Gamepad PS

PS2 搖桿控制模組

版本： V2.0

產品介紹：

GamepadPS 模組提供簡易的設定與位置取得指令，搭配 12 個按鈕，讓使用者規劃符合自己需求的操作模式。透過 cmdBUS 與 BASIC Commander 連接，可以用簡單的指令與 PS2 搖桿溝通，取得按鍵資訊製作專屬的應用指令。

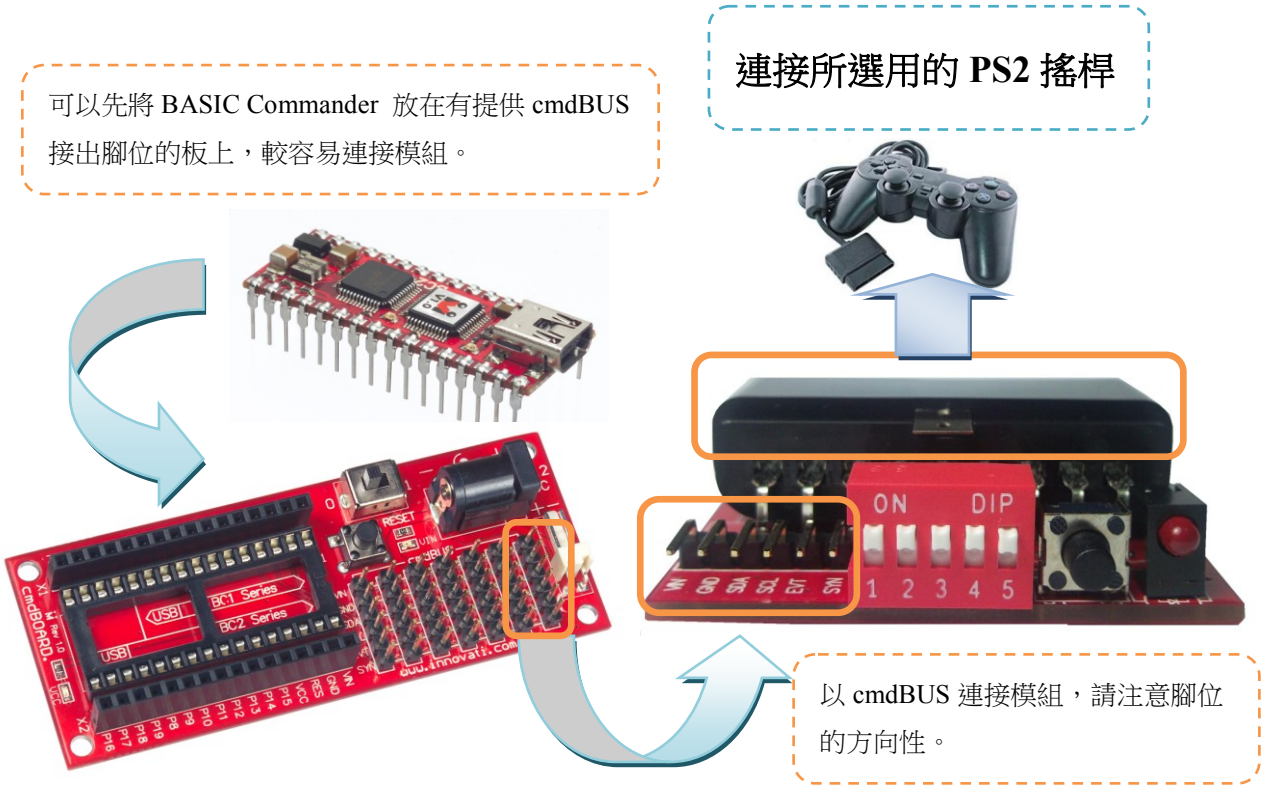
應用方向：

- 連結機器人，設定按鍵達成控制動作與行進等行為。
- 各種測試機具的操作。
- 可與無線 PS2 搖桿結合，控制各種搖控車、飛機等應用。
- 控制利基應用科技的各項應用套件。

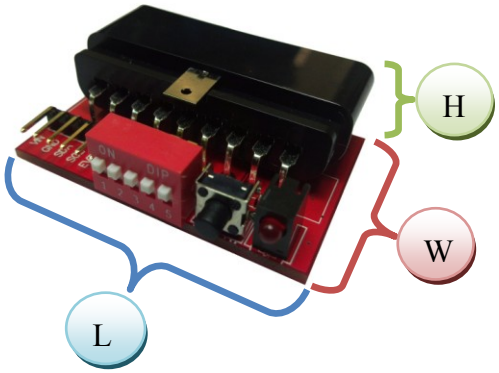
產品特色：

- 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。
- 操縱桿部份，可設定類比回傳、四向或八向操縱桿位置回傳。
- 操縱桿原點範圍可自由設定 0~10%的變動值，避免跳動。
- 方向鍵，可設定四向或八向位置回傳。
- 十二個功能鍵，可單獨控制或組合控制。
- 提供校正功能，並有校正按鈕，操作中可以隨時中斷，進行操縱桿的校正。
- 可自行定義按鈕功能，包括按鍵的連續觸發起動時間，以及連續觸發的速率，都可透過指令設定。
- 可鎖定是否啟動類比操縱桿，避免誤觸。
- 可自行定義搖桿震動強度與時間。
- 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，連接上 PS2 搖桿後，就可透過 BASIC Commander 執行操作。



產品規格：



L * W * H：47 * 31* 16 (㎜)

操作注意事項:

模組操作溫度　　-40 ℃~ 123.8 ℃
模組儲存溫度　　-40 ℃~125℃
本模組適用於原廠 PS2 搖桿，副廠 PS2 搖桿不在保正範圍內。
校正模式操作方式：
1：久按校正鈕(或以軟體方式)，進入校正模式。校正燈恆亮。
2：請將想要校正的操縱桿推到頂點，再沿著頂點繞兩圈以取得 XY 軸的最大與最小值。
3：最後將操縱桿靜置於中心點，等候三秒鐘，讓操縱桿記錄 XY 軸的中心點值。
4：按下功能鍵(△、○、╳、□)，完成校正。校正燈熄滅。
※若校正燈閃爍，表示校正失敗，請重新校正。
若誤入校正模式，可再按一下校正鈕，離開校正。

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 GamepadPS 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 GamepadPS 指令前，請先於程式開頭定義對應參數與編號，例:

Peripheral *ModuleName* As GamepadPS @ *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
校正搖桿相關指令	
LStickCalibration( )	啟動**左邊**操縱桿校正模式。 執行此命令後，操縱桿會進入校正模式校正 LED 長亮，此時請將操縱桿推到頂點，再沿著頂點繞兩圈，以取得 XY 軸向的最大與最小值，最後將操縱桿靜置於中心點等候三秒，讓操縱桿記錄完 XY 軸的中心點值，最後再按壓功能鍵離開，校正 LED 燈熄滅完成校正。 ※校正 LED 燈閃爍，表示校正失敗。 功能鍵：△、○、╳、□
RStickCalibration( )	啟動**右邊**操縱桿校正模式。 執行此命令後，操縱桿會進入校正模式校正 LED 長亮，此時請將操縱桿推到頂點，再沿著頂點繞兩圈，以取得 XY 軸向的最大與最小值，最後將操縱桿靜置於中心點等候三秒，讓操縱桿記錄完 XY 軸的中心點值，最後再按壓功能鍵離開，校正 LED 燈熄滅完成校正。 ※校正 LED 燈閃爍，表示校正失敗。 功能鍵：△、○、╳、□
CmdBUS： **StickCalibration( )** **I2C：** **MID+155+CheckSum1+Dummy**	同時啟動左右邊操縱桿校正模式。 執行此命令後，操縱桿會進入校正模式校正 LED 長亮，此時請將操縱桿推到頂點，再沿著頂點繞兩圈，以取得 XY 軸向的最大與最小值，最後將操縱桿靜置於中心點等候三秒，讓操縱桿記錄完 XY 軸的中心點值，最後再按壓功能鍵離開，校正 LED 燈熄滅完成校正。 ※校正 LED 燈閃爍，表示校正失敗。 功能鍵：△、○、╳、□
SetCalibrationLX(*LxMin,LxCen,LxMax*)	設定**左邊**操縱桿 **X** 軸的校正值，需要輸入三個 Byte 參數，分別是 ***LxMin*** 為設定最小的操縱桿校正值， ***LxCen*** 代表中心點值， ***LxMax*** 設定最大的操縱桿校正值。 ※手動設定時請注意設定順序。 輸入範圍 0~255 之間的整數值。
SetCalibrationLY(*LyMin,LyCen,LyMax*)	設定**左邊**操縱桿 **Y** 軸的校正值，需要輸入三個 Byte 參數，分別是 ***LyMin*** 為設定最小的操縱桿校正值， ***LyCen*** 代表中心點值， ***LyMax*** 設定最大的操縱桿校正值。 ※手動設定時請注意設定順序。 輸入範圍 0~255 之間的整數值。

SetCalibrationRX(*RxMin,RxCen,RxMax*)	設定**右邊**操縱桿 **X** 軸的校正值,需要輸入三個 Byte 參數，分別是 ***RxMin*** 為設定最小的 R 操縱桿校正值， ***RxCen*** 代表中心點值， ***RxMax*** 設定最大的操縱桿校正值。 ※手動設定時請注意設定順序。 輸入範圍 0~255 之間的整數值。
SetCalibrationRY(*RyMin,RyCen,RyMax*)	設定**右邊**操縱桿 **Y** 軸的校正值,需要輸入三個 Byte 參數，分別是 ***RyMin*** 為設定最小的 R 操縱桿校正值，***RyCen*** 代表中心點值， ***RyMax*** 設定最大的操縱桿校正值。 ※手動設定時請注意設定順序。 輸入範圍 0~255 之間的整數值。
GetCalibrationLX(*LxMin,LxCen,LxMax*)	取得**左邊**操縱桿 **X** 軸的校正值。分別將最小值存入 ***LxMin***，中心點值存入 ***LxCen***，最大值存入 ***LxMax***。回傳值為 0~255 之間的整數值。
GetCalibrationLY(*LyMin,LyCen,LyMax*)	取得**左邊**操縱桿 **Y** 軸的校正值。分別將最小值存入 ***LyMin***，中心點值存入 ***LyCen***，最大值存入 ***LyMax***。回傳值為 0~255 之間的整數值。
GetCalibrationRX(*RxMin,RxCen,RxMax*)	取得**右邊**操縱桿 **X** 軸的校正值。分別將最小值存入 ***RxMin***，中心點值存入 ***RxCen***，最大值存入 ***RxMax***。回傳值為 0~255 之間的整數值。
GetCalibrationRY(*RyMin,RyCen,RyMax*)	取得**右邊**操縱桿 **Y** 軸的校正值。分別將最小值存入 ***RyMin***，中心點值存入 ***RyCen***，最大值存入 ***RyMax***。回傳值為 0~255 之間的整數值。
設定相關指令	
RestoreSettings( )	執行指令後將恢復出廠設定，內容如下： ・ 所有校正值範圍設定為： Min=0, Cen=128 ,Max=255 ・操縱桿中心點範圍設定為：5 % ・操縱桿極限範圍值設定為：80 % ・關閉連續按鍵功能 ・操縱桿解析度設定為：128 ・關閉所有事件 ・關閉震動功能
SetLStickDeadZone(*DZx,DZy*)	設定**左邊**操縱桿中心點範圍。 操縱桿中心區域的範圍，經由 ***DZx*** 與 ***DZy*** 分別設定 XY 軸中心區域，以百分比為單位設定輸入範圍為 0~10 之間的整數值。 當操縱桿移動沒有趕出所設定的區域，都會判定操縱桿在中心點。

SetRStickDeadZone(*DZx,DZy*)	設定**右邊**操縱桿中心點範圍。 操縱桿中心區域的範圍，經由 ***DZx*** 與 ***DZy*** 分別設定 XY 軸中心區域，以百分比為單位設定輸入範圍為 0~10 之間的整數值。 當操縱桿移動沒有趕出所設定的區域，都會判定操縱桿在中心點。
GetLStickDeadZone(*DZx,DZy*)	取得**左邊**操縱桿中心範圍設定值。 XY 軸設定資訊分別存放於 ***DZx*** 、***DZy*** 中，回傳範圍為 0~10 之間的整數，單位為百分比。
GetRStickDeadZone(*DZx,DZy*)	取得**右邊**操縱桿中心範圍設定值。 XY 軸設定資訊分別存放於 ***DZx*** 、***DZy*** 中，回傳範圍為 0~10 之間的整數，單位為百分比。
SetLStickSaturation(*SATx,SATy*)	設定**左邊**操縱桿極限範圍值。 操縱桿極限範圍值，經由 ***SATx*** 與 ***SATy*** 分別設定 XY 軸極限範圍值，以百分比為單位設定輸入範圍為 60~100 之間的整數值。 執行指令後，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。
SetRStickSaturation(*SATx,SATy*)	設定**右邊**操縱桿極限範圍值。 操縱桿極限範圍值，經由 ***SATx*** 與 ***SATy*** 分別設定 XY 軸極限範圍值，以百分比為單位設定輸入範圍為 60~100 之間的整數值。 執行指令後，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。
GetLStickSaturation(*SATx,SATy*)	取得**左邊**操縱桿極限範圍設定值。 XY 軸設定資訊分別存放於 ***SATx*** 、***SATy*** 中，回傳範圍為 60~100 之間的整數，單位為百分比。
GetRStickSaturation(*SATx,SATy*)	取得**右邊**操縱桿極限範圍設定值。 XY 軸設定資訊分別存放於 ***SATx*** 、***SATy*** 中，回傳範圍為 60~100 之間的整數，單位為百分比。

<table>
<tr><td>SetLStickRes(RESx,RESy)</td><td>設定左邊操縱桿解析度。
經由 RESx 與 RESy 分別設定 XY 軸解析度，在可辨識的範圍內，所要切割的刻度數。
設定刻度範圍為 0~128 的整數。
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，反向也就是由 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。</td></tr>
<tr><td>SetRStickRes(RESx,RESy)</td><td>設定右邊操縱桿解析度。
經由 RESx 與 RESy 分別設定 XY 軸解析度，在可辨識的範圍內，所要切割的刻度數。
設定刻度範圍為 0~128 的整數。
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，反向也就是由 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。</td></tr>
<tr><td>GetLStickRes(RESx,RESy)</td><td>取得左邊操縱桿解析度設定值。
XY 軸設定資訊分別存放於 RESx，RESy 中，回傳範圍為 0~128 的整數值。</td></tr>
<tr><td>GetRStickRes(RESx,RESy)</td><td>取得右邊操縱桿解析度設定值。
XY 軸設定資訊分別存放於 RESx，RESy 中，回傳範圍為 0~128 的整數值。</td></tr>
<tr><td>SetKeyRepeatFunc(Key_ID)</td><td>設定是否啟動重複輸入判定。
啟動 =1，關閉 =0
<table>
<tr><td>Key_ID</td><td>Bit</td><td>對應按鈕</td><td>十進制</td></tr>
<tr><td></td><td>0</td><td>△</td><td>1</td></tr>
<tr><td></td><td>1</td><td>○</td><td>2</td></tr>
<tr><td></td><td>2</td><td>×</td><td>4</td></tr>
<tr><td></td><td>3</td><td>□</td><td>8</td></tr>
<tr><td></td><td>4</td><td>L1</td><td>16</td></tr>
<tr><td></td><td>5</td><td>R1</td><td>32</td></tr>
<tr><td></td><td>6</td><td>L2</td><td>64</td></tr>
<tr><td></td><td>7</td><td>R2</td><td>128</td></tr>
<tr><td></td><td>8</td><td>Select</td><td>256</td></tr>
<tr><td></td><td>9</td><td>Start</td><td>512</td></tr>
<tr><td></td><td>10</td><td>L3</td><td>1024</td></tr>
<tr><td></td><td>11</td><td>R3</td><td>2048</td></tr>
</table>
EX：若想啟動△、○則 Key_ID 可設定為：&B11(二進制)，或 3(十進制)。</td></tr>
</table>

<table>
<tr><td>GetKeyRepeatFunc(Key_ID)</td><td>取得是否啟動重複輸入判定設定。
啟動 =1，關閉 =0
<table><tr><td rowspan="13">Key_ID</td><td>Bit</td><td>對應按□</td><td>十進制</td></tr><tr><td>0</td><td>△</td><td>1</td></tr><tr><td>1</td><td>○</td><td>2</td></tr><tr><td>2</td><td>×</td><td>4</td></tr><tr><td>3</td><td>□</td><td>8</td></tr><tr><td>4</td><td>L1</td><td>16</td></tr><tr><td>5</td><td>R1</td><td>32</td></tr><tr><td>6</td><td>L2</td><td>64</td></tr><tr><td>7</td><td>R2</td><td>128</td></tr><tr><td>8</td><td>Select</td><td>256</td></tr><tr><td>9</td><td>Start</td><td>512</td></tr><tr><td>10</td><td>L3</td><td>1024</td></tr><tr><td>11</td><td>R3</td><td>2048</td></tr></table></td></tr>
<tr><td>SetRepeatTime(Time)</td><td>設定重複輸入判定時間值。
經由 Time 設定，可輸入範圍為 0~255 之間的整數值，單位為 10 ms。</td></tr>
<tr><td>GetRepeatTime(Time)</td><td>取得重複輸入判定時間設定值。
回傳值存放於 Time。回傳範圍為 0~255 之間的整數值，單位為 10 ms。</td></tr>
<tr><td>SetRepeatRate(Rate)</td><td>設定重複輸入判定速率值。
經由 Rate 設定，可輸入範圍為 0~255 之間的整數值，單位為 10 ms。</td></tr>
<tr><td>GetRepeatRate(Rate)</td><td>取得重複輸入判定速率設定值。
回傳值存放於 Rate。回傳範圍為 0~255 之間的整數值，單位為 10 ms。</td></tr>
<tr><td colspan="2">應用相關指令</td></tr>
<tr><td>CmdBUS：
GetLXYPos(POSx,POSy)
I2C：
Out：MID+113+CheckSum1+Dummy
In：MID+PosX+PosY+CheckSum3</td><td>取得左邊操縱桿座標值。
回傳 XY 座標，分別儲存於 POSx，POSy 中，預設範圍為 -127~+127。</td></tr>
<tr><td>CmdBUS：
GetRXYPos(POSx,POSy)
I2C：
Out：MID+114+CheckSum1+Dummy
In：MID+ PosX + PosY +CheckSum3</td><td>取得右邊操縱桿座標值。
回傳 XY 座標，分別儲存於 POSx，POSy 中，預設範圍為 -127~+127。</td></tr>
</table>

<table>
<tr>
<td>CmdBUS：
GetL4WayValue(Dir)
I2C：
Out：MID+115+CheckSum1+Dummy
In：MID+Dir+CheckSum3</td>
<td>以四向表示方式，取得左邊操縱桿位置。
回傳值存放於 Dir 是方向值，只會有 0~4 的回傳值，分別代表：
0：操縱桿位於中心點　1：操縱桿位於右方→
2：操縱桿位於下方↓　3：操縱桿位於左方←
4：操縱桿位於上方↑</td>
</tr>
<tr>
<td>CmdBUS：
GetR4WayValue(Dir)
I2C：
Out：MID+116+CheckSum1+Dummy
In：MID+Dir+CheckSum3</td>
<td>以四向表示方式，取得右邊操縱桿位置。
回傳值存放於 Dir 是方向值，只會有 0~4 的回傳值，分別代表：
0：操縱桿位於中心點　1：操縱桿位於右方→
2：操縱桿位於下方↓　3：操縱桿位於左方←
4：操縱桿位於上方↑</td>
</tr>
<tr>
<td>CmdBUS：
GetL8WayValue(Dir)
I2C：
Out：MID+117+CheckSum1+Dummy
In：MID+Dir+CheckSum3</td>
<td>以八向表示方式，取得左邊操縱桿位置。
回傳值存放於 Dir 是方向值，只會有 0~8 的回傳值，分別代表：
0：操縱桿位於中心點　1：操縱桿位於右方→
2：操縱桿位於右下方↘3：操縱桿位於下方↓
4：操縱桿位於左下方↙5：操縱桿位於左方←
6：操縱桿位於左上方↖7：操縱桿位於上方↑
8：操縱桿位於右上方↗</td>
</tr>
<tr>
<td>CmdBUS：
Status = GetR8WayValue( )
I2C：
Out：MID+118+CheckSum1+Dummy
In：MID+Dir+CheckSum3</td>
<td>以八向表示方式，取得右邊操縱桿位置。
回傳值存放於 Dir 是方向值，只會有 0~8 的回傳值，分別代表：
0：操縱桿位於中心點　1：操縱桿位於右方→
2：操縱桿位於右下方↘3：操縱桿位於下方↓
4：操縱桿位於左下方↙5：操縱桿位於左方←
6：操縱桿位於左上方↖7：操縱桿位於上方↑
8：操縱桿位於右上方↗</td>
</tr>
<tr>
<td>CmdBUS：
Status = GetKeyStatus( )
I2C：
Out：MID+138+CheckSum1+Dummy
In：
MID+Status_L+Status_H+CheckSum3</td>
<td>取得按鈕狀態存放於 Status 中。
啟動 ＝1，關閉 ＝0
<table>
<tr><td rowspan="7">Status</td><td>Bit</td><td>對應按鈕</td><td>十進制</td></tr>
<tr><td>0</td><td>△</td><td>1</td></tr>
<tr><td>1</td><td>○</td><td>2</td></tr>
<tr><td>2</td><td>×</td><td>4</td></tr>
<tr><td>3</td><td>□</td><td>8</td></tr>
<tr><td>4</td><td>L1</td><td>16</td></tr>
<tr><td>5</td><td>R1</td><td>32</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>6</td><td>L2</td><td>64</td></tr>
<tr><td></td><td>7</td><td>R2</td><td>128</td></tr>
<tr><td></td><td>8</td><td>Select</td><td>256</td></tr>
<tr><td></td><td>9</td><td>Start</td><td>512</td></tr>
<tr><td></td><td>10</td><td>L3</td><td>1024</td></tr>
<tr><td></td><td>11</td><td>R3</td><td>2048</td></tr>
</table>
EX：若 Status = 3 則△、〇被啟動。
</td></tr>
<tr><td>CmdBUS：
GetDir4Way(Dir)
I2C：
Out：MID+141+CheckSum1+Dummy
In：MID+Dir+CheckSum3</td><td>取得方向鍵狀態，以四向方式回傳。
回傳值存放於 Dir，只會有 0~4 的回傳值，分別代表：
0：無方向 1：右方→ 2：下方↓
3：左方← 4：上方↑</td></tr>
<tr><td>CmdBUS：
GetDir8Way(Dir)
I2C：
Out：MID+142+CheckSum1+Dummy
In：MID+Dir+CheckSum3</td><td>取得方向鍵狀態，以八向方式回傳。
回傳值存放於 Dir，只會有 0~8 的回傳值，分別代表：
0：無方向 1：右方→ 2：右下方↘
3：下方↓ 4：左下方↙ 5：左方←
6：左上方↖ 7：上方↑ 8：右上方↗</td></tr>
<tr><td>SetAnalog(Mode)</td><td>設定類比操縱桿狀態。
經由 Mode 設定，可輸入範圍為 0~3，分別代表：
0：關閉類比操縱桿
1：啟動類比操縱桿為類比回傳
2：鎖定類比操縱桿為類比開啟
3：鎖定類比操縱桿為類比關閉
※Default: 1 (開啟)
0、1 為一般啟動、關閉模式，設定後仍可使用操縱桿上按鈕切換模式。
2、3 為鎖定模式，設定後無法使用操縱桿上按鈕切換模式。也無法使用 0、1 模式切換回一般模式，使用時請注意。</td></tr>
<tr><td>CmdBUS：
StartVib(Time,Level)
I2C：
MID+152+CheckSum1+Time+
Level+Dummy</td><td>啟動搖桿震動功能。經由 Time 設定震動時間，Level 設定震動強度。
Time：可設定範圍為 0~255 的整數。
0：持續指動直到下達 StopVib 指令。
1 為 1 秒，之後每加 1 則增加 100 ms。
Level：可設定範圍為 0~255 的整數。
0：不震動，1~255 數字越大震動越強。</td></tr>
</table>

CmdBUS： **StopVib( )** **I2C：** **MID+153+CheckSum1+Dummy**	停止搖桿震動功能。
GetVibStatus(*Status,Time,Level*)	取得搖桿震動狀態。 分別存放 S***tatus,Time,Level*** 中。 ***Status***：目前搖桿震動狀態 0：沒有啟動震動，1：震動啟動中。 ***Time***：剩餘的震動時間。 0：震動狀態為 0，要等待 StopVib 指令停止。 1：剩餘時間小於 1 秒 2~255：剩餘 1+(Time-1)*100 ㎳ ***Level***：設定的震動強度。範圍為 0~255 之間。
GetAnalog(*Mode*)	取得類比搖桿設定狀態。 存放於 ***Mode*** 中，回傳值為 0、1。 分別代表： 0：關閉、1：啟動。(無法取得是否鎖定)
GetConnect(*Status*)	取得控制器連接狀態。 存放於 ***Status*** 中，回傳值為 0、1。 分別代表： 0：沒有偵測到搖桿，1：搖桿正常連接
應用事件相關指令	
SetStickRefreshRate(*Rate*)	設定操縱桿連續變動時，最快產生 EVENT 速率。 ***Rate*** 輸入數值 1~255, 單位 10ms, 其餘數值無效 0=1 皆為 10ms。
GetStickRefreshRate(*Rate*)	取得操縱桿連續變動時，最快產生 EVENT 速率。 回傳值存入 ***Rate***，範圍 1~255, 單位 10ms。
EnableLStickEvent( )	啟動**左邊**操縱桿 **StickEvent** 事件。 經由 **SetStickRefreshRate** 指令決定產生速率。
DisableLStickEvetn( )	關閉**左邊**操縱桿 **StickEvent** 事件。
EnableRStickEvent( )	啟動**右邊**操縱桿 **StickEvent** 事件。 經由 **SetStickRefreshRate** 指令決定產生速率。
DisableRStickEvetn( )	關閉**右邊**操縱桿 **StickEvent** 事件。
EnableL4WayEvent( )	啟動**左邊**操縱桿 **4WayEvent** 事件。
DisableL4WayEvent( )	關閉**左邊**操縱桿 **4WayEvent** 事件。
EnableR4WayEvent( )	啟動**右邊**操縱桿 **4WayEvent** 事件。
DisableR4WayEvent( )	關閉**右邊**操縱桿 **4WayEvent** 事件。
EnableL8WayEvent( )	啟動**左邊**操縱桿 **8WayEvent** 事件。
DisableL8WayEvent( )	關閉**左邊**操縱桿 **8WayEvent** 事件。

EnableR8WayEvent( )	啟動**右邊**操縱桿 **8WayEvent** 事件。
DisableR8WayEvent( )	關閉**右邊**操縱桿 **8WayEvent** 事件。
EnableKeyPressedEvent( )	啟動 **KeyPressedEvent** 事件。
DisableKeyPressedEvent( )	關閉 **KeyPressedEvent** 事件。
EnableKeyRelesedEvent( )	啟動 **KeyRelesedEvent** 事件。
DisableKeyRelesedEvent( )	關閉 **KeyRelesedEvent** 事件。
EnableDir4WayEvent( )	啟動 **Dir4WayEvetn** 事件。
DisableDir4WayEvent( )	關閉 **Dir4WayEvetn** 事件。
EnableDir8WayEvent( )	啟動 **Dir8WayEvetn** 事件。
DisableDir8WayEvent( )	關閉 **Dir8WayEvetn** 事件。

模組提供應用事件:

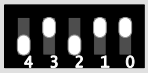
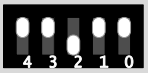
事件名稱 (Event)	啟動條件
LStickEvent	當**左邊**操縱桿開始移動時產生事件。 依照 **SetStickEvent** 所設定的時間頻率回傳。
RStickEvent	當**右邊**操縱桿開始移動時產生事件。 依照 **SetStickEvent** 所設定的時間頻率回傳。
L4WayEvent	當**左邊**操縱桿方向改變時產生事件。與 **SetStickEvent** 無關。
R4WayEvent	當**右邊**操縱桿方向改變時產生事件。與 **SetStickEvent** 無關。
L8WayEvent	當**左邊**操縱桿方向改變時產生事件。與 **SetStickEvent** 無關。
R8WayEvent	當**右邊**操縱桿方向改變時產生事件。與 **SetStickEvent** 無關。
KeyPressedEvent	所有按鈕共用。 當 **RepeatKey** 關閉時，按下按鈕即產生事件。 當 **RepeatKey** 啟動時，按下按鈕，到達 **RepeatTime** 設定時間，以及每隔 **RepeatRate** 設定的時間就會產生事件。
KeyReleasedEvent	所有按鈕共用。 每當偵測到 KeyRelese 的動作就產生事件。
Dir4WayEvent	當方向鍵改變時產生事件。
Dir8WayEvent	當方向鍵改變時產生事件。
CalibrationEndEvent	校正結束後產生事件。Always Enable
ConChangeEvent	偵測到控制器接上或拔除時產生事件。Always Enable

範例程式:

```
Peripheral PS As GamePadPS @ 31          '設定模組編號
Dim b4Dir As Byte                        '儲存取得的方向值
Dim b8WayL,b8WayR As Byte                '儲存取得的操縱桿方向值
Dim wStatus As Word                      '儲存取得的按鈕狀態值

Sub Main()
    PS.EnableKeyPressedEvent()           '啟動按鈕按下事件
    PS.EnableKeyReleasedEvent()          '啟動按鈕放開事件
    Debug "/////   GamePadPS Demo ////"  '終端視窗顯示規劃
    Debug CSRXY(1,2),"Direction:"
    Debug CSRXY(1,3),"RStick8Way:"
    Debug CSRXY(1,4),"LStick8Way:"
    Debug CSRXY(1,5),"GetKeyStatus:"

    Do
        PS.GetDir4Way(b4Dir)             '以四向回傳方式，取得方向鍵狀態
        Debug CSRXY(11,2),b4Dir          '於終端視窗(第 11 行，第 2 列)顯示

        PS.GetR8WayValue(b8WayR)         '以八向回傳方式，取得右邊操縱桿狀態
        Debug CSRXY(12,3),b8WayR         '於終端視窗(第 12 行，第 3 列)顯示

        PS.GetL8WayValue(b8WayL)         '以八向回傳方式，取得左邊操縱桿狀態
        Debug CSRXY(12,4),b8WayL         '於終端視窗(第 12 行，第 4 列)顯示

        Debug CSRXY(15,5),%BIN12 wStatus '於終端視窗(第 15 行，第 5 列)，以二進制顯示
    Loop

End Sub

Event PS.KeyPressedEvent()               '按鈕按下事件
    wStatus = PS.GetKeyStatus            '取得當下按鈕狀態，存放於 wStatus 中
End Event

Event PS.KeyReleasedEvent()              '按鈕放開事件
    wStatus = PS.GetKeyStatus            '更新當下按鈕狀態，存放於 wStatus 中
End   Event
```

附錄

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