X-Y　Table 與 DS Actuator of Slider 控制

A.X-Y　Table 數值控制平台

一、 實驗題目舉例

X-Y　Table 數值控制平台

1.請使用麥克筆與紙在 XY 平台上繪出'十'字

2.讓平台以快慢快的速度作直線運動

3.請使用麥克筆與紙以 60 秒的時間在 XY 平台上繪出一半徑 3 公分的圓

DS Actuator of Slider 控制

1. 連結馬達向右前進三秒（時速約 3㎜/sec）再向左後退三秒（時速　約 3㎜/sec）, 三次後停止

2. 連結馬達向右前進一秒（時速約 3㎜/sec)，停止二秒，再向右前進二秒（時速約 3㎜/sec)，再向左後退三秒（時速約 3㎜/sec）後停止

3. 連結馬達向右前進二秒（時速約 3㎜/sec）， 再向左後退四秒（時速約 3㎜/sec)， 二次後停止

4. 連結馬達向右前進一秒，停止兩秒，共前進三次後停止

5. 連結馬達向左前進四秒（時速約 3㎜/sec)，停止三秒，再向右退後三秒（時速約 3㎜/sec）, 三次後停止

6. 連結馬達向左前進，每前進一秒後停止兩秒，共前進三次後停止，並向右前進至原出發點

7. 連結馬達向左前進一秒（時速約 3㎜/sec)，停止二秒，再向左前進二秒（時速約 3㎜/sec)，二次後停止

8. 連結馬達向左前進三秒（時速約 3㎜/sec），停止二秒，再向右前進四秒（時速約 3㎜/sec）後回到原出發點

進階實驗題目舉例

1. 驅動線性自動滑軌承載螢幕及影像擷取裝置在透鏡後移動，以影像處理裝置擷取影像並分析之，當紅、藍色定位影像最清晰時，紀錄四角定位之中心點座標之位置。
2. 驅動線性自動滑軌承載螢幕及影像擷取裝置在透鏡後移動，以影像處理裝置擷取影像並分析之，當光束直徑最小時，紀錄半透明螢幕之位置、光束影像直徑與光束中心點位置。
3. 驅動線性自動滑軌承載螢幕及影像擷取裝置在透鏡後移動，以影像處理裝置擷取影像並分析之，當四角定位影像最清晰時，紀錄四角定位之中心點座標之位置。
4. 驅動線性自動滑軌承載螢幕及影像擷取裝置在透鏡後移動，以影像處理裝置擷取色差影像並分析之
5. 驅動線性自動滑軌承載螢幕及影像擷取裝置在透鏡後移動，以影像處理裝置擷取球像差影像並分析之
6. 驅動線性自動滑軌承載螢幕及影像擷取裝置在透鏡後移動，以影像處理裝置擷取畸變像差影像並分析之

二、實驗目的

本實驗在於介紹如何使用 NC CODE 及 BCB 來控制 XY TABLE。使用 NC CODE 來控制 XY TABLE 是目前常用的一種方式，也是非常簡單的一種方式。過程中同學將學習到如何編寫 NC CODE 程式來對 XY TABLE 進行控制，並熟悉目前 NC CODE 的一些常用的格式與 XY TABLE 的控制。

三、實驗儀器

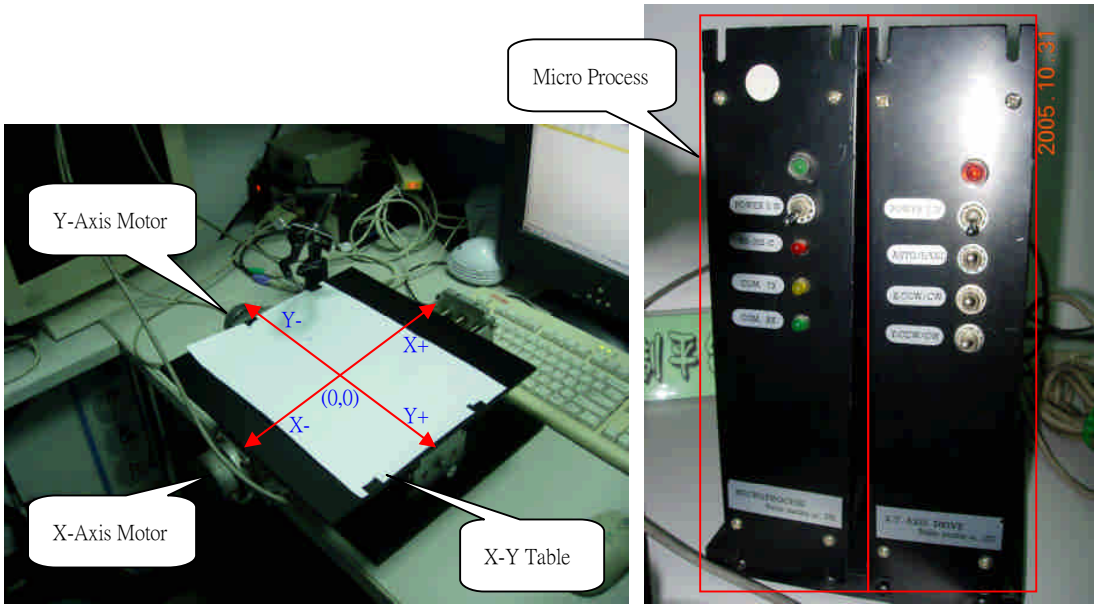
XY TABLE 平台
步進馬達驅動器與微處理器
個人電腦
麥克筆與紙

四、 實驗原理

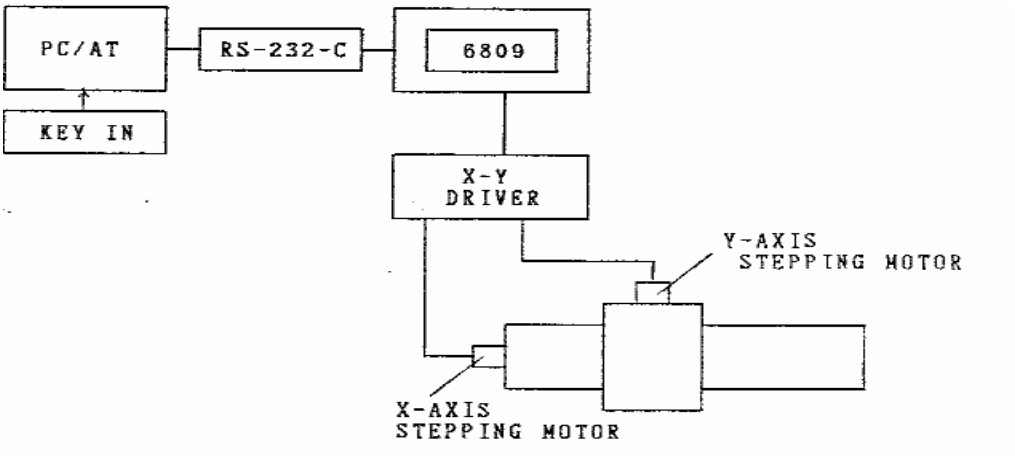
TC-4017(X-Y 平台定位模組)控制系統，係一開放迴路控制(OPEN LOOP CONTROL)。本模組提供使用者操作機台的軟件(SOFTWARE)和微處理器(MICROPROCESS)經過標準通信介面傳輸 RS-232-C 與個人電腦連線，所以使用者可以在個人電腦上規劃機台(TABLE)移動的路徑和操作方式。一般稱為編輯 NC(數值控制)程式。這些數據(NC 程式)經由個人電腦內 RS-232-C 傳輸至微處理器。微處理器內部有 CPU(6809 系列)和 RAM-ROM-I/0 介面……等，解讀由 RS-232-C 收到的串列數據，並且執行機能碼(G-CODE)已經規劃完成的機械移動路徑數學方程式，而使 X 軸或 Y 軸做單軸移動或雙軸同時移動的機械位移。微處理器每次輸出一個脈衝(PULSE)信號，可以使步進馬達(STEPMOTOR)旋轉 1．8degree，所以經由軸連接器連接滾珠導螺桿(BALLSCREW)而驅動滑台(SLIDER)移動，這些驅動機械位移的脈衝信號皆由人為的電氣信號所控制。模組 TC-4017 提供使用者一個安全操作機械設備和研習機械與電氣控制整合的環境。

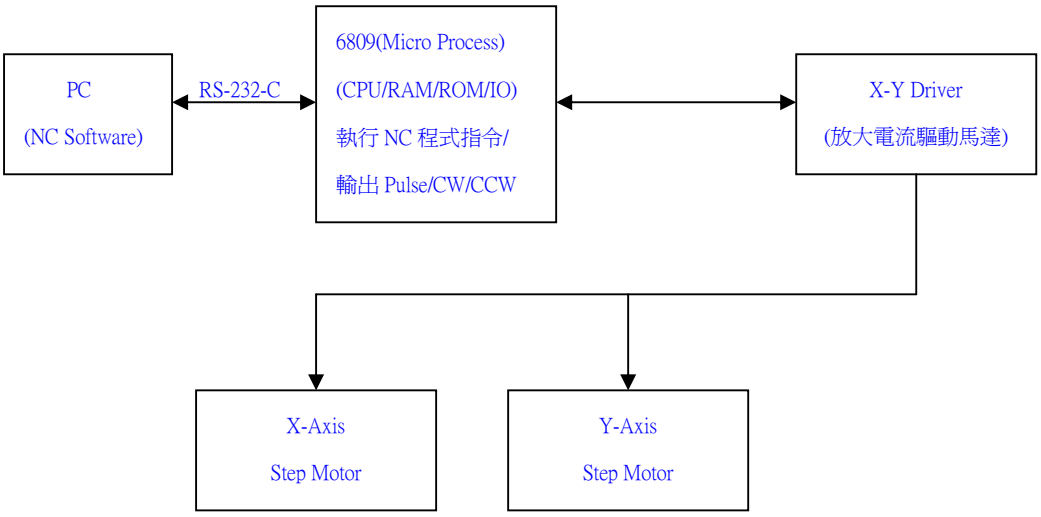
1-1：系統控制方式(OPEN LOOP CONTROL)說明

本模組 TC-4017 係開放迴路控制系統(OPEN LOOP CONTROL)，使用個人電腦做 NC 程式的編輯，經過 RS-232-C 傳輸至微處理器，執行 G-CODE 機能碼的特定軌跡路徑。



系統方塊圖如下列:



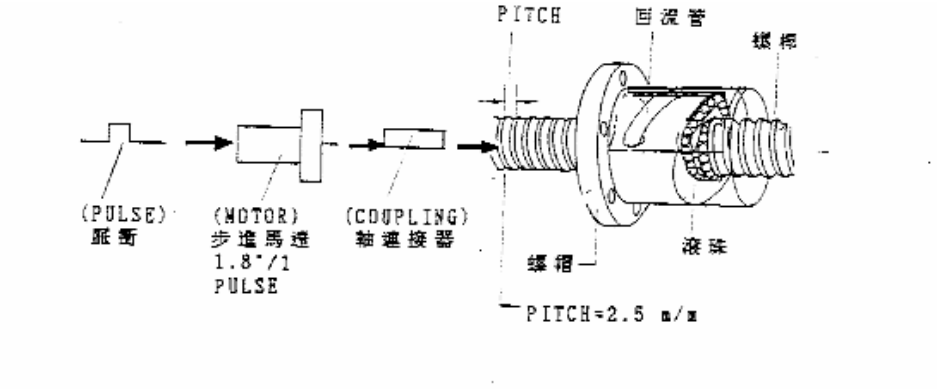


使用者由軟件的執行檔（X-Y AXIS CONTROL）操作 X 軸或 Y 軸的滑台移動，做各種不同的機械位移路徑規劃。本模組軟件皆由 TURBO-C 語言規劃和編輯而成的。在軟件程式結構編寫有三種基本流程提供使用者參考:

1. 循序結構---指程式編寫時，是依循序前進一個接一個執行指令，直到沒有指令可執行為止。
2. 選擇結構---編寫選擇結構是要讓程式具有判斷的功能，選擇合乎條件的程式段來執行。
3. 循環結構---為了讓程式能夠反覆執行某一特定的工作，程式必須設計

一迴路控制。

當使用者操作軟件做機台位移量控制的時候，必須具備一些機電結合的概念:



200 PULSE X1.8°= 360°= 1 REV =2.5m/m 上式可以了解旋轉一圈 360°的滾珠導螺桿上所帶動的滑台位移 2·5 m/m, 因此每一個 PULSE 驅動滑台的位移量為 0.0125m/m=0.01m/m 亦稱為機械位移的解析度(RESOLUTION)。

1-2：PC/AT 或 PC/XT 與微處理器之間的通信協定

個人電腦和微處理器通信協定如下述 ：

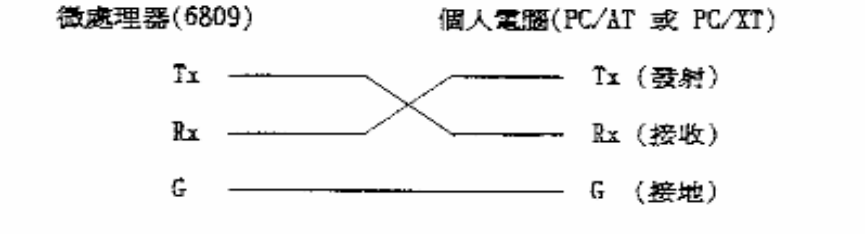
(1)傳輸速率(BAUD RATE)--------- 2400

(2)同位核對(PARITY CHECK)------- (NULL)

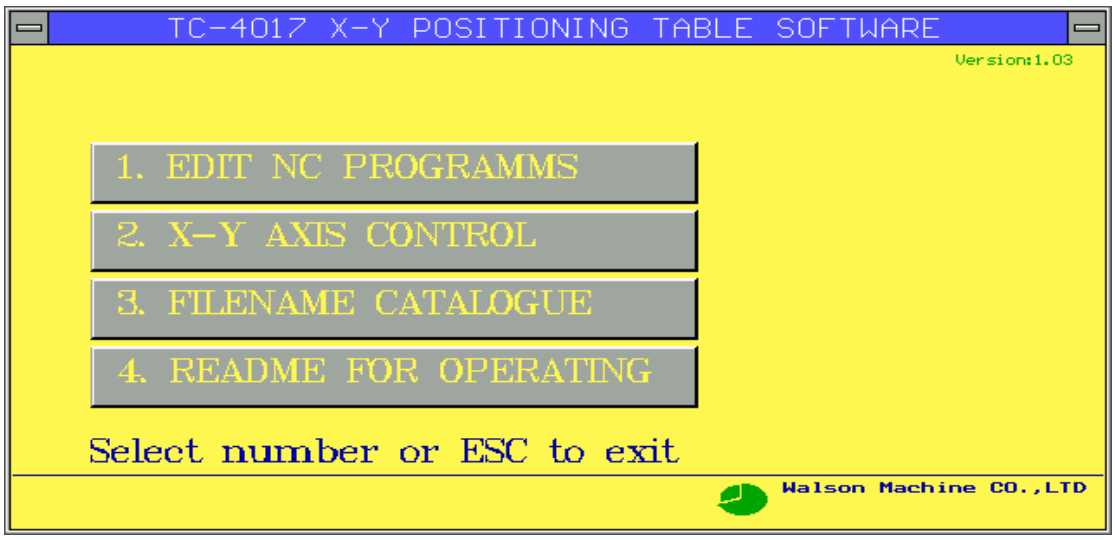
(3)停止位元(STOP BIT)---------- 1

(4)傳送位元(BIT SETS)---------- 8

在 RS-232-C 標準傳輸介面的資料串列輸送互相接線如下 ：



1-3：軟件 (SOFTWARE)控制說明



(1)EDIT NC Program ----編輯 NC 程式檔，此檔提供使用者交談式的 NC 程式編輯。

(2)X-Y AXIS Control ----此擋為執行機械位移的方式，有(I)寸動位移(2)連續位移方式，和滑台移動速度設定，程式參考點設定.....等。

(3)Filename Catalogue----使用者由此檔可以查看檔案內所儲存的 NC 程式的檔名。

(4)Readme for Operating----使用者由此檔案內可以知道本模組的零組件規格和軟件內部功能鍵的用途。

1-4：微處理器(面板控制操作)說明

使用者若是聯接各種背面連接器時，務必將電源開闊(POWER S.W) 往下按(關掉電源)若是與個人電腦聯線工作完成後,RS-232-C 的指示燈會一直亮著。當使用者操作軟件(SOFTWARE)時，選擇總檔案(main meau)的第二項 X-Y axis control 表示此時由軟件做電腦聯線工作，若是聯線工作完成，則 COM.Tx(黃色 LED)會開始閃爍。同時使用者可以由電腦的監視器(MONITOR)上觀察到有二組數據出現 X=000.00 和 Y=000.00，和一組滑台進給率的數據出現 FEED RATE=0040 mm/min 等。此刻使用者可以依照操作手冊的步驟執行機台的檢測工作。

2-1:如何使用軟件編輯 NC 程式

使用者必須在主目錄(MAIN MENU)中選擇第一項 EDIT NC program，然後進入 NC 程式的編輯檔內。使用編輯黨內的功能可分為二類：

第一類：編輯新輸入的 NC 程式；第二類：修改已經存在的 NC 程式。這兩類方式最大的差別在於修改已經存在的 NC 程式中必須呼叫檔案庫(主目錄中第三項 FILENAME CATALOG)，依照使用者指定檔名的 NC 程式會自動依序列出於編輯檔中，然後再修改 NC 程式。首先使用者必須了解 NC 程式編輯檔於畫面顯示的各種功能鍵的應用和各小塊光棒區域警示語句的用途，分別敘述如下:

A. N，G，X，Y，R，F----------此列各字母代表意義：

N----表示 N=01，02，03……..99 等 N 個列。

G----表示輸入 G-CODE 機能碼代碼，例如 G00，G01，G02…..等。

X----表示 X 軸輸入的數值，例 X=001.347 m/m。

Y----表示 Y 軸輸入的數值，例 Y:002.34 m/m。

F----表示滑台移動的進給率(FEEDRATE)。

R----表示圓弧軌跡的半徑（RADIUS）。

B. Diagnostic-----此列含有多塊光棒區域，對於使用者目前填寫進入暫存區的數值(最下端一列)有問題時顯示警告之意，必須更改再次填入。

N 值-------表示使用者利用鍵盤上的方向鍵 T 或 J 移動光棒於列表區域內(N01，N02…….N99 等)。光棒(Highlight-BAR 所指出的位置點(某一列)等於 Diagnostic 中的 N 值。

DATA-ERROR-----表示填入暫存區內的 X 軸移動範圍或是 Y 軸…..等。若是超過機台模組控制的軟件所設定的範圍，則會出現 DATA-ERROR 警示使用者，必須取消已經填入的數值，更改新的數值輸入。其各項數值範圍如下表：

G-CODE.....…G00，G01，G02，G03，G50

X-AXIS…...+-200m/m

Y-AXIS.....+-60m/m

R-(RADIUS).....2<R<100m/m

F-(FEEDRATE)...10<F<600m/m

若填入數值超過以上所列範圍，電腦會自動顯示 DATA-ERROR，必須取消原先填入的數值(請按 ESC 取消已經在暫存區內某項的數值)。

N=HL - BAR ----- 表示使用者利用鍵盤上的方向鍵來移動光棒(HL-BAR)選擇所需要的 N 值。以便做新的 NC 程式編輯或移動光棒至所需要修改的 NC 程式中的那一列以便修改 NC 程式。

C. NC-FILENAME--表示 NC 程式需要一檔名(限 12 個英文字之內，不必有副檔名)以利 NC 程式儲存或呼叫， 所以當使用者新建立 NC 程式的時候，按 F5 後在暫存區內會出現 INPUT 檔名，若鍵入新建立 NC 程式的檔名後，在 NC-FILENAME 之空格區會出現檔名。

D. NC-FILE(OP/CL)----表示目前列表區域內之 NC 程式檔是 OPEN 或 CLOSE 中，使用者必須於 FILEOPEN 時才能夠輸入新的 NC 程式或修改。

使用者明瞭上述編輯檔的各項功能鍵後，能夠自行操作本軟件而做下列二項工作:

甲、建立新的 NC 程式首先按 F5:EDIT 後，在 DIAGNOSTIC 區域內最下端一列會出現 INPUT:_____，請輸入使用者希望的檔名(限 12 個英文字之內，不必有副檔名)，再依照下列流程處理:如此依序建立 NC 程式於列表區域(N01，N02，N03……N99 等)。再利用 SAVE(儲存)指令儲存 NC 程式於磁片中做永久保存，或再次呼叫出來於列表區域內做修改 NC 程式用。

乙、 修改 NC 程式

當使用者希望更改某些機械位移的軌跡而必須修改 NC 程式的某一列或數列 NC 程式的時候。首先必須找出此 NC 程式的檔名，然後按 F3：LOAD 輸入此檔名則編輯檔的列表區域內會自動列出此 NC 程式全部的程式。使用者利用 HL-BAR 移動光棒尋找需要更改或取消的那列程式，再由 FUNCTION KEY 執行(1)取消(F8:DELETE)或(2)插入(F7:INSERT)新鍵入的一列 NC 程式……等。

2-2：如何使用軟件操作 X 軸和 Y 軸

●注意:將微處理器經由 RS-232-C 與個人電腦聯線完成後，才能夠執行主目錄(MAIN MENU) 中的第二項 X-Y AXIS CONTROL。

使用者於主目錄中鍵入"2"項(X-Y AXIS CONTROL)電腦會自動執行聯線工作，若是聯線工作完成，即刻在個人電腦監視器(MONITOR)上會自動顯示出:

X=000.00m/m

Y=000.00m/m

F=0040.m/m

此 X 軸參考點數值和 Y 軸參考點數值及 F 數值(滑台移動速率)皆由微處理持續地傳送(經由 RS-232-C 介面傳輸)給個人電腦而在監視器上顯示出來，若是使用者在個人電腦上利用方向鍵而驅動滑台(鍵入新的 X 軸或 Y 軸位移數據經由 RS-232-C 傳送給微處理器)。這時刻微處理器接到此數據時，依照內部韌體(HARDWARE)程式判斷是否能夠有效執行的數據而加以處理，一方面驅動步進馬達而移動滑台，另一方面將此可以接受的

數據傳迴給個人畫腦由監視器上顛示出

X=010.32m/m

Y=010.34m/m

表示X軸已經由X=000.00m/m移動10.327而Y軸亦往正的軸向移動10.347 m/m等。

X軸和Y軸在本模組軟件驅動上可分為三類:(1)寸動控制(SINGLE CONTROL)(2:連續控制(CONTINUE CONTROL)(3)執行NC程式控制等。分別敘述如下:

(I)寸動控制-----目地是讓使用者能夠很緩和地移動滑台(每按一次鍵可以移動滑台位移量0・02m/m至使用者希望滑台停駐的位置點。使用者可以按F2:s/c功能鍵來選擇S(SINGLE CONTROL)或C(CONTINUE CONTROL)在此段說明假設使用者選擇S(SINGLE CONTROL)寸動控制來執行兩軸的滑台移動方式。此時F=0040mm/min亦是當使用者選擇SINGLE CONTROL時候，軟件就自動定滑台移動速率每分鐘(MIN)移動40m/m以下的速度進行。

(2)連續控制-----目地是讓使用者能夠很快地移動X軸或Y軸滑台接近使用者希望滑台停駐的位置點，再利用寸動控制緩慢地達到正確位置點。其滑台連續移動速率最低為40mm/min，最高為600mm/min。最佳的快速移動速率為450 mm / min。

(3)執行NC控制程式---首先使用者必須按F5:LOAD FILE呼叫一檔名內有已經編輯好的NC程式。假設使用者在X-Y軸的TABLE上放置一張預備畫圖的紙（假設 X-Y TABLE已經放置在一筆架之下)，當使用者移動X軸或Y軸同時使筆架上的筆尖端觸及畫圖紙，即能夠描繪出X軸或Y軸滑台移動軌跡的位移量於畫圖紙上。當使用者利用連續控制方式快速移動滑台或寸動控制方式移動滑台至程式參考點(預估兩軸的滑台移動行程不能超出編輯NC程式的X軸或Y軸位移量)。然後按F8: SET X軸=0 或按F9:SET Y軸=0 此刻電腦即知TABLE在X軸和Y軸上的位置亦稱為程式原點。

2-3:編輯檔功能鍵(FUNCTION KEY)說明

F2:EXIT-------返回主畫面。

F3:LOAD------鍵入NC程式儲存於檔案申的任一檔名，電腦會呼叫此檔名的NC程式於編輯檔中的列表區域內。

F4:SAVE------將編輯完成的 NC 程式存入磁碟檔案中。

F5:EDIT-------編輯新建的 NC 程式檔案必需輸入新建 NC 程式檔名。

F6:WRITE-----新建 NC 程式或修改 NC 程式，當每次要輸入新的數值時皆要按 F6:WRITE，才能夠把數值填入 DIAGNOSTIC 區域內檢查是否有效數值，再按 ENTER 鍵把整列各項數值從暫存區域轉入列表區域內。

F7:INSERT----移動光棒(HIGHLIGHT BAR)選擇您所需要輸入的那一列位置後，按 F7:INSERT，則電腦會自動將那一列之後的 NC 程式往後 N+l 位置順移一位，以利暫存區域那列程式填人列表區域內。

F8:DELETE---移動光棒至您所需要取消的那一列 NC 程式，按 F8:DELETE，則電腦會將那一列 NC 程式自動取消。

F9:PRINT-----若您要將目前在畫面所編輯的新建 NC 程式或舊有 NC 程式列印，請按 F9:PRINT，則列表機會自動列印出您所需要的檔名所屬的 NC 程式。

2-4:執行檔工能鍵說明

Fl:EDIT----------返回 NC 程式編輯檔畫面。

F2:SINGLE/CONTINUE---選擇方向鍵控制的方法

(1)寸動控制
(SINGLE CONTROL)
(2)連續控制
(CONTINUE CONTROL)

F3:FEED DOWN-------滑台移動速度轉慢，調整 FEED RATE 數值。

F4:FEED UP----------滑台移動速度轉快，調整 FEEE RATE 數值。

F5:LOAD FILE--------呼叫儲存於磁片之 NC 程式。

F6:MAIN MENU-------返回主畫面 。

F7:RUN PROGRAMS---執行已經 LOAD 完成的 NC 程式而驅動平台做機械位移運動。

F8:SET X=0----------移動 X 軸滑台至您所需要的機械位移參考點:SET X 軸=0。

F9:SET Y=0----------移動 Y 軸滑台至您所需要的機械位移參考點:SET Y 軸=0。

F10:HELP ----------讀取 README 內有關本模組 TC-4017 的規格說明和功能特性等參考資料。

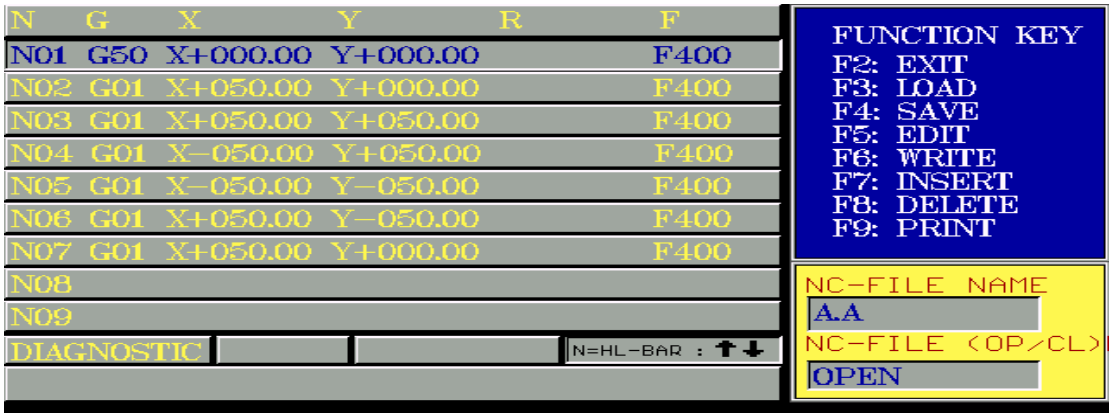
2-5:G-CODE 機能碼說明

G01:快速直線運動。
G0I:直線位移指令。
G02:圓弧指令(CW)。
G03:圓弧指令(CCW)。
G50:程式原點。

五、 實驗步驟

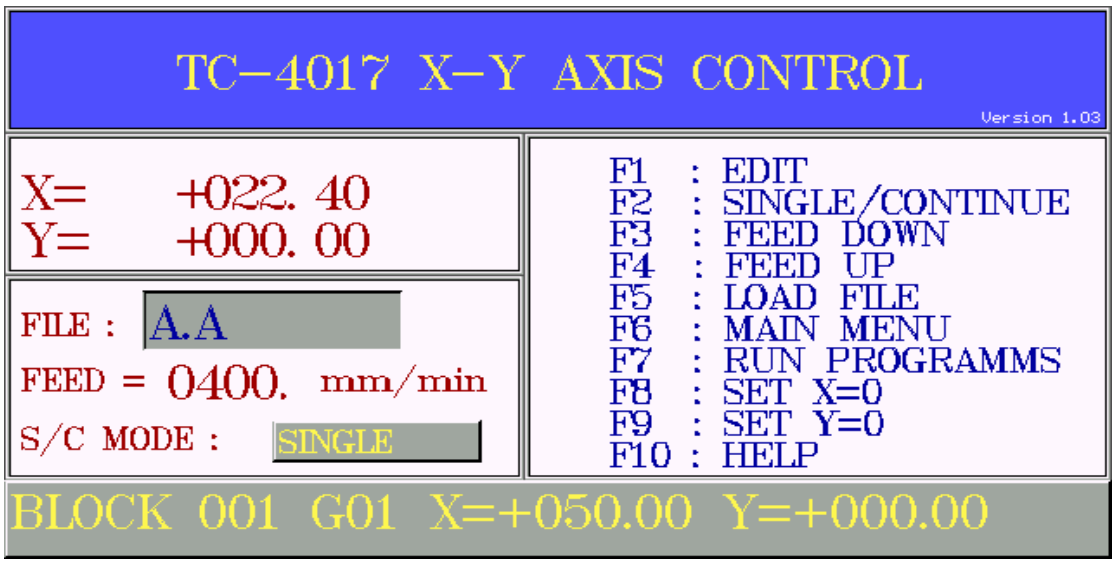
1. 確定 X-Y Table 與電腦連線。
2. 執行 Table 程式。
3. 進入 1 EDIT NC Program。(強烈建議使用 edit 來編輯，使用應用軟體所附的編輯器容易當機)
4. 將以下程式載入：

```
N01 G50 X+000.00 Y+000.00          F400
N02 G01 X+050.00 Y+000.00          F400
N03 G01 X+050.00 Y+050.00          F400
N04 G01 X-050.00 Y+050.00          F400
N05 G01 X-050.00 Y-050.00          F400
N06 G01 X+050.00 Y-050.00          F400
N07 G01 X+050.00 Y+000.00          F400
```



5. 儲存本檔案並離開 EDIT 視窗，進入 2 X-Y AXIS CONTROL。

6. 載入程式，並按 F7 RUN，則 X-Y Table 會跑出一個正方形。



NC 程式範例

1. 三角形

```
N01 G50 X+000.00 Y+000.00          F400
N02 G01 X+050.00 Y+000.00          F400
N03 G01 X+050.00 Y+050.00          F400
N04 G01 X-050.00 Y+000.00          F400
N05 G01 X+050.00 Y-050.00          F400
N06 G01 X+050.00 Y+000.00          F400
N07 G01 X+000.00 Y+000.00          F400
```

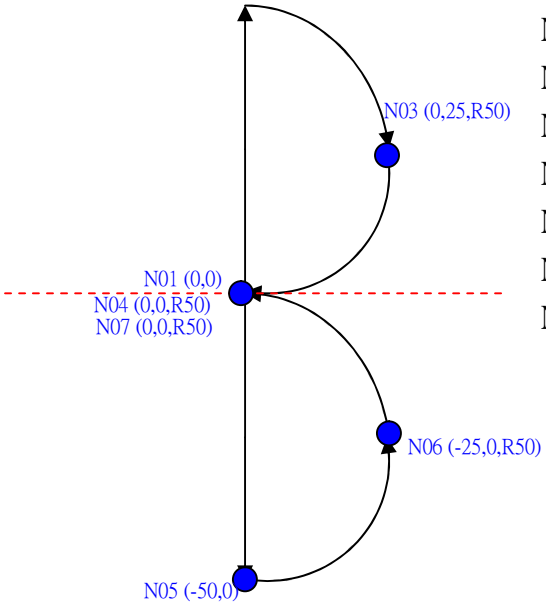
2. 圓形

```
N01 G50 X+000.00 Y+000.00           F400
N02 G01 X+050.00 Y+000.00           F400
N03 G02 X+000.00 Y+050.00 R+050.00 F400
N04 G02 X-050.00 Y+000.00 R+050.00 F400
N05 G02 X+000.00 Y-050.00 R+050.00 F400
N06 G02 X+050.00 Y+000.00 R+050.00 F400
```

3. POP 字母 M 的撰寫（XY02.DMO）

```
N01 G50 X+000.00 Y+000.00          F120
N02 G02 X-005.00 Y+005.00 R+020.00 F400
N03 G02 X-020.00 Y+005.00 R+020.00 F400
```

```
N04 G02 X-020.00 Y-025.00 R+020.00 F400
N05 G02 X-005.00 Y-025.00 R+020.00 F400
N06 G02 X-005.00 Y-010.00 R+020.00 F400
N07 G02 X+000.00 Y-015.00 R+020.00 F400
N08 G02 X+005.00 Y-010.00 R+020.00 F400
N09 G02 X+005.00 Y-025.00 R+020.00 F400
N10 G02 X+020.00 Y-025.00 R+020.00 F400
N11 G02 X+020.00 Y+005.00 R+020.00 F400
N12 G02 X+005.00 Y+005.00 R+020.00 F400
N13 G02 X+000.00 Y+000.00 R+020.00 F400
```




```
N01 G50 X+000.00 Y+000.00                F400
N02 G01 X+050.00 Y+000.00                F400
N03 G02 X+000.00 Y+025.00 R+050.00       F400
N04 G02 X+000.00 Y+000.00 R+050.00       F400
N05 G01 X-050.00 Y+000.00                F400
N06 G03 X-025.00 Y+000.00 R+050.00       F400
N07 G03 X+000.00 Y+000.00 R+050.00       F400
```

B. DS Actuator of Slider 控制

一、實驗目的:

a. 本實驗藉 Window Software(BCB Software)透過 RS-232-C 傳送 Command 至 IAI DS Controller 來控制 IAI X-Y Table 動作

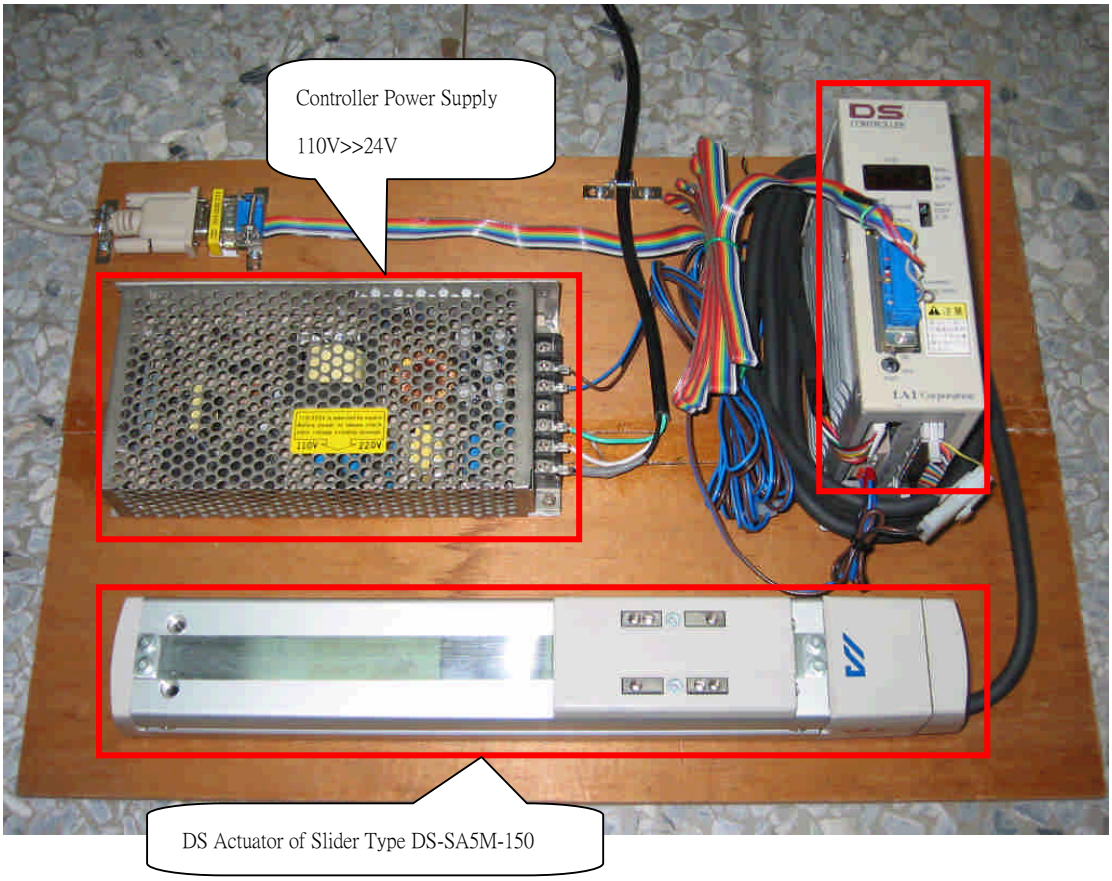
二、實驗儀器

a. DS Controller DS-S-C1
b. DS Actuator of Slider Type DS-SA5M-150
c. Power Supply

d. 電腦+BCB Software

DS Controller DS-S-C1

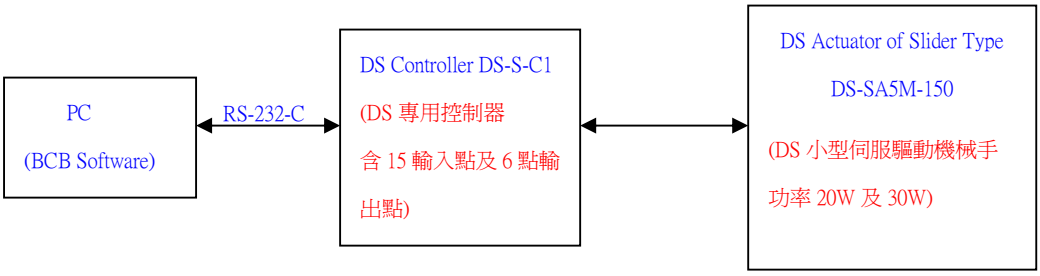
DS Controller DS-S-C1 & DS Actuator of Slider Type DS-SA5M-150

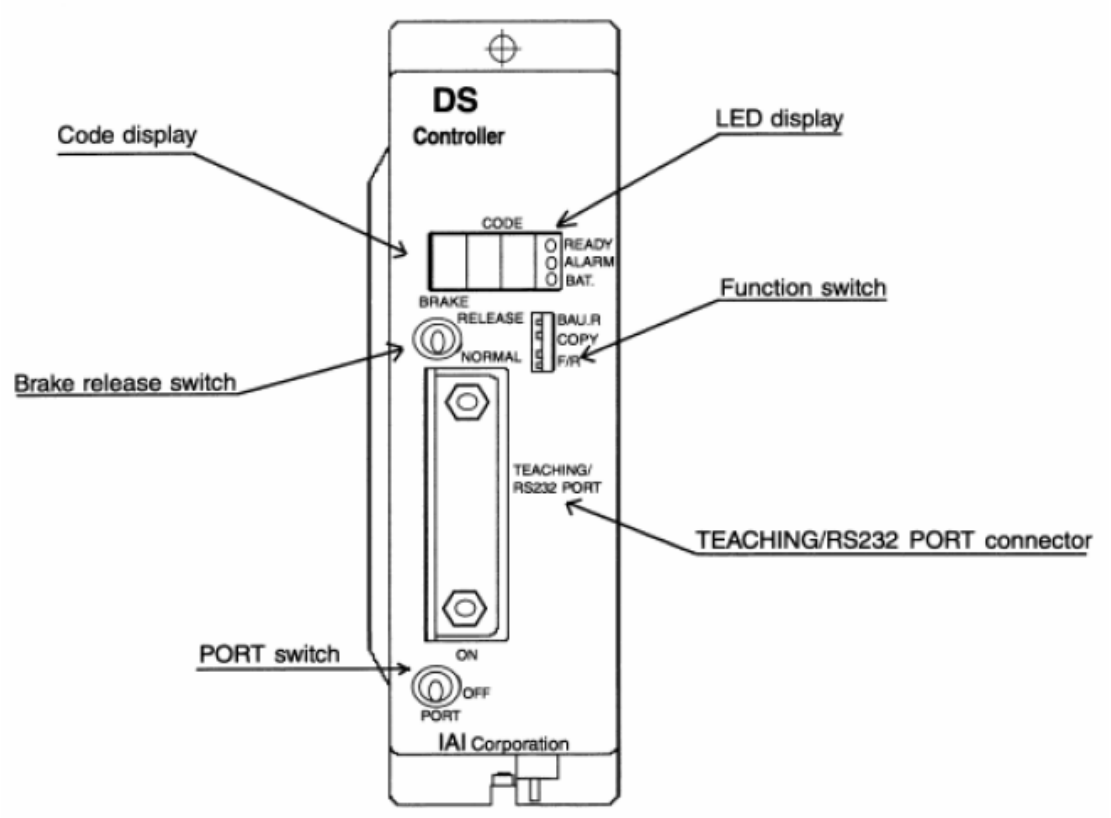


一、

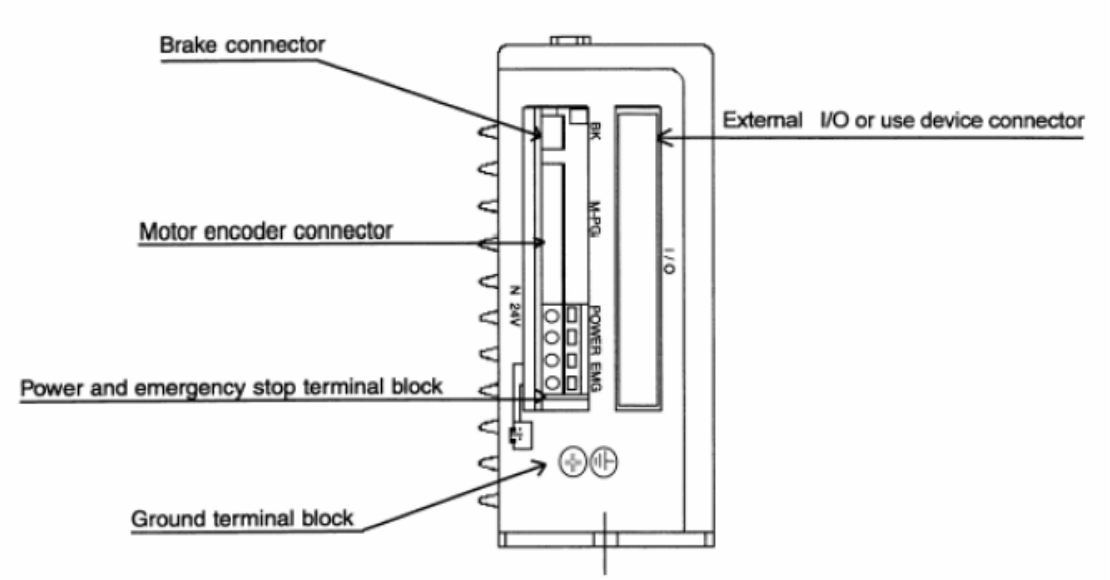
此馬達內並無極限 Sensor，其歸零方式採當其撞到底端時，偵測其扭力大於某值一段時間後，即回走固定的一段距離，以此點為原點，因此每次歸零的位置皆不同，最大誤差為 6 條；但其在兩點間來回移動時，每次回到同依點的誤差在 2 條內。

另外，因其通訊方式採 RS-232，故其同一時間只能傳遞一個位元組資料，因此無法作到及時的馬達位置監控，因為會發生兩組訊號相衝突的情形，導致只能收到其中一組訊號。

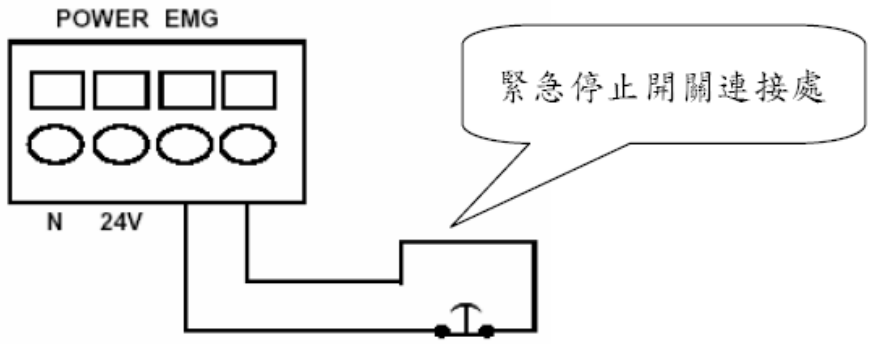




1. Code Display：三位數的顯示裝置，用來顯示控制器運作的狀態。
2. LED Display：READY：表示控制器已準備運作。
   ALARM：表示控制器發生故障。
   BAT.：表示控制器電壓過低。
3. Break Release Switch：Release：煞車釋放。
   Normal：煞車開啟。（預設）
4. Port Switch：ON：使用 Teaching/RS232 Port Connector。
   OFF：不使用 Teaching/RS232 Port Connector。
5. Function Switch：BAU.R：是否改變 Bund rate。
   COPY： 是否從ROM拷貝資料到 FLASH Memory。
   F/R：ROM與FLASH切換。
6. Teaching/RS232 Port Connector：25 PIN的RS232連接裝置。



1. Break Connector：驅動器的煞車連接處。
2. Motor Encoder Connector：驅動器與馬達及Encoder的連接處。
3. I/O Device Connector：34 PIN的IO連接處。
4. Power and Emergency Stop Terminal Block：



三、軟體控制：

1. 作法：透過RS-232傳送Command至DS-Controller，達到馬達控制的目的。
2. Command語法：指令+前置字元+指令集（指令參數）+後置字元。
3. 指令：主要可分為執行指令(！)與查詢指令(？)兩種。
4. 前置字元：99。
5. 指令集：歸零(HOM0110)，停止(HLT01)，移動(MOV01)，JOG(JOG01)與馬達狀態(STA)。

PS. 歸零、停止與馬達狀態可單獨使用，移動與JOG須加指令參數。

6. 指令參數：(0.01~0.99)，速度(0001~0400)，位置(0000.000~0150.999)與JOG方向(0或1)。

PS. 移動＋加速度＋速度＋位置，JOG＋JOG方向。

PS. 位置須由軟體保護，避免超出軌道長度。

7. 後置字元：@@+ASCII Code 13+ASCII Code 10。

8. Example (以下用(N)代表ASCII Code N)

!99HOM0110@@(13)(10) → 歸零

!99HLT01@@(13)(10) (停止

?99STA@@(13)(10) (查詢馬達狀態

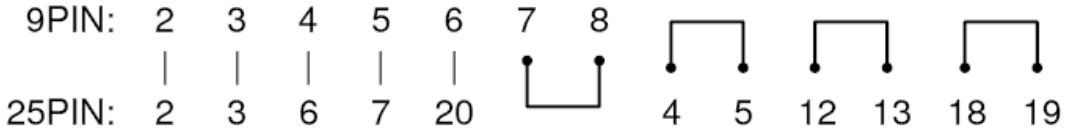
!99JOG010@@(13)(10) (JOG (方向：遠離馬達)

!99MOV010.1001000050.123@@(13)(10)

→以100的速度與 0.10 的加速度移動到 50.123 的位置。

四、接線圖：

1. RS-232 Connect (9PIN 轉 25PIN)：



2. I/O Port Connect (Position Mode)：

I/O

NC	NC	NC	Ready	NC	Pos No.200	Pos No.80	Pos No.20	Pos No.8	Pos No.2	NC	Hold	CPU Reset	NC	NC	NC	24V
17A	16A	15A	14A	13A	12A	11A	10A	9A	8A	7A	6A	5A	4A	3A	2A	1A
17B	16B	15B	14B	13B	12B	11B	10B	9B	8B	7B	6B	5B	4B	3B	2B	1B
0V	NC	NC	Position Comple-tion Alarm	Alarm	Pos No.400	Pos No.100	Pos No.40	Pos No.10	Pos No.4	Pos No.1	NC	Start	NC	NC	NC	NC

24V Input Output 0V

二、 實驗步驟：

1. 確認設備與電腦 COM1 已連線
2. 開啟 Borland C++撰寫之馬達控制程式
3. 選擇通訊埠為 COM1 並點選開啟通訊埠
4. 輸入馬達往復運動範圍並調整好馬達速度與加速度

5. 執行程式往復運動功能


```
//---------------------------------------------------------------------
----------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------
----------
#pragma package(smart_init)
#pragma link "SPComm"
#pragma link "SPComm"
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
```

```
----------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
 strSpeed="0050";
 strAcc="0.10";
 strStarPos="0000";
 strEndPos="0000";
}
//---------------------------------------------------------------------
----------

void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction
&Action)
{
   Comm1->StopComm();
}
//---------------------------------------------------------------------
----------

void __fastcall TForm1::Comm1ReceiveData(TObject *Sender, Pointer
Buffer,
      WORD BufferLength)
{
   Memo1->Text=(char*)Buffer;
}
//---------------------------------------------------------------------
----------

void __fastcall TForm1::RadioButton1Click(TObject *Sender)
{
   Comm1->CommName="COM1";
}
//---------------------------------------------------------------------
----------

void __fastcall TForm1::RadioButton2Click(TObject *Sender)
{
```

```
   Comm1->CommName="COM2";
}
//---------------------------------------------------------------
----------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
   try
   {
      Comm1->StartComm();

      if(Comm1->CommName=="COM1")
      {
         Shape1->Brush->Color=clLime;
         Shape2->Brush->Color=clSilver;
      }
      else if(Comm1->CommName=="COM2")
      {
         Shape1->Brush->Color=clSilver;
         Shape2->Brush->Color=clLime;
      }
   }
   catch(...)
   {
      MessageBox(0, "開啟通訊埠錯誤!!", "Comm Error", MB_OK);
   }
}
//---------------------------------------------------------------
----------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
   Comm1->StopComm();
   Shape1->Brush->Color=clSilver;
   Shape2->Brush->Color=clSilver;
}
//---------------------------------------------------------------
----------
```

```
void __fastcall TForm1::Button9Click(TObject *Sender)
{
   Close();
}
//---------------------------------------------------------------
-----------

void __fastcall TForm1::TrackBar1Change(TObject *Sender)
{
   strAcc = (double)TrackBar1->Position/100;
   strAcc = strAcc.FormatFloat("0.00",
(double)TrackBar1->Position/100);
   Label4->Caption = strAcc;
}
//---------------------------------------------------------------
-----------

void __fastcall TForm1::TrackBar2Change(TObject *Sender)
{
   strSpeed = TrackBar2->Position;
   strSpeed = strSpeed.FormatFloat("0000", TrackBar2->Position);
   Label6->Caption = strSpeed;
}
//---------------------------------------------------------------
-----------

void __fastcall TForm1::UpDown1Click(TObject *Sender, TUDBtnType
Button)
{
   Edit2->Text=UpDown1->Position;
}
//---------------------------------------------------------------
-----------

void __fastcall TForm1::Edit2Change(TObject *Sender)
{
   UpDown1->Position = StrToInt(Edit2->Text);
```

```
}
//---------------------------------------------------------------------
-----------

void __fastcall TForm1::Button3Click(TObject *Sender)
{
 int Pos1;

 try
   {
     Pos1=Edit2->Text.ToInt();
 }
 catch(EConvertError &E)
   {
     ShowMessage(E.Message+".\n 請輸入 0-150 之整數!!");
     return;
 }

 try
   {
     int Pos2=Edit3->Text.ToInt();
 }
 catch(EConvertError &E)
   {
     ShowMessage(E.Message+".\n 請輸入整數!!");
     return;
 }

 if(Pos1>150||Pos1<0)
   {
     ShowMessage("請輸入 0-150 之整數!!");
     return;
 }

 String Temp1, Temp2;
 String mov="MOV01";     //移動指令
 String acc=strAcc;          //加速度
 String vel=strSpeed;        //移動速度
```

```
 Temp1 = Edit2->Text;
   Temp1 = Edit3->Text;
   Temp1 = Temp1.FormatFloat("0000", Edit2->Text.ToInt());
   Temp2 = Temp2.FormatFloat("000", Edit3->Text.ToInt());

 Temp1 = Temp1 + "." + Temp2;    //Temp1=XXXX.XXX
 Temp1 = mov + acc + vel + Temp1;

   SendData("!99" + Temp1);
}
//---------------------------------------------------------------
-----------

void __fastcall TForm1::Edit4Change(TObject *Sender)
{
 try
   {
     int tmp=Edit4->Text.ToInt();
 }
 catch(EConvertError &E)
   {
     ShowMessage(E.Message+"請輸入整數!!");
 }

 strStarPos=Edit4->Text;
   strStarPos = strStarPos.FormatFloat("0000",
Edit4->Text.ToInt());
}
//---------------------------------------------------------------
-----------

void __fastcall TForm1::Edit5Change(TObject *Sender)
{
 try
   {
     int tmp=Edit5->Text.ToInt();
 }
```

```
 catch(EConvertError &E)
   {
     ShowMessage(E.Message+"請輸入整數!!");
 }

 strEndPos=Edit5->Text;
   strEndPos = strEndPos.FormatFloat("0000", Edit5->Text.ToInt());
}
//---------------------------------------------------------------
-----------

void __fastcall TForm1::Button5Click(TObject *Sender)
{
   SendData("!99HOM0110");
}
//---------------------------------------------------------------
-----------

void __fastcall TForm1::Button6Click(TObject *Sender)
{
 SendData("!99HLT01");
}
//---------------------------------------------------------------
-----------

void __fastcall TForm1::Button5MouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
   IsSendCom=true;
}
//---------------------------------------------------------------
-----------

void __fastcall TForm1::Button5MouseUp(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
   IsSendCom=false;
}
```

```
//---------------------------------------------------------------
----------


void __fastcall TForm1::Button7MouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
 IsSendCom=true;
 String Temp;
 String acc=strAcc;
 String vel="0050";
 String jog="JOG01";
 char z='1';
 Temp=jog+acc+vel+z;

 SendData("!99"+Temp);
}
//---------------------------------------------------------------
----------

void __fastcall TForm1::Button7MouseUp(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
 SendData("!99HLT01");
 IsSendCom=false;
}
//---------------------------------------------------------------
----------

void __fastcall TForm1::Button8MouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
 IsSendCom=true;
 String Temp;
 String acc=strAcc;
 String vel="0050";
 String jog="JOG01";
 char z='0';
```

```
 Temp=jog+acc+vel+z;

 SendData("!99"+Temp);
}
//---------------------------------------------------------------
-----------

void __fastcall TForm1::Button4Click(TObject *Sender)
{
   Memo1->Clear();
}
//---------------------------------------------------------------
-----------

void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
   AnsiString Temp=Edit1->Text;
   Comm1->WriteCommData(Temp.c_str(), Temp.Length());
}
//---------------------------------------------------------------
-----------

void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
 String tmpStarPos="MOV01"+strAcc+strSpeed+strStarPos+".000";
 String tmpEndPos="MOV01"+strAcc+strSpeed+strEndPos+".000";
 static bool LoopDir=false;
 IsSendCom=false;

 if(SpeedButton1->Down)
   {
     IsSendCom=true;
     if(LoopDir)
      {
         SendData("!99" + tmpStarPos);
         LoopDir=false;
    }
    else
```

```
      {
         SendData("!99" + tmpEndPos);
         LoopDir=true;
      }
   }
}
//---------------------------------------------------------------
----------

void __fastcall TForm1::Timer2Timer(TObject *Sender)
{
   SendData("?99STA");

   AnsiString Temp=Memo1->Text;
 Temp=Temp.SubString(13,7);
   Label2->Caption=Temp;
}
//---------------------------------------------------------------
----------

void __fastcall TForm1::Timer3Timer(TObject *Sender)
{
 if(IsSendCom)
     Timer2->Enabled=false;
 else
     Timer2->Enabled=true;
}
//---------------------------------------------------------------
----------

//副程式

void __fastcall TForm1::SendData(String Command)
{
   AnsiString Temp;
   Temp = Command + "@@"+'\xD'+'\xA';     //'\xA':ACSII Code
13,'\xD':ACSII Code 10
```

```
    Comm1->WriteCommData(Temp.c_str(), Temp.Length());
}
//---------------------------------------------------------------------
-----------
```