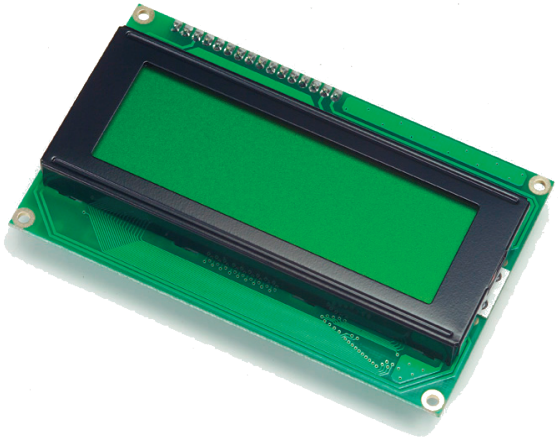
LCD 4x20A

4x20 字元 LCD 顯示模組

版本: V2.0



產品介紹: 利基 LCD 4x20A 模組提供多樣化顯示功能，並且可透過簡單的聯接，直接由利基之 BASIC Commander 操控各項應用。在此模組上可同時顯示四行訊息，各二十字元，另外透過游標控制指令，可隨時變更任意位置的顯示字元。此模組有背光功能，藉由點亮背光，可以讓訊息更容易讀取。另外也可以透過自訂字元，顯示自己所想要的特殊字型。

應用方向:

- 可加上 RTC 模組即時顯示時間，就是簡單的電子時鐘。
- 於各種應用中即時顯示操作狀態。
- 不經由 PC 直接將錯誤狀態或錯誤訊息顯示於螢幕。
- 藉由自訂字元創造特殊圖案，提供創意訊息。

產品特色:

- 可以同時顯示四列各二十個字元。
- 每個字元解析度為 5x8 dot。
- 可透過輸入 ASCII 碼顯示對應字元。
- 直接使用顯示指令，模組將自動轉換，根據字串或是常數輸入，轉為對應的字元或數字顯示。
- 透過設定，背光可提供 255 段多種亮度顯示。
- 連續輸入時，模組會直接換行顯示，並自動覆蓋原本顯示訊息。
- 各種移動游標顯示方式，可以直接設定游標位址，任意跳行或跳列顯示，也可以設定 Tab 值，執行自動前移字元數，輕鬆達成版面控制需求。當不確定游標位置時，直接輸入 Home 指令，就會回到畫面起始點。
- 多樣化清除螢幕指令，可設定全螢幕清除，往前清除單一字元，自游標清除至列尾，或是由游標處清除到螢幕尾端。
- 可設定自訂字元顯示各種創意文字。
- 不使用時可單獨執行關閉螢幕指令，節省耗電。
- 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(如圖 1)，就可透過 BASIC Commander 執行操作。Vin 與 GND 請與提供 6~12V 之電源與地端連接。

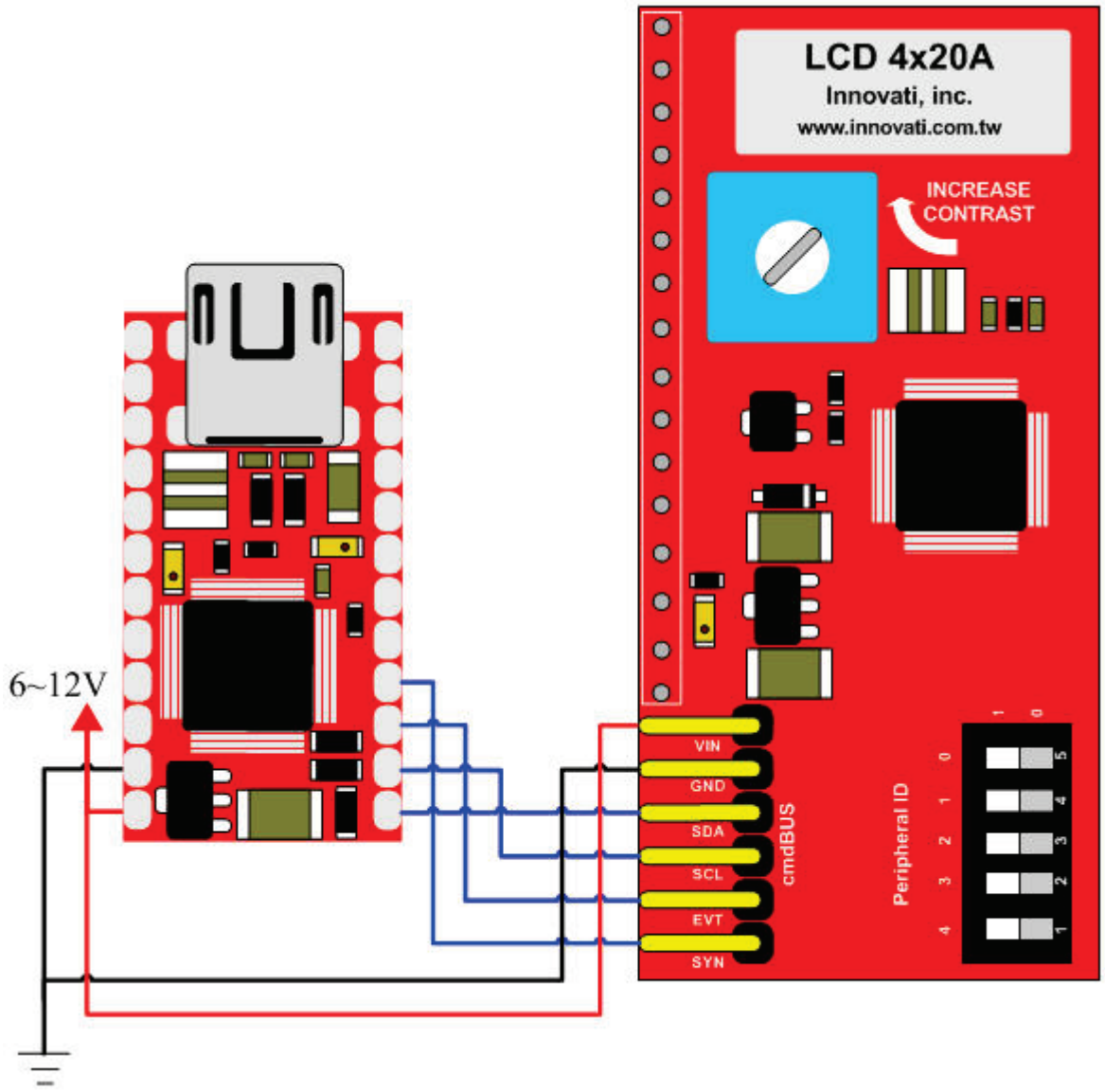


圖 1: 連接 LCD 4x20 A 與 BASIC Commander

產品規格:

cmdBUS接腳，將此處腳位與BASIC Commander對應腳位相接，即可透過BASIC Commander操控LCD模組，連接時請注意腳位對應，將Vin對接BASIC Commander上的Vin腳位，若是腳位錯誤可能造成模組損毀

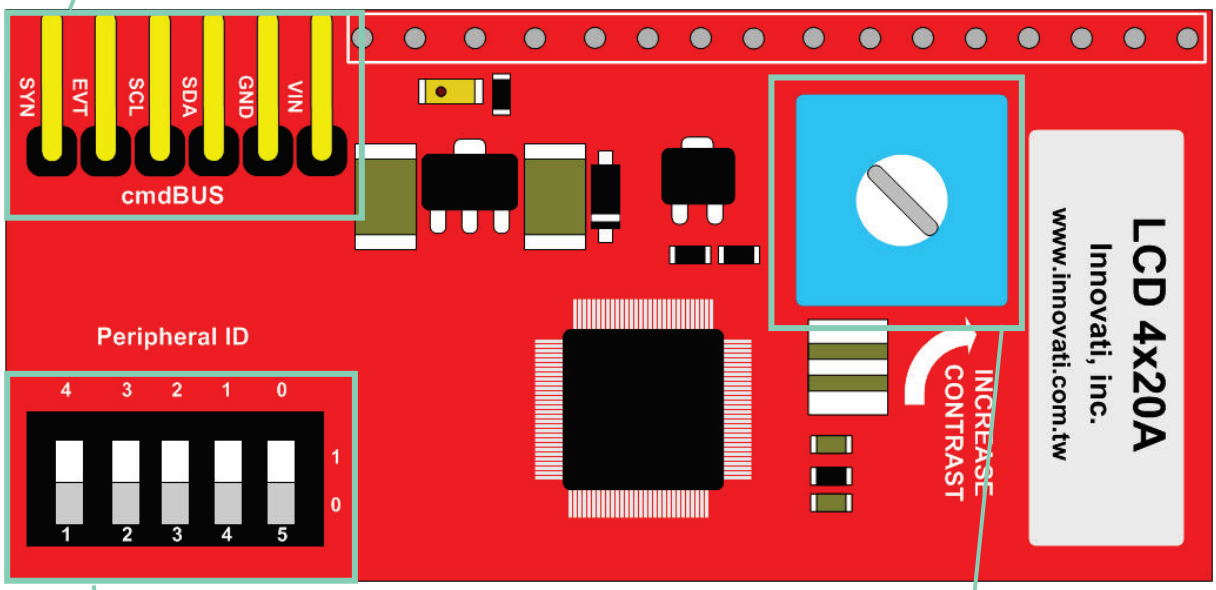


圖 2: 模組腳位與開關介紹

模組編號設定開關，由右至左以二進制設定LCD模組的模組編號，編號可以讓BASIC Commander操控時，判斷想要控制的模組。(請參考附錄2)

對比調整螺絲，請以十字起旋轉，順時針方向旋轉可調高對比，逆時針方向調低對比，可調整範圍有限，請勿過度旋轉，以免造成零件毀損。

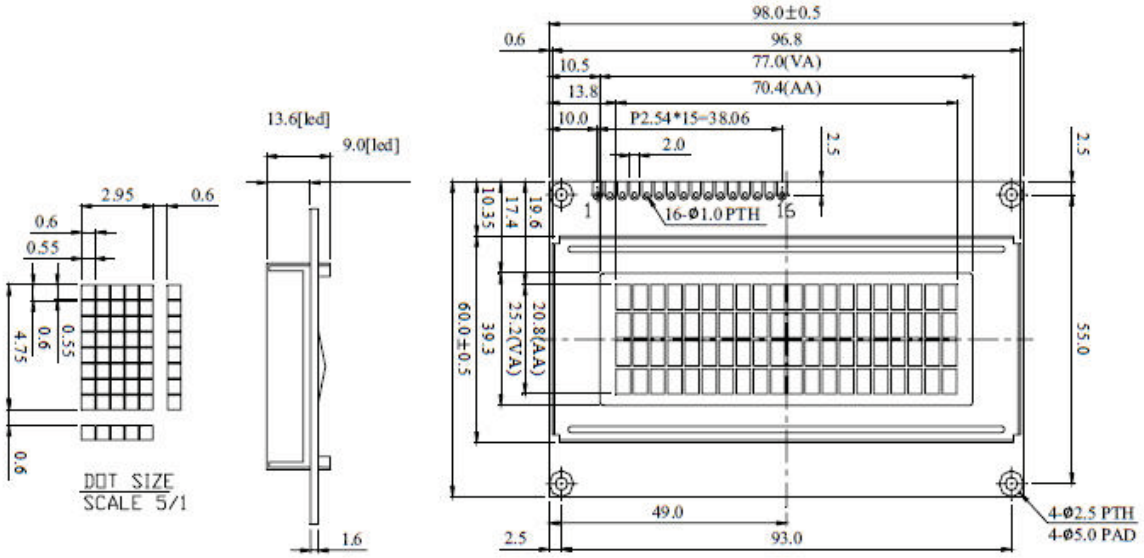


圖 3: LCD 螢幕尺寸(單位 mm)

Item	Standard Value	Unit
Display type	20 characters x 4 Lines	—
Module dimension (L x W x H)	98.0 x 60.0 x 13.1 (Max) - LED array B/L STN Positive / 6 o'clock / Transflective	mm
Viewing Area	77.0 x 25.2	mm
Active Area	70.4 x 20.8	mm
Dot Size	0.55 x 0.55	mm
Dot Pitch	0.60 x 0.60	mm
Character size (L x W)	2.95 x 4.75	mm
Character pitch (L x W)	3.55 x 5.35	mm

表 1: LCD 螢幕規格

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
View Angle	(V) θ	CR≧2	10		45	deg
	(H) φ	CR≧2	-30		30	deg
Contrast Ratio	CR	—		3		—
Response Time 25℃	T rise	—		110	170	ms
	T fall	—		150	200	ms

表 2: LCD 螢幕視角與對比

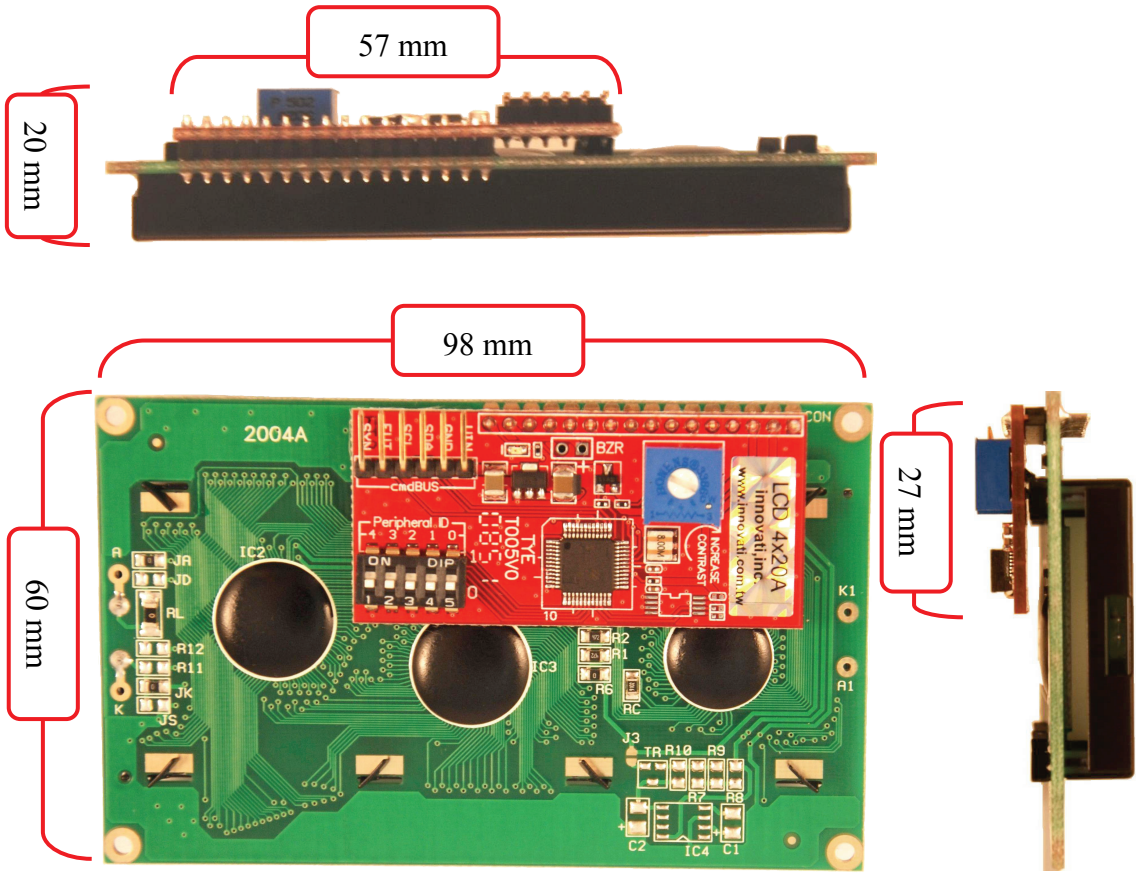


圖 4: 模組外觀尺寸

操作注意事項:

操作溫度　　0 ℃~50 ℃ (> 180 hr)

儲存溫度　　-20 ℃~ 70 ℃ (> 180 hr)

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	Operating Current	7.5	Backlight On	—	185	—	mA
			Backlight Off	—	5	—	mA

表 3: 工作電流特性 (於 25 ℃之環境)

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 LCD 4x20 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 LCD 4x20 A 指令前，請先於程式開頭定義對應參數與編號，例:

Peripheral *ModuleName* As LCD2x16A @ *ModuleID*

I2C 通訊協議(Protocol):

請注意下列通訊協議中的"+"號，代表連接，並非運算符號。

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	指令功能敘述
移動游標相關指令	
CarriageReturn() **CR()**	讓游標移動到下一列
CursorColumn(*Col*) **CursorCol(*Col*)**	將游標移動至 ***Col*** 指定的行，***Col*** 請輸入 1~20 之間的整數值
CursorDown()	將游標下移一列
CursorLeft()	將游標向左移一個字元
CmdBUS： **CursorRC(*Row*, *Col*)** **I2C：** **MID+101+CheckSum1+** **Col+Row+CheckSum2+Dummy**	將游標移動到 ***Row*** 所指定的列與 ***Col*** 所指定的行，***Row*** 請輸入 1~4 之間的整數值，***Col*** 請輸入 1~20 之間的整數值
CursorRight()	將游標向右移一個字元
CursorRow(*Row*)	將游標移動至 ***Row*** 指定的列，***Row*** 請輸入 1 到 4 之間的整數值
CursorUp()	將游標上移一列
Home()	將游標移動到第一行第一列
Tab()	讓游標向右移動 **Tab** 設定的字元個數
清除顯示相關指令	
CmdBUS： **BackSpace()** **I2C：** **MID+94+CheckSum1+Dummy**	將游標往前移動一個字元，並清除在此位置上顯示的字元
CmdBUS： **Clear()** **I2C：** **MID+88+CheckSum1+Dummy**	清除畫面上所有顯示的字元
CmdBUS： **ClearEOL()** **I2C：** **MID+99+CheckSum1+Dummy**	清除由游標所在位置開始，到列尾的所有字元
CmdBUS： **ClearEOS()** **I2C：** **MID+100+CheckSum1+Dummy**	清除由游標所在位置開始，到螢幕最後顯示的所有字元

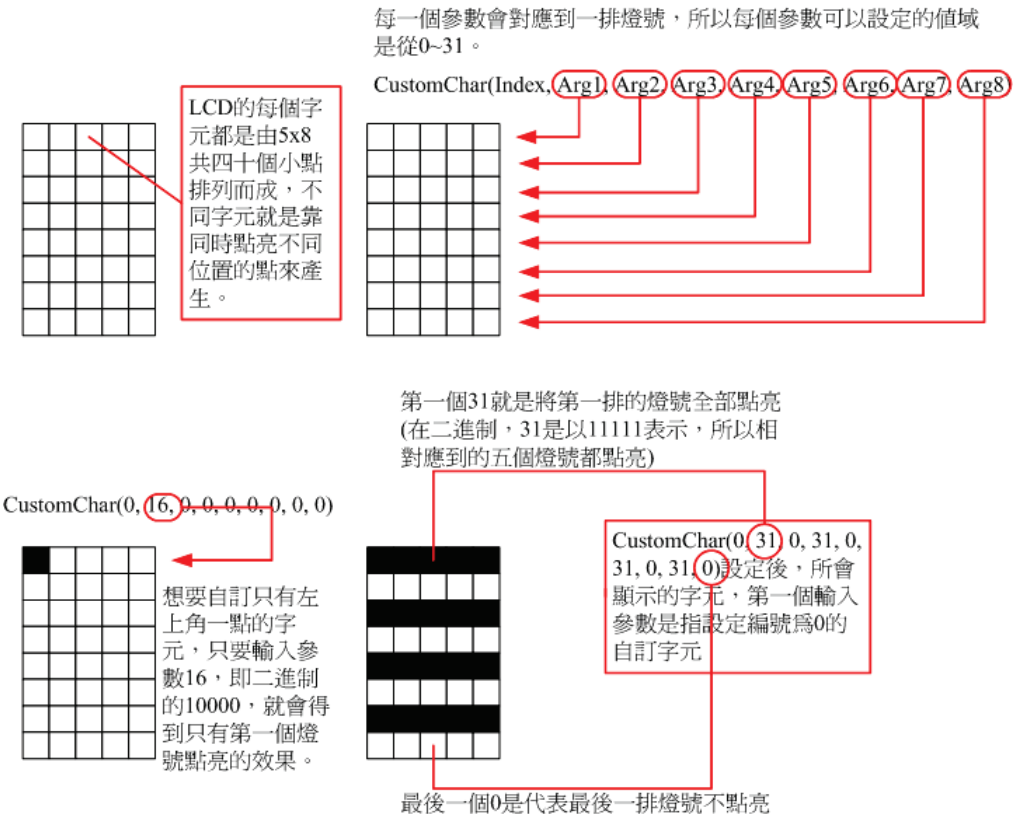
顯示字元相關指令	
CmdBUS： **Display(*Parameter*)** **I2C：** **ShowByte：** **MID+105+CheckSum1+0** **+Parameter+CheckSum2+Dummy** **ShowShort：** **MID+106+CheckSum1+0** **+Parameter+CheckSum2+Dummy** **ShowWord：** **MID+107+CheckSum1+0+arameter_L** **Parameter_H+CheckSum2+Dummy** **ShowDWord：** **MID+136+CheckSum1+0+Parameter_4** **Parameter_3+Parameter_2** **Parameter_1+CheckSum2+Dummy** **ShowInteger：** **MID+108+CheckSum1+0+Parameter_L** **Parameter_H+CheckSum2+Dummy** **ShowLong：** **MID+137+CheckSum1+0+Parameter_4** **Parameter_3+Parameter_2** **Parameter_1+CheckSum2+Dummy** **ShowString：** **MID+113+CheckSum1+DataSize** **+StringSize+String** **+CheckSum2+Dummy** **註：於字串傳輸時，** **StringSize = 字串字數** **DataSize = StringSize + 3** **CheckSum2 = 255 – (所有字元 ASⅡ碼)** **EX：傳輸字串為 Hi，則** **StringSize = 2 DataSize = 5,** **CheckSum2 = 255 – 72 -105**	根據 ***Parameter*** 參數形式，如果是 **String** 會直接顯示字串，如果是浮點數，會用科學記號表示，其它數值則以十進制顯示

CmdBUS： **DisplayBin(*Value*)** **I2C：** **ShowBinByte：** **MID+109+CheckSum1+Value** **+CheckSum2+Dummy** **ShowBinWord：** **MID+110+CheckSum1+Value_L** **Value_H+CheckSum2+Dummy** **ShowBinDWord：** **MID+138+CheckSum1+Value_4** **Value_3+Value_2** **Value_1+CheckSum2+Dummy**	將 ***Value*** 以二進制顯示，***Value*** 請輸入整數值
CmdBUS： **DisplayChar(*Chr*, …)** **I2C：** **MID+115+CheckSum1+Chr+....+** **CheckSum2+Dummy**	顯示 ***Chr*** 所設定的字元，***Chr*** 請輸入 0~255 之間的整數值，也可以輸入 0~7 顯示所代表的自訂字元，可重複輸入多項字元與參數，輸入值將以 ASCII 碼代表值顯示，請參照附錄 3
DisplayFloat(*FloatNum*, *Digits*)	以 ***Digits*** 設定的有效位數，將 ***FloatNum*** 以科學記號形式顯示，***Digits*** 請輸入 0~255 的整數值，***FloatNum*** 請輸入浮點數值
CmdBUS： **DisplayHex(*Value*)** **I2C：** **ShowHexByte：** **MID+111+CheckSum1+Value** **+CheckSum2+Dummy** **ShowHexWord：** **MID+112+CheckSum1+Value_L** **Value_H+CheckSum2+Dummy** **ShowHexLong：** **MID+139+CheckSum1+Value_4** **Value_3+Value_2+** **Value_1+CheckSum2+Dummy**	將 ***Value*** 以十六進制顯示，***Value*** 請輸入整數值

DisplayLeft(*Value*, *Num*)	依 ***Num*** 所設定的寬度，請輸入 0~255 之間的整數值，靠左以十進位顯示 ***Value***；如果輸入超過寬度的數值，會自動調整為符合的寬度，***Value*** 請輸入非字串的數值
DisplayReal(*Real*, *Digits*)	以 ***Digits*** 所設定的有效位數，將 ***Real*** 以實數形式顯示，***Real*** 請輸入浮點數，***Digits*** 請輸入 0~255 之間的整數值
DisplayRight (*Value*, *Num*)	依 ***Num*** 所設定的寬度，請輸入 0~255 之間的整數值，靠右以十進位顯示 ***Value***；如果輸入超過寬度的數值，會自動調整為符合的寬度，***Value*** 請輸入非字串的數值
各種設定相關指令	
CmdBUS： **BacklightOff()** **I2C：** **MID+121+CheckSum1+Dummy**	關閉背光
CmdBUS： **BacklightOn(*Time*)** **I2C：** **MID+120+CheckSum1+Dummy**	以 ***Time*** 值設定背光要點亮的時間，若設為 0 則恆亮，***Time*** 請輸入 0~255 之間的整數值
CmdBUS： **CursorBlinkOff()** **I2C：** **MID+125+CheckSum1+Dummy**	停止游標閃爍
CmdBUS： **CursorBlinkOn()** **I2C：** **MID+124+CheckSum1+Dummy**	讓游標開始閃爍
CmdBUS： **CursorOff()** **I2C：** **MID+123+CheckSum1+Dummy**	關閉游標顯示
CmdBUS： **CursorOn()** **I2C：** **MID+122+CheckSum1+Dummy**	讓游標顯示於螢幕

CustomChar(*Index*, *Arg1*, *Arg2*, *Arg3*, *Arg4*, *Arg5*, *Arg6*, *Arg7*, *Arg8*)	由 ***Index*** 值選擇所要設定的自訂字元編號，請輸入 0~7 之間的整數值，***Arg1*~*Arg8*** 則分別表示，自訂字元各列要顯示的值，將以二進制的方式點亮各列的顯示，請輸入 0~255 之間的整數值 *1
CmdBUS： **DisplayOff()** **I2C：** **MID+133+CheckSum1+Dummy**	關閉螢幕顯示
CmdBUS： **DisplayOn()** **I2C：** **MID+132+CheckSum1+Dummy**	開啟螢幕顯示
GetTab(*TabCount*)	取得設定的 **Tab** 值存放於 ***TabCount*** 中，***TabCount*** 會回傳 0~255 之間的整數值
CmdBUS： **RotateLeft(*Line*, *Spd*)** **I2C：** **MID+128+CheckSum1+Line +Spd+CheckSum2+Dummy**	將 ***Line*** 所指定的列，各字元不斷向左移動，到最左端的字元會再由右方顯示，移動的速度由 ***Spd*** 值決定，越小則速度越快，***Line*** 請輸入 1~4 之間的整數值，***Spd*** 請輸入 0~255 之間的整數值
CmdBUS： **RotateOff()** **I2C：** **MID+130+CheckSum1+Dummy**	停止各行自動向左右移動的動作
CmdBUS： **RotateRight (*Line*, *Spd*)** **I2C：** **MID+129+CheckSum1+Line +Spd+CheckSum2+Dummy**	將 ***Line*** 所指定的列，各字元不斷向右移動，到最右端的字元會再由左方顯示，移動的速度由 ***Spd*** 值決定，越小則速度越快，***Line*** 請輸入 1~4 之間的整數值，***Spd*** 請輸入 0~255 之間的整數值
CmdBUS： **SetBacklight (*Arg*)** **I2C：** **MID+118+CheckSum1+Arg +CheckSum2+Dummy**	以 ***Arg*** 設定背光亮度，***Arg*** 請輸入 0~255 之間的整數值
SetTab(*TabCount*)	以 ***TabCount*** 值設定每次執行 **Tab** 所要移動的行數，***TabCount*** 請輸入 0~255 之間的整數值

*1　LCD 字元的顯示與自訂字元的顯示可參考下面的範例:



範例程式:

```
Peripheral myLCD As LCD4x20A @ 0        '    設定模組編號為 0

Sub Main()                              '    主程式
    myLCD.DisplayOn()                   '    開啟螢幕顯示
    myLCD.SetBacklight(255)             '    設定 LCD 亮度為最大
    myLCD.Backlighton(0)                '    開啟 LCD 背光恆亮
    myLCD.Display("Hello World!")       '    在螢幕上顯示  "Hello World!"
    Pause 3000
    myLCD.RotateRight(1, 10)            '    讓"Hello Word!"由左向右移動
    Pause 5000
    myLCD.RotateOff()                   '    停止"Hello Word!"的移動
    myLCD.Clear()                       '    清除螢幕上的所有顯示

'   設定編號為 0 的自訂字元，在第一，三，五，七列顯示橫線
'   31 在二進制是五個一，所以設定 31 的列為燈號全亮
'   顯示這個字元時，畫面上所顯示的將會是四條橫線
    myLCD.Customchar(0, 31, 0, 31, 0, 31, 0, 31,0)
'   在螢幕上顯示八個編號為 0 的自訂字元
    myLCD.DisplayChar(0, 0, 0, 0, 0, 0, 0, 0)
    Pause 2000
'   設定編號為 1 的自訂字元，在第一，三，五行顯示豎線
'   21 在二進制是 10101，所以 1,3,5 行的燈號會亮起
'   顯示這個字元時，畫面上所顯示的將會是三條豎線
    myLCD.Customchar(1, 21, 21, 21, 21, 21, 21, 21,21)
'   在螢幕上增加顯示八個編號為 1 的自訂字元
    myLCD.DisplayChar(1, 1, 1, 1, 1, 1, 1, 1)
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

3. ASCII 表:

- American Standard Code for Information Interchange，美國信息互換標準代碼，是基於拉丁字母的一套電腦編碼系統，此處的 ASCII 碼是根據標準編碼再做調整得到，由使用者輸入的編號轉換為相對應的字元。
- 左方欄位表示的是二進制的低四位元，上方欄位表示的是二進制的高四位元。欄位中的 L 代表 0，H 代表 1，LLLL 就是二進制的 0000，十進制即為 0。
- 最左上方的表格代表，輸入 ASII 碼 0 時，LCD 會顯示的字元圖案。例: CG RAM1 是會輸出使用者所設定的自訂字元 1，往下依序遞增，往右一行所代表的 ASCII 碼輸入值為 16，依此類推，最右下的字元是輸入 255 所得到的顯示畫面。