUCCESS	在可拾取零件且判斷為不需要進一步進行送料器運作時指定。系統會呼叫PF_Robot回呼函數。注意：系統不會重新生成零件座標佇列。若在執行送料器運作後傳回此值，送料器上的零件座標與零件座標佇列的資料之間會產生差異。

常數（PartFeeding.inc中定義）	Part Feeding程序的運作
PF_CALLBACK_RESTART	在執行送料器運作後指定。系統會重新生成所有零件座標佇列，並再次呼叫PF_Feeder回呼函數。
PF_CALLBACK_RESTART_ACTIVEPART	在執行送料器運作後指定。系統重新生成僅限主動零件的零件座標佇列，並再次呼叫PF_Feeder回呼函數。

描述

此函數能讓使用者處理自訂的送料器運作。通常，由系統判斷並執行送料器振動處理。選擇[料件送料] - [零件] - [振動]中的「透過PF_Feeder回呼控制振動」時，系統會在指定時機執行PF_Feeder回呼。PF_Feeder回呼函數可用於解決難以應用Part Feeding程序的應用程式。
例如：Part Feeding程序無法為自訂平台（您進行了孔、溝槽等加工的特殊平台）判斷適當的處理。但是，您可在PF_Feeder回呼函數中編寫自訂的送料器處理。

可利用引數「正面零件數量」及「背面零件數量」，判斷可否轉移至拾取動作，或決定適當的振動動作。例如，當正面零件數量>0時，可判斷為可拾取零件且不需要進一步的送料器振動。
引數「狀態」表示建議的動作或設備狀態。這有助於決定送料器的動作類型或條件。

回傳值需為上表所示的其中一個值。回傳值表示讓系統執行的程序。

程式範例1：

以下範例展示如何利用引數「狀態」，根據不同條件執行送料器振動動作。

```
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
```

程式範例2：
以下範例展示如何利用引數「正面零件數量」及「背面零件數量」，執行送料器振動動作。

```
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

編寫執行PF_Stop命令後的處理。

格式
Function PF_CycleStop(零件ID As Integer)
'（停止時的處理）
Fend

參數

- 零件ID
  輸入零件ID（整數值1～32）。
  多種零件運作時，輸入主動零件。

傳回值
無

描述
在執行PF_Stop後，待執行中的回呼函數結束後執行本函數。在此編寫設備的處理等內容。編寫的內容包括將機器人原點復歸、關閉機器人馬達等處理。

程式範例

執行PF_Stop命令後，執行將機器人原點復歸並關閉馬達的處理。

```
Function PF_CycleStop(partID As Integer)

    Home
    Motor Off

Fend
```

3.5 Part Feeding日誌檔

3.5.1 概述

Part Feeding日誌檔是用來記錄以下運作在運作程序中的運作時間和結果的日誌檔。

- 視覺處理
- 送料器運作
- 回呼函數運作（Robot、Status）

Part Feeding日誌檔可用於以下用途：

- 估算拾取零件的週期時間。
- 估算每1次可拾取的零件數量以調整料斗放入數量。
- 依據各運作的處理時間估算耗時較長的運作並加以改善。

若要使用本功能，需要將PC連接到控制器。

3.5.2 啟用日誌功能

將PC連接到控制器。編寫時，應確保PF_InitLog的執行順序先於PF_Start。
如需詳細資訊，請參閱以下內容。
PF_InitLog

3.5.3 日誌檔的格式

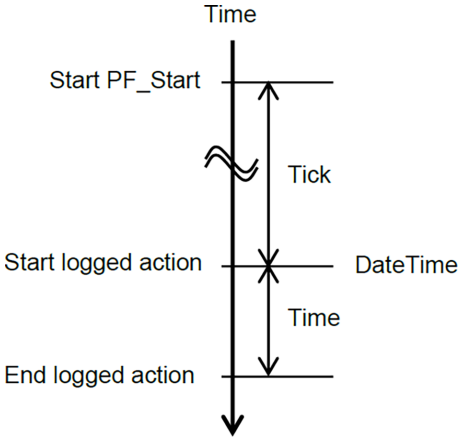
3.5.3.1 通用事項

檔案格式為CSV。檔案名稱以PF_InitLog的引數指定。
在1個日誌檔中，會按照時間順序記錄以下資料。「資料」欄位則視日誌類型而異。其他欄位則均通用。

列	列名稱	類型	內容
1	DateTime	使用字串	動作開始時間（yyyy/mm/dd hh:MM）
2	Tick	實數	自PF_Start開始後的經過時間 [秒]（s.sss）
3	Time	實數	處理時間 [秒]（s.sss）
4	Type	使用字串	動作類型
5	ID	整數	零件ID
6	Data1	因資料類型（Type）而異。	

列	列名稱	類型	內容
7	Data2		
8	Data3		
9	PartName	使用字串	零件名稱
10	RobotNo	整數	分配給零件的機器人編號
11	FeederNo	整數	分配給零件的送料器編號
12	Project	使用字串	Epson RC+專案名稱

DateTime時間與Tick、Time時間之間的關係如下圖所示。



3.5.3.2 視覺序列運作日誌

本日誌用於記錄Part Feeding程序在處理[零件視覺序列]所指定之視覺序列時的所需時間，以及檢測到的正面零件、背面零件之數量。
如需詳細資訊，請參閱以下內容。
視覺系統

列	列名稱	類型	內容
4	Type	使用字串	日誌類型（「UserVision」）
5	ID	整數	零件ID
6	NumFront	整數	檢測到的正面零件數量
7	NumBack	整數	檢測到的背面零件數量

3.5.3.3 系統視覺序列運作日誌

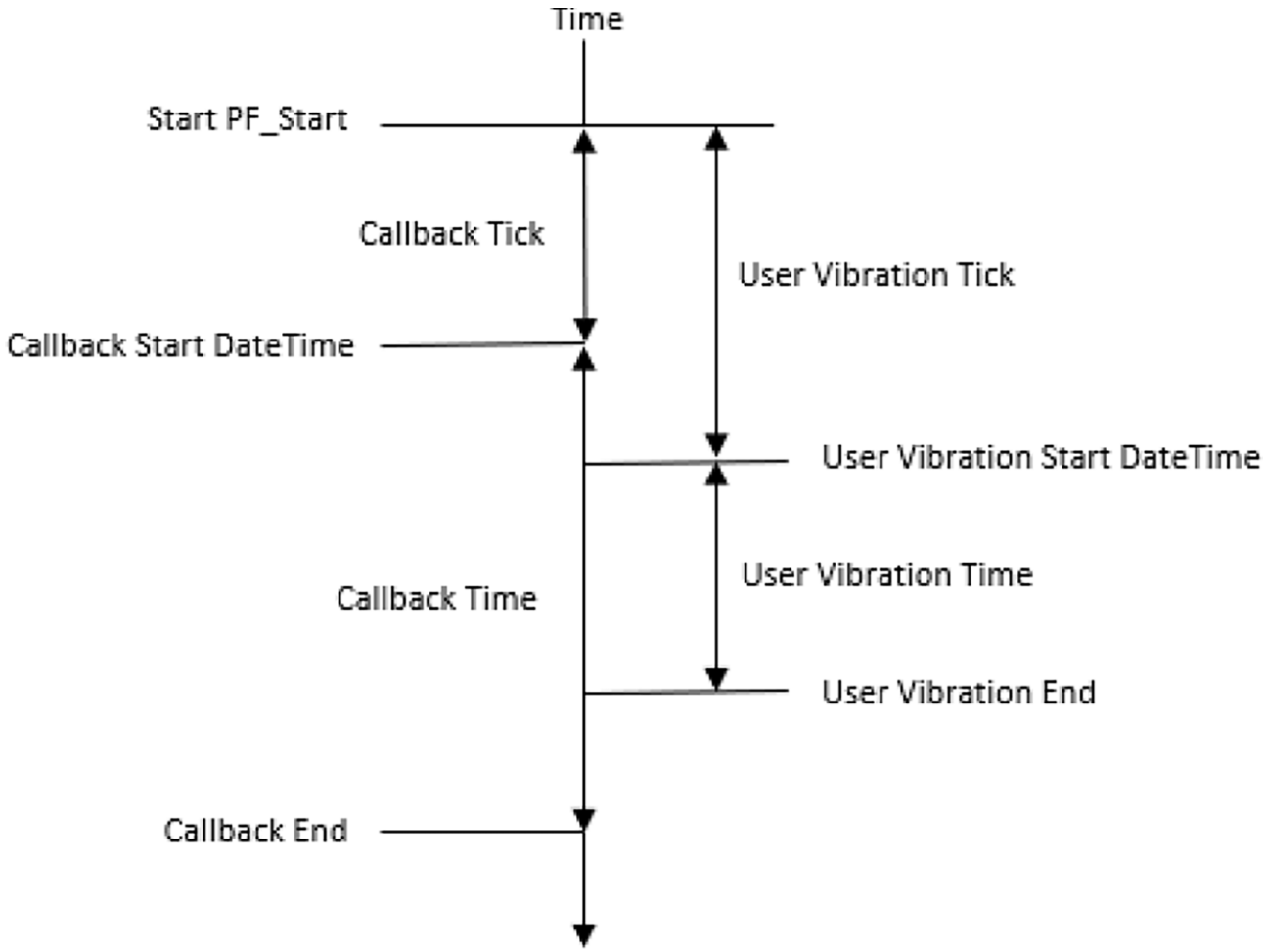
本日誌用於記錄Part Feeding程序為了檢測零件分布等而處理內部生成之視覺序列所需的時間。

列	列名稱	類型	內容
4	Type	使用字串	日誌類型（「SystemVision」）
5	ID	整數	零件ID

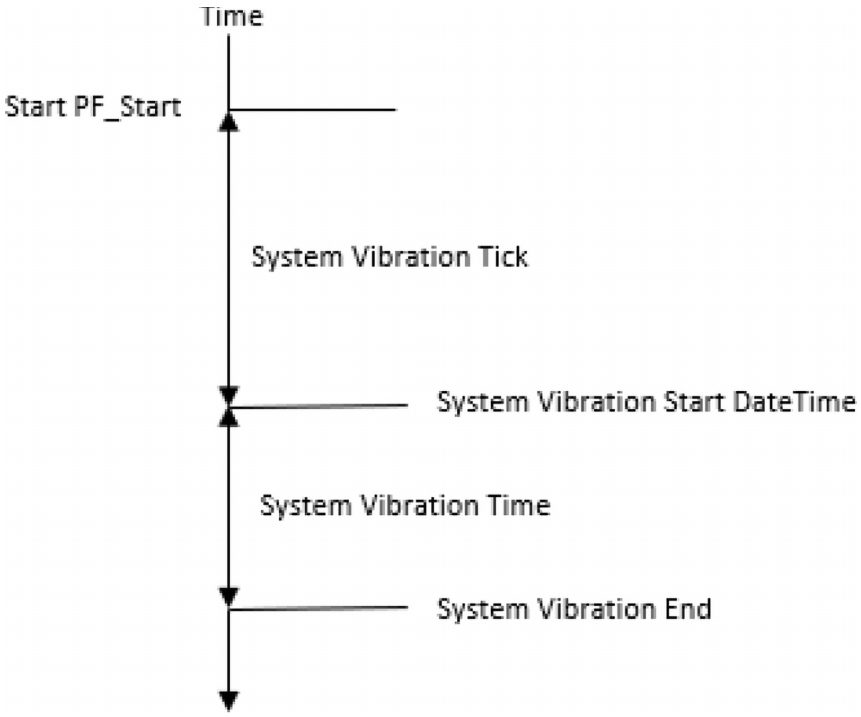
3.5.3.4 振動日誌

此日誌將記錄由系統或使用者執行的送料器振動之動作類型。

列	列名稱	類型	內容	
4	Type	使用字串	動作類型：	
			Separation	分離
			Centering	集中
			Shift	位移
			BackShift	反向位移
			Flip	翻轉
			CenterByShift	使用位移的集中
			Purge	清除
			QtyAdjHopperTime	QtyAdjHopperTime
5	ID	整數	零件ID	
6	Callback Name	使用字串	執行振動的回呼（或系統）名稱：	
			System	送料器
			Robot	視覺系統
			Control	MobileCam
			CycleStop	狀態窗格



時序圖：使用者操作所引起的振動，於回呼內執行



時序圖：由系統引起的振動

提示

若振動是由系統引起的，日誌檔的Data1列將會顯示「System」。在其他情況下，Data1將會顯示由使用者引起振動的回呼名稱。

3.5.3.5 PF_Robot回呼函數運作日誌

本日誌用於記錄透過PF_Robot回呼函數處理的零件數量。

列	列名稱	類型	內容
4	Type	使用字串	動作類型（「Robot」）
5	ID	整數	零件ID
6	Num	整數	零件處理數量（僅限主動零件） （呼叫前的座標佇列註冊數量 － 呼叫後的座標佇列註冊數量）

3.5.3.6 PF_MobileCam回呼函數運作日誌

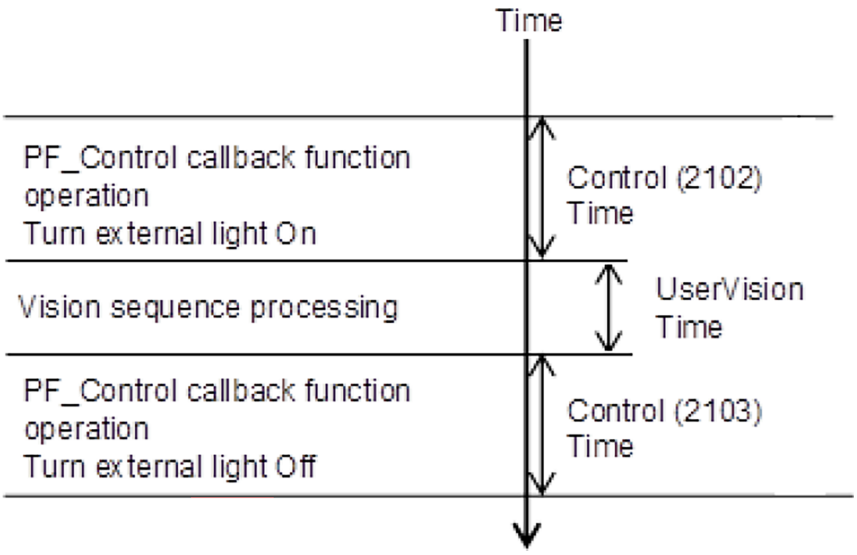
本日誌用於紀錄PF_MobileCam回呼函數的動作類型。

列	列名稱	類型	內容	
4	Type	使用字串	動作類型（「MobileCam」）	
5	ID	整數	零件ID	
6	Action	整數	將機器人移動至成像位置	2001
			讓機器人退避	2002

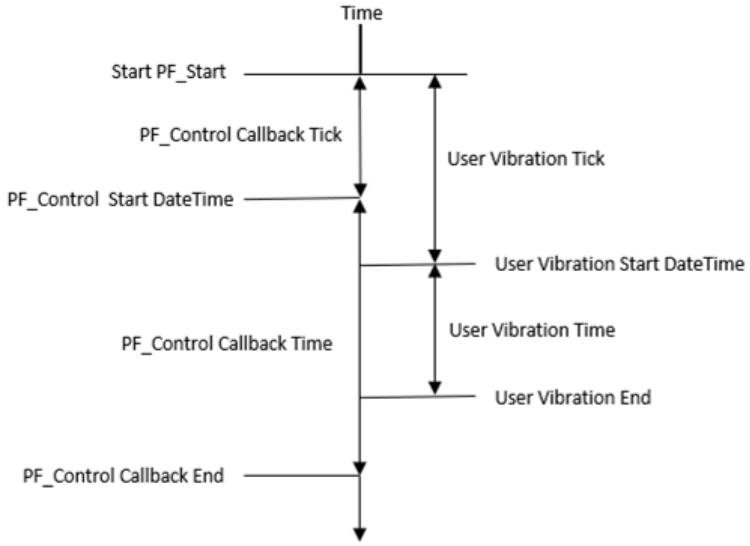
3.5.3.7 PF_Control回呼函數運作日誌

本日誌用於記錄PF_Control回呼函數的動作類型。

列	列名稱	類型	內容	
4	Type	使用字串	動作類型（「Control」）	
5	ID	整數	零件ID	
6	Action	整數	料斗操作（零件數0）	2100
			料斗操作（新增零件）	2101
			自訂照明On	2102
			自訂照明Off	2103



時序圖：使用前側光的視覺系統



時序圖：使用者操作所引起的振動，於PF_Control回呼內執行

3.5.3.8 PF_Status回呼函數運作日誌

本日誌用於記錄PF_Status回呼函數的狀態類型。

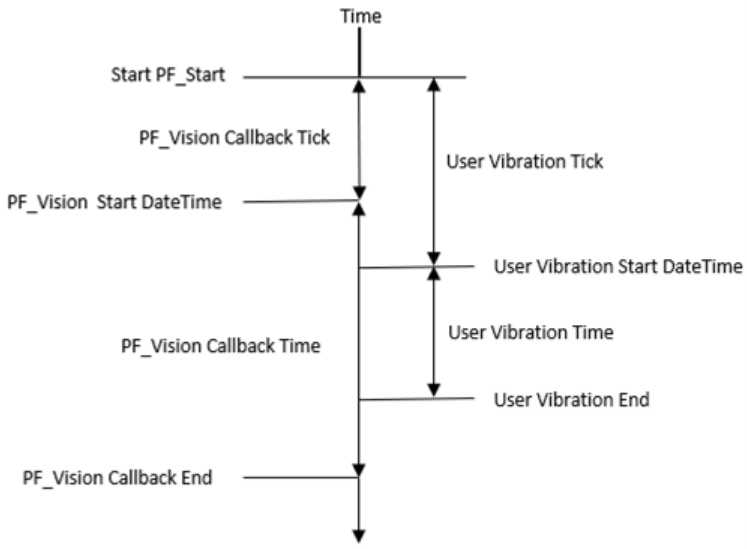
列	列名稱	類型	內容
4	Type	使用字串	動作類型（「Status」）
5	ID	整數	零件ID

列	列名稱	類型	內容	
6	Status	整數	發生的狀態或使用者錯誤	
			正常	0
			未供應零件	2200
			零件過多	2201
			ID錯誤	2202
			參數不正確	2203
			校準未完成	2204
			系統錯誤	2205
			存在無法檢測到的零件	2206
			零件Blob序列異常	2207
			零件視覺序列異常	2208
			正在使用送料器	2209
			未啟用零件	2210
			清除功能已停用	2211
			使用者錯誤	8000 - 8999

3.5.3.9 PF_Vision回呼函數運作日誌

本日誌用於記錄透過PF_Vision回呼函數檢測到的正面零件、背面零件之數量。

列	列名稱	類型	內容
4	Type	使用字串	動作類型（「VisionCallback」）
5	ID	整數	零件ID
6	NumFront	整數	正面零件檢測數量
7	NumBack	整數	背面零件檢測數量

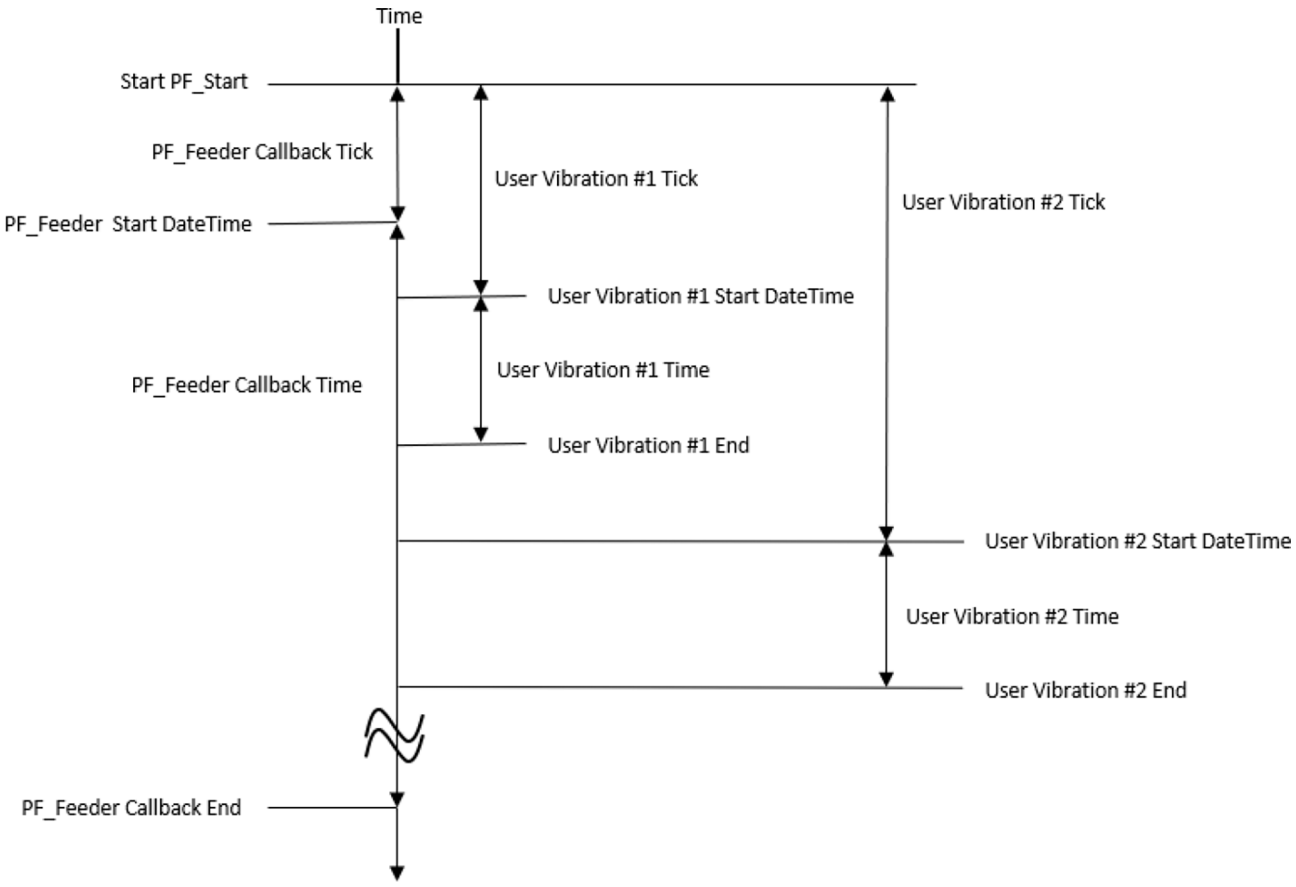


時序圖：使用者操作所引起的振動，於PF_Vision回呼內執行

3.5.3.10 PF_Feeder回呼函數運作日誌

本日誌用於記錄PF_Feeder回呼函數的狀態。

列	列名稱	類型	內容	
4	Type	使用字串	動作類型（「Feeder」）	
5	ID	整數	零件ID	
6	State	整數	建議動作：	
			無法判定	0
			可進行取放	1
			可追加供應零件	2
			翻轉	3
			位移	4
			集中和翻轉	5
			料斗已空	6
			向後方位移	7
			從料斗供料，並進行集中和翻轉	8
			零件過多	9
			混入其他種類的零件	10



時序圖：使用者操作所引起的振動，於PF_Feeder回呼內執行

3.5.3.11 PF_CycleStop回呼函數運作日誌

本日誌用於記錄PF_CycleStop回呼函數的狀態。

列	列名稱	類型	內容
4	Type	使用字串	日誌類型（「CycleStop」）
5	ID	整數	零件ID

3.5.4 日誌範例

以下為Part Feeding日誌的範例。

```
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選配件所使用的視覺序列

使用Part Feeding時，需要建立以下2種視覺序列：

- 送料器校準用視覺序列
- 零件檢測用視覺序列

關於視覺指南、視覺序列物件的詳細資訊，請參閱以下手冊。
「Vision Guide - 軟體篇」
關於視覺屬性的詳細資訊，請參閱以下手冊。
「Vision Guide - Properties and Results Reference」

3.6.1 視覺校準

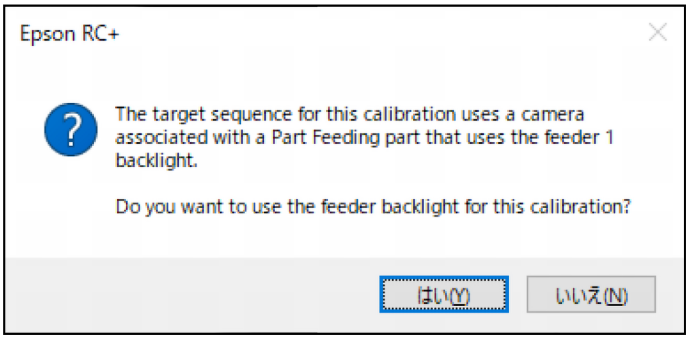
在送料器的平台面上進行視覺校準。
關於視覺校準方法，請參閱以下手冊。
「視覺指南8.0軟體手冊 - 視覺校準」

以下是視覺校準的重要屬性：

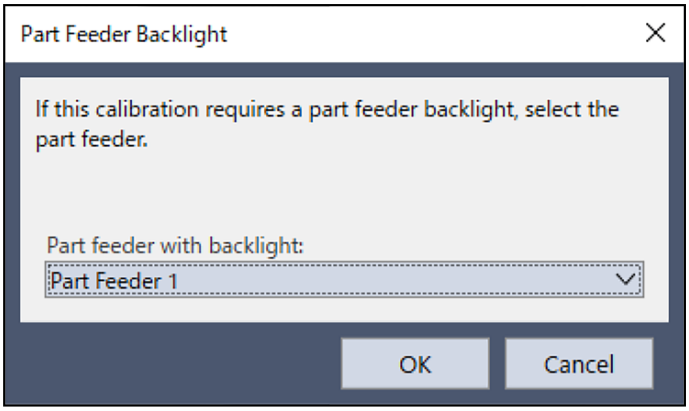
屬性	如何設定使用者定義的遠程輸出I/O
Camera	設定攝影機編號。
CameraOrientation	設定攝影機的固定方式。 朝下固定攝影機-Fixed downward 移動式攝影機-Mobile J4或Mobile J6
RobotLocal	指定機器人本地編號。
RobotNumber	指定機器人編號。 與以下指定的機器人編號保持一致。 一般設定
RobotTool	指定機器人工具編號。

至少有1個帶背光的送料器時：

> 提示
>
> 校準的TargetSequence攝影機與零件序列相同時，會顯示以下訊息。



校準的TargetSequence攝影機未在零件序列中使用時，會顯示以下對話方塊。



3.6.2 零件檢測視覺序列

零件檢測視覺序列用於檢測零件並取得零件座標。按照以下要點建立視覺序列。

> 提示
>
> 請將零件檢測視覺序列與送料器校準用視覺序列分開建立。

3.6.2.1 簡單的零件

下面說明建立視覺序列的範例，該視覺序列用於檢測不考慮姿態（正反面）或旋轉量，只需簡單抓住的零件。

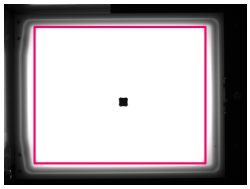
- 視覺序列
  設定以下屬性。請務必設定或確認以下內容。必要時，也可設定其他屬性。

屬性	如何設定使用者定義的遠程輸出I/O
Calibration	設定對送料器進行成像的攝影機之視覺校準。 指定與校準用視覺序列中指定時相同的視覺校準。 視覺校準需要已完成。
Camera	設定對送料器進行成像的攝影機之編號。 指定與校準用視覺序列中指定時相同的視覺校準。
ExposureTime	調整時，確保在送料器背光開啟的狀態下良好地對零件進行成像。同時確保使送料器周圍不顯得太暗。

- 視覺物件
  註冊可取得機器人座標的以下任一物件。
  - ArcFinder
  - ArcInspector
  - Blob
  - BoxFinder
  - ColorMatch
  - CornerFinder
  - Correlation
  - DefectFinder
  - Edge
  - Geometric
  - LineInspector
  - Point
  - Polar

設定物件的以下屬性。請務必設定或確認以下內容。根據物件類型，也可設定其他屬性。

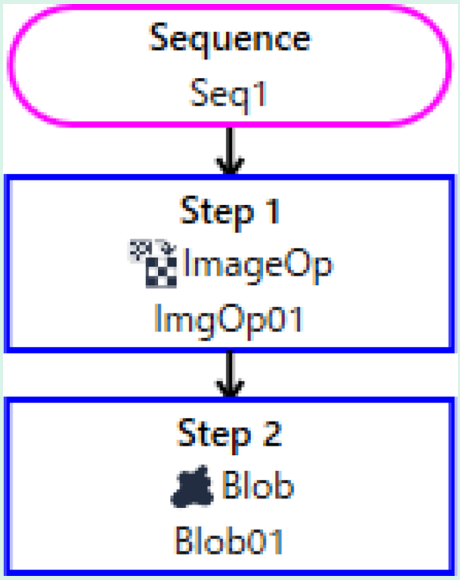
屬性	如何設定使用者定義的遠程輸出I/O
NumberToFind	設定為All。

屬性	如何設定使用者定義的遠程輸出I/O
SearchWindow	配合平台的內圓周。此外，設定時避免平台周邊暗部也包含進來。 

提示

也可併用其他物件。

例如：使用ImageOp物件進行二值化處理



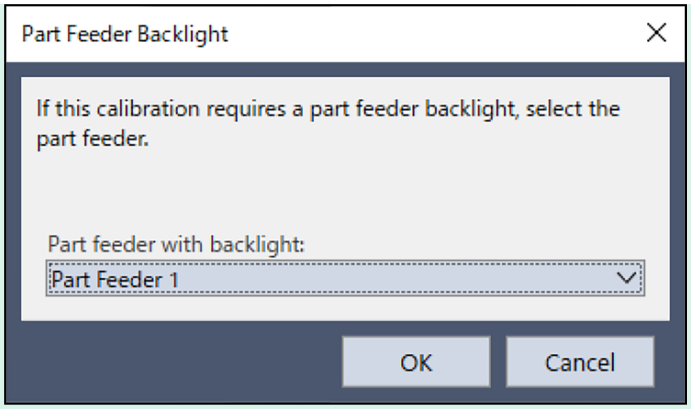
提示

建立新序列時，若有1個以上帶背光的零件送料器，會視需要顯示用於選擇帶背光送料器的對話方塊。

從Epson RC+ Vision Guide視窗執行視覺序列或物件時，背光會根據選擇自動開啟。

若帶背光的送料器是IF-80，拍攝影像後，背光會關閉，視頻會暫時靜止。透過靜止，即使背光關閉也能確認影像。點擊視頻區域可切換到即時視頻。

主要在建立了新專案但未追加供應零件時發生。



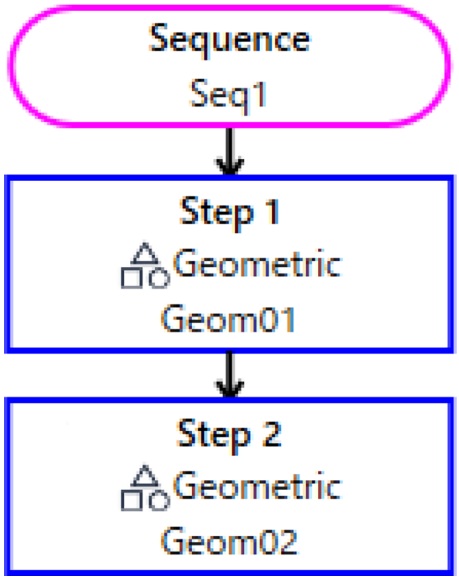
3.6.2.2 有正反面的零件

對於有正反面的零件，若只想取得其中1面，可如下設定。

- 視覺序列
  參閱以下內容，以相同方式建立。
  簡單的零件

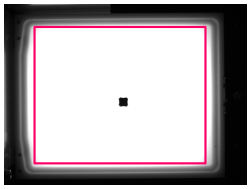
- 視覺物件

1. 註冊1個檢測零件正面的物件（例如：Geometric）。
2. 註冊1個檢測零件背面的物件（例如：Geometric）。
   例如：若要進行正反面判別，建立2個Geometric。（Geom01用於檢測正面，Geom02用於檢測背面）



Geometric的屬性設定範例如下。

屬性	如何設定使用者定義的遠程輸出I/O
Accept	設定為能確實檢測零件的值。
NumberToFind	設定為All。

屬性	如何設定使用者定義的遠程輸出I/O
SearchWindow	配合平台的內圓周。 
ModelWindow	配合零件外周，對零件進行教導。

提示

使用Geometric或Correlation物件時，將Timeout屬性設定為比執行序列所需時間稍長一點是有效的做法。未設定時，處理時間最多可能需要2000毫秒（預設值）。

範例：

在Geometric物件中設定NumberToFind = 9

存在大量零件

執行VRun

→ 視覺時間為75毫秒（例）

設定NumberToFind = All

執行VRun

→ 視覺時間約為2000毫秒（超時）

將Timeout屬性設為100毫秒

執行VRun

→ 視覺時間約為100毫秒（超時）

因此，在耗時的應用程式中，正確設定超時非常重要。

3.6.2.3 考慮機器人末端夾具空間時

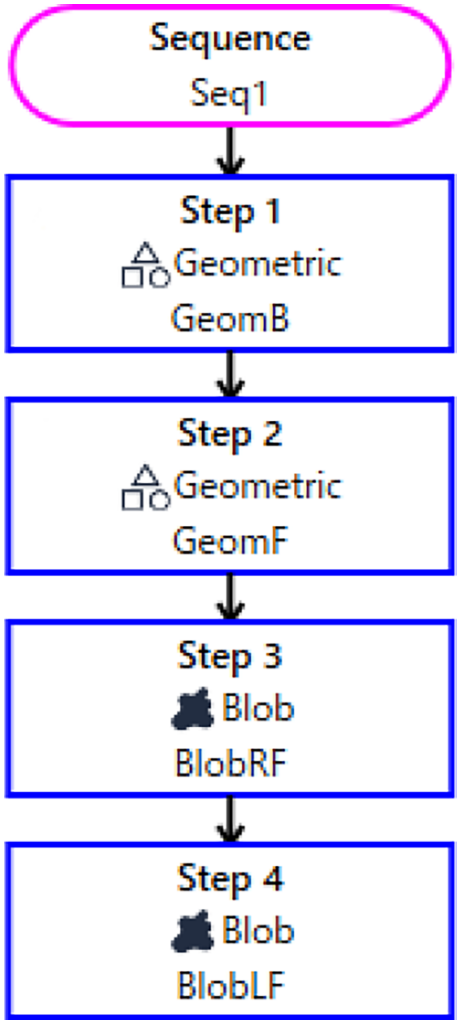
使用圖樣匹配（Geometric、Correlation）進行檢測時，即使零件部分重疊也可能被檢測到，這可能會影響機器人的拾取動作。為了排除這類零件，請按以下方式設定。

- 視覺序列
  參閱以下內容，以相同方式建立。
  簡單的零件

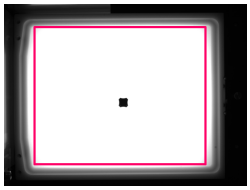
- 視覺物件

1. 註冊2個檢測零件的物件。（例如：Geometric）
   範例：建立GeomF（用於檢測正面）和GeomB（用於檢測背面）。

2. 新增視覺物件，用於確認零件周圍是否有機器人末端夾具進入的空間。
範例：建立Blob物件，用於確認空間。設定Blob物件的CheckClearanceFor屬性。



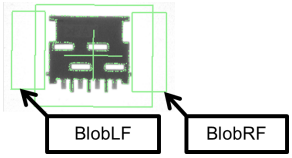
Geometric的屬性設定範例

屬性	設定範例
Accept	設定為能確實檢測零件的值。
NumberToFind	設定為「All」。
SearchWin	配合平台的內圓周。 
ModelWin	配合零件外周，對零件進行教導。
Name	GeomB：檢測背面零件的視覺物件 GeomF：檢測正面零件的視覺物件

Blob的屬性設定範例

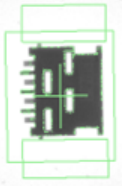
屬性	設定範例
CheckClearanceFor	GeomF 檢測正面零件的視覺物件
ClearanceCondition	NotFound
FailColor	LightGreen
Name	BlobRF：確認機器人末端夾具右爪進入的空間之物件 BlobLF：機器人末端夾具左爪用
PassColor	Red
SearchWin Type	RotatedRectangle
ThresholdHigh	192

此範例中，僅對正面零件確認機器人末端夾具夾爪進入的空間。

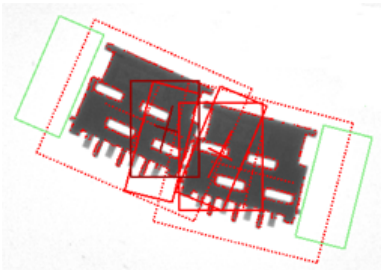


1. 將1個零件放置在送料器中央。
2. 執行視覺序列。

3. 將放在送料器中央的零件旋轉90°，再次執行視覺序列。



請確認Blob的SearchWindow會隨零件旋轉而旋轉。上圖為有機器人末端夾具夾爪的進入空間時的檢測結果。沒有空間時，檢測結果如下。

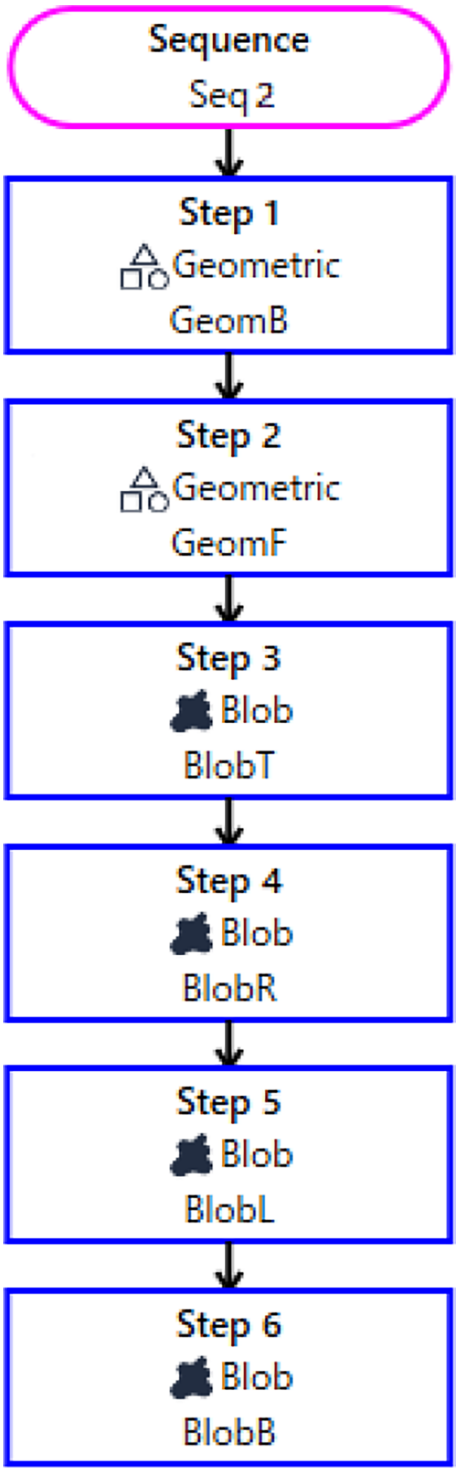


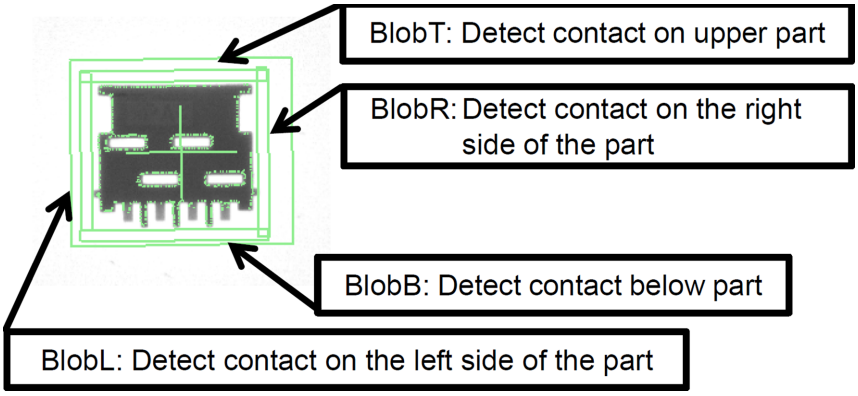
3.6.2.4 配置特殊視覺

當使用2個以上的視覺序列或多個照明來檢測零件時，可啟用視覺回呼函數，編寫所有照明控制和零件檢測。
請參閱以下內容。
PF_Vision

3.6.2.5 與相鄰零件接觸時不進行拾取的範例

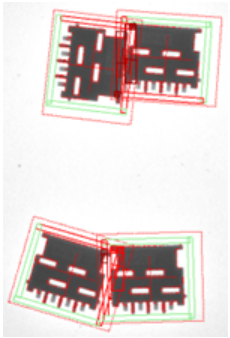
下面範例展示了當檢測到與相鄰零件接觸或間隙過小時不作為拾取對象。參閱以下內容，以相同方式建立視覺序列。
考慮機器人末端夾具空間時





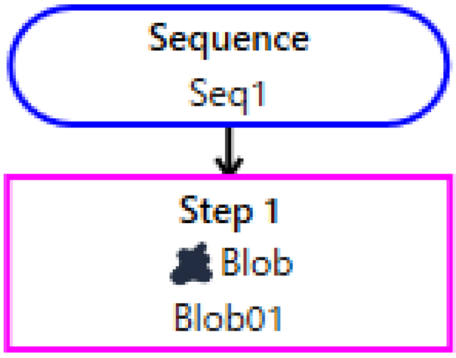
與相鄰零件無接觸時的檢測範例：

與相鄰零件有接觸時的檢測範例：



3.6.3 零件Blob視覺序列

建立送料器校準用視覺序列。



提示

請將送料器校準用視覺序列與零件檢測用視覺序列分開建立。

3.6.3.1 視覺序列

設定以下屬性。請務必設定或確認以下內容。其他屬性請使用預設值。

屬性	如何設定使用者定義的遠程輸出I/O
Calibration	參閱以下內容，並設定所建立的視覺校準。 **視覺校準** 視覺校準需要已完成（CalComplete結果為「True」）。
Camera	設定對送料器進行成像的攝影機之編號。
ExposureTime	調整時，確保在送料器背光開啟的狀態下適當地對零件進行成像。同時確保使平台周邊部分不顯得太暗。 對於容易透過光線的零件，需要謹慎設定此值。

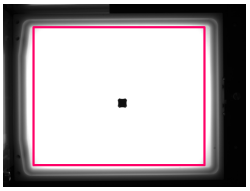
提示

若送料器上有不想檢測的暗部或污漬等，可使用「檢測遮罩」將其從檢測對象中排除。

如需詳細資訊，請參閱以下內容。
程式範例 8.3

3.6.3.2 視覺物件

設定以下屬性。請務必設定或確認。以下內容以外的屬性，請使用預設值。

屬性	如何設定使用者定義的遠程輸出I/O
MaxArea	保持預設值（攝影機的width × height）。
MinArea	將數值設定為零件面積的約0.9倍。 按照以下步驟估算零件面積： 1. 讓送料器的背光亮起 2. 在平台上放置數個不重疊的零件 3. 執行視覺序列 4. 以Blob結果中的Area之平均值作為零件面積
NumberToFind	設定為「All」。
SearchWin	盡量精確配合平台的內圓周。此外，設定時避免平台周邊暗部也包含進來。  若要旋轉搜尋視窗（在[Type]中選擇[Rotated Rectangle]），旋轉角度請設定在±45° 以內。

屬性	如何設定使用者定義的遠程輸出I/O
ThresholdColor	設為「Black」。
ThresholdHigh	設定時，確保可檢測到零件且不會檢測到平台外周等暗部。 對於容易透過光線的零件，需要謹慎設定此值。
ThresholdLow	設定為「0」。

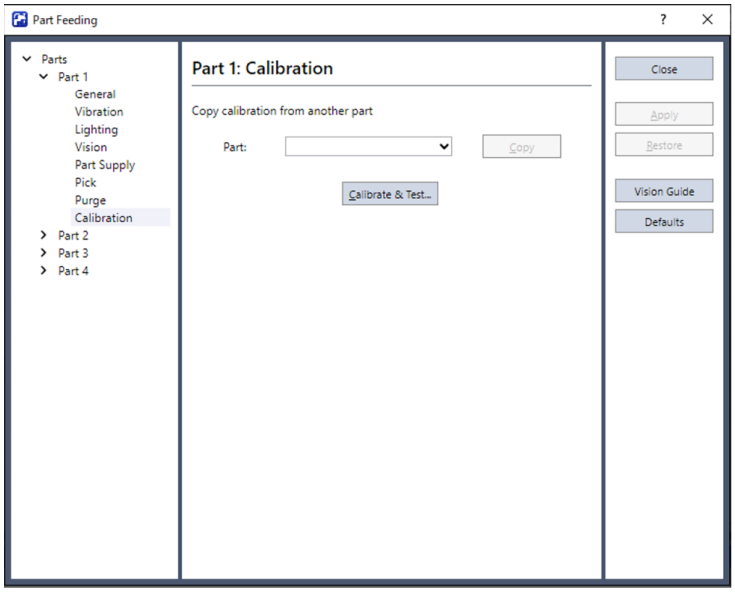
3.7 料斗的調整方法

下面說明料斗調整方法的範例。

3.7.1 料斗（Gen.1）的調整方法

假設料斗和送料器的安裝及連接已完成。

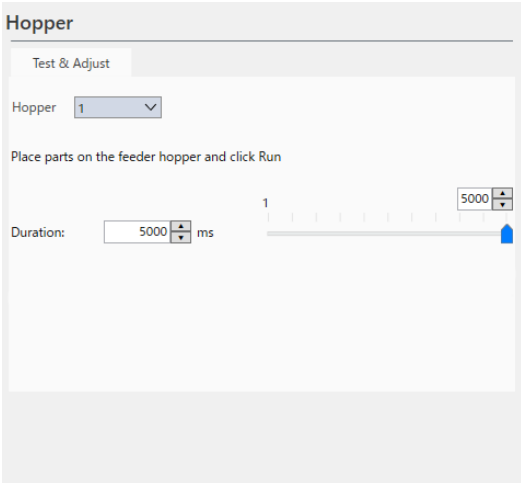
1. 在料件送料對話方塊中選擇[校準]，並點擊[校準&測試]。



2. 選擇[料斗]標籤。

3. 將最佳放入零件數量的5～10倍零件放入料斗。最佳放入零件數量可在[最佳放入零件數量]標籤確認。

4. 在測試和調整畫面中，只能調整振動時間。振動振幅則需要用料斗控制器手動調整。



5. 為了讓零件的放入速度保持一致，逐漸增大料斗控制器的振動振幅滑桿。調整振動時間時，確保在按下[執行]按鈕時供應適量的零件。

料斗（Gen.1）可使用PF_Hopper、PF_OutputOnOff命令進行控制。兩個命令的動作相同。

連接2台料斗（Gen.1）時，使用PF_Output命令。
如需命令範例，請參閱以下內容。
Part Feeding SPEL+命令參考手冊

測試步驟

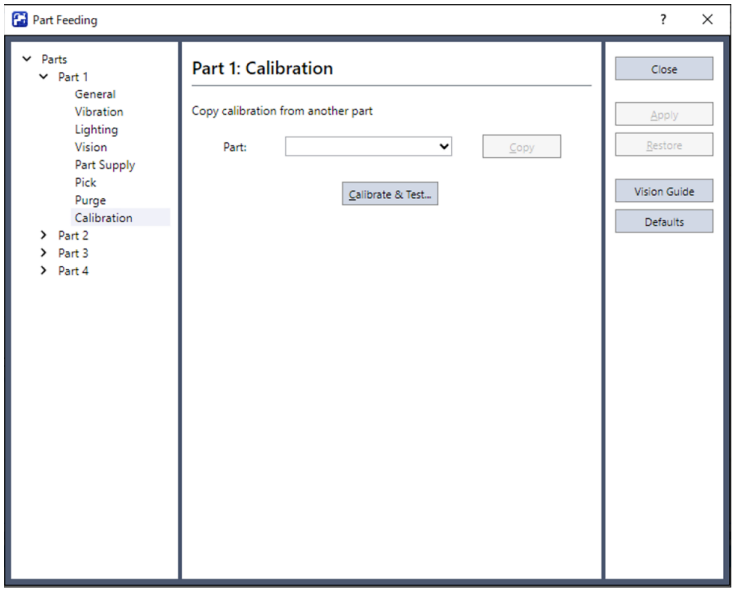
1. 將零件放入料斗。
2. 點擊[執行]按鈕。
3. 確認是否供應了最佳數量的零件。

根據需要調整參數、將零件放回料斗，並再次點擊[執行]按鈕。

3.7.2 料斗（Gen.2）的調整方法

假設料斗和送料器的安裝及連接已完成。

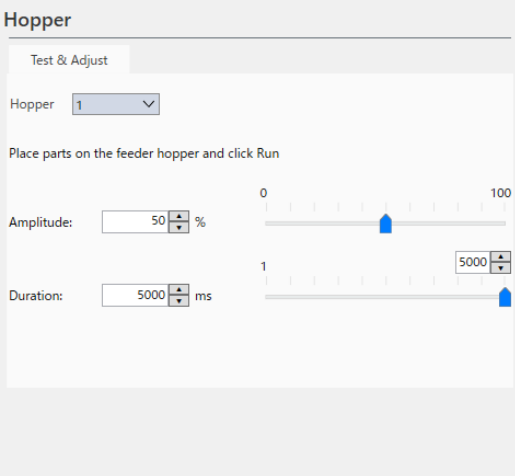
1. 在料件送料對話方塊中選擇[校準]，並點擊[校準&測試]。



2. 選擇[料斗]標籤。

3. 將最佳放入零件數量的5～10倍零件放入料斗。最佳放入零件數量可在[最佳放入零件數量]標籤確認。

4. 在測試和調整畫面中，可調整振動振幅和振動時間。



5. 調整振動振幅，讓料斗的零件放入速度保持一致。調整振動時間時，確保在按下[Run]按鈕時供應適量的零件。

料斗（Gen.2）可使用PF_Hopper命令進行控制。
如需命令範例，請參閱以下內容。
Part Feeding SPEL+命令參考手冊

若設定值超過以下表格中建議的振動振幅最大值，控制器將無法達到目標振幅。振動振幅會隨著料斗變空而變大。

提示

- 在調整振動振幅和振動時間之前執行校準。有關詳細資訊，請參閱以下手冊。
  「Epson RC+ 8.0選配件 Part Feeding 8.0 Hopper篇」
- 料斗（Gen.2）內建的智慧型感測器會調整振動振幅。料斗會檢測目前的零件數量，並自動調整振動振幅以穩定供應零件。

料斗S（1L/2L）的最大振動振幅

零件數量	振動振幅建議值
< 0.5 kg	100%以下
< 1 kg	75%以下
< 1.5 kg	50%以下
< 2 kg	25%以下

料斗M（3L/7L）的最大振動振幅

零件數量	振動振幅建議值
< 4 kg	100%以下
< 6 kg	75%以下
< 9 kg	50%以下

零件數量	振動振幅建議值
< 12 kg	25%以下

料斗L（14L）的最大振動振幅

零件數量	振動振幅建議值
< 5 kg	100%以下
< 10 kg	75%以下
< 15 kg	50%以下
< 20 kg	25%以下

測試步驟

1. 將零件放入料斗。
2. 點擊[執行]按鈕。
3. 確認是否供應了最佳數量的零件。

根據需要調整參數、將零件放回料斗，並再次點擊[執行]按鈕。

3.7.3 IF-80料斗的調整方法

IF-80為料斗一體型。下面說明料斗的調整範例。
執行料斗的校準。如需詳細資訊，請參閱以下內容。
料斗 - 測試和調整

修正程式。IF-80料斗可使用PF_Hopper、PF_OutputOnOff命令進行控制。兩個命令的動作相同。PF_Hopper命令適用於料斗（Gen.1）、料斗（Gen.2）和IF-80所有類型的料斗。

關於命令的詳細資訊，請參閱以下內容。
Part Feeding SPEL+命令參考手冊

3.8 操作RC+時發生的錯誤

訊息	原因、對策
未設定攝影機。 請在[設置] - [Vision Guide設定]中設定攝影機。	系統中未註冊攝影機。 請註冊攝影機。
虛擬控制器不支援Part Feeding。	Part Feeding選配件不支援虛擬控制器。 請連接至控制器。
未安裝或未啟用Part Feeding選配件。	未啟用Part Feeding選配件。 本選配件為付費選配件。 請向供應商購買選配件密鑰後進行設定。
未設定啟用的送料器。	系統中未註冊送料器。或未啟用送料器。 請註冊或啟用送料器。

訊息	原因、對策
未設定校準視覺序列。	未指定校準用視覺序列。 請在料件送料視窗的視覺系統頁面中，指定校準用視覺序列。
不可再增加零件。	1個專案可註冊的零件種類最多為32種。 請刪除未使用的零件，或直接覆蓋未使用零件的參數。
需要對不正確的視覺序列機器人攝影機進行校準。	在零件檢測用視覺序列或送料器校準用視覺序列中尚未設定視覺校準。 請在各視覺序列的Calibration屬性中設定有效的視覺校準。
校準錯誤：零件過多。	送料器校準運作中，零件放入數量過大。 或是因視覺設定不適當而誤檢測零件。請依照畫面顯示的內容放入正確數量的零件。或重新設定視覺系統。
校準錯誤：零件過少。	送料器校準運作時的零件放入量過少。或是因視覺系統的設定不當而無法檢測到零件。請依照畫面顯示的內容放入正確數量的零件。或重新設定視覺系統。
無法檢測到零件。	在送料器校準運作時未放入零件。或是因視覺系統的設定不當而無法檢測到零件。 請依照畫面顯示的內容放入正確數量的零件。或重新設定視覺系統。
送料器的通訊埠開啟/關閉失敗。請檢查給料器的連接情況。	1) 送料器的連接中斷。請確認送料器與控制器之間的乙太網路連接是否正常（檢查電纜是否斷線，集線器是否故障或未供電）。請檢查給料器的電源。 2) 送料器的通訊設定（送料器型號、IP位址、子網路遮罩、連接埠）錯誤。請重新進行設定。
無法連接到送料器。	（同上）
指定了未定義的函數。	請關閉[料件送料]視窗，重新建立專案。
無法透過目前使用的設定值連接到送料器。	送料器的通訊設定（送料器的型號、IP位址、子網路遮罩、連接埠）錯誤。請重新進行設定。
若要使用料件送料，以此版本的Epson RC+ 8.0而言，則需要配備版本為**以上的控制器韌體。	控制器的版本與RC+的版本不一致。請更新控制器的韌體。
料件送料的零件 ** 已設定為使用零件送料器**（型號**），但控制器的送料器** 卻是型號 **。若零件送料器的型號已變更，則需要重新校準。是否繼續？	由於系統配置中的送料器型號已變更，與零件的送料器設定產生不一致。 請在系統配置中將送料器型號恢復原狀，或針對目標零件進行重新校準。
零件送料器的韌體版本（**）不支援此版本的Epson RC+ 8.0。型號 ** 的零件送料器韌體版本需為 ** 以上。	1) 料件送料功能不支援此送料器的韌體版本。請更新送料器的韌體。 2) 已連接其他廠商製造的送料器。請使用從本公司購買的送料器。

訊息	原因、對策
料件送料的零件中使用了韌體版本為**的緊湊型視覺。若要使用料件送料功能，緊湊型視覺的韌體版本需為 ** 以上。	1) 若要在料件送料中使用CV，CV的韌體版本需為3.0.0.0以上。請更新CV的韌體。 2) 若要在料件送料中使用CV，請使用CV2-SA/HA、CV2-HB/SB/LB。不支援CV1和CV2-S/H/L。

3.9 應用程式範例

下面說明用於因應各類情況的應用程式範例。

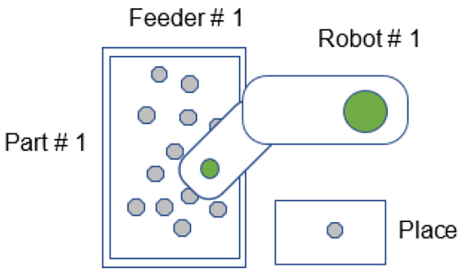
3.9.1 每個送料器對應1台機器人&每個送料器對應1種零件

3.9.1.1 程式範例 1.1

範例類型：
在動作中使用PF_Robot回呼

配置

- 機器人數量：1
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：1
  攝影機的朝向：送料器#1上的朝下固定的攝影機



描述
零件從送料器移動到箱子。當所有零件移動完畢時，Control回呼會請求零件。由操作員或料斗向送料器補充零件後，繼續移動循環。執行PF_Stop命令時，目前的循環停止，應用程式結束。（但在以下範例程式碼中未使用PF_Stop命令。）

範例程式碼

Main.prg

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

PartFeeding.prg

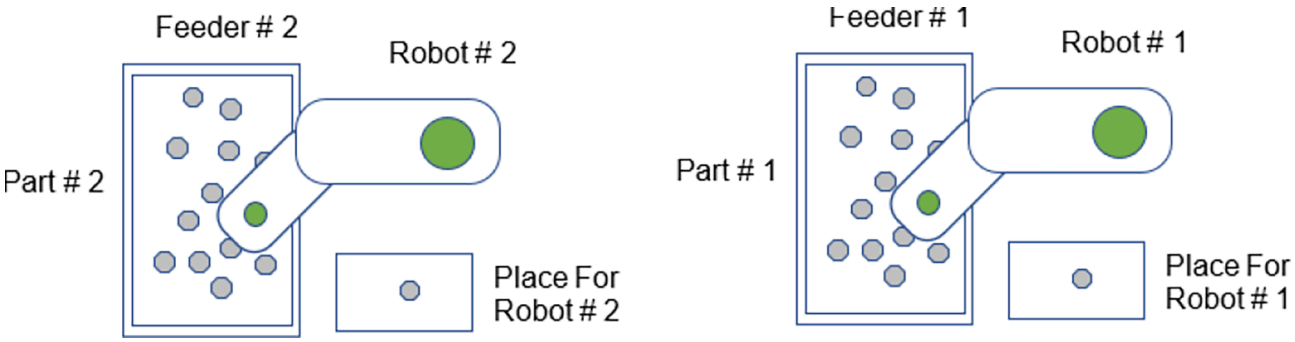
```
Function PF_Robot(PartID As Integer) As Integer
    Do While PF_QueLen(PartID) > 0
        P0 = PF_QueGet(PartID)
        Jump P0
        On Gripper; Wait 0.2
        Jump Place
        Off Gripper; Wait 0.2
        PF_QueRemove PartID
        If PF_IsStopRequested(PartID) = True Then
          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

3.9.1.2 程式範例 1.2

範例類型：
多重機器人系統 - 每個送料器對應1台機器人&每個送料器對應1種零件

配置

- 機器人數量：2
  將機器人1連接至控制裝置。機器人2則連接到驅動裝置。
- 送料器數量：2
- 送料器上的零件種類數：1
- 放置位置數量：每台機器人1個
- 攝影機的朝向：每個送料器配置朝下固定的攝影機



描述
各機器人分別進行零件取放。每台機器人都有專用的送料器和放置位置。此外，機器人1和機器人2的動作範圍不重疊。每台機器人在各自的點檔案中都有教導的「Park」、「Pick」、「Place」點位。

範例程式碼
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

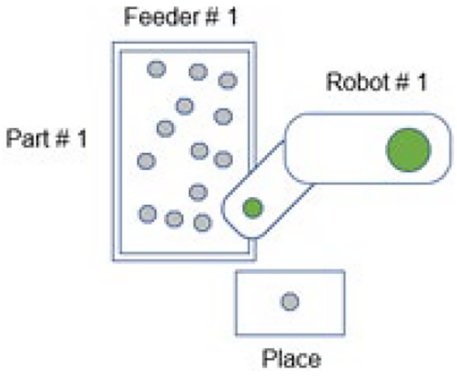
```
Function PF_Robot(PartID As Integer) As Integer
    Integer gripperOutput

    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 程式範例 1.3

範例類型：
機器人動作的視覺與振動平行處理

配置

- 機器人數量：1
- 送料器數量：1
- 零件種類數：1
- 放置位置數量：1
- 攝影機的朝向：朝下固定的攝影機



描述
機器人從零件送料器取出零件#1，並放置到固定治具中。此動作持續進行，直到所有「正面」零件從送料器移除為

止。放置最後一個零件時（送料器運作的90%），送料器開始振動，並視需要由料斗向送料器補充零件。
此範例說明為了優化處理能力，送料器運作如何與機器人動作同時進行。

範例程式碼

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
```

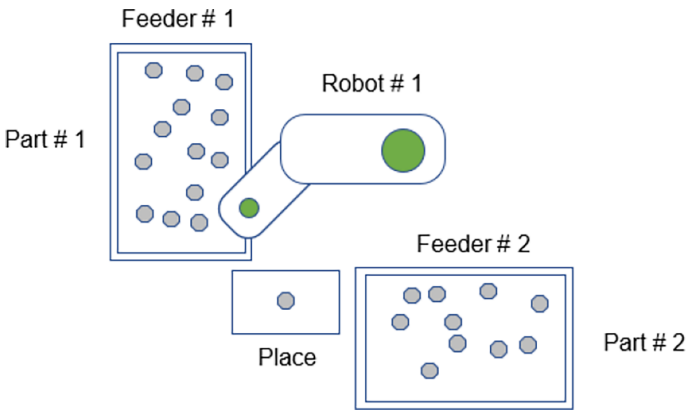
3.9.1.4 程式範例 1.4

範例類型：
每個送料器對應機器人1台、送料器2個、零件1種類的配置

配置

- 機器人數量：1

- 送料器數量：2
- 各送料器上的零件種類數：1
- 放置位置數量：1
- 攝影機的朝向：各送料器配置朝下固定的攝影機



描述

機器人從送料器#1取出零件#1，並放置到箱子中。此動作持續進行，直到所有「正面」零件從送料器#1移除為止。之後，機器人從送料器#2取出「正面」零件，並移動到箱子。攝影機和送料器同時運作。

範例程式碼

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

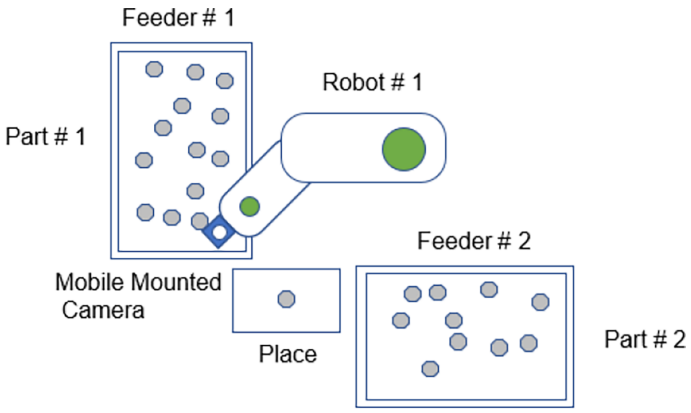
```
Function PF_Robot(PartID As Integer) As Integer

    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 程式範例 1.5

範例類型：
每個送料器對應機器人1台、送料器2個、零件1種類 - 移動式安裝攝影機

配置

- 機器人數量：1
- 送料器數量：2
- 送料器上的零件種類數：1
- 放置位置數量：1
- 攝影機的朝向：各送料器使用機器人的移動式安裝攝影機。
  （此範例不使用朝下固定的攝影機。）



描述

機器人將移動式安裝攝影機移至送料器#1上方並拍攝影像。機器人從送料器#1取出各零件，並放置到箱子中。此動作持續進行，直到所有「正面」零件從送料器移除為止。機器人將移動式安裝攝影機移至送料器#2上方並拍攝影像。機器人從送料器#2取出正確方向的零件，並放置到箱子中。

> **提示**
>
> 使用配備背光的多個送料器的移動式攝影機時，請為每個送料器所綁定的每個零件使用不同的視覺序列。

範例程式碼

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
```

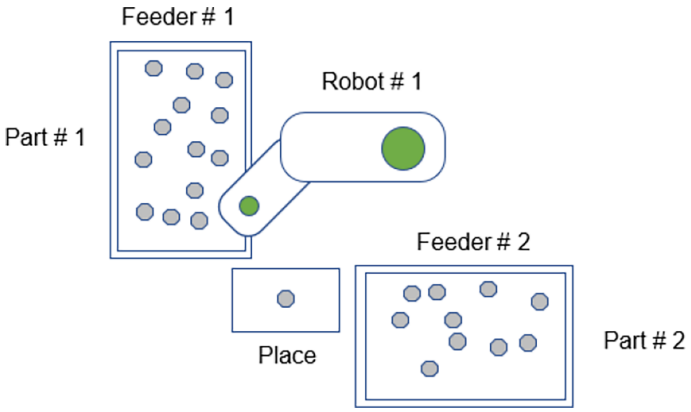
```
            MemOff mobileCamBeforeMembit
            MemOn mobileCamAfterMembit
            Wait MemSw(mobileCamInPosMembit) = Off
    Send

    PF_MobileCam = PF_CALLBACK_SUCCESS
Fend
```

3.9.1.6 程式範例 1.6

範例類型：
指定拾取零件的數量

配置

- 機器人數量：1
- 送料器數量：2
- 各送料器上的零件種類數：1
- 放置位置數量：1
- 攝影機的朝向：每個送料器配置朝下固定的攝影機



描述
機器人從送料器#1取出4個零件#1，並分別放置。之後，機器人從送料器#2取出5個零件#2，並放置。攝影機和送料器同時運作。
此範例說明機器人如何從各送料器取出特定數量的零件。在範例程式碼中，透過將numToPick參數設為「ALL_AVAILABLE」，可取出所有零件。
在取出所需數量的零件後，如果任一送料器上仍有「正面」零件，送料器不會振動。

範例程式碼
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
```

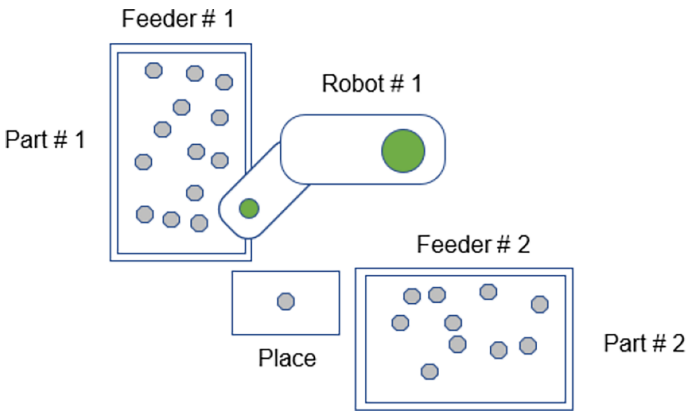
3.9.1.7 程式範例 1.7

範例類型：

確認機器人與零件使用的機器人的編號為一致

配置

- 送料器數量：1以上
- 各送料器上的零件種類數：1
- 攝影機的朝向：每個送料器配置朝下固定的攝影機



描述

使用多個送料器時，通常會在PF_Robot回呼外的別的任務中描述機器人動作，而非在PF_Robot回呼內部。也就是說，PF_Robot回呼只負責將零件位置（點位）放入零件座標佇列中。

機器人動作任務中，會在零件座標佇列中使用這些點位。以這種方式配置程式碼時，請在機器人動作任務中確認目前的機器人編號與零件選擇的機器人編號是否一致。

此確認可防止機器人移動到錯誤的點位或發生碰撞。

範例程式碼

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 程式範例 1.8

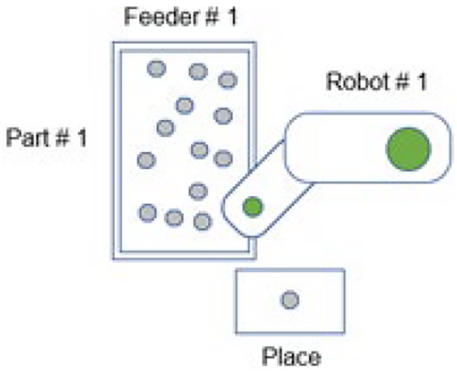
範例類型：

拾取後取得新影像並重新載入佇列

配置

- 機器人數量：1
- 送料器數量：1
- 零件種類數：1

- 放置位置數量：1
- 攝影機的朝向：朝下固定的攝影機



描述

機器人從零件送料器取出零件#1，並放置到固定治具中。取放操作後，取得新影像並重新載入佇列。
此範例中，關注的是在拾取過程中周圍零件可能移動的問題。PF_Robot回呼函數的回傳值
「PF_CALLBACK_RESTART」會強制對所有零件重新執行視覺，並重新載入所有零件佇列。
雖然這種方法效率不高，但在對每個循環取得影像，可提高性能精度這種特定情況下，
「PF_CALLBACK_RESTART」很有用。視覺和送料器振動與機器人動作同時進行。

範例程式碼

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
```

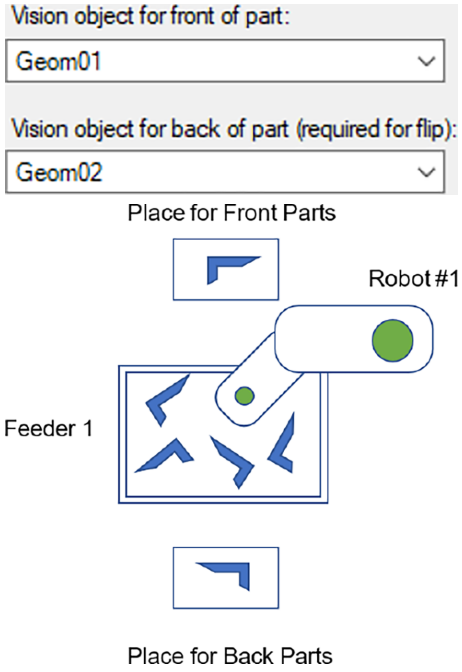
3.9.1.9 程式範例 1.9

範例類型：
按零件正反面分類

配置

- 機器人數量：1
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：2（1個用於「正面」朝向，另1個用於「背面」朝向）
- 攝影機的朝向：送料器#1上的朝下固定的攝影機
- 零件1一般：

- 零件1視覺：



描述
在此應用中，機器人根據零件的「正面」和「背面」朝向來分類零件。正面朝上的零件被放置在標有「PlaceFront」標籤的點位，背面朝上的零件被放置在標有「PlaceBack」標籤的另一個點位。
在此應用中，拾取順序不重要。機器人會盡可能取放更多的正面朝上零件，然後盡可能取放更多的背面朝上零件。
當「需要翻轉」核取方塊未勾選，並且選擇了「檢測正面零件的物件」和「檢測背面零件的物件」的視覺物件時，系統會將正面和背面朝上的零件都載入到座標佇列中，從而設定Part Orientation的值。
「需要翻轉」設定告訴系統您想要將零件置於特定朝向。不勾選「需要翻轉」核取方塊，告訴系統您想要將零件置於正面和背面兩種朝向。

在此情況下，正面朝上零件的朝向資料（零件座標佇列內）會自動設定為常數「PF_PARTORIENT_FRONT」。
背面朝上零件的資料會自動設定為常數「PF_PARTORIENT_BACK」

範例程式碼

Main.prg

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

PartFeeding.prg

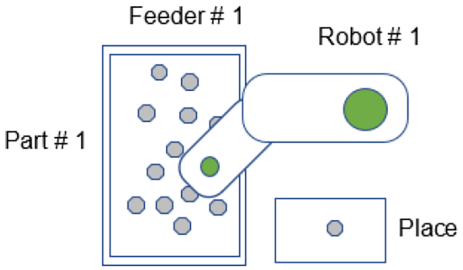
```
Function PF_Robot(PartID As Integer) As Integer
    Do While PF_QueLen(PartID) > 0
        P0 = PF_QueGet(PartID)
        Jump P0
        On Gripper; Wait 0.2
        If PF_QuePartOrient(PartID) = PF_PARTORIENT_FRONT Then
            Jump PlaceFront
        ElseIf PF_QuePartOrient(PartID) = PF_PARTORIENT_BACK 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 程式範例 1.10

範例類型：
使用PF_Vision回呼函數

配置

- 機器人數量：1
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：1
- 攝影機的朝向：送料器#1上的朝下固定的攝影機



描述

此範例說明如何使用PF_Vision回呼來取得影像並載入包含視覺結果的零件座標佇列。若要使用此功能，請在Epson RC+ 8.0功能表 - [工具] - [料件送料] - [視覺]中選擇[由PF_Vision callback進行視覺處理]。

範例程式碼

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
```

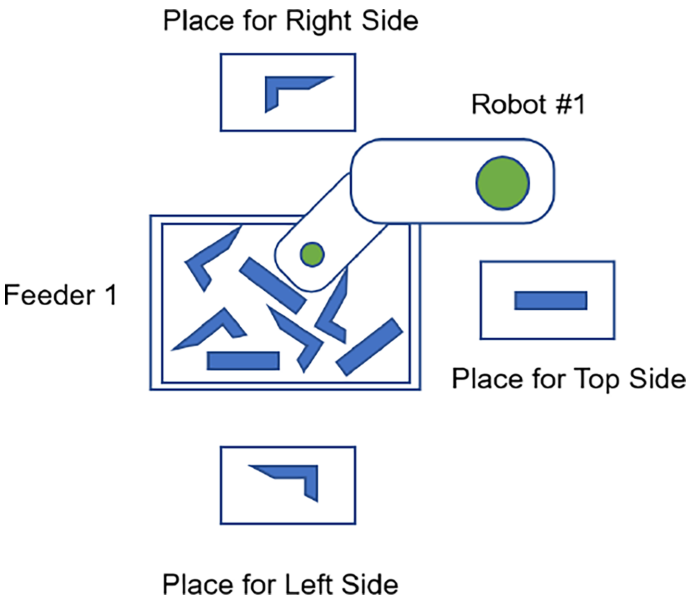
```
Fend
```

3.9.1.11 程式範例 1.11

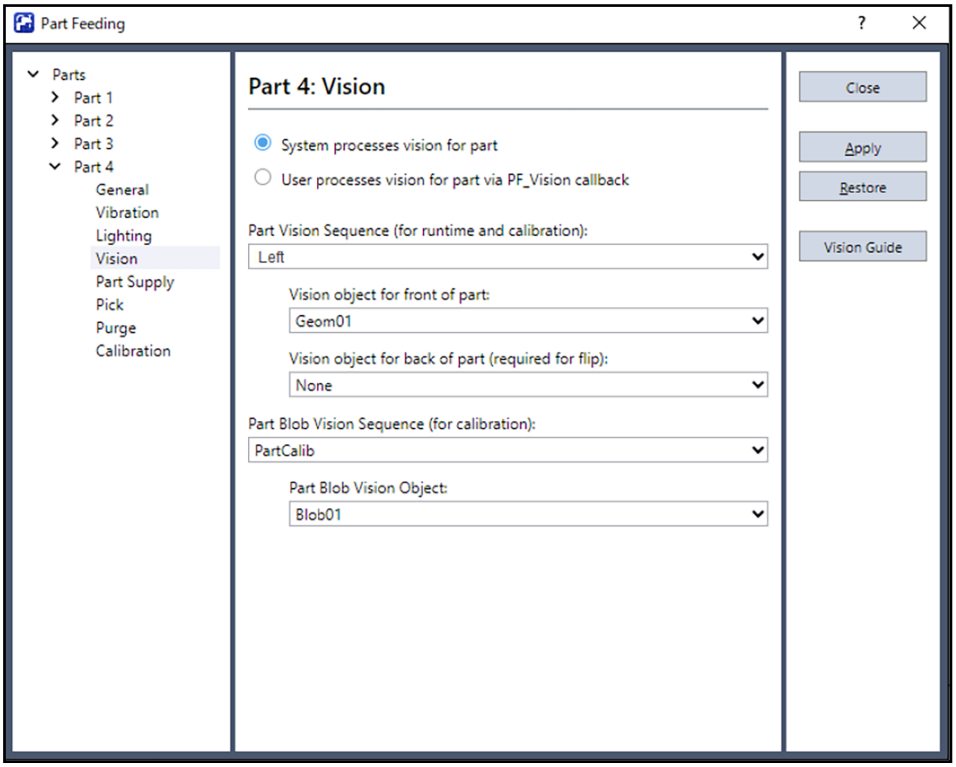
範例類型：
多重面向零件的選擇

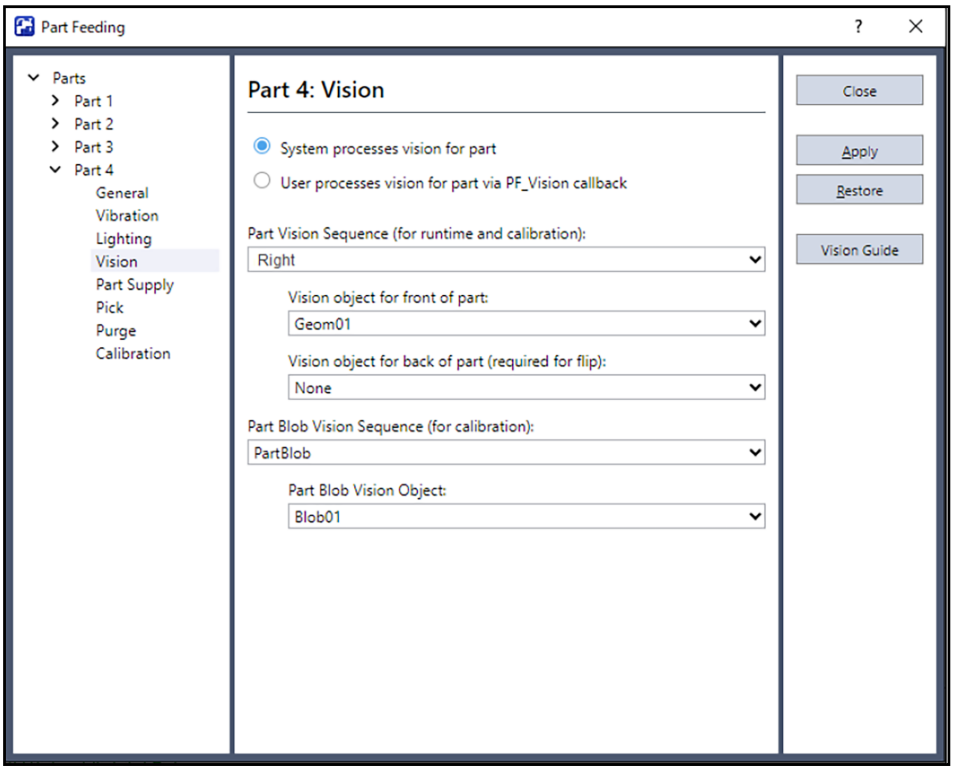
配置

- 機器人數量：1
- 送料器數量：1
- 送料器上的零件種類數：1
- 零件姿態數量：3
- 放置位置數量：3（零件各方向各1個放置位置）
- 攝影機的朝向：送料器#1上的朝下固定的攝影機



描述
此範例中，送料器上有1種實體零件。零件有3種姿態（右、左、上）。機器人可以以3種姿態中的任一種拾取零件。雖然實際上是同1個零件，但從視覺系統的角度來看，每種姿態都是不同的零件。零件送料支援在同一送料器上同時執行4個零件。因此，為每種姿態創建單獨的零件。創建3個零件視覺序列，命名為「Left」、「Right」、「Top」。
每個視覺序列使用Geometric物件將零件放置在特定姿態。在[料件送料]對話方塊中，創建3個零件：零件1、2、3。
零件1將零件放置在「左側」姿態。
零件2將零件放置在「右側」姿態。
零件3將零件放置在「上側」姿態。
3個零件全部都未勾選「需要翻轉」核取方塊。3個零件全部都使用名為「PartBlob」的相同零件Blob視覺序列。


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

教導每個零件的拾取Z（拾取高度可能因零件各姿態而異）。左側零件放置在標有「PlaceLeft」標籤的點位。右側零件放置在標有「PlaceRight」標籤的點位。上側零件放置在標有「PlaceTop」標籤的點位。機器人先取放所有左方向零件，然後取放所有右方向零件，最後取放所有上方向零件。

範例程式碼

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
```

提示

此範例展示了使用多種零件功能處理多重面向零件的簡單方法。另1種方法是通過PF_Vision回呼進行「User to Process Vision」(由使用者處理視覺)。使用PF_QueAdd可以在零件座標中包含使用者資料。使用者資料可以是表示零件姿態的數值 (1=左、2=右、3=上)。可以使用PF_QueUserData函數取得方向值，以便將零件放置在正確位置。此程式碼還需要考慮零件各側面高度的差異。

3.9.1.12 程式範例 1.12

範例類型：

使用PF_Vision回呼和多重搜尋的零件檢測視覺序列

配置

- 機器人數量：1
- 送料器數量：1

- 送料器上的零件種類數：1
- 放置位置數量：1
- 攝影機朝向：送料器#1上的朝下固定的攝影機

描述

PF_Vision回呼用於零件檢測視覺序列難以處理的情況（例如，需要多個視覺序列檢測零件，或需要特殊外部照明控制的情況）。

此範例展示如何使用PF_Vision回呼取得影像並將視覺結果載入零件座標佇列。若要使用此功能，請在Epson RC+ 8.0功能表 - [工具] - [料件送料] - [視覺]中選擇「由PF_Vision callback進行視覺處理」。

此範例使用視覺的多重搜尋。多重搜尋是搜尋物件的CenterPointObject屬性或Frame物件各結果的功能。在多重搜尋中，找到的結果可能不會按預期順序排列。PF_Vision展示了先通過迭代所有結果，並對找到的結果進行處理的方法。

- 使用CenterPointObject的多重搜尋範例

1. 創建用於搜尋多個零件的物件，如Blob。
2. 創建使用Blob的CenterPointObject的另一個物件，如Polar。
3. 將CenterPntObjResult屬性設為ALL。
4. 執行序列。對於每個找到的Blob結果，會顯示Polar物件的實例。

- 使用Frame的多重搜尋範例

1. 創建用於搜尋多個零件的物件，如Blob。
2. 創建Frame物件，並將OriginPoint屬性設為Blob。
3. 將OriginPntObjResult屬性設為All。
4. 創建使用Frame的另一個物件，如Polar。
5. 將FrameResult屬性設為All。
6. 執行序列。對於每個找到的Blob結果，顯示Frame物件的實例，對於每個Frame結果，顯示Polar物件的實例。

範例程式碼

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
```

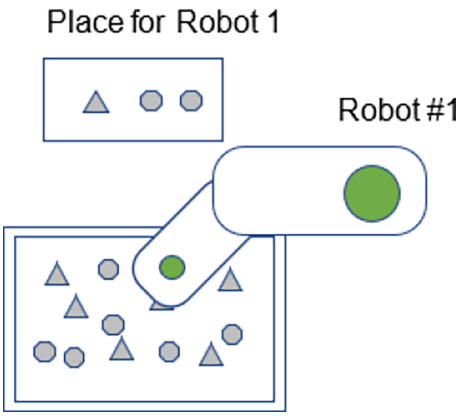
3.9.2 機器人1台 - 多種零件

3.9.2.1 程式範例 2.1

範例類型：
機器人1台與多種零件 - 機器人動作的視覺與振動平行處理

配置

- 機器人數量：1
- 送料器數量：1
- 送料器上的零件種類數：2
- 放置位置數量：1
- 攝影機的朝向：朝下固定的攝影機



描述
有2種零件（實體不同）。機器人連續取放2個零件#1，然後取放1個零件#2。通過交替執行「PF_ActivePart」來實現這樣的動作。在此應用中，拾取順序很重要（例如：零件組裝的情況）。當最後一個零件放置後，「PF_ActivePart」會被更改，以供應所需的零件，並向系統發送振動信號，必要時取得影像。這在機器人動作（在移動到「放置」位置的路徑達到30%時）過程中同時實現。

範例程式碼
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
```

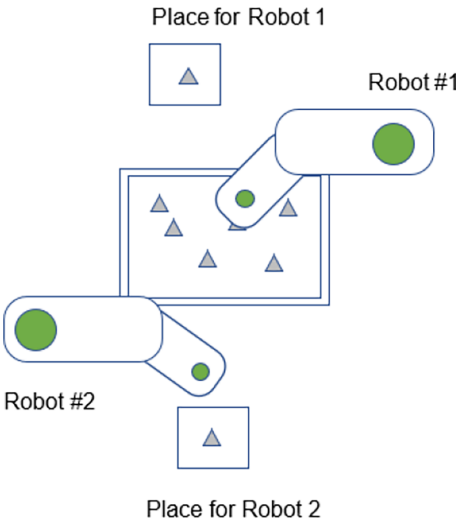
3.9.3 機器人2台 - 零件1種類

3.9.3.1 程式範例 3.1

範例類型：
機器人2台與實體零件1種類 - 在PF_Robot回呼中的動作 - 特定順序拾取

配置

- 機器人數量：2
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：2
- 攝影機的朝向：朝下固定的攝影機



描述
有2台機器人和1個送料器。只有1種實體零件。由於每台機器人有自己的攝影機校準，因此有2種邏輯部分（機器人1的零件1和機器人2的零件2）。
機器人按順序從送料器拾取。在此應用中，拾取順序很重要。交替的拾取順序通過「PF_ActivePart」執行。
機器人動作在PF_Robot回呼內執行。
此範例中沒有送料器和機器人動作的平行處理。程式碼簡單但效率不高。每台機器人都有標有「park」標籤的點位和標有「place」標籤的點位。此範例的重要概念是PF_Robot回呼函數的回傳值「PF_CALLBACK_RESTART_ACTIVEPART」。
此回傳值使多台機器人能使用同一個送料器，而兩個零件的佇列中的零件不會調整。回傳值強制只為PF_ActivePart取得新影像，並只載入PF_ActivePart的佇列。

範例程式碼

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

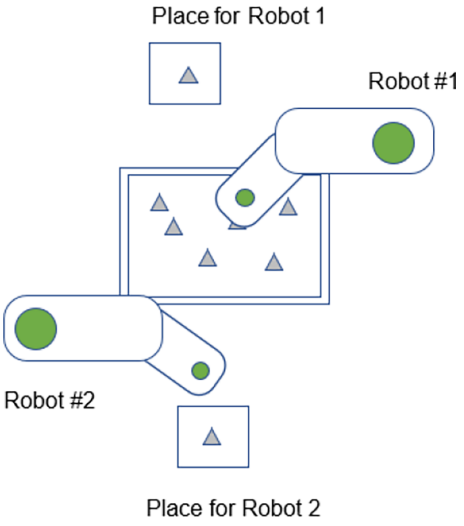
```
Function PF_Robot(PartID As Integer) As Integer
    If PF_QueLen(PartID) > 0 Then
        Select PartID
            Case 1
                Robot 1
                P0 = PF_QueGet(1)
                PF_QueRemove (1)
                Jump P0 /R
                On rbt1Gripper
                Wait 0.25
                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 程式範例 3.2

範例類型：
機器人2台與1種實體零件 - 在別的任務中的動作 - 拾取順序無關 - 特定順序拾取

配置

- 機器人數量：2
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：2
- 攝影機的朝向：朝下固定的攝影機



描述

有2台機器人和1個送料器。只有1種實體零件。由於每台機器人有自己的攝影機校準，因此有2種邏輯部分（機器人1的零件1和機器人2的零件2）。拾取順序無關 - 先到先得。

此範例的不同之處在於每個循環都會為每個零件取得視覺。這有助於解決在拾取過程中周圍零件可能移動的情況。

PF_Robot回呼函數的回傳值「PF_CALLBACK_RESTART」會強制對所有零件重新執行視覺，並重新載入所有零件佇列。

雖然這種方法效率不高，但「PF_CALLBACK_RESTART」在特定情況下很有用。

範例程式碼

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

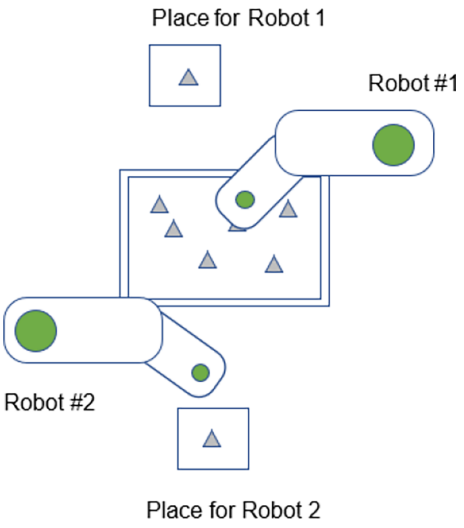
```
Function PF_Robot(PartID As Integer) As Integer
    MemOn PartsToPick
    Wait MemSw(PartsToPick) = Off

    PF_Robot = PF_CALLBACK_RESTART 'Force vision and vibration to refresh
Fend
```

3.9.3.3 程式範例 3.3

範例類型：
機器人2台與1種實體零件 - 在別的任務中的動作 - 先到先得 - 模擬的程序延遲

配置

- 機器人數量：2
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：2
- 攝影機的朝向：朝下固定的攝影機



描述

有2台機器人和1個送料器。只有1種實體零件。由於每台機器人有自己的攝影機校準，因此有2個邏輯部分（機器人1的零件1和機器人2的零件2）。此範例中，各機器人的程序時間是可變的（通過隨機等待時間模擬）。
每台機器人在從送料器取出零件後，會變為忙碌狀態以執行其他操作。
記憶體位元「Rbt1Complete」和「Rbt2Complete」用於在機器人從送料器拾取零件後，完成其他操作並準備好從送料器拾取另一個零件時發送信號。當所需的零件（PF_ActivePart）與當前零件不同時（即另一台機器人拾取時），PF_Robot回呼函數會傳回「PF_CALLBACK_RESTART_ACTIVEPART」值。這可防止機器人佇列中的點位重複。會取得PF_ActivePart的新影像，並且只載入PF_ActivePart的佇列。但是，當下一個零件與當前零件相同時（即同一台機器人從送料器拾取時），PF_Robot回呼函數的回傳值為「PF_CALLBACK_SUCCESS」。PF_AccessFeeder和PF_ReleaseFeeder確保機器人在存取送料器時不會碰撞。

範例程式碼

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
```

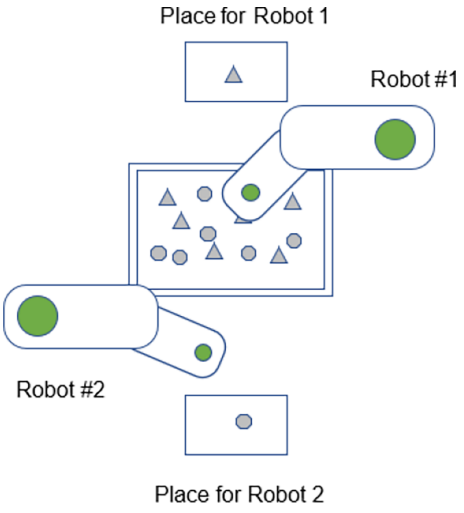
3.9.4 機器人2台 - 多種零件

3.9.4.1 程式範例 4.1

範例類型：
機器人2台，送料器1個，多種零件 - 在PF_Robot callback中的動作 - 特定順序拾取

配置

- 機器人數量：2
- 送料器數量：1
- 送料器上的零件種類數：2
- 放置位置數量：2
- 攝影機的朝向：朝下固定的攝影機



描述
有2台機器人和1個送料器。每台機器人拾取具特有特徵（實體不同）的零件。機器人按順序從送料器拾取。在此應用中，拾取順序很重要。
交替的拾取順序通過「PF_ActivePart」執行。機器人動作在PF_Robot回呼內執行。此範例中沒有送料器和機器人動作的平行處理。程式碼簡單但效率不高。
機器人1取放零件#1。機器人2取放零件#2。每台機器人都有標有「park」標籤的點位和標有「place」標籤的點位。

範例程式碼
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

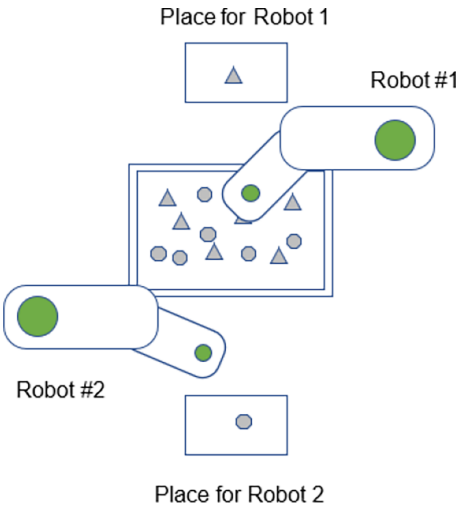
```
Function PF_Robot(PartID As Integer) As Integer
    If PF_QueLen(PartID) > 0 Then
        Select PartID
            Case 1
                Robot 1
                P0 = PF_QueGet(1)
                PF_QueRemove (1)
                Jump P0 /R
                On rbt1Gripper
                Wait 0.25
                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 程式範例 4.2

範例類型：
機器人2台，送料器1個，多種零件 - 在別的任務中的動作 - 特定順序拾取

配置

- 機器人數量：2
- 送料器數量：1
- 送料器上的零件種類數：2
- 放置位置數量：2
- 攝影機的朝向：朝下固定的攝影機



描述

有2台機器人和1個送料器。每台機器人拾取具特有特徵（實體不同）的零件。機器人按順序從送料器拾取。通過交替執行「PF_ActivePart」來實現這樣的動作。

當1台機器人沒有零件可取時，另一台機器人可以繼續從送料器拾取，直到零件用完。機器人1取放零件#1。機器人2取放零件#2。

每台機器人都有標有「park」標籤的點位和標有「place」標籤的點位。「PF_AccessFeeder」和「PF_ReleaseFeeder」用於防止兩台機器人同時存取送料器。當任一機器人移動到放置位置距離的30%時，另一台機器人可以存取送料器。

範例程式碼

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
```

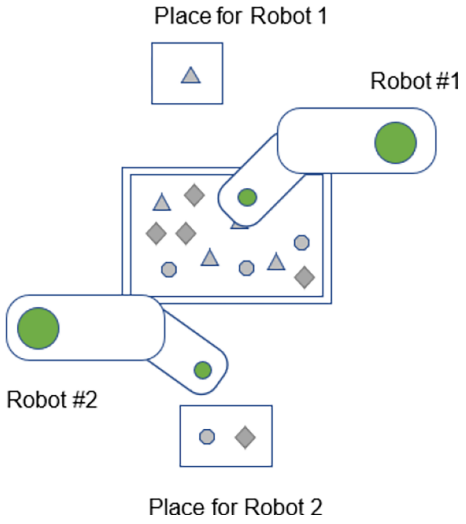
3.9.4.3 程式範例 4.3

範例類型：
機器人2台，送料器1個，多種零件 - 在別的任務中的動作 - 特定順序拾取

配置

- 機器人數量：2
- 送料器數量：1
- 送料器上的零件種類數：3
- 放置位置數量：2

- 攝影機的朝向：朝下固定的攝影機



描述

有2台機器人和1個送料器。每台機器人取出不同的零件。
機器人1取放零件1中的一個。
機器人2取放零件4和零件5中的一個。在此應用中，拾取順序很重要。交替的拾取順序通過「PF_ActivePart」執行。
機器人動作與送料器振動同時進行。

範例程式碼

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 在PF_Feeder回呼函數中控制振動

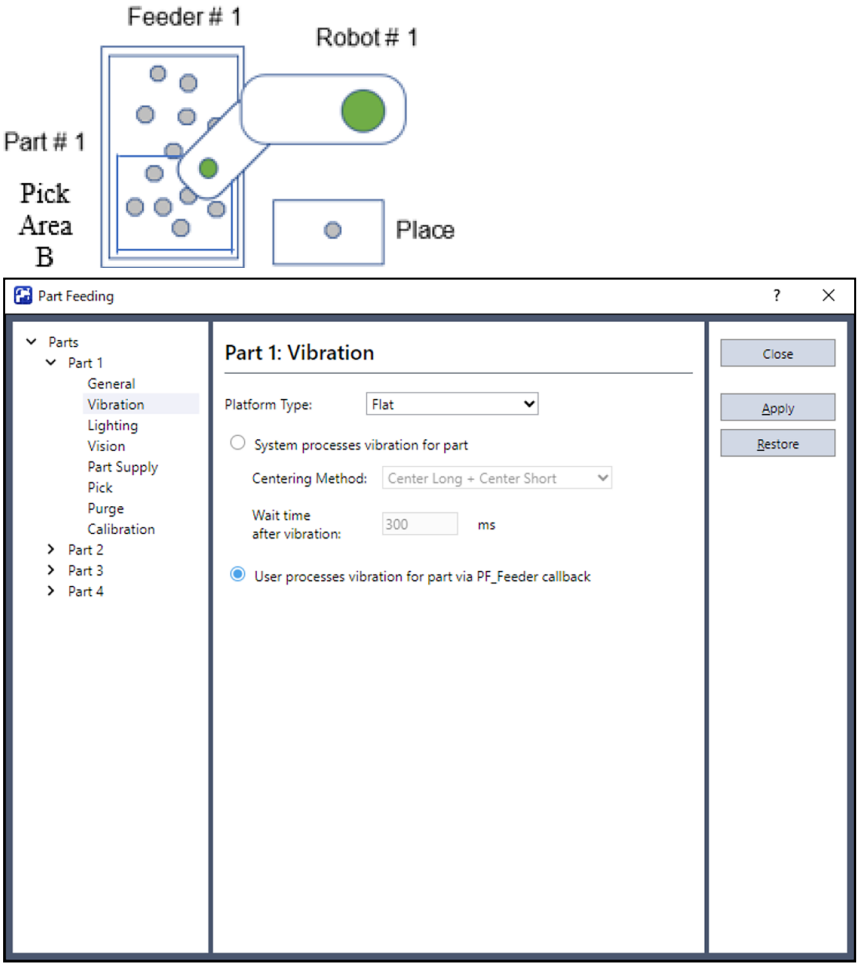
3.9.5.1 程式範例 5.1

範例類型：
平面平台 - 在PF_Feeder回呼函數中控制振動

配置

- 機器人數量：1
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：1
- 平台類型：平面
- 拾取區域：區域 B

- 攝影機的朝向：朝下固定的攝影機



描述

此範例使用標準的平面平台。使用平面平台時，通常選擇功能表 - [工具] - [料件送料] - [零件] - [振動]中的[系統控制振動]按鈕。

此範例介紹選擇[在PF_Feeder callback中控制振動]，並自行描述振動處理的方法。使用者的振動處理在PF_Feeder回呼函數內執行。若需要與系統提供的振動處理不同的振動處理，請選擇[在PF_Feeder callback中控制振動]。使用自訂平台（孔洞、溝槽、凹槽等）時，需要通過PF_Feeder回呼函數自行處理振動。

如需詳細資訊，請參閱以下內容。

程式範例 5.2

選擇[在PF_Feeder callback中控制振動]時（適用於平面、防粘附、防滾動等各種標準平台），系統仍會判斷各種情況下零件的最適當處理方式。判斷結果通過引數「state」提供給PF_Feeder回呼函數。各種狀態的常數值在「PartFeeding.inc」檔案中定義。

例如，常數「PF_FEEDER_PICKOK」表示零件可以被機器人取放。另一個例子，當系統判斷翻轉零件是最佳處理方式時，常數「PF_FEEDER_FLIP」會傳遞給PF_Feeder回呼函數。如何處理引數「state」的值由使用者決定。

概念上，使用者可以使用PF_Feeder回呼函數的引數「state」和適當的送料器控制命令來重建系統處理。再次強調，對於平面平台，通常選擇[系統控制振動]。這表示此範例展示如何使用PF_Feeder回呼函數和送料器控制命令來模擬系統處理。

範例程式碼

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
```

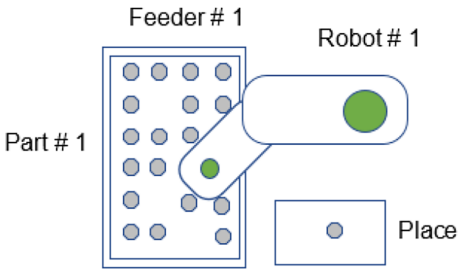
```
        ' Parts have gathered against the platform wall
        Case PF_FEEDER_SHIFT_BACKWARDS
            PF_Shift PartID, PF_SHIFT_BACKWARD
            PF_Feeder = PF_CALLBACK_RESTART

        ' 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 程式範例 5.2

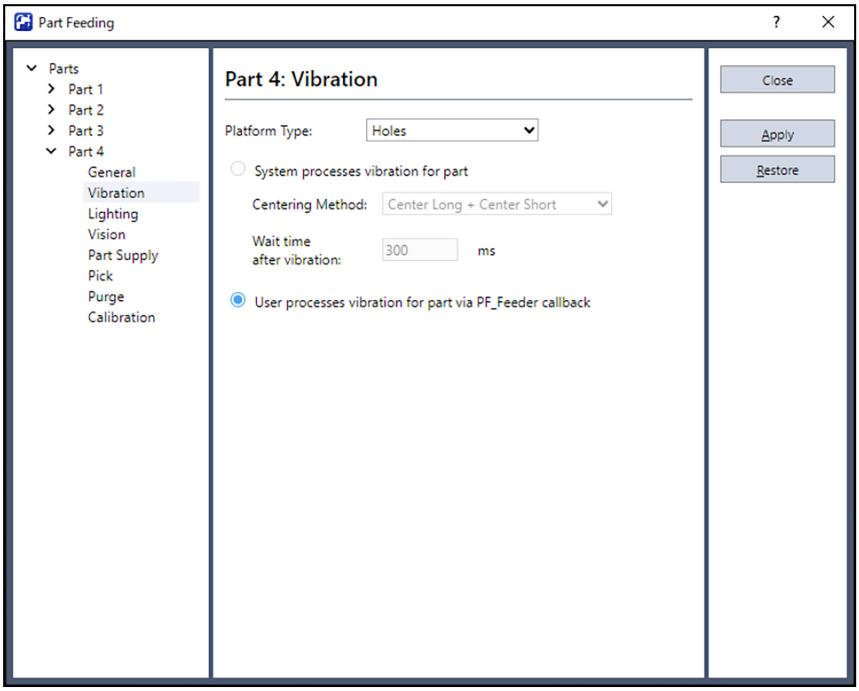
範例類型：
自訂平台（帶孔洞）- 在PF_Feeder回呼中控制振動

配置

- 機器人數量：1
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：1
- 平台類型：孔洞
- 拾取區域：全面
- 攝影機的朝向：朝下固定的攝影機



描述
由於是自訂平台，[在PF_Feeder回呼中控制振動]會自動被選擇。



視覺取得影像並載入零件佇列後，會呼叫PF_Feeder回呼函數。使用者的程式碼需要在PF_Feeder回呼函數中判斷如何使送料器振動。正面朝上和背面朝上的零件數量會作為引數提供給PF_Feeder回呼。這些引數是「NumFrontParts」和「NumBackParts」。在此範例中，在NumFrontParts大於「0」時機器人可以取放零件，因此不需要送料器振動。這種情況下，回呼的回傳值為「PF_CALLBACK_SUCCESS」。此回傳值指示系統呼叫PF_Robot回呼函數。

當NumFrontParts為「0」時，範例程式碼會執行VRun以執行零件Blob序列，判斷是否有零件堆積或完全沒有零件。如果零件Blob序列未找到零件，則開啟料斗。如果零件Blob序列找到了一些東西，則送料器會執行翻轉動作、前向位移動作和後向位移動作，使零件落入孔洞中。當零件在送料器上振動後，系統必須重新取得視覺影像。（因為振動會改變零件的位置）這通過設定回傳值為「PF_CALLBACK_RESTART」來實現。這會重新取得新影像，重新載入零件座標佇列，然後再次呼叫PF_Feeder回呼函數，以判斷是否需要進一步動作。

TIP

翻轉、長時間前向位移、短時間後向位移是對各種自訂平台通用的有效動作。

提示

當平台類型為孔洞、溝槽、凹槽時，常數PF_FEEDER_UNKNOWN將會被傳遞至PF_Feeder回呼函數。這是因為在使用自訂平台時，系統無法決定適當的送料器運作所導致。

範例程式碼

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
```

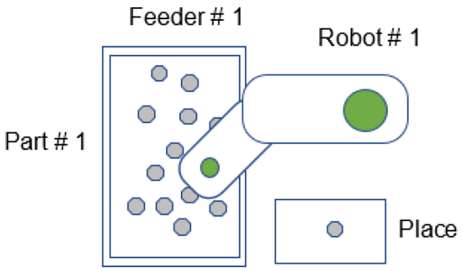
3.9.6 錯誤處理

3.9.6.1 程式範例 6.1

範例類型：
回呼函數內潛在錯誤狀態的處理

配置

- 機器人數量：1
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：1
- 攝影機的朝向：送料器#1上的朝下固定的攝影機



描述
此範例展示如何檢測並處理回呼函數內的潛在錯誤狀態，以防止Part Feeding程序迴圈中發生錯誤。此範例使用PF_Vision回呼來取得影像並將視覺結果載入零件座標佇列。機器人有較大的工具偏移量。在某些情況下，機器人可能會超出工作範圍，因此無法使工具配合零件角度（由視覺檢測）。
若不進行錯誤處理，將導致「座標轉換」錯誤。此範例程式碼在將座標載入零件座標佇列前，會檢查機器人是否能以該角度拾取。這通過TargetOK陳述式實現。

範例程式碼

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
```

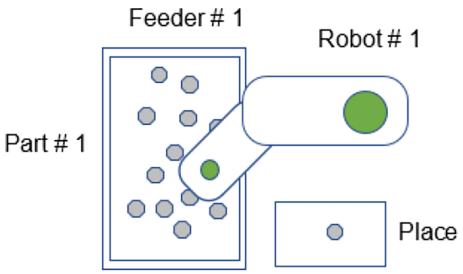
```
    Do While PF_QueLen(PartID) > 0
        P0 = PF_QueGet(PartID)
        Jump P0
        On Gripper; Wait 0.2
        Jump Place
        Off Gripper; Wait 0.2
        PF_QueRemove PartID
        If PF_IsStopRequested(PartID) = True Then
            Exit Do
        EndIf
    Loop
    PF_Robot = PF_CALLBACK_SUCCESS

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 程式範例 6.2

範例類型：
回呼函數內處理發生錯誤的處理

配置

- 機器人數量：1
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：1
- 攝影機的朝向：送料器#1上的朝下固定的攝影機



描述

此範例使用PF_Vision回呼來取得影像並將視覺結果載入零件座標佇列。此範例中，需要在送料器上有最小數量的可拾取零件（此例中為5個）才能載入零件座標佇列。如果嘗試3次後仍未能載入零件座標佇列，會顯示訊息詢問操作員是否繼續搜尋零件或停止。

範例程式碼

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
```

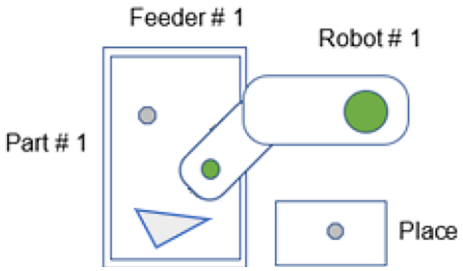
3.9.6.3 程式範例 6.3

範例類型：

PF_Status回呼中的狀態錯誤處理

配置

- 機器人數量：1
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：1
- 攝影機的朝向：送料器#1上的朝下固定的攝影機



描述

此範例中，檢測到托盤中的最後一個零件為不良零件。托盤安裝並啟用了清除閘門選項。檢測到的「不良零件」會從托盤排出，然後從料斗供應新零件，並執行集中和翻轉動作。

範例程式碼

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
```

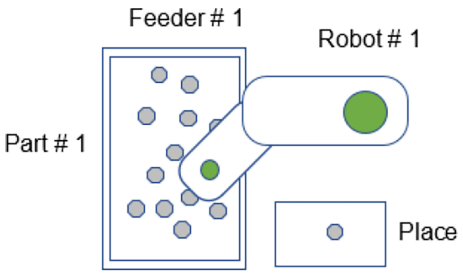
3.9.6.4 程式範例 6.4

範例類型：

PF_Status回呼中的使用者錯誤處理

配置

- 機器人數量：1
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：1
- 攝影機的朝向：送料器#1上的朝下固定的攝影機



描述

機器人配備有帶真空開關的真空杯夾具，用於檢測零件是否正確取起。當真空感應器無法檢測到零件時，機器人會嘗試重新拾取。連續3次失敗後，會生成使用者錯誤(8000)。

使用者錯誤通過將PF_Robot的回傳值設為使用者錯誤編號，傳送到PF_Status回呼。PF_Status回呼會顯示一個訊息方塊，允許操作員選擇繼續或結束應用程式。

範例程式碼

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
```

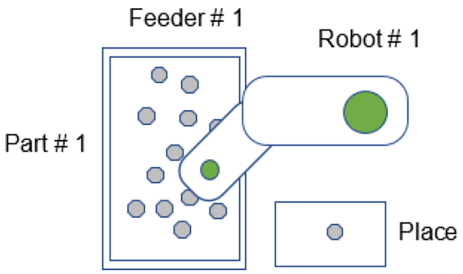
3.9.6.5 程式範例 6.5

範例類型：

Part Feeding回呼中的控制器錯誤的處理

配置

- 機器人數量：1
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：1
- 攝影機的朝向：送料器#1上的朝下固定的攝影機



描述

通常，當控制器錯誤發生時，Part Feeding程序會自動將常數PF_STATUS_ERROR設定到Status參數中，並呼叫PF_Status回呼函數。這無需追加使用者程式碼即可發生。PF_Status回呼函數會輸出或顯示錯誤編號和訊息。
PF_Status回呼函數結束後，Part Feeding程序也會結束。
然而，此範例中需要在PF_Robot回呼函數內處理特定的控制器錯誤。
所有其他控制器錯誤會以Status參數設為PF_STATUS_ERROR的狀態呼叫PF_Status回呼函數。
在此情況下，為防止夾具的電氣佈線和氣壓管路損壞，必須確保U軸（SCARA機器人）不會旋轉超過+/-360°。
第4軸的動作範圍在Epson RC+的機器人管理器中受到限制。限制第4軸的動作範圍可防止電氣佈線和氣壓管路損壞。
若送料器上的零件需要使第4軸旋轉超出其動作範圍，將發生錯誤4001「手臂已達到動作範圍的極限」。錯誤處理程序會從佇列中刪除該零件，機器人會繼續取起剩餘的所有零件。
為防止視覺系統重新拍攝同一被拒絕零件的影像並再次將其添加到佇列中，在所有零件被選取且佇列為空後，會執行PF_Flip。

範例程式碼

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 多攝影機的使用

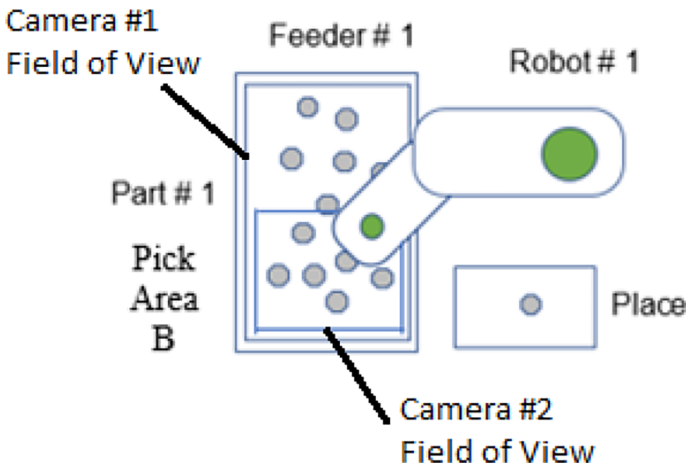
本章說明如何使用多個攝影機提高拾取精度。

3.9.7.1 程式範例 7.1

範例類型：
使用多個朝下固定的攝影機提高拾取區域精度

配置

- 機器人數量：1
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：1
- 平台類型：平面
- 拾取區域：區域B
- 攝影機的朝向
  - 攝影機#1：朝下固定。視野覆蓋整個送料器托盤。用於零件Blob序列
  - 攝影機#2：朝下固定。視野與區域B相同（托盤的一半）。用於零件檢測序列



描述

攝影機#1的視野設定為能夠拍攝整個送料器托盤。零件Blob序列使用攝影機#1。零件Blob序列用於根據托盤內零件的數量和分佈來決定送料器的振動類型。攝影機#2的視野設定為能夠拍攝整個拾取區域B。為了有效利用攝影機視野，需要將攝影機#2相對於攝影機#1旋轉90度（也就是說，兩個攝影機互相垂直）。攝影機#2用於零件檢測序列。零件檢測序列檢測到的零件用於生成零件座標佇列。由於視野是攝影機#1視野的一半大小，因此可以大幅提高解析度（mm/像素）。

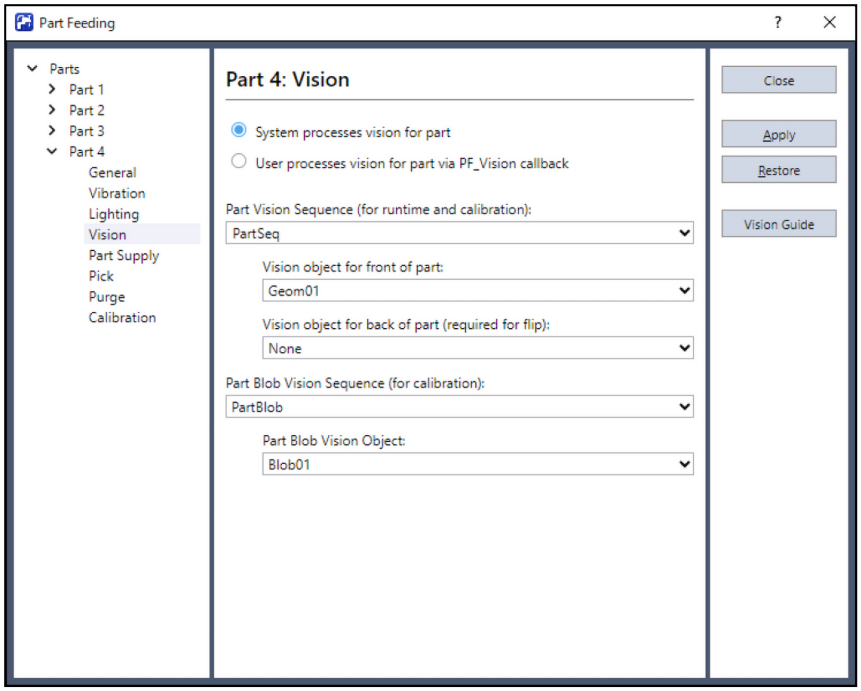
此外，由於攝影機#1僅用於判斷送料器的振動方法，所以可以使用比攝影機#2解析度更低的攝影機。例如，攝影機#1的解析度為640×480像素，而攝影機#2的解析度為5472×3648像素。

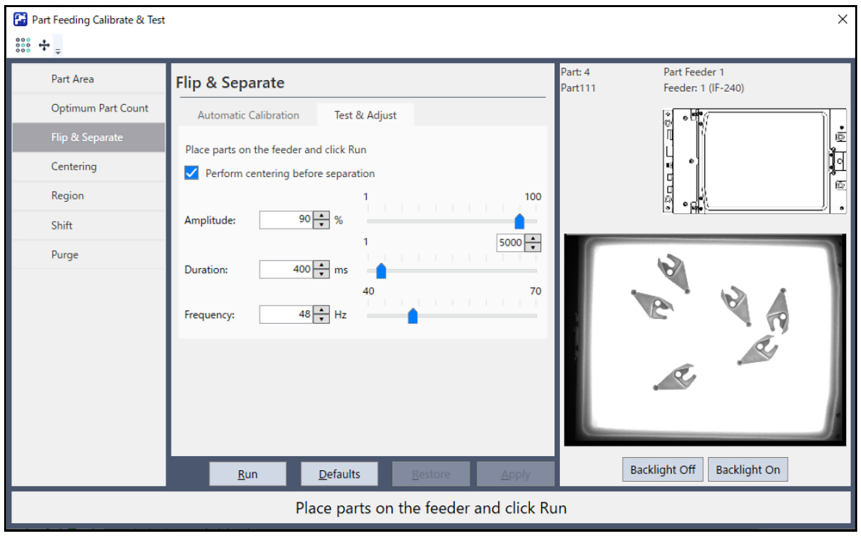
通過縮小視野和提高攝影機解析度，可以提高機器人的拾取精度。

翻轉與分離的自動校準使用零件檢測序列來確認有找到可拾取的零件。當小視野攝影機用於零件檢測序列時，如果不採取追加步驟，翻轉與分離的自動校準將無法正常運作。

請執行以下任一方法：

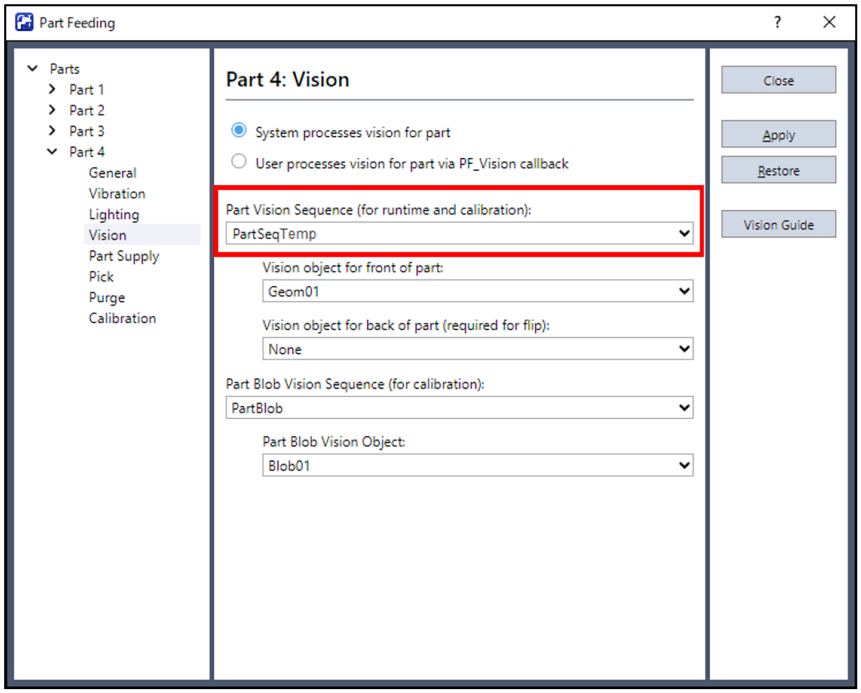
1. 跳過自動校準，手動調整翻轉與分離校準參數。



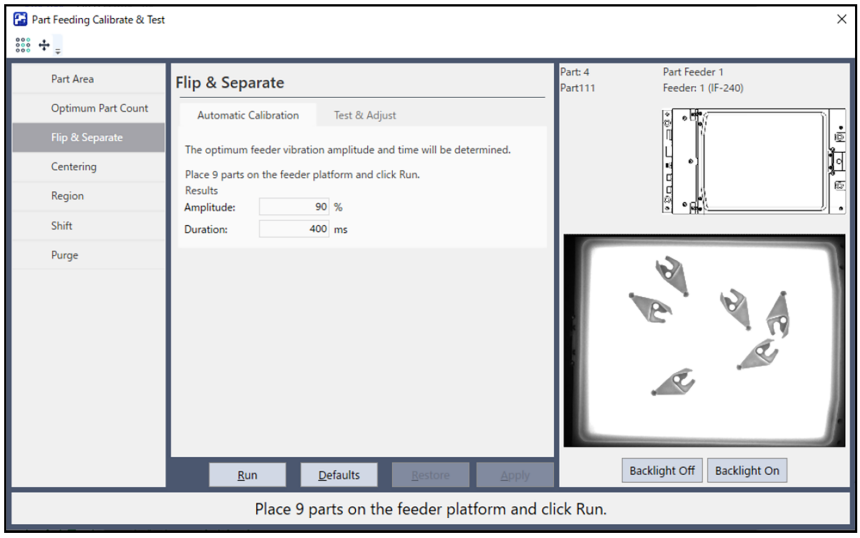


2. 創建一個僅用於自動校準的臨時零件檢測序列，使用攝影機#1（廣視野）。
在此範例中，使用攝影機#2（窄視野）的零件檢測序列名為「PartSeq」。使用攝影機#1（廣視野）的臨時零件檢

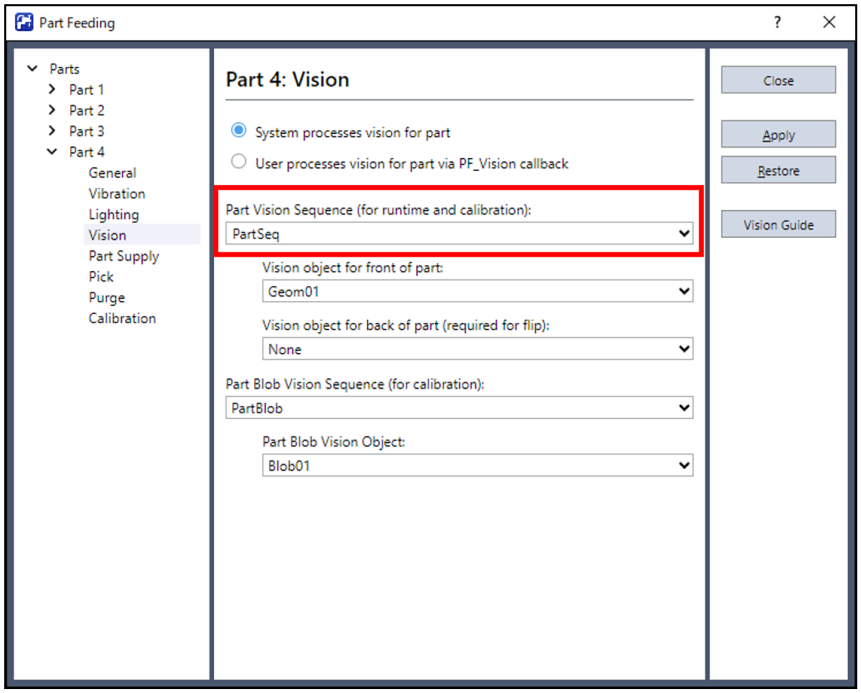
測序列名為「PartSeqTemp」。「PartSeqTemp」使用Geometric物件搜尋零件。
「PartSeqTemp」僅臨時用於送料器校準。選擇「PartSeqTemp」作為零件檢測視覺序列（參閱以下內容）。



3. 前往Part Feeding對話方塊的校準與測試頁面，執行分離的自動校準。



4. 完成所有必要的校準後，關閉校準與測試對話方塊。將零件檢測視覺序列改回「PartSeq」（使用窄視野的攝影機#2）。執行時，使用「PartSeq」的結果生成零件座標佇列。



無需特殊程式碼。

範例程式碼

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 程式範例 7.2

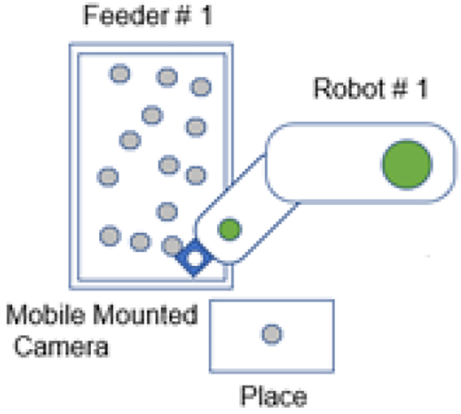
範例類型：

使用朝下固定的攝影機和移動式攝影機來提高拾取精度

配置

- 機器人數量：1

- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：1
- 平台類型：平面
- 拾取區域：全面
- 攝影機的朝向
  - 攝影機#1：朝下固定。視野覆蓋整個送料器托盤。用於零件Blob序列及零件檢測序列
  - 攝影機#2：移動式（J2軸）。視野略大於零件。拾取前取得零件的二維影像



描述

移動式攝影機（第2軸）定義了「手臂」（僅SCARA機器人）。使用6軸機器人時，移動式攝影機（第6軸）可以定義工具（而非手臂）。

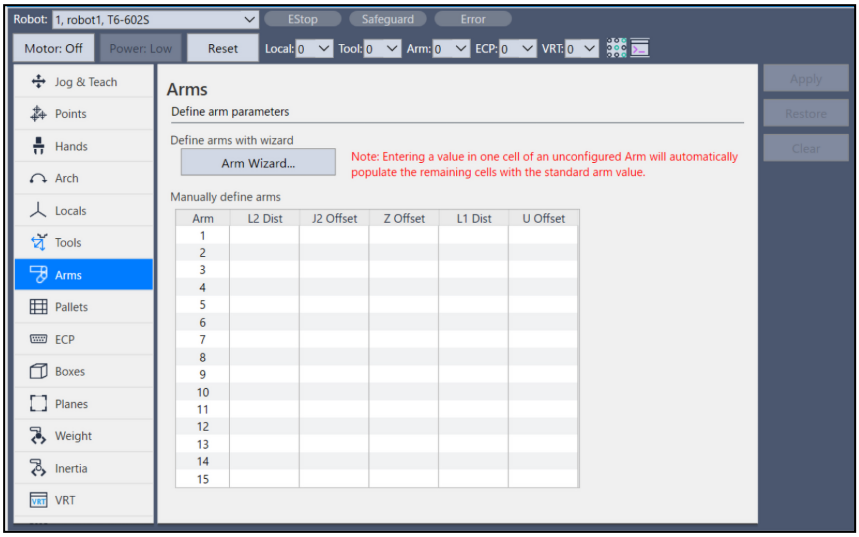
不是直接指示夾具移動到零件座標，而是攝影機#2移動到零件上方。執行追加的視覺序列，確定更精確的零件拾取位置。

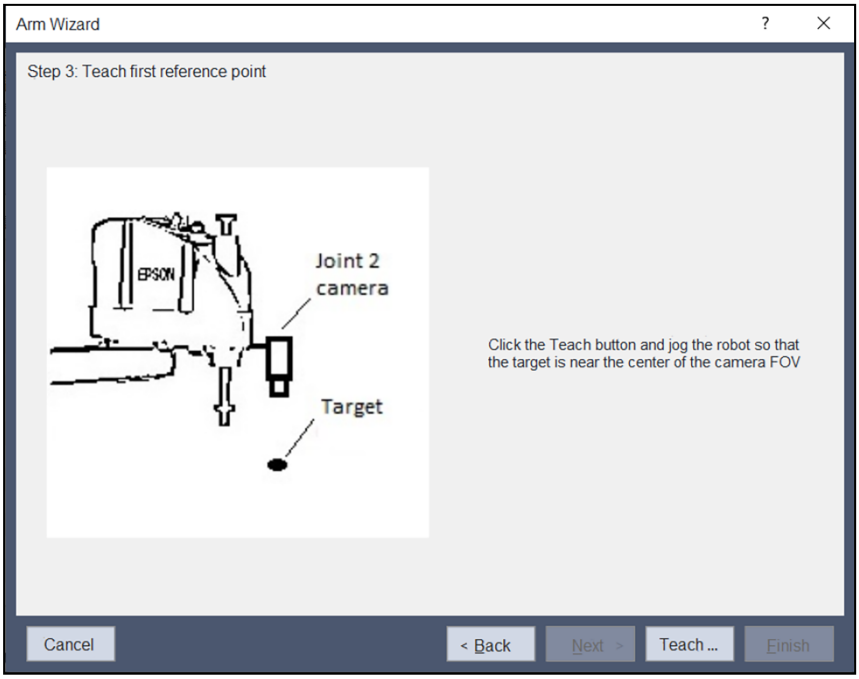
由於移動式攝影機的視野比朝下固定的攝影機小得多，因此提高了拾取精度。由於移動攝影機移動到零件上方需要追加動作和視覺處理，整體週期時間會增長。

若要自動定義手臂，請前往Epson RC+ - 工具 - 機器人管理器。選擇機器人管理器的[增設型手臂]標籤。在此範例中，選擇手臂1，然後點擊[增設型手臂設定精靈]按鈕。執行精靈的各個步驟。

如需精靈的詳細資訊，請參閱以下手冊。

「Vision Guide 8.0軟體篇 - 攝影機安裝位置的手臂設定」





校準移動式攝影機，並創建能在送料器上找到單個零件的視覺序列。此序列在PF_Robot回呼內執行(VRun)。在此範例中，序列名為「MobileCam」。在零件送料對話方塊中不選擇「MobileCam」。零件Blob序列和零件檢測序列使用攝影機#1，並按照正常方式在零件送料對話方塊中選擇。

範例程式碼

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
```

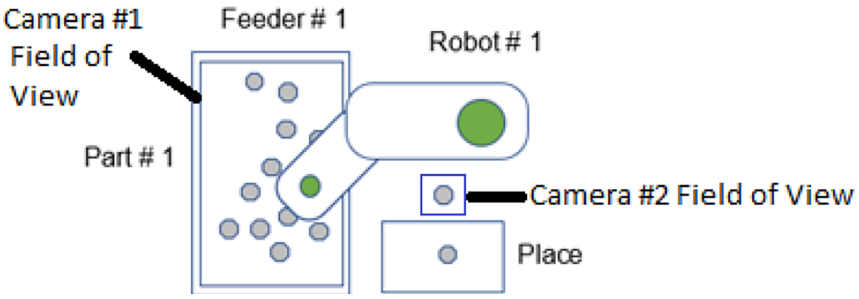
```
    Do While PF_QueLen(PartID) > 0
        P0 = PF_QueGet(PartID)
        Arm 1 'Select the Arm that is defined for the Mobile Camera
        Jump P0 :Z(0) 'Position the Mobile camera over the part
        VRun MobileCam
        VGet MobileCam.Geom01.RobotXYU, found, x, y, u
        Arm 0 'Select 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 程式範例 7.3

範例類型：
使用多個固定攝影機提高拾取精度

配置

- 機器人數量：1
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：1
- 平台類型：平面
- 拾取區域：全面
- 攝影機的朝向
  - 攝影機#1：朝下固定。視野覆蓋整個送料器托盤。用於零件Blob序列及零件檢測序列
  - 攝影機#2：朝上固定。用於為機器人夾具持有的零件創建工具偏移



描述
攝影機#1位於送料器托盤上方，固定朝下。攝影機#2固定朝上。攝影機#1的視野覆蓋整個送料器托盤。零件Blob視覺序列和零件檢測序列使用攝影機#1。
攝影機#2的視野略大於零件尺寸。機器人從送料器拾取零件後，會將零件移至攝影機#2上方。攝影機#2用於動態創建夾具持有零件的工具偏移。機器人使用新定義的工具偏移來放置零件。工具偏移用於補正從送料器拾取的不準確性。
攝影機#2僅在PF_Robot回呼中使用。在Epson RC+ 8.0功能表 - [工具] - [料件送料] - [視覺]中不設定攝影機#2的視覺序列。範例程式碼中使用攝影機#2的VGet RobotToolXYU結果來確定工具偏移。

此PF_Robot函數首先執行使用夾具拾取零件的步驟。接著，使用VGet RobotToolXYU取得工具偏移，並使用TLSet定義工具。然後，機器人使用新的工具偏移放置零件。

在本範例中，需要對朝上固定的攝像機進行校準。如需朝上固定的攝像機的校準方法詳細資訊，請參閱以下手冊。
「Vision Guide 8.0 Software - 校準步驟：朝上固定的攝像機」

範例程式碼

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 視覺結果的改進

本章說明如何改進視覺結果。

3.9.8.1 程式範例 8.1

範例類型：
使用ImageBuffer與ImageOp、SubtractAbs

配置

- 機器人數量：1

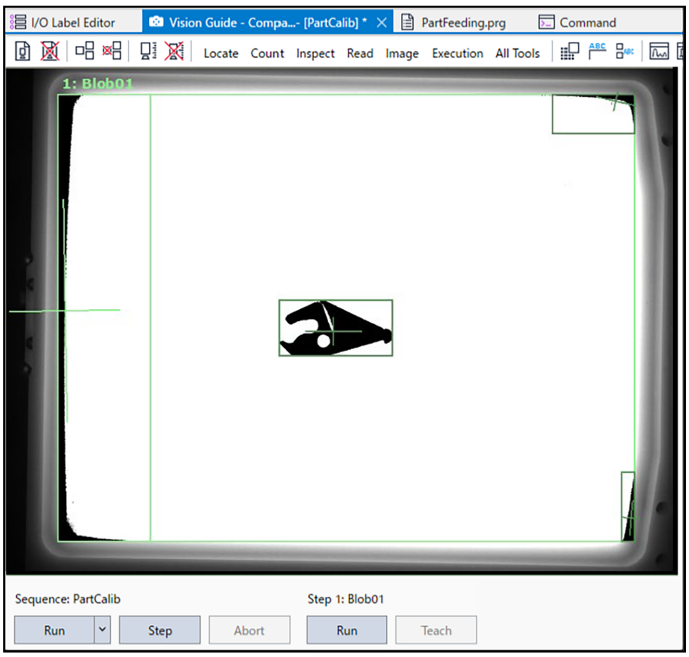
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：1
- 攝影機的朝向：送料器#1上的朝下固定的攝影機

描述

即使開啟背光，托盤角落有時也會形成陰影。如果Blob物件的搜尋視窗包含托盤角落，且閾值未適當調整，陰影可能被誤認為零件。

零件Blob視覺序列用於檢測個別零件或零件總數。如果零件Blob視覺序列將陰影視為零件，系統可能對振動方法做出錯誤判斷，或由於系統認為托盤上有足夠零件，導致PF_Control回呼不開啟料斗。

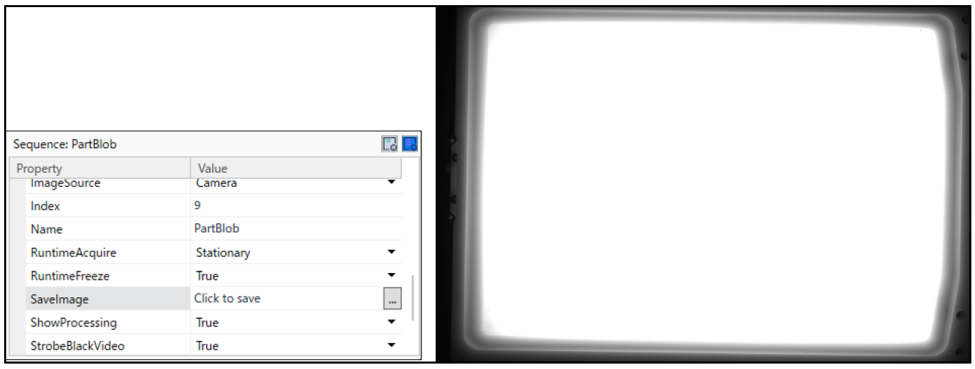
以下是托盤角落的陰影可能被誤認為零件的例子。



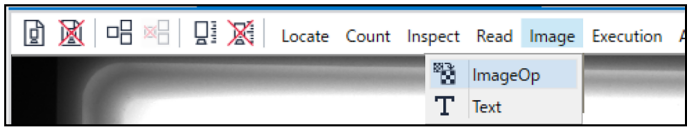
將ImageOp視覺物件追加到PartBlob序列中可幫助解決此問題。使用ImageOp的SubtractAbs處理。SubtractAbs輸出2個影像緩衝區之間的差異。在此範例中，ImageBuffer1屬性為空送料器的影像檔案，ImageBuffer2屬性為攝影機拍攝的影像（值「0」表示攝影機的影像緩衝區）。影像緩衝區相減後，送料器的影像被有效移除。

步驟如下：

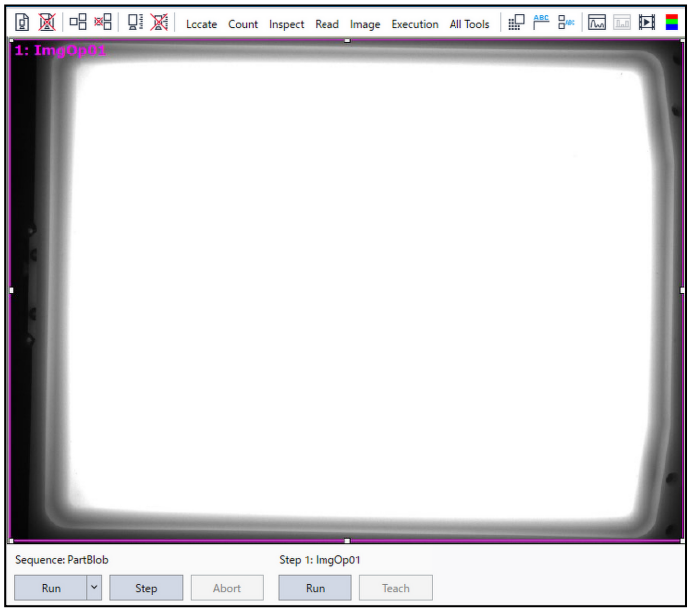
1. 創建新的視覺序列「PartBlob」。
2. 開啟送料器的背光，並移除送料器托盤上的所有零件。
3. 點擊PartBlob中SaveImage屬性的[Click to save]按鈕。在此範例中，將影像檔案命名為「EmptyIF-240」。檔案的圖片如下所示。



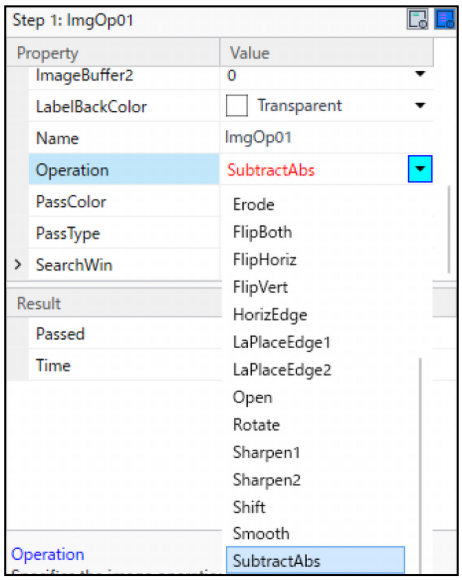
4. 從Vision Guide工具欄拖放ImageOp物件。



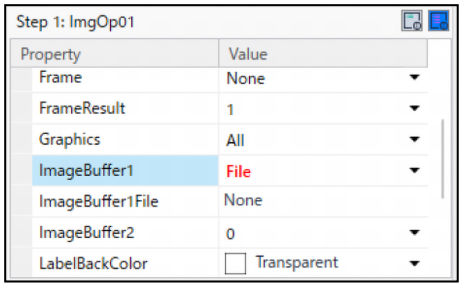
5. 調整ImageOp大小，使其覆蓋整個攝影機視野。



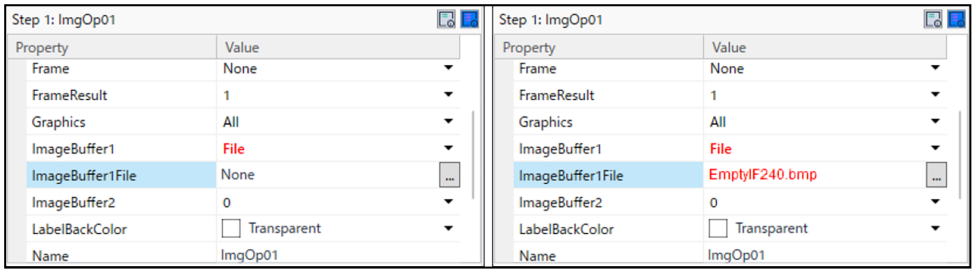
6. 將ImageOp的Operation屬性更改為「SubtractAbs」。



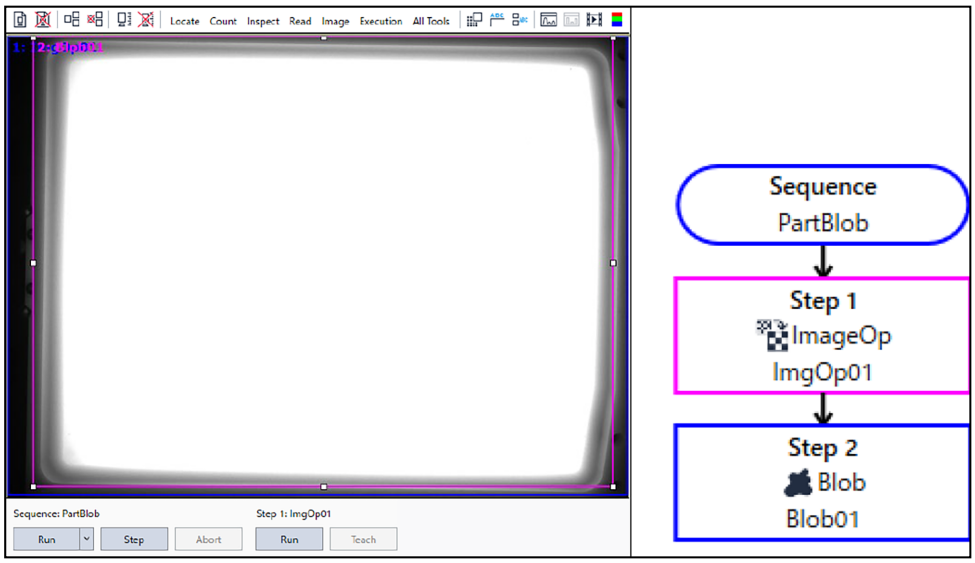
7. 將ImageOp的ImageBuffer1屬性更改為「File」。



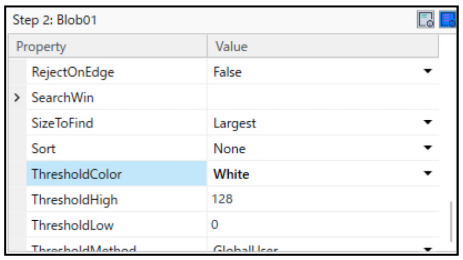
8. 點擊ImageBufferFile按鈕，選擇之前保存為「EmptyIF-240」的檔案。



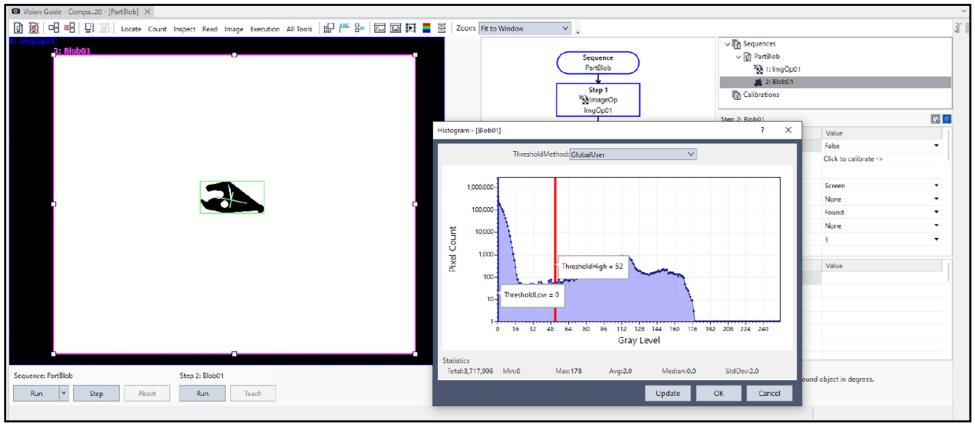
9. 從Vision Guide工具欄追加Blob物件。調整Blob的搜尋視窗大小，使其大於送料器托盤。



10. 將Blob的ThresholdColor屬性更改為「White」。

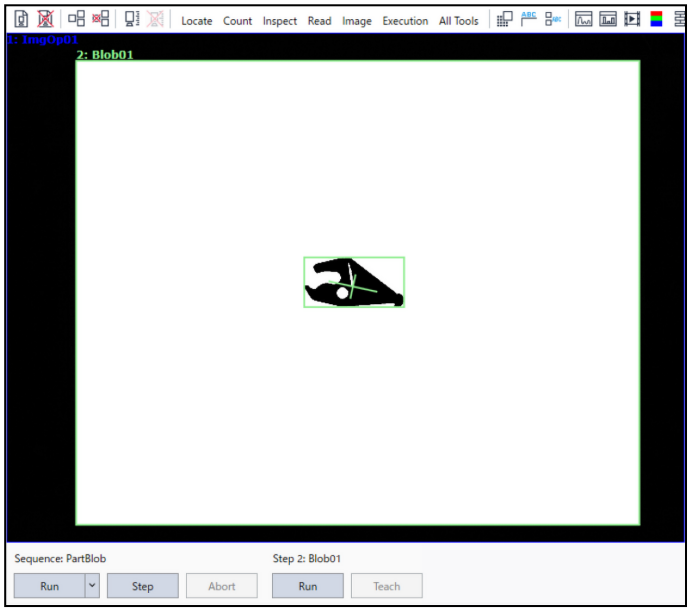


11. 放置零件在1個托盤上，按下Vision Guide工具欄上的[直方圖]。

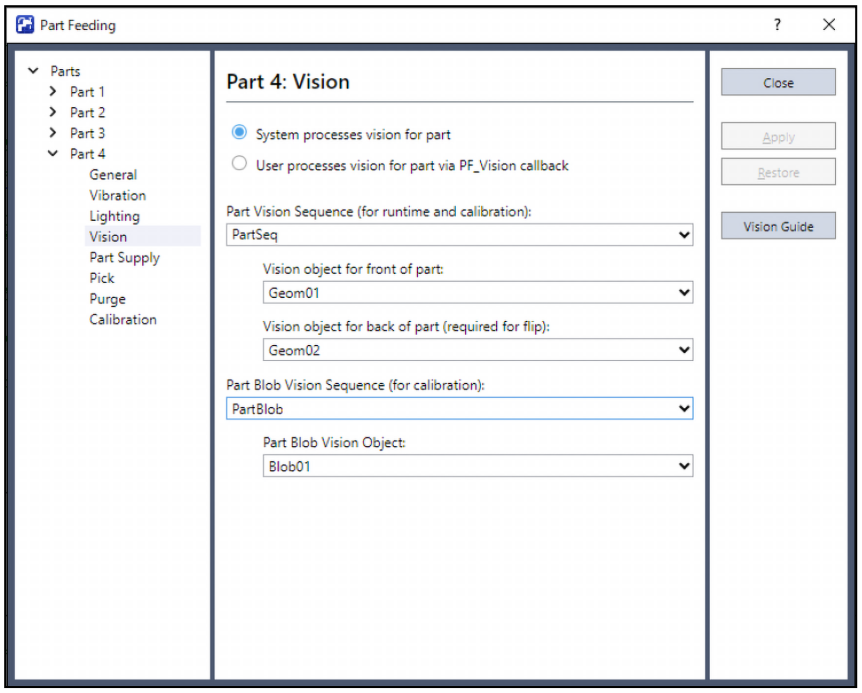


拖動直方圖上的紅色ThresholdHigh條，直到零件適當二值化。根據需要點擊[更新]，完成後點擊直方圖視窗的[OK]按鈕。

如此一來，當PartBlob序列執行時，送料器的影像會完全被移除。零件顯示為完全白色背景上的黑色物體。送料器被有效地從PartBlob序列中遮罩。



12. 在Epson RC+ 8.0功能表 - [工具] - [料件送料]視覺頁面中，將「PartBlob」設定為所需零件的零件Blob視覺序列。



無需特殊程式碼。

範例程式碼

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
```

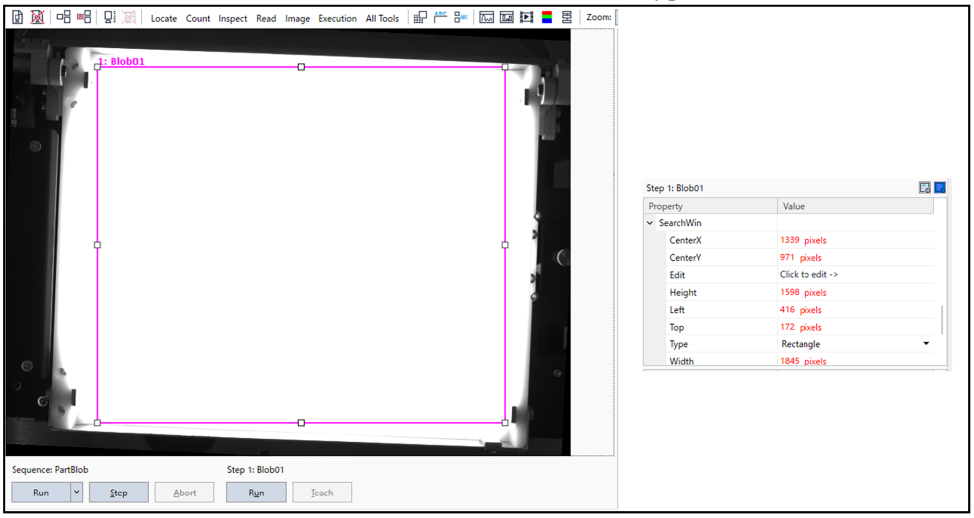
3.9.8.2 程式範例 8.2

範例類型：
在零件Blob視覺序列的SearchWinType中使用RotatedRectangle

配置

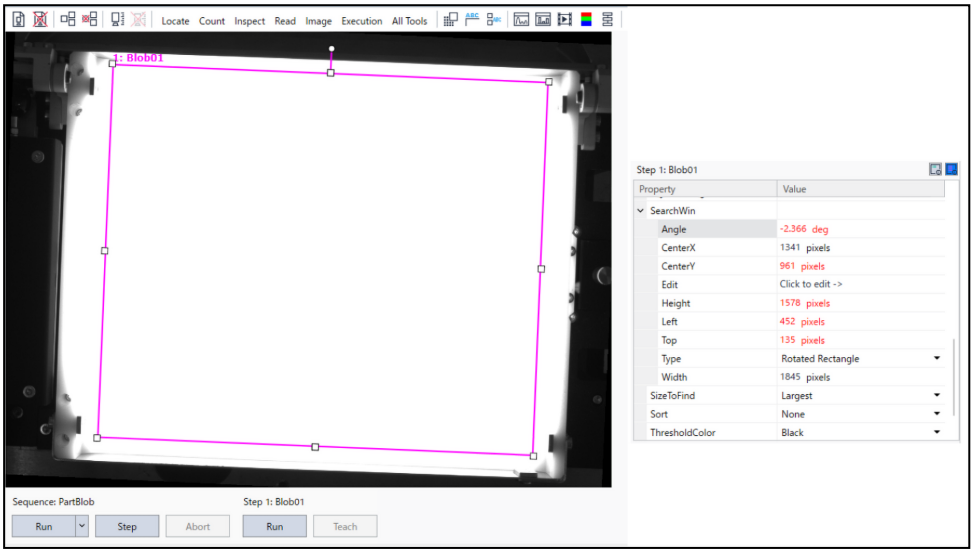
- 機器人數量：1
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：1
- 攝影機的朝向：送料器#1上的朝下固定的攝影機

說明
當攝影機視野與送料器托盤不互相平行時，可能難以適當設定PartBlob的搜尋視窗大小。如果搜尋視窗過大，PartBlob可能將托盤檢測為零件。如果搜尋視窗過小，系統無法適當判斷托盤上零件的數量和分布。在EPSON RC+ 7.5.2及更新的版本中，PartBlob物件可以在SearchWinType屬性中使用「RotatedSearchWin」。



以下是SearchWinType設為「Rectangle」時的例子。

當Part Blob的SearchWinType設為「RotatedRectangle」時，可以將攝影機視野與送料器托盤對齊。



提示

SearchWin Angle屬性應設為+/- 45°以內。

無需特殊程式碼。

範例程式碼

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 程式範例 8.3

範例類型：
在零件Blob視覺序列的搜尋視窗中使用檢測遮罩

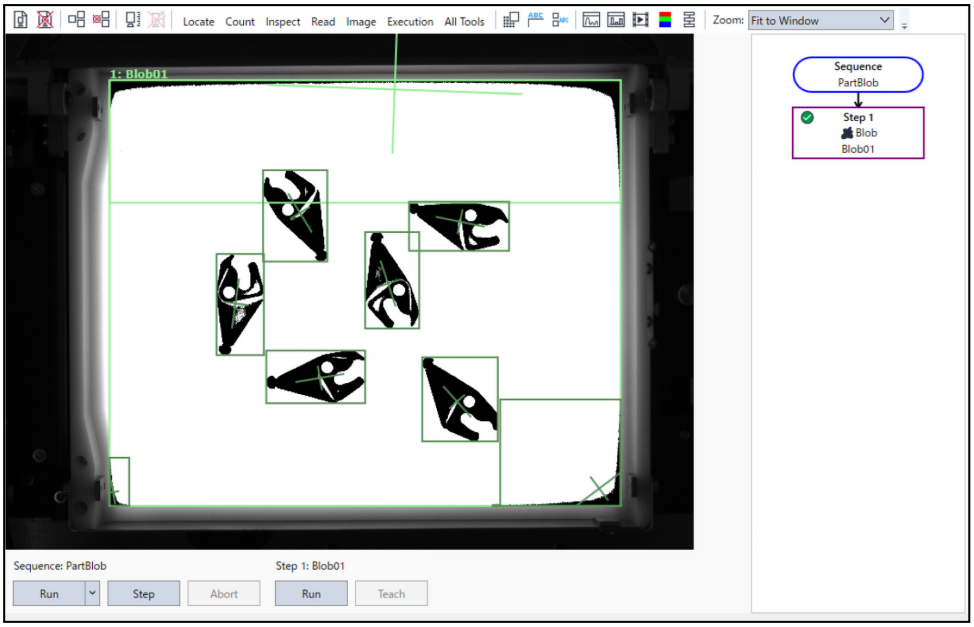
配置

- 機器人數量：1
- 送料器數量：1
- 送料器上的零件種類數：1
- 放置位置數量：1
- 攝影機的朝向：送料器#1上的朝下固定的攝影機

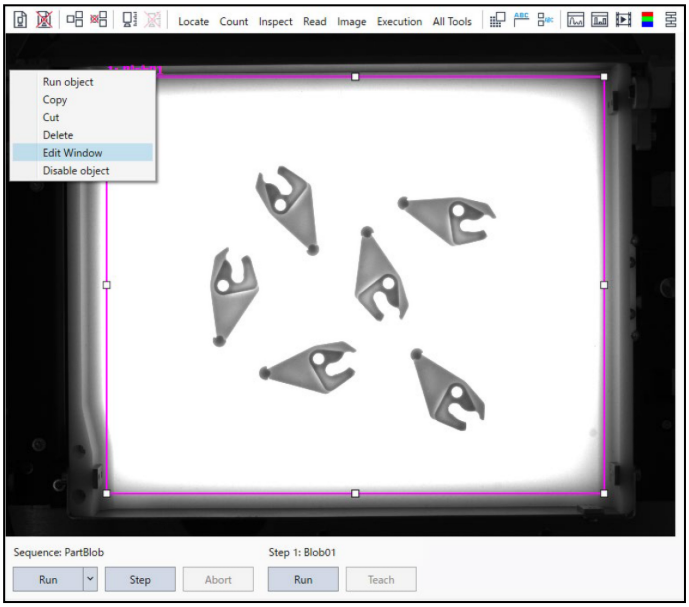
描述

即使開啟背光，托盤角落有時也會形成陰影。如果Blob物件的搜尋視窗包含托盤角落，且閾值未適當調整，陰影可能被誤認為零件。零件Blob視覺序列用於檢測個別零件或零件堆積。若Blob將陰影視為零件，系統可能對振動方法做出錯誤判斷。

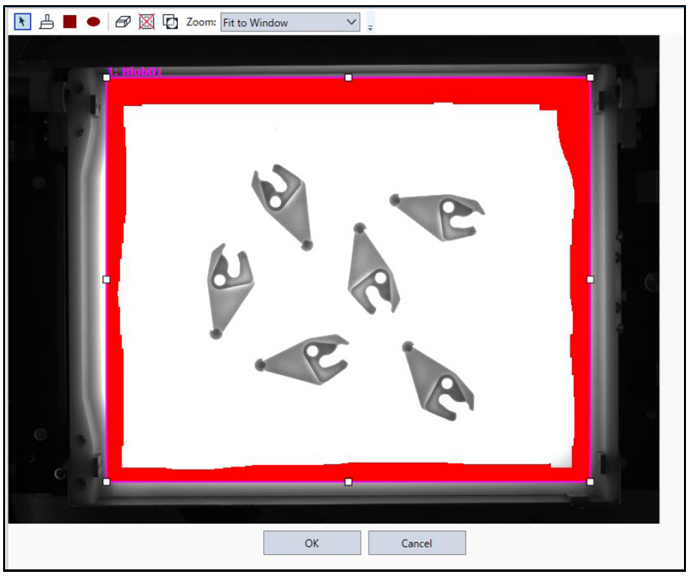
以下是托盤角落的陰影可能被誤認為零件的例子。



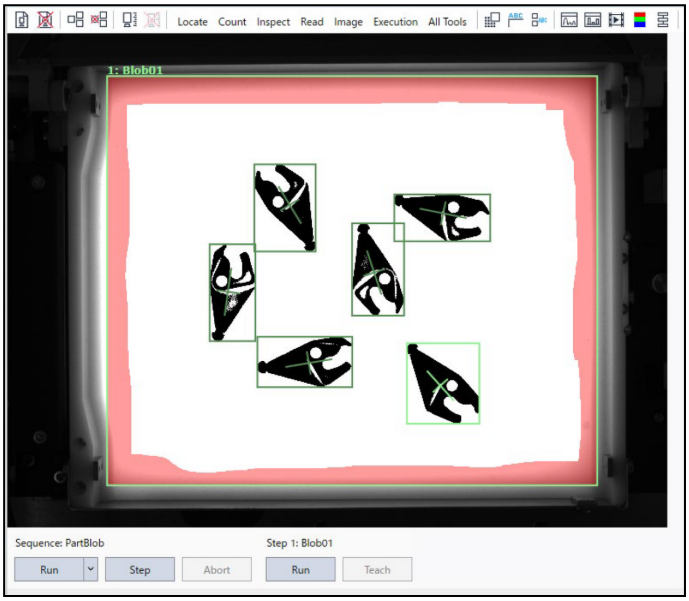
可以在Blob的搜尋視窗中，為不想識別的區域（在此情況下是托盤角落）設定檢測遮罩。
右鍵點擊Blob，從彈出功能表選擇[編輯視窗]，或選擇Blob的SearchWin的[編輯視窗]屬性。



使用畫筆工具繪製，使托盤從搜尋視窗中移除。



這樣，執行Blob時，托盤將不再包含在搜尋視窗中（參閱以下內容）。



無需特殊程式碼。

範例程式碼

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. 發展篇

4.1 多零件 & 多機器人

本章節將說明如何在同一個送料器上同時使用多種零件。此外，也將說明在處理零件群組時，如何讓多台機器人安全地存取同一個送料器。

4.1.1 多零件 & 多機器人的規格和需求

- 1台機器人控制器最多可使用4台零件送料器。
- 零件送料器無法在機器人控制器之間共享。
  例如，2台T/VT系列的機器人無法共享同一個送料器。若要使用多台機器人控制1台送料器，則必須使用可單台支援多台機器人的機器人控制器。
- 而在單1個系統中，則可任意搭配送料器型號。
  例如，可採取各使用1台IF-80、IF-240、IF-380、IF-530之類的各種搭配。
- 同一個送料器上最多可同時傳送4種零件。
  要在同一個送料器上同時傳送多種零件時，這些零件必須具有相似的物理特性。也就是說，其重量必須相同、尺寸和材質類似、且面積大致相同。各零件會使用其固有的振動參數。因此必須確保零件的振動參數不會導致其他零件飛出送料器。
- 最多可讓2台機器人同時使用同一個送料器。
  針對零件群組執行PF_Start（PF_Start 1, 2, 5），並將零件分配給2台以上的機器人時，將會發生錯誤。
- 執行PF_Start時，單一的零件或零件群組會被傳送到送料器上。
  針對分配到相同送料器編號的零件多次執行PF_Start時，將會顯示「送料器已在使用中」的訊息（由PF_Status回呼發出），而不會執行第2次的PF_Start。
- 要針對零件的群組執行PF_Start時PF_Start（PF_Start 1, 2, 5），所有零件必須分配到相同的送料器編號。
- PF_Start會將單一的零件或零件群組傳送到送料器上。
  固有的控制器任務編號將會被分配到運作中的各個送料器。（詳見下表）
  在使用者的應用程式程式碼中，請勿使用保留給送料器的任務。未使用送料器編號時，則可在使用者的程式碼中使用該任務。使用PF_InitLog陳述式擷取Part Feeding資料的日誌時，特定的計時器編號會保留給系統使用，因此無法在使用者的程式碼中使用。使用送料器控制命令（PF_Center、PF_ConterByShift、PF_Flip、PF_Shift）時，特定的SyncLock編號會保留給系統使用，因此無法在使用者的程式碼中使用。任務編號、計時器編號、SyncLock編號都是送料器編號固有的。（詳見下表）

送料器編號	任務	計時器	SyncLock
1	32	63	63
2	31	62	62
3	30	61	61
4	29	60	60

- PF_Start陳述式中記載的第1個零件，是初始的「主動零件」（最初要請求的零件）。下一個要請求的零件，則可使用PF_ActivePart陳述式來選擇。關於PF_ActivePart的資訊，將會在本章節中詳細說明。
- 送料動作（振動、拾取區域、供應方式）是針對目前的主動零件執行的。當PF_Start陳述式中記載的所有零件被選為主動零件時，將會使用各自的設定。例如，若應用場合需要，PF_Start群組中的各零件可能會有不同的拾取區域。

- 傳遞給各回呼函數的零件ID（PartID），則為目前的主動零件（目前請求的零件）之零件編號。
- 系統會為PF_Start陳述式中記載的每個零件擷取影像，並將各零件的佇列與視覺影像的結果一同載入。各零件則會使用各自的視覺設定和照明設定。例如，PF_Start列表中的某個零件可能使用「system processes vision」，而另一個零件使用「user processes vision」，所有零件會使用各自的設定。同樣地，每個零件可以使用自己的照明條件，例如前側光、無背光、不同的亮度等。
- Part Feeding日誌檔（由PF_InitLog陳述式啟動）中，將包含1台送料器上的所有零件的資料。
- 要讓多台機器人存取同一個送料器時，使用者的應用程式程式碼必須使用PF_AccessFeeder和PF_ReleaseFeeder陳述式，藉此確保機器人不會相互碰撞。這將在本章詳細說明。
- 所有零件都必須個別校準。並且預設同時傳送在1台送料器上的零件具有相似的特性（尺寸、重量、材質等）。

4.1.2 多零件 & 多機器人的主要概念

4.1.2.1 PF_ActivePart

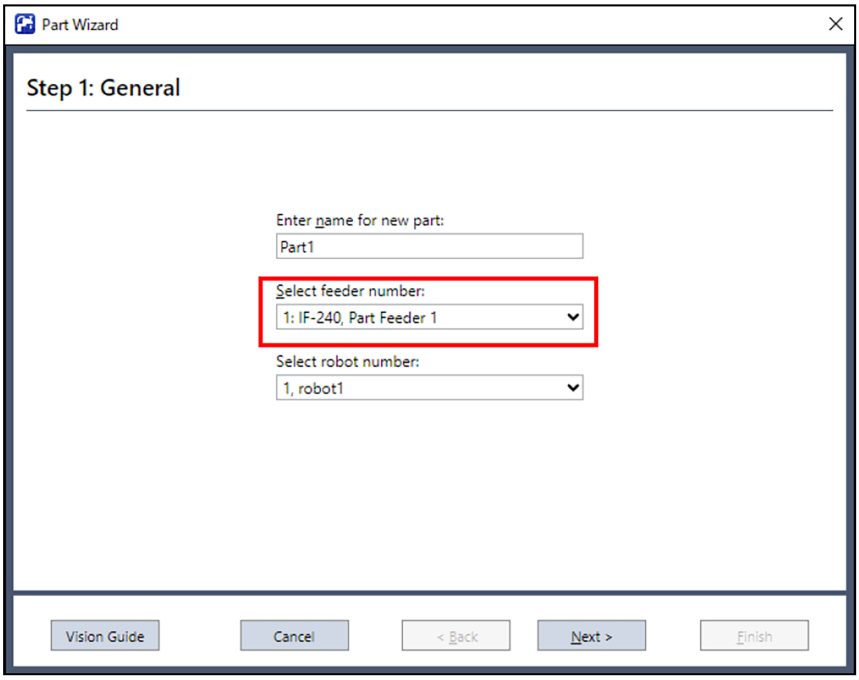
PF_ActivePart陳述式在執行時會向Part Feeding處理程序告知使用者的意圖。系統則必須了解目前請求的是哪一個零件。而為了確保請求的零件可以使用，系統將會投入零件。(使用適當的振動設定，透過料斗供應零件等)。

以下將簡單說明為何執行時需要PF_ActivePart。此處是以所有零件都交由同一個零件送料器投入的「套件組裝」之應用作為說明範例。空箱子會被放在輸送帶上，分發給機器人。箱子的條碼上顯示著要放入箱子的零件類型和數量。因此在箱子被條碼讀取器讀取之前，並無法得知需要哪種零件和拾取數量。送料器將會振動，搜尋所需的零件並完成命令。

接著再另以另一個使用PF_ActivePart的例子來說明2種零件的組裝操作。在同一個零件送料器上會有2種零件。在此應用場合中，機器人必須拾取1個零件#1，將其放入固定裝置中。接著，機器人必須拾取2個零件#2並將其插入到零件#1。對此，PF_ActivePart可用來指示系統需要投入哪個零件才能完成組裝程序。為了讓此應用更貼近現實，在此先假定要在組裝前使用視覺系統檢查每個零件。若零件#1未通過檢查，PF_ActivePart將保持在零件#1的階段，藉此確保機器人能回到送料器並取得合適的零件#1。若零件#1通過檢查，PF_ActivePart將會切換到零件#2的階段。
PF_Start陳述式中的第1個零件ID，是最初的主動零件。PF_ActivePart通常會被設定在PF_Robot回呼結束前，專門用於送料動作所請求的零件。
關於PF_ActivePart的資訊，將會在本章節中詳細說明。

4.1.2.2 PF_Start

如先前所述，同一個送料器上最多可同時處理4種零件。此配置可透過指定各零件ID的PF_Start陳述式的執行時機來實現。
例如：假設零件1、2、3、4都使用送料器#1。送料器編號則要在首次建立各零件時透過[零件精靈]指定。



只有分配到同一個送料器的零件才能透過PF_Start同時傳送。若要在送料器#1中同時傳送所有4種零件，則採用下列程式碼：

```
PF_Start 1, 2, 3, 4
```

在此範例中，零件#1是初始的主動零件。送料器將在最一開始使用零件#1的振動參數。在此例中，直到PF_ActivePart變更為另一個零件ID之前，值「1」都將會作為各回呼函數的PartID參數送出。

提示

直到以另一個零件ID執行PF_ActivePart之前，PF_Start列表中的第1個零件都會是主動零件。

若使用者程式碼嘗試在同一個送料器上啟動另一個零件，則會發生「送料器使用中」的錯誤。錯誤將會使用PF_Status回呼處理，狀態值為常數PF_D_STATUS_FEEDERINUSE_ERROR。
以下是發生「送料器使用中」錯誤的例子：

零件1、2、3、4都正在使用送料器#1
PF_Start 1, 3, 4　　'在送料器#1啟動零件分組

在執行下列程式碼時，發生了「送料器使用中」錯誤

```
PF_Start 2
```

若要在送料器#1中同時傳送所有4種零件，則必須執行下列陳述式：

```
PF_Start 1, 2, 3, 4
```

最多可讓2台機器人同時使用同一個送料器。若PF_Start嘗試將有2台以上之機器人同時使用的零件傳送在同一個送料器上，則會發生「達到每個送料器的最大機器人數」的錯誤。
以下為發生錯誤的例子：

零件1：使用機器人#1和送料器#1
零件3：使用機器人#2和送料器#1

零件5：使用機器人#3和送料器#1
PF_Start 1,3,5 <- Error 7731超過控制器類型的最大同時送料器數。

4.1.2.3 視覺系統和佇列的載入

在預設狀態下，系統會為傳送在同一個送料器上的所有零件擷取視覺影像。也有僅為主動零件擷取影像的機制（將在下一節說明）。而各個零件將會使用各自的照明需求（前側光/背光、移動式攝影機、使用PF_Vision回呼等）。

當所有的零件視覺序列完成後，所有佇列會與視覺影像的結果一同載入。

TIP

為了提高效率，可針對所有零件視覺序列設定相同的CameraBrightness和CameraContrast。在PF_Start列表中第一個零件的視覺序列裡，將RuntimeAcquire屬性設為「Stationary」。並於使用相同送料器的其餘零件視覺序列中，將RuntimeAcquire設為「None」。如此一來即可減少多餘的影像擷取時間。不過，此方法是以所有零件都能在相同照明條件下被搜尋到的狀態作為前提。

4.1.2.4 PF_Robot回傳值

所有的回呼函數都需要回傳數值。回傳值通常是代表操作已正確完成。而（常數PF_CALLBACK_SUCCESS）回傳值，則可被用於變更系統的運作方式。PF_Robot回傳值可用於變更主要的程序，並根據「使用情境」提供不同的動作。開發者能透過不同的回傳值，在各種狀況下控制Part Feeding的處理程序。

以下為PF_Robot回呼函數的各種回傳值之說明。

PF_CALLBACK_SUCCESS:
當PF_Robot回呼函數結束，且「PF_Robot = PF_CALLBACK_SUCCESS」時，系統將會確認該佇列中是否可使用下一個主動零件（請求的零件）。若可使用主動零件，系統將會再次調用PF_Robot函數。傳遞給PF_Robot的PartID參數將會是該主動零件的零件編號。在主動零件可以使用時，由於請求的零件已列入佇列中，因此不需要擷取新影像或讓送料器進行振動。在主動零件無法使用時，系統將會為PF_Start陳述式中執行的所有零件擷取新影像。視覺影像的結果會被載入至各零件佇列中，並由系統判斷是否要讓送料器進行振動。

PF_CALLBACK_RESTART:
當PF_Robot回呼函數結束，且「PF_Robot = PF_CALLBACK_RESTART」時，無論主動零件的佇列中是否有可使用的零件，系統都會為PF_Start陳述式中執行的所有零件擷取新影像。視覺影像的結果會被載入至所有零件佇列中，並由系統判斷是否要讓送料器進行振動。PF_CALLBACK_RESTART回傳值的主要目的，在於強制從主要Part Feeding處理程序迴圈的一開始擷取新影像、重新載入佇列、重新判斷、重新投入。當需要在每個取放循環中擷取新影像時，可有效活用此回傳值。例如，若在取放過程中意外分散了周圍的零件，用來擷取新影像並更新佇列時會很方便。

PF_CALLBACK_RESTART_ACTIVEPART:
當PF_Robot回呼函數結束，且「PF_Robot = PF_CALLBACK_RESTART_ACTIVEPART」時，系統僅會為主動零件（並非PF_Start陳述式中記載的所有零件）擷取新影像。此回傳值在讓多台機器人共享同一個送料器時特別能發揮作用。使用PF_CALLBACK_RESTART_ACTIVEPART，即可防止零件重複出現在多個佇列中。此回傳值將會強制系統僅為主動零件擷取新影像，且僅載入主動零件的佇列。PF_Start陳述式中記載的所有其他零件之佇列將會被清除。以下將舉出一個例子來說明這一點：

假設送料器平台上有1種實體零件，且交由2台機器人從送料器進行拾取。在此應用場合中，就會在Part Feeding對話方塊中新增2種零件。各個零件將會被分配不同的機器人編號。此外，各個零件的視覺序列應具備不同的視覺校準。（因為攝影機會針對特定的機器人進行校準）即使實體零件種類只有1種，也需要建立2個邏輯部分。（每個機器人各1個）PF_Start陳述式之中包含兩者的零件編號。系統會為兩者的邏輯部分擷取視覺影像，並載入兩者的佇列。由於兩

個視覺序列都在平台上尋找相同的實體零件，佇列中將會發生座標重複。當機器人#1拾取零件並從佇列中移除零件後，該零件仍然保留在另一個機器人的佇列中。導致機器人#2嘗試拾取已被機器人#1移除的零件。而PF_CALLBACK_RESTART_ACTIVEPART可確保零件只載入到1個佇列中。因此以結果而言，並不需要採取特別的排序或特別的佇列分配。關於此使用情境的詳細程式設計資訊，請參閱以下內容。
機器人2台 - 零件1種類

4.1.2.5 PF_AccessFeeder / PF_ReleaseFeeder

PF_AccessFeeder / PF_ReleaseFeeder可透過鎖定和解除鎖定針對送料器的存取，防止多機器人單一送料器系統中的碰撞。在有2台機器人同時共享同一個送料器時，則必須使用這些命令。PF_AccessFeeder是互斥鎖。若已取得互斥鎖，PF_AccessFeeder會暫停任務（待命至輪到自己），直到鎖定解除或是達到指定的時間（可選項目）。必須使用PF_ReleaseFeeder陳述式來解除鎖定，確保在機器人完成使用送料器後，能夠讓另一個機器人使用送料器。
PF_ReleaseFeeder可在動作命令的「!...!」平行處理陳述式中使用。此做法能在一台機器人離開送料器時，讓另一台機器人接近送料器。
防止機器人在存取送料器時發生碰撞的程式碼如以下所示：

```
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_Stop陳述式中包含作為參數的PartID。在單一零件系統中，PartID與傳送中的零件相同。在多零件系統中，PartID是傳送在送料器上的任一個零件之零件編號。也就是說，可以指定PF_Start列表中的任何零件。但請注意，傳遞給PF_CycleStop回呼函數的PartID，將會是目前主動零件的零件ID。

4.1.2.7 PF_InitLog

PF_InitLog將會執行Part Feeding日誌檔。PF_Start陳述式中列出的所有零件資料都會記錄在該檔案中。日誌檔的各個項目中包含目前的主動零件之零件ID。例如，會將PF_Robot回呼內所處理的目前的主動零件之零件數量記錄下來。如需詳細資訊，請參閱以下內容。
Part Feeding日誌檔

4.1.2.8 PF_QtyAdjHopperTime

PF_QtyAdjHopperTime函數，將會計算出提供最佳數量之零件所需的料斗運作時間。該時間則會根據特定時間內供應的零件數量、目前送料器平台上的大致零件數量、以及在零件其「零件區域」校準中計算出的「最佳零件數量」值來進行計算。PF_QtyAdjHopperTime函數可在使用者程式碼的任何位置執行。例如，可設定在PF_Start陳述式前執行，確保在運作前供應零件。Part Feeding系統並無法判斷哪一個零件群組將使用送料器。有可能是讓單1種零件傳送在送料器上，也有可能會讓4種不同零件同時傳送在送料器上。若在PF_Start之前執行PF_QtyAdjHopperTime，在計算上就只能先假設由PartID參數指定的零件是平台上唯一的零件。並且只使用（該PartID的）Part Blob視覺序列。若在PF_Start之後執行PF_QtyAdjHopperTime，Part Feeding系統就能了解送料器上有哪些零件。（因為已在PF_Start陳述式中指定零件。）此時，提供之PartID的Part Blob視覺序列會與各個零件序列一起執行。當多個零件同時傳送時，系統會將各類型零件保持相同數量視為最佳選項。但此判斷可能並不適用於部分情況。例如，希望將零件#2的拾取數量設為零件#1的2倍的時候。於此情況下，則需透過使用者的程式碼按比例調整（乘以或除以）PF_QtyAdjHopperTime函數所回傳的計算時間，確保從料斗供應所需數量的零件。

4.1.3 教學

本節將具體說明如何安裝多零件的應用程式和多零件、多機器人的應用程式。但僅針對大多應用案例簡單說明步驟，不詳細說明執行內容。本章節內容，則是以使用者已了解如何建立新零件、執行零件校準、以及如何編寫1台機器人／1種零件的應用程式為前提所提出的說明。
如需詳細資訊，請參閱以下內容。
開始使用

4.1.3.1 教學1：1台機器人、1台送料器、2種零件

本教學範例中，具有2種不同的零件同時傳送在同一個送料器上。因此零件會有不同的物理性質。此處使用的是零件#1和零件#2。並且有在送料器上方設置朝下固定的攝影機。機器人將會取放2個零件#1，接著再取放1個零件#2。並以迴圈持續執行此操作。

各個零件將使用不同的末端夾具（輸出位元）拾取。所以在此組裝應用程式中，零件數量和拾取順序非常重要。零件#1和零件#2之間的處理程序切換，則使用PF_ActivePart執行。首先，要在PF_Robot回呼函數內編寫執行機器人動作的程式碼。接著，透過個別的多工處理程序執行動作、平行處理送料器的振動，藉此提升機器人的總處理量。

1. 建立Epson RC+的新專案。

2. 對朝下固定的攝影機進行校準。

3. 透過Vision Guide建立Part Blob序列。
   將「Part Blob序列」用於零件校準期間和執行時的回饋，藉此決定零件的最佳投入方法。Part Blob序列會在執行時檢測各個零件或零件群組。通常Part Blob序列當中會包含單一的Blob視覺物件。Part Blob序列則需要已分配給Calibration屬性的攝影機校準。
   將Blob物件的NumberToFind屬性設定為「All」。將Blob的ThresholdAuto屬性設定為「False」（預設值）。
   在平台上放置零件，開啟[直方圖]視窗，並持續調整ThresholdHigh和ThresholdLow直到檢測到零件。將Blob的MinArea設為零件區域之MinArea的0.9倍左右。若零件有正反面之別，則應設定MinArea，確保零件朝向任何方向都能被檢測到。（送料器上的）2種零件可各自設定個別的Part Blob序列，但通常可針對所有零件使用相同的Part Blob序列。
   確認Part Blob序列能檢測到傳送在此送料器上的每種零件。Blob物件的搜尋視窗必須盡可能地覆蓋到廣泛的平台區域。但重點在於讓Blob物件只檢測到零件，而不會檢測到送料器托盤本身。若Part Blob序列檢測到托盤的一部分，將使得系統無法正常運作。

4. 透過Vision Guide為此送料器運作的2種零件各自建立零件序列。
   並確認各個序列都有設定Calibration屬性。一般而言，Geometric物件會在需要識別正面時使用（第2個Geometric物件則是在需要翻轉以識別零件的背面時才會使用）。而視覺物件的NumberToFind屬性，則設定為「All」。

5. 顯示[工具] - [料件送料]，並依照「零件精靈」的操作指示新增新零件。
   如需零件精靈的詳細資訊，請參閱以下內容。
   開始使用

6. 點擊新零件的[校準] - [校準]按鈕，以開始「校準精靈」。
   如需校準精靈的詳細使用方法，請參閱以下內容。
   校準&測試

7. 點擊零件#1的[拾取] - [教導]按鈕。
   使用點動操作將機器人移動至零件的拾取高度後，針對「Pick Z」進行教導。

8. 點擊[料件送料] - [增加]按鈕，以新增零件#2。
   使用「零件精靈」設定零件。

9. 點擊[校準] - [校準]按鈕，以開始校準精靈。

10. 點擊零件#2的[拾取] - [教導]按鈕。
    使用點動操作將機器人移動至零件的拾取高度後，針對「Pick Z」進行教導。

11. 關閉[料件送料]對話方塊。
    料件送料的範本程式碼（料件送料的回呼函數）將會自動建立。

12. 接著教導機器人停止（park）和放置（place）的點位，將其分別命名為「park」和「place」。

13. 依照下列內容針對範本程式碼進行修正：

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
```

上述的範例程式碼當中，送料器會在機器人完成「放置」動作以及PF_Robot函數結束之後，視需要進行振動。而這段程式碼還能再進一步重組，讓送料器的振動與機器人動作一併執行。
並且使用PF_Robot回呼函數（例如：Function Main）來通知動作任務何時可以拾取。
當零件可使用時，則使用記憶體IO（命名為「PartsToPick1」、「PartsToPick2」）發送訊號。
當最後一個可使用的零件被放置時（在此範例中為動作完成80%的狀態），動作任務將會向PF_Robot函數發送訊號並結束，接著回傳數值。此回傳值，能讓系統得知已可擷取新影像、進行振動、以及透過料斗供應零件等。此處則再修改教學範例的程式碼，讓送料器動作和機器人動作能夠一併執行：

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 教學2：2台機器人、1台送料器、2種零件

本教學範例中，具有2種不同的零件同時傳送在同一個送料器上。因此零件會有不同的物理性質。此處使用的是零件#1和零件#2。並且有在送料器上方設置朝下固定的攝影機。2台機器人將共享同一個送料器。2台機器人會交替從送料器上拾取。各個機器人將會取放各自的零件。機器人#1會先拾取1個零件#1，機器人#2則接著拾取1個零件#2。
在此應用場合中，拾取的順序很重要。拾取順序則可透過「PF_ActivePart」來切換。首先，要在PF_Robot回呼函數內編寫執行機器人動作的程式碼。在此應用案例中，機器人動作將會依序進行。一次只會有1台機器人行動。
接著，在個別的多工處理程序中執行動作，以提升機器人的總處理量。修改後的教學範例中也包含PF_AccessFeeder / PF_ReleaseFeeder。PF_AccessFeeder / PF_ReleaseFeeder可用來防止機器人從送料器拾取時發生碰撞。

1. 建立Epson RC+的新專案。
2. 對用於機器人#1的朝下固定的攝影機進行校準。
3. 對用於機器人#2的朝下固定的攝影機進行校準。
4. 從Vision Guide建立零件#1的Part Blob序列。通常Part Blob序列當中會包含單一的Blob視覺物件。序列需要對已分配給Calibration屬性的機器人#1執行的攝影機校準。將Blob物件的NumberToFind屬性設定為「All」。
   將Blob的ThresholdAuto屬性設定為「False」（預設值）。在平台上放置零件，開啟[直方圖]視窗，並持續調整ThresholdHigh和ThresholdLow直到檢測到零件。將Blob的MinArea設為零件區域之MinArea的0.9倍左右。
   若零件有正反面之別，則應設定MinArea，確保零件朝向任何方向都能被檢測到。確認Part Blob序列能檢測到零件#1。
   Blob物件的搜尋視窗必須盡可能地覆蓋到廣泛的平台區域。但重點在於讓Blob物件只檢測到零件，而不會檢測到送料器托盤本身。
5. 從Vision Guide建立零件#2的Part Blob序列。通常Part Blob序列當中會包含單一的Blob視覺物件。序列需要對已分配給Calibration屬性的機器人#2執行的攝影機校準。將Blob物件的NumberToFind屬性設定為「All」。
   將Blob的ThresholdAuto屬性設定為「False」（預設值）。
   在平台上放置零件，開啟[直方圖]視窗，並持續調整ThresholdHigh和ThresholdLow直到檢測到零件。

將Blob的MinArea設為零件區域之MinArea的0.9倍左右。
若零件有正反面之別，則應設定MinArea，確保零件朝向任何方向都能被檢測到。確認Part Blob序列能檢測到零件#2。Blob物件的搜尋視窗必須盡可能地覆蓋到廣泛的平台區域。
但重點在於讓Blob物件只檢測到零件，而不會檢測到送料器托盤本身。

6. 透過Vision Guide建立用於檢測零件#1的零件序列。
   請確認Calibration屬性設定為針對機器人#1執行的校準。
   一般而言，Geometric物件會在需要識別正面時使用。（第2個Geometric物件則是在需要翻轉以識別零件的背面時才會使用。）將視覺物件的NumberToFind屬性設定為「All」。
7. 透過Vision Guide建立用於檢測零件#2的零件序列。
   請確認Calibration屬性設定為針對機器人#2執行的校準。
   一般而言，Geometric物件會在需要識別正面時使用。（第2個Geometric物件則是在需要翻轉以識別零件的背面時才會使用。）將視覺物件的NumberToFind屬性設定為「All」。
8. 顯示[工具] - [料件送料]，新增零件#1的新零件。依照「零件精靈」的操作指示新增零件。確認在零件精靈的第一頁已選擇機器人#1。
   如需零件精靈的詳細資訊，請參閱以下內容。
   開始使用
9. 點擊新零件#1的[校準] - [校準]按鈕，以開始「校準精靈」。
   如需校準精靈的詳細使用方法，請參閱以下內容。
   校準&測試
10. 點擊零件#1的[拾取] - [教導]按鈕。
    使用點動操作將機器人#1移動至零件的拾取高度後，針對「Pick Z」進行教導。
11. 點擊[料件送料] - [增加]按鈕，以新增零件#2。使用「零件精靈」設定零件。
    確認在零件精靈的第一頁已選擇機器人#2。
12. 點擊[校準] - [校準]按鈕，以開始校準精靈。
13. 點擊零件#2的[拾取] - [教導]按鈕。使用點動操作將機器人#2移動至零件的拾取高度後，針對「Pick Z」進行教導。
14. 關閉[料件送料]對話方塊。
    零件送料的範本程式碼（Part Feeding的回呼函數）將會自動建立。
15. 依照下列內容針對範本程式碼進行修正：

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
```

此處將會修正範例程式碼，讓機器人動作在另外的多工處理程序中執行。當1台機器人離開送料器時，另一台機器人將可以開始朝送料器移動。要讓多台機器人使用並行動作共享1台送料器時，使用PF_AccessFeeder命令和PF_ReleaseFeeder命令來防止機器人碰撞非常重要。
此外，修改後的程式碼可平行處理送料器振動和機器人動作。在本教學中，當各機器人達到其放置位置的80%之處時，機器人將會清除攝影機的視野，並且可擷取影像。
此外，當各機器人達到其放置位置的80%之處時，另一台機器人可以安全地開始朝送料器移動。實際動作的百分比取決於機器人在特定情況下的速度和相對定位。各機器人都有「park」點和「place」點。在本教學範例中，機器人#1是以右手姿態從送料器拾取，機器人#2則是以左手姿態從送料器拾取。
修改後的範本程式碼則如以下所示：

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 多零件 & 多機器人的總結

在此將介紹多零件、多機器人零件送料器應用場合所需的主要概念。

- **使用PF_Start**，即可指定最多4種零件同時傳送在同一個送料器上。
- **使用PF_ActivePart**，即可選擇目前需要的零件。系統將會進行振動，確保請求的零件可使用。

- 最多可讓2台機器人同時存取同一個送料器。
  PF_AccessFeeder和**PF_ReleaseFeeder**可確保兩邊的機器人能安全地從送料器拾取零件而不發生碰撞。
- **PF_Robot回傳值**可控制主要的處理程序。
  以下的回傳值會提供各類功能：
  PF_CALLBACK_SUCCESS
  若PF_ActivePart（請求的零件）可使用，系統將會再次調用PF_Robot函數，不會再重新擷取影像或重新載入零件佇列。若PF_ActivePart無法使用，系統將會為各零件擷取新影像，並重新載入零件佇列。
  PF_CALLBACK_RESTART
  系統會為PF_Start陳述式中執行的各零件擷取新影像，並重新載入所有零件佇列。
  PF_CALLBACK_RESTART_ACTIVEPART
  系統僅為PF_ActivePart擷取新影像。PF_ActivePart的佇列與視覺影像的結果將會一同載入，而其他所有的零件佇列將會被清除。

4.2 平台類型

本章將會說明平台的類型。

4.2.1 標準平台的類型

IF-80、IF-240、IF-380、IF-530的標準平台一共有3種：平面（Flat）、防黏著（Anti-stick）和防滾動（Anti-roll）。所有平台皆有白色和黑色可供選擇。
IF-A1520、IF-A2330的標準平台一共有2種：平面（Flat）和防滾動（Anti-roll）。兩種平台均為白色。

4.2.1.1 平台顏色

在大多數的應用場合中，使用送料器的內建背光可實現最佳的影像處理。要使用背光時，則使用白色半透明平台。背光會投影出零件的輪廓。
零件表面的特徵則無法透過視覺系統辨識。此外，若希望將對比度最大化，也可使用黑色平台和自訂前側光來達成。但一般而言，系統在設計上預設是與背光選配件一起使用的。透過亮起背光來檢測零件。（右上|上下、角方向等。）
透明或半透明的物體在使用背光時可能會有對比度不佳的狀況。此時針對透明零件使用非白色的背光，能夠有效取得較佳的對比度。

4.2.1.2 平台材質

可依據送料器型號採用防靜電（ESD）、或是醫療應用場合的特殊材料。如需詳細資訊，請參閱各給料器的硬體手冊。

4.2.1.3 標準平台的使用方法

- **平面：**
  具穩定姿態的零件可使用的平面平台。零件必須能在送料器振動之後的短時間內靜止。為了多品項小批量生產，大多數的應用場合會使用平面平台。

外觀

剖面圖

a	零件
b	平台

EPSON提供的平台符合以下表格中的平面度和平行度之規格，確保能夠維持拾取的精確度。

	IF-80	IF-240	IF-380 / IF-530	IF-A1520 / IF-A2330
表面的平面度 [mm]	0.1	0.2	0.6	0.3
表面與基準間的平行度 [mm]	0.1	0.5	0.6	0.6

- **防黏著：**
  防黏著平台具有狹窄的溝槽，可用於減少表面的接觸阻力。藉此可減少摩擦力，改善零件在送料器表面的移動性。因運動摩擦（滑動摩擦或動摩擦）而難以分散的零件，則相當適合使用防黏著平台來因應。

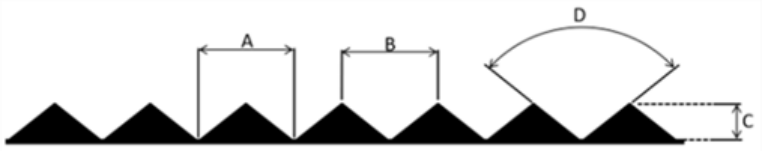
表面

剖面圖

a	零件
b	平台

	A	B	C	D
IF-80	0.4	0.4	0.2	90

小型零件用標準防黏著平台的結構



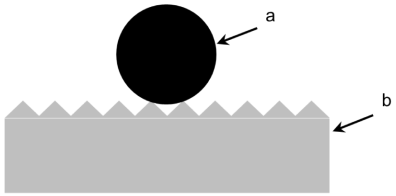
	A	B	C
IF-240	0.7	1.3	0.5

大型零件用標準防黏著平台的結構

- **防滾動：**

防滾動平台採用經過機械加工的結構面，能讓易滾動的零件靜止。在要供應圓柱形的料件時，使用防滾動平台特別有效。防滾動平台可防止零件滾動，藉此縮短穩定所需的時間。

表面

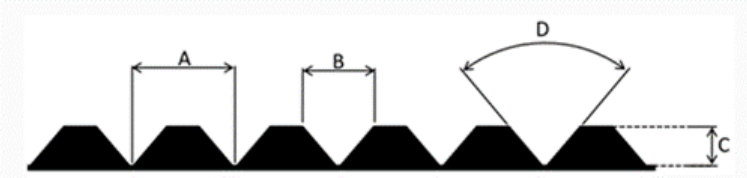


剖面圖

a	零件
b	平台

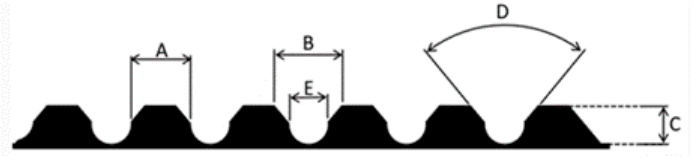
	A	B	C	D	適合的零件
IF-80	1.25	1	0.5	90	ø 0.7mm - ø 1.5mm
IF-80	2.75	2.5	1.25	90	ø 1.5mm - ø 3.5mm
IF-240	3	2.5	1.25	90	ø 1.7mm - ø 3.5mm
IF-240	5.5	5	2.5	90	ø 3.5mm - ø 7mm
IF-240	10.5	10	5	90	ø 7mm - ø 14mm
IF-380	3	2.5	0.722	120	ø 3mm - ø 5mm
IF-380	5.5	5	1.443	120	ø 5mm - ø 10mm
IF-530	6.5	6	1.732	120	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

小型零件用標準防滾動平台的結構



	A	B	C	D	E	適合的零件
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

大型零件用標準防滾動平台的結構



4.2.2 自訂平台

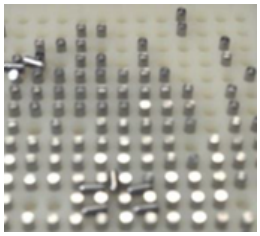
提示

自訂平台請由客戶自行設計及製作。

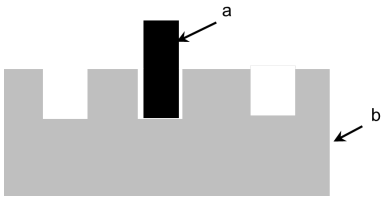
4.2.2.1 自訂平台的基本設計

自訂平台有3種基本設計：孔、溝槽和凹槽。自訂平台的目的在於透過孔、溝槽、凹槽排列零件，藉此達到所需的週期時間。當零件的自然靜止位置與拾取方向不一致時，則必須使用自訂平台，來確保零件能夠正確地配置。
以下為自訂平台的一般概要。

- **孔：**



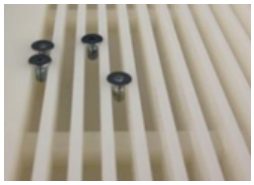
表面



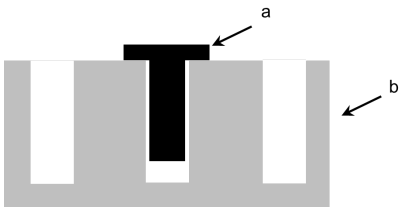
剖面圖

a	零件
b	平台

- **溝槽：**



表面

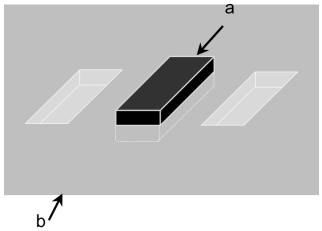


剖面圖

a	零件
b	平台

- **凹槽：**

表面



斜視圖

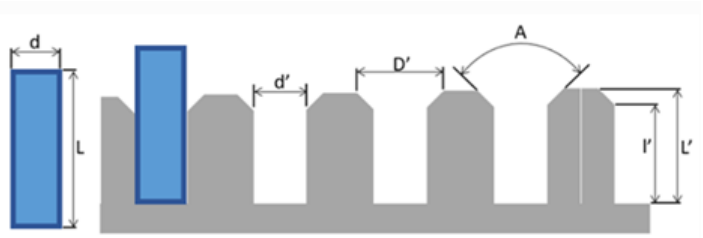
a	零件
b	平台

4.2.2.2 自訂平台的設計指南

- **孔：**
  在要供應圓柱形零件並讓其處在直立狀態時，很適合採用孔加工。
  而在設計帶孔的平台時，應考量以下事項：
  - 一般而言應採用簡單的結構。
  - 孔的直徑（d'）是平板上最重要的尺寸。
    直徑必須夠大才能讓零件直立在孔的內側。若零件的直徑不超過3.5mm，孔徑則應設定為最大零件直徑（d）加上0.05mm。但針對直徑超過3.5mm的大型零件或是圓錐形零件之類的非圓柱形零件，則必須採用更大的直徑。

- 孔必須具有足夠的深度（l'），在必要時能透過壁面來導引零件。
  若工件放置在底部，則不需要長的導引件。
- 建議在零件高度（L）的3分之1（可能的話則在2分之1）以上之處導引零件，讓零件盡量保持直立。
- 也請注意平板（L'）上零件的剩餘高度。
- 倒角（A）是用來將零件放入孔中的必備構造。
  在多數情況下，倒角角度會設為60° 。
- 平台的重量應與標準的平面平台相同。若重量有大幅變化，就可能會改變共振頻率並影響送料器的運作。於此情況下，可透過調整送料器的運作頻率來因應。

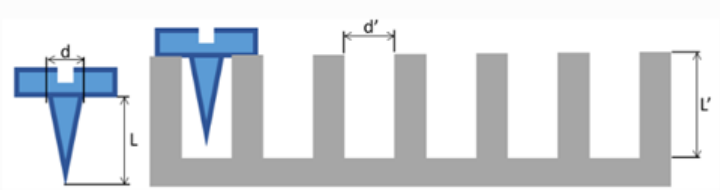
	D′	d′	l′	A
IF-80	>0.5*L	d+0.05mm	0.5*L	60
IF-240	>0.5*L	d+0.1mm	0.5*L	60
IF-380	>0.5*L	d+0.5mm	0.5*L	60
IF-530	>0.5*L	d+1.0mm	0.5*L	60
IF-A1520	>0.5*L	d+1.0mm	0.5*L	60
IF-A2330	>0.5*L	d+1.5mm	0.5*L	60



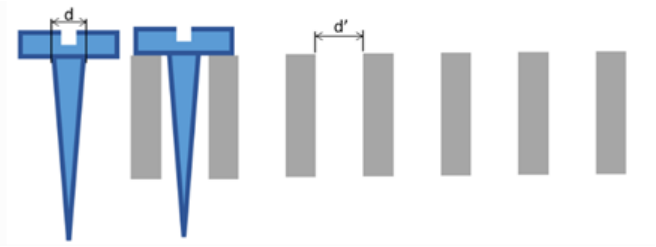
- **溝槽**

  在要供應需垂直安裝的螺絲類型零件時，會使用採用溝槽設計的平台。
  要設計帶溝槽的平台時，應考量以下項目：

  - 一般而言應採用簡單的結構。
  - 若使用無貫通溝槽的平台，則可供應最長60mm的零件。針對更長的零件，設計溝槽時則應考量平板的厚度。
  - 並根據零件的直徑（d）來設定溝槽寬度（d'）。一般而言，溝槽寬度會設定為零件最大直徑加0.05mm ~ 0.1mm（考量到容許誤差）。也請考量溝槽寬度的公差。此部分會因加工而異，且所有尺寸的送料器也並不相同。建議採用0/+的公差。
  - 由於工件不會放置在底部，在大多數情況下，溝槽深度（L'）並不是那麼重要。因此，也更是需要足夠大的空間，確保零件在傾斜時不會碰觸到底部。而公差的部分，則通常適用0/+的公差。



長度未超過60mm的大型零件不需要貫通式溝槽。



針對超過60mm的零件，請在設計貫通式溝槽時考量平板的厚度。

	d′	L′
IF-80	d+0.05mm	L'>L
IF-240	d+0.1mm	L'>L
IF-380	d+0.5mm	-
IF-530	d+1.0mm	-
IF-A1520	d+1.0mm	L'>L
IF-A2330	d+0.5mm	-

- 若溝槽會貫通平台的底部，則必須使用「內部擴散板」，藉此防止操作員直接看到LED背光以及防止攝影機直接接收背光的光線。「內部擴散板」則是配置在背光的上方。
- 平台的重量應與標準的平面平台相同。若重量有大幅變化，就可能會改變共振頻率並影響送料器的運作。於此情況下，可透過調整送料器的運作頻率來因應。

- **凹槽**

  要設計帶凹槽的平台時，應考量以下項目：
  - 設計的凹槽能讓機器人輕易抓取零件。在理想情況下，讓零件平坦的一面朝上，以便於真空夾具拾取。
  - 零件不需要透過凹槽完全地排列（定位）。畢竟視覺系統的目的在於從2D平面上檢測零件的座標和旋轉角度。且平台的結構越簡單，製作成本就越低。此外，一般而言，簡單的平台設計會有更好的結果。

這些只是一般準則。仍必須根據具體情況因應調整。

4.2.2.3 平台的容許重量

	平台的最大重量**1	零件的最大重量**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

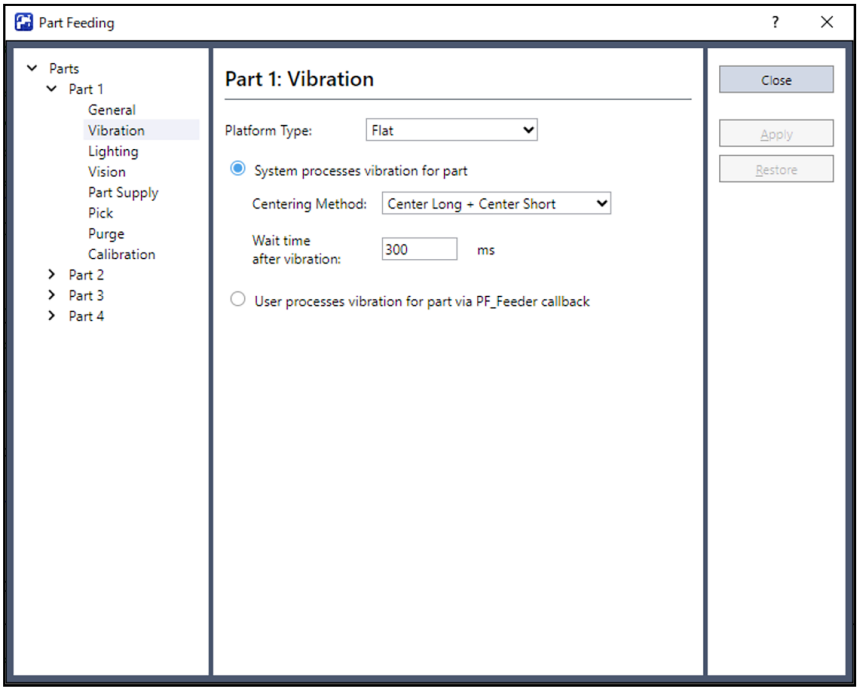
	平台的最大重量**1	零件的最大重量**2
IF-A2330	2.5 kg	1.5 kg

**1
平台表示在送料器上部振動的部分（不含零件）。平台由平板部分和框架部分組成。

**2
表示平台上可搭載零件的最大重量。

4.2.3 選擇平台

透過Epson RC+ 8.0功能表 - [工具] - [料件送料] - [振動]來選擇平台類型。如需詳細資訊，請參閱以下內容。
振動

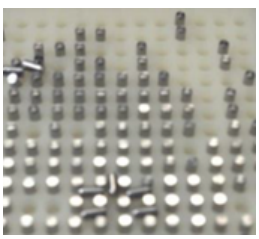


選擇標準平台（平面、防黏著、防滾動）時，可以選擇由系統處理送料器振動，或是在PF_Feeder回呼函數中處理送料器振動。通常交由系統處理送料器振動是最佳選擇。
在振動頁面選擇「由PF_Feeder callback進行振動控制」時，使用者可自行編寫送料器的振動控制。系統會將建議的送料器振動類型作為「state」參數提供給PF_Feeder回呼。此部分將在下一章詳細說明。

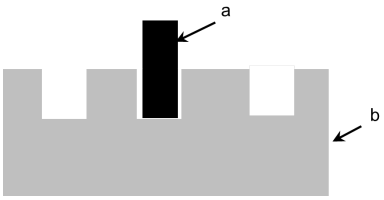
選擇自訂平台（溝槽、孔、凹槽）時，使用者必須透過PF_Feeder回呼函數處理零件的振動。使用自訂平台時，系統不會判斷建議的送料器振動類型。（最佳的振動類型會依據平台的加工而改變。）

4.2.3.1 自訂平台處理的程式範例

使用帶孔的自訂平台，垂直放置插銷。



表面



剖面圖

a	零件
b	平台

使用者需自行執行視覺系統並重新載入零件座標佇列時，選擇功能表 - [工具] - [料件送料] - [零件] - [視覺系統]中的[由PF_Vision callback進行視覺處理]。於此範例中，則是交由系統處理視覺程序。
機器人的取放動作將在PF_Robot回呼內執行。平台類型選擇為「孔」。
由於屬於自訂平台，所以僅能選擇「由PF_Feeder callback進行振動控制」。插銷的上部會透過零件檢測視覺序列進行檢測，座標則將會自動載入到零件座標佇列中。
在此範例中，不存在用於搜尋背面零件的視覺物件。
在視覺系統擷取影像、表面零件載入到零件座標佇列之後，將會調用PF_Feeder回呼函數。使用者程式碼則會在PF_Feeder回呼函數內判斷送料器的振動方式並加以執行。
表面零件的數量將作為引數「NumFrontParts」傳遞給PF_Feeder回呼函數。於此範例中，在引數「NumFrontParts」大於「0」時，機器人可以取放零件，因此不需要送料器振動。於此情況下，PF_Feeder回呼函數的回傳值必須設為「PF_CALLBACK_SUCCESS」。
此回傳值指示系統呼叫PF_Robot回呼函數。若「NumFrontParts」等於「0」，範例程式碼將會對零件Blob序列執行VRun，藉此判斷是否有零件集中或完全沒有零件。若零件Blob序列未搜尋到零件，料斗將會開啟並向送料器供應零件。若零件Blob序列有搜尋到零件，送料器將會執行翻轉動作，並在向前位移之後向後位移，藉此讓插銷落入孔中。在每次送料器振動時，系統必須重新擷取新的視覺影像。
因此會將回傳值設為「PF_CALLBACK_RESTART」。系統將會重新擷取新影像，在重新載入零件佇列之後，再次調用PF_Feeder回呼函數。使用者程式碼則會判斷是否需要進一步的送料器動作。

TIP

翻轉、長時間的前向位移、短時間的後向位移是自訂平台的典型有效動作。
如需詳細資訊，請參閱以下內容。
程式範例 5.2

提示

平台類型為孔、溝槽、凹槽時，常數PF_FEEDER_UNKNOWN將會被傳遞至PF_Feeder回呼函數的「state」引數中。這是因為在使用自訂平台時，系統無法決定適當的送料器運作所導致。
如需詳細資訊，請參閱以下內容。
程式範例 5.2

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

    ' 帶孔結構化平台的示例 state = PF_FEEDER_UNKNOWN

    Integer PFControlReturnVal
    Integer numFound

    Select True
```

```
        ' 可進行Pick時
        Case NumFrontParts > 0
            ' 有可進行Pick的零件，因此調用PF_Robot
            PF_Feeder = PF_CALLBACK_SUCCESS

        ' 沒有正面朝上的零件，但有背面朝上的零件時
        Case NumFrontParts = 0 And NumBackParts <> 0

            ' 翻轉、前向位移、後向位移
            PF_Flip PartID, 500
            PF_Shift PartID, PF_SHIFT_FORWARD, 1000
            PF_Shift PartID, PF_SHIFT_BACKWARD, 300

            PF_Feeder = PF_CALLBACK_RESTART ' 重新啟動後，重新擷取影像

        ' 正面朝上、背面朝上的零件都未搜尋到時
        ' 零件可能呈塊狀或托盤上完全沒有零件
        ' 從Part Blob序列重新擷取影像以進行判斷
        Case NumFrontParts = 0 And NumBackParts = 0

            PF_Backlight 1, On ' 啟動背光
            VRun PartBlob ' 擷取影像
            PF_Backlight 1, Off ' 關閉背光
            VGet PartBlob.Blob01.NumberFound, numFound ' 是否有搜尋到Blob？

            If numFound > 0 Then ' 搜尋到零件塊時

                ' 翻轉、前向位移、後向位移
                PF_Flip PartID, 500
                PF_Shift PartID, PF_SHIFT_FORWARD, 1000
                PF_Shift PartID, PF_SHIFT_BACKWARD, 300

            Else ' 未搜尋到零件時

                ' 調用零件供應用的Control回呼函數
                PFControlReturnVal = PF_Control(PartID, PF_CONTROL_SUPPLY_FIRST)

            EndIf

            PF_Feeder = PF_CALLBACK_RESTART ' 重新啟動後，重新擷取影像
    Send

Fend
```

4.2.3.2 使用PF_Feeder回呼函數的標準平面平台的例子

使用標準的平面平台。系統通常能以最佳方式處理平面平台的振動。
但在此範例中，已判斷必須採取特殊的振動處理。平台類型為「平面」、「吸附防止」或「防滾動」時，系統也能以最適當的方式振動零件。系統的判斷將會透過引數「state」提供給PF_Feeder回呼函數。「PartFeed.inc」檔案中有將多個狀態定義為常數。
例如，常數「PF_FEEDER_PICKOK」代表零件可透過機器人進行取放。另一個例子，則是當系統判斷翻轉零件是最佳動作時，常數「PF_FEEDER_FLIP」就會被傳遞至PF_Feeder回呼函數。使用PF_Feeder回呼函數的引數「state」和送料器控制命令，即可建立替代系統處理的自訂處理程序。
以下範例將示範使用引數「state」執行建議操作的基本概念。此範例並非詳盡無遺。

如需較完整的範例，請參閱以下內容。

程式範例 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 ' 有可拾取的零件,

            ' 因此調用PF_Robot

        Case PF_FEEDER_SUPPLY
            ' 從料斗供應
            PFControlReturnVal = PF_Control(PartID, PF_CONTROL_SUPPLY_FIRST)
            PF_CenterByShift PartID ' 從料斗供應後再將零件置中
            PF_Feeder = PF_CALLBACK_RESTART ' 重新啟動後，重新擷取影像

        Case PF_FEEDER_FLIP
            PF_Flip PartID
            PF_Feeder = PF_CALLBACK_RESTART ' 重新啟動後，重新擷取影像

        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 ' 重新啟動後，重新擷取影像

        Case PF_FEEDER_HOPPER_EMPTY
            ' 通知使用者料斗已空
            PFStatusReturnVal = PF_Status(PartID, PF_STATUS_NOPART)
            ' 從料斗供應零件
            PFControlReturnVal = PF_Control(PartID, PF_CONTROL_SUPPLY_FIRST)
            PF_Center PartID, PF_CENTER_LONG_AXIS
            ' 置中、翻轉、零件分離
            PF_Center PartID, PF_CENTER_SHORT_AXIS
            PF_Flip PartID
            PF_Feeder = PF_CALLBACK_RESTART ' 重新啟動後，重新擷取影像

        Case PF_FEEDER_WRONGPART
            If PF_Info(PartID, PF_INFO_ID_OBJECT_PURGE_ENABLED) Then
                ' 若清除功能已啟用，將使用Vision回饋進行清除
                ' 各清除嘗試將持續1500ms
                ' 平台上可保留的零件為3個
                ' 重試5次
                PFPurgeStatus = PF_Purge(1, 2, 1500, 3, 5)
                If PFPurgeStatus = False Then
                    Print "Purge was not successful"
                    Quit All
                EndIf
            Else ' 通知使用者送料器上可能有錯誤的零件
                Print "Wrong part may be on the feeder"
                Quit All
            EndIf
```

```
    Send
Fend
```

5. 故障排除

5.1 故障排除列表

5.1.1 不清楚設定在送料器上的IP位址為何

1. 對IP位址進行初始化。
   如需詳細步驟，請參閱以下內容。
   「Epson RC+ 8.0 選配件 Part Feeding 8.0 If-***篇 - IP位址的初始化」
   ***：送料器機型名稱（IF-80、IF-240或IF-380/530） 「Epson RC+ 8.0選配件 Part Feeding 8.0 IF-A1520 & IF-A2330篇 - 重設」
2. 指定已初始化的IP位址並連接到送料器。
   之後再變更IP位址。
   如需詳細步驟，請參閱以下內容。
   開始使用

5.1.2 送料器不振動或振動太弱

- 送料器的LED顯示是否亮起或閃爍？
  若未亮起或閃爍，則可能是未供電。
- Epson RC+上若有顯示訊息，請依照各錯誤訊息的因應對策處理。
- 有可能是送料器校準失敗。
  請重新執行給料器校準。
  關於執行方法，請參閱以下內容。
  校準&測試
  若仍無法解決，請載入預設值並重新執行校準。

若以上原因皆非，則有可能是送料器故障。請聯繫供應商。

5.1.3 送料器上的零件移動不順暢或偏移

- 可能是安裝送料器的架台剛性不足，振動能量無法有效傳遞到平台。請改用高剛性的台座。
- 有可能是送料器校準失敗。
  請重新執行給料器校準。
  關於執行方法，請參閱以下內容。
  校準&測試
  若仍無法解決，請載入預設值並重新執行校準。
- 平台有可能會因為長期使用而磨損。
  平台屬於消耗品。請更換平台。

若以上原因皆非，則有可能是送料器故障。請聯繫供應商。

5.1.4 料斗不振動

- 若屬於從供應商購買的料斗，請確認料斗和送料器的連接。
  如需詳細資訊，請參閱以下內容。
  送料器、料斗
- 請確認與料斗之間的通訊設定是否正確。
- 請確認料斗操作的程式設計是否正確。
  關於確認方法，請參閱以下內容。
  PF_Control

5.1.5 平台被零件填滿

- 可能是每1次料斗運作投入送料器的零件數量過多。請適當調整料斗的投入數量。

5.1.6 平台上的零件用盡

- 請確認料斗中是否有零件。
- 有可能是料斗的零件投入不順暢。
  請調整料斗的投入數量。
- 若屬於從供應商購買的料斗，請確認料斗和送料器的連接。

5.1.7 不知道如何取得備份

- 請參閱以下內容。
  控制器的備份、還原

5.2 故障受理單

[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]
