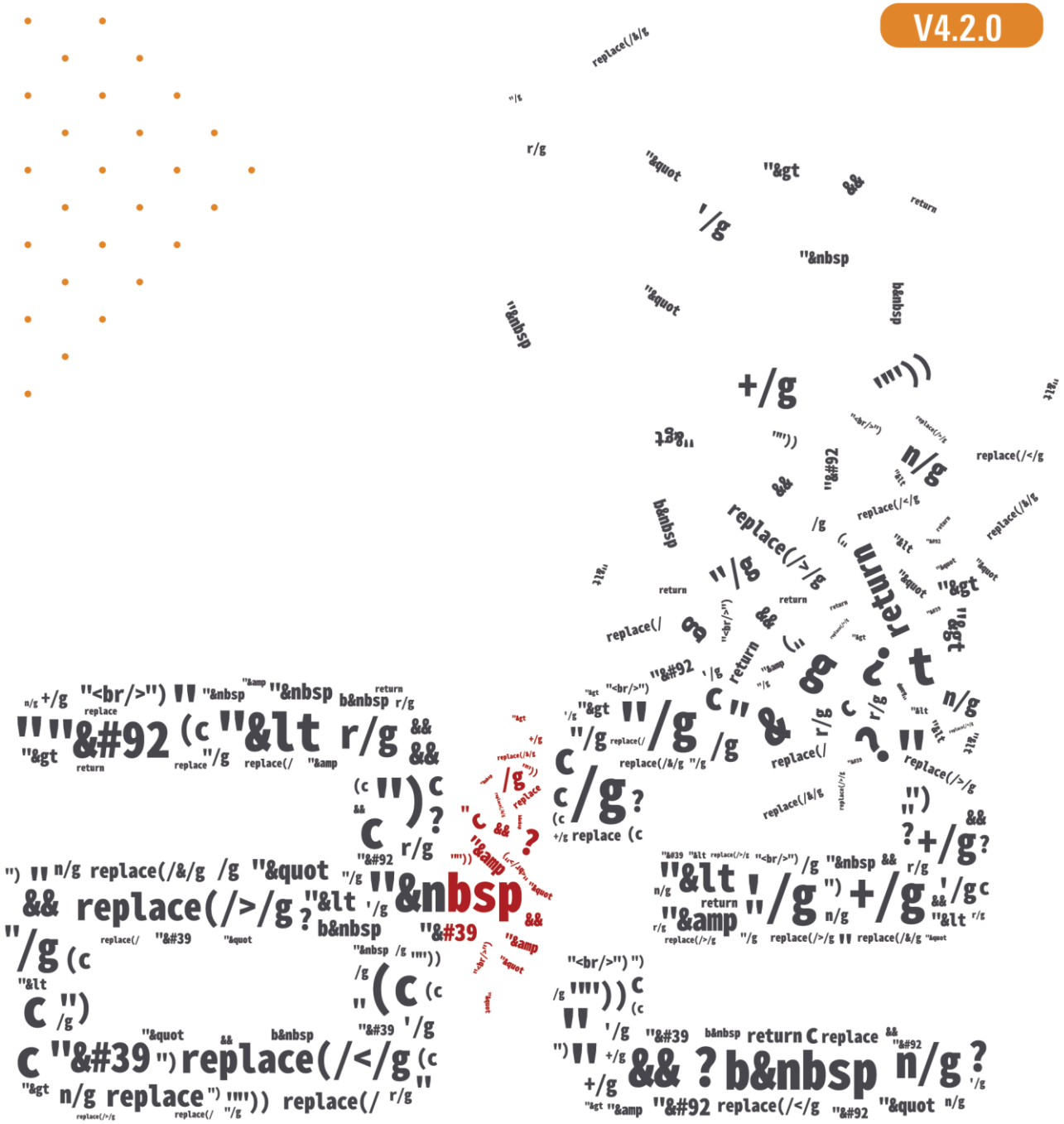
ARL程式設計手冊

V4.2.0



台配天机器人
SMARTER·STRONGER

前言

關於本手冊

ARL 程式設計手冊詳細描述了 ARL 程式設計格式規範以及 ARL 支持的程式設計指令。

目標群體

- 操作人員
- 產品技術人員
- 技術服務人員
- 機器人示教員

常見標識含義

手冊中出現標識及其含義詳見下表 1。

表 1 本文中使用的標識

標誌	含義
 危險	如不按照說明進行操作，就會發生事故，導致嚴重或致命的人員傷害，或嚴重的物品損壞
 警告	如不按照說明進行操作，可能發生事故，導致嚴重或致命的人員傷害，或嚴重的物品損壞
 注意	提示您需要註意的環境條件和重要事項，或快捷操作方法
 提示	提示您參閱其他文獻和說明，以便獲取附加信息或更加詳細的操作說明

手冊說明

文檔相關信息見表 2。

表 2 文檔相關信息

文檔編號	BJM/SS-UG-02-003
文檔版本	V4.2.0
軟體版本號	2.6.3

本手冊內容會有補充和修改，請定時留意我公司網站的“下載中心”，及時獲取最新版本的手冊。

我公司網站網址：http://robot.peitian.com/

通用安全說明

感謝貴公司購買本公司產品，本說明資料為安全使用本公司產品而需要遵守的內容，在使用之前，請務必仔細閱讀相關手冊，並且在理解該內容的前提下正確使用。


警告

程式設計時盡可能在安全柵欄外進行，因不得已情形而需要在安全柵欄內進行時，應註意下列事項：

- 仔細查看安全柵欄內情況，確認沒有危險後再進入柵欄內部。
- 要做到隨時都可以按下急停按鈕。
- 應以低速運行操作機。
- 應在確認整個系統的狀態後進行作業，避免由於針對外圍設備的遙控指令或動作等而導致作業人員陷入危險境地。


注意

在程式設計結束後，務必按照規定步驟進行測試運轉，此時，作業人員務必在安全柵欄外進行操作。


提示

進行程式設計的作業人員，務必通過本公司的相關培訓接受適當的培訓。

符號約定

如下為一種含指令 TPWrite 的簡化語法示例。

movej j:,[v: | vp:],[s: | sp: | sl:],[t:],[dura:]

- 括號中不含強制性參數。
- 用方括號“[ ]”將可選參數括起來，可忽略這些參數。
- 互相排斥的參數不能同時存在於同一指令中，在同一指令中就要用豎線“|”隔開。
- 用波形括號“{ }”將可重覆任意次的參數括起來。
- 上述示例採用瞭如下參數：
- j 為強制性參數。
- v、vp、s、sp、sl、t 和 dura 為可選參數。
- v 和 vp 互相排斥。
- s、sp 和 sl 互相排斥。

修訂記錄

修訂記錄累積了每次文檔更新的說明。最新版本的文檔包含以前所有文檔版本的更新內容。

表 2 本文中使用的標識

版本	發佈時間	修改說明
V4.2.0	2020/10/30	第五次正式發佈 ■ 軟體版本升級到 V2.6.3 ■ 新增結構體類型：Centroid_Pos（工具負載質心）/Inertia_Tensor（工具負載慣性主軸方向）/ToolInertiaPara（工具負載類型） ■ 新增運動指令：ccir(連續圓弧運動) ■ 新增 IO 相關函數：getnostopdi（不停前瞻異步獲取單路 DI）/syncao（同步輸出模擬量信號）/getao（獲取模擬量輸出信號） ■ 新增系統變量：$WOBJ_OFFSET(工件坐標系偏移)/$TOOL_OFFSET(工具坐標系偏移)

安全預防措施

在運行操作機和外圍設備及其組成的操作機系統前，必須充分研究作業人員和系統的安全預防措施，圖 1 為工業機器人安全工作示意圖。

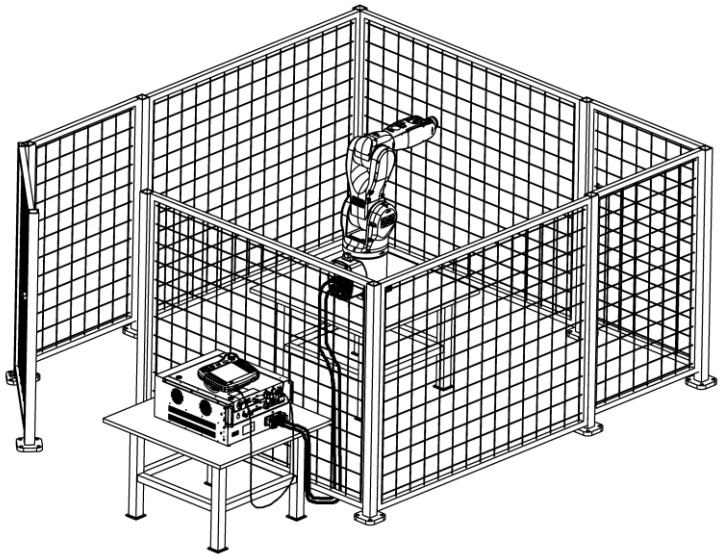


圖 1 工業機器人安全工作示意圖

作業人員定義

操作機的作業人員主要分為操作員、示教員、維護工程師三種，這三種作業人員需滿足的條件描述如下：

操作員

- 進行操作機電源 ON/OFF 的操作。
- 通過操作面板來啟動操作機程序。
- 無權進行安全柵欄內的作業。

示教員

- 具備操作員的職能。
- 可以在安全柵欄內進行操作機示教等。

維護工程師

- 具備示教員的職能。
- 可以進行操作機維護（修理、調整、更換等）作業。

作業人員的安全

在進行操作機操作、程式設計、維護時，操作員、示教員、維護工程師必須註意安全，至少應穿戴下列物品進行作業：

- 適合於作業內容的工作服
- 安全鞋
- 安全帽

在運用自動系統時，必須設法確保作業人員安全，進入操作機作業範圍是十分危險的，應採取防止作業人員進入操作機動作範圍的措施。

下麵列出一般性註意事項，請妥善採取確保作業人員安全的相應措施：

- 運行操作機系統的作業人員，應接受本公司的培訓並通過相關考核。
- 在設備運行時，即使操作機看上去已經停止，也有可能是因為操作機在等待啟動信號而處在即將動作的狀態。此狀態也應該視為操作機處在操作狀態。為了確保作業人員安全，應當以警報燈等的顯示或響聲等來確認操作機處在停止狀態或安全狀態。
- 務必在系統周圍設置安全柵欄和安全門，使得不打開安全門，作業人員就不能夠進入安全柵欄內。安全門上應該設置互鎖開關、安全插銷等，以使作業人員打開安全門時，操作機就會停下。
- 外圍設備均應電氣接地。
- 應盡可能地將外圍設備設置在操作機動作範圍之外。
- 應採用在地板上畫上線條等方式來標清操作機動作範圍，使得操作者清楚包括操作機上配備的機械手等工具在內的操作機動作範圍。
- 應在地板上設置墊片開關或者安裝光電開關等，以便當作業人員將要進入操作機動作範圍時，通過蜂鳴器和光等發出警報，使得操作機停下，由此確保作業人員安全。
- 應根據需要，設置一把鎖，除負責操作的作業人員外，不能接通操作機電源。
- 在進行外圍設備的單個調試時，務必斷開操作機的電源。

示教員的安全

在進行操作機示教作業時，某些情況下需要進入操作機工作範圍內，此時尤其要註意安全：

- 在不需要進入操作機動作範圍的情況下，務必在操作機動作範圍外進行作業。
- 在進行示教作業之前，應確認操作機或外圍設備處在安全狀態。
- 在迫不得已情況下需要進入操作機動作範圍內進行示教作業時，應事先確認安全裝置（如急停按鈕，示教器緊急自動停機開關等）的位置和狀態等。
- 示教員應特別註意，勿使其他人員進入操作機動作範圍。
- 在操作機啟動前，應充分確認操作機動作範圍內沒有人員且沒有異常後再執行。
- 在示教結束後，務必按照下列步驟執行測試運轉：
  1. 在低速下，單步執行至少執行一個循環，確認沒有異常。
  2. 在低速下，連續運轉至少一個循環，確認沒有異常。
  3. 在中速下，連續運轉至少一個循環，確認沒有異常。
  4. 在運轉速度下，連續運轉一個循環，確認沒有異常。
  5. 自動運行模式下執行程序。
- 示教員在操作機進行自動運轉時，務必撤離到安全柵欄外。

目錄

1 ARL 概述

1.1 術語和縮寫詞

- ARL AE Robot Language，即 AE 機器人程式設計語言。
- CP 運動 笛卡爾路徑運動，主要包括直線和圓弧運動，對應運動指令中的 lin 和 cir。
- PTP 運動 點到點運動，對應運動指令中的 ptp 和 movej。
- DI digital input，數字輸入信號。
- DO digital output，數字輸出信號。
- AO analog output，模擬輸出信號。
- RPP return to path point，RPP 運動是指機器人從當前位置返回到路徑點的運動。

1.2 程序文件

ARL 為 AE Robot Language 縮寫，即 AE 機器人程式設計語言。用戶可以通過 ARL 編寫控制機器人的用戶程序。

ARL 程序文件後綴名為 arl，例如：AEMoveObj.arl。

1.3 代碼註釋

ARL 中的代碼註釋的 2 種格式：

- 使用“//”代表註釋該行，“//”後面的部分即為註釋

程序示例：

```
//waittime 5

waituntil getdi(2) //等待通道 2 DI 信號變為 true
```

編譯器讀入一行後，首先將“//”後面的部分忽略，之後再進行解析編譯執行。

- 使用“/*”和“*/”註釋一個區域

程序示例：

```
/* this prog is for moving object*/

movej j:{j1 20}

waittime 5
```

1.4 換行連接符

當程序中一行太長，希望將其分為兩行或者多行編寫時，可以採用換行連接符“\”。

程序示例：

```
if(~x == -6 && y == 3 && A == 60 && (A&B) == 12 \&& (A|B) == 61 && (A^B) == 49 && (~A) == -61
\&& (A<<2) == 240 && (A>>2) == 15)
```

return true

else

return false

endif

1.5 子程序

利用子程序技術可以讓機器人程序更加模塊化，可幫助程式設計人員有效的結構化設計程序。目的是不必將所有指令寫入一個程序，而是將特定的流程、計算或過程放到單獨的程序中，以達到模塊化復用的目的。

- 當子程序和主程序處於同一目錄下時的調用方法：

  SubProg::func()

- 當子程序和主程序處於不同目錄下時的調用方法：

  import "/home/ae/.../SubProg.arl" //導入子程序文件

  SubProg::func()

  程序指針執行到上述程序段時，會跳轉到 SubProg 程序的 func 函數中。

- 子程序結構與一般程序沒有明顯區別，只是可以不包含主函數。
- 子程序被調用的函數結束後（即執行到 endfunc 後）程序指針會返回調用處，如果想提前結束子程序，可以在需要終止的地方插入 return 指令，這樣會提前終止子程序的運行。

1.6 函數

當處理相似的，通常是重覆的程序部分時，為了減少字元的數量和程序的長度，引入函數功能。

- 一個 ARL 程序可以由一個或多個函數組成

函數定義格式如下：

func returntype funcname(argtype argname,argtype argname……)

內部為函數體實現

endfunc

其中，returntype 為返回值類型，funcname 為函數名，argtype 為參數類型，argname 為參數名。

- 實現兩個整數加法的函數的示例

程序示例：

func int add(int a,int b)

return a+b

endfunc

函數的調用格式如下：

funcname(arg1,arg2,……)

- 調用之前定義的 add 函數的方式

例如：

int c = add(1,2) //c 的值為 3

- 函數的返回值可以繼續用在表達式中

例如：

int c = add(1,2) * 3 //c 的值為 9

- ARL 程序沒有指針的概念

ARL 程序沒有指針的概念，函數參數只是值傳遞，如果要某個函數內修改傳入參數的值，需要將該參數聲明為全局變量。

下述程序示例中，print 列印結果是 1，而不是 2。因為 add 函數對 a 進行的自加操作不會傳遞給變量 a。

程序示例：

```
func void add(int a)
a++
endfunc
func void main()
init()
int a=1
add(a)
print a
endfunc
```

- main 函數

ARL 程序中必須要實現一個 main 函數，也就是程序的入口函數。程序複位後，程序指針將指向 main 函數中的第一行代碼。程序運行時，將從 main 函數的第一行代碼開始執行。

main 函數沒有參數也沒有返回值，所以按如下形式定義：

main 函數定義

```
func void main()
……
endfunc
```

其中，void 表示函數沒有返回值。

- 使用 ARL 語言編寫 HelloWorld 程序

程序示例：

```
//HelloWorld.arl

func void main()

print "Hello world!"

endfunc
```

print 為列印輸出函數，默認輸出到 HMI 的消息欄中。加載並運行該程序，可以看到 HMI 的消息欄中列印出“Hello world!”。

注意

需要註意的是如果在同一個文件中定義不同的函數，則被調用的函數必須在調用函數之前定義，否則加載時會報出找不到被調用函數的錯誤。

2 變量與運算

2.1 常量

程序執行過程中不能修改其值的量稱為常量。

ARL 支持四種類型的基本常量，分別是：

- 整型（int）
- 浮點型（double）
- 布爾型（bool）
- 字元串類型（string）

其中，整型常量可以使用十進制、二進制（數字後面跟 b）或者十六進制（數字後面跟 h）來表達；浮點型常量可以以小數形式或科學計數法來表達。

例如，下麵都是合法的常量：

- 1 //十進制整型
- 10b //2 進制整型，值為 2
- 3eh //16 進制整型，值為 62
- 3.245 //浮點型
- 0.15e2 //科學計數法表達的浮點型，表示 0.15 乘以 10 的平方，值為 15
- true //布爾型，值為真
- false //布爾型，值為假

2.2 變量和名稱

程序執行過程中可以修改其值的量稱為變量。ARL 支持 6 種類型的基本變量，分別是：

- 整型 (int)
- 無符號整形 (uint)
- 位元組型 (byte)
- 浮點型 (double)
- 布爾型 (bool)
- 字元串類型 (string)

使用變量前必須要對該變量進行聲明，變量名為用戶自定義，可由字母，下劃線，數字的組合組成。變量名不能以數字開頭。變量名不能命名為系統關鍵字。ARL 系統關鍵字參見。

- 在使用一個變量之前需要首先聲明它，變量聲明語句的一般格式如下：

變量類型 變量名

或

變量類型 變量名 = 初值

程序示例：

int a = 1

該聲明語句聲明瞭一個 int 類型，名字為 a 的變量並賦予了初始值 1。

- 同一類型的變量可以在同一行中聲明，格式為：

變量類型 變量名 = 初值，變量名 = 初值，……

程序示例：

int x = 1,y =3

該聲明語句聲明瞭兩個 int 類型的變量，名字為 x 的變量並賦予了初始值 1，名字為 y 的變量並賦予了初始值 3。

- 如果在定義的變量之前加上 const 關鍵字，則表示該變量不允許被再次賦值。

程序示例：

const double pi = 3.1415926

pi = 3.4

則加載程序時會報錯。

2.3 基本數據類型

ARL 支持 6 種類型的基本變量：

- 整型 (int)
- 無符號整形 (uint)
- 位元組型 (byte)
- 浮點型 (double)
- 布爾型 (bool)
- 字元串類型 (string)

2.3.1 int(整型)

int 類型數據為一個 32bit 數，可以表達一個範圍在-231~231-1 的整數。

程序示例：

int counter = 4

表示定義了一個整型變量 counter，且它的數值等於 4。

2.3.2 uint(無符號整型)

uint 類型數據為一個 32bit 數，可以表達一個範圍在 0~232-1 的整數。

程序示例：

uint ae = 5

表示定義了一個無符號整型變量 ae，且它的數值等於 5。

2.3.3 byte(位元組型)

byte 類型數據為一個 8bit 數，可以表達一個範圍在 0~255 的整數。

程序示例：

byte b = ffh

表示定義了一個 byte 型變量 b，且它的數值等於 ffh，也就是十進制的 255。

2.3.4 double(浮點型)

double 類型數據為一個 64bit 數，可以表達一個範圍在-1.7*10-308~1.7*10308 的小數。

程序示例：

double scbd=1

表示定義了一個浮點型變量 scbd，且值等於 1。

double 類型數據存在精度問題。請儘量避免兩個浮點數之間比較是否相等的表達式。

程序示例（錯誤用法）：

double data = 0

……

if(data == 0)

……

endif

程序示例（正確用法）：

double data = 0

……

if(abs(data) < 0.000001)

……

endif

2.3.5 bool(布爾型)

bool 型變量表示邏輯真或假，一個 bool 類型變量的值只能為 true 或者 false。

例如：

bool io16 = true

表示定義了一個 bool 型變量 io16，且值為 true。

2.3.6 string(字元串類型)

字元串類型數據表達了一個由 1 個或多個字元組成的字元串。

例如：

string name = "Vincent"

這條語句定義了一個字元串變量 name，並初始化為“Vincent”。

但是某些字元沒有對應的單獨符號，例如換行符，所以需要依賴以反斜杠“\”開頭的轉義字元來表達。

ARL 中支持的轉義字元如下：

- \n 換行（LF），將當前位置移到下一行開頭
- \r 回車（CR），將當前位置移到本行開頭
- \t 水平製表（HT），跳到下一個 tab 位置
- \” 代表一個雙引號字元
- \\ 代表一個反斜杠字元“\”

2.3.7 基本數據隱式類型轉換

某些情況下，系統會對基本類型變量做隱式類型轉換，允許進行隱式類型轉換的類型如表 2-1 所示：

表 2-1 隱式類型轉換表

from \ to	byte	int	double	bool
byte		√	√	√
int			√	√

表中畫有“√”的項表示兩種類型之間允許從左側的類型隱式類型轉換為上方的類型。如 byte 類型就允許隱式類型轉換為 int 類型。

例如：

```
int a = 3

double b = 4

double c = a+b //系統將隱式地把 a 轉換成 double 類型再與 b 相加。

int counter = 5

while(counter--)

……

Endwhile //系統會將 counter 隱式地轉換成 bool 型數據。非 0 值會被轉為 true，0 值會被轉為 false。所以此段代碼循環將會執行 5 次。
```

2.4 結構體類型（適用於六軸機器人）

ARL 中支持系統預定義的結構體類型，結構體類型是由基本類型組成的複合類型。

2.4.1 結構體變量的聲明，初始化及引用

以 pos 類型為例，空間的任意一個點坐標由 x，y，z 3 個坐標分量組成，所以 pos 類型，也就是空間點坐標類型由 3 個 double 類型的子分量組成。

pos 類型的定義如下：

```
struct pos
{
double x
double y
double z
}
```

結構體類型變量的聲明與基本類型變量的聲明相同，格式為：

變量類型 變量名

例如：

```
pos a
```

聲明瞭一個名為 a 的 pos 類型變量，如果我們用 print 指令列印 a 的值：

```
print a
```

系統將輸出：

```
{9e+09,9e+09,9e+09}
```

逗號隔開的 3 個數字分別為 a 變量中每個分量的值，可見，默認情況下，pos 類型變量 a 的分量 x，y，z 分別被初始化為 9e+09，9e+09，9e+09。

可以在變量聲明的同時對變量進行初始化。結構體變量初始化的格式有以下 2 種方式：

- 初始化結構體成員的前 n 個成員，格式為：

結構體類型 變量名 = {分量 1 的值，分量 2 的值，……分量 n 的值}

- 初始化結構體成員的某 1 個或幾個成員：

結構體類型 變量名 = {分量名字 分量值，分量名字 分量值，……}

對於以上兩種結構體變量初始化方式，未初始化的分量將被系統初始化為默認值。

程序舉例如下：

```
pos a = {1,2}
print a //輸出“{1,2,9e+09}”
pos b = {x 3,z 4}
print b //輸出“{3,9e+09,4}”
```

使用成員操作符“.”訪問一個結構體變量的成員分量，格式為：

結構體變量名.分量名

程序示例：

```
pos a
a.x = 1
a.y = 2
a.z = 3
double b = a.x
print a //輸出“{1,2,3}”
print b //輸出“1”
```

結構體支持嵌套，也就是說某個結構體類型的某個成員可能是另外一個結構體類型。考慮如下結構體類型 tool 的定義：

```
struct tool
{
frame t_frame
bool stationary
}
```

聲明一個 tool 類型的變量並輸出：

```
tool a
print a
```

系統將會輸出：

```
{{0,0,0,0,0,0},false}
```

這裡結構體變量將會以嵌套的花括號的形式輸出。同樣初始化結構體變量的時候也要以嵌套的花括號形式。

程序舉例如下：

```
tool a = {{10,10,0,0,90,0},false}
print a //輸出“{{10,10,0,0,90,0},false}”
print a.t_frame //輸出“{10,10,0,0,90,0}”
print a.t_frame.x //輸出“10”
tool b = {{x 10,b 90},false}
print b //輸出“{{10,0,0,0,90,0},false}”
```

2.4.2 pos(空間點坐標)

描述

pos 結構體類型用於描述空間任意一點的坐標。

定義

```
struct pos
{
double x
double y
double z
}
```

成員

pos 結構體類型的成員詳見表 2-2。

表 2-2 pos 結構體類型的成員

名稱	說明
x	類型：double
	空間一點坐標的 x 分量，單位毫米
y	類型：double
	空間一點坐標的 y 分量，單位毫米
z	類型：double
	空間一點坐標的 z 分量，單位毫米

用法舉例

```
pos p1 = {0,0,0}
print p1 //輸出“{0,0,0}”
p1.x = p1.x +50 //以對結構體成員進行運算
print p1 //輸出“{50,0,0}”
```

2.4.3 frame(坐標系)

描述

frame 結構體類型用於描述空間兩個直角坐標系之間的轉換關係。

定義

```
struct frame
{
double x
double y
double z
```

double a
double b
double c
}

成員

frame 結構體類型的成員詳見表 2-3。

表 2-3 frame 結構體類型的成員

名稱	說明
x	類型：double
	轉換後坐標系原點在原坐標系中坐標的 x 分量，單位毫米
y	類型：double
	轉換後坐標系原點在原坐標系中坐標的 y 分量，單位毫米
z	類型：double
	轉換後坐標系原點在原坐標系中坐標的 z 分量，單位毫米
a	類型：double
	轉換後坐標系姿態在原坐標系中歐拉角表達的 a 分量，單位度
b	類型：double
	轉換後坐標系姿態在原坐標系中歐拉角表達的 b 分量，單位度
c	類型：double
	轉換後坐標系姿態在原坐標系中歐拉角表達的 c 分量，單位度


注意

歐拉角按照旋轉所繞軸的次序不同，共有 12 種不同的歐拉角表達方式，我們這裡採用的是 ZYX 型歐拉角，即從初始坐標系姿態轉換為變換坐標系姿態的順序為先繞 z 軸旋轉 a 角度，再繞新的 y 軸旋轉 b 角度，最後繞新的 x 軸旋轉 c 角度。

用法舉例

frame f = {10,10,0,0,90,0}

2.4.4 pose(笛卡爾目標點)

描述

pose 結構體類型使用笛卡爾坐標的方式來描述機器人各軸以及外軸的位置。

定義

struct pose
{
double x
double y
double z

double a
double b
double c
int cfg
int turn
double ej1
double ej2
double ej3
double ej4
double ej5
double ej6
}

成員

pose 結構體類型的成員詳見表 2-4。

表 2-4 pose 結構體類型的成員

名稱	說明
x	類型：double
	機器人 TCP 點相對當前工件坐標系下的 x 方向分量，單位毫米
y	類型：double
	機器人 TCP 點相對當前工件坐標系下的 y 方向分量，單位毫米
z	類型：double
	機器人 TCP 點相對當前工件坐標系下的 z 方向分量，單位毫米
a	類型：double
	機器人工具姿態在當前工件坐標系中歐拉角表達的 a 分量，單位度
b	類型：double
	機器人工具姿態在當前工件坐標系中歐拉角表達的 b 分量，單位度
c	類型：double
	機器人工具姿態在當前工件坐標系中歐拉角表達的 c 分量，單位度
cfg	類型：int
	機器人軸配置。由於機器人可能存在幾種不同的方式使 TCP 到達同一個位姿，所以這裡需要通過 cfg 參數（取值 0-7，代表可能的 8 種方式，參考圖 21，其中，“beta”指代五軸的角度）來指定其中一種方式，從而唯一確定一組機器人軸位置。參數取值及位置關係見表 2-5 和圖 21.所示。
turn	類型：int
	配合 cfg 用於確定反解的軸位置。這裡只用到 turn 的低 6 個 bit 位，bit0 表示 1 軸，bit1 表示 2 軸，以此類推。 ■ 當 turn 值為-1 時，表示自動選取距離起點最近的解； ■ 當 turn 值為 1 時，表示該軸選小於 0 的解； ■ 當 turn 值為 0 時，表示該軸選大於 0 的解。

名稱	說明
	■ 當軸的運動範圍大於 360 度時可能會用到這個參數輔助選解，其他大部分情況下都不需要設置該值。
ej1~ej6	類型：double
	外 1 軸~外 6 軸的位置，直線軸單位為 mm，旋轉軸單位為度

表 2-5 cfg 取值及位姿關係

cfg 取值	相對於軸 1 的腕中心	相對於大臂的腕中心	5 軸角度
0	在右側	在右側	正
1	在右側	在右側	負
2	在右側	在左側	負
3	在右側	在左側	正
4	在左側	在左側	正
5	在左側	在左側	負
6	在左側	在右側	負
7	在左側	在右側	正

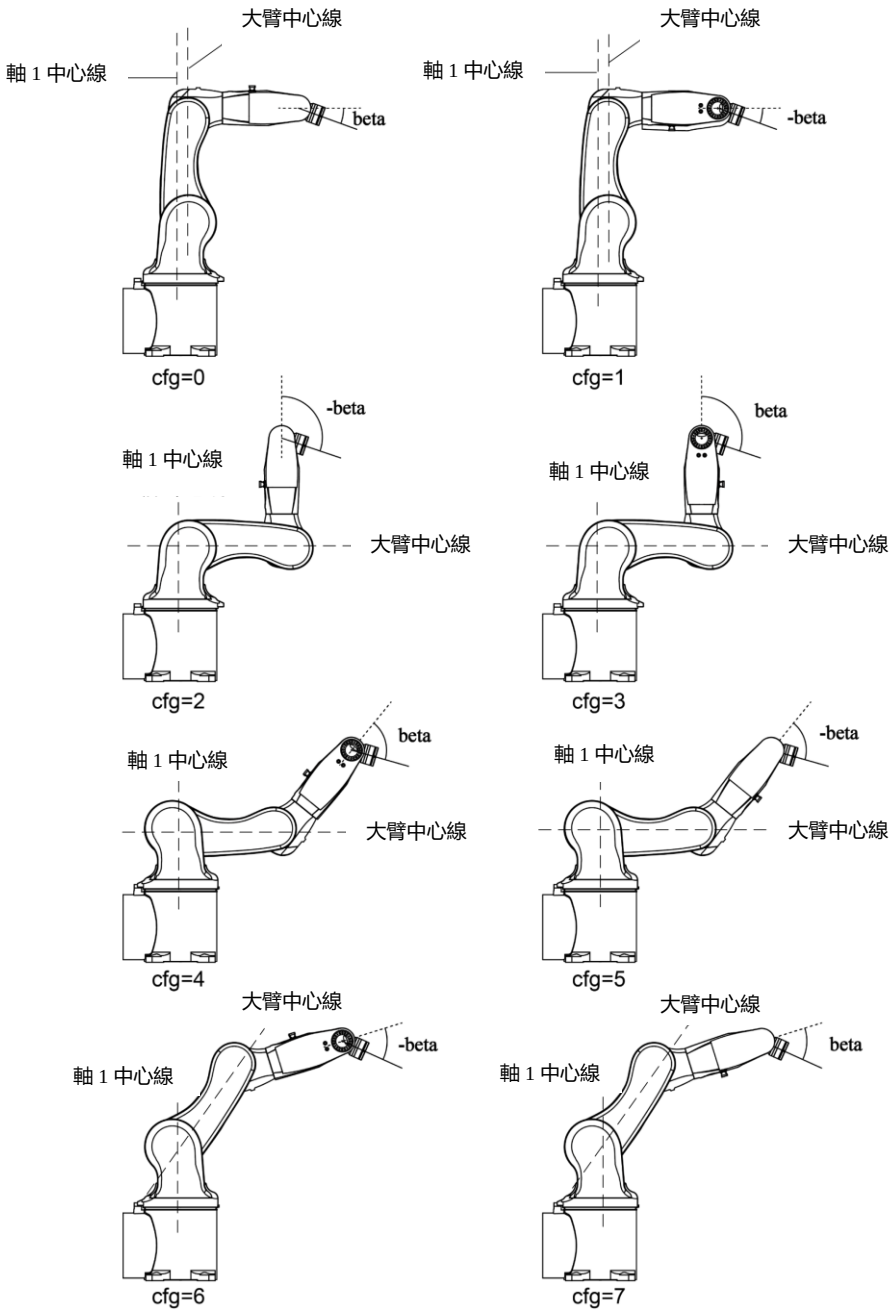


圖 2-1 cfg 參數值姿態說明

注意

歐拉角定義參見第 2.4.3 章节。

用法舉例

pose p1 = {x 0,y 10,z 15,a 0,b 90,c 0,cfg 0,ej1 20}

pose p2 = {0,-10,15,0,90,0,0,-1,10}

2.4.5 joint(軸目標點)

描述

joint 結構體類型用於直接描述機器人各軸以及外軸的位置。

定義

```
struct joint
{
double j1
double j2
double j3
double j4
double j5
double j6
double ej1
double ej2
double ej3
double ej4
double ej5
double ej6
}
```

成員

joint 結構體類型的成員詳見表 2-6。

表 2-6 joint 結構體類型的成員

名稱	說明
j1~j6	類型：double
	機器人 1 軸~6 軸的軸位置，單位度
ej1~ej6	類型：double
	外 1 軸~外 6 軸的位置，直線軸單位為 mm，旋轉軸單位為度

用法舉例

```
joint p1 = {j1 0,j3 15,j5 0,ej1 20}
print p1 //輸出“{0,9e+09,15, 9e+09,0,9e+09,20,9e+09,9e+09,9e+09,9e+09,9e+09}”
joint p2 = {0,10,20,30,40,50,10}
print p2 //輸出“{0, 10,20,30,40,50,10,9e+09,9e+09,9e+09,9e+09,9e+09,9e+09}”
```

注意

當所有軸位置均指定時，joint 中的參數名稱可預設，但必須全部預設。例如以下的寫法就是錯誤的：j:{0,0,90,0,0,0,ej1 0}。

2.4.6 tool(工具)

描述

tool 結構體類型用於描述一個工具。

定義

```
struct tool
```

```
{
frame t_frame
bool stationary
}
```

成員

tool 結構體類型的成員詳見表 2-7。

表 2-7 tool 結構體類型的成員

名稱	說明
t_frame	類型：frame
	在工具上定義的工具坐標系
stationary	類型：bool
	■ false：指定該工具為安裝在機器人法蘭末端的工具。此時工具坐標系參照法蘭坐標系定義 ■ true：指定該工具為外部的固定工具。此時工具坐標系參照世界坐標系定義

用法舉例

```
frame f = {10,10,0,0,90,0}
tool t1
t1.t_frame = f
t1.stationary = false
print t1 //輸出“{{10,10,0,0,90,0},false}”
tool t2 = {{0,10,20,0,0,0},true}
print t2 //輸出“{{0,10,20,0,0,0},true}”
```

2.4.7 wobj(工件坐標系)

描述

wobj 結構體類型用於描述一個工件坐標系。

定義

```
struct wobj
{
frame w_frame
bool robhold
}
```

成員

wobj 結構體類型的成員詳見表 2-8。

表 2-8 wobj 結構體類型的成員

名稱	說明
w_frame	類型：frame
	工件坐標系
robhold	類型：bool
	■ false：指定工件坐標系參照世界坐標系定義 ■ true：指定該工件由機器人抓取。此時工件坐標系參照法蘭坐標系定義

用法舉例

```
frame f = {10,10,0,0,90,0}
wobj w1
w1.w_frame = f
print w1 //輸出“{{10,10,0,0,90,0},false}”
wobj w2 = {{0,10,20,0,0,0},false}
print w2 //輸出“{{0,10,20,0,0,0},false}”
```

2.4.8 weavedata(擺動圖形參數)

描述

weavedata 結構體類型用於描述擺動的圖形參數。

定義

```
struct weavedata
{
enum weaveshape
double frequency
double amplitude
double dwell_left
double dwell_right
double dwell_middle
bool track
double swing_angle
double radius
enum weaverotaxis
rotation_angle
bool vibrat
}
```

成員

weavedata 結構體類型的成員詳見表 2-9。

表 2-9 weavedata 結構體類型的成員

名稱	說明
weaveshape	類型：enum
	枚舉類型，包含了所有疊加軌跡支持的圖形類型，詳見第 2.6.8 章節
frequency	類型：double
	擺動的循環頻率，單位為赫茲
amplitude	類型：double
	擺動振幅，單位毫米
dwell_left	類型：double
	左側停留長度，循環類型為頻率時，該參數表示停留時長，單位秒，循環類型為波長時，該參數表示停留距離，單位毫米。該參數的最小值為 0
dwell_right	類型：double
	右側停留長度，循環類型為頻率時，該參數表示停留時長，單位秒，循環類型為波長時，該參數表示停留距離，單位毫米。該參數的最小值為 0
dwell_middle	類型：double
	中心停留長度，循環類型為頻率時，該參數表示停留時長，單位秒，循環類型為波長時，該參數表示停留距離，單位毫米。該參數的最小值為 0
track	類型：bool
	是否進行焊縫跟蹤，true 表示跟蹤，false 表示不跟蹤，可預設，默認值為 false
swing_angle	類型：double
	空間三角擺和空間 v 型擺的擺動角度，單位度
radius	類型：double
	圖形半徑，單位毫米
weaverotaxis	類型：enum
	該枚舉類型數據包含了疊加軌跡支持的擺動平面偏轉所參考的坐標軸
Vibrat	類型：bool
	是否開啟起振功能。
rotation_angle	類型：double
	擺動平面的偏轉角度，單位度

用法舉例

```
weavedata weavedatalin1 = {2,15,1,1,0,true}
weavedata weavedatalin2 = {2,15}
```

2.4.9 speed(速度)

描述

speed 結構體類型用於描述一條運動指令的速度參數。

定義

```
struct speed
{
double per
double tcp
double ori
double exj
double exl
}
```

成員

speed 結構體類型的成員詳見表 2-10。

表 2-10 speed 結構體類型的成員

名稱	說明
per	類型：double
	速度百分比，ptp 和 movej 指令使用該速度參數，表示軸最大速度百分比，取值範圍 0.001~100
tcp	類型：double
	TCP 點移動速度，lin 和 cir 指令使用該速度參數，表示 TCP 點平動速度，單位毫米/秒，不同機型取值範圍如下： AIR3/4/6L/7L/10/20：TCP=2500 AIR50、165：TCP=1300
ori	類型：double
	工具坐標系姿態轉動速度，lin 和 cir 指令使用該速度參數，表示 TCP 點轉動速度，單位度/秒，不同機型取值範圍如下： AIR3/4/6L/7L/10：ORI=500 AIR20：ORI=500 AIR50、165：ORI=250
exj	類型：double
	旋轉外軸速度。當有旋轉外軸時，移動時使用該速度參數，單位度/秒
exl	類型：double
	直線外軸速度。當有直線外軸時，移動時使用該速度參數，單位毫米/秒

用法舉例

```
speed v = {per 10}
print v //輸出“{10,9e+09,9e+09,9e+09,9e+09}”
```

2.4.10 slip(平滑參數)

描述

slip 結構體類型用於描述一條運動指令的平滑參數。

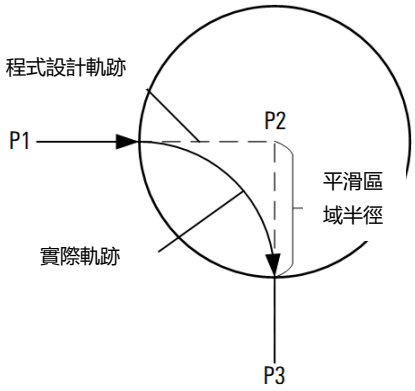
定義

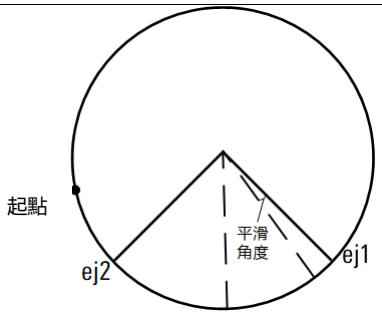
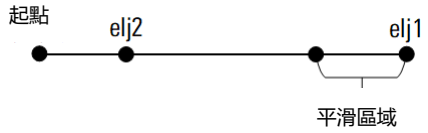
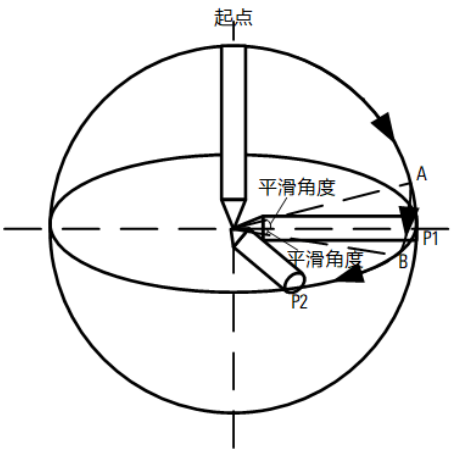
```
struct slip
{
double pdis
double ejdis
double eldis
double odis
double perdis
}
```

成員

slip 結構體類型的成員詳見表 2-11。

表 2-11 slip 結構體類型的成員

名稱	說明
pdis	類型：double
	pos distance，lin 和 cir 指令使用這項平滑參數，表示 TCP 距離目標點多少毫米（平滑區域半徑）時開始平滑，單位毫米。 如下圖所示，程式設計軌跡為“P1→P2→P3”，當 TCP 進入平滑區域後，沿着平滑的圓弧軌跡運動，離開平滑區域後沿直線運動到 P3。 
ejdis	類型：double
	ex-joint distance，有旋轉型外軸的運動指令使用這項平滑參數，表示最慢的外軸距離目標點多少度（平滑角度）時開始平滑，外軸為旋轉軸時，單位為度；外軸為直線軸時，單位為毫米。 如下圖所示，外軸的程式設計軌跡為“起點→ej1→ej2”，當外軸旋轉到平滑角度範圍內開始平滑運動，並開始向著 ej2 點的方向平滑運動。

名稱	說明
	
eldis	類型：double
	exlinearjoint distance，有移動型外軸的運動指令使用這項平滑參數，表示最慢的外軸距離目標點多少毫米（平滑距離）時開始平滑，外軸為旋轉軸時，單位為度；外軸為直線軸時，單位為毫米。 如下圖所示，外軸的程式設計軌跡為“起點→elj1→elj2”，當外軸直線運動到平滑區域內開始平滑運動．並開始向著 ej2 點方向平滑運動。  平滑區域
odis	類型：double
	orientation distance，lin 和 cir 指令使用這項平滑參數，表示姿態距離目標點多少角度（平滑角度）時開始平滑，單位度。 如下圖所示，工具的程式設計軌跡為“起點→P1→P2”，當工具沿圓弧運動到 A 點（即平滑角度內）時，在平滑區域內開始平滑運動到 B 點，再從 B 點運動到 P2。 
perdis	類型：double
	Percentage distance，所有運動指令均可以使用此平滑參數。當此平滑參數生效時，其他平滑參數將會無效。表示軌跡在可平滑段開啟平滑的百分比，100%為完全平滑，0%為不平滑

用法舉例

slip s = {100,2,3,4,5,5}


注意

- 當 pdis 參數小於等於 0 時，表示使用它的運動語句不平滑。
- 請儘量在初始化結構體變量時將結構體的所有成員都寫全，以防止由於用戶的疏忽導致實際的運動軌跡或者運動速度和用戶預期不符，進而導致一些不必要的損失或傷害。

■ perdis 優先級最高，在有 perdis 的情況下，其餘參數無效；當輸入的 pdis，odis，ejdis 生效時，平滑點更靠近目標點的值生效。

2.4.11 jvel(關節速度)

描述

jvel 結構體類型用於描述機器人各關節速度值。

定義

```
struct jvel
{
double jv1
double jv2
double jv3
double jv4
double jv5
double jv6
double ev1
double ev2
double ev3
double ev4
double ev5
double ev6
}
```

成員

jvel 結構體類型的成員詳見表 2-12。

表 2-12 jvel 結構體類型的成員

名稱	說明
jv1~jv6	類型：double
	機器人 1 軸~6 軸的軸速度，單位：rad/s
ev1~ev6	類型：double
	機器人外 1 軸~外 6 軸的軸速度，單位：rad/s

用法舉例

```
jvel jvel1 = {jv1 1, jv3 2, ev1 3}
print jvel1 //輸出“{1,9e+09,2,9e+09,9e+09,9e+09,3,9e+09,9e+09,9e+09,9e+09,9e+09}”
jvel jvel2 = {0,1,2,3,4,5,1}
print jvel2 //輸出“{0,1,2,3,4,5,1,9e+09,9e+09,9e+09,9e+09,9e+09,9e+09}”
```

2.4.12 Centroid_Pos(工具負載質心)

描述

Centroid_Pos 結構體類型用於描述工具負載質心的位置。

格式

struct Centroid_Pos
{
double x
double y
double z
}

參數

Centroid_Pos 指令的參數詳見表 2-13。

表 2-13 Centroid_Pos 指令的參數

名稱	說明
x	類型：double
	工具負載質心在參考坐標系中坐標的 x 分量，單位毫米。
y	類型：double
	工具負載質心在參考坐標系中坐標的 y 分量，單位毫米。
z	類型：double
	工具負載質心在參考坐標系中坐標的 z 分量，單位毫米。

用法舉例

Centroid_Pos cen_pos = {10, 20, 30}
print cen_pos //輸出“{10, 20, 30}”

2.4.13 Inertia_Tensor(工具負載慣性主軸方向)

描述

Inertia_Tensor 結構體類型用於描述工具負載的慣性參數。

格式

struct Inertia_Tensor
{
double Ixx
double Ixy
double Ixz
double Iyy
double Iyz
double Izz

}

參數

Inertia_Tensor 指令的參數詳見表 2-15。

表 2-14 Inertia_Tensor 指令的參數

名稱	說明
Ixx	類型：double
	負載慣性矩陣的 xx 分量，單位 g*mm^2。
Ixy	類型：double
	負載慣性矩陣的 xy 分量，單位 g*mm^2。
Ixz	類型：double
	負載慣性矩陣的 xz 分量，單位 g*mm^2。
Iyy	類型：double
	負載慣性矩陣的 yy 分量，單位 g*mm^2。
Iyz	類型：double
	負載慣性矩陣的 yz 分量，單位 g*mm^2。
Izz	類型：double
	負載慣性矩陣的 zz 分量，單位 g*mm^2。

用法舉例

Inertia_Tensor I_T = {10, 20, 30,40,50,60}

print I_T //輸出“{10, 20, 30,40,50,60}”

2.4.14 ToolInertiaPara(工具負載類型)

描述

ToolInertiaPara 結構體類型包含了負載質量、質心、慣性主軸方向及轉動慣量。

格式

struct ToolInertiaPara

{

double m

Centroid_Pos centroidpos

Inertia_Tensor inertiatensor

}

參數

ToolInertiaParah 指令的參數詳見表 2-15。

表 2-15 ToolInertiaPara 指令的參數

名稱	說明
m	類型：double
	工具負載質量，單位 g。
centroidpos	Centroid_Pos
	工具負載質心位置
inertiatensor	Inertia_Tensor
	描述工具負載的慣性參數

用法舉例

```
ToolInertiaPara tip //定義工具負載
tip.m = 30 //質量
tip.centroid_pos = {15, 25, 100} //定義質心位置
tip. Inertia_Tensor = {10, 20, 30,40,50,60}//定義慣性主軸方向
print tip //輸出“{30,{15,25,100},{10, 20, 30,40,50,60}}”
```

2.5 結構體類型（適用於 SCARA 機器人）

ARL 中支持系統預定義的結構體類型，結構體類型是由基本類型組成的複合類型。

2.5.1 結構體變量的聲明，初始化及引用

以 pos 類型為例，空間的任意一個點坐標由 x，y，z 3 個坐標分量組成，所以 pos 類型，也就是空間點坐標類型由 3 個 double 類型的子分量組成。

pos 類型的定義如下：

```
struct pos
{
double x
double y
double z
}
```

結構體類型變量的聲明與基本類型變量的聲明相同，格式為：

變量類型 變量名

例如：

```
pos a
```

聲明瞭一個名為 a 的 pos 類型變量，如果我們用 print 指令列印 a 的值：

```
print a
```

系統將輸出：

{9e+09,9e+09,9e+09}

逗號隔開的 3 個數字分別為 a 變量中每個分量的值，可見，默認情況下，pos 類型變量 a 的分量 x，y，z 分別被初始化為 9e+09，9e+09，9e+09。

可以在變量聲明的同時對變量進行初始化。結構體變量初始化的格式有以下 2 種方式：

■ 初始化結構體成員的前 n 個成員，格式為：

結構體類型 變量名 ={分量 1 的值，分量 2 的值，……分量 n 的值}

■ 初始化結構體成員的某 1 個或幾個成員：

結構體類型 變量名 ={分量名字 分量值，分量名字 分量值，……}

對於以上兩種結構體變量初始化方式，未初始化的分量將被系統初始化為默認值。

程序舉例如下：

```
pos a = {1,2}
print a //輸出“{1,2,9e+09}”
pos b = {x 3,z 4}
print b //輸出“{3,9e+09,4}”
```

使用成員操作符“.”訪問一個結構體變量的成員分量，格式為：

結構體變量名.分量名

程序示例：

```
pos a
a.x = 1
a.y = 2
a.z = 3
double b = a.x
print a //輸出“{1,2,3}”
print b //輸出“1”
```

結構體支持嵌套，也就是說某個結構體類型的某個成員可能是另外一個結構體類型。考慮如下結構體類型 tool 的定義：

```
struct tool
{
frame t_frame
bool stationary
}
```

聲明一個 tool 類型的變量並輸出：

```
tool a
print a
```

系統將會輸出：

{{0,0,0,0,0,0},false}

這裡結構體變量將會以嵌套的花括號的形式輸出。同樣初始化結構體變量的時候也要以嵌套的花括號形式。

程序舉例如下：

```
tool a = {{0,0,0,0,0,0},false}
print a //輸出“{{0,0,0,0,0,0},false}”
print a.t_frame //輸出“{0,0,0,0,0,0}”
print a.t_frame.x //輸出 “0”
tool b = {{x 10},false}
print b //輸出“{{10,0,0,0,0,0},false}”
```

2.5.2 pos(空間點坐標)

描述

pos 結構體類型用於描述空間任意一點的坐標。

定義

```
struct pos
{
double x
double y
double z
}
```

成員

pos 結構體類型的成員詳見表 2-16。

表 2-16 pos 結構體類型的成員

名稱	說明
x	類型：double
	空間一點坐標的 x 分量，單位毫米
y	類型：double
	空間一點坐標的 y 分量，單位毫米
z	類型：double
	空間一點坐標的 z 分量，單位毫米

用法舉例

```
pos p1 = {0,0,0}
print p1 //輸出“{0,0,0}”
p1.x = p1.x +50
print p1 //輸出“{50,0,0}”
```

2.5.3 frame(坐標系)

描述

frame 結構體類型用於描述空間兩個直角坐標系之間的轉換關係。

定義

```
struct frame
{
double x
double y
double z
double a
double b
double c
}
```

成員

frame 結構體類型的成員詳見表 2-17。

表 2-17 frame 結構體類型的成員

名稱	說明
x	類型：double
	轉換後坐標系原點在原坐標系中坐標的 x 分量，單位毫米
y	類型：double
	轉換後坐標系原點在原坐標系中坐標的 y 分量，單位毫米
z	類型：double
	轉換後坐標系原點在原坐標系中坐標的 z 分量，單位毫米
a	類型：double
	轉換後坐標系姿態在原坐標系中歐拉角表達的 a 分量，單位度
b	類型：double
	轉換後坐標系姿態在原坐標系中歐拉角表達的 b 分量，單位度
c	類型：double
	轉換後坐標系姿態在原坐標系中歐拉角表達的 c 分量，單位度

歐拉角按照旋轉所繞軸的次序不同，共有 12 種不同的歐拉角表達方式，我們這裡採用的是 ZYX 型歐拉角，即從初始坐標系姿態轉換為變換坐標系姿態的順序為先繞 z 軸旋轉 a 角度，再繞新的 y 軸旋轉 b 角度，最後繞新的 x 軸旋轉 c 角度。

用法舉例

```
frame f = {10,10,0,0,0,0}
print f //輸出“{10,10,0,0,0,0}”
```

2.5.4 pose(笛卡爾目標點)

描述

pose 結構體類型使用笛卡爾坐標的方式來描述機器人各軸以及外軸的位置。

定義

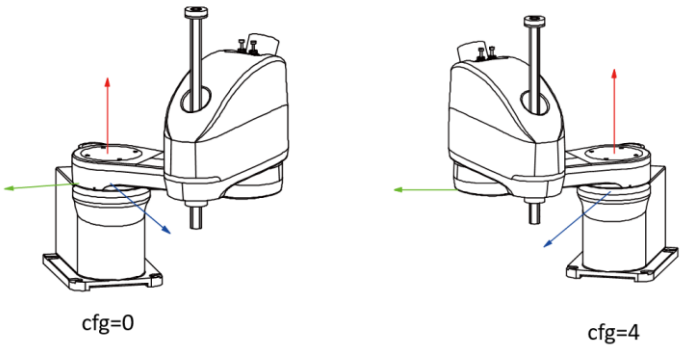
```
struct pose
{
double x
double y
double z
double a
double b
double c
int cfg
int turn
double ej1
double ej2
double ej3
double ej4
double ej5
double ej6
}
```

成員

pose 結構體類型的成員詳見表 2-18。

表 2-18 pose 結構體類型的成員

名稱	說明
x	類型：double
	機器人 TCP 點相對當前工件坐標系下的 x 方向分量，單位毫米
y	類型：double
	機器人 TCP 點相對當前工件坐標系下的 y 方向分量，單位毫米
z	類型：double
	機器人 TCP 點相對當前工件坐標系坐標的 z 分量，單位毫米
a	類型：double
	機器人工具姿態在當前工件坐標系中歐拉角表達的 a 分量，單位度
b	類型：double
	機器人工具姿態在當前工件坐標系中歐拉角表達的 b 分量，單位度
c	類型：double
	機器人工具姿態在當前工件坐標系中歐拉角表達的 c 分量，單位度
cfg	類型：int

名稱	說明
	機器人軸配置。由於機器人可能通過左手系/右手系兩種方式使 TCP 到達同一個位姿，所以這裡需要通過 cfg 參數（參考圖 2-2，0 表示左手系，4 表示右手系）來指定其中一種方式，從而唯一確定一組機器人軸位置  圖 22 cfg 參數值姿態說明
turn	類型：int
	配合 cfg 用於確定反解的軸位置。當 turn 值為-1 時，表示自動選取距離起點最近的解；當某個位值為 1 時，表示該軸選小於 0 的解；當某個位值為 0 時，表示該軸選大於 0 的解。當軸的運動範圍大於 360 度時可能會用到這個參數輔助選解，其他大部分情況下都不需要設置該值，取默認值-1 即可
ej1~ej6	類型：double
	外 1 軸~外 6 軸的位置，直線軸單位為 mm，旋轉軸單位為度

歐拉角定義參見第 2.5.3 章节。

用法舉例

```
pose p1 = {x 266,y 88,z -50,a -34,b 90,c 0,cfg 0,ej1 20}
print p1 //輸出“{266, 88, -50, -34,90,0,0,-1,20,9e+09,9e+09,9e+09,9e+09,9e+09}”
pose p2 = {308, 52, -50, -33,90,0,0,-1,10}
print p2 //輸出“{308, 52, -50, -33,90,0,0,-1,10,9e+09,9e+09,9e+09,9e+09,9e+09}”
```

2.5.5 joint(軸目標點)

描述

joint 結構體類型用於直接描述機器人各軸以及外軸的位置。

定義

```
struct joint
{
double j1
double j2
double j3
double j4
```

double ej1
double ej2
double ej3
double ej4
double ej5
double ej6
}

成員

joint 結構體類型的成員詳見表 2-19。

表 2-19 joint 結構體類型的成員

名稱	說明
j1~j4	類型：double
	機器人 1 軸~4 軸的軸位置，直線軸單位為 mm，旋轉軸單位為度
ej1~ej6	類型：double
	外 1 軸~外 6 軸的位置，直線軸單位為 mm，旋轉軸單位為度

用法舉例

joint j1={j1 0, j2 0, j3 -10, j4 0}
print to:hmi,argtoprint:j1 //輸出“{0,0，-10,0,9e+9, 9e+9, 9e+9, 9e+9, 9e+9, 9e+9, 9e+9, 9e+9}”


注意

當所有軸位置均指定時，joint 中的參數名稱可預設，但必須全部預設。例如以下的寫法就是錯誤的：j:{0,0,0,0,0,0,ej1 0}。

2.5.6 tool(工具)

描述

tool 結構體類型用於描述一個工具。

定義

struct tool
{
frame t_frame
bool stationary
}

成員

tool 結構體類型的成員詳見表 2-20。

表 2-20 tool 結構體類型的成員

名稱	說明
t_frame	類型：frame
	在工具上定義的工具坐標系
stationary	類型：bool
	■ false：指定該工具為安裝在機器人法蘭末端的工具。此時工具坐標系參照法蘭坐標系定義 ■ true：指定該工具為外部的固定工具。此時工具坐標系參照世界坐標系定義

用法舉例

```
frame f = {10,10,0,0,0,0}
tool t1
t1.t_frame = f
t1.stationary = false
print t1 //輸出“{{10,10,0,0,0,0},false}”
tool t2 = {{0,10,0,0,0,0},true}
print t2 //輸出“{{0,10,0,0,0,0},true}”
```

2.5.7 wobj(工件坐標系)

描述

wobj 結構體類型用於描述一個工件坐標系。

定義

```
struct wobj
{
frame w_frame
bool robhold
}
```

成員

wobj 結構體類型的成員詳見表 2-21。

表 2-21 wobj 結構體類型的成員

名稱	**說明**
w_frame	類型：frame
	工件坐標系
robhold	類型：bool
	■ false：指定工件坐標系參照世界坐標系定義 ■ true：指定該工件由機器人抓取。此時工件坐標系參照法蘭坐標系定義

用法舉例

```
frame f = {10,10,0,0,0,0}
```

```
wobj w1
w1.w_frame = f
print w1 //輸出“{{10,10,0,0,0,0},false}”
wobj w2 = {{0,10,0,0,0,0},false}
print w2 //輸出“{{0,10,0,0,0,0},false}”
```

2.5.8 speed(速度)

描述

speed 結構體類型用於描述一條運動指令的速度參數。

定義

```
struct speed
{
double per
double tcp
double ori
double exj
double exl
}
```

成員

speed 結構體類型的成員詳見表 2-22。

表 2-22 speed 結構體類型的成員

名稱	說明
per	類型：double
	速度百分比，ptp 和 movej 指令使用該速度參數，表示軸最大速度百分比，取值範圍 0.001~100
tcp	類型：double
	TCP 點移動速度，lin 和 cir 指令使用該速度參數，表示 TCP 點平動速度，單位毫米/秒
ori	類型：double
	工具坐標系姿態轉動速度，lin 和 cir 指令使用該速度參數，表示 TCP 點轉動速度，單位度/秒
exj	類型：double
	旋轉外軸速度。當有旋轉外軸時，移動時使用該速度參數，單位度/秒
exl	類型：double
	直線外軸速度。當有直線外軸時，移動時使用該速度參數，單位毫米/秒

用法舉例

```
speed v = {per 10}
print v //輸出“{10,9e+09,9e+09,9e+09,9e+09,9e+09}”
```

2.5.9 slip(平滑參數)

描述

slip 結構體類型用於描述一條運動指令的平滑參數。

定義

```
struct slip
{
double pdis
double ejdis
double eldis
double odis
double perdis
}
```

成員

slip 結構體類型的成員詳見表 2-23。

表 2-23 slip 結構體類型的成員

名稱	說明
pdis	類型：double
	pos distance，lin 和 cir 指令使用這項平滑參數，表示 TCP 距離目標點多少 mm 時開始平滑，單位毫米
ejdis	類型：double
	ex-joint distance，有旋轉型外軸的運動指令使用這項平滑參數，表示最慢的外軸距離目標點多少度時開始平滑，外軸為旋轉軸時，單位為度；外軸為直線軸時，單位為毫米
eldis	類型：double
	exlinearjoint distance，有移動型外軸的運動指令使用這項平滑參數，表示最慢的外軸距離目標點多少毫米時開始平滑，外軸為旋轉軸時，單位為度；外軸為直線軸時，單位為毫米
odis	類型：double
	orientation distance，lin 和 cir 指令使用這項平滑參數，表示姿態距離目標點多少角度時開始平滑，單位度
perdis	類型：double
	Percentage distance, 所有運動指令均可以使用此平滑參數。當此平滑參數生效時，其他平滑參數將會無效。表示軌跡在可平滑段開啟平滑的百分比，100%為完全平滑，0%為不平滑

用法舉例

```
slip s = {100,2,3,4,5,5}
print s //輸出“{100,2,3,4,5,5}”
```

注意

- 當 pdis 參數小於等於 0 時，表示使用它的運動語句不平滑。
- 請儘量在初始化結構體變量時將結構體的所有成員都寫全，以防止由於用戶的疏忽導致實際的運動軌跡或者運動速度和用戶預期不符，進而導致一些不必要的損失或傷害。
- perdis 優先級最高，在有 perdis 的情況下，其餘參數無效；當輸入的 pdis，odis，ejdis 生效時，平滑點更靠近目標點的值生效。

2.5.10 jvel(關節速度)

描述

jvel 結構體類型用於描述機器人各關節速度值。

定義

```
struct jvel
{
double jv1
double jv2
double jv3
double jv4
double ev1
double ev2
double ev3
double ev4
double ev5
double ev6
}
```

成員

jvel 結構體類型的成員詳見表 2-24。

表 2-24 jvel 結構體類型的成員

名稱	說明
jv1~jv4	類型：double
	機器人 1 軸~4 軸的軸速度，單位：rad/s
ev1~ev6	類型：double
	機器人外 1 軸~外 6 軸的軸速度，單位：rad/s

用法舉例

```
jvel jvel1 = {jv1 1, jv3 2, ev1 3}
print jvel1 //輸出“{1,9e+09,2,9e+09,9e+09,9e+09,3,9e+09,9e+09,9e+09,9e+09,9e+09,9e+09}”
jvel jvel2 = {0,1,2,3,4}
print jvel2 //輸出“{0,1,2,3,4, 9e+09,9e+09,9e+09,9e+09,9e+09,9e+09,9e+09,9e+09}”
```

2.5.11 control(門型動作的控制參數)

描述

control 結構體類型用於描述門型動作直升和直降階段的速度、加速度、減速度。

定義

```
struct control
```

```
{
double rising_vel
double rising_acc
double rising_dec
double falling_vel
double falling_acc
double falling_dec
}
```

成員

control 結構體類型的成員詳見表 2-25。

表 2-25 control 結構體類型的成員

名稱	說明
rising_vel	類型：double
	門型運動上升階段的速度百分比，0 表示默認速度
rising_acc	類型：double
	門型運動上升階段的加速度百分比，0 表示默認加速度
rising_dec	類型：double
	門型運動上升階段的減速度百分比，0 表示默認減速度
falling_vel	類型：double
	門型運動下降階段的速度百分比，0 表示默認減速度
falling_acc	類型：double
	門型運動下降階段的加速度百分比，0 表示默認減速度
falling_dec	類型：double
	門型運動下降階段的減速度百分比，0 表示默認減速度

用法舉例

```
control ctr1 = { rising_vel 0, rising_acc 0, rising_dec 0, falling_vel 0, falling_acc 0, falling_dec 100}
print ctr1 //輸出”{0, 0, 0, 0, 0, 100}”
```

2.6 枚舉類型

ARL 支持系統預定義枚舉類型，枚舉類型變量的值只能從固定的幾個值中選取。

例如如下枚舉類型定義：

```
enum $VEL_PROFILE
{
T_type,
S_type,
O_type
}
```

該枚舉類型定義了速度曲線的不同類型，這裡 T_type，S_type，O_type 分別代表 T 型速度曲線、S 型速度曲線和 O 型速度曲線。枚舉類型是一種特殊的整型，T_type，S_type，O_type 實際分別對應整型的 0，1，2。

引用枚舉變量的格式為：

枚舉類型名::枚舉常量名

例如：

```
print $VEL_PROFILE::S_TYPE //輸出“S_TYPE”
```

通常情況下，程序中可以省略枚舉類型名，直接使用枚舉常量名來引用枚舉常量。

例如：

```
print S_TYPE //輸出“S_TYPE”
```

但是當程序中定義了與枚舉常量名相同的變量名時，系統將優先將該名字識別為變量名而不是枚舉常量名。

例如：

```
int S_TYPE = 6
print S_TYPE //輸出“6”
```

此時，引用枚舉常量時則不能省略枚舉類型名。

例如：

```
int S_TYPE = 6
print $VEL_PROFILE::S_TYPE //輸出“S_TYPE”
```

2.6.1 $VEL_PROFILE(速度曲線)

描述

$VEL_PROFILE 枚舉類型包含了所有運動軌跡支持的速度曲線類型。

定義

```
enum $VEL_PROFILE
{
T_type,
S_type,
O_type
}
```

成員

$VEL_PROFILE 枚舉類型的成員詳見表 2-26。

表 2-26 $VEL_PROFILE 枚舉類型的成員

名稱	說明
T_type	T 型速度曲線

名稱	說明
S_type	S 型速度曲線
O_type	O 型速度曲線

用法舉例

$VEL_PROFILE = S_type //指定以下運動指令採用 S 型速度規劃

movej j:j1,vp:5%,sp:-1%

關於速度曲線更詳細內容請參見相關運動指令。

2.6.2 printto(輸出定向)

描述

printto 枚舉類型包含了所有 print 指令可以設置的輸出定向。

定義

enum printto

{

off,

hmi,

file,

console

}

成員

printto 枚舉類型的成員詳見表 2-27。

表 2-27 printto 枚舉類型的成員

名稱	說明
off	關閉輸出，即不輸出到任何地方
hmi	輸出到 HMI(人機界面)的消息欄
file	輸出到指定文件
console	輸出到終端，用於開發調式，一般用戶不會用到此設置項

用法舉例

print to:hmi, "hello world" //HMI 消息欄輸出“hello world”

關於輸出定向更詳細內容參見。

2.6.3 num_base(輸出數制)

描述

num_base 枚舉類型包含了所有 print 指令可以設置的輸出數制。

定義

```
enum num_base
{
hex,
dec
}
```

成員

num_base 枚舉類型的成員詳見表 2-28。

表 2-28 num_base 枚舉類型的成員

名稱	說明
hex	16 進制
dec	10 進制

用法舉例

```
print hex,ffh //輸出“ff”
print dec,ffh //輸出“255”
```

關於輸出數制更詳細內容請參見。

2.6.4 stoptype(停止類型)

描述

stoptype 枚舉類型包含了所有運動軌跡的停止方式。

定義

```
enum stoptype
{
general,
fast
}
```

成員

stoptype 枚舉類型的成員詳見表 2-29。

表 2-29 stoptype 枚舉類型的成員

名稱	說明
general	普通減速停止方式
fast	快速減速停止方式，是普通減速停止方式加速度的 5 倍。

用法舉例

```
stopmove fast //以快速停止方式停止當前運動
```

詳細信息參見。

2.6.5 controlmode(控制模式)

描述

controlmode 枚舉類型定義了系統的控制模式。

定義

enum controlmode
{
MANUAL,
AUTO,
MANUFAST
}

成員

controlmode 枚舉類型的成員詳見表 2-30。

表 2-30 controlmode 枚舉類型的成員

名稱	說明
MANUAL	手動低速模式
AUTO	AUTO
MANUFAST	手動高速模式

2.6.6 stopbits(停止位)

描述

stopbits 枚舉類型定義了串口通信中的停止位類型。

定義

enum stopbits
{
one,
two
}

成員

stopbits 枚舉類型的成員詳見表 2-31。

表 2-31 stopbits 枚舉類型的成員

名稱	說明
one	表示 1 位停止位
two	表示 2 位停止位

2.6.7 parity(奇偶校驗類型)

描述

parity 枚舉類型定義了串口通信中的奇偶校驗類型。

定義

```
enum parity
{
none,
odd,
even
}
```

成員

parity 枚舉類型的成員詳見表 2-32。

表 2-32 parity 枚舉類型的成員

名稱	說明
none	表示不使用奇偶校驗位
odd	表示使用奇校驗
even	表示使用偶校驗

2.6.8 weaveshape(疊加軌跡類型)

描述

weaveshape 枚舉類型包含了所有疊加軌跡支持的圖形類型。

定義

```
enum weaveshape
{
simple,
V_shape,
triangle,
ellipse,
eight
}
```

成員

weaveshape 枚舉類型的成員詳見表 2-33。

表 2-33 weaveshape 枚舉類型的成員

名稱	說明
simple	橫擺
V_shape	V 型擺
triangle	空間三角擺
ellipse	橢圓擺
eight	“8”字擺

用法舉例

weavedata weavedata = {V_shape,2,15,90} //指定該疊加圖形為 V 型擺

2.6.9 weaverotaxis(疊加軌跡擺動平面偏轉軸)

描述

weaverotaxis 枚舉類型包含了疊加軌跡支持的擺動平面偏轉所參考的坐標軸。

定義

```
enum weaverotaxis
{
rot_x,
rot_y,
rot_z
}
```

成員

weaverotaxis 枚舉類型的成員詳見表 2-34。

表 2-34 weaverotaxis 枚舉類型的成員

名稱	說明
rot_x	參考 x 軸偏轉
rot_y	參考 y 軸偏轉
rot_z	參考 z 軸偏轉

用法舉例

weavedata weavedata = {V_shape,2,15,90, rot_x,10} //指定該疊加軌跡的擺動平面參考 x 軸進行偏轉

2.7 其他類型

2.7.1 socket(套接字類型)

描述

socket 套接字類型用於通過網口與外部設備通信。該數據類型配合 connect，accept，write，read，readuntil 等函數使用。

用法舉例

參見函數 connect，accept，write，read，readuntil。

2.7.2 iodev(IO 設備類型)

描述

IO 設備類型用於 IO 讀寫，例如通過串口讀寫與外部設備通信。該數據類型配合 open，write，read，readuntil 等函數使用。

用法舉例

參見函數 open，write，read，readuntil。

2.8 數組

相同類型的變量的集合，放在一起處理比較方便。這種情況下，可以使用數組。數組，就是相同數據類型的元素按一定順序排列的集合。數組可以是一維的、也可以是多維的：

- 一維數組

元素不是數組的數組。

- 多維數組

二維數組及以上的數組。以數組作為元素的數組是二維數組，以二維數組為元素的數組是三維數組，當然也可以生成更高的數組。

2.8.1 數組的聲明（使用數組前的準備）

聲明一個數組變量的格式為：

變量類型名 數組變量名[第 1 維大小][第 2 維大小]……

例如：

double a[10] //聲明瞭一個 double 類型的一維數組，數組大小為 10
double b[2][3] //聲明瞭一個 double 類型的二維數組，第 1 維大小為 2，第 2 維大小為 3。

2.8.2 數組初始化

數組在聲明的同時可以進行初始化，初始化數據的格式為：

{第 1 維的第 1 個數據值，第 1 維的第 2 個數據值，……第 n 維的第 1 個數據值，第 n 維的第 2 個數據值……第 n 維的第 n 個數據值}

例如：

double a[3] = {1,2,3}
double b[2][3] = {1,2,3,1,2,3}

允許只初始化數組的前 m 個元素，未初始化的元素保持默認值。

2.8.3 訪問數組（數組的使用方法）

使用下標來訪問數組中的元素，數組每一維的下標從 0 開始。

例如：

```
double a[3] = {1,2,3}
double b[2][3] = {1,2,3,4,5,6}
print a[0] //輸出“1”
print b[0][1] //輸出“2”
double c = a[0] + b[0][1]
print c //輸出“3”
```


注意

一維數組的下標限製為 10000，多維數組下標乘積限製為 10000。

2.8.4 數組管理結構體變量

結構體變量也可以組成數組來管理。

例如：

```
pos p[3] = {{0,0,0},{10,10,10},{20,20,20}}
print p[0] //輸出“{0,0,0}”
print p[1].x //輸出“10”
```

2.9 運算符

2.9.1 算術運算符

算術運算符的分類及作用詳見表 2-35：

表 2-35 算數運算符

操作符	作用	適用數據類型
+	加	int,byte,double,string,joint
-	減/取負	int,byte,double,joint
*	乘	int,byte,double,frame,pose
/	除	int,byte,double,
%	取餘	int,byte,double,
--	自減 1	int
++	自加 1	int

string 類型的數據相加表示字元串連接。

frame 類型的數據相乘表示坐標系變換。

各種算術運算符使用舉例如下：

```
int sa = 1
```

int sb = 2

int sc = -sb //取負

int sac = sa * sc //乘法

sac = -2

“++”和“--”用法如下：

“x=m++”表示將 m 的值賦給 x 後, m 加 1；

“x=++m”表示 m 先加 1 後，再將新值賦給 x。

“--”操作符和“++”用法相同。

2.9.2 按位運算符

按位運算符的分類及作用詳見表 2-36：

表 2-36 按位運算符

操作符	作用	適用數據類型
<<	向左移位	int,byte
>>	向右算術移位	int,byte
&	按位與	int,byte
|	按位或	int,byte
^	按位異或	int,byte
~	按位取非	int,byte

各種按位運算符使用舉例如下：

int A = 00111100b

int B = 00001101b

print A," ",A&B," ",A|B," ",A^B," ",~A," ",A<<2," ",A>>2 //輸出“ 60 12 61 49 -61 240 15”

2.9.3 邏輯運算符

邏輯運算符的分類及作用詳見表 2-37：

表 2-37 邏輯運算符

操作符	**作用**	**適用數據類型**
&&	邏輯與	int,byte,bool
||	邏輯或	int,byte,bool
<	小於	int,byte,double,string
>	大於	int,byte,double,string
<=	小於等於	int,byte,double,string
>=	大於等於	int,byte,double,string
==	等於	int,byte,double,bool,string

操作符	作用	適用數據類型
!=	不等於	int,byte,double,bool,string
!	取邏輯非	int,byte, bool

邏輯與“&&”表達式為真的條件是兩邊的結果都為真，而邏輯或“||”為真的條件是兩邊只要有一個條件為真即可。

各種邏輯運算符使用舉例如下：

```
int i = 0
while(getdi(1) && !getdi(2))
if(i<100)
i++
endif
endwhile
```

2.9.4 賦值運算符

賦值運算符的分類及作用詳見表 2-38：

表 2-38 賦值運算符

操作符	作用
=	賦值
+=	加等
-=	減等
*=	乘等
/=	除等
%=	取模等
<<=	左移等
>>=	算術右移等
&=	按位與等
|=	按位或等

所有數據類型均支持賦值操作，XX=運算符支持的數據類型與 XX 運算符支持的數據類型一致。

“a += b”等同於“a = a + b”。其他同理。

各種賦值運算符使用舉例如下：

```
int a = 1
print a //輸出“1”
a += 1
print a //輸出“2”
a <<= 2
```

print a //輸出“8”

2.9.5 其他運算符

其他運算符的分類及作用詳見表 2-39：

表 2-39 其他運算符

操作符	作用
[ ]	數組下標 數組下標操作符的用法參見“2.8”一節
( )	圓括號 取結構體成員變量操作符參見“2.4”一節
.	取結構體成員變量

圓括號用於指定運算的優先級，如：

int num = (1+2)*3

print num //輸出“9”

2.9.6 運算符優先級

不同運算符的優先級順序如下表 2-40 所示，該表中優先級數字越小，表示優先級越高。同一優先級的運算符，運算次序由結合方向所決定。

表 2-40 運算符優先級

優先級	運算符	名稱或含義	使用形式	結合方向	說明
1	[ ]	數組下標	數組名[常量表達式]	左到右	-
	( )	圓括號	（表達式）/函數名(形參表)		-
	.	結構體變量成員選擇	結構體變量.成員名		-
2	-	負號運算符	-表達式	右到左	單目運算符
	++	自增運算符	++變量名/變量名++		單目運算符
	--	自減運算符	--變量名/變量名--		單目運算符
	!	邏輯非運算符	!表達式		單目運算符
	~	按位取反運算符	~表達式		單目運算符
3	/	除	表達式/表達式	左到右	雙目運算符
	*	乘	表達式*表達式		雙目運算符
	%	餘數（取模）	整型表達式/整型表達式		雙目運算符
4	+	加	表達式+表達式	左到右	雙目運算符
	-	減	表達式-表達式		雙目運算符
5	<<	左移	變量<<表達式	左到右	雙目運算符
	>>	右移	變量>>表達式		雙目運算符
6	>	大於	表達式>表達式	左到右	雙目運算符

優先級	運算符	名稱或含義	使用形式	結合方向	說明
	>=	大於等於	表達式>=表達式		雙目運算符
	<	小於	表達式<表達式		雙目運算符
	<=	小於等於	表達式<=表達式		雙目運算符
7	==	等於	表達式==表達式	左到右	雙目運算符
	!=	不等於	表達式!= 表達式		雙目運算符
8	&	按位與	表達式&表達式		雙目運算符
9	^	按位異或	表達式^表達式	左到右	雙目運算符
10	|	按位或	表達式|表達式		雙目運算符
11	&&	邏輯與	表達式&&表達式	左到右	雙目運算符
12	||	邏輯或	表達式||表達式	左到右	雙目運算符
13	=	賦值運算符	變量=表達式	右到左	-
	/=	除後賦值	變量/=表達式		
	*=	乘後賦值	變量*=表達式		
	%=	取模後賦值	變量%=表達式		
	+=	加後賦值	變量+=表達式		
	-=	減後賦值	變量-=表達式		
	<<=	左移後賦值	變量<<=表達式		
	>>=	右移後賦值	變量>>=表達式		
	&=	按位與後賦值	變量&=表達式		
	^=	按位異或後賦值	變量^=表達式		
	|=	按位或後賦值	變量|=表達式		


注意

實際應用中如果不確定運算符的優先級大小，請儘量使用圓括號顯示表達希望的運算順序，否則表達式的計算結果可能和用戶的預期不符。

2.10 變量的作用域

按變量的作用域來劃分，變量可以分為局部變量，全局變量和系統變量。ARL 中，在函數內部聲明的變量為局部變量，在函數外部聲明的變量默認為全局變量，系統變量為系統預定義的變量，不需要聲明，可以直接通過”$”符號引用。

局部變量只在當前函數內有效。

例如下麵的程序：

```
func void myfunc()
for(int i=1;i<2;i++)
int a = 1 //該變量只在 for 和 endfor 範圍內有效
```

```
print a //這裡可以訪問到 a
endfor
print a //這裡將報出變量 a 不存在
endfunc
```

在同一個作用域範圍內聲明的變量名不能重覆，否則系統將報出變量重覆定義的錯誤。但是在不同的作用域範圍內聲明的變量名可以重覆，此時使用該名字訪問的變量是最近作用域範圍內的變量。如：

```
func void myfunc()
int a = 6
for(int i=1;i<2;i++)
int a = 1 //該變量只在 for 和 endfor 範圍內有效
print a //這裡輸出“1”
endfor
print a //這裡輸出“6”
endfunc
```

全局變量在聲明後的當前程序中都可以訪問。如：

```
int a = 1
func void myfunc()
a = 2
print a //這裡輸出“2”
endfunc
func void main()
print a //這裡輸出“1”
myfunc()
endfunc
```

系統變量是任意通道、函數、程序都共用的，如：

```
func void main()
$DFSPPED = {10,50,5,5,5} //這裡將默認速度設置為：{10,50,5,5,5}
print $DFSPPED //這裡輸出“{10,50,5,5,5}”
……
endfunc
```

3 順序指令

3.1 順序指令的一般格式

除流程控制指令以外，ARL 順序指令具有如下所示的一般格式：

指令名 參數名：參數值，參數名：參數值，……

其中有些參數是強制參數，如果不寫，系統會報出指令格式錯誤。有些參數是可選參數，可選參數允許預設，預設時將取默認值或者保持之前設定的值。以下指令格式中[ ]內的參數表示該參數為可選參數。

某些參數的參數名部分也可以預設而只寫參數值即可，而某些參數的參數名部分則不能預設。以下指令格式中使用“參數名：”的形式表示該參數的參數名不能預設。在允許參數名預設的語法規則有一種特例，當參數值是以花括號“{ }”形式表達的結構體常量而不是變量時，參數名部分不能預設，否則系統將會報出指令格式錯誤。

3.2 運動指令（適用於六軸機器人）

3.2.1 movej(移動軸)

描述

movej 指令用於將機器人軸或外軸移動到一個指定的軸位置。所有軸同時到達目標軸位置。

格式

movej j:,[v: | vp:],[s: | sp: | sl:],[t:],[dura:]

參數

movej 指令的參數詳見表 3-1。

表 3-1 movej 指令的參數

名稱	說明
j	數據類型：joint
	該參數指定機器人各軸和外軸的目標點位置。參數中值為 9e+09 的結構體分量將保持起點位置不變。程序中如果第一條執行的運動指令為 movej，則該 movej 指令的目標點 j1~j6 以及配置的外軸均不允許預設
v	數據類型：speed
	該參數指定運動速度。movej 指令使用 speed 結構體變量中的 per 和 exj 分量，分別用於指定軸運動速度百分比和外軸運動速度。這裡如果預設了 v 參數，則默認使用系統變量$DFSPEED 作為 v 參數的值；如果指定了 v 參數，則成員分量的值會被更新到$DFSPEED 對應的分量中 在簡化程式設計中，v 可由速度百分比參數 vp 代替，格式見用法舉例
vp	數據類型：double
	該參數指定運動速度百分比。可替代速度參數 v，用於不需要精確指定速度大小的場合。格式為 vp:10%，表示當前行速度是機器人最大速度的 10%
s	數據類型：slip

名稱	說明
	該參數指定平滑距離。movej 指令使用 slip 結構體的 perdis 和 ejdis 分量，用於描述脫離原軌跡進入平滑軌跡點。其中，perdis 用於指定距離目標點百分比，ejdis 用於指定外軸距離目標點軸位置角度。這裡如果預設 s 參數，則默認使用系統變量$DFSLIP 作為 s 參數的值；如果指定了 s 參數，則成員分量的值會被更新到$DFSLIP 對應的分量中 在簡化程式設計中，s 可由平滑百分比參數 sp 代替，格式見用法舉例 註： ■ 當 perdis>=0 時，其餘分量將被忽略； ■ 當同時指定 ejdis 時，更靠近目標點的分量將會被選用； ■ 當上述分量均為 0 時，表示不平滑，但插補點不一定與目標點重合； ■ 當上述分量均小於 0 時，表示不平滑，目標點肯定會有插補點，即準停
sp	數據類型：double
	該參數指定平滑百分比。可替代平滑參數 s，用於不需要精確指定平滑大小的場合。格式為 sp:10%，表示當前行目標點的平滑距離是最大平滑距離的 10%
sl	數據類型：double
	該參數指定平滑距離。可替代平滑參數 s，用於需要精確指定平滑大小的場合。格式為 sl:10mm，表示當前行的平滑距離是 10mm
t	該參數指定工具。用戶需要在法蘭末端安裝的工具上自定義工具坐標系，該工具坐標系與工具固連，目標點 p 指定的就是工具坐標系在 w 參數指定的坐標系中的位姿。這裡如果預設了 t 參數，則默認使用系統變量$DFTOOL 作為 t 參數的值；如果指定了 t 參數，則成員分量的值會被更新到$DFTOOL 對應的分量中，該參數用於限制手動低速模式下運行程序時，工具末端的線速度不大於 250 毫米/秒。
dura	數據類型：double
	該參數指定軌跡時間。用戶可以直接指定運動時間而不是運動速度。如果用戶指定了 dura 參數，系統將忽略 speed 參數，而是通過自動調整速度來滿足 dura 參數指定的時間要求。dura 參數單位是秒

用法舉例

```
//模糊指定速度和平滑大小
joint j1 = {j1 10,j2 20,j3 30,j4 40,j5 50,j6 60}
movej j:j1,vp:5%,sp:5%
movej j:{j1 10,j2 20,j3 30,j4 40,j5 50,j6 60}
movej j:{j1 20,ej1 30}
joint p = {0,20,30,40,50,60}
speed v = {per 50}
movej p,v
```

3.2.2 ptp(點到點)

描述

ptp 指令用於將機器人從一個點快速運動到另一個點而又不要求 TCP 點所走軌跡形狀時。所有軸同時到達目標點。

格式

```
ptp p:,[ v: | vp: ],[ s: | sl: | sp: ],[ t: ],[ w: ],[ dura: ]
```

參數

ptp 指令的參數詳見表 3-2。

表 3-2 ptp 指令的參數

名稱	說明
p	數據類型：pose
	該參數指定機器人 TCP 目標位姿和外軸的目標點位置。參數中值為 9e+09 的結構體分量將保持起點位置不變。cfg 分量預設默認值為 0，turn 分量預設默認值為-1。程序中如果第一條執行的運動指令為 ptp，則該 ptp 指令的目標點 X,Y,Z,A,B,C 以及配置的外軸均不允許預設
v	數據類型：speed
	該參數指定運動速度。ptp 指令使用 speed 結構體變量中的 per 和 exj 分量，分別用於指定軸運動速度百分比和外軸運動速度。這裡如果預設了 v 參數，則默認使用系統變量$DFSPEED 作為 v 參數的值；如果指定了 v 參數，則成員分量的值會被更新到$DFSPEED 對應的分量中 在簡化程式設計中，v 可由速度百分比參數 vp 代替，格式見用法舉例
vp	數據類型：double
	該參數指定運動速度百分比。可替代速度參數 v，用於不需要精確指定速度大小的場合。格式為 vp:10%，表示當前行速度是機器人最大速度的 10%
s	數據類型：slip
	該參數指定平滑距離。PTP 指令使用 slip 結構體的 perdis 和 ejdis 分量，用於描述脫離原軌跡進入平滑軌跡點。其中，perdis 用於指定距離目標點百分比，ejdis 用於指定外軸距離目標點軸位置角度。這裡如果預設 s 參數，則默認使用系統變量$DFSLIP 作為 s 參數的值；如果指定了 s 參數，則成員分量的值會被更新到$DFSLIP 對應的分量中 在簡化程式設計中，s 可由平滑百分比參數 sp 代替，格式見用法舉例 註： ■ 當 perdis>=0 時，其餘分量將被忽略； ■ 當同時指定 ejdis 時，更靠近目標點的分量將會被選用； ■ 當上述分量均為 0 時，表示不平滑，但插補點不一定與目標點重合； ■ 當上述分量均小於 0 時，表示不平滑，目標點肯定會有插補點，即準停
sp	數據類型：double
	該參數指定平滑百分比。可替代平滑參數 s，用於不需要精確指定平滑大小的場合。格式為 sp:10%，表示當前行目標點的平滑距離是最大平滑距離的 10%
sl	數據類型：double
	該參數指定平滑距離。可替代平滑參數 s，用於需要精確指定平滑大小的場合。格式為 sl:10mm，表示當前行的平滑距離是 10mm
t	數據類型：tool 結構體
	該參數指定工具。用戶需要在法蘭末端安裝的工具上自定義工具坐標系，該工具坐標系與工具固連，目標點 p 指定的就是工具坐標系在 w 參數指定的坐標系中的位姿。這裡如果預設了 t 參數，則默認使用系統變量$DFTOOL 作為 t 參數的值；如果指定了 t 參數，則成員分量的值會被更新到$DFTOOL 對應的分量中
w	數據類型：wobj 結構體
	該參數指定工件坐標系。用戶可以自定義工件坐標系。目標點 p 指定的就是工具坐標系在 w 參數指定的坐標系中的位姿。在某些情況下，用戶可能通過在特定的坐標系中程式設計更方便。另外，當工件移動位置時，只需重新標定工件坐標系即可，不需要修改用戶程序。這裡如果預設了

名稱	說明
	w 參數，則默認使用系統變量$DFWOBJ 作為 w 參數的值；如果指定了 w 參數，則成員分量的值會被更新到$DFWOBJ 對應的分量中
dura	數據類型：double
	該參數指定軌跡時間。用戶可以直接指定運動時間而不是運動速度。如果用戶指定了 dura 參數，系統將忽略 speed 參數，通過自動調整速度來滿足 dura 參數指定的時間要求。dura 參數單位是秒

以上結構體數據詳細信息請參考。

用法舉例

```
pose p = {x 1500,y 500,z 500,a 0,b 90,c 0,cfg 0}
//模糊指定速度和平滑大小
ptp p:p1,vp:5%,sp:5%
ptp p,v:{per 10}
ptp p:{x 1000,y 500,z 500,a 0,b 90,c 0}
ptp p,dura:10 //10s 運動至 p 點
tool t = {{x 0,y 0,z 10,a 0,b 0,c 0}}
wobj w = {{x 1000,y 0,z 500,a 0,b 10,c 0}}
speed v = {per 10,tcp 50,ori 5,exj 10}
pose p2 = {x 10,y 10,z 10,a 0,b 90,c 0,cfg 0}
ptp p2,v,t,w
```

3.2.3 lin(直線運動)

描述

lin 指令用於將機器人 TCP 點沿直線路徑運動到目標點位姿；位置移動和姿態轉動同步。

格式

```
lin p:,[ v: | vl: | vp: ],[ s: | sl: | sp: ],[ t: ],[ w: ],[ dura: ]
```

參數

lin 指令的參數詳見表 3-3。

表 3-3 lin 指令的參數

名稱	說明
p	數據類型：pose
	該參數指定機器人 TCP 目標位姿和外軸的目標點位置。參數中值為 9e+09 的結構體分量將保持起點位置不變。cfg 參數被忽略，與起點 cfg 參數保持一致，turn 分量預設默認值為-1
v	數據類型：speed
	該參數指定運動速度。lin 指令使用 speed 結構體中的 tcp、ori 和 exj 分量，分別用於指定 TCP 點運動速度、TCP 姿態變化速度和外軸運動速度。這裡如果預設了 v 參數，則默認使用系統變量$DFSPEED 作為 v 參數的值；如果指定了 v 參數，則成員分量的值會被更新到$DFSPEED 對應的分量中 在簡化程式設計中，v 可由速度百分比參數 vp 或速度絕對值參數 vl 代替

名稱	說明
vp	數據類型：double
	該參數指定運動速度百分比。可替代速度參數 v，用於不需要精確指定速度大小的場合。格式為 vp:10%，表示當前行速度是機器人最大速度的 10%
vl	數據類型：double
	該參數指定運動線速度。可替代速度參數 v，用於需要精確指定速度大小的場合。格式為 vl:200mm/s，表示當前行的 TCP 最大速度的 200mm/s
s	數據類型：slip
	該參數指定平滑距離。LIN 指令使用 slip 結構體的 perdis、pdis、odis 和 ejdis 分量，用於描述脫離原軌跡進入平滑軌跡點。其中，perdis 用於指定距離目標點百分比，pdis 用於指定距離目標點軸路徑距離，odis 用於指定距離目標點姿態角度，ejdis 用於指定外軸距離目標點軸位置角度。這裡如果預設 s 參數，則默認使用系統變量$DFSLIP 作為 s 參數的值；如果指定了 s 參數，則成員分量的值會被更新到$DFSLIP 對應的分量中 在簡化程式設計中，s 可由平滑百分比參數 sp 或平滑絕對值參數 sl 代替，格式見用法舉例 註： ■ 當 perdis>=0 時，其餘分量將被忽略； ■ 當同時指定 pdis、odis 和 ejdis 時，更靠近目標點的分量將會被選用； ■ 當上述分量均為 0 時，表示不平滑，但插補點不一定與目標點重合； ■ 當上述分量均小於 0 時，表示不平滑，目標點肯定會有插補點，即準停； 使用笛卡爾空間軌跡（即 LIN、CIR）平滑時，建議使用 pdis 平滑，可以使平滑軌跡在前後軌跡上均勻
sp	數據類型：double
	該參數指定平滑百分比。可替代平滑參數 s，用於不需要精確指定平滑大小的場合。格式為 sp:10%，表示當前行目標點的平滑距離是最大平滑距離的 10%
sl	數據類型：double
	該參數指定平滑距離。可替代平滑參數 s，用於需要精確指定平滑大小的場合。格式為 sl:10mm，表示當前行的平滑距離是 10mm
t	數據類型：tool 結構體
	該參數指定工具。用戶需要在法蘭末端安裝的工具上自定義工具坐標系，該工具坐標系與工具固連，目標點 p 指定的就是工具坐標系在 w 參數指定的坐標系中的位姿。這裡如果預設了 t 參數，則默認使用系統變量$DFTOOL 作為 t 參數的值；如果指定了 t 參數，則成員分量的值會被更新到$DFTOOL 對應的分量中 該參數是為了保證手動低速運行程序時限制 TCP 速度不大於 250mm/s
w	數據類型：wobj 結構體
	該參數指定工件坐標系。用戶可以自定義工件坐標系。目標點 p 指定的就是工具坐標系在 w 參數指定的坐標系中的位姿。在某些情況下，用戶可能通過在特定的坐標系中程式設計更方便。另外，當工件移動位置時，只需重新標定工件坐標系即可，不需要修改用戶程序。這裡如果預設了 w 參數，則默認使用系統變量$DFWOBJ 作為 w 參數的值；如果指定了 w 參數，則成員分量的值會被更新到$DFWOBJ 對應的分量中
dura	數據類型：double
	該參數指定軌跡時間。用戶可以直接指定運動時間而不是運動速度。如果用戶指定了 dura 參數，系統將忽略 speed 參數，而是通過自動調整速度來滿足 dura 參數指定的時間要求。dura 參數單位是秒

以上結構體數據詳細信息請參考。

用法舉例

```
pose p = {x 1500,y 500,z 500,a 0,b 90,c 0,cfg 0}
ptp p,v:{per 10}
//模糊指定速度和平滑大小
lin p:p1,vp:5%,sp:5%
//精確指定速度和平滑距離
lin p:p1, vl:100mm/s,sl:5mm
lin p:{x 1000,y 500,z 500,a 0,b 90,c 0}
lin p,dura:10
//精確指定速度、平滑結構體
tool t = {{x 0,y 0,z 10,a 0,b 0,c 0}}
wobj w = {{x 1000,y 0,z 500,a 0,b 10,c 0}}
speed v = {per 10,tcp 50,ori 5,exj 10}
slip s = {pdis 10, ejdis 10, odis 10}
pose p2 = {x 10,y 10,z 10,a 0,b 90,c 0,cfg 0}
lin p2,t,w,v,s
```

3.2.4 cir(圓弧運動)

描述

cir 指令用於將機器人 TCP 點沿圓弧路徑運動到目標點；平移運動和旋轉運動同步。

格式

```
cir m:,p:,[ v: | vl: | vp: ],[ s: | sl: | sp: ],[ t: ],[ w: ],[ CA: ],[ dura: ]
```

參數

cir 指令的參數詳見表 3-4。

表 3-4 cir 指令的參數

名稱	說明
m	數據類型：pose
	該參數指定圓弧輔助點。起點、輔助點和目標點，這 3 點唯一確定了一段圓弧。輔助點中只有 x，y，z 分量被使用，他們中值為 9e+09 的分量將保持起點位置值，其他分量被忽略
p	數據類型：pose
	該參數指定機器人 TCP 目標位姿和外軸的目標點位置。參數中值為 9e+09 的結構體分量將保持起點位置不變。cfg 參數被忽略，與起點 cfg 參數保持一致，turn 分量預設默認值為-1
v	數據類型：speed
	該參數指定運動速度。cir 指令使用 speed 結構體中的 tcp、ori 和 exj 分量，分別用於指定 TCP 點運動速度、TCP 姿態變化速度和外軸運動速度。這裡如果預設了 v 參數，則默認使用系統變量 $DFSPEED 作為 v 參數的值；如果指定了 v 參數，則成員分量的值會被更新到$DFSPEED 對應的分量中

名稱	說明
	在簡化程式設計中，v 可由速度百分比參數 vp 或速度絕對值參數 vl 代替，格式見用法舉例
vp	數據類型：double
	該參數指定運動速度百分比。可替代速度參數 v，用於不需要精確指定速度大小的場合。格式為 vp:10%，表示當前行速度是機器人最大速度的 10%
vl	數據類型：double
	該參數指定運動線速度。可替代速度參數 v，用於需要精確指定速度大小的場合。格式為 vl:500mm/s，表示當前行的 TCP 最大速度的 500mm/s
s	數據類型：slip
	該參數指定平滑距離。LIN 指令使用 slip 結構體的 perdis、pdis、odis 和 ejdis 分量，用於描述脫離原軌跡進入平滑軌跡點。其中，perdis 用於指定距離目標點百分比，pdis 用於指定距離目標點軸路徑距離，odis 用於指定距離目標點姿態角度，ejdis 用於指定外軸距離目標點軸位置角度。這裡如果預設 s 參數，則默認使用系統變量$DFSLIP 作為 s 參數的值；如果指定了 s 參數，則成員分量的值會被更新到$DFSLIP 對應的分量中 在簡化程式設計中，s 可由平滑百分比參數 sp 或平滑絕對值參數 sl 代替，格式見用法舉例 註： ■ 當 perdis>=0 時，其餘分量將被忽略； ■ 當同時指定 pdis、odis 和 ejdis 時，更靠近目標點的分量將會被選用； ■ 當上述分量均為 0 時，表示不平滑，但插補點不一定與目標點重合； ■ 當上述分量均小於 0 時，表示不平滑，目標點肯定會有插補點，即準停； 使用笛卡爾空間軌跡（即 LIN、CIR）平滑時，建議使用 pdis 平滑，可以使平滑軌跡在前後軌跡上均勻
sp	數據類型：double
	該參數指定平滑百分比。可替代平滑參數 s，用於不需要精確指定平滑大小的場合。格式為 sp:10%，表示當前行目標點的平滑距離是最大平滑距離的 10%。
sl	數據類型：double
	該參數指定平滑距離。可替代平滑參數 s，用於需要精確指定平滑大小的場合。格式為 sl:10mm，表示當前行的平滑距離是 10mm。
w	數據類型：wobj 結構體
	該參數指定工件坐標系。用戶可以自定義工件坐標系。目標點 p 指定的就是工具坐標系在 w 參數指定的坐標系中的位姿。在某些情況下，用戶可能通過在特定的坐標系中程式設計更方便。另外，當工件移動位置時，只需重新標定工件坐標系即可，不需要修改用戶程序。這裡如果預設了 w 參數，則默認使用系統變量$DFWOBJ 作為 w 參數的值；如果指定了 w 參數，則成員分量的值會被更新到$DFWOBJ 對應的分量中
CA	數據類型：double
	該參數指定圓心角（Circle Angle）。用戶可以不直接指定目標點，而通過指定圓弧轉過的圓心角的方式來指定目標點。如果用戶指定了 CA 參數，則 p 參數只用來和輔助點一起確定圓弧的幾何形狀，而不是真正的目標點，真正的目標點系統通過用戶指定的圓心角自動計算。CA 參數單位：度
dura	數據類型：double
	該參數指定軌跡時間。用戶可以直接指定運動時間而不是運動速度。如果用戶指定了 dura 參數，系統將忽略 speed 參數，而是通過自動調整速度來滿足 dura 參數指定的時間要求。dura 參數單位：秒

以上結構體數據詳細信息請參考。

用法舉例

```
pose p3={x 1000,y 200,z 500,a 90,b 0,c 90}
pose p4={x 800,y 0,z 500,a 90,b 0,c 90}
tool tool1 = {{x 0,y 10,z 15,a 0,b 90,c 0}}
cir p3,p4,tool1
//指定圓弧圓心角為 360 度
cir p3,p4,CA:360
//模糊指定速度和平滑大小
cir m:p1,p:p2,vp:5%,sp:5%
//精確指定速度和平滑距離
cir m:p3,p:p4,vl:500mm/s,sl:5mm
//精確指定速度和平滑結構體
tool t1 = {{x 0,y 0,z 10,a 0,b 0,c 0}}
wobj w1 = {{x 1000,y 0,z 500,a 0,b 10,c 0}}
speed v2 = {per 10,tcp 50,ori 5,exj 10}
slip s = { pdis 10, ejdis 10, odis 10 }
cir p3,p4,t1,w1,v2, s
```

在單步或段調試模式下，cir 指令分為兩步執行，第一步運行至輔助點，第二步運行至目標點。

雖然用戶可以通過使用參數預設來簡化程序，但仍建議用戶在編寫運動指令時儘量將參數寫全，以防止由於程式設計人員的疏忽而導致的實際的運動與用戶預期不符，進而導致一些不必要的損失或傷害。

3.2.5 ccir(連續圓弧運動)

描述

在 cir 指令中，用戶需要對經由點和終點的 2 個位置進行示教。在 ccir 指令中，只需示教一個點，但至少需要連續示教兩條 ccir 指令，用於確定圓弧路徑。

ccir 指令與 cir 指令相比具有如下特征：

- 可以在圓弧動作的經由點和終點分別指定速度和平滑。
- 可以在經由點和終點之間示教邏輯指令。但可進行示教的邏輯指令會受到限制。
- 可在兩條 ccir 之間插入 ccir，用於微調圓弧軌跡。

當出現以下情況時，無法創建圓弧，且系統報告“[12002][0]非法圓弧平面”。

- 當創建的圓弧點數小於 3 個時，無法構成圓弧。
- 當創建的圓弧上的 3 點，構成一條直線時，無法創建圓弧。
- 當 ccir 指令中出現連續的相同點時，無法創建圓弧。

ccir 指令之間不允許有停前瞻的語句，例如：ccir 指令之間有 waittime 0 的時候，會報警非法圓弧平面

格式

```
ccir p:,[ v: | vl: | vp: ],[ s: | sl: | sp: ],[ t: ],[ w: ]
```

參數

ccir 指令的參數詳見表 3-3。

表 3-5 ccir 指令的參數

名稱	說明
p	數據類型：pose
	該參數指定機器人 TCP 目標位姿和外軸的目標點位置。參數中值為 9e+09 的結構體分量將保持起點位置不變。cfg 參數被忽略，與起點 cfg 參數保持一致，turn 分量預設默認值為-1
v	數據類型：speed
	該參數指定運動速度。lin 指令使用 speed 結構體中的 tcp、ori 和 exj 分量，分別用於指定 TCP 點運動速度、TCP 姿態變化速度和外軸運動速度。這裡如果預設了 v 參數，則默認使用系統變量$DFSPEED 作為 v 參數的值；如果指定了 v 參數，則成員分量的值會被更新到$DFSPEED 對應的分量中 在簡化程式設計中，v 可由速度百分比參數 vp 或速度絕對值參數 vl 代替
vp	數據類型：double
	該參數指定運動速度百分比。可替代速度參數 v，用於不需要精確指定速度大小的場合。格式為 vp:10%，表示當前行速度是機器人最大速度的 10%
vl	數據類型：double
	該參數指定運動線速度。可替代速度參數 v，用於需要精確指定速度大小的場合。格式為 vl:200mm/s，表示當前行的 TCP 最大速度的 200mm/s
s	數據類型：slip
	該參數指定平滑距離。LIN 指令使用 slip 結構體的 perdis、pdis、odis 和 ejdis 分量，用於描述脫離原軌跡進入平滑軌跡點。其中，perdis 用於指定距離目標點百分比，pdis 用於指定距離目標點軸路徑距離，odis 用於指定距離目標點姿態角度，ejdis 用於指定外軸距離目標點軸位置角度。這裡如果預設 s 參數，則默認使用系統變量$DFSLIP 作為 s 參數的值；如果指定了 s 參數，則成員分量的值會被更新到$DFSLIP 對應的分量中 在簡化程式設計中，s 可由平滑百分比參數 sp 或平滑絕對值參數 sl 代替，格式見用法舉例 註： ■ 當 perdis>=0 時，其餘分量將被忽略； ■ 當同時指定 pdis、odis 和 ejdis 時，更靠近目標點的分量將會被選用； ■ 當上述分量均為 0 時，表示不平滑，但插補點不一定與目標點重合； ■ 當上述分量均小於 0 時，表示不平滑，目標點肯定會有插補點，即準停； 使用笛卡爾空間軌跡（即 LIN、CIR）平滑時，建議使用 pdis 平滑，可以使平滑軌跡在前後軌跡上均勻
sp	數據類型：double
	該參數指定平滑百分比。可替代平滑參數 s，用於不需要精確指定平滑大小的場合。格式為 sp:10%，表示當前行目標點的平滑距離是最大平滑距離的 10%
sl	數據類型：double
	該參數指定平滑距離。可替代平滑參數 s，用於需要精確指定平滑大小的場合。格式為 sl:10mm，表示當前行的平滑距離是 10mm
t	數據類型：tool 結構體
	該參數指定工具。用戶需要在法蘭末端安裝的工具上自定義工具坐標系，該工具坐標系與工具固連，目標點 p 指定的就是工具坐標系在 w 參數指定的坐標系中的位姿。這裡如果預設了 t 參數，則默認使用系統變量$DFTOOL 作為 t 參數的值；如果指定了 t 參數，則成員分量的值會被更新到$DFTOOL 對應的分量中 該參數是為了保證手動低速運行程序時限制 TCP 速度不大於 250mm/s

名稱	說明
w	數據類型：wobj 結構體
	該參數指定工件坐標系。用戶可以自定義工件坐標系。目標點 p 指定的就是工具坐標系在 w 參數指定的坐標系中的位姿。在某些情況下，用戶可能通過在特定的坐標系中程式設計更方便。另外，當工件移動位置時，只需重新標定工件坐標系即可，不需要修改用戶程序。這裡如果預設了 w 參數，則默認使用系統變量$DFWOBJ 作為 w 參數的值；如果指定了 w 參數，則成員分量的值會被更新到$DFWOBJ 對應的分量中

以上結構體數據詳細信息請參考。

用法舉例

例一：常用方法

示例程序：

```
lin P:P1
lin P:P2
ccir P:P3
ccir P:P4
ccir P:P5
lin P:P6
```

程序執行示意圖如圖 3-1 所示。

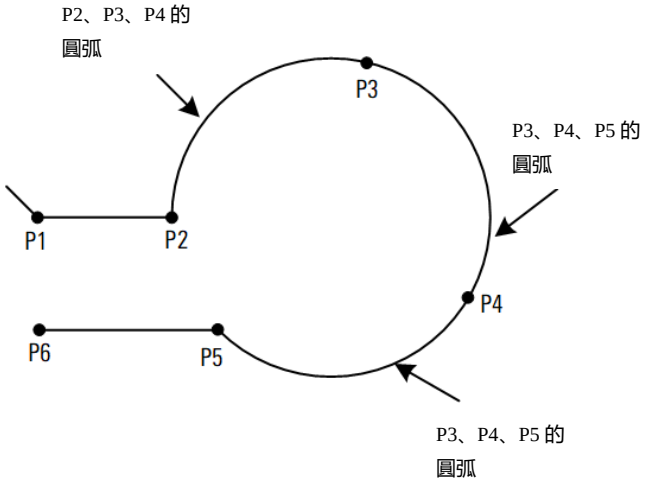


圖 3-1 程序執行示意圖

最初的 lin 指令為直線運動，如圖 3-2（a）所示。在執行第 1 個 ccir 指令時，機器人在通過現在位置、該指令的目標位置、下一個 ccir 的目標位置的 3 點的圓周上運動（如圖 3-2（b）所示）。當程序執行到下一個動作指令不是 ccir 運動指令時，最後的 ccir 運動指令的目標點被視為圓弧的終點。在向著終點運動時，機器人在通過之前一個 ccir 指令的目標位置、現在位置、該指令的目標位置的 3 點的圓周上運動（如圖 3-2（c）所示）。

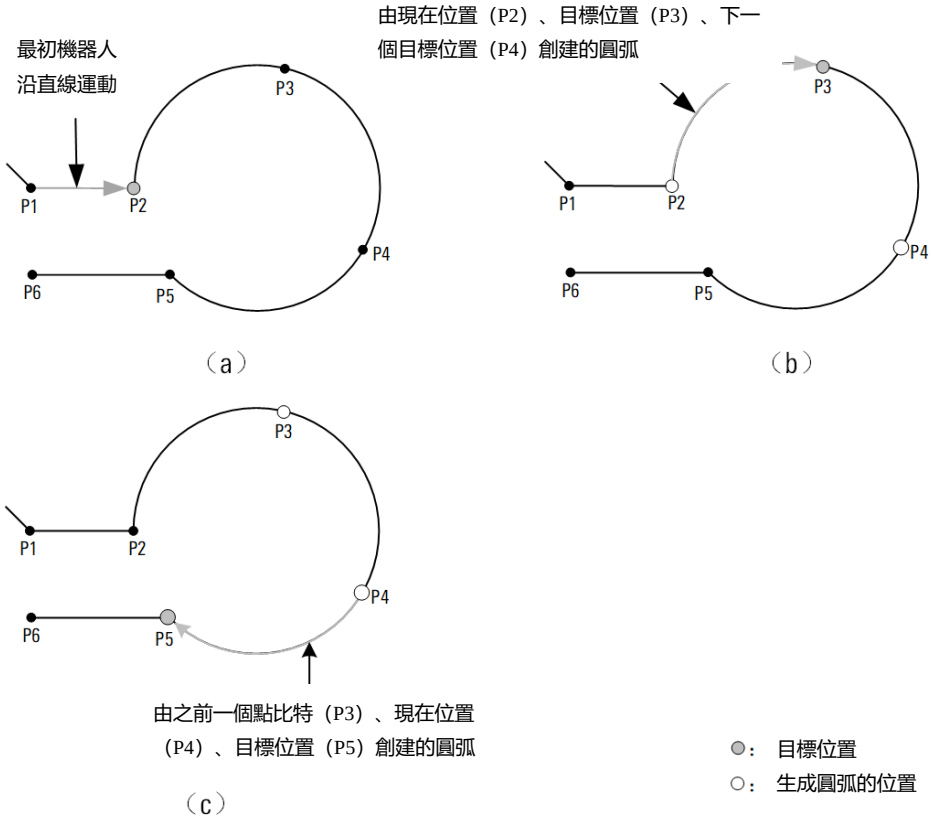


圖 3-2 程序分佈執行示意圖

例二：圓弧朝向的判斷

示例程序：

```
lin P:P1
lin P:P2
ccir P:P3
ccir P:P4
```

執行上述程序的第三行時，機器人通過經由 P2、P3、P4 的 3 點的圓弧，並向著 P3 運動。通過此圓弧向著 P3 的路徑如圖 3-3 所示，機器人沿着 P2→P3→P4 朝向的圓弧一直運動到 P3。

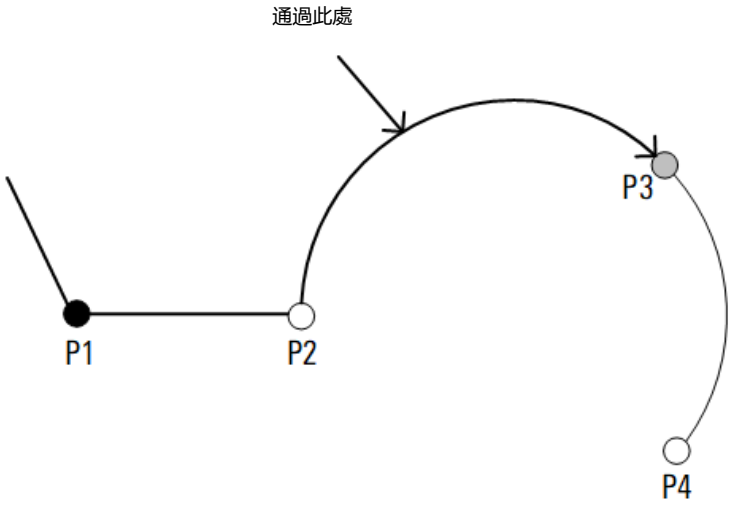


圖 3-3 圓弧朝向判斷程序示意圖

例三：暫停並點動後再繼續移動

機器人在 ccir 指令的中途暫停，在該狀態下再繼續移動時，會沿着之前的運動軌跡再繼續移動。若機器人在 ccir 指令的中途暫停併進行點動，當再繼續移動時，機器人沿直線返回到暫停位置，然後按照之前的路徑繼續運動。

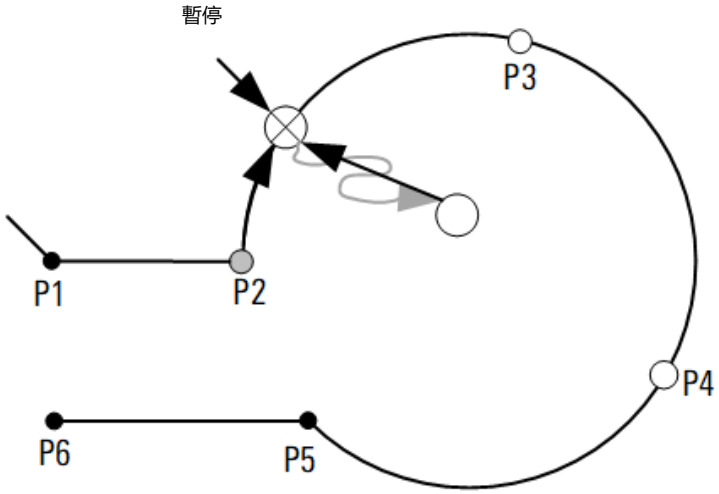


圖 3-4 暫停並點動後再繼續移動示意圖

例四：下一個目標點的修正

將例一中的 P4 進行變更時，機器人沿着通過修正後的 P4 的圓弧再繼續移動。

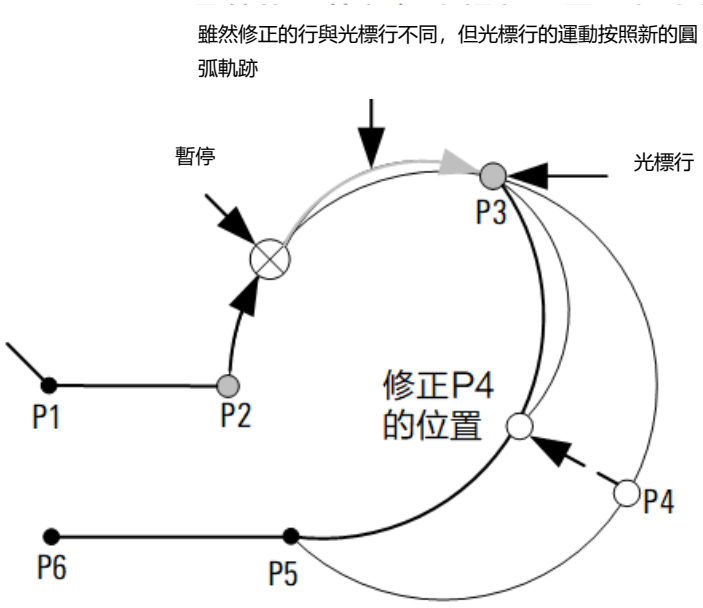


圖 3-5 下一個目標點的修正示意圖

例五：暫停後跳轉

圖 3-6 所示是程式設計為在向著 P2 的中途暫停，從 P3 再繼續移動的示例。機器人從暫停位置，直線跳轉到 P3，再按照 P3→P4→P5 的順序移動。

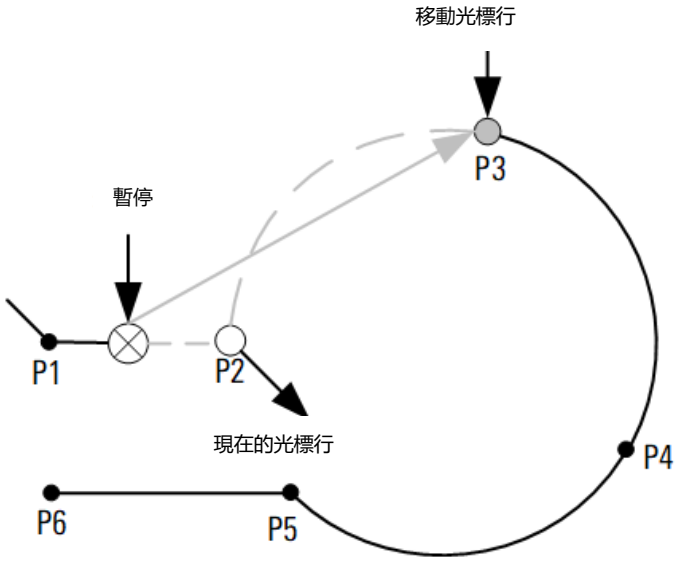


圖 3-6 暫停後跳轉示意圖

3.2.6 疊加軌跡指令

3.2.6.1 startweave(開啟疊加軌跡)

描述

startweave 指令用戶開啟疊加軌跡。

格式

startweave weave:weavedata

參數

startweave 指令的參數詳見表 3-6。

表 3-6 startweavelin 指令的參數

名稱	說明
weavedata	數據類型：weavedata
	用於指定擺動軌跡的參數，包括： ■ weave_type（擺動類型） ■ frequency（擺動頻率） ■ amplitude（振幅） ■ dwell_left（左停留時間） ■ dwell_right（右停留時間） ■ dwell_middle（中間停留時間） ■ swing_angle（夾角） ■ radius（半徑） ■ axis（偏轉軸） ■ rotation_angle（偏轉角度） 詳細描述見 2.4.8 節

提示

- axis（偏轉軸）可以預設。
- rotation_angle （偏轉角度）可以預設。
- 當振幅較大時，可能出現超速告警情況，此時可嘗試降低軌跡期望速度或振幅，避免告警。

用法舉例

横擺：

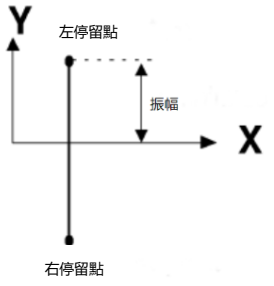
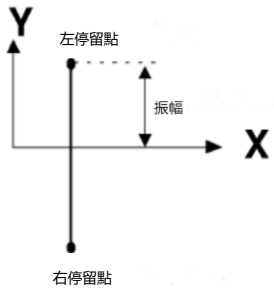
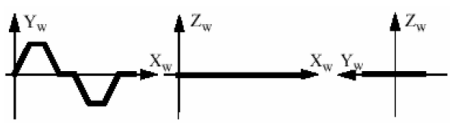
```
weavedata weavedata1
startweave weavedata1={simple,10,15,0,45}
tool t = {{x 0,y 0,z 10,a 0,b 0,c 0}}
wobj w = {{x 1000,y 0,z 500,a 0,b 10,c 0}}
speed v = {per 10,tcp 50,ori 5,exj 10}
pose p2 = {x 10,y 10,z 10,a 0,b 90,c 0,cfg 0}
lin p2,t,w,v
endweave
```

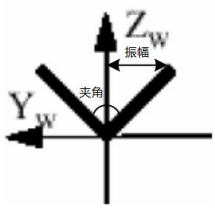
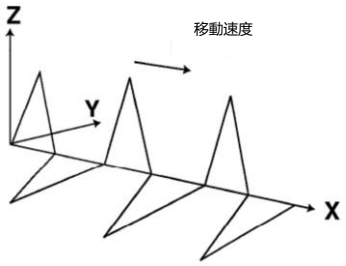
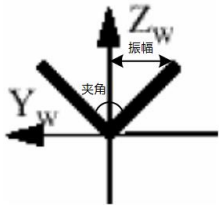
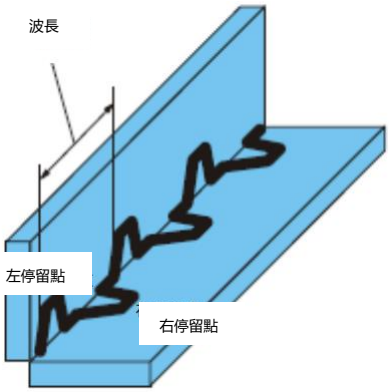
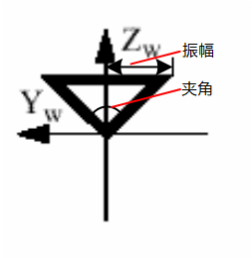
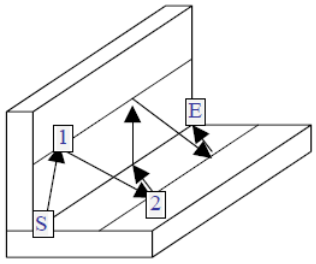
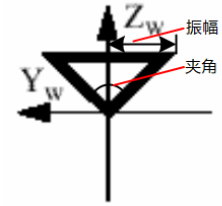
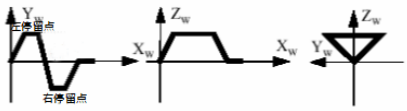
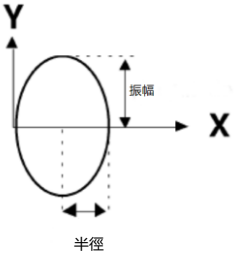
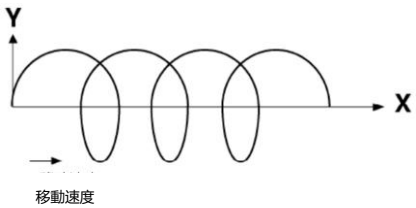
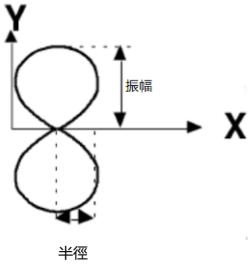
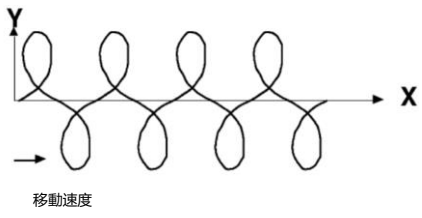
三角擺：

```
weavedata weavedata1 = {V_shape,2,15,90,1,1, 0,1}
startweavespa weavedata1
tool t = {{x 0,y 0,z 10,a 0,b 0,c 0}}
wobj w = {{x 1000,y 0,z 500,a 0,b 10,c 0}}
speed v = {per 10,tcp 50,ori 5,exj 10}
pose p2 = {x 10,y 10,z 10,a 0,b 90,c 0,cfg 0}
lin p2,t,w,v
endweave
```

疊加軌跡需支持對疊加圖樣的圖形參數進行調節，具體參考表 3-7：

表 3-7 疊加軌跡的圖形參數

名稱	疊加圖樣	組合圖樣	圖形參數	備註
横擺			幅值	
線性鋸齒横擺			幅值 左側停留長度/時間 右側停留長度/時間 中間停留長度/時間	若基於頻率循環，則採用停留時間，若基於波長，則採用停留長度

名稱	疊加圖樣	組合圖樣	圖形參數	備註
“V”型擺			幅值 夾角	當夾角為 0 時，擺動方向與 z 軸重合
“V”型鋸齒擺			幅值 夾角 左側停留長度/時間 右側停留長度/時間 中間停留長度/時間	若基於頻率循環，則採用停留時間，若基於波長，則採用停留長度
空間三角擺			幅值 夾角	
空間三角鋸齒擺			幅值 夾角 左側停留長度/時間 右側停留長度/時間 中間停留長度/時間	若基於頻率循環，則採用停留時間，若基於波長，則採用停留長度
螺旋線			幅值 半徑	
“8”字形			幅值 半徑	

3.2.6.2 endweave(結束疊加軌跡)

描述

endweave 指令用戶結束疊加軌跡。

格式

endweave

用法舉例

weavedatalin weavedatalin1 = {2,15}
startweavelin weavedatalin1
lin p:{x 1000,y 500,z 500,a 0,b 90,c 0}
endweave

3.2.7 組合指令

“組合指令”的使用方法請參考本公司的《多機聯動使用說明書》。

3.2.8 傳送帶

“傳送帶”相關指令的使用方法請參考本公司的《傳送帶跟蹤使用說明書》。

3.2.9 軟浮動

“軟浮動”相關指令的使用方法請參考本公司的《軟浮動使用說明書》。

3.2.10 軌跡補償

3.2.10.1 startcompen(開始軌跡補償)

描述

startcompen 指令用於開啟機器人的軌跡補償功能。

格式

startcompen data:data

參數

startcompen 指令的參數詳見表 3-8。

表 3-8 startcompen 指令的參數

名稱	說明
data	數據類型：compendata
	該參數指定機器人軌跡補償過程中 TCP 的最大速度、加速度、加加速度、角速度、角加速度、角加加速度

用法舉例

```
const compendata data3 = {tcp_max_vel 200 ,tcp_max_acc 600 ,tcp_max_jerk 1000 ,ori_max_vel 200 ,ori_max_acc 50 ,ori_max_jerk 1000}
startcompen data:data3
```

3.2.10.2 compen(軌跡補償)

描述

compen 指令用於 startcompen 和 endcompen 之間，用於對機器人的軌跡進行補償。

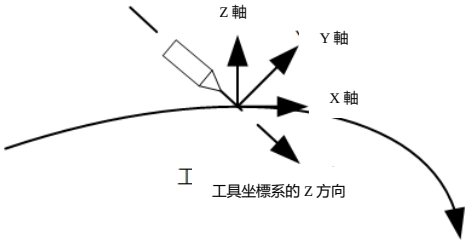
格式

compen [x:],[y:],[z:],[a:],[b:],[c:],type:

參數

compen 指令的參數詳見表 3-9。

表 3-9 compen 指令的參數

名稱	說明
type	數據類型：枚舉型
	可選擇 TOOL，WOBJ、TOOL_PATH 或 WORLD 表示軌跡補償量所參考的坐標系，各個坐標系類型說明如下： ■ TOOL：工具坐標系 ■ WOBJ：工件坐標系 ■ TOOL_PATH：工具-路徑坐標系，定義如下圖所示。  1. 坐標系 x 方向為主軌跡的切線方向； 2. 坐標系 y 方向由坐標系 x 方向和工具坐標系的 z 方向叉乘確定； 3. 坐標系 z 方向由坐標系 x 方向和坐標系 y 方向叉乘確定； 提示 如果主轨迹的切线方向和工具坐标系的 z 方向平行，则无法求出坐标系 y 方向，此时需给出告警。 ■ WORLD：世界坐標系
x	參考坐標系下軌跡上 TCP 沿 x 方向補償的值
y	參考坐標系下軌跡上 TCP 沿 y 方向補償的值
z	參考坐標系下軌跡上 TCP 沿 z 方向補償的值
a	參考坐標系下軌跡上工具坐標系沿 a 方向補償的值
b	參考坐標系下軌跡上工具坐標系沿 b 方向補償的值

名稱	說明
c	參考坐標系下軌跡上工具坐標系沿 c 方向補償的值

用法舉例

圖 37 所示示例為機器人的軌跡相對於工件坐標系 X 方向偏移了 100 毫米。

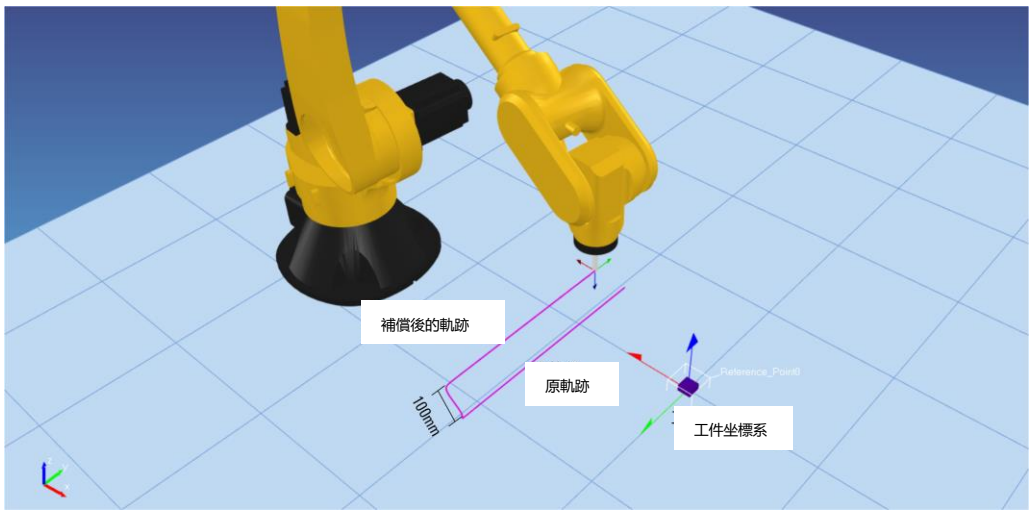


圖 37 軌跡補償示意圖

圖片中對應的程序示例為：

```
const compendata data1 = { tcp_max_vel 250 , tcp_max_acc 900 , tcp_max_jerk 6000 , ori_max_vel 50 , ori_max_acc 75 , ori_max_jerk 375 }

const pose p1 = { x 911.138, y -5.704, z 739.959, a -178.159, b 43.881, c 2.201, cfg 1, turn 000000b, ej1 9.000e+09, ej2 0.000, ej3 9.000e+09, ej4 9.000e+09, ej5 9.000e+09, ej6 9.000e+09 }

startcompen data:data1

compen x:100,y:0,z:0,a:0,b:0,c:0,type:"WOBJ" //相對於工件坐標系 X 方向偏移了 100mm。

lin p:p1,vl:50mm/s,sl:0mm,t:$tool0,w:$WORLD

endcompen
```

> 注意
>
> 單步或段調試模式下，軌跡補償不生效。

3.2.10.3 endcompen(結束軌跡補償)

描述

endcompen 指令用於結束機器人的軌跡補償功能。

格式

```
endcompen
```

用法舉例

參考第 3.2.10.2 章节。


注意

endcompen 後，直接將補償後點位作為起點開始運行下一條不補償的指令。如需先回到原軌跡，可在 endcompen 後增加一條 lin 指令返回原路徑。

3.3 運動指令（適用於 SCARA 機器人）

3.3.1 movej(移動軸)

描述

movej 指令用於將機器人軸或外軸移動到一個指定的軸位置。所有軸同時到達目標軸位置。

格式

movej j:,[v: | vp:],[s: | sl: | sp:],[dura:]

參數

movej 指令的參數詳見表 3-10。

表 3-10 movej 指令的參數

名稱	說明
j	數據類型：joint
	該參數指定機器人各軸和外軸的目標點位置。參數中值為 9e+09 的結構體分量將保持起點位置不變。程序中如果第一條執行的運動指令為 movej，則該 movej 指令的目標點 j1~j4 以及配置的外軸均不允許預設
v	數據類型：speed
	該參數指定運動速度。movej 指令使用 speed 結構體變量中的 per 和 exj 分量，分別用於指定軸運動速度百分比和外軸運動速度。這裡如果預設了 v 參數，則默認使用系統變量$DFSPEED 作為 v 參數的值；如果指定了 v 參數，則成員分量的值會被更新到$DFSPEED 對應的分量中 在簡化程式設計中，v 可由速度百分比參數 vp 代替，格式見用法舉例
vp	數據類型：double
	該參數指定運動速度百分比。可替代速度參數 v，用於不需要精確指定速度大小的場合。格式為 vp:10%，表示當前行速度是機器人最大速度的 10%
s	數據類型：slip
	該參數指定平滑距離。Movej 指令使用 slip 結構體的 perdis 和 ejdis 分量，用於描述脫離原軌跡進入平滑軌跡點。其中，perdis 用於指定距離目標點百分比，ejdis 用於指定外軸距離目標點軸位置角度。這裡如果預設 s 參數，則默認使用系統變量$DFSLIP 作為 s 參數的值；如果指定了 s 參數，則成員分量的值會被更新到$DFSLIP 對應的分量中 在簡化程式設計中，s 可由平滑百分比參數 sp 代替，格式見用法舉例 註： ■ 當 perdis>=0 時，其餘分量將被忽略； ■ 當同時指定 ejdis 時，更靠近目標點的分量將會被選用； ■ 當上述分量均為 0 時，表示不平滑，但插補點不一定與目標點重合； ■ 當上述分量均小於 0 時，表示不平滑，目標點肯定會有插補點，即準停
sp	數據類型：double

名稱	說明
	該參數指定平滑百分比。可替代平滑參數 s，用於不需要精確指定平滑大小的場合。格式為 sp:10%，表示當前行目標點的平滑距離是最大平滑距離的 10%
sl	數據類型：double
	該參數指定平滑距離。可替代平滑參數 s，用於需要精確指定平滑大小的場合。格式為 sl:10mm，表示當前行的平滑距離是 10mm
dura	數據類型：double
	該參數指定軌跡時間。用戶可以直接指定運動時間而不是運動速度。如果用戶指定了 dura 參數，系統將忽略 speed 參數，而是通過自動調整速度來滿足 dura 參數指定的時間要求。dura 參數單位：秒

用法舉例

```
//模糊指定速度和平滑大小
movej j:j1,vp:5%,sp:5%
movej j:{j1 0,j20,j3 0,j4 0, }
movej j:{j1 0}
movej j:{j1 20,ej1 30}
movej j:{ej1 10}
joint p = {0,0,0,0}
speed v = {per 50}
movej p,v
```

3.3.2 ptp(點到點)

描述

ptp 指令用於將機器人從一個點快速運動到另一個點而又不要求 TCP 點所走軌跡形狀時。所有軸同時到達目標點。

格式

```
ptp p:,[ v: | vp: ],[ s: | sl: | sp: ],[ t: ],[ w: ],[ dura: ]
```

參數

ptp 指令的參數詳見表 3-11。

表 3-11 ptp 指令的參數

名稱	說明
p	數據類型：pose
	該參數指定機器人 TCP 目標位姿和外軸的目標點位置。參數中值為 9e+09 的結構體分量將保持起點位置不變。cfg 分量預設默認值為 0，turn 分量預設默認值為-1。程序中如果第一條執行的運動指令為 ptp，則該 ptp 指令的目標點 X,Y,Z,A,B,C 以及配置的外軸均不允許預設
v	數據類型：speed

名稱	說明
	該參數指定運動速度。ptp 指令使用 speed 結構體變量中的 per 和 exj 分量，分別用於指定軸運動速度百分比和外軸運動速度。這裡如果預設了 v 參數，則默認使用系統變量$DFSPEED 作為 v 參數的值；如果指定了 v 參數，則成員分量的值會被更新到$DFSPEED 對應的分量中 在簡化程式設計中，v 可由速度百分比參數 vp 代替，格式見用法舉例
vp	數據類型：double
	該參數指定運動速度百分比。可替代速度參數 v，用於不需要精確指定速度大小的場合。格式為 vp:10%，表示當前行速度是機器人最大速度的 10%
s	數據類型：slip
	該參數指定平滑距離。PTP 指令使用 slip 結構體的 perdis 和 ejdis 分量，用於描述脫離原軌跡進入平滑軌跡點。其中，perdis 用於指定距離目標點百分比，ejdis 用於指定外軸距離目標點軸位置角度。這裡如果預設 s 參數，則默認使用系統變量$DFSLIP 作為 s 參數的值；如果指定了 s 參數，則成員分量的值會被更新到$DFSLIP 對應的分量中 在簡化程式設計中，s 可由平滑百分比參數 sp 代替，格式見用法舉例 註： ■ 當 perdis>=0 時，其餘分量將被忽略； ■ 當同時指定 ejdis 時，更靠近目標點的分量將會被選用； ■ 當上述分量均為 0 時，表示不平滑，但插補點不一定與目標點重合； ■ 當上述分量均小於 0 時，表示不平滑，目標點肯定會有插補點，即準停
sp	數據類型：double
	該參數指定平滑百分比。可替代平滑參數 s，用於不需要精確指定平滑大小的場合。格式為 sp:10%，表示當前行目標點的平滑距離是最大平滑距離的 10%
sl	數據類型：double
	該參數指定平滑距離。可替代平滑參數 s，用於需要精確指定平滑大小的場合。格式為 sl:10mm，表示當前行的平滑距離是 10mm
t	數據類型：tool 結構體
	該參數指定工具。用戶需要在法蘭末端安裝的工具上自定義工具坐標系，該工具坐標系與工具固連，目標點 p 指定的就是工具坐標系在 w 參數指定的坐標系中的位姿。這裡如果預設了 t 參數，則默認使用系統變量$DFTOOL 作為 t 參數的值；如果指定了 t 參數，則成員分量的值會被更新到$DFTOOL 對應的分量中 該參數是為了保證手動低速運行程序時限制 TCP 速度不大於 250mm/s
w	數據類型：wobj 結構體
	該參數指定工件坐標系。用戶可以自定義工件坐標系。目標點 p 指定的就是工具坐標系在 w 參數指定的坐標系中的位姿。在某些情況下，用戶可能通過在特定的坐標系中程式設計更方便。另外，當工件移動位置時，只需重新標定工件坐標系即可，不需要修改用戶程序。這裡如果預設了 w 參數，則默認使用系統變量$DFWOBJ 作為 w 參數的值；如果指定了 w 參數，則成員分量的值會被更新到$DFWOBJ 對應的分量中
dura	數據類型：double
	該參數指定軌跡時間。用戶可以直接指定運動時間而不是運動速度。如果用戶指定了 dura 參數，系統將忽略 speed 參數，通過自動調整速度來滿足 dura 參數指定的時間要求。dura 參數單位：秒

以上結構體數據詳細信息請參考。

用法舉例

```
pose p = {x 266,y 88,z -50,a -34,b 0,c 0,cfg 4}
//模糊指定速度和平滑大小
ptp p:p1,vp:5%,sp:5%
ptp p,v:{per 10}
ptp p:{x 266,y 88,z -50,a 0,b 0,c 0}
ptp p,dura:10
tool t = {{x 0,y 0,z 0,a 0,b 0,c 0}}
wobj w = {{x 0,y 0,z 500,a 0,b 0,c 0}}
speed v = {per 10,tcp 50,ori 5,exj 10}
pose p2 = {x 266,y 88,z -50,a 0,b 0,c 0,cfg 0}
ptp p2,t,w
```

3.3.3 lin(直線運動)

描述

lin 指令用於將機器人 TCP 點沿直線路徑運動到目標點位姿；位置移動和姿態轉動同步。

格式

lin p:,[v: | vl: | vp:],[s: | sl: | sp:],[t:],[w:],[dura:]

參數

lin 指令的參數詳見表 3-12。

表 3-12 lin 指令的參數

名稱	說明
p	數據類型：pose
	該參數指定機器人 TCP 目標位姿和外軸的目標點位置。參數中值為 9e+09 的結構體分量將保持起點位置不變。cfg 參數被忽略，與起點 cfg 參數保持一致，turn 分量預設默認值為-1
v	數據類型：speed
	該參數指定運動速度。lin 指令使用 speed 結構體中的 tcp、ori 和 exj 分量，分別用於指定 TCP 點運動速度、TCP 姿態變化速度和外軸運動速度。這裡如果預設了 v 參數，則默認使用系統變量$DFSPEED 作為 v 參數的值；如果指定了 v 參數，則成員分量的值會被更新到$DFSPEED 對應的分量中 在簡化程式設計中，v 可由速度百分比參數 vp 或速度絕對值參數 vl 代替
vp	數據類型：double
	該參數指定運動速度百分比。可替代速度參數 v，用於不需要精確指定速度大小的場合。格式為 vp:10%，表示當前行速度是機器人最大速度的 10%
vl	數據類型：double
	該參數指定運動線速度。可替代速度參數 v，用於需要精確指定速度大小的場合。格式為 vl:500mm/s，表示當前行的 TCP 最大速度的 500mm/s
s	數據類型：slip
	該參數指定平滑距離。LIN 指令使用 slip 結構體的 perdis、pdis、odis 和 ejdis 分量，用於描述脫離

名稱	說明
	原軌跡進入平滑軌跡點。其中，perdis 用於指定距離目標點百分比，pdis 用於指定距離目標點軸路徑距離，odis 用於指定距離目標點姿態角度，ejdis 用於指定外軸距離目標點軸位置角度。這裡如果預設 s 參數，則默認使用系統變量$DFSLIP 作為 s 參數的值；如果指定了 s 參數，則成員分量的值會被更新到$DFSLIP 對應的分量中 在簡化程式設計中，s 可由平滑百分比參數 sp 或平滑絕對值參數 sl 代替，格式見用法舉例 註： ■ 當 perdis>=0 時，其餘分量將被忽略； ■ 當同時指定 pdis、odis 和 ejdis 時，更靠近目標點的分量將會被選用； ■ 當上述分量均為 0 時，表示不平滑，但插補點不一定與目標點重合； ■ 當上述分量均小於 0 時，表示不平滑，目標點肯定會有插補點，即準停； 使用笛卡爾空間軌跡（即 LIN、CIR）平滑時，建議使用 pdis 平滑，可以使平滑軌跡在前後軌跡上均勻
sp	數據類型：double
	該參數指定平滑百分比。可替代平滑參數 s，用於不需要精確指定平滑大小的場合。格式為 sp:10%，表示當前行目標點的平滑距離是最大平滑距離的 10%
sl	數據類型：double
	該參數指定平滑距離。可替代平滑參數 s，用於需要精確指定平滑大小的場合。格式為 sl:10mm，表示當前行的平滑距離是 10mm
t	數據類型：tool 結構體
	該參數指定工具。用戶需要在法蘭末端安裝的工具上自定義工具坐標系，該工具坐標系與工具固連，目標點 p 指定的就是工具坐標系在 w 參數指定的坐標系中的位姿。這裡如果預設了 t 參數，則默認使用系統變量$DFTOOL 作為 t 參數的值；如果指定了 t 參數，則成員分量的值會被更新到$DFTOOL 對應的分量中 該參數是為了保證手動低速運行程序時限制 TCP 速度不大於 250mm/s
w	數據類型：wobj 結構體
	該參數指定工件坐標系。用戶可以自定義工件坐標系。目標點 p 指定的就是工具坐標系在 w 參數指定的坐標系中的位姿。在某些情況下，用戶可能通過在特定的坐標系中程式設計更方便。另外，當工件移動位置時，只需重新標定工件坐標系即可，不需要修改用戶程序。這裡如果預設了 w 參數，則默認使用系統變量$DFWOBJ 作為 w 參數的值；如果指定了 w 參數，則成員分量的值會被更新到$DFWOBJ 對應的分量中
dura	數據類型：double
	該參數指定軌跡時間。用戶可以直接指定運動時間而不是運動速度。如果用戶指定了 dura 參數，系統將忽略 speed 參數，而是通過自動調整速度來滿足 dura 參數指定的時間要求。dura 參數單位：秒

以上結構體數據詳細信息請參考。

用法舉例

```
pose p = {x 266,y 88,z -50,a 0,b 0,c 0,cfg 0}
ptp p,v:{per 10}
//模糊指定速度和平滑大小
lin p:p1,vp:5%,sp:5%
//精確指定速度和平滑距離
lin p:p1, vl:100mm/s,sl:5mm
```

```
lin p:{x 266,y 88,z -50,a 0,b 90,c 0}
lin p,dura:10
//精確指定速度、平滑結構體
tool t = {{x 0,y 0,z 10,a 0,b 0,c 0}}
wobj w = {{x 0,y 0,z 0,a 0,b 0,c 0}}
speed v = {per 10,tcp 50,ori 5,exj 10}
slip s = { pdis 10, ejdis 10, odis 10 }
pose p2 = {x 266,y 88,z -50,a 0,b 0,c 0,cfg 0}
lin p2,t,w,v,s
```

3.3.4 cir(圓弧運動)

描述

cir 指令用於將機器人 TCP 點沿圓弧路徑運動到目標點；平移運動和旋轉運動同步。

格式

cir m:,p:,[v:],[s:],[t:],[w:],[CA:],[dura:]

參數

cir 指令的參數詳見表 3-13。

表 3-13 cir 指令的參數

名稱	說明
m	數據類型：pose
	該參數指定圓弧輔助點。起點、輔助點和目標點，這 3 點唯一確定了一段圓弧。輔助點中只有 x，y，z 分量被使用，他們中值為 9e+09 的分量將保持起點位置值，其他分量被忽略
p	數據類型：pose
	該參數指定機器人 TCP 目標位姿和外軸的目標點位置。參數中值為 9e+09 的結構體分量將保持起點位置不變。cfg 參數被忽略，與起點 cfg 參數保持一致，turn 分量預設默認值為-1
v	數據類型：speed
	該參數指定運動速度。cir 指令使用 speed 結構體中的 tcp、ori 和 exj 分量，分別用於指定 TCP 點運動速度、TCP 姿態變化速度和外軸運動速度。這裡如果預設了 v 參數，則默認使用系統變量 $DFSPEED 作為 v 參數的值；如果指定了 v 參數，則成員分量的值會被更新到$DFSPEED 對應的分量中 在簡化程式設計中，v 可由速度百分比參數 vp 或速度絕對值參數 vl 代替，格式見用法舉例
vp	數據類型：double
	該參數指定運動速度百分比。可替代速度參數 v，用於不需要精確指定速度大小的場合。格式為 vp:10%，表示當前行速度是機器人最大速度的 10%
vl	數據類型：double
	該參數指定運動線速度。可替代速度參數 v，用於需要精確指定速度大小的場合。格式為 vl:500mm/s，表示當前行的 TCP 最大速度的 500mm/s
s	數據類型：slip
	該參數指定平滑距離。cir 指令使用 slip 結構體的 perdis、pdis、odis 和 ejdis 分量，用於描述脫離

名稱	說明
	原軌跡進入平滑軌跡點。其中，perdis 用於指定距離目標點百分比，pdis 用於指定距離目標點軸路徑距離，odis 用於指定距離目標點姿態角度，ejdis 用於指定外軸距離目標點軸位置角度。這裡如果預設 s 參數，則默認使用系統變量$DFSLIP 作為 s 參數的值；如果指定了 s 參數，則成員分量的值會被更新到$DFSLIP 對應的分量中 在簡化程式設計中，s 可由平滑百分比參數 sp 或平滑絕對值參數 sl 代替，格式見用法舉例 註： ■ 當 perdis>=0 時，其餘分量將被忽略； ■ 當同時指定 pdis、odis 和 ejdis 時，更靠近目標點的分量將會被選用； ■ 當上述分量均為 0 時，表示不平滑，但插補點不一定與目標點重合； ■ 當上述分量均小於 0 時，表示不平滑，目標點肯定會有插補點，即準停； 使用笛卡爾空間軌跡（即 LIN、CIR）平滑時，建議使用 pdis 平滑，可以使平滑軌跡在前後軌跡上均勻
sp	數據類型：double
	該參數指定平滑百分比。可替代平滑參數 s，用於不需要精確指定平滑大小的場合。格式為 sp:10%，表示當前行目標點的平滑距離是最大平滑距離的 10%。
sl	數據類型：double
	該參數指定平滑距離。可替代平滑參數 s，用於需要精確指定平滑大小的場合。格式為 sl:10mm，表示當前行的平滑距離是 10mm。
w	數據類型：wobj 結構體
	該參數指定工件坐標系。用戶可以自定義工件坐標系。目標點 p 指定的就是工具坐標系在 w 參數指定的坐標系中的位姿。在某些情況下，用戶可能通過在特定的坐標系中程式設計更方便。另外，當工件移動位置時，只需重新標定工件坐標系即可，不需要修改用戶程序。這裡如果預設了 w 參數，則默認使用系統變量$DFWOBJ 作為 w 參數的值；如果指定了 w 參數，則成員分量的值會被更新到$DFWOBJ 對應的分量中
CA	數據類型：double
	該參數指定圓心角（Circle Angle）。用戶可以不直接指定目標點，而通過指定圓弧轉過的圓心角的方式來指定目標點。如果用戶指定了 CA 參數，則 p 參數只用來和輔助點一起確定圓弧的幾何形狀，而不是真正的目標點，真正的目標點系統通過用戶指定的圓心角自動計算。CA 參數單位：度
dura	數據類型：double
	該參數指定軌跡時間。用戶可以直接指定運動時間而不是運動速度。如果用戶指定了 dura 參數，系統將忽略 speed 參數，通過自動調整速度來滿足 dura 參數指定的時間要求。dura 參數單位：秒

以上結構體數據詳細信息請參考。

用法舉例

```
pose p3={x 266,y 88,z -50,a 0,b 0,c 0}
pose p4={x 308,y 52,z -50,a 0,b 0,c 0}
tool tool1 = {{x 0,y 0,z 0,a 0,b 0,c 0}}
cir p3,p4,tool1
//指定圓弧圓心角為 360 度
cir p3,p4,CA:360
//模糊指定速度和平滑大小
cir m:p3,p:p4,vp:5%,sp:5%
```

```
//精確指定速度和平滑距離
cir m:p3,p:p4,vl:500mm/s,sl:5mm
//精確指定速度和平滑結構體
tool t1 = {{x 0,y 0,z 0,a 0,b 0,c 0}}
wobj w1 = {{x 0,y 0,z 0,a 0,b 0,c 0}}
speed v2 = {per 10,tcp 50,ori 5,exj 10}
slip s = { pdis 10, ejdis 10, odis 10 }
cir p3,p4,t1,w1,v2, s
```


注意

- 在單步或段調試模式下，cir 指令分為兩步執行，第一步運行至輔助點，第二步運行至目標點。
- 雖然用戶可以通過使用參數預設來簡化程序，但仍建議用戶在編寫運動指令時儘量將參數寫全，以防止由於程式設計人員的疏忽而導致的實際的運動與用戶預期不符，進而導致一些不必要的損失或傷害。

3.3.5 jump(門形運動)

描述

jump 指令用於將機器人從一個點快速抬起並運動到另一個點而又不要求 TCP 點所走軌跡形狀時。所有軸同時到達目標點。

格式

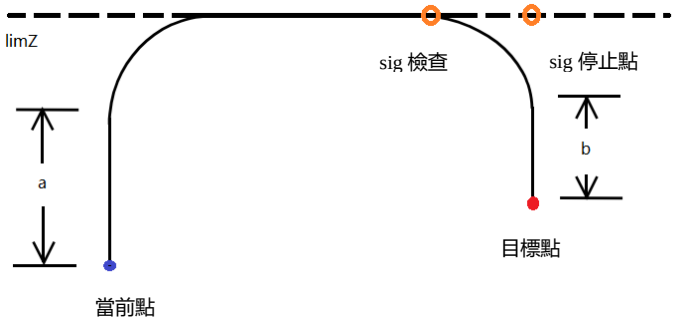
jump p:,[v: | vp:],[t:],[w:],[a:],[b:],[Z:],[ctr:],[s:],[sig:]

參數

jump 指令的參數詳見表 3-14。

表 3-14 jump 指令的參數

名稱	說明
p	數據類型：pose
	該參數指定機器人 TCP 目標位姿和外軸的目標點位置。參數中值為 9e+09 的結構體分量將保持起點位置不變。cfg 分量預設默認值為 0，表示右手系，turn 分量預設默認值為-1，表示運行至最近的 turn 值。程序中如果第一條執行的運動指令為 jump，則該 jump 指令的目標點 X,Y,Z,A,B,C 以及配置的外軸均不允許預設
v	數據類型：speed
	該參數指定運動速度。ptp 指令使用 speed 結構體變量中的 per 和 exj 分量，分別用於指定軸運動速度百分比和外軸運動速度。這裡如果預設了 v 參數，則默認使用系統變量$DFSPEED 作為 v 參數的值；如果指定了 v 參數，則成員分量的值會被更新到$DFSPEED 對應的分量中 在簡化程式設計中，v 可由速度百分比參數 vp 代替，格式見用法舉例
vp	數據類型：double
	該參數指定運動速度百分比。可替代速度參數 v，用於不需要精確指定速度大小的場合。格式為 vp:10%，表示當前行速度是機器人最大速度的 10%
t	數據類型：tool 結構體
	該參數指定工具。用戶需要在法蘭末端安裝的工具上自定義工具坐標系，該工具坐標系與工具固連，目標點 p 指定的就是工具坐標系在 w 參數指定的坐標系中的位姿。這裡如果預設了 t 參數，則默認使用系統變量$DFTOOL 作為 t 參數的值；如果指定了 t 參數，則成員分量的值會被更新到

名稱	說明
	$DFTOOL 對應的分量中 該參數是為了確定機器人 TCP 位置，同時保證手動低速運行程序時限制 TCP 速度不大於 250mm/s
w	數據類型：wobj 結構體
	該參數指定工件坐標系。用戶可以自定義工件坐標系。目標點 p 指定的就是工具坐標系在 w 參數指定的坐標系中的位姿。在某些情況下，用戶可能通過在特定的坐標系中程式設計更方便。另外，當工件移動位置時，只需重新標定工件坐標系即可，不需要修改用戶程序。這裡如果預設了 w 參數，則默認使用系統變量$DFWOBJ 作為 w 參數的值；如果指定了 w 參數，則成員分量的值會被更新到$DFWOBJ 對應的分量中
a	數據類型：double
	拱形指令從起始點垂直上升的相對距離，單位 mm，預設值為 25mm
b	數據類型：double
	拱形指令到達目標點前垂直下降的距離，單位 mm，預設值為 25mm
Z	數據類型：double
	拱形指令 Z 方向允許達到的最大絕對高度，單位 mm，預設值為 0（最大限高）
s	數據類型：bool
	是否平滑（不到達目標點，而是在降速之前拐入下一條語句），true 表示平滑，預設值為 false
ctr	數據類型：control
	設置拱形指令垂直上升段和下降段的速度、加速度、減速度。當需要單獨指定上升或下降段的速度、穩定性時需要指定，詳見第二章 control 結構體
sig	數據類型：bool
	在 jump 指令下降前（圖中 sig 檢查點），系統會檢查 sig 參數，True 則停在目標點正上方（圖中 sig 停止點），三軸不下降，false 表示到達目標點。預設為 false  該處必須為一個 bool 型或者能隱式轉換為 bool 型的表達式。該表達式與 if，while 語句中的判斷語句相同 舉例來說，以下表達式都屬於這種類型的表達式： ■ 1 ■ True ■ getdi(5) ■ getdi(5) == true ■ getdi(5) && getdi(6) ■ counter>=20 ■ T(3.4)

以上結構體數據詳細信息請參考。

用法舉例

pose p3={x 266,y 88,z -50,a 0,b 0,c 0}

jump p:p1,vp:5%,t:$FLANGE,w:$WORLD,ctr:ctr1,s:false

注意

- jump 指令只適用於 scara 機器人，對六軸機器人無效。
- jump 指令為追求節拍最快，低速動作時的軌跡會低於高速動作時的軌跡，因此，即使確認高速沒有發生碰撞，低速動作也可能發生碰撞。
- 如果起始點、目標點超過了 Z 限高，則會告警“12042：Jump 軌跡始末點超笛卡爾 Z 軸限制”。
- 如果垂直上升距離 a 或垂直下降距離 b 設置超過了 Z 限高，則 Z 限高被滿足，a 或 b 不會被滿足，且軌跡退化為直角門型軌跡。

3.3.6 傳送帶

“傳送帶”相關指令的使用方法請參考本公司的《傳送帶跟蹤使用說明書》。

3.4 過程控制

3.4.1 waittime(延時等待)

描述

程序等待指定的時長。執行 waittime 指令時，將自動暫停運動指令的提前規劃。

格式

waittime time:

參數

waittime 指令的參數詳見表 3-15。

表 3-15 waittime 指令的參數

名稱	說明
time	數據類型：double
	該參數指定等待的時間，參數取值應大於 0，單位為秒

用法舉例

waittime time:5 //表示等待 5 秒的時間

3.4.2 waituntil(條件等待)

描述

程序等待，直到某個條件表達式的值為真或者超時，則繼續向下執行。

格式

waituntil cond:,[maxtime:],[timeoutflag:]

參數

waituntil 指令的參數詳見表 3-16。

表 3-16 waituntil 指令的參數

名稱	說明
cond	數據類型：bool
	該參數為一個 bool 的條件表達式，指定程序等待的條件
maxtime	數據類型：double
	最大等待時間，單位秒。當等待的時間超過該參數設置的時間時，即使 cond 條件表達式的值沒有變為 true，程序也將繼續向下執行 該參數預設時默認值為無窮大，即如果程序中不寫這個參數，程序會一直等待，直到 cond 條件表達式的值變為 true
timeoutflag	數據類型：bool
	當 waituntil 指令中指定了 maxtime 參數時，則可同時指定 timeoutflag 參數。這個參數的值部分必須是一個 bool 型變量。waituntil 指令執行完後可以根據這個變量的值判斷 waituntil 指令是否超時。如果該變量值為 true，表示超時 cond 條件表達式的值仍然沒有變為 true；如果該變量值為 false，表示超時之前 cond 條件表達式的值就已經變為 true

用法舉例

waituntil getdi(6) //等待第 6 通道 di 信號值為 true

waituntil getdi(6),maxtime:5 //等待第 6 通道 di 信號值為 true 時,才會執行後面的語句。最大等待時間為 5s

bool time_flag

waituntil getdi(6),maxtime:2,timeoutflag:time_flag //等待第 6 通道 di 信號值為 true 時,才會執行後面的語句。最大等待時間為 2s

if(time_flag)

print "timeout." //等待時間超過 2s

else

print "cond satisfy within 2 sec." //等待時間小於 2s

endif

3.4.3 pause(暫停)

描述

暫停程序運行。程序會在 pause 語句的前一句執行完畢後進入暫停狀態，必須通過示教器的開始或外部控制按鍵繼續程序運行。

格式

pause

用法舉例

func void main()

movej j:j1,vp:5%,sp:-1%

pause //程序運行到這裡將暫停，通過示教器開始按鍵繼續程序//執行

```
movej j:j2,vp:5%,sp:-1%
endfunc
```

當 pause 指令後有運動指令，且程序運行至 pause 指令時，程序不會進入暫停狀態，當已前瞻的軌跡全部運行後，程序才會進入暫停狀態。故當機器人需要立即停止時，請不要使用 pause 指令。

3.4.4 exit(退出程序)

描述

程序退出執行。

exit 與 return 區別為：return 返回到上一級函數繼續執行；而 exit 則不管當前程序在執行哪個函數，整個程序都將直接退出，再次運行程序時從主函數第一行開始運行。

格式

```
exit
```

用法舉例

```
func void main()
movej j:j1,vp:5%,sp:-1%
//如果第 6 通道 di 信號為 true，則退出程序執行
if(getdi(6))
exit
endif
movej j:j2,vp:5%,sp:-1%
endfunc
```

3.4.5 restart(重啟程序)

描述

程式再次執行。當程式執行到該指令時，程式指針將返回到主函數的第一行以重新開始執行。

格式

```
restart
```

用法舉例

```
func void main()
movej j:{j1 10}
movej j:{j1 20}
restart
endfunc
```

3.4.6 stopmove(停止當前運動)

描述

停止前瞻運動語句。該指令一般用在中斷處理函數中，用於停止被中斷函數中正在執行的運動。與 startmove 配合使用，與需根據 startmove 後面參數來決定啟動後軌跡要跳的條數。

格式

stopmove [type:]

stopmove 指令的詳細用法參見第 6.6 章節。

參數

stopmove 指令的參數詳見表 3-17。

表 3-17 stopmove 指令的參數

名稱	說明
type	數據類型：stoptype
	該參數指定運動停止的減速類型，包括普通停止和快速停止，參見 stoptype 類型定義。該參數可預設，預設值為 general 普通停止。 ■ 普通停止是按照當前最大加速度和加加速度停止。 ■ 快速停止與普通停止相比停止的更快，採用 5 倍加速度和加加速度。

3.4.7 startmove(重新啟動停止的運動)

描述

重新啟動停止的運動。該指令一般用在中斷處理函數中，與 stopmove 配合使用，繼續執行被 stopmove 指令停止的運動，需根據 startmove 後面參數來決定啟動後軌跡要跳的條數。

格式

startmove [skip:]

startmove 指令的詳細用法參見。

參數

startmove 指令的參數詳見表 3-18。

表 3-18 startmove 指令的參數

名稱	說明
skip	數據類型：int
	skip 後的數字表示從停止的行數起，重新啟動後軌跡要跳的條數。其中：0 表示不跳，即先回到停止點（CP 軌跡以直線回，PTP 軌跡以 PTP 軌跡回），再繼續原軌跡。1 表示跳一條，則會直接回到被中斷的軌跡目標點（CP 軌跡以直線回，PTP 軌跡以 PTP 軌跡回）。2 表示跳 2 條，以此類推。該參數預設時默認值為 0  注意 當 skip 數值大於當前運動指令條數的時候，會運動到起始點後再運行後續的程式，即重新開始運行程式。

3.5 輔助指令

3.5.1 print(列印輸出)

描述

print 指令用於列印輸出到某個位置。可以使用該函數列印一個或多個表達式的值到 HMI 消息欄、優盤、某個指定的文件或者一個字元串，該指令多用於程序調試，當然也可用於用戶輸出日誌。

格式

print [to:],[to:],[to:],[to:],[tostr:],[filepath:],[precision:],[numbase:],{ argtoprint: }

參數

print 指令的參數詳見表 3-19。

表 3-19 print 指令的參數

名稱	說明
to	數據類型：printto
	指定輸出定向，可以是 off，hmi，file 或 console。 ■ off 表示關閉任何列印輸出； ■ hmi 表示列印輸出到 HMI 的消息欄； ■ file 表示列印輸出到指定文件，文件路徑由 filepath 參數指定； ■ console 表示列印輸出到終端，該選項用戶一般用不到。 可以同時存在多個 to 參數（最多 4 個），用於同時輸出到 hmi 和 file，print 指令中只要存在值為 off 的 to 參數，off 前指定的參數即無效。to 參數為模態參數，只要在一個 print 指令中指定了 to 參數，之後將一直保持該輸出定向設置直到下一次指定 to 參數。程序被加載後默認的配置為只輸出到 hmi
tostr	數據類型：string
	輸出到某個 string 變量。可以使用該參數將變量對應的字元串輸出到某個 string 類型的變量中
filepath	數據類型：string
	當某個 to 參數的值為 file 時，該參數用於指定輸出到文件的路徑。該路徑以 script 文件夾作為根文件夾。如指定文件路徑為“mylog/mylog.log”，則實際輸出到“script/mylog/mylog.log”中。此外，用戶需確保目錄“script/mylog”存在，否則運行時會報錯
precision	數據類型：int
	用於指定 double 類型變量輸出的有效數字位數 d
numbase	數據類型：num_base
	用於指定 int 類型變量輸出的數制格式。可以為 hex(16 進制)，dec(10 進制)
argtoprint	數據類型：anytype
	要列印的表達式，argtoprint 參數可以有 1 個或多個，最終系統將這些表達式的值轉為字元串並連接起來列印輸出到指定位置。有一個比較特殊的常量“endl”表示換行符，列印 endl 將會使列印的字元串換到新的一行

用法舉例

```
print to:hmi, "hello world!"
int i = 1
while(i <= 3)
print to:file,filepath:"mylog","loop ",i,endl
i++
endwhile
print precision:3,"PI:",PI
print "255 =",hex,255,"h"
```

程序運行將輸出：

HMI 消息欄：

hello world!

mylog 文件中：

loop 1
loop 2
loop 3
PI:3.14
255 = ffh

3.5.2 scan(掃描輸入)

描述

scan 指令用於掃描一個字元串，將其中使用某個分隔符分隔的一系列子串按類型讀入到一系列的變量中。

格式

```
scan from:,delimiter:,{ argtosave: }
```

參數

scan 指令的參數詳見表 3-20。

表 3-20 scan 指令的參數

名稱	說明
from	數據類型：string
	待掃描的字元串
delimiter	數據類型：string
	分隔符
argtosave	數據類型：int/double/bool/string
	要保存到的變量，argtosave 參數可以有 1 個或多個，這些變量必須是之前定義過的，系統會將 from 參數中一系列字元串按照每個變量的類型解析為對應的數值並存儲到每個變量中

用法舉例

```
double x,y,z
bool b
scan from:"1.1,1.2,1.3,true",delimiter:",",x,y,z,b
print x, " ",y, " ",z, " ",b
```

程序運行將輸出：

1.1 1.2 1.3 true

3.5.3 import(導入 ARL 模塊)

描述

導入一個 arl 模塊。

格式

```
import modpath:
```

參數

import 指令的參數詳見表 3-21。

表 3-21 import 指令的參數

名稱	說明
modpath	數據類型：string
	該參數指定導入的 arl 文件路徑，如果是和當前文件在同一個目錄下，可以直接輸入文件名。

import 指令的具體用法參見。

3.5.4 velset(速度調節)

描述

velset 指令可用於降低或提升之後所有運動指令的程式設計規劃速度倍率，也可用於設置運動段最大速度。

格式

```
velset override:,max:
```

參數

velset 指令的參數詳見表 3-22。

表 3-22 velset 指令的參數

名稱	說明
override	數據類型：int
	程式設計規劃速度倍率的百分比值，包括軸速度、TCP 速度、ORI 速度，100%表示使用程序設定速度，參數取值範圍為 0~100

名稱	說明
	程式設計規劃速度倍率用於運動規劃，程序運行中操作示教器進行速度倍率調節在此基礎上生效，即機器人運行的實際速度倍率是規劃倍率和示教器速度倍率的乘積
max	數據類型：int
	程式設計規劃最大 TCP 速度，單位 mm/s 指令中的最大速度限制的是程式設計規劃速度，不是對實際運行速度的限制，即當程式設計設定速度大於指令中設置的最大速度時，系統以指令中的 max 值進行速度規劃，但實際運行速度還受示教器速度倍率影響，由於速度倍率一定小於 100%，因此實際運行速度一定不會超過 max。

用法舉例

velset override:50, max:800

所有程式設計規劃速度降低到程序設定速度的 50%，TCP 速度在任何情況下不允許超過 800mm/s。

3.5.5 accset(加速度調節)

描述

accset 指令調節機器人運動的加速度和加加速度，常用於機器人加持易碎負載時，可允許較低的加速度和減速度，結果使機器人運動更加柔順，也可適當提速。

格式

accset acc:,ramp:

參數

accset 指令的參數詳見表 3-23。

表 3-23 accset 指令的參數

名稱	說明
acc	數據類型：double
	機器人實際加速度和減速度相對於最大值的百分比形式，100%表示使用系統最大加速度。最大值：300%，指令輸入小於 20%時，取 20%做為實際值
ramp	數據類型：double
	機器人實際加加速度相對於最大值的百分比形式，100%表示使用系統最大加加速度。最大值：300%，指令輸入小於 10%時，取 10%做為實際值

用法舉例

accset acc:50, ramp:300

加速度限制到最大值的 50%.

accset acc:300, ramp:50

加加速度限制到最大值的 50%.

3.5.6 toolload(工具負載設置)

描述

toolload 指令用於指定機器人的工具負載。

格式

toolload toolinertia:

參數

toolload 指令的參數詳見表 3-24。

表 3-24 toolload 指令的參數

名稱	說明
toolinertia	數據類型：ToolInertiaPara
	該參數指定機器人工具負載的質量、質心、慣性主軸方向及轉動慣量。可以自定義 ToolInertiaPara 類型的數據作為 toolload 的參數。也可以將系統變量$TOOL_INERTIA[i]直接作為參數，切換到第 i 號工具負載

用法舉例

CentroidPos cen_pos = {x 15, y 25, z 100} //定義質心位置
InertiaTensor inertia_tensor = {Ixx 30, Ixy 40, Ixz 50, Iyy 60, Iyz 70, Izz 80} //定義慣性張量

ToolInertiaPara tip //定義工具負載
tip.m = 30 //質量
tip.centroid_pos = cen_pos
tip.centroid_dir = cen_dir
tip.moment_inertia = inertia
toolload tip //切換工具負載

//定義工具負載並初始化
ToolInertiaPara tip1 = {20, {30, 35, 40}, {45, 50, 55}, {60, 65, 70}}
toolload toolinertia:tip1

toolload $TOOL_INERTIA[3] //
切換到系統中[3]號工具的負載

3.5.7 toolswitch(工具負載切換)

描述

toolswitch 指令用於切換當前機器人的工具負載序號。

格式

toolswitch toolindex:

參數

toolswitch 指令的參數詳見表 3-25。

表 3-25 toolswitch 指令的參數

名稱	說明
toolindex	數據類型：int
	該參數指定機器人工具負載的質量、質心、慣性主軸方向及轉動慣量。將系統變量 $TOOL_INERTIA[i]直接作為參數，切換到第 i 號工具負載

用法舉例

toolswitch toolindex:2

movej j:{j1 10,j2 20,j3 30,j4 40,j5 50,j6 60}

3.6 功能包

關於弧焊、碼垛、折彎等功能的相關指令，其具體用法請參考本公司的各個功能包對應的使用說明書。

SCARA 機器人目前不支持上述功能包相關指令。

4 邏輯控制指令

4.1 if (條件語句)

格式

```
if(bool 型表達式)
……
elseif(bool 型表達式)
…….
elseif(bool 型表達式)
……
else
…….
endif
```

描述

條件執行語句。

系統將從上向下依次計算 if 後的 bool 型表達式的值，直到某一個表達式的值為真，則執行這個 if 和下一個 elseif 或 else 之間的指令，執行完後跳到 endif 後繼續執行。

其中 elseif 的個數不限，也可以沒有 elseif 和/或 else 的部分。

用法舉例

```
int count = 1
if(count == 1)
setdo(5,true)
endif
if(count > 2)
setdo(5,true)
elseif(count < 2)
setdo(6,true)
else
setdo(7,true)
endif
if(count > 2)
setdo(5,true)
else
setdo(6,true)
endif
```

4.2 compact if (緊湊條件語句)

格式

```
if(bool 型表達式) ……
```

描述

緊湊條件語句。

當條件語句只有一條 if，並且當條件表達式值為 true 時，執行的指令行只有一行時，此時可以使用緊湊型條件語句將 3 行指令縮減為 1 行。

用法舉例

```
if(getdi(3))    setdo(3,true)
```

4.3 while(while 循環)

格式

```
while (bool 型表達式)
……
endwhile
```

描述

循環執行語句。

當 while 後的 bool 型表達式的值為真時，則執行 while 到 endwhile 之間的指令，執行完後重新計算 bool 型表達式的值，如果為真，則重新執行 while 到 endwhile 之間的指令，如此循環，直到 while 後的 bool 型表達式的值為假，則跳到 endwhile 後繼續執行。

用法舉例

```
int a = 0
while(a<3)
a++
endwhile
print a //while 循環體會被執行 3 次，結果輸出“3”
```

4.4 repeat(repeat 循環)

格式

```
repeat
……
until (bool 型表達式)
```

描述

循環執行語句。

repeat 和 until 循環體內的指令至少會被執行 1 次。當 until 後的 bool 型表達式的值為真時，則退出循環；反之，如果 until 後的 bool 型表達式的值為假時，則繼續 repeat 循環。

用法舉例

```
int a = 0
repeat
```

```
a++
until(a >= 3)
print a //repeat 循環體會被執行 3 次，結果輸出“3”
```

4.5 loop(無限循環)

格式

```
loop
……
endloop
```

描述

無限循環執行語句。

該循環指令相當於 while(1)或 while(true)。循環永遠都不會停止，除非內部執行 break/exit/restart/return 指令。

用法舉例

```
int a = 5
loop
if(a-- == 0) break
endloop
print a //結果輸出“-1”
```

4.6 for(for 循環)

格式

```
for (初始化語句；bool 型表達式；迭代表達式)
……
endfor
```

描述

循環執行語句。

首先執行初始化語句，然後判斷 bool 型表達式是否為真，如果為真，則執行 for 和 endfor 之間的指令，之後執行迭代表達式，然後再次判斷 bool 型表達式是否為真，如果為真，則執行 for 和 endfor 之間的指令。直到 bool 型表達式的值為假時跳到 endfor 之後繼續執行。

用法舉例

```
int b = 0
for(int i = 0;i<5;i++)
b++
endfor
print b //for 循環體會被執行 5 次，結果輸出“5”
```


注意

while 循環和 for 循環並不完全等同於循環運行模式。使用 while 循環或 for 循環時，所有的系統變量以及允許預設的參數並不會被系統重置，而使用循環模式時，當一次循環執行完畢後，程序會執行複位動作。這意味着那些在程序複位時會被重置的系統變量，以及可以預設的參數將被恢復為默認值。

例如：

ptp P0

lin P1,s:{pdis 10}

如果設置了循環運行模式，當第 1 次循環執行完畢後，程序會執行複位的動作。當第 2 次執行 PTP 語句時，slip 參數已經被覆位，所以這條 PTP 運動將不會有平滑行為。如果希望在循環運行的過程中，PTP 語句目標點要平滑，請按如下方式明確指定 PTP 的 slip 參數而不要預設它。

例如：

ptp P0,s:{pdis 10}

lin P1,s:{pdis 10}

4.7 break(跳出循環)

格式

break

描述

跳出循環。在 while，for，loop，repeat 循環體內，如果執行到該語句，程序將退出最近一層循環繼續向下執行。

用法舉例

```
int counter = 0
while(1)
if(counter == 5)
break //跳出 while 循環。
else
counter++
    endif
endwhile
print count //輸出“2”
```

4.8 continue(繼續下一個循環)

格式

continue

描述

結束當前循環，繼續下一次循環。在 while，for，loop，repeat 循環體內，如果執行到該語句，程序將結束當前循環，繼續執行最近一層循環的下一次循環。

用法舉例

```
int count = 0
```

```
while(1)
count++
if(count == 1)
continue //執行到這裡則直接跳回執行 count++
else
break //執行到這裡則退出 while 循環，繼續執行 endwhile 下麵的指令
endif
endwhile
```

4.9 switch(條件分支)

格式

```
switch(表達式)
case 常量表達式：
……
case 常量表達式：
……
……
default：
……
endswitch
```

描述

條件分支指令是 if 語句的另一種寫法。系統將首先計算 switch 後的表達式的值，然後從上到下依次與每一個 case 後面的常量表達式做比較，直到比較相等，則執行這個 case 與下一個 case 之間的指令。執行完後跳到 endswitch 後繼續執行。如果沒有與任何一個 case 匹配，則執行 default 後面的指令。

用法舉例

```
func void TestSwitch()
int j = 3
int i = 0
switch(j)
case 0:
i = 0
case 1:
i = 1
case 2:
i = 2
case 3:
i = 3 //程序會執行到這裡
default:
i = 4
endswitch
endfunc
```

4.10 goto(跳轉)

格式

label:
goto label

描述

可以在一個函數內的任意一行定義一個 label 標簽，標簽名的命令規則同變量。執行到 goto label 一行時，程序將跳轉到 label 標簽所在行的下一行繼續執行。

用法舉例

```
func void TestGoto()
int i = 0
next:
i++
if(i<5)
goto next
else
goto end
endif
end:
print i //輸出“5”
endfunc
```

4.11 return(函數返回)

格式

return

或

return 表達式

描述

函數返回。

程序遇到 return 指令時，如果程序當前處於被調用的函數中，則程序將返回到上一級函數中。如果程序當前處於主函數中，則程序直接結束。

在有返回值的函數中，return 後面需要接一個函數返回值類型的表達式，系統會計算該表達式的值並將結果返回給調用該函數的表達式中。

用法舉例

```
func int add (int x,int y)
return x+y //return 語句
endfunc
```

```
func void main()
print add(2,3) //輸出“5”
endfunc
```

5 系統預定義函數

系統預先定義了很多功能函數，這些函數可以在程序中直接調用。

5.1 數學函數

5.1.1 abs(求絕對值)

函數原型

double abs(double x)

或者 int abs(int x)

描述

計算參數 x 的絕對值。

參數

表 5-1 abs 函數的參數

名稱	說明
x	類型：double or int
	輸入值

返回值

類型：double or int

返回參數 x 的絕對值。

用法舉例

```
double x = -11.11
int y = -10
double resultx = abs(x)
int resulty = abs(y)
print resultx //輸出“11.11”
print resulty //輸出“10”
```

5.1.2 sin(正弦函數)

函數原型

double sin(double x)

描述

計算參數 x 對應的正弦值。

參數

表 5-2 sin 函數的參數

名稱	說明
x	類型：double
	角度值，單位弧度

返回值

類型：double

返回參數 x 對應的正弦值，範圍[-1,1]。

用法舉例

```
double x = PI/6
double y = sin(x)
print y //輸出“0.5”
```

5.1.3 cos(餘弦函數)

函數原型

```
double cos(double x)
```

描述

計算參數 x 對應的餘弦值。

參數

表 5-3 cos 函數的參數

名稱	說明
x	類型：double
	角度值，單位弧度

返回值

類型：double

返回參數 x 對應的餘弦值，範圍[-1,1]。

用法舉例

```
double x = PI/3
double y = cos(x)
print y //輸出“0.5”
```

5.1.4 tan(正切函數)

函數原型

double tan(double x)

描述

計算參數 x 對應的正切值。

參數

表 5-4 tan 函數的參數

名稱	說明
x	類型：double
	角度值，單位弧度

返回值

類型：double

返回參數 x 對應的正切值。

用法舉例

```
double x = PI/4
double y = tan(x)
print y //輸出“1”
```

5.1.5 asin(反正弦函數)

函數原型

double asin(double x)

描述

計算參數 x 對應的反正弦值。

參數

表 5-5 asin 函數的參數

名稱	說明
x	類型：double
	參數範圍 [-1,1]

返回值

類型：double

返回參數 x 對應的反正弦值，單位弧度，範圍[-PI/2，PI/2]。

用法舉例

```
double x = 0.5
double y = asin(x)
print y //輸出“0.523599”
```

5.1.6 acos(反餘弦函數)

函數原型

```
double acos(double x)
```

描述

計算參數 x 對應的反餘弦值。

參數

表 5-6 acos 函數的參數

名稱	說明
x	類型：double
	參數範圍[-1,1]

返回值

類型：double

返回參數 x 對應的反餘弦值，單位弧度，範圍[0，PI]。

用法舉例

```
double x = 0.5
double y = acos(x)
print y //輸出“1.0472”
```

5.1.7 atan(反正切函數)

函數原型

```
double atan(double x)
```

描述

計算參數 x 對應的反正切值。

參數

表 5-7 atan 函數的參數

名稱	說明
x	類型：double
	輸入值

返回值

類型：double

返回參數 x 對應的反正切值，單位弧度，範圍(-PI/2，+PI/2)。

用法舉例

```
double x = 1
double y = atan(x)
print y //輸出“0.785398”
```

5.1.8 atan2(反正切函數)

函數原型

```
double atan2(double y,double x)
```

描述

計算參數 y/x 對應的反正切值。

參數

表 5-8 atan2 函數的參數

名稱	說明
y	類型：double
	分子數值
x	類型：double
	分母數值

返回值

類型：double

返回參數 y/x 對應的反正切值，單位弧度，範圍[-PI,+PI]。

用法舉例

```
double x = 1
double y = 1
double z = atan2(y,x)
print z //輸出“0.785398”
```

5.1.9 sinh(雙曲正弦函數)

函數原型

```
double sinh(double x)
```

描述

計算參數 x 對應的雙曲線正弦值，其中 sinh(x)=(exp(x)-exp(-x))/2。

參數

表 5-9 sinh 函數的參數

名稱	說明
x	類型：double
	輸入值，單位是弧度

返回值

類型：double

返回參數 x 對應的雙曲線正弦值。

用法舉例

```
double x = 1
double y = sinh(x)
print y //輸出“1.1752”
```

5.1.10 cosh(雙曲餘弦函數)

函數原型

```
double cosh(double x)
```

描述

計算參數 x 對應的雙曲餘弦值，其中 cosh(x)=(exp(x)+exp(-x))/2。

參數

表 5-10 cosh 函數的參數

名稱	說明
x	類型：double
	輸入值，單位是弧度

返回值

類型：double

返回參數 x 對應的雙曲餘弦值。

用法舉例

```
double x = 1
double y = cosh(x)
```

print y //輸出“1.54308”

5.1.11 tanh(雙曲正切函數)

函數原型

double tanh(double x)

描述

計算參數 x 對應的雙曲線正切值，其中 tanh(x)=sinh(x)/cosh(x)。

參數

表 5-11 tanh 函數的參數

名稱	說明
x	類型：double
	輸入值，單位是弧度

返回值

類型：double

返回參數 x 對應的雙曲線正切值，範圍(-1,+1)。

用法舉例

double x = 1
double y = tanh(x)
print y //輸出“0.76159”

5.1.12 exp(指數函數)

函數原型

double exp(double x)

描述

計算參數 x 的以 e 為底的 x 次方值。

參數

表 5-12 exp 函數的參數

名稱	**說明**
x	類型：double
	指數值。

返回值

類型：double

返回參數 x 對應的 e 為底的 x 次方值，範圍(0,+∞)。

用法舉例

```
double x = 1
double y = exp(x)
print y //輸出“2.71828”
```

5.1.13 pow(指數函數)

函數原型

```
double pow(double x,double y)
```

描述

計算以 x 為底的 y 次方值。

參數

表 5-13 pow 函數的參數

名稱	說明
x	類型：double
	底數部分
y	類型：double
	指數部分

返回值

類型：double

返回以 x 為底的 y 次方值。

用法舉例

```
double x = 2
double y = 3
double z = pow(x,y)
print z //輸出“8”
```

5.1.14 pow10(指數函數)

函數原型

```
double pow10(double x)
```

描述

計算參數 x 的以 10 為底的 x 次方值。

參數

表 5-14 pow10 函數的參數

名稱	說明
x	類型：double
	指數值

返回值

類型：double

返回參數 x 的以 10 為底的 x 次方值。

用法舉例

```
double x = -1
double y = pow10(x)
print y //輸出“0.1”
```

5.1.15 log(對數函數)

函數原型

```
double log(double x)
```

描述

計算以 e 為底的 x 對數值。

參數

表 5-15 log 函數的參數

名稱	說明
x	類型：double
	參數範圍 x>0

返回值

類型：double

返回以 e 為底的 x 對數值，即 ln(x)的值。

用法舉例

```
double x = 2
double y = log(x)
```

print y //輸出“0.69315”

5.1.16 log10(對數函數)

函數原型

double log10(double x)

描述

計算以 10 為底的 x 對數值。

參數

表 5-16 log10 函數的參數

名稱	說明
x	類型：double
	參數範圍 x>0

返回值

類型：double

返回以 10 為底的 x 對數值，即 log10(x)的值。

用法舉例

double x = 100
double y = log10(x)
print y //輸出“2”

5.1.17 sqrt(開方函數)

函數原型

double sqrt(double x)

描述

計算參數 x 的平方根值。

參數

表 5-17 sqrt 函數的參數

名稱	說明
x	類型：double
	參數範圍 x>=0

返回值

類型：double

返回參數 x 的平方根值，即的值。

用法舉例

```
double x = 2
double y = sqrt(x)
print y //輸出“1.41421”
```

5.1.18 floor(向下取整)

函數原型

```
double floor(double x)
```

描述

計算不大於參數 x 的最大整數值。

參數

表 5-18 floor 函數的參數

名稱	說明
x	類型：double
	輸入值

返回值

類型：double

返回不大於參數 x 的最大整數值。

用法舉例

```
double x = 2.36
double y = floor(x)
print y //輸出“2”
```

5.1.19 ceil(Rounded up)

函數原型

```
double ceil(double x)
```

描述

計算不小於參數 x 的最小整數值。

參數

表 5-19 ceil 函數的參數

名稱	說明
x	類型：double
	輸入值

返回值

類型：double

返回不小於參數 x 的最小整數值。

用法舉例

```
double x = 2.36
double y = ceil(x)
print y //輸出“3”
```

5.1.20 frexp(分解浮點數)

函數原型

```
double frexp(double val,int &exp)
```

描述

將參數 val 分解成數字部分 x 和以 2 為底的指數部分 n，即 val=x*2n，其中 n 存放在 exp 指向的變量中。

參數

表 5-20 frexp 函數的參數

名稱	說明
val	類型：double
	輸入值
exp	類型：int
	輸出指數值

返回值

類型：double

返回參數 val 分解成數字部分 x 的值，範圍[0.5,1)。

用法舉例

```
double val = 64
int exp
```

```
double y = frexp(val,exp)
print y //輸出“0.5”
print exp //輸出“7”
```

5.1.21 ldexp(装載浮點數)

函數原型

```
double ldexp(double val,int exp)
```

描述

計算參數 val 與 2 的 exp 次方值的乘積，即計算 val*2exp。

參數

表 5-21 ldexp 函數的參數

名稱	說明
val	類型：double
	輸入值
exp	類型：int
	輸入指數值

返回值

類型：double

返回參數 val 與 2 的 exp 次方值的乘積。

用法舉例

```
double val = 3
int exp = 3
double y = ldexp(val,exp)
print y //輸出“24”
```

5.1.22 fmod(求模)

函數原型

```
double fmod(double x,double y)
```

描述

計算參數 x 對 y 的模，即 x/y 的餘數。

參數

表 5-22 fmod 函數的參數

名稱	說明
x	類型：double
	分子參數
y	類型：double
	分母參數，y 不能為 0

返回值

類型：double

返回參數 x/y 的餘數。

用法舉例

```
double x = 12.58
double y = 2.6
double z = fmod(x,y)
print z //輸出“2.18”
```

5.1.23 modf(分解浮點數)

函數原型

```
double modf(double x,int &y)
```

描述

把浮點數 x 分解成整數部分和小數部分，並將整數部分存到 y 變量中。

參數

表 5-23 modf 函數的參數

名稱	說明
x	類型：double
	被分解的浮點數
y	類型：int
	返回的浮點數的整數部分

返回值

類型：double

返回雙精度數 x 的小數部分，範圍(0,1)。

用法舉例

```
double x = 123.456
double y
double z = modf(x,y)
print y //輸出“123”
print z //輸出“0.456”
```

5.1.24 hypot(求直角三角形斜邊長)

函數原型

```
double hypot(double x,double y)
```

描述

計算直角三角形的兩個直角邊分別是 x，y 時，其斜邊的長度。

參數

表 5-24 hypot 函數的參數

名稱	說明
x	類型：double
	直角三角形的一條直角邊長度
y	類型：double
	直角三角形的另一條直角邊長度

返回值

類型：double

返回 x 平方與 y 平方的和的平方根，即$\sqrt{x^2+y^2}$。

用法舉例

```
double x = 6
double y = 8
double z = hypot(x,y)
print z //輸出“10”
```

5.1.25 rand(產生隨機數)

函數原型

```
int rand(void)
```

描述

隨機產生一個整數。

參數

無

返回值

類型：int

返回一個隨機整數，範圍[0,+2147483647]。

用法舉例

```
int x = rand()
print x //隨機輸出一個整數，比如輸出“15”
```

5.1.26 norm(求向量長度)

函數原型

```
double norm(pos p)
```

描述

計算當前位置點距離坐標系原點的長度。

參數

表 5-25 norm 函數的參數

名稱	說明
p	類型：pos
	向量坐標值

返回值

類型：double

返回當前位置點 p 距離坐標系原點的長度。

用法舉例

```
pos p = {x 300,y 400,z 500}
double len = norm(p)
print len //輸出“707.107”
```

5.1.27 trunc(截斷浮點數)

函數原型

```
double trunc(double num,int n,bool round)
```

描述

將某一數值按照是否四捨五入的要求截斷成小數點後指定位數的數值。

參數

表 5-26 trunc 函數的參數

名稱	說明
num	類型：double
	被截斷的浮點數
y	類型：double
	保留小數的位數。n>=0
round	類型：bool
	true 表示四捨五入，false 表示直接截斷

返回值

類型：double

返回截斷後的結果。

用法舉例

```
double num = 3.1415926
int n = 4
double result1 = trunc(num,n,true)
print result1 //輸出“3.1416”
double result2 = trunc(num,n,false)
print result2 //輸出“3.1415”
```

5.2 位操作函數

5.2.1 bitclear(位清 0)

函數原型

void bitclear(byte &data,int pos)

或者 void bitclear(int &data,int pos)

描述

將一個整型或位元組型數 data 的第 pos 位清為零。

參數

表 5-27 bitclear 函數的參數

名稱	說明
data	類型：byte 或 int
	輸入值
pos	類型：int

名稱	說明
	參數範圍[0,7]（當 data 類型是 byte 時）或者[0,31]（當 data 類型是 int 時）

返回值

無

用法舉例

byte data1= 00001111b
bitclear(data1,2)
print data1 //輸出“0bh”
int data2= 15
bitclear(data2,2)
print data2 //輸出“11”

5.2.2 bitset(位置 1)

函數原型

void bitset(byte &data,int pos)

或者 void bitset(int &data,int pos)

描述

將一個整型或位元組型數 data 的第 pos 位設置成 1。

參數

表 5-28 bitset 函數的參數

名稱	說明
data	類型：byte 或 int
	待設置位的數。
pos	類型：int
	參數範圍[0,7]（當 data 類型是 byte 時）或者[0,31]（當 data 類型是 int 時）

返回值

無

用法舉例

byte data1= 00001000
bitset(data1,2)
print data1 //輸出“0ch”
int data2 = 8
bitset(data2,2)
print data2 //輸出“12”

5.2.3 bitcheck(位檢查)

函數原型

bool bitcheck(byte &data,int pos)

或者 bool bitcheck(int &data,int pos)

描述

檢查一個整型或位元組型數 data 的第 pos 位是否為 1，如果是則函數返回 true，如果不是則函數返回 false。

參數

表 5-29 bitcheck 函數的參數

名稱	說明
data	類型：byte 或 int
	待檢查位的數。
pos	類型：int
	參數範圍[0,7]（當 data 類型是 byte 時）或者[0,31]（當 data 類型是 int 時）

返回值

類型：bool

返回參數 data 的第 pos 位是否等於 1，等於 1 函數返回 true，不等於 1 函數返回 false

用法舉例

```
byte data1= 00001000b
bool result = bitcheck(data1,2)
print result //輸出“false”
int data2= 8
print bitcheck(data2,3) //輸出“true”
```

5.2.4 bitlcs(循環左移多位)

函數原型

int bitlcs(int data,int n)

或者 byte bitlcs(byte data,int n)

描述

將一個整型或位元組型數 data 循環左移 n 位。

參數

表 5-30 bitlcs 函數的參數

名稱	說明
data	類型：byte 或 int
	待移位的數。
n	類型：int
	n >= 0。參數 n 可以預設，預設默認值為 1

返回值

類型：int 或 byte

返回參數 data 循環左移位後的結果。

用法舉例

```
byte data= 11000000b
print bitlcs(data,2) //輸出“03h”
print bitlcs(data) //輸出“81h”
```

5.2.5 bitrcs(循環右移多位)

函數原型

int bitrcs(int data,int n)

或者 byte bitrcs(byte data,int n)

描述

將一個整型或位元組型數 data 循環右移 n 位。

參數

表 5-31 bitrcs 函數的參數

名稱	說明
data	類型：byte 或 int
	待移位的數。
n	類型：int
	n >= 0。參數 n 可以預設，預設默認值為 1

返回值

類型：int 或 byte

返回參數 data 循環右移位後的結果。

用法舉例

```
byte data= 00000011b
print bitrcs(data,2) //輸出“c0h”
print bitrcs(data) //輸出“81h”
```

5.3 時鐘函數

5.3.1 clock(時鐘類型)

描述

clock 時鐘類型用於程序中計時。該數據類型配合 clkreset，clkstart，clkstop 和 clkread 函數使用。

用法舉例

```
clock c
clkreset(c)
clkstart(c)
waittime 3
clkstop(c)
print clkread(c) //輸出“3.00098”,表示從 clkstart 到 clkstop 共耗時約 3 秒。
```

5.3.2 clkstart(啟動時鐘計時)

函數原型

```
void clkstart(clock &c)
```

描述

啟動時鐘計時。一個時鐘被啟動後，可以對他進行 clkread，clkstop 和 clkreset 動作。

參數

表 5-32 clkstart 函數的參數

名稱	說明
c	類型：clock
	參見 數據類型

返回值

無

用法舉例

參見 clkread 函數。

5.3.3 clkstop(停止時鐘計時)

函數原型

```
void clkstop(clock &c)
```

描述

停止時鐘計時。一個時鐘被停止後，可以對他進行 clkread，clkstart 和 clkreset 動作。

參數

表 5-33 clkstop 函數的參數

名稱	說明
c	類型：clock
	參見 數據類型

返回值

無

用法舉例

參見 clkread 函數。

5.3.4 clkreset(時鐘清 0)

函數原型

void clkreset(clock &c)

描述

將時鐘變量重新設置為 0。

參數

表 5-34 clkreset 函數的參數

名稱	說明
c	類型：clock
	參見 數據類型

返回值

無

用法舉例

參見 clkread 函數。

5.3.5 clkread(讀取時鐘)

函數原型

double clkread(clock &c)

描述

讀取時鐘變量 c 的數值。

參數

表 5-35 clkread 函數的參數

名稱	說明
c	類型：clock
	參見 數據類型

返回值

類型：double

返回時鐘變量 c 的數值，單位是秒

用法舉例

clock c

clkstart(c)

byte data= 00001100b

int n = 2

byte result = bitrcs1(data,n)

clkstop(c)

print clkread(c) //輸出從上一個 clkstart 函數到 clkstop 函數之間運行花費的時間，例如運行時間為 0.001s,則輸出“0.001”

clkreset(c)

print clkread(c) //輸出“0”

5.4 字元串相關函數

5.4.1 strlen(獲取字元串長度)

函數原型

int strlen(string s)

描述

計算字元串 s 的長度。

參數

表 5-36 strlen 函數的參數

名稱	說明
s	類型：string
	字元數量有待統計的字元串。

返回值

類型：int

返回字元串 s 的長度值。

用法舉例

```
string s = "hello world!"
int len = strlen(s)
print len //輸出“12”
```

5.4.2 substr(截取字元串)

函數原型

```
string substr(string s,int startpos,int len)
```

描述

從字元串的指定位置處開始截取特定長度的字元子串。

參數

表 5-37 substr 函數的參數

名稱	說明
s	類型：string
	待截取的字元串。
startpos	類型：int
	截取字元串的起始位置。startpos 從 0 開始，如果 startpos 為負數或者大於等於字元串總長度，將會產生一個運行時錯誤
len	類型：int
	截取字元串的長度。如果沒有指定該參數，則子字元串將延續到字元串的結尾。如果 len 為負數，將會產生一個運行時錯誤

返回值

類型：string

返回從字元串 s 的指定位置 startpos 處開始截取 len 長度的字元子串。

用法舉例

```
s1 = "hello world!"
int statpos = 0
int len = 5
string s2 = substr(s1,startpos,len)
print s2 //輸出“hello”
```

5.4.3 toascii(獲取字元對應的 ASCII 碼)

函數原型

int toascii(string s)

描述

該函數返回字元 s 對應的 ASCII 碼。

參數

表 5-38 toascii 函數的參數

名稱	說明
s	類型：string
	輸入字元串 s，該字元串只能包含 1 個字元

返回值

類型：int

字元 s 對應的 ASCII 碼值。

用法舉例

print toascii("a") //輸出“97”

5.5 IO 相關函數及指令

5.5.1 setdo(異步輸出單路 DO)

函數原型

void setdo(int chan,bool value)

描述

設置一路 DO 信號為 true 或者 false。

參數

表 5-39 setdo 函數的參數

名稱	說明
chan	類型：int
	參數範圍為[1,DO 埠數]。如：[1,32*DO 從站數]（針對 MIII 總線版控制系統）或[1,40*PLC_MF 從站數]（針對 RTEX 總線版控制系統）
value	類型：bool
	指定輸出值

返回值

無

用法舉例

setdo(3,true) //第 3 通道置為 true

5.5.2 setdo(異步輸出多路 DO)

函數原型

void setdo(int from_chan,int to_chan,int value)

描述

設置連續多路 DO 信號。

參數

表 5-40 setdo 函數的參數

名稱	說明
from_chan	類型：int
	參數範圍為[1,DO 埠數]。如：[1,32*DO 從站數]（針對 MIII 總線版控制系統）或 [1,40*PLC_MF 從站數]（針對 RTEX 總線版控制系統）
to_chan	類型：int
	結束通道號。參數範圍[1,32*DO 從站數]（針對 MIII 總線版控制系統）或[1,40*PLC_MF 從站數]（針對 RTEX 總線版控制系統） from_chan <= to_chan 即使 to_chan 超過實際可控地址範圍，只要滿足上述條件，仍舊可以正常輸出可控地址範圍內的 DO，不會產生報警
value	類型：int
	因為 int 型數據長度為 32 位，所以這裡最多只能同時設置連續的 32 路 DO

返回值

無

用法舉例

setdo(5,8,1100b)

//將第 5 和第 6 路 DO 信號置為 false，將第 7 和第 8 路信號置為 true

5.5.3 setdoinmv(不停前瞻異步輸出單路 DO)

函數原型

setdoinmv channel:int chan,value:bool value

描述

設置一路 DO 信號為 true 或者 false，在下一行運動指令開始前輸出。

參數

表 5-41 setdoinmv 指令的參數

名稱	說明
chan	類型：int
	參數範圍為[1,DO 埠數]。如：[1,32*DO 從站數]（針對 MIII 總線版控制系統）或[1, 40*PLC_MF 從站數]（針對 RTEX 總線版控制系統）
value	類型：bool
	指定輸出值

返回值

無

用法舉例

```
movej j:j1,vp:5%,sp:5%
setdoinmv channel:3,value:true //將第 3 通道置為 true
movej j:j2,vp:5%,sp:5%
```

5.5.4 syncdo(同步輸出單路 DO)

函數原型

```
void syncdo(int chan,bool value)
```

描述

同步輸出一路 DO 信號為 true 或者 false。同步輸出與異步輸出的區別為同步輸出保證 IO 埠信號已經輸出後再繼續向下執行程序，而異步輸出則將數據發送出去後便繼續向下執行程序。

參數

表 5-42 syncdo 函數的參數

名稱	說明
chan	類型：int
	參數範圍為[1,DO 埠數]。如：[1,32*DO 從站數]（針對 MIII 總線版控制系統）或[1,40*PLC_MF 從站數]（針對 RTEX 總線版控制系統）
value	類型：bool
	指定輸出值

返回值

無

用法舉例

```
syncdo(3,true) //將第 3 通道置為 true
```

5.5.5 syncdo(同步輸出多路 DO)

函數原型

void syncdo(int from_chan,int to_chan,int value)

描述

同步輸出連續多路 DO 信號。同步輸出與異步輸出的區別為同步輸出保證 IO 埠信號已經輸出後再繼續向下執行程序，而異步輸出則將數據發送出去後便繼續向下執行程序。

參數

表 5-43 syncdo 函數的參數

名稱	說明
from_chan	類型：int
	參數範圍為[1,DO 埠數]。如：[1,32*DO 從站數]（針對 MIII 總線版控制系統）或 [1,40*PLC_MF 從站數]（針對 RTEX 總線版控制系統）
to_chan	類型：int
	結束通道號。參數範圍為[1,DO 埠數]。如：[1,32*DO 從站數]（針對 MIII 總線版控制系統）或 [1,40*PLC_MF 從站數]（針對 RTEX 總線版控制系統） from_chan <= to_chan
value	類型：int
	因為 int 型數據長度為 32 位，所以這裡最多只能同時設置連續的 32 路 DO

返回值

無

用法舉例

```
syncdo(5,8,1100b)
//將第 5 和第 6 路 DO 信號置為 false，將第 7 和第 8 路信號置為 true
```

5.5.6 pulsedo(輸出單路脈衝信號)

函數原型

void pulsedo(int chan,bool value,double width)

描述

該函數用於輸出單路指定脈寬的脈衝信號。相比使用 setdo 函數配合 waittime 指令實現的脈衝信號，使用 pulsedo 函數更加高效，因為它不會等待脈衝信號執行完，就會繼續向下執行程序，也就是說脈衝輸出的執行和程序的執行是並行的。

參數

表 5-44 pulsedo 函數的參數

名稱	說明
chan	類型：int
	參數範圍為[1,DO 埠數]。如：[1,32*DO 從站數]（針對 MIII 總線版控制系統）或[1,40*PLC_MF 從站數]（針對 RTEX 總線版控制系統）
value	類型：bool
	true 表示輸出高脈衝，false 表示輸出低脈衝。
width	類型：double
	指定脈衝寬度，單位 s。該值必須大於等於 0，最小的輸出脈寬約 25ms 左右

返回值

無

用法舉例

```
pulsedo(5,true,0.2)
//第 5 路 DO 輸出脈寬為 200ms 的高脈衝信號
```

註意事項

- 當一個脈衝輸出函數還未執行完時，在該路 DO 又執行了另一個 pulsedo 函數，則尚未執行完的那個 pulsedo 信號會終止。

  例如，對於如下程序：

  ```
  pulsedo(5,true,3)
  waittime 2
  pulsedo(5,false,1)
  ```

  將在第 5 路 DO 上輸出如下圖 5-1 中的波形：

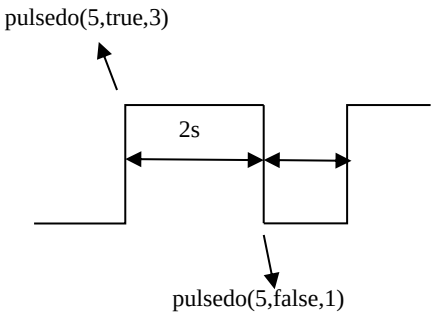


圖 5-1 程序示例對應波形示意圖

- 某個通道中如果存在尚未執行完畢的脈衝信號時，則該通道將一直處於運行或暫停狀態，直到所有的脈衝輸出信號全部執完畢才會變成停止狀態。
- 程序暫停時脈衝輸出信號不會停止執行。
- 程序複位時，如果存在未執行完畢的脈衝輸出信號，則這些信號會立即停止執行。

5.5.7 getdo(獲取單路 DO)

函數原型

bool getdo(int chan)

描述

獲取單路 DO 信號值。

參數

表 5-45 getdo 函數的參數

名稱	說明
chan	類型：int
	參數範圍為[1,DO 埠數]。如：[1,32*DO 從站數]（針對 MIII 總線版控制系統）或[1,40*PLC_MF 從站數]（針對 RTEX 總線版控制系統）

返回值

類型：bool

獲取到的第 chan 路 DO 信號值。

用法舉例

```
int chan = 3
setdo(chan,true)
bool value = getdo(chan)
print value //輸出“true”
```

5.5.8 getdo(獲取多路 DO)

函數原型

int getdo(int from_chan,int to_chan)

描述

獲取連續多路 DO 信號值。

參數

表 5-46 getdo 函數的參數

名稱	說明
from_chan	類型：int
	起始通道號。參數範圍為[1,DO 埠數]。如：[1,32*DO 從站數]（針對 MIII 總線版控制系統）或[1, 40*PLC_MF 從站數]（針對 RTEX 總線版控制系統）
to_chan	類型：int
	結束通道號。參數範圍為[1,DO 埠數]。如：[1,32*DO 從站數]（針對 MIII 總線版控制系統）或[1, 40*PLC_MF 從站數]（針對 RTEX 總線版控制系統）

名稱	說明
	from_chan <= to_chan

返回值

類型：int

因為 int 型數據長度為 32 位，所以這裡最多只能同時獲取連續 32 路 DO 信號的值。

用法舉例

```
int from = 5
int to = 8
setdo(from,to,1001b)
print getdo(from,to) //輸出“9”
```

5.5.9 getdi(獲取單路 DI)

函數原型

```
bool getdi(int chan)
```

描述

獲取單路 DI 信號值。

參數

表 5-47 getdi 函數的參數

名稱	說明
chan	類型：int
	參數範圍為[1,DO 埠數]。如：[1,32*DO 從站數]（針對 MIII 總線版控制系統）或[1, 40*PLC_MF 從站數]（針對 RTEX 總線版控制系統）

返回值

類型：bool

獲取到的第 chan 路 DI 信號值

用法舉例

```
//假設第 4 通道 DI 輸入了高電平
print getdi(4) //輸出“true”
```

5.5.10 getdi(獲取多路 DI)

函數原型

```
int getdi(int from_chan,int to_chan)
```

描述

獲取連續多路 DI 信號值。

參數

表 5-48 getdi 函數的參數

名稱	說明
from_chan	類型：int
	起始通道號。參數範圍為[1,DO 埠數]。如：[1,32*DO 從站數]（針對 MIII 總線版控制系統）或[1, 40*PLC_MF 從站數]（針對 RTEX 總線版控制系統）
to_chan	類型：int
	結束通道號。參數範圍為[1,DO 埠數]。如：[1,32*DO 從站數]（針對 MIII 總線版控制系統）或[1, 40*PLC_MF 從站數]（針對 RTEX 總線版控制系統） from_chan <= to_chan

返回值

類型：int

因為 int 型數據長度為 32 位，所以這裡最多只能同時獲取連續 32 路 DI 信號的值。

用法舉例

```
//假設第 6 到第 9 通道的 DI 輸入了高電平
int value = getdi(6,9)
print value //輸出“15”
```

5.5.11 getai(獲取模擬量輸入信號)

函數原型

double getai(int chan)

描述

獲取一路模擬量信號輸入值。其中，AI 信號類型包括：電流型和電壓型，通過 PLC 從站類型配置 AI 信號類型。當輸入信號為電流型時，返回值單位為毫安；當輸入信號為電壓型時，返回值單位為伏。

參數

表 5-49 getai 函數的參數

名稱	說明
chan	類型：int
	參數範圍為[1,DO 埠數]。如：[1,32*DO 從站數]（針對 MIII 總線版控制系統）或[1,40*PLC_MF 從站數]（針對 RTEX 總線版控制系統）

用法舉例

print getai(2) //輸出“第 2 通道的模擬量輸入值”（單位與配置的 AI 信號類型有關）

5.5.12 getnostopdi（不停前瞻異步獲取單路 DI）

函數原型

bool getnostopdi(int chan)

描述

獲取單路 DI 信號值，在下一行運動指令開始前獲取。

與 getdi 相比，該指令不停前瞻，所以不會導致機器人停止卡頓等問題。但是如果前瞻條數太多，可能會使機器人在未執行完前面的運動語句時就觸發該指令，導致產生不期望的動作。

參數

名稱	說明
chan	類型：int
	參數範圍為[1,DO 埠數]。如：[1,32*DO 從站數]（針對 MIII 總線版控制系統）或[1, 40*PLC_MF 從站數]（針對 RTEX 總線版控制系統）

返回值

類型：bool

獲取到的第 chan 路 DI 信號值

用法舉例

//假設第 4 通道 DI 輸入了高電平
print getnostopdi(4) //輸出“true”

5.5.13 setao(異步輸出模擬量信號)

函數原型

void setao(int chan,double value)

描述

設置一路模擬量信號輸出。其中，AO 信號類型包括：電流型和電壓型，通過 PLC 從站類型配置 AO 信號類型。

參數

表 5-50 setao 函數的參數

名稱	說明
chan	類型：int

名稱	說明
	模擬量輸出埠號，參數範圍[1,2*AO 從站數]（針對 MIII 總線版控制系統）或[1,2*PLC_MF 從站數]（針對 RTEX 總線版控制系統）
value	類型：double
	當輸出信號為電流型時，參數範圍[4,20]，單位毫安；當輸出信號為電壓型時，參數範圍[-10,10]，單位伏

用法舉例

setao(2,5.2) //若第 2 路 AO 配置為電流型時，則將第 2 通道置為 5.2mA
setao(2,5.2) //若第 2 路 AO 配置為電壓型時，則將第 2 通道置為 5.2V

5.5.14 syncao(同步輸出模擬量信號)

函數原型

void syncao(int chan,double value)

描述

同步輸出一路模擬量信號。同步輸出與異步輸出的區別為同步輸出保證 IO 埠信號已經輸出後再繼續向下執行程序，而異步輸出則將數據發送出去後便繼續向下執行程序。

參數

表 5-51 syncaov 函數的參數

名稱	說明
chan	類型：int
	模擬量輸出埠號，參數範圍[1,2*AO 從站數]（針對 MIII 總線版控制系統）或[1,2*PLC_MF 從站數]（針對 RTEX 總線版控制系統）
value	類型：double
	當輸出信號為電流型時，參數範圍[4,20]，單位毫安；當輸出信號為電壓型時，參數範圍[-10,10]，單位伏

返回值

無

用法舉例

syncao(2,5.2) //若第 2 路 AO 配置為電流型時，則將第 2 通道置為 5.2mA
syncao(2,5.2) //若第 2 路 AO 配置為電壓型時，則將第 2 通道置為 5.2V

5.5.15 getao(獲取模擬量輸出信號)

函數原型

double getao(int chan)

描述

獲取一路模擬量信號輸出值。其中，AO 信號類型包括：電流型和電壓型，通過 PLC 從站類型配置 AO 信號類型。當輸出信號配置為電流型時，返回值單位為毫安；當輸出信號配置為電壓型時，返回值單位為伏。

參數

表 5-52 getao 函數的參數

名稱	說明
chan	類型：int
	模擬量輸出埠號，參數範圍[1,2*AO 從站數]（針對 MIII 總線版控制系統）或[1,2*PLC_MF 從站數]（針對 RTEX 總線版控制系統）

返回值

類型：double

返回第 chan 路模擬量埠信號的值。

用法舉例

```
int chan = 1
setao(chan,5.2)
double value = getao(chan)
print value //輸出“5.2”
```

5.5.16 getintdo(讀取單路 PLC_INT 的 DO 信號值)

函數原型

bool getintdo(int chan)

描述

獲取 INT 單路 DO 信號值。

參數

表 5-53 getintdo 函數的參數

名稱	說明
chan	類型：int
	參數範圍為[1,DO 埠數]。如：[1,32*DO 從站數]（針對 MIII 總線版控制系統）或[1,40*PLC_MF 從站數]（針對 RTEX 總線版控制系統）

返回值

類型：bool

獲取到的 INT 第 chan 路 DO 信號值。

用法舉例

bool value = getintdo(chan)

print value // DO status of channel chan on output INT

5.5.17 getintdi(讀取單路 PLC_INT 的 DI 信號值)

函數原型

bool getintdi(int chan)

描述

獲取 INT 單路 DI 信號值。

參數

表 5-54 getintdi 函數的參數

名稱	說明
chan	類型：int
	參數範圍為[1,DO 埠數]。如：[1,32*DO 從站數]（針對 MIII 總線版控制系統）或[1,40*PLC_MF 從站數]（針對 RTEX 總線版控制系統）

返回值

類型：bool

獲取到的 INT 第 chan 路 DI 信號值。

用法舉例

bool value = getintdi(chan)

print value //輸出 INT 上第 chan 路 DI 狀態

5.5.18 setpwm(設置一路 PWM 通道的頻率和占空比)

函數原型

bool setpwm(int channel, int freq, int ratio)

描述

設置一路 PWM 通道的頻率和占空比。

參數

表 5-55 setpwmi 函數的參數

名稱	說明
channel	類型：int
	PWM 通道號，參數範圍：[1,3]
freq	類型：int
	PWM 頻率，單位：Hz，參數範圍：[10,10000]
chan	類型：int
	PWM 占空比，單位：%，參數範圍：[0,100]

返回值

類型：bool

用法舉例

setpwm(1,20,50) //對第 1 路 PWM 通道進行設置，頻率設為 20Hz，占空比設為 0.5

5.6 通信相關函數

5.6.1 socket 通信相關函數

5.6.1.1 setip(設置對外網口 IP)

函數原型

bool setip(string ip,string gate,string mask,string if_name)

bool setip(string ip,string gate,string mask)

描述

如果想要通過對外網口與外部設備通信，首先需要設置網口 IP，網關，子網掩碼和網口名稱。可以通過此函數來進行設置。

參數

表 5-56 setip 函數的參數

名稱	說明
ip	類型：string
	要設置的 IP 字元串
gate	類型：string
	要設置的網關
mask	類型：string
	要設置的子網掩碼

名稱	說明
if_name	類型：string
	要設置的網口名稱。註：對於 SCARA 控制櫃，有三個用戶網口：eth1、eth2、eth3，對於其他控制櫃，只有一個用戶網口：eth1(當該參數預設時，默認為 eth1)

返回值

類型：bool

true 表示設置成功，false 表示設置失敗。

設置 IP 時，網關地址也必須是真實存在的 IP，否則返回 false。

用法舉例

setip("10.20.210.93","10.20.210.255","255.255.255.0","eth1") //設置的用戶網口名稱為 eth1
setip("10.20.210.93","10.20.210.255","255.255.255.0") //設置的用戶網口名稱為 eth1

5.6.1.2 getip(獲取對外網口 IP)

函數原型

bool getip(string& ip,string if_name)

bool getip(string& ip)

描述

如果想要通過獲取對外網口 IP 字元串。可以通過此函數來進行設置。

參數

表 5-57 getip 函數的參數

名稱	說明
ip	類型：string
	要設置的 IP 字元串
if_name	類型：string
	指定獲取哪個用戶網口的 IP。註：對於 SCARA 控制櫃，有三個用戶網口：eth1、eth2、eth3，對於其他控制櫃，只有一個用戶網口：eth1(當該參數預設時，默認為 eth1)

返回值

類型：bool

true 表示獲取成功，false 表示獲取失敗。

用法舉例

string ip
getip(ip,"eth2") //獲取名稱為"eth2"的用戶網口的 IP

getip(ip) //獲取名稱為"eth1"的用戶網口的 IP

5.6.1.3 connect(發起 socket 連接)

函數原型

bool connect(socket s,string ip,int port)

描述

如果希望機器人控制器作為發起連接的一方，也就是 client 端，則需要使用此函數發起 socket 連接。須配合 waituntil 指令實現同步連接，也就是說等待直到連接成功程序才繼續運行。

參數

表 5-58 connect 函數的參數

名稱	說明
s	類型：socket
	預先定義好的套接字變量
ip	類型：string
	要連接的對方設備 IP 字元串
port	類型：int
	要連接的對方設備的埠

返回值

類型：bool

未連接成功時，返回 false；當返回 true 時表示連接成功。

用法舉例

```
socket s
//等待，直到連接成功，這裡也可以使用 waituntil 指令的 maxtime
//等參數實現超時機制
waituntil connect(s,"192.168.0.40",2888)
```

5.6.1.4 accept(接受 socket 連接)

函數原型

bool accept(socket s,string ip,int port)

描述

如果希望機器人控制器作為被動接受連接的一方，也就是 server 端，則需要使用此函數接受 socket 連接。須配合 waituntil 指令實現同步連接，也就是說等待直到對方發起連接才繼續運行。

參數

表 5-59 accept 函數的參數

名稱	說明
s	類型：socket
	預先定義好的套接字變量
ip	類型：string
	本機 IP 字元串
port	類型：int
	本機埠

返回值

類型：bool

未建立連接時，返回 false；當返回 true 時表示建立連接成功。

用法舉例

```
socket s
//等待，直到建立連接，這裡也可以使用 waituntil 指令的 maxtime
//等參數實現超時機制
waituntil accept(s,"192.168.0.40",2888)
```

5.6.1.5 write(通過 socket 發送數據)

函數原型

bool write(socket s,string data)或

bool write(socket s,byte[] data,int len)

描述

不管是 server 端還是 client 端均通過此函數發送數據。須配合 waituntil 指令實現同步發送，也就是說等待直到數據發送完成才繼續運行。

參數

表 5-60 write 函數的參數

名稱	說明
s	類型：socket
	預先定義好的套接字變量
data	類型：string/byte[]
	要發送的字元串數據或二進制數據

名稱	說明
len	類型：int
	當發送的數據為位元組數組時，該參數表示寫入的位元組長度

返回值

類型：bool

發送未完成時，返回 false；當返回 true 時表示發送完成。

用法舉例

```
socket s
waituntil connect(s,"192.168.0.40",2888)
waituntil write(s,"hello")
byte bdata[3] = {1,2,3}
waituntil write(s,bdata,3)
```

socket 斷線重連功能

當 write 過程中出現 socket 斷開時，ARCS 會產生報警並停止運行，如果此時不希望程序停止，可在斷線後嘗試重連的方法，可在 ARL 程序中進行設置：$DETECT_SOCK_CLOSE=true。

斷線後設置當前 socket 的錯誤狀態為$ERR_SOCK_CLOSED，用戶可以通過 geterror(s)來獲取 socket s 的錯誤狀態。

當需要使用斷線重連功能時，使用定時中斷 timer 會影響斷線狀態的檢測，建議使用 while 和 waittime 替代定時中斷 timer。

實現斷線重連的方法如下：

用戶可以聲明中斷，當某個 socket 的錯誤狀態為$ERR_SOCK_CLOSED 時，觸發某個中斷處理函數，在該中斷處理函數中重新進行 socket 連接。

用法舉例：

```
socket s
func void handler()
close(s)
waituntil connect(s,"10.20.220.1",8080)
endfunc
func viod main()
init()
$DETECT_SOCK_CLOSE=true
interrupt 1, when:geterror(s)==$ERR_SOCK_CLOSED, do:handler()
setip(.....)
waituntil connect(s,.......)
waituntil write (s,......)
```

endfunc

5.6.1.6 read(通過 socket 接收數據)

函數原型

bool read(socket s,string data,int len)

bool read(socket s,byte[] data,int len)

描述

不管是 server 端還是 client 端均通過此函數接收指定長度的數據。須配合 waituntil 指令實現同步接收，也就是說等待直到數據接收完成才繼續運行。

參數

表 5-61 read 函數的參數

名稱	說明
s	類型：socket
	預先定義好的套接字變量
data	類型：string/byte[]
	預先定義的字元串變量或位元組數組變量
len	類型：int
	接收數據長度，當 data 參數為 string 類型時，該長度指字元串中字元個數；當 data 參數為位元組數組類型時，該長度指接收的位元組個數

返回值

類型：bool

未接收到指定長度的字元串時，返回 false；當返回 true 時表示接收完成。

用法舉例

```
socket s
waituntil connect(s,"192.168.0.40",2888)
string data
waituntil read(s,data,5)
print data
byte bdata[3]
waituntil read(s,bdata,3)
```

socket 斷線重連功能

當 read 過程中出現 socket 斷開時，ARCS 會產生報警並停止運行，如果此時不希望程序停止，可在斷線後嘗試重連的方法，可在 ARL 程序中進行設置，$DETECT_SOCK_CLOSE=true。

斷線後設置當前 socket 的錯誤狀態為$ERR_SOCK_CLOSE，用戶可以通過 geterror(s)來獲取 socket s 的錯誤狀態。

實現斷線重連的方法如下：

用戶可以聲明中斷，當某個 socket 的錯誤狀態為$ERR_SOCK_CLOSED 時，觸發某個中斷處理函數，在該中斷處理函數中重新進行 socket 連接。

用法舉例：

```
socket s
func void handler()
close(s)
waituntil connect(s, "10.20.220.1", 8080)
endfunc
func viod main()
init()
$DETECT_SOCK_CLOSE=true
interrupt 1, when:geterror(s)== $ERR_SOCK_CLOSED, do:handler()
setip(.....)
waituntil connet(s,.......)
waituntil read (s,......)
endfunc
```

5.6.1.7 readuntil(通過 socket 接收數據)

函數原型

bool readuntil(socket s,string data,string cut)

描述

不管是 server 端還是 client 端均通過此函數接收直到讀到某個指定的字元或字元串的數據。須配合 waituntil 指令實現同步接收，也就是說等待直到接收到指定的字元或字元串數據才繼續運行。

參數

表 5-62 readuntil 函數的參數

名稱	說明
s	類型：socket
	預先定義好的套接字變量
data	類型：string
	預先定義的字元串變量
cut	類型：string
	指定截斷字元或字元串，如果希望該字元為換行符，則使用\n 作為這裡的 cut 參數

返回值

類型：bool

未接收到指定字元或字元串時，返回 false；當返回 true 時表示接收到指定的字元或字元串。

用法舉例

```
socket s
waituntil connect(s,"192.168.0.40",2888)
string data
//如果對方發過來的數據是“hello”,則這裡 data 將列印出 he
waituntil readuntil(s, data, "e")
print data
```

5.6.1.8 clearbuff(清空 socket 的輸入緩衝區)

函數原型

bool clearbuff(socket s)

描述

清空 socket 的輸入緩衝區（當輸入緩衝區為空時，通過 read 函數接受不到任何數據）。

參數

表 5-63 clearbuff 函數的參數

名稱	說明
s	類型：socket
	預先定義好的套接字變量

返回值

類型：bool

返回 true 表示操作成功，false 表示操作失敗。

用法舉例

```
socket s
waituntil accept(s,"192.168.1.1",8080), maxtime:10,timeoutflag:time_flag
waituntil read(s,data,3), maxtime:10,timeoutflag:time_flag
clearbuff（s）
……
```

5.6.2 串口通信相關函數

5.6.2.1 open(打開串口設備)

函數原型

bool open(iodev& dev,string devname,int baud_rate,int character_size,stopbits stop_bits,parity prt)

描述

打開並配置一個串口設備。

參數

表 5-64 open 函數的參數

名稱	說明
dev	類型：iodev
	預先定義好的 IO 設備
devname	類型：string
	柜子類型為 ARCC 系列時： 設備名，“/dev/ttyS0”表示 com1，“/dev/ttyS1”表示 com2 柜子類型為 ARCCD 系列時： 設備名，“/dev/ttyS0”表示 com1
baud_rate	類型：int
	波特率，可以設置為 300，600，1200，2400，4800，9600，19200，38400，57600,115200，單位 bps
character_size	類型：int
	字元大小，可以設置為 5，6，7，8
stop_bits	類型：stopbits
	停止位類型，參見 stopbits 枚舉類型，可以設置為 1 位和 2 位
prt	類型：parity
	奇偶校驗類型，參見 parity 枚舉類型，可以設置為不校驗，奇校驗或偶校驗

返回值

類型：bool

返回 true 表示打開設備成功。

用法舉例

```
iodev s
open(s,"/dev/ttyS0",9600,8,one,none)
waituntil write(s, "hello")
```

5.6.2.2 close(關閉串口設備)

函數原型

void close(iodev& dev)

描述

關閉一個之前打開的串口設備。

參數

表 5-65 close 函數的參數

名稱	說明
dev	類型：iodev
	之前打開的 IO 設備

返回值

無

用法舉例

```
iodev s
open(s,"/dev/ttyS0",9600,8,one,none)
waituntil write(s,"hello")
close(s)
```

5.6.2.3 read/write/readuntil(讀寫串口設備)

描述

以上 3 個函數的用法與 socket 通信函數中的 read/write/readuntil 用法完全相同，只要將第一個參數換成 iodev 即可。

用法舉例

```
iodev s
//打開 com1 口
open(s,"/dev/ttyS0",9600,8,one,none)
waittime 1
//寫字元串
write(s,"hello")
write(s,"world")
string data
//按結束符讀字元串
waituntil readuntil(s,data,"o")
print "data:",data
//按數量讀字元串
waituntil read(s,data,4)
//寫二進制數據
byte sdata[4]={1,2,3,4}
write(s,sdata,4)
close(s)
```

5.6.2.4 clearbuff(清空串口設備的輸入緩衝區)

函數原型

bool clearbuff(iodev& dev)

描述

清空串口設備的輸入緩衝區。

參數

表 5-66 clearbuff 函數的參數

名稱	說明
dev	類型：iodev
	之前打開的 IO 設備

返回值

類型：bool

返回 true 表示操作成功，false 表示操作失敗。

用法舉例

```
iodev s
open(s,"/dev/ttyS0",9600,8,one,none)
waituntil write(s,"hello")
clearbuff（s）
……
close(s)
```

5.6.3 devicenet 總線通信相關函數

5.6.3.1 dnwrite(向 devicenet 總線發送數據)

函數原型

int dnwrite(int offset,int len, byte[] data)

描述

向 devicenet 總線 buffer 寫入數據。

參數

表 5-67 dnwrite 函數的參數

名稱	說明
offset	類型：int

名稱	說明
	起始位元組的偏移地址
len	類型：int
	寫入數據的長度，單位：位元組數
data	類型：byte[]
	要發送的二進制數據數組

返回值

類型：int

返回 0 時，表示成功寫入。非 0 表示寫入失敗，值表示錯誤碼。

用法舉例

byte bdata[3] = {1,2,3}
dnwrite(0,3,bdata) //從位元組 0 開始，向 devicenet 設備寫 3 個位元組數據

5.6.3.2 dnread(從 devicenet 總線讀入數據)

函數原型

int dnread(int offset,int len, byte[] data)

描述

從 devicenet 總線 buffer 讀入數據。

參數

表 5-68 dnread 函數的參數

名稱	說明
offset	類型：int
	起始位元組的偏移地址
len	類型：int
	讀入數據的長度，單位：位元組數
data	類型：byte[]
	要讀入的二進制數據數組

返回值

類型：int

返回 0 時，表示成功讀出。非 0 表示讀失敗，值表示錯誤碼。

用法舉例

byte bdata[3]

dnread(0,3,bdata) //從位元組 0 開始，從 devicenet 設備讀出 3 個位元組數據

5.6.4 Melsec 通訊相關的 ARL 函數

5.6.4.1 open(打開 melsec 從站設備)

函數原型

bool open(melsec_dev dev, string ip, int port, int slave_id)

描述

打開並配置一個 Melsec 協議 PLC 設備

參數

表 5-69 open 函數的參數

名稱	說明
dev	類型：melsec_dev
	預先定義好的 melsec 套接字變量
ip	類型：string
	設備 ip,比如取值“192.168.1.101”
port	類型：int
	要連接的對方設備的埠
slave_id	類型：int
	Melsec 從站號，取值範圍:0~247 以及 255；其中 0 是 Melsec 廣播地址;在 TCP 模式下，255 能被用來恢復默認值。

返回值

類型：bool

true 表示打開成功，false 表示打開失敗。

用法舉例

melsec_dev mc //定義套接字，與 socket 功能類似

waituntil open(mc,"10.20.220.92",8080,255)

5.6.4.2 close(關閉 melsec 從站設備)

函數原型

void close(melsec_dev dev)

描述

關閉一個已打開的 Melsec 協議 PLC 設備

參數

表 5-70 close 函數的參數

名稱	說明
dev	類型：melsec_dev
	預先定義好的 melsec 套接字變量

返回值

類型：void

用法舉例

close(mc) //關閉 Melsec 協議 PLC 設備

5.6.4.3 read(通過 melsec 從主站接收數據)

函數原型

bool read(melsec_dev dev, string data, in start, int len)

bool read(melsec_dev dev, byte[] data, in start, int len)

描述

以 start 為起始地址、len 為長度(單位：位元組)，將 plc 寄存器內容讀取至預先定義的字元串變量或位元組數組變量中。

不管是 server 端還是 client 端均通過此函數接收指定長度的數據。須配合 waituntil 指令實現同步接收，也就是說等待直到數據接收完成才繼續運行。

參數

表 5-71 read 函數的參數

名稱	**說明**
dev	類型：melsec_dev
	預先定義好的 melsec 套接字變量
data	類型：string 或 byte[]
	預先定義的字元串變量或位元組數組變量
start	類型：int
	起始偏移地址
len	類型：int

名稱	說明
	接收數據長度，當 data 參數為 string 類型時，該長度指字元串中字元個數；當 data 參數為位元組數組類型時，該長度指接收的位元組個數。

返回值

類型：bool

未接收到指定長度的字元串時，返回 false；當返回 true 時表示接收完成。

用法舉例

```
melsec mc
waituntil open(mc,"10.20.220.92",8080,255)
string data
waituntil read(mc,data,6000,10)
byte bdata24]
waituntil read(mc,bdata,6000,24)
```

5.6.4.4 write(通過 melsec 向主站發送數據)

函數原型

bool write(melsec_dev dev, string data, in start, int len)

bool write(melsec_dev dev, string data, in start).

bool write(melsec_dev dev, byte[] data, in start, int len)

描述

以 start 為起始地址、len 為長度(單位：位元組)，把輸入發送的字元串變量或位元組數組變量寫入 plc 寄存器。

不管是 server 端還是 client 端均通過此函數發送數據。須配合 waituntil 指令實現同步發送，也就是說等待直到數據發送完成才繼續運行。

參數

表 5-72 write 函數的參數

名稱	說明
dev	類型：melsec_dev
	預先定義好的 melsec 套接字變量
data	類型：string 或 byte[]
	預先定義的字元串變量或位元組數組變量
start	類型：int
	起始偏移地址

名稱	說明
len	類型：int
	發送數據長度，當 data 參數為 string 類型時，該長度指字元串中字元個數(參數預設時默認為 string 的長度)；當 data 參數為位元組數組類型時，該長度指接收的位元組個數。

返回值

類型：bool

發送未完成時，返回 false；當返回 true 時表示發送完成。

用法舉例

```
melsec mc
waituntil open(mc,"10.20.220.92",8080,255)
string data = "abcd"
waituntil write(mc1,data,6000)
byte bdata[4] = {1,2,3,4}
waituntil write (mc,bdata,6000,4)
```

5.6.5 modbus-rtu 通信相關函數

5.6.5.1 open(打開 modbus 從站設備)

函數原型

bool open(modbus_dev& dev, string devname, int slave_id, int baud_rate, ParityType parity, int databits, StopBitType stop_bits)

描述

打開並配置一個 modbus 從站設備。

參數

表 5-73 參數說明

參數名稱	參數類型	說明
dev	modbus_dev	預先定義好的 modbus 設備
devname	string	設備名，對於二代櫃來說，設備名為”rtserMB0”
slave_id	int	Modbus 從站號，取值範圍 1-247
baud_rate	int	波特率，可以設置為 4800，9600，19200，38400，57600,115200，單位是 bps
parity	ParityType	奇偶校驗類型，參見 modbus_parity 枚舉類型，可以設置為不校驗，奇校驗或偶校驗
databits	int	數據位，可以設置為 5，6，7，8
stop_bits	StopBitType	停止位類型，參見 modbus_stopbits 枚舉類型，可以設置為 1 位或 2 位

返回值

類型：bool

返回 true 表示操作成功，false 表示操作失敗。

用法舉例

```
modbus_dev m
open(m,"rtserMB0",1,115200,one,8,none)
```

5.6.5.2 close(關閉 modbus 從站設備)

函數原型

```
void close(modbus_dev& dev)
```

描述

關閉一個之前打開的 modbus 從站設備。

參數

表 5-74 參數說明

參數名稱	參數類型	說明
dev	modbus_dev	之前打開的 modbus 設備

返回值

無

用法舉例

```
modbus_dev m
open(m,"rtserMB0",1,115200,one,8,none)
close(m)
```

5.6.5.3 read(通過 modbus 從主站接收數據)

函數原型

```
bool read(modbus_dev dev, string data, in start, int len)

bool read(modbus_dev dev, byte[] data, in start, int len)
```

描述

以 start 為起始地址、len 為長度(單位：位元組)，將 modbus 從站設備寄存器內容讀取至預先定義的字元串變量或位元組數組變量中。

參數

表 5-75 參數說明

參數名稱	參數類型	說明
dev	modbus_dev	已經打開的 modbus 設備
data	string/byte[]	預先定義的字元串變量或位元組數組變量
start	int	所要讀取數據的起始地址
len	int	所要讀取數據的長度

返回值

類型：bool

未接收到指定長度的字元串時，返回 false；當返回 true 時表示接收完成。

用法舉例

```
modbus_dev m
open(m,"rtserMB0",1,115200,one,8,none)
string data
read(m, data, 0, 4)
close(m)
```

5.6.5.4 write(通過 modbus 向主站發送數據)

函數原型

bool write(modbus_dev dev,string data,in start)

bool write(modbus_dev dev,byte[] data,in start,int len)

描述

以 start 為起始地址、len 為長度(單位：位元組)，將輸入發送的字元串變量或位元組數組變量寫入 modbus 從站設備寄存器中。

參數

表 5-76 參數說明

參數名稱	參數類型	說明
dev	modbus_dev	已經打開的 modbus 設備
data	string/byte[]	要發送的字元串變量或位元組數組變量
start	int	所要寫入數據的起始地址
len	int	所要寫入數據的長度，寫入 string 類型數據不需要該參數

返回值

類型：bool

發送未完成時，返回 false；當返回 true 時表示發送完成。

用法舉例

```
modbus_dev m
open(m,"rtserMB0",1,115200,one,8,none)
byte sdata[4]={31h,32h,33h,34h}
write(m, sdata, 0, 4)
close(m)
```

5.6.5.5 clearbuff(清空 modbus 從站的數據緩衝區)

函數原型

```
bool clearbuff(modbus_dev dev)
```

描述

清空 modbus 從站的數據緩衝區（當輸入緩衝區為空時，通過 read 函數接受不到任何數據）。

參數

表 5-77 參數說明

參數名稱	參數類型	說明
dev	modbus_dev	已經打開的 modbus 設備

返回值

類型：bool

返回 true 表示操作成功，false 表示操作失敗。

用法舉例

```
modbus_dev m
open(m,"rtserMB0",1,115200,one,8,none)
string data
read(m, data, 0, 4)
clearbuff(m)
close(m)
```

5.6.5.6 setcycle(設置與 modbus 設備通訊的讀寫周期)

函數原型

```
void setcycle(modbus_dev& dev,int cycle)
```

描述

設置控制櫃與 modbus 設備通訊的讀寫周期。

參數

表 5-78 參數說明

參數名稱	參數類型	說明
dev	modbus_dev	已經打開的 modbus 設備
cycle	int	讀寫周期，單位：μs，參數範圍：>500000

返回值

無

用法舉例

```
modbus_dev m
open(m,"rtserMB0",1,115200,one,8,none)
setcycle(m,1000000)
clearbuff(m)
close(m)
```

5.7 數據類型轉換函數

5.7.1 toint(轉換成整型)

函數原型

int toint(double d)或

int toint(byte[] data,int start)

int toint(string number_string, base number_base)

描述

將 double 型數據類型、byte 數組或者字元串數據轉換成整型。對於 byte 數組，系統會將從 data 數組的第 start 個位元組開始的 4 個位元組轉換為整型數據。

參數

表 5-79 toint 函數的參數

名稱	說明
d	類型：double
	被轉換的 double 型數據
data	類型：byte[]
	被轉換的 byte 數組

名稱	說明
start	類型：int
	起始偏移地址
number_string	類型：string
	被轉換的字元串數據
number_base	類型：base
	字元串數據的顯示數制

返回值

類型：int

轉換後的整型數據。

用法舉例

```
double value =3.1415
int s = toint(value)
print s //輸出“3”
byte data[4] = {01h,02h,03h,04h}
print hex,toint(data,0) //輸出“4030201h”
```

5.7.2 todouble(轉換成浮點型)

函數原型

```
double todouble(byte[] data,int start)
```

描述

將 byte 數組轉換成浮點型。系統會將從 data 數組的第 start 個位元組開始的 8 個位元組轉換為浮點型數據。

參數

表 5-80 todouble 函數的參數

名稱	說明
data	類型：byte[]
	被轉換的 byte 數組
start	類型：int
	起始偏移地址

返回值

類型：double

轉換後的浮點型數據。

用法舉例

```
byte data[8] = {00h,00h,00h,00h,00h,00h,08h,40h}
print data //輸出 3
byte data[8] = {00h,00h,00h,00h,00h,00h,08h,C0h}
print data //輸出-3
byte data[8] = {00h,00h,00h,00h,00h,00h,00h,40h}
print data //輸出 2
```

5.7.3 tobytes(轉換成位元組數組)

函數原型

void tobytes(string/int/double src,byte[] dest,int start)

描述

將 string 型、int 型或 double 型數據類型轉換成 byte 數組。int 數據會被轉換為 4 個位元組，double 數據會被轉換為 8 個位元組，string 類型轉換成 byte，長度無法確定，要根據轉換的字元串來看，不同的字元串長度是不同的。

參數

表 5-81 tobytes 函數的參數

名稱	說明
dest	類型：byte[]
	轉換後數據存入的 byte 數組
start	類型：int
	存儲的起始偏移地址

返回值

類型：void

用法舉例

```
byte result[4]
tobytes(14030201h,result,0)
print hex,result //輸出“{01h,02h,03h,14h}”
```

5.7.4 tostr(強制轉換成字元串)

函數原型

string tostr(anytype v)

string tostr(double v,int precision)

string tostr(int v, enum number_base) //將整型轉化為 10 進制/16 進制顯示的字元串

描述

將任意數據類型的數據轉換成字元串類型。

參數

表 5-82 tostr 函數的參數

名稱	說明
v	類型：可以是任意數據類型
	被轉為字元串的表達式 ■ 當 v 是 double 類型時，可以指定第二個參數 precision 設置保留幾位有效數字，如果預設該參數，則默認保留 6 位有效數字 ■ 當 v 是 int 類型時，可以指定第二個參數 number_base 設置顯示數制，如果預設該參數，則默認顯示十進制
precision	類型：int
	精度，保留的有效數字位數。
number_base	類型：enum 枚舉類型
	進制轉換類型，取值如下： ■ hex：十六進制 ■ dec：十進制

返回值

類型：string

返迴轉換後的字元串。

用法舉例

```
double value =3.1415
string s = tostr(value)
print s //輸出“3.1415”
```

5.8 機器人位姿函數

5.8.1 cjoint(獲取當前軸位置)

函數原型

```
joint cjoint()
```

描述

獲取當前軸位置，包括外軸。後臺不能使用。

參數

無

返回值

類型：joint

返回當前軸位置。

用法舉例

```
joint a = {j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}
movej a
waittime 0
print cjoint() //輸出“{ 0, 10, 20, 30, 40, 50, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”
//實際列印結果可能因誤差原因與設定值略有偏差（位置精度範圍內）。
```

5.8.2 cpose(獲取當前 TCP 位姿)

函數原型

```
pose cpose(tool t,wobj w)
```

描述

獲取當前 TCP 位姿以及外軸位置。

參數

表 5-83 cpose 函數的參數

名稱	說明
t	類型：tool
	當前工具
w	類型：wobj
	參考的工件坐標系

返回值

類型：pose

返回當前 TCP 位姿以及外軸位置。

用法舉例

```
pose p = { x 763.869,y 207.323,z 1422.841,a 129.538,b 0.480,c 92.084,cfg 0}
ptp p
waittime 0
wobj wobj1 = {{10,0,0,0,0,0}}
print cpose($FLANGE,wobj1) //輸出“{753.869,207.323,1422.841,129.538,0.480,92.084,0,-1,9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”
```

5.8.3 getpose(獲取某組軸位置對應的 TCP 位姿，即運動學正解)

函數原型

pose getpose(joint j,tool t,wobj w)

描述

獲取軸位置 j 對應的 TCP 位姿以及外軸位置。

參數

表 5-84 getpose 函數的參數

名稱	說明
j	類型：joint
	機器人及外軸軸位置
t	類型：tool
	指定的工具
w	類型：wobj
	參考的工件坐標系

返回值

類型：pose

返回對應的 TCP 位姿以及外軸位置。

用法舉例

```
joint j = {j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}
wobj wobj1 = {{10,0,0,0,0,0}}
print getpose(j,$FLANGE,wobj1)
```

5.8.4 getjoint(獲取某個 TCP 位姿對應的機器人軸位置，即運動學逆解)

函數原型

joint getjoint(pose p,tool t,wobj w)

描述

獲取位姿 p 對應的機器人各軸位置。

參數

表 5-85 getjoint 函數的參數

名稱	說明
p	類型：pose
	機器人位姿，參考

名稱	說明
t	類型：tool
	指定的工具，參考
w	類型：wobj
	參考的工件坐標系，參考

返回值

類型：joint

返回對應的機器人軸位置數據。

用法舉例

```
pose p = {x 763.869,y 207.323,z 1422.841,a 129.538,b 0.480,c 92.084,cfg 0}
print getjoint(p,$FLANGE,$WORLD)
```

5.8.5 poseinv(計算一個 pose 的逆 pose)

函數原型

```
pose poseinv(pose p)
```

描述

計算一個 pose 的逆 pose。

參數

表 5-86 poseinv 函數的參數

名稱	說明
p	類型：pose
	輸入一個坐標系 A 在另一個坐標系 B 中的位姿

返回值

類型：pose

返回坐標系 B 在坐標系 A 中的位姿。

用法舉例

```
pose p = {x 1,y 2,z 3,a 0,b 0,c 0}
print poseinv(p)
//輸出“{-1,-2,-3,0,0,0,0,-1,9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”
```

5.8.6 offset(目標點相對於工件坐標系的移位函數)

函數原型

pose offset(pose p,double dx,double dy,double dz,double rz,double ry,double rx)

描述

offset 函數用於將指定的目標點相對於工件坐標系進行平移及旋轉，並獲取平移後的新的目標點。

參數

表 5-87 offset 函數的參數

名稱	說明
p	類型：pose
	當前目標點
dx	類型：double
	沿工件坐標系 x 方向的平移量，單位毫米（mm）
dy	類型：double
	沿工件坐標系 y 方向的平移量，單位毫米（mm）
dz	類型：double
	沿工件坐標系 z 方向的平移量，單位毫米（mm）
rz	類型：double
	繞工件坐標系 z 方向的旋轉量，單位度（°）
ry	類型：double
	繞工件坐標系 y 方向的旋轉量，單位度（°）
rx	類型：double
	繞工件坐標系 x 方向的旋轉量，單位度（°）

注意

如果同時指定了 rz、ry、rx 中的多個值，則採用的是 ZYX 型歐拉角進行旋轉，即旋轉順序為先繞 z 軸旋轉 rz 角度，再繞新的 y 軸旋轉 ry 角度，最後繞新的 x 軸旋轉 rx 角度。

返回值

類型：pose

平移後的目標點。

用法舉例

```
pose p = {x 1500,y 500,z 500,a 0,b 0,c 0}
double dx = 10
double dy = 20
double dz = 30
```

```
double rz = 60
double ry = 50
double rx = 40
ptp p:p,vp:5%,sp:5%
waittime 0
print cjoint() //輸出“{1510, 520, 530, 40, 50, 60, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”，實際列印結果可能因誤差原因與設定值略有偏差（位置精度範圍內）。
```

5.8.7 reltool(目標點相對於工具坐標系的移位函數)

函數原型

```
pose reltool(pose p,double dx,double dy,double dz,double rz,double ry,double rx)
```

描述

reltool 函數用於將指定的目標點相對於工具坐標系進行平移及旋轉，並獲取移位後的新的目標點。

參數

表 5-88 reltool 函數的參數

名稱	說明
p	類型：pose
	當前目標點
dx	類型：double
	沿工具坐標系 x 方向的平移量，單位毫米（mm）
dy	類型：double
	沿工具坐標系 y 方向的平移量，單位毫米（mm）
dz	類型：double
	沿工具坐標系 z 方向的平移量，單位毫米（mm）
rz	類型：double
	繞工具坐標系 z 方向的旋轉量，單位度（°）
ry	類型：double
	繞工具坐標系 y 方向的旋轉量，單位度（°）
rx	類型：double
	繞工具坐標系 x 方向的旋轉量，單位度（°）


注意

這裡採用的是 ZYX 型歐拉角進行旋轉，即旋轉順序為先繞 z 軸旋轉 rz 角度，再繞新的 y 軸旋轉 ry 角度，最後繞新的 x 軸旋轉 rx 角度。

返回值

類型：pose

移位後的目標點。

用法舉例

```
pose p1= {x 1500,y 500,z 500,a 0,b 0,c 0}
double dx = 10
double dy = 20
double dz = 30
double rz = 40
double ry = 50
double rx = 60
pose p2 = reltool(p1, dx, dy, dz, rz, ry, rx)
ptp p:p2,vp:5%,sp:5%
waittime 0
print cpose($FLANGE,WORLD) //輸出“{1510,520,530,40,50,60,0,-1,9e+09,9e+09, 9e+09, 9e+09, 9e+09, 9e+09}，實際列印結果可能因誤差原因與設定值略有偏差（位置精度範圍內）。
```

5.8.8 cjttq(獲取機器人各個軸的輸出力矩)

描述

jttq 結構體類型用於直接描述機器人各軸的輸出力矩。該函數用於獲取指定前臺通道當前各個軸的輸出力矩（單位 N*m）。在前臺通道、後臺通道均可使用。

格式

```
jttq cjttq(int chan_no, bool Is_cmd)
```

定義

```
struct jttq
{
double jt1
double jt2
double jt3
double jt4
double jt5
double jt6
double et1
double et2
double et3
```

double et4

double et5

double et6

}

成員

cjttq 指令的成員詳見表 5-89。

表 5-89 cjttq 指令的成員

名稱	說明
j1~j6	類型：double
	機器人 1 軸~6 軸的輸出力矩，單位 N*M（牛米）
et1~et6	類型：double
	外軸 1 軸～6 軸的輸出力矩，單位 N*M（牛米）

參數

cjttq 指令的參數詳見表 5-90。

表 5-90 cjttq 指令的參數

名稱	說明
chan_no	類型：int
	指定前臺通道號
Is_cmd	類型：bool
	true—輸出指令力矩，false—輸出反饋力矩

返回值

類型：jttq

返回指定前臺通道當前各個軸的輸出力矩（單位 N*m）。

用法舉例

//前臺通道 1 執行程序

pose p = {x 763.869,y 207.323,z 1422.841,a 129.538,b 0.480,c 92.084,cfg 0}

ptp p

print cjttq(1,true) //輸出“{*,*,*,*,*,*,*,*,*,*,*,*,*}”，*表示力矩是未知狀態，反映當下時刻各個軸的力矩數值。

5.8.9 cjtci(獲取機器人各個軸的電機電流)

描述

jtci 結構體類型用於直接描述機器人各軸的電機電流 Iq。該函數用於獲取指定前臺通道當前各個軸的電機電流（單位 A 安培）。在在前臺通道、後臺通道均可使用。

格式

jtci cjtci(int chan_no, bool Is_cmd)

定義

struct jtci

{

double ji1

double ji2

double ji3

double ji4

double ji5

double ji6

double ei1

double ei2

double ei3

double ei4

double ei5

double ei6

}

成員

cjtci 指令的成員詳見表 5-91。

表 5-91 cjtci 指令的參數

名稱	說明
ji1~ji6	類型：double
	機器人 1 軸~6 軸的電機電流 Iq，單位 A 安培
ei1~ei6	類型：double
	外軸 1 軸～6 軸的電機電流 Iq，單位 A 安培

參數

cjtci 指令的參數詳見表 5-92。

表 5-92 cjtci 指令的參數

名稱	說明
chan_no	類型：int
	指定前臺通道號
Is_cmd	類型：bool
	是否為指令電流：true—輸出指令電流，false—輸出反饋電流

返回值

類型：jtci

返回指定前臺通道當前各個軸的電機電流（單位 A 安培）。

用法舉例

//前臺通道 1 執行程序

pose p = {x 763.869,y 207.323,z 1422.841,a 129.538,b 0.480,c 92.084,cfg 0}

ptp p

print cjtci(1,true) //輸出“{*,*,*,*,*,*,*,*,*,*,*,*,*}”，*表示電流是未知狀態，反映當下時刻各個軸的電機電流。

5.8.10 channeltojoint(獲取某通道的機器人運行目標點的軸位置)

描述

獲取指定前臺通道運行目標點的軸位置，包括外軸，在前臺通道、後臺通道均可使用。

格式

joint channeltojoint(int chan_no)

參數

channeltojoint 指令的參數詳見表 5-93。

表 5-93 channeltojoint 指令的參數

名稱	說明
chan_no	類型：const int
	指定前臺通道號

返回值

類型：joint

返回指定通道運行目標點各個軸的位置。

用法舉例

```
//前臺通道 1 執行程序
joint a = {j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}
movej a
print channeltojoint(1) //輸出“{ 0, 10, 20, 30, 40, 50, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”
```

5.8.11 channeltopose(獲取某通道的機器人運行目標點 TCP 位姿)

描述

獲取指定前臺通道運行目標點 TCP 位姿以及外軸位置。再在前臺通道、後臺通道均可使用。

格式

```
pose channeltopose(int chan_no,tool t,wobj w)
```

參數

channeltopose 指令的參數詳見表 5-94。

表 5-94 channeltopose 指令的參數

名稱	說明
chan_no	類型：int
	指定前臺通道號
t	類型：tool
	當前工具
w	類型：wobj
	參考的工件坐標系

返回值

類型：pose

返回指定前臺通道當前 TCP 位姿以及外軸位置。

用法舉例

```
//前臺通道 1 執行程序
pose p = {x 753.869,y 207.323,z 1422.841,a 129.538,b 0.480,c 92.084,cfg 0}
ptp p
wobj wobj1 = {{10,0,0,0,0,0}}
print channeltopose(1,$FLANGE,wobj1) //輸出“{753.869,207.323, 1422.841,129.538,0.480,92.084,0,-
1,9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”
```

5.8.12 channeljoint(獲取指定前臺通道當前軸位置)

描述

獲取指定前臺通道當前軸位置，包括外軸，在前臺通道、後臺通道均可使用。

格式

joint channeljoint(int chan_no,bool Is_cmd)

參數

channeljoint 指令的參數詳見表 5-95。

表 5-95 channeljoint 指令的參數

名稱	說明
chan_no	類型：int
	指定前臺通道號
Is_cmd	類型：bool
	是否為指令位置： ■ true—輸出指令位置，即當本體處於運行狀態時，則反回的指令位置為當前軌跡的目標點。 ■ false—輸出反饋位置，即反饋本體當前的點位。

返回值

類型：joint

返回指定前臺通道當前軸位置。

用法舉例

```
//前臺通道 1 執行程序
joint a = {j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}
movej a
waittime 0
//後臺通道執行程序
print channeljoint(1,true) //輸出“{0, 10, 20, 30, 40, 50, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”
```

5.8.13 channelpose(獲取指定前臺通道當前 TCP 位姿)

描述

獲取指定前臺通道當前 TCP 位姿以及外軸位置，在前臺通道、後臺通道均可使用。

格式

pose channelpose(int chan_no,tool t,wobj w,bool Is_cmd)

參數

channelpose 指令的參數詳見表 5-96。

表 5-96 channelpose 指令的參數

名稱	說明
chan_no	類型：int

名稱	說明
	指定前臺通道號
t	類型：tool
	當前工具
w	類型：wobj
	參考的工件坐標系
Is_cmd	類型：bool
	是否為指令位置： ■ true—輸出指令位置，即當本體處於運行狀態時，則反回的指令位置為當前軌跡的目標點。 ■ false—輸出反饋位置，即反饋本體當前的點位。

返回值

類型：pose

返回指定前臺通道當前 TCP 位姿以及外軸位置。

用法舉例

```
//前臺通道 1 執行程序
pose p = {x 753.869,y 207.323,z 1422.841,a 129.538,b 0.480,c 92.084,cfg 0}
ptp p
waittime 0
wobj wobj1 = {{10,0,0,0,0,0}}
//後臺通道執行程序
print channelpose(1,$FLANGE,wobj1,true) //輸出“{753.869,207.323, 1422.841,129.538,0.480,92.084,0,-1,9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”
```

5.8.14 channeljointvel(獲取機器人各個軸的速度)

描述

獲取指定前臺通道當前各個軸的速度（單位 rad/s），包括外軸。在前臺通道、後臺通道均可使用。

格式

```
jvel channeljointvel(int chan_no,bool Is_cmd)
```

參數

channeljointvel 指令的參數詳見表 5-97。

表 5-97 channeljointvel 指令的參數

名稱	說明
chan_no	類型：const int
	指定前臺通道號

名稱	說明
Is_cmd	類型：bool
	是否為指令速度： ■ true—輸出指令位置，即當本體處於運行狀態時，則反回的指令位置為當前軌跡的目標點。 ■ false—輸出反饋位置，即反饋本體當前的點位。

返回值

類型：jvel

返回指定前臺通道當前各個軸的速度（單位 rad/s），jvel 結構體定義請參考。

用法舉例

//前臺通道 1 執行程序

joint a = {j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}

movej a

print channeljointvel(1,$FLANGE,wobj1,true) //輸出“{ *,*, *, *, *, *, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”，*表示列印當下的軸速度是未知狀態，*的具體數值反映當下時刻速度。

5.8.15 channeltcpvel(獲取機器人 TCP 點速度)

描述

獲取指定前臺通道當前 TCP 點的速度（單位 mm/s）。在前臺通道、後臺通道均可使用。

格式

double channeltcpvel(int chan_no, tool t,wobj w,bool Is_cmd)

參數

channeltcpvel 指令的參數詳見表 5-98。

表 5-98 channeltcpvel 指令的參數

名稱	說明
chan_no	類型：const int
	指定前臺通道號
t	類型：tool
	當前使用的工具
w	類型：wobj
	當前參考的工件坐標系
Is_cmd	類型：bool
	是否為指令速度：true—輸出指令位置，即當本體處於運行狀態時，則反回的指令位置為當前軌跡的目標點。false—輸出反饋位置，即反饋本體當前的點位。

返回值

類型：double

返回指定前臺通道當前 TCP 點的速度（單位 mm/s）。

用法舉例

//前臺通道 1 執行程序

joint a = {j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}

movej a

print channeltcpvel(1,$FLANGE,wobj1,true) //輸出“*”，*表示列印當下的 TCP 速度是未知狀態，*的具體數值反映當下時刻速度。

5.8.16 ctcpforce(獲取機器人 TCP 點六維力矢量)

描述

tcpforce 結構體類型使用笛卡爾坐標的方式來描述機器人 TCP 點輸出力和力矩。該函數用於獲取指定前臺通道當前 TCP 點輸出力（單位 N）和輸出力矩（單位 N*m）。在前臺通道使用。

格式

tcpforce ctcpforce(int chan_no, tool t,wobj w)

定義

struct tcpforce

{

double fx

double fy

double fz

double tx

double ty

double tz

}

成員

ctcpforce 指令的成員詳見表 5-99。

表 5-99 ctcpforce 指令的參數

名稱	說明
fx	類型：double
	機器人 TCP 點輸出力相對當前工件坐標系坐標的 x 分量，單位 N(牛)

名稱	說明
fy	類型：double
	器人 TCP 點輸出力相對當前工件坐標系坐標的 y 分量，單位 N(牛)
fz	類型：double
	器人 TCP 點輸出力相對當前工件坐標系坐標的 z 分量，單位 N(牛)
tx	類型：double
	機器人工具輸出力矩在當前工件坐標系的 x 分量，單位 N*m(牛米)
ty	類型：double
	機器人工具輸出力矩在當前工件坐標系的 y 分量，單位 N*m(牛米)
tz	類型：double
	機器人工具輸出力矩在當前工件坐標系的 z 分量，單位 N*m(牛米)

參數

ctcpforce 指令的參數詳見表 5-100。

表 5-100 ctcpforce 指令的參數

名稱	說明
chan_no	類型：int
	指定前臺通道號
t	類型：tool
	當前使用的工具
w	類型：wobj
	當前參考的工件坐標系

返回值

類型：tcpforce

返回指定前臺通道當前 TCP 點輸出力（單位 N）和輸出力矩（單位 N*m）。

用法舉例

```
//前臺通道 1 執行程序
pose p = {x 763.869,y 207.323,z 1422.841,a 129.538,b 0.480,c 92.084,cfg 0}
ptp p
wobj wobj1 = {{10,0,0,0,0,0}}
print ctcpforce(1,$FLANGE,wobj1) //輸出“{*,*,*,*,*,*}”，*表示力和力矩是未知狀態，表示當下時刻 TCP 點的力和力矩數值。
```

5.9 坐標系標定函數

5.9.1 getwobj_3p(3 點法標定工件坐標系)

函數原型

wobj getwobj_3p(joint j1, joint j2, joint j3,tool t)

描述

通過 3 個示教點，標定工件坐標系。

參數

表 5-101 getwobj_3p 函數的參數

名稱	說明
j1	類型：joint
	待標定的工件坐標系的原點對應的示教點
j2	類型：joint
	待標定的工件坐標系的 X 軸正向一點對應的示教點
j3	類型：joint
	待標定的工件坐標系的 XY 平面上 Y 軸為正的一點對應的示教點
t	類型：tool
	標定時使用的工具

返回值

類型：wobj

返回計算出的工件坐標系標定結果。

用法舉例

```
joint j1 = {j1 1,j2 0.005,j3 120.001,j4 0.001,j5 30.015, j6 -0.139}
joint j2 = {j1 10,j2 0.005,j3 120.001,j4 0.001,j5 30.015, j6 -0.139}
joint j3 = {j1 15,j2 0.005,j3 120.001,j4 0.001,j5 30.015, j6 -0.139}
print getwobj_3p(j1,j2,j3,$TOOL0)
```

5.9.2 getwobj_indi(間接法標定工件坐標系)

函數原型

wobj getwobj_indi(joint j1, joint j2, joint j3,pos p1,pos p2,pos p3,tool t)

描述

通過已知在待標定工件坐標系中坐標的 3 個測試點，間接標定工件坐標系。

參數

表 5-102 getwobj_indi 函數的參數

名稱	說明
j1	類型：joint
	p1 點對應的示教點
j2	類型：joint
	p1 點對應的示教點
j3	類型：joint
	p3 點對應的示教點
p1	類型：pos
	p1 點在待標定工件坐標系中的坐標
p2	類型：joint
	p2 點在待標定工件坐標系中的坐標
p3	類型：joint
	p3 點在待標定工件坐標系中的坐標
t	類型：tool
	標定時使用的工具

返回值

類型：wobj

返回計算出的工件坐標系標定結果。

用法舉例

```
joint j1 = j:{j1 1,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
joint j2 = j:{j1 10,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
joint j3 = j:{j1 15,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
pos p1 = {10,20,20}
pos p2 = {40,60,20}
pos p3 = {50,20,40}
print getwobj_indi(j1,j2,j3,p1,p2,p3, $TOOL0)
```

5.9.3 getwobj_flange(法蘭參照法標定工件坐標系)

函數原型

```
wobj getwobj_flange(joint j1, joint j2, tool t,bool xydefault)
```

描述

該函數通過一個標準工具標定工件坐標系或外部固定工具坐標系原點，通過機器人法蘭坐標系參照標定工件坐標系或外部固定工具坐標系方向。該標定方法主要用於外部工具坐標系的標定。

標定方法：

- 在機器人法蘭上安裝好標準工具。
- 用標準工具的 TCP 點駛向待測坐標系的原點，記下此時各軸位置角度。
- 將法蘭坐標系的 Z-方向與待測坐標系的 Z+方向移至平行。
- 如果希望標定待測坐標系的 X，Y 方向，則還需要將法蘭坐標系的 X+方向與待測坐標系的 X+方向移至平行。如果不關心待測坐標系的 X，Y 方向，則這步可以省略。
- 記下此時各軸位置角度。
- 執行該函數計算得到待測工件坐標系。

參數

表 5-103 getwobj_flange 函數的參數

名稱	說明
j1	類型：joint
	標定原點時記錄的軸位置
j2	類型：joint
	標定方向時記錄的軸位置
t	類型：tool
	標定原點時法蘭端安裝的標準工具，這裡只用到工具中的 xyz 分量，abc 分量忽略
xydefault	類型：bool
	是否默認 xy 軸方向 ■ true：只標定 z 軸方向，xy 軸方向由系統默認 ■ false：xyz 軸方向均標定

返回值

類型：wobj

返回計算出的工件坐標系標定結果。

用法舉例

```
tool knowntool = {{0,0,0,0,0,0}}
joint j1 = { j1 4.229 ,j2 42.310 ,j3 84.665, j4 88.315, j5 84.614 ,j6 -36.429}
joint j2 = { j1 3.456, j2 41.824, j3 86.464 ,j4 92.073, j5 84.982, j6 -36.506}
wobj w = getwobj_flange(j1,j2,knowntool,true)
print w
$TOOLS[0].t_frame = w.w_frame
$TOOLS[0].stationary = true
```

5.9.4 gettooltcp_ref(標準工具參照法標定工具坐標系 xyz)

函數原型

void gettooltcp_ref(joint j1, joint j2, tool ref,tool& t)

描述

通過標準工具標定工具坐標系 x，y，z。

標定方法：

- 選取一個參考點和一個已知 TCP 坐標的參考工具。
- 將該參考工具安裝於機器人法蘭，手動控制機器人靠近參考點，記錄下該示教點。
- 將參考工具卸下，將待測工具安裝於機器人法蘭，手動控制機器人靠近參考點，記錄下該示教點。

參數

表 5-104 gettooltcp_ref 函數的參數

名稱	說明
j1	類型：joint
	安裝參考工具時對應的示教點
j2	類型：joint
	安裝待測工具時對應的示教點
tool	類型：ref
	參考工具，只需知道 x，y，z 值
tool	類型：tool
	待標定的工具坐標系，標定結果將會保存到該變量的 x，y，z 中

返回值

無

用法舉例

```
joint j1 = j:{j1 1,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
joint j2 = j:{j1 10,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
tool ref = {{x 10,y 10,z 50}}
tool t
gettooltcp_ref(j1,j2,ref,t)
print t
```

5.9.5 gettoolrot_world(世界坐標系參照法測量工具坐標系 abc)

函數原型

void gettoolrot_world(joint j,tool& t)

描述

通過將工具坐標系的軸調整為與世界坐標系的軸平行，由此計算工具坐標系的 abc。

標定方法：

- 將與調成平行，與平行，與平行，記下此時的示教點。

參數

表 5-105 gettoolrot_world 函數的參數

名稱	說明
j	類型：joint
	記錄的示教點
tool	類型：tool
	待標定的工具坐標系，標定結果將會保存到該變量的 a，b，c 中

返回值

無

用法舉例

```
joint j = j:{j1 1,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
tool t
gettoolrot_world (j,t)
print t
```

5.9.6 gettoolrot_3p(3 點法測量工具坐標系 abc)

函數原型

```
void gettoolrot_3p(joint j1,joint j2,joint j3,tool& t)
```

描述

通過 3 點法標定工具坐標系 abc。

標定方法：

- 用工具 TCP 駛向任何一個參考點，記下此時示教點。
- 用工具坐標系 X 軸負向的一個點駛向參考點，記下此時示教點。
- 用工具坐標系 XY 平面上的 Y 軸為負值的一點駛向參考點，記下此時示教點。

參數

表 5-106 gettoolrot_3p 函數的參數

名稱	說明
j1	類型：joint
	第 1 步獲取的示教點

名稱	說明
j2	類型：joint
	第 2 步獲取的示教點
j3	類型：joint
	第 3 步獲取的示教點
tool	類型：tool
	待標定的工具坐標系，t_frame 分量的 x，y，z 分量需傳入正確的值，標定結果將會保存到該變量的 t_frame 分量的 a，b，c 分量中

返回值

無

用法舉例

```
joint j1 = j:{j1 1,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
joint j2 = j:{j1 10,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
joint j3 = j:{j1 15,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
tool t
gettoolrot_3p (j1,j2,j3,t)
print t
```

5.9.7 gettool_3p(3 點法標定工具坐標系)

函數原型

```
tool gettool_3p(joint j1, joint j2, joint j3,wobj w)
```

描述

該函數通過一個已知坐標的參考點來標定相對法蘭坐標系定義的工具坐標系或機器人抓取工件坐標系。該標定方法主要用於標定機器人抓取工件坐標系。

標定方法：

- 用坐標系原點駛向已知參考點，記下此時示教點。
- 用坐標系 X 軸正向的一個點駛向已知參考點，記下此時示教點。
- 用坐標系 XY 平面上的 Y 軸為正值的一點駛向已知參考點，記下此時示教點。

參數

表 5-107 gettool_3p 函數的參數

名稱	說明
j1	類型：joint
	第 1 步獲取的示教點
j2	類型：joint

名稱	說明
	第 2 步獲取的示教點
j3	類型：joint
	第 3 步獲取的示教點
w	類型：wobj
	已知在世界坐標系中坐標值的參考點，這裡只用到工件坐標系中的 xyz 分量，abc 分量忽略

返回值

類型：tool

返回計算出的工具坐標系標定結果。

用法舉例

```
wobj w = {{935,0,1300,0,0,0}}
joint j1 = {0,0,90,0,90,0}
joint j2 = {0,0,90,0,-90,0}
joint j3 = {0,0,90,90,90,0}
tool t = gettool_3p(j1,j2,j3,w)
print t
$WOBJS[0].w_frame = t.t_frame
$WOBJS[0].robhold = true
```

5.9.8 getbase_3p(3 點法標定基礎坐標系)

函數原型

void getbase_3p(joint j1, joint j2, joint j3,tool t, int index)

描述

通過 3 個示教點，標定基礎坐標系。

參數

表 5-108 getbase_3p 函數的參數

名稱	說明
j1	類型：joint
	世界坐標系原點對應的示教點
j2	類型：joint
	世界坐標系 X 軸正向一點對應的示教點
j3	類型：joint
	世界坐標系 XY 平面上 Y 軸為正的一點對應的示教點
t	類型：tool

名稱	說明
	標定時使用的工具
index	類型：int
	機械單元序號，參數範圍：[1,n]。其中 n 為當前通道的機械單元總數

返回值

無

用法舉例

```
joint j1 = j:{j1 1,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
joint j2 = j:{j1 10,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
joint j3 = j:{j1 15,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
getbase_3p(j1,j2,j3, $TOOL0,1)
print $BASE
```

5.10 軌跡觸發相關函數

5.10.1 T(當前運動軌跡是否到達某個時間點)

函數原型

```
bool T(double t)
```

描述

返回當前運動軌跡距離起點是否已經走了 t 秒。該函數主要用於軌跡觸發中的事件定義。

參數

表 5-109 T 函數的參數

名稱	說明
t	類型：double
	單位秒

返回值

類型：bool

- false：當前運動軌跡尚未到達 t 時間點。
- true：當前運動軌跡已經到達 t 時間點。

用法舉例

```
trigger 1,when:T(1),do:setdo(1,true) //在下麵的軌跡上聲明一個軌跡觸發：當距離起點時間 1 秒時第 1 路 DO 輸出信號 true
movej j:{j1 30},v:{per 2}
```

5.10.2 S(當前運動軌跡是否到達某個路程點)

函數原型

bool S(double s)

描述

返回當前運動軌跡距離起點是否已經走了 s 毫米。該函數主要用於軌跡觸發中的事件定義。

參數

表 5-110 S 函數的參數

名稱	說明
s	類型：double
	單位毫米

返回值

類型：bool

- false：當前運動軌跡尚未到達距離起點 s 的位置點。
- true：當前運動軌跡已經到達距離起點 s 的位置點。

用法舉例

trigger 1,when:S(100),do:setdo(1,true) //在下麵的軌跡上聲明一個軌跡觸發：當距離起點時間 100 毫米時第 1 路 DO 輸出信號 true

lin p1,v:{tcp 20,ori 10}

5.10.3 StoEnd(當前運動軌跡是否到達距離目標點某個距離的位置點)

函數原型

bool StoEnd(double s)

描述

返回當前運動軌跡距離目標點是否還剩 s 毫米。該函數主要用於軌跡觸發中的事件定義。

參數

表 5-111 StoEnd 函數的參數

名稱	說明
s	類型：double
	單位毫米

返回值

類型：bool

- false：當前運動軌跡尚未到達距離目標點還剩 s 毫米的點。
- true：當前運動軌跡已經到達距離目標點還剩 s 毫米的點。

用法舉例

trigger 1,when:StoEnd(100),do:setdo(1,true) //在下麵的軌跡上聲明一個軌跡觸發：當距離目標點 100 毫米時第 1 路 DO 輸出信號 true

lin p1,v:{tcp 20,ori 10}

5.11 點位配方修改的相關 ARL 函數

5.11.1 savearl(複製 arl 文件)

函數原型

bool savearl(const string& file_src, const string& file_dst, bool mode)

描述

複製當前文件夾下指定名稱的 arl 文件，並對新 arl 文件進行命名

參數

表 5-112 savearl 函數的參數

名稱	說明
file_src	類型：string
	原 arl 文件名稱
file_dst	類型：string
	複製後的 arl 文件名稱
mode	類型：bool
	是否覆蓋同名文件。true：覆蓋同名文件；false：不覆蓋同名文件

返回值

類型：bool

複製成功則返回 true，複製失敗則返回 false。

用法舉例

print savearl("Recipe1.arl",Recipe2.arl",true) //當前文件夾下已有 Recipe2.arl 文件，則輸出 true，並覆蓋 Recipe2.arl 文件

print savearl("Recipe1.arl","Recipe2.arl",false) //當前文件夾下已有 Recipe2.arl 文件，則輸出 false，不做任何操作

5.11.2 savefilepose(將點位信息保存至 data 文件)

函數原型

bool savefilepose(const string& file_name,const string& pose_name,const pose& p)

描述

將 pose 點位信息保存至指定 arl 文件的指定點位中（重新加載後生效）

參數

表 5-113 savefilepose 函數的參數

名稱	說明
file_name	類型：string
	存入點位的_data.arl 文件名稱
p_name	類型：string
	存入點位信息的 pose 變量名稱
p	類型：pose
	待存入的 pose 變量

返回值

類型：bool

存入成功則返回 true，存入失敗則返回 false。

用法舉例

pose a = {x 0,y 10,z 15,a 0,b 90,c 0,cfg 0,ej1 20}

savefilepose("Recipe1_data.arl", p1, a) //將{x 0,y 10,z 15,a 0,b 90,c 0,cfg 0,ej1 20}保存至 Recipe1_data.arl 的 p1 中。

5.11.3 savefilejoint(將點位信息保存至 data 文件)

函數原型

bool savefilejoint(const string& file_name,const string& pose_name,const joint& j)

描述

將 joint 點位信息保存至指定 arl 文件的指定點位中（重新加載後生效）

參數

表 5-114 savefilejoint 函數的參數

名稱	說明
file_name	類型：string
	存入點位的_data.arl 文件名稱
joint_name	類型：string
	存入點位信息的 joint 變量名稱
j	類型：joint
	待存入的 joint 變量

返回值

類型：bool

存入成功則返回 true，存入失敗則返回 false。

用法舉例

joint a = {j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}

savefilejoint("Recipe1_data.arl", j1, a) //將{j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}保存至 Recipe1_data.arl 中的 j1 點位。

5.11.4 saveposenow(將點位信息保存至某通道的程序)

函數原型

bool saveposenow(unsigned int channel, const string& file_name, const string& pose_name, const pose& p)

描述

將 pose 類型點位信息保存至指定前臺通道下指定程序的指定點位中（立即生效）。

參數

表 5-115 saveposenow 函數的參數

名稱	說明
channel	類型：unsigned int
	存入點位的通道號
file_name	類型：string
	存入點位的 arl 程序名
pose_name	類型：string
	存入點位信息的 pose 變量名稱
p	類型：pose
	待存入的 pose 變量

返回值

類型：bool

存入成功則返回 true，存入失敗則返回 false。

用法舉例

pose a = {x 0,y 10,z 15,a 0,b 90,c 0,cfg 0,ej1 20}

saveposenow(1,"prog1.arl","p1", a) //將{x 0,y 10,z 15,a 0,b 90,c 0,cfg 0,ej1 20}保存至前臺通道 1 下 prog1.arl 的 p1 中。

5.11.5 savejointnow(將點位信息保存至某通道的程序)

函數原型

bool savejointnow(unsigned int channel,const string& file_name,const string& pose_name,const joint& j)

描述

將 joint 類型點位信息保存至指定前臺通道下指定程序的指定點位中（立即生效）。

參數

表 5-116 savejointnow 函數的參數

名稱	說明
channel	類型：unsigned int
	存入點位的通道號
file_name	類型：string
	存入點位的 arl 程序名
joint_name	類型：string
	存入點位信息的 joint 變量名稱
j	類型：joint
	待存入的 joint 變量

返回值

類型：bool

存入成功則返回 true，存入失敗則返回 false。

用法舉例

joint a = {j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}

savejointnow(1,"prog1.arl","j1", a) //將{j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}保存至前臺通道 1 下 prog1.arl 的 j1 中。

5.11.6 switcharl(切換前臺通道加載的 arl 程序)

函數原型

bool switcharl(unsigned int channel,const string& file_name)

描述

將指定通道的程序暫停、複位、卸載，並將指定名稱的 arl 文件加載到該通道中並運行。僅允許在後臺通道使用。

參數

表 2-6 switcharl 函數的參數

名稱	說明
channel	類型：unsigned int
	前臺通道號，範圍：[1,6]
file_name	類型：string
	待加載的 arl 程序文件名

返回值

類型：bool

切換成功則返回 true，失敗則返回 false。

用法舉例

print switcharl(1,"Recipe1.arl") //輸出 true，並將 Recipe1.arl 加載至前臺通道 1

5.12 動力學函數

5.12.1 motionsup(打開/關閉碰撞檢測)

描述

motionsup 指令用於打開/關閉碰撞檢測，並設置靈敏度。

格式

motionsup(bool ison, int tunevalue)

參數

motionsup 指令的參數詳見表 5-117。

表 5-117 motionsup 指令的參數

名稱	說明
ison	數據類型：bool
	ison 表示程序運行模式碰撞檢測是否開啟 ■ ison 為 true 時，程序運行模式碰撞檢測開啟 ■ ison 為 false 時，程序運行模式碰撞檢測關閉
tunevalue	數據類型：int
	tunevalue 表示程序運行模式碰撞檢測靈敏度，值越小越靈敏，設置範圍為 1%~300%。 ■ ison 為 true，tunevalue 不輸入時使用參數配置中的靈敏度 ■ ison 為 false，不允許輸入 tunevalue

用法舉例

motionsup(true, 200) //打開碰撞檢測，靈敏度為 200%

碰撞檢測是在機器人運行時，對機器人發生碰撞的事件進行檢測並立即做出停機等安全響應的功能。（碰撞檢測功能並不能完全避免設備的損傷，例如，如果機器人發生碰撞時正處於全速運行，則通常損傷無法避免）。

5.13 其他函數

5.13.1 typeof(獲取參數類型名)

函數原型

string typeof(anytype v)

描述

返回任意數據類型的類型名。

參數

表 5-118 typeof 函數的參數

名稱	說明
v	類型：可以是任意數據類型的表達式
	待獲取類型的表達式

返回值

類型：string

返回該類型的類型名字元串。

用法舉例

double value =3.1415

print typeof(value) //輸出“double”

5.13.2 ctime(獲取當前時間字元串)

函數原型

string ctime()

描述

以字元串的形式表示當前的時間。

參數

無

返回值

類型：string

返回當前時間，用“時：分：秒”的字元串形式表示。

用法舉例

print ctime() //假設當前時間是 10 點 10 分 10 秒，則輸出“10:10:10”

5.13.3 cdate(獲取當前日期字元串)

函數原型

string cdate()

描述

以字元串的形式表示當前日期。

參數

無

返回值

類型：string

返回當前日期，用“年-月-日”的字元串形式表示。

用法舉例

print cdate() //假設當前日期是 2014 年 10 月 25 日，則輸出“2014-11-25”

5.13.4 assert(斷言)

函數原型

void assert(bool x)

描述

判斷 x 的值是否為真，如果不為真，則報出警告，並說明 assert 函數所在的文件和行號。

參數

表 5-119 assert 函數的參數

名稱	說明
x	類型：bool
	被斷言的 bool 型表達式

返回值

無

用法舉例

int a = 6

assert(a == 5) //程序執行到這一行將退出，並報出錯誤

5.13.5 savesv(存儲系統變量)

函數原型

void savesv(string svname)

描述

存儲變量名為 svname 的系統變量。

參數

表 5-120 savesv 函數的參數

名稱	說明
svname	類型：string
	預保存的系統變量的名字。如果輸入的變量名不存在，則會產生一個運行時錯誤

返回值

類型：void

用法舉例

savesv("TOOLS") //保存工具系統變量，這樣下次再啟動系統時 TOOLS 的值不會丟失。

5.13.6 init(恢復系統變量為默認值)

函數原型

void init()

描述

恢復中說明的系統變量為默認值。

參數

無

返回值

無

用法舉例

$DFSPPED = {10,50,5,5,5}

init()

print $DFSPPED //這裡輸出“{5,50,5,5,5}”

5.13.7 gettextstr(讀取文本文件中的某一行內容)

函數原型

gettextstr(string file_path,int line_no,bool external_file)

描述

用於讀取文本文件中的某一行內容。

參數

表 5-121 gettextstr 函數的參數

名稱	說明
file_path	類型：string
	文件名，例如 script 目錄下的文件 1.txt，則輸入文件名為 1.txt，如果 script 目錄下的文件 subdir/2.txt，則輸入文件名為 subdir/2.txt
line_no	類型：int
	行號，從 1 開始，當輸入文件名不存在時，會報警 10064，文件不存在或損壞，當輸入行號大於文件內容最大行號時，報警 10065，輸入行號大於文件最大行號
external_file	類型：bool
	文件存儲位置類型，預設默認值為 false。true 代表外部存儲文件，讀取/home/USB 目錄下的文件；false 代表內部存儲文件，讀取/home/ae/script 目錄下的文件

返回值

類型：string

用法舉例

gettextstr("test1.txt",5, false) //讀取路徑/home/ae/script 目錄下 test1.txt 第 5 行的內容；

gettextstr("test/test1.txt",5, true) //讀取路徑/home/ae/USB/test 目錄下 test1.txt 第 5 行的內容；

6 中斷

當在複雜的應用中使用機器人時，需要機器人可以立即對某些外部或內部事件作出反應，必須中斷正在運行的機器人程序，啟動中斷處理函數。中斷處理函數執行完成後，再回到被中斷的程序中繼續執行。

6.1 中斷聲明

格式

interrupt [name:],[priority:],when:,do:

參數

表 6-1 interrupt 的參數說明

名稱	說明
name	數據類型：string
	指定中斷名，該名字如果指定則不允許為空，並且不能和其他聲明過的中斷重名 之後用戶可以通過該中斷名使能或者屏蔽該中斷 該參數可以預設，預設值為空字元串
priority	數據類型：int
	指定中斷優先級 中斷優先級必須為範圍在 0~255 的整數 該參數可預設，預設值為 0 關於中斷優先級更詳細的信息見 6.2 章节
when	數據類型：bool
	定義中斷事件 該中斷事件為一個 bool 型表達式，只要該 bool 型表達式的值為 true，則中斷事件就執行，表示一個中斷事件發生 關於中斷事件更詳細的信息見 6.3 章节
do	數據類型：any
	定義中斷處理動作 當中斷事件發生時，則會執行該表達式動作 關於中斷處理動作更詳細的信息見 6.4 章节

6.2 中斷優先級

中斷優先級為一個 0~255 的整數，0 的優先級最高，255 優先級最低。

中斷優先級表示當在同一個中斷掃描周期同時有 2 個事件發生時，則優先響應優先級高的那一個，也就是說先執行高優先級中斷的中斷處理動作，執行完畢後，再執行低優先級中斷的中斷處理動作，最後再回到被中斷函數繼續執行。如果 2 個事件的優先級相同，則按聲明的先後順序依次響應。

相關示例程序如下：

```
func void inthandler1()
```

```
print "-->inthandler1"
endfunc
func void inthandler2()
print "-->inthandler2"
endfunc
func void main()
//聲明中斷，中斷優先級為 1
interrupt 1,when:getdi(5),do:inthandler1()
//聲明中斷，中斷優先級為 0
interrupt 0,when:getdi(6),do:inthandler2()
while(1)
waittime 1
endwhile
endfunc
```

假設某一時刻 DI 板的通道 5 與通道 6 同時有信號產生時，則系統會優先響應中斷優先級為 0 的中斷，再響應中斷優先級為 1 的中斷。

所以最終的輸出結果為：

```
-->inthandler2
-->inthandler1
```

中斷允許嵌套。在系統執行中斷處理函數的過程中，如果發生了更高優先級的中斷事件，則系統會暫停當前中斷函數的執行，進入新的中斷處理函數執行，執行完畢後，再回到上一級中斷處理函數中繼續執行。

相關示例程序如下：

```
func void inthandler1()
print "-->inthandler1"
endfunc
func void inthandler2()
print "-->inthandler2"
movej j:{j2 -20}
movej j:{j2 -30}
waittime 0
endfunc
func void main()
//聲明中斷，中斷優先級為 0
interrupt 0,when:getdi(5),do:inthandler1()
//聲明中斷，中斷優先級為 1
interrupt 1,when:getdi(6),do:inthandler2()
while(1)
waittime 1
endwhile
endfunc
```

假設在 t1 時刻 DI 的第 6 通道產生信號，則程序進入到 inthandler2 中執行，執行過程中，如果 DI 的第 5 通道又產生了信號，則程序會直接進入到 inthandler1 中執行，執行完畢後再回到 inthandler2 中接着被中斷的程序繼續執行。

6.3 中斷事件

中斷事件必須為一個 bool 型或者能隱式轉換為 bool 型的表達式。該表達式與 if，while 語句中的判斷語句相同。

舉例來說，以下表達式都屬於這種類型的表達式：

- 1
- true
- getdi(5)
- getdi(5) == true
- getdi(5) && getdi(6)
- counter>=20
- T(3.4)

有中斷事件發生的定義為：

在上一個中斷掃描周期中，中斷事件表達式的值為 false，在本中斷掃描周期中，中斷事件表達式的值為 true。也就是說 ARL 中中斷為邊沿觸發，只有當中斷事件表達式的值從 false 變為 true 時，才表示該中斷事件發生。

6.4 中斷處理動作

中斷處理動作為一個表達式。當中斷事件發生時，如果滿足執行條件，該表達式會被執行。

interrupt 聲明指令中的 do 參數表達式一般為調用一個函數，該函數既可以是用戶定義的中斷處理函數，也可以是系統預定義函數。

如果中斷處理動作比較簡單，例如只是輸出 1 路或多路 DO，則可以像如下形式聲明中斷：

interrupt 1,when:getdi(6),do:setdo(1,true) //聲明一個優先級為 1 的中斷，當 DI 的第 6 通道有信號時，則將 DO 的第一個通道信號置為 true。

如果中斷處理動作要執行比較複雜的事情，則只能定義中斷處理函數。中斷處理函數與普通函數的定義沒有區別，只是一般的中斷處理函數都沒有參數和返回值。

6.5 中斷使能，屏蔽與刪除

當一個中斷聲明過後該中斷默認就已經使能。

如果希望在某些情況下屏蔽該中斷，可以使用 disableint 指令。如果希望再次使能該中斷，可以使用 enableint 指令。如果希望刪除某個聲明的中斷，可以使用 delint 指令。

disableint，enableint 和 delint 指令用法如下：

格式

disableint/enableint/delint [name:],[priority:]

參數

disableint/enableint/delint 指令單獨使用時，即不帶任何參數時表示屏蔽/使能/刪除所有中斷。

表 6-2 disableint/enableint/delint 的參數說明

名稱	說明
name	數據類型：string
	指定要屏蔽/使能/刪除的中斷的中斷名 ■ 如果希望通過中斷名來屏蔽/使能/刪除某個中斷。該中斷在聲明時必須執行 name 參數為該中斷取個合適的名字 ■ 如果這裡指定的中斷名的中斷並不存在，則會產生一個運行時錯誤
priority	數據類型：int
	指定屏蔽/使能/刪除某個中斷優先級的中斷 ■ disableint 指令允許屏蔽/使能/刪除某個中斷優先級的中斷 ■ 如果這裡指定的中斷優先級的中斷並不存在，則會產生一個運行時錯誤

如果 name 和 priority 參數同時指定，則表示既使能名字為 name 的中斷，也使能優先級為 prioriy 的所有中斷。

中斷屏蔽與使能的用法舉例如下：

```
func void inthandler1
disalbeint//屏蔽任何中斷，即執行該中斷函數期間不響應任何其他中斷
……
enableint//使能全部中斷
endfunc
func void main
//聲明中斷，中斷優先級為 0
interrupt 0,when:getdi(5),do:inthandler1()
while(1)
waittime 1
endwhile
endfunc
```

6.6 在中斷處理函數中停止當前機器人動作

默認情況下，當發生中斷事件時，主函數中提前規劃的軌跡仍然會運行完之後才會執行中斷處理函數中的運動。

如果希望中斷事件發生時即停止當前的運動，則需要用到 stopmove 指令。

退出中斷處理函數時如果希望繼續被中斷的運動，則需要用到 startmove 指令。

stopmove 和 startmove 指令的用法舉例如下：

```
func void inthandler1()
stopmove fast //快速停止當前運動
waituntil getdi(6) //等待第 6 通道 DI 信號
startmove skip:1 //跳過停止的程序段，並重新啟動當前運動
```

```
endfunc
func void main()
//聲明中斷，中斷優先級為 0
interrupt 0,when:getdi(5),do:inthandler1()
pose p = {x 1500,y 500,z 500,a 0,b 90,c 0,cfg 0}
ptp p,v:{per 10}
while(true)
lin p:{x 1000,y 500,z 500,a 0,b 90,c 0},v:{tcp 10}
lin p:{x 1100,y 500,z 500,a 0,b 90,c 0},v:{tcp 10}
endwhile
endfunc
```

stopmove 會沿程式設計所設定的路徑減速停止，需要一定的停止時間和停止距離，速度較快時甚至可能停止到下一條軌跡上，程式設計時需要註意。

6.7 定時中斷

描述

定時中斷指令是一種特殊的中斷聲明。它以時鐘作為中斷源，可以應用於需要實現一段時間之後觸發一次中斷，或者每隔一段時間就觸發一次中斷的場合。

格式

timer [name:],[priority:],interval:,[rmode:],do:

參數

表 6-3 timer 的參數說明

名稱	說明
name	數據類型：string
	指定中斷名，該名字如果指定則不允許為空，並且不能和其他聲明過的中斷重名 之後用戶可以通過該中斷名使能或者屏蔽該中斷。該參數可以預設，預設值為空字元串
priority	數據類型：int
	指定中斷優先級。中斷優先級必須為範圍在 0~255 的整數，該參數可預設，預設值為 0 關於中斷優先級更詳細的信息見 6.2 章节
interval	數據類型：double
	定義觸發中斷的時間間隔，單位：秒 當 rmode 參數值為 true 時，interval 的值最小可以指定為 0.001，當 rmode 參數值為 false 時，interval 的值最小可以指定為 0
rmode	數據類型：bool
	定義 repeat 模式 ■ false 表示只在時間間隔 interval 後觸發一次中斷

名稱	說明
	■ true 表示每隔時間間隔 interval 後反覆觸發中斷
do	數據類型：any
	定義中斷處理動作 當中斷事件發生時，則會執行該表達式動作

用法舉例

timer name:"t1",priority:1,interval:1,rmode:true,do:setdo(1,true) //每隔 1s 執行一次 setdo(1,true)

7 軌跡觸發

在某些情況下，用戶希望在一條運動軌跡的中間某個點觸發一些事件而又不希望破壞這條運動軌跡的動態特性。這時可以使用軌跡觸髮指令。ARL 軌跡觸髮指令可以實現在一條運動軌跡的多個時間點或者多個距離點觸發多個觸發函數來達到用戶希望的行為。

7.1 軌跡觸發聲明

ARL 中軌跡觸發實際是一種特殊形式的中斷，所以軌跡觸發的聲明與中斷聲明指令格式差不多，唯一的區別是軌跡觸發聲明中沒有名字參數。

指令格式

trigger [priority:],when:,do:

參數

軌跡觸發聲明參數詳見表 7-1。

表 7-1 軌跡觸發聲明參數

名稱	說明
priority	數據類型：int
	指定軌跡觸發優先級 該優先級的定義同中斷優先級，並且當一個觸發事件和一個中斷事件在一個中斷掃描周期同時發生時也會優先響應具有較高優先級的中斷或者觸發事件 該參數可預設，預設值為 10
when	數據類型：bool
	定義觸發事件 觸發事件與中斷事件的定義相同。只是因為軌跡觸發是在軌跡運動過程中掃描的中斷，所以軌跡觸發事件一般都與軌跡信息相關，如軌跡的運行時間和運行距離 關於軌跡觸發事件更詳細的信息見
do	數據類型：any
	定義軌跡觸發動作 當軌跡觸發事件發生時，則會執行該表達式動作。軌跡觸發動作的定義與中斷觸發動作的定義相同

聲明的軌跡觸發將作用在該觸發聲明語句下麵最近的一條運動指令。

例如：

```
trigger 1,when:T(1),do:setdo(2,true) //該聲明作用於下一條 movej 指令
movej j:{j1 30},v:{per 2}
ptp p0,v:{per 50}
trigger 1,when:S(100),do:setdo(3,true) //該聲明作用於下一條 lin 指令
lin p1,v:{tcp 20,ori 10}
```

7.2 軌跡觸發事件

一般以 4 個系統預定義函數來定義觸發事件：

- 軌跡經過時間（T）
- 軌跡經過距離（S）
- 軌跡剩餘距離（StoEnd）

例如：

trigger 0,when:T(0.2),do:setdo(2,true)

該聲明的含義是在下一條運動語句的軌跡從軌跡起點出發 0.2s 時將 DO 的第 2 路信號置為 true。

例如：

trigger 0,when:StoEnd(10),do:setdo(2,true)

該聲明的含義是在下一條運動語句的軌跡運動到距離目標點 10 毫米時將 DO 的第 2 路信號置為 true。

更複雜的軌跡觸發事件可以通過系統變量$TRAJ_ ELAPSE _TIME，$TRAJ_LEFT_TIME，$TRAJ_ ELAPSE _DIS 和$TRAJ_LEFT_DIS 實現。

7.3 使用軌跡觸發註意事項

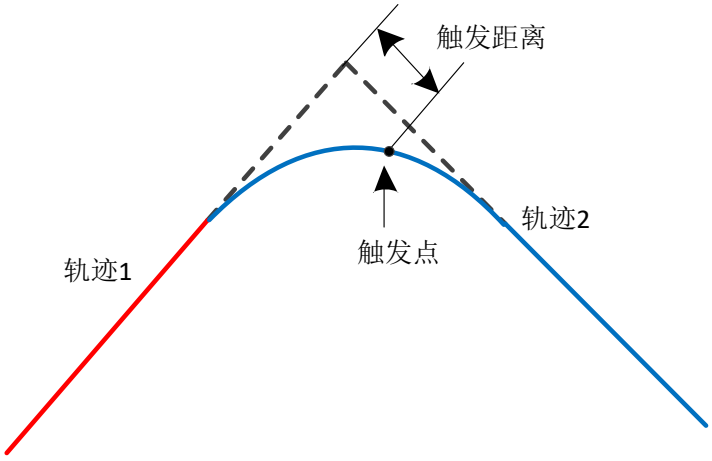


圖 71 軌跡觸發

使用軌跡觸發註意事項主要有以下幾點：

- PTP 和 MOVEJ 指令因為不關心 TCP 點軌跡，所以對於 PTP 和 MOVEJ 指令如果使用 S 和 StoEnd 函數指定軌跡觸發事件，觸發函數將不會被觸發。
- 由於插補精度以及信號輸出速度的問題，指定的觸發時間或者觸發距離與實際的觸發事件或者觸發距離可能會有幾個毫秒或者幾個毫米的誤差。
- 當指定的時間或者距離參數不在 0 至軌跡總時間或總距離範圍內時，聲明的軌跡觸發將不會被觸發。
- 在指定了平滑行為的情況下，平滑部分的軌跡將被認為屬於後一條軌跡，所以如果希望在平滑軌跡過程中觸發事件，需要在第 2 條軌跡前聲明觸發事件。軌跡路程仍會按照前一條軌跡未平滑時的目標點作為起點計算，如下圖 71 所示。

7.4 並行處理

描述

執行運動指令的過程中，並行處理一些動作，功能上與軌跡觸發一致。併列處理格式無法指定優先級（默認為 10），其餘功能與軌跡觸發完全一致。

指令格式

運動指令;觸發事件,觸發動作

運動指令;觸發事件 1,觸發動作 1;...;觸發事件 i,觸發動作 i;...;觸發事件 n,觸發動作 n

參數

名稱	說明
觸發事件	定義觸發事件 觸發事件與中斷事件的定義相同。只是因為軌跡觸發是在軌跡運動過程中掃描的中斷，所以軌跡觸發事件一般都與軌跡信息相關，如軌跡的運行時間和運行距離關於軌跡觸發事件更詳細的信息見 第 7.3 章節
觸發動作	定義軌跡觸發動作 當軌跡觸發事件發生時，則會執行該表達式動作。軌跡觸發動作的定義與中斷觸發動作的定義相同

使用舉例

舉例如下：

movej j:{j1 30},v:{per 2};T(1)，setdo(2,true)

//功能完全等同於以下 2 條指令

trigger 10,when:T(1),do:setdo(2,true)
movej j:{j1 30},v:{per 2}

lin p1,v:{tcp 20,ori 10};T(1)，setdo(2,true)；S(100),setdo(3,true)

//功能完全等同於以下 3 條指令

trigger 10,when:T(1),do:setdo(2,true)
trigger 10,when:S(100),do:setdo(3,true)
lin p1,v:{tcp 20,ori 10}

8 模塊化程式設計

ARL 支持模塊化程式設計，即可以在一個 arl 文件中調用另一個 arl 文件中定義的函數或者訪問另一個 arl 文件中定義的全局變量。

該功能通過“::”作用域操作符實現。

例如：

a.arl 文件

```
double gc = 1
func double add(double x,double y)
return x+y
endfunc
```

該文件中定義了一個全局變量 gc 和一個函數 add。

如果想在 b.arl 文件中調用 add 函數或者引用變量 gc，可以使用如下方式：

b.arl 文件

```
func void main()
print a::add(1,2) + a::gc
endfunc
```

運行程序後，將輸出 4。

以上程序假設 a.arl 文件與 b.arl 文件在同一目錄下。

如果 a.arl 文件與 b.arl 文件不在同一個目錄下，則需要借助 import 指令：

b.arl 文件

```
import “/XXX/XXX/a.arl”
func void main()
print a::add(1,2) + a::gc
endfunc
```

其中，/XXX/XXX/a.arl 指 a.arl 文件的絕對路徑。

- 加載一個 arl 文件時，系統自動導入一個名字為該 arl 文件+_data.arl 模塊（如果該文件存在的話），例如對於 b.arl 文件，如果同目錄下存在一個叫 b_data.arl 的文件，則系統會自動將它導入，此時引用 b_data.arl 模塊中的全局變量時則可以預設“模塊名::”，而直接通過變量名引用。
- 模塊名即為不帶.arl 的文件名。
- 被調用函數所在的文件名命名必須符合變量命名規則，且被調用 arl 文件名不允許為純數字或中文。
- 引用其他模塊的函數或者變量時，必須通過“模塊名::”的方式。

9 外部自動控制

可通過外部自動控制功能實現通過其他控制器（如機床 PLC）控制機器人系統。

使用外部自動控制功能需要完成以下工作：

- 外部自動控制功能配置。
- 根據配置連接外部控制器和機器人控制系統埠。
- 編寫外部自動控制主程序並測試運行。

9.1 外部自動控制功能配置

由於外部自動控制功能使用用戶 IO 埠實現和外部控制器之間的交互，所以首先必須對各種外部自動控制功能配置相應的埠號。

這裡包括對輸入端信號以及輸出端信號的配置。這裡的輸入輸出端是從機器人控制系統的角度來說。

9.1.1 輸入端信號配置

$EXT_CTL_ACT_DI

- 變量類型：int
- 變量功能：定義外部自動控制激活信號 DI 埠號。

如果將該系統變量定義為[1~40*系統接入的 MF 個數]之間的一個整數時，則外部控制器必須通過將該埠信號置 true 來激活機器人控制系統的外部自動控制功能；定義為 0 時，表示不使用該激活功能，即機器人控制系統默認激活外部自動控制功能。

$SERVO_ON_DI

- 變量類型：int
- 變量功能：定義伺服上電信號 DI 埠號。

如果將該系統變量定義為一個有效埠號整數時，在此輸入埠上給入脈寬至少 30ms 的高脈衝，則機器人控制系統會進行伺服上電動作；定義為 0 時表示不啟用該功能。

$SERVO_OFF_DI

- 變量類型：int
- 變量功能：定義伺服斷電信號 DI 埠號。

如果將該系統變量定義為一個有效埠號整數時，在此輸入埠上給入脈寬至少 30ms 的高脈衝，則機器人控制系統會進行伺服斷電動作；定義為 0 時表示不啟用該功能。

$START_PROG_DI

- 變量類型：int
- 變量功能：定義啟動程序信號 DI 埠號。

如果將該系統變量定義為一個有效埠號整數時，在此輸入埠上給入一個上升沿信號，則機器人控制系統會啟動通道中加載的程序的執行（這個程序通常為用戶編寫的外部自動控制主程序）；定義為 0 時表示不啟用該功能。

$PAUSE_PROG_DI

- 變量類型：int
- 變量功能：定義暫停程序信號 DI 埠號。

如果將該系統變量定義為一個有效埠號整數時，在此輸入埠上給入一個上升沿信號，則機器人控制系統會暫停通道中加載的程序的執行（這個程序通常為用戶編寫的外部自動控制主程序）；定義為 0 時表示不啟用該功能。

$RESET_PROG_DI

- 變量類型：int
- 變量功能：定義複位程序信號 DI 埠號。

如果將該系統變量定義為一個有效埠號整數時，在此輸入埠上給入 true 信號，則機器人控制系統會複位通道中加載的程序；定義為 0 時表示不啟用該功能。註意，如果使用該信號，如要啟動程序時，需要先給該 DI 信號輸入 false 信號，否則程序將不能夠啟動。

$CLEAR_ALARM_DI

- 變量類型：int
- 變量功能：定義清除報警信號 DI 埠號。

如果將該系統變量定義為一個有效埠號整數時，在此輸入埠上給入一個上升沿信號，則機器人控制系統會清除當前報警；定義為 0 時表示不啟用該功能。

$PGNO_TYPE

- 變量類型：int
- 變量功能：定義程序號格式。取值範圍為 0，1，2。含義如下表 9-1 所示。

表 9-1 $PGNO_TYPE 變量取值範圍含義

值	說明	示例
0	以二進制格式讀取	0 0 0 0 0 1 1 1 $PGNO = 7
1	以 BCD 編碼格式讀取，每 4 個二進制位表示一個 BCD 碼。該格式下每 4 個二進制位對應的十進制數不能大於 9	0 0 0 1 0 1 0 0 $PGNO = 14
2	以“N 選 1”格式讀取。該格式下 N 位長度中必須只能有 1 位是 1，其他位都是 0	0 0 0 0 0 0 0 1 $PGNO = 0 0 0 0 1 0 0 0 0 $PGNO = 4

$PGNO_LENGTH

- 變量類型：int
- 變量功能：定義程序號位寬

取值範圍為 1~16。當$PGNO_TYPE 值為 1，也就是 BCD 編碼格式讀取程序號時，$PGNO_LENGTH 取值只能為 4、8、12、16。

$PGNO_FBIT_DI

- 變量類型：int

- 變量功能：定義程序號第一位所在的 DI 埠號。

例如當此變量設置為 21，並且程序號位寬設置為 4 時，則程序號由第 21、22、23、24 這 4 路 DI 信號決定。

$PGNO_PARITY_DI

- 變量類型：int
- 變量功能：定義奇偶校驗信號 DI 埠號。

該變量值可以為正數，也可以為負數。該變量值的絕對值大小表示所用的埠號，負值表示使用奇校驗，正值表示使用偶校驗。其他無效值表示不對程序號進行奇偶校驗。當$PGNO_TYPE 值為 2，也就是“N 選 1”格式讀取程序號時，不管$PGNO_PARITY_DI 值為何值都不進行奇偶校驗。

$PGNO_VALID_DI

- 變量類型：int
- 變量功能：定義程序號有效信號 DI 埠號。

外部控制器給入該信號時，機器人控制器開始讀入程序號。該變量值為正數，值的大小表示所用的埠號，且信號上升沿有效。

$EXT_CTL_CHAN_DI

- 變量類型：int
- 變量功能：外部控制通道選擇起始位 DI 埠號

當機器人控制系統配置了多個前臺通道時，可通過該功能選擇外部控制功能生效的通道號。此參數為外部自動控制通道選擇 DI 的第一個邏輯地址，當該參數定義為有效 DI 邏輯地址時，以二進制格式讀取連續三路 DI 狀態，序號範圍為：0~5，對應第 1~6 個前臺通道。

舉例

若設置$EXT_CTL_CHAN_DI = 1 時：

- 當 1~3 路 DI 的電平狀態分別為 000（對應序號 1）時，則前臺第 1 通道的外部控制功能生效。
- 當 1~3 路 DI 的電平狀態分別為 011（對應序號 3）時，則前臺第 4 通道的外部控制功能生效。

9.1.2 輸出端信號配置

$EXT_CTL_ACT_CONF_DO

- 變量類型：int
- 變量功能：定義外部自動控制功能激活確認信號 DO 埠號。

當外部自動控制被激活時，該信號輸出 true，否則該信號輸出 false。激活外部自動控制功能需要進行如下操作：

通過 HMI 的設置頁面使能外部自動控制功能；外部控制器給入激活信號；將該變量設置為一個不存在的通道號時表示不啟用該功能。

$SERVO_ON_DO

- 變量類型：int
- 變量功能：定義伺服上電信號 DO 埠號。

當機器人控制系統處於伺服上電狀態時，該信號輸出 true，否則該信號輸出 false。將該變量設置為一個不存在的通道號時表示不啟用該功能。

$PGNO_REQ_DO

- 變量類型：int
- 變量功能：定義請求程序號信號 DO 埠號。

當用戶將系統變量$PGNO_REQ 置為 true 時，系統會將$PGNO_REQ_DO 定義的 DO 信號置 true 表示向外部控制器請求程序號。將該變量設置為一個不存在的通道號時表示不啟用該功能。

$AT_HOME_DO

- 變量類型：int[5]
- 變量功能：定義處於 HOME 點信號 DO 埠號。

系統可設置 5 個 home 點，每個 HOME 點代表位置可在實時位置界面修改。該 DO 信號表明機器人當前是否處於某個 home 點的位置。

$AT_T1_DO

- 變量類型：int
- 變量功能：定義系統處於手動低速模式信號 DO 埠號。

當系統處於手動低速模式時，該信號輸出 true，否則該信號輸出 false。將該變量設置為一個不存在的通道號時表示不啟用該功能。

$AT_T2_DO

- 變量類型：int
- 變量功能：定義系統處於手動高速模式信號 DO 埠號。

當系統處於手動高速模式時，該信號輸出 true，否則該信號輸出 false。將該變量設置為一個不存在的通道號時表示不啟用該功能。

$AT_AUT_DO

- 變量類型：int
- 變量功能：定義系統處於自動模式信號 DO 埠號。

當系統處於自動模式時，該信號輸出 true，否則該信號輸出 false。將該變量設置為一個不存在的通道號時表示不啟用該功能。

$PGNO_ACK_FBIT_DO

- 變量類型：int
- 變量功能：定義程序號確認信號第一位所在的 DO 埠號。

當機器人系統收到外部通過 DI 信號給入的程序號後，如果該程序號合法，則會通過此系統變量定義的 DO 信號組將程序號輸出給外部控制器，外部控制器可以通過這個程序號反饋來校驗機器人控制器是否收到程序號。將該變量設置為一個不存在的通道號時表示不啟用該功能。

$CHAN_STATE_DO

- 變量類型：int

- 變量功能：當前通道狀態起始邏輯地址號。

如值為 5 時，表示使用 5、6 地址號作為通道狀態輸出。00:未加載；10:運行；01:暫停；11:停止。將該變量設置為一個不存在的通道號時表示不啟用該功能。

10 系統變量

ARL 中變量分為 3 種類型（參見）：

- 局部變量
- 全局變量
- 系統變量

系統變量為系統預定義的有特殊含義的變量。用戶在使用時不需要聲明，可以直接使用。使用系統變量和使用用戶自定義變量沒有區別。

每個系統變量都有其自己的變量類型，默認值和含義。有些系統變量還有最大值和最小值的限制。用戶只能向這些系統變量寫入最大值和最小值之間的值，如果試圖向這些系統變量賦超限值時，系統會報錯。

以下說明可以在用戶程序中引用的系統變量。

10.1 數據類型系統變量

10.1.1 $I_NAME(整形系統變量名稱)

表 10-1 $I_NAME 變量

屬性	說明
變量名	整形系統變量名稱
數據類型	string 數組[100]
最大值	\
最小值	\
默認值	
何時恢復默認值	該變量改動立即生效，掉電重啟後仍為上次修改值
功能描述	系統預定義的整形變量名稱數組，其含義由用戶自定義
值含義	用戶自定義

10.1.2 $B(布爾型系統變量)

表 10-2 $B 變量

屬性	說明
變量名	布爾型系統變量
數據類型	bool 數組[100]
最大值	-
最小值	-
默認值	出廠默認值為全 false
何時恢復默認值	用戶在程序中給數組的某些元素賦予新值後，可以通過 savesv(“B”)函數將數組所有元素的值存儲到配置文件中，使得數據掉電不丟失。如果不保存，則下次啟動系統時數據恢復為上一次保存的值

屬性	說明
功能描述	系統預定義的布爾型變量數組，其含義用戶可以自定義
值含義	用戶自定義

10.1.3 $D(浮點型系統變量)

表 10-3 $D 變量

屬性	說明
變量名	浮點型系統變量
數據類型	double 數組[100]
最大值	浮點型數據的最大值
最小值	浮點型數據的最小值
默認值	出廠默認值為全 0
何時恢復默認值	用戶在程序中給數組的某些元素賦予新值後，可以通過 savesv(“D”)函數將數組所有元素的值存儲到配置文件中，使得數據掉電不丟失。如果不保存，則下次啟動系統時數據恢復為上一次保存的值
功能描述	系統預定義的浮點型變量數組，其含義用戶可以自定義
值含義	用戶自定義

10.1.4 $P(結構體類型系統變量 P)

表 10-4 $P 變量

屬性	說明
變量名	結構體類型系統變量 P
數據類型	pose 數組[100]
最大值	-
最小值	-
默認值	出廠默認值為全 false
何時恢復默認值	用戶在程序中給數組的某些元素賦予新值後，可以通過 savesv(“P”)函數將數組所有元素的值存儲到配置文件中，使得數據掉電不丟失。如果不保存，則下次啟動系統時數據恢復為上一次保存的值
功能描述	系統預定義的結構體類型變量數組，其含義用戶可以自定義
值含義	用戶自定義

10.1.5 $J(結構體類型系統變量 J)

表 10-5 $J 變量

屬性	說明
變量名	結構體類型系統變量 J
數據類型	joint 數組[100]

屬性	說明
最大值	-
最小值	-
默認值	出廠默認值為全 false
何時恢復默認值	用戶在程序中給數組的某些元素賦予新值後，可以通過 savesv(“J”)函數將數組所有元素的值存儲到配置文件中，使得數據掉電不丟失。如果不保存，則下次啟動系統時數據恢復為上一次保存的值
功能描述	系統預定義的結構體類型變量數組，其含義用戶可以自定義
值含義	用戶自定義

10.1.6 $TOOLS(工具坐標系)

表 10-6 $TOOLS 變量

屬性	說明
變量名	工具坐標系
數據類型	tool 結構體數組[32]
最大值	-
最小值	-
默認值	-
何時恢復默認值	該系統變量通過 HMI 修改將會永久保存，如果通過程序修改則在系統重啟後將恢復為之前保存的值
功能描述	存儲用戶自定義工具坐標系
值含義	用戶自定義工具坐標系

10.1.7 $TOOLS_NAME(工具坐標系名稱)

表 10-7 $TOOLS_NAME 變量

屬性	說明
變量名	工具坐標系名稱
數據類型	tool 結構體數組[32]
最大值	-
最小值	-
默認值	-
何時恢復默認值	該變量改動立即生效，掉電重啟後仍為上次修改值
功能描述	存儲用戶自定義工具坐標系
值含義	用戶自定義工具坐標系

10.1.8 $WOBJS(工件坐標系)

表 10-8 $WOBJS 變量

屬性	說明
變量名	工件坐標系
數據類型	wobj 結構體數組[32]
最大值	-
最小值	-
默認值	-
何時恢復默認值	該系統變量通過 HMI 修改將會永久保存，如果通過程序修改則在系統重啟後將恢復為之前保存的值
功能描述	存儲用戶自定義工件坐標系
值含義	用戶自定義工件坐標系

10.1.9 $WOBJS_NAME(工件坐標系名稱)

表 10-9 $WOBJS_NAME 變量

屬性	說明
變量名	工件坐標系名稱
數據類型	wobj 結構體數組[32]
最大值	-
最小值	-
默認值	-
何時恢復默認值	該變量改動立即生效，掉電重啟後仍為上次修改值
功能描述	存儲用戶自定義工件坐標系
值含義	用戶自定義工件坐標系

10.1.10$BASE(基礎坐標系)

表 10-10 $BASE[]變量

屬性	說明
變量名	基礎坐標系
數據類型	wobj
取值範圍	0,1,2
最大值	-
最小值	-
默認值	{{0,0,0,0,0,0},false} {{0,0,0,0,0,0},false} {{0,0,0,0,0,0},false}
何時恢復默認值	該系統變量通過 HMI 修改將會永久保存，如果通過程序修改則在系統重啟後將恢復為之前保存的值

屬性	說明
功能描述	定義在機器人足底的基礎坐標系
值含義	基礎坐標系，相對世界坐標系定義

10.1.11$FLANGE(法蘭坐標系)

表 10-11 $FLANGE 常量

屬性	說明
變量名	法蘭坐標系
數據類型	tool
值	{{0,0,0,0,0,0,},false,}
何時恢復默認值	-
功能描述	存儲系統預定義的特殊工具坐標系：法蘭坐標系
值含義	系統預定義法蘭坐標系

10.1.12$WORLD(世界坐標系)

表 10-12 $WORLD 常量

屬性	說明
變量名	世界坐標系
數據類型	wobj
值	{{0,0,0,0,0,0,},false, WORLD}
何時恢復默認值	-
功能描述	存儲系統預定義的特殊工件坐標系：世界坐標系
值含義	系統預定義世界坐標系

10.2 功能類型系統變量

10.2.1 $WRIST(開啟腕部奇異點避讓)

表 10-13 $WRIST 變量

屬性	說明
變量名	布爾型系統變量
數據類型	bool
最大值	\
最小值	\
默認值	出廠默認值為 false
何時恢復默認值	軟體啟動時、程序複位時
功能描述	機器人軌跡理論上不能穿越奇異點，在接近奇異點過程中會報警超速。其中，腕部奇異點是指機器人 5 軸為 0 的位置。當需要穿越 5 軸為 0 的位置進行 lin、cir 等笛

屬性	說明
	卡爾軌跡運動時，可將系統變量設置為 true，開啟腕部奇異點避讓功能。此時，機器人會部分犧牲姿態精度，保證 TCP 精度，從而穿過奇異點。完成穿越後，將系統變量設置為 false，可繼續進行普通運動。
值含義	■ true：開啟 ■ false：關閉

10.2.2 $I(整型系統變量)

表 10-14 $I 變量

屬性	說明
變量名	整型系統變量
數據類型	int 數組[100]
最大值	整型數據的最大值
最小值	整型數據的最小值
默認值	出廠默認值為全 0
何時恢復默認值	用戶在程序中給數組的某些元素賦予新值後，可以通過 savesv(“I”)函數將數組所有元素的值存儲到配置文件中，使得數據掉電不丟失。如果不保存，則下次啟動系統時數據恢復為上一次保存的值
功能描述	系統預定義的整型變量數組，其含義用戶可以自定義
值含義	用戶自定義

10.2.3 $B_NAME(布爾形系統變量名稱)

表 10-15 $B_NAME 變量

屬性	說明
變量名	布爾形系統變量名稱
數據類型	string 數組[100]
最大值	\
最小值	\
默認值	
何時恢復默認值	該變量改動立即生效，掉電重啟後仍為上次修改值
功能描述	系統預定義的布爾形變量名稱數組，其含義由用戶自定義
值含義	用戶自定義

10.2.4 $ config_check (軸配置檢查使能)

表 10-16 $ config_check 變量

屬性	說明
變量名	軸配置檢查使能

數據類型	bool
默認值	true
何時恢復默認值	軟體啟動時，程序複位時
功能描述	機器人的 TCP 到達同一個空間中的點位，對應的軸位置可能是不同的，可參考 2.4.4 節 pose 結構體中 turn 值的說明。如果在示教時，兩個點位的軸位置不同（比如出現了六軸差了 1 圈的情況），實際運行時需要用該參數決定是否要運動到示教時所獲取的軸位置。
值含義	■ true：按照示教時軸配置運動 ■ false：按照距當前軸位置最近的軸配置運動

10.2.5 $D_NAME(浮點形系統變量名稱)

表 10-17 $D_NAME 變量

屬性	**說明**
變量名	浮點形系統變量名稱
數據類型	string 數組[100]
最大值	\
最小值	\
默認值	
何時恢復默認值	該變量改動立即生效，掉電重啟後仍為上次修改值
功能描述	系統預定義的浮點形變量名稱數組，其含義由用戶自定義
值含義	用戶自定義

10.2.6 $DFSPEED(默認速度參數)

表 10-18 $DFSPEED 變量

屬性	**說明**
變量名	默認速度參數
數據類型	speed
最大值	-
最小值	-
默認值	{5,50,400,5,5}
何時恢復默認值	軟體啟動時、程序複位時
功能描述	當指令中未指定速度參數 v 時則使用此默認參數。
值含義	參見 speed 類型定義

10.2.7 $DFSLIP(默認平滑參數)

表 10-19 $DFSLIP 變量

屬性	說明
變量名	默認平滑參數
數據類型	slip
最大值	-
最小值	-
默認值	{-1.0,-1.0}
何時恢復默認值	軟體啟動時、程序複位時
功能描述	當指令中未指定平滑參數 s 時則使用此默認參數
值含義	參見 slip 類型定義

10.2.8 $DFTOOL(默認工具參數)

表 10-20 $DFTOOL 變量

屬性	說明
變量名	默認工具參數
數據類型	tool
最大值	-
最小值	-
默認值	{{0,0,0,0,0,0},false}
何時恢復默認值	軟體啟動時、程序複位時
功能描述	當指令中未指定工具參數時則使用此默認參數
值含義	參見 tool 類型定義

10.2.9 $DFWOBJ(默認工件坐標系參數)

表 10-21 $DFWOBJ 變量

屬性	說明
變量名	默認工件坐標系參數
數據類型	wobj
最大值	-
最小值	-
默認值	{{0,0,0,0,0,0},false}
何時恢復默認值	軟體啟動時、程序複位時
功能描述	當指令中未指定工件坐標系參數時則使用此默認參數
值含義	參見 wobj 類型定義

10.2.10$IGNORE_ORI(方向忽略使能)

表 10-22 $IGNORE_ORI 變量

屬性	說明
變量名	方向忽略使能
數據類型	bool
最大值	-
最小值	-
默認值	false
何時恢復默認值	軟體啟動時、程序複位時
功能描述	該值為 true 時，將忽略運動語句中的目標點的 A,B,C 方向參數，目標點方向將保持和起點一致
值含義	true：使能

10.2.11$ORI_REF_PATH(圓弧方向參照路徑坐標系)

表 10-23 $ORI_REF_PATH 變量

屬性	說明
變量名	圓弧方向參照路徑坐標系
數據類型	bool
最大值	-
最小值	-
默認值	false
何時恢復默認值	軟體啟動時、程序複位時
功能描述	設置圓弧軌跡方向是否參照圓弧路徑坐標系
值含義	true：cir 語句的軌跡方向參照圓弧路徑坐標系

10.2.12$VEL_PROFILE(速度輪廓類型)

表 10-24 $VEL_PROFILE 變量

屬性	說明	
變量名	速度輪廓類型	
數據類型	$VEL_PROFILE	
最大值	-	
最小值	-	
默認值	不同的機型（對應示教器中的機械單元型號）默認值不同	3A、6A、6L、10A 的默認值為 O_type
		其它機型默認值為 S_type
何時恢復默認值	軟體啟動時、程序複位時	
功能描述	設置運動軌跡的速度輪廓 每條運動指令都會以該參數中設置了速度輪廓類型來進行規劃	

屬性	說明
	■ S 型速度規劃優點為平穩無衝擊，缺點為規划出來的運動時間會比較長 ■ T 型速度規劃的優點為規划出來的運動時間比較短，缺點為對機械有衝擊 ■ O 型速度規劃優點為節拍快且機械衝擊小
值含義	■ S_type：S 型速度輪廓 ■ T_type：T 型速度輪廓 ■ O_type：O 型速度輪廓

10.2.13$ACC_OVERRIDE(加速度倍率)

表 10-25 $ACC_OVERRIDE 變量

屬性	說明
變量名	加速度倍率
數據類型	double
最大值	100
最小值	0.01
默認值	100
何時恢復默認值	軟體啟動時、程序複位時
功能描述	加速度倍率 ■ Double ■ 100 ■ 0.01 ■ 100 軟體啟動時、程序複位時
值含義	值除以 100 表示倍率百分比，在用戶程序中給該系統變量賦值，以下所有運動指令的加速度將以系統最大加速度乘以該倍率才是最終的加速度值

10.2.14$JERK_OVERRIDE(加加速度倍率)

表 10-26 $JERK_OVERRIDE 變量

屬性	說明
變量名	加加速度倍率
數據類型	double
最大值	100
最小值	0.01
默認值	100
何時恢復默認值	軟體啟動時、程序複位時
功能描述	設置加加速度倍率。默認情況下，系統以定義的最大加加速度來進行速度規劃。如果希望機器人的運動更加平穩，可以通過這個系統變量重新定義加加速度值
值含義	值除以 100 表示倍率百分比，在用戶程序中給該系統變量賦值，以下所有運動指令

屬性	說明
	的加加速度將以系統最大加加速度都乘以該倍率才是最終的加加速度值

10.2.15$TRAJ_ELAPSE_TIME(軌跡經過時間)

表 10-27 $TRAJ_ELAPSE_TIME 變量

屬性	說明
變量名	軌跡經過時間
數據類型	double
最大值	-
最小值	-
默認值	0
何時恢復默認值	程序複位時由系統恢復為默認值
功能描述	存儲軌跡經過時間，一般用於軌跡觸發聲明中
值含義	軌跡經過時間，單位秒

10.2.16$TRAJ_LEFT_TIME(軌跡剩餘時間)

表 10-28 $TRAJ_LEFT_TIME 變量

屬性	說明
變量名	軌跡剩餘時間
數據類型	double
最大值	-
最小值	-
默認值	0
何時恢復默認值	程序複位時由系統恢復為默認值
功能描述	存儲軌跡剩餘時間，一般用於軌跡觸發聲明中
值含義	軌跡剩餘時間，單位秒

10.2.17$TRAJ_ELAPSE_DIS(軌跡經過路程)

表 10-29 $TRAJ_ELAPSE_DIS 變量

屬性	說明
變量名	軌跡經過路程
數據類型	double
最大值	-
最小值	-
默認值	0
何時恢復默認值	程序複位時由系統恢復為默認值

屬性	說明
功能描述	存儲軌跡經過路程，一般用於軌跡觸發聲明中
值含義	軌跡經過路程，單位毫米

10.2.18$TRAJ_LEFT_DIS(軌跡剩餘路程)

表 10-30 $TRAJ_LEFT_DIS 變量

屬性	說明
變量名	軌跡剩餘路程
數據類型	double
最大值	-
最小值	-
默認值	0
何時恢復默認值	程序複位時由系統恢復為默認值
功能描述	存儲軌跡剩餘路程，一般用於軌跡觸發聲明中
值含義	軌跡剩餘路程，單位毫米

10.2.19$CJOINT(當前軸位置點)

表 10-31 $CJOINT 變量

屬性	說明
變量名	當前軸位置點
數據類型	joint
最大值	-
最小值	-
默認值	-
何時恢復默認值	程序複位時由系統恢復為默認值
功能描述	該系統變量的值始終由系統賦值為上一條運動軌跡的目標點軸位置，所以可使用該系統變量實現增量程式設計
值含義	當前軸位置點

10.2.20$RPP_ENABLE(RPP 使能)

表 10-32 $RPP_ENABLE 變量

屬性	說明
變量名	RPP 使能
數據類型	bool
最大值	-
最小值	-

屬性	說明
默認值	false
何時恢復默認值	軟體啟動時、程序複位時
功能描述	RPP(return to path point)軌跡是指自動程序運行的第一條軌跡。 ■ 當 RPP 功能生效時，RPP 軌跡退化為 movej 軌跡，速度為手動點動速度； ■ 當 RPP 功能失效時，RPP 軌跡根據客戶設定的指令類型移動，但運行速度由手動限速控制。由於無法確定機器人在 RPP 移動前的位置，可能會出現無法到達的警報
值含義	■ true：使能 RPP 功能 ■ false：屏蔽 RPP 功能

10.2.21$AT_HOME(是否處於 HOME 位置)

表 10-33 $AT_HOME 變量

屬性	說明
變量名	是否處於 HOME 位置
數據類型	bool 型數組[5]
最大值	-
最小值	-
默認值	{false,false,false,false,false}
何時恢復默認值	-
功能描述	系統可設置 5 個 home 點，每個 HOME 點代表位置可在實時位置界面修改。該系統變量表明機器人當前是否處於某個 home 點的位置。該系統變量對用戶只讀，由系統內部修改
值含義	■ true：機器人處於 HOME 位置 ■ false：機器人不處於 HOME 位置

10.2.22$EXT_CTL_ACT(外部自動控制被激活)

表 10-34 $EXT_CTL_ACT 變量

屬性	說明
變量名	外部自動控制被激活
數據類型	bool
最大值	-
最小值	-
默認值	-
何時恢復默認值	-
功能描述	該系統變量表明當前外部自動控制是否被激活。該系統變量對用戶只讀，由系統內部修改
值含義	■ true：外部自動控制被激活

屬性	說明
	■ false：外部自動控制未激活

10.2.23$PGNO_REQ(請求程序號狀態)

表 10-35 $PGNO_REQ 變量

屬性	說明
變量名	請求程序號狀態
數據類型	bool
最大值	-
最小值	-
默認值	false
何時恢復默認值	軟體啟動時，程序複位時
功能描述	該系統變量表明當前系統處於請求程序號狀態。該系統變量用於使能外部自動控制時，將該系統變量值置為 true 時，系統變量$PGNO_REQ_DO 定義的 DO 輸出埠會輸出 true，表明當前系統處於請求程序號狀態
值含義	■ true：系統處於請求程序號狀態 ■ false：系統不處於請求程序號狀態

10.2.24$PGNO(從外部獲取的程序號)

表 10-36 $PGNO 變量

屬性	說明
變量名	從外部獲取的程序號
數據類型	int
最大值	-
最小值	-
默認值	-1
何時恢復默認值	軟體啟動時
功能描述	當使能外部自動控制，外部控制系統通過 DI 埠給入程序號有效信號時，系統會從 DI 埠讀入程序號，如果該程序號為有效的程序號，則系統會將該系統變量置為對應的程序號。用戶可以在外部自動控制主程序中根據該系統變量的不同值來執行不同的子程序
值含義	從外部獲取的有效程序號

10.2.25$FAST_SMOOTH(快速平滑模式)

表 10-37 $FAST_SMOOTH 變量

屬性	說明
變量名	快速平滑模式
數據類型	bool

屬性	說明
最大值	-
最小值	-
默認值	false
何時恢復默認值	軟體啟動時，程序複位時
功能描述	當使用平滑功能時，快速平滑模式相比普通平滑模式可以提高更多的效率，但使用快速平滑模式時，平滑軌跡段的速度可能會超過用戶設置速度，同時快速平滑模式不保證不同速度設置下的平滑軌跡路徑一致性。一般情況下建議使用普通平滑模式
值含義	■ true：切換到快速平滑模式 ■ false：切換到普通平滑模式

10.2.26$PI(圓周率)

表 10-38 $PI 常量

屬性	說明
變量名	圓周率
數據類型	double
值	3.1415926535897932
何時恢復默認值	-
功能描述	系統預定義圓周率常數
值含義	圓周率

10.2.27$CTL_MODE(當前控制模式)

表 10-39 $CTL_MODE 變量

屬性	說明
變量名	當前控制模式
數據類型	controlmode
最大值	-
最小值	-
默認值	-
何時恢復默認值	-
功能描述	該系統變量表明當前系統處於何種控制模式。該系統變量對用戶只讀，由系統內部修改
值含義	■ T1：手動低速模式 ■ T2：手動高速模式 ■ AUT：自動模式

10.2.28$WOBJ_OFFSET(工件坐標系偏移)

表 10-40 $WOBJ_OFFSET 變量

屬性	說明
變量名	工件坐標系偏移，此次的工件坐標系是運動指令參考的坐標系。可以為：世界坐標系、工件坐標系、基座標系、工具坐標系和法蘭坐標系。
數據類型	double 數組[6]
最大值	-
最小值	-
默認值	{0,0,0,0,0,0}
何時恢復默認值	軟體啟動時、程序複位時
功能描述	通過修改此系統變量，可實現目標點位的批量偏移
值含義	用戶自定義工件坐標系偏移值

使用示例一如下：

```
pose p1 = { x 400, y 0, z 800, a 0, b 0, c 00}
$WOBJ_OFFSET={50,50,50,0,0,0}
lin p:p1,vp:50%,sp:-1%,t:$FLANGE,w:$WORLD //此時，$WORLD = {{50,50,50,0,0,0},false,WORLD}
print cpose($FLANGE,$WORLD) //輸出“{ 450, 50, 850, 0, 0, 0, 0, -1, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”
```

10.2.29$TOOL_OFFSET(工具坐標系偏移)

表 10-41 $TOOL_OFFSET 變量

屬性	說明
變量名	工具坐標系偏移
數據類型	double 數組[6]
最大值	-
最小值	-
默認值	{0,0,0,0,0,0}
何時恢復默認值	軟體啟動時、程序複位時
功能描述	通過修改此系統變量，可實現目標點位的批量偏移
值含義	用戶自定義工具坐標系偏移值

使用示例如下：

```
pose p1 = { x 400, y 0, z 800, a 0, b 0, c 00}
$TOOL_OFFSET={50,0,0,0,0,0}
```

lin p:p1,vp:50%,sp:-1%,t:$FLANGE,w:$WORLD //此時，$FLANGE = {{50,0,0,0,0,0},false}

print cpose($FLANGE,$WORLD) //輸出“{ 350, 0, 800, 0, 0, 0, -1, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”

附錄 A ARL 關鍵字

附表 A ARL 關鍵字一覽表

序號	名稱	序號	名稱	序號	名稱
1	accset	23	joint	45	stopbits
2	bool	24	jvel	46	stopmove
3	break	25	lin	47	stoptype
4	byte	26	loop	48	string
5	byte	27	movej	49	switch
6	ccir	28	num_base	50	timer
7	cir	29	parity	51	tool
8	clock	30	pause	52	ToolInertiaPara
9	compen	31	pos	53	toolload
10	continue	32	pose	54	toolswitch
11	controlmode	33	print	55	trigger
12	double	34	printto	56	uint
13	endcompen	35	ptp	57	velset
14	endweave	36	repeat	58	waittime
15	exit	37	restart	59	waituntil
16	for	38	return	60	weavedata
17	frame	39	scan	61	weaverotaxis
18	goto	40	slip	62	weaveshape
19	if	41	speed	63	while
20	import	42	startcompen	64	wobj
21	int	43	startmove		
22	interrupt	44	startweave		

附錄 B 指令索引表

配天机器人

微信公眾號

官方網站

服務熱線：400-990-0909
官方網站：http://robot.peitian.com

BJM/SS-UG-02-003 / V4.2.0 / 2021.03.30
© 版權所有2011–2021配天機器人保留所有權利.

有關產品特性和可用性說明並不構成效能保證，僅供參考。所交付產品和所執行的服務範圍以具體契约為准。