Smarter. Greener. Together.

台達電子工業股份有限公司

機電事業群

33068 桃園市桃園區興隆路 18 號

TEL: 886-3-3626301

FAX: 886-3-3716301

DELTA_IA-HMI_LUA_UM_TC_20211208.pdf

台達 Lua 指令操作手冊

台達 Lua 指令操作手冊

www.deltaww.com

Smarter. Greener. Together.

Lua 指令操作手冊

1. Lua 簡介

Lua 為一個輕量級的程式語言，由巴西 Computer Graphics Technology Group (Tecgraf) 所創造，其指令碼語言簡單、易學、易用，能讓程式設計師快速完成程式的編寫工作，台達 HMI 支援 Lua 程式語言，可以讓使用者在設計畫面上更加有彈性。

■ Lua 編輯視窗

點選 [專案] → [程序] → [Main]，即可開始撰寫 Lua 程序，如圖 1.1。

撰寫 Lua 程序的過程中，可搭配程序範例助手，以快速認識指令，也可將滑鼠指標移至指令上，顯示該指令的相關說明，請參考圖 1.2。

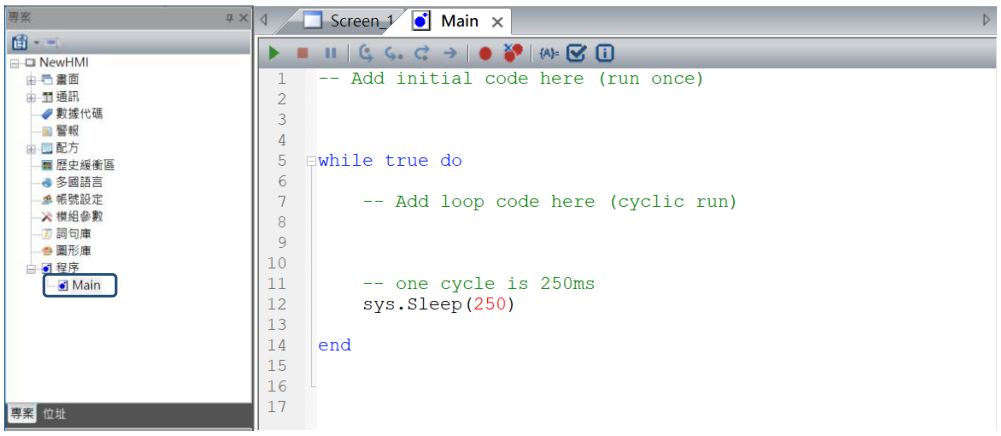


圖 1.1 Lua 程式撰寫介面

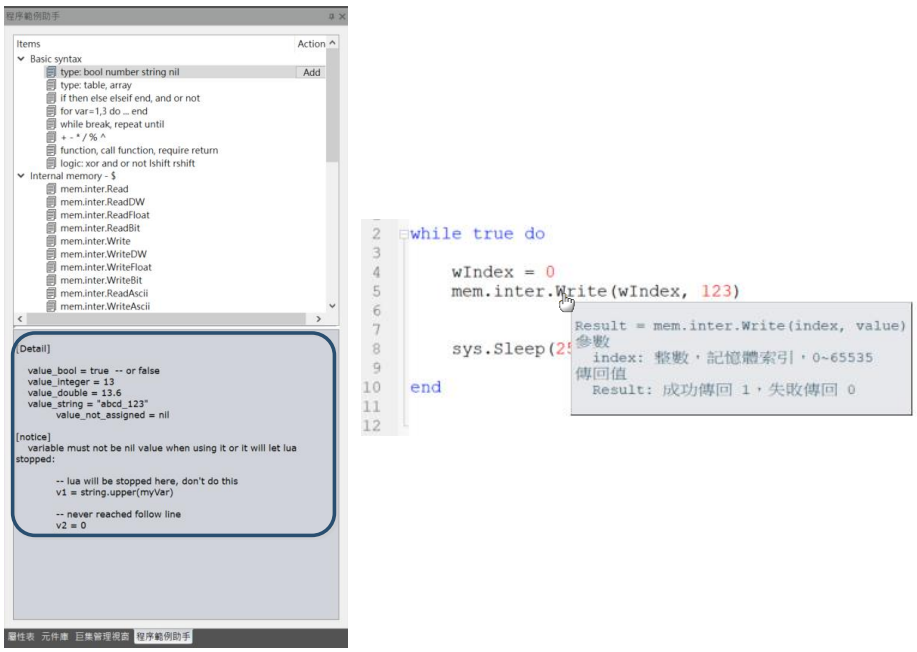


圖 1.2 Lua 程序助手及說明

使用者也可以至 Lua 視窗中的 Action 欄位，按下 **Add**，即可快速的將簡易的 Lua 範例指令加入至程序中。

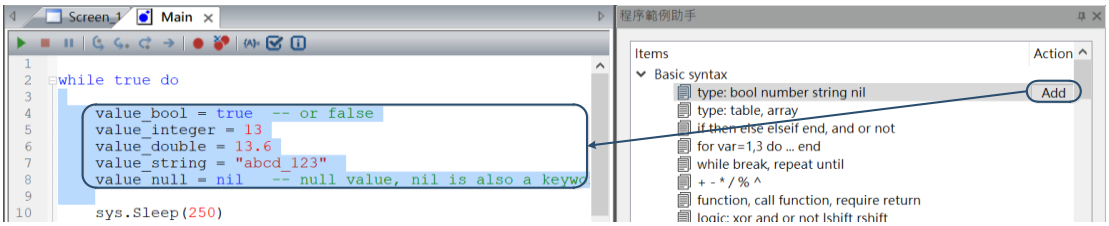


圖 1.3 Lua 指令快速輸入

2. Lua 程式基本語法

以下為 Lua 程式的基本語法摘要。

<table>
<tr><th>項目</th><th>描述</th></tr>
<tr><td>區塊 (chunk 或 block)</td><td>■ Lua 程式的一段陳述稱為 chunk，它可以是指一條指令，也可以是一群指令，執行結果都是一樣的。
例如：
==========
a=1
mem.inter.WriteBit(1,0,a)
或者
a=1 mem.inter.WriteBit(1,0,a)</td></tr>
<tr><td>程式註解</td><td>Lua 註解分為兩種：
■ 單行註解：使用開頭的兩個 dash (--)作為單行註解符號(一直到行尾)，例如：
-- Lua
■ 多行註解：使用兩個 dash 與雙中括號 --[[開頭, 一直到下一個雙中括號為止，所包含的區域編譯會略過不予處理：
--[[這是
多行
註解]]</td></tr>
<tr><td>識別字與保留字</td><td>■ 在 Lua 中，變數和函式名稱以識別字來表示，使用英文字母、數字以及底線為組合，但字串的第一個字元不可以為數字，並且在設定變數過程中，以下這 22 個字為保留字不可被使用。
<table>
<tr><td>local</td><td>nil</td><td>not</td><td>or</td></tr>
<tr><td>and</td><td>if</td><td>then</td><td>else</td></tr>
<tr><td>elseif</td><td>for</td><td>end</td><td>in</td></tr>
<tr><td>do</td><td>while</td><td>until</td><td>repeat</td></tr>
<tr><td>break</td><td>true</td><td>false</td><td>function</td></tr>
<tr><td>return</td><td>goto</td><td>-</td><td>-</td></tr>
</table>
注意：Lua 有區分大小寫，所以雖然 goto 為保留字，但是 GOTO 可以為變數。</td></tr>
</table>

<table>
<tr><th>項目</th><th>描述</th></tr>
<tr><td>全域變數與區域變數</td><td>■ 只要不是用 local 宣告的變數都是全域變數，例如：
==========
A=100
■ 區域變數是以 local 宣告的變數，其作用域 (scope) 僅限於所在之區塊 (block) 內，例如在函式、分歧控制、迴圈或 do end 結構中。
==========
if a=1 then
local A=100
end</td></tr>
<tr><td>多重指派</td><td>■ 可以在一個陳述中同時為多個變數賦值，各變數以逗號隔開，例如：
==========
a,b,c=1,2,3
mem.inter.Write(1,b)</td></tr>
<tr><td>資料型態</td><td>■ 變數是無型態的，宣告時不需要指定資料型態，但變數的值有型態，分別為：nil、boolean、number、string。
表 2.1 Lua 資料類型
<table><tr><th>基本類型</th><th>說明</th></tr><tr><td>nil</td><td>無效值</td></tr><tr><td>boolean</td><td>False 和 true</td></tr><tr><td>number</td><td>雙精度類型的實浮點數</td></tr><tr><td>string</td><td>字符串；由一對雙引號或單引號來表示</td></tr></table></td></tr>
</table>

3. Lua 指令列表

指令	指令運算式	說明
Basic syntax (基本語法)	Type: bool number string nil	資料型態
	Type: table, array	矩陣運算
	if then else elseif end, and or not	比較
	for var=1,3 do … end	for 迴圈
	while break, repeat until	while、repeat 迴圈
	+-*/%^	數學運算
	function, call function, require return	流程控制
	Logic: xor and or not lshift rshift	邏輯運算
Internal memory - $ (內部暫存器($))	mem.inter.Read	讀取內部暫存器數值 (單位：Word)
	mem.inter.ReadDW	讀取內部暫存器數值 (單位：Double Word)
	mem.inter.ReadFloat	讀取內部暫存器數值 (單位：Float)
	mem.inter.ReadBit	讀取內部暫存器位元
	mem.inter.ReadDouble	讀取內部暫存器數值 (單位：雙精度浮點數 64 bits)
	mem.inter.Write	寫入內部暫存器數值 (單位：Word)
	mem.inter.WriteDW	寫入內部暫存器數值 (單位：Double Word)
	mem.inter.WriteFloat	寫入內部暫存器數值 (單位：Float)
	mem.inter.WriteBit	寫入內部暫存器位元
	mem.inter.WriteDouble	寫入內部暫存器數值 (單位：雙精度浮點數 64 bits)
	mem.inter.ReadAscii	讀取內部暫存器字串
	mem.inter.WriteAscii	寫入內部暫存器字串
Static memory - $M (斷電保持內部暫存器($M))	mem.static.Read	讀取斷電保持暫存器數值 (單位：Word)
	mem.static.ReadDW	讀取斷電保持暫存器數值 (單位：Double Word)
	mem.static.ReadFloat	讀取斷電保持暫存器數值 (單位：Float)
	mem.static.ReadBit	讀取斷電保持暫存器位元

指令	指令運算式	說明
	mem.static.ReadDouble	讀取斷電保持暫存器數值 (單位：雙精度浮點數 64 bits)
	mem.static.Write	寫入斷電保持暫存器數值 (單位：Word)
	mem.static.WriteDW	寫入斷電保持暫存器數值 (單位：Double Word)
	mem.static.WriteFloat	寫入斷電保持暫存器數值 (單位：Float)
	mem.static.WriteBit	寫入斷電保持暫存器位元
	mem.static.WriteDouble	寫入斷電保持暫存器數值 (單位：雙精度浮點數 64 bits)
	mem.static.ReadAscii	讀取斷電保持暫存器字串
	mem.static.WriteAscii	寫入斷電保持暫存器字串
External link (外部記憶體位址)	link.Read	讀取外部記憶體位址數值 (單位：Word)
	link.ReadDW	讀取外部記憶體位址數值 (單位：Double Word)
	link.ReadFloat	讀取外部記憶體位址數值 (單位：Float)
	link.ReadBit	讀取外部記憶體位址位元
	link.ReadAscii	讀取外部記憶體位址字串
	link.ReadDouble	讀取外部記憶體位址數值 (單位：雙精度浮點數 64 bits)
	link.Write	寫入外部記憶體位址數值 (單位：Word)
	link.WriteDW	寫入外部記憶體位址數值 (單位：Double Word)
	link.WriteFloat	寫入外部記憶體位址數值 (單位：Float)
	link.WriteBit	寫入外部記憶體位址位元
	link.WriteAscii	寫入外部記憶體位址字串
	link.WriteDouble	寫入外部記憶體位址數值 (單位：雙精度浮點數 64 bits)
	link.CopyFromInter	從人機內部暫存器位址複製資料至外部記憶體位址
	link.CopyToInter	從外部記憶體位址複製資料至人機內部暫存器位址
	link.CopyArray	從人機內部/外部記憶體位址複製資料至人機內部/外部記憶體位址

指令	指令運算式	說明
	link.DownloadPLC	使用 COM 的通訊方式透過人機下載 isp 或 dvp 程序至 PLC 中
	link.DownloadEthPLC	使用網路的通訊方式透過人機下載 isp 或 dvp 程序至 PLC 中
	link.WritePasswordPLC	下載系統密碼至台達 PLC
	link.SetDefaultStationNo	設定人機欲通訊的 PLC 預設站號
	link.SetHMIStationNo	設定 HMI Slave 站號(Modbus Slave)
	link.CODESYSAppDownload	使用網路的通訊方式透過人機下載 CODESYS 程序至 PLC 中
	link.CODESYSAppUpload	使用網路的通訊方式上載 CODESYS 程序至 HMI 的 USB 中
File (文件讀取/寫入/匯出/清單)	file.Open	建立/開啟檔案
	file.Read	讀取檔案資料
	file.ReadLine	讀取檔案(以一行為單位)
	file.Write	寫入檔案
	file.Length	讀取檔案長度
	file.GetLineCount	讀取檔案總行數
	file.Seek	設定指標
	file.GetPos	取得當前指標位置
	file.GetError	檢查文件
	file.Close	關閉檔案
	file.List	取得存於人機內部的文件清單
	file.Export	匯出檔案
	file.Delete	刪除檔案
	file.DeleteDir	刪除目錄
	file.ToPDF	將檔案轉為 PDF
	file.ToPrinter	列印文件
	file.ListExternal	取得存於外部裝置的文件清單
	file.Exist	確認檔案是否存在
	file.PDFToPrinter	列印 PDF 文件
	file.Copy	複製檔案
	file.Move	移動檔案
FileSlot (檔案讀取/寫入/匯出)	fileslot.Read	fileslot 檔案讀取
	fileslot.Write	fileslot 檔案寫入
	fileslot.ReadValue	讀取 fileslot 數值內容
	fileslot.WriteValue	寫入數值至 fileslot
	fileslot.GetLength	取得 fileslot 內容長度
	fileslot.Remove	移除 fileslot
	fileslot.Import	fileslot 檔案匯入

指令	指令運算式	說明
	fileslot.Export	fileslot 檔案匯出
	fileslot.SetName	設定 fileslot 檔案名稱
	fileslot.GetName	取得 fileslot 檔案名稱
	fileslot.GetID	取得 fileslot 檔案 ID
FTP 傳輸功能	ftpc.Download	FTP 下載
	ftpc.Upload	FTP 上傳
Math (數學運算)	math.abs	取得數字的絕對值
	math.exp	取得以 e 為底數的指數函數數值
	math.log	取得對數數值
	math.sin	取得正弦數值
	math.sinh	取得雙曲線正弦數值
	math.cos	取得餘弦數值
	math.cosh	取得雙曲線餘弦數值
	math.tan	取得正切數值
	math.tanh	取得雙曲線正切數值
	math.asin	取得反正弦數值
	math.acos	取得反餘弦數值
	math.atan	取得反正切數值
	math.atan2	取得反正切數值 (兩個參數)
	math.deg	取得弧度對應的角度
	math.rad	取得角度對應的弧度
	math.min	取得最小值
	math.max	取得最大值
	math.modf	分割數值為整數和小數
	math.pi	圓周率 π
	math.pow	取得次方數值
	math.randomseed	亂數種子
	math.random	取得隨機數
	math.sqrt	取得開根號數值
Recipe (配方)	recipe.GetCurRcpNoIndex	取得當前配方組別索引
	recipe.GetCurRcpGIndex	取得當前配方群組索引
	recipe.GetRcpWord	取得指定配方位址的數值 (Word)
	recipe.GetRcpDWord	取得指定配方位址的數值 (Double Word)
	recipe.GetRcpFloat	取得指定配方位址的數值 (Float)
	recipe.GetCurEnRcpNoName	取得當前加強型配方組別索引名稱
	recipe.GetCurEnRcpGName	取得當前加強型配方群組索引名稱
	recipe.GetCurEnRcpNoIndex	取得當前加強型配方組別索引
	recipe.GetCurEnRcpGIndex	取得當前加強型配方群組索引

指令	指令運算式	說明
	recipe.GetEnRcpWord	取得指定加強型配方位址的數值 (Word)
	recipe.GetEnRcpDWord	取得指定加強型配方位址的數值 (Double Word)
	recipe.GetEnRcpFloat	取得指定加強型配方位址的數值 (Float)
	recipe.GetEnRcpAscii	取得指定加強型配方位址的字串
	recipe.SetRcpWord	設定參數至配方位址 (Word)
	recipe.SetRcpDWord	設定參數至配方位址 (Double Word)
	recipe.SetRcpFloat	設定參數至配方位址 (Float)
	recipe.SetCurEnRcpNoName	設定加強型配方組別名稱
	recipe.SetCurEnRcpGName	設定加強型配方群組名稱
	recipe.SetEnRcpWord	設定參數至加強型配方位址 (Word)
	recipe.SetEnRcpDWord	設定參數至加強型配方位址 (Double Word)
	recipe.SetEnRcpFloat	設定參數至加強型配方位址 (Float)
	recipe.SetEnRcpAscii	設定字串至加強型配方位址
	recipe.ChangeRcpNoIndex	更改配方組別索引
	recipe.ChangeRcpGIndex	更改配方群組索引
	recipe.ChangeEnRcpNoIndex	更改加強型配方組別索引
	recipe.ChangeEnRcpGIndex	更改加強型配方群組索引
	recipe.SetEnRcpDouble	設定參數至加強型配方位址 (雙精度浮點數)
	recipe.GetEnRcpDouble	取得指定加強型配方位址的數值 (雙精度浮點數)
Screen (螢幕控制)	screen.Open	開啟指定畫面
	screen.CloseSub	關閉指定畫面
	screen.IsOpened	確認指定畫面是否開啟
	screen.Capture	擷取畫面圖片至外部儲存裝置
String (字串運算)	string.len	計算字串長度
	string.format	字串格式化
	string.split	將字串分開
	string.find	尋找字串位置
	string.sub	尋找字串
	string.rep	重複字串
	string.trim	將字串去除前後空白
	string.lower	將字串轉換成小寫
	string.upper	將字串轉換成大寫
	string.reverse	將字串反轉

指令	指令運算式	說明
	string.byte	將字串轉換為十進位數值
	string.char	將十進位數值轉換為字串
	string.gsub	以字串取代指定的字串
	string.gmatch	於字串中尋找模式字串匹配的部分，找到匹配的參數後回傳 註：需搭配 for 迴圈。
	string.match	於字串中尋找模式字串匹配的部分，找到匹配的參數後回傳。 註：string.match 和 string.gmatch 的差異為 string.gmatch 會回傳所有匹配的字串，而 string.match 只會回傳第一組匹配的字串。
System library (系統參數)	sys.Sleep	系統延遲
	sys.GetTick	取得截至目前為止人機的總開機時間
	sys.GetInterParam	取得人機內部參數數值
	sys.BuzzerOn	開啟蜂鳴器
	sys.GetDate	取得當前時間
	sys.GetDateString	取得當前時間 (單位：字串)
	sys.GetDays	取得 1970/01/01 到設定日期所經過的天數
	sys.GetSecs	取得 00:00:00 到設定時間所經過的秒數
	sys.GetTime	取得系統時間
	sys.ToDate	取得 1970/01/01 經過所設定天數後的日期
	sys.ToTime	取得 00:00:00 經過所設定秒數後的時間
	sys.GetDiskSpace	取得外部儲存裝置空間
Serial Port communication (COM 自由通訊)	com.Open	開啟 com 接口通訊
	com.ReadChars	從指定通訊埠讀取字元
	com.WriteChars	於指定通訊埠寫入字元
	com.ClearBuffer	清除緩衝區資料
	com.StationCheck	透過選擇通訊埠以及站號來確認站號通訊是否成功
	com.Close	關閉通訊埠
	com.CheckAlive	透過選擇通訊參數來確認通訊是否成功
	com.StationOn	站號啟動
	com.StationOff	站號關閉

指令	指令運算式	說明
	com.GetStatus	取得 com 接口狀態
TCP communication (TCP 自由通訊)	tcp.Open	開啟 TCP 網路通訊
	tcp.Read	讀取字元(TCP)
	tcp.Write	寫入字元(TCP)
	tcp.Close	關閉連線(TCP)
	tcp.GetMaxCount	取得最大連線數量(TCP)
	tcp.GetRunCount	取得正在運行的 socket 數量(TCP)
	tcp.GetStatus	確認 socket 的通訊狀況(TCP)
UDP communication (UDP 自由通訊)	udp.Open	開啟 UDP 網路通訊
	udp.Read	讀取字元(UDP)
	udp.Write	寫入字元(UDP)
	udp.Close	關閉連線(UDP)
	udp.GetMaxCount	取得最大連線數量(UDP)
	udp.GeRunCount	取得正在運行的 socket 數量(UDP)
	udp.GetStatus	確認 socket 的通訊狀況(UDP)
Text encoding (編碼格式變更)	text.GbkToUtf8	將 GBK 格式轉換為 UTF-8
Utility (CRC 運算)	util.Crc16Modbus	計算 crc 數值
Convert (浮點數轉換)	convert.IntToFloat	整數格式轉換成浮點數格式
	convert.ToNum	字串轉換為 64 位元浮點數
Account (權限密碼設定)	account.Add	新增權限帳號
	account.Delete	刪除權限帳號
	account.ChangeName	更改權限帳號名稱
	account.ChangePassword	更改權限密碼
	account.ChangeLevel	更改權限帳號等級
	account.GetPassword	取得使用者的密碼
	account.GetLevel	取得權限等級
	account.GetCurrentLogin	取得當前登入帳號
	account.IsExist	確認帳號是否存在
	account.Login	登入權限
	account.ResetLockStatus	解除已鎖定的帳戶
	account.ChangeUserExpiredDays	更改帳號期限
	account.ChangePwdExpiredDays	更改密碼期限
	account.GetStatus	取得帳戶狀態
	account.GetLockedList	取得已鎖定的帳戶之清單
Mail (信件功能)	mail.Status	信件功能狀態
	mail.Send	傳送信件
	mail.SendFile	傳送信件 (含檔案)

指令	指令運算式	說明
	mail.SendAlarm	傳送信件 (含警報)
	mail.SendHistory	傳送信件 (含歷史資料)
Draw (繪圖功能)	draw.Point	繪圖(點)
	draw.Line	繪圖(線)
	draw.Rect	繪圖(長方形)
	draw.Ellipse	繪圖(橢圓形)
	draw.Clear	清除繪圖
	draw.SetAntialiasing	啟用/取消反鋸齒功能

4. Lua 指令詳細說明

Lua 指令包括：

- Basic syntax (基本語法)
- Internal memory - $ (內部暫存器($))
- Static memory - $M (斷電保持內部暫存器($M))
- External link (外部連接暫存器)
- File read/write/export/delete/pdf/print (文件讀取/寫入/匯出/清單)
- fileSlot (檔案存取)
- FTP Client (FTP 傳輸功能)
- Math (數學運算)
- Recipe (配方)
- Screen (螢幕控制)
- String (字串運算)
- System library (系統參數)
- Serial Port communication (COM 自由通訊)
- TCP communication (TCP 自由通訊)
- UDP communication (UDP 自由通訊)
- Text encoding (編碼格式變更)
- Utility (CRC 運算)
- Convert (浮點數轉換)
- Account (權限密碼設定)
- Mail (信件功能)
- Draw (繪圖功能)

以下章節將依序介紹 Lua 指令種類。

4.1 Basic syntax (基本語法)

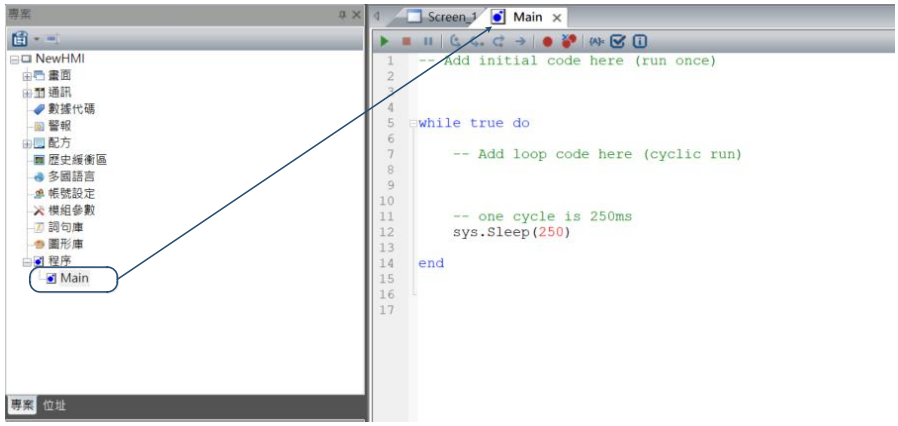
基本語法可供使用者進行矩陣的運算、比較、for 迴圈、while 迴圈、簡單的數學運算，子程式的製作、邏輯運算，其中指令包含：

指令	指令運算式	說明
Basic syntax (基本語法)	Type: bool number string nil	資料型態
	Type: table, array	矩陣運算
	if then else elseif end, and or not	比較
	for var=1,3 do … end	for 迴圈
	while break, repeat until	while、repeat 迴圈
	+-*/%^	數學運算
	function, call function, require return	流程控制
	logic: xor and or not lshift rshift	邏輯運算

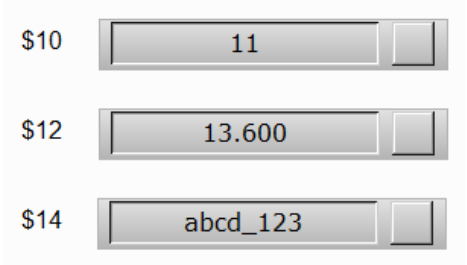
以下章節將一一詳細說明。

4.1.1 Type: bool, number, string, nil (資料型態)

變數是無型態的，宣告時不需要指定資料型態，但變數的值有型態，Lua 提供的變數種類分為布林表示式、整數、小數、字串，使用範例如下：

範例	
開啟 Lua	■ 點選 [專案] → [程序] → [Main]，顯示 Lua 編輯視窗。 

建立
Lua 程序
	■ 設定 value_bool = true，value_integer = 11，value_double = 13.6，value_string = "abcd_123"。
	當 value_bool 為 true 時，以 value_integer 為長度將 value_integer(整數)寫入至$10，並以 value_double 為長度將 value_double(小數)寫入至$12，再以 value_ string 為長度將 value_ string (字串)寫入至$14。
	
	while true do
	value_bool=true
	value_integer=11
	value_double = 13.6
	value_string = "abcd_123"
	if value_bool==true then
	mem.inter.Write(10,value_integer,string.len(value_integer))
	mem.inter.WriteFloat(12,value_double,string.len(value_double))
	mem.inter.WriteAscii(14,value_string,string.len(value_string))
	end
	end

範例	
建立數值輸入元件及文數字輸入元件	■ 建立數值輸入元件，寫入記憶體位址分別為$10、$12，數值單位為 Word、Double Word，數值格式分別為 Unsigned Decimal 和 Floating，參數設定如下：   ■ 建立文數字輸入元件，寫入記憶體位址為$14，字串長度為 10，設定如下： 
執行結果	■ 完成 Lua 程式的編寫和元件的建立後，請下載專案至人機。 ■ 記憶體位址$10、$12、$14 分別顯示對應的結果： 

除了以上這幾種變數定義 (布林表示式、整數、小數、字串)，其他未定義的變數在 Lua 中是不允許的。若使用未定義的變數將會導致 Lua 程式的終止，以下為簡易的範例說明。

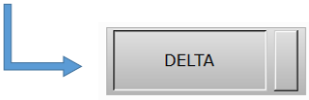
範例

- 如下圖所示，使用者欲取得 Delta 的大寫表示為 v1，並將 v1 字串寫入$100。

```
==========================================================
    while true do
        v1=string.upper(Delta)
        mem.inter.WriteAscii(100,v1,string.len(v1))
    end
```

- 由於 Delta 為一個為未定義之變數(非字串、非常數)，所以 Lua 程序到第二行會終止，並在畫面顯示「Lua 執行錯誤」的錯誤訊息，而沒有執行到寫入的動作。
- 正確語法應為：

```
==========================================================
    while true do
        v1=string.upper("Delta")
        mem.inter.WriteAscii(100,v1,string.len(v1))
    end
```



- 將 Delta 加上" "後，Delta 為一個字串，v1 便為 Delta 的大寫表示，最後並可將 v1 字串寫入$100。

4.1.2 Type: table, array (矩陣運算)

此指令可供使用者進行矩陣的運算以及設計，以下將詳細介紹。

下表為陣列指令表。

■ { }：定義陣列

指令名稱	t = { }
指令運算式	t = **{**var1, var2, var3, var4, var5**}**
參數定義	var1、var2、…：元素

範例	t= { 11, 22, 33, "s1", "s2" }
	t [1] = 111;
	t [2] = 222;
	t [3] = 333;

範例說明	建立陣列 t= { 11, 22, 33, "s1", "s2" }後，將 t [1]從 11 被改為 111 ; t [2]從 22 被改為 222 ; t [3]從 33 被改為 333。
回傳值	無回傳值

■ table.count：取得陣列內元素個數

指令名稱	table.count
指令運算式	Count = **table.count**(myTable)
參數定義	myTable：表格
範例	t = {11, 22, 33} count = table.count(t) t ["a1"] = 10 count = table.count(t)
範例說明	建立陣列 t = {11, 22, 33}，t 陣列中涵蓋 3 個元素。 →Count = 3。 加入 t ["a1"] = 10，此時 t 陣列中涵蓋 4 個元素。 →Count = 4。
回傳值	Count = 整數；表格中的項目個數

■ table.insert：插入元素至陣列中

指令名稱	table.insert
指令運算式	**table.insert**(myTable, [pos,]value)
參數定義	myTable：表格 pos：插入的指定位置，可省略 value：基本型別
範例	t1 = {1, 3, "four"} table.insert(t1, 2, "two") t2 = {1, 3, "four"} table.insert(t2, "five")
範例說明	將字串 two 插入至 t 陣列中指定的第二個位置。 t1 = {1, "two", 3, "four"} 若無指定位置，則將字串 five 加至 t 陣列最後方。 t2 = {1, 3, "four", "five"}
回傳值	無回傳值

■ table.remove：移除元素

指令名稱	table.remove
指令運算式	**table.remove**(myTable, [pos,])
參數定義	myTable：表格 pos：刪除的指定位置，可省略
範例	t = {1, 4, "three"} table.remove(t, 2) t = {1, 4, "three"} table.remove(t)
範例說明	將陣列 t 指定的第二個位置之參數移除。 t = {1, "three"} 若無指定位置，則將最後一個參數移除。 t2 = {1, 4}
回傳值	無回傳值

■ table.concat：將陣列組成字串

指令名稱	table.concat
指令運算式	valueString = **table.concat**(myTable, sepChar)
參數定義	myTable：表格 sepChar：字串，表格中的參數以此字串連接
範例	t = {1, 2, "three", 4, "five"} str = table.concat(t, ",")
範例說明	將陣列以 ” ,” 作為連接組成字串，str = 1,2,three,4,five。
回傳值	str = 字串；連接後的字串

■ table.sort：排列陣列

指令名稱	table.sort
指令運算式	**table.sort**(myTable, compareFunc)
參數定義	myTable：表格 compareFunc：函式
範例	t = {3, 2, 5, 1, 4} table.sort (t, function(a,b) return a>b end)
範例說明	將陣列中的數值由大排到小並把值放入陣列。 t = {5 ,4 ,3 ,2 ,1}
回傳值	無回傳值

4.1.3 if…then…else…elseif…end, and or not (比較)

此指令可供使用者於 Lua 中設計 if 條件式的迴圈，以下將詳細介紹。

下表為 if…else…end 指令表。

基本語法	指令	運算式	敘述內容
比較	If…then…end	If Var1 == Var2 then -- Do Something A end	如果 Var1 等於 Var2，則執行動作 A。

	If…then…else…end	If Var1 == Var2 then
		--Do Something A
		else
		--Do Something B
		end
			如果 Var1 等於 Var2，則執行動作 A，否則執行動作 B。

	If…then…elseif…else…end	If Var1 > Var2 then
		--Do Something A
		elseIf Var1 < Var3
		--Do Something B
		else
		--Do Something C
		end
			如果 Var1 大於 Var2，則執行動作 A，如果 Var1 小於 Var3，則執行動作 B，否則執行動作 C。

範例 (if…then…elseif…else…end)

建立 Lua 程序 (溫度應變措施)

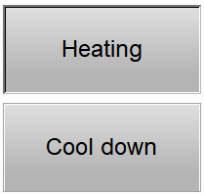
■ 讀取位址$10 為溫度 (Var1)。如果溫度 (Var1)大於 100，啟用冷卻器(位址 1.0)並關閉加熱器，如果溫度 (Var1)小於 20，啟用加熱器(位址 2.0)並關閉冷卻器，否則關閉冷卻器(位址 1.0)和加熱器(位址 2.0)。

```
while true do
    Var1 = mem.inter.Read(10)
        if (Var1 > 100) then
            mem.inter.WriteBit(1, 0, 1)
            mem.inter.WriteBit(2, 0, 0)
        elseif (Var1 < 20) then
            mem.inter.WriteBit(2, 0, 1)
            mem.inter.WriteBit(1, 0, 0)
        else
            mem.inter.WriteBit(1, 0, 0)
            mem.inter.WriteBit(2, 0, 0)
        end
end
```

建立數值顯示、交替型按鈕元件

■ 建立數值顯示元件，讀取記憶體位址為$10。

■ 建立交替型按鈕，寫入記憶體位址為$1.0 和$2.0。

<table>
<tr><th colspan="2">範例 (if…then…elseif…else…end)</th></tr>
<tr><td>執行結果</td><td>
■ 完成 Lua 程式的編寫和元件的建立後，請下載專案至人機。
■ 當溫度大於 100 時，觸發$1.0 啟用冷卻，當溫度小於 20 時，關閉$1.0，觸發$2.0 啟用加熱器，當溫度介於 20 到 110 時，關閉$1.0 和$2.0。
<table>
<tr><th></th><th>情境一</th><th>情境二</th><th>情境三</th></tr>
<tr><td>條件</td><td>$100>0
  </td><td>$100<20
  </td><td>20<$100<110
  </td></tr>
<tr><td>執行結果</td><td>Bit on $1.0
  </td><td>Bit on $2.0
  </td><td>Bit off $1.0 和$2.0
  </td></tr>
</table>
</td></tr>
</table>

4.1.4 for var=1, 3 do … end (for 迴圈)

此指令可供使用者於 Lua 中設計 for 迴圈，以下將詳細介紹。

下表為 for 指令表。

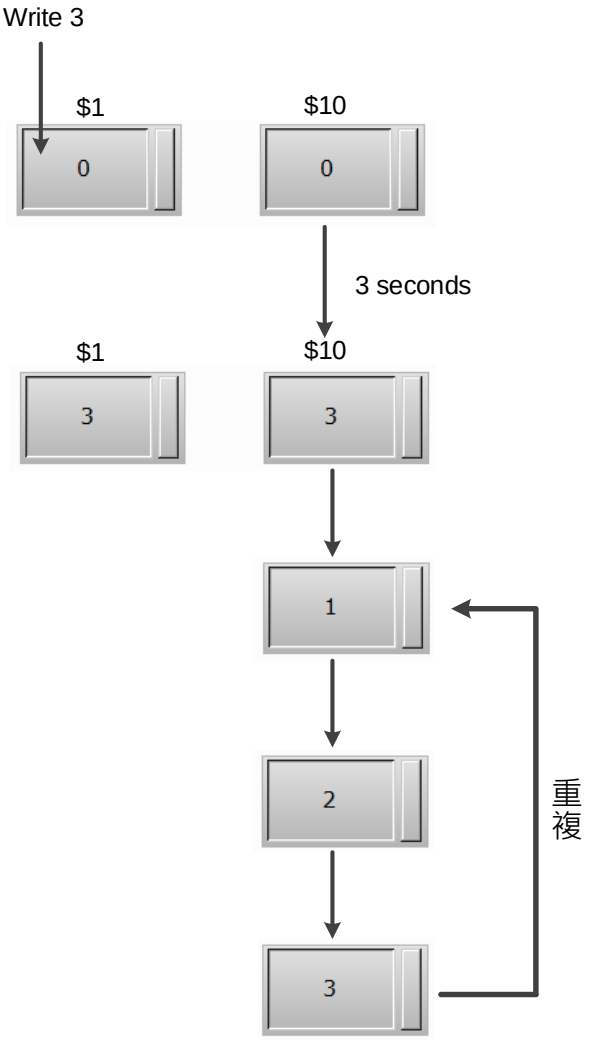
■ for：執行迴圈

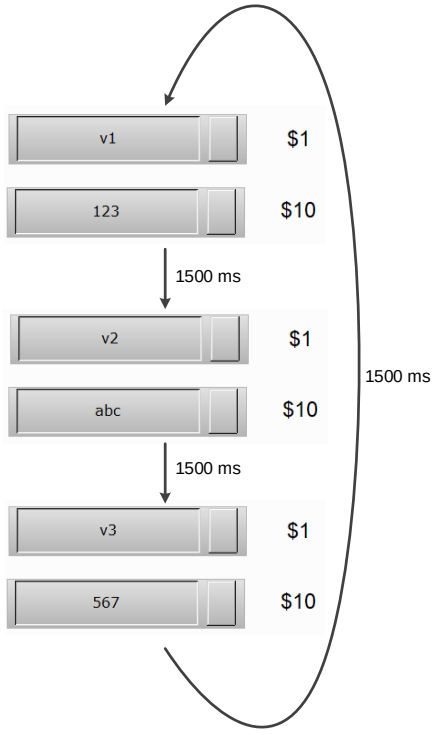
指令名稱	for
指令運算式	**for** [condition1] do --[code block 1] end
範例	for t= 1, 6 do mem.inter.Write(t, 123) end
範例說明	執行 6 次迴圈，分別對內部暫存器$1 ~ $6 寫入數值 123。
回傳值	無回傳值

範例一	

建立 Lua 程序	■ 建立初始值 v 為 0，讀取$1 為數值 v1，進入 for 迴圈，每 1000 ms 執行 v = v + 1 並將 v 寫入記憶體位址$10，共執行 v1 次，最後令 v 為 0，重複以上步驟。
	v = 0
	while true do
	v1=mem.inter.Read(1)
	for i = 1,v1 do
	v = v + 1
	mem.inter.Write(10, v)
	sys.Sleep(1000)
	end
	v=0
	end

建立數值輸入元件	■ 建立數值輸入元件，寫入記憶體位址設為$1。 ■ 建立數值輸入元件，寫入記憶體位址設為$10。

範例一	
執行結果	■ 完成 Lua 程式的編寫和元件的建立後，請下載專案至人機。 ■ 於記憶體位址$1 輸入 3，每 1000 ms，執行 v = v + 1 並寫入記憶體位址$10，共執行 3 次，最後重複 1、2、3。 

<table>
<tr><th colspan="2">範例二</th></tr>
<tr><td>建立 Lua
(1.5 秒移動
1 次)</td><td>■ 建立 t 陣列，每 1500 ms 執行一次，不重複地搜尋 t 陣列的每個參數，並且將參數依序對應到 key 和 value 變數，並寫入$10 和$1。
while true do
t = {v1=123, v2="abc", v3=567}
for key,value in pairs(t) do
mem.inter.WriteAscii(10,key,string.len(key))
mem.inter.WriteAscii(1,value,string.len(value))
sys.Sleep(1500)
end
end</td></tr>
<tr><td>建立文數字輸入元件</td><td>■ 建立文數字輸入元件，寫入記憶體位址設為$1、$10。</td></tr>
<tr><td>執行結果</td><td>■ 完成 Lua 程式的編寫和元件的建立後，請下載專案至人機。
■ 每 1500 ms 執行一次迴圈，依序將 key 寫入$1，value 寫入$10。
 
</td></tr>
</table>

4.1.5 while break, repeat until (while、repeat 迴圈)

此指令可供使用者於 Lua 中設計 while、repeat 的迴圈，以下將詳細介紹。

■ while…end：當…時，執行迴圈

指令名稱	while…end
指令運算式	**while** [condition1] do -- [code block 1] **end**
參數定義	無參數

範例	while i ~= 5 do
	i = i+1
	If i > 100 then
	break
	end
	end

範例說明	當 i 不等於 5 時，執行 i = i+1 迴圈，如果 i > 100 時，透過 break 指令跳出迴圈。
回傳值	無回傳值

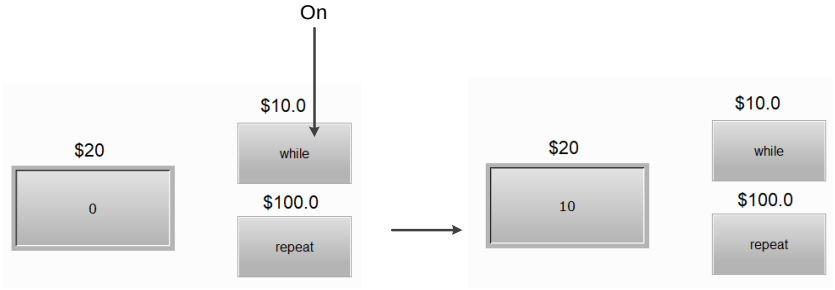
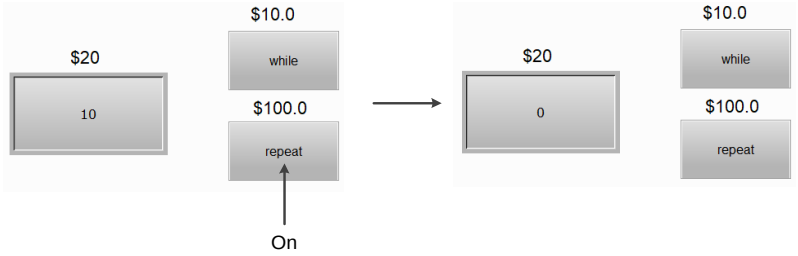
■ repeat…until：重複執行迴圈，直到…

指令名稱	repeat…until
指令運算式	**Repeat** -- [code block 1] **Until** [condition1]
參數定義	無參數
範例	repeat i = i + 1 until(i > 15)
範例說明	重複執行 i = i + 1 直到 i > 15。
回傳值	無回傳值

範例 (while、repeat)	

建立 Lua 程式	■ 程式如下所示：
	while true do
	if (mem.inter.ReadBit(10,0)==1) then
	v1 = mem.inter.Read(20)
	while v1~=10 do
	v1 = v1 + 1
	sys.Sleep(100)
	mem.inter.Write(20, v1)
	if v1>=10 then
	em.inter.WriteBit(10,0,0)
	break
	end
	end
	end
	if (mem.inter.Read(100,0)==1) then
	repeat
	v1 = v1 - 1
	sys.Sleep(100)
	mem.inter.Write(20, v1)
	until(v1 ==0)
	mem.inter.WriteBit(100,0,0)
	end
	end
	■ 如果$10.0 被觸發，讀取$20 為 v1，當 v1 不等於 10 時，執行 v1 = v1 + 1，並將 v1 寫入$20，如果 v1 大於等於 10，即停止執行。
	if (mem.inter.ReadBit(10,0)==1) then
	v1 = mem.inter.Read(20)
	while v1~=10 do
	v1 = v1 + 1
	sys.Sleep(100)
	mem.inter.Write(20, v1)
	if v1>=10 then
	mem.inter.WriteBit(10,0,0)
	break
	end
	end
	end
	■ 如果$100.0 被觸發，重複執行 v1 = v1 - 1，並將 v1 寫到$20，直到 v1 = 0，再關閉$100.0。
	if (mem.inter.Read(100,0)==1) then
	repeat
	v1 = v1 - 1
	sys.Sleep(100)
	mem.inter.Write(20, v1)
	until(v1 ==0)
	mem.inter.WriteBit(100,0,0)
	end

建立數值顯示、交替型按鈕元件	■ 建立數值顯示元件，寫入記憶體位址設為$20。 ■ 建立交替型按鈕元件，寫入記憶體位址設為$10.0 及$100.0。

範例 (while、repeat)	
執行結果	■ 完成 Lua 程式的編寫和元件的建立後，請下載專案至人機。 ■ 觸發$10.0，執行 v1 = v1 + 1，當 v1 大於等於 10 時停止，並關閉 $10.0。  ■ 觸發$100.0，執行 v1 = v1 - 1，直到 v1 = 0，並關閉$100.0。 

4.1.6 +-*/%^ (數學運算)

此指令可供使用者於 Lua 中進行簡易的數學運算，以下將詳細介紹。

下表為+-*/%^指令表。

基本語法種類	指令	運算式	敘述內容
數學運算	+	Var1=Var2+Var3	加法運算
	-	Var1=Var2-Var3	減法運算
	*	Var1=Var2*Var3	乘法運算
	/	Var1=Var2/Var3	除法運算
	%	Var1=Var2%Var3	餘數運算
	^	Var1=Var2^Var3	乘方(次方)運算

4.1.7 function, call function, require return (流程控制)

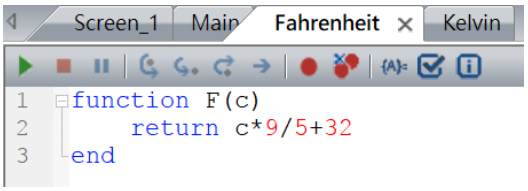
函式(function)可以將 Lua 程式碼區塊包裝起來，並給予命名，透過呼叫的方式使用此函式(function)，讓使用者在程式上規劃上更簡潔及便利，以下將詳細介紹。

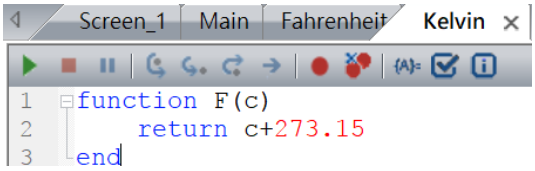
■ function：建立函式

指令名稱	function
指令運算式	**function** [function name 1] -- [code block 1] return [result 1] end
參數定義	無參數
範例	function Add (a, b) return a+b, a*b end
範例說明	建立 Add(a, b)的指令，回傳 a+b 及 a*b 的結果。
回傳值	無回傳值

■ require：載入模組

指令名稱	require
指令運算式	**require** [program name 1]
參數定義	無參數
範例	-- in Prog001 program function Add (a, b) return a+b, a*b end -- in main program require "Prog001" v1, v2 = Add (10, 20)
範例說明	在 Prog001 program 中建立函式 function Add(a, b)，透過 require 指令於主程式載入 Prog001 program，便可以使用 Add(a, b)函式，讓 v1=10+20，v2=10*20，v1=30，v2=200。
回傳值	無回傳值

<table>
<tr><th colspan="2">範例 (function, call function)</th></tr>
<tr><td>建立 Lua 子程式</td><td>■ 於副程式介面中，建立 Lua 子程式，F(c)=c*9/5+32 表示攝氏換成華氏溫度。
 

==========
function F(c)
return c*9/5+32
end
■ 於副程式介面中，建立 Lua 子程式，K(c)=c+273.15 表示攝氏換成凱氏溫度。
 

==========
function K(c)
return c+273.15
end</td></tr>
<tr><td>建立 Lua 主程式</td><td>■ 於主程式中透過 require 指令呼叫子程式檔名，此時便可直接使用 F(c)和 K(c)，執行 F(c)=c*9/5+32 以及 K(c)=c+273.15，並將結果分別寫入至$200 和$300。
while true do
require"Fahrenheit"
require"Kelvin"
c= mem.inter.Read(100)
K_temperature=K(c)
F_temperature=F(c)
mem.inter.Write(200, K_temperature)
mem.inter.Write(300, F_temperature)
end</td></tr>
<tr><td>建立數值輸入元件</td><td>■ 建立數值輸入元件，寫入記憶體位址分別為$100、$200、$300。</td></tr>
</table>

範例 (function, call function)	
執行結果	■ 完成 Lua 程式的編寫和元件的建立後，請下載專案至人機。 ■ 於$100 輸入攝氏溫度 30，則$200、$300 分別顯示凱氏溫度和華氏溫度。 

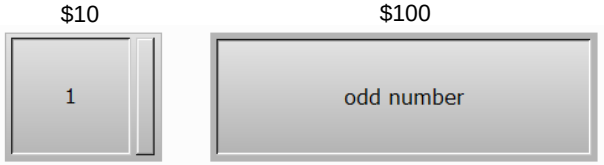
4.1.8 logic: xor and or not lshift rshift (邏輯運算)

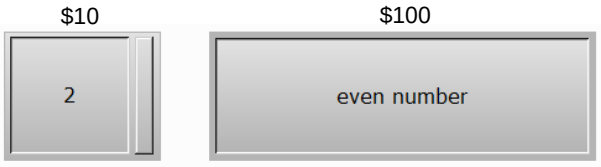
此指令可供使用者進行邏輯運算，以下將詳細介紹。

下表為邏輯運算指令表。

基本語法	指令	運算式範例	敘述內容
邏輯運算	math.bxor	Var1 = math.bxor(0x01, 0x03)	Var1 為 0x01 和 0x03 進行 xor 運算。 -- output: Var1 = 2
	math.band	Var1 = math.band(0x01, 0x03)	Var1 為 0x01 和 0x03 進行 and 運算。 -- output: Var1 = 1
	math.bor	Var1 = math.bor(0x01, 0x03)	Var1 為 0x01 和 0x03 進行 or 運算。 -- output: Var1 = 3
	math.bnot	Var1 = math.bnot(0x01)	Var1 為 0x01 進行 not 運算。 -- output: Var1 = 0xFFFFFFFE
	math.lshift	Var1 = math.lshift(0x01, 2)	將十六進制的 1 轉換為二進制後左移兩位。 -- output: Var1 = 4
	math.rshift	Var1 = math.rshift(0x04, 2)	將十六進制的 4 轉換為二進制後右移兩位。 -- output: Var1 = 1

邏輯運算表				
運算元 1	0	1	0	1
運算元 2	0	0	1	1
XOR	0	1	1	0
AND	0	0	0	1
OR	0	1	1	1

<table>
<tr><th colspan="2">範例 (logic)</th></tr>
<tr><td>建立 Lua 程序
(奇偶數判斷)</td><td>■ 讀取$10 為 c，將 c 和 1 進行 and 邏輯運算(已知 and 運算中只有 1 和 1 運算出 1，也就是說只有奇數與 1 進行 and 運算會為 1，偶數會為 0)，如此，如果變數 g 為 1 時 judge 為字串 odd number，g 為 0 時 judge 為字串 even number，最後將 judge 寫入$100。
==========
while true do
c = mem.inter.Read(10)
g = math.band(1, c)
if g==1 then
judge="odd number"
else
judge="even number"
end
mem.inter.WriteAscii(100, judge, string.len(judge))
end</td></tr>
<tr><td>建立數值輸入、文數值顯示元件</td><td>■ 建立數值輸入元件，寫入記憶體位址為$10。
■ 建立文數值顯示元件，讀取記憶體位址為$100。</td></tr>
<tr><td>執行結果</td><td>■ 完成 Lua 程式的編寫和元件的建立後，請下載專案至人機。
■ 於$10 輸入 1，$100 顯示為 odd number；於$10 輸入 2，$100 顯示為 even number。
 


</td></tr>
</table>

4.2 Internal memory - $ (內部暫存器($))

此指令可供使用者可以對內部暫存器位址做讀和寫的動作，基本語法包含：

指令	指令運算式	說明
Internal memory - $ (內部暫存器($))	mem.inter.Read	讀取內部暫存器數值 (單位：Word)
	mem.inter.ReadDW	讀取內部暫存器數值 (單位：Double Word)
	mem.inter.ReadFloat	讀取內部暫存器數值 (單位：Float)
	mem.inter.ReadBit	讀取內部暫存器位元
	mem.inter.ReadDouble	讀取內部暫存器數值 (單位：雙精度浮點數)
	mem.inter.Write	寫入內部暫存器數值 (單位：Word)
	mem.inter.WriteDW	寫入內部暫存器數值 (單位：Double Word)
	mem.inter.WriteFloat	寫入內部暫存器數值 (單位：Float)
	mem.inter.WriteBit	寫入內部暫存器位元
	mem.inter.ReadAscii	讀取內部暫存器字串
	mem.inter.WriteAscii	寫入內部暫存器字串
	mem.inter.WriteDouble	寫入內部暫存器數值 (單位：雙精度浮點數)

以下將一一詳細介紹。

■ mem.inter.Read：讀取內部暫存器位址數值 (單位：Word)

指令名稱	mem.inter.Read
指令運算式	Value = **mem.inter.Read**(index, [value_format])
參數定義	index：整數；記憶體索引，範圍為 0 ~ 199999 value_format：格式字串；有號數填入：“signed”，可省略
範例	v1 = mem.inter.Read(0)
範例說明	以 Word 為單位讀取$0 數值。
回傳值	v1：0x0000 ~ 0xFFFF，失敗回傳 0

■ mem.inter.ReadDW：讀取內部暫存器位址數值 (單位：Double Word)

指令名稱	mem.inter.ReadDW
指令運算式	Value = **mem.inter.ReadDW**(index, [value_format])
參數定義	index：整數；記憶體索引，範圍為 0 ~ 199998 value_format：格式字串；有號數填入：“signed”，可省略
範例	dw = mem.inter.ReadDW(0)
範例說明	以 Double Word 為單位讀取$0 數值。
回傳值	v1：0x00000000 ~ 0xFFFFFFFF，失敗回傳 0

■ mem.inter.ReadFloat：讀取內部暫存器位址數值 (單位：Float)

指令名稱	mem.inter.ReadFloat
指令運算式	Value = **mem.inter.ReadFloat**(index)
參數定義	index：整數；記憶體索引，範圍為 0 ~ 199998
範例	f1 = mem.inter.ReadFloat(0)
範例說明	以 Float 為單位讀取$0 數值。
回傳值	f1：單精度浮點數值

■ mem.inter.ReadBit：讀取內部暫存器位址位元

指令名稱	mem.inter.ReadBit
指令運算式	Result = **mem.inter.ReadBit**(index, bit)
參數定義	index：整數；記憶體索引，範圍為 0 ~ 199999 bit：整數；BIT 索引，範圍為 0 ~ 15
範例	b1 = mem.inter.ReadBit(0, 15)
範例說明	讀取$0.15 位元。
回傳值	b1：整數；1 或 0；失敗回傳 0

■ mem.inter.ReadDouble：讀取內部暫存器位址數值 (單位：雙精度浮點數)

指令名稱	mem.inter.ReadDouble
指令運算式	Result = **mem.inter.ReadDouble**(index)
參數定義	index：整數；記憶體索引，範圍為 0 ~ 199996
範例	Q1 = mem.inter.ReadDouble(0)
範例說明	以雙精度浮點數為單位讀取$0 數值。
回傳值	Q1：雙精度浮點數值

■ mem.inter.Write：寫入數值到內部暫存器位址 (單位：Word)

指令名稱	mem.inter.Write
指令運算式	Result = **mem.inter.Write**(index, value)
參數定義	index：整數；記憶體索引，範圍為 0 ~ 199999 value：十進制整數
範例	result = mem.inter.Write(0, 123)
範例說明	以 Word 為單位寫入數值 123 至$0。
回傳值	result：成功回傳 1，失敗回傳 0

■ mem.inter.WriteDW：寫入數值到內部暫存器位址 (單位：Double Word)

指令名稱	mem.inter.WriteDW
指令運算式	Result = **mem.inter.WriteDW**(index, value)
參數定義	index：整數；記憶體索引，範圍為 0 ~ 199998 value：十進制整數
範例	result = mem.inter.WriteDW(0, 65536)
範例說明	以 Double Word 為單位寫入數值 65536 至$0。
回傳值	result：成功回傳 1，失敗回傳 0

■ mem.inter.WriteFloat：寫入數值到內部暫存器位址 (單位：Float)

指令名稱	mem.inter.WriteFloat
指令運算式	Result = **mem.inter.WriteFloat**(index, float_value)
參數定義	index：整數；記憶體索引，範圍為 0 ~ 199998 float_value：單精度浮點數值
範例	result = mem.inter.WriteFloat(0, 1.23)
範例說明	以 Float 為單位寫入數值 1.23 至$0。
回傳值	result：成功回傳 1，失敗回傳 0

■ mem.inter.WriteBit：寫入內部暫存器位址位元

指令名稱	mem.inter.WriteBit
指令運算式	Result = **mem.inter.WriteBit**(index, bit, logic)
參數定義	index：整數；記憶體索引，範圍為 0 ~ 199999 bit：整數；BIT 索引，範圍為 0 ~ 15 logic：整數；1 或 0
範例	result = mem.inter.WriteBit(0, 0, 1)
範例說明	觸發$0.0 位元。
回傳值	result：成功回傳 1，失敗回傳 0

■ mem.inter.WriteDouble：寫入內部暫存器位址數值 (單位：雙精度浮點數)

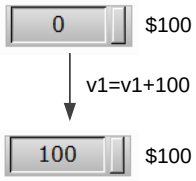
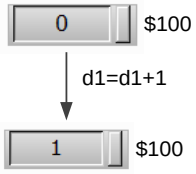
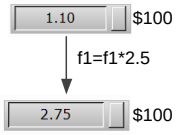
指令名稱	mem.inter.WriteDouble
指令運算式	Result = **mem.inter.WriteDouble**(index, Double_value)
參數定義	index：整數，記憶體索引，範圍為 0 ~ 199996 Double_value：雙精度浮點數值
範例	result = mem.inter.WriteDouble(10, 123456789.99)
範例說明	以雙精度浮點數為單位寫入 123456789.99 至$10。
回傳值	result：成功回傳 1，失敗回傳 0

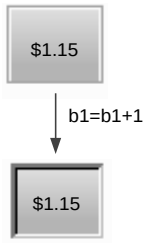
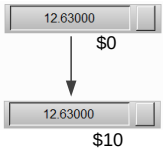
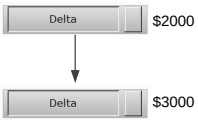
■ mem.inter.ReadAscii：讀取內部暫存器位址字串

指令名稱	mem.inter.ReadAscii
指令運算式	ascii_string = **mem.inter.ReadAscii**(start_index, string_len)
參數定義	start_index：整數；記憶體索引，範圍為 0 ~ 199999 string_len：整數；ascii 個數(bytes)
範例	ascii_string = mem.inter.ReadAscii(0, 4)
範例說明	以 4 個 bytes 為長度讀取$0 字串。
回傳值	ascii_string：ascii 字串

■ mem.inter.WriteAscii：寫入字串到內部暫存器位址

指令名稱	mem.inter.WriteAscii
指令運算式	Result = **mem.inter.WriteAscii**(start_index, asci_string, string_len)
參數定義	start_index：整數；記憶體索引，範圍為 0 ~ 199999 asci_string：ascii 字串 string_len：整數；ascii 個數(bytes)
範例	Result = mem.inter.WriteAscii(0, "posheng", string.len("posheng"))
範例說明	以"posheng"的字串長度寫入 posheng 至$0。 註：寫入的資料最後會再加上"\0"字元。
回傳值	Result：整數，寫入 ascii 個數(bytes)

指令	範例	結果
mem.inter.Read mem.inter.Write	while true do v1 = mem.inter.Read(100) v1 = v1 + 100 mem.inter.Write(100, v1) end	
	指令說明	
	讀取內部暫存器位址$100 為 v1，將 v1 加上 100 後，將 v1 寫入內部暫存器位址$100。	
指令	範例	結果
mem.inter.ReadDW mem.inter.WriteDW	while true do d1 = mem.inter.ReadDW(100) d1 = d1 + 1 mem.inter.WriteDW(100, d1) end	
	指令說明	
	讀取內部暫存器$100 為 d1，將 d1 加上 1 後，將結果寫入內部暫存器位址$100，讀取和寫入的單位為 Double Word。	
指令	範例	結果
mem.inter.ReadFloat mem.inter.WriteFloat	while true do mem.inter.WriteFloat(100,1.1) d1 = mem.inter.ReadFloat(100) d1 = d1 *2.5 mem.inter.WriteFloat(100, d1) end	
	指令說明	
	將浮點數 1.1 寫入內部暫存器位址$100，讀取內部暫存器位址$100 浮點數為 f1，將 f1 乘上 2.5 後，將 f1 寫入內部暫存器位址$100，讀取和寫入的格式為浮點數。	

指令	範例	結果
mem.inter.ReadBit mem.inter.WriteBit	while true do b1 = mem.inter.ReadBit(1,15) b1=b1+1 mem.inter.WriteBit(1,15,b1) end **指令說明** 讀取內部暫存器位址$1.15 為 b1，將 b1 加上 1 後，將結果寫入內部暫存器位址 $1.15。	
指令	**範例**	**結果**
mem.inter.ReadDouble mem.inter.Write Double	while true do q1 = mem.inter.ReadDouble (0) mem.inter.WriteDouble (10, q1) end **指令說明** 讀取內部暫存器位址$0 為 q1 後，將 q1 寫入內部暫存器位址$10，讀取和寫入的單位為雙精度浮點數。	
指令	**範例**	**結果**
mem.inter.ReadAscii mem.inter.WriteAscii	while true do str2 = mem.inter.ReadAscii(2000, 5) mem.inter.WriteAscii(3000, str2, string.len(str2)) end **指令說明** 以字串長度為 5，讀取內部暫存器位址 $2000 字串為 str2，將字串 str2 寫入至內部暫存器位址$3000。 註：字串長度單位為 bytes。	

4.3 Static memory - $M (斷電保持內部暫存器($M))

此指令可供使用者對人機中的斷電保持內部暫存器位址做讀寫動作，基本語法包含：

指令	指令運算式	說明
Static memory - $M (斷電保持內部暫存器($M))	mem.static.Read	讀取斷電保持暫存器數值 (單位：Word)
	mem.static.ReadDW	讀取斷電保持暫存器數值 (單位：Double Word)
	mem.static.ReadFloat	讀取斷電保持暫存器數值 (單位：Float)
	mem.static.ReadBit	讀取斷電保持暫存器位元
	mem.static.ReadDouble	讀取斷電保持暫存器數值 (單位：雙精度浮點數)
	mem.static.Write	寫入斷電保持暫存器數值 (單位：Word)
	mem.static.WriteDW	寫入斷電保持暫存器數值 (單位：Double Word)
	mem.static.WriteFloat	寫入斷電保持暫存器數值 (單位：Float)
	mem.static.WriteBit	寫入斷電保持暫存器位元
	mem.static.WriteDouble	寫入斷電保持暫存器數值 (單位：雙精度浮點數)
	mem.static.ReadAscii	讀取斷電保持暫存器字串
	mem.static.WriteAscii	寫入斷電保持暫存器字串

以下將一一詳細說明。

■ mem.static.Read：讀取斷電保持器位址數值 (單位：Word)

指令名稱	mem.static.Read
指令運算式	Value = **mem.static.Read**(index, [value_format])
參數定義	index：整數；記憶體索引，範圍為 0 ~ 1023 value_format：格式化字串；有號數填入：“signed”，可省略
範例	v1 = mem.static.Read(0)
範例說明	以 Word 為單位讀取$M0 數值。
回傳值	v1：0x0000 ~ 0xFFFF；失敗回傳 0

■ mem.static.ReadDW：讀取斷電保持器位址數值 (單位：Double Word)

指令名稱	mem.static.ReadDW
指令運算式	Value = **mem.static.ReadDW**(index, [value_format])
參數定義	index：整數；記憶體索引，範圍為 0 ~ 1022 value_format：格式化字串；有號數填入：“signed”，可省略
範例	dw = mem.static.ReadDW(0)
範例說明	以 Double Word 為單位讀取$M0 數值。
回傳值	dw：0x00000000 ~ 0xFFFFFFFF，失敗回傳 0

■ mem.static.ReadFloat：讀取斷電保持器位址數值 (單位：Float)

指令名稱	mem.static.ReadFloat
指令運算式	Value = **mem.static.ReadFloat**(index)
參數定義	index：整數；記憶體索引，範圍為 0 ~ 1022
範例	f1 = mem.static.ReadFloat(0)
範例說明	以 Float 為單位讀取$M0 數值。
回傳值	f1：單精度浮點數值

■ mem.static.ReadBit：讀取斷電保持器位址位元

指令名稱	mem.static.ReadBit
指令運算式	Result = **mem.static.ReadBit**(index, bit)
參數定義	index：整數；記憶體索引，範圍為 0 ~ 1023 bit：整數；BIT 索引，0 ~ 15
範例	b1 = mem.static.ReadBit(0, 15)
範例說明	讀取斷電保持器$M0.15 位元。
回傳值	b1：整數；1 或 0

■ mem.static.ReadDouble：讀取斷電保持器位址數值 (單位：雙精度浮點數)

指令名稱	mem.static.ReadDouble
指令運算式	Result = **mem.static.ReadDouble**(index)
參數定義	index：整數；記憶體索引，範圍為 0 ~ 1020
範例	Q1 = mem.static.ReadDouble(0)
範例說明	以雙精度浮點數為單位讀取$M0 數值。
回傳值	Q1：雙精度浮點數值

■ mem.static.ReadAscii：讀取斷電保持器位址字串

指令名稱	mem.static.ReadAscii
指令運算式	ascii_string = **mem.static.ReadAscii**(start_index, string_len)
參數定義	start_index：整數；記憶體索引，範圍為 0 ~ 1023 string_len：整數；ascii 個數(bytes)
範例	ascii_string = mem.static.ReadAscii(0, 4)
範例說明	以 4 個 bytes 為長度讀取斷電保持器位址$M0 的字串。
回傳值	ascii_string：ascii 字串

■ mem.static.Write：寫入數值到斷電保持器位址 (單位：Word)

指令名稱	mem.static.Write
指令運算式	Result = **mem.static.Write**(index, value)
參數定義	index：整數；記憶體索引，範圍為 0 ~ 1023 value：整數
範例	result = mem.static.Write(0, 123)
範例說明	以 Word 為單位寫入數值 123 至$M0。
回傳值	result：成功回傳 1，失敗回傳 0

■ mem.static.WriteDW：寫入數值到斷電保持器位址 (單位：Double Word)

指令名稱	mem.static.WriteDW
指令運算式	Result = **mem.static.WriteDW**(index, value)
參數定義	index：整數；記憶體索引，範圍為 0 ~ 1023 value：整數
範例	result = mem.static.WriteDW(0, 65535)
範例說明	以 Double Word 為單位寫入數值 65535 至$M0。
回傳值	result：成功回傳 1，失敗回傳 0

■ mem.static.WriteFloat：寫入數值到斷電保持器位址 (單位：Float)

指令名稱	mem.static.WriteFloat
指令運算式	Result = **mem.static.WriteFloat**(index, value_float)
參數定義	index：整數；記憶體索引，範圍為 0 ~ 1023 value_float：單精度浮點數值
範例	result = mem.static.WriteFloat(0, 1.23)
範例說明	以 Float 為單位寫入數值 1.23 至$M0。
回傳值	result：成功回傳 1，失敗回傳 0

■ mem.static.WriteBit：寫入斷電保持器位元

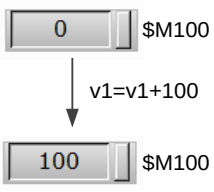
指令名稱	mem.static.WriteBit
指令運算式	Result = **mem.static.WriteBit**(index, bit, logic)
參數定義	index：整數；記憶體索引，範圍為 0 ~ 1023 bit：整數；BIT 索引，0 ~ 15 logic：整數；1 或 0
範例	result = mem.static.WriteBit(0, 0, 1)
範例說明	觸發斷電保持器$M0.0 位元。
回傳值	result：成功回傳 1，失敗回傳 0

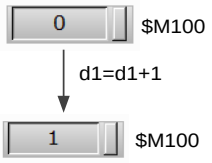
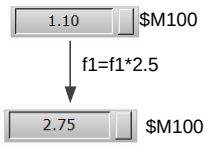
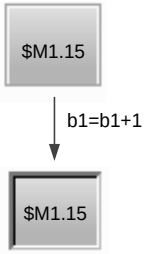
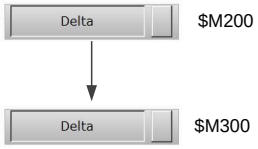
■ mem. static.WriteDouble：寫入斷電保持器位址數值 (單位：雙精度浮點數)

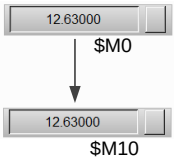
指令名稱	mem.static.WriteDouble
指令運算式	Result = **mem.static.WriteDouble**(index, Double_value)
參數定義	index：整數；記憶體索引，範圍為 0 ~ 1020 Double_value：雙精度浮點數值
範例	result = mem.static.WriteDouble(10, 123456789.99)
範例說明	以雙精度浮點數為單位寫入 123456789.99 至$M10。
回傳值	result：成功回傳 1，失敗回傳 0

■ mem.static.WriteAscii：寫入字串到斷電保持器位址

指令名稱	mem.static.WriteAscii
指令運算式	Result = **mem.static.WriteAscii**(start_index, ascii_string, string_len)
參數定義	start_index：整數；記憶體索引，範圍為 0 ~ 1023 ascii_string：ascii 字串 string_len：整數；ascii 個數(bytes)
範例	result = mem.static.WriteAscii(0, "posheng", string.len("posheng"))
範例說明	以"posheng"的字串長度寫入 posheng 至$M0。 註：寫入的資料最後會再加上"\0"字元。
回傳值	result：成功回傳 1，失敗回傳 0

指令	範例	結果
mem.static.Read mem.static.Write	v1 = mem.static.Read(100) v1 = v1 + 100 mem.static.Write(100, v1)	
	指令說明	
	讀取斷電保持器位址$M100 為 v1，將 v1 加上 100 後，將結果寫入斷電保持器位址$M100。	

指令	範例	結果
mem.static.ReadDW mem.static.WriteDW	d1 = mem.static.ReadDW(100) d1 = d1 + 1 mem.static.WriteDW(100, d1) **指令說明** 讀取斷電保持器位址$M100 為 d1，將 d1 加上 1 後，將 d1 寫入斷電保持器位址$M100，讀取和寫入的單位為 Double Word。	
指令	**範例**	**結果**
mem.static.ReadFloat mem.static.WriteFloat	mem.static.WriteFloat(100,1.1) f1 = mem.static.ReadFloat(100) f1 = f1 *2.5 mem.static.WriteFloat(100, f1) **指令說明** 將浮點數 1.1 寫入斷電保持器位址$M100，讀取斷電保持器位址$M100 浮點數為 f1，將 f1 乘上 2.5 後，將 f1 寫入斷電保持器位址$M100，讀取和寫入的格式為浮點數。	
指令	**範例**	**結果**
mem.static.ReadBit mem.static.WriteBit	b1 = mem.static.ReadBit(1,15) b1=b1+1 mem.static.WriteBit(1,15,b1) **指令說明** 讀取斷電保持器位址$M1.15 為 b1，將 b1 加上 1 後，將 b1 寫入斷電保持器位址$M1.15。	
指令	**範例**	**結果**
mem.static.ReadAscii mem.static.WriteAscii	str2 = mem.static.ReadAscii(200, 5) mem.static.WriteAscii(300, str2, 5) **指令說明** 以字串長度為 5，讀取斷電保持器位址$M200 字串為 str2，將字串 str2 寫入至斷電保持器位址$M300，字串長度為 5。 註：字串長度單位為 bytes。	

<table>
<tr><th>指令</th><th>範例</th><th>結果</th></tr>
<tr><td rowspan="3">mem.static.ReadDouble
mem.static.WriteDouble</td><td>while true do
q1 = mem.static.ReadDouble (0)
mem.static.WriteDouble (10, q1)
end</td><td rowspan="3"> 
</td></tr>
<tr><th>指令說明</th></tr>
<tr><td>讀取斷電保持器位址$M0 為 q1 後，將 q1 寫入斷電保持器位址$M10，讀取和寫入的單位為雙精度浮點數。</td></tr>
</table>

4.4 External link (外部記憶體位址)

此指令可供使用者可以對人機中的外部裝置記憶體位址做讀寫動作，基本語法包含：

指令	指令運算式	說明
External link (位址)	link.Read	讀取外部記憶體位址數值 (單位：Word)
	link.ReadDW	讀取外部記憶體位址數值 (單位：Double Word)
	link.ReadFloat	讀取外部記憶體位址數值 (單位：Float)
	link.ReadBit	讀取外部記憶體位址位元
	link.ReadAscii	讀取外部記憶體位址字串
	link.ReadDouble	讀取外部記憶體位址數值 (單位：雙精度浮點數)
	link.Write	寫入外部記憶體位址數值 (單位：Word)
	link.WriteDW	寫入外部記憶體位址數值 (單位：Double Word)
	link.WriteFloat	寫入外部記憶體位址數值 (單位：Float)
	link.WriteBit	寫入外部記憶體位址位元
	link.WriteAscii	寫入外部記憶體位址字串
	link.WriteDouble	寫入外部記憶體位址數值 (單位：雙精度浮點數)
	link.CopyFromInter	從人機內部暫存器位址複製資料至外部記憶體位址
	link.CopyToInter	從外部記憶體位址複製資料至人機內部暫存器位址
	link.CopyArray	從內部/外部記憶體位址複製資料至人機內部/外部記憶體位址
	link.DownloadPLC	使用 COM 的通訊方式透過人機下載 isp 或 dvp 程序至 PLC 中
	link.DownloadEthPLC	使用網路的通訊方式透過人機下載 isp 或 dvp 程序至 PLC 中
	link.WritePasswordPLC	下載系統密碼至台達 PLC
	link.SetDefaultStationNo	設定人機欲通訊的 PLC 預設站號
	link.SetHMIStationNo	設定 HMI Slave 站號(Modbus Slave)
	link.CODESYSAppDownload	使用網路的通訊方式透過人機下載 CODESYS 程序至 PLC 中
	link.CODESYSAppUpload	使用網路的通訊方式上載 CODESYS 程序至 HMI 的 USB 中

以下將一一詳細介紹。

■ link.Read：讀取外部記憶體位址數值 (單位：Word)

指令名稱	link.Read
指令運算式	Result = **link.Read**(addr, [value_format])
參數定義	addr：字串，記憶體位址，例如："{Link2}1@D1" value_format：格式化字串；有號數填入："signed"，可省略
範例	v1 = link.Read("{Link2}1@D1")
範例說明	以 Word 為單位讀取以下外部記憶體位址之數值：{Link2}通訊連線 2 的站號 1 控制器、通訊地址 D1。
回傳值	v1：整數

■ link.ReadDW：讀取外部記憶體位址數值 (單位：Double Word)

指令名稱	link.ReadDW
指令運算式	Result = **link.ReadDW**(addr, [value_format])
參數定義	addr：字串，記憶體位址，例如："{Link2}1@D1" value_format：格式化字串；有號數填入："signed"，可省略
範例	dw = link.ReadDW("{Link2}1@D1")
範例說明	以 Double Word 為單位讀取以下外部記憶體位址之數值：{Link2}通訊連線 2 的站號 1 控制器、通訊地址 D1。
回傳值	dw：整數

■ link.ReadFloat：讀取外部記憶體位址數值 (單位：Float)

指令名稱	link.ReadFloat
指令運算式	Result = **link.ReadFloat**(addr)
參數定義	addr：字串，記憶體位址，例如："{Link2}1@D1"
範例	f1 = link.ReadFloat("{Link2}1@D1")
範例說明	以 Float 為單位讀取以下外部記憶體位址之數值：{Link2}通訊連線 2 的站號 1 控制器、通訊地址 D1。
回傳值	f1：單精度浮點數值

■ link.ReadBit：讀取外部記憶體位址位元

指令名稱	link.ReadBit
指令運算式	Result = **link.ReadBit**(addr)
參數定義	addr：字串，記憶體位址，例如："{Link2}1@M100"
範例	b = link.ReadBit("{Link2}1@M1")
範例說明	讀取以下外部記憶體位址位元：{Link2}通訊連線 2 的站號 1 控制器、通訊地址 M1 位元。
回傳值	v1：整數；1 或 0

■ link.ReadAscii：讀取外部記憶體位址字串

指令名稱	link.ReadAscii
指令運算式	asciiString, result, errMsg = **link.ReadAscii**(addr, string_len)
參數定義	addr：字串，記憶體位址，例如："{Link2}1@D1" string_len：整數，ascii 個數(bytes)
範例	ascii, ret, errMsg = link.ReadAscii("{Link2}1@D1", 20)
範例說明	以 20 bytes 為單位讀取以下外部記憶體位址之字串：{Link2}通訊連線 2 的站號 1 控制器、通訊地址 D1。
回傳值	ascii：ascii 字串 ret：1：成功；0：失敗 errMsg：字串，錯誤訊息

■ link.ReadDouble：讀取外部記憶體位址數值 (單位：雙精度浮點數)

指令名稱	link.ReadDouble
指令運算式	Result = **link.ReadDouble**(addr)
參數定義	addr：字串，記憶體位址，例如："{Link2}1@D1"
範例	A1 = link.ReadDouble("{Link2}1@D1")
範例說明	以雙精度浮點數為單位讀取以下外部記憶體位址之數值：{Link2}通訊連線 2 的站號 1 控制器、通訊地址 D1。
回傳值	A1：雙精度浮點數值

■ link.Write：寫入數值到外部記憶體位址 (單位：Word)

指令名稱	link.Write
指令運算式	Result = **link.Write**(addr, value_word)
參數定義	addr：字串，記憶體位址，例如："{Link2}1@D0" value_word：整數
範例	Result = link.Write("{Link2}1@D1", 123)
範例說明	以 Word 為單位寫入數值 123 至以下外部記憶體位址：{Link2}通訊連線 2 的站號 1 控制器、通訊地址 D1。
回傳值	Result：成功回傳 1，失敗回傳 0

■ link.WriteDW：寫入數值到外部記憶體位址 (單位：Double Word)

指令名稱	link.WriteDW
指令運算式	Result = **link.WriteDW**(addr, value_dword)
參數定義	addr：字串，記憶體位址，例如："{Link2}1@D0" value_dword：整數
範例	Result = link.WriteDW("{Link2}1@D1", 65536)
範例說明	以 Double Word 為單位寫入數值 65536 至以下外部記憶體位址：{Link2}通訊連線 2 的站號 1 控制器、通訊地址 D1。
回傳值	Result：成功回傳 1，失敗回傳 0

■ link.WriteFloat：寫入數值到外部記憶體位址 (單位：Float)

指令名稱	link.WriteFloat
指令運算式	Result = **link.WriteFloat**(addr, value_float)
參數定義	addr：字串，記憶體位址，例如："{Link2}1@D0" value_float：單精度浮點數值
範例	Result = link.WriteFloat("{Link2}1@D1", 1.23)
範例說明	以 Float 為單位寫入數值 1.23 至以下外部記憶體位址：{Link2}通訊連線 2 的站號 1 控制器、通訊地址 D1。
回傳值	Result：成功回傳 1，失敗回傳 0

■ link.WriteBit：寫入外部記憶體位址位元

指令名稱	link.WriteBit
指令運算式	Result = **link.WriteBit**(addr, value_bit)
參數定義	addr：字串，記憶體位址，例如："{Link2}1@M100" value_bit：整數，1 或 0
範例	Result = link.WriteBit("{Link2}1@M1", 1)
範例說明	觸發以下外部記憶體位址：{Link2}通訊連線 2 的站號 1 控制器、通訊地址 M1。
回傳值	Result：成功回傳 1，失敗回傳 0

■ link.WriteAscii：寫入字串到外部記憶體位址

指令名稱	link.WriteAscii
指令運算式	Result = **link.WriteAscii**(addr, ascii, asci_len)
參數定義	addr：字串，記憶體位址，例如："{Link2}1@D0" ascii：字串，UTF-8 編碼(bytes) asci_len：整數，ascii 個數(bytes)
範例	Result = link.WriteAscii("{Link2}1@D1", "posheng", string.len("posheng"))
範例說明	以"posheng"的字串長度寫入 posheng 至以下外部記憶體位址：{Link2}通訊連線 2 的站號 1 控制器、通訊地址 D1 位址。
回傳值	Result：成功回傳 1，失敗回傳 0

■ link.WriteDouble：寫入數值到外部記憶體位址 (單位：雙精度浮點數)

指令名稱	link.WriteDouble
指令運算式	Result = **link.WriteDouble**(addr, value_double)
參數定義	addr：字串，記憶體位址，例如："{Link2}1@D0" value_double：雙精度浮點數值
範例	Result = link.WriteDouble("{Link2}1@D1", 1234567.89)
範例說明	以雙精度浮點數為單位寫入數值 1234567.89 至以下外部記憶體位址：{Link2}通訊連線 2 的站號 1 控制器、通訊地址 D1。
回傳值	Result：成功回傳 1，失敗回傳 0

■ link.CopyFromInter：從人機內部暫存器位址複製資料至外部記憶體位址

指令名稱	link.CopyFromInter
指令運算式	result = **link.CopyFromInter**(addr, interMemIndex, wordLen)
參數定義	addr：字串，記憶體位址，例如："{Link2}1@D1" interMemIndex：整數值，內部暫存器的起始位址 wordLen：整數，個數(Word)
範例	Result = link.CopyFromInter("{Link2}1@D1", 100, 6)
範例說明	以長度 6 個 words 將人機位址$100 中的資料搬移至以下外部記憶體位址：{Link2}通訊連線 2 的站號 1 控制器、通訊地址 D1。
回傳值	Result：成功回傳 1，失敗回傳 0

■ link.CopyToInter：從外部記憶體位址複製資料至人機內部暫存器位址

指令名稱	link.CopyToInter
指令運算式	result = **link.CopyToInter**(addr, interMemIndex, wordLen)
參數定義	addr：字串，記憶體位址，例如："{Link2}1@D1" interMemIndex：整數值，內部暫存器的起始位址 wordLen：整數，個數(Word)
範例	Result = link.CopyToInter("{Link2}1@D1", 100, 6)
範例說明	以長度 6 個 words 將{Link2}通訊連線 2 的站號 1 控制器，通訊地址 D1 的資料搬移至$100。
回傳值	Result：成功回傳 1，失敗回傳 0

■ link.CopyArray：從人機內部/外部記憶體位址複製資料至人機內部/外部記憶體位址

指令名稱	link.CopyArray
指令運算式	result = **link.CopyArray**(dst_addr, dst_offset, src_addr, src_offset, wordLen)
參數定義	dst_addr：字串或整數，目的端位址 dst_offset：整數，目的端偏移長度 src_addr：字串或整數，來源端位址 src_offset：整數，來源端偏移長度 wordLen：整數，複製長度
範例	範例一：result = link.CopyArray(95, 0, 190, 3, 6) 範例二：result = link.CopyArray(95, 0,"{Link2}1@D0", 0, 6)
範例說明	範例一：將$193 ~ $198 以長度 6 個 words 搬移至$95 ~ $100。 範例二：將{Link2}1@D0 ~ {Link2}1@D5 以長度 6 個 words 搬移至$95 ~ $100。
回傳值	result：成功回傳 1，失敗回傳 0，參數錯誤回傳-1

■ link.DownloadPLC：使用 COM 的通訊方式透過人機下載 isp 或 dvp 程序至 PLC 中

指令名稱	link.DownloadPLC
指令運算式	ret, errDesc = **link.DownloadPLC**(comNo, stationNo, diskID, fileName, projectPwd, plcSystemSecurityPwd)
參數定義	comNo：整數值，通訊串口號，COM1 填 1，COM2 填 2，… stationNo：整數：1 ~ 255 diskID：整數，磁片 ID；2：USB；3：SD fileName：字串；檔案名，如 ss.dvp、ss.isp projectPwd：字串；專案密碼 plcSystemSecurityPwd：字串；PLC 系統安全密碼
範例	ret, errDesc = link.DownloadPLC(2, 1, 2, "EH.dvp", "1234", "5678")
範例說明	以 COM2 為接口，將儲存於 USB 的 EH.dvp 以專案密碼 1234、系統密碼 5678 下載程序到 PLC。 註： 1. 此指令只支援 COM 通訊，不支援網路通訊 2. 需先確認人機和該 PLC 可以通訊。 3. 目前支援台達 DVP、AS、AH 系列 PLC。
回傳值	ret：成功回傳 1；失敗回傳 0 errDesc：字串；錯誤訊息

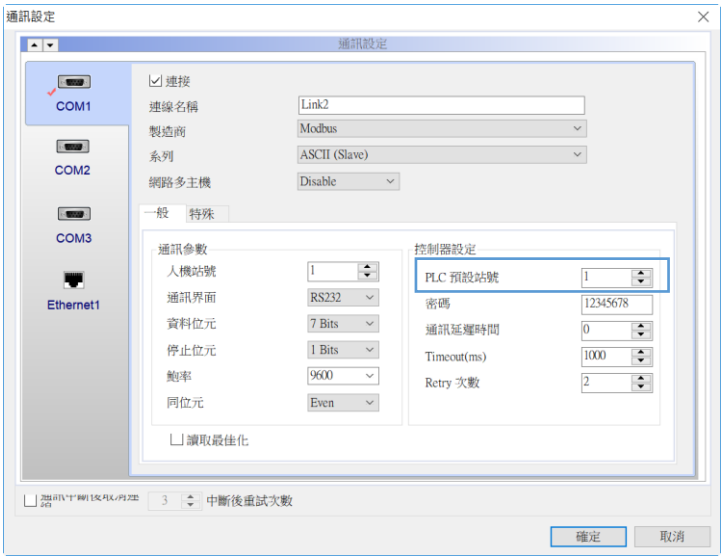
■ link.DownloadEthPLC：使用網路的通訊方式透過人機下載 isp 或 dvp 程序至 PLC 中

指令名稱	link.DownloadEthPLC
指令運算式	ret, errDesc = **link.DownloadEthPLC**(ip, port, stationNo, diskID, fileName, projectPwd, plcSystemSecurityPwd)
參數定義	ip：字串；“192.168.0.1”，… port：整數 stationNo：整數值；站號：1 ~ 255 diskID：整數，磁片 ID；2：USB；3：SD fileName：字串；檔案名，如 delta.dvp、delta.isp projectPwd：字串；專案密碼 plcSystemSecurityPwd：字串；PLC 系統安全密碼
範例	ret, errDesc = link.DownloadEthPLC("192.168.123.205", 502, 1, 2, "delta.isp", "1234", "5678")
範例說明	以網路為接口，將儲存於 USB 的 delta.isp 以專案密碼 1234、系統密碼 5678 下載程序到 PLC。 註： 1. 此指令只支援網路通訊，不支援 COM 通訊。 2. 需先確認人機和該 PLC 在相同網域。 3. 目前支援台達 DVP、AS、AH 系列 PLC。
回傳值	ret：成功回傳 1；失敗回傳 0 errDesc：字串；錯誤訊息

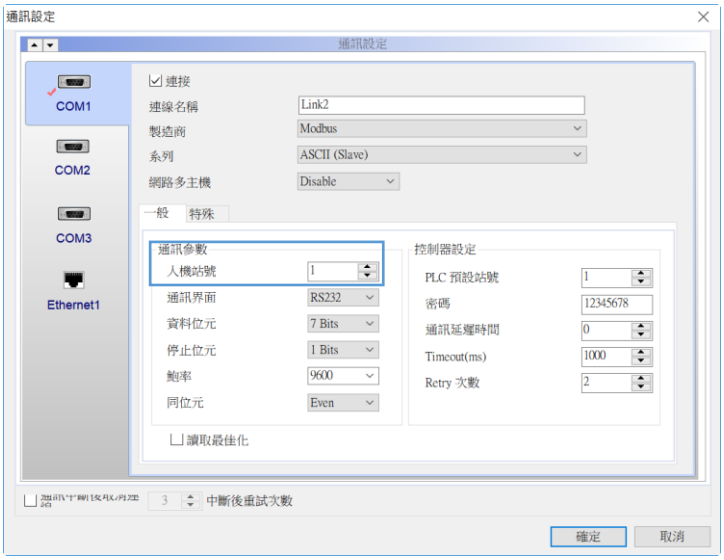
■ link.WritePasswordPLC：使用 COM 的通訊方式寫入系統密碼至 PLC 中

指令名稱	link.WritePasswordPLC
指令運算式	ret, errDesc = **link.WritePasswordPLC**(comNo, stationNo, oldPlcSystemPwd, newPlcSystemPwd)
參數定義	comNo：整數值，通訊串口號，COM1 填 1，COM2 填 2，… stationNo：整數：1 ~ 255 oldPlcSystemPwd：字串；舊的 PLC 系統安全密碼 newPlcSystemPwd：字串；新的 PLC 系統安全密碼
範例	ret, errDesc = link.WritePasswordPLC(2, 1, "1234", "2222")
範例說明	以 COM2 為接口，更改 PLC 密碼為 2222，假設 PLC 設有系統密碼，會比對是否和”1234”相符合。 註： 1. 此指令只支援 COM 通訊，不支援網路通訊。 2. 需先確認人機和該 PLC 可以通訊。 3. 目前支援台達 DVP、AS、AH 系列 PLC。
回傳值	ret：成功回傳 1；失敗回傳 0 errDesc：字串；錯誤訊息

■ link.SetDefaultStationNo：設定人機欲通訊的 PLC 預設站號

指令名稱	link.SetDefaultStationNo
指令運算式	ret = **link.SetDefaultStationNo**(link_number, station_number)
參數定義	link_number：整數值，通訊串口號，COM1 填 0，COM2 填 1，… station_number：整數：1 ~ 255
範例	ret = link.SetDefaultStationNo(0, 1)
範例說明	線上更改 COM1 通訊的 PLC 預設站號為 1。 
回傳值	ret：成功回傳 1；參數錯誤回傳-1

■ link.SetHMIStationNo：設定 HMI Slave 站號(Modbus Slave)

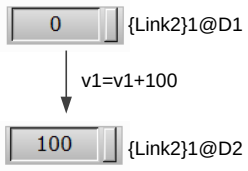
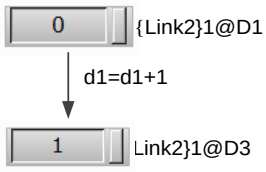
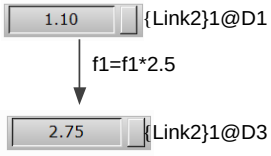
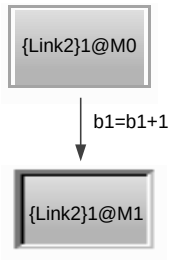
指令名稱	link.SetHMIStationNo
指令運算式	ret = **link.SetHMIStationNo**(link_number, station_number)
參數定義	link_number：整數值，通訊串口號，COM1 填 0，COM2 填 1，… station_number：整數：1 ~ 255
範例	ret = link.SetHMIStationNo(0, 1) 註：此指令只能用於當 HMI 為 Modbus Slave 時，故通訊參數須選為 Modbus ASCII/RTU (Slave)。
範例說明	更改 COM1 人機的站號為 1，通訊參數需設定為 Modbus ASCII/RTU (Slave)。 
回傳值	ret：成功回傳 1；參數錯誤回傳-1

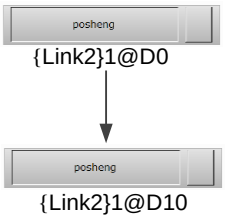
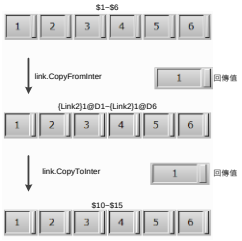
- link.CODESYSAppDownload：使用網路的通訊方式透過人機下載 CODESYS 程序至 PLC 中

指令名稱	link.CODESYSAppDownload
指令運算式	ret, errDesc = **link.CODESYSAppDownload**(connMethodID, address, diskID, appFile, showMsgBox, username, password)
參數定義	connMethodID：整數，連線代碼；0：PLC IP 位址；1：PLC 邏輯位址 address：字串，PLC IP 位址：“192.168.0.1”；PLC IP 邏輯位址：“002D” diskID：整數，磁片 ID；2：USB；8：AX8 appFile：字串，應用程序檔名，例如："Application.app" showMsgBox：整數；1：下載失敗會顯示錯誤視窗；0：下載失敗不會顯示錯誤視窗 username：字串，認證使用者名稱："Admin"；填空字串””代表無認證 password：字串，認證使用者密碼："12345"；填空字串””代表無認證
範例	connMethodID = 0 address = "192.168.123.23" diskID = 2 appFile = "AddressOffset.app" showMsgBox = 1 username = "" password = "" ret, errDesc = link.CODESYSAppDownload(connMethodID, address, diskID, appFile, showMsgBox, username, password)
範例說明	將位於人機的 USB 儲存裝置中的"AddressOffset.app"程序下載至 CODESYS PLC，如下載失敗會顯示錯誤視窗及訊息。 註：下載的資料夾下需含有.CRC 檔，如 AddressOffset.CRC。
回傳值	ret：成功回傳 1；失敗回傳 0 errDesc：字串；錯誤訊息

- link.CODESYSAppUpload：使用網路的通訊方式上載 CODESYS 程序至 HMI 的 USB 中

指令名稱	link.CODESYSAppUpload
指令運算式	ret, errDesc = **link.CODESYSAppUpload**(connMethodID, address, diskID, appFile, showMsgBox, username, password)
參數定義	connMethodID：整數，連線代碼；0：PLC IP 位址；1：PLC 邏輯位址 address：字串，PLC IP 位址：“192.168.0.1”；PLC IP 邏輯位址：“002D” diskID：整數，磁片 ID；2：USB；8：AX8 appFile：字串，應用程序檔名，例如："Application.app" showMsgBox：整數；1：下載失敗會顯示錯誤視窗；0：下載失敗不會顯示錯誤視窗 username：字串，認證使用者名稱："Admin"；填空字串””代表無認證 password：字串，認證使用者密碼："12345"；填空字串””代表無認證
範例	connMethodID = 0 address = "192.168.123.23" diskID = 2 appFile = "DELTA.app" showMsgBox = 1 username = "" password = "" ret, errDesc = link.CODESYSAppUpload(connMethodID, address, diskID, appFile, showMsgBox, username, password)
範例說明	上傳 CODESYS PLC 程序至位於人機的 USB 儲存裝置中，如上傳失敗會顯示錯誤視窗及訊息。
回傳值	ret：成功回傳 1；失敗回傳 0 errDesc：字串；錯誤訊息

指令	範例	結果
link.Read link.Write	v1 = link.Read("{Link2}1@D1") v1 = v1 + 100 link.Write("{Link2}1@D2",v1) **指令說明** 讀取外部記憶體位址{Link2}1@D1 為 v1，將 v1 加上 100 後，將結果寫入外部記憶體位址{Link2}1@D2，讀取和寫入的單位為 Word。	
指令	**範例**	**結果**
link.ReadDW link.WriteDW	d1 = link.ReadDW("{Link2}1@D1") d1 = d1 + 1 link.WriteDW("{Link2}1@D3",d1) **指令說明** 讀取外部記憶體位址{Link2}1@D1 為 d1，將 d1 加上 100 後，將結果寫入外部記憶體位址{Link2}1@D3，讀取和寫入的單位為 Double Word。	
指令	**範例**	**結果**
link.ReadFloat link.WriteFloat	link.WriteFloat("{Link2}1@D1", 1.1) f1 = link.ReadFloat("{Link2}1@D1") f1 = f1 * 2.5 link.WriteFloat("{Link2}1@D3",f1) **指令說明** 將浮點數 1.1 寫入外部記憶體位址 {Link2}1@D1，讀取外部記憶體位址 {Link2}1@D1 浮點數為 f1，將 f1 乘上 2.5 後，將結果寫入外部記憶體位址 {Link2}1@D3，讀取和寫入的單位為浮點數。	
指令	**範例**	**結果**
link.ReadBit link.WriteBit	b1 = link.ReadBit("{Link2}1@M0") b1 = b1 + 1 link.WriteBit("{Link2}1@M1",b1) **指令說明** 讀取外部記憶體位址{Link2}1@M0 為 b1，將 b1 加上 1 後，將結果寫入暫存區位址{Link2}1@M1。	

<table>
<tr><th>指令</th><th>範例</th><th>結果</th></tr>
<tr><td rowspan="2">link.ReadAscii
link.WriteAscii</td><td>link.WriteAscii("{Link2}1@D0",
"posheng",7)
ascii, ret, errMsg =
link.ReadAscii("{Link2}1@D0", 20)
link.WriteAscii("{Link2}1@D10", ascii,
20)</td><td rowspan="2"> 
</td></tr>
<tr><td>指令說明
以字串長度為 7，將字串 posheng 寫入至外部記憶體位址{Link2}1@D0，以 20 個 bytes 為單位讀取外部記憶體位址{Link2}1@D0 為字串 ascii，並以 20 個 bytes 為單位將字串 ascii 寫入至外部記憶體位址{Link2}1@D10。</td></tr>
<tr><th>指令</th><th>範例</th><th>結果</th></tr>
<tr><td rowspan="2">link.CopyFromInter
link.CopyToInter</td><td>result =
link.CopyFromInter("{Link2}1@D1", 1, 6)
mem.inter.Write(100,result)
result =
link.CopyToInter("{Link2}1@D1", 10, 6)
mem.inter.Write(200,result)</td><td rowspan="2"> 
</td></tr>
<tr><td>指令說明
以長度 6，將$1 的資料複製至{Link2}1@D1，並將回傳值寫入$100，再將{Link2}1@D1 的資料複製至$10，並將回傳值寫入$200。</td></tr>
</table>

4.5 File (文件讀取/寫入/匯出/刪除/列印)

此指令可供使用者對文件做讀、寫、匯出、製作 pdf 檔和列印的動作，基本語法包含：

指令	指令運算式	說明
File (文件讀取/寫入/匯出/清單)	file.Open	建立/開啟檔案
	file.Read	讀取檔案資料
	file.ReadLine	讀取檔案(以一行為單位)
	file.Write	寫入檔案
	file.Length	讀取檔案長度
	file.GetLineCount	讀取檔案總行數
	file.Seek	設定指標
	file.GetPos	取得當前指標位置
	file.GetError	檢查文件
	file.Close	關閉檔案
	file.List	取得存於人機內部的文件清單
	file.Export	匯出檔案
	file.Delete	刪除檔案
	file.DeleteDir	刪除目錄
	file.ToPDF	將檔案轉為 PDF
	file.ToPrinter	列印文件
	file.ListExternal	取得存於外部裝置的文件清單
	file.Exist	確認檔案是否存在
	file.PDFToPrinter	列印 PDF 文件
	file.Copy	複製檔案
	file.Move	移動檔案

以下將一一詳細介紹。

■ file.Open：建立/開啟檔案

指令名稱	file.Open
指令運算式	ret, fileHandle = **file.Open**(disk_id, file_name, add_utf8_id)
參數定義	disk_id：整數；0：人機內部；2：USB；3：SD file_name：字串；檔案名稱 add_utf8_id：1：增加 utf-8 識別字元；others：不額外加入識別字元
範例	ret, fileHandle = file.Open(2, "myFile001.txt")
範例說明	從外部儲存裝置 USB 開啟或建立檔名為"myFile001.txt"之檔案。fileHandle 為檔案資料供後續指令使用。 註：如儲存裝置含有名稱檔案，會開啟檔案；若不含有此名稱檔案，會建立檔案。
回傳值	ret：成功時回傳 1；失敗時回傳 0 fileHandle：整數；檔案指標

回傳值	說明
-1	不合法參數
-106	指定磁碟未備妥
-107	無法打開指定檔案
-136	不合法的檔案路徑

■ file.Read：讀取檔案資料

指令名稱	file.Read
指令運算式	ret, data = **file.Read**(fileHandle, len)
參數定義	fileHandle：整數；檔案指標 len：整數；字元長度
範例	ret, data = file.Read(fileHandle, 10) 註：fileHandle 參數由 file.Open 指令產生。
範例說明	以當前指標為初始位址，以 10 bytes 為長度讀取檔案 fileHandle 資料，讀取完後，當前指標位址更改為[初始位址+讀取長度]。 註：如需讀取特定指標位址，可以使用 file.Seek 指令更改當前指標位址。
回傳值	ret：整數；成功：讀取字串之長度；失敗：-1 data：字元集；成功：字元內容；失敗：nil

■ file.ReadLine：讀取檔案(以一行為單位)

指令名稱	file.ReadLine
指令運算式	ret, data = **file.ReadLine**(fileHandle)
參數定義	fileHandle：整數；檔案指標
範例	ret, data = file.ReadLine(fileHandle) 註：fileHandle 參數由 file.Open 指令產生。
範例說明	以當前指標為初始位址，以一行為長度，讀取檔案 fileHandle 資料。 註： 1. 一行的判斷標準為讀到回車符號(“/r”)或換行符號(“/n”)。 2. 可以使用第三方軟體 Notepad++讀取文檔，來確認回車符號和換行符號。
回傳值	ret：整數；成功：讀取字串之長度；失敗：回傳負數 回傳值 / 說明： -1 / 不合法參數 -102 / 讀取失敗 data：字元集；成功：字元內容；失敗：nil

■ file.Write：寫入檔案

指令名稱	file.Write
指令運算式	ret = **file.Write**(fileHandle, buffer, len)
參數定義	fileHandle：整數；檔案指標 buffer：Ascii 字元集；文字內容(UTF-8 格式) len：整數；寫入字元長度
範例	ret = file.Write(fileHandle, "posheng", 6) 註：fileHandle 參數由 file.Open 指令產生。
範例說明	以當前指標為初始位址，以 6 bytes 為長度，寫入字串 posheng 至 fileHandle 檔案。 註：如需讀取特定指標位址，可以使用 file.Seek 指令更改當前指標位址。
回傳值	ret：整數；成功：寫入字串之長度；失敗：-1

■ file.Length：讀取檔案長度

指令名稱	file.Length
指令運算式	ret, len = **file.Length**(fileHandle)
參數定義	fileHandle：整數；檔案指標
範例	ret, len = file.Length(fileHandle) 註：fileHandle 參數由 file.Open 指令產生。
範例說明	讀取檔案 fileHandle 的長度，單位為 Byte。
回傳值	ret：成功：1；失敗：0 len：成功：檔案長度；失敗：小於 0

■ file.GetLineCount：讀取檔案總行數

指令名稱	file.GetLineCount
指令運算式	ret, lineCount = **file.GetLineCount**(fileHandle)
參數定義	fileHandle：整數；檔案指標
範例	ret, lineCount = file.GetLineCount(fileHandle) 註：fileHandle 參數由 file.Open 指令產生
範例說明	讀取檔案 fileHandle 之總行數。 註： 1. 一行的判斷標準為讀到回車符號(“/r”)或換行符號(“/n”)。 2. 可以使用第三方軟體 Notepad++讀取文檔，來確認回車符號和換行符號。
回傳值	ret：成功：1；失敗：0 lineCount：成功：總行數；失敗：-1

■ file.Seek：設定指標

指令名稱	file.Seek
指令運算式	ret = **file.Seek**(fileHandle, offset, origin)
參數定義	fileHandle：整數；檔案指標 offset：整數；檔案位移數 origin：位移起始參考位置；檔案開頭位置：0，當前位置：1，檔案尾巴位置：2
範例	ret = file.Seek(fileHandle, 5, 0) 註：fileHandle 參數由 file.Open 指令產生。
範例說明	將當前指標設定在檔案開頭偏移 5 個 bytes 的位置。
回傳值	ret：成功：1；其它：不合法參數

■ file.GetPos：取得當前指標位置

指令名稱	file.GetPos
指令運算式	ret, pos = **file.GetPos**(fileHandle)
參數定義	fileHandle：整數；檔案指標
範例	ret, pos = file.GetPos(fileHandle) 註：fileHandle 參數由 file.Open 指令產生。
範例說明	得到當前指標位置。
回傳值	ret：整數；成功：1；失敗：-1 pos：整數；檔案指標位置

■ file.GetError：檢查文件

指令名稱	file.GetError
指令運算式	ret = **file.GetError**(fileHandle)
參數定義	fileHandle：整數；檔案指標
範例	ret = file.GetError(fileHandle) 註：fileHandle 參數由 file.Open 指令產生。
範例說明	檢查文件。
回傳值	ret：整數；成功：1；失敗：負數

回傳值	說明
-1	無效
-117	資料讀取失敗
-118	檔案存取失敗

■ file.Close：關閉檔案

指令名稱	file.Close
指令運算式	**file.Close**(fileHandle)
參數定義	fileHandle：整數；檔案指標
範例	file.Close(fileHandle) 註：fileHandle 參數由 file.Open 指令產生。
範例說明	關閉 fileHandle 檔案。
回傳值	無回傳值

■ file.List：取得存於人機內部的文件清單

指令名稱	file.List
指令運算式	ret, nameList = **file.List**()
參數定義	無參數
範例	ret, nameList = file.List()
範例說明	取得存於人機內部的文件清單，nameList 為清單，nameList[1]為第一筆檔案名稱，nameList[2]為第二筆檔案名稱，依此類推。
回傳值	ret：成功：1；失敗或無檔案：0 nameList：矩陣 table；成功：檔案名稱清單；失敗：nil

■ file.Export：匯出檔案

指令名稱	file.Export
指令運算式	ret, errCode = **file.Export**(name, disk_id)
參數定義	name：字串；檔案名稱 disk_id：整數；2：USB；3：SD
範例	ret, errCode = file.Export(“myFile.txt”, 2)
範例說明	將 file.Open 建立的 myFile.txt 匯出至外部裝置 USB。 註：只支援 txt 檔。
回傳值	ret：整數；成功：1；失敗：0 errCode：錯誤代碼

回傳值	說明
-1	參數設定錯誤
-106	外部儲存裝置未準備
-110	文件不存在
-111	匯出失敗
-136	不合法的路徑

■ file.Delete：刪除檔案

指令名稱	file.Delete
指令運算式	ret, err, errText = **file.Delete**(diskNo, fileName)
參數定義	diskNo：整數；0：HMI；2：USB；3：SD filename：字串；檔案名稱
範例	ret, err, errText = file.Delete(0, “myFile.txt”)
範例說明	刪除存於 HMI 的 myFile.txt 資料。
回傳值	ret：整數；成功：1；失敗：0 err：整數；錯誤代碼

回傳值	說明
-1	參數設定錯誤
-106	外部儲存裝置未準備
-114	刪除失敗
-136	不合法的路徑

errText：字串；錯誤描述

■ file.DeleteDir：刪除目錄

指令名稱	file.DeleteDir
指令運算式	ret, err, errText = **file.DeleteDir**(diskNo, dirName)
參數定義	diskNo：整數；0：HMI；2：USB；3：SD dirName：字串；目錄名稱
範例	ret, err, errText = file.DeleteDir(2, "/DELTA")
範例說明	刪除存於 USB 中名稱為 DELTA 的目錄。
回傳值	ret：整數；成功：1；失敗：0 err：整數；錯誤號碼 回傳值 / 說明 -1 / 參數設定錯誤 -106 / 外部儲存裝置未準備 -114 / 刪除失敗 -136 / 不合法的路徑 errText：字串；錯誤描述

■ file.ToPDF：將檔案轉為 PDF

指令名稱	file.ToPDF
指令運算式	ret, errCode = **file.ToPDF**(disk_id, srcFileName, pdfFileName, paperSize, fontSize, landscape)
參數定義	disk_id：整數；0：HMI；2：USB；3：SD srcFileName：字串；來源檔案名稱 pdfFileName：PDF 檔案名稱 paperSize：字串；“A3”、“A4” fontSize：整數；字體大小：8、10、12… landscape：整數；0：portrait；1：landscape
範例	ret, errCode = file.ToPDF(2, "file001.txt", "out001.pdf", "A4", 10, 0)
範例說明	將儲存於 USB 的檔案 file001.txt 轉為 PDF，並且設定名稱為 out001.pdf、格式為 A4、portrait、字體大小為 10。
回傳值	ret：整數；成功：1；失敗：0 errCode：整數；錯誤號碼

■ file.ToPrinter：列印文件

指令名稱	file.ToPrinter
指令運算式	ret, errCode = **file.ToPrinter**(disk_id, srcFileName, fontSize, outputPDF)
參數定義	disk_id：整數；0：HMI；2：USB；3：SD srcFileName：字串；來源檔案名稱(需含副檔名)，必須是 txt 或 csv 檔 fontSize：整數；8、10、12… outputPDF：整數；0：不輸出 PDF；1：輸出 PDF
範例	ret, errCode = file.ToPrinter(2, "file001.txt", 10, 1)
範例說明	將儲存於 USB 中名稱為 file001.txt 的檔案列印，並且產生 pdf 文件至 USB，字體大小為 10。 註： 1. 必須先設置印表機，並且確定印表機與人機有連線。 2. 印表機設定方式請參考 DOPSoft 手冊第 26 章。
回傳值	ret：整數；成功：1；失敗：0 errCode：整數；錯誤號碼

■ file.ListExternal：取得存於外部裝置的文件清單

指令名稱	file.ListExternal
指令運算式	ret, nameList = **file.ListExternal**(disk_id, sub_dir)
參數定義	disk_id：整數；2：USB；3：SD sub_dir：字串；子目錄；可以是 nil
範例	ret, nameList = file.ListExternal(2,"file1/file2")
範例說明	取得儲存於 USB 路徑/ file1/file2 下的文件清單，nameList 為清單， nameList[1]為第一筆檔案名稱，nameList[2]為第二筆檔案名稱，依此類推。
回傳值	ret：整數；成功：1；失敗或無檔案：0 nameList：陣列 table；成功：檔案名稱清單；失敗：nil

■ file.Exist：確認檔案是否存在

指令名稱	file.Exist
指令運算式	ret, err, errText = **file.Exist**(disk_id, file_name)
參數定義	disk_id：整數；0：HMI；2：USB；3：SD file_name：字串；檔案名稱
範例	ret, err, errText = file.Exist(0, "myFile001")
範例說明	確認人機內 myFile001.txt 檔案是否存在。
回傳值	ret：整數；成功：1；失敗：0 err：整數；錯誤號碼 errText：字串；錯誤描述

■ file.PDFToPrinter：列印 PDF 文件

指令名稱	file.PDFToPrinter
指令運算式	ret, errCode = **file.PDFToPrinter**(disk_id, srcFileName)
參數定義	disk_id：整數；0：HMI；2：USB；3：SD srcFileName：字串；來源 PDF 檔案名稱
範例	ret, errCode = file.PDFToPrinter(2, "file001.pdf")
範例說明	將儲存於 USB 檔名為"file001.pdf"的檔案列印。 註：列印前，請先確定人機是否和印表機連線。
回傳值	ret：整數；成功：1；失敗：0 errCode：整數；錯誤號碼

■ file.Copy：複製檔案

指令名稱	file.Copy
指令運算式	ret, errCode = **file.Copy**(src_name, src_disk_id, dest_name, dest_disk_id)
參數定義	src_name：字串；來源檔案名稱 src_disk_id：整數；0：HMI；2：USB；3：SD dest_name：字串；目的地檔案名稱 dest_disk_id：整數；0：HMI；2：USB；3：SD
範例	ret, err = file.Copy("myFile001.txt", 0, "myFile001Out.txt", 2)
範例說明	將儲存於 HMI 的 myFile001.txt 複製到 USB，檔名為 myFile001Out.txt。 註：如目的位址含有相同檔名，會覆寫檔案。
回傳值	ret：整數；成功：1；失敗：0 err：整數；錯誤號碼

回傳值	說明
0	無錯誤
-1	不合法參數
-106	指定磁碟未備妥
-110	指定檔名不存在
-111	無法複製檔案
-136	不合法檔案路徑

■ file.Move：移動檔案

指令名稱	file.Move
指令運算式	ret, errCode = **file.Move**(src_name, src_disk_id, dest_name, dest_disk_id)
參數定義	src_name：字串；來源檔案名稱 src_disk_id：整數；0：HMI；2：USB；3：SD dest_name：字串；目的地檔案名稱 dest_disk_id：整數；0：HMI；2：USB；3：SD
範例	ret = file.Move("myFile001.txt", 0, "myFile001Out.txt", 2)
範例說明	將儲存於 HMI 的 myFile001.txt 移動至 USB，檔名為 myFile001Out.txt。 註：如目的位址含有相同檔名，會覆寫檔案。
回傳值	ret：整數；成功：1；失敗：0 err：整數；錯誤號碼

回傳值	說明
0	無錯誤
-1	不合法參數
-106	指定磁碟未備妥
-110	指定檔名路徑不存在
-111	無法移動檔案
-136	不合法檔案路徑

範例 (透過 file 交握兩台人機資料)

■ 人機交握概念如下，於一號人機透過 file 指令建立 txt 文檔、將內部暫存器位址資料寫入 txt 文檔並匯出至 USB，再插入二號人機，由二號人機透過 file 指令讀取相關參數並寫入內部暫存器。

■ 建立 Lua 程序，如下所示。

建立 Lua 程序

1 號人機 Lua 程序：

```
disk_id = 0
file_name = " DELTA.txt"
ret, fileHandle = file.Open(disk_id, file_name)

while true do

    if mem.inter.ReadBit(0,0)==1 then
        mem.inter.WriteAscii(1,"posheng",7)
        mem.inter.WriteBit(0,0,0)
    end
    if mem.inter.ReadBit(0,1)==1 then
        str_100w=mem.inter.ReadAscii(1,7)
        file.Write(fileHandle,str_100w,string.len(str_100w))
        file.Close(fileHandle)
```

範例 (透過 file 交握兩台人機資料)

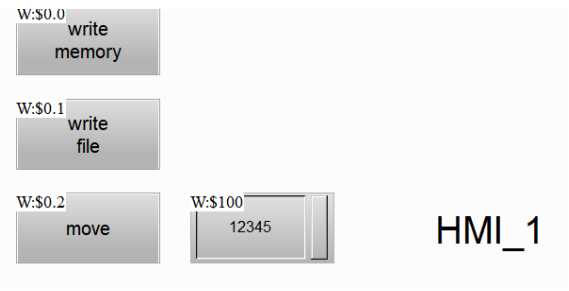
```
            mem.inter.WriteBit(0,1,0)
        end

        if mem.inter.ReadBit(0,2)==1 then
            SrcDiskNo = 0
            DestDiskNo = 2
            ret = file.Move("DELTA.txt", SrcDiskNo, " DELTA.txt", DestDiskNo)
            mem.inter.Write(100,ret)
            mem.inter.WriteBit(0,2,0)
        end

    end
```

2 號人機 Lua 程序：

```
    while true do

        if mem.inter.ReadBit(0,3)==1 then
            disk_id = 2
            file_name = " DELTA.txt"
            ret, fileHandle = file.Open(disk_id, file_name)
            mem.inter.WriteBit(0,3,0)
        end

        if mem.inter.ReadBit(0,4)==1 then
            ret, len = file.Length(fileHandle)
            ret, data = file.Read(fileHandle, len)
            mem.inter.WriteAscii(1000,data,string.len(data))
            mem.inter.WriteBit(0,4,0)
        end

    end
```

建立元件

■ 於 1 號人機建立 3 個交替型按鈕，寫入記憶體位址分別為$0.0、$0.1、$0.2，和一個數值輸入元件，寫入記憶體位址為$100。



■ 於 2 號人機建立 2 個交替型按鈕，寫入記憶體位址分別為$0.3、$0.4，和一個文數字輸入元件，寫入記憶體位址為$1000。

範例 (透過 file 交握兩台人機資料)

執行結果

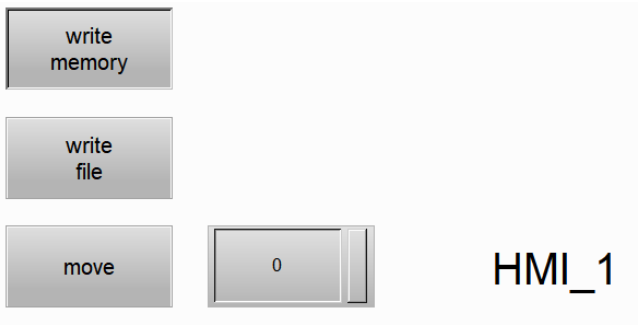
■ 完成元件建立後分別載入至人機。

■ 於第一台人機開機時，人機會在內部建立”DELTA.txt”文檔。

```
disk_id = 0
file_name = "DELTA.txt"
ret, fileHandle = file.Open(disk_id, file_name)
```

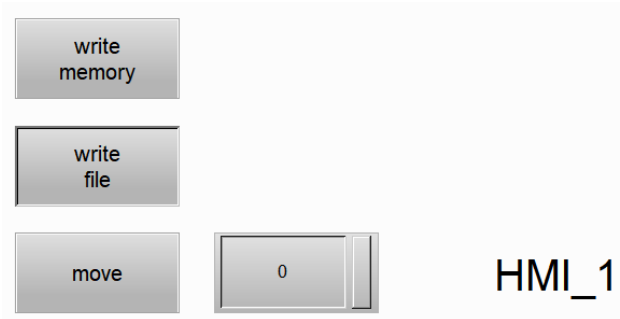
■ 按下$0.0 後，會將”posheng”字串寫入至內部暫存器$1。

```
if mem.inter.ReadBit(0,0)==1 then
    mem.inter.WriteAscii(1,"posheng",7)
    mem.inter.WriteBit(0,0,0)
end
```

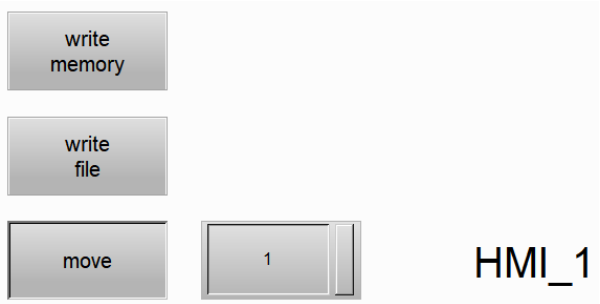


■ 按下$0.1 後，會以 7 個 bytes 的字串長度讀取$1 資料，然後將資料寫入至 DELTA.txt 檔案。

```
if mem.inter.ReadBit(0,1)==1 then
    str_100w=mem.inter.ReadAscii(1,7)
    file.Write(fileHandle,str_100w,string.len(str_100w))
    file.Close(fileHandle)
    mem.inter.WriteBit(0,1,0)
end
```



範例 (透過 file 交握兩台人機資料)	

執行結果	■ 按下$0.2 後，會將儲存於 HMI 內部的 DELTA.txt 檔案移動至 USB。
	`if mem.inter.ReadBit(0,2)==1 then`
	`  SrcDiskNo = 0`
	`  DestDiskNo = 2`
	`  ret = file.Move("DELTA.txt", SrcDiskNo, " DELTA.txt", DestDiskNo)`
	`  mem.inter.Write(100,ret)`
	`  mem.inter.WriteBit(0,2,0)`
	`end`
	 
	■ 此時已完成將資料儲存於 USB，完成儲存後將 USB 插入 2 號人機。
	■ 於 2 號人機，按下$0.3 後，會開啟”DELTA.txt”文檔。
	`if mem.inter.ReadBit(0,3)==1 then`
	`  disk_id = 2`
	`  file_name = " DELTA.txt"`
	`  ret, fileHandle = file.Open(disk_id, file_name)`
	`  mem.inter.WriteBit(0,3,0)`
	`end`
	 
	■ 按下$0.4 後，會讀取”DELTA.txt”，並將資料寫入$1000，並可得到資料為”posheng”。
	`if mem.inter.ReadBit(0,4)==1 then`
	`  ret, len = file.Length(fileHandle)`
	`  ret, data = file.Read(fileHandle, len)`
	`  mem.inter.WriteAscii(1000,data,string.len(data))`
	`  mem.inter.WriteBit(0,4,0)`
	`end`
	 

4.6 fileSlot (檔案存取)

此指令可供使用者對文件做讀、寫、匯出、製作 pdf 檔的動作，基本語法包含：

指令	指令運算式	說明
FileSlot (檔案讀取/寫入/匯出)	fileslot.Read	fileslot 檔案讀取
	fileslot.Write	fileslot 檔案寫入
	fileslot.ReadValue	讀取 fileslot 數值內容
	fileslot.WriteValue	寫入數值至 fileslot
	fileslot.GetLength	取得 fileslot 內容長度
	fileslot.Remove	移除 fileslot
	fileslot.Import	fileslot 檔案匯入
	fileslot.Export	fileslot 檔案匯出
	fileslot.SetName	設定 fileslot 檔案名稱
	fileslot.GetName	取得 fileslot 檔案名稱
	fileslot.GetID	取得 fileslot 檔案 ID

以下將一一詳細介紹。

■ fileslot.Read：fileslot 檔案讀取

指令名稱	fileslot.Read
指令運算式	ret, err = **fileslot.Read**(FS_ID, FS_Pos, Device, MemIdx, length)
參數定義	FS_ID：無號整數；FileSlot ID FS_Pos：無號整數；FileSlot 讀取位置 Device：無號整數；1：內部暫存器($)；2：斷電保持記憶體($M) MemIdx：無號整數；裝置讀寫位址 length：無號整數；資料長度
範例	ret, err = fileslot.Read(1, 0, 1, 20, 5)
範例說明	將ID 1的FileSlot內容從0的位置開始的5個Word資料，讀取至目的位址$20。
回傳值	ret：整數；成功：1；失敗：0 err：整數；錯誤號碼

回傳值	說明
1	無錯誤
-1	不合法參數
-13	資料讀取錯誤
-14	資料寫入錯誤
-24	記憶體錯誤
-51	FileSlot ID 錯誤
-52	FileSlot 處理程序錯誤
-53	FileSlot 讀取錯誤
-54	FileSlot 查找錯誤
-55	FileSlot 寫入錯誤
-56	FileSlot 刪除錯誤
-57	FileSlot 資料長度錯誤
-62	FileSlot 操作錯誤
-63	外部文件操作錯誤
-84	FileSlot 獲取名稱錯誤
-86	FileSlot 設定名稱錯誤
-87	FileSlot 獲取 ID 錯誤
-88	FileSlot 寫入位置錯誤

■ fileslot.Write：fileslot 檔案寫入

指令名稱	fileslot.Write
指令運算式	ret, err = **fileslot.Write**(FS_ID, FS_Pos, Device, MemIdx, length)
參數定義	FS_ID：無號整數；FileSlot ID FS_Pos：無號整數；FileSlot 讀取位置 Device：無號整數；1：內部暫存器($)；2：斷電保持記憶體($M) MemIdx：無號整數；裝置讀寫位址 length：無號整數；資料長度
範例	ret, err = fileslot.Write(1, 0, 1, 10, 5)
範例說明	將$10 開始的 5 個 Word 資料，寫入至 FileSlot ID 為 1、位址為 0 的 FileSlot。
回傳值	ret：整數；成功：1；失敗：0 err：整數；錯誤號碼

回傳值	說明
1	無錯誤
-1	不合法參數
-13	資料讀取錯誤
-14	資料寫入錯誤
-24	記憶體錯誤
-51	FileSlot ID 錯誤
-52	FileSlot 處理程序錯誤
-53	FileSlot 讀取錯誤
-54	FileSlot 查找錯誤
-55	FileSlot 寫入錯誤
-56	FileSlot 刪除錯誤
-57	FileSlot 資料長度錯誤
-62	FileSlot 操作錯誤
-63	外部文件操作錯誤
-84	FileSlot 獲取名稱錯誤
-86	FileSlot 設定名稱錯誤
-87	FileSlot 獲取 ID 錯誤
-88	FileSlot 寫入位置錯誤

■ fileslot.ReadValue：讀取 fileslot 數值內容

指令名稱	fileslot.ReadValue
指令運算式	ret, err = **fileslot.ReadValue**(FS_ID, FS_Pos, Format, [value_format])
參數定義	FS_ID：無號整數；FileSlot ID FS_Pos：無號整數；FileSlot 讀取位置 Format：無號整數；2：WORD 長度；3：DWORD 長度 value_format：字串；有號數填入：“signed”，可省略
範例	ret, err = fileslot.ReadValue(1, 0, 2, "SIGNED")
範例說明	以數值單位為 Word 的有號數，讀取 FileSlot ID 為 1 位址為 0 的 FileSlot 的內容。
回傳值	ret：整數；成功：1；失敗：0 err：整數；錯誤號碼

回傳值	說明
1	無錯誤
-1	不合法參數
-13	資料讀取錯誤
-14	資料寫入錯誤
-24	記憶體錯誤
-51	FileSlot ID 錯誤
-52	FileSlot 處理程序錯誤
-53	FileSlot 讀取錯誤
-54	FileSlot 查找錯誤
-55	FileSlot 寫入錯誤
-56	FileSlot 刪除錯誤
-57	FileSlot 資料長度錯誤
-62	FileSlot 操作錯誤
-63	外部文件操作錯誤
-84	FileSlot 獲取名稱錯誤
-86	FileSlot 設定名稱錯誤
-87	FileSlot 獲取 ID 錯誤
-88	FileSlot 寫入位置錯誤

■ fileslot.WriteValue：寫入數值至 fileslot

指令名稱	fileslot.WriteValue
指令運算式	ret, err = **fileslot.WriteValue**(FS_ID, FS_Pos, Format, Value)
參數定義	FS_ID：無號整數；FileSlot ID FS_Pos：無號整數；FileSlot 讀取位置 Format：無號整數；2：WORD 長度；3：DWORD 長度 Value：無號整數；資料內容
範例	ret, err = fileslot.WriteValue(1, 0, 2, 99)
範例說明	以 Word 為數值單位，將無號整數 99 寫入至 FileSlot ID 為 1 位址為 0 的 FileSlot。
回傳值	ret：整數；成功：1；失敗：0 err：整數；錯誤號碼

回傳值	說明
1	無錯誤
-1	不合法參數
-13	資料讀取錯誤
-14	資料寫入錯誤
-24	記憶體錯誤
-51	FileSlot ID 錯誤
-52	FileSlot 處理程序錯誤
-53	FileSlot 讀取錯誤
-54	FileSlot 查找錯誤
-55	FileSlot 寫入錯誤
-56	FileSlot 刪除錯誤
-57	FileSlot 資料長度錯誤
-62	FileSlot 操作錯誤
-63	外部文件操作錯誤
-84	FileSlot 獲取名稱錯誤
-86	FileSlot 設定名稱錯誤
-87	FileSlot 獲取 ID 錯誤
-88	FileSlot 寫入位置錯誤

■ fileslot.GetLength：取得 fileslot 內容長度

指令名稱	fileslot.GetLength
指令運算式	ret, err = **fileslot.GetLength**(FS_ID)
參數定義	FS_ID：無號整數；FileSlot ID
範例	ret, err = fileslot.GetLength(1)
範例說明	取得 FileSlot ID 1 的內容長度。
回傳值	ret：整數；成功：1；失敗：0 err：整數；錯誤號碼

回傳值	說明
1	無錯誤
-1	不合法參數
-51	FileSlot ID 錯誤
-57	FileSlot 資料長度錯誤

■ fileslot.Remove：移除 fileslot

指令名稱	fileslot.Remove
指令運算式	ret, err = **fileslot.Remove**(FS_ID)
參數定義	FS_ID：無號整數；FileSlot ID
範例	ret, err = fileslot.Remove(1)
範例說明	移除 FileSlot ID 1。
回傳值	ret：整數；成功：1；失敗：0 err：整數；錯誤號碼

回傳值	說明
1	無錯誤
-1	不合法參數
-51	FileSlot ID 錯誤
-56	FileSlot 刪除錯誤

■ fileslot.Import：fileslot 檔案匯入

指令名稱	fileslot.Import
指令運算式	ret, err = **fileslot.Import**(FS_ID, DiskID, ExtFileName)
參數定義	FS_ID：無號整數；FileSlot ID DiskID：整數，磁片 ID；2：USB；3：SD ExtFileName：字串，遠端檔案名稱
範例	ret, err = fileslot.Import(1, 2, "myfile.txt")
範例說明	匯入位於 USB 且名為"myfile.txt"的文字檔到 FileSlot ID 1。
回傳值	ret：整數；成功：1；失敗：0 err：整數；錯誤號碼

回傳值	說明
1	無錯誤
-1	不合法參數
-59	FileSlot 匯出錯誤
-61	FileSlot 匯入錯誤
-62	FileSlot 操作錯誤
-63	外部文件操作錯誤
-64	FileSlot 複製錯誤
-106	外部儲存裝置未備妥
-136	外部檔案名稱錯誤

■ fileslot.Export：fileslot 檔案匯出

指令名稱	fileslot.Export
指令運算式	ret, err = **fileslot.Export**(FS_ID, DiskID, ExtFileName)
參數定義	FS_ID：無號整數；FileSlot ID DiskID：整數，磁片 ID；2：USB；3：SD ExtFileName：字串，遠端檔案名稱
範例	ret, err = fileslot.Export(10, 2, "Newfile.txt")
範例說明	匯出 FileSlot ID 10 的資料於 USB 且命名為"Newfile.txt"。
回傳值	ret：整數；成功：1；失敗：0 err：整數；錯誤號碼

回傳值	說明
1	無錯誤
-1	不合法參數
-59	FileSlot 匯出錯誤
-61	FileSlot 匯入錯誤
-62	FileSlot 操作錯誤
-63	外部文件操作錯誤
-64	FileSlot 複製錯誤
-106	外部儲存裝置未備妥
-136	外部檔案名稱錯誤

■ fileslot.SetName：設定 fileslot 檔案名稱

指令名稱	fileslot.SetName
指令運算式	et, err = **fileslot.SetName**(FS_ID, FS_Name)
參數定義	FS_ID：無號整數；FileSlot ID FS_Name：字串，檔案名稱
範例	ret, err = fileslot.SetName(1, "myfile.txt") ret, err = fileslot.Export(1, 2, "folder1/")
範例說明	將 FileSlot ID 1 命名為 myfile.txt，再將此檔案匯出至 folder1 資料夾。 註：需先於 USB 儲存裝置建立 folder1 資料夾。
回傳值	ret：整數；成功：1；失敗：0 err：整數；錯誤號碼

回傳值	說明
1	無錯誤
-1	不合法參數
-51	FileSlot ID 錯誤
-66	FileSlot 設定名稱操作失敗

■ fileslot.GetName：取得 fileslot 檔案名稱

指令名稱	fileslot.GetName
指令運算式	ret, err = **fileslot.GetName**(FS_ID)
參數定義	FS_ID：無號整數；FileSlot ID
範例	ret, err = fileslot.GetName(5)
範例說明	取得 FileSlot ID 5 的名稱。
回傳值	ret：整數；成功：檔案名稱字串；失敗：0 err：整數；錯誤號碼

回傳值	說明
1	無錯誤
-1	不合法參數
-51	FileSlot ID 錯誤
-84	FileSlot 取得名稱操作失敗

■ fileslot.GetID：取得 fileslot 檔案 ID

指令名稱	fileslot.GetID
指令運算式	ret, err = **fileslot.GetID**(FS_Name)
參數定義	FS_Name：字串，檔案名稱
範例	ret, err = fileslot.SetName(1, "myfile") ret, err = fileslot.GetID("myfile")
範例說明	將 FileSlot ID 1 命名為 myfile，取得名為 myfile 的 FileSlot ID，結果為 ID 1。
回傳值	ret：整數；成功：對應的 FileSlot ID；不存在或失敗：0 err：整數；錯誤號碼

回傳值	說明
1	無錯誤
-1	不合法參數
-51	FileSlot ID 錯誤
-87	FileSlot 取得 FileSlot ID 操作失敗

4.7 FTP Client (FTP 傳輸功能)

此指令可供使用者執行 FTP 上下載的功能，語法包含：

指令	指令運算式	說明
FTP 傳輸功能	ftpc.Download	FTP 下載
	ftpc.Upload	FTP 上傳

以下將一一詳細介紹。

■ ftpc.Download：FTP 下載

指令名稱	ftpc.Download
指令運算式	ret, err = **ftpc.Download**(IPAddress, Port, UserName, Password, LocalFileName, RemoteFileName, OverWrite)
參數定義	IPAddress：字串，IP 位址 Port：無號整數，Port number，1 ~ 65535 UserName：字串，登入帳號，若不需帳號則設為""(匿名登入) Password：字串，登入密碼，若不需密碼則設為"" LocalFileName：字串，本地檔案名稱，使用"/HMI/filename" 或"/FILESLOT/FS_ID" 來選擇在人機或 FileSlot 的檔案 RemoteFileName：字串，遠端檔案名稱 OverWrite：無號整數，0：若檔案存在則不覆寫；1：若檔案存在則覆寫
範例	IPAddress = "192.168.123.144" Port = 21 UserName = "chen" Password = "123" LocalFileName = "/FILESLOT/1" RemoteFileName = "delta.txt" OverWrite = 0 ret, err = ftpc.Download(IPAddress, Port, UserName, Password, LocalFileName, RemoteFileName, OverWrite)
範例說明	以帳號"chen"密碼"123"透過 IP 位址 192.168.123.144 的 port 21 登入 FTP，並下載"delta.txt"檔案寫入至 FileSlot ID 1 的空間內。
回傳值	ret：整數；成功：1；失敗：0 err：整數；錯誤號碼

回傳值	說明
0	無錯誤
-10	不合法參數(IP 位址)
-11	不合法參數(Port)
-12	不合法參數(登入帳號)
-13	不合法參數(登入密碼)
-14	不合法參數(本地檔案名稱)
-15	不合法參數(遠端檔案名稱)
-16	不合法參數(覆寫參數)
-20	登入失敗
-21	登出失敗

	-22	設置目錄失敗
	-23	獲取遠程目錄資訊失敗
	-24	本地文件不存在
	-25	本地文件存在
	-26	遠程文件不存在
	-27	遠程文件存在
	-28	下載文件失敗
	-29	上傳失敗
	-41	FileSlot 名稱重複
	-42	FileSlot ID 錯誤
	-43	FileSlot 沒有足夠的空間

■ ftpc.Upload：FTP 上傳

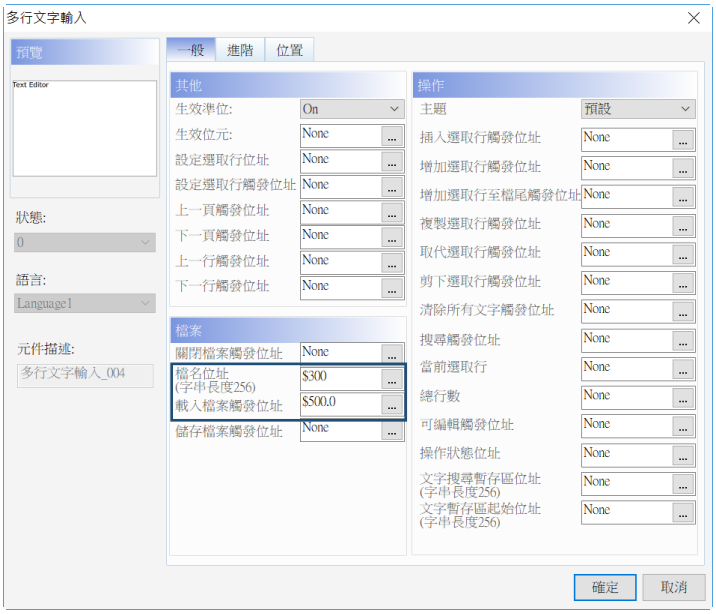
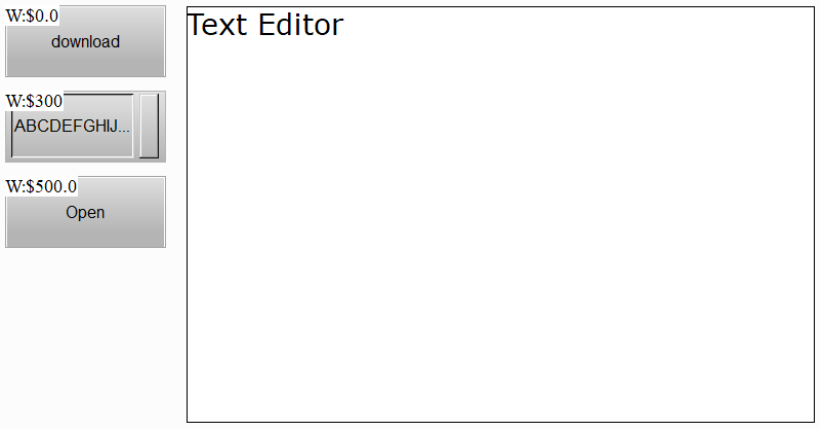
指令名稱	ftpc.Upload
指令運算式	ret, err = **ftpc.Upload**(IPAddress, Port, UserName, Password, LocalFileName, RemoteFileName, OverWrite)
參數定義	IPAddress：字串，IP 位址 Port：無號整數，Port number，1 ~ 65535 UserName：字串，登入帳號，若不需帳號則設為""(匿名登入) Password：字串，登入密碼，若不需密碼則設為"" LocalFileName：字串，本地檔案名稱，使用"/HMI/filename" 或"/FILESLOT/FS_ID" 來選擇在人機或 FileSlot 的檔案 RemoteFileName：字串，遠端檔案名稱 OverWrite：無號整數，0：若檔案存在則不覆寫；1：若檔案存在則覆寫
範例	IPAddress = "192.168.123.144" Port = 21 UserName = "chen" Password = "123" LocalFileName = "/FILESLOT/1" RemoteFileName = "delta.txt" OverWrite = 0 ret, err = ftpc.Upload(IPAddress, Port, UserName, Password, LocalFileName, RemoteFileName, OverWrite)
範例說明	以帳號"chen"密碼"123"透過 IP 位址 192.168.123.144 的 port 21 登入 FTP， 並上傳 FileSlot ID 1 的內容至 FTP Server 目錄，並以"delta.txt"命名。
回傳值	ret：整數；成功：1；失敗：0 err：整數；錯誤號碼

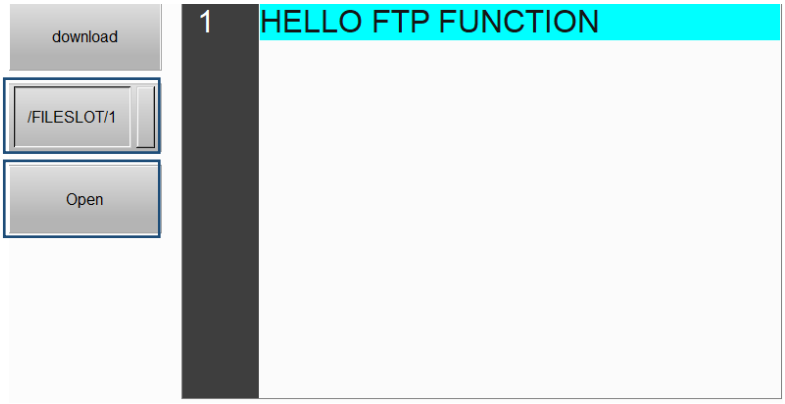
回傳值	說明
0	無錯誤
-10	不合法參數(IP 位址)
-11	不合法參數(Port)
-12	不合法參數(登入帳號)
-13	不合法參數(登入密碼)
-14	不合法參數(本地檔案名稱)
-15	不合法參數(遠端檔案名稱)
-16	不合法參數(覆寫參數)
-20	登入失敗

	-21	登出失敗
	-22	設置目錄失敗
	-23	獲取遠程目錄資訊失敗
	-24	本地文件不存在
	-25	本地文件存在
	-26	遠程文件不存在
	-27	遠程文件存在
	-28	下載文件失敗
	-29	上傳失敗
	-41	FileSlot 名稱重複
	-42	FileSlot ID 錯誤
	-43	FileSlot 沒有足夠的空間

範例 (透過 ftp 指令上傳/下載檔案)	

建立 Lua 程序	建立 Lua 程序：
	`if mem.inter.ReadBit(0,0)==1 then`
	`IPAddress = "192.168.123.144"`
	`Port = 21`
	`UserName = "chen"`
	`Password = ""`
	`LocalFileName = "/FILESLOT/1"`
	`RemoteFileName = "delta.txt"`
	`OverWrite = 0`
	`ret, err = ftpc.Download(IPAddress, Port, UserName, Password, LocalFileName, RemoteFileName, OverWrite)`
	`mem.inter.WriteBit(0,0,0)`
	`end`

建立元件	■ 建立 2 個交替型按鈕，寫入記憶體位址分別為$0.0、$500.0。 ■ 建立 1 個文數字輸入元件，寫入記憶體位址設定為$300。 ■ 建立 1 個多行文字輸入元件，檔案位址設定為$300，載入檔案觸發位址設定為$500.0。  

範例 (透過 ftp 指令上傳/下載檔案)	
執行結果	■ 下載第三方 FTP Server 軟體，透過軟體建立 FTP Server 目錄。 ■ 建立一份記事本，並命名為”delta.txt”，放置上述 FTP Server 目錄。 ■ 將專案下載至人機，觸發$0.0，將”delta.txt”下載至 fileslot ID 1。 if mem.inter.ReadBit(0,0)==1 then IPAddress = "192.168.123.144" Port = 21 UserName = "chen" Password = "" LocalFileName = "/FILESLOT/1" RemoteFileName = "delta.txt" OverWrite = 0 ret, err = ftpc.Download(IPAddress, Port, UserName, Password, LocalFileName, RemoteFileName, OverWrite) mem.inter.WriteBit(0,0,0) end ■ 於$300 輸入"/FILESLOT/1"，並觸發$500.0，如此便可以把 FileSlot ID 1 的內容顯示於多行文字輸入元件。 

4.8 Math (數學運算)

此指令可供使用者進行數學運算，語法包含：

指令	指令運算式	說明
Math (數學運算)	math.abs	取得數字的絕對值
	math.exp	取得以 e 為底數的指數函數數值
	math.log	取得對數數值
	math.sin	取得正弦數值
	math.sinh	取得雙曲線正弦數值
	math.cos	取得餘弦數值
	math.cosh	取得雙曲線餘弦數值
	math.tan	取得正切數值
	math.tanh	取得雙曲線正切數值
	math.asin	取得反正弦數值
	math.acos	取得反餘弦數值
	math.atan	取得反正切數值
	math.atan2	取得反正切數值 (兩個參數)
	math.deg	取得弧度對應的角度
	math.rad	取得角度對應的弧度
	math.min	取得最小值
	math.max	取得最大值
	math.modf	分割數值為整數和小數
	math.pi	圓周率 π
	math.pow	取得次方數值
	math.randomseed	亂數種子
	math.random	取得隨機數
	math.sqrt	取得開根號數值

以下將一一詳細介紹。

■ math.abs：取得數字的絕對值

指令名稱	math.abs
指令運算式	value = **math.abs**(value_number)
參數定義	value_number：有號整數值
範例	v = math.abs(-123)
範例說明	取得-123 的絕對值，v=123。
回傳值	v：無號整數值

■ math.exp：取得以 e 為底數的指數函數數值

指令名稱	math.exp
指令運算式	value = **math.exp**(x)
參數定義	x：次方數
範例	v = math.exp(1)
範例說明	取得自然指數的 1 次方數值，v=2.718281828459。
回傳值	v：自然指數的 x 次方值

■ math.log：取得對數數值

指令名稱	math.log
指令運算式	value= **math.log**(x, base)
參數定義	x：數值 base：基底數；若不輸入則為自然指數
範例	v = math.log(1000, 10)
範例說明	計算以 10 為底數，1000 的對數為 v，v=3。
回傳值	v：對數

■ math.sin：取得正弦數值

指令名稱	math.sin
指令運算式	value = **math.sin**(radian)
參數定義	radian：浮點數；弧度
範例	v = math.sin(math.rad(30))
範例說明	v 為 sin30°數值，v=0.499999。
回傳值	v：浮點數

■ math.sinh：取得雙曲線正弦數值

指令名稱	math.sinh
指令運算式	value = **math.sinh**(tangenhValue)
參數定義	tangenhValue：數字；tangens hiperboliczny values
範例	v = math.sinh(2)
範例說明	取得 sinh(2)數值，v=3.62860。
回傳值	v：雙曲函數數值

■ math.cos：取得餘弦數值

指令名稱	math.cos
指令運算式	value = **math.cos**(radian)
參數定義	radian：浮點數；弧度
範例	v = math.cos(math.rad(30))
範例說明	取得 cos30°數值，v=0.866025。
回傳值	v：浮點數

■ math.cosh：取得雙曲線餘弦數值

指令名稱	math.cosh
指令運算式	value = **math.cosh**(tangenhValue)
參數定義	tangenhValue：數字；tangens hiperboliczny values
範例	v = math.cosh(2)
範例說明	取得 cosh(2)數值，v=3.7621956910。
回傳值	v：雙曲函數數值

■ math.tan：取得正切數值

指令名稱	math.tan
指令運算式	value = **math.tan**(radian)
參數定義	radian：浮點數；弧度
範例	v = math.tan(math.rad(30))
範例說明	取得 tan30°數值，v=0.599350。
回傳值	v：浮點數

■ math.tanh：取得雙曲線正切數值

指令名稱	math.tanh
指令運算式	value = **math.tanh**(tangenhValue)
參數定義	tangenhValue：數字；tangens hiperboliczny values
範例	v = math.tanh(2)
範例說明	取得 tanh(2)數值，v=0.964027580。
回傳值	v：雙曲函數數值

■ math.asin：取得反正弦數值

指令名稱	math.asin
指令運算式	value = **math.asin**(radian)
參數定義	radian：浮點數；弧度
範例	v = math.asin(math.rad(30))
範例說明	取得 arcsin 30°數值，v=0.551069。
回傳值	v：浮點數

■ math.acos：取得反餘弦數值

指令名稱	math.acos
指令運算式	value = **math.acos**(radian)
參數定義	radian：浮點數；弧度
範例	v = math.acos(math.rad(30))
範例說明	取得 arccos 30°數值，v=1.019726。
回傳值	v：浮點數

■ math.atan：取得反正切數值

指令名稱	math.atan
指令運算式	value = **math.atan**(radian)
參數定義	radian：浮點數；弧度
範例	v = math.atan(math.rad(30))
範例說明	取得 arctan 30°數值，v=0.482347。
回傳值	v：浮點數

■ math.atan2：取得反正切數值 (兩個參數)

指令名稱	math.atan2
指令運算式	value = **math.atan2**(yRadian, xRadian)
參數定義	yRadian：數字；y 軸長度 xRadian：數字；x 軸長度，可以為 0
範例	v = math.atan2(math.rad(30), math.rad(60))
範例說明	取得 atan2 (30°, 60°)數值，v=0.463647609000806。 註：atan2 (30°, 60°)=atan(30°/60°)。
回傳值	v：浮點數

■ math.deg：取得弧度對應的角度

指令名稱	math.deg
指令運算式	angle = **math.deg**(radian)
參數定義	radian：浮點數；弧度
範例	v = math.deg(math.pi)
範例說明	將π轉為角度，v=180。
回傳值	v：浮點數，角度

■ math.rad：取得角度對應的弧度

指令名稱	math.rad
指令運算式	radian = **math.rad**(angle)
參數定義	angle：浮點數；角度
範例	v = math.rad(180)
範例說明	將π轉為弧度，v=3.14159265。
回傳值	v：正數

■ math.min：取得最小值

指令名稱	math.min
指令運算式	min_value = **math.min**(v1, v2)
參數定義	v1, v2：浮點數；數值
範例	v = math.min(-2, 3.6)
範例說明	取得-2、3.6 的最小值，v=-2。
回傳值	v：浮點數；最小數值

■ math.max：取得最大值

指令名稱	math.max
指令運算式	max_value = **math.max**(v1, v2)
參數定義	v1, v2：浮點數；數值
範例	v = math.max(-2, 3.6)
範例說明	取得-2、3.6 的最大值，v=3.6。
回傳值	v：浮點數；最大數值

■ math.modf：分割數值為整數和小數

指令名稱	math.modf
指令運算式	Integer, fraction = **math.modf**(v1)
參數定義	v1：浮點數
範例	v1, v2 = math.modf(3.6)
範例說明	分割數值 3.6 為整數和分數，v1=3，v2=0.6。
回傳值	v1：整數值 v2：小數值

■ math.pi：圓周率 π

指令名稱	math.pi
指令運算式	value = **math.pi**
參數定義	無參數
範例	v = math.pi
範例說明	v=3.1415926535898。
回傳值	v：圓周率π

■ math.pow：取得次方數值

指令名稱	math.pow
指令運算式	value = **math.pow**(x, y)
參數定義	x：基數 y：次方數
範例	v = math.pow(2, 3)
範例說明	取得 2 的 3 次方數值，v=8。
回傳值	v：x 的 y 次方

■ math.randomseed：亂數種子 (搭配 math.random 指令使用)

指令名稱	math.randomseed
指令運算式	**math.randomseed**(seedValue)
參數定義	seedValue：初始亂數種子，此變數可以設定為時間
範例	math.randomseed(sys.GetTick()) v = math.random(0, 1000)
範例說明	■ 因為 random 函式初始時會以一個亂數種子(Random Seed)來產生亂數，在未改變這個亂數種子之前，亂數會以一定的順序產生，使用者可以先利用 randomSeed 函式指定一個亂數種子，再使用 random 來產生亂數。 ■ v 為從 0 ~ 1000 中隨機產生的任一整數。
回傳值	v：整數

■ math.random：取得隨機數

指令名稱	math.random
指令運算式	value = **math.random**(lower, upper)
參數定義	Lower：上限 Upper：下限
範例	math.randomseed(sys.GetTick()) v = math.random(0, 1000)
範例說明	■ 因為 random 函式初始時會以一個亂數種子(Random Seed)來產生亂數，在未改變這個亂數種子之前，亂數會以一定的順序產生，使用者可以先利用 randomSeed 函式指定一個亂數種子，再使用 random 來產生亂數。 ■ v 為從 0 ~ 1000 中隨機產生的任一整數。
回傳值	v：整數

■ math.sqrt：取得開根號數值

指令名稱	math.sqrt
指令運算式	sqrt = **math.sqrt**(nSquare)
參數定義	NSquare：欲開根號之數值
範例	v = math.sqrt(100)
範例說明	取得 100 開根號數值，v=10。
回傳值	v：平方根

4.9 Recipe (配方)

此指令可供使用者對配方做讀寫的動作，配方有包含 16 位元配方、32 位元配方、加強型配方。使用者如需指定 16 位元配方，則該配方群組為 0，如需使用 32 位元配方時，該配方群組為 1 至最大 255 組，Lua 的配方語法包含：

指令	指令運算式	說明
Recipe (配方)	recipe.GetCurRcpNoIndex	取得當前配方組別索引
	recipe.GetCurRcpGIndex	取得當前配方群組索引
	recipe.GetRcpWord	取得指定配方位址的數值 (Word)
	recipe.GetRcpDWord	取得指定配方位址的數值 (Double Word)
	recipe.GetRcpFloat	取得指定配方位址的數值 (浮點數)
	recipe.GetCurEnRcpNoName	取得當前加強型配方組別索引名稱
	recipe.GetCurEnRcpGName	取得當前加強型配方群組索引名稱
	recipe.GetCurEnRcpNoIndex	取得當前加強型配方組別索引
	recipe.GetCurEnRcpGIndex	取得當前加強型配方群組索引
	recipe.GetEnRcpWord	取得指定加強型配方位址的數值(Word)
	recipe.GetEnRcpDWord	取得指定加強型配方位址的數值 (Double Word)
	recipe.GetEnRcpFloat	取得指定加強型配方位址的數值(Float)
	recipe.GetEnRcpAscii	取得指定加強型配方位址的字串
	recipe.SetRcpWord	設定參數至配方位址(Word)
	recipe.SetRcpDWord	設定參數至配方位址(Double Word)
	recipe.SetRcpFloat	設定參數至配方位址(Float)
	recipe.SetCurEnRcpNoName	設定加強型配方組別名稱
	recipe.SetCurEnRcpGName	設定加強型配方群組名稱
	recipe.SetEnRcpWord	設定參數至加強型配方位址 (Word)
	recipe.SetEnRcpDWord	設定參數至加強型配方位址 (Double Word)
	recipe.SetEnRcpFloat	設定參數至加強型配方位址 (Float)
	recipe.SetEnRcpAscii	設定字串至加強型配方位址
	recipe.ChangeRcpNoIndex	更改配方組別索引
	recipe.ChangeRcpGIndex	更改配方群組索引
	recipe.ChangeEnRcpNoIndex	更改加強型配方組別索引
	recipe.ChangeEnRcpGIndex	更改加強型配方群組索引
	recipe.SetEnRcpDouble	設定參數至加強型配方位址(雙精度浮點數)
	recipe.GetEnRcpDouble	取得指定加強型配方位址的數值(雙精度浮點數)

以下將一一詳細介紹。

■ recipe.GetCurRcpNoIndex：取得當前配方組別索引

指令名稱	recipe.GetCurRcpNoIndex
指令運算式	ret, noIdx = **recipe.GetCurRcpNoIndex**()
參數定義	無參數
範例	ret, noIdx = recipe.GetCurRcpNoIndex()
範例說明	取得當前配方組別索引。
回傳值	ret：成功回傳 1；失敗回傳 0 noIdx：整數；配方組別名稱

■ recipe.GetCurRcpGIndex：取得當前配方群組索引

指令名稱	recipe.GetCurRcpGIndex
指令運算式	ret, gIdx = **recipe.GetCurRcpGIndex**()
參數定義	無參數
範例	ret, gIdx = recipe.GetCurRcpGIndex()
範例說明	取得當前配方群組索引。
回傳值	ret：成功回傳 1；失敗回傳 0 gIdx：整數，配方群組名稱

■ recipe.GetRcpWord：取得指定配方位址的數值 (Word)。

指令名稱	recipe.GetRcpWord
指令運算式	ret, value = **recipe.GetRcpWord**(index)
參數定義	index：整數，配方索引
範例	ret, value = recipe.GetRcpWord(1)
範例說明	取得指定配方位址(RCP1)的數值，單位為無符號的 Word。
回傳值	ret：成功回傳 1；失敗回傳 0 value：整數；配方無號數，單位：Word

■ recipe.GetRcpDWord：取得指定配方位址的數值 (Double Word)

指令名稱	recipe.GetRcpDWord
指令運算式	ret, value = **recipe.GetRcpDWord**(index, [value_format])
參數定義	Index：整數；配方索引 value_format：格式字串；有號數填入“signed”，可省略
範例	ret, value = recipe.GetRcpDWord(1)
範例說明	取得指定配方位址(RCP1)的數值，單位為 Double Word。
回傳值	ret：成功回傳 1；失敗回傳 0 value：整數；配方，單位：Double Word

■ recipe.GetRcpFloat：取得指定配方位址的數值 (Float)

指令名稱	recipe.GetRcpFloat
指令運算式	ret, value = **recipe.GetRcpFloat**(index)
參數定義	Index：整數；配方索引
範例	ret, value = recipe.GetRcpFloat(1)
範例說明	取得指定配方位址(RCP1)的數值，單位為 Float。
回傳值	ret：成功回傳 1；失敗回傳 0 value：浮點數；配方 32 位元浮點數

■ recipe.GetCurEnRcpNoName：取得當前加強型配方組別索引名稱

指令名稱	recipe.GetCurEnRcpNoName
指令運算式	ret, noName = **recipe.GetCurEnRcpNoName**()
參數定義	無參數
範例	ret, noName = recipe.GetCurEnRcpNoName()
範例說明	取得當前加強型配方組別索引名稱。
回傳值	ret：成功回傳 1；失敗回傳 0 noName：字串，當前作用的加強型配方組別名稱

■ recipe.GetCurEnRcpGName：取得當前加強型配方群組索引名稱

指令名稱	recipe.GetCurEnRcpGName
指令運算式	ret, gName = **recipe.GetCurEnRcpGName**()
參數定義	無參數
範例	ret, gName = recipe.GetCurEnRcpGName()
範例說明	取得當前加強型配方群組索引名稱。
回傳值	ret：成功回傳 1；失敗回傳 0 gName：字串；當前作用中的加強型配方群組名稱

■ recipe.GetCurEnRcpNoIndex：取得當前加強型配方組別索引

指令名稱	recipe.GetCurEnRcpNoIndex
指令運算式	ret, noIdx = **recipe.GetCurEnRcpNoIndex**()
參數定義	無參數
範例	ret, noIdx = recipe.GetCurEnRcpNoIndex()
範例說明	取得當前加強型配方組別索引。
回傳值	ret：成功回傳 1；失敗回傳 0 noIdx：整數；當前作用中的加強型配方組別索引

■ recipe.GetCurEnRcpGIndex：取得當前加強型配方群組索引

指令名稱	recipe.GetCurEnRcpGIndex
指令運算式	ret, gIdx = **recipe.GetCurEnRcpGIndex**()
參數定義	無參數
範例	ret, gIdx = recipe.GetCurEnRcpGIndex()
範例說明	取得當前加強型配方群組索引。
回傳值	ret：成功回傳 1；失敗回傳 0 gIdx：整數；當前作用中的加強型配方群組索引

■ recipe.GetEnRcpWord：取得指定加強型配方位址的數值 (Word)

指令名稱	recipe.GetEnRcpWord
指令運算式	ret, value = **recipe.GetEnRcpWord**(index)
參數定義	Index：整數；加強配方索引
範例	ret, value = recipe.GetEnRcpWord(2)
範例說明	取得指定加強型配方位址(ENRCP2)的數值，單位為無符號的 Word。
回傳值	ret：成功回傳 1；失敗回傳 0 value：整數；加強型配方 WORD

■ recipe.GetEnRcpDWord：取得指定加強型配方位址的數值 (Double Word)

指令名稱	recipe.GetEnRcpDWord
指令運算式	ret, value = **recipe.GetEnRcpDWord**(index, [value_format])
參數定義	Index：整數，加強配方索引 value_format：格式字串；有號數填入：“signed”，可省略
範例	ret, value = recipe.GetEnRcpDWord(2)
範例說明	取得指定加強型配方位址(ENRCP2)的數值，單位為 Double Word。
回傳值	ret：成功回傳 1；失敗回傳 0

■ recipe.GetEnRcpFloat：取得指定加強型配方位址的數值 (Float)

指令名稱	recipe.GetEnRcpFloat
指令運算式	ret, value = **recipe.GetEnRcpFloat**(index)
參數定義	Index：整數，加強配方索引
範例	ret, value = recipe.GetEnRcpFloat(2)
範例說明	取得指定加強型配方位址(ENRCP2)的數值，單位為 Float。
回傳值	ret：成功回傳 1；失敗回傳 0 value：浮點數；加強型配方 32 位元浮點數

■ recipe.GetEnRcpAscii：取得指定加強型配方位址的字串

指令名稱	recipe.GetEnRcpAscii
指令運算式	ret, str = **recipe.GetEnRcpAscii**(index)
參數定義	Index：整數；加強配方索引
範例	ret, str = recipe.GetEnRcpAscii(2)
範例說明	取得指定加強型配方指定位址(ENRCP2)的字串。
回傳值	ret：成功回傳 1；失敗回傳 0 str：字串；加強型配方字串

■ recipe.SetRcpWord：設定參數至配方位址 (Word)

指令名稱	recipe.SetRcpWord
指令運算式	ret = **recipe.SetRcpWord**(index, word)
參數定義	Index：整數；配方索引 Word：整數；配方無號數 WORD
範例	ret = recipe.SetRcpWord(1, 5)
範例說明	設定 5 至配方位址 RCP1。
回傳值	ret：成功回傳 1；失敗回傳 0

■ recipe.SetRcpDWord：設定參數至配方位址 (Double Word)

指令名稱	recipe.SetRcpDWord
指令運算式	ret = **recipe.SetRcpDWord**(index, dword)
參數定義	Index：整數，配方索引 dword：整數，配方 DWORD
範例	ret = recipe.SetRcpDWord(1, 65536)
範例說明	設定 65536 至配方位址 RCP1。
回傳值	ret：成功回傳 1；失敗回傳 0

■ recipe.SetRcpFloat：設定參數至配方位址 (Float)

指令名稱	recipe.SetRcpFloat
指令運算式	ret = **recipe.SetRcpFloat**(index, floatValue)
參數定義	Index：整數；配方索引 floatValue：浮點數；配方 32 位元浮點數
範例	ret = recipe.SetRcpFloat(1, 9.9)
範例說明	設定 9.9 至配方位址 RCP1。
回傳值	ret：成功回傳 1；失敗回傳 0

■ recipe.SetCurEnRcpNoName：設定加強型配方組別名稱

指令名稱	recipe.SetCurEnRcpNoName
指令運算式	ret = **recipe.SetCurEnRcpNoName**(newName)
參數定義	newName：字串；欲設定的加強型配方組別名稱
範例	ret = recipe.SetCurEnRcpNoName("POSHENG")
範例說明	設定 POSHENG 為當前加強型配方指定組別的名稱。
回傳值	ret：成功回傳 1；失敗回傳 0

■ recipe.SetCurEnRcpGName：設定加強型配方群組名稱

指令名稱	recipe.SetCurEnRcpGName
指令運算式	ret = **recipe.SetCurEnRcpGName**(newName)
參數定義	newName：字串；欲設定的加強型配方群組名稱
範例	ret = recipe.SetCurEnRcpGName("POSHENG")
範例說明	設定 POSHENG 為當前加強型配方指定群組的名稱。
回傳值	ret：成功回傳 1；失敗回傳 0

■ recipe.SetEnRcpWord：設定參數至加強型配方位址 (Word)

指令名稱	recipe.SetEnRcpWord
指令運算式	ret = **recipe.SetEnRcpWord**(index, word)
參數定義	Index：整數；加強型配方索引 Word：整數；加強型配方 WORD
範例	ret = recipe.SetEnRcpWord(1, 88)
範例說明	設定 88 於當前加強型配方 ENRCP1。
回傳值	ret：成功回傳 1；失敗回傳 0

■ recipe.SetEnRcpDWord：設定參數至加強型配方位址 (Double Word)

指令名稱	recipe.SetEnRcpDWord
指令運算式	ret = **recipe.SetEnRcpDWord**(index, dword)
參數定義	Index：整數；加強型配方索引 dword：整數；加強型配方 DWORD
範例	ret = recipe.SetEnRcpDWord(2, 65536)
範例說明	設定 65536 於當前加強型配方 ENRCP2。
回傳值	ret：成功回傳 1；失敗回傳 0

■ recipe.SetEnRcpFloat：設定參數至加強型配方位址 (Float)

指令名稱	recipe.SetEnRcpFloat
指令運算式	ret = **recipe.SetEnRcpFloat**(index, floatValue)
參數定義	Index：整數；加強型配方索引 floatValue：浮點數；加強型配方 32 位元浮點數
範例	ret = recipe.SetEnRcpFloat(3, 99.9)
範例說明	設定 99.9 至加強型配方位址 ENRCP3。
回傳值	ret：成功回傳 1；失敗回傳 0

■ recipe.SetEnRcpAscii：設定字串至加強型配方位址

指令名稱	recipe.SetEnRcpAscii
指令運算式	ret = **recipe.SetEnRcpAscii**(index, str, len)
參數定義	Index：整數；加強型配方索引 str：字串；加強型配方字串 len：整數；字串長度
範例	ret = recipe.SetEnRcpAscii(4, "POSHENG", 6)
範例說明	以 6 個 bytes 設定 POSHENG 至加強型配方位址 ENRCP4。
回傳值	ret：成功回傳 1；失敗回傳 0

■ recipe.ChangeRcpNoIndex：更改配方組別索引

指令名稱	recipe.ChangeRcpNoIndex
指令運算式	ret = **recipe.ChangeRcpNoIndex**(noIdx)
參數定義	noIdx：整數；配方組別索引
範例	ret = recipe.ChangeRcpNoIndex(1)
範例說明	切換配方組別索引為 1。
回傳值	ret：成功回傳 1；失敗回傳 0

■ recipe.ChangeRcpGIndex：更改配方群組索引

指令名稱	recipe.ChangeRcpGIndex
指令運算式	ret = **recipe.ChangeRcpGIndex**(gIdx)
參數定義	gIdx：整數；配方群組索引
範例	ret = recipe.ChangeRcpGIndex(1)
範例說明	切換配方群組索引為 1。
回傳值	ret：成功回傳 1；失敗回傳 0

■ recipe.ChangeEnRcpNoIndex：更改加強型配方組別索引

指令名稱	recipe.ChangeEnRcpNoIndex
指令運算式	ret = **recipe.ChangeEnRcpNoIndex**(noIdx)
參數定義	noIdx：整數，加強型配方組別索引
範例	ret = recipe.ChangeEnRcpNoIndex(3)
範例說明	切換加強型配方組別索引為 3。
回傳值	ret：成功回傳 1；失敗回傳 0

■ recipe.ChangeEnRcpGIndex：更改加強型配方群組索引

指令名稱	recipe.ChangeEnRcpGIndex
指令運算式	ret = **recipe.ChangeEnRcpGIndex**(gIdx)
參數定義	gIdx：整數；加強型配方群組索引
範例	ret = recipe.ChangeEnRcpGIndex(2)
範例說明	切換加強型配方群組索引為 2。
回傳值	ret：成功回傳 1；失敗回傳 0

■ recipe.SetEnRcpDouble：設定參數至加強型配方位址(雙精度浮點數)

指令名稱	recipe.SetEnRcpDouble
指令運算式	ret = **recipe.SetEnRcpDouble**(index, doubleValue)
參數定義	Index：整數；加強型配方索引 doubleValue：浮點數；加強型配方 64 位元浮點數
範例	ret = recipe.SetEnRcpDouble(0, 22.22)
範例說明	設定 22.22 至加強型配方位址 ENRCP0。
回傳值	ret：成功回傳 1；失敗回傳 0

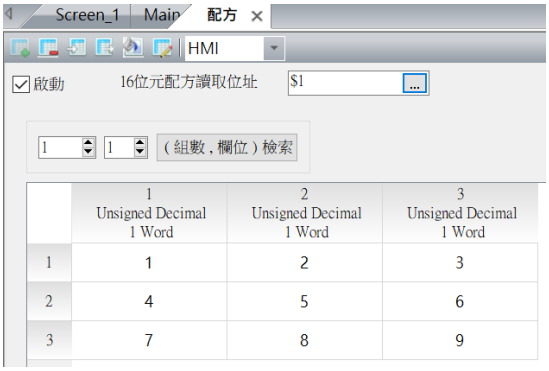
■ recipe.GetEnRcpDouble：取得指定加強型配方位址的數值 (雙精度浮點數)

指令名稱	recipe.GetEnRcpDouble
指令運算式	ret, value = **recipe.GetEnRcpDouble**(index)
參數定義	Index：整數；加強型配方索引
範例	ret, value = recipe.GetEnRcpDouble(0)
範例說明	取得指定加強型配方位址(ENRCP0)的數值，單位為 64 位元浮點數。
回傳值	ret：成功回傳 1；失敗回傳 0

範例 (recipe.GetCurRcpNoIndex、recipe.GetCurRcpGIndex、recipe.GetRcpWord、recipe.GetRcpDWord、recipe.GetRcpFloat)

建立 16 位元、32 位元配方

■ 於 DOPSoft 中，[選項] → [配方] → [16 位元配方] → 新增配方 。



	1 Unsigned Decimal 1 Word	2 Unsigned Decimal 1 Word	3 Unsigned Decimal 1 Word
1	1	2	3
2	4	5	6
3	7	8	9

■ 於 DOPSoft 中，[選項] → [配方] → [32 位元配方] → 新增兩組配方 。

群組 1：

	1 Unsigned Decimal 2 Word	2 Unsigned Decimal 2 Word	3 Unsigned Decimal 2 Word
1	10000	20000	30000
2	40000	50000	60000
3	70000	80000	90000

群組 2：

	1 Floating 2 Word	2 Floating 2 Word	3 Floating 2 Word
1	1.10	2.20	3.30
2	4.40	5.50	6.60
3	7.70	8.80	9.90

範例 (recipe.GetCurRcpNoIndex、recipe.GetCurRcpGIndex、recipe.GetRcpWord、recipe.GetRcpDWord、recipe.GetRcpFloat)	

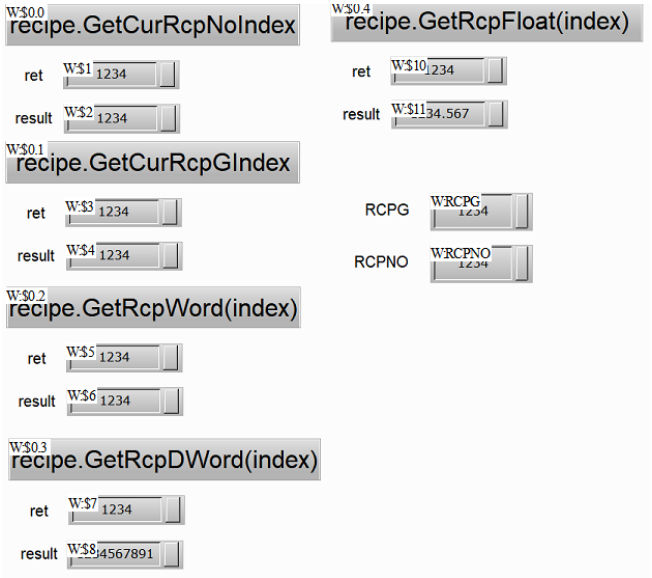
建立 Lua 程序	■ 輸入 Lua 指令如下。
	index=1
	while true do
	if mem.inter.ReadBit(0,0)==1 then
	ret, noIdx = recipe.GetCurRcpNoIndex()
	mem.inter.Write(1, ret)
	mem.inter.Write(2,noIdx)
	end
	
	if mem.inter.ReadBit(0,1)==1 then
	ret, gIdx = recipe.GetCurRcpGIndex()
	mem.inter.Write(3, ret)
	mem.inter.Write(4,gIdx)
	end
	
	if mem.inter.ReadBit(0,2)==1 then
	ret, value = recipe.GetRcpWord(index)
	mem.inter.Write(5, ret)
	mem.inter.Write(6,value)
	end
	
	if mem.inter.ReadBit(0,3)==1 then
	ret, value = recipe.GetRcpDWord(index)
	mem.inter.Write(7, ret)
	mem.inter.WriteDW(8,value)
	end
	
	if mem.inter.ReadBit(0,4)==1 then
	ret, value = recipe.GetRcpFloat(index)
	mem.inter.Write(10, ret)
	mem.inter.WriteFloat(11,value)
	end
	end

建立元件	■ 建立 5 個交替型按鈕，寫入記憶體位址分別為$0.0、$0.1、$0.2、$0.3、$0.4。 ■ 建立 8 個數值輸入元件，數值單位為 Word，數值格式為 Unsigned Decimal，寫入記憶體位址分別設定為$1、$2、$3、$4、$5、$6、$7、$10。 ■ 建立 1 個數值輸入元件，數值單位為 Double Word，數值格式 Floating，寫入記憶體位址為$11。 ■ 建立 1 個數值輸入元件，數值單位為 Double Word，數值格式 Unsigned Decimal，寫入記憶體位址為$8。

範例 (recipe.GetCurRcpNoIndex、recipe.GetCurRcpGIndex、recipe.GetRcpWord、recipe.GetRcpDWord、recipe.GetRcpFloat)

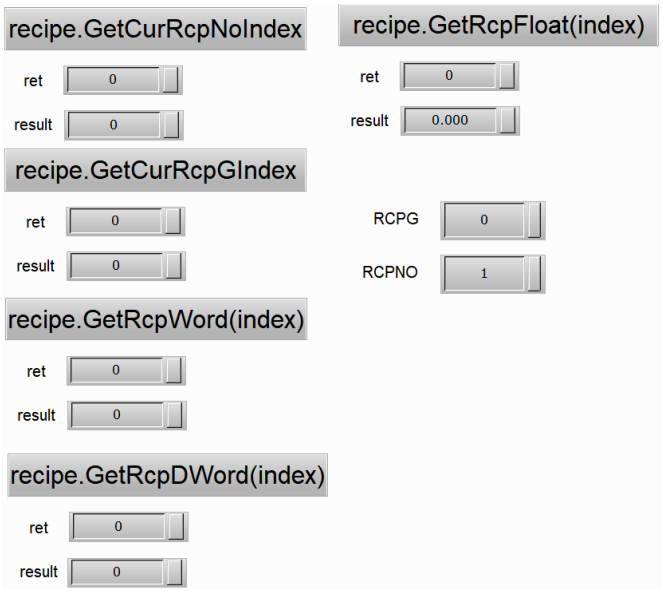
建立元件

■ 建立 2 個數值輸入元件，寫入記憶體位址分別為 RCPG 及 RCPNO。



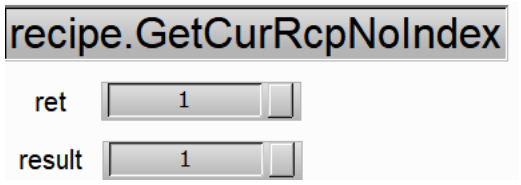
執行結果

■ 完成元件建立後載入至人機。



■ 按下$0.0，將回傳值寫入$1，並將當前組別索引 RCPNO 寫入$2。

```
if mem.inter.ReadBit(0,0)==1 then
    ret, noIdx = recipe.GetCurRcpNoIndex()
    mem.inter.Write(1, ret)
    mem.inter.Write(2,noIdx)
end.
```

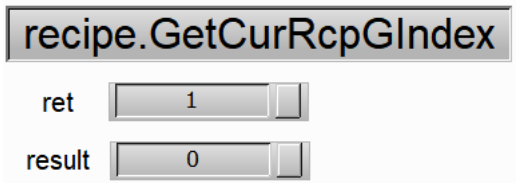


範例 (recipe.GetCurRcpNoIndex、recipe.GetCurRcpGIndex、recipe.GetRcpWord、recipe.GetRcpDWord、recipe.GetRcpFloat)

執行結果

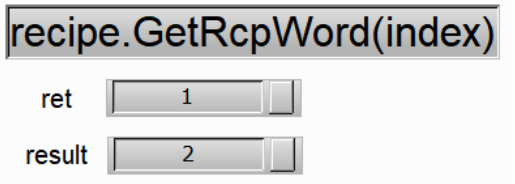
■ 按下$0.1，將回傳值寫入$3，並將當前群組索引 RCPG 寫入$4。

```
if mem.inter.ReadBit(0,1)==1 then
    ret, gIdx = recipe.GetCurRcpGIndex()
    mem.inter.Write(3, ret)
    mem.inter.Write(4,gIdx)
end
```



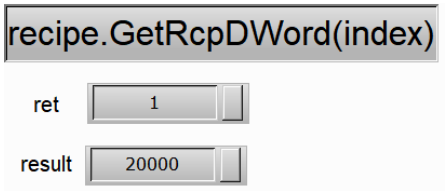
■ 按下$0.2，因 index 為 1，所以透過 recipe.GetRcpWord 取得 RCP1 的數值，最後把結果寫入$5、$6。

```
if mem.inter.ReadBit(0,2)==1 then
    ret, value = recipe.GetRcpWord(index)
    mem.inter.Write(5, ret)
    mem.inter.Write(6,value)
end
```



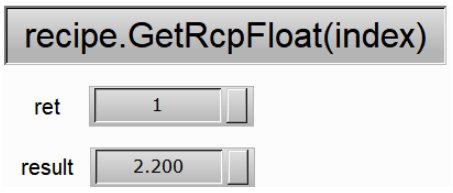
■ 切換配方群組 RCPG 為 1，按下$0.3，將 RCP 以 Double Word 格式寫入至$8。

```
if mem.inter.ReadBit(0,3)==1 then
    ret, value = recipe.GetRcpDWord(index)
    mem.inter.Write(7, ret)
    mem.inter.WriteDW(8,value)
end
```



■ 切換配方群組 RCPG 為 2，按下$0.4，將 RCP1 寫入至$11。

```
if mem.inter.ReadBit(0,4)==1 then
    ret, value = recipe.GetRcpFloat(index)
    mem.inter.Write(10, ret)
    mem.inter.WriteFloat(11,value)
end
```



範例 (recipe.GetCurEnRcpNoName、recipe.GetCurEnRcpGName、recipe.GetCurEnRcpNoIndex、recipe.GetCurEnRcpGIndex、recipe.GetEnRcpWord、recipe.GetEnRcpDWord、recipe.GetEnRcpFloat、recipe.GetEnRcpAscii)

建立加強型配方

■ 於 DOPSoft 中，[選項] → [配方] → [加強型配方] → 新增加強型配方  。

1:recipe1 (4X4)

	RCPNO名稱	1 Unsigned Decimal 1 Word	2 Floating 2 Word	3 Char 3 Word	4 Unsigned Decimal 2 Word
標題名稱					
1	1	1	1.1	A	11
2	2	2	2.2	B	22
3	3	3	3.3	C	33
4	4	4	4.4	D	44

建立 Lua 程序

■ 輸入 Lua 指令如下。

```
while true do
        if mem.inter.ReadBit(0,6)==1 then
            ret, noName = recipe.GetCurEnRcpNoName()
            mem.inter.Write(15,ret)
            mem.inter.WriteAscii(16,noName,string.len(noName))
        end

        if mem.inter.ReadBit(0,7)==1 then
            ret, gName = recipe.GetCurEnRcpGName()
            mem.inter.Write(20,ret)
            mem.inter.WriteAscii(21,gName,string.len(gName))
        end

        if mem.inter.ReadBit(0,8)==1 then
            ret, noIdx = recipe.GetCurEnRcpNoIndex()
            mem.inter.Write(25,ret)
            mem.inter.Write(26,noIdx)
        end

        if mem.inter.ReadBit(0,9)==1 then
            ret, gIdx = recipe.GetCurEnRcpGIndex()
            mem.inter.Write(27,ret)
            mem.inter.Write(28,gIdx)
        end

        if mem.inter.ReadBit(0,10)==1 then
            ret, value = recipe.GetEnRcpWord(0)
            mem.inter.Write(29,ret)
            mem.inter.Write(30,value)
        end

        if mem.inter.ReadBit(0,11)==1 then
            ret, value = recipe.GetEnRcpDWord(3)
            mem.inter.Write(31,ret)
            mem.inter.WriteFloat(32,value)
        end

        if mem.inter.ReadBit(0,12)==1 then
            ret, value = recipe.GetEnRcpFloat(1)
            mem.inter.Write(34,ret)
            mem.inter.WriteFloat(35,value)
        end
```

範例 (recipe.GetCurEnRcpNoName、recipe.GetCurEnRcpGName、recipe.GetCurEnRcpNoIndex、recipe.GetCurEnRcpGIndex、recipe.GetEnRcpWord、recipe.GetEnRcpDWord、recipe.GetEnRcpFloat、recipe.GetEnRcpAscii)

```
        if mem.inter.ReadBit(0,13)==1 then
            ret, str = recipe.GetEnRcpAscii(2)
            mem.inter.Write(37,ret)
            mem.inter.WriteAscii(38,str,string.len(str))
        end
end
```

建立元件

- 建立 8 個交替型按鈕，寫入記憶體位址分別為 $0.6、$0.7、$0.8、$0.9、$0.10、$0.11、$0.12、$0.13。
- 建立 11 個數值輸入元件，數值單位為 Word，數值格式為 Unsigned Decimal，寫入記憶體位址分別設定為$15、$20、$25、$26、$27、$28、$29、$30、$31、$34、$37。
- 建立 3 個文數字輸入元件，寫入記憶體位址分別為$16、$21、$38，字串長度分別為 4、10、10。
- 建立 2 個數值輸入元件，數值單位為 Double Word，數值格式為 Float，寫入記憶體位址分別為$32、$35。
- 建立 2 個數值輸入元件，寫入記憶體位址分別為 ENRCPG 及 ENRCPNO。

執行結果

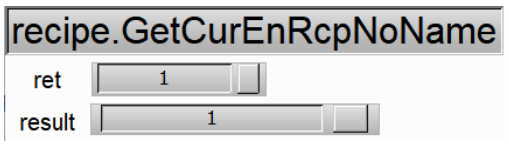
- 完成元件建立後載入至人機。

範例 (recipe.GetCurEnRcpNoName、recipe.GetCurEnRcpGName、recipe.GetCurEnRcpNoIndex、recipe.GetCurEnRcpGIndex、recipe.GetEnRcpWord、recipe.GetEnRcpDWord、recipe.GetEnRcpFloat、recipe.GetEnRcpAscii)

執行結果

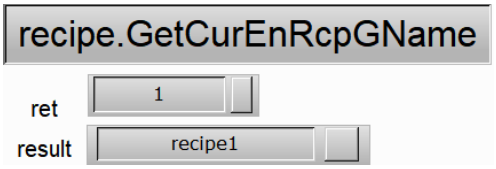
■ 按下$0.6，將回傳值寫入$15，並將當前加強型配方組別名稱寫入$16。

```
if mem.inter.ReadBit(0,6)==1 then
    ret, noName = recipe.GetCurEnRcpNoName()
    mem.inter.Write(15,ret)
    mem.inter.WriteAscii(16,noName,string.len(noName))
end
```



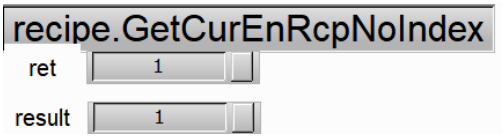
■ 按下$0.7，將回傳值寫入$20，並將當前加強型配方群組名稱寫入$21。

```
if mem.inter.ReadBit(0,7)==1 then
    ret, gName = recipe.GetCurEnRcpGName()
    mem.inter.Write(20,ret)
    mem.inter.WriteAscii(21,gName,string.len(gName))
end
```



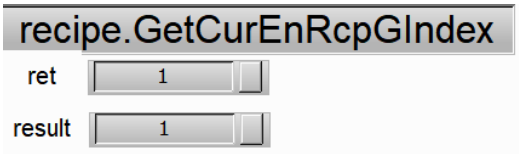
■ 按下$0.8，將回傳值寫入$25，並將當前加強型配方組別索引寫入$26。

```
if mem.inter.ReadBit(0,8)==1 then
    ret, noIdx = recipe.GetCurEnRcpNoIndex()
    mem.inter.Write(25,ret)
    mem.inter.Write(26,noIdx)
end
```

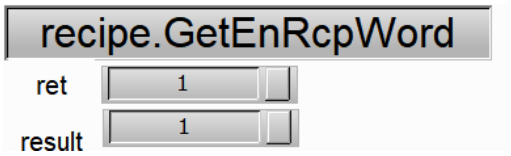
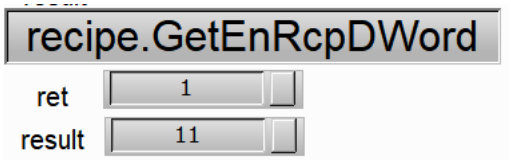
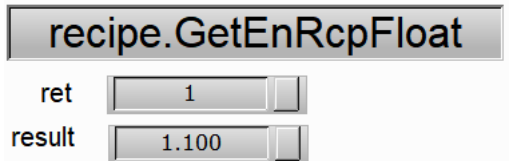
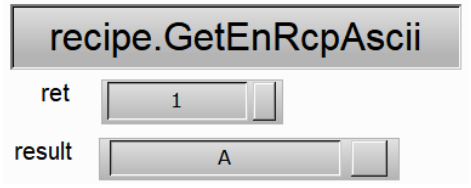


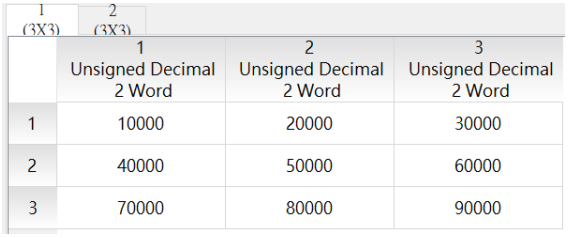
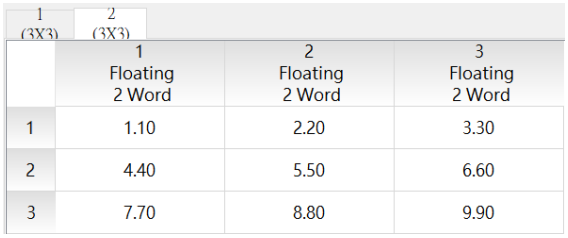
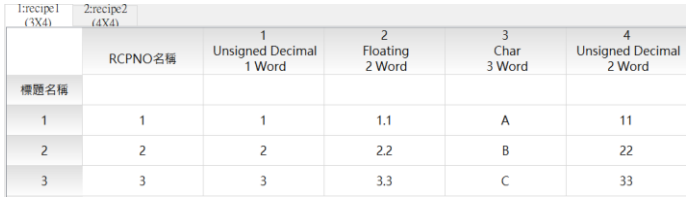
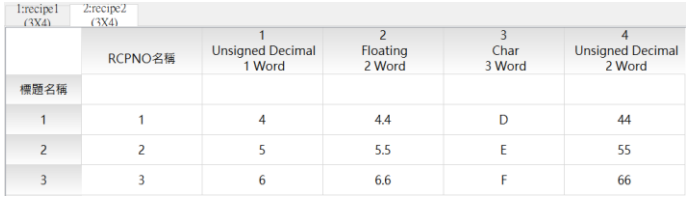
■ 按下$0.9，將回傳值寫入$27，並將當前加強型配方群組索引寫入$28。

```
if mem.inter.ReadBit(0,9)==1 then
    ret, gIdx = recipe.GetCurEnRcpGIndex()
    mem.inter.Write(27,ret)
    mem.inter.Write(28,gIdx)
end
```



範例 (recipe.GetCurEnRcpNoName、recipe.GetCurEnRcpGName、recipe.GetCurEnRcpNoIndex、recipe.GetCurEnRcpGIndex、recipe.GetEnRcpWord、recipe.GetEnRcpDWord、recipe.GetEnRcpFloat、recipe.GetEnRcpAscii)	

執行結果	■ 按下$0.10，將回傳值寫入$29，並將 ENRCP0 的數值以 word 格式寫入$30。
	`if mem.inter.ReadBit(0,10)==1 then`
	`ret, value = recipe.GetEnRcpWord(0)`
	`mem.inter.Write(29,ret)`
	`mem.inter.Write(30,value)`
	`end`
	 
	■ 按下$0.11，將回傳值寫入$31，並將 ENRCP3 的數值以 Double Word 格式寫入$32。
	`if mem.inter.ReadBit(0,11)==1 then`
	`ret, value = recipe.GetEnRcpDWord(3)`
	`mem.inter.Write(31,ret)`
	`mem.inter.WriteFloat(32,value)`
	`end`
	 
	■ 按下$0.12，將回傳值寫入$34，並將 ENRCP1 以浮點數格式寫入$35。
	`if mem.inter.ReadBit(0,12)==1 then`
	`ret, value = recipe.GetEnRcpFloat(1)`
	`mem.inter.Write(34,ret)`
	`mem.inter.WriteFloat(35,value)`
	`end`
	 
	■ 按下$0.13，將回傳值寫入$37，並 ENRCP2 以字串格式寫入$38。
	`if mem.inter.ReadBit(0,13)==1 then`
	`ret, str = recipe.GetEnRcpAscii(2)`
	`mem.inter.Write(37,ret)`
	`mem.inter.WriteAscii(38,str,string.len(str))`
	`end`
	 

範例 (recipe.SetRcpWord、recipe.SetRcpDWord、recipe.SetRcpFloat、recipe.SetCurEnRcpNoName、recipe.SetCurEnRcpGName、recipe.SetEnRcpWord、recipe.SetEnRcpDWord、recipe.SetEnRcpFloat、recipe.SetEnRcpAscii、recipe.ChangeRcpNoIndex、recipe.ChangeRcpGIndex、recipe.ChangeEnRcpNoIndex、recipe.ChangeEnRcpGIndex)	
建立配方	■ 於 DOPSoft 中，[選項] → [配方] → [16 位元配方] → 新增配方 。  ■ 於 DOPSoft 中，[選項] → [配方] → [32 位元配方] → 新增兩組配方 。 群組 1：  群組 2：  ■ 於 DOPSoft 中，[選項] → [配方] → [加強型配方] → 新增兩組加強型配方 。 群組 1：  群組 2： 

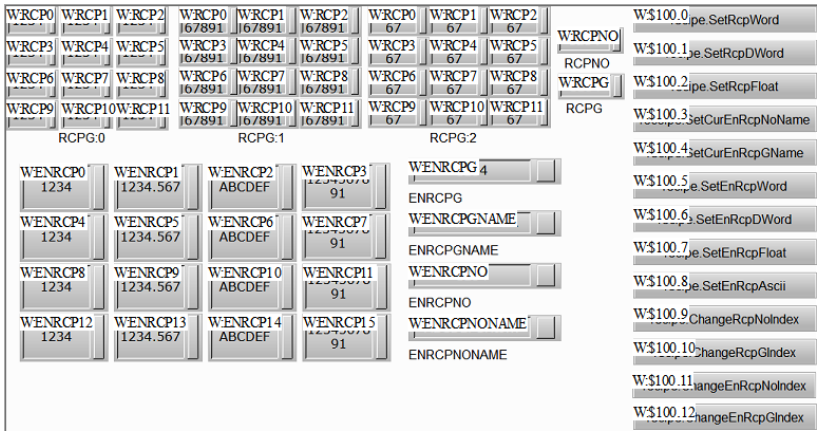
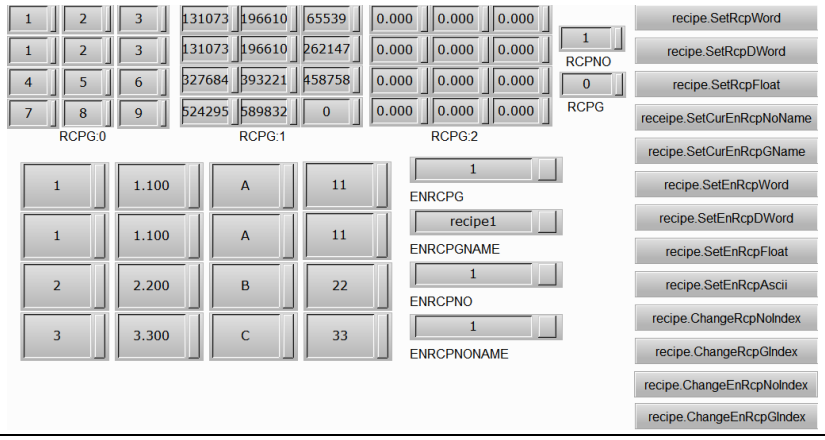
範例（recipe.SetRcpWord、recipe.SetRcpDWord、recipe.SetRcpFloat、recipe.SetCurEnRcpNoName、recipe.SetCurEnRcpGName、recipe.SetEnRcpWord、recipe.SetEnRcpDWord、recipe.SetEnRcpFloat、recipe.SetEnRcpAscii、recipe.ChangeRcpNoIndex、recipe.ChangeRcpGIndex、recipe.ChangeEnRcpNoIndex、recipe.ChangeEnRcpGIndex）	

建立 Lua 程序	■ 輸入 Lua 指令如下。
	while true do
	if mem.inter.ReadBit(100,0)==1 then
	index=3
	word=100
	ret = recipe.SetRcpWord(index, word)
	end
	
	if mem.inter.ReadBit(100,1)==1 then
	index=3
	dword=70000
	ret = recipe.SetRcpDWord(index, dword)
	end
	
	if mem.inter.ReadBit(100,2)==1 then
	index=3
	floatValue=10.10
	ret = recipe.SetRcpFloat(index, floatValue)
	end
	
	if mem.inter.ReadBit(100,3)==1 then
	newName="recipe_NoName"
	ret = recipe.SetCurEnRcpNoName(newName)
	end
	
	if mem.inter.ReadBit(100,4)==1 then
	newName="recipe_GName"
	ret = recipe.SetCurEnRcpGName(newName)
	end
	
	if mem.inter.ReadBit(100,5)==1 then
	index=4
	word=88
	ret = recipe.SetEnRcpWord(index, word)
	end
	
	if mem.inter.ReadBit(100,6)==1 then
	index=7
	dword=70000
	ret = recipe.SetEnRcpDWord(index, dword)
	end
	
	if mem.inter.ReadBit(100,7)==1 then
	index=5
	dword=99.99
	ret = recipe.SetEnRcpFloat(index, floatValue)
	end
	
	if mem.inter.ReadBit(100,8)==1 then
	index=6
	str="POSHEN"

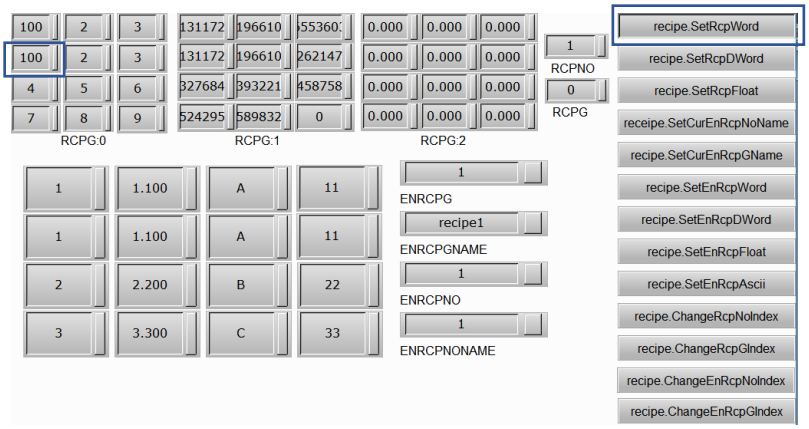
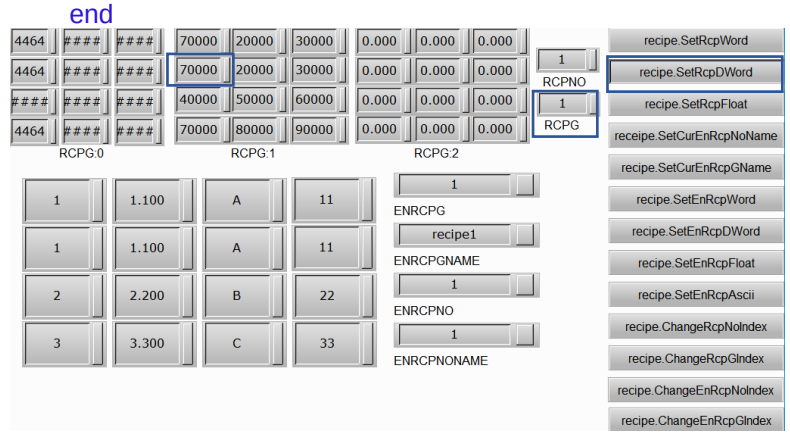
範例（recipe.SetRcpWord、recipe.SetRcpDWord、recipe.SetRcpFloat、recipe.SetCurEnRcpNoName、recipe.SetCurEnRcpGName、recipe.SetEnRcpWord、recipe.SetEnRcpDWord、recipe.SetEnRcpFloat、recipe.SetEnRcpAscii、recipe.ChangeRcpNoIndex、recipe.ChangeRcpGIndex、recipe.ChangeEnRcpNoIndex、recipe.ChangeEnRcpGIndex）	

	len=6
	ret = recipe.SetEnRcpAscii(index, str, len)
	end
	
	if mem.inter.ReadBit(100,9)==1 then
	noIdx=2
	ret = recipe.ChangeRcpNoIndex(noIdx)
	end
	
	if mem.inter.ReadBit(100,10)==1 then
	gIdx=2
	ret = recipe.ChangeRcpGIndex(gIdx)
	end
	
	if mem.inter.ReadBit(100,11)==1 then
	noIdx=3
	ret = recipe.ChangeEnRcpNoIndex(noIdx)
	end
	
	if mem.inter.ReadBit(100,12)==1 then
	gIdx=2
	ret = recipe.ChangeEnRcpGIndex(gIdx)
	end
	end

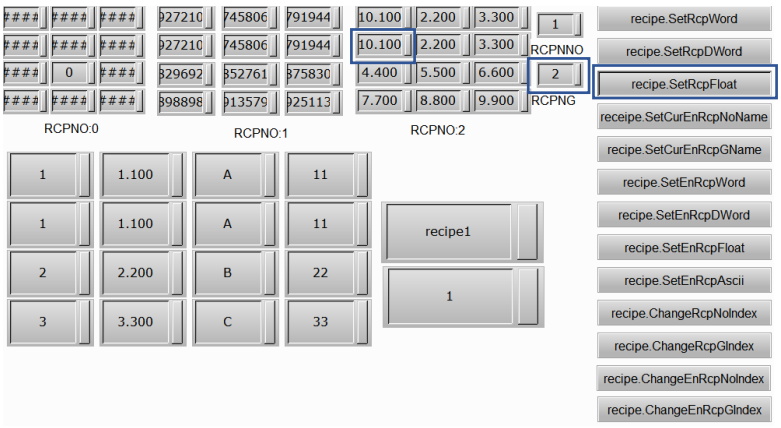
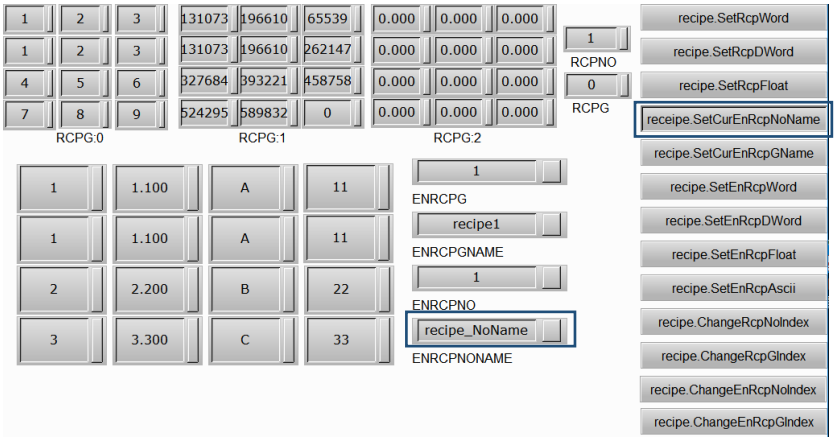
建立元件	■ 建立 12 個數值輸入元件，寫入記憶體位址分別為 RCP0、RCP1、RCP2…RCP11，數值單位為 Word，數值格式為 Unsigned Decimal。 ■ 建立 12 個數值輸入元件，寫入記憶體位址分別為 RCP0、RCP1、RCP2…RCP11，數值單位為 Double Word，數值格式為 Unsigned Decimal。 ■ 建立 12 個數值輸入元件，寫入記憶體位址分別為 RCP0、RCP1、RCP2…RCP11，數值單位為 Double Word，數值格式為 Floating。 ■ 建立四個數值輸入元件，寫入記憶體位址分別設為 ENRCP0、ENRCP4、ENRCP8、ENRCP12，數值單位為 Word，數值格式為 Unsigned Decimal。 ■ 建立四個數值輸入元件，寫入記憶體位址分別設為 ENRCP1、ENRCP5、ENRCP9、ENRCP13，數值單位為 Double Word，數值格式為 Floating。 ■ 建立四個文數字輸入元件，寫入記憶體位址分別設為 ENRCP2、ENRCP6、ENRCP10、ENRCP14，字串長度設定為 6。 ■ 建立四個數值輸入元件，寫入記憶體位址分別設為 ENRCP3、ENRCP7、ENRCP11、ENRCP15，數值單位為 Double Word，數值格式為 Unsigned Decimal。

範例 (recipe.SetRcpWord、recipe.SetRcpDWord、recipe.SetRcpFloat、recipe.SetCurEnRcpNoName、recipe.SetCurEnRcpGName、recipe.SetEnRcpWord、recipe.SetEnRcpDWord、recipe.SetEnRcpFloat、recipe.SetEnRcpAscii、recipe.ChangeRcpNoIndex、recipe.ChangeRcpGIndex、recipe.ChangeEnRcpNoIndex、recipe.ChangeEnRcpGIndex)	
建立元件	■ 建立 2 個數值輸入元件，寫入記憶體位址分別為 RCPNO 及 RCPG。 ■ 建立 2 個文數字輸入元件，寫入記憶體位址分別為 ENRCPGNAME 及 ENRCPNONAME。 ■ 建立 2 個數值輸入元件，寫入記憶體位址分別為 ENRCPNO 及 ENRCPG。 ■ 建立 13 個交替型按鈕，寫入記憶體位址分別為$100.0、$100.1、$100.2…$100.12。 ■ 建立完後如下圖。 
執行結果	■ 完成元件建立後載入至人機。 

範例 (recipe.SetRcpWord、recipe.SetRcpDWord、recipe.SetRcpFloat、recipe.SetCurEnRcpNoName、recipe.SetCurEnRcpGName、recipe.SetEnRcpWord、recipe.SetEnRcpDWord、recipe.SetEnRcpFloat、recipe.SetEnRcpAscii、recipe.ChangeRcpNoIndex、recipe.ChangeRcpGIndex、recipe.ChangeEnRcpNoIndex、recipe.ChangeEnRcpGIndex)	

執行結果	■ 按下$100.0，以 100 數值寫入配方第 RCP3 位址，單位為 Word。
	`if mem.inter.ReadBit(100,0)==1 then`
	`index=3`
	`word=100`
	`ret = recipe.SetRcpWord(index, word)`
	`end`
	 
	
	■ 切換 RCPG 為 1，按下$100.1，以 70000 數值寫入配方第 RCP3 位址，單位為 DWord。
	`if mem.inter.ReadBit(100,1)==1 then`
	`index=3`
	`dword=70000`
	`ret = recipe.SetRcpDWord(index, dword)`
	`end`
	 

範例 (recipe.SetRcpWord、recipe.SetRcpDWord、recipe.SetRcpFloat、recipe.SetCurEnRcpNoName、recipe.SetCurEnRcpGName、recipe.SetEnRcpWord、recipe.SetEnRcpDWord、recipe.SetEnRcpFloat、recipe.SetEnRcpAscii、recipe.ChangeRcpNoIndex、recipe.ChangeRcpGIndex、recipe.ChangeEnRcpNoIndex、recipe.ChangeEnRcpGIndex)	

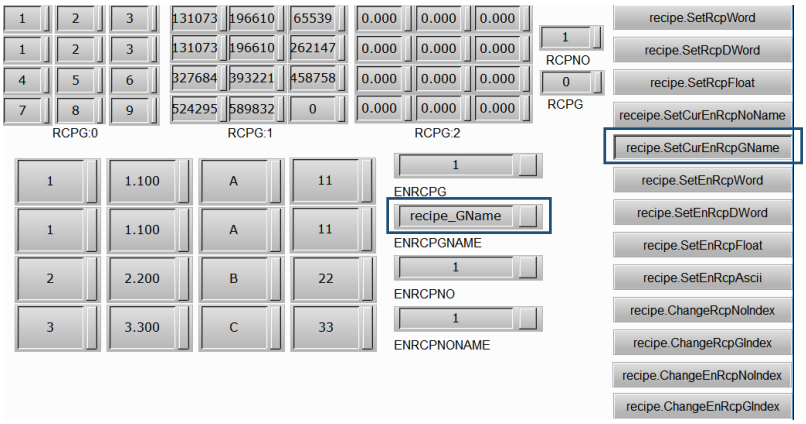
執行結果	■ 切換 RCPG 為 2，按下$100.2，以 10.10 數值寫入配方第 RCP3 位址，單位為 Float。
	`if mem.inter.ReadBit(100,2)==1 then`
	`  index=3`
	`  floatValue=10.10`
	`  ret = recipe.SetRcpFloat(index, floatValue)`
	`end`
	 
	
	■ 按下$100.3，以 recipe_NoName 字串設定至配方組別索引名稱。
	`if mem.inter.ReadBit(100,3)==1 then`
	`  newName="recipe_NoName"`
	`  ret = recipe.SetCurEnRcpNoName(newName)`
	`end`
	 

範例 (recipe.SetRcpWord、recipe.SetRcpDWord、recipe.SetRcpFloat、recipe.SetCurEnRcpNoName、recipe.SetCurEnRcpGName、recipe.SetEnRcpWord、recipe.SetEnRcpDWord、recipe.SetEnRcpFloat、recipe.SetEnRcpAscii、recipe.ChangeRcpNoIndex、recipe.ChangeRcpGIndex、recipe.ChangeEnRcpNoIndex、recipe.ChangeEnRcpGIndex)

執行結果

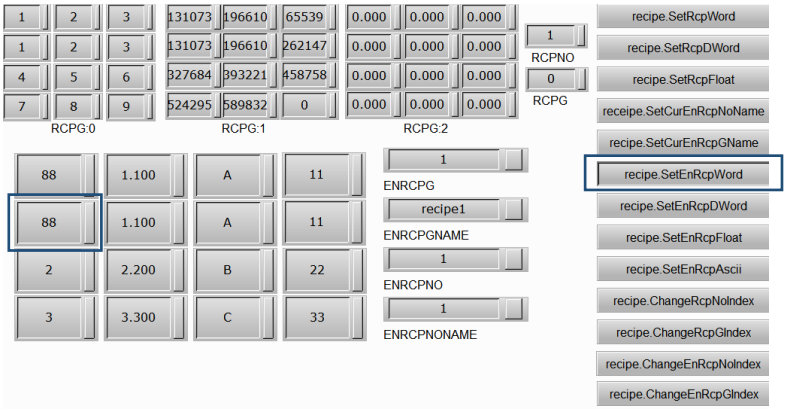
■ 按下$100.4，以 recipe_GName 字串設定至配方群組索引名稱。

```
if mem.inter.ReadBit(100,4)==1 then
    newName="recipe_GName"
    ret = recipe.SetCurEnRcpGName(newName)
end
```



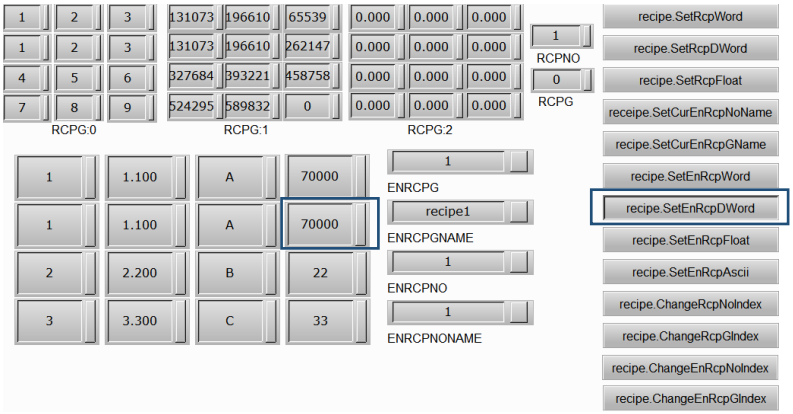
■ 按下$100.5，以 88 數值寫入至 ENRCP4 位址，數值單位為 Word。

```
if mem.inter.ReadBit(100,5)==1 then
    index=4
    word=88
    ret = recipe.SetEnRcpWord(index, word)
end
```

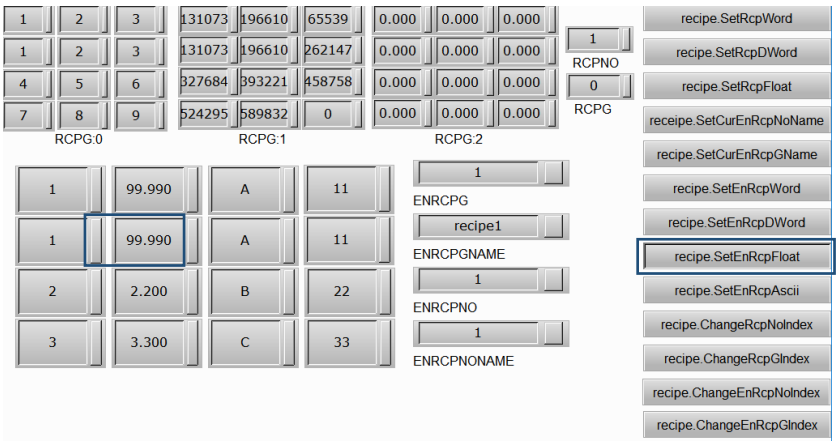


範例 (recipe.SetRcpWord、recipe.SetRcpDWord、recipe.SetRcpFloat、recipe.SetCurEnRcpNoName、recipe.SetCurEnRcpGName、recipe.SetEnRcpWord、recipe.SetEnRcpDWord、recipe.SetEnRcpFloat、recipe.SetEnRcpAscii、recipe.ChangeRcpNoIndex、recipe.ChangeRcpGIndex、recipe.ChangeEnRcpNoIndex、recipe.ChangeEnRcpGIndex)	

	■ 按下$100.6，以 70000 數值寫入至 ENRCP7 位址，數值單位為 Double Word。
	`if mem.inter.ReadBit(100,6)==1 then`
	`index=7`
	`dword=70000`
	`ret = recipe.SetEnRcpDWord(index, dword)`
	`end`

執行結果	

	■ 按下$100.7，以 99.99 數值寫入至 ENRCP5 位址，數值單位為 Float。
	`if mem.inter.ReadBit(100,7)==1 then`
	`index=5`
	`dword=99.99`
	`ret = recipe.SetEnRcpFloat(index, floatValue)`
	`end`

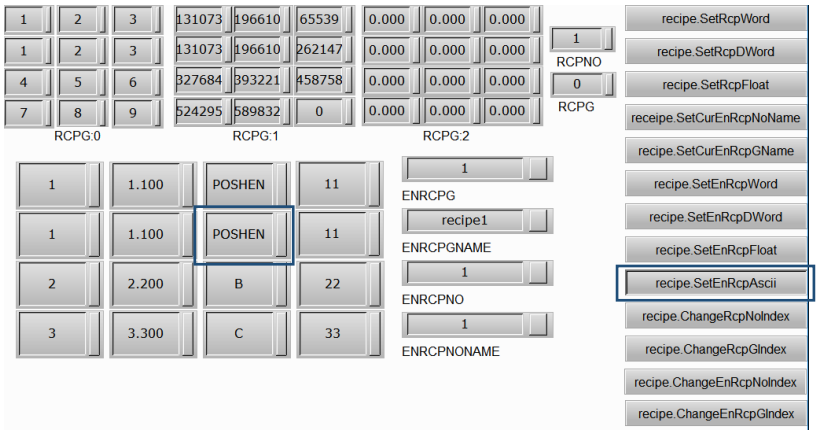
	

範例 (recipe.SetRcpWord、recipe.SetRcpDWord、recipe.SetRcpFloat、recipe.SetCurEnRcpNoName、recipe.SetCurEnRcpGName、recipe.SetEnRcpWord、recipe.SetEnRcpDWord、recipe.SetEnRcpFloat、recipe.SetEnRcpAscii、recipe.ChangeRcpNoIndex、recipe.ChangeRcpGIndex、recipe.ChangeEnRcpNoIndex、recipe.ChangeEnRcpGIndex)

執行結果

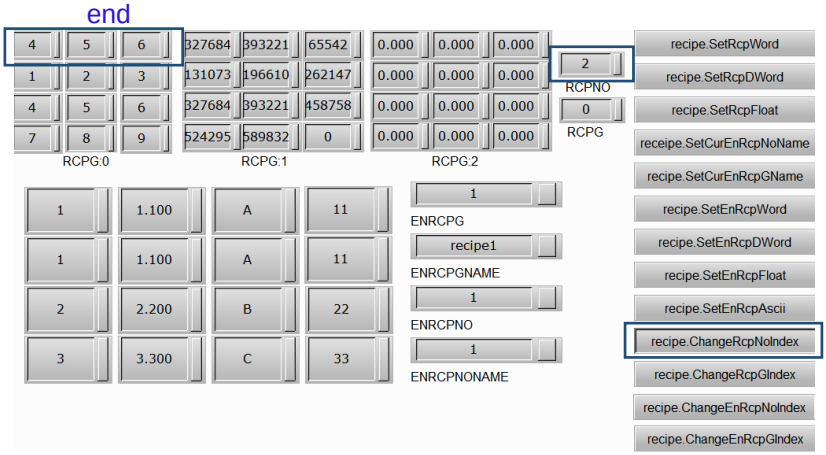
■ 按下$100.8，"POSHEN"字串以長度 6 個 bytes 寫入至 ENRCP6 位址。

```
if mem.inter.ReadBit(100,8)==1 then
    index=6
    str="POSHEN"
    len=6
    ret = recipe.SetEnRcpAscii(index, str, len)
end
```

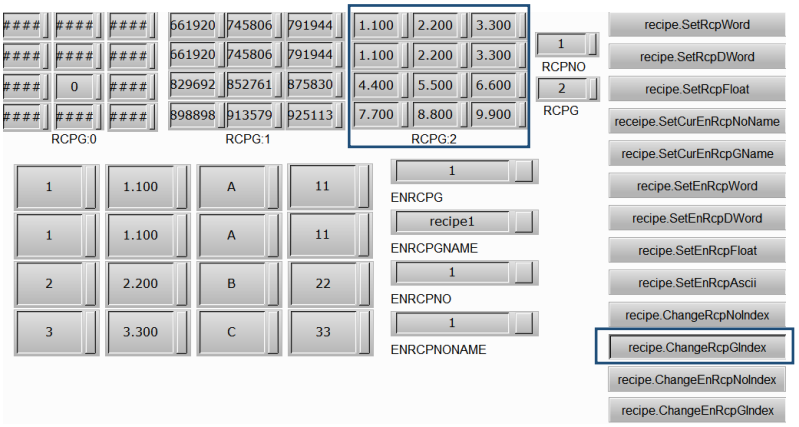
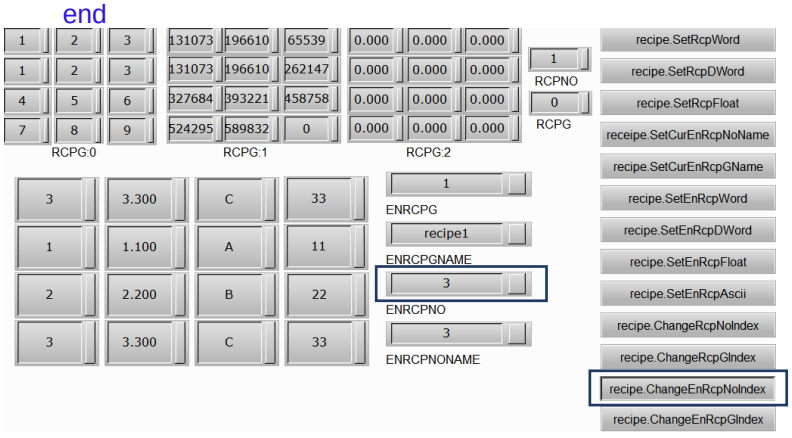


■ 按下$100.9，切換 RCPNO 為 2。

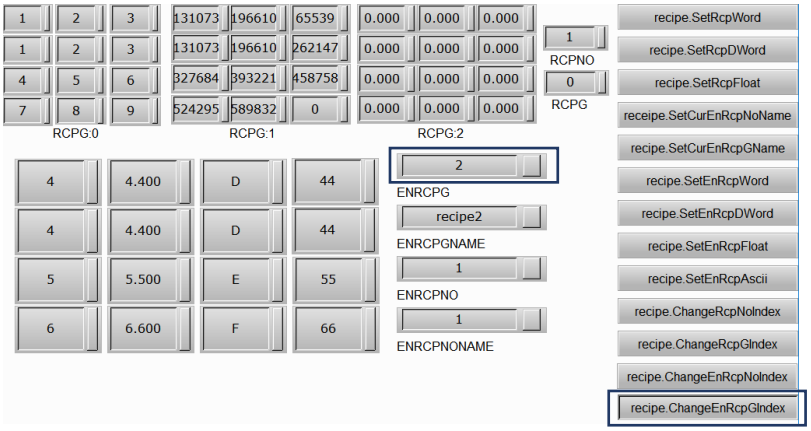
```
if mem.inter.ReadBit(100,9)==1 then
    noIdx=2
    ret = recipe.ChangeRcpNoIndex(noIdx)
end
```



範例 (recipe.SetRcpWord、recipe.SetRcpDWord、recipe.SetRcpFloat、recipe.SetCurEnRcpNoName、recipe.SetCurEnRcpGName、recipe.SetEnRcpWord、recipe.SetEnRcpDWord、recipe.SetEnRcpFloat、recipe.SetEnRcpAscii、recipe.ChangeRcpNoIndex、recipe.ChangeRcpGIndex、recipe.ChangeEnRcpNoIndex、recipe.ChangeEnRcpGIndex)	

執行結果	■ 按下$100.10，切換 RCPG 為 2。
	`if mem.inter.ReadBit(100,10)==1 then`
	`gIdx=2`
	`ret = recipe.ChangeRcpGIndex(gIdx)`
	`end`
	 
	
	■ 按下$100.11，切換 ENRCPNO 為 3。
	`if mem.inter.ReadBit(100,11)==1 then`
	`noIdx=3`
	`ret = recipe.ChangeEnRcpNoIndex(noIdx)`
	`end`
	 

範例 (recipe.SetRcpWord、recipe.SetRcpDWord、recipe.SetRcpFloat、recipe.SetCurEnRcpNoName、recipe.SetCurEnRcpGName、recipe.SetEnRcpWord、recipe.SetEnRcpDWord、recipe.SetEnRcpFloat、recipe.SetEnRcpAscii、recipe.ChangeRcpNoIndex、recipe.ChangeRcpGIndex、recipe.ChangeEnRcpNoIndex、recipe.ChangeEnRcpGIndex)	

執行結果	■ 按下$100.12，切換 ENRCPG 為 2。
	`if mem.inter.ReadBit(100,12)==1 then`
	`gIdx=2`
	`ret = recipe.ChangeEnRcpGIndex(gIdx)`
	`end`
	 

4.10 Screen (螢幕控制)

此指令可供使用者對畫面進行控制的功能，此指令包含：

指令	指令運算式	說明
Screen (螢幕控制)	screen.Open	開啟指定畫面
	screen.CloseSub	關閉指定畫面
	screen.IsOpened	確認指定畫面是否開啟
	screen.Capture	擷取畫面圖片至外部儲存裝置

以下將一一詳細介紹。

■ screen.Open：開啟指定畫面

指令名稱	screen.Open
指令運算式	Result= **screen.Open**(screen_id)
參數定義	screen_id：整數，畫面 ID；1 ~ 65535
範例	ret=screen.Open(2)
範例說明	開啟編號 ScreenID2 畫面。
回傳值	ret：成功回傳 1；失敗回傳 0

■ screen.CloseSub：關閉指定畫面

指令名稱	screen.CloseSub
指令運算式	ret = **screen.CloseSub**(screen_id)
參數定義	screen_id：整數，畫面 ID；1 ~ 65535
範例	ret=screen.CloseSub(2)
範例說明	關閉編號 ScreenID2 畫面。
回傳值	ret：成功回傳 1；失敗回傳 0

■ screen.IsOpened：確認指定畫面是否開啟

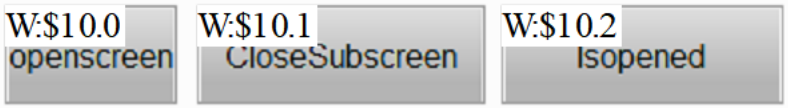
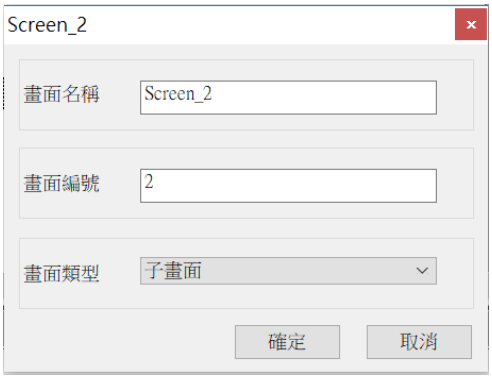
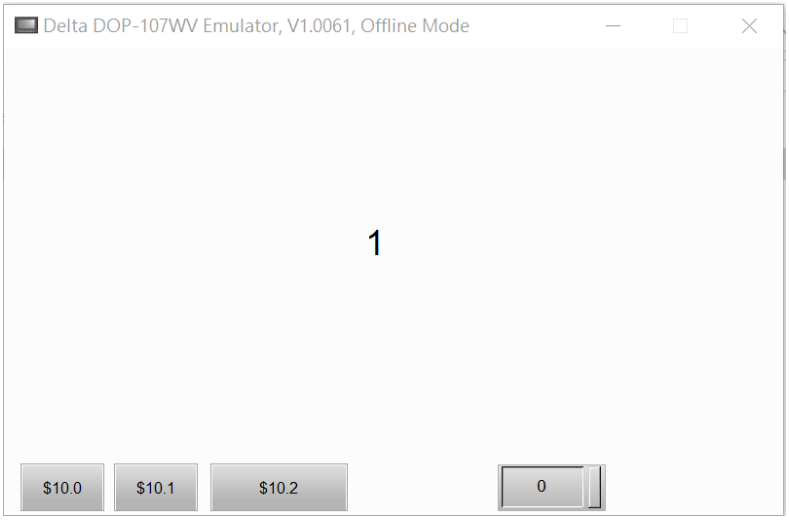
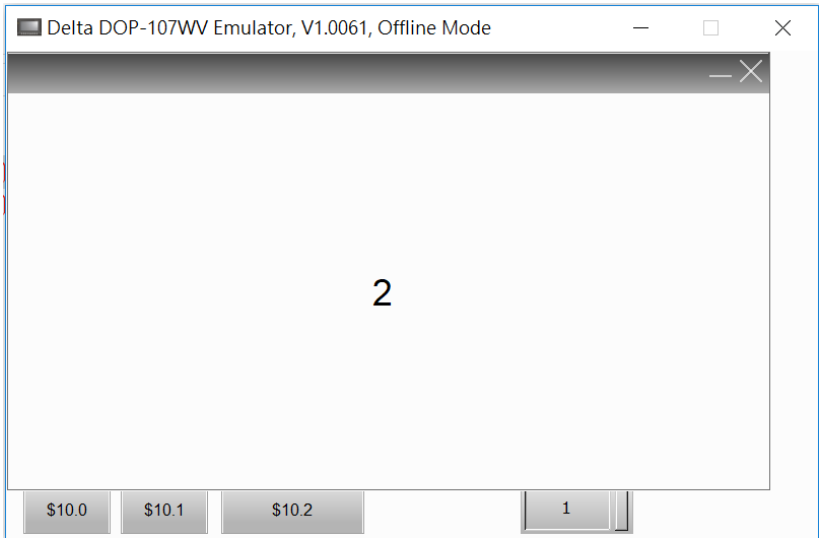
指令名稱	screen.IsOpened
指令運算式	result = **screen.IsOpened**(screen_id)
參數定義	screen_id：整數，畫面 ID；1 ~ 65535
範例	result = screen.IsOpened(1)
範例說明	確認編號 ScreenID1 畫面是否開啟。
回傳值	ret：開啟回傳 1；未開啟回傳 0

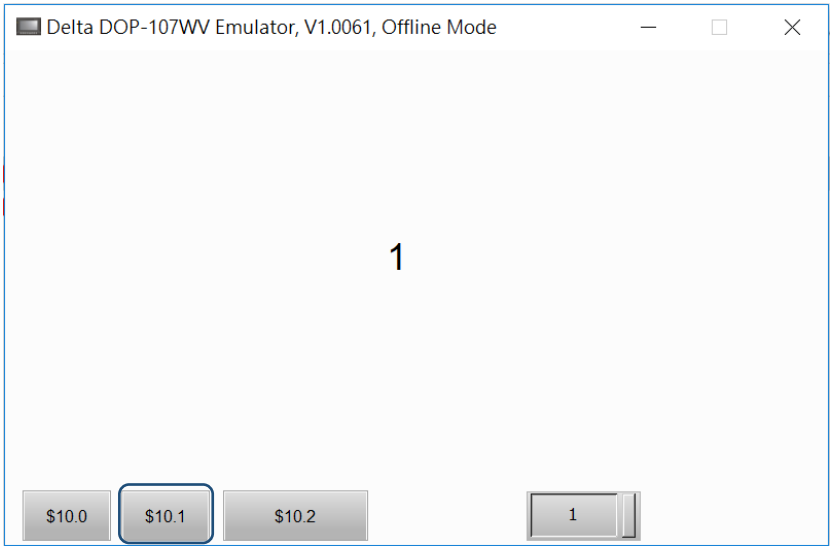
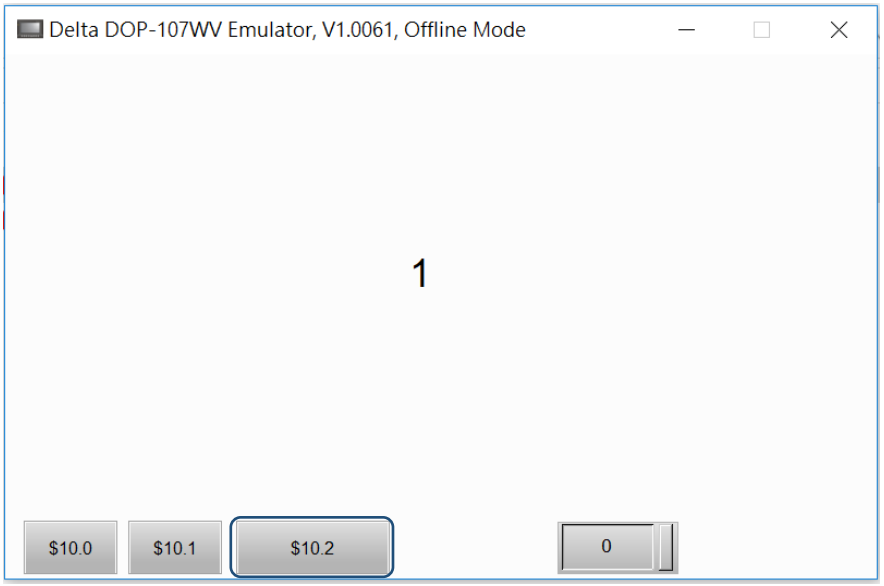
■ screen.Capture：擷取畫面圖片至外部儲存裝置

指令名稱	screen.Capture
指令運算式	Result = **screen.Capture**(disk_ID)
參數定義	disk_ID：整數，磁碟 ID；2：USB；3：SD
範例	Result = screen.Capture(2)
範例說明	擷取畫面圖片到 USB。
回傳值	ret：成功回傳 1；失敗回傳 0

範例 (Screen)	
建立 Lua 程序	■ 建立 Lua 程序。 ■ 當$10.0 被觸發時，開啟畫面 2 並將回傳值寫入$1000，最後將$10.0 關閉。 ■ 當$10.1 被觸發時，關閉子畫面 2 並將回傳值寫入$1000，最後將$10.1 關閉。 ■ 當$10.2 被觸發時，確認畫面 2 是否開啟，最後將$10.2 關閉。

```
while true do
    if (mem.inter.ReadBit(10,0)==1) then
        ret=screen.Open(2)
        mem.inter.Write(1000,ret)
        mem.inter.WriteBit(10, 0, 0)
    end
    if (mem.inter.ReadBit(10,1)==1) then
        screenID = 2
        ret = screen.CloseSub(screenID)
        mem.inter.Write(1000,ret)
        mem.inter.WriteBit(10, 1, 0)
    end
    if (mem.inter.ReadBit(10,2)==1) then
        diskID = 2
        ret = screen.IsOpened(diskID)
        mem.inter.Write(1000,ret)
        mem.inter.WriteBit(10, 2, 0)
    end
end
```

範例 (Screen)	
建立 交替型按鈕	■ 建立三個交替型按鈕，寫入記憶體位址分別為$10.0、$10.1、$10.2。 
建立畫面二	■ 於 DOPSoft 中，[畫面] → [建立新畫面]，建立新畫面。 
執行結果	■ 完成 Lua 程式的編寫和元件的建立後下載至人機，畫面如下：  ■ 當觸發$10.0，開啟畫面 2，並回傳成功值至$1000。 

範例 (Screen)	
執行結果	■ 當觸發$10.1，關閉子畫面 2，並回傳成功值至$1000。  ■ 觸發$10.2，檢查畫面 2 是否被開啟，因為畫面 2 是被關閉的狀況，所以回傳 0 至$1000。 

4.11 String (字串運算)

此指令可供使用者對字串做讀寫控制的功能，此指令包含：

指令	指令運算式	說明
String (字串運算)	string.len	計算字串長度
	string.format	字串格式化
	string.split	將字串分開
	string.find	尋找字串位置
	string.sub	尋找字串
	string.rep	重複字串
	string.trim	將字串去除前後空白
	string.lower	將字串轉換成小寫
	string.upper	將字串轉換成大寫
	string.reverse	將字串反轉
	string.byte	將字串轉換為十進位數值
	string.char	將十進位數值轉換為字串
	string.gsub	以字串取代指定的字串
	string.gmatch	於字串中尋找模式字串匹配的部分，找到匹配的參數後回傳 註：需搭配 for 迴圈。
	string.match	於字串中尋找模式字串匹配的部分，找到匹配的參數後回傳 註：string.match 和 string.gmatch 的差異為 string.match 搜尋完匹配的字串後會停止，string.gmatch 則不會。

以下將一一詳細介紹。

■ string.len：計算字串長度

指令名稱	string.len
指令運算式	length = **string.len**(string_src)
參數定義	string_src：ASCII 字串
範例	v1 = string.len("ABCDE")
範例說明	計算字串"ABCDE"的長度，v1 為 5。
回傳值	v1：字串長度

■ string.format：字串格式化

指令名稱	string.format
指令運算式	string_ret= **string.format**(string_fmt, var1, var2, …)
參數定義	string_fmt：ASCII 字串；變數：%d、%x、%X、%s、%c、%f var1：第一個變數 var2：第二個變數

範例

%運算子	範例	結果
%d	v1 = string.format("%d", 10)	v1=10
%X/%x	v1 = string.format("%x", 15)	v1=f
%s	v1 = string.format("%s, "posheng")	v1= posheng
%c	v1 = string.format("%c, 0x30)	v1=0
%o	v1 = string.format("%o, 9)	v1=11
%u	v1 = string.format("%u, 3.14)	v1=3
%E/%e	v1 = string.format("%e", 1000)	v1=1.000000e+03
%f	v1 = string.format("%3.3f", 1.12345)	v1=1.123
%G/%g	v1 = string.format("%g", 1000)	v1=1000
	v1 = string.format("%g", 1000000)	v1= 1e+006
%q	v1 = string.format("%q", "posheng")	v1="posheng"

範例說明

字串格式化方法是使用%運算子，前面放置輸出的文字樣板，後面放置要處理的資料。

%運算子	功能
%d	十進位整數方式輸出字串。
%X/%x	以十六進位整數方式輸出字串。
%s	以字串方式輸出字串。
%c	以字元方式輸出字串。
%o	以八進位整數方式輸出字串。
%u	以無符號整數方式輸出字串。
%E/%e	以科學記號法方式輸出字串。
%f	以浮點數方式輸出字串。
%G/%g	以%e 及%f 中較短的格式輸出字串。
%q	將字串以完整字串方式輸出。

回傳值	v1：格式組合後的字串

■ string.split：將字串分開

指令名稱	string.split
指令運算式	tName = **string.split**(src, ",")
參數定義	src：ASCII 字串
範例	src="John,Andy,Mike" tName = string.split(src, ",")
範例說明	將字串以“,”分開，tName[1]="John"、tName[2]="Andy"、tName[3]="Mike"。
回傳值	tName：陣列

■ string.find：尋找字串位置

指令名稱	string.find
指令運算式	found_index, end_index = **string.find**(string, pattern, [start_index])
參數定義	string：ASCII 原始字串 pattern：目標字串 start_index：整數值，字串起始索引，索引基底為 1，可省略
範例	src="ABCDE" v1, v2= string.find(src, "BCD")
範例說明	於變數 src 尋找字串 BCD，v1 為 2、v2 為 4。
回傳值	v1：整數值；找到的字串起始索引，索引基底為 1 v2：整數值；找到的字串結束索引，索引基底為 1

■ string.sub：尋找字串

指令名稱	string.sub
指令運算式	string_ret = **string.sub**(string_src, start_index, [end_index])
參數定義	string_src：ASCII 字串 start_index：整數值；字串起始索引，索引基底為 1 end_index：整數值；字串結束索引，索引基底為 1，可省略
範例	v1 = string.sub("ABCDE", 2, 4)
範例說明	於字串“ABCDE”取得第 2 個到第 4 個 bytes 的字串，v1= “BCD”。
回傳值	v1：字串；字串的子集合

■ string.rep：重複字串

指令名稱	string.rep
指令運算式	destString = **string.rep**(srcString, repeatCount)
參數定義	srcString：字串 repeatCount：整數值，重複次數
範例	v1 = string.rep("AB+", 2)
範例說明	重複字串“AB+”兩次，v1= “AB+AB+”。
回傳值	v1：字串；結果字串

■ string.trim：將字串去除前後空白

指令名稱	string.trim
指令運算式	trimStr = **string.trim**(string_src)
參數定義	string_src：ASCII 字串
範例	src=" ABCDE " v1 = string.trim(src)
範例說明	將字串“ABCDE ”去除前後空白字串，v1= “ABCDE”。
回傳值	v1：字串；去除前後空白的字串

■ string.lower：將字串轉換成小寫

指令名稱	string.lower
指令運算式	destString = **string.lower**(srcString)
參數定義	srcString：字串
範例	name = string.lower(“Posheng”)
範例說明	將字串“Posheng”改為小寫，name=“posheng”。
回傳值	name：字串；轉換為小寫的字串

■ string.upper：將字串轉換成大寫

指令名稱	string.upper
指令運算式	destString = **string.upper**(srcString)
參數定義	srcString：字串
範例	name = string.upper("Posheng”)
範例說明	將字串“Posheng”改為大寫，name= “POSHENG”。
回傳值	name：字串；轉換為大寫的字串

■ string.reverse：將字串反轉

指令名稱	string.reverse
指令運算式	destString = **string.reverse**(srcString)
參數定義	srcString：字串
範例	v1 = string.reverse("ABCDE")
範例說明	將字串"ABCDE"反轉，v1=“EDCBA”。
回傳值	v1：字串，反轉後的字串

■ string.byte：將字串轉換為十進位數值

指令名稱	string.byte
指令運算式	num1, num2, … = **string.byte**(string, [start_index, [end_index]])
參數定義	string：ASCII 字串 start_index：整數值；字串起始索引，索引基底為 1 end_index：整數值；字串結束索引，索引基底為 1，可省略
範例	v1 = string.byte("ABCDE", 1)
範例說明	將字串"ABCDE"的第一個 byte 轉換為 10 進位，v1=65。
回傳值	v1：回傳的整數值

■ string.char：將十進位數值轉換為字串

指令名稱	string.char
指令運算式	destString = **string.char**(il, i2, …)
參數定義	i1：整數值，ASCII 對應的數值 i2：整數值，ASCII 對應的數值
範例	v1 = string.char(65,66,67)
範例說明	將 10 進位的數值 65、66、67 轉成字串，v1=“ABC”。
回傳值	v1：組合後的字串

■ string.gsub：以字串取代指定的字串

指令名稱	string.gsub
指令運算式	destString = **string.gsub**(srcString, patternString, replacedString)
參數定義	srcString：字串 patternString：尋找欲更改的字串 replacedString：將找到的字串，用此 replacedString 變數取代
範例	s = string.gsub("ABCDE", "C", "x")
範例說明	將字串“ABCDE”的“C”以 “x”取代，s= “ABxDE”。
回傳值	s：取代後的最後的字串

■ string.gmatch：於字串中尋找模式字串匹配的部分，找到匹配的參數後回傳

指令名稱	string.gmatch
指令運算式	findingIterator = **string.gmatch**(srcString, pattern)
參數定義	srcString：字串 pattern：字串，模式字串
範例	for word in string.gmatch("Hello world", "%a+") do w = word end 註：此指令需要搭配 for 迴圈。
範例說明	方法是使用%模式字串，根據前方的字串，透過模式字串，輸出相對應內容。 迴圈執行兩次，變數 w 分別為”Hello”和”world”。 註：%a 為尋找字串，%a+為尋找字串至結束。
回傳值	findingIterator：字串；結果字串

■ string.match：於字串中尋找模式字串匹配的部分，找到匹配的參數後回傳(string.match 和 string.gmatch 的差異為 string.gmatch 會回傳所有匹配的字串，而 string.match 只會回傳第一組匹配的字串)

指令名稱	string.match
指令運算式	destString = **string.match**(srcString, pattern)
參數定義	srcString：字串 pattern：字串，模式字串
範例	範例一：word =string.match("Hello world", "%a") 範例二：s1, s2, s3 = string.match("1/22/333","(%d)/(%d+)/(%d)")
範例說明	範例一結果：word=“Hello”。 範例二結果：s1=1、s2=22、s3=3。 註：%d 為尋找數字，%d+為尋找數字至結束。
回傳值	destString：字串；結果字串

4.12 System library (系統參數)

System library 可供使用者針對系統參數做讀和寫的動作，此指令包含：

指令	指令運算式	說明
System library (系統參數)	sys.Sleep	系統延遲
	sys.GetTick	取得截至目前為止人機的總開機時間
	sys.GetInterParam	取得人機內部參數數值
	sys.BuzzerOn	開啟蜂鳴器
	sys.GetDate	取得當前時間
	sys.GetDateString	取得當前時間 (單位：字串)
	sys.GetDays	取得 1970/01/01 到設定日期所經過的天數
	sys.GetSecs	取得 00:00:00 到設定時間所經過的秒數
	sys.GetTime	取得系統時間
	sys.ToDate	取得 1970/01/01 經過所設定天數後的日期
	sys.ToTime	取得 00:00:00 經過所設定秒數後的時間
	sys.GetDiskSpace	取得外部儲存裝置空間

以下將一一詳細介紹。

■ sys.Sleep：系統延遲

指令名稱	sys.Sleep
指令運算式	**sys.Sleep**(time)
參數定義	time：整數值(ms)
範例	sys.Sleep(1000)
範例說明	系統延遲 1000 ms。
回傳值	無回傳值

■ sys.GetTick：取得截至目前為止人機的總開機時間

指令名稱	sys.GetTick
指令運算式	startTick=**sys.GetTick**()
參數定義	無參數
範例	startTick=sys.GetTick()
範例說明	startTick 為截至目前為止人機的總開機時間，單位：毫秒。
回傳值	startTick：時間，單位：毫秒

■ sys.GetInterParam：取得人機內部參數數值

指令名稱	sys.GetInterParam
指令運算式	value, ret =**sys.GetInterParam**("paraName")
參數定義	paraName：字串；內部系統參數名稱，同元件位址內的 internal parameter 名稱
範例	value, ret = sys.GetInterParam("TP_Y")
範例說明	value 為當前使用者按壓實體人機的 Y 軸座標位置。
回傳值	y：整數或字串；視內部系統參數而定 ret：成功時回傳 1；失敗時回傳 0

■ sys.BuzzerOn：開啟蜂鳴器

指令名稱	sys.BuzzerOn
指令運算式	**sys.BuzzerOn**(buzzerType)
參數定義	buzzerType：0：關閉蜂鳴器；1：開啟蜂鳴器；2：持續開啟蜂鳴器
範例	sys.BuzzerOn(1)
範例說明	開啟蜂鳴器。
回傳值	無回傳值

■ sys.GetDate：取得當前時間

指令名稱	sys.GetDate
指令運算式	year, month, day, week = **sys.GetDate**()
參數定義	無參數
範例	year, month, day, week = sys.GetDate()
範例說明	取得當前時間(年、月、日、星期)。
回傳值	year 為年、month 為月份、day 為日期、week 為 0 ~ 6 分別代表星期一到星期日。

■ sys.GetDateString：取得當前時間 (單位：字串)

指令名稱	sys.GetDateString
指令運算式	dateStr = **sys.GetDateString**()
參數定義	無參數
範例	dateStr = sys.GetDateString()
範例說明	取得當前時間(以字串表示)。
回傳值	dateStr 為系統時間(字串)，年/月/日。 如果系統時間為 2019/01/31，變數 dateStr 則為 "2019/01/31"。

■ sys.GetDays：取得 1970/01/01 到設定日期所經過的天數

指令名稱	sys.GetDays
指令運算式	days = **sys.GetDays**(year, month, day)
參數定義	year：整數值；年 month：整數值；月 day：整數值；日
範例	days = sys.GetDays(1970, 1, 2)
範例說明	days 為 1970/01/01 到 1970/01/02 所經過的天數，days 為 1。
回傳值	days：成功時回傳所經過的天數；失敗回傳 nil

■ sys.GetSecs：取得 00:00:00 到設定時間所經過的秒數

指令名稱	sys.GetSecs
指令運算式	Result = **sys.GetSecs**(hour, minute, second)
參數定義	hour：整數值；時 minute：整數值；分 second：整數值；秒
範例	secs = sys.GetSecs(0, 0, 59)
範例說明	secs 為 00:00:00 到 00:00:59 所經過的秒數，secs 為 59。
回傳值	secs：成功時回傳所經過的秒數；失敗回傳 nil

■ sys.GetTime：取得系統時間

指令名稱	sys.GetTime
指令運算式	h, m, s = **sys.GetTime**()
參數定義	無參數
範例	h, m, s = sys.GetTime()
範例說明	取得系統時間。
回傳值	如果系統時間為 11 點 20 分 35 秒，h 為 11、m 為 20、s 為 35。

■ sys.ToDate：取得 1970/01/01 經過所設定天數後的日期

指令名稱	sys.ToDate
指令運算式	year, month, day = **sys.ToDate**(days)
參數定義	days：所經過的天數
範例	year, month, day = sys.ToDate(5)
範例說明	year、month、day 為 1970/01/01 經過 5 天數後的日期，year 為 1970、month 為 1、day 為 6。
回傳值	year、month、day 分別為 1970/01/01 經過所設定天數後的年、月、日。

■ sys.ToTime：取得 00:00:00 經過所設定秒數後的時間

指令名稱	sys.ToTime
指令運算式	hour, minute, second = **sys.ToTime**(seconds)
參數定義	seconds：所經過的秒數
範例	hour, minute, second = sys.ToTime(61)
範例說明	hour、minute、second 為 00:00:00 經過 61 秒後的時間，hour 為 0、minute 為 1、second 為 1。
回傳值	hour、minute、second 分別為 00:00:00 經過所設定秒數後的時、分、秒。

■ sys.GetDiskSpace：取得外部儲存裝置空間

指令名稱	sys.GetDiskSpace
指令運算式	ret, total, free = **sys.GetDiskSpace**(disk_id)
參數定義	disk_id：整數；2：USB；3：SD
範例	ret, total, free = sys.GetDiskSpace(2)
範例說明	取得 USB 儲存裝置的總空間和剩餘空間。
回傳值	ret：成功時回傳 1；失敗時回傳負數 回傳值 / 說明 -1 / 參數設定錯誤 -106 / 磁碟未備妥 total：整數，總磁碟空間(MB) free：整數，剩餘磁碟空間(MB)

4.13 Serial Port communication (COM 自由通訊)

自由通訊意指當兩個裝置沒有共同的通訊協定時，我們可以透過自由通訊的方式做溝通，台達 HMI 提供 COM、TCP 以及 UDP 建立自由通訊連線及交握資料。以下將介紹 COM 的自由通訊指令，以及如何透過此指令建立連線。COM 的自由通訊指令語法包含：

指令	指令運算式	說明
Serial Port communication (COM 自由通訊)	com.Open	開啟 com 接口通訊
	com.ReadChars	從指定通訊埠讀取字元
	com.WriteChars	於指定通訊埠寫入字元
	com.ClearBuffer	清除緩衝區資料
	com.StationCheck	透過選擇通訊埠以及站號來確認站號通訊是否成功
	com.Close	關閉通訊埠
	com.CheckAlive	透過選擇通訊參數來確認通訊是否成功
	com.StationOn	站號啟動
	com.StationOff	站號關閉
	com.GetStatus	取得 com 接口狀態

以下將一一詳細介紹。

■ com.Open：開啟 com 接口通訊

指令名稱	com.Open
指令運算式	ret = **com.Open**(com_num, interface, databits, parity, stopbits, baudrate, flowcontrol)
參數定義	com_num：整數值；串列通訊接口號碼，COM1 填 1，COM2 填 2，依此類推 interface：字串；“RS232”、“RS422”、“RS485” databits：整數值；7、8 parity：字串；“NONE”、“ODD”、“EVEN”、“MARK”、“SPACE” stopbits：整數值；1、2 baudrate：整數值；9600，… flowcontrol：字串；“OFF”、“CTS_RTS”
範例	ret = com.Open(1, "RS232", 8, "EVEN", 1, 19200, "OFF")
範例說明	開啟通訊埠 com1 通訊，通訊介面為 RS232、資料位元為 8、同位元為“EVEN”、停止位元為 1、鮑率為 19200、流量控制為 OFF。
回傳值	ret：成功時回傳 1；不合法參數回傳-1；開啟 com 接口失敗時回傳-101

■ com.ReadChars：從指定通訊埠讀取字元 (COM)

指令名稱	com.ReadChars
指令運算式	bytes_read, buffer = **com.ReadChars**(com_num, len, timeout)
參數定義	com_num：整數值；串列通訊接口號碼，COM1 填 1，COM2 填 2，依此類推 len：整數值；ASCII 字串長度 timeout：整數值；延遲時間 (單位：ms)
範例	bytes_read, buffer = com.ReadChars(1, 10, 1000)
範例說明	以 1000 ms 通訊延遲時間及資料長度 10 bytes 從通訊埠 com1 來讀取字元。
回傳值	bytes_read：成功時回傳讀取資料的長度(bytes)；失敗時回傳負數

回傳值	說明
0	讀取逾時
-1	參數設定錯誤
-100	連接埠未開啟
-102	讀取失敗

buffer：讀回來的字串

■ com.WriteChars：於指定通訊埠寫入字元 (COM)

指令名稱	com.WriteChars
指令運算式	ret = **com.WriteChars**(com_num, buffer, len, timeout)
參數定義	com_num：整數值；串列通訊接口號碼，COM1 填 1，COM2 填 2，… buffer：ASCII 字串 len：整數值，ASCII 字串長度 timeout：整數值，延遲時間 (單位：ms)
範例	ret = com.WriteChars(1, “posheng”, 6, 1000)
範例說明	以 1000 ms 通訊延遲時間及長度 6 bytes 寫入字串 “posheng”至通訊埠 com1。
回傳值	bytes_read：成功時回傳寫入的資料長度(bytes)；失敗時回傳負數 buffer：寫入的字串

回傳值	說明
-1	參數設定錯誤
-100	連接埠未開啟
-103	寫入失敗

■ com.ClearBuffer：清除緩衝區資料

指令名稱	com.ClearBuffer
指令運算式	ret = **com.ClearBuffer**(com_num, clear_type)
參數定義	com_num：整數值；串列通訊接口號碼，COM1 填 1，COM 填 2，依此類推 clear_type：整數值；1：清除讀取緩衝區 (read buffer)；0：清除寫入緩衝區 (write buffer)
範例	ret = com.ClearBuffer(1, 1)
範例說明	清除通訊埠 com1 的讀取緩衝區資料。
回傳值	ret：成功時回傳 1；失敗時回傳負數

回傳值	說明
-1	參數設定錯誤
-100	連接埠未開啟
-104	清除緩衝失敗

■ com.StationCheck：透過選擇通訊埠以及站號來確認站號通訊是否成功

指令名稱	com.StationCheck
指令運算式	ret = **com.StationCheck**(com_num, station)
參數定義	com_num：整數值，串列通訊接口號，COM1 填 1，COM 填 2 station：整數值，站號
範例	ret = com.StationCheck(1, 1)
範例說明	確認通訊埠 com1 的站號 1 通訊是否成功。
回傳值	

回傳值	說明
1	成功
0	失敗
-1	不合法參數

■ com.Close：關閉通訊埠

指令名稱	com.Close
指令運算式	ret = **com.Close**(com_num)
參數定義	com_num：整數值；串列通訊接口號碼，COM1 填 1，COM 填 2，依此類推
範例	ret = com.Close(1)
範例說明	關閉 COM 1 連線。
回傳值	ret：成功時回傳 1；參數不合法回傳-1

■ com.CheckAlive：透過選擇通訊參數來確認通訊是否成功

指令名稱	com.CheckAlive
指令運算式	ret = **com.CheckAlive**(modbus_mode, com_num, interface, databits, parity, stopbits, baudrate, flowcontrol, station, timeout)
參數定義	modbus_mode：字串；modbus 模式“MODBUS_ASCII”、“MODBUS_RTU” com_num：整數值；串列通訊接口號碼，COM1 填 1，COM2 填 2，依此類推 interface：字串，“RS232”、“RS422”、“RS485” databits：整數值；7、8 parity：字串；“NONE”、“ODD”、“EVEN”、“MARK”、“SPACE” stopbits：整數值；1、2 baudrate：整數值；9600，… flowcontrol：字串；“OFF”、“CTS_RTS” station：整數值；站號：0 ~ 255 timeout：整數值；逾時值(ms)：0 ~ 15000
範例	ret = com.CheckAlive("MODBUS_ASCII", 1, "RS485", 8, "EVEN", 1, 19200, "OFF", 1, 1000)
範例說明	於通訊埠 COM1，以通訊協議為 MODBUS_ASCII，通訊介面為 RS485、資料位元為 8、同位元為“EVEN”、停止位元為 1、鮑率為 19200、流量控制為 OFF 之設定，送出指令確認該通訊是否存在，等待回覆時間為 1000 ms。 註：此指令只支援台達 PLC。
回傳值	ret：成功時回傳 1；失敗時回傳以下對應數值

回傳值	說明
0	失敗
-1	不合法參數
-101	開始 COM 失敗

■ com.StationOn：站號啟動

指令名稱	com.StationOn
指令運算式	ret = **com.StationOn**(com_num, station)
參數定義	com_num：整數值；串列通訊接口號碼，COM1 填 1，COM2 填 2，依此類推 station：整數值；站號：0 ~ 255
範例	ret = com.StationOn(1, 1)
範例說明	開啟通訊連線1的站號1控制器，此時人機便可與該站控制器通訊。 註：此指令與com.Open開啟的通訊無關。
回傳值	ret：成功時回傳 1；參數不合法回傳-1

■ com.StationOff：站號關閉

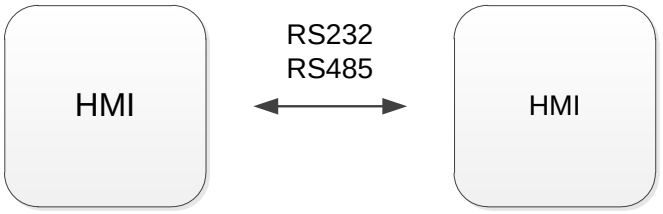
指令名稱	com.StationOff
指令運算式	ret = **com.StationOff**(com_num, station)
參數定義	com_num：整數值；串列通訊接口號碼，COM1 填 1，COM2 填 2，依此類推 station：整數值；站號：0 ~ 255
範例	ret = com.StationOff(1, 1)
範例說明	關閉通訊連線1的站號1控制器，此時人機便無法與該站控制器通訊。 註：此指令與com.Open開啟的通訊無關。
回傳值	ret：成功時回傳 1；參數不合法回傳-1

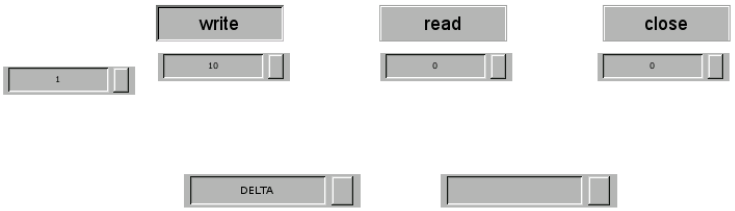
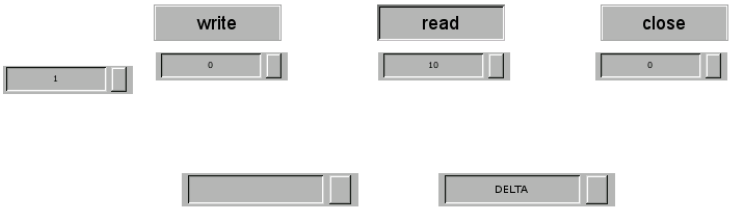
■ com.GetStatus：取得 com 接口狀態

指令名稱	com.GetStatus
指令運算式	ret = **com.GetStatus**(com_num)
參數定義	com_num：整數值；串列通訊接口號碼，COM1 填 1，COM2 填 2，依此類推
範例	ret = com.GetStatus(1)
範例說明	確認自由通訊的通訊連線1的通訊狀況。
回傳值	ret：成功時回傳 1；失敗回傳負數

回傳值	說明
-1	不合法參數
-100	連接埠未開啟
-101	開啟 COM 接口失敗

參數設定內容			
變數	**選項**	**選項內容**	**對應代號**
modbus_mode	通訊格式	MODBUS_ASCII	"MODBUS_ASCII"
		MODBUS_RTU	"MODBUS_RTU"
com_num	通訊埠	COM 1	1
		COM 2	2
		COM 3	3
interface	通訊界面	RS232	"RS232"
		RS422	"RS422"
		RS485	"RS485"
databits	資料位元	7 Bits	7
		8 Bits	8
parity	同位元	NONE	"NONE"
		ODD	"ODD"
		EVEN	"EVEN"
		MARK	"MARK"
		SPACE	"SPACE"
stopbits	停止位元	1 Bit	1
		2 Bits	2
baudrate	鮑率	300	300
		600	600
		900	900
		1200	1200
		2400	2400
		4800	4800
		9600	9600
		14400	14400
		19200	19200
		28800	28800
		38400	38400
		57600	57600
		115200	115200
flowcontrol	流量控制	OFF	"OFF"
		CTS_RTS	"CTS_RTS"
station	站號	1 ~ 255	1 ~ 255
timeout	通訊時間	0 ~ 15000 (ms)	0 ~ 15000

<table>
<tr><th colspan="2">範例(COM 自由通訊)</th></tr>
<tr><td>建立硬體連線</td><td>■ 接線概念圖
 
</td></tr>
<tr><td>建立 Lua 自由通訊指令</td><td>指令如下：
<pre>var1 = com.Open(2, "RS485", 8, "NONE", 1, 9600, "OFF")
mem.inter.Write(11, var1)
while true do
    if (mem.inter.ReadBit(50,0)==1) then
        buffer = mem.inter.ReadAscii(1000,10)
        ret = com.WriteChars(2, buffer, string.len(buffer), 3000)
        mem.inter.Write(12, ret)
    end
    if (mem.inter.ReadBit(60,0)==1) then
        var10, buffer = com.ReadChars(2, 10, 3000)
        mem.inter.WriteAscii(4,buffer,string.len(buffer))
        mem.inter.Write(13, var10)
    end
    if (mem.inter.ReadBit(70,0)==1) then
        ret = com.Close(2)
        mem.inter.Write(14, ret)
    end
end</pre>
■ 開啟 COM 通訊埠，並將回傳值寫入記憶體位址$1。
<pre>var1 = com.Open(2, "RS485", 8, "NONE", 1, 9600, "OFF")
mem.inter.Write(11, var1)</pre>
■ 當觸發$50.0 時，以長度 10 bytes 讀取$1000 記憶體字串為變數的 buffer，將此變數 buffer 寫入至緩衝區，並將回傳值寫入記憶體位址$12。
<pre>if (mem.inter.ReadBit(50,0)==1) then
    buffer = mem.inter.ReadAscii(1000,10)
    ret = com.WriteChars(2, buffer, string.len(buffer), 3000)
    mem.inter.Write(12, ret)
end</pre>
■ 當觸發$60.0 時，於通訊埠 2 以資料長度 10 bytes 讀取資料為變數的 buffer，將讀到的變數 buffer 寫入到$4，並將回傳值寫入記憶體位址$13。
<pre>if (mem.inter.ReadBit(60,0)==1) then
    var10, buffer = com.ReadChars(2, 10, 3000)
    mem.inter.WriteAscii(4,buffer,string.len(buffer))
    mem.inter.Write(13, var10)
end</pre>
■ 當觸發$70.0 時，關閉通訊埠 com2，並將回傳值寫入$14。
<pre>if (mem.inter.ReadBit(70,0)==1) then
    ret = com.Close(2)
    mem.inter.Write(14, ret)
end</pre></td></tr>
</table>

範例(COM 自由通訊)	
建立交替型按鈕、數值輸入、文數字輸入元件	■ 建立交替型按鈕，寫入記憶體位址為$50.0、$60.0、$70.0。 ■ 建立數值輸入元件，寫入記憶體位址為$11、$12、$13、$14。 ■ 建立文數字輸入元件，寫入記憶體位址為$1000 和$4。
執行結果	■ 完成 Lua 程式的編寫和元件的建立後，下載專案至人機(兩台人機畫面一致)。 ■ 載入至人機後，開啟 COM 通訊埠成功，回傳成功值於$11。 ■ 於$1000 寫入字串 DELTA，並觸發$50.0，寫入字串至緩衝區。  ■ 於第二台人機上，觸發$60.0，讀取緩衝區資料，並寫入記憶體位址$4。  ■ 透過觸發$70.0，關閉通訊埠 com2。

4.14 TCP communication (TCP 自由通訊)

自由通訊意指當兩個裝置沒有共同的通訊協定時，我們可以透過自由通訊的方式做溝通，台達 HMI 提供 COM、TCP 以及 UDP 建立自由通訊連線及交握資料。以下將介紹 TCP 的自由通訊指令，以及如何透過此指令建立連線。TCP 的自由通訊指令語法包含：

指令	指令運算式	說明
TCP communication (TCP 自由通訊)	tcp.Open	開啟 TCP 網路通訊
	tcp.Read	讀取字元 (TCP)
	tcp.Write	寫入字元 (TCP)
	tcp.Close	關閉連線 (TCP)
	tcp.GetMaxCount	取得最大連線數量 (TCP)
	tcp.GetRunCount	取得正在運行的 socket 數量 (TCP)
	tcp.GetStatus	確認 socket 的通訊狀況 (TCP)

以下將一一詳細介紹。

■ tcp.Open：開啟 TCP 網路通訊

指令名稱	tcp.Open
指令運算式	Socket = **tcp.Open**(ip, port)
參數定義	ip：字串；“192.168.0.1”，… port：整數
範例	Socket = tcp.Open(“192.168.0.1”, 502)
範例說明	開啟網路通訊；ip 為 192.168.0.1；port 通訊埠為 502。
回傳值	Socket：成功時回傳 socket 編號；失敗回傳負值

回傳值	說明
> 0	成功
-1	不合法參數
-101	開啟 socket 失敗
-145	已開啟太多個 socket

■ tcp.Read：讀取字元 (TCP)

指令名稱	tcp.Read
指令運算式	bytes_read, buffer = **tcp.Read**(socket, len, timeout)
參數定義	socket：整數；1 ~ 8 len：整數；字串長度 timeout：整數；延遲時間 (單位：ms)
範例	bytes_read, buffer = tcp.Read(1, 10, 1000)
範例說明	透過選擇接口 1，以所設定之通訊延遲時間 1000 ms 以及資料長度 10 bytes 來讀取字元。
回傳值	bytes_read：成功時回傳讀取資料的長度(bytes)；失敗時回傳負數 buffer：讀回來的字串

回傳值	說明
> 1	成功
-1	不合法參數
-100	Socket 未開啟
-102	讀取失敗

■ tcp.Write：寫入字元 (TCP)

指令名稱	tcp.Write
指令運算式	ret = **tcp.Write**(socket, buffer, len, timeout)
參數定義	socket：整數；1 ~ 8 buffer：字串 len：整數；字串長度 timeout：整數；延遲時間(ms)
範例	ret = tcp.Write(1, “abc123”, 6, 1000)
範例說明	以長度 6 bytes 送出字串 abc123 至通訊埠 1。
回傳值	ret：成功時回傳 1；失敗時回傳負值

回傳值	說明
-1	不合法參數
-100	Socket 未開啟
-103	寫入失敗

■ tcp.Close：關閉連線 (TCP)

指令名稱	tcp.Close
指令運算式	ret = **tcp.Close**(socket)
參數定義	socket：整數 1 ~ 8
範例	ret = tcp.Close(1)
範例說明	關閉 TCP 自由通訊 socket1 連線。
回傳值	ret：成功時回傳 1；失敗時回傳 0 或負值

回傳值	說明
-1	不合法參數
-100	Socket 未開啟

■ tcp.GetMaxCount：取得最大連線數量(TCP)

指令名稱	tcp.GetMaxCount
指令運算式	count = **tcp.GetMaxCount**()
參數定義	無參數
範例	count = tcp.GetMaxCount()
範例說明	取得 TCP 自由通訊最大連線個數。
回傳值	count：系統最大 socket 個數

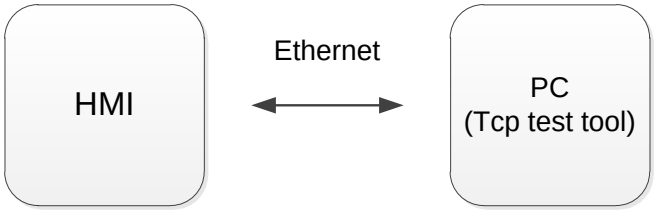
■ tcp.GetRunCount：取得正在運行的 socket 數量(TCP)

指令名稱	tcp.GetRunCount
指令運算式	count = **tcp.GetRunCount**()
參數定義	無參數
範例	count = **tcp.GetRunCount**()
範例說明	取得 TCP 自由通訊正在運行的 socket 數。
回傳值	count：正在運行的 socket 數

■ tcp.GetStatus：確認 socket 的通訊狀況(TCP)

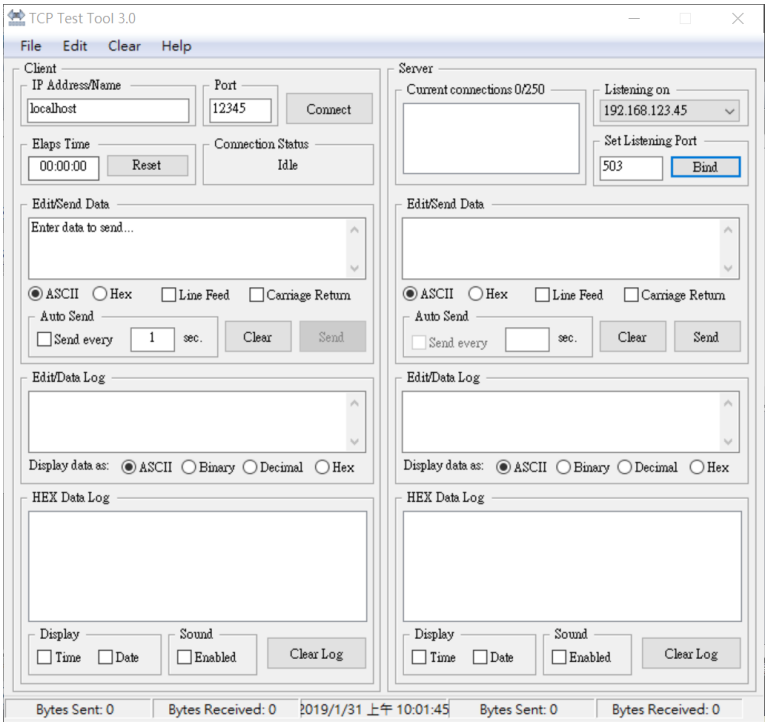
指令名稱	tcp.GetStatus
指令運算式	status = **tcp.GetStatus**(socket)
參數定義	socket：整數 1 ~ 8
範例	status = tcp.GetStatus(1)
範例說明	確認 TCP 自由通訊 socket1 的通訊狀況。
回傳值	Status：開啟時回傳 1；關閉時回傳 0

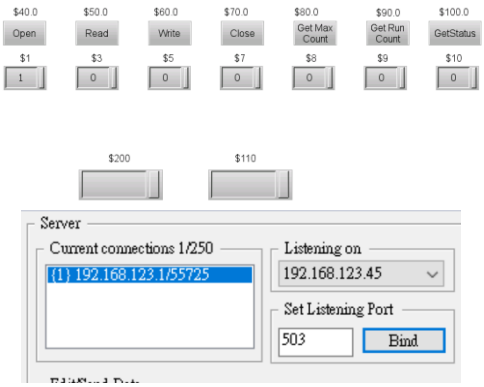
以下為自由通訊範例詳細說明。

<table>
<tr><th colspan="2">TCP 自由通訊</th></tr>
<tr><td>建立硬體連線</td><td>■ 接線概念圖，以人機作為 Client，電腦當作 Server 接收 Client 資料。
 
</td></tr>
<tr><td>建立 Lua 自由通訊指令</td><td>指令如下：
<pre>
while true do
    if (mem.inter.ReadBit(40,0)==1) then
        ip = "192.168.123.45"
        port = 503
        socket = tcp.Open(ip, port)
        mem.inter.Write(1,Socket)
            if socket==1 then
                mem.inter.WriteBit(40,0,0)
            end
    end
    if (mem.inter.ReadBit(50,0)==1) then
        len = 5
        timeout = 3000
        bytes_read, buffer = tcp.Read(socket, len, timeout)
        mem.inter.Write(3,bytes_read)
        mem.inter.WriteAscii(200,buffer,string.len(buffer))
    end
    if (mem.inter.ReadBit(60,0)==1) then
        buffer = mem.inter.ReadAscii(110,10)
        ret = tcp.Write(1, buffer, string.len(buffer), 1000)
        mem.inter.Write(5, ret,string.len(ret))
    end
    if (mem.inter.ReadBit(70,0)==1) then
        socket = 1
        ret = tcp.Close(socket)
        mem.inter.Write(7,ret)
    end
    if (mem.inter.ReadBit(80,0)==1) then
        count = tcp.GetMaxCount()
        mem.inter.Write(8,count)
    end
    if (mem.inter.ReadBit(90,0)==1) then
        count = tcp.RunCount(socket)
        mem.inter.Write(9,count)
    end
    if (mem.inter.ReadBit(100,0)==1) then
        status = tcp.GetStatus(socket)
        mem.inter.Write(10,status)
    end
end
</pre></td></tr>
</table>

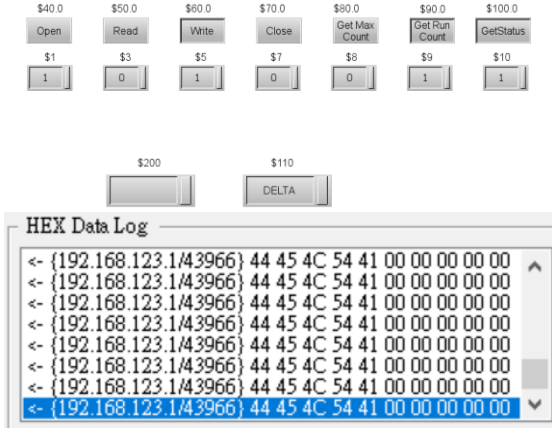
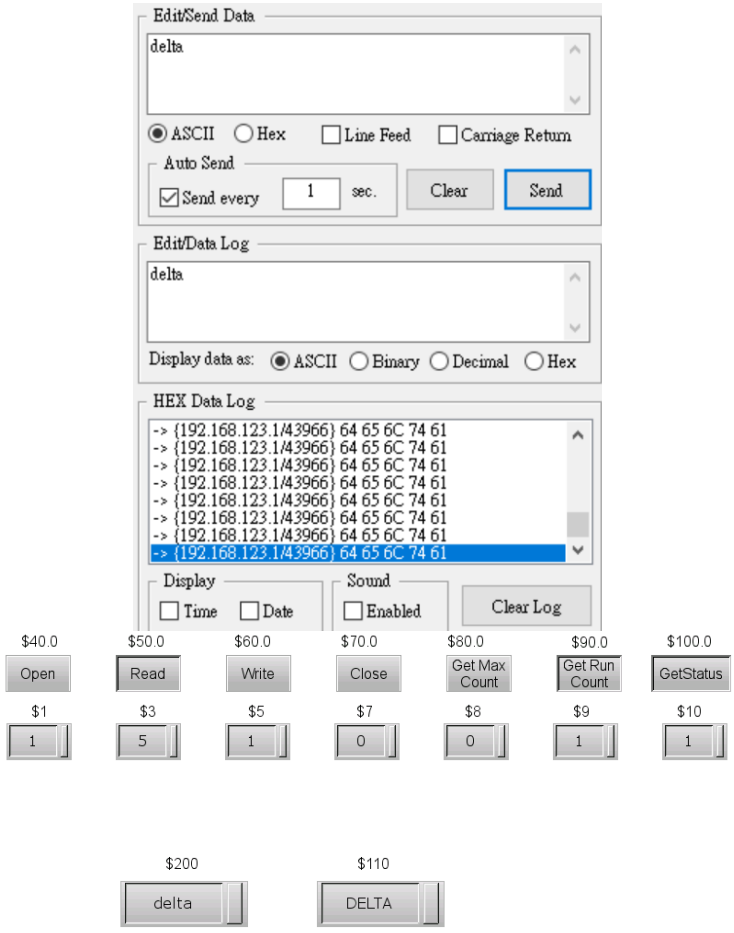
TCP 自由通訊	

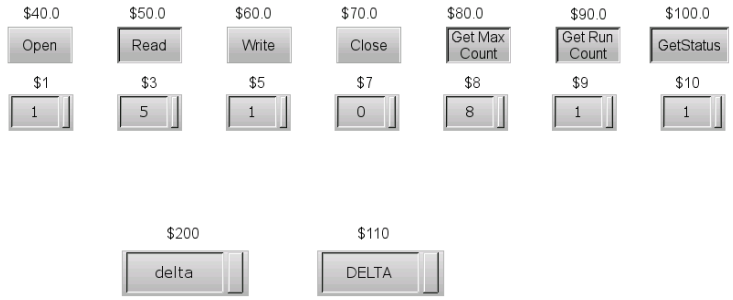
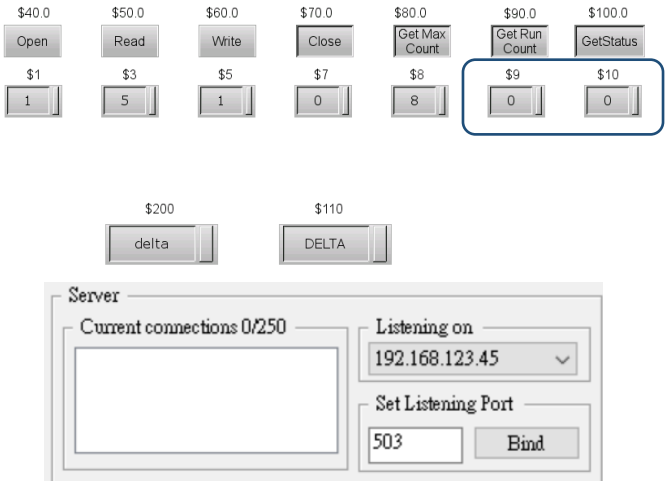
建立 Lua 自由通訊指令	■ 當觸發$40.0 時，開啟網路，並將回傳值寫入記憶體位址$1，當建立第 1 個連線後，關閉$40.0。
	`if (mem.inter.ReadBit(40,0)==1) then`
	`  ip = "192.168.123.45"`
	`  port = 503`
	`  socket = tcp.Open(ip, port)`
	`  mem.inter.Write(1,Socket)`
	`  if socket==1 then`
	`  mem.inter.WriteBit(40,0,0)`
	`  end`
	`end`
	■ 當觸發$50.0 時，以長度 5 bytes，通訊時間 3000 ms 進行讀取，讀取資料為 buffer，將回傳值寫入記憶體位址$3，並將資料 buffer 寫入$200。
	`if (mem.inter.ReadBit(50,0)==1) then`
	`  len = 5`
	`  timeout = 3000`
	`  bytes_read, buffer = tcp.Read(socket, len, timeout)`
	`  mem.inter.Write(3,bytes_read)`
	`  mem.inter.WriteAscii(200,buffer,string.len(buffer))`
	`end`
	■ 當觸發$60.0 時，以長度 10 bytes 讀取$110 記憶體位址，讀取到的資料為字串 buffer，並將此讀取到的資料寫入第 1 個(socket)連線，並將回傳值寫入$5。
	`if (mem.inter.ReadBit(60,0)==1) then`
	`  buffer = mem.inter.ReadAscii(110,10)`
	`  ret = tcp.Write(1, buffer, string.len(buffer), 1000)`
	`  mem.inter.Write(5, ret,string.len(ret))`
	`end`
	■ 當觸發$70.0 時，關閉第 1 個(socket)通訊，將回傳值寫入$7。
	`if (mem.inter.ReadBit(70,0)==1) then`
	`  socket = 1`
	`  ret = tcp.Close(socket)`
	`  mem.inter.Write(7,ret)`
	`end`
	■ 當觸發$80.0 時，取得人機最大支援連線數之資訊。
	`if (mem.inter.ReadBit(80,0)==1) then`
	`  count = tcp.GetMaxCount()`
	`  mem.inter.Write(8,count)`
	`end`
	■ 當觸發$90.0 時，確認指定之 socket 通訊是否使用中，並將回傳值寫入$9。
	`if (mem.inter.ReadBit(90,0)==1) then`
	`  count = tcp.RunCount(socket)`
	`  mem.inter.Write(9,count)`
	`end`
	■ 當觸發$100.0 時，確認指定之 socket 的連線狀態。
	`if (mem.inter.ReadBit(100,0)==1) then`
	`  status = tcp.GetStatus(socket)`
	`  mem.inter.Write(10,status)`
	`end`

<table>
<tr><th colspan="2">TCP 自由通訊</th></tr>
<tr><td>建立交替型按鈕、數值輸入、文數字輸入元件</td><td>■ 建立交替型按鈕，寫入記憶體位址為$40.0、$50.0、$60.0、$70.0、$80.0、$90.0、$100.0。
■ 建立數值輸入元件，寫入記憶體位址為$1、$3、$5、$7、$8、$9、$10。
■ 建立文數字輸入元件，寫入記憶體位址為$200 和$110。</td></tr>
<tr><td>執行結果</td><td>■ 完成 Lua 程式的編寫和元件的建立後，下載專案至人機(兩台人機畫面一致)。
■ 開啟 TCP Test Tool，電腦當作 Server，設定監聽的通訊埠(port)之後按 Bind 來等待 Client 連線。
 

■ 觸發$40.0，建立連線，並將回傳值寫入$1，最後關閉$40.0，可於 TCP Test Tool 介面中看到連線訊息。
 

■ 對$100 寫入字串觸發$60.0，讀取$110 為字串 buffer，並寫入指定之 socket 的通訊中，並將回傳值寫入$5。</td></tr>
</table>

TCP 自由通訊	
執行結果	■ 觸發$90.0、$100.0，將回傳值分別寫入$9、$10，因為第 1 個(socket)正在寫入資料，所以$9 和$10 分別為 1，代表第 1 個(socket)通訊正在使用中。  ■ 由 TCP Test Tool 發送字串 delta 至緩衝區，人機通過觸發$50.0，讀取緩衝區資料，將資料寫入$100，並將回傳值寫入$3。 

TCP 自由通訊	
執行結果	■ 觸發$80.0，得到最多支援連線數為 8。  ■ 觸發$70.0，關閉第 1 個(socket)連線，$9 和$10 分別顯示 0，代表此連線確實被關閉，TCP Test Tool 介面中也可看到連線消失。 

4.15 UDP communication (UDP 自由通訊)

自由通訊意指當兩個裝置沒有共同的通訊協定時，我們可以透過自由通訊的方式做溝通，台達 HMI 提供 COM、TCP 以及 UDP 建立自由通訊連線及交握資料。以下將介紹 UDP 的自由通訊指令，以及如何透過此指令建立連線。UDP 的自由通訊指令語法包含：

指令	指令運算式	說明
UDP communication (UDP 自由通訊)	udp.Open	開啟 UDP 網路通訊 (UDP)
	udp.Read	讀取字元 (UDP)
	udp.Write	寫入字元 (UDP)
	udp.Close	關閉連線 (UDP)
	udp.GetMaxCount	取得最大連線數量 (UDP)
	udp.GeRunCount	取得正在運行的 socket 數量 (UDP)
	udp.GetStatus	確認 socket 的通訊狀況 (UDP)

以下將一一詳細介紹。

■ udp.Open：開啟網路通訊 (UDP)

指令名稱	udp.Open
指令運算式	Socket = **udp.Open**(ip, port, local_port)
參數定義	ip：字串；接收端的 ip；“192.168.0.1”，… port：整數，接收端的 port local_port：整數，發送端的 port
範例	Socket = udp.Open(“192.168.123.134”, 502, 602)
範例說明	開啟 UDP 網路通訊；ip 為 192.168.123.134；port 通訊埠為 602；接收端通訊埠為 502。
回傳值	Socket：成功時回傳 socket 編號；失敗回傳負值

回傳值	說明
> 0	成功
-1	不合法參數
-101	開啟 socket 失敗
-145	已開啟太多個 socket

■ udp.Read：讀取字元 (UDP)

指令名稱	udp.Read
指令運算式	bytes_read, buffer = **udp.Read**(socket, len, timeout)
參數定義	socket：整數；1 ~ 8 len：整數；字串長度 timeout：整數；延遲時間 (單位：ms)
範例	bytes_read, buffer = udp.Read(1, 15, 1000)
範例說明	透過選擇接口 1，以所設定之通訊延遲時間 1000 ms 以及資料長度 15 bytes 來讀取字元。
回傳值	bytes_read：成功時回傳讀取資料的長度(bytes)；失敗時回傳負數 回傳值 / 說明： > 1 / 成功 -1 / 不合法參數 -100 / Socket 未開啟 -102 / 讀取失敗 buffer：讀回來的字串

■ udp.Write：寫入字元 (UDP)

指令名稱	udp.Write
指令運算式	ret = **udp.Write**(socket, buffer, len, timeout)
參數定義	socket：整數；1 ~ 8 buffer：字串 len：整數；字串長度 timeout：整數；延遲時間(ms)
範例	ret = udp.Write(1, “DELTA”, 5, 1000)
範例說明	以長度 5 bytes 寫入字串 DELTA 至通訊埠 1。
回傳值	ret：成功時回傳 1；失敗時回傳負值 回傳值 / 說明： -1 / 不合法參數 -100 / Socket 未開啟 -103 / 寫入失敗

■ udp.Close：關閉連線 (UDP)

指令名稱	udp.Close
指令運算式	ret = **udp.Close**(socket)
參數定義	socket：整數 1 ~ 8
範例	ret = udp.Close(1)
範例說明	關閉 UDP 自由通訊 socket1 連線。
回傳值	ret：成功時回傳 1；失敗時回傳 0 或負值 回傳值 / 說明： -1 / 不合法參數 -100 / Socket 未開啟

■ udp.GetMaxCount：取得最大連線數量 (UDP)

指令名稱	udp.GetMaxCount
指令運算式	count = **udp.GetMaxCount**()
參數定義	無參數
範例	count = udp.GetMaxCount()
範例說明	取得 UDP 自由通訊最大連線個數。
回傳值	count：系統最大 socket 個數

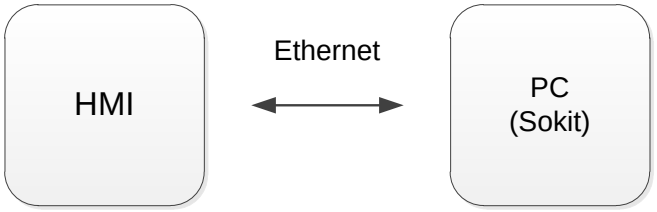
■ udp.GetRunCount：取得正在運行的 socket 數量 (UDP)

指令名稱	udp.GetRunCount
指令運算式	count = **udp.GetRunCount**()
參數定義	無參數
範例	count = udp.GetRunCount()
範例說明	取得 UDP 自由通訊正在運行的 socket 數。
回傳值	count：正在運行的 socket 數

■ udp.GetStatus：確認 socket 的通訊狀況 (UDP)

指令名稱	udp.GetStatus
指令運算式	status = **udp.GetStatus**(socket)
參數定義	socket：整數 1 ~ 8
範例	status = udp.GetStatus(1)
範例說明	確認 UDP 自由通訊 socket1 的通訊狀況。
回傳值	Status：開啟時回傳 1；關閉時回傳 0

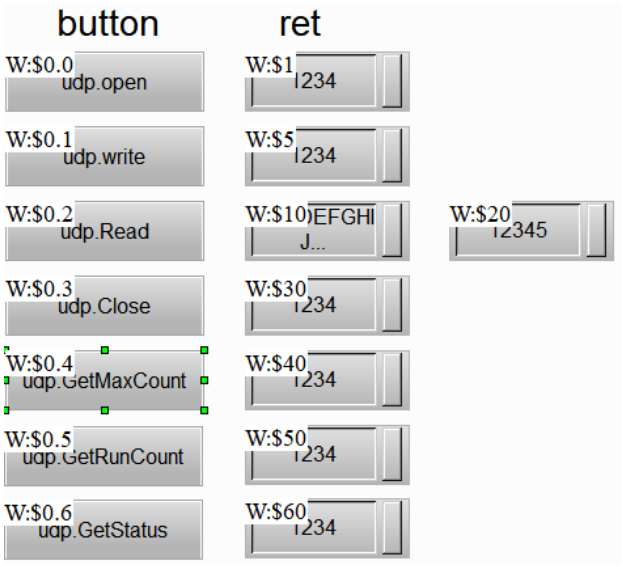
以下為自由通訊範例詳細說明。

UDP 自由通訊	
建立硬體連線	■ 接線概念圖。 ■ 以人機作為 Client，電腦當作 Server 接收 Client 資料；或以電腦作為 Client，人機當作 Server 接收 Client 資料。 

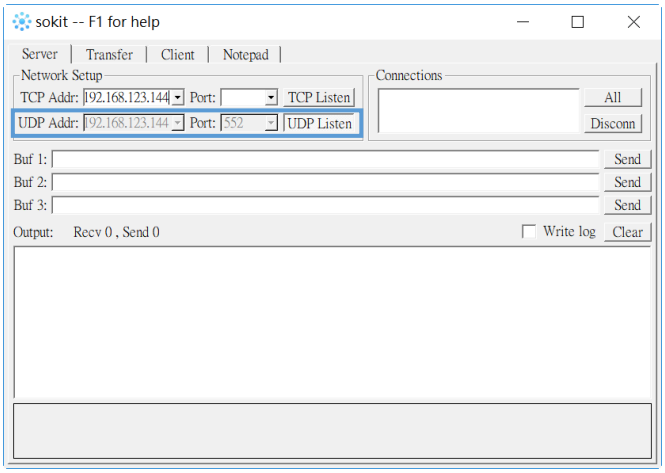
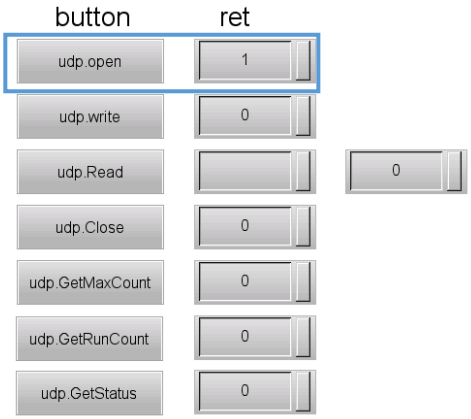
建立 Lua 自由通訊指令	指令如下：
	`while true do`
	`  if mem.inter.ReadBit(0,0)==1 then`
	`  ip = "192.168.123.144"`
	`  port = 552`
	`  local_port = 602`
	`  Socket = udp.Open(ip, port, local_port)`
	`  mem.inter.Write(1, Socket)`
	`  mem.inter.WriteBit(0,0,0)`
	`  end`
	
	`  if mem.inter.ReadBit(0,1)==1 then`
	`  socket = 1`
	`  buffer = "DELTA"`
	`  len = 5`
	`  timeout = 1000`
	`  ret = udp.Write(socket, buffer, len, timeout)`
	`  mem.inter.Write(5, ret)`
	`  mem.inter.WriteBit(0,1,0)`
	`  end`
	
	`  if mem.inter.ReadBit(0,2)==1 then`
	`  socket = 1`
	`  len = 15`
	`  timeout = 1000`
	`  bytes_read, buffer = udp.Read(socket, len, timeout)`
	`  mem.inter.WriteAscii(10, buffer,string.len(buffer))`
	`  mem.inter.Write(20,bytes_read)`
	`  mem.inter.WriteBit(0,2,0)`
	`  end`
	
	`  if mem.inter.ReadBit(0,3)==1 then`
	`  socket = 1`
	`  ret = udp.Close(socket)`
	`  mem.inter.Write(30,ret)`
	`  mem.inter.WriteBit(0,3,0)`
	`  end`
	
	`  if mem.inter.ReadBit(0,4)==1 then`
	`  count = udp.GetMaxCount()`
	`  mem.inter.Write(40,count)`
	`  mem.inter.WriteBit(0,4,0)`
	`  end`

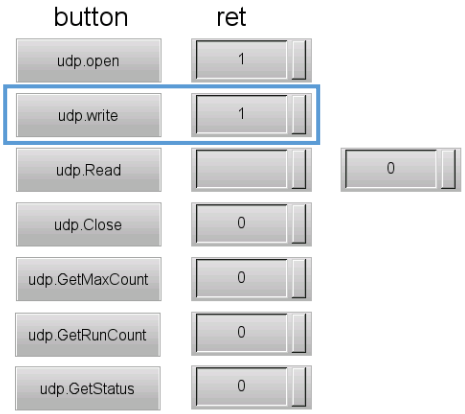
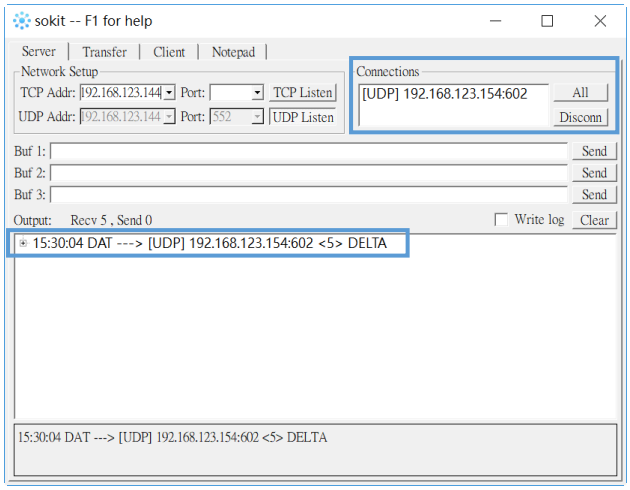
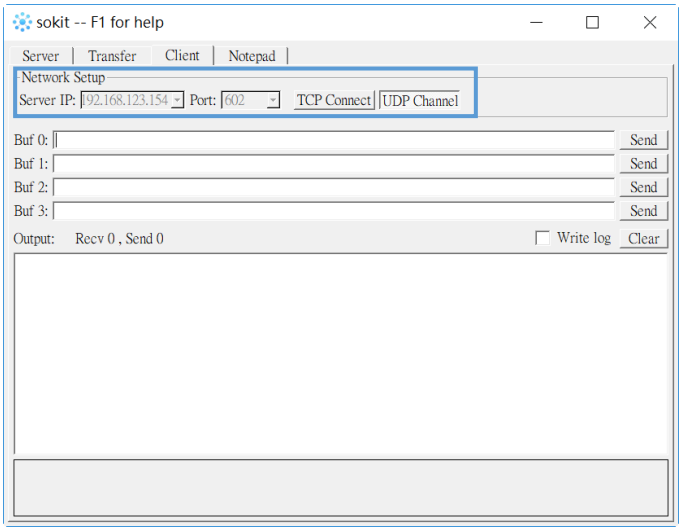
UDP 自由通訊	

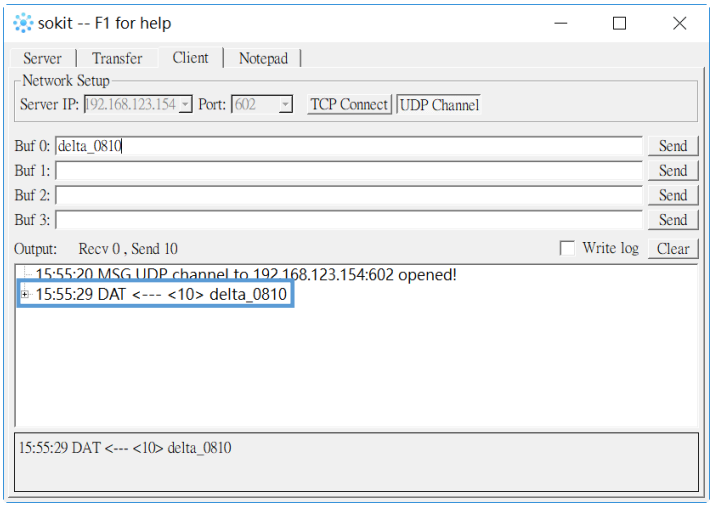
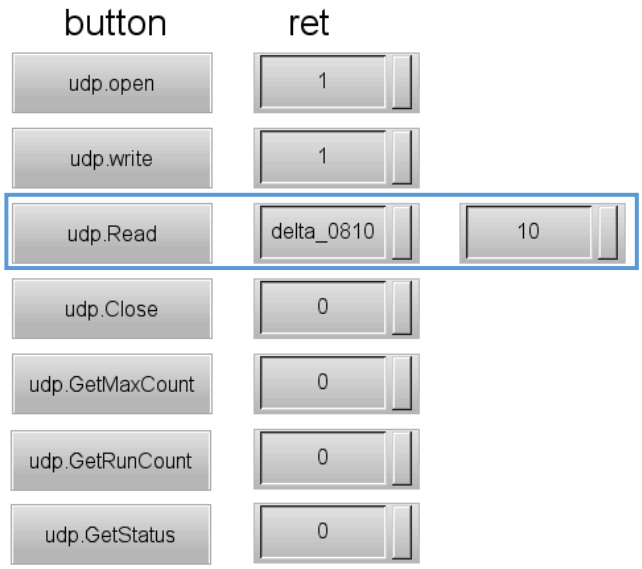
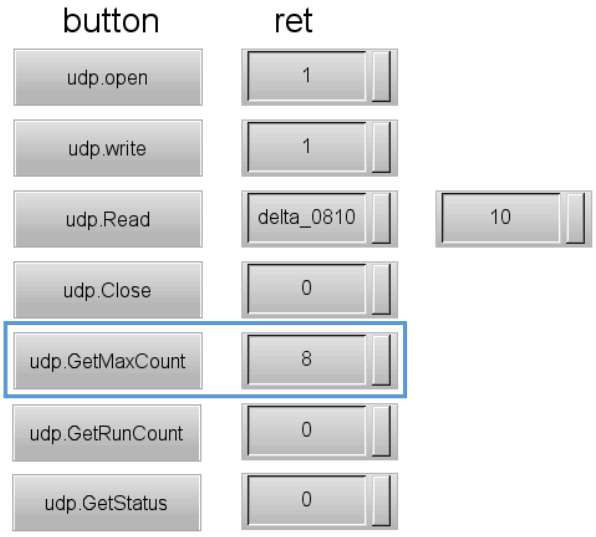
建立 Lua 自由通訊指令	if mem.inter.ReadBit(0,5)==1 then
	count = udp.GetRunCount()
	mem.inter.Write(50,count)
	mem.inter.WriteBit(0,5,0)
	end
	
	if mem.inter.ReadBit(0,6)==1 then
	status = udp.GetStatus(1)
	mem.inter.Write(60,count)
	mem.inter.WriteBit(0,6,0)
	end
	end
	■ 當觸發$0.0 時，建立網路設定，人機 Port 為 602，目的地 IP 為 192.168.123.144，Port 為 552 並將回傳值寫入記憶體位址$1，最後關閉$0.0。
	if mem.inter.ReadBit(0,0)==1 then
	ip = "192.168.123.144"
	port = 552
	local_port = 602
	Socket = udp.Open(ip, port, local_port)
	mem.inter.Write(1, Socket)
	mem.inter.WriteBit(0,0,0)
	end
	■ 當觸發$0.1 時，以長度 5 bytes 將字串”DELTA”寫入第 1 個(socket)連線，並將回傳值寫入$5，最後關閉$0.1。
	if mem.inter.ReadBit(0,1)==1 then
	socket = 1
	buffer = "DELTA"
	len =5
	timeout = 1000
	ret = udp.Write(socket, buffer, len, timeout)
	mem.inter.Write(5, ret)
	mem.inter.WriteBit(0,1,0)
	end
	■ 當觸發$0.2 時，以長度 15 bytes，通訊時間 1000 ms 進行讀取，讀取資料為 buffer，並將資料 buffer 寫入$10，將回傳值寫入記憶體位址$20，最後關閉$0.2。
	if mem.inter.ReadBit(0,2)==1 then
	socket = 1
	len = 15
	timeout = 1000
	bytes_read, buffer = udp.Read(socket, len, timeout)
	mem.inter.WriteAscii(10, buffer,string.len(buffer))
	mem.inter.Write(20,bytes_read)
	mem.inter.WriteBit(0,2,0)
	end
	■ 當觸發$0.3 時，關閉第 1 個(socket)通訊，將回傳值寫入$30，最後關閉$0.3。
	if mem.inter.ReadBit(0,3)==1 then
	socket = 1
	ret = udp.Close(socket)
	mem.inter.Write(30,ret)
	mem.inter.WriteBit(0,3,0)
	end

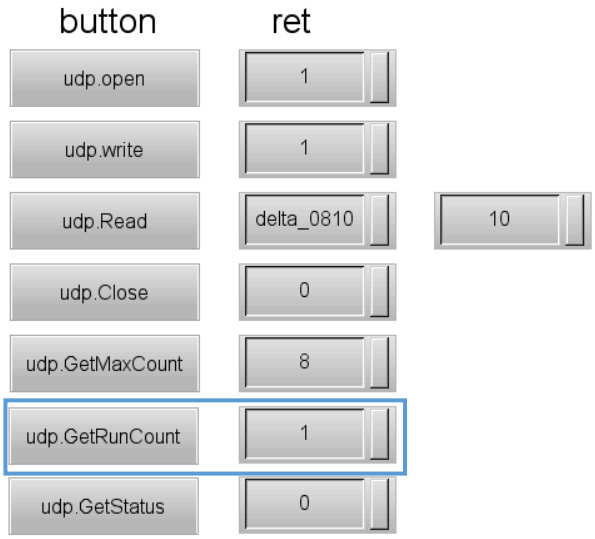
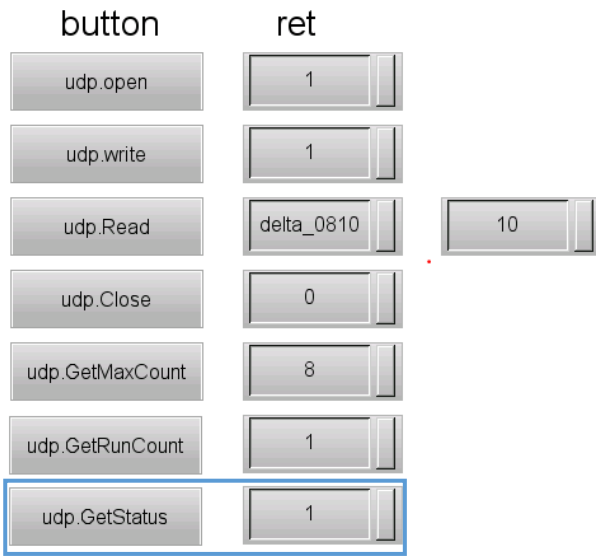
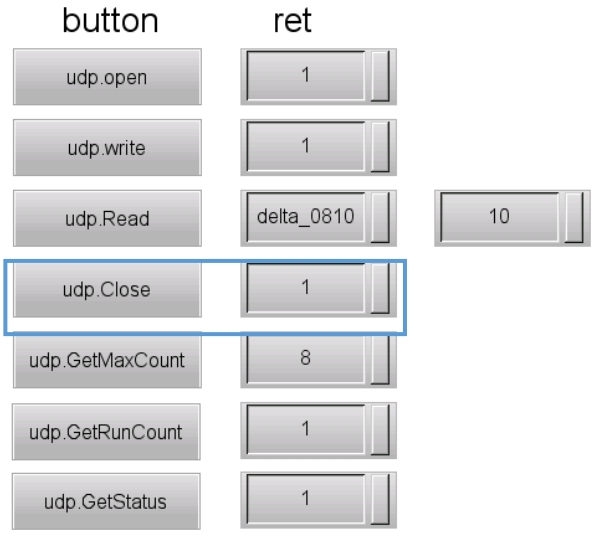
<table>
<tr><th colspan="2">UDP 自由通訊</th></tr>
<tr><td>建立 Lua 自由通訊指令</td><td>
■ 當觸發$0.4 時，取得人機最大支援連線數之資訊，並將數值寫入$40，最後關閉$0.4
<pre>
if mem.inter.ReadBit(0,4)==1 then
    count = udp.GetMaxCount()
     mem.inter.Write(40,count)
    mem.inter.WriteBit(0,4,0)
end
</pre>
■ 當觸發$0.5 時，確認當前運行的接口個數，並將數值寫入$50，最後關閉$0.5。
<pre>
if mem.inter.ReadBit(0,5)==1 then
    count = udp.GetRunCount()
      mem.inter.Write(50,count)
    mem.inter.WriteBit(0,5,0)
end
</pre>
■ 當觸發$0.6 時，確認指定之 socket 的連線狀態，並將數值寫入$60，最後關閉$0.6。
<pre>
if mem.inter.ReadBit(0,6)==1 then
    status = udp.GetStatus(1)
      mem.inter.Write(60,count)
    mem.inter.WriteBit(0,6,0)
end
</pre>
</td></tr>
<tr><td>建立交替型按鈕、數值輸入、文數字輸入元件</td><td>
■ 建立交替型按鈕，寫入記憶體位址為$0.0、$0.1、$0.2、$0.3、$0.4、$0.5、$0.6。
■ 建立數值輸入元件，寫入記憶體位址為$1、$5、$20、$30、$40、$50、$60。
■ 建立文數字輸入元件，寫入記憶體位址為$10。


</td></tr>
</table>

UDP 自由通訊	
執行結果	■ 完成 Lua 程式的編寫和元件的建立後，下載專案至人機。 ■ 使用第三方軟體 sokit，電腦當作 Server，設定監聽的通訊埠(port)之後按 **UDP Listen** 來等待 Client 連線。  ■ 觸發$0.0，建立連線設定，並將回傳值寫入$1，最後關閉$0.1。 

UDP 自由通訊	
執行結果	■ 觸發$0.1，寫入字串 DELTA 至指定之 socket 的通訊中，並將回傳值寫入$5，可以於第三方軟體 sokit 看到寫入內容。   ■ 將 PC 設定為 Client，設定 Server IP 和 Port 並建立連線。 

UDP 自由通訊	
執行結果	■ 透過 sokit 寫入字串”delta_0810”至緩衝區，並於人機觸發$0.2，讀取緩衝區資料，將資料寫入$10，並將回傳值寫入$20。   ■ 觸發$0.4，得到最多支援連線數為 8。 

UDP 自由通訊	
執行結果	■ 觸發$0.5，取得正在運行的 Socket 個數，並將結果寫至$10。  ■ 觸發$0.6，取得指定的 Socket 狀態，1 代表開啟中。  ■ 觸發$0.3，將 Socket 關閉。 

4.16 Text encoding (編碼格式變更)

此指令可供使用者將 GBK 格式轉換為 UTF-8 格式，基本語法包含：

指令	指令運算式	說明
Text encoding (編碼格式變更)	text.GbkToUtf8	將 GBK 格式轉換為 UTF-8

以下將詳細介紹。

■ text.GbkToUtf8：將 GBK 格式轉換為 UTF-8

指令名稱	text.GbkToUtf8
指令運算式	utf8_len, utf8_string = **text.GbkToUtf8**(ascii_string, ascii_len)
參數定義	ascii_string：GBK 字串 ascii_len：整數值；字串長度 (Bytes)
範例	utf8_len, utf8_string = text.GbkToUtf8("简体", string.len("简体"))
範例說明	以 string.len("简体")為字串長度，將"简体"字串轉換為 UTF-8 格式之字串。
回傳值	utf8_len：整數值；轉換後的 UTF-8 byte 長度；小於等於 0：異常 utf8_string：字串；轉換後的 UTF-8 字串；nil：異常

範例 (text.GbkToUtf8)

建立 Lua 程序

■ 建立 Lua 轉換數值格式。

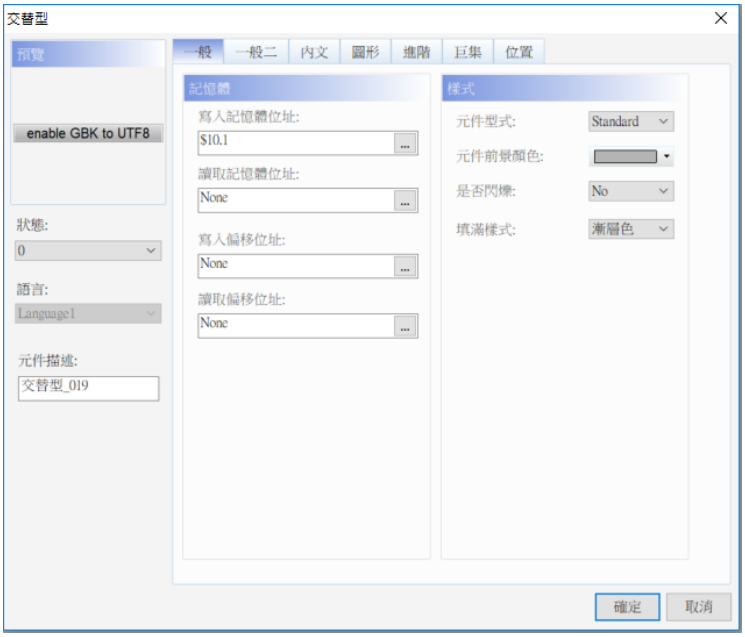
■ 當$10.1 被觸發時，以長度為 4、讀取位址為$100 的字串為 buffer，將字串 buffer 轉換為 UTF-8 編碼格式，最後將結果字串寫入$600。

```
while true do
    if (mem.inter.ReadBit(10, 1) == 1) then
        buffer = mem.inter.ReadAscii(100, 4)
        str_bytes, utf_str = text.GbkToUtf8(buffer, 4)
        mem.inter.WriteAscii(600, utf_str, str_bytes)
    end
end
```

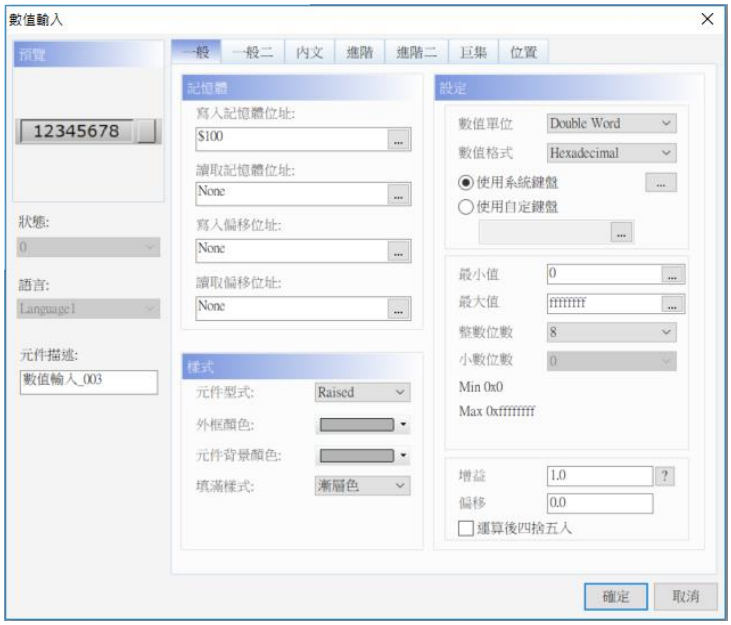
範例 (text.GbkToUtf8)

建立交替型按鈕、數值輸入、多語輸入元件

■ 建立交替型按鈕元件，其寫入記憶體位址為$10.1。



■ 建立數值輸入元件，其寫入記憶體位址為$100，數值格式設為Hexadecimal。

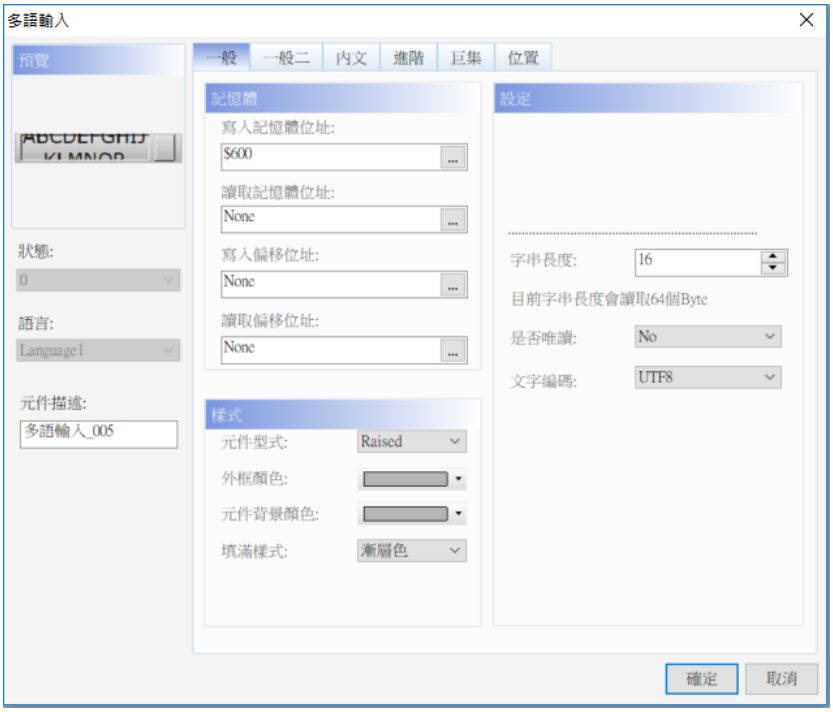


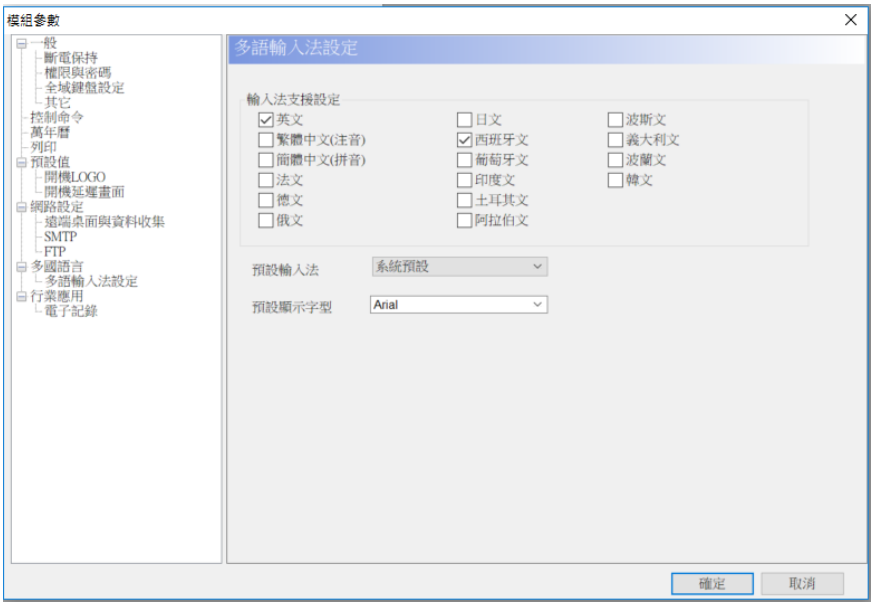
範例 (text.GbkToUtf8)

建立交替型按鈕、數值輸入、多語輸入元件

■ 建立多語輸入元件，其寫入記憶體位址為$600，文字編碼設為 UTF8，字串長度為 16。

註：使用多語輸入元件需搭配一種以上多語輸入法。

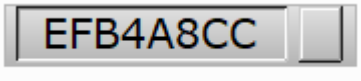
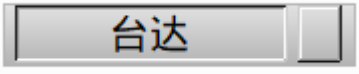




執行結果

■ 完成 Lua 程式的編寫和元件的建立後，下載專案至人機。

■ 對$100 輸入 EFB4A8CC，觸發$10.1，$600 顯示對應資料。

GBK	UTF-8
	

4.17 Utility (CRC 運算)

此指令可供使用者計算 CRC 數值，循環冗餘校驗 (Cyclic redundancy check)，通稱「CRC」，用於驗證資料傳輸過程中是否產生錯誤，關於 CRC 相關資訊可以透過網路公開資料獲取，這裡不多加說明，基本語法包含：

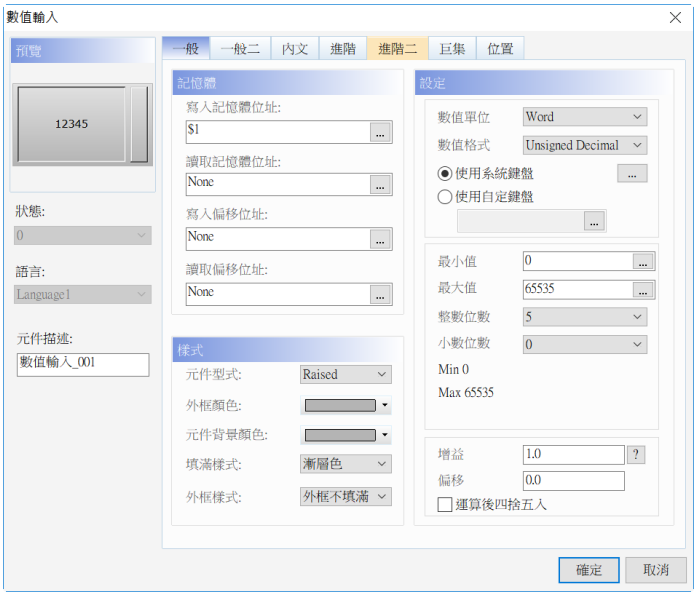
指令	指令運算式	說明
Utility (CRC 運算)	util.Crc16Modbus	計算 crc 數值

以下將詳細介紹。

■ util.Crc16Modbus：計算 crc 數值

指令名稱	util.Crc16Modbus
指令運算式	crc16 = **util.Crc16Modbus**(str, strLen, initValue)
參數定義	str：ASCII 字串 strLen：整數值；字串長度 (Bytes) initValue：初始值；預設為 0xFFFF
範例	crc16 = util.Crc16Modbus("abc123", 6, 0xFFFF)
範例說明	建立初始值為 16 進位的 0xFFFF，以 6 個 Bytes 長度將字串“abc123”與初始值進行運算後，得到其 crc 數值，詳細運算內容請參考網路資訊。
回傳值	crc16：整數值；crc16 結果

範例 (Utility)	
利用 Lua 轉換格式	■ 建立 Lua 轉換數值格式，並將數值寫入記憶體位址$1。 while true do str = "abc123" strLen = string.len(str) initValue = 0xFFFF crc16 = util.Crc16Modbus(str, strLen, initValue) mem.inter.Write(1,crc16) end

<table>
<tr><th colspan="2">範例 (Utility)</th></tr>
<tr><td>建立數值輸入元件</td><td>■ 建立數值輸入元件，寫入記憶體位址為$1。
 
</td></tr>
<tr><td>執行結果</td><td>■ 完成 Lua 程式的編寫和元件的建立後，下載專案至人機。
■ 數值輸入元件顯示字串”abc123”與初始值計算出的 CRC 結果。
<table>
<tr><th>欲計算 CRC 的字串</th><th>CRC 數值</th></tr>
<tr><td>abc123</td><td> 
</td></tr>
</table></td></tr>
</table>

4.18 Convert (浮點數轉換)

此指令可供使用者進行數值格式的轉換，此指令包含：

指令	指令運算式	說明
Convert (浮點數轉換)	convert.IntToFloat	整數格式轉換成浮點數格式
	convert.ToNum	字串轉換為 64 位元浮點數

以下將一一詳細介紹。

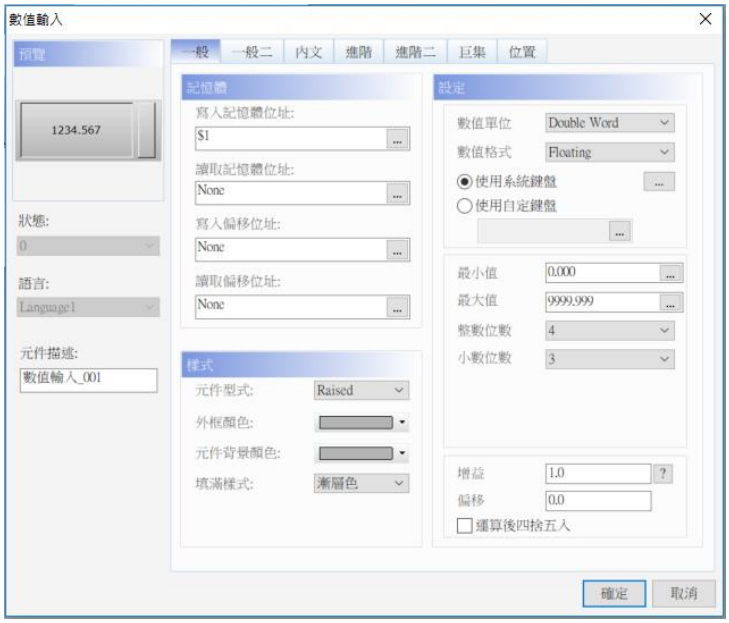
■ convert.IntToFloat：整數格式轉換成浮點數格式

指令名稱	convert.IntToFloat
指令運算式	fVal, ret = **convert.IntToFloat**(iVal)
參數定義	iVal：32 位元整數
範例	fVal, ret = convert.IntToFloat(0x42F6E666)
範例說明	將 0x42F6E666 以整數格式轉換成浮點數格式。
回傳值	ret：成功時回傳 1；失敗時回傳 0 fVal：轉換後的單精準浮點數

■ convert.ToNum：字串轉換為 64 位元浮點數

指令名稱	convert.ToNum
指令運算式	dVal, ret = **convert.ToNum**(str)
參數定義	str：字串，例如："123"
範例	dVal, ret = convert.ToNum("123")
範例說明	將字串"123"轉換為 64 位元浮點數格式，dVal = 123。
回傳值	ret：成功時回傳 1；失敗時回傳 0 dVal：轉換後的 64 位元浮點數

<table>
<tr><th colspan="2">convert</th></tr>
<tr><td>利用 Lua
轉換格式</td><td>■ 建立 Lua 轉換數值格式，並將數值寫入記憶體位址$1。

while true do
fVal = convert.IntToFloat(0x42F6E666)
mem.inter.WriteFloat(1,fVal)
end</td></tr>
<tr><td>建立數值輸
入元件</td><td>■ 建立數值輸入元件，寫入記憶體位址為$1，數值格式設為 Floating。
 
</td></tr>
<tr><td>執行結果</td><td>■ 完成 Lua 程式的編寫和元件的建立後，下載專案至人機。
■ 數值輸入元件顯示 0x42F6E666 轉換為浮點式的結果。
<table><tr><th>Integer</th><th>Float</th></tr><tr><td>0x42F6E666</td><td> 
</td></tr></table></td></tr>
</table>

4.19 Account (權限密碼設定)

此指令可供使用者對權限和密碼做管理的動作，指令包括：

指令	指令運算式	說明
Account (權限密碼設定)	account.Add	新增權限帳號
	account.Delete	刪除權限帳號
	account.ChangeName	更改權限帳號名稱
	account.ChangePassword	更改權限密碼
	account.ChangeLevel	更改權限帳號等級
	account.GetPassword	取得使用者的密碼
	account.GetLevel	取得權限等級
	account.GetCurrentLogin	取得當前登入帳號
	account.IsExist	確認帳號是否存在
	account.Login	登入權限
	account.ResetLockStatus	解除已鎖定的帳戶
	account.ChangeUserExpiredDays	更改帳號期限
	account.ChangePwdExpiredDays	更改密碼期限
	account.GetStatus	取得帳戶狀態
	account.GetLockedList	取得已鎖定的帳戶之清單

以下將一一詳細介紹。

■ account.Add：新增權限帳號

指令名稱	account.Add
指令運算式	ret = **account.Add**(name, password, level)
參數定義	name：帳號名稱 password：帳號密碼 level：整數；帳號權限
範例	ret = account.Add("DELTA", "1234", 5)
範例說明	新增權限 5，使用者名稱為 DELTA，密碼為 1234。
回傳值	ret：成功時回傳 1；失敗時回傳 0

■ account.Delete：刪除權限帳號

指令名稱	account.Delete
指令運算式	ret = **account.Delete**(name)
參數定義	name：帳號名稱
範例	ret = account.Delete(“posheng”)
範例說明	刪除使用者名稱為 posheng 之權限。
回傳值	ret：成功時回傳 1；失敗時回傳 0

■ account.ChangeName：更改權限帳號名稱

指令名稱	account.ChangeName
指令運算式	ret = **account.ChangeName**(srcName, newName)
參數定義	srcName：字串；要被更改的帳號名稱 newName：字串；新的帳號名稱
範例	ret = account.ChangeName("DELTA", “posheng”)
範例說明	變更使用者名稱“DELTA”為“posheng”。
回傳值	ret：成功時回傳 1；失敗時回傳 0

■ account.ChangePassword：更改權限密碼

指令名稱	account.ChangePassword
指令運算式	ret = **account.ChangePassword**(Name, newPassword)
參數定義	name：字串；要被更改密碼的帳號名稱 newPassword：字串；新的密碼
範例	ret = account.ChangePassword("DELTA", "0101")
範例說明	變更使用者 DELTA 的密碼為 0101。
回傳值	ret：成功時回傳 1；失敗時回傳 0

■ account.ChangeLevel：更改權限帳號等級

指令名稱	account.ChangeLevel
指令運算式	ret = **account.ChangeLevel**(name, newLevel)
參數定義	name：字串；要被更改權限的帳號名稱 newLevel：整數；新的權限
範例	account.ChangeLevel("DELTA", 1)
範例說明	改變使用者 DELTA 的權限等級為 1。
回傳值	ret：成功時回傳 1；失敗時回傳 0

■ account.GetPassword：取得使用者的密碼

指令名稱	account.GetPassword
指令運算式	ret, password = **account.GetPassword**(name)
參數定義	name：欲取得密碼的帳號名稱
範例	ret, password = account.GetPassword(“DELTA”)
範例說明	得到使用者 DELTA 的密碼為 password。
回傳值	ret：成功時回傳 1；失敗時回傳 0 password：字串；取得之密碼

■ account.GetLevel：取得權限等級

指令名稱	account.GetLevel
指令運算式	ret, level = **account.GetLevel**(name)
參數定義	name：字串；欲取得權限的帳號名稱
範例	ret, level = account.GetLevel("DELTA")
範例說明	得到使用者 DELTA 的權限等級為 level。
回傳值	ret：成功時回傳 1；失敗時回傳 0 level：整數；取得之權限等級

■ account.GetCurrentLogin：取得當前登入帳號

指令名稱	account.GetCurrentLogin
指令運算式	ret, name, level = **account.GetCurrentLogin**()
參數定義	無參數
範例	ret, name, level = account.GetCurrentLogin()
範例說明	取得當前登入帳號。
回傳值	ret：成功時回傳 1；失敗時回傳 0 name：字串；取得之帳號名稱 level：整數；取得之權限等級

■ account.IsExist：確認帳號是否存在

指令名稱	account.IsExist
指令運算式	ret = **account.IsExist**(name)
參數定義	name：字串；帳號名稱
範例	ret = account.IsExist("DELTA")
範例說明	確認使用者 DELTA 的權限是否存在。
回傳值	ret：存在時回傳 1；不存在時回傳 0

■ account.Login：登入權限

指令名稱	account.Login
指令運算式	ret = **account.Login**(name, password)
參數定義	name：字串；登入帳號 password：字串；登入密碼
範例	ret = account.Login("DELTA", "0101")
範例說明	以使用者名稱為 DELTA，密碼為 0101 登入使用者權限。
回傳值	ret：成功時回傳 1；失敗時回傳 0

■ account.ResetLockStatus：解除已鎖定的帳戶

指令名稱	account.ResetLockStatus
指令運算式	ret = **account.ResetLockStatus**(name)
參數定義	name：字串；帳號名稱
範例	ret = account.ResetLockStatus("user01")
範例說明	解除已鎖定的帳號 user01。
回傳值	ret：成功時回傳 1；失敗時回傳 0

■ account.ChangeUserExpiredDays：更改帳號期限

指令名稱	account.ChangeUserExpiredDays
指令運算式	ret = **account.ChangeUserExpiredDays**(name, expiredDays)
參數定義	name：字串；要被更改帳號期限的帳號名稱 expiredDays：整數；0 ~ 9999
範例	ret = account.ChangeUserExpiredDays("11", 2)
範例說明	更改帳號名稱”11”的帳號期限為 2 天。 註：如登入此帳號會顯示帳號已過期。
回傳值	ret：成功時回傳 1；失敗時回傳 0；參數設定錯誤回傳 -1

■ account.ChangePwdExpiredDays：更改密碼期限

指令名稱	account.ChangePwdExpiredDays
指令運算式	ret =**account.ChangePwdExpiredDays**(name, expiredDays)
參數定義	name：字串；要被更改密碼期限的帳號名稱 expiredDays：整數；0 ~ 9999
範例	ret = account.ChangePwdExpiredDays("33", 2)
範例說明	更改帳號名稱“33”的密碼期限為 2 天。 註：如登入此帳號會顯示密碼已過期。
回傳值	ret：成功時回傳 1；失敗時回傳 0；參數設定錯誤回傳 -1

■ account.GetStatus：取得帳戶狀態

指令名稱	account.GetStatus
指令運算式	ret = **account.GetStatus**(name)
參數定義	name：字串；欲取得帳戶狀態的權限名稱
範例	ret = account.GetStatus("33")
範例說明	取得帳號名稱“33”的帳戶狀態。
回傳值	ret：帳號鎖定時回傳 1；帳號未鎖定時回傳 0；參數設定錯誤回傳 -1

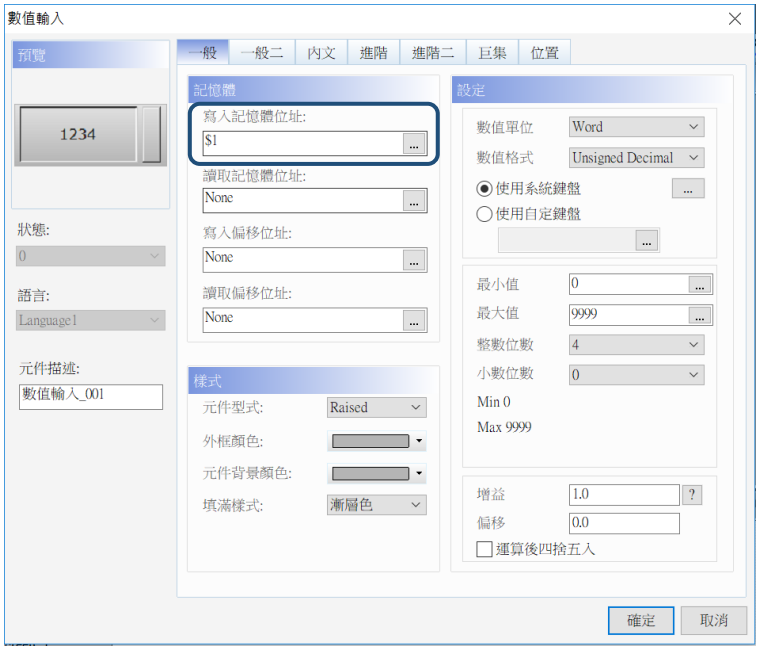
■ account.GetLockedList：取得已鎖定的帳戶之清單

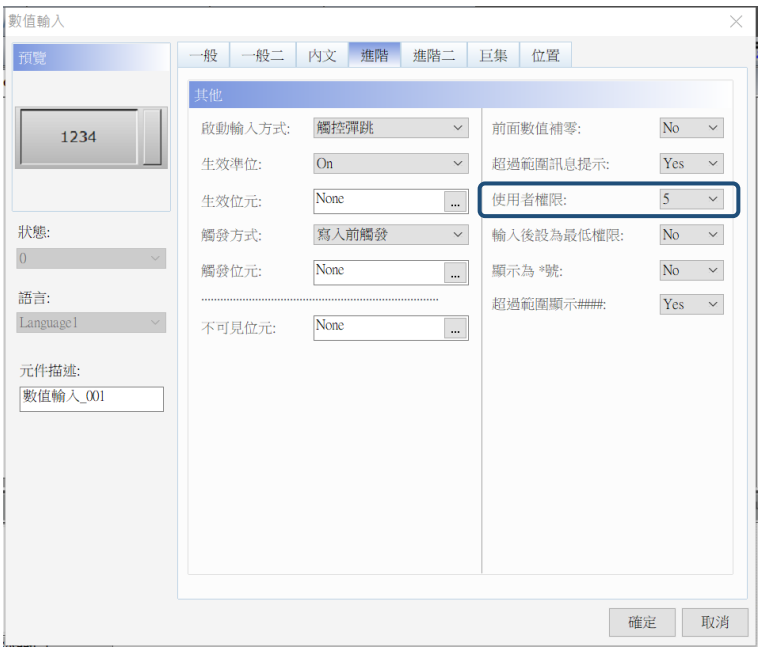
指令名稱	account.GetLockedList
指令運算式	ret, nameList = **account.GetLockedList**()
參數定義	無參數
範例	ret, nameList = account.GetLockedList()
範例說明	取得已鎖定的帳戶之清單。
回傳值	ret：成功時回傳 1；失敗時回傳 0 nameList：矩陣 table；成功：檔案名稱清單；失敗：nil

範例 (account)

建立數值輸入元件

■ 建立數值輸入元件，寫入記憶體位址設為$1，使用者權限設為 5。

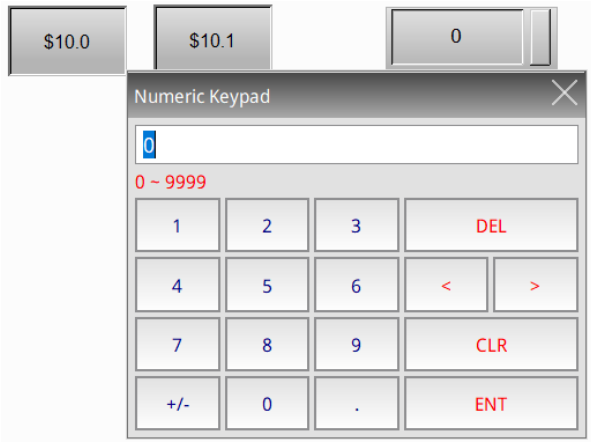




<table>
<tr><th colspan="2">範例 (account)</th></tr>
<tr><td>利用 Lua
建立權限</td><td>■ 利用 Lua 建立權限，當記憶體位址$10.0 被觸發時建立一個使用者權限，使用者名稱為 POSHENG，密碼為 DELTA，權限等級為 5。
if (mem.inter.ReadBit(10,0)==1) then
strName = "POSHENG"
strPassword = "DELTA"
intLevel = 5
ret = account.Add(strName, strPassword, intLevel)
end</td></tr>
<tr><td>Lua
登入權限</td><td>■ 輸入 Lua 指令如下圖。
if (mem.inter.ReadBit(10,1)==1) then
strName = "POSHENG"
strPassword = "DELTA"
ret = account.Login(strName, strPassword)
end</td></tr>
<tr><td>建立交替型
按鈕</td><td>■ 建立交替型按鈕，寫入記憶體位址為$10.0 和$10.1。
 
</td></tr>
<tr><td>執行結果</td><td>■ 完成 Lua 程式的編寫和元件的建立後，下載專案至人機。
 

■ 觸發$10.0，建立權限。
 

■ 按下數值輸入元件，此時可直接輸入使用者名稱和密碼進行使用。
 
</td></tr>
</table>

範例 (account)	
執行結果	■ 也可以透過觸發$10.1，登入使用者，便可直接使用數值輸入元件。 

4.20 Mail (信件功能)

此指令可供使用者透過人機呼叫 Mail 伺服器寄出信件或檔案，但使用者需要先完成 SMTP 的相關設定。其指令包括：

指令	指令運算式	說明
Mail (信件功能)	mail.Status	信件功能狀態
	mail.Send	傳送信件
	mail.SendFile	傳送信件 (含檔案)
	mail.SendAlarm	傳送信件 (含警報)
	mail.SendHistory	傳送信件 (含歷史資料)

以下將一一詳細介紹。

■ mail.Status：信件功能狀態

指令名稱	mail.Status
指令運算式	status = **mail.Status**()
參數定義	無參數
範例	status = mail.Status()
範例說明	取得當前信件功能狀態。
回傳值	回傳值 / 說明

回傳值	說明
1	郵件已成功派送
0	初始值；沒有郵件正在派送，或者派送任務剛剛開始
-100	主機連線失敗
-101	已斷線
-102	需要認證
-103	認證失敗
-999	未知錯誤

■ mail.Send：傳送信件

指令名稱	mail.Send
指令運算式	result, error = **mail.Send**(receiver, subject, content)
參數定義	receiver：字串；郵件收件者 subject：字串；郵件主旨。如不需要主旨，請設為 nil content：字串；郵件內容。如不需要內容，請設為 nil
範例	result, error = mail.Send("test@test.com", "test mail subject", "test mail content")
範例說明	傳送訊息到 test@test.com，主旨為"test mail subject"、內容為 "test mail content"。
回傳值	result：整數；0：尚未開始郵件派送；1：郵件派送已開始 error：整數

回傳值	說明
0	初始值；沒有郵件正在派送，或者派送任務剛剛開始
-1	不合法參數
-121	SMTP 未設定

■ mail.SendFile：傳送信件 (含檔案)

指令名稱	mail.SendFile
指令運算式	result, error = **mail.SendFile**(receiver, subject, content, disk_id, file_path, password)
參數定義	receiver：字串；郵件收件者 subject：字串；郵件主旨。如不需要主旨，請設為 nil content：字串；郵件內容。如不需要內容，請設為 nil disk_id：整數；0：HMI；2：USB；3：SD file_path：字串；相對於 disk_id 的檔案路徑 password：字串；檔案密碼。如不需要密碼，請設為 nil
範例	result, error = mail.SendFile("test@test.com", "test mail subject", "test mail content", 2, "test/file.txt", "1234")
範例說明	壓縮 USB 儲存裝置內的"test/file.txt"，並設密碼“1234”，然後以主旨 test mail subject，內容 test mail content 傳送信件到 test@test.com。
回傳值	result：整數；0：尚未開始郵件派送；1：郵件派送已開始 error：整數

回傳值	說明
0	無錯誤
-1	不合法參數
-106	指定磁碟未備妥
-107	無法打開指定檔案
-110	指定檔名路徑不存在
-121	SMTP 未設定
-122	壓縮檔案時發生錯誤

■ mail.SendAlarm：傳送信件 (含警報)

<table>
<tr><td>指令名稱</td><td>mail.SendAlarm</td></tr>
<tr><td>指令運算式</td><td>result, error = mail.SendAlarm(receiver, subject, content, password)</td></tr>
<tr><td>參數定義</td><td>receiver：字串；郵件收件者
subject：字串；郵件主旨。如不需要主旨，請設為 nil
content：字串；郵件內容。如不需要內容，請設為 nil
password：字串；檔案密碼。如不需要密碼，請設為 nil</td></tr>
<tr><td>範例</td><td>result, error = mail.SendAlarm("test@test.com", "test mail subject", "test mail content", "1234")</td></tr>
<tr><td>範例說明</td><td>壓縮警報 CSV 並設密碼“1234”，然後以主旨 test mail subject，內容 test mail content 傳送信件到 test@test.com。</td></tr>
<tr><td>回傳值</td><td>result：整數；0：尚未開始郵件派送；1：郵件派送已開始
error：整數
<table>
<tr><th>回傳值</th><th>說明</th></tr>
<tr><td>0</td><td>無錯誤</td></tr>
<tr><td>-1</td><td>不合法參數</td></tr>
<tr><td>-107</td><td>無法打開指定檔案</td></tr>
<tr><td>-121</td><td>SMTP 未設定</td></tr>
<tr><td>-122</td><td>壓縮檔案時發生錯誤</td></tr>
<tr><td>-126</td><td>無警報啟用</td></tr>
<tr><td>-127</td><td>輸出警報 CSV 失敗</td></tr>
</table></td></tr>
</table>

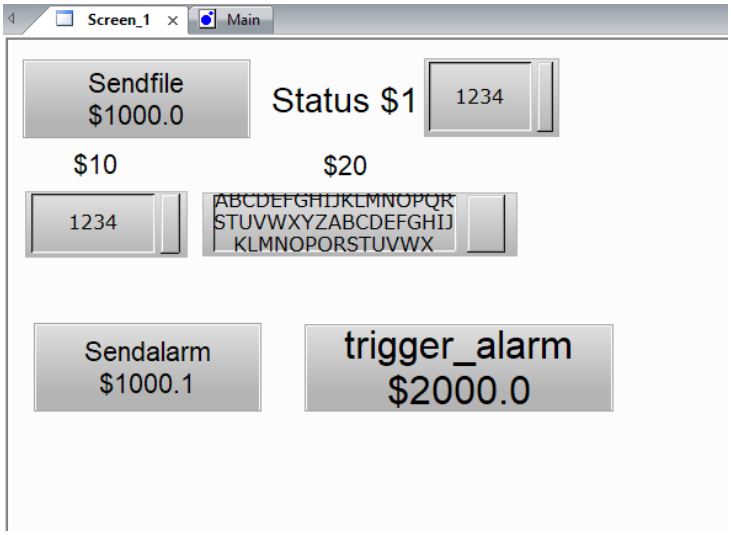
■ mail.SendHistory：傳送信件(含歷史資料)

指令名稱	mail.SendHistory
指令運算式	result, error = **mail.SendHistory**(bufferNo, dayRange, receiver, subject, content, password)
參數定義	bufferNo：整數；歷史緩衝區編號 dayRange：整數；歷史緩衝區 CSV 的天數範圍。此參數允許設定 0 ~ 7，且必須啟用分檔功能，若設定 0 代表所有天數內容 receiver：字串；郵件收件者 subject：字串；郵件主旨。如不需要主旨，請設為 nil content：字串；郵件內容。如不需要內容，請設為 nil password：字串；檔案密碼。如不需要密碼，請設為 nil
範例	result, error = mail.SendHistory(1, 0, "test@test.com", "test mail subject", "test mail content", "1234")
範例說明	壓縮歷史編號 1 的 CSV 並設密碼為”1234”，然後以主旨 test mail subject，內容 test mail content 傳送信件傳送到 test@test.com。
回傳值	result：整數；0：尚未開始郵件派送；1：郵件派送已開始 error：整數

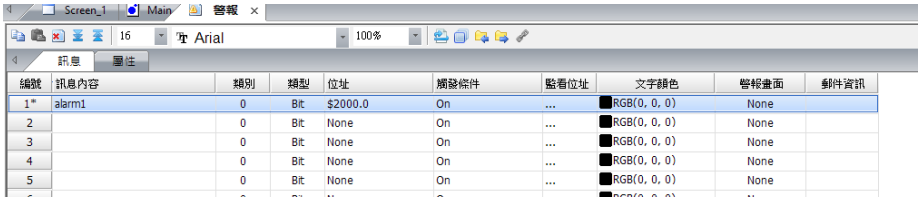
回傳值	說明
0	無錯誤
-1	不合法參數
-107	無法打開指定檔案
-121	SMTP 未設定
-128	無歷史緩衝區啟用
-129	不正確的歷史緩衝區編號
-130	不正確的天數範圍
-131	輸出歷史 CSV 失敗
-132	複製歷史 CSV 失敗
-133	無符合檔案

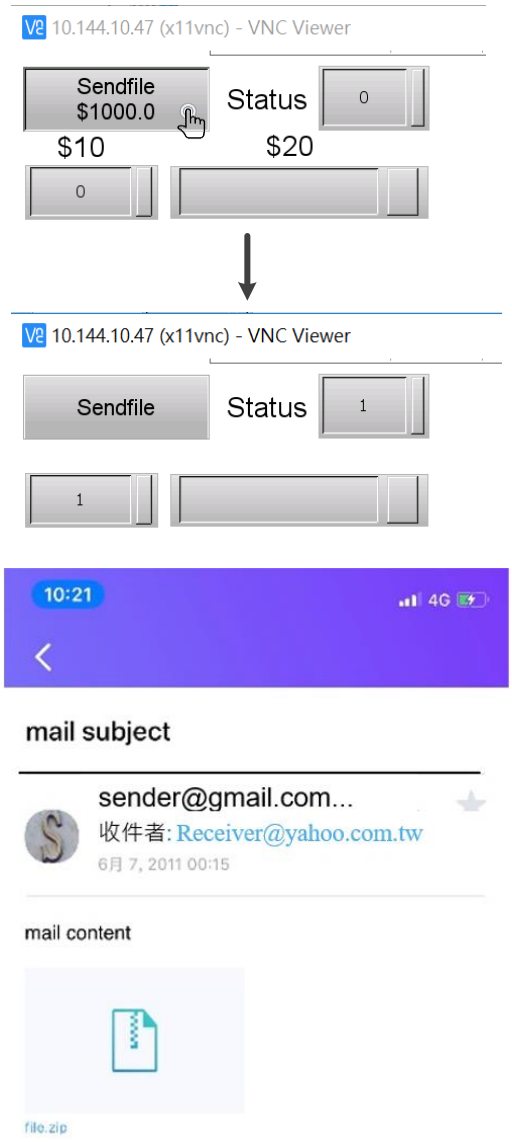
範例 (mail)	

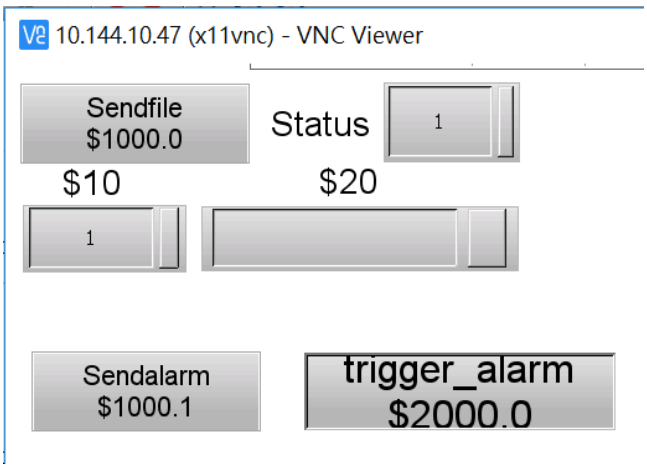
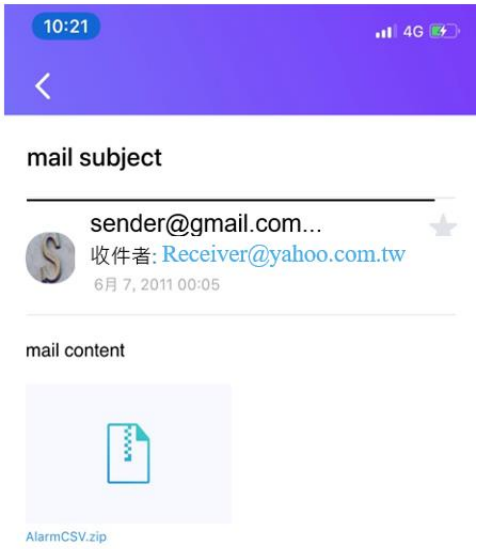
撰寫Lua程序	■ 於軟體左側專案樹 → [Main]，建立Lua指令(mail.SendFile)。
	if mem.inter.ReadBit(1000,0)==1 then
	disk_id = 2
	file_name = "posheng.txt"
	ret, fileHandle = file.Open(disk_id, file_name)
	sys.Sleep(1000)
	result, error = mail.SendFile("Receiver@yahoo.com.tw", "mail subject", "mail content", 2, "posheng.txt", "1234")
	mem.inter.Write(10,result)
	if result ~=1 then
	mem.inter.WriteAscii(20,error,string.len(error))
	end
	mem.inter.WriteBit(1000,0,0)
	status = mail.Status()
	mem.inter.Write(1,status)
	end
	■ 程式說明：
	當$1000.0被觸發時，建立檔名為“posheng.txt”的文件，等待1000 ms，建立完成後，將文件以密碼1234進行加密，接著壓縮成zip格式的檔案，以郵件主旨為“mail subject”、郵件內容為“mail content”的格式傳送給Receiver@yahoo.com.tw。最後，將回傳值的結果寫入$10，若回傳值不等於1(即傳送失敗)，將錯誤碼(指定的檔案路徑不存在、無法打開指定的檔案或是磁碟未備妥等資訊)寫入$20，最後將$1000.0設為off，以及透過mail.Status查看SMTP的連線狀態資訊(如連線SMTP Server是否成功、認證是否成功等)並回傳資料至$1。

<table>
<tr><th colspan="2">範例 (mail)</th></tr>
<tr><td>建立Lua程序</td><td>
■ 於軟體左側專案樹 → [Main]，建立Lua指令(mail.SendAlarm)。
<pre>
if mem.inter.ReadBit(1000,1)==1 then
    result, error = mail.SendAlarm("Receiver@yahoo.com.tw", "mail subject", "mail content", "1234")
    mem.inter.Write(10,result)
    if result ~=1 then
        mem.inter.WriteAscii(20,error,string.len(error))
    end
    status = mail.Status()
    mem.inter.Write(1,status)
    mem.inter.WriteBit(1000,1,0)
end
</pre>
■ 程序說明：
當$1000.1被觸發時，將警報資料(.CSV)檔案以密碼1234進行加密，接著壓縮成zip格式的檔案，以郵件主旨為“mail subject”、郵件內容為“mail content”的格式傳送給Receiver@yahoo.com.tw。接著，將回傳值的結果寫入$10，若回傳值不等於1(代表失敗)，將錯誤碼寫入$20，最後將$1000.1設為off，並透過mail.Status查看SMTP的連線狀態資訊，接著回傳資料至$1。
</td></tr>
<tr><td>建立元件</td><td>
■ 建立三個交替型按鈕，分別設定寫入記憶體位址為$1000.0、$1000.1、$2000.0。
■ 建立兩個數值輸入元件，分別設定寫入記憶體位址為$1、$10。
■ 建立文數值輸入元件，設定寫入記憶體位址為$20。


</td></tr>
</table>

<table>
<tr><th colspan="2">範例 (mail)</th></tr>
<tr><td>設定警報</td><td>■ 於 [選項] → [警報設定]，設定警報訊息並將觸發位址設為$2000.0。
 

註：當$2000.0被觸發時，觸發警報alarm1。</td></tr>
<tr><td>設定SMTP功能</td><td>■ 設定SMTP功能，詳細說明可參考DOPSoft軟體操作手冊CH27。</td></tr>
<tr><td>執行結果</td><td>■ 完成Lua程式的編寫和元件的建立後，請下載專案至人機。
■ 點選Sendfile，即可收到所寄出的檔案。
 
</td></tr>
</table>

範例 (mail)	
執行結果	■ 點選$2000.0來觸發警報，再點選**Sendalarm**，以“mail subject“為主旨送出信件，使用者即可在信箱收到警報的CSV資料。  

4.21 Draw (繪圖功能)

此指令可供使用者對人機進行繪圖的功能，指令包括：

指令	指令運算式	說明
Draw (繪圖功能)	draw.Point	繪圖(點)
	draw.Line	繪圖(線)
	draw.Rect	繪圖(長方形)
	draw.Ellipse	繪圖(橢圓形)
	draw.Clear	清除繪圖
	draw.SetAntialiasing	啟用/取消反鋸齒功能

以下將一一詳細介紹。

■ draw.Point：繪圖(點)

指令名稱	draw.Point
指令運算式	ret = **draw.Point**(x, y, color)
參數定義	x：整數；人機 x 軸座標 y：整數；人機 y 軸座標 color：整數；RGB565，十進位數值；0 ~ 65535
範例	ret = draw.Point(1, 1, 0)
範例說明	於 HMI 的(1, 1)座標繪點，顏色為黑色 RGB565 #0。
回傳值	ret：成功時回傳 1；失敗時回傳 0

■ draw.Line：繪圖(線)

指令名稱	draw.Line
指令運算式	ret = **draw.Line**(x1, y1, x2, y2, color, penWidth)
參數定義	x1：整數；線的起始點 x 座標 y1：整數；線的起始點 y 座標 x2：整數；線的終點 x 座標 y2：整數；線的終點 y 座標 color：整數；RGB565，十進位數值；0 ~ 65535 penWidth：整數；線的寬度
範例	ret = draw.Line(1, 1, 100, 100, 53388, 5)
範例說明	於 HMI 的(1, 1)座標到(100, 100)座標繪線，顏色為桃紅色 RGB565 #53388。
回傳值	ret：成功時回傳 1；失敗時回傳 0

■ draw.Rect：繪圖(長方形)

指令名稱	draw.Rect
指令運算式	ret = **draw.Rect**(x, y, w, h, color)
參數定義	x：整數；長方形中左上的 x 座標 y：整數；長方形中左上的 y 座標 w：整數；長方形的寬度 h：整數；長方形的高度 color：整數；RGB565，十進位數值；0 ~ 65535
範例	ret = draw.Rect(50, 50, 100, 20, 53388)
範例說明	於 HMI 的(50, 50)座標繪長方形，寬度為 100，高度為 20，顏色為桃紅色 RGB565 #53388。
回傳值	ret：成功時回傳 1；失敗時回傳 0

■ draw.Ellipse：繪圖(橢圓形)

指令名稱	draw.Ellipse
指令運算式	ret = **draw.Ellipse**(x, y, w, h, color)
參數定義	x：整數；橢圓中心的 x 座標 y：整數；橢圓中心的 y 座標 w：整數；橢圓形的寬度 h：整數；橢圓形的高度 color：整數；RGB565，十進位數值；0 ~ 65535
範例	ret = draw.Ellipse(50, 50, 100, 100, 53388)
範例說明	於 HMI 的(50, 50)座標繪橢圓形，寬度為 100，高度為 100，顏色為桃紅色 RGB565 #53388。
回傳值	ret：成功時回傳 1；失敗時回傳 0

■ draw.Clear：清除繪圖

指令名稱	draw.Clear
指令運算式	ret = **draw.Clear**()
參數定義	無參數
範例	ret = draw.Clear()
範例說明	清除繪圖。
回傳值	ret：成功時回傳 1；失敗時回傳 0

■ draw.SetAntialiasing：啟用/取消反鋸齒功能

指令名稱	draw.SetAntialiasing
指令運算式	ret = **draw.SetAntialiasing**(flag)
參數定義	flag：1：啟用反鋸齒功能；0：取消反鋸齒功能
範例	ret = draw.SetAntialiasing(1)
範例說明	啟用反鋸齒功能，此功能可以讓繪出的圖更加圓滑。
回傳值	ret：成功時回傳 1；失敗時回傳 0

更新履歷

發行日期	版本	更新章節	更新內容
September, 2021	V2.0 (第二版)	4.2	新增 Internal memory - $指令和說明： (1) mem.inter.ReadDouble (2) mem.inter.WriteDouble 修改 Internal memory - $指令和說明： (1) 內部記憶體由 0 ~ 65535 擴增至 0 ~ 199999
		4.3	新增 Static memory - $M 指令和說明： (1) mem.static.ReadDouble (2) mem.static.WriteDouble
		4.4	新增 External link 指令和說明： (1) link.ReadDouble (2) link.WriteDouble (3) link.CopyArray (4) link.DownloadEthPLC (5) link.WritePasswordPLC (6) link.SetDefaultStationNo (7) link.SetHMIStationNo (8) link.CODESYSAppDownload (9) link.CODESYSAppUpload
		4.5	新增 File 指令和說明： (1) file.DeleteDir
		4.6	新增指令 FileSlot 指令和說明： (1) fileslot.Read (2) fileslot.Write (3) fileslot.ReadValue (4) fileslot.WriteValue (5) fileslot.GetLength (6) fileslot.Remove (7) fileslot.Import (8) fileslot.Export (9) fileslot.SetName (10) fileslot.GetName (11) fileslot.GetID

發行日期	版本	更新章節	更新內容
		4.7	新增 FTP Client 指令和說明： (1) ftpc.Download (2) ftpc.Upload
		4.9	新增 Recipe 指令和說明： (1) recipe.SetEnRcpDouble (2) recipe.GetEnRcpDouble 修改 Recipe 指令和說明： (1) recipe.ChangeRcpNoIndex 範例說明
		4.10	修改 Screen 指令和說明： (1) screen.IsOpened 說明 (2) screen.Open、screen.CloseSub、screen.IsOpened 的畫面 ID 由 1 開始
		4.11	新增 String 指令和說明： (1) string.gmatch (2) string.gsub (3) string.match
		4.12	新增 System library 指令和說明： (1) sys.GetDiskSpace
		4.13	新增 Serial Port communication 指令和說明： (1) com.StationOn (2) com.StationOff (3) com.GetStatus
		4.15	新增 UDP communication 指令和說明： (1) udp.Open (2) udp.Read (3) udp.Write (4) udp.Close (5) udp.GetMaxCount (6) udp.GetRunCount (7) udp.GetStatus
		4.18	新增 Convert 指令和說明： (1) convert.ToNum
		4.19	新增 Account 指令和說明： (1) account.ResetLockStatus (2) account.ChangeUserExpiredDays (3) account.ChangePwdExpiredDays (4) account.GetStatus (5) account.GetLockedList

發行日期	版本	更新章節	更新內容
		4.21	新增 Draw 指令和說明： (1) draw.Point (2) draw.Line (3) draw.Rect (4) draw.Ellipse (5) draw.Clear (6) draw.SetAntialiasing
June, 2021	V1.0 (第一版)		

(此頁有意留為空白)