OpenCNC_MACRO發展工具操作手冊

匯出日期：2023-05-10
修改日期：2022-09-11

1 前言

為增加控制器應用彈性，新代控制器提供MACRO程式編輯功能。

當加工程式被宣告成MACRO格式後，該檔案將如同一般程式語言，可使用特定數學函數，如此一來，除了原本即有的移動和補償指令功能，更擁有邏輯判斷及數學演算功能。

2 檔案格式

程式內容第一行需使用"%"宣告為標題行，并加入關鍵字"@MACRO"，才會被視為 MACRO 格式檔處理。否則該檔案將被視為ISO格式檔處理。

ISO 格式檔與 MACRO 格式檔的程式內容解讀差異如下

<table>
<tr><th>ISO格式檔</th><th>MACRO格式檔</th></tr>
<tr><td>不支援 MACRO 語法使用
(有部分 MACRO 語法無法被正確解讀，可能會跳出"COM-003 控制器解譯宏程序時發現程式句法有誤"的警報)</td><td>可以使用 MACRO 語法的完整功能。</td></tr>
<tr><td>每一行結尾加或不加分號";"皆可。</td><td>1. 除了特定語法外(請參考語法說明內容)，每一行結尾都要加上分號";"。
2. 該行沒有加上分號時，會和下一行合併處理。若沒有檢查出警報，仍會正常執行；若有檢查出來，會跳出COM-008 子句中沒有結束的符號';'。
範例:
<table>
<tr><th>原加工程式</th><th>等效程式</th><th>說明</th></tr>
<tr><td>%@MACRO
#1= SIN(100) // 漏加";"
G01 Y100.;
M99;</td><td>%@MACRO
#1= SIN(100)G01 Y100.;
M99;</td><td>由於合併後指令不合法
會跳出 COM-008</td></tr>
<tr><td>%@MACRO
G01 X100. // 漏加";"
G01 Y100.;
M99;</td><td>%@MACRO
G01 X100.G01 Y100.;
M99;</td><td>由於合併後指令可直接執行
所以不會跳出警報
會直接走到 X100. Y100.</td></tr>
</table></td></tr>
<tr><td>使用到"(...)"，刮號內的內容...，會視為注解省略。</td><td>使用到"(* ... *)"，刮號內的內容...，會視為注解省略。</td></tr>
<tr><td>依照 PR3201 車床程序所設定的操作習慣書寫。</td><td>只能以車床習慣C-type書寫。</td></tr>
</table>

備註:

1. 被當作副程式（子程式）或巨集呼叫的加工檔，建議不要使用多軸群檔案（含$1、$2）。
   若有此應用情境，多軸群副程式檔案必須小於60KB(60000bytes)，否則控制器將報警COR-203 加工程序格式不符；單軸群副程式則不在此限。
2. 不支援MACRO格式的多軸群（含$1、$2）的副程式。

3. 檔案若超過60KB(60000bytes)，無法支援IF、CASE、REPEAT、FOR、WHILE等等有範圍區間的語法，否則會報警COR-204 檔案太大。
4. 除了程式註解、使用字串做為引數的Macro語法以外，僅允許以ASCII撰寫加工程式。若使用者在加工程式中撰寫了ASCII以外的字元，將會跳警報COM-027 無效的字元。
5. 支援 CR ("\r")、LF ("\n")、CR LF ("\r\n") 三種換行方式。

3 指令格式Block Format

單行動作控制指令的撰寫格式敘述如下:

/	N	G	X	Y	Z	A	B	C	I	J	K	F	S	T	D	M

/	單節選擇性跳躍功能，需配合PLC的C41；不支援函數語法及變數運算
N	單節次序碼，必須撰寫在該單節的第一碼位置，且不可在同一行後面撰寫MACRO指令或變數指定
G	功能指定碼，需撰寫在N碼之後
X	X軸的移動命令，或是擴充G碼的引數，需撰寫在G碼後
Y	Y軸的移動命令，或是擴充G碼的引數，需撰寫在G碼後
Z	Z軸的移動命令，或是擴充G碼的引數，需撰寫在G碼後
A	A軸的移動命令，或是擴充G碼的引數，需撰寫在G碼後
B	B軸的移動命令，或是擴充G碼的引數，需撰寫在G碼後
C	C軸的移動命令，或是擴充G碼的引數，需撰寫在G碼後
I	X方向的的半徑命令，或是擴充G碼的引數，需撰寫在G碼後
J	Y方向的的半徑命令，或是擴充G碼的引數，需撰寫在G碼後
K	Z方向的的半徑命令，或是擴充G碼的引數，需撰寫在G碼後
F	單節進給速度，或是擴充G碼的引數
S	主軸旋轉速度，或是擴充G碼的引數
T	刀具選擇功能，或是擴充G碼的引數
D	刀具補償功能，或是擴充G碼的引數
M	輔助功能，或是擴充G碼的引數

核心解譯處理順序（1.最先 ~ 10.最後）：

1. 模態G碼（G15、G17、G70等）、擴充G碼巨集（G73、G84等）
2. M碼巨集、T碼巨集
3. S碼
4. F碼
5. H碼
6. D碼
7. T碼
8. M碼
9. B碼
10. 插值G碼（G0、G1等）、功能G碼（G4、G51、G68等）

備註:

1. 其餘未說明之指令格式由相關G碼以引數形式帶入
2. 一般在副程式中會使用"GETARG"函數來讀取引數內容，而在主程式（父程式）中可使用之引數形式規則如下:
    a. 使用引數D、E、H、I、J、K、L、M、P、Q、R、T，僅能使用單一符號傳入引數，例如「G101 X30. Y40. D50. ;」，若在其後附帶數字將引發警報，例如「G101 X30. Y40. D1=50. ;」
    b. 使用引數A、B、C、F、S、U、V、W、X、Y、Z，除了使用單一符號傳入引數，亦可在其後附帶數字，例如「G101 X30. Y40. Z1=50. ;」
    c. 以上所有動作指令之後，僅可使用數值或存值為數值之變數，否則可能因程式解譯之編碼限制而造成系統錯誤，此誤用情況不在控制器的保護範圍內。

4 運算子（Operator）

運算子	符號	執行順序
括號	( ) [ ]	1
函數賦值	Identifier (引數)	2
負號	-	3
補數	NOT	3
乘號	*	4
除號	/	4
模數（餘數）	MOD	4
加號	+	5
減號	-	5
比較	<,>,<=,>=	6
等於	=	7
不等於	<>	8
布林運算"且"	&AND,	9
布林運算"互斥"	XOR	10
布林運算"或"	OR	11

註1:

使用『/』元件（除號）時，需注意若分子與分母都是整數，所得結果仍為整數。整數及非整數之差別在有無加入小數點。

範例:

- 分子為非整數：1. / 2 = 0.5
- 分母為非整數：1/ 2.0 = 0.5
- 分子、分母均為整數：1/2 = 0

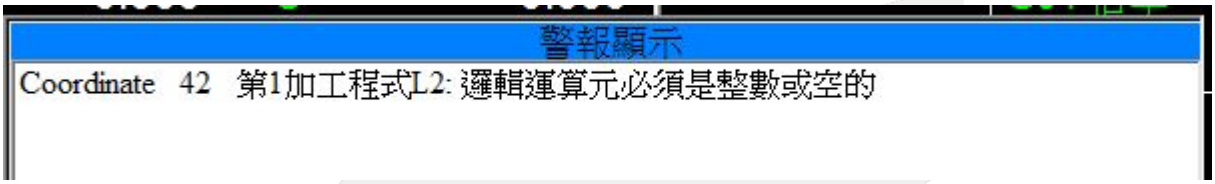
- 刮號內均為整數：(1/2)*1.0 = 0

註2:

MOD運算子（模數）僅適用於數值型態為Long，若數值型態為Double，則會出現下述警報訊息。

範例:

```
%@MACRO
@1:= 4. MOD 3;
M99;
```



5 語法說明

5.1 變數指定

變數指定	
語法	**<變數>** := **<敘述>**;
說明	指定變數內容
範例一 **直接定值**	

```
@1 := 123;
#1 := 456;

AR1 := 789;      // App 區域變數
MAR1 := 789;     // App 永久儲存變數

#10 := "12";     // 區域變數#10內容為12
@10 := "12";     // 公用變數@10內容為12849
```

範例二 **間接定值**	

```
#1:= 123;

@[#1] := 567;        // @123=567
@[#1+7]:=890;        // @130=890

AR[#1] := 789;       // AR123=789
MAR[#1] := 789;      // MAR123=789
```

備註	1. 範例一中的"12"為字串寫法，表示要將字串存入變數中，只有存入公用變數時，控制器會先進行ASCII轉碼，區域變數則不會。 2. 承上，欲正確讀取公用變數所儲存的字串內容，請使用**SCANTEXT函數**。 3. 範例二中請留意用的是"中刮號" 4. App變數(AR、MAR)使用限制: a. 支援版本：10.118.39及之後版本。 b. 只有APP專用的Macro才支援存取AR、MAR。若在其他加工程式呼叫，則跳警報COR-016。 c. 若欲存取的變數超出範圍，例如只有512個MAR卻下MAR512 := 1;，則跳警報COR-016。(AR、MAR變數號碼為0-Based) d. 直接存取負的變數位址，例如MAR-1 := 1;，則跳警報COM-003。 e. 直接存取小數的變數位址，例如MAR1.1 := 1;，則跳警報COM-003。 f. 間接存取負的變數位址，例如MAR[-1] := 1;，則跳警報COR-016。 g. 間接存取小數的變數位址，例如MAR[1.1] := 1;，則跳警報COR-016。 h. AR、MAR皆不支援存字串。 i. 僅支援Macro格式，不支援ISO格式。

5.2 GOTO

GOTO	
語法	**GOTO n;**
說明	此語法需配合單節次序碼使用，可跳到指定的N行號執行其內容，假設程式中同時存在兩個N行號，則以該程式中第一個N行號為準。

範例	%@MACRO
	#1 := 1;
	#2 := 10;
	IF #1 = 1 THEN
	GOTO #2;
	END_IF;
	IF#1 = 2 THEN
	GOTO 100;
	END_IF;
	N10;
	G01 G90 X50. Y0. F1000;
	M30;
	N100;
	G01 G90 X0. Y50. F1000;
	M30;

備註	使用REPEAT/WHILE/FOR/GOTO等迴圈功能時，應謹慎注意無窮迴圈問題，當此問題發生時，會造成人機畫面鎖死或加工程式卡死。 建議在迴圈中適時加入**SLEEP()函數**，若不小心進入無窮迴圈，仍有機會操作人機畫面以中止程式執行。

5.3 CASE

CASE	
語法	**CASE** <條件變數> **OF** **<變數>:** <陳述列表> **<變數>,<變數>:** <陳述列表> **<變數>,<變數>,<變數>:** <陳述列表> **ELSE** <陳述列表> **END_CASE;**
說明	多條件判斷，根據條件變數內容，分別執行不同程式區塊。請注意"**變數**"內容需為大於等於0之**整數**型態。
範例	%@MACRO #1 := 1; G01 G90 X0. Y0. F1000; CASE #1 OF 1: X(1.0*#1) Y1.0*#1);.. 2: X(2.0*#1) Y2.0*#1);.. 3, 4, 5: X(3.0*#1) Y3.0*#1);.. ELSE X(4.0*#1) Y4.0*#1);.. END_CASE; M30;

5.4 IF

IF

語法	**IF** <條件> **THEN**
	  <陳述列表>
	ELSEIF <條件> **THEN**
	  <陳述列表>
	ELSE
	  <陳述列表>
	END_IF;

說明	IF條件判斷

範例	%@MACRO
	#1 := 3.0;
	G01 G90 X0. Y0. F1000;
	IF #1 = 1 THEN
	  X(1.0*#1) Y(1.0*#1);..
	ELSEIF #1 = 2 THEN
	  X(2.0*#1) Y(2.0*#1);..
	ELSEIF #1 = 3 THEN
	  X(3.0*#1) Y(3.0*#1);..
	ELSE
	  X(4.0*#1) Y(4.0*#1);..
	END_IF;
	M30;

5.5 REPEAT

REPEAT	
語法	**REPEAT** <陳述列表> **UNTIL** <條件> **END_REPEAT**;
說明	REPEAT迴圈控制

範例	%@MACRO
	#10 := 30.;
	#11 := 22.5.;
	#12 := #10/2;
	#13 := #11/2;
	#14 := 2.0;
	#15 := 1.5;
	G01 G90 X#12 Y#13 F1000;
	REPEAT
	  G00 X(#12+#14) Y(#13+#15);
	  G01 X(#12+#14) Y(#13-#15);
	  G01 X(#12-#14) Y(#13-#15);
	  G01 X(#12-#14) Y(#13+#15);
	  G01 X(#12+#14) Y(#13+#15);
	  #14 := #14 + 2.0;
	  #15 := #15 + 1.5;
	UNTIL (#14 > #12) OR (#15 > #13) END_REPEAT;
	M30;

備註	使用REPEAT/WHILE/FOR/GOTO等迴圈功能時，應謹慎注意無窮迴圈問題，當此問題發生時，會造成人機畫面鎖死或加工程式卡死。 建議在迴圈中適時加入**SLEEP()函數**，若不小心進入無窮迴圈，仍有機會操作人機畫面以中止程式執行。

5.6 WHILE

WHILE	
語法	**WHILE** <條件> **DO** <陳述列表> **END_WHILE**;
說明	WHILE迴圈控制

範例	%@MACRO
	#10 := 30.;
	#11 := 22.5.;
	#12 := #10/2;
	#13 := #11/2;
	#14 := 2.0;
	#15 := 1.5;
	G01 G90 X#12 Y#13 F1000;
	WHILE (#14 <= #12) AND (#15 <= #13) DO
	  G00 X(#12+#14) Y(#13+#15);
	  G01 X(#12+#14) Y(#13-#15);
	  G01 X(#12-#14) Y(#13-#15);
	  G01 X(#12-#14) Y(#13+#15);
	  G01 X(#12+#14) Y(#13+#15);
	  #14 := #14 + 2.0;
	  #15 := #15 + 1.5;
	END_WHILE;
	M30;

備註	使用REPEAT/WHILE/FOR/GOTO等迴圈功能時，應謹慎注意無窮迴圈問題，當此問題發生時，會造成人機畫面鎖死或加工程式卡死。 建議在迴圈中適時加入**SLEEP()函數**，若不小心進入無窮迴圈，仍有機會操作人機畫面以中止程式執行。

5.7 FOR

FOR	
語法	**FOR** <變數1> := 敘述1> **TO** <敘述2> **BY** <敘述3> **DO** <陳述列表> **END_FOR;** 變數1：控制迴圈次數的變數 敘述1：迴圈計數的起始次數，可為數值或運算式 敘述2：迴圈計數的終止次數，可為數值或運算式 敘述3：迴圈計數每次的累加次數，可為數值或運算式 陳述列表：每次迴圈之執行內容
說明	FOR迴圈控制

範例	%@MACRO #10 := 30.; #11 := 22.5.; #12 := #10/2; #13 := #11/2; #14 := 2.0; #15 := 1.5; G01 G90 X#12 Y#13 F1000; FOR #6 := 0 TO 3 BY 1.0 DO G00 X(#12+#14) Y(#13+#15); G01 X(#12+#14) Y(#13-#15); G01 X(#12-#14) Y(#13-#15); G01 X(#12-#14) Y(#13+#15); G01 X(#12+#14) Y(#13+#15); #14 := #14 + 2.0; #15 := #15 + 1.5; END_FOR; M30;
備註	1. 使用REPEAT/WHILE/FOR/GOTO等迴圈功能時，應謹慎注意無窮迴圈問題，當此問題發生時，會造成人機畫面鎖死或加工程式卡死。 2. 建議在迴圈中適時加入**SLEEP()函數**，若不小心進入無窮迴圈，仍有機會操作人機畫面以中止程式執行。 3. 請勿在 FOR 迴圈內使用會跳轉進出迴圈的命令，例如複式切削循環(G72-G78)、使用GOTO跳轉出迴圈再跳轉回來、M98 H_，此會造成迴圈的累加次數(<敘述3>)數值異常 a. 導致執行異常範例

```
// FOR 執行時會變為每次累加 5
%@MACRO
FOR #10:=1 TO 100 BY 1 DO
GOTO 12;
N13;
END_FOR;
M30;

N12;
M00;
@1:=@1+5;
GOTO 13;
M99;
```

5.8 EXIT

EXIT	
語法	**EXIT;**

說明	中斷迴圈，跳離迴圈控制

範例	%@MACRO
	#10 := 30.;
	#11 := 22.5.;
	#12 := #10/2;
	#13 := #11/2;
	#14 := 2.0;
	#15 := 1.5;
	#16 := 1.0;
	G01 G90 X#12 Y#13 F1000;
	FOR #6 := 0 TO 3 BY 1.0 DO
	IF((#14 = 4) & (#16 = 1)) THEN
	EXIT;
	END_IF;
	G00 X(#12+#14) Y(#13+#15);
	G01 X(#12+#14) Y(#13-#15);
	G01 X(#12-#14) Y(#13-#15);
	G01 X(#12-#14) Y(#13+#15);
	G01 X(#12+#14) Y(#13+#15);
	#14 := #14 + 2.0;
	#15 := #15 + 1.5;
	END_FOR;
	M30;

5.9 程式註解

程式註解	
語法	**(*** <陳述列表> ***)** **//** <陳述列表>
說明	程式註解（Comment）

範例一
單行註解
	%@MACRO
	G00 G90 X0. Y0.; // 移動回原點
	M30;

範例二 **區塊註解**	%@MACRO (* 此區塊為註解區 不管內容為何均不影響程式執行 *) G00 G90 X0. Y0.; G00 G90 X10. Y0.; G00 G90 X10. Y10.; G00 G90 X0. Y10.; G00 G90 X0. Y0.; M30;
備註	若於符合註解語法以外的陳述列表區域添加屬於註解之文字，可能會因解譯編碼之限制造成系統錯誤，此誤用情況不在控制器的保護範圍內。

5.10 程式執行範圍

程式執行範圍	
語法	% 加工程序 %
說明	1. 程式內容中使用到"% ... %"，兩個%符號內的加工程序，會被系統執行；兩個%符號外(包含第一個%符號前面和第二個%符號后面)的加工程序，系統會予以忽略。 2. 程式內容中僅使用了一個%符號，則從第一個%符號開始執行到檔案結束。
範例一	G91 G00 X10. % G91 G00 Y10. % G91 G00 Z10. M30 //執行後可以發現只有Y軸會移動10.，其他軸向不動。
範例二	G91 G00 X10. % G91 G00 Y10. G91 G00 Z10. M30 //執行後可以發現Y、Z軸都會移動10.，而X軸保持不動。

MACRO讀取/處理流程

圖示	說明

	以下依序解釋父程式（主程式）每一行的動作：（左方框框內的程式）
	• N1：設定座標系為G54，並採絕對模式G90移動
	• N2：呼叫G0201巨集，並透過GETARG函數將X1引數帶入
	  • 進入G0201後，將X1引數儲存於區域變數#1
	  • 以#10備份G90/G91狀態
	  • Y軸以G00增量移動10mm
	  • 還原G90/G91狀態
	  • 返回父程式
	• N3：由於離開G0201前最後一個插值模式為G00，故此單節X軸仍以G00移動
	• N4：呼叫G0202巨集，並透過#24將X引數帶入
	  • 進入G0202後，將X引數儲存於區域變數#1
	  • 以#10備份G90/G91狀態
	  • Y軸以G00增量移動-10mm
	  • 還原G90/G91狀態
	  • 設定插值模式為202
	  • 返回父程式
	• N5：由於離開G0202前將插值模式儲存為202，故執行此單節時，系統將再次呼叫G0202
	• N6：程式結束

6 MACRO撰寫注意事項

- MACRO內建議多使用區域變數（Local Variables，#1 ~ #400），除非有跨MACRO的需求才使用公用變數（Global Variables，@1 ~ @165535）。
- 執行MACRO時，使用者的資料是透過引數（A_、B_、…,、Z_、X1=、*Y1*=、…）傳入，引數與區域變數相通，下表為引數與區域變數之關係。
- 針對擴充之引數位址，例如X1引數，請使用GETARG函數來讀取其數值。

引數	對應變數	引數	對應變數	引數	對應變數
A	#1	J	#5	T	#20
B	#2	K	#6	U	#21
C	#3	L	#12	V	#22
D	#7	M	#13	W	#23
E	#8	P	#16	X	#24
F	#9	Q	#17	Y	#25
H	#11	R	#18	Z	#26
I	#4	S	#19	X1	GETARG(X1)

- 模式變數（Modal Variables，#2001 ~ #2100、#3001 ~ #3100）在系統重置時，會回復成VACANT狀態，因此可應用於多個MACRO間進行資料交換之時機，以節省變數資源之使用。
- MACRO若需要內定初始值，可多加利用使用者參數（Customer Parameter，#4001 ~ #4100、#5001 ~ #5100）。
- 執行MACRO副程式（子程式）時，若模式G碼會被改變（G91/G90、G40/G41/G42,…等），請在一進入MACRO時，即備份當前的狀態，待離開MACRO前再回復原有狀態。
- 若想在離開MACRO後，繼續保留此MACRO之插值模式（#1000），建議在離開MACRO前，將#1000指定為該MACRO之號碼。爾後只要是軸向位移之指令單節，系統將自動呼叫此MACRO，而不用再次指定。
    - 插值模式在遇到G00/G01/G02/G03/G31/G33或#1000內容變動時，將被自動改寫。
- 對於長度或角度的引數，在運算前請使用STD函數將單位標準化，以符合工具機使用習慣。
- 不可改變座標系統設定，像是G92/G54/G52等與座標系統相關之指令，否則圖形模擬功能將失去參考意義。
- 執行加工時，核心會預解MACRO內容，因此MACRO執行速度會超前G/M碼指令，若變數指定或資料讀取需與G/M碼動作同步，請於變數指定或資料讀取前加入WAIT函數，以確保動作正確。
  注意：WAIT 函數目的是為了確保在運動指令執行完畢前，就因預解而提早讀寫相關數值所造成的錯誤，因此，其只確保 WAIT 前的 G/M 碼執行完畢前不會繼續往後預解 (其中 M98、M99、M198 三者為例外)，對其他指令則不會有作用。
- MACRO程式最後需加"M99;"，才能返回主程式（父程式）。
- 請養成良好習慣，在程式中多加入註解，增加程式可讀性，幫助後續維護及問題排除。

6.1 登錄G碼巨集

- 開發人員可以根據機臺需求來新增標準G碼以外的G碼巨集，也可以客制標準G碼的內容。
- 透過【Pr3701~3710 登錄G碼呼叫巨集】設定，登錄想要客制的標準G碼，當程序中執行到對應的G碼時，將不再執行標準G碼之動作，而是執行客制G碼巨集之內容。
- 以下是【Pr3701~3710 登錄G碼呼叫巨集】設定值與目前開放客制的標準G碼對照表

Pr3701	標準G碼	G碼巨集檔案名稱
0	無	無
-1	G00	G0000
1	G01	G0001
2	G02	G0002
3	G03	G0003
4	G53	G0053
5	G40	G0040
6	G41	G0041
7	G42	G0042
8	G43	G0043
9	G44	G0044
10	G49	G0049
11	G04	G0004

- 以下是登錄G碼巨集之運作規格
    - 巨集特性視同G碼巨集特性
    - 登錄G碼巨集程序內的所有G碼，均是標準G碼
    - 標準G碼中為插值模式的G碼（例如G00、G01、G02、G03）在登錄客製G碼後，仍然具有繼承功能
        - 例如執行
          G00 X100.

Y100.
其中Y100.也會執行G00巨集，并且會讀取占用Y引數

- 若登錄巨集中，有改變插值模式，則務必在離開登錄巨集前，將插值模式還原
  - 例如登錄G00為登錄G碼巨集
    在G0000巨集中，如果將插值狀態變更為G01
    則在離開G0000前，需將插值模式變更回G00
    避免離開登錄巨集之後的狀態錯亂
- 登錄G碼巨集遇到以下指令時，將無法運作
  - 車床 G7.1
  - 車床 G12.1
  - 車床 ,A
  - 車床 ,R
  - 車床 ,C
  - 車床 所有加工循環指令
  - 銑床 所有加工循環指令
  - T碼巨集

- 相容性異動

版本	異動
~Before	登錄G碼與"G碼巨集"的解譯順序相同。
10.114.50	• 登錄G碼與"標準G碼"的解譯順序相同。 • 不再支援原先在G碼巨集中將插值模式改為900000（#1000 := 900000）的特殊用法
10.116.16A、10.116.17	提供G53以自訂MACRO（G0053）取代
10.118.22F、10.118.26	提供G40、G41、G42以自訂MACRO（G0040、G0041、G0042）取代
10.118.45	提供G43、G44、G49以自訂MACRO（G0043、G0044、G0049）取代
10.118.52D, 10.118.54	提供G04以自訂MACRO（G0004）取代

7 函數表（Function List）

功能	說明

ABS	計算某數值的絕對值
	EX:
	#10 := -1.1;
	#1 := ABS(#10); // #1 = 1.1
	#2 := ABS(-1.2); // #2 = 1.2

ACOS	計算某數值的反餘弦值
	EX:
	#10 := 1;
	#1 := ACOS(#10); // #1 = 0
	#2 := ACOS(-1); // #2 = 180

ALARM	觸發巨集警報
	EX:
	ALARM(300); // 觸發巨集第300號警報
	ALARM(301, "ALARM 301 Content");
	備註:
	1. 會伴隨觸發警報COR-027【宏程序發出警報】。
	2. 警報內容有字串長度限制，中文 : 19個字，英文 : 39個字。
	3. 警報ID只允許使用0~65535之間的整數，否則跳警報COR-02

ASIN	計算某數值的反正弦值
	EX:
	#10 := 1;
	#1 := ASIN(#10); // #1 = 90
	#2= ASIN(-1); // #2 = -90

ATAN	計算某數值的反正切值。計算角度範圍介於±90°
	EX:
	#10 := 1;
	#1 := ATAN(#10); // #1 = 45
	#2 := ATAN(-1); // #2 = -45

功能	說明

ATAN2(Y, X)	計算Y/X的反正切值，並根據輸入引數(X, Y)所處的象限位置決定
	EX:
	#10 := 1;
	#20 := -1
	#1 := ATAN2(#10, #20); // #1 = 135
	#2 := ATAN2(#20, #10); // #2 = -45
	#3 := ATAN2(1, 0); // #3 = 90
	注意事項:
	1. 有效版本: 10.118.29W, 10.118.40C, 10.118.42
	2. 引數X和引數Y必須為數字，否則會跳警報COR-023 【語意錯
	3. 引數X和引數Y不可同時為0，否則會跳警報COR-004 【運算錯
	錯誤範例:
	@1 := ATAN2("1", 1); // 引數非數字，跳警報COR-023
	@2 := ATAN2(0, 0); // 引數皆為0，跳警報COR-004

AXID	查詢軸名稱所對應的軸編號，當該軸名稱不存在時，回傳值為
	EX:
	假設第六軸名稱為Y2（Pr326=202），第二軸名稱為Y（Pr
	#1 := AXID(Y); // #1 = 2
	#2 := AXID(Y2); // #2 = 6

CEIL	回傳大於或等於某數值的最小整數
	EX:
	#10 := 1.4;
	#1 := CEIL(#10); // #1 = 2
	#2 := CEIL(1.5); // #2 = 2

CLOSE	關閉由OPEN函數開啟的檔案，程式結束後該檔案亦會自動關閉 EX: CLOSE(); // 關閉檔案

<table>
<tr><th>功能</th><th>說明</th></tr>
<tr><td>INT DBOPEN("File name")</td><td>功能:
• 開啟既有的 Cycle 檔案
影響:
• 執行成功後，會指定此檔案為後續操作的目標檔案
引數 - File name:
• 欲開啟檔案的檔名
回傳值:<table><tr><th>數值</th><th>意義</th></tr><tr><td>0~N</td><td>開檔成功，檔案中的 Cyc</td></tr><tr><td>0</td><td>開檔失敗</td></tr></table>備註:
1. 引數型態錯誤會觸發警報 COR-023 語義錯誤
2. 此 Macro 語法不支援圖形模擬，會固定回傳 0
3. 指定路徑可能被客制 Action（CUSTOMFILE_CYCLE1~5）影響
4. 同時間僅能開啟一個 Cycle 檔案，而 DBOPEN 以及 DBNEW
Ex: 先使用 DBOPEN("FileA")，再使用 DBNEW("FileB")，此時
5. 若有重新載入檔案的需求，需先重置系統（在背景運算元件
EX:
<pre>1 // 1.
2 DBOPEN("Test.cyc");
3 // 載入GNCFILES\\Test.cyc資料檔
4
5 // 2.
6 #1 = 51;
7 DBOPEN("FLAT\\\\AB#1[3]ZZ.cyc");
8 // 載入FLAT\\\\AB051ZZ.cyc 資料檔, [3]表</pre></td></tr>
</table>

<table>
<tr><th>功能</th><th>說明</th></tr>
<tr><td>BOOL DBLOAD(Index)</td><td>功能:
• 讀取第 "Index" 筆的 Cycle 資料
影響:
• 執行成功後，會以目標位置資料的 Cycle name 作為
引數 - Index:
• 指定要讀取第幾筆資料
• 限制: 最大值為"目前開啟的 Cycle 檔的最後位置"，e
回傳值:<table><tr><th>數值</th><th>意義</th></tr><tr><td>1</td><td>讀取成功</td></tr><tr><td>0</td><td>讀取失敗</td></tr></table>備註:
1. 引數型態錯誤會觸發警報 COR-023 語義錯誤
2. 此 Macro 語法不支援圖形模擬，會固定回傳 0
3. 此 Macro 語法會操作目前開啟 Cycle 檔。Cycle 檔的開啟可
4. 同時間僅能指定一個 Cycle name，而 DBLOAD 以及 DBINSE
Ex:先使用 DBLOAD(0)，Cycle name 會指定為位置"0"資料的
EX:<pre>1 DBOPEN("FLAT\\TAB01");
2 // 載入FLAT\\TAB01資料檔
3 DBLOAD(0);
4 // 讀取第0筆資料
5 DBLOAD(1);
6 // 讀取第1筆資料</pre></td></tr>
</table>

<table>
<tr><th>功能</th><th>說明</th></tr>
<tr><td>BOOL DBSAVE(Index)</td><td>功能:
• 根據目前指定的 Cycle name，覆蓋第 "Index" 筆的 C
引數 - Index:
• 指定要覆寫第幾筆資料
• 限制: 最大值為"目前開啟的 Cycle 檔的最後位置"，e
回傳值:
<table>
<tr><th>數值</th><th>意義</th></tr>
<tr><td>1</td><td>覆寫成功</td></tr>
<tr><td>0</td><td>覆寫失敗</td></tr>
</table>
備註:
1. 引數型態錯誤會觸發警報 COR-023 語義錯誤
2. 此 Macro 語法不支援圖形模擬，會固定回傳 0
3. 此 Macro 語法會操作目前開啟 Cycle 檔。Cycle 檔的開啟可
4. 此 Macro 語法會使用目前指定的 Cycle name 覆寫目標位置
5. 版本：支援10.118.39及之後版本
EX:
<pre>1 DBOPEN("GrinderToolTable.cyc");   /
2
3 DBLOAD(0);   /
4 DBSAVE(0);   /</pre></td></tr>
</table>

<table>
<tr><th>功能</th><th>說明</th></tr>
<tr><td>INT DBNEW("File name")</td><td>功能:
• 新增並開啟全新的 Cycle 檔案
影響:
• 執行成功後，會指定此檔案為後續操作的目標檔案
引數 - File name:
• 欲新增檔案的檔名
• 限制: 字串長度最大為30個字符
回傳值:<table><tr><th>數值</th><th>意義</th></tr><tr><td>1</td><td>新增檔案成功</td></tr><tr><td>0</td><td>新增檔案失敗，非預期錯誤</td></tr><tr><td>-1</td><td>新增檔案失敗，File name 過長</td></tr><tr><td>-2</td><td>新增檔案失敗，指定路徑下已存在該檔名</td></tr></table>備註:
1. 引數型態錯誤會觸發警報 COR-023 語義錯誤
2. 此 Macro 語法不支援圖形模擬，會固定回傳 0
3. 指定路徑可能被客制 Action（CUSTOMFILE_CYCLE1~5）影響
4. 此 MACRO 語法僅會開新檔案，檔案內並無 Cycle 資料，無法
5. 同時間僅能開啟一個 Cycle 檔案，而 DBOPEN 以及 DBNEW
Ex: 先使用 DBOPEN("FileA")，再使用 DBNEW("FileB")，此時
6. 版本: 支援 10.118.56L, 10.118.60F, 10.118.66 以及以後的版本
Ex:
<pre>1 // 執行暖機健檢
2
3 #30 = DBNEW("WarmTest_20220621");
4 IF(#30 = 1) THEN
5 DBINSERT(0,"WarmTest");
6 ELSE
7 // 錯誤處理
8 END_IF;</pre></td></tr>
</table>

<table>
<tr><th>功能</th><th>說明</th></tr>
<tr><td>INT DBINSERT(Index,"Cycle Name")</td><td>功能:
• 蒐集 Cycle name 指定的資料，新增 Cycle 資料於第
影響:
• 執行成功後，指定位置以及後方的 Cycle 資料會往後
• 執行成功後，會以此 Cycle name 作為後續操作用的
引數 - Index:
• 指定 Cycle 資料要插在第幾筆
• 限制: 最大值為"目前開啟的 Cycle 檔的最後資料位置
引數 - Cycle name:
• 指定要記錄的資料格式
• 限制: 須為 Schema 中有定義的
回傳值:
<table><tr><th>數值</th><th>意義</th></tr><tr><td>1</td><td>插入 Cycle 資料成功</td></tr><tr><td>0</td><td>插入 Cycle 資料失敗，非預期錯誤</td></tr><tr><td>-1</td><td>插入 Cycle 資料失敗，Index 超過範圍</td></tr><tr><td>-2</td><td>插入 Cycle 資料失敗，Cycle 檔未開啟</td></tr><tr><td>-3</td><td>插入 Cycle 資料失敗，使用未定義的 Cycl</td></tr></table>備註:
1. 引數型態錯誤會觸發警報 COR-023 語義錯誤
2. 此 Macro 語法不支援圖形模擬，會固定回傳 0
3. 此 Macro 語法會操作目前開啟 Cycle 檔。Cycle 檔的開啟可
4. 同時間僅能指定一個 Cycle name，而 DBLOAD 以及 DBINSE
Ex:先使用 DBLOAD(0)，Cycle name 會指定為 Cycle 檔中位
5. 版本: 支援 10.118.56L, 10.118.60F, 10.118.66 以及以後的版本
Ex:
<pre>1 // DBOPEN 搭配 DBINSERT 把資料插入在最後-
2 #30 = DBOPEN("File name"); // 回
3 DBINSERT(#30,"Cycle name"); // 把</pre></td></tr>
</table>

<table>
<tr><th>功能</th><th>說明</th></tr>
<tr><td>INT DBDELETE(Index)</td><td>功能:
• 刪除第 "Index" 筆的 Cycle 資料
影響:
• 執行成功後，所有於指定位置後方的 Cycle 資料會往
引數 - Index:
• 指定要刪除第幾筆 Cycle 資料
• 限制: 最大值為"目前開啟的 cycle 檔的最後位置"，e
回傳值:<table><tr><th>數值</th><th>意義</th></tr><tr><td>1</td><td>刪除 Cycle 資料成功</td></tr><tr><td>0</td><td>刪除 Cycle 資料失敗，非預期錯誤</td></tr><tr><td>-1</td><td>刪除 Cycle 資料失敗，Index 超過範圍</td></tr><tr><td>-2</td><td>刪除 Cycle 資料失敗，Cycle 檔未開啟</td></tr></table>備註:
1. 引數型態錯誤會觸發警報 COR-023 語義錯誤
2. 此 Macro 語法不支援圖形模擬，會固定回傳 0
3. 此 Macro 語法會操作目前開啟 Cycle 檔。Cycle 檔的開啟可
4. 版本: 支援 10.118.56L, 10.118.60F, 10.118.66 以及以後的版本
Ex:<pre>1 // DBOPEN 搭配 DBDELETE 刪除最後一筆資料
2 #30 = DBOPEN("File name"); // 回
3 DBDELETE(#30 - 1); // 刪</pre></td></tr>
<tr><td>DRAWHOLE</td><td>依據刀具半徑及SETDRAW函數所定義之顏色，在目前位置畫一</td></tr>
<tr><td>EXP</td><td>計算以自然數當作基底的指數值
EX:
#1:=EXP(1); // e^1 = 2.71828
有效版本：10.116.16</td></tr>
</table>

功能	說明

FLOOR	回傳小於或等於某數值的最大整數
	EX:
	#10 := 1.4;
	#1 := FLOOR(#10); // #1 = 1
	#2 := FLOOR(1.5); // #2 = 1

GETARG	讀取呼叫者傳遞的引數
	EX:
	假設O0001主程式內容為
	G101 X30. Y40. Z1=40. Z2=50.;
	G0101擴充巨集程式使用GETARG讀取引數內容
	#1 := GETARG(X); // 將X引數內容30.存到#1
	#2 := GETARG(Z1); // 將Z1引數內容40.存到#2
	#3 := GETARG(W);
	// 因W不存在，所以#3內容為(VACANT, #0)

GETTRAPARG	讀取G66/G66.1在Trap單節內的引數內容
	EX:
	假設O0001主程式內容為
	G66 P100 X100. Y100.
	G01 X20.
	O0100副程式使用GETTRAPARG讀取引數內容
	#1 := GETARG(X);
	// 將X引數內容100.存到#1
	#2 := GETTRAPARG(X);
	// 將Trap單節X引數內容20.存到#2

LN	計算以自然數當作基底的對數值
	EX:
	#2:=LN(100); // ln100 = 4.60517
	注意事項：後方引數需為正數，否則會跳警報
	有效版本：10.116.16

MAX	決定兩輸入值的最大值
	EX:
	#10 := 1.2;
	#20 := 4.5;
	#1 := MAX(#10, #20); // #1 = 4.5
	#2 := MAX(-1.2, -4.5); // #2 = -1.2

功能	說明

MIN	決定兩輸入值的最小值
	EX:
	#10 := 1.2;
	#20 := 4.5;
	#1 := MIN(#10, #20); // #1 = 1.2
	#2 := MIN(-1.2, -4.5); // #2 = -4.5

MSG	自訂提示，詳情請參閱『MACRO自訂提示』
	EX:
	MSG(100); // 提示ID
	MSG("鑽頭移失"); // 提示顯示內容
	MSG(100, "鑽頭遺失"); // 提示ID + 顯示內容
	備註:
	1. 提示內容有字串長度限制，中文 : 19個字，英文 : 39個字。
	2. 提示ID只允許使用0~65535之間的整數，否則跳警報COR-02

功能	說明

OPEN("檔名") or
OPEN("檔名", "寫檔方式")
	在 NcFiles 資料夾(資料夾位置請參考 Pr3219)開啟一文字檔，其
	若檔案名稱為"COM"時，表示打開RS232/RS485傳輸埠，其設定
	EX:
	OPEN("COM"); // 打開傳輸埠
	PRINT("\\p"); // 輸出'%'字元
	FOR #1 = 1 TO 5000 DO
	#30 := #1 * 10.;
	PRINT("G01 X#30"); // 輸出G01 X10.0
	END_FOR;
	PRINT("\\p"); // 輸出'%'字元
	CLOSE(); // 關閉傳輸埠
	"寫檔方式"決定OPEN時，保留或清空原先的檔案內容。（有效
	(i) 指定為"a"：保留原先的檔案內容，並將新的資料接續輸出在
	EX:
	OPEN("PROBE.NC", "a");
	// 開啟並保留PROBE.NC檔案內容，準備作資料輸出
	(ii) 不指定或指定為"w"：清空原先的內容，並將新的資料重新
	EX:
	OPEN("PROBE.NC");
	// 開啟並清空PROBE.NC檔案內容，準備作資料輸出
	OPEN("PROBE.NC", "w");
	// 開啟並清空PROBE.NC檔案內容，準備作資料輸出
	(iii) 與上述兩點不同之寫檔方式：寫檔方式指定錯誤，系統會發
	EX:
	OPEN("PROBE.NC", "abc");
	// 寫檔方式指定錯誤，發出COR-301警報，無法開啟PROBE.NC
	(iv) 使用#或@變數轉成字串，小數點輸出位數會與Pr17連動（不
	(v) 使用#或@變數轉成字串，若在後方加上[*]，小數點輸出位數

PARAM	讀取系統參數的內容 EX: #1 := PARAM(3204); // 讀取Pr3204(PLC掃瞄時間)之內容

功能	說明

POP	將堆疊（STACK）裡面的資料取出，依序由最上層一路取到最
	EX:
	PUSH(5); // 將數字5塞入堆疊中
	#1 := POP(); // 取出堆疊中最上層數值(#1 = 5)

POW	計算以某數值當作基底的指定乘冪次方值 EX: #3:=POW(16,0.5); // 16^0.5 = 4 注意事項：基底不可為負值，否則會跳警報COR-122 有效版本：10.116.16

PRINT	此函數用來輸出字串，輸出字串中的變數名稱會被取代成該變
	字元"\"為逃脫字元，相關特殊字元定義如下:
	"\\": 表示"\"字元
	"\@": 表示"@"字元
	"\#": 表示"#"字元
	"\p": 表示"%"字元
	使用#或@變數轉成字串，小數點輸出位數會與Pr17連動（有效
	使用#或@變數轉成字串，若在後方加上[*]，小數點輸出位數會
	EX:
	假設Pr17=2，且為公制
	@53 = 20 ;
	#3 = 23.1234 ;
	PRINT("G01 X#3 Y@53 Z20.0") ;
	輸出結果為G01 X23.123 Y20 Z20.0 ; // 顯示到小數點後3位
	EX:
	@53 = 20 ;
	#3 = 23.1234 ;
	PRINT("G01 X#3[2] Y@53 Z20.0") ; // #3[2]代表只要小數點
	輸出結果為G01 X23.12 Y20 Z20.0 ; // 顯示到小數點後2位

PUSH	將資料塞進堆疊（STACK），最先PUSH進入控制器的數值會堆 EX: PUSH(#1); // 將變數#1塞入堆疊中
RANDOM	產生0~32767間的一個隨機數 EX: #1 := RANDOM();

功能	說明

READDI (I點編號)	以READDI、READDO指令後方小括弧內的數字，決定所讀取的I
	EX:
	@52 := READDI(31); // 把I31的值讀出來填到@52
	#88 := READDO(11); // 把O11的值讀出來填到#88
	G90 G10 L1000 P4000 R READDI(15); // 把I15的值讀出來填到R40(
	注意事項:
	1. 有效版本：10.116.23
	2. I/O點的讀取是在預解階段，但需在加工程式實際執行到RE
	3. 讀取的I/O點編號範圍僅限於0~511間，若範圍錯誤，系統將

READDO(O點編號)	

READABIT(A點編號)	以READABIT指令後方小括弧內的數字，決定所讀取的A點編號。
	EX:
	@52 := READABIT(31); // 把A31的值讀出來填到@52
	#88 := READABIT(11); // 把A11的值讀出來填到#88
	注意事項:
	1. 有效版本：10.116.44
	2. A點的讀取是在預解階段，但需在加工程式實際執行到REAI
	3. 讀取的A點編號範圍僅限於0~511間，若範圍錯誤，系統將發

READRREGBIT(R值編號,指定Bit)	以READRREGBIT指令後方小括弧內的2個數字，決定所讀取的R
	EX:
	@52 := READRREGBIT(31,3); // 把R31的第三個Bit的值讀出來填至
	注意事項:
	1. 有效版本：10.116.39
	2. R值的讀取是在預解階段，但需在加工程式實際執行到REAI
	3. R值編號若小於0或大於65535，系統將發出COR-135 R值讀寫
	4. R值編號若為不正確字元，系統將發出COR-5 程式載入失敗
	5. 指定Bit若小於0或大於31，系統將發出COR-135 R值讀寫指令
	6. 指定Bit若為不正確字元，或R值編號及指定Bit皆為不正確字

ROUND	回傳某數值完成四捨五入進位後的值
	EX:
	#10 := 1.4;
	#1 := ROUND(#10); // #1 = 1
	#2 := ROUND(1.5); // #2 = 2

功能	說明
SCANTEXT	此函數用來讀取公用變數所儲存的字串內容 將字串存入公用變數時，控制器會先進行ASCII轉碼，因此若直 EX: %@MACRO @1:="12"; // 16進位HEX=3231，10進位DEC=12849 #1:=SCANTEXT(1); OPEN("NC"); PRINT("@1"); PRINT("#1"); CLOSE(); M30; 所得結果為 @1 = 12849 #1 = 12

SETDO(O點編號, O點開或關)	以SETDO指令後方小括弧內的2個數字，決定所設定的O點編號
	EX:
	SETDO(3, 1); // 設定O3 on
	SETDO(8, 0); // 設定O8 off
	注意事項:
	1. 有效版本：10.116.23
	2. O點的寫入是在插值階段，因此在執行時可不必減速到0，
	3. PLC與SETDO應避免混用，例如MACRO中以SETDO讓O1 on，
	4. 設定的O點編號範圍僅限於0~511間，若範圍錯誤，系統將

SETABIT(A點編號, A點開或關)	以SETABIT指令後方小括弧內的2個數字，決定所設定的A點編
	EX:
	SETABIT(3, 1); // 設定A3 on
	SETABIT(8, 0); // 設定A8 off
	注意事項:
	1. 有效版本：10.116.44
	2. A點的寫入是在插值階段，因此在執行時可不必減速到0，
	3. PLC與SETABIT應避免混用，例如MACRO中以SETABIT讓A1 o
	4. 設定的A點編號範圍僅限於0~511間，若範圍錯誤，系統將

功能	說明

SETRREGBIT(R值編號, 指定Bit, 開或關)	以SETRREGBIT指令後方小括弧內的3個數字，決定所設定的R值
	EX:
	SETRREGBIT(50,3,1); // 設定R50的第3個Bit on
	SETRREGBIT(50,4,0); // 設定R50的第4個Bit off
	注意事項:
	1. 有效版本：10.116.39
	2. R值的寫入是在插值階段，因此在執行時可不必減速到0，
	3. PLC與SETRREGBIT應避免混用，例如MACRO中以SETRREGB
	4. R值編號若小於0或大於65535，系統將發出COR-135 R值讀寫
	5. 指定Bit若小於0或大於31，系統將發出COR-135 R值讀寫指令
	6. 第三個引數若不是0（off）或1（on），系統將發出COR-13
	7. 任意引數輸入不正確字元，系統將發出COR-5 程式載入失敗

<table>
<tr><th>功能</th><th>說明</th></tr>
<tr><td>SETDRAW(路徑顏色) or
SETDRAW(路徑顏色,填充顏色,刀具半徑)</td><td>定義圖形模擬的畫圖樣式:
1. 路徑顏色：設定輪廓線的顏色，可使用BGR碼，或參考模擬
2. 填充顏色：設定DRAWHOLE函數圓內填充的顏色，可使用B
3. 刀具半徑：設定DRAWHOLE函數圓半徑大小，影響包含可使
常用BGR碼如下所示:
 

<table>
<tr><th>顏色代碼</th><th>BGR碼</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>8388608</td></tr>
<tr><td>2</td><td>32768</td></tr>
<tr><td>3</td><td>8421376</td></tr>
<tr><td>4</td><td>128</td></tr>
<tr><td>5</td><td>8388736</td></tr>
<tr><td>6</td><td>32896</td></tr>
<tr><td>7</td><td>12632256</td></tr>
</table>
注意事項:
• BGR碼與RGB碼，只差在B(Blue)與R(Red)的擺放數值順序不
e.g.
RGB碼紅色為0xFF0000，轉到BGR碼則為0x0000FF = 255 (上述顏
• SETDRAW會同時設定到路徑及畫圓的顏色，若想讓路徑及畫
e.g.</td></tr>
</table>

功能	說明

	%@MACRO
	#3:=SETDRAW(#1,#2,#18);
	// 使用#3記錄原本路徑顏色，#2定義填充顏色，#18定義畫圓斗
	DRAWHOLE();
	SETDRAW(#3); // 畫圓後將路徑顏色改回
	M99;

SIGN	回傳某數值的正負號，負數為-1，正數為1，0為0
	EX:
	#10 := 4;
	#1 := SIGN(#10); // #1 = 1
	#2 := SIGN(-4); // #2 = -1
	#3 := SIGN(0); // #3 = 0

SIN	計算某數值的正弦值
	EX:
	#10 := 90;
	#1 := SIN(#10); // #1 = 1
	#2 := SIN(-90); // #2 = -1

SLEEP	暫時放棄此次巨集循環的執行權，一般配合迴圈使用（FOR、\ EX: SLEEP();

SQRT	計算某數值的平方根值
	EX:
	#10 := 4;
	#1 := SQRT(#10); // #1 = 2
	#2 := SQRT(9); // #2 = 3

功能	說明

STD(引數1,引數2)	根據Pr17將數值轉換成當時系統設定的輸入單位（Input Unit, I
	1. 引數1為欲改變單位之數值
	2. 引數2為標準化單位，一般使用#1600，而#1600設定值來自
	公制:
	EX1:
	當Pr17=2，#1600對應LIU = 0.001mm
	#9 := 100;
	#10 :=STD(#9,#1600); // #9為100個BLU，故#10為0.1mm(100*0.0
	Ex2:
	當Pr17=3，#1600對應LIU = 0.0001mm
	#9 := 100;
	#10 :=STD(#9,#1600); // #9為100個BLU，故#10為0.01mm(100*0.
	英制:
	Ex3:
	當Pr17=2，#1600對應LIU = 0.0001inch
	#9 := 100;
	#10 :=STD(#9,#1600); // #9為100個BLU，故#10為0.01inch(100*0.

STDAX(引數1,引數2)	將數值轉成對應軸向的標準單位 引數1為變數，引數2為對應軸向的名稱 EX: #24 := STDAX(#24,X); #3 := STDAX(#3,A);

STKTOP	將堆疊（STACK）裡面的資料複製出來
	EX:
	PUSH(5); // 將數字5塞入堆疊中
	PUSH(6); // 將數字6塞入堆疊中
	PUSH(7); // 將數字7塞入堆疊中
	#1 := STKTOP[0]; // #1 = 7
	#2 := STKTOP[1]; // #2 = 6
	#3 := STKTOP[2]; // #3 = 5

功能	說明
SYSVAR(軸群識別碼,系統變數碼)	讀取特定軸群中的的系統變數 軸群識別碼：1為第一軸群組，2為第二軸群組，以此類推。 系統變數碼：欲讀取的系統變數編號 EX: #1 := SYSVAR(1, 1000); // 讀取第一軸群的系統變數#1000（插值模式）

TAN	計算某數值的正切值
	EX:
	#10 := 45;
	#1 := TAN(#10); // #1 = 1
	#2 := TAN(-45); // #2 = -1

WAIT	系統停止預解，直到WAIT之前的指令執行完畢
	EX:
	%@MACRO
	@50:=1; // @50內容填成1
	G90 G01 X100. F1000; // 假設此時Reset系統
	WAIT();
	@50:=0; // @50內容填成0
	M30;
	假設在G01尚未執行完畢時重置系統，由於下一個單節

CHKMN("機械廠代碼")	檢查機械廠代碼，1：一致，0：不符
	EX:
	%@MACRO
	#51 := CHKMN("5566"); //#51之值為檢查結果
	IF #51=0 THEN //若機械廠代碼不符則發出警報
	ALARM(501, "The manufacturer code is invalid.");
	END_IF;
	目標版本：10.116.6A

功能	說明

CHKSN("序號")	檢查控制器序號，1：一致，0：不符
	EX:
	%@MACRO
	#52 := CHKSN("M9A0001"); //#52之值為檢查結果
	IF #52=0 THEN //若序號不符則發出警報
	ALARM(502, "The serial number is invalid.");
	END_IF;
	目標版本：10.116.6A

CHKMT("機床屬性")	檢查機床屬性，1：一致，0：不符
	EX:
	%@MACRO
	#53 := CHKMT("MILL"); //#53之值為檢查結果
	IF #53=0 THEN //若機床屬性不符則發出警報
	ALARM(503, "The machine type is invalid.");
	END_IF;
	目標版本：10.116.6A

CHKMI("機型")	檢查控制器機型，1：一致，0：不符
	除了SUPER系列需輸入S，其他機型皆按照實際型號輸入，如1
	EX:
	%@MACRO
	#54 := CHKMI("S"); //#54之值為檢查結果
	IF #54=0 THEN //若機型不符則發出警報
	ALARM(504, "The hardware type is invalid.");
	END_IF;
	目標版本：10.116.6A

功能	說明

CHKINF(類別編號, "代碼")	檢查代碼與類別編號對應的內容是否一致，一致：1 不一致：0
	類別編號1~5分別為:
	1. 機械廠代碼
	2. 序號
	3. 機床屬性
	4. 機型
	5. 專機代碼
	機型代碼除了Super系列簡寫成s之外，其余系列皆維持相同ex.
	EX:
	%@MACRO
	#51 := CHKINF(1, "5566"); //#51之值為檢查結果
	IF #51=0 THEN //若機械廠代碼不符則發出警報
	ALARM(501, "The manufacturer code is invalid.");
	END_IF;
	#52 := CHKINF(2, "M9A0001"); //#52之值為檢查結果
	IF #52=0 THEN //若序號不符則發出警報
	ALARM(502, "The serial number is invalid.");
	END_IF;
	#53 := CHKINF(3, "MILL"); //#53之值為檢查結果
	IF #53=0 THEN //若機床屬性不符則發出警報
	ALARM(503, "The machine type is invalid.");
	END_IF;
	#54 := CHKINF(4, "S"); //#54之值為檢查結果
	IF #54=0 THEN //若機型不符則發出警報
	ALARM(504, "The hardware type is invalid.");
	END_IF;
	#55 := CHKINF(5, "10"); //#55之值為檢查結果
	IF #55=0 THEN //若專機代碼不符則發出警報
	ALARM(505, "The industrial machine ID is invalid.");
	END_IF;
	參數型態需正確，且類別編號不可超出範圍，否則會跳警報CC
	EX:
	%@MACRO
	#51 := CHKINF(60, "Mill"); //檢查編號超出範圍
	#51 := CHKINF("2", "Mill"); //參數一型態不正確
	#53 := CHKINF(5, 12345); //參數二型態不正確
	目標版本：10.118.22M、10.118.28B、10.118.30

功能	說明

STR2INT("字串")	轉換數字字串為整數型態
	EX1:
	%@MACRO
	@1:="5555";
	#1:= SCANTEXT(1); // #1 = 字串5555
	#2:= STR2INT("#1"); // #2 = 5555
	EX2:
	%@MACRO
	#1:=STR2INT("100"); // #1 = 100
	EX3:
	%@MACRO
	#1:=STR2INT("123.456"); // #1 = 123
	注意事項：只要有文字的部分，即為不合法之輸入

SYSDATA(系統診斷變數號碼)	讀取系統診斷變數
	範例:
	// 假設要取得系統診斷變數 D336、D77
	WAIT(); // 擋預解，否則無法取得當前最新的數值
	#1 := SYSDATA(336); // 軸卡交換時間(D336)
	#2 := SYSDATA(77); // 硬體剩餘記憶體(D77)
	OPEN("DbgData.txt", "a"); // 開檔，檔名為"DbgData.txt"
	PRINT("#1 #2"); // 將變數值輸出到檔案內
	CLOSE(); // 關檔
	注意事項:
	1. 有效版本：10.118.23U, 10.118.28H、10.118.33
	2. 執行函數前建議下 WAIT() 函數擋預解，以取得執行當前的
	3. 引數格式須為整數，否則會導致函數無法正常執行
	4. 變數號碼須在診斷變數號碼範圍內，否則發 COR-016【不合法
	錯誤範例:
	SYSDATA("77"); // 引數須為整數，發 COR-023 警報
	SYSDATA(77.0); // 引數須為整數，發 COR-023 警報
	SYSDATA(D77); // 引數須為整數，此輸入導致語法錯誤，
	SYSDATA(10000); // 變數號碼超出診斷變數數值範圍，發

功能	說明

DRVDATA(站號, 10進位格式變數號碼)
DRVDATA(站號, "16進位格式變數號碼")
	讀取驅動器狀態變數
	變數號碼有10進位與16進位兩種輸入方式
	10進位：將變數號碼轉成10進位
	16進位：格式為 "xxxh", x=0~F，在字串內輸入16進位的編號(3
	範例：
	// 假設要取得第一軸(站號1000)速度命令(Pn-D26)與第一
	WAIT(); // 擋預解，否則無法取得當前最新的數值
	#1 := DRVDATA(1000, 3366); // 第一軸速度命令(Pn-D26，
	#2 := DRVDATA(1003, "D61h"); // 第一主軸編碼器內部溫度
	OPEN("DbgData.txt", "a"); // 開檔，檔名為"DbgData.txt"
	PRINT("Pn-D26: #1, Pn-D61: #2"); // 將變數值輸出到檔案內
	CLOSE(); // 關檔
	DbgData.txt 可能的輸出為
	Pn-D26: 150, Pn-D61: 430
	注意事項：
	1. 有效版本：10.118.23U, 10.118.28I、10.118.34
	2. 執行函數前建議下 WAIT() 函數擋預解，以取得執行當前的
	3. 由於通訊限制，驅動器診斷變數並非及時從驅動器讀取上
	4. 第一引數格式須為整數，否則發 COR-023 **【語義錯誤】**
	5. 第二引數如為字串需為16進位格式("xxxh", x=0~F)，否則發
	6. 第二引數需以10進位整數/16進位字串表示，否則發 COM-0
	7. 變數號碼須在驅動器狀態變數號碼範圍內，且控制器支援讀取
	8. 如使用的驅動器不支援該狀態變數號碼，但控制器支援讀
	9. 站號無對應的驅動器，或使用非新代 M3 驅動器，函數回傳
	10. 由於狀態變數在開機時需要一段時間進行讀取，建議於開
	錯誤範例：
	// 發 COM-3 警報
	DRVDATA(1003, D61h); // 第二引數需以10進位整數/16進
	// 發 COR-023 警報
	DRVDATA("1003", 3425); // 站號數值必須為整數，發 CO
	DRVDATA(1003, "G21h"); // 第二引數如為字串需為16進位
	DRVDATA(1003, "3425"); // 第二引數如為字串需為16進位
	DRVDATA(1003, "0D61H"); // 第二引數如為字串需為16進
	// 發 COR-016 警報
	DRVDATA(1003, "DFFh"); // 驅動器不支援此變數號碼，發
	// 回傳 VACANT
	DRVDATA(9999, "D61h"); // 站號無對應的驅動器，回傳 V
	DRVDATA(9999, "DFFh"); // 站號無對應的驅動器，不檢查

8 副程式呼叫

8.1 呼叫方式

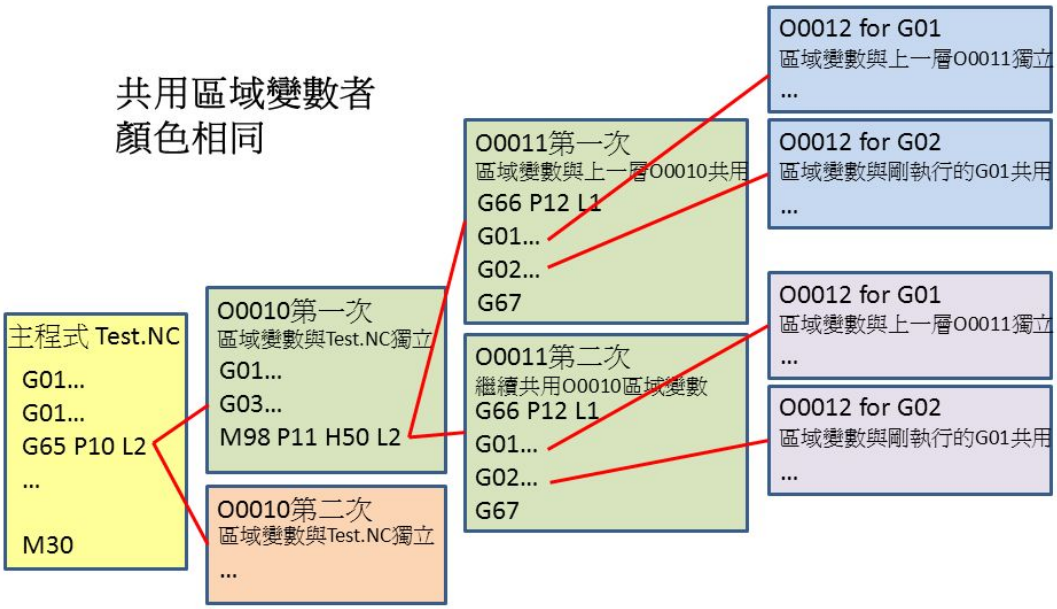
語法	說明	呼叫類型	區域變數	範例
M98 P_ H_ L_	呼叫副程式 P_ 副程式名稱 H_ 起始序號 L_ 重覆次數	子程式	繼承主程式之區域變數#1~#400	M98 P10 L2; 說明: 呼叫O0010兩次
M198 P_ H_ L_ （當M198未被Pr3601~登錄時）	呼叫副程式 P_ 副程式名稱 H_ 起始序號 L_ 重覆次數	子程式	繼承主程式之區域變數#1~#400	M198 P10 L2; 說明: 呼叫O0010兩次
G65 P_ L_	呼叫單一巨集程式 P_ 副程式名稱 L_ 重覆次數	巨集程式	使用獨立於主程式之區域變數#1~#400，並於#1~#26記載呼叫時所帶之引數	G65 P10 L2 X10.0 Y10.0; 說明: 呼叫O0010兩次， 並輸入引數。
G66 P_ L_	使用移動指令來呼叫模式巨集程式 P_ 副程式名稱 L_ 重覆次數	模式巨集程式	每次呼叫G66時會創建一塊獨立的#1~#400，這塊區域變數會被共用直到執行到G67，執行完G67後，此塊區域變數會被回收清除。 但請注意，這塊區域變數只在G66 P引數指定到的副程式內共用，並不與呼叫G66、G66.1的該層程式的區域變數共用。	G66 P10 X10.0 Y10.0; **X20.** **Y20.** 說明: X20.與Y20.移動指令會呼叫O0010，並輸入引數X10.0、Y10.0。
G66.1 P_ L_	每個單節皆會呼叫模式巨集程式 P_ 副程式名稱 L_ 重覆次數	模式巨集程式	與G66相同	G66.1 P10 X10.0 **X20.** **G04 X2.;** **M30;** 說明: 每一單節指令都會呼叫O0010，並輸入引數X10.0。

G_ L_	呼叫擴充G碼 L_ 重覆次數	巨集程式	每次呼叫都會創建一塊新的區域變數 #1~#400，於該Macro結束時恢復上一層的區域變數。	G128 L3 X1.0; 說明: 呼叫G0128三次
G_	呼叫客制G碼（G00、G01、G02、G03、G53、G40、G41、G42）使用前必須在Pr3701~登錄。	巨集程式	每次呼叫都會創建一塊新的區域變數 #1~#400，於該Macro結束時恢復上一層的區域變數。	G01 A_ B_ C_; 說明: 呼叫客制G01
T_	呼叫副程式T0000來換刀， 在副程式內的T碼為普通T碼，不會再呼叫T0000。	若Pr3215 = 1，則為子程式	繼承主程式之區域變數#1~#400	T3; 說明: 呼叫T0000
		若Pr3215 = 2，則為巨集程式	每次呼叫都會創建一塊新的區域變數 #1~#400，於該Macro結束時恢復上一層的區域變數。	
M_	呼叫M碼巨集程式， 在巨集內的M碼為普通M碼，不會再呼叫M碼巨集使用前必須在Pr3601~登錄。	巨集程式	每次呼叫都會創建一塊新的區域變數 #1~#400，於該Macro結束時恢復上一層的區域變數。	M13 A_ B_ C_; 說明: 呼叫M0013之巨集程式。

備註:

- 上表內之L引數未指定時，系統內定預設值為1。
- 上表內呼叫的副程式之區域變數（#1~#400），其生命週期請參考Macro變數規格。

變數生命週期之範例:



8.2 返回方式

語法	說明	範例
M99	返回主程式	M99;
M99 P_	返回主程式特定序號，P_：欲到達的**序號**編號	M99 P100; 回到主程式序號N100位置
M99 Q_	返回主程式特定行號，Q_：欲到達的**行號**編號	M99 Q100; 回到主程式行號100行位置
G67	取消G66	G67;

9 變數規格

- 有關#及@的變數規格說明，請參考Macro變數規格
- 將不同型態的變數(Long和Double)一起做四則運算，會強制轉型為Double再做計算，例如

```
@1 := 1;         // 以Long型態指派變數 @1
@2 := 2.0;       // 以Double型態指派變數 @2
@3 := @1 + @2;   // 得到結果為"@3 := 3.0"型態為Double

@4 := 2;         // 以Long型態指派變數 @4
@5 := 100;       // 以Long型態指派變數 @5
@6 := @4 * @5;   // 得到結果為"@6 := 200"型態仍為Long
```

10 MACRO自訂警報

10.1 MACRO警報觸發語法

```
%@MACRO
ALARM(xxx);// xxx為警報編號
M30;
```

10.2 DOS系統警報內容編輯說明

- 檔案路徑:
  -> 繁簡中文: C:\\CNC\\EXE\\APPDATA.RES\\CNCCHI.STR
  -> 英文版: C:\\CNC\\EXE\\APPDATA.RES\\CNCENG.STR
  -> 其他: C:\\CNC\\EXE\\APPDATA.RES\\CNCLOC.STR
- 內容格式: **24xxx="1;MSG=警報內容"**，其中xxx為警報編號，請選擇未使用過的號碼作為自訂警報編號，並請注意識別代號為**24**。
- 範例:
  -> CNCCHI.STR:
  24003="1;MSG=**最大圓弧弦長小於或等於0**"

  -> CNCENGSTR:
  24003="1;MSG= **max arc length can not be negative**"

10.3 WinCE系統警報內容編輯說明

- 檔案路徑:
  -> 中文版: DiskC\\OCRes\\CHT\\String\\AlarmMacro_CHT.Xml
  -> 英文版: DiskC\\OCRes\\Common\\String\\AlarmMacro_Com.Xml
  -> 通則: DiskC\\OCRes{color:#0000ff}**L**\\String\\AlarmMacro_**L**.Xml，**其中L為各語系之名稱**
- 檔案格式: <Message ID="AlarmMsg::**Macro**::ID=xxx" Content="**警報內容**" />，其中xxx為警報編號，請選擇未使用過的號碼作為自訂警報編號，並請注意識別字母為**Macro**。警報內容的字串長度可容納48個英文字，或是31個中文字，多餘的字串會超出警報視窗。
- 範例:
  -> CusMacroAlarmMsg_CHT.Xml:
  <Message ID="AlarmMsg::**Macro**::ID=3" Content="**最大圓弧弦長小於或等於0**" />

  -> CusMacroAlarmMsg_Common.Xml:
  <Message ID="AlarmMsg::**Macro**::ID=3" Content="**max arc length can not be negative**" />

10.4 透過SI（SyntecIDE）進行警報字串編輯

MACRO警報字串編輯功能已整合到SI內，相關操作說明請參考Macro警報字串編輯器

11 MACRO自訂提示（MSG）

11.1 MSG規格說明

- MACRO警報發生時，必須重置系統才可消除警報；而MSG自訂提示僅需按下"ESC"即可消除，可用於單純提示狀態，但須注意當程式結束時，提示也會自動消失。
- 提示內容有字串長度限制，中文 : 19個字，英文 : 39個字。
- 提示ID只允許使用0~65535之間的整數，否則跳警報COR-023【語意錯誤】。

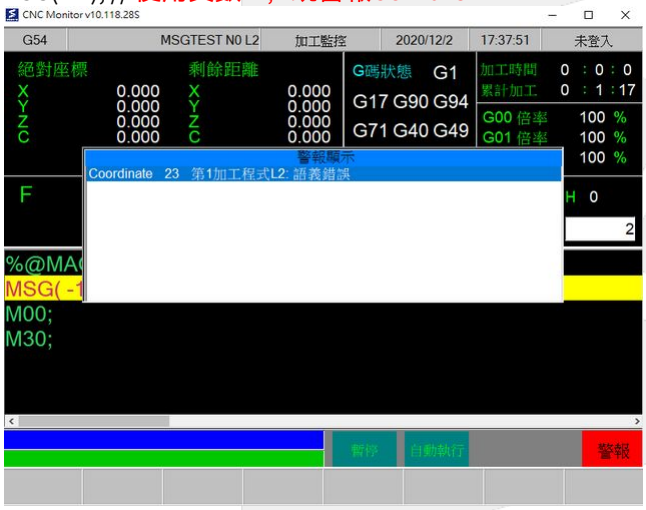
11.2 MSG觸發語法

- MSG(100); //警報ID

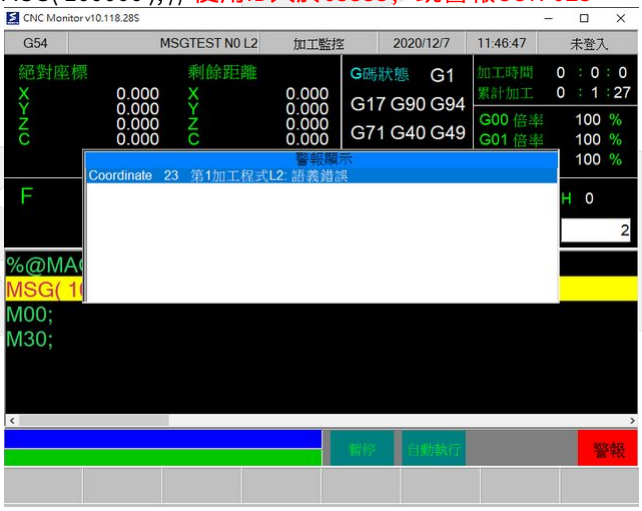
- MSG("鑽頭移失");//警報顯示內容

- MSG(100, "鑽頭遺失");//警報ID+顯示內容

- MSG(-1); // 使用負數ID，跳警報COR-023



- MSG(100000); // 使用ID大於65535，跳警報COR-023



12 附錄

12.1 巨集使用說明

12.1.1 引言

- 控制器內建的基本G、T、M碼可能無法滿足各產業應用之需求，對此，新代科技提供客制巨集的功能，讓開發人員可以根據其機臺之特性及需求，開發對應的巨集程序，提供機臺專用之特殊動作，提升機臺價值。
- 在開始介紹前，先針對主程序中，呼叫其他程序的方式進行簡單基本定義
    - 呼叫副程序：執行呼叫之程序，不可讀取占用引數。
    - 呼叫巨集：執行呼叫之程序，可以讀取占用引數。
    - 引數的定義請參閱 引數說明
- 後續章節將介紹巨集程序的相關規格，呼叫副程序的規格將不贅述。

12.1.2 巨集分類

Unable to render include or excerpt-include. Could not retrieve page.

12.1.3 巨集運作流程說明

Unable to render include or excerpt-include. Could not retrieve page.

12.1.4 引數用途簡介

- 巨集可以執行開發人員設計的動作，從簡單的狀態轉換，到復雜的多工序動作。
- 但如果一個巨集只能執行一個單一的流程，那開發人員為了各種狀況就需要撰寫數不清的巨集。
- 比如某一個巨集可以切削出10公分的正方形，但如果想要切削20公分的正方形時，卻需要再撰寫一個新的巨集。
- 這樣的巨集太過呆板，此時就可以透過引數，對巨集的執行內容進行變數調整，例如正方形的邊長，讓該巨集具有更高的彈性。
- 參考下面的范例
- 范例
    - 主程序中為了切削出10公分的正方形，所以下了G200
    - 主程序中為了切削出20公分的正方形，所以下了G201
    - 主程序為了切削出任意大小的正方形，所以下了G202，大小邊長由G202的引數X來決定。

- **范例_Main**

```
// Example002_Main
G90 G00 X0. Y0.;
G200;
G201;
G202 X30.;
M30;
```

范例_G0200

```
// G0200
%@MACRO
#101:=#1000;
#102:=#1004;
G91 G01 X10.;
G91 G01 Y10.;
G91 G01 X-10.;
G91 G01 Y-10.;
G#101;
G#102;
M99;
```

范例_G0201

```
// G0201
%@MACRO
#101:=#1000;
#102:=#1004;
G91 G01 X20.;
G91 G01 Y20.;
G91 G01 X-20.;
G91 G01 Y-20.;
G#101;
G#102;
M99;
```

范例_G0202

```
// G0202
%@MACRO
#101:=#1000;
#102:=#1004;
#103:=#24;
G91 G01 X#103;
G91 G01 Y#103;
G91 G01 X-#103;
G91 G01 Y-#103;
G#101;
G#102;
M99;
```

12.1.5 引數說明

- 引數乃是采用英文26個字母，除了G/N/O以外，每一個字母都有一個對應的程式變數(#值)，如下表所示。

引數	#		引數	#		引數	#	
A	#1		J	#5		S	#19	
B	#2		K	#6		T	#20	
C	#3		L	#12		U	#21	
D	#7		M	#13		V	#22	
E	#8		N			W	#23	
F	#9		O			X	#24	
G			P	#16		Y	#25	
H	#11		Q	#17		Z	#26	
I	#4		R	#18				

說明				
分類	軸向引數	條件引數	特殊引數	例外引數

- 引數會被巨集讀取占用。
- 「讀取」表示該巨集內有撰寫到該引數對應的程式變數#值，故該巨集讀取該引數進行運算。
- 「占用」表示該引數被讀取後，無法再被其他巨集讀取，稱之為引數該巨集占用。
- 引數被讀取之後不代表就會被占用，要看讀取的巨集特性。請參閱後續章節【巨集解譯處理順序】【巨集特性】。
- 引數沒有被讀取也不代表不會被占用，要看引數的特性。請參閱此章節後續說明。
- 同一行內有重復的引數，并不會發生警報。
- 同一行內有重復的引數，只有最後一個引數的數值會被讀取。
- 可以理解為重復的引數均是對同一個程式變數#值填值，後面的引數數值蓋掉前面的引數數值。
- 引數基本上可以分為以下幾類

<table>
<tr><th>類型</th><th>引數</th><th>特點</th><th>其他說明</th></tr>
<tr><td>軸向引數</td><td>X Y Z A B C I J K U V W</td><td>只要其中一個引數被某一個巨集占用，則其他軸向引數均會一起被該巨集占用</td><td>• B碼受到【Pr3806 啟動第二輔助碼B碼】影響
<table>
<tr><th>Pr3806</th><th>類型</th><th>說明</th></tr>
<tr><td>0</td><td>軸向引數</td><td></td></tr>
<tr><td>1</td><td>B碼輔助碼</td><td>會將B碼數值填到R5</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>類型</th><th>引數</th><th>特點</th><th>其他說明</th></tr>
<tr><td>條件引數</td><td>F S T D E H P Q R M B</td><td>相較於軸向引數，條件引數彼此都是獨立的，不會因為任一引數被占用，就一起被該巨集占用</td><td>
• T碼受到【Pr3215 選刀時呼叫模式】影響
<table>
<tr><th>Pr3215</th><th>類型</th><th>說明</th></tr>
<tr><td>0</td><td>T碼輔助碼，不呼叫T000</td><td>僅將T碼數值填到#1036及對應的R值
(各軸群對應的R值不同)</td></tr>
<tr><td>1</td><td>透過副程式呼叫T0000</td><td>不接受任何引數，不算是巨集。</td></tr>
<tr><td>2</td><td>透過巨集呼叫T0000</td><td>可接受其他引數。</td></tr>
</table>
• M碼受到【Pr3601~3610 登錄M碼呼叫巨集】影響
<table>
<tr><th>Pr3601~3610</th><th>類型</th><th>說明</th></tr>
<tr><td>沒有登錄的M碼</td><td>M碼輔助碼</td><td>僅將M碼數值填到#1038及R值
(各軸群對應的R值不同)</td></tr>
<tr><td>有登錄的M碼</td><td>M碼巨集</td><td>可接受其他引數</td></tr>
</table>
• B碼受到【Pr3806 啟動第二輔助碼B碼】影響
</td></tr>
</table>

<table>
<tr><th>類型</th><th>引數</th><th>特點</th><th>其他說明</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Pr3806</th><th>類型</th><th>說明</th></tr>
<tr><td>0</td><td>軸向引數</td><td></td></tr>
<tr><td>1</td><td>B碼輔助碼</td><td>會將B碼數值填到R5</td></tr>
</table>
</td></tr>
<tr><td>特殊引數</td><td>L</td><td>L引數乃是設定該巨集的連續執行次數。
• 例如：G200 L10，表示G200會連續執行10次。
• 就算呼叫巨集時沒有提供L引數，系統也都會自動補上L值=1，讓該巨集執行一次。</td><td>• T碼巨集不會讀取L碼，則不論L碼數值多少，T碼巨集永遠只會執行一次。</td></tr>
<tr><td>例外引數</td><td>G N O</td><td>這三個字母乃是控制器語法中的關鍵字，無法被當做引數使用，所以沒有對應的#值。</td><td>
<table>
<tr><th>類型</th><th>說明</th></tr>
<tr><td>G</td><td>乃是用來作為G碼指令或者G碼巨集使用</td></tr>
<tr><td>N</td><td>乃是用來作為程序旗標使用</td></tr>
<tr><td>O</td><td>乃是一般加工程序的檔名開頭</td></tr>
</table>
</td></tr>
</table>

12.1.6 巨集解譯處理順序

- 在說明宏程序讀取占用引數之前，先說明宏程序解譯處理順序。
- 加工程序會從第一行開始依序往下解譯，如果每一行都只撰寫一個宏程序/指令，則不會發生引數搶占的問題。
- 當同一行內下達了多種宏程序或指令，系統就會依照解譯處理順序來處理引數，以下說明「解譯處理順序」。

	類型	細項	舉例
1	部分G碼	G碼宏程序	G73、G84等
		模態G碼	G15、G17、G70等
		非模態調用宏程序G碼	G65
		以上各項根據撰寫順序（由左至右），先解譯者先取得引數。	
2	宏程序	M碼宏程序	
		T碼宏程序	
		以上各項根據撰寫順序（由左至右），先解譯者先取得引數。	
3	S碼		
4	F碼		
5	H碼		
6	D碼		
7	T碼		
8	M碼		
9	B碼		
10	功能G碼		G04、G51、G68等
11	插值G碼		G00、G01等

- 通常解譯處理順序同時也代表著引數占用的順序，表示「先解譯處理的宏程序，優先讀取占用引數」；但并非絕對成立，還要根據該宏程序之特性而定(部分宏程序有先解譯后執行的特性，將不滿足先解譯先占用的規則)。
- 舉例來說:
  1. 功能G碼和插值G碼有在解譯處理順序上有先后，因此功能G碼會先于插值G碼占用軸向引數;
     無論撰寫的先后，在同一單節執行時，功能G碼會優先占用軸向引數。

2. 從解譯處理順序中可以發現，M碼宏程序及T碼宏程序兩者的順序等級都是Lv2，
所以兩者如果寫在同一行，解譯處理的順序是根據撰寫的先後，寫在前面的會先被解譯處理。

12.1.7 巨集特性

Unable to render include or excerpt-include. Could not retrieve page.

12.1.8 巨集呼叫範例

Unable to render include or excerpt-include. Could not retrieve page.

01_引數與程式變數（區域變數）

- 除了例外引數以外，所有的引數都對應到一個程式變數#值。

- **Main**

```
1  // Main
2  G205 A1 B2 C3 D7 E8 F9 H11 I4 J5 K6 L12 M13 P16 Q17 R18 S19 T20 U21
   V22 W23 X24 Y25 Z26;
3  M30;
```

G0200

```
1  // G0200
2  %@MACRO
3
4  //軸向引數
5  @101 := #1;                    // A
6  @102 := #2;                    // B
7  @103 := #3;                    // C
8  @104 := #4;                    // I
9  @105 := #5;                    // J
10 @106 := #6;                    // K
11 @121 := #21;                   // U
12 @122 := #22;                   // V
13 @123 := #23;                   // W
14 @124 := #24;                   // X
15 @125 := #25;                   // Y
16 @126 := #26;                   // Z
17
18 //條件引數
19 @107 := #7;                    // D
20 @108 := #8;                    // E
21 @109 := #9;                    // F
22 @111 := #11;                   // H
23 @113 := #13;                   // M
24 @116 := #16;                   // P
```

```
@117 := #17;                    // Q
@118 := #18;                    // R
@119 := #19;                    // S
@120 := #20;                    // T

//特殊引數
@112 := #12;                    // L

M00;                            // 切換畫面到【診斷功能】→【共用變數】及
【程式變數】
WAIT();
M99;
```

02_引數重複撰寫

- 重復的引數只有最後一個引數會被讀取
- 可以理解為重復的引數均是對同一個程式變數#值填值，後面的引數數值會蓋掉前面的引數數值。

Main

```
// Main
G01 X0.;
G200 X10. X-10.;    // X引數寫了兩次，只有X-10.會被G200讀取
                    // 可以視為#24先被X10.填成10.後
                    // 又被X-10.填成-10.
                    // G200執行完畢後，執行G01
                    // 因為此行的引數均已被占用
                    // 所以G01讀取不到任何引數。
M30;
```

G0200

```
// G0200
%@MACRO
#100 := #1000;      // 備份插值模式
#101 := #24;        // 讀取到的#24是-10.
G01 X#101;          // 移動到X-10.
M00;
WAIT();
G#100;              // 還原插值模式
M99;
```

03_軸向引數與條件引數

- 軸向引數只要其中一個被占用，則其他軸向引數都會一起被占用。
- 條件引數各自獨立，不會互相影響。

Main

```
// Main
G01 X0. Y0. Z0.;
G200 X10. Y20. Z30.;                // 雖然G200只有讀取X引數
                                    // 但Y/Z引數也一起被占用
                                    // G200執行完畢後，會執行G01
                                    // 因為此行的引數均已被占用
                                    // 所以G01讀取不到任何引數。
G01 X0. Y0. Z0. F100.;
G201 P20. X10. Y20. Z30. F200.;     // G201只有讀取P引數
                                    // 所以XYZF引數都還沒有被占用
                                    // G201執行完畢後，執行G01
                                    // G01會讀取到X10. Y20. Z30. F200.
                                    // 并執行對應動作
M30;
```

G0200

```
// G0200
%@MACRO
@1:=#24;            // 只有讀取X引數。
M00;
WAIT();
M99;
```

G0201

```
// G0201
%@MACRO
@1:=#16;            // 只有讀取P引數。
M00;
WAIT();
M99;
```

04_H碼作為條件引數

- H碼作為條件引數。

Main

```
// Main
```

```
G01 X0. Y0. Z0.;
G200 X10. Y20. Z30. H40.;            // G200只有讀取H引數
                                     // H引數為條件引數
                                     // G200執行完畢後，執行G01
                                     // G01會讀取占用X10. Y20. Z30.
                                     // 并執行對應動作
M30;
```

G0200

```
// G0200
%@MACRO
#101 := #11;      // 只有讀取H引數。
M00;
WAIT();
M99;
```

05_H碼作為軸向引數

- 【Pr3809 *UVWH為XYZC軸增量指令】設定為1，H碼除了是條件引數外，**同時**也被視為軸向引數。
- 但在MACRO中讀取H引數，僅視為條件引數被占用；因此在G碼巨集執行完畢后，H引數會再次作為軸向引數被占用，并執行對應動作。

Main

```
// Main
G01 X0. Y0. Z0.;
G200 X10. Y20. Z30. H40.;            // G200只有讀取H引數
                                     // 因為Pr3809=1，所以H引數同時被視為軸向引數及條件引數
                                     // 但在MACRO中其僅視為條件引數被占用
                                     // G200執行完畢後，執行G01
                                     // G01會讀取占用X10. Y20. Z30. H40.
                                     // 并執行對應動作
                                     // H40.是因被視為軸向引數的部分還未被占用，而再次被G01作為軸向引數占用
M30;
```

G0200

```
// G0200
%@MACRO
#101 := #11;      // 只有讀取H引數。
M00;
```

```
WAIT();
M99;
```

- 在MACRO中讀取其他軸向引數，軸向引數只要其中一個被占用，則其他軸向引數都會一起被占用。
- 所以，H引數作為軸向引數的部分，同樣也會視為被占用。

Main

```
// Main
G01 X0. Y0. Z0.;
G200 X10. Y20. Z30. H40.;            // G200只有讀取X引數
                                     // 因為Pr3809=1，所以H引數同時被視為軸向引數及條件引數
                                     // 作為軸向引數的H引數也一起被占用
                                     // G200執行完畢後，執行G01
                                     // 因為此行的軸向引數均已被占用
                                     // 所以G01讀取不到任何引數
                                     // 這同樣也包括H引數作為軸向引數的部分，也不會被讀取到
M30;
```

G0200

```
// G0200
%@MACRO
#101 := #24;      // 只有讀取X引數。
M00;
WAIT();
M99;
```

06_B碼作為軸向引數

B碼作為軸向引數

- 【Pr3806 啟動第二輔助碼B碼】設定為0，B碼是軸向引數。

Main

```
// Main
G01 X0. Y0. Z0.;
G200 X10. Y20. Z30. B40.;            // G200只有讀取B引數
                                     // 因為Pr3806=0，所以B引數被判定為軸向引數
                                     // XYZ引數將被G200占用
                                     // G200執行完畢後，執行G01
                                     // 因為此行的引數均已被占用
```

```
                                        // 所以G01讀取不到任何引數
M30;
```

G0200

```
// G0200
%@MACRO
#101 := #2;      // 只有讀取B引數。
M00;
WAIT();
M99;
```

07_B碼作為條件引數

【Pr3806 啟動第二輔助碼B碼】設定為1，B碼是條件引數。

Main

```
// Main
G01 X0. Y0. Z0.;
G200 X10. Y20. Z30. B40.;          // G200只有讀取B引數
                                   // 因為Pr3806=1，所以B引數被判定為條件引數
                                   // XYZ引數將不會被G200占用
                                   // G200執行完畢後，執行G01
                                   // G01會讀取占用X10. Y20. Z30.
                                   // 并執行對應動作
M30;
```

G0200

```
// G0200
%@MACRO
#101 := #2;      // 只有讀取B引數。
M00;
WAIT();
M99;
```

08_多個G碼巨集在同一行

Unable to render include or excerpt-include. Could not retrieve page.

09_G碼指令與G碼巨集在同一行

Unable to render include or excerpt-include. Could not retrieve page.

10_T碼巨集

Unable to render include or excerpt-include. Could not retrieve page.

11_G碼巨集與T碼巨集在同一行

Unable to render include or excerpt-include. Could not retrieve page.

12_T碼巨集不受L引數影響

Unable to render include or excerpt-include. Could not retrieve page.

13_多個T碼巨集在同一行

Unable to render include or excerpt-include. Could not retrieve page.

14_M碼巨集

Unable to render include or excerpt-include. Could not retrieve page.

15_G碼巨集與M碼巨集在同一行

Unable to render include or excerpt-include. Could not retrieve page.

16_G碼巨集與M碼巨集在同一行，G碼巨集讀取佔用軸向引數

Unable to render include or excerpt-include. Could not retrieve page.

17_多個M碼巨集在同一行

Unable to render include or excerpt-include. Could not retrieve page.

18_T碼巨集與M碼巨集在同一行

Unable to render include or excerpt-include. Could not retrieve page.

19_非模態呼叫巨集（G65）

Unable to render include or excerpt-include. Could not retrieve page.

20_模態呼叫巨集（G66）

Unable to render include or excerpt-include. Could not retrieve page.

21_非模態呼叫巨集（G66.1）

Unable to render include or excerpt-include. Could not retrieve page.

22_G65及G66/G66.1必須是該行最後一個G碼

- 非模態呼叫巨集G碼(G65)以及模態呼叫巨集G碼(G66/G66.1)必須是該行最後一個G碼。

- **Main**

```
// Main
G01 X0. Y0. Z0.;
G65 P1 X10. Y20. Z30. G200; // 此行會發出警報COR-013
                            // 因為G65和G200在同一行
                            // 但G65不是最後一個G碼
M30;
```

G0200

```
// G0200
%@MACRO
#101 := #24;
M00;
WAIT();
M99;
```

O0001

```
// O0001
%@MACRO
#101 := #24;
#102 := #25;
#103 := #26;
M00;
WAIT();
M99;
```

12.2 MACRO XML資料應用

- MACRO可以使用特殊的函數讀取xml檔案，分別是DBLOAD和DBOPEN，DBOPEN用來載入xml資料檔案，DBLOAD用來讀取資料內容。
- 應用範例：下圖為某產機客制人機畫面，該畫面將自行產生一xml檔案，以記錄相關加工資料，之後在巨集規劃動程時，同步讀取該xml內容，以作為規劃依據。

尺寸編輯	模式關閉	前座外徑	12.98	線徑	1.80	方向	右旋	材質	SW-A	間隙	直軸	切刀	一般

No	X 間隙	Y 送線	Z 右旋	A 上切	B 左旋	C 下切	R	K	T	起始速度
1	0.00	17.63	12.98	267.54						
2	2.26	21.34	13.20	300.87						
3	2.26	91.19	13.20	443.29						
4	0.00	21.55	12.98	116.95						
5	-6.05	21.16	12.98	150.00						

-> 該客制人機首先將使用者設定內容輸出成xml檔案，其語法格式定義如下，並且將該xml檔案存放於使用者所指定的GNCFILES路徑中（參閱Pr3219說明）：

```
<?xml version="1.0" encoding="UTF-16"?>
<CycleFile>
    <Cycle Name="Cycle_HerdonProg"> ←第一筆資料開頭
        <Field Name="Col_Y" Value="17.63"/>
        <Field Name="Col_Z" Value="12.98"/>
        <Field Name="Col_X" Value="0.00"/>
        <Field Name="Col_A" Value="267.54"/>
    </Cycle> ←第一筆資料結尾
    <Cycle Name="Cycle_HerdonProg"> ←第二筆資料開頭
        <Field Name="Col_Y" Value="21.34"/>
        <Field Name="Col_Z" Value="13.20"/>
        <Field Name="Col_X" Value="2.26"/>
        <Field Name="Col_A" Value="300.87"/>
    </Cycle> ←第二筆資料結尾
    <Cycle Name="Cycle_HerdonProg"> ←第三筆資料開頭
        <Field Name="Col_Y" Value="91.19"/>
        <Field Name="Col_Z" Value="13.20"/>
        <Field Name="Col_X" Value="2.26"/>
        <Field Name="Col_A" Value="443.29"/>
    </Cycle> ←第三筆資料結尾
    <Cycle Name="Cycle_HerdonProg"> ←第四筆資料開頭
        <Field Name="Col_Y" Value="21.55"/>
        <Field Name="Col_Z" Value="12.98"/>
        <Field Name="Col_X" Value="0.00"/>
        <Field Name="Col_A" Value="116.95"/>
    </Cycle> ←第四筆資料結尾
    <Cycle Name="Cycle_HerdonProg"> ←第五筆資料開頭
        <Field Name="Col_Y" Value="21.16"/>
        <Field Name="Col_Z" Value="12.98"/>
```

```
            <Field Name="Col_X" Value="-6.05"/>
            <Field Name="Col_A" Value="150.00"/>
        </Cycle> ←第五筆資料結尾
</CycleFile>
```

-> 使用者需自行編撰xml資料結構設定檔，用來宣告當巨集利用DBLOAD函數讀取資料時，要將所讀取的相關資料，存放至哪些對應變數中。
其語法格式定義如下，並且要將此設定檔儲存於OCRes\\Common\\Schema\\中。
若為Dipole前後臺連線時，schema檔案放置於**後臺控制器**之OCRes\\Common\\Schema\\中。

```
<?xml version="1.0" encoding="UTF-16"?>
<Schema>
    <Cycle name="Cycle_HerdonProg">
        <Field>
            <Name>Col_X</Name>
            <InputStorage>@1200</InputStorage> ←Col_X存入的變數中
            <InputFormat>Variant</InputFormat>
            <DefaultValue></DefaultValue>
        </Field>
        <Field>
            <Name>Col_Y</Name>
            <InputStorage>@1201</InputStorage> ←Col_Y存入的變數中
            <InputFormat>Variant</InputFormat>
            <DefaultValue></DefaultValue>
        </Field>
        <Field>
            <Name>Col_Z</Name>
            <InputStorage>@1202</InputStorage> ←Col_Z存入的變數中
            <InputFormat>Variant</InputFormat>
            <DefaultValue></DefaultValue>
        </Field>
        <Field>
            <Name>Col_A</Name>
            <InputStorage>@1203</InputStorage> ←Col_A存入的變數中
            <InputFormat>Variant</InputFormat>
            <DefaultValue></DefaultValue>
        </Field>
        <Field>
            <Name>Col_B</Name>
            <InputStorage>@1204</InputStorage> ←Col_B存入的變數中
            <InputFormat>Variant</InputFormat>
            <DefaultValue></DefaultValue>
        </Field>
        <Field>
            <Name>Col_C</Name>
            <InputStorage>@1205</InputStorage> ←Col_C存入的變數中
```

```
            <InputFormat>Variant</InputFormat>
            <DefaultValue></DefaultValue>
        </Field>
    </Cycle>
</Schema>
```

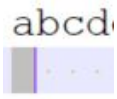
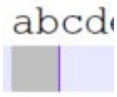
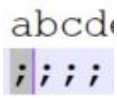
-> MACRO範例

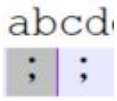
```
// 載入GNCFILES\\Test資料數，總共5筆，因此@1:=5;
@1:=DBOPEN("Test");

// 載入第1筆資料，DBLOAD引數為0
// @1200=0.00 @1201=17.63 @1202=12.98 @1203=267.54
DBLOAD( 0 );

// 載入第2筆資料，DBLOAD引數為1
// @1200=2.26 @1201=21.34 @1202=13.20 @1203=300.87
DBLOAD( 1 );
```

12.3 建議PC端編輯環境

- notepad++：https://notepad-plus-plus.org/downloads/
  - 開啟顯示所有字元：檢視→特殊字元→顯示所有字元
    - 可以檢視 Marco 檔案是否有未支援字元

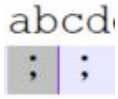
常見字元判斷

名稱	圖式	說明	是否支援
半形空白符號	 (灰色選取部分)	半形空白符號為紅色點點，一點代表一格空白	是
全形空白符號	 (空白，灰色選取部分)	全形空白符號為完全空白，較不易排查	否
半形分號符號	 (灰色選取部分)	半形分號符號如左圖所示	是

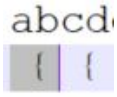
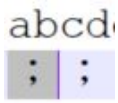
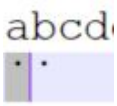
名稱	圖式	說明	是否支援
全形分號符號	 (灰色選取部分)	全形分號符號如左圖所示	否

- 切換編碼 (語系): 編碼→字元集
  - 可以切換顯示的語系，協助排查問題
- notepad++ 標準排查步驟
  i. 確認使用的編碼
     1. ANSI、UTF-8: 跳至第二點
     2. 繁體中文編碼 (Big5) 與簡體中文編碼 (GB2312): 建議切換兩種編碼檢視，因為如果檔案中有全形符號，顯示內容可能會改變，造成排查問題。
        編碼切換顯示差異

        繁體中文→簡體中文

名稱	繁體中文編碼	簡體中文編碼	是否支援
全形空白符號	 (空白，灰色選取部分)	 (空白，灰色選取部分)	否
全形分號符號	 (灰色選取部分)	 (空白，灰色選取部分)	否

        簡體中文→繁體中文

名稱	簡體中文編碼	繁體中文編碼	是否支援
全形空白符號	 (空白，灰色選取部分)	 (灰色選取部分)	否
全形分號符號	 (灰色選取部分)	 (灰色選取部分)	否

  ii. 確認是否有未支援字元
     - 全形空白/分號建議排查方式

a. 點選搜尋→搜尋
b. 在尋找內容輸入全形空白/分號
c. 點選在目前文件中全部尋找
d. 如有找到，在搜尋結果中會顯示
e. 修改為支援字元

12.4 MACRO 支援字元

- 支援字元 (僅支援 ASCII 字元)

可顯示字元	數字	1	2	3	4	5	6	7	8	9	0										
	英文字母	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
	其他(半形)	(space)	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	:				
		;	<	>	=	?	@	[	]	\	^	_	`	{	}	|	~					
控制字元	Unicode表示法	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI	DLE				
		DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US	DEL					

ASCII 字元詳細內容請參考：ASCII