Joystick 3A

三軸搖桿與搖桿按鍵模組

版本: V2.0

產品介紹: 利基 Joystick 3A 模組提供簡易的設定與位置取得指令，讓使用者可以規劃符合自己需求的搖桿。透過 cmdBUS 與利基的 BASIC Commander 連接，就可以執行各種專屬的應用指令。不論是機器手臂，機器人操縱，都可以直覺的達成。提供直角坐標與極座標兩種定址方式，使用者可以根據需求，任意更換所要的回傳座標系。除了平面的控制，也有旋轉軸向與按鍵，可以控制更複雜的應用。

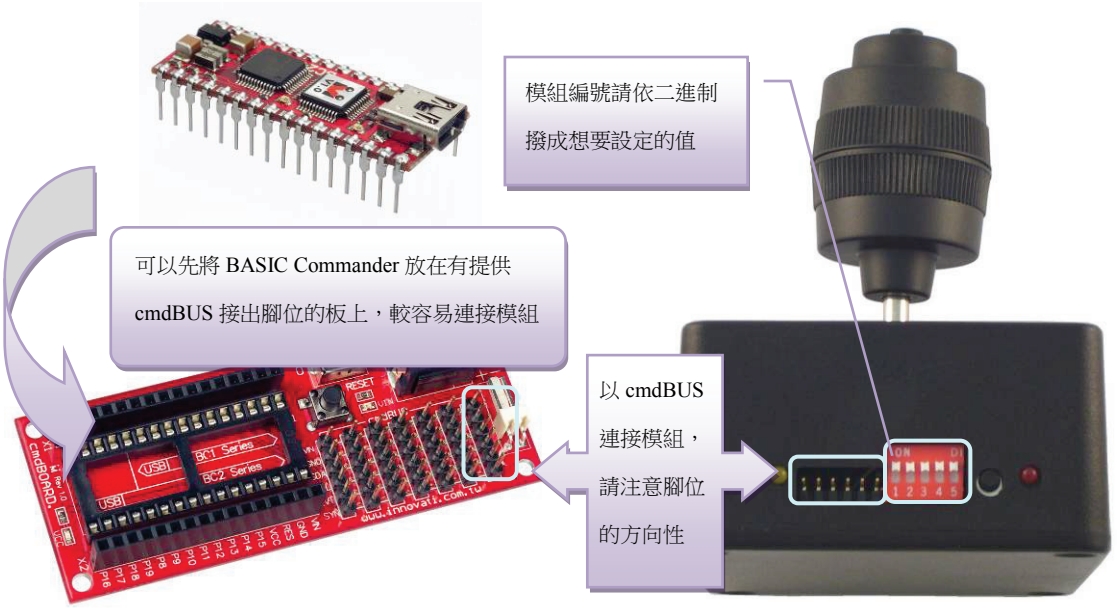
應用方向:

- 連結機器手臂，以極座標直接設定手臂所要旋轉的角度。
- 加裝配合無線輸出，控制各種遙控車，飛機等應用。
- 各種測試機具的操作。
- 搭配馬達模組做馬達加速控制，可以搭配按鍵做定速功能。
- 控制利基應用科技的各項應用套件。

產品特色:

- 設定容易，只要使用 cmdBUS 連接 BASIC Commander，就可以用專屬的指令做各種應用。
- 三軸操作，可以移動搖桿在平面做兩軸的操作，也可以轉動搖桿進行第三軸的控制。
- 提供兩種回傳座標值:直角坐標與極座標，可以隨時選擇所要的回傳方式，或是混合使用。
- 可以回傳四向與八向搖桿位置，快速直覺應用到各種基本控制。
- 原點範圍可以自由設定 0~10%的變動值，避免跳動。
- 各種座標值的回傳，都可以設定極限值範圍，讓使用者可以限定搖桿所要的操作範圍。
- 回傳刻度可以設定 128 個刻度值，極座標角度範圍值可以設定 360 個刻度值，可以分開設定所需要的精準度。
- 提供搖桿按鍵，在搖桿上方有自定按鍵，可以自行設定按鍵功能，包括按鍵的連續觸發起動時間，以及連續觸發的速率，都可以透過指令設定。
- 提供校正功能，並有校正按鈕，操作中可以隨時中斷，進行搖桿的校正。
- 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。



產品規格:

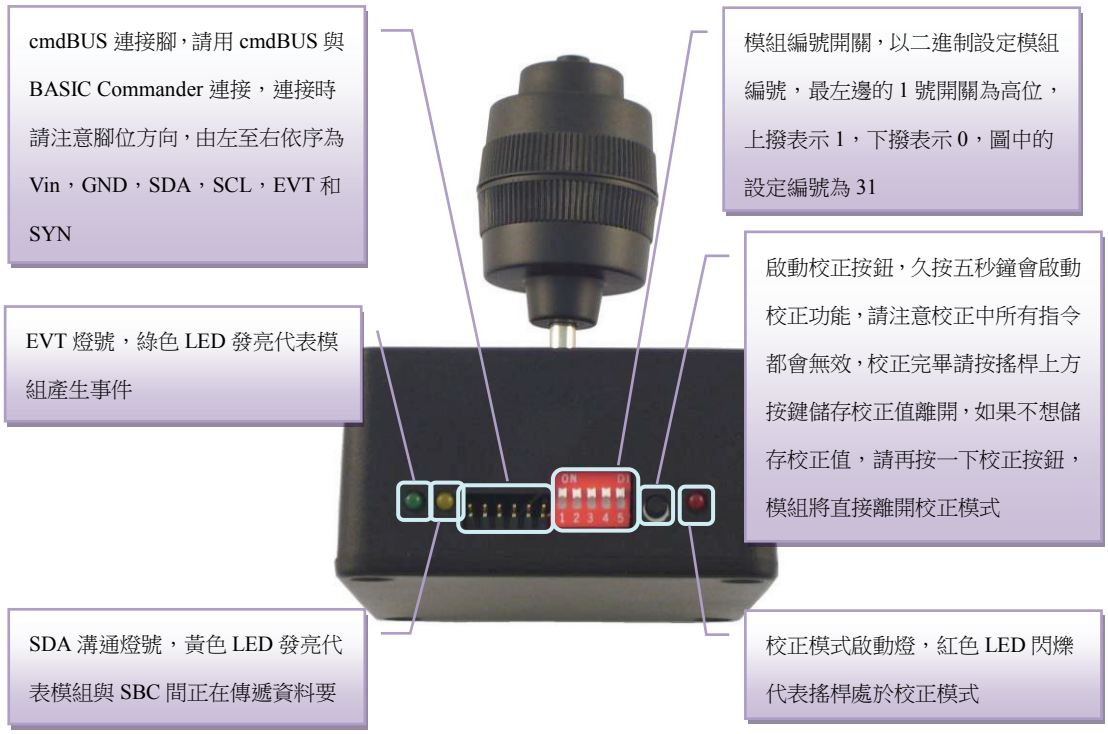


圖 1: 模組腳位與開關介紹

進入校正模式後(紅色 LED 燈閃爍時)，請將搖桿推到頂點，再沿著頂點繞兩圈，以取得 XY 軸向的最大與最小值，接著旋轉 Z 軸，請分別旋轉到左極限與右極限，並在極限處停留兩秒，讓搖桿記錄 Z 軸最大與最小值，最後將搖桿靜置於中心點，等候三秒，讓搖桿記錄完 XYZ 軸的中心點值，最後再按下搖桿上方的按鈕，結束校正模式。
若是不小心啟動校正模式，可以再按下校正鈕，就可以離開校正模式。

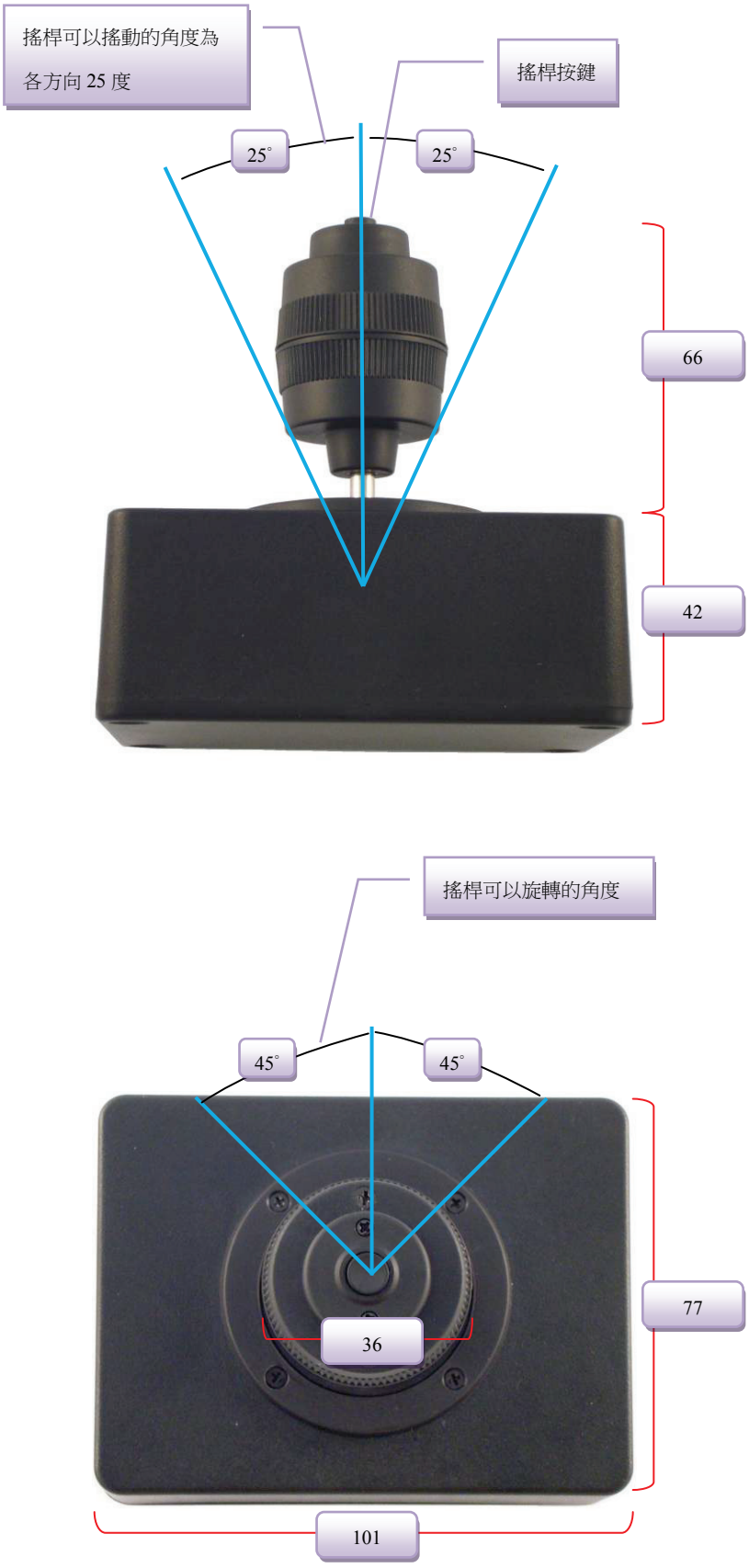


圖 2: 搖桿規格 (單位 mm)

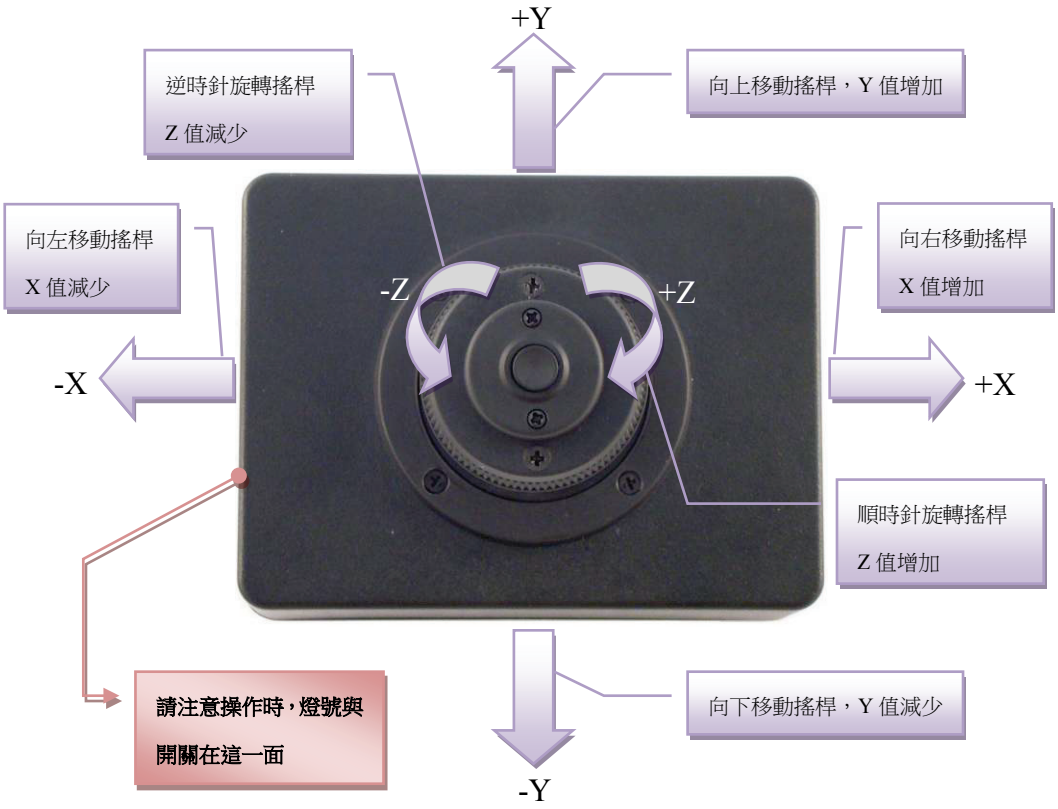


圖 3: 直角座標系操作軸向

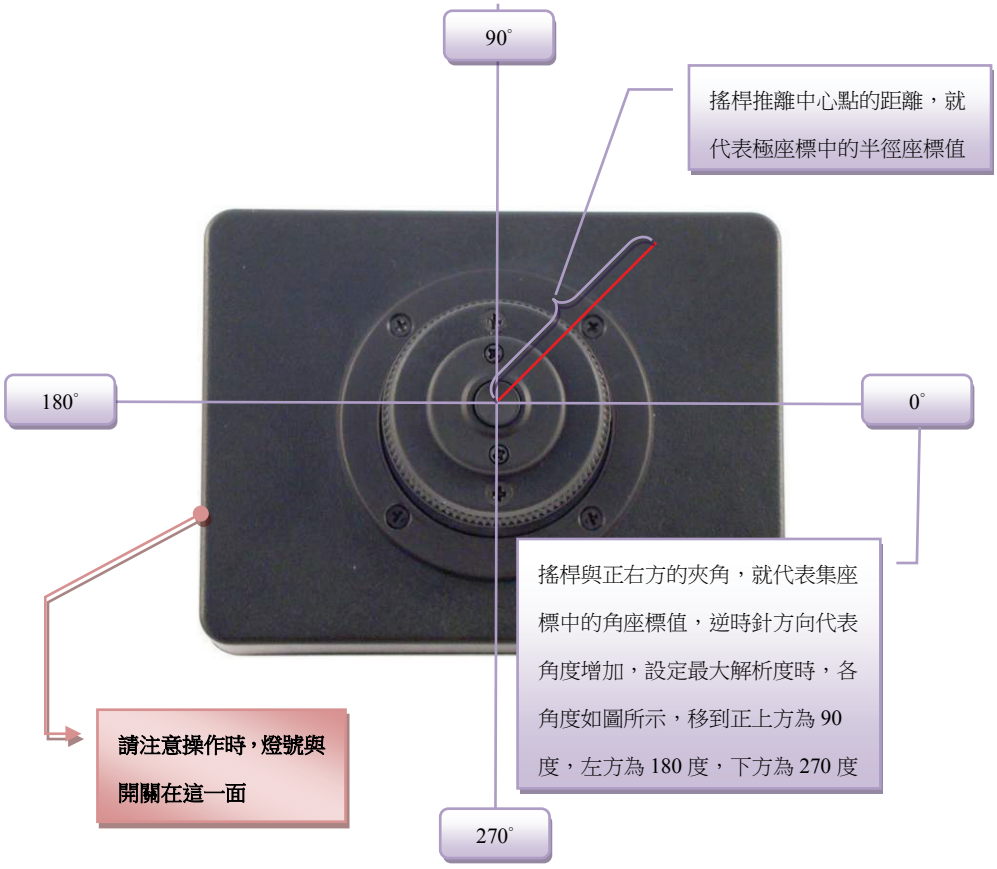


圖 4: 極座標系操作軸向

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{IN}	Conditions				
I_{IN}	Operating Current	7.5	—	—	10.5	—	mA

表 1: 工作電流特性 (於 25 ℃之環境)

操作注意事項:

轉軸可承受約 500,000 次的旋轉動作。

操作溫度　　-10 ℃~ 80 ℃

儲存溫度　　-10 ℃~ 80 ℃

模組下達指令的方式可分為兩種：cmdBUS、I2C 控制方式

cmdBUS 指令表:

請在程式一開始先以”Peripheral”指令，定義程式中搖桿模組的名稱，請設定對應到的模組名稱為”Joystick3A”，完整的設定請參考範利程式的第一行。

(指令格式中，各參數以”,”分隔開，並以”參數型態 參數”方式表列)

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
校正搖桿相關指令	
GetCalibrationX(*Xmin*,*Xcen Xmax*)	取得 X 軸方向的搖桿校正值，三個回傳值是電壓值轉換為數位的結果，***Xmin*** 為設定的最小 X 軸值，***Xcen*** 代表中心點值，***Xmax*** 則為取得的最大值。***Xmin***，***Xcen***，與 ***Xmax*** 回傳值為 0~65535 之間的整數值。
GetCalibrationY(*Ymin*,*Ycen*, *Ymax*)	取得 Y 軸方向的搖桿校正值，三個回傳值是電壓值轉換為數位的結果，***Ymin*** 為設定的最小 Y 軸值，***Ycen*** 代表中心點值，***Ymax*** 則為取得的最大值。***Ymin***，***Ycen***，與 ***Ymax*** 回傳值為 0~65535 之間的整數值。
GetCalibrationZ(*Zmin*,*Zcen*, *Zmax*)	取得 Z 軸方向的搖桿校正值，三個回傳值是電壓值轉換為數位的結果，***Zmin*** 為設定的最小 Z 軸值，***Zcen*** 代表中心點值，***Zmax*** 則為取得的最大值。***Zmin***，***Zcen***，與 ***Zmax*** 回傳值為 0~65535 之間的整數值。

SetCalibrationX(*Xmin*,*Xcen*, *Xmax*)	設定 X 軸向的搖桿校正值，需要輸入三個 word 參數，分別是 ***Xmin*** 設定最小的搖桿校正值，***Xcen*** 設定搖桿中心點的校正值，***Xmax*** 設定最大的搖桿校正值，手動設定時請注意設定順序，***Xmin*** 請輸入最小值，而 ***Xmax*** 請輸入最大值。***Xmin***，***Xcen***，與 ***Xmax*** 請輸入 0~65535 之間的整數值。
SetCalibrationY(*Ymin*, *Ycen*, *Ymax*)	設定 Y 軸向的搖桿校正值，需要輸入三個 word 參數，分別是 ***Ymin*** 設定最小的搖桿校正值，***Ycen*** 設定搖桿中心點的校正值，***Ymax*** 設定最大的搖桿校正值，手動設定時請注意設定順序，***Ymin*** 請輸入最小值，而 ***Ymax*** 請輸入最大值。***Ymin***，***Ycen***，與 ***Ymax*** 請輸入 0~65535 之間的整數值。
SetCalibrationZ(*Zmin*, *Zcen*, *Zmax*)	設定 Z 軸向的搖桿校正值，需要輸入三個 word 參數，分別是 ***Zmin*** 設定最小的搖桿校正值，***Zcen*** 設定搖桿中心點的校正值，***Zmax*** 設定最大的搖桿校正值，手動設定時請注意設定順序，***Zmin*** 請輸入最小值，而 ***Zmax*** 請輸入最大值。***Zmin***，***Zcen***，與 ***Zmax*** 請輸入 0~65535 之間的整數值。
CmdBUS： **StartCalibration()** **I2C：** **MID+88+CheckSum1+Dummy**	啟動搖桿校正模式。執行此命令後，搖桿會進入校正模式，此時請將搖桿推到頂點，再沿著頂點繞兩圈，以取得 XY 軸向的最大與最小值，接著旋轉 Z 軸，請分別旋轉到左極限與右極限，並在極限處停留兩秒，讓搖桿記錄 Z 軸最大與最小值，最後將搖桿靜置於中心點，等候三秒，讓搖桿記錄完 XYZ 軸的中心點值，最後再按下搖桿上方的按鈕，結束校正模式。
取得搖桿座標相關指令	
CmdBUS： **GetXY(*X*, *Y*)** **I2C：** **Out：** **MID+123+CheckSum1+Dummy** **In：** **MID+X+Y+CheckSum3**	以直角坐標系，取得搖桿現在的 XY 軸座標，***X*** 為 X 軸向的座標刻度值，***Y*** 為 Y 軸向的座標刻度值，預設範圍都是-127~127，可以使用 **SetXYRes** 指令更改刻度範圍。***X***，***Y*** 回傳值為-127~127 之間的整數值。

CmdBUS： **GetZ(*Z*)** **I2C：** **Out：** **MID+124+CheckSum1+Dummy** **In：** **MID+Z +CheckSum3**	取得搖桿現在的 Z 軸座標，***Z*** 為 Z 軸向的座標刻度值，刻度值預設範圍是 -127~127，可以使用 **SetKnobRes()** 指令更改刻度範圍。***Z*** 回傳值為 -127~127 之間的整數值。
GetPolarBinaryRadian(*Radius*, *Angle*)	取得整數值表示的搖桿極座標值，***Radius*** 是半徑值，回傳值為 0~127 之間的整數值，***Angle*** 是角度值，回傳值為 0~359 之間的整數值，以 X 軸為零度，逆時針方向增加角度值。半徑刻度預設範圍是 0~127，可以使用 **SetRadiusRes** 指令更改刻度範圍。角度刻度預設範圍是 0~359，可以使用 **SetRadianRes** 更改刻度範圍。
CmdBUS： **GetPolarRadian(*Radius*, *Radian*)** **I2C：** **Out：** **MID+125+CheckSum1+Dummy** **In：** **MID+Radius+Radian_L+Radian_R +CheckSum3**	取得浮點數表示的搖桿極座標值。第一個回傳值 ***Radius***，是以浮點數表示的半徑值，範圍是 0~1。第二個回傳值 ***Radian***，是以浮點數表示的角度值，以弧度為單位，範圍是 0~6.37。
CmdBUS： ***Dir* = Get4WayStatus()** **I2C：** **Out：** **MID+127+CheckSum1+Dummy** **In：** **MID+Dir+CheckSum3**	取得以四向表示的搖桿位置。***Dir*** 是方向值，只會有 0~4 的回傳值，分別代表: 0 ➔ 搖桿位於中心點　　1 ➔ 搖桿位置於右方 2 ➔ 搖桿位置於下方　　3 ➔ 搖桿位置於左方 4 ➔ 搖桿位置於上方
CmdBUS： ***Dir* = Get8WayStatus()** **I2C：** **Out：** **MID+128+CheckSum1+Dummy** **In：** **MID+Dir+CheckSum3**	取得以八向表示的搖桿位置。***Dir*** 是方向值，只會有 0~8 的回傳值，分別代表: 0 ➔ 搖桿位於中心點　　1 ➔ 搖桿位置於右方 2 ➔ 搖桿位置於右下方　　3 ➔ 搖桿位置於下方 4 ➔ 搖桿位置於左下方　　5 ➔ 搖桿位置於左方 6 ➔ 搖桿位置於左上方　　7 ➔ 搖桿位置於上方 8 ➔ 搖桿位置於右上方

取得搖桿範圍設定相關指令	
GetStickDeadZone(*DZx*, *DZy*)	取得 XY 軸設定的中心點範圍值。***DZx*** 和 ***DZy*** 會回傳 0~10 之間的整數值，單位為%，代表目前以直角座標系取得座標時，判定搖桿在中心點的範圍值。
GetKnobDeadZone(*DZz*)	取得 Z 軸設定的中心點範圍值。***DZz*** 會回傳 0~10 之間的整數值，單位為%，代表目前取得 Z 軸座標時，判定搖桿在中心點的範圍值。
GetRadiusDeadZone(*DZr*)	取得半徑設定的中心點範圍值。***DZr*** 會回傳 0~10 之間的整數值，單位為%，代表目前以極座標取得半徑值時，判定搖桿在中心點的範圍值。
GetXYSaturation(*SATx*, *SATy*)	取得 XY 軸設定的極限範圍值。***SATx*** 和 ***SATy*** 會回傳 60~100 之間的整數值，單位為%。代表目前以直角座標系取得座標時，不論正負向，只要到達設定值，就會回傳最大或最小值，設定的刻度值，只會在最大與最小值範圍內，再做分割計算。
GetKnobSaturation(*SATz*)	取得 Z 軸設定的極限範圍值。***SATz*** 會回傳 60~100 之間的整數值，單位為%。代表目前取得 Z 軸座標時，不論正負向，只要到達設定值，就會回傳最大或最小值，設定的刻度值，只會在最大與最小值範圍內，再做分割計算。
GetRadiusSaturation(*SATr*)	取得半徑設定的極限範圍值。***SATr*** 會回傳 60~100 之間的整數值，單位為%。代表目前以極座標系取得座標時，只要到達設定值，就會回傳最大值，設定的刻度值，只會在最大值範圍內，再做分割計算。
GetXYRes(*RESx*, *RESy*)	取得 XY 軸解析度設定值。***RESx*** 和 ***RESy*** 會回傳 0~128 之間的整數值。代表目前以直角座標系取得座標時，在可辨識的範圍內，所要切割的刻度數。由於計數時包含 0，所以設定 128 就代表正向切分為 0~127 共 128 個刻度，反向也是由 0~-127 共 128 個刻度值。
GetKnobRes(*RESz*)	取得 Z 軸解析度設定值。***RESz*** 會回傳 0~128 之間的整數值。代表目前取得 Z 軸座標時，在可辨識的範圍內，所要切割的刻度數。由於計數時包含 0，所以設定 128 就代表正向切分為 0~127 共 128 個刻度，反向也是由 0~-127 共 128 個刻度值。

GetRadiusRes(*RESr*)	取得半徑解析度設定值。***RESr*** 會回傳 0~128 之間的整數值。代表目前以極座標系取得座標時，在可辨識的範圍內，半徑值所要切割的刻度數。由於計數時包含 0，所以設定 128 就代表切分為 0~127 共 128 個刻度值。
GetRadianRes(*RESa*)	取得角度解析度設定值。***RESa*** 會回傳 0~360 之間的整數值。代表目前以極座標系取得座標時，角度所要切割的刻度數。由於計數時包含 0，所以設定 360 就代表切分為 0~359 共 360 個刻度值。
自訂搖桿範圍設定相關指令	
RestoreSettings()	回復所有搖桿設定到出廠值。
SetStickDeadZone(*DZx*, *DZy*)	設定搖桿在取得直角座標系時，XY 軸中心區域的範圍，**DZx** 與 ***DZy*** 依序為輸入 X 軸與 Y 軸，以百分比為單位，請輸入 0~10 之間的整數值，當搖桿移動沒有超出所設定的區域，都會判定搖桿在 XY 軸向的中心點。
SetKnobDeadZone(*DZz*)	設定搖桿 Z 軸中心區域的範圍，***DZz***，以百分比為單位，請輸入 0~10 之間的整數值，當搖桿轉動沒有超出所設定的區域，都會判定搖桿在 Z 軸的中心點。
SetRadiusDeadZone(*DZr*)	設定搖桿在取得極座標系時，半徑的中心區域範圍，***DZr***，以百分比為單位，請輸入 0~10 之間的整數值，當搖桿移動沒有超出所設定的區域，都會判定搖桿在極座標的中心點。
SetXYSaturation(*SATx*, *SATy*)	設定直角座標系，XY 軸的極限範圍值。***SATx*** 和 ***SATy*** 請輸入 60~100 之間的整數值，單位為%。執行指令後，以直角座標系取得座標時，不論正負向，只要到達設定值，就會回傳最大或最小值，設定的刻度值，只會在最大與最小值範圍內，再做分割計算。
SetKnobSaturation(*SATz*)	設定 Z 軸的極限範圍值。***SATz*** 請輸入 60~100 之間的整數值，單位為%。執行指令後，取得 Z 軸座標時，不論正負向，只要到達設定值，就會回傳最大或最小值，設定的刻度值，只會在最大與最小值範圍內，再做分割計算。

SetRadiusSaturation(*SATr*)	設定極座標系，半徑的極限範圍值。***SATr*** 請輸入 60~100 之間的整數值，單位為%。執行指令後，以極座標系取得座標時，只要到達設定值，就會回傳最大值，設定的刻度值，只會在最大值範圍內，再做分割計算。
SetXYRes(*RESx*, *RESy*)	設定 XY 軸解析度。***RESx*** 和 ***RESy*** 請輸入 0~128 之間的整數值。代表目前以直角座標系取得座標時，在可辨識的範圍內，所要切割的刻度數。由於計數時包含 0，所以設定 128 就代表正向切分為 0~127 共 128 個刻度，反向也是由 0~-127 共 128 個刻度值。請注意雖然可以輸入 0 和 1 兩個值，但設定後，取得的 XY 值都會是 0。
SetKnobRes(*RESz*)	設定 Z 軸解析度。***RESz*** 請輸入 0~128 之間的整數值。代表目前取得 Z 軸座標時，在可辨識的範圍內，所要切割的刻度數。由於計數時包含 0，所以設定 128 就代表正向切分為 0~127 共 128 個刻度，反向也是由 0~-127 共 128 個刻度值。請注意雖然可以輸入 0 和 1 兩個值，但設定後，取得的 Z 值都會是 0。
SetRadiusRes(*RESr*)	設定半徑解析度。***RESr*** 請輸入 0~128 之間的整數值。代表目前以極座標系取得座標時，在可辨識的範圍內，半徑值所要切割的刻度數。由於計數時包含 0，所以設定 128 就代表切分為 0~127 共 128 個刻度值。請注意雖然可以輸入 0 和 1 兩個值，但設定後，取得的半徑值都會是 0。
SetRadianRes(*RESa*)	設定角度解析度。***RESa*** 請輸入 0~360 之間的整數值。代表目前以極座標系取得座標時，角度所要切割的刻度數。由於計數時包含 0，所以設定 360 就代表切分為 0~359 共 360 個刻度值。請注意雖然可以輸入 0 和 1 兩個值，但設定後，取得的角度值都會是 0。
按鈕應用相關指令	
CmdBUS： ***Sta* = GetButtonStatus()** **I2C：** **Out：** **MID+130+CheckSum1+Dummy** **In：** **MID+Sta+CheckSum3**	取得按鍵的狀態，放在 ***Sta*** 參數中，回傳值如下: 關閉連續按鍵功能時 0: 按鍵被壓下 1: 沒有按按鍵 開啟連續按鍵功能時 0: 沒有偵測到新的按鍵事件 1: 按鍵剛被按下時，或是按鍵久按到達 Repeat Time

	所設定的時間，以及久按超過 Repeat Time 設定時間後，每隔 Repeat Rate 所設定的時間，就會再次設定為 1，在沒有執行 **GetButtonStatus** 指令前，都會保持 1 的值。
ButtonRepeatFunc(*Rep*)	啟動與關閉搖桿的自動連續按鍵功能，***Rep*** 請輸入 0 或 1， 0: 關閉自動連續按鍵功能，搖桿按鍵只有在按下時會產生一次 ButtonEvent，並且在按住搖桿按鍵時，回傳的搖桿按鍵狀態都是按下的狀態。 1: 啟動自動連續按鍵功能，在按住搖桿按鍵時，會根據所設定的 Repeat Time 和 Repeat Rate，不斷產生 ButtonEvent 直到放開搖桿按鍵為止。
SetRepeatTime(*Time*)	設定自動重複按鍵啟動時間。***Time*** 請輸入 0~255 之間的整數值，以 10 ms 為單位，設定後，當按住按鍵超過設定時間後，就會再回傳 **ButtonEvent**，並且開始依據 Repeat Rate 的設定，重複回傳 **ButtonEvent**。請注意重複按鍵模式，必須先執行 **ButtonRepeatFunc** 指令，啟動重複按鍵功能。另外當 Repeat Time 設定為 0 時，並不會關閉重複按鍵功能，而是直接依據 Repeat Rate 的設定，直接開始回傳 **ButtonEvent**。
GetRepeatTime(*Time*)	取得自動重複按鍵啟動時間的設定值。***Time*** 會回傳 0~255 之間的整數值。
SetRepeatRate(*Rate*)	設定自動重複按鍵間隔時間。***Rate*** 請輸入 0~255 之間的整數值，以 10 ms 為單位，設定後，當按住按鍵超過 Repeat Time 設定時間後，就會依據 ***Rate*** 的設定，每隔該時間重複回傳 **ButtonEvent**。請注意重複按鍵模式，必須先執行 **ButtonRepeatFunc** 指令，啟動重複按鍵功能。如果將 ***Rate*** 設定為 0，則在過了 Repeat Time 設定，就不會再啟動 **ButtonEvent**。
GetRepeatRate(*Rate*)	取得自動重複按鍵間隔時間的設定值。***Rate*** 會回傳 0~255 之間的整數值。
啟動與關閉事件相關指令	
EnableStickEvent()	啟動 **StickEvent** 事件
DisableStickEvent()	關閉 **StickEvent** 事件
EnableKnobEvent()	啟動 **KnobEvent** 事件
DisableKnobEvent()	關閉 **KnobEvent** 事件
Enable4WayEvent()	啟動 **Change4WayEvent** 事件

Disable4WayEvent()	關閉 **Change4WayEvent** 事件
Enable8WayEvent()	啟動 **Change8WayEvent** 事件
Disable8WayEvent()	關閉 **Change8WayEvent** 事件
EnableButtonEvent()	啟動 **ButtonEvent** 事件
DisableButtonEvent()	關閉 **ButtonEvent** 事件
EnableCalEndEvent()	啟動 **CalEndEvent** 事件
DisableCalEndEvent()	關閉 **CalEndEvent** 事件
SetEventRefreshRate(*Rate*)	設定 Event 產生的速率，***Rate*** 請輸入 0~255 之間的整數值，單位為 10 ms，設定後兩個 Event 產生的時間，就會間隔設定的時間。請注意設定為 0 等同設定為 1。
GetEventRefreshRate(*Rate*)	取得 Event 產生速率的設定，***Rate*** 會回傳 0~255 之間的整數值。

模組提供應用事件:

事件名稱 (Event)	啟動條件
StickEvent	當偵測到搖桿有移動時，就會產生此事件。
KnobEvent	當偵測到搖桿有轉動時，就會產生此事件。
Change4WayEvent	每當搖桿的四向值變更時，就會產生此事件。
Change8WayEvent	每當搖桿的八向值變更時，就會產生此事件。
ButtonEvent	當搖桿的連續按鍵功能關閉時: 按下搖桿按鍵就會產生。 當搖桿的連續按鍵功能開啟時: 按下搖桿按鍵，到達 Repeat Time 設定時間，以及每隔 Repeat Rate 設定時間就會產生。
CalEndEvent	當 Calibration 動作結束時。

範例程式:

(偵測搖桿移動與轉動，並在偵測到時，回傳 XYZ 值)

```
Peripheral myJoy As JoyStick3A @ 0 ‘ 設定模組編號

Sub Main()
    myJoy.SetStickDeadZone(2, 2)        ‘ 設定 XY 軸中心範圍
    myJoy.SetKnobDeadZone(2)            ‘ 設定 Z 軸中心範圍
    myJoy.SetXYSaturation(80, 80)       ‘ 設定 XY 軸極限值範圍
    myJoy.SetKnobSaturation(80)         ‘ 設定 Z 軸極限值範圍
    myJoy.SetXYRes(128, 128)            ‘ 設定 XY 軸刻度值
    myJoy.SetKnobRes(128)               ‘ 設定 Z 軸刻度值

    myJoy.SetEventRefreshRate(1)        ‘ 設定事件間隔時間
    myJoy.EnableStickEvent()            ‘ 啟動搖桿移動事件
    myJoy.EnableKnobEvent()             ‘ 啟動搖桿轉動事件

    Do
    Loop
End Sub

Event myJoy.StickEvent()
    Dim sX, sY As Short

    myJoy.GetXY(sX, sY)
    Debug CSRXY(1, 1), “X: “, %DEC4R sX, “, Y: “, %DEC4R sY, CR
End Event

Event myJoy.KnobEvent()
    Dim sZ As Short

    myJoy.GetZ(sZ)
    Debug CSRXY(17, 1), “, Z: “, %DEC4R sZ, CR
End Event
```

附錄

1. 已知問題:
   - V1.0 版本，Restore 指令只會將值存於 RAM 中，斷電後重新開啟，會回到最後的設定值。

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