SIEMENS

SINUMERIK

SINUMERIK 840D sl/ SINUMERIK 828D 工作計畫

編程手冊

適用範圍

控制系統
SINUMERIK 840D sl/840DE sl
SINUMERIK 828D

軟體	版
840D sl/840DE sl NCU 系統軟體	2.6
828D 的 NCU 系統軟體	2.6

06/2009
6FC5398-2BP20-0MA0

法律聲明

警告事項意涵

為了您的人身安全以及避免財產損失，必須注意本手冊中的提示。有關人身安全的提示通過一個警告三角表示，僅與財產損失有關的提示不帶警告三角。

⚠危險
表示如果不采取相應的小心措施，將會導致死亡或者嚴重的人身傷害。

⚠警告
表示如果不采取相應的小心措施，可能導致死亡或者嚴重的人身傷害。

⚠小心
帶有警告三角，表示如果不采取相應的小心措施，可能導致輕微的人身傷害。

小心
不帶警告三角，表示如果不采取相應的小心措施，可能導致財產損失。

注意
不帶警告三角，表示如果不注意相應的提示，可能會出現不希望的結果或狀態。

當出現多個危險等級的情況下，每次總是使用最高等級（較低數字）的警告提示。如果在某個警告提示中帶有警告可能導致人身傷害的警告三角，則可能在該警告提示中另外還附帶有可能導致財產損失的警告。

合格的崙業人員

唯有與各項工作要求**資格符合的人員**才能操作本文件所屬產品/系統，遵照各附帶文件說明，特別是其中的安全及警告提示。資格符合的人員由於具備相關訓練及經驗，可以察覺本產品/系統的風險，並避免可能的危險。

按規定使用 Siemens 產品

請注意下列說明：

⚠警告
Siemens 產品只允許用於目錄和相關技術文件中規定的使用情況。如果要使用其他公司的產品和組件，必須得到 Siemens 推薦和允許。正確的運輸、儲存、組裝、裝配、安裝、調試、操作和維護是產品安全、正常運行的前提。必須保證允許的環境條件。必須注意相關檔中的提示。

商標

所有帶有標記符號 ® 的都是西門子股份有限公司的注冊商標。標簽中的其他符號可能是一些其他商標，任何第三方將其用于其他目的都會損坏所有者的利益。

責任免除

我們已對印刷品中所述內容與硬件和軟件的一致性作過檢查。然而不排除存在偏差的可能性，因此我們不保証印刷品中所述內容與硬件和軟件完全一致。印刷品中的數據都按規定經過檢測，必要的修正值包含在下一版本中。同時歡迎您提出改進建議。

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
德國

文檔訂購號: 6FC5398-2BP20-0MA0
℗06/2009

前言

SINUMERIK 文件

SINUMERIK 文件可分為三個部分：

- 一般文件
- 使用者文件
- 製造商 / 服務文件

下列主題資訊可於下列網址取得：http://www.siemens.com/motioncontrol/docu

- 訂購文件
  在此可找到最新出版品簡介
- 下載文件
  連結「維修支援中心」取得更多下載檔案資訊。
- 線上搜尋文件
  DOConCD 上的資訊並且直接存取 DOConWEB 內的出版品
- 使用「我的文件管理員(My Documentation Manager，MDM)」使用西門子文件內容編輯個人專屬文件，請參閱：http://www.siemens.com/mdm。
  「我的文件管理員」提供一系列功能，讓您產生自己的機床文件。
- 訓練和常見問題集
  瀏覽網頁即可取得有關訓練課程和常見問題集範圍方面的資訊。

目標群組

本出版品適用於下列對象：

- 程式設計師
- 專案工程師

用途

使用本編程手冊，目標群組群得以研發、撰寫、測試以及修正程式與軟體使用者介面。

標準範圍

本程式設計指南描述標準功能所提供的各項功能。 機器刀具製造商所作之功能擴充或更改由機器刀具製造商自行提供文件。

未於文件中述及的其他功能，有可能會在控制系統中執行。 然而，並不代表應於新控制系統內或維修時提供此類功能。

此外，為了簡化文件，本文件不包含所有產品類型的詳細資訊，且無法涵蓋所有可理解的安裝案例、操作或維護。

技術支援

如果您有任何疑問，請與我們的服務熱線聯繫：

	歐洲 / 非洲
電話	+49 180 5050 - 222
傳真	+49 180 5050 - 223
德國國內長途電話每分鐘 0.14 歐元，行動電話費率可能有所不同。	
網際網路	http：//www.siemens.de/automation/support-request

	美洲
電話	+1 423 262 2522
傳真	+1 423 262 2200
電子郵件	請傳送至：techsupport.sea@siemens.com

	亞太地區
電話	+86 1064 757575
傳真	+86 1064 747474
電子郵件	請傳送至：support.asia.automation@siemens.com

說明

其他國家之技術支援電話號碼，請造訪：
http://www.automation.siemens.com/partner

對於本手冊的疑問

如果您對本文件有疑問（建議、更正），請傳送至下列傳真號碼或電子郵件地址：

傳真： +49 9131- 98 2176
電子郵件： mailto:docu.motioncontrol@siemens.com

本文件附錄備有傳真表單。

SINUMERIK 網址

http://www.siemens.com/sinumerik

「基礎」與「進階」編程手冊

NC 程式設計說明分成兩份手冊：

1. **基礎**

   此“基礎”編程手冊乃針對具鑽孔、銑削及車削操作方面適當專業技能的操作員使用。 簡易的程式設計範例用於解釋指令及敘述，同樣也是根據 DIN 66025 定義。

2. **工作計畫**

   “進階”編程手冊係針對具進階程式設計知識的技術人員。 藉著特殊程式設計語言，SINUMERIK 控制系統讓使用者能進行複雜的工件程式設計（例如：任意形狀的表面、通道協調,...），並可輕易完成複雜作業的程式設計。

所敘述的 NC 語言元素的可用性

在手冊中所敘述的 NC 語言元素在 SINUMERIK 840D sl都有提供。 SINUMERIK 828D 中的可用功能列於「敘述清單 (頁 633)」的「828D」欄中。

目錄

1 彈性的 NC 程式設計

1.1 變數

1.1.1 有關變數的一般資訊

變數的應用，尤其在配合算術函數及檢查結構時，能讓工件程式及循環得以高度的彈性進行設定。為此，系統可使用三種不同類型的變數。

- 系統變數

  系統變數為預先定義有固定意義的變數，其定義在系統中且可讓使用者使用。其亦可由系統軟體讀取或寫入。範例：機械參數

  大體而言，系統變數的意義由系統預先定義且其屬性固定。然而，使用者可利用重定義的方式對這些特性做小幅修改。請參閱"重新定義系統變數、使用者變數、以及NC語言指令（REDEF） (頁 27)"。

- 使用者變數

  使用者變數為系統不了解其意義的變數；系統並不會將其列入評估。其意義僅能由使用者定義。

  使用者變數可劃分為：

  – 預定義使用者變數

    預定義使用者變數為系統中已定義的變數且使用者僅需透過特定機械參數將其編號參數化即可的變數。使用者可對這些變數的屬性做明顯改變。請參閱"重新定義系統變數、使用者變數、以及NC語言指令（REDEF） (頁 27)"。

  – 使用者自訂變數

    使用者自訂變數為僅能由使用者定義且系統在未達執行時間不會產生的變數。這些編號、資料類型、可見性、及其他特性均僅能由使用者定義。

    請參閱"使用者變數的定義（DEF） (頁 22)"。

另請參見

系統變數 (頁 16)

預定義使用者變數：算術參數（R） (頁 18)

預定義使用者變數：連結變數 (頁 19)

1.1.2 系統變數

系統變數為系統中預先定義且可和控制系統一同存取目前工件程式及循環中參數設定，如機台、控制系統及處理狀態，的變數。

前置處理變數

前置處理變數為前置處理前後關係時；換言之便是在達到已程式設計的系統變數轉譯後的工件程式單節時，讀取及寫入的變數。前置處理變數不會觸發前置處理停止。

主要執行變數

主要執行變數為主要執行前後關係時；換言之便是在達到已程式設計的系統變數執行後的工件程式單節時，讀取及寫入的變數。以下為主要執行變數：

- 可在同步動作（讀取/寫入）中程式設計的系統變數
- 可程式設計在工件程式中並觸發前置處理停止（讀取/寫入）的系統變數
- 可程式設計在工件程式中且其值可在前置處理時加以計算但在未達主要執行時不會寫入的系統變數（主要執行同步：僅可寫入）。

前置系統

為能明確判斷，系統變數的名稱前面通常含有$號並接著 1 或 2 個字母及底線。

$ + 1st letter	**意義：資料類型**
在前置處理時讀取/寫入的系統變數	
$M	機械參數 1)
$S	設定資料，保護區 1)
$T	刀具管理資料
$P	程式設計值
$C	ISO 封套循環的循環變數
$O	選項資料
R	R 參數（算術參數）2)
在主要執行時讀取/寫入的系統變數	
$$M	機械參數 1)
$$S	設定資料 1)
$A	最新主要執行資料
$V	伺服資料
$R	R 參數（算術參數）2)
1）當機械參數及設定資料在工件程式/循環中作為前置處理變數使用時，其前面會加上 1 個$號。當其在同步動作中做為主要執行變數使用時，其前面會加上 2 個 $ 號。 2）當 R 參數在工件程式/循環中作為前置處理變數使用時，會省略前置符號，例如 R10。當其在同步動作中做為主要執行變數使用時，會在前面加上一個$號，例如$R10。	

2nd letter	意義：可見性
N	NCK 全域變數（**N**CK）
C	通道專屬變數（**通道**）
A	軸專屬變數（**軸**）

一般條款

前置字元系統的例外情況

以下系統變數不符合上述的前置字元系統規則：

- $TC_...:此處第 2 個字母 C 並非表示通道專屬系統變數而指刀把專屬系統變數（TC= 刀盤）。
- $P_ ...:通道專屬系統變數

機械參數及設定資料在同步動作中的應用

當同步動作中使用機械參數及設定資料時，前置字元可用於定義要在前置處理執行或主要執行同時讀取機械參數或設定資料。

若在加工時該資料維持不變，其可與前置處理執行同時讀取。為此，機械參數或設定資料前面會加上 1 個 $ 號：

程式碼

```
ID=1 WHENEVER G710 $AA_IM[z] < $SA_OSCILL_REVERSE_POS2[Z]—6 DO $AA_OVR[X]=0
```

若在加工時該資料改變，則其必須與主要執行同時讀取/寫入。為此，機械參數或設定資料前面會加上 2 個 $ 號：

程式碼

```
ID=1 WHENEVER $AA_IM[z] < $$SA_OSCILL_REVERSE_POS2[Z]—6 DO $AA_OVR[X]=0
```

說明

寫入機械參數

當寫入機械參數或設定資料時，確保在工件程式/循環執行時啟動的存取層級允許寫入且該資料能"立即"生效是很重要的。

參考

所有系統變數的特性清單請參閱：

/PGA1/參數手冊，系統變數

另請參見

資料類型 (頁 49)

1.1.3 預定義使用者變數：算術參數（R）

功能

算術參數或 R 參數為具有 R 符號的預定義使用者變數，其定義為一個實數資料類型的陣列。基於歷史因素，R 參數可同時標記為有陣列索引，例如 R[10]，及無陣列索引，例如 R10。

當採用同步動作時，前置字元需含$符號，例如$R10。

句法

當作為前置處理變數使用時：
R<n>
R[<算式>]
當作為主要執行變數使用時：
$R<n>
$R[<算式>]

意義

R: 做為前置處理變數時的識別碼，例如在工件程式中
$R: 做為主要執行變數時的識別碼，例如在同步動作中

類型：REAL

值域：對於非指數標記法：
±（0.000 0001 至 9999 9999）
注意：
最多 8 個小數點位數。
對於指數標記法：
± (1*10^{-300} 至 1*10^{+300}）
注意事項：
- 標記：<尾數>EX<指數>例如 8.2EX-3
- 包含符號與小數點，最多允許 10 個字元。

<n>: R 參數的編號

類型：INT

值域：0 - MAX_INDEX
請注意
MAX_INDEX 是從參數化的 R 參數編號計算得到的：
MAX_INDEX =（MD28050
$MN_MM_NUM_R_PARAM）- 1

<算式>: 陣列索引
所有的算式均可作為陣列索引，只要該算式結果能轉換成整數資料類型（整數、實數、布林值、字元）即可。

範例

在數學函數中配置並應用 R 參數：

程式碼		註解
R0=3.5678	;	在前置處理中配置
R[1]=-37.3	;	在前置處理中配置
R3=-7	;	在前置處理中配置
$R4=-0.1EX-5	;	在主要執行中配置：R4 = -0.1 * 10^-5
$R[6]=1.874EX8	;	在主要執行中配置：R6 = 1.874 * 10^8
R7=SIN(25.3)	;	在前置處理中配置
R[R2]=R10	;	使用 R 參數間接定址
R[(R1+R2)*R3]=5	;	使用數學算式間接定址
X=(R1+R2)	;	將軸 X 移動至 R1 及 R2 加總所產生的位置
Z=SQRT（R1*R1+R2*R2）	;	將軸 Z 移動至（R1^2 + R2^2）的平方根位置

1.1.4 預定義使用者變數：連結變數

功能

連結變數可用於"NCU 連結"函數中作為連結在同一網路上 NCU 間的循環資料交換。其可加速對連結變數記憶體的資料格式專屬存取。連結變數記憶體由使用者/機台製造商以系統專屬的方式同時透過大小及資料結構加以定義。

連結變數為系統全域使用者變數，可由在已設定連結通訊的連結中所涵蓋的所有 NCU 讀取及寫入至工件程式及循環中。有別於全域使用者變數（GUD），連結變數亦可用在同步動作中。

在無啟用 NCU 連結的系統中，連結變數可用在控制系統本機上作為全域使用者變數（GUD）旁的額外全域使用者變數。

句法

```
$A_DLB[<索引>]
$A_DLW[<索引>]
$A_DLD[<索引>]
$A_DLR[<索引>]
```

意義

$A_DLB: 位元組資料格式（1 位元組）的連結變數
資料類型： UINT
值域： 0 到 255

$A_DLW: 字組資料格式（2 位元組）的連結變數
資料類型： INT
值域： -32768 到 32767

$A_DLD: 雙字組資料格式（4 位元組）的連結變數
資料類型： INT
值域： -2147483648 到 2147483647

$A_DLR: 實數資料格式（8 位元組）的連結變數
資料類型： REAL
值域： $\pm(2{,}2*10^{-308} \ldots 1{,}8*10^{+308})$

<索引>: 位址索引以位元組為單位，從連結變數記憶體的起點開始計算
資料類型： INT
值域： 0 - MAX_INDEX

注意事項

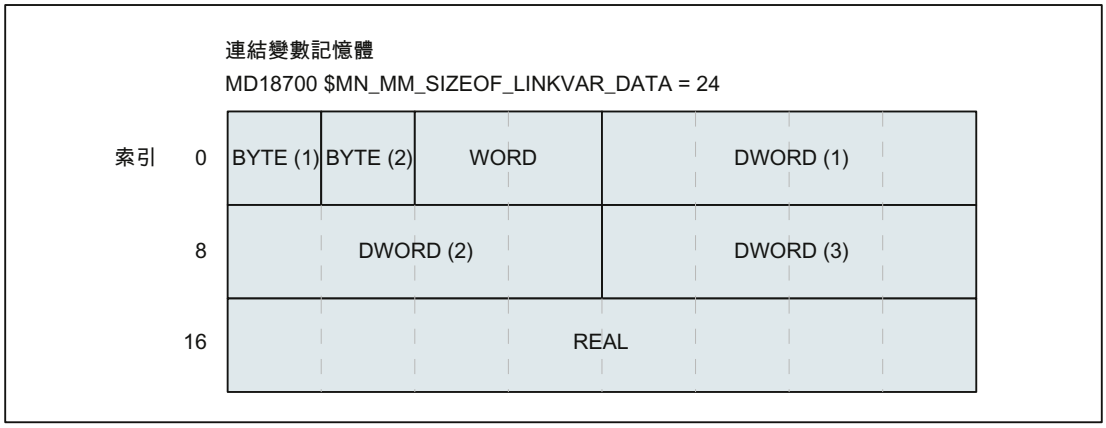
- MAX_INDEX 可從連結變數記憶體的參數化大小計算得到：MAX_INDEX =（MD18700 $MN_MM_SIZEOF_LINKVAR_DATA）- 1
- 僅有索引可程式設計故定址在連結變數記憶體中的位元組位於資料格式臨界值⇒
  索引= n *位元組，其中 n = 0、1、2、等。
  - $A_DLB[i]：i = 0、1、2、等。
  - $A_DLW[i]：i = 0、2、4、等。
  - $A_DLD[i]：i = 0、4、8、等。
  - $A_DLR[i]：i = 0、8、16、等。

範例

一個自動裝置系統含有 2 個 NCU（NCU1 及 NCU2）。機械軸 AX2 連接至 NCU1。其以 NCU2 的連結軸方式移動。

NCU1 反覆的將軸 AX2 的實際電流值（$VA_CURR）寫入至連結變數記憶體中。NCU2 反覆讀取透過連結通訊傳送的實際電流值並在超過臨界值時顯示警報 61000。

連結變數記憶體的資料結構如下圖所示。實際電流值以實數值傳輸。



NCU1

NCU1 利用連結變數$A_DLR[16]在靜態同步動作中的插補循環時反覆的將軸 AX2 的實際電流值寫入至連結變數記憶體中。

程式碼

```
N111 IDS=1 WHENEVER TRUE DO $A_DLR[16]=$VA_CURR[AX2]
```

NCU2

NCU2 利用連結變數$A_DLR[16]在靜態同步動作中的插補循環時反覆的將軸 AX2 的實際電流值從連結變數記憶體中讀取出來。若實際電流值大於 23.0 A，便會顯示警報 61000。

程式碼

```
N222 IDS=1 WHEN $A_DLR[16] > 23.0 DO SETAL(61000)
```

1.1.5 使用者變數的定義（DEF）

功能

DEF 用於定義使用者專屬變數並為其指派值。為使其與系統變數不同，這些變數被稱為使用者定義變數或使用者變數（使用者資料）。

根據有效範圍區分（換言之，便是該變數的可見範圍），使用者變數分為以下幾類：

- 局部使用者變數（LUD）

  局部使用者變數（LUD）為定義在執行時不屬於主要程式之工件程式中的變數。其會在呼叫工件程式時產生並在工件程式結束或 NC 重置時刪除。局部使用者變數僅能在定義他們的工件程式中存取。

- 程式全域使用者變數（PUD）

  程式全域使用者變數（PUD）為在做為主要程式之工件程式中定義的使用者變數。其會在工件程式啟動時產生並在工件程式結束或 NC 重置時刪除。可以在主程式和主程式的所有副程式中存取 PUD。

- 全域使用者變數（GUD）

  全域使用者變數（GUD）為在資料單節（SGUD、MGUD、UGUD、GUD4 至 GUD9）中定義的 NC 或通道全域變數且在關機或重新啟動後仍維持不變。所有工件程式都可以對 GUD 進行存取。

使用者變數需在使用（讀取/寫入）前先做定義。此時，必須注意以下規則：

- GUD 必須在定義在定義檔案中，例如_N_DEF_DIR/_M_SGUD_DEF。
- PUD 及 LUD 必須定義在工件程式的定義區段中。
- 該資料必須在專用的單節中定義。
- 每個資料定義僅能使用一種資料類型。
- 相同資料類型的數個變數可針對各個資料定義做定義。

句法

```
DEF<範圍> <類型> <PP_stop> <init_time> <phys_unit> <limit_values>
<access_rights> <名稱>[<value_1>,<value_2>,<value_3>]=<init_value>
```

意義

DEF:	用於定義 GUD、PUD、LUD 使用者變數的指令
<範圍>:	有效範圍，僅和 GUD 有關：
	NCK: NC 全域使用者變數
	CHAN: 通道全域使用者變數
<類型>:	資料類型：
	INT: 含正負號之整數
	REAL: 實數編號（符合 IEEE 的長實數）
	BOOL: 真值 TRUE（真）（=1），FALSE（偽）（=0）
	CHAR: ASCII 字元
	字串[<最大長度>]: 已定義長度的字元串
	AXIS: 軸 / 主軸識別碼
	“框：: 靜態座標轉換的幾何資料
	請參閱“資料類型 (頁 49)”
<PP_stop>:	前置處理停止，僅與 GUD（選用）有關
	SYNR: 讀取時前置處理停止
	SYNW: 寫入時前置處理停止
	SYNRW: 讀取/寫入時前置處理停止
<init_time>:	在該變數重新初始化（選用）的時候
	INIPO: 啟動
	INIRE: 主程式結束，NC 重置或啟動
	INICF: 新設定或主程式結束，NC 重置或啟動
	PRLOC: 主程式結束，NC 重置接著本機變更或啟動
	請參閱“屬性：初始化值 (頁 30)”
<phys_unit>:	實體元件（選用）
	PHU <元件>:
	請參閱“屬性：實體元件（PHU） (頁 34)”
<臨界值>:	下臨界與上臨界值（選用）
	LLI <臨界值>: 下臨界值（下臨界值）
	ULI <臨界值>: 上臨界值（上臨界值）
	請參閱“屬性：臨界值（LLI、ULI） (頁 33)”
<存取權限>:	透過工件程式或 OPI（選用）讀取/寫入 GUD 的存取權限
	APRP <保護等級>: 讀取：工件程式
	APWP <保護等級>: 寫入：工件程式
	APRB <保護等級>: 讀取：OPI
	APWB <保護等級>: 寫入：OPI
	保護等級 值域：0 到 7
	請參閱“屬性： 存取權力（APR、APW、APRP、APWP、APRB、APWB） (頁 36)”

<名稱>:	變數名稱 **注意事項** • 最多 31 個字元 • 前兩個字元必須為一個字母及 / 或一個底線。 • $符號僅保留給系統變數故不可使用。
[<value_1>, <value_2>, <value_3>]:	1 至最大 3 相陣列變數的陣列大小規格（選用）
<init_value>:	初始化值（選用） 請參閱“屬性：初始化值 (頁 30)” 有關陣列變數的初始化： 請參閱“陣列變數（DEF，SET，REP）之定義與初始化 (頁 42)”

範例

範例 1：機台製造商的資料單節中之使用者變數定義

程式碼

```
%_N_MGUD_DEF                                                    ; GUD 單節：機械製造商
$PATH=/_N_DEF_DIR
DEF CHAN REAL PHU 24 LLI 0 ULI 10 STROM_1, STROM_2
; 說明
; 2 個 GUD 項目的定義：STROM_1, STROM_2
; 有效範圍：貫穿通道
; 資料類型：REAL
; PP 停止：未程式設計=>預設值=無 PP 停止
; 實體元件：24 = [A]
; 臨界值低= 0.0，高= 10.0
; 存取權限：未程式設計=>預設值= 7 =按鍵操作開關位置 0
; 初始化值：未程式設計=>預設值= 0.0

DEF NCK REAL PHU 13 LLI 10 APWP 3 APRP 3 APWB 0 APRB 2 ZEIT_1=12, ZEIT_2=45
; 說明
; 2 個 GUD 項目的定義：ZEIT_1, ZEIT_2
; 有效範圍：貫穿 NCK
; 資料類型：REAL
; PP 停止：未程式設計=>預設值=無 PP 停止
; 實體元件：13 = [s]
; 臨界值低= 10.0，高=未程式設計=>上定義範圍臨界值
; 存取權限：
;    工件程式：寫入/讀取= 3 =最終使用者
;    OPI：寫入= 0 = Siemens，讀取= 3 =最終使用者
; 初始化值：ZEIT_1 = 12.0, ZEIT_2 = 45.0
```

程式碼

```
DEF NCK APWP 3 APRP 3 APWB 0 APRB 3 STRING[5] GUD5_NAME = "計數器"
; 說明
; 1 個 GUD 項目的定義：GUD5_NAME
; 有效範圍：貫穿 NCK
; 資料類型：字串，最大 5 個字元
; PP 停止：未程式設計=>預設值=無 PP 停止
; 實體元件：未程式設計=>預設值= 0 =無實體元件
; 臨界值未程式設計的 => 定義範圍臨界值：低= 0，高= 255
; 存取權限：
;    工件程式：寫入/讀取= 3 =最終使用者
;    OPI：寫入= 0 = Siemens，讀取= 3 =最終使用者
; 初始化值：“計數器”
M30
```

範例 2：程式全域及程式－局部使用者變數（LUD / PUD）

程式碼	註解
PROC MAIN	; 主程式
DEF INT VAR1	; PUD 定義
...	
SUB2	; 副程式呼叫
...	
M30	

程式碼	註解
PROC SUB2	; 副程式 SUB2
DEF INT VAR2	; LUD DEFINITION
...	
IF (VAR1==1)	; 讀取 PUD
VAR1=VAR1+1	; 讀取與寫入 PUD
VAR2=1	; 寫入 LUD
ENDIF	
SUB3	; 副程式呼叫
...	
M17	

程式碼	註解
PROC SUB3	; 副程式 SUB3
...	
IF (VAR1==1)	; 讀取 PUD
VAR1=VAR1+1	; 讀取與寫入 PUD
VAR2=1	; 錯誤：從 SUB2 來的 LUD 不明
ENDIF	
...	
M17	

範例 3：資料類型軸的使用者變數定義與用法

程式碼	註解
DEF AXIS ABSCISSA	; 第一幾何座標軸
DEF AXIS SPINDLE	; 主軸
...	
IF ISAXIS(1) == FALSE GOTOF CONTINUE	
ABSCISSA = $P_AXN1	
CONTINUE:	
...	
SPINDLE=(S1)	第一主軸
OVRA[SPINDLE]=80	; 主軸手動倍率調整 = 80%
SPINDLE=(S3)	第三主軸

一般條款

全域使用者變數（GUD）

在全域使用者變數（GUD）的定義關係中，需將以下機械參數列入考慮：

號碼	識別碼：$MN_	意義
11140	GUD_AREA_ SAVE_TAB	GUD 單節的輔助儲存
18118	MM_NUM_GUD_MODULES	在啟動檔案系統中的數個 GUD 檔案
18120	MM_NUM_GUD_NAMES_NCK	數個全域 GUD 名稱
18130	MM_NUM_GUD_NAMES_CHAN	數個通道專屬 GUD 名稱
18140	MM_NUM_GUD_NAMES_AXIS	數個軸專屬 GUD 名稱
18150	MM_GUD_VALUES_MEM	全域 GUD 值的記憶體位置
18660	MM_NUM_SYNACT_GUD_REAL	實數資料類型的可設定 GUD 數量
18661	MM_NUM_SYNACT_GUD_INT	整數資料類型的可設定 GUD 數量
18662	MM_NUM_SYNACT_GUD_BOOL	布林資料類型的可設定 GUD 數量
18663	MM_NUM_SYNACT_GUD_AXIS	軸資料類型的可設定 GUD 數量
18664	MM_NUM_SYNACT_GUD_CHAR	字元資料類型的可設定 GUD 數量
18665	MM_NUM_SYNACT_GUD_STRING	字串資料類型的可設定 GUD 數量

程式全域使用者變數（PUD）

注意
程式局部使用者變數（PUD）的可見度 定義在主程式中的程式局部使用者變數（PUD）僅在設定了以下機械參數值可見於子程式中： MD11120 $MN_LUD_EXTENDED_SCOPE = 1 若 MD11120 = 0，則主程式中所定義的程式局部使用者變數只可見於主程式中。

軸資料類型的 NCK 全域使用者變數之跨通道應用

在單節使用軸識別碼定義時所產生的軸資料類型的 NCK 全域使用者變數僅有在其他 NC 通道軸具有相同編號時，才可在其他 NC 通道中使用。

若否，則該變數需置於工件程式起點，否則，需如下列範例一樣使用AXNAME（...）函數（請參閱"軸函數（AXNAME、AX、SPI、AXTOSPI、ISAXIS、AXSTRING、MODAXVAL） (頁 589)"。

程式碼	註解
`DEF NCK STRING[5] ACHSE="X"`	`;` 資料單節內的定義
`N100 AX[AXNAME(ACHSE)]=111 G00`	`;` 用於工件程式中

另請參見

重新定義系統變數、使用者變數、以及NC語言指令（REDEF） (頁 27)

1.1.6 重新定義系統變數、使用者變數、以及 NC 語言指令（REDEF）

功能

REDEF 指令可用於變更系統變數、使用者變數及 NC 語言指令的屬性。重新定義的基本條件為其對應的定義需已過期。

在重新定義時不可同時變更多的屬性。需針對要變更的各個屬性程式設計個別的 REDEF 操作。

若已程式設計二或多個同時發生的屬性變更，最後的變更永遠有效。

可重新定義的屬性

請參閱"可定義與可重新定義屬性之概觀 (頁 41)"。

局部使用者變數（PUD/LUD）

局部使用者變數不可重新定義（PUD/LUD）。

句法

```
REDEF <名稱> <PP_stop>
REDEF <名稱> <phys_unit>
REDEF <名稱> <limit_values>
REDEF <名稱> <access_rights>
REDEF <名稱> <init_time>
REDEF <名稱> <init_time> <init_value>
```

意義

REDEF:	用於重新定義特定系統變數、使用者變數、及 NC 語言指令屬性的指令
<名稱>:	預先定義的變數或 NC 語言指令之名稱
<PP 停止>:	前置處理停止 SYNR: 讀取時前置處理停止 SYNW: 寫入時前置處理停止 SYNRW: 讀取/寫入時前置處理停止
<phys_unit>:	實體元件 PHU <元件>: 請參閱“屬性：實體元件（PHU） (頁 34)”
	Note 無法重新定義的項目： • 系統變數 • 全域使用者資料（GUD） • 資料類型：布林、軸、字串、框架
<臨界值>:	下限及（或）上限值 LLI <臨界值>: 下臨界值（下臨界值） ULI <臨界值>: 上臨界值（上臨界值） 請參閱“屬性：臨界值（LLI、ULI） (頁 33)”
	Note 無法重新定義的項目： • 系統變數 • 全域使用者資料（GUD） • 資料類型：布林、軸、字串、框架
<存取權限>:	透過工件程式或 OPI 讀取/寫入的存取權力 APX <保護等級>: 執行：NC 語言元件 APRP <保護等級>: 讀取：工件程式 APWP <保護等級>: 寫入：工件程式 APRB <保護等級>: 讀取：OPI APWB <保護等級>: 寫入：OPI 保護等級 值域：0 到 7 請參閱“屬性： 存取權力（APR、APW、APRP、APWP、APRB、APWB） (頁 36)”

<init_time>: 變數重新初始化的時間點

INIPO:	啟動
INIRE:	主程式結束，NC 重置或啟動
INICF:	新設定或主程式結束，NC 重置或啟動
PRLOC:	主程式結束，NC 重置接著本機變更或啟動

請參閱“屬性：初始化值 (頁 30)”

<init_value>: 初始化值
每當初始化值重新定義時，亦需指定初始化時間（請參閱 <init_time>）。
請參閱“屬性：初始化值 (頁 30)”
有關陣列變數的初始化：
請參閱“陣列變數（DEF，SET，REP）之定義與初始化 (頁 42)”

Note
無法重新定義的項目：
- 具例外設定資料的系統變數

範例

重新定義機台製造商資料單節中的系統變數$TC_DPC1

程式碼

```
%_N_MGUD_DEF                                          ; GUD 單節：機械製造商
$PATH=/_N_DEF_DIR
REDEF $TC_DPC1 APWB 2 APWP 3
REDEF $TC_DPC1 PHU 21
REDEF $TC_DPC1 LLI 0 ULI 200
REDEF $TC_DPC1 INIPO（100, 101, 102, 103）
; 說明
; 寫入存取權限：OPI =保護等級 2，工件程式=保護等級 3
; 注意事項
; 當使用存取檔案時需將存取權限重新定義從
; _N_MGUD_DEF 變成_N_MACCESS_DEF.
; 實體元件= [ % ]
; 臨界值低= 0，高= 200
; 陣列變數在啟動時以 4 個值產生。
M30
```

一般條款

頻率

重新定義每次均套用至以名稱做為識別的整個變數中。其中，陣列變數並不支援將不同屬性指派至個別陣列元素中。

另請參見

使用者變數的定義（DEF） (頁 22)

1.1.7 屬性：初始化值

定義（DEF）使用者變數

在定義時可先將初始化值指派給以下使用者變數：

- 全域使用者變數（GUD）
- 程式全域使用者變數（PUD）
- 局部使用者變數（LUD）

重新定義（REDEF）系統及使用者變數

在重新定義時可先將初始化值指派給以下變數：

- 系統資料
  - 設定資料
- 使用者資料
  - R－參數
  - 同步動作變數（$AC_PARAM、$AC_MARKER、$AC_TIMER）
  - 同步動作 GUD（SYG_xy[ ]，其中 x=R、I、B、A、C、S 而 y=S、M、U、4 至 9）
  - EPS 參數
  - 刀具資料 OEM
  - 刀庫資料 OEM
  - 全域使用者變數（GUD）

重新初始化時間

在重新定義時可指定變數重新初始化（即重置為初始化值）的時間點。

- INIPO（啟動）
  該變數在啟動時重新初始化。
- INIRE（重置）
  該變數在 NC 重置、模式群組重置、工件程式（M02/M30）結束或啟動時重新初始化。
- INICF（新設定）
  該變數在 HMI 透過工件程式指令 NEWCONFIG 提出 NewConf 請求或 NC 重置、模式群組重置、工件程式（M02/M30）結束或啟動時重新初始化。
- PRLOC（程式局部變更）
  該變數僅在 NC 重置、模式群組重置或工件程式（M02/M30）結束時若在目前的工件程式中有變更時才重新初始化。
  PRLOC 屬性僅能和可程式設計的設定資料一同變更（請參閱下表）。

表格 1- 1　可程式設計的設定資料

編號	識別碼	G 指令 1)
42000	$SC_THREAD_START_ANGLE	SF
42010	$SC_THREAD_RAMP_DISP	DITS/DITE
42400	$SA_PUNCH_DWELLTIME	PDELAYON
42800	$SA_SPIND_ASSIGN_TAB	SETMS
43210	$SA_SPIND_MIN_VELO_G25	G25
43220	$SA_SPIND_MAX_VELO_G26	G26
43230	$SA_SPIND_MAX_VELO_LIMS	LIMS
43300	$SA_ASSIGN_FEED_PER_REV_SOURCE	FPRAON
43420	$SA_WORKAREA_LIMIT_PLUS	G26
43430	$SA_WORKAREA_LIMIT_MINUS	G25
43510	$SA_FIXED_STOP_TORQUE	FXST
43520	$SA_FIXED_STOP_WINDOW	FXSW
43700	$SA_OSCILL_REVERSE_POS1	OSP1
43710	$SA_OSCILL_REVERSE_POS2	OSP2
43720	$SA_OSCILL_DWELL_TIME1	OST1
43730	$SA_OSCILL_DWELL_TIME2	OST2
43740	$SA_OSCILL_VELO	FA
43750	$SA_OSCILL_NUM_SPARK_CYCLES	OSNSC
43760	$SA_OSCILL_END_POS	OSE
43770	$SA_OSCILL_CTRL_MASK	OSCTRL
43780	$SA_OSCILL_IS_ACTIVE	OS
43790	$SA_OSCILL_START_POS	OSB
1）此 G 指令用於處理設定資料。		

一般條款

初始化值：全域使用者變數（GUD）

- 需設定以下機械參數方可在使用 INIRE（重置）或 INICF（新設定）進行定義或重新定義時將全域使用者變數（GUD）重置為其初始化值：

  MD11270 $MN_DEFAULT_VALUES_MEM_MASK, BIT0 = 1
  若未設定機械參數，則會將全域使用者變數（GUD）設定為相關資料類型的不明確初始化值。

- 僅 INIPO（啟動）可定義具 NCK 有效範圍之全域使用者變數（GUD）的初始化時間。
- 除了 INIPO（啟動），INIRE（重置）或 INICF（新設定）可用於定義具 CHAN 有效範圍的全域使用者變數（GUD）的初始化時間。
- 在具有 CHAN 有效範圍的全域使用者變數（GUD）及 INIRE（重置）或 INICF（新設定）初始化時間的情況中，當 NC 重置、模式群組重置及新設定時，這些變數僅可在已命名事件觸發時方可在通道中重新初始化。

初始化值：框架資料類型

不可對框架資料類型的變數指定初始化值。框架資料類型的變數不會明確的初始化且通常配合預設框架進行。

初始化值：字元資料類型

針對字元資料類型的變數，對應的 ASCII 字元可程式設計在雙引號中，而不需使用 ASCII 碼（0 至 255），例如"A"。

初始化值：STRING 資料類型

在字串資料類型的變數中，字串需包覆在雙引號中，例如...= "MASCHINE_1"

初始化值：軸資料類型

在軸資料類型的變數中，為表示延伸位址，需將軸識別碼包覆在括號中，例如...=（X3）。

初始化值：系統變數

針對系統變數，重新定義功能不可用於定義使用者專屬初始化值。系統變數的初始化值有系統指定且不可變更。然而，重新定義可用於在系統變數重新初始化時變更時間點（INIRE、INICF）。

不明確初始化值：軸資料類型

針對軸資料類型的變數，會使用以下的不明確初始化值：

- 系統資料："第一幾何軸"
- 同步 GUD（指定：SYG_A*、PUD、LUD：
  取自機械參數的軸識別碼：MD20082 $MC_AXCONF_CHANAX_DEFAULT_NAME

不明確初始化值：刀具與刀庫資料

刀具與刀庫資料的初始化值可用以下機械參數加以定義：MD17520 $MN_TOOL_DEFAULT_DATA_MASK

注意
同步 當全域變數讀入至不同位置為使用者/機台製造商的專屬責任時，由重新初始化該變數所觸發的事件之同步。

1.1.8 屬性：臨界值（LLI、ULI）

定義範圍的上下臨界值僅可定義為下列資料類型：

- INT
- REAL
- CHAR

定義（DEF）使用者變數：臨界值及不明確初始化值

若在定義上述資料類型的使用者變數時未定義明確的初始化值，該變數會設定為該資料類型的不明確初始化值。

- INT：0
- REAL：0.0
- CHAR：0

若該不明確初始化值超出程式設計的臨界值所定義之範圍，則該變數為以最接近不明確初始化值的臨界值進行初始化：

- 不明確初始化值<下臨界值（LLI）⇒
  初始化值=下臨界值
- 不明確初始化值<上臨界值列（ULI）⇒
  初始化值=上臨界值

範例：

程式碼	註解
DEF REAL GUD1	; 下臨界值=定義範圍臨界值
	; 上臨界值=定義範圍臨界值
	; 初始化值=
	; 不明確初始化值=0.0
DEF REAL LLI 5.0 GUD2	; 下臨界值= 10.0
	; 上臨界值=定義範圍臨界值
	; 初始化值=10.0
DEF REAL ULI –5 GUD3	; 下臨界值=定義範圍臨界值
	; 上臨界值= -10.0
	; 初始化值=-10.0

重新定義（DEF）使用者變數：臨界值與目前實際值

當重新定義了使用這變數的臨界值時，若其變更使目前實際值超出新定義的範圍，則會發出警報並拒絕該臨界值。

說明

重新定義（DEF）使用者變數

當重新定義了使用者變數的臨界值時，需特別小心以確保後續的值均能一致地變更：

- 極限值
- 實際值
- 以 INIPO、INIRE 或 INICF 為基礎重新定義或自動重新初始化時的初始化值

1.1.9 屬性：實體元件（PHU）

僅有下列資料類型的變數可程式設計實體元件：

- INT
- REAL

可程式設計的實體元件（PHU）

將該實體元件定義為固定點編號。
可程式設計的實體元件如下：

<元件>	意義	實體元件
0	非實體元件	-
1	線性或角度位置 1) 2)	[mm]、[inch]、[degree]
2	線性位置 2)	[mm]、[inch]
3	角度位置	[degree]
4	線性或角度速率 1) 2)	[mm/min]、[英吋 / 分]、[rpm]
5	線性速率 2)	[mm/min]
6	轉速	[rpm]
7	線性或旋轉加速度 1) 2)	[m/s^2]、[$inch/s^2$]、[rev/s^2]
8	線性加速度 2)	[m/s^2]、[$inch/s^2$]
9	旋轉加速度	[rev/s^2]
10	線性或旋轉加加速 1) 2)	[m/s^3]、[$inch/s^3$]、[rev/s^3]
11	線性加加速度 2)	[m/s^3]、[$inch/s^3$]
12	角度加加速度	[rev/s^3]
13	時間	[s]
14	位置控制器增益	[16.667/s]
15	迴轉進給率 2)	[mm/rev]、[inch/rev]
16	公制、英制 1) 2)	[mm]、[inch]
18	力	[N]
19	重量	[kg]
20	重量慣性矩 3)	[kgm^2]
21	百分比	[%]
22	頻率	[Hz]
23	電壓	[V]
24	電流	[A]
25	溫度	[°C]
26	角度	[degree]
27	KV	[1000/min]
28	線性或角度位置 3)	[mm]、[inch]、[degree]
29	切削率 2)	[m/min]、[feet/min]
30	圓周速率 2)	[m/s]、[feet/s]
31	阻抗	[ohm]
32	電感	[mH]

<元件>	意義	實體元件
33	扭矩 3)	[Nm]
34	扭矩常數 3)	[Nm/A]
35	電流控制器增益	[V/A]
36	速度控制器增益 3)	[Nm/(rad*s)]
37	轉速	[rpm]
42	功率	[kW]
43	電流，低	[µA]
46	扭矩，低 3)	[µNm]
48	每 mil	-
49	-	[Hz/s]
65	流率	[l/min]
66	壓力	[bar]
67	體積 3)	[cm^3]
68	控制系統增益 3)	[mm/(V*min)]
69	力控制器控制系統增益	[N/V]
155	螺距 3)	[mm/rev]、[inch/rev]
156	螺距變化 3)	[mm/rev / rev]、[inch/rev / rev]
1）實體元件視軸類型而定：線性或旋轉軸。		
2）該變數會自動轉換成 NC 目前的度量衡系統（英制/公制）。		
2）該變數**不會**自動轉換成 NC 目前的度量衡系統（英制/公制）。轉換為使用者/機台製造商的工作。		

說明

因格式轉換而造成溢位

所有具實體元件長度的使用者變數（GUD/PUD/LUD）之內部儲存格式為公制。在 NCK 的主執行中，例如同步動作，過度使用這些類型的變數，會在切換度量衡系統時在插補階段造成 CPU 時間溢位，而產生警報 4240。

注意

元件的相容性

使用變數（配置、比較、計算等）時，需檢查相關元件的相容性。必要時需轉換，此為使用者/機台製造商的責任。

1.1.10 屬性： 存取權力（APR、APW、APRP、APWP、APRB、APWB）

在程式設計時需指定下列與存取權限對應的保護等級：

存取權益	保護等級
系統密碼	0
機台製造商密碼	1
服務密碼	2
最終使用者密碼	3
按鍵操作開關位置 3	4
按鍵操作開關位置 2	5
按鍵操作開關位置 1	6
按鍵操作開關位置 0	7

定義（DEF）使用者變數

以下變數可定義存取權限（APR.../APW...）：

- 全域使用者資料（GUD）

重新定義（REDEF）系統及使用者變數

以下變數可重新定義存取權限（APR.../APW...）：

- 系統資料
  - 機械參數
  - 設定資料
  - FRAME
  - 製程資料
  - 導螺桿誤差補正資料（LEC）
  - 懸垂補正（CEC）
  - 象限錯誤補正（QEC）
  - 刀庫資料
  - 刀具資料
  - 保護區
  - 具有定向功能的刀把
  - 動態鍊
  - 3D 保護區
  - 工作區限制
  - ISO 刀具資料

- 使用者資料
  - R－參數
  - 同步動作變數（$AC_PARAM、$AC_MARKER、$AC_TIMER）
  - 同步動作 GUD（SYG_xy[ ]，其中 x=R、I、B、A、C、S 而 y=S、M、U、4 至 9）
  - EPS 參數
  - 刀具資料 OEM
  - 刀庫資料 OEM
  - 全域使用者變數（GUD）

說明

在重新定義時，可任意的將該存取權限指派給介於最低保護等級 7 及特定保護等級，例如 1（機台製造商），間的變數中。

重新定義（REDEF）NC 語言指令

下列 NC 語言指令的存取或執行權限（APX）可重新定義：

- G 系列功能 / 預備函數

  參考：

  /PG/ 基礎程式設計手冊；章節： G 系列功能 / 預備函數
- 預先定義之函數

  參考：

  /PG/ 基礎程式設計手冊；章節： 預先定義之函數
- 預定義之副程式呼叫

  參考：

  /PG/ 基礎程式設計手冊；章節： 預定義之副程式呼叫
- DO 搭配同步動作操作
- 循環程式識別碼

  該循環需儲存在循環目錄中且必須含 PROC 操作。

與工件程式與循環有關的存取權限（APRP, APWP）

不同的存取權限可依工件程式或循環中的存取加速以下項目：

- APRP 0/APWP 0
  - 在工件程式處理時需設定系統密碼。
  - 該循環需儲存在_N_CST_DIR 目錄（系統）中。
  - 該執行權限需設定在系統_N_CST_DIR 目錄的 MD11160 $MN_ACCESS_EXEC_CST 中。
- APRP 1/APWP 1 或 APRP 2/APWP 2
  - 在工件程式處理時需設定機台製造商或服務密碼。
  - 該循環需儲存在_N_CMA_DIR（機台製造商）或_N_CST_DIR（系統）目錄中。
  - 該執行權限至少需設定在機台製造商之_N_CMA_DIR 或_N_CST_DIR 目錄的機械參數 MD11161 $MN_ACCESS_EXEC_CMA 或 MD11160 $MN_ACCESS_EXEC_CST 中。
- APRP 3/APWP 3
  - 在工件程式處理時需設定最終使用者密碼。
  - 該循環需儲存在_N_CUS_DIR（使用者）、_N_CMA_DIR 或_N_CST_DIR 目錄中。
  - 該執行權限至少需設定在最終使用者之_N_CUS_DIR、_N_CMA_DIR 或_N_CST_DIR 目錄的機械參數 MD11162 $MN_ACCESS_EXEC_CUS、MD11161 $MN_ACCESS_EXEC_CMA 或 MD11160 $MN_ACCESS_EXEC_CST 中。
- APRP 4 至 7/APWP 4 至 7
  - 在工件程式處理時需將按鍵操作開關設定為 3 ... 0。
  - 該循環需儲存在_N_CUS_DIR、_N_CMA_DIR 或_N_CST_DIR 目錄中。
  - 該執行權限至少需設定在對應按鍵操作開關位置之_N_CUS_DIR、_N_CMA_DIR 或 _N_CST_DIR 目錄的機械參數 MD11162 $MN_ACCESS_EXEC_CUS、MD11161 $MN_ACCESS_EXEC_CMA 或 MD11160 $MN_ACCESS_EXEC_CST 中。

與 OPI 有關的存取權限（APRB、APWB）

該存取權限（APRB、APWB）透過 OPI 平等的限制了所有系統元件 （HMI、PLC、外部計算機、EPS 服務、等）對系統的存取及使用者變數。

說明

局部 HMI 存取權限

當變更對系統資料的存取權限時，需特別小心以確保這些變更與使用 HMI 機制所定義的存取權限一致。

APR/APW 存取屬性

APR/APW 存取屬性對應如下：

APR 0 至 7	⇒	APRP 0 至 7
APW 0 至 7	⇒	APWP 0 至 7
APR 10 至 17	⇒	APRB 0 至 7
APW 10 至 17	⇒	APWB 0 至 7

說明

影響 APR 的限制

當使用 APR 時需謹記以下限制：

- 對於工件程式/具 APR 0 至 7 之循環中的**讀取**系統及使用者變數的存取權限限制無效。
- 對於透過使用 APR 10 至 17 的 OPI 之讀取系統與使用者變數之存取限制僅在從 HMI 進行存取時方有效。 從其他元件（PLC、外部計算機、EPS 服務等）存取並不受限。

使用 ACCESS 檔案設定存取權限

當使用 ACCESS 檔案指派存取權限時，對於系統資料、使用者資料、及 NC 語言指令之存取權限的重新定義僅能接續程式設計在這些 ACCESS 檔案中。 但全域使用者資料（GUD）除外。 針對此資料，需繼續在對應的定義檔案中重新定義（若有此必要）存取權限。

為持續進行存取保護，該執行權限的機械參數及對應目錄的存取保護需一併修改。

原則上，步驟如下：

- 建立必要的定義檔案：
  - _N_DEF_DIR/_N_SACCESS_DEF
  - _N_DEF_DIR/_N_MACCESS_DEF
  - _N_DEF_DIR/_N_UACCESS_DEF
- 將定義檔案的寫入權限設定為重新定義所需要的值：
  - MD11170 $MN_ACCESS_WRITE_SACCESS
  - MD11171 $MN_ACCESS_WRITE_MACCESS
  - MD11172 $MN_ACCESS_WRITE_UACCESS

- 為存取循環中受保護的元件，需修改循環目錄_N_CST_DIR、_N_CMA_DIR 以及 _N_CST_DIR 的執行與寫入權限。

  執行權限

  – MD11160 $MN_ACCESS_EXEC_CST
  – MD11161 $MN_ACCESS_EXEC_CMA
  – MD11162 $MN_ACCESS_EXEC_CUS

  寫入權限

  – MD11165 $MN_ACCESS_WRITE_CST
  – MD11166 $MN_ACCESS_WRITE_CMA
  – MD11167 MN_ACCESS_WRITE_CUS

  該執行權限的保護等級至少需設定為與該元件所使用的保護等級相同。

  該寫入權限的保護等級至少需設定為該執行權限所使用的相同。

- 局部 HMI 循環目錄之寫入權限的保護等級至少需設定為與局部 NC 循環目錄所使用的相同。

  參考

  /BAD/ 操作手冊，HMI Advanced，

  一章： 服務操作區 > 資料管理 > 修改特性

ACCESS 檔案中的副程式呼叫

為進一步建構存取保護，可在 ACCESS 檔案中呼叫（SPF 或 MPF 識別碼）副程式。 該副程式繼承了呼叫之 ACCESS 檔案的存取權限。

說明

ACCESS 檔案中僅能重新定義存取權限。 其他的屬性需繼續在對應的定義檔案中程式設計/重新定義。

1.1.11 可定義與可重新定義屬性之概觀

下列表顯示出可定義的屬性（DEF）與對資料類型重定義的（REDEF）。

系統資料

資料類型	初始值	極限值	實體元件	存取權限
機械參數	---	---	---	REDEF
設定資料	REDEF	---	---	REDEF
框架資料	---	---	---	REDEF
製程資料	---	---	---	REDEF
導螺桿錯誤元件（EEC）	---	---	---	REDEF
懸垂補正（CEC）	---	---	---	REDEF
象限錯誤補正（ QEC）	---	---	---	REDEF
刀庫資料	---	---	---	REDEF
刀具資料	---	---	---	REDEF
保護區	---	---	---	REDEF
刀把，具定向功能	---	---	---	REDEF
動態鍊	---	---	---	REDEF
3D 保護區	---	---	---	REDEF
工作區限制	---	---	---	REDEF
ISO 刀具資料	---	---	---	REDEF

使用者資料

資料類型	初始值	極限值	實體元件	存取權限
R-參數	REDEF	REDEF	REDEF	REDEF
同步動作變數（$AC_...）	REDEF	REDEF	REDEF	REDEF
同步動作 GUD（SYG_...）	REDEF	REDEF	REDEF	REDEF
EPS 參數	REDEF	REDEF	REDEF	REDEF
刀具資料 OEM	REDEF	REDEF	REDEF	REDEF
刀庫資料 OEM	REDEF	REDEF	REDEF	REDEF
全域使用者變數（GUD）	DEF/REDEF	DEF	DEF	DEF/REDEF
區域使用者變數（PUD/LUD）	DEF	DEF	DEF	---

1.1.12 陣列變數（DEF，SET，REP）之定義與初始化

功能

使用者變數可定義成 1 維至最多 3 維陣列。

- 1 維：DEF <資料類型> <變數名稱>[<n>]
- 2 維：DEF <資料類型> <變數名稱>[<n>，<m>]
- 3 維：DEF<資料類型> <變數名稱>[<n>，<m>，<o>]

說明

STRING 資料類型使用者變數可被定義成最多 2 維陣列。

資料類型

使用者變數可被定義成供下列資料類型所用的陣列：BOOL，CHAR，INT，REAL，STRING，AXIS，FRAME

指派值給陣列元件

可在下列時間即時將值指派給陣列元件：

- 在陣列定義期間（初始值）
- 在程式執行期間

可藉由下列方式指派值：

- 陣列元件的明確規格
- 作為啟動元件與值清單的規格之陣列元件的明確規格（SET）
- 作為啟動元件、值的規格及其重複之處的頻率之陣列元件的明確規格（REP）

說明

框架資料類型使用者變數，無法被指派為初始值。

句法（DEF）

```
DEF<資料類型> <變數名稱>[<n>，<m>，<o>]
DEF   STRING[<字串長度>] <變數名稱>[<n>，<m>]
```

句法（**DEF...=SET...**）

使用值清單：

- 在定義期間：
  DEF <資料類型> <變數名稱>[<n>，<m>，<o>]=SET（<值 1>，<值 2>，等等）
  等同於：
  DEF <資料類型> <變數名稱>[<n>，<m>，<o>]=SET（<值 1>，<值 2>，等等）

  說明
  SET 不需透過值清單來指定初始化。

- 在值指派期間：
  <變數名稱>[<n>，<m>，<o>]=SET（<值 1>，<值 2>，等等）

句法（**DEF...=REP...**）

使用具有重複的值

- 在定義期間：
  DEF <資料類型> <變數名稱>[<n>，<m>，<o>]=REP（<值>）

  DEF <資料類型> <變數名稱>[<n>，<m>，<o>]=REP（<值>，<number_array_elements>）

- 在值指派期間：
  <變數名稱>[<n>，<m>，<o>]=REP（<值>）
  DEF <資料類型> <變數名稱>[<n>，<m>，<o>]=REP（<值>，<number_array_elements>）

意義

DEF：	用來定義變數的指令
<資料類型>：	變數的資料類型 值域： • 供系統變數 BOOL，CHAR，INT，REAL，STRING，AXIS • 對於 GUD 或 LUD 變數： BOOL，CHAR，INT，REAL，STRING，AXIS，FRAME
<字串長度>：	STRING 資料類型的最大字元數
<檔案名稱>：	變數名稱。
[<n>，<m>，<o>]：	陣列大小或陣列索引
<n>：	1 維的陣列大小或陣列索引 類型： INT（對於系統變數，以及 AXIS） 值域： 最大陣列大小：65535 陣列索引：0 ≤ n ≤ 65534

<m>:	2 維的陣列大小或陣列索引 類型： INT（對於系統變數，以及 AXIS） 值域： 最大陣列大小：65535 陣列索引：0 ≤ m ≤ 65534
<o>:	3 維的陣列大小或陣列索引 類型： INT（對於系統變數，以及 AXIS） 值域： 最大陣列大小：65535 陣列索引：0 ≤ o ≤ 65534
SET: （<值 1>，<值 2>，等等）:	使用特定值清單的值指派 值清單
REP: <值>:	值指派，使用特定<值> 以 REP 初始化時，陣列元素應被寫入的值。
<number_array_elements>	要以特定值寫入的陣列元素數目，之特定<值>.下列對於剩餘陣列元件的使用，依時間點而定： • 在定義陣列時初始化： →零點被寫入至剩餘的陣列元件。 • 在程式執行期間的指派： →陣列元素的實際值保持不變。 若參數未程式設計，則所有陣列元件都會被寫入<值>。 若參數等於零點，則下列應用依時間點而定： • 在定義陣列時初始化： →所有元素皆被預指派為零 • 在程式執行期間的指派： →陣列元素的實際值保持不變。

陣列索引

陣列元件的明確順序，例如在使用 SET 或 REP 進行值指派時，因為陣列索引的重複，故為由右至左。

範例：具有 24 個陣列元件的 3 維陣列之初始化：

```
DEF INT FELD[2, 3, 4] = REP(1, 24)
  FELD[0, 0, 0] = 1    第 1 陣列元件
  FELD[0, 0, 1] = 1    第 2 陣列元件
  FELD[0, 0, 2] = 1    第 3 陣列元件
  FELD[0, 0, 3] = 1    第 4 陣列元件
  ...
  FELD[0, 1, 0] = 1    第 5 陣列元件
  FELD[0, 1, 1] = 1    第 6 陣列元件
  ...
  FELD[0, 2, 3] = 1    第 12 陣列元件
  FELD[1, 0, 0] = 1    第 13 陣列元件
  FELD[1, 0, 1] = 1    第 14 陣列元件
  ...
  FELD[1, 2, 3] = 1    第 24 陣列元件
```

對應至：

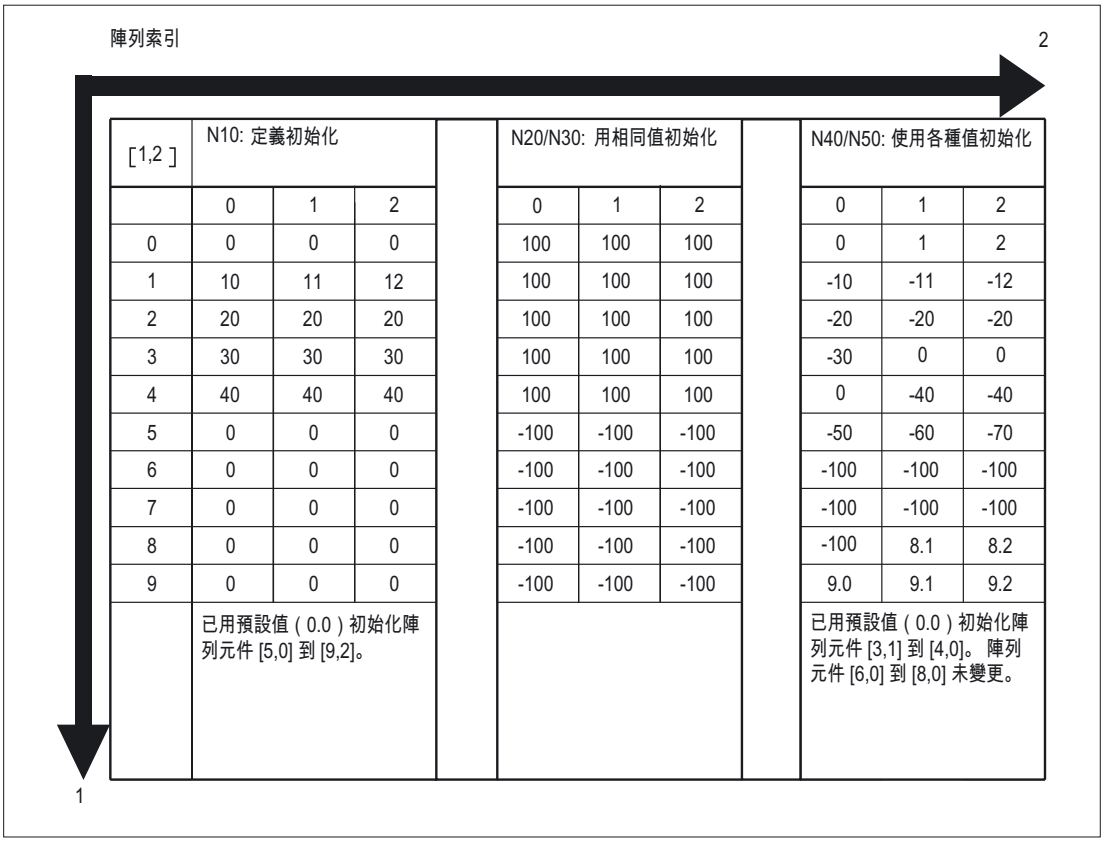
```
FOR n=0 TO 1
  FOR m=0 TO 2
    FOR o=0 TO 3
      FELD[n, m, o] = 1
    ENDFOR
  ENDFOR
ENDFOR
```

範例：初始化完整的變數陣列

對於實際指派情形，請參考圖示。

程式碼

```
N10 DEF REAL FELD1[10, 3]=SET(0, 0, 0, 10, 11, 12, 20, 20, 20, 30, 30, 30, 40, 40, 40, )
N20 ARRAY1[0, 0] = REP(100)
N30 ARRAY1[5, 0] = REP(-100)
N40 FELD1[0, 0]=SET(0, 1, 2, -10, -11, -12, -20, -20, -20, -30, , , , -40, -40, -50, -60, -70)
N50 FELD1[8, 1]=SET(8.1, 8.2, 9.0, 9.1, 9.2)
```



[1,2]	N10: 定義初始化			N20/N30: 用相同值初始化			N40/N50: 使用各種值初始化		
	0	1	2	0	1	2	0	1	2
0	0	0	0	100	100	100	0	1	2
1	10	11	12	100	100	100	-10	-11	-12
2	20	20	20	100	100	100	-20	-20	-20
3	30	30	30	100	100	100	-30	0	0
4	40	40	40	100	100	100	0	-40	-40
5	0	0	0	-100	-100	-100	-50	-60	-70
6	0	0	0	-100	-100	-100	-100	-100	-100
7	0	0	0	-100	-100	-100	-100	-100	-100
8	0	0	0	-100	-100	-100	-100	8.1	8.2
9	0	0	0	-100	-100	-100	9.0	9.1	9.2
	已用預設值 (0.0) 初始化陣列元件 [5,0] 到 [9,2]。						已用預設值 (0.0) 初始化陣列元件 [3,1] 到 [4,0]。 陣列元件 [6,0] 到 [8,0] 未變更。		

詳細資訊（SET）

在定義期間初始化

- 從第 1 陣列元件開始，因為在值清單中有已程式設計的值清單，所以有許多陣列元件，會以值清單中的值來指派。
- 將值 0 指派至陣列元件，沒有在值清單中明確宣告值（值清單中的空白處）。
- 對於 AXIS 類型的變數，不允許在值清單中有空白處存在。
- 若值清單中包含的值，超過了已定義的陣列元件時，會顯示警報。

在程式執行期間的值指派

在程式設計期間的值指派時，上述的規則係用於定義之。下列選項亦提供支援：

- 算式亦可作為值清單中的元件。
- 值指派會以已程式設計的陣列索引開始。可以將值選擇性指派到子陣列中。

範例：

程式碼	註解
`DEF INT ARRAY[5, 5]`	`;` 陣列定義
`ARRAY[0, 0]=SET(1, 2, 3, 4, 5)`	`;` 值指派至前 5 個陣列元件[0, 0] - [0, 4]
`FELD[0, 0]=SET(1, 2, , , 5)`	`;` 具有空白處的值指派至前 5 個陣列元件[0, 0] - [0, 4]，陣列元件[0, 2] and [0, 3] = 0
`ARRAY[2, 3]=SET(VARIABLE, 4*5.6)`	`;` 具有變數的值指派和在陣列索引[2, 3]啟動的算式： [2, 3] = VARIABLE [2, 4] = 4 * 5.6 = 22.4

詳細資訊（REP）

在定義期間初始化

- 所有或可選擇的特定數量的陣列元件，以特定值（常數）初始化。
- 無法初始化 FRAME 類型的變數。

範例：

程式碼	註解
`DEF REAL varName[10]=REP(3.5, 4)`	`;` 初始化陣列定義與陣列元件[0]至[3]具有值 3.5。

在程式執行期間的值指派

在程式設計期間的值指派時，上述的規則係用於定義之。下列選項亦提供支援：

- 算式亦可作為值清單中的元件。
- 值指派會以已程式設計的陣列索引開始。可以將值選擇性指派到子陣列中。

範例：

程式碼	註解
DEF REAL varName[10]	; 陣列定義
varName[5]=REP（4.5，3）	; 陣列元件[5]至[7] = 4.5
R10=REP（2.4，3）	; R 參數 R10 至 R12 = 2.4
DEF FRAME FRM[10]	; 陣列定義
FRM[5] = REP（CTRANS （X，5））	; 陣列元件[5]至[9] = CTRANS（X，5）

其它資訊（一般）

值指派至軸機械參數

基本上，軸機械參數具有一個為軸資料類型的陣列索引。在值指派至機械參數的軸項目時，使用 SET 或 REP，此陣列索引會被忽略或不被處理。

範例：值指派至機械參數 MD36200 $MA_AX_VELO_LIMIT

```
$MA_AX_VELO_LIMIT[1，AX1]=SET（1.1，2.2，3.3）
```

等同於：

```
$MA_AX_VELO_LIMIT[1，AX1] = 1.1
$MA_AX_VELO_LIMIT[2，AX1] = 2.2
$MA_AX_VELO_LIMIT[3，AX1] = 3.3
```

注意

值指派至軸機械參數

在值指派至機械參數的軸項目時，使用 SET 或 REP，軸資料 類型陣列索引會被忽略或不被處理。

記憶體需求

資料類型	各元素所需記憶體
BOOL	1 個位元組
CHAR	1 個位元組
INT	4 個位元組
REAL	8 個位元組
STRING	（字串長度+ 1）位元組
FRAME	~ 400 位元，視軸的數目而定
AXIS	4 個位元組

1.1.13 資料類型

NC 有下列可用的資料類型：

資料類型	意義	值域
INT	含正負號之整數	-2147483648 ... +2147483647
REAL	實數（LONG REAL 至 IEEE）	±（~2，$2*10^{-308}$ … ~1，$8*10^{+308}$）
BOOL	真值 TRUE（真）（=1）而 FALSE（偽）（=0）	1, 0
CHAR	ASCII 字元	ASCII 代碼 0 至 255
STRING	已定義長度的字元字串	最多 200 個字元（無特殊字元）
AXIS	軸 / 定位主軸	通道軸識別碼
FRAME	用於靜態座標轉換的幾何參數（轉譯，旋轉，刻度，鏡像）	---

明確的資料類型轉換

下列的資料類型轉換是可行的，且會在指派和參數轉換期間明確執行：

從↓/至→	REAL	INT	BOOL
REAL	x	o	&
INT	x	x	&
BOOL	x	x	x
x：可行，無限制 o：因為值域過衝的可能的資料損失⇒警報； 倒圓角：二進制位置值≥ 0.5 ⇒將進位，二進制位置值< 0.5 ⇒將捨去 &: 值≠ 0 ⇒ TRUE，值== 0 ⇒ FALSE			

1.2 間接程式設計

功能

間接程式設計位址時，會以適當類型的變數替換延伸位址（索引）。

說明

無法將間接程式位址供以下使用：
- N（單節號碼）
- L（副程式）
- 可設定的位址
  （例如 X[1]不可以取代 X1）

句法

```
<ADDRESS>[<索引>]
```

意義

`<ADDRESS>[...]`:	具有副檔名（索引）的固定位址
`<索引>`:	索引變數，例如，主軸號碼，座標軸，....

範例

範例 1：間接程式設計一個主軸號碼

直接程式設計：

程式碼	註解
`S1=300`	`;` 1 號主軸的 rpm 速度。

間接程式設計：

程式碼	註解
`DEF INT SPINU=1`	`;` 定義變數，類型 INT，以及值的指派。
`S[SPINU]=300`	`;` 主軸速度為 300 rpm，該數值儲存在 SPINU 變數（在此範例 1 中，採用的是 1 號主軸）。

範例 2：間接程式設計軸

直接程式設計：

程式碼	註解
FA[U]=300	; 將座標軸 “U”的進給率指定為 300。

間接程式設計：

程式碼	註解
DEF AXIS AXVAR2=U	; 定義變數，類型 AXIS，以及值的指派。
FA[AXVAR2]=300	; 把進給率 300，指定給座標軸，該座標軸係以 AXVAR2 為位址名稱，儲存在變數中。

範例 3：間接程式設計軸

直接程式設計：

程式設計	註解
$AA_MM[X]	; 讀取座標軸 “X”的探針測量值（MCS）。

間接程式設計：

程式碼	註解
DEF AXIS AXVAR3=X	; 定義變數，類型 AXIS，以及值的指派。
$AA_MM[AXVAR3]	; 讀取探針測量值，（MCS），其名稱係儲存在變數 AXVAR3 中。

範例 4：間接程式設計軸

直接程式設計：

程式碼
X1=100 X2=200

間接程式設計：

程式碼	註解
DEF AXIS AXVAR1 AXVAR2	; 定義兩個 AXIS 類型的變數。
AXVAR1=（X1） AXVAR2=（X2）	; 指派座標軸名稱。
AX[AXVAR1]=100 AX[AXVAR2]=200	; 移動座標軸，該座標軸係以 AXVAR1 和 AXVAR2 作為位址名稱，儲存在變數中。

範例 5：間接程式設計軸

直接程式設計：

程式碼
G2 X100 I20

間接程式設計：

程式碼	註解
DEF AXIS AXVAR1=X	; 定義變數，類型 AXIS，以及值的指派。
G2 X100 IP[AXVAR1]=20	; 對座標軸進行間接程式設計中心點資料，該座標軸係以 AXVAR1 作為位址名稱，儲存於變數中。

範例 6：間接程式設計陣列元素

直接程式設計：

程式碼	註解
DEF INT ARRAY1[4，5]	; 定義陣列 1

間接程式設計：

程式碼	註解
DEFINE DIM1 AS 4	; 對於陣列維度，陣列大小必須指派為固定值。
DEFINE DIM2 AS 5	
DEF INT ARRAY[DIM1，DIM2]	
ARRAY[DIM1-1，DIM2-1]=5	

範例 7：間接副程式呼叫

程式碼	註解
CALL "L" << R10	; 呼叫程式，該程式之號碼位於 R10（字串串連）。

1.2.1 間接程式設計 G 代碼

功能

間接程式設計 G 代碼，允許對循環進行有效的程式設計。

句法

G[<群組>]=<編號>

意義

G[...]：	具有副檔名（索引）的 G 指令
<群組>：	索引參數：G 碼功能群組 類型： INT **注意：** 只有模態 G 碼功能群組可以進行間接程式設計。有效進行順時針方向的 G 碼功能群組，會被拒絕，並發出警報。
<號碼>：	供 G 代碼數字所用的變數 類型： INT 或 REAL **注意：** 在間接 G 代碼程式設計中，不允許有算術函數。若要計算 G 代碼數字，則必須在進行間接 G 代碼程式設計之前，在分開的工件程式行先行計算完成。

範例

範例 1：可設定的零點偏移量（G 碼功能群組 8）

```
程式碼                                  註解
N1010 DEF INT INT_VAR
N1020 INT_VAR = 2
...
N1090 G[8]=INT_VAR G1 X0 Y0             ; G54
N1100 INT_VAR=INT_VAR+1                 ; G 代碼計算
N1110 G[8]=INT_VAR G1 X0 Y0             ; G55
```

範例 2：層級選擇（G 碼功能群組 6）

```
程式碼                                  註解
N2010 R10=$P_GG[6]                      ; 讀取 G 碼功能群組 6 的主動 G 碼功能
...
N2090 G[6]=R10
```

參考

若要 G 碼功能群組的資訊，請參考：
基礎程式設計手冊；“G 碼功能清單 / 預備函數”一章

1.2.2 間接程式設計定位屬性（BP）

功能

定位屬性，例如，座標軸定位的增量或絕對程式設計，可以連同關鍵字組間接程式設計為變數 BP。

應用

定位屬性的間接程式設計係用於**替換循環**，在此情況中，程式設計定位屬性為關鍵字組（例如 IC，AC，...），會有下列優點：

由於間接程式設計為變數，因此**不需要** CASE 指令，否則將為所有可能的定位屬性分支。

句法

```
<POSITIONING COMMAND>[<軸/主軸>]=
BP（<位置>，<位置屬性）
<軸/主軸>=BP（<位置>，<position attribute）
```

意義

<POSITIONING COMMAND>[]:	下列定位指令可以和關鍵字組 P 一起程式設計： POS，POSA，SPOS，SPOSA 也有可能： • 所有的座標軸和主軸識別碼在通道中出現： <軸/主軸> • 變數座標軸 / 主軸識別碼 AX
<軸/主軸>:	待定位的座標軸 / 主軸
GP（）:	用以定位的關鍵字
<位置>:	參數 1 座標軸 / 主軸定位為常數或變數
<定位屬性>	參數 2 定位屬性（例如，定位逼近模式為變數（例如，$P_SUB_SPOSMODE）或為關鍵字組 IC，AC，...）

由變數所提供的值，具有下列特性：

值	意義	允許給：
0	定位屬性保持不變	
1	AC	POS，POSA，SPOS，SPOSA，AX，座標軸位址
2	IC	POS，POSA，SPOS，SPOSA，AX，座標軸位址
3	DC	POS，POSA，SPOS，SPOSA，AX，座標軸位址
4	ACP	POS，POSA，SPOS，SPOSA，AX，座標軸位址
5	ACN	POS，POSA，SPOS，SPOSA，AX，座標軸位址
6	OC	-
7	PC	-
8	DAC	POS，POSA，AX， 座標軸位址
9	DIC	POS，POSA，AX， 座標軸位址
10	RAC	POS，POSA，AX， 座標軸位址
11	RIC	POS，POSA，AX， 座標軸位址
12	CAC	POS，POSA
13	CIC	POS，POSA
14	CDC	POS，POSA
15	CACP	POS，POSA
16	CACN	POS，POSA

範例

對於介於螺距主軸 S1 和後續主軸 S2 之間的主動同步主軸耦合，會使用主程式中的 SPOS 指令，呼叫後續的替換循環，來定位主軸。

定位是使用 N2230 中的指令來完成：
SPOS[1]=GP（$P_SUB_SPOSIT，$P_SUB_SPOSMODE） SPOS[2]=GP（$P_SUB_SPOSIT，$P_SUB_SPOSMODE）

待逼近的定位，會由從系統變數$P_SUB_SPOSIT 中來讀取；而定位逼近模式，則是從系統變數$P_SUB_SPOSMODE 中讀取。

程式碼	註解
N1000 PROC LANG_SUB DISPLOF SBLOF	
...	
N2100 IF（$P_SUB_AXFCT==2）	
N2110	; SPOS / SPOSA / M19 指令的替換，供主動同步主軸耦合所用。
N2185 DELAYFSTON	; 開始停止延遲區域
N2190 COUPOF（S2，S1）	; 停用同步主軸耦合
N2200	; 定位螺距與後續主軸
N2210 IF（$P_SUB_SPOS==TRUE） OR （$P_SUB_SPOSA==TRUE）	
N2220	; 以 SPOS 定位主軸：
N2230 SPOS[1]=GP（$P_SUB_SPOSIT，$P_SUB_SPOSMODE）	
SPOS[2]=GP（$P_SUB_SPOSIT，$P_SUB_SPOSMODE）	
N2250 ELSE	

程式碼	註解
N2260	; 使用 M19 定位主軸：
N2270 M1=19 M2=19	; 定位螺距與後續主軸
N2280 ENDIF	
N2285 DELAYFSTOF	; 停止延遲區域終點
N2290 COUPON（S2，S1）	; 啟動同步主軸耦合
N2410 ELSE	
N2420	; 查詢進一步替換
...	
N3300 ENDIF	
...	
N9999 RET	

補充條件

- 定位屬性的間接程式設計，無法在同步動作中進行。

參考

功能手冊冊基本功能；BAG，通道，程式操作，重置回應（K1），
藉由副程式替換 NC 函數一章

1.2.3 間接程式設計工件程式行（EXECSTRING）

功能

工件程式指令 EXECSTRING，會傳遞一個字串，作為已包含了待執行的工件程式行之參數。

句法

```
EXECSTRING （<string_variable>）
```

參數

EXECSTRING:	字串變數的傳輸，該字串變數具有待執行的工件程式行。
<字串變數>:	具有實際上待執行的工件程式行之參數

說明

所有可以在工件程式中程式設計的工件程式語法，都可以輸出。這表示 PROC 與 DEF 指令也會和一般的使用一樣，被排除在 INI 與 DEF 檔案之外。

範例

間接工件程式行

程式碼	註解
N100 DEF STRING[100] BLOCK	; 用來接受工件程式行的字串變數
N110 DEF STRING[10] MFCT1="M7"	
N200 EXECSTRING（MFCT1 << "M4711"）	; 執行工件程式行“M7 M4711”
N300 R10=1	
N310 BLOCK="M3"	
N320 IF（R10）	
N330 BLOCK = BLOCK << MFCT1	
N340 ENDIF	
N350 EXECSTRING（BLOCK）	; 執行工件程式行“M3 M4711”

1.3 運算功能

功能

算術函數主要係供 REAL 類型的 R 參數與變數（或常數與函數）使用。 INT 和 CHAR 類型也為認可。

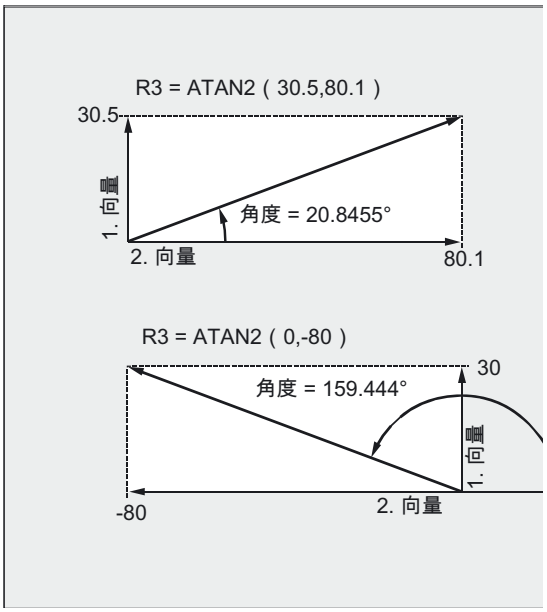
操作員 / 算術函數	意義
+	加法
-	減法
*	乘法
/	除法 **注意:** （INT 類型）/（INT 類型）=（REAL 類型）；範例： 3/4 = 0.75
DIV	除法，用於變數類型 INT 與 REAL **注意:** （INT 類型）DIV（INT 類型）=（INT 類型）；範例： 3 DIV 4 = 0
MOD	餘數法（限用於類型 INT）提供了 INT 除法的餘數 範例： 3 MOD 4 = 3
:	串鍊運算子（供 FRAME 變數所用）
Sin（）	正弦
COS（）	餘弦
TAN（）	正切
ASIN（）	反正弦函數
ACOS（）	反餘弦函數
ATAN2（ ， ）	反切線 2
SQRT（）	平方根
ABS（）	絕對值
POT（）	2. 次方（平方）
TRUNC（）	去尾數成整數 比較指令的精確度，可用TRUNC（參閱"比較錯誤（TRUNC）的精確性修正 (頁 62)"）來設定
ROUND（）	標準整數法
LN（）	自然對數
EXP（）	指數功能
MINVAL （）	兩變數的較小值 （請參閱"變數最小，最大與範圍（MINVAL，MAXVAL與BOUND） (頁 64)"）
MAXVAL （）	兩變數的較大值 （請參閱"變數最小，最大與範圍（MINVAL，MAXVAL與BOUND） (頁 64)"）
BOUND （）	在所定義值域中的變數值 （請參閱"變數最小，最大與範圍（MINVAL，MAXVAL與BOUND） (頁 64)"）
CTRANS（）	轉譯
CROT （）	旋轉
CSCALE（）	變更比例
CMIRROR（）	鏡射

程式設計

一般的算數標記法可用於算術函數。 執行的優先順序係由圓括號來表示。 角度係用於三角函數及其反函數（直角 = 90°）。

範例

範例 1：ATAN2



算術函數 ATAN2 會計算兩個互相垂直的向量之間的總向量角度。
結果會落在四個象限之一（–180° < 0 < +180°）。
角度的參考會根據正向的第二個值。

範例 2： 初始化全部的變數陣列

程式碼	註解
`R1=R1+1`	; 新 R1 =舊 R1 +1
`R1=R2+R3 R4=R5-R6 R7=R8*R9`	
`R10=R11/R12 R13=SIN（25.3）`	
`R14=R1*R2+R3`	; 先乘除後加減。
`R14=（R1+R2）*R3`	; 圓括號先運算。
`R15=SQRT（POT（R1）+POT（R2））`	; 內部圓括號先運算： R15 =（R1+R2）的平方根
`RESFRAME=FRAME1：FRAME2` `FRAME3=CTRANS（…）：CROT（…）`	; 串鍊運算子會連結框架 形成一個結果框架或指派值 給框架元件。

1.4 比較與邏輯運算

功能

舉例來說，**比較運算**可以用來制訂跳躍條件。也可比較複雜運算式。

比較運算適用於變數類型 CHAR，INT，REAL 與 BOOL。代碼值會與 CHAR 類型做比較。
對於 STRING，AXIS 與 FRAME 等類型，下列為可行的運算：==與<>，可用於 STRING 類型的運算，即使在同步動作中亦然。

比較運算的結果永遠會是 BOOL 類型。

邏輯運算子係用於連結真值。

邏輯運算僅能用於 BOOL 類型的變數。然而，透過內部類型轉換，他們還可用於 CHAR，INT 與 REAL 等資料類型。

對於邏輯（布林）運算，下列可用於 BOOL，CHAR，INT 與 REAL 等資料類型：

- 0 對應至：FALSE（偽）
- 不等於 0 即表示：TRUE（真）

位元邏輯運算子

邏輯運算也能用於 CHAR 與 INT 類型的單位元。類型轉換是自動進行的。

程式設計

關係運算子	**意義**
==	等於
<>	不等於
>	大於
<	小於
>=	大於或等於
<=	小於或等於

邏輯運算子	**意義**
AND	AND
OR	OR
NOT	否
XOR	排除 OR

位元邏輯運算子	**意義**
B_AND	位元串列 AND
B_OR	位元串列 OR
B_NOT	位元串列否
B_XOR	位元串列排除 OR

說明

在算術運算式中，可藉由圓括號，來指定各運算子的執行順序，藉此手動倍率一般規則的優先順序。

說明

在 BOOLEAN 運算元和運算子中間必須空一格。

說明

運算子 B_NOT，僅參照一個運算元。位於運算子之後。

範例

範例 1：關係運算子

```
IF R10>=100 GOTOF DEST
```

或

```
R11=R10>=100
IF R11 GOTOF DEST
```

R10>=100 的比較結果，會先暫存在 R11 之中。

範例 2：邏輯運算子

```
IF （R10<50） AND （$AA_IM[X]>=17.5） GOTOF DESTINATION
```

或

```
IF NOT R10 GOTOB START
```

NOT，僅參照一個運算元。

範例 3：位元邏輯運算子

```
IF $MC_RESET_MODE_MASK B_AND 'B10000' GOTOF  ACT_PLANE
```

1.5 比較錯誤（TRUNC）的精確性修正

功能

TRUNC 指令會去除運算元與精確係數相乘後的尾數。

用於比較指令的可設定的精確性

REAL 類型的程式資料會在 IEEE 格式中，以 64 位元的方式於內部顯示。這種顯示格式會造成顯示出來的小數位不夠精確，並會導致在與理想計算值進行比較時，出現無法預期的結果。

相對等式

為了防止這種因為顯示格式所導致的不精確情形，干擾了程式流程，比較指令會檢查相對等式，而非絕對等式。

句法

比較錯誤的精確性修正

```
TRUNC （R1*1000）
```

意義

TRUNC: 去掉小數點位數

相對等式 10^{-12} 會考慮到

- 等式：(==)
- 不等式：(<>)
- 大於或等於：(>=)
- 小於或等於：(<=)
- 大於或小於：（><）絕對等式
- 大於：(>)
- 小於：(<)

相容性

為求一致性，檢查相對等式（>）和（<），可以透過設定機械參數 MD10280 $MN_PROG_FUNCTION_MASK Bit0 = 1，來停用。

說明

與 REAL 類型的資料做比較，會因為上述原因而造成一定程度的不精確。若誤差超過了接受範圍，可藉由將一個精確係數與運算元相乘，使用 INTEGER 計算，然後再用 TRUNC 去掉尾數。

同步動作

定義比較指令的回應，也可套用至同步動作。

範例

範例 1：精確性考量

程式碼	註解
N40 R1=61.01 R2=61.02 R3=0.01	; 指派初始值
N41 IF ABS（R2-R1） > R3 GOTOF ERROR	; 跳躍一直執行
N42 M30	; 程式結尾
N43 ERROR: SETAL（66000）	;
R1=61.01 R2=61.02 R3=0.01	; 指派初始值
R11=TRUNC（R1*1000） R12=TRUNC（R2*1000） R13=TRUNC（R3*1000）	; 精確度校正
IF ABS（R12-R11） > R13 GOTOF ERROR	; 不再執行跳躍
M30	; 程式結尾
ERROR: SETAL（66000）	;

範例 2：計算並求出兩個運算元的商數。

程式碼	註解
R1=61.01 R2=61.02 R3=0.01	; 指派初始值
IF ABS（（R2-R1）/R3）-1） > 10EX-5 GOTOF ERROR	; 未執行跳躍
M30	; 程式結尾
ERROR: SETAL（66000）	;

1.6 變數最小，最大與範圍（MINVAL，MAXVAL 與 BOUND）

功能

MINVAL 與 MAXVAL 指令可用來比較兩變數之值。較小的值（在 MINVAL）的情形中，或較大的值（在 MAXVAL 的情形中）會各自被傳送。

BOUND 指令可以用來測試，測試變數的值是否落在已定義的值域中。

句法

<較小值>=MINVAL（<變數 1>，<變數 2>）
<較大值>=MAXVAL（<變數 1>，<變數 2>）
<傳回值>=<BOUND>（<最小>，<最大>，<測試變數>）

意義

MINVAL:	取得**二變數的較小** 值（<變數 1>，<變數 2>）
<較小值>:	用於 MINVAL 指令的結果變數 設定較小的變數值。
MAXVAL:	取得**二變數的較大** 值（<變數 1>，<變數 2>）
<較大值 >:	用於 MAXVAL 指令的結果變數 設定較大的變數值。
BOUND:	測試（<測試變數）這個變數是否在已定義的值域之中。
<最小>:	定義了值域中最小值的變數。
<最大>:	定義了值域中最大值的變數。
<傳回值 >:	用於 BOUND 指令的結果變數 若測試變數的值在已定義的值域之中，結果變數會設定為測試變數的值。 若測試變數的值大於最大值，結果變數會設定為定義域中的最大值。 若測試變數的值小於最小值，結果變數會設定為定義域中的最小值。

說明

MINVAL，MAXVAL，與 BOUND 也可同步程式設計。

說明

若值相等時的行為

若值相等，MINVAL/MAXVAL 會設定此相等值。在 BOUND 的情形中，要測試的變數值會再次傳回。

範例

程式碼	註解
DEF REAL rVar1 = 10.5，rVar2 = 33.7，rVar3，rVar4，rVar5，rValMin，rValMax，rRetVar	
rValMin=MINVAL（rVar1，rVar2）	; rValMin 設定為值 10.5。
rValMax=MAXVAL（rVar1，rVar2）	; rValMax 設定為值 33.7。
rVar3=19.7	
rRetVar=BOUND（rVar1，rVar2，rVar3）	; rVar3 在臨界值中，rRetVar 設定為 19.7。
rVar3=1.8	
rRetVar=BOUND（rVar1，rVar2，rVar3）	; rVar3 低於最小臨界值，rRetVar 設定為 10.5。
rVar3=45.2	
rRetVar=BOUND（rVar1，rVar2，rVar3）	; rVar3 高於最大臨界值，rRetVar 設定為 33.7。

1.7 運算優先順序

功能

每個運算子都已分配優先順序。當運算式已求出，具有最高優先權的運算子，永遠都要優先套用。.若運算子具有同等的優先順序，演算時必須由左至右。

在算術運算式中，可藉由圓括號，來指定各運算子的執行順序，藉此手動倍率一般的優先順序規則。

運算子的順序

優先順序由高至低

1.	NOT，B_NOT	否，位元串列否
2.	*，/，DIV，MOD	乘法，除法
3.	+, –	加法，減法
4.	B_AND	位元 AND
5.	B_XOR	位元串列排除 OR
6.	B_OR	位元串列 OR
7.	AND	AND
8.	XOR	排除 OR
9.	OR	OR
10.	<<	串鍊字串，結果 STRING 類型
11.	==, <>, >, <, >=, <=	比較運算子

說明

串鍊運算子“:”用於框架，不可在相同的運算式中做為其他運算子用。因此這種運算子不需要具有優先順序層級。

範例：IF 敘述

```
If （otto==10） and （anna==20） gotof end
```

1.8 可行類型轉換

功能

指派時的類型轉換

常數值，變數，或指派給變數的運算式，必須要有相容的變數類型。在這種情形中，當值被指派時，會自動轉換類型。

可行類型轉換

至	REAL	INT	BOOL	CHAR	STRING	AXIS	FRAME
從							
REAL	是	是*	是 1)	是*	–	–	–
INT	是	是	是 1)	是 2)	–	–	–
BOOL	是	是	是	是	是	–	–
CHAR	是	是	是 1)	是	是	–	–
STRING	–	–	是 4)	是 3)	是	–	–
AXIS	–	–	–	–	–	是	–
FRAME	–	–	–	–	–	–	是

說明

* 從 REAL 轉換至 INT 類型時，>=0.5 的小數值會被進位，若小於則會被捨去（參見 ROUND 函數）。

1) 值 <> 0 為 TRUE；值 == 0 為 FALSE

2) 若值在許可範圍

3) 若只有一個字元

4) 字串長度 0 = >FALSE，其餘為 TRUE

說明

若轉換產生的值，大於目標範圍，則會輸出錯誤訊息。

若在運算式中發生混合類型的情形，則會自動進行類型轉換。同步動作時也可進行類型轉換，請參考“動作同步動作，固有類型轉換”一章。

1.9 字串運算

字串運算

除了典型的運算“指派”與“比較”以外，下列為可用的字串運算：

- 類型轉換為 STRING（AXSTRING）
- 從 STRING 類型轉換（NUMBER、ISNUMBER、AXNAME）
- 串鍊字串（<<）
- 轉換為小寫/大寫字母（TOLOWER，TOUPPER）
- 定義字串長度（STRLEN）
- 在字串中搜尋字元 / 字串（INDEX、RINDEX、MINDEX、MATCH）
- 選擇部分字串（SUBSTR）
- 選擇一個獨立字元（STRINGVAR、STRINGFELD）

0 字元的特殊意涵

在內部，0 字元會解讀為字串辨識碼的終點。若某字元被替換成 0 字元，則該字串會被去掉尾數。
範例：

程式碼	註解
`DEF STRING[20] STRG="axis . stationary"`	
`STRG[6]="X"`	
`MSG（STRG）`	; 產生訊息“座標軸 X 固定”。
`STRG[6]=0`	
`MSG（STRG）`	; 產生訊息“座標軸”。

1.9.1 類型轉換為 STRING（AXSTRING）

功能

使用函數“type conversion to STRING”，不同類型的變數便可用來作為訊息（MSG）元件。
使用<<運算子時，可供INT、REAL、CHAR及BOOL等資料類型內部通用（請參考 “串鍊字串（<<） (頁 70)” ）
一個 INT 值將會轉換成為一般可讀取的格式。REAL 值會轉換成最多具有 10 個小數點位數。
使用 AXSTRING 指令，可將類型 AXIS 變數轉換成為 STRING。

句法

```
<STRING_ERG> = << <any_type>
<STRING_ERG> = AXSTRING（<軸識別碼>）
```

意義

<STRING_ERG>:	供類型轉換結果所用的變數 類型： STRING
<any_type>:	變數類型 INT，REAL，CHAR，STRING 與 BOOL
AXSTRING:	AXSTRING 指令提供了特定的座標軸識別碼作為字串。
<軸識別碼>:	作為座標軸識別碼的變數 類型： AXIS

說明

FRAME 變數無法轉換。

範例

範例 1：

```
MSG（"Position: "<<$AA_IM[X]）
```

範例 2：AXSTRING

程式碼	註解
DEF STRING[32] STRING_ERG	
STRING_ERG=AXSTRING（X）	; STRING_ERG == "X"

1.9.2 從 STRING 類型轉換（NUMBER、ISNUMBER、AXNAME）

功能

使用 NUMBER 指令，可將 STRING 類型轉換成 REAL。使用 ISNUMBER 指令，可檢查轉換的能力。

使用 AXNAME 指令，可將字串轉換成座標軸資料類型。

句法

```
<REAL_ERG>=NUMBER（"<字串>"）
<BOOL_ERG>=ISNUMBER（"<字串>"）
<AXIS_ERG>=AXNAME（"<字串>"）
```

意義

號碼	NUMBER 指令會傳回由<string>所呈現的數字，作為 REAL 值。
<字串>:	待轉換的 STRING 變數
<REAL_ERG>:	供具有 NUMBER 的類型轉換之結果的變數 類型： REAL
ISNUMBER:	ISNUMBER 指令可用來檢查是否可將<字串>轉換成有效數字。
<BOOL_ERG>:	供具有 ISNUMBER 之詢問結果的變數 類型： BOOL 值： TRUE（真） ISNUMBER 提供了值 TRUE，若<string>依照語言規則，呈現了有效的 REAL 數字。 FALSE（偽） 若 ISNUMBER 提供了值 FALSE，當呼叫 NUMBER 時，是以相同的<string>進行，則會產生警報。
AXNAME:	AXNAME 指令會將特定的<string>轉換成座標軸識別碼。 **注意：** 若無法將<string>指派給一個已配置的座標軸識別碼，則會產生警報。
<AXIS_ERG>:	供具有 AXNAME 的類型轉換之結果的變數 類型： AXIS

範例

程式碼	註解
DEF BOOL BOOL_ERG	
DEF REAL REAL_ERG	
DEF AXIS AXIS_ERG	
BOOL_ERG=ISNUMBER（"1234.9876Ex-7"）	; BOOL_ERG == TRUE
BOOL_ERG=ISNUMBER（"1234XYZ"）	; BOOL_ERG == FALSE
REAL_ERG=NUMBER（"1234.9876Ex-7"）	; REAL_ERG == 1234.9876Ex-7
AXIS_ERG=AXNAME（"X"）	; AXIS_ERG == X

1.9.3 串鍊字串（<<）

功能

“串鍊字串”函數允許從獨立元件中配置字串。

使用運算元“<<”來表示串鍊。這個運算子，以 STRING 為一目標類型，作為所有基礎類型 CHAR、BOOL、INT、REAL 以及 STRING，結合時的基本類型。所有可能需要的轉換，會依照現有的規則來實行。

句法

```
<any_type> << <any_type>
```

意義

<any_type>：	變數類型 CHAR、BOOL、INT、REAL 或 STRING
<< ：	運算子用來串連變數（<any_type>），以配置一個字元字串（STRING 類型）。 這個運算子也可單獨使用，作為所謂的“unary”變數。並可用來做精確的類型轉換為 STRING（不是 FRAME 和 AXIS）： << <any_type>

例如，可以從文字清單配置的訊息或指令，或可以被插入（例如一個單節名稱）的參數：
MSG（STRG_TAB[LOAD_IDX]<<BAUSTEIN_NAME）

小心
這種字串串鍊的中間結果，不可以超過最大字串長度。

說明

FRAME 和 AXIS 類型，無法和運算子“**<<**”一起使用。

範例

範例 1：串鍊字串

程式碼	註解
DEF INT IDX=2	
DEF REAL VALUE=9.654	
DEF STRING[20] STRG="INDEX: 2"	
IF STRG=="Index: "<<IDX GOTOF NO_MSG	
MSG（"Index: "<<IDX<<"/value: "<<VALUE）	; 顯示： “Index: 2/value: 9.654”
NO_MSG:	

範例 2：以<<進行精確的類型轉換

程式碼	註解
DEF REAL VALUE=3.5	
<<VALUE	; 特定類型的 REAL 變數會轉換成一個 STRING 類型。

1.9.4 轉換為大小寫字母（TOLOWER、TOUPPER）

功能

“Conversion to lower/upper case letters”（轉換至大小寫字母）函數允許所有字串的字母，轉換成標準代表的意義。

句法

```
<STRING_ERG>=TOUPPER（"<字串>"）
<STRING_ERG>=TOLOWER（"<字串>"）
```

意義

TOUPPER:	使用 TOUPPER 指令，所有字元字串中的字母，都會轉換成**大寫**字母。
TOLOWER:	使用 TOLOWER 指令，所有字元字串中的字母，都會轉換成**小寫**字母。
<字串>:	待轉換的字元字串 類型： STRING
<STRING_ERG>:	供轉換結果所用的變數 類型： STRING

範例

因為使用者輸入可以從從操作員介面上開始，所以他們會被給定標準（大寫或小寫）：

程式碼

```
DEF STRING [29] STRG
...
IF "LEARN.CNC"==TOUPPER（STRG） GOTOF LOAD_LEARN
```

1.9.5 決定字串長度（STRLEN）

功能

STRLEN 指令可以用來決定字元字串的長度。

句法

```
<INT_ERG>=STRLEN（"<字串>"）
```

意義

STRLEN:	STRLEN 指令可決定特定字元字串的長度。 字元數字不是 0 的字元，會從傳回的字串開頭，開始計數。
<字串>:	長度待決定的字元字串 類型： STRING
<INT_ERG>:	供維度結果所用的變數 類型： INT

範例

和單字元存取一起，此函數允許字元字串的結尾待決定：

程式碼

```
IF （STRLEN（BAUSTEIN_NAME）>10） GOTOF ERROR
```

1.9.6 在字串中搜尋字元 / 字串（INDEX、RINDEX、MINDEX、MATCH）

功能

此功能會在字串中搜尋單字元或字串。此函數結果會指定已搜尋過的字串中，已定位字元 / 字串的所在處。

句法

```
INT_ERG=INDEX（STRING，CHAR）； 結果類型：INT
INT_ERG=RINDEX（STRING，CHAR）； 結果類型：INT
INT_ERG=MINDEX（STRING，STRING）； Result type：INT
INT_ERG=MATCH（STRING，STRING）； 結果類型：INT
```

語法

搜尋函數：提供了字串（第一參數），已成功搜尋到的位置。若無法找到字元 / 字串，則會傳回值-1。第一字元具有位置 0。

意義

INDEX:	在第一參數中，搜尋已指定為第二參數（從頭開始找）的字元。
RINDEX:	在第一參數中，搜尋已指定為第二參數（從結尾開始找）的字元。
MINDEX:	對應至 INDEX 函數，除了已傳輸字元清單（作為字串），且當中找到的字元索引會被傳回的情形以外。
MATCH:	在字串中搜尋字串。

這允許字串能根據特定條件來打斷，例如，在空白處或路徑分隔符號（“/”）之處。

範例

打斷一個輸入，使其進入路徑和單節名稱

程式碼	註解
`DEF INT PFADIDX, PROGIDX`	
`DEF STRING[26] INPUT`	
`DEF INT LISTIDX`	
`INPUT = "/_N_MPF_DIR/_N_EXECUTE_MPF"`	
`LISTIDX = MINDEX （INPUT, "M, N, O, P"） + 1`	; 傳回 LISTIDX 的值為 3；因為從清單選擇開始處起算，“N”是參數 INPUT 中的第一個字元。
`PFADIDX = INDEX （INPUT, "/"） +1`	; 因此下列均適用：PFADIDX = 1
`PROGIDX = RINDEX （INPUT, "/"） +1`	; 因此下列均適用：PROGIDX = 12
	下一章節會介紹的 SUBSTR 函數，可以用來打斷在“路徑”和“模組”元件中的 INPUT 變數：
`VARIABLE = SUBSTR （INPUT, PFADIDX, PROGIDX-PFADIDX-1）`	; 隨後傳回 “_N_MPF_DIR”
`VARIABLE = SUBSTR （INPUT, PROGIDX）`	; 隨後傳回 “_N_EXECUTE_MPF”

1.9.7 選擇子字串（SUBSTR）

功能

此功能會從字串中萃取子字串。基於這個理由，會指定第一字元的索引，以及所要的字串長度（如果適用）。若沒有指定長度資訊，那麼字串資料會參考剩餘的字串。

句法

```
STRING_ERG = SUBSTR （STRING，INT）； 結果類型：INT
STRING_ERG = SUBSTR（STRING，INT，INT）； 結果類型：INT
```

語法

在第一個情況中，由第二參數指定的位置，當中的子字串，會傳回到字串結尾。
在第二個情況中，會將結果字串限制在最大長度，由第三參數來指定。
若啟始位置在字串結尾之後，則會傳回空字串（“”）。
若啟始位置或長度為負，則會輸出警報。

範例

程式碼	註解
`DEF STRING[29] ERG`	
`ERG = SUBSTR （"ACK：10 to 99"，10，2）`	`；因此下列均適用：ERG == "10"`

1.9.8 選擇單字元（STRINGVAR，STRINGFELD）

功能

此功能會從字串選擇一個單字元。此適用於讀取和寫入存取運算。

句法

```
CHAR_ERG = STRINGVAR [IDX]； Result type：CHAR
CHAR_ERG = STRINGFELD [IDX_FELD，IDX_CHAR]； 結果類型：CHAR
```

語法

會在字串中讀取 / 寫入在特定位置的字元。若位置參數為負或大於字串，則會輸出警報。

範例訊息：

將座標軸識別碼插入已準備好的字串。

程式碼	註解
`DEF STRING [50] MESSAGE = "Axis n has reached position"`	
`MESSAGE [6] = "X"`	
`MSG （MESSAGE）`	`；傳回“座標軸 X 已到達位置”訊息`

參數

單字元存取僅能用於使用者自訂變數（LUD、GUD 與 PUD 資料）。
這種類型的存取，同樣可能僅供副程式呼叫中的“傳值呼叫”類型的參數使用。

範例

範例 1：單字元對系統資料，機械參數進行存取，等等

程式碼	註解
DEF STRING [50] STRG	
DEF CHAR ACK	
...	
STRG = $P_MMCA	
ACK = STRG [0]	; 求出確認元件

範例 2：以傳址呼叫參數進行的單字元存取：

程式碼	註解
DEF STRING [50] STRG	
DEF CHAR CHR1	
EXTERN UP_CALL （VAR CHAR1）	; 傳址呼叫參數!
...	
CHR1 = STRG [5]	
UP_CALL （CHR1）	; 傳址呼叫
STRG [5] = CHR1	

1.10 程式跳躍與分支

1.10.1 傳回跳躍至程式開始處（GOTOS）

功能

GOTOS 指令能用來跳回主程式或副程式的開始處，以便能重複執行該程式。
機械參數可以用來設定每一次的傳回，要跳到程式開始處：

- 程式執行時間設定為“0”。
- 工件計數因為值“1”增加。

句法

```
GOTOS
```

意義

GOTOS： 目的地為程式開始處的跳躍指令。
執行是透過 NC/PLC 介面訊號來控制：
DB21，至 DBX384.0（控制程式分支）

值：	意義：
0	沒有將跳躍傳回程式開始處。會在 GOTOS 之後的下一個工件程式單節重新開始執行程式。
1	將跳躍傳回程式開始處。已重複執行工件程式。

補充條件

- GOTOS 會內部初始一個 STOPRE （預先處理的停止）。
- 對於具有資料定義（LUD 變數）的副程式，若該資料定義具有 GOTOS，則會在定義區段之後，跳躍至第一個程式單節，換言之，資料定義不會被重複執行。這就是所定義的變數，在到達 GOTOS 單節時，仍可保持其值，並且不會被重置為定義區段中所程式設計標準值的原因。
- GOTOS 指令不可用於同步動作和技術循環。

範例

程式碼	註解
N10 ...	; 程式開始
...	
N90 GOTOS	; 跳躍至程式開始處
...	

1.10.2 程式跳躍至跳躍標記（GOTOB、GOTOF、GOTO、GOTOC）

功能

可在程式中設定跳躍標記（標籤），透過指令 GOTOF，GOTOB，GOTO 或 GOTOC，便能在相同程式中，從其他位置跳躍至此。程式會從馬上接在目的地標記之後的指令，重新開始執行。這表示程式中可以實現分支程式。

除了跳躍標記以外，主單節和副單節數字也可作為跳躍指定。

若在跳躍指令之前，已形成一個跳躍條件（IF ...），則只會在滿足跳躍條件時，才會執行程式跳躍。

句法

```
GOTOB <跳躍目的地>
IF <跳躍條件> = TRUE GOTOB <跳躍目的地>
GOTOF <跳躍目的地>
IF <跳躍條件> = TRUE GOTOF <跳躍目的地>
GOTO <跳躍目的地>
IF <跳躍條件> = TRUE GOTO <跳躍目的地>
GOTOC <跳躍目的地>
IF <跳躍條件> = TRUE GOTOC <跳躍目的地>
```

意義

GOTOB：具有朝向程式開始處的跳躍目的地，其跳躍指令。

GOTOF：具有朝向程式結尾處的跳躍目的地，其跳躍指令。

GOTO：具有跳躍目的地搜尋的跳躍指令。搜尋會先朝向程式結尾處進行，然後再朝向程式開始處進行。

GOTOC：有些影響，例如具有差距的 GOTO，不會發出警報 14080“找不到跳躍指定”。

這表示在搜尋跳躍目的地失敗的情況中，不會中斷程式執行，且會繼續依照 GOTOC 指令執行程式行。

<跳躍目的地>：跳躍目的地參數

可能的資料包含：

- <跳躍標記>：跳躍目的地是以使用者自訂之名稱於程式中設定的跳躍標記（標籤），該自訂名稱為：<jump marker>：
- <單節號碼>：跳躍目的地為主單節或副單節號碼（例如：200，N300）
- 字串類型變數：變數跳躍目的地。此變數表示跳躍標記或單節號碼。

IF：形成跳躍條件的關鍵字。

跳躍條件允許使用所有的比較和邏輯運算（結果：TRUE 或 FALSE）。若運算結果為 TRUE，則會執行程式跳躍。

說明

跳躍標記（標籤）

跳躍標記（標籤）永遠位於單節開始處。若同時有程式編號存在，則跳躍標記會立即位於在單節編號之後。

當為跳躍標記命名時，需套用下列規則：

- 字元數：
  - 至少 2
  - 最多 32
- 認可字元包含：
  - 字母
  - 號碼
  - 底線符號
- 前兩個字元必須為字母或底線。
- 跳躍標記的名稱在冒號（“：”）後。

補充條件

- 跳躍目的地僅能是具有跳躍標記或單節號碼的單節，或是**位於程式中**的單節。
- 不具有跳躍條件的跳躍指令，必須在獨立單節進行程式設計。此限制不適用於具有跳躍條件的跳躍指令。在這種情況中，可在單節中形成數個跳躍指令。
- 具有跳躍指令，但不具有跳躍條件的程式，程式結尾 M2/M30，不一定要放在程式結尾。

範例

範例 1：跳躍至跳躍標記

程式碼	註解
N10…	
N20 GOTOF Label_1	; 朝向程式結尾處，跳躍至跳躍標記“Label_1”。
N30 …	
N40 Label_0: R1=R2+R3	; 跳躍標記“Label_0”設定
N50 …	
N60 Label_1:	; 跳躍標記“Label_1”設定
N70 …	
N80 GOTOB Label_0	; 朝程式開始處，跳躍至跳躍標記“Label_0”。
N90 …	

範例 2：間接跳躍至單節號碼

程式碼	註解
N5 R10=100	
N10 GOTOF "N"<<R10	; 跳躍至單節號碼位於 R10 的單節。
...	
N90 ...	
N100 ...	; 跳躍目的地
N110 ...	
...	

範例 3：跳躍至變數跳躍目的地

程式碼	註解
DEF STRING[20] DESTINATION	
DESTINATION = "Marker2"	
GOTOF DESTINATION	; 朝向程式結尾處，跳躍至變數跳躍目的地 DESTINATION。
標記 1: T="Drill1"	
...	
標記 2: T="Drill2"	; 跳躍目的地
...	

範例 4：以跳躍條件跳躍

程式碼	註解
N40 R1=30 R2=60 R3=10 R4=11 R5=50 R6=20	; 指派初始值。
N41 LA1: G0 X=R2*COS（R1）+R5 Y=R2*SIN（R1）+R6	; 跳躍標記 LA1 設定。
N42 R1=R1+R3 R4=R4-1	
N43 IF R4>0 GOTOB LA1	; 若滿足了跳躍條件，便會朝向程式開始處，跳躍至跳躍標記 LA1。
N44 M30	; 程式結尾

1.10.3 程式分支（CASE ... OF ... DEFAULT ...）

功能

CASE 函數提供了檢查（類型：INT）變數或算術函數的實際值的可能性，並依照此結果，跳躍至程式中不同的位置。

句法

```
CASE（<工式>） OF <constant_1> GOTOF <jump destination_1>
<constant_2> GOTOF <jump destination_2> ... DEFAULT GOTOF <jump
destination_n>
```

意義

CASE:	跳躍敘述
<算式>:	變數或算術函數
OF:	形成有條件的分支程式的關鍵字。
<constant_1>:	變數或算術函數的第一指定常數值 類型： INT
<constant_2>:	變數或算術函數的第二指定常數值 類型： INT
DEFAULT:	若變數或算術函數沒有採用任何特定常數值，則 DEFAULT 指令可用來決定分支程式目的地。 **注意：** 若尚未程式設計 DEFAULT 指令，則在此情形中，跟在 CASE 指令後面的單節，就會成為跳躍目的地。
GOTOF:	具有朝向程式結尾處的跳躍目的地，其跳躍指令。 不使用 GOTOF，可程式設計所有其他的 GOTO 指令（請參考"程式跳躍至跳躍標記"）。
<jump destination_1>:	若變數值或算術函數對應到第一特定常數，則會對此跳躍目的地執行一個分支。 可將跳躍目的地指定為： **<跳躍標記>**：跳躍目的地是以使用者自訂之名稱於程式中設定之跳躍標記（標籤），該自訂名稱為：<jump marker>： **<單節號碼>**：跳躍目的地為主單節或副單節號碼（例如：200，N300） 字串類型變數：變數跳躍目的地。此變數表示跳躍標記或單節號碼。
<jump destination_12>:	若變數值或算術函數對應到第二特定常數，則會對此跳躍目的地執行一個分支。
<jump destination_n>:	若變數值或算術函數不採用特定常數值，則會對此跳躍目的地執行一個分支。

範例

程式碼

```
...
N20 DEF INT VAR1 VAR2 VAR3
N30 CASE（VAR1+VAR2-VAR3） OF 7 GOTOF Label_1 9 GOTOF Label_2 DEFAULT GOTOF Label_3
N40 Label_1: G0 X1 Y1
N50 Label_2: G0 X2 Y2
N60 Label_3: G0 X3 Y3
...
```

CASE 指令，從 N30 而來，定義了下列分支程式的可能性：

1. 若算術函數的值 VAR1+VAR2-VAR3 = 7，便跳躍至具有跳躍標記定義“Label_1”的單節（→ N40）。
2. 若算術函數的值 VAR1+VAR2-VAR3 = 9，便跳躍至具有跳躍標記定義“Label_2”的單節（→ N50）。
3. 若算術函數的值 VAR1+VAR2-VAR3 不等於 7 也不等於 9，便跳躍至具有跳躍標記定義“Label_3”的單節（→N60）。

1.11 重複程式區段（REPEAT，REPEATB，ENDLABEL，P）

功能

程式區段重複允許您在所有程式中，以任何順序重複現有的程式區段。
要重複的程式行或程式區段，由跳躍標記來識別（標籤）。

說明

跳躍標記（標籤）

跳躍標記永遠位於單節開始處。若同時有程式編號存在，則跳躍標記會立即位於在單節編號之後。

當為跳躍標記命名時，需套用下列規則：

- 字元數：
  - 最小 2
  - 最大 32
- 下列為認可的字元：
  - 字母
  - 號碼
  - 底線
- 前兩個字元必須為字母或底線。
- 跳躍標記的名稱在冒號（“：”）後。

句法

1. 重複獨立的程式行：

```
<跳躍標記>: ...
...
REPEATB <跳躍標記> P=<n>
...
```

2. 重複介於跳躍標記和 REPEAT 敘述之間的程式區段：

```
<跳躍標記>: ...
...
REPEAT<跳躍標記> P=<n>
...
```

3. 重複介於跳躍標記之間的區段：

```
<開始跳躍標記>: ...
...
<結尾跳躍標記>: ...
...
REPEAT <開始跳躍標記> <結尾跳躍標記> P=<n>
...
```

說明

不可以在圓括號中，將具有兩個跳躍標記的 REPEAT 敘述，套巢狀使用在一起。若<開始跳躍標記>在 REPEAT 敘述之前出現，且<結尾跳躍標記>，未在 REPEAT 敘述之前到達，則會重複介於<開始跳躍標記>與 REPEAT 敘述之間區段。

4. 介於跳躍標記和 ENDLABEL 之間的重複區段：

```
<跳躍標記>: ...
...
ENDLABEL: ...
...
REPEAT <跳躍標記> P=<n>
...
```

說明

不可在圓括號中，將 REPEAT 敘述套巢狀使用，且該 REPEAT 敘述具有<跳躍標記>與 ENDLABEL。若<跳躍標記>在 REPEAT 敘述之前出現，且<ENDLABEL>，未在 REPEAT 敘述之前到達，則會重複介於<跳躍標記>與 REPEAT 敘述之間區段。

意義

REPEATB:	用來重複程式行的指令
REPEAT:	用來重複程式區段的指令
<跳躍標記>:	<跳躍標記> 識別： • 要重複的程式行（在 REPEATB 的情況下） 或 • 要重複的程式區段開始處（在 REPEAT 的情況下） 程式行係由<跳躍標記>識別，可在 REPEAT/REPEATB 敘述之前或之後出現。此搜尋會朝程式開始處方向進行。若此方向沒有找到跳躍標記，則會朝向程式結尾繼續進行搜尋。 **例外：** 若介於跳躍標記與 REPEAT 敘述之間的程式區段，需要被重複（請參考 2. 句法），由<跳躍標記> 所識別的程式行，必須出現在 REPEAT 敘述**之前**，因為在此情況中，搜尋**只會**朝向程式開始處執行。 若具有<跳躍標記>的程式行中，包含了進一步的操作，則會在各個重複時再次執行。
ENDLABEL:	關鍵字標記要重複的程式區段結尾。 若具有<ENDLABEL>的程式行中，包含了進一步的操作，則會在各個重複時再次執行。 ENDLABEL 在可程式中使用超過一次。
P:	用來指定重複次數的位址
<n>:	程式區段重複的號碼 類型： INT 重複了要重複的程式區段<n> times.在最後一次重複後程式會在 REPEAT/REPEATB 行之後，繼續執行。 **注意：** 若少了指定給 P=<n>的號碼，程式區段只會重複一次。

範例

範例 1：重複獨立程式行

```
N10 POSITION1: X10 Y20
N20 POSITION2: CYCLE（0，，9，8）          ; 位置循環
N30 ...
N40 REPEATB POSITION1 P=5                 ; 執行 BLOCK N10 五次。
N50 REPEATB POSITION2                     ; 執行單節 N20 一次。
N60 ...
N70 M30
```

範例 2：重複介於跳躍標記和 REPEAT 敘述之間的程式區段

```
N5 R10=15
N10 Begin: R10=R10+1          ; 寬度
N20 Z=10-R10
N30 G1 X=R10 F200
N40 Y=R10
N50 X=-R10
N60 Y=-R10
N70 Z=10+R10
N80 REPEAT BEGIN P=4          ; 執行從 N10 到 N70 的區段四次
N90 Z10
N100 M30
```

範例 3：重複介於兩跳躍標記之間的區段

```
N5 R10=15
N10 Begin: R10=R10+1              ; 寬度
N20 Z=10-R10
N30 G1 X=R10 F200
N40 Y=R10
N50 X=-R10
N60 Y=-R10
N70 END: Z = 10
N80 Z10
N90 CYCLE（10，20，30）
N100 REPEAT BEGIN END P=3         ; 執行從 N10 到 N70 的區段三次
N110 Z10
N120 M30
```

範例 4：介於跳躍標記和 ENDLABEL 之間的重複區段

程式碼	註解
N10 G1 F300 Z-10	
N20 BEGIN1:	
N30 X10	
N40 Y10	
N50 BEGIN2:	
N60 X20	
N70 Y30	
N80 ENDLABEL: Z10	
N90 X0 Y0 Z0	
N100 Z-10	
N110 BEGIN3: X20	
N120 Y30	
N130 REPEAT BEGIN3 P=3	; 執行從 N110 到 N120 的區段三次
N140 REPEAT BEGIN2 P=2	; 執行從 N50 到 N80 的區段兩次
N150 M100	
N160 REPEAT BEGIN1 P=2	; 執行從 N20 到 N80 的區段兩次
N170 Z10	
N180 X0 Y0	
N190 M30	

範例 5：銑削，以不同技術進行機台鑽頭定位

程式碼	註解
N10 CENTER DRILL（）	; 載入中心鑽頭。
N20 POS_1:	; 鑽孔定位 1
N30 X1 Y1	
N40 X2	
N50 Y2	
N60 X3 Y3	
N70 ENDLABEL:	
N80 POS_2:	; 鑽孔定位 2
N90 X10 Y5	
N100 X9 Y-5	
N110 X3 Y3	
N120 ENDLABEL:	
N130 DRILL（）	; 變更鑽頭與鑽孔循環。
N140 THREAD（6）	; 載入螺牙 M6 以及螺紋循環。
N150 REPEAT POS_1	; 從 POS_1 向上到 ENDLABEL，重複程式區段一次。
N160 DRILL（）	; 變更鑽頭與鑽孔循環。
N170 THREAD（8）	; 載入螺牙 M8 以及螺紋循環。
N180 REPEAT POS_2	; 從 POS_2 向上到 ENDLABEL，重複程式區段一次。
N190 M30	

其他資訊

- 可以巢狀使用程式區段重複。每個呼叫使用一個副程式層級。
- 若在進行程式區段重複的過程中，程式設計了 M17 或 RET，則該重複將會取消。程式會接在 REPEAT 行之後的單節，重新開始。
- 在實際的程式顯示中，程式區段重複會以獨立的副程式層級顯示。
- 若在程式區段重複時取消層級，則程式會在程式區段重複呼叫之後，重新進行。
  範例：

程式碼	註解
N5 R10=15	
N10 BEGIN: R10=R10+1	; 寬度
N20 Z=10-R10	
N30 G1 X=R10 F200	
N40 Y=R10	; 中斷層級
N50 X=-R10	
N60 Y=-R10	
N70 END: Z10	
N80 Z10	
N90 CYCLE（10，20，30）	
N100 REPEAT BEGIN END P=3	
N120 Z10	; 重新進行程式執行。
N130 M30	

- 檢查結構與程式區段重複可配合使用。然而，二者之間不能有所重疊。程式區段重複應出現在檢查結構的範圍之內，或檢查結構應當出現在程式區段重複之內。
- 若跳躍與程式區段重複混和，則該單節僅會單純地依序執行。例如，若從程式區段重複處，執行一個跳躍，則會繼續進行，直到找到了程式設計的程式區段結尾為止。
  範例：

```
程式碼
N10 G1 F300 Z-10
N20 BEGIN1:
N30 X=10
N40 Y=10
N50 GOTOF BEGIN2
N60 ENDLABEL:
N70 BEGIN2:
N80 X20
N90 Y30
N100 ENDLABEL: Z10
N110 X0 Y0 Z0
N120 Z-10
N130 REPEAT BEGIN1 P=2
N140 Z10
N150 X0 Y0
N160 M30
```

說明

`REPEAT` 敘述應出現在移動單節之後。

1.12 檢查結構

功能

控制會將 NC 單節，處理為已程式設計的順序之標準。

此順序，可能會因為程式設計替代的程式單節和程式迴圈，而有所不同。使用檢查結構元素（關鍵字） `IF...ELSE`，`LOOP`，`FOR`，`WHILE` 以及 `REPEAT`，來程式設計這些檢查結構。

⚠ 小心

檢查結構僅能在程式的敘述區段中插入。程式表頭中的定義，不可附加條件或重複地執行。

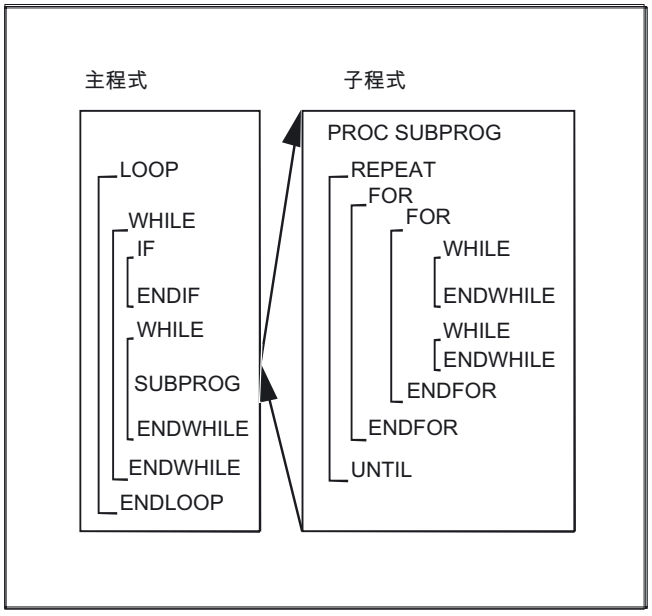
不允許將巨集，加在用來檢查結構的關鍵字，或分支目的地之上。定義巨集時，尚未做過這種檢查。

效果

在程式的範圍內局部套用檢查結構。

檢查結構的

在各副程式層級中，可設定至多 16 個檢查結構的巢狀深度。



檢查結構的

在解譯器模式（啟用為標準）中，藉由使用分支程式，能夠比用檢查結構更有效地縮短程式執行時間。

在預編譯的循環中，分支程式和檢查結構之間沒有差異存在。

補充條件

- 具有檢查結構元素的單節，無法被停用。
- 在具有檢查結構元素的單節中，不允許有跳躍標記（標籤）。
- 檢查結構會被解譯進行。當偵測到迴圈結尾，會允許在程序中找到的檢查結構進行迴圈開始搜尋。基於這個理由，在解譯器模式中，程式的單節結構不會被徹底檢查。
- 通常不建議混合使用檢查結構與分支程式。
- 會進檢查，以確保當處理到循環時，檢查結構為正確巢狀。

1.12.1 具有可供選擇的（IF，ELSE，ENDIF）程式迴圈

功能

當程式迴圈應包含有可供選擇的程式單節時，需使用 IF 和 ELSE 語法：若滿足了 IF 條件，便會執行接在 IF 之後的程式單節。若沒有滿足 IF 條件，便會執行接在 **ELSE** 之後可供選擇的程式單節。

說明

若不需要可供選擇的程式單節，那麼 IF 迴圈也可以不具有 ELSE 的指令來進行程式設計，並執行接在 ELSE 之後的程式單節。

句法

```
IF <條件>
...
ELSE
...
ENDIF
```

意義

IF:	引進 IF 迴圈。
ELSE:	引進可供選擇的程式單節。
ENDIF:	標記 IF 迴圈的結尾，以及造成傳回跳躍到迴圈開始處的結果。
<條件>:	用來決定要執行哪一個程式單節的條件。

範例

換刀副程式

程式碼	註解
PROC L6	; 換刀常式
N500 DEF INT TNR_AKTUELL	; 供有效 T 數字所用的變數
N510 DEF INT TNR_VORWAHL	; 供預選擇的 T 數字所用的變數
	; 決定當前刀具
N520 STOPRE	
N530 IF $P_ISTEST	; 在程式測試模式中...
N540 TNR_AKTUELL = $P_TOOLNO	; ... 從程式文字讀取"當前"刀具。
N550 ELSE	; 否則...
N560 TNR_AKTUELL = $TC_MPP6[9998，1]	; ... 已讀取了主軸的刀具。
N570 ENDIF	
N580 GETSELT（TNR_VORWAHL）	; 在主軸中讀取預選擇刀具之 T 號碼。
N590 IF TNR_AKTUELL <> TNR_VORWAHL	; 若預選擇刀具還不是當前刀具，那麼...
N600 G0 G40 G60 G90 SUPA X450 Y300 Z300 D0	; ... 逼近刀具更換定位...
N610 M206	; ... 以及執行一個換刀。
N620 ENDIF	
N630 M17	

1.12.2 連續程式迴圈（LOOP，ENDLOOP）

功能

無窮迴圈係用於無窮程式。在迴圈結尾，永遠會有一個回到開始處的分支。

句法

```
LOOP
...
ENDLOOP
```

意義

LOOP: 初始化無窮迴圈
ENDLOOP: 標記迴圈的結尾，以及造成傳回跳躍到迴圈開始處的結果。

範例

程式碼
...
LOOP
MSG （"no tool cutting edge active"）
M0
STOPRE
ENDLOOP
...

1.12.3 計數迴圈（FOR ... TO ...，ENDFOR）

功能

當必須以執行的固定數字重複運算時，需使用計數迴圈。

句法

```
FOR <變數> = <初始值> TO <結束值>
...
ENDFOR
```

意義

FOR:	初始化計數迴圈
ENDFOR:	只要尚未達到計數結束值，就標記迴圈的結尾，並造成傳回跳躍到迴圈開始處的結果。
<變數>:	計數變數，係從初始值漸增到結束值，且每次執行時以值“1”增加。 類型 INT 或 REAL **注意：** 例如，當 R 參數被程式設計供計數迴圈所用，則需使用 REAL 類型。若計數變數為 REAL 類型，必須將其值去尾數，成為整數。
<初始值>:	計數初始值 條件：起始值必須小於結束值。
<完整比例值>:	計數結束值

範例

範例 1：INTEGER 變數或作為計數變數的 R 參數

作為計數變數的 INTEGER 變數：

程式碼	註解
DEF INT iVARIABLE1	
R10=R12-R20*R1 R11=6	
FOR iVARIABLE1 = R10 TO R11	; 計數變數= INTEGER 變數
R20=R21*R22+R33	
ENDFOR	
M30	

作為計數變數的 R 參數：

程式碼	註解
R11=6	
FOR R10=R12-R20*R1 TO R11	; 計數變數 = R 參數（實數變數）
R20=R21*R22+R33	
ENDFOR	
M30	

範例 2：生產固定品質的零件

程式碼	註解
DEF INT WKPCCOUNT	; 以“WKPCCOUNT”名稱，定義類型 INT 變數。
FOR WKPCCOUNT = 0 TO 100	; 初始化計數迴圈“WKPCCOUNT”變數，從初始值“0”，朝結束值“100”增加。
G01 …	
ENDFOR	; 計數迴圈結尾
M30	

1.12.4 在迴圈起點處，附有條件的程式迴圈（WHILE，ENDWHILE）

功能

對於 WHILE 迴圈，條件就在迴圈開始處。只要滿足條件，就會執行 WHILE 迴圈。

句法

```
WHILE <條件>
...
ENDWHILE
```

意義

WHILE:	初始化程式迴圈
ENDWHILE:	標記迴圈的結尾，以及造成傳回跳躍返回至迴圈開始的結果。
<條件>:	必須滿足條件，才會執行 WHILE 迴圈。

範例

程式碼	註解
...	
WHILE $AA_IW[DRILL_AXIS] > -10	; 在下列情況下，呼叫 WHILE 迴圈：供鑽孔座標軸所用的實際工件座標系統 WCS 設定點，必須大於-10。
G1 G91 F250 AX[DRILL_AXIS] = -1	
ENDWHILE	
...	

1.12.5 在迴圈結尾處，附有條件的程式迴圈（REPEAT，UNTIL）

功能

對於 REPEAT 迴圈，條件就在迴圈結尾處。REPEAT 迴圈會執行一次，並不斷重複直到滿足條件為止。

句法

```
REPEAT
...
UNTIL <意義>
```

意義

REPEAT:	初始化程式迴圈
UNTIL:	標記迴圈的結尾，以及造成傳回跳躍返回至迴圈開始的結果。
<條件>:	必須滿足條件，REPEAT 迴圈才會不再執行。

範例

程式碼	註解
...	
REPEAT	; 呼叫 REPEAT 迴圈。
...	
UNTIL ...	; 檢查條件是否滿足。
...	

1.12.6 具有巢狀檢查結構的程式範例

程式碼	註解
LOOP	
IF NOT $P_SEARCH	; 無單節搜尋
G01 G90 X0 Z10 F1000	
WHILE $AA_IM[X] <= 100	
G1 G91 X10 F500	; 鑽孔樣版
Z–F100	
Z5	
ENDWHILE	
Z10	
ELSE	
MSG（"No drilling during block search"）	
ENDIF	
$A_OUT[1] = 1	; 下個鑽孔盤
G4 F2	
ENDLOOP	
M30	

1.13 程式一致性（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM）

功能

通道

各通道可以獨立處理其各自的程式。可透過程式，暫時將軸和主軸指派給通道。

在啟動期間，可將二個或更多的通道設定給控制。

程式一致性

若在工件的加工中，包含了數個通道，則可能需要使程式同步。

有些特殊的敘述（指令）可用來供程式一致性所用。各個敘述係在單節中各自程式設計。

說明

程式一致性也有可用於各自的通道內。

程式一致性敘述

- 具有絕對路徑的規格

INIT （n，"/_HUGO_DIR/_N_name_MPF" ） or

INIT （n，"/_N_MPF_DIR/_N_name_MPF" ）

絕對路徑係根據下列規則程式設計：

- Current directory/_N_name_MPF
  “當前目錄”表示所選擇的工件目錄或標準 directory/_N_MPF_DIR。
- 選擇特定程式，供特定通道執行：
  n：通道號碼，其值係依照控制配置
  完整程式名稱而定

-

範例：

INIT（2，"/_N_WKS_DIR/_DRESS_MPF"）

G01F0.1

START

INIT （2，"/_N_WKS_DIR/_N_UNDER_1_SPF"）

最多到 SW 3：

在 **init** 指令（沒有同步）和 **NC start** 之間，至少要有一個可執行的單節。

隨著副程式呼叫，必須將 “_SPF”新增入路徑。

- 相對路徑規格

範例：

INIT（2，"DRESS"）

INIT（3，"UNDER_1_SPF"）

將相同的值，套用在相對路徑定義，作為程式呼叫。

隨著副程式呼叫，必須將 “_SPF”新增入程式名稱。

參數

所有通道可以存取的變數（NCK 特定全域變數），可用來作為程式之間的資料交換。除此之外，則必須將各個程式寫入各自的通道中。

INIT（n，path name，acknowledgement mode）	用來在通道中執行的指令。選擇一個具有絕對或相對路徑名稱的程式。
START （n，n）	從其他通道中，所選定的程式開始 n，n：計算通道的號碼：依照控制配置而決定的值
WAITM （marker no.，n，n，...）	在相同通道中設定標記“marker no.”。以精確停止，來終止前一個單節。等待在特定通道“n”中（不必指定當前通道），具有相同“marker no.”的標記。在同步後刪除標記。 各通道可同步設定 10 個標記。
WAITMC （marker No.，n，n，	在相同通道中設定標記“marker no.”。唯有在其他通道尚未到達標記時，才會初始精確停止。等待在特定通道“n”中（不必指定當前通道），具有相同“marker no.”的標記。一旦在特定通道中到達了“marker no.”標記，則會不用精確停止終止，而會繼續。
WAITE （n，n，...）	等待特定通道的程式結尾（尚未指定當前通道），範例：在 Start 指令後，程式設計一個延遲時間。 N30 START（2） N31 G4 F0.01 N40 WAITE（2）
SETM （marker No.，marker No.，	在相同通道中，設定“marker no.”標記，而不會影響到目前的處理。在 RESET 和 NC START 之後，SETM（）仍為有效。
CLEARM （marker No.，marker No.，	在相同通道中，刪除“marker no.”標記，而不會影響到目前的處理。所有標記皆可用 CLEARM（）來刪除。CLEARM（0）會刪除標記“0”。在 RESET 和 NC START 之後，CLEARM（）仍為有效。
n	對應到通道號碼或通道名稱。

說明

上述所有指令，必須在各自的單節中程式設計。

標記的號碼係依照所用的 CPU 而定。

通道號碼

最多可將 10 個通道，指定為通道號碼（整數值），供需要一致性的通道所用。

通道名稱

必須使用變數（請參考“變數與算術參數”），將通道名稱轉換為號碼。若使用 $MC_CHAN_NAME（識別碼或關鍵字）來定義通道名稱，則會被程式設計而不會成為通道號碼。所定義的名稱必須符合 NC 名稱轉換（例如，前兩個字元必須是字母或底線）。

⚠小心
保護號碼指派，才不會無意中變更了號碼。 名稱不能和現存於 NC 中，具有特殊意義的用字相同，例如關鍵字，指令，座標軸名稱之類。

SETM（）與 CLEARM（）

SETM（）與 CLEARM（）也能在同步動作中，獨自被程式設計。請參考“設定 / 刪除等待標記一章：SETM CLEARM”

範例

稱為“MACHINE”的通道，包含了通道號碼 1。

稱為“LOADER”的通道，包含了通道號碼 2。

```
DEF INT MACHINE=1, LOADER=2
```

給變數相同的名稱做為通道。

因此敘述 START：

```
START（MACHINE）
```

範例：程式一致性

通道 1：

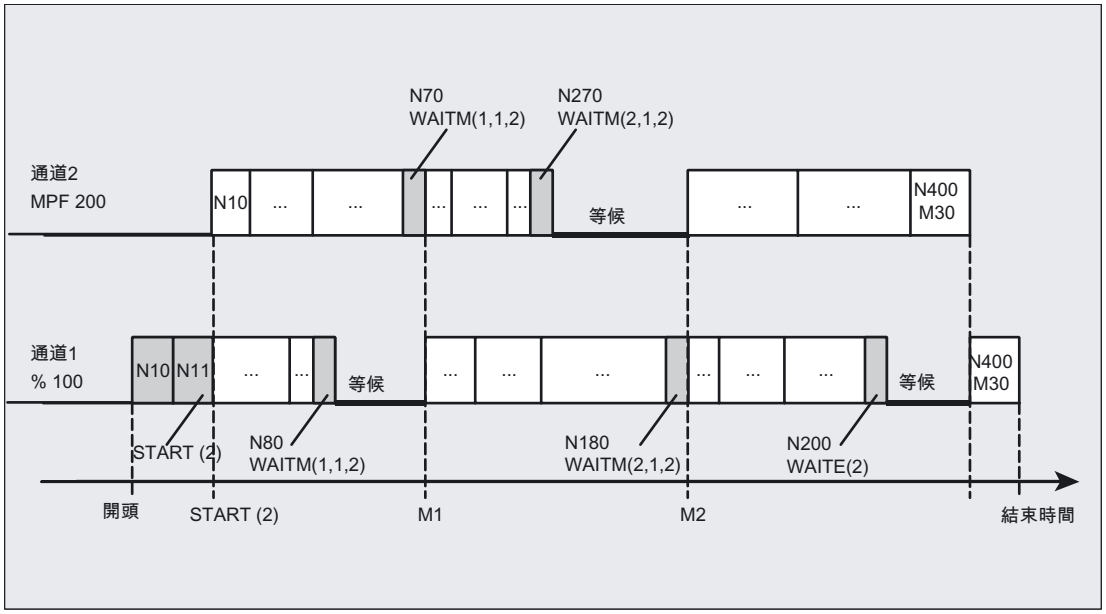
_N_MPF100_MPF

程式碼	註解
N10 INIT（2, "MPF200"）	
N11 START（2）	; 在通道 2 中進行
...	
N80 WAITM（1, 1, 2）	; 等待通道 1 和在通道 1 中另外處理的通道 2 中的 WAIT 標記 1
...	
N180 WAITM（2, 1, 2）	; 等待通道 1 和在通道 2 中另外處理的通道 2 中的 WAIT 標記 1
...	
N200 WAITE（2）	; 為通道 2 等待程式結尾
N201 M30	; 程式通道 1 的結尾，總結尾
...	

通道 2：

_N_MPF200_MPF

程式碼	註解
; $PATH=/_N_MPF_DIR	
	; 在通道 2 中進行
N70 WAITM（1，1，2）	; 等待通道 1 和在通道 2 中另外處理的通道 1 中的 WAIT 標記 2
...	
N270 WAITM（2，1，2）	; 等待通道 1 和在通道 2 中另外處理的通道 2 中的 WAIT 標記 2
...	
N400 M30	; 通道 2 中的程式結尾



範例：工件的程式

程式碼
N10 INIT（2，"/_N_WKS_DIR/_N_SHAFT1_WPD/_N_CUT1_MPF"）

範例：具有相對路徑規格的 INIT 指令

程式/_N_MPF_DIR/_N_MAIN_MPF 在通道 1 中選定

程式碼	註解
N10 INIT（2，"MYPROG"）	; 在通道 2 中選擇程式/_N_MPF_DIR/_N_MYPROG_MPF

範例：通道名稱與具有整數變數的通道號碼

```
$MC_CHAN_NAME[0]= "CHAN_X"；第一通道的名稱
$MC_CHAN_NAME[1]= "CHAN_Y"；第二通道的名稱
```

程式碼	註解
START（1，2）	; 在第一和第二通道開始執行

類似情況，以通道識別碼進行程式設計：

程式碼	註解
START（CHAN_X，CHAN_Y）	; 在第一和第二通道開始執行 ; channel_X 和 channel_Y 識別碼分別內部代表通道號碼 1 和 2，係根據 $MC_CHAN_NAME 機械參數。因此他們也能在第一和第二通道中，執行啟動。

以整數變數進行程式設計：

程式碼	註解
DEF INT chanNo1，chanNo2）	; 定義通道號碼
chanNo1=CHAN_X chanNo2=CHAN_Y	
START（chanNo1，chanNo2）	

1.14 中斷常式（ASUB）

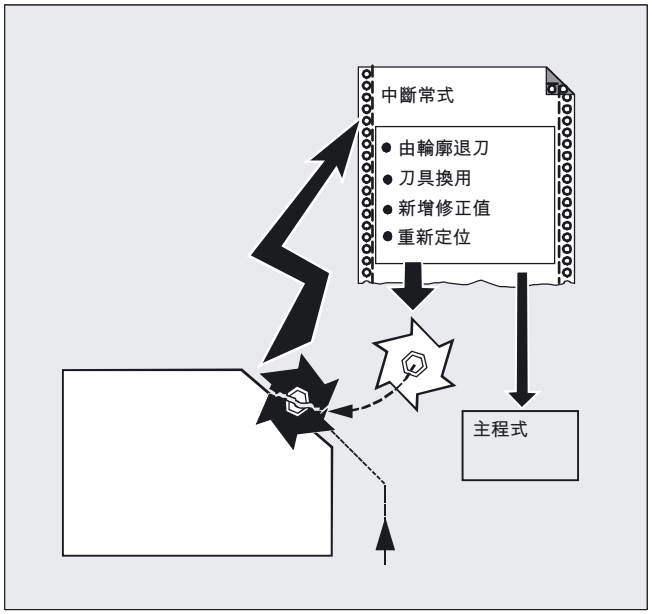
1.14.1 中斷程式的函數

說明

詞語"非同步副程式（ASUB）"與"中斷程式"，在底下的說明中可交替使用，以參考相同的功能。

功能

典型的範例，應能解釋中斷程式的函數：



刀具在加工時停止。這會觸發訊號來停止目前的加工處理，並同步啟動一個副程式–所謂的中斷程式。中斷程式包含了所有在此情況中，有待執行的敘述。

當中斷程式執行完畢後，且機台已準備好繼續操作，則控制會跳躍返回至主程式並繼續在中斷處加工－視REPOS指令（請參考“ 重新定位輪廓 (頁 424) ”）而定。

⚠ 小心

若在副程式中，還沒執行到 REPOS 指令，那麼控制會到跟在中斷的單節後面的單節結束點。

參考

功能手冊，基本功能；BAG，通道，程式操作，重置回應（K1），章節："非同步副程式（ASUBs），中斷程式"

1.14.2 建立一個中斷常式

建立中斷程式作為副程式

在此定義中，中斷程式係定義為副程式。
範例：

程式碼	註解
PROC LIFT_Z	; 程式名稱 “ABHEB_Z”
N10 ...	; NC 單節隨後跟著：
...	
N50 M17	; 在結尾，以及程式結尾返回主程式。

儲存模態 G 碼功能（SAVE）

藉由以 SAVE 定義，來標明中斷程式。

SAVE屬性表示在呼叫中斷程式之前，就先儲存了主動模態G碼功能，並在中斷程式結尾之後，可以重新啟用（請參考“ 具有SAVE機制（SAVE）的副程式 (頁 142) ”）。

這表示在中斷程式結束後，可以在中斷點重新開始處理。
範例：

程式碼
PROC LIFT_Z SAVE
N10 ...
...
N50 M17

指派其他的中斷程式（SETINT）

SETINT指令可以在中斷程式的範圍之內被程式設計（請參考 “指派並啟動中斷程式（SETINT）” (頁 102)），因此可以啟動其他的中斷程式。 他們係透過輸入來觸發。

參考

您將在“副程式，巨集”章節中發現更多關於如何建立副程式的資訊。

1.14.3 指派並啟動中斷程式（SETINT，PRIO，BLSYNC）

功能

控制具有訊號（輸入 1...8），能夠觸發正被中斷的執行程式，並能啟動對應的中斷程式。

使用 SETINT 指令，就能作為從哪一個輸入開始，到哪一個程式是被執行過的標記。

若在工件程式中有數個 SETINT 指令，且因此可同步接收數個訊號，則必須優先配置指派中斷程式，用來定義要執行的中斷程式之順序：PRIO=<值>

當在執行中斷程式時，若要接受新的訊號，會以較高優先順序的常式，來中斷目前的中斷程式。

句法

```
SETINT（<n>） PRIO=<值> <名稱>
SETINT（<n>） PRIO=<值> <名稱> BLSYNC
SETINT（<n>） PRIO=<值> <名稱>LIFTFAST
```

意義

SETINT（<n>）：	指令：指派輸入<n>至一個中斷程式。所指派的中斷程式，會在輸入<n>開關時啟動。 **注意：** 若已將一個已指派的輸入，配置給新的常式，那麼舊的指派就會自動取消。
<n>：	參數：輸入號碼 類型： INT 值域： 1 ... 8
PRIO= ：	指令：定義優先順序
<值>：	優先順序值 類型： INT 值域： 1 ... 128 優先順序 1 表示最高優先權。
<名稱>：	應執行的副程式（中斷程式）名稱。
BLSYNC：	若 SETINT 操作，與 BLSYNC 一起進行程式設計，當接收道中斷訊號時，正在處理中的程式單節會繼續被處理；唯有等到完成此部分時，才會開始處理中斷程式。
LIFTFAST：	若SETINT 操作，與LIFTFAST一起被程式設計，當接收到中斷訊號，則會在開始中斷程式之前，先執行"從輪廓快速回退刀具"（請參考"從輪廓快速回退（SETINT LIFTFAST, ALF） (頁 105)"）。

範例

範例 1：指派中斷程式並定義優先順序

程式碼	註解
...	
N20 SETINT（3） PRIO=1 ABHEB_Z	; 若輸入 3 開關，那麼中斷程式“ABHEB_Z”應當啟動。
N30 SETINT（2） PRIO=2 ABHEB_X	; 若輸入 2 開關，那麼中斷程式“ABHEB_X”應當啟動。
...	

若輸入同步變成可用（同步啟用），中斷程式會以優先值的順序來執行，：第一“ABHEB_Z”，“ABHEB_X”。

範例 2：新指派中斷程式

程式碼	註解
...	
N20 SETINT（3） PRIO=2 ABHEB_Z	; 若輸入 3 開關，那麼中斷程式“ABHEB_Z”應當啟動。
…	
N120 SETINT（3） PRIO=1 LIFT_X	; 輸入 3 被指派至新的中斷程式：若輸入 3 開關，應啟動“ABHEB_X”，取代“ABHEB_Z”。

1.14.4 停用 / 重新啟用指派中斷程式（DISABLE，ENABLE）

功能

SETINT 指令，可以 DISABLE 來將其停用，並可以 ENABLE 將其重新啟用，而不會失去輸入→中斷程式指派。

句法

```
DISABLE（<n>）
ENABLE（<n>）
```

意義

DISABLE（<n>）：	指令：	**停用**輸入的中斷程式指派**<n>**
ENABLE（<n>）：	指令：	**重新啟用**輸入的中斷程式指派**<n>**
<n>：	參數：	輸入號碼
	類型：	INT
	值域：	1 ... 8

範例

程式碼	註解
...	
N20 SETINT（3） PRIO=1 ABHEB_Z	; 若輸入 3 開關，那麼中斷程式“ABHEB_Z”應當啟動。
...	
N90 DISABLE（3）	; 從 N20 來的 SETINT 指令停用。
...	
N130 ENABLE（3）	; 從 N20 來的 SETINT 指令重新啟用。
...	

1.14.5 刪除指派的中斷程式（CLRINT）

功能

使用 SETINT 來定義的輸入→中斷程式，可以 CLRINT 將其刪除。

句法

CLRINT（<n>）

意義

CLRINT（<n>）：	指令：刪除輸入的中斷程式指派<n>	
<n>：	參數：輸入號碼	
	類型：	INT
	值域：	1 ... 8

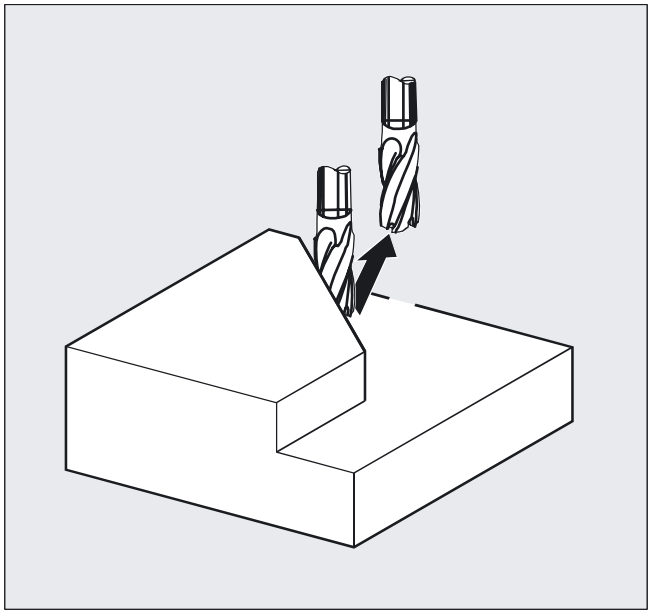
範例

程式碼	註解
...	
N20 SETINT（3） PRIO=2 ABHEB_Z	;
...	
N50 CLRINT（3）	; 刪除介於輸入 “3”和中斷程式“ABHEB_Z”之間的指派。
...	

1.14.6 從輪廓快速回退（SETINT LIFTFAST, ALF）

功能

對於 SETINT 指令，其具有 LIFTFAST，當切換輸入時，使用快速回退，刀具從工件輪廓退出。



後續順序是，依照中斷程式是否包含 SETINT 指令，除了 LIFTFAST 之外，而有所不同：

具有中斷程式：	在快速回退**之後**，會執行中斷程式。
不具有中斷程式：	在快速回退之後，停止加工，並輸出警報。

句法

```
SETINT(<n>) PRIO=1 LIFTFAST
SETINT(<n>) PRIO=1 <名稱> LIFTFAST
```

意義

SETINT（<n>）：	指令： 指派輸入<n>至一個中斷程式。 所指派的中斷程式，會在輸入<n>開關時啟動。	
<n>：	參數： 輸入號碼	
	類型：	INT
	值域：	1 ... 8
PRIO= ：	定義優先順序	
<值>：	優先順序值	
	值域：	1 ... 128
	優先順序 1 表示最高優先權。	

<名稱>:	將執行的副程式（中斷程式）名稱。
LIFTFAST:	指令： 由輪廓快速回退
ALF=… :	指令： 可程式設計移動方向（在動作單節中） 關於以ALF進行程式設計的可能性，請參考“ 從輪廓快速回退的移動方向 (頁 107) ”。

補充條件

具有主動框架的動作

當決定回退方向，會檢查看看是否有鏡像框架啟用中。 在這種情況中，回退方向會參考切線方向，而左右交換。 刀具方向中的元件方向，並非鏡像。 此行為會以使用 MD 設定來啟用：

MD21202 $MC_LIFTFAST_WITH_MIRROR = TRUE

範例

破損的刀具應當自動被備用刀具替換。 隨後會繼續以新的刀具加工。

主程式：

主程式	註解
N10 SETINT（1） PRIO=1 W_WECHS LIFTFAST	; 當切換為輸入 1，刀具會立即以快速回退的方式，從輪廓回退（代碼為 7 號，供刀具半徑補正 G41 所用）。 執行中斷程式“W_WECHS”。
N20 G0 Z100 G17 T1 ALF=7 D1	
N30 G0 X-5 Y-22 Z2 M3 S300	
N40 Z-7	
N50 G41 G1 X16 Y16 F200	
N60 Y35	
N70 X53 Y65	
N90 X71.5 Y16	
N100 X16	
N110 G40 G0 Z100 M30	

副程式：

副程式	註解
PROC W_CHANGE SAVE	; 儲存實際運算狀態的副程式
N10 G0 Z100 M5	; 換刀定位，主軸停止
N20 T11 M6 D1 G41	; 更換刀具
N30 REPOSL RMB M3	; 在輪廓處重新定位，並跳躍返回至主程式（這是在單節中的程式設計）。

1.14.7 從輪廓快速回退的移動方向

回退移動

下列 G code 控制碼定義了回退移動平面：

- LFTXT
  回退移動平面係由路徑切線和刀具方向（預設設定）來決定。
- LFWP
  回退移動的方向，是用 G code 控制碼 G17，G18 或 G19 所選擇的主動工作 / 加工平面。
  回退移動的方向，不相依於路徑切線。 這允許和座標軸平行的快速回退可做程式設計。
- LFPOS
  已宣告的座標軸的回退，使用 POLFMASK/POLFMLIN，對以 POLF 進行程式設計的絕對座標軸定位。
  ALF 對於數個軸和在線性系統中的數個軸，其掀起方向不具有影響。
  參考：
  程式設計手冊，基礎，章： "螺紋切削時的快速回退"

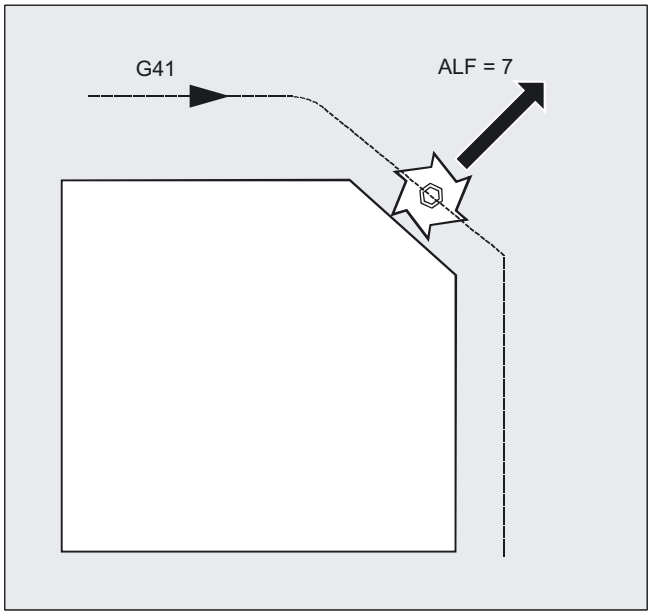
可程式設計移動方向（ALF=...）

程式設計的方向，會在回退移動平面中採用 ALF，以 45 度的離散步驟進行。
可能的移動方向會儲存在控制上的特殊代碼數字中，並可使用三個數字來呼叫。
範例：

程式碼

```
N10 SETINT（2） PRIO=1 LIFT_Z LIFTFAST
ALF=7
```

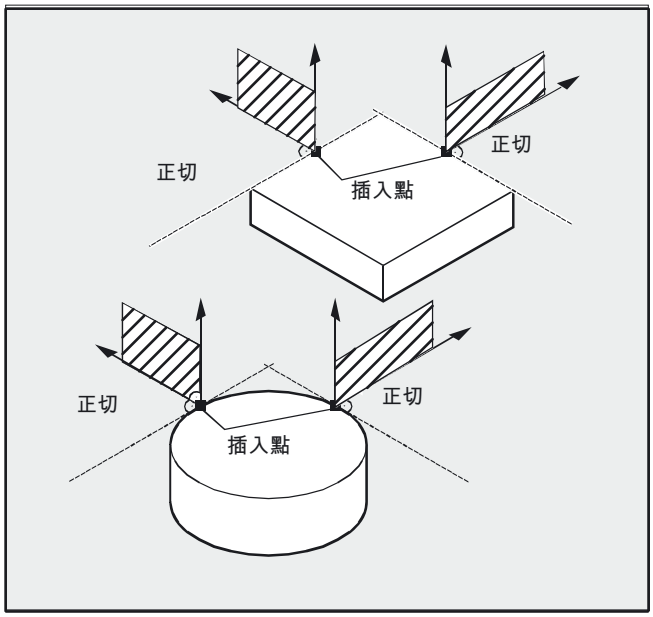
隨著 G41 啟動（對於左側輪廓的加工方向）刀具從輪廓垂直移出。



用來定義移動方向 LFTXT 的參考平面

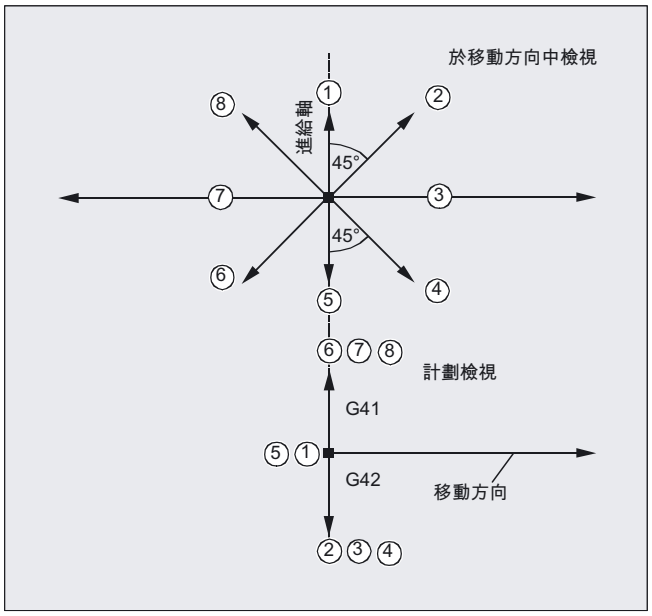
在將刀具應用在程式設計的輪廓處，刀具會被夾在用來作為參考的平面，以便具體指定以對應代碼數字所掀起來的移動。

參考平面係從縱向刀具座標軸（進給方向）而來，並有一個向量對準此軸，以及對準應用刀具的切線來定位。



具有供 LFTXT 所用的移動方向之代碼數字

從參考平面開始，您將在下列圖示中，發現具有移動方向的代碼數字。



刀具的回退方向係定義為 ALF=1。

“快速回退”函數係以 ALF=0 停用。

⚠小心

當刀具半徑補正啟用時，則：
- 對於 G41 代碼 2、3、4
- 對於 G42 代碼 6、7、8

在這種情況下**不可以**使用，刀具會移動到輪廓，且會與工件抵觸。

具有供 LFWP 所用的移動方向之代碼數字

對於 LFWP，工件方向 / 加工平面具有下列指派：

- G17： X/Y 平面

  ALF=1： X 方向之回退

  ALF=3： Y 方向之回退
- G18： Z/X 平面

  ALF=1： Z 方向之回退

  ALF=3： X 方向之回退
- G19： Y/Z 平面

  ALF=1： Y 方向之回退

  ALF=3： Z 方向之回退

1.14.8 供中斷程式所用的順序

不具有 LIFTFAST 的中斷程式

座標軸的動作係沿著路徑減速並停止（零速度）。隨後啟動中斷程式。

靜止位置會儲存為中斷位置，並在中斷程式結尾逼近 REPOS，以 RMI。

具有 LIFTFAST 的中斷程式

座標軸的動作係沿著路徑減速。LIFTFAST 動作會同步執行，作為覆蓋動作。若路徑動作與 LIFTFAST 動作已靜止（零速），則會啟動中斷程式。

輪廓上的位置會儲存起來作為中斷位置，該中斷位置就是 LIFTFAST 動作的開始處，也是路徑離開之處。

具有 LIFTFAST 和 ALF=0 的中斷程式，和不具 LIFTFAST 的中斷程式，以同樣精確的方式運作。

說明

使用機械參數，能將從輪廓快速回退時，幾何軸移動的絕對值，加以設定。

1.15 座標軸替換，主軸替換（RELEASE，GET，GETD）

功能

一或多個軸或主軸，僅能在一個通道中插補。若座標軸必須要從兩個不同的通道之間做出選擇（例如，工作台交換），則必須先在目前的通道中啟用，隨後傳輸至其他通道。座標軸替換在兩個通道中有效。

座標軸替換副檔名

欲替換座標軸 / 主軸，若非以介於前置處理和主程式之間的前置處理停止和同步，就是不能使用前置處理停止。座標軸替換，也可能可以透過：

- 座標軸空間旋轉 AXCTSWE 或 AXCTWED 使用內建的 GET/GETD
- 若此程序將座標軸和其他軸連結，則框架能旋轉。
- 同步動作，請參考動作－同步動作，“座標軸替換 RELEASE，GET”。

機械製造商

請參閱機台製造商說明。若要達成座標軸替換的目的，必須要在配置有機械參數的所有通道中，各自定義一個座標軸，並使用機械參數，來設定座標軸替換特性。

句法

RELEASE （axis name，axis name，...） 或 RELEASE （S1）

GET （axis name，axis name，...） 或 GET （S2）

GETD（axis name，axis name，...） 或 GETD（S3）

隨著 GETD（直接 GET），座標軸會直接從其他通道拿走。那表示對於其他通道中的 GETD，沒有適當的 RELEASE，必須進行程式設計。也表示其他通道通訊，有待建立（例如，等待標記）。

意義

RELEASE（軸名稱，軸名稱，等等）：	釋放座標軸（軸）
GET（軸名稱，軸名稱，等等）：	存取座標軸（軸）
GETD（軸名稱，軸名稱，等等）：	直接存取座標軸（軸）
軸名稱：	系統中的座標軸：AX1、AX2、...或特定機械軸名稱
RELEASE （S1）	釋放主軸 S1、S2、...
GET（S2）	接受主軸 S1、S2、...
GETD（S3）	直接存取主軸 S1、S2、...

不具前置處理停止的 GET 要求

若跟在**不具**前置處理的 GET 要求之後，座標軸再次以 RELEASE（axis）或 WAITP（axis）啟用，則後續的 GET 會產生一個 GET，**具有**前置處理停止。

⚠小心

以 GET 接受的座標軸或主軸，仍然指派至此通道，即使在關鍵或程式 RESET 之後。

當程式重新啟動被替換的軸或主軸，若該座標軸仍須在原通道中使用，則必須在程式中重新指派。

指派給在 POWER ON 時，定義在機械參數中的通道。

範例

範例 1：介於兩通道之間的座標軸交換

在 6 的軸中，下列可用於在通道 1 中加工：1., 2., 3. 和第四座標軸。
第五和第六軸在通道 2 中，會用於工件更改。

在 POWER ON 後，座標軸 2 應當可在兩通道之間交換，並可指派給通道 1。

通道 1 中的程式“MAIN”：

程式碼	註解
INIT （2，"TRANSFER2"）	; 在通道 2 中選擇程式 TRANSFER2。
N… START （2）	; 在通道 2 中啟動程式。
N… GET （AX2）	; 接受座標軸 AX2。
...	
N… RELEASE （AX2）	; 釋放座標軸 AX2。
N… WAITM （1，1，2）	; 等待通道 1 和 2 中的 WAIT 標記，用來在兩個通道中同步。
...	; 在座標軸替換之後的其他程式。
N… M30	

通道 2 中的程式“TRANSFER2”：

程式設計	註解
N… RELEASE （AX2）	
N160 WAITM（1，1，2）	; 等待通道 1 和 2 中的 WAIT 標記，用來在兩個通道中同步。
N150 GET（AX2）	; 接受座標軸 AX2。
...	; 在座標軸替換之後的其他程式。
N… M30	

範例 2：不具同步的座標軸替換

若座標軸不必同步，則 GET 不會產生任何前置處理。

程式設計	註解
N01 G0 X0	
N02 RELEASE（AX5）	
N03 G64 X10	
N04 X20	
N05 GET（AX5）	; 若不需要同步，那麼就無法執行這個單節。
N06 G01 F5000	; 無法被執行的單節。
N07 X20	; 無法被執行的單節，例如 N04 中的 X 定位。
N08 X30	; 在 N05 之後第一個可以執行的單節。
...	

範例 3：啟動不具有前置處理的座標軸交換

先決條件：不具有前置處理的座標軸替換，必須要透過機械參數來配置。

程式設計	註解
N010 M4 S100	
N011 G4 F2	
N020 M5	
N021 SPOS=0	
N022 POS[B]=1	
N023 WAITP（B）	; 座標軸 B 成為中立座標軸。
N030 X1 F10	
N031 X100 F500	
N032 X200	
N040 M3 S500	; 座標軸不會觸發前置處理停止/REORG。
N041 G4 F2	
N050 M5	
N099 M30	

若主軸座標軸 B 係以像是 180 度來移動，在單節 N023 後並立即退回 1 度，作為 **PLC 座標軸**，則此軸將返回其中立狀態，並將不會在單節 N40 中觸發前置處理停止。

需求

座標軸替換的前提

- 該座標軸必須在所有通道中被定義，且該通道必須在機械參數中使用到這些座標軸。
- POWER ON 之後，在**軸**專屬機械參數之中，必須要定義座標軸將指派至哪一個通道。

說明

釋放座標軸：RELEASE

當啟用了座標軸，請注意：

1. 此軸不可以被包含在轉換中。
2. 所有包含在座標軸連結（切線控制）中的軸，必須被啟用。
3. 在此情況中，無法替換並行定位軸。
4. 門型軸的所有跟隨軸，會以第一主要軸傳輸。
5. 至於耦合軸（耦合動作，主動值耦合，電動齒輪），只有群組的引導軸可以啟用。

接受座標軸：GET

實際的座標軸替換會以此指令執行。用來執行指令的通道，必須對座標軸負起全責。

GET 的效果：

同步進行座標軸替換：

若軸同時指派給了其他通道或 PLC，且在 GET 之前，尚未以“WAITP”，G74 或刪除剩餘距離，進行再同步，則該軸必須永遠被同步。

- 接著出現一個前置處理停止（STOPRE）。
- 會中斷執行，直到完成替換。

自動“GET”

若通道中基本上有一個座標軸，但目前沒有定義為“通道軸”，則會自動執行 GET。若座標軸 / 軸已經同步，則不會產生前置處理停止。

驗證座標軸替換行為

可以使用機械參數，將軸的轉換點設定為：

- 當座標軸已經被 WAITP（像之前那樣回應）還原成中立狀態，則隨後會發生介於兩個通道之間的自動座標軸替換。
- 當要求一個座標軸空間旋轉，則可被指定給執行通道的座標軸空間其所有的軸，同時會被內建的 GET 或 GETD 帶入通道。一旦完成了座標軸空間旋轉，則只會認可後續的座標軸替換。
- 當在主程式中插入一個中間單節，則會進行檢查，以決定是否需要重新制訂。唯有當此單節的座標軸狀態，**不符合**目前的座標軸狀態時，才需進行重新制訂。
- 以介於前置處理和主程式之間的前置處理停止和同步，取代 GET 單節，則不需前置處理停止，也能進行軸替換。在此情況中，會以 GET 要求，產生一個中間單節。在主程式中，當執行了此單節，則系統會檢查單節中的軸狀態是否符合目前的座標軸狀態。

若要更多有關座標軸或主軸替換如何運作的資訊，請參考
/FB2/ 功能手冊，延伸函模式群組，通道座標軸替換（K5）。

1.16 將座標軸傳輸至其他通道（AXTOCHAN）

功能

AXTOCHAN 語言指令可用來要求一個軸，將其移動到其他通道。此座標軸，可以從 NC 工件程式，也可以從同步動作，來移動到對應的通道。

句法

```
AXTOCHAN（axis name，channel number[，axis name，channel
number[，...]]）
```

意義

AXTOCHAN：	要求座標軸作為特定通道
軸名稱：	系統中的座標軸：與 X、Y、… 或機械軸名稱的輸入有關。執行通道不必是同一個通道，或甚至可以是目前正在處理插補供座標軸所用的通道也行。
通道號碼：	通道號碼對應至那個有待指派的座標軸

說明

競爭定位座標軸和 PLC 控制座標軸互斥

PLC 座標軸無法將通道替換為競爭定位座標軸。被 PLC 互斥控制的座標軸，無法被指派給 NC 程式。

參考：
功能手冊，延伸功能；位置軸（P2）

範例

NC 程式內的 AXTOCHAN

軸 X 和 Y 已在第二通道中宣告。目前，通道 1 具有插補，且後續程式在此通道中開始。

程式碼	註解
N110 AXTOCHAN（Y，2）	; 將 Y 座標軸移動至第二通道。
N111 M0	
N120 AXTOCHAN（Y，1）	; 重新取回 Y 座標軸（中立）。
N121 M0	
N130 AXTOCHAN（Y，2，X，2）	; 將 Y 座標軸和 X 座標軸移動至第二通道（軸為中立）。
N131 M0	
N140 AXTOCHAN（Y，2）	; 將 Y 座標軸移動到第二通道（NC 程式）。
N141 M0	

其他資訊

NC 程式內的 AXTOCHAN

只有在座標軸被相同通道中的 NC 程式要求時，才會執行 GET。（這表示系統等待狀態會實際改變）。若是其他通道要求座標軸，或是座標軸在相同通道中變成中立軸，則會送出此要求。

從同步動作而來的 AXTOCHAN

當座標軸被相同通道要求時，從同步動作而來的 AXTOCHAN，會對應至從同步動作而來的 GET。在此情形中，座標軸會在第一次被相同通道邀請時，成為中立座標軸。第二次要求時，座標軸會以相同方式被指派至 NC 程式，作為在 NC 程式之中的 GET 要求。若要更多關於從同步動作而來的 GET 要求之資訊，請參考"動作－同步動作"。

1.17 啟動機械參數（NEWCONF）

功能

NEWCONF 指令用於設定所有有效的"NEW_CONFIG"有效層級之機械參數。在 HMI 使用者介面中，也能藉由按下“MD 資料有效”軟鍵，來啟用函數。

執行“NEWCONF”函數時，會有一個內建的前置處理停止，亦即表示中斷了路徑移動。

句法

NEWCONF

意義

NEWCONF: 用來設定所有有效層級的“NEW_CONFIG”的機械參數，皆被設定為有效。

從工件程式，跨通道執行 NEWCONF

若對工件程式中的機械參數進行變更，接著使用 NEWCONF 啟用，那麼 NEWCONF 將只會啟用包含了對工件程式通道會產生影響之變更的機械參數。

說明

為了確保已套用所有變更，必須在所有通道中執行 NEWCONF 敘述，而該通道中的軸或函數，必須是已經計算、且受到機械參數變更的影響。

沒有軸的機械參數對 NEWCONF 有效。

軸的 RESET，必須對受 PLC 控制的軸執行。

範例

銑削：以不同技術進行機台鑽頭定位

程式碼	註解
N10 $MA_CONTOUR_TOL[AX]=1.0	; 更改機械參數。
N20 NEWCONF	; 啟動機械參數。
...	

1.18 寫入檔案（WRITE）

功能

WRITE 指令可用來從工件程式，寫入單節/資料至特定檔案的結尾（日誌檔），或至正在執行的工件程式結尾。單節/資料會被插入在檔案結尾，例如在 M30 之後。

說明

如果在 NC 中沒有存在這種檔案，則會建立一個，並可使用 WRITE 指令將其寫入。

儲存位置為一個靜態的 NC 記憶體。在 SINUMERIK 840D sl 中為 CF 卡。和 SINUMERIK 840D 的不同之處在於，這增加了大約 75 ms 的 WRITE 指令的執行時間。

若在硬碟中，有相同的檔名存在，則會在關閉該檔案後（在 NC 中）覆寫。解決方式：在服務操作區中，使用“屬性”軟鍵，在 NC 之中變更名稱。

條件

目前的設定保護等級，必須等於或大於檔案的 WRITE 權限。在此情形中，會以錯誤訊息（錯誤變數的傳回值=13）來否決存取。

句法

```
DEF INT <錯誤>
WRITE（<錯誤>, "<檔案名稱>", "<單節/資料>"）
```

意義

寫入：　用來在特定檔案結尾，附加一個單節或資料的指令。

<錯誤>：　用來傳回錯誤值的變數。

類型。 INT

值：

0	無錯誤
1	不認可的路徑
2	找不到路徑
3	找不到檔案
4	不正確的檔案類型
10	檔案已滿
11	檔案使用中
12	無可用資源
13	無存取權限
20	其他錯誤

<檔案名稱>: 要被特定單節或特定資料加入的檔案名稱。
類型: STRING
下列為在指定檔案名稱時應注意的事項:

- 特定檔案名稱不可以包含任何空格或控制字元（ASCII 代碼≤ 32 的字元），否則 WRITE 指令會以錯誤代碼 1 "路徑不允許"來取消。
- 可以用路徑資料和檔案識別碼來指定檔名。
  - 路徑資料

    路徑資料必須為絕對，例如以"/"開始。

    若無特定路徑，檔名會以目前的目錄儲存（= 所選程式的目錄）。
  - 檔案識別碼

    若檔名沒有包含域識別碼（“_N_”），便可新增。

    若檔案名稱的倒數第四個字元為底線"_"，則最後三個字元會被轉譯為識別碼。為了要能對所有檔案指令使用相同的檔案名稱，例如透過字串類型變數，則只有_SPF 與_MPF 檔案識別碼可被使用。

    若無識別碼（“_MPF”或“_SPF”），則檔名會自動以_MPF 來完成。
- 檔名長度最多可到 32 位元，路徑長度最多可到 128 位元。

範例:

```
"PROTFILE"
"_N_PROTFILE"
"_N_PROTFILE_MPF"
"/_N_MPF_DIR_/_N_PROTFILE_MPF/"
```

<單節/資料: 要被加入至特定檔案的區塊或資料。
類型: STRING

注意:
隨後加入內部 LF；這表示字元字串增加了一個字元。

一般條件

- **最大檔案大小（→機台製造商）**

  日誌檔的最大可能檔案大小係以機械參數設定:

  MD11420 $MN_LEN_PROTOCOL_FILE

  最大檔案長度可用於所有以 WRITE 指令建立的檔案。若超過，會輸出一個錯誤訊息，且該單節或資料不會被儲存。若自由記憶體足夠，則可建立一個新的檔案。

範例

範例 1：沒有絕對路徑資料的 WRITE 指令

程式碼	註解
N10 DEF INT ERROR	; 錯誤變數之定義。
N20 WRITE（ERROR，"TEST1"，"LOG FROM 7.2.97"）	; 從"LOG FROM 7.2.97"寫入文字至檔案_N_TEST1_MPF。
N30 IF ERROR	; 錯誤評估
N40 MSG （"Error with WRITE command: "<<錯誤）	
N50 M0	
N60 ENDIF	
...	

範例 2：有絕對路徑資料的 WRITE 指令

程式碼
...
WRITE（ERROR，"/_N_WKS_DIR/_N_PROT_WPD/_N_PROT_MPF"，"LOG FROM 7.2.97"）
...

1.19 刪除檔案（DELETE）

功能

DELETE 指令可用來刪除所有檔案，不論這些檔案是否使用 WRITE 指令所建立。使用較高存取權限來建立的檔案，也能用 DELETE 來刪除。

句法

```
DEF INT <錯誤>
DELETE（<錯誤>, "<檔案名稱>"）
```

意義

DELETE: 用來刪除特定檔案的指令。

<錯誤>: 用來傳回錯誤值的變數。

類型。 INT

值:

0	無錯誤
1	不認可的路徑
2	找不到路徑
3	找不到檔案
4	不正確的檔案類型
11	檔案使用中
12	無可用資源
20	其他錯誤

<檔案名稱>： 待刪除之檔案的名稱
類型： STRING

下列為在指定檔案名稱時應注意的事項：

- 特定檔案名稱不可以包含任何空格或控制字元（ASCII 代碼≤ 32 的字元），否則 DELETE 指令會以錯誤代碼 1 "路徑不允許"來取消。
- 可以用路徑資料和檔案識別碼來指定檔名。
  - 路徑資料

    路徑資料必須為絕對，例如以"/"開始。

    若不指定特定路徑，則會在目前的目錄中搜尋檔案（= 所選程式的目錄）。
  - 檔案識別碼

    若檔名沒有包含域識別碼（“_N_”），便可新增。

    若檔案名稱的倒數第四個字元為底線"_"，則最後三個字元會被轉譯為識別碼。為了要能對所有檔案指令使用相同的檔案名稱，例如透過字串類型變數，則只有_SPF 與_MPF 檔案識別碼可被使用。

    若無識別碼（“_MPF”或“_SPF”），則檔名會自動以_MPF 來完成。
- 檔名長度最多可到 32 位元，路徑長度最多可到 128 位元。

範例：

```
"PROTFILE"
"_N_PROTFILE"
"_N_PROTFILE_MPF"
"/_N_MPF_DIR_/_N_PROTFILE_MPF/"
```

範例

程式碼	註解
N10 DEF INT ERROR	; 錯誤變數之定義。
N15 STOPRE	; 停止前置處理
N20 DELETE（ERROR, "/_N_SPF_DIR/_N_TEST1_SPF"）	; 在副程式目錄中刪除檔案 TEST1。
N30 IF ERROR	; 錯誤評估
N40 MSG（"error for DELETE command: "<<錯誤）	
N50 M0	
N60 ENDIF	

1.20 在檔案中讀入行（READ）

功能

READ 指令會在特定檔案中，讀入一或多行，並儲存陣列類型 STRING 的資訊。在此陣列中，每一個讀入行佔用了一個陣列元素。

說明

檔案必須儲存在 NCK'的靜態使用者記憶體中（被動檔案系統）。

條件

目前的設定保護等級，必須等於或大於檔案的 READ 權限。在此情形中，會以錯誤訊息（錯誤變數的傳回值=13）來否決存取。

句法

```
DEF INT<錯誤>
DEF  STRING[<字串長度>] <結果>[<n>，<m>]
READ（<錯誤>、"<檔案名稱>"、<起始指令列>、<指令列數>、<結果>）
```

意義

讀取：用於從特定檔案中讀取指令列並將這些指令列儲存在變數陣列中。

<錯誤>：用於傳回錯誤值的變數（傳址呼叫參數）

類型。 INT

值：

0	無錯誤
1	不認可的路徑
2	找不到路徑
3	找不到檔案
4	不正確的檔案類型
13	存取權限不足
21	指令列不存在（<起始指令列>或<指令列數>參數超過指定檔案中的指令列數）。
22	結果變數（<結果>）的欄位長度過短。
23	指令列範圍過大（選擇了<指令列數>個參數故讀取時會超出檔案的終點）。

<檔案名稱>: 欲讀取之檔案的名稱（傳值呼叫參數）
類型: STRING
指定檔案名稱時應遵守以下幾點:

- 指定的檔案名稱不可包含任何空白或控制字元（ASCII 碼≤ 32 的字元），否則會以錯誤代碼 1"不允許的路徑"取消 READ 指令。
- 可以用路徑資料和檔案識別碼來指定檔名。
  - 路徑資料

    路徑資料需為絕對值，亦即開頭有"/"。

    若不指定特定路徑，則會在目前的目錄中搜尋檔案（= 所選程式的目錄）。
  - 檔案識別碼

    若檔名沒有包含域識別碼（“_N_”），便可新增。

    若檔案名稱的倒數第四個字元為底線"_"，則接下來三格字元會轉譯為檔案識別碼。為能讓所有的檔案指令均可使用相同的檔案名稱，例如透過 STRING 類型變數，僅會使用_SPF 與_MPF 檔案識別碼。

    若無識別碼（“_MPF”或“_SPF”），則檔名會自動以_MPF 來完成。
- 檔名長度最多可到 32 位元，路徑長度最多可到 128 位元。

範例:

```
"PROTFILE"
"_N_PROTFILE"
"_N_PROTFILE_MPF"
"/_N_MPF_DIR_/_N_PROTFILE_MPF/"
```

<起始指令列>: 要讀取之檔案區段的起始指令列（傳值呼叫參數）
類型: INT

值:		
	0	在結束檔案前讀取以<指令列數>參數指定的指令列數。
	1 到 n	待讀取的第一行數字。

<指令列數>: 欲讀取之指令列數（傳值呼叫參數）
類型: INT

<結果>: 結果變數（傳址呼叫參數）
儲存了讀取之文字的變數陣列。
類型: 字串（最大長度：255)
若在<指令列數>參數中指定的列數小於結果變數的陣列大小[<n>, <m>]，則不會修改剩餘的陣列元素。
藉由控制字元“LF”（換行）或“CR LF”（歸位換行），所終止的行，**不會**儲存在變數“結果”之中。
當讀取的指令列較定義的字串長度更長時則會進行裁切。未輸出錯誤訊息。

說明

無法讀入二元檔案。會輸出"資料類型不正確"錯誤訊息（錯誤變數的傳回值= 4）。下列類型的檔案無法讀取：_BIN，_EXE，_OBJ，_LIB，_BOT，_TRC，_ACC，_CYC，_NCK.

範例

程式碼	註解
N10 DEF INT ERROR	; 錯誤變數的定義。
N20 DEF STRING[255] RESULT[5]	; 結果變數的定義。
N30 READ（ERROR，"/_N_CST_DIR/_N_TESTFILE_MPF"，1，5，RESULT）	; 有域、檔案識別碼和路徑規格的檔名
N40 IF ERROR <>0	; 錯誤評估。
N50 MSG（"ERROR"<<ERROR<<"ON READ COMMAND"）	
N60 M0	
N70 ENDIF	
...	

1.21 確認檔案（ISFILE）是否存在

功能

ISFILE 指令可用於確認檔案是否存在 NCK 靜態使用者記憶體（被動檔案系統）中。

句法

<結果>=ISFILE（"<檔案名稱>"）

意義

ISFILE: 用於確認指定檔案是否存在被動檔案系統中的指令。

<檔案名稱>: 待確認是否存在被動檔案系統中之檔案名稱。

類型： STRING

指定檔案名稱時應遵守以下幾點：

- 指定的檔案名稱不可含有空白或控制字元（ASCII 碼≤ 32 的字元）。
- 可以用路徑資料和檔案識別碼來指定檔名。
  - 路徑資料

    路徑資料需為絕對值，亦即開頭有"/"。

    若不指定特定路徑，則會在目前的目錄中搜尋檔案（= 所選程式的目錄）。
  - 檔案識別碼

    若檔名沒有包含域識別碼（“_N_”），便可新增。

    若檔案名稱的倒數第四個字元為底線"_"，則接下來三格字元會轉譯為檔案識別碼。為能讓所有的檔案指令均可使用相同的檔案名稱，例如透過 STRING 類型變數，僅會使用_SPF 與_MPF 檔案識別碼。

    若無識別碼（“_MPF”或“_SPF”），則檔名會自動以_MPF 來完成。
- 檔名長度最多可到 32 位元，路徑長度最多可到 128 位元。

範例：

```
"PROTFILE"
"_N_PROTFILE"
"_N_PROTFILE_MPF"
"/_N_MPF_DIR_/_N_PROTFILE_MPF/"
```

<結果>: 用於指派檢查結果的結果變數。

類型。 BOOL

值：		
	TRUE	檔案存在
	FALSE（偽）	檔案不存在

範例

程式碼	註解
N10 DEF BOOL RESULT	; 結果變數的定義。
N20 RESULT=ISFILE（"TESTFILE"）	
N30 IF（RESULT==FALSE）	
N40 MSG（"FILE DOES NOT EXIST"）	
N50 M0	
N60 ENDIF	
...	

or:

程式碼	註解
N10 DEF BOOL RESULT	; 結果變數的定義。
N20 RESULT=ISFILE（"TESTFILE"）	
N30 IF（NOT ISFILE（"TESTFILE"））	
N40 MSG（"FILE DOES NOT EXIST"）	
N50 M0	
N60 ENDIF	
...	

1.22 讀出檔案資訊（FILEDATE、FILETIME、FILESIZE、FILESTAT、FILEINFO）

函數

The FILEDATE、FILETIME、FILESIZE、FILESTAT、及 FILEINFO 指令可用於讀出最後存取的日期/時間、目前檔案的大小、檔案狀態等特定檔案資訊或以上全部的資訊。

說明

檔案必須儲存在 NCK'的靜態使用者記憶體中（被動檔案系統）。

條件

目前的設定保護等級，必須等於或大於上級目錄的顯示權限。 在此情形中，會以錯誤訊息（錯誤變數的傳回值=13）來否決存取。

句法

```
DEF INT<錯誤>
DEF STRING[<字串長度>] <結果>
FILE....(<錯誤>、"<檔案名稱>"、<結果>)
```

意義

FILEDATE:	FILEDATE 指令會傳回最後對指定檔案進行寫入存取的**日期**。
FILETIME:	FILETIME 指令會傳回最後對指定檔案進行寫入存取的**時間**。
FILESIZE:	FILESIZE 指令會傳回指定檔案**目前的大小**。
FILESTAT:	FILESTAT 指令會傳回與指定檔案的讀取、寫入與存取等權限有關的**狀態**。
FILEINFO:	The FILEINFO 指令會傳回指定檔案所有可使用 FILEDATE、FILETIME、FILESIZE、與 FILESTAT 讀取的**所有檔案資訊**。
<錯誤>:	用於傳回錯誤值的變數（傳址呼叫參數）
	類型。 INT
	值： 0 無錯誤
	1 不認可的路徑
	2 找不到路徑
	3 找不到檔案
	4 不正確的檔案類型
	13 存取權限不足
	22 結果變數（<結果>）的字串長度過短。

<檔案名稱>: 欲讀出檔案資訊的檔案之名稱。
類型: STRING
指定檔案名稱時應遵守以下幾點:

- 指定的檔案名稱不可包含任何空白或控制字元（ASCII 碼≤ 32 的字元），否則會以錯誤代碼 1"不允許的路徑"取消 `FILE...`指令。
- 可以用路徑資料和檔案識別碼來指定檔名。
  - 路徑資料

    路徑資料需為絕對值，亦即開頭有"/"。

    若不指定特定路徑，則會在目前的目錄中搜尋檔案（= 所選程式的目錄）。
  - 檔案識別碼

    若檔名沒有包含域識別碼（“_N_”），便可新增。

    若檔案名稱的倒數第四個字元為底線"_"，則接下來三格字元會轉譯為檔案識別碼。 為能讓所有的檔案指令均可使用相同的檔案名稱，例如透過 STRING 類型變數，僅會使用_SPF 與_MPF 檔案識別碼。

    若無識別碼（“_MPF”或“_SPF”），則檔名會自動以_MPF 來完成。
- 檔名長度最多可到 32 位元，路徑長度最多可到 128 位元。

範例:

```
"PROTFILE"
"_N_PROTFILE"
"_N_PROTFILE_MPF"
"/_N_MPF_DIR_/_N_PROTFILE_MPF/"
```

<結果>: 結果變數（傳址呼叫參數）
儲存了要求之檔案資訊的變數。
類型: STRING 搭配: `FILEDATE`
格式: "日日.月月.年年"
⇒字串長度需為 **8**。
`FILETIME`
格式: "時時：分分.秒秒"
⇒字串長度需為 **8**。
`FILESTAT`
格式: "rwxsd"
（r: **讀**取，w: **寫**入，x: 執**行**，s: **顯**示，d: **刪**除）
⇒字串長度需為 **5**。
`FILEINFO`
格式: “rwxsd nnnnnnnn dd. hh：mm：ss”
⇒字串長度需為 **32**。

INT 搭配: `FILESIZE`
將檔案大小以位元輸出

範例

程式碼	註解
N10 DEF INT ERROR	; 錯誤變數的定義。
N20 STRING[32] RESULT	; 結果變數的定義。
N30 FILEINFO（ERROR, "/_N_MPF_DIR/_N_TESTFILE_MPF", RESULT）	; 有域、檔案識別碼和路徑規格的檔名
N40 IF ERROR <>0	; 錯誤評估
N50 MSG("ERROR"<<ERROR<<"ON FILEINFO COMMAND")	
N60 M0	
N70 ENDIF	
...	

該範例可利用 RESULT 結果變數傳回以下結果，例如：

"77777 12345678 26.05.00 13:51:30"

1.23 使用陣列進行總合檢查碼計算（CHECKSUM）

功能

CHECKSUM 指令可用於以陣列計算總合檢查碼。該總合檢查碼可和稍早所做之總合檢查碼計算結果做比較以確認變更的陣列資料為何。

應用

檢查在材料移除期間，初始輪廓是否有改變。

句法

```
DEF INT<錯誤>
DEF STRING[<字串長度>] <總合檢查碼>
DEF ... <陣列>[<n>，<m>，<o>]
<錯誤>=CHECKSUM（<總合檢查碼>，"<陣列>"[，<起始欄>，<結尾欄位>]）
```

意義

CHECKSUM: 用於以陣列計算總合檢查碼的指令。

<錯誤>: 用於傳回錯誤值之變數。

類型。 INT

值：

0	無錯誤
1	找不到符號
2	沒有陣列
3	索引 1 過大
4	索引 2 過大
5	無效的資料類型
10	總和檢查碼溢位

<總合檢查碼>: 用於寫入總合檢查碼計算結果的結果變數（傳址呼叫參數）

類型： STRING

需要的字串長度： 16

總合檢查碼會以 16 個 16 進位數字的字元字串表示。然而，沒有格式字元可以指定。

範例："A6FC3404E534047C"

<陣列>: 欲用於產生總合檢查碼的陣列名稱（傳值呼叫參數）

類型： STRING

最大字串長度： 32

可允許的陣列為以下類型的 1 至 3 維陣列：

布林、字元、整數、實數、字串

注意：

不允許機械參數陣列。

<起始欄位>: 用於計算總合檢查碼（選用參數）的起始欄位數

<結束欄位>: 用於計算總合檢查碼（選用參數）的結束欄位數

說明

The <起始欄位>與<結尾欄位>參數均為選用。若沒有指定欄位索引，則會在整個陣列上形成總合檢查碼。

總合檢查碼的結果永遠是明確的。若變更了陣列元素，則結果字串也會跟著變更。

範例

程式碼	註解
`N10 DEF INT ERROR`	`;` 錯誤變數的定義。
`N20 DEF STRING[16] MY_CHECKSUM`	`;` 結果變數的定義。
`N30 DEF INT MY_VAR[4, 4]`	`;` 陣列定義。
`N40 MY_VAR=...`	
`N50 ERROR=CHECKSUM(MY_CHECKSUM, "MY_VAR", 0, 2)`	
`...`	

該範例可利用 MY_CHECKSUM 結果變數傳回以下結果，例如：

"A6FC3404E534047C"

1.24 無條件進位（ROUNDUP）

功能

輸入值，類型 REAL（具有小數點），使用 ROUNDUP 函數，會將值無條件進位成下一個較大的整數。

句法

```
ROUNDUP（<值>）
```

意義

ROUNDUP：	作為無條件進位輸入值的指令
<值>：	輸入值，類型 REAL

說明

輸入值，類型 INTEGER（整數），則不會改變。

範例

範例 1：多種輸入值，及其無條件進位的結果

範例	無條件進位結果
ROUNDUP（3.1）	4.0
ROUNDUP（3.6）	4.0
ROUNDUP（-3.1）	-3.0
ROUNDUP（-3.6）	-3.0
ROUNDUP（3.0）	3.0
ROUNDUP（3）	3.0

範例 2：在 NC 程式中的 ROUNDUP

程式碼

```
N10 X=ROUNDUP（3.5） Y=ROUNDUP（R2+2）
N15 R2=ROUNDUP（$AA_IM[Y]）
N20 WHEN X=100 DO Y=ROUNDUP（$AA_IM[X]）
...
```

1.25 副程式技術

1.25.1 一般資訊

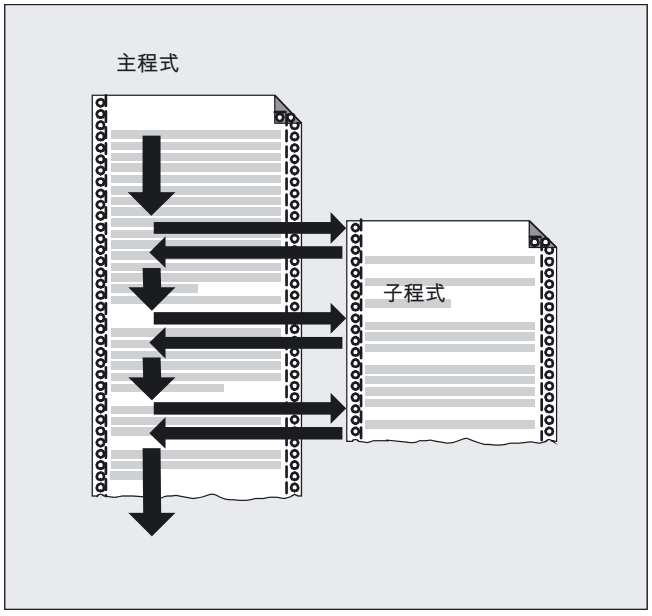
1.25.1.1 副程式

功能

當工件程式嚴謹的區分為主與副程式時，"副程式"具有其原點。主程式為選擇用於在控制時處理並啟動的工件程式。副程式為從主程式中呼叫的工件程式。

在今日的 SINUMERIK NC 程式語言中已無嚴格的區別。基本上，每個工件程式均可選擇作為主程式並執行或作為副程式從其他工件程式中呼叫。

相對的，副程式可用於代表從其他工件程式中所呼叫的工件程式。



應用

與在高階程式語言中一樣，在 NC 程式語言中，副程式可用於取代在獨立、內含之程式中使用超過一次以上的程式區段。

副程式提供下列優點：

- 增加了程式的透明度和可讀取性
- 透過再利用已測試過的程式段增加品質
- 對於建立特定的加工庫，提供了可能性
- 儲存記憶體空間

1.25.1.2 副程式名稱

命名規則

在命名副程式時需遵守以下規則：

- 前兩個字元需為字母（A - Z、a - z）。
- 接下來的字元可由任意字母、數字（0 到 9）、及底線（"_"）組成。
- 最多可使用 31 個字元。

說明

SINUMERIK NC 程式語言**並不**區分大小寫。

程式名稱擴張

在建立程式時所指派的程式名稱在控制系統中可加上附加的字首與字尾作擴張。

- 字首：_N_
- 字尾：
  - 主程式：_MPF
  - 副程式：_SPF

使用程式名稱

在使用程式名稱時，例如在副程式呼叫中，所有的字首、程式名稱、及字尾組合均可使用。
範例：

使用程式名稱"SUB_PROG"的副程式可使用下列呼叫啟動：

1. `SUB_PROG`
2. `_N_SUB_PROG`
3. `SUB_PROG_SPF`
4. `_N_SUB_PROG_SPF`

說明

具有相同名稱的主程式與副程式

若主程式（.MPF）與副程式（.SPF）使用相同的名稱，當該程式名稱用於工件程式中時，需指定相關的字尾以便能給該程式獨特的識別。

1.25.1.3 巢狀的副程式

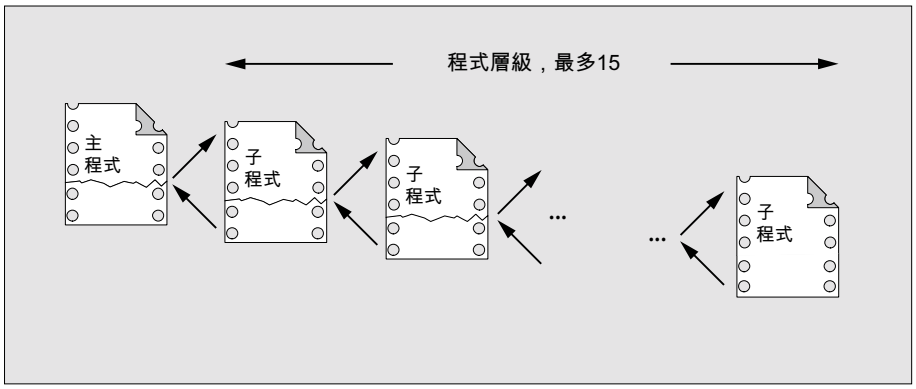
一個主程式可呼叫會呼叫更多副程式的副程式。此時，這些程式的執行順序彼此為巢狀結構。每個程式均在專屬的程式層次上執行。

巢狀深度

NC 程式語言目前提供 16 個程式層次。主程式永遠在最上層的程式層次，0 上執行。副程式永遠在呼叫後次低的程式層級上執行。因此程式層級 1 為第一副程式層級。

程式層級的分配：

- 程式層級 0：主程式層級
- 程式層級 1 至 15：副程式層級 1 至 15



中斷程式（ASUB）

若在中斷程式中呼叫副程式，則其不會在通道（n）中目前啟用的程式層級上執行，而是會在次低的程式層級（n+1）上執行。因此即便在最低的程式層級，在中斷程式中仍可有 2 個額外的程式層級（16 與 17）。

若需要超過 2 個程式層級，則必須明確的考慮建立工件程式在通道中執行的架構。換言之，僅能使用眾多程式層級中最大的層級使中斷處理能有足夠的程式層級可用。

例如，若中斷處理需要 4 個程式層級，則需將工件程式建構為最大使用程式層級 13 的架構。在中斷時，其便可使用所需要的 4 個程式層級（14 至 17）。

Siemens 週期

Siemens 週期需 3 個程式層級。因此，Siemens 週期最晚的呼叫點為：

- 工件程式處理：程式層級 12：
- 中斷程式：程式層級 14：

1.25.1.4 搜尋路徑

當呼叫副程式但未指定路徑時，控制系統會以指定的順序搜尋以下目錄：

順序	目錄	說明
1.	目前的目錄	含有呼叫程式的目錄
2.	/_N_SPF_DIR /	全域副程式目錄
3.	/_N_CUS_DIR /	使用者週期
4.	/_N_CMA_DIR /	製造商循環
5.	/_N_CST_DIR /	標準循環

1.25.1.5 形式與實際參數

形式與實際參數會與使用參數傳輸對副程式進行定義與呼叫時發生。

形式參數

當定義副程式時，欲傳送至該副程式中的參數（稱為形式參數）需定義有類型與參數名稱。
因此，形式參數定義了副程式的介面。
範例：

程式碼	註解
`PROC CONTOUR（實數 X，實數 Y）`	`; 形式參數：X 與 Y，均為實數類型`
`N20 X1=X Y1=Y`	`; 將軸 X1 移動至位置 X 並將 Y1 移動至位置 Y`
`...`	
`N100 RET`	

實際參數

當呼叫副程式時，需將絕對值或變數傳（稱為實際參數）送至副程式中。
此時，實際參數會在稍後呼叫時指派最新的值至副程式介面中。
範例：

程式碼	註解
`N10 DEF REAL WIDTH`	`; 變數定義`
`N20 WIDTH=20.0`	`; 變數配置`
`N30 CONTOUR（5.5，WIDTH）`	`; 使用實際參數進行副程式呼叫：5.5 及 WIDTH`
`...`	
`N100 M30`	

1.25.1.6 參數傳輸

定義具有參數傳輸的副程式

具有參數傳輸的副程式可使用 PROC 關鍵字及副程式所需的所有參數列表進行定義。

不完整參數傳輸

當呼叫副程式時，並非所有定義在副程式介面中的參數均需明確的傳輸。若有參數被省略，則傳輸時會以預設值"0"取代。

如此一來便可獨立識別參數的順序，但是，需永遠包含作為參數分隔字元的逗號。最後的參數除外。若在呼叫時忽略它，則最後的逗號亦會被省略。

範例：

副程式：

程式碼	註解
PROC SUB_PROG（實數 X、實數 Y、實數 Z）	; 形式參數：X、Y、和 Z
...	
N100 RET	

主程式：

程式碼	註解
PROC MAIN_PROG	
...	
N30 SUB_PROG（1.0、2.0、3.0）	; 具完整參數傳輸的副程式呼叫： X=1.0，Y=2.0，Z=3.0
…	
N100 M30	

在具有不完整參數傳輸的 N30 中進行副程式呼叫的範例：

```
N30 SUB_PROG（、2.0、3.0）        ; X=0.0，Y=2.0，Z=3.0
N30 SUB_PROG（1.0、、3.0）        ; X=1.0，Y=0.0，Z=3.0
N30 SUB_PROG（1.0，2.0）          ; X=1.0，Y=2.0，Z=0.0
N30 SUB_PROG（、、3.0）           ; X=0.0，Y=0.0，Z=3.0
N30 SUB_PROG（、、）              ; X=0.0，Y=0.0，Z=0.0
```

小心
傳址呼叫參數傳輸 使用傳址呼叫傳輸的參數在副程式呼叫時不可省略。

小心
軸資料類型 軸資料類型參數在副程式呼叫時不可省略。

確認傳輸參數

系統變數$P_SUBPAR [n] where n = 1、2、等，可用於確認參數是有明確的傳輸或是在副程式中被省略。索引 n 代表形式參數的順序。Index n = 1 表示第一形式參數，index n = 2 表示第二形式參數，依此類推。

以下程式摘錄顯示如何以第一形式參數為基礎執行確認的範例：

程式規劃	註解
PROC SUB_PROG（實數 X、實數 Y、實數 Z）	; 形式參數：X、Y、和 Z
N20 IF $P_SUBPAR[1]==TRUE	; 確認第一形式參數 X。
...	; 若形式參數已明確傳輸時會執行這些動作。
N40 ELSE	
...	; 若形式參數未傳輸時會執行這些動作。
N60 ENDIF	
...	; 一般動作
N100 RET	

1.25.2 副程式的定義

1.25.2.1 沒有參數傳輸的副程式：

功能

在定義沒有參數傳輸的副程式時，可忽略位在程式開頭處的定義指令列。

句法

```
[PROC <程式名稱>]
...
```

意義

PROC：	在程式開始處的定義操作
<程式名稱>：	程式的名稱

範例

範例 1：具有 PROC 操作的副程式

程式碼	註解
PROC SUB_PROG	; 定義指令列
N10 G01 G90 G64 F1000	
N20 X10 Y20	
...	
N100 RET	; 副程式傳回

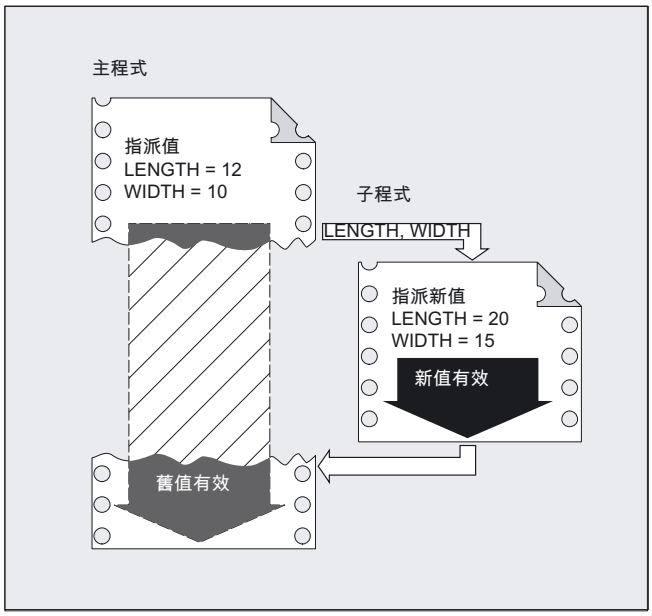
範例 2：沒有 PROC 操作的副程式

程式碼	註解
N10 G01 G90 G64 F1000	
N20 X10 Y20	
...	
N100 RET	; 副程式傳回

1.25.2.2 具有傳值呼叫參數傳輸（PROC）的副程式

函數

具有傳值呼叫參數傳輸的副程式可利用在程式名稱後的 PROC 關鍵字以及所有副程式所需具有類型與名稱的完整參數清單進行定義。 該定義操作需出現在程式指令列的第一列。
傳值呼叫參數傳輸對呼叫程式並無影響。 呼叫程式僅會將實際參數的值傳輸至副程式中。



說明

最大可傳輸 127 個參數。

句法

PROC <程式名稱>（<參數類型> <參數名稱>、等）

意義

PROC：	在程式開始處的定義操作
<程式名稱>：	程式的名稱
<參數類型>：	參數的資料類型（例如實數、整數、布林值）
<參數名稱>：	參數的名稱

注意
指定在 PROC 關鍵字之後的程式名稱需與使用者介面上指派的程式名稱相符。

範例

定義具有 2 個實數類型參數的副程式。

程式碼	註解
PROC SUB_PROG （REAL LENGTH，REAL WIDTH）	; 參數 1： 類型： 實數，名稱： 長度 參數 2：類型：實數，名稱： 寬度
...	
N100 RET	; 副程式傳回

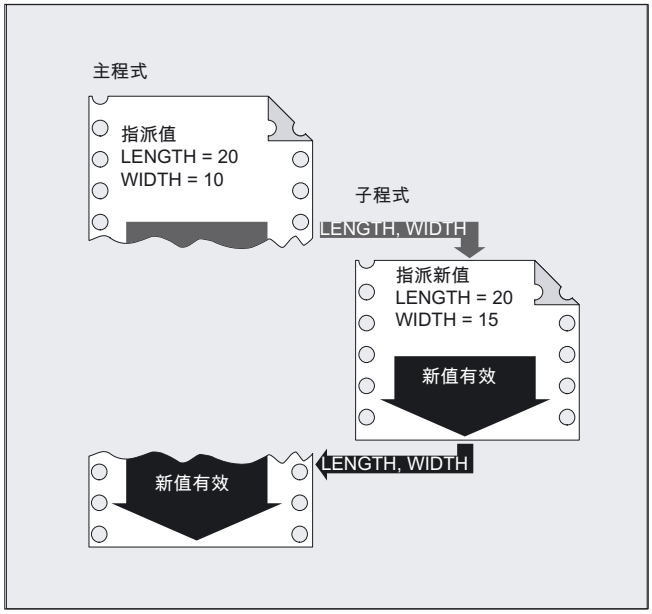
1.25.2.3 具有傳址呼叫參數傳輸（PROC、VAR）的副程式

功能

具有傳址呼叫參數傳輸的副程式可利用在程式名稱後的 PROC 關鍵字以及所有副程式所需具有 VAR 關鍵字、類型與名稱的完整參數清單進行定義。該定義操作需出現在程式指令列的第一列。

使用傳址呼叫參數傳輸時，亦會傳輸陣列的參考位址。

傳址呼叫參數傳輸會對呼叫程式造成影響。呼叫程式設計會將參考的實際參數傳輸至副程式，進而讓副程式可直接存取對應的變數。



說明

最大可傳輸 127 個參數。

說明

只有在呼叫程式（LUD）中定義了已傳輸的變數時才需要傳址呼叫參數傳輸。通道全域或 NC 全域變數不需傳輸，因其無法從副程式中直接進行存取。

句法

```
PROC <程式名稱>（VAR<參數類型> <參數名稱>、等）
PROC <程式名稱>（VAR <陣列類型> <陣列名稱>、[<m>，<n>，<o>]、等）
```

意義

PROC:	在程式開始處的定義操作
VAR:	透過參考位址傳輸之參數的關鍵字
<程式名稱>:	程式的名稱
<參數類型>:	參數的資料類型（例如實數、整數、布林值）
<參數名稱>:	參數的名稱
<陣列類型>:	陣列元素的資料類型（例如實數、整數、布林值）
<陣列名稱>:	陣列的名稱
[<m>，<n>，<o>]:	陣列大小 目前，最多可使用 3 維陣列： <m>: 1 維陣列的維度 <n>: 2 維陣列的維度 <o>: 3 維陣列的維度

注意

指定在 PROC 關鍵字之後的程式名稱需與使用者介面上指派的程式名稱相符。

說明

對於未定義陣列長度的陣列，副程式可以處理將變數長度作為正式參數的陣列。當定義二維陣列作為正式參數時，舉例來說，不會指定一維的長度。然而，必須寫入逗號。

範例：PROC <程式名稱> （VAR REAL ARRAY[, 5]）

範例

以參考至實數類型的 2 個參數定義副程式：

程式碼	註解
PROC SUB_PROG（VAR REAL LENGTH，VAR REAL WIDTH）	; 參數 1：參考的類型：實數，名稱：長度 參數 2：參考的類型：實數，名稱：寬度
...	
N100 RET	

1.25.2.4 儲存模態 G 碼功能（SAVE）

函數

SAVE 屬性表示在副程式呼叫之前，儲存了已啟動的 G 碼功能，並在副程式結尾之後，重新啟動。

小心

不可中斷的連續路徑模式

在啟用了連續路徑模式且不可由副程式呼叫中斷時，需使用 SAVE。

句法

PROC <副程式名稱> SAVE

意義

SAVE: 在副程式呼叫之前，儲存模態 G 碼功能，並在副程式結尾之後，重新儲存。

範例

模態 G 碼功能 G91 在 CONTOUR 副程式（增量尺寸）中有效。 模態 G 碼功能 G91 在主程式（絕對尺寸）中有效。 在副程式結尾之後的主程式之中，G90 再次有效，因為具有 SAVE 副程式定義的。

副程式定義：

程式碼	註解
PROC CONTOUR （REAL VALUE1） SAVE	; 具有 SAVE 參數的副程式定義
N10 G91 …	; 模態 G 碼功能 G91： 增量進給
N100 M17	; 副程式結尾

主程式：

程式碼	註解
N10 G0 X… Y… G90	; 模態 G 碼功能 G90： 絕對尺寸
N20…	
...	
N50 CONTOUR （12.4）	; 副程式呼叫
N60 X… Y…	; 使用 SAVE 重新啟用模態 G 碼功能

補充條件

框架

關於具有 SAVE 屬性的副程式，其框架之行為，會依照框架時間而有所不同，並可設定使用機械參數。

參考

功能手冊基本功能；座標軸、座標系統、框架（K2）
章節： "以 SAVE 傳回的副程式"

1.25.2.5 抑制獨立單節執行（SBLOF，SBLON）

功能

用來完成程式的單一單節停用：

當啟用了單一單節時，以 SBLOF 設計的程式會被完整執行，就像一個單節一樣。換言之，會停用單一單節執行，以完成程式。

SBLOF 係在 PROC 行之中，並保持有效，直到副程式結尾或直到它被中斷為止。在傳回指令處，會決定是否要在副程式結尾停止：

以 M17 傳回跳躍：	在副程式結尾停止
以 RET 傳回跳躍：	在副程式結尾不停止

在程式中的單一單節停用：

SBLOF 必須單獨保持在單節中。會停用在此單節之後的單一單節，直到：

- 下一個 SBLON
  或
- 到主動副程式層級結尾

句法

用來完成程式的單一單節停用：

```
PROC ... SBLOF
```

在程式中的單一單節停用：

```
SBLOF
...
SBLON
```

意義

PROC：	程式中的第一個指令
SBLOF：	用來停用單一單節執行的指令 SBLOF 可以寫入在 PROC 單節之中，或單獨存在於單節之中。
SBLON：	用來啟用單一單節執行的指令 SBLON 必須為在獨立單節中。

限制

- **單一單節停用與單節顯示**

  目前的單節顯示，可使用 DISPLOF，在循環 / 副程式中停用。若 DISPLOF 和 SBLOF 一起被程式設計，那麼循環 / 副程式呼叫會繼續在單一單節在循環 / 副程式中的停止處繼續顯示。

- **在系統 ASUB 或使用者 ASUB 中的單一單節停用**

  若單一單節在系統中停止，或使用機械參數設定 MD10702 $MN_IGNORE_SINGLEBLOCK_MASK （bit0 = 1 or bit1 = 1），停用了使用者 ASUB，那麼可藉由 ASUB 中的 SBLON 來進行程式設計，重新啟用該單一單節停止。

  若使用機械參數設定 MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP，停用了使用者 ASUB 中的單一單節停止，那麼該單一單節停止，將**無法**藉由在 ASUB 中的 SBLON 進行程式設計而重新啟用。

- **對於數個不同的單一單節執行類型，其單一單節停用的特殊部分**

  當啟用單一單節執行 SBL2（在各工件程式單節後停止）時，**沒有**執行停止，在 SBLON 單節中，若在 MD10702 $MN_IGNORE_SINGLEBLOCK_MASK 之中，將位元 12 設定為 "1"。（防止單一單節停止）。

  當單一單節執行 SBL3 啟用時（在各工件程式單節後，還在該循環中），SBLOF 指令會被停用。

範例

範例 1：在程式中的單一單節停用：

程式碼	備註
N10 G1 X100 F1000	
N20 SBLOF	; 停用單一單節
N30 Y20	
N40 M100	
N50 R10=90	
N60 SBLON	; 重新啟用單一單節
N70 M110	
N80 ...	

在 N20 與 N60 之間的區域，會以在單一單節模式中的一個步驟來執行。

範例 2：循環會以類似指令的方式來對使用者動作

主程式：

程式碼
N10 G1 X10 G90 F200
N20 X-4 Y6
N30 CYCLE1
N40 G1 X0
N50 M30

循環 CYCLE1：

程式碼	註解
N100 PROC CYCLE1 DISPLOF SBLOF	; 停用單一單節
N110 R10=3*SIN（R20）+5	
N120 IF （R11 <= 0）	
N130 SETAL（61000）	
N140 ENDIF	
N150 G1 G91 Z=R10 F=R11	
N160 M17	

CYCLE1 會進入啟用的單一單節執行，換言之，必須按下一次開始鍵來進行 CYCLE1。

範例 3：

為了要啟用一個修正過的零點偏移量以及刀具偏移量，由自行執行一個 PLC 啟動的 ASUB。

程式碼

```
N100 PROC ZO SBLOF DISPLOF
N110 CASE $P_UIFRNUM OF     0 GOTOF _G500
                            1 GOTOF _G54
                            2 GOTOF _G55
                            3 GOTOF _G56
                            4 GOTOF _G57
                            DEFAULT GOTOF END
N120 _G54: G54 D=$P_TOOL T=$P_TOOLNO
N130 RET
N140 _G54: G55 D=$P_TOOL T=$P_TOOLNO
N150 RET
N160 _G56: G56 D=$P_TOOL T=$P_TOOLNO
N170 RET
N180 _G57: G57 D=$P_TOOL T=$P_TOOLNO
N190 RET
N200 END: D=$P_TOOL T=$P_TOOLNO
N210 RET
```

範例 4：沒有以 MD10702 Bit 12 = 1 停止

初始狀態：

- 單一單節已啟用。
- MD10702 $MN_IGNORE_SINGLEBLOCK_MASK Bit12 = 1

主程式：

程式碼	註解
N10 G0 X0	; 在此工件程式行中停止。
N20 X10	; 在此工件程式行中停止。
N30 CYCLE	; 由循環產生的移動單節。
N50 G90 X20	; 在此工件程式行中停止。
M30	

循環 CYCLE：

程式碼	註解
PROC CYCLE SBLOF	; 停用單一單節停止：
N100 R0 = 1	
N110 SBLON	; 因為 MD10702 bit12=1，執行不會在工件程式行中停止。
N120 X1	; 執行在此工件程式行中停止。
N140 SBLOF	
N150 R0 = 2	
RET	

範例 5：用於巢狀程式的單一單節停用

初始狀態：

單一單節已啟用。

巢狀程式

程式碼	註解
N10 X0 F1000	; 執行在此單節中停止。
N20 UP1（0）	
PROC UP1（INT _NR） SBLOF	; 停用單一單節停止：
N100 X10	
N110 UP2（0）	
PROC UP2（INT _NR）	
N200 X20	
N210 SBLON	; 啟用單一單節停止：
N220 X22	; 執行在此單節中停止。
N230 UP3（0）	
PROC UP3（INT _NR）	
N300 SBLOF	; 停用單一單節停止：
N305 X30	
N310 SBLON	; 啟用單一單節停止：
N320 X32	; 執行在此單節中停止。
N330 SBLOF	; 停用單一單節停止：
N340 X34	
N350 M17	; SBLOF 為啟用。
N240 X24	; 執行在此單節中停止。SBLOF 為啟用。
N250 M17	; 執行在此單節中停止。SBLOF 為啟用。
N120 X12	
N130 M17	; 執行在此 傳回跳躍單節中停止。PROC 的 SBLOF 指令為啟用。
N30 X0	; 執行在此單節中停止。
N40 M30	; 執行在此單節中停止。

其它資訊

對非同步的副程式禁用單一單節

為了在一個步驟中執行 ASUB，必須在 ASUB 中以 SBLOF，程式設計一個 PROC 指令。亦可套用至函數"可編輯的系統 ASUB"（MD11610 $MN_ASUP_EDITABLE）。

可編輯系統 ASUB 之範例：

程式碼	註解
N10 PROC ASUB1 SBLOF DISPLOF	
N20 IF $AC_ASUP=='H200'	
N30 RET	; 沒有 REPOS 供模式變更所用.
N40 ELSE	
N50 REPOSA	; REPOS 在所有其他情形中。
N60 ENDIF	

於單一單節模式中程式控制

藉由單一單節執行函數，使用者可以一個單節一個單節的方式，執行工件程式。下列設定類型存在於：

- SBL1：在各機台函數單節之後，具有停止的 IPO 單一單節。
- SBL2：在各單節之後，具有停止的單一單節。
- SBL3：在循環中停止（選擇 SBL3，停用 SBLOF 指令）。

用於巢狀程式的單一單節停用

若 SBLOF 在副程式中的 PROC 指令中進行程式設計，那麼執行會在副程式以 M17 傳回跳躍處停止。此舉使得在呼叫中的程式之中的下一個單節不會被執行。若 SBLOF，沒有 SBLOF，在副程式中的 PROC 指令進行程式設計，那麼單一單節停用會啟用，執行只會在呼叫中程式的下一個加工函數單節之後，才會停止。若不想這樣，則必須在副程式中，程式設計 SBLON，且必須再傳回（M17）之前。執行不會因為以 RET 傳回跳躍到更高層級的程式而停止。

1.25.2.6 抑制目前的單節顯示（DISPLOF、DISPLON、ACTBLOCNO）

功能

目前程式單節會顯示於單節顯示中作為標準。目前單節的顯示可使用 DISPLOF 指令在循環與副程式中加以抑制。會顯示呼叫的循環或副程式，而不是目前的單節。DISPLON 指令可用於撤回對單節顯示的抑制。

DISPLOF 與 DISPLON 和 PROC 操作一同程式設計在程式指令列中，且對整個副程式均有效，且對於從其中所呼叫的不含 DISPLON 或 DISPLOF 指令的副程式亦有潛在的效果。此對於所有的 ASUB 均適用。

句法

```
PROC … DISPLOF
PROC … DISPLOF ACTBLOCNO
PROC … DISPLON
```

意義

DISPLOF:	用來停用目前單節顯示的指令。 位置: 在具有 PROC 操作的程式指令列結尾 有效範圍: 往上到從副程式傳回的跳躍或程式結尾。 **注意:** 若使用 DISPLOF 指令從該副程式中呼叫了其他副程式，則目前的單節顯示在這些副程式中亦會受到抑制，除非其中有明確的程式設計了 DISPLON。
DISPLON:	用於撤回對目前單節顯示的抑制 位置: 在具有 PROC 操作的程式指令列結尾 有效範圍: 往上到從副程式傳回的跳躍或程式結尾。 **注意:** 若使用 DISPLON 指令從該副程式中呼叫了其他副程式，則目前的單節在這些副程式中亦會顯示，除非其中有明確的程式設計了 DISPLOF。
ACTBLOCNO:	DISPLOF 和 ACTBLOCNO 屬性，表示在警報的狀態下，於警報發生處，輸出實際的單節號碼。當 DISPLOF 在較低程式層級中進行程式設計時，也能套用。 另一方面，DISPLOF，不含 ACTBLOCNO，從上一個沒有以 DISPLOF 指派的程式層級而來的循環或副程式呼叫，會顯示其單節號碼。

範例

範例 1：在循環中停用目前的單節顯示

程式碼	註解
PROC CYCLE （AXIS TOMOV，REAL POSITION） SAVE DISPLOF	; 取代停用目前的單節顯示，應顯示循環呼叫，例如：CYCLE（X，100.0）
DEF REAL DIFF	; 循環內容
G01 ...	
...	
RET	; 副程式傳回跳躍。跟在循環呼叫之後的單節，顯示在單節顯示中。

範例 2：用來輸出警報的單節顯示

副程式 SUBPROG1（含 ACTBLOCNO）：

程式碼	註解
PROC SUBPROG1 DISPLOF ACTBLOCNO	
N8000 R10 = R33 + R44	
...	
N9040 R10 = 66 X100	; 輸出警報 12080
...	
N10000 M17	

副程式 SUBPROG2 （with ACTBLOCNO）：

程式碼	註解
PROC SUBPROG2 DISPLOF	
N5000 R10 = R33 + R44	
...	
N6040 R10 = 66 X100	; 輸出警報 12080
...	
N7000 M17	

主程式：

程式碼	註解
N1000 G0 X0 Y0 Z0	
N1010 ...	
...	
N2050 SUBPROG1	; 警報輸出 ="12080 通道 K1 單節 N9040 句法錯誤文字 R10="
N2060 ...	
N2350 SUBPROG2	; 警報輸出 ="12080 通道 K1 單節 N2350 句法錯誤文字 R10="
...	
N3000 M30	

範例 3：撤回對目前單節顯示的抑制

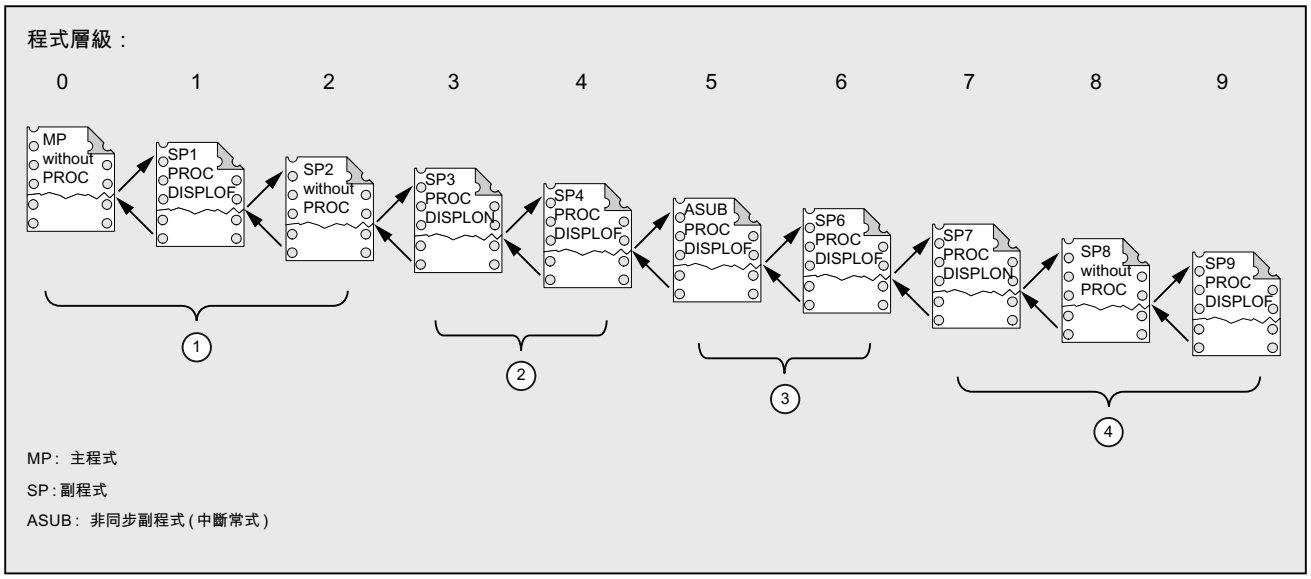
具有抑制的副程式 SUB1：

程式碼	註解
PROC SUB1 DISPLOF	; 抑制 SUB1 副程式中的目前單節顯示。取而代之的是，該單節會在 SUB1 呼叫時顯示。
...	
N300 SUB2	; 呼叫副程式 SUB2。
...	
N500 M17	

不含抑制的副程式 SUB2：

程式碼	註解
PROC SUB2 DISPLON	; 撤回對副程式 SUB2 中目前單節顯示的抑制。
...	
N200 M17	; 返回副程式 SUB1。對目前單節顯示的抑制重新儲存在 SUB1 中。

範例 4：當使用不同的 DISPLON/DISPLOF 組合時顯示回覆



① 來自程式層級 0 的工件程式指令列會顯示在目前單節顯示中。
② 來自程式層級 3 的工件程式指令列會顯示在目前單節顯示中。
③ 來自程式層級 3 的工件程式指令列會顯示在目前單節顯示中。
④ 來自程式層級 7/8 的工件程式指令列會顯示在目前單節顯示中。

1.25.2.7 以準備（PREPRO）定義副程式

功能

在啟動過程中，所有檔案皆可在 PROC 行結尾處，以 PREPRO 關鍵字來定義。

說明
這種程式準備，依機械參數的相關設定而有所不同。請遵守製造商說明。
參考資料：
功能手冊，特殊函數，前置處理（V2）

句法

```
PROC … PREPRO
```

意義

PREPRO: 在啟動期間，準備了用來辨識所有檔案的關鍵字（對於儲存在循環目錄中的 NC 程式）

以準備和副程式呼叫來讀取副程式

循環目錄會以相同的順序處理，提供在啟動期間和在副程式呼叫時，以參數處理的副程式所用。

1. _N_CUS_DIR 使用者循環
2. _N_CMA_DIR 製造商循環
3. _N_CST_DIR 標準循環

在 NC 程式共用相同名稱但有不同特性時，會啟用第一個發現的 PROC 操作，而其他的 PROC 操作則會被忽略且不發出警報訊息。

1.25.2.8 副程式返回 M17

函數

返回指令 M17（或工件程式結尾指令 M30）會出現在副程式結尾。 其會提示返回至工件程式單節中位於副程式呼叫之後的呼叫程式中。

說明

M17 與 M30 在 NC 程式語言中視為相同。

句法

```
PROC <程式名稱>
...
M17/M30
```

一般條款

在連續路徑模式中副程式返回的影響

若 M17（或 M30）單獨出現在工件程式單節中，會中斷通道中所啟用的連續路徑模式。

為避免連續路徑模式中斷，最後移動的單節中需含有 M17（或 M30）。 更進一步，需將以下機械參數設為"0"：

MD20800 $MC_SPF_END_TO_VDI = 0（沒有 M30/M17 輸出至 NC/PLC 介面）

範例

1. 在個別單節中具有 M17 的副程式

程式碼	註解
N10 G64 F2000 G91 X10 Y10	
N20 X10 Z10	
N30 M17	; 利用中斷連續路徑模式的方式返回跳躍點。

2. 在最後移動單節中具有 M17 的副程式

程式碼	註解
N10 G64 F2000 G91 X10 Y10	
N20 X10 Z10 M17	; 不使用中斷連續路徑模式的方式返回跳躍點。

1.25.2.9 RET 副程式返回

功能

RET 指令亦可用在副程式中替代 M17 返回跳躍指令。RET 必須程式設計在獨立的工件程式單節中。與 M17 類似，RET 會提示返回至工件程式單節中緊接在副程式呼叫之後的呼叫程式中。

說明

可程式設計參數以變更RET的跳躍行為（請參閱"$可參數化的副程式傳回跳躍（RET ...）(頁 154)"）。

應用

RET 操作應當隨後使用，如果 G64 連續路徑模式（G641 至 G645）不是即將被傳回中斷。

條件

使用 RET 的條件為該副程式並無 SAVE 屬性。

句法

```
PROC <程式名稱>
...
RET
```

範例

主程式：

程式碼	註解
PROC MAIN_PROGRAM	; 程式開始
...	
N50 SUB_PROG	; 副程式呼叫：SUB_PROG
N60 ...	
...	
N100 M30	; 程式結尾

副程式：

程式碼	註解
PROC SUB_PROG	
...	
N100 RET	; 提示返回跳躍至主程式中的單節 N60。

1.25.2.10 可參數化的副程式傳回跳躍（RET ...）

功能

一般而言，一個 RET 或 M17 的副程式結尾，會從副程式被呼叫之處，傳回給程式，並從副程式呼叫之後的程式行繼續處理。

然而，換句話說，也會有要讓程式到另外一個位置重新進行的應用，例如：

- 在 ISO 同源模式（在描述輪廓後）中，呼叫材料移除循環之後，重新執行程式。
- 因為錯誤掌控，而從任何副程式層級（就算在 ASUB 之後）返回到主程式。
- 透過數個程式層級傳回跳躍，作為提供編譯循環和 ISO 同源模組中的特殊應用。

在這些情況中，RET 指令會和“傳回跳躍參數”一起被程式設計。

句法

RET（"<目的地單節>"）
RET（"<目的地單節>"，<目的地單節之後之單節>）
RET（"<目的地單節>"，<目的地單節之後之單節> <傳回跳躍階層之數量>）
RET（"<目的地單節>"，，<傳回跳躍階層之數量>）
RET（"<目的地單節>"，<目的地單節之後之單節>，<傳回跳躍階層之數量>，
<傳回跳躍至程式開始處>）
RET（ ，，<傳回跳躍階層之數量>，<傳回跳躍至程式開始處>）

意義

RET: 副程式結尾（取代 M17 的使用）

<目的地單節>: 傳回跳躍參數 1

宣告為跳躍目的地，程式執行的單節，應當重新啟動。

若傳回的跳躍參數 3 沒有程式設計，那麼跳躍目的地就是在程式中，呼叫本副程式之處。

可能的資料包含：

"<單節編號>"	目的地單節的編號
"<跳躍標記>"	必須在目的地單節中設定的跳躍標記。
"<字元字串>"	在程式中必定知道的字元字串（例如程式或變數名稱）。 當程式設計字元字串時，套用下列規則： • **在結尾處空白**（與跳躍標記相反，係在結尾 e end）以“：”指定。 • **在字元字串之前**僅有一個單節號碼或跳躍標記可供設定，**沒有程式指令**。

<目的地單節之後之單節>: 傳回跳躍參數 2

請參考傳回跳躍參數 1。

類型： INT

值：

0	傳回跳躍會用在，使用傳回跳躍參數 1 指定的單節上。
> 0	傳回跳躍會用在，以傳回跳躍參數指定的單節後面。

<傳回跳躍階層之數量>: 傳回跳躍參數 3

為了要到達程式應該繼續執行的程式層級，指定必須被跳回的層級號碼。

類型： INT

值：

1	程式會在“目前程式層級－1”之處重新開始（例如沒有參數的 RET）。
2	程式會在“現行的程式層級－2”重新開始，換言之，會跳過一個層級。
3	程式會在“現行的程式層級－3”重新開始，換言之，會跳過一個層級。
...	

值域：

1 ... 15

<傳回跳躍至程式開始處>: 傳回跳躍參數 4

類型： BOOL

值：

1	若傳回跳躍至主程式中，且 **ISO 同源模式** 在該處主動，那麼程式會朝向該程式開始處分支。

說明

對於副程式傳回跳躍，以字元字串指定目的地單節搜尋，一開始總是都會在呼叫程式中，搜尋跳躍標記。

若要使用字元字串將跳躍目的地獨自定義，則不允許讓該字元字串的名稱，和跳躍標記的名稱相同，否則副程式傳回跳躍，永遠只會跳躍到跳躍標記，而不會跳躍到字元字串（請參考範例 2）。

一般條款

當透過數個程式層級進行傳回跳躍，則會求得獨立程式層級 SAVE 指令。

若有一個傳回跳躍，跳過了數個程式層級，而另有一個模態副程式啟用，如果其中有一個被跳過的程式中有取消選擇指令 MCALL，並已對模態副程式進行程式設計，那麼模態副程式仍保持有效。

⚠小心

對於跳過好幾個程式層級的傳回跳躍，程式設計師要特別小心確保程式會以正確的模態設定繼續。這是可行的，例如程式設計一個適當的主單節。

範例

範例 1：在 ASUB 執行後，重新進行主程式

程式設計	註解
N10010 CALL "UP1"	; 程式層級 0（主程式）
N11000 PROC UP1	; 程式層級 1
N11010 CALL "UP2"	
N12000 PROC UP2	; 程式層級 2
...	
N19000 PROC ASUB	; 程式層級 3（ASUB 執行）
...	
N19100 RET（"N10900", , $P_STACK）	; 副程式傳回。
N10900	; 重新執行主程式。
N10910 MCALL	; 停用模態副程式。
N10920 G0 G60 G40 M5	; 修正其餘模態設定。

範例 2：作為指定目的地單節搜尋的字元字串（字串>）

主程式：

程式碼	註解
PROC MAIN_PROGRAM	
N1000 DEF INT iVar1=1, iVar2=4	
N1010 ...	
N1200 subProg1	; 呼叫副程式“subProg1”
N1210 M2 S1000 X10 F1000	
N1220	
N1400 subProg2	; 呼叫副程式“subProg2”
N1410 M3 S500 Y20	
N1420 ..	
N1500 lab1: iVar1=R10*44	
N1510 F500 X5	
N1520 ...	
N1550 subprog1: G1 X30	; “subProg1”在此定義為跳躍標記。
N1560 ...	
N1600 subProg3	呼叫副程式“subProg3”
N1610 ...	
N1900 M30	

副程式 subProg1：

程式碼	註解
PROC subProg1	
N2000 R10=R20+100	
N2010 ...	
N2200 RET（"subProg2"）	; 在單節 N1400，將跳躍傳回給主程式

副程式 subProg2：

程式碼	註解
PROC subProg2	
N2000 R10=R20+100	
N2010 ...	
N2200 RET（"iVar1"）	; 在單節將跳躍傳回給主程式

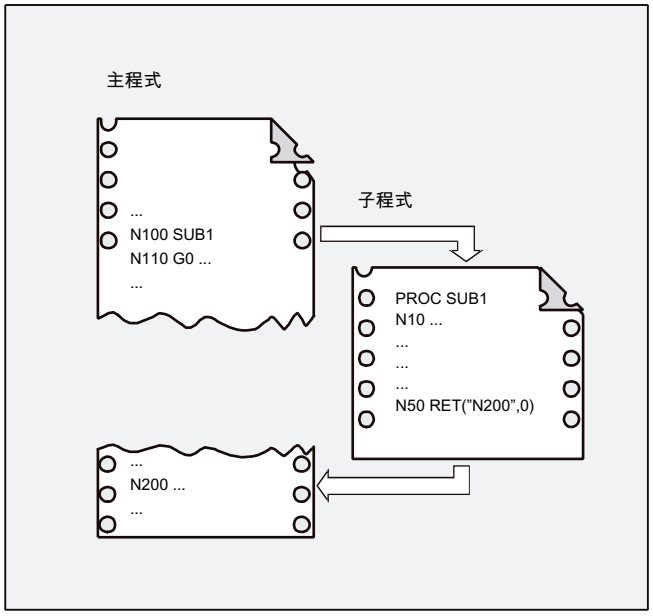
副程式 subProg3：

程式碼	註解
PROC subProg3	
N2000 R10=R20+100	
N2010 ...	
N2200 RET（"subProg1"）	; 在單節 N1550 節將跳躍傳回給主程式

其他資訊

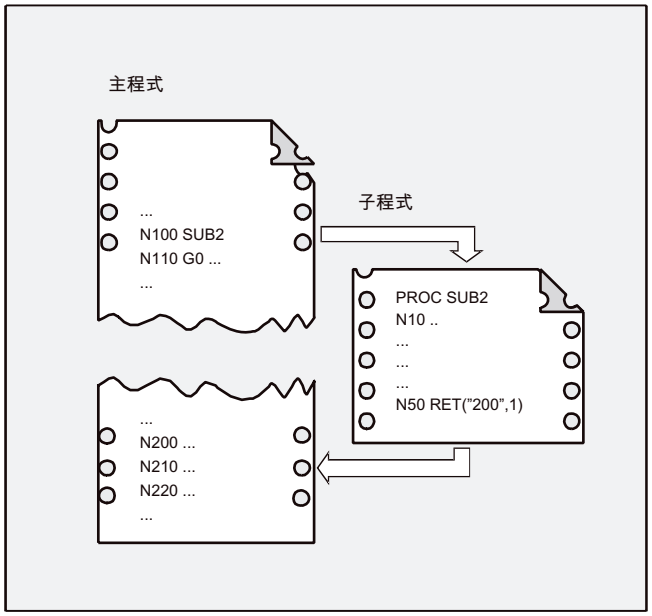
傳回跳躍參數 1 至 3，其效果的不同之處，以下圖解釋。

第一傳回跳躍參數 1 =“N200”，傳回跳躍參數 2 = 0



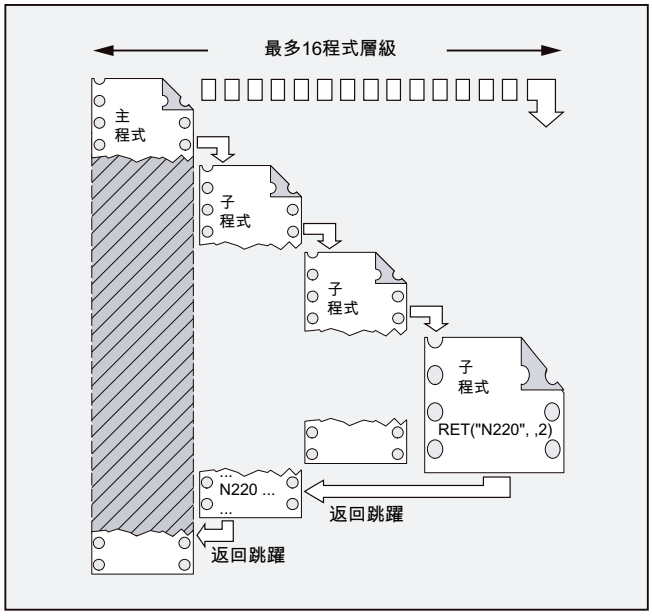
在 RET 指令之後，程式會繼續在主程式中，以 N200 單節執行。

第二傳回跳躍參數 1 =“N200”，傳回跳躍參數 2 = 1



在 RET 指令之後，程式會繼續在主程式中，以 N210）單節執行，該單節係跟在 N200 單節之後。

第三傳回跳躍參數 1 =“N200”，傳回跳躍參數 3 = 2



在 RET 指令之後，兩個程式層級會跳躍，並繼續以 N220 單節執行程式。

1.25.3 副程式呼叫

1.25.3.1 沒有參數傳輸的呼叫：

功能

副程式會以位址 L 及副程式編號或透過指定程式名稱的方式進行呼叫。

主程式也能被呼叫作為副程式。在此情況中，在主程式中設定的程式結尾 M2 或 M30，會被評估為 M17（具有傳回呼叫程式的程式結尾）。

說明

相對的，副程式也能被啟用作為主程式。

控制的搜尋策略：

有無任何*_MPF？

有無任何*_SPF？

這表示：若待呼叫的副程式名稱，和主程式名稱相同，則已發出呼叫的主程式會再被呼叫一次。一而言，這是無法預期的影響，且必須藉由指派獨特的名稱給副程式和主程式來避免。

說明

亦可從初始化檔案中呼叫不需參數傳輸的副程式。

句法

L<編號>/<程式名稱>

說明

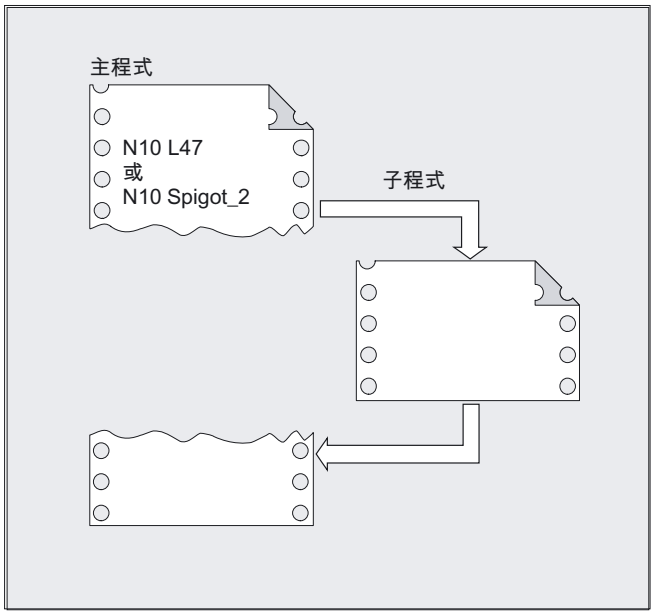
副程式呼叫必須永遠在 NC 單節中被程式設計。

意義

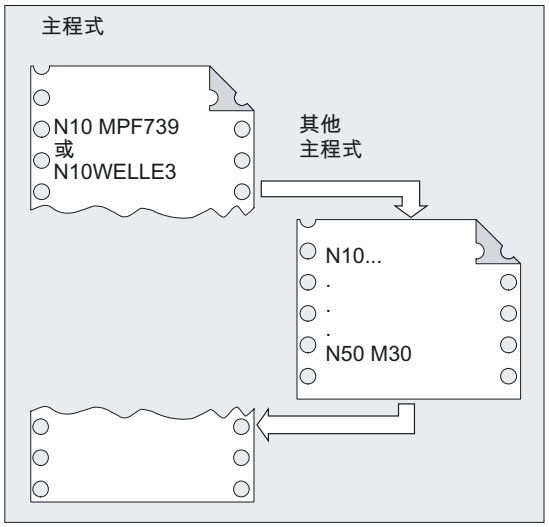
L：	副程式呼叫的位址	
<編號>：	副程式的名稱	
	類型：	INT
	值：	最多 7 個小數點位數
		注意： 名稱中的前置零是有意義的（⇒ L123，L0123 和 L00123 是三個不同的副程式）。
<程式名稱>：	副程式（或主程式）的名稱	

範例

範例 1：沒有參數傳輸的呼叫：



範例 2：以呼叫副程式的方式呼叫主程式



1.25.3.2 以參數傳輸（EXTERN）進行副程式呼叫

功能

對於具有參數傳輸的副程式呼叫，可以直接傳輸變數或值（但不是 VAR 參數）。

具有參數傳輸的副程式，必須在他們被主程式呼叫之前，先在主程式之中，以 EXTERNAL 宣告（例如，在程式開始處）。副程式和變數類型的名稱，從而會依照他們被傳輸的順序來指定。

⚠ 小心

變數類型和傳輸順序，二者必須符合在副程式中，PROC 底下所宣告的定義。在主程式和副程式中的參數名稱可以不同。

句法

```
EXTERNAL <程式名稱>（<type_Par1>，<type_Par2>，<type_Par3>）
...
<程式名稱>（<value_Par1>，<value_Par2>，<value_Par3>）
```

⚠ 小心

副程式呼叫必須永遠在 NC 單節中被程式設計。

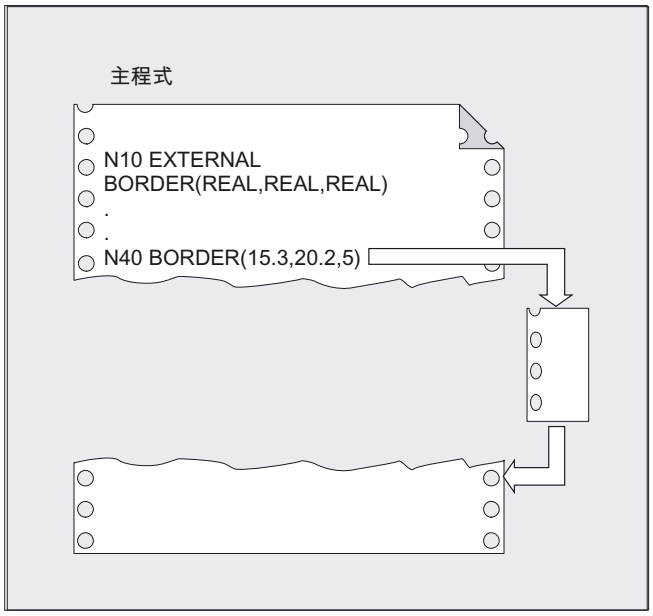
意義

<程式名稱>:	副程式的名稱
EXTERN:	作為以參數傳輸宣告副程式的關鍵字。 **注意：** 若副程式在工件中，或在全域副程式目錄中，您只需要指定 EXTERNAL。循環不需宣告為 EXTERN。
<type_par1>，<type_par2>，<type_par3>:	依序待傳輸參數之變數類型
<value_par1>，<value_par2>，<value_par3>:	供待傳輸的參數所用的變數類型

範例

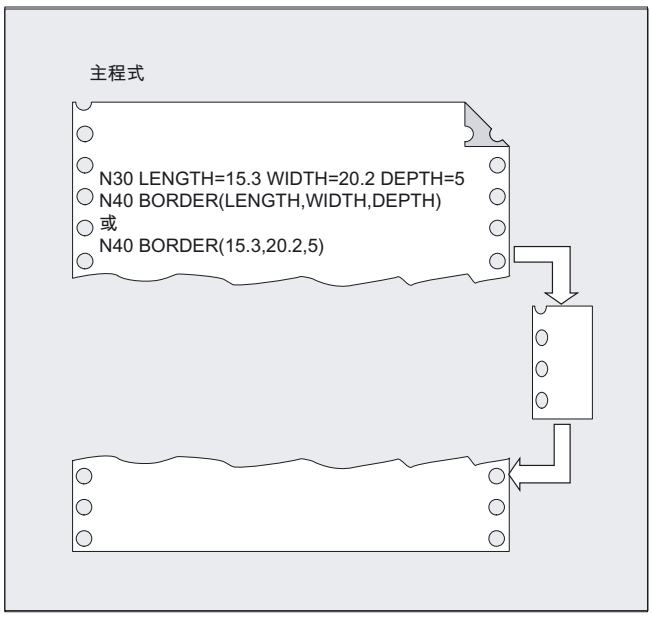
範例 1：副程式呼叫在宣告之後進行

程式碼	註解
`N10 EXTERNAL BORDERS（REAL，REAL，REAL）`	; 指定副程式
`...`	
`N40 BORDER（15.3，20.2，5）`	; 呼叫具有參數傳輸的副程式：



範例 2：副程式呼叫時不使用宣告

程式碼	註解
N10 DEF REAL LENGTH，WIDTH，DEPTH	
N20…	
N30 LENGTH=15.3 WIDTH=20.2 DEPTH=5	
N40 BORDER（LENGTH，WIDTH，DEPTH）	; or: N40 BORDER（15.3，20.2，5）



1.25.3.3 程式重覆的次數（P）

功能

若有一個副程式連續執行數次，則程式重複所需的次數可用副程式呼叫的方式，在單節中的位址 P 輸入。

⚠ 小心

具有程式重複和參數傳輸的副程式呼叫

唯有當程式被呼叫時，才會傳輸參數，例如在第一次執行時。對於其餘的重複，參數會維持不變。若您想要在程式重複期間變更參數，您必須在副程式中適當提出。

句法

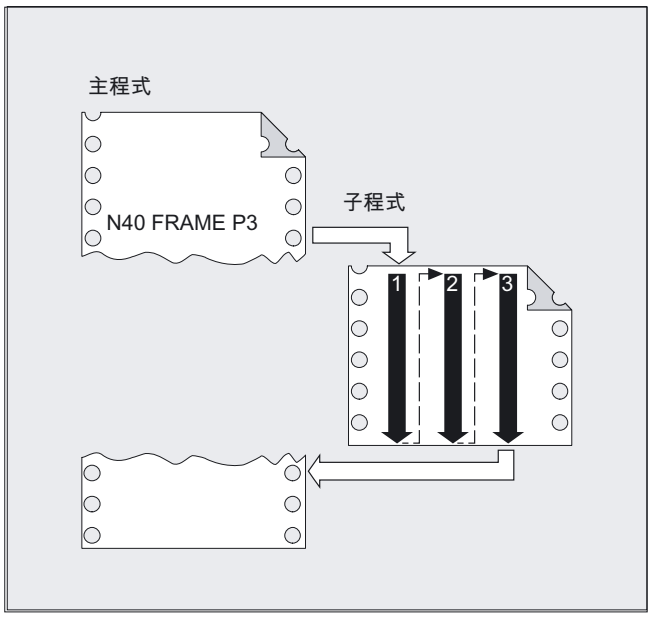
<程式名稱> P<值>

意義

<程式名稱>：	子程式呼叫	
P：	作為程式設計程式重複的位址	
<值>：	程式重覆之次數	
	類型：	INT
	值域：	1 … 9999 （沒有符號）

範例

程式碼	註解
...	
N40 FRAME P3	; BORDER 副程式要連續執行三次。
...	



1.25.3.4 模態副程式呼叫（MCALL）

功能

以 MCALL 進行模態副程式呼叫，在具有路徑動作的各個單節之後，副程式會自動被呼叫和執行。這允許了待在不同工件位置執行的自動副程式呼叫（例如，建立鑽孔圖形）。

MCALL 係作為停用沒有副程式呼叫的函數，或是藉由程式設計一個新的模態副程式呼叫，供新副程式所用。

⚠ 小心

在程式執行中，一次只有一個 MCALL 呼叫有效。參數只會為 MCALL 呼叫轉換一次。

在下列情況中，模態副程式仍以沒有動作程式設計的方式呼叫：

- 程式設計位址 S 與 F，若 G0 或 G1 有效。
- 若 G0/G1 是獨自在單節中，或與其他的 G 代碼，一起被程式設計。

句法

MCALL<程式名稱>

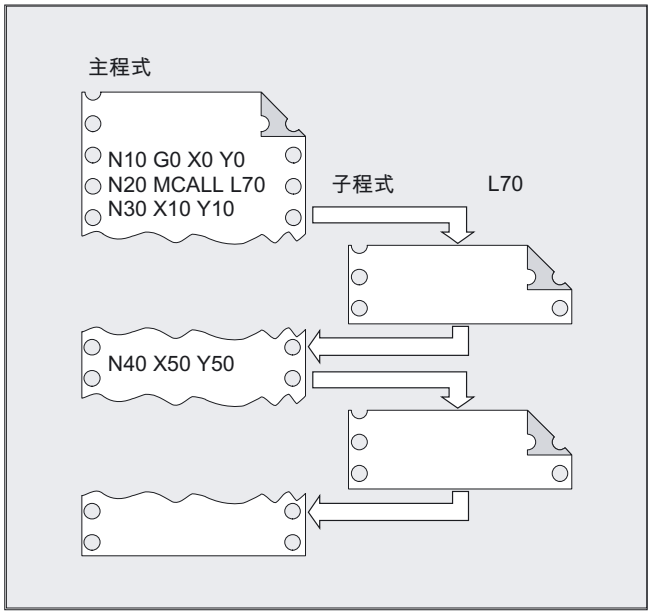
意義

MCALL:	用於模態副程式呼叫的指令
<程式名稱>:	副程式的名稱

範例

範例 1：

程式碼	註解
N10 G0 X0 Y0	
N20 MCALL L70	; 模態子程式呼叫。
N30 X10 Y10	; 逼近程式設計的位置接著執行副程式 L70。
N40 X50 Y50	; 逼近程式設計的位置接著執行副程式 L70。



範例 2：

程式碼
N10 G0 X0 Y0
N20 MCALL L70
N30 L80

在此範例中，下列具有程式設計路徑軸的 NC 單節，係在副程式 L80 中。L70 由 L80 呼叫

1.25.3.5 間接子程式呼叫（CALL）

功能

依照在程式中，特定點的一般條件而定，不同的副程式可以取消。副程式的名稱儲存在類型 STRING 的變數中。副程式呼叫可用 CALL 和變數名稱表示。

> ⚠ 小心
>
> 唯有不具參數傳輸的副程式，才能使用間接副程式呼叫。對於直接副程式呼叫，將名稱儲存在 STRING 常數中。

句法

```
CALL <程式名稱>
```

意義

CALL:	用於間接副程式呼叫的指令
<程式名稱>:	副程式的名稱（變數或常數） 類型： STRING

範例

以 STRING 常數直接呼叫：

程式碼	註解
...	
CALL "/_N_WKS_DIR/_N_SUBPROG_WPD/_N_PART1_SPF"	; 以 CALL 直接呼叫至副程式 PART1。
...	

透過變數間接呼叫：

程式碼	註解
...	
DEF STRING[100] PROGNAME	; 定義變數。
PROGNAME="/_N_WKS_DIR/_N_SUBPROG_WPD/_N_PART1_SPF"	; 將副程式 PART1 指派至 PROGNAME 變數中。
CALL PROGNAME	; 透過 CALL 及 PROGNAME 變數間接呼叫至副程式 PART1。
...	

1.25.3.6 使用呼叫程式部份的規格間接進行副程式呼叫（CALL BLOCK ... TO ...）

功能

CALL 及關鍵字組合 BLOCK ... TO 可用於間接呼叫副程式並執行由起始與結尾標籤所標示的程式部份。

句法

```
CALL <程式名稱> BLOCK <起始標籤> TO <結尾標籤>
CALL BLOCK <起始標籤> TO <結尾標籤>
```

意義

CALL:	用於間接副程式呼叫的指令
<程式名稱>:	包含待執行之程式部分的副程式名稱（變數或常數）（**規格選配**）。 類型： STRING **注意：** 若有一個<程式名稱>尚未被程式設計，由<start label>和<end label>標明的程式部分，會在目前的程式中搜尋並執行。
BLOCK ... TO ...:	供間接程式部分執行所用的關鍵字組合
<起始標籤>:	參考有待執行的程式部分開始之變數。 類型： STRING
<結尾標籤>:	參考有待執行的程式部分結尾之變數。 類型： STRING

範例

主程式：

程式碼	註解
...	
DEF STRING[20] STARTLABEL, ENDLABEL	; 起始與結尾標籤的變數定義。
STARTLABEL="LABEL_1"	
ENDLABEL="LABEL_2"	
...	
CALL "CONTUR_1" BLOCK STARTLABEL TO ENDLABEL	; 間接副程式呼叫及與呼叫程式部份搭配的識別碼。
...	

副程式：

程式碼	註解
PROC CONTUR_1 ...	
LABEL_1	; 起始標籤：程式部分執行的開始
N1000 G1 ...	
...	
LABEL_2	; 結尾標籤：程式部分執行的結尾
...	

1.25.3.7 間接呼叫以 ISO 語言（ISOCALL）進行程式設計的程式

功能

使用間接程式呼叫 ISOCALL，可以呼叫出用 ISO 語言程式設計的程式。隨後會啟用在機械參數中設定的 ISO 模式。在程式結尾，原始的執行模式再度變為有效。若機械參數中沒有設定任何 ISO 模式，則會在 Siemens 模式中呼叫副程式。

若需要對於 ISO 模式的進一步資訊，請參考
參考：
ISO 同源函數說明

句法

```
ISOCALL <program_name>
```

意義

ISOCALL:	用以設定在有效機械參數中的 ISO 模式，進行間接副程式呼叫的關鍵字。
<程式名稱>:	以 ISO 語言程式設計的程式名稱（變數或常數，類型 STRING）

範例：從 ISO 模式，呼叫具有循環程式設計的輪廓

程式碼	註解
0122_SPF	; 在 ISO 模式中的輪廓說明
N1010 G1 X10 Z20	
N1020 X30 R5	
N1030 Z50 C10	
N1040 X50	
N1050 M99	
N0010 DEF STRING[5] PROGNAME =“0122“	; Siemens 工件程式（循環）
...	
N2000 R11 = $AA_IW[X]	
N2010 ISOCALL PROGNAME	
N2020 R10 = R10+1	; 在 ISO 模式中執行程式 0122.spf
...	
N2400 M30	

1.25.3.8 以路徑規格和參數（PCALL）呼叫子程式

功能

藉由 PCALL，您可以絕對路徑和參數傳輸呼叫副程式。

句法

PCALL <路徑/程式名稱>（<參數 1>，…，<參數 n>）

意義

PCALL:	作為以絕對路徑名稱呼叫副程式的關鍵字
<路徑/程式名稱>:	絕對路徑名稱以 “/”開始，包含副程式名稱 若沒有指定絕對路徑名稱，PCALL 會以具有程式識別碼的標準副程式呼叫的方式運作。 會以沒有前導_N_和沒有副檔名的方式寫入程式識別碼。 若程式名稱有待以前導字元和副檔名進行程式設計，那麼它必須使用指令 EXTERN，以前導字元和副檔名的方式進行宣告。
<參數 1>、等:	遵守副程式之 PROC 操作的實際參數。

範例

程式碼
PCALL/_N_WKS_DIR/_N_WELLE_WPD/WELLE（parameter1，parameter2，…）

1.25.3.9 供副程式呼叫（CALLPATH）所用的延伸搜尋路徑

功能

使用指令，可以延伸供副程式呼叫所用的搜尋路徑 CALLPATH。
這表示副程式也能從非選擇的工件目錄呼叫，而不必指定副程式的完整的、絕對的路徑名稱。
搜尋路徑副檔名在輸入使用者循環（_N_CUS_DIR）之前便已完成。
搜尋路徑副檔名會因為下列事件再次被取消選擇：

- 具有空白的 CALLPATH
- 具有參數的 CALLPATH
- 工件程式結尾
- 重置

句法

CALLPATH（"<路徑名稱>"）

意義

CALLPATH:	可程式設計搜尋路徑副檔名的關鍵字 在分離的工件程式行中是可程式設計的。
<路徑名稱>:	常數或變數，類型 STRING 包含了目錄的絕對路徑名稱，亦是搜尋路徑應被延伸之處。路徑名稱以"/"開頭。路徑必須以字首和字尾完全指定。最大路徑長度為 128 位元。 若<路徑名稱>包含了空白，或沒用參數呼叫 CALLPATH，則搜尋路徑指令會再度重置。

範例

程式碼

```
CALLPATH （"/_N_WKS_DIR/_N_MYWPD_WPD"）
```

表示會設定下列的搜尋路徑（位置 5 為新增）：

1. Current directory / subprogram name
2. Current directory / subprogram identifier_SPF
3. Current directory / subprogram identifier_MPF
4. /_N_SPF_DIR/subprogram identifier_SPF
5. **/_N_WKS_DIR/_N_MYWPD/subprogram identifier_SPF**
6. /N_CUS_DIR/_N_MYWPD/subprogram identifier_SPF
7. /_N_CMA_DIR/subprogram identifier_SPF
8. /_N_CST_DIR/subprogram identifier_SPF

補充條件

- CALLPATH 檢查程式設計路徑名稱是否確實存在。當發生工件程式錯誤時，會以修正單節警報 14009 中斷執行。
- CALLPATH 亦能程式設計為 INI 檔案。只有在需要執行 INI 檔案時有效（WPD-INI 檔案或給 NC 主動資料，例如在第 1 通道中的框架_N_CH1_UFR_INI）。搜尋路徑會再次重置。

1.25.3.10 執行外部子程式（EXTCALL）

功能

EXTCALL 能從“從外部資源來執行”模式之中的 HMI 重新載入程式。所有能透過 HMI 目錄結構存取的程式，都會被重新載入並執行。

句法

EXTCALL（"<路徑/程式名稱>"）

意義

EXTCALL:	用於呼叫外部副程式的指令。
<路徑/程式名稱>:	類型 STRING 的常數 / 變數 可指定絕對 / 相對路徑或者程式名稱。 會以具有前導_N_和沒有副檔名的方式寫入程式名稱。可以使用<_>字元，將副檔名加在程式名稱之後。 範例： "/_N_WKS_DIR/_N_WELLE_WPD/_N_WELLE_SPF" 或 "SHAFT"

說明

外部副程式不可包含跳躍陳述式，例如 GOTOF、GOTOB、CASE、FOR、LOOP、WHILE 或 REPEAT。

IF-ELSE-ENDIF 結構是允許的。

您可使用副程式呼叫及巢狀 EXTCALL 呼叫。

RESET（重置），POWER ON（開機）

RESET（重置）及 POWER ON（開機）將導致外部副程式中斷，並釋出相關的載入記憶體。

在 RESET / 工件程式結尾之後，由“從外部資源執行”所選擇的副程式，仍保持不變。選擇一個 POWER ON 刪除。

範例

1. 從主機硬碟執行

系統具有進階 HMI 的 SINUMERIK solution line / powerline

主程式“_N_MAIN_MPF”儲存於 NC 記憶體並且選定為要執行之程式：

程式碼

```
N010 PROC MAIN
N020 ...
N030 EXTCALL （"ROUGHING"）
N040 ...
N050 M30
```

要重新載入的“_N_ROUGHING_SPF”副程式，儲存在目錄“_N_WKS_DIR/_N_WST1”中的主機硬碟。

副程式路徑是在 SD42700 中預設。

SD42700 $SC_EXT_PROG_PATH = "_N_WKS_DIR/_N_WST1"

程式碼

```
N010 PROC ROUGHING
N020 G1 F1000
N030 X= ... Y= ... Z= ...
N040 ...
...
...
N999999 M17
```

2. 從網路磁碟執行

系統具有 HMI sl/HMI，Advanced/HMI Embedded 的 SINUMERIK solution line / powerline。

要重新載入的“Contour2.spf”程式，係儲存在網路磁碟的“\\R4711\\Workpieces”目錄中。

程式碼

```
...
N... EXTCALL（"\\R4711\Workpieces\Contour2.spf"）
...
```

外部程式路徑

您可利用設定資料，預設外部副程式目錄的路徑：

SD42700 $SC_EXT_PROG_PATH

其值再加上 EXTCALL 呼叫中所設定的副程式或識別碼，即是欲呼叫之程式的完整路徑。

效果

含絕對路徑名稱的 EXTCALL 呼叫

如果副程式存在於指定路徑，則將在 EXTCALL 呼叫之後執行。如果不存在，則取消執行程式。

含相對路徑名稱 / 沒有路徑名稱的 EXTCALL 呼叫

如果 EXTCALL 呼叫使用相對路徑名稱或沒有路徑名稱，系統將搜尋下列程式記憶體：

- 如果 SD42700 中已預設路徑名稱，系統會先從這個路徑開始搜尋 EXTCALL 呼叫中所指定的資料（即程式名稱或相路徑名稱）。結合下列字元後即可得出絕對路徑：
  - SD42700 中的預設路徑名稱
  - “/”字元做為分隔符號
  - EXTCALL 中程式設定的副程式路徑或識別碼
- 如果在預設路徑中找不到欲呼叫的副程式，系統接著會在使用者記憶體目錄中搜尋 EXTCALL 呼叫中所指定的資料。
- 如果在目前搜尋的程式記憶體（例如 CF 卡）中找不到欲呼叫的副程式，那麼將依據第 1 點及第 2 點規定搜尋下一個程式記憶體（例如網路磁碟）。
- 找到第一個副程式後，搜尋結束。如果搜尋時沒有找到任何符合項目，則程式取消。

說明

具有內建 HMI 的 SINUMERIK 電源行

必須總是在具有內建 HMI 之中，指定絕對路徑。

外部程式記憶體

依照系統（SINUMERIK solution line / powerline）的不同，可能會將可用的使用者介面（HMI sl / HMI 進階 / HMI 內建）和所需的選項、外部程式記憶體，儲存在下列的資料載體中：

- CF 卡
- 網路磁碟
- USB 磁碟
- 主機硬碟

說明

透過具有 SINUMERIK solution line 的 USB 介面，從外部資源執行。

若要透過 USB 介面，從外部 USB 磁碟（USB 快閃磁碟），傳輸到外部程式，則只能透過 X203（稱為“TCU_1”）的介面使用。

注意

- USB 快閃磁碟不適合作為持久性記憶體媒介。
- 不建議用 USB 快閃磁碟進行“從外部來執行”。
  理由：
  - USB 快閃磁碟，在操作中可能會有脫落或接觸不良的問題。結果－加工中斷。
  - 當意外敲擊操作面版，USB 快閃磁碟可能會斷裂。

說明

-232 透過具有 SINUMERIK 電動行的 USB 介面，從外部資源透過 RS232 執行。

內崁 HMI 中，“Execution from external source”（從外部來源執行）軟鍵能夠讓外部程式經由 RS-232 介面傳送到 NC 上。

可調式重新載入記憶體（FIFO 緩衝）

NCK 需要重新載入記憶體，才能以“Execution from external source”（從外部來源執行）模式執行程式（主程式或副程式）。重新載入記憶體的大小預設為 30 Kbytes，而且它和其他所有與記憶體有關的機台資料一樣，如果其值不符合您的需求，則必須由機台製造商進行變更。

每個程式（主程式及副程式）都必須設定一個重新載入記憶體，如此才能在“Execution from external source”（從外部來源執行）模式下同時執行。

機械製造商

如果想要擴充重新載入記憶體的大小及個數，請與機台製造商連絡。

若要有關“從外部資源執行”的進一步資訊，請參考：

參考：

功能手冊，基本功能；模式群組，通道程式操作模式（K1）。

單節顯示，NC 停止的單一單節與行為

當從硬碟，以 EXTCALL 執行，只能使用“程式執行”HMI 進階 3 單節顯示。此設定用於單一單節或 NC 停止狀態。

1.25.4 循環

1.25.4.1 循環：設定使用者循環之參數

功能

您可以使用 cov.com 和 uc.com 檔案來參數化您自己的循環。

cov.com 檔案會在寄送時，包含在標準循環中，隨後再被打開。uc.com 檔案有待使用者建立。

需要在“使用者循環”目錄中的被動檔案系統，載入兩個檔案（或必須給定適當的路徑規格）：

```
; $PATH=/_N_CUS_DIR
```

於程式中。

檔案與路徑

cov.com_COM	循環概要
uc.com	循環呼叫說明

使用 cov.com –循環概觀

由標準循環所提供的 cov.com 檔案，具有下列結構：

```
%_N_COV_COM                                檔案名稱
; $PATH=/_N_CST_DIR                        路徑
; Vxxx 11.12.95 Sca cycle overview         註解行
C1（CYCLE81） drilling, centering          呼叫第一循環
C2（CYCLE82） drilling, counterboring      呼叫第二循環
...
C24（CYCLE98） chaining of threads         呼叫最後循環
M17                                        檔案結尾
```

句法

對於給一個新增的循環，必須依照下列句法新增一行：

C<編號> （<cycle_name>） comment_text

號碼：只要沒在此檔案中用過的整數；

循環名稱：將被包含的循環之程式名稱

註解文字對循環的選擇性註解文字

範例：

```
C25 （MY_CYCLE_1） usercycle_1
C26 （SPECIAL CYCLE）
```

範例：uc.com 檔案-使用者循環說明

說明係基於接下來的
範例：
對於後續兩個循環，必須新建立一個循環參數設定：

程式設計	註解
PROC MY_CYCLE_1 （REAL PAR1, INT PAR2, CHAR PAR3, STRING[10] PAR4）	
此循環具有下列傳輸參數：	
PAR1:	; 實數，範圍為–1000.001 <= PAR2 <= 123.456，預設為 100
PAR2:	; 正數，值介於 0 <= PAR3 <= 999999，預設為 0
PAR3:	; 1 個 ASCII 字元
PAR4:	; 長度 10 的字串，作為副程式名稱
...	
M17	;

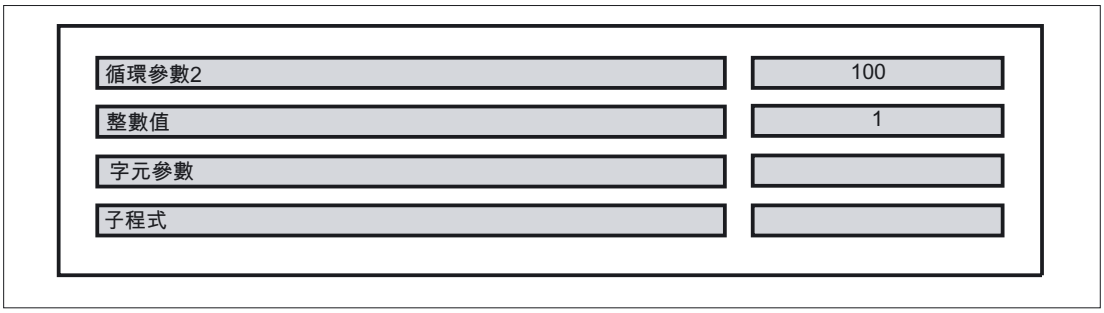
程式設計	註解
PROC SPECIAL CYCLE （REAL VALUE1, INT VALUE2）	
此循環具有下列傳輸參數：	
VALUE1:	; 沒有值域限制和預設值的實數
VALUE2:	; 沒有值域限制和預設值的整數
...	
M17	;

相關檔案 uc.com：

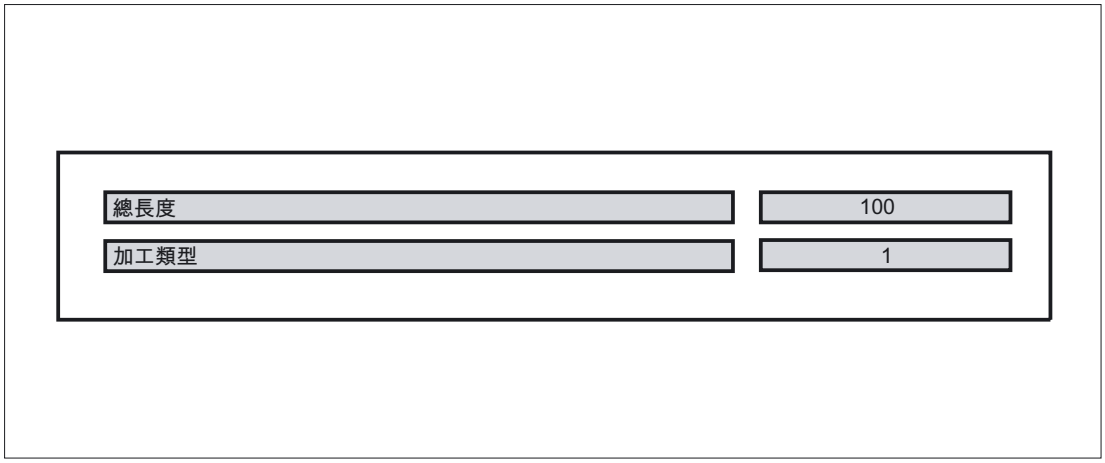
程式設計
%_N_UC_COM
; $PATH=/_N_CUS_DIR
//C25（MY_CYCLE_1） usercycle_1
（R/-1000.001 123.456 / 100 /Parameter_2 of cycle）
（I/0 999999 / 1 / Integer value）
（C//"A" / Character parameter）
（S///Subprogram name）
//C26（SPECIALCYCLE）
（R///Entire length）
（I/*123456/3/Machining type）
M17

範例：兩個循環

顯示畫面，循環 MY_CYCLE_1



顯示畫面，循環 SPECIAL CYCLE



uc.com 檔案的句法說明－使用者循環說明

各循環的表頭行：
在 cov.com 檔案中，“//”之後

```
//C <編號> （<cycle_name>） comment_text
```

範例：

```
//C25（MY_CYCLE_1） usercycle_
```

描述每一個參數的行：

```
（<data_type_id> / <minimum_value> <maximum_value>
/ <preset_value> /
```

資料類型識別碼：

R	用於實數
I	用於整數
C	用於字元（1 字元）
S	用於字串

最小值，最大值（可送出）
輸入值的臨界值，在輸入時會被確認；超過這個範圍的值無法輸入。透過切換鍵，可指定值的細目；並列在“*”之後表示，其他值便不再被認可。

範例：

```
（I/*123456/1/Machining type）
```

字串和字元類型沒有限制。

預設值（可忽略）
呼叫循環時，在對應螢幕中的是預設值；可透過操作員輸入來變更。
備註
在循環的呼叫螢幕中，最多可在參數輸入欄位上，顯示 50 個字元。

1.26 巨集技術（DEFINE ... AS）

⚠小心
使用巨集能挑選控制'的程式設計語言！因此，使用巨集時要小心注意。

功能

巨集是一連串的獨立敘述，並一起被指派一個屬於自己的名稱。G、M 與 H 函數，或 L 副程式名稱也能用來作為巨集。在程式執行期間呼叫巨集時，在程式名稱底下程式設計的敘述，會一個接著一個執行。

應用

重複的指令順序只會在巨集專用的巨集單節中（巨集檔案），或在程式開始處，才會被程式設計。隨後可在任何主程式或副程式中，呼叫巨集並執行。

啟動

為了使用 NC 程式中的巨集檔案之巨集，必須將巨集檔案下載至 NC。

句法

巨集定義：
```
DEFINE <巨集名稱> AS <指令 1> <指令 2> ...
```
在 NC 程式中呼叫：
<巨集名稱>

意義

`DEFINE ... AS`:	用來定義巨集的關鍵字組合
<巨集名稱>:	巨集名稱 只有識別碼允許作為巨集名稱。 使用巨集名稱，從 NC 程式呼叫巨集。
<操作>:	應包含在巨集中的程式設計指令。

定義巨集時的規則

- 任何識別碼，G、M、H 函數與 L 程式名稱皆可在巨集中定義。
- 也可在 NC 程式中定義巨集。
- G 碼功能巨集只能在巨集模組中，對所有的控制（模態）定義。
- 可用 2 位元對 H 和 L 函數進行程式設計。
- 可用 3 位元對 M 和 G 函數進行程式設計。

⚠小心
使用巨集，不能重新定義關鍵字和保留名稱。

一般條款

不認可巢狀巨集。

範例

範例 1：在程式開始處的巨集定義

程式碼	註解
DEFINE LINE AS G1 G94 F300	; 巨集定義:
...	
...	
N70 LINE X10 Y20	; 巨集呼叫
...	

範例 2：在巨集檔案中的巨集定義

程式碼	註解
DEFINE M6 AS L6	; 在換刀時，呼叫副程式來掌控所需的資料傳輸。在副程式中輸出實際的 M 函數（例如，M106）。
DEFINE G81 AS DRILL（81）	; 仿效 DIN-G 碼功能。
DEFINE G33 AS M333 G333	; 在螺紋切削同步期間，需要 PLC。藉由機械參數，將 G 碼功能 G33，重新命名為 G333，讓使用者可以進行程式設計。

範例 3：外部巨集檔案

在將外部巨集檔案讀入控制之後，必須將巨集檔案下載至 NC。只有這樣，才能在 NC 程式中使用巨集。

程式碼	註解
%_N_UMAC_DEF	
; $PATH=/_N_DEF_DIR	; 用戶專屬巨集
DEFINE PI AS 3.14	
DEFINE TC1 AS M3 S1000	
DEFINE M13 AS M3 M7	; 主軸順時針，切削水開啟
DEFINE M14 AS M4 M7	; 主軸逆時針，切削水開啟
DEFINE M15 AS M5 M9	; 主軸停止，切削水關閉
DEFINE M6 AS L6	; 呼叫換刀程式
DEFINE G80 AS MCALL	; 取消選擇鑽孔循環
M30	

檔案與程式管理

2

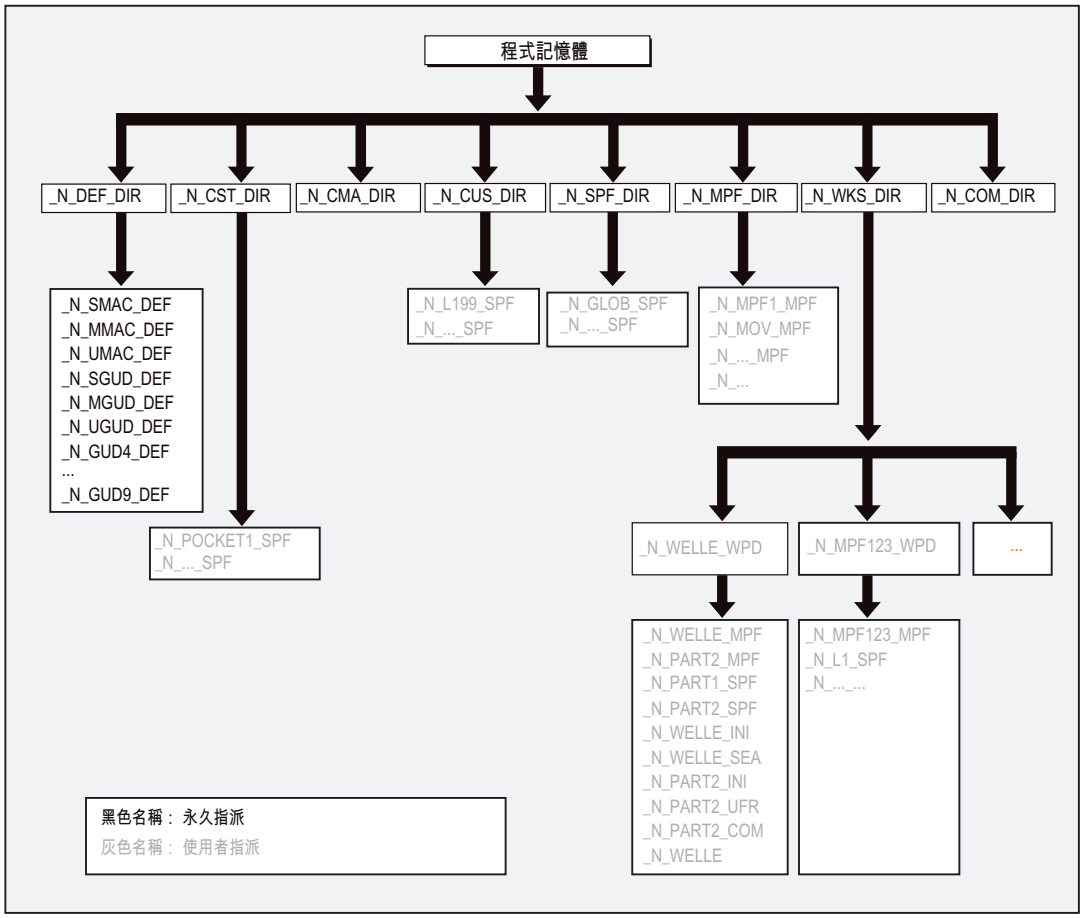
2.1 程式記憶體

功能

檔案與程式（例如，主程式與副程式、巨集定義）會儲存在非揮發性程式記憶體中（→ 被動檔案系統）。

參考資料：
功能手冊，延伸功能，記憶體設定（S7）

有些檔案類型也會暫時儲存於此；如有需要，它們也可以傳輸至工作記憶體（例如，對於在特定工件上加工的初始化目的）。



標準目錄

目錄的標準資料如下：

資料夾	內容
_N_DEF_DIR	資料模組與巨集模組
_N_CST_DIR	標準循環
_N_CMA_DIR	製造商循環
_N_CUS_DIR	使用者週期
_N_WKS_DIR	工件
_N_SPF_DIR	全域副程式
_N_MPF_DIR	主程式
_N_COM_DIR	註解

檔案類型

主記憶體中可以儲存下列檔案類型：

檔案類型	說明
name_MPF	主程式
name_SPF	副程式
name_TEA	機械參數
name_SEA	設定資料
name_TOA	刀具偏移
name_UFR	零點偏移量/框架
name_INI	Initialization files
name_GUD	通用使用者資料
name_RPA	R 參數
name_COM	註解
name_DEF	全域使用者資料與巨集之定義

工件主目錄（_N_WKS_DIR）

工件主目錄存在於程式記憶體的標準設定中，在 _N_WKS_DIR 名稱底下。工件主目錄包含所有您已程式設計的工件之工件目錄。

工件目錄（..._WPD）

若要讓資料和程式掌控有更多談性的特定資料和程式，可使其成為群組或儲存在獨立的工件目錄中。

工件目錄包含了所有需要用來加工工件的檔案。這些可以是主程式、副程式、任何初始化程式及註解檔案。

第一次啟動工件程式時，會執行一次初始化程式，依選擇程式（依照機械參數 MD11280 $MN_WPD_INI_MODE）有所不同。

範例：

為 SHAFT 工件所建立的工件目錄_N_WELLE_WPD，包含了下列檔案：

檔案	意義
_N_SHAFT_MPF	主程式
_N_PART2_MPF	主程式
_N_PART1_SPF	副程式
_N_PART2_SPF	副程式
_N_SHAFT_INI	工件資料的一般初始化程式
_N_SHAFT_SEA	設定資料初始化程式
_N_PART2_INI	工件 2 程式資料的一般初始化程式
_N_PART2_UFR	工件 2 程式框架資料的初始化程式
_N_SHAFT_COM	註解檔案

在外部 PC 上建立工件目錄

下面說明的步驟會在外部資料站中執行。請直接參考控制上的您的操作指南'，檔案與程式管理（從 PC 至控制系統）。

以路徑名稱（$PATH=...）建立一個工件目錄

目的地路徑$PATH=...會在檔案的第二行中指定。檔案隨後儲存於特定路徑。

範例：

程式碼

```
%_N_SHAFT_MPF
; $PATH=/_N_WKS_DIR/_N_SHAFT_WPD
N40 G0 X... Z...
...
M2
```

檔案_N_WELLE_MPF 儲存在目錄/_N_WKS_DIR/_N_WELLE_WPD 中.

建立一個沒有路徑名稱的工件目錄

若遺失了路徑名稱，副檔名為_SPF 的檔案，會儲存在目錄/_N_SPF_DIR 中，副檔名為_INI 的檔案，會儲存在 RAM 中，而所有其他檔案會儲存在目錄/_N_MPF_DIR 中。

範例：

程式碼

```
%_N_SHAFT_SPF
...
M17
```

檔案_N_WELLE_SPF 儲存在目錄/_N_SPF_DIR 中。

選擇加工用的工件

可以選擇一個工件目錄，用來在通道中執行。若有一個具有**相同名稱**的主程式，或只有一個單一的主程式（_MPF）儲存在此目錄中，便會自動被選擇執行。

參考：
/BAD/操作手冊 HMI 進階；“工作清單”與“選擇一個用來執行的程式”

搜尋副程式呼叫的路徑

當呼叫副程式（或初始化檔案）時，若沒有在工件程式中明確指定搜尋路徑，則呼叫程式會在固定的搜尋路徑上進行搜尋。

具有絕對路徑的副程式呼叫

範例

程式碼

```
...
CALL"/_N_CST_DIR/_N_CYCLE1_SPF"
...
```

不具絕對路徑的副程式呼叫

通常不會以指定路徑來呼叫程式。
範例

程式碼

```
...
CYCLE1
...
```

會依下列順序於下列目錄中搜尋呼叫程式：

號碼	資料夾	意義
1	目前的目錄 / *名稱*	工件主目錄或標準目錄_N_MPF_DIR
2	目前的目錄 / *名稱_SPF*	
3	目前的目錄 / *名稱_MPF*	
4	/_N_SPF_DIR / *名稱_SPF*	全域副程式
5	/_N_CUS_DIR / *名稱_SPF*	使用者週期
6	/_N_CMA_DIR / *名稱_SPF*	製造商循環
7	/_N_CST_DIR / *名稱_SPF*	標準循環

副程式呼叫的程式設計搜尋路徑（CALLPATH）

CALLPATH 工件程式指令係用來延伸副程式呼叫的搜尋路徑。

範例：

程式碼

```
CALLPATH （"/_N_WKS_DIR/_N_MYWPD_WPD"）
...
```

依照特定程式設計，搜尋路徑會儲存在位置 5（使用者循環）之前。

若需要以 CALLPATH 進行副程式呼叫的可程式設計搜尋路徑之相關資訊，請參考“以 CALLPATH 延伸副程式呼叫的搜尋路徑”一章。

2.2 工作記憶體（CHANDATA，COMPLETE，INITIAL）

功能

工作記憶體包含目前的系統，以及和控制（主動檔案系統）一起操作的使用者資料，例如：

- 主動機械參數
- 刀具偏移量資料
- 工作偏移
- ...

初始化程式

這些是和工作記憶體資料一起被初始化的程式。下列檔案類型可供使用：

檔案類型	意義
name_TEA	機械參數
name_SEA	設定資料
name_TOA	刀具偏移
name_UFR	零點偏移量/框架
name_INI	Initialization files
name_GUD	通用使用者資料
name_RPA	R 參數

您將在操作手冊中，針對操作員介面，對所有檔案類型上的資訊進行微調。

資料區

可在將要套用的不同區域中組織資料。例如，一個控制系統可以具有數個通道，或像大多數的情況一樣，會有數個軸供其自由使用。

下列存在：

識別碼	資料區
NCK	NCK 特定資料
CH<n>	通道專屬資料（<n>指定通道名稱）
AX<n>	軸專屬資料（<n>指定機械軸的數量）
TO	刀具資料
COMPLETE	所有資料

在外部 PC 建立初始化程式

資料區域辨識碼和資料類型辨識碼，可以用來決定那個區域將在儲存資料時成為一個元件：

_N_AX5_TEA_INI	軸 5 的機械參數
_N_CH2_UFR_INI	通道 2 框架
_N_COMPLETE_TEA_INI	所有機械參數

初始啟動控制時，會自動載入一個集合的資料，以確保控制的適當操作。

多通道控制的步驟（CHANDATA）

提供數個通道使用的 CHANDATA（<通道編號>），僅在檔案_N_INITIAL_INI 中認可。這是已初始化的參考，其中所有資料的調試檔。

程式碼	註解
%_N_INITIAL_INI	
CHANDATA（1）	
	; 加工軸指派，通道 1：
$MC_AXCONF_MACHAX_USED[0]=1	
$MC_AXCONF_MACHAX_USED[1]=2	
$MC_AXCONF_MACHAX_USED[2]=3	
CHANDATA（2）	
	; 加工軸指派，通道 2：
$MC_AXCONF_MACHAX_USED[0]=4	
$MC_AXCONF_MACHAX_USED[1]=5	
CHANDATA（1）	
	; 軸機械參數
	; 精確停止視窗粗調
$MA_STOP_LIMIT_COARSE[AX1]=0.2	; 軸 1
$MA_STOP_LIMIT_COARSE[AX2]=0.2	; 軸 2
	; ；精確停止視窗微調：
$MA_STOP_LIMIT_FINE[AX1]=0.01	; 軸 1
$MA_STOP_LIMIT_FINE[AX1]=0.01	; 軸 2

⚠ 小心

CHANDATA 敘述

在工件程式中，CHANDATA 指令只能設定給 NC 程式有在執行的通道使用；換言之，該指令可以用來保護 NC 程式，使其不會在工作通道中被執行。

若發生錯誤則取消處理程式。

說明

工作清單中的 INI 檔案，不包含任何 CHANDATA 指令。

儲存初始化程式（COMPLETE，INITIAL）

可將工作記憶體的檔案，儲存在外部 PC，並可再次從該處讀入。

- 以 COMPLETE 來儲存檔案。
- INITIAL 係用來在所有區域之上建立一個 INI 檔案（_N_INITIAL_INI）。

讀入初始化程式

注意
若以名稱“INITIAL_INI”讀入檔案。那麼在檔案中所有沒供應的資料，都會使用標準資料來初始化。唯有機械參數是例外。這表示**設定資料、刀具資料、ZO、GUD 值、...**係由標準資料（通常為“零”）供應。

例如，檔案 COMPLETE_TEA_INI 適用於讀入獨立機械參數。控制僅能預期在此檔案中的機械參數。這就是其他資料區域在此情況中仍保持不受影響的原因。

載入初始化程式

若 INI 程式僅使用一個通道的資料，那麼也能被選為，和被呼叫成為工件程式。這表示也有可能初始化程式控制資料。

2.3 步驟編輯器中的結構指令（SEFORM）

功能

結構指令 SEFORM 會在步驟編輯器（以編輯器為基礎的程式輔助）評估以產生 HMI Advanced 的步驟檢視。步驟檢視可增進 NC 副程式的可讀性。

句法

SEFORM（<區段名稱>，<等級>，<圖示>）

意義

SEFORM（）	結構指令與參數之函數呼叫<區段名稱>，<等級>及<圖示>	
<區段名稱>	作動中步驟的識別碼	
	類型： STRING	
<等級>	主或子層級的索引	
	類型： INT	
	值： 0	主層級
	1，...，<n>	較低層級 1，...，較低層級<n>
<圖示>	顯示於本節的圖示名稱。	
	類型： STRING	

說明

SEFORM 指令會在步驟編輯器中產生。

使用<區段名稱>傳輸的參數會與 MSG 指令類似，以主要同步執行的方式儲存於 OPI 變數中。在被下個 SEFORM 指令覆寫之前，資訊不變。工件程式的重置與結束清除內容。

NCK 在工件程式執行期間會檢查<等級>與<圖示>但不會進一步進行處理。

參考

如需更多關於以編輯器為基礎的程式輔助，請參閱
HMI 進階操作手冊。

保護區

3

3.1 保護區（CPROTDEF、NPROTDEF）定義

函數

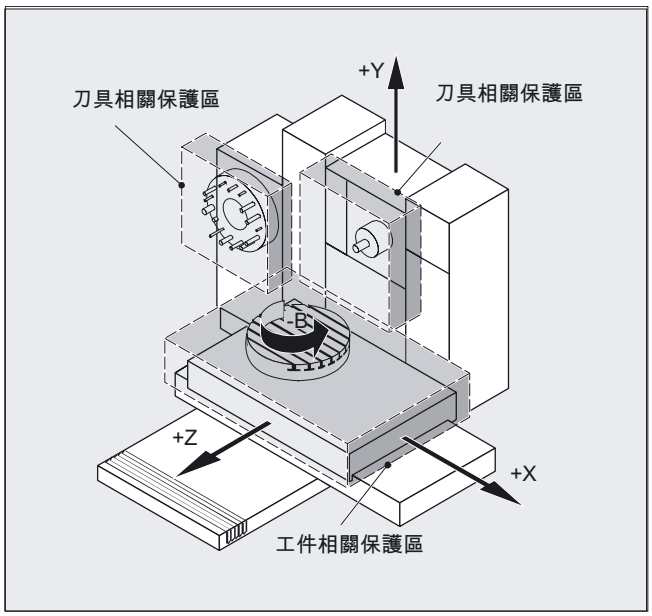
您可以使用保護區保護機台上的多個元件、其組件與工件，確保正確移動。

刀具導向保護區：

對於屬於刀具的零件（如刀具、刀把）

工件導向保護區：

對於屬於工件的零件（如工件的零件、鉗緊表、主軸夾頭、尾座）。



句法

```
DEF INT NOT_USED
CPROTDEF（<n>，<t>，<applim>，<applus>，<appminus>）
NPROTDEF（<n>，<t>，<applim>，<applus>，<appminus>）
EXECUTE（NOT_USED）
```

意義

DEF INT NOT_USED:	定義區域變數、資料類型 INTEGER （請參閱"動作同步動作"一節）
CPROTDEF:	定義通道專屬保護區（僅供 NCU 572/573）
NPROTDEF:	定義機台專屬保護區
EXECUTE:	結束定義
<n>:	已定義之保護區號碼
<t>:	保護區類型 TRUE: **刀具**相關保護區 FALSE: **工件**相關保護區
<applim>:	第三尺寸的限制類型 0: 無限制 1: 增加方向上的限制 2: 減少方向上的限制 3: 正與負方向限制
<applus>:	3 維中正向的限制值
<appminus>:	3 維中負向的限制值
NOT_USED:	錯誤變數對於含 EXECUTE 的保護區無作用

其他資訊

保護區之定義

保護區之定義包括：

- 通道專屬保護區的 CPROTDEF
- 機台專屬保護區的 NPROTDEF
- 保護區輪廓說明
- 使用 EXECUTE 結束定義

當保護區在 NC 工件程式中啟動後，您可為保護區之參考點指定相對的偏移量。

輪廓說明參考點

工件導向保護區於基本座標系統中定義。

刀具導向保護區會參考刀盤參考點 F 定義。

保護區輪廓定義

保護區輪廓於選擇的平面最多可指定 11 個移動動作。 移動至輪廓會是第一個移動動作。 有效的保護區是輪廓剩下的區域。 CPROTDEF 或 NPROTDEF 及 EXECUTE 間程式設計之移動動作將不會執行，僅會定義保護區。

平面

所需之平面會在 G17、G18、G19 於 CPROTDEF 或 NPROTDEF 之前選擇，且在 EXECUTE 之後不可變更。 套用必須不得於 CPROTDEF 或 NPROTDEF 及 EXECUTE 間程式設計。

輪廓元件

您可以執行下列動作：

- G0、G1 表示直線輪廓元素
- G2 表示順時針圓弧區段（僅供工件導向保護區）
- G3 表示於逆時針方向之圓弧區段。

說明

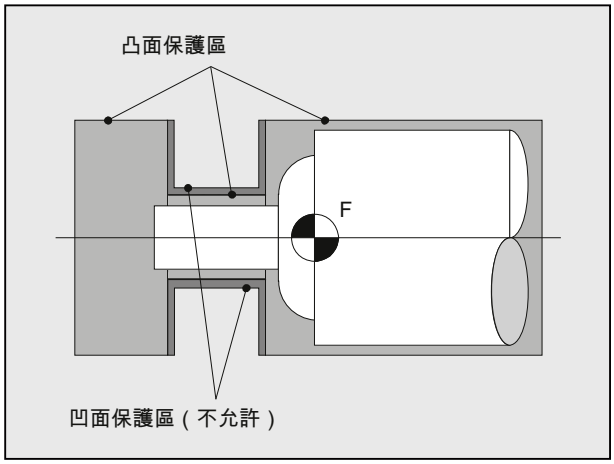
若完整的圓弧描述保護區，則必須將它分為兩個半圓。 不允許 G2、G3 或 G3、G2 的順序。 如有必要，必須插入簡短的 G1 單節。

輪廓說明最後的點必須與第一個相同。

外部保護區（僅可供工件相關保護區）必須以順時針方向定義。

對於對稱旋轉保護區（如主軸夾頭），您必須描述完整的輪廓且不只於旋轉中心！

刀具相關保護區需永遠為凸面。 若需要凹面保護區，它應被次分為數個凸面保護區。



限制

保護區定義期間：

- 銑刀或刀鼻半徑補正無須啟用
- 轉換無須啟用
- 必須無任何框架為啟用。

或必須無以程式設計之原點復歸（G74），固定點逼近（G75），單節搜尋停止或程式結尾。

3.2 啟動 / 停用保護區（CPROT、NPROT）

功能

啟動與預先啟動先前定義的保護區，以便監控衝突、停用保護區。
能在同一通道上同時啟動的最大保護區數量，定義在機台資料中。
如果與刀具相關的保護區不在啟動狀態中，則刀具路徑的檢核運作針對與工件相關的保護區。

說明
如果與工件相關的保護區不在啟動狀態中，則保護區監控功能不運作。

句法

```
CPROT（<n>，<state>，<xMov>，<yMov>，<zMov>）
NPROT（<n>，<state>，<xMov>，<yMov>，<zMov>）
```

含義

CPROT:	呼叫通道專屬保護區（僅供 NCU 572/573）
NPROT:	呼叫機台專屬保護區
<n>:	保護區號碼
<state>:	狀態參數
	0: 停用保護區
	1: 預啟動保護區
	2: 啟動保護區
	3: 預啟動含條件停止之保護區
<xMov>，<yMov>，<zMov>:	於幾何軸上移動已定義之保護區

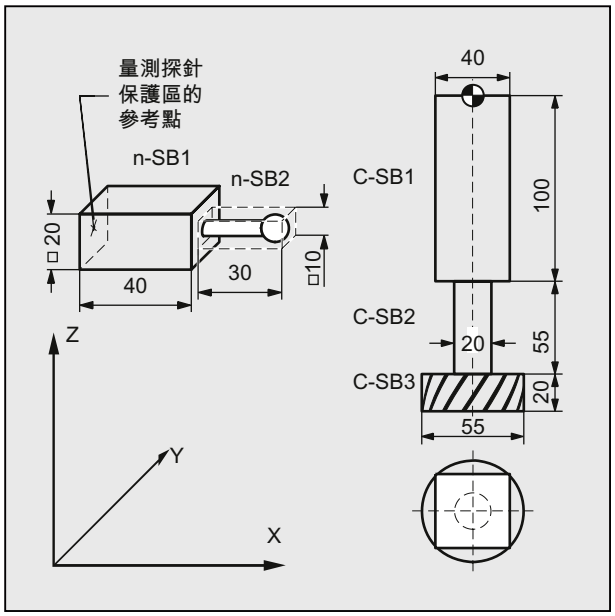
範例

銑刀與量測探針間可能的碰撞會在銑削機台上監控。 量測探針之位置可在功能啟動時以偏移量定義。 為此，已定義下列保護區：

- 量測探針座（n-SB1）與量測探針本身（n-SB2）之機台專屬與工件相關之保護區。
- 銑刀刀盤（c-SB1）、銑刀握把（c-SB2）及銑刀本身（c-SB3）之通道專屬與刀具導向保護區。

於 Z 方向之所有保護區的方向。

功能啟用時量測探針之參考點位置必須為 X = -120、Y = 60 及 Z = 80。



程式碼	註解
DEF INT PROTECTB	; 輔助說明變數定義
保護區 G17 定義	; 設定方向
NPROTDEF（1，FALSE，3，10，—10）G01 X0 Y—10	; 保護區 n—SB1
X40	
Y10	
X0	
Y—10	
EXECUTE（PROTECTB）	
NPROTDEF（2，FALSE，3，5，—5）	; 保護區 n—SB2
G01 X40 Y—5	
X70	
Y5	
X40	
Y—5	
EXECUTE（PROTECTB）	

程式碼	註解
CPROTDEF（1，TRUE，3，0，—100）	; 保護區 c—SB1
G01 X—20 Y—20	
X20	
Y20	
X—20	
Y—20	
EXECUTE（PROTECTB）	
CPROTDEF（2，TRUE，3，—100，—150）	; 保護區 c—SB2
G01 X0 Y—10	
G03 X0 Y10 J10	
X0 Y—10 J—10	
EXECUTE（PROTECTB）	
CPROTDEF（3，TRUE，3，—150，—170）	; 保護區 c—SB3
G01 X0 Y—27，5	
G03 X0 Y27，5 J27，5	
X0 Y27，5 J—27，5	
EXECUTE（PROTECTB）	
啟用保護區	
NPROT（1，2，—120，60，80）	; 以偏移量啟動保護區 n—SB1
NPROT（2.2，—120，60，80）	; 以偏移量啟動保護區 n—SB2
CPROT（1，2，0，0，0）	; 以偏移量啟動保護區 c—SB1
CPROT（2，2，0，0，0）	; 以偏移量啟動保護區 c—SB2
CPROT（3，2，0，0，0）	; 以偏移量啟動保護區 c—SB3

其他資訊

啟用狀態（<state>）

- **<state>=2**

  保護區通常以 status = 2 於工件程式中啟用。

  即使對於機台導向之保護區，狀態永遠為通道專屬。

- **<state>=1**

  若 PLC 使用者程式設定由 PLC 使用者程式生效的保護區，所需之預先啟動將由 status = 1 實現。

- **<state>=3**

  在含條件停止之預啟動情況下，系統絕對不會停在違反預啟動保護區的正前方，只有保護區已啟動時才會停止。保護區在特殊狀況下才啟動時，這有助於處理不受中斷。注意：如果在動作之前才立即啟動保護區，由於煞車斜率的結果，將導致系統移動到保護區內。

  含條件停止的預啟動可透過使用 status = 3 來達成。

- **<state>=0**

  保護區已停用，因此以 Status = 0 停用。無須偏移量。

（預）啟動之保護區移動

偏移量可於 1、2 或 3 尺寸發生。 偏移量參照至：

- 於工件專屬保護區之機械零點。
- 刀具專屬保護區之刀盤參考點 F。

啟動後的狀態

保護區可直接於啟動與後續參考點逼近後啟動。 因此系統變數$SN_PA_ACTIV_IMMED[<n>]或$SC_PA_ACTIV_IMMED[<n>]必須設定為 TRUE。 永遠以 Status = 2 啟動且無偏移量。

多重啟用保護區

保護區可於數個通道中同時啟動（如有兩個相反面之尾座）。 保護區僅當已參考所有幾何軸之後監控。

下述應用：

- 在單一通道中保護區無法以不同的偏移量同時啟動。
- 機台導向保護區於兩個通道中必須有相同的方向。

3.3 檢查保護區差異、工作區限制與軟體限制（CALCPOSI）

功能

檢查是否從已定義之起點啟動的 CALCPOSI 函數，幾何軸可以已定義之路徑移動且不違反軸限制（軟體限制）、工作區限制或保護區。

若已定義之路徑無法移動，將傳回最大允許路徑。

CALCPOSI 函數已於副程式中預定義。它必須單獨存在於單節中。

句法

```
Status=CALCPOSI（_STARTPOS，_MOVDIST，_DLIMIT，_MAXDIST，_BASE_SYS，
_TESTLIM）
```

意義

狀態

0: 函數正常，
可完整於指定之路徑移動。

–: 在_DLIMIT 中至少一元件為負

–: 轉換計算中發生一錯誤

若無法於指定之路徑移動，將傳回一正、十進位編碼值：

個位數（違反限制的類型）：

1: 軟體限制限制了移動路徑。

2: 工作區限制限制了移動路徑。

3: 軟體限制限制了移動路徑。

若同時違反數個限制（如軟體限制及保護區），於特定移動路徑之導致最嚴重限制之限制將以個位數表示。

十位數：

10:

初始值違反限制

20:

特定直線違反限制。若終點無違反任何限制本身，但自起點至終點之路徑會導致一值違反（如經過保護區、非線性轉換於 WCS 弧線軟體限制），亦將傳回該值。

百位數

100:

正限制值違反（僅當個位數為 1 或 2 時，如軟體限制與工作區限制）

100:

NCK 保護區違反（僅當個位數為 3 時）

200:

負限制值違反（僅當個位數為 1 或 2 時，如軟體限制與工作區限制）

200:

通道專屬保護區違反（僅當個位數為 3 時）

千位數

1000:

透過係數軸違反的限制數量將倍增（僅當個位數為 1 或 2 時，如軟體限制與工作區限制）。

軸計數從 1 開始且參照已違反的軟體限制（個位數 = 1）至加工軸與違反的工作區限制（個位數 = 2）至幾何軸。

1000:

違反保護區數量之係數將倍增（僅當個位數為 3 時）

若違反數個保護區，於移動路徑的最大限制的限制結果會以保護區的百位數及千位數顯示

_STARTPOS 橫座標 [0]、縱座標 [1] 及套用 [2] 於（工件座標系統（WCS））中的起始值

_MOVEDIST 橫座標 [0]、縱座標 [1] 及套用 [2] 的增量路徑定義

_DLIMIT [0] - [2]: 指派至幾何軸之最小間距。

[3]: 若無幾何軸可至派時，指派至非線性轉換之線性機械軸的最小間距。

[4]: 若無幾何軸可至派時，指派至非線性轉換之旋轉機械軸的最小間距。若將監控 SW 限制，則僅供特殊轉換。

_MAXDIST 供傳回值之陣列 [0] - [2]。全部三個幾何軸的增量路徑，無違反相關機械軸之軸限制的已定義最小間距。

若未限制移動路徑，此傳回參數之內容與_MOVDIST 的內容相同。

_BASE_SYS FALSE 或未指定參數：

評估位置與長度資料時，將評估群組 13 之 G 代碼（G70、G71、G700、G710；英制 / 公制）。若已啟用 G70 且基本系統為公制（或 G17 以英吋啟用），將於基本系統中提供 WCS 系統變數$AA_IW[X] 及 $AA_MW[X]），且若有需要，會使用 CALCPOSI 功能轉換。

TRUE：

評估位置與長度資料時，永遠會獨立使用控制的基本系統群組 13 中有效 G 的值。

_TESTLIM 將檢查限制（二進制編碼）：

1: 監控軟體限制

2: 監控工作區限制

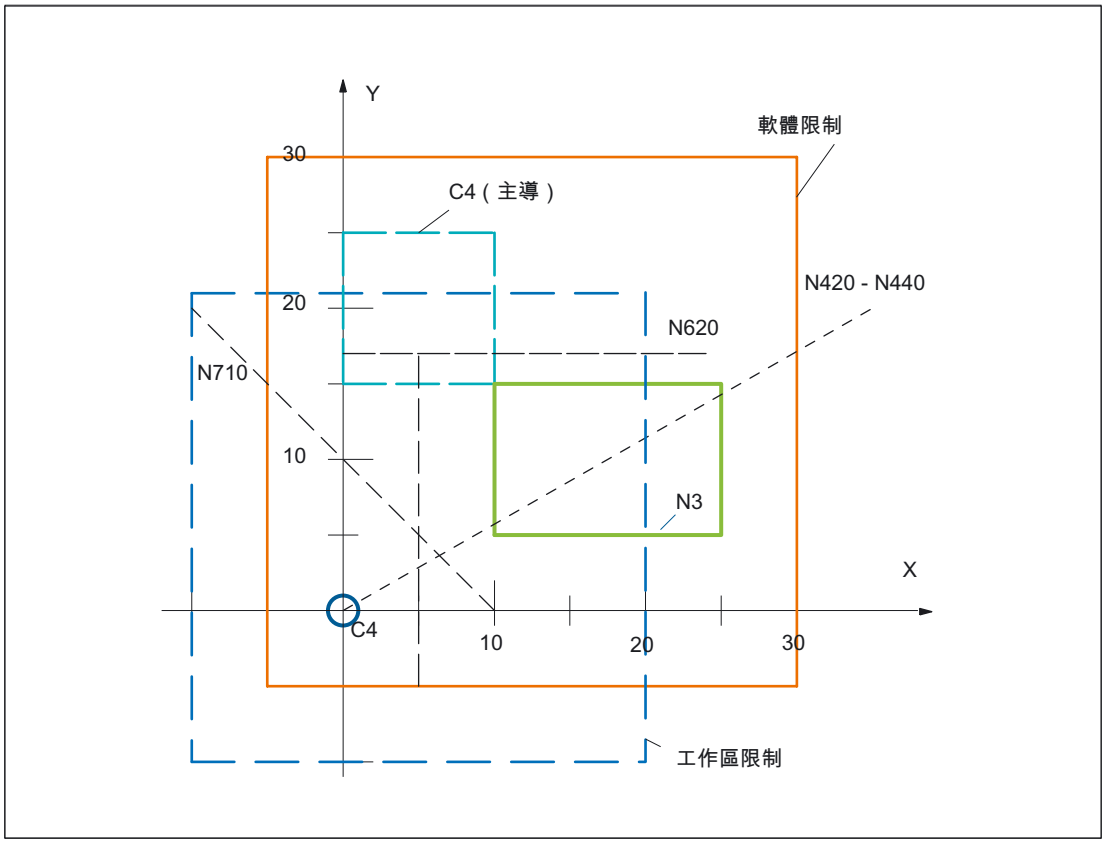
3: 監控有效保護區

4: 監控預啟動保護區

增加值以合併。預設值：15；檢查全部。

範例

於範例中（見圖），將顯示 X 軟體限制與工作區域限制。此外，已定義三個保護區，兩個通道專屬保護區 C2 及 C4 以及 NCK 保護區 N3。C2 為圓弧、有效、刀具相關之保護區，其半徑為 2 毫米。C4 為正方形、預啟用、工件相關之保護區，其邊長為 10 毫米。N3 為矩形、有效保護區，其邊長為 10 毫米與 15 毫米。於下列 NC 中，一開始，已粗略定義保護區與工作欄位限制，且會指派數個參數呼叫 CALCPOSI 功能。各 CALCPOSI 呼叫的結果簡列於範例結尾的表格中。



程式碼	註解
N10 def real _STARTPOS[3]	
N20 def real _MOVDIST[3]	
N30 def real _DLIMIT[5]	
N40 def real _MAXDIST[3]	
N50 def int _SB	
N60 def int _STATUS	
N70 cprotdef（2，true，0）	; 刀具相關保護區
N80 g17 g1 x–y0	
N90 g3 i2 x2	
N100 i–x–	
N110 execute（_SB）	

程式碼	註解
N120 cprotdef（4，false，0）	; 工件相關保護區
N130 g17 g1 x0 y15	
N140 x10	
N150 y25	
N160 x0	
N170 y15	
N180 execute（_SB）	
N190 nprotdef（3，false，0）	; 機台相關保護區
N200 g17 g1 x10 y5	
N210 x25	
N220 y15	
N230 x10	
N240 y5	
N250 execute（_SB）	
N260 cprot（2，2，0，0，0）	; 啟動或預啟動保護區
N270 cprot（4，1，0，0，0）	
N280 nprot（3，2，0，0，0）	
N290 g25 XX=—YY=—	; 定義工作區限制群組
N300 g26 xx= 20 yy= 21	
N310 _STARTPOS[0] = 0.	
N320 _STARTPOS[1] = 0.	
N330 _STARTPOS[2] = 0.	
N340 _MOVDIST[0] = 35.	
N350 _MOVDIST[1] = 20.	
N360 _MOVDIST[2] = 0.	
N370 _DLIMIT[0] = 0.	
N380 _DLIMIT[1] = 0.	
N390 _DLIMIT[2] = 0.	
N400 _DLIMIT[3] = 0.	
N410 _DLIMIT[4] = 0.	
；多個功能呼叫	; 其他起點
N420 _STATUS = calcposi（_STARTPOS，_MOVDIST，_DLIMIT，_MAXDIST）	
N430 _STATUS = calcposi（_STARTPOS，_MOVDIST，_DLIMIT，_MAXDIST，，3）	
N440 _STATUS = calcposi（_STARTPOS，_MOVDIST，_DLIMIT，_MAXDIST，，1）	
N450 _STARTPOS[0] = 5.	; 其他目的地
N460 _STARTPOS[1] = 17.	
N470 _STARTPOS[2] = 0.	
N480 _MOVDIST[0] = 0.	
N490 _MOVDIST[1] =—.	
N500 _MOVDIST[2] = 0.	

程式碼	註解
；多個功能呼叫	
N510 _STATUS = calcposi（_STARTPOS，_MOVDIST，_DLIMIT，_MAXDIST，，14）	
N520 _STATUS = calcposi（_STARTPOS，_MOVDIST，_DLIMIT，_MAXDIST，，6）	
N530 _DLIMIT[1] = 2.	
N540 _STATUS = calcposi（_STARTPOS，_MOVDIST，_DLIMIT，_MAXDIST，，6）	
N550 _STARTPOS[0] = 27.	
N560 _STARTPOS[1] = 17.1	
N570 _STARTPOS[2] = 0.	
N580 _MOVDIST[0] =–.	
N590 _MOVDIST[1] = 0.	
N600 _MOVDIST[2] = 0.	
N610 _DLIMIT[3] = 2.	
N620 _STATUS = calcposi（_STARTPOS，_MOVDIST，_DLIMIT，_MAXDIST，，12）	
N630 _STARTPOS[0] = 0.	
N640 _STARTPOS[1] = 0.	
N650 _STARTPOS[2] = 0.	
N660 _MOVDIST[0] = 0.	
N670 _MOVDIST[1] = 30.	
N680 _MOVDIST[2] = 0.	
N690 trans x10	
N700 arot z45	
N710 _STATUS = calcposi（_STARTPOS，_MOVDIST，_DLIMIT，_MAXDIST）	
N720 M30	

測試結果於範例中：

單節編號 N...	_STATUS	_MAXDIST [0] （= X）	_MAXDIST [1] （= Y）	註解
420	3123	8.040	4.594	保護區 SB N3 違反。
430	1122	20.000	11.429	未監控任何保護區，違反工作區限制。
440	1121	30.000	17.143	現在僅軟體限制監控有效。
510	4213	0.000	0.000	起點違反保護區 C4
520	0000	0.000	–.000	未監控預啟動保護區 C4。可完全於已定義之路徑移動。
540	2222	0.000	–.000	由於_DLIMIT[1]=2，移動路徑受到工作區限制的限制。
620	4223	–.000	0.000	由於 C2 與_DLIMIT[3]，至 C4 之間距共 4 毫米。間距 C2－N3 為 0.1 毫米且不對移動路徑造成限制。
710	1221	0.000	21.213	含轉譯與旋轉之框架有效 _MOVDIST 中允許的移動路徑，套用於移動且旋轉的座標系統中（WCS）。

特殊情況與細節

所有的路徑資料永遠輸入為半徑，對於含有效 G 代碼“DIAMON”的平面軸亦然。若其中一相關軸之工件無法完全移動，將會依據_MAXDIST 的傳回值減少其他軸的路徑，讓產生的終點位在指定的路徑上。

允許不為一或多個相關軸定義軟體限制、操作範圍限制或保護區。所有限制僅當有相關或參照全之軸時才監控。任何相關旋轉軸僅當它們不是模組軸時才監控。

與一般移動操作相同，軟體限制的監控與操作範圍限制視有效設定（選擇軟體設定 1、軟體設定 2、GWALIMON/WALIMOF 的介面訊號，操作範圍限制特定啟用的設定參數以及操作範圍限制之監控是否須考量有效刀具之半徑的指定）而定。

某些動態轉換（如 TRANSMIT）機械軸的位置無法自工件座標系統（WCS）中單獨決定（非唯一性）。於一般移動操作中，唯一性通常從先前於 WCS 中對應至機械軸連續移動之連續移動記錄與條件所產生。當使用 CALCPOSI 功能監控軟體限制，目前的機台位置可在這類狀況下用來決定唯一性。若有需要，必須在 CALCPOSI 之前程式設計 **STOPRE**，將有效的機械軸位置輸入至功能中。

不保證在指定移動路進上的移動於_DLIMIT[3] 中指定的保護區區隔永遠有效。因此若是傳回 _MOVDIST 中的終點增加了此長度，雖然直線將會很接近保護區但將不違反任一保護區。

說明

您可於
/PG/ 基本程式設計中取得更多關於工作區限制的詳細資訊，

或於
/FB1/ 功能手冊、基本功能；軸監控、保護區（A3）中取得關於軟體限制的詳細資訊。

4 特殊監控指令

4.1 逼近編碼位置（CAC、CIC、CDC、CACP、CACN）

功能

您可使用下列指令透過儲存於機械參數表中的固定軸位置之位置編號移動線性及旋轉軸。此類型的程式設計稱為“逼近已編碼位置”。

句法

```
CAC（<n>）
CIC（<n>）
CACP（<n>）
CACN（<n>）
```

意義

CAC（<n>）	從位置編號 n 逼近已編碼位置
CIC（<n>）	從實際位置編號開始，逼近已編碼位置 n 向前（+n）或向後（-n）定位
CDC（<n>）	沿最短路徑從位置編號 n 逼近位置 （僅供旋轉軸）
CACP（<n>）	從位置編號 n 以正方向逼近已編碼位置 （僅供旋轉軸）
CACN（<n>）	從位置編號 n 以負方向逼近已編碼位置 （僅供旋轉軸）
<n>	於機械參數表中的位置編號 值域：0、1、...（表位置的最大編號－1）

範例：逼近定位軸的已編碼位置

程式設計碼	註解
N10 FA[B]=300	; 定位軸 B 的進給率
N20 POS[B]=CAC（10）	; 從位置編號 n 逼近已編碼位置 10
N30 POS[B]=CIC（-4）	; 從“目前位置編號”－4 逼近已編碼位置

參考

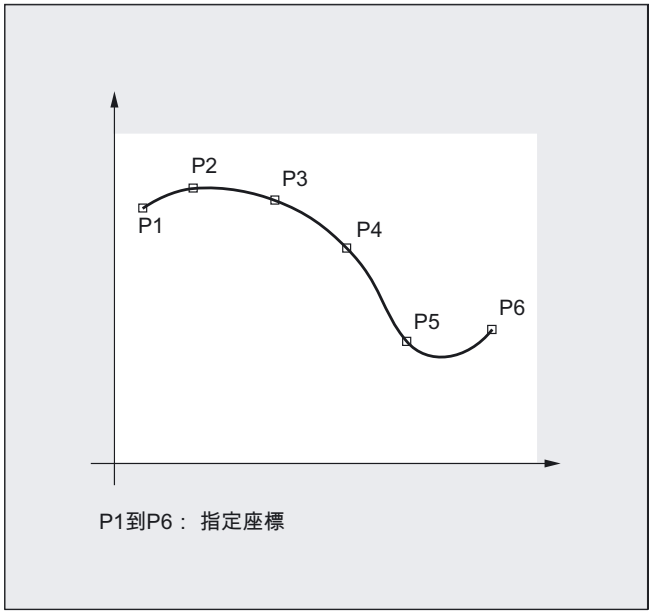
- 功能手冊延伸功能；索引軸（T1）
- 功能手冊，已同步動作

4.2 曲線插補（ASPLINE、BSPLINE、CSPLINE、BAUTO、BNAT、BTAN、EAUTO、ENAT、ETAN、PW、SD、PL）

功能

隨機曲線工件輪廓無法於可分析形狀中精確地定義。這就是這些輪廓類型於曲線僅使用有限數量的點之原因，例如當數位化曲線時。沿著曲線的點必須相連已定義輪廓，已產生工件的數位化平面。曲線插補允許此動作。

曲線能定義自二次或三次多項式形成的曲線。沿著曲線之曲線的點特性可視所使用之曲線類型定義。



SINUMERIK solution line 提供下列曲線類型：

- A 曲線
- B 曲線
- C 曲線

句法

一般：

```
ASPLINE X... Y... Z... A... B... C...
BSPLINE X... Y... Z... A... B... C...
CSPLINE X... Y... Z... A... B... C...
```

可為 B 曲線額外程式設計：

```
PW=<n>
SD=2
PL=<值>
```

可為 A 與 C 曲線額外程式設計：

```
BAUTO / BNAT / BTAN
EAUTO / ENAT / ETAN
```

意義

曲線插補類型：

ASPLINE	啟動 A 曲線插補的指令
BSPLINE	啟動 B 曲線插補的指令
CSPLINE	啟動 C 曲線插補的指令
	ASPLINE，BSPLINE 與 CSPLINE 指令模態有效且屬於動作指令群組。

沿著曲線的點與檢查點：

X... Y... Z... A... B... C...	直角座標位置

點重（僅於 B 曲線）：

PW	使用 PW 指令，可為沿著曲線上之所有點程式設計所謂的“點重”。		
<n>	“點重”		
	值域：	0 ≤ n ≤ 3	
	Increment：	0.0001	
	影響：	n > 1	檢查點會更吸引曲線。
		n < 1	檢查點會較不吸引曲線。

曲線角度（僅 B 曲線）：

SD	標準會使用三度多邊形，不過若是程式設計 SD=2，則將使用二度多邊形。

節點間的距離（僅 B 曲線）：

PL	節點間的距離適合於內部計算。控制也可加工預定義所謂使用 PL 指令於參數間隔長度指定之節點間距。	
<值>	參數間隔長度	
	值域：	作為路徑尺寸

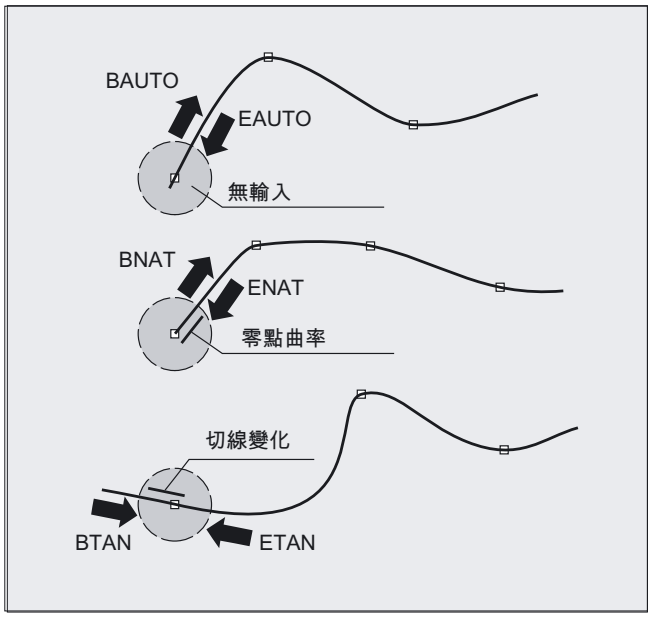
於曲線開始的變化行為（僅 A 或 C 曲線）：

BAUTO	變化行為無規格開始由第一點的位置決定。
BNAT	無彎曲
BTAN	切線變化至上個單節（刪除位置）

於曲線結尾的變化行為（僅 A 或 C 曲線）：

EAUTO	變化行為無規格結尾由最後一點的位置決定。
ENAT	無彎曲

ETAN	切線變化至上個單節（刪除位置）



說明

可程式設計之變化行為對於 B 曲線無影響。B 曲線永遠為切線以檢查多邊形的起點與終點。

補充條件

- 可能會使用刀具半徑補正。
- 會於投射之平面上進行碰撞監控。

範例

範例 1：B 曲線

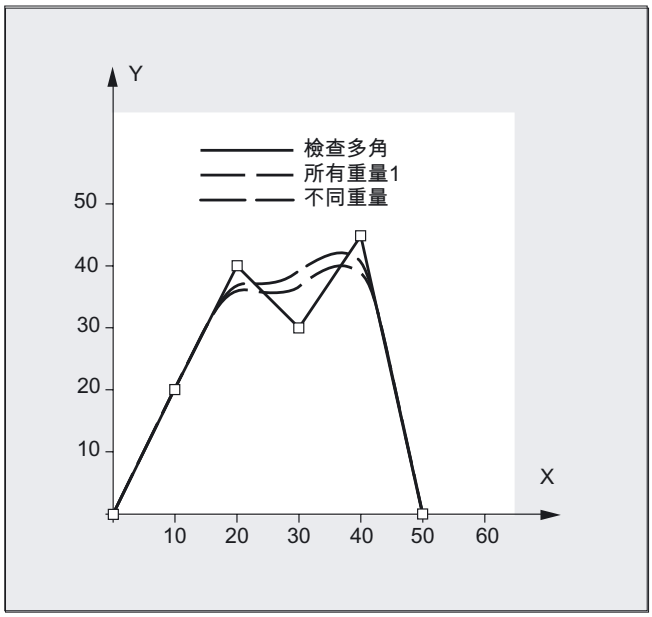
程式碼 1（所有重量 1）

```
N10 G1 X0 Y0 F300 G64
N20 BSPLINE
N30 X10 Y20
N40 X20 Y40
N50 X30 Y30
N60 X40 Y45
N70 X50 Y0
```

程式碼 2（不同重量）

```
N10 G1 X0 Y0 F300 G64
N20 BSPLINE
N30 X10 Y20 PW=2
N40 X20 Y40
N50 X30 Y30 PW=0.5
N60 X40 Y45
N70 X50 Y0
```

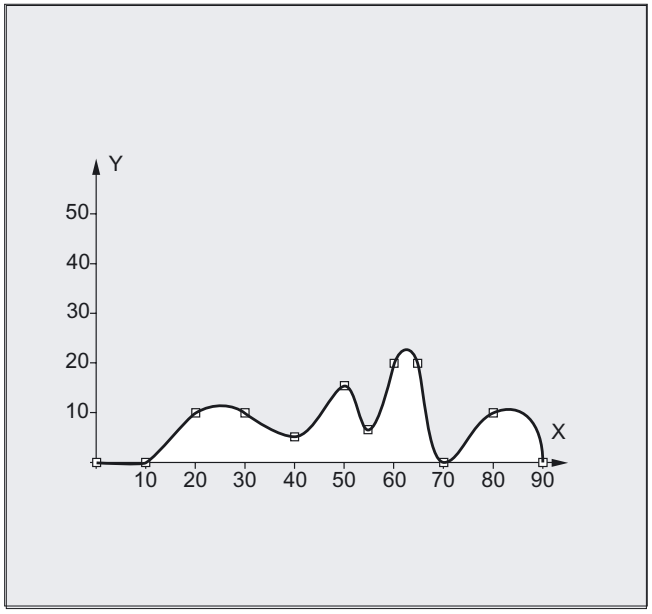
程式碼 3（檢查多邊形）	註解
N10 G1 X0 Y0 F300 G64	
N20	; 不適用
N30 X10 Y20	
N40 X20 Y40	
N50 X30 Y30	
N60 X40 Y45	
N70 X50 Y0	



範例 2：C 曲線，於開始及結尾無彎曲

程式碼

```
N10 G1 X0 Y0 F300
N15 X10
N20 BNAT ENAT
N30 CSPLINE X20 Y10
N40 X30
N50 X40 Y5
N60 X50 Y15
N70 X55 Y7
N80 X60 Y20
N90 X65 Y20
N100 X70 Y0
N110 X80 Y10
N120 X90 Y0
N130 M30
```



範例 3：曲線插補（A 曲線）及座標轉換（ROT）

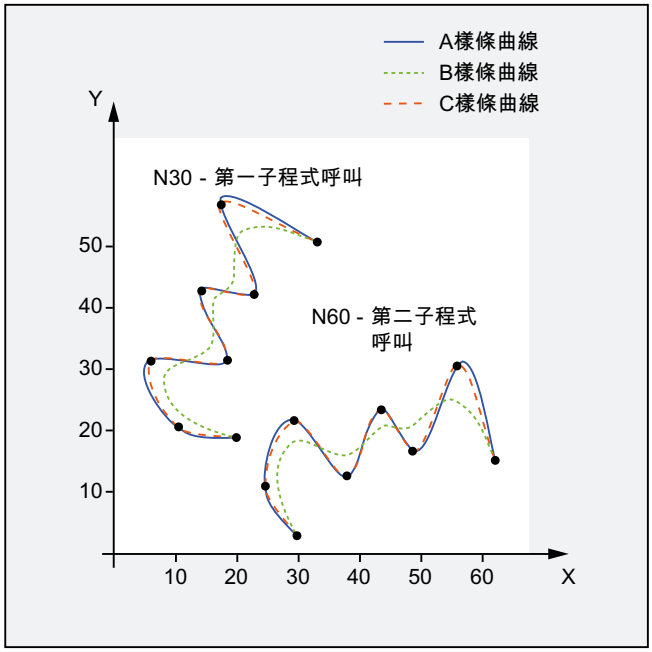
主程式：

程式碼	註解
N10 G00 X20 Y18 F300 G64	; 逼近起點
N20 ASPLINE	; 啟動插補類型 A 曲線。
N30 CONTOUR	; 第一個副程式呼叫。
N40 ROT Z-45	; 座標轉換 WCS 透過 Z 軸-45° 的旋轉。
N50 G00 X20 Y18	; 逼近輪廓起點。
N60 CONTOUR	; 第二個副程式呼叫。
N70 M30	; 程式結尾

副程式“contour”（包括沿著曲線之點的座標）：

程式碼
N10 X20 Y18
N20 X10 Y21
N30 X6 Y31
N40 X18 Y31
N50 X13 Y43
N60 X22 Y42
N70 X16 Y58
N80 X33 Y51
N90 M1

除了曲線，從範例程式（ASPLINE）產生的結果，下列圖示包含了啟動 B 或 C 曲線插補（BSPLINE、CSPLINE）可獲得的曲線：



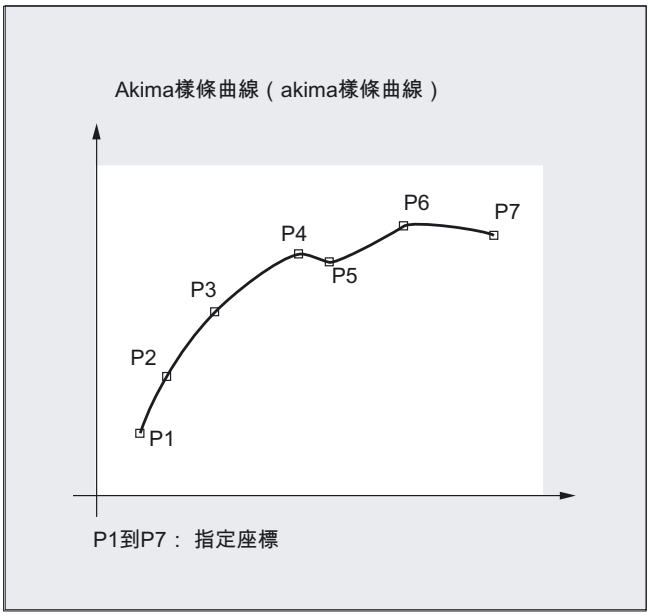
其它資訊

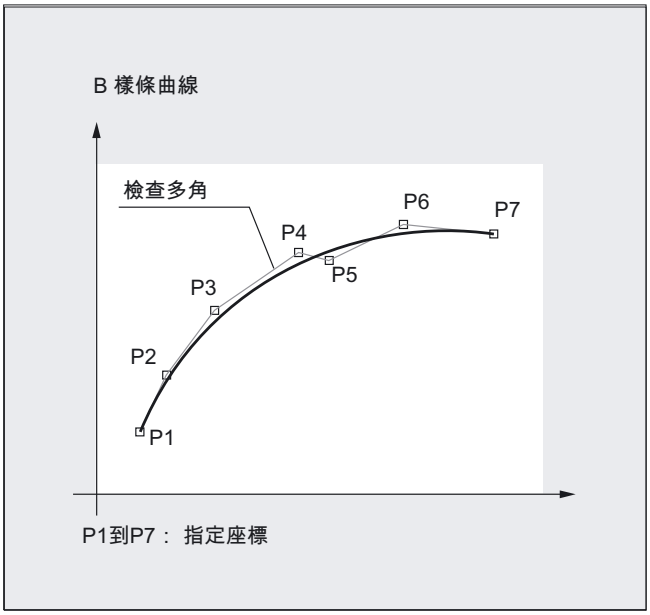
曲線插補的優點

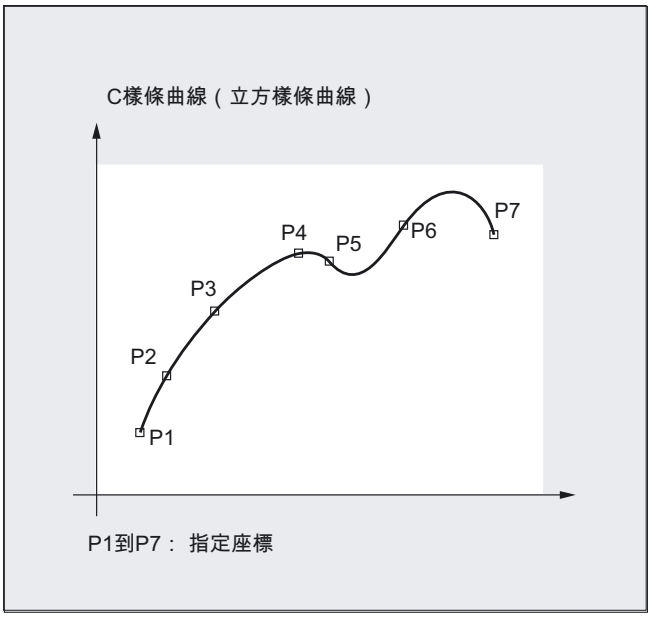
及使用直線單節 G01 相反，使用曲線插補含下列優點：

- 減少需用來定義輪廓的工件程式單節數量
- 於工件程式單節間轉換時，能減少機械系統壓力的平緩、曲線特性。

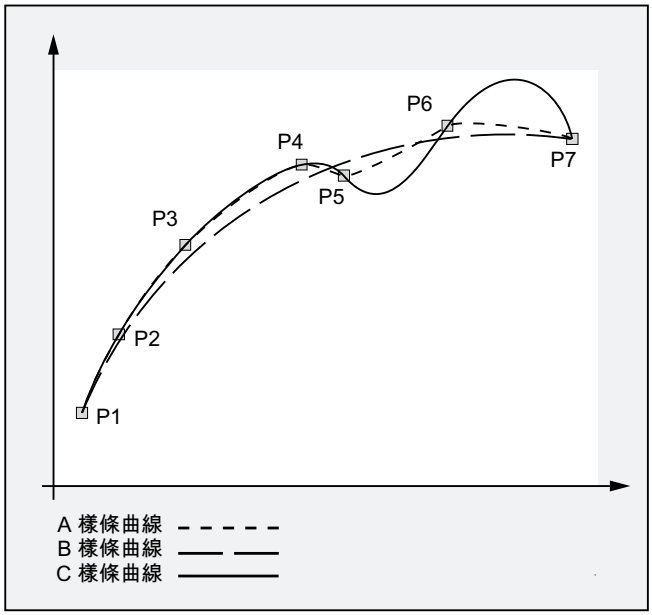
多種曲線的特性及使用

曲線類型	特性及使用
A 曲線	 **特性：** • 精確地沿曲線穿越指定的中間點。 • 曲線特性為切線，但無連續曲率。 • 幾乎無產生任何非預期振盪。 • 沿曲線中間點的變更僅影響本機。表示沿曲線中間點的變更，最多僅能影響 6 個相鄰的中間點。 **應用：** A 曲線特別適合插補含大變化梯度的曲線（例如梯形曲線及特性）。

曲線類型	特性及使用
B 曲線	 **特性：** • 請勿通過指定的沿曲線中間點，僅可靠近。不吸引曲線的中間點。透過使用係數增加中間點的重量，可另外影響曲線特性。 • 曲線特性為含連續曲率的切線。 • 不會產生任何非預期振盪。 • 沿曲線中間點的變更僅影響本機。表示沿曲線中間點的變更，最多僅影響 6 個相鄰的中間點。 **應用：** 其 B 曲線主要作為與 CAD 系統間的介面。

曲線類型	特性及使用
C 曲線	 **特性：** • 精確地沿曲線穿越指定的中間點。 • 曲線特性為含連續曲率的切線。 • 特別在梯度變化的位置，會經常產生非預期振盪。 • 中間點的變更影響為全域。表示若變更中間點，則將影響全部的曲線特性。 **應用：** 若中間點位於曲線上，且可透過分析計算來定義（圓弧、拋物線、雙曲線），則可善用 C 曲線。

以相同的插補點比較三種曲線類型



曲線單節的最少數量

G 代碼 ASPLINE、BSPLINE 及 CSPLINE 使用曲線連接單節結尾。為此，需同時計算一系列的單節（終點）。以 10 個單節為計算的緩衝大小標準。並非每個單節資訊皆為曲線終點。然而，每 10 個單節，控制器便需某些數量的曲線終點單節：

曲線類型	曲線單節的最少數量
A 曲線：	每 10 個單節中，至少需有 **4** 個單節為曲線單節。 不包含註解單節或參數計算。
B 曲線：	每 10 個單節中，至少需有 **6** 個單節為曲線單節。 不包含註解單節或參數計算。
C 曲線：	所需最小曲線單節數量為下列總和的結果： MD20160 $MC_CUBIC_SPLINE_BLOCKS + 1 的值 在 MD20160 輸入用來計算曲線區段的點數量。預設設定為 8。依照標準，每 10 個單節中至少需有 **9** 單節為曲線單節。

說明

若容許值為下衝，且當涉入於曲線中的其中一座標軸，經程式設計為定位軸時，將輸出警報。

結合短曲線單節

曲線插補可產生短曲線單節，無需減少路徑速率。“結合短曲線單節”函數，可讓您結合單節，因此單節長度足夠，且不會減低路徑速率。

透過通道專屬的機械參數啟用函數：

MD20488 $MC_SPLINE_MODE（曲線插補設定）。

參考資料：
功能手冊，基本功能；連續路徑模式，精確停止，預見控制（B1），
一章：結合短曲線單節

4.3 曲線群組（SPLINEPATH）

功能

使用 SPLINEPATH 指令選擇曲線群組中欲插補之軸。曲線插補群組中最多可包含八個路徑軸。

說明

若未明確程式設計 SPLINEPATH，則以曲線群組般移動通道的前三個軸。

句法

在獨立單節中定義曲線群組：

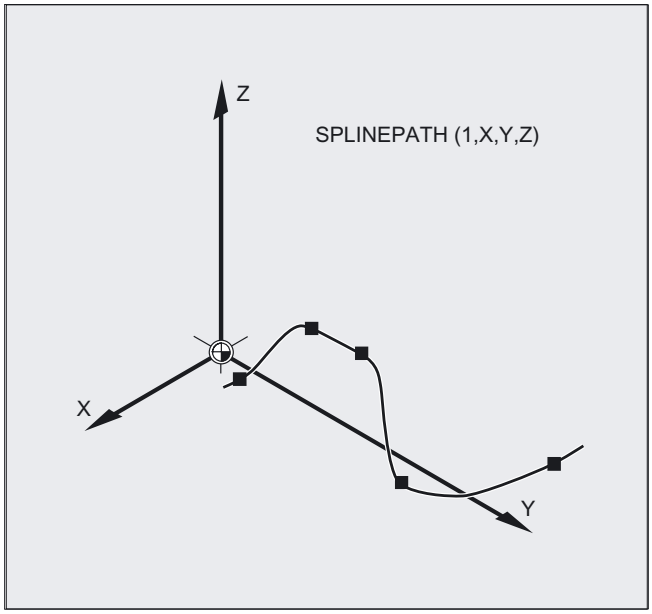
SPLINEPATH（n，X，Y，Z，…）

意義

SPLINEPATH	定義樣條群組的指令
n	=1（固定值）
X，Y，Z，…	曲線群組中欲插補路徑軸的識別碼

範例：含三個路徑軸的曲線群組

程式碼	註解
N10 G1 X10 Y20 Z30 A40 B50 F350	
N11 SPLINEPATH（1，X，Y，Z）	; 曲線群組
N13 CSPLINE BAUTO EAUTO X20 Y30 Z40 A50 B60	; C 曲線
N14 X30 Y40 Z50 A60 B70	; 中間點
…	
N100 G1 X… Y…	; 取消選擇曲線插補



4.4 NC 單節壓縮（COMPON、COMPCURV、COMPCAD、COMPOF）

功能

一般而言，CAD/CAM 系統產生的線性單節皆符合設定的準確規格。在複雜輪廓的情況下，會產生大量的資料及短路徑區段。較短的路徑區段會限制處理的速度。

使用壓縮函數時，使用線性單節指定的輪廓，會由使用多項式單節而逼近。這具有下列優勢：

- 減少敘述工件輪廓所需的工件程式單節數量
- 連續單節變化
- 較高的最大路徑速率

可供使用之壓縮函數如下：

- COMPON

  單節變化僅**速率**恆定，而相關軸的加速度可存在於單節變化的跳躍中。
- COMPCURV

  單節變化具**連續加速度**。可確保所有單節變化上的軸有平穩的速率及加速度。
- COMPCAD

  視表面品質及速度，最佳化使用大量計算時間及記憶體空間的壓縮。僅當由 CAD/CAM 程式無法預先量測表面的改善時，才使用 COMPCAD。

COMPOF 終止壓縮功能。

句法

```
COMPON
COMPCURV
COMPCAD
COMPOF
```

意義

COMPON:	啟動壓縮函數 COMPON 的指令。 有效範圍: 模態
COMPCURV:	啟動壓縮函數 COMPCURV 的指令。 有效範圍: 模態
COMPCAD:	啟動壓縮函數 COMPCAD 的指令。 有效範圍: 模態
COMPOF :	停用目前有效壓縮函數的指令。

說明

可使用倒圓角函數 G642 及震動停止 SOFT 改善表面的品質。需於程式開頭寫入指令。

補充條件

- 僅可在線性單節（G1）上執行 NC 單節壓縮。
- 僅壓縮遵守簡單句法的單節：

  N... G1X... Y... Z... F... ; comment

  不變更執行所有其他單節（無壓縮）。
- 亦壓縮含如 C=100 或 A=AC（100）等延伸位址的動作單節。
- 無需直接程式設計，也可間接使用參數指派指定位置值，例如 X=R1*（R2+R3）。
- 若選項“方向轉換”可用，則在其中使用方向向量程式設計刀具方向（及其相關者，刀具旋轉亦可）的NC單節，亦可壓縮（請參閱“壓縮方向 (頁 313)”）。
- 由任何其他類型的 NC 指令中斷，例如，輔助函數輸出。

範例

範例 1：COMPON

程式碼	註解
N10 COMPON	; 開啟壓縮函數 COMPON。
N11 G1 X0.37 Y2.9 F600	; 終點及進給前的 G1。
N12 X16.87 Y–.698	
N13 X16.865 Y–.72	
N14 X16.91 Y–.799	
...	
N1037 COMPOF	; 關閉壓縮函數。
...	

範例 2：COMPCAD

程式碼	註解
G00 X30 Y6 Z40	
G1 F10000 G642	; 開啟混合函數 G642。
SOFT	; 開啟震動停止 SOFT。
COMPCAD	; 開啟壓縮函數 COMPCAD。
STOPFIFO	
N24050 Z32.499	
N24051 X41.365 Z32.500	
N24052 X43.115 Z32.497	
N24053 X43.365 Z32.477	
N24054 X43.556 Z32.449	
N24055 X43.818 Z32.387	
N24056 X44.076 Z32.300	
...	
COMPOF	; 關閉壓縮函數。
G00 Z50	
M30	

參考資料

功能手冊，基本功能，連續路徑模式，精確停止，預見控制（B1）
一章：“NC 單節壓縮”

4.5 多項式插補（POLY、POLYPATH、PO、PL）

功能

多項式插補（POLY）實際上並非曲線插補。其主要目的，是要當作由外部程式設計已產生的曲線所用之介面，且在該處可直接程式設計曲線區段。

此插補模式減少 NC 計算多項式係數的作業。在直接由 CAD 系統或後處理器供應係數的情況下，可以達到最佳化運用。

句法

3 次多項式：

```
POLY PO[X]=（xe，a2，a3） PO[Y]=（ye，b2，b3） PO[Z]=（ze，c2，c3） PL=n
```

或延伸至 5 次多項式及新多項式句法：

```
POLY X=PO（xe，a2，a3，a4，a5） Y=PO（ye，b2，b3，b4，b5） Z=PO（ze，c2，
c3，c4，c5） PL=n
POLYPATH（"AXES"，"VECT"）
```

意義

`POLY`	使用包含 POLY 的單節啟用多項式插補。
`POLYPATH`	可為 AXIS 或 VECT 軸群組選擇多項式插補
`PO[axis identifier/variable]`	終點及多項式係數
`X，Y，Z`	軸定義碼
`xe，ye，ze`	特定軸的結束位置規格；值域與路徑尺寸相同
`a2，a3，a4，a5`	係數 a_2、a_3、a_4、及 a_5 以其值寫入；值域與路徑尺寸相同。若係數等於 0，則可忽略各情況下的最後係數。
`PL`	定義多項式的參數間隔長度（函數 f（p）定義範圍）。 間隔永遠從 0 開始，p 可假設值為 0 到 PL。 PL 的理論值域： 0，0001 … 99 999，9999 **注意：** PL 值將套用至其所在之單節。若未程式設計 PL，則套用 PL=1。

啟動 / 停用 POLY

多項式插補屬於位於第一個 G 群組中的 G0、G1、G2、G3、A 曲線、B 曲線及 C 曲線。若此為有效，則無需程式設計多項式句法：將僅以其名稱及終點程式設計之軸，線性移動至其終點。若皆以此方式程式設計所有軸，則控煞車作與 G1 相同。

使用 G 群組中的另一個指令（例如，G0、G1）停用多項式插補。

多項式係數

PO 值（PO[]=）或 ...=PO（...）為一個軸指定所有多項式係數。以相應多項式次方的逗號指定、分隔數個值。一個單節中的各軸可有不同的多項式次方。

使用 PO 的新增多項式句法：先前的句法仍有效。

副程式呼叫 POLYPATH

可使用 POLYPATH 為下列軸群組選擇指定多項式插補：

- POLYPATH（"AXES"）
  所有路徑軸及輔助軸。
- POLYPATH（"VECT"）
  方向轉換（用於方向轉換）。

已程式設計的多項式，也經插補為兩軸群組的多項式標準。

範例：

POLYPATH（"VECT"）

僅方向軸經選為多項式插補用－其他所有軸皆線性移動。

POLYPATH （ ）

停用所有軸的多項式插補

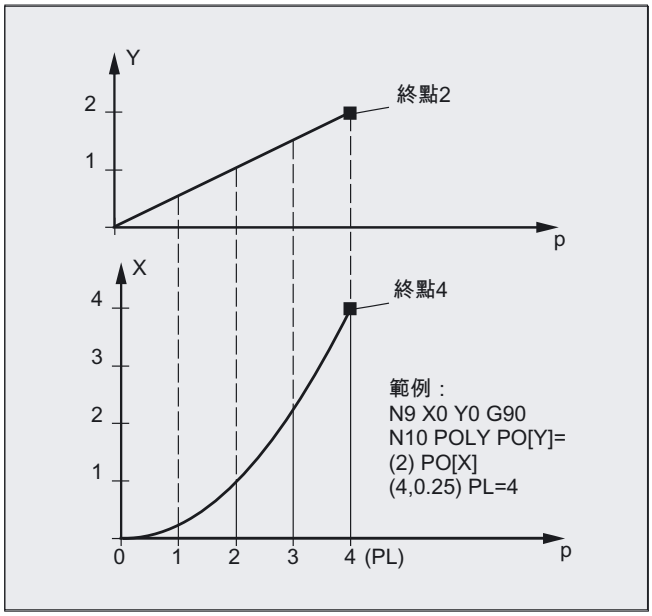
範例

程式碼	註解
N10 G1 X… Y… Z… F600	
N11 POLY PO[X]=（1，2.5，0.7） PO[Y]=（0.3，1，3.2） PL=1.5	; 開啟多項式插補
N12 PO[X]=（0，2.5，1.7） PO[Y]=（2.3，1.7） PL=3	
...	
N20 M8 H126 …	
N25 X70 PO[Y]=（9.3，1，7.67） PL=5	; 軸的混合資料
N27 PO[X]=（10，2.5） PO[Y]=（2.3）	; 未程式設計 PL，套用 PL=1
N30 G1 X… Y… Z.	; 關閉多項式插補
…	

範例：可套用含 PO 的多項式句法

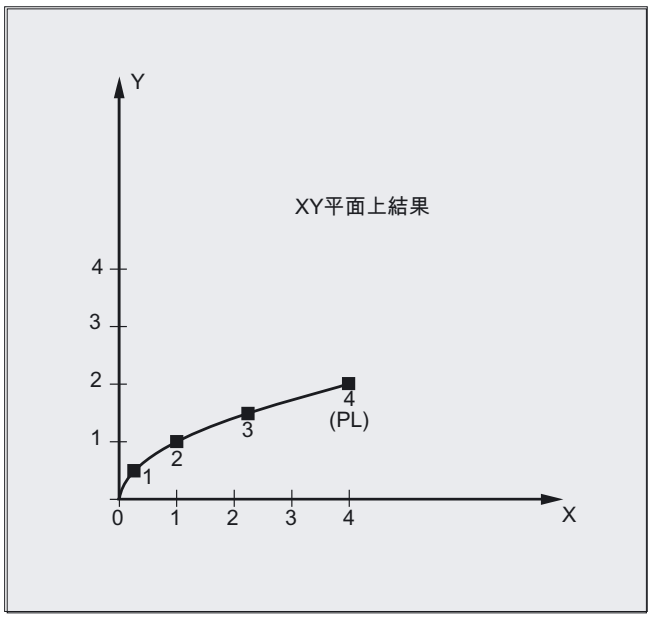
前一個多項式句法仍有效	（從 SW 6）新增多項式句法
PO[axis identifier]=（.. , ..）	Axis identifier=PO（.. , ..）
PO[PHI]=（.. , ..）	PHI=PO（.. , ..）
PO[PSI]=（.. , ..）	PSI=PO（.. , ..）
PO[THT]=（.. , ..）	THT=PO（.. , ..）
PO[]=（.. , ..）	PO（.. , ..）
PO[variable]=IC（.. , ..）	variable=PO IC（.. , ..）

範例：X/Y 平面上曲線。



程式碼

```
N9 X0 Y0 G90 F100
N10 POLY PO[Y]=（2） PO[X]=（4，0.25） PL=4
```



說明

控制系統能移動曲線（路徑），其中所選的每個路徑軸，可如最高達五次的多項式函數般操作。

用來表示多項式函數的方程式通常如下：

$f(p) = a_0 + a_1p + a_2p^2 + a_3p^3$

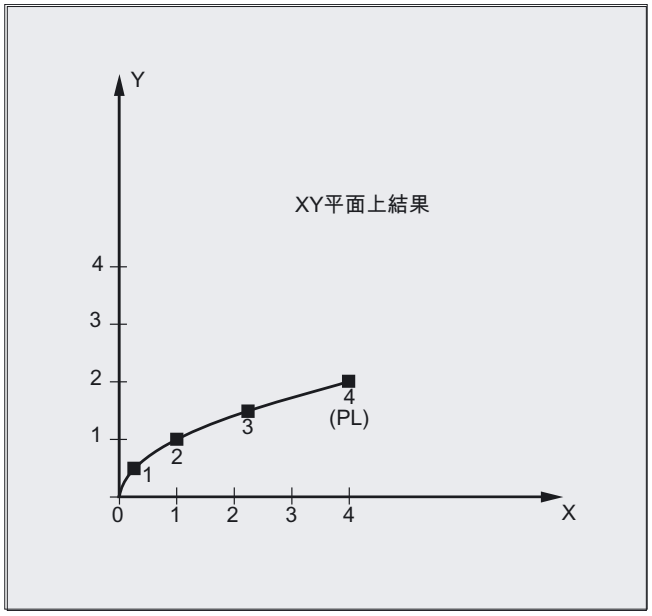
或

$f(p) = a_0 + a_1p + a_2p^2 + a_3p^3 + a_4p^4 + a_5p^5$

關鍵字：

a_n：常數係數

p：參數



指派具體值至該係數，可產生多樣的曲線形狀，如直線、拋物線及冪函數。

例如，設定係數 $a_2 = a_3 = 0$ 或 $a_2 = a_3 = a_4 = a_5 = 0$ 收益率，則一條直線使用：

$f(p) = a_0 + a_1p$

下述應用：

a_0 = 於前面單節結尾的軸位置

p = PL

$a_1 = (x_E - a_0 - a_2*p^2 - a_3*p^3)/p$

無需有效的 POLYG 代碼，即可程式設計多項式。然而，在此情況下，不插補已程式設計的多項式，而以線性逼進各軸分別已程式設計的終點（G1）。則可透過程式設計 POLY 啟用多項式插補。

並且，若 G 代碼 POLY 有效，且含預先定義副程式 POLYPATH（...），則您可選擇以多項式插補之軸。

分母多項式的特殊特徵

可使用 PO[ ]=（…）指令為幾何軸（無需指定軸名稱）程式設計一個普通的分母多項式，即將幾何軸的動作插補為兩個多項式的商數。

使用此程式設計選項，可精確呈現如圓錐（圓弧、橢圓、拋物線、雙曲線）的形狀。

範例

程式碼	註解
POLY G90 X10 Y0 F100	; 幾何軸線性移動至位置 X10 Y0。
PO[X]=（0，—） PO[Y]=（10） PO[]=（2，1）	; 幾何軸沿象限移動至 X0 Y10。

永遠假定分母多項式的常數係數（a_0）為 1，指定的終點與 G90 / G91 無關。

上述範例所含的結果如下：

X（p） = 10（1） /（1+p^2）及

Y（p） = 20p /（1+p^2）

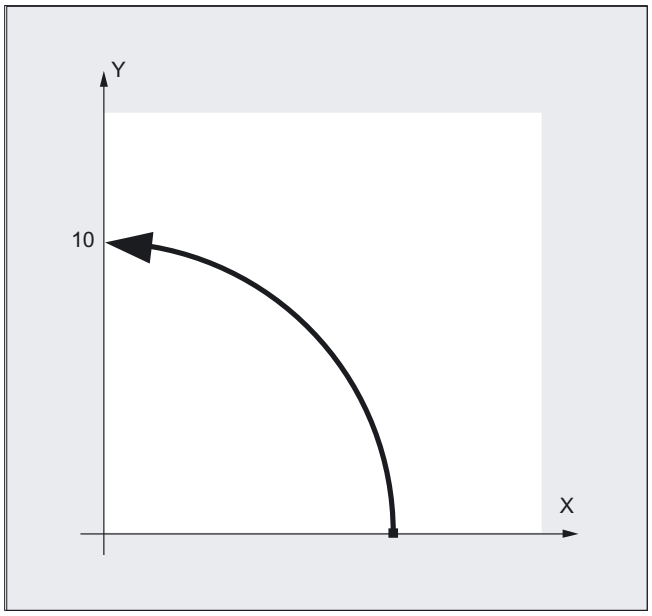
且 0 ≤ p ≤ 1

作為已程式設計的起點、終點，係數 a_2 及 PL=1 的結果，其中間值如下：

分子（X）= 10+0*p-p^2

分子（Y） = 0+20*p+0*p^2

分母 = 1+2*p+1*p^2



若多項式插補有效，且已在間隔 [0，PL] 內使用零點程式設計分母多項式，則拒絕並輸出警報。分母多項式不會影響特殊軸的動作。

說明

可使用 G41、G42 啟動刀具半徑補正，搭配多項式插補，並可與在線性或圓弧插補模式以相同的方式套用。

4.6 可設定路徑參考（SPATH、UPATH）

功能

多項式插補期間，使用者於速率間可能需要兩個不同的關係，以決定 FGROUP 軸及其他路徑軸：後者應受控制、同步至 FGROUP 軸路徑，或與曲線參數同步。

因此，對於不包含在 FGROUP 的軸，有兩個跟隨路徑的方式：

- 與路徑 S（SPATH）同步。

  或

- 與 FGROUP 軸的曲線參數 U 同步（UPATH）

兩種路徑插補類型在不同的應用程式中使用，且可透過 G 代碼 SPATH 及 UPATH 將其變更。

句法

```
SPATH
UPATH
```

意義

SPATH	FGROUP 軸的路徑參考為圓弧長
UPATH	FGROUP 軸的路徑參考為曲線參數
FGROUP	含路徑進給的定義軸

SPATH、UPATH

可使用 G 代碼之一（SPATH、UPATH）選擇及程式設計所需行為。

該指令為模態的。若 SPATH 有效，則軸與路徑同步移動；若 UPATH 有效，則與曲線參數同步移動。

UPATH 及 SPATH 也使用路徑動作定義 F 字組多項式（FPOLY、FCUB、FLIN）的相互關係。

FGROUP 啟用

不包含在 FGROUP 中的軸路徑參考，是透過包含在第 45 組 G 代碼群組中的 SPATH 及 UPATH 兩個語言指令設定。

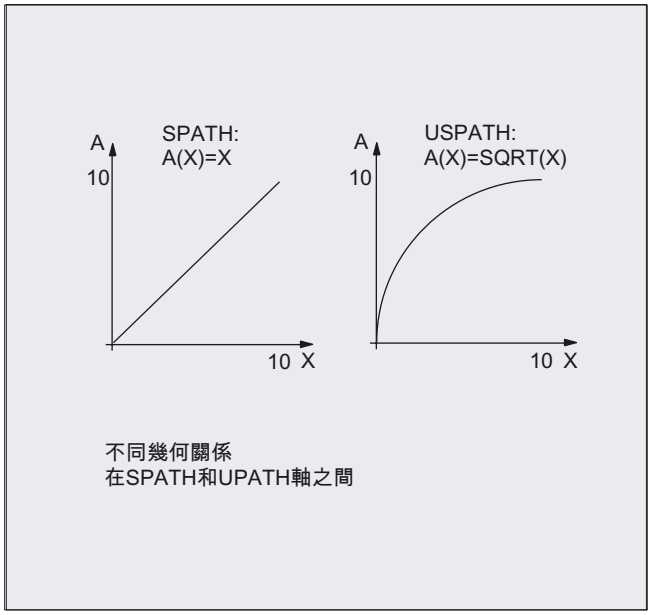
範例 1

下列範例顯示，以 G643 磨圓邊長 20 毫米正方形的角。為各軸精確輪廓的最大偏差，是使用機械參數 MD33100 $MA_COMPRESS_POS_TOL[...]來定義。

程式碼	註解
N10 G1 X… Y… Z… F500	
N20 G643	; 使用 G643 倒圓角單節內部的角
N30 XO Y0	
N40 X20 Y0	; 軸的毫米稜邊長度
N50 X20 Y20	
N60 X0 Y20	
N70 X0 Y0	
N100 M30	

範例 2

下列範例顯示不同類型動作控制間的差異。預設設定 FGROUP（X，Y，Z）在兩種情況下皆有效。



程式設計
N10 G1 X0 A0 F1000 SPATH
N20 POLY PO[X]=（10，10） A10

或：

程式設計

```
N10 G1 X0 F1000 UPATH
N20 POLY PO[X]=（10，10） A10
```

於單節 N20 中，FGROUP 軸的路徑 S 需仰賴曲線參數 U 的正方形。因此，沿著路徑 X 的同步軸 A 形成的不同位置，根據 SPATH 或 UPATH 而有效。

限制

路徑參考於下列情況中不重要

- 線性及圓弧插補，
- 於螺紋單節及
- 若 FGROUP 中包含所有路徑軸。

說明

於多項式插補期間－因此多項式插補在下列情況中永遠合理

- 狹義的情況下（POLY），
- 所有曲線插補的型號（ASPLINE、BSPLINE、CSPLINE）及
- 含壓縮機的線性插補（COMPON、COMPCURV）

－由多項式 pi（U）指定的所有路徑軸位置。曲線參數 U 於 NC 單節內從 0 移動至 1，因此它已標準化。

可透過語言指令 FGROUP 從路徑軸中，選取與已程式設計的路徑進給相關的軸。然而，多項式插補期間，於該軸的路徑 S 上使用恆定速率的插補，通常代表曲線參數 U 的非恆定變更。

重置及機械參數 / 選項資料控制行為

重置後，MD 20150：GCODE_RESET_VALUES [44]使某些 G 代碼有效（第 45 組 G 代碼群組）。

可使用 MD 20150 指定平滑化類型初始狀態：GCODE_RESET_VALUES [9]（第 10 組 G 代碼群組）。

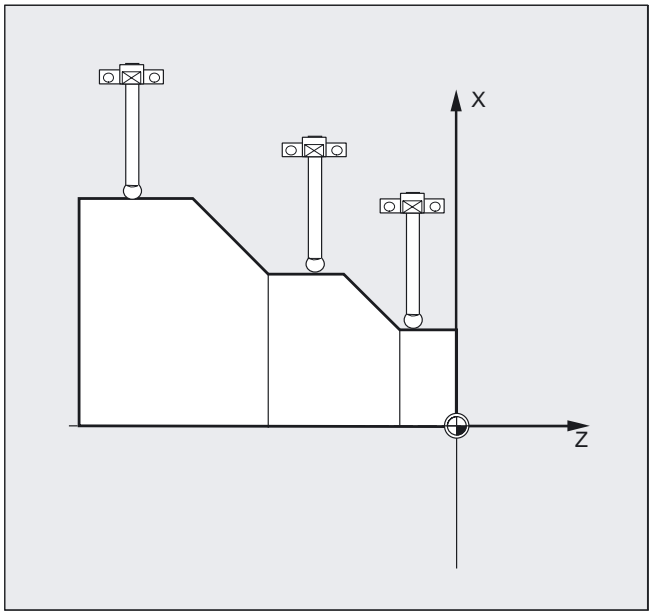
由機械參數 MD 20150 決定重置後有效的 G 代碼群組值：GCODE_RESET_VALUES [44].為維持現有安裝的相容性，設定 SPATH 為預設值。

軸機械參數 MD 33100：COMPRESS_POS_TOL 有延伸的意義：包含壓縮函數及使用 G642 倒圓角函數之允差。

4.7 使用觸發式探針量測（MEAS、MEAW）

功能

"使用觸發式探針量測"可用於在逼近工件上的實際位置。在探針的切換稜邊，會測量所有程式設計在量測單節中的軸之位置並寫入至各個軸適當的記憶體中。



程式設計量測單節

以下兩個指令可用於程式設計量測單節：

- MEAS
  MEAS 指令可用於刪除實際與設定點位置間的剩餘距離。
- MEAW
  MEAW 指令可用在程式設計的位置永遠需要逼近的量測作業中。

MEAS 與 MEAW 為非模態；其會與動作操作一同程式設計。針對目前的量測作業需選擇適當的進給率與插補類型（G0、G1、等）以及軸數量。

讀取量測結果

可於下列變數中取得透過探針而得的軸測量值：

- $AA_MM[<軸>]
  使用機械座標系統的量測結果
- $AA_MW[<軸>]
  使用工件座標系統的量測結果

讀取這些變數時，不會產生內部前置處理停止。

說明

需於程式中適當的位置使用 STOPRE 程式設計前置處理停止。否則系統會讀取錯誤的值。

句法

```
MEAS=<TE> G... X... Y... Z...
MEAW=<TE> G... X... Y... Z...
```

意義

MEAS	指令：量測時**含**刪除的剩餘距離 有效範圍： 非模態
MEAW	指令：量測時**不含**刪除的剩餘距離 有效範圍： 非模態
<TE>	觸發開始量測的事件 類型： INT 值域： -2, -1, 1, 2 **注意：** 最多可有 2 個探針（視設定層級而定）。 意義： (+)1 探針 1 的上升稜邊（量測輸入 1） -1 探針 1 的下降稜邊（量測輸入 1） (+)2 探針 2 的上升稜邊（量測輸入 2） -2 探針 2 的下降稜邊（量測輸入 2） **注意：** 最多可有 2 個探針（視設定層級而定）。
G...	插補類型，例如 G0、G1、G2 或 G3
X... Y... Z...	直角座標中的終點

範例

程式碼	註解
N10 MEAS=1 G1 F1000 X100 Y730 Z40	; 使用探針於第一個量測輸入及線性插補量測單節。自動產生前置處理停止。
...	

其它資訊

量測工作狀態

若程式需評估以得知是否已觸發探針，可查詢狀態變數$AC_MEA[n] （n=量測探針數量）：

值	意義
0	量測作業未完成。
1	量測作業已完成（已觸發探針）

說明

若在程式中偏移程式，則變數設為 1。於量測單節啟始處，變數自動設為探針的初始狀態。

讀取量測值

取得單節中所有移動路徑及定位軸之位置（最大軸數量視控制設定而定）。在使用 MEAS 時，該動作會以定義的趨勢在觸發探針後減速。

說明

若在量測單節中程式設計幾何軸，則貯存該量測值用於所有目前幾何軸。
若在量測單節中程式設計轉換中的軸，則會記錄所有轉換中的軸量測值。

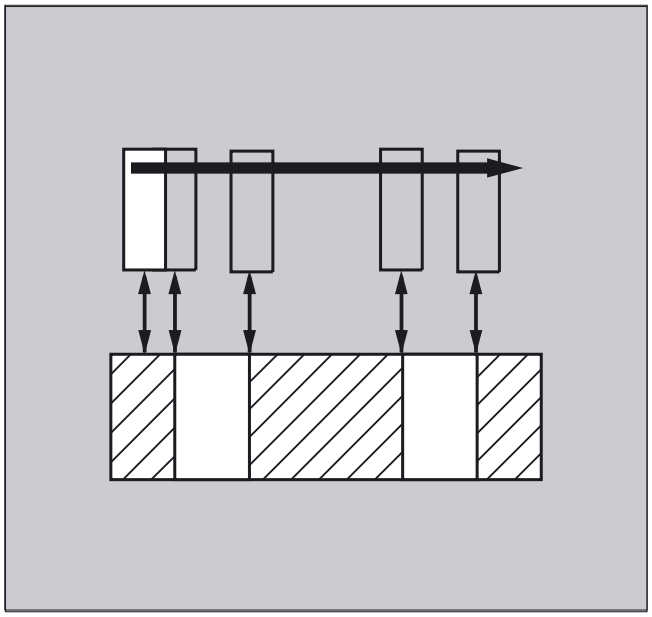
4.8 延伸量測函數（MEASA、MEAWA、MEAC）（選用）

功能

可使用數個探針及量測系統於軸量測。

MEASA 或 MEAWA 指令可用於取得最多四個與程式設計的軸對應之量測值；接著這些值會依據觸發事件儲存在系統變數中。

連續量測作業可使用 MEAC 指令執行。在此情況下，量測結果會儲存於 FIFO 變數中。使用 MEAC 每次量測的量測值最大數量亦為 4。



讀取量測結果

量測結果可存在以下變數中：

- $AA_MM1...4[<軸>]
  使用機械座標系統的量測結果
- $AA_MW1...4[<軸>]
  使用工件座標系統的量測結果

句法

```
MEASA[<軸>]=（<模式>、<TE1>、...、<TE4>）
MEAWA[<軸>]=（<模式>、<TE1>、...、<TE4>）
MEAC[<軸>]=（<模式>、<量測記憶體>、<TE1>、...、<TE4>）
```

說明

MEASA 與 MEAWA 為非模態，其可一同程式設計在一個單節中。然而，若於同一單節中與 MEAS/MEAW 一同程式設計 MEASA/MEAWA，將輸出錯誤訊息。

意義

MEASA	指令：含刪除之剩餘距離的軸量測 有效範圍： 非模態
MEAWA	指令：不含刪除之剩餘距離的軸量測 有效範圍： 非模態
MEAC	指令：持續量測軸，不含刪除之剩餘距離 有效範圍： 非模態
<軸>	用於量測的通道軸名稱
<模式>	二位數字顯示操作模式（量測模式與量測系統） **量測模式**（十位數）： 0 取消量測作業。 1 最多可同時啟動 4 個不同的觸發事件。 2 最多可連續啟動 4 個觸發事件。 3 最多可連續啟動 4 個觸發事件，但在 START 不監控觸發事件 1（停用警報 21700/21703）。 **注意：** 此模式 MEAC 並不支援。 **量測系統**（十進位）： 0（或無資料） 主動量測系統 1 量測系統 1 2 量測系統 2 3 兩個量測系統
<TE>	觸發開始量測的事件 類型： INT 值域： -2, -1, 1, 2 意義： (+)1 探針 1 上升稜邊 -1 探針 1 下降稜邊 (+)2 探針 2 的上升稜邊 -2 探針 2 的下降稜邊
<量測記憶體>	FIFO 數量（循環儲存）

範例

範例 1：具有刪除模式 1 之剩餘距離的軸量測（以時間順序評估）

a） 使用 1 量測系統

程式碼	註解
...	
N100 MEASA[X]=（1，1，-1） G01 X100 F100	; 於模式 1 使用主動量測系統量測。等候使用上升 / 下降稜邊的量測訊號，以從探針 1 移動路徑至 X=100。
N110 STOPRE	; 前置處理停止
N120 IF $AC_MEA[1]==FALSE GOTOF END	; 檢查量測是否成功。
N130 R10=$AA_MM1[X]	; 儲存於第一個已程式設計的觸發事件（上升稜邊）取得的量測值。
N140 R11=$AA_MM2[X]	; 儲存於第二個已程式設計的觸發事件（下降稜邊）取得的量測值。
N150 END:	

b） 使用 2 量測系統

程式碼	註解
...	
N200 MEASA[X]=（31，1，-1） G01 X100 F100	; 於模式 1 使用兩個量測系統量測。等候上升 / 下降稜邊的量測訊號，以從探針 1 移動路徑至 X=100。
N210 STOPRE	; 前置處理停止
N220 IF $AC_MEA[1]==FALSE GOTOF END	; 檢查量測是否成功。
N230 R10=$AA_MM1[X]	; 儲存上升稜邊上量測系統的量測值。
N240 R11=$AA_MM2[X]	; 儲存上升稜邊上量測系統 2 的量測值。
N250 R12=$AA_MM3[X]	; 儲存下降稜邊上量測系統 1 的量測值。
N260 R13=$AA_MM4[X]	; 儲存下降稜邊上量測系統 2 的量測值。
N270 END:	

範例 2：具有刪除模式 2 之剩餘距離的軸量測（以程式設計的順序評估）

程式碼	註解
...	
N100 MEASA[X]=（2，1，-1，2，-2） G01 X100 F100	; 於模式 2 使用主動量測系統量測。當移動路徑至 X=100 時，以上升稜邊探針 1、下降稜邊探針 1、上升稜邊探針 2、下降稜邊探針 2 的順序等候量測訊號。
N110 STOPRE	; 前置處理停止
N120 IF $AC_MEA[1]==FALSE GOTOF PROBE2	; 檢查是否成功使用探針 1 量測。
N130 R10=$AA_MM1[X]	; 儲存第一個已程式設計的觸發事件上（上升稜邊，探針 1）取得的量測值。
N140 R11=$AA_MM2[X]	; 儲存第二個已程式設計的觸發事件上（上升稜邊，探針 1）取得的量測值。
N150 PROBE2:	
N160 IF $AC_MEA[2]==FALSE GOTOF END	; 檢查是否成功使用探針 2 量測。
N170 R12=$AA_MM3[X]	; 儲存第三個已程式設計的觸發事件上（上升稜邊，探針 2）取得的量測值。
N180 R13=$AA_MM4[X]	; 儲存第四個已程式設計的觸發事件上（上升稜邊，探針 2）取得量測值。
N190 END:	

範例 3：使用模式 1 的連續軸量測（以時間順序評估）

a） 最多 100 個量測值的量測

程式碼	註解
...	
N110 DEF REAL MEASVALUE[100]	
N120 DEF INT loop=0	
N130 MEAC[X]=（1，1，-1） G01 X1000 F100	; 在模式中使用主動量測系統量測，將量測值儲存於$AC_FIFO1，在移動路徑至 X=1000 時，等待含探針 1 之下降稜邊的量測訊號。
N135 STOPRE	
N140 MEAC[X]=（0）	; 達到軸位置時終止量測。
N150 R1=$AC_FIFO1[4]	; 於參數 R1 中儲存累計的量測值。
N160 FOR loop=0 TO R1-1	
N170 MEASURED VALUE[loop]=$AC_FIFO1[0]	; 從$AC_FIFO1 讀出並儲存量測值。
N180 ENDFOR	

b） 在 10 個量測值後，量測含刪除的剩餘距離

程式碼	註解
...	
N10 WHEN $AC_FIFO1[4]>=10 DO MEAC[x]=（0） DELDTG（x）	; 刪除剩餘距離。
N20 MEAC[x]=（1，1，1，-1） G01 X100 F500	
N30 MEAC [X]=（0）	
N40 R1 = $AC_FIFO1[4]	; 量測值數量。
...	

其他資訊

量測工作

可於工件程式中或從同步動作中程式設計量測作業（請參閱名為“動態同步動作”一節）。請注意在任何時間下各軸每次僅啟用一個量測工作。

說明

需調整進給以符合目前的量測作業。

在 MEASA 與 MEAWA 的情況下，僅可確保在沒有超過 1 個相同類型的觸發事件且在每個位置控制器詢環中沒有發生超過 4 種不同類型的觸發事件時的進給率結果正確。

在使用 MEAC 連續量測的情況下，插補循環及位置控制循環間的比例不可超過 8：1。

觸發事件

觸發事件包含量測訊號的探針及觸發條件（上升或下降稜邊）的數量。

每次量測最多可處理定址探針中 4 個觸發事件；換言之，即 2 個探針最多各含 2 個量測訊號稜邊。處理順序及觸發事件最大數量視選擇模式而定。

說明

以下資料套用至量測模式 1：在一個量測作業中，相同的觸發事件僅能程式設計一次。

操作模式

操作模式的第一位（十進位）可選擇需要的量測系統。若僅安裝一個量測系統，但已程式設計第二個，則會自動選擇安裝的系統。

第二位（十位數）可選擇需要的量測模式。量測程序可藉此針對相關控制系統所支援的選項做調整：

- **模式 1**

  依發生的時間順序評估觸發事件。選擇此模式時，僅可為六軸模組程式設計一個觸發事件。若已指定一個以上的觸發事件，則會自動切換模式選擇至模式 2（無訊息）。

- **模式 2**

  依已程式設計的順序評估觸發事件。

- **模式 3**

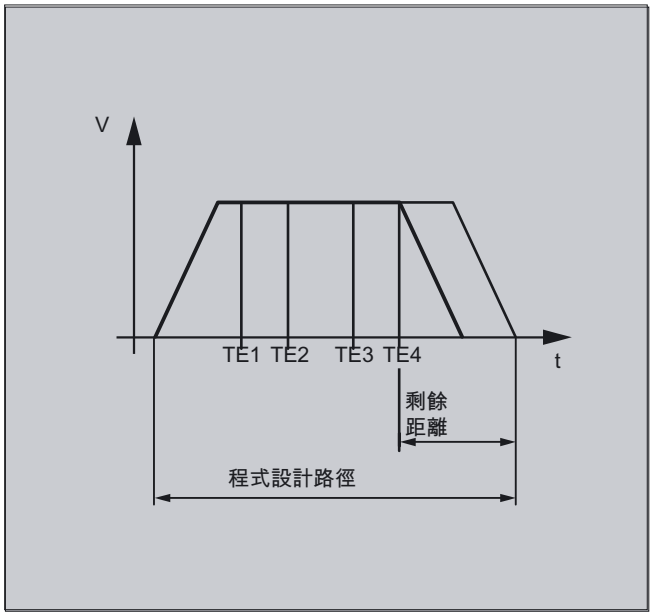
  依已程式設計的順序評估觸發事件，但不包括 START 監控觸發事件 1。

  說明

  若正在使用量測系統，則無法程式設計兩個以上的觸發事件。

量測時含與不含刪除的剩餘距離

程式設計 MEASA 指令時，在記錄所有所需量測的值之前，不會刪除剩餘距離。
在需逼近特殊量測作業的已程式設計位置時，可執行 MEAWA 函數。



說明

同步動作中無法程式設計 MEASA。此外，可程式設計 MEAWA 加上刪除剩餘距離為同步動作。
若已從同步動作中啟動含 MEAWA 的量測工作，則量測值僅在機械座標系統中可用。

MEASA、MEAWA 的量測結果

量測結果於下列系統變數中可用：

- 在機械座標系統中：

$AA_MM1[<軸>]	觸發事件 1 上已程式設計的量測系統量測值
...	...
$AA_MM4[<軸>]	觸發事件 4 上已程式設計的量測系統量測值

- 在工件座標系統中：

$AA_WM1[<軸>]	觸發事件 1 上已程式設計的量測系統量測值
...	...
$AA_WM4[<軸>]	觸發事件 4 上已程式設計的量測系統量測值

說明

讀取這些變數時，不會產生內部前置處理停止。需於程式中適當的位置使用 STOPRE 程式設計前置處理停止。否則會讀入錯誤的值。

幾何軸/轉換

若欲為幾何軸啟動軸量測，需明確地為所有剩餘幾何軸程式設計相同的量測工作。同樣套用至涉入轉換的軸。

範例：

```
N10 MEASA[Z]=（1，1） MEASA[Y]=（1，1） MEASA[X]=（1，1） G0 Z100
```

或

```
N10 MEASA[Z]=（1，1） POS[Z]=100
```

使用兩個量測系統的量測工作

若由兩個量測系統執行量測工作，則會取得兩個相關軸量測系統中，兩個可能觸發事件之一。因此預設保留變數的指派為：

$AA_MM1[<軸>]	或	$AA_MW1[<軸>]	觸發事件 1 量測系統 1 之量測值
$AA_MM2[<軸>]	或	$AA_MW2[<軸>]	觸發事件 1 量測系統 2 之量測值
$AA_MM3[<軸>]	或	$AA_MW3[<軸>]	觸發事件 2 量測系統 1 之量測值
$AA_MM4[<軸>]	或	$AA_MW4[<軸>]	觸發事件 2 量測系統 2 之量測值

探針狀態

探針狀態可存在下列系統變數中：

$A_PROBE[<n>]

<n>=探針

值	意義
1	探針偏移
0	探針未偏移

MEASA、MEAWA 量測工作狀態

若程式中需要評估，則可透過$AC_MEA[n]確認量測作業狀態，其中<n> =探針編號。一旦發生單節中程式設計探針<n>的所有觸發事件，則此變數會傳回“1”值。否則，其值為 0。

說明

若從同步動作開始量測，則不會再更新$AC_MEA。在此情況下，需確認新的 PLC 狀態訊號 DB31、...DBB62 位元 3，或相等變數$AA_MEAACT[<軸>]。

含義：

$AA_MEAACT==1：量測啟用

$AA_MEAACT==0：量測未啟用

連續量測（MEAC）

MEAC 量測值在機械座標系統中可用，並儲存在已程式設計的 FIFO[m]記憶體（循環緩衝器）中。若設定兩個用於量測的探針，則第二個探針的量測值，會分別儲存於特別為此而設的 FIFO[n+1]記憶體中（在機械參數中定義）。

FIFO 記憶體為一個循環緩衝器，根據循環原則寫入其量測值至$AC_FIFO 變數中，請參閱"動作同步動作"一章。

說明

僅可從循環儲存中讀取 FIFO 內容一次。若將重複使用該量測資料，則需在使用者資料中緩衝該量測資料。

若 FIFO 記憶體量測值的數量，超過機械參數中定義的最大值，則自動終止量測。

可循環讀出量測值以進行無限量測流程。在此狀況下，讀出資料的頻率需與讀入新量測值的頻率相同。

辨識出的程式設計錯誤

適當偵測並指出下列程式設計錯誤：

- MEASA/MEAWA 與 MEAS/MEAW 程式設計在相同單節中

  範例：

  ```
  N01 MEAS=1 MEASA[X]=（1，1） G01 F100 POS[X]=100
  ```

- MEASA/MEAWA 具有<2 或>5 的數個參數

  範例：

  ```
  N01 MEAWA[X]=（1） G01 F100 POS[X]=100
  ```

- MEASA/MEAWA 觸發事件不等於 1/ -1/ 2/ -2。

  範例：

  ```
  N01 MEASA[B]=（1，1，3） B100
  ```

- 具有無效模式的 MEASA/MEAWA

  範例：

  ```
  N01 MEAWA[B]=（4，1） B100
  ```

- 具有程式設計了兩次觸發事件的 MEASA/MEAWA

  範例：

  ```
  N01 MEASA[B]=（1，1，-1，2，-1） B100
  ```

- MEASA/MEAWA 及遺失幾何軸

  範例：

  ```
  N01 MEASA[X]=（1，1） MESA[Y]=（1，1） G01 X50 Y50 Z50 F100；幾何軸 X/Y/Z
  ```

- 量測作業與幾何軸不一致

  範例：

  ```
  N01 MEASA[X]=（1，1） MEASA[Y]=（1，1） MEASA[Z]=（1，1，2） G01 X50 Y50 Z50 F100
  ```

4.9 OEM 使用者的特殊函數（OEMIPO1、OEMIPO2、G810 至 G829）

功能

OEM 位址

由 OEM 使用者決定 OEM 位址的意義。透過編譯循環結合其函數性。已保留五個 OEM 位址。可設定位址識別碼。可於任何單節中程式設計 OEM 位址。

參數

保留的 G 群組

群組 1 含 OEMIPO1、OEMIPO2

OEM 使用者可定義另外兩個 G 函數 OEMIPO1、OEMIPO2 的名稱。透過編譯循環整合並保留其函數性給 OEM 使用者。

- 群組 31 含 **G810 至 G819**
- 群組 32 含 **G820 至 G829**

各保留兩個含 10 個 OEM G 系列功能的 G 群組，給 OEM 使用者。**可讓外部應用程式存取 OEM 使用者整合的函數。**

函數及子程式

OEM 使用者亦可使用參數傳輸，設定預先定義的函數及子程式。

4.10 含轉角減速的進給率減慢功能（FENDNORM，G62、G621）

功能

使用自動轉角減速時，將在到達轉角前依據鐘形曲線減慢進給率。也可透過設定資料，參數化與加工相關的刀具行為範圍。分別為：

- 進給率減慢功能的開始與結束
- 進給率減慢時所遵照的手動倍率內容
- 偵測相關轉角

相關轉角的內角小於設定資料中參數化的轉角。

預設值 FENDNORM 會停用自動轉角手動倍率的函數。

參考資料：
/FBFA/“函數說明 ISO 語言代碼”

句法

```
FENDNORM
G62 G41
G621
```

意義

FENDNORM	自動轉角減速 OFF
G62	刀具半徑偏移量有效時，轉角在內角減速
G621	刀具半徑偏移量有效時，轉角在所有轉角減速

G62 僅適用於

- 具有有效的刀具半徑偏移量 G41、G42 以及
- 有效的連續路徑模式 G64、G641 的內角。

逼近轉角所使用的減慢進給率是利用以下公式計算得出：

F*（為了減慢進給率所作的手動倍率）*進給率調整

最大可能進給率的減慢幅度，取自轉角上刀具要進行轉向的準確位置，並且參考（刀具）中心點路徑。

類似於 G62，G621 適用於 FGROUP 之指定軸上的每一個轉角。

4.11 已程式設計的動作結尾條件（FINEA、COARSEA、IPOENDA、IPOBRKA、ADISPOSA）

功能

類似路徑插補的單節變更條件（G601、G602 及 G603），也可為工件程式中的單軸插補，或同步動作中的指令 / PLC 軸，程式設計移動結尾條件。

結尾動作條件集會影響單軸移動的工件程式單節及科技循環單節，其完成的快慢。透過 FC15/16/18 同樣套用於 PLC。

句法

```
FINEA[<軸>]
COARSEA[<軸>]
IPOENDA[<軸>]
IPOBRKA（<軸>[，<即時>]）
ADISPOSA[<軸>]=（<模式>，<視窗大小>）
```

意義

FINEA:	於達到“精確停止微調”時動作結束
COARSEA:	於達到“精確停止粗調”時動作結束
IPOENDA:	到達"插補停止"時動作結束
IPOBRKA:	可於煞車斜率中單節變更。
ADISPOSA:	動作條件結束的允差視窗大小
<軸>:	通道軸名稱（X、Y、....）
<即時>:	即時單節變更，參考煞車斜率為%
<模式>:	模式 類型: INT 值域: 0 允差視窗未啟用 1 允差視窗與設定位置相關 2 允差視窗與實際位置相關
<視窗尺寸>:	允差視窗尺寸 於設定參數 SD43610 $SA_ADISPOSA_VALUE 中，主要執行同步輸入該值。 類型: REAL

範例

範例 1：達到插補器停止時動作結束

程式碼	註解
...	
N110 G01 POS[X]=100 FA[X]=1000 ACC[X]=90 IPOENDA[X]	當輸入 1 有效時，以 1000rpm 路徑速率、90%加速值移動至位置 X100，並在達到插補器停止時動作結束。
...	
N120 EVERY $A_IN[1] DO POS[X]=50 FA[X]=2000 ACC[X]=140 IPOENDA[X]	; 當輸入 1 有效時，以路徑速率 2000 轉、140%加速值移動至位置 X50，並在達到插補器停止時動作結束。
...	

範例 2：單節變更條件，工件程式中的"煞車斜率"

程式碼	註解
	; 預設設定有效
N40 POS[X]=100	; 若 X 軸達到位置 100 且達到精確停止微調，則變更單節
N20 IPOBRKA（X，100）	; 單節變更條件，啟動煞車斜率。
N30 POS[X]=200	; 一旦 X 軸開始煞車，則變更單節。
N40 POS[X]=250	; X 軸未於位置 200 煞車，而繼續移動至位置 250，一旦 X 軸煞車，則變更單節。
N50 POS[X]=0	; X 軸煞車並返回位置 0－於位置 0 變更單節並精確停止微調。
N60 X10 F100	
N70 M30	
...	

範例 3：單節變更條件，同步動作中的煞車斜率

程式碼	註解
	; 於科技循環中：
FINEA	; 動作條件結束，精確停止微調。
POS[X]=100	; 若 X 軸達到位置 100 且達到精確停止微調，則科技循環單節變更。
IPOBRKA（X，100）	; 單節變更條件，啟動煞車斜率。
POS[X]=100	; POS[X]=100；一旦 X 軸開始煞車，則科技循環單節變更。
POS[X]=250	; X 軸未於位置 200 煞車，而繼續移動至位置 250，一旦 X 軸開始煞車，則於科技循環中變更單節。
POS[X]=250	; X 軸煞車並返回位置 0－於位置 0 變更單節並精確停止微調。
M17	

其他資訊

讀取動作結尾條件

設定的動作結尾條件可使用系統變數$AA_MOTEND[<軸>]進行確認：

值	意義
1	以“精確停止微調”動作結束
2	以“精確停止粗調”動作結束
3	以“IPO Stop”動作結束
4	軸動作之單節變更條件煞車斜率
5	煞車斜率中的單節變更，含相對於“設定位置”的允差視窗
6	煞車斜率中的單節變更，含相對於“實際位置”的允差視窗

說明

RESET（重置）後仍保留上次已程式設計的值。

煞車斜率中單節變更條件

該百分比值輸入在與主執行同步的設定資料中：

SD43600 $SA_IPOBRAKE_BLOCK_EXCHANGE

若未指定值，則此設定參數目前的值將生效。

可於 0%至 100%間調整範圍。

IPOBRKA 的額外允差視窗

除了煞車斜率中既有的單節變更條件外，尚可額外選擇第二個單節變更條件，"允差視窗"。其僅當軸在下列情況時才會啟用：

- 與先前相同，達到其煞車斜率指定之%值

  及

- 其目前實際或設定的位置不比單節中軸結束的允差遠。

參考

若需更多有關定位軸的單節變更條件之資訊，請參閱：

- 功能手冊，延伸功能；定位軸（P2）
- 程式設計手冊，基礎；"進給率控制"一章。

4.12 可程式設計的伺服參數集（SCPARA）

功能

可使用 SCPARA 程式設計工件程式及同步動作中的參數集（由 MD 組成）（目前為止，僅可透過 PLC）。

DB3n DBB9 bit3

若要確保 PLC 與 NCK 間無碰撞產生，可於 PLC → NCK 介面定義一額外位元：

DB3n DBB9 bit3“停用由 SCPARA 選擇之參數集”。

若禁止 SCPARA 輸入參數集，則若已程式設計，將不輸出錯誤訊息。

句法

SCPARA[<軸>]=<值>

意義

SCPARA	定義參數單節
<軸>	通道軸名稱（X、Y、....）
<值>	所需參數單節（1<= 值 <=6）

說明

可使用系統變數$AA_SCPAR[<軸>]確定實際參數集。

G33、G331 或 G332，由控制系統選擇的最適合的參數集。

若**伺服參數集**將於工件程式、同步動作或 PLC 中**變更**，則需擴展 PLC 使用者程式。

參考資料：
/FB1/ 功能手冊基本功能；進給率（V1），
“進給率衝擊”一節。

範例

程式碼	註解
...	
N110 SCPARA[X]= 3	; 已為 X 軸選擇第三參數集。
...	

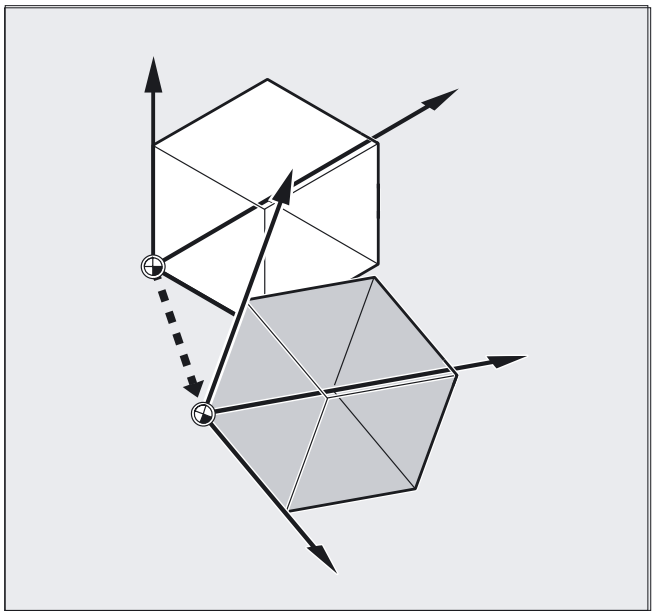
座標轉換（FRAMES）

5

5.1 透過框架變數座標轉換

功能

除於“基礎”程式設計指南中所定義之程式設計選項外，您亦可使用預先定義的框架變數來說明座標系統。



已定義下列座標系統：

MCS：機械座標系統

BCS：基本座標系統

BZS：基本原始系統

SZS：可設定零點系統

WCS：工件座標系統

何為預先定義的框架變數？

預先定義的框架變數，為已於控制語言中，定義其用途與效果的關鍵字，可於 NC 程式中處理。

可能的框架變數：

- 基本框架（基本偏移量）
- 可設定的框架
- 可程式設計框架

值指派並讀取實際值

框架變數 / 框架關係

座標轉換可經由指派框架值至框架變數來啟動。

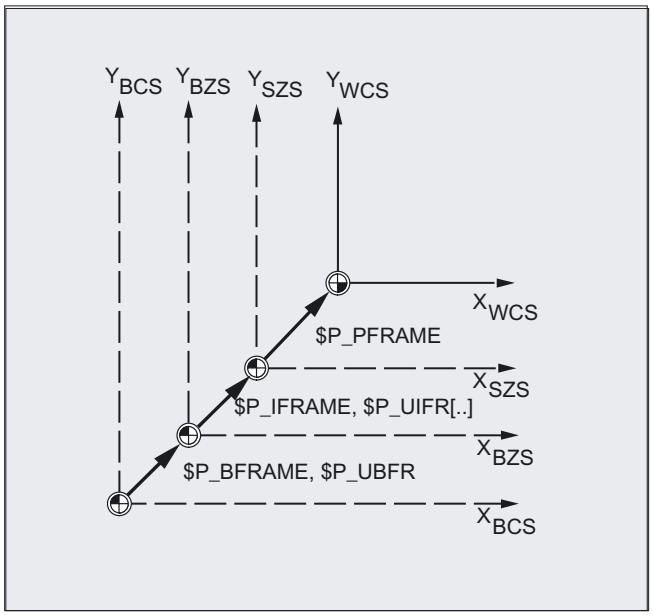
範例：`$P_PFRAME=CTRANS（X，10）`

框架變數：

`$P_PFRAME` 表示：目前可程式設計的框架。

框架：

`CTRANS（X，10）`表示：X 軸可程式設計的零點偏移量 10 毫米。



讀取實際值

可透過工件程式中預先定義的變數讀出座標系統中目前的實際值：

$AA_IM[axis]：於機械座標系統（MCS）讀取實際值

$AA_IB[axis]：於 BCS 讀取實際值

$AA_IBN[axis]：於 BOS 讀取實際值

$AA_IEN[axis]：於 SZS 讀取實際值

$AA_IW[axis]：於工件座標系統（WCS）讀取實際值

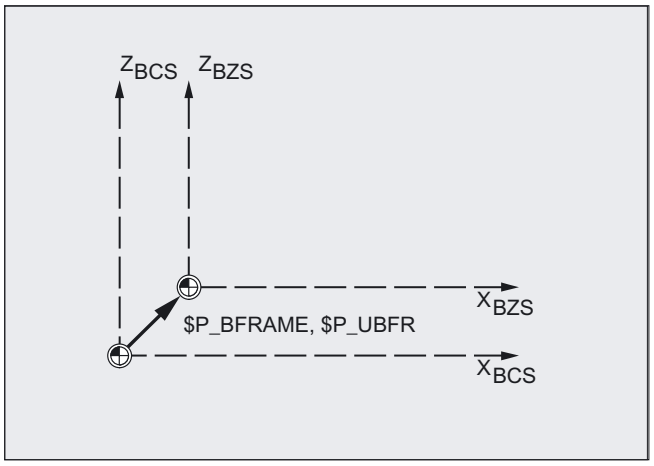
5.1.1 預先定義的框架變數（$P_BFRAME、$P_IFRAME、$P_PFRAME、$P_ACTFRAME）

$P_BFRAME

目前基本框架變數會建立基本座標系統（BCS）與基本原始系統（BOS）間的參考資料。

欲立即在程式中啟用透過$P_UBFR 定義的基本框架，您必須

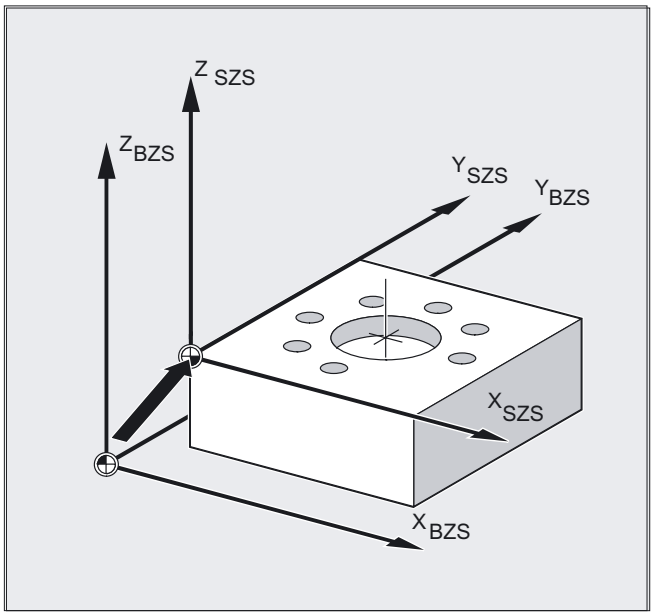
- 程式設計 G500， G54...G599 或
- 需要定義$P_UBFR 和$P_BFRAME。



$P_IFRAME

目前可設定的框架變數，會建立基本原始系統（BOS）與可設定零點系統（SZS）間的參考資料。

- $P_IFRAME 對應於$P_UIFR[$P_IFRNUM]
- 程式設計 G54 後，例如$P_IFRAME 包含由 G54 定義的轉譯、旋轉、刻度及鏡像。

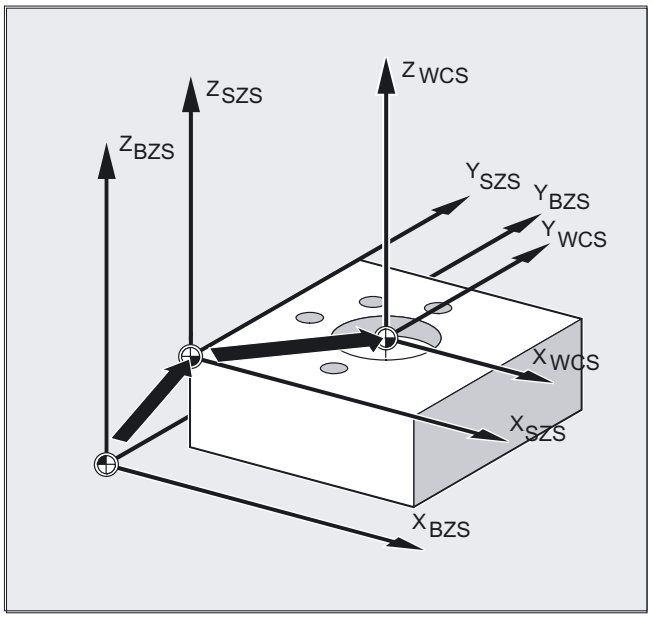


$P_PFRAME

目前可程式設計的框架變數，會建立可設定零點系統（SZS）與工件座標系統（WCS）間的參考資料。

$P_PFRAME 包含結果框架，造成

- **程式設計** TRANS/ATRANS、ROT/AROT、SCALE/ASCALE、MIRROR/AMIRROR 或
- **指派** CTRANS、CROT、CMIRROR、CSCALE 至已程式設計的 FRAME 產生的結果框架。



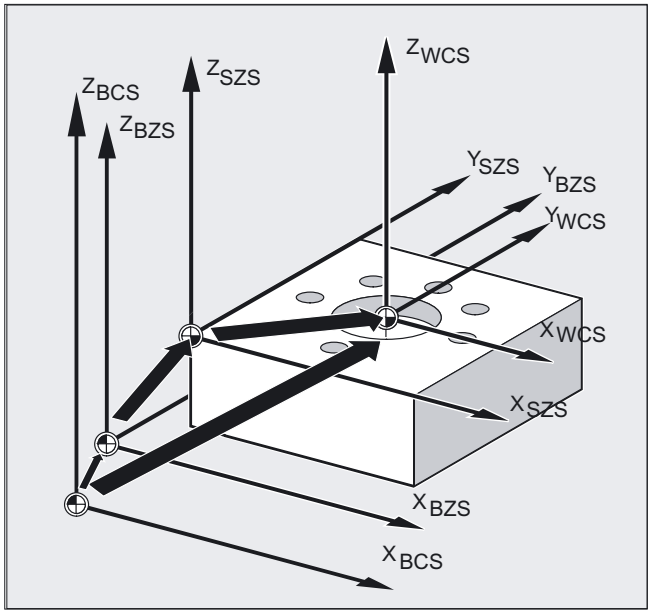
$P_ACTFRAME

目前，由鏈結產生完整框架

- 目前基本框架變數$P_BFRAME、
- 目前可設定框架變數$P_IFRAME 含系統框架，且
- 目前可程式設計框架變數$P_IFRAME 含系統框架。

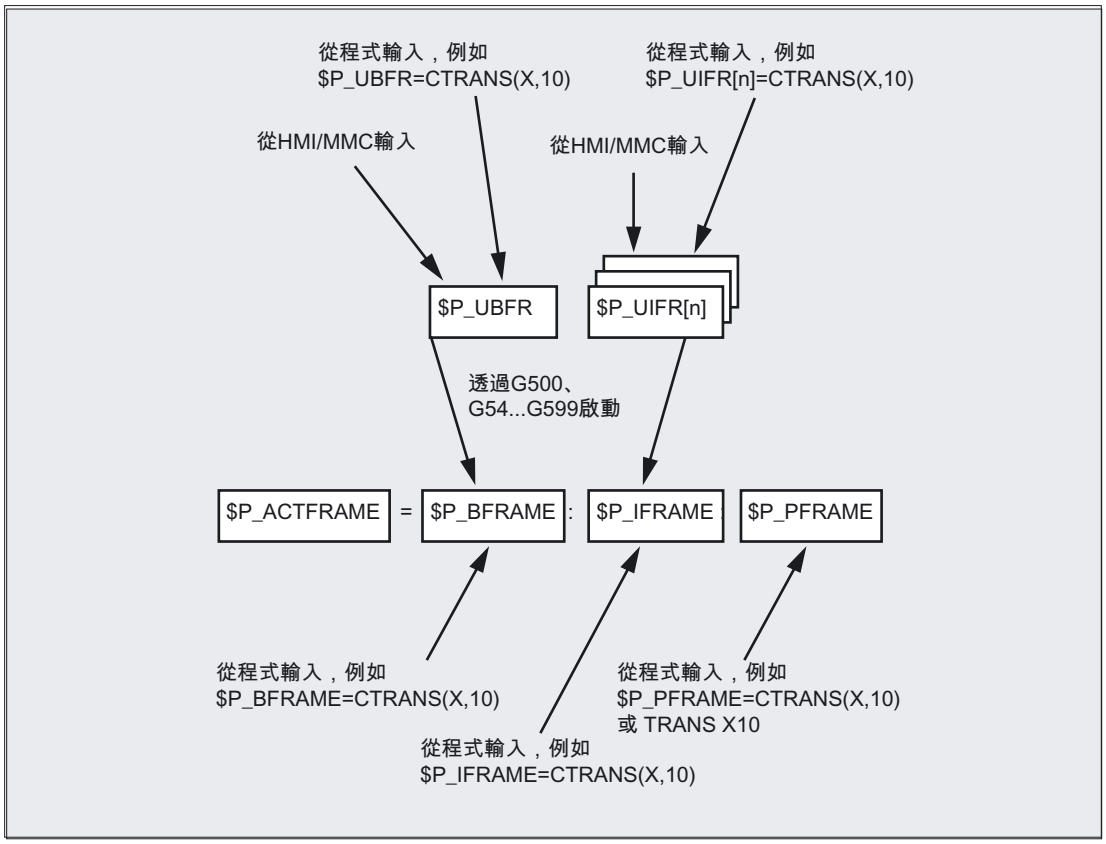
系統框架，請參閱"通道中動作之框架"一節

$P_ACTFRAME 定義目前有效的工件零點。



若已變更$P_IFRAME、$P_BFRAME 或$P_PFRAME，將重新計算$P_ACTFRAME。

$P_ACTFRAME 對應於$P_BFRAME：$P_IFRAME：$P_PFRAME



若 MD 20110 RESET_MODE_MASK 如下設定，則基本框架及可設定框架在重置後有效：

Bit0=1，bit14=1 --> $P_UBFR（基本框架）動作

Bit0=1，bit5=1 --> $P_UIFR [$P_UIFRNUM]（可設定框架）動作

預先定義的可設定框架$P_UBFR

使用$P_UBFR 程式設計基本框架，但不會在工件程式中同時有效。計算時包含使用 $P_UBFR 程式設計基本框架，若

- 啟動重置，並已在 MD RESET_MODE_MASK 設定位元 0 及 14，且
- 已執行 G500、G54...G599 敘述。

預先定義的可設定框架$P_UIFR[n]

可使用預先定義的框架變數$P_UIFR[n]，從工件程式中讀取或寫入可設定零點偏移量 G54 至 G599。

該變數產生稱為$P_UIFR[n]的 FRAME 類型一維陣列。

指派至 G 指令

如標準情況，可使用其位址值儲存五個可設定框架$P_UIFR[0]... $P_UIFR[4] 或五個同等 G 指令－G500 及 G54 至 G57。

$P_IFRAME=$P_UIFR[0]對應於 G500

$P_IFRAME=$P_UIFR[1]對應於 G54

$P_IFRAME=$P_UIFR[2]對應於 G55

$P_IFRAME=$P_UIFR[3]對應於 G56

$P_IFRAME=$P_UIFR[4]對應於 G57

您可使用機械參數變更框架數量：

$P_IFRAME=$P_UIFR[5]對應於 G505

... ...

$P_IFRAME=$P_UIFR[99]對應於 G599

說明

這允許產生最多 100 個座標系統，可於各個不同的程式中呼叫，例如，各種裝置的零點。

小心

需在 NC 程式中的個別 NC 單節裡，程式設計框架變數。**例外：**使用 G54、G55、... 程式設計可設定框架。

5.2 框架變數 / 指派值至框架

5.2.1 指派直接值（軸值、角度、比例）

功能

您可在 NC 程式中直接指派值至框架或框架變數。

句法

```
$P_PFRAME=CTRANS （X，axis value，Y，axis value，Z，axis value，…）
$P_PFRAME=CROT （X，angle，Y，angle，Z，angle，…）
$P_UIFR[..]=CROT （X，angle，Y，angle，Z，angle，…）
$P_PFRAME=CSCALE （X，scale，Y，scale，Z，scale，…）
$P_PFRAME=CMIRROR （X，Y，Z）
```

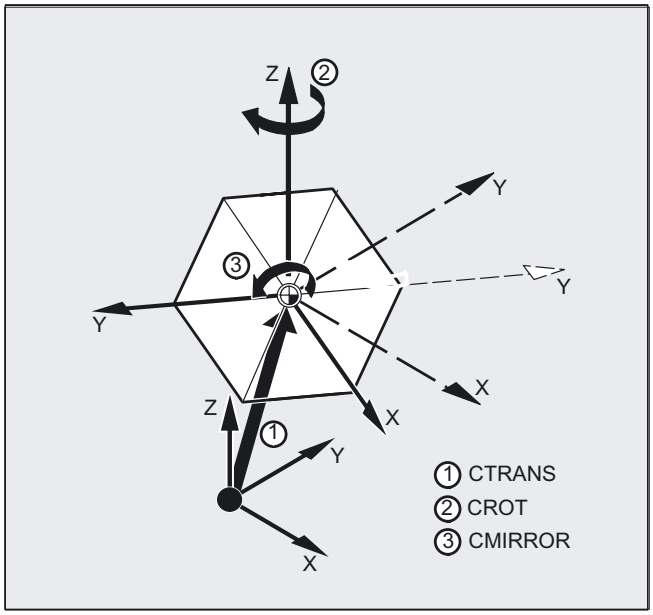
$P_BFRAME 程式設計執行，類比至$P_PFRAME。

意義

CTRANS	指定軸之轉譯
CROT	繞指定軸旋轉
CSCALE	指定軸比例變更
CMIRROR	指定軸反轉方向
X Y Z	指定幾何軸之方向偏移量值
軸值	指派偏移量的軸值
角度	指派繞指定軸旋轉的角度
比例	變更比例

範例

可透過指派值至目前可程式設計的框架，啟動轉譯、旋轉及鏡像。




```
N10 $P_PFRAME=CTRANS（X，10，Y，20，Z，5）：CROT（Z，45）：CMIRROR（Y）
```

紅色框架元件為預先指派的其他值

使用 CROT，預先指派三個含值的 UIFR 元件

程式碼	註解
`$P_UIFR[5] = CROT（X，0，Y，0，Z，0）`	
`N100 $P_UIFR[5，y，rt]=0`	
`N100 $P_UIFR[5，x，rt]=0`	
`N100 $P_UIFR[5，z，rt]=0`	

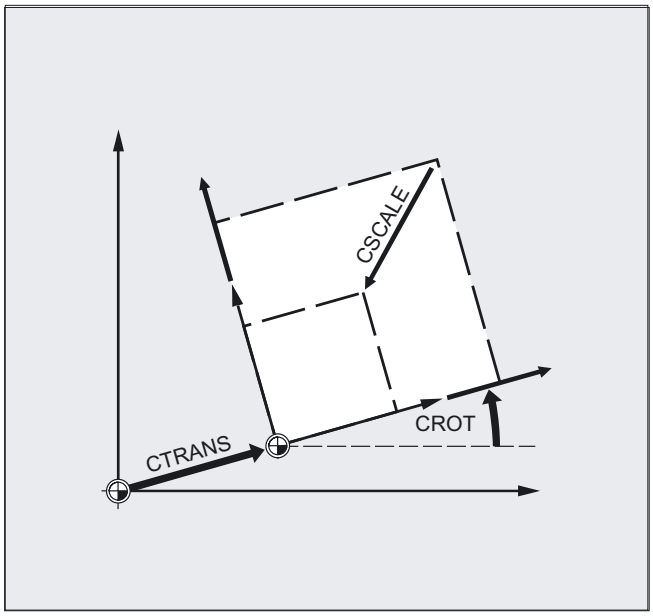
說明

可連續程式設計數個算數規則。

範例：

```
$P_PFRAME=CTRANS（…）：CROT（…）：CSCALE…
```

請注意需由冒號連鎖運算子連接指令：(...):(...). 使其指令依已程式設計的順序連結，然後接著執行。



說明

指派含上述指令的已程式設計值至框架並儲存。

在指派值至有效框架變數$P_BFRAME 或$P_PFRAME 的框架前，不會啟動該值。

5.2.2 讀取及變更框架元件（TR、FI、RT、SC、MI）

功能

此功能可存取框架的**個別**資料，例如，特定的偏移量值或旋轉角度。可修改這些值或將它們指派至另一變數。

句法

`R10=$P_UIFR[$P_UIFNUM，X，RT]`	從目前有效的可設定零點偏移量$P_UIFRNUM，指派繞 X 軸的旋轉角度 RT 至變數 R10。
`R12=$P_UIFR[25，Z，TR]`	從設定框架第 25 號的資料記錄，指派 Z 中的偏移量值 TR 至變數 R12。
`R15=$P_PFRAME[Y，TR]`	指派目前可程式設計框架 Y 中的偏移量值 TR 至變數 R15。
`$P_PFRAME[X，TR] = 25`	修改目前可程式設計框架 X 中的偏移量值 TR。立即套用 X25。

意義

`$P_UIFRNUM`	該指令會自動建立參考至目前有效的可設定零點偏移量。
`P_UIFR[n，…，…]`	指定框架編號 n 以存取可設定框架編號 n。
	指定欲讀取或修改的元件：
`TR`	TR 轉譯
`FI`	FI 轉譯微調
`RT`	RT 旋轉
`SC`	SC 調整比例修改
`MI`	MI 鏡像
`X Y Z`	亦指定對應軸 X、Y、Z（請見範例）。

RT 旋轉值域

繞第 1 幾何軸旋轉：	-180°到+180°
繞第 2 幾何軸旋轉：	-90°到+90°
繞第 3 幾何軸旋轉：	-180°到+180°

說明

呼叫框架

藉由指定系統變數$P_UIFRNUM，您可使用$P_UIFR 或 G54、G55、...
存取目前零點偏移量集（$P_UIFRNUM 包含目前設定框架數量）。

可透過指定適當數量的$P_UIFR[n]，呼叫其它已儲存的可設定$P_UIFR 框架。

對預先定義的框架變數或使用者自訂框架，指定其名稱，例如，$P_IFRAME。

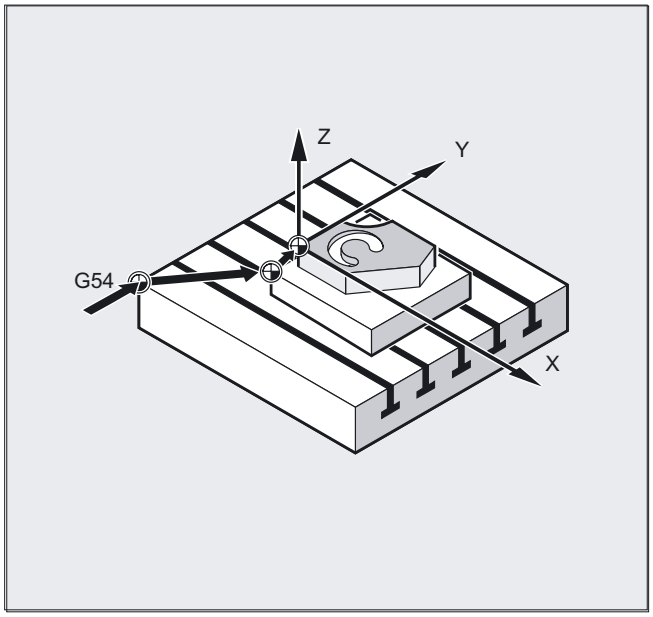
呼叫資料

將欲存取或修改之值的軸名稱及框架元件，寫入至方括號中，例如，[X，RT] 或 [Z，MI]。

5.2.3 連接完整框架

功能

完整的框架可指派至其它框架，或可於 NC 程式中相互鏈結框架。
框架鏈結適合說明多個工件，其配置於同個程序中待加工的棘爪。



框架元件僅可包含棘爪作業說明的中間值。並鏈結以產生多個工件零點。

句法

指派框架

```
DEF FRAME SETTING1
SETTING1=CTRANS（X，10）
$P_PFRAME=SETTING1
DEF FRAME SETTING4
SETTING4=$P_PFRAME
$P_PFRAME=SETTING4
```

指派使用者框架 SETTING1 的值至目前可程式設計的框架。

已暫時儲存並可再次呼叫目前可程式設計的框架。

框架鏈結

依已程式設計的順序鏈結框架。另外連續執行框架元件（轉譯、旋轉等）。

$P_IFRAME=$P_UIFR[15]：$P_UIFR[16]

$P_UIFR[15]包含，例如，零點偏移量的資料。$P_UIFR[16]的資料，例如，會另外接著處理旋轉資料。

經由鏈結可設定的框架 4 及 5，以建立可設定框架 3。

$P_UIFR[3]=$P_UIFR[4]：$P_UIFR[5]

說明

需使用連續冒號相互連結框架：.

5.2.4 定義新框架（DEF FRAME）

功能

除上述預先說明可設定的框架外，您也可選擇建立新框架。可透過建立 FRAME 類型變數，達成依您選擇所指派的名稱。

您可使用函數 CTRANS、CROT、CSCALE 或 CMIRROR 來指派值至您在 NC 程式中的框架。

句法

```
DEF FRAME PALETTE1
PALETTE1=CTRANS（…）：CROT（…）…
```

意義

DEF FRAME	建立新框架。
PALETTE1	新框架名稱
=CTRANS（...）：CROT（...）...	指派值至可能的函數

5.3 粗調或微調偏移量（CFINE、CTRANS）

功能

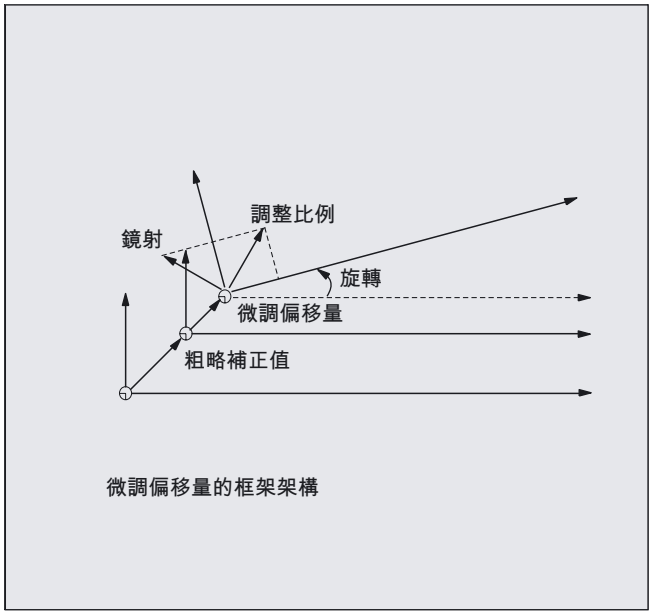
微調偏移量

可使用 CFINE （X，..，Y，...）指令，程式設計基本框架及所有其它可設定框架的微調偏移量。

僅若 MD18600 $MN_MM_FRAME_FINE_TRANS=1 時，才產生微調偏移量。

粗調偏移量

以 CTRANS（...）定義粗調偏移量。



微調偏移量的框架架構

粗調及微調偏移量相加為總偏移量。

句法

```
$P_UBFR=CTRANS（x，10） ：CFINE（x，0.1） ：CROT（x，45）   ；偏移量鏈結，
                                                            ；微調偏移量及旋轉
$P_UIFR[1]=CFINE（x，0.5 y，1.0，z，0.1）                   ；以 CFINE 覆寫完整框架
                                                            ；包含粗調偏移量
```

可透過元件規格 FI（轉譯微調）存取微調偏移量的個別元件。

```
DEF REAL FINEX                                              ；FINEX 變數定義
FINEX=$P_UIFR[$P_UIFNUM，x，FI]                             ；使用 FINEX 變數
                                                            ；取得微調偏移量
FINEX=$P_UIFR[3，x，FI]$P                                   ；使用 FINEX 變數
                                                            ；取得第 3 框架中的 X 軸
                                                            ；的微調偏移量
```

意義

CFINE（x，value，y，value，z，value）	多軸微調偏移量。附加偏移量（轉譯）。
CTRANS（x，value，y，value，z，value）	多軸粗調偏移量。絕對偏移量（轉譯）。
x y z	軸的零點移動（最大為 8）
值	轉譯工件

機台製造商

可使用 MD18600 $MN_ MM_FRAME_FINE_TRANS，於下列版本中設定微調偏移量：

0:
無法輸入或程式設計微調偏移量。無法使用 G58 或 G59。

1:
可輸入 / 程式設計可設定框架、基本框架、可程式設計框架、G58 及 G59 的微調偏移量。

說明

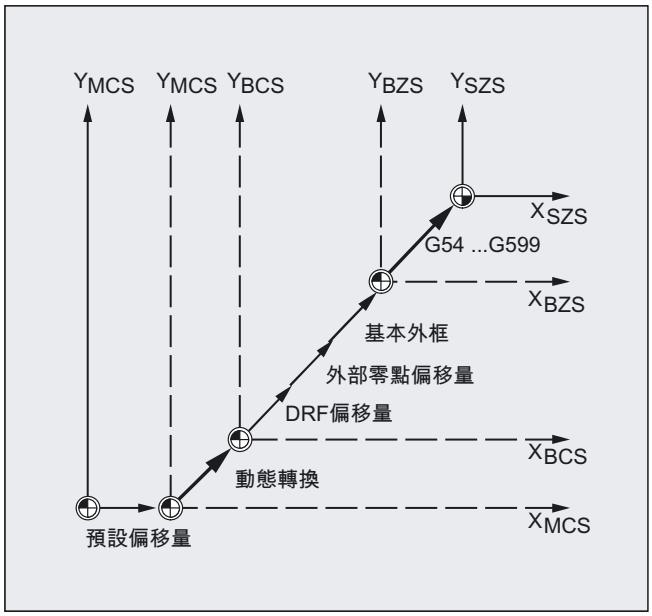
在對應框架啟動前，無法套用使用 HMI 操作變更之微調偏移量，即透過 G500，G54...G599 啟動。一旦啟動，框架的微調偏移量在框架有效期間內仍有效。

可程式設計框架無微調偏移量。若指派含微調偏移量之框架至可程式設計框架，則可透過新增粗調及微調偏移量，建立總偏移量。讀取可程式設計框架時微調偏移量永遠為零點。

5.4 外部零點偏移量

功能

此為另一個於基本及工件座標系統間移動零點的方法。
僅線性轉譯可使用外部零點偏移量程式設計。



程式設計

可透過指派軸專屬系統變數程式設計$AA_ETRANS 偏移量值。

指派偏移量值

```
$AA_ETRANS[axis]=RI
```

RI 為包含新值的類型 REAL 算術變數。

一般透過 PLC 設定，而非在工件程式中指定外部偏移量。

說明

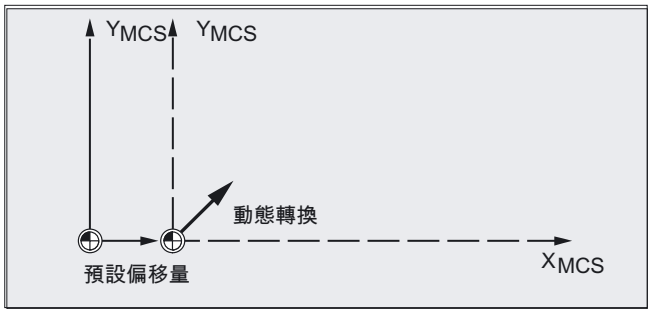
工件程式中的輸入值僅當於 VDI 介面
（NCU-PLC 介面）啟用對應訊號後才有效。

5.5 預設偏移量（PRESETON）

功能

特殊應用下，需指派新已程式設計的實際值至一或多個目前位置（固定）上的軸。

> ⚠ 小心
>
> 使用 PRESETON 函數，則參考點無效。所以應僅可對不需參考點的軸使用本函數。若欲回復原始系統，需使用 G74 逼近參考點－請參閱“檔案及程式管理”一節。



句法

```
PRESETON（axis，value）
```

意義

PRESETON	預設實際值記憶體
軸	機械軸參數
值	套用至指定軸之新增實際值

說明

僅可使用 “WHEN”或“EVERY”關鍵字執行含同步動作的預設模式。

範例

實際值指派至機械座標系統－其值參考機械軸。

```
N10 G0 A760
N20 PRESETON（A1，60）
```

A 軸移動至位置 760。於位置 760，指派新實際值 60 至機械軸 A1。然後從該點，在新實際值系統中執行定位。

5.6 從空間中的三個量測點計算框架（MEAFRAME）

功能

MEAFRAME 為 840D 語言的副檔名，用來支援量測循環。

函數 MEAFRAME 從三處理想及對應的量測點計算框架。

當工件已定位待加工，一般來說，會參考其理想位置，移動及旋轉其相對於直角座標機台系統之位置。對精確加工或量測，需費時實體調整工件或工件程式中定義的動作必須變更。

可透過取樣空間中已知其理想位置的三點定義框架。觸發式探針或光學感測器可用來取樣，觸碰在支撐板或探頭精確固定的特別孔洞。

句法

```
MEAFRAME IDEAL_POINT，MEAS_POINT，FIT_QUALITY）
```

意義

MEAFRAME	空間中三量測點的框架計算	
IDEAL_POINT	包含理想點三座標的實數資料陣列	
MEAS_POINT	包含量測點三座標的實數資料陣列	
FIT_QUALITY	REAL 變數，	傳回下列資訊：
	-1:	理想點幾乎位於一條直線上：無法計算框架。傳回的框架變數包含一個中性框架。
	-2:	量測點幾乎位於一條直線上：無法計算框架。傳回的框架變數包含一個中性框架。
	-4:	旋轉矩陣的計算因不同原因而失敗。
	正值：	需用扭曲總和（點與點間的距離） 將量測的三角轉換為與 理想三角形全等的三角形。

說明

量測品質

為使用旋轉及轉譯將已量測之座標繪製於理想座標上，經量測點形成之三角形需與理想三角形全等。可透過補正演算法達成，以將量測三角形重塑為理想三角形所需的離差平方和減至最低。

由於可使用有效扭曲來判斷量測的品質，MEAFRAME 視其為其他變數傳回。

說明

由 MEAFRAME 建立的框架可由 ADDFRAME 函數在框架鏈結中，如從 SW 6.3 般，轉換為另一框架。
範例：框架鏈結“以 ADDFRAME 連鎖”。

關於 ADDFRAME（FRAME、STRING）參數的其他資訊，請參閱
/FB1/ 功能手冊基本功能；軸、座標系統、框架（K2），“FRAME 鏈結”。

範例

程式碼	註解
	; 工件程式 1
DEF FRAME CORR_FRAME	

設定量測點

程式設計	註解
DEF REAL IDEAL_POINT[3, 3] = SET（10.0, 0.0, 0.0, 0.0, 10.0, 0.0, 0.0, 0.0, 10.0）	
DEF REAL MEAS_POINT[3, 3] = SET （10.1, 0.2, -0.2, -0.2, 10.2, 0.1, -0.2, 0.2, 9.8）	; 供測試
DEF REAL FIT_QUALITY = 0	
DEF REAL ROT_FRAME_LIMIT = 5	; 最多允許工件位置 5 度旋轉
DEF REAL FIT_QUALITY_LIMIT = 3	; 理想與量測三角形間最大允許 3 毫米偏移量
DEF REAL SHOW_MCS_POS1[3]	
DEF REAL SHOW_MCS_POS2[3]	
DEF REAL SHOW_MCS_POS3[3]	

程式碼	註解
N100 G01 G90 F5000	
N110 X0 Y0 Z0	
N200 CORR_FRAME=MEAFRAME（IDEAL_POINT, MEAS _POINT, FIT_QUALITY）	
N230 IF FIT_QUALITY < 0	
SETAL（65000）	
GOTOF NO_FRAME	
ENDIF	
N240 IF FIT_QUALITY > FIT_QUALITY_LIMIT	
SETAL（65010）	
GOTOF NO_FRAME	
ENDIF	
N250 IF CORR_FRAME[X, RT] > ROT_FRAME_LIMIT	; 限制第一 RPY 角度
SETAL（65020）	
GOTOF NO_FRAME	
ENDIF	
N260 IF CORR_FRAME[Y, RT] > ROT_FRAME_LIMIT	; 限制第二 RPY 角度
SETAL（65021）	
GOTOF NO_FRAME	

程式碼	註解
ENDIF	
N270 IF CORR_FRAME[Z，RT] > ROT_FRAME_LIMIT	; 限制第三 RPY 角度
SETAL（65022）	
GOTOF NO_FRAME	
ENDIF	
N300 $P_IFRAME=CORR_FRAME	; 啟動含可設定框架之樣本框架
	; 將幾何軸定位於理想點以檢查框架
N400 X=IDEAL_POINT[0，0] Y=IDEAL_POINT[0，1] Z=IDEAL_POINT[0，2]	
N410 SHOW_MCS_POS1[0]=$AA_IM[X]	
N410 SHOW_MCS_POS1[1]=$AA_IM[X]	
N430 SHOW_MCS_POS1[2]=$AA_IM[Z]	
N500 X=IDEAL_POINT[1，0] Y=IDEAL_POINT[1，1] Z=IDEAL_POINT[1，2]	
N510 SHOW_MCS_POS2[0]=$AA_IM[X]	
N520 SHOW_MCS_POS2[1]=$AA_IM[Y]	
N530 SHOW_MCS_POS2[2]=$AA_IM[Z]	
N600 X=IDEAL_POINT[2，0] Y=IDEAL_POINT[2，1] Z=IDEAL_POINT[2，2]	
N610 SHOW_MCS_POS3[0]=$AA_IM[X]	
N620 SHOW_MCS_POS3[1]=$AA_IM[Y]	
N630 SHOW_MCS_POS3[2]=$AA_IM[Z]	
N700 G500	; 停用可設定框架，如同用零點框架（無輸入、預先指派之值）。
No_FRAME	; 停用可設定框架，如同以零點框架預先指派（無輸入值）。
M0	
M30	

連鎖框架範例

偏移量 MEAFRAME 鏈結

MEAFRAME（ ）函數提供一個偏移量框架。若此偏移量框架，與使用呼叫函數時已啟用設定框架$P_UIFR[1]連鎖，例如，G54，接收可設定之框架，以供步驟或加工進一步轉換。

以 ADDFRAME 連鎖

若您希望框架鏈結中的此偏移量框架套用至其它位置，或在可設定框架前有其它啟用的框架，可使用 ADDFRAME（ ）函數鏈結至其中一個通道基本框架或一個系統框架中。

下列為框架中不可啟用的：

- 以 MIRROR 產生鏡像
- 以 SCALE 產生比例

設定點及實際值之輸入參數為工件座標。在控制器的基本系統中，需永遠

- 以公制或英制（G71/G70）且
- 參考半徑（DIAMOF）

指定該座標。

5.7 NCU 全域框架

功能

NCU 所有通道僅使用一組 NCU 全域框架。可於所有通道讀寫 NCU 全域框架。NCU 全域框架在個別通道中啟用。

含偏移量之**通道軸及機械軸**可透過全域框架產生比例及鏡像。

幾何關係及框架鏈結

使用全域框架時，軸之間並無幾何關係。因此無法執行旋轉或程式設計幾何軸辨識碼。

- 全域框架上無法使用旋轉。拒絕程式設計旋轉，並發出警報：顯示“18310 通道%1 單節%2 框架：不允許旋轉”。
- 可鏈結全域框架及通道專屬框架。結果框架包括內含所有軸旋轉的所有框架元件。拒絕指派含旋轉元件的框架至全域框架，並輸出警報“框架：不允許旋轉”。

NCU 全域框架

NCU 全域基本框架$P_NCBFR[n]

最多可設定八個 NCU 全域基本框架：

亦可用通道專屬基本框架。

可於所有 NCU 通道讀寫全域框架。寫入全域框架時，使用者需確保通道座標。例如，可使用等候標記（WAITMC）達成。

機械製造商

使用機械參數設定全域基本框架的數量，請參閱
/FB1/ 功能手冊基本功能；軸、座標系統、框架（K2）。

NCU 全域可設定框架$P_UIFR[n]

可經由 NCU 全域或通道專屬，設定所有可設定框架 G500、 G54...G599。

機械製造商

可透過機械參數$MN_MM_NUM_GLOBAL_USER_FRAMES 的協助，重新設定所有可設定框架為全域框架。

可使用通道軸識別碼及機械軸識別碼，作為框架程式指令中的座標軸識別碼。拒絕程式設計幾何識別碼，並發出警報。

5.7.1 通道專屬框架（$P_CHBFR、$P_UBFR）

功能

可由操作員動作或從 PLC 讀寫可設定框架及基本框架：

- 透過工件程式，或
- 透過操作員面板介面。

微調偏移量也可供全域框架使用。亦產生全域框架停止，與通道專屬框架情況相同，透過 G53、G153、SUPA 及 G500。

機械製造商

可於通道中透過 MD 28081 MM_NUM_BASE_FRAMES 設定基本框架的數量。標準設定為每個通道至少一個基本框架。每個通道最多支援八個基本框架。除八個基本框架外，通道中也可有八個 NCU 全域基本框架。

通道專屬框架

$P_CHBFR[n]

可使用系統變數$P_CHBFR[n]讀寫基本框架。寫入基本框架時，在執行 G500、G54...G599 指令前不會啟動鏈結的總基本框架。變數主要係用來儲存寫入作業至 HMI 或 PLC 上的基本框架。該框架變數係由資料備份儲存。

通道中第一個基本框架

寫入至預先定義的$P_UBFR 變數時，未同時啟動含現場設備 0 的基本框架，僅當執行 G500、G54...G599 指令後才會啟動。也可於程式中讀寫該變數。

$P_UBFR

$P_UBFR 與$P_CHBFR[0]相同。預設的通道中永遠存在一個基本框架，因此系統變數可與較舊版本相容。若無通道專屬基本框架，讀取 / 寫入時會發出警報：“框架：指令不允許”。

5.7.2 通道中有效框架

功能

從工件程式透過該框架相關系統變數，輸入通道中有效框架。亦包含系統框架。可透過工件程式中的系統變數，讀寫目前系統框架。

通道中有效框架

概觀

目前系統框架	為：
$P_PARTFRAME	TCARR 及 PAROT
$P_SETFRAME	PRESET（預設）及括擦
$P_EXTFRAME	外部零點偏移量
$P_NCBFRAME[n]	目前 NCU 全域基本框架
$P_CHBFRAME[n]	目前通道基本框架
$P_BFRAME	目前通道中第一個基本框架
$P_ACTBFRAME	完整基本框架
$P_CHBFRMASK 及$P_NCBFRMASK	完整基本框架
$P_IFRAME	目前可設定框架
目前系統框架	為：
$P_TOOLFRAME	TOROT 及 TOFRAME
$P_WPFRAME	工件參考點
$P_TRAFRAME	轉換
$P_PFRAME	目前可程式設計框架
目前系統框架	為：
$P_CYCFRAME	循環
P_ACTFRAME	目前總框架
FRAME 鏈結	由總基本框架組成目前框架

$P_NCBFRAME[n]目前 NCU 全域基本框架

可使用系統變數$P_NCBFRAME[n]讀寫目前全域基本框架欄位元件。由通道中的寫入程序計算出所得的總基本框架。

僅當通道中之框架已程式設計時，才會啟動已修改之框架。若欲為 NCU 中所有通道修改框架，需同時寫入$P_NCBFR[n]及$P_NCBFRAME[n]。此時其它通道需啟動框架，例如，使用 G54。無論何時寫入基本框架，皆會再次計算完整基本框架。

$P_CHBFRAME[n]目前通道基本框架

可使用系統變數$P_CHBFRAME[n]讀寫目前通道基本框架欄位元件。於通道中計算所得的完整基本框架，作為寫入作業之結果。無論何時寫入基本框架，皆會再次計算完整基本框架。

$P_BFRAME 目前通道中第一個基本框架

可使用預先定義的框架變數$P_BFRAME，以 0 現場設備讀寫目前基本框架，其於通道、工件程式中有效。寫入的基本框架立即包含於計算中。

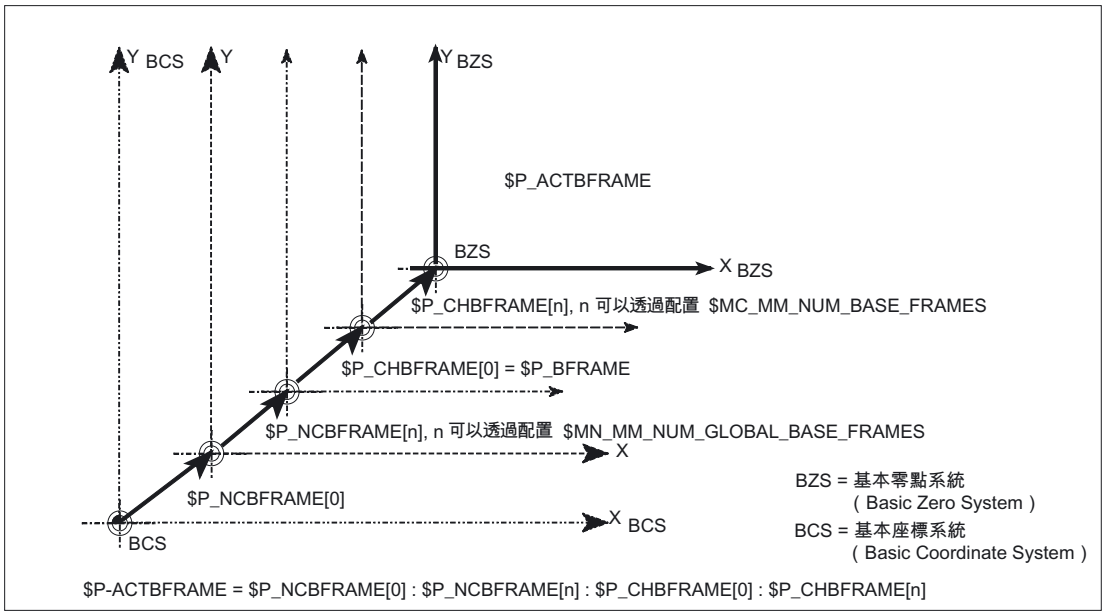
$P_UBFR 與$P_CHBFR[0]相同。系統變數永遠有一個有效的預設值。若無通道專屬基本框架，讀取 / 寫入時會發出警報：“框架：指令不允許”。

$P_ACTBFRAME 完整基本框架

$P_ACTFRAME 變數決定鏈結的完整基本框架。變數為唯讀。

$P_ACTFRAME 對應於

$P_NCBFRAME[0] : ... : $P_NCBFRAME[n] : $P_CHBFRAME[0] : ... : $P_CHBFRAME[n].



$P_CHBFRMASK 及$P_NCBFRMASK 完整基本框架

可使用系統變數$P_CHBFRMASK 及$P_NCBFRMASK 來選擇，包含於“完整”基本框架計算中的基本框架。變數僅可於程式中程式設計，並透過操作員面板介面來讀取。變數值經轉譯為位元遮罩，並決定包含於計算中的$P_ACTFRAME 基本框架欄位元件。

可使用$P_CHBFRMASK 定義包含於計算中的通道專屬基本框架，並可使用$P_NCBFRMASK 定義包含於計算中的 NCU 全域基本框架。

程式設計變數後，會再次計算總基本框架及總框架。重置回預設設定後，

$P_CHBFRMASK = $MC_CHBFRAME_RESET_MASK 及

$P_NCBFRMASK = $MC_CHBFRAME_RESET_MASK 的值。

例如

$P_NCBFRMASK = 'H81' ; $P_NCBFRAME[0] : $P_NCBFRAME[7]

$P_CHBFRMASK = 'H11' ; $P_CHBFRAME[0] : $P_CHBFRAME[4]

$P_IFRAME 目前可設定框架

可使用預先定義的框架變數$P_IFRAME 讀寫目前可設定框架，其於通道、工件程式中有效。寫入的可設定框架立即包含於計算中。

在 NCU 全域可設定框架的情況下，已修改的框架僅在框架已程式設計的通道中才會動作。若欲為 NCU 中所有通道修改框架，需同時寫入$P_UIFR[n]及$P_IFRAME。而此時其它通道需啟動對應的框架，例如，使用 G54。

$P_IFRAME 目前可程式設計框架

$P_PFRAME 為由程式設計的 TRANS/ATRANSG58/G59、ROT/AROT、SCALE/ASCALE、MIRROR/AMIRROR 或由指派 CTRANS、CROT、CMIRROR、CSCALE 至已程式設計的 FRAME 後所得的已程式設計框架。

目前，可程式設計的框架變數，建立了參考資料，介於可設定的

- 零點系統（SZS），及
- 工件座標系統（WCS）。

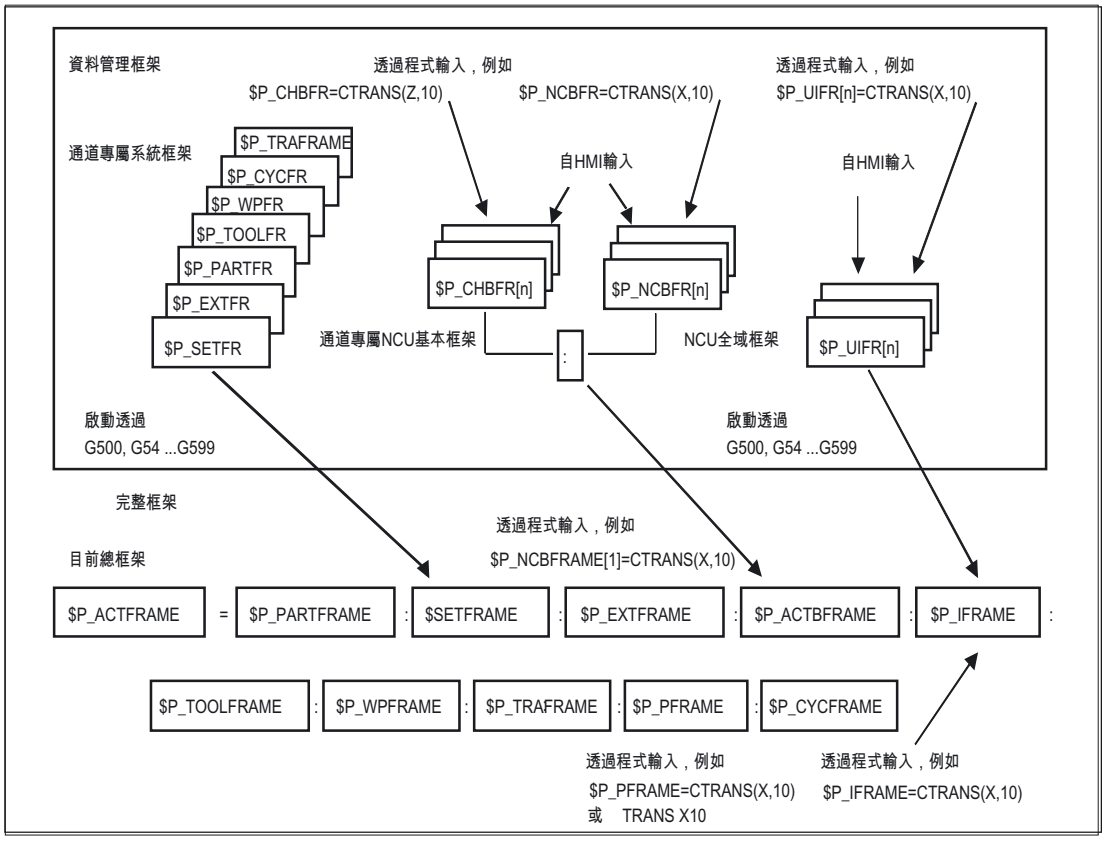
P_ACTFRAME 目前完整框架

所得的目前完整框架$P_ACTFRAME，現為所有基本框架、目前可設定框架及可程式設計的框架間的鏈結。只要變更框架元件，便會更新目前框架。

$P_ACTFRAME 對應於

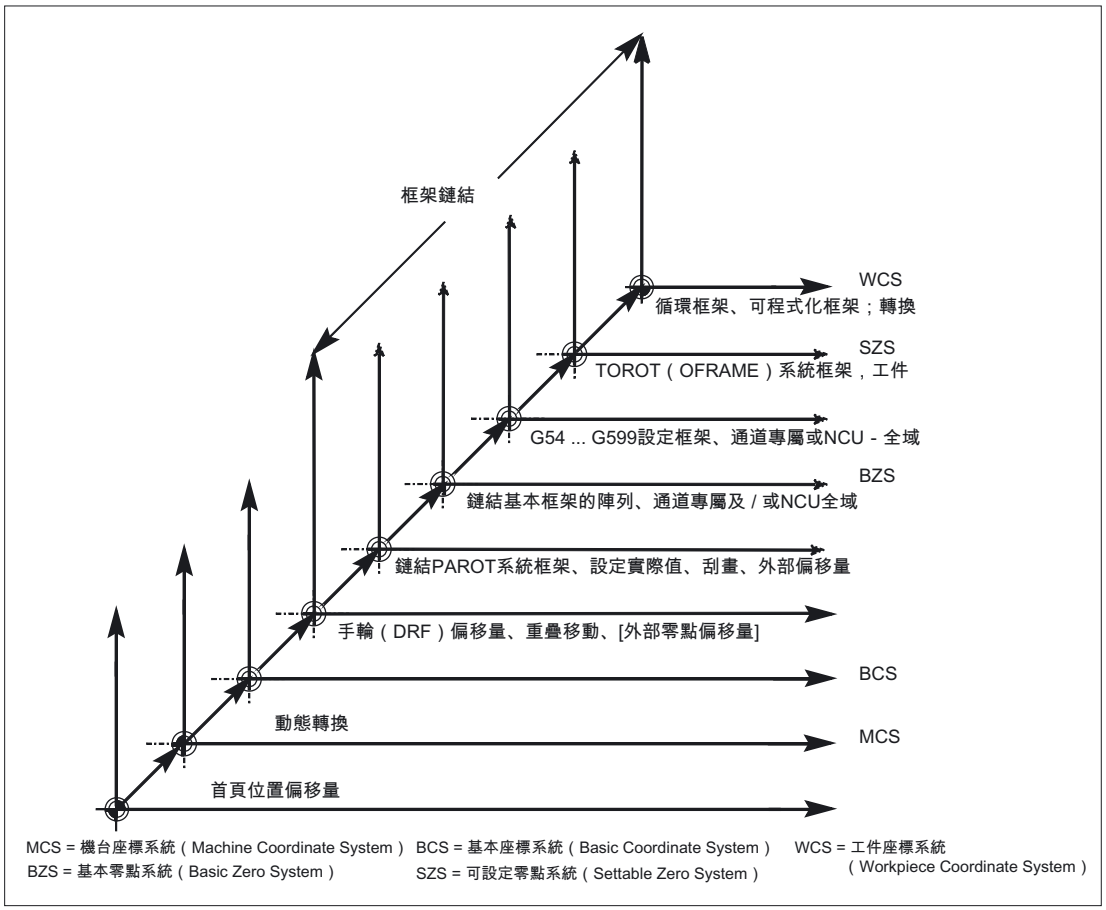
$P_PARTFRAME：$P_SETFRAME：$P_EXTFRAME：$P_ACTBFRAME：$P_IFRAME：

$P_TOOLFRAME：$P_WPFRAME：$P_TRAFRAME：$P_PFRAME：$P_CYCFRAME：



框架鏈結

目前框架由總基本框架、可設定框架、系統框架及可程式設計框架，根據上述的目前總框架組成。



6 轉換

6.1 轉換類型的一般程式設計

一般函數

您可選擇使用適當的參數程式設計轉換類型，來控制多種機械動態。可使用該參數，根據所選的轉換，宣告空間中的刀具方向及旋轉軸的方向移動。

三、四及五軸轉換中，已程式設計的位置資料永遠與刀具的刀尖相關，在空間中與已加工的平面成直角移動。直角座標從基本座標系統轉換至機械座標系統，且與幾何軸相關。由此定義操作點。虛擬旋轉軸定義空間中的刀具方向，並且使用 TRAORI 程式設計。

在動態轉換的情況下，可於直角座標系統中程式設計位置。控制將以 TRANSMIT、TRACYL 及 TRAANG 程式設計的直角座標系統移動動作，安排至實際機械軸的移動動作上。

程式設計

三、四及五軸轉換（TRAORI）

使用 TRAORI 指令及轉換數量、方向向量及旋轉軸偏移量三個可能的參數，啟動宣告的方向轉換。

TRAORI（轉換數量，方向向量，旋轉軸偏移量）

動態轉換

TRANSMIT（轉換數量）經宣告的轉換為動態轉換的範例。

TRACYL（作業直徑，轉換數量）

TRAANG（偏移軸角度，轉換數量）

停用有效轉換

可使用 TRAFOOF 來停用目前有效轉換。

方向轉換

三、四及五軸轉換（TRAORI）

對於機台工作區空間中，所設定的表面最佳加工，加工刀具除 X、Y 及 Z 三個線性軸外，還需其他的軸。其他軸定義空間中的方向，且於後續區段中稱做方向軸。它們可利用為四種具有不同動力學的機台上之旋轉軸。

1. 兩軸迴轉頭，例如，含一個在固定刀具表中，與線性軸平行之旋轉軸的萬向刀具頭。
2. 兩軸旋轉表，例如，含刀具表的固定迴轉頭，其可繞兩軸旋轉。
3. 單軸迴轉頭及單軸旋轉表，例如，一個含刀具表旋轉刀具的可旋轉迴轉頭，可繞一軸旋轉。
4. 兩軸迴轉頭及單軸旋轉表，例如，在刀具表上，可繞一軸旋轉，和一個含可自轉之刀具的可旋轉迴轉頭。

3 及 4 軸轉換為 5 軸轉換的特殊類型，並以與 5 軸轉換相同的方式程式設計。

“一般 3/4/5/6 軸轉換”的函數範圍，適合直角旋轉軸及通用銑削頭之轉換，與其他的方向轉換相同，也可使用 TRAORI 啟動這四種機台類型。在一般 5/6 軸轉換情況下，刀具方向有第三度自由度，刀具可相對於刀具方向自轉，因此可在空間中依所需導向。

參考資料：/FB3/ 功能手冊，特殊函數；3 至 5 軸轉換（F2）

不管動力學，初始刀具方向設定

ORIRESET

若使用 TRAORI 啟用方向轉換，則 ORIRESET 可用來以選配的參數 A、B、C，指定最多三方向軸的初始設定。已程式設計的參數順序，依照由轉換所定義的方向軸順序指派至圓軸。程式設計 ORIRESET（A，B，C），使方向軸以線性及同步動作，從其目前位置移動至指定的初始設定位置。

動態轉換

TRANSMIT 與 TRACYL

對車削機台上銑削，

1. 使用 TRANSMIT 於車削夾具平面加工，或
2. 使用 TRACYL 在圓柱體上以任何路徑溝槽加工

可為宣告轉換程式設計。

TRAANG

若需要傾斜進給的設定進給軸選項（例如，用於研磨技術），可使用 TRAANG 為已宣告的轉換，程式設計一可設定角度。

直角座標 PTP 移動

動態轉換亦包括所謂的“直角座標 PTP 移動”，最多可程式設計 8 個不同的關節接頭位置 STAT=。雖然位置已程式設計在直角座標系統中，但在機械座標系統中會發生機台移動。

參考資料：
/FB2/ 函數延伸功能的說明；動態轉換（M1）

鏈結轉換

可接連切換兩轉換。在此鏈接的第二個轉換，會從第一個轉換取得軸的動作零件。

第一個轉換可為：

- 方向轉換 TRAORI
- 極轉換 TRANSMIT
- 柱體轉換 TRACYL
- 傾斜軸轉換 TRAANG

第二個轉換需為傾斜軸 TRAANG 類型的轉換。

6.1.1 轉換的方向移動

移動動作及方向移動

已程式設計方向的移動動作主要由機台類型決定。 對使用 TRAORI 之三、四及五軸類型的轉換，旋轉軸或於樞軸上的線性軸定義刀具的方向移動。

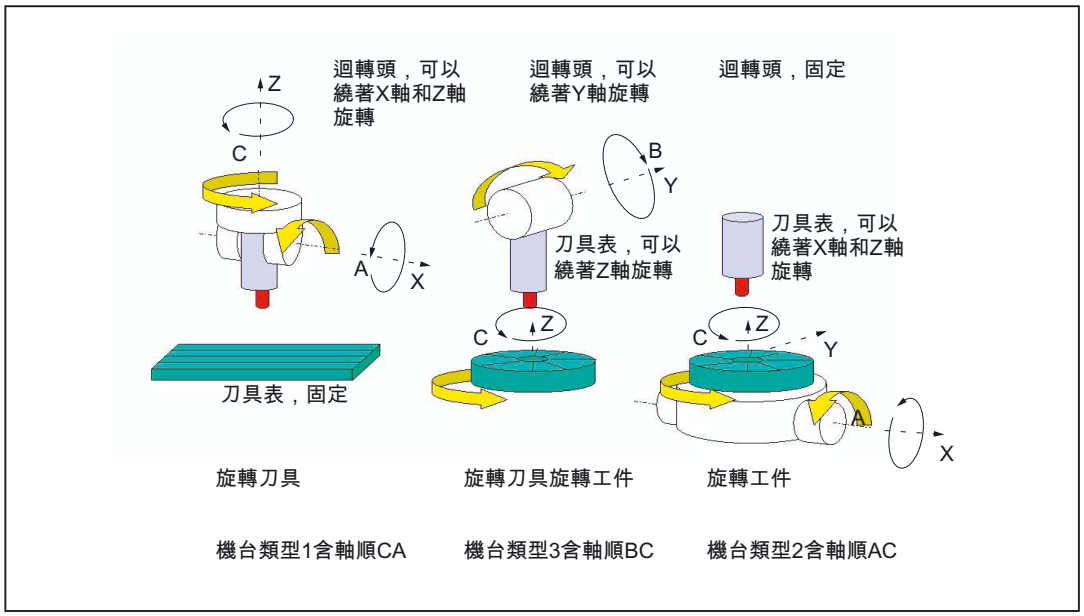
涉入方向轉換的旋轉軸位置中之變更，會導致剩餘機械軸的補正移動。 刀具刀尖的位置維持不變。

可使用虛擬軸的旋轉軸辨識碼 A...、B...、C...，視情況應用程式設計刀具的方向移動，經由輸入 Euler 或 RPY 角度或方向，或表面法線向量、錐形旋轉軸或錐形柱面上中間方向的正規化向量。

使用 TRANSMIT、TRACYL 及 TRAANG 動態轉換的情況下，控制會將已程式設計的直角座標系統移動動作，安排至實際機械軸的移動動作上。

三、四及五軸轉換之機械動態（TRAORI）

最多皆可使用兩個旋轉軸旋轉刀具及刀具表。 亦可迴轉頭及旋轉表的組合（單軸情況下）。



機台類型	程式設計方向
三軸轉換機台類型 1 及 2	僅在與旋轉軸垂直的平面中，程式設計刀具方向。 有 **兩個**平移軸（線性軸）及 **一個**旋轉的軸（旋轉軸）。
四軸轉換機台類型 1 及 2	僅在與旋轉軸垂直的平面中，程式設計刀具方向。 有 **三個**平移軸（線性軸）及 **一個**旋轉的軸（旋轉軸）。
五軸轉換機台類型 3 單軸迴轉頭及單軸旋轉表	程式設計方向轉換。 含 **三個**線性軸及**兩個**直角旋轉軸的動力學。 旋轉軸與三個線性軸其中兩軸平行。 兩由兩個直角線性軸移動第一個旋轉軸。 它使用刀具旋轉第三個線性軸。 第二個旋轉軸旋轉工件。

一般 5/6 軸轉換

機台類型	程式設計方向轉換
一般五 / 六軸轉換機台類型 4 含刀具的兩軸迴轉頭會自轉並繞單軸旋轉表旋轉	程式設計方向轉換。 含 三個線性軸及三個直角旋轉軸的動力學。 旋轉軸與三個線性軸其中兩軸平行。兩由兩個直角線性軸移動第一個旋轉軸。它使用刀具旋轉第三個線性軸。 第二個旋轉軸旋轉工件。 可使用其他以 THETA 旋轉角度自轉的刀具旋轉，程式設計基本刀具方向。

呼叫“一般三、四及五 / 六軸轉換”時，亦可傳輸刀具的基本方向。 旋轉軸方向的相關限制不再適用。 若旋轉軸間不完全垂直，或現有旋轉軸與線性軸間不完全平行，“一般五 / 六軸轉換”能為刀具方向提供更好的結果。

動態轉換 TRANSMIT、TRACYL 及 TRAANG

對車削機台上的銑削，或研磨時可供傾斜進給設定之軸，應套用下列預設的符合已宣告之轉換軸配置：

TRANSMIT	極轉換之啟動
車削夾具中的平面加工	旋轉軸 與旋轉軸垂直的進給軸 與旋轉軸平行的縱向軸

TRACYL	圓柱表面轉換的啟動
在圓柱體以任何路徑溝槽加工。	旋轉軸 與旋轉軸垂直的進給軸 與旋轉軸平行的縱向軸

TRAANG	傾斜軸轉換的啟動
以傾斜進給軸來進行加工	旋轉軸 含可參數化角度的進給軸 與旋轉軸平行的縱向軸

直角座標 PTP 移動

機台於機械座標中移動且使用下列項目程式設計：

TRAORI	轉換啟動
PTP 點對點動作	於直角座標系統（MCS）中逼近位置
CP	於（BCS）中直角座標軸的路徑動作
STAT	關節接頭位置與轉換相關
TU	軸以最短路徑移動的角度

含一般 5/6 軸轉換之 PTP 橫向

機台使用機械座標及刀具方向移動，可使用圓軸位置及 Euler 及 / 或 RPY 角度向量程式設計該移動，不管動力學或方向向量。

在這些情況下，可於錐形柱面上，使用大圓弧插補或方向向量插補，進行圓軸插補、向量插補。

範例： 通用銑削頭上的三至五軸轉換

機台工具至少有五個軸：

- 三個用來在直線裡移動的平移軸，將操作點移動至工作區的任何位置上。
- 以可設定角度（通常為 45 度）配置兩旋轉迴轉軸，使刀具迴轉至限制設定為 45 度的半圓位置空間中。

6.1.2 方向轉換概觀 TRAORI

可與 TRAORI 搭配的程式設計類型

機台類型	含有效轉換 TRAORI 的程式設計
機台類型 1、2 或 3 兩軸迴轉頭或兩軸旋轉表，或單軸迴轉頭與單軸旋轉表的組合。	可以使用依照機械動態的機械參數之 **特定機台**為基準， 或根據 **特定工件**，以獨立於機械動態的可程式設計之方向 設定方向軸的軸順序及刀具的軸方向。 參考系統中，以下列項目程式設計方向軸的旋轉方向： －ORIMKS 參考系統 = 機械座標系統 －ORIWKS 參考系統 = 工件座標系統 預設設定為 ORIWKS。 使用下列項目程式設計方向軸： 機械軸位置 A、B、C 管理 A2、B2、C2 角度程式設計虛擬軸 －使用 ORIEULER 透過尤拉角（標準） －ORIPY 透過 RPY 角 －ORIVIRT1 透過虛擬方向軸第一個定義 －ORIVIRT2 透過虛擬方向軸第二個定義， 包含插補類型間之差異： **線性插補** －方向軸 ORIAXES 或機械軸 **大半徑圓弧插補**（方向向量的插補） －從方向軸的 ORIVECT， 經由指定向量元件（方向 / 表面正常）A3、B3、C3 程式設計方向軸。 程式設計最終刀具方向。 在單節開始時，程式設計表面正常向量的 A4、B4、C4。 在單節結束時，程式設計單節垂直於平面的向量 A5、B5、C5。 LEAD 刀具方向的導角。 TILT 刀具方向的傾斜角度。

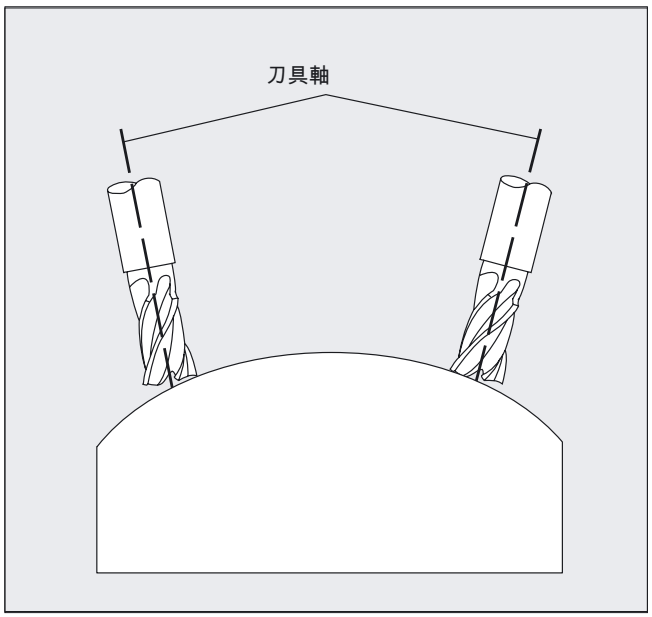
機台類型	含有效轉換 TRAORI 的程式設計
	錐形柱面上的方向向量插補。 使用插補，**方向變更至**空間中任何一處的 錐形柱面上： －平面中 ORIPLANE（大半徑圓弧插補） －錐形柱面順時針方向上的 ORICONCW －錐形柱面逆時針方向上的 ORICONCCW。 A6、B6、C6 取向向量（錐形旋轉軸） －OICONIO 於錐形柱面插補，使用： A7、B7、C7 中間向量（初始及最終方向）或 －錐形柱面上的 ORICONTO，切線變化 變更方向與含下列項目**的路徑**相關 －ORICURVE 的兩接點移動使用 PO[XH]=（xe，x2，x3，x4，x5），方向多項式最高達五次 PO[YH]=（ye，y2，y3，y4，y5）方向多項式最高達五次 PO[ZH]=（ze，z2，z3，z4，z5）方向多項式最高達五次 －ORIPATHS 使用 A8、B8、C8 刀具再定向階段， 方向特性的平滑化對應於：回退移動時的刀具方向及路徑長度
機台類型 1 及 3 其它另含刀具自轉的機台類型，需要第三旋轉軸 方向轉換，例如一般 6 軸轉換。方向向量旋轉。	使用 LEAD 角度**程式設計**刀具方向旋轉， 相對於表面正常向量的角度。 PO[PHI]程式設計多項式最高達五次； TILT 角度，繞路徑切線旋轉（Z 方向）； PO[PSI] 程式設計多項式最高達五次； THETA 旋轉角度（依 Z 繞刀具方向旋轉）； THETA = 到達單節結束之值； THETA=AC（...） 絕對非模態切換的至尺寸； THETA=IC（...）非模態切換至鏈結尺寸； THETA=Θ_e 插補已程式設計之角度 G90/G91； PO[THT]=（..） 程式設計多項式最高達五次。 程式設計旋轉向量 －ORIROTA 旋轉，絕對 －ORIROTR 相對旋轉向量 －ORIROTT 切線旋轉向量。
為有關路徑之方向變更所用的有關路徑之方向，或切線至路徑的旋轉向量之旋轉	在**相對於路徑**的方向中變更，使用 - ORIPATH 刀具方向相對於路徑 - ORIPATHS 亦於方向特性中的起伏 程式設計旋轉向量 - ORIROTC 切線旋轉向量，至路徑切線的旋轉

6.2 3, 4 及 5 軸轉換(TRAORI)

6.2.1 通用刀具頭的一般關係

功能

當加工表面具有三維曲度時，為了取得最理想的切削條件，必須能改變刀具的設定角度。

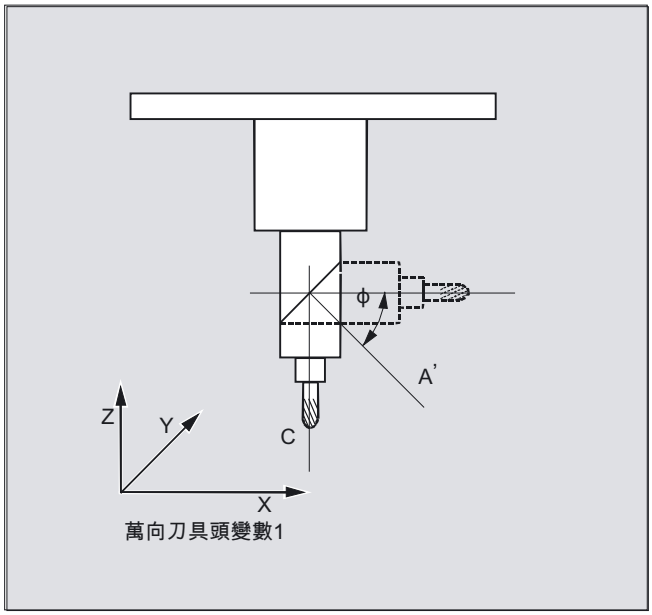


座標軸資料中儲存可達成本項目標的機台設計。

五軸轉換

萬向刀具頭

在此，由三線性軸（X、Y、Z）及兩方向軸（C、A）定義刀具的設定角度和操作點。將兩方向軸之一設為傾斜軸，在我們的範例 A'中－在許多情況下，都是置於 45°角。

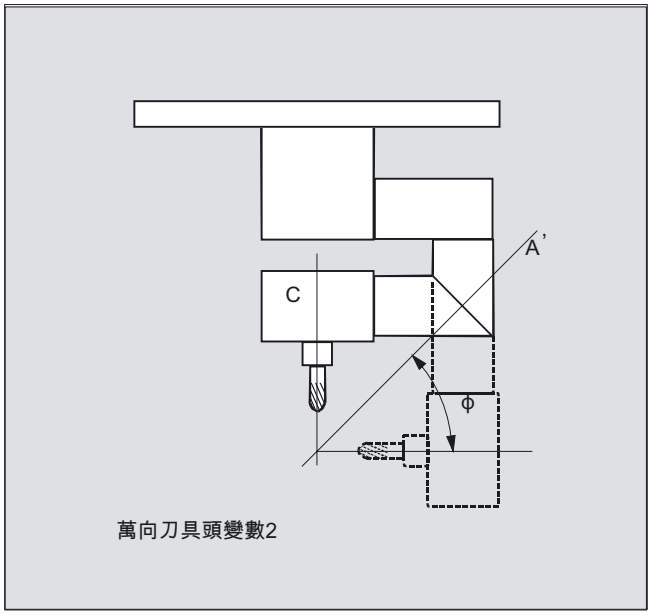


萬向刀具頭變數1

如此處範例中所示，您可看到圖示中以 CA 機台動力學配合萬向刀具頭的配置！

機台製造商

方向軸的軸順序及刀具的定位方向，可使用機械參數，視情況為機台動力學設定。



萬向刀具頭變數2

在本例中，A'位於與 X 軸成 φ 角度之下。

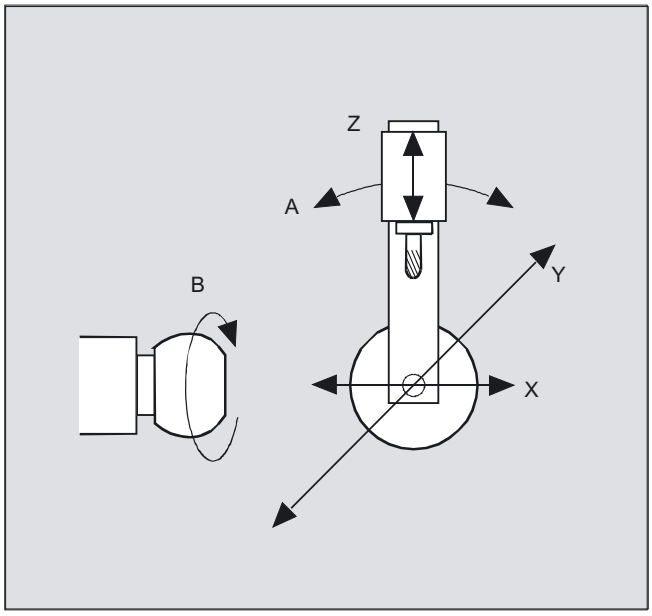
下列可能關係一般皆有效：

A'位於與 X 軸成 φ 角度之下	X 軸
B'位於與 Y 軸成 φ 角度之下	Y 軸
C'位於與 Z 軸成 φ 角度之下	Z 軸

ϕ 角度可使用機械參數設定為 0°至+89°的範圍。

使用迴轉線性軸

此為具移動工件及移動刀具的配置。其動力學包含三個線性軸（X、Y、Z）及兩個成直角配置的旋轉軸。例如，第一個旋轉軸移過兩個線性軸的複式滑座，而刀具則與第三個線性軸平行。第二個旋轉軸轉動工件。第三個線性軸（迴轉軸）則位於複式滑座平面中。



旋轉軸的軸順序及刀具的定位方向，可使用機械參數，視情況為機台動力學設定。
有下列可能關係：

軸：	**軸順序：**
1. 旋轉軸	A A B B C C
2. 旋轉軸	B C A C A B
迴轉線性軸	Z Y Z X Y X

如需有關刀具定位方向可設定之軸順序的更多詳細資訊，請參閱

參考：/FB3/ 功能手冊，特殊函數；三至五軸轉換（F2），通用銑削頭一節，“參數設定”。

6.2.2 三、四及五軸轉換（TRAORI）

函數

使用者可設定兩個或三個平移軸和一個旋轉軸。 轉換會假設旋轉軸與方向平面成直角。
僅在垂直於旋轉軸的平面中才有刀具方向。 轉換支援具可移動刀具和可移動工件的機台類型。

三及四軸轉換的設定，及程式設計的方式，與五軸轉換相同。

參考資料：
/FB3/ 功能手冊，特殊函數；三至五軸轉換（F2）

程式設計

```
TRAORI（n）
或
TRAORI（n，X，Y，Z，A，B）
或
TRAFOOF
```

參數

`TRAORI:`	啟動第一個指定的方向轉換
`TRAORI(n):`	啟動由 n 指定的方向轉換
`n:`	轉換的數目（n = 1 或 2），TRAORI（1）與方向轉換啟動相同
`X，Y，Z:`	刀具指向的方向向量元件
`A，B:`	旋轉軸的可程式設計偏移
`TRAFOOF:`	停用轉換運作

刀具方向

依為刀具所選的定位方向而定，需在 NC 程式中設定有效工作平面（G17、G18、G19），讓刀長偏移朝向刀具定位方向工作。

說明

啟用轉換時，位置資料（X、Y、Z）永遠與刀具刀尖相關。 變更涉入轉換的旋轉軸位置，會使其餘機械軸產生許多補正移動，讓刀尖位置保持不變。

方向轉換永遠從刀尖指向刀具接頭。

一般轉換之範例

刀具的基本方向如下所示：

`TRAORI（1，0，0，1）` Z 方向

`TRAORI（1，0，1，0）` Y 方向

`TRAORI（1，0，1，1）` Y/Z 方向（符合-45°位置）

方向軸的偏移

啟動方向轉換後，可為方向軸直接程式設計其他偏移。

若程式設計時使用正確的順序，則可省略參數。

範例

`TRAORI（，，，，A，B）`若只需輸入單一偏移。

作為直接程式設計的替代方案，方向軸的其他偏移，也可以從目前啟用的零點偏移自動傳輸。在機械參數中設定傳輸。

6.2.3 方向程式設計及初始設定的變數（ORIRESET）

使用 TRAORI 之刀具方向的方向程式設計

結合可程式設計 TRAORI 方向轉換時，除線性軸 X、Y、Z 外，也可使用軸識別碼 A...、B...、C...，以角度或向量元件程式設計軸位置或虛擬軸。方向軸及機械軸可有各種類型的插補。無論目前啟用何種 PO[angle]方向多項式及 PO[axis]軸多項式，皆可程式設計許多不同類型的多項式。其中包含 G1、G2、G3、CIP 或 POLY。

在某些情況下，甚至可使用方向向量程式設計刀具方向中的變更。在此情況下，可透過直接程式設計向量，或是程式設計旋轉軸位置，來設定各單節的最終方向。

說明

三至五軸轉換的方向程式設計變數

下列關於三至五軸轉換的變數：

1. A、B、C 機械軸位置的直接項目
2. A2、B2、C2 使用尤拉角或 RPY 角的虛擬軸角度程式設計
3. A3、B3、C3 向量元件的項目
4. LEAD、TILT 相對於路徑及表面的螺距與傾斜角度項目
5. A4、B4、C4 及 A5、B5、C5 單節開始與單節結束的表面法線向量
6. A6、B6、C6 和 A7、B7、C7 在錐形柱面上方向向量的插補
7. A8、B8、C8 刀具重新定向、回退移動的方向及路徑長度

相互排斥。

若嘗試程式設計混合值，則會輸出警報訊息。

初始刀具方向設定 ORIRESET

透過程式設計 ORIRESET （A，B，C），方向軸便會以線性及同步動作，由目前位置移至指定的初始設定位置。

若未程式設計軸的初始設定位置，則會使用相關的機械參數 $MC_TRAFO5_ROT_AX_OFFSET_1/2 中的定義位置。會忽略任何可能顯示的圓軸主動框架。

說明

僅在以 TRAORI（...）啟用方向轉換時，才能無視動力學，使用 ORIRESET（...）程式設計初始設定刀具方向值，而不發出警報 14101。

範例

```
1. 機械動態 CA（通道軸名稱 C、A）之範例
ORIRESET（90，45）       ; C 在 90 度、A 在 45 度
ORIRESET（，30）         ; C 在$MC_TRAFO5_ROT_AX_OFFSET_1/2[0]、A 在 30 度
ORIRESET（ ）            ; C 在$MC_TRAFO5_ROT_AX_OFFSET_1/2[0]，
                         ; A 在$MC_TRAFO5_ROT_AX_OFFSET_1/2[1]
2. 機械動態 CAC（通道軸名稱 C、A、B）之範例
ORIRESET（90，45，90）   ; C 在 90 度、A 在 45 度、B 在 90 度
ORIRESET（ ）            ; C 在$MC_TRAFO5_ROT_AX_OFFSET_1/2[0]，
                         ; A 在$MC_TRAFO5_ROT_AX_OFFSET_1/2[1]，
                         ; B 在$MC_TRAFO5_ROT_AX_OFFSET_1/2[2]
```

程式設計 LEAD、TILT 及 THETA 旋轉

關於三至五軸轉換，使用 LEAD 及 TILT 角度程式設計刀具方向的旋轉。

關於含第三個旋轉軸的轉換，為使用向量元件及含 LEAD、TILT 角度項目的兩個方向，允許 C2 的其他程式設計設定（方向向量旋轉）。

加上另外第三個旋轉軸，可以 THETA 旋轉角度，程式設計刀具自轉。

6.2.4 程式設計刀具方向（A...、B...、C...、LEAD、TILT）

功能

程式設計刀具方向時，可使用下列選項：

1. 直接程式設計旋轉軸動作。方向變更永遠發生在基本或機械座標系統中。並依照同步軸移動方向軸。
2. 尤拉或 RPY 角中的程式設計依照使用 A2、B2、C2 的角度定義
3. 使用 A3、B3、C3 程式設計方向向量，方向向量由刀具刀尖指向刀具接頭。
4. 用 A4、B4、C4 在單節開始程式設計表面法線向量，並用 A5、B5、C5 在單節結束程式設計（平面銑削）。
5. 使用螺距角度 LEAD 及傾斜角度 TILT 進行程式設計
6. 使用 A6、B6、C6 程式設計錐形旋轉軸為正規化向量，或使用 A7、B7、C7 程式設計錐形柱面上的中間方向
   請參閱“沿錐形柱面進行方向程式設計（ORIPLANE、ORICONxx）”。
7. 在回退移動期間，使用 A8、B8、C8，
   請參閱“平滑化方向特性（ORIPATHS A8=、B8=、C8=）”

說明

在所有情況下，僅在啟用方向轉換時，才可程式設計方向。
優點：可傳輸該程式至任何機械動態。

透過 G 代碼定義刀具方向

說明

機台製造商

可使用機械參數在尤拉角或 RPY 角間切換。若機械參數依此設定，則無論是否依照群組 50 的有效 G 代碼，都可進行變換。您可選擇下列設定選項：

1. 若定義方向軸及方向角度的機械參數，皆透過 G 代碼設定為零點：
   使用 A2、B2、C2 程式設計的角度皆**依照機械參數而定**；方向程式設計的角度定義則轉譯為尤拉角或 RPY 角。
2. 若透過 G 代碼定義方向軸的機械參數設定為一，則依**群組 50 的有效 G 代碼**來變換：
   使用 A2、B2、C2 程式設計的角度皆依照有效 G 代碼 ORIEULER、ORIRPY、ORIVIRT1、ORIVIRT2、ORIAXPOS 及 ORIPY2 轉譯。使用方向軸程式設計的值，亦依照群組 50 的有效 G 代碼，轉譯為方向軸。
3. **若定義方向角的機械參數設定為一，而透過 G 代碼定義方向軸的機械參數是設定為零點，則不依群組 50 的有效 G 代碼：**
   使用 A2、B2、C2 程式設計的角度皆依照有效 G 代碼 ORIEULER、ORIRPY、ORIVIRT1、ORIVIRT2、ORIAXPOS 及 ORIPY2 其中之一轉譯。使用方向軸程式設計的值則會轉譯為圓軸位置，與群組 50 的有效 G 代碼無關。

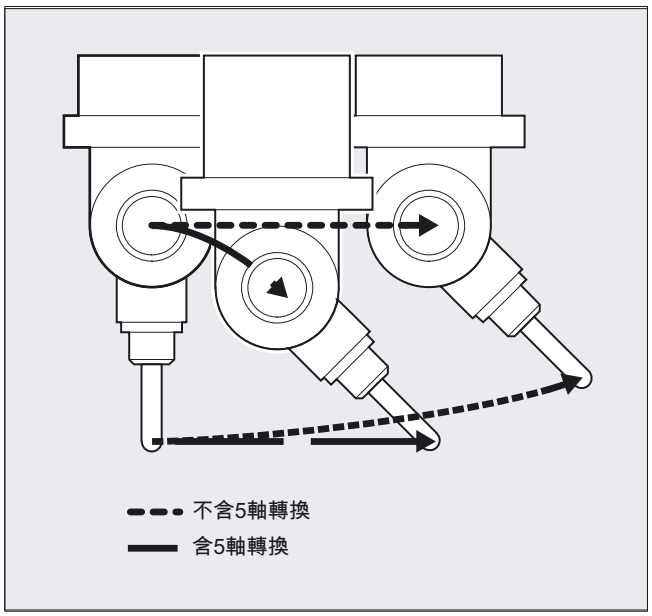
程式設計

`G1 X Y Z A B C`	旋轉軸動作的程式設計
`G1 X Y Z A2= B2= C2=`	於尤拉角中程式設計
`G1 X Y Z A3== B3== C3==`	方向性向量的程式設計
`G1 X Y Z A4== B4== C4==`	在單節起點程式設計表面法線向量
`G1 X Y Z A5== B5== C5==`	在單節結束程式設計表面法線向量
`LEAD=`	程式設計刀具方向的螺距角度
`TILT=`	程式設計刀具方向的傾斜角度

參數

`G....`	旋轉軸動作的細節
`X Y Z`	線性軸的細節
`A B C`	旋轉軸的機械軸位置細節
`A2 B2 C2`	虛擬軸或方向軸的角度程式設計（尤拉角或 RPY 角）
`A3 B3 C3`	方向向量元件的細節
`A4 B4 C4`	例如，平面銑削的細節，在單節起點的表面法線向量的元件細節
`A5 B5 C5`	例如，平面銑削的細節，在單節結束的表面法線向量的元件細節
`LEAD`	在平面中，由路徑切線及表面法線向量提供，相對於表面法線向量的角度
`TILT`	平面中的角度，和相對於表面法線向量的路徑切線垂直

範例：有與沒有和五軸轉換的比較



說明

五軸程式一般是由 CAD/CAM 系統產生，而非在控制系統中輸入。故下列說明，主要是針對後處理器的程式設計人員。

在 G 代碼群組 50 中，定義方向程式設計的類型：

ORIEULER 透過尤拉角
ORIRPY 透過 RPY 角（旋轉順序 ZYX）
ORIVIRT1 透過虛擬方向軸（定義 1）
ORIVIRT2 透過虛擬方向軸（定義 2）
ORIAXPOS 以圓軸位置透過虛擬軸
ORIPY2 透過 RPY 角（旋轉順序 XYZ）

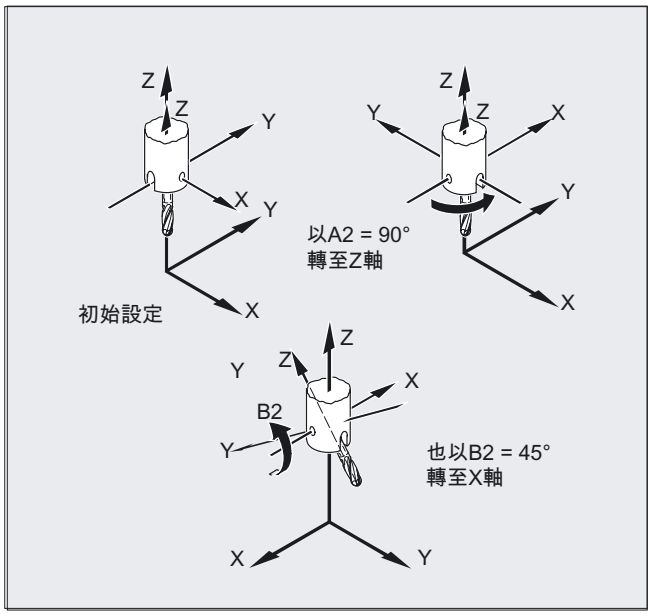
機台製造商

機台製造商可使用機械參數來定義各種變數。請參閱機台製造商指示。

於尤拉角程式設計 ORIEULER

在使用 A2、B2、C2 方向程式設計期間，轉譯程式設計的值為尤拉角（以度為單位）。

首先以 A2 繞 Z 軸，依 Z 方向轉動向量，然後以 B2 繞新 X 軸轉動，最後以 C2 繞新 Z 軸轉動，而得出方向向量。



在此 C2 值的情況（繞新 Z 軸旋轉）毫無意義，且無需程式設計。

於 RPY 角程式設計 ORIRPY

在使用 A2、B2、C2 方向程式設計期間，轉譯程式設計的值為 RPY 角（以度為單位）。

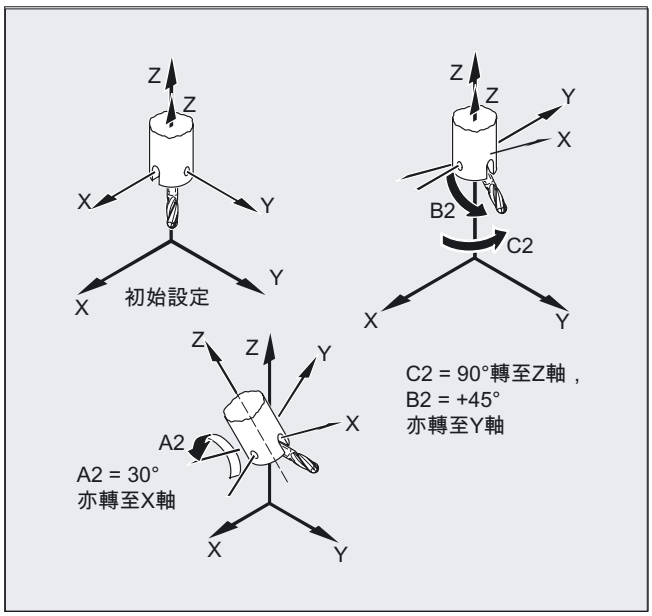
說明

相較於尤拉角程式設計，在此三個值皆會影響方向向量。

機台製造商

透過 RPY 角定義含方向角度之角度時，對方向軸$MC_ORI_DEF_WITH_G_CODE = 0

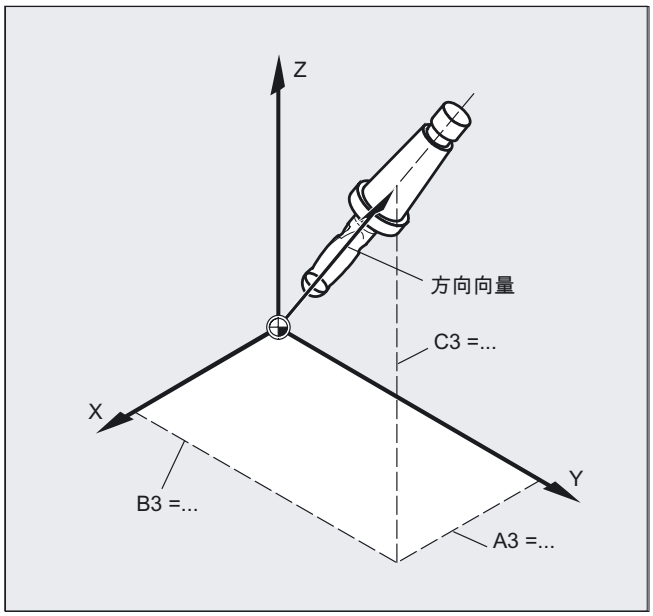
首先以 C2 繞 Z 軸，依 Z 方向轉動向量，然後以 B2 繞新 Y 軸轉動，最後以 A2 繞新 X 軸轉動，而取得方向向量。



透過 G 代碼定義方向軸時，若機械參數為$MC_ORI_DEF_WITH_G_CODE = 1，則：
首先以 A2 繞 Z 軸，依 Z 方向轉動向量，然後以 B2 繞新 X 軸轉動，最後以 C2 繞新 Z 軸轉動，而取得方向向量。

方向性向量的程式設計

以 A3、B3、C3 程式設計方向向量的元件。向量指向刀具轉接頭，向量的長度不重要。
未程式設計的向量元件，經設定等於零點。



以 LEAD=及 TILT=程式設計刀具方向

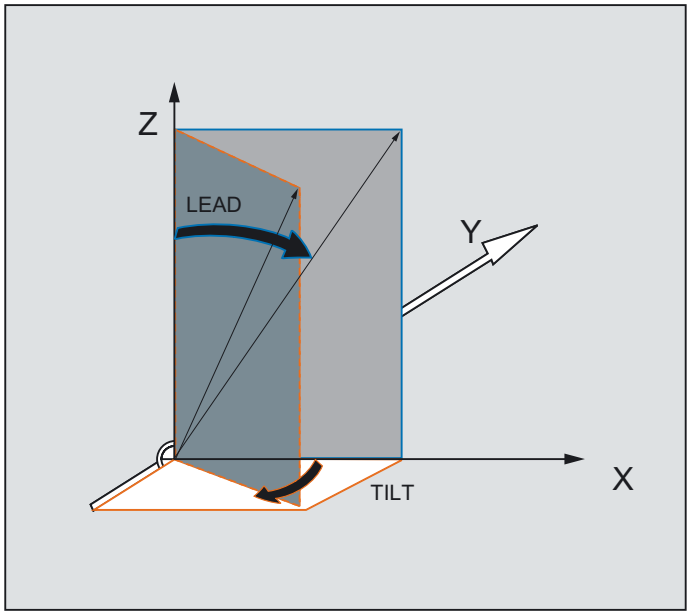
由下列各項決定出刀具方向：

- 路徑切線
- 在單節起點的
表面法線向量 `A4`、`B4`、`C4` 及在單節結尾的 `A5`、`B6`、`C5`
- 由路徑切線及表面法線向量，定義平面中的螺距角度 `LEAD`
- 在單節結束的傾斜角度 `TILT`
與路徑切線垂直，並與表面法線向量相對

內角行為（用於 3D 刀具補正）

若在內角處縮短，則最終刀具方向亦會於單節結尾處達成。

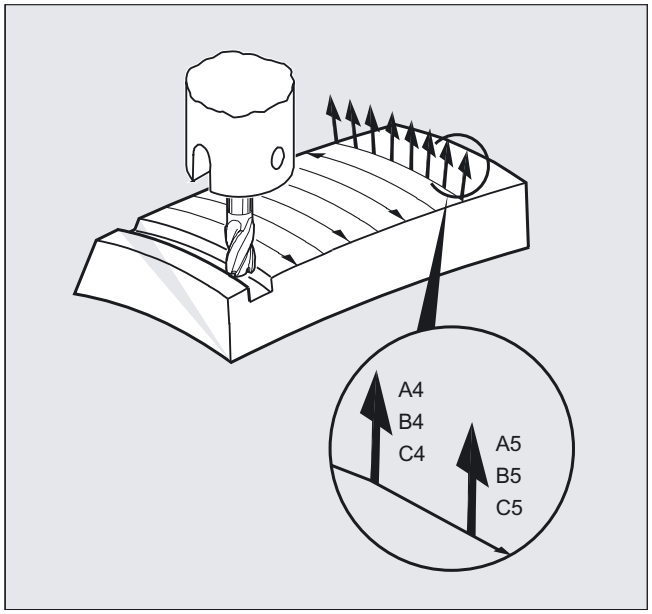
以 LEAD=及 TILT=定義刀具方向



6.2.5 平面銑削（3D 銑削 A4、B4、C4、A5、B5、C5）

功能

平面銑削用於所有加工曲面。



該種 3D 銑削，需在工件表面上逐行定義 3D 路徑。

計算需考慮刀具形狀及規格，其一般是在 CAM 中執行計算；然後再將計算完成的 NC 單節，透過後處理器讀入控制系統中。

程式設計路徑曲率

表面說明

以下列元件由表面法線向量說明路徑曲率：

A4、B4、C4 在單節起點的起始向量

A5、B5、C5 在單節結尾的結束向量

若單節僅含起始向量，則單節中所有表面法線向量將維持恆定。若單節僅含結束向量，將透過大半徑圓弧插補，從前一單節的結束值到已程式設計的結束值執行插補。

若起始及結束向量皆以程式設計，則亦透過大半徑圓弧插補，在兩個方向間執行插補。如此可持續平滑化產生路徑。

不管有效的 G17 到 G19 層級為何，在初始設定中，表面法線向量指向 Z 方向。

向量的長度毫無意義。

設定未已程式設計的向量元件為零點。

使用有效 ORIWKS（請參閱"方向軸參考（ORIWKS、ORIMKS）"），表面法線向量與主動框架相關，並在框架旋轉時旋轉。

機台製造商

表面法線向量需需垂直於路徑切線，透過機械參數設定在臨界值內，否則將輸出警報。

6.2.6 方向軸參考（ORIWKS、ORIMKS）

功能

對於在工件座標系統中程式設計的方向時，當使用

- 尤拉角或 RPY 角，或
- 方向向量

可使用 ORIMKS/ORIWKS 來設定旋轉動作的課程。

說明

機台製造商

以機械參數指定的方向之插補類型：

MD21104 $MC_ORI_IPO_WITH_G_CODE

= FALSE：參考點由 G 系列功能 ORIWKS 與 ORIMKS 提供。

= TRUE：參考點由 51 st 群組（ORIAXES，ORIVECT，ORIPLANE，等等）中的 G 系列功能提供。

句法

```
ORIMKS=...
ORIWKS=...
```

意義

ORIMKS	在機械座標系統中旋轉
ORIWKS	在工件座標系統中旋轉

說明

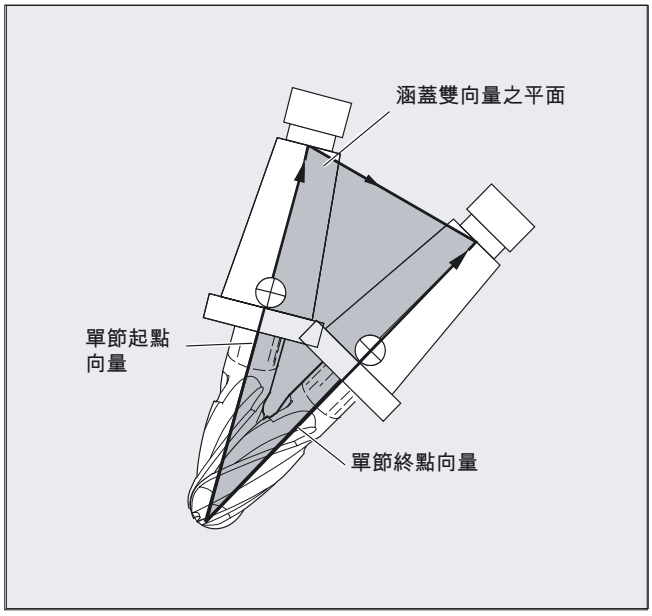
ORIWKS 為基本設定。在五軸程式的情況下，若欲執行的機台未立即明確，則必需選擇 ORIWKS。機台實際執行的動作依照機械動態而定。

ORIMKS 可被用於程式設計實際的機台移動（用來避免和設備碰撞或類似情形，例如）。

說明

使用 ORIMKS 時，刀具執行動作**依照**機械動態。在空間中固定點刀具刀尖方向變更的情況下，會在旋轉軸位置間進行線性插補。

使用 ORIWKS 時，刀具執行動作**不依照**機械動態。含固定刀具刀尖的方向變更，在依起始及結束向量設定的平面中移動刀具。



單一位置

說明

ORIWKS

五軸機台單一設定區中的方向動作需要大量機械軸動作（例如，以 C 為旋轉軸，而以 A 為迴轉軸的旋轉迴轉頭，具 A = 0 的所有位置皆為單一位置）。

機台製造商

速率控制會大幅降低接近單一位置的刀具路徑速率，以避免機械軸過載。

使用以下機械參數時

$MC_TRAFO5_NON_POLE_LIMIT

$MC_TRAFO5_POLE_LIMIT

可參數化轉換，讓接近極點的方向動作穿透極點，且能迅速加工。

只能使用 MD $MC_TRAFO5_POLE_LIMIT 處理單一位置。

參考資料：
/FB3/ 功能手冊，特殊函數：三至五軸轉換（F2），
“單一點及其處理方式”一節。

6.2.7 程式設計方向軸（ORIAXES，ORIVECT，ORIEULER，ORIRPY，ORIRPY2，ORIVIRT1，ORIVIRT2）

功能

方向軸函數定義刀具在空間中的方向，並經由程式設計旋轉軸的偏移達成。也可讓刀具本身旋轉，而達成額外的第三度自由度。在此情況下，透過第三個旋轉軸在空間定向刀具，該動作需六軸轉換。刀具旋轉是使用與旋轉向量的插補類型一致的 THETA 旋轉角度來定義（請參閱“刀具方向的旋轉（ORIROTA/TR/TT、ORIROTC、THETA）”一節）。

程式設計

使用軸識別碼 A2、B2 及 C2 程式設計方向軸。

`N... ORIAXES` 或 `ORIVECT`	線性或大半徑圓弧插補
`N... G1 X Y Z A B C`	
或	或
`N... ORIPLANE`	平面的方向插補
或	或
`N... ORIEULER` 或 `ORIRPY/ORIRPY2`	方向角度尤拉 / RPY 角
`N... G1 X Y Z A2= B2= C2=`	虛擬軸的角度程式設計
或	或
`N... ORIVIRT1` 或 `ORIVIRT2`	虛擬方向軸的定義 1 或 2 方向向量程式設計
`N... G1 X Y Z A3= B3= C3=`	

可沿空間中錐形柱面，為方向變更程式設計方向軸的其他旋轉軸偏移；請參閱“沿錐形柱面進行方向程式設計（ORIPLANE、ORICONxx）”。

參數

ORIAXES	機台或方向軸的線性插補
ORIVECT	大半徑圓弧插補（與 ORIPLANE 完全相同）
ORIMKS	在機械座標系統中旋轉
ORIWKS	在工件座標系統中旋轉
	說明，請參閱“刀具方向的旋轉”一節
A= B= C=	程式設計機械軸位置
ORIEULER	透過尤拉角進行方向程式設計
ORIRPY	透過 RPY 角進行方向程式設計旋轉順序為 XYZ，且： A2 為繞 X 的旋轉角度 B2 為繞 Y 的旋轉角度 C2 為繞 Z 的旋轉角度
ORIRPY2	透過 RPY 角進行方向程式設計旋轉順序為 ZYX，且： A2 為繞 Z 的旋轉角度 B2 為繞 Y 的旋轉角度 C2 為繞 X 的旋轉角度
A2= B2= C2=	虛擬軸的角度程式設計
ORIVIRT1	使用虛擬方向軸的方向程式設計
ORIVIRT2	（定義 1），根據 MD $MC_ORIAX_TURN_TAB_1 的定義
	（定義 2），根據 MD $MC_ORIAX_TURN_TAB_2 的定義
A3= B3= C3=	方向軸的方向向量程式設計

說明

機台製造商

使用 MD $MC_ORI_DEF_WITH_G_CODE 指定已程式設計的角度 A2、B2、C2 的定義方式：

根據 MD $MC_ORIENTATION_IS_EULER（標準），或根據 G 群組 50（ORIEULER、ORIRPY、ORIVIRT1、ORIVIRT2）來定義。

使用 MD $MC_ORI_IPO_WITH_G_CODE 來定義有效的插補模式類型：ORIWKS/ORIMKS 或 ORIAXES/ORIVECT。

寸動進給（JOG）模式

在本操作模式中的方向角度插補永遠為線性。在透過移動鍵持續及增量移動期間，僅可移動一個方向軸。可同時使用手輪移動方向軸。

使用方向軸的手動移動時，以快速移動手動倍率通道專屬進給手動倍率開關，或快速移動手動倍率開關。

可用下列機械參數進行獨立的速率設定：

$MC_JOG_VELO_RAPID_GEO

$MC_JOG_VELO_GEO

$MC_JOG_VELO_RAPID_ORI

$MC_JOG_VELO_ORI

說明

SINUMERIK 840D 與"處理轉換套件"

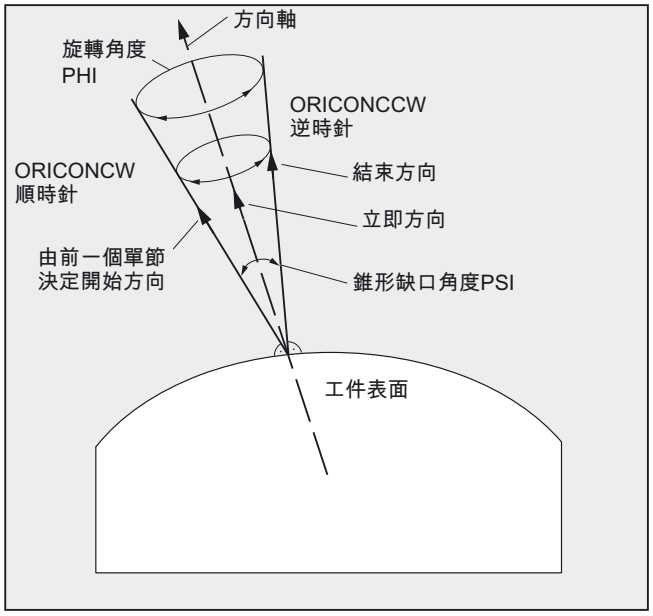
在寸動進給模式中使用"直角座標手動移動"功能，另外可由參考系統－機械座標系統（MCS）、工件座標系統（WCS）及 TCS 中，互相設定幾何軸的轉譯。

參考資料：
/FB2/ 函數延伸功能的說明；動態轉換（M1）

6.2.8 沿著錐形的柱面進行方向程式設計（ORIPLANE，ORICONCW，ORICONCCW，ORICONTO，ORICONIO）

函數

使用延伸方向，可沿空間中的錐形柱面執行方向變更。 使用 ORICONxx 模態指令，在錐形柱面上插補方向向量。 可在平面上，使用插補的 ORIPLANE 程式設計結束方向。 通常由前一個單節定義起始方向。



程式設計

使用 A2、B2、C2，在尤拉角或 RPY 角中指定角度程式設計，或使用 A、B、C 程式設計旋轉軸位置來定義結束方向。沿錐形柱面的方向軸需要進一步的程式設計細節：

- 將錐形旋轉軸做為使用 A6、B6、C6 的向量
- 使用識別碼 NUT 之缺口角度 PSI
- 使用 A7、B7、C7 之中間方向錐形外部

說明

為錐形的旋轉軸程式設計方向向量 A6、B6、C6

結束方向的程式設計絕非必要； 若無指定結束方向，則會插補 360 度的完整外錐形。

用 NUT=angle 程式設計錐形缺口角度

需指定結束方向。

無法以此方式插補 360 度的完整外錐形。

在錐形外部程式設計中間方向 A7、B7、C7

需指定結束方向。 僅可透過三個向量：“起始”方向、“結束”方向及“中間”方向，各自定義變更方向及旋轉方向。 三個向量需為不同。 若已程式設計的中間方向與起始或結束方向平行，便在由起始及結束向量定義的平面中，執行方向的線性大半徑圓弧插補。

錐形柱面上的延伸方向插補

`N... ORICONCW` 或 `ORICONCCW`	在錐形外部的**插補**，使用
`N... A6= B6= C6= A3= B3= C3=`	錐形及結束方向的順時針 / 逆時針方向中的方向向量，或
或	
`N... ORICONTO`	結束方向的
`N... G1 X Y Z A6= B6= C6=`	切線變化及規格
或	
`N... ORICONIO`	錐形外部的結束方向及
`N... G1 X Y Z A7= B7= C7=`	中間方向規格，使用
`N... PO[PHI]=（a2，a3，a4，a5）`	旋轉角度的多項式及
`N... PO[PSI]=（b2，b3，b4，b5）`	缺口角度的多項式

參數

ORIPLANE	平面中插補（大半徑圓弧插補）
ORICONCW	錐形柱面上順時針方向插補
ORICONCCW	錐形柱面上逆時針方向插補
ORICONTO	錐形柱面上的插補含切線變化
A6= B6= C6=	錐形旋轉軸的程式設計（正規化向量）
NUT=angle	錐形的缺口角度，以度為單位
NUT=+179	移動角度小於或等於 180 度
NUT=-181	移動角度大於或等於 180 度
ORICONIO	錐形柱面上的插補
A7= B7= C7=	中間方向（程式設計為正規化向量）
PHI	繞錐形方向軸的方向旋轉角度
PSI	錐形缺口角度
Possible polynomials PO[PHI]=（a2，a3，a4，a5） PO[PSI]=（b2，b3，b4，b5）	除不同角度外，也可程式設計多項式最高達五次

不同方向變更之範例

```
...
N10 G1 X0 Y0 F5000
N20 TRAORI（1）                                  ; 開啟方向轉換。
N30 ORIVECT                                      ; 插補刀具方向為向量。
...                                              ; 平面中刀具方向。
N40 ORIPLANE                                     ; 選擇大半徑圓弧插補。
N50 A3=0 B3=0 C3=1
N60 A3=0 B3=1 C3=1                               ; Y/Z 平面中的方向旋轉超過 45 度，在單節結束時已
                                                   達方向（0，1，2，1/√2，1/√2）。
...
N70 ORICONCW                                     ; 在錐形外部進行方向程式設計：
N80 A6=0 B6=0 C6=1 A3=0 B3=0 C3=1                ; 方向向量是以從（0，0，1）直到（1/√2，0，
                                                   1/√2）的順時針方向插補在錐形的外部，旋轉角度為
                                                   270 度。
N90 A6=0 B6=0 C6=1                               ; 刀具方向在相同的錐形外部通過完整的轉數。
```

說明

若要在空間任何一處，沿錐形柱面說明變更方向，需先知道欲旋轉刀具方向的向量。亦需指定起始及結束方向。從前一個單節產生起始方向，並透過其他條件程式設計或定義結束方向。

在與 ORIVECT 相同的 ORIPLANE 平面中程式設計

大半徑圓弧插補的程式設計，與角度多項式一同相當於輪廓的線性及多項式插補。在由起始及結束方向定義的平面中插補刀具方向。若程式設計了其他多項式，則方向向量也可傾斜至平面外。

在平面 G2/G3、CIP 及 CT 中圓弧的程式設計

與平面中圓弧插補相符合的延伸方向。關於具中心或半徑（如 G2/G3）的圓弧、經過中間點 CIP 的圓弧、及切線圓弧 CT 的對應程式設計選項，請參閱

參考資料：程式設計手冊基礎“程式設計動作指令”。

方向程式設計

錐形柱面上，方向向量的插補 ORICONxx

可從 G 代碼群組 51 選擇四種不同類型插補，在錐形柱面上插補方向：

1. 以結束方向及錐形方向或缺口角度的規格，依順時針方向在錐形外部插補 ORICONCW。以識別碼 A6、B6、C6 程式設計方向向量，並以識別碼 NUT=值範圍，間隔 0 度到 180 度，程式設計錐形缺口角度。
2. 以結束方向及錐形方向或缺口角度的規格，依逆時針方向在錐形外部插補 ORICONCCW。以識別碼 A6、B6、C6 程式設計方向向量，並以識別碼 NUT=值範圍，間隔 0 度到 180 度，程式設計錐形缺口角度。
3. 以識別碼 A7、B7、C7 程式設計的結束及中間方向規格，在錐形外部插補 C7ORICONIO。
4. 以切線變化及結束方向的規格，在錐形外部插補 ORICONTO。以識別碼 A6、B6、C6 程式設計方向向量。

6.2.9 兩接點的方向規格（ORICURVE、PO[XH]=、PO[YH]=、PO[ZH]=）

功能

使用空間中第二個曲度程式設計方向變更 ORICURVE

除使用刀尖沿空間中曲度外，另一個程式設計方向變更的方式，為使用 ORICURVE 程式設計第二個接點的動作。在此方式中，僅可定義刀具方向變更，與程式設計刀具本身向量時一樣。

機台製造商

關於可以透過機械參數，設定為程式設計刀具第二個方向路徑的座標軸識別碼，請參閱機台製造商的說明。

程式設計

可使用該類型插補程式設計空間中兩曲度的點（使用 G1）或多項式（使用 POLY）。不得使用圓弧與漸開線。也可啟動 BSPLINE 曲線插補及“結合短曲線單節”函數。

參考資料：
/FB1 / 功能手冊，基本功能；連續路徑模式，精確停止，預見控制（B1），章節：結合短曲線單節

不得使用其他曲線類型 ASPLINE 及 CSPLINE，以及使用 COMPON、COMPCURV 或 COMPCAD 啟動壓縮機。

程式設計座標的方向多項式時，可預先定義刀具的兩接點動作最高達五次。

具空間中其他曲度及座標多項式的延伸方向插補

```
N... ORICURVE                          刀具的第二個接點動作規格，及該座標的其他多項式
N... PO[XH]=（xe，x2，x3，x4，x5）
N... PO[YH]=（ye，y2，y3，y4，y5）
N... PO[ZH]=（ze，z2，z3，z4，z5）
```

參數

ORICURVE	指定刀具兩接點間動作的方向插補
XH YH ZH	做為空間中曲度之其他輪廓的第二個刀具接點之座標識別碼
Possible polynomials PO[XH]=（xe，x2，x3，x4，x5） PO[YH]=（ye，y2，y3，y4，y5） PO[ZH]=（ze，z2，z3，z4，z5）	除使用適當的終點外，也可使用多項式程式設計空間中曲度。
xe，ye，ze	空間中曲度的終點
xi，yi，zi	多項式的係數最高達五次

說明

程式設計第二個方向路徑的識別碼 XH YH ZH

識別碼的需經選擇，才不會與其他識別碼或線性軸產生衝突

X Y Z 軸

及旋轉軸，例如

A2 B2 C2 尤拉角或 RPY 角

A3 B3 C3 方向向量

A4 B4 C4 或 A5 B5 C5 表面法線向量

A6 B6 C6 旋轉向量或 A7 B7 C7 中間點座標

或其他插補參數。

6.3 方向多項式（PO[角度]、PO[座標]）

功能

與目前啟用的 G 代碼群組 1 的多項式插補無關，可為三軸或至五軸轉換程式設計兩個不同類型的方向多項式最高達五次。

1. **角度**的多項式：螺距角度 LEAD、傾斜角度 TILT
   與以起始及結束方向所定義平面的關係。
2. **座標**的多項式：刀具上參考點的刀具方向之空間中第二個曲度的 XH、YH、ZH。

有了六軸轉換，除刀具方向外，可為刀具本身旋轉，以最高達五次的多項式，程式設計旋轉向量 THT 之旋轉。

句法

類型 1 **角度**的方向多項式

```
N… PO[PHI]=（a2，a3，a4，a5）          三軸至五軸轉換
N… PO[PSI]=（b2，b3，b4，b5）
                                       三軸至五軸轉換
```

類型 2 **座標**的方向多項式

```
N… PO[XH]=（xe，x2，x3，x4，x5）       刀具方向之第二個方向路徑的座標識別碼
N… PO[YH]=（ye，y2，y3，y4，y5）
N… PO[ZH]=（ze，z2，z3，z4，z5）
```

在這兩種情況下，有了六軸轉換，也可為**旋轉**使用方向向量程式設計一多項式

```
N… PO[THT]=（c2，c3，c4，c5）          相對於路徑的旋轉插補
或
N… PO[THT]=（d2，d3，d4，d5）          方向變更的絕對、相對及切線插補
```

方向向量若轉換支援含可程式設計偏移的旋轉向量，並使用 THETA 旋轉角度進行插補，即可進行上述作業。

含義

PO[PHI]	平面中起始與結束方向間的角度
PO[PSI]	定義由平面起始與結束方向間方向角度的傾斜
PO[THT]	旋轉的角度，是由旋轉群組 54 的 G 代碼之一的旋轉向量而產生，而該代碼是用 THETA 所程式設計。
PHI	螺距角度 LEAD
PSI	傾斜角度 TILT
THETA	依 Z 繞刀具方向的旋轉
PO[XH]	刀具上參考點的 X 座標
PO[YH]	刀具上參考點的 Y 座標
PO[ZH]	刀具上參考點的 Z 座標

說明

以下情況無法程式設計方向多項式：

- 若已啟用 ASPLINE、BSPLINE、CSPLINE 曲線插補。
  類型 1 方向角度的多項式可用於曲線外所有類型的插補，即，含快速移動 G00 或進給率 G01 的線性插補
  與使用 POLY 及
  圓弧 / 漸開線插補 G02、G03、CIP、CT、INVCW 及 INCCCW 的多項式插補
  。
  然而，類型 2 方向座標的多項式僅在若
  含快速移動 G00 或進給率 G01 的線性插補，或
  已啟用 POLY 的多項式插補下使用。
- 若使用 ORIAXES 軸插補來插補方向。在此情況下，可直接使用方向軸 A 及 B 的 PO[A]及 PO[B]程式設計多項式。

含 ORIVECT、ORIPLANE 及 ORICONxx 類型 1 方向多項式

僅類型 1 方向多項式可用於大半徑圓弧插補，及以 ORIVECT、ORIPLANE 及 ORICONxx 在錐形外部插補。

含 ORICURVE 類型 2 方向多項式

若啟用具空間中其他曲度的插補 ORICURVE，則會插補方向向量的直角座標元件，且僅能使用類型 2 方向多項式。

6.4 刀具方向的旋轉（ORIROTA、ORIROTR、ORIROTT、ORIROTC、THETA）

功能

若您亦欲在具有可移動刀具的機台類型上變更刀具方向，請程式設計含結束方向的每個單節。依照機械動態，您可程式設計方向軸的定位方向，或方向向量的旋轉方向 THETA。可為下列旋轉向量程式設計不同插補類型：

- ORIROTA：至絕對旋轉方向的旋轉角度。
- ORIROTR：相對於平面起始與結束方向間的旋轉角度。
- ORIROTT：相對於方向向量中變更的旋轉角度。
- ORIROTC：至路徑切線的旋轉切線角度。

句法

僅在若啟用插補類型 ORIROTA 時，才可在以下四個模式中，程式設計旋轉角度或旋轉向量：

1. 直接做為旋轉軸位置 A、B、C
2. 使用 A2、B2、C2 之尤拉角（以度為單位）
3. 使用 A2、B2、C2 之 RPY 角（以度為單位）
4. 透過 A3、B3、C3 的方向向量（使用 THETA=value 的旋轉角度）

若已啟用 ORIROTR 或 ORIROTT，僅能直接使用 THETA 程式設計旋轉角度。

旋轉也可在未發生方向變更的獨立單節中，進行程式設計。在此情況下，ORIROTR 及 ORIROTT 不相關。在此情況下，皆使用絕對方向的參考轉譯旋轉角度（ORIROTA）。

```
N... ORIROTA                          定義旋轉向量的插補
N... ORIROTR
N... ORIROTT
N... ORIROTC
N... A3= B3= C3= THETA=value          定義方向向量的旋轉
N... PO[THT]=（d2，d3，d4，d5）        使用五次多項式的旋轉插補角度
```

意義

ORIROTA	絕對旋轉方向的旋轉角度。
ORIROTR	相對於平面起始與結束方向間的旋轉角度。
ORIROTT	做為方向變更之切線旋轉向量的旋轉角度
ORIROTC	做為路徑切線之切線旋轉向量的旋轉角度
THETA	方向向量的旋轉
THETA=value	單節結尾達成的旋轉角度度數
THETA=Θe	含旋轉向量結束角度 Θ_e 的旋轉角度
THETA=AC（...）	絕對尺寸的非模態切換
THETA=AC（…）	增量尺寸的非模態切換
Θe	啟用含 G90 絕對及 G91 相對（增量尺寸標註）旋轉向量的結束角度
PO[THT]=（....）	旋轉角度的多項式

方向旋轉之範例

程式碼	註解
N10 TRAORI	; 啟動方向轉換
N20 G1 X0 Y0 Z0 F5000	; 刀具方向
N30 A3=0 B3=0 C3=1 THETA=0	; 以旋轉角度 0，依 Z 方向旋轉
N40 A3=1 B3=0 C3=0 THETA=90	; 依 X 方向，並旋轉約 90 度
N50 A3=0 B3=1 C3=0 PO[THT]=（180, 90）	; 方向
N60 A3=0 B3=1 C3=0 THETA=IC（-90）	; 依 Y 方向，並旋轉約 180 度
N70 ORIROTT	; 維持恆定，並旋轉至 90 度
N80 A3=1 B3=0 C3=0 THETA=30	; 相對於方向變更的旋轉角度
	; 以 30 度旋轉向量至 X/Y 平面

插補單節
N40 時，以線性插補由初始值 0 度至最終值 90 度的旋轉角度。在單節 N50 中，根據拋物線 $\theta(u) = +90u^2$，將旋轉角度由 90 度變更為 180 度。在 N60 中，也可在未發生方向變更下執行旋轉。

在 N80 中，由 Y 方向朝 X 方向旋轉刀具方向。在 X/Y 平面中進行方向變更，而旋轉向量定義與此平面成 30 度之角度。

說明

ORIROTA

使用空間中絕對方向的參考，插補旋轉角度 THETA。基本旋轉方向在機械參數中定義。

ORIROTR

相對於由起始及結束方向定義的平面，轉譯旋轉角度 THETA。

ORIROTT

相對於方向變更，轉譯旋轉角度 THETA。當 THETA=0 時，會切線插補旋轉向量至方向變更，且僅在若為方向的“傾斜角度 PSI”至少程式設計一個多項式時，才與 ORIROTR 不同；結果為在平面中並未執行的方向變更。然後可使用旋轉 THETA 的其他角度來插補旋轉向量，如此將產生參考方向變更的特定值。

ORIROTC

旋轉向量係以用 THETA 角度程式設計的偏移，插補到相對的路徑。多項式 PO[THT]=（c2，c3，c4，c5）也可為偏移角度，程式設計最高達五次。

6.5 相對於路徑的方向

6.5.1 相對於路徑的方向類型

功能

使用此展開函數，相對方向不僅在單節結尾處，且會跨越整個軌道而達成。使用大半徑圓弧插補，將在前一個單節中達成的方向傳輸至程式設計結束方向。基本上，有兩種方式可以程式設計相對於路徑所要方向：

1. 如同刀具旋轉，使用 ORIPATH、ORPATHTS 插補相對於路徑的刀具方向。
2. 以一般方式程式設計及插補方向向量。使用 ORIROTC，開始相對於路徑切線的方向向量旋轉。

句法

使用用下列句法程式設計方向的插補類型及刀具的旋轉：

`N... ORIPATH`	相對於路徑的方向
`N... ORIPATHS`	相對於具方向特性的平滑化路徑之方向
`N... ORIROTC`	相對於路徑之旋轉向量的插補

可使用 `ORIPATHS` 來平滑化在軌道上由轉角造成的方向起伏。使用元件 `A8=X`、`B8=Y`、`C8=Z`，透過向量程式設計回退移動的方向及路徑長度。

可使用 `ORIPATH/ORIPATHS` 透過三個角度，為整個軌道程式設計各種路徑切線的參考。

- `LEAD=`相對於路徑及表面的螺距角度規格
- `TILT=`相對於路徑及表面的傾斜角度規格
- `THETA=`旋轉角度

用於整個軌道。除使用旋轉角度 `THETA` 外，也可使用 `PO[THT]=（...）`程式設計最高達五次的多項式。

說明

機台製造商

請參閱機台製造商指示。透過可設定的機台及設定參數，產生其他相對於路徑的方向設定。如需更多詳細資訊，請參考

參考資料：
/FB3 / 功能手冊，特殊函數：三至五軸轉換（F2），
“方向”一章

意義

可透過機械參數產生角度 LEAD 及 TILT 插補的各種設定：

- 使用 LEAD 及 TILT 將程式設計的刀具方向參考保留在整個單節中。
- 螺距角度 LEAD：繞垂直於切線及法線向量 TILT 的方向旋轉：繞法線向量的方向旋轉
- 螺距角度 LEAD：繞垂直於切線及法線向量“傾斜”角度 TILT 的方向旋轉：依路徑切線方向的方向旋轉。
- 旋轉角度 THETA：以另外第三個旋轉軸做為六軸轉換中的方向軸，繞刀具本身旋轉。

說明

相對於路徑的方向不得搭配 OSC、OSS、OSSE、OSD 及 OST

不得搭配以群組 34 的 G 代碼方向特性平滑化，程式設計相對於路徑的方向插補，即 ORIPATH 或 ORIPATHS 及 ORIOTC。此動作需啟用 OSOF。

6.5.2 相對於路徑的刀具方向旋轉（ORIPATH、ORIPATHS、旋轉角度）

函數

使用六軸轉換時，以第三個旋轉軸在空間中，依所需的定向刀具，可使刀具繞本身旋轉。 使用 ORIPATH 或 ORIPATHS 相對於路徑的刀具方向旋轉時，可透過 THETA 旋轉角度程式設計其他的旋轉。 或者可使用位於與刀具方向垂直的平面向量，程式設計 LEAD 及 TILT 角度。

機台製造商

請參閱機台製造商指示。 可使用機械參數將 LEAD 及 TILT 角度的插補設定為不同插補。

句法

刀具方向及刀具的旋轉

使用 ORIPATH 或 ORIPATHS，啟動相對於路徑的刀具方向類型。

N... ORIPATH	啟動相對於路徑的方向類型
N... ORIPATHS	啟動含方向特性的平滑化，相對於路徑的方向類型
啟動可旋轉的三個角度：	
N... LEAD=	相對於表面法線向量，已程式設計的方向角度
N... TILT=	與相對於表面法線向量的路徑切線垂直，且在平面中已程式設計的方向角度
N... THETA=	在第三個旋轉軸的刀具方向中，相對於方向變更的旋轉角度

在單節結尾使用 LEAD=value、TILT=value 或 THETA=value 程式設計角度值。 除恆定角外，皆可為三個角度程式設計多項式最高達五次。

N... PO[PHI]=（a2，a3，a4，a5）	螺距角度 LEAD 的多項式
N... PO[PSI]=（b2，b3，b4，b5）	傾斜角度 TILT 的多項式
N... PO[THT]=（d2，d3，d4，d5）	旋轉角度 THETA 的多項式

較高的多項式係數為零點時，則可在程式設計時省略。 範例： PO[PHI]=a2 會產生 LEAD 角度的拋物線。

意義

相對於路徑的刀具方向

ORIPATH	與路徑相關的刀具方向
ORIPATHS	相對於路徑的刀具方向；已平滑化方向特性的起伏
LEAD	由路徑切線及表面法線向量定義的平面中，相對於表面法線向量的角度
TILT	依 Z 方向的方向，或繞路徑切線旋轉
THETA	朝 Z 繞刀具方向旋轉
PO[PHI]	LEAD 角度的方向多項式
PO[PSI]	TILT 角度的方向多項式
PO[THT]	THETA 旋轉角度的方向多項式

說明

旋轉角度 THETA

需有六軸轉換，才能用以繞自身方向軸的第三個旋轉軸，來旋轉刀具。

6.5.3 相對於路徑的刀具旋轉插補（ORIROTC、THETA）

功能

使用旋轉向量的插補

以 ORIROTC 程式設計刀具旋轉的旋轉向量，相對於路徑切線，亦可以偏移插補，而此偏移可以使用 THETA 旋轉角度來進行程式設計。因此，可使用 PO[THT]，為偏移角度程式設計最高達五次的多項式。

句法

`N... ORIROTC`	啟動相對於路徑切線的刀具旋轉
`N... A3= B3= C3= THETA=value`	定義方向向量的旋轉
`N... A3= B3= C3= PO[THT]=（c2，c3，c4，c5）`	以最高達五度的多項式插補偏移角度

也可在未發生方向變更之獨立單節中，程式設計旋轉。

意義

六軸轉換中，相對於路徑的刀具旋轉插補

ORIROTC	啟動相對於路徑切線的切線旋轉向量
THETA=value	單節結尾達到的旋轉角度度數
THETA=θe	含旋轉向量結束角度 Θ_e 的旋轉角度
THETA=AC（…）	切換至每個單節的絕對尺寸
THETA=IC（…）	切換至每個單節的增量尺寸
PO[THT]=（c2，c3，c4，c5）	以五度多項式插補偏移角度

說明

旋轉向量的插補 ORIROTC

僅能以六軸轉換，啟動與刀具方向相反方向中，相對於路徑切線的刀具旋轉。

使用有效 ORIROTC

無法程式設計旋轉向量 ORIROTA。若正進行程式設計，便會輸出“警報”14128“使用有效 ORIROTC 的刀具旋轉絕對程式設計”。

三軸至五軸轉換的刀具定位方向

與三軸至五軸轉換相同，可透過尤拉角、RPY 角或方向向量，程式設計刀具的定位方向。也可透過程式設計達成空間中的刀具方向變更，程式設計包括大半徑圓弧插補 ORIVECT、方向軸的線性插補 ORIAXES、錐形柱面上所有插補 ORICONxx，及空間中含兩個刀具接點曲度外的插補 ORICURVE。

G....	旋轉軸動作的細節
X Y Z	線性軸的細節
ORIAXES	機械軸或方向軸的線性插補
ORIVECT	大半徑圓弧插補（與 ORIPLANE 完全相同）
ORIMKS	在機械座標系統中旋轉
ORIWKS	在工件座標系統中旋轉 說明，請參閱“刀具方向的旋轉”一節
A= B= C=	程式設計機械軸位置
ORIEULER	透過尤拉角的方向程式設計
ORIRPY	透過 RPY 角的方向程式設計
A2= B2= C2=	虛擬軸的角度程式設計
ORIVIRT1	使用虛擬方向軸的方向程式設計
ORIVIRT2	（定義 1），根據 MD $MC_ORIAX_TURN_TAB_1 的定義 （定義 2），根據 MD $MC_ORIAX_TURN_TAB_2 的定義
A3= B3= C3=	方向軸的方向向量程式設計
ORIPLANE	平面中插補（大半徑圓弧插補）
ORICONCW	順時針方向的錐形柱面插補
ORICONCCW	逆時針方向的錐形柱面插補
ORICONTO	含切線變化的錐形柱面插補
A6= B6= C6=	錐形旋轉軸的程式設計（正規化向量）
NUT=angle	錐形的缺口角度，以度為單位
NUT=+179	移動角度小於或等於 180 度
NUT=-181	移動角度大於或等於 180 度
ORICONIO	錐形柱面上插補
A7= B7= C7=	中間方向（程式設計為正規化向量）
ORICURVE XH YH ZH，e.g.，with polynomials PO[XH]=（xe，x2，x3，x4，x5）	指定刀具兩接點間動作的方向插補。除終點外，也可程式設計其他曲度多項式。

說明

若透過方向軸補插含有效 ORIAXES 的刀具方向，僅在單節結束啟動相對於路徑的旋轉角度。

6.5.4 方向特性的平滑化（ORIPATHS A8=、B8=、C8=）

功能

在輪廓上一特別在輪廓轉角處，以恆定加速進行的方向變更，可能造成不想要的路徑動作中斷。可插入不同的中間單節，以平滑化在方向特性中形成的起伏。若重新定向期間已啟用 ORIPATHS，則在恆定加速時發生方向變更。刀具可在此階段回退。

機械製造商

請參考機台製造商，關於使用任何預先定義之機械參數及設定資料，啟用本函數的說明。

可使用機械參數來設定轉譯回退向量的方式：

1. 在 TCS 中，依刀具方向定義 Z 座標。
2. 在工件座標系統（WCS）中，依有效平面來定義 Z 座標。

如需有關“相對於路徑的方向”函數的更多詳細資訊，請參考
參考資料：/FB3/ 功能手冊，特殊函數；三至五軸轉換（F2）

句法

需在輪廓轉角上進一步程式設計細節，才能恆定相對於路徑為整體的刀具方向。此動作的方向及路徑長度，是透過向量，使用元件 A8=X、B8=Y C8=Z，來進行程式設計。

```
N... ORIPATHS A8=X B8=Y C8=Z
```

意義

`ORIPATHS`	相對於路徑的刀具方向；已平滑化方向特性中的起伏
`A8= B8= C8=`	方向及路徑長度的向量元件
`X，Y，Z`	刀具方向中的回退移動

說明

程式設計方向向量 A8、B8、C8

若向量的長度正好為零點，則不會執行回退移動。

ORIPATHS

使用 ORIPATHS 啟動相對於路徑的刀具方向。否則，方向會以線性大半徑圓弧插補，由起始方向傳輸至結束方向。

6.6 路徑的壓縮（COMPON、COMPCURV、COMPCAD）

功能

在其中啟用方向轉換（TRAORI），且該方向轉換係用方向向量程式設計的 NC 程式，若維持在指定限制中，則可進行壓縮。

說明

僅在啟用大半徑圓弧插補時，才能壓縮方向動作，因此該動作視方向插補的 G 代碼而定。這可透過機械參數來設定，如同各軸或壓縮函數之路徑進給率的最長路徑長度及最大允差。請參閱機台製造商說明。

程式設計

刀具方向

若在五軸機台啟用方向轉換（TRAORI），可依下列方式程式設計刀具方向（獨立於動力學）：

- 方向**向量**的程式設計透過：

  A3=<...> B3=<...> C3=<...>

- 尤拉**角**或 RPY **角**的程式設計透過：

  A2=<...> B2=<...> C2=<...>

刀具旋轉

在**六軸**機台上，除刀具方向外，亦可程式設計刀具旋轉。

程式設計旋轉角度，使用：

THETA=<...>

請參閱“刀具方向的旋轉 (頁 305)”。

說明

NC 單節中，僅若以**線性方式**變更旋轉角度時，才可壓縮程式設計的其他旋轉；即為，不應程式設計含旋轉角度 PO[THT]=（...）的多項式。

可壓縮 NC 單節的一般結構

因此，可壓縮 NC 單節的一般結構如下所示：

N... X=<...> Y=<...> Z=<...> A3=<...> B3=<...> C3=<...> THETA=<...> F=<...>

或

N... X=<...> Y=<...> Z=<...> A2=<...> B2=<...> C2=<...> THETA=<...> F=<...>

說明

可直接輸入位置值（例如 X90），或透過參數設定間接輸入（例如 X=R1*（R2+R3））。

使用旋轉軸位置程式設計刀具方向

也可使用旋轉軸位置指定刀具方向，例如，使用以下結構：

```
N... X=<...> Y=<...> Z=<...> A=<...> B=<...> C=<...> THETA=<...> F=<...>
```

在此情況下，以兩種不同方式執行壓縮，視是否執行大半徑圓弧插補而定。若無進行大半徑圓弧插補，則透過旋轉軸的軸多項式，以一般方式呈現已壓縮的方向變更。

輪廓準確度

依所選的壓縮模式（MD20482 $MC_COMPRESSOR_MODE）而定，設定的軸專屬公差（MD33100 $MA_COMPRESS_POS_TOL），或下列通道專屬公差－使用設定資料所設定－在壓縮幾何軸及方向軸時有效：

SD42475 $SC_COMPRESS_CONTUR_TOL（最大輪廓偏差）

SD42476 $SC_COMPRESS_ORI_TOL（刀具方向的最大角度偏差）

SD42477 $SC_COMPRESS_ORI_ROT_TOL（刀具旋轉角度的最大角度偏差）（僅適用於五軸機台）

參考資料：
功能手冊基本功能；三至五軸轉換（F2）
：“路徑的壓縮”一章

啟用 / 停用

使用模態 G 代碼 COMPON、COMPCURV 或 COMPCAD 啟用壓縮函數。

COMPOF 會終止壓縮函數。

請參閱“NC單節壓縮（COMPON、COMPCURV、COMPCAD） (頁 221)”。

說明

僅在啟用大半徑圓弧插補時，才壓縮方向動作（即在由起始及結束方向決定的平面中變更刀具方向）。

依下列條件執行大半徑圓弧插補：

- MD21104 $MC_ORI_IPO_WITH_G_CODE = 0，
  啟用 ORIWKS，且
  程式設計方向為向量（使用 A3、B3、C3 或 A2、B2、C2）。
- 啟用 MD21104 $MC_ORI_IPO_WITH_G_CODE = 1 及
  ORIVECT 或 ORIPLANE。

  可程式設計刀具方向為方向向量，或以旋轉軸位置程式設計刀具方向。若已啟用 G 代碼 ORICONxx 或 ORICURVE 其中之一，或若程式設計方向角度的多項式（PO[PHI]及 PO[PSI]），則將不會執行大型圓弧插補。

範例

在以下範例程式中，會壓縮以多邊形定義逼近的圓弧。刀具方向同時在錐形外部移動。雖然逐一執行已程式設計的方向變更，但並不穩定，故壓縮函數會產生方向的平滑化動作。

```
程式設計                                    註解
DEF INT NUMBER=60
DEF REAL RADIUS=20
DEF INT COUNTER
DEF REAL ANGLE
N10 G1 X0 Y0 F5000 G64

$SC_COMPRESS_CONTUR_TOL=0.05          ; 最大輪廓偏差 = 0.05 毫米
$SC_COMPRESS_ORI_TOL=5                ; 最大方向偏差 = 5 度

TRAORI
COMPCURV
                                      ; 此動作定義由多邊形產生的圓弧。方向以 45 度缺口角度在
                                        錐形上繞 Z 軸移動。
N100 X0 Y0 A3=0 B3=-1 C3=1
N110 FOR COUNTER=0 TO NUMBER
N120 ANGLE=360*COUNTER/NUMBER
N130 X=RADIUS*cos（angle） Y=RADIUS*sin（angle）
A3=sin（angle） B3=-cos（angle） C3=1
N140 ENDFOR
```

6.7 平滑化方向特性（ORISON，ORISOF）

功能

"平滑化方向特性（ORISON）"函數可用來對受震盪影響的數個單節進行平滑化。目標是要達成對於方向和輪廓二者都達到平滑特性。

條件

"平滑化方向特性（ORISON）"函數僅可用於具有 5-/6 軸轉換的系統中。

句法

```
ORISON
...
ORISOF
```

意義

ORISON:	方向特性平滑化 ON 有效範圍: 模態
ORISOF:	方向特性平滑化 OFF 有效範圍: 模態

設定資料

平滑的方向特性會符合：

- 已定義的最大公差（刀具方向在角度上的最大有角度偏差）
- 已定義的最大路徑距離

這些規格會在設定資料中定義：

- SD42678 $SC_ORISON_TOL（用於平滑化方向特性的公差）
- SD42680 O$SC_ORISON_DIST（用於平滑化方向特性的路徑距離）

範例

程式碼	註解
...	
TRAORI（）	; 程式設計方向啟用。
ORISON	; 方向平滑化啟用。
$SC_ORISON_TOL=1.0	; 方向公差平滑化= 1.0 度。
G91	
X10 A3=1 B3=0 C3=1	
X10 A3=–1 B3=0 C3=1	
X10 A3=1 B3=0 C3=1	
X10 A3=–1 B3=0 C3=1	
X10 A3=1 B3=0 C3=1	
X10 A3=–1 B3=0 C3=1	
X10 A3=1 B3=0 C3=1	
X10 A3=–1 B3=0 C3=1	
X10 A3=1 B3=0 C3=1	
X10 A3=–1 B3=0 C3=1	
...	
ORISOF	; 方向平滑化停用。
...	

在 XZ 平面上，方向旋轉了超過 90 度，從-45 度到+45 度。因為方向特性的平滑化，方向不再能達到-45 度或+45 度的最大角度值。

其他資訊

單節號碼

透過設定在機械參數 MD28590 $MC_MM_ORISON_BLOCKS 中，已設定的單節號碼，對方向特性進行平滑化。

說明

若以 ORISON 來啟用方向特性的平滑化，而沒有足夠的具有已設定給它（MD28590 < 4）的單節記憶體，會輸出一個警報訊號，且該函數無法執行。

最大單節路徑長度

方向特性只在某些單節中平滑化，那些單節是其移動距離比已設定的最大單節路徑長度要短的單節（MD20178 $MC_ORISON_BLOCK_PATH_LIMIT）。具有較長移動距離的單節，中斷了平滑化，且由已程式設計的方式來移動。

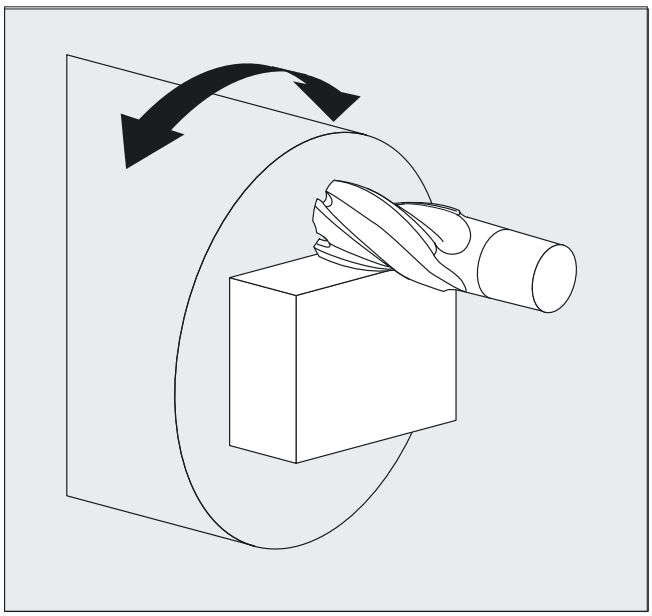
6.8 動態轉換

6.8.1 車削零件上銑削（TRANSMIT）

函數

TRANSMIT 函數可啟動下列作業：

- 在車削夾具中的車削零件（鑽孔、輪廓）上平面加工。
- 使用直角座標系統程式設計這些加工操作。
- 控制系統將直角座標系統的程式設計移動分布，套用在實際機台軸的移動運動上（標準狀況）：
  - 旋轉軸
  - 進給軸，垂直於旋轉軸
  - 縱向軸與旋轉軸平行
  - 線性軸定位為相互垂直。
- 允許相對於車削中心的刀具中心偏移。
- 速度控制對於旋轉定義的臨界值有允差。



TRANSMIT 轉換類型

TRANSMIT 加工操作有兩個可參數形式：

- 標準情況下使用（TRAFO_TYPE_n = 256）的 TRANSMIT
- 使用其他 Y 線性軸（TRAFO_TYPE_n = 257）的 TRANSMIT

可使用延伸轉換類型 257，例如，以實數 Y 軸補償刀具的鉗緊補償。

句法

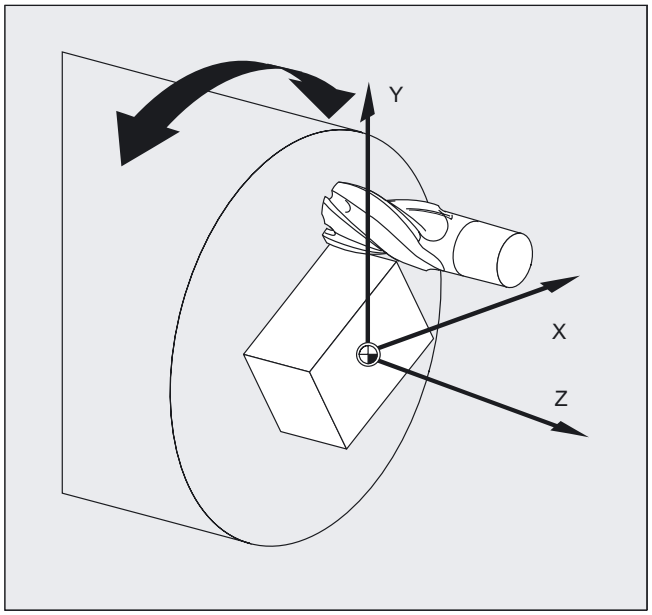
TRANSMIT 或 TRANSMIT（n）
TRAFOOF
旋轉軸
因為幾何軸已佔用旋轉軸，故無法程式設計該旋轉軸，且無法直接程式設計其為通道軸。

意義

TRANSMIT:	啟動第一個宣告的 TRANSMIT 函數。 也可指派此函數為極轉換。
TRANSMIT(n):	啟動第 n 個宣告的 TRANSMIT 函數：n 最多可為 2（TRANSMIT（1）與 TRANSMIT 相同）。
TRAFOOF:	停用有效的轉換。
OFFN:	標準輪廓偏移： 從程式設計的參考輪廓到平面加工的距離。

說明
若已在相關通道中啟用其他轉換之一，（例如 TRACYL、TRAANG、TRAORI）則亦停用有效的 TRANSMIT 轉換。

範例



程式碼	註解
N10 T1 D1 G54 G17 G90 F5000 G94	; 刀具選擇
N20 G0 X20 Z10 SPOS=45	; 逼近起點
N30 TRANSMIT	; 啟動 TRANSMIT 功能
N40 ROT RPL=–45	; 設定框架
N50 ATRANS X–2 Y10	

程式碼	註解
N60 G1 X10 Y–10 G41 OFFN=1OFFN	; 正方形粗加工：1 毫米公差
N70 X–10	
N80 Y10	
N90 X10	
N100 Y–10	
N110 G0 Z20 G40 OFFN=0	; 換刀
N120 T2 D1 X15 Y–15	
N130 Z10 G41	
N140 G1 X10 Y–10	; 正方形精加工
N150 X–10	
N160 Y10	
N170 X10	
N180 Y–10	
N190 Z20 G40	; 取消選擇框架
N200 TRANS	
N210 TRAFOOF	
N220 G0 X20 Z10 SPOS=45	; 逼近起點
N230 M30	

說明

極點

兩種穿過極點的方式：

- 沿線性軸移動
- 移動到極點，在極點旋轉旋轉軸，然後由極點移開

使用 MD 24911 及 24951 做選擇。

使用其他 Y 線性軸的 TRANSMIT（轉換類型 257）：

該極轉換的轉換變數使用機台的冗餘與另一個線性軸，以執行改進刀具補正。然後將下列條件套用於第二個線性軸：

- 較小的工作區
- 不應為工件程式的回退使用第二個線性軸。

某些用於工件程式，及 BCS 或機械座標系統（MCS）中，相對應軸指派的機械參數設定，請參閱

參考資料

/FB2/ 功能手冊延伸功能；動態轉換（M1）

6.8.2 圓柱表面轉換（TRACYL）

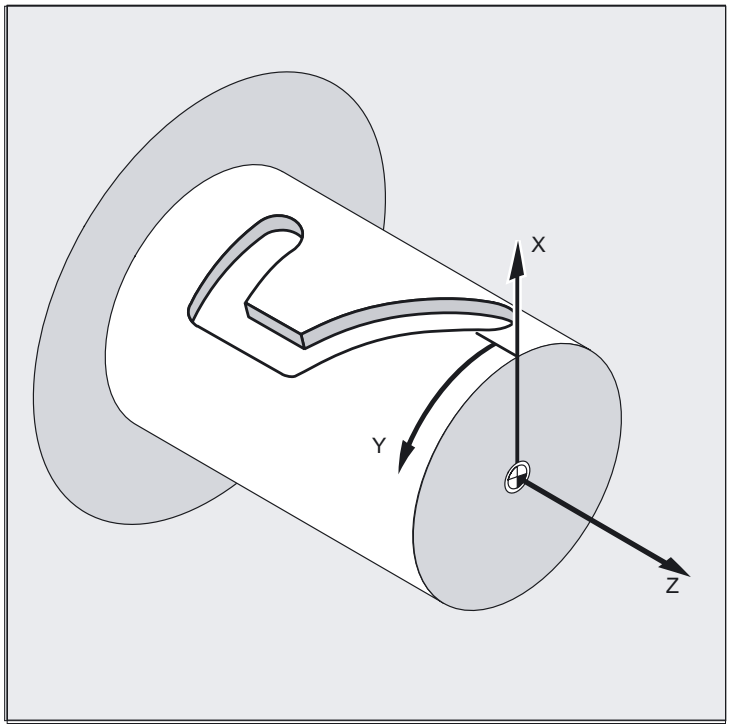
功能

TRACYL 圓柱表面轉換函數可用於下列作業：

Machine（機械）

- 在圓柱體上開縱向溝槽、
- 在圓柱物件上開橫向溝槽、
- 在圓柱體上以任何路徑開溝槽。

會參考圓柱體的開放和水平表面，程式設計溝槽的路徑。



TRACYL 轉換類型

圓柱表面座標轉換有三種形式：

- 不含溝壁偏移（TRAFO_TYPE_n=512）之 TRACYL
- 含溝壁偏移之 TRACYL：（TRAFO_TYPE_n=513）
- TRACYL 含其他線性軸及溝壁偏移：（TRAFO_TYPE_n=514）
  使用第三個參數以 TRACYL 參數化溝壁偏移。

以溝槽側邊補正進行柱體週邊曲線轉換時，補正使用的軸應定位於零點（y=0），使溝槽中心對準已程式設計的溝槽中心線。

軸利用

不得使用跟隨軸為定位軸或往返軸：

- 柱體柱面（Y 軸）的柱面方向中的幾何軸
- 溝槽側邊補正（Z 軸）的其他線性軸。

句法

TRACYL（d） 或 TRACYL（d，n） 或
為轉換類型 514
TRACYL（d，n，groove side offset）
TRAFOOF

旋轉軸

因為幾何軸已佔用旋轉軸，故無法程式設計該旋轉軸，且無法直接程式設計其為通道軸。

意義

TRACYL（d）	啟動通道機械參數中，指定的第一個 TRACYL 函數；d 為工作直徑的參數。
TRACYL （d，n）	啟動通道機械參數中，指定第 n 個 TRACYL 函數。 n 最大為 2，TRACYL（d，1）與 TRACYL（d）相符合。
D	工作直徑之值。 工作直徑為刀具刀尖與車削中心間距離的兩倍。 該直徑永遠需指定，且需大於 1。
n	選配 TRACYL 資料單節 1（預選）或 2 的第二個參數。
凹槽側邊補正	選配使用機械參數模式預選第三個參數的 TRACYL 值。 值域： 0： 不含溝壁偏移之轉換類型 514，如前 1 所示： 含溝壁偏移之轉換類型 514
TRAFOOF	轉換 OFF（關閉）（BCS 及機械座標系統（MCS）再次相同）。
OFFN	標準輪廓偏移： 從程式設計參考輪廓到溝槽邊的距離。

說明

若已在相關通道中啟用其他轉換之一，（例如 TRANSMIT、TRAANG、TRAORI）則亦停用有效的 TRACYL 轉換。

範例： 刀具定義

以下範例適用於測試 TRACYL 圓柱轉換的參數設定：

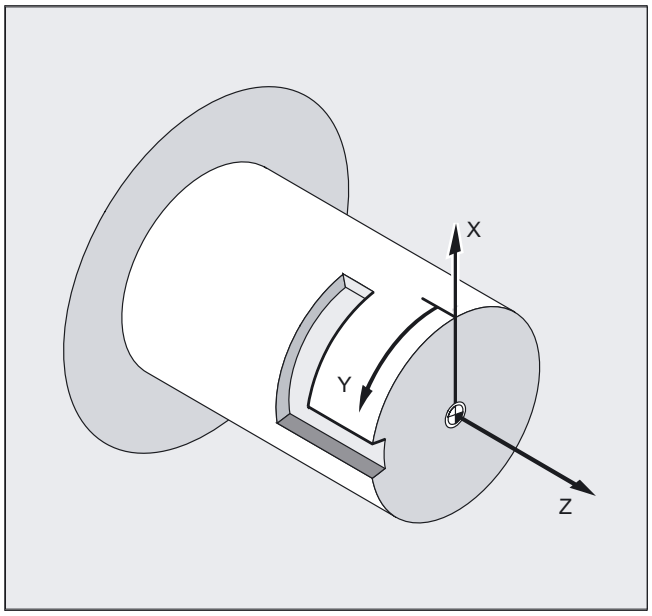
程式碼	註解	
刀具參數	意義	備註
數量（DP）		
$TC_DP1[1，1]=120	刀具類型	銑削刀具
$TC_DP2[1，1]=0	刀鼻位置	僅用於車刀

程式碼	註解	
幾何	長度補正	
$TC_DP3[1，1]=8.	長度偏移向量	根據類型及平面的計算
$TC_DP4[1，1]=9.		與平面
$TC_DP5[1，1]=7.		

程式碼	註解	
幾何	半徑	
$TC_DP6[1，1]=6.	半徑	刀具半徑
$TC_DP7[1，1]=0	切槽鋸之槽寬 b，銑削刀具之倒圓角半徑	
$TC_DP8[1，1]=0	投影 k	僅用於切槽鋸
$TC_DP9[1，1]=0		
$TC_DP10[1，1]=0		
$TC_DP11[1，1]=0	錐形銑削刀具角度	

程式碼	註解	
Wear	工具長度及半徑補正	
$TC_DP12[1，1]=0	其餘參數至$TC_DP24=0	刀具基準尺寸 / 轉接頭

範例： 產生一個 L 形的溝槽



啟動圓柱表面轉換：

程式碼	註解
N10 T1 D1 G54 G90 F5000 G94	; 刀具選擇
N20 SPOS=0	; 逼近起點
N30 G0 X25 Y0 Z105 CC=200	
N40 TRACYL （40）	; 啟動圓柱表面轉換
N50 G19	; 平面選擇

加工一個鉤形的溝槽：

程式碼	註解
N60 G1 X20	; 進給刀具至溝槽基準
N70 OFFN=12	; 定義相對於溝槽中心線 12 毫米的溝槽側邊空間
N80 G1 Z100 G42	; 逼近溝槽右側
N90 G1 Z50	; 溝槽切削方向與圓柱軸平行
N100 G1 Y10	; 溝槽切削方向與圓周平行
N110 OFFN=4 G42	; 逼近溝槽左側；定義相對於溝槽中心線 4 毫米的溝槽側邊空間
N120 G1 Y70	; 溝槽切削方向與圓周平行
N130 G1 Z100	; 溝槽切削方向與圓柱軸平行
N140 G1 Z105 G40	; 由溝壁回退
N150 G1 X25	; 回退
N160 TRAFOOF	
N170 G0 X25 Y0 Z105 CC=200	; 逼近起點
N180 M30	

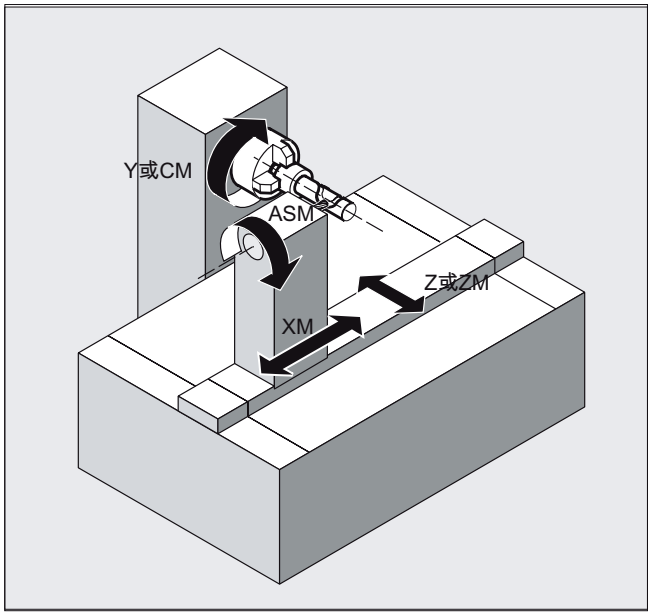
說明

不含溝壁偏移（轉換類型 512）：

控制系統會將座標系統的已程式設計移動動作，轉換到實際機械軸的移動動作。

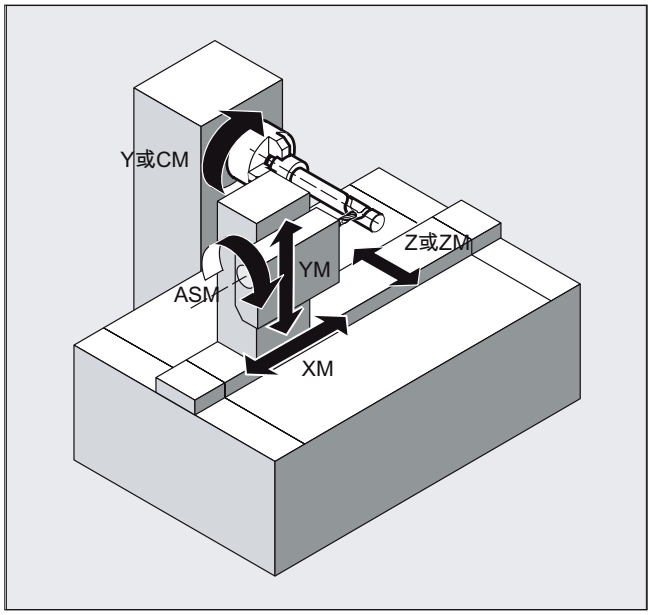
- 旋轉軸
- 進給軸垂直於旋轉軸
- 縱向軸與旋轉軸平行

線性軸定位為相互垂直。進給軸切削旋轉軸。



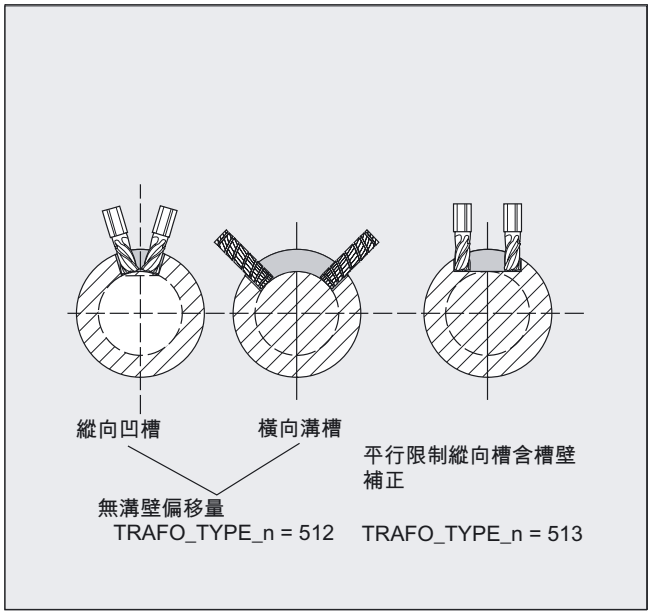
含溝壁偏移（轉換類型 513）：

動力學如上述，但另外一個縱向軸與柱面方向平行
線性軸定位為相互垂直。
速率控制對於旋轉定義的臨界值有公差。



溝槽移動區段

在軸設定 1 的情況下，僅在若溝槽寬度與刀具半徑完全相符合時，沿著旋轉軸的縱向溝槽才會受制於平行限制。
與柱面平行的溝槽（橫向溝槽）在開始及結束時非平行。



含其他線性軸及溝壁偏移（轉換類型 514）：

在含第二個線性軸的機台上，該轉換變數使用冗餘，以執行改進刀具補正。然後將下列條件套用於第三個線性軸：

- 較小的工作區
- 第二個線性軸不應為在工件程式的移動使用。

某些用於工件程式，及 BCS 或機械座標系統（MCS）中相對應軸指派的機械參數設定，請參閱

參考資料

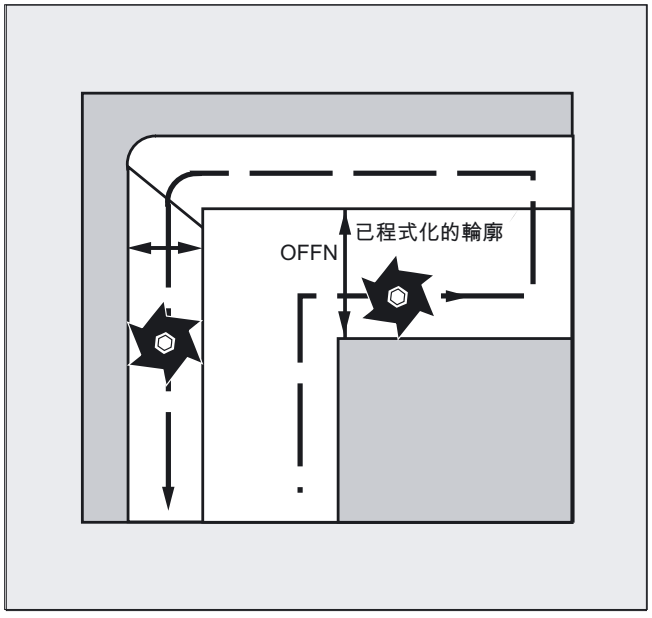
/FB2/ 功能手冊延伸功能；動態轉換（M1）

標準輪廓偏移 OFFN（轉換類型 513）

為使用 TRACYL 銑削溝槽，下列項目已程式設計：

- 工件程式中的溝槽中心線、
- 使用 OFFN 程式設計溝槽一半寬度。

OFFN 僅在啟用刀具半徑補正時才動作，避免損壞溝槽側邊。**此外，OFFN 應該>=刀具半徑，以避免損傷到對面的溝槽側邊。**



銑削溝槽的工件程式一般包含下列步驟：

1. 選擇刀具
2. 選擇 TRACYL
3. 選擇合適的座標偏移（框架）
4. 定位
5. 程式設計 OFFN
6. 選擇 TRC
7. 逼近單節（定位 TRC 並逼近溝槽邊）
8. 溝槽中心線輪廓

9. 取消選擇 TRC
10. 回退單節（退出 TRC 並離開溝槽邊）
11. 定位
12. TRAFOOF
13. 重新選擇原始座標偏移（框架）

特殊功能

- TRC 選擇：

  程式設計 TRC 與溝槽側邊無關，而是相對於已程式設計的溝槽中心線。輸入 G42（而非輸入 G41），避免刀具移至溝槽側邊左側。若欲在 OFFN 中避免此情況，請以負號輸入溝槽寬度。

- OFFN 動作與 TRACYL 不同。啟用 TRC 時，即使沒有 TRACYL 也會加入 OFFN， 所以應在 TRAFOOF 之後將 OFFN 重置為零。.
- 您可在工件程式內變更 OFFN。藉此由中心移動溝槽中心線（參見圖示）。
- 導溝：

  TRACYL 不會為導溝產生相同溝槽，需使用含直徑的刀具挖出溝槽寬度。基本上，因為溝槽測邊較大，故不可能用較小的圓柱刀具，產生相同的溝槽側邊幾何。TRACYL 可將錯誤減到最少。為避免準確度的問題，刀具半徑應只比溝寬的一半還小一些。

說明

OFFN 及 TRC

使用 TRAFO_TYPE_n = 512 時，其值在 OFFN 下做為 TRC 的允差有效。

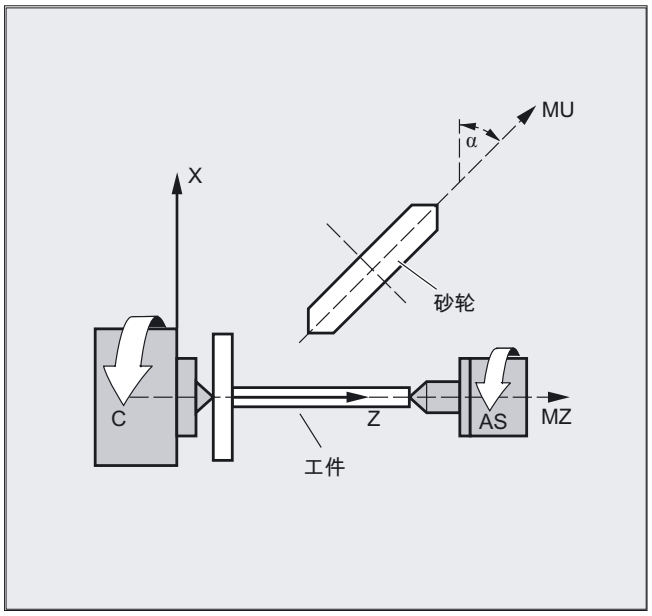
使用 TRAFO_TYPE_n = 513 時，在 OFFN 中程式設計溝槽一半寬度。以 OFFN-TRC 回退該輪廓。

6.8.3 傾斜軸（TRAANG）

功能

傾斜軸函數是為研磨技術而設計，並且輔助以下運作：

- 以傾斜進給軸來進行加工
- 直角座標系統可用於程式設計。
- 控制系統將直角座標系統的程式設計移動分布，套用在實際機台軸的移動運動上（標準狀況）： 傾斜進給軸。



句法

```
TRAANG（α）或 TRAANG（α，n）
TRAFOOF
```

意義

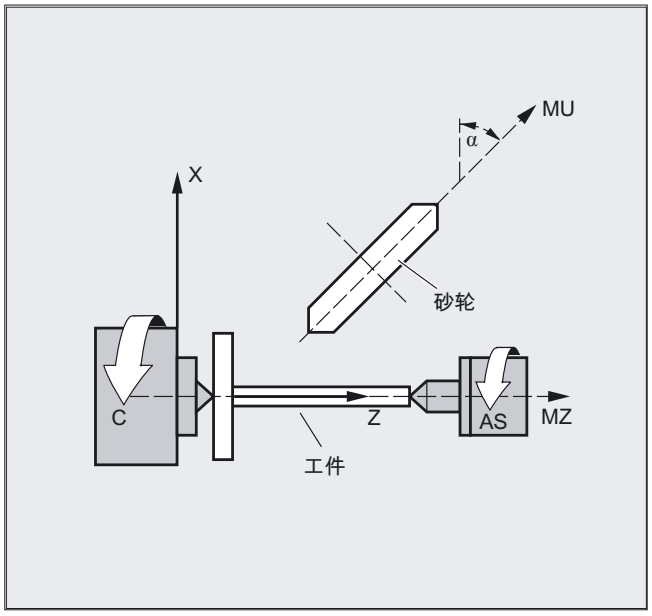
TRAANG（ ）或 TRAANG（ ，n）	使用前單節的參數化來啟動轉換。
TRAANG（α）	啟動第一項指定的傾斜軸轉換
TRAANG（α，n）	啟動第 n 項符合的傾斜軸轉換。 n 的最大值為 2。TRAANG（α，1）與 TRAANG（α）相符合。
αA	斜式軸的角度 α 的容許值為： -90 度 < α < + 90 度
TRAFOOF	轉換關閉
n	符合轉換的數量

角度 α 省略或為零

若省略 α（角度）（例如，TRAANG（），TRAANG（，n））， 則使用前單節的參數設定來啟動轉換運作。 首次選擇時，係用依照機台資料的預設設定。

角度 α = 0 （例如， TRAANG（0）， TRAANG（0，n））為有效的參數設定，並且不再與省略參數的情況相等，一如舊版的情況。

範例



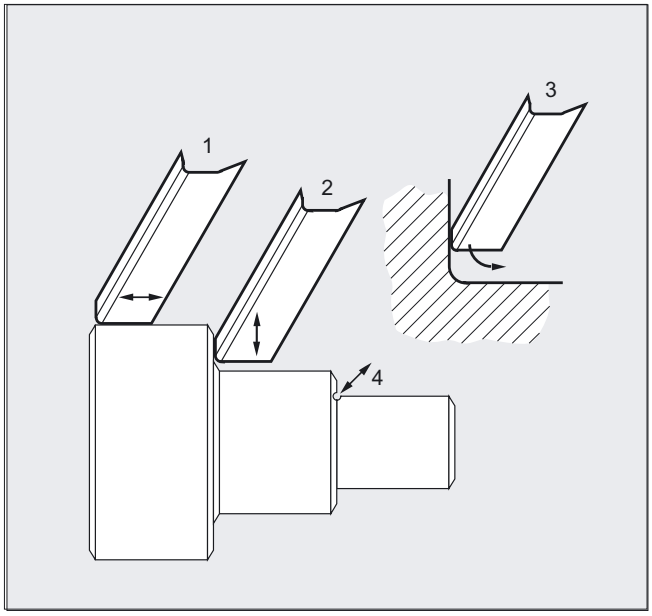
程式碼	註解
N10 G0 G90 Z0 MU=10 G54 F5000 ->	; 刀具選擇
-> G18 G64 T1 D1	平面選擇
N20 TRAANG（45）	; 啟動傾斜軸轉換
N30 G0 Z10 X5	; 逼近起點
N40 WAITP（Z）	; 釋放軸進行振盪
N50 OSP[Z]=10 OSP2[Z]=5 OST1[Z]=–2 ->	; 振盪直到到達尺寸為止
-> OST2[Z]=–2 FA[Z]=5000	（振盪，請參閱“振盪”一章）
N60 OS[Z]=1	
N70 POS[X]=4.5 FA[X]=50	
N80 OS[Z]=0	
N90 WAITP（Z）	; 釋放振盪軸為定位軸
N100 TRAFOOF	; 停用轉換運作
N110 G0 Z10 MU=10	; 回退
N120 M30	;

-> 單一單節中的程式

說明

加工操作如下所述：

1. 縱向研磨
2. 平面研磨
3. 特定輪廓的研磨
4. 斜向直進切削研磨。



機械製造商

下列設定於機械參數中定義：

- 機台軸與斜向軸間的角度，
- 刀具相對於原始座標系統的零點位置由“傾斜軸”函數指定，
- 平行軸之保留速度可供補正移動使用，
- 平行軸之保留軸加速度可供補正移動使用，

軸設定

若要在直角座標系統中進行程式設計，必須告知控制系統此座標系統與實際現有之機台軸（MU、MZ）間的關係：

- 幾何軸指派名稱
- 將幾何軸指派給通道軸
  - 一般狀況（傾斜軸未啟用）
  - 傾斜軸啟用
- 指派通道軸至機械軸編號
- 辨識主軸
- 配置機械軸名稱。

除了“傾斜軸啟用”外，程序也與一般軸設定程序相同。

6.8.4 傾斜軸程式設計（G05、G07）

功能

在 Jog（寸動進給）模式中，研磨輪的運動可以直角或以傾斜軸的方向進行（其顯示仍為直角）。所有移動的軸均為實際 U 軸，Z 軸的顯示被更新。

在寸動進給模式中，必須使用直角座標移動 REPOS－偏移。

含有效“PTP–travel（PTP－移動）”的寸動進給模式，監控直角操作範圍極限，以防過度移動，並事先緊急停住相關軸。若“PTP–travel（PTP－移動）”未啟動，則軸恰好可移動至操作範圍極限。

參考資料

/FB2/ 函數延伸功能的說明；動態轉換（M1）

句法

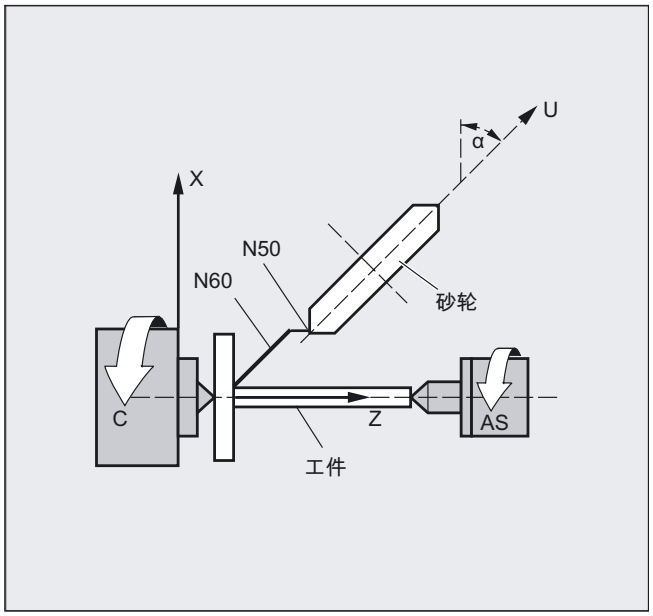
```
G07
G05
```

指令 G07/G05 係用來輔助傾斜軸的程式設計。可在直角座標系統上程式設計及顯示位置。刀具補償及零點偏移包含在直角座標中。在 NC 程式中程式設計傾斜軸角度後，便可逼近開始位置（G07），然後運作斜式插入切削（G05）。

意義

G07	逼近開始位置
G05	啟動斜式插入切削

範例



程式設計		註解
N..G18	;	程式設計傾斜軸的角度
N50 G07 X70 Z40 F4000	;	逼近開始位置
N60 G05 X70 F100	;	斜式插入切削
N70 ...	;	

6.9 直角座標 PTP 移動

功能

可使用該函數程式設計直角座標系統中的位置，然而，機台動作發生在機械座標中。例如，若動作穿過奇點，變更關節接頭的位置時，可使用該函數。

說明

該函數僅在搭配有效轉換使用時才有意義。此外，“PTP 移動”僅可搭配 G0 及 G1 執行。

句法

```
N... TRAORI
N... STAT='B10' TU='B100' PTP
N... CP
```

含一般 5/6 軸轉換之 PTP 橫向移動

若在含 PTP 一般 5/6 軸轉換有效期間，已在機械座標系統（ORIMKS）中啟動點對點橫向移動，則可用旋轉軸位置程式設計刀具方向

```
N... G1 X Y Z A B C
```

也可使用尤拉角及 / 或 RPY 角向量，而不管動力學

```
N... ORIEULER 或 ORIRPY
N... G1 X Y Z A2 B2 C2
```

或方向向量

```
N... G1 X Y Z A3 B3 C3
```

已程式設計。可啟用旋轉軸插補、含大圓弧插補 ORIVECT 的向量插補，或錐形柱面上方向向量插補 ORICONxx。

含向量方向的非獨特性

程式設計含向量的方向時，可使用旋轉軸位置中的非獨特性。可透過程式設計 STAT = <...>選擇欲逼近的旋轉軸位置。If

若 STAT = 0 已程式設計（與預設值相等），
則會逼近離起始位置距離最近的位置。If

若 STAT = 1 已程式設計，
則會逼近離起始位置距離較大的位置。

意義

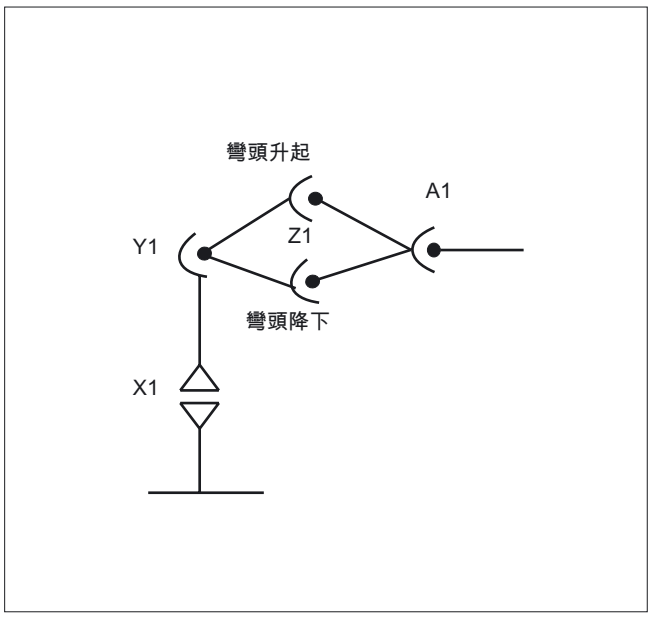
以模態方式執行 PTP 及 CP 指令。CP 為預設設定。

若在程式設計 STAT 值時套用模態，則 TU 程式設計= <...>非模態。

另一項差異為，程式設計 STAT 值僅在向量插補期間有效，而程式設計 TU 也於旋轉軸插補期間進行評估。

PTP	**點對點**（點對點動作） 以同步軸動作執行該動作；動作中最慢的涉入軸為速率的控制軸。
CP	**持**續**路**徑（路徑動作） 以直角座標路徑動作執行該動作。
STAT=	關節接頭的位置；該值依轉換而定。
TU=	TURN 資訊區塊式動作。因此可準確逼近-360 度與+360 度間的軸角度。

範例




```
N10 G0 X0 Y-30 Z60 A-30 F10000     初始設定
                                   →彎頭升起
N20 TRAORI（1）                    轉換啟動
N30 X1000 Y0 Z400 A0
N40 X1000 Z500 A0 STAT='B10'       不轉換重新定向
TU='B100' PTP                      →彎頭降下
N50 X1200 Z400 CP                  轉換再度啟用
N60 X1000 Z500 A20
N70 M30
```

含一般 5 軸轉換之 PTP 橫向移動

假設：基於直角 CA 動力學。

程式碼	註解
TRAORI	; 轉換 CA 動力學啟動
PTP	; 啟動 PTP 移動
N10 A3 = 0 B3 = 0 C3 = 1	; 旋轉軸位置 C = 0 A = 0
N20 A3 = 1 B3 = 0 C3 = 1	; 旋轉軸位置 C = 90 A = 45
N30 A3 = 1 B3 = 0 C3 = 0	; 旋轉軸位置 C = 90 A = 90
N40 A3 = 1 B3 = 0 C3 = 1 STAT = 1	; 旋轉軸位置 C = 270 A = –45

選擇旋轉軸位置的準確逼近位置：

在單節 N40 中，透過程式設計 STAT = 1，旋轉軸即可從起點（C=90，A=90）移動長路徑至終點（C=270，A=–45），而非若 STAT = 0，該情況可能會移動最短路徑至終點（C=90，A=45）。

說明

PTP 及 CP 指令會影響直角座標移動及機械軸移動間的變換。

含一般 5/6 軸轉換之 PTP 橫向移動

在 PTP 橫向移動不同於 5/6 軸轉換，若僅有方向變更，TCP 一般不會維持固定。以線性方式逼近所有轉換軸的轉換結束位置（3 個線性軸及最多 3 個圓軸）時，無需仍然有效的轉換。

經由程式設計模態 G 代碼 CP 停用 PTP 橫向移動。

本文件說明各種不同的轉換：
/FB3/ 功能手冊，特殊函數；處理變換套件（TE4）。

程式設計位置（STAT=）

並非僅以直角座標的位置資料及刀具方向決定機台位置。依與其有關的動力學而定，可能有多達八個不同且重要的關節接點位置。其皆專屬於轉換。需使用指令 STAT=指定關節接點的位置，才能僅轉換直角座標位置為軸角度。“STAT”指令中每個可能位置皆含一個為二進制值的位元。

如需“STAT”欲程式設計的設定位元相關資訊，請參閱：
/FB2/ 函數延伸功能的說明；動態轉換（M1），“直角座標 PTP 移動”一節。

程式設計軸角度（TU=）

為能準確逼近< ± 360 度的軸角度，需使用“TU=”指令程式設計本資訊。

軸由最短路徑移動：

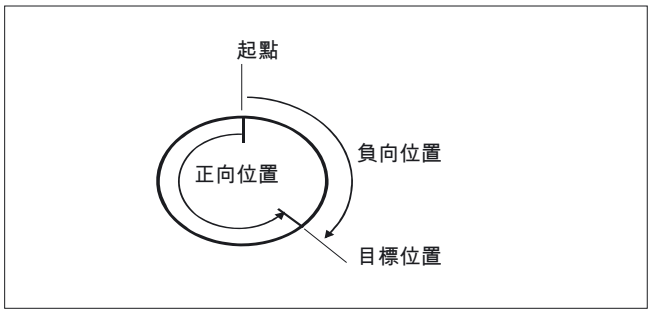
- 未為位置程式設計 TU 時，
- 使用移動範圍為> ±360 度的軸。

範例：

可依正向或負向逼近圖示中所示的目標位置。在位址 A1 下程式設計方向。

A1=225°，TU=Bit 0，→ 正向

A1=−135°，TU=Bit 1，→ 負向



為一般 5/6 軸轉換及目標位置評估 TU 之範例

變數 TU 包含一個指示轉換中涉入的每個軸移動方向位元。TU 位元的指派符合圓軸的通道軸視點。最多僅為包含在轉換內的 3 個可能圓軸評估 TU 資訊。

Bit0：Axis 1，TU bit = 0：0 度<=圓軸角度<360 度

Bit1：Axis 2，TU bit = 1：–360 度<圓軸角度<0 度

圓軸的起始位置為 C = 0。程式設計 C = 270，使圓軸移至下列目標位置：

C = 270：TU 位元 0，正向旋轉

C = –90：TU 位元 1，負向旋轉

更多行為

模式變更

“直角座標 PTP 移動”函數僅在 AUTO 及 MDA 操作模式中才有用。模式變更為寸動進給時，會保持目前設定。

設定 G 代碼 PTP 時，各軸皆依機械座標系統（MCS）移動。設定 G 代碼 CP 時，各軸皆依工件座標系統（WCS）移動。

POWER ON（開機）/RESET（重置），

POWER ON（開機）或 RESET（重置）後，設定依機械參數 $MC_GCODE_REST_VALUES[48]而定。預設橫向移動模式設定為“CP”。

REPOS

若在中斷單節期間設定“直角座標 PTP 移動”函數，也可使用 PTP 來重新定位。

覆蓋移動

僅在直角座標 PTP 移動中，才可有限度地執行 DRF 偏移或外部零點偏移。由 PTP 變更為 CP 動作時，BCS 不可取消。

平滑化 CP 與 PTP 間動作

單節間可程式設計的變化倒圓角可使用 G641。

倒圓角區域的大小為以毫米或英吋為單位的路徑，為單節變化欲磨圓的起點至終點的區域。其大小需如下指定：

- G0 單節使用 ADISPOS
- 其他所有動作指令則使用 ADIS。

路徑計算符合非 G0 單節 F 位址考量。為 FGROUP（..）中指定的軸保持進給。

進給計算

在 CP 單節上，使用基本座標系統的直角座標軸來計算。

在 PTP 單節上，使用機械座標系統的對應軸來計算。

6.9.1 TRANSMIT 的 PTP

功能

可使用 TRANSMIT 的 PTP，以時間最佳化逼近 G0 及 G1。不以線性方式移動基本座標系統軸（CP），而以線性方式移動機械軸（PTP）。其效果為機械軸動作接近極點，導致更加快速到達單節結束點。

工件程式仍寫在直角座標工件座標系統內，而所有座標偏移、旋轉及框架的程式設計設定仍然有效。也會在直角工件座標系統中顯示 HMI 上的模擬。

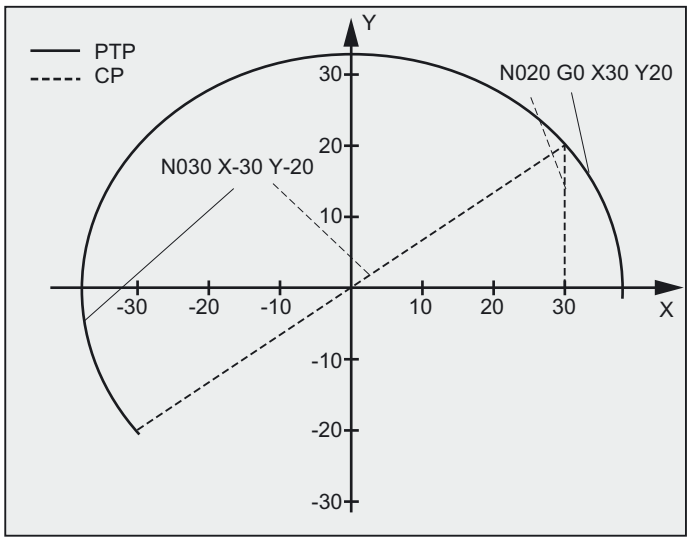
句法

```
N... TRANSMIT
N... PTPG0
N... G0 ...
...
N... G1 ...
```

意義

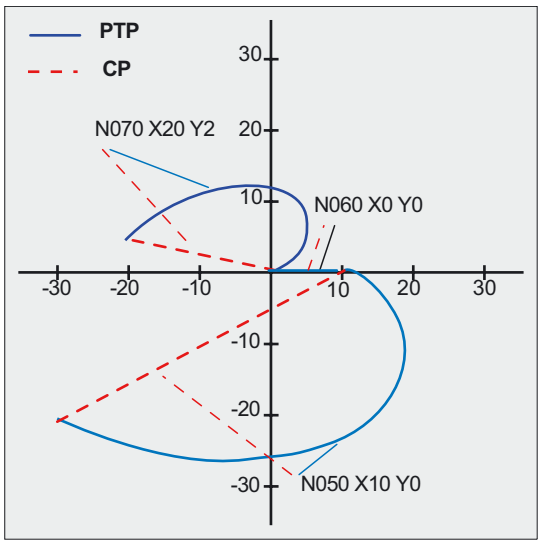
TRANSMIT	啟動第一個宣告的 TRANSMIT 函數 （請參閱“車削零件上銑削：TRANSMIT”一節）
PTPG0	**點對點 G0**（每個 G0 單節上的自動點對點動作，然後再次設定 CP） 由於 STAT 及 TU 為模態，故皆以最近的程式設計值動作。
PTP	**點對點**；（點對點動作） 對於 TRANSMIT，PTP 代表在直角座標中，繞極點或從極點回退螺線至阿基米得螺線。其最終刀具動作執行與 CP 極為不同，並在相關程式設計範例中呈現。
STAT=	處理關於極點的非獨特性
TU=	TU 與含 TRANSMIT 的 PTP 無關

使用 PTP 及 TRANSMIT 的極點繞行之範例



程式碼	註解
N001 G0 X30 Z0 F10000 T1 D1 G90	; 初始設定，絕對尺寸
N002 SPOS=0	
N003 TRANSMIT	; 轉換 TRANSMIT
N010 PTPG0	; 每一個 G0 單節， PTP 之後自動接 CP
N020 G0 X30 Y20	
N030 X-30 Y-20	
N120 G1 X30 Y20	
N110 X30 Y0	
M30	

使用 PTP 及 TRANSMIT 從極點回退之範例



程式設計	註解
N001 G0 X90 Z0 F10000 T1 D1 G90	; 初始設定
N002 SPOS=0	
N003 TRANSMIT	; 轉換 TRANSMIT
N010 PTPG0	; 每一個 G0 單節，PTP 之後自動接 CP
N020 G0 X90 Y60	
N030 X-90 Y-60	
N040 X-30 Y-20	
N050 X10 Y0	
N060 X0 Y0	
N070 X-20 Y2	
N170 G1 X0 Y0	
N160 X10 Y0	
N150 X-30 Y-20	
M30	

說明

PTP 及 PTPG0

為所有可處理 PTP 轉換考慮的 PTPG0。PTPG0 與其他情況皆無關。

在 CP 模式處理 G0 單節。

在工件程式中執行 PTP 或 PTPG0 的選擇，或在機械參數$MC_GCODE_RESET_VALUES[48]中取消選擇 CP。

⚠ 小心

限制

關於刀具動作及碰撞，需套用許多限制以及需排除的特定函數，例如

不能以 PTP 啟用刀具半徑補正（TRC）。

使用 PTPG0 時，以 CP 移動啟用的刀具半徑補正（TRC）。

PTP 不允許滑順逼近及回退（SAR）。

使用 PTPG0 時，使用 CP 移動來平滑化逼近及回退（SAR）。

PTP 不允許切削循環（CONTPRON、CONTDCON）。

使用 PTPG0 切削循環（CONTPRON、CONTDCON）時，以 CP 移動。

忽略倒角（CHF、CHR）及磨圓（RND、RNDM）。

壓縮機與 PTP 不相容，並在 PTP 單節中自動取消選擇。

不可在 PTP 區段中變更與插補重疊的軸。

若指定 G643，在以軸精確度平滑化後，會自動切換為 G642。

啟用 PTP 時，轉換軸無法同時為定位軸。

參考資料：

/FB2/ 功能手冊延伸功能；動態轉換（M1），
“直角座標 PTP 移動”一節。

TRACON 的 PTP：

可以 TRACON 使用 PTP，其供應的第一個連鎖轉換支援 PTP。

TRANSMIT 的 STAT=及 TU=意義

若欲將旋轉軸轉 180 度，或將 CP 輪廓穿過極點，旋轉軸可依機械參數
$MC_TRANSMIT_POLE_SIDE_FIX_1/2 [48]旋轉-/+ 180 度，並以順時針或逆時針方向移動。該參數也可設定，是否移動穿過極點，或是否執行繞極點旋轉。

6.10 選擇轉換時的限制

功能

可透過工件程式或 MDA（手動輸入）選擇轉換。請注意：

- 不可插入中間動作單節（倒角 / 半徑）。
- 需排除曲線單節順序；否則會顯示訊息。
- 需取消選擇微調刀具補正（FTOCOF）；否則會顯示訊息。
- 需取消選擇刀具半徑補正（G40）；否則會顯示訊息。
- 已啟動的刀長偏移，係由控制系統包含在轉換中。
- 控制系統取消選擇目前啟用的框架，再進行轉換。
- 控制系統為受轉換影響的軸，取消選擇啟用的操作範圍限制（與 WALIMOF 相同）。
- 取消選擇保護區監控。
- 中斷連續路徑控制以及倒圓角作業。
- 需相對於單節同步處理所有在機械參數中指定的軸。
- 將已交換的軸交換回來；否則會顯示訊息。
- 輸出相關軸的訊息。

刀具換用

僅在取消選擇刀具半徑補正函數時，才能變更刀具。

刀長偏移中的變更，及刀具半徑補正選擇 / 取消選擇，不得在同一單節中進行程式設計。

框架變更

允許所有只參考基本座標系統的敘述（FRAME、刀具半徑補正）。然而，使用 G91 的框架變更（增量尺寸）不同於未啟用的轉換，不分別做處理。在新框架的工件座標系統中評估欲移動的增量，不管前一個單節中為哪個有效框架。

例外情況

不能使用受轉換影響的軸

- 為預先設定的軸（警報）、
- 逼近檢查點（警報）、
- 供參考（警報）。

6.11 取消選擇轉換（TRAFOOF）

功能

TRAFOOF 指令停用所有有效的轉換與框架。

說明

需由更新的程式設計啟動之後所需的框架。

請注意：

選擇的相關限制同樣適用於取消選擇轉換（請參閱“選擇轉換時的限制”一節）。

句法

```
TRAFOOF
```

意義

TRAFOOF	用來停用所有有效的轉換/框架的指令

6.12 連鎖轉換（TRACON、TRAFOOF）

功能

可連鎖**兩個**轉換，因此使用來自第一個轉換軸的動作元件，為連鎖第二個轉換的輸入資料。在機械軸上來自第二個轉換動作的動作零件。

連鎖可包含**兩個**轉換。

說明

永遠指派刀具給連鎖中的第一個轉換。 後續的轉換便如有效刀長為零點般動作。 連鎖中的第一個轉換，僅在機械參數（_BASE_TOOL_）中設定的基本刀長才有效。

機械製造商

記下由機台製造商提供，以機械參數預先定義的任何轉換資訊。

可選擇轉換及連鎖轉換。 目前的目錄會提供特定控制連鎖中，可用的特定轉換相關資訊。

應用

- 使用傾斜研磨輪，程式設計研磨輪廓為圓柱側邊線（TRACYL），例如，刀具研磨。
- 使用傾斜研磨輪，修飾以 TRANSMIT 產生且非圓形的輪廓切削。

句法

```
TRACON（trf，par）        可啟動連鎖轉換。
TRAFOOF
```

意義

TRACON	可啟動連鎖轉換。 若已先啟動其他轉換，則由 TRACON（）隱含停用。
TRAFOOF	將停用最近啟動（連鎖）的轉換。
trf	連鎖轉換的編號： 0 或 1 代表第一個 / 單一連鎖轉換。 若此處無程式設計項目，則與指定值 0 或 1 時意義相同，即啟動第一個 / 單一轉換。 2 代表第二個連鎖轉換 （值不等於 0 - 2 會產生錯誤警報）。
par	可為連鎖預期參數中的轉換，以逗號分隔一個或多個參數，例如，傾斜軸的角度。 若未設定參數，則預設值或上次使用的參數會生效。 若先前參數的預設設定會生效，則需使用逗號，以確保會依預期的順序評估指定的參數。特別是，雖無需指定轉換，但至少一個參數前需一個逗號。 例如 TRACON（，3.7）.

需求

第二個轉換需為**“傾斜軸”**（TRAANG）。第一個轉換可為：

- 方向轉換（TRAORI），包括通用銑削頭
- TRANSMIT
- TRACYL
- TRAANG

使用連鎖轉換啟動指令的條件為，以機械參數定義各個欲連鎖的轉換，及欲啟動的連鎖轉換。
個別轉換說明中所指的補充條件及特殊情況，也適用於連鎖轉換中。
有關設定轉換機械參數的資訊，請參閱：
/FB2/ 功能手冊延伸功能；動態轉換（M1）及
/FB3/ 功能手冊，特殊函數；三至五軸轉換（F2）。

7 刀具偏移量

7.1 偏移記憶體

功能

偏移記憶體的結構

可使用 T 及 D 編號呼叫（“Flat D 編號”除外）每個資料欄位；除刀具的幾何資料外，也包含其他資訊，例如，刀具類型。

Flat D 編號結構

若在 NCK 外進行刀具管理，則使用“Flat D 編號結構”。在此情況下，使用未指派至工具且對應的刀具補正單節，產生 D 編號。

可繼續在工件程式中程式設計 T。然而，T 不會參考已程式設計的 D 編號。

使用者刀刃資料

可透過機械參數設定使用者刀刃資料。請參閱機台製造商說明。

刀具參數

說明

偏移記憶體中的各值

可透過系統變數，由程式讀取及寫入偏移記憶體 P1 到 P25 的各值。保留其他所有參數。

刀具參數$TC_DP6 到$TC_DP8、$TC_DP10 及$TC_DP11，以及$TC_DP15 到$TC_DP17、$TC_DP19 及$TC_DP20 依刀具類型而有其他意義。

[1] 也適用於 3D 平面銑削的銑削刀具
[2] 切槽鋸的刀具類型
[3] 保留：非由 SINUMERIK 840D 使用

刀具參數編號（DP）	系統變數的意義	註解
$TC_DP1	刀具類型	概觀請參閱清單
$TC_DP2	刀鼻位置	僅用於車刀
幾何	**長度補正**	
$TC_DP3	長度 1	配置方式
$TC_DP4	長度 2	類型及層級
$TC_DP5	長度 3	
幾何	**半徑**	
$TC_DP6[1] $TC_DP6[2]	半徑 1 / 長度 1 直徑 d	銑削 / 車削 / 研磨刀具 切槽鋸
$TC_DP7[1] $TC_DP7[2]	長度 2 / 轉角半徑，錐形銑削刀具 槽寬度 b 轉角半徑	銑削刀具 切槽鋸
$TC_DP8[1] $TC_DP8[2]	銑削刀具的倒圓角半徑 1 投影長度 k	銑削刀具 切槽鋸
$TC_DP9[1,3]	倒圓角半徑 2	保留
$TC_DP10[1]	角度 1 刀具端面	錐形銑削刀具
$TC_DP11[1]	角度 2 刀具縱向軸	錐形銑削刀具
Wear	**長度及半徑補正**	
$TC_DP12	長度 1	
$TC_DP13	長度 2	
$TC_DP14	長度 3	
$TC_DP15[1] $TC_DP15[2]	半徑 1 / 長度 1 直徑 d	銑削 / 車削 / 研磨刀具 切槽鋸
$TC_DP16[1] $TC_DP16[3]	長度 2 / 轉角半徑、錐形銑削刀具、槽寬度 b 轉角半徑	銑削刀具 切槽鋸
$TC_DP17[1] $TC_DP17[2]	銑削刀具的倒圓角半徑 1 投影長度 k	銑削 / 3D 平面銑削 切槽鋸
$TC_DP18[1,3]	倒圓角半徑 2	保留
$TC_DP19[1]	角度 1 刀具端面	錐形銑削刀具
$TC_DP20[1]	角度 2 刀具縱向軸	錐形銑削刀具
刀具基本尺寸 / 轉接頭	**刀長偏移**	
$TC_DP21	長度 1	
$TC_DP22	長度 2	
$TC_DP23	長度 3	
技術		
$TC_DP24	間隙角度	僅用於車刀
$TC_DP25		已保留

註解

有多個可供幾何變數使用的輸入元件（例如，長度 1 或半徑）相加以產生值（例如，全長 1，總半徑），然後使用該值計算。

無需的偏移值需指派為零。

刀具參數$TC-DP1 到$TC-DP23，搭配輪廓刀具

說明

未列於表中的刀具參數，例如$TC_DP7，將不予評估，即其內容無意義。

刀具參數編號（DP）	意義	切削 Dn		註解
$TC_DP1	刀具類型			400 ... 599
$TC_DP2	刀鼻位置			
幾何	**長度補正**			
$TC_DP3	長度 1			
$TC_DP4	長度 2			
$TC_DP5	長度 3			
幾何	**半徑**			
$TC_DP6	半徑			
幾何	**角度限制**			
$TC_DP10	最小角度限制			
$TC_DP11	最大角度限制			
Wear	**長度及半徑補正**			
$TC_DP12	磨損長度 1			
$TC_DP13	磨損長度 2			
$TC_DP14	磨損長度 3			
$TC_DP15	磨損半徑			
磨損	**角度限制**			
$TC_DP19	最小磨損角度限制			
$TC_DP20	最大磨損角度限制			
刀具基本尺寸 / 轉接頭	**長度補正**			
$TC_DP21	長度 1			
$TC_DP22	長度 2			
$TC_DP23	長度 3			

基本值及磨損值

每個結果值皆為基本值及磨損值的總合，（例如，半徑為$TC_DP6 + $TC_DP15）。第一個刀刃的刀長也會加入基本量測（$TC_DP21 – $TC_DP23）。其他所有參數可能會影響標準刀具的有效刀長，也會影響該刀長（轉接頭、方向性刀把、設定資料）。

角度限制 1 及 2

角度限制 1 及 2 各與圓心參考點的刀刃中心點向量相關，並依順時針計算。

7.2 附加偏移

7.2.1 選擇附加偏移（DL）

功能

附加偏移可視為程序偏移，可在加工中程式設計該偏移。參考自刀刃的幾何資料，因此為刀具切削資料的元件。

使用 DL 編號定址附加偏移的資料（DL：位置相關；關於使用位置的偏移），並透過操作員介面輸入。

應用

因使用位置而造成的尺寸錯誤可用附加偏移補償。

句法

```
DL=<編號>
```

意義

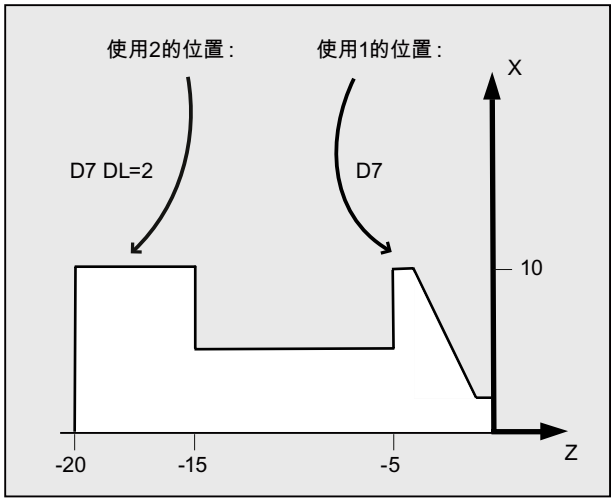
DL	啟動附加偏移的指令
<編號>	使用<編號>參數指定欲啟動的附加刀具偏移資料。

說明

使用機械參數定義附加偏移的數量，並加以啟動（→請小心觀察機台 OEM'的資料！）

範例

兩個軸承座使用相同的刀刃：



程式碼	註解
`N110 T7 D7`	; 快轉區所在位置為 7．D7，並啟動 DL=1，且移動通過下一個單節。
`N120 G0 X10 Z1`	
`N130 G1 Z-6`	
`N140 G0 DL=2 Z-14`	; 除 D7 外也啟動 DL=2，並移動通過下一個單節。
`N150 G1 Z-21`	
`N160 G0 X200 Z200`	; 逼近換刀位置。
`...`	

7.2.2 指定磨損及設定值（$TC_SCPxy[t，d]，$TC_ECPxy[t，d]）

功能

可使用系統變數讀取及寫入磨損及設定值。以刀具及刀鼻的對應系統變數邏輯為其邏輯基準。

系統變數

系統變數	意義
$TC_SCPxy[<t>，<d>]	透過 xy 指派磨損值給特定的幾何參數，x 與磨損值的數量相同，而 y 則建立幾何參數的參考。
$TC_ECPxy[<t>，<d>]	透過 xy 指派設定值給特定的幾何參數，x 與設定值的數量相同，而 y 則建立幾何參數的參考。
<t>：刀具的 T 編號 **<d>**：刀具刀刃的 D 編號	

說明

將定義的磨損及設定值加入至幾何參數及其他偏移參數中（D 編號）

範例

長度 1 的磨損值為<d>刀具<t>刀刃設定為 1.0。

參數：$TC_DP3（長度 1，搭配車刀）

磨損值：$TC_SCP13 到$TC_SCP63

設定值：$TC_ECP13 到$TC_ECP63

$TC_SCP43 [<t>，<d>] = 1.0

7.2.3 刪除附加偏移（DELDL）

功能

DELDL 指令會刪除刀具刀刃的附加偏移（以釋出記憶體空間）。定義的磨損值及設定值均刪除。

句法

```
DELDL[<t>，<d>]
DELDL[<t>]
DELDL
<Status>=DELDL[<t>，<d>]
```

意義

DELDL	刪除附加偏移的指令
<t>	刀具 T 編號
<d>	刀具刀刃的 D 編號
DELDL[<t>，<d>]	刪除刀具<t>刀刃<d>的所有附加偏移。
DELDL[<t>]	刪除刀具<t>所有刀刃的全部附加偏移。
DELDL	刪除 TO 元件所有刀具之所有刀刃的全部附加偏移（在已程式設計指令的通道中）。
<狀態>	刪除狀態 值：　意義： 0　刪除已順利完成。 -　偏移未刪除（若參數設定正好指定一個刀具稜邊），或未完全刪除（若參數設定指定多個刀刃）。

說明

無法刪除啟用刀具的磨損及設定值（基本上與 D 或刀具資料的刪除行為相同）。

7.3 刀具補正的特殊處理方式

功能

使用設定資料 SD42900 到 SD42960 控制刀長及磨損的符號評估。

進行幾何軸鏡像或變更加工平面，及依刀具方向補正溫度時，同樣適用於磨損元件的行為。

磨損值

若參考下述情形中的磨損值，則應視為實際磨損值的總和（$TC_DP12 到$TC_DP20），及含磨損值的總和偏移（$SCPX3 到$SCPX11），以及設定值（$ECPX3 到$ECPX11）。

更多關於總和偏移的資訊，請參閱：

參考資料：

功能手冊，刀具管理

設定資料

設定資料	意義
SD42900 $SC_MIRROR_TOOL_LENGTH	刀長元件的鏡像及刀具基準尺寸的元件。
SD42910 $SC_MIRROR_TOOL_WEAR	刀長元件磨損值的鏡像
SD42920 $SC_WEAR_SIGN_CUTPOS	將磨損值元件的符號評估為刀鼻位置的函數。
SD42930 $SC_WEAR_SIGN	反轉磨損尺寸的符號。
SD42935 $SC_WEAR_TRANSFORM	磨損值的轉換。
SD42940 $SC_TOOL_LENGTH_CONST	將刀長元件指派給幾何軸。
SD42950 $SC_TOOL_LENGTH_TYPE	獨立於刀具類型之刀長元件指派
SD42960 $SC_TOOL_TEMP_COMP	刀具方向的溫度補正值。程式設計刀具方向時，也會運作。

參考資料

功能手冊基本功能；刀具偏移（W1）

其它資訊

啟用修改的設定資料

當上述說明的設定資料已修改，需到下次選擇刀具稜邊時，才會重新評估刀具元件。若已啟用刀具，且欲重新評估該刀具的資料，則需再次選擇刀具。

該原則亦適用於，因軸鏡像狀態變更，而修改結果刀長的情況。需在鏡像指令後再次選擇刀具，才能啟動修改的刀長元件。

可定向刀把及新的設定資料

設定資料 SD42900 到 SD42940 對啟用含方向功能的刀把元件無任何影響。然而，以可定向刀把計算，可得出刀具及其總結果長度（刀長+磨損+刀具基準尺寸）。以設定資料啟動的所有修改，皆包含在結果總長度的計算中，即，可定向刀把的向量獨立於加工平面。

說明

使用可定向刀把時，為非鏡像基本系統定義所有刀具非常實用，即使是僅用於鏡像加工的刀具。以鏡像軸加工時，刀把會旋轉，以使刀具的實際位置被正確地說明。然後所有刀長元件會自動依正確方向動作，依各個軸的鏡像狀態而定，不需透過設定資料控制各個元件評估。

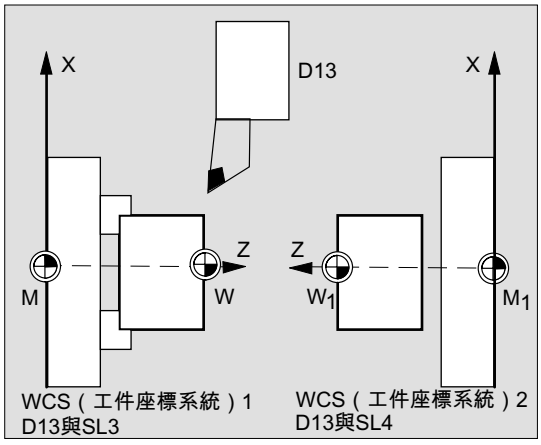
其他應用程式選項

若機台上無車刀的實體選項，即使已永久安裝不同方向的刀具，可定向刀把功能的使用也可以很實用。可在基本方向中一起執行刀具尺寸標註，根據虛擬刀把的旋轉計算與加工相關的尺寸。

7.3.1 刀長的鏡像

功能

當設定資料 SD42900 $SC_MIRROR_TOOL_LENGTH 及 SD42910 $SC_MIRROR_TOOL_WEAR 未設定為零點時，即可進行刀長元件及含磨損值與其相關軸的基本尺寸元件之鏡像。



SD42900 $SC_MIRROR_TOOL_LENGTH

設定資料**不等於**零點：

該刀長元件（$TC_DP3、$TC_DP4 及$TC_DP5）及基本尺寸的元件（$TC_DP21、$TC_DP22 及$TC_DP23）是對相關軸進行鏡像，也由反轉符號進行鏡像。

磨損值**不會**進行鏡像。若欲進行鏡像該值，則需加以設定設定資料 SD42910 $SC_MIRROR_TOOL_WEAR。

SD42910 $SC_MIRROR_TOOL_WEAR

設定資料**不等於**零點：

刀長元件的磨損值－其關聯軸會進行鏡像－也由反轉符號進行鏡像。

7.3.2 磨損符號評估

功能

當設定資料 SD42920 $SC_WEAR_SIGN_CUTPOS 及 SD42930 $SC_WEAR_SIGN 未設定為零點時，即可反轉磨損元件的符號評估。

SD42920 $SC_WEAR_SIGN_CUTPOS

設定資料**不等於**零點：

對於含相關刀鼻位置的刀具（車削及研磨刀具，刀具類型 400），此時加工平面中磨損元件的符號評估視刀鼻位置而定。該設定資料對於不含相關刀鼻位置的刀具類型無意義。

在下表中，用 SD42920（不等於零點）反轉的尺寸符號係使用 X 設計：

刀鼻位置	長度 1	長度 2
1		
2		X
3	X	X
4	X	
5		
6		
7		X
8	X	
9		

說明

使用 SD42920 及 SD42910 的符號評估相互獨立。例如若使用兩個設定資料變更尺寸的符號，則結果符號維持不變。

SD42930 $SC_WEAR_SIGN

設定資料**不等於**零點：

反轉所有磨損尺寸的符號。此會影響刀長及其他變數，例如刀具半徑、倒圓角半徑等等。

若輸入正號磨損尺寸，刀具則變為“較短”且“較細”，請參閱“刀具偏移，特殊處理”，啟動變更的設定資料”一章。

7.3.3 生效加工操作的座標系統（TOWSTD、TOWMCS、TOWWCS、TOWBCS、TOWTCS、TOWKCS）

功能

依機台的動力學或可定向刀把的可用性而定，將在座標系統之一中量測的磨損值轉換成適當的座標系統。

主動加工操作的座標系統

使用下列會產生刀具偏移的座標系統，將刀長元件“磨損”，透過相對應群組 56 的 G 代碼，結合至生效刀具中。

- 機械座標系統（MCS）
- 基本座標系統（BCS）
- 工件座標系統（WCS）
- 刀具座標系統（TCS）
- 動態轉換的刀具座標系統（KCS）

句法

```
TOWSTD
TOWMCS
TOWWCS
TOWBCS
TOWTCS
TOWKCS
```

意義

TOWSTD	刀長磨損值中偏移的初始設定值
TOWMCS	機械座標系統（MCS）中刀長偏移
TOWWCS	工件座標系統（WCS）中刀長偏移
TOWBCS	基本座標系統（BCS）中刀長偏移
TOWTCS	刀把參考點上的刀長偏移（可定向刀具）
TOWKCS	刀具頭上的刀長偏移（動態轉換）

其它資訊

區分特性

最重要的區分特性顯示於下表中：

G 代碼	磨損值	生效可定向刀把
TOWSTD	初始值，刀長	磨損值受制於旋轉。
TOWMCS	機械座標系統（MCS）中的磨損值。若未啟用可定向的刀把，則 TOWMCS 與 TOWSTD 完全相同。	僅旋轉在不計入磨損下所得出的刀長向量。
TOWWCS	磨損值在工件座標系統（WCS）中轉換成機械座標系統（MCS）。	在不計入磨損下如 TOWMCS 般計算刀具向量。
TOWBCS	磨損值會在基準座標系統（BCS）中轉換成機械座標系統（MCS）。	在不計入磨損下如 TOWMCS 般計算刀具向量。
TOWTCS	磨損值在工件座標系統中轉換成機械座標系統（MCS）。	在不計入磨損下如 TOWMCS 般計算刀具向量。

TOWWCS、TOWBCS、TOWTCS：將磨損向量加入刀具向量中。

線性轉換

僅在由 BCS 線性轉換產生機械座標系統（MCS）時，在機械座標系統（MCS）中定義刀長才有意義。

非線性轉換

例如，若以 TRANSMIT 啟用非線性轉換，則指定機械座標系統（MCS）為所求座標系統時，即自動使用 BCS。

無動態轉換及可定向刀把

若無啟用動態轉換及可定向刀把，則會結合其他四個座標系統（工件座標系統（WCS）除外）。所以只有工件座標系統（WCS）與其他系統不同。由於只需評估刀長，座標系統間的轉譯互不相關。

參考資料：
如需有關刀具補正的更多資訊，請參閱：
功能手冊基本功能；刀具偏移（W1）

計算中包含磨損值

設定資料 **SD42935 $SC_WEAR_TRANSFORM** 定義三個磨損元件之一：

- 磨損
- 總偏移微調
- 總偏移粗調

若已啟用下列 G 代碼之一，則應受制於使用轉接頭轉換或可定向的刀把所進行的旋轉：

- TOWSTD 預設設定
  刀長中的偏移
- TOWMCS 磨損值
  在機械座標系統（MCS）中

- TOWWCS 磨損值
  在工件座標系統（WCS）中
- TOWBCS 磨損值（BCS）
  在基本座標系統中
- 在刀盤參考（T 刀把參考）工具座標系統中的 TOWTCS 磨損值
- 動態轉換刀具頭座標系統中的 TOWKCS 磨損值

說明

各個磨損元件的評估（指派給幾何軸，符號評估）受下列項目影響：
- 有效平面
- 轉接頭轉換
- 下列設定資料：
  - SD42910 $SC_MIRROW_TOOL_WEAR
  - SD42920 $SC_WEAR_SIGN_CUTPOS
  - SD42930 $SC_WEAR_SIGN
  - SD42940 $SC_TOOL_LENGTH_CONST
  - SD42950 $SC_TOOL_LENGTH_TYPE

7.3.4 刀長及平面變更

功能

設定資料 SD42940 $SC_TOOL_LENGTH_CONST 未設定等於零點時，即可在變更平面時，指派刀長元件－例如，長度、磨損及基本尺寸－至車削及研磨刀具的幾何軸。

SD42940 $SC_TOOL_LENGTH_CONST

設定資料**不等於**零點：

變更加工平面時，不會變更指派刀長元件（長度、磨損及刀具基本尺寸）至幾何軸（G17 - G19）。

下表顯示指派給車削及研磨刀具（刀具類型 400 到 599）幾何軸的刀長元件：

內容	長度 1	長度 2	長度 3
17	Y	X	Z
*)	X	Z	Y
19	Z	Y	X
-17	X	Y	Z
-18	Z	X	Y
-19	Y	Z	X

*）不等於零也不等於列出的六個值，會評估為值 18。

下表顯示指派給其他刀具（刀具類型<400 或>599）幾何軸的刀長元件：

操作平面	長度 1	長度 2	長度 3
*)	Z	Y	X
18	Y	X	Z
19	X	Z	Y
-17	Z	X	Y
-18	Y	Z	X
-19	X	Y	Z

*）不等於零也不等於列出的六個值，會評估為值 17。

說明

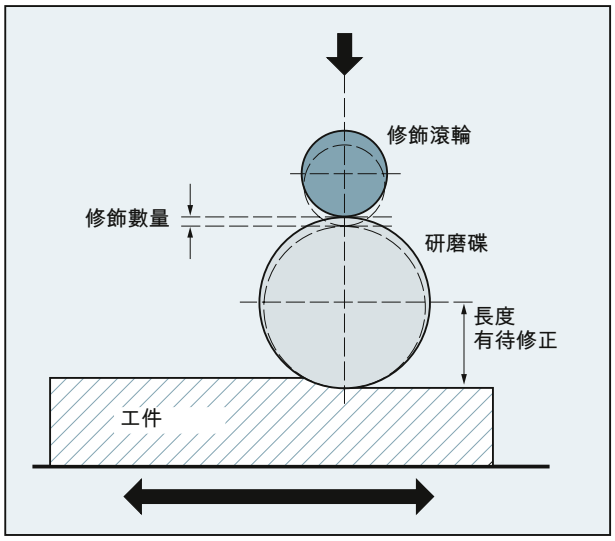
表中代表的意義為，假設最多用 X、Y、Z 指定 3 個幾何軸。以軸順序而非軸識別碼決定補正與軸間的指派。

7.4 線上刀具偏移（PUTFTOCF、FCTDEF、PUTFTOC、FTOCON、FTOCOF）

功能

當"線上刀具偏移量"函數生效時，從加工所衍生的刀長偏移，會立即被套用至研磨刀具。

一個應用範例為 CD 修飾，其研磨輪以平行於加工方向的方式修飾。



可從加工通道或平行通道（修整器通道）變更刀長偏移。

線上刀具偏移量，係依照修飾處理的時間，使用數個函數將其寫入：

- 連續寫入，非模態（PUTFTOCF）

  以 PUTFTOCF，修飾會和加工同時執行。

  根據一次、二次或三次的多項式函數，在加工通道中持續變更刀具偏移，多項式函數需先以 FCTDEF 定義。

  PUTFTOCF 總是非模態；換句話說，其對後續移動單節具有影響。
- 持續寫入，模態：ID=1 DO FTOC（請參考"線上刀具偏移（FTOC） (頁 523)"）
- 分散寫入（PUTFTOC）

  以 PUTFTOC，修飾不會在平行的通道中，和加工同時執行。以 PUTFTOC 具體指定偏移值，在目的地通道中立即產生影響。

說明

線上刀具偏移僅套用於研磨刀具。

句法

在目的地通道中，啟動/停用線上刀具偏移量：

```
FTOCON
...
FTOCOF
```

寫入線上刀具偏移：

- 持續，非模態：

```
FCTDEF（<函數>，<LLimit>，<ULimit>，<a0>，<a1>，<a2>，<a3>）
PUTFTOCF（<函數>，<參考值>，<刀具參數>，<通道>，<主軸>）
...
```

- 分散：

```
PUTFTOC（<偏移值>，<刀具參數>，<通道>，<主軸>）
...
```

意義

FTOCON: 啟用線上刀具偏移
FTOCON 必須被寫入至線上刀具偏移發生影響的通道中。

FTOCOF: 取消線上刀具偏移
使用 FTOCOF 時，不再適用偏移，然而會在切削稜邊的專屬偏移資料中，修正使用 PUTFTOC 寫入的完整值。

注意：
In order to 確定停用線上刀具偏移量，刀具（T...）仍需要在 FTOCOF 之後，被選擇或取消選擇。.

FCTDEF: FCTDEF 係用來定義 PUTFTOCF 的多項式函數
參數：

<函數>:	多項式函數的編碼 類型： INT
<LLimit>:	下限值 類型： REAL
<ULimit>:	上限值 類型： REAL
<a0> ... <a3>:	多項式函數的係數 類型： REAL

PUTFTOCF: 呼叫"線上刀具偏移量的連續非模態寫入"函數

參數:

<函數>: 多項式函數的編碼

類型: INT

注意:
必須符合用於 FCTDEF 的設定。

<參考值>: 從要被衍生出的偏移量（例如，更改實際值），所得到變數參考值。

類型: VAR REAL

<刀具參數: 偏移值要被加入的磨耗參數（長度 1，2 或 3）之號碼。

類型: INT

<通道>: 線上刀具偏移量發生影響的通道號碼。

類型: INT

注意:
若偏移量沒有在有效通道中產生效果，則只需指定通道號碼。

<主軸>: 線上刀具偏移量發生影響的主軸號碼。

類型: INT

注意:
若偏移量要被套用至無效的研磨輪，而不是正在使用中的有效刀具，則只需指定主軸號碼。

PUTFTOC: 呼叫"線上刀具偏移量的分散寫入"函數。

參數:

<偏移值>: 要被加入至磨耗參數的偏移值。

類型: REAL

<刀具參數: 請參考 PUTFTOCF

<通道>: 線上刀具偏移量發生影響的通道號碼。

類型: INT

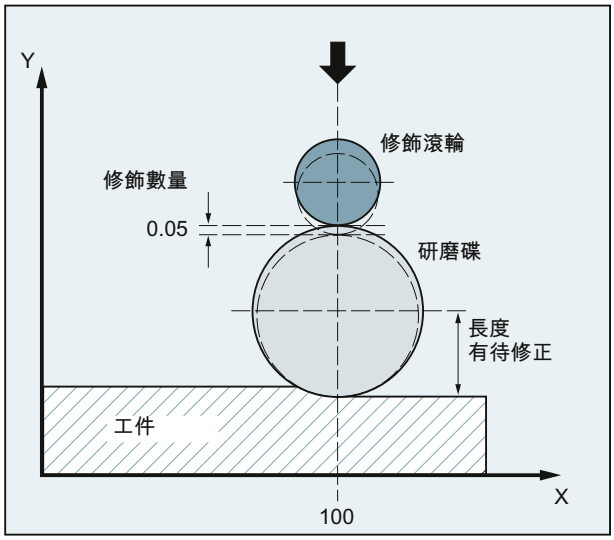
<主軸>: 請參考 PUTFTOCF

範例

平面磨床，以：

- Y：研磨輪的進給軸
- V：修飾滾輪的進給軸
- 加工通道通道 1 含軸 X、Z、Y
- 修飾通道：通道 2 含軸 V

在開始研磨動作後，研磨輪會被以 0.05 X100 的量來修飾。修飾量會被以"連續寫入線上偏移量"，套用在研磨刀具上。



通道 1 中加工程式：

程式碼	註解
...	
N110 G1 G18 F10 G90	; 基本位置。
N120 T1 D1	; 選擇目前刀具。
N130 S100 M3 X100	; 主軸開啟，移動至啟動位置。
N140 INIT（2，"DRESS"，"S"）	; 於通道 2 選擇修飾程式。
N150 START（2）	; 於通道 2 啟動修飾程式。
N160 X200	; 移動至目標點。
N170 FTOCON	; 啟動線上偏移。
N… G1 X100	; 其他加工。
N… M30	

於通道 2 修飾程式：

程式碼	註解
…	
N40 FCTDEF（1，–1000，1000，–$AA_IW[V]，1）	; 定義函數：直線，斜度= 1
N50 PUTFTOCF（1，$AA_IW[V]，3，1）	; 持續寫入線上刀具偏移：由 V 軸動作衍生，在通道 1 中補償有效研磨輪的長度 3。
N60 V–0.05 G1 F0.01 G91	; 修飾的進給動作，PUTFTOCF 只在本單節中有效。
…	
N… M30	

其他資訊

關於線上 TO 的一般資訊

在持續寫入（為各插補循環）後啟用評估函數的情況下，在磨損記憶體中附加計算各項變更（以避免設定點跳躍）。

在兩種情況下：線上刀具偏移可在各主軸上及磨損參數的長度 1、2 或 3 上動作。

參考目前工作平面指派到幾何軸的長度。

使用刀具資料，以 GWPSON 與 TMON 將主軸指派至刀具，除非這是生效的研磨輪。

在未啟用刀具上，偏移套用於目前刀具側邊或刀具左側的磨損參數。

說明

當多個刀具側邊的偏移相同時，應透過鏈結規則，自動傳輸該值到第二個刀具側邊。

若為加工通道定義線上偏移，則目前刀具的磨損值不能於加工程式的通道中、或透過操作員動作進行變更。

除刀具監控（TMON）外，套用線上刀具偏移也與恆定研磨輪周邊速度（GWPS）相關。

7.5 啟動 3D 刀具補正(CUT3DC..., CUT3DF...)

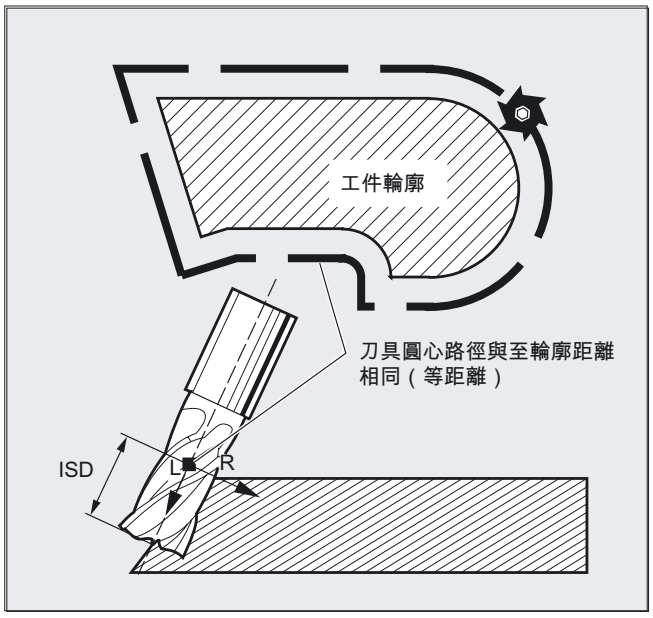
7.5.1 啟動 3D 刀具偏移（CUT3DC、CUT3DF、CUT3DFS、CUT3DFF、ISD）

功能

將刀具方向的變更，納入圓柱刀具的刀具半徑補正之中。

相同的程式設計指令適用於 3D 刀具半徑補正，以及 2D 刀具半徑補正。使用 G41/G42，依動作方向指定左 / 右偏移。永遠使用 NORM 控制逼近回應。3D 半徑補正只在選擇 5 軸轉換時有效。

3D 刀具半徑補正也稱為 5D 刀具半徑補正，因為在此情況下，空間中的刀具方向可使用 5 度自由度。



2 1/2D 及 3D 刀具半徑補正間的差距

在 3D 刀具半徑補正中，可變更刀具方向。使用 2 1/2D 刀具半徑補正，為假設僅使用恆定方向的刀具。

句法

```
CUT3DC
CUT3DFS
CUT3DFF
CUT3DF
ISD=<值>
```

意義

CUT3DC	啟用圓周銑削的 3D 半徑偏移
CUT3DFS	含恆定方向之平面銑削的 3D 刀具偏移。以 G17 - G19 決定刀具方向，且不受框架影響。
CUT3DFF	含恆定方向之平面銑削的 D 刀具偏移。刀具方向為以 G17 - G19 定義的方向，且在某些情況下，由框架旋轉。
CUT3DF	含方向變更之平面銑削的 3D 刀具偏移（僅使用啟用的 5 軸轉換）。
G40 X... Y... Z...	停用：含幾何軸的線性單節 G0/G1
ISD	插入深度

說明

指令為模態有效，並寫入與 CUT2D 及 CUT2DF 同一群組中。直到在目前平面中執行下一個動作時，才會取消選擇指令。永遠套用於 G40，並獨立於 CUT 指令。

在 3D 刀具半徑補正中允許中間單節。該定義適用 2 1/2D 刀具半徑補正。

補充條件

- **G450/G451 及 DISC**

  圓弧單節皆永遠插入外角。G450/G451 無意義。
  不會評估 DISC 指令。

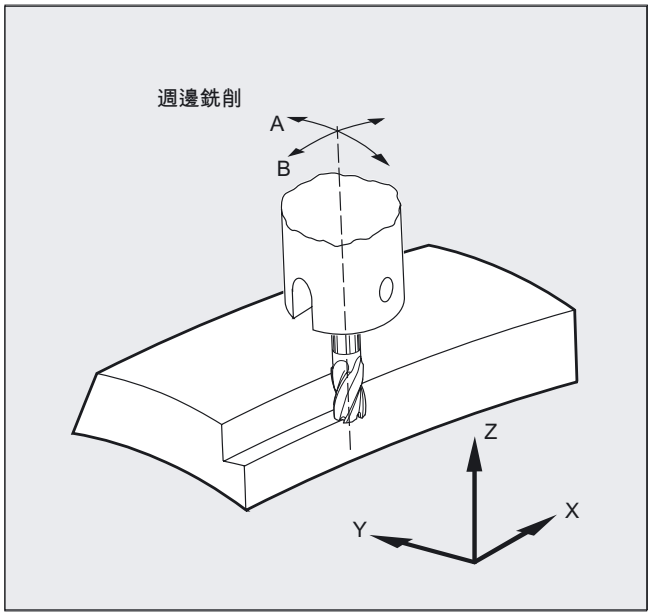
範例

程式碼	註解
N10 A0 B0 X0 Y0 Z0 F5000	
N20 T1 D1	; 刀具呼叫，呼叫刀具偏移值。
N30 TRAORI（1）	; 轉換選擇
N40 CUT3DC	; 3D 刀具半徑補正選項
N50 G42 X10 Y10	; 刀具半徑補正選項
N60 X60	
N70 …	

7.5.2 3D 刀具偏移周邊銑削、平面銑削

圓周銑削

透過定義路徑（引導線）及對應方向，進行用於此處的銑削類型。在該類型加工中，與路徑上的刀具形狀無關。尤其已確定刀具切入點的半徑。

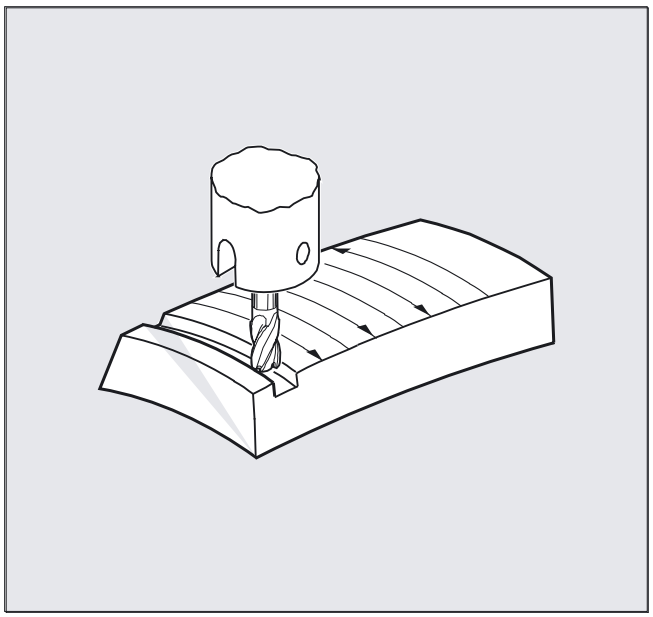


說明

函數 3D TRC 僅限用於圓柱刀具。

平面銑削

該類型 3D 銑削，在工件表面上需有 3D 路徑逐行說明。刀具形狀及尺寸需納入計算考量－一般在 CAM 中執行。後處理器寫入工件程式－除 NC 單節外－至方向（啟用 5 軸轉換）及所需 3D 刀具偏移的 G 代碼。代表機台操作員可使用較小一點的刀具，脫離用來計算 NC 路徑的刀具。



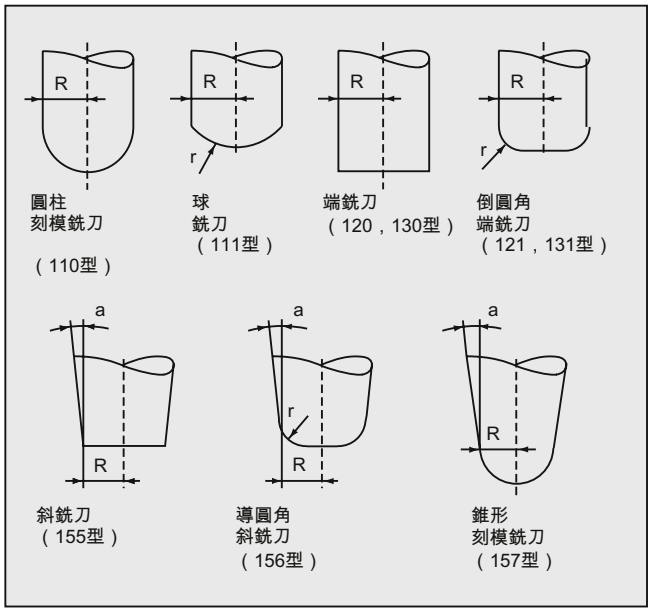
範例：

NC 單節係使用 10 毫米銑削刀具來計算。在此情況下，可使用直徑 9.9 毫米的銑削刀具加工－可藉以消除修改的粗糙輪廓。

7.5.3 平面銑削的 3D 刀具偏移刀具形狀及刀具資料

銑削形狀、刀具資料

刀具形狀的概觀，可用於下面列出的平面銑削操作及刀具資料臨界值。刀具軸的形狀不列入考量。刀具類型 120 和 156 的效果相同。



若在 NC 程式中，指定與圖示中不同的型號，則系統會自動使用刀具型號 110（圓柱刻模銑削刀具）。若違反刀具資料限制，則輸出警報。

銑刀類型	型號	R	r	a
圓柱刻模銑刀	110	> 0	-	-
圓頭銑刀	111	> 0	>R	-
端銑刀、角頭銑刀	120, 130	> 0	-	-
端銑刀，含轉角磨圓的角頭銑刀	121, 131	>r	> 0	-
斜銑刀	155	> 0	-	> 0
倒圓角斜銑刀	156	> 0	> 0	> 0
錐形刻模銑刀	157	> 0	-	> 0

R =軸半徑（刀具半徑）

r =轉角半徑

a =刀具縱向軸與環面上端之間的角度

- =不評估

刀具資料	刀具參數	
刀具尺寸	幾何	磨損
R	$TC_DP6	$TC_DP15
r	$TC_DP7	$TC_DP16
a	$TC_DP11	$TC_DP20

刀長偏移

刀具刀尖為長度偏移的參考點（交點縱向軸 / 表面）。

3D 刀具偏移，換刀

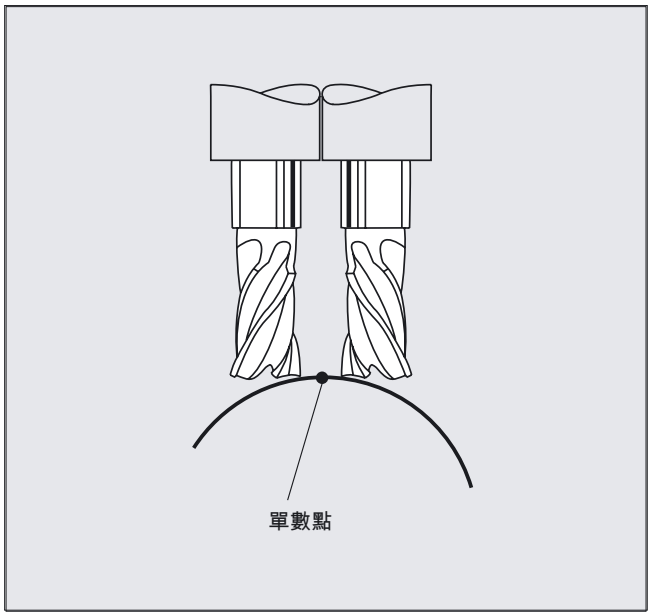
含修改尺寸（R，r，a）或不同軸的新增刀具，僅能以 G41 或 G42 程式設計指定（G40 變化至 G41 或 G42，重新程式設計 G41 或 G42）。該項規則不適用於其他任何刀具資料，例如，刀長，因此無需重新程式設計 G41 或 G42 載入刀具。

7.5.4 路徑、路徑曲率、插入深度上的 3D 刀具偏移補正（CUT3DC、ISD）

功能

路徑上補正

關於平面銑削，建議在刀具表面上的接點“跳躍”時，檢查其情況，如範例中右方所示，正使用垂直定位刀具加工凸起的表面。範例中所示的應用應視為邊界情況。



由控制系統監控該邊界情況，該控制系統可根據刀具與一般表面向量間的角度逼近動作，偵測到機台原點的瞬間變更。控制在該位置插入線性單節，以執行動作。

根據儲存於機械參數中，其允許的側邊角度範圍，計算該線性單節。若違反儲存於機械參數中的臨界值，則系統輸出警報。

路徑曲率

未監控路徑曲率。在此情況下，亦建議僅使用不違反輪廓的刀具類型。

插入深度（ISD）

僅在已啟用 3D 刀具半徑補正時，評估插入深度 ISD。

使用程式指令 ISD（插入深度）程式設計圓周銑削操作的刀具插入深度。即可在刀具外表面上變更機台原點的位置。

句法

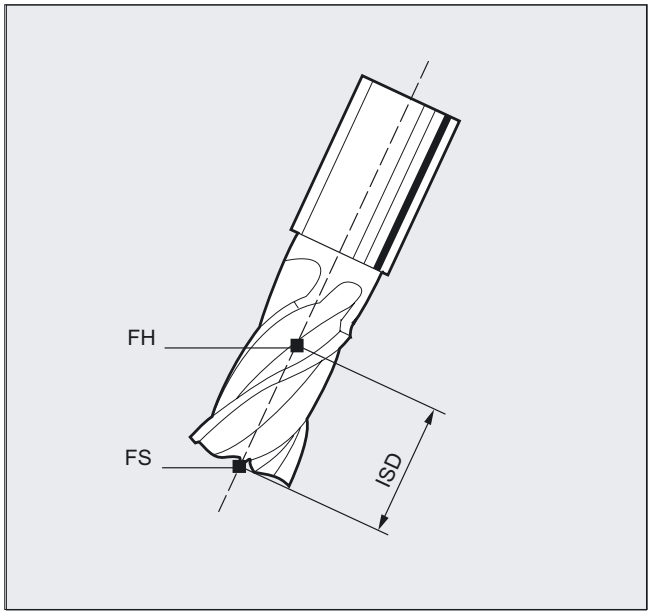
3D 刀具補正圓周銑削
CUT3DC
ISD=<值>

意義

CUT3DC	啟動圓周銑削的 3D 刀具偏移，例如，含傾斜側壁之口袋型銑削。
ISD	使用 ISD 指令指定銑削刀尖（FS）與銑削刀具建構點（FH）間的間隙<值>。

銑削刀具參考點

透過將程式設計機台原點投射到刀具軸上，以取得銑削刀具參考點（FH）。



其它資訊

為含 CUT3DC 之圓周銑削，以傾斜側壁進行口袋型銑削

在此 3D 刀具半徑補正中，銑削半徑的偏差，係由朝欲加工之表面法線向量的進給補償。若插入深度 ISD 維持不變，則銑削刀具平面所在的平面也維持不變。例如，用小於標準刀具半徑的銑削刀具無法到達口袋基部－此處亦為限制表面。對於自動刀具進給，控制系統需知道限制表面，請參閱"含限制表面的 3D 圓周銑削"一節。

有關碰撞監控的其他資訊，請參閱：
參考資料：
程式設計手冊基本概念；“刀具偏移”一章。

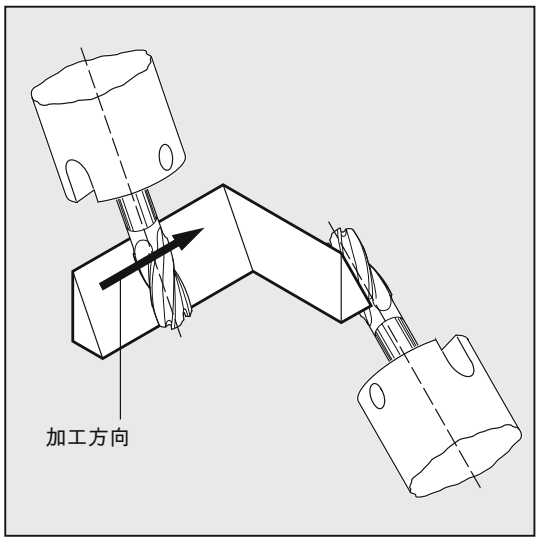
7.5.5 3D 刀具偏移內 / 外角及交點步驟（G450/G451）

功能

內角 / 外角

分別處理內角及外角。內角及外角依刀具方向而定。

例如，當方向在角上變更時，則可能在加工進行時，變更轉角的類型。一旦發生，便會以錯誤訊息取消加工操作。



句法

```
G450
G451
```

意義

G450	變化圓弧（刀具以圓弧路徑繞工件轉角移動）
G451	等距路徑（刀具自工件轉角返回）的交點

其它資訊

3D 補正的交點步驟

進行 3D 圓周銑削時，可即即評估 G 代碼 G450/G451，即，可逼近偏移曲線的交點。到 SW 4 為止，皆在外角插入圓弧。交點步驟對一般由 CAD 產生的 3D 程式特別有利。該程式常由短的直單節構成（以模擬平滑化曲線），其中相鄰單節間的變化幾乎為切線。

目前為止，在輪廓外部使用刀具半徑補正時，通常為插入圓弧繞行外角。因為幾乎為切線變化，所以單節可能很短，而導致不樂見的速率下降。

在此情況下，類比至 2 ½ D 半徑補正，延伸涉入的兩條曲線，並逼近兩延伸曲線的交點。

在與刀具方向垂直的平面中，於角落由延伸兩相關單節的偏移曲線，及定義兩單節的交點，來決定交點。若無此交點，則照先前方式處理轉角，即插入圓弧。

如需有關交點步驟的更多資訊，請參閱：
參考資料：
功能手冊，特殊函數；3D 刀具半徑補正（W5）

7.5.6 3D 刀具偏移含限制表面的 3D 圓周銑削

適應 CAD 程式設計條件的 3D 圓周銑削

由 CAD 系統產生的 NC 程式，一般以大量短線性單節來模擬標準刀具的中心路徑。為確保以該方式產生的許多工件輪廓單節能盡量精確地套用原始輪廓，需對工件程式進行某些更改。

工件程式中不再提供欲達成最佳補正所需的重要資訊，需使用適當方式取代。以下為補正重要變化的一般手法，可直接在工件程式中或在決定實際輪廓時補償（例如，使用刀具進給）。

應用

除一般應用程式會取代標準刀具外，實際刀具會說明中心點路徑，也說明含 3D 刀具補正的圓柱刀具。在此情況下，已程式設計的路徑會參考加工表面的輪廓。相關的限制表面獨立於刀具。就跟傳統刀具半徑補正一樣，使用整個半徑來計算限制表面的垂直偏移。

7.5.7 3D 刀具偏移將限制表面納入考量（CUT3DCC、CUT3DCCD）

功能

含實際刀具的 3D 圓周銑削

在刀具方向裡有一持續或經常變更的 3D 圓周銑削中，經常依已定義的標準刀具程式設計刀具中心點路徑。由於在實務上經常無適當的標準刀具可用，故可使用與標準刀具偏差不大的刀具。

CUT3DCCD 將程式設計標準刀具所定義，有實際差距的刀具限制表面考慮在內。NC 程式會定義標準刀具的中心點路徑。

CUT3DCC 使用圓柱刀具時，將程式設計標準刀具可能到達的限制表面考慮在內。NC 程式定義加工表面的輪廓。

句法

```
CUT3DCCD
CUT3DCC
```

意義

CUT3DCCD	為含限制表面的圓周銑削，在刀具中心點路徑上以不同刀具，啟用 3D 刀具偏移：進給至限制表面。
CUT3DCC	為含限制表面的圓周銑削，以 3D 半徑補正啟用 3D 刀具偏移：加工表面的輪廓

說明

使用 G41、G42 的刀具半徑補正

若在啟用 CUT3DCCD 或 CUT3DCC 時，程式設計使用 G41、G42 的刀具半徑補正，則也需啟用“方向轉換”選項。

含倒圓角磨圓的標準刀具

以刀具參數$TC_DP7 定義使用標準刀具的倒圓角磨圓。刀具參數$TC_DP16 說明與標準刀具相比下，實際刀具的倒圓角磨圓偏差。

範例

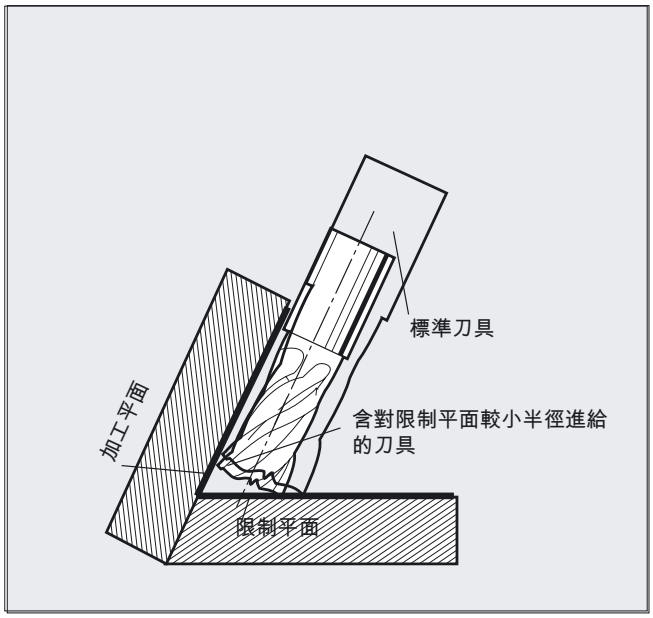
半徑小於標準刀具的環型銑刀刀具尺寸。

刀具類型	R = 端半徑	r = 倒圓角半徑
含倒圓角磨圓的標準刀具	R = $TC_DP6	r = $TC_DP7
具角磨圓的實際刀具： 刀具類型 121 及 131 環形銑削刀具（軸銑削刀具）	R' = $TC_DP6 + $TC_DP15 + OFFN	r' = $TC_DP7 + $TC_DP16
在本範例中，$TC_DP15 + OFFN 及$TC_DP16 皆為負數。 評估刀具型號（$TC_DP1）。		
只認可具圓柱端（圓柱或端銑刀）的銑刀類型、環型銑刀（型號 121 及 131），和在有限情況下的柱形模銑刀（型號 110）。	已認可的銑刀類型，倒圓角半徑 r 與端半徑 R 完全相同。其他所有已認可的刀具類型皆轉譯為圓柱銑刀，且不評估倒圓角磨圓指定的尺寸。	
已認可型號 1 - 399 的所有刀具，類型 111 及 155 到 157 除外。		

其它資訊

含進給直到限制表面的刀具中心點路徑 CUT3DCCD

若使用半徑小於適當標準刀具的刀具，則使用銑削刀具，依縱向進給持續加工直到口袋底部（基部）為止。刀具會從加工表面及限制表面形成的倒圓角，盡可能切削材料。此處涉到結合圓周及平面銑削的加工類型。對於半徑漸增的刀具，類比於含較小半徑的刀具，進給在反方向。



與所有群組 22 之 G 代碼的其他刀具偏移相反，為 CUT3DCCD 指定的刀具參數$TC_DP6 不會影響刀具半徑及其產生的補正。

補正偏移為下列項目的總和：

- 刀具半徑的磨損值（刀具參數$TC_DP15）
- 及程式設計刀具偏移 OFFN，以計算與限制表面垂直的偏移。

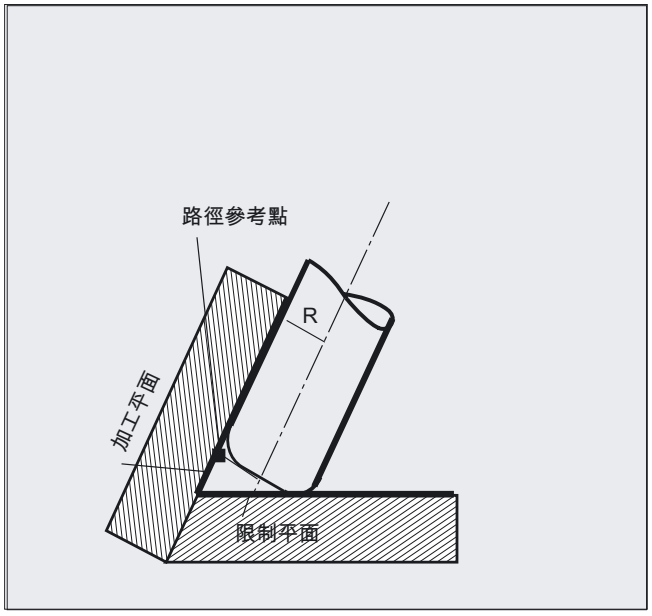
產生的工件程式未指定欲加工的平面在路徑的右側或左側。因此假設半徑為正值，而原始刀具的磨損值為負值。負磨損值永遠以較小的直徑定義刀具。

使用圓柱刀具

使用圓柱刀具時，僅在自銳角（小於 90 度）加工表面及限制表面時，才需進給。若使用環面銑削刀具（含倒圓角的圓柱），銳角及鈍角皆需縱向中的刀具進給。

含 CUT3DCC 的 3D 半徑補正，加工表面的輪廓

若環面銑削刀具啟用 CUT3DCC，則已程式設計的路徑，會參考具有相同直徑的虛設圓柱銑削刀具。得出的路徑參考點顯示於下列環面銑削刀具的圖示中。



即使在同一單節中，加工與限制表面間的角度可能從銳角變成鈍角，或從鈍角變銳角。

實際使用的刀具可能大於或小於標準刀具。然而，所得的倒圓角半徑不得為負值，且需保留所得的刀具半徑符號。

CUT3DCC，NC 工件程式會參考加工表面的輪廓。至於傳統的刀具半徑補正，使用包含下列總和的總刀具半徑：

- 刀具半徑（刀具參數$TC_DP6）
- 磨損值（刀具參數$TC_DP15）
- 及程式設計以計算限制表面之垂直偏移的刀具偏移 OFFN。

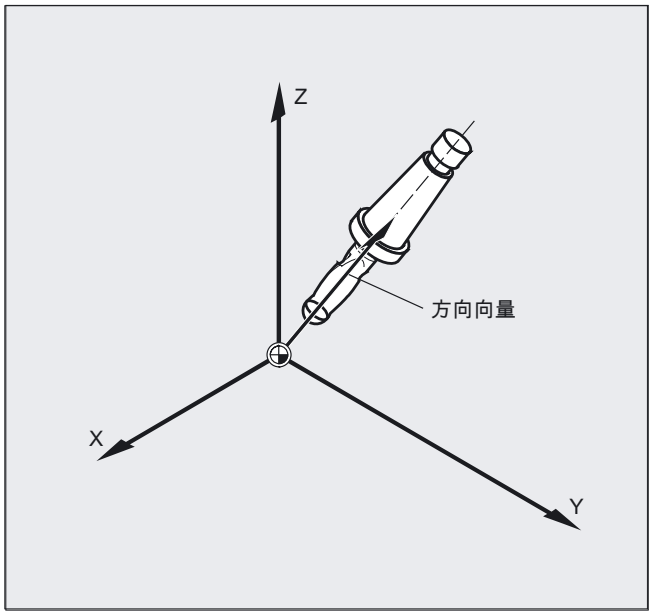
依下列兩個值的差距來決定限制表面的位置：

- 標準刀具的尺寸
- 刀具半徑（刀具參數$TC_DP6）

7.6 刀具方向（ORIC、ORID、OSOF、OSC、OSS、OSSE、ORIS、OSD、OST）

功能

刀具方向一詞定義在空間中刀具的幾何基準。在五軸機台刀具上的刀具方向可以透過程式指令設定。



以 OSD 和 OST 啟動的倒圓角動作會依刀具方向的插補類型而以不同方式形成。

如果已啟用向量插補，已平滑化的方向特性也是使用向量插補進行插補。另一方面，如果已啟用旋轉軸插補，是直接使用旋轉軸動作平滑化方向。

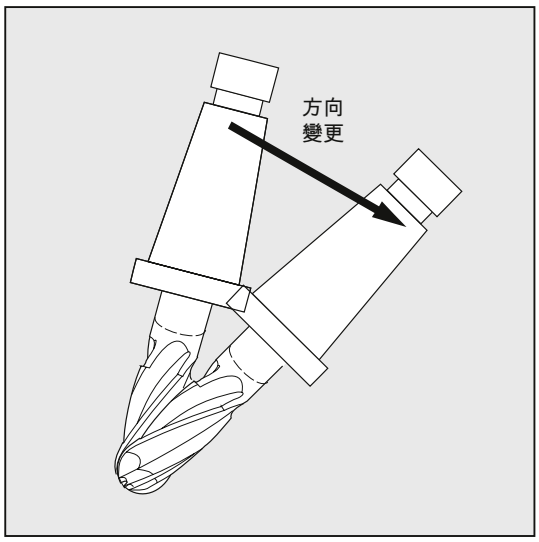
程式設計

將方向變更程式設計：

刀具方向的變更可以依下列方式程式設計：

- 直接將旋轉軸 A、B、C 程式設計（旋轉軸插補）
- 尤拉角或 RPY 角
- 方向向量（指定 A3 或 B3 或 C3 進行向量插補）
- LEAD/TILT（平面銑削）

參考座標系統是機械座標（ORIMKS）或目前的工件機械座標（ORIWKS）。



程式設計刀具方向：

指令	意義
ORIC:	方向與路徑動作平行
ORID:	方向與路徑動作連續進行
OSOF:	沒有方向平滑化
OSC:	經常定向
OSS:	只在單節開始處進行方向平滑化
OSSE:	只在單節開始與結尾處進行方向平滑化
ORIS:	已啟動方向平滑化的方向變更速率是每毫米以度計算（對 OSS 與 OSSE）有效
OSD:	以設定資料來指定平滑化距離的方向平滑化： SD42674 $SC_ORI_SMOOTH_DIST
OST:	使用設定資料，透過為向量插補以度指定角度允差來平滑化方向： SD42676 $SC_ORI_SMOOTH_TOL 進行旋轉軸插補時，就會取指定的允差做為方向軸的最大變異。

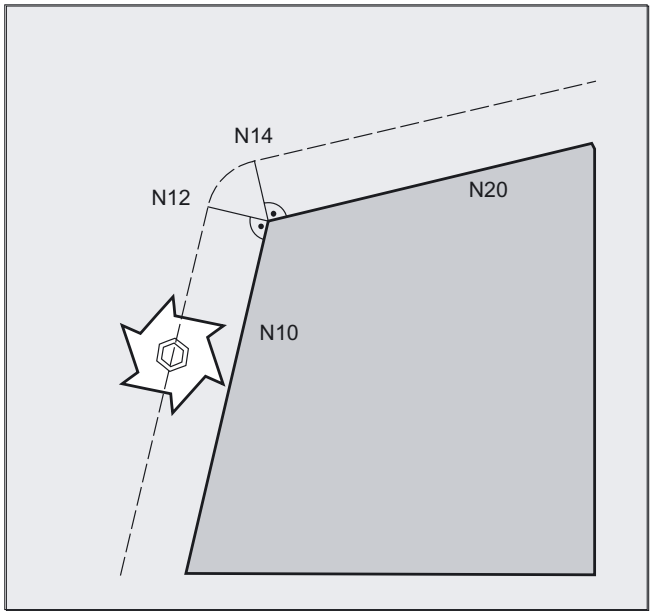
說明

所有用於平滑化刀具方向的指令（OSOF，OSC，OSS，OSSE，OSD，與 OST）皆歸納在 G 系列功能群組 34 中。他們為模態；換句話說，一次只有一個指令可以生效。

範例

範例 1：ORIC

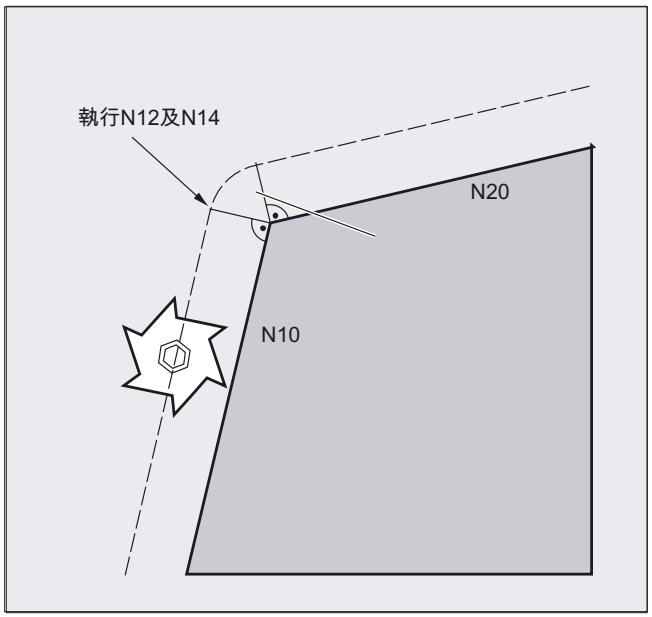
如果已啟用 ORIC，而且有兩個以上單節已將移動單節 N10 與 N20 之間的方向變更程式設計（例如，A2=... B2=... C2=...）則會根據角度中的絕對變更，將插入的圓弧單節散佈在這些中間單節之間。



程式碼	註解
ORIC	
N8 A2=… B2=… C2=…	
N10 X… Y… Z…	
N12 C2=… B2=…	; 插入外部角的圓弧單節是散佈於 N12 與 N14 之間，對應於方向中的變更。圓弧動作和方向變更是平行執行。
N14 C2=… B2=…	
N20 X =…Y=… Z=… G1 F200	

範例 2：ORID

如果已啟用 ORID，則兩個移動單節之間的所有單節都會在第一個移動單節結束時執行。具有指定方向的圓弧單節就在第二個移動單節之前執行。



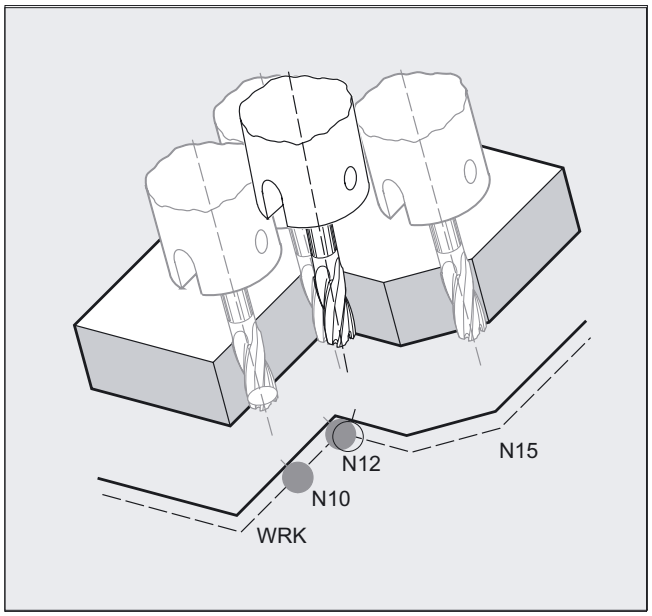
程式碼	註解
ORID	
N8 A2=… B2=… C2=…	
N10 X… Y… Z…	
N12 A2=… B2=… C2=…	; N12 和 N14 單節是在 N10 結束時執行，然後再執行具有實際方向的圓弧單節。
N14 M20	; 輔助說明函數等
N20 X… Y… Z…	

說明

用來在外角變更方向的方法是使用在外角的第一個移動單節中啟用的程式指令來決定。

沒有變更方向時：如果沒有在單節界限變更方向，刀具的斷面是圓弧，會同時接觸兩個輪廓。

範例 3：在內角變更方向



程式碼

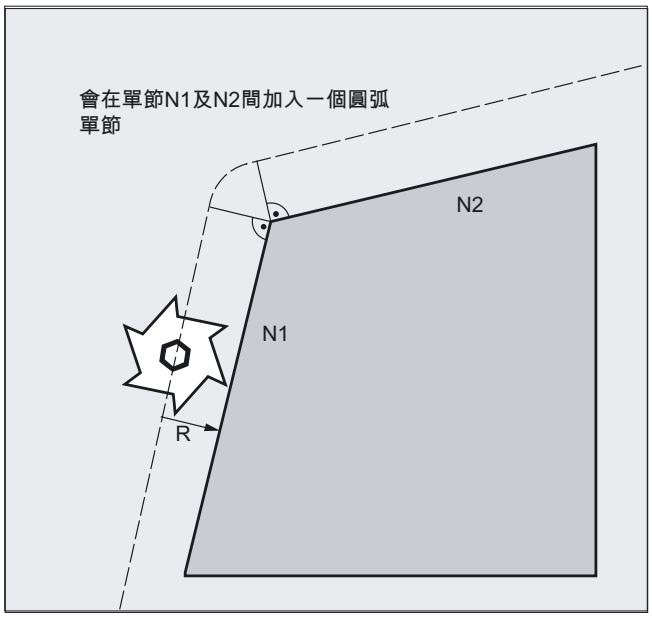
```
ORIC
N10 X …Y… Z… G1 F500
N12 X …Y… Z… A2=… B2=… C2=…
N15 X …Y… Z… A2=… B2=… C2=…
```

其它資訊

在外角的行為

具有銑刀半徑的圓弧單節永遠都插入外角。

ORIC 和 ORID 程式指令是用來決定 N1 和 N2 單節之間已程式設計的方向變更是在插入圓弧單節之前處理，或是同時處理。



如果外角需要方向變更，可以與插補同時執行，或是分別配合路徑動作執行。

當 ORID 已程式時，插入的單節會在沒有路徑動作之前先執行。產生角的圓弧單節就插入於兩個移動單節的第二個之前。

如果在外部角插入多個方向單節，而且已選擇 ORIC，圓弧動作就會根據方向變更的絕對值，散佈在各個插入的單節之間。

以 OSD 或 OST 平滑化方向

使用 G642 調合時，輪廓軸的最大變異與方向軸很不相同。兩個之中較小的允差決定平滑化動作類型及 / 或角度允差，以堅定地平滑化方向特性，而不必接受較高的輪廓偏差。

OSD 與 OST 可啟動為"很全面性地"平滑極小的偏差，從具有特定平滑化距離的方向特性，以及不具有具體輪廓偏差的有角度允差。

說明

與使用（G642 的輪廓磨圓（和方向特性）處理不同，使用 OSD 及 / 或 OST 磨圓方向時，不會形成獨立單節，而是直接將磨圓動作加入已程式設計的原始單節中。

使用 OSD 和 / 或 OST 時，如果刀具方向的插補類型中有變更（向量→旋轉軸，旋轉軸→向量）時，就無法磨圓單節變化。必要時，這些單節變化可以使用標準的磨圓函數 G641、G642 和 G643 進行磨圓。

7.7 自由定義 D 編號,切削刀刃編號

7.7.1 任意指派 D 數量、刀刃數量（CE 位址）

D 數量

D 數量可用來做為輪廓數量。您也可以經由位址 CE，定出刀刃數量的位址。您可以使用系統變數$TC_DPCE，定義刀刃數量。

預設值：補正數量==刀具稜邊數量

機械參數是用來定義最大的 D 數量（刀刃數量），以及每個刀具最大的刀刃數量（→機台製造 OEM）。只有在最大刀刃數量（MD18105）指定為大於每個刀具的刀刃數量（MD18106）時，下列指令才實際可用。請參閱機台製造商規格。

說明

除了相對 D 數量配置以外，也可以指派 D 數量做為“均一”或“絕對”D 數量（1-32000），而不需要參考 T 數量（在“均一 D 數量結構”函數之中）。

參考

功能手冊基本功能；刀具偏移（W1）

7.7.2 任意指派 D 數量檢查 D 數量（CHKDNO）

功能

您可以使用 CKKDNO 指令，檢查是現有 D 數量是否為唯一指派。在 TO 元件中定義的所有刀具 D 數量不可能出現超過一次。沒有為替換刀具定出允差。

句法

```
state=CHKDNO（Tno1，Tno2，Dno）
```

意義

狀態	= TRUE：	D 數量是唯一地指派給已檢查的區域。
	= FALSE：	發生 D 數量衝撞，或參數無效。Tno1、Tno2 和 Dno 會傳回導致衝撞的參數。這些資料現在可以在工件程式中進行評估。
CHKDNO（Tno1，Tno2）	指定的工件所有 D 數量都已檢查。	
CHKDNO（Tno1）	Tno1 的所有 D 數量都已依其他所有刀具檢查。	
CHKDNO	所有刀具的所有 D 數量都已依其他所有刀具檢查。	

7.7.3 任意指派 D 數量重新命名 D 數量（GETDNO，SETDNO）

功能

您必須指派唯一的 D 數量。刀具的兩個不同刀刃不能有相同的 D 數量。

GETDNO

這個指令會以刀具數量 t 傳回刀具之特定刀刃的 D 數量（ce）；如果輸入的參數沒有 D 數量，就會設定 d=0。如果 D 數量無效，就會傳回大於 32000 的值。

SETDNO

這個指令指派 D 數量的 d 值給刀具 t 的刀刃 ce，此敘述的結果是經由狀態（TRUE 或 FALSE）傳回。如果指定的參數沒有資料單節，就會傳回值 FALSE。句法錯誤會產生警報。D 數量不可明確地設定為 0。

句法

```
d = GETDNO （t，ce）
state = SETDNO （t，ce，d）
```

意義

d	刀具稜邊的 D 數量
t	刀具 T 數量
CE	刀具的刀刃數量（CE 數量）
狀態	指出是否可執行指令（TRUE 或 FALSE）。

重新命名 D 數量的範例

程式設計	註解
$TC_DP2[1.2]=120	;
$TC_DP3[1，2] = 5.5	;
$TC_DPCE[1，2] = 3	; 刀刃數量 CE
...	;
N10 def int DNoOld，DNoNew = 17	;
N20 DNoOld = GETDNO（1，3）	;
N30 SETDNO（1，3，DNoNew）	;

然後，新的 D 值 17 就會指派給刀刃 CE=3。現在刀刃的資料是經由 D 數量 17 定位址；經由系統變數，也在 NC 位址的程式設計中。

7.7.4 任意指派 D 數量決定指定 D 數量的 T 數量（GETACTTD）

功能

您可以使用 GETACTTD 指令，決定與絕對 D 數量相關聯的 T 數量。不會有唯一性的檢查。如果在 TO 元件之中有多個相同的 D 數量，會傳回搜尋中所找到第一個刀具的 T 數量。這個指令不適合搭配“均一”D 數量使用，因為在此情況下，永遠都會傳回值“1”（資料庫中沒有 T 數量）。

句法

```
status=GETACTTD（Tnr，Dnr）
```

意義

Dno	會進行 T 數量搜尋的 D 數量。	
Tno	找到 T 數量	
狀態	值：	意義：
	0	已經找到 T 數量。Tno 包含 T 數量的值。
	-1	指定的 D 數量中沒有 T 數量；Tno=0。
	-2	D 數量不是絕對的。Tno 接到所找到第一個刀具的值，其中包含具有 Dno 值的 D 數量。
	-5	函數因為其他原因而無法執行。

7.7.5 任意指派 D 數量無效的 D 數量（DZERO）

功能

DZERO 指令是在重新用工具加工時用來支援。以這個指令標記的補正資料集不再以 CHKDNO 指令進行驗證。這些資料集可以透過再次使用 SETDNO 設定 D 數量再次存取。

句法

```
DZERO
```

意義

DZERO	將 TO 元件的所有 D 數量標記為無效。

7.8 刀把動力學

先決條件

刀把可以將刀具定向為空間中所有可能方向的條件是：

- 同時有兩個旋轉軸 v_1 和 v_2。
- 旋轉軸彼此垂直相交。
- 刀具縱向軸垂直於第二個旋轉軸 v_2。

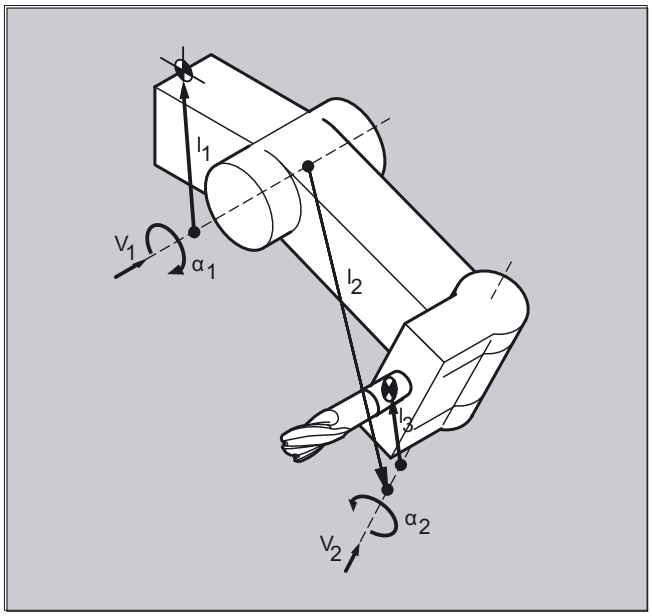
此外，下列先決條件也適用於必須可設定所有可能方向的機台：

- 刀具縱向軸必須垂直於第一個旋轉軸 v_1。

功能

最多兩個旋轉軸 v_1 或 v_2 的刀把動力學是使用 17 個系統變數$TC_CARR1[m]到$TC_CARR17[m]來定義。.刀把的說明包含：

- 離刀把 I_1 第一個旋轉軸的向量距離、從第一個旋轉軸到第二個旋轉軸 I_2 的向量距離、從第二個旋轉軸到刀具 I_3 之參考點的距離。
- 兩個旋轉軸 v_1、v_2 的方向向量。
- 兩個軸的旋轉角度 α_1、α_2。旋轉角度是依旋轉軸向量的觀看方向、正向、旋轉的順時針方向計算。



對含**解析動力學**的機台（刀具和工件都可以旋轉），系統變數已經用輸入延伸

- $TC_CARR18[m] to $TC_CARR23[m].

參數

可定向刀把之系統變數的函數			
指定	**x 元件**	**y 元件**	**y 元件**
l_1 偏移向量	$TC_CARR1[m]	$TC_CARR2[m]	$TC_CARR3[m]
l_2 偏移向量	$TC_CARR4[m]	$TC_CARR5[m]	$TC_CARR6[m]
v_1 旋轉軸	$TC_CARR7[m]	$TC_CARR8[m]	$TC_CARR9[m]
v_2 旋轉軸	$TC_CARR10[m]	$TC_CARR11[m]	$TC_CARR12[m]
旋轉的 α_1 角度旋轉的 α_2 角度	$TC_CARR13[m] $TC_CARR14[m]		
l_3 偏移向量	$TC_CARR15[m]	$TC_CARR16[m]	$TC_CARR17[m]

可定向刀把之系統變數的延伸			
名稱	**x 元件**	**y 元件**	**z 元件**
l_4 偏移向量	$TC_CARR18[m]	$TC_CARR19[m]	$TC_CARR20[m]
軸識別碼旋轉軸 v_1 旋轉軸 v_2	旋轉軸 v_1 和 v_2 的軸識別碼（以零點初始化） $TC_CARR21[m] $TC_CARR22[m]		
動力學類型 **刀**具 **工**件 **混合**模式	$TC_CARR23[m]		
	動力學類型 T ->	動力學類型 P ->	動力學類型 M
	只有刀具可以旋轉（預設值）。	只有工件可以旋轉	工件和刀具都可以旋轉
偏移 旋轉軸 v_1 旋轉軸 v_2	旋轉軸 v_1 和 v_2 的角度度數 $TC_CARR24[m] $TC_CARR25[m]		
角度偏移，旋轉軸 v_1 旋轉軸 v_2	旋轉軸 v_1 和 v_2 之 Hirth 齒系統的偏移度數 $TC_CARR26[m] $TC_CARR27[m]		
角度增量 v_1 旋轉軸 v_2 旋轉軸	旋轉軸 v_1 和 v_2 之 Hirth 齒系統的偏移度數 $TC_CARR28[m] $TC_CARR29[m]		
最小位置旋轉軸 v_1 旋轉軸 v_2	旋轉軸 v_1 和 v_2 之最小位置的軟體限制 $TC_CARR30[m] $TC_CARR31[m]		
最大位置旋轉軸 v_1 旋轉軸 v_2	旋轉軸 v_1 和 v_2 之最大位置的軟體限制 $TC_CARR32[m] $TC_CARR33[m]		
刀把名稱	刀把可以命名而不用編號。$TC_CARR34[m]		
使用者： 軸名稱 1 軸名稱 2 識別碼 **定位**	預計使用於使用者量測循環$TC_CARR35[m] $TC_CARR36[m] $TC_CARR37[m]		
	$TC_CARR38[m]	$TC_CARR39[m]	$TC_CARR40[m]

可定向刀把之系統變數的延伸			
微調偏移	可加入**基本參數中值**的參數。		
l_1 偏移向量	$TC_CARR41[m]	$TC_CARR42[m]	$TC_CARR43[m]
l_2 偏移向量	$TC_CARR44[m]	$TC_CARR45[m]	$TC_CARR46[m]
l_3 偏移向量	$TC_CARR55[m]	$TC_CARR56[m]	$TC_CARR57[m]
l_4 偏移向量	$TC_CARR58[m]	$TC_CARR59[m]	$TC_CARR60[m]
v_1 旋轉軸	$TC_CARR64[m]		
v_2 旋轉軸	$TC_CARR65[m]		

說明

參數的說明

“m”指定要程式設計的刀把編號。

$TC_CARR47 到$TC_CARR54 以及$TC_CARR61 到$TC_CARR63 並未定義，而且如果嘗試讀取或寫入存取時會產生警報。

軸上距離向量的開始 / 結尾可以任意選擇。繞著兩個軸的旋轉角度 α_1、α_2 是在刀把的初始狀態中以 0°定義。以這種方式，刀把的動力學可以為任何可能數量進行程式設計。

只有一個旋轉軸或沒有旋轉軸的刀把可以透過將一個或兩個旋轉軸的方向向量設定為零點來說明。
如果刀把沒有旋轉軸，距離向量可做為其他刀具補正，其元件不能受加工平面的變更影響（G17 到 G19）。

參數延伸

旋轉軸的參數

系統變數已由整個輸入$TC_CARR24[m]到$TC_CARR33[m]加以延伸，並說明如下：

旋轉軸 v_1、v_2 的偏移	為已定向刀把的初始設定變更旋轉軸 v_1 或 v_2 的位置。
旋轉軸 v_1、v_2 的**角度偏移 / 角度增量**	旋轉軸 v_1 和 v_2 之 Hirth 齒系統的偏移或角度增量。程式設計或計算的角度捨入為當 n 為整數時 phi = s + n * d 所產生的下一個值。
旋轉軸 v_1、v_2 的**最小和最大位置**	旋轉軸 v1 和 v2 之旋轉軸限制角度（軟體限制）的最小和最大位置。

使用者參數

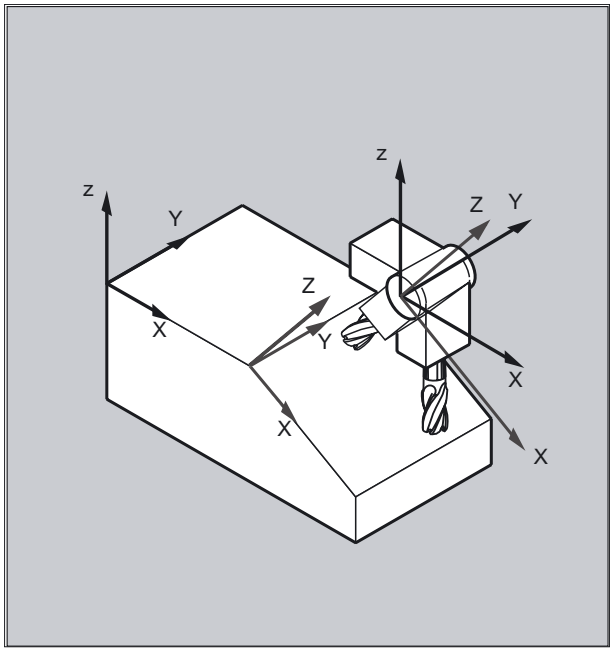
$TC_CARR34 到$TC_CARR40 包含使用者可自由使用的參數，一直到 SW 6.4 都與標準相同，在 NCK 沒有進一步評估，也無有效性。

微調偏移參數

$TC_CARR41 到$TC_CARR65 包含微調偏移參數，可加入至基本參數中的值。指派給基本參數的微調偏移值是在值 40 加入參數數字時取得。

範例

使用於下列範例中的刀把可以完全由繞著 Y 軸的旋轉加以說明。



程式碼	註解
`N10 $TC_CARR8[1]=1`	; 刀把 1 第一個旋轉軸之 Y 元件的定義。
`N20 $TC_DP1[1, 1] = 120`	; 軸銑刀的定義。
`N30 $TC_DP3[1, 1]=20`	; 軸銑刀的定義，20 毫米。
`N40 $TC_DP6[1, 1]=5`	; 軸銑刀的定義，5 毫米半徑。
`N50 ROT Y37`	; 繞著 Y 軸 37° 旋轉的框架定義。
`N60 X0 Y0 Z0 F10000`	; 逼近開始位置。
`N70 G42 CUT2DF TCOFR TCARR=1 T1 D1 X10`	; 設定半徑補正、旋轉框架中的刀長補正（TLC）、選擇刀把 1、刀具 1。
`N80 X40`	; 在 37° 旋轉下執行加工。
`N90 Y40`	
`N100 X0`	
`N110 Y0`	
`N120 M30`	

其它資訊

解析動力學

在具解析動力學的機台（刀具和工作都可以旋轉）上，系統變數已經由輸入$TC_CARR18[m]擴充到$TC_CARR23[m]，而說明如下：

可旋轉刀具表包含：

- 第二個旋轉軸 V_2 到可旋轉第三個旋轉軸之 I_4 的刀具表參考點之間的向量間隙。

旋轉軸包含：

- 旋轉軸 v_1 和 v_2 參考的兩個通道識別碼，需要時，即存取其位置，以決定可定向刀把的方向。

含 T、P 或 M 其中一個值的動力學類型：

- 動力學類型 T：只有刀具可以旋轉。
- 動力學類型 P 只有工件可以旋轉。
- 動力學類型 M：工件和刀具都可以旋轉。

清除刀把資料

所有刀把資料集的資料都可以使用`$TC_CARR1[0]=0` 刪除。

動力學類型`$TC_CARR23[T]=T` 必須指派給三個允許的大寫或小寫字母（T、P、M），因此不可刪除。

變更刀把資料

每個已說明值都可以透過在工件程式中指派新值來修改。嘗試啟動可以定向的刀把時，T、P 或 M 以外的任何字元都會產生警報。

讀取刀把資料

每個已說明值都可以透過指派給工件程式中的變數來讀取。

微調偏移

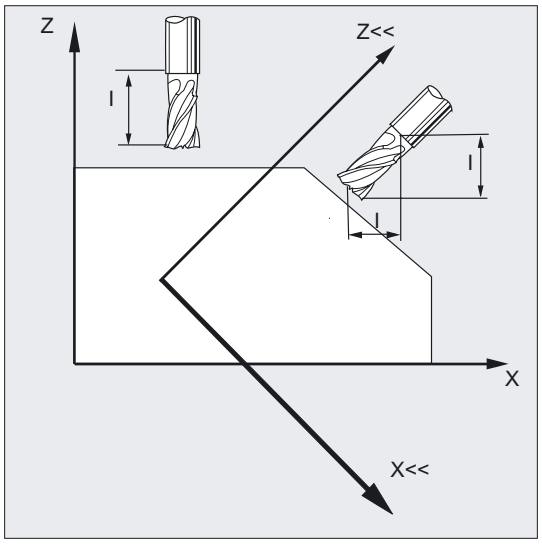
只有在啟動可以定向的刀把時，才會偵測所有不合規定的微調偏移值，此刀盤中包含不合規定的值，同時設定資料 SD42974 $SC_TOCARR_FINE_CORRECTION = TRUE。

最大允許的微調偏移受限於機械參數中允許的值。

7.9 用於可定向刀把的刀具長度補正（TCARR，TCOABS，TCOFR，TCOFRX，TCOFRY，TCOFRZ）

功能

當刀具的空間方向變更時，其刀長元件也會跟著變更。



在重置之後，例如手動設定或變更有固定空間方向的刀把，刀長元件也必須再度決定。這是使用 TCOABS 和 TCOFR 路徑指令執行。

對可以定向主動框架的刀把來說，使用 TCOFRZ、TCOFRY 和 TCOFRX 選擇刀具時，可以定義刀具應該指的方向。

句法

```
TCARR=[<m>]
TCOABS
TCOFR
TCOFRZ
TCOFRY
TCOFRX
```

意義

TCARR=[<m>]:	以數量 “m”要求刀把
TCOABS:	從目前刀把的方向決定刀長元件
TCOFR:	從主動框架的方向決定刀長元件
TCOFRZ:	主動框架的可定向刀把，含指著 Z 方向的刀具
TCOFRY:	主動框架的可定向刀把，含指著 Y 方向的刀具
TCOFRX:	主動框架的可定向刀把，含指著 X 方向的刀具

其它資訊

從刀把的方向決定刀長補正

TCOABS 從刀把的目前方向角度計算刀長補正；儲存於系統變數$TC_CARR13 and $TC_CARR14 中。

關於含系統變數的刀把動力學定義，請參閱“刀盤動力學 (頁 384)”。

框架變更時，若要重新計算刀長補正，必須再次選擇刀具。

主動框架的刀具方向

設定了含定向功能的刀把，讓刀具指向下列方向。

- 使用 TCOFR 或 TCOFRZ，指向 Z 方向
- 使用 TCOFRY，指向 Y 方向
- 使用 TCOFRX，指向 X 方向

刀長補正是在 TCOFR 與 TCOABS 之間變換時重新計算。

要求刀把（TCARR）

使用 TCARR 時，刀把數量 m 是以其幾何資料要求（補正記憶體）。

使用 m=0 時，則取消選擇已啟用刀把。

只有在呼叫刀具之後，刀把的幾何資料才會有效。選擇的刀具會在刀把變更發生後維持啟用。

刀把目前的幾何資料也可以在工件程式中，經由對應系統變數定義。

重新計算框架變更的刀長補正（TCOABS）

框架變更時，若要重新計算刀長補正，必須再次選擇刀具。

說明

刀具方向必須手動手動倍率以適應主動框架。

計算刀長補正（TLC）時，刀把的旋轉角度是在一個中間步進計算。使用有兩個旋轉軸的刀把時，一般是有兩組旋轉軸，可用來手動倍率刀具方向以適應主動框架，因此儲存在系統變數中的旋轉角度值必須至少大約對應於以機械方式設定的旋轉角度。

說明

刀具方向

控制系統不可能檢查旋轉角度是否經由機台可設定的框架方向來計算。

如果刀把的旋轉軸排列方式，使得刀具方向是經由無法到達的框架方向來計算，就會輸出警報。

不允許在可移動刀把上結合使用刀具精準補正和刀長補正的函數。如果同時呼叫這兩個函數，就會發出錯誤訊息。

TOFRAME 函數允許根據所選擇刀把的定位方向來定義框架。如需其他詳細資訊，請參閱“框架”一章。

啟用方向轉換（三、四或五軸轉換）時，可以選擇方向是從零點位置衍生而來的刀把，而不會造成警報輸出。

標準和量測循環的傳輸參數

在標準和量測循環的傳輸參數上，適用下列已定義的值域。

角度值的已定義值域如下：

- 繞著第一個幾何軸旋轉：-180 度到+180 度
- 繞著第二個幾何軸旋轉：-90 度到+90 度
- 繞著第三個幾何軸旋轉：-180 度到+180 度

請參閱“框架”一章的“可程式設計旋轉（ROT，AROT，RPL）”。

說明

當傳輸角度值至標準或量測循環時，應該小心觀察下列規則：

值若小於 NC 之計算的解析度就應該捨入至零點！

角度位置之 NC 的計算解析度是在機械參數中定義：

MD10210 $MN_INT_INCR_PER_DEG

7.10 線上刀長補正（TLC）（TOFFON，TOFFOF）

功能

使用系統變數$AA_TOFF[ ]以覆疊有效刀長，而與實時三度空間中的三個刀具方向一致。
使用三個幾何軸識別碼做為索引。這同時也會定義已啟用幾何軸的有效補正方向數。
可同時啟動所有補正。
線上刀長補正函數可以用於：

- 方向轉換 TRAORI
- 可定向刀把 TCARR

機械製造商

線上刀長補正是**選項**，必須事先啟用。只有結合有效方向轉換或已啟用可定向刀把時，此函數才有實效可用。

句法

```
N..TRAORI
N..TOFFON（X，25）
N..WHEN TRUE DO $AA_TOFF[tool direction] 在同步化動作中
```

如需在動作同步動作中程式設計線上刀長補正的相關詳細資訊，請參閱“同步化動作中的動作”。

意義

TOFFON	Tool Offset ON（啟動線上刀長補正） 啟動時，可以為補正的相關方向指定偏移值，這項指定會立即恢復。
TOFFOF	Tool Offset ON（重置線上刀長補正） 重置相關的補正值，並啟動前置處理停止。
X，Y，Z	為 TOFFON 指出偏移值的補正方向

刀長補正選擇之範例。

```
程式碼                                    註解
MD21190 $MC_TOFF_MODE = 1                 ; 逼近絕對值
MD21194 $MC_TOFF_VELO[0] =1000
MD21196 $MC_TOFF_VELO[1] =1000
MD21194 $MC_TOFF_VELO[2] =1000
MD21196 $MC_TOFF_ACCEL[0] =1
MD21196 $MC_TOFF_ACCEL[1] =1
MD21196 $MC_TOFF_ACCEL[2] =1
N5 DEF REAL XOFFSET
N10 TRAORI（1）                           ; 轉換啟動
N20 TOFFON（Z）                           ; 在 Z 刀具方向
                                            啟用刀長補正（TLC）
N30 WHEN TRUE DO $AA_TOFF[Z] = 10         ; 為 Z 刀具方向
G4 F5                                       插補 TLC 10
...
N100 XOFFSET = $AA_TOFF_VAL[X]            ; 在 X 方向為
N120 TOFFON（X，-XOFFSET）                  指派實際補正，
G4 F5                                       刀長補正（TLC）會再度縮減為 0
```

取消刀長補正選擇之範例。

```
程式碼                                    註解
N10 TRAORI（1）                           ; 轉換啟動
N20 TOFFON（X）                           ; 啟動 Z 刀具方向
N30 WHEN TRUE DO $AA_TOFF[X] = 10         ; 為 X 刀具方向
G4 F5                                       插補 TLC 10
...
N80 TOFFOF（X）                           ; 刪除了 X 刀具方向的
                                            位置偏移：...$AA_TOFF[X] = 0
                                            沒有移動任何軸，
                                            位置偏移加入
                                            工件座標系統（WCS）的實際位置，
                                            與目前的方向一致
```

說明

單節準備

在前置處理中的單節準備期間，也會將目前在主要執行中啟用的刀長偏移考慮在內。為了允許延伸使用最大容許軸速率，必須在設定刀具偏移時，使用 STOPRE 前置處理停止，停止單節準備。

在程式啟動後未變更刀長偏移時，或是如果變更刀具偏移後，所處理過的單節超過 IPO 緩衝區在旋進與主要旋進之間所能容納的單節，刀具偏移在旋進時永遠都是已知。

變數$AA_TOFF_PREP_DIFF

目前在插補器中啟用的補正與在單節準備時啟用的補正之間差異的尺寸可以在變數 $AA_TOFF_PREP_DIFF[ ]修齊。

手動倍率機械參數和設定資料

下列機械參數可供線上刀長偏移使用：

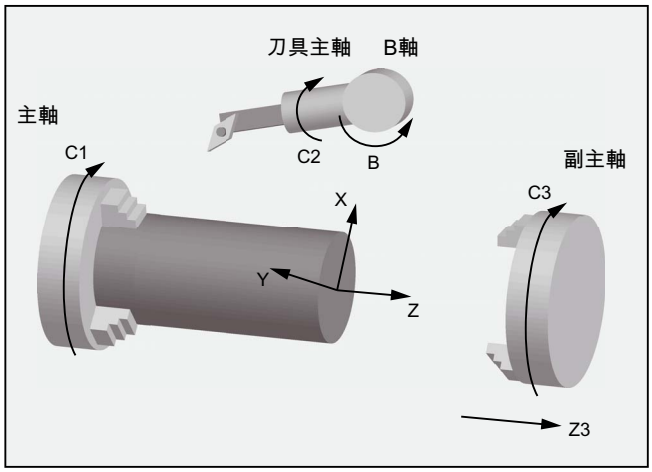
- MD 20610：ADD_MOVE_ACCEL_RESERVE 覆疊動作的加速度極限
- MD 21190：TOFF_MODE：系統變數$AA_TOFF[ ]的內容已恢復或整合為絕對值
- MD 21194：TOFF_VELO 線上刀長偏移的速率。
- MD 21196：TOFF_VELO 線上刀長偏移的加速度。
- 預設限制值
  SD 42970 的設定資料：TOFF_LIMIT 刀長偏移值的上限。

參考：/FB3/ 功能手冊，特殊函數；三至五軸轉換（F2）。

7.11 可旋轉刀具的切削資料修改（CUTMOD）

功能

使用函數“可旋轉刀具的切削資料修改”，在旋轉刀具（絕大多數是車刀，但也包括鑽孔和銑削刀具）時，相對於正在加工的工件所取得的已變更幾何關係可以與刀具補正一起考慮。



圖像 7-1 車床的可旋轉刀具

實際的刀具旋轉永遠都是從目前已啟用的可定向刀把來決定（請參閱“可定向刀盤的刀長補正（TLC） (頁 389)”）。

此函數是使用_movePathCircular（）指令啟動。

句法

```
CUTMOD=<值>
```

意義

CUTMOD	轉入使用“可旋轉刀具的切削資料修改”函數的指令	
<值>	下列的值可以指派給 CUTMOD 指令：	
	0	函數已停用。 從系統變數$P_AD...供應的值與對應刀具參數相同。
	> 0	如果已啟用含指定數量的可定向刀把，就啟動函數，也就是，函數啟用是與特定可定向刀盤相連結。 從系統變數$P_AD...供應的值可以配合對應刀具參數修改，取決於已啟用的旋轉。 停用所指定可定向刀把會暫時停用此函數；啟用其他可定向刀盤則永遠停用此函數。因此在第一種情況下，函數會在再次選擇相同可定向刀把時重新啟動；而在第二種情況下，即使在後續時都必須要有新的選擇，可定向刀盤會以指定的數量重新啟動。 此函數不會受重置影響。
	-1	如果已啟用可定向刀把，永遠都會啟動此函數。 變更刀把時，或取消選擇刀盤而後續新增選擇時，不必再次設定 CUTMOD。
	-2	如果所啟用的刀把其數量與目前啟用的可定向刀盤相同，則永遠都會啟動此函數。 如果未啟用可定向刀把，則其有效性與 CUTMOD=0 相同。如果已啟用可定向刀把，則此函數的有效性與直接指定實際刀盤數量時相同。
	< -2	小於 2 的值將不予理會，也就是，這種情況視同未程式設計 CUTMOD 處理。 **注意：** 不應該使用此值域，因為已保留作為可能的後續擴充。

說明

SD42984 $SC_CUTDIRMOD

此函數可使用 CUTMOD 指令加以啟動，取代可使用設定資料 SD42984 $SC_CUTDIRMOD 啟動的函數，但是此函數的可用性維持不變。但因為平行使用兩個函數沒有意義，所以只有在 'CUTMOD 等於零點時，才能加以啟動。

範例

以下範例是指刀鼻位置為 3 並有可定向刀把的刀具，可以繞著 B 軸旋轉刀具。

註解中的數值是依 X、Y、Z 順序，定機械座標系統（MCS）的單節位置結尾。

```
N10 $TC_DP1[1，1]=500
N20 $TC_DP2[1，1]=3                                          ; 刀鼻位置
N30 $TC_DP3[1，1]=12
N40 $TC_DP4[1，1]=1
N50 $TC_DP6[1，1]=6
N60 $TC_DP10[1，1]=110                                       ; 刀盤角度
N70 $TC_DP11[1，1]=3                                         ; 切削方向
N80 $TC_DP24[1，1]=25                                        ; 間隙角度

N90 $TC_CARR7[2]=0 $TC_CARR8[2]=1 $TC_CARR9[2]=0             ; B axis
N100 $TC_CARR10[2]=0 $TC_CARR11[2]=0 $TC_CARR12[2]=1         ; C 軸
N110 $TC_CARR13[2]=0
N120 $TC_CARR14[2]=0
N130 $TC_CARR21[2]=X
N140 $TC_CARR22[2]=X
N150 $TC_CARR23[2]="M"

N160 TCOABS CUTMOD=0
N170 G18 T1 D1 TCARR=2                                         X          Y          Z
N180 X0 Y0 Z0 F10000                                         ; 12.000     0.000      1.000

N190 $TC_CARR13[2]=30
N200 TCARR=2
N210 X0 Y0 Z0                                                ; 10.892     0.000      -5.134
N220 G42 Z—10                                                ; 8.696      0.000      —17.330
N230 Z—20                                                    ; 8.696      0.000      —21.330
N240 X10                                                     ; 12.696     0.000      —21.330
N250 G40 X20 Z0                                              ; 30.892     0.000      —5.134

N260 CUTMOD=2 X0 Y0 Z0                                       ; 8.696      0.000      —7.330
N270 G42 Z—10                                                ; 8.696      0.000      —17.330
N280 Z—20                                                    ; 8.696      0.000      —21.330
N290 X10                                                     ; 12.696     0.000      —21.330
N300 G40 X20 Z0                                              ; 28.696     0.000      —7.330

N310 M30
```

說明：

在單節 N180 中，剛開始是為 CUTMOD=0 和非旋轉的可定向刀把選擇刀具。由於所有可定向刀把的偏移向量都是 0，因此是逼近對應於$TC_DP3[1，1]和$TC_DP4[1，1]中所指定長度的位置。

可以繞著 B 軸旋轉 30°定向的刀把是在單節 N200 中啟動。由於因 CUTMOD=0 而未修改刀鼻位置，舊的刀鼻參考點仍固定如前不改動。因此會在單節 N210 中逼近此位置，而讓舊刀鼻參考點保持在零點（也就是，向量（1，12）是在 Z/X 平面中穿過 30°旋轉）。

在單節 N260 中，與單節 N200 相反，是 CUTMOD=2 生效。由於可定向刀把旋轉的結果，修改的刀鼻位置變成 8，因此也是不同的軸位置。

刀具半徑補正（TRC）是在單節 N220 和 / 或 N270 中啟動。在這兩個程式區段中不同的刀鼻位置不會影響已啟用 TRC 的單節結尾位置，因上對應的位置完全相同。不同的刀鼻位置只會在取消選擇單節 N260 和 / 或 N300 時再度生效。

其它資訊

已修改切削資料的有效性

修改的刀鼻位置和修改的刀鼻參考點是在程式設計時立即生效，即使對已經啟用的刀具也一樣有效。不必特別再重新選擇刀具。

有效加工平面的影響

為了決定修改的刀鼻位置、切削方向與刀盤或間隙角度，有效平面中刀刃的評估（G17 - G19）是決定性的。

但是，如果設定資料 SD42940 $SC_TOOL_LENGTH_CONST（選擇平面時的刀長 元件變更），有效值不等於零點（正或負 17、18 或 19），則其內容會定義平面，在其中評估相關的數量。

系統變數

下列系統變數可供使用：

系統變數	意義
$P_CUTMOD_ANG / $AC_CUTMOD_ANG	為使用 CUTMOD 和 / 或$SC_CUTDIRMOD 啟動的函數，在有效加工平面提供（非圓）角度，用來做為切削資料（刀鼻位置、切削方向、間隙角度和刀盤角）修改的基準。 $P_CUTMOD_ANG 是指前置處理中的實際狀態，$AC_CUTMOD_ANG 是指實際主要執行單節。
$P_CUTMOD / $AC_CUTMOD	讀取上次使用 CUTMOD 指令程式設計的目前有效值（應該為切削資料修改啟動的刀把數量）。 如果上次程式設計的 CUTMOD 值= -2（以目前啟用的可定向刀把啟動），則$P_CUTMOD 中不會傳回值-2，而傳回程式設計時的已啟用可定向刀盤數量。 $P_CUTMOD 是指前置處理中的實際狀態，$AC_CUTMOD 是指實際主要執行單節。
$P_CUT_INV / $AC_CUT_INV	旋轉刀具時提供值 TRUE，讓旋轉的主軸方向必須逆反。若要執行這項作業，必須在讀取操作所指的單節中滿足下列四個條件： 1. 如果已啟用車削或研磨刀具（刀具型號 400 到 599 和 / 或 SD42950 $SC_TOOL_LENGTH_TYPE = 2）。 2. 已使用語言指令 CUTMOD 啟動切削影響。 3. 已啟用使用 CUTMOD 之數值指定的可定向刀把。將 Check－I 重新公式化。 4. 可定向刀把在加工平面中繞著軸（一般是 C 軸）旋轉刀具，讓所產生刀具刀刃的垂直面是與初始位置形成大於 90°（一般是 180°）的位置旋轉。 如果四個指定的條件至少有一個未滿足，變數的內容就是 FALSE。如果是未定義刀鼻位置的刀具，則變數的值永遠是 FALSE。 $P_CUT_INV 是指前置處理中的實際狀態，$AC_CUT_INV 是指實際主要執行單節。

所有主要執行變數（$AC_CUTMOD_ANG、$AC_CUTMOD 和$AC_CUT_INV）都可以讀入同步化動作中。前置處理的讀取存取操作會產生前置處理停止。

修改的切削資料：

如果如果已啟用刀具旋轉，修改的資料就可以供下列系統變數使用：

系統變數	意義
$P_AD[2]	刀鼻位置
$P_AD[10]	刀盤角度
$P_AD[11]	切削方向
$P_AD[24]	間隙角度

說明

如果已使用 CUTMOD 指令啟動“可旋轉刀具的切削資料修改”函數，而且已啟動造成旋轉的可定向刀把，則資料永遠都會配合對應的刀具參數（$TC_DP2[...，...]等）進行修改。

參考

如需有關“可旋轉刀具的切削資料修改”函數的其他資訊，請參閱：

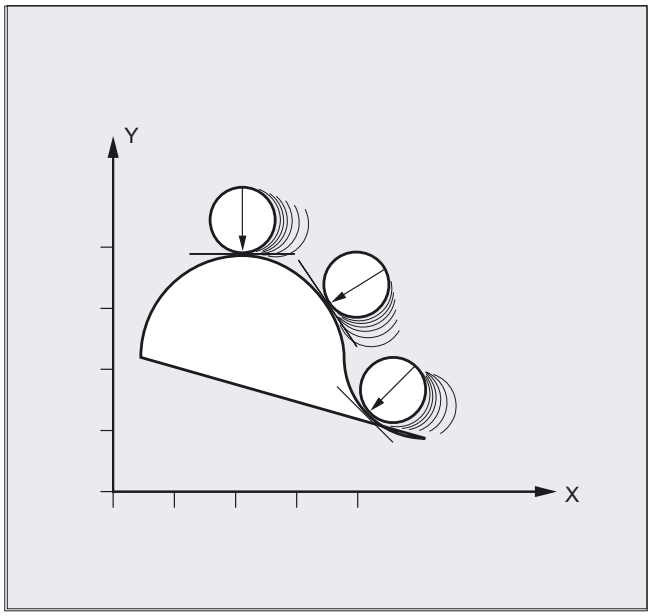
功能手冊基本功能；刀具偏移（W1）

8 路徑移動行為

8.1 切線控制（TANG，TANGON，TANGOF，TLIFT，TANGDEL）

功能

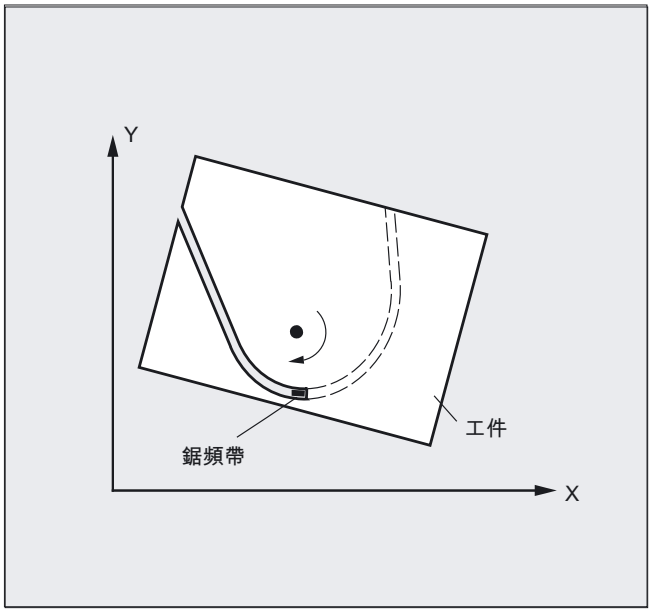
跟隨軸會教導先導軸沿著切線的路徑。這允許刀具與輪廓平行對齊。使用在 TANGON 指令中程式設計的角度，刀具可以相對於切線定位。



應用

切線控制可用於下列應用程式中：

- 切片時可旋轉刀具的切線定位
- 帶鋸的後續工件對齊（參見圖示）。
- 研磨輪上修整刀具的定位
- 玻璃或紙張加工的切削輪定位
- 切線進給 5 軸焊絲。



句法

```
TANG （Faxis，Laxis1，Laxis2，Coupling，CS，Opt）
TANGON （Faxis，Angle，Dist，Angletol）
TANGOF （Faxis）
TLIFT （Faxis）
TANGDEL （FAxis）
```

簡化的程式設計：

耦合係數 1 不必明確地進行程式設計。

TANG（C，X，Y，1，"B"，"P"）可以簡寫為 TANG（C，X，Y，，，"P"）。跟以前一樣，TANG（C，X，Y，1，"B"，"S"）可以寫成 TANG（C，X，Y）。

TLIFT（...）敘述必須在使用 TANG（...）指派軸之後，立即進行程式設計。範例：

```
TANG（C，X，Y...）
TLIFT （C）
```

停用 TLIFT

重複軸指派 TANG（...）後面不需要再加上 TLIFT（...）。

TANGDEL 刪除後續切線的定義

如果在準備呼叫 TANG 中定義了以相同跟隨軸後續的新增切線，則必須刪除現有的使用者自訂後續切線。只有在停用與 TANGOF（Faxis）耦合時，才可以進行刪除作業。

意義

TANG	切線後續定義的準備敘述；預設值：1 **TANG（C，X，Y，1，"B"）代表：** 旋轉軸 C 跟隨幾何軸 X，並轉出 Y。TLIFT
TANGON	啟動指定跟隨軸的切線控制及必要的跟隨軸偏移角度，必要時還有磨圓路徑和角度偏差。 **TANGON（C，90）代表：** C 軸是跟隨軸。在路徑軸的每一個動作上，轉作與路徑切線成 90°的位置。
TANGOF	停用指定跟隨軸的切線控制。 指定跟隨軸，以便停用切線控制： **TANGOF（C）**
TLIFT	在輪廓角落插入中間單節
TANGDEL	刪除後續切線的定義 **範例：TANGDEL （FAxis）**
Faxis	跟隨軸：跟隨旋轉軸的其他切線。
Laxis1，Laxis2	先導軸：路徑軸，可決定跟隨軸的切線。
耦合	耦合係數：切線的角度變更與跟隨軸之間的關係。 參數可選配；預設值：1
C	識別座標系統的字母 “B”=基準座標系統；輸入是可選配的；預設設定 “W”=工件座標系統無法使用
Opt	最佳化： “S”標準，預設值 “P”經過切線軸和輪廓的時間後自動手動倍率
角度	隨動軸的偏移角度
Dist	隨動軸的平滑化化路徑（Opt "P"為必要條件）
Angletol	隨動軸的角度允差值（選擇性），唯有當 Opt = "P"時才會進行驗算

Opt、Dist 和 Angletol 最佳化可能性

Opt="P"為先導軸的速限指定跟隨軸的動態行為，尤其是在使用動態轉換時，特別建議使用。

參數（Dist 和 Angletol）會精確的限制隨動軸與主導軸切線之間的偏差。

範例：平面變更

程式碼	註解
N10 TANG（A，X，Y，1）	; 1. 定義切線追蹤。
N20 TANGON（A）	; 啟動耦合。
N30 X10 Y20	; 半徑
...	
N80 TANGOF（A）	; 停用第一個耦合。
N90 TANGDEL（A）	; 刪除第一個定義。
...	
N120 TANG（A，X，Z）	; 2. 定義切線追蹤。
TANGON（A）	; 啟動新增耦合
...	
N200 M30	

幾何軸切換和 TANGDEL 的範例

且不會發出警報。

程式碼	註解
N10 GEOAX（2，Y1）	; Y1 是幾何軸 2。
N20 TANG（A，X，Y）	
N30 TANGON（A，90）	
N40 G2 F8000 X0 Y0 I0 J50	
N50 TANGOF（A）	; 停用以 Y1 進行追蹤。
N60 TANGDEL（A）	; 刪除第一個定義。
N70 GEOAX（2，Y2）	; Y2 是新增幾何軸 2
N80 TANG（A，X，Y）	; 2. 定義切線追蹤。
N90 TANGON（A，90）	; 啟動以第二個定義追蹤。
...	

範例：以自動最佳化進行切線追蹤

使用 Dist 和角度允差進行自動最佳化。

程式碼	註解
N80 G0 C0	; Y1 是幾何軸 2。
N100 F=50000	
N110 G1 X1000 Y500	
N120 TRAORI	; 含軸允差的磨圓。
N130 G642	
N171 TRANS X—Y—	; 路徑速率的自動最佳化。
N180 TANG（C，X，Y，1，，"P"）	; 磨圓（調合）距離 5 毫米
N190 TANGON（C，0，5.0，2.0）	; 角度允差，2 度
N210 G1 X1310 Y500	; 啟動以第二個定義追蹤。
N215 G1 X1420 Y500	
N220 G3 X1500 Y580 I=AC（1420） J=AC（580）	
N230 G1 X1500 Y760	
N240 G3 X1360 Y900 I=AC（1360） J=AC（760）	
N250 G1 X1000 Y900	
N280 TANGOF（C）	
N290 TRAFOOF	
N300 M02	

定義跟隨軸和先導軸

使用 TANG 定義跟隨軸和先導軸。

耦合係數指定切線角度變動與隨動軸角度變動兩者之間的關係。其值一般是 1（預設值）。

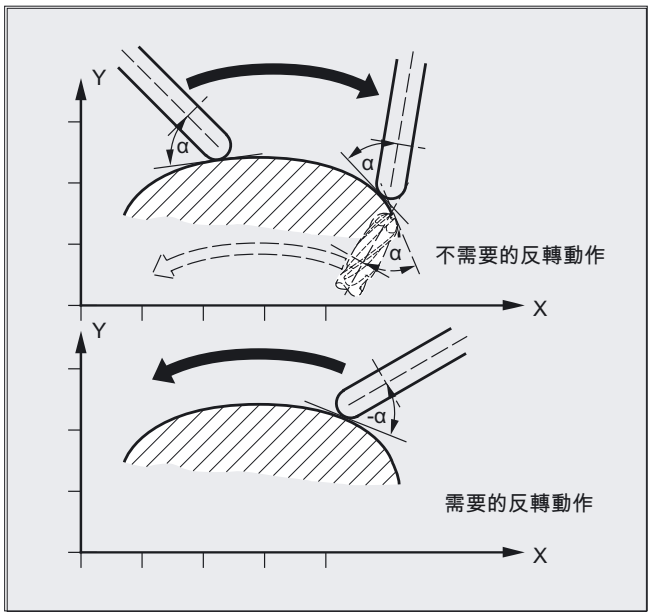
使用工作區限制的限制角度

對於來回振盪的路徑動作，切線在路徑上的車削點跳躍穿過 180°，而跟隨軸的方向也依此變更。

這種行為一般來說是不適當的：回轉動作應該以與逼近動作相同的負偏移角度移動。

若要執行這項作業，請限制跟隨軸的工作區（G25、G26）。工作區限制必須在路徑逆轉的情況下啟用（WALIMON）。

如果偏移角度位於工作區限制之外，就會嘗試以負偏移角度回到允許的工作區。



在輪廓角落插入中間單節，TLIFT

輪廓某一角落的切線發生變化，所以隨動軸的設定點位置亦隨之改變。軸一般會嘗試以其最大可能速率補償步進變更。但是，這會造成在該角落之後，輪廓上某段距離間所要切線位置的偏差。如果在技術上無法接受這種偏差，可以使用 TLIFT 指令，強迫控制在角落停止，然後在自動產生的中間單節中，將跟隨軸轉成新的切線方向。

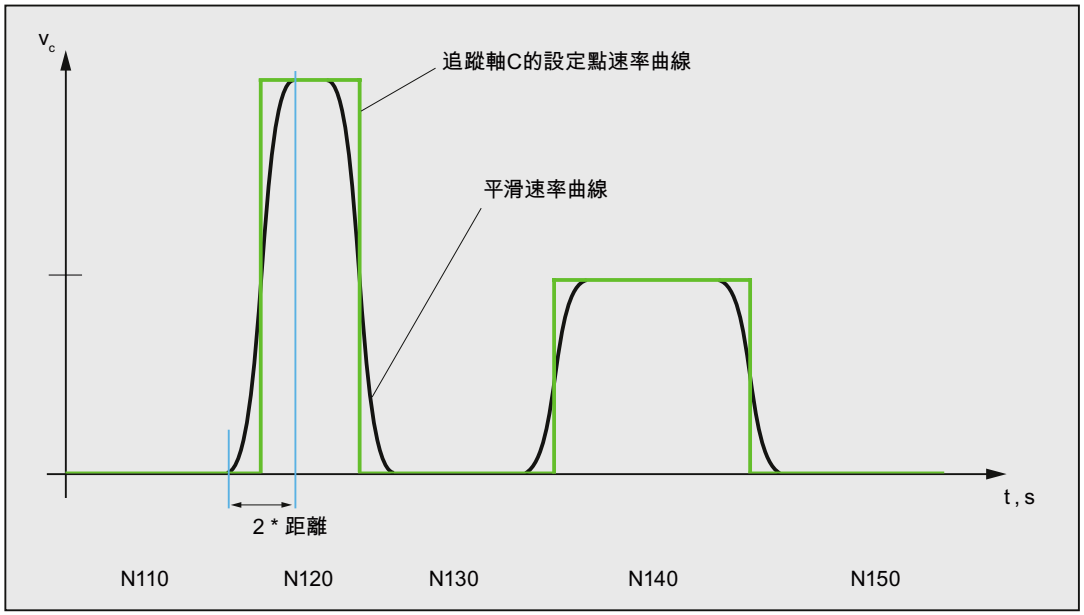
如果已使用跟隨軸做為路徑軸一次，則使用路徑軸進行車削。跟隨軸的最大軸速率可以使用 TFGREF[ax] = 0.001 函數達到。

如果後續軸不是先前做為路徑軸移動，現在是做為定位軸移動。此時速率就取決於機械參數中的定位速率。

軸是以其最大可能速率旋轉。

最佳化可能性

因先導軸輪廓中的跳躍而造成的跟隨軸速率跳躍是使用（Dist 和 Angletol）磨圓及平滑化。
跟隨軸是使用預見控制（參見圖示）加以控制，讓偏差盡可能減到最小。



定義角度變更

可自動插入中間單節的角度變更限制是經由機械參數$MA_EPS_TLIFT_TANG_STEP 定義。

轉換受到的影響

套用跟進控制的旋轉軸位置，可作為轉換時的輸入值。

跟隨軸的明確定位

如果跟隨先導軸的軸經過明確定位，該位置就會加入已程式設計的偏移角度。
所有路徑定義都可能：路徑和定位軸動作。

耦合狀態

您可以使用下列系統參數，在 NC 程式中查詢耦合的狀態：
$AA_COUP_ACT[axis]
0: 沒有啟用耦合
1,2,3: 已啟用切線後續

8.2 進給率回應（FNORM，FLIN，FCUB，FPO）

功能

為了允許進給特性的彈性定義，根據
DIN 66205 程式設計的進給已由線性和三次特性加以延伸。

三次特性可以直接或做為插補曲線進行程式設計。由於這些附加的特性，使連續的平滑化速度特性得以依照待加工工件的曲率進行程式設計。

由於這些附加的特性，使連續的平滑化速度特性得以依照待加工工件的曲率進行程式設計。

句法

```
F… FNORM
F… FLIN
F… FCUB
F=FPO（…, …, …）
```

意義

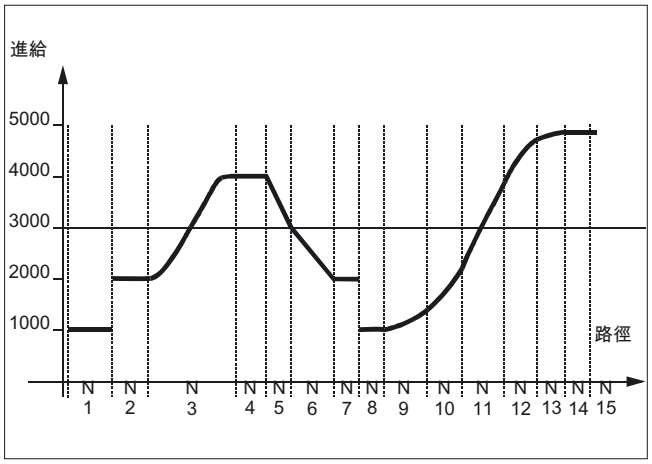
FNORM	基本設定。進給值會被指定為該單節中移動路徑的函數，隨後成為一個有效的模態值。
FLIN	路徑速率輪廓**線性：** 進給值是經由在單節開始和單節結束目前值的移動路徑，以線性方式逼近，然後成為有效的模態值。回應可以結合 G93 和 G94。
FCUB	路徑速率輪廓**三次：** 整個單節已程式設計的 F 值（相對於單節結尾）是以曲線連接。曲線是與前後定義的進給率形成切線開始及結束，並使用 G93 和 G94 生效。 如果單節中少了 F 位址，就使用要已程式設計的最後 F 值。
F=FPO…	**多項式**路徑速率輪廓： F 位址經由多項式定義從目前值到單節結尾的進給特性。此後結束值即為有效模態值。

曲線路徑區段上的進給最佳化

進給多項式 F=FPO 和進給曲線 FCUB 應該永遠都以恆定切削速率 CFC 移動，從而允許產生無震動設定點量變曲線。這樣才能夠建立持續加速度設定點進給輪廓。

範例：各種不同的進給輪廓

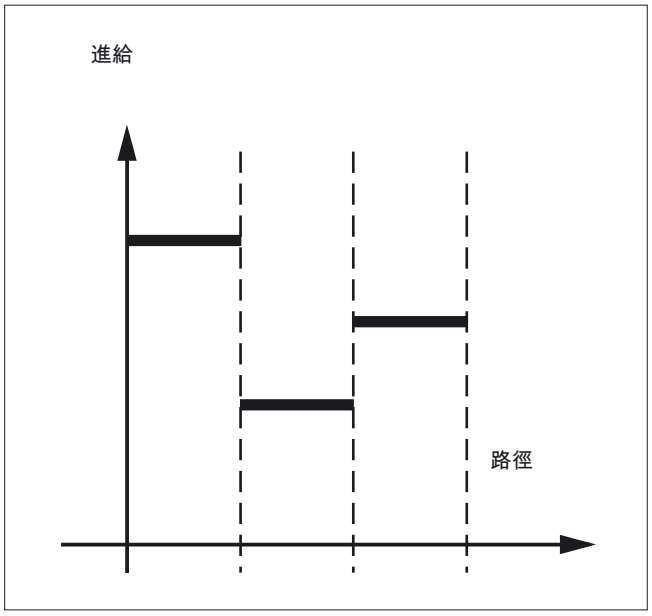
這個範例為您示範，各種不同範例輪廓的程式設計和圖形表示。



程式碼	註解
N1 F1000 FNORM G1 X8 G91 G64	; 恆定進給率輪廓，增量尺寸資料
N2 F2000 X7	; 設定點速率步進變更
N3 F=FPO（4000，6000，-4000）	; 在單節結尾使用進給 4000 經由多項式進給輪廓
N4 X6	; 多項式進給率 4000 為有效模態值
N5 F3000 FLIN X5	; 線性進給率輪廓
N6 F2000 X8	; 線性進給率輪廓
N7 X5	線性進給率有效模態值
N8 F1000 FNORM X5	; 含加速度步進變更的恆定進給率
N9 F1400 FCUB X8	; 在單節中已程式設計的下列所有 F 值是與曲線連接
N10 F2200 X6	
N11 F3900 X7	
N12 F4600 X7	
N13 F4900 X5	; 轉出曲線外形
N14 FNORM X5	
N15 X20	

FNORM

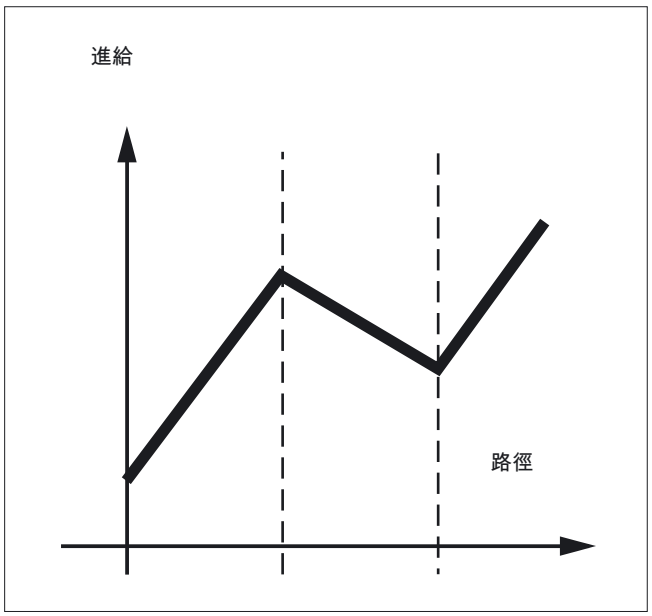
進給位址 F 根據 DIN 66025，將路徑進給定義為常數值。
請參閱程式設計手冊“基本概念”，取得有關此主題的詳細資訊。



FLIN

進給特性是以線性方式，從目前進給值逼近已程式設計的 F 值，直到單節結尾。
範例：

```
N30 F1400 FLIN X50
```

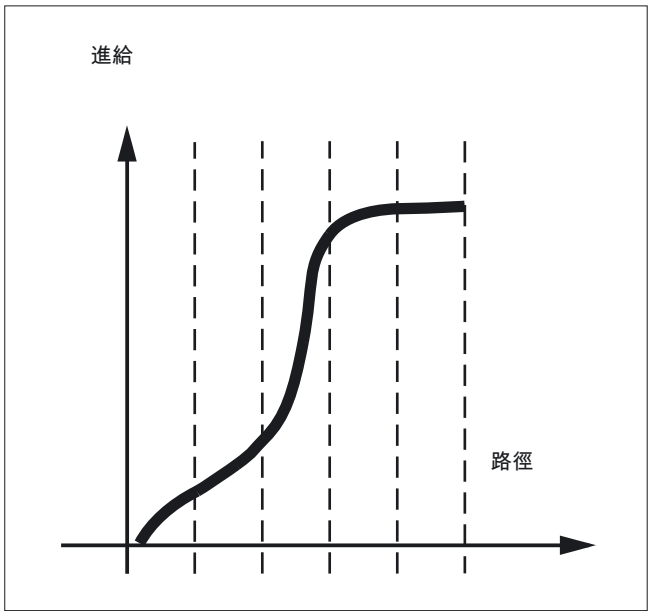


FCUB

進給是根據三次特性，從目前進給值逼近已程式設計的 F 值，直到單節結尾。控制使用曲線，連接具有效 FCUB 而以非模態方式已程式設計的所有進給值。進給值在這裡是做為插補點，以進行曲線插補的計算。

範例：

```
N50 F1400 FCUB X50
N60 F2000 X47
N70 F3800 X52
```



F=FPO（...，...，...）

進給特性是經由多項式直接進行程式設計。多項式係數是根據用於多項式插補的相同方法指定。

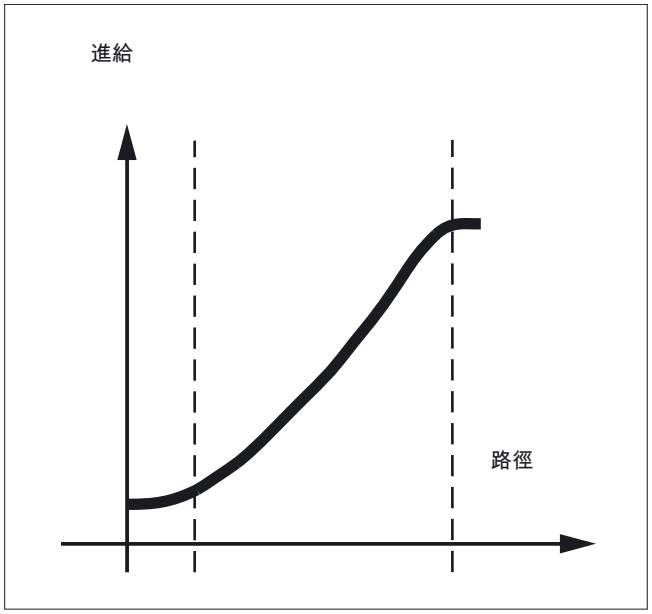
範例：

```
F=FPO（endfeed，quadf，cubf）
```

endfeed、quadf 和 cubf 是事先定義的變數。

endfeed：	在單節結尾進給
quadf：	二次多項式係數
cubf：	三次多項式係數

使用有效 FCUB，曲線是沿切線連結至在單節開始和單節結束時經由 FPO 定義的特性。



限制

不管程式設計的進給特性是什麼，已程式設計路徑移動特性的函數都適用。

不管是 G90 或 G91，已程式設計的進給特性永遠都是絕對的。

進給回應 FLIN 和 FCUB 是配合

G93 和 G94 啟用。

FLIN 和 FCUB **不會配合**

G95、G96/G961 和 G97/G971 啟用。

已啟用的壓縮 COMPON

使用已啟用的壓縮 COMPON，會在多個單節結合形成曲線區段時，套用下列各項：

FNORM：

群組中最後單節的 F 字組套用於曲線區段。

FLIN：

群組中最後單節的 F 字組套用於曲線區段。
已程式設計的 F 值一直套用到區段結尾，然後再以線性方式逼近。

FCUB：

產生的進給曲線是依不超過機械參數$MC_COMPESS_VELO_TOL 中所設定值的量，從已程式設計的終點衍生。

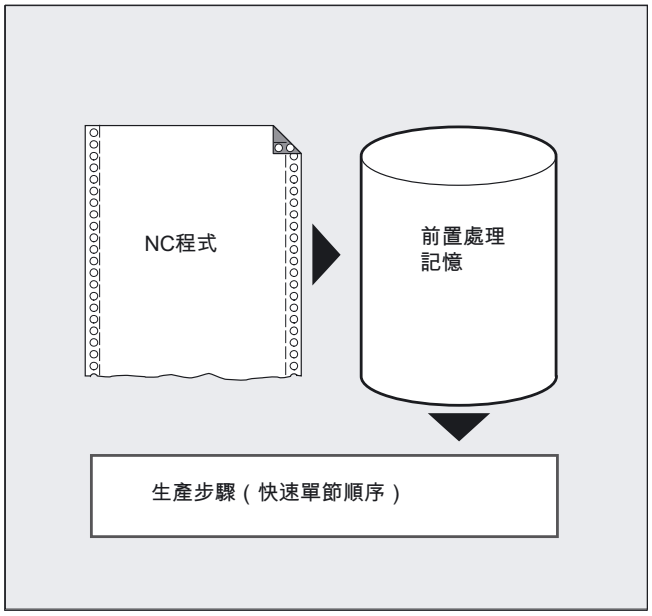
F=FPO（...，...，...）

這些單節不會進行壓縮。

8.3 具有前置處理記憶體（STOPFIFO，STARTFIFO，FIFOCTRL，STOPRE）的程式順序

功能

依其擴充程度而定，控制系統具有一定數量的所謂前置處理記憶體，程式執行之前已準備的單節即儲存在其中，然後在加工進行中，以高速單節順序輸出。這些順序可以讓短路徑以高速率移動。如果有足夠的剩餘控制時間可供使用，前置處理記憶體永遠都會填滿。



指定加工步進

要在前置處理記憶體中緩衝的加工步驟之開頭與結尾，會各自以 STOPFIFO 與 STARTFIFO，在工件程式中定義。前置處理的程序，以及緩衝的單節，只會在 STARTFIFO 指令之後開始，或是如果是前置處理記憶體滿載。

自動前置處理記憶體控制

自動前置處理記憶體控制係以 FIFOCTRL 指令來呼叫。FIFOCTRL 初始工作就像 STOPFIFO。不論怎麼程式設計，都不會開始處理，直到前置處理記憶體滿載為止。然而，對空的前置處理記憶體的回應會有所不同：以 FIFOCTRL，路徑速率會逐漸降低，直到填滿層級達到 2/3，才能防止全空以及減速到靜止狀態。

前置處理停止

在單節中程式設計 STOPRE 指令，會停止單節前置處理與緩衝。一直到所有前置處理以及儲存的單節，都已經被執行了，才會執行下列單節。前面的單節會以精確停止暫停（如同使用 G9）。

句法

表格 8- 1　定義加工步驟：

```
STOPFIFO
...
STARTFIFO
```

表格 8- 2　自動前置處理記憶體控制

```
...
FIFOCTRL
...
```

表格 8- 3　前置處理停止

```
...
STOPRE
...
```

說明

STOPFIFO，STARTFIFO，FIFOCTRL，與 STOPRE 指令必須在獨立單節中被程式設計。

意義

STOPFIFO:	STOPFIFO 對要在前置處理記憶體中，被緩衝的加工步驟之起點，進行定義。STOPFIFO 停止處理並填滿前置處理記憶體，直到： • STARTFIFO 或 STOPRE 被偵測到 或 • 前置處理記憶體滿載 或 • 到達程式結尾
STARTFIFO:	STARTFIFO 開始加工步驟的快速處理；前置處理記憶體以平行於此的方式被填滿。
FIFOCTRL:	啟用自動前置處理記憶體控制系統
STOPRE:	停止前置處理

說明

前置處理記憶體未填滿，或是，如果加工步驟包含需要取消緩衝操作的指令（搜尋參照、量測函數等），填入會中斷。

說明

控制會在對狀態資料（$SA...）進行存取的事件中，產生一個內部的前置處理停止。

⚠小心

若已啟用刀具偏移量或曲線插補時，不可程式設計 STOPRE 指令，因為這會導致連續單節順序中斷。

範例：停止前置處理

程式碼	註解
...	
N30 MEAW=1 G1 F1000 X100 Y100 Z50	; 使用探針於第一個量測輸入及線性插補量測單節。
N40 STOPRE	; 停止前置處理
...	

8.4 條件式可中斷程式區段（DELAYFSTON, DELAYFSTOF）

功能

條件式可中斷工件程式區段稱為停止延遲區段。不應該發生**停止**，而且在某些程式區段中**進給**也不應該變更。基本上，短程式區段（例如，進行螺紋加工）應該受到保護，幾乎所有停止事件都不應影響到它。要一直等到完成程式區段之後，停止才會生效。

句法

```
DELAYFSTON
DELAYFSTOF
```

指令是在工件程式行中分別進行程式設計。

只有在工件程式中才認可這兩個指令，而在同步動作中並不認可。

意義

DELAYFSTON	定義範圍的開始，在其中“軟”停止會一直延遲，直到停止延遲區段結束為止。
DELAYFSTOF	定義停止延遲區段的結束

說明

使用機械參數 MD11550 $MN_STOP_MODE_MASK Bit 0 = 0（預設值）時，如果已啟用 G331/G332，而且已程式設計的路徑動作或 G4，停止延遲區段是以隱含方式定義。

範例：停止事件

在停止延遲區段中，會忽略**進給率和進給率抑制**中的變更；只有在停止延遲區段之後，才會生效。

停止事件分成：

“軟”停止事件	回應：延遲
“硬”停止事件	回應：立即

選擇一些停止事件，至少會導致短暫停止：

事件名稱	回應	中斷參數
重置	立即	IS：DB21，… DBX7.7 and DB11，… DBX20.7
PROG_END	警報 16954	NC 程式：M30
INTERRUPT	延遲	IS：FC-9 和 ASUP DB10，... DBB1
SINGLEBLOCKSTOP	延遲	已啟動停止延遲區段中的單一單節模式： NC 會在停止延遲區段之外的第一個單節結束時停止。 在停止延遲區段之前已經選擇的單一單節： NST：“NC 在單節限制停止”DB21，... DBX7.2
STOPPROG	延遲	IS：DB21，… DBX7.3 and DB11，… DBX20.5
PROG_STOP	警報 16954	NC 程式：M0 和 M1
WAITM	警報 16954	NC 程式：WAITM
WAITE	警報 16954	NC 程式：WAITE
STOP_ALARM	立即	警報：警報設定 STOPBYALARM
RETREAT_MOVE_THREAD	警報 16954	NC 程式：以 LFON 發出警報 16954 （在 G33 中不可能停止和快速抬起）
WAITMC	警報 16954	NC 程式：WAITMC
NEWCONF_PREP_STOP	警報 16954	NC 程式：NEWCONF
SYSTEM_SHUTDOWN	立即	以 840Di 關閉系統
ESR	延遲	延伸停止並返回
EXT_ZERO_POINT	延遲	外部的零點偏移
STOPRUN	警報 16955	OPI：PI "_N_FINDST" STOPRUN

回應說明

立即（“硬”停止事件）	即使在停止延遲區段中，也立即停止。
延遲（“軟”停止事件）	等到停止延遲區段之後才停止（連短期停止也沒有）。
警報 16954	程式已取消，因為停止延遲區段中已使用了不合規定的程式指令。
警報 16955	程式繼續，停止延遲區段中已發生不合規定的動作。
警報 16957	以 DELAYFSTON 和 DELAYFSTOF 包住的程式區段（停止延遲區段）無法啟動。每個停止都會在區段中立即發生作用，而且不可延遲。

如需一份其他回應停止事件的清單，請參閱：

參考：

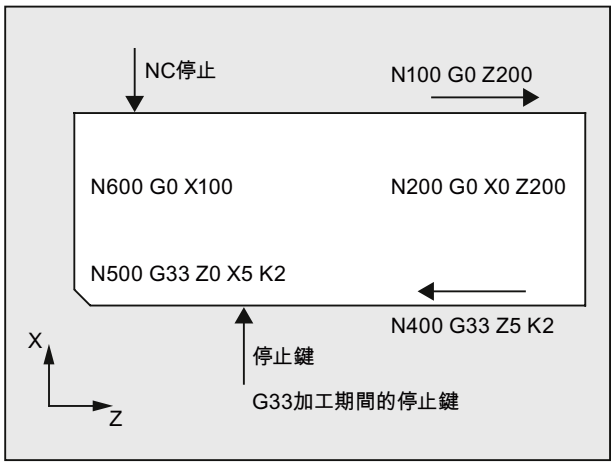
功能手冊基本功能；模式群組，通道，程式操作，（K1），“對停止事件的影響和衝擊”

範例：在兩個程式層級中巢狀嵌入停止延遲區段

程式碼	註解
N10010 DELAYFSTON（）	; 使用 N10xxx 程式層級 1 的單節。
N10020 R1 = R1 + 1	
N10030 G4 F1	; 停止延遲區段開始。
...	
N10040 subprogram2	
...	
...	; 副程式 2 的解譯
N20010 DELAYFSTON（）	; 不生效，重複開始，第二層級。
...	
N20020 DELAYFSTOF（）	; 不生效，在其他層級結束。
N20030 RET	
N10050 DELAYFSTOF（）	; 停止延遲區段在相同層級結束。
...	
N10060 R2 = R2 + 2	
N10070 G4 F1	; 停止延遲區段結束。從現在起，停止立即運作。

範例：程式摘錄

後面的程式單節是在迴圈中重複：



如圖示中所示，使用者在停止延遲區段中按下“停止”，NC 在停止延遲區段之外開始敘述，也就是，在單節 N100 中。因而導致 NC 在 N100 開始時停止。

程式碼	註解
...	
N99 MY_LOOP:	
N100 G0 Z200	
N200 G0 X0 Z200	
N300 DELAYFSTON（）	
N400 G33 Z5 K2 M3 S1000	
N500 G33 Z0 X5 K3	
N600 G0 X100	
N700 DELAYFSTOF（）	
N800 GOTOB MY_LOOP	

SERUPRO 類型單節搜尋和進給結合 G331/G332 進給，以進行無補償夾頭的攻牙相關詳細資訊，請參閱：

參考：

功能手冊基本功能；模式群組、通道、程式操作（K1）
功能手冊基本功能；進給率（V1）

停止延遲區段的優點

程式區段處理時並不會造成速率下降。

如果使用者以 RESET 停止之後取消程式，取消的程式單節是在受保護區段之後。此程式單節很適合做為後續單節搜尋的目標。

只要停止延遲區段正在作業，下列主要執行軸就不會停止：

- 指令軸以及
- 使用 POSA 移動的定位軸

工件程式指令 G4 在停止延遲區段中認可，而會造成暫停的其他工件程式指令則不受認可（例如，WAITM）。

G4 與路徑動作相同，可啟動停止延遲區段和 / 或讓它保持啟用。

範例：進給率干預

如果手動倍率是在停止延遲區段之前降為 6%，手動倍率會在停止延遲區段中啟用。

如果手動倍率是在停止延遲區段中從 100%降為 6%，停止延遲區段是以 100%完成，然後程式再以 6%繼續。

進給停用在停止延遲區段中沒有作用；該程式會等到停止延遲區段之後再停止。

覆疊 / 巢狀

如果兩個停止延遲區段重疊，一個來自 NC 指令，另一個來自機械參數 MD 11550：STOP_MODE_MASK，就會產生最大可能的停止延遲區段。

下列特性會利用巢狀和子程式結束，來調節管理 NC 指令 DELAYFSTON 與 DELAYFSTOF 之間的互動：

1. DELAYFSTOF 是在呼叫 DELAYFSTON 的子程式結束時以隱含方式啟動。
2. DELAYFSTON 停止延遲區段沒有作用。
3. 如果子程式 1 在停止延遲區段中呼叫子程式 2，整個子程式 2 都是停止延遲區段。特別是 DELAYFSTOF 在子程式 2 中就會沒有作用。

說明

REPOSA 是子程式指令的結束，而 DELAYFSTON 永遠都會取消選擇。

如果“硬”停止事件與“停止延遲區段”正好重疊，就會取消選擇整個“停止延遲區段。因此，如果在這個程式區段中發生任何其他停止，就會立即被阻止。必須新增程式設定（新增 DELAYFSTON），才能開始新的停止延遲區段。

如果在停止延遲區段之前按下“停止”鍵，NCK 必須移入停止延遲區段中進行煞車，NCK 會在停止延遲區段之中停止，而停止延遲區段就會維持為取消選擇。

以 0%覆疊輸入的停止延遲區段將**不會**被接受！

這適用於所有“軟”停止事件。

STOPALL 可用來在停止延遲區段中進行減速。但是 STOPALL 會立即啟動先前已延遲的其他所有停止事件。

系統變數

停止延遲區段可以在工件程式中，使用$P_DELAYFST 進行偵測。如果系統變數的位元 0 設定為 1，工件程式處理就會進入停止延遲區段。

停止延遲區段可以同步的動作中，使用$AC_DELAYFST 進行偵測。如果系統變數的位元 0 設定為 1，工件程式處理就會進入停止延遲區段。

相容性

機械參數 MD 11550 的預設值：STOP_MODE_MASK 位元 0 = 0 會在 G 代碼群組 G331/G332 期間，以及已程式設計的路徑動作或 G4 時，觸發隱含停止延遲區段。

位元 0 = 1 在 G 代碼群組 G331/G332 期間，以及已程式設計的路徑行為或 G4 （動作直到 SW 6）時，允許停止。必須使用 DELAYFSTON/DELAYFSTOF 指令來定義停止延遲區段。

8.5 防止 SERUPRO 的程式位置（IPTRLOCK，IPTRUNLOCK）

功能

在機台上某些複雜的加工情況下，必須停止單節搜尋 SERUPRO。

透過使用可程式設計的中斷指示器，可以在無法追蹤點之前干預“在中斷點搜尋”。

也可以在 NCK 尚無法重新輸入的工件程式區段中，定義無法追蹤的區段。當程式被中斷時，NCK 會記下最後處理的單節，然後可以經由 HMI 操作員介面進行搜尋。

句法

```
IPTRLOCK
IPTRUNLOCK
```

這些指令位於工件程式行中，並允許可程式設計的中斷指示器。

意義

IPTRLOCK	無法追蹤的程式區段開始
IPTRUNLOCK	無法追蹤的程式區段結尾

只有在工件程式中才認可這兩個指令，但在同步動作中並不認可。

範例

將兩個程式層級中無法追蹤的程式區段，使用隱含的 IPTRUNLOCK，組成巢狀。副程式 1 中的隱含 IPTRUNLOCK 會結束無法追蹤的區段。

程式碼	備註
N10010 IPTRLOCK（）	
N10020 R1 = R1 + 1	
N10030 G4 F1	; 暫停開始搜尋停用程式區段的單節。
...	
N10040 subprogram2	
...	; 副程式 2 的解譯
N20010 IPTRLOCK （）	; 沒有作用，重複開始。
...	
N20020 IPTRUNLOCK （）	; 不生效，在其他層級結束。
N20030 RET	
...	
N10060 R2 = R2 + 2	
N10070 RET	; 搜尋停用程式區段結束。
N100 G4 F2	; 繼續主程式。

中斷指示器就會再度在 100 產生中斷。

取得及尋找無法追蹤的區段

非可搜尋程式區段是使用語言指令 IPTRLOCK 和 IPTRUNLOCK 進行識別。

IPTRLOCK 指令會在主要執行中可執行的單一單節凍結中斷指示器（SBL1）。下文中將此單節稱為暫停單節。如果在 IPTRLOCK 之後取消程式，此暫停單節可以從 HMI 使用者介面進行搜尋。

從目前單節繼續

中斷指示器是使用 IPTRUNLOCK 放置於目前的單節上，做為後續程式區段的中斷點。

找到搜尋目標之後，就可以使用暫停單節重複新的搜尋目標。

必須再度經由 HMI 移除使用者編輯的中斷指示器。

巢狀的規則：

下列特性會利用巢狀和子程式結束，來調節管理 NC 指令 IPTRLOCK 與 IPTRUNLOCK 之間的互動：

1. IPTRLOCK 是在呼叫 IPTRUNLOCK 的子程式結束時以隱含方式啟動。
2. 無法追蹤區段中的 IPTRLOCK 沒有作用。
3. 如果子程式 1 在無法追蹤的區段中呼叫子程式 2，整個子程式 2 都會無法追蹤。特別是 IPTRUNLOCK 在子程式 2 中就會沒有作用。

如需詳細資訊，請參閱
/FB1/ 功能手冊，基本功能；模式群組、通道、程式操作模式（K1）。

系統變數

無法追蹤的區段可以在工件程式中，使用$P_IPTRLOCK 進行偵測。

自動中斷指示器

自動中斷指示器會自動將先前已定義的耦合類型定義為無法追蹤。以下項目的機械參數

- 電子變速箱與 EGON
- 與 LEADON 耦合的軸螺距值

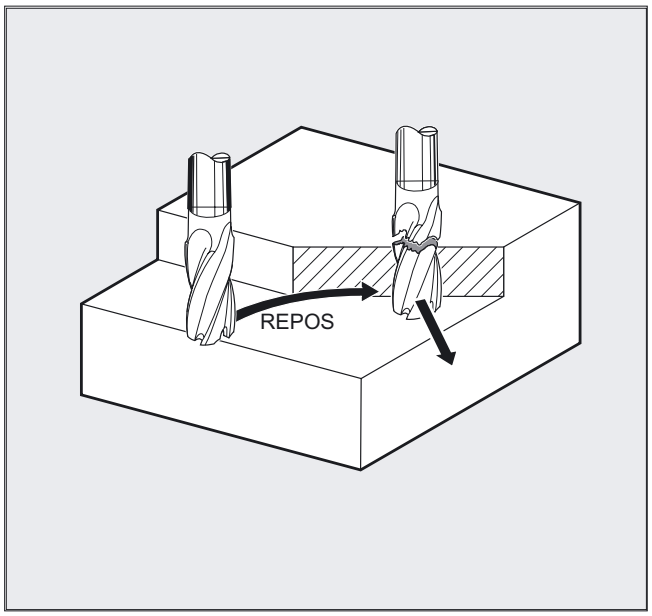
會用來啟動自動中斷指示器。如果已程式設計的中斷指示器與使用自動中斷指示器啟動的中斷指示器覆疊，就會產生可能的最大無法追蹤區段。

8.6 重新定位至輪廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN）

功能

如果在加工操作期間中斷程式執行，並讓刀具回退，例如，由於刀具已破損，或您要檢查量測，可以在輪廓上選擇任何一點在程式控制下重新定位。

REPOS 指令的運作方式與副程式回轉跳躍相同（例如，經由 M17）。在中斷程式中指令之後已程式設計的單節不會執行。



如需有關中斷程式執行的資訊，也請參閱本程式設計手冊中“中斷程式”一章的“彈性 NC 程式設計”一節。

句法

```
REPOSA RMI DISPR=…
REPOSA RMB
REPOSA RME
REPOSA RMN
REPOSL RMI DISPR=…
REPOSL RMB
REPOSL RME
REPOSL RMN
REPOSQ RMI DISPR=… DISR=…
REPOSQ RMB DISR=…
REPOSQ RME DISR=…
REPOSQA DISR=…
REPOSH RMI DISPR=… DISR=…
REPOSH RMB DISR=…
REPOSH RME DISR=…
REPOSHA DISR=…
```

意義

逼近路徑

REPOSA	在所有軸上沿直線逼近
REPOSL	沿直線逼近
REPOSQ DISR=…	使用半徑 DISR，沿象限弧逼近
REPOSQA DISR=…	在所有軸上，使用半徑 DISR，沿象限弧逼近
REPOSH DISR=…	使用直徑 DISR，沿半圓弧逼近
REPOSHA DISR=…	在所有軸上，使用直徑 DISR，沿半圓弧逼近

重新逼近點

RMI	逼近中斷點
RMI DISPR=…	在中斷點之前，毫米 / 英吋 DISPR 距離的進入點
RMB	逼近單節起點
RME	逼近單節結尾
RME DISPR=…	在終點之前距離 DISPR 逼近單節結束點
RMN	在最接近的路徑點逼近
A0 B0 C0	要進行逼近的軸

範例：沿直線逼近 REPOSA，REPOSL

刀具沿直線逼近重新定位點。

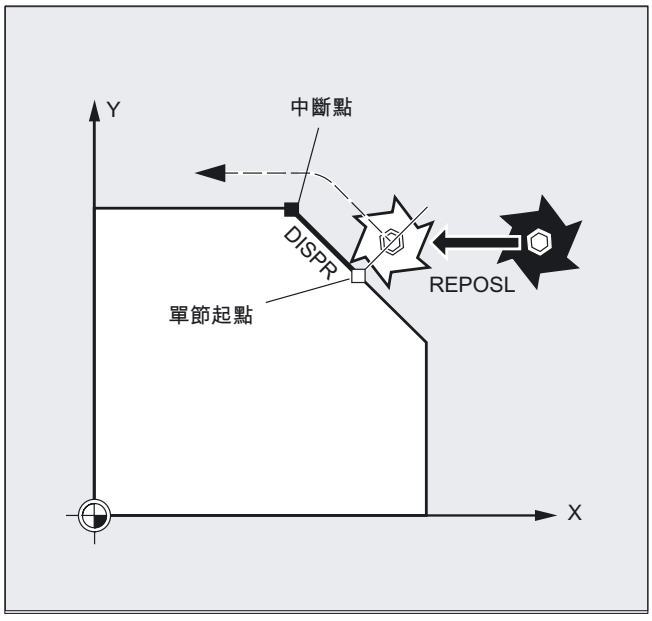
所有軸都會自動隨 REPOSA 指令移動。您可以使用 REPOSL，指定要移動的軸。

範例：

```
REPOSL RMI DISPR=6 F400
```

或

```
REPOSA RMI DISPR=6 F400
```

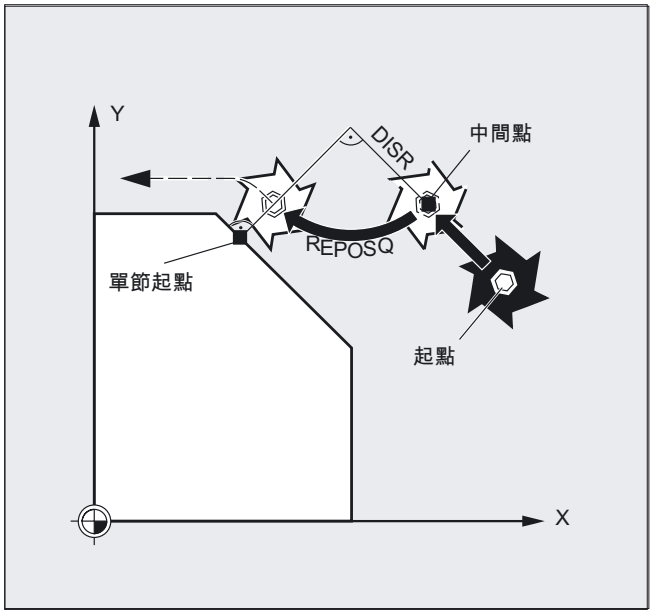


範例：以圓弧象限逼近，REPOSQ，REPOSQA

刀具以 DISR=…的半徑，沿象限弧逼近重新定位點。控制系統會自動計算起點與重新定位點之間的中間點。

範例：

```
REPOSQ RMI DISR=10 F400
```

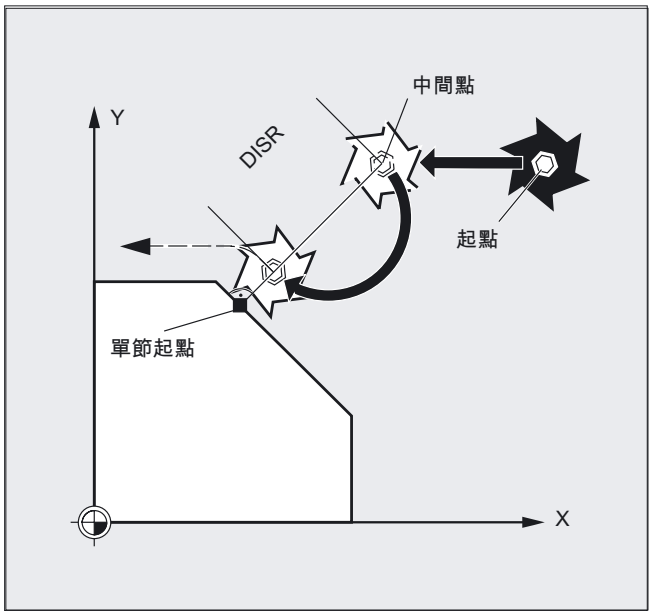


範例：以半圓逼近刀具，REPOSH，REPOSHA

刀具以 DISR=…的直徑，沿半圓弧逼近重新定位點。控制系統會自動計算起點與重新定位點之間的必要中間點。

範例：

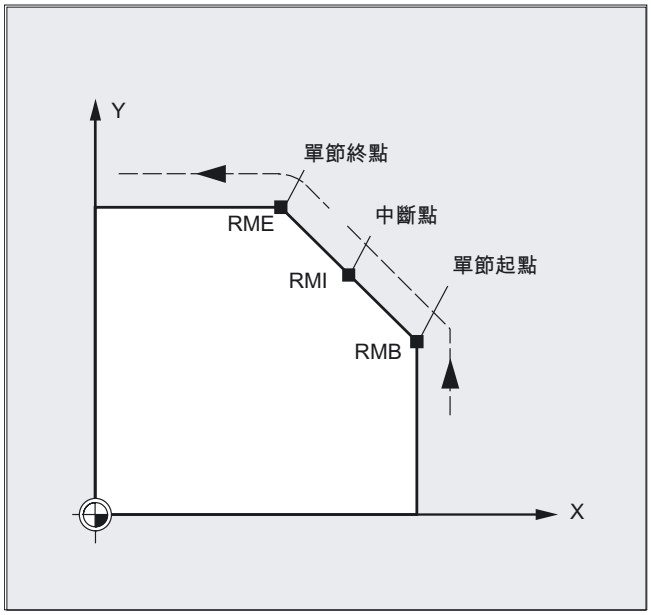
```
REPOSH RMI DISR=20 F400
```



指定重新定位點（不適用於 SERUPRO 使用 RMN 逼近）

在參考已中斷程式執行的 NC 單節下，可以選擇三個不同重新定位點的其中一個：

- RMI，中斷點
- RMB，單節起點或上一個終點
- RME，單節結束點



RMI DISPR=…或 RME DISPR=…允許您選擇位於中斷點或單節結束點之前的重新定位點。

DISPR=…允許您以毫米 / 英吋定義重新定位點與終點**前**中斷之間的輪廓距離。即使是高值，這個點也不能超過單節起點。

如果沒有程式設計 DISPR=…指令，則套用 DISPR=0，加上中斷點（使用 RMI）或單節結束點（使用 RME）。

DISPR 符號

DISPR 符號會進行評估。如果是加號，行為如前。

如果是減號，逼近是在中斷點之後，或使用 RMB 時，在單節起點之後。

中斷點與逼近點之間的距離取決於 DISPR 的值。即使值較高，這個點最遠只能位於單節結束點。

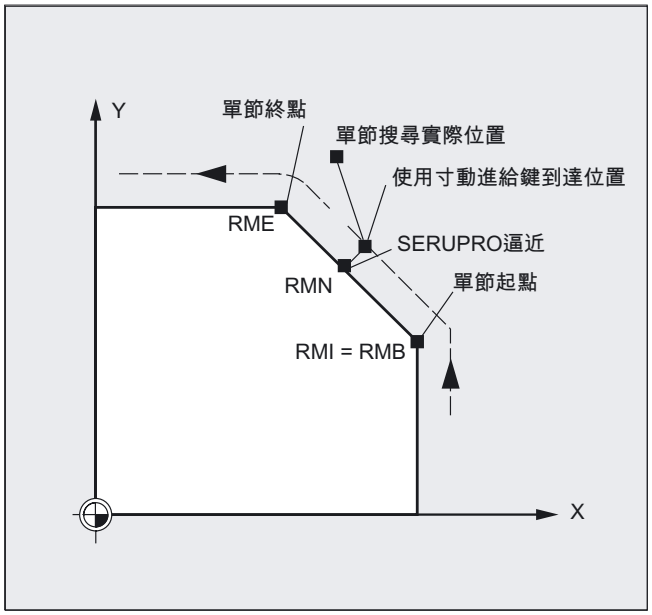
範例應用程式：

感應器會識別逼近夾具。會啟動 ASUP，以略過夾具。

之後，負值 DISPR 在夾具之後一點上重新定位，然後程式繼續執行。

SERUPRO 使用 RMN 逼近

如果在任何位置進行加工期間強迫取消，則使用 SERUPRO 逼近和 RMN，逼近離取消點最短的路徑，以便之後只處理剩餘距離。使用者在中斷單節啟動 SERUPRO 程序，然後使用寸動進給鍵，以移至目標單節的問題元件之前。



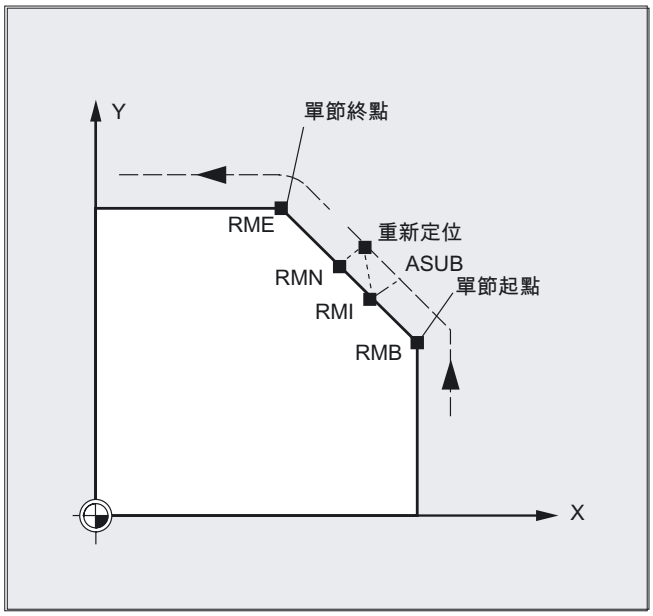
說明

SERUPRO

對 SERUPRO 來說，RMI 和 RMB 是完全相同。RMN 不是僅限用於 SERUPRO，而是一般都適用。

從最接近的路徑點逼近 RMN

解譯 REPOSA 時，使用 RMN 的重新定位單節在中斷後並不是完整啟動，而是只處理剩餘距離；逼近的是中斷單節的最接近路徑點。



有效 REPOS 模式的狀態

中斷單節的有效 REPOS 模式可以使用同步動作和變數$AC_ REPOS_PATH_MODE 進行讀取：

0: 未定義逼近

1 RMB：逼近開始處

2 RMI：逼近中斷點

3 RME：逼近單節結尾

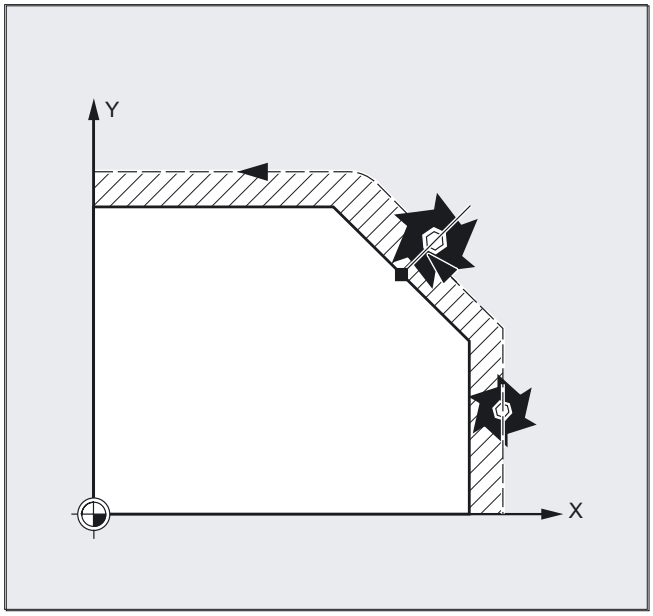
4 RMN：逼近中斷單節的下一個路徑點

使用新增刀具逼近

如果由於刀具破損而停止程式執行，則適用下列情形：

已程式設計新增 D 數量時，加工程式是以修改的刀具偏移值，在重新定位點繼續執行。

已修改刀具偏移值之處，也許無法重新逼近中斷點。在此情況下，就逼近新輪廓上最接近中斷點的點（可能是由 DISPR 修改）。



逼近輪廓

刀具在輪廓上重新定位的動作可以程式設計。為要移動的軸位址輸入零點。

REPOSA、REPOSQA 和 REPOSHA 指令會自動將所有軸重新定位。各個軸名稱並不需要重新指定。

程式設計 REPOSL、REPOSQ 和 REPOSH 時，所有幾何軸都會自動移動，也就是，不需要在指令中指定。其他所有軸都必須在指令中指定。

下列情形適用於 REPOSH 和 REPOSQ 圓弧動作：

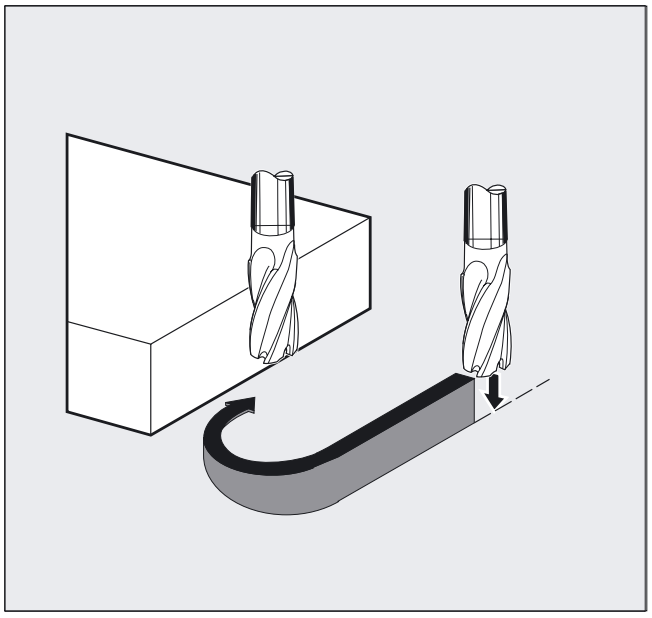
圓弧是在指定的工作平面 G17 到 G19 中移動。

如果在逼近單節中指定第三個幾何軸（進給方向），如果刀具位置與進給方向的程式設計位置並未重疊，就會沿螺旋線逼近重新定位點。

在下列情況中，控制會自動

切換成線性逼近 REPOSL：

- 您並未指定 DISR 的值。
- 沒有定義的逼近方向可使用（在單節中程式設計中斷而沒有移動資訊）。
- 逼近方向與目前的工作平面垂直相交。



8.7 影響動作控制

8.7.1 比例震動校正（JERKLIM）

功能

在重要程式區段中，可能必須將震動限制在最大可能值之下，例如，以減低機台上的壓力層級。加速度模式 SOFT 必須有效。此函數只影響路徑軸。

句法

```
JERKLIM[<軸>]=<值>
```

意義

JERKLIM	震動校正的指令
<軸>	要手動倍率震動臨界值的機械軸。
<值>	依參考機械參數中為軸設定值所允許的最大震動比例變更。 值域： 1 ... 200 值 100 不會影響震動。此設定會在重置並啟動工件程式後生效。

範例

在 AUTOMATIC 操作模式下，已程式設計的軸震動臨界值是限制於儲存在機械參數中震動臨界值的指定比例：

```
N60 JERKLIM[X]=75
```

含義：以 X 方向滑動的軸應該只有在軸所允許的 75%震動值下才進行加速 / 減速。

8.7.2 比例速率修正（VELOLIM）

功能

在重要程式區段中，可能必須將速率限制於最大可能值之下，例如，在減低機台上壓力層級的情況下，或提升加工品質的情況。此函數只影響路徑和定位軸。

句法

```
VELOLIM[<軸>]=<值>
```

意義

VELOLIM	速率校正的指令
<軸>	應該手動倍率速率限制值的機械軸。
<值>	依參考機械參數中為軸設定值所允許的最大速率比例變更。 值域： 1 ... 100 速率不受值 100 的影響，這個設定是在重置並啟動工件程式後生效。

範例

在 AUTOMATIC 操作模式下，程式設計軸的速率限制值是限制於儲存在機械參數中速率限制值的指定比例：

```
N70 VELOLIM[X]=80
```

含義：以 X 方向滑動的軸應該只有在軸所允許的 80%速率時才移動。

8.7.3 JERKLIM 和 VELOLIM 的程式範例

下列程式示範依比例限制震動和速率的應用程式範例：

程式碼	註解
N1000 G0 X0 Y0 F10000 SOFT G64	
N1100 G1 X20 RNDM=5 ACC[X]=20	
ACC[Y]=30	
N1200 G1 Y20 VELOLIM[X]=5	; 以 X 方向滑動的軸應該只有在軸所允許的最多 5%速率時才移動。
JERKLIM[Y]=200	; 以 X 方向滑動的軸可以在軸所允許的最大 200%震動值下進行加速 / 減速。
N1300 G1 X0 JERKLIM[X]=2	; 以 X 方向滑動的軸應該只有在軸所允許的最大 2%震動值下才進行加速 / 減速。
N1400 G1 Y0	
M30	

8.8 可程式設計輪廓/方向允差（CTOL、OTOL、ATOL）

功能

CTOL、OTOL、與 ATOL 指令可用於調整針對壓縮函數（COMPON、COMPCURV、COMPCAD）、平滑化類型 G642、G643、G645、OST、以及使用 NC 程式中的機械函數與設定資料的方向 ORISON 所定義的加工允差。

程式設計值在被重新程式設計或以指派負值的方式刪除之前均保持有效。其亦會在程式結尾、通道重置、模式群組重置、NCK 重置（熱重開機）、以及啟動（冷重開機）時被刪除。在刪除這些值時，會將來自機械參數與設定資料的值重新儲存。

句法

```
CTOL=<值>
OTOL=<值>
ATOL[<軸>]=<值>
```

意義

CTOL　用於程式設計**輪廓允差**的指令

CTOL 可用於：

- 所有的壓縮函數
- 除了 G641 與 G644 以外的所有倒圓角類型

<值>：　輪廓允差的值指定為長度。

類型：　REAL

單位：　inch/mm（視目前的尺寸設定而定）

OTOL　用於程式設計**方向允差**的指令

OTOL 可用於：

- 所有的壓縮函數
- ORISON 方向平滑化
- 除了 G641、G644、OSD 以外的所有倒圓角類型

<值>：　方向允差的值指定為角度。

類型：　REAL

單位：　度

ATOL　用於程式設計**軸專屬公差**的指令

ATOL 可用於：

- 所有的壓縮函數
- ORISON 方向平滑化
- 除了 G641、G644、OSD 以外的所有倒圓角類型

<軸>：　要程式設計軸允差的軸名稱

<值>：　軸允差的值會視軸的類型（線性或旋轉軸）指定為長度或角度。

類型：　REAL

單位：　用於線性軸：　inch/mm（視目前的尺寸設定而定）

用於旋轉軸：　度

說明

CTOL 與 OTOL 的優先順序高於 ATOL。

一般條款

比例調整框架

比例調整框架會以和軸位置相同的方式影響程式設計的允差，換言之，相對允差會維持不變。

範例

程式碼	註解
COMPCAD G645 G1 F10000	; 啟用 COMPCAD 壓縮函數。
X... Y... Z...	; 機械參數與設定資料套用於此。
X... Y... Z...	
X... Y... Z...	
CTOL=0.02	; 從此處開始套用 0.02 mm 的輪廓允差。
X... Y... Z...	
X... Y... Z...	
X... Y... Z...	
ASCALE X0.25 Y0.25 Z0.25	; 從此處開始套用 0.005 mm 的輪廓允差。
X... Y... Z...	
X... Y... Z...	
X... Y... Z...	
CTOL=–1	; 機械參數與設定資料再次從此處開始套用。
X... Y... Z...	
X... Y... Z...	
X... Y... Z...	

其他資訊

讀取允差值

在更進階的應用或在診斷時，可透過系統變數讀取壓縮函數（COMPON、COMPCURV、COMPCAD）目前有效的允差、平滑化類型 G642、G643、G645、OST、以及方向平滑化 ORISON 而不考慮其原因。

- 在同步動作或透過系統變數在工件程式中產生前置處理停止：

$AC_CTOL	當處理目前的主要執行記錄時，輪廓允差啟動。 若無有效的輪廓允差，$AC_CTOL 會傳回從幾何軸允差平方和得到的平方根值。
$AC_OTOL	當處理目前的主要執行記錄時，方向允差啟動。 若無有效的方向允差，$AC_OTOL 會傳回從在啟動方向轉換時方向軸允差平方和得到的平方根值。否則，其會傳回值"-1"。
$AA_ATOL[<軸>]	當處理目前的主要執行記錄時，軸允差啟動。 若無有效的輪廓允差，$AA_ATOL[<幾何軸>]會傳回除以幾何軸數根值的輪廓允差。 若啟用了方向允差及方向轉換，$AA_ATOL[<方向軸>]會傳回除以方向軸數根值的方向允差。

說明

若未程式設計允差值，$A 變數不會與個別函數中允差的潛在差異有充份的不同，因其僅可宣告一個值。

此種情況會在機械參數與設定資料為壓縮函數、平滑化及方向平滑化設定不同允差時出現。接著該變數會傳回對目前啟用函數有利的最大值。

例如，若壓縮函數啟用時的方向允差為 0.1°且 ORISON 方向平滑化為 1°，則$AC_OTOL 變數會傳回值"1"。若停用了方向平滑化，則僅會繼續讀取值"0.1"。

- 透過系統變數在工件程式中不產生前置處理停止：

$P_CTOL	程式設計的輪廓允差
$P_OTOL	程式設計的方向允差
$PA_ATOL	程式設計的軸允差

說明

若未程式設計允差值，則$P 變數會傳回值"-1"。

軸耦合

9

9.1 耦合動作（TRAILON，TRAILOF）

功能

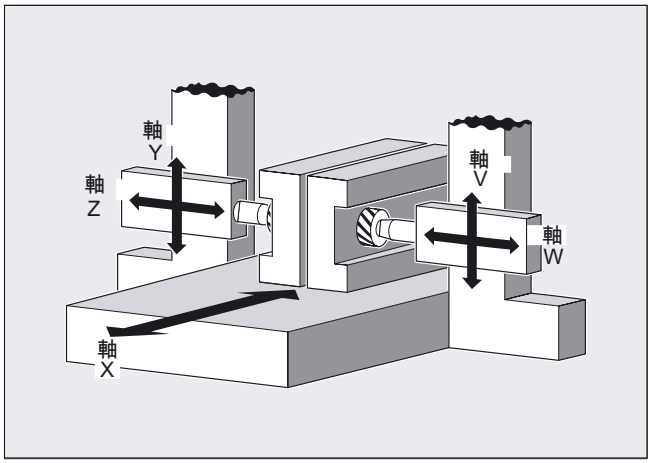
已定義的先導軸移動時，指派給它的耦合動作軸（=跟隨軸）會移動通過由先導軸說明的距離，以允許耦合係數。

主導軸與及隨動軸同在，便構成耦合軸。

應用

- 藉由模擬軸移跨一個軸。主導軸是模擬軸，而耦合軸則是實軸。以此方式，在耦合係數功能下，實軸可被移跨。
- 使用 2 個耦合動作群組進行雙側加工：

  1. 先導軸 Y，耦合動作軸 V
  2. 先導軸 Z，耦合動作軸 W



句法

```
TRAILON（<跟隨軸>、<先導軸>、<耦合係數>）
TRAILOF（<跟隨軸>、<先導軸>、<先導軸 2>）
TRAILOF（<跟隨軸>）
```

意義

TRAILON	用於啟用及定義已耦合的軸群組之指令 主動: 模態
<跟隨軸>	參數 1：尾隨軸的軸名稱 **注意：** 已耦合的動作軸亦可作為其他已耦合之動作軸的先導軸。以此方法，可以創造一系列不同的耦合軸組合。
<先導軸>	參數 2：尾隨軸的軸名稱
<耦合係數>	參數 3：耦合係數 耦合係數指定所想要的耦合動作軸之路徑與先導軸之間關係。 **<**耦合係數**> =** 耦合動作軸的路徑 / 先導軸的路徑 類型： REAL 預設值： 1 輸入負值會導致主動及耦合軸以相反方向移動。 如果耦合係數未程式設計，則會自動套用耦合係數 1。
TRAILOF	用於停用已耦合之軸群組的指令 有效範圍: 模態 具有 2 個參數的 TRAILOF 僅會啟用指定先導軸的耦合: TRAILOF（<跟隨軸>、<先導軸>） 若已耦合的動作軸有 2 個先導軸，則可使用 3 個參數呼叫 TRAILOF 以便將這兩個耦合均停用。 TRAILOF（<跟隨軸>、<先導軸>、<先導軸 2>） 程式設計 TRAILOF 時不指定先導軸會產生相同的結果: TRAILOF（<跟隨軸>）

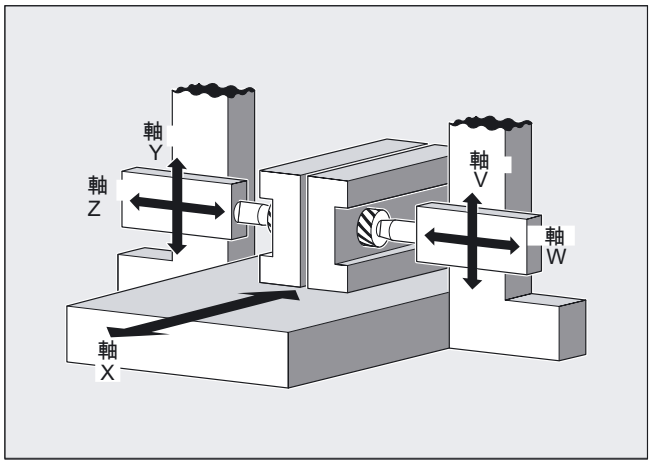
說明

耦合軸動作永遠都是在基準座標系統（BCS）中執行。

能同時被啟動的耦合軸組數量，僅被機台上軸組合的最大可能數量所限制。

範例

工件要使用圖示中顯示的軸設定，在雙側進行加工。若要執行這項作業，要建立兩個耦合軸組合。



程式碼	註解
…	
N100 TRAILON（V，Y）	; 啟用第一個耦合軸群組
N110 TRAILON（W，Z，–1）	; 啟用第 2 個已耦合的軸群組，負耦合係數：；耦合動作軸移動；與先導軸的方向相反。
N120 G0 Z10	; 朝與軸方向相反的方向進給 Z 軸與 W 軸。
N130 G0 Y20	; Y 和 V 軸依相同的軸方向進給。
…	
N200 G1 Y22 V25 F200	; 將已耦合之動作軸 V 的相依與獨立移動重疊。
…	
TRAILOF（V，Y）	; 停用第一個耦合軸群組
TRAILOF（W，Z）	; 停用第二個耦合軸群組

其它資訊

軸類型

耦合軸群組可以包含任何想要的線性軸和旋轉軸組合。模擬軸也可以定義為先導軸。

耦合動作軸

最多兩個先導軸可以同時指派給尾隨軸。指派是以不同的耦合軸組合進行。

已耦合之動作軸可使用完整範圍的可用動作指令（G0、G1、G2、G3、等）進行程式設計。耦合軸會根據耦合係數，不但移動獨立定義的路徑，也移動自其先導軸衍生的路徑。

動態臨界值

動態臨界值視已耦合之軸群組的啟用類型而定：

- 在工件程式中啟用

  若在工件程式中執行啟用且所有的先導軸均在啟用的通道中啟用作為程式設計軸，則在移動先導軸時請將所有已耦合的動作軸之動態回應列入考慮以避免已耦合的動作軸過載。

  若在工件程式中未將啟用通道中的先導軸啟用為程式設計軸（$AA_TYP ≠ 1）而執行啟用，則在先導軸移動時不需將已耦合的動作軸之動態回應列入考慮。如此會造成已耦合之動作軸具有少於比耦合所需要的動態回應而過載。

- 已同步動作啟用

  若在同步動作中執行啟用，已耦合之動作軸的動態回應在先導軸移動時不需列入考慮。如此會造成已耦合之動作軸具有少於比耦合所需要的動態回應而過載。

⚠小心
若啟用了已耦合的軸群組： • 在同步動作中 • 在具有在已耦合的動作軸之通道中非程式軸的先導軸之工件程式中 採取適當的動作確保移動先導軸時不會使已耦合的動作軸過載為使用者/機台製造商的責任。

耦合狀態

工件程式中的軸耦合狀態可使用以下系統變數確認：

$AA_COUP_ACT[<axis>]

值	意義
0	沒有啟用耦合
8	已啟用耦合動作

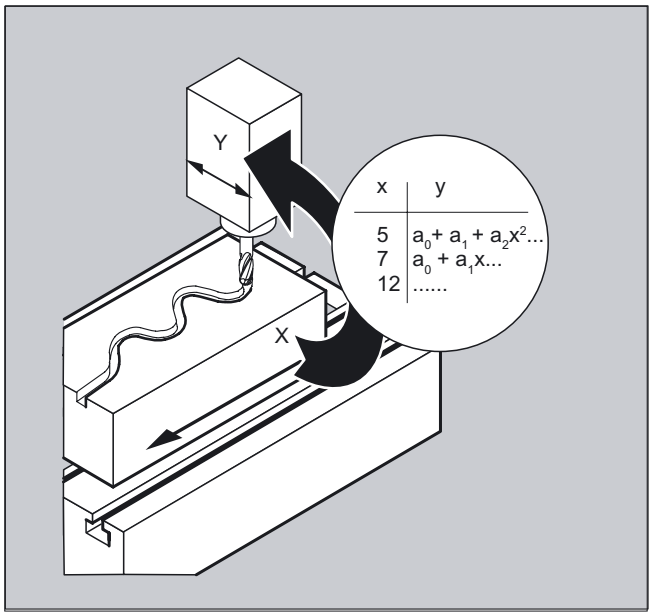
9.2 曲線表（CTAB）

功能

曲線表可用於程式設計兩軸（先導軸與跟隨軸）間的位置與速率關係。曲線表是在工件程式中定義的。

應用

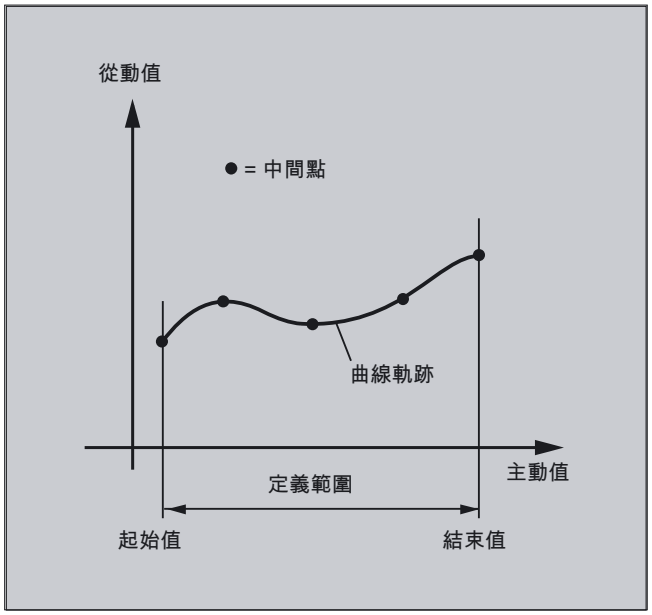
以曲線表取代機械檔塊。曲線表透過建立先導值與跟隨值之間函數關係，形成軸主動值耦合的基礎：在適當的程式設計下，控制會從先導軸和跟隨軸的相對位置，計算對應於擋塊的多項式。



9.2.1 定義曲線表（CTABDEF、CATBEND）

功能

曲線表代表工件程式或工件程式的區段，是用 CTABDEF 在開始處，並用 CTABEND 在結尾閉鎖。
在此工件程式區段中，使用動作操作將唯一的跟隨軸位置指派至先導軸的個別位置中；這些跟隨軸位置可在以最多五次多項式的行式計算曲線定義時作為中間點。



條件

該 MD 必須做對應的設定以確保有保留足夠的記憶體空間給曲線表的定義使用（→機台製造商）。

句法

CTABDEF（<跟隨軸>、<先導軸>、<n>、<週期性>[、<記憶體位置>]）
...
CTABEND

意義

CTABDEF （ ）	曲線表定義的起點
CTABEND	曲線表定義的結尾
<跟隨軸>	使用曲線表計算動作的軸
<先導軸>	提供計算跟隨軸動作所需之主動值的軸
<n>	曲線表的編號（**ID**）
	曲線表的編號是唯一且與記憶體位置無關的。因此，靜態與動態 NC 記憶體中不可能同時存在相同編號的曲線表。
<週期性>	表週期性
	0 表格非週期性（該表僅處理一次，旋轉軸亦同）
	1 與先導軸相關時，表是週期性
	2 與先導軸和跟隨軸相關時，表是週期性
<記憶體位置>	指定記憶體位置（選用）
	“SRAM” 該曲線表建立在**靜態** NC 記憶體中。
	“DRAM” 該曲線表建立在**動態** NC 記憶體中。
	注意： 若未替此參數程式設計值，會使用以 MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE 設定的預設記憶體位置。

說明

覆寫

一旦其編號（<n>）在其他曲線表定義中使用後，便會將曲線表覆寫（例外：曲線表在軸耦合中啟用或以 CTABLOCK 加以鎖定）。**覆寫曲線表時，不會輸出任何警告。**

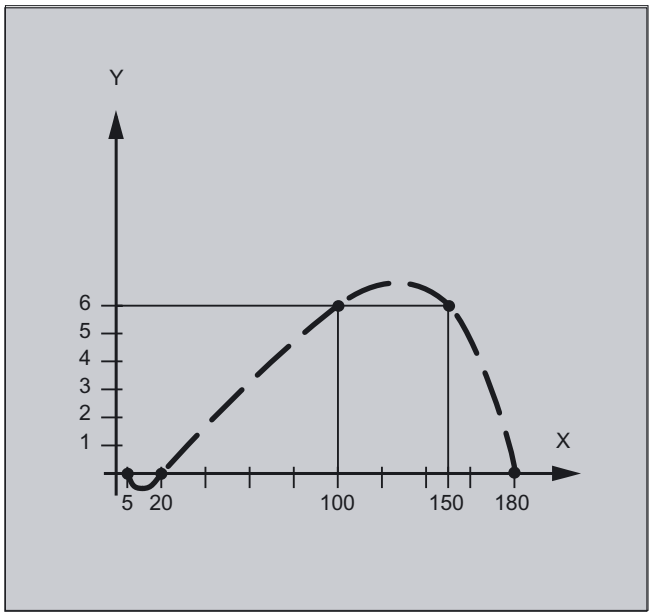
範例

範例 1：曲線表定義的程式區段

必須使用未變更的程式區段，以供定義曲線表。處理停止 STOPRE 的指令可以保留，只要程式區段未用於表定義，而且已移除 CTABDEF 和 CTABEND，就會立即再度啟用：

程式碼	註解
…	
CTABDEF（Y, X, 1, 1）	; 曲線表的定義。
…	
IF NOT （$P_CTABDEF）	
STOPRE	
ENDIF	
…	
CTABEND	

範例 2：非週期性曲線表的定義



程式碼	註解
N100 CTABDEF（Y，X，3，0）	; 定義的開始；非週期性曲線表，編號 3。
N110 X0 Y0	; 第 1 動作操作，定義起始值和第一個中間點： 主動值：0，跟隨值：0
N120 X20 Y0	; 第 2 中間點 主動值：0…20，跟隨值：起始值…0
N130 X100 Y6	; 第 3 中間點： 主動值：20…100，跟隨值：0 到 6
N140 X150 Y6	; 第 4 中間點： 主動值：100…150，跟隨值：6 到 6
N150 X180 Y0	; 第 5 中間點： 主動值：150…180，跟隨值：6 到 0
N200 CTABEND	; 定義結束。曲線表以其內在表象最高 5 次多項式的方式產生。使用指定中間點進行的曲線定義計算視模態選擇的插補類型（圓弧、線性、曲線插補）而定。開始定義前的工件程式狀態會重新儲存。

範例 3：週期性曲線表的定義

週期性曲線表的定義，編號 2，主動值域 0 到 360，跟隨軸動作從 0 到 45，然後回到 0：

程式碼	註解
N10 DEF REAL DEPPOS	
N20 DEF REAL GRADIENT	
N30 CTABDEF（Y，X，2，1）	；定義起始
N40 G1 X=0 Y=0	
N50 POLY	
N60 PO[X]=（45.0）	
N70 PO[X]=（90.0） PO[Y]=（45.0，135.0，-90）	
N80 PO[X]=（270.0）	
N90 PO[X]=（315.0） PO[Y]=（0.0，-135.0，90）	
N100 PO[X]=（360.0）	
N110 CTABEND	；定義結束
；透過耦合 Y 至 X，測試曲線：	
N120 G1 F1000 X0	
N130 LEADON（Y，X，2）	
N140 X360	
N150 X0	
N160 LEADOF（Y，X）	
N170 DEPPOS=CTAB（75.0，2，GRADIENT）	；讀取主動值 75.0 的表函數。
N180 G0 X75 Y=DEPPOS	；為先導軸和跟隨軸定位。
；啟動耦合之後，不需要跟隨軸的同步處理。	
N190 LEADON（Y，X，2）	
N200 G1 X110 F1000	
N210 LEADOF（Y，X）	
N220 M30	

其他資訊

曲線表的起始值和結束值

曲線表的定義範圍開始的起始值是曲線表定義內第一個指定的相關聯軸位置（第一個移動敘述）。曲線表定義範圍的結束值是根據最後的移動指令來決定。

可用的語言範圍

在曲線表的定義之內，您可以使用整個 NC 語言。

說明

在曲線表定義中不可使用以下項目：
- 前置處理停止
- 先導軸動作中的跳躍（例如在變更轉換時）
- 只有跟隨軸的移動敘述
- 先導軸的逆轉，也就是先導軸的位置必須永遠都是唯一的
- 在不同程式層級的 CTABDEF 與 CTABEND 操作。

模態操作的效果

表定義完成後，在曲線表定義內做的所有模態敘述都無效。因此，在其中完成表定義的工件程式在表定義之前和之後都處於相同狀態。

對 R 參數的配置

表格定義中對 R 參數的配置會在 CTABEND 之後重置。

範例：

程式碼	註解
...	
R10=5 R11=20	; R10=5
...	
CTABDEF	
G1 X=10 Y=20 F1000	
R10=R11+5	; R10=25
X=R10	
CTABEND	
...	; R10=5

啟動 ASPLINE，BSPLINE，CSPLINE

如果已在曲線表定義 CTABDEF（ ）... CTABEND 之內啟動 ASPLINE 、 BSPLINE 或 CSPLINE，至少有一個起點應該在啟用此曲線之前進行程式設計。必須避免 CTABDEF 之後立即啟用，否則曲線將取決於曲線表定義之前的目前軸位置。

範例：

程式碼

```
...
CTABDEF（Y，X，1，0）
X0 Y0
ASPLINE
X=5 Y=10
X10 Y40
...
CTABEND
```

重覆使用曲線表

如果選擇的曲線表已儲存於靜態 NC 記憶體（SRAM），利用曲線表計算得出的先導軸與跟隨軸函數關係，在工件程式結束之後以及關機之後，將仍保留於該表編號之下。

建立於動態記憶體（DRAM）中的曲線表將於開機後刪除，可能必須重新產生。

只要建立了曲線表，便可套用至任一先導軸與尾隨軸軸組合，且該表與建立曲線表時所使用的軸無關。

覆寫曲線表

一旦在其他表定義中使用某曲線表編號，將立刻覆寫該曲線表。

例外情形：曲線表在軸耦合中啟用或以 CTABLOCK 加以鎖定。

說明

覆寫曲線表時，不會輸出任何警告。

是否已啟用曲線表定義？

在工件程式中可隨時使用$P_CTABDEF 系統變數確認曲線表是否已啟用。

撤回曲線表定義

一旦排除了與曲線表定義有關的操作後，該工件程式區段便可再次作為實數工件程式使用。

使用"從外部來源執行"功能載入曲線表。

若從外部來源執行曲線表，則在 MD18360 $MN_MM_EXT_PROG_BUFFER_SIZE 中，重新載入緩衝器（動態記憶體）大小的選擇需支援能同時儲存重新載入緩衝器中所定義的曲線表項目。若否，則會取消處理並發出警報。

在跟隨軸中跳躍

依機械參數
MD20900 $MC_CTAB_ENABLE_NO_LEADMOTION
中的設定而定，如果先導軸中少了某個動作，可能會容許跟隨軸中的跳躍。

9.2.2 確認曲線表是否存在（CTABEXISTS）

功能

CTABEXISTS 指令可用於確認指定的曲線表編號是否存在 NC 記憶體中。

句法

```
CTABEXISTS（<n>）
```

意義

CTABEXISTS	確認靜態或動態 NC 記憶體中是否有存在曲線表編號<n>。 0 表格不存在 1 表格存在
<n>	曲線表的編號（ID）

9.2.3 刪除曲線表（CTABDEL）

功能

CTABDEL 可用來刪除曲線表。

說明

在軸耦合中啟用的曲線表無法刪除。

句法

```
CTABDEL（<n>）
CTABDEL（<n>，<m>）
CTABDEL（<n>，<m>，<記憶體位置>）
CTABDEL （）
CTABDEL（，，<記憶體位置>）
```

意義

CTABDEL	用於刪除曲線表的指令
<n>	待刪除的曲線表編號（**ID**）
	當刪除曲線表範圍 CTABDEL（<n>，<m>）時，<n>可用於指定該範圍中第一個曲線表的編號。
<m>	當刪除曲線表範圍 CTABDEL（<n>，<m>）時，<m>可用於指定該範圍中最後一個曲線表的編號。
	<m>需大於<n>。
<記憶體位置>	指定記憶體位置（選用）
	在**未**指定記憶體位置而進行刪除時，則會將靜態與動態 NC 記憶體中指定的曲線表刪除。
	在**有**指定記憶體位置而進行刪除時，則僅會將位於指定記憶體位置上的曲線表刪除。其餘會保留。
	“SRAM” 在**靜態** NC 記憶體中執行刪除
	“DRAM” 在**動態** NC 記憶體中執行刪除

若程式設計了 CTABDEL 但未指定欲刪除的曲線表時，則會將**所有**的曲線表或在指定記憶體中的所有曲線表均刪除：

CTABDEL （）	將靜態與動態 NC 記憶體中的所有曲線表均刪除
CTABDEL（，，“SRAM”）	將靜態 NC 記憶體中所有的曲線表均刪除
CTABDEL（，，“DRAM”）	將動態 NC 記憶體中所有的曲線表均刪除

說明

在使用 CTABDEL（<n>，<m>）或 CTABDEL（）中進行多重刪除時，若至少有一個待刪除的曲線表啟動為耦合狀態，則不會執行刪除指令；換言之，不會刪除**任何**處理的曲線表。

9.2.4 鎖住曲線表以防止刪除與覆寫（CTABLOCK、CTABUNLOCK）

功能

鎖定可設定用於保護曲線表免於被意外刪除或覆寫。一旦設定鎖定後，可隨時將其撤回。

句法

Lock：

```
CTABLOCK（<n>）
CTABLOCK（<n>，<m>）
CTABLOCK（<n>，<m>，<記憶體位置>）
CTABLOCK（）
CTABLOCK（，，<記憶體位置>）
```

取消鎖定：

```
CTABUNLOCK（<n>）
CTABUNLOCK（<n>，<m>）
CTABUNLOCK（<n>，<m>，<記憶體位置>）
CTABUNLOCK（）
CTABUNLOCK（，，<記憶體位置>）
```

意義

CTABLOCK	用於**設定**防止刪除/覆寫之鎖定的指令
CTABUNLOCK	用於**撤回**防止刪除/覆寫之鎖定的指令 CTABUNLOCK 可將使用 CTABLOCK 鎖定的曲線表取消鎖定。涉入已啟用耦合的曲線表保持為鎖定，而無法被刪除。使用 CTABLOCK 所做的鎖定會在因停用耦合而使因耦合所造成的鎖定取消後便取消鎖定。因此，這個表可以刪除。不必再次呼叫 CTABUNLOCK。
<n>	待鎖定/取消鎖定的曲線表編號（**ID**） 當鎖定/取消鎖定曲線表範圍 CTABLOCK（<n>，<m>）/CTABUNLOCK（<n>，<m>）後，<n>可用於指定該範圍中第一個曲線表的編號。
<m>	當鎖定/取消鎖定曲線表範圍 CTABLOCK（<n>，<m>）/CTABUNLOCK（<n>，<m>）後，<m>可用於指定該範圍中最後一個曲線表的編號。 <m>需大於<n>。
<記憶體位置>	指定記憶體位置（選用） 在**未**指定記憶體位置而進行鎖定/取消鎖定時，則會將靜態與動態 NC 記憶體中指定的曲線表鎖定/取消鎖定。 在**有**指定記憶體位置而進行鎖定/取消鎖定時，則僅會將位於指定記憶體位置上的曲線表鎖定/取消鎖定。其餘的不鎖定/取消鎖定。 “SRAM” 在**靜態** NC 記憶體中進行鎖定/取消鎖定 “DRAM” 在**動態** NC 記憶體中進行鎖定/取消鎖定

若程式設計了 CTABLOCK/CTABUNLOCK 但未指定欲鎖定/取消鎖定的曲線表時，則會將**所有**的曲線表或指定記憶體中的所有曲線表均鎖定/取消鎖定：

CTABLOCK（）	將靜態與動態 NC 記憶體中的所有曲線表均鎖定
CTABLOCK（，，“SRAM”）	將靜態 NC 記憶體中所有的曲線表均鎖定
CTABLOCK（，，“DRAM”）	將動態 NC 記憶體中所有的曲線表均鎖定
CTABUNLOCK（）	將靜態與動態 NC 記憶體中的所有曲線表均取消鎖定
CTABUNLOCK（，，“SRAM”）	將靜態 NC 記憶體中所有的曲線表均取消鎖定
CTABUNLOCK（，，“DRAM”）	將動態 NC 記憶體中所有的曲線表均取消鎖定

9.2.5 曲線表：決定取線表特性（CTABID、CTABISLOCK、CTABMEMTYP、CTABPERIOD）

功能

這些指令可用於選取曲線表的重要特性行（表編號、鎖定狀態、記憶體位置、週期性）。

句法

```
CTABID（<p>）
CTABID（<p>，<記憶體位置>）
CTABISLOCK（<n>）
CTABMEMTYP（<n>）
TABPERIOD（<n>）
```

意義

CTABID

傳回指定記憶體中第<p>個曲線表所輸入的**表編號**。

範例：

CTABID（1，"SRAM"）會傳回靜態 NC 記憶體中第一個曲線表的編號。此時，第一個曲線表便是具有最高表編號的曲線表。

注意：

若記憶體中的曲線表順序在連續呼叫 CTABID 時改變，例如因使用 CTABDEL 刪除曲線表時，CTABID（<p>，...）會傳回具有相同編號<p>的不同曲線表。

CTABISLOCK

傳回曲線表編號<n>的**鎖定狀態**：

0	表未鎖定
1	表被 CTABLOCK 鎖定
2	表被啟動耦合鎖定
3	表被 CTABLOCK 與啟用耦合鎖定
-1	表格不存在

CTABMEMTYP

傳回曲線表編號<n>的**記憶體位置**：

0	表在靜態 NC 記憶體中
1	表在動態 NC 記憶體中
-1	表格不存在

CTABPERIOD	傳回曲線表編號<n>的**週期性**： 0 表不是週期性 1 表在先導軸中週期運作 2 表在先導與跟隨軸中週期運作 -1 表格不存在
<p>	記憶體中的輸入編號
<n>	曲線表的編號（**ID**）
<記憶體位置>	指定記憶體位置（選用） “SRAM” **境態** NC 記憶體 “DRAM” **動態** NC 記憶體 **注意：** 若未替此參數程式設計值，會使用以 MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE 設定的預設記憶體位置。

9.2.6 讀取曲線表值（CTABTSV、CTABTEV、CTABTSP、CTABTEP、CTABSSV、CTABSEV、CTAB、CTABINV、CTABTMIN、CTABTMAX）

功能

下列曲線表值可在工件程式中讀取：

- 在曲線表之起點與結尾的跟隨軸與先導軸值
- 在曲線區段之起點與結尾的跟隨軸值
- 先導軸值的跟隨軸值
- 跟隨軸值的先導軸值
- 跟隨軸最小與最大值
  - 在曲線表的整個定義範圍中
    或
  - 在已定義的曲線表之間

句法

```
CTABTSV（<n>，<斜度>[，<跟隨軸>]）
CTABTEV（<n>，<斜度>[，<跟隨軸>]）
CTABTSP（<n>，<斜度>[，<先導軸>]）
CTABTEP（<n>，<斜度>[，<先導軸>]）
CTABSSV（<主動值>，<n>，<斜度>[，<跟隨軸>]）
CTABSEV（<主動值>，<n>，<斜度>[，<跟隨軸>]）
CTAB（<主動值>，<n>，<斜度>[，<跟隨軸>，<先導軸>]
CTABINV（<後續值>，<大略值>，<n>，<斜度>[，<跟隨軸>，<先導軸>]
CTABTMIN（<n>[，<跟隨軸>]）
CTABTMAX（<n>[，<跟隨軸>]）
CTABTMIN（<n>，<a>，<b>[，<跟隨軸>，<先導軸>]）
CTABTMAX（<n>，<a>，<b>[，<跟隨軸>，<先導軸>]）
```

意義

CTABTSV	在曲線表編號<n>的**起點**讀取跟隨軸值。
CTABTEV	在曲線表編號<n>的**結尾**讀取跟隨軸值。
CTABTSP	在曲線表編號<n>的**起點**讀取先導軸值。
CTABTEP	在曲線表編號<n>的**結尾**讀取先導軸值。
CTABSSV	在屬於指定先導軸值的曲線區段**起點**讀取跟隨軸值（<主動值>）
CTABSEV	在屬於指定先導軸值的曲線區段**結尾**讀取跟隨軸值（<主動值>）
CTAB	讀取指定先導軸值（<主動值>）的跟隨軸值
CTABINV	讀取指定跟隨軸值（<跟隨值>）的先導軸值
CTABTMIN	定義跟隨軸**最小值**： • 在曲線表的整個定義範圍中 或 • 在定義的區間<a>至<b>
CTABTMAX	定義跟隨軸**最大值**： • 在曲線表的整個定義範圍中 或 • 在定義的區間<a>至<b>
<n>	曲線表的編號（**ID**）
<斜度>	<斜度>參數會傳回計算位置上的曲線表函數之**傾斜度**。
<跟隨軸>	使用曲線表計算動作的軸（選用）
<先導軸>	提供計算跟隨軸動作所需之主動值的軸（選用）
<後續值>	用於使用 CTABINV 讀取相關先導軸值的先導軸值
<主動值>	先導軸值： • 用於使用 CTAB 讀取相關跟隨軸值 或 • 用於使用 CTABSSV/CTABSEV 選擇曲線區段
<大約值>	使用 CTABINV 指派至跟隨軸的先導軸值不可每次是唯一的。因此，CTABINV 需要針對期望的先導軸值提供一個大略值作為參數。
<a>	具有 CTABTMIN/CTABTMAX 的主動值區間**下**臨界值
<b>	具有 CTABTMIN/CTABTMAX 的主動值區間**上**臨界值 **注意：** 主動值區間<a>至<b>永遠都必須在曲線表的定義範圍內。

範例

範例 1：

在曲線表的開頭與結尾處定義跟隨軸與先導軸值，其中跟隨軸的最大與最小值均在整個曲線表的定義範圍內。

程式碼	註解
N10 DEF REAL STARTPOS	
N20 DEF REAL ENDPOS	
N30 DEF REAL STARTPARA	
N40 DEF REAL ENDPARA	
N50 DEF REAL MINVAL	
N60 DEF REAL MAXVAL	
N70 DEF REAL GRADIENT	
...	
N100 CTABDEF（Y，X，1，0）	; 表定義開始
N110 X0 Y10	; 第一表區段起始位置
N120 X30 Y40	; 第一表區段結束位置 = 第二表區段開始位置
N130 X60 Y5	; 結束位置第 2 表區段 = 等等。
N140 X70 Y30	
N150 X80 Y20	
N160 CTABEND	; 表定義結尾
...	
N200 STARTPOS = CTABTSV（1，GRADIENT）	; 曲線表開頭的跟隨軸值= 10
N210 ENDPOS = CTABTEV（1，GRADIENT）	; 曲線表結尾的跟隨軸值= 20
N220 STARTPARA=CTABTSP（1，GRADIENT）	; 曲線表開頭的先導軸值= 0
N230 ENDPARA = CTABTEP（1，GRADIENT）	; 曲線表結尾的先導軸值= 80
N240 MINVAL = CTABTMIN（1）	; 當 Y=5 時跟隨軸的最小值
N250 MAXVAL = CTABTMAX（1）	; 當 Y=40 時跟隨軸的最大值

範例 2：

搭配先導軸值 X=30 決定曲線區段開頭與結尾處的跟隨軸值。

程式碼	註解
N10 DEF REAL STARTPOS	
N20 DEF REAL ENDPOS	
N30 DEF REAL GRADIENT	
...	
N100 CTABDEF（Y，X，1，0）	; 表定義開始
N110 X0 Y0	; 第一表區段起始位置
N120 X20 Y10	; 第一表區段結束位置 = 第二表區段開始位置
N130 X40 Y40	結束位置第 2 表區段 = 等等。
N140 X60 Y10	
N150 X80 Y0	
N160 CTABEND	; 結束表定義
...	
N200 STARTPOS=CTABSSV（30.0，1，GRADIENT）	; 第二區段開始位置 Y = 10
N210 ENDPOS=CTABSEV（30.0，1，GRADIENT）	; 第二區段結束位置 Y = 40

其他資訊

於同步動作中使用

所有用來讀取曲線表值的指令，也可用於同步動作中（請參考"動作-同步動作"章節）。

當使用 CTABINV，CTABTMIN，與 CTABTMAX 指令，請確認：

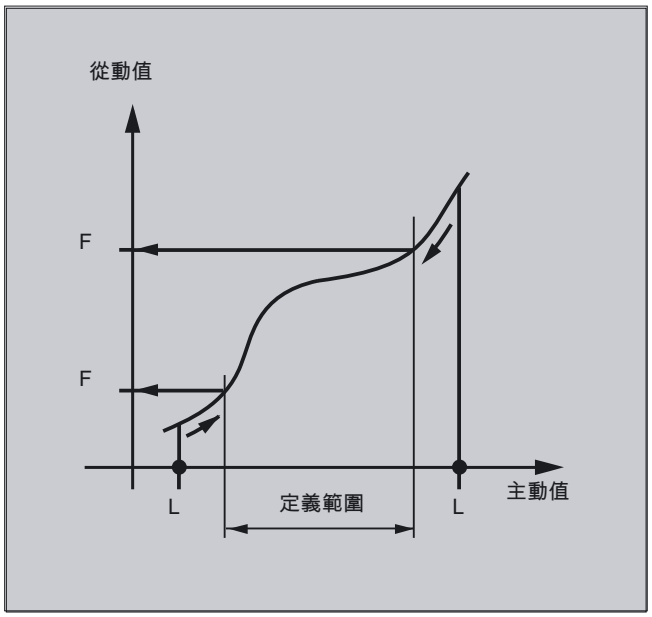
- 在執行時有足夠的 NC 電源可用

  或

- 在曲線表中的區段號碼，會在呼叫之前被查詢，因此如有必要，可再對相關的表進行子分割。

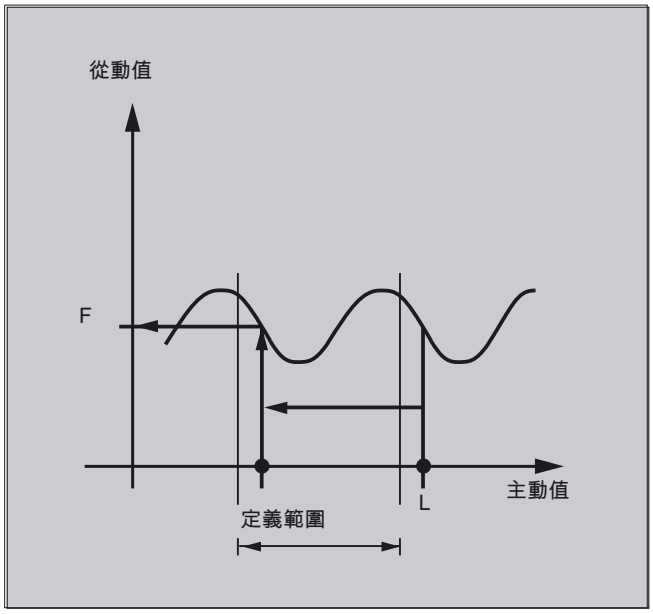
具有非週期性曲線表的 CTAB

若特定的<主動值>落在定義域之外，則上臨界值或下臨界值會被輸出為下列的值：



具有週期性曲線表的 CTAB

若特定的<主動值>超出定義範圍，主動值則將估算為定義範圍的模數，並輸入對應的跟隨值：



CTABINV 的大約值

CTABINV 指令，因此，需要一個大約的值來作為預測的主動值。CTABINV 將傳回最接近概估值的主動值。例如，大約的值可以是從前一個插補循環而來的主動值。

曲線表函數的斜度

斜度的輸出（<斜度>），使得計算先導軸或在對應位置的跟隨軸之速率這件事，變得可能。

先導軸或跟隨軸的規格

如果先導軸與跟隨軸的長度單位不同，則先導軸或跟隨軸選配規格就格外重要。

CTABSSV，CTABSEV

在下列情況中，指令 CTABSSV 與 CTABSEV **不適合**用來查詢已程式設計的區段：

- 已程式設計圓形或漸開線。
- 含 CHF/ RND 的倒角或倒圓角為啟用狀態
- 含 G643 的平滑為啟用狀態
- 含 COMPON/COMPCURV/COMPCAD 的 NC 單節壓縮為啟用狀態

9.2.7 曲線表：檢查資源的使用（CTABNO，CTABNOMEM，CTABFNO，CTABSEGID，CTABSEG，CTABFSEG，CTABMSEG，CTABPOLID，CTABPOL，CTABFPOL，CTABMPOL）

功能

程式設計者可使用這些指令，來取得關於曲線表，表區段以及多項式的資源最新使用資訊。

句法

```
CTABNO
CTABNOMEM（<記憶體位置>）
CTABFNO（<記憶體位置>）
CTABSEGID（<n>，<記憶體位置>）
CTABSEG（<記憶體位置>，<區段類型>）
CTABFSEG（<記憶體位置>，<區段類型>）
CTABMSEG（<記憶體位置>，<區段類型>）
CTABPOLID（<n>）
CTABPOL（<記憶體位置>）
CTABFPOL（<記憶體位置>）
CTABMPOL（<記憶體位置>）
```

意義

CTABNO	決定**定義的**曲線表號碼（在靜態與動態 NC 記憶體中）
CTABNOMEM	決定**定義的**曲線表號碼，在特定的<記憶體位置>
CTABFNO	決定**可能剩餘在**特定的<記憶體位置>中的曲線表號碼
CTABSEGID	決定特定的 <區段類型>的曲線區段之號碼，該區段類型被曲線表號碼<n>所用
CTABSEG	決定特定的<區段類型>，**用在**特定的<記憶體位置>中，其曲線區段的號碼
CTABFSEG	決定特定的<區段類型>，**可能剩餘在**特定的<記憶體位置>中，其曲線區段的號碼
CTABMSEG	決定曲線區段的**最大可能**號碼，其屬於特定的<區段類型>，在特定的<記憶體位置>中
CTABPOLID	決定被曲線表號碼<n>所用的曲線多項式之號碼
CTABPOL	決定**用在**特定的<記憶體位置>中，曲線多項式的號碼
CTABFPOL	決定**可能剩餘在**特定的<記憶體位置>中的曲線多項式號碼
CTABMPOL	決定曲線多項式的**最大可能**號碼，在特定的<記憶體位置>中
<n>	曲線表的編號（**ID**）
<記憶體位置>	指定記憶體位置（選用） “SRAM” **境態** NC 記憶體 “DRAM” **動態** NC 記憶體 **注意：** 若未替此參數程式設計值，會使用以 MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE 設定的預設記憶體位置。
<區段類型>	指定區段類型（選用） “L” 線性區段 “P” 多項式區段 **注意：** 若沒有對此參數程式設計值，線性與多項式區段的總和會輸出。

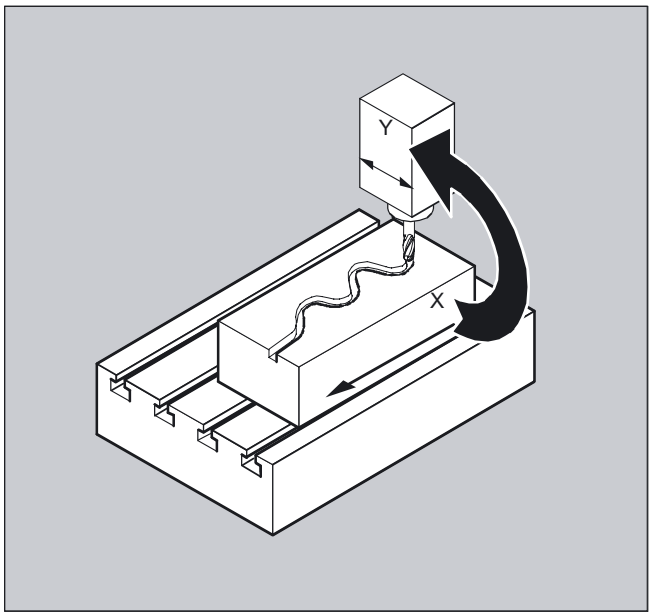
9.3 軸先導值耦合（LEADON，LEADOF）

說明

SINUMERIK 828D 並未提供此功能！

功能

利用軸主動值耦合，可同步移動先導軸與跟隨軸。 透過曲線表或是先導軸（如有必要亦可是模擬軸）某個位置上唯一計算得出的多項式，亦可指派跟隨軸位置。



先導軸是提供輸入值給曲線表的軸。 **跟隨軸**是利用曲線表接收算出之位置的軸。

實際值與設定點耦合

以下各項資料可作為主控值，亦即跟隨軸位置計算結果的輸出值：

- 先導軸位置實際值： 實際值耦合
- 先導軸位置設定點： 設定點值耦合

基本座標系統一律套用主控值耦合。

關於建立曲線表，請參閱“曲線表”一節。
關於主動值耦合，請參閱/FB/，M3，耦合軸動作與主動值耦合。

句法

```
LEADON（FAxis，LAxis，n）
LEADOF（FAxis，LAxis）
```

或不指定主導軸而停用

```
LEADOF（FAxis）
```

您可從工件程式以及從同步動作的移動期間，啟動或停用主動值耦合，請參閱“指示同步動作”一節。

意義

LEADON	啟動主動值耦合
LEADOF	停用主動值耦合
Faxis	跟隨軸
Laxis	主導軸
n	曲線表編號
$SA_LEAD_TYPE	切換設定點耦合與實際值耦合

停用主動值耦合，LEADOF

當您停用主動值耦合時，跟隨軸會再次成為一般指令軸!

軸主動值耦合與不同的操作狀態，RESET

根據機械參數中的設定，主動值耦合是以 RESET（重置）停用。

來自同步動作的主動值耦合範例

在衝壓廠裏，將傳輸軸與輔助軸所構成之傳輸系統的軸與先導軸（立軸）做一般機械性耦合，是為了讓電子耦合系統能夠取代之。

這示範了電子傳輸系統如何取代機械傳輸系統。 耦合與去耦流程是**靜態同步動作**。

從先導軸 LV（立軸）控制傳輸軸與輔助軸時，其方法等同於利用曲線表加以定義的跟隨軸。

跟隨軸

X 進給或主動值軸
YL 閉合軸或橫向軸
ZL 起重軸、
U 滾筒式進給、輔助軸
V 導頭、輔助軸
W 潤滑、輔助軸

動作

舉例來說，發生的動作包含下列同步動作：

- 啟動耦合，LEADON（跟隨軸、先導軸、曲線表編號）
- 停用耦合，LEADOF（跟隨軸，先導軸）
- 設定實際值，PRESETON（軸，值）
- 設定標記，$AC_MARKER[i]= value
- 耦合類型： 實際 / 虛擬主動值
- 逼近軸位置，POS[軸]=值

條件

快速數位輸入，利用布林運算子 AND 將時間變數$AC_MARKER 以及位置對照連結起來，作為評估條件。

說明

在以下範例中，換行、縮排以及使用**粗體**字型的唯一目的只是增進程式的可讀性。 對於控制系統而言，行號之後的所有文字便構成單行。

註解

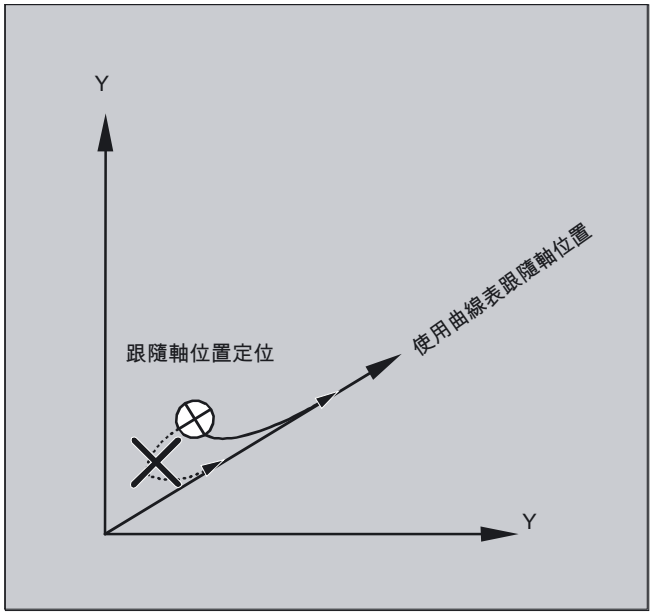
程式碼	註解
	; 定義所有靜態同步動作。
	; ****重置標記
N2 $AC_MARKER[0]=0 $AC_MARKER[1]=0 $AC_MARKER[2]=0 $AC_MARKER[3]=0 $AC_MARKER[4]=0 $AC_MARKER[5]=0 $AC_MARKER[6]=0 $AC_MARKER[7]=0	
	; **** E1 0=>1，傳輸 ON
N10 IDS=1 EVERY （$A_IN[1]==1） AND （$A_IN[16]==1） AND （$AC_MARKER[0]==0） DO LEADON（X，LW，1） LEADON（YL，LW，2） LEADON（ZL，LW，3） $AC_MARKER[0]=1	
	; **** E1 0=>1，耦合滾輪進給 ON
N20 IDS=11 EVERY （$A_IN[1]==1） AND （$A_IN[5]==0） AND （$AC_MARKER[5]==0） DO LEADON（U，LW，4） PRESETON（U，0） $AC_MARKER[5]=1	
	; **** E1 0->1，耦合對齊準頭 ON
N21 IDS=12 EVERY （$A_IN[1]==1） AND （$A_IN[5]==0） AND （$AC_MARKER[6]==0） DO LEADON（V，LW，4） PRESETON（V，0） $AC_MARKER[6]=1	
	; **** E1 0->1，潤滑耦合 ON
N22 IDS=13 EVERY （$A_IN[1]==1） AND （$A_IN[5]==0） AND （$AC_MARKER[7]==0） DO LEADON（W，LW，4） PRESETON（W，0） $AC_MARKER[7]=1	
	; **** E2 0=>1，耦合 OFF
N30 IDS=3 EVERY （$A_IN[2]==1） DO LEADOF（X，LW） LEADOF（YL，LW） LEADOF（ZL，LW） LEADOF（U，LW） LEADOF（V，LW） LEADOF（W，LW） $AC_MARKER[0]=0 $AC_MARKER[1]=0 $AC_MARKER[3]=0 $AC_MARKER[4]=0 $AC_MARKER[5]=0 $AC_MARKER[6]=0 $AC_MARKER[7]=0	
....	
N110 G04 F01	
N120 M30	

說明

主動值耦合要求先導軸與跟隨軸必須同步。啟動主動值耦合時，跟隨軸必須在根據曲線表所計算得出的曲線定義允差範圍內，唯有如此才能達到上述同步要求。

跟隨軸位置的允差範圍是利用機械參數 MD 37200 加以定義：COUPLE_POS_POL_COARSE A_LEAD_TYPE.

啟動主動值耦合時，若跟隨軸尚未在正確位置上，則等待所計算得出的跟隨軸位置設定點值大約等於真正跟隨軸位置時，將隨即自動開始同步執行。跟隨軸將於同步程序期間以該軸設定點速度所定義的方向移動（此速度是利用 CTAB 曲線表從主控主軸計算得出）。

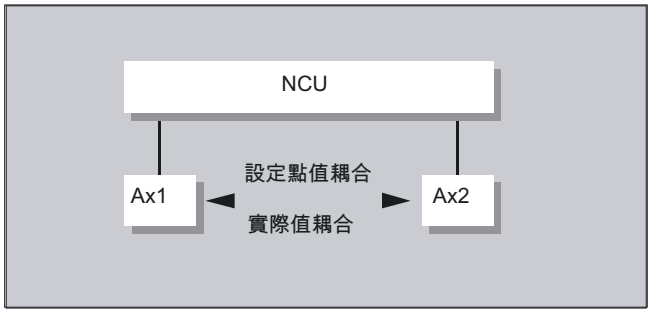


未同步

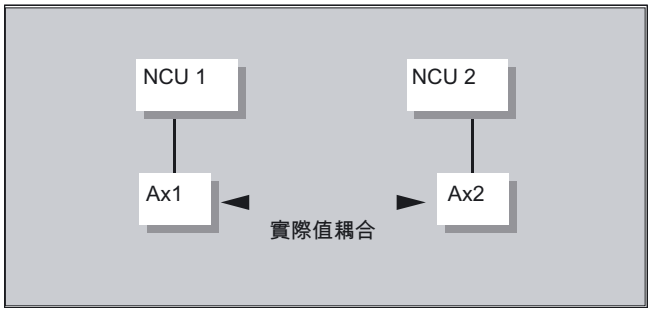
如果啟動主動值耦合時，所計算得出的跟隨軸位置正在遠離目前跟隨軸位置，那麼將無法執行同步。

實際值與設定點耦合

與實際值耦合相較，設定點耦合能夠提供更佳的“先導軸與跟隨軸同步”功能，因此為預設選項。



先導軸與跟隨軸必須由同一 NCU 插補，如此才能進行設定點耦合。如果使用外部先導軸，則跟隨軸僅能以實際值與先導軸耦合。



切換功能可透過設定資料$SA_LEAD_TYPE 進行程式設計
跟隨軸停止時，您永遠必須在實際值耦合與設定點耦合之間進行切換。當軸靜止不動時，您必須進行切換，然後才能重新同步。

應用程式

機台嚴重震動且無發生錯誤時，您無法讀取實際值。如果您在壓製傳輸時使用主動值耦合，可能必須在最大震動的加工步驟中，將實際值耦合切換成設定點耦合。

使用設定點耦合時的**主控值模擬**
利用機械參數，您可從伺服中斷插補器與先導軸的連接。如此一來，您不需實際移動先導軸，就能產生設定點耦合所需的設定點。
您可從下列變數讀取由設定點連結產生的主動值，然後便可在，例如，同步動作中使用該值：

- $AA_LEAD_P	主動值位置
- $AA_LEAD_V	主動值速率

建立主控值

您亦可選擇使用其他自程方式產生主動值。以此方式產生的主動值是利用變數加以讀寫

- $AA_LEAD_SP	主動值位置
- $AA_LEAD_SV	主動值速率

您必須先將設定資料設定為$SA_LEAD_TYPE = 2，然後才能開始使用這些變數。

耦合狀態

您可利用下列系變數在 NC 程式中查詢耦合狀態：

$AA_COUP_ACT[[axis]]
0：無耦合生效
16：主動值耦合為啟用中

同步動作的狀態管理

切換事件與耦合事件是利用時間變數加以管理：

$AC_MARKER[i] = n
是以
i 旗標編號與
n 狀態值加以管理

9.4 電子齒輪（EG）

功能

“電子齒輪”函數可讓您根據線性移動單節，控制**跟隨軸**的移動，一個函數可輸入高達五個**先導軸**。各個先導軸與跟隨軸的關係是以耦合係數加以定義。

將個別先導軸動作部分乘以各自的耦合係數後再加總，即可得出跟隨軸的動作部分。啟動 EG 軸群組時，您可將已定義之位置的相關跟隨軸同步化。所謂齒輪群組可以是：

- 已定義的、
- 已啟動的、
- 已停用的、
- 刪除。

跟隨軸動作可選擇性取自

- 先導軸的設定點，以及
- 先導軸的實際值。

利用**曲線表**，各先導軸與跟隨軸之間的非線性關係亦可認知為一項延伸性資料（請參閱“路徑移動行為”一節）。電子齒輪可以是串聯式架構，意即某一電子齒輪的跟隨軸可以是另一電子齒輪的先導軸。

9.4.1 定義電子齒輪（EGDEF）

功能

透過指定跟隨軸以及一至五個先導軸（包括各自相關的耦合類型），即可定義 EG 軸群組。

需求

定義 EG 軸群組的先決條件：

不得定義跟隨軸的軸耦合（或：若有軸耦合則必須先以 EGDEL 刪除之）。

句法

EGDEF（跟隨軸、先導軸 1、耦合類型 1、先導軸 2、耦合類型 2、...）

意義

EGDEF	定義電子齒輪
跟隨軸	受先導軸影響的軸
先導軸 1 , ...,	影響跟隨軸的軸
先導軸 5	
耦合類型 1 , ...,	耦合類型 先導軸的耦合類型不需完全一致，但個別主控值必須各自單獨加以程式設計
耦合類型 5	**值：** **意義：** 0 跟隨軸受到對應先導軸之**實際值**的影響。 1 跟隨軸受到對應先導軸之**設定點**的影響。

說明

若已定義 EG 軸群組，耦合係數預設為零。

說明

EGDEF 會觸發前置處理停止。如果系統中有一或多個先導軸透過**曲線表**影響跟隨軸，含 EGDEF 之變速箱定義在使用時亦須保持一致。

範例

程式碼	註解
EGDEF（C，B，1，Z，1，Y，1）	; 定義 EG 軸群組。先導軸 B，Z，Y 透過設定點影響跟隨軸 C。

9.4.2 切換至電子變速箱（EGON、EGONSYN、EGONSYNE）

功能

切換至 EG 軸群組的方法共有三種。

句法

格式變異 1：

選擇以未同步化方式切換至 EG 軸群組時，請使用：
`EGON（FA，“單節變更模式”，LA1，Z1，N1，LA2，Z2，N2，...，LA5，Z5，N5）`

格式變異 2：

選擇以同步化方式啟動 EG 軸群組時，請使用：
`EGONSYN（FA，“單節變更模式”，SynPosFA，[，LAi，SynPosLAi，Zi，Ni]）`

格式變異 3：

選擇以同步化方式採逼近模式切換至 EG 軸群組時，請使用：
`EGONSYNE（FA，“單節變更模式”，SynPosFA，逼近模式，[，LAi，SynPosLAi，Zi，Ni]）`

意義

情形 1：

`FA`	跟隨軸
單節變更模式	您可使用下列模式： `"NOC"` 單節立刻變化 `"FINE"` 單節變更是以“微調同步”之方式進行 `"COARSE"` 單節變更是以“粗調同步”之方式進行 `"IPOSTOP"` 依設定點同步執行單節變更
`LA1，... LA5`	先導軸
`Z1，... Z5`	耦合係數 i 的計數
`N1，... N5`	耦合係數 i 的分母
	耦合係數 i = 計數 i / 分母 i

先導軸必須事先以 `EGDEF` 指令加以指定，如此才能在啟用列上進行程式設計。您必須程式設計至少一個先導軸。

情形 2：

FA	跟隨軸
單節變更模式	您可使用下列模式： "NOC" 單節立刻變化 "FINE" 單節變更是以“微調同步”之方式進行 "COARSE" 單節變更是以“粗調同步”之方式進行 "IPOSTOP" 依設定點同步執行單節變更
[, LAi, SynPosLAi, Zi, Ni]	（請勿寫入方括弧） 以下各項可寫上至少一個最多五個：
LA1, ... LA5	先導軸
SynPosLAi	第 i 先導軸的同步位置
Z1, ... Z5	耦合係數 i 的分子
N1, ... N5	耦合係數 i 的分母
	耦合係數 i = 分子 i / 分母 i

先導軸必須事先以 EGDEF 指令加以指定，如此才能在啟用列上進行程式設計。透過跟隨軸（SynPosFA）與先導軸（SynPosLA）的已程式設計“同步位置”，其軸群組已轉譯成*同步*”的位置可完成定義。切換至群組時，若電子齒輪不在同步狀態，則跟隨軸將移動至其已定義的同步位置。

情形 3：

參數對應至情形 2 plus 的參數：

進刀模式	您可使用下列模式： "NTGT" 逼近下一個時間最佳化的齒槽 "NTGP" 逼近下一個路徑最佳化的齒槽 "ACN" 沿負方向移動旋轉軸（絕對） "ACP" 沿正方向移動旋轉軸（絕對） "DCT" 對於已程式設計之同步位置為時間最佳化的逼近方式 "DCP" 對於已程式設計之同步位置為距離最佳化的逼近方式

格式變數 3 將僅影響已與“模數先導軸”耦合的“模數跟隨軸”。所謂時間最佳化將會考量跟隨軸的速限。

其它資訊

切換版本說明

情形 1：

切換至群組的瞬間，先導軸與跟隨軸的位置便儲存為“同步位置”。您可利用系統變數 $AA_EG_SYN 讀取“同步位置”。

情形 2：

如果耦合群組中包含模數軸，則其位置值為減折模數。如此確保系統是逼近下一個可能的同步位置（這就是所謂的“*相對同步*”：例如下一個齒槽）。如果系統向跟隨軸發佈“啟用跟隨軸手動超調”介面訊號 DB（30＋軸編號），DBX 26 位元 4，則將僅逼近同步位置。如果未發佈此訊號，程式將停止在 EGONSYN 單節上，並等待上述訊號完成後才輸出警報 16771，該警報將自行清除。

情形 3：

齒距（角度）的計算方式如下：360 * Zi/Ni.如果跟隨軸在進行呼叫時停止動作，則路徑最佳化傳回的內容將與時間最佳化完全一致。

如果跟隨軸已在動作中，NTGP 將不可考慮跟隨軸目前速率，逕自在下一個齒槽上進行同步化。如果跟隨軸已在動作中，NTGP 將根據跟隨軸目前速率，在下一個齒槽上進行同步化。軸亦會視需要減速。

曲線表

如果**曲線表**是用於其中一個先導軸：

Ni	線性耦合的耦合係數分母必須設為 0。（分母為 0 不合線性耦合之規定。）分子為零等於告訴控制系統
Zi	是應使用的曲線表編號。指定編號的曲線表在電源開啟時就應已定義。
LAi	指定的先導軸利用耦合係數對應至耦合指定先導軸（線性耦合）。

關於使用曲線表以及串聯並同步化電子齒輪的詳細資訊，請參閱：

參考：
功能手冊，特殊功能；耦合軸與 ESR（M3），“耦合動作與先導值耦合”。

電源開啟時、重置時、模式變更時以及單節搜尋時的電子齒輪回應

- 電源開啟後**無**啟用的耦合。
- 耦合的啟用狀態不受重置或操作模式切換的影響。
- 單節搜尋期間，將不會執行或囤集有關切換、刪除與定義電子齒輪等的指令，系統將直接略過。

電子齒輪的系統變數

藉由電子齒輪的系統變數，工件程式可判定 EG 軸群組的目前狀態並視需要採取相關措施。

電子變速箱系統變數的名稱標示如下所示：

$AA_EG_ ...

或

$VA_EG_ ...

參考：
系統變數手冊

9.4.3 切換至電子變速箱（EGOFS，EGOFC）

功能

切換出啟用之 EG 軸群組的方法共有三種。

程式設計

情形 1：

句法	說明
EGOFS（跟隨軸）	停用電子齒輪。跟隨軸開始剎車直到靜止。此項呼叫將觸發前置處理停止。

情形 2：

句法	說明
EGOFS（跟隨軸，先導軸 1，…，先導軸 5）	這樣的指令參數設定可讓您**選擇性**移除個別先導軸對於跟隨軸動作的影響。

您必須指定至少一個先導軸。選擇性禁止指定之先導軸影響從動裝置。此項呼叫將觸發前置處理停止。如果此項呼叫仍包含啟用之先導軸，則從動裝置將在受到其影響下繼續操作。如果這個方法排除了所有先導軸的影響，則跟隨軸開始剎車直到靜止。

情形 3：

句法	說明
EGOFC（跟隨主軸 1）	停用電子齒輪。跟隨主軸將以停用瞬間當時所套用的速度 / 速率繼續移動。此項呼叫將觸發前置處理停止。

說明

此情形僅適用於主軸。

9.4.4 刪除電子齒輪的定義（EGDEL）

功能

您必須切換出 EG 軸群組之後，才能刪除其定義。

程式設計

句法	意義
EGDEL（跟隨軸）	刪除軸群組的耦合定義。在達到最大同時啟動之軸群組數量以前，您可藉由 EGDEF 定義其他軸群組。此項呼叫將觸發前置處理停止。

9.4.5 旋轉進給率（G95） / 電子齒輪（FPR）

功能

FPR 指令可用來指定，某一電子齒輪的跟隨軸決定繞轉進給率。請注意以下這個指令的相關事項：

- 將由電子齒輪跟隨軸的設定點速率決定進給率。
- 而設定點速率則是從（非路徑軸之）先導主軸與模數軸的速度及其相關耦合係數計算得出。
- 將不考慮線性先導軸或非模數先導軸的速度部分以及跟隨軸被覆蓋的動作。

9.5 同步主軸

功能

同步操作涉及一個跟隨主軸（FS）以及一個先導主軸（LS），它們就合稱為“**同步主軸組**”。當耦合為啟用狀態時（同步操作），跟隨主軸將依照已定義功能上的相互關係，仿效先導主軸移動。

您可利用通道專屬機械參數指派固定設定給每個機台的同步主軸組，或者透過 CNC 工件程式，針對特定應用定義之。每條 NC 通道上最多可同時操作兩個同步主軸組。

關於下列耦合動作，請參閱工件程式

- 從工件程式定義或變更
- 啟動
- 停用
- 刪除

耦合動作。

此外，根據軟體狀態

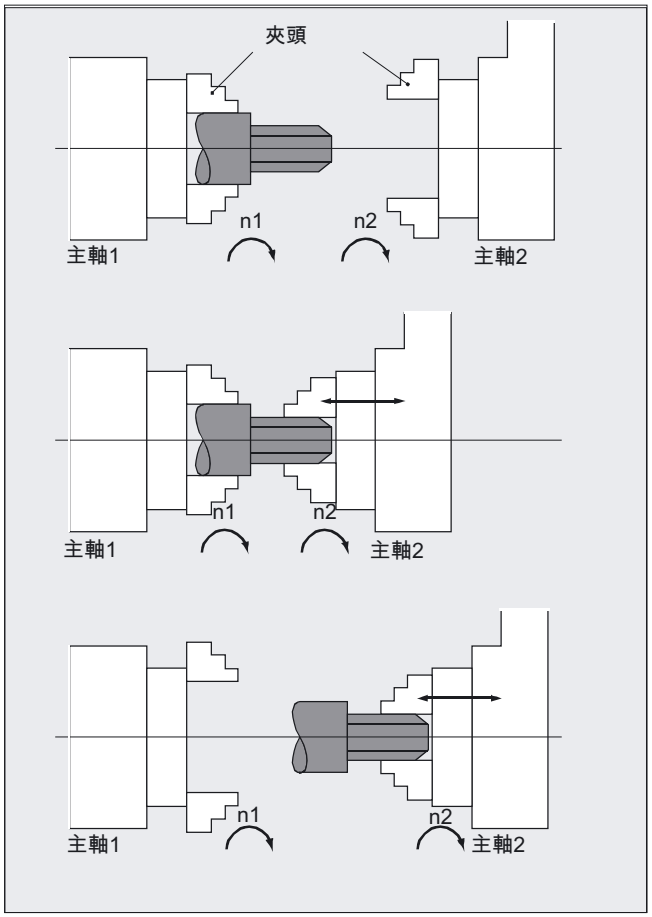
- 進入同步狀態之前可能需要等待一段時間
- 單節變更方式可能有所改變
- 可能已選擇設定點耦合類型或實際值耦合類型，或者已指定主控主軸與跟隨主軸之間的角度偏移
- 啟動耦合時，可能將傳輸跟隨軸的上一個程式設計
- 可能修正已測得或已知的同步差異

9.5.1 同步主軸：程式設計（COUPDEF、COUPDEL、COUPON、COUPONC、COUPOF、COUPOFS、COUPRES、WAITC）

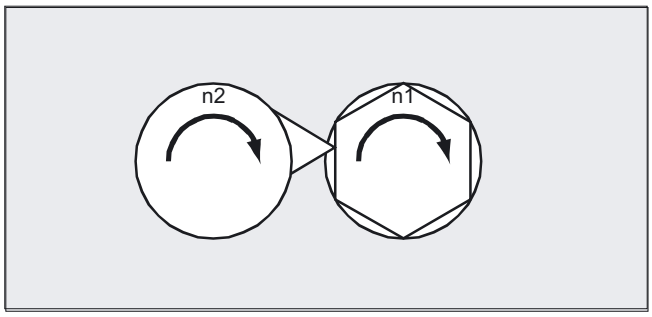
功能

同步主軸函數增進了兩個主軸的同步移動（跟隨軸 FS 與先導軸 LS），例如，即時工件轉換。此函數可支援下列模式：

- 速度同步（$n_{FS} = n_{LS}$）
- 位置同步（$\phi_{FS} = \phi_{LS}$）
- 含角度偏移之位置同步（$\phi_{FS} = \phi_{LS} + \Delta\phi$）



介於先導和後續軸間的轉換比例不等於 1，可藉由此轉換比例的規格，來支援多邊加工。



句法

```
COUPDEF（FS、LS、TFS、TLS、單節變更，耦合類型）
COUPON（FS，LS，POSFS）
COUPONC（FS，LS）
COUPOF（FS，LS，POSFS，POSLS）
COUPOFS（FS，LS）
COUPOFS（FS，LS，POSFS）
COUPRES（FS，LS）
COUPDEL（FS，LS）
WAITC（FS、單節變更、LS、單節變更）
```

說明

簡化標記

COUPOF，COUPOFS，COUPRES，與 COUPDEL 敘述，支援不具有先導軸規格的簡化標記。

意義

COUPDEF	以使用者自訂基準來定義/更改耦合
COUPON	啟動耦合。跟隨主軸與主要主軸將依照目前速度同步化
COUPONC	以 M3 S...或 M4 S...的上一個程式設計啟動時，傳輸耦合。 立即傳輸跟隨主軸的速度差異。
COUPOF	停用耦合。 • 以立即單節更改： COUPOF（S2，S1） • 單節更改僅在 POSFS 或 POSLS 停用位置已被跨過之後： COUPOF（S2，S1，POSFS） COUPOF（S2，S1，POSFS，POSLS）
COUPOFS	跟隨主軸停止時停用耦合。 以即時單節變更之方式儘快進行單節變更： COUPOFS（S2，S1） 僅於傳遞關閉位置後進行單節變更： COUPOFS（S2，S1，POSFS）
COUPRES	將耦合參數重置為已設定之 MD 與 SD
COUPDEL	刪除使用者自訂之耦合

WAITC	等待同步執行狀態 （單節變更期間 NOC 增加至 IPO）
FS	跟隨主軸指定

選配參數：

LS	主要主軸指定 含主軸編號之規格：例如 S2、S1
ÜFS，ÜLS	介於 FS 和 LS 之間的轉換比率。 TFS =分子，TLS =分母 預設值：TFS/TLS = 1.0；分母選項的規格
單節變化	單節變更行為 單節變更是： "NOC" 立即 "FINE" 到達"同步微調" "COARSE" 到達"同步粗調" "IPOSTOP" 到達 IPOSTOP；換句話說，在設定點基準同步之後（預設） 單節變更行為為模態有效。
耦合類型	耦合類型：FS 與 LS 間的耦合 "DV" 設定點連接（預設） "AV" 實際值耦合 "VV" 速度耦合 耦合類型為模態。
POSFS	先導主軸與跟隨主軸間的角度偏移 值域： 0
POSFS，POSLS	跟隨主軸與先導主軸的關閉位置 “待傳遞 POS_{FS}、POS_{LS} 之後立即啟用單節變更” 值域： 0

範例

範例 1：使用先導主軸與跟隨主軸進行加工

程式設計	註解
	; 先導主軸 = 主控主軸 = 主軸 1
	; 跟隨主軸 = 主軸 2
N05 M3 S3000 M2=4 S2=500	; 先導主軸以 3，000 rpm 旋轉，
	; 跟隨主軸以 500 rpm 旋轉
N10 COUPDEF（S2，S1，1，1，"NOC"，"Dv"）	; 耦合定義（亦可加以設定）
...	
N70 SPCON	; 將先導主軸帶入閉迴圈位置控制（設定點耦合）
N75 SPCON（2）	; 將跟隨主軸帶入閉迴圈位置控制
N80 COUPON（S2，S1，45）	; 即時耦合至偏移位置 = 45 度
...	
N200 FA[S2]=100	; 定位速度 = 100 度 / 分鐘
N205 SPOS[2]=IC（-90）	; 沿負方向以 90° 覆蓋之方式移動
N210 WAITC（S2，"Fine"）	; 等待“微調”同步
N212 G1 X... Y... F...	; 加工
...	
N215 SPOS[2]=IC（180）	; 沿正方向以 180° 覆蓋之方式移動
N220 G4 S50	; 主控主軸停頓時間 = 50 轉數
N225 FA[S2]=0	; 啟動已設定之速率（MD）
N230 SPOS[2]=IC（-7200）	; 以設定速度的 20 轉數，在
	; 負向
...	
N350 COUPOF（S2，S1）	; 即時退耦，S=S2=3000
N355 SPOSA[2]=0	; 在 0 度上停止 FS
N360 G0 X0 Y0	;
N365 WAITS（2）	; 等待主軸 2
N370 M5	; 停止 FS
N375 M30	

範例 2：程式設計速度差異

程式規劃	註解
	; 先導主軸 = 主控主軸 = 主軸 1
	; 跟隨主軸 = 主軸 2
N01 M3 S500	; 先導主軸以 500 rpm 旋轉
N02 M2=3 S2=300	; 跟隨主軸以 300 rpm 旋轉
...	
N10 G4 F1	; 主控主軸停頓時間
N15 COUPDEF （S2，S1，-1）	; 耦合係數比為-1：1
N20 COUPON（S2，S1）	; 啟動耦合。下列主軸的速度已決定
	; 藉由先導軸與耦合係數的速度。
...	
N26 M2=3 S2=100	; 程式設計速度差異

範例 3：速度不同時的移動傳輸範例

1. 在上一個以 COUPON 進行跟隨主軸程式設計期間，啟動耦合

程式規劃	註解
	; 先導主軸 = 主控主軸 = 主軸 1
	; 跟隨主軸 = 主軸 2
N05 M3 S100 M2=3 S2=200	; 先導主軸以 100 rpm 旋轉，跟隨主軸以 200 rpm 旋轉
N10 G4 F5	; 主控主軸停頓時間 = 5 秒
N15 COUPDEF（S2，S1，1）	; FS 至 LS 的轉換比率為 1.0（預設）
N20 COUPON（S2，S1）	; 即時耦合至先導主軸
N10 G4 F5	; 跟隨主軸以 100 rpm 旋轉

2. 在上一個以 COUPONC 進行跟隨主軸程式設計期間，啟動耦合

程式規劃	註解
	; 先導主軸 = 主控主軸 = 主軸 1
	; 跟隨主軸 = 主軸 2
N05 M3 S100 M2=3 S2=200	; 先導主軸以 100 rpm 旋轉，跟隨主軸以 200 rpm 旋轉
N10 G4 F5	; 主控主軸停頓時間 = 5 秒
N15 COUPDEF（S2，S1，1）	; FS 至 LS 的轉換為 1.0（預設）
N20 COUPONC（S2，S1）	; 即時耦合至先導主軸並傳輸
	; 之前至 S2 的速度
N10 G4 F5	; S2 以 100 rpm + 200 rpm = 300 rpm 之速度旋轉

3. 於跟隨主軸靜止的情況下利用 COUPON 啟動耦合

程式規劃	註解
	; 先導主軸 = 主控主軸 = 主軸 1
	; 跟隨主軸 = 主軸 2
N05 SPOS=10 SPOS[2]=20	; 跟隨主軸 S2 處於定位模式下
N15 COUPDEF（S2，S1，1）	; FS 至 LS 的轉換為 1.0（預設）
N20 COUPON（S2，S1）	; 即時耦合至先導主軸
N10 G4 F1	; 關閉耦合，S2 停止在 20 度

4. 於跟隨主軸靜止的情況下利用 COUPONC 啟動耦合

說明

定位模式或軸模式

如果耦合前跟隨主軸處於定位模式或軸模式，則不論是使用 COUPON（FS，LS）或是 COUPONC（FS，LS），跟隨主軸的動作皆相同。

注意

前導主軸與軸操作

若在定義耦合之前，前導主軸在軸操作中，從機械參數而來的速率臨界值

MD32000 $MA_MAX_AX_VELO（最大軸速率）將仍然使用，即使在啟用了耦合之後。

為了避免此行為，在定義耦合之前，必須將軸切換到主軸模式（M3 S...或 M4 S...）。

定義同步主軸組

設定耦合：

對於已設定的耦合，前導與跟隨主軸在機械參數中定義。已設定的主軸，無法在工件程式中更改。使用 COUPDEF 可以在工件程式中對耦合參數化（這種情況下，寫入保護皆為無效）。

使用者自訂之耦合

COUPDEF 可備用來定義或更改工件程式中的耦合。若已啟用耦合，則必須在定義新耦合之前，先以 COUPDEL 來刪除。

定義耦合 COUPDEF

於整體定義耦合：

COUPDEF（FS，LS，T_{FS}，T_{LS}，單節更改行為，耦合類型）

跟隨主軸（FS）與前導主軸（LS）

用於下列主軸（FS）與前導主軸（LS）的軸名稱，會備用來定義各自的耦合。必須以各個 COUPDEF 操作，來對軸名稱進行程式設計。.其他耦合參數為模態，且必須被程式設計，若其變更。

範例：

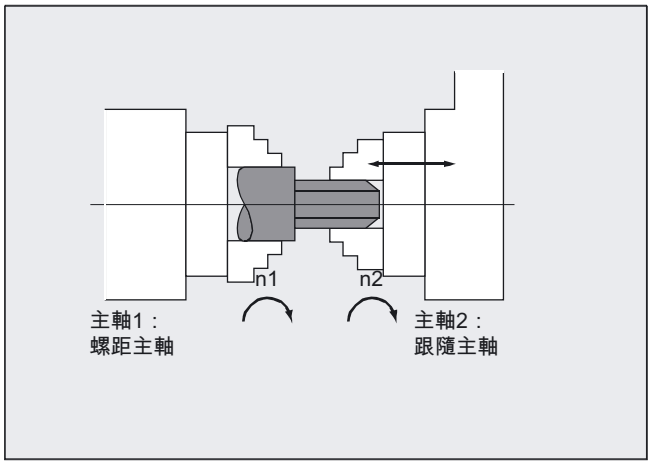
```
COUPDEF（S2，S1）
```

轉換比率 T_{FS}/T_{LS}

轉換比率被指定為介於跟隨主軸（分子）和前導主軸（分母）之間的速度比率。您必須程式設計分子。若缺少了程式設計值，則將分母設定為= 1.0。

範例：跟隨主軸 S2 與前導主軸 S1，轉換比率= 1/4 = 0.25。

```
COUPDEF （S2，S1，1.0，4.0）
```



說明

可立即更改轉換比率（當耦合生效且主軸旋轉中）。

單節變更行為 NOC、FINE、COARSE、IPOSTOP

下列的簡化標記可在程式設計單節更改行為時使用：

- 否：立即（預設）
- FI：到達"同步微調"
- CO：到達"同步粗調"
- IP：到達 IPOSTOP；換句話說，在設定點基準同步之後

耦合類型 DV、AV

⚠小心

唯有當耦合停用時，才可變更耦合類型！

啟動同步模式 COUPON、POSFS

- 可利用 LS 與 FS 間的任一角度參考啟動耦合
  - COUPON（S2，S1）
  - COUPON（S2，S1，POSFS）
  - COUPON（S2）
- 以有角度的偏移 POSFS 來啟用耦合

  用於異型工件的位置同步耦合。
  POSFS 在正旋轉方向時參照前導主軸之 0°位置
  POSFS 值域：0°... 359，999°：
  - COUPON（S2，S1，30）

  利用這個方法，即使是耦合已啟用時亦可變更角度偏移。

定位跟隨主軸

當同步主軸耦合為啟用狀態時，即使是由主控主軸開始動作，但您仍可在±180°的範圍內定位跟隨主軸。

定位 SPOS

您可利用 SPOS=…補插跟隨主軸。

範例：

```
SPOS[2] = IC（-90）
```

有關 SPOS 的詳細資訊出現在：

參考
程式設計手冊，基礎。

速度差異 M3 S...或 M4 S...

會有速度差異是因為兩個帶正負號的速度來源重疊，將於同步主軸耦合為啟用狀態的期間重新進行程式設計，以供速度控制模式的跟隨主軸使用，例如其中 Sn=...或 Mn=3，Mn=4。處理期間，將利用耦合係數從主要主軸取得此速度元件，然後再將跟隨主軸連同正確的前置字元一起加入至該元件中。

說明

當旋轉方向為 M3 或 M4 時，速度 S...亦必須重新進行程式設計，否則將會觸發警報向使用者回報程式設計內容遺失。

如需速度差異的詳細資訊，請參閱

參考：/FB2/ 功能手冊，延伸功能；同步主軸（S3）。

COUPONC 的速度差異

速度不同時的移動傳輸

當以 COUPONC 啟用了同步主軸耦合，目前的生效速度會覆蓋在跟隨軸上（M3S... or M4S...）。

說明

啟用重疊

使用 COUPONC （M3S...或 M4S...）來重疊主軸速度，需要在其生效前啟用。

前導主軸的動態限制

當套用覆蓋至跟隨主軸時，前導主軸的動態回應必須被限制在特定範圍，且沒有超過其動態臨界值。

速率，加速度：FA、ACC、OVRA、VELOLIMA

軸速率與跟隨主軸的加速度，可用下列來程式設計：

- FA[SPI（Sn）]與 FA[Sn]（軸速率）
- ACC[SPI（Sn）]與 ACC[Sn]（軸加速度）
- OVRA[SPI（Sn）]與 OVRA[Sn]（軸的手動倍率）
- VELOLIMA[SPI（Sn）]與 VELOLIMA[Sn]（軸的速率各自增加與減少）

Where n = 1，2，3，等等（跟隨主軸的主軸號碼）

參考
程式設計手冊，基礎

說明

加速度元件 JERKLIMA[Sn]

程式設計一個軸速率的增加或減少，目前對主軸無效。

有關設定軸動態回應的進一步資訊，請參考：

參考：
/FB2/功能手冊，延伸功能；旋轉軸（R2）。

可程式設計單節變更行為 WAITC

WAITC 可用來定義單節更改行為，例如，在以數個不同的同步條件（粗調，微調，IPOSTOP），更改了為耦合參數或位置動作之後。若沒有指定同步條件，則將套用在 COUPDEF 定義中的單節更改行為。

範例：

根據要被滿足的 COUPDEF，等待同步條件

```
WAITC（ ）
```

等待要為跟隨軸 S2 滿足的同步條件 FINE，以及要為跟隨軸 S4 滿足的 COARSE：

```
WAITC（S2，"FINE"，S4，"COARSE"）
```

停用耦合 COUPOF

COUPOF 可被用來定義耦合的關閉行為：

- 以利及的單節更改來停用耦合：
  - COUPOF（S2，S1）（具有前導主軸的規格）
  - COUPOF（S2）（沒有前導主軸的規格）
- 在已跨過關閉位置之後，停用耦合。在已跨過關閉位置之後，發生了單節更改。
  - COUPOF（S2，S1，150）（關閉位置 FS：150°)
  - COUPOF（S2，S1，150，30）（開關位置 FS：150°，LS：30°)

以下列主軸停止 COUPOFS 來停用耦合

COUPOFS 可用來定義，以跟隨主軸停止所進行的耦合之關閉行為：

- 以跟隨主軸停止與立即單節更改所進行的停用耦合：
  - COUPOFS（S2，S1）（具有前導主軸的規格）
  - COUPOFS（S2）（沒有前導主軸的規格）
- 在關閉位置已被跟隨主軸停止跨過之後，停用耦合。在已跨過關閉位置之後，發生了單節更改。
  - COUPOFS（S2，S1，150）（關閉位置 FS：150°)

刪除耦合 COUPDEL

COUPDEL 刪除耦合：

- COUPDEL（S2，S1）（具有前導主軸的規格）
- COUPDEL（S2）（沒有前導主軸的規格）

重置耦合參數，COUPRES

COUPRES 啟動了在機械參數和設定資料中參數化的耦合值：

- COUPRES（S2，S1）（具有前導主軸的規格）
- COUPRES（S2）（沒有前導主軸的規格）

系統變數

跟隨主軸的目前耦合狀態

目前跟隨主軸的耦合狀態，可使用跟隨系統變數來讀取：

$AA_COUP_ACT[FS]

值	意義
0	無生效之耦合
4	同步主軸耦合為生效狀態
注意 其他系統變數值參考到軸操作。 **參考** /LGA1/系統變數手冊	

目前角度偏移

與前導主軸相關的，跟隨主軸目前的有角度偏移，可使用跟隨系統變數來讀取：

- $AA_COUP_OFFS[FS]（設定點有角度偏移）
- $VA_COUP_OFFS[FS]（實際值有角度偏移）

說明

如果在耦合為啟用狀態的跟進模式下停用控制器後再重新啟用，那麼，控制器重新啟用時位置偏移將不同於原本程式設計的數值。在此情況下，您可在工件程式中讀取新的位置偏移並視需要加以修正。

9.6 主控 / 從屬群組（MASLDEF、MASLDEL、MASLON、MASLOF、MASLOFS）

功能

在過去的 SW 6.4 及其之前版本中，主控 / 附屬耦合要求必須在所有相關軸皆停止的情況下才允許附屬軸耦合至其主控軸。

SW 6.5 延伸版本則允許**旋轉**、速控主軸與動態設定的耦合及退耦。

句法

```
MASLON（Slv1，Slv2，...，）
MASLOF（Slv1，Slv2，...，）
MASLDEF（Slv1、Slv2、...、主控軸）        動態設定的延伸版本
MASLDEL（Slv1，Slv2，...，）              動態設定延伸的延伸版本
MASLOFS（Slv1，Slv2，...，）              附屬主軸延伸版本
```

說明

使用 MASLOF/MASLOFS 時，不需隱含前置處理停止。由於少了前置處理停止，在下一個程式設計之前，附屬軸的$P 系統變數不會提供更新後的數值。

意義

一般

MASLON	關閉（切換至）臨時耦合。
MASLOF	開啟一個有效耦合。必須觀察主軸延伸規格。

動態設定延伸版本

MASLDEF	使用機械參數之使用者自訂耦合或亦可從工件程式建立 / 變更。
MASLOFS	開啟 MASLOF 的耦合類比，而且附屬主軸開始自動剎車。
MASLDEL	退耦主控 / 附屬軸群組並刪除群組定義。
Slv1，Slv2，...	由主控軸帶動之附屬軸。
主控軸	在主控 / 附屬群組中負責控制已定義之附屬軸的軸。

範例

範例 1：主控 / 附屬耦合的動態設定

從工件程式進行主控 / 附屬耦合的動態設定：
軸容器旋轉之後相關軸必須成為主控軸。

程式碼	註解
MASLDEF（AUX，S3）	; AUX 由 S3 主控
MASLON（AUX）	; 關閉 AUX 耦合
M3=3 S3=4000	; 順時針旋轉方向
MASLDEL（AUX）	; 刪除設定並開啟耦合
AXCTSWE（CT1）	; 容器旋轉

範例

範例 2：附屬軸的實際值耦合

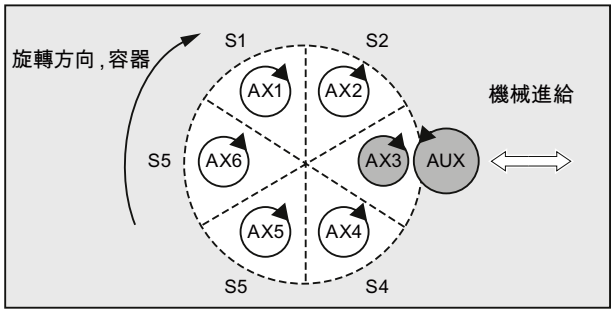
利用 PRESETON 以實際值耦合之方式將附屬軸與主控軸相同值耦合。
進行永久主控 / 附屬耦合時，附屬軸端的實際值需以 PRESETON 進行變更。

程式碼	註解
N37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[AX2]=0	; 短暫切換出永久耦合。
N37263 NEWCONF	
N37264 STOPRE	
MASLOF（Y1）	; 臨時耦合開啟。
N5 PRESETON（Y1，0，Z1，0，B1，0，C1，0，U1，0）	; 將不具參考點之附屬軸的實際值設為電源開啟時的啟動初值。
N37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[AX2]=1	; 啟動永久耦合。
N37263 NEWCONF	

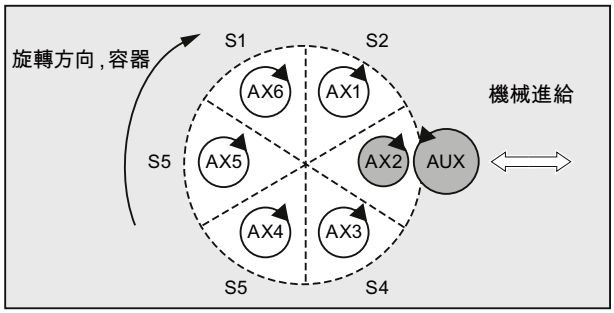
範例 3：耦合順序，位置 3 / 容器 CT1

容器旋轉之後若要啟用其他主軸的耦合，必須先將先前的耦合退耦、清除設定並設定新的耦合。

初始狀態：



旋轉一個插槽之後：



參考：

功能手冊，延伸功能；多個操作面板正面以及 NCU（B3），“軸容器”一章

其它資訊

一般

MASLOF	您可在速度控制模式下直接針對主軸執行這個指令。此時附屬主軸將保持原來的速度繼續旋轉，直到程式設計新速度為止。

動態設定延伸版本

MASLDEF	從工件程式定義主控 / 附屬群組。事先利用機械參數建立唯一的定義。
MASLDEL	此指令將取消對主控軸的附屬軸指派，並且同時開啟（例如）MASLOF 耦合。 將保留機械參數中所指定的主控 / 附屬定義。
MASLOFS	使用 MASLOFS，可讓附屬主軸在耦合開啟時自動剎車。 定位模式下的軸及主軸僅能在完全靜止（速度為零）的情況下關閉與開啟耦合。

說明

您可利用 PRESETON 讓附屬軸實際值與主控軸實際值同步化。若要這麼做，您必須短暫切換出永久 / 附屬耦合，以便將不具參考點之附屬軸的實際值設為電源開啟時的主控值 / 軸值。然後耦合便重新建立為永久耦合。

永久主控 / 附屬耦合是利用機械參數設定
MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE = 1 加以啟動－此設定對臨時耦合的語言指令無效。

主軸的耦合行為!

對於開放迴圈速度控制模式下的主軸而言，是利用機械參數
MD37263 $MA_MS_SPIND_COUPLING_MODE 明確定義 MASLON、MASLOF、MASLOFS 與 MASLDEL 的耦合行為。

如果將 MD37263 預設為 0，則附屬軸僅能在相關軸皆完全靜止的情況下進行耦合及退耦。MASLOFS 等同於 MASLOF。

當 MD37263 = 1 時，將立即執行耦合指令及動作。使用 MASLON 時，將立即關閉耦合；使用 MASLOFS 或 MASLOF 時則將立即開啟耦合。對於 MASLOFS，當時正在旋轉的附屬主軸隨即自動剎車；而對於 MASLOF，該主軸將保持其旋轉速度，直到程式設計新速度為止。

10 指示同步動作

10.1 基礎

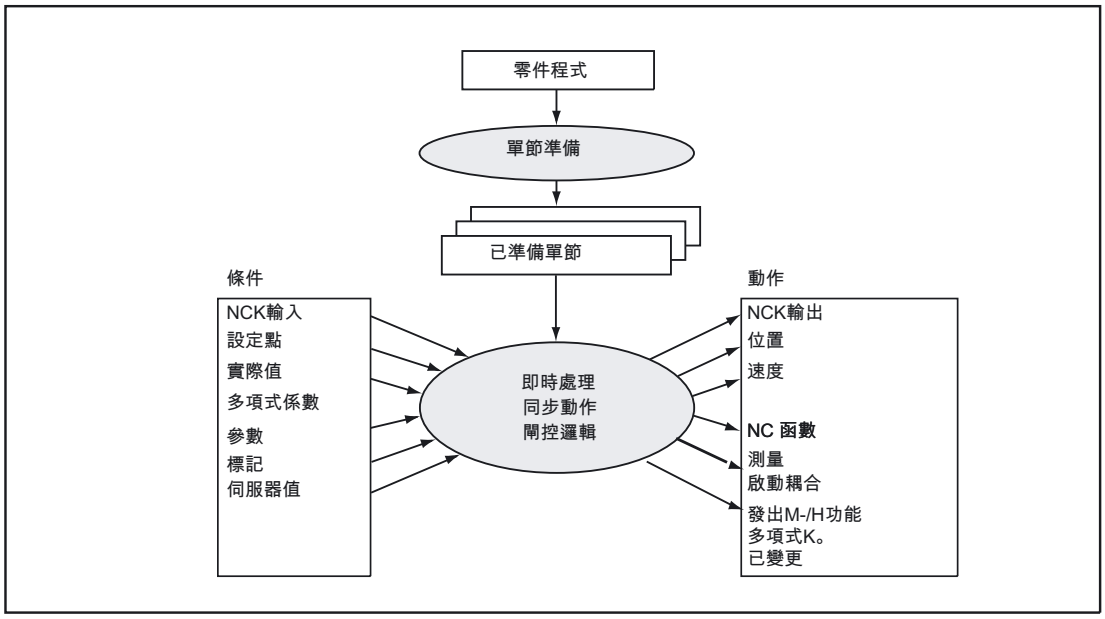
功能

同步動作可讓執行的動作與加工單節同步。

執行動作的時間點可根據各種條件加以定義。而這些條件是在插補循環加以監控。因此，動作是即時事件的回應，其執行不受限於單節之界限。

同步動作中還包含其服務壽命以及已程式設計之主要執行變數的掃描頻率等相關資訊，因此亦包含動作啟動頻率相關資訊。如此一來，動作在插補循環中可僅僅觸發一次或週期性觸發。

可能的應用

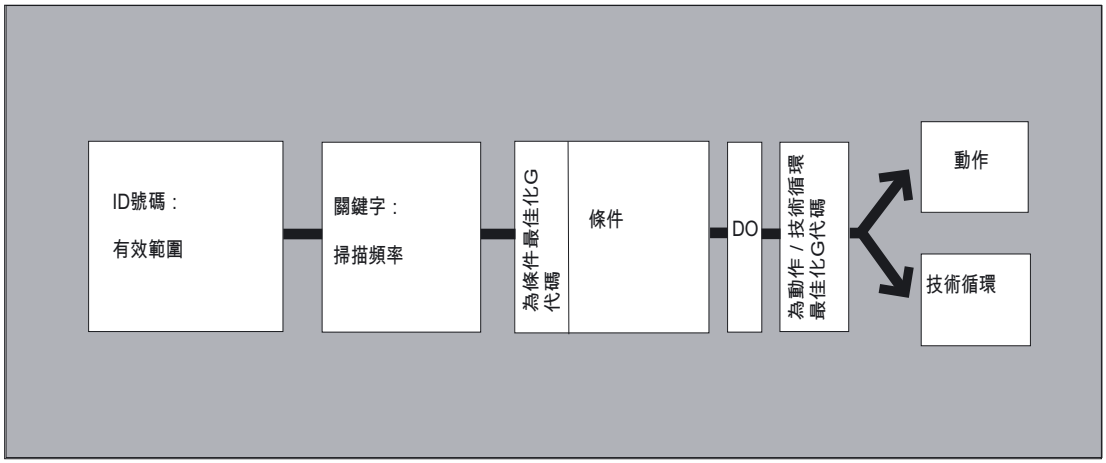


- 將執行時間緊迫的應用程式（例如換刀）最佳化
- 快速回應外部事件
- 程式設計 AC 控制
- 參數設定安全功能
- ....

程式設計

同步動作自成一個獨立單節，您可在其中對它進行程式設計；此動作會在下一個可執行單節上觸發機台功能（例如使用 G0、G1、G2、G3 的移動動作）。

同步動作可包含最多 5 個不同作業的指令元件：



句法：

```
DO <動作 1> <動作 2> …
<KEYWORD> <條件> DO <動作 1> <動作 2> …
ID=<n> <KEYWORD> <條件> DO <動作 1> <動作 2> …
IDS=<n> <KEYWORD> <條件> DO <動作 1> <動作 2> …
```

意義：

DO	開始已程式設計之動作的一個指令 唯有滿足<條件>時（如有對此進行程式設計的話），才會生效。 →請參閱“動作”
<動作 1> <動作 2> …	欲啟動之動作 範例： • 指派變數 • 開始技術循環
<關鍵字>	同步動作的<條件>週期檢查是利用關鍵字（WHEN、WHENEVER、FROM 或 EVERY）加以定義。 →請參閱“條件的週期檢查”
<條件>	主要執行變數的開控邏輯 條件是在 IPO 時脈循環中進行檢查
ID=<n> 或 IDS=<n>	識別碼 加工順序中的適用區域及位置是利用識別碼加以定義。 →請參閱“適用區域與加工順序”

協調同步動作 / 技術循環

下列指令可用來協調同步動作 / 技術循環：

指令	意義
CANCEL（<n>）	取消同步動作 →請參閱“取消同步動作”
LOCK（<n>）	停用同步動作
UNLOCK（<n>）	解鎖同步動作
重置	重置技術循環 關於 LOCK、UNLOCK 與 RESET: →請參閱“禁止、啟用、重置”

範例

程式碼	註解
WHEN $AA_IW[Q1]>5 DO M172 H510	; 如果軸 Q1 的實際值超過 5 毫米，輔助功能 M172 與 H510 將輸出至 PLC 介面。

10.1.1 適用區域與加工順序（ID、IDS）

功能

適用區域

同步動作的適用區域是以識別 ID 或 IDS 加以定義：

無模態 ID:	自動模式下的非模態同步動作
ID:	ID=在程式結束之前於自動模式下共有 n 個模態同步動作
IDS:	靜態同步動作在所有操作模式下皆為模態有效，即使是程式結束之後亦為如此

應用

- 寸動進給模式下的 AC 迴圈
- 安全整合邏輯操作
- 監控功能，在所有模式下回應機台狀態

執行順序

靜態有效之模態同步動作是在插補時脈循環中依照其 ID 或 IDS 編號之順序進行處理（ID=<n>或 IDS=<n>）。

非模態同步動作（無 ID 編號）是在模態同步動作執行完畢之後才按程式設計之順序執行。

說明

模態同步動作可避免機台參數設定對其進行修改或刪除（→機台架構 OEM！）。

程式設計

句法	意義
無模態 ID	同步動作僅於**自動模式**下有效。這條規則若僅適用下一個可執行單節（包含動作敘述或其他機台動作的單節），即為**非模態**。 範例： `WHEN $A_IN[3]==TRUE DO $A_OUTA[4]=10`
`ID=<n> ...`	在後續單節中，同步動作為**模態**套用，您可利用 `CANCEL（<n>）`停用之，或者以相同 ID 程式設計新的同步動作亦可覆寫之。 在 `M30` 單節中為啟用狀態的同步動作延遲程式結束。 ID 同步動作僅於**自動模式**下有效。 **<n>**的值域：　1 ... 255 範例： `ID=2 EVERY $A_IN[1]==1 DO POS[X]=0`
`IDS=<n>`	靜態同步動作在**所有模式下**皆以模態方式作用。此類動作即使在程式結束之後仍為啟用狀態，您可於電源開啟後直接以非同步子程序啟用之。這表示，您可以不用顧慮所選操作模式啟用應於 NC 中執行的動作。 **<n>**的值域：　1 ... 255 範例： `IDS=1 EVERY $A_IN[1]==1 DO POS[X]=100`

10.1.2 週期性檢查條件（WHEN、WHENEVER、FROM、EVERY）

功能

關鍵字是用來定義同步動作條件的週期檢查。若未程式設計關鍵字，每個 IPO 循環都將執行一次同步動作中的所有動作。

關鍵字

無關鍵字	執行動作時不受任何條件限制。將於每一插補循環中週期性執行此動作。
`WHEN`	每個插補循環中皆會掃描條件直到條件滿足為止，然後立刻執行一次相關動作。
`WHENEVER`	插補循環中所包含的每一個循環皆會檢查條件。當條件滿足時，將在每一插補循環中執行相關動作。
`FROM`	每個插補循環中皆會檢查條件直到條件滿足為止。然後等待同步動作為啟用狀態時（即使此時條件已不滿足）執行動作。
`EVERY`	每個插補循環中皆會掃描條件。條件一旦滿足時立即執行動作。 邊緣觸發： 當條件從 FALSE 狀態轉變為 TRUE 狀態時，再次執行動作。

主要執行變數

所使用的變數會在插補循環中進行估算。同步動作中的主要執行變數不會觸發前置處理停止。

分析：

如果主要執行變數出現在工件程式中（例如實際值、數位輸入 / 輸出位置等等），則前置處理將停止，等待上一個單節執行完畢並取得主要執行變數的數值。

範例

範例 1：無關鍵字

程式碼	註解
`DO $A_OUTA[1]=$AA_IN[X]`	; 實際值輸出至類比輸出。

範例 2：WHENEVER

程式碼	註解
`WHENEVER $AA_IM[X] > 10.5*SIN（45） DO …`	; 與前置處理期間計算出來的運算式作比較
`WHENEVER $AA_IM[X] > $AA_IM[X1] DO …`	; 與其他主要執行變數作比較。
`WHENEVER （$A_IN[1]==1） OR （$A_IN[3]==0） DO ...`	; 兩項比較操作互相鬧控。

範例 3：EVERY

程式碼	註解
`ID=1 EVERY $AA_IM[B]>75 DO POS[U]=IC（10） FA[U]=900`	; 每當 B 軸實際值超過機械座標值 75 時，U 軸應按照軸進給向前移動 10 個單位。

其它資訊

條件

條件是一個可利用布林運算子以任何方式建立的邏輯運算式。布林運算式應一律外加括弧。

在插補循環中會檢查條件。

G 碼可以放置在條件之前。這可在不顧慮目前工件程式狀態的情況下，使用已定義之設定來評估條件以及欲執行之動作 / 技術循環。您必須將同步動作從程式環境中單獨隔離出來，因為同步動作必須能夠在觸發條件滿足時，隨時從已定義之初始狀態執行其動作。

應用

透過 G 代碼 G70、G71、G700、 G710 定義條件評估與動作的量測系統

如果動作沒有指定獨立的 G 碼，則條件評估與動作將適用原指定供條件使用的 G 碼。

條件的每一個部分僅能程式設計 G 碼群組中的**一個 G 碼**。

可能的條件

- 比較主要執行變數（類比 / 數位輸入或輸出等等）
- 比較結果的布林閘控
- 即時運算式的計算
- 從單節開始以來的時間 / 距離
- 至單節結束的距離
- 量測值，量測結果
- 伺服值
- 速率，軸狀態

10.1.3 動作（DO）

函數

您可在同步動作中程式設計一個以上動作。在相同插補循環中，同一單節的所有動作皆為生效狀態。

句法

```
DO <動作 1> <動作 2> ...
```

意義

DO	條件滿足時開始執行動作或技術循環。
<動作>	條件滿足時開始動作，例如：指派變數、啟動軸耦合、設定 NCK 輸出、輸出 M、S 與 H 功能、指定已程式設計之 G 代碼...

您可在動作 / 技術循環的同步動作中程式設計 **G 代碼**。此 G 碼可以不同於單節與技術循環中所有動作條件的 G 代碼。如果動作部分包含技術循環，則即使技術循環已經完成，G 碼仍將對所有動作維持模態啟用狀態，直到下一個 G 代碼為止。

每一個動作區段僅能程式設計 G 代碼群組中的**一個 G 代碼**（G70、 G71、 G700、 G710）。

範例：包含兩個動作的同步動作

程式碼	註解
`WHEN $AA_IM[Y]>=35.7 DO M135 $AC_PARAM=50`	; 如果條件滿足，在 PLC 上輸出 M135 且手動倍率為 50%。

10.2 條件與動作的運算子

比較 （==、<>、<、>、<=、>=）	在條件中，您可比較變數或部分運算式。結果的資料類型永遠為 BOOL。所有常見比較運算子皆可使用。
布林運算子 （NOT、AND、OR、XOR）	變數、常數或比較可利用布林運算子相互連結在一起。
逐一位元運算子 （B_NOT、B_AND、B_OR、B_XOR）	您可使用位元運算子（B_NOT、B_AND、B_OR、B_XOR）。
基本算術運算 （+、-、*、/、DIV、MOD）	主要執行變數可以依照基本運算的格式將彼此互相連結或者與常數連結。
數學函數 （SIN、COS、TAN、ASIN、ACOS、ABS、TRUNC、ROUND、LN、EXP、ATAN2、POT、SQRT、CTAB、CTABINV）。	數學函數不適用於 REAL 資料類型的變數。
索引	可利用主要執行運算式進行索引。

範例

- **混合運用的基本算術運算**

先乘除後加減，可在運算式加上括弧。DIV 與 MOD 運算子可用於 REAL 資料類型。

程式設計	註解
DO $AC_PARAM[3] = $A_INA[1]-$AA_IM[Z1]	; ；兩者相減
	；主要執行變數
WHENEVER $AA_IM[x2] < $AA_IM[x1]-1.9 DO $A_OUT[5] = 1	; ；變數減去常數
DO $AC_PARAM[3] = $INA[1]-4*SIN（45.7 $P_EP[Y]）*R4	；常數運算式，於前置處理時計算得出

- **數學函數**

程式設計	註解
DO $AC_PARAM[3] = COS（$AC_PARAM[1]）	;
	;

- **即時運算式**

程式設計	註解
ID=1 WHENEVER （$AA_IM[Y]>30） AND （$AA_IM[Y]<40） DO $AA_OVR[S1]=80	; 選擇位置視窗
ID=67 DO $A_OUT[1]=$A_IN[2] XOR $AN_MARKER[1]	; 估算 2 個布林訊號
ID=89 DO $A_OUT[4]=$A_IN[1] OR （$AA_IM[Y]>10）	; 輸出 比較結果

- 已索引之主要執行變數

程式設計	註解
WHEN…DO $AC_PARAM[$AC_MARKER[1]] = 3	;
不合規定	;
$AC_PARAM[1] = $P_EP[$AC_MARKER]	;

10.3 同步動作的主要執行變數

10.3.1 系統變數

功能

您可在系統變數的協助下讀寫 NC 資料。前置處理變數與主要執行系統變數兩者之間存在明顯差異。前置處理變數永遠是在前置處理期間執行。而主要執行變數則是永遠根據目前主要執行狀態才計算其值。

名稱

系統變數的名稱通常以“$”字元為首：

前置處理變數：

$M...	機械參數
$S...	設定資料，保護區
$T...	刀具管理資料
$P...	經程式設計的數值，前置處理資料
$C...	ISO 封套循環的循環變數
$O...	選項資料
R...	R 參數

主要執行變數：

$$A...	實際主要執行資料
$$V...	伺服資料
$R...	R 參數

第二個字母代表存取變數的選項：

N...	NCK 全域值（通常為有效值）
C...	通道專屬值
A...	軸專屬值

第二個字母通常僅用於主要執行變數。執行前置處理變數（例如$P_）時通常不會用到第二個字母。

前置字元（$號後面會有一至二個字母）之後是底線及後續的變數名稱（通常是英文指定或縮寫）。

資料類型

主要執行變數的資料類型包含：

INT	整個值是前置字元符號加上整數
REAL	實數代表有理計數
BOOL	布林數 TRUE 與 FALSE
CHAR	ASCII 字元
STRING	含英數字元之字元字串
AXIS	軸位址與主軸

前置處理變數的資料類型亦包含：

FRAME	座標轉換

變數陣列

系統變數可設定為 1 至 3 維陣列。

支援資料類型如下：BOOL，CHAR，INT，REAL，STRING，AXIS

索引的資料類型可以是 INT 或 AXIS，因此該值可依要求加以排序。

STRING 變數可設定為 2 維陣列。

陣列定義範例：

```
DEF BOOL $AA_NEWVAR[x, y, 2]
DEF CHAR $AC_NEWVAR[2, 2, 2]
DEF INT $AC_NEWVAR[2, 10, 3]
DEF REAL $AA_VECTOR[x, y, z]
DEF STRING $AC_NEWSTRING[3, 3]
DEF AXIS $AA_NEWAX[x, 3, y]
```

說明

系統變數若有 OPI 變數，則可不受任何限制地顯示 3 維系統變數

10.3.2 隱含類型轉換

功能

於數值指派與參數傳輸的期間，將指派或傳輸不同資料類型的變數。
隱含類型轉換將會觸發數值的內部類型轉換。

可能的類型轉換

收件人	REAL	INT	BOOL	CHAR	STRING	AXIS	FRAME
from							
REAL	是：	是*	是 1)	–	–	–	–
INT	是：	是：	是 1)	–	–	–	–
BOOL	是：	是：	是：	–	–	–	–

說明

* 進行由 REAL 至 INT 的類型轉換時，分數值將採四捨五入至整數（請參照 ROUND 函數）。
如果超過數值，則將輸出警報。

1) 值 <> 0 為 TRUE；值 == 0 為 FALSE

結果

```
由 REAL 或 INTEGER 至 BOOL 的類型轉換
Result BOOL = TRUE              如果 REAL 或 INTEGER 值不等於零
Result BOOL = FALSE             如果 REAL 或 INTEGER 值等於零
由 BOOL 或 REAL 至 INTEGER 的類型轉換
Result REAL TRUE                如果 BOOL 值 = TRUE （1）
Result INTEGER = TRUE           如果 BOOL 值 = TRUE （1）
由 BOOL 或 REAL 至 INTEGER 的類型轉換
Result REAL FALSE）             如果 BOOL 值= FALSE （0）
Result INTEGER = FALSE          如果 BOOL 值= FALSE （0）
```

隱含類型轉換範例

```
由 INTEGER 至 BOOL 的類型轉換
$AC_MARKER[1]=561
ID=1 WHEN $A_IN[1] == TRUE DO $A_OUT[0]=$AC_MARKER[1]

由 REAL 至 BOOL 的類型轉換
R401 = 100.542
WHEN $A_IN[0] == TRUE DO $A_OUT[2]=$R401

由 BOOL 至 INTEGER 的類型轉換
ID=1 WHEN $A_IN[2] == TRUE DO $AC_MARKER[4] = $A_OUT[1]]

由 BOOL 至 REAL 的類型轉換
R401 = 100.542
WHEN $A_IN[3] == TRUE DO $R10 = $A_OUT[3]
```

10.3.3 GUD 變數

GUD 變數能同步動作

和特定的系統變數一樣，已定義的全域同步動作使用者變數（同步動作 GUD）也能在同步動作中使用。可供使用者使用的同步動作 GUD 之號碼，係參數化給各特定資料類型，並可使用下列機械參數來存取：

- MD18660 $MM_NUM_SYNACT_GUD_**REAL**[<x>] = <number>
- MD18661 $MM_NUM_SYNACT_GUD_**INT**[<x>] = <number>
- MD18662 $MM_NUM_SYNACT_GUD_**BOOL**[<x>] = <number>
- MD18663 $MM_NUM_SYNACT_GUD_**AXIS**[<x>] = <number>
- MD18664 $MM_NUM_SYNACT_GUD_**CHAR**[<x>] = <number>
- MD18665 $MM_NUM_SYNACT_GUD_**STRING**[<x>] = <number>

索引**<x>**可用來指定資料單節（存取權限）以及值**<號碼>**，來指定同步動作 GUDs 的特定號碼給各資料類型（REAL，INT，等等）使用。具有下列命名機制的 1 維陣列變數，隨後會在相關資料單節中被建立，供各資料類型所用：SYG_<data type><access right>[<index>]：

索引 <x>		資料類型 （MD18660 至 MD18665）					
	單節	REAL	INT	BOOL	AXIS	CHAR	STRING
0	SGUD	SYG_RS[i]	SYG_IS[i]	SYG_BS[i]	SYG_AS[i]	SYG_CS[i]	SYG_SS[i]
1	MGUD	SYG_RM[i]	SYG_IM[i]	SYG_BM[i]	SYG_AM[i]	SYG_CM[i]	SYG_SM[i]
2	UGUD	SYG_RU[i]	SYG_IU[i]	SYG_BU[i]	SYG_AU[i]	SYG_CU[i]	SYG_SU[i]
3	GUD4	SYG_R4[i]	SYG_I4[i]	SYG_B4[i]	SYG_A4[i]	SYG_C4[i]	SYG_S4[i]
4	GUD5	SYG_R5[i]	SYG_I5[i]	SYG_B5[i]	SYG_A5[i]	SYG_C5[i]	SYG_S5[i]
5	GUD6	SYG_R6[i]	SYG_I6[i]	SYG_B6[i]	SYG_A6[i]	SYG_C6[i]	SYG_S6[i]
6	GUD7	SYG_R7[i]	SYG_I7[i]	SYG_B7[i]	SYG_A7[i]	SYG_C7[i]	SYG_S7[i]
7	GUD8	SYG_R8[i]	SYG_I8[i]	SYG_B8[i]	SYG_A8[i]	SYG_C8[i]	SYG_S8[i]
8	GUD9	SYG_R9[i]	SYG_I9[i]	SYG_B9[i]	SYG_A9[i]	SYG_C9[i]	SYG_S9[i]
Where i = 0 to （<number> - 1） Block：_N_DEF_DIR/_N_ ... _DEF，例如，SGUD ⇒ _N_DEF_DIR/_N_**SGUD**_DEF							

屬性

同步動作 GUD 具有下列特性：

- 同步動作 GUD 在同步動作中，以及工件程式/循環中被讀取與寫入。
- 同步動作 GUD 可透過 OPI 來存取。
- 同步動作 GUD 可顯示在"參數"操作區中的 HMI 使用者介面上。
- 同步動作 GUD 可在精靈中的 HMI 上、變數檢視中以及變數日誌中使用。
- STRING 類型同步動作 GUD 的陣列大小，設定為固定值，32（31 字元 + \0）。
- 即使沒有手動建立定義檔供全域使用者資料（GUD）使用，使用機械參數定義的同步動作 GUD，可從 HMI，在對應的 GUD 單節中讀取。

注意

若沒有同步動作 GUD 曾以相同的名稱（MD18660 - MD18665）參數化，使用者變數（GUD，PUD，LUD）僅能使用和同步動作 GUD 相同的名稱來定義（`DEF ... SYG_xy`）。這些 GUD 的使用者自訂項目**不能**用在同步動作中。

存取權限

在 GUD 定義檔中定義的存取權限，仍保持有效，且僅參考定義在此 GUD 定義檔中的 GUD 變數。

刪除行為：

如果重新啟動特定 GUD 定義檔的內容，則會先刪除現用檔案系統中的舊 GUD 資料單節。已設定的同步動作 GUD 也會在此重置。透過 HMI 到操作區“服務”（Services）>“定義與已啟動之使用者資料（GUD）”（Define and activated user data（GUD）），亦可進行此一流程。

10.3.4 預設軸識別碼（NO_AXIS）

功能

若欲使用尚未用數值初始化的 AXIS 類型變數或參數，您可運用已定義之預設軸識別碼。未定義之軸變數便是以此預設值進行初始化。

您可藉由查詢“NO_AXIS”變數，辨識出同步動作中的非初始化之有效軸名稱。機械參數會指派已設定之預設軸識別碼給此非初始化之軸識別碼。

機械製造商

您必須利用機械參數定義並預先指派至少一個有效之現有軸識別碼。不過，您亦可將現有之有效軸識別碼全部指派。請參閱機台製造商說明。

說明

定義時，會自動提供一個值給新建立的變數，該值即是預設軸名稱之機械參數中所儲存的值。

如需有關可利用機械參數套用之定義的詳細資訊，請參閱：

參考：

/FBSY/ 功能手冊，同步函數

句法

```
PROC UP（AXIS PAR1=NO_AXIS，AXIS PAR2=NO_AXIS）
IF PAR1 <>NO_AXIS...
```

意義

PROC	副程式定義
SR	供辨識之用的副程式名稱
PARn	參數 n
NO_AXIS	以預設軸識別碼初始化公式參數

範例：定義主程式中的軸變數

程式碼

```
DEF AXIS AXVAR
UP（ ，AXVAR）
```

10.3.5 同步動作標記（$AC_MARKER[n]）

功能

您可在同步動作中讀寫陣列變數$AC_MARKER[n]。這些變數儲存在主動或被動檔案系統的記憶體中。

同步動作變數：INT 資料類型

$AC_MARKER[n]	通道專屬標記 / 計數器，INTEGER 資料類型
$MC_MM_NUM_AC_MARKER	設定通道專屬標記編號以利移動時所使用的機械參數。同步動作
n	變數陣列索引 0-n

讀寫標記變數範例

程式碼

```
WHEN ... DO $AC_MARKER[0] = 2
WHEN ... DO $AC_MARKER[0] = 3
WHENEVER $AC_MARKER[0] == 3 DO $AC_OVR=50
```

10.3.6 同步動作參數（$AC_PARAM[n]）

功能

在同步動作中使用同步動作參數$AC_PARAM[n]是為了計算目的，並且亦可作為中間記憶體。這些變數儲存在主動或被動檔案系統的記憶體中。

同步動作變數：資料類型：REAL

這些參數亦以相同名稱一度存在於各個通道中。

$AC_PARAM[n]	指示同步動作的算術變數（REAL）
$MC_MM_NUM_AC_PARAM	設定參數編號以利移動時所使用的機械參數。同步動作的最大數量可高達 20000 個。
n	參數陣列索引 0n

同步動作參數$AC_PARAM[n]範例

程式碼

```
$AC_PARAM[0]=1.5
$AC_MARKER[0]=1
ID=1 WHEN $AA_IW[X]>100 DO $AC_PARAM[1]=$AA_IW[X]
ID=2 WHEN $AA_IW[X]>100 DO $AC_MARKER[1]=$AC_MARKER[2]
```

10.3.7 算術參數（$R[n]）

功能

在工件程式以及同步動作中使用此靜態陣列變數是為了計算目的。

句法

工件程式中的程式設計：
```
REAL R[n]
REAL Rn
```
同步動作中的程式設計：
```
REAL $R[n]
REAL $Rn
```

算術參數

利用算術參數您可以：

- 儲存您想在程式結束、NC 重置以及電源開啟之後仍保留下來的數值
- 在 R 參數顯示中顯示已儲存的數值。

範例

程式碼	註解
WHEN $AA_IM[X]>=40.5 DO $R10=$AA_MM[Y]	; 在同步動作中使用 R10。
G01 X500 Y70 F1000	
STOPRE	; 前置處理停止
IF R10>20	; 評估算術變數。

程式碼

```
SYG_AS[2]=X
SYG_IS[1]=1
WHEN $AA_IM[SGY_AS[2]]>10 DO $R3=$AA_EG_DENOM[SYG_AS[1]], SYG_AS[2]]
WHEN $AA_IM[SGY_AS[2]]>12 DO $AA_SCTRACE[SYG_AS[2]]=1
SYG_AS[1]=X
SYG_IS[0]=1
WHEN $AA_IM[SGY_AS[1]]>10 DO $R3=$$MA_POSCTRL_GAIN[SYG_IS[0]], SYG_AS[1]]
WHEN $AA_IM[SGY_AS[1]]>10 DO $R3=$$MA_POSCTRL_GAIN[SYG_AS[1]]
WHEN $AA_IM[SGY_AS[1]]>15 DO $$MA_POSCTRL_GAIN[SYG_AS[0]], SYG_AS[1]]=$R3
```

10.3.8 讀寫 NC 機台與 NC 設定資料

功能

您亦可讀寫同步動作的 NC 機台 / 設定資料。在程式設計期間讀寫機械參數陣列元素時，可省略索引。如果這是發生在工件程式裡，則當您讀取**第一個**陣列元素或當您進行寫入時，陳列的所有元素將連同此值一併說明。

因此，在同步動作中，將僅讀寫**第一個**元素。

定義

MD、SD 帶有：

$：於同步動作的插補時間讀取值

$$：於主要執行期間讀取值

於前置處理時間讀取 MD 值與 SD 值

您可在前置處理時間利用**$**字元於同步動作中找出這些值並加以估算。

```
ID=2 WHENEVER $AA_IM[z]<$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
; 此處是將反轉範圍 2 假設為於操作期間保持靜態，並將之找出以進行震盪。
```

於主要執行時間讀取 MD 值與 SD 值

您可在主要執行時間利用**$**字元於同步動作中找出這些值並加以估算。

```
ID=1 WHENEVER $AA_IM[z]<$$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
; 此處假設反位可在加工期間以指令修改
```

於主要執行時間寫入 MD 與 SD

目前所設定的存取授權層級必須具有允許寫入之權限。關於所有 MD 與 SD 之啟用狀態詳細列述於**參考：**/LIS/，清單（書 1）。

您必須在欲寫入之 MD 與 SD 前面加上$$，然後找出其位置。

範例

程式碼	註解
ID=1 WHEN $AA_IW[X]>10 DO $$SN_SW_CAM_PLUS_POS_TAB_1[0]=20 $$SN_SW_CAM_MINUS_POS_TAB_1[0]=30	; 變更軟體擋塊切換位置。注意事項：您必須在抵達切換位置之前，變更二至三插補循環中的該位置。

10.3.9 計時器變數（$AC_Timer[n]）

功能

系統變數$AC_TIMER[n]允許於延遲期間過後才開始動作。

計時器變數：資料類型：REAL

$AC_TIMER[n]	REAL 資料類型之通道專屬計時器
秒	單位是秒
n	計時器變數索引

設定計時器

利用值指派計時器變數開始累加：
```
$AC_TIMER[n]=value
```

n：	時間變數編號
：	起始值（通常是“0”）

停止計時器

藉由指派負值可停止計時器變數累加：
```
$AC_TIMER[n]=-1
```

讀取計時器

無論計時器變數是在執行中或是已停止，您皆可讀取目前的計時器值。透過指派-1 值將計時器變數停止之後，會一直保存目前時間值，因此可進行讀取。

範例

於偵測到數位輸入之後，經由類比輸出 500 毫秒輸出實際值：

程式碼	註解
`WHEN $A_IN[1] == 1 DO $AC_TIMER[1]=0`	`;` 重置與啟動計時器
`WHEN $AC_TIMER[1]>=0.5 DO $A_OUTA[3]=$AA_IM[X] $AC_TIMER[1]=-1`	

10.3.10 FIFO 變數（$AC_FIFO1[n] ... $AC_FIFO10[n]）

功能

共有 10 個 FIFO 變數（循環緩衝儲存器）可用來儲存相關資料順序。
資料類型：REAL

應用：

- 週期性量測
- 執行傳遞

讀取或寫入時每個元素皆可存取

FIFO 變數

可用之 FIFO 變數數量定義於機械參數 MD28260 $MC_NUM_AC_FIFO 中。

可寫入至 FIFO 變數的數值數量是利用機械參數 MD28264 $MC_LEN_AC_FIFO 加以定義。
所有 FIFO 變數的長度皆相同。

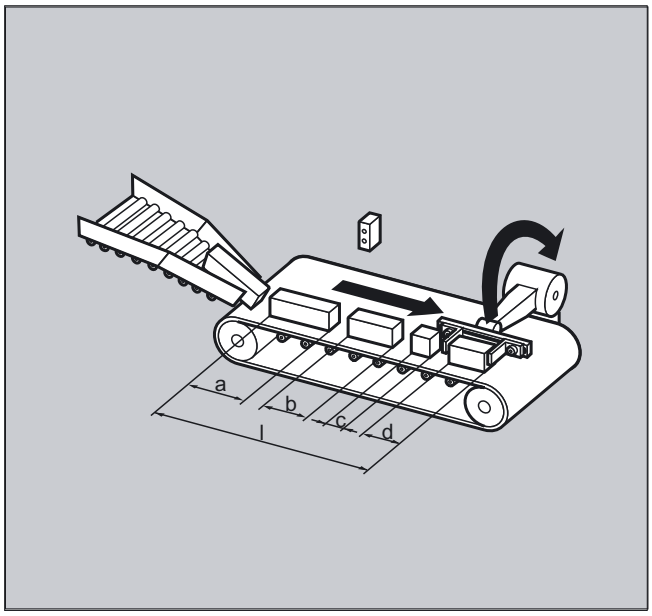
唯有在 MD28266 $MC_MODE_AC_FIFO 中設定位元 0 時，才會產出所有 FIFO 元素總和。

索引 **0 至 5** 具有特別意義：

索引	意義	
0	寫入期間：	新值儲存於 FIFO 中。
	讀取期間：	讀取最古老的元素並將之從 FIFO 移除。
1	存取最先儲存的元素	
2	存取最後儲存的元素	
3	所有 FIFO 元素總和	
4	FIFO 中可用之元素數量 讀寫存取可指派至每一個 FIFO 元素。您可透過重置元素數量（例如第一個 FIFO 變數），藉此重置 FIFO 變數：$AC_FIFO1[4]=0	
5	與 FIFO 開頭相關的目前寫入索引	
6 至 n_{max}	存取第 n 個 FIFO 元素	

範例：循環記憶體

執行產出期間，會利用一條傳送帶來傳輸不同長度的產出（a，b，c，d）。傳送帶的傳輸長度將視程流中相關個別產出的長度而定，因此該傳送帶所攜帶的產出數量不定。在固定的傳輸速度下，從傳送帶移除產出之功能必須配合產出的變數抵達時間。



程式碼	註解
`DEF REAL INTV=2.5`	; 擱下之產出間的固定距離。
`DEF REAL TOTAL=270`	; 長度量測與移除位置間的距離。
`EVERY $A_IN[1]==1 DO $AC_FIFO1[4]=0`	; 流程開始時 重置 FIFO
`EVERY $A_IN[2]==1 DO $AC_TIMER[0]=0`	; 若有產出中斷光障壁，則開始時間量測。
`EVERY $A_IN[2]==0 DO $AC_FIFO1[0]=$AC_TIMER[0]*$AA_VACTM[B]`	; 如果產出離開光障壁，則利用所測得之時間與傳輸速度計算出 產出長度並將之儲存於 FIFO 中。
`EVERY $AC_FIFO1[3]+$AC_FIFO1[4]*ZWI>=GESAMT DO POS[Y]=-30` `$R1=$AC_FIFO1[0]`	; 一旦所有產出長度與中間距離的總和若大於或等於 產出擱下之位置與 其移除位置之間的長度，則立即在移除位置上將產出從傳送帶移除，並從 FIFO 讀出相關產出長度。

10.3.11 插補器中單節類型的相關資訊（$AC_BLOCKTYPE、$AC_BLOCKTYPEINFO、$AC_SPLITBLOCK）

功能

針對同步動作共有以下系統變數可在主要執行期間提供正在執行之單節的相關資訊：

- $AC_BLOCKTYPE
- $AC_BLOCKTYPEINFO
- $AC_SPLITBLOCK

單節類型與單節類型資訊變數

$AC_BLOCKTYPE		$AC_BLOCKTYPEINFO				
值：		值：				
0	不等於 0	T	H	Z	E	含義：
原始單節	中間單節					中間單節的觸發：
	1	1	0	0	0	內部產生的單節，無詳細資訊
	2	2	0	0	1	倒角 / 倒圓角直線
	2	2	0	0	2	倒角 / 倒圓角圓形
	3	3	0	0	1	WAB：以直線方式逼近
	3	3	0	0	2	WAB：以四分之一圓方式逼近
	3	3	0	0	3	WAB：以半圓逼近
						刀具補償
	4	4	0	0	1	於 STOPRE 之後的逼近單節
	4	4	0	0	2	找不到交點時的連接單節
	4	4	0	0	3	內角上的點型圓弧 （僅適用於 TRACYL 上）
	4	4	0	0	4	外角上的旁通圓弧（圓錐切削）
	4	4	0	0	5	抑制偏移的逼近單節
	4	4	0	0	6	WRC 重複啟用時的逼近單節
	4	4	0	0	7	因曲率過大而導致的單節分割
	4	4	0	0	8	3D 平面銑削時的補正單節（刀具向量||面積向量）
						轉角磨圓時使用：
	5	5	0	0	1	G641
	5	5	0	0	2	G642
	5	5	0	0	3	G643
	5	5	0	0	4	G644

$AC_BLOCKTYPE		$AC_BLOCKTYPEINFO				
值：		值：				
0	不等於 0	T	H	Z	E	含義：
原始單節	中間單節					中間單節的觸發：
						TLIFT 單節：
	6	6	0	0	1	切線軸線性移動但無提升動作
	6	6	0	0	2	切線軸非線性移動（多項式）但無提升動作
	6	6	0	0	3	提升動作，同時開始切線軸動作與提升動作
	6	6	0	0	4	提升動作，切線軸等到抵達特定提升位置後才會啟動。
						路徑分段：
	7	7	0	0	1	已程式設計之路徑分段為啟用狀態但不進行沖孔或切片
	7	7	0	0	2	已程式設計之路徑分段且包含沖孔或切片動作
	7	7	0	0	3	內部自動產生之路徑分段
						編譯循環：
	8	ID 應用				產生單節之編譯循環應用的 ID

T：千位數
H：百位數
Z：十位數：
E：個位數

說明

如有中間單節，則$AC_BLOCKTYPEINFO 內亦必定包含單節類型值，該值位於千位數（T）。在不等於 0 的$AC_BLOCKTYPE 中，將不接受此千位數。

$AC_SPLITBLOCK	
值：	意義：
0	已程式設計之單節未變更（由壓縮機產生的單節其處理方式與已程式設計之單節相同）
1	表示有一內部產生的單節或是有一原始單節被縮短
3	在一連串由內部產生之單節或已縮短之原始單節中的最後一個單節可供使用

範例：計算順接單節數量

程式碼	註解
`$AC_MARKER[0]=0`	
`$AC_MARKER[1]=0`	
`$AC_MARKER[2]=0`	
`...`	
	; 定義用來計算 順接單節數量的同步動作。
	; 所有順接單節皆是在$AC_MARKER[0]中計算其數量：
`ID=1 WHENEVER （$AC_TIMEC==0） AND （$AC_BLOCKTYPE==5） DO $AC_MARKER[0]=$AC_MARKER[0]+1`	
`...`	
	; 所有以 G641 產生之順接單節皆是在$AC_MARKER[1]中計算其數量：
`ID=2 WHENEVER （$AC_TIMEC==0） AND （$AC_BLOCKTYPEINFO==5001） DO $AC_MARKER[1]=$AC_MARKER[1]+1`	
	; 所有以 G642 產生之順接單節皆是在$AC_MARKER[2]中計算其數量：
`ID=3 WHENEVER （$AC_TIMEC==0） AND （$AC_BLOCKTYPEINFO==5002） DO $AC_MARKER[2]=$AC_MARKER[2]+1`	
`...`	

10.4 同步動作中的動作

10.4.1 同步動作中之可能動作概觀

同步動作中的動作是由數值指派、函數或參數呼叫以及關鍵字或技術循環等組成。利用運算子可進行複雜的執行作業。

可能的應用包含：

- 計算 IPO 循環中的複雜運算式
- 軸動作與主軸控制
- 您可變更來自同步動作的設定資料並在線上加以估算（例如 PLC 或 NC I/O 上軟體擋塊輸出的位置與次數）
- 至 PLC 的輔助函數輸出
- 設定其他安全功能
- 設定疊覆動作、線上刀具偏移與間隙控制
- 在所有操作模式下執行動作
- 從 PLC 影響同步動作
- 執行技術循環
- 輸出數位與類比訊號
- 錄製插補循環上的同步動作效能記錄以及載入報告的位置控制器計算時間
- 使用者介面中的診斷功能

同步動作	說明
DO $V…= DO $A...=	指派變數（伺服值） 指派變數（主要執行變數）
DO $AC…[n]= DO $AC_MARKER[n]= DO $AC_PARAM[n]=	特殊主要執行變數 讀取或寫入同步動作標記 讀取或寫入同步動作參數
DO $R[n]=	讀取或寫入算術變數
DO $MD...= DO $$SD...=	正當進行插補之時讀取 MD 值 於主要執行期間寫入 SD 值
DO $AC_TIMER[n]=Start value	計時器
DO $AC_FIFO1[n] …FIFO10[n]=	FIFO 變數
DO $AC_BLOCKTYPE= DO $AC_BLOCKTYPEINFO= DO $AC_SPLITBLOCK=	插補目前單節（主要執行變數）
DO M-、S 與 H，例如 M07	輸出 M、S 與 H 輔助函數
DO RDISABLE	設定讀入停用
DO STOPREOF	取消前置處理停止
DO DELDTG	在不執行前置處理停止的情況下快速刪除剩餘距離
FTCDEF（多項式、LL、UL、係數） DO SYNFCT（多項式、輸出、輸入）	定義多項式 啟用同步功能：適應性控制
DO FTOC	線上刀具偏移

同步動作	說明
DO G70/G71/G700/G710	定義定位作業的尺寸系統（尺寸為英制或公制單位）
DO POS[axis]= / DO MOV[axis]= DO SPOS[spindle]=	啟動 / 定位 / 停止指令軸 啟動 / 定位 / 停止主軸
DO MOV[Axis]=value	指令軸的啟動 / 定位無限動作
DO POS[Axis]= FA [Axis]=	軸進給 FA
ID=1 ... DO POS[axis]= FA [axis]= ID=2 ... DO POS[axis]= $AA_IM[axis] FA [axis]=	位置取自同步動作
DO PRESETON（axis，value）	設定實際值（從同步動作預設）
ID=1 EVERY $A_IN[1]=1 DO M3 S… ID=2 EVERY $A_IN[2]=1 DO SPOS=	啟動 / 定位 / 停止主軸
DO TRAILON（FA，LA，coupling factor） DO LEADON（FA，LA，NRCTAB，OVW）	啟動耦合軸運作 啟動前導值耦合
DO MEAWA（axis）= DO MEAC（axis）=	啟動軸量測 啟動連續量測
DO [array n，m]=SET（value，value，...） DO [array n，m]=REP（value，value，...）	以值清單初始化陣列變數 以相同值初始化陣列變數
DO SETM（Marker No.） DO CLEARM（Marker No.）	設定等候標記 刪除等候標記
DO SETAL（警報編號）	設定循環警報（其他安全功能）
DO FXS[axis]= DO FXST[axis]= DO FXSW[axis]= DO FOCON[axis]= DO FOCOF[axis]=	選擇移動至固定停止點 變更鉗位扭矩 變更監控視窗 移動以受限扭矩 / 力道啟動（模態）FOC 移動以受限扭矩 / 力道停用 （用於特定單節上的同步動作）
ID=2 EVERY $AC_BLOCKTYPE==0 DO $R1=$AC_TANEB	目前單節結尾上之路徑切線與位於已程式設計之後續單節開頭的路徑切線兩者間的夾角
DO $AA_OVR= DO $AC_OVR= DO $AA_PLC_OVR DO $AC_PLC_OVR DO $AA_TOTAL_OVR DO $AC_TOTAL_OVR	軸手動倍率 路徑手動倍率 屬於從 PLC 指定的軸手動倍率 屬於從 PLC 指定的路徑手動倍率 最後計算出來的軸手動倍率 最後計算出來的路徑手動倍率
$AN_IPO_ACT_LOAD= $AN_IPO_MAX_LOAD= $AN_IPO_MIN_LOAD= $AN_IPO_LOAD_PERCENT= $AN_SYNC_ACT_LOAD= $AN_SYNC_MAX_LOAD= $AN_SYNC_TO_IPO=	實際 IPO 計算時間 最長 IPO 計算時間 最短 IPO 計算時間 實際 IPO 計算時間相對於 IPO 時脈循環的比率 所有通道上實際同步動作計算時間 所有通道上最長同步動作計算時間 全部同步動作的百分比元件
DO TECCYCLE	執行技術循環

同步動作	說明
DO LOCK（n，n，...） DO UNLOCK（n，n，...） DO RESET（n，n，...）	禁止 啟用 技術循環的 RESET
CANCEL（n，n，...）	以 ID（S）指定刪除工件程式中的模態同步動作

10.4.2 輔助函數的輸出

功能

輸出時機

您可在同步動作中於動作輸出時間直接輸出輔助函數。輔助函數相關機械參數中所定義的輸出時機未啟用。

條件滿足之時，就可確定輸出時機。

範例：

在特定軸位置上開啟切削水：

```
WHEN $AA_IM[X]>=15 DO M07 POS[X]=20 FA[X]=250
```

非模態同步動作中所認可的關鍵字（無模態 ID）

在非模態同步動作中的輔助函數（無模態 ID），僅能以 WHEN 或 EVERY 等關鍵字來執行。

說明

同步動作中不允許下列輔助函數：

- M0，M1，M2，M17，M30：程式停止/結尾（M2，M17，M30 技術循環可使用）
- 以 M6 或利用機械參數設定、用於換刀的 M 碼功能

範例

程式碼	註解
`WHEN $AA_IW[Q1]>5 DO M172 H510`	; 如果 Q1 軸的實際值超過 5 毫米，將輸出輔助說明函數 M172 與 H510 至 PLC 介面。

10.4.3 設定讀入停用（RDISABLE）

功能

如果使用 RDISABLE，則當條件滿足時將保留主程式中其他單節的處理。已程式設計之同步指示動作仍將執行，後續單節仍將準備就緒。

在路徑控制模式下，無論 RDISABLE 是否為啟用狀態，只要一進入同步動作中使用 RDISABLE 的單節時便會觸發精確停止。

範例

根據外部輸入，在插補循環中啟動程式。

程式碼	註解
`...`	
`WHENEVER $A_INA[2]<7000 DO RDISABLE`	; 如果輸入 2 上的電壓未達 7V，停止執行程式（1000= 1V）。
`N10 G1 X10`	; 如果條件滿足，N10 結尾上的禁止讀入隨即生效
`N20 G1 X10 Y20`	
`...`	

10.4.4 取消前置處理停止（STOPREOF）

功能

如果啟用之同步動作啟動了已明確程式設計的前置處理停止 STOPRE，或者啟用之同步動作隱含啟動了前置處理停止，則一旦條件滿足後，STOPREOF 會在下一個加工單節之後取消前置處理停止。

說明

STOPREOF 必須以關鍵字 WHEN 及非模態之方式（無 ID 編號）進行程式設計。

範例

單節結束時快速分支程式。

程式碼	註解
`WHEN $AC_DTEB<5 DO STOPREOF`	; 如果至單節結束的距離少於 5 毫米，則撤回前置處理停止。
`G01 X100`	; 執行線性插補之後，撤回前置處理停止。
`IF $A_INA[7]>500 GOTOF MARKE1=X100`	; 如果輸入 7 上的電壓超過 5V，；跳躍至標籤 1。

10.4.5 刪除剩餘距離（DELDTG）

功能

可視條件，為路徑或特定的軸觸發刪除剩餘距離。
可能性包括：

- 快速，已準備刪除剩餘距離
- 未準備刪除剩餘距離

使用 DELDTG 的準備已刪除剩餘距離會允許一對觸發事件的快速反應，因此會應用於時間緊迫的情況，例如，若

- 介於刪除剩餘距離與下個單節開始之間的時間必須非常短暫。
- 非常有可能滿足刪除剩餘距離之條件。

說明

於 DELDTG 後括號中的軸指定只對一定位軸有效。

句法

為路徑
DO DELDTG 刪除剩餘距離
軸刪除剩餘距離
DO DELDTG（axis1） DELDTG（axis2） ...

快速刪除剩餘距離路徑之範例

程式碼	註解
WHEN $A_IN[1]==1 DO DELDTG	
N100 G01 X100 Y100 F1000	; 若輸入已設定，動作將被中斷
N110 G01 X…	
IF $AA_DELT>50…	

快速軸刪除剩餘距離之範例

程式碼	註解
取消位置移動：	
ID=1 WHEN $A_IN[1]==1 DO MOV[V]=3 FA[V]=700	; 啟始軸
WHEN $A_IN[2]==1 DO DELDTG（V）	; 刪除剩餘距離，使用 MOV=0 使軸停止
視輸入電壓而定，刪除剩餘距離：	
WHEN $A_INA[5]>8000 DO DELDTG（X1）	; 待電壓輸入 5 超過 8V 時，刪除軸 X1 之剩餘距離。 路徑動作持續
POS[X1]=100 FA[X1]=10 G1 Z100 F1000	

其它資訊

於一已觸發之已準備刪除剩餘距離的移動單節結尾，會先暗中啟動前置處理停止。
因此會於單節結尾使用快速刪除剩餘距離，中斷或停止連續路徑模式或定位軸移動。

說明

已準備刪除剩餘距離：
- 無法與生效刀具半徑修正一併使用。
- 該動作僅可於非模態同步動作（不含 ID 編號）中程式設計。

10.4.6 多項式定義（FCTDEF）

功能

FCTDEF 可用來定義第三順序之多項式，其格式為 $y=a_0+a_1x+a_2x^2+a_3x^3$。此類多項式是提供給線上刀具偏移（FTOC）與評估函數（SYNFCT）使用。

句法

```
FCTDEF（Polynomial No.，LLIMIT，ULIMIT，a0，a1，a2，a3）
```

意義

Polynomial_No.	三次多項式的編號
LLIMIT	函數值下限
ULIMIT	函數值上限
a0，a1，a2，a3	多項式係數

這些值可透過系統變數存取

$AC_FCTLL[n]	函數值下限
$AC_FCTUL[n]	函數值上限
$AC_FCT0[n]	a_0
$AC_FCT1[n]	a_1
$AC_FCT2[n]	a_2
$AC_FCT3[n]	a_3

說明

寫入系統變數

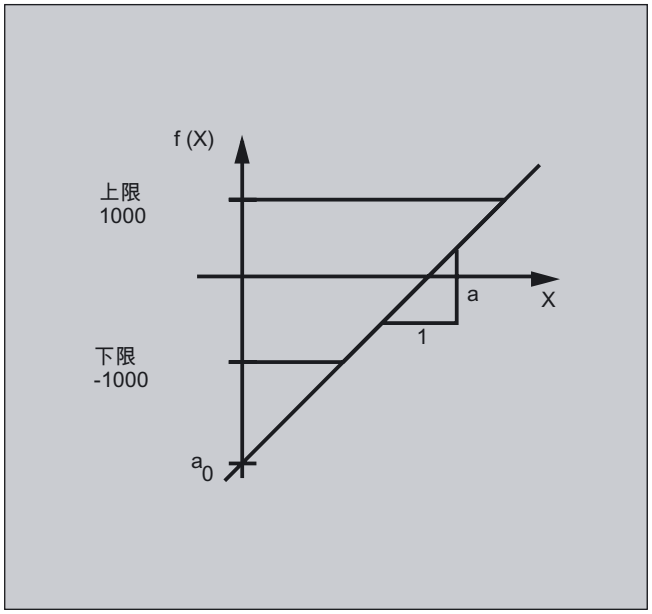
- 可從工件程式或從同步動作寫入系統變數。從工件程式寫入時，請程式設計 STOPRE，以確保寫入作業為單節同步。
- $AC_FCTLL[n]、$AC_FCTUL[n]、$AC_FCT0[n] 至$AC_FCTn[n]等系統變數可從同步動作加以變更

從同步動作寫入時，多項式係數與函數值限制將立即啟用。

直線區段多項式範例

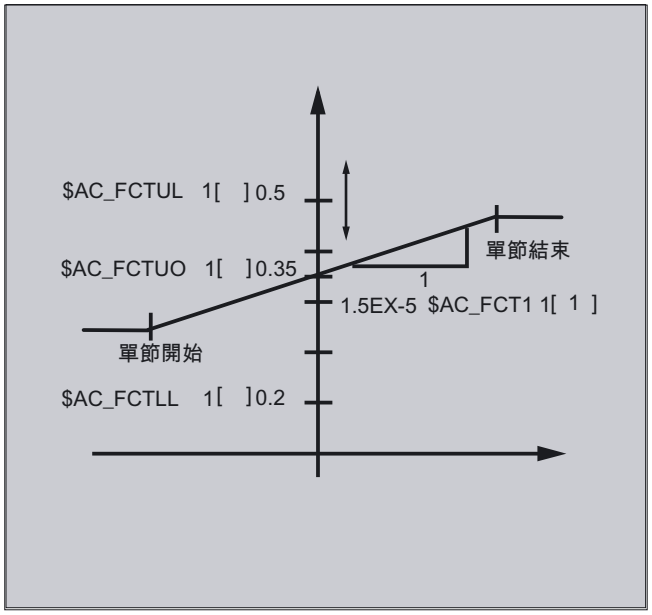
假設上限 1000、下限 -1000、縱座標部分 a_0=$AA_IM[X] 且線形漸層為 1，則多項式為：

FCTDEF（1，-1000，1000，$AA_IM[X]，1）



雷射輸出控制範例

多項式定義的可能應用之一便是雷射輸出控制。
雷射輸出控制是指：
根據（例如）路徑速率，影響類比輸出。



程式碼	註解
`$AC_FCTLL[1]=0.2`	; 定義多項式係數
`$AC_FCTUL[1]=0.5`	
`$AC_FCT0[1]=0.35`	
`$AC_FCT1[1]=1.5EX-5`	
`STOPRE`	
`ID=1 DO $AC_FCTUL[1]=$A_INA[2]*0.1 +0.35`	; 線上變更上限。
`ID=2 DO SYNFCT（1，$A_OUTA[1]，$AC_VACTW）`	; 根據路徑速率（儲存於$AC_VACTW 中），透過類比輸出 1 控制雷射功率控制

說明

以 SYNFCT 使用上述定義之多項式。

10.4.7 同步函數（SYNFCT）

功能

SYNFCT 會利用輸入變數計算出 3 階加權多項式的輸出值。結果位於輸出變數中，具有最大值與最小值限制。
此評估函數用於

- AC 控制（適應性控制），
- 雷射輸出控制，
- 與位置前饋連用

句法

`SYNFCT` （多項式編號、主要執行變數輸出、主要執行變數輸入）

意義

關於輸出變數，您可選擇以下種類變數：

- 具有附加的影響效果
- 具有乘積影響效果
- 可作為位置偏移或者
- 可直接

影響加工流程。

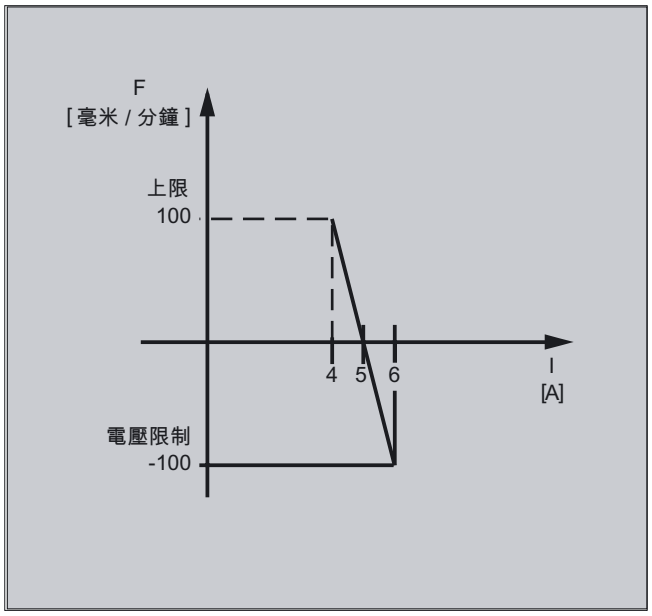
`DO SYNFCT`	啟用評估函數
多項式編號	使用以 FCTDEF 定義之多項式（請參閱“多項式定義”一節）。
主要執行變數輸出	寫入主要執行變數
主要執行變數輸入	讀取主要執行變數

適應性控制範例（附加的）

對於已程式設計之進給率具附加的影響效果

打算利用 X 軸（進給軸）電流對已程式設計之進給率採附加的控制：

進給率僅能以+/- 100 毫米 / 分增減，而環繞 5A 工作點之電流以+/-1A 波動。



1.多項式定義

決定係數

$y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

a_1 = -100mm/1 min A

a_0 = -（-100）*5 =500

$a_2 = a_3 = 0$（無平方與立方元件）

上限 = 100

下限 = -100

這表示：

```
FCTDEF（1，–100，100，500，–100，0，0）
```

2.啟動 AC 控制

```
ID=1 DO SYNFCT（1，$AC_VC，$AA_LOAD[x]）
```

；透過$AA_LOAD[x]讀取目前軸負載（最大驅動電流的百分比），
以上述定義之多項式計算路徑進給率手動倍率。

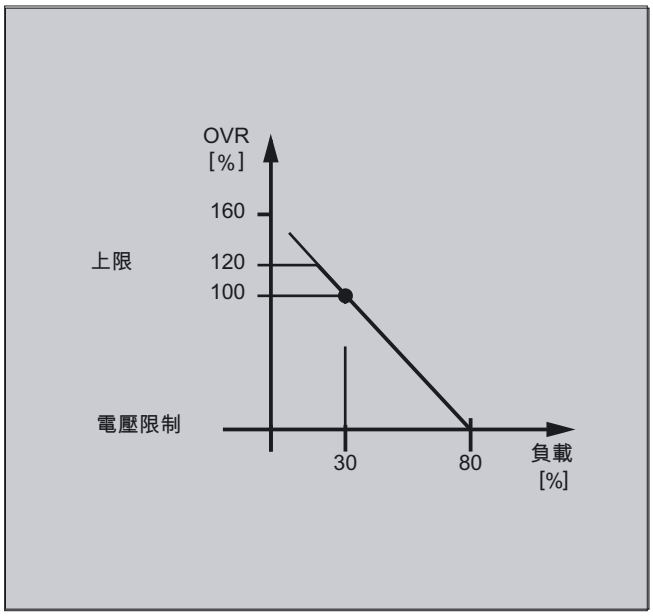
適應性控制範例（乘積）

對於已程式設計之進給率具乘積影響效果

目的是要對已程式設計之進給率產生乘積影響效果。進給率不得超過特定臨界值－根據驅動器上的負載：

- 達到 80%驅動負載時進給率停止：手動倍率 = 0
- 驅動負載為 30%時，得以已程式設計之進給率移動：
  手動倍率 = 100%。

可超過進給率 20%：
最大手動倍率 = 120%。



1.多項式定義

決定係數

$y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

$a_1 = -100\%/(80-30)\% = -2$

$a_0 = 100 + (2*30) = 160$

$a_2 = a_3 = 0$（無平方與立方元件）

上限 = 120

上限 = 0

這表示：

```
FCTDEF（2，0，120，160，-2，0，0）
```

2.切換至 AC 控制

```
ID=1 DO SYNFCT（2，$AC_OVR，$AA_LOAD[x]）
```

；透過$AA_LOAD[x]讀取目前軸負載（最大驅動電流的百分比），
以上述定義之多項式計算進給率手動倍率。

10.4.8 含受限修正之閉迴圈間隙控制（$AA_OFF_MODE）

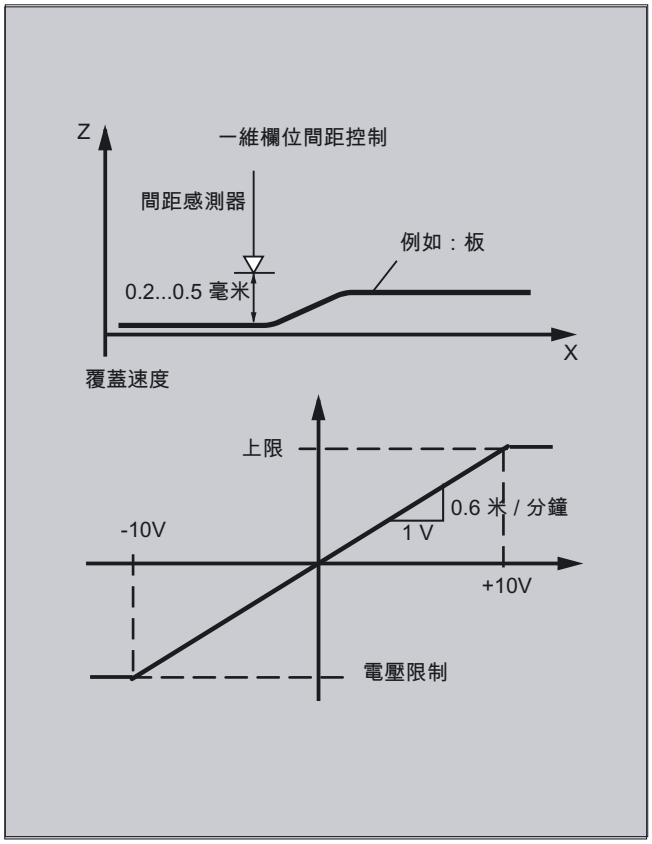
說明

SINUMERIK 828D 並未提供此功能！

功能

間隙值進行積分計算的同時，將一併完成臨界值範圍檢查：

$AA_OFF_MODE = 1



注意
更高層級之控制迴圈的迴圈增益取決於插補循環的設定。 解決方式： 讀入 IPO 時脈循環的 MD 並將之列入考量。

說明

利用 MD 32020 限制疊覆插補器的速度： JOG_VELO 的 Ipo 時脈循環為 12 毫秒。速度公式：

$$\frac{0.120mm}{0.6ms}/mV = 0.6\frac{m}{\min}/V$$

範例

副程式 “AON”： 間隙控制開啟

程式碼	註解
PROC AON	
$AA_OFF_LIMIT[Z]=1	; 指定臨界值。
FCTDEF（1，-10，+10，0，0.6，0.12）	; 多項式定義
ID=1 DO SYNFCT（1，$AA_OFF[Z]，$A_INA[3]）	; 間隙控制為生效狀態。
ID=2 WHENEVER $AA_OFF_LIMIT[Z]<>0 DO $AA_OVR[X] = 0	; 超過臨界值範圍時，禁止 X 軸。
RET	
ENDPROC	

副程式 “AOFF”： 間隙控制關閉

程式碼	註解
PROC AOFF	
CANCEL（1）	; 刪除同步動作，間隙控制
CANCEL（2）	; 取消臨界值範圍檢查
RET	
ENDPROC	

主程式 “MAIN”

程式碼	註解
AON	; 間隙控制開啟
...	
G1 X100 F1000	
AOFF	; 間隙控制關閉
M30	

其它資訊

基本座標系統的位置偏移

利用系統變數$AA_OFF[軸]，您可疊覆通道中任一軸的動作。它的作用如同基本座標系統的位置偏移。

不論程式是否正在移動軸，該軸中以此方式程式設計的位置偏移將立即被覆蓋。

限制主要執行變數輸出：

您可將欲進行修正的絕對值（主要執行變數輸出）限制在設定資料 SD43350 $SA_AA_OFF_LIMIT 中所儲存的值以內。

間隙疊覆類型可利用機械參數 MD36750 $MA_AA_OFF_MODE 加以定義：

值	意義
0	比例評估
1	積分評估

利用系統變數$AA_OFF_LIMIT[軸]，便可根據方向查詢修正值是否在臨界值範圍以內。您可從同步動作中斷此系統變數，此外，當達到臨界值時亦可使用此變數來（例如）停止軸或設定警報。

0: 偏移值不在範圍內

1 已達到正向的偏移臨界值

-1: 已達到負向的偏移臨界值

10.4.9 線上刀具偏移（FTOC）

功能

若有一多項式是根據參考值（例如可以是某軸的實際值）以 FCTDEF 進行程式設計，則 FTOC 允許在此多項式之後的幾何軸覆蓋動作。

係數 $_0$，屬於函數定義 FCTDEF（...），可用於評估 FTOC。上臨界值與下臨界值與 $_0$ 相依。

FTOC 可用於程式模態線上刀具偏移量，或作為同步動作的間隙控制。

該函數可讓您在同一通道或不同通道（加工與修飾通道）中加工工件及修飾研磨輪。

修飾研磨輪的一般條件與規格，套用FTOC，用和以PUTFTOCF套用線上刀具偏移一樣的方式（請參考"線上刀具偏移（PUTFTOCF、FCTDEF、PUTFTOC、FTOCON、FTOCOF） (頁 358)"）。

句法

```
FCTDEF（<函數>，<LLimit>，<ULimit>，<a0>，<a1>，<a2>，<a3>）
FTOC（<函數>，<參考值>，<刀具參數>，<通道>，<主軸>）
...
```

意義

FCTDEF: FCTDEF 係用來定義 FTOC 的多項式函數
參數:
<函數>: 多項式函數的編碼
類型: INT
值域: 1 到 3
<LLimit>: 下限值
類型: REAL
<ULimit>: 上限值
類型: REAL
<a0>至<a3>: 多項式函數的係數
類型: REAL

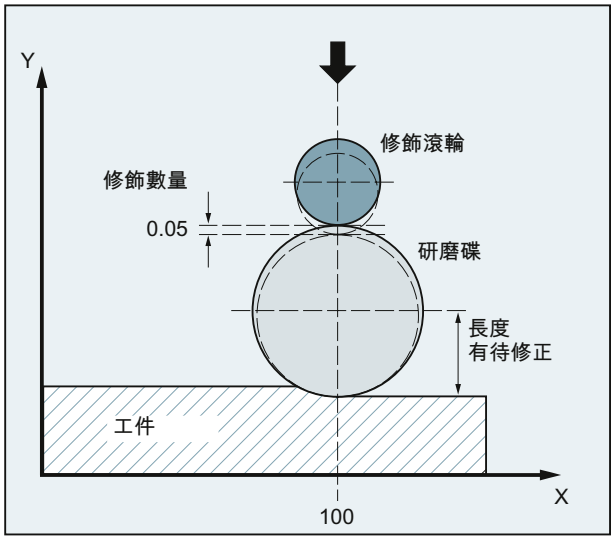
DO FTOC: 執行"線上刀具偏移量的連續非模態寫入"函數
參數:
<函數>: 多項式函數的編碼
類型: INT
值域: 1 到 3
注意:
必須符合用於 FCTDEF 的設定。
<參考值>: 用於各函數值的主執行變數，要被用以 FCTDEF 定義的多項式函數來計算。
類型: VAR REAL
<刀具參數: 偏移值要被加入的磨耗參數（長度 1，2 或 3）之號碼。
類型: INT
<通道>: 線上刀具偏移量發生影響的通道號碼。
類型: INT
注意:
若偏移量沒有在有效通道中產生效果，則只需指定通道號碼。
<主軸>: 線上刀具偏移量發生影響的主軸號碼。
類型: INT
注意:
若偏移量要被套用至無效的研磨輪，而不是正在使用中的有效刀具，則只需指定主軸號碼。

說明

FTOCON 必須在目標通道中啟動。

範例

有效研磨輪的長度有待補償。



程式碼	註解
FCTDEF（1，-1000，1000，-$AA_IW[V]，1）	; 函數定義。
ID=1 DO FTOC（1，$AA_IW[V]，3，1）	; 選擇線上刀具偏移：V 軸實際值是多項式 1 的輸入值。計算結果加入至通道 1 中作為現用研磨盤長度 3 的補正值。
WAITM（1，1，2）	; 與加工通道同步。
G1 V-0.05 F0.01 G91	; 進行修飾的進給動作。
G1 V-0.05 F0.02	
...	
CANCEL（1）	; 取消選擇線上偏移。
...	

10.4.10 線上刀長補正（$AA_TOFF）

功能

利用系統變數$AA_TOFF[ ]，從三度空間依照刀具三個方向即時覆蓋有效刀長。
這三個幾何軸識別碼當作索引。因此，偏移現用方向編號是由同時啟用之幾何軸決定。
可同時主動所有偏移。

句法

```
N..TRAORI
N..TOFFON（X，25）
N..WHEN TRUE DO $AA_TOFF[X]
N..TOFFON（Y，25）
N..WHEN TRUE DO $AA_TOFF[Y]
N..TOFFON（Z，25）
N..WHEN TRUE DO $AA_TOFF[Z]
```

意義

TOFFON	**刀具偏移 ON**（啟動線上刀長偏移） 啟動時，可指定相關偏移方向的偏移值，但此值立刻就會再重新取得。
TOFFOF	**刀具偏移 OFF**（重置線上刀長偏移） 重置相關偏移值並開始執行前置處理停止。
X，Y，Z	指定給 TOFFON 的偏移值補正方向
$AA_TOFF[X]=值	X 軸方向疊覆
$AA_TOFF[Y]=值	Y 軸方向疊覆
$AA_TOFF[Z]=值	Z 軸方向疊覆

範例

範例 1：選擇刀長補正

程式碼	註解
N10 TRAORI（1）	; 轉換開啟。
N20 TOFFON（Z）	; 啟用 Z 軸刀具方向的線上刀長補正。
N30 WHEN TRUE DO $AA_TOFF[Z]=10 G4 F5	; Z 軸刀具方向所插補的 TLC（刀長補正）為 10。
N40 TOFFON（X）	; 啟用 X 軸刀具方向的線上刀長補正。
N50 ID=1 DO $AA_TOFF[X]=$AA_IW[X2] G4 F5	; X 軸刀具方向補正是根據 X2 軸位置而建置。
...	
	; 指派 X 軸方向的實際補正。X 軸刀具方向的 TLC 減為 0：
N100 XOFFSET=$AA_TOFF_VAL[X] N120 TOFFON（X，-XOFFSET） G4 F5	

範例 2：取消選擇刀長補正

程式碼	註解
N10 TRAORI（1）	; 轉換開啟。
N20 TOFFON（X）	; 啟動 Z 軸刀具方向。
N30 WHEN TRUE DO $AA_TOFF[X] = 10 G4 F5	; X 軸刀具方向所插補的 TLC（刀長補正）為 10。
...	
N80 TOFFOF（X）	; 刪除 X 軸刀具方向的位置偏移：...$AA_TOFF[X]=0 未移動任何軸，將對應實際方向的 n 位置偏移加入至工件座標系統（WCS）的實際位置。

10.4.11 定位移動

功能

對於工件程式，您可從同步動作將軸完全定位成非同步。針對高度依賴事件的週期程序或操作，為其從同步動作程式設計軸定位，是一項明智的作法。從同步動作程式設計的軸就稱為**指令軸**。

程式設計

參考：
/PG/ 基礎程式設計指南；“路徑詳細資訊”一節
/FBSY/ 函數說明，同步動作；“啟動命令軸”

參數

同步動作中的定位作業量測系統是以 G 代碼 G70/G71/G700/G710 加以指定。

藉由在同步動作中程式設計 G 系列功能，便可定義與工件程式內容無關的同步動作 INCH/METRIC 評估。

10.4.12 定位軸（POS）

功能

定位軸的動作對工件程式之執行毫無影響，這與您從工件程式進行程式設計時不同。

句法

POS[軸]=值

意義

DO POS	啟動 / 定位指令軸
軸	欲移動之軸的名稱
值	欲移動之距離值（視移動模式而定）

範例

範例 1：

程式碼	註解
ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100	; U 軸移動，根據移動模式，連續遞增 100（英吋 / 毫米）或從"控制零點"移動至位置 100（英吋 / 毫米）。
	; 移動 U 軸，其移動距離是從主要執行變數計算而來：
ID=1 EVERY $AA_IM[B]>75 DO POS[U]=$AA_MW[V]-$AA_IM[W]+13.5	

範例 2：

程式環境影響定位軸的定位移動
（同步動作的動作元件中不含 G 函數）：

程式碼	註解
N100 R1=0	
N110 G0 X0 Z0	
N120 WAITP（X）	
N130 ID=1 WHENEVER $R==1 DO POS[X]=10	
N140 R1=1	
N150 G71 Z10 F10	; Z=10mm X=10mm
N160 G70 Z10 F10	; Z=254mm X=254mm
N170 G71 Z10 F10	; Z=10mm X=10mm
N180 M30	

同步動作之動作元件中的 G71 清楚定義了定位軸（公制）的定位移動與程式環境無關：

程式碼	註解
N100 R1=0	
N110 G0 X0 Z0	
N120 WAITP（X）	
N130 ID=1 WHENEVER $R==1 DO G71 POS[X]=10	
N140 R1=1	
N150 G71 Z10 F10	; Z=10mm X=10mm
N160 G70 Z10 F10	; Z=254mm X=10mm（X 軸永遠定位至 10mm）
N170 G71 Z10 F10	; Z=10mm X=10mm
N180 M30	

如果不在單節開始啟動軸動作，那麼從同步動作一直到所要求的啟動時刻，軸手動倍率可維持為 0：

程式碼	註解
WHENEVER $A_IN[1]==0 DO $AA_OVR[W]=0 G01 X10 Y25 F750 POS[W]=1500 FA=1000	; 只要數位輸入 1=0，則定位軸保持不動。

10.4.13 位置位於指定參考範圍內（POSRANGE）

功能

POSRANGE（ ）函數可用來判斷軸之目前插補設定點位置是否位於指定參考位置所在的視窗內。位置規格可參照座標系統，而座標系統亦可另行指定。

查詢模組軸的實際軸位置時，將考量模組偏移。

說明

此函數僅可從同步動作呼叫。如果從工件程式呼件它，將觸發警報 14091 %1 單節%2，不允許此函數，索引：呼叫了索引 5 的%3。

句法

```
BOOL POSRANGE（Axis，Refpos，Winlimit，[Coord]）
```

意義

BOOL POSRANGE	指令軸目前位置位於指定參考位置的視窗內。
AXIS <軸>	機械軸、通道軸或幾何軸的軸識別碼
REAL Refpos	Coord 所指定之座標系統的參考位置
REAL Winlimit	形成位置視窗臨界值的數
INT Coord	機械座標系統（MCS）為啟用（之選項）。您可執行下列動作： 0 代表 MCS（機械座標系統） 1 代表 BCS（基本座標系統） 2 代表 SZS（可設定零點系統） 3 代表 WCS（工件座標系統）

函數值

目前設定點取決於指定之座標系統的位置詳細資訊

函數值：TRUE	如果 Refpos（Coord） - abs（Winlimit） ≤ Actpos（Coord） ≤ Refpos（Coord） + abs（Winlimit）
函數值：FALSE	否則

10.4.14 啟動 / 停止軸（MOV）

功能

利用 MOV[軸]=值，便可直接啟動指令軸，不需指定結束位置。在其他動作或定位指令設定另一項移動或者您利用停止指令停止軸以前，軸將沿已程式設計之方向移動。

句法

```
MOV[軸] = 值
```

意義

DO MOV	啟動命令軸動作
軸	欲啟動之軸的名稱
值	移動 / 停止動作的啟動指令。 正負號將決定動作方向。 值的資料類型是 INTEGER。
Value >0 （usually +1）	正向位置
Value <0 （usually -1）	負向
Value ==0	停止軸動作

說明

如果利用 MOV[軸]=0 將索引軸停止，則該軸靜止在下一個索引位置上。

範例

程式碼	註解
... DO MOV[U]=0	; 停止 U 軸

10.4.15 軸更換（RELEASE，GET）

功能

關於換刀，利用 GET（軸）可要求對應之指令軸作為同步動作中的一個動作。利用 $AA_AXCHANGE_TYPE 系統變數，可查詢指派給此通道的軸類型以及因此連結上的插補權利。根據實際狀況以及擁有此軸目前插補權利之通道，可能會有不同流程。

一旦完成換刀後，可利用 RELEASE（軸）將此指令軸釋出給通道作為同步動作中的一個動作。

機械製造商

相關軸必須經由機械參數指派給通道。請參閱機器製造商說明書。

句法

GET（軸[，軸{，...}]）取得軸
RELEASE（軸[，軸{，...}]）釋出軸

意義

DO RELEASE	將軸釋出作為中立軸
DO GET	取得軸以便進行軸更換
軸	欲啟動之軸的名稱

範例：軸更換的程式順序，有兩個通道

Z 軸已宣告在第一及第二通道中。

第一通道中的程式順序：

程式碼	註解
WHEN TRUE DO RELEASE（Z）	; Z 軸成為中立軸
WHENEVER（$AA_TYP[Z]==1） DO RDISABLE	; 如果 Z 軸是程式軸則禁止讀入
N110 G4 F0.1	
WHEN TRUE DO GET（Z）	; Z 軸再次成為 NC 程式軸
WHENEVER（$AA_TYP[Z]<>1） DO RDISABLE	; 在 Z 軸成為程式軸之前禁止讀入
N120 G4 F0.1	
WHEN TRUE DO RELEASE（Z）	; Z 軸成為中立軸
WHENEVER（$AA_TYP[Z]==1） DO RDISABLE	; 如果 Z 軸是程式軸則禁止讀入
N130 G4 F0.1	;
N140 START（2）	; 啟動第二通道

第二通道中的程式順序：

程式碼	註解
WHEN TRUE DO GET（Z）	; ；將 Z 軸移至第二通道
WHENEVER（$AA_TYP[Z]==0） DO RDISABLE	; ；如果 Z 軸位於其他通道，則停用讀入 ;
N210 G4 F0.1	
WHEN TRUE DO GET（Z）	; ；Z 軸是 NC 程式軸
WHENEVER（$AA_TYP[Z]<>1） DO RDISABLE	; ；在 Z 軸成為程式軸之前停用讀入
N220 G4 F0.1	
WHEN TRUE DO RELEASE（Z）	; ；第二通道中的 Z 軸是中立軸
WHENEVER（$AA_TYP[Z]==1） DO RDISABLE	; ；如果 Z 軸是程式軸則停用讀入
N230 G4 F0.1	
N250 WAITM（10，1，2）	; 與通道 1 同步

第一通道中的程式順序繼續：

程式碼	註解
N150 WAIM（10，1，2）	; 與通道 2 同步
WHEN TRUE DO GET（Z）	; 將 Z 軸移至此通道
WHENEVER（$AA_TYP[Z]==0） DO RDISABLE	; 如果 Z 軸位於另一通道，則禁止讀入
N160 G4 F0.1	
N199 WAITE（2）	
N999 M30	; 等待通道 2 中的程式結束

範例：技術循環中的軸更換

U 軸：U （$MA_AUTO_GET_TYPE=2）已宣告在第一及第二通道中，通道 1 目前擁有插補權利。以下技術循環是在通道 2 中啟動：

程式碼	註解
GET（U）	; 將 U 軸擷取至通道
POS[U]=100	; U 軸應移至 位置 100

必須等到 U 軸移至通道 2 後，才會執行 POS[U]指令軸動作列。

順序

啟動 GET （軸）動作時所要求的軸可以透過系統變數（$AA_AXCHANGE_TYP[<軸>]讀取它的軸更換之軸類型：

- 0: 指派給 NC 程式的軸
- 1: 指派給 PLC 的軸或是當作指令軸或震盪軸使用的軸
- 2: 另一通道擁有插補權利
- 3: 軸為中立軸
- 4: 中立軸受控於 PLC
- 5: 另一通道擁有插補權利，NC 程式要求使用的軸
- 6: 另一通道擁有插補權利，要求當作中立軸使用的軸
- 7: 用於 PLC 之有效軸或是當作指令軸或震盪軸使用的軸，PLC 程式要求使用的軸
- 8: 用於 PLC 之有效軸或是當作指令軸或震盪軸使用的軸，要求當作中立軸使用的軸

界限條件

相關軸必須經由機械參數指派給通道。

僅受控制 PLC 的軸無法指派給 NC 程式。

參考：
/FB2/ 功能手冊，延伸功能；定位軸（P2）

利用 GET 向另一通道要求取得軸

啟動 GET 動作時，如果**授權另一通道寫入**至軸（$AA_AXCHANGE_TYP[<軸>] == 2）（表示該通道擁有插補權利），則"軸更換"函數是用來從此通道取得該軸（$AA_AXCHANGE_TYP[<軸>]==6），並儘快將之指派給提出要求的通道。

於是該軸成為中立軸（$AA_AXCHANGE_TYP[<軸>]==3）。

提出要求的通道不會進行重新組織。

指派**作為 NC 程式軸並進行重新組織：**

啟動 GET 動作時，如果作業進行中已嘗試讓軸成為中立軸（$AA_AXCHANGE_TYP[<軸>]==6），則 NC 程式將要求使用該軸（$AA_AXCHANGE_TYP[<axis>]==5），此軸會儘快指派給通道上的 NC 程式（$AA_AXCHANGE_TYP[<軸>]==0）。

軸已指派給接受要求的通道

指派**作為 NC 程式軸並進行重新組織：**

如果啟動之時接受要求的軸已經指派給提出要求的通道，且該軸狀態是已成為中立軸（不受控於 PLC）（$AA_AXCHANGE_TYP[<軸>]==3），那麼它將指派給 NC 程式（$AA_AXCHANGE_TYP[<軸>]==0）。

軸處於受 PLC 控制之中立軸狀態

如果軸處於受 PLC 控制之中立軸狀態（$AA_AXCHANGE_TYP[<軸>]==4），則將要求此軸作為中立軸（$AA_AXCHANGE_TYP[<軸>] == 8）。根據 MD 10722 中位元 0 的值，此項動作將鎖定用於通道間自動軸更換的軸：AXCHANGE_MASK （bit 0 == 0）.這等同於（$AA_AXCHANGE_STAT[<軸>] == 1）。

軸啟用作為中立指令軸 / 震盪軸，或者指派給 PLC

如果軸啟用作為指令軸 / 震盪軸，或者指派給 PLC 以便進行移動，則 PLC 軸==同步定位軸（$AA_AXCHANGE_TYP[<軸>]==1），將要求此軸作為中立軸（$AA_AXCHANGE_TYP[<軸>] == 8）。根據 MD 10722 中位元 0 的值，此項動作將鎖定用於通道間自動軸更換的軸：AXCHANGE_MASK （bit 0 == 0）.這等同於（$AA_AXCHANGE_STAT[<軸>] == 1）。

新的 GET 動作將要求此軸供 NC 程式使用（$AA_AXCHANGE_TYP[<軸>]變更為== 7）。

已指派給 NC 程式的軸

如果軸已指派給 NC 程式（$AA_AXCHANGE_TYP[<軸>]==0）或者如果是要求指派（例如 NC 程式觸發軸更換）（$AA_AXCHANGE_TYP[<軸>]==5 或$AA_AXCHANGE_TYP[<軸>] == 7），則狀態將不變。

10.4.16 軸進給（FA）

功能

指令軸的軸進給是採模態運作。

句法

```
FA[<軸>]=<值>
```

範例

程式碼	註解
`ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100 FA[U]=990`	; 輸入固定進給率值。
	; 從主要執行變數產生進給率值：
`ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100 FA[U]=$AA_VACTM[W]+100`	

10.4.17 軟體極限開關

功能

根據設定資料$SA_WORKAREA_PLUS_ENABLE，指令軸將考量以 G25/G26 進行程式設計之工作區限制。

在工件程式中以 G 函數 WALIMON/WALIMOF 開啟或關閉工作區限制，將對指令軸毫無影響。

10.4.18 軸協調

功能

一般而言，軸不是從工件程式移動就是從同步動作做為定位軸。

如果同一軸交替從工件程式當作路徑 / 定位軸移動以及從同步動作進行移動，如此一來，兩個軸動作彼此間產生“協調傳輸”。

如果後續又從工件程式移動指令軸，則必須重新組織前置處理。這將輪流導致工件程式處理中斷以及前置處理停止。

交替從工件程式以及從同步動作移動 X 軸之範例

程式碼	註解
N10 G01 **X100** Y200 F1000	; 在工件程式中程式設計 X 軸
…	
N20 ID=1 WHEN $A_IN[1]==1 DO	; 如有數位輸入，則從同步動作啟動定位
POS[X]=150 FA[X]=200	
…	
CANCEL（1）	; 選擇同步動作
…	
N100 G01 **X240** Y200 F1000	; X 軸成為路徑軸； 如果數位輸入為 1 且從同步動作定位 X 軸，則在動作之前，會因為軸傳輸而發生延遲。

範例：變更同一軸之移動指令

程式碼	註解
ID=1 EVERY $A_IN[1]>=1 DO POS[V]=100 FA[V]=560	; 如果數位>= 1，從同步動作啟動定位
ID=2 EVERY $A_IN[2]>=1 DO POS[V]=$AA_IM[V] FA[V]=790	; 移動期間，即時追蹤軸軌（已設定第二輸入，也就是兩個同步動作同時啟用時的 V 軸結束位置與進給率） 。

10.4.19 設定實際值（PRESETON）

功能

執行 PRESETON（軸，值）時，並未變更目前的軸位置，而是為其指派新值。

可為下列項目程式設計同步動作之 PRESETON

- 由工件程式啟動的模數旋轉軸及
- 所有由同步動作啟動的指令軸

句法

```
DO PRESETON（axis，value）
```

意義

DO PRESETON	在同步動作中設定實際值
軸	將變更其控制零點的軸
值	控制零點將改成的數值

軸之限制

PRESETON 無法程式設計參與轉換的軸。

無法同時由工件程式及同步動作移動同一個軸。因此，若已先在同步動作中對軸程式設計，則從工件程式中程式設計同一個軸時會發生延遲現象。

若該軸係由兩者交替使用，則會對兩軸間的移動傳輸設定座標。為此，必須中斷工件程式的執行。

範例

移動軸的控制零點

程式碼	註解
`WHEN $AA_IM[a] >= 89.5 DO PRESETON（a4，10.5）`	; 將軸 a 的控制零點朝 軸的正方向移動 10.5 個長度單位（英吋或毫米）

10.4.20 主軸動作

功能

可在同步動作中，將主軸定位在與工件程式完全不同步的位置。建議對循環順序或與事件強烈相關的操作進行此類程式設計。

若同時啟用同步動作對主軸發出衝突指令，會優先選配最近的主軸指令。

啟動 / 停止 / 定位主軸之範例

程式碼	註解
`ID=1 EVERY $A_IN[1]==1 DO M3 S1000`	; 設定旋轉方向及速度
`ID=2 EVERY $A_IN[2]==1 DO SPOS=270`	; 定位主軸

設定主軸之方向及旋轉 / 定位速度之範例

程式碼	註解
`ID=1 EVERY $A_IN[1]==1 DO M3 S300`	; 設定旋轉方向及速度
`ID=2 EVERY $A_IN[2]==1 DO M4 S500`	; 指定新的旋轉方向及速度
`ID=3 EVERY $A_IN[3]==1 DO S1000`	; 指定新的速度
`ID=4 EVERY （$A_IN[4]==1） AND （$A_IN[1]==0） DO SPOS=0`	; 定位主軸

10.4.21 耦合動作（TRAILON、TRAILOF）

功能

由同步動作啟動耦合時，先導軸可以移動。於此情況下，跟隨軸的速率會提高到設定速率。先導軸進行速率同步時的位置，即為耦合軸動作的起始位置。耦合軸動作的函數性於“路徑移動行為”一節中說明。

句法

開始耦合動作

```
DO TRAILON（following axis，leading axis，coupling factor）
```

停用耦合軸運作

```
DO TRAILOF （following axis，leading axis，leading axis 2）
```

意義

啟動非同步耦合動作：

`... DO TRAILON（FA，LA，Kf）`	其中： FA：跟隨軸 LA：先導軸 Kf：耦合係數

停用非同步耦合動作：

`... DO TRAILOF（FA，LA，LA2）`	其中： FA：跟隨軸 LA：先導軸，選配 LA2：先導軸 2，選配
`... DO TRAILOF（FA）`	停止進行所有與跟隨軸進行的耦合。

範例

程式碼	註解
`$A_IN[1]==0 DO TRAILON（Y，V，1）`	; 若 數位輸入為 1，則啟用第一組耦合動作群組
`$A_IN[2]==0 DO TRAILON（Z，W，-1）`	; 啟用第二組耦合軸群組
`G0 Z10`	; 相反軸方向之 Z 與 W 軸的進給
`G0 Y20`	; 相同軸 方向之 Y 及 V 軸的進給
`...`	
`G1 Y22 V25`	; 相互重疊及分開；耦合動作軸“V”之移動
`...`	
`TRAILOF （Y，V）`	; 停用第一組耦合軸群組
`TRAILOF （Z，W）`	; 停用第二組耦合軸群組

使用 TRAILOF 防止衝突之範例

使用軸的 TRAILOF 函數可再次釋放耦合軸，令其成為可供存取之通道軸。需確保在通道請求有關的軸之前，執行 TRAILOF。但在本例中並非如此

```
...
N50 WHEN TRUE DO TRAILOF（Y，X）
N60 Y100
...
```

在本例中，因為非模態同步動作已使用 N60 與 TRAILOF 同步啟用，所以軸並未及早釋放，請參閱動作同步動作的“結構，基本資訊”一節。
為避免發生衝突，請務必遵守

以下步驟。

```
...
N50 WHEN TRUE DO TRAILOF（Y，X）
N55 WAITP（Y）
N60 Y100
```

10.4.22 先導值耦合（LEADON、LEADOF）

說明

SINUMERIK 828D 並未提供此功能！

功能

可在同步動作中不受限地程式設計軸的先導值耦合。 只有在同步動作中，才可不先進行再同步動作而變更現有耦合的曲線表。

句法

啟動主動值耦合

```
DO LEADON （following axis，leading axis，curve table no.，OVW）
```

停用先導值耦合

```
DO LEADOF（following axis，leading axis，leading axis 2）
```

意義

啟動軸向先導值耦合：	
`...DO LEADON（FA，LA，NR，OVW）`	其中： FA： 跟隨軸 LA： 先導軸 NR： 已儲存的曲線表之編號 OVW： 可使用變更後的曲線表覆寫現有耦合
停用軸向先導值耦合：	
`...DO LEADOF（FA，LA）`	使用： FA：跟隨軸 LA： 先導軸，選配
`... DO LEADOF（FA）`	未指定先導軸之簡短格式

利用同步動作 RELEASE 啟動存取

使用待耦合之軸的 RELEASE 函數可釋放該軸，以供同步動作存取。

範例：

```
RELEASE （XKAN）
ID=1 every SR1==1 to LEADON（CACH，XKAN，1）
```

OVW=0（預設值））

如未進行再同步，將無法為現有的耦合指定新的曲線圖。 若要更改曲線表，必須先停用現有耦合，並使用變更後的曲線表編號重新啟動。 讓耦合進行再同步。

使用 OVW=1 來變更現有耦合的曲線表

可使用 OVW=1 為現有耦合指定新的曲線表。 不用進行再同步。 跟隨軸會盡快跟上新曲線表中指定的位置值。

即時切割範例

必須將持續通過切削刀具操作區的擠壓材料切成等分長度。

X 軸： 擠壓材料移動的軸。 工件座標系統（WCS）
X1 軸： 擠壓材料的機械軸，機械座標系統（MCS）
Y 軸： 切削刀具"輾過"擠壓材料的軸

假設為切削刀具的進給及控制係由 PLC 控制。 可藉由評估 PLC 介面上的訊號，以判斷擠壓材料與切削刀具是否同步。

動作
啟動耦合，LEADON
停用耦合，LEADOF
設定實際值，PRESETON

程式碼	註解
N100 R3=1500	; 待切割之工件長度
N200 R2=100000 R13=R2/300	
N300 R4=100000	
N400 R6=30	; Y 軸起始位置
N500 R1=1	; 輸送軸啟動條件
N600 LEADOF（Y，X）	; 刪除現有一切耦合
N700 CTABDEF（Y，X，1，0）	; 表之定義
N800 X=30 Y=30	; 數值對
N900 X=R13 Y=R13	
N1000 X=2*R13 Y=30	
N1100 CTABEND	; 結束表定義
N1200 PRESETON（X1，0）	; 於開始時 PRESET
N1300 Y=R6 G0	; Y 軸起始位置，該軸為線性
N1400 ID=1 WHENEVER $AA_IW[X]>$R3 DO PESETON（X1，0）	; 長度 R3 通過後 PRESET，中斷連線後重新開始
N1500 RELEASE（Y）	
N1800 ID=6 EVERY $AA_IM[X]<10 DO LEADON（Y，X，1）	; 在 X < 10 時，藉由表 1 耦合 Y 至 X
N1900 ID=10 EVERY $AA_IM[X]>$R3-30 DO EADOF（Y，X）	; > 30 則在移動至切割距離前，停用耦合
N2000 WAITP（X）	
N2100 ID=7 WHEN $R1==1 DO MOV[X]=1 FA[X]=$R4	; 持續設定動作中的擠壓材料軸
N2200 M30	

10.4.23 量測（MEAWA、MEAC）

功能

相對於在工件程式的移動單節中使用，量測函數可視需要啟動及停用。
如需有關量測的其他資訊，請參閱特殊動作指令“延伸量測函數”

句法

不含刪除之剩餘距離的軸量測
`MEAWA[axis]=（mode，trigger_event_1，..._4）`
不刪除剩餘距離的持續量測
`MEAC[axis]=（mode，measurement_memory，trigger_event_1，..._4）`

意義

程式碼		註解	
`DO MEAWA`	;	開始軸量測	
`DO MEAC`	;	開始持續量測	
軸	;	將量測之軸名稱	
模式	;	十進位 0 規格：啟用量測系統 量測系統的編號（取決於模式） 1：1. 量測系統 2：2. 量測系統 3：兩個量測系統皆使用	十**位數** 0 規格：取消量測工作 最多可啟用 4 個觸發事件 1：同時 2：接續 3：與 2 相同，但在啟動時並不監控觸發事件 1

`Trigger_event_1`
`to`
`trigger_event_4`
	;	**:** 上升稜邊，探針
		-1：下降稜邊，探針 1 選配
		2：上升稜邊，探針 2 選配
		-2：下降稜邊，探針 2 選配	

量測記憶體	;	FIFO 循環儲存器的編號	

10.4.24 陣列變數初始化（SET、REP）

功能

在同步動作中，可使用特定數值來初始化或說明陣列變數。

說明

僅可使用可在同步動作中說明的變數。因此無法將機械參數初始化。無法利用 NO_AXIS 值指定軸變數。

句法

```
DO ARRAY[n，m]=SET（<值 1>，<值 2>，...）
DO ARRAY[n，m]=REP（<值>）
```

意義

ARRAY[n，m]	已程式設計的陣列索引集 於已程式設計的陣列索引值開始初始化。在二維陣列中，會先遞增第二索引值。並非以軸索引進行。
SET（<值 1>，<值 2>，...）	以值清單進行初始化 陣列由已程式設計的陣列索引值開始，使用 SET 參數說明。程式設計眾多被指派為值的陣列元素。若程式設計的值數超出剩餘陣列元素中的值，將觸發警報。
REP（<值>）	以相同值進行初始化 自已程式設計的陣列索引到陣列結尾間說明陣列，並重複使用 REP 的參數（<值>）。

範例

程式碼	註解
WHEN TRUE DO SYG_IS[0]=REP（0）	; 結果：
WHEN TRUE DO SYG_IS[1]=SET（3，4，5）	; SYG_IS[0]=0
	SYG_IS[1]=3
	SYG_IS[2]=4
	SYG_IS[3]=5
	SYG_IS[4]=0

10.4.25 設定 / 刪除等待標記（SETM、CLEARM）

功能

可在同步動作中設定或刪除等待標記，例如，為了協調各通道。

句法

```
DO SETM（<標記編號>）
DO CLEARM（<標記編號>）
```

意義

SETM	設定通道等待標記的指令 可程式設計 SETM 指令至工件程式及同步動作的動作元件中。該指令會設定其套用的通道標記（<標記編號>）。
CLEARM	刪除通道等待標記的指令 可程式設計 CLEARM 指令至工件程式及同步動作的動作元件。該指令會刪除其套用的通道標記（<標記編號>）。
<標記編號>	等待標記

10.4.26 錯誤回應（SETAL）

功能

可使用同步動作程式設計錯誤回應。狀態變數接受查詢，相應動作開始。
可能的錯誤狀況回應為：

- 停止軸（override=0）
- 設定警報
  SETAL 可在同步動作中設定循環警報。
- 設定輸出
- 同步動作中所有可能的動作

句法

設定循環警報：

```
DO SETAL（<警報編號>）
```

意義

SETAL	設定循環警報之指令	
<警報編號>	警報編號	
	使用者可使用的循環警報範圍：	65000 ... 69999

範例

程式碼	註解
`ID=67 WHENEVER （$AA_IM[X1]-$AA_IM[X2]）<4.567 DO $AA_OVR[X2]=0`	; 若 X1 軸及 X2 軸間的安全間隙過小，則停止 X2 軸。
`ID=67 WHENEVER （$AA_IM[X1]-$AA_IM[X2]）<4.567 DO SETAL（65000）`	; 若 X1 軸及 X2 軸間的安全間距過小，則將警報設為 65000。

10.4.27 移動至固定停止點（FXS、FXST、FXSW、FOCON、FOCOF）

功能

使用工件程式的 FXS、FXST 及同步動作 / 技術循環中的 FXSW 指令程式設計函數"移動至固定停止點"的指令。

可不移動便啟用函數，則扭矩會立即受限。一旦軸透過設定點移動，便會啟動限制停止監控器。

以有限的扭矩 / 力量（FOC）移動

本函數可在同步動作中隨時變更扭矩 / 力量，並可按模態啟動或依單節啟動。

句法

```
FXS[<軸>]
FXST[<軸>]
FXSW[<軸>]
FOCON[<軸>]
FOCOF[<軸>]
```

意義

FXS	僅可在配有數位驅動（FDD、MSD、HLA）的系統中選取
FXST	變更鉗緊扭矩 FXST
FXSW	變更監控視窗 FXSW
FOCON	啟動模態有效的扭矩 / 力量限制
FOCOF	停用扭矩 / 力量限制
<軸>	軸定義碼 您可以執行下列動作： • 幾何軸識別碼 • 通道軸識別碼 • 機械軸定義碼

說明

每個選項只能執行一次。

範例

範例 1：移動至固定停止點（FXS），用同步動作開始

程式碼	註解
Y 軸：	; 靜態同步動作
啟動：	
N10 IDS=1 WHENEVER （（$R1==1） AND $AA_FXS[y]==0））	D $R1=0 FXS[Y]=1 FXST[Y]=10 FA[Y]=200 POS[Y]=150
	; 設定$R1=1 可啟動 Y 軸的 FXS，有效扭矩會降為 10%，並開始朝固定停止點移動。
N11 IDS=2 WHENEVER （$AA_FXS[Y]==4） DO FXST[Y]=30	; 一旦偵測到固定停止點（$AA_FXS[Y]==4），扭矩會降為 30%。
N12 IDS=3 WHENEVER （$AA_FXS[Y]==1） DO FXST[Y]=$R0	; 抵達固定停止點後，將依照 R0 控制扭矩。
N13 IDS=4 WHENEVER （（$R3==1） AND $AA_FXS[Y]==1））	DO FXS[Y]=0 FA[Y]=1000 POS[Y]=0
	; 依照 R3 取消選擇並回退
N20 FXS[Y]=0 G0 G90 X0 Y0	; 一般程式執行：
N30 RELEASE（Y）	; 釋放 Y 軸，以在同步動作中進行動作。
N40 G1 F1000 X100	; 其他軸的動作。
N50 ...	
N60 GET（Y）	; 包括 Y 軸在內，返回路徑群組。

範例 2：啟動扭矩 / 力量限制（FOC）

程式碼	註解
N10 FOCON[X]	; 限制模態啟用。
N20 X100 Y200 FXST[X]=15	; 以降低的扭矩（15%）移動 X。
N30 FXST[X]=75 X20	; 扭矩變更至 75%，以有限扭矩移動 X。
N40 FOCOF[X]	; 關閉扭矩限制。

其它資訊

多重選取

若因程式設計錯誤，導致啟用後（FXS[軸] = 1））再次呼叫本函數，將輸出以下警報：

警報 20092“仍啟動移動至固定停止點”

進行程式設計時，在輸入$AA_FXS[ ]或該函數專屬的位元記憶體（在此為 R1）的條件下，可避免多次啟動“工件程式區段”函數：

程式碼

```
N10 R1=0
N20 IDS=1 WHENEVER （$R1==0 AND
$AA_IW[AX3] > 7） DO R1=1 FXST[AX1]=12
```

單節相關式同步動作

透過程式設計單節相關式同步動作，可在逼近動作中連接移動至固定停止點之函數。
範例：

程式碼	註解
N10 G0 G90 X0 Y0	
N20 WHEN $AA_IW[X] > 17 DO FXS[X]=1	; 若 X 的到達位置超過 17 毫米則啟動 FXS。
N30 G1 F200 X100 Y110	

靜態及單節相關式同步動作

在靜態及單節相關式同步動作中，可如同一般執行工件程式般，使用相同的 FXS、FXST 及 FXSW 指令。可計算出指派之值。

10.4.28 在同步動作中決定路徑切線

功能

可在同步動作中讀取的系統變數$AC_TANEB（單節結尾切線角度 **T**angent**AN**gle at**E**nd of**B**lock）會計算目前單節結束的路徑切線與已程式設計的後續單節起點之路徑切線間的角度。

參數

切線角度的輸出值永遠是介於 0.0 到 180.0 度間的正值。如果在主要執行程式中沒有後續單節，將輸出-180.0 度角。

無法為系統所產生的單節（中間單節）讀取系統變數$AC_TANEB。可使用系統變數 $AC_BLOCKTYPE 判斷該單節是否為程式設計單節（主單節）。

範例

```
ID=2 EVERY $AC_BLOCKTYPE==0 DO $SR1 = $AC_TANEB
```

10.4.29 決定目前手動倍率

功能

目前手動倍率

（NC 元件）可使用系統變數進行讀取及寫入：

$AA_OVR 軸向手動倍率

$AC_OVR 路徑手動倍率

於同步動作中。

PLC 定義的手動倍率讓同步動作可讀入至系統變數中：

$AA_PLC_OVR 軸向手動倍率

$AC_PLC_OVR 路徑手動倍率

得出的手動倍率

讓同步動作可讀入至系統變數中：

$AA_TOTAL_OVR 軸向手動倍率

$AC_TOTAL_OVR 路徑手動倍率

得出的手動倍率可計算為：

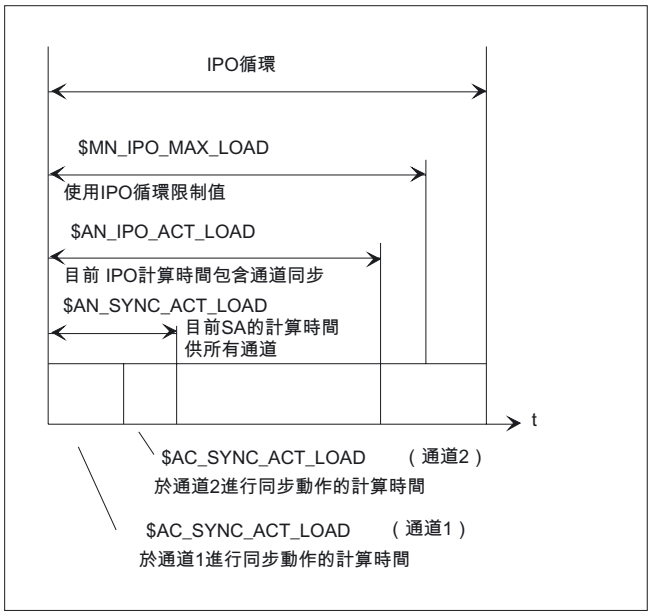
$AA_OVR * $AA_PLC_OVR 或

$AC_OVR * $AC_PLC_OVR

10.4.30 同步動作之時間使用評估

功能

在插補循環中，同步動作必須為由 NC 轉譯及計算出的動作。下列系統變數為同步動作提供目前該同步動作所有之插補循環的分時共享資訊，以及位置控制器的計算時間。



意義

只有在機械參數$MN_IPO_MAX_LOAD 大於 0 時變數的值才有效。否則，SINUMERIK powerline 及 solution line 中之變數會指定淨計算時間，此期間內 HMI 造成的中斷並不會被納入考量。淨計算時間得自：

- 同步動作時間，
- 位置控制時間及
- 無 HMI 中斷的剩餘 IPO 計算時間

系統變數永遠包含**先前** IPO 循環的值。

$AN_IPO_ACT_LOAD	目前 IPO 計算時間（包含所有通道的同步動作）
$AN_IPO_MAX_LOAD	最長 IPO 計算時間（包含所有通道的同步動作）
$AN_IPO_MIN_LOAD	最短 IPO 計算時間（包含所有通道的同步動作）
$AN_IPO_LOAD_PERCENT	目前 IPO 計算時間佔 IPO 循環的百分比（%）
$AN_SYNC_ACT_LOAD	所有通道同步動作的目前計算時間
$AN_SYNC_MAX_LOAD	所有通道同步動作的最長計算時間
$AN_SYNC_TO_IPO	同步動作擁有的完整 IPO 電腦時間（所有通道）百分比

$AC_SYNC_ACT_LOAD	通道中同步動作的目前計算時間
$AC_SYNC_MAX_LOAD	通道中同步動作的最長計算時間
$AC_SYNC_AVERAGE_LOAD	通道中同步動作的平均計算時間
$AN_SERVO_ACT_LOAD	位置控制器的目前計算時間
$AN_SERVO_MAX_LOAD	位置控制的最長計算時間
$AN_SERVO_MIN_LOAD	位置控制的最短計算時間

過載通知之變數：

使用機械參數 $MN_IPO_MAX_LOAD 設定淨 IPO 計算時間（佔 IPO 循環之百分比），系統變數 $AN_IPO_LOAD_LIMIT 在此設為 TRUE。若目前負載降到臨界值之下，則將該變數再次設為 FALSE。若機械參數為 0，則整個診斷函數會停用。

評估$AN_IPO_LOAD_LIMIT 可讓使用者定出避免層級溢位的方式。

10.5 技術循環

功能

可藉由同步動作中的動作啟動程式。該程式僅能由可在同步動作中動作的函數組成。以此方式構成的程式稱為技術循環。

技術循環以副程式的形式儲存在控制系統中。

可在同一通道中平行處理數個技術循環或動作。

程式設計

程式設計技術循環適用以下規則：

- 以 M02/M17/M30/RET 來程式設計程式結尾。
- ICYCOF 中指定的所有動作，無須等待循環，即可在單一程式層級中處理。
- 每一個同步動作最多可連續呼叫 8 個技術循環。
- 也可在依單節有效（非模態）的同步動作中使用技術循環。
- 可程式設計 IF 檢查結構及 GOTO，GOTOF 與 GOTOB 等跳躍指令。
- 以下規則適用於含 DEF 及 DEFINE 指令之單節：
  - DEF 及 DEFINE 指令會讀取（忽略語法錯誤）技術循環。
  - 若句法錯誤或不完整則會出現警報訊息。
  - 該指令無需設定即可讀取而不發出其自設的警報訊息。
  - 在工件程式循環中，會完整考量該指令以及數值指派。

參數傳輸

可傳輸參數至技術循環。會同時考量以“傳值呼叫”形式參數傳輸的簡單資料類型，以及在呼叫技術循環時有效的預設設定。分別為：

- 當未程式設計傳輸參數時，已程式設計的預設值。
- 含初始值之標準參數。
- 會傳輸含有預設值之未初始化實際參數。

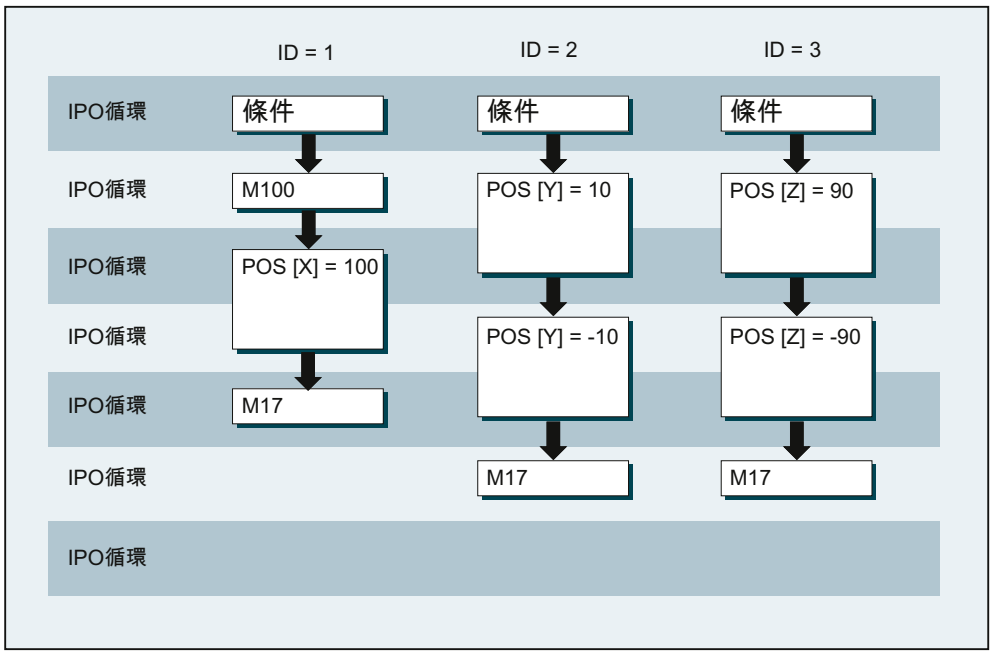
順序

技術循環會在滿足其條件後即刻啟動。技術循環中每一行程式皆在不同的 IPO 循環中處理。執行定位軸工作需要數個 IPO 循環。在同一循環中執行其他函數。在技術循環中依序執行單節。

若在同一插補循環中呼叫了相互排斥的動作，將啟用在同步動作中具較高 ID 編號的動作。

範例

範例 1：設定數位輸入開始軸程式



主程式：

程式碼	註解
`ID=1 EVERY $A_IN[1]==1 DO AXIS_X`	`;` 若輸入 1 為 1，啟動軸程式 AXI_X。
`ID=2 EVERY $A_IN[2]==1 DO AXIS_Y`	`;` 若輸入 2 為 1，啟動軸程式 AXI_Y。
`ID=3 EVERY $A_IN[3]==1 DO AXIS_Z`	`;` 若輸入 3 為 1，啟動軸程式 AXI_Z。
`M30`	

軸程式 AXIS_X：

程式碼

```
M100
POS[X]=100 FA[X]=300
M17
```

軸程式 AXIS_Y：

程式碼

```
POS[Y]=10 FA[Y]=200
POS[Y]=-10
M17
```

軸程式 AXIS_Z：

程式碼
```
POS[Z]=90 FA[Z]=250
POS[Z]=-90
M17
```

範例 2：技術循環中不同的程式順序

程式碼
```
PROC CYCLE
N10 DEF REAL VALUE=12.3
N15 DEFINE ABC AS G01
```
不需設立變數及 / 或巨集，讀取兩單節。

程式碼
```
PROC CYCLE
N10 DEF REAL
N15 DEFINE ABC G01
```
因未寫出正確句法，兩單節皆產生 NC 警報。

程式碼
```
PROC CYCLE
N10 DEF AXIS AXIS1=XX2
```
如 XX2 軸未知，則輸出警報 12080。否則讀取該單節，不發警報且不設立變數。

程式碼
```
PROC CYCLE
N10 DEF AXIS AXIS1
N15 G01 X100 F1000
N20 DEF REAL VALUE1
```
單節 N20 會永遠產生警報 14500，因為在第 1 程式指令後不允許定義操作。

10.5.1 環境變數（$P_TECCYCLE）

功能

$P_TECCYCLE 變數可用來將程式區分為同步動作程式及前置處理程式。之後便可處理單節或程式順序，使其正確寫好（依據句法），或將其當成工件程式循環處理。

轉譯環境變數

$P_TECCYCLE 系統變數可在技術循環中，控制程式區段的環境專屬轉譯：

```
IF $P_TECCYCLE==TRUE
...                          ; 同步動作中技術循環的程式順序
ELSE
...                          ; 工件程式循環的程式順序
ENDIF
```

說明

含錯誤或未經授權程式句法的單節，以及未知的數值指派都會使工件程式循環發出警報訊息。

範例

在技術循環中查詢$P_TECCYCLE 的程式順序：

程式碼	註解
PROC CYCLE	
N10 DEF REAL VALUE1	; 於技術循環中讀取。
N15 G01 X100 F1000	
N20 IF $P_TECCYCLE==TRUE	
...	; “技術循環的程式順序（不含變數值 1）
N30 ELSE	
...	; 工件程式循環的程式順序（變數值 1 可用）
N40 ENDIF	

10.5.2 傳值呼叫參數

功能

可使用傳值呼叫參數來定義技術循環。INT，REAL，CHAR，STRING，AXIS 及 BOOL 等簡單資料類型可當成參數使用。

說明

傳輸至傳值呼叫的形式參數不可為陣列。

實際參數亦可包含預設參數（請參閱"預設參數初始化 (頁 554)"）。

句法

```
ID=1 WHEN $AA_IW[X]>50 DO TEC（IVAL，RVAL，，SVAL，AVAL）
```

為未初始化實際參數傳輸預設值：

```
ID=1 WHEN $AA_IW[X]>50 DO TEC（IVAL，RVAL，，SYG_SS[0]，AVAL）
```

10.5.3 預設參數初始化

功能

在 PROC 指令中，也能以初始值提供預設參數。

句法

在技術循環中指派預設參數：

```
PROC TEC （INT IVAL=1，REAL RVAL=1.0，CHAR CVAL='A'，STRING[10]
SVAL="ABC"，AXIS AVAL=X，BOOL BVAL=TRUE）
```

若目前參數由預設參數組成，則會由 PROC 指令傳輸其初始值。此於工件程式及同步動作皆適用。

範例

程式碼	註解
`TEC （IVAL，RVAL，SVAL，AVAL）`	`;` CVAL 及 BVAL 適用初始值

10.5.4 技術循環之控制處理（ICYCOF、ICYCON）

功能

使用 ICYCOF 及 ICYCON 語法指令可控制技術循環的時間處理。

技術循環中所有單節會在使用 ICYCOF 的單一插補循環中處理。所有需要數個循環的動作會與 ICYCOF 平行處理。

應用

使用 ICYCON 時，指令軸移動可能會導致技術循環處理延遲。若不希望發生此種情況，可在單一插補循環中使用 ICYCOF 處理所有動作，而不須等待時間。

句法

下列說明適用於技術循環的循環處理：

`ICYCON`ICYCON 之後每個技術循環單節皆在不同的 IPO 循環中處理。

`ICYCOF`ICYCOF 之後所有技術循環單節皆在單一插補循環中處理。

說明

ICYCON 及 ICYCOF 這兩個語法指令僅在程式層級中才有效。這兩個指令在工件程式中沒有回應，因此容易被忽略。

ICYCOF 處理模式範例

程式碼	註解
IPO 循環	; PROC TECHNOCYC
1.	; $R1=1
2.25	; POS[X]=100
26.	; ICYCOF
26.	; $R1=2
26.	; $R2=$R1+1
26.	; POS[X]=110
26.	; $R1=3
26.	; RET

10.5.5 串聯技術循環

功能

最多可同時處理 8 個技術循環。因此，在單一同步動作中可程式設計數個技術循環。

句法

```
ID=1 WHEN $AA_IW[X]>50 DO TEC1（$R1） TEC2 TEC3（X）
```

執行順序

技術循環會依據前述程式設計，由左而右依序（以串聯方式）處理。若在 ICYCON 模式中處理循環，將延遲所有後續處理動作。警報會取消所有後續動作。

10.5.6 非模態同步動作中的技術循環

功能

在非模態同步動作中也可使用技術循環。
若技術循環的處理時間較相關單節的處理時間長，則會在單節變更時取消技術循環。

說明

技術循環不會阻止單節變更。

10.5.7 檢查結構（IF）

功能

可在同步動作中使用 IF 檢查結構，為技術循環建立處理順序的分支。

句法

```
IF <條件>
$R1=1
[ELSE] optional
$R1=0
ENDIF
```

10.5.8 跳躍指令（GOTO、GOTOF、GOTOB）

功能

可在技術循環中使用跳躍指令（GOTO、GOTOF、GOTOB）。為避免觸發警報，副程式中必須存在被指定的標籤。

說明

標籤及單節編號僅能為常數。

句法

無條件跳躍

GOTO 標籤，單節編號
GOTOF 標籤，單節編號
GOTOB 標籤，單節編號

跳躍指令及跳躍目的地

GOTO	跳躍先向前，再向後
GOTOF	向前跳躍
GOTOB	向後跳躍
標籤:	跳躍標記
單節編號	本單節的跳躍目的地
N100	單節編號為子單節
:100	單節編號為主單節

10.5.9 鎖定、解除鎖定、重置（LOCK、UNLOCK、RESET）

功能

可使用其他的模態同步動作來鎖定、解除鎖定或重置技術循環的順序。

句法

```
LOCK（<n1>，<n2>，...）
UNLOCK（<n1>，<n2>，...）
RESET（<n1>，<n2>，...）
```

意義

LOCK	鎖定同步動作的指令 中斷啟用中動作。
UNLOCK	解除鎖定同步動作的指令
RESET	重置技術循環的指令
<n1>，<n2>，...	待鎖定、解除鎖定或重置之同步動作或技術循環的辨識碼。

連鎖同步動作

可由 PLC 將 ID 編號**<n> = 1 ... 64** 的模態同步動作連鎖。將不再評估相關條件，且相關函數的執行將鎖在 NCK 裡。

在 PLC 介面中可使用單一訊號任意鎖定所有同步動作。

說明

已程式設計的同步動作將啟用為標準狀態，並能受到保護，不被機台資料設定覆寫或鎖定。終端使用者應無法修改機台製造商所定義的同步動作。

範例

範例 1：鎖定同步動作（LOCK）

程式碼

```
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130
...
N200 ID=2 WHENEVER $A_IN[2]==1 DO LOCK（1）
```

範例 2：解除鎖定同步動作（UNLOCK）

程式碼

```
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130
...
N200 ID=2 WHENEVER $A_IN[2]==1 DO LOCK（1）
...
N250 ID=3 WHENEVER $A_IN[3]==1 DO UNLOCK（1）
```

範例 3：中斷技術循環（RESET）

程式碼

```
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130
...
N200 ID=2 WHENEVER $A_IN[2]==1 DO RESET（1）
```

10.6 刪除同步動作（CANCEL）

功能

CANCEL 指令可用於取消（刪除）模態或來自工件程式的靜態同步動作。

若在由其所啟動之定位軸移動仍在啟用中時取消同步動作，則該定位軸移動會被中斷。若不希望發生此種情況，可在 CANCEL 指令之前以刪除軸剩餘距離的方式來停止軸動作。

句法

CANCEL（<n1>，<n2>，...）

意義

CANCEL:	刪除程式設計同步動作的指令
<n1>、<n2>、等:	待刪除之同步動作的辨識碼 **注意：** 若未指定識別編號，則會刪除**所有**的模態/靜態同步動作。

範例

範例 1：取消同步動作

程式碼	註解
N100 ID=2 WHENEVER $A_IN[1]==1 DO M130	
...	
N200 CANCEL（2）	; 刪除模態同步動作編號 2。

範例 2：在取消同步動作之前刪除剩餘距離

程式碼	註解
N100 ID=17 EVERY $A_IN[3]==1 DO POS[X]=15 FA[X]=1500	; 啟動定位軸動作。
...	
N190 WHEN ... DO DELDTG（X）	; 結束定位軸動作。
N200 CANCEL（17）	; 刪除模態同步動作編號 17。

10.7 以特定操作狀態控制行為

POWER ON

在 POWER ON 的過程中不會啟用任何同步動作。可由 PLC 啟動的非同步子程序（ASUB）啟動靜態同步動作。

模式變更

以關鍵字 IDS 啟動的同步動作在操作模式變更後仍有效。其他所有同步動作在操作模式變更後（如軸定位）會失效，並在重新定位及回到自動模式後再次啟用。

RESET

NC 重置會終止停止所有非模態及模態同步動作。靜態同步動作仍有效。可以開始新動作。若在 RESET 過程中有指令軸移動啟用，則該移動會取消。先前執行的的 WHEN 類型同步動作在 RESET 後不會再處理一次。

RESET 之後的回應		
同步動作 技術循環	**模態 / 非模態**	**靜態（IDS）**
	取消啟用中動作，刪除同步動作。	取消啟用中動作，重置技術循環。
軸 / 定位主軸	已取消動作。	已取消動作。
速度控制主軸	$MA_SPIND_ACTIVE_AFTER_RESET==1： 主軸仍有效 $MA_SPIND_ACTIVE_AFTER_RESET==0： 主軸停止。	
主動值耦合	$MC_RESET_MODE_MASK，bit13 == 1： 主動值耦合保持啟用 $MC_RESET_MODE_MASK，bit13 == 0： 主動值耦合分離	
量測操作	取消由同步動作啟動的量測程序。	取消由靜態同步動作啟動的量測程序。

NC 停止

靜態同步動作於 NC 停止時仍有效。由靜態同步動作啟動的移動不會取消。位於該程式且屬於有效單節的同步動作仍有效，由其所啟動的移動則停止。

程式結尾

程式結尾，與同步動作不會相互影響。即使程式結尾後，也會完成目前的同步動作。M30 單節中的同步動作仍有效。若不想希望發生此情況，必須在程式結尾前使用 CANCEL 取消同步動作。

程式結尾後之回應		
同步動作 技術循環	**模態/非模態動作 →已取消**	**靜態動作（IDS） 仍有效**
軸 / 定位主軸	延遲 M30 直到軸 / 主軸固定。	動作繼續。
速度控制主軸	程式結尾： $MA_SPIND_ACTIVE_AFTER_RESET==1： 主軸仍有效。 $MA_SPIND_ACTIVE_AFTER_RESET==0： 主軸停止。 在變更操作模式時主軸保持有效。	主軸仍有效。
主動值耦合	$MC_RESET_MODE_MASK，bit13 == 1： 主動值耦合保持啟用。 $MC_RESET_MODE_MASK，bit13 == 0： 主動值耦合分離。	由靜態同步動作啟動之耦合仍有效。
量測程序	取消由同步動作啟動的量測程序。	由靜態同步動作發動的量測程序仍有效。

單節搜尋

NC 啟動時，會在單節搜尋時收集同步動作並進行評估；如有必要會啟動相關動作。靜態同步動作在進行單節搜尋時可用。若在進行單節搜尋時發現使用 FCTDEF 程式設計的多項式係數，該係數將立即啟用。

使用非同步副程序 ASUB 中斷程式

ASUB（非同步副程式）啟動：

模態及靜態動作同步動作仍有效，且亦可在非同步副程式中操作。

ASUB（非同步副程式）結束：

若未以 REPOS 繼續執行非同步副程式，在非同步副程式中修改過的模態及靜態動作同步動作在主程式中將仍有效。

重新定位（REPOS）

重新定位（REPOS）之後，將再次啟用在被中斷單節內有效的同步動作。執行 REPOS 後處理剩餘單節時，被非同步副程式變更過的模態同步動作將不再有效。

非同步副程式及 REPOS 不會影響以 FCTDEF 程式設計的多項式係數。執行 REPOS 後，不論於何處程式設計，非同步副程式及主程式皆可在任何時刻使用該係數。

回應警報

以同步動作方式所啟動的軸與主軸動作會在回應具有動作停止的警報時停止。其他的所有動作（例如設定輸出）仍繼續執行。

若同步動作自行觸發了警報，則處理程序會中斷而此同步動作中後續的動作都將不會執行。若該同步動作為模態，則其在下一個插補循環中將不會進一步處理；亦即該警報僅會輸出一次。其他同步動作的處理則繼續照常進行。

產生解譯器停止的警報僅在處理過預先編碼的單節後才會生效。

若技術循環產生具有動作停止的警報，則相關循環的處理均會停止。

11 振盪

11.1 非同步震盪（OS、OSP1、OSP2、OST1、OST2、OSCTRL、OSNSC、OSE、OSB）

功能

振盪軸以所定義的進給率在反轉點 1 及 2 之間前後移動，直至振盪運動停用為止。

振盪運動期間，可依所需對其他軸進行插補。經由路徑移動或定位軸，可進行連續性進給，不過，**無特定關係**存在於振盪運動與進給運動之間。

非同步振盪的屬性

- 非同步振盪不受制於區塊限度，以軸為基礎而進行有效運作。
- 工件程式確保振盪移動之單節導向啟用。
- 不可進行數個軸的聯合插補，也不可將振盪軸重疊運作。

程式設計

下列指令可用於啟用及控制工件程式的非同步震盪。

程式化的數值輸入至含主執行時間中區塊同步化的相關設定資料內，此數值持續有效，直至再次變更為止。

句法

```
OSP1[<軸>]=<值> OSP2[<軸>]=<值>
OST1[<軸>]=<值> OST2[<軸>]=<值>
FA[<軸>]=<值>
OSCTRL[<軸>]=（<設定選項>，<重置選項>）
OSNSC[<軸>]=<值>
OSE[<軸>]=<值>
OSB[<軸>]=<值>
OS[<軸>] = 1
OS[<軸>] = 0
```

意義

<軸>	震盪軸的名稱
OS	啟動/停用震盪 值： 1 切換震盪為**開** 0 切換震盪為**關**
OSP1	定義反轉點 1 的位置
OSP2	定義反轉點 2 的位置 **注意：** 若啟用增量移動，則該位置會增量計算至程式設計在 NC 程式中的最後一個對應反轉位置。

OST1　以[秒]為單位定義反轉點 1 的停止時間

OST2　以[秒]為單位定義反轉點 2 的停止時間

<值>：
- -2　不等待精確停止而持續進行插補運作
- -1　等待精確停止粗調
- 0　等待精確停止微調
- >0　等待精確停止微調，然後等待特定停止時間

注意：
停止時間的單位與使用 G4 程式設計的停止時間相同。

FA　定義進給率

進給率為所定義的定位軸進給率。若未定義進給率，則適用儲存於機台資料中的數值。

OSCTRL　指定設定與重置選項

選項值 0 至 3 將停用時反轉點上的行為轉為密碼。可選擇 0 至 3 其中一個變數。剩餘的設定可自由與選擇的變數相結合。多重選項會附加加號（+）字元。

<值>：
- 0　在停用震盪時停在下一個反轉點上（預設）

  注意：
  只有在重置值 1 與值 2 時才可使用。
- 1　在振盪運作停用時，在反轉點 1 停止
- 2　在振盪運作停用時，在反轉點 2 停止
- 3　在振盪運作停用時，若未程式化滑磨衝程，不可逼近反轉點
- 4　在滑磨後逼近結束位置
- 8　若透過刪除剩餘距離的方式將震盪取消，則接下來需執行滑磨衝程並在必要時逼近結束位置。
- 16　若以刪除剩餘距離的方式取消震盪，則需與關機時一樣逼近對應的反轉點。
- 32　新進給只在下個反轉點後有效
- 64　FA 等於 0，FA = 0：路徑覆蓋啟動中

  FA 不等於 0，FA <> 0：速度覆蓋啟動中
- 128　用於旋轉軸 DC（最短路徑）
- 256　滑磨衝程是雙衝程（預設）。1=單衝程。
- 512　首先逼近起始位置

OSNSC　定義滑磨衝程次數

OSE　定義在停用震盪後要逼近的結束位置（使用工件座標系統）。

注意：
當程式設計 OSE 時，選項 4 在 OSCTRL 會不明確的生效。

OSB　定義在啟用震盪前要逼近的開始位置（使用工件座標系統）。

在反轉點 1 前先逼近開始位置。若開始位置與反轉位置 1 重疊，則接下來逼近反轉位置 2。當到達開始位置後，不會套用停止時間，即便此位置與反轉位置 1 重疊亦然；取而代之的是，該軸會等候精確停止微調訊號。已滿足任何已設定的精確停止條件。

注意：
設定資料 SD43770 $SA_OSCILL_CTRL_MASK 中的位元 9 需設定以便初始化至開始位置的逼近。

範例

範例 1：將震盪軸在兩個反轉點間震盪

振盪軸 Z 會在位置 10 與 100 間振盪。反轉點 1 會以精確停止微調逼近，反轉點 2 會以精確停止粗調逼近。震盪軸的進給率需為 250。在加工操作結尾時需執行 3 次滑磨衝程且震盪需逼近結束位置 200。進給軸的進給率需為 1 且以 X 方向進行的進給需在位置 15 結束。

程式碼	註解
WAITP（X，Y，Z）	; 起點。
G0 X100 Y100 Z100	; 變更為定位軸操作
WAITP（X，Z）	
OSP1[Z]=10 OSP2[Z]=100	; 反轉點 1、反轉點 2。
OSE[Z]=200	; 結束位置。
OST1[Z]=0 OST2[Z]=–1	; 在 U1 的停止時間：精確停止微調
	; 在 U2 的停止時間：精確停止粗調
FA[Z]=250 FA[X]=1	; 用於震盪軸的進給率，用於進給軸的進給率。
OSCTRL[Z]=（4，0）	; 設定選項。
OSNSC[Z]=3	; 3 次滑磨衝程。
OS[Z]=1	; 開始震盪
WHEN $A_IN[3]==TRUE DO DELDTG（X）	; 剩餘距離的刪除。
POS[X]=15	; 起始位置 X 軸
POS[X]=50	結束位置 X 軸
OS[Z]=0	; 停止震盪。
M30	

說明

OSP1[Z]=...至 OSNCS[Z]=...指令串亦可程式設計在一個單節中。

範例 2：使用反轉位置的線上修改進行震盪

可於工件程式中設定非同步震盪之必要設定資料。

若直接於程式中說明設定資料，則變更會於前置處理期間生效。同步回應可透過 STOPRE 前置處理停止達成。

程式碼	註解
$SA_OSCILL_REVERSE_POS1[Z]=-10	
$SA_OSCILL_REVERSE_POS2[Z]=10	
G0 X0 Z0	
WAITP（Z）	
ID=1 WHENEVER $AA_IM[Z] < $$AA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[X]=0	; 若震盪軸之實際值超出反轉點，則進給軸將停止。
ID=2 WHENEVER $AA_IM[Z] < $$AA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[X]=0	
OS[Z]=1 FA[X]=1000 POS[X]=40	; 啟動震盪。
OS[Z]=0	; 停用震盪。
M30	

其他資訊

振盪軸

振盪軸有以下特點：

- 所有的軸都可做為振盪軸之用。
- 數個振盪軸可同時有效（至多：定位軸的數量）。
- 線性插補 G1 對振盪軸總是有效 – 無論 G 指令目前於程式中是否有效。

振盪軸可以：

- 做為動態轉換的輸入軸而運作
- 做為門式（gantry）及聯合運動軸的導軸而運作
- 移動：
  - 無震動臨界值（BRISK）
    或
  - 有震動臨界值（SOFT）
    或
  - 含有彎處的加速度曲線（做為定位軸）。

振盪反轉點

在振盪位置如下定義時，必須將目前的偏移量列入考量：

- 絕對規格
  OSP1[Z]=<value>
  反轉點位置 = 偏移總量 + 程式化數值
- 相對規格
  OSP1[Z]=IC（<value>）
  反轉點位置 = 反轉點 1 + 程式化數值

範例：

程式碼

```
N10 OSP1[Z]=100 OSP2[Z]=110
...
...
N40 OSP1[Z]=IC(3)
```

WAITP

如果是以幾何軸來執行震盪，則此軸必須以 WAITP 啟動，來執行震盪。
當震盪結束時，WAITP 會再次被用來輸入震盪軸作為定位軸，因此可以繼續正常使用。

同步動作之震盪與停止次數

一旦設定停止時間期滿，內部單節更改會於震盪期間執行（由與軸的新剩餘距離顯示）。單節更改時會檢查關閉功能。關閉功能係依照動作順序的控制設定所定義（OSCTRL）。*此動態回應可由受進給手動超調影響。*

然後，可能會在火花消除衝程開始或結束位置逼近前執行一震盪衝程。*儘管看起來關閉回應已更改，然而實際上並非如此。*

11.2 由同步動作控制的震盪（OSCILL）

功能

此震盪模式下，進給動作可能只可於反轉點或於已定義之反轉區域中執行。
視需求而定，震盪移動可為

- 連續的，或
- 在進給執行完成前保持停止狀態。

句法

1. **定義震盪參數**
2. **定義動作－同步動作**
3. **指派軸，定義進給**

含義

OSP1[<震盪軸>]=	反轉點 1 位置
OSP2[<震盪軸>]=	反轉點 2 位置
OST1[<震盪軸>]=	於反轉點 1 的停止時間，單位為秒
OST2[<震盪軸>]=	於反轉點 2 的停止時間，單位為秒
FA[<震盪軸>]=	振盪軸的進給
OSCTRL[<震盪軸>]=	設定或重置選項
OSNSC[<震盪軸>]=	滑磨衝程次數
OSE[<震盪軸>]=	結束位置
WAITP（<震盪軸>）	啟用震盪之軸

軸指派，進給

OSCILL[<震盪軸>]=（<進給軸 1>，<進給軸 2>，<進給軸 3>）
POSP[<進給軸>]=(<結束位置>，<部分長度>，<模式>）

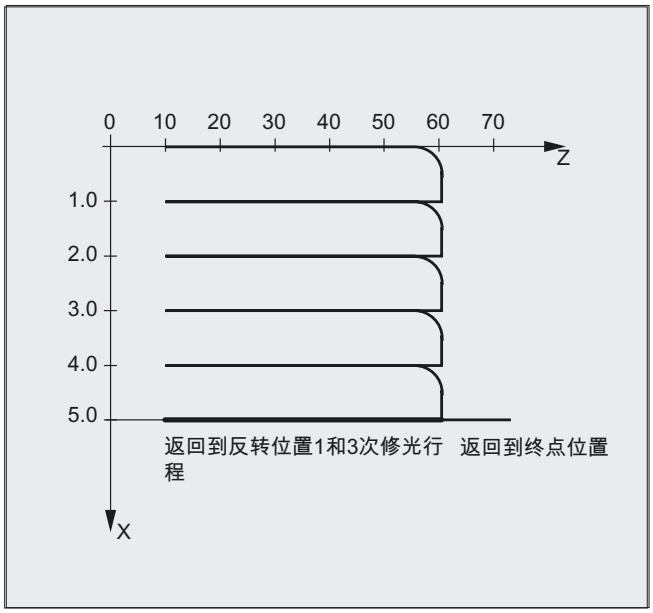
OSCILL:	為震盪軸指派進給軸或軸
POSP:	定義完整與部分的進給（請參閱“檔案與程式管理”一章）
結束位置:	當所有部分進給均移動後的進給軸結束位置。
部分長度:	於反轉點 / 反轉區域的部分進給長度
模式:	將完整進給分為部分進給的分割 0 = 同等尺寸的兩個剩餘步進（預設）； 1 = 同等尺寸的所有部分進給

動作－同步動作

WHEN… … DO	當...，進行...
WHENEVER … DO	每當...，進行...

範例

於反轉點 1 無須發生任何進給。於反轉點 2，進給會在反轉點 2 之前、距離 ii2 處進行，且震盪軸不需於反轉點等待部分進給結束。 軸 Z 為震盪軸而軸 X 為進給軸。



1. 震盪參數

程式碼	註解	
DEF INT ii2	;	為反轉區域 2 定義變數
OSP1[Z]=10 OSP2[Z]=60	;	定義反轉點 1 和 2
OST1[Z]=0 OST2[Z]=0	;	反轉點 1： 精確停止微調 反轉點 2： 精確停止微調
FA[Z]=150 FA[X]=0.5	;	震盪軸 Z 進給率，進給軸 X 進給率
OSCTRL[Z]=（2+8+16，1）	;	於反轉點 2 停用震盪動作；於刪除 DTG 火花消除及逼近結束位置後；於刪除 DTG 逼近反轉位置後
OSNC[Z]=3	;	滑磨衝程
OSE[Z]=70	;	結束位置 = 70
ii2=2	;	設定反轉點範圍
WAITP（Z）	;	為 Z 軸啟用震盪

2. 動作－同步動作

程式碼		註解
WHENEVER $AA_IM[Z]<$SA_OSCILL_REVERSE_POS2[Z]DO -> -> $AA_OVR[X]=0 $AC_MARKER[0]=0	;	於 MCS 中，若震盪軸 Z 的實際位置小於反轉範圍 2 的起始處，則將進給軸 X 的軸手動倍率設為 0%，並將索引 0 之位元記憶體的值設為 0。
WHENEVER $AA_IM[Z]>=$SA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[Z]=0	;	於 MCS 中，若震盪軸 Z 的實際位置大於反轉位置 2，則將震盪軸 Z 的軸手動倍率設為 0%。
WHENEVER $AA_DTEPW[X] == 0 DO $AC_MARKER[0]=1	;	若剩下的剩餘距離與部分進給相同，則將索引 0 之位元記憶體的值設為 1。
WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0 $AA_OVR[Z]=100	;	若索引 0 之位元記憶體不變，則將進給軸的軸手動倍率設為 100%－如此可以預防過早的進給發生（震盪軸 Z 仍未再度離開反轉範圍 2，不過進給軸 X 已就緒進行新進給），必須將震盪軸 Z 的手動倍率設為 100%（如此可取消第二個同步動作）。

-> 必須在單一單節中程式化。

3. 開始震盪

程式碼		註解
OSCILL[Z]=（X） POSP[X]=（5，1，1）	;	軸開始 已將震盪軸 Z 指派給 X 軸作為進給軸。 在結束位置 5 之前，軸 X 應以步進 1 移動。
M30	;	程式結尾

說明

1. **定義震盪參數**
   震盪的參數應在含有進給與震盪軸指派，及進給定義的移動單節之前定義（請參閱“非同步震盪”）。
2. **定義動作－同步動作**
   可定義下列同步條件：
   抑制進給直到震盪軸設置在反轉區域內（ii1、ii2）或位於反轉點（U1、U2）時為止。於反轉點進給時**停止震盪動作**。部分進給完成時**重新開始震盪移動**。定義**下個部分進給的開始**。
3. **指派震盪及進給軸**以及**部分與完整進給**。

定義震盪參數

指派震盪與進給軸：OSCILL

```
OSCILL[oscillating axis] = （infeed axis1，infeed axis2，infeed
axis3）
```

使用 OSCILL 指令定義軸指派與震盪移動的開始。

可為一震盪軸指派至多三個進給軸。

說明

震盪開始前，必須先為軸行為定義同步條件。

定義進給：POSP

```
POSP[infeed axis] = （End pos，partial length，mode）
```

透過 POSP 指令向控制系統做以下宣告：

- 完整進給（參考終點）
- 部分進給於反轉點或在反轉區域中的長度
- 達到終點時的部分進給回應（並參考模式）

Mode = 0	最後兩次部分進給距離目的地的剩餘距離，會被等分為兩個相同的步進（預設設定）。
Mode = 1	所有部分進給的尺寸相同會從完整的進給中計算出。

定義動作－同步動作

一般震盪會使用下列之同步－動作動作。

提供個別作業的解決方案，可作為建立使用者專屬之震盪移動的模組

說明

個別案例中，可以不同的方式程式設計同步條件。

關鍵字

WHEN … DO …	若…，則…
WHENEVER … DO	每當…，則…

功能

您可使用下列詳述之語言資源建置以下功能:

1. 於反轉點的進給
2. 於反轉區域的進給
3. 於兩反轉點進給
4. 於反轉點停止震盪移動。
5. 重新開始震盪移動。
6. 切勿過早開始部分進給。

為在此舉出之同步動作範例，提出下列假設：

- 反轉點 1 < 反轉點 2
- Z = 震盪軸
- X = 進給軸

說明

如需詳細資訊，請參閱“動作－同步動作”一節。

指派震盪及進給軸以及部分與完整進給。

於反轉點範圍內進給

進給動作必須在抵達反轉點之前，於反轉區域內開始。
這些同步動作在震盪軸到達反轉區域內之前將禁止進給移動。
下列指令受上述假設約束：

反轉點範圍 1：

```
WHENEVER
$AA_IM[Z]>$SA_OSCILL
 _RESERVE_POS1[Z]+ii1
DO $AA_OVR[X] = 0
```

於 MCS 中，每當震盪軸的實際位置大於反轉範圍 1 起始處，則將進給軸的軸手動倍率設為 0%。

反轉點範圍 2：

```
WHENEVER
$AA_IM[Z]<$SA_OSCILL
 _RESERVE_POS2[Z]+ii2
DO $AA_OVR[X] = 0
```

於 MCS 中，每當震盪軸的實際位置小於反轉範圍 2 起始處，則將進給軸的軸手動倍率設為 0%。

於反轉點的進給

只要震盪軸未到達反轉點，進給軸便不會移動。
在已知假設下取得下列指令（參考上文）：

反轉範圍 1：

```
WHENEVER
$AA_IM[Z]<>$SA_OSCIL
L_RESERVE_POS1[Z] DO
$AA_OVR[X] = 0 →
→ $AA_OVR[Z] = 100
```

於 MCS 中，每當震盪軸 Z 的實際位置大於或小於反轉點 1 的位置，則將進給軸 X 的軸手動倍率設為 0%，並將震盪軸 Z 的軸手動倍率設為 100%。

反轉範圍 2：

針對反轉點 2：

```
WHENEVER
$AA_IM[Z]<>$SA_OSCIL
L_RESERVE_POS2[Z] DO
$AA_OVR[X] = 0 →
→ $AA_OVR[Z] = 100
```

於 MCS 中，每當震盪軸 Z 的實際位置大於或小於反轉點 2 的位置，則將進給軸 X 的軸手動倍率設為 0%，並將震盪軸 Z 的軸手動倍率設為 100%。

於反轉點停止震盪移動。

震盪軸於反轉點處停止，而進給動作同時開始。進給移動完成時，震盪動作仍繼續。

同時，若同步動作因目前仍有效的上一個同步動作而中斷，則此同步動作可用來啟動進給移動。

在已知假設下取得下列指令（參考上文）：

反轉範圍 1：

```
WHENEVER
$SA_IM[Z]==$SA_OSCIL
L_RESERVE_POS1[Z] DO
$AA_OVR[X] = 0 →
→ $AA_OVR[Z] = 100
```

於 MCS 中，每當震盪軸 Z 的實際位置與反位置 1 相同，則將震盪軸的軸手動倍率設為 0%，並將進給軸的軸手動倍率設為 100%。

反轉範圍 2：

```
WHENEVER
$SA_IM[Z]==$SA_OSCIL
L_RESERVE_POS2[Z] DO
$AA_OVR[X] = 0 →
→ $AA_OVR[Z] = 100
```

於 MCS 中，每當震盪軸 Z 的實際位置與反位置 2 相同，則將震盪軸 X 的軸手動倍率設為 0%，並將進給軸的軸手動倍率設為 100%。

反轉點的線上評估

若主要執行變數之比較右側為$$，則兩變數會在 IPO 循環中持續相互評估與比較。

說明

請參考“動作同步動作”一節，取得更多資訊。

震盪移動重新開始

本同步動作的目的是在工件進給移動完成時，使進給軸仍能持續移動。

在已知假設下取得下列指令（參考上文）：

```
WHENEVER
$AA_DTEPW[X]==0 DO
$AA_OVR[Z]= 100
```

於 WCS 中，每當進給軸 X 之部分進給的剩餘距離等於零，則將震盪軸的軸手動倍率設為 100%。

下一個部分進給

進給完成時，必須禁止下個部分進給過早開始。

通道專屬標記（$AC_MARKER[Index]）可用來達成此目的。它於部分進給結束時（部分剩餘距離 ≡ 0）啟用，且於軸離開反轉區域時被刪除。同步動作會預防下個進給移動發生。

以這些已知假設為基礎，下列指令將適用於反轉點 1：

1. 設定標記：

```
WHENEVER
$AA_DTEPW[X] == 0 DO
$AC_MARKER[1]=1
```

於 WCS 中，每當進給軸 X 之部分進給的剩餘距離等於零，則將含索引 1 的位元記憶體設為 1。

2. 刪除標記

```
WHENEVER $AA_IM[Z]<>
$SA_OSCILL_RESERVE_P
OS1[Z] DO
$AC_MARKER[1] = 0
```

於 MCS 中，每當若震盪軸 Z 的實際位置大於或小於反轉點 1 的位置，則將位元記憶體 1 設為 0。

3. 禁止進給

```
WHENEVER
$AC_MARKER[1]==1 DO
$AA_OVR[X]=0
```

每當位元記憶體 1 相同，則將進給軸 X 的軸手動倍率設為 0%。

12 沖孔與切片

12.1 啟用，停用

12.1.1 沖孔與切片開啟或關閉（SPOF、SON、PON、SONS、PONS、PDELAYON、PDELAYOF、PUNCHACC）

功能

啟動/停用沖孔與切片

PON 與 SON 可用於啟動沖孔與切片函數。SPOF 結束所有沖孔及切片專屬函數。模態指令 PON 及 SON 相互排斥，例如 PON 停用 SON 也會停用。

以前導程式進行沖孔/切片

SONS 及 PONS 函數亦可啟動沖孔或切片功能。

與 SON/PON 相反（於補插層級進行衝擊控制），這些函數於伺服層級進行衝擊初始動作的訊號相關控制。這啟用了更高的衝程頻率，並因此增加了要達成的沖孔容量。

當訊號在前導程式中被評估，所有造成函數切片或沖孔軸改變位置的函數（例如，手輪移動，透過 PLC 更改至框架，量測函數）皆禁止使用。

含延遲的沖孔

PDELAYON 會延遲沖孔衝程的輸出。模態有效指令具有準備的功能，因此通常放在 PON 之前。執行 PDELAYOF 後會繼續進行正常沖孔。

說明

延遲時間係設定在設定資料 SD42400 $SC_PUNCH_DWELLTIME 中。

依賴移動的加速度

PUNCHACC 可用來指定加速度特定，依照孔洞間距來定義不同的加速度率。

第二沖孔介面

需要使用第二沖孔介面的加工（第二沖孔元件或可相比的媒介），可選擇切換至控制面版上的第二對快速數位輸入與輸出（I/O 組）。兩種介面均提供完整的沖孔 / 切片功能。SPIF1 與 SPIF2 指令可用來在第一和第二沖孔介面之間切換。

說明

條件：第二 I/O 組必須在機械參數中定義給沖孔功能（→請參考機台製造商規格'）。

句法

```
PON G... X... Y... Z...
SON G... X... Y... Z...
SONS G... X... Y... Z...
PONS G... X... Y... Z...
PDELAYON
PDELAYOF
PUNCHACC(<S 最小>,<A 最小>,<S 最大>,<A 最大>)
SPIF1/SPIF2
SPOF
```

意義

PON	啟動沖孔。
SON	啟動切片。
PONS	使用前導程式啟動沖孔。
SONS	使用前導程式啟動切片。
SPOF	停用沖孔/切片。
PDELAYON	使用延遲啟動沖孔。
PDELAYOF	以延遲來停用沖孔
PUNCHACC	啟動移動相依加速度。 參數： <S 最小> 最小孔洞間距 <A 最小> 初始加速度 <A 最小>可大於<A 最大>。 <S 最大> 最大孔洞間距 <A 最大> 最終加速度 <A 最大>可大於<A 最小>。
SPIF1	啟動**第一**沖孔介面。 衝程係使用第一組快速 I/O 來控制。
SPIF2	啟動**第二**沖孔介面。 衝程係使用第二組快速 I/O 來控制。

注意：
重置後或控制系統開機後，第一沖孔介面便永遠為啟用中。如果只使用一個沖孔介面，則不必將其程式設計。

範例

範例 1：啟用沖壓

程式碼	註解
...	
N70 X50 SPOF	; 無初始沖孔定位。
N80 X100 SON	; 啟用沖壓；在動作之前（X=50）以及在完成程式設計之移動時（X=100），啟動衝程。
...	

範例 2：含延遲的沖孔

程式碼	註解
...	
N170 PDELAYON X100 SPOF	; 不含沖孔啟動的位置，啟用延遲沖孔啟動。
N180 X800 PON	; 啟動沖孔。抵達結束位置時，沖孔衝程會延遲輸出。
N190 PDELAYOF X700	; 以延遲停用沖孔，一般沖孔初始對已程式設計移動的補償。
...	

範例 3：以兩個沖孔介面進行沖孔

程式碼	註解
...	
N170 SPIF1 X100 PON	; 單節結束時，在第一快速輸出上啟動衝程。在第一輸入上監控“衝程啟用”訊號。
N180 X800 SPIF2	; 在第二高速輸出上啟動第二衝程。在第二輸入上監控“衝程啟用”訊號。
N190 SPIF1 X700	; 使用第一介面控制其他所有衝程。
...	

其他資訊

以前導程式進行沖孔及切片（PONS/SONS）

不能同時在同一通道中以前導程式進行沖孔及切片。PONS 或 SONS 一次只能在一個通道中啟用。

移動相關加速度（PUNCHACC）

範例：

PUNCHACC（2，50，10，100）

孔洞間距離小於 2 毫米：

軸會以最高加速度的 50%的速率加速。

孔洞間距離介於 2 毫米至 10 毫米間：

加速度會增加到 100%，與間距成比例。

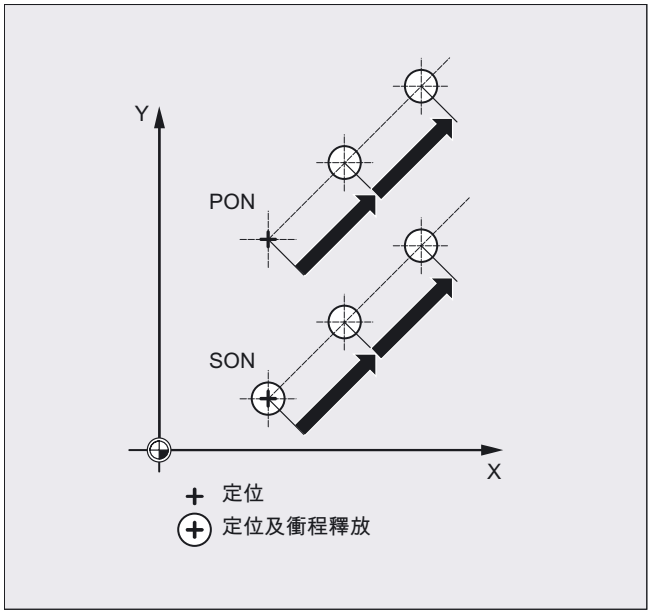
孔洞間距離大於 10 毫米：

以 100%之加速度移動。

初次衝程之初始化

啟用本函數後發生首次衝擊的時間點取決於所選的是切片或沖孔：

- PON/PONS：
  - 所有衝程－包括啟用後位於第一個單節的衝程－皆在單節結尾時執行。
- SON/SONS：
  - 在單節開始時執行的切片函數啟用後之首次衝程。
  - 後續各衝程皆在單節結尾時啟動。



現地沖孔及切片

衝程僅在若單節中包含沖孔或切片軸（位於有效平面的軸）的移動資訊時啟動。

然而，若要在同一位置啟動衝程，可程式設計沖孔 / 切片軸其中一個的移動路徑為 0。

以可旋轉工具加工

說明

若您想將可旋轉工具定位在已程式設計的路徑之切線上，請使用切線控制函數。

M 指令的使用

如同較早的版本，巨集技術允許特殊 M 系列功能的使用，不必採用語言指令（一致性）。M 系列功能以及對等的語言指令，亦用於較早的系統如下：

```
M20，M23        SPOF
M22             SON
M25             PON
M26             PDELAYON
```

巨集檔案範例：

程式碼	註解
DEFINE M25 AS PON	; 沖孔 ON（開啟）
DEFINE M125 AS PONS	; 含前導的沖孔開啟
DEFINE M22 AS SON	; 切片功能開啟
DEFINE M122 AS SONS	; 含前導的切片開啟
DEFINE M26 AS PDELAYON	; 含延遲的沖孔開啟
DEFINE M20 AS SPOF	; 沖孔 / 切片功能關閉
DEFINE M23 AS SPOF	; 沖孔與切片關閉

程式設計範例：

程式碼	註解
...	
N100 X100 M20	; 無初始沖孔定位。
N110 X120 M22	; 啟動切片，在動作之前和之後的初始衝程，
N120 X150 Y150 M25	; 啟動沖孔，在動作結尾初始衝程。
...	

12.2 自動路徑分段

功能

分割為路徑區段

沖孔或切片為啟動時，SPP 及 SPN 皆會將為路徑軸程式設計的總移動區段，分割成數個等長的路徑區段（等距路徑區段）。每個路徑區段內部會與一單節對應。

衝程數量

沖孔時，會在第一個路徑區段的終點進行首次衝擊；而切片時，則會在第一個路徑區段的起點進行首次衝擊。因此，完整的移動區段會得到以下數量：

沖孔：衝程數量=路徑區段數量

切片：衝程數量=路徑區段數量+1

輔助函數

輔助函數在產生的第一個單節中執行。

句法

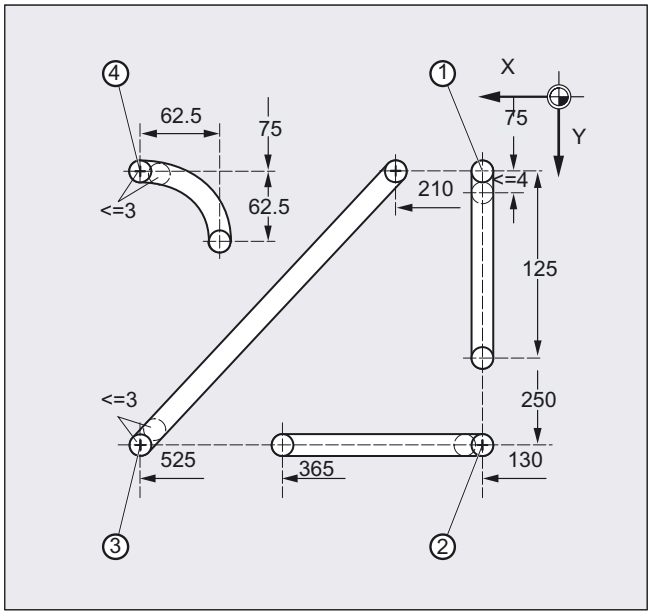
```
SPP=
SPN=
```

意義

SPP	路徑區段大小（衝程間的最大距離）；模態
SPN	每一單節之路徑區段數量；模態有效

範例 1

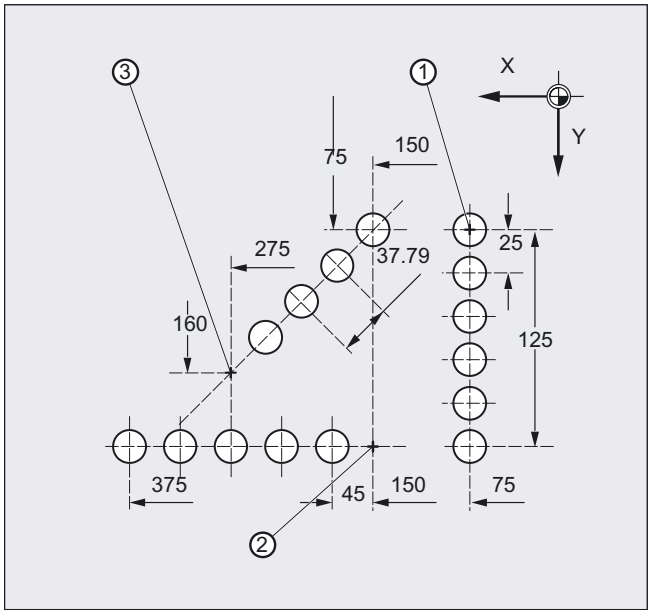
應自動將程式設計切片區段分割成路徑區段。



程式碼		註解
N100 G90 X130 Y75 F60 SPOF	;	定位於起點 1
N110 G91 Y125 SPP=4 SON	;	切片開啟；自動路徑分段之最長路徑區段長度：4 毫米（3.81 磅）
N120 G90 Y250 SPOF	;	切片關閉，定位至起點 2
N130 X365 SON	;	切片開啟；自動路徑分段之最長路徑區段長度：4 毫米（3.81 磅）
N140 X525 SPOF	;	切片關閉，定位至起點 3
N150 X210 Y75 SPP=3 SON	;	切片開啟；自動路徑分段之最長路徑區段長度：3 毫米（3.81 磅）
N140 X525 SPOF	;	切片關閉，定位至起點 4
N170 G02 X-62.5 Y62.5 I J62.5 SPP=3 SON	;	切片開啟；自動路徑分段之最長路徑區段長度：3 毫米（3.81 磅）
N180 G00 G90 Y300 SPOF	;	切片關閉

範例 2

應為各組孔洞進行自動路徑分段。並為分段指定最長路徑區段長度（SPP 值）。



程式碼		註解
N100 G90 X75 Y75 F60 PON	;	定位至起點 1； 沖出單一孔洞
N110 G91 Y125 SPP=25	;	自動路徑分段之最長路徑區段長度： 25 毫米（3.81 磅）
N120 G90 X150 SPOF	;	沖孔關閉，定位至 起點 2
N130 X375 SPP=45 PON	;	沖孔開啟；自動路徑分段之最長路徑區段長度： 45 毫米（3.81 磅）
N140 X275 Y160 SPOF	;	沖孔關閉，定位至 起點 3
N150 X150 Y75 SPP=40 PON	;	沖孔開啟，相對於使用已程式設計的路徑區段長度 40 毫米， 改用計算出的路徑區段長度 37.79 毫米。
N160 G00 Y300 SPOF	;	沖孔關閉；定位

12.2.1 路徑軸的路徑分割

SPP 路徑區段長度

SPP 用來指定衝程間最大間距，也就是其中跨移距離總長所分割的路徑區段之最長長度。以 SPOF 或 SPP=0 停用指令。

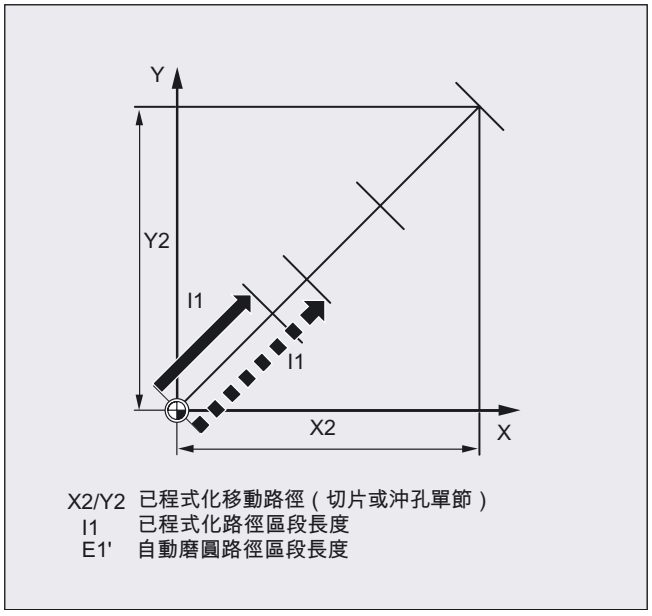
範例：

```
N10 SON X0 Y0
N20 SPP=2 X10
```

總長 10 毫米的移跨距離將分割成五個路徑區段，每段 2 毫米（SPP=2）。

說明

經由 SPP 所產生的路徑區段總是等距，即所有區段的長度都相同。也就是說，只有在總跨移距離的商數（quotient）及 SPP 值是整數時，所程式設計的路徑區段規格（SPP 設定）才有效如果不是如此，路徑區段規格便會為了產生整數商數，在內部減少。



X2/Y2 已程式化移動路徑（切片或沖孔單節）
I1 已程式化路徑區段長度
E1' 自動磨圓路徑區段長度

範例：

```
N10 G1 G91 SON X10 Y10
N20 SPP=3.5 X15 Y15
```

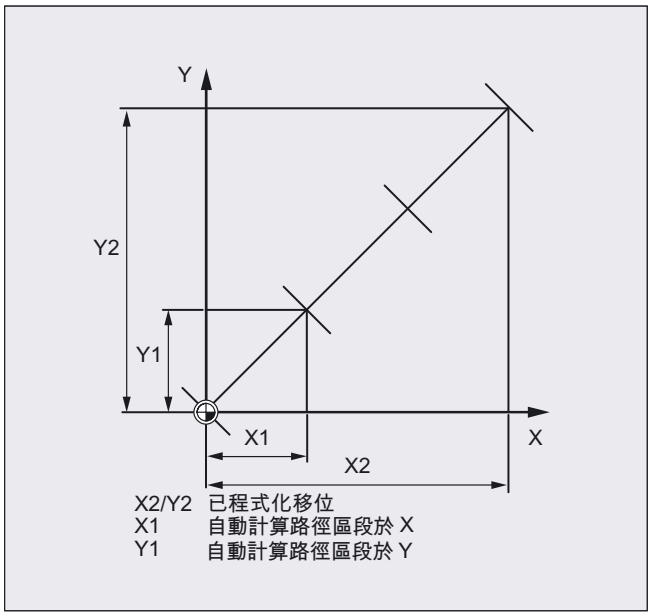
總跨移距離為 15 毫米，而路徑區段長度為 3.5 毫米時，其商數不是整數值（4.28）。在這樣的情況下，SPP 值被降至下個可能的整數商數。此範例的結果會是路徑區段長度為 3 毫米。

SPN 路徑區段數量

SPN 定義由跨移距離總長所產生的路徑區段數量。區段長度自動計算。因為 SPN 為非模態，所以沖孔或沖壓必須事先分別使用 PON 或 SON 啟動。

同單節中的 SPP 與 SPN

若在同一單節中將路徑區段長度（SPP）及路徑區段數量（SPN）程式設計，則 SPN 適用於此單節，而 SPP 適用於所有後續單節。如果 SPP 在 SPN 之前啟動，則 SPP 在含 SPN 的單節之後，才會再度有效。



說明

若在控制系統中可使用沖孔 / 切片函數，則可獨立於本技術使用 SPN 或 SPP 程式設計自動路徑分段函數。

12.2.2 單一軸的路徑分割

如果除路徑軸之外，單一軸也被定義為沖孔 / 切片軸，則可為單一軸啟動自動路徑分割函數。

單一軸對 SPP 的回應

已程式設計的路徑區段長度（SPP）基本上是指路徑軸。因此，在含有單一軸動作及 SPP 值的單節中，忽略 SPP 值，但不包含已程式設計的路徑軸。

若在單節中程式設計單一軸與路徑軸，則單一軸依照相關機台資料的設定而反應。

1. 標準設定

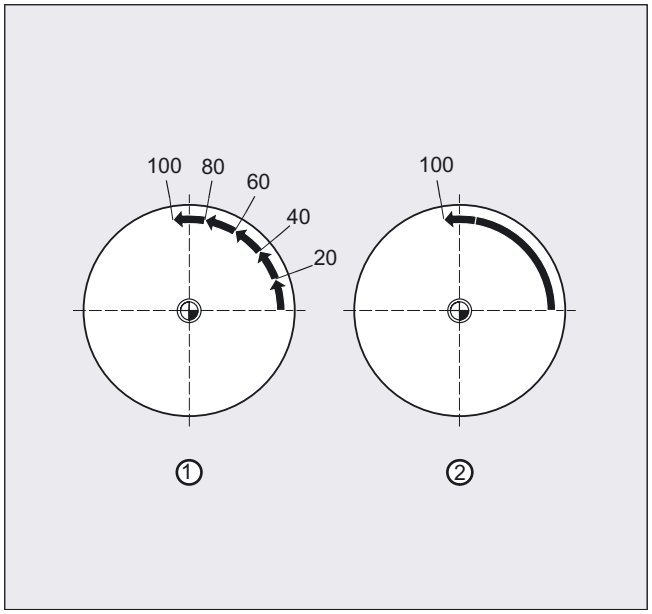
   單一軸所移跨的路徑，均勻分布在由 SPP 所產生的中間單節上。

範例：

```
N10 G1 SON X10 A0
N20 SPP=3 X25 A100
```

衝程間已程式設計的間距為 3 毫米，因此為 15 毫米的 X 軸（路徑軸）跨越距離總長，產生五個單節。

A 軸便在每個單節中，以 20°旋轉。



1. 無路徑分割的單一軸
   單一軸在產生的第一個單節之總長度上移跨。
2. 有 / 無路徑分割
   單一軸的回應，視路徑軸插補而定：

- 圓弧插補：路徑分割
- 線性插補：無路徑分段

對 SPN 的回應

即使在同一個單節中未程式設計路徑軸，仍適用已程式設計的路徑區段數量。
需求：單一軸定義為沖孔 / 切片軸。

13 研磨

13.1 工件程式中研磨專屬刀具監控（TMON、TMOF）

功能

使用 TMON 指令，可在 NC 工件程式中啟動研磨刀具（類型 400 - 499）的幾何及速度監控。直到在工件程式中用 TMOF 指令停用前，監控仍有效。

說明

請遵守機台製造商'之指示!

需求

需設定研磨專屬刀具參數$TC_TPG1 to $TC_TPG9。

句法

TMON（<T-編號>）
TMOF（<T-編號>）

說明

TMON	**開啟**研磨專屬刀具監控之指令
TMOF	**關閉**研磨專屬刀具監控之指令
<T-編號>	指定 T 編號 **注意:** 僅需在刀具未啟用該 T 編號時使用。
TMOF（0）	停用監控所有刀具

其它資訊

研磨專屬刀具參數

參數	意義	資料類型
$TC_TPG1	主軸編號	INT
$TC_TPG2	鏈結規則 讓研磨輪的左右側自動維持相同的參數。	INT
$TC_TPG3	最短輪半徑	REAL
$TC_TPG4	最短輪寬度	REAL
$TC_TPG5	目前輪寬度	REAL
$TC_TPG6	最快速度	REAL
$TC_TPG7	最快周邊速度	REAL
$TC_TPG8	斜式輪的角度	REAL
$TC_TPG9	半徑計算之參數編號	INT

參考資料：
功能手冊基本功能；刀具偏移（W1）

藉由選擇刀具來啟動刀具監控

根據機械參數設定，啟動刀具選擇時，可自動啟動研磨刀具（類型 400-499）之刀具監控。
在任何時刻都只能對各主軸啟用**一個**監控常式。

幾何監控

監控目前輪半徑及目前寬度。
依照含主軸手動倍率允差之速度限制，週期性地監控設定速度。
速度限制為，將最快速度與自輪週邊速度和目前輪半徑所計算出的速度做比較之後，所得出其中較小的值。

不含 T 或 D 編號之工作

可為每一機械參數設定一個標準 **T** 編號及標準 **D** 編號，
不需重新程式設計，在電源開啟 / 重置後即可生效。
範例：以同一個研磨輪執行所有加工。
可使用機械參數設定重置後仍有效的刀具（請參閱“任意D編號指派，刀刃編號 (頁 381)”）。

14 附加功能

14.1 軸函數（AXNAME、AX、SPI、AXTOSPI、ISAXIS、AXSTRING、MODAXVAL）

功能

例如，若在軸名稱不明時，可使用 AXNAME 產生一般有效的循環。

使用 AX 可間接程式設計幾何及同步軸。軸辨識碼存在 AXIS 類型的變數中，或由 AXNAME 或 SPI 等指令提供。

若為主軸（例如同步主軸）程式設計軸函數，可使用 SPI。

使用 AXTOSPI 可將軸辨識碼轉換成主軸索引（SPI 之反函數）。

使用 AXSTRING 可將軸辨識碼（資料類型 AXIS）轉換成字串（AXNAME 之反函數）。

在通用循環中使用 ISAXIS，可確保特定幾何軸存在，而不會發出錯誤訊息取消後續一切 $P_AXNX 呼叫。

使用 MODAXVAL 可判定模數旋轉軸之模數位置。

句法

```
AXNAME（"string"）
AX[AXNAME（"string"）]
SPI（n）
AXTOSPI（A）或 AXTOSPI（B）或 AXTOSPI（C）
AXSTRING（ SPI（n） ）
ISAXIS（<幾何軸編號>）
<模數位置>=MODAXVAL(<軸>,<軸位置>)
```

意義

AXNAME	將輸入字串轉換成軸辨識碼；輸入字串需含有效的軸名稱。
AX	變數軸識別碼
SPI	將主軸編號轉換成軸辨識碼；傳輸參數需含有效的主軸編號。
n	主軸編號
AXTOSPI	將軸辨識碼轉換成整數主軸索引。AXTOSPI 是 SPI 的反轉函數。
X，Y，Z	將 AXIS 類型的軸辨識碼當成變數或常數
AXSTRING	字串會隨相關主軸編號輸出。
ISAXIS	檢查指定的幾何軸是否存在。
MODAXVAL	判定模數旋轉軸之模數位置；其符合參考參數化模數範圍的模態休止點（在預設設定中，為 0 到 360 度；可利用 MD30340 MODULO_RANGE_START 及 MD30330 $MA_MODULO_RANGE 變更模數範圍的起點及大小。）

說明

SPI 副檔名

軸函數 SPI（n）亦可用來讀寫框架元件。這代表可使用例如$P_PFRAME[SPI（1），TR]=2.22 句法寫入框架。

可另外使用位址 AX[SPI（1）]=<軸位置>程式設計軸位置來移動軸。前提是主軸需在定位或軸模式之下。

範例

範例 1：AXNAME、AX、ISAXIS

程式碼	註解
OVRA[AXNAME（"Transverse axis"）]=10	; 手動倍率橫向軸
AX[AXNAME（"Transverse axis"）]=50.2	; 橫向軸結束位置
OVRA[SPI（1）]=70	; 手動倍率主軸 1
AX[SPI（1）]=180	; 主軸 1 結束位置
IF ISAXIS（1） == FALSE GOTOF CONTINUE	; 可使用橫座標?
AX[$P_AXN1]=100	; 移動橫座標
CONTINUE:	

範例 2：AXSTRING

使用 AXSTRING[SPI（n）]進行程式設計時，指派給主軸的軸索引值將不再以主軸編號形式輸出，而是輸出字串"Sn"。

程式碼	註解
AXSTRING[SPI（2）]	; 輸出字串 "S2"。

範例 3：MODAXVAL

模數旋轉軸 A 的模數位置待判定。

計算的起始值為軸位置 372.55。

參數化模數範圍為 0 到 360 度：

MD30340 MODULO_RANGE_START = 0

MD30330 $MA_MODULO_RANGE = 360

程式碼	註解
R10=MODAXVAL（A，372.55）	; 計算出模數位置 R10=12.55。

範例 4：MODAXVAL

若已程式設計的軸辨識碼並未參考模數旋轉軸，則會傳回原先欲轉換之值（<軸位置>）。

程式碼	註解
R11=MODAXVAL（X，372.55）	; X 為線性軸；R11 = 372.55。

14.2 可更換的幾何軸（GEOAX）

功能

“可更換幾何軸”函數可透過欲從工件程式修改的機械參數，來設定幾何軸群組。此處的任何幾何軸可以定義為同步特殊軸的通道軸替換。

句法

```
GEOAX(<n>,<通道軸>,<n>,<通道軸>,<n>,<通道軸>)
GEOAX（）
```

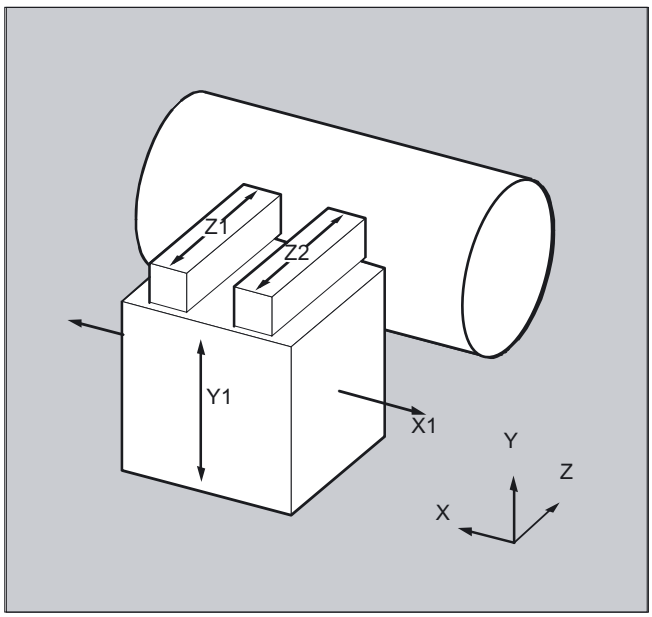
意義

GEOAX（...）	變更（更換）幾何軸之指令 **注意：** GEOAX（）為無任何參數呼叫幾何軸的基本設定。
<n>	可用參數指定應指派給後續指定通道軸的幾何軸編號。 值域：　1、2 或 3 **注意：** 使用**<n>=0** 可由幾何軸群組－無需更換－完全移除後續指定通道軸。
<通道軸>	可用參數指定應包含在幾何軸群組的通道軸名稱。

範例

範例 1：在兩軸間切換決定幾何軸

可使用通道軸 X1、Y1、Z1、Z2 來移動刀具滑座：



幾何軸已設定，故開啟電源後，首先 Z1 便為有效第三幾何軸，幾何軸名稱為“Z”，並與 X1 及 Y1 共同構成幾何軸群組。

現在在工件程式中會使用 Z1 及 Z2，並在兩者間切換作為幾何軸 Z。

程式碼	註解
...	
N100 GEOAX（3，Z2）	; 通道軸 Z2 扮演第三座標軸（Z）。
N110 G1 ...	
N120 GEOAX（3，Z1）	; 通道軸 Z1 扮演第三座標軸（Z）。
...	

範例 2：變更六通道軸之幾何軸。

一個機台有六個通道軸，名稱分別為 XX、YY、ZZ、U、V、W。

以機械參數進行座標軸設定的基本設定為：

通道軸 XX = 第一幾何軸（X 軸）

通道軸 YY = 第二幾何軸（Y 軸）

通道軸 ZZ = 第三幾何軸（Z 軸）

程式碼	註解
N10 GEOAX（）	; 幾何軸的基本設定生效。
N20 G0 X0 Y0 Z0 U0 V0 W0	; 所有軸快速移動至位置 0。
N30 GEOAX（1，U，2，V，3，W）	; 通道軸 U 成為第一幾何軸（X），V 成為第二幾何軸（Y），而 W 為第三幾何軸（Z）。
N40 GEOAX（1，XX，3，ZZ）	; 通道軸 XX 成為第一幾何軸（X），ZZ 為第三幾何軸（Z）。通道軸 V 仍為第二幾何軸（Y）。
N50 G17 G2 X20 I10 F1000	; X/Y 平面中的完整圓弧。通道軸 XX 及 V 移動
N60 GEOAX（2，W）	; 通道軸 W 成為第二幾何軸（Y）。
N80 G17 G2 X20 I10 F1000	; X/Y 平面中的完整圓弧。通道軸 XX 及 W 移動。
N90 GEOAX（）	; 重置至初始狀態。
N100 GEOAX（1，U，2，V，3，W）	; 通道軸 U 成為第一幾何軸（X），V 成為第二幾何軸（Y），而 W 為第三幾何軸（Z）。
N110 G1 X10 Y10 Z10 XX=25	; 通道軸 U、V、W 分別移動至位置 10。XX 為輔助軸，移動至位置 25。
N120 GEOAX（0，V）	; 自幾何軸群組中移除 V。U 及 W 仍為第一幾何軸（X）及第三幾何軸（Z）。仍未指派第二幾何軸（Y）。
N130 GEOAX（1，U，2，V，3，W）	; 通道軸 U 仍為第一幾何軸（X），V 成為第二幾何軸（Y），W 仍為第三幾何軸（Z）。
N140 GEOAX（3，V）	; V 成為第三幾何軸（Z），藉以覆寫 W 並使其脫離幾何軸群組。現在如同之前，未指派第二幾何軸（Y）。

說明

軸設定

使用以下機械參數指派幾何軸、特殊軸、通道軸、機械軸，及各軸類型的名稱：

MD20050 $MC_AXCONF_GEOAX_ASIGN_TAB（將幾何軸指派至通道軸）

MD20060 $MC_AXCONF_GEOAX_NAME_TAB（通道中的幾何軸名稱）

MD20070 $MC_AXCONF_MACHAX_USED（通道中有效的機械軸編號）

MD20080 $MC_AXCONF_CHANAX_NAME_TAB（通道中的通道軸名稱）

MD10000 $MN_AXCONF_MACHAX_NAME_TAB（機械軸名稱）

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX（指派主軸至機械軸）

參考資料：

功能手冊基本功能；座標軸、座標系統、框架（K2）

限制

- 無法在以下情況中改變幾何軸：
  - 轉換啟用
  - 主動曲線插補
  - 主動刀具半徑補正
  - 主動微調刀具補正
- 若幾何軸及通道軸的名稱相同，將無法變更該幾何軸。
- 涉及變更的軸無法進行超出單節限制的動作－例如類型 A 定位軸或為跟隨軸。
- 若在系統電源開啟時已啟用幾何軸（代表無法定義新的幾何軸），則僅能使用 GEOAX 指令來更換幾何軸。
- 若在執行輪廓表（CONTPRON、CONTDCON）時嘗試以 GEOAX 指令更換軸，則會輸出警報。

補充條件

更換後之軸狀態

藉由通道軸名稱變更操作後，可在幾何軸群組中將已更換的軸程式設計為輔助軸。

框架、保護區、工作區限制

在變更幾何軸後，會刪除所有框架、保護區及工作區限制。

極座標

以 GEOAX 更換幾何軸會使用 G17-G19 將類比設為層級變更，模態極座標的值會則為 0。

DRF、ZO

變更後，合適的手輪偏移（DRF）或外部零點偏移（ZO）仍然有效。

幾何軸之基本設定

GEOAX（）指令會呼叫幾何軸群組的基本設定。

系統在 POWER ON 後，以及變更為"參考點逼近"模式時，會自動變回基本設定。

刀長補正

變更操作後，啟用刀長補正也生效。然而，新增或被取代位置的幾何軸，會被認定尚未移動補正距離。該幾何軸的第一個動作指令，最終的移動距離是總刀長補正及程式設計移動距離的總和。

在替換操作後，幾何軸仍在軸群組中保有其位置，也保有其刀長補正的狀態。

啟用轉換的幾何軸設定

無法使用"可更換的幾何軸"函數來變更啟用轉換（以機械參數定義）適用的幾何軸設定。

若需配合轉換修改幾何軸，僅可使用額外轉換。

啟動轉換會刪除使用 GEOAX 變更的幾何軸設定。

若轉換的機械參數設定與變更後的幾何軸設定不符，則將優先採用轉換中的設定。

範例：

有一轉換啟用中。根據機械參數，應保留該轉換供重置用；但在重置時，應建立幾何軸的基本設定。在本情況中，會保留隨轉換定義的幾何軸設定。

14.3 軸容器（AXCTSWE、AXCTSWED）

功能

在旋轉式索引機台 / 多主軸機台上，持有工件的軸會由一個機台元件移至下一個。由於機台元件受限於不同的 NCU 通道，若變更狀態 / 位置，則需重新動態指派固定工件軸到對應的 NCU 通道。因此使用軸容器。

本地機台元件上只能同時啟用一個工件固定軸 / 主軸。軸容器結合所有可能連結至固定軸 / 主軸的連結，一次僅啟用一個軸來加工元件。

可使用透過設定資料（插槽數量）輸入的遞增值切換軸容器中的項目（“軸容器旋轉”），以變更軸容器中定義的可用軸。

可使用 AXCTSWE 或 AXCTSWED 指令由工件程式呼叫軸容器旋轉。

句法

AXCTSWE（<軸容器>）
AXCTSWED（<軸容器>）

意義

AXCTSWE	旋轉軸容器之指令 若可在控制系統中，啟用容器軸所有通道的可用訊號，將以儲存在 SD41700 $SN_AXCT_SWWIDTH[<軸容器>]的容器專屬遞增值來旋轉容器。
AXCTSWED	旋轉只在啟用通道有效的軸容器之指令（調試用指令版！） **注意：** 只有在容器內其他含軸的通道重置時，才會釋放容器內的輸入軸。
<軸容器>	軸容器辨識碼應移動。 可能的資料包括：
CT<軸容器>	軸容器編號附在 CT 字組之後。 範例：CT3
<容器名稱>	以 MD12750 $MN_AXCT_NAME_TAB 設定軸容器個別名稱。 範例：A_CONT3

其他資訊

軸容器

可經由軸容器指派。

- 本地軸及 / 或
- 連結軸

含連結軸的軸容器，為經由控制設定座標之跨 NCU 裝置（NCU 全域）。也可使用僅用於管理本地軸的軸容器。

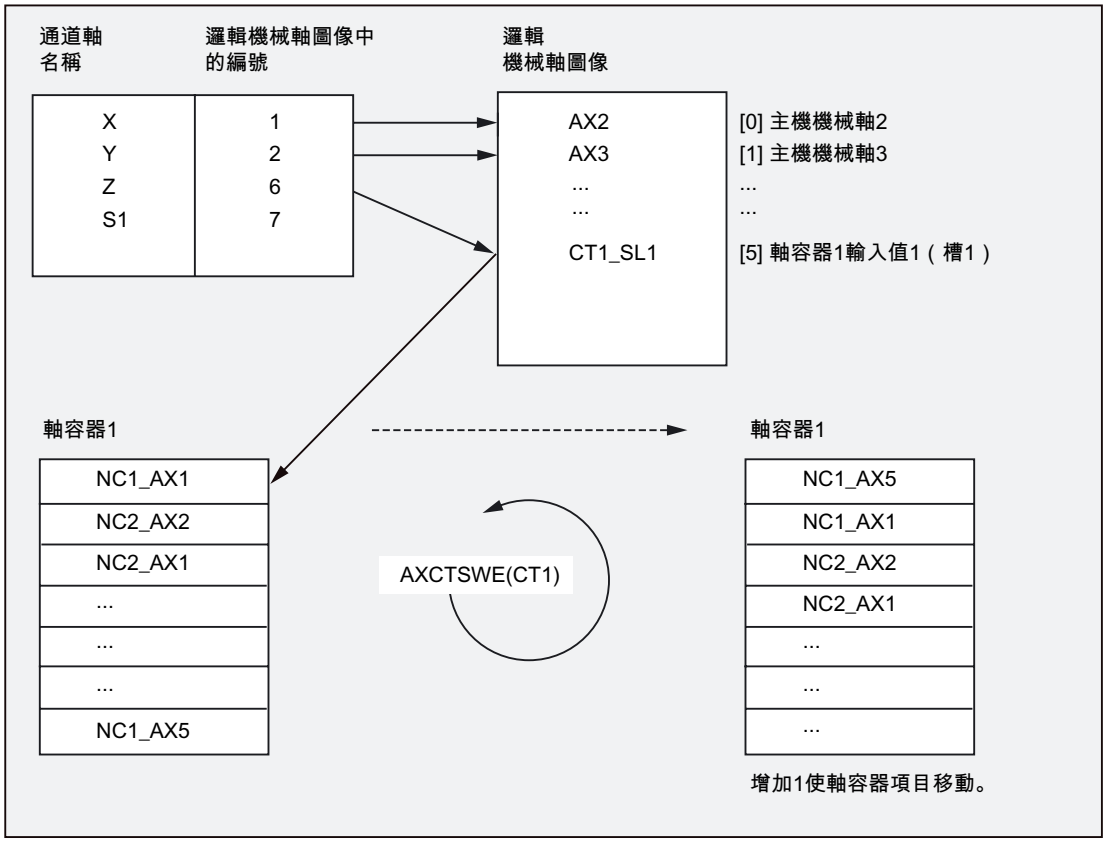
參考資料：
關於設定軸容器之詳細資訊，請參閱：
功能手冊，延伸功能；數個多重 NCU 的控制面板，分散式系統（B3）

可用條件

AXCTSWE（）

完成位置 / 狀態的加工後，每個指定容器已輸入其軸的通道，會發出容器旋轉的啟用訊號。若控制系統已收到容器軸所有通道的啟用訊號，便以儲存在 SD41700 $SN_AXCT_SWWIDTH[<容器編號>]的容器專屬遞增值來旋轉容器。

範例：



軸容器旋轉通過 1 後，將 NCU 1 上的通道軸 Z，而非 NCU 上的軸 AX1，指派給軸 AX5。

AXCTSWED（）

可使用指令版的 AXCTSWED（）簡化調試。軸容器經由儲存在 SD41700 $SN_AXCT_SWWIDTH[<容器編號>]的容器專屬遞增值，以有效的啟用通道旋轉。本呼叫僅在其他含軸容器的通道處於**重置**狀態時才能使用。

效果

軸容器旋轉後，所有 NCU 皆參與新軸指派，其通道參照經由邏輯機械軸影像之旋轉軸容器。

軸容器以未指定 GET/GETD 旋轉。

啟用軸容器旋轉時，所有軸指派至通道的軸容器以 GET 或 GETD 指派到通道。軸無法在軸容器旋轉前釋放。

說明

可使用參數設定此行為。請參閱機台製造商指示

說明

處於主執行軸狀態的軸（如 PLC 軸），使用未指定 GET / GETD **進行軸容器旋轉**，為了進行軸容器旋轉，該軸必須離開主執行狀態。

14.4 檢查顯示 NC 語言之範圍（STRINGIS）

功能

可使用 STRINGIS 指令，檢查 SINUMERIK 840D sl 所產生的 NC 語言範圍－包括啟用的 GUD/ 巨集定義及已安裝並啟用的循環程式－其實用性及其程式專屬特性。例如可在程式轉譯啟動時，檢查未啟用之函數的有效性。

傳回值係以 HMI 使用者介面編碼輸出，並包含基本資訊及含附加編碼的細節資訊。

句法

```
STRINGIS（STRING name） = return value with coding
```

在目前展開階段，識別待檢查之（STRING name）如下：

000 為未知。

100 為無法程式設計之 NC 語言指令。

所有啟用為選項或函數的可程式設計 NC 語言指令，皆使用

2xx 識別。值域下有更詳細的相關詳細資訊。

意義

機台製造商使用機械參數，定義如何處理及應使用之 NC 語言指令。

語言指令若已程式設計，且未啟用函數或在目前範圍內為未知，將發出警報訊息。在此情況下請參考機台製造商'的規格。

STRINGIS	檢查特屬於本指令的現有 NC 語言範圍及 NC 循環名稱、使用者變數、巨集及標籤名稱：是否存在、有效、已定義或生效。STRINGISNC 語言指令為整數類型變數。
Especially for STRINGIS	**NC 循環名稱**（生效循環） **GUD 變數** **LUD 變數** **巨集** **標籤名稱**
STRING 名稱	待檢查之 NC 語言範圍的變數辨識碼，及認定為 STRING 類型值的傳輸參數。

ISVAR 語言指令是 STRINGIS 指令的子集，可用於進行特定檢查。

NC 語言範圍

所有可用的，特別是不需要且未啟用的語言指令，仍為 SINUMERIK 電源線所知。
SINUMERIK solution line 待檢查之語言範圍，係依照預設的機械參數，並包含所有已知 / 剛認可之選項，或在目前 NC 語言範圍中啟用之函數。

NC language scope	NC 範圍包括： 現有所有 G 代碼群組之 **G 代碼**，如：G0、G1、G2、INVCW、POLY、ROT、KONT、SOFT、CUT2D、CDON、RMB、SPATH **DIN 或 NC 位址**，如 ADIS、RNDM、SPN、SR、MEAS **NC 語言函數**，如預先定義副程式 TANG（Faxis1..n，Laxis1..n，coupling factor）。 **NC 語言步驟**（**含**傳回值之預先定義步驟）如以參數傳輸 GETMDACT 之呼叫副程式。 **NC 語言步驟**（**不含**傳回值之預先定義步驟）如停用單一單節停止 SBLOF。 **NC 關鍵字**如：ACN、ACP、AP、RP、DEFINE、SETMS **機械參數**$MN 一般、$MA 軸向、$MC 通道專屬以及所有設定資料$S...及選配資料$O... **工件程式及同步動作中之** NC 系統變數$以及 **NC 計算參數** R。

傳回值

基本資訊 STRINGIS	傳回值已編碼。可再細分內含之基本資訊為 y，現有詳細資訊為 x。
編碼：	**測試結果**。是否在實際展開階段：
000	NCK 未發現 STRING 名稱。
100	STRING 名稱為語言指令，但**無法程式設計**，即本函數未啟用。
2xx	STRING 名稱為**可程式設計的**語言指令，即本函數已啟用。
y00	無法指派
y01 to y11	現有的已知詳細資訊之值域。
400	無 xx=01 或 xx=10 且非 G 代碼 G 或計算參數 R 之 NC 位址，請參閱註解（**1**）。

說明

檢查時，STRINGIS 若**未**找到其他編碼，則可程式設計相對應之 NC 語言指令，且適用 2xx 編碼。

詳細資訊的 2xx 值域

詳細資訊	**測試結果含義：**
200	無法轉譯
201	已定義 DIN 位址或 NC 位址，即名稱是否已由此處辨識出位址字母，請參閱註解（**1**）
202	已辨識出取自現有 G 代碼群組之 G 代碼。
203	顯示含傳回值及參數傳輸之 NC 語言函數。
204	顯示含傳回值及參數傳輸之 NC 語言函數。
205	顯示 NC 關鍵字。
206	顯示一般、軸向或通道專屬機械參數（$M...）、設定資料（$S...）或選項資料（$O...）。
207	顯示使用者變數，如以$...開頭之 NC 系統變數，或以 R 開頭之計算參數。
208	NCK 已載入循環名稱且循環程式已啟用，請參閱註解（**2**）。
209	辨識出定義名稱，並由全域使用者變數發現啟用之 GUD 變數（GUD 變數）。
210	找到巨集名稱、定義的名稱，及在巨集定義檔中啟用的巨集，請參閱註解（**3**）。
211	屬於本地使用者變數（LUD 變數），其名稱包含於目前程式中。

說明

關於個別傳回值的註解

（**1**）將固定、標準化位址辨識為 DIN 位址。下列幾何軸定義適用於含可手動倍率辨識碼之 NC 位址：

A、B、C 用於指定旋轉軸，為延伸保留 E 而
I、J、K、Q、U、V、W、X、Y、Z 則用於指定線性軸。

軸辨識碼可以位址延伸來進行程式設計，並可將其寫入以供測試，如：201 = STRINGIS（"A1"）。

無法以位址延伸寫入下列位址以供測試，且總是傳回固定值 400。

範例：400 = STRINGIS（"D"）或在
0 = STRINGIS（"M02"）中位址延伸之規格為 400 = STRINGIS（"M"）。

（**2**） 無法以 STRINGIS 檢查循環參數名稱。

（**3**） 將定義為巨集的 NC 位址字母 G、H、L、M 辨識為巨集。

不含位址延伸而含固定值 400 的有效 NC 位址

NC 位址輸入 D、F、G、H、R 及 L、M、N、O、P、S、T 皆有效。則

400	D 為刀具修正、刀刃編號（D 函數） F 為進給（F 函數） G 定義為 G 代碼（非本例中路徑條件） H 代表輔助函數（H 函數） R 定義為系統參數 L 代表副程式呼叫、M 代表附加函數、N 代表子單節、 O 可用於延伸、 P 代表程式執行數量、 S 代表主軸速度（S 函數）、 T 代表刀具編號（T 函數）。

可程式設計的輔助函數 T 之範例

程式設計	註解
T 定義為輔助函數，且永遠可程式設計	
400 = STRINGIS（"T"）	;
0 = STRINGIS（"T3"）	；不含位址延伸之傳回值 ；含位址延伸之傳回值

NC 語言 2xx 可程式設計範圍的其他檢查範例

程式設計	註解
X is defined as axis	; 軸為線性軸 X
201 = STRINGIS（"X"）	; 線性軸 X 的傳回值
201 = STRINGIS（"X1"）	線性軸 X1 的傳回值
A2 is an NC address with extension	含延伸之 NC 位址 A2
201 = STRINGIS（"A"）	NC 位址 A 之傳回值
201 = STRINGIS（"A2"）	含延伸之 NC 位址 A2
NVCW is a defined G code	INVCW 為 G 代碼衍生之 順時針插補。
202 = STRINGIS（"INVCW"）	已知 G 代碼之傳回值
GETMDACT is an NC language function	顯示 NC 語言函數 GETMDACT 。
203 = STRINGIS（"GETMDACT"）	GETMDACT 為 NC 語言函數
DEFINE is an NC key word	存在 DEFINE 關鍵字用於 巨集辨識。
205 = STRINGIS（"DEFINE"）	DEFINE 顯示為關鍵字
the $MC_GCODES_RESET_VALUES is channel-specific machine data	機械參數，$MC_GCODE_RESET_VALUES 存在。
206 = STRINGIS（"$MC_GCODE_RESET_VALUES"）	$MC_GCODE_RESET_VALUES 已 辨識為機械參數

程式設計	註解
$TC_DP3 is a system variable for the tool length components	存在用於刀長元件之 NC 系統變數$TC_DP3 。
207 = STRINGIS（"$TC_DP3"）	辨識$TC_DP3 為系統變數。
$TC_TP4 is a system variable for a tool size	存在用於刀具尺寸之 NC 系統變數$TC_TP4 。
207 = STRINGIS（"$TC_TP4"）	$TC_TP4 辨識為系統變數。
$TC_MPP4 is a system variable for the magazine space status	檢查刀庫管理
207 = STRINGIS（"$TC_MPP4"）	刀庫管理啟用中
0 = STRINGIS（"$TC_MPP4"）	無法使用 刀庫管理（4）
MACHINERY_NAME is defined as GUD variable	全域使用者變數定義為 MACHINERY_NAME。
209 = STRINGIS（"MACHINERY_NAME"）	找到作為 GUD 的 MACHINERY_NAME
LONGMACRO is defined as macro	巨集名稱為 LONGMACRO
210 = STRINGIS（"LONGMACRO"）	巨集辨識為 LONGMACRO
MYVAR is defined as LUD variable	本地使用者變數已命名 MYVAR
211 = STRINGIS（"MYVAR"）	LUD 變數包含在目前 程式中，名稱為 MYVAR
X，Y，Z is a command not known in the NC	X、Y、Z 為未知的語言指令 亦非 GUD/ 巨集 / 循環名稱
0 = STRINGIS（"XYZ"）	未知 STRING 名稱 X、Y、Z

（4） 對於刀庫管理的系統參數，下列特性特別適用：無論設為設定 NC 語言範圍的機械參數使用何值，若未啟用函數，則 STRINGIS 的結果值將永遠為 0。

14.5 函數呼叫 ISVAR 及讀取機械參數陣列索引

功能

ISVAR 指令為以 NC 語言定義之函數，具有

- 類型 BOOL 之函數值
- 類型 STRING 之傳輸參數

若傳輸參數包含 NC 中已知之變數（機械參數、設定資料、系統變數、GUD 等一般變數），則 ISVAR 指令傳回 TRUE。

句法

```
ISVAR（<變數識別碼>）
ISVAR(<識別碼>,[<值>,<值>])
```

意義

<變數識別碼>	字串類型的傳輸參數可為無維度、一維或二維。
<識別碼>	含已知變數的辨識碼，使用或不使用陣列索引為機械參數、設定資料、系統變數或一般變數。 **副檔名：** 一般及通道專屬的機械參數，即使未指定索引值，也會讀取陣列的第一個元素。
<值>	類型 BOOL 之函數值

檢查

根據傳輸參數進行下列檢查：

- 辨識碼是否存在
- 是否為一維或二維陣列
- 陣列索引是否允許

僅在所有檢查結果皆為肯定時才傳回 TRUE。若檢查結果為否定或發生句法錯誤，則傳回 FALSE。軸向變數視為座標軸名稱的索引，但不檢查該變數。

副檔名：不使用索引，讀取機械參數及設定資料陣列。

若無供**一般及通道專屬**機械參數使用的索引，將**不再**輸出警報 12400“通道%1 單節%2 陣列%3 元素不存在”。

至少仍需為**軸專屬**機械參數程式設計**軸索引**。否則，會發出警報 12400。

範例：函數呼叫 ISVAR

程式碼	註解
DEF INT VAR1	
DEF BOOL IS_VAR=FALSE	; 傳輸參數為一般變數
N10 IS_VAR=ISVAR（"VAR1"）	; 該情況下的 IS_VAR 為 TRUE
DEF REAL VARARRAY[10，10]	
DEF BOOL IS_VAR=FALSE	; 各種句法版本
N20 IS_VAR=ISVAR（"VARARRAY[，]"）	; IS_VAR 為 TRUE， 含 2 維陣列
N30 IS_VAR=ISVAR（"VARARRAY"）	; IS_VAR 為 TRUE，變數存在
N40 IS_VAR=ISVAR（"VARARRAY[8，11]"）	; IS_VAR 為 FALSE，不允許陣列索引
N50 IS_VAR=ISVAR（"VARARRAY[8，8"）	; IS_VAR 為 FALSE，遺失“]”的 句法錯誤
N60 IS_VAR=ISVAR（"VARARRAY[，8]"）	; IS_VAR 為 TRUE，允許陣列索引
N70 IS_VAR=ISVAR（"VARARRAY[8，]"）	; IS_VAR 為 TRUE
DEF BOOL IS_VAR=FALSE	; 傳輸參數是一種機械參數
N100 IS_VAR=ISVAR（"$MC_GCODE_RESET_VALUES[1]"	; IS_VAR 為 TRUE
DEF BOOL IS_VAR=FALSE	; 傳輸參數為一種系統變數
N10 IS_VAR=ISVAR（"$P_EP"）	; 此情況下 IS_VAR 為 TRUE
N10 IS_VAR=ISVAR（"$P_EP[X]"）	; 此情況下 IS_VAR 為 TRUE

範例：使用及不使用索引，讀取機械參數陣列。

為以下項目讀取第一個元素

```
R1=$MC_EXTERN_GCODE_RESET_VALUES
```

如前所述，這對應於

```
R1=$MC_EXTERN_GCODE_RESET_VALUES[0]
```

或讀取第一個元素

```
R1=$MA_POSTCTRL_GAIN[X1]
```

如前述之對應

```
R1=$MA_POSTCTRL_GAIN[0，X1]
```

也會為以下項目讀取同步動作中第一個元素

```
WHEN TRUE DO $R1 = $MC_EXTERN_GCODE_RESET_VALUES
```

如前述之對應

```
WHEN TRUE DO $R1 = $MC_EXTERN_GCODE_RESET_VALUES[0]
```

且**不會先讀取**並發出警報 12400。

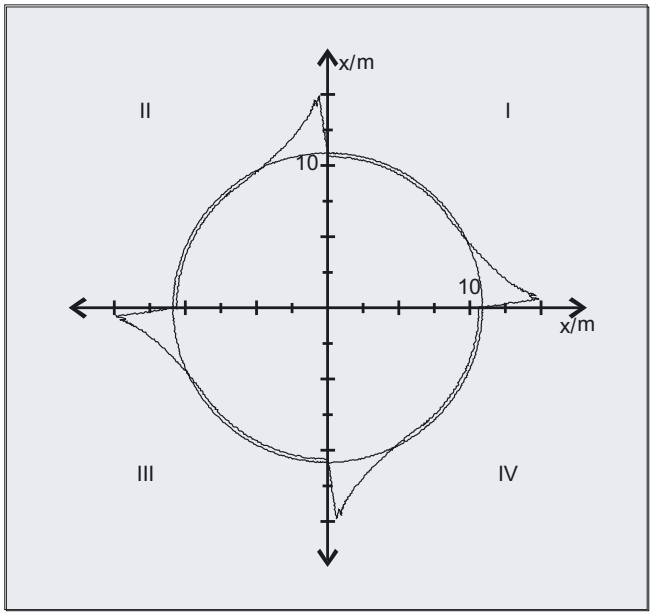
仍為以下項目發出警報 12400

```
R1=$MA_POSTCTRL_GAIN
```

14.6 學習補正特性（QECLRNON、QECLRNOF）

功能

象限錯誤補正（QEC）減少因機械非線性狀況（如摩擦、背隙）或扭力，而在移動方向反轉上發生的輪廓錯誤。在神經網路的基礎上，可在學習階段由控制來手動倍率最佳補正資料，以自動判斷補正特性。最多可同時在四個軸上學習。



句法

```
QECLRNON
QECLRNOF
```

啟動學習操作：QECLRNON

可在 NC 程式中，使用 QECLRNON 指令指定軸來啟動實際學習操作：

```
QECLRNON （X1，Y1，Z1，Q）
```

僅在指令啟用時才變更特性。

停用學習：QECLRNOF

完成所需軸的學習動作後，可使用 QECLRNOF 同時停用所有軸的學習程序。

意義

QECLRNON （axis 1，…4）	啟動“學習象限錯誤補正”函數
QECLRNO	停用“學習象限錯誤補正”函數
QECLRN.SPF	學習循環
QECDAT.MPF	指派系統變數及參數化學習循環的樣本 NC 程式
QECTEST.MPF	循環性測試的樣本 NC 程式

說明

在 NC 程式輔助下，產生學習程序所需的移動動作。在程式中以學習循環的形式儲存學習動作。

首次教導

在控制啟動期間的初始學習階段中，使用標準 PLC 程式磁碟中所含的樣本 NC 程式，以教導移動及指派 QEC 系統變數：

重新學習

可以後續學習最佳化學習特性。可使用使用者記憶體中儲存的資料為最佳化的基礎。使樣本 NC 程式配合您的需求來執行最佳化。

可能需為“重新學習”變更學習循環參數（例如，QECLRN.SPF）。

- 設定“學習模式” = 1
- 必要時請減少“學習通過次數”
- 必要時請啟動“模態學習”並定義區域限制。

14.7 由工件程式相互呼叫視窗（MMC）

功能

您可由工件程式使用 MMC 指令，在 HMI 上顯示使用者自訂的對話視窗（對話方塊顯示）。

對話視窗的外觀係以純文字設定定義（循環目錄中的 COM 檔案），而 HMI 系統軟體維持不變。

無法同時在不同的通道中呼叫使用者自訂的對話視窗。

句法

```
MMC（CYCLES，PICTURE_ON，T_SK.COM，BILD，MGUD.DEF，BILD_3.AWB，TEST_1，
A1"，"S"）
```

意義

MMC	由 HMI 上的工件程式相互呼叫對話方塊視窗
CYCLES	在操作區裡建置設定的使用者對話方塊。
PICTURE_ON 或 PICTURE_OFF	指令：顯示選擇或顯示取消選擇。
T_SK.COM	Com 檔案：對話方塊顯示檔案名稱（使用者循環）。在此定義對話方塊顯示設計。對話方塊畫面用於顯示使用者變數及 / 或註解文字。
DISPLAY	對話方塊顯示名稱：透過對話方塊顯示名稱，選取各個顯示。
MGUD.DEF	在讀取 / 寫入變數時，輸入使用者資料定義檔。
PICTURE_3.AWB	圖形檔
TEST_1	顯示時間或確認變數。
A1	文字變數...”，
"S"	確認模式：同步，透過“OK”軟鍵確認。

參考資料

請參考“調試手冊”，以取得關於程式設計 MMC 指令的詳細資訊（包括程式設計範例）。

14.8 程式執行時間 / 工件計數器

14.8.1 程式執行時間 / 工件計數器（概觀）

提供關於程式執行時間及工件計數器之資訊，以支援機台刀具操作員。

可在 NC 及 / 或 PLC 程式中做為系統變數處理本資訊。亦可在操作員介面上顯示本資訊。

14.8.2 程式執行時間一章

功能

“程式執行時間”功能提供內部 NC 計時器以監控技術處理，該資訊可透過 NC 及通道專屬系統變數，讀入至工件程式及同步動作。

執行時間量測之觸發（$AC_PROG_NET_TIME_TRIGGER）為該函數唯一可寫入的系統變數，並使用於量測選取的程式區段。這表示可透過 NC 程式中寫入該觸發，以啟動及停用時間量測。

系統變數	意義	活動
NC 專屬		
$AN_SETUP_TIME	距離上次以預設值（"冷開機"）控制開機的時間，以分鐘計算： 每次以預設值控制開機，會自動重置為"0"。	永遠有效
$AN_POWERON_TIME	距離上次正常控制開機（"暖開機"）的時間，以分鐘計算。 每次正常控制開機，會自動重置為"0"。	
通道專屬		
$AC_OPERATING_TIME	在自動模式中，NC 程式的總執行時間，以秒計算。 每次控制開機，該值會自動重置為"0"。	• 透過 MD27860 啟動 • 僅有 AUTOMATIC 模式
$AC_CYCLE_TIME	選定 NC 程式的執行時間，單位為秒。 每次有新的 NC 程式開機，該值會自動重置為"0"。 可設定 MD27860 來確保此值會被刪除，就算以 GOTOS 跳過程式開頭，或在非同步副程式的事件中，以及 PROG_EVENTs 啟動時。	
$AC_CUTTING_TIME	以秒為單位的處理時間 所有介於 NC 啟動與程式結束 / NC 重置之間無快速移動且刀具有效時的所有程式路徑軸（至少一個有效）之執行時間。出現停頓時間時，測量會中斷。 每次以預設值控制開機，該值會自動重置為"0"。	
$AC_ACT_PROG_NET_TIME	目前的 NC 程式之實際執行時間，以秒為單位。 當新的 NC 程式開始時，會自動重置為"0"。	• 永遠有效 • 僅有 AUTOMATIC 模式
$AC_OLD_PROG_NET_TIME	剛以 M30 正確終止的程式執行時間，以秒表示	

系統變數	意義		活動
$AC_OLD_PROG_NET_TIME_COUNT	變更為$AC_OLD_PROG_NET_TIME_COUNT 在 POWER ON（開機）後，$AC_OLD_PROG_NET_TIME_COUNT 在“0”。 若新寫入控制至$AC_OLD_PROG_NET_TIME，則永遠增加$AC_OLD_PROG_NET_TIME_COUNT。		
$AC_PROG_NET_TIME_TRIGGER	執行時間量測之觸發：		僅有 AUTOMATIC 模式
	0	中立狀態 觸發未啟用。	
	1	離開 結束量測，並將數值由 $AC_ACT_PROG_NET_TIME 複製到 $AC_OLD_PROG_NET_TIME。 $AC_ACT_PROG_NET_TIME 設為“0”並繼續執行。	
	2	開始 開始量測，並因此將 $AC_ACT_PROG_NET_TIME 設為“0”。未變更 $AC_OLD_PROG_NET_TIME。	
	3	停止 量測停止。不變更$AC_OLD_PROG_NET_TIME 並保持$AC_ACT_PROG_NET_TIME 恆定，直到繼續執行	
	4	繼續 繼續執行量測，即繼續先前停止的量測。 $AC_ACT_PROG_NET_TIME 繼續執行。未變更 $AC_OLD_PROG_NET_TIME。	
POWER ON（開機）後，將所有系統變數重置為 0!			

說明

可選擇計時器的啟用，與在特定情況下，有效時機機制的行為（例如，當測試執行進給率生效時，在程式測試期間，等等）會使用機械參數 MD27860 $MC_PROCESSTIMER_MODE（→ machine manufacturer）來控制。

參考文獻：
功能手冊基本功能；BAG、通道、程式操作、重置回應（K1），章節：程式執行時間一章

說明

工件的剩餘時間

若連續生產同樣的工件，則計時器的值：

- 處理最後一個工件所花的時間（請參考$AC_OLD_PROG_NET_TIME）

及

- 目前的處理時間（請參考$AC_ACT_PROG_NET_TIME）

可用來決定工件的剩餘時間。

除了目前的處理時間以外，剩餘的時間也會顯示在操作介面上。

注意
使用 STOPRE 系統變數$AC_OLD_PROG_NET_TIME 及$AC_OLD_PROG_NET_TIME_CTR 不會產生任何不明的前置處理停止。若系統變數值來自先前的程式執行，則在工件程式中使用時便不重要。然而，若因$AC_OLD_PROG_NET_TIME 改變頻繁，而頻頻寫入執行時間量測之觸發（$AC_PROG_NET_TIME_TRIGGER），則應在工件程式中明確使用 STOPRE。

一般條件

- **單節搜尋**
  沒有程式執行時間透過單節搜尋決定。
- **REPOS**
  REPOS 的處理時間會被加在目前的處理時間上（$AC_ACT_PROG_NET_TIME）。

範例

範例 1：量測"mySubProgrammA"的期間

程式碼

```
...
N50 DO $AC_PROG_NET_TIME_TRIGGER=2
N60 FOR ii= 0 TO 300
N70 mySubProgrammA
N80 DO $AC_PROG_NET_TIME_TRIGGER=1
N95 ENDFOR
N97 mySubProgrammB
N98 M30
```

在程式處理 N80 行後，"mySubProgrammA"的淨執行時間置於 $AC_OLD_PROG_NET_TIME。

來自$AC_OLD_PROG_NET_TIME 的數值：

- 在 M30 之後仍舊保留。
- 在每次執行迴圈時更新。

範例 2：量測“mySubProgrammA”及“mySubProgrammC”的期間

程式碼

```
...
N10 DO $AC_PROG_NET_TIME_TRIGGER=2
N20 mySubProgrammA
N30 DO $AC_PROG_NET_TIME_TRIGGER=3
N40 mySubProgrammB
N50 DO $AC_PROG_NET_TIME_TRIGGER=4
N60 mySubProgrammC
N70 DO $AC_PROG_NET_TIME_TRIGGER=1
N80 mySubProgrammD
N90 M30
```

14.8.3 工件計數器

功能

"工件計數"函數使可用的數個技術器，可用於控制中特定的輪廓工件上。

該計數器以通道專屬系統變數形式存在，可在數值範圍 0 到 999，999，999 間讀寫存取。

系統變數	意義
$AC_REQUIRED_PARTS	待生產的工件數量（工件的設定點號碼） 在此計數器中，您可以定義工件數量，在此實際的工件計數器 $AC_ACTUAL_PARTS 重設為"0"。
$AC_TOTAL_PARTS	所有完成的工件（實際的工件加總） 此計數器會指定了從開始到現在所生產的所有工件的總數量。當以預設值控制開機，該值只會自動重置為"0"。
$AC_ACTUAL_PARTS	所有完成的工件（實際的工件加總） 此計數器會記錄了從開始到現在所生產的所有工件的總數量。在 $AC_REQUIRED_PARTS > 0 的條件下，當達到所需的工件數量（$AC_REQUIRED_PARTS）時，該計數器會自動重置為零。
$AC_SPECIAL_PARTS	使用者選擇的工件數量 此計數支援使用者自訂工件計數。當到達了工件的設定點號碼（$AC_REQUIRED_PARTS），可定義輸出一個警報。使用者必須自行重置計數器。

說明

在控制系統以預設值啟動時，所有工件計數器皆設為零，並獨立於啟動而讀寫計數器。

說明

通道專屬機械參數可用於控制計數器啟用、計數器重置時間以及計數演算法。

說明

以使用者自訂 M 指令進行工件計數

可設定機械參數，因此用於數個工件輪廓的計數脈衝，便可透過使用者自訂 M 指令來觸發，而不用等到程式結尾（M2/M30）。

參考

若要有關"工件計數"函數的進一步資訊，請參考：

- 功能手冊，基本功能；BAG，通道，程式操作，重置回應（K1），章節：工件計數器

14.9 警報（SETAL）

功能

可在 NC 程式中設定警報。警報顯示於操作員介面中的另一個欄位。警報永遠伴隨控制器依警報類別所發出的回應。

參考資料：
關於警報回應的其他資訊，請參考“調試手冊”。

句法

```
SETAL（<警報編號>）
SETAL(<警報編號>,<字元字串>)
```

意義

SETAL	程式設計警報的關鍵字。 SETAL 必須在獨立 NC 單節中進行程式設計。
<警報編號>	INT 類別變數。內含警報編號。 警報編號有效範圍介於 60000 及 69999 間，其中保留 60000 到 64999 供 SIEMENS 循環使用，65000 到 69999 供使用者使用。
<字元字串>	在程式設計使用者循環警報時，可另外指定最多含四個參數的字元字串。 可在這些參數中定義變數使用者文字。 然而，可使用下列預先定義的參數：

參數	意義
%1	通道編號
%2	單節編號，標籤
%3	循環警報的文字索引
%4	附加警報參數

說明
警報文字必須在操作員介面中設定。

範例

程式碼	註解
...	
N100 SETAL （65000）	; 設定警報 No. 65000
...	

15 使用者材料移除程式

15.1 材料移除的支援函數

功能

為材料移除提供預先程式設計的材料移除程式。此外，您可使用下列函數產生自己的材料移除程式：

- 產生輪廓表（CONTPRON）
- 產生編碼輪廓表（CONTDCON）
- 停用輪廓準備（EXECUTE）
- 決定兩輪廓元素間的交點（INTERSEC）
  （僅適用使用 CONTPRON 產生的表）
- 按單節執行表的輪廓元素（EXECTAB）
  （僅適用使用 CONTPRON 產生的表）
- 計算圓弧資料（CALCDAT）

說明
該函數不僅可用在材料移除，也可用在所有地方。

先決條件

呼叫 CONTPRON 或 CONTDCON 函數前，需完成下列項目：

- 需逼近允許無碰撞加工的起點。
- 需以 G40 停用切削半徑補正。

15.2 產生輪廓表（CONTPRON）

功能

使用 CONTPRON 指令啟動輪廓準備。不執行之後呼叫的 NC 單節，而是分隔成個別的移動並存入輪廓表。每個輪廓元素對應於輪廓表二維陣列中的一列。傳回凸紋切削的次數。

句法

啟動輪廓準備：
CONTPRON(<輪廓表>,<加工類型>,<凸紋切削>,
<加工方向>)
停用輪廓準備並回到正常執行模式：
EXECUTE（<ERROR>）
請參閱“停用輪廓準備（EXECUTE）”

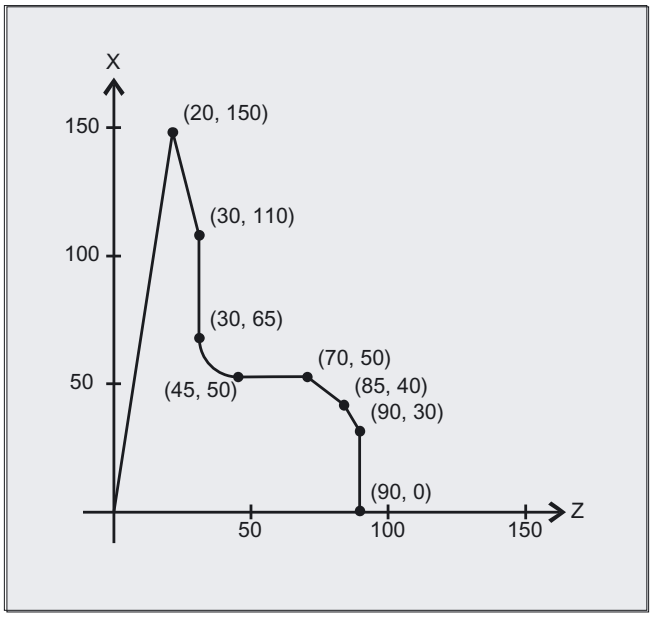
意義

CONTPRON	啟動輪廓準備以產生輪廓表之指令
<輪廓表>	輪廓表名稱
<加工類型>	加工類型參數
	類型： CHAR
	值： “G” 縱向車削：內部加工
	“L” 縱向車削：外部加工
	“N” 平面車削：內部加工
	“P” 平面車削：外部加工
<凸紋切削>	發生凸紋切削元件數量的結果變數
	類型： INT
<加工方向>	加工方向之參數
	類型： INT
	值： 0 輪廓準備，向前（預設值）
	1 輪廓準備，雙向

範例 1

產生輪廓表，使用：

- 名稱“KTAB”
- 最多 30 個輪廓元素（圓弧、直線）
- 發生凸紋切削元件數量的一變數
- 錯誤訊息的一變數



NC 程式：

程式碼	註解
N10 DEF REAL KTAB[30, 11]	; 含 KTAB 名稱及最多 30 個輪廓元素、參數值 11（表欄位數）的輪廓表，為固定數量。
N20 DEF INT ANZHINT	; 使用名稱 ANZHINT 的凸紋切削元件數量之變數。
N30 DEF INT ERROR	; 錯誤反饋訊號的變數（0=無錯誤，1=錯誤）。
N40 G18	
N50 CONTPRON（KTAB, "G", ANZHINT）	; 啟動輪廓準備。
N60 G1 X150 Z20	; N60 至 N120：輪廓說明
N70 X110 Z30	
N80 X50 RND=15	
N90 Z70	
N100 X40 Z85	
N110 X30 Z90	
N120 X0	
N130 EXECUTE（ERROR）	; 停止填入輪廓表，改為正常程式操作。
N140 …	; 繼續處理該表。

輪廓表 KTAB：

索引 直線	欄									
（0）	（1）	（2）	（3）	（4）	（5）	（6）	（7）	（8）	（9）	（10）
7	7	11	0	0	20	150	0	82.40535663	0	0
0	2	11	20	150	30	110	-1111	104.0362435	0	0
1	3	11	30	110	30	65	0	90	0	0
2	4	13	30	65	45	50	0	180	45	65
3	5	11	45	50	70	50	0	0	0	0
4	6	11	70	50	85	40	0	146.3099325	0	0
5	7	11	85	40	90	30	0	116.5650512	0	0
6	0	11	90	30	90	0	0	90	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

欄位內容說明：

(0) 下一個輪廓元素的指標（指向該欄位之列數）

(1) 前一個輪廓元素的指標

(2) 將動作的輪廓模式編碼

X = abc 的可能值

a = 10^2 G90 = 0 G91 = 1

b = 10^1 G70 = 0 G71 = 1

c = 10^0 G0 = 0 G1 = 1 G2 = 2 G3 = 3

(3), (4) 輪廓元素起點

（3）=橫座標、（4）=目前平面的縱座標

(5), (6) 輪廓元素起點

（5）=橫座標、（6）=目前平面的縱座標

(7) 最大 / 最小指示碼：辨識出本機輪廓中的最大值及最小值。

(8) 輪廓元素及橫座標（用於縱向加工）或縱座標（用於平面切削）間的最大值。角度依已程式設計的加工類型而定。

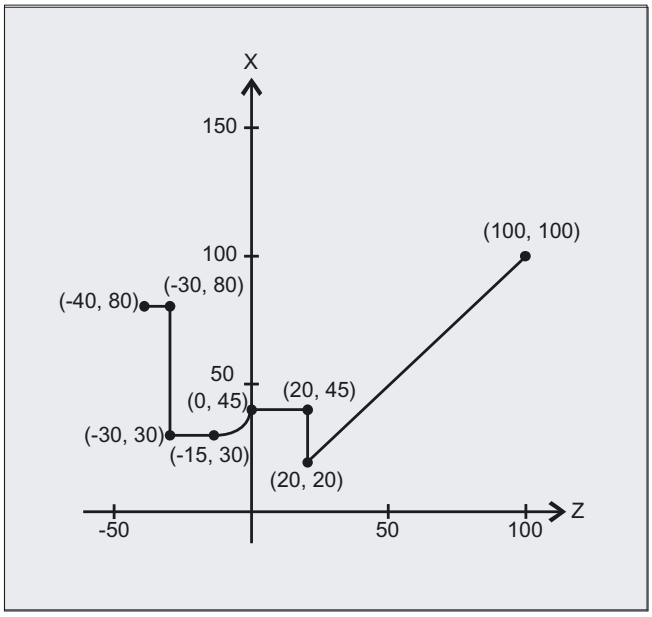
(9), (10) 輪廓元素的中心點座標，若其為圓弧單節。

（9） = 橫座標，（10） = 縱座標

範例 2

產生輪廓表，使用

- 名稱 KTAB
- 最多 92 個輪廓元素（圓弧、直線）
- 模式：縱向車削，外部加工
- 準備，向前及向後



NC 程式：

程式碼	註解
N10 DEF REAL KTAB[92, 11]	; 含 KTAB 名稱及最多 92 個輪廓元素、參數值 11 的輪廓表，為固定數量。
N20 DEF CHAR BT="L"	; CONTPRON 的模式：縱向車削，外部加工
N30 DEF INT HE=0	; 凸紋切削元件數量=0
N40 DEF INT MODE=1	; 準備，向前及向後
N50 DEF INT ERR=0	; 錯誤反饋訊號
...	
N100 G18 X100 Z100 F1000	
N105 CONTPRON（KTAB, BT, HE, MODE）	; 啟動輪廓準備。
N110 G1 G90 Z20 X20	
N120 X45	
N130 Z0	
N140 G2 Z-15 X30 K=AC（-15） I=AC（45）	
N150 G1 Z-30	
N160 X80	
N170 Z-40	
N180 EXECUTE（ERR）	; 停止填入輪廓表，改為正常程式操作。
...	

輪廓表 KTAB：

完成輪廓準備後，可雙向使用輪廓。

索引	欄										
直線	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
0	6[1)]	7[2)]	11	100	100	20	20	0	45	0	0
1	0[3)]	2	11	20	20	20	45	-3	90	0	0
2	1	3	11	20	45	0	45	0	0	0	0
3	2	4	12	0	45	-15	30	5	90	-15	45
4	3	5	11	-15	30	-30	30	0	0	0	0
5	4	7	11	-30	30	-30	45	-1111	90	0	0
6	7	0[4)]	11	-30	80	-40	80	0	0	0	0
7	5	6	11	-30	45	-30	80	0	90	0	0
8	1[5)]	2[6)]	0	0	0	0	0	0	0	0	0
	...										
83	84	0[7)]	11	20	45	20	80	0	90	0	0
84	90	83	11	20	20	20	45	-1111	90	0	0
85	0[8)]	86	11	-40	80	-30	80	0	0	0	0
86	85	87	11	-30	80	-30	30	88	90	0	0
87	86	88	11	-30	30	-15	30	0	0	0	0
88	87	89	13	-15	30	0	45	-90	90	-15	45
89	88	90	11	0	45	20	45	0	0	0	0
90	89	84	11	20	45	20	20	84	90	0	0
91	83[9)]	85[10)]	11	20	20	100	100	0	45	0	0

欄位內容說明及第 0、1、6、8、83、85 及 91 行之註解。

適用範例 1 中對欄位內容的說明。

永遠在表列 0：

1） 前置：第 n 行包含輪廓結尾（向前）

2） 後續：第 n 行為輪廓表結尾（向前）

一旦輪廓元素中每個皆向前：

3） 前置：輪廓開始（向前）

4） 後續：輪廓結束（向前）

永遠在輪廓表結束（向前）+1 行：

5） 前置：凸紋切削次數（向前）

6） 後續：凸紋切削次數（向後）

一旦輪廓元素中每個皆向後：

7） 後續：輪廓結束（向後）

8） 前置：輪廓開始（向後）

永遠在表的最後一行：

9） 前置：第 n 行為輪廓表開始（向後）

10） 後續：第 n 行包含輪廓開始（向後）

其它資訊

認可的移動指令、座標系統

可在輪廓程式設計中使用下列 G 指令：

- G 群組 1：G0、G1、G2、G3

除此之外，下列也可能：

- 倒圓角與倒角
- 圓弧程式設計使用 CIP 及 CT

曲線、多項式及螺紋函數會產生錯誤。

在 CONTPRON 及 EXECUTE 間不可透過啟動框架來變更座標系統。此限制亦適用於 G70 及 G71 或 G700 及 G710 間的變更。

若準備輪廓表時，以 GEOAX 更換幾何軸，會發生警報。

凸紋切削元件

可在副程式或個別單節中，執行個別凸紋切削元件的輪廓說明。

材料移除獨立於已程式設計的輪廓方向

以 CONTPRON 展開輪廓準備，如此在呼叫後，可用輪廓表獨立於已程式設計的方向。

15.3 產生編碼輪廓表（CONTDCON）

功能

以 CONTDCON 啟動輪廓準備時，下列呼叫的 NC 單節會以編碼形式儲存於六欄輪廓表，以最佳化記憶體使用。每個輪廓元素對應於輪廓表的一列。熟悉下述編碼規則後，例如，您可從表列中的循環結合 DIN 代碼程式。輸出點的資料，用編號 0 存入表列中。

句法

啟動輪廓準備：

CONTDCON(<輪廓表>,<加工方向>)

停用輪廓準備並回到正常執行模式：

EXECUTE（<ERROR>）

請參閱“停用輪廓準備（EXECUTE）”

意義

CONTDCON	啟動輪廓準備以產生編碼輪廓表之指令		
<輪廓表>	輪廓表名稱		
<加工方向>	加工方向的參數		
	類型：	INT	
	值：	0	根據輪廓單節順序的輪廓準備（預設值）
		1	不允許

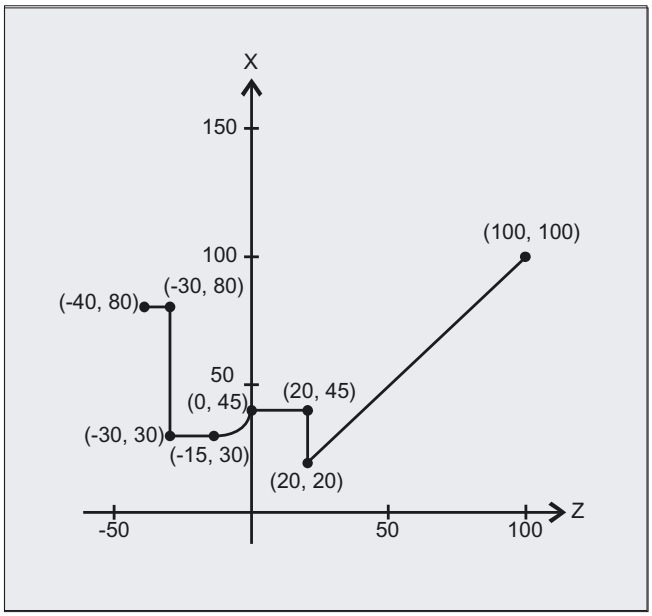
說明

在程式區段中欲包含至該表，為 CONTDCON 所認可的 G 代碼，比為 CONTPRON 認可的還要廣泛。此外，為各輪廓區段儲存進給率及進給率類型。

範例

產生輪廓表，使用：

- 名稱“KTAB”
- 輪廓元素（圓弧、直線）
- 模式：車削
- 加工方向：向上



NC 程式：

程式碼	註解
N10 DEF REAL KTAB[9，6]	; 含名稱 KTAB 及 9 個列表的輪廓表。允許 8 個輪廓集。參數值 6（表中欄位數）為固定尺寸。
N20 DEF INT MODE = 0	; 加工方向的變數標準值 0：僅在輪廓已程式設計的方向中。
N30 DEF INT ERROR = 0	; 錯誤反饋訊號的變數。
...	
N100 G18 G64 G90 G94 G710	
N101 G1 Z100 X100 F1000	
N105 CONTDCON （KTAB，MODE）	; 呼叫輪廓準備（可忽略 MODE）。
N110 G1 Z20 X20 F200	; 輪廓說明。
N120 G9 X45 F300	
N130 Z0 F400	
N140 G2 Z-15 X30 K=AC（-15） I=AC（45）F100	
N150 G64 Z-30 F600	
N160 X80 F700	
N170 Z-40 F800	
N180 EXECUTE（ERROR）	; 停止填寫輪廓表，改為正常程式操作。
...	

輪廓表 KTAB：

	欄位索引					
	0	1	2	3	4	5
行索引	輪廓模式	終點 橫座標	終點 縱座標	中心點 橫座標	中心點 縱座標	進給速率
0	30	100	100	0	0	7
1	11031	20	20	0	0	200
2	111031	20	45	0	0	300
3	11031	0	45	0	0	400
4	11032	-15	30	-15	45	100
5	11031	-30	30	0	0	600
6	11031	-30	80	0	0	700
7	11031	-40	80	0	0	800
8	0	0	0	0	0	0

欄位內容說明：

起點的 0 行編碼：

欄 0：10^0（個位數）：G0 = 0
10^1（十位數）：G70 = 0，G71 = 1，G700 = 2，G710 = 3
欄 1：橫座標起點
欄 2：縱座標起點
欄 3-4：0
欄 5：表中最後一個輪廓片段的行索引

1 至 n 行：**輪廓片段**項目

欄 0：10^0（個位數）：G0 = 0，G1 = 1，G2 = 2，G3 = 3
10^1（十位數）：G70 = 0，G71 = 1，G700 = 2，G710 = 3
10^2（百位數）：G90 = 0，G91 = 1
10^3（千位數）：G93 = 0，G94 = 1，G95 = 2，G96 = 3
10^4（萬位數）：G60 = 0，G44 = 1，G641 = 2，G642 = 3
10^5（十萬位數）：G9 = 1
欄 1：終點橫座標
欄 2：終點縱座標
欄 3：圓弧插補的中心點橫座標
欄 4：圓弧插補的中心點縱座標
欄 5：進給速率

其它資訊

認可的移動指令、座標系統

可在輪廓程式設計中，使用下列 G 群組及 G 指令：

G 群組 1：	G0、G1、G2、G3
G 群組 10：	G60，G64，G641，G642
G 群組 11：	G9
G 群組 13：	G70，G71，G700，G710
G 群組 14：	G90，G91
G 群組 15：	G93，G94，G95，G96，G961

除此之外，下列也可能：

- 倒圓角與倒角
- 圓弧程式設計使用 CIP 及 CT

曲線、多項式及螺紋函數會產生錯誤。

在 CONTDCON 及 EXECUTE 間不可透過啟動框架來變更座標系統。此限制亦適用於在 G70 及 G71 或 G700 及 G710 間的變更。

若準備輪廓表時，以 GEOAX 更換幾何軸，會發生警報。

加工方向

使用 CONTDCON 產生的輪廓表，可用於輪廓已程式設計的方向上之材料移除。

15.4 決定兩輪廓元素間的交點（INTERSEC）

功能

INTERSEC 從使用 CONTPRON 產生的輪廓表，決定兩個正常化輪廓元素間的交點。

句法

```
<狀態>=INTERSEC(<contour table_1>[<contour element_1>],
<contour table_2>[<contour element_2>],<交點>,<加工類型>)
```

意義

INTERSEC	從以 CONTPRON 產生的輪廓表，決定兩輪廓元素間的交點關鍵字。		
<狀態>	交點狀態的變數		
	類型：	BOOL	
	值：	TRUE	找到交點
		FALSE	未找到交點
<contour table_1>	第一輪廓表名稱		
<contour element_1>	第一輪廓表輪廓元素的數量		
<contour table_2>	第二輪廓表名稱		
<contour element_2>	第二輪廓表輪廓元素的數量		
<交點>	生效平面中的交點座標（G17 / G18 / G19）		
	類型：	REAL	
<加工類型>	加工類型的參數		
	類型：	INT	
	值：	0	有效平面中含參數 2 的交點計算點（標準值）
		1	交點計算點獨立於傳輪平面

說明

請注意，變數需在使用前先定義。

傳輪輪廓時，需觀察以 CONTPRON 定義之值：

參數	意義
2	移動輪廓模式的編碼
3	輪廓起點橫座標
4	輪廓起點縱座標
5	輪廓終點橫座標
6	輪廓終點縱座標
9	橫座標的中心點座標（僅用於圓弧輪廓）
10	縱座標的中心點座標（僅用於圓弧輪廓）

範例

計算表 TABNAME1 中輪廓元素 3，與表 TABNAME2 中輪廓元素 7 的交點。有效平面上的交點座標會儲存於變數 ISCOORD（第一元素 = 橫座標，第二元素 = 縱座標）。若無交點存在，則程式跳躍至 NOCUT（未找到交點）。

程式碼	註解
DEF REAL TABNAME1[12，11]	; 輪廓表 1
DEF REAL TABNAME2[10，11]	; 輪廓表 2
DEF REAL ISCOORD [2]	; 交點座標點的變數。
DEF BOOL ISPOINT	; 交點狀態的變數。
DEF INT MODE	; 加工類型的變數。
…	
MODE=1	; 獨立於有效平面的計算。
N10 ISPOINT=INTERSEC（TABNAME1[3]，TABNAME2[7]，ISCOORD，MODE）	; 呼叫輪廓元素的交點。
N20 IF ISPOINT==FALSE GOTOF NOCUT	; 跳躍至 NOCUT
…	

15.5 按單節執行表的輪廓元素（EXECTAB）

功能

使用 EXECTAB 指令，可按單節執行表的輪廓元素－使用 CONTPRON 指令產生的元素。

句法

EXECTAB（<輪廓表>[<輪廓元素>]）

意義

EXECTAB	執行輪廓元素的指令
<輪廓表>	輪廓表名稱
<輪廓元素>	輪廓元素的數量

範例

需按單節執行表 KTAB 中的輪廓元素 0 至 2。

程式碼	註解
N10 EXECTAB（KTAB[0]）	; 移動表 KTAB 的元素 0。
N20 EXECTAB（KTAB[1]）	; 移動表 KTAB 的元素 1。
N30 EXECTAB（KTAB[2]）	; 移動表 KTAB 的元素 2。

15.6 計算圓弧資料（CALCDAT）

功能

使用 CALCDAT 指令，可從圓上已知的三或四個點，計算出半徑及中心點座標。需為不同指定點。四個點並未直接位於圓上，由中心點及半徑算出平均值。

句法

```
<Status>=CALCDAT(<圓點>[<編號>,<類型>],<編號>,<結果>)
```

意義

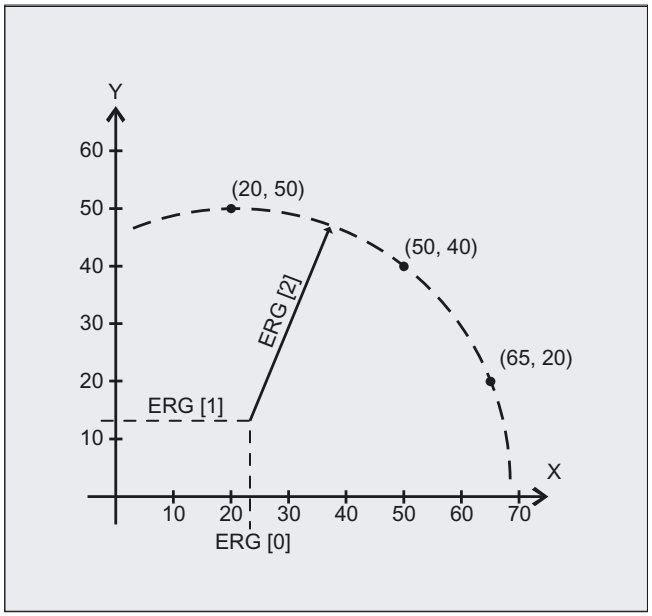
CALCDAT	由 3 或 4 個點計算圓半徑及中心點座標的指令。
<狀態>	圓弧計算狀態的變數 類型： BOOL 值： TRUE 指定點位於圓上。 FALSE 指定點**並未**位於圓上。
<圓點>[ ]	使用參數指定圓弧點的變數 <編號> 圓弧點的數量（3 或 4） <類型> 座標資料類型， 例如，2 表示 2 點座標
<編號>	用於計算（3 或 4）的點數量之參數
<結果>[3]	結果的變數： 中心點座標及半徑 0 中心點座標：橫座標值 1 中心點座標：縱座標值 2 半徑

說明

請注意，變數需在使用前先定義。

範例

應使用三個點，以決定三個點是否位在一圓弧區段上。



程式碼	註解
N10 DEF REAL PT[3, 2]=(20, 50, 50, 40, 65, 20)	; 指定圓弧上點之變數
N20 DEF REAL RES[3]	; 結果的變數
N30 DEF BOOL STATUS	; 狀態的變數
N40 STATUS=CALCDAT(PKT, 3, ERG)	; 呼叫已決定的圓弧資料。
N50 IF STATUS == FALSE GOTOF ERROR	; 跳躍至錯誤

15.7 停用輪廓準備（EXECUTE）

功能

EXECUTE 指令用於停用輪廓準備，同時系統會回到正常執行模式。

句法

EXECUTE（<ERROR>）

意義

EXECUTE	終止輪廓準備的指令
<FAULT>	錯誤反饋訊號的變數 類型： INT 變數值指出輪廓是否能以無錯誤做好準備： 0 錯誤 1 無錯誤

範例

程式碼

```
...
N30 CONTPRON（...）
N40 G1 X... Z...
...
N100 EXECUTE（...）
...
```

16 表格

16.1 敘述清單

圖例：

1) 參考包含有詳細操作說明的文件：
- *PG* 程式設計手動，基礎
- *PGA* 程式設計手冊，工作計畫
- *BHD* 操作手冊，HMI sl 車削
- *BHF* 操作手冊，HMI sl 銑削
- *FB1（）* 功能手冊，基本功能（具有相對應函數的字母與數字之縮寫，函數以括弧說明）
- *FB2（）* 功能手冊，延伸功能（具有相對應函數的字母與數字縮寫，函數以括弧說明）
- *FB3（）* 功能手冊，特殊函數（具有相對應函數的字母與數字縮寫，函數以括弧說明）
- *FBSI* 功能手冊，安全整合
- *FBSY* 功能手冊，同步動作
- *FBW* 功能手冊，刀具管理

2) 操作的效果：
- m 模態
- n 非模態

3) SINUMERIK 828D 的可用性（D = 車削，F = 銑削）：
- ● 標準值
- ○ 選項
- - 無法使用

4) 程式開始之預設設定（如程式中未程式設計，為該控制項目之出廠設定）。

操作	含義	說明，請參考 1)	W 2)	828D 3) PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
:	NC 主單節編號，跳躍標籤中止，串鍊運算子	*PGA* 運算功能 (頁 58)		●	●	●	●
*	乘法之運算子	*PGA* 運算功能 (頁 58)		●	●	●	●
+	加法之運算子	*PGA* 運算功能 (頁 58)		●	●	●	●
-	減法之運算子	*PGA* 運算功能 (頁 58)		●	●	●	●
<	比較運算子，少於	*PGA* 運算功能 (頁 58)		●	●	●	●
<<	字串的串鍊運算子	*PGA* 運算功能 (頁 58)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - D - - - - - F	PPU280 / 281 D - - - - - F
<=	比較運算子，少於或等於	*PGA* 運算功能 (頁 58)		●	●	●	●
=	指派運算子	*PGA* 運算功能 (頁 58)		●	●	●	●
>=	比較運算子，大於或等於	*PGA* 運算功能 (頁 58)		●	●	●	●
/	除法之運算子	*PGA* 運算功能 (頁 58)		●	●	●	●
/0 … … /7	單節已略過（第一略過層級） 單節已略過（第八略過層級）	*PG*		● ○	● ○	● ○	● ○
A	軸名稱	*PGA* 程式設計刀具方向（A...、B...、C...、LEAD、TILT） (頁 286)	m/n	●	●	●	●
A2	刀具方向： RPY 或尤拉角	*PGA* 程式設計刀具方向（A...、B...、C...、LEAD、TILT） (頁 286)	n	●	●	●	●
A3	刀具方向： 方向/表面法線向量元件	*PGA* 程式設計刀具方向（A...、B...、C...、LEAD、TILT） (頁 286)	n	●	●	●	●
A4	刀具方向： Surface normal vector for beginning of block	*PGA* 平面銑削（3D銑削A4、B4、C4、A5、B5、C5） (頁 293)	n	●	●	●	●
A5	刀具方向： 單節結尾的表面法線向量	*PGA* 平面銑削（3D銑削A4、B4、C4、A5、B5、C5） (頁 293)	秒	●	●	●	●
ABS	絕對值（量）	*PGA* 運算功能 (頁 58)		●	●	●	●
AC	座標/位置的絕對位置	*PG*	n	●	●	●	●
ACC	目前的軸加速度之效果	*PG*	m	●	●	●	●
ACCLIMA	目前的最大軸加速度之效果	*PG*	m	●	●	●	●
ACN	旋轉軸之絕對尺寸，負向上的逼近位置	*PG*	n	●	●	●	●
ACOS	反餘弦（三角函數）	*PGA* 運算功能 (頁 58)		●	●	●	●
ACP	旋轉軸之絕對尺寸，正向上的逼近位置	*PG*	n	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
ACTBLOCNO	輸出目前的警報單節的單節號碼，就算"目前單節顯示抑制"（DISPLOF）為生效。	*PGA* 抑制目前的單節顯示（DISPLOF、DISPLON、ACTBLOCNO） (頁 148)		●	●	●	●
ADDFRAME	包含可能的量測框架之啟用	*PGA*、*FB1（K2）* 從空間中的三個量測點計算框架（MEAFRAME） (頁 266)		●	●	●	●
ADIS	路徑函數 G1、G2、G3、...之倒圓角間隙	*PG*	m	●	●	●	●
ADISPOS	快速移動 G0 之倒圓角間隙	*PG*	m	●	●	●	●
ADISPOSA	IPOBRKA 允差視窗之大小	*PGA* 已程式設計的動作結尾條件（FINEA、COARSEA、IPOENDA、IPOBRKA、ADISPOSA） (頁 245)	m	●	●	●	●
ALF	LIFTFAST 角度	*PGA* 從輪廓快速回退（SETINT LIFTFAST, ALF） (頁 105)	m	●	●	●	●
AMIRROR	可程式設計的鏡像	*PG*	n	●	●	●	●
AND	邏輯 AND	*PGA* 比較與邏輯運算 (頁 60)		●	●	●	●
ANG	輪廓角度	*PG*	n	●	●	●	●
AP	極角	*PG*	m/n	●	●	●	●
APR	讀取/顯示存取保護	*PGA* 屬性： 存取權力（APR、APW、APRP、APWP、APRB、APWB） (頁 36)		●	●	●	●
APRB	讀取存取權限，OPI	*PGA* 屬性： 存取權力（APR、APW、APRP、APWP、APRB、APWB） (頁 36)		●	●	●	●
APRP	讀取存取權限，工件程式	*PGA* 屬性： 存取權力（APR、APW、APRP、APWP、APRB、APWB） (頁 36)		●	●	●	●
APW	寫入存取保護	*PGA* 屬性： 存取權力（APR、APW、APRP、APWP、APRB、APWB） (頁 36)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
APWB	寫入存取權限，OPI	*PGA* 屬性： 存取權力（APR、APW、APRP、APWP、APRB、APWB） (頁 36)		●	●	●	●
APWP	寫入存取權限，工件程式	*PGA* 屬性： 存取權力（APR、APW、APRP、APWP、APRB、APWB） (頁 36)		●	●	●	●
APX	用來執行特定語言元件的存取權限之定義	*PGA* 屬性： 存取權力（APR、APW、APRP、APWP、APRB、APWB） (頁 36)		●	●	●	●
AR	缺口角度	*PG*	m/n	●	●	●	●
AROT	可程式設計的旋轉	*PG*	n	●	●	●	●
AROTS	具立體角的可程式設計框架旋轉	*PG*	n	●	●	●	●
SL	巨集定義	*PGA* 巨集技術（DEFINE ... AS） (頁 180)		●	●	●	●
ASCALE	可程式設計的比例	*PG*	n	●	●	●	●
ASIN	算術函數，反正弦	*PGA* 運算功能 (頁 58)		●	●	●	●
ASPLINE	Akima 曲線	*PGA* 曲線插補（ASPLINE、BSPLINE、CSPLINE、BAUTO、BNAT、BTAN、EAUTO、ENAT、ETAN、PW、SD、PL） (頁 208)	m	-	○	-	○
ATAN2	反正切 2	*PGA* 運算功能 (頁 58)		●	●	●	●
ATOL	壓縮函數、方向平滑化、與平滑化類型的軸專屬公差	*PGA* 可程式設計輪廓/方向允差（CTOL、OTOL、ATOL） (頁 434)		-	●	-	●
ATRANS	附加的可程式設計轉譯	*PG*	n	●	●	●	●
AX	變數軸識別碼	*PGA* 軸函數（AXNAME、AX、SPI、AXTOSPI、ISAXIS、AXSTRING、MODAXVAL） (頁 589)	m/n	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
AXCSWAP	前容器軸	*PGA* 軸容器（AXCTSWE、AXCTSWED） (頁 595)		-	-	-	-
AXCTSWE	前容器軸	*PGA* 軸容器（AXCTSWE、AXCTSWED） (頁 595)		-	-	-	-
AXCTSWED	旋轉軸容器	*PGA* 軸容器（AXCTSWE、AXCTSWED） (頁 595)		-	-	-	-
AXIS	軸識別碼，軸位址	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
AXNAME	將輸入字串轉換為軸識別碼	*PGA* 軸函數（AXNAME、AX、SPI、AXTOSPI、ISAXIS、AXSTRING、MODAXVAL） (頁 589)		●	●	●	●
AXSTRING	轉換字串主軸號碼	*PGA* 軸函數（AXNAME、AX、SPI、AXTOSPI、ISAXIS、AXSTRING、MODAXVAL） (頁 589)		●	●	●	●
AXTOCHAN	請求特定通道之主軸。 可能來自 NC 程式及同步化動作。	*PGA* 將座標軸傳輸至其他通道（AXTOCHAN）： (頁 114)		●	●	●	●
AXTOSPI	將軸識別碼轉換為主軸索引	*PGA* 軸函數（AXNAME、AX、SPI、AXTOSPI、ISAXIS、AXSTRING、MODAXVAL） (頁 589)		●	●	●	●
B	軸名稱	*PGA* 程式設計刀具方向（A...、B...、C...、LEAD、TILT） (頁 286)	m/n	●	●	●	●
B2	刀具方向：RPY 或尤拉角	*PGA* 程式設計刀具方向（A...、B...、C...、LEAD、TILT） (頁 286)	n	●	●	●	●
B3	刀具方向：方向/表面法線向量元件	*PGA* 程式設計刀具方向（A...、B...、C...、LEAD、TILT） (頁 286)	n	●	●	●	●
B4	刀具方向：Surface normal vector for beginning of block	*PGA* 平面銑削（3D銑削A4、B4、C4、A5、B5、C5） (頁 293)	n	●	●	●	●
B5	刀具方向：單節結尾的表面法線向量	*PGA* 平面銑削（3D銑削A4、B4、C4、A5、B5、C5） (頁 293)	n	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
B_AND	位元 AND	*PGA* 比較與邏輯運算 (頁 60)		●	●	●	●
B_OR	位元 OR	*PGA* 比較與邏輯運算 (頁 60)		●	●	●	●
B_NOT	位元否定	*PGA* 比較與邏輯運算 (頁 60)		●	●	●	●
B_XOR	位元互斥式 OR	*PGA* 比較與邏輯運算 (頁 60)		●	●	●	●
BAUTO	以接下來 3 個點定義最初的曲線區段	*PGA* 曲線插補（ASPLINE、BSPLINE、CSPLINE、BAUTO、BNAT、BTAN、EAUTO、ENAT、ETAN、PW、SD、PL） (頁 208)	m	-	○	-	○
BLOCK	和關鍵字 TO 一起定義，在間接子程式呼叫中待處理的程式片段	*PGA* 使用呼叫程式部份的規格間接進行副程式呼叫（CALL BLOCK ... TO ...） (頁 169)		●	●	●	●
BLSYNC	僅隨下一個單節變更開始處理中斷程式	*PGA* 指派並啟動中斷程式（SETINT，PRIO，BLSYNC） (頁 102)		●	●	●	●
BNAT [4]	自然變化至第一個曲線單節	*PGA* 曲線插補（ASPLINE、BSPLINE、CSPLINE、BAUTO、BNAT、BTAN、EAUTO、ENAT、ETAN、PW、SD、PL） (頁 208)	m	-	○	-	○
BOOL	資料類型： 布林值 TRUE/FALSE 或 1/0	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
BOUND	測試數值是否介於定義之值域間。若數值相同，將傳回測試值	*PGA* 變數最小，最大與範圍（MINVAL，MAXVAL與BOUND） (頁 64)		●	●	●	●
BRISK [4]	路徑非平滑化之快速加速	*PG*	m	●	●	●	●
BRISKA	啟動已程式設計之軸的路徑輕快加速	*PG*		●	●	●	●
BSPLINE	B-spline	*PGA* 曲線插補（ASPLINE、BSPLINE、CSPLINE、BAUTO、BNAT、BTAN、EAUTO、ENAT、ETAN、PW、SD、PL） (頁 208)	m	-	○	-	○

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
BTAN	沿切線變化至第一個曲線單節	*PGA* 曲線插補（ASPLINE、BSPLINE、CSPLINE、BAUTO、BNAT、BTAN、EAUTO、ENAT、ETAN、PW、SD、PL） (頁 208)	m	-	○	-	○
C	軸名稱	*PGA* 程式設計刀具方向（A...、B...、C...、LEAD、TILT） (頁 286)	m/n	●	●	●	●
C2	刀具方向：RPY 或尤拉角	*PGA* 程式設計刀具方向（A...、B...、C...、LEAD、TILT） (頁 286)	n	●	●	●	●
C3	刀具方向：方向/表面法線向量元件	*PGA* 程式設計刀具方向（A...、B...、C...、LEAD、TILT） (頁 286)	n	●	●	●	●
C4	刀具方向：Surface normal vector for beginning of block	*PGA* 平面銑削（3D銑削A4、B4、C4、A5、B5、C5） (頁 293)	n	●	●	●	●
C5	刀具方向：單節結尾的表面法線向量	*PGA* 平面銑削（3D銑削A4、B4、C4、A5、B5、C5） (頁 293)	n	●	●	●	●
CAC	絕對位置逼近	*PGA* 逼近編碼位置（CAC、CIC、CDC、CACP、CACN） (頁 207)		●	●	●	●
CACN	列於表中的值，以反方向絕對逼近	*PGA* 逼近編碼位置（CAC、CIC、CDC、CACP、CACN） (頁 207)		●	●	●	●
CACP	列於表中的值，以同方向絕對逼近	*PGA* 逼近編碼位置（CAC、CIC、CDC、CACP、CACN） (頁 207)		●	●	●	●
CALCDAT	從 3 或 4 個點計算半徑和圓的中心點	*PGA* 計算圓弧資料（CALCDAT） (頁 629)		●	●	●	●
CALCPOSI	檢查保護區差異、工作區限制與軟體限制	*PGA* 檢查保護區差異、工作區限制與軟體限制（CALCPOSI） (頁 200)		●	●	●	●
CALL（呼叫）	間接的子程式呼叫	*PGA* 間接子程式呼叫（CALL） (頁 168)		●	●	●	●
CALLPATH	子程式呼叫之可程式設計的搜尋路徑	*PGA* 供副程式呼叫（CALLPATH）所用的延伸搜尋路徑 (頁 172)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
CANCEL	取消模態同步化動作	*PGA* 刪除同步動作（CANCEL） (頁 559)		●	●	●	●
CASE	條件式程式分支	*PGA* 程式分支（CASE ... OF ... DEFAULT ...） (頁 81)		●	●	●	●
CDC	直接逼近位置	*PGA* 逼近編碼位置（CAC、CIC、CDC、CACP、CACN） (頁 207)		●	●	●	●
CDOF [4]	碰撞偵測 OFF	*PG*	m	●	●	●	●
CDOF2	碰撞偵測 OFF，對於 3D 周長銑削	*PG*	m	●	●	●	●
CDON	碰撞偵測 ON	*PG*	m	●	●	●	●
CFC [4]	輪廓上的常數進給率	*PG*	m	●	●	●	●
CFIN	恆定進給率僅適用內部半徑，不適用外部半徑	*PG*	m	●	●	●	●
CFINE	指派微調偏移量給 FRAME 變數	*PGA* 粗調或微調偏移量（CFINE、CTRANS） (頁 262)		●	●	●	●
CFTCP	刀具中心點之恆定進給率（圓心路徑）	*PG*	m	●	●	●	●
CHAN	指定資料的有效範圍	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
CHANDATA	設定通道編號以存取通道資料	*PGA* 工作記憶體（CHANDATA，COMPLETE，INITIAL） (頁 188)		●	●	●	●
CHAR	資料類型： ASCII 字元	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
CHECKSUM	將陣列的總合檢查碼變成固定長度的 STRING	*PGA* 使用陣列進行總合檢查碼計算（CHECKSUM） (頁 130)		●	●	●	●
CHF	倒角； 值 = 倒角長度	*PG*	n	●	●	●	●
CHKDM	刀庫中的唯一性檢查	*FBW*		●	●	●	●
CHKDNO	檢查特有的 D 編號	*PGA* 任意指派D數量檢查D數量（CHKDNO） (頁 381)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
CHR	倒角； 值=移動方向的倒角長度	*PG*		●	●	●	●
CIC	藉由增量的逼近位置	*PGA* 逼近編碼位置（CAC、CIC、CDC、CACP、CACN） (頁 207)		●	●	●	●
CIP	經由中間點的圓弧插補	*PG*	m	●	●	●	●
CLEARM	為通道座標重置一個 / 數個標記	*PGA* 程式一致性（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM） (頁 95)		-	-	-	-
CLRINT	取消選取中斷：	*PGA* 刪除指派的中斷程式（CLRINT） (頁 104)		●	●	●	●
CMIRROR	座標軸上之鏡射	*PGA* 運算功能 (頁 58)		●	●	●	●
COARSEA	於達到“精確停止粗調”時動作結束	*PGA* 已程式設計的動作結尾條件（FINEA、COARSEA、IPOENDA、IPOBRKA、ADISPOSA） (頁 245)	m	●	●	●	●
COMPCAD	壓縮機能 ON： CAD 程式的合理表面品質	*PGA* NC單節壓縮（COMPON、COMPCURV、COMPCAD、COMPOF） (頁 221)	m	-	○	-	○
COMPCURV	壓縮機能 ON： 含恆定曲率之多項式	*PGA* NC單節壓縮（COMPON、COMPCURV、COMPCAD、COMPOF） (頁 221)	m	-	○	-	○
COMPLETE	用來讀取和寫入資料的控制指令	PGA 工作記憶體（CHANDATA，COMPLETE，INITIAL） (頁 188)		●	●	●	●
COMPOF [4]	壓縮程序 OFF（關閉）	*PGA* NC單節壓縮（COMPON、COMPCURV、COMPCAD、COMPOF） (頁 221)	m	-	○	-	○
COMPON	壓縮機能 ON	*PGA* NC單節壓縮（COMPON、COMPCURV、COMPCAD、COMPOF） (頁 221)		-	○	-	○
CONTDCON	表格化輪廓解碼 ON	*PGA* 產生編碼輪廓表（CONTDCON） (頁 622)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
CONTPRON	啟動參考前置處理	*PGA* 產生輪廓表（CONTPRON） (頁 616)		●	●	●	●
CORROF	取消選擇所有啟用的重疊動作	*PG*		●	●	●	●
COS	餘弦 （三角函數）	*PGA* 運算功能 (頁 58)		●	●	●	●
COUPDEF	定義 ELG 群組/同步主軸群組	*PGA* 同步主軸：程式設計（COUPDEF、COUPDEL、COUPON、COUPONC、COUPOF、COUPOFS、COUPRES、WAITC） (頁 472)		○	-	○	-
COUPDEL	刪除 ELG 群組	*PGA* 同步主軸：程式設計（COUPDEF、COUPDEL、COUPON、COUPONC、COUPOF、COUPOFS、COUPRES、WAITC） (頁 472)		○	-	○	-
COUPOF	ELG 群組/同步主軸組 ON	*PGA* 同步主軸：程式設計（COUPDEF、COUPDEL、COUPON、COUPONC、COUPOF、COUPOFS、COUPRES、WAITC） (頁 472)		○	-	○	-
COUPOFS	停用 ELG 群組 / 同步化主軸對，搭配跟隨主軸之停止	*PGA* 同步主軸：程式設計（COUPDEF、COUPDEL、COUPON、COUPONC、COUPOF、COUPOFS、COUPRES、WAITC） (頁 472)		○	-	○	-
COUPON	ELG 群組/同步主軸組 ON	*PGA* 同步主軸：程式設計（COUPDEF、COUPDEL、COUPON、COUPONC、COUPOF、COUPOFS、COUPRES、WAITC） (頁 472)		○	-	○	-
COUPONC	傳輸 ELG 群組 / 同步化主軸對，含先前之程式設計	*PGA* 同步主軸：程式設計（COUPDEF、COUPDEL、COUPON、COUPONC、COUPOF、COUPOFS、COUPRES、WAITC） (頁 472)		○	-	○	-
COUPRES	重置 ELG 群組	*PGA* 同步主軸：程式設計（COUPDEF、COUPDEL、COUPON、COUPONC、COUPOF、COUPOFS、COUPRES、WAITC） (頁 472)		○	-	○	-

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - D - - - - -	828D [3] PPU260 / 261 - - - F - - - - -	828D [3] PPU280 / 281 D - - - - -	828D [3] PPU280 / 281 F - - - - -
CP	路徑動作	*PGA* 直角座標PTP移動 (頁 333)	m	●	●	●	●
CPRECOF [4]	可程式設計的輪廓精密度 OFF	*PG*	m	●	●	●	●
CPRECON	可程式設計的輪廓精密度 ON	*PG*	m	●	●	●	●
CPROT	通道專屬之保護區 ON/OFF	*PGA* 啟動 / 停用保護區（CPROT、NPROT） (頁 196)		●	●	●	●
CPROTDEF	定義通道專屬保護區	*PGA* 保護區（CPROTDEF、NPROTDEF）定義 (頁 193)		●	●	●	●
CR	圓形半徑	*PG*	n	●	●	●	●
CROT	目前座標系統之旋轉。	*PGA* 運算功能 (頁 58)		●	●	●	●
CROTS	可程式設計的框架以固定角度旋轉（在指定軸中之旋轉）	*PG*	n	●	●	●	●
CRPL	任何平面中的框架旋轉	*FB1（K2）*		●	●	●	●
CSCALE	數個軸的比例係數	*PGA* 運算功能 (頁 58)		●	●	●	●
CSPLINE	三次曲線	*PGA* 曲線插補（ASPLINE、BSPLINE、CSPLINE、BAUTO、BNAT、BTAN、EAUTO、ENAT、ETAN、PW、SD、PL） (頁 208)	m	-	○	-	○
CT	含切線的圓弧	*PG*	m	●	●	●	●
CTAB	依據由曲線表取得之先導軸位置定義跟隨軸位置	*PGA* 讀取曲線表值（CTABTSV、CTABTEV、CTABTSP、CTABTEP、CTABSSV、CTABSEV、CTAB、CTABINV、CTABTMIN、CTABTMAX） (頁 452)		-	-	-	-
CTABDEF	表定義 ON	*PGA* 定義曲線表（CTABDEF、CATBEND） (頁 442)		-	-	-	-
CTABDEL	清除曲線表	*PGA* 刪除曲線表（CTABDEL） (頁 448)		-	-	-	-
CTABEND	表定義 OFF	*PGA* 定義曲線表（CTABDEF、CATBEND） (頁 442)		-	-	-	-

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
CTABEXISTS	以數字 n 檢查曲線表	*PGA* 確認曲線表是否存在（CTABEXISTS） (頁 448)		-	-	-	-
CTABFNO	記憶體中仍可使用的曲線表數量	*PGA* 曲線表：檢查資源的使用（CTABNO，CTABNOMEM，CTABFNO，CTABSEGID，CTABSEG，CTABFSEG，CTABMSEG，CTABPOLID，CTABPOL，CTABFPOL，CTABMPOL） (頁 457)		-	-	-	-
CTABFPOL	記憶體中仍可使用的多項式數量	*PGA* 曲線表：檢查資源的使用（CTABNO，CTABNOMEM，CTABFNO，CTABSEGID，CTABSEG，CTABFSEG，CTABMSEG，CTABPOLID，CTABPOL，CTABFPOL，CTABMPOL） (頁 457)		-	-	-	-
CTABFSEG	記憶體中仍可使用的曲線區段數量	*PGA* 曲線表：檢查資源的使用（CTABNO，CTABNOMEM，CTABFNO，CTABSEGID，CTABSEG，CTABFSEG，CTABMSEG，CTABPOLID，CTABPOL，CTABFPOL，CTABMPOL） (頁 457)		-	-	-	-
CTABID	傳回第 n 個曲線表的表編號	*PGA* 曲線表：決定取線表特性（CTABID、CTABISLOCK、CTABMEMTYP、CTABPERIOD） (頁 451)		-	-	-	-
CTABINV	依據由曲線表取得之跟隨軸位置定義先導軸位置	*PGA* 讀取曲線表值（CTABTSV、CTABTEV、CTABTSP、CTABTEP、CTABSSV、CTABSEV、CTAB、CTABINV、CTABTMIN、CTABTMAX） (頁 452)		-	-	-	-
CTABISLOCK	利用數字 n 傳回曲線表之所定狀態	*PGA* 曲線表：決定取線表特性（CTABID、CTABISLOCK、CTABMEMTYP、CTABPERIOD） (頁 451)		-	-	-	-
CTABLOCK	刪除與覆寫，鎖定	*PGA* 鎖住曲線表以防止刪除與覆寫（CTABLOCK、CTABUNLOCK） (頁 450)		-	-	-	-

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
CTABMEMTYP	傳回建立編號 n 曲線表的記憶體	*PGA* 曲線表：決定取線表特性（CTABID、CTABISLOCK、CTABMEMTYP、CTABPERIOD） (頁 451)		-	-	-	-
CTABMPOL	可在記憶體中取得多項式的最大編號	*PGA* 曲線表：檢查資源的使用（CTABNO，CTABNOMEM，CTABFNO，CTABSEGID，CTABSEG，CTABFSEG，CTABMSEG，CTABPOLID，CTABPOL，CTABFPOL，CTABMPOL） (頁 457)		-	-	-	-
CTABMSEG	可在記憶體中取得曲線區段的最大編號	*PGA* 曲線表：檢查資源的使用（CTABNO，CTABNOMEM，CTABFNO，CTABSEGID，CTABSEG，CTABFSEG，CTABMSEG，CTABPOLID，CTABPOL，CTABFPOL，CTABMPOL） (頁 457)		-	-	-	-
CTABNO	在 SRAM 或 DRAM 中已定義之曲線表數量	*FB3（M3）*		-	-	-	-
CTABNOMEM	在 SRAM 或 DRAM 中已定義之曲線表數量	*PGA* 曲線表：檢查資源的使用（CTABNO，CTABNOMEM，CTABFNO，CTABSEGID，CTABSEG，CTABFSEG，CTABMSEG，CTABPOLID，CTABPOL，CTABFPOL，CTABMPOL） (頁 457)		-	-	-	-
CTABPERIOD	傳回曲線表編號 n 的表週期性：	*PGA* 曲線表：決定取線表特性（CTABID、CTABISLOCK、CTABMEMTYP、CTABPERIOD） (頁 451)		-	-	-	-
CTABPOL	記憶體中已使用的多項式數量	*PGA* 曲線表：檢查資源的使用（CTABNO，CTABNOMEM，CTABFNO，CTABSEGID，CTABSEG，CTABFSEG，CTABMSEG，CTABPOLID，CTABPOL，CTABFPOL，CTABMPOL） (頁 457)		-	-	-	-

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
CTABPOLID	編號為 n 之曲線表使用之曲線多項式數量	*PGA* 曲線表：檢查資源的使用（CTABNO，CTABNOMEM，CTABFNO，CTABSEGID，CTABSEG，CTABFSEG，CTABMSEG，CTABPOLID，CTABPOL，CTABFPOL，CTABMPOL） (頁 457)		-	-	-	-
CTABSEG	記憶體中已使用的曲線區段數量	*PGA* 曲線表：檢查資源的使用（CTABNO，CTABNOMEM，CTABFNO，CTABSEGID，CTABSEG，CTABFSEG，CTABMSEG，CTABPOLID，CTABPOL，CTABFPOL，CTABMPOL） (頁 457)		-	-	-	-
CTABSEGID	編號為 n 之曲線表使用之曲線區段數量	*PGA* 曲線表：檢查資源的使用（CTABNO，CTABNOMEM，CTABFNO，CTABSEGID，CTABSEG，CTABFSEG，CTABMSEG，CTABPOLID，CTABPOL，CTABFPOL，CTABMPOL） (頁 457)		-	-	-	-
CTABSEV	傳回曲線表中某區段之跟隨軸的最終值	*PGA* 讀取曲線表值（CTABTSV、CTABTEV、CTABTSP、CTABTEP、CTABSSV、CTABSEV、CTAB、CTABINV、CTABTMIN、CTABTMAX） (頁 452)		-	-	-	-
CTABSSV	傳回曲線表中某區段之跟隨軸的初始值	*PGA* 讀取曲線表值（CTABTSV、CTABTEV、CTABTSP、CTABTEP、CTABSSV、CTABSEV、CTAB、CTABINV、CTABTMIN、CTABTMAX） (頁 452)		-	-	-	-
CTABTEP	傳回曲線表結尾處之先導軸的值	*PGA* 讀取曲線表值（CTABTSV、CTABTEV、CTABTSP、CTABTEP、CTABSSV、CTABSEV、CTAB、CTABINV、CTABTMIN、CTABTMAX） (頁 452)		-	-	-	-

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
CTABTEV	傳回曲線表結尾處之跟隨軸的值	*PGA* 讀取曲線表值（CTABTSV、CTABTEV、CTABTSP、CTABTEP、CTABSSV、CTABSEV、CTAB、CTABINV、CTABTMIN、CTABTMAX） (頁 452)		-	-	-	-
CTABTMAX	傳回曲線表中跟隨軸的最大值	*PGA* 讀取曲線表值（CTABTSV、CTABTEV、CTABTSP、CTABTEP、CTABSSV、CTABSEV、CTAB、CTABINV、CTABTMIN、CTABTMAX） (頁 452)		-	-	-	-
CTABTMIN	傳回曲線表中跟隨軸的最小值	*PGA* 讀取曲線表值（CTABTSV、CTABTEV、CTABTSP、CTABTEP、CTABSSV、CTABSEV、CTAB、CTABINV、CTABTMIN、CTABTMAX） (頁 452)		-	-	-	-
CTABTSP	傳回曲線表起始處之先導軸的值	*PGA* 讀取曲線表值（CTABTSV、CTABTEV、CTABTSP、CTABTEP、CTABSSV、CTABSEV、CTAB、CTABINV、CTABTMIN、CTABTMAX） (頁 452)		-	-	-	-
CTABTSV	傳回曲線表起始處之跟隨軸的值	*PGA* 讀取曲線表值（CTABTSV、CTABTEV、CTABTSP、CTABTEP、CTABSSV、CTABSEV、CTAB、CTABINV、CTABTMIN、CTABTMAX） (頁 452)		-	-	-	-
CTABUNLOCK	撤回刪除與覆寫，鎖定	*PGA* 鎖住曲線表以防止刪除與覆寫（CTABLOCK、CTABUNLOCK） (頁 450)		-	-	-	-
CTOL	壓縮函數、方向平滑化、與平滑化類型的輪廓允差	*PGA* 可程式設計輪廓/方向允差（CTOL、OTOL、ATOL） (頁 434)		-	○	-	○
CTRANS	數個軸的零點偏移	*PGA* 粗調或微調偏移量（CFINE、CTRANS） (頁 262)		●	●	●	●
CUT2D [4]	2D 刀具偏移	*PG*	m	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
CUT2DF	2D 刀具偏移量，刀具偏移量配合目前的框架使用（斜面）。	*PG*	m	●	●	●	●
CUT3DC	3D 刀具偏移量周長銑削	*PGA* 啟動 3D刀具補正(CUT3DC…, CUT3DF…) (頁 363)	m	-	-	-	-
CUT3DCC	3D 刀具偏移周長銑削，搭配限制表面	*PGA* 3D刀具偏移將限制表面納入考量（CUT3DCC、CUT3DCCD） (頁 371)	m	-	-	-	-
CUT3DCCD	3D 刀具偏移周長銑削，搭配含差別刀具之限制表面	*PGA* 3D刀具偏移將限制表面納入考量（CUT3DCC、CUT3DCCD） (頁 371)	m	-	-	-	-
CUT3DF	3D 刀具偏移量平面銑削	*PGA* 啟動 3D刀具補正(CUT3DC…, CUT3DF…) (頁 363)	m	-	-	-	-
CUT3DFF	3D 刀具偏移銑削，使用相依於啟用中框架之恆定刀具方向	*PGA* 啟動 3D刀具補正(CUT3DC…, CUT3DF…) (頁 363)	m	-	-	-	-
CUT3DFS	3D 刀具偏移銑削，使用獨立於啟用中框架之恆定刀具方向	*PGA* 啟動 3D刀具補正(CUT3DC…, CUT3DF…) (頁 363)	m	-	-	-	-
CUTCONOF [4]	刀具半徑補正 OFF（關閉）	*PG*	m	●	●	●	●
CUTCONON	刀具半徑補正 ON（開啟）	*PG*	m	●	●	●	●
CUTMOD	啟動“可旋轉刀具偏移資料之修改”	*PGA* 可旋轉刀具的切削資料修改（CUTMOD） (頁 395)		●	●	●	●
CYCLE...	量測循環	*BHD/BHF*					
D	刀具偏移編號	*PG*		●	●	●	●
D0	使用 D0 時，刀具的偏移量會生效。	*PG*		●	●	●	●
DAC	絕對非模態、軸專屬直徑程式設計	*PG*	n	●	●	●	●
DC	旋轉軸之絕對直徑，直接逼近位置	*PG*	n	●	●	●	●
DEF	變數定義	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
DEFINE	用於巨集定義的關鍵字	*PGA* 巨集技術（DEFINE ... AS） (頁 180)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
DEFAULT	CASE 分支內之分支	*PGA* 程式分支（CASE ... OF ... DEFAULT ...） (頁 81)		●	●	●	●
DELAYFSTON	定義停止延遲區段的開始	*PGA* 程式分支（CASE ... OF ... DEFAULT ...） (頁 81)	m	●	●	●	●
DELAYFSTOF	定義停止延遲區段的結束	*PGA* 條件式可中斷程式區段（DELAYFSTON, DELAYFSTOF） (頁 417)	m	●	●	●	●
DELDL	刪除附加的偏移量	*PGA* 刪除附加偏移（DELDL） (頁 350)		●	●	●	●
DELDTG	剩餘距離的刪除	*PGA* 刪除剩餘距離（DELDTG） (頁 514)		●	●	●	●
DELETE	刪除指定的檔案。可以用路徑和檔案識別碼來指定檔名。	*PGA* 刪除檔案（DELETE） (頁 120)		●	●	●	●
DELTOOLENV	刪除說明刀具環境的資料記錄	*FB1（W1）*		●	●	●	●
DIACYCOFA	軸專屬模態半徑程式設計：於循環中 OFF	*FB1（P1）*	m	●	●	●	●
DIAM90	適用於 G90 之直徑程式設計，適用於 G91 之半徑程式設計	*PGA*	m	●	●	●	●
DIAM90A	適用於 G90 及 AC 的軸專屬模態直徑程式設計，適用於 G91 及 IC 的半徑程式設計	*PG*	m	●	●	●	●
DIAMCHAN	MD 軸函數中所有處於直徑程式設計通道狀態之軸的傳輸	*PG*		●	●	●	●
DIAMCHANA	直徑程式設計通道狀態的傳輸	*PG*		●	●	●	●
DIAMCYCOF	通道專屬直徑程式設計：於循環中 OFF	*FB1（P1）*	m	●	●	●	●
DIAMOF [4]	直徑程式設計：OFF 正常位置，請參閱機台製造商	*PG*	m	●	●	●	●
DIAMOFA	軸專屬模態半徑程式設計：OFF 正常位置，請參閱機台製造商	*PG*	m	●	●	●	●
DIAMON	直徑程式設計： 開啟	*PG*	m	●	●	●	●
DIAMONA	軸專屬模態半徑程式設計： ON 如需啟用，請洽詢機台製造商	*PG*	m	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
DIC	相對非模態、軸專屬直徑程式設計	*PG*	n	●	●	●	●
DILF	回退路徑（長度）	*PG*	m	●	●	●	●
DISABLE	中斷 OFF	*PGA* 停用 / 重新啟用指派中斷程式（DISABLE，ENABLE） (頁 103)		●	●	●	●
DISC	變化圓弧過衝刀具半徑補正	*PG*	m	●	●	●	●
DISCL	快速進給動作之結束位置與加工平面間的間隙	*PG*		●	●	●	●
DISPLOF	抑制目前單節顯示	*PGA* 抑制目前的單節顯示（DISPLOF、DISPLON、ACTBLOCNO） (頁 148)		●	●	●	●
DISPLON	撤回對目前單節顯示的抑制	*PGA* 抑制目前的單節顯示（DISPLOF、DISPLON、ACTBLOCNO） (頁 148)		●	●	●	●
DISPR	重新定位之路徑差異	*PGA* 重新定位至輪廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (頁 424)	n	●	●	●	●
DISR	重新定位距離	*PGA* 重新定位至輪廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (頁 424)	n	●	●	●	●
DITE	螺紋旋出路徑	*PG*	m	●	●	●	●
DITS	螺紋旋進路徑	*PG*	m	●	●	●	●
DIV	整數除法	*PGA* 運算功能 (頁 58)		●	●	●	●
DL	選擇視位置而定的附加刀具偏移量（DL，總設定偏移量）	*PGA* 選擇附加偏移（DL） (頁 348)	m	-	-	-	-
DO	同步動作的關件字，當條件滿足時會觸發動作	*PGA* 動作（DO） (頁 492)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
DRFOF	停用手輪偏移（DRF）	*PG*	m	●	●	●	●
DRIVE	路徑速率相依式加速	*PG*	m	●	●	●	●
DRIVEA	啟用已程式設計之軸的彎道加速度特性曲線	*PG*		●	●	●	●
DYNFINISH	平滑化精加工之動態回應	*PG*	m	●	●	●	●
DYNNORM	標準動態回應	*PG*	m	●	●	●	●
DYNPOS	定位模式、攻牙的動態回應	*PG*	m	●	●	●	●
DYNROUGH	粗加工的動態回應	*PG*	m	●	●	●	●
DYNSEMIFIN	精加工的動態回應	*PG*	m	●	●	●	●
DZERO	將 TO 元件的所有 D 數量標記為無效	*PGA* 任意指派D數量無效的D數量（DZERO） (頁 383)		●	●	●	●
EAUTO	以最後 3 個點定義最後的曲線區段	*PGA* 曲線插補（ASPLINE、BSPLINE、CSPLINE、BAUTO、BNAT、BTAN、EAUTO、ENAT、ETAN、PW、SD、PL） (頁 208)	m	-	○	-	○
EGDEF	定義電子齒輪	*PGA* 定義電子齒輪（EGDEF） (頁 464)		-	-	-	-
EGDEL	刪除跟隨軸的耦合定義	*PGA* 刪除電子齒輪的定義（EGDEL） (頁 470)		-	-	-	-
EGOFC	連續地關閉電子齒輪	*PGA* 切換至電子變速箱（EGOFS，EGOFC） (頁 469)		-	-	-	-
EGOFS	選擇性地關閉電子齒輪	*PGA* 切換至電子變速箱（EGOFS，EGOFC） (頁 469)		-	-	-	-
EGON	啟動電子齒輪	*PGA* 切換至電子變速箱（EGOFS，EGOFC） (頁 469)		-	-	-	-
EGONSYN	啟動電子齒輪	*PGA* 切換至電子變速箱（EGOFS，EGOFC） (頁 469)		-	-	-	-

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
EGONSYNE	使用逼近模式的規格啟動電子齒輪	*PGA* 切換至電子變速箱（EGOFS，EGOFC） (頁 469)		-	-	-	-
ELSE	程式分支，若未滿足 IF 條件	*PGA* 具有可供選擇的（IF，ELSE，ENDIF）程式迴圈 (頁 89)		●	●	●	●
ENABLE	中斷 ON	*PGA* 停用 / 重新啟用指派中斷程式（DISABLE，ENABLE） (頁 103)		●	●	●	●
ENAT [4]	自然轉變至下一個移動單節	*PGA* 曲線插補（ASPLINE、BSPLINE、CSPLINE、BAUTO、BNAT、BTAN、EAUTO、ENAT、ETAN、PW、SD、PL） (頁 208)	m	-	○	-	○
ENDFOR	FOR 計數器迴圈之最終指令列	*PGA* 計數迴圈（FOR ... TO ...，ENDFOR） (頁 91)		●	●	●	●
ENDIF	IF 分支之最終指令列	*PGA* 具有可供選擇的（IF，ELSE，ENDIF）程式迴圈 (頁 89)		●	●	●	●
ENDLABEL	使用 REPEAR 的工件程式重覆之結尾標籤	*PGA*、*FB1（K1）* 重複程式區段（REPEAT，REPEATB，ENDLABEL，P） (頁 83)		●	●	●	●
ENDLOOP	無限程式迴圈 LOOP 之最終指令列	*PGA* 連續程式迴圈（LOOP，ENDLOOP） (頁 90)		●	●	●	●
ENDPROC	含起始指令列 PROC 之程式的最終指令列			●	●	●	●
ENDWHILE	WHILE 迴圈之最終指令列	*PGA* 在迴圈起點處，附有條件的程式迴圈（WHILE，ENDWHILE） (頁 93)		●	●	●	●
ETAN	沿切線變化至曲線開始處下一個移動單節	*PGA* 曲線插補（ASPLINE、BSPLINE、CSPLINE、BAUTO、BNAT、BTAN、EAUTO、ENAT、ETAN、PW、SD、PL） (頁 208)	m	-	○	-	○
EVERY	當條件從 FALSE 變化為 TRUE 時執行同步動作	*PGA* 週期性檢查條件（WHEN、WHENEVER、FROM、EVERY） (頁 490)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
EX	以指數標記法配置值的關鍵字	*PGA* 預定義使用者變數：算術參數（R） (頁 18)		●	●	●	●
EXECSTRING	字串變數的傳輸，該字串變數具有執行中的工件程式行。	*PGA* 間接程式設計工件程式行（EXECSTRING） (頁 56)		●	●	●	●
EXECTAB	執行取自動作表之元素	*PGA* 間接程式設計工件程式行（EXECSTRING） (頁 56)		●	●	●	●
EXECUTE	程式執行 ON	*PGA* 停用輪廓準備（EXECUTE） (頁 631)		●	●	●	●
EXP	指數函數範例	*PGA* 運算功能 (頁 58)		●	●	●	●
EXTCALL	執行外部子程式	*PGA* 執行外部子程式（EXTCALL） (頁 173)		●	●	●	●
EXTERN	宣告具有參數傳輸的副程式	*PGA* 沒有參數傳輸的呼叫： (頁 160)		●	●	●	●
F	進給率值（搭配 G4 時，停頓時間亦使用 F 程式設計）	*PG*		●	●	●	●
FA	軸進給率	*PG*	m	●	●	●	●
FAD	軟逼近與回退的進給率	*PG*		●	●	●	●
FALSE（偽）	邏輯常數： 不正確	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
FB	非模態進給率	*PG*		●	●	●	●
FCTDEF	定義多項式函數	*PGA* 線上刀具偏移（PUTFTOCF、FCTDEF、PUTFTOC、FTOCON、FTOCOF） (頁 358)		-	-	-	-
FCUB	依三次曲線之定進給率變數	*PGA* 進給率回應（FNORM，FLIN，FCUB，FPO） (頁 408)	m	●	●	●	●
FD	手輪手動倍率之路徑進給率	*PG*	n	●	●	●	●
FDA	手輪手動倍率軸進給率	*PG*	n	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
FENDNORM	角落減速 OFF	*PGA* 含轉角減速的進給率減慢功能（FENDNORM，G62、G621） (頁 244)	m	•	•	•	•
FFWOF [4]	前饋控制 OFF（關閉）	*PG*	m	•	•	•	•
FFWON	前饋控制 ON（開啟）	*PG*	m	•	•	•	•
FGREF	旋轉軸或方向軸之路徑參考係數的參考半徑（向量插補）	*PG*	m	•	•	•	•
FGROUP	含路徑進給率之軸的定義	*PG*		•	•	•	•
FI	存取框架資料之參數： 微調偏移量	*PGA* 讀取及變更框架元件（TR、FI、RT、SC、MI）(頁 259)		•	•	•	•
FIFOCTRL	前置處理緩衝器之控制	*PGA* 具有前置處理記憶體（STOPFIFO，STARTFIFO，FIFOCTRL，STOPRE）的程式順序 (頁 414)	m	•	•	•	•
FILEDATE	傳回最近對檔案做寫入存取的日期	*PGA* 讀出檔案資訊（FILEDATE、FILETIME、FILESIZE、FILESTAT、FILEINFO） (頁 127)		•	•	•	•
FILEINFO	傳回列出 FILEDATE、FILESIZE、FILESTAT、與 FILETIME 的摘要資訊	*PGA* 讀出檔案資訊（FILEDATE、FILETIME、FILESIZE、FILESTAT、FILEINFO） (頁 127)		•	•	•	•
FILESIZE	傳回目前的檔案大小	*PGA* 讀出檔案資訊（FILEDATE、FILETIME、FILESIZE、FILESTAT、FILEINFO） (頁 127)		•	•	•	•
FILESTAT	傳回檔案之讀取、寫入、執行、顯示、刪除權限之狀態（rwxsd）	*PGA* 讀出檔案資訊（FILEDATE、FILETIME、FILESIZE、FILESTAT、FILEINFO） (頁 127)		•	•	•	•
FILETIME	傳回最近對檔案做寫入存取的時間	*PGA* 讀出檔案資訊（FILEDATE、FILETIME、FILESIZE、FILESTAT、FILEINFO） (頁 127)		•	•	•	•

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - D - - - - -	828D [3] PPU260 / 261 - - - F - - - - -	828D [3] PPU280 / 281 D - - - - -	828D [3] PPU280 / 281 F - - - - -
FINEA	於達到“精確停止微調”時動作結束	*PGA* 已程式設計的動作結尾條件（FINEA、COARSEA、IPOENDA、IPOBRKA、ADISPOSA） (頁 245)	m	●	●	●	●
FL	同步軸的臨界速率	*PG* 讀取及變更框架元件（TR、FI、RT、SC、MI） (頁 259)	m	●	●	●	●
FLIN	進給線性變數	*PGA* 進給率回應（FNORM，FLIN，FCUB，FPO） (頁 408)	m	●	●	●	●
FMA	軸的多重進給率	*PG*	m	-	-	-	-
FNORM [4]	垂直於 DIN 66025 之進給率	*PGA* 進給率回應（FNORM，FLIN，FCUB，FPO） (頁 408)	m	●	●	●	●
FOCOF	移動以受限扭矩 / 力道停用	*PGA* 移動至固定停止點（FXS、FXST、FXSW、FOCON、FOCOF） (頁 544)	m	○	-	○	-
FOCON	移動以受限扭矩 / 力道啟用	*PGA* 移動至固定停止點（FXS、FXST、FXSW、FOCON、FOCOF） (頁 544)	m	○	-	○	-
FOR	含固定通過次數之計數器迴圈	*PGA* 計數迴圈（FOR ... TO ...，ENDFOR） (頁 91)		●	●	●	●
FP	固定點： 待逼近之固定點數量	*PG*	n	●	●	●	●
FPO	透過多項式程式設計的進給率特性	*PGA* 進給率回應（FNORM，FLIN，FCUB，FPO） (頁 408)		-	-	-	-
FPR	旋轉軸識別碼	*PG*		●	●	●	●
FPRAOF	停用旋轉進給率	*PG*		●	●	●	●
FPRAON	啟動旋轉進給率	*PG*		●	●	●	●
FRAME	座標系統定義的日期類型	*PGA* 定義新框架（DEF FRAME） (頁 261)		●	●	●	●
FRC	半徑及倒角之進給率	*PG*	n	●	●	●	●
FRCM	半徑及倒角之進給率，模態	*PG*	m	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
FROM	一旦滿足該條件且啟用同步動作時便會執行該動作	*PGA* 週期性檢查條件（WHEN、WHENEVER、FROM、EVERY） (頁 490)		●	●	●	●
FTOC	更改微調刀具偏移	*PG* 線上刀具偏移（PUTFTOCF、FCTDEF、PUTFTOC、FTOCON、FTOCOF） (頁 358)		●	●	●	●
FTOCOF [4]	線上微調刀具偏移 OFF	*PGA* 線上刀具偏移（PUTFTOCF、FCTDEF、PUTFTOC、FTOCON、FTOCOF） (頁 358)	m	●	●	●	●
FTOCON	線上微調刀具偏移 ON	*PGA* 線上刀具偏移（PUTFTOCF、FCTDEF、PUTFTOC、FTOCON、FTOCOF） (頁 358)	m	●	●	●	●
FXS	“啟至固定停止點”開啟	*PG* 移動至固定停止點（FXS、FXST、FXSW、FOCON、FOCOF） (頁 544)	m	●	●	●	●
FXST	移動至固定點的轉矩限制。	*PG* 移動至固定停止點（FXS、FXST、FXSW、FOCON、FOCOF） (頁 544)	m	●	●	●	●
FXSW	監控視窗以移動至固定停止點	*PG* 移動至固定停止點（FXS、FXST、FXSW、FOCON、FOCOF） (頁 544)		●	●	●	●
FZ	刀齒進給率	*PG*	m	●	●	●	●
G0	線性插補含快速移動（快速移動動作）	*PG*	m	●	●	●	●
G1 [4]	使用進給率進行線性插補（線性插補）	*PG*	m	●	●	●	●
G2	順時針圓弧插補	*PG*	m	●	●	●	●
G3	逆時針圓弧插補	*PG*	m	●	●	●	●
G4	停頓時間，預設	*PG*	n	●	●	●	●
G5	斜向直面進給研磨。	*PGA* 傾斜軸（TRAANG） (頁 328)	n	●	●	●	●
G7	傾斜降切研磨時之補正動作	*PGA* 傾斜軸（TRAANG） (頁 328)	n	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
G9	精確停止－減速	*PG*	n	●	●	●	●
G17 [4]	作用平面 X/Y 之選擇	*PG*	m	●	●	●	●
G18	作用平面 Z/X 之選擇	*PG*	m	●	●	●	●
G19	作用平面 Y/Z 之選擇	*PG*	m	●	●	●	●
G25	工作區下限	*PG*	n	●	●	●	●
G26	工作區上限	*PG*	n	●	●	●	●
G33	以恆定螺距進行螺紋切削	*PG*	m	●	●	●	●
G34	使用線性遞增螺距進行螺紋切削	*PG*	m	●	●	●	●
G35	使用線性遞減螺距進行螺紋切削	*PG*	m	●	●	●	●
G40 [4]	刀具半徑補正 OFF（關閉）	*PG*	m	●	●	●	●
G41	刀具半徑補正輪廓左側	*PG*	m	●	●	●	●
G42	刀具半徑補正輪廓右側	*PG*	m	●	●	●	●
G53	抑制目前的工作偏移（非模態）	*PG*	n	●	●	●	●
G54	第 1 個可調整的工件偏移量	*PG*	m	●	●	●	●
G55	2. 可調整工件偏移量	*PG*	m	●	●	●	●
G56	3. 可調整工件偏移量	*PG*	m	●	●	●	●
G57	4. 可調整工件偏移量	*PG*	m	●	●	●	●
G58	軸可程式設計的工件偏移量、絕對、粗調偏移量	*PG*	n	●	●	●	●
G59	軸可程式設計的工件偏移量、附加的、精調偏移量	*PG*	n	●	●	●	●
G60 [4]	精確停止 - 速率減速	*PG*	m	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
G62	啟動刀具半徑偏移（G41，G42）時，於內角的角度減量	*PGA* 含轉角減速的進給率減慢功能（FENDNORM，G62、G621）(頁 244)	m	●	●	●	●
G63	搭配補正夾頭進行攻牙	*PG*	n	●	●	●	●
G64	連續路徑模式	*PG*	m	●	●	●	●
G70	用於幾何規格（長度）的英制尺寸	*PG*	m	●	●	●	●
G71 [4]	用於幾何規格（長度）的公制尺寸	*PG*	m	●	●	●	●
G74	參考點搜尋	*PG*	n	●	●	●	●
G75	逼近固定點	*PG*	n	●	●	●	●
G90 [4]	絕對尺寸	*PG*	m/n	●	●	●	●
G91	增量尺寸	*PG*	m/n	●	●	●	●
G93	反向時間進給率 rpm	*PG*	m	●	●	●	●
G94 [4]	線性進給率 F，單位為毫米 / 分或英吋 / 分以及度 / 分	*PG*	m	●	●	●	●
G95	旋轉進給率 F，單為為毫米 / 轉或英吋 / 轉	*PG*	m	●	●	●	●
G96	恆定切削率（供 G95 使用）ON	*PG*	m	●	●	●	●
G97	恆定切削率（供 G95 使用）OFF	*PG*	m	●	●	●	●
G110	極點程式設計相對於上一次已程式設計之設定點的位置	*PG*	n	●	●	●	●
G111	極點程式設計相對於目前工件座標系統的零點	*PG*	n	●	●	●	●
G112	極點程式設計相對於上一個有效的極點	*PG*	n	●	●	●	●
G140 [4]	以 G41/G42 定義之 SAR 逼近方向	*PG*	m	●	●	●	●
G141	朝輪廓左側逼近之 SAR 逼近方向	*PG*	m	●	●	●	●
G142	朝輪廓右側逼近之 SAR 逼近方向	*PG*	m	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
G143	相依於切線之 SAR 逼近方向	*PG*	m	●	●	●	●
G147	以直線方式緩逼近	*PG*	n	●	●	●	●
G148	以直線方式緩返回	*PG*	n	●	●	●	●
G153	抑制含基本框架的目前框架	*PG*	n	●	●	●	●
G247	以四分之一圓方式緩逼近	*PG*	n	●	●	●	●
G248	以四分之一圓方式緩返回	*PG*	n	●	●	●	●
G290	切換至 SINUMERIK 模式 ON	*FBW*	m	●	●	●	●
G291	切換至 ISO2/3 模式 ON	*FBW*	m	●	●	●	●
G331	剛性攻牙、正螺距、順時針	*PG*	m	●	●	●	●
G332	剛性攻牙、負螺距、逆時針	*PG*	m	●	●	●	●
G340 [4]	立體逼近單節（同時進行深度與平面（螺旋線））	*PG*	m	●	●	●	●
G341	垂直軸（z）上之初始進給，然後逼近平面	*PG*	m	●	●	●	●
G347	以半圓緩逼近	*PG*	n	●	●	●	●
G348	以半圓緩返回	*PG*	n	●	●	●	●
G450 [4]	圓形變化	*PG*	m	●	●	●	●
G451	同距交點	*PG*	m	●	●	●	●
G460 [4]	啟動逼近與回退單節的碰撞偵測	*PG*	m	●	●	●	●
G461	將圓弧插入至 TRC 單節中	*PG*	m	●	●	●	●
G462	將直線插入至 TRC 單節中	*PG*	m	●	●	●	●
G500 [4]	停用所有可調整的框架，啟用基本框架	*PG*	m	●	●	●	●
G505 到 G599	5 ... 99. 可調整工件偏移量	*PG*	m	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - D - - - - - F	PPU280 / 281 D - - - - - F		
G601 [4]	於精確停止微調處之單節變更	*PG*	m	●	●	●	●
G602	於精確停止粗調處之單節變更	*PG*	m	●	●	●	●
G603	於 IPO 單節結尾處之單節變更	*PG*	m	●	●	●	●
G621	所有角的轉角減速	*PGA* 含轉角減速的進給率減慢功能（FENDNORM，G62、G621） (頁 244)	m	●	●	●	●
G641	具有符合路徑條件（=可程式設計的倒圓角間隙）之平滑化的連續路徑模式	*PG*	m	●	●	●	●
G642	具在定義允差內之平滑化的連續路徑模式	*PG*	m	●	●	●	●
G643	具有在定義允差內（單節內部）之平滑化的連續路徑模式	*PG*	m	●	●	●	●
G644	具有含最大可能動態回應之平滑化的連續路徑模式	*PG*	m	●	●	●	●
G645	具有在定義允差內之平滑化與切線單節變化的連續路徑模式	*PG*	m	●	●	●	●
G700	用於幾何與技術規格（長度、進給率）的英制尺寸	*PG*	m	●	●	●	●
G710 [4]	用於幾何與技術規格（長度、進給率）的公制尺寸	*PG*	m	●	●	●	●
G751	經由中間點逼進固定點	PG	n	●	●	●	●
G810 [4]、...、G819	保留給 OEM 使用者的 G 群組	*PGA* OEM使用者的特殊函數（OEMIPO1、OEMIPO2、G810 至G829） (頁 243)		●	●	●	●
G820 [4]、...、G829	保留給 OEM 使用者的 G 群組	*PGA* OEM使用者的特殊函數（OEMIPO1、OEMIPO2、G810 至G829） (頁 243)		●	●	●	●
G931	以移動時間指定之進給率		m	●	●	●	●
G942	凍結線性進給率及恆定切削速率或主軸速率		m	●	●	●	●
G952	凍結旋轉進給率及恆定切削速率或主軸速率		m	●	●	●	●
G961	恆定切削速率及線性進給率	*PG*	m	●	●	●	●
G962	線性或旋轉進給率及恆定切削速率	*PG*	m	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
G971	凍結主軸速率及線性進給率	*PG*	m	●	●	●	●
G972	凍結線性或旋轉進給率及恆定主軸速率	*PG*	m	●	●	●	●
G973	不含主軸速率限制之旋轉進給率	*PG*	m	●	●	●	●
GEOAX	指派新通道軸給幾何座標軸 1 - 3	*PGA* 可更換的幾何軸（GEOAX） (頁 591)		●	●	●	●
GET	更換通道間啟用的軸	*PGA* 座標軸替換，主軸替換（RELEASE，GET，GETD） (頁 110)		●	●	●	●
GETACTT	由刀具群組中找出具相同名稱的啟用中刀具	*FBW*		●	●	●	●
GETACTTD	取得具有絕對 D 編號的 T 編號	*PGA* 任意指派D數量決定指定D數量的T數量（GETACTTD） (頁 383)		●	●	●	●
GETD	直接更換通道間的軸	*PGA* 座標軸替換，主軸替換（RELEASE，GET，GETD） (頁 110)		●	●	●	●
GETDNO	傳回刀具（T）的刀刃（CE）之 D 編號	*PGA* 任意指派D數量重新命名D數量（GETDNO，SETDNO） (頁 382)		●	●	●	●
GETEXET	讀取載入的 T 編號	*FBW*		●	●	●	●
GETFREELOC	為已知刀具在刀庫中尋找可用空間	*FBW*		●	●	●	●
GETSELT	傳回選取的 T 編號	*FBW*		●	●	●	●
GETT	為刀具名稱取得 T 編號	*FBW*		●	●	●	●
GETTCOR	讀出刀長及（或）刀長元件	*FB1（W1）*		●	●	●	●
GETTENV	讀取 T、D 與 DL 編號	*FB1（W1）*		●	●	●	●
GOTO	先向前然後向後之跳躍操作（其初始方向是朝程式結尾，之後朝程式開始）	*PGA* 程式跳躍至跳躍標記（GOTOB、GOTOF、GOTO、GOTOC） (頁 78)		●	●	●	●
GOTOB	向後跳躍（前往程式開始）	*PGA* 程式跳躍至跳躍標記（GOTOB、GOTOF、GOTO、GOTOC） (頁 78)		●	●	●	●
GOTOC	同 GOTO，但抑制警報 14080“找不到跳躍目的地”	*PGA* 程式跳躍至跳躍標記（GOTOB、GOTOF、GOTO、GOTOC） (頁 78)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
GOTOF	向前跳躍（前往程式結尾）	*PGA* 程式跳躍至跳躍標記（GOTOB、GOTOF、GOTO、GOTOC） (頁 78)		●	●	●	●
GOTOS	向後跳躍至程式開始處	*PGA* 傳回跳躍至程式開始處（GOTOS） (頁 77)		●	●	●	●
GP	將位置屬性間接程式設計的關鍵字	*PGA* 間接程式設計定位屬性（BP） (頁 54)		●	●	●	●
GWPSOF	取消選擇研磨輪圓柱恆定速率（GWPS）	*PG*	n	●	●	●	●
GWPSON	選擇研磨輪圓柱恆定速率（GWPS）	*PG*	n	●	●	●	●
H...	輔助函數輸出至 PLC	*PG/FB1（H2）*		●	●	●	●
HOLES1	鑽孔模式循環，孔洞順序	*BHD/BHF*		●	●	●	●
HOLES2	鑽孔模式循環，孔洞圓弧	*BHD/BHF*		●	●	●	●
I	插補參數	*PG*	n	●	●	●	●
I1	中間點座標	*PG*	n	●	●	●	●
IC	增量尺寸	*PG*	n	●	●	●	●
ICYCOF	所有單節在 ICYCOF 之後都會在單一插補循環中處理	*PGA* 技術循環之控制處理（ICYCOF、ICYCON） (頁 555)		●	●	●	●
ICYCON	ICYCON 之後的每一個技術循環單節皆在不同的插補循環中處理	*PGA* 技術循環之控制處理（ICYCOF、ICYCON） (頁 555)		●	●	●	●
ID	模態同步動作的識別碼	*PGA* 適用區域與加工順序（ID、IDS） (頁 489)	m	●	●	●	●
IDS	模態靜態同步動作的識別碼	*PGA* 適用區域與加工順序（ID、IDS） (頁 489)		●	●	●	●
IF	工件程式 / 技術循環中傳統跳躍函數之介紹	*PGA* 具有可供選擇的（IF，ELSE，ENDIF）程式迴圈 (頁 89)		●	●	●	●
INDEX	定義輸入字串中字元的索引值	*PGA* 在字串中搜尋字元 / 字串（INDEX、RINDEX、MINDEX、MATCH） (頁 73)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
INIPO	在 POWER ON 將變數初始化	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
INIRE	在重置時將變數初始化	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
INICF	在 NewConfig 時將變數初始化	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
INIT	選擇特定 NC 程式，供特定通道執行：	*PGA* 程式一致性（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM） (頁 95)		-	-	-	-
INITIAL	產生一個適用所有區域的 INI 檔	*PGA* 工作記憶體（CHANDATA，COMPLETE，INITIAL） (頁 188)		●	●	●	●
INT	資料類型： 含正負號之整數	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
INTERSEC	計算兩個輪廓元素間的交叉點	*PGA* 決定兩輪廓元素間的交點（INTERSEC） (頁 626)		●	●	●	●
INVCCW	軌跡漸開線，逆時針	*PG*	m	-	-	-	-
INVCW	軌跡漸開線，順時針	*PG*	m	-	-	-	-
INVFRAME	從框架計算出反向框架	*FB1（K2）*		●	●	●	●
IP	變數插補參數	*PGA* 間接程式設計 (頁 50)		●	●	●	●
IPOBRKA	自啟用煞車斜率開始動作條件	*PGA* 已程式設計的動作結尾條件（FINEA、COARSEA、IPOENDA、IPOBRKA、ADISPOSA） (頁 245)	m	●	●	●	●
IPOENDA	於抵達 “IPO 停止”時動作結束	*PGA* 已程式設計的動作結尾條件（FINEA、COARSEA、IPOENDA、IPOBRKA、ADISPOSA） (頁 245)	m	●	●	●	●
IPTRLOCK	將下一個加工函式單節中無法追蹤的程式區段之啟動予以凍結	*PGA* 防止SERUPRO的程式位置（IPTRLOCK，IPTRUNLOCK） (頁 422)	m	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - D - - - - - F	PPU280 / 281 D - - - - - F
IPTRUNLOCK	中斷時，設定在目前的單節中無法追蹤之程式區段的結束	*PGA* 防止SERUPRO的程式位置（IPTRLOCK，IPTRUNLOCK） (頁 422)	m	●	● ● ●
ISAXIS	檢查幾何座標軸 1 是否被指定為參數	*PGA* 軸函數（AXNAME、AX、SPI、AXTOSPI、ISAXIS、AXSTRING、MODAXVAL） (頁 589)		●	● ● ●
ISD	插入深度	*PGA* 啟動 3D刀具偏移（CUT3DC、CUT3DF、CUT3DFS、CUT3DFF、ISD） (頁 363)	m	●	● ● ●
ISFILE	檢查檔案在 NCK 應用程式記憶體中是否存在	*PGA* 確認檔案（ISFILE）是否存在 (頁 125)		●	● ● ●
ISNUMBER	檢查輸入字串是否可轉換為數字	*PGA* 從STRING類型轉換（NUMBER、ISNUMBER、AXNAME） (頁 69)		●	● ● ●
ISOCALL	間接呼叫以 ISO 語言程式設計的程式	*PGA* 間接呼叫以ISO語言（ISOCALL）進行程式設計的程式 (頁 170)		●	● ● ●
ISVAR	檢查傳輸參數是否包含在 NC 中已宣告的變數	*PGA* 函數呼叫ISVAR及讀取機械參數陣列索引 (頁 603)		●	● ● ●
J	插補參數	*PG*	n	●	● ● ●
J1	中間點座標	*PG*	n	●	● ● ●
JERKA	透過 MD 為已程式設計之軸啟動加速度回應			●	● ● ●
JERKLIM	最大軸向震動之減少或過衝	*PGA* 比例震動校正（JERKLIM） (頁 432)	m	●	● ● ●
JERKLIMA	最大軸向震動之減少或過衝	*PG*	m	●	● ● ●
K	插補參數	*PG*	n	●	● ● ●
K1	中間點座標	*PG*	n	●	● ● ●
KONT	在刀具偏移上沿輪廓移動	*PG*	m	●	● ● ●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
KONTC	利用連續曲率多項式逼近 / 回退	*PG*	m	●	●	●	●
KONTT	利用連續切線多項式逼近 / 回退	*PG*	m	●	●	●	●
L	副程式編號	*PGA* 沒有參數傳輸的呼叫： (頁 160)	n	●	●	●	●
LEAD	螺距角度 1. 刀具方向 2. 方向多項式	*PGA* 程式設計刀具方向（A...、B...、C...、LEAD、TILT） (頁 286)	m	● -	● -	● -	● -
LEADOF	主動值耦合 OFF	*PGA* 軸先導值耦合（LEADON，LEADOF） (頁 459)		-	-	-	-
LEADON	主動值耦合開啟	*PGA* 軸先導值耦合（LEADON，LEADOF） (頁 459)		-	-	-	-
LENTOAX	將有關啟用刀具的刀長配置資訊 L1、L2 與 L3 提供給橫座標、縱座標及備選座標	*FB1（W1）*		●	●	●	●
LFOF [4]	關閉螺紋切削時的快速回退	*PG*	m	●	●	●	●
LFON	開啟螺紋切削時的快速回退	*PG*	m	●	●	●	●
LFPOS	LFPOS 將以 POLFMASK 或 POLFMLIN 宣告的軸回退至以 POLF 程式設計的絕對軸位置	*PG*	m	●	●	●	●
LFTXT	快速回退時的回退移動之平面由目前刀具方向的路徑切線決定	*PG*	m	●	●	●	●
LFWP	快速回退時的回退移動之平面由目前的工作平面（G17/G18/G19）決定	*PG*	m	●	●	●	●
LIFTFAST	快速回退	*PG* 從輪廓快速回退（SETINT LIFTFAST, ALF） (頁 105)		●	●	●	●
LIMS	G96/G961 與 G97 的速度臨界值	*PG*	m	●	●	●	●
LLI	變數中的下臨界值	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
LN	自然對數	*PGA* 運算功能 (頁 58)		●	●	●	●
LOCK	利用 ID 停用同步化動作（停止技術循環）	*PGA* 鎖定、解除鎖定、重置（LOCK、UNLOCK、RESET） (頁 557)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
LONGHOLE	圓弧上細長孔的銑削模式循環	*BHD/BHF*		-	-	-	-
LOOP	無限迴圈之介紹	*PGA* 連續程式迴圈（LOOP，ENDLOOP） (頁 90)		●	●	●	●
M0	程式設計停止	*PG*		●	●	●	●
M1	選擇性停止	*PG*		●	●	●	●
M2	以返回程式起點的方式結束主程式	*PG*		●	●	●	●
M3	主動主軸的旋轉方向為順時針	*PG*		●	●	●	●
M4	主動主軸的旋轉方向為逆時針	*PG*		●	●	●	●
M5	主軸停止用於主動主軸	*PG*		●	●	●	●
M6	刀具換用	*PG*		●	●	●	●
M17	副程式結尾	*PG*		●	●	●	●
M19	供 SSL 累積主軸程式設計之用	*PG*		●	●	●	●
M30	程式結束，效果同 M2	*PG*		●	●	●	●
M40	自動檔位變更	*PG*		●	●	●	●
M41 至 M45	齒輪檔位 1，...，5	*PG*		●	●	●	●
M70	變化至軸模式	*PG*		●	●	●	●
MASLDEF	定義主動 / 從動軸群組	*PGA* 主控 / 從屬群組（MASLDEF、MASLDEL、MASLON、MASLOF、MASLOFS） (頁 483)		●	●	●	●
MASLDEL	解除主動 / 從動軸群組之耦合並清除分組定義	*PGA* 主控 / 從屬群組（MASLDEF、MASLDEL、MASLON、MASLOF、MASLOFS） (頁 483)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
MASLOF	停用暫時耦合	*PGA* 主控 / 從屬群組（MASLDEF、MASLDEL、MASLON、MASLOF、MASLOFS） (頁 483)		●	●	●	●
MASLOFS	透過附屬軸自動停止停用暫時耦合	*PGA* 主控 / 從屬群組（MASLDEF、MASLDEL、MASLON、MASLOF、MASLOFS） (頁 483)		●	●	●	●
MASLON	啟用暫時耦合	*PGA* 主控 / 從屬群組（MASLDEF、MASLDEL、MASLON、MASLOF、MASLOFS） (頁 483)		●	●	●	●
MATCH	在字串中搜尋字串	*PGA* 在字串中搜尋字元 / 字串（INDEX、RINDEX、MINDEX、MATCH） (頁 73)		●	●	●	●
MAXVAL	兩變數中較大數值（數學函數）	*PGA* 變數最小，最大與範圍（MINVAL，MAXVAL與BOUND） (頁 64)		●	●	●	●
MCALL	模態副程式呼叫	*PGA* 模態副程式呼叫（MCALL） (頁 166)		●	●	●	●
MEAC	不含刪除之剩餘距離的持續量測	*PGA* 延伸量測函數（MEASA、MEAWA、MEAC）（選用） (頁 235)	n	-	-	-	-
MEAFRAME	由量測點進行之框架計算	*PGA* 從空間中的三個量測點計算框架（MEAFRAME） (頁 266)		●	●	●	●
MEAS	使用觸發式測針量測	*PGA* 使用觸發式探針量測（MEAS、MEAW） (頁 232)	n	●	●	●	●
MEASA	量測時含刪除的剩餘距離	*PGA* 延伸量測函數（MEASA、MEAWA、MEAC）（選用） (頁 235)	n	-	-	-	-
量測	工件與刀具量測的計算方法	*FB2（M5）* 使用觸發式探針量測（MEAS、MEAW） (頁 232)		●	●	●	●
MEAW	以觸發式探針進行量測，不刪除剩餘距離	*PGA* 使用觸發式探針量測（MEAS、MEAW） (頁 232)	n	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - D - - - - - F	PPU280 / 281 D - - - - - F		
MEAWA	量測時不含刪除的剩餘距離	*PGA* 延伸量測函數（MEASA、MEAWA、MEAC）（選用） (頁 235)	n	-	-	-	-
MI	存取框架資料： 鏡像	*PGA* 讀取及變更框架元件（TR、FI、RT、SC、MI） (頁 259)		●	●	●	●
MINDEX	定義輸入字串中字元的索引值	*PGA* 在字串中搜尋字元 / 字串（INDEX、RINDEX、MINDEX、MATCH） (頁 73)		●	●	●	●
MINVAL	兩變數中較小數值（數學函數）	*PGA* 變數最小，最大與範圍（MINVAL，MAXVAL與BOUND） (頁 64)		●	●	●	●
MIRROR	可程式設計的鏡像	*PGA*	n	●	●	●	●
MMC	由 HMI 上之工件程式以互動方式呼叫對話方塊	*PGA* 由工件程式相互呼叫視窗（MMC） (頁 607)		●	●	●	●
MOD	模態除法	*PGA* 運算功能 (頁 58)		●	●	●	●
MODAXVAL	定義模態旋轉軸之模態位置	*PGA* 軸函數（AXNAME、AX、SPI、AXTOSPI、ISAXIS、AXSTRING、MODAXVAL） (頁 589)		●	●	●	●
MOV	啟動定位軸	*PGA* 啟動 / 停止軸（MOV） (頁 530)		●	●	●	●
MSG	可程式設計訊息	*PG*	m	●	●	●	●
MVTOOL	用於移動刀具的語言指令	*FBW*		●	●	●	●
N	NC 輔助單節編號	*PG*		●	●	●	●
NCK	資料有效範圍的規格	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
NEWCONF	套用修改後的機械參數（對應至"啟動機械參數"）	*PGA* 啟動機械參數（NEWCONF） (頁 116)		●	●	●	●
NEWT	建立新增刀具	*PGA* 讀取及變更框架元件（TR、FI、RT、SC、MI） (頁 259)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - D - - - - - F - - - - -		PPU280 / 281 D - - - - - F - - - - -	
NORM [4]	起點及終點之標準設定，含刀具偏移	*PG*	m	●	●	●	●
NOT	邏輯 NOT（否定）	*PGA* 比較與邏輯運算 (頁 60)		●	●	●	●
NPROT	機台專屬之保護區 ON/OFF	*PGA* 啟動 / 停用保護區（CPROT、NPROT） (頁 196)		●	●	●	●
NPROTDEF	定義機台專屬保護區	*PGA* 保護區（CPROTDEF、NPROTDEF）定義 (頁 193)		●	●	●	●
NUMBER	將輸入字串轉換成數字	*PGA* 從STRING類型轉換（NUMBER、ISNUMBER、AXNAME） (頁 69)		●	●	●	●
OEMIPO1	OEM 插補 1	*PGA* OEM使用者的特殊函數（OEMIPO1、OEMIPO2、G810 至G829） (頁 243)	m	●	●	●	●
OEMIPO2	OEM 插補 2	*PGA* OEM使用者的特殊函數（OEMIPO1、OEMIPO2、G810 至G829） (頁 243)	m	●	●	●	●
OF	含 CASE 分支之關鍵字	*PGA* 程式分支（CASE ... OF ... DEFAULT ...） (頁 81)		●	●	●	●
OFFN	已程式設計之輪廓的允差	*PG*	m	●	●	●	●
OMA1	OEM 位址 1		m	●	●	●	●
OMA2	OEM 位址 2		m	●	●	●	●
OMA3	OEM 位址 3		m	●	●	●	●
OMA4	OEM 位址 4		m	●	●	●	●
OMA5	OEM 位址 5		m	●	●	●	●
OR	邏輯運算子，OR 操作	*PGA* 比較與邏輯運算 (頁 60)		●	●	●	●
ORIAXES	機械軸或方向軸之線性插補	*PGA* 程式設計方向軸（ORIAXES，ORIVECT，ORIEULER，ORIRPY，ORIRPY2，ORIVIRT1，ORIVIRT2） (頁 296)	m	●	●	●	●
ORIAXPOS	透過虛擬方向軸搭配旋轉軸位置取得之方向角		m	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
ORIC [4]	在待插入之圓弧單節上外部角落處的方向變更是覆蓋的	*PGA* 刀具方向（ORIC、ORID、OSOF、OSC、OSS、OSSE、ORIS、OSD、OST） (頁 375)	m	•	•	•	•
ORICONCCW	圓柱面上逆時針方向之插補	*PGA/FB3（F3）* 沿著錐形的柱面進行方向程式設計（ORIPLANE，ORICONCW，ORICONCCW，ORICONTO，ORICONIO） (頁 298)	m	•	•	•	•
ORICONCW	圓柱面上順時針方向之插補	*PGA/FB3（F4）* 沿著錐形的柱面進行方向程式設計（ORIPLANE，ORICONCW，ORICONCCW，ORICONTO，ORICONIO） (頁 298)	m	•	•	•	•
ORICONIO	在圓柱面上依照中間方向設定進行插補	*PGA/FB3（F4）* 沿著錐形的柱面進行方向程式設計（ORIPLANE，ORICONCW，ORICONCCW，ORICONTO，ORICONIO） (頁 298)	m	•	•	•	•
ORICONTO	在圓柱面上依照切線變化進行插補（最終方向）	*PGA/FB3（F5）* 沿著錐形的柱面進行方向程式設計（ORIPLANE，ORICONCW，ORICONCCW，ORICONTO，ORICONIO） (頁 298)	m	•	•	•	•
ORICURVE	搭配刀具兩接點之規格進行方向插補	*PGA/FB3（F6）* 兩接點的方向規格（ORICURVE、PO[XH]=、PO[YH]=、PO[ZH]=） (頁 301)	m	•	•	•	•
ORID	方向變更在圓弧單節之前進行	*PGA* 刀具方向（ORIC、ORID、OSOF、OSC、OSS、OSSE、ORIS、OSD、OST） (頁 375)	m	•	•	•	•
ORIEULER	透過尤拉角取得之方向角	*PGA* 程式設計方向軸（ORIAXES，ORIVECT，ORIEULER，ORIRPY，ORIRPY2，ORIVIRT1，ORIVIRT2） (頁 296)	m	•	•	•	•
ORIMKS	機械座標系統中之刀具方向	*PGA* 方向軸參考（ORIWKS、ORIMKS） (頁 294)	m	•	•	•	•
ORIPATH	與路徑相關的刀具方向	*PGA* 相對於路徑的刀具方向旋轉（ORIPATH、ORIPATHS、旋轉角度） (頁 309)	m	•	•	•	•

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - D - - - - -	PPU260 / 261 - - - F - - - - -	PPU280 / 281 D - - - - -	PPU280 / 281 F - - - - -
ORIPATHS	刀具相對於路徑之方向，方向特性中的上下變動經平滑化化	*PGA* 相對於路徑的刀具方向旋轉（ORIPATH、ORIPATHS、旋轉角度） (頁 309)	m	●	●	●	●
ORIPLANE	平面中插補（對應於 ORIVECT），大半徑圓弧插補	*PGA* 沿著錐形的柱面進行方向程式設計（ORIPLANE，ORICONCW，ORICONCCW，ORICONTO，ORICONIO） (頁 298)	m	●	●	●	●
ORIRESET	具有最多 3 個方向軸的初始刀具方向	*PGA* 方向程式設計及初始設定的變數（ORIRESET） (頁 285)		●	●	●	●
ORIROTA	絕對旋轉方向的旋轉角度	*PGA* 刀具方向的旋轉（ORIROTA、ORIROTR、ORIROTT、ORIROTC、THETA） (頁 305)	m	●	●	●	●
ORIROTC	相對於路徑切線之切線旋轉向量	*PGA* 刀具方向的旋轉（ORIROTA、ORIROTR、ORIROTT、ORIROTC、THETA） (頁 305)	m	●	●	●	●
ORIROTR	相對於起始角度及結束角度間之平面的旋轉角度	*PGA* 刀具方向的旋轉（ORIROTA、ORIROTR、ORIROTT、ORIROTC、THETA） (頁 305)	m	●	●	●	●
ORIROTT	相對於方向向量上之變更的旋轉角度	*PGA* 刀具方向的旋轉（ORIROTA、ORIROTR、ORIROTT、ORIROTC、THETA） (頁 305)	m	●	●	●	●
ORIRPY	透過 RPY 角度（XYZ）取得之方向角度	*PGA* 程式設計方向軸（ORIAXES，ORIVECT，ORIEULER，ORIRPY，ORIRPY2，ORIVIRT1，ORIVIRT2） (頁 296)	m	●	●	●	●
ORIRPY2	透過 RPY 角度（ZYX）取得之方向角度	*PGA* 程式設計方向軸（ORIAXES，ORIVECT，ORIEULER，ORIRPY，ORIRPY2，ORIVIRT1，ORIVIRT2） (頁 296)	m	●	●	●	●
ORIS	變更方向	*PGA* 刀具方向（ORIC、ORID、OSOF、OSC、OSS、OSSE、ORIS、OSD、OST） (頁 375)	m	●	●	●	●
ORISOF [4]	方向特性平滑化 OFF	*PGA* 平滑化方向特性（ORISON，ORISOF） (頁 316)	m	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
ORISON	方向特性平滑化 ON	*PGA* 平滑化方向特性（ORISON，ORISOF） (頁 316)	m	●	●	●	●
ORIVECT	大半徑圓弧插補（與 ORIPLANE 相同）	*PGA* 程式設計方向軸（ORIAXES，ORIVECT，ORIEULER，ORIRPY，ORIRPY2，ORIVIRT1，ORIVIRT2） (頁 296)	m	●	●	●	●
ORIVIRT1	透過虛擬方向軸取得之方向角度（定義 1）	*PGA* 程式設計方向軸（ORIAXES，ORIVECT，ORIEULER，ORIRPY，ORIRPY2，ORIVIRT1，ORIVIRT2） (頁 296)	m	●	●	●	●
ORIVIRT2	透過虛擬方向軸取得之方向角度（定義 1）	*PGA* 程式設計方向軸（ORIAXES，ORIVECT，ORIEULER，ORIRPY，ORIRPY2，ORIVIRT1，ORIVIRT2） (頁 296)	m	●	●	●	●
ORIWKS [4]	工件座標系統中之刀具方向	*PGA* 方向軸參考（ORIWKS、ORIMKS） (頁 294)	m	●	●	●	●
OS	振盪開啟 / 關閉	*PGA* 非同步震盪（OS、OSP1、OSP2、OST1、OST2、OSCTRL、OSNSC、OSE、OSB） (頁 563)		-	-	-	-
OSB	擺動： 起點	*FB2（P5）*	m	-	-	-	-
OSC	連續性刀具方向平滑化處理	*PGA* 刀具方向（ORIC、ORID、OSOF、OSC、OSS、OSSE、ORIS、OSD、OST） (頁 375)	m	●	●	●	●
OSCILL	軸： 1 - 3 進給軸	*PGA* 由同步動作控制的震盪（OSCILL） (頁 567)	m	-	-	-	-
OSCTRL	震盪選項	*PGA* 非同步震盪（OS、OSP1、OSP2、OST1、OST2、OSCTRL、OSNSC、OSE、OSB） (頁 563)	m	-	-	-	-
OSD	利用 SD 指定平滑化距離進行刀具方向之平滑化	*PGA* 刀具方向（ORIC、ORID、OSOF、OSC、OSS、OSSE、ORIS、OSD、OST） (頁 375)	m	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
OSE	擺動結束位置	*PGA* 非同步震盪（OS、OSP1、OSP2、OST1、OST2、OSCTRL、OSNSC、OSE、OSB） (頁 563)	m	-	-	-	-
OSNSC	擺動： 滑磨循環編號	*PGA* 非同步震盪（OS、OSP1、OSP2、OST1、OST2、OSCTRL、OSNSC、OSE、OSB） (頁 563)	m	-	-	-	-
OSOF [4]	刀具方向平滑化處理 OFF	*PGA* 非同步震盪（OS、OSP1、OSP2、OST1、OST2、OSCTRL、OSNSC、OSE、OSB） (頁 563)	m	●	●	●	●
OSP1	擺動： 左反轉點	*PGA* 非同步震盪（OS、OSP1、OSP2、OST1、OST2、OSCTRL、OSNSC、OSE、OSB） (頁 563)	m	-	-	-	-
OSP2	碰撞右反轉點	*PGA* 非同步震盪（OS、OSP1、OSP2、OST1、OST2、OSCTRL、OSNSC、OSE、OSB） (頁 563)	m	-	-	-	-
OSS	於單節結尾進行之刀具方向平滑化處理	*PGA* 刀具方向（ORIC、ORID、OSOF、OSC、OSS、OSSE、ORIS、OSD、OST） (頁 375)	m	●	●	●	●
OSSE	於單節開始及結尾進行之刀具方向平滑化處理	*PGA* 刀具方向（ORIC、ORID、OSOF、OSC、OSS、OSSE、ORIS、OSD、OST） (頁 375)	m	●	●	●	●
OST	利用 SD 以角度為單位指定角度允差對刀具方向進行平滑（與設計之方向特性間最大的偏差）	*PGA* 刀具方向（ORIC、ORID、OSOF、OSC、OSS、OSSE、ORIS、OSD、OST） (頁 375)	m	●	●	●	●
OST1	擺動： 左逆轉點之停止點	*PGA* 非同步震盪（OS、OSP1、OSP2、OST1、OST2、OSCTRL、OSNSC、OSE、OSB） (頁 563)	m	-	-	-	-
OST2	擺動： 右逆轉點之停止點	*PGA* 非同步震盪（OS、OSP1、OSP2、OST1、OST2、OSCTRL、OSNSC、OSE、OSB） (頁 563)	m	-	-	-	-

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
OTOL	壓縮函數、方向平滑化、與平滑化類型的方向允差	*PGA* 可程式設計輪廓/方向允差（CTOL、OTOL、ATOL）(頁 434)		-	●	-	●
OVR	速度偏移量	*PGA*	m	●	●	●	●
OVRA	軸速度偏移量	*PGA*	m	●	●	●	●
OVRRAP	快速進給速率手動倍率	*PGA*	m	●	●	●	●
P	副程式循環數量	*PGA* 程式重覆的次數（P）(頁 164)		●	●	●	●
PAROT	對齊工件上的工件座標系統	*PG*	m	●	●	●	●
PAROTOF	停用與相對於工件的框架旋轉	*PG*	m	●	●	●	●
PCALL	利用絕對路徑及參數傳輸呼叫副程式	*PGA* 以路徑規格和參數（PCALL）呼叫子程式 (頁 171)		●	●	●	●
PDELAYOF	含延遲的沖孔 OFF（關閉）	*PGA* 沖孔與切片開啟或關閉（SPOF、SON、PON、SONS、PONS、PDELAYON、PDELAYOF、PUNCHACC）(頁 575)	m	-	-	-	-
PDELAYON [4]	含延遲的沖孔 ON（開啟）	*PGA* 沖孔與切片開啟或關閉（SPOF、SON、PON、SONS、PONS、PDELAYON、PDELAYOF、PUNCHACC）(頁 575)	m	-	-	-	-
PHU	變數的實體元件	*PGA* 使用者變數的定義（DEF）(頁 22)		●	●	●	●
PL	1. B 曲線： 結點間隙 2. 多項式插補： 多項式插補的參數間隔長度	*PGA* 1. 曲線插補（ASPLINE、BSPLINE、CSPLINE、BAUTO、BNAT、BTAN、EAUTO、ENAT、ETAN、PW、SD、PL）(頁 208) 2. 多項式插補（POLY、POLYPATH、PO、PL）(頁 224)	n	- -	○ -	- -	○ -
PM	每分鐘	*PG*		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
PO	多項式插補的多項式係數	*PGA* 多項式插補（POLY、POLYPATH、PO、PL） (頁 224)	n	-	-	-	-
POCKET3	銑削循環，矩形腔（任意銑刀）	*BHD/BHF*		●	●	●	●
POCKET4	銑削循環，圓弧腔（任意銑刀）	*BHD/BHF*		●	●	●	●
POLF	LIFTFAST 回退位置	*PG/PGA*	m	●	●	●	●
POLFA	含$AA_ESR_TRIGGER 之單軸的回退起始位置	*PG*	m	●	●	●	●
POLFMASK	啟用回退之軸不含軸間連接	*PG*	m	●	●	●	●
POLFMLIN	啟用回退之軸含軸間線性連接	*PG*	m	●	●	●	●
POLY	多項式插補	*PGA* 多項式插補（POLY、POLYPATH、PO、PL） (頁 224)	m	-	-	-	-
POLYPATH	可為 AXIS 或 VECT 軸群組選擇多項式插補	*PGA* 多項式插補（POLY、POLYPATH、PO、PL） (頁 224)	m	-	-	-	-
PON	沖孔 ON（開啟）	*PGA* 沖孔與切片開啟或關閉（SPOF、SON、PON、SONS、PONS、PDELAYON、PDELAYOF、PUNCHACC） (頁 575)	m	-	-	-	-
PONS	插補循環中之沖孔 ON	*PGA* 沖孔與切片開啟或關閉（SPOF、SON、PON、SONS、PONS、PDELAYON、PDELAYOF、PUNCHACC） (頁 575)	m	-	-	-	-
POS	定位軸	*PG*		●	●	●	●
POSA	跨單節界線之定位軸	*PG*		●	●	●	●
POSM	定位刀庫	*FBW*		●	●	●	●
POSP	在區段中定位（震盪）	*PG*		●	●	●	●
POSRANGE	決定軸目前的插補位置設定點是否落在位於預先定義之參考位置的視窗中	*PGA* 位置位於指定參考範圍內（POSRANGE）(頁 529)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
POT	正方形（算術功能）	*PGA* 運算功能 (頁 58)		●	●	●	●
PR	每轉數	*PG*		●	●	●	●
PREPRO	定義具有準備的副程式	*PGA* 以準備（PREPRO）定義副程式 (頁 151)		●	●	●	●
PRESETON	設定程式設計之軸的實際值	*PGA* 預設偏移量（PRESETON） (頁 265)		●	●	●	●
PRIO	設定中斷處理優先順序之關鍵字	*PGA* 指派並啟動中斷程式（SETINT，PRIO，BLSYNC） (頁 102)		●	●	●	●
PROC	程式中的第一個操作	*PGA* 以路徑規格和參數（PCALL）呼叫子程式 (頁 171)		●	●	●	●
PTP	指向點動作	*PGA* 直角座標PTP移動 (頁 333)	m	●	●	●	●
PTPG0	點對點動作，僅搭配 G0 或 CP	*PGA* TRANSMIT的PTP (頁 337)	m	●	●	●	●
PUNCHACC	切片的移動相關加速度	*PGA* 沖孔與切片開啟或關閉（SPOF、SON、PON、SONS、PONS、PDELAYON、PDELAYOF、PUNCHACC） (頁 575)		-	-	-	-
PUTFTOC	平行加工之刀具微調偏移量	*PGA* 線上刀具偏移（PUTFTOCF、FCTDEF、PUTFTOC、FTOCON、FTOCOF） (頁 358)		●	●	●	●
PUTFTOCF	刀具微調偏移量視以 FCTDEF 定義的平行加工之函數而定	*PGA* 線上刀具偏移（PUTFTOCF、FCTDEF、PUTFTOC、FTOCON、FTOCOF） (頁 358)		●	●	●	●
PW	B 曲線，重量點	*PGA* 曲線插補（ASPLINE、BSPLINE、CSPLINE、BAUTO、BNAT、BTAN、EAUTO、ENAT、ETAN、PW、SD、PL） (頁 208)	n	-	○	-	○
QECLRNOF	象限錯誤補正學習功能 OFF	*PGA* 學習補正特性（QECLRNON、QECLRNOF） (頁 605)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
QECLRNON	象限錯誤補正學習功能 ON	*PGA* 學習補正特性（QECLRNON、QECLRNOF） (頁 605)		●	●	●	●
QU	（輔助）函數輸出	*PG*		●	●	●	●
R...	算數參數亦為可設定之位址辨識碼，含數字副檔名	*PGA* 預定義使用者變數：算術參數（R） (頁 18)		●	●	●	●
RAC	絕對非模態、軸專屬半徑程式設計	*PG*	n	●	●	●	●
RDISABLE	讀入停用	*PGA* 設定讀入停用（RDISABLE） (頁 513)		●	●	●	●
READ	讀入指定檔案中一至數行，並將讀入之資訊儲存到陣列中	*PGA* 在檔案中讀入行（READ） (頁 122)		●	●	●	●
REAL	資料類型： 含正負號浮點數（實數）	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
REDEF	設定機械參數、NC 語言參數及系統變數，指定可看到前數設定之使用者群組	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
RELEASE	鬆開機械軸以便進行軸交換	*PGA* 座標軸替換，主軸替換（RELEASE，GET，GETD） (頁 110)		●	●	●	●
REP	將陣列中具相同數值之所有元素初始化的關鍵字	*PGA* 陣列變數（DEF，SET，REP）之定義與初始化 (頁 42)		●	●	●	●
REPEAT	重覆程式迴圈	*PGA* 重複程式區段（REPEAT，REPEATB，ENDLABEL，P） (頁 83)		●	●	●	●
REPEATB	重覆程式指令列	*PGA* 重複程式區段（REPEAT，REPEATB，ENDLABEL，P） (頁 83)		●	●	●	●
REPOSA	以所有軸進行線性重新定位	*PGA* 重新定位至輪廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (頁 424)	n	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - D - - - - - F	PPU280 / 281 D - - - - - F
REPOSH	使用半圓重新定位	*PGA* 重新定位至輪廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (頁 424)	n	●	● ● ●
REPOSHA	以所有軸進行重新定位；幾何座標軸以半圓表示	*PGA* 重新定位至輪廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (頁 424)	n	●	● ● ●
REPOSL	線性重新定位	*PGA* 重新定位至輪廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (頁 424)	n	●	● ● ●
REPOSQ	在象限中重新定位	*PGA* 重新定位至輪廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (頁 424)	n	●	● ● ●
REPOSQA	以所有軸進行線性重新定位，幾何座標軸以象限表示	*PGA* 重新定位至輪廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (頁 424)	n	●	● ● ●
RESET	重置技術循環	*PGA* 鎖定、解除鎖定、重置（LOCK、UNLOCK、RESET） (頁 557)		●	● ● ●
RESETMON	用於設定點啟用的語言指令	*FBW*		●	● ● ●
RET	副程式結尾	*PGA* 可參數化的副程式傳回跳躍（RET ...）(頁 154)		●	● ● ●
RIC	相對非模態、軸專屬半徑程式設計	*PG* 座標軸替換，主軸替換（RELEASE，GET，GETD） (頁 110)	n	●	● ● ●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
RINDEX	定義輸入字串中字元的索引值	*PGA* 在字串中搜尋字元 / 字串（INDEX、RINDEX、MINDEX、MATCH） (頁 73)		●	●	●	●
RMB	重新定位至單節起點	*PGA* 重新定位至輪廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (頁 424)	m	●	●	●	●
RME	重新定位至單節結尾	*PGA* 重新定位至輪廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (頁 424)	m	●	●	●	●
RMI [4]	重新定位至中斷點	*PGA* 重新定位至輪廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (頁 424)	m	●	●	●	●
RMN	重新定位至最近的路徑點	*PGA* 重新定位至輪廓（REPOSA，REPOSL，REPOSQ，REPOSQA，REPOSH，REPOSHA，DISR，DISPR，RMI，RMB，RME，RMN） (頁 424)	m	●	●	●	●
RND	磨圓輪廓轉角	*PG*	n	●	●	●	●
RNDM	模態倒圓角	*PG*	m	●	●	●	●
ROT	可程式設計的旋轉	*PG*	n	●	●	●	●
ROTS	具立體角的可程式設計框架旋轉	*PG*	n	●	●	●	●
ROUND	小數點位數的四捨五入	*PGA* 運算功能 (頁 58)		●	●	●	●
ROUNDUP	輸入值的向上捨去	*PGA* 無條件進位（ROUNDUP） (頁 132)		●	●	●	●
RP	極半徑	*PG*	m/n	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - D - - - - - F	PPU280 / 281 D - - - - - F		
RPL	平面旋轉	*PG*	n	●	●	●	●
RT	存取框架資料之參數： 旋轉	*PGA* 讀取及變更框架元件（TR、FI、RT、SC、MI） (頁 259)		●	●	●	●
RTLIOF	G0 不含線性插補（單軸插補）	*PG*	m	●	●	●	●
RTLION	用線性插補進行 G0	*PG*	m	●	●	●	●
S	主軸速度或 （在 G4、G96/G961，其他意義）	*PG*	m/n	●	●	●	●
SAVE	當呼叫副程式時，用來儲存資訊的屬性	*PGA* 儲存模態G碼功能（SAVE） (頁 142)		●	●	●	●
SBLOF	停用單一單節	*PGA* 抑制獨立單節執行（SBLOF，SBLON） (頁 144)		●	●	●	●
SBLON	撤回單一單節的抑制	*PGA* 抑制獨立單節執行（SBLOF，SBLON） (頁 144)		●	●	●	●
SC	存取框架資料之參數： 刻度	*PGA* 讀取及變更框架元件（TR、FI、RT、SC、MI） (頁 259)		●	●	●	●
SCALE	可程式設計的比例	*PG*	n	●	●	●	●
SCC	選擇指派至 G96/G961/G962 之橫向軸。 軸辨識碼的形式可為幾何座標軸、通道或機械軸	*PG*		●	●	●	●
SCPARA	程式伺服參數設定	*PGA* 可程式設計的伺服參數集（SCPARA） (頁 248)		●	●	●	●
SD	曲線度數	*PGA* 曲線插補（ASPLINE、BSPLINE、CSPLINE、BAUTO、BNAT、BTAN、EAUTO、ENAT、ETAN、PW、SD、PL） (頁 208)	n	-	○	-	○
SEFORM	步進編輯器中產生供 HMI Advanced 使用之步驟檢視的結構操作	*PGA* 步驟編輯器中的結構指令（SEFORM） (頁 191)		●	●	●	●
SET	將陣列中含列出之數值的所有元素初始化的關鍵字	*PGA* 陣列變數（DEF，SET，REP）之定義與初始化 (頁 42)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
SETAL	設定警報	*PGA* 警報（SETAL） (頁 613)		●	●	●	●
SETDNO	指派刀具（T）的刀刃（CE）之 D 編號	*PGA* 任意指派D數量重新命名D數量（GETDNO，SETDNO） (頁 382)		●	●	●	●
SETINT	定義於 NCK 輸出存在時，應啟動哪一個中斷程式	*PGA* 指派並啟動中斷程式（SETINT，PRIO，BLSYNC） (頁 102)		●	●	●	●
SETM	專用通道中，標記的設定	PGA 程式一致性（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM） (頁 95)		-	-	-	-
SETMS	將機械參數中定義的主動主軸重置			●	●	●	●
SETMS（n）	設主軸 n 為主動主軸	*PG*		●	●	●	●
SETMTH	設定主刀把號碼	*FBW*		●	●	●	●
SETPIECE	為指派至該主軸之所有刀具設定工件編號	*FBW*		●	●	●	●
SETTA	由磨損群組啟動刀具	*FBW*		●	●	●	●
SETTCOR	刀具元件的修改，將一般條件列入考量	*FB1（W1）*		●	●	●	●
SETTIA	由磨損群組停用刀具	*FBW*		●	●	●	●
SF	螺紋切削用起點偏移	*PG*	m	●	●	●	●
SIN	正弦（三角函數）	*PGA* 運算功能 (頁 58)		●	●	●	●
SIRELAY	啟動以 SIRELIN，SIRELOUT，與 SIRELTIME 參數化的安全函數	*FBSI*		●	●	●	●
SIRELIN	函數單節的初始化輸入變數	*FBSI*		●	●	●	●
SIRELOUT	函數單節的初始化輸出變數	*FBSI*		●	●	●	●
SIRELTIME	函數單節的初始化計時器	*FBSI*		●	●	●	●
SLOT1	銑削圖案循環，圓上的溝槽	*BHD/BHF*		●	●	●	●
SLOT2	銑削圖案循環，圓弧溝槽	*BHD/BHF*		●	●	●	●
SOFT	軟路徑加速度	*PG*	m	●	●	●	●
SOFTA	啟動程式設計之軸的軸軟性加速	*PG*		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - D - - - - - F	PPU280 / 281 D - - - - - F
SON	沖壓功能 ON（開啟）	*PGA* 沖孔與切片開啟或關閉（SPOF、SON、PON、SONS、PONS、PDELAYON、PDELAYOF、PUNCHACC）(頁 575)	m	-	-	-	-
SONS	插補循環中之切片 ON	*PGA* 沖孔與切片開啟或關閉（SPOF、SON、PON、SONS、PONS、PDELAYON、PDELAYOF、PUNCHACC）(頁 575)	m	-	-	-	-
SPATH [4]	FGROUP 軸之路徑參照為弧長	*PGA* 可設定路徑參考（SPATH、UPATH）(頁 229)	m	●	●	●	●
SPCOF	將主動主軸或主軸由定位控制切換至速率控制	*PG*	m	●	●	●	●
SPCON	將主動主軸或主軸由速率控制切換至定位控制	*PGA*	m	●	●	●	●
SPI	將主軸號碼轉換為軸識別碼	*PGA* 軸函數（AXNAME、AX、SPI、AXTOSPI、ISAXIS、AXSTRING、MODAXVAL）(頁 589)		●	●	●	●
SPIF1 [4]	快速 NCK 輸入/輸出給沖孔/切片位元組 1 使用	*FB2（N4）*	m	-	-	-	-
SPIF2	快速 NCK 輸入/輸出給沖孔/切片位元組 2 使用	*FB2（N4）*	m	-	-	-	-
SPLINEPATH	定義曲線群組	*PGA* 曲線群組（SPLINEPATH）(頁 219)		-	○	-	○
SPN	各單節路徑區段數量	*PGA* 自動路徑分段 (頁 580)	n	-	-	-	-
SPOF [4]	衝擊 OFF，沖孔，切片 OFF	*PGA* 沖孔與切片開啟或關閉（SPOF、SON、PON、SONS、PONS、PDELAYON、PDELAYOF、PUNCHACC）(頁 575)	m	-	-	-	-
SPOS	主軸位置	*PG*	m	●	●	●	●
SPOSA	跨單節界線主軸位置	*PG*	m	●	●	●	●
SPP	路徑區段長度	*PGA* 自動路徑分段 (頁 580)	m	-	-	-	-

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
SQRT	平方根（算術功能）	*PGA* 運算功能 (頁 58)		●	●	●	●
SR	同步動作的震盪回退路徑	*PG*	n	-	-	-	-
SRA	同步化動作含外部軸向輸入之震盪回退路徑	*PG*	m	-	-	-	-
ST	同步化動作之震盪火花消除時間	*PG*	n	-	-	-	-
STA	同步化動作之震盪火花消除時間軸	*PG*	m	-	-	-	-
START	由目前的程式同時在數個通道中啟動選取之程式	*PGA* 程式一致性（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM） (頁 95)		-	-	-	-
STARTFIFO [4]	執行；同時填充前置處理記憶體	*PGA* 具有前置處理記憶體（STOPFIFO，STARTFIFO，FIFOCTRL，STOPRE）的程式順序 (頁 414)	m	●	●	●	●
STAT	接點位置	*PGA* 直角座標PTP移動 (頁 333)	n	●	●	●	●
STOPFIFO	停止加工；填充前置處理記憶體直到偵測到STARTFIFO、前置處理記憶體完整或程式結尾	*PGA* 具有前置處理記憶體（STOPFIFO，STARTFIFO，FIFOCTRL，STOPRE）的程式順序 (頁 414)	m	●	●	●	●
STOPRE	前置處理停止，直到所有預置單節已在主程式中執行	*PGA* 具有前置處理記憶體（STOPFIFO，STARTFIFO，FIFOCTRL，STOPRE）的程式順序 (頁 414)		●	●	●	●
STOPREOF	撤回前置處理停止	*PGA* 取消前置處理停止（STOPREOF） (頁 513)		●	●	●	●
STRING	資料類型： 字元字串	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
STRINGFELD	從程式字串欄位選擇單一字元	*PGA* 選擇單字元（STRINGVAR，STRINGFELD） (頁 75)		●	●	●	●
STRINGIS	檢查 NC 語言及 NC 循環名稱、使用者變數、巨集及屬於本旨令之標籤名稱目前的範圍，以確定其係存在、有效、已定義或啟用	*PGA* 檢查顯示NC語言之範圍（STRINGIS） (頁 598)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
STRINGVAR	從程式字串選擇單一字元	*PGA* 選擇單字元（STRINGVAR，STRINGFELD） (頁 75)		-	-	-	-
STRLEN	定義字串長度	*PGA* 決定字串長度（STRLEN） (頁 73)		●	●	●	●
SUBSTR	定義輸入字串中字元的索引值	*PGA* 選擇子字串（SUBSTR） (頁 75)		●	●	●	●
SUPA	抑制目前的工作偏移，包括程式設計之偏移、系統框架、手輪偏移（DRF）、外部工作偏移及覆蓋動作	*PG*	n	●	●	●	●
SVC	刀具切削率	*PG*	m	●	●	●	●
SYNFCT	評估將多項式作為動作同步動作中條件的函數	*PGA* 同步函數（SYNFCT） (頁 518)		●	●	●	●
SYNR	同步讀取變數，例如，在執行時	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
SYNRW	同步讀取和寫入變數，例如，在執行時	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
SYNW	同步寫入變數，例如，在執行時	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
T	呼叫刀具（僅於有在機械參數中指定時改變；否則須使用 M6 指令）	*PG*		●	●	●	●
TAN	正切（三角函數）	*PGA* 運算功能 (頁 58)		●	●	●	●
TANG	定義軸群組切線修正	*PGA* 切線控制（TANG，TANGON，TANGOF，TLIFT，TANGDEL） (頁 401)		-	-	-	-
TANGDEL	刪除軸群組切線修正之定義	*PGA* 切線控制（TANG，TANGON，TANGOF，TLIFT，TANGDEL） (頁 401)		-	-	-	-
TANGOF	切線修正 OFF	*PGA* 切線控制（TANG，TANGON，TANGOF，TLIFT，TANGDEL） (頁 401)		-	-	-	-

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
TANGON	切線修正 ON	*PGA* 切線控制（TANG，TANGON，TANGOF，TLIFT，TANGDEL） (頁 401)		-	-	-	-
TCA	刀具選擇/換刀，不考慮刀具狀態	*FBW*		●	●	●	●
TCARR	請求刀把（編號“m”）	*PGA* 用於可定向刀把的刀具長度補正（TCARR，TCOABS，TCOFR，TCOFRX，TCOFRY，TCOFRZ） (頁 389)		-	●	-	●
TCI	從緩衝器，將刀具載入至刀庫	*FBW*		●	●	●	●
TCOABS [4]	從目前刀把的方向決定刀長元件	*PGA* 用於可定向刀把的刀具長度補正（TCARR，TCOABS，TCOFR，TCOFRX，TCOFRY，TCOFRZ） (頁 389)	m	-	●	-	●
TCOFR	從主動框架的方向決定刀長元件	*PGA* 用於可定向刀把的刀具長度補正（TCARR，TCOABS，TCOFR，TCOFRX，TCOFRY，TCOFRZ） (頁 389)	m	-	●	-	●
TCOFRX	選擇刀具及在 X 方向之刀具點來判定啟用中框架的刀具方向	*PGA* 用於可定向刀把的刀具長度補正（TCARR，TCOABS，TCOFR，TCOFRX，TCOFRY，TCOFRZ） (頁 389)	m	-	●	-	●
TCOFRY	選擇刀具及在 Y 方向之刀具點來判定啟用中框架的刀具方向	*PGA* 用於可定向刀把的刀具長度補正（TCARR，TCOABS，TCOFR，TCOFRX，TCOFRY，TCOFRZ） (頁 389)	m	-	●	-	●
TCOFRZ	選擇刀具及在 Z 方向之刀具點來判定啟用中框架的刀具方向	*PGA* 用於可定向刀把的刀具長度補正（TCARR，TCOABS，TCOFR，TCOFRX，TCOFRY，TCOFRZ） (頁 389)	m	-	●	-	●
THETA	旋轉角度	*PGA* 刀具方向的旋轉（ORIROTA、ORIROTR、ORIROTT、ORIROTC、THETA） (頁 305)	n	●	●	●	●
TILT	傾斜角度	*PGA* 程式設計刀具方向（A...、B...、C...、LEAD、TILT） (頁 286)	m	●	●	●	●
TLIFT	在切線控制中，在輪廓轉角處插入中間單節	*PGA* 切線控制（TANG，TANGON，TANGOF，TLIFT，TANGDEL） (頁 401)		-	-	-	-

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
TMOF	取消選擇刀具監控	*PGA* 工件程式中研磨專屬刀具監控（TMON、TMOF） (頁 587)		●	●	●	●
TMON	啟用刀具監控	*PGA* 工件程式中研磨專屬刀具監控（TMON、TMOF） (頁 587)		●	●	●	●
TO	指派 FOR 計數迴圈內之結束值	*PGA* 計數迴圈（FOR ... TO ...，ENDFOR） (頁 91)		●	●	●	●
TOFF	刀長元件之方向上的刀長偏移，與索引值中指定之幾何座標軸平行	*PG*	m	●	●	●	●
TOFFL	刀長元件之方向上的刀長偏移 L1、L2 或 L3	*PG*	m	●	●	●	●
TOFFOF	停用線上刀具偏移	*PGA* 線上刀長補正（TLC）（TOFFON，TOFFOF） (頁 392)		●	●	●	●
TOFFON	啟用線上刀具長度偏移	*PGA* 線上刀長補正（TLC）（TOFFON，TOFFOF） (頁 392)		●	●	●	●
TOFFR	刀具半徑偏移量	*PG*	m	●	●	●	●
TOFRAME	透過旋轉框架的方式將工件座標系統的 Z 軸調整為與工件方向平行	*PG*	m	●	●	●	●
TOFRAMEX	透過旋轉框架的方式將工件座標系統的 X 軸調整為與工件方向平行	*PG*	m	●	●	●	●
TOFRAMEY	透過旋轉框架的方式將工件座標系統的 Y 軸調整為與工件方向平行	*PG*	m	●	●	●	●
TOFRAMEZ	與 TOFRAME 相同	*PG*	m	●	●	●	●
TOLOWER	將字串字元轉換成小寫	*PGA* 轉換為大小寫字母（TOLOWER、TOUPPER） (頁 72)		●	●	●	●
TOOLENV	儲存目前狀態，這對於評估儲存在記憶體中的刀具資料，非常重要	*FB1（W1）*		●	●	●	●
TOROT	透過旋轉框架的方式將工件座標系統的 Z 軸調整為與工件方向平行	*PG*	m	●	●	●	●
TOROTOF	框架在刀具方向上之旋轉 OFF	*PG*	m	●	●	●	●
TOROTX	透過旋轉框架的方式將工件座標系統的 X 軸調整為與工件方向平行	*PG*	m	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
TOROTY	透過旋轉框架的方式將工件座標系統的 Y 軸調整為與工件方向平行	*PG*	m	●	●	●	●
TOROTZ	與 TOROT 相同	*PG*	m	●	●	●	●
TOUPPER	將字串字元轉換成大寫	*PGA* 轉換為大小寫字母（TOLOWER、TOUPPER） (頁 72)		●	●	●	●
TOWBCS	基本座標系統（BCS）中之磨損值	*PGA* 生效加工操作的座標系統（TOWSTD、TOWMCS、TOWWCS、TOWBCS、TOWTCS、TOWKCS） (頁 354)	m	-	●	-	●
TOWKCS	刀頭座標系統之磨損值，供動力學轉換之用（與機械作標系統不同點在於刀具旋轉）	*PGA* 生效加工操作的座標系統（TOWSTD、TOWMCS、TOWWCS、TOWBCS、TOWTCS、TOWKCS） (頁 354)	m	-	●	-	●
TOWMCS	機械座標系統中之磨耗值	*PGA* 生效加工操作的座標系統（TOWSTD、TOWMCS、TOWWCS、TOWBCS、TOWTCS、TOWKCS） (頁 354)	m	-	●	-	●
TOWSTD	刀長偏移的初始設定值	*PGA* 生效加工操作的座標系統（TOWSTD、TOWMCS、TOWWCS、TOWBCS、TOWTCS、TOWKCS） (頁 354)	m	-	●	-	●
TOWTCS	刀具座標系統中之磨損值（位於刀把之刀把參考點 T）	*PGA* 生效加工操作的座標系統（TOWSTD、TOWMCS、TOWWCS、TOWBCS、TOWTCS、TOWKCS） (頁 354)	m	-	●	-	●
TOWWCS	工件座標系統中之磨耗值	*PGA* 生效加工操作的座標系統（TOWSTD、TOWMCS、TOWWCS、TOWBCS、TOWTCS、TOWKCS） (頁 354)	m	-	●	-	●
TR	框架變數的偏移元件	*PGA* 讀取及變更框架元件（TR、FI、RT、SC、MI） (頁 259)		●	●	●	●
TRAANG	轉換傾斜軸	*PGA* 傾斜軸（TRAANG） (頁 328)		-	-	○	-

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
TRACON	Cascaded transformation	*PGA* 連鎖轉換（TRACON、TRAFOOF） (頁 343)		-	-	○	-
TRACYL	圓柱： 柱面轉換	*PGA* 圓柱表面轉換（TRACYL） (頁 321)		○	○	○	○
TRAFOOF	在通道中停用生效轉換	*PGA* 連鎖轉換（TRACON、TRAFOOF） (頁 343)		●	●	●	●
TRAILOF	非同步耦合動作 OFF（關閉）	*PGA* 耦合動作（TRAILON，TRAILOF） (頁 437)		●	●	●	●
TRAILON	非同步耦合動作 ON（開啟）	*PGA* 耦合動作（TRAILON，TRAILOF） (頁 437)		●	●	●	●
TRANS	可程式設計的偏移	*PG*	n	●	●	●	●
TRANSMIT	極點轉換（平面加工）	*PGA* 車削零件上銑削（TRANSMIT） (頁 318)		○	○	○	○
TRAORI	4-軸，5-軸轉換，一般轉換	*PGA* 三、四及五軸轉換（TRAORI） (頁 283)		-	●	-	●
TRUE	邏輯常數： 真	*PGA* 使用者變數的定義（DEF） (頁 22)		●	●	●	●
TRUNC	小數點位數的四捨五入	*PGA* 比較錯誤（TRUNC）的精確性修正 (頁 62)		●	●	●	●
TU	軸角度	*PGA* 直角座標PTP移動 (頁 333)	n	●	●	●	●
TURN	螺旋旋轉數	*PG*	n	●	●	●	●
ULI	變數中的上臨界值	*PGA* 屬性：臨界值（LLI、ULI） (頁 33)		●	●	●	●
UNLOCK	利用 ID 啟用同步化動作（繼續技術循環）	*PGA* 鎖定、解除鎖定、重置（LOCK、UNLOCK、RESET） (頁 557)		●	●	●	●
UNTIL	終止 REPEAT 迴圈之條件	*PGA* 在迴圈起點處，附有條件的程式迴圈（WHILE，ENDWHILE） (頁 93)		●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
UPATH	FGROUP 軸之路徑參照為曲線參數	*PGA* 可設定路徑參考（SPATH、UPATH） (頁 229)	m	●	●	●	●
VAR	關鍵字： 參數傳輸類型：	*PGA* 以參數傳輸（EXTERN）進行副程式呼叫 (頁 162)		●	●	●	●
VELOLIM	最大軸向速率之過衝的減少	*PGA* 比例速率修正（VELOLIM） (頁 433)	m	●	●	●	●
VELOLIMA	最大軸向速率之減少或過衝	*PG*	m	●	●	●	●
WAITC	等待有待滿足的耦合單節更改條件，供軸/主軸所用	*PGA* 程式一致性（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM） (頁 95)		-	-	○	-
WAITE	等待另一通道中的程式結束	*PGA* 程式一致性（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM） (頁 95)		-	-	-	-
WAITM	等待指定通道中之標記；以精確停止終止先前單節	*PGA* 程式一致性（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM） (頁 95)		-	-	-	-
WAITMC	等待指定通道中之標記；僅在另一通道尚未抵達標記時進行精確停止。	*PGA* 程式一致性（INIT，START，WAITM，WAITMC，WAITE，SETM，CLEARM） (頁 95)		-	-	-	-
WAITP	等到移動結束	*PG*		●	●	●	●
WAITS	等待有待達到的主軸位置	*PG*		●	●	●	●
WALCS0	工件座標系統工作區限制取消選擇	*PG*	m	●	●	●	●
WALCS1	工件座標系統工作區限制群組 1 生效	*PG*	m	●	●	●	●
WALCS2	工件座標系統工作區限制群組 2 生效	*PG*	m	●	●	●	●
WALCS3	工件座標系統工作區限制群組 3 生效	*PG*	m	●	●	●	●
WALCS4	工件座標系統工作區限制群組 4 生效	*PG*	m	●	●	●	●
WALCS5	工件座標系統工作區限制群組 5 生效	*PG*	m	●	●	●	●

操作	含義	說明，請參考 [1]	W [2]	828D [3] PPU260 / 261 - - - PPU280 / 281 D - - - - - F - - - - - D - - - - - F			
WALCS6	工件座標系統工作區限制群組 6 生效	*PG*	m	●	●	●	●
WALCS7	工件座標系統工作區限制群組 7 生效	*PG*	m	●	●	●	●
WALCS8	工件座標系統工作區限制群組 8 生效	*PG*	m	●	●	●	●
WALCS9	工件座標系統工作區限制群組 9 生效	*PG*	m	●	●	●	●
WALCS10	工件座標系統工作區限制群組 10 生效	*PG*	m	●	●	●	●
WALIMOF	BCS 工作區限制 OFF（關閉）	*PG*	m	●	●	●	●
WALIMON [4]	BCS 工作區限制 ON（開啟）	*PG*	m	●	●	●	●
WHEN	條件滿足之時，就可以循環執行動作。	*PGA* 週期性檢查條件（WHEN、WHENEVER、FROM、EVERY） (頁 490)		●	●	●	●
WHENEVER	條件滿足之時，就可以執行動作一次。	*PGA* 週期性檢查條件（WHEN、WHENEVER、FROM、EVERY） (頁 490)		●	●	●	●
WHILE	WHILE 程式迴圈之起點	*PGA* 在迴圈起點處，附有條件的程式迴圈（WHILE，ENDWHILE） (頁 93)		●	●	●	●
WRITE	將文字寫入檔案系統 將單節附加至指定檔案之結束	*PGA* 寫入檔案（WRITE） (頁 117)		●	●	●	●
WRTPR	延遲不具有中斷連續路徑模式的加工工作			●	●	●	●
X	軸名稱	*PG*	m/n	●	●	●	●
XOR	邏輯互斥 OR	*PGA* 比較與邏輯運算 (頁 60)		●	●	●	●
Y	軸名稱	*PG*	m/n	●	●	●	●
Z	軸名稱	*PG*	m/n	●	●	●	●

附錄

A

A.1 縮寫表

A	輸出
AS	自動化系統
ASCII	美國訊息交換標準代碼
ASIC	應用程式專屬積體電路：使用者開關電路
ASUB	非同步子程式
AuxF	輔助功能
AV	工作計畫
BA	操作模式
BB	準備執行
BCD	二進位編碼小數：以二進制代碼編碼之十進位數字
BCS	基準座標系統
BIN	二進制碼檔案（**Bin**ary 檔案）
BIOS	基本輸出輸入系統
BOT	啟動備檔：SIMODRIVE 611 數位之啟動備檔
BP	基本程式
C Bus	通訊匯流排
CAD	電腦輔助設計
CAM	電腦輔助製造
CNC	電腦數值控制電腦數值控制
COM	通訊
COR	座標旋轉
CP	通訊處理器
CPU	中央處理器：中央處理器
CR	歸位
CRC	銑刀半徑補正
CRT	陰極射線管映像管
CSB	中央機板：PLC 模組
CSF	功能計畫（PLC 程式設計方法）
CTS	清除以便傳送自序列資料介面發出的訊號
CUTOM	銑刀半徑補正刀具半徑補正
DAC	數位到類比的轉換器
DB	PLC 內資料單節
DBB	PLC 內資料單節位元組
DBW	PLC 內資料單節字
DBX	PLC 內資料單節位元
DC	直接控制：旋轉軸在單次轉動內經最短路徑移往絕對位置的移動

DCD	資料媒介偵測
DDE	動態資料交換
DIN	德國工業標準
DIO	資料輸入 / 輸出：資料傳輸顯示
DIR	目錄：目錄
DLL	動態聯結函式庫
DOE	資料傳輸設備
DOS	磁碟作業系統
DPM	雙通訊埠記憶體
DPR	雙通訊埠 RAM
DRAM	動態隨機存取記憶體
DRF	差動解算器功能：差動解算器功能（DRF）
DRY	空跑：空跑進給率
DSB	解碼單一單節解碼單一單節
DTE	終端資料設備
DW	資料字組
E	輸入
EIA 編碼	特殊的打孔帶編碼，每字元的孔數皆為奇數
ENC	編碼器：實際值編碼器
EPROM	抹除式唯讀記憶體
FB	函數單節
FBS	薄型螢幕
FC	函數呼叫：PLC 內函數單節
FDB	產品資料庫
FDD	軟碟
FDD	進給驅動
FEPROM	快閃抹除式唯讀記憶體：讀寫記憶體
FIFO	先進先出：記憶體，不指定位址運作，且以相同的儲存順序讀入資料。
FIPO	微調插補器
FM	函式模組
FPU	浮點數元件浮點數元件
FRA	框架區塊
FRAME	資料紀錄（框架）
FST	進給停止：進給停止
GUD	全域使用者資料全域使用者資料
HD	Hard Disk（硬碟）
HEX	十六進位數之縮寫
HHU	手持裝置
HMI	人機介面
HMI	人機介面 SINUMERIK 供操作、程式設計及模擬使用之操作員功能。
HMS	高解析度量測系統
HW	硬體
I/O	輸入 / 輸出

I/R	SIMODRIVE 611 數位之進給 / 再生式反饋元件（電源）
IBN	啟動
IF	磁碟模組脈衝啟用
IK （GD）	未指定通訊（全域資料）
IKA	插補式補正插補補正
IM	介面模組互連模組
IMR	介面模組接收：用於接收資料之內部連結模組
IMS	介面模組傳送：用於傳送資料之內部連結模組
INC	增量：增量
INI	初始化資料初始化資料
IPO	插補器
IS	介面訊號
ISA	工業標準架構
ISO	國際標準組織
ISO 編碼	特殊的打孔帶編碼，每字元的孔數皆為偶數
JOG（寸動進給）	寸動進給：設定模式
K1 ..K4	通道 1 至通道 4
K_{UE}	速率比率
K_v	伺服增益因子
LAD	階梯圖（PLC 程式設計方法）
LCD	液晶顯示器：液晶顯示器
LEC	導螺桿錯誤補正
LED	發光二極管：發光二極體
LF	LINE FEED
LR	定位控制器
LUD	區域使用者資料
MB	百萬位元組
MC	量測電路
MCP	機機械座標械控制面板
MCS	機械座標系統
MD	機械參數
MDI	手動資料自動執行：手動輸入
MLFB	機器可讀之產品指定
MPF	主程式檔案：NC 工件程式（主程式）
MPI	多重連接埠介面多重連接埠介面
MS	微軟（軟體製造商）
MSD	主軸磁碟
NC	數值控制：數值控制
NCK	數值控制核心：數值控制核心
NCU	數值控制裝置：NCK 之硬體單元
NRK	NCK 操作系統之名稱
NURBS	非均勻有理 B－曲線
OB	PLC 內組織單節

OEM	原設備製造商
OP	操作面板前端
OP	操作員面板：操作設定
OPI	操作員面板介面
OPI	操作者面板介面：連接到操作者面板的介面
OPT	選項：選項
OSI	開放系統內部連結：電腦通訊標準
PC	個人電腦
PCIN	與控制項目進行資料交換時 SW 的名稱
PCMCIA	國際個人電腦記憶卡協會：隨插即用記憶卡之標準
PCU	PC 元件：PC 主機（電腦元件）
PG	程式設計裝置
PLC	可程式邏輯控制器：介面控制
PLC	可程式設計的邏輯控制器
PMS	位置量測系統
POS	定位
P 匯流排	周邊匯流排
RAM	隨機存取記憶體：可讀取和寫入的程式記憶體
REF	參考點逼近功能
REPOS	重新定位功能
RISC	減少指令集電腦：小型指令集的處理器類型，具高速處理指令能力
ROV	快速手動倍率：輸入修正
RPA	啟用 R 參數：NCK 上針對 R 參數編號的 記憶體區
RPY	Roll Pitch Yaw 三方位：座標系統之旋轉類型
RS-232-C	序列介面（DTE 及 DCE 間交換線路之定義）
RTS	請求傳送：RTS，序列資料介面的控制訊號
SBL	單一單節：單一單節
SD	設定參數
SDB	系統資料單節
SEA	設定資料生效：設定資料之辨識碼（檔案類型）
SFB	系統函數單節
SFC	系統函數呼叫
SK	軟鍵
SKP	SKiP：略過單節
SM	步進馬達
SPF	副程式檔案：子程式
SR	子程式
SRAM	靜態 RAM（非揮發性）
SSI	序列同步介面序列同步介面
STL	敘述清單
SW	軟體
SYF	系統檔案系統檔案
T	刀具

TC	刀具換用
TEA	啟用測試資料：機械參數用的識別碼
TLC	刀長補正
TNRC	刀鼻半徑補正
TO	刀具偏移：刀具偏移
TO	刀具偏移
TOA	刀具偏移已啟用刀具偏移之辨識碼（檔案類型）
TRANSMIT	將銑削轉換為車削在車削機台上為進行銑削操作的座標轉換
TRC	刀具半徑補正
UFR	使用者框架：零點偏移
UI	使用者介面
WCS	工件座標系統
WOP	工作場導向式程式設計
WPD	工件目錄：工件目錄
ZO	零點偏移
ZOA	啟用零點偏移：零點偏移之辨識碼（檔案類型）
µC	巨集控制器
模式群組	模式群組
錯誤	印表機之錯誤

A.2 本文件之意見反應

本文件將持續在品質及使用方便上進行改善。請利用電子郵件或傳真方式提供寶貴意見與建議，以協助我們進行這項改善工作：

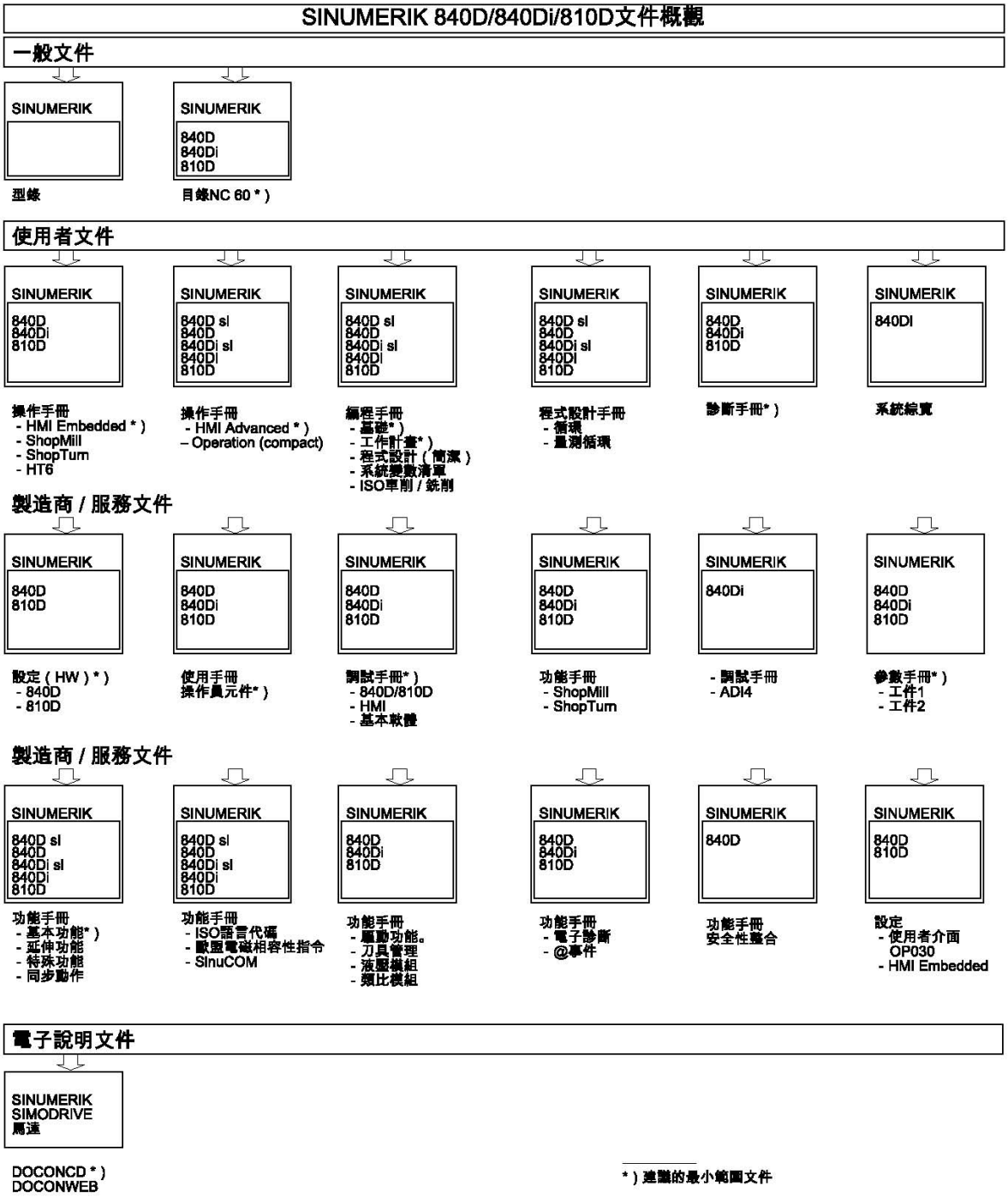
電子郵件： mailto:docu.motioncontrol@siemens.com

傳真： +49 9131 - 98 2176
請使用本頁背面傳真表單。

收件人	寄件人
SIEMENS AG I DT MC MS1 P.O. Box 3180 D-91050 Erlangen / Germany	名稱：
	公司 / 部門地址
	地址：
	郵遞區號： 城市：
	電話： /
傳真： +49 9131 - 98 2176（文件）	傳真： /

建議及（或）修正

A.3 綜覽


SINUMERIK 840D/840Di/810D文件概觀
一般文件
SINUMERIK
型錄
SINUMERIK
840D
840Di
810D
目錄NC 60 *)
使用者文件
SINUMERIK
840D
840Di
810D
操作手冊
- HMI Embedded *)
- ShopMill
- ShopTurn
- HT6
SINUMERIK
840D sl
840D
840Di sl
840Di
810D
操作手冊
- HMI Advanced *)
– Operation (compact)
SINUMERIK
840D sl
840D
840Di sl
840Di
810D
編程手冊
- 基礎*)
- 工作計畫*)
- 程式設計 (簡潔)
- 系統變數清單
- ISO車削 / 銑削
SINUMERIK
840D sl
840D
840Di sl
840Di
810D
程式設計手冊
- 循環
- 量測循環
SINUMERIK
840D
840Di
810D
診斷手冊*)
SINUMERIK
840DI
系統綜覽
製造商 / 服務文件
SINUMERIK
840D
810D
設定 (HW) *)
- 840D
- 810D
SINUMERIK
840D
840Di
810D
使用手冊
操作員元件*)
SINUMERIK
840D
840Di
810D
調試手冊*)
- 840D/810D
- HMI
- 基本軟體
SINUMERIK
840D
840Di
810D
功能手冊
- ShopMill
- ShopTurn
SINUMERIK
840Di
- 調試手冊
- ADI4
SINUMERIK
840D
840Di
810D
參數手冊*)
- 工件1
- 工件2
製造商 / 服務文件
SINUMERIK
840D sl
840D
840Di sl
840Di
810D
功能手冊
- 基本功能*)
- 延伸功能
- 特殊功能
- 同步動作
SINUMERIK
840D sl
840D
840Di sl
840Di
810D
功能手冊
- ISO語言代碼
- 歐盟電磁相容性指令
- SinuCOM
SINUMERIK
840D
840Di
810D
功能手冊
- 驅動功能。
- 刀具管理
- 液壓模組
- 類比模組
SINUMERIK
840D
840Di
810D
功能手冊
- 電子診斷
- @事件
SINUMERIK
840D
功能手冊
安全性整合
SINUMERIK
840D
810D
設定
- 使用者介面
OP030
- HMI Embedded
電子說明文件
SINUMERIK
SIMODRIVE
馬達
DOCONCD *)
DOCONWEB
*) 建議的最小範圍文件

字彙表

CNC

請參閱→NC

COM

NC 的元件，用於通訊的完成和協調。

CPU

中央處理器，請參閱 → PLC

C 曲線

C 曲線是最為熟知及廣為使用的曲線。 插補點上的變化以切線及曲線持續進行。 使用三元多項式。

C 軸

刀具主軸定義受控制旋轉和定位動作所環繞的軸。

DRF

差動解算器功能： 在自動模式中搭配電動手輪使用而產生增量零點偏移的 NC 功能

HIGHSTEP

AS300/AS400 系統 PLC 的→程式設計選項摘要。

JOG（寸動進給）

控制操作模式（設定模式） 可在寸動進給模式中設定機台。 在寸動進給模式中，可使用方向鍵移動個別的軸和主軸。 寸動進給模式中的其他功能包括： → 原點復歸→重新定位與→預設（設定實際值）。

KV

伺服器增益係數，一種在控制迴路中的控制變數。

MDA（手動輸入）

控制操作模式： 手動資料自動執行。 可在 MDA（手動輸入）模式中，個別程式單節或未參考主程式或子程式的單節順序可以在啟動 NC 起始鍵之後立刻輸入並執行。

NC

數值控制： 數值控制（NC）包含加工刀具控制的所有元件： → NCK→PLC，HMI→COM。

說明

更能正確描述 SINUMERIK 840D sl 控制的詞彙應為： 電腦數值控制

NCK

數值控制核心： 執行工件程式並基本協調加工刀具動作操作的 NC 元件。

NRK

數值自動核心（NCK→的操作系統）

NURBS

發生在控制中的動作控制和路徑插補是在 NURBS（**N**on **U**niform **R**ational **B**-**S**plines）基礎上執行的。 因此，SINUMERIK 840D 所有插補的控制中便有一致的程序。

PCIN 資料傳輸程式

PCIN 是透過一個連續介面用於發送及接收 CNC 使用者資料的輔助程式（例如：工件程式、刀具偏移量等等）。 PCIN 程式可於標準工業電腦上以 MS-DOS 執行。

PLC

Programmable **L**ogic **C**ontrol（可程式設計的邏輯控制系統）： →可程式設計的邏輯控制器。
NC→元件： 可供處理加工刀具控制邏輯的可程式設計控制器。

PLC 程式記憶體

SINUMERIK 840D： PLC 使用者程式、使用者資料及基礎 PLC 程式一併儲存於 PLC 使用者記憶體。

PLC 程式設計

PLC 是以 **STEP 7** 軟體進行程式設計。 STEP 7 程式設計軟體以 **WINDOWS** 標準操作系統為基礎，並包含創新加強的 STEP 5 程式設計功能。

Programmable Logic Control（可程式設計邏輯控制）

可程式設計邏輯控制器（PLC）為電子式控制系統，其功能被儲存為控制元件中的程式。這代表裝置的配置和佈線與控制的功能無關。 可程式設計邏輯控制器具有與電腦相同的結構；包括一個含記憶體的 CPU（中央模組）、輸入 / 輸出模組與內部匯流排系統。週邊設備和程式設計語言符合控制技術的需求。

R 參數

可在程式中由工件程式的程式設計師為任何目的設定或查詢的算數參數。

SRT

轉換率

TOA 元件

每個→TOA 區可以有超過一個 TOA 元件。 可能的 TOA 元件數量限於啟用中通道的最大數量。 TOA 元件包括一個刀具資料單節和一個刀庫資料單節。 此外，TOA 元件也可包含一個刀把資料單節（選配）。

TOA 區

TOA 區包含所有的刀具和刀庫資料。 預設中，當達到資料時，此區與→通道區會重疊。 但是，加工資料可用來指定多通道分享→一個 TOA 元件，因此，通道才能取得一般刀具管理資料。

WinSCP

WinSCP 為 Windows 用來傳送檔案之任意開放原始碼程式。

Workpiece（工件）

要由加工刀具製作或加工的工件。

中斷常式

中斷常式是特殊的→子程式，可在加工處理中由事件（外部訊號）啟動。 加工中的工件程式單節已中斷，並自動儲存中斷點的軸位置。

中繼單節

含選定→刀具偏移量 (G41 / G42) 的動作，可使用限數量的中繼單節（偏移量平面中不含軸動作的單節）中斷，藉此仍可正確補正刀具偏移量。控制系統事先讀取的中繼單節許可數量，可在系統參數中設定。

主導軸

先導軸→是操作員及程式設計師眼中的龍門軸，因此可如標準 NC 軸般受影響。

主程式

→由數字或識別碼命名的工件程式，可用來呼叫其他主程式、子程式或→循環。

主要單節

前綴為":"的單節 含有啟動執行工件程式所需的所有參數。

主軸準停

於特定角度位置停止工件主軸，例如為了要在一特定位置上執行其他的加工。

使用者程式

S7-300 自動裝置系統的使用者程式是使用程式設計語言 STEP 7 而建立的。使用者程式具有模組配置並含有個別單節。

基本單節類型為：

- 代碼單節

  這些單節含有 STEP 7 指令。

- 資料單節

  這些單節含 STEP 7 程式的常數和變數。

使用者自訂變數

使用者可在→工件程式或資料單節（全域使用者資料）中為任何目的登錄自己的變數。 定義包含資料類型規格和變數名稱。 請參閱→系統變數。

使用者記憶體

所有的程式和資料，如工件程式、子程式、註解、刀具偏移量和零偏移量 / 框架，以及通道和程式使用者資料，皆可儲存在共用的 CNC 使用者記憶體中。

保護區

→ 工作區中刀具刀尖不可通過的 3D 區域。

備用電池

備用電池可確保→CPU 中的使用者程式已儲存→，因此不會受停電影響，且指定的資料區和位元記憶體、計時器和計數器能夠妥善保存。

傳輸率

資料傳輸率（位元 / 秒）。

傾斜表面加工

不在機台座標平面上的工件錶面，可利用“傾斜表面加工”功能輕易地執行鑽孔銑削操作。

公制測量系統

單位的標準化系統： 以長度而言，如 mm（毫米）、m（公尺）。

具震動限制之加速度

為了最佳化機台的加速度回應，並同時同步保護機械元件，可在加工程式中切換瞬間加速與持續（無晃動）加速。

刀具

用來實施加工的加工刀具上的有效工件（如車削刀具、銑削、刀具、鑽頭、雷射光束等）。

刀具偏移

在計算路徑時考慮刀具尺寸。

刀具半徑補正

為了直接將→所需的工件輪廓程式設計，控制必須將基部重合路徑移動到程式設計輪廓，並考慮到正在使用刀具的半徑（G41/G42）。

刀鼻半徑補正

輪廓程式設計假設已指向刀具。由於這不是實際情況，因此必須將所使用的刀具曲率半徑傳送到控制，以便將其納入考量。曲率中心必須和輪廓維持等距，以曲率半徑偏移。

刻度

→ 框架元件，使用軸專屬的比例修改。

加工通道

藉由平行連續動作，通道結構可用來縮短停滯次數，例如加工時同時移動負載架。 在此，CNC 通道必須視為分離於解碼、單節準備和插補的 CNC 控制系統。

動態前饋控制

→ 使用動態、加速度相關的前饋控制，可實際消除由於下列錯誤所造成的輪廓不正確性。即使使用高→路徑速度，仍能產生極佳的加工正確性。可透過→工件程式，根據軸專屬基準選擇或取消選擇前饋控制。

原製造商

針對想要在控制中建立自己的操作者介面或整合處理導向功能的加工製造商，已提供 SINUMERIK 840D sl 個別解決方案（OEM 應用程式）的範圍。

參考點

機械軸參考測量系統的加工刀具位置。

反時限進給率

有了 SINUMERIK 840D，單節的路徑移動所需時間可程式設計為軸動作，而不是進給速率（G93）。

可程式設計工作區限制

刀具到某一空間的動作空間限制是以程式設計限制而定義。

可程式設計框架

可程式設計→當正在執行工件程式時，框架可供動態定義新的座標系統輸出點。使用新框架和參考現有起點的附加定義，區分絕對定義。

同步

工件程式中，在某些加工點不同通道內順序協調的敘述。

同步動作

1. 輔助功能輸出

   工件加工時，技術性功能（→輔助功能）可由 CNC 程式輸出至 PLC。 例如，這些輔助功能用來控制加工刀具的附加設備，如套管、抓取器、固定夾頭等等。

2. 快速輔助功能輸出

   針對有時間性的切換功能，輔助功能的確認時間可以縮到最短，→也可避免加工程序中不必要的暫停點。

同步化軸

同步軸是→龍門軸，其設定位置不斷來自主導軸的動作→，因此會與主導軸同步移動。 從程式設計師和操作員眼中來看，同步軸"並不存在"。

同步軸

同步軸移動其路徑的時間，和幾何軸移動路徑的時間相同。

啟動

開機後載入系統程式。

單節

"Block"（單節）是為任何建立和處理程式需要的檔案提供的詞語。

單節搜尋

為了除錯目的或程式中止之後，可使用"Block search"（單節搜尋）功能選擇工件程式中要啟動或繼續的任何位置。

固定機台原點

加工刀具唯一定義的點，例如加工參考點。

圓形插補

刀具在輪廓上的指定點之間以已知進給率呈圓弧狀移動，藉以加工工件。

地址：

位址是用於特定運算元或運算元範圍的識別碼，如輸入、輸出等。

基準座標系統

藉由轉換而對應到機台座標系統上的直角座標系統。

程式設計師使用→工件程式中基準座標系統的軸名。 存在的基準座標系統平行於→機台座標系統，如果不是→則會啟用轉換。 兩個座標系統的差異在於→軸識別碼中。

基準軸

其設定點或實際值位置構成補正值計算基準的軸。

增量

根據增量數指定移動路徑距離。增量數可儲存為→設定資料，或利用合適標示按鍵加以選擇（即 10、100、1000、10000）。

增量進給

並增量尺寸： 軸移動的目的地是由待涵蓋的距離以及參考已到達點的方向所定義。 請參閱→絕對尺寸。

外部的零點偏移

由 → PLC 指定。

多項式插補

多項式插補可以產生許多種的弧線，諸如**直線、拋物線、指數函數**（SINUMERIK 840D）。

子單節

前面加上 “N”的單節含順序資訊（如位置資料）。

子程式

→工件程式的敘述順序，可使用不同的定義參數重複呼叫。子程式是從主程式呼叫。每個子程式可加上保護，以防止未經授權的讀取和顯示。→ 循環採取子程式的形式。

安全性功能

控制配備了永久啟用的監控功能，可即時偵測 CNC、PLC 和機台中的錯誤，大幅避免工件、刀具或機台損壞。萬一發生錯誤，加工操作會中斷，驅動器也會停止。記錄故障的原因，將以警報輸出。於此同時，PLC 會收到觸發 CNC 警報的通知。

定位軸

在加工刀具上執行輔助動作的軸（如刀匣、棘爪傳輸）。位置軸是不使用→路徑軸插補的軸。

定向主軸回退

RETTOOL：若加工被中斷（例如刀具斷裂時），可使用一個程式指令向使用者指定方向回退一個已定義的距離。

導螺桿錯誤補正

補正進給中使用的導螺桿的機械性不精確。控制系統使用補正的儲存誤差值。

尺寸規格、公制和英制

位置和螺距值可使用英制單位寫入加工程式。 與可程式設計尺寸無關（G70/G71），控制器設為基準系統。

工件座標系統

工件座標系統具有在→工件零點中的起點。在工件座標系統中程式設計加工操作，尺寸和方向參考此系統。

工件程式單節

→ 工件程式的工件，以換行作為區分。可分成兩種類型：→ 主單節和→子單節。

工件程式管理

工件程式管理可由工件組織而成。 使用者記憶體的大小會決定可管理的程式數量及資料量。每個檔案（程式及資料）名稱最多可含 24 個字母數字字元。

工件輪廓

設定要建立或加工的→工件輪廓。

工件零點

工件零點為→工件座標系統的起點。依據到→機械零點的距離而定義。

工作區

刀尖根據加工刀具的實體設計而可移入的 3D 區域。 請參閱→保護區。

工作區限制

有了工作區限制的輔助，除了限制開關以外，可進一步限制軸的移動範圍。 每軸可用一對數值定義受保護的工作區。

工作記憶體

RAM 為處理應用程式時，處理器於 CPU 中存取的工作記憶體。

巨集技術

在單一識別碼下聚集一組敘述。 識別碼代表程式中一組整合敘述。

幾何

工件座標系統中的→工件→說明。

幾何軸

幾何軸用來說明工件座標系統中的 2D 或 3D 區域。

序列 RS-232-C 介面

針對資料輸入 / 輸出，PCU 20 有一個序列 V.24 介面（RS232），而 PCU 50/70 則有兩個 V.24 介面。 可透過這些介面載入和儲存加工程式、製造商和使用者資料。

座標系統

請參閱→機台座標系統→工件座標系統。

後力補正

機械機台後作力的補正，例如，倒轉滾珠螺絲的後作力。 可個別輸入各軸後作力補正。

循環

受保護子程式可用來使用→在工件上的重複加工執行。

快速移動

軸的最高移動率。 例如，快速移動用於當刀具從靜止位置逼近→工件輪廓或將刀具從工件輪廓退刀時。 快速移動速率是以機台專屬為基準，使用機械參數元件設定。

手動超調

手動或可程式設計控制功能，可讓使用者配合特定工件或材質而調整程式設計進給率或速度。

按鍵開關

機械控制面板→上的按鍵開關有四個位置，由控制操作系統指派功能。 按鍵開關有三種不同顏色的按鍵，可以按指定位置移除。

接地線

接地的目的在於萬一故障時，即使有危險的接點電壓，裝置的所有非啟用連結總零件也不會作用。

插補器

→NCK 的邏輯元件，可根據在工件程式中指定的結束位置，定義要在個別軸中執行動作的中間值。

插補補正

插補補正是一種工具，提供製造商相關的導螺桿錯誤和測量系統錯誤補正(SSFK, MSFK)。

操作模式

SINUMERIK 控制的操作概念。 定義了下列模式： → 寸動進給→手動輸入→自動。

操作者介面

使用者介面（UI）是 CNC 顯示的螢幕顯示媒介。 具橫向和縱向軟鍵。

整體重置

萬一整體重置，則會刪除→CPU 的以下記憶體：

- →工作記憶體
- →負載記憶體的讀取 / 寫入區域
- →系統記憶體
- →備份記憶體

文字編輯器

請參閱→編輯器

旋轉

框架元件→定義環繞特定角度旋轉的座標系統。

曲率

輪廓曲率 k 為輪廓點(k = 1/r)中巢狀環的反向半徑。

曲線插補

有了曲線插補，控制器可以從一組輪廓的幾個指定插補點，便能產生平滑的曲線特性。

框架

框架是一種算術規則，可將一種直角座標系統轉換成其他的直角座標系統。 框架含有下列元件： →零點偏移，→旋轉，→比例，→鏡像。

極座標

一個座標系統，根據點到原點的距離與由半徑向量和定義軸所形成的角度，定義點在平面上的位置。

標準循環

標準循環是為經常重複的加工操作而提供的。

- 用於鑽孔 / 銑削的循環
- 用於銑床技術

可用的循環列在“Program”（程式）操作區中的“Cycle support”（循環支援）功能表中。一旦選擇了所需的加工循環，指定值所需的參數就會以純文字顯示。

模式群組

技術上相關的軸和主軸可合併成一個模式群組。 可由一個或多個→通道控制 BAG 的軸 / 主軸。 相同的→模式類型通常指派至模式群組的通道。

機械座標系統

與加工刀具的軸相關的座標系統。

機械控制面板

加工刀具上的操作員面板，例如具按鍵、旋轉開關等操作元件，以及如 LED 的簡單指示燈。透過 PLC 用來直接影響加工刀具。

機械軸

加工刀具上實體存在的軸。

機械零點

可加以追蹤所有（衍生的）測量系統的加工刀具固定點。

歸檔

從**外部**記憶體裝置上的檔案和 / 或目錄讀取。

無補正夾頭的攻牙

這項功能允許無補正夾頭的攻牙螺紋。 藉由將主軸作為轉軸、鑽孔軸的插補法，螺紋可切削成精確的最終鑽孔深度，例如，用於閉孔螺紋（需求：軸操作中的主軸）。

由輪廓快速回退

發生中斷時，可以透過 CNC 加工程式初始一個動作，使得刀具可以快速從目前加工的工件輪廓回退。 回退角度和回退距離也可以參數化。 快速回退後，也可以執行中斷常式（SINUMERIK 840D）。

直線軸

與轉軸相反，直線軸用來定義直線。

磨圓軸

磨圓軸將工件或刀具旋轉至對應指標網格的角度位置。 當到達網格指標時，磨圓軸即"到位"。

移動範圍

線性軸的最大允許移動範圍為±90。絕對值取決於選定輸入和位置控制解析度以及測量單位（英制或公制）。

程式單節執行

程式單節包括→ 工件程式的主程式和子程式。

程式設計鍵

在工件程式的程式設計語言中已有經定義的意義之字元和字元字串。

空白

加工前的工件原貌。

精加工工件輪廓

精加工工件輪廓。 請參閱→原始工件。

精確停止

當程式設定精確停止陳述式時，會精確逼近在單節中指定的位置，且若有需要，會使用非常緩慢的速度。為了降低逼近時間 → 為快速移動和進給定義精確停止臨界值。

精確停止臨界值

當所有路徑軸到達其精確停止臨界值時，控制會將其視為已到達其精確目的地點而回應。
→ 產生工件程式前單節。

系統記憶體

系統記憶體是 CPU 內的記憶體，用來儲存以下資料：

- 操作系統所需的資料
- 運算元次數、計數、標記

系統變數

無→ 工件程式的程式設計師輸入，就已存在的變數。是以資料類型和變數名稱之前加上字元**$**而定義的。請參閱→使用者定義變數。

絕對尺寸

軸移動的目的地是以參考目前有效座標系統原點的尺寸所定義的。 請參閱 → 增量尺寸

網路

網路係多重 S7-300 和其他終端裝置的連結，例如透過連接線的程式設計裝置。 資料交換發生於連結裝置間的網路上。

線性插補

加工工件時，刀具會沿著直線移動到目的地點。

編輯器

編輯器可供建立、編輯、延伸、聯結和匯入程式 / 文字 / 程式單節。

自動

控制的操作模式（根據 DIN 的單節順序操作）： NC 系統的操作模式→連續選擇和執行副程式。

英制測量系統

以英制和英制分數定義距離的測量系統。

螺旋插補

螺旋插補功能非常適用於使用成形銑刀加工內部和外部螺紋以及銑削潤滑溝槽。
螺旋有兩種移動方式：

- 在一平面上的螺旋移動
- 垂直於此平面的線性移動

補正值

軸位置之間的差距由編碼器和所需的程式設計軸位置測量。

補正表

含插補點的表格。 在基準軸上的選定位置提供補正軸的補正值。

補正記憶體

於控制系統中儲存含刀具偏移量的資料範圍。

補正軸

含設定點或由補正值修改實際值的軸。

訊息

在工件程式中程式設計的所有訊息→系統偵測到的警報，會以含有日期時間的純文字與取消條件的對應符號顯示在操作員面板上。 警報和訊息會分別顯示。

設定資料

傳遞加工刀具特性到 NC 的資料，如系統軟體所定義。

診斷

1. 控制的操作區。
2. 控制具有自我診斷程式以及用來提供服務的測試功能： 狀態、警報和服務畫面

識別碼

依據 DIN 66025，字組是用變數（算術變數、系統變數、使用者變數）、子程式、關鍵字和含多重位址字母的字之識別碼（名稱）附加而成的。 這些補充和單節格式相關字組具有相同的意義。 識別碼必須是獨一無二的。 不可將相同的識別碼用在不同的物件上。

警報

全部→訊息和警報會以含有日期時間的純文字與取消條件的相應符號顯示在操作員面板上。 警報和訊息會分別顯示。

1. 工件程式中的警報和訊息：
   可直接從工件程式以純文字顯示警報和訊息。
2. 自 PLC 來的警報和訊息
   可直接從 PLC 程式以純文字顯示警報和訊息。 此用途不需要其他的功能單節套件。

變數定義

變數定義包括指定資料類型和變數名稱。變數名稱可用來存取變數值。

象限錯誤補正

導引路徑上摩擦力改變所導致的象限變化上的輪廓錯誤，可利用象限錯誤補正完全排除。 象限錯誤補正的參數化由電路測試執行。

負載記憶體

負載記憶體和→PLC 之 CPU 314 的 RAM 一樣。

資料單節

1. HIGHSTEP 程式可存取的→PLC→資料元件。
2. →NC 的資料元件： 資料模組包含全域使用者資料的資料定義。 這些資料在定義後可直接初始化。

資料字組

在一個資料單節中→的雙位元資料單元。

路徑軸

路徑軸包含由插補器以同時啟動、加速、停止和到達終點的方式所控制通道的所有機械軸。

路徑速率

最大可程式設計路徑速度取決於輸入解析度。 例如，解析度 0.1 毫米時，最大可程式設計路徑的速率為 1000 公尺 / 分鐘。

路徑進給率

路徑進給效果→路徑軸。 代表→相關幾何軸的進給率幾何總和。

軟鍵

名稱顯示在畫面區域上的按鍵。軟鍵選項會動態配合操作狀況而顯示。 可在軟體中為可自由指派的功能鍵（軟鍵）指派定義的功能。

軟體極限開關

軟體限制開關會限制軸的移動範圍，避免硬體限制開關上的滑板瞬間停止。可為每個軸指定兩個數值對，並使用→PLC 分別加以啟用。

軸

CNC 軸可根據其功能範圍細分為：

- 軸： 插補路徑軸
- 輔助軸： 含軸專屬進給率的非插補進給與位置軸。 輔助軸與實際加工無關，例如刀具進給器、刀匣等。

軸位址

請參閱→軸識別碼

軸名稱

請參閱→軸識別碼

軸識別碼

如 DIN 66217 中所定義，軸是以 X、Y 和 Z 來識別，適用於順時針、直角的→座標系統。

轉軸會環繞 X、Y 和 Z 旋轉，以 A、B 和 C 來識別。可使用其他字母命名其他和指定軸平行的軸。

輔助功能

輔助功能可使→工件程式傳輸→參數到→PLC，然後會觸發由機台製造商定義的反應。

輪廓

工件的輪廓。

輪廓監控

作為輪廓精確度的測量方法，可定義的允差頻帶中監控到下列錯誤。例如，高到令人無法接受的後續錯誤會造成驅動器過載。在這種情況下，會輸出警告並停止各軸。

轉換

軸的附加或絕對零點偏移量。

轉軸

將工件或刀具旋轉套用到定義角度位置的轉軸。

通道

通道的特性是可以→獨立處理其他通道的工件程式。通道以特定方式控制指派的軸和主軸。不同通道的工件程式執行可透過同步化而協調一致。

速率控制

為了使每個單節上極輕微動作達成可接受的移動率，可指定數個單節預期評估（→ 預見）。

連接線

連接線為預先組合或使用者組合的雙絞線，每一端各有一個接頭。此連接線藉著多點式介面（MPI）將→CPU 連接至→程式設計裝置→或至其他的 CPU。

連續路徑模式

連續路徑模式的目標在於避免實質上的減速→工件程式單節邊界上的路徑軸，並儘可能接近相同路徑速度變更下一個單節。

週邊模組

I/O 模組呈現 CPU 和處理之間的連結。

I/O 模組為：

- → 數位輸入 / 輸出模組
- → 類比輸入 / 輸出模組
- →模擬模組

進給手動超調

程式設計速率是透過 → 機台控制面板或從→PLC（0 到 200%）所作的目前速率設定而調整。也可以利用加工程式中的可程式設計百分比係數(1-200%)修正進給率。

逼近固定點

加工刀具可使用定義方式逼近定點，如刀具變更點、載入點、棘爪變更點等。 這些點的座標會儲存在控制系統中。 控制系統於可行的狀況下，便會快速移動→相關軸。

選取

到 NC 的一系列敘述，會共同作用以產生特定的工件。同樣地，此條件適用在執行已知→原始工件上的特定加工操作。

鏡射

鏡像反轉輪廓相對於軸的座標值符號。可一次相對於多軸進行鏡射。

關鍵字

具有特定標記法、且在→工件程式的程式設計語言中有已定義意義的字組。

限制速度

最高 / 最低（主軸）速度： 可藉由指定機械參數、PLC 或設定參數來限制主軸的最高速度。

零點偏移

藉由參考現有零點和→框架，為座標系統指定新的參考點。

1. 可設定的

   SINUMERIK 840D： 每個 CNC 軸都有可設定數量的可設定零點偏移量。 以 G 系列功能選擇的偏移量會輪流生效。

2. 外部

   除了所有定義工件零點位置的偏移量之外，外部零點偏移量可藉由手輪（DRF 偏移量）或從 PLC 手動超調。

3. 可程式設計的

   可使用 TRANS 敘述來程式設計所有路徑和位置軸的零點偏移量。

非同步子程式

可使用中斷訊號非同步啟動至（獨立的）目前程式狀態（如 “Rapid NC input”（快速 NC 輸入）訊號）的工件程式。

預先一致

當路徑距離逼近到相當於端點位置的指定舊資料時，發生單節更改。

預讀控制

預測功能藉由預測移動單節的可指派數目，以獲得最佳加工速度。

驅動器

驅動為 CNC 的元件，根據 NC 的設定執行速度和扭矩控制。

高速數位輸入 / 輸出

數位輸入可用於如啟動快速 CNC 程式（中斷常式）等方面。 數位 CNC 輸出可用來觸發快速程式控制的切換功能（SINUMERIK 840D）。

高階 CNC 語言

高階語言提供：→ 使用者定義變數→系統變數→巨集技術。

索引

符號

0

3

A

B

C

D

E

F

G

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

A

二劃

三劃

四劃

五劃

六劃

七劃

八劃

九劃

十劃

十一劃

十二劃

十三劃

十四劃

十五劃

十六劃

十七劃

十八劃

十九劃

二十劃

二十二劃

二十三劃