



EMotion

指令手冊

版本：1.1.0

2016 年 5 月 31 日

目錄

第 1 章 編程規範	1
<i>Program</i>	1
<i>Sub</i>	1
<i>Function</i>	1
<i>Call</i>	2
<i>dim</i>	4
<i>Print</i>	4
第 2 章 運動指令詳解	5
<i>Delay</i>	5
<i>Speed</i>	5
<i>SpeedEx</i>	6
<i>Accel</i>	6
<i>Decel</i>	7
<i>Move</i>	7
<i>ArchMove</i>	10
<i>GMove</i>	10
<i>DriveA</i>	11
<i>DriveI</i>	11
<i>MotorMotion</i>	12
第 3 章 I/O 指令詳解.....	14
<i>SetDO</i>	14
<i>SetDOEx</i>	14
<i>ResetDO</i>	14
<i>ResetDOEx</i>	15
<i>GetDI</i>	15
<i>Wait</i>	16
<i>SetDI</i>	17
<i>ResetDI</i>	17
<i>GetDO</i>	18
<i>GetADValue</i>	18
<i>SetDAValue</i>	18

第 4 章 位置指令詳解	20
<i>PosX</i>	20
<i>PosY</i>	20
<i>PosZ</i>	20
<i>PosRX</i>	20
<i>PosRY</i>	20
<i>PosRZ</i>	21
<i>LetX</i>	21
<i>LetY</i>	21
<i>LetZ</i>	22
<i>LetRX</i>	22
<i>LetRY</i>	22
<i>LetRZ</i>	22
<i>CurPos</i>	23
<i>CurJnt</i>	23
<i>Jnt</i>	23
<i>LetJ</i>	24
<i>SavePoint</i>	24
第 5 章 通訊功能指令	26
<i>NetSlave</i>	26
<i>NetServer</i>	27
<i>SerialPort</i>	28
第 6 章 文件操作與計時指令	30
<i>FileSystem</i>	30
<i>Timer</i>	31
第 7 章 流程控制指令詳解	33
<i>if-elseif-else</i>	33
<i>select-case</i>	34
<i>while</i>	34
<i>for</i>	35
<i>do loop</i>	35
<i>Goto</i>	36
<i>And</i>	37

<i>Or</i>	37
<i>Pause</i>	37
<i>Stop</i>	37
第 8 章 數學函式指令詳解	39
<i>sin</i>	39
<i>cos</i>	39
<i>tan</i>	39
<i>asin</i>	39
<i>acos</i>	39
<i>atan</i>	39
<i>exp</i>	40
<i>log</i>	40
<i>sqrt</i>	40
<i>ran</i>	40
<i>abs</i>	41
<i>fabs</i>	41
<i>mod</i>	41
第 9 章 字串處理指令詳解	42
<i>Left</i>	42
<i>Right</i>	42
<i>Mid</i>	42
<i>Len</i>	42
<i>Asc</i>	42
<i>Str</i>	42
<i>Chr</i>	43
<i>Val</i>	43
<i>Split</i>	43
第 10 章 多任務	45
<i>RunTask</i>	45
<i>PauseTask</i>	46
<i>ResumeTask</i>	46
<i>KillTask</i>	46
<i>GetTask</i>	47
<i>TakeArm</i>	49

<i>GiveArm</i>	49
第 11 章 其他功能指令	51
<i>Matrix</i>	51
<i>HomeEnable</i>	51
<i>Calibration</i>	52
<i>CalibrationStart</i>	52
<i>CalibrationStop</i>	52
<i>SetCrabTool</i>	53
<i>CalPosForCam</i>	53
第 12 章 視覺指令	55
<i>NetSlave</i>	55
<i>SetTrackParam</i>	56
<i>SetTrackTarget</i>	56
<i>GetTrackTarget</i>	57
<i>GetNextVision</i>	57
<i>TrackStart</i>	57
<i>GetObjectPos</i>	58

第 1 章 編程規範

Program

功能：當前項目程序的執行入口

格式：Program <名稱>

說明：<名稱>在整個程序中不能重名，此標識符在整個程序中隻能出現一次

舉例：

```
Program Main
    Print "this is main"
End Program
```

Sub

功能：執行子程序或函數

格式：Sub <名稱><參數>

說明：<名稱>在整個程序中不能重名，不能有返回值

舉例：

```
Sub subName(byval Num1 as integer,byval Num2 as integer)
    Print "Num1+Num2=", Num1+Num2
End Sub
Program Main
    subName(3,5)
End Program
```

Function

功能：執行自定義函數

格式：Function <名稱><參數>AS<返回數據類型>

說明：<名稱>在整個程序中不能重名，並且必須有返回值

舉例：

```
Function funName(byval Num1 as integer,byval Num2 as integer) as integer
    Dim Ret as integer
```

```

    Ret = Num1+Num2
    Print "Num1+Num2=", Ret
    Return Ret
End Function
Program Main
    funName(3,5)
End Program

```

Call

功能：呼叫函數。可將參數傳入函數做指定的處理，也可以把值傳回

格式：Call <函數名稱>

舉例：

```

Function AddFunc(byval Num1 as integer,byval Num2 as integer) as integer
    Dim rtnval As Integer
    rtnval = Num1+Num2
    Print " Num1,"+" ,Num2,"=" " ,rtnval
    return rtnval '傳回兩個數值相加的結果
End Function

```

```

Program Main
    Dim result As Integer
    Call AddFunc(3,5) '第一種呼叫函數的方式
    AddFunc(3,5) '第二種呼叫函數的方式
    result = AddFunc(3,5) '第三種呼叫函數的方式
                        '利用變量result 來承接AddFunc 的回傳值
    Print "result=", result '印出result=8
End Program

```

你可以在將函數定義在別的分 PAC 文件(在此我們將 Main 稱作主 PAC，將 Hello、Goodbye 稱作副 PAC)

舉例：

主PAC

```

Program Main           \ 主程序
    Call Hello(1) \ 列印出 "Hello
FoxBox"
    Call Hello(2) \ 列印出 "Hello
Foxconn"
    Goodbye( )         \ 列印出 "Goodbye
FoxBot"
End Program

```

副PAC

```

Sub Hello(byval para1 as integer)
'Hello.pac
    if para1 = 1 then
        Print "Hello FoxBot"
    else
        Print "Hello Foxconn"
    End if
End Sub

```

副PAC

```

Sub Goodbye( )      'Goodbye.pac
    Print "Goodbye FoxBot"
End Sub

```

你可以單獨執行副 PAC，並指派初始值給參數

舉例:

指定數值 1 給參數 para1，該數值只在單獨執行該 PAC 時才有作用

```

Sub Hello(para1=1)           'Hello.pac
    if para1 = 1 then
        Print "Hello FoxBot"
    else
        Print "Hello Foxconn"
    End if
End sub

```

注意:

副程序仍可以存取主程序所宣告的變量，但是函式不能，例如:

```

Function Hello()
    if status = 1 then           \ 會產生 RUN ERROR:Undefined variable
        print "Hello,Foxbot"
    end if

```


End Function

Program Main

Dim status As Integer

status = GetDI(0)

Call Hello

End Program

dim

功能：申明一個變量

格式：**dim**<變量名>**as**<變量類型>

舉例：

Dim n as integer

Dim rec as string

Print

功能：打印字符串

格式：**print** <字符串>

舉例：

print "the main program is running"

第 2 章 運動指令詳解

Delay

功能：使目前程序延遲指定的時間

格式：Delay <延遲時間>

說明：<延遲時間>的單位為 ms

舉例：

```
Program Main
  Dim duration As Integer
  duration=100
  Delay 100          `延遲 100ms(0.1s)
  Delay duration+10 `延遲 duration+10ms
End Program
```

Speed

功能：指定程序的內部速度比例

格式：Speed <運動速度比例>

說明：運動速度是內部關節合成最大速度的百分比(%)，範圍：1-200%。最大速度是指工具中心點(TCP)的最大速度。注意：機器人有內部速度和外部速度之分，外部速度是指示教器上設定的速度，內部速度是指用 Speed 指令設定的值。實際速度=最大速度*內部速度*外部速度，例如：內部速度=70%，外部速度 30%，實際速度=最大速度*0.7*0.3。如果在程序中省略了速度設定，而在之前使用了 Speed 指令，則以 Speed 指令設定的速度為主，若程序中沒有 Speed 指定，並且諸如 Move 的運動指令也沒有 S 選項，則系統的內部默認速度比例為 0.01%。

舉例：

```
Program Main
  TakeArm 0
  Move P,@P,P[1],S=80
  Speed 50          `指定內部速度為 50%
```

```

                                ` 在該程序中所有運動指令預設的內部速度均為 50%
Move P,@0,P[2]                ` 以目前位置運行到 P(2), 內部速度為 50%
                                ` @0: 到達目標位置的確認方式
Move P,@P,P[3],S=100          ` 改以內部速度 100% 的速度運動到 P(3)
GiveArm 0
End Program

```

SpeedEx

功能：指定程序的外部速度比例

格式：SpeedEx <運動速度比例> · 外部速度可以設置 0%-100%。

舉例：

```

Program Main
  TakeArm 0
  SpeedEx 50                ` 指定外部速度為 50%
  Move P,@0,P[2],S=20        ` 以目前位置運行到 P(2), 內部速度為 20% · 外部速度為 50%
                                ` @0: 到達目標位置的確認方式
  GiveArm 0
End Program

```

Accel

功能：指定內部加速度比例

格式：Accel <加速度比例> · 設置範圍 0-200%。

舉例：

```

Program Main
  TakeArm 0
  Move L,@E,P[3],S=100
  Accel 50
  Move L,@E,P[2],S=100
  GiveArm 0
End Program

```

Decel

功能：指定內部減速度比例

格式：Decel <減速度比例>，設置範圍 0-200%。

舉例：

```
Program Main
  TakeArm 0
  Move L,@E,P[3],S=100
  Decel 50
  Move L,@E,P[2],S=100
  GiveArm 0
End Program
```

Move

功能：移動到指定位置

格式：Move <插補方法>,@<到達目標位置確認方式>,<目標位置>,<S=速度比例>
>/<Accel=加速度比例>/<Decel=減速度比例>

說明：從目前位置移動到指定的目標位置。速度/加速度/減速度的設定可被省略，如果在指令中省略了速度/加速度/減速度的設定，則延續 Move 指令之前的設定，例如：

```
Program Main
  TakeArm 0
  Move P,@P,P[0],speed=80  `用 speed 指定速度為 80
  Move P,@P,P[1],S=90      `用 S(大小寫均可) 指定速度為 90
  Speed 50                  `用 Speed 設定速度為 50
  Move P,@P,P[3]            `延續之前的設定，所以速度為 50
  GiveArm 0
End Program
```

插補方法：

運動插補的方法有 P(點對點)、L(直線)及 C(弧線)，詳細說明看表 2-1:

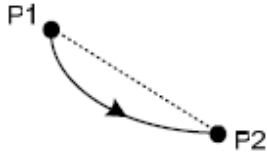

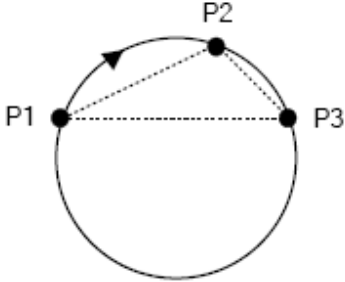
插補方法	說 明
P	<p>採用點對點的控制方式從P1運動到指定的目標位置P2，運動軌跡取決於各軸間之運動，絕大部份是非直線。</p>  <p>舉例:</p> <pre>Move P,@E,P[2],S=100</pre>
L	<p>從目前位置 P1 沿直線運動到指定的目標位置 P2</p>  <p>舉例:</p> <pre>Move L,@E,P[2],S=100</pre>
C	<p>產生一個連接 P1-P2-P3 的圓弧，其中 P1 為起始位置(目前位置)，P2 為中繼點，P3 為目標位置</p>  <p>舉例:</p> <pre>Move C,@E,P[2],P[3],S=100</pre>

表 2-1

到達目標位置確認方法有三種類型，說明看表 2-2:

目標確認	說 明
@0	<p>運動位置到達目標位置就算到達，不須檢測編碼器的值</p> <p>舉例:</p> <pre>Move L,@0,P[2],S=10 Move L,@0,P[3],S=10</pre>
@P	<p>實際位置到達目標位置附近即算到達，且不減速而往下一位置運動</p> <p>舉例:</p> <pre>Move P,@P,P[2],S=10 Move P,@0,P[3],S=10</pre>
@E	<p>檢查編碼器值來判斷是否到達目標位置</p>

	<p>舉例:</p> <pre>Move L,@E,P[2],S=10 Move L,@0,P[3],S=10</pre>
--	---

表 2-2

ArchMove

功能：以凸凹方式移動到指定位置

格式：ArchMove <運動結束方式> , <目標位置> , <上升/下降距離> , <S=速度比例> / <Accel=加速度比例> / <Decel=減速度比例>

說明：從當前位置垂直上升或下降指定距離後再移動到指定的目標位置。速度/加速度/減速度的設定可被省略，如果在指令中省略了速度/加速度/減速度的設定，則使用 ArchMove 指令之前的設定。

舉例:

Program Main

TakeArm 0 'Arm 0 獲得運動控制權限

Move P,@P,P[0],speed=80 '首先運行到P[0]處

ArchMove @0,P[1],10 · S=90

· 用S(大小寫均可)指定速度為90，從P[0]處先垂直上升10mm，再以直線方式運動到P[1]正上方10mm處，再垂直向下運動到P[1]處，ArchMove 整個運動過程中以@E方式插值。

GiveArm 0 Arm 0 釋放運動控制權限

End Program

GMove

功能：優化組合運動軌迹運動

格式：GMove <開啟 / 關閉>

說明：GMove 只保持起始點和結束點@E的方式運動，中間過程運動的@E方式就被忽略

舉例:

Program Main

```

TakeArm 0      ' Arm 0 獲得運動控制權限

GMove 1        ' 開啟 GMove

Move P,@E,P[0] ' 以@E 的方式運動到 P[0]

Move P,@E,P[1] ' 以@P 的方式運動到 P[1]

Move P,@E,P[2] ' 以@P 的方式運動到 P[2]

Move P,@E,P[3] ' 以@E 的方式運動到 P[3]

GMove 0        ' 關閉 GMove

GiveArm 0      ' Arm 0 釋放運動控制權限

```

End Program

DriveA

功能：指定軸運動到指定的距離

格式：DriveA @<到達目標位置確認方式> , <軸號> , <目標位置>

說明：若未指定到達目標位置確認方式方式，則使用@E 的方式

舉例：參考 DriveI

DriveI

功能：指定軸在當前位置以固定的距離運動一段偏移量

格式：DriveI @<到達目標位置確認方式> , <軸號> , <目標位置>

擴展功能: DriveI @<到達目標位置確認方式> , <軸號> , <目標位置> , stopon(1,1,iotype=2)

Stopon(1,1,iotype=2) 解釋: (1 : IO 端口 1; 1 : IO 狀態 HIGH, 可設 0 ; iotype=2 : 為虛擬 IO, 0 為 IPCH 內部 IO, 1 為 RIO.)

例如: DriveI 0,40,stopon(1,1,iotype=1) J1 軸以@E 運行方式, 走 40mm , 在遇到 RIO 端口 1 有信號輸入(1 高電平)後, 停止運行.

說明：若未指定到達目標位置確認方式，則以@E 的方式

舉例:

Program Main

TakeArm 0 'Arm0 獲得運動控制權限

Speed=80

DriveA @0,1,20

'Arm0 的J1 軸以默認的內部速度@0 方式從當前位置運動到 20mm 處

DriveI @E,1,-3 's=50

'Arm0 的J1 軸以內部速度 50%,@E 方式 20mm 處運動到軸座標為 17mm 處

DriveA 1,-3,accel=100

'Arm0 的J1 軸以加速度為 100% ·@E 方式 · 從 17mm 處運動到軸座標為-3mm 處

DriveI 1,2 's=20

'Arm0 的J1 軸以@E 方式從-3mm 處運動到軸座標為-1mm 處

GiveArm 0 Arm0 釋放運動控制權限

End Program

MotorMotion

功能：單軸運動控制

格式：MotorMotion <速度參數>

說明：配合“速度參數”使用，可實現單軸速度控制

舉例：

Program Main

Dim s As SpeedParam

' 定義速度參數變數 s

s.dir = 1

' 1 表示正方向運動，-1 表示反方向運動

TakeArm 0

' Arm 0 獲得運動控制權限

while TRUE

 if GetDI(10)=True and GetDI(11)=False then

' 檢測到DI(10)

 s.speed = 50

' 設置運動速度比例

```
MotorMotion s                                `執行運動指令
elseif GetDI(10)=False and GetDI(11)=True then `檢測到DI(11)
    s.speed = 0                                `設置運動速度比例
    MotorMotion s                              `執行運動指令
End if
Delay 10
wend
GiveArm 0   Arm 0 釋放運動控制權限
End Progr
```

第 3 章 I/O 指令詳解

SetDO

功能：將指定的輸出端口狀態設定成 ON

格式：SetDO <輸出端口位置>

舉例：

Program Main

SetDO 40 *、設定輸出端口 40 為 ON*

Move P,@0,P[1],S=100

End Program

SetDOEx

功能：在運動過程中，當距離目標位置 *distance* 時，指定的輸出端口狀態設定成 ON

格式：SetDOEx(<輸出端口位置> , *distance*)

舉例：

Program Main

TakeArm 0

、運動距離目標位置 10mm 時，設定輸出端口 40 為 ON

Move P,@0,P[1],S=100,SetDoEx(40,10)

GiveArm 0

End Program

ResetDO

功能：將指定的輸出端口狀態設定成 OFF

格式：ResetDO <輸出端口位置>

舉例：

Program Main

SetDO 20 *、設定輸出端口 20 為 ON*

Delay 1000

ResetDO 20 ‧ 設定輸出端口 20 為 OFF

End Program

ResetDOEx

功能：在運動過程中，當距離目標位置 *distance* 時，指定的輸出端口狀態設定成 OFF

格式：ResetDOEx (<輸出端口位置> , *distance*)

舉例：

Program Main

TakeArm 0

‧ 運動距離目標位置 10mm 時，設定輸出端口 40 為 OFF

Move P,@0,P[1],S=100,ResetDoEx(40,10)

GiveArm 0

End Program

GetDI

功能：取得輸入端口的狀態。

格式：GetDI<輸入端口位置>

舉例：

Program Main

Dim status as Integer

status = GetDI(10) ‧ 取得輸入端口 10 的狀態，並且儲存在變量 *status* 中

if status = 0 then

Move L,@P,P[2],S=100 ‧ 當輸入端口 10 的狀態為 OFF，運動到 P[2] 位置

else

Move L,@P,P[1],S=100 ‧ 當輸入端口 10 的狀態為 ON，運動到 P[1] 位置

end if

End Program

Wait

功能：程序在指定的時間內等待某個輸入端口的狀態觸發成預期的狀態。

格式：Wait <輸入端口位置>,<預期狀態>,<等待時間>,<超時狀態>

說明：

(1). <預期狀態>的值可指定 1, 2, 3, 4

1：上升沿

2：下降沿

3：高電平

4：低電平

(2). <等待時間>的單位為秒

(3). <超時狀態>必須傳入整數變量

(4). 等待時間參數可以被省略。當等待時間參數被省略時，程序無限等待直到指定的輸入端口觸發成預期的狀態後，程序才會往下執行

(5). 當等待時間參數沒有被省略，輸入端口在等待時間內觸發成預期的狀態，程序會往下執行，且儲存超時狀態的變數的值為 0。反之，如果輸入端口在等待時間內沒有觸發成預期的狀態，程式也會

往下執行，但是儲存超時狀態的變數的值會變成 1

舉例：

Program Main

Dim response As Integer

Const VIO_READY=20 '定義虛擬IO

Const VIO_FINISH=21

Wait 0,1,response '等待輸入端口0的狀態觸發成下降沿

Wait 1,1,timeout=10,response '等待輸入端口1的上升沿觸發，等待時間為10秒

if response = 1 then '判斷等待是否超時

SetDO(2) '等待超時

else

SetDO(3) '輸入端口1的狀態觸發成OFF，等待未超時

end if

```

wait VIO_READY,HIGH,iotype=2    ' 虛擬 IO 模式要加 iotype=2, IPCH 內部 IO 模式要
                                ' 加 iotype=0 ,RIO 模式是 iotype=1

ReSetDI 21                      ' 復位虛擬 IO
End Program

```

SetDI

功能：將指定的虛擬輸入端口狀態設定成 ON

格式：SetDI <輸入端口位置>

備註： 在使用虛擬 IO 時,要標明 iotype=2

備註：輸入端口號為虛擬 IO[0~23]中的任何一個數字，[0~47]為實體輸入端口號,有擴張板另算.

舉例：

```

Program Main
    SetDI 23                    ' 設定輸入端口 23 為 ON
End Program

```

ResetDI

功能：將指定的虛擬輸入端口狀態設定成 OFF

格式：ResetDI <輸入端口位置>

備註：備註： 在使用虛擬 IO 時,要標明 iotype=2

輸入端口號必須為虛擬 IO[0~23]中的任何一個數字，[0~47]為實體輸入端口號

舉例：

```

Program Main
    SetDI 22                    ' 設定輸入端口 22 為 ON
    Delay 1000
    ResetDI 22                  ' 設定輸入端口 22 為 OFF
End Program

```

GetDO

功能：獲得輸出端口的狀態

格式：GetDO <輸出端口號>

備註：輸出端口號必須為[0~63]中的任何一個數字，有擴展 IO 板則增加數字，如果端口在 I/O 板上不存在，則 GetDO 的返回值為 0

舉例：

```
Program Main
  Dim state As Integer
  state = GetDO 26          ' 將輸出端口 26 的狀態保存變數 state
  print "state=",state     ' state=1 時輸出端口 26 為 ON，為 0 時為 OFF
End Program
```

GetADValue

功能：獲取指定通道的模擬量

格式：GetADValue <通道號>，<超時>

備註：通道號為[0~7]中的任意一個整數，<超時>的單位為毫秒

舉例：

```
Program Main
  Dim value As Double
  value = GetADValue(0, timeout=50) ' 獲取通道 0 的模擬量
  print "value=",value
End Program
```

SetDAValue

功能：在指定的通道輸出模擬量

格式：SetADValue <通道號>，<輸出值>

備註：通道號為[0~5]中的任意一個整數，輸出值是範圍[-10~10]中的任意一個浮點型數值，單位為伏

舉例：

```
Program Main
  Dim value As Double
  value = 5
  SetADValue(0, value)  ' 在通道 0 輸出 5V 電壓
End Program
```


第 4 章 位置指令詳解

PosX

功能：取得點位的 X 座標值

格式：PosX <點位變量>

舉例：參考 PosRZ

PosY

功能：取得點位的 Y 座標值

格式：PosY <點位變量>

舉例：參考 PosRZ

PosZ

功能：取得點位的 Z 座標值

格式：PosZ <點位變量>

舉例：參考 PosRZ

PosRX

功能：取得點位的 a 座標值

格式：PosRX <點位變量>

舉例：參考 PosRZ

PosRY

功能：取得點位的 b 座標值

格式：PosRY <點位變量>

舉例：參考 PosRZ

PosRZ

功能：取得點位的 c 座標值

格式：PosRZ <點位變量>

備註：Cartesian 座標存儲的格式為 (x, y, z, a, b, c, u, v, w)

舉例：

```
Program Main
  TakeArm 0
  Dim X,Y,Z As double
  Dim a,b,c As double
  X = PosX(P[0])
  Y = PosY(P[0])
  Z = PosZ(P[0])
  a = PosRX(P[0])
  b = PosRY(P[0])
  c = PosRZ(P[0]) ' 獲取 P[0,0] 的 c 值
  GiveArm 0
End Program
```

LetX

功能：設定點位的 X 座標值

格式：LetX <點位變量> = <設定值>

舉例：參考 LetRZ

LetY

功能：設定點位的 Y 座標值

格式：LetY <點位變量> = <設定值>

舉例：參考 LetRZ

LetZ

功能：設定點位的 Z 座標值

格式：LetZ <點位變量> = <設定值>

舉例：參考 LetRZ

LetRX

功能：設定點位的 RX 座標值，即是修改 Cartesian 座標的 a 值

格式：LetRX <點位變量> = <設定值>

舉例：參考 LetRZ

LetRY

功能：設定點位的 RY 座標值，即是修改 Cartesian 座標的 b 值

格式：LetRY <點位變量> = <設定值>

舉例：參考 LetRZ

LetRZ

功能：設定點位的 RZ 座標值，即是修改 Cartesian 座標的 c 值

格式：LetRZ <點位變量> = <設定值>

備註：Cartesian 座標存儲的格式為 (x, y, z, a, b, c, u, v, w)

舉例：

```
Program Main
  TakeArm 1
  LetX P[0] = -1
  LetY P[0] = -2
```

```

LetZ P[0] = -3
LetRX P[0] = 10
LetRY P[0] = 20
LetRZ P[0] = 30 `設定P[0,1]的c 值

Print "P[0]=", P[0] `訊息輸出P[0]=(-1.000,-2.000,-3.000,10.000
'20.000,30.000)
GiveArm 1

```

End Program

CurPos

功能：獲得當前 Cartesian 座標位置

格式：CurPos (ArmID)

舉例：參考 CurJnt

CurJnt

功能：獲得當前 Joint 座標位置

格式：CurJnt (ArmID)

舉例：

```

Program Main
  TakeArm 1
  Dim pos As Position
  Dim jt As Joint
  pos = CurPos(1) `將 Arm1 的當前 Cartesian 座標位置保存在 pos 變數
  jt = CurJnt(0) `將 Arm0 的當前 Joint 座標位置保存在 jt 變數
  GiveArm 1
End Program

```

Jnt

功能：獲得 Joint 點位座標的指定軸座標值

格式：Jnt (目標點位<軸號>)

備註：軸號範圍為 0~9

LetJ

功能：設定 Joint 點位座標的指定軸號座標值

格式：LetJ (目標點位<軸號>) = <設定值>

備註：軸號範圍為 0~9

舉例：

```
Program Main
    TakeArm 1
    Dim jt As Joint
    Dim res As Double
    res = Jnt(J[2][4]) ' 將 Arm1 的 J[2] 點 J4 軸的數值保存到 res 變數中
    LetJ J[2][4] = 1.2 ' 將 Arm1 的 J[2] 點 J4 軸的數值修改為 1.2

    jt = J[3]
    LetJ jt[1] = 2 ' 將臨時變數 jt 的 J1 軸的座標數值設定為 2
    GiveArm 1
End Program
```

SavePoint

功能:修改點位列表中的已有點位的信息的指令。

格式:SavePoint <目標點位號> , <與目標點位類型相同的臨時點位/點位列表中的已有點位號>

```
Program Main
    Dim tempCar As Position
    Dim tempJnt As Joint
    TakeArm 1
    Accel 20
    Speed 100
    SavePoint P[3], P[4] ' 用 P[4] 覆蓋 P[3] , P[3] 點位被更新
    Delay 200
```

```
tempCar = CurPos(1)
SavePoint P[3], tempCar `用當前位置的直角座標覆蓋 P[3] · P[3] 點被更新

Delay 200
tempJnt = CurJnt(1)
SavePoint J[1], tempJnt `用當前位置的軸座標覆蓋 J[1] · J[1] 點被更新
GiveArm 1
End Program
```

第 5 章 通訊功能指令

NetSlave

功能：網路客戶端類，用于為客戶端通訊服務的管理

方法：init · start · send · recv, stop

說明：

init：用于客戶端的初始化，包括目標服務器的 IP 地址，目標服務器的端口，超時時間

start: 用于啟動客戶端網路通訊功能函數

send: 客戶端發送字符串函數

recv: 客戶端接收字符串函數

stop: 客戶端停止網路通訊功能函數

建立網路結點 1，服務器 IP 為“192.168.1.101”，服務器端口為 6260

1000 為網路超時時間。

舉例：

Program Main

Dim net_slave As NetSlave，定義一個網路客戶端對象

Dim buffer As String

net_slave.init(1)

用網路結點 1 初始化該客戶端對象，結點 1 的信息自動從數據文件中獲取

在參數界面設定，把客戶端 1 的 IP 地址設：192.168.1.101，網路端口：6260，超時時間設：

1000。

net_slave.start()，啟動客戶端網路通信

while TRUE

buffer = net_slave.recv()，客戶端接收數據，收到的內容保存在 buffer 中

print "Client rece:",buffer

buffer=buffer+"aa"

net_slave.send(buffer)，客戶端發送數據

wend

net_slave.stop()、客戶端停止通訊服務

End Program

NetServer

功能：網路服務器類，用于為服務端通訊服務的管理

方法：init、start、send、recv、stop、addclient

說明：(在使用網路通訊指令時，要在參數介面設置伺服器 and 用戶端參數)

init：用於服務器的初始化，包括目標服務器的 IP 地址，目標服務器的端口，超時時間

start: 用於啟動服務器網路通訊功能函數

send: 服務器端發送字符串函數

recv: 服務器端接收字符串函數

stop: 服務器端停止網路通訊功能函數

addclient：添加客戶端對象

舉例：

Program Main

Dim net_server As NetServer、定義一個網路服務對象

Dim strCmd As String

Dim buffer As string

Id=0

net_server.init(1)、用網路結點 1 初始化該服務對象，結點 1 的信息自動從數據文件中獲取
(在參數設定中，伺服器 1，設置最大的連接數 5，設置網路端口 6260，網路超時時間 10.0)

net_server.start()、啟動服務器網路通信

net_server.addclient("192.168.1.100","ar")

、將 IP 為 "192.168.1.100"，用戶名為 "ar" 客戶端 IP 添加到服務通訊隊列

While true

id=id+1


```
strCmd=str(id)+",0.0,0.0"  
net_server.send(strCmd,'ar') `服務器發送數據  
  
buffer = net_server.recv() `服務器接收數據，收到的內容保存在 buffer 中  
print "server recv:", net_server.recv("ar")  
  
wend  
net_serve.stop() `服務器停止所有通訊服務
```

End Program

SerialPort

功能：串口類，用于為串口通訊服務的管理

方法：init · send · recv

說明：(在使用串口通訊指令時，要在參數介面設置串口參數)

init：用於串口的初始化

send: 串口發送字符串函數

recv: 串口接收字符串函數

在參數設定，串口通訊，設定端口 0，設定：波特率 115200，數據尺寸 8，控制位 s，奇偶校驗位 N，串口名 1，停止位 1.0，超時時間 1000.0。

舉例：

Program Main

```
Dim com As SerialPort `定義一個串口通訊對象  
Dim msg String  
Dim index as integer  
Com.init(0,10) `串口 0 初始化  
  
Com.start(0) `開啟串口接收數據  
index=0
```

```
while true

index =index+1
msg=com.send(str(index)) `串口發送數據

print "sent:",msg


`msg=com.recv(` 串口接收數據 · 收到的內容保存到msgRecv
`if msg<>" "then
`end if
`print"receiver:",msg

wend
End Program
```

第 6 章 文件操作與計時指令

FileSystem

功能：文件類，用於文件讀寫操作

方法：open，read，save，append，read_db，save_db

說明：

open：打開文件，支持兩種文件格式 txt，db，若文件後綴名為 db，則打開數據庫文件，除此之外都將文件以文本文件的方式打開

save：當文件以文本文件方式打開時，對文件進行重寫操作，覆蓋之前的內容

append：當文件以文本文件方式打開時，保留文件之前的內容，在文件尾進行追加操作

read_db：當文件以數據庫方式打開時，讀指定的 key / value，key / value 均是字符串

save_db：當文件以數據庫方式打開時，修改或插入新的 key / value，key / value 均是字符串

舉例：

Program Main

```
Dim fd As FileSystem '定義一個文件操作對象
```

```
Dim key, value, table As String
```

```
Dim content As String
```

```
Dim txtFile, dbFile As String
```

```
txtFile = "record.txt"
```

```
dbFile = "claw.db"
```

```
key = "account"
```

```
value = "$100"
```

```
table = "parameter"
```

```
fd.open(txtFile) '以文本文檔的方式打開 record.txt 文件
```

```
content = fd.read() '讀取 record.txt 的內容到 content
```

```
print "File content=" & content '輸出文件 record.txt 的內容
```

```
fd.append("I am into IT") '在 record.txt 文件尾插入 "I am into IT"
```

```
fd.open(dbFile) '以數據庫操作的方式打claw.db 文件
fd. save_db(value , key , table)
' 將key="account" , value = "$100" 寫入到表名為parameter 到claw.db 文件
' 若account 不存在則是插入 , 存在則是修改
' 若parameter 表不存在則是創建該表
```

```
fd. Save_db(value , key)
' 將key="account" , value = "$100"
' 寫入到表名為default_table 到claw.db 文件
```

```
print fd.read_db(key, table) '輸出表parameter,key=account 的內容
print fd.read_db(key) '輸出表default_table,key=account 的內容
```

```
contont = "update parameter set value='-2' where key='account'"
fd.exe_cmd(content) '將表parameter 的account 的內容更新為-2
```

End Program

Timer

功能：計時類，用於計算某一段 PAC 的執行時間間隔

方法：start , end

說明：

start：計時器開始計時

end：計時器計算最近一次 start 開始到當前時刻的時間間隔，返回值的單位為秒

舉例：

Program Main

```
Dim pacTimer As Timer '定義一個計時器對象
Dim span As Double
TakeArm 0
pacTimer.start() '計時器開始計時
```

```
Move P,@0,P[0] ,S=20
Delay 100
Move P,@0,P[1],S=10
print pacTimer.end()  `輸出計時器開始的時間間隔

pacTimer.start()      `計時器重新開始計時
Move P,@0,P[3] ,S=10
Delay 100
Move P,@0,P[1],S=100
span = pacTimer.end() `計算從P[3]點運動開始到P[1]點結束的時間間隔
GiveArm 0
End Program
```

第 7 章 流程控制指令詳解

if-elseif-else

功能: 根據條件執行指定的指令區塊

格式:

```
if 條件判斷式 1 then
    指令區塊 1
elseif 條件判斷式 2 then
    指令區塊 2
.....
elseif 條件判斷式 n then
    指令區塊 n
else
    指令區塊 n+1
end if
```

舉例:

```
Program Main
    if GetDI(1)=1 then
        print "Execution A"
    elseif GetDI(2)=0 then
        print "Execution B"
    else
        print "Execution C"
    end if
End Program
```

也可以利用邏輯運算子 and、or 合成判斷式

注意:請使用括號()做為區隔

```
Program Main
    if ( GetDI(1)=1 ) and ( GetDI(2)=1 ) then
        print "Execution A"
    elseif ( GetDI(3)= 0 ) or ( GetDI(4)=0 ) then
```

```

        print "Execution B"
    else
        print "Execution C"
    end if
End Program

```

select-case

功能: 多重分支指令，根據變數的資料值，執行 case 內某段的指令區塊

舉例:

```

Program Main
    Dim status As Integer
    select case status
        case 0
            print "Execution A"
        case 1
            print "Execution B"
        case 2
            print "Execution C"
    end select
End Program

```

while

功能: 循環指令，當 while 條件成立時，執行 while 內的指令區塊

格式:

```

while 條件判斷式
    指令區塊
wend

```

舉例:

```

Program main
    while GetDI(0) = 1 ' 條件判斷, 當輸入端口 0 的狀態為 ON, 則執行以下區塊
        Move L,@E,(0,0,0),S=10
        Delay 100
        Move L,@E,(10,10,10),S=10
        if GetDI(0)=1 then

```

```

        break ' 呼叫 break, 可以強制跳離 while 循環
    end if
wend ' 結尾
End Program

```

for

功能: 循環指令，當控制變數數值在規定範圍內，則執行 for 區塊內的指令

格式:

```

for 變數 = 起始值 to 結束值 step 累加值
    指令區塊
next 變數

```

說明:

累加值如果被省略，則累加值默認為 1

舉例:

```

Program Main
    Dim i as integer
    for i = 0 to 10 step 2 ' i 值的變化為 0,2,4,6,8,10
        if GetDI(0) = 1 then
            break ' 呼叫 break, 可以強制跳離 for 循環
        end if
        if GetDI(1) = 0 then
            exit for ' 也可以呼叫 exit for 強制跳離 for 循環
        end if
        SetDO(i)
    next i
End Program

```

do loop

功能: 循環指令，當條件成立時，執行區塊內的指令

格式:

do

指令區塊

loop until 條件判斷式

說明：

do loop 與 while 循環的不同處是 do loop 必定執行區塊一次才開始檢查條件

舉例：

Program Main

do

takearm 0

Move L,@E,(0,0,0),S=10

Delay 100

Move L,@E,(10,10,10),S=10

if GetDI(1) = 1 then

break `呼叫break, 可以跳離do 循環

end if

if GetDI(1) = 0 then

exit do `也可以呼叫exit do 強制跳離do 循環

end if

loop until GetDI(0)=1

Givearm 0

End Program

Goto

功能：跳轉指令

格式：

Goto *lable

指令塊

*lable:

指令塊

舉例：

Program Main

Print "main program run"

*Goto *lable*

Print "will be ignore"

**lable:*

Print "come here"

End program

運行結果：

main program run

come here

And

功能：邏輯與

格式：<A 條件> And <B 條件>

Or

功能：邏輯或

格式：<A 條件> Or <B 條件>

Pause

功能：暫停運行，此指令相當于在控制面板上按下暫停鍵

格式：Pause



Stop

功能：中止運行，此指令相當于在控制面板上按下中止鍵

格式：Stop

舉例：

Program Main

if GetDI(0) then



```
    Pause  `暫停`  
elseif GetDI(1) then  
    Stop   `中止`  
else  
    Move L,@E,(0,0,0),S=10  
End program
```

第 8 章 數學函式指令詳解

sin

功能：正弦函數

格式：sin <弧度值>

cos

功能：餘弦函數

格式：cos <弧度值>

tan

功能：正切函數

格式：tan <徑度值>

asin

功能：反正弦函數

格式：asin <數值>

acos

功能：反餘弦函數

格式：acos <數值>

atan

功能：反正切函數

格式：atan <數值>

exp

功能：指數函數

格式：exp <數值>

log

功能：底數為自然數的對數函數

格式：log<數值>

sqrt

功能：計算及傳回指定數字的平方根

格式：sqrt <數值>

ran

功能：產生亂數

格式：ran <數值範圍>

說明：若數值範圍指定 n，則 ran 將產生大於等於 0 且小於 n 的亂數。若不指定數值範圍，ran 將產生大於 0 且小於 1 的亂數

舉例：

Program Main

Dim PI As Double

Dim Value As Double

*PI = 4*atan(1)* ‧取得圓周率

*Value = sin(90*PI/180)* ‧取得角度 90 度之正弦值，得到 1

```
Value = asin(Value)*180/PI  '取得反正弦值，得到 90
Value = cos(60*PI/180)      '取得角度 60 度之餘弦值，得到 0.5
Value = acos(Value)*180/PI  '取得反餘弦值，得到 60
Value = tan(45*PI/180)      '取得角度 45 度之正切值，得到 1
Value = atan(Value)*180/PI  '取得反正切值，得到 45
Value = exp(1)               '取得數學常數 e
Value = log(Value)           '取得數學常數 e 之對數，得到 1
Value = sqrt(100)            '取得數值 100 的正平方根
Value = ran(10)              '產生大於等於 0 且小於 10 的亂數
Value = ran()                '產生大於 0 且小於 1 的亂數
End Program
```

abs

功能：求整型數值的絕對值

格式：Abs(整型數值)

fabs

功能：求 double 型數值的絕對值

格式：Fabs(double 整型數值)

mod

功能：求整型數值相除後的餘數

格式：Mod(數值)

第 9 章 字串處理指令詳解

Left

功能：取得來源字串中最左邊 N 個字元所形成的子字串

格式：Left <來源字串>,<取樣字元個數>

Right

功能：取得來源字串中最右邊 N 個字元所形成的子字串

格式：Right <來源字串>,<取樣字元個數>

Mid

功能：取得來源字串中中間的子字串

格式：Mid <來源字串>,<取樣起始位置>,<取樣字元個數>

Len

功能：取得來源字串的長度

格式：Len <來源字串>

Asc

功能：取得來源字串的第一個字元的 ASCII 數值

格式：Asc <來源字串>

Str

功能：將數值轉換成字串

格式：Str <數值>

舉例:

Program Main

```
Dim source,substr As String
Dim length,asciivalue As Integer
source_ = "FoxBot"
substr = Left(source,3) 'substr 為 "Fox"
substr = Right(source,3) 'substr 為 "Bot"
substr = Mid(source,2,4) 'substr 為 "oxBo"
length = Len(substr) 'substr 的長度為 4
asciivalue = Asc(substr) 'asciivalue 為 111
substr = Str(asciivalue) '將 asciivalue 轉換成字串
```

End Program

Chr

功能：把整數值轉換為 ASCII 碼

格式：Chr(數值)

Val

功能：將字符轉換成 double 型數值

格式：Val(字符串)

Split

功能：按照指定規則解析字符串

格式：split(字符串, "字符")

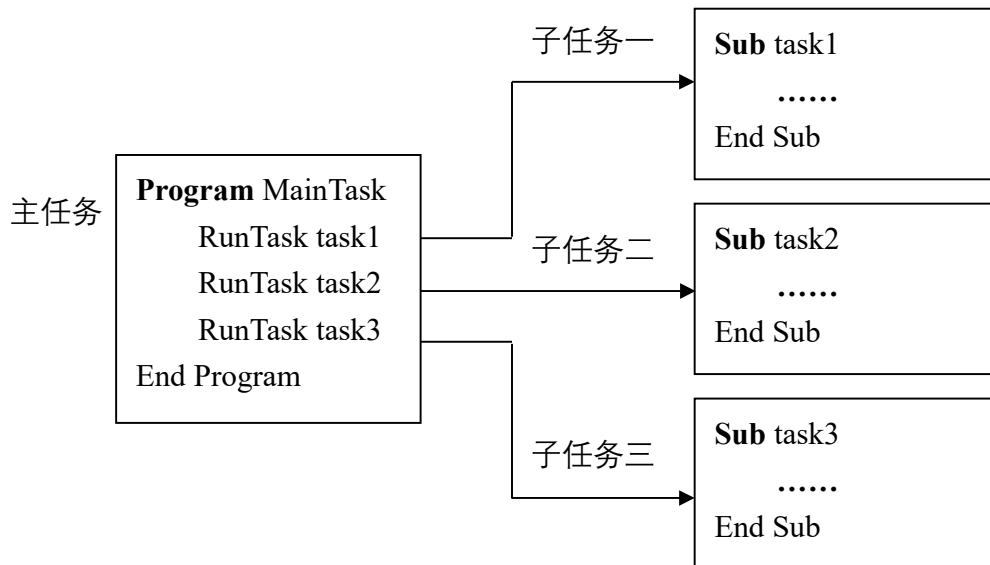
舉例：


```
Dim rec as string  
Dim offset[3] as string  
Rec = "T1,1,1"  
Offset = split(rec,",")
```

運行結束後,offset[0] = "T1", offset[1] = "1", offset[2] = "1"

第 10 章 多任務

多任務(multitask)是一種多工處理的機制。每個專案都會有一個以 Program 宣告的主任務(Main Task)，或稱作主程序。由主任務創建子任務(task)，並且指定任務的執行內容。系統允許最多 8 個任務同時運行。



RunTask

功能：

啟動任務

格式：

RunTask *task_name*, *priority*

參數：

task_name

任務名，請指定已經宣告的 Subroutine。可傳入參數。

priority

任務優先權，設定值的範圍為 1~8，默認為 8，數值越小代表優先權越高。優先權的作用將配合後面的 semaphore 做進一步說明

PauseTask

功能：

暫停任務

格式：

PauseTask *task_name*

參數：

task_name

任務名

ResumeTask

功能：

使已經暫停的任務，繼續執行

格式：

ResumeTask *task_name*

參數：

task_name

任務名

KillTask

功能：

終止任務

格式：

KillTask *task_name*

參數：

task_name

任務名

說明：

- (1). 任務無法用 KillTask 終止自身程序
- (2). 任務可以透過呼叫 stop 來終止自身程序，但不會終止主程序
- (3). 任務可以用 KillTask 終止其它任務，但是無法終止主程序

GetTask

功能：

取得任務目前狀態

格式：

GetTask *task_name*

參數：

task_name

任務名

說明：

該指令之回傳值為一整數數值，其數值與狀態對照表如下

回傳值	系統常數	狀態
0	STOPPED	未執行
1	RUNNING	執行中
2	PAUSED	暫停中
3	PENDING	等待 semaphore 訊號中
4	DELAYING	執行 delay 指令延遲中
5	WAITING	執行 wait 指令等待訊號觸發中
6	ERROR	發生錯誤

未釋放 Semaphore 之前皆必須等待。直到任務 A 釋出控制權，其它等待的任務必須按照順序或者優先權

取得控制權。使用 Semaphore 可以避免多個任務同時對同一個資源進行存取的情況。Semaphore 相關指令

如下：

TakeArm

功能：取得 Motion 控制權

格式：TakeArm 本體編號

說明：

- (1). 每個任務在執行 Motion 指令之前，必須先呼叫 TakeArm 取得 Motion 控制權。
- (2). 如果沒有先以 TakeArm 取得控制權，執行以下命令會產生錯誤。

Move 運動指令

GiveArm

功能：釋出 Motion 控制權

格式：GiveArm

範例：

```
Program Main
    Speed 50
    Accel 50
    RunTask task1          ` 啟動 task1
    while TRUE
        Wait 10,HIGH
        TakeArm 0          ` Arm0 取得Motion 控制權
        Move P,@P,P[0]     ` 移動到P[0]位置
        GiveArm 0          ` Arm0 釋出Motion 控制權
```

```
wend
End Program

Sub task1
  while TRUE
    Wait 20,HIGH
    TakeArm 1      `Arm1 取得Motion 控制權
    Move P,@P,P[1] `移動到P[1] 位置
    GiveArm 1      `Arm1 釋出Motion 控制權
  wend
End Sub
```

第 11 章 其他功能指令

Matrix

功能：已知四頂點定義一個設定名稱，固定行列的矩陣

格式：Matrix name,n,m,p1,p2,p3,p4

Name:定義矩陣名稱

N,m:矩陣的行數，列數

P1,p2,p3,p4 分別為矩陣的四個頂點

舉例：

```
Program main
  Dim matr as matrix
  Dim m,n as integer
  m=5
  n=5
  takearm 0
  accel 50
  decel 50
  Matrix matr,m,n,P[1],P[2],P[3],P[4]
  Move p,@0,matr[3,2],s=50
  Givearm
End program
```

HomeEnable

功能：解除或啟用目標 ARM 的回原點的限制。

格式：HomeEnable <目標 ARM 號>，<TRUE / FALSE>

備註：該指令用在 home.pac 文件中，用于回原點之前的安全檢查

舉例：

```
Program home
  Dim result As Integer
  HomeEnable 0,FALSE `保持 ARM0 回原點禁用狀態，無法進行回原點操作

  Setdo 19
```


、輸出端口 19 置高使汽缸回到其原點位置，輸入端口 21 檢查汽缸是否在其原點位置，為高時
、表示在其原點位置，為低時不在其原點位置

Wait 21,HIGH,2,result 、檢查輸入端口 21 是否為高，2 秒超時等待

If result = 1 then 、2 秒內輸入端口 21 仍然為 OFF，汽缸不在其原點位置

HomeEnable 1,FALSE 、保持 ARM1 回原點禁用狀態，無法進行回原點操作

Elseif result = 0 then 、汽缸是在其原點位置，回原點安全

HomeEnable 1,TRUE 、解除 ARM1 回原點禁用狀態，可以進行回原點操作

End if

End Program

Calibration

功能：將指定的 Arm 所有軸進行原點校正操作。

格式：Calibration <目標 Arm>

Program Main

Calibration(0) 、將 Arm0 的所有軸進行原點校正，也是清零操作

End Program

CalibrationStart

功能：解除目標 Arm 的軟限位約束。

格式：CalibrationStart <目標 Arm>

CalibrationStop

功能：開啟目標 Arm 的軟限位約束

格式：CalibrationStop <目標 Arm>

舉例：

Program Main

CalibrationStart(0) 、將 Arm0 解除軟限位約束，所有軸可進行任意無距離限制移動

Calibration(0) 、將 Arm0 的所有軸進行原點校正，也是清零操作

CalibrationStop(0) 、將 Arm0 開啟軟限位約束

End Program

SetCrabTool

功能：設置工具參數

格式：SetCrabTool <工具長度> <工具角度度>

備註：僅限 Crab 機型有效

舉例：

Program Main

Dim matr1 as matrix

Dim n, m, a, b as integer

SetCrabTool 100,45 、100 和 45 是由“校正小螃蟹工具參數”計算所得

n = 10

m = 10

Matrix matr1,m,n,p[1],p[2],p[3],p[4]

TakeArm 0

for a = 0 to 9

for b = 0 to 9

Move P,@0,matr1[a,b],s=50 、運動到經過校正後的點位

next b

next a

GiveArm 0

End Program

CalPosForCam

功能：計算工具中心實際點位值

格式：CalPosForCam <目標點位> <差值 X> < 差值 Y>

返回值：實際點位值

舉例：

Program Main

```
Dim pos as Position
Dim deltaX ,deltaY as Double
Dim offsetX, offsetY as Double
Dim theta as Double
SetCrabTool 100,45          `100 和 45 是由"校正小螃蟹工具參數"計算所得
theta = 1.0 * 3.1415926/180
offsetX = 20.0              `由 Camera 得到
offsetY = 30.0              `由 Camera 得到
deltaX = offsetY * sin(theta) + offsetX * cos(theta)
deltaY = offsetY * cos(theta) + offsetX * sin(theta)
Pos = CalPosForCam(P[0],deltaX,deltaY)      `計算實際點位 Pos
TakeArm 0
Move P,@P,Pos                          `運動到實際點位 Pos
GiveArm 0
End Program
```

第 12 章 視覺指令

NetSlave

功能：網路客戶端類，用于為客戶端通訊服務的管理

方法：init · vsend · vrecv, trigger

說明：(同第 5 章 NetSlave 指令)

init：用于客戶端的初始化，包括目標服務器的 IP 地址，目標服務器的端口，超時時間

vsend: 客戶端向視覺服務端發送字符串函數

vrecv: 客戶端接收視覺服務端發來字符串函數

trigger: 用于開啓/關閉客戶端 IO 觸發相機功能函數

舉例：

Program Main

```
Dim buffer As String
```

```
Dim oldbuffer As String
```

```
vNet.init(1)
```

```
vNet.trigger(0)
```

```
'set up pattern
```

```
vNet.vsend(PATTERN, GENERIC_GROUP, PATTERN_RECT, PATTERN_NAME)
```

```
' turn on tracking
```

```
vNet.vsend(TRACK, CONTOUR_GROUP, "1")
```

```
vNet.trigger(1)
```

```
Delay 100
```

```
while TRUE
```

```
buffer = vNet.vrecv()
```

```
if buffer <> "" then
```

```
if buffer <> oldbuffer then
```

```
SetTrackTarget buffer
```

```
Print "vision:", buffer
```

```
oldbuffer = buffer
```

```
End if
```

```
End if
```

```
Delay 50
```

```
wend
```

```
vNet.trigger(0)
```

End Program

SetTrackParam

功能：

設定流水線動態抓取所需的相關參數

格式：

SetTrackParam *cam_base*,*motion_dir*,*conveyer_vel*,*min_dis*,*max_dis*,
conveyer_scale,*tool_pos*,*track_time*,*transf*

參數：

cam_base 為 Position 數據類型，在機器人坐標系中，機器人抓取產品的位置與視覺模板位置的差

motion_dir 為 Vector 數據類型，表示機器人在流水綫方向上 2 點之間的向量

conveyer_vel 為流水綫的實際速度

min_dis 機器人抓取產品位置距離相機方向上的距離

max_dis 機器人抓取產品位置相機反方向上的距離

conveyer_scale 編碼器變化值與實際流水綫移動距離的比例關係

tool_pos 為多維數組，機器人吸盤與機器人末端中心位置的關係

track_time 機器人抓取產品時，隨著流水綫跟追的時間

transf 機器人坐標系與視覺坐標系統的方向關係

舉例：見 TrackStart 指令

SetTrackTarget

功能：

用來存儲視覺軟件發過來的數據

格式：

SetTrackTarget *buffer*

參數：

buffer 待存儲的字符串

舉例：見 NetSlave 指令

GetTrackTarget

功能：

獲取 SetTrackTarget 指令存儲的數據

格式：

$vision = \text{GetTrackTarget}(n)$

參數：

n 獲取第 n 組數據

返回值：

$vision$ 為一個 7 個元素的一維數組，分別表示：X，Y，角度，拍照時間，編碼器值，狀態，編號

舉例：見 TrackStart 指令

GetNextVision

功能：

與 GetTrackTarget 指令功能類似，獲取 SetTrackTarget 指令存儲的數據，主要用來在抓取動作之前一次性獲取一批次視覺數據，做好抓取準備

格式：

$vision = \text{GetNextVision}()$

返回值：

$vision$ 為一個 7 個元素的一維數組，分別表示：X，Y，角度，拍照時間，編碼器值，狀態，編號

舉例：見 TrackStart 指令

TrackStart

功能：

機器人抓取動作

格式：

$\text{Angle}[m] = \text{TrackStart } offset, tool_pos[m], turn \cdot \text{SetDOEx}(io_port, distance)$

參數：

offset 機器人從當前位置上移的高度

tool_pos[m] 指定機器人某個吸盤與機器人末端中心位置的關係

turn 當值為 1 表示機器人在抓取產品不作旋轉，當產品實際角度大于 90 度，

$\text{Angle}[m]=180.0$ ，小于-90 度， $\text{Angle}[m]=-180.0$ ，其它情況 $\text{Angle}[m]$ 均為 0.0；當值為 0 或缺省，表示機器人會根據實際產品的角度在抓取時作旋轉，并且 $\text{Angle}[m]$ 值均為 0.0

SetDOEx(io_port, distance) 見 SetDoEx 指令說明

GetobjectPos

功能：

獲取機器人運行的目標點位。

格式：

```
objectpos= GetobjectPos( clawindex,p[7],Vision1[n,4],Vision1[n,0],Vision1[n,2],wait-time)
print "object:("objectpos[0]," " · objectpos[1]," " · objectpos[2],") · wait_time : " · wait_time
move_to_wait_pos(objectpos[0], objectpos[1], objectpos[2], Vision[n,2], clawindex)
```

舉例：

```
Sub track_proc()
    Dim cam_base As Position
    Dim motion_dir As Vector
    Dim tool_pos[2,4] As Double
    Dim transf[4] As Double
    Dim conveyer_vel As Double
    Dim conveyer_scale As Double
    Dim track_time As Double
    Dim n,m,i As Integer
    Dim vision0[7], vision[7] As Double
    track_time = 0.30
    conveyer_vel = 160.0
```

```

conveyer_scale = 62.019291
motion_dir = (344.6850,7.6800,2.5285)
tool_pos[0] = (0.0,43.0,0.0,0.0)
tool_pos[1] = (0.0,-43.0,0.0,0.0)
transf = (1.0,0.0,0.0,1.0)

TakeArm 0
Speed 100
Accel 100
Decel 100
cam_base = P[7]
takegasoff(0)
LetX cam_base = -646.5
LetY cam_base = 392.0
LetZ cam_base = 220.0

SetTrackParam cam_base, motion_dir,
conveyer_vel,-70,200.0,conveyer_scale,tool_pos,track_time,transf
DriveA 2,15.0

while TRUE
    ArchMove @E,P[7],-25,S=50
    For i = 0 to 1
        vision0 = GetNextVision()
        Print vision0[0],"", vision0[1],"", vision0[2]
    Next i
    for n = 0 to 1
        vision = GetTrackTarget(n)
        if vision[5] = 1 then
            Print "Image NG or Missing"
        else
            m = left_or_right(vision[1])
            TrackStart -30.0,tool_pos[m],SetDOEx(val_on[m],10.0)
        End if
    next n
    ArchMove @E,P[12],-25,S=50
wend
GiveArm 0
End Sub

```