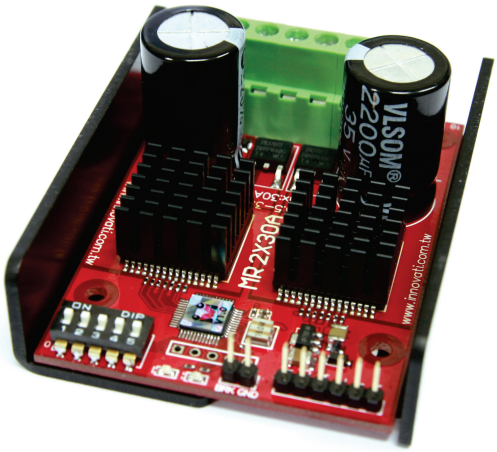
MR 2x30a

雙直流馬達控制模組

版本: V2.0



產品介紹: 利基 MR 2x30a 模組可以達到透過簡易的指令設定，自由操控兩顆直流馬達的需求。可以隨時動態的更改馬達轉速，並取得馬達現在的設定狀態包含轉速或是方向。與 MR 2x5a 相比，能承受更高的電壓與電流值。

應用方向:

- 控制馬達趨動，設定模型車的前進與後退。
- 需要轉速回傳的設備，動態調整轉速。
- 可以直接加上小風扇，並操作風量強度。

產品特色:

- 以簡單指令控制兩個馬達的轉向與轉速。
- 可承受最大±30A 的連續輸出電流。
- 輸入電壓最高可承受至 35V。
- 內部固定頻率 10KHz PWM 電流控制。
- 提供過熱自動斷電保護(150℃)。
- 提供過載電流保護。
- 提供 Crossover-Current Protection 與低壓閉鎖保護(UVLO)
- 透過 Brake 指令能快速停止馬達的動作。
- 可以設定 1025 階不同轉速。
- 兩組馬達能分別設定不同速度與方向。
- 透過指令能隨時取得現在馬達轉速或轉向等各種設定。
- 提供外接停止訊號，接上簡單的外部按鈕，就能由按鈕停止馬達轉動。
- 錯誤提醒事件，在錯誤狀態排除後，可以指令快速回復前一狀態
- 可透過 I2C 方式，下達指令。

連接方式: 直接將 ID 開關撥至欲設定的編號，再將 cmdBUS 連接至 Basic Commander 上對應的腳位，就可透過 BASIC Commander 執行操作。請將要操作的直流馬達請依據腳位連接至對應的 1A1B 與 2A2B 馬達輸入接腳，並將模組的馬達電源 VM 與 GND 腳位，連接至能提供符合馬達需求的電源。指令操作時，如果馬達轉向與指令相反，代表 1A 與 1B 連接反向，或是 2A 與 2B 連接反向，此時可以將 A 與 B 對調，或是將程式中的前進與後退指令顛倒。

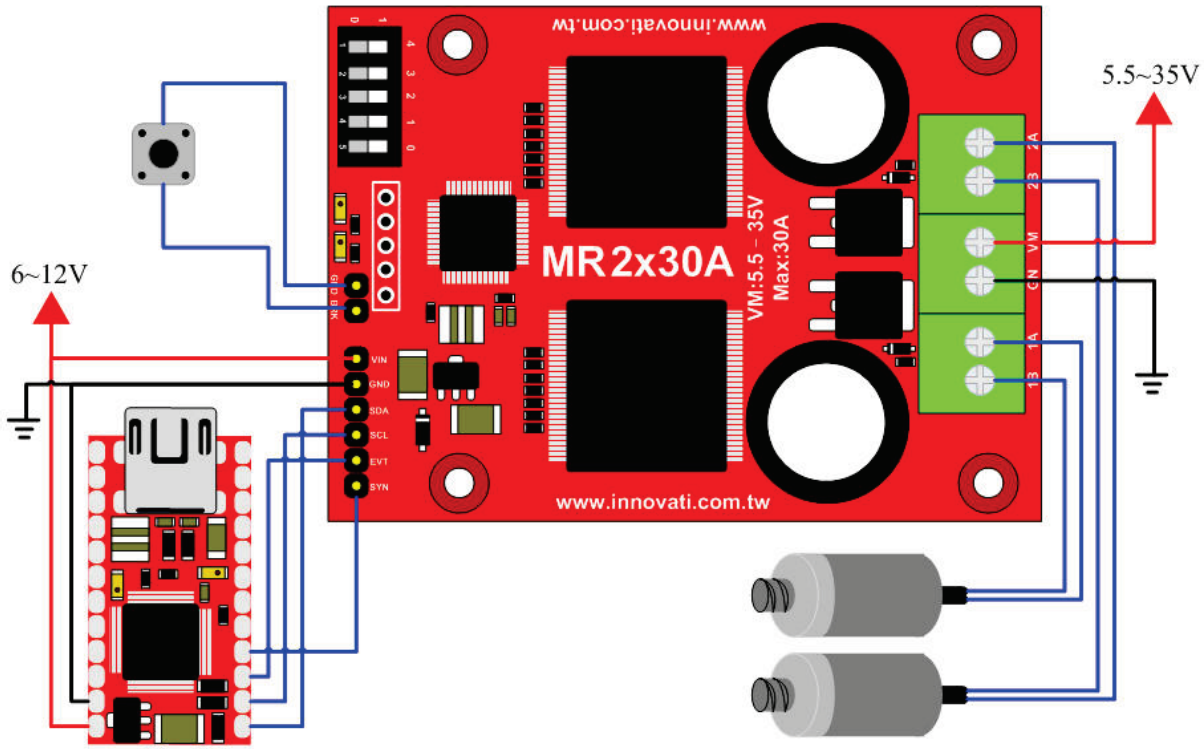


圖 1: 馬達模組連接範例

產品規格:

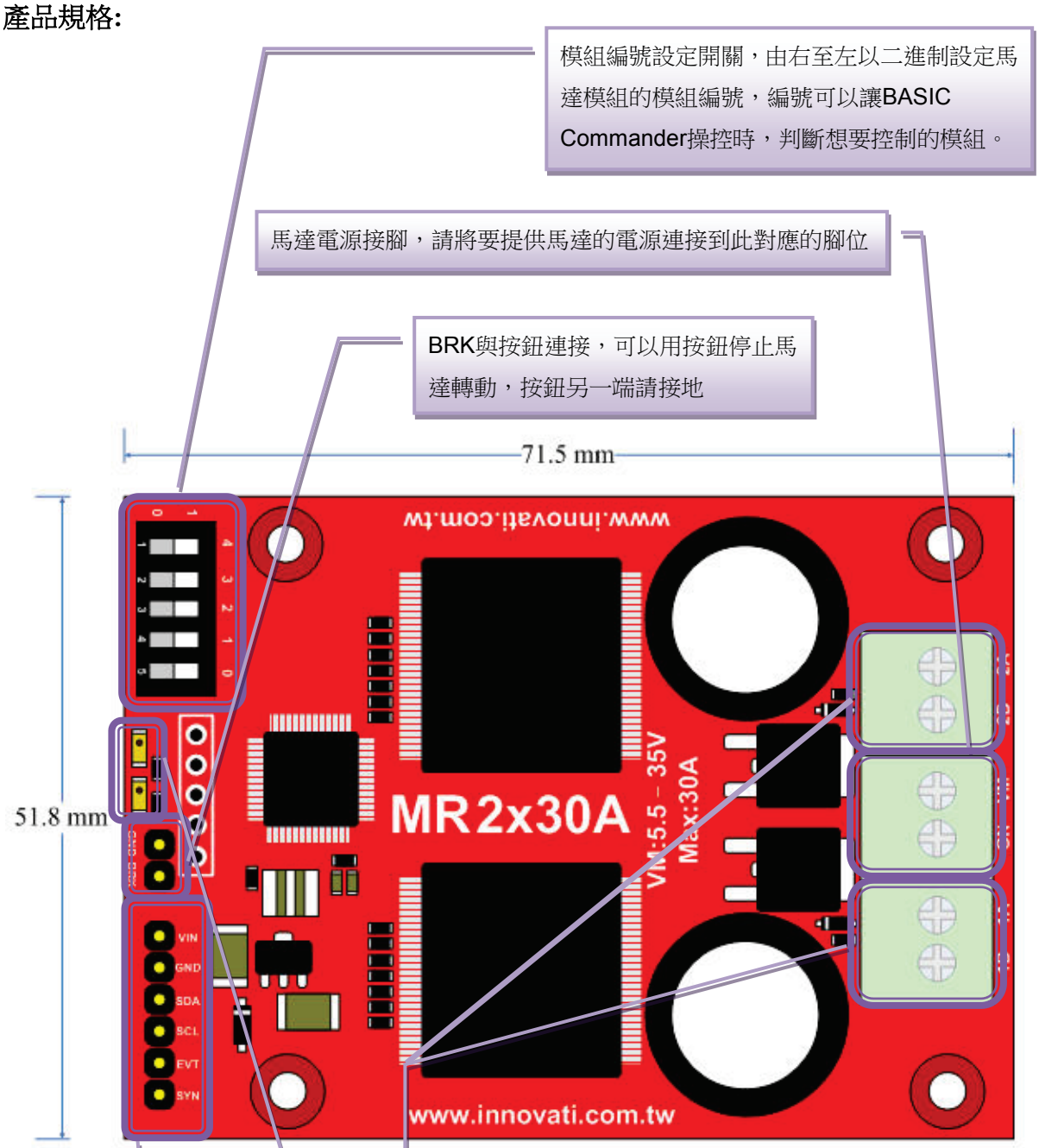


圖 2: 模組腳位與開關介紹

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	Operating Current	7.5	No I/O	—	9.3	—	mA
f_{pwm}	PWM Output frequency	—	—	0	—	10	kHz

表 1: 工作電流特性 (於 25℃之環境)

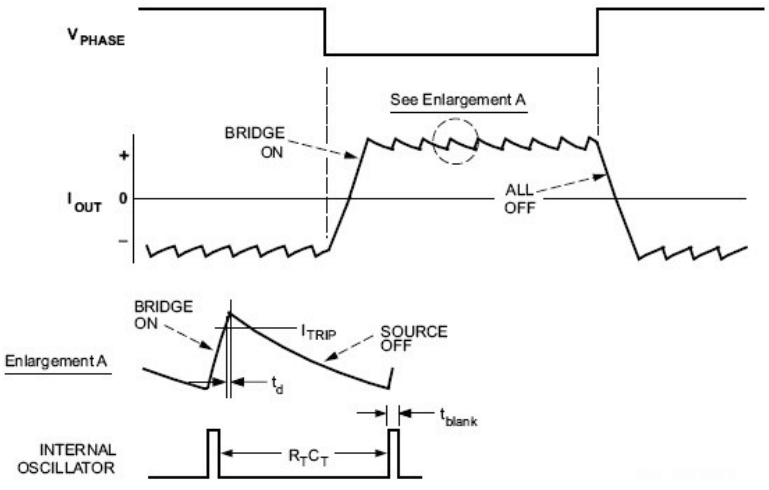
Test Condition: T_A=25℃，V_M=5V-35V

Characteristic	Symbol	Test Conditions	Limits			
			Min.	Typ.	Max.	Units
Load Supply Voltage Range	V_M	Operating	5.5	-	35	V
Thermal Shutdown Temp.	T_J	V_{IN} = 3.25V	150	170	200	℃
Thermal Shutdown Hysteresis.	△T_J		7	15	-	℃

表 2: 馬達相關電氣特性

過熱保護動作: 過熱保護電路在感測到驅動 IC 內部溫度到達 165℃時，將自動斷路，此時馬達即停止動作，當溫度下降 8℃後，保護電路自動回復導通，馬達就繼續先前的動作。

電流限流保護動作: 請參照右圖，在 H-bridge 開始輸出時，電流隨著馬達轉動增加，當電流值超過 I_{TRIP} (如右下圖 Enlargement A 中之指示)，就會停止 H-bridge 的輸出，直到內部震盪器下一個時脈傳送出(如右下圖 INTERNAL OSCILLATOR)，又會開始電流的傳送，如此反覆，電流會被固定在如圖的範圍內。



操作注意事項:

馬達模組提供一組馬達連接腳位，請確認所連接的馬達為直流馬達。

模組出廠時並未加上散熱片，在低電流熱對流良好環境，模組可以正常操作，但在大電流通過，或是於高熱無法靠一般對流散除時，建議將所附的散熱片加上。下表是在室溫(25℃)，熱對流良好環境下，以較大電流測試，部安裝散熱片的情況下，模組可以正常

運作的約略時間:

Current (A)	Time to overheat protection (Sec)
10	>300
12	~107
15	~40
18	~20

表 3: 電流與過熱保護啟動時間(未安裝散熱片)

模組操作溫度　　0 ℃~ 70 ℃ (馬達之操作溫度請另行確認)
模組儲存溫度　　-50 ℃~125℃

模組下達指令的方式可分為兩種：cmdBUS、I2C 控制方式
cmdBUS 指令表:

下面的指令表是專供控制 MR 2x30a 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 MR 2x30a 指令前，請先於程式開頭定義對應參數與編號，例:

Peripheral *ModuleName* As MR2x30a @ *ModuleID*

I2C 通訊協議(Protocol):

為了使更廣泛的使用者能控制模組，提供了部份指令的通訊協議讓使用者應用。透過通訊規格，使用者可使用 I2C 通訊協議為模組下達命令。
通訊協議常見的封包如下：

MID：模組 ID 編號，空間大小為 Byte 的變數。對應於硬體的指播開關。
CID：命令 ID 編號，空間大小為 Byte 的變數。依不同命令而改變。
CheckSum1：驗證位元_1，空間大小為 Byte 的變數。
　　定義方式：255 – (MID * 2) – CID
CheckSum2：驗證位元_2，空間大小為 Byte 的變數。
　　定義方式：255 – (CheckSum1~CheckSum2 之間的變數總和)
CheckSum3：驗證位元_3，空間大小為 Byte 的變數。
　　定義方式：255 – MID-(MID~CheckSum3 之間的變數總和)
Dummy：虛設位元，可為任意變數。空間大小為 Byte 的變數。

於通訊規格每筆資料空間大小階為 Byte，若資料空間大小超過一個 Byte 時，需將資料拆開，並由 Low Byte 開始傳送。
Ex:傳送資料 Temp 為一筆空間大小為 Word 的資料,則需將 Temp 拆開,分為 Temp_L、Temp_H，並且先傳送 Temp_L。

Ex1 模組編號為 2，命令編號為 153，傳送參數 Byte 為 100，通訊協議為 MID+CID+CheckSum1+Byte+CheckSum2+Dummy 則：

MID = 2

CID = 153

CheckSum1 = 255 – (2*2) – 153 = 98

Byte = 100

CheckSum2 = 255 – 100

Dummy = 0~255 之間的任意數

Ex2 模組編號為 2，命令編號為 153，傳送參數 Temp 為 511，通訊協議為 MID+CID+CheckSum1+Temp_L+Temp_H+CheckSum2+Dummy 則：

MID = 2

CID = 153

CheckSum1 = 255 – (2*2) – 153 = 98

Temp_L = 255，Temp_H = 1

CheckSum2 = 255 – Temp_L–Temp_H = 255

Dummy = 0~255 之間的任意數

指令格式	指令功能
馬達加速相關指令	
CmdBUS： **BackwardA(*DutyCycle*)** **I2C：** **MID+92+CheckSum1** **+DutyCycle_L+DutyCycle_H** **+Dummy** **BackwardAB(*DutyCycleA*, *DutyCycleB*)** **CmdBUS：** **BackwardB(*DutyCycle*)** **I2C：** **MID+93+CheckSum1** **+DutyCycle_L+DutyCycle_H** **+Dummy** **BackwardDual(*DutyCycleAll*)**	命令 A，B 或 A 與 B 馬達進行向後轉的動作，並且根據 ***DutyCycle*** 所給的值，決定馬達的轉速，請輸入 0~1024 之間的整數值(***DutyCycle*** 值越高，轉速越快)
CmdBUS： **ForwardA(*DutyCycle*)** **I2C：** **MID+88+CheckSum1** **+DutyCycle_L+DutyCycle_H** **+Dummy**	命令 A，B 或 A 與 B 馬達進行向前轉的動作，並且根據 ***DutyCycle*** 所給的值，決定馬達的轉速，請輸入 0~1024 之間的整數值(***DutyCycle*** 值越高，轉速越快)

ForwardAB(*DutyCycleA*, *DutyCycleB*) **CmdBUS：** **ForwardB(*DutyCycle*)** **I2C：** **MID+89+CheckSum1 +DutyCycle_L+DutyCycle_H +Dummy** **ForwardDual(*DutyCycleAll*)**	
馬達停止相關指令	
CmdBUS： **BrakeA ()** **I2C：** **MID+99+CheckSum1+Dummy**	快速停止 A，B 或 A 與 B 馬達模組的動作
CmdBUS： **BrakeB ()** **I2C：** **MID+100+CheckSum1+Dummy**	
CmdBUS： **BrakeDual()** **I2C：** **MID+101+CheckSum1+Dummy**	
CmdBUS： **StopA()** **I2C：** **MID+96+CheckSum1+Dummy**	停止 A，B 或 A 與 B 馬達模組的動作
CmdBUS： **StopB()** **I2C：** **MID+97+CheckSum1+Dummy**	
CmdBUS： **StopDual()** **I2C：** **MID+98+CheckSum1+Dummy**	

設定與狀態相關指令	
GetDCA(*DutyCycle*)	取得 A 或 B 的馬達轉速，存放於 ***DutyCycle*** 參數中 (***DutyCycle*** 值越高，轉速越快)，***DutyCycle*** 會回傳 0~1024 之間的整數值
GetDCB(*DutyCycle*)	
GetDCAB(*DutyCycleA*, *DutyCycleB*)	
GetDirA(*Dir*)	取得 A 或 B 的馬達轉向，存放於 ***Dir*** (***Dir*** 為 0 表示向前，為 1 表示向後)
GetDirB(*Dir*)	
GetDirAB(*DirA*, *DirB*)	
SetDCA(*DutyCycle*)	以 ***DutyCycle*** 設定 A，B 或 A 與 B 馬達模組的轉速，請輸入 0~1024 之間的整數值 (DutyCycle 值越高，轉速越快)
SetDCAB(*DutyCycleA*, *DutyCycleB*)	
SetDCB(*DutyCycle*)	
SetDCDual(*DutyCycleAll*)	
SetDirA(*Dir*)	以 ***Dir*** 設定 A，B 或 A 與 B 馬達模組的轉向 (***Dir*** 為 0 表示向前，為 1 表示向後)
CmdBUS： **SetDirAB(*DirA*, *DirB*)** **I2C：** **MID+104+CheckSum1** **+DirA+DirB** **+CheckSum2+Dummy**	
SetDirB(*Dir*)	
SetDirDual(*DirAll*)	
GetVelA(*DutyCycle*)	取得 A 或 B 的馬達轉速，並以正負表示方向，存放於 ***DutyCycle*** 參數中，***DutyCycle*** 絕對值越高，轉速越快，正值代表向前，負值代表向後，***DutyCycle*** 會回傳-1024~1024 之間的整數值
GetVelB(*DutyCycle*)	
GetVelAB(*DutyCycleA*, *DutyCycleB*)	
SetVelA(*DutyCycleA*)	以 ***DutyCycle*** 設定 A，B 或 A 與 B 馬達模組的轉速，請輸入-1024~1024 之間的整數值，***DutyCycle*** 絕對值越高，轉速越快，輸入正值代表向前轉動，負值代表向後轉動
SetVelB(*DutyCycleB*)	
CmdBUS： **SetVelAB(*DutyCycleA*, *DutyCycleB*)** **I2C：** **MID+118+CheckSum1** **+DutyCycleA_L+DutyCycleA_H** **+DutyCycleB_L+DutyCycleB_H** **+CheckSum2+Dummy**	
SetVelDual(*DutyCycle*)	
停止按鈕相關指令 ＊1	
CmdBUS： **ClrBrakeButStatus()** **I2C：** **MID+124+CheckSum1+Dummy**	將停止按鈕的狀態值清除為 0，在按下停止按鈕後，如果沒有執行清除，就無法重新啟動馬達動作

CmdBUS： ***bStatus* = GetBrakeButStatus()** **I2C：** **Out：** **MID+123+CheckSum1+Dummy** **In：** **MID+bStatus+CheckSum2+Dummy**	回傳停止按鈕現在的狀態 ***bStatus*** 0: 沒有按下停止按鈕 1: 停止按鈕被按下，還未執行清除的動作
錯誤狀態檢測相關指令	
CmdBUS： **EnFaultStop()** **I2C：** **MID+125+CheckSum1+Dummy**	啟動錯誤停止機制，當錯誤狀態產生，就會自動停止馬達轉動，並且在執行清除錯誤狀態指令前，無法執行啟動馬達等相關指令，預設為不啟動
CmdBUS： **DisFaultStop()** **I2C：** **MID+126+CheckSum1+Dummy**	關閉錯誤停止機制，當錯誤狀態產生，系統會自動進行錯誤狀態排除，並繼續馬達的轉動，但若有啟動錯誤事件提醒，仍會回傳錯誤提醒事件，預設為關閉錯誤停止機制
CmdBUS： ***bStatus* = GetFaultStatus()** **I2C：** **Out：** **MID+127+CheckSum1+Dummy** **In：** **MID+bStatus+CheckSum2+Dummy**	取得錯誤狀態儲存於 ***bStatus*** 中，在關閉錯誤停止機制下，錯誤狀態會被自動清除為 0，並繼續原先設定的馬達轉動；但若是啟動錯誤停止機制下，一旦有錯誤產生，若是馬達 1 的錯誤，則會回傳 1，馬達 2 的錯誤會回傳 2，兩個馬達都發生錯誤狀態則回傳 3。回傳的錯誤狀態在沒有執行清除動作前，都會維持住，此時無法執行轉動馬達的動作 **＊2**
CmdBUS： **ClrFaultStatus()** **I2C：** **MID+128+CheckSum1+Dummy**	清除錯誤停止機制的停止狀態，在啟動錯誤停止機制下，一旦發生錯誤狀態，就必須執行此指令，才能重新啟動馬達的轉動
EnFaultEvent()	啟動錯誤狀態提醒事件，預設為啟動
DisFaultEvent()	關閉錯誤狀態提醒事件，預設為關閉
CmdBUS： **RestoreStatus()** **I2C：** **MID+131+CheckSum1+Dummy**	回復錯誤狀態發生時的各個設定值

＊1 停止按鈕相關指令，需要在外部加裝停止按鈕後，才能產生效果

＊2 在 DisFaultStop()狀態下，收到 FaultEvent 後，系統就會自動進行 ClearFault()的動作，執行 GetFaultStatus()仍會得到 0

模組提供應用事件:

事件名稱 (Event)	啟動條件
FaultEvent	當偵測到過熱等狀態造成馬達啟動防護機制，並且在啟動錯誤事件提醒狀態，就會產生此事件

範例程式:

```
Peripheral myMotor As MR2x30a @ 0        '   設定模組編號為 0

Sub Main()
    Debug CLS
    MyMotor.ForwardDual(800)             '   讓馬達以 800 的速度向前轉動
    Pause 3000
    MyMotor.StopDual()                   '   停止馬達轉動
    Pause 3000
    MyMotor.BackwardDual(800)            '   讓馬達以 800 的速度向後轉動
    Pause 3000
    MyMotor.SetDirDual(0)                '   設定馬達轉向改為向前轉動
    Pause 3000
    MyMotor.SetDCDual(600)               '   將馬達轉速改為 600
    Pause 3000
    MyMotor.BrakeDual()                  '   快速停止馬達
End Sub
```

附錄

1. 已知問題:

2. 模組編號開關對應編號表:

	0		8		16		24
	1		9		17		25
	2		10		18		26
	3		11		19		27
	4		12		20		28
	5		13		21		29
	6		14		22		30
	7		15		23		31