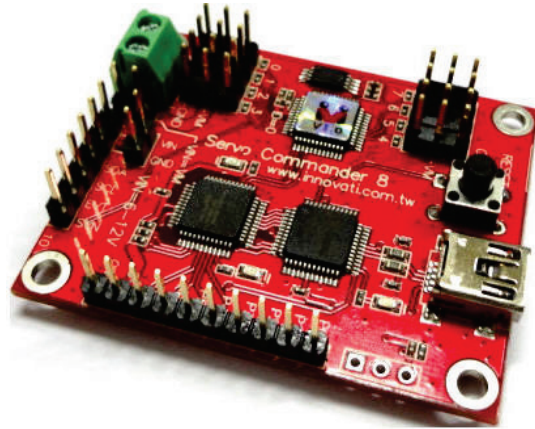


Servo Commander 8

八組伺服機輸出控制模組

版本: V1.0



產品介紹: 利基 Servo Commander 8 模組結合了簡易的 BASIC Commander - BC1 架構與一個 Servo Runner 8A，使用者能大幅縮小模組所需要面積，減少連接線材，並保留完整的 Servo Runner 8A 功能: 可以一次控制八個伺服機，運用整合好的指令，讓使用者可以直接設定以固定速度或共同時間，決定伺服機的移動方式。設有多達 60 組記憶體可以儲存伺服機目標位置與移動方式(速度或時間)，讓各種動作輕易組合完成。

應用方向:

- 各種伺服機的操作與應用，包括機械手臂，機器人關節。
- 小型化的各式伺服機設計相關應用。

產品特色:

- 保留 BC1 的部分功能與 Servo Runner 8A 的八組操控硬體介面*¹。
- BC1 與 Servo Runner 8A 連接之 cmdBUS 內建，無須額外連接。
- 伺服機與電子元件共用電源，可以用單一電源提供伺服機與電路動作*²。
- 八組伺服機輸出介面，可同時控制八組伺服機。
- 可控制伺服機位置由 0.5 ms 至 2.5 ms。
- 軟體微調指令，不用機械拆裝，僅由軟體設定，就可以達到微調各個伺服機轉向角度的目的，可設定-128~127 μ S。
- 程式可以設定伺服機轉向速度，使用者可根據需求設定多段的伺服機轉向速度。
- 使用者可以設定一個共同時間，讓各個伺服機在同時間達到不同的轉向角度。
- 內建 60 組伺服機記憶空間，每組可以儲存目前設定好的八個伺服機目標位置，與速度或時間參數，在需要時直接呼叫，可以免去重覆設定的動作，也可以快速組合出多樣化的效果。
- 設有四組事件提醒，讓使用者可以在判斷動作完成後，自動進行下一項操作。各事件可以設定任意 1~8 個伺服機作為判斷依據。
- 各種狀態取得指令，使用者可以隨時確認伺服機是否動作完成，取得現在位置，目標位置，微調參數，或是設定的時間與速度值。
- 解析度可達 2 μ S。

*1 BC1 僅可控制 Port0(P0~P7)。

*2 BC1 可以接受 6~12V 的電壓輸入，請選擇可搭配此電壓範圍的伺服機。

 **注意!** 本使用手冊介紹以伺服機相關控制為主，BASIC Commander 相關指令與系統設定，請參考“利基 BASIC Commander 使用手冊”。需要執行伺服機操控相關指令時，模組編號請設定為 0

連接方式:在模組上有以三個腳位為一組，共八組伺服機連接座，提供各伺服機的控制訊號與電源，根據伺服機的對應腳位連接就可以動作(如右圖)。提供給 Servo Commander 8 的電源連接方式如圖 1。連接電源前，請確定伺服機所需的電流與電壓，以免伺服機產生不正常動作，損毀伺服機。

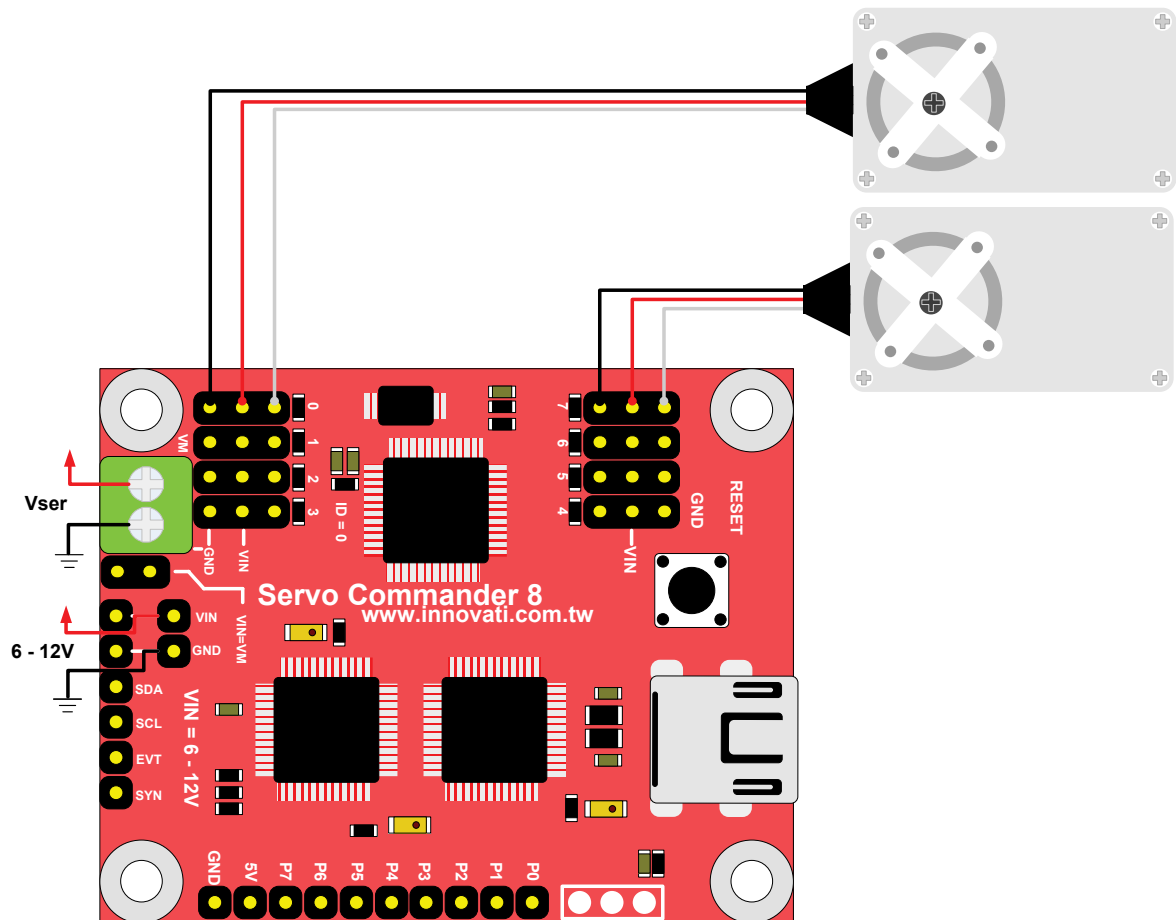
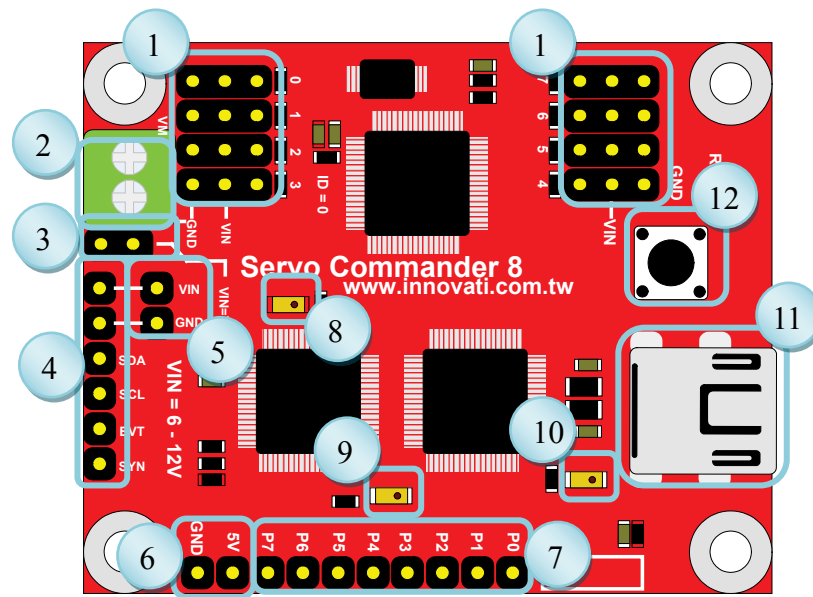


圖 1: 伺服機與電子元件使用相同電源



使用圖中的連接方式，請確認連接的電源提供伺服機的電壓值，是在伺服機所能承受的電壓範圍內，以免造成伺服機受損

產品規格:



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。

圖 2: 模組腳位與元件介紹

使用電流量: 32.3 mA (Servo Commander 8 模組未接上伺服機於 VIN 的耗電量)

模組尺寸: 46.8 x 54.8 (釐米)

操作注意事項: 請確認所連接之伺服機所需之電壓範圍，與所需電流大小，選擇合適之電源，由 Vser 連接正確之電源。

伺服機的 Pulse 腳位與模組連接須符合圖 1 規範→(此為模組所能動作之範圍)

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{IN} =7.5V	Conditions				
V _{OH}	I/O Port output high voltage	-	No loading	-	5	-	V
V _{OL}	I/O Port output low voltage	-	No loading	-	0	-	V
I _{OL}	I/O Port Sink Current	-	V _{load} =0.1V _{OH}	10	20	-	mA
I _{OH}	I/O Port Source Current	-	V _{load} =0.9V _{OH}	-5	-10	-	mA

表 1: Servo Commander 8 模組電流限制 (Test Temperature=25°C)

模組操作溫度 0 °C ~ 70 °C (伺服機之操作溫度請於伺服機規格確認)

模組儲存溫度 -50 °C ~ 125 °C

指令表:

下面的指令表是專供控制 Servo Commander 8 模組的各種指令，必要輸入的指令名稱與參數，以粗底或粗斜體表示，粗體的文字在輸入時請不要更改，粗斜體的文字請自行定義適當格式的參數填入。輸入時請注意 innoBASIC™ Workshop 大寫與小寫會視為相同字。

在執行 Servo Commander 8 指令前，請先於程式開頭定義對應參數與編號，例:

Peripheral *ModuleName* As ServoRunner8A @ 0

指令格式	指令功能
伺服機位置設定相關指令	
SetPos (<i>SerID</i>, <i>Pos</i>)	以 <i>SerID</i> 指定所要操作的伺服機，請輸入 0~7 之間的整數值，設定該伺服機所要到達的位置為 <i>Pos</i> ，若設定 Run 則立刻開始動作 (<i>Pos</i> 請輸入 499~2500 之間的整數值，單位為 μS，若設定超過此範圍的值，這次設定的命令將不被執行)
SetPosAndRun(<i>SerID</i>, <i>Pos</i>)	
SetPosSpd(<i>SerID</i>, <i>Pos</i>, <i>Spd</i>)	以 <i>SerID</i> 指定所要操作的伺服機，請輸入 0~7 之間的整數值，設定伺服機所要到達的位置為 <i>Pos</i> ， <i>Pos</i> 請輸入 499~2500 之間的整數值，並且設定伺服機以 <i>Spd</i> 移動伺服機，若設定 Run 則立刻開始動作(<i>Spd</i> 請輸入 0~65535 之間的整數值，0 代表全速，值越大移動越快，單位為 μS/S)
SetPosSpdAndRun(<i>SerID</i>, <i>Pos</i>, <i>Spd</i>)	
SetPosTime(<i>SerID</i>, <i>Pos</i>, <i>Time</i>)	以 <i>SerID</i> 指定所要操作的伺服機，設定伺服機所要到達的位置為 <i>Pos</i> ， <i>Pos</i> 請輸入 499~2500 之
SetPosTimeAndRun(<i>SerID</i>, <i>Pos</i>,	

<i>Time</i>)	間的整數值，並且設定伺服機需要以固定速度，花 <i>Time</i> 所設定的時間到達指定位置，若設定 <i>Time</i> 則立刻開始動作(<i>Time</i> 請輸入 0~65535 之間的整數值， <i>Time</i> 若設為 0 則會以全速移動， <i>Time</i> 的單位為毫秒[ms]，如果設定時間太短，伺服機就以全速移動)
伺服機啟動相關指令	
Run1Servo(<i>SerID</i>)	根據所設定的 <i>SerID</i> ，請輸入 0~7 之間的整數值，啟動各伺服機執行所設定的動作，若起動伺服機時，只進行位置的設定，而未設定速度或時間，伺服機會用最快的速度動作
...	
Run7Servo(<i>SerID1</i> , ..., <i>SerID7</i>)	
RunAllServo()	
Run1ServoWithEventA(<i>SerID</i>)	根據所設定的 <i>SerID</i> ，請輸入 0~7 之間的整數值，啟動各伺服機執行所設定的動作，並且在動作完成後，產生事件 A
...	
Run7ServoWithEventA(<i>SerID1</i> , ..., <i>SerID7</i>)	
RunAllServoWithEventA()	
Run1ServoWithEventB(<i>SerID</i>)	根據所設定的 <i>SerID</i> ，請輸入 0~7 之間的整數值，啟動各伺服機執行所設定的動作，並且在動作完成後，產生事件 B
...	
Run7ServoWithEventB(<i>SerID1</i> , ..., <i>SerID7</i>)	
RunAllServoWithEventB()	
Run1ServoWithEventC(<i>SerID</i>)	根據所設定的 <i>SerID</i> ，請輸入 0~7 之間的整數值，啟動各伺服機執行所設定的動作，並且在動作完成後，產生事件 C
...	
Run7ServoWithEventC(<i>SerID1</i> , ..., <i>SerID7</i>)	
RunAllServoWithEventC()	
Run1ServoWithEventD(<i>SerID</i>)	根據所設定的 <i>SerID</i> ，請輸入 0~7 之間的整數值，啟動各伺服機執行所設定的動作，並且在動作完成後，產生事件 D
...	
Run7ServoWithEventD(<i>SerID1</i> , ..., <i>SerID7</i>)	
RunAllServoWithEventD()	
伺服機停止相關指令	
Pause1Servo(<i>SerID</i>)	根據所設定的 <i>SerID</i> ，請輸入 0~7 之間的整數值，暫停各伺服機正在執行的動作
...	
Pause7Servo(<i>SerID1</i> , ..., <i>SerID7</i>)	
PauseAllServo()	
Stop1Servo (<i>SerID</i>)	根據所設定的 <i>SerID</i> ，請輸入 0~7 之間的整數值，停止各伺服機正在執行的動作，同時停止供應給伺服機電流，此時伺服機若受外力移動，將改變位置
...	
Stop7Servo (<i>SerID1</i> , ..., <i>SerID7</i>)	
StopAllServo ()	

伺服機狀態與記憶相關指令	
Get1ServoReadyStatus(<i>SerID</i>, <i>Status</i>)	取得 <i>SerID</i> 指定之伺服機狀態，請輸入 0~7 之間的整數值，並將狀態值存放於 <i>Status</i> 中 (若所指定的伺服機中，有任一個還沒有到達指定位置，所傳回的狀態將會是 0，當所有伺服機皆到達指定位置，傳回狀態則為 1)
...	
Get7ServoReadyStatus(<i>SerID1</i>, ..., <i>SerID7</i>, <i>Status</i>)	
GetAllServoReadyStatus(<i>Status</i>)	
GetNowPos (<i>SerID</i>, <i>Pos</i>)	取得 <i>SerID</i> ，請輸入 0~7 之間的整數值，所指定之伺服機現在位置，存放於 <i>Pos</i> 中， <i>SerID</i> 請輸入 0~7 之間的整數值， <i>Pos</i> 會回傳 499~2500 之間的整數值
GetPos(<i>SerID</i>, <i>Pos</i>)	取得 <i>SerID</i> ，請輸入 0~7 之間的整數值，所指定之伺服機目標位置，存放於 <i>Pos</i> 中， <i>SerID</i> 請輸入 0~7 之間的整數值， <i>Pos</i> 會回傳 499~2500 之間的整數值
GetPosOffset(<i>SerID</i>, <i>Offset</i>)	取得 <i>SerID</i> ，請輸入 0~7 之間的整數值，所指定之伺服機微調值，存放於 <i>Offset</i> 中 (<i>Offset</i> 單位為微秒[μS]，範圍可從-128 到 127)
GetSpdAndTime(<i>SerID</i>, <i>Type</i>, <i>Value</i>)	取得 <i>SerID</i> ，請輸入 0~7 之間的整數值，所指定伺服機所設定的移動方式 <i>Type</i> ，與設定值 <i>Value</i> ，若是所設定的移動方式為速度，則 <i>Type</i> 取回值為 1，若為時間則 <i>Type</i> 為 0， <i>Value</i> 會回傳 0~65535 之間的整數值
LoadFrame(<i>FrameID</i>)	讀取 <i>FrameID</i> 所指定記憶位址的紀錄值，作為現在伺服機的目標位置與移動方式， <i>FrameID</i> 請輸入 0~59 之間的整數值
LoadOffset()	將 EEPROM 中儲存的 Offset 值，取代現有的設定值
SaveFrame(<i>FrameID</i>)	將目前所設定的各伺服機位置，儲存於 <i>FrameID</i> 所指定的記憶體中， <i>FrameID</i> 請輸入 0~59 之間的整數值
SaveOffset()	將現在設定的 Offset 值，儲存到 EEPROM 中
SetPosOffset(<i>SerID</i>, <i>Offset</i>)	設定 <i>SerID</i> ，請輸入 0~7 之間的整數值，所指定的伺服機微調值為 <i>Offset</i> ， <i>Offset</i> 請輸入 -128~127 之間的整數值

模組提供應用事件:

事件名稱 (Event)	啟動條件
ServoPosReadyEventA	執行 Run1ServoWithEventA 類別的指令(1 可為 1~7 或 All)，當所有指定伺服機都到達指定位置，就會啟動此事件
ServoPosReadyEventB	執行 Run1ServoWithEventB 類別的指令(1 可為 1~7 或 All)，

	當所有指定伺服機都到達指定位置，就會啟動此事件
ServoPosReadyEventC	執行 Run1ServoWithEventC 類別的指令(1 可為 1~7 或 All)，當所有指定伺服機都到達指定位置，就會啟動此事件
ServoPosReadyEventD	執行 Run1ServoWithEventD 類別的指令(1 可為 1~7 或 All)，當所有指定伺服機都到達指定位置，就會啟動此事件

範例程式:

```

' 範例程式中的伺服機位置是以多數伺服機的範圍設定，
' 請根據所使用的伺服機可設定的位置做調整，以免造成伺服機毀損

Peripheral mySer As ServoRunner8A @ 0 ' 設定模組編號為 0，注意!使用 Servo Commander
                                     ' 8 上的伺服機相關指令，一定要設定編號為 0

Dim EventEnd As Byte ' 儲存事件處理完成的判斷參數
Dim i As Byte ' 儲存迴圈的判斷值
Dim SerStatus As Byte ' 儲存 Servo 的 Status 值

Sub Main() ' 主程式
    mySer.SetPosOffset(0, 0) ' 設定 Servo0 的微調值為 0
    mySer.SetPosAndRun(0, 1500) ' 啟動 Servo0 移動到 1500 的位置
    Pause 1000 ' 暫停一段時間讓伺服機移動到指定位置

    mySer.SetPos(0, 2200) ' 指定 Servo0 目標位置為 2200
    mySer.SaveFrame(0) ' 將目前設定的伺服機動作存到 Frame0 中
    mySer.Run1Servo(0) ' 讓 Servo0 開始動作
    Pause 500

    mySer.SetPosSpdAndRun(0, 700, 1000) ' 啟動 Servo0 以速度 1000 移動到 700 的位置
    Pause 2000
    mySer.SetPosTimeAndRun(0, 2200, 1000) ' 啟動 Servo0 花一秒的時間移動到 2200 的位置
    Pause 1000

    EventEnd=0
    mySer.SetPosTime(0, 700, 1000) ' 設定 Servo0 花一秒的時間移動到 700 的位置
    mySer.SaveFrame(1) ' 將目前設定的伺服機動作存到 Frame1 中
    mySer.Run1ServoWithEventA(0) ' 啟動 Servo0 並於動作結束產生 EventA

Do
    Pause 1
Loop Until EventEnd=1

```

```

' 下面的迴圈反覆讀取 Frame0 與 Frame1 的設定值，再啟動 Servo0 執行
' Frame0 儲存的位置為 2200，Frame1 儲存的位置為 700
' Servo0 將在這兩個位置間來回移動四次
For i=0 To 3
    mySer.LoadFrame(1) ' 讀取儲存於 Frame1 的設定值
    mySer.Run1Servo(0)
    Pause 1000
    mySer.LoadFrame(0) ' 讀取儲存於 Frame0 的設定值
    mySer.Run1Servo(0)
    Pause 1000
Next

mySer.SetPosAndRun(0, 1500)
' 下面的迴圈反覆執行讀取 Status 的動作，
' 在確定動作結束後，才會跳出迴圈
Do
    mySer.Get1ServoReadyStatus(0, SerStatus) ' 讀取 Servo0 的狀態存於 SerStatus 中
Loop Until SerStatus>0

End Sub

Event mySer.ServoPosReadyEventA()
    mySer.SetPosAndRun(0, 2200)
    Pause 1000
    EventEnd=1
End Event

```

附錄

1. 已知問題:

- 無

※版本標示於模組上的雷射貼紙