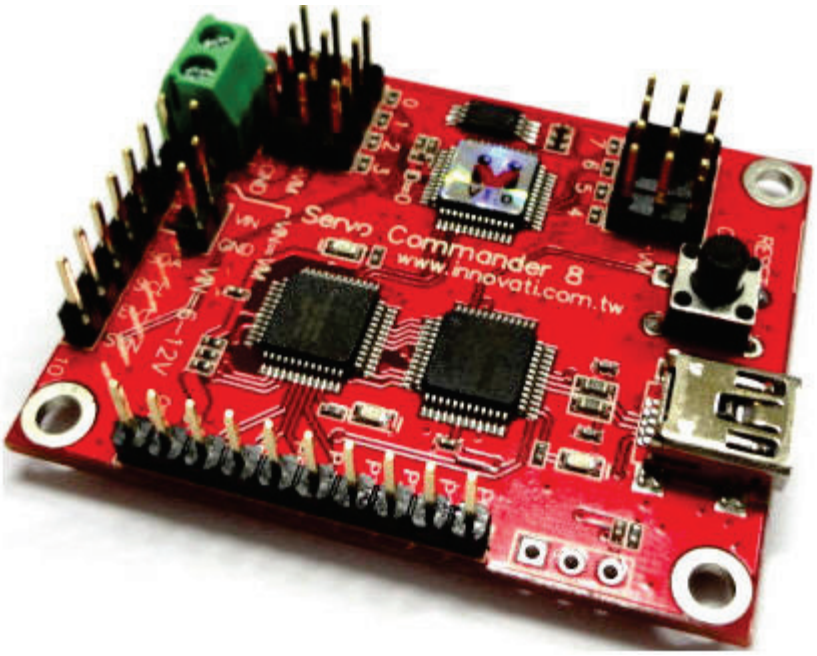
Servo Commander™ 8

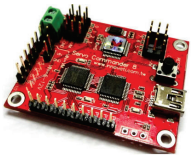
基本操作手冊



一、 Servo Commander™ 8 是甚麼?

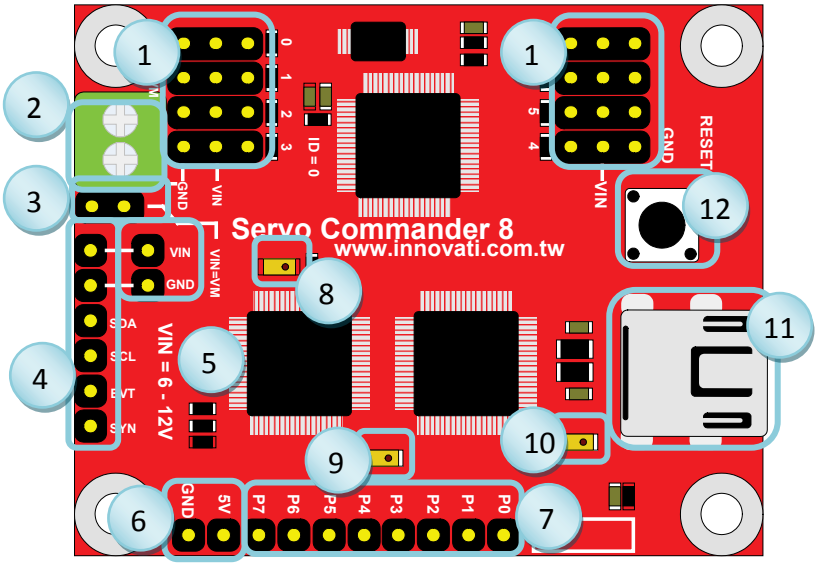
Servo Commander™ 8 結合了 BASIC Commander®與 Servo Runner A，讓使用者可以透過 USB 連接線，下載編寫好的 innoBASIC™程式，直接執行程式內容，還可以同時連接最多八個伺服機，操作伺服機的動作。如果想要體驗簡易機器人的設計與操作，Servo Commander™ 8 將是最好的入門學習工具。Servo Commander™ 8 簡稱為 SC8，以下文件中的 SC8 就是指 Servo Commander™ 8。

二、 操作元件與設備

項次	品名	圖片	數量
1	個人電腦(PC) *1		1
2	Servo Commander™ 8		1
3	USB 連接線		1
4	伺服機 *2		1

*1 電腦安裝的系統需要是 Window XP，Vista，或 Win 7。
*2 請注意電源電壓需要為伺服機所能承受的電壓。

三、 認識 Servo Commander™ 8



編號	功能說明
1	伺服機連接腳位，左右各四組共八組，編號為 0~7，對應到程式中的各個伺服機操控指令。每組腳位有三根接腳，左邊四組，由左到右依序為地，電源與訊號腳，右邊四組腳位與左邊四組相反，連接時請注意對應腳位，避免錯誤連接造成元件毀損。
2	伺服機電源連接插孔，連接時請注意電源極性，接入電壓須依照伺服機的規格而定。
3	輸入電源與伺服機電源共用排針，當短路時輸入電源 VIN 與伺服機電源 VM 為共用。
4	cmdBUSTM 連接腳，可以另外連結各種不同功能的模組。
5	電源連接腳，可以做為電源的輸入。
6	5V 電源接腳，可以提供 5V 的轉換電源。
7	通用輸出入腳(GPIO)，可以透過指令控制腳位輸出高電位或低電位(輸出)，也可以檢測腳位上的電壓是高電位或低電位(輸入)，還可以使用指令，當成 I2C 或 UART 的溝通腳位，總共有八根腳位可以使用。
8	紅色 LED，電源指示燈，點亮代表有接上電源。
9	橘色 LED，模組溝通指示燈，閃爍代表與模組在傳送資料。
10	綠色 LED，USB 溝通指示燈，閃爍代表與電腦進行資料收送。
11	USB 插孔，插入 USB 連接線與電腦連接，才可以進行程式的下載，或是傳送訊息至終端視窗。
12	重置按鈕，可以重新啟動下載的程式內容。

四、 認識 innoBASIC™

1. 下載 innoBASIC™ Workshop:

可以直接連結下載網頁並點選立即下載下載。

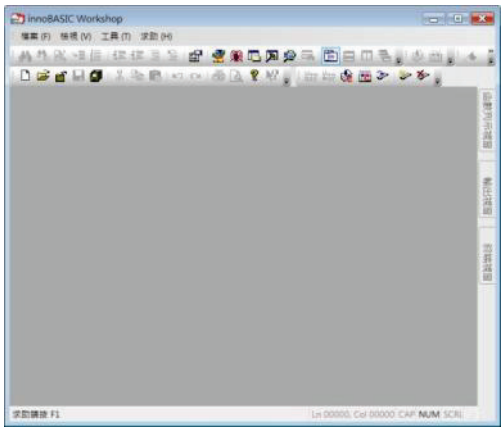
http://www.innovati.com.tw/website/down/html/?113.html

下載完成後，講檔案解壓縮，直接執行安裝檔並依步驟安裝。

2. 開啟 innoBASIC™ Workshop:

安裝完成後會在桌面上看到 innoBASIC™ Workshop 的圖示，點擊 innoBASIC™ Workshop 圖示，啟動 innoBASIC™ Workshop。

確認畫面上出現 innoBASIC™ Workshop 的工作視窗。



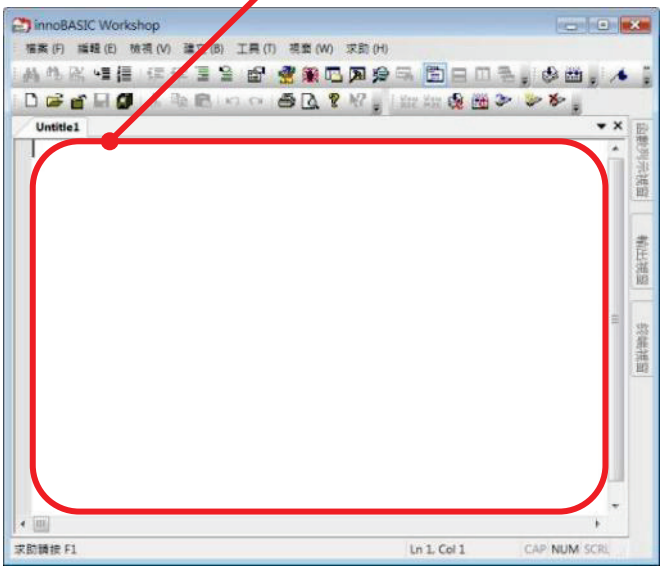
3. 連接電腦與 SC8:

電腦第一次安裝 innoBASIC™ Workshop，在 USB 與 SC8 連接時，會出現安裝驅動程式選項，請依照步驟執行安裝，安裝完成後，下一次連結時，就不會再出現驅動程式安裝視窗。

4. 開啟新檔:

在 innoBASIC™ Workshop 的工具列，點選“開新檔案”按鈕，就可以開啟新的編輯檔案。

按下開新檔案按鈕後，視窗就會出現如下圖的編輯畫面，空白的區域，就是提供使用者編寫程式的位置



5. **輸入程式:**

```
1  Sub Main()
2      Debug "Hello"
3  End Sub
```

6. **程式說明:**

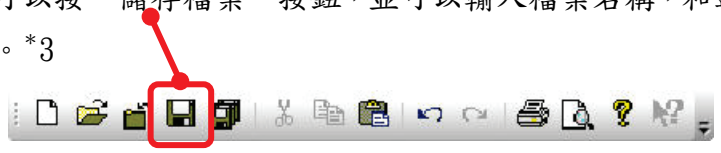
第 1 行: 主程式開始點。

第 2 行: 執行顯示指令，顯示“Hello”在終端視窗。

第 3 行: 程式結束。

7. **儲存檔案:**

編輯完成後可以按“儲存檔案”按鈕，並可以輸入檔案名稱，和選擇要儲存的位置。*3



8. **編譯及下載:**

按下“建立”按鈕，就可以將檔案下載到 SC8 並執行。

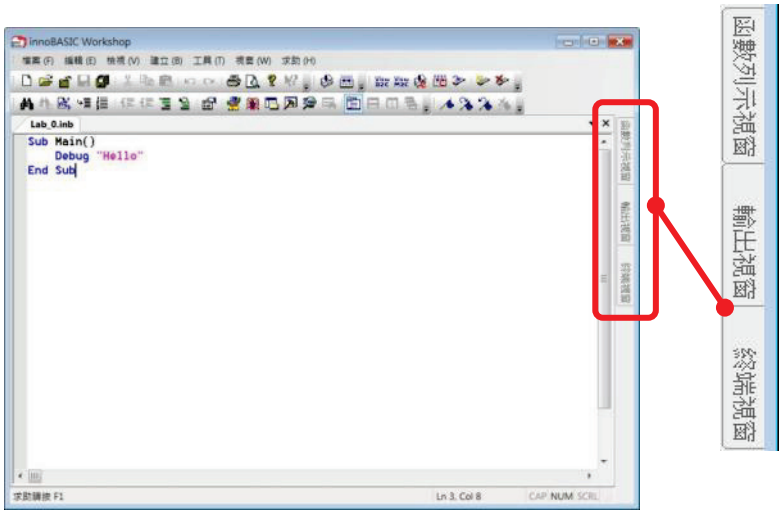
經過編譯後，若程式正確，下載完成後會在輸出視窗顯示程式使用率，並且整個系統會開始執行程式；若程式編譯有錯，會在輸出視窗顯示錯誤訊息，此時需將程式除錯完成才可將程式下載至 SC8。程式在下載過程中靠近 USB 接頭的綠燈會閃爍，代表程式正在下載至 SC8。*4

*3 在儲存後沒有更改程式內容時，按鈕會呈現灰色並且無法點選。

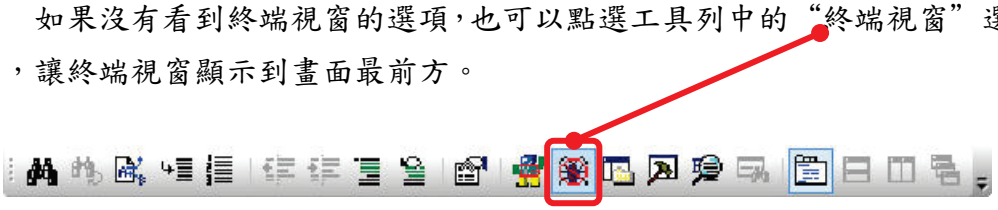
*4 下載過程中請勿移除 USB 線。

五、 執行結果

在 innoBASIC™ Workshop 工作區可以看到三個視窗選項，分別是函數列示視窗，輸出視窗與終端視窗，可以用拖拉的方式移動到不同位置，也可以分離或合併。



如果沒有看到終端視窗的選項，也可以點選工具列中的“終端視窗”選項，讓終端視窗顯示到畫面最前方。



可以看到終端視窗上顯示出“Hello”文字。

六、 基本指令操作介紹

一個程式都要有開始點與結束點，前面範例程式的第一行與第三行，就是每個程式都要有的架構，讓系統知道開始結束的位置，第二行的 Debug 指令，會將後面敘述中，雙引號中的文字，顯示到終端視窗，指令的格式如下:

Debug *Item*, {, *Item*}

- *Item* – 用來顯示在終端視窗上的訊息、控制碼或變數。如果超過一個，可以用逗號分開。

Debug 指令可以幫助使用者在需要與 BASIC Commander®做溝通時，扮演傳遞資訊的角色，它可以傳遞使用者內建的訊息到終端視窗，也可以將參數的變化顯示到畫面上，在除錯時能發揮很大的功效。

除了 Debug 指令，用在輸出外，還可以使用 Debugin 指令，讀取從終端視窗輸入的訊息，讀入的資料，則是必須先用 DIM 宣告，相關指令說明如下:

Dim *Variable* As *Type* {* *Size*}

- *Variable* – 用來儲存值的變數。
- *Type* – 合乎規定的變數形態名稱，包括布林值、位元組、整數、字元組、長整數、浮點數、持久位元組、持久整數、持久字元組、持久長整數和持久浮點數。
- *Size* – 定義字串形態變數大小的常數，字串的大小受限於可獲得的 RAM 資料記憶體。

在程式中要使用的變數，都必須要先做宣告，宣告可以讓系統知道變數的大小限制，以及是否要有正負號的轉換，確定系統是否有足夠的記憶體空間。DIM 指令必須要放在程式的開始點，不能在程式中再臨時宣告新參數。

Debugin *Item*, {, *Item*}

- *Item* – 用來顯示在終端視窗上的訊息、控制碼或變數。如果超過一個，可以用逗號分開。

Debugin 指令與 Debug 指令有點相似，可以將指令後面，雙引號夾著的字串顯示到終端視窗，但 Debugin 指令不會將變數內的資料輸出，反而是將使用者從終端視窗輸入的資料，放到變數中。所以在需要使用者輸入資料時，就可以使用 Debugin 指令，而根據後面所接的變數個數，使用者也要輸入對應個數的資料。

如果想要從終端視窗讀取文字，可以參考下面的範例程式:

```
1  Sub Main()
2      Dim sName As String * 10
3
4      Debugin "請輸入英文名: ", sName, CR
5      Debug sName, ", 歡迎使用 innoBASIC!"
6  End Sub
```

第 1 行: 主程式開始點。

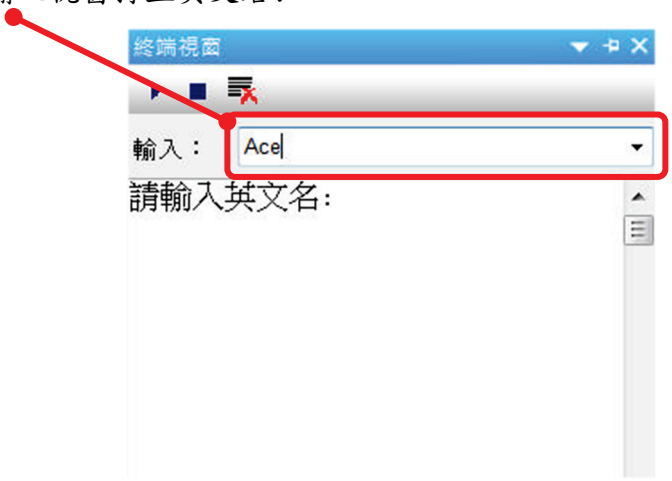
第 2 行: 宣告字串參數“sName”。

第 4 行: 執行顯示與輸入指令，顯示“請輸入英文名”在終端視窗，並等待終端視窗的輸入，會將終端視窗的輸入文字，放到 sName 參數中。

第 5 行: 執行顯示指令，將參數 sName 的字串顯示在終端視窗，並顯示“歡迎使用 innoBASIC!”文字。

第 6 行: 程式結束。

按下下載按鈕，將程式下載到 SC8，就會看到終端視窗出現下面的訊息，請在輸入視窗打上英文名:



接著按下輸入鍵，就可以看到輸入的英文名顯示在終端視窗:

更多的指令說明可以參考使用手冊上的介紹:

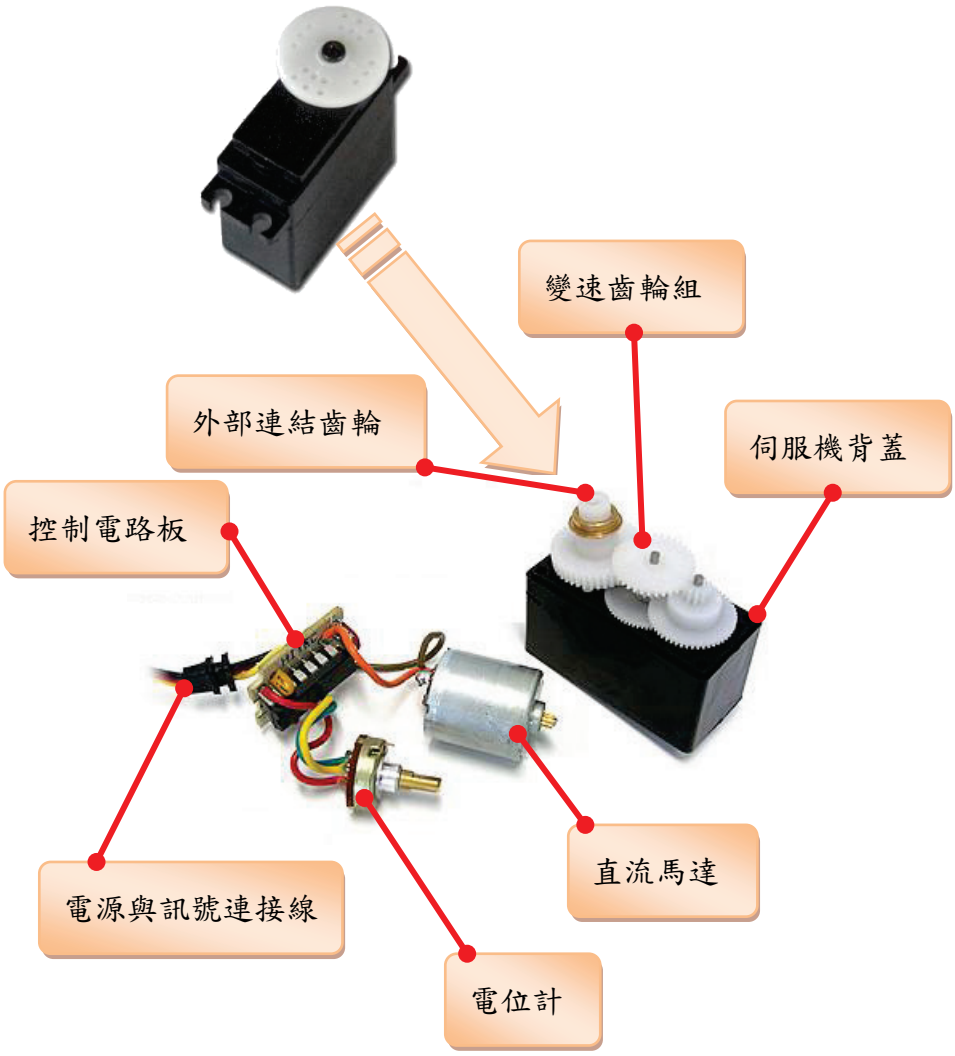
http://www.innovati.com.tw/website/down/html/?33.html

七、 認識伺服機

伺服機簡介

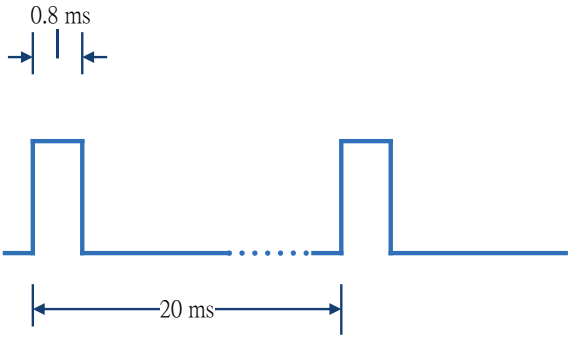
這邊介紹的伺服機，常被人稱做 RC 伺服機(RC Servo)，或是伺服馬達(servo motor)，常可以在遙控模型上看到，包括遙控飛機上，操控機翼轉向，遙控車上，控制輪台轉向，都會用到伺服機。伺服機主要功能就是根據接收的訊號，控制舵片的旋轉角度，通常會有至少一百八十度的旋轉範圍，但根據種類的不同，也有可以旋轉兩百度以上，甚至有三百六十度的伺服機。伺服機可以控制旋轉角度的特性，很適合用在模擬人或動物的關節，所以在許多機器人或相關生物的模擬上，都可以看到伺服機的蹤影。伺服機有分直流伺服機，交流伺服機或線性伺服機等許多種類，這邊只介紹由訊號脈波寬度控制轉角，使用直流馬達製作的標準直流伺服機。

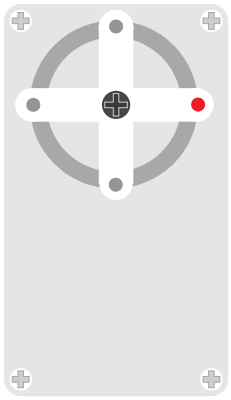
伺服機的結構



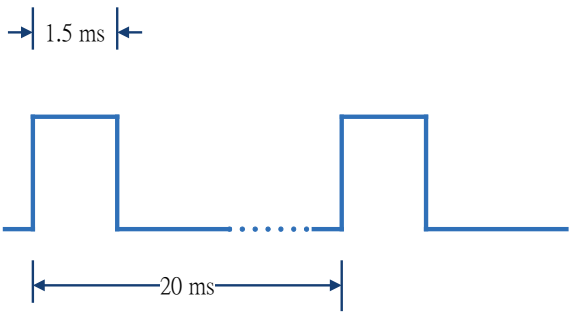
控制伺服機

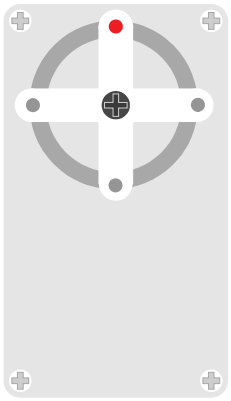
伺服機上有一根訊號線，要控制伺服機，就要每隔 20ms(毫秒，等於千分之一秒)，透過訊號線，傳送一個脈波給伺服機，伺服機的控制電路板，就會根據接受到訊號的脈波寬度，以μs(微秒，百萬分之一秒)為單位，轉換成伺服機要旋轉並固定的角度。注意!不同伺服機可能會對應不同角度。



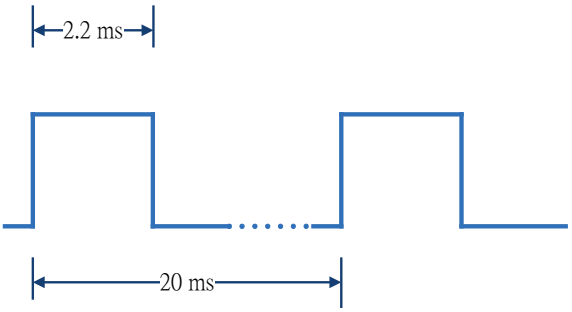


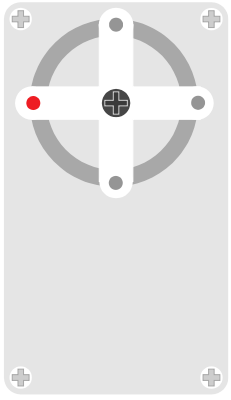
伺服機每隔 20ms 送出 0.8ms 的脈波，伺服機舵片固定在 0 度。





伺服機每隔 20ms 送出 1.5ms 的脈波，伺服機舵片固定在 90 度。





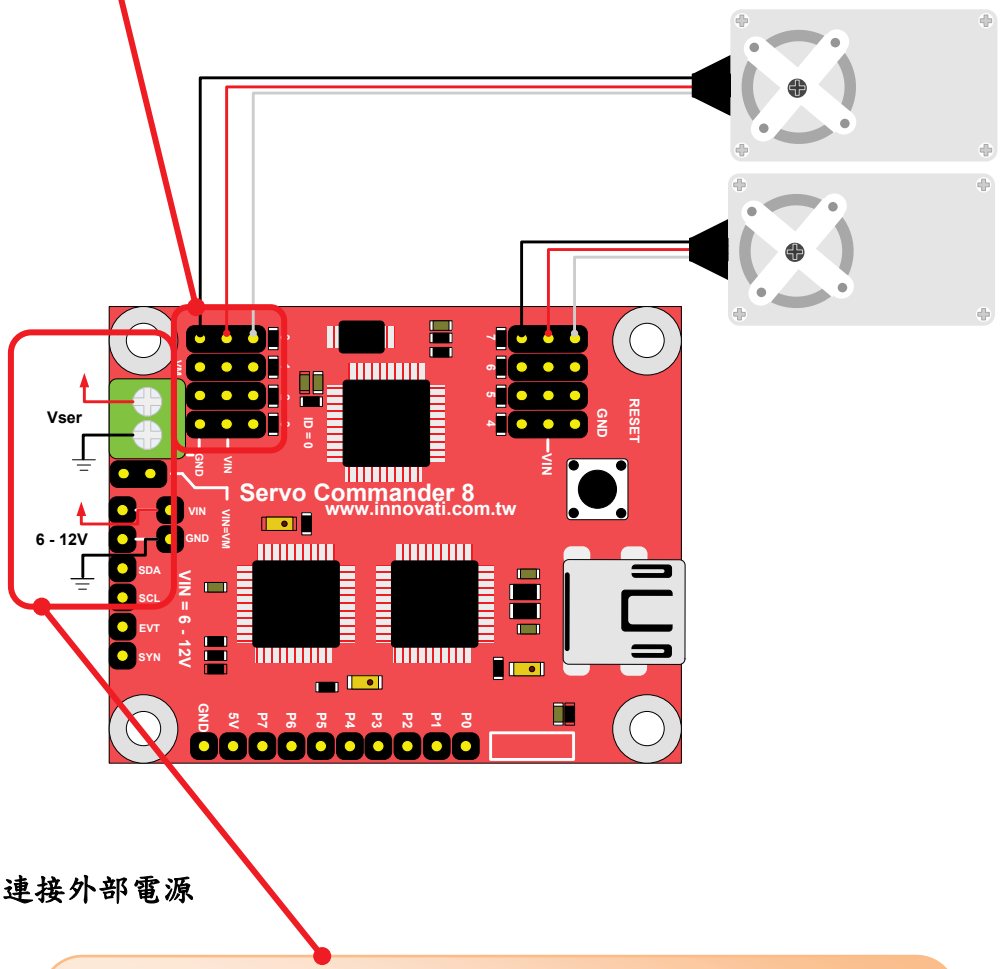
伺服機每隔 20ms 送出 2.2ms 的脈波，伺服機舵片固定在 180 度。

八、 操作動作編輯器

1. 連接伺服機與 SC8

如圖連接伺服機到第 0 組伺服機腳位。

伺服機連接線可能有其他顏色，務必根據伺服機說明連接到正確腳位，錯誤的連接可能會毀損伺服機與 SC8



連接外部電源

這一部份是電源示意圖，紅色的電路接線請與電源的高電位端連接，黑色的電路接線請與電源的低電位端連接。

Vin 請輸 6~12V，Vser 電壓請確認是否為伺服機所能承受的電壓。

若 Vser 電壓於 6~12V 的範圍，可使用 Jumper 讓 2 種電源共用。

2. 開啟動作編輯器(Servo Motion Editor)

開啟動作編輯器前，須先確定 USB 連接線，已經連接電腦與 SC8，並且 innoBASIC™ Workshop 已經正確啟動。

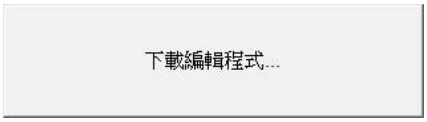
在 innoBASICTM Workshop 的工具列上，找到機器人圖案的動作編輯器按鈕:

點擊動作編輯器按鈕，就會啟動動作編輯器初始設定視窗，此時可以選擇所使用的模組:

把左方的使用模組選為“Servo Commander 8”:

確定左方的使用模組名稱後，點選“確定”按鈕，就會開始下載動作編輯器:

在下載動作編輯器時，會出現下載提醒視窗，此時請不要拔除 USB 連接線，並等待下載完成:



下載完成會出現提醒視窗，請點選“確定”按鈕進入動作編輯器:

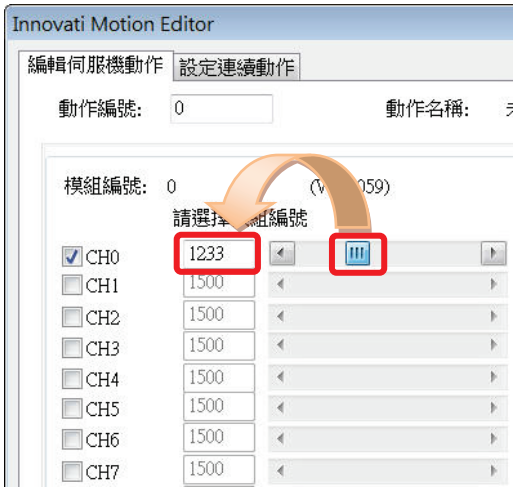
確認動作編輯器視窗:

3. 使用拖曳方式設定伺服機位置

先勾選“CH0”的伺服機啟動選框:

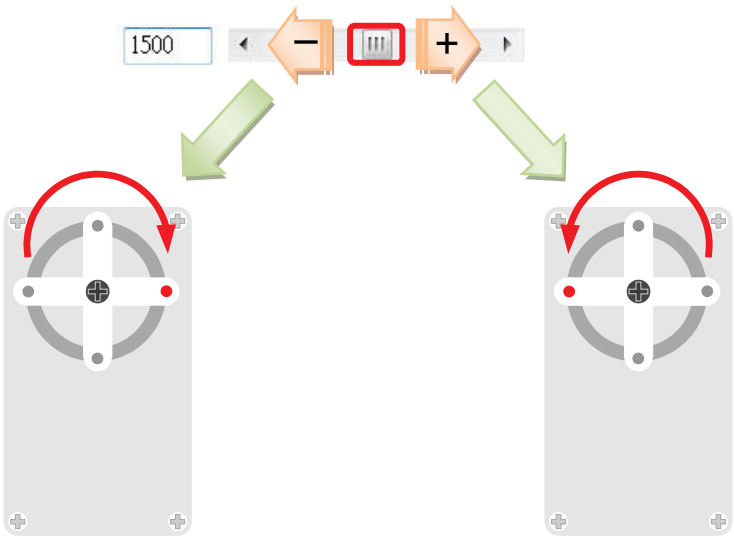
確認伺服機位置由灰色變為黑色，並且出現拖曳方塊:

在拖曳方塊上按住滑鼠左鍵，再將滑鼠左右移動，就可以改變畫面上的伺服機位置:

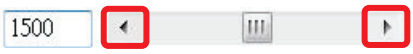


放開滑鼠左鍵後，伺服機就會開始移動，一直到拖曳方塊所設定的伺服機位置。

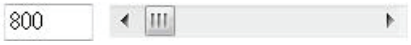
將拖曳方塊往左移動，伺服機位置的數值就會減少，並且讓伺服機往順時針方向轉動，如果將拖曳方塊往右移動，伺服機位置的數值就會增加，並且讓伺服機往逆時針方向轉動。



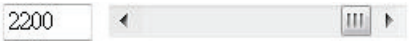
也可以用滑鼠左鍵，點擊左邊或右邊的箭頭按鈕，讓伺服機位置一次只改變一個刻度，如果在箭頭符號上久按滑鼠左鍵，則可以讓伺服機位置不斷增加或減少，直到伺服機到達設定的極限值。



當伺服機位置到達最小值，再按箭頭符號也不會改變伺服機位置，伺服機位置的最小值預設為 800:



當伺服機位置到達最大值，再按箭頭符號也不會改變伺服機位置，伺服機位置的最大值預設為 2200:



4. 直接設定目標位置

直接點選伺服機位置欄位，可以看到游標現閃爍，就可以在欄位中，輸入想要轉動的位置，輸入時必須設定範圍在 800 ~ 2200 之間的整數值，單位為μs，代表要輸出的脈波寬度：

設定完成後請按下鍵盤上的輸入鍵，伺服機就會開始移動到設定的位置。

5. 設定轉動速度

點擊“Speed”按鈕，就可以設定伺服機轉動的模式，更改為用固定速度轉動：

在“Speed”按鈕右方可以輸入速度值：

速度值可以輸入 0 ~ 65535 間的整數值，完成這兩項設定後，當伺服機位置更動的時候，都會用固定的速度轉動，不管轉動的角度大或小，設定速度的單位是μs 每秒，就是代表每秒伺服機要移動多少個μs 的角度，這邊的μs 已經不是時間的單位，而是轉換成伺服機轉動角度的單位，如果現在把伺服機位置改設定到 2200，並按下輸入鍵，伺服機就會花約七秒的時間，從 1500 的位置，移動到 2200 的位置:

原先的伺服機位置為 1500，目標位置為 2200，因為(2200 － 1500)=700，所以伺服機要移動 700μs 的角度，而設定的速度為 100μs 每秒，所以(700 ÷ 100)=7，伺服機要花七秒的時間移動。

在需要伺服機以固定轉速旋轉的應用時，就可以設定伺服機以速度模式移動，在知道開始位置和目標位置的條件下，也可以計算出伺服機轉動所要花的時間。可以試著更改不同速度值，設定不同的伺服機位置，就可以對速度模式有更多的了解，另外要注意不同伺服機，轉速不同，如果設定的速度超過伺服機上限，伺服機也只會用最快速度移動。

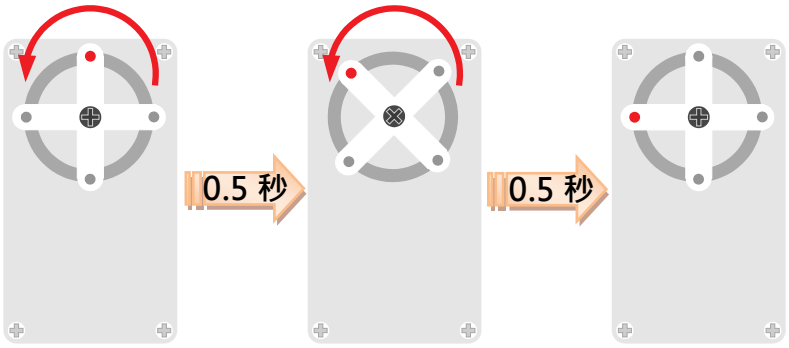
動動腦: 如果讓伺服機從 800 的位置，移動到 1600 的位置，設定速度是 200，伺服機要轉動多久?

6. 設定轉動時間

點擊“Time”按鈕，就可以設定伺服機轉動的模式，更改為用設定時間轉動:

在“Time”按鈕右方可以輸入時間值:

時間值可以輸入 0 ~ 65535 間的整數值，完成這兩項設定後，當伺服機位置更動的時候，都會開始以固定速度轉動，這個速度是由模組內部計算，不管設定的起始與目標位置差異，都要花上設定的時間值，才會到達目標位置。設定時間的單位是 ms，就是代表伺服機要花多少 ms，才能到達目標位置，如果現在設定目標位置為 2200，伺服機就會花一秒的時間，轉動到 2200 的位置:



伺服機是以等速移動，所以可以計算出，經過一半的時間，伺服機也旋轉了一半的角度。要注意的是，不同伺服機的最快轉速不同，如果伺服機的轉速，無法在指定的時間內，到達目標位置，伺服機將會以最快的轉速旋轉。在同時操控多個伺服機，希望讓每個伺服機，在同樣時間，到達不同目標位置時，設定時間模式，是一個很方便的方式，尤其是在製作以伺服機為關節的機器人時，但要注意的是，在時間模式，伺服機是以等速移動，如果一個伺服機需要有不同的速度，表現更好流暢性時，就要組合更多的設定位置，或是改用其他模式。

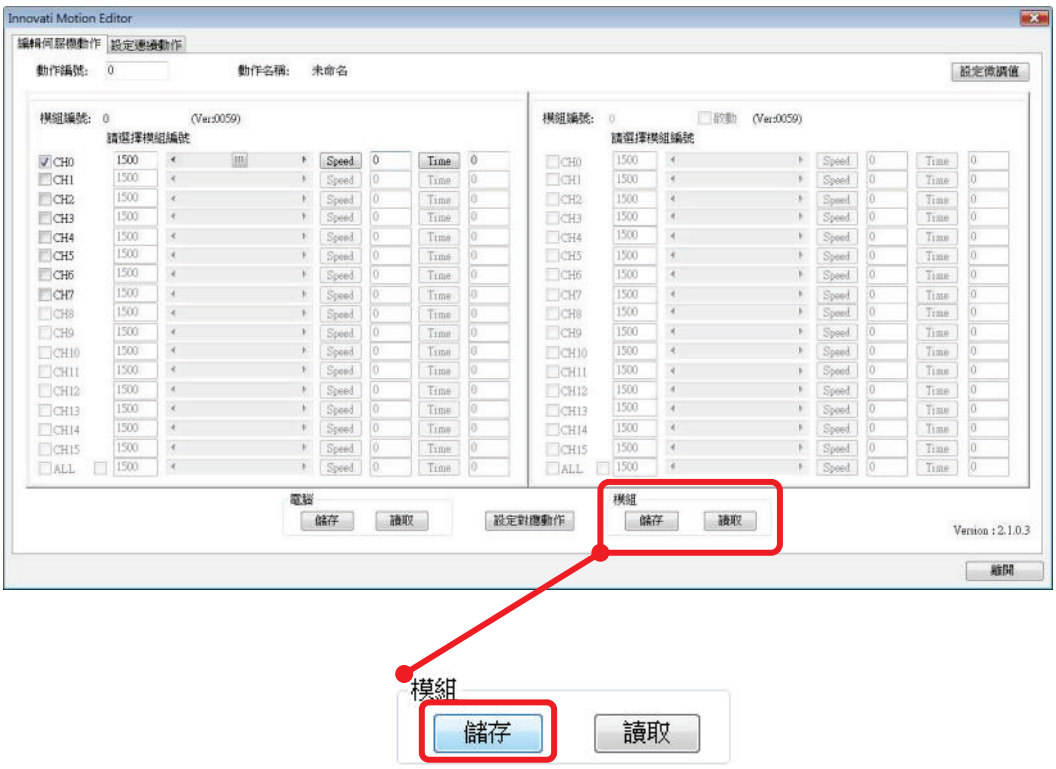
動動腦:如果想讓伺服機從 2000 的起始位置,移動到 1000 的目標位置，希望用時間模式，花五秒到達，要怎麼操作?如果用速度模式呢?

設定時要注意，時間模式與速度模式不能同時存在，如果點選時間模式，再去點選速度模式，動作編輯器就會自動取消時間模式，如果設定在時間模式，再點一下時間模式按鈕，就可以取消時間模式，這個時候，伺服機會以最快的速度移動。同樣的，在速度模式，也可以用再點擊一下速度模式按鈕，取消速度模式。

7. 儲存與讀取設定動作

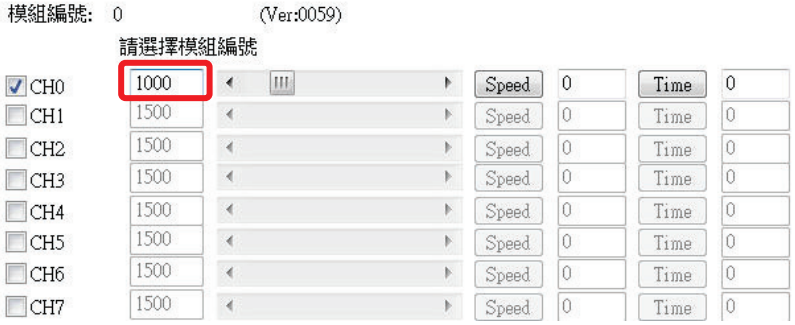
SC8 還可以將設定好的伺服機目標位置，伺服機轉動模式，以及轉動模式的設定值，儲存到模組中，每一組伺服機位置與轉動模式的相關資料，就被稱做一個動作(frame)，可以想像一個在播放的影片，暫停的時候，就是一個這樣靜止的畫面。SC8 可以儲存最多六十組動作，要儲存設定好的動作，就要先在動作編輯器的畫面左上方，輸入動作編號:

動作編號可以輸入 0 ~ 59 間的整數值，接著再點選動作編輯器右下角，模組欄中的“儲存”按鈕:



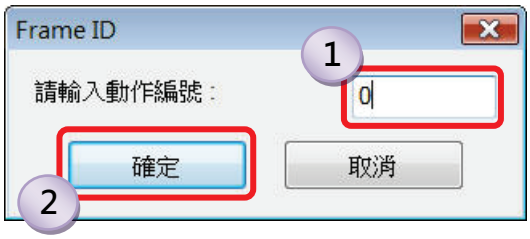
在畫面上會出現儲存訊息，代表模組正在儲存動作:

當儲存訊息消失，代表動作已經儲存到模組，如果想要測試剛剛的動作，是否有被正確儲存，就可以用讀取的方式，檢查儲存的動作資料，但要先將動作位置，設定到與剛剛儲存值不同的位置，例如 1000:



再點選模組欄中的“讀取”按鈕:

接著畫面上會出現動作編號輸入視窗，輸入之前儲存的動作編號，例如範例中為 0，再按下確定按鈕:



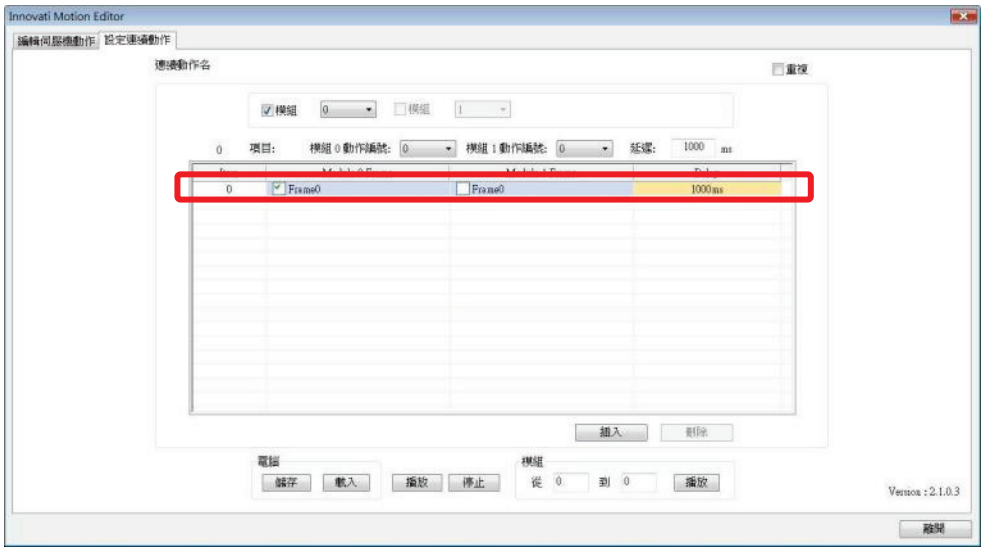
然後就會看到畫面上的伺服機位置回到 1500，而且是依照設定的模式，最快的速度返回，用同樣的方法，也可以儲存速度與時間模式，以後只要依序呼叫不同的動作編號，就可以組合出不同的動作。

動動腦:想要儲存速度模式與時間模式的範例，到動作編號 1 與動作編號 2 裡，該怎麼操作?

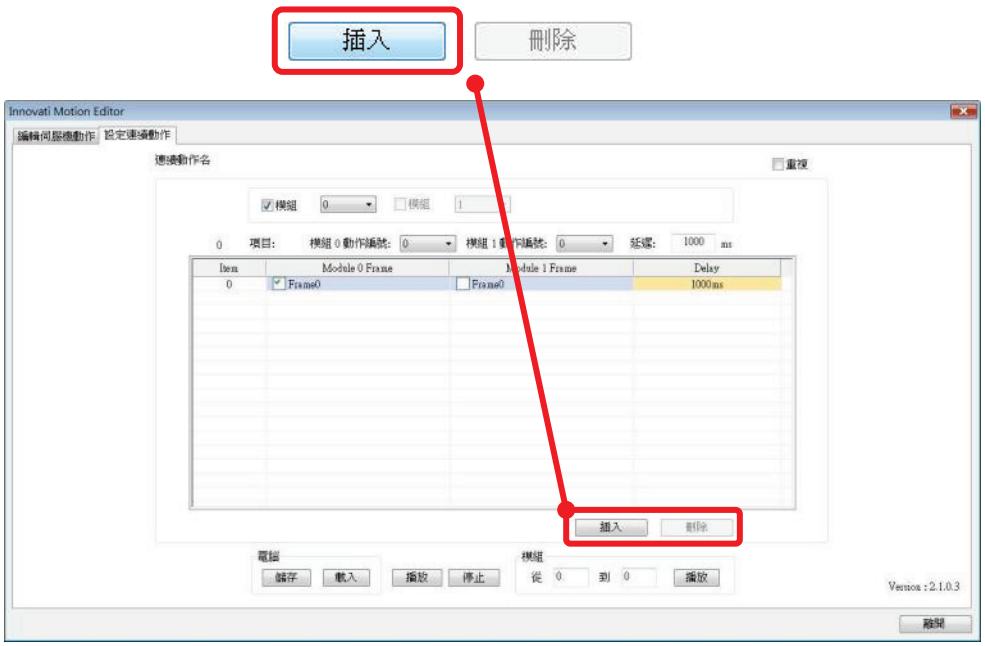
9. 設定連續動作

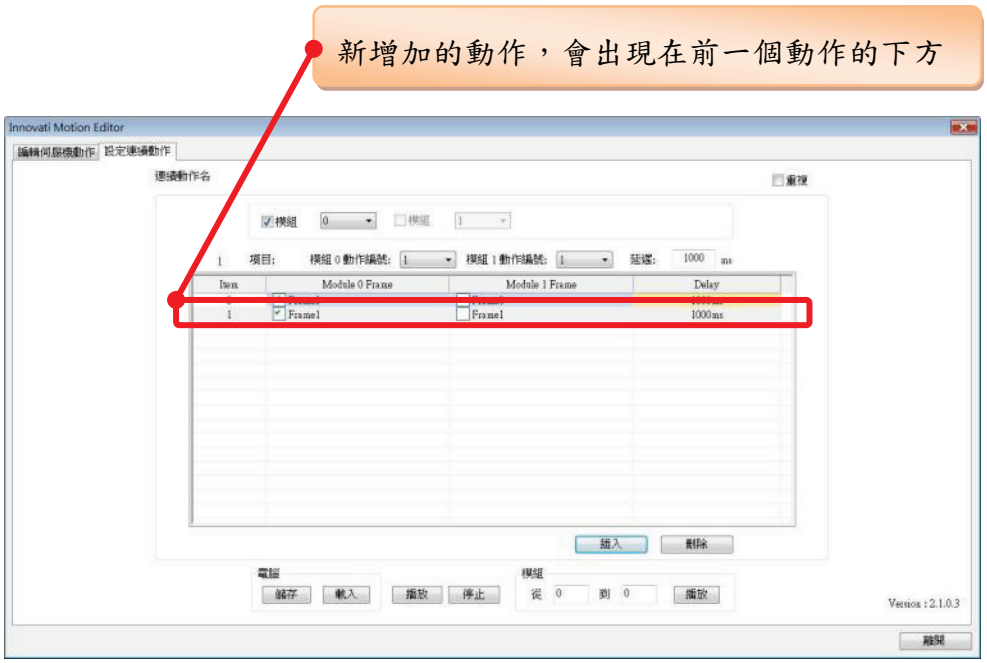
如果想要快速組合不同動作，就可以在設定連續動作頁面，設定要讀取的動作編號，並依序播放，就能看到伺服機展現設定的動作，。首先利用前一節的方式，先儲存動作 0 為目標位置 1500，動作 1 為目標位置 2200，動作 2 為目標位置 800，並且都設定為速度模式，速度值設定為 100，接著點選動作編輯器上的“設定連續動作”標籤:

動作編輯器會將畫面切換到設定連續動作頁面，可以看到預設就有一組動作編號 0:



點選下方的“插入”按鈕，就可以新增加一個動作:





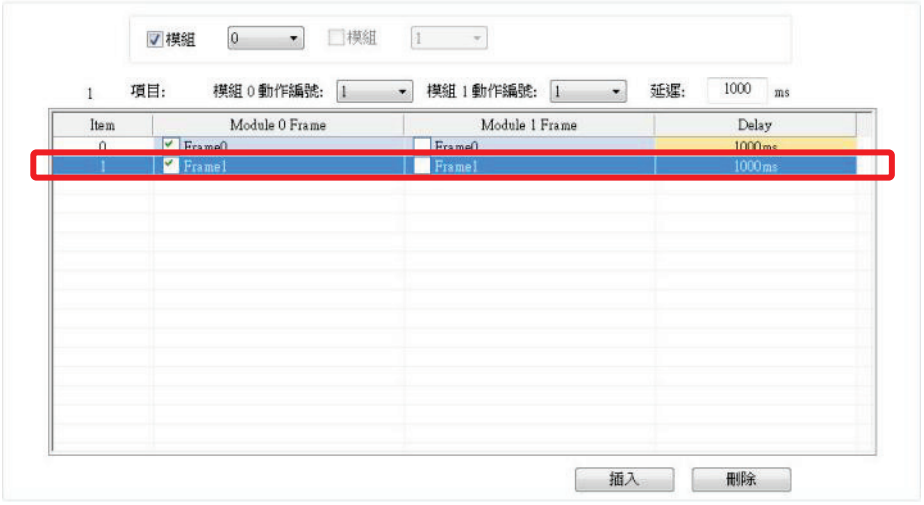
在每一個動作的最後一個一個欄位是“Delay”欄位，可以儲存要等待的時間，也就是要用多久的時間，完成這個動作。由於第一個動作位置是1500，而第二個動作的位置是2200，設定的速度是100，可以算出至少要有七秒讓伺服機轉動到指定的位置，計算方式如下:

| 目標位置 － 開始位置 | ÷ 速度 ＝ 所需的延遲時間（秒）

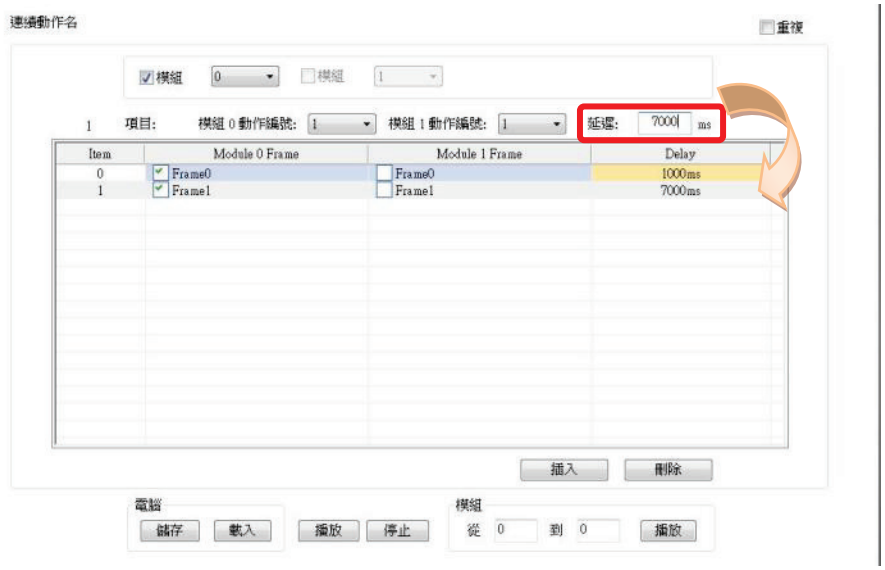
即:

| 2200 － 1500 | ÷ 100 ＝ 7（秒）

如果是使用時間模式，就只要直接填入所設定的時間就可以了。設定延遲時間的方式，要先點選所要設定的動作編號欄位，會看到選擇的欄位顏色整個改變:



然後在“延遲”後方的欄位，輸入所要設定的延遲時間，時間是以 ms 為單位，這時請輸入“7000”，在輸入的同時，就會看到點選的模組編號欄位，最後面的延遲時間也跟著被改變:



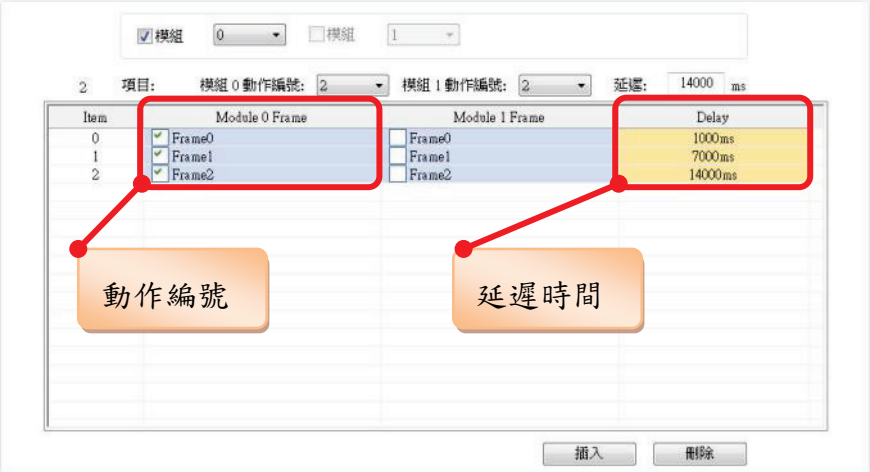
重覆上面的方式，利用插入，再增加一個動作編號為 2 的動作，並設定延遲時間為延遲十四秒，延遲時間的計算式如下:

| 目標位置 － 開始位置 | ÷ 速度 ＝ 所需的延遲時間（秒）

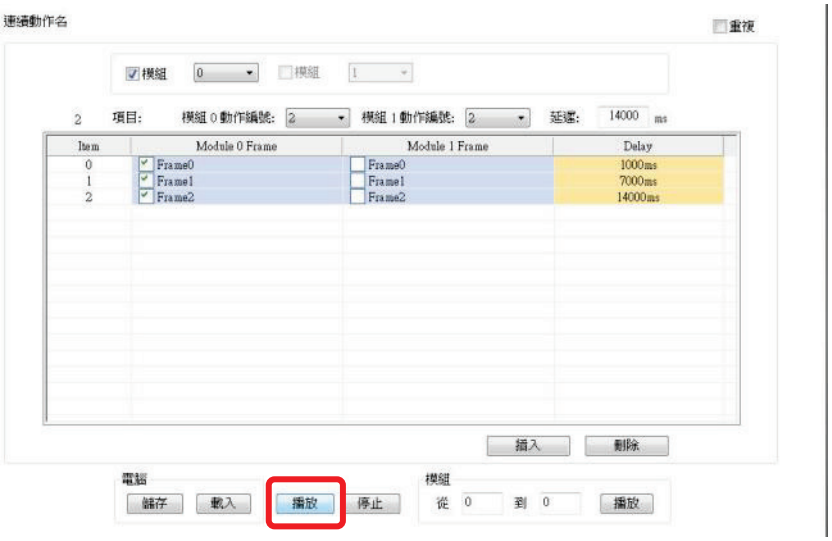
即:

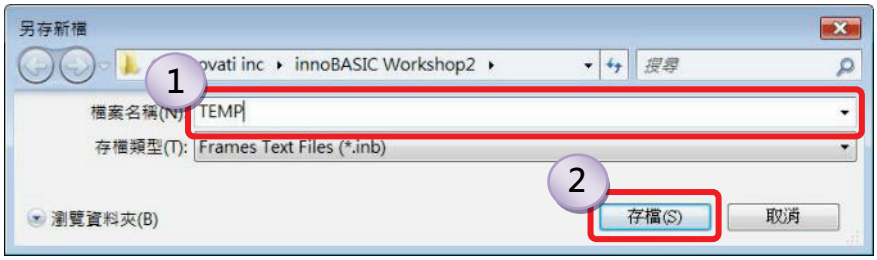
| 800- 2200 | ÷ 100 ＝ 14（秒）

完成的設定畫面如下，再次確認動作編號與延遲時間:

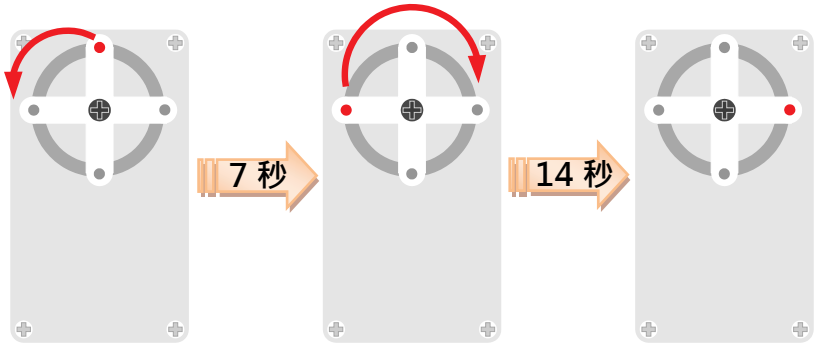


設定都正確後，點選下方的“播放”按鈕，畫面上會跳出一個另存新檔的視窗，輸入想要儲存的檔案名稱，再按下“存檔”按鈕:

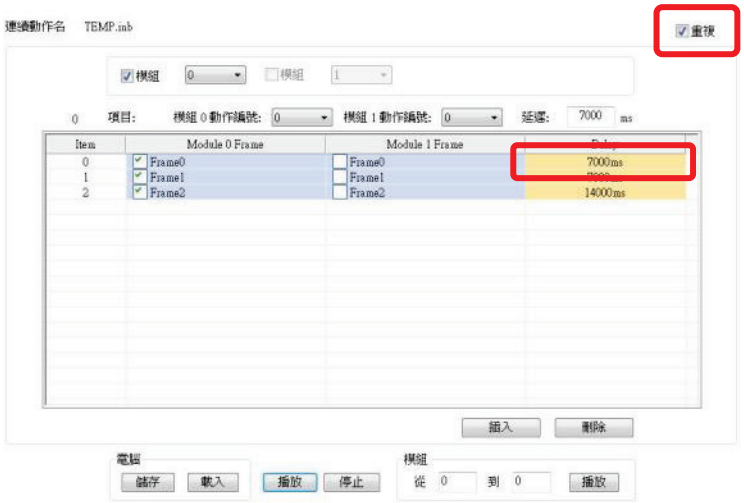


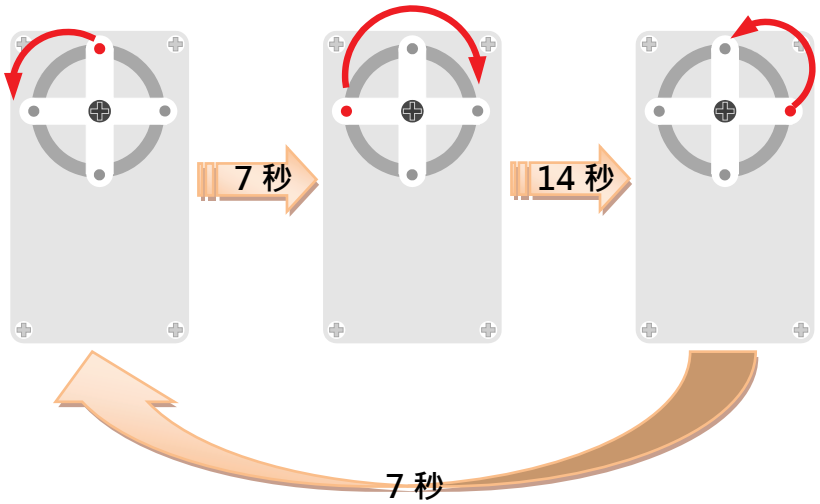


存檔並下載完成後，伺服機就會開始依序執行設定的三個動作:



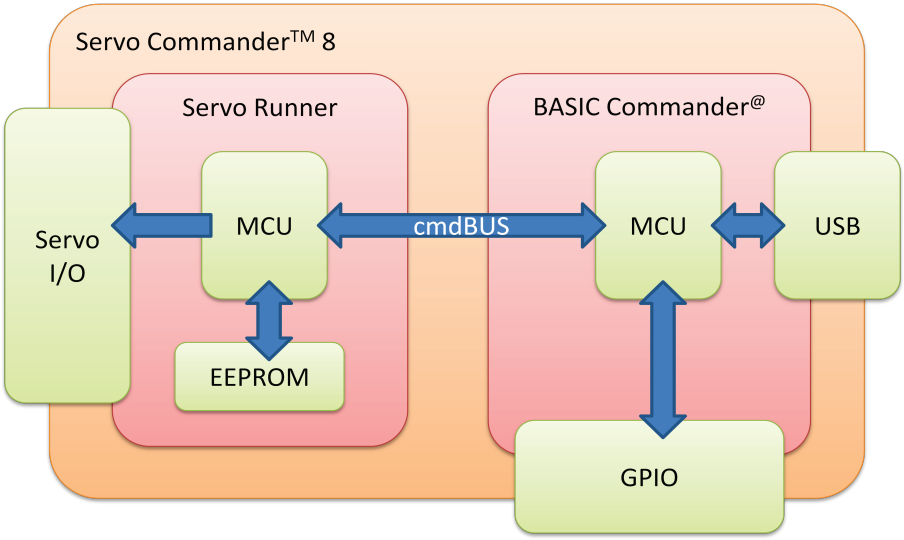
最後伺服機會停止在 800 的位置，如果希望伺服機可以重覆原先的動作，可以勾選右上角的“重複”選框，並且把動作編號 0 的延遲時間，更改為 7000，如此伺服機在執行到最後一個動作後，會再從頭執行第一個動作:





九、 伺服機控制程式

SC8 是包含了 BASIC Commander®與伺服機控制模組的整合控制板，BASIC Commander®是透過 cmdBUS™ 控制伺服機模組，但這一部份都隱藏在電路板中，SC8 的控制架構如下圖：



要控制伺服機模組的部份，除了前一章的動作編輯器，還可以透過直接編寫程式的方式，讓 SC8 中的 BASIC Commander®透過 cmdBUS™，傳送控制指令給伺服機模組，而且會比使用動作編輯器有更大的彈性，下面的各節就是介紹如何設計這樣的程式。

1. 宣告伺服機控制模組

在程式中如果想要操縱連接的模組，就要先設定模組在程式中的指定名稱，也就是宣告模組參數，宣告的方式如下：

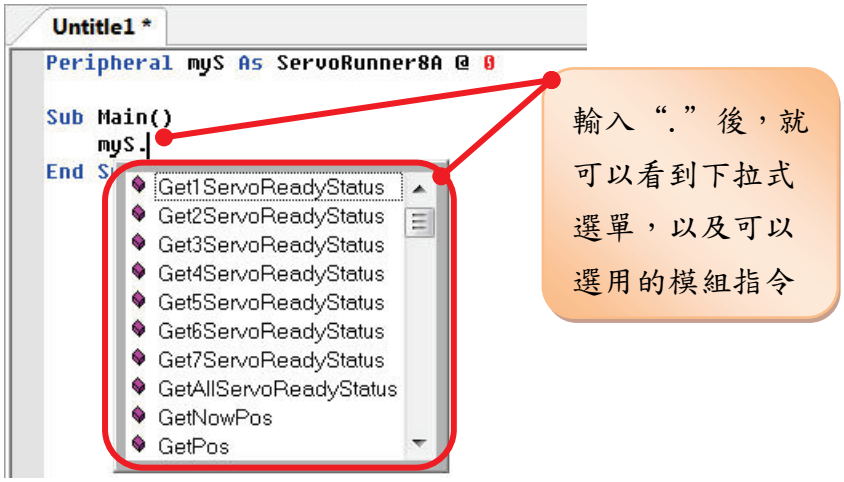
Peripheral *myS* As ServoRunner8A @ 0

粗體字的部份是關鍵字，不能更改，而粗斜體的部份，則是可以根據需求，更改為適合程式的參數名稱，模組的宣告，都必須在程式一開始的地方，而且不能重覆宣告。要宣告 SC8 的伺服機模組，要依照第五章的操作，先開啟一個空白新檔案，然後輸入如下的程式：

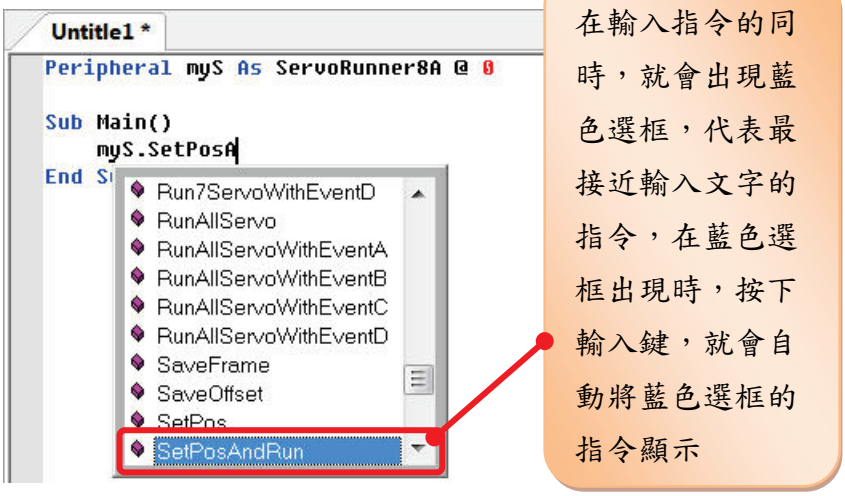
```
1  Peripheral myS As ServoRunner8A @ 0
2
3  Sub Main()
4  End Sub
```

2. 設定目標位置與啟動

在 innoBASIC 的程式中，每一個可以被宣告的模組，都有自己的專屬指令，只要先輸入模組在程式中的參數名稱，接著再後方輸入“.”，就會在後面看到下拉視窗，出現可以選用的指令，如果繼續輸入，下拉視窗就會跳到以輸入文字開頭的指令，這時可以用上下鍵選擇要使用的指令，也可以用滑鼠直接點選：



請先輸入 SetPosAndRun 指令，或是在輸入到“SetPosA”時，可以看到畫面上的下拉選單，會出現藍色的最接近指令，此時按下輸入鍵，就會自動把指令補齊為 SetPosAndRun：



最後輸入程式如下：

```
1  Peripheral myS As ServoRunner8A @ 0
2
3  Sub Main()
4      myS.SetPosAndRun(0, 1500)
5  End Sub
```

第 1 行: 宣告模組名稱為“myS”，並設定模組編號為“0”。
第 3 行: 主程式開始點。
第 4 行: 執行指令設定伺服機的目標位置，並開始轉動。
第 5 行: 程式結束。

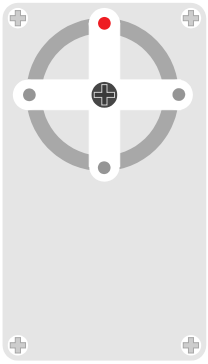
這邊使用到的指令為“SetPosAndRun”，指令說明如下:

SetPosAndRun *ID*, *Pos*

- *ID* – 常數或變數(0 ~ 7)，輸入要設定的伺服機編號。
- *Pos* – 常數或變數(500 ~ 2500)，輸入伺服機目標位置。

SetPosAndRun 指令會根據輸入的伺服機編號和目標位置，設定該伺服機轉動到設定的目標位置，並且立刻開始轉動，在沒有設定速度或時間模式的狀態下，伺服機會以最快的速度轉動。

指令輸入完成後，一樣按下下載按鈕，輸入要儲存的檔名，下載程式到 SC8 中，程式下載完畢，SC8 就會轉動在編號 0 的伺服機，到 1500 的目標位置，結果如下圖:



如果伺服機沒有正確轉動，可以檢查以下幾點:

- 伺服機連接線是否正確連接?
- 伺服機電源是否正確連接?
- 伺服機電源提供的電壓，是否在伺服機所能承受的電壓範圍內?
- USB 連接線是否正確連接 SC8 與電腦?
- 輸出視窗是否有顯示錯誤訊息?可以嘗試重新連接並再次下載。

更多關於 SC8 的伺服機控制指令介紹與說明，可以在使用手冊中看到，請至網站下載最新的使用手冊:
http://www.innovati.com.tw/website/down/html/?151.html

3. 以速度模式轉動

也可以用程式設定伺服機的轉動模式，只要在啟動伺服機前，先執行轉動模式的相關設定，再執行伺服機的轉動指令，伺服機就會依照設定的速度值轉動。要注意的是，在程式中，並不知道伺服機一開始的位置，所以設定的第一個位置，只能用最快速度轉動，另外每次執行轉動指令後，也要保留讓伺服機轉動到目標位置的時間，這次程式中需要增加使用下述指令:

Pause *Duration*

- *Duration* – 常數或變數，用來定義暫停動作的持續時間。持續時間單位為 1 毫秒。

Pause 為系統指令，會在前一個段落及後一個段落間放入延遲。Pause 命令允許放在程式的任何地方，讓使用者對程式執行速度有些許的控制力。持續時間單位為 1 毫秒。

SetPosSpd *ID*, *Pos*, *Speed*

- *ID* – 常數或變數(0 ~ 7)，輸入要設定的伺服機編號。
- *Pos* – 常數或變數(500 ~ 2500)，輸入伺服機目標位置。
- *Speed* – 常數或變數(0 ~ 65535)，設定伺服機往目標位置旋轉的速度。

SetPosSpd 指令會根據輸入的伺服機編號值，目標位置，及轉動模式的轉速值，設定該編號伺服機，下次執行伺服機轉動指令，所要轉動到的目標位置。旋轉速度的單位為μs 每秒。請注意這個指令並不會讓伺服機開始轉動，必須要再執行相關的轉動指令，才會讓伺服機依設定的模式，轉動到指令位置。

Run1Servo *ID*

- *ID* – 常數或變數(0 ~ 7)，輸入要啟動的伺服機編號。

Run1Servo 指令會讓指定編號的伺服機，開始轉動到指令的目標位置，並且依設定的模式轉動，如果目標位置與現在位置相同，則伺服機不會改變位置。

輸入下面的範例程式:

```
1  Peripheral myS As ServoRunner8A @ 0
2
3  Sub Main()
4      myS.SetPosAndRun(0, 1500)
5      Pause 1000
6      myS.SetPosSpd(0, 2000, 100)
7      myS.Run1Servo(0)
8  End Sub
```

第 1 行: 宣告模組名稱為 “myS” ，並設定模組編號為 “0” 。

第 3 行: 主程式開始點。

第 4 行: 執行指令設定伺服機的目標位置為 1500，並開始轉動。

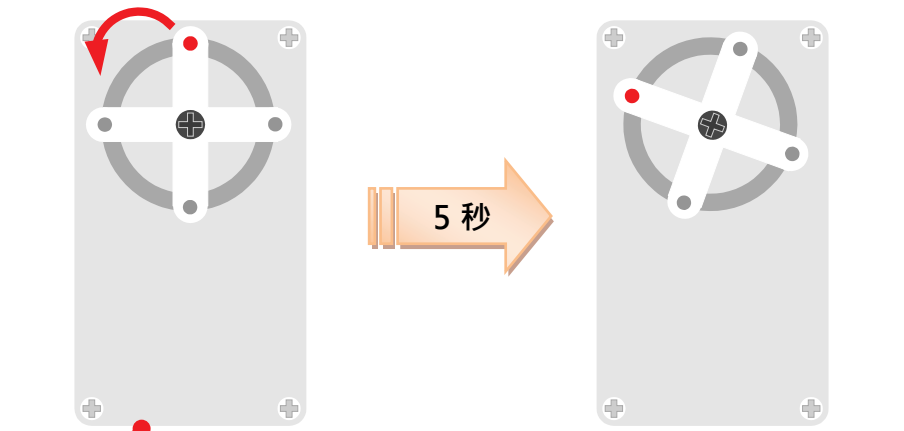
第 5 行: 暫停一秒，讓伺服機轉動到指定的位置。

第 6 行: 執行指令設定編號 0 的伺服機目標位置為 2000，並且轉動模式為以指定速度，每秒 100 μs 轉動。

第 7 行: 執行指令讓編號 0 的伺服機開始轉動。

第 8 行: 程式結束。

下載程式執行後，伺服機動作如下:



伺服機會先以最快速度，轉動到 1500 的位置後停止，接著再次轉動往 2000 的位置，第一次開始轉動，到第二次開始轉動，間隔約一秒的時間

4. 以時間模式轉動

另外一種設定伺服機的轉動模式，就是讓伺服機以固定速度，使用規定的時間，到達目標位置，方法一樣是在啟動伺服機前，先執行設定指令，再執行伺服機的轉動指令。這次程式中需要新增加的指令如下:

SetPosTime *ID*, *Pos*, *Time*

- *ID* – 常數或變數(0 ~ 7)，輸入要設定的伺服機編號。
- *Pos* – 常數或變數(500 ~ 2500)，輸入伺服機目標位置。
- *Time* – 常數或變數(0 ~ 65535)，設定伺服機到達目標位置，所要花的時間。

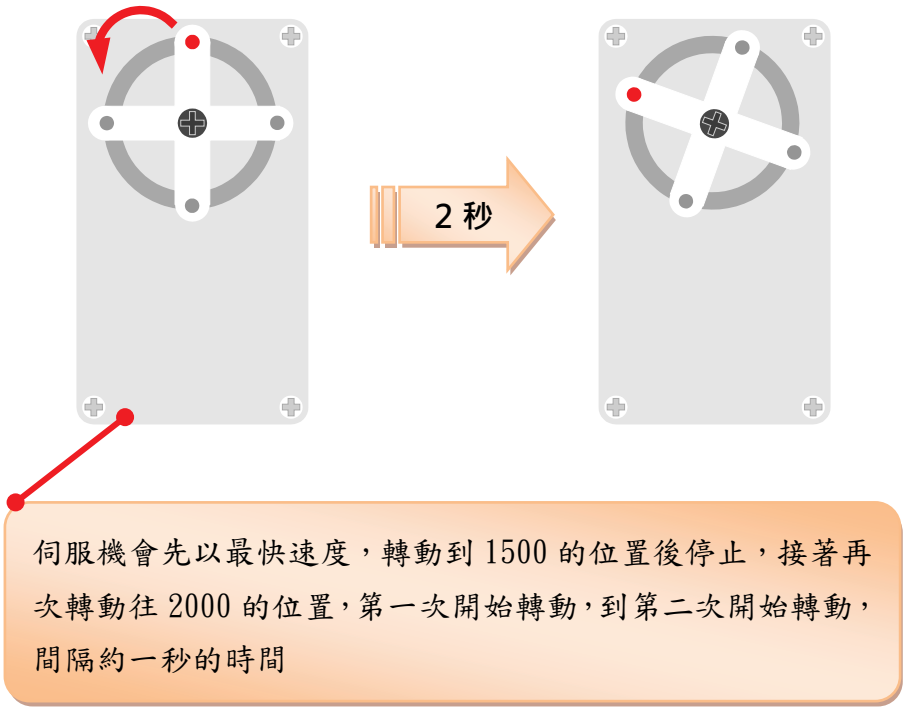
SetPosTime 指令會根據輸入的伺服機編號值，目標位置，及時間值，設定該編號伺服機，下次執行伺服機轉動指令，所要轉動到的目標位置，以及轉動所要花的時間。轉動所花時間的輸入值，單位為 ms。請注意這個指令並不會讓伺服機開始轉動，必須要再執行相關的轉動指令，才會讓伺服機依設定的模式，轉動到指令位置。

輸入下面的範例程式:

```
1  Peripheral myS As ServoRunner8A @ 0
2
3  Sub Main()
4      myS.SetPosAndRun(0, 1500)
5      Pause 1000
6      myS.SetPosTime(0, 2000, 2000)
7      myS.Run1Servo(0)
8  End Sub
```

第 1 行: 宣告模組名稱為“myS”，並設定模組編號為“0”。
第 3 行: 主程式開始點。
第 4 行: 執行指令設定伺服機的目標位置為 1500，並開始轉動。
第 5 行: 暫停一秒，讓伺服機轉動到指定的位置。
第 6 行: 執行指令設定編號 0 的伺服機目標位置為 2000，並且轉動所花的時間為兩秒。
第 7 行: 執行指令讓編號 0 的伺服機開始轉動。
第 8 行: 程式結束。

下載程式執行後，伺服機動作如下:



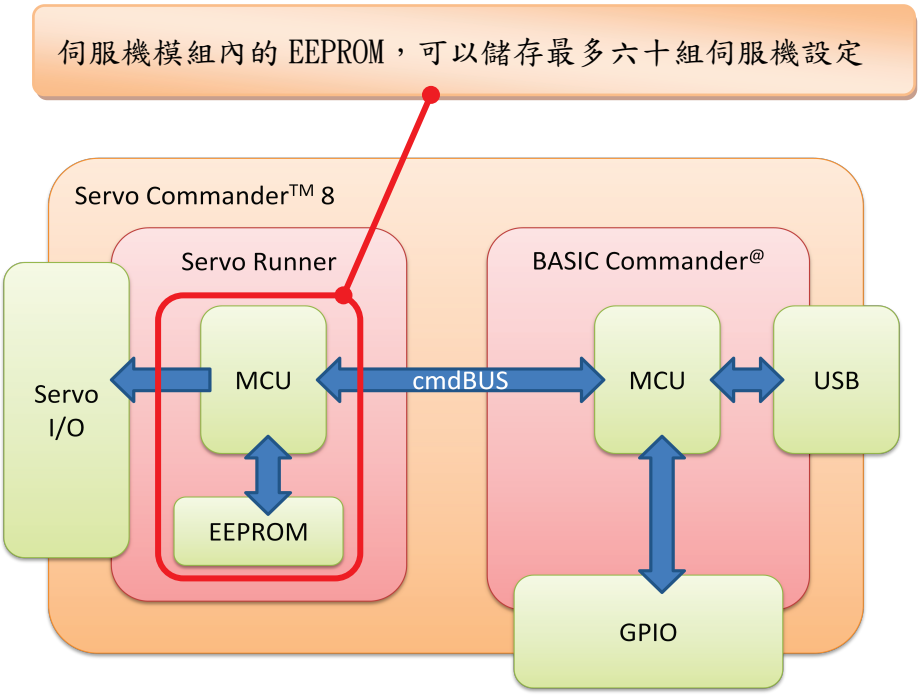
速度模式與時間模式的比較:

	速度模式	時間模式
到達目標位置時間	須要根據開始位置與目標位置計算	根據輸入值決定
轉動速度	根據輸入值決定	需要用轉動距離與轉動時間計算
使用時機	希望所有伺服機有一致的轉速，或是希望伺服機能夠以特定轉速轉動時	希望不同起始位置的伺服機，能夠同時到達不同位置時

如果沒有指定速度或時間模式，伺服機都會以最快轉速，轉動到指定的目標位置。

5. 儲存與讀取設定動作

SC8 可以將動作設定，儲存在伺服機模組內部的記憶體中，並且能透過模組指令，在需要的時候，再讀出儲存的動作，可以避免不斷重複設定相同的位置與轉動模式，也可以節省寶貴的程式記憶體空間。



可以用 SaveFrame 與 LoadFrame，儲存與讀取現在的動作位置，只要輸入對應的動作編號就可以，指令格式如下:

SaveFrame *Num*

- *Num* – 常數或變數(0 ~ 59)，輸入要儲存的動作編號。

SaveFrame 指令將現在的伺服機目標位置，和轉動模式，儲存到指定的動作編號中。想要讀出儲存的動作資料，只要另外執行 LoadFrame 就可以，就可以避免重新一個一個設定伺服機。

LoadFrame *Num*

- *Num* – 常數或變數(0 ~ 59)，輸入要讀取的動作編號。

LoadFrame 指令會讀取指定編號的動作資料，包括各個伺服機的目標位置，轉動模式，成為現在伺服機的目標位置與轉動模式，接著只要

再執行啟動的相關指令，就可以讓伺服機，轉動到動作編號所儲存的目標位置。

可以用下面的程式，儲存四組伺服機動作:

```
1  Peripheral myS As ServoRunner8A @ 0
2
3  Sub Main()
4      myS.SetPosAndRun(0, 1500)
5      myS.SaveFrame(0)
6      Pause 1000
7      myS.SetPosTime(0, 2000, 2000)
8      myS.SaveFrame(1)
9      myS.SetPosTime(0, 1000, 5000)
10     myS.SaveFrame(2)
11     myS.SetPosTime(0, 1500, 2500)
12     myS.SaveFrame(3)
13     Debug "下載完成"
14 End Sub
```

第 1 行: 宣告模組名稱為“myS”，並設定模組編號為“0”。

第 3 行: 主程式開始點。

第 4 行: 執行指令設定伺服機的目標位置為 1500，並開始轉動。

第 5 行: 執行指令儲存動作於動作編號 0 的位置。

第 6 行: 暫停一秒，讓伺服機轉動到指定的位置。

第 7 行: 執行指令設定編號 0 的伺服機目標位置為 2000，並且轉動所花的時間為兩秒。

第 8 行: 執行指令儲存動作於動作編號 1 的位置。

第 9 行: 執行指令設定編號 0 的伺服機目標位置為 1000，並且轉動所花的時間為五秒。

第 10 行: 執行指令儲存動作於動作編號 2 的位置。

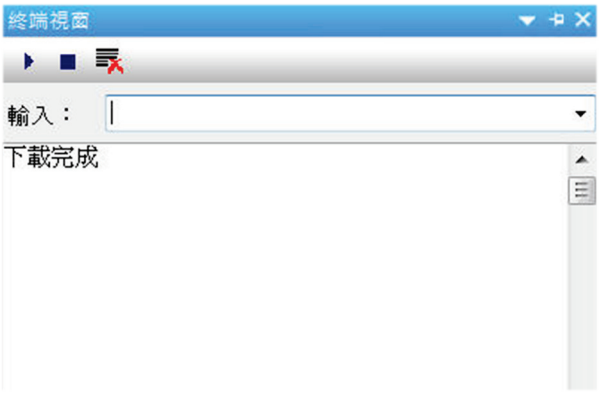
第 11 行: 執行指令設定編號 0 的伺服機目標位置為 1500，並且轉動所花的時間為 2.5 秒。

第 12 行: 執行指令儲存動作於動作編號 3 的位置。

第 13 行: 在終端視窗輸出“下載完成”。

第 14 行: 程式結束。

執行程式並下載到 SC8，伺服機只會轉動一次到 1500 的位置，就不會動作了，接著在終端視窗會出現“下載完成”的訊息，代表設定的動作，都已經設定到指定的動作編號裡。



可以再用下面的範例程式，執行儲存的動作:

```
1   Peripheral myS As ServoRunner8A @ 0
2
3   Sub Main()
4       myS.LoadFrame(0)
5       myS.Run1Servo(0)
6       Pause 1000
7       myS.LoadFrame(1)
8       myS.Run1Servo(0)
9       Pause 2000
10      myS.LoadFrame(2)
11      myS.Run1Servo(0)
12      Pause 5000
13      myS.LoadFrame(3)
14      myS.Run1Servo(0)
15      Pause 2500
16  End Sub
```

第 1 行: 宣告模組名稱為“myS”，並設定模組編號為“0”。

第 3 行: 主程式開始點。

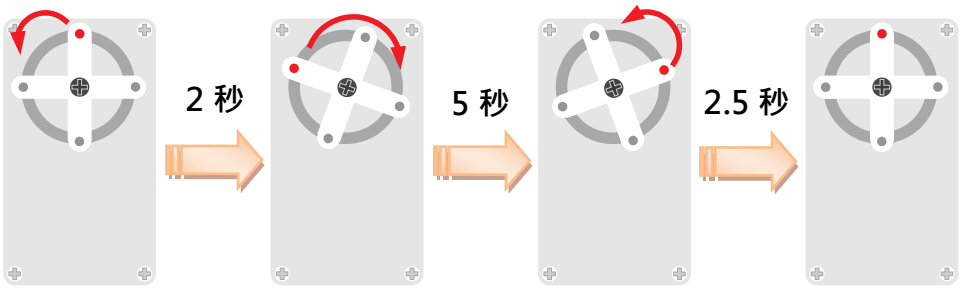
第 4 行: 執行指令讀取動作編號為 0 的資訊。

第 5 行: 執行指令啟動編號 0 的伺服機，轉動到目標位置。

第 6 行: 暫停一秒，讓伺服機轉動到指定的位置。

第 7 行: 執行指令讀取動作編號為 1 的資訊。
第 8 行: 執行指令啟動編號 0 的伺服機，轉動到目標位置。
第 9 行: 暫停二秒，讓伺服機轉動到指定的位置。
第 10 行: 執行指令讀取動作編號為 2 的資訊。
第 11 行: 執行指令啟動編號 0 的伺服機，轉動到目標位置。
第 12 行: 暫停五秒，讓伺服機轉動到指定的位置。
第 13 行: 執行指令讀取動作編號為 3 的資訊。
第 14 行: 執行指令啟動編號 0 的伺服機，轉動到目標位置。
第 15 行: 暫停 2.5 秒，讓伺服機轉動到指定的位置。
第 16 行: 程式結束。

下載執行後，可以看到伺服機如下圖動作:



藉由 SaveFrame 可以預先儲存設計好的動作，再另外以 LoadFrame 指令讀取，就能省去重複設定伺服機目標位置的指令，也可以讓程式更簡潔，範例中都是只有使用一個伺服機，但動作的儲存，是可以一次把八個伺服機的目標位置與轉動模式，都儲存到一個動作編號中，在操作多個伺服機時，可以有更大的幫助。

6. 從終端視窗設定轉動位置

在一開始的程式有介紹，可以在終端視窗顯示訊息，也可以從終端視窗輸入資料給 SC8，把這樣的功能跟伺服機控制指令結合，就可以從終端視窗控制伺服機的位置，達到類似動作編輯器的效能，也等於可以製作專屬於自己，不同效能的動作編輯器。

在程式中會用到一個迴圈指令，Do…Loop，可以讓這兩行指令中間所夾的程式，反覆執行，也就是程式執行到尾端 Loop 的地方，又會從 Do 的地方開始執行，詳細的指令說明可以在系統使用手冊中查閱，詳細的程式內容如下:

```
1  Peripheral myS As ServoRunner8A @ 0
2
3  Sub Main()
4      Dim wPos As Word
5
6      Debug CLS, "伺服機位置: "
7
8      Do
9          Debugin wPos
10         myS.SetPosAndRun(0, wPos)
11         Debug CSRXY(13, 1), %DEC4R wPos
12     Loop
13 End Sub
```

第 1 行: 宣告模組名稱為“myS”，並設定模組編號為“0”。

第 3 行: 主程式開始點。

第 4 行: 宣告參數“wPos”，儲存目標位置。

第 6 行: 清除終端視窗訊息，並顯示“伺服機位置”在終端視窗。

第 8 行: 迴圈開始位置。

第 9 行: 讀取輸入值，儲存到參數“wPos”中。

第 8 行: 顯示參數值在終端視窗。

第 10 行: 執行指令設定編號 0 的伺服機的目標位置，並轉動。

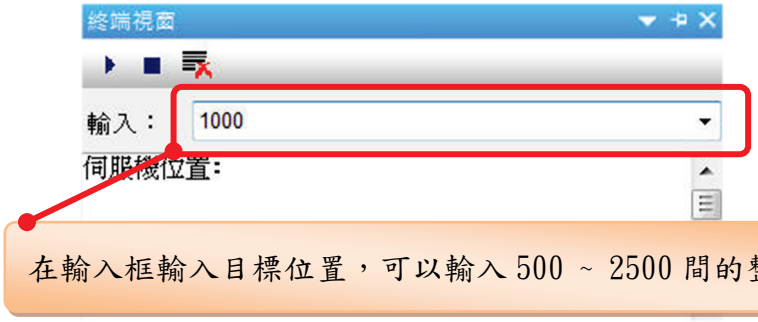
第 11 行: 將輸入的設定值，顯示在終端視窗第一列第 13 行的位置。

第 12 行: 迴圈返回點。

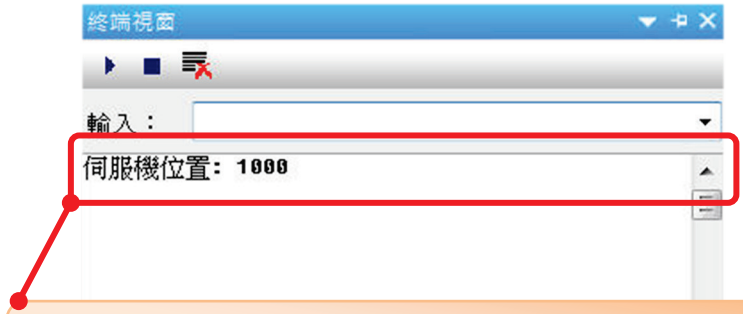
第 12 行: 程式結束。

下載程式並執行後，會在終端視窗看到訊息，此時可以在終端視窗輸入希望轉動的位置。

終端視窗會顯示伺服機位置訊息



在輸入框輸入目標位置，可以輸入 500 ~ 2500 間的整數值



伺服機的位置會顯示在終端視窗，可以再輸入其他數值，轉動伺服機到其他位置

十、 更多應用

除了前面幾章介紹的，操作伺服機的應用，SC8 還可以與其他不同功能的應用模組做結合，例如與 LCD 顯示模組結合，就可以不用與電腦連接，直接獨立顯示訊息，或是與鍵盤輸入模組結合，就可以取得鍵盤的按鍵值，不用從電腦端輸入資料。甚至整合這兩個模組，就可以有一個小型的計算機，也可以顯示伺服機的位置，與輸入伺服機要轉動的位置，更多模組的介紹，可以到網路上查詢:

http://www.innovati.com.tw/website/cp/class/?2.html

如果想控制機器人，可以參考下面的影片:

http://www.youtube.com/watch?v=Ue4GCSBu5j0&feature=player_embedded

影片中的機器人是以紙摺出來，用 SC8 當核心，控制四個伺服機，讓機器人可以扭腰擺手與轉頭，並增加了紅外線接收器，能接收紅外線遙控器的訊號，控制機器人的動作，另外還加上了蜂鳴器，播放寫在程式中的音樂，而眼睛上的兩個 LED，可以設定亮滅。紙機器人沒有用到額外的模組，全都是使用 IO 腳位完成控制額外的元件。

其他還有六軸的半身機器人，也可以使用 SC8 完成控制:

http://www.youtube.com/watch?v=Z0q8sDKfmzk&feature=player_embedded

在熟練了較少軸數的控制以後，還可已再試著挑戰更多軸數的變化與應用，例如十二軸的半身人形機器人:

http://www.youtube.com/watch?v=YKpm16YoF-Q&feature=player_embedded

或是十六軸的全身人形機器人:

http://www.youtube.com/watch?v=SOrZUK1i34g&feature=player_embedded

歡迎立刻造訪我們的網站，查看更豐富的應用展示!

http://resource.innovati.com.tw/ying-pian-zhuan-qu#TOC-3