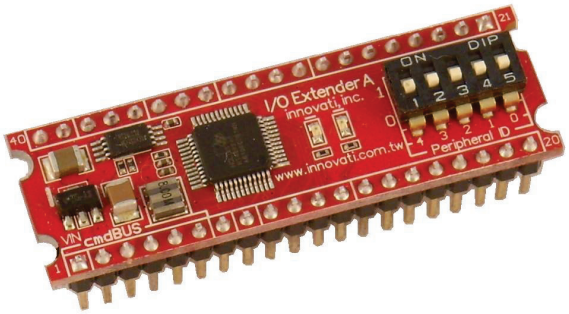
I/O Extender A

輸出入擴充模組

版本: V2.0



產品介紹: 透過利基 I/O Extender A 模組，使用者可以取得除了 Basic Commander 外，增加 3 個埠，共 24 個腳位，每個腳位都能以類似的指令加上模組名稱，做出與 Basic Commander 上各腳位相近的效能。除此之外，更增加各種腳位變化的事件偵測，輸入 Pulse 的計數等多樣功能。

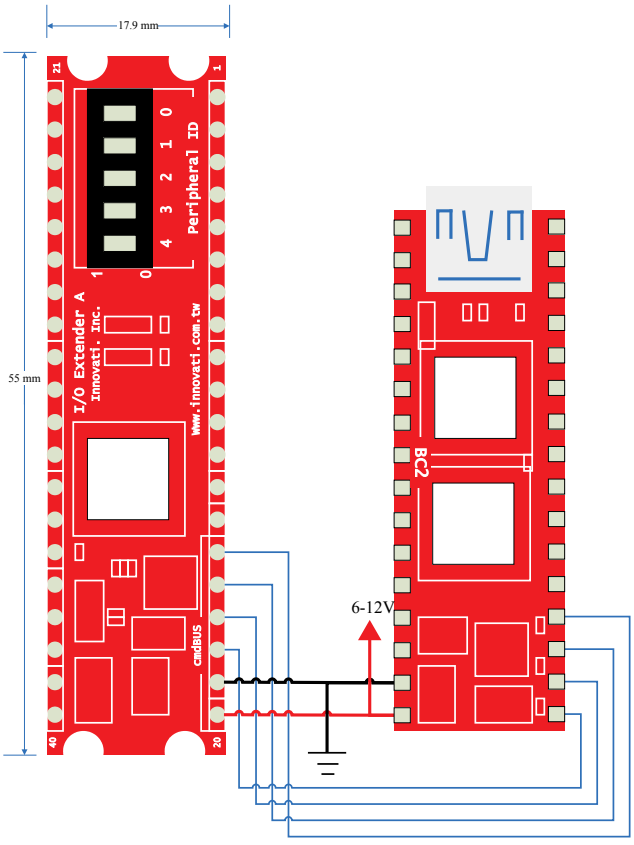
應用方向:

- 可以同時增加獨立控制多個燈號的顯示變化。
- 經由腳位變化能達到感應偵測的效果。
- 應用 AD 轉換腳位，可以將類比訊號轉為數位處理。

產品特色:

- 三個埠各含八個腳位，共二十四根腳位可提供增加應用。
- 獨立腳位訊號寬度量測與訊號個數計數，精準度可至微秒(μ s)。
- 獨立腳位可變頻率輸出。
- 八通道類比轉數位(AD)量測輸入。
- 可提供腳位電壓高低變化各八個事件，最多可同時偵測十六個腳位變化。
- 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:

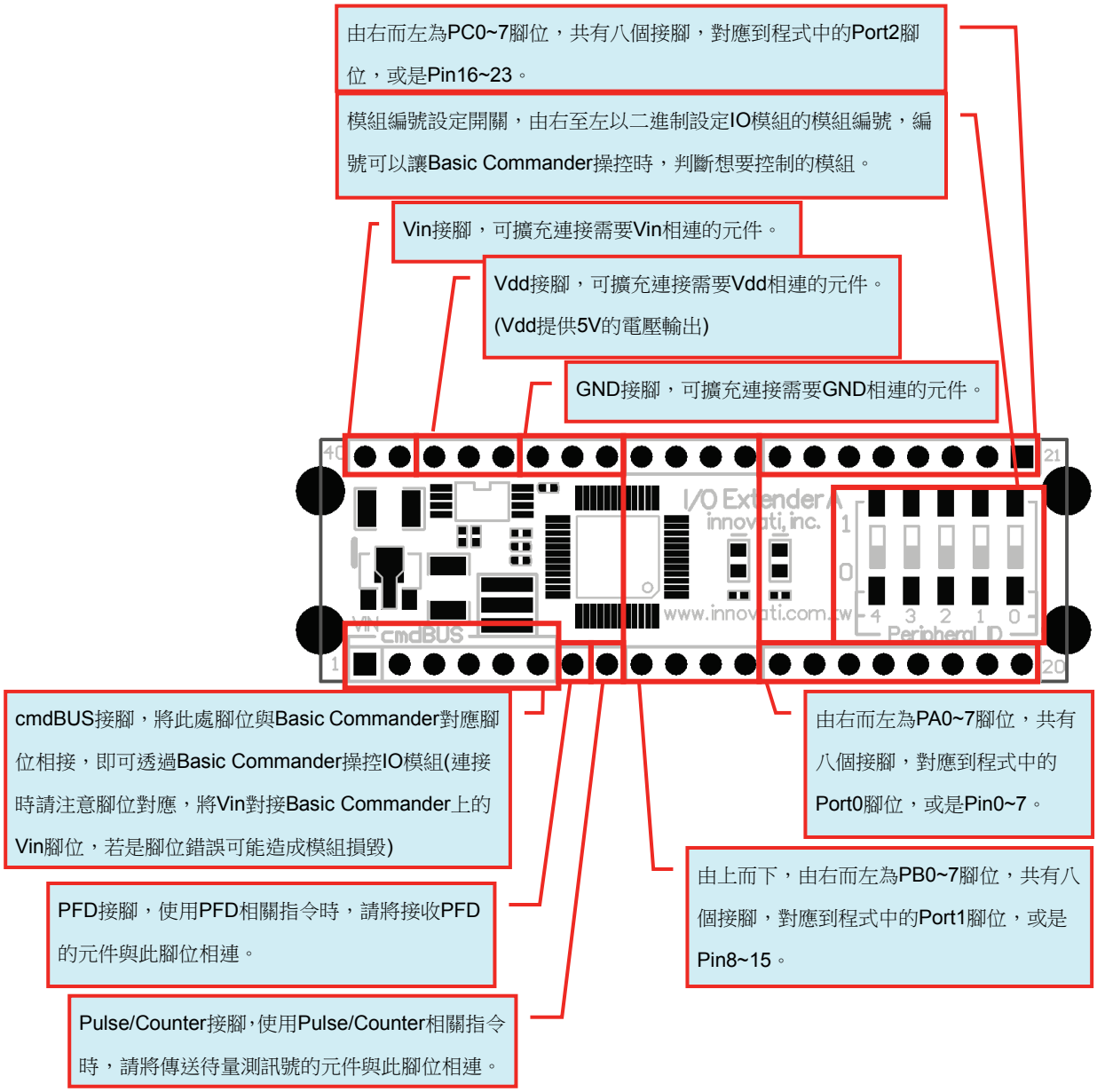


圖 1: 模組腳位與開關介紹

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{IN}= 7.5	Conditions				
I_{DD}	Operating Current	—	No I/O	—	4	—	mA
V_{IL}	Input Low Voltage for I/O Ports.	—	—	0	—	1.5	V
V_{IH}	Input High Voltage for I/O Ports.	—	—	3.5	—	5	V
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
V_{AD}	A/D Input Voltage	—	—	0	—	5	V
I_{ADC}	Additional Power Consumption if A/D Converter is Used	—	—	—	1.5	3	mA

表 1: 電壓電流特性 (於 25 ℃之環境)

PWM 通道輸出頻率約為 40Hz。

操作注意事項:

操作溫度　　0 ℃~70℃
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 I/O Extender A 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 I/O Extender A 指令前，請先於程式開頭定義對應參數與編號，例:

Peripheral *ModuleName* As IOExtenderA @ *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
各埠腳位讀取寫入相關指令	
CmdBUS： **High(*BitNum*)** **I2C：** **MID+120+CheckSum1+ BitNum+CheckSum2+Dummy**	設定 ***BitNum*** 所指定腳位輸出高電位 (***BitNum*** 範圍為 0~23，若設定超出此範圍則不動作) *1
CmdBUS： **In(*BitNum*, *Value*)** **I2C：** **Out：** **MID+118+CheckSum1+ BitNum+CheckSum2+Dummy** **In：** **MID+Value+CheckSum3**	取得 ***BitNum*** 所指定腳位的電位，為 0 或 1，存放於 ***Value*** (***BitNum*** 範圍為 0~23，若設定超出此範圍則不動作) *2
CmdBUS： **Low (*BitNum*)** **I2C：** **MID+119+CheckSum1+ BitNum+CheckSum2+Dummy**	設定 ***BitNum*** 所指定腳位輸出低電位 (***BitNum*** 範圍為 0~23，若設定超出此範圍則不動作) *1

CmdBUS： **PulseOut(*BitNum*, *Mode*, *Width*)** **I2C：** **MID+113+CheckSum1+BitNum +Mode_L+Mode_H +Width_L+Width_H+CheckSum2+Dummy**	從 ***BitNum*** 所指定的腳位，輸出以毫秒(ms)為單位，長度為 ***Width*** 的 pulse，***Width*** 可以輸入 0~65535 的整數值，***Mode*** 為 0 則輸出低電位，若 ***Mode*** 為 1，則輸出高電位(***BitNum*** 範圍為 0~23，***Mode*** 則只可輸入 0 或 1，當輸入超出此範圍的值，指令都不會被執行) *3
CmdBUS： **ReadPort(*PortNum*, *Value*)** **I2C：** **Out：** **MID+143+CheckSum1+PortNum +CheckSum2+Dummy** **In：** **MID+Value+CheckSum3**	根據 ***PortNum*** 選擇要讀取的埠，***PortNum*** 可以輸入 0~2 的整數值，再以 ***Value*** 儲存該埠各腳位之電位，回傳值為 0~255 的整數值 *4
ReadPort0(*Value*) **ReadPort1(*Value*)** **ReadPort2(*Value*)**	以 ***Value*** 儲存 Port0~2 各腳位之電位，回傳值為 0~255 的整數值*4
Toggle(*BitNum*)	將 ***BitNum*** 指定腳位的輸出電位反相(***BitNum*** 範圍為 0~23，若設定超出此範圍則不動作) *1
TogglePort0() **TogglePort1()** **TogglePort2()**	將 Port0~2 各腳位的輸出電位反相 *4
CmdBUS： **WritePort(*PortNum*, *Value*)** **I2C：** **MID+142+CheckSum1+PortNum +Value+CheckSum2+Dummy**	根據 ***PortNum*** 選擇要輸出的埠，請輸入 0~2 之間的整數值，再以 ***Value*** 輸出該埠各腳位之電位 ***Value*** 可以為範圍在 0~255 之間的整數值 *4
WritePort0(*Value*) **WritePort1(*Value*)** **WritePort2(*Value*)**	將 ***Value*** 所設定的值，可以設定範圍為 0~255 的整數值，於 Port0~2 輸出 *4
設定腳位模式相關指令 (讀取或設定皆以 0 為輸出模式，1 為輸入模式)	
GetDirPort(*Port*, *Result*)	取得 ***Port*** 所指定的埠各腳位設定的輸出入模式，存入 ***Result*** 中，***Port*** 請輸入 0~2 之間的整數值，***Result*** 會回傳範圍在 0~255 間的整數值
GetDirPort0(*Result*)	取得埠 0 各腳位設定的輸出入模式，存入 ***Result*** 中，***Result*** 會回傳範圍在 0~255 間的整數值

GetDirPort1(*Result*)	取得埠 1 各腳位設定的輸出入模式，存入 ***Result*** 中，***Result*** 會回傳範圍在 0~255 間的整數值
GetDirPort2(*Result*)	取得埠 2 各腳位設定的輸出入模式，存入 ***Result*** 中，***Result*** 會回傳範圍在 0~255 間的整數值
CmdBUS： **SetDirPort(*Port*, *Dir*)** **I2C：** **MID+134+CheckSum1+Port +Dir+CheckSum2+Dummy**	由 ***Dir*** 設定 ***Port*** 所指定的埠各腳位設定的輸出入模式，***Port*** 請輸入 0~2 之間的整數值，***Dir*** 請輸入 0~255 間的整數值
SetDirPort0(*Dir*)	由 ***Dir*** 設定埠 0 各腳位設定的輸出入模式，***Dir*** 請輸入 0~255 間的整數值
SetDirPort1(*Dir*)	由 ***Dir*** 設定埠 1 各腳位設定的輸出入模式，***Dir*** 請輸入 0~255 間的整數值
SetDirPort2(*Dir*)	由 ***Dir*** 設定埠 2 各腳位設定的輸出入模式，***Dir*** 請輸入 0~255 間的整數值
啟動關閉事件相關指令	
DisablePinLowEvent1()	停止 **PinLowEvent1~8** 的產生
…	
DisablePinLowEvent8()	
DisablePinHighEvent1()	停止 **PinHighEvent1~8** 的產生
…	
DisablePinHighEvent8()	
DisablePort0ChangeEvent()	停止 **Port0~2ChangeEvent** 的產生
DisablePort1ChangeEvent()	
DisablePort2ChangeEvent()	
DisablePort0MatchEvent()	停止 **Port0~2MatchEvent** 的產生
DisablePort1MatchEvent()	
DisablePort2MatchEvent()	
DisablePort0LowEvent()	停止 **Port0~2LowEvent** 的產生
DisablePort1LowEvent()	
DisablePort2LowEvent()	
DisablePort0HighEvent()	停止 **Port0~2HighEvent** 的產生
DisablePort1HighEvent()	
DisablePort2HighEvent()	

EnablePinLowEvent1(*BitNumber*, *Repeat*) **…** **EnablePinLowEvent8(*BitNumber*, *Repeat*)**	偵測 ***BitNumber*** 指定腳位，若是腳位為低電位，就啟動 **PinLowEvent1 ~ 8**，並根據 ***Repeat*** 值，設定重複產生 Event 的時間，單位為毫秒(ms) (若 ***Repeat*** 值設為 0，則不論處於低電位時間多長，只會有一次事件產生)，***BitNumber*** 請輸入 0~23 之間的整數值，***Repeat*** 請輸入 0~65535 之間的整數值
EnablePinHighEvent1 (*BitNumber*, *Repeat*) **…** **EnablePinHighEvent8 (*BitNumber*, *Repeat*)**	偵測 ***BitNumber*** 指定腳位，若是腳位為高電位，就啟動 **PinHighEvent1 ~ 8**，並根據 ***Repeat*** 值，設定重複產生 Event 的時間，單位為毫秒(ms) (若 ***Repeat*** 值設為 0，則不論處於高電位時間多長，只會有一次事件產生) ，***BitNumber*** 請輸入 0~23 之間的整數值，***Repeat*** 請輸入 0~65535 之間的整數
EnablePort0ChangeEvent(*BitMask*) **EnablePort1ChangeEvent(*BitMask*)** **EnablePort2ChangeEvent(*BitMask*)**	根據 ***BitMask*** 所對應的 Port0~2 腳位，偵測到電位變化(包含高電位轉低電位，或低電位轉高電位)，就產生相對應的 **Port0~2ChangeEvent**。***BitMask*** 的設定，是以相對應的腳位設為 1，則啟動該腳位的事件判斷。例如，在設定 Port0 時，若 ***BitMask*** 值設定為 1，則 Port0 的第一個腳位變化會啟動事件，若設為 3，則第一個或第二個腳位變化都會啟動事件。***BitMask*** 請輸入 0~255 之間的整數值
EnablePort0MatchEvent(*Pattern*, *Repeat*) **EnablePort1MatchEvent(*Pattern*, *Repeat*)** **EnablePort2MatchEvent(*Pattern*, *Repeat*)**	當 Port0~2 所收到的值與 ***Pattern*** 所設定值相同時，就啟動 **Port0~2MatchEvent**，並且在每次該值維持如 ***Repeat*** 所設定的時間後，重複產生該事件 (***Repeat*** 之單位為 ms，若設為 0，則不會重複產生事件) ，***Pattern*** 請輸入 0~255 之間的整數值，***Repeat*** 請輸入 0~65535 之間的整數
EnablePort0LowEvent(*BitMask*) **EnablePort1LowEvent(*BitMask*)** **EnablePort2LowEvent(*BitMask*)**	根據 ***BitMask*** 所對應的 Port0~2 腳位，偵測到低電位，就產生相對應的 **Port0~2LowEvent**。***BitMask*** 請輸入 0~2 之間的整數值
EnablePort0HighEvent(*BitMask*) **EnablePort1HighEvent(*BitMask*)** **EnablePort2HighEvent(*BitMask*)**	根據 ***BitMask*** 所對應的 Port0~2 腳位，偵測到高電位，就產生相對應的 **Port0~2HighEvent**。***BitMask*** 請輸入 0~2 之間的整數值
***Status*=GetPort0LowEventStatus()** ***Status*=GetPort1LowEventStatus()**	取得 Port0~2 相對應產生 **Port0~2LowEvent** 的腳位，存放於 ***Status*** 中 (在每次執行此命

***Status*=GetPort2LowEventStatus()**	令後，狀態值就會清回 0)，Status 請設定一個 byte 值儲存回傳值，***Status*** 回傳值為 0~255 之間的整數值
***Status* =GetPort0HighEventStatus()**	取得 Port0~2 相對應產生 **Port0~2HighEvent** 的腳位，存放於 ***Status*** 中(在每次執行此命令後，狀態值就會清回 0)，，***Status*** 回傳值為 0~255 之間的整數值
***Status* =GetPort1HighEventStatus()**	
***Status* =GetPort2HighEventStatus()**	
***Status* =GetPort0ChangeEventStatus()**	取得 Port0~2 相對應產生 **Port0~2ChangeEvent** 的腳位，存放於 ***Status*** 中(在每次執行此命令後，狀態值就會清回 0)，，***Status*** 回傳值為 0~255 之間的整數值
***Status* =GetPort1ChangeEventStatus()**	
***Status* =GetPort2ChangeEventStatus()**	
PFD 腳位相關指令	
PFDOff()	關閉 PFD 輸出
PFDOn()	開啟 PFD 輸出
SetPFD(*Freq*)	以 ***Freq*** 設定 PFD 的輸出頻率，***Freq*** 請輸入 0~65535 之間的整數值 *5
Pulse/Counter 腳位相關指令	
CounterOff()	停止 Counter 的計數
CounterOn(*Mode*,*Value16*,*EventEn*)	根據輸入的 ***Mode*** 值，有以下四種不同模式: 0➔根據 ***Value16*** 所設定的時間，單位為毫秒(ms)，計數輸入腳位出現高電位的次數 1➔根據 ***Value16*** 所設定的時間，單位為毫秒(ms)，計數輸入腳位出現低電位的次數 2➔計數腳位出現高電位的次數，達到 ***Value16*** 所設定的值就停止計數 3➔計數腳位出現低電位的次數，達到 ***Value16*** 所設定的值就停止計數 各模式條件達成時，根據 ***EventEn***，若設定為 1，就會產生 **CountStopEvent** 事件 *6 *7 ***Value16*** 請輸入 0~65535 之間的整數值， ***EventEn*** 請輸入 0 或 1
***Status* = GetCounter(*Counter*)**	可由 ***Status*** 取得目前計數器的狀態，並且可由 ***Counter*** 中得到計數器的計數值 (**Status** 若為 1 就代表計數結束)，***Status*** 會回傳 0 或 1，***Counter*** 則是回傳範圍在 0~65535 間的整數值
***Status* = GetPulseWidth(*Width*)**	取得量測 Pulse 之狀態，若 ***Status*** 為 1 則代表量測停止，並將 Pulse 量測所得值，存於 ***Width*** 中，單位為微秒(μs) ，***Status*** 會回傳 0 或 1，***Width*** 則是回傳範圍在 0~65535 間的整數值

PulseInOff()	停止 Pulse 的量測
PulseInOn(*Mode*, *EventEn*)	以 ***Mode*** 設定要量測輸入波形的電位(0 為量測低電位，1 為量測高電位)，並以 ***EventEn*** 設定，0 為不啟動，1 為啟動，是否要於量測結束後啟動事件 (**PulseMeasureEndEvent**)，當所要量測的波形超過 65535 微秒(μs)仍未結束時，會啟動 **PulseOverflowEvent** 的事件，並且將計數歸零，重新計數。 *7
AD 轉換相關指令	
CmdBUS： **GetADC (*ChanNum*, *Value*)** **I2C：** **Out：** **MID+133+CheckSum1+ChanNum +CheckSum2+Dummy** **In：** **MID+Value_L+Value_H+CheckSum3**	取得 ***ChanNum***，範圍為 0~7 之間的整數值，指定頻道的電位轉換值，存放在 ***Value*** 中 (***Value*** 範圍為 0~1023，當輸入電壓高於 VDD 則皆以 1023 表示)
CmdBUS： **SetADC (*ChanNum*)** **I2C：** **MID+132+CheckSum1+ChanNum +CheckSum2+Dummy**	根據 ***ChanNum*** 值設定 ADC 所使用的頻道如下: 0➔ADC 轉換關閉 1➔CH0 開啟 2➔CH0~CH1 開啟 3➔CH0~CH2 開啟 4➔CH0~CH3 開啟 5➔CH0~CH4 開啟 6➔CH0~CH5 開啟 7～8➔CH0~CH7 開啟 其中 CH0~CH7 分別是 PA0~PA7
計時器(Timer)相關指令	
CmdBUS： **GetTimer(*Timer*)** **I2C：** **Out：** **MID+109+CheckSum1+Dummy** **In：** **MID+Time_L+Time_H+CheckSum3**	讀取計時器值存放於 ***Timer*** 中 (計時器值在每次啟動計算就會清為零)，回傳值為 0~65535 之間的整數值

CmdBUS： **SetTimer(*Mode*, *Timer*)** **I2C：** **MID+112+CheckSum1+Mode +Time_L+Time_H+CheckSum2+Dummy**	以 ***Mode*** 設定計時器的模式，1 為重複模式，0 為單次模式，並以 ***Timer*** 設定所要計時的時間值，單位為毫秒(ms)，請輸入 0~65535 之間的整數值
CmdBUS： **TimerOff** **I2C：** **MID+111+CheckSum1+Dummy**	停止計時器計時
CmdBUS： **TimerOn** **I2C：** **MID+110+CheckSum1+Dummy**	啟動計時器計時

*1: 此命令會將腳位設為輸出模式
*2: 此命令會將腳位設為輸入模式
*3: 當原先輸出電位已經是高電位，又再要求輸出高電位的 Pulse，模組就會先傳送一小段的低電位再轉為高電位
*4: 此命令不會改變腳位之輸出入模式，請使用 SetDirPort 相關指令來幫助轉換輸出入模式
*5: 設定新的 PFD 值之後，需要再次執行一次 PFDOn()，才會產生需要的訊號
*6: 在模式 0 與 1，若是在設定的時間結束前，已經計數到 65535，計數會自動停止，同時傳送 CountStopEvent。在模式 2 與 3，若是將目標值設定為 0，則會在計數到 65535 時停止計數，並產生 CountStopEvent。
*7: 執行此命令後須等待至少 200 μs，此功能才開始執行

模組提供應用事件:

事件名稱 (Event)	啟動條件
CounterStopEvent	執行 **CounterOn**，並且設定啟動事件，當計數條件達成時
PinLowEvent1~8	執行 **EnablePinLowEvent1~8**，當所設定的腳位偵測到低電位時
PinHighEvent1~8	執行 **EnablePinHighEvent1~8**，當所設定的腳位偵測到高電位時
Port0~2ChangeEvent	執行 **EnablePort0~2ChangeEvent**，當所設定的埠偵測到電位變化時
Port0~2HighEvent	執行 **EnablePort0~2HighEvent**，當所設定的腳位偵測到高電位時
Port0~2LowEvent	執行 **EnablePort0~2LowEvent**，當所設定的腳位偵測到低電

	位時
Port0~2MatchEvent	執行 **EnablePort0~2 Match Event**，當所設定的埠偵測到與設定值相同的輸入時
PortNumberErrorEvent	當所設定的腳位(0~23)或埠號(0~2)超出範圍時
PulseInOverflowEvent	執行 **PulseInOn**，若所偵測到的訊號電位維持超過 65535 微秒(μs)時
PulseMeasureEndEvent	執行 **PulseInOn**，當偵測訊號電位離開所要偵測電位值時
TimerOverFlowEvent	執行 **TimerOn**，當到達所設定的計數時間時

範例程式:

```
Peripheral myIO As IOExtenderA @ 0          '   設定模組編號為 0

Dim PinLevel As Byte                        '   儲存腳位狀態
Dim CountStop As Byte                       '   儲存計數器停止狀態
Dim ChValue As Word                         '   儲存類比轉數位的值
Dim TimeUp As Byte                          '   儲存倒數計時完成的狀態

Sub Main()
    myIO.High(0)                            '   由第 0 腳位輸出高電位
    myIO.Low(1)                             '   由第 1 腳位輸出低電位
    myIO.In(2, PinLevel)                    '   取得第 2 腳位輸入之電位
    myIO.PulseOut(3, 1, 2)                  '   由第 3 腳輸出 2ms 的高電位波形
    myIO.Toggle(0)                          '   將第 0 腳位的輸出電位反相 (原為高電位將變為低電位)

    myIO.SetDirPort1(0)                     '   設定埠 1 皆轉為輸出

    myIO.EnablePinLowEvent1(16, 0)          '   於第 16 腳位偵測高電位轉低電位狀態
    myIO.DisablePinLowEvent1()              '   關閉低電位狀態偵測

    myIO.SetPFD(1000)                       '   設定 PFD 輸出 1k Hz 的波形
    myIO.PFDOn()                            '   啟動 PFD 輸出
    myIO.PFDOff()                           '   關閉 PFD 輸出

    CountStop=0
    myIO.CounterOn(0, 2000, 1)              '   啟動計數器，計算在兩秒內，在 Counter 腳位出現的正緣數

'   下面的迴圈在計數器時間結束才會跳出
    Do
    Loop Until CountStop>0

    myIO.CounterOff()                       '   關閉計數器
```

```
        myIO.SetPWM1(4, 128)                    '   將 PWM 輸出於第 4 腳位，Duty Cycle 為 0.5 (128/256)
        myIO.PWM1On()                           '   開啟 PWM1 的輸出
        myIO.PWM1Off()                          '   關閉 PWM1 的輸出

        myIO.SetADC (1)                 '   開啟 PB0 作為 ADC 轉換的輸入
        myIO.GetADC (0, ChValue)        '   取得 CH0 的類比輸入轉換為數位的值，存入 ChValue 中

        myIO.SetTimer(0, 1000)                  '   設定倒數計時 1 秒
        TimeUp=0
        myIO.TimerOn()                          '   開啟倒數計時

'       下面的迴圈在倒數計時器倒數結束才會跳出
        Do
        Loop Until TimeUp>0

        myIO.TimerOff()                         '   關閉倒數計時
End Sub

Event MyIO.PinLowEvent1()
        Debug "偵測到低電位出現!", CR
End Event

Event MyIO.CounterStopEvent()
        Debug "計數結束!", CR
        CountStop=1
End Event

Event MyIO.TimerOverflowEvent()
        Debug "倒數計時結束!", CR
        TimeUp=1
End Event
```

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